```
                ;                       ====================
                ;                       X  I  O  S  -  8  6
                ;                       ====================
                ;
                ;                       Concurrent  CP/M-86
                ;                       eXtended I/O System
                ;                            for the
                ;                       IBM Personal Computer
                ;
                ;                       Copyright (c) 1983
                ;                       Digital Research, Inc.
                ;                       Box 579, Pacific Grove
                ;                       California, 93950
                ;
                ;       This XIOS is presented as an example
                ;       hardware interface the CCP/M-86 operating system.
                ;
                ;       Permission is hereby granted to use or
                ;       abstract the following program in  the
                ;       implementation of Concurrent CP/M-86,
                ;       CP/M, MP/M or CP/Net for the 8086 or 8088
                ;       micro-processor.
                ;
                title 'Example CCP/M-86 XIOS'
                ;
                ;***************************************************************************
                ;*                                                                         *
                ;*                          XIOS ORGANIZATION                              *
                ;*                                                                         *
                ;***************************************************************************
                ;
                ;       This XIOS is organized into the several major sections which
                ;       begin with the section name centered and underlined.
                ;       The following Table of Contents lists the major sections,
                ;       and most of the sub-sections.
                ;
                ;       Section Name                                    Page
                ;         Sub Section
                ;
                ;       CCP/M Formats                                   6
                ;         SYSDAT Format                                 6
                ;         CCP/M System Call Equates                     6
                ;         Process Descriptor Format                     7
                ;         Console Control Block Format                  8
                ;         Memory Descriptor Format                      9
                ;
                ;       XIOS Equates                                    11
                ;         XIOS Flag Assignments                         11
                ;         ASCII Codes                                   11
                ;
                ;       XIOS HEADER                                     13
                ;       AND CCP/M INTERFACE
```

```
107
108                     ;       Disk I/O Data                                   80
109
110                     ;       INIT:                                           84
111
112                     ;       Miscellaneous Routines                          93
113                     ;         PRINT_MSG:                                    93
114                     ;         RESET:                                        93
115
116
117                     ;       The XIOS functions invoked by the CCP/M kernel through
118                     ;       the JMP to ENTRY:, are denoted by a row of equal signs
119                     ;       above and below the function label.  For example the IO_CONIN
120                     ;       label has the form:
121
122                     ;       ;========
123                     ;       io_conin:
124                     ;       ;========
125
126                     ;       Internal XIOS functions that are called from various
127                     ;       parts of the XIOS are of the form:
128
129                     ;       ;--------------------
130                     ;       set_physical_cursor:
131                     ;       ;--------------------
132
133                     ;       Interrupt routines are similar to the latter but begin
134                     ;       with the string "i_", for example:
135
136                     ;       ;----------
137                     ;       i_keyboard:
138                     ;       ;----------
139
140                     ;       Subroutines local to a specific function are underlined:
141
142                     ;       action_key:
143                     ;       ;----------
144
145
146                     ;       Register usage for XIOS "io_" functions and INIT:
147
148                     ;       Entry:  AL = function # (at ENTRY: label)
149                     ;               CX = entry parameter
150                     ;               DX = entry parameter
151                     ;               DS = SYSDAT (at ENTRY: and INIT:)
152                     ;               ES = User Data Area (UDA) of currently running process
153
154                     ;       Exit:   AX = return
155                     ;               BX = AX (in exit)
156                     ;               ALL SEGMENT REGISTERS PRESERVED:
157                     ;               CS,DS,ES,SS must be preserved though call
158
159                     ;       Far calls to the Supervisor from the XIOS:
```

```
                        ;               CL = function
                        ;               CH = 0
                        ;               DX = parameter
                        ;               DS = data segment of parameter if DX is an address
                        ;               ES = UDA of currently running process
                        ;
                        ;       Far jumps to the Dispatcher made from XIOS interrupt routines:
                        ;               All registers the same as on entry to the interrupt
                        ;               routine, except CS and IP which are changed by the JMPF.
                        ;               (the FLAGS, CS, IP are on the stack
                        ;               by virtue of the INT instruction)
                        ;
                        ;       DEV_SETFLAG calls to the Supervisor from XIOS interrupt routines:
                        ;               CL = DEV_SETFLAG function number
                        ;               CH = 0
                        ;               DX = flag number to set
                        ;               DS = SYSDAT
                        ;               ES = don't care (must be restored at end of the
                        ;                       interrupt service routine, however)
                        ;
                        ;       Notes: The DEV_SETFLAG call and the jump to the dispatcher are
                        ;               the only legal operating system "entry" points
                        ;               for interrupt routines.
                        ;
                        ;               Changes have been made in the
                        ;               register conventions from
                        ;               the CP/M-86 BIOS and the MP/M-86 XIOS.
                        ;
                        ;       The XIOS must be assembled using the following memory model:
                        ;
                        ;       8080 model:
                        ;       -----------
                        ;       The code and data segments are the same,
                        ;       mixed code and data.  The code segment
                        ;       is org'd at 0C00H relative to the system
                        ;       data area.
                        ;
                        ;       high    +-----------------+\
                        ;               |  system tables  | |
                        ;               +-----------------+ |
                        ;               |  XIOS (C and D) |  >  system data
                        ;               +-----------------+ |
                        ;               |  SYSDAT         | |
                        ;               +-----------------+x
                        ;               |  system code    | >   system code
                        ;       low     +-----------------+/
                        ;
                        ;       Notes on coding style:
                        ;
                        ;       Indentation of the code is used to emphasize branching
```

```
                        ;       after conditional jumps.  Lower case is used for all code
                        ;       and variables for easier reading.  Code labels, equates
                        ;       and variable names are in upper case within comments to
                        ;       distinguish them from the english.
                        ;
                        ;       Equates and data variables are placed in proximity to where
                        ;       they are used.  Having the code and data intermixed requires
                        ;       the use of equates and ORG statements throughout the XIOS.
                        ;
                        ;       The XIOS may be broken up into smaller files with the use
                        ;       of the ASM86 "include" statement.  All of the data variables
                        ;       may be placed in one separate data segment and the extra
                        ;       equates and ORGs removed.
```

```
                        eject
                        ;                       CCP/M Formats
                        ;                       -------------

                        ;*****************************************************************
                        ;*                                                               *
                        ;*                       SYSDAT Format                           *
                        ;*                                                               *
                        ;*****************************************************************

                        DSEG
                        ORG     0
0000                    supmod          rw      2       ;internal entry point to SUP
                                org     038h
0038                    dispatcher      rw      2       ;interrupt routines may exit with
                                org     040h            ;a JMPF here
0040                    osseg           rw      1       ;beginning paragraph of O.S.
                                org     044h
0044                    endseg          rw      1       ;1st paragraph after O.S.
                                org     04Bh
0048                    sys_disk        rb      1       ;system disk number
                                org     050h
0050                    temp_disk       rb      1       ;temporary disk number
                                org     058h
0058                    mdul            rw      1       ;root of unused memory descriptors
005A                    mfl             rw      1       ;root of memory free list
                                org     068h
0068                    rlr             rw      1       ;Ready List Root, 1st PD is
                                org     072h            ;running process
007C                    thrdrt          rw      1       ;Thread Root, list of active
0074                    qlr             rw      1       ;processes
                                org     078h
0078                    version         rw      1       ;address of version string
                                                        ;relative to SUP code segment
007A                    bvernum         rw      1       ;BDOS version number
007C                    osvernum        rw      1       ;O.S. version number
007E                    tod_day         rw      1       ;binary days since January 1, 1978
0080                    tod_hour        rb      1       ;BCD
0081                    tod_min         rb      1       ;BCD
0082                    tod_sec         rb      1       ;BCD
                                org     088h
0088                    open_vec        rw      1       ;16 bit vector of drives with
                                                        ;open files - used by status line
                                                        ;routine

                        ;*****************************************************************
                        ;*                                                               *
                        ;*      CCP/M System Calls Used by this XIOS                     *
                        ;*                                                               *
                        ;*****************************************************************

                                                        ;process control functions:
```

```
                        p_delay         equ     141             ;delay specified number of ticks
                        p_dispatch      equ     142             ;let other another process run
                        p_pdadr         equ     156             ;get double word pointer of process
                                                                ;descriptor
                        p_term          equ     143             ;terminate process

                                                                ;device control functions:
                        dev_poll        equ     131             ;poll device
                        dev_waitflag    equ     132             ;wait for flag to be set
                        dev_setflag     equ     133             ;set a specified flag

                                                                ;Time function:
                        t_seconds       equ     155             ;get date,hours,minutes,seconds


                        ;************************************************************************
                        ;*                                                                      *
                        ;*              Process Descriptor Format                               *
                        ;*                                                                      *
                        ;************************************************************************

                        ;       The Process Descriptor (PD) along with the
                        ;       associated User Data Area (UDA), describe
                        ;       the current state of a Process under CCP/M-86.
                        ;       The process descriptor is always within the System
                        ;       Data Segment.

                        ;       Process Descriptor:

                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+
                        ; 00|   link    |  thread   |stat |prior|   flag    |
                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+
                        ; 08|                       name                    |
                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+
                        ; 10|    uda    |disk | user|       reserved        |
                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+
                        ; 18|                 resereved           |  parent   |
                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+
                        ; 20|cons |     reserved    |list |    reserved     |
                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+
                        ; 28|                     reserved                    |
                        ;   +-----+-----+-----+-----+-----+-----+-----+-----+

                        ;       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 address of user data area
                        ;       disk    - current default disk
                        ;       user    - current default user number
```

```
                        ;       mem     - pointer to MD list of memory owned
                        ;                 by this process
                        ;       parent  - process that created this process
                        ;       cons    - default console device (doesn't imply ownership)
                        ;       list    - default list device (doesn't imply ownership)

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 + 8
0012                    p_disk          equ     byte ptr p_uda + word
0013                    p_user          equ     byte ptr p_disk + byte
0016                    p_mem           equ     word ptr p_user + 3
001E                    p_parent        equ     word ptr p_mem + 8
0020                    p_cons          equ     byte ptr p_parent + word
0024                    p_list          equ     byte ptr p_cons + 4

0030                    pd_len          equ     30H

                        ;*****************************************************************************
                        ;*                                                                           *
                        ;*              Console Control Block Format                                 *
                        ;*                                                                           *
                        ;*****************************************************************************
                        ;*
                        ;*      The CCB is used by the system to control the virtual
                        ;*      consoles.  There must be one correctly initialiazed CCB
                        ;*      in the XIOS per virtual console.
                        ;*
                        ;*      +---------+---------+---------+---------+
                        ;*  00  |      owner        |      reserved     |
                        ;*      +---------+---------+---------+---------+
                        ;*  04  |               reserved                  |
                        ;*      +---------+---------+---------+---------+
                        ;*  08  |  mimic  |      reserved                 |
                        ;*      +---------+---------+---------+---------+
                        ;*  0C  |      reserved     |       state       |
                        ;*      +---------+---------+---------+---------+
                        ;*  10  |     maxbufsiz     |     reserved      |
                        ;*      +---------+---------+---------+---------+
                        ;*  14  |               reserved                  |
                        ;*      +---------+---------+---------+---------+
                        ;*  18  |               reserved                  |
                        ;*      +---------+---------+---------+---------+
                        ;*  1C  |               reserved                  |
                        ;*      +---------+---------+---------+---------+
                        ;*  20  |               reserved                  |
                        ;*      +---------+---------+---------+---------+
                        ;*  24  |               reserved                  |
```

```
                        ;*      +---------+---------+---------+---------+
                        ;*  28  |                 reserved                |
                        ;*      +---------+---------+---------+---------+
                        ;*
                        ;*      owner   - current owner of device
                        ;*                if 0, no owner
                        ;*      mimic   - list dev that mimics us.
                        ;*                0ffh means no mimic device
                        ;*      state   - current state of virtual console
                        ;*      maxbufsiz - maximum file size for buffered mode
                        
0000                    c_owner         equ     word ptr 00h
0008                    c_mimic         equu    byte ptr 08h
000E                    c_state         equ     word ptr 0Eh
0010                    c_maxbufsiz     equ     word ptr 10h
002C                    ccblen          equ     2ch
                        
                        ;CCB state flags
                        
0001                    csm_buffered    equ     00001h
0002                    csm_background  equ     00002h
0004                    csm_purging     equ     00004h
0008                    csm_noswitch    equ     00008h
0080                    csm_ctrlS       equ     00080h
0100                    csm_ctrlQ       equ     00100h
0200                    csm_ctrlP       equ     00200h
                        
                        ;*****************************************************************************
                        ;*                                                                           *
                        ;*              Memory Descriptor Format                                     *
                        ;*                                                                           *
                        ;*****************************************************************************
                        
                        ;*      GENCCPM creates an MD per memory partition specified,
                        ;*      links them together and roots the list at the Memory Free
                        ;*      list location in SYSDAT.
                        ;*      The XIOS in the INIT: routine can trim this list to
                        ;*      be in the bounds of actual memory
                        
                        ;*      +---------+---------+---------+---------+
                        ;*  04  |       link        |       start       |
                        ;*      +---------+---------+---------+---------+
                        ;*  08  |      length       |      reserved     |
                        ;*      +---------+---------+---------+---------+
                        ;*  0C  |      reserved     |
                        ;*      +-------------------+
                        
                        ;*      link    - offset of next MD, 0 if end of list
                        ;*      start   - beginning paragraph of free memory
                        ;*      length  - size of partition in paragraphs
                        
0000                    md_link         equ     word ptr 0
```

```
440
441     0002                md_start        equ     word ptr md_link + word
442     0004                md_length       equ     word ptr md_start + word
443     0008                mdlen           equ     md_length + 4
444
```

```
                        eject
                        ;                       XIOS Equates
                        ;                       ---- -------

FFFF                    true            equ     0ffffh
0000                    false           equ     0

                        ;***************************************************************************
                        ;*                                                                         *
                        ;*                       XIOS Flag Assignments                             *
                        ;*                                                                         *
                        ;***************************************************************************

                        ;       Definition of flag table used by by CCP/M and this XIOS

                                                                ;CCP/M flags:
                                                                ;0 - is reserved
0001                    tick_flag       equ     1               ;tick flag number
0002                    sec_flag        equ     2               ;seconds flag number
0003                    min_flag        equ     3               ;minutes flag number

                                                                ;XIOS flags:
0004                    fdc_flag        equ     4               ;disk operation flag
0005                    key_flag        equ     5               ;key available flag
0006                    bell_flag       equ     6
0006                    last_flag       equ     bell_flag

                        ;***************************************************************************
                        ;*                                                                         *
                        ;*                       ASCII Codes                                       *
                        ;*                                                                         *
                        ;***************************************************************************

0000                    nul     equ     00h
0001                    soh     equ     01h
0002                    stx     equ     02h
0003                    etx     equ     03h
0004                    eot     equ     04h
0005                    enq     equ     05h
0006                    ack     equ     06h
0007                    bel     equ     07h
0008                    bs      equ     08h
0009                    ht      equ     09h
000A                    lf      equ     0ah
000B                    vt      equ     0bh
000C                    ff      equ     0ch
000D                    cr      equ     0dh
000E                    so      equ     0eh
000F                    shi     equ     0fh
0010                    dle     equ     10h
0011                    dc1     equ     11h
0012                    dc2     equ     12h
```

```
498
499     0013            dc3     equ     13h
500     0014            dc4     equ     14h
501     0015            nak     equ     15h
502     0016            syn     equ     16h
503     0017            etb     equ     17h
504     0018            can     equ     18h
505     0019            em      equ     19h
506     001A            subb    equ     1ah
507     001B            esc     equ     1bh
508     001C            fs      equ     1ch
509     001D            gs      equ     1dh
510     001E            rds     equ     1eh
511     001F            us      equ     1fh
512     007F            del     equ     7fh
513
514     0011            xon     equ     dc1
515     0013            xoff    equ     dc3
516
```

```
                        eject
                        ;               XIOS HEADER AND CCP/M INTERFACE
                        ;               ------------------------------

                        ;*****************************************************************
                        ;*                                                               *
                        ;*                        XIOS Header                            *
                        ;*                                                               *
                        ;*****************************************************************

                        CSEG
                        ORG     0C00h                           ;8080 model XIOS base
                                                                ;CCP/M-86 entry points to XIOS:
0C00 E94413     1F47            jmp     init                    ;called once at system boot
0C03 E93700     0C30            jmp     entry                   ;subsequent calls go through here

  0004                  num_vir_cons    equ     4               ;number of virtual consoles

0C06 0000               sysdat          dw      0
0C08                    supervisor_o    rw      1
0C0A                    supervisor_s    rw      1
  0C08                  supervisor      equ     dword ptr supervisor_o

  0C0C                  SL1     equ     offset $
                        DSEG
                        ORG     SL1

  003C                  ticks_per_second        equ     60

0C0C 00                 tick            db      false           ;set by I_TICK
0C0D 3C                 ticks_sec       db      ticks_per_second
0C0E 00                 door            db      false           ;used if door open interrupt
                                                                ;is available to XIOS
0C0F                                    rb      2
0C11 04                 nvcns           db      num_vir_cons    ;4 virtual consoles
0C12 04                 nccb            db      4               ;total number of CCBs
0C13 02                 nlcb            db      2               ;2 list devices

0C14 3818               ccb             dw      offset ccb_tab  ;pointer to the first CCB
0C16 7D19               lcb             dw      offset lcb_tab  ;pointer to the first LCB

                                                                ;disk parameter header offsets
0C18 E91E               dph_tbl         dw      offset dphA     ;A
0C1A F01E                               dw      offset dphB     ;B
0C1C 000000000000                       dw      0,0,0,0,0,0,0   ;C-I
     000000000000
     0000
0C2A 000000000000                       dw      0,0,0           ;J-L
0C30 111F00000000       dph_m_adr       dw      dphM,0,0,0      ;M-P
     0000

                        ;       If Mdisk memory is not present, INIT: 0's DPH_M_ADR.
```

```
                        ;       We put this DPBM in now so GENCCPM will perform
                        ;       DPH fixups for us.
                        
0C38 C003               genccpm_buf     dw      screens_size    ;GENCCPM will alloc screens_size
                                                                ;# of paragraphs and put the
                                                                ;segment address of the buffer in
                                                                ;the variable GENCCPM_BUF
                        
0C3A 9515               trans_table     dw      key_table       ;allow a transient to change
                                                                ;the keyboard mapping
0C3C 00                 debug           db      false           ;allow CP/M-86 to be used
                                                                ;for debugging, see INIT:
                        
                        ;***************************************************************************
                        ;*                                                                         *
                        ;*                          CCP/M INTERFACE                                *
                        ;*                                                                         *
                        ;***************************************************************************
                        
  0C3D                  SL2     equ     offset $
                        CSEG
                        ORG     SL2
                        
                        ;=====
                        ;=====
                        entry:                                  ;arrive here from JMP at
                        ;=====                                  ;03H in XIOS code segment
                        ;=====
                        
                        ;       entry:  AL = function number
                        ;               CX, DX parameters
                        ;       exit:   AX = BX = return
                        ;               ALL SEGMENT REGISTERS PRESERVED:
                        ;               CS,DS,ES,SS must be preserved though call
                        
                        ;       Note: no alteration of stack is allowed during entry except
                        ;       for the return address caused by the "CALL FUNCTION_TABLE[BX]"
                        ;       instruction.
                        
0C3D FC                         cld                             ;set the direction flag
0C3E 32E4                       xor ah,ah
0C40 3C0E                       cmp al,xios_funcs
0C42 730A               0C4E    jae ill_func
0C44 D0E0                         shl al,1                      ;multiply by 2
0C46 8BD8                         mov bx,ax                     ;put in pointer register
0C48 FF974F0C                     call function_table[bx]       ;no range checking needed
0C4C 8BD8                         mov bx,ax                     ;only called by O.S.
                        
                        ill_func:
0C4E CB                         retf                            ;return to O.S. kernel
```

```
                        SL3     equ     offset $
                        DSEG
                        ORG     SL3

                xios_funcs      equ     14

                        function_table:
0C4F E30F                       dw      io_const                ; 0 console status
0C51 E60F                       dw      io_conin                ; 1 console input
0C53 F810                       dw      io_conout               ; 2 console output
0C55 3E16                       dw      io_liststs              ; 3 list status
0C57 5616                       dw      io_list                 ; 4 list output
0C59 7116                       dw      io_auxin                ; 5 auxillary input
0C5B 7216                       dw      io_auxout               ; 6 auxillary out
0C5D 7214                       dw      io_switch               ; 7 switch screen
0C5F E214                       dw      io_statline             ; 8 update or print new status
0C61 9119                       dw      io_seldsk               ; 9 select disk
0C63 A119                       dw      io_read                 ;10 read logical sector
0C65 F119                       dw      io_write                ;11 write logical sector
0C67 841C                       dw      io_flushbuf             ;12 flush buffers
0C69 790C                       dw      io_poll                 ;13 poll device

  0C6B                  SL4     equ     offset $
                        CSEG
                        ORG     SL4

                        ;-------
                        setflag:
                        ;-------
                        ;       entry:  DL = flag number
                        ;       exit:   AX = BX = return

0C6B B185                       mov cl,dev_setflag
0C6D EB02       0C71            jmps supif

                        ;--------
                        waitflag:
                        ;--------
                        ;       entry:  DL = flag number
                        ;       exit:   AX = BX = return

0C6F B184                       mov cl,dev_waitflag
                                ;jmps supif

                        ;-----
                        supif:
                        ;-----
                        ;       entry:  CL = function number
                        ;               CH = 0
                        ;               DX parameter
                        ;               ES = user data area
                        ;       exit:   AX = BX = return
```

```
                        ;               CX = error code from O.S.
                        ;               ES = UDA or return value

0C71 32ED                       xor ch,ch                       ;ensure CH is 0
0C73 2EFF1E080C                 callf supervisor
0C78 C3                         ret

                        ;=======
                        io_poll:
                        ;=======
                        ;       entry:  DL =    device #
                        ;       exit:   AL =    0 if not ready, 0FFH if ready
                        ;               BX =    device# * 2.
                        ;               ALL SEGMENT REGISTERS PRESERVED:
                        ;               CS,DS,ES,SS must be preserved though call

                        ;       IO_POLL is called from the dispatcher after a process makes
                        ;       a DEV_POLL call to the O.S.

0C79 33DB                       xor bx,bx
0C7B 8ADA                       mov bl,dl
0C7D D1E3                       shl bx,1
0C7F FFA7930C                   jmp poll_table[bx]

                        ;----
                        poll:                                   ;poll device
                        ;----
                        ;       entry:  DL = device number
                        ;       exit:   AL = 0ffh device ready
                        ;               AL = 0 if not ready
                        ;               BX = device# * 2

0C83 E8F3FF     0C79            call io_poll                    ;check hardware first
0C86 85C07401   0C8B            test ax,ax ! jz p_os            ;AX=0 if not ready
0C8A C3                           ret                           ;BX=device# * 2 set by io_poll
                        p_os:                                   ;DL=device#
0C8B 53                         push bx                         ;save device# * 2
0C8C B183                       mov cl,dev_poll                 ;give up the CPU resource
0C8E E8E0FF     0C71            call supif
0C91 5B                         pop bx                          ;device# * 2
0C92 C3                         ret

  0C93                  SL5     equ     offset $
                        DSEG
                        ORG     SL5

                        poll_table      dw      listst          ;each of these functions must
0C93 4416                               dw      listst          ;preserve BX and DX
0C95 4416                               dw      0,0,0,0         ;replace with other poll device
0C97 000000000000
     0000                                                       ;routines address, when needed.
```

```
                        eject
                        ;                       Interrupt Handlers
                        ;                       ------------------

                        ;*****************************************************************
                        ;*                                                               *
                        ;*                      IBM PC Interrupt Structure               *
                        ;*                                                               *
                        ;*****************************************************************

0001                    s_step_int              equ     01h             ;interrupt numbers
0002                    nmi_int                 equ     02h
0003                    one_byte_int            equ     03h
0008                    tick_int                equ     08h
0009                    key_int                 equ     09h
000E                    disk_int                equ     0Eh
0011                    equip_int               equ     11h
0012                    mem_size_int            equ     12h
0014                    rom_rs232_int           equ     14h             ;see init for debugging use
0017                    rom_printer_int         equ     17h             ;       "
00E0                    os_int                  equ     224             ;normal CCP/M-86 entry
00E1                    debug_int               equ     225             ;debugger entry to O.S.

0004                    iv_s_step               equ     s_step_int*4    ;interrupt vector offsets
0008                    iv_nmi                  equ     nmi_int*4
000C                    iv_one_byte             equ     one_byte_int*4
0020                    iv_tick                 equ     tick_int*4
0024                    iv_key                  equ     key_int*4
0038                    iv_disk                 equ     disk_int*4
0050                    iv_rs232                equ     rom_rs232_int*4
0384                    iv_debug                equ     debug_int*4

                        ;*****************************************************************
                        ;*                                                               *
                        ;*              IBM PC Keyboard Ports                            *
                        ;*                                                               *
                        ;*****************************************************************

0060                    kbd_data        equ     060h            ;input port for the key board data
0061                    kbd_control     equ     061h            ;control port for the key board

                        ;*****************************************************************
                        ;*                                                               *
                        ;*      8259 Programmable Interrupt Controller Commands          *
                        ;*                      and Ports                                *
                        ;*                                                               *
                        ;*****************************************************************

0020                    pic_even_port           equ     020h    ;port 0
0021                    pic_odd_port            equ     021h    ;bit 0 is A0 of 8259 instructions

0020                    pic_nseoi               equ     020h    ;non specific end of interupt
```

```
                                ;*************************************************************
                                ;*                                                           *
                                ;*            8253-5 Counter Timer Port and Data Equates     *
                                ;*                                                           *
                                ;*************************************************************

0006                    disk_channel            equ     06h
0001                    keyboard_channel        equ     01h
0000                    timer_channel           equ     00h

0040                    timer_0_reg             equ     040h            ;first counter timer port
0041                    timer_1_reg             equ     041h
0042                    timer_2_reg             equ     042h
0043                    timer_cmd_reg           equ     043h            ;command port

4DAE                    timer_60_hz             equ     19886           ;constant for 60.001 hz tick
0533                    timer_1000_hz           equ     533h            ;constant for 1000 hz tone

00B6                    bell_cmd                equ     10110110b       ;timer 2 lsb,msb binary
0003                    bell_on                 equ     03h
00FC                    bell_off                equ     0fch

0060                    port_a                  equ     060h
0061                    port_b                  equ     061h
0062                    port_c                  equ     062h

                        ;*************************************************************
                        ;*                                                           *
                        ;*                          I_TICK                           *
                        ;*                                                           *
                        ;*************************************************************

                        ;       The following routine gets control on a timer interrupt,
                        ;       from the 8253-5.

0C9F                    SL6     equ     offset $
                        CSEG
                        ORG     SL6

                        ;------
                        i_tick:
                        ;------
                        ;       Entry:  Currently running process's register state
                        ;       Exit:   All registers preserved

0C9F 1E                         push ds                         ;use one level of user stack
0CA0 2E8E1E060C                 mov ds,sysdat
0CA5 803E840D00                 cmp tcnt,0                      ;keep entry count to allow
0CAA 7510               0CBC    jnz t_noswitch                  ;debugging under DDT86
0CAC 8C168800D                    mov i_tick_ss,ss
```

```
0834
0835  0CB0 8925320D                    mov i_tick_sp,sp
0836  0CB4 2E8E16060C                  mov ss,sysdat
0837  0CB9 BCB00D                      mov sp,offset i_tick_stack
0838                            t_noswitch:
0839  0CBC FE06840D                    inc t_cnt
0840
0841  0CC0 FE06540D                    inc int_cnt                    ;interrupt counter
0842  0CC4 FB                          sti
0843
0844  0CC5 505351                      push ax ! push bx ! push cx
0845  0CC8 525557                      push dx ! push bp ! push di
0846  0CCB 5606                        push si ! push es
0847
0848                            ;       Bells on the IBM PC must turn on a speaker for a given duration.
0849                            ;       A "bell cycle" must have an off period so we can distinguish
0850                            ;       it from the next bell generated.  A process sending a bell
0851                            ;       to IO_CONOUT, waits for a DEV_SETFLAG operation from the I_TICK
0852                            ;       routine that signals the end of the bell.
0853                            ;
0854                            ;       Handling bells sent to background processes presents some
0855                            ;       problems.  If we generate bells sent to a background
0856                            ;       console, the bells would not pertain to the displayed image.
0857                            ;       If we keep a bell count per console, and produce the stacked up
0858                            ;       bells when we switch in a background virtual console
0859                            ;       the bells would be out of "sync" with the
0860                            ;       the displayed image.  Stopping a background process because
0861                            ;       it produced a bell would defeat the purpose of virtual screens.
0862                            ;       Therefore, bells sent to background consoles are ignored.
0863                            ;
0864                            ;       Note: the screen structures are defined in the SERIAL IO section.
0865
0866  0CCD 803E520D00                  cmp bell_ticks,0               ;in bell cycle if non 0
0867  0CD2 7418          0CEF            jz check_motor_off           ;BELL_TICKS is set by IO_CONOUT
0868  0CD4 FE0E520D                    dec bell_ticks
0869  0CD8 7508          0CE2          jnz check_bell_off
0870  0CDA BA0600                        mov dx,bell_flag
0871  0CDD E888FF        0C68            call setflag                 ;wake up process that generated
0872  0CE0 EB0D          0CEF            jmps check_motor_off         ;the bell, ES doesn't need to
0873                            check_bell_off:                      ;be the UDA on DEV_SETFLAG calls
0874  0CE2 803E520D01                    cmp bell_ticks,1             ;turn bell on last tick of cycle
0875  0CE7 7505          0CEF            jne check_motor_off
0876  0CE9 E461                            in al,port_b
0877  0CEB 24FC                            and al,bell_off
0878  0CED E661                            out port_b,al
0879
0880                            check_motor_off:
0881  0CEF 803EA71CFF                  cmp disk_busy,true             ;leave motor on while disk
0882  0CF4 7416          0D0C          je check_seconds               ;operation is underway ...
0883                                                                  ;helpful for debuging disk routines
0884  0CF6 803EA81C00                    cmp motor_off_counter,0      ;motor is off if 0
0885  0CFB 740F          0D0C            je check_seconds             ;motor on routine sets counter to
0886  0CFD FE0EA81C                        dec motor_off_counter      ;0ffh
```

```
                                jnz check_seconds           ;not timed out yet
0D03 8BF203                       mov dx,fdc_port
0D06 B00C                         mov al,fdc_on             ;do not reset the FDC
0D08 A2A61C                       mov motor_flags,al        ;deselect the motors
0D0B EE                           out dx,al                 ;turn it off
                                                            ;only one can be on at a time
                            check_seconds:
0D0C FE0E530D                       dec tick_counter        ;has 1 second elapsed ?
0D10 7508               0D1D        jnz no_second_flag
0D12 C606530D3C                   mov tick_counter,ticks_per_second     ;yes - set
0D17 BA0200                       mov dx,sec_flag           ;the second flag
0D1A E84EFF             0C68      call setflag              ;DS=SYSDAT but ES does not
                                                            ;need to be UDA on DEV_SETFLAG
                            no_second_flag:
0D1D 803E0C0C00                     cmp tick,false          ;only set the tick flag
0D22 7406               0D2A        je i_tick_exit          ;when the tick variable in
0D24 BA0100                       mov dx,tick_flag          ;the XIOS header is true
0D27 E841FF             0C68      call setflag              ;DS must = SYSDAT out
                                                            ;ES does not need = UDA
                                                            ;on DEV_SETFLAG
                            i_tick_exit:
0D2A 075E5F                         pop es ! pop si ! pop di
0D2D 5D5A59                         pop bp ! pop dx ! pop cx
0D30 5B                             pop bx                  ;AX still on stack

0D31 FA                             cli                     ;stop reentrancy

0D32 B020                           mov al,pic_nseoi        ;signal end of interrupt
0D34 E620                           out pic_even_port,al    ;to 8259

0D36 58                             pop ax

0D37 FE0EB40D                       dec t_cnt
0D3B 7508               0D45        jnz no_stack_restore
0D3D 8E16B00D                     mov ss,i_tick_ss
0D41 8B26B20D                     mov sp,i_tick_sp

                            no_stack_restore:
0D45 FE0E540D                       dec int_cnt
0D49 1F                             pop ds                  ;restore DS of interrupted process
0D4A 7401               0D4D        jz it_disp
0D4C CF                           iret
                            it_disp:
0D4D 2EFF2E3800                     jmpf cs:dword ptr dispatcher    ;go run the next ready process

                            ;*****************************************************************
                            ;*                                                               *
                            ;*                          I_TICK Data                          *
                            ;*                                                               *
                            ;*****************************************************************

  0D52                      SL7       equ     offset $
```

```
                        DSEG
                        ORG SL7

0D52 00                 bell_ticks      db      0
0D53 3C                 tick_counter    db      ticks_per_second        ;counter for tick interupts
0D54 00                 int_cnt         db      0

                        org (offset $ + 1) and 0fffeh

0D56 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D5C CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D62 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D68 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D6E CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH

0D74 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D7A CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D80 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D86 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D8C CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH

0D92 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D98 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0D9E CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0DA4 CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH
0DAA CCCCCCCCCCCC                       dw      0CCCCh,0CCCCH,0CCCCH

0DB0                    i_tick_stack    rw      0

0DB0                    i_tick_ss       rw      1
0DB2                    i_tick_sp       rw      1
0DB4 00                 t_cnt           db      0

                        ;*****************************************************************************
                        ;*                                                                           *
                        ;*                          I_KEYBOARD                                       *
                        ;*                                                                           *
                        ;*****************************************************************************

                        ;       The following routine gets control after a keyboard interrupt.
                        ;       Interrupts are generated on the IBM PC when a key is depressed
                        ;       or released.

0DB5                    SL8     equ     offset $
                        CSEG
                        ORG     SL8

                        ;----------
                        i_keyboard:
                        ;----------
                        ;       Entry:  Currently running process's register state
                        ;       Exit:   All registers preserved
```

```
 993
 994
 995   0DB5 1E2E8E1E060C             push ds ! mov ds,sysdat          ;use one level of user stack
 996   0DBb 8C16960E                 mov i_keyboard_ss,ss             ;use data segment, keep
 997   0DBF 8926980E                 mov i_keyboard_sp,sp             ;XIOS code and data separate
 998   0DC3 2E8E16060C               mov ss,sysdat
 999   0DC8 BC960E                   mov sp,offset i_keyboard_stack   ;keyboard interrupt stack
1000
1001   0DCB FE06540D                 inc int_cnt                      ;keep other processes from
1002   0DCF FB                       sti                              ;running
1003
1004   0DD0 505351                   push ax ! push bx ! push cx
1005   0DD3 525557                   push dx ! push bp ! push di
1006   0DD6 5606                     push si ! push es
1007
1008   0DD8 E460                     in al,kbd_data                   ;get the scan code
1009   0DDA 8AD0                     mov dl,al
1010
1011   0DDC A0BE0E                   mov al,kb_cnt                    ;# chars in buffer
1012   0DDF 3C20                     cmp al,kblen                     ;if it is full throw away
1013   0DE1 7419           0DFC      je ik_full                       ;scan code, PTM rings the
1014                                                                  ;bell when VINQ gets full.
1015   0DE3 FE06BE0E                   inc kb_cnt                     ;we have one more char in buf
1016   0DE7 A1BC0E                     mov ax,kb_in                   ;ptr to next char
1017   0DEA 8BD8                       mov bx,ax
1018   0DEC 8817                       mov byte ptr [bx],dl           ;put the scan code in buffer
1019   0DEE 43                         inc bx                         ;next free byte in buffer
1020   0DEF 81FBBA0E                   cmp bx,kblen+offset kb_buf     ;wrap around ?
1021   0DF3 7503           0DF8        jne ik_ptr_ok
1022   0DF5 BB9A0E                       mov bx,offset kb_buf         ;back to beginning of buffer
1023                              ik_ptr_ok:
1024   0DF8 891EBC0E                   mov kb_in,bx                   ;save new pointer
1025
1026                              ik_full:
1027   0DFC 803E8F0E00               cmp kb_wait,false                ;only call DEV_SETFLAG when
1028   0E01 7408           0E0E      je ik_no_flag                    ;a process is waiting for
1029   0E03 BA0500                     mov dx,key_flag                ;keyboard input
1030   0E06 E862FE         0C68        call setflag                   ;process doesn't wake up
1031   0E09 C6068F0E00                 mov kb_wait,false              ;until dispatch occurs
1032
1033                              ik_no_flag:
1034   0E0E E461                     in al,kbd_control                ;get the current control port
1035   0E10 8AE0                     mov ah,al                        ;save the current state
1036   0E12 0C80                     or al,080h                       ;reset the key board
1037
1038   0E14 075E5F                   pop es ! pop si ! pop di
1039   0E17 5D5A59                   pop bp ! pop dx ! pop cx
1040   0E1A 58                       pop bx                           ;AX still on stack
1041
1042   0E1B E661                     out kbd_control,al               ;send to control port
1043   0E1D 8AC4                     mov al,ah
1044   0E1F E661                     out kbd_control,al               ;set keyboard back to normal state
1045
```

```
1046
1047   0E21 FA                          cli                             ;keep this interrupt handler
1048                                                                    ;from being reentered
1049   0E22 B020                        mov al,pic_nseoi                ;signal end of interrupt
1050   0E24 E620                        out pic_even_port,al            ;to 8259
1051
1052   0E26 58                          pop ax
1053
1054   0E27 8E16960E                    mov ss,i_keyboard_ss
1055   0E2B 8B26980E                    mov sp,i_keyboard_sp
1056
1057   0E2F FE0E5400                    dec int_cnt
1058   0E33 1F                          pop ds
1059   0E34 7401          0E37          jz ik_disp
1060   0E36 CF                            iret
1061                              ik_disp:
1062   0E37 2EFF2E3800                  jmpf cs:dword ptr dispatcher
1063
1064                                    ;*************************************************************************
1065                                    ;*                                                                       *
1066                                    ;*                          i_KEYBOARD Data                              *
1067                                    ;*                                                                       *
1068                                    ;*************************************************************************
1069
1070     0E3C                           SL9     equ     offset $
1071                                    DSEG
1072                                    ORG     SL9
1073
1074                                    ;       Keyboard interrupt routine stack
1075
1076                                    org     (offset $ + 1) and 0fffeh
1077
1078   0E3C CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1079   0E42 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1080   0E48 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1081   0E4E CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1082   0E54 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1083
1084   0E5A CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1085   0E60 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1086   0E66 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1087   0E6C CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1088   0E72 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1089
1090   0E78 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1091   0E7E CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1092   0E84 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1093   0E8A CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1094   0E90 CCCCCCCCCCCC                        dw      0CCCCH,0CCCCH,0CCCCH
1095
1096   0E96                             i_keyboard_stack        rw      0
1097
1098   0E96 0000                        i_keyboard_ss   dw      0               ;keyboard interrupt
```

```
1099
1100  0E98 0000              i_keyboard_sp   dw      0               ;handler
1101
1102                         ;       Keyboard input buffer, filled by I_KEYBOARD:, read by
1103                         ;       IO_CONIN:
1104
1105    0020                 kblen           equ     32              ;length of character buffer
1106
1107  0E9A                   kb_buf          rb      kblen           ;char buffer
1108  0EBA 9A0E              kb_out          dw      kb_buf          ;pts to next char to read
1109  0EBC 9A0E              kb_in           dw      kb_buf          ;pts to next free char buffer
1110  0EBE 00                kb_cnt          db      0               ;# of chars in buffer
1111  0EBF 00                kb_wait         db      false           ;is any process waiting for
1112                                                                 ;keyboard input
1113
1114                         ;*****************************************************************
1115                         ;*                                                               *
1116                         ;*                       I_DISK                                  *
1117                         ;*                                                               *
1118                         ;*****************************************************************
1119
1120                         ;       The following routine gets control from an interrupt from the
1121                         ;       FDC.  The current state is saved and DEV_SETFLAG is called.
1122
1123    0EC0                 SL10    equ     offset $
1124                         CSEG
1125                         ORG     SL10
1126
1127                         ;------
1128                         i_disk:
1129                         ;------
1130                         ;       Entry:  Currently running process's register state
1131                         ;       Exit:   All registers preserved
1132
1133  0EC0 1E2E8E1E060C              push ds ! mov ds,sysdat
1134  0EC6 8C16660F                  mov disk_ss,ss                  ;save the stack segment
1135  0ECA 8926680F                  mov disk_sp,sp                  ;and stack pointer
1136  0ECE 2E8E16060C                mov ss,sysdat                   ;set up interrupt stack
1137  0ED3 BC660F                    mov sp,offset i_disk_stack
1138
1139  0ED6 FE06540D                  inc int_cnt
1140  0EDA FB                        sti
1141
1142  0EDB 505351                    push ax ! push bx ! push cx
1143  0EDE 525557                    push dx ! push bp ! push di
1144  0EE1 5606                      push si ! push es
1145
1146  0EE3 BA0400                    mov dx,fdc_flag                 ;DEV_SETFLAG call must have
1147  0EE6 E882FD       0C6B         call setflag                    ;DS = SYSDAT, but ES
1148                                                                 ;does not need to be set to UDA
1149                                                                 ;on flag set calls
1150  0EE9 075E5F                    pop es ! pop si ! pop di
1151  0EEC 5D5A59                    pop bp ! pop dx ! pop cx
```

```
1152
1153   0EEF 58                          pop bx                          ;AX still on stack
1154
1155   0EF0 FA                          cli                             ;keep this interrupt
1156                                                                    ;handler from being
1157                                                                    ;reentered
1158   0EF1 B020                        mov al,pic_nseoi                ;signal end of interrupt
1159   0EF3 E620                        out pic_even_port,al            ;to 8259
1160
1161   0EF5 58                          pop ax
1162
1163   0EF6 8E16660F                    mov ss,disk_ss                  ;restore user's stack
1164   0EFA 8B26680F                    mov sp,disk_sp
1165   0EFE FE0E5400                    dec int_cnt
1166   0F02 1F                          pop ds
1167   0F03 7401          0F06          jz id_disp
1168   0F05 CF                            iret
1169                            id_disp:
1170   0F06 2EFF2E3800                  jmpf cs:dword ptr dispatcher
1171
1172                            ;*************************************************************************
1173                            ;*                                                                       *
1174                            ;*                          I_DISK Data                                  *
1175                            ;*                                                                       *
1176                            ;*************************************************************************
1177
1178     0F0B                   SL11     equ      offset $
1179                            DSEG
1180                            ORG      SL11
1181
1182                            ;        Stack switch area for disk interrupt
1183
1184                            org      (offset $ + 1) and 0fffeh
1185
1186   0F0C CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1187   0F12 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1188   0F18 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1189   0F1E CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1190   0F24 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1191
1192   0F2A CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1193   0F30 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1194   0F36 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1195   0F3C CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1196   0F42 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1197
1198   0F48 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1199   0F4E CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1200   0F54 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1201   0F5A CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1202   0F60 CCCCCCCCCCCC                      dw       0CCCCH,0CCCCH,0CCCCH
1203
1204   0F66                   i_disk_stack     rs       0
```

```
1205
1206   0F66 0000              disk_ss         dw      0
1207   0F68 0000              disk_sp         dw      0
1208
1209                          ;************************************************************************
1210                          ;*                                                                      *
1211                          ;*                      I_UNEXPECTED                                    *
1212                          ;*                                                                      *
1213                          ;************************************************************************
1214
1215     0F6A                 SL12    equ     offset $
1216                          CSEG
1217                          ORG     SL12
1218
1219                          ;------------
1220                          i_unexpected:                   ;unknown interrupts go here
1221                          ;------------
1222
1223                          ;       Display on the console the interrupt vector number that caused
1224                          ;       the CPU to be here.  We will terminate the process that caused
1225                          ;       the unknown interrupt, even if its keep flag is on.
1226
1227   0F6A 803E3C0CFF                        cmp debug,true
1228   0F6F 7501          0F72                jne u_notd              ;break to CP/M-86
1229   0F71 CC                                  int 3                 ;when debugging
1230                              u_notd:
1231
1232   0F72 2EA1060C                          mov ax,sysdat
1233   0F76 8ED8                              mov ds,ax
1234   0F78 8EC0                              mov es,ax               ;ES,DS = SYSDAT SEGMENT
1235   0F7A BFBE0F                            mov di,offset unex_msg_pd
1236   0F7D 8B366800                          mov si,rlr              ;PD offset
1237   0F81 85F6                              test si,si
1238   0F83 7408          0F90                jz iu_nopd
1239   0F85 8D7408                              lea si,p_name[si]
1240   0F88 BFBE0F                              mov di,offset unex_msg_pd
1241   0F8B B90400                              mov cx,4
1242   0F8E F3A5                                rep movsw              ;get PD name into message string
1243
1244                              iu_nopd:
1245   0F90 BE800F                            mov si,offset unex_msg
1246   0F93 E88E12        2224                call print_msg          ;print unexpected interrupt message
1247
1248   0F96 8B1E6800                          mov bx,rlr              ;terminate the running process
1249   0F9A 85DB                              test bx,bx              ;if no process is running
1250   0F9C 7501          0F9F                jnz iu_abort            ;halt the machine
1251   0F9E F4                                  hlt                   ;this should only happen if
1252                                                                  ;kernel image has been corrupted
1253                              iu_abort:
1254   0F9F C747060000                        mov p_flag[bx],0        ;force termination to succeed
1255   0FA4 8E4710                            mov es,p_uda[bx]        ;set up UDA
1256   0FA7 B98F00                            mov cx,p_term
1257   0FAA BAFFFF                            mov dx,0ffffh
```

```
                                ;process is terminated, interrupts
                                ;are forced on when next process is
                                ;run is brought into context

                ;*****************************************************************
                ;*                                                               *
                ;*                      I_UNEXPECTED Data                        *
                ;*                                                               *
                ;*****************************************************************

0FB0            SL13    equ     offset $
                OSEG
                ORG     SL13

0FB0 0D0A54657726D      unex_msg        db      cr,lf,'Terminating '
     696E6174696E
     6720
0FBE 202020202020       unex_msg_pd     db      '       '
     2020
0FC6 3A20756E6578               db      ': unexpected interrupt',cr,lf,0
     706563746564
     20696E746572
     727570740D0A
     00
```

```
1284
1285                    eject
1286                    ;                       Character I/O
1287                    ;                       -------------
1288
1289                    ;***************************************************************************
1290                    ;*                                                                         *
1291                    ;*                      IBM PC Screen Equates                              *
1292                    ;*                                                                         *
1293                    ;***************************************************************************
1294
1295    0018            rows_per_screen         equ     24
1296    0050            columns_per_screen      equ     80
1297    0780            screen_siz              equ     rows_per_screen * columns_per_screen
1298                                                    ;in words
1299
1300    03C0            screens_size            equ     ((2 * screen_siz + 15)/16) * num_vir_cons
1301                                                    ;storage for all the screens
1302                                                    ;in paragraphs
1303    B000            bw_video_seg            equ     0B000h          ;segment address of
1304                                                    ;start of video ram
1305    0F00            bw_video_status_line    equ     screen_siz * 2
1306                                                    ;byte offset of status line
1307
1308                    ;***************************************************************************
1309                    ;*                                                                         *
1310                    ;*              IBM PC Parallel Port Equates                               *
1311                    ;*                                                                         *
1312                    ;***************************************************************************
1313
1314  0FDF 8C03         list_ports      dw      03bch           ;BW card parallel port
1315  0FE1 7803                         dw      0378h           ;0378h parallel port card as shipped
1316
1317                    ;***************************************************************************
1318                    ;*                                                                         *
1319                    ;*             6845 CRT Controller Port and Command Equates                *
1320                    ;*                                                                         *
1321                    ;***************************************************************************
1322
1323                    ;       The IBM PC's monochrome memory mapped video display begins
1324                    ;       at paragraph 0B000H.  It represents a screen 80 X 25.
1325                    ;       Each video character requires a word value, the low byte
1326                    ;       is the ASCII code (characters codes > 128 are also displayed)
1327                    ;       and the high byte is an attribute byte.  The 25th line
1328                    ;       is reserved by this XIOS as a status line.
1329
1330    03B4            bw_card                 equ     003b4h
1331
1332    0029            video_on                equ     00029h
1333    0021            video_off               equ     00021h
1334
1335    000A            cursor_start            equ     10
1336    000B            cursor_end              equ     11
```

```
1337
1338     000C                display_start_hi        equ     12
1339     000D                display_start_low       equ     13
1340     000E                cursor_hi               equ     14
1341     000F                cursor_low              equ     15
1342
1343                         ;*************************************************************************
1344                         ;*                                                                       *
1345                         ;*                          IO_CONST                                     *
1346                         ;*                                                                       *
1347                         ;*************************************************************************
1348
1349     0FE3                SL14    equ     offset $
1350                         CSEG
1351                         ORG SL14
1352
1353                         ;========
1354                         io_const:
1355                         ;========
1356                         ;       entry:  DL console number
1357                         ;       exit:   AL = 0ffh if ready
1358                         ;               AL = 0 if not
1359                         ;               ALL SEGMENT REGISTERS PRESERVED:
1360                         ;               CS,DS,ES,SS must be preserved though call
1361
1362   0FE3 33C0                     xor ax,ax
1363   0FE5 C3                       ret
1364
1365                         ;*************************************************************************
1366                         ;*                                                                       *
1367                         ;*                          IO_CONIN                                     *
1368                         ;*                                                                       *
1369                         ;*************************************************************************
1370
1371                         ;========
1372                         io_conin:
1373                         ;========
1374                         ;       entry:  DL console number
1375                         ;       exit:   AH = 0 and AL = character data
1376                         ;               AH = 0ffh and AL screen to switch to
1377                         ;               ALL SEGMENT REGISTERS PRESERVED:
1378                         ;               CS,DS,ES,SS must be preserved though call
1379
1380   0FE6 9CFA                     pushf ! cli
1381   0FE8 803EBE0E00               cmp kb_cnt,0
1382   0FED 750E          0FFD       jne ci_haveone
1383                              ci_wait:
1384   0FEF C606BF0EFF                 mov kb_wait,true
1385   0FF4 9D                         popf
1386   0FF5 BA0500                     mov dx,key_flag             ;get the flag number
1387   0FF8 E874FC        0C6F         call waitflag               ;wait for a scan code
1388   0FFB EB01          0FFE         jmps ci_haveone2
1389                              ci_haveone:
```

```
1390
1391  0FFD 9D                           popf
1392                                ci_haveone2:
1393  0FFE FE0EB6E0                     dec kb_cnt
1394  1002 A1B6E0                       mov ax,kb_out
1395  1005 8BD8                         mov bx,ax
1396  1007 8A07                         mov al,[bx]                   ;scan code in AL
1397  1009 43                           inc bx
1398  100A 81FBB8E0                     cmp bx,offset kb_buf+kblen
1399  100E 7503          1013           jne ci_ptr_ok
1400  1010 BBB9E0                         mov bx,offset kb_buf
1401                                ci_ptr_ok:
1402  1013 891EB6E0                     mov kb_out,bx
1403  1017 06                           push es                       ;save UDA
1404  1018 1E07                         push ds ! pop es              ;ES=SYSDAT
1405  101A E81000        1020           call scan_trans               ;returns AH=0 and AL=char to
1406  101D 07                           pop es                        ;return to process
1407                                                                  ;else AH=0ffh - check for
1408                                                                  ;switch screen
1409  101E 84E4                         test ah,ah                    ;AH=0 then done
1410  1020 740A          102C           jz conin_done
1411                                                                  ;AH=0ffh and AL=0 if released
1412                                                                  ;key, AL=special key otherwise
1413  1022 3CF9                           cmp al,0f9h                 ;0F0h-0F9h are switch screen
1414  1024 77C0          0FE6             ja io_conin                 ;codes
1415  1026 3CF0                             cmp al,0f0h
1416  1028 72BC          0FE6               jb io_conin               ;implement check for programable
1417                                                                  ;function keys here
1418  102A 240F                               and al,0fh               ;switch screen range 0-9
1419                                          ;jmp conin_done
1420                                conin_done:
1421  102C C3                           ret
1422
1423                                scan_trans:
1424                                ;----------
1425                                ;       entry:  AL=scan code
1426                                ;       exit:   AH=0 if ASCII and AL=ASCII code
1427                                ;               AH=0FFH if AL is special code
1428
1429                                ;       Test for CTRL, ALT already being down and DEL being current
1430                                ;       scan code.  If yes, call the ROM reset routine.
1431
1432  102D A27316                       mov scan_code,al              ;save scan code
1433  1030 3C537511      1045           cmp al,83 ! jnz mask_release  ;is scan code the DEL key ?
1434  1034 F606741601                     test down_bits,ctrl_bit     ;yes $ is control down ?
1435  1039 740A          1045             jz mask_release
1436  103B F606741608                       test down_bits,alt_bit    ;yes $ is alt down ?
1437  1040 7403          1045               jz mask_release
1438  1042 E9F411        2239                 jmp reset               ;yes alt and ctrl are down
1439
1440                                mask_release:                     ;parity is on when key is released
1441  1045 247F                         and al,7fh                    ;turn off key release bit
1442  1047 3C537E03      104E           cmp al,83 ! jle valid_key     ;test for non$existent key
```

```
1443
1444   1046 E98000        10CE            jmp no_key
1445
1446                              ;       Test for CTRL, SHIFT, ALT, NUMBLOCK, CAPSLOCK
1447                              ;       keys.  The scan codes for these keys are in action_key_table.
1448                              ;       and are refered to as action keys in the comments below.
1449                              ;       The bit position in AH corresponds with what kind of key is
1450                              ;       found, see equates below.
1451                              ;       The DOWN_BITS byte has bits on for action keys that are currently
1452                              ;       being held down by the operator.  The TOGGLE_BITS byte is similar
1453                              ;       but for keys that have toggle action, NUM LOCK, CAPS LOCK.
1454                              ;       Note, there is one only one set of DOWN_BITS, whereas
1455                              ;       the TOGGLE_BITS are kept on a per virtual console basis in the
1456                              ;       screen structures.
1457
1458                              valid_key:                              ;AL=scan code without parity
1459   104E 8B363618                      mov si,foreground_ss            ;foreground screen structure
1460   1052 BF7516                        mov di,offset action_key_table  ;look for CTRL,SHIFT,ALT,NUMLOCK
1461   1055 33C9                          xor cx,cx                       ;CAPSLOCK
1462   1057 B106                          mov cl,num_action_keys          ;get count
1463   1059 F2AE                          repnz scasb                     ;look for the byte
1464   105B 750C          1069            jnz not_action_key
1465   105D 4F                              dec di                        ;backup to matched action key
1466   105E 81EF7516                        sub di,offset action_key_table
1467                                                                      ;DI= 0 relative index
1468   1062 8AA57B16                        mov ah,offset action_key_masks[di]
1469                                                                      ;get the corresponding bit
1470   1066 E96A00        10D3            jmp action_key                  ;pattern
1471
1472                              not_action_key:
1473   1069 A07316                        mov al,scan_code
1474   106C 0AC07903      1073            or al,al ! jns key_make         ;sign is on if key just released
1475   1070 E95800        10CE              jmp no_key                    ;ignore release condition on
1476                                                                      ;non action type key
1477                              key_make:
1478   1073 BB9516                        mov bx,offset key_table         ;translation table for single keys
1479   1076 F606741601                    test down_bits,ctrl_bit         ;was control key already down ?
1480   107B 7405          1082            jz test_for_keypad              ;no - try numlock
1481   107D BB3D17                        mov bx,offset control_table     ;yes - point to control key table
1482   1080 EB20          10A2            jmps translate
1483
1484                              test_for_keypad:
1485   1082 3C477212      1098            cmp al,71 ! jb test_for_shift   ;test for keypad
1486   1086 F6440D10                        test ss_mode[si],ssm_numlock  ;from keypad, numlock on ?
1487   108A 740C          1098              jz test_for_shift
1488   108C F606741602                        test down_bits,shft_bit
1489   1091 750F          10A2                jnz translate               ;use key_table if numlock,shifted
1490   1093 BBE916                            mov bx,offset shift_table
1491   1096 EB0A          10A2                jmps translate              ;use shift table if numlock
1492
1493                              test_for_shift:
1494   1098 F606741602                    test down_bits,shft_bit
1495   109D 7403          10A2            jz translate                    ;no $ return key from key_table
```

```
1496
1497   109F BBE916                      mov bx,offset shift_table     ;yes $ get key from shift_table
1498
1499                              translate:
1500   10A2 D7                          xlat bx                       ;look up the key
1501   10A3 A880                        test al,80h                   ;is it special ?
1502   10A5 7529              10D0      jnz special_key               ;yes then done
1503   10A7 F6440D20                      test ss_mode[si],ssm_capslock ;test for caps lock
1504   10AB 7412              10BF        jz test_for_alt
1505   10AD 3C7A                            cmp al,'z'                ;yes
1506   10AF 770E              10BF          ja test_for_alt             ;not alphabetic
1507   10B1 3C61                              cmp al,'a'              ;is it lower case ?
1508   10B3 7308              10BD            jae do_case_change      ;yes, switch case
1509   10B5 3C5A                                cmp al,'Z'
1510   10B7 7706              10BF              ja test_for_alt       ;not alphabetic
1511   10B9 3C41                                  cmp al,'A'          ;test for upper case
1512   10BB 7202              10BF                jb test_for_alt     ;not alphabetic
1513                              do_case_change:
1514   10BD 3420                        xor al,020h                   ;switch the case
1515                              test_for_alt:
1516   10BF 247F                        and al,07fh                   ;mask off the sign bit from table
1517   10C1 F606741608                  test down_bits,alt_bit        ;is alt key currently down ?
1518   10C6 7402              10CA      jz printable                  ;no
1519   10C8 0C80                          or al,080h                  ;yes $ turn on the msb for alt key
1520                              printable:
1521   10CA 32E4                        xor ah,ah                     ;signal printable char
1522   10CC EB04              10D2      jmps scan_done
1523
1524                              no_key:
1525   10CE B000                        mov al,0                      ;AH=FF, AL=0 force TO_CONIN:
1526                                                                  ;to get another scan code
1527                              special_key:
1528   10D0 B4FF                        mov ah,0ffh
1529
1530                              scan_done:
1531   10D2 C3                          ret
1532
1533                              action_key:
1534                              ;----------
1535                              ;       Scan code in AL indicates an action key.  AH is
1536                              ;       the bit mask for the action key.
1537                              ;       Live action keys are those that must remain pressed down
1538                              ;       to have an effect, i.e., CTRL, SHIFT, ALT.  The state of live
1539                              ;       action keys are bits stored in DOWN_BITS.
1540                              ;       Toggle action keys are those that switch back and forth in
1541                              ;       function each time they are depressed, i.e., NUMLOCK, CAPSLOCK.
1542                              ;       The state of the toggle action keys are bits stored in the
1543                              ;       SS_MODE byte of the foreground screen structure.
1544
1545                              ;       entry:  AL = scan code, one of CTRL, SHIFT,
1546                              ;               ALT, NUMLOCK, CAPSLOCK with parity masked off
1547                              ;               AH = bit mask
1548                              ;               SI = foreground screen structure
```

```
1549
1550                            ;       exit:   jumps to NO_KEY since no printable
1551                            ;               character results
1552
1553                                                            ;ALT is last live key in table
1554   10D3 3C38                        cmp al,alt              ;see if live action: ALT,SHIFT,CTRL
1555   10D5 A07316                      mov al,scan_code        ;get scan code with parity
1556   10D8 7712          10EC          ja toggle_action        ;no - it is a toggle type key
1557   10DA A880                          test al,80H           ;pressed or released ?
1558   10DC 7408          10E6            jz action_make        ;no parity bit - key is down
1559   10DE F6D4                            not ah              ;parity bit on - key is released
1560   10E0 20267416                        and down_bits,ah    ;clear the bit for this live
1561   10E4 EB10          10F6              jmps action_key_done  ;action key
1562                                action_make:
1563   10E6 08267416                        or down_bits,ah     ;set bit for this live action key
1564   10EA EB0A          10F6              jmps action_key_done
1565
1566                                toggle_action:              ;NUMLOCK or CAPSLOCK
1567   10EC A880                          test al,80H           ;depressed or released ?
1568   10EE 7506          10F6            jnz action_key_done   ;ignore release of toggle
1569   10F0 30640D                          xor ss_mode[si],ah  ;it is toggle: reverse the state
1570   10F3 E8EA03        14E0              call update_status
1571                                        ;jmps action_key_done
1572
1573                                action_key_done:
1574   10F6 EBD6          10CE            jmps    no_key
1575
1576
1577                                ;***************************************************************************
1578                                ;*                                                                         *
1579                                ;*                          IO_CONOUT                                      *
1580                                ;*                                                                         *
1581                                ;***************************************************************************
1582
1583                                ;=========
1584                                io_conout:
1585                                ;=========
1586                                ;       entry:  CL = character to output
1587                                ;               DL = device number
1588                                ;       exit:   None
1589                                ;               ALL SEGMENT REGISTERS PRESERVED:
1590                                ;               CS,DS,ES,SS must be preserved though call
1591
1592                                ;       While in IO_CONOUT routines:
1593                                ;               AL = character to output
1594                                ;               AH = device number
1595                                ;               BX = screen structure
1596
1597   10F8 E80100        10FC            call ioo_conout
1598   10FB C3                            ret
1599
1600                                ioo_conout:
1601   10FC 3A16110C                        cmp dl,nvcns
```

```
1602
1603   1100 7201          1103          jb co_ok
1604   1102 C3                            ret                          ;support of character devices
1605                                                                    ;which are not virtual consoles
1606                                                                    ;can go here ...
1607                              co_ok:
1608                                                                    ;NOTE: PIN will not call
1609                                                                    ;IO_SWITCH, while in
1610                                                                    ;IO_CONOUT and we are
1611                                                                    ;the foreground or requested
1612                                                                    ;screen.
1613   1103 8AC1                        mov al,cl
1614   1105 8AE2                        mov ah,dl                      ;AH = virtual console #
1615   1107 32F6                        xor dh,dh                      ;get screen structure for
1616   1109 8BDA                        mov bx,dx                      ;this console
1617   110B D1E3                        shl bx,1                       ;word index
1618   110D 8B9F2B18                    mov bx,screen_struct_addrs[bx]
1619   1111 FF6704                      jmp ss_escape[bx]              ;escape handler if in the middle
1620                                                                    ;of and escape sequence else
1621                                                                    ;jump to CO_NO_ESCAPE
1622
1623                              co_no_escape:                         ;look for CR,LF,backspace,ESC
1624   1114 BF9117                      mov di,offset special_char_tab ;and jump to special handler
1625   1117 BE9617                      mov si,offset special_func_tab
1626   111A E84900        1166          call co_lookup
1627   111D E301          1120          jcxz co_simple_char
1628   111F C3                            ret                          ;all done if special handling
1629                                                                    ;or expanding and escape
1630                                                                    ;sequence
1631
1632                              co_simple_char:                       ;simple character output
1633
1634                              ;       Put character in screen and update cursor position
1635                              ;       BX = screen structure
1636                              ;       AL = character
1637
1638   1120 3C07                        cmp al,bel                     ;is it a bell ?
1639   1122 751F          1143          jne no_bell
1640   1124 3B1E3618                      cmp bx,foreground_ss         ;bells to background consoles
1641   1128 7518          1142            jne ignore_bell            ;are ignored
1642   112A 803E520D00                      cmp bell_ticks,0           ;are we generating a bell
1643   112F 7511          1142              jne ignore_bell          ;already ?
1644   1131 E461                              in al,port_b             ;and then off for 1/60
1645   1133 0C03                              or al,bell_on
1646   1135 E661                              out port_b,al
1647   1137 C606520D07                        mov bell_ticks,7         ;a bell is on 6/60 of a sec
1648   113C BA0600                            mov dx,bell_flag         ;wait for tick interrupt
1649   113F E82DFB        0C6F                call waitflag            ;(I_TICK:) to call DEV_SETFLAG
1650                                                                    ;when bell is done
1651
1652                              ignore_bell:                          ;NOTE: PIN will not call
1653   1142 C3                          ret                            ;IO_SWITCH, while in
1654                                                                    ;IO_CONOUT and we are
```

```
1655
1656                                                    ;the foreground or requested
1657                                                    ;screen.
1658                            no_bell:
1659    1143 893F                       mov di,ss_cursor[bx]        ;cursor byte offset
1660    1145 06                         push es
1661    1146 E81303     1461            call set_up_es
1662
1663                            co_back:
1664    1149 50                         push ax                     ;save virtual console number
1665    114A 8A670C                     mov ah,ss_attribute[bx]
1666    114D AB                         stosw                       ;update data and attribute
1667    114E 58                         pop ax
1668    114F 07                         pop es
1669                                                    ;DI points at next data
1670                                                    ;and attribute word
1671    1150 807F094F                   cmp ss_column[bx],columns_per_screen-1
1672    1154 7209       1163            jb inc_col
1673    1156 F6470D02                     test ss_mode[bx],ssm_no_wrap
1674    115A 7506       1162              jnz co_nowrap
1675    115C E82800     1187                call carriage_return
1676    115F E93100     1193                jmp line_feed
1677                            co_nowrap:
1678    1162 C3                           ret                       ;don't move cursor
1679                                                    ;when at end of line
1680                            inc_col:
1681    1163 E9A000     1206            jmp r_right                 ;move cursor right
1682
1683
1684                            co_look_up:
1685                            ;----------
1686                            ;       entry:  BX = address of screen structure
1687                            ;               AH = device#
1688                            ;               AL = character to scan for
1689                            ;               DI = ptr to lookup table, first byte is length
1690                            ;               SI = table of functions to jump to if char found in
1691                            ;                    lookup table
1692                            ;       exit:   CX = 0ffffh if special character found
1693                            ;                    and special function has been performed
1694                            ;               CX = 0 if char not found, no special handling is performed
1695                            ;               AX,BX preserved if not special function
1696
1697    1166 8CC38CDA                   mov bp,es ! mov dx,ds       ;use registers instead of
1698    116A 8EC2                       mov es,dx                   ;stack for speed
1699    116C 33C98A0D                   xor cx,cx ! mov cl,[di]     ;length of lookup
1700    1170 8BD1                       mov dx,cx                   ;save the length
1701    1172 47                         inc di
1702    1173 F2AE                       repne scasb                 ;is it in table ?
1703    1175 8EC5                       mov es,bp                   ;restore DDA
1704    1177 7401       117A            je lu_func                  ;CX=0 if not found
1705    1179 C3                           ret
1706
1707                            lu_func:
```

```
1708
1709   117A 41                          inc cx                          ;iteration we matched on
1710   117B 2B01                        sub dx,cx                       ;function number
1711   117D D1E2                        shl dx,1                        ;jump to special function
1712   117F 03F2                        add si,dx
1713   1181 FF14                        call word ptr [si]
1714   1183 33C949                      xor cx,cx ! dec cx              ;return CX=0FFFFH
1715   1186 C3                          ret
1716
1717                        ;*************************************************************************
1718                        ;*                                                                       *
1719                        ;*              Special Output Character Routines                        *
1720                        ;*                                                                       *
1721                        ;*************************************************************************
1722
1723                        ;---------------
1724                        carriage_return:
1725                        ;---------------
1726                        ;       entry:  BX = screen structure
1727                        ;       exit:   BX preserved
1728
1729   1187 33D2                        xor dx,dx
1730   1189 865709                      xchg ss_column[bx],dl           ;set column to 0
1731   118C D1E2                        shl dx,1                        ;back up cursor
1732   118E 2917                        sub ss_cursor[bx],dx
1733                        cr_done:
1734   1190 E98102          1444        jmp set_physical_cursor
1735
1736                        ;---------
1737                        line_feed:
1738                        ;---------
1739                        ;       entry:  BX = screen structure
1740                        ;       exit:   BX preserved
1741
1742   1193 807F0817                    cmp ss_row[bx],rows_per_screen-1
1743   1197 7403            119C        je lf_scroll
1744   1199 E98A00          1226          jmp d_down                    ;no-just move cursor down
1745                        lf_scroll:                                   ;yes-scroll screen up
1746   119C 32D2                          xor dl,dl                     ;start row
1747   119E B617                          mov dh,rows_per_screen-1      ;end row
1748   11A0 E82402          13C7          call scroll_up
1749   11A3 E92201          12C8          jmp erase_line
1750
1751                        ;----------
1752                        back_space:
1753                        ;----------
1754                        ;       entry:  BX = screen structure
1755                        ;       exit:   BX preserved
1756
1757   11A6 8B3F                        mov di,ss_cursor[bx]
1758   11A8 85FF                        test di,di                      ;at home position, return
1759   11AA 7420            11CC        jz bs_ret
1760   11AC 807F0900                      cmp ss_column[bx],0           ;if column 0, go up to previous line
```

```
1761
1762   11B0 7405          1187          je bs_row
1763   11B2 FE4F09                        dec ss_column[bx]
1764   11B5 E80D          11C4          jmps bs_set_cursor
1765                            bs_row:
1766   11B7 F6470D02                      test ss_mode[bx],ssm_no_wrap  ;don't back up to previous row
1767   11BB 750F          11CC            jnz bs_ret                    ;if in NOWRAP mode
1768   11BD FE4F08                        dec ss_row[bx]
1769   11C0 C647094F                      mov ss_column[bx],columns_per_screen - 1
1770                            bs_set_cursor:
1771   11C4 83EF02                        sub di,2
1772   11C7 893F                          mov ss_cursor[bx],di
1773   11C9 E97802        1444            jmp set_physical_cursor
1774                            bs_ret:
1775   11CC C3                          ret
1776
1777                            ;*************************************************************************
1778                            ;*                                                                       *
1779                            ;*                      Escape Sequence Routines                         *
1780                            ;*                                                                       *
1781                            ;*************************************************************************
1782
1783                            escape:                                 ;expand escape sequences
1784                            ;------
1785                            ;       entry:  BX = screen_structure address for this device console
1786                            ;               AL = current byte in escape sequence
1787                            ;       exit:   ss_escape = address of where to continue
1788                            ;                           escape sequence or 0 if done
1789
1790   11CD C747040311                  mov ss_escape[bx],offset escape1
1791   11D2 C3                          ret                            ;back to co_look_up
1792
1793                            escape1:                                ;first char of escape sequence
1794   11D3 C747041411                  mov ss_escape[bx],offset co_no_escape
1795                                                                   ;assume done with escape
1796   11D8 BF9E17                      mov di,offset esc_tab1         ;if more chars are needed
1797   11DB BE8B17                      mov si,offset esc_func_tab1    ;in the escape sequence, the
1798   11DE E935FF        1166          jmp co_look_up                 ;escape field is set by the
1799                                                                   ;handler - see X_AND_Y:
1800                            ;------------
1801                            erase_screen:                           ;ESC E
1802                            ;------------
1803                            ;       entry:  BX = screen structure whose screen data to blank
1804                            ;               AH = virtual console #
1805                            ;       exit:   AX,CX,DI are used
1806                            ;               BX = screen structure
1807
1808   11E1 5006                        push ax ! push es
1809   11E3 E87902        1461          call set_up_es
1810   11E6 B98007                      mov cx,screen_siz
1811   11E9 33FF                        xor di,di
1812   11EB B82007                      mov ax,0720h
1813   11EE F3AB                        rep stosw
```

```
1814
1815   11F0 0758                        pop es ! pop ax
1816                                    ;jmps home
1817
1818                            ;----
1819                            home:                                   ;ESC H
1820                            ;----
1821                            ;       entry:  BX = screen structure
1822                            ;       exit:   BX preserved
1823
1824   11F2 33D2                        xor dx,dx
1825   11F4 8917                        mov ss_cursor[bx],dx
1826   11F6 885708                      mov ss_row[bx],dl
1827   11F9 885709                      mov ss_column[bx],dl
1828   11FC E94502      1444            jmp set_physical_cursor
1829
1830                            ;-----
1831                            right:                                  ;ESC C
1832                            ;-----
1833                            ;       entry:  BX = screen structure
1834                            ;       exit:   BX preserved
1835
1836   11FF 807F094F                    cmp ss_column[bx],columns_per_screen-1
1837   1203 7201        1206            jb r_right
1838   1205 C3                            ret
1839                           r_right:
1840   1206 830702                        add ss_cursor[bx],2
1841   1209 FE4709                        inc ss_column[bx]
1842   120C E93502      1444              jmp set_physical_cursor
1843
1844                            ;----
1845                            left:                                   ;ESC D
1846                            ;----
1847                            ;       entry:  BX = screen structure
1848                            ;       exit:   BX preserved
1849
1850   120F 807F0900                    cmp ss_column[bx],0
1851   1213 7701        1216            ja l_left
1852   1215 C3                            ret
1853                           l_left:
1854   1216 832F02                        sub ss_cursor[bx],2
1855   1219 FE4F09                        dec ss_column[bx]
1856   121C E92502      1444              jmp set_physical_cursor
1857
1858                            ;----
1859                            down:                                   ;ESC B
1860                            ;----
1861                            ;       entry:  BX = screen structure
1862                            ;       exit:   BX preserved
1863
1864   121F 807F0817                    cmp ss_row[bx],rows_per_screen-1
1865   1223 7201        1226            jb d_down
1866   1225 C3                            ret
```

```
1867
1868                            d_down:
1869   1226 8107A000                      add ss_cursor[bx],columns_per_screen*2
1870   122A FE4708                        inc ss_row[bx]
1871   122D E91402          1444          jmp set_physical_cursor
1872
1873                            ;--
1874                            up:                                     ;ESC A
1875                            ;--
1876                            ;       entry:  BX = screen structure
1877                            ;       exit:   BX preserved
1878
1879   1230 807F0800                      cmp ss_row[bx],0
1880   1234 7701            1237          ja d_up
1881   1236 C3                            ret
1882                            d_up:
1883   1237 812FA000                      sub ss_cursor[bx],columns_per_screen*2
1884   123B FE4F08                        dec ss_row[bx]
1885   123E E90302          1444          jmp set_physical_cursor
1886
1887                            ;--------------
1888                            up_with_scroll:                         ;ESC I
1889                            ;--------------
1890                            ;       entry:  BX = screen structure
1891                            ;       exit:   BX preserved
1892
1893   1241 807F0800                      cmp ss_row[bx],0                ;if not the top line
1894   1245 77E9            1230          ja up                           ;just do a cursor up
1895   1247 32D2                          xor dl,dl                       ;start row
1896   1249 B617                          mov dh,rows_per_screen-1        ;ending row
1897   124B E8A801          13F6          call scroll_down                ;down one row
1898   124E E87700          12C8          call erase_line                 ;blank out current row
1899   1251 C3                            ret                             ;don't move the cursor
1900
1901                            ;------
1902                            report:
1903                            ;------
1904   1252 C3                          ret
1905
1906                            ;----
1907                            save:                                   ;ESC j
1908                            ;----
1909                            ;       entry:  BX = screen structure
1910                            ;       exit:   BX preserved
1911                            ;               AX destroyed
1912
1913   1253 8B4708                      mov ax,ss_xy[bx]
1914   1256 89470A                      mov ss_old_xy[bx],ax
1915   1259 8B07                        mov ax,ss_cursor[bx]
1916   125B 894702                      mov ss_oldcursor[bx],ax
1917   125E C3                          ret
1918
1919                            ;-------
```

```
1920
1921                            restore:                                ;ESC k
1922                            ;-------
1923                            ;       entry:  BX = screen structure
1924                            ;       exit:   AX,BX preserved
1925
1926   125F 8B4F0A                      mov cx,ss_oldxy[bx]
1927   1262 894F08                      mov ss_xy[bx],cx
1928   1265 8B4F02                      mov cx,ss_oldcursor[bx]
1929   1268 890F                        mov ss_cursor[bx],cx
1930   126A E9D701      1444            jmp set_physical_cursor
1931
1932                            ;-------
1933                            x_and_y:                                ;ESC Y
1934                            ;-------
1935                            ;       entry:  BX = screen structure
1936                            ;       exit:   BX preserved
1937
1938   126D C747047312                  mov ss_escape[bx],offset xy_row ;wait for row
1939   1272 C3                          ret
1940                            xy_row:
1941   1273 2C20                        sub al,32                       ;make row relative to 0
1942   1275 3C17                        cmp al,rows_per_screen-1
1943   1277 7605        127E            jbe row_ok
1944   1279 C747049812                    mov ss_escape[bx],offset xy_err
1945                            row_ok:
1946   127E 884708                      mov ss_row[bx],al
1947   1281 C747048712                  mov ss_escape[bx],offset xy_col ;wait for column
1948   1286 C3                          ret
1949                            xy_col:
1950   1287 2C20                        sub al,32                       ;make column relative to 0
1951   1289 3C4F                        cmp al,columns_per_screen-1
1952   128B 7602        128F            jbe xy_set_col
1953   128D B04F                          mov al,columns_per_screen-1
1954                            xy_set_col:
1955   128F 884709                      mov ss_column[bx],al
1956   1292 E89701      142C            call compute_cursor
1957   1295 E8AC01      1444            call set_physical_cursor
1958                            xy_err:
1959   1298 C747041411                  mov ss_escape[bx],offset co_no_escape
1960   129D C3                          ret
1961
1962                            ;-----------
1963                            erase_begin:                            ;ESC b
1964                            ;-----------
1965                            ;       entry:  BX = screen structure
1966                            ;               AH = virtual screen number
1967                            ;       exit:   AX,CX,DI are used
1968                            ;               BX = screen structure
1969
1970   129E 06                          push es
1971   129F E88F01      1461            call set_up_es
1972
```

```
1973
1974   12A2 8B07                    mov ax,ss_cursor[bx]          ;cursor byte offset
1975   12A4 D1E8                    shr ax,1                      ;make word offset
1976   12A6 40                      inc ax                        ;make relative to 1
1977   12A7 8BC8                    mov cx,ax                     ;number of words to erase
1978   12A9 B82007                  mov ax,0720h                  ;7 = normal attribute
1979                                                              ;20 = ASCII space
1980   12AC 33FF                    xor di,di                     ;start at beginning
1981   12AE F3AB                    rep stosw                     ;of display
1982   12B0 07                      pop es
1983   12B1 C3                      ret
1984
1985                        ;---------
1986                        erase_end:                            ;ESC j
1987                        ;---------
1988                        ;       entry:  BX = screen structure
1989                        ;               AH = virtual screen number
1990                        ;       exit:   AX,CX,DI are used
1991                        ;               BX = screen structure
1992
1993   12B2 06                      push es
1994   12B3 E8AB01       1461       call set_up_es
1995
1996   12B6 8B07                    mov ax,ss_cursor[bx]          ;cursor byte offset
1997   12B8 8BF8                    mov di,ax                     ;starting offset to erase
1998   12BA D1E8                    shr ax,1                      ;make word offset
1999   12BC B98007                  mov cx,screen_siz             ;screen size in words
2000   12BF 2BC8                    sub cx,ax                     ;less 0 relative cursor position
2001   12C1 B82007                  mov ax,0720h                  ;7 = normal attribute
2002                                                              ;20 = ASCII space
2003   12C4 F3AB                    rep stosw
2004   12C6 07                      pop es
2005   12C7 C3                      ret
2006
2007                        ;----------
2008                        erase_line:                           ;ESC l
2009                        ;----------
2010                        ;       entry:  BX = screen structure
2011                        ;       exit:   BX preserved
2012
2013   12C8 06                      push es
2014   12C9 E89501       1461       call set_up_es
2015
2016   12CC 33C0                    xor ax,ax
2017   12CE 8A4708                  mov al,ss_row[bx]             ;current row
2018   12D1 B95000                  mov cx,columns_per_screen     ;words to move
2019   12D4 F7E1                    mul cx                        ;word offset of current row
2020   12D6 D1E0                    shl ax,1                      ;byte offset
2021   12D8 8BF8                    mov di,ax                     ;destination
2022   12DA B82007                  mov ax,0720h                  ;7 = normal attribute
2023                                                              ;20 = ASCII space
2024   12DD F3AB                    rep stosw
2025   12DF 07                      pop es
```

```
2026
2027   12E0 C3                          ret
2028
2029                            ;---------
2030                            erase_bol:                          ;ESC o
2031                            ;---------                          ;erase from beginning
2032                            ;       entry:  BX = screen structure ;of line
2033                            ;       exit:   BX preserved
2034
2035   12E1 06                          push es
2036   12E2 E87C01       1461           call set_up_es
2037
2038   12E5 33C0                        xor ax,ax
2039   12E7 8A4708                      mov al,ss_row[bx]           ;current row
2040   12EA B9A000                      mov cx,columns_per_screen*2
2041   12ED F7E1                        mul cx                      ;byte offset of row
2042   12EF 8BF9                        mov di,ax
2043   12F1 33C9                        xor cx,cx
2044   12F3 8A4F09                      mov cl,ss_column[bx]        ;words to blank
2045   12F6 41                          inc cx                      ;include cursor
2046   12F7 B82007                      mov ax,0720h                ;7 = normal attribute
2047                                                                ;20 = ASCII space
2048   12FA F3AB                        rep stosw
2049   12FC 07                          pop es
2050   12FD C3                          ret
2051
2052                            ;---------
2053                            erase_eol:                          ;ESC K
2054                            ;---------                          ;erase to end of line
2055                            ;       entry:  BX = screen structure
2056                            ;       exit:   BX preserved
2057
2058   12FE 06                          push es
2059   12FF E85F01       1461           call set_up_es
2060
2061   1302 8B3F                        mov di,ss_cursor[bx]
2062   1304 B95000                      mov cx,columns_per_screen
2063   1307 2A4F09                      sub cl,ss_column[bx]        ;words to blank
2064   130A B82007                      mov ax,0720h                ;7 = normal attribute
2065                                                                ;20 = ASCII space
2066   130D F3AB                        rep stosw
2067   130F 07                          pop es
2068   1310 C3                          ret
2069
2070                            ;-----------
2071                            insert_line:                        ;ESC L
2072                            ;-----------
2073                            ;       entry:  BX = screen structure
2074                            ;       exit:   BX preserved
2075
2076   1311 50                          push ax                     ;save virtual console number
2077   1312 8A5708                      mov dl,ss_row[bx]           ;start row
2078   1315 B617                        mov dh,rows_per_screen-1    ;ending row
```

```
2079
2080   1317 E8DC00       13F6          call scroll_down                ;move screen down one row
2081   131A E8A8FF       12C8          call erase_line                 ;blank out the line
2082   131D 33C0                       xor ax,ax                       ;save virtual console number
2083   131F 864709                     xchg al,ss_column[bx]           ;set cursor to beginning
2084   1322 D1E0                       shl ax,1                        ;of line
2085   1324 2907                       sub ss_cursor[bx],ax
2086   1326 58                         pop ax
2087   1327 E91A01       1444          jmp set_physical_cursor
2088
2089                            ;-----------
2090                            delete_line:                            ;ESC M
2091                            ;-----------
2092                            ;       entry:  BX = screen structure
2093                            ;       exit:   BX preserved
2094
2095   132A 50                         push ax                         ;save virtual console number
2096   132B 8A5708                     mov dl,ss_row[bx]               ;move screen up one line
2097   132E B617                       mov dh,rows_per_screen-1        ;to present row
2098   1330 E8D400       13C7          call scroll_up
2099   1333 FF7708                     push ss_xy[bx]                  ;save row and column
2100   1336 C6470817                   mov ss_row[bx],rows_per_screen-1
2101   133A E88BFF       12C8          call erase_line                 ;erase last line
2102   133D 8F4708                     pop ss_xy[bx]                   ;restore row and column
2103   1340 58                         pop ax                          ;restore virtual console number
2104   1341 E943FE       1187          jmp carriage_return             ;put cursor at beginning of line
2105
2106                            ;-----------
2107                            delete_char:                            ;ESC N
2108                            ;-----------
2109                            ;       entry:  BX = screen structure
2110                            ;       exit:   BX preserved
2111
2112   1344 06                         push es
2113   1345 E81901       1461          call set_up_es
2114   1348 8B3F                       mov di,ss_cursor[bx]
2115   134A 8BF7                       mov si,di
2116   134C 83C602                     add si,2
2117
2118   134F 33C9                       xor cx,cx
2119   1351 B14F                       mov cl,columns_per_screen-1
2120   1353 2A4F09                     sub cl,ss_column[bx]
2121
2122   1356 1E061F                     push ds ! push es ! pop ds
2123   1359 F3A5                       rep movsw
2124   135B C7052007                   mov 0[di],0720H                 ;put a blank in last column
2125   135F 1F07                       pop ds ! pop es
2126   1361 C3                         ret
2127
2128                            ;-------------
2129                            enter_reverse:                          ;ESC p
2130                            ;-------------
2131                            ;       entry:  BX = screen structure
```

```
2132
2133                            ;       exit:   BX preserved
2134
2135                            ;       video attribute byte has the following structure:
2136                            ;
2137                            ;     -- high nibble ---        --- low nibble ----
2138                            ;       7       6   5   4         3       2   1   0
2139                            ;     blink   background        bright   foreground
2140                            ;               color                      color
2141                            ;
2142                            ;       So we must swap the colors and keep the blink and bright
2143                            ;       bits unchanged.
2144
2145  1362 F6470D01                             test ss_mode[bx],ssm_reverse    ;already in reverse video ?
2146  1366 7517             137F                jnz rev_done
2147  1368 804F0D01                               or ss_mode[bx],ssm_reverse    ;turn on reverse video mode bit
2148                                              ;jmps rev_swap
2149
2150                                  rev_swap:
2151                                  ;--------
2152  136C 8A470C                               mov al,ss_attribute[bx]
2153  136F 8AE0                                 mov ah,al                       ;save blink and bright bits
2154  1371 2477                                 and al,077h                     ;mask off blink and bright bits
2155  1373 B104                                 mov cl,4
2156  1375 D2C8                                 ror al,cl                       ;swap nibbles and colors
2157  1377 80E488                               and ah,088h                     ;get just the blink and bright bits
2158  137A 0AC4                                 or al,ah                        ;put them together again
2159  137C 88470C                               mov ss_attribute[bx],al         ;save the new attribute
2160                                  rev_done:
2161  137F C3                                   ret
2162
2163                                  ;------------
2164                                  exit_reverse:                           ;ESC q
2165                                  ;------------
2166                                  ;       entry:  BX = screen structure
2167                                  ;       exit:   BX preserved
2168
2169  1380 F6470D01                             test ss_mode[bx],ssm_reverse
2170  1384 74F9             137F                jz rev_done
2171  1386 80670DFE                               and ss_mode[bx],not ssm_reverse       ;turn off reverse mode bit
2172  138A EBE0             136C                  jmps rev_swap
2173
2174                                  ;-----------
2175                                  enter_blink:                            ;ESC s
2176                                  ;-----------
2177                                  ;       entry:  BX = screen structure
2178                                  ;       exit:   BX preserved
2179
2180  138C 804F0C80                             or ss_attribute[bx],080h
2181  1390 C3                                   ret
2182
2183                                  ;----------
2184                                  exit_blink:                             ;ESC t
```

```
                        ;----------
                        ;       entry:  BX = screen structure
                        ;       exit:   BX preserved
                        
1391 80670C7F                   and ss_attribute[bx],not (080h)
1395 C3                         ret
                        
                        ;------------
                        enter_bright:                           ;ESC r
                        ;------------
                        ;       entry:  BX = screen structure
                        ;       exit:   BX preserved
                        
1396 804F0C08                   or ss_attribute[bx],08h
139A C3                         ret
                        
                        ;-----------
                        exit_bright:                            ;ESC u
                        ;-----------
                        ;       entry:  BX = screen structure
                        ;       exit:   BX preserved
                        
139B 80670CF7                   and ss_attribute[bx],not (08h)
139F C3                         ret
                        
                        ;-------------
                        enable_cursor:                          ;ESC e
                        ;-------------
                        ;       entry:  BX = screen structure
                        ;       exit:   BX preserved
                        
13A0 80670DFB                   and ss_mode[bx],not ssm_no_cursor       ;used by IO_SWITCH:
13A4 B40B                       mov ah,0bh                      ;on cursor byte
13A6 EB06           13AE        jmps cursor_on_off
                        
                        ;-------------
                        disable_cursor:                         ;ESC f
                        ;-------------
                        ;       entry:  BX = screen structure
                        ;       exit:   BX preserved
                        
13A8 804F0D04                   or ss_mode[bx],ssm_no_cursor    ;used by IO_SWITCH:
13AC B426                       mov ah,26h                      ;off cursor byte
                        
                        cursor_on_off:
13AE BAB403                     mov dx,bw_card                  ;get the control register
13B1 B00A                       mov al,cursor_start             ;register to update
13B3 EE                         out dx,al
13B4 42                         inc dx
13B5 8AC4                       mov al,ah                       ;turn off/on cursor
13B7 EE                         out dx,al
13B8 C3                         ret
```

```
2238
2239
2240                            ;-----------
2241                            enable_wrap:                    ;ESC v
2242                            ;-----------
2243                            ;       entry:  bX = screen structure
2244                            ;       exit:   nothing preserved since ID_STATLINE is
2245                            ;               called
2246
2247   13B9 806700FD                    and ss_mode[bx],not ssm_no_wrap
2248   13BD E92001      14E0            jmp update_status
2249
2250                            ;------------
2251                            disable_wrap:                   ;ESC w
2252                            ;------------
2253                            ;       entry:  BX = screen structure
2254                            ;       exit:   nothing preserved since ID_STATLINE is
2255                            ;               called
2256
2257   13C0 804F0D02                    or ss_mode[bx],ssm_no_wrap
2258   13C4 E91901      14E0            jmp update_status
2259
2260                            ;*****************************************************************************
2261                            ;*                                                                           *
2262                            ;*              Console Output Sub-Routines                                  *
2263                            ;*                                                                           *
2264                            ;*****************************************************************************
2265
2266                            scroll_up:                              ;scroll up in range given in
2267                            ;---------                              ;DX
2268                            ;       entry:  AH = virtual console number
2269                            ;               BX = screen structure
2270                            ;               DL = start row, 0 relative
2271                            ;               DH = ending row, 0 relative
2272                            ;       exit:   AX,BX,ES preserved
2273
2274   13C7 5006                        push ax ! push es
2275   13C9 E89500      1461            call set_up_es
2276   13CC 8BCA                        mov cx,dx                      ;copy of start,ending rows
2277
2278   13CE 33C0                        xor ax,ax                      ;set up SI,DI
2279   13D0 8AC2                        mov al,dl                      ;AX = beginning row
2280   13D2 BAA000                      mov dx,columns_per_screen*2
2281   13D5 F7E2                        mul dx                         ;byte offset of beginning row
2282   13D7 8BF8                        mov di,ax                      ;first destination word
2283   13D9 05A000                      add ax,columns_per_screen*2    ;first word to copy
2284   13DC 8BF0                        mov si,ax
2285                                                                   ;set up CX
2286   13DE 8BC1                        mov ax,cx                      ;AL=start,AH=end
2287   13E0 2AE0                        sub ah,al                      ;end always > start
2288   13E2 32C0                        xor al,al
2289   13E4 86C4                        xchg al,ah                     ;AX=number or rows to move
2290
```

```
2291
2292   13E6 8A5000                  mov dx,columns_per_screen
2293   13E9 F7E2                    mul dx                          ;words to move up
2294   13EB 8BC8                    mov cx,ax                       ;get count
2295   13ED 1E061F                  push ds ! push es ! pop ds
2296   13F0 F3A5                    rep movsw
2297   13F2 1F0758                  pop ds ! pop es ! pop ax
2298   13F5 C3                      ret
2299
2300                        scroll_down:                            ;scroll down in range given
2301                        ;-----------                            ;in DX
2302                        ;       entry:  AH = virtual console number
2303                        ;               BX = screen structure
2304                        ;               DL = start row, 0 relative
2305                        ;               DH = ending row, 0 relative
2306                        ;       exit:   AX,BX,ES preserved
2307
2308   13F6 5006                    push ax ! push es
2309   13F8 E86500        1461      call set_up_es
2310   13FB 8BCA                    mov cx,dx                       ;copy of start,ending rows
2311                                                                ;set up SI,DI
2312   13FD 33C0                    xor ax,ax
2313   13FF 8AC6                    mov al,dh                       ;AX=ending row
2314   1401 FEC0                    inc al                          ;next row after end row
2315   1403 8AA000                  mov dx,columns_per_screen*2
2316   1406 F7E2                    mul dx                          ;byte offset of next row
2317   1408 2D0200                  sub ax,2                        ;last word of ending row
2318   140B 8BF8                    mov di,ax                       ;first destination word
2319   140D 2DA000                  sub ax,columns_per_screen*2     ;first word to copy
2320   1410 8BF0                    mov si,ax
2321   1412 FD                      std                             ;auto-decrement SI,DI
2322                                                                ;set up CX
2323   1413 8BC1                    mov ax,cx                       ;AL=start,AH=ending
2324   1415 2AE0                    sub ah,al                       ;end always > start
2325   1417 32C0                    xor al,al
2326   1419 86C4                    xchg al,ah                      ;# rows to move
2327   141B 8A5000                  mov dx,columns_per_screen
2328   141E F7E2                    mul dx                          ;words to move down
2329   1420 8BC8                    mov cx,ax                       ;get count
2330   1422 1E061F                  push ds ! push es ! pop ds
2331   1425 F3A5                    rep movsw
2332   1427 1F0758                  pop ds ! pop es ! pop ax
2333   142A FC                      cld                             ;zero direction flag !
2334   142B C3                      ret
2335
2336                        compute_cursor:                 ;update SS_CURSOR field from
2337                        ;--------------                 ;SS_ROW and SS_COLUMN fields
2338                        ;       entry:  BX = screen structure address
2339                        ;               AH = virtual console device #
2340                        ;               ES = segment of video
2341                        ;       exit:   CX,DX used
2342                        ;               AX,BX preserved
2343
```

```
2344
2345   142C 50                          push ax                         ;save virtual console number
2346   142D 33C0                        xor ax,ax
2347   142F 8A4708                      mov al,ss_row[bx]               ;row and column
2348   1432 B9A000                      mov cx,columns_per_screen*2
2349   1435 F7E1                        mul cx
2350   1437 33C9                        xor cx,cx
2351   1439 8A4F09                      mov cl,ss_column[bx]
2352   143C D1E1                        shl cx,1
2353   143E 03C1                        add ax,cx
2354   1440 8907                        mov ss_cursor[bx],ax            ;word offset in screen array
2355   1442 58                          pop ax                          ;restore virtual console number
2356   1443 C3                          ret
2357
2358                            set_physical_cursor:
2359                            ;-------------------
2360                            ;       If the screen is the foreground screen then set the
2361                            ;       cursor register in the 6845 to SS_CURSOR[BX]
2362                            ;
2363                            ;       entry:  BX = screen structure address
2364                            ;               AH = virtual console device #
2365                            ;               ES = segment of video
2366                            ;       exit:   AX,CX,DX used
2367
2368   1444 3A263318                    cmp ah,foreground_screen
2369   1448 7516          1460          jne sp_ret
2370   144A 8B0F                          mov cx,ss_cursor[bx]          ;byte offset of cursor
2371   144C D1E9                          shr cx,1                      ;make a word value
2372   144E BA8403                        mov dx,bw_card                ;get the control register
2373   1451 B00F                          mov al,cursor_low             ;register to update
2374   1453 EE                            out dx,al
2375   1454 428AC1                        inc dx ! mov al,cl
2376   1457 EE                            out dx,al
2377   1458 4A8A0E                        dec dx ! mov al,cursor_hi
2378   1458 EE                            out dx,al
2379   145C 428AC5                        inc dx ! mov al,ch
2380   145F EE                            out dx,al
2381                            sp_ret:
2382   1460 C3                          ret
2383
2384                            set_up_es:                              ;set up ES for subsequent
2385                            ;---------                              ;string operation
2386                            ;       entry:  BX = screen structure
2387                            ;               AH = virtual console number
2388                            ;       exit:   ES = physical video RAM segment if
2389                            ;                       foreground, else ES =
2390                            ;                       virtual console RAM for this screen
2391                            ;               CX = destroyed
2392
2393   1461 B90080                      mov cx,bw_video_seg
2394   1464 8EC1                        mov es,cx
2395   1466 3A263318                    cmp ah,foreground_screen
2396   146A 7405          1471          je set_r
```

```
2397
2398   146C 884F06                    mov cx,ss_screen_seg[bx]
2399   146F 8EC1                      mov es,cx
2400                          set_r:
2401   1471 C3                        ret
2402
2403                          ;*************************************************************************
2404                          ;*                                                                       *
2405                          ;*                          Switch Screen                                *
2406                          ;*                                                                       *
2407                          ;*************************************************************************
2408
2409                          ;=========
2410                          io_switch:
2411                          ;=========
2412                          ;       entry:  DL = Screen to switch to
2413                          ;       exit:   None
2414                          ;               ALL SEGMENT REGISTERS PRESERVED:
2415                          ;               CS,DS,ES,SS must be preserved though call
2416
2417                          ;       PIN process does range checking of DL
2418
2419   1472 06                        push es                         ;save UDA
2420
2421   1473 33C08AC2                  xor ax,ax | mov al,dl
2422   1477 8ADAB92C00                mov bl,dl | mov cx,ccblen
2423   147C F7E10306140C              mul cx | add ax,ccb             ;compute new foreground CCB
2424   1482 A33418                    mov foreground_ccb,ax
2425
2426   1485 861E3318                  xchg bl,foreground_screen       ;save destination screen
2427                                                                  ;and get screen to save
2428   1489 32FF                      xor bh,bh                       ;index into screen structure array
2429   148B D1E3                      shl bx,1                        ;word pointer
2430   148D 8B9F2B18                  mov bx,screen_struct_addrs[bx]  ;get pointer to old environment
2431   1491 B98007                    mov cx,screen_siz               ;in words
2432   1494 8E4706                    mov es,ss_screen_seg[bx]        ;destination segment
2433   1497 33F68BFE                  xor si,si | mov di,si
2434   149B B800B0                    mov ax,bw_video_seg
2435   149E 1E                        push ds                         ;save DS
2436   149F 8ED8                      mov ds,ax                       ;source segment
2437   14A1 F3A5                      rep movsw                       ;do the copy
2438   14A3 1F                        pop ds                          ;restore DS
2439
2440   14A4 8A1E3318                  mov bl,foreground_screen
2441
2442   14A8 32FF                      xor bh,bh                       ;requested one
2443   14AA D1E3                      shl bx,1
2444   14AC 8B9F2B18                  mov bx,screen_struct_addrs[bx]
2445   14B0 891E3618                  mov foreground_ss,bx            ;new foreground screen SS address
2446   14B4 33F68BFE                  xor si,si | mov di,si
2447   14B8 B800B0                    mov ax,bw_video_seg
2448   14BB 8EC0                      mov es,ax
2449   14BD B98007                    mov cx,screen_siz               ;in words
```

```
2450
2451   14C0 1E                          push ds
2452   14C1 8E5F06                      mov ds,ss_screen_seg[bx]
2453   14C4 F3A5                        rep movsw
2454   14C6 1F07                        pop ds ! pop es                 ;restore DDA
2455   14C8 8A263318                    mov ah,foreground_screen        ;param to set_physical_cursor
2456   14CC 50                          push ax                         ;save AH = foreground screen
2457   14CD F6470D04                    test ss_mode[bx],ssm_no_cursor  ;turn on/off cursor ?
2458   14D1 7505          14D8          jnz sw_no_cursor
2459   14D3 E8CAFE        13A0            call enable_cursor
2460   14D6 EB03          14DB            jmps sw_done
2461                            sw_no_cursor:
2462   14D8 E8C0FE        13A8            call disable_cursor
2463                            sw_done:
2464   14DB 58                          pop ax                          ;restore AH = foreground screen
2465   14DC E965FF        1444          jmp set_physical_cursor
2466   14DF C3                          ret
2467
2468                            ;*************************************************************************
2469                            ;*                                                                       *
2470                            ;*                        Status Line Routine                            *
2471                            ;*                                                                       *
2472                            ;*************************************************************************
2473
2474                            ;-------------
2475                            update_status:                  ;XIOS call for normal status update
2476                            ;-------------
2477   14E0 33C9                        xor cx,cx
2478
2479                            ;===========
2480                            io_statline:
2481                            ;===========
2482                            ;       entry:  CX = 0 regular status update
2483                            ;               else DX:CX->string to display
2484                            ;               CX = 0ffffh stop display of special status
2485                            ;               line, and start normal status line update
2486                            ;       exit:   AX<>0 if we couldn't print
2487                            ;               the requested status line: i.e., another
2488                            ;               process is already updating the status
2489                            ;               line, or a special status line is being
2490                            ;               displayed.
2491                            ;               ALL SEGMENT REGISTERS PRESERVED:
2492                            ;               CS,DS,ES,SS must be preserved though call
2493
2494   14E2 5152                        push cx ! push dx
2495                            s_wait:                         ;status line is not reentrant code
2496   14E4 B0FF                        mov al,true
2497   14E6 8606E818                    xchg al,sts_locked
2498   14EA 3CFF                        cmp al,true
2499   14EC 750A          14F8          jne s_own
2500   14EE B18D                        mov cl,p_delay                  ;wait one tick
2501   14F0 BA0100                      mov dx,1
2502   14F3 E878F7        0C71          call supif
```

```
2503
2504   14F6 74EC          14E4          je s_wait                       ;try again
2505                            s_own:
2506   14F8 5A59                          pop dx ! pop cx
2507   14FA 06                            push es                         ;save UDA
2508   14FB 85C9                          test cx,cx                      ;CX=0 normal status update
2509   14FD 7426          1525            jz s_norm
2510   14FF 83F9FF                          cmp cx,0ffffh
2511   1502 7507          150B              jne s_special
2512   1504 C606E91800                        mov sts_special,false
2513   1509 EB27          1532                jmps s_norm_ok
2514                            s_special:
2515   150B 803EE918FF                      cmp sts_special,true
2516   1510 7505          1518              jne s_special_ok                ;is status line busy with
2517   1512 B8FFFF                            mov ax,0ffffh                 ;an earlier special display ?
2518   1515 E91F01        1637                jmp s_exit
2519
2520                            s_special_ok:
2521   1518 C606E918FF                      mov sts_special,true
2522   151D 1E                              push ds                         ;save SYSDAT
2523   151E 8EDA                            mov ds,dx
2524   1520 8BF1                            mov si,cx
2525   1522 E9FE00        1623              jmp s_prt
2526                            s_norm:
2527   1525 803EE918FF                    cmp sts_special,true            ;if special line is currently
2528   152A 7505          1532            jne s_norm_ok                   ;being displayed skip normal
2529   152C B8FFFF                          mov ax,0ffffh
2530   152F E90501        1637              jmp s_exit                      ;update
2531                            s_norm_ok:
2532   1532 8CDA8EC2                      mov dx,ds ! mov es,dx
2533   1536 BFEA18                        mov di,offset status_msg_start
2534   1539 B020                          mov al,' '
2535   153B B95000                        mov cx,columns_per_screen
2536   153E F3AA                          rep stosb                       ;blank out local status string
2537
2538   1540 BE5A19                        mov si,offset constr            ;put in 'Console='
2539   1543 BFEA18                        mov di,offset smsg_con
2540   1546 B90800                        mov cx,length constr
2541   1549 F3A4                          rep movsb
2542
2543   154B BE6219                        mov si,offset prnstr            ;put in 'Printer='
2544   154E BF1519                        mov di,offset smsg_prn
2545   1551 B90800                        mov cx,length prnstr
2546   1554 F3A4                          rep movsb
2547
2548   1556 A03318                        mov al,foreground_screen
2549   1559 0430                          add al,'0'
2550   155B A2F218                        mov smsg_cnum,al                ;set console number
2551
2552   155E 881E3418                      mov bx,foreground_ccb           ;get process which owns the
2553   1562 8B37                          mov si,c_owner[bx]              ;foreground console
2554   1564 85F67413      1578            test si,si ! jz s_nopd          ;make sure its non-zero
2555   1568 8A4424                          mov al,p_list[si]
```

```
2556
2557   156B 0430A21D19                  add al,'0' ! mov sms...pnum,al
2558   1570 807408                    lea si,p_name[si]               ;offset of process name
2559   1573 BFFD18                    mov di,offset smsg_pd
2560   1576 B90400F3A5                mov cx,4 ! rep movsw
2561
2562                            s_nopd:
2563                            s_ctrlS:
2564   157B 88470E                    mov ax,c_state[bx]
2565   157E A980007408   1588       test ax,csm_ctrlS ! jz s_ctrlQ
2566   1583 C7050D195E53 1596         mov s_msgctrlS,'S^' ! jmps s_ctrlP
2567        EB03         1596
2568                            s_ctrlQ:
2569   158B A900017406   1596       test ax,csm_ctrlQ ! jz s_ctrlP  ;ctrl S and ctrl Q are mutally
2570   1590 C7060D195E4F              mov s_msgctrlS,'Q^'             ;exclusive print one of ^S or ^Q
2571                            s_ctrlP:
2572   1596 A900027412   15AD       test ax,csm_ctrlP ! jz s_mode
2573   159B C70610195E50              mov s_msgctrlP,'P^'
2574   15A1 B23D8A7708                mov dl,'=' ! mov dh,c_mimic[bx]
2575   15A6 80C630891612              add dh,'0' ! mov smsg_ctrlP_num,dx
2576        19
2577                            s_mode:
2578   15AD BFF418                    mov di,offset s_msgmode
2579   15B0 B90800                    mov cx,length dynstr
2580
2581   15B3 BE5219                    mov si,offset nosstr                    ;test for noswitch
2582   15B6 A908007513   15CE       test ax,csm_noswitch ! jnz s_movmode
2583   15BB BE3A19                    mov si,offset dynstr
2584   15BE A901007408   15CE         test ax,csm_buffered ! jz s_movmode   ;lsb bit=0: dynamic
2585   15C3 BE4219                      mov si,offset bufstr                ;lsb bit=1:buffered
2586   15C6 A904007403   15CE           test ax,csm_purging ! jz s_movmode  ;check for purging
2587   15CB BE4A19                        mov si,offset purstr
2588
2589                            s_movmode:
2590   15CE F3A4                      rep movsb
2591
2592                            s_getopenvec:
2593                                                                   ;display a letter for each drive
2594                                                                   ;with open files
2595   15D0 B91000                    mov cx,16                       ;loop count - 16 bits to check
2596   15D3 B041                      mov al,'A'                      ;A through P
2597   15D5 B406                      mov ah,6                        ;count of drive letters displayed
2598   15D7 88168800                  mov dx,open_vec                 ;BDOS sets this vector in SYSDAT
2599   15DB BF0619                    mov di,offset smsg_openvec
2600
2601                            open_nxt:
2602   15DE D1EA                      shr dx,1                        ;lowest bit is A drive, highest is P
2603   15E0 7305         15E7       jnc s_open_nxt                  ;no carry then no open files
2604   15E2 AA                        stosb                           ;store letter, incr DI
2605   15E3 FECC                      dec ah                          ;count letters displayed, 6 max
2606   15E5 7404         15EB         jz s_capslock                 ;used up drive display field
2607                            s_open_nxt:
2608   15E7 FEC0                      inc al                          ;next letter
```

```
2609
2610   15E9 E2F3          150E          loop open_nxt                    ;tested all 16 drive bits ?
2611
2612                                  s_capslock:
2613   15EB 8B1E3618                    mov bx,foreground_ss
2614   15EF 8A470D                      mov al,ss_mode[bx]
2615   15F2 A8207408      1601          test al,ssm_capslock ! jz s_numlock
2616   15F6 BE6A19                        mov si,offset capstr
2617   15F9 BF1F19                        mov di,offset smsg_capslock
2618   15FC B90800                        mov cx,length capstr
2619   15FF F3A4                          rep movsb
2620
2621                                  s_numlock:
2622   1601 A8107408      1610          test al,ssm_numlock ! jz s_wrap
2623   1605 BE7219                        mov si,offset numstr
2624   1608 BF2819                        mov di,offset smsg_numlock
2625   160B B90700                        mov cx,length numstr
2626   160E F3A4                          rep movsb
2627
2628                                  s_wrap:                          ;BX=foreground_ss
2629   1610 A802                        test al,ssm_no_wrap            ;if bit is set leave WRAP
2630   1612 7508          161F          jnz s_display                  ;field blank
2631   1614 BE7919                        mov si,offset wrpstr
2632   1617 BF3019                        mov di,offset smsg_wrap
2633   161A B90400                        mov cx,length wrpstr
2634   161D F3A4                          rep movsb
2635
2636                                  s_display:
2637   161F BEEA18                      mov si,offset status_msg_start
2638   1622 1E                          push ds                        ;save SYSDAT
2639                                  s_prt:
2640   1623 BF000F                      mov di,bw_video_status_line
2641   1626 B800B0                      mov ax,bw_video_seg
2642   1629 8EC0                        mov es,ax
2643   162B B409                        mov ah,09h                     ;underlined and enhanced attribute
2644   162D B95000                      mov cx,columns_per_screen
2645
2646                                  s_dloop:
2647   1630 AC                          lodsb                          ;get character
2648   1631 AB                          stosw                          ;display character and attribute
2649   1632 E2FC          1630          loop s_dloop
2650   1634 1F                          pop ds                         ;restore SYSDAT
2651   1635 33C0                        xor ax,ax
2652                                  s_exit:
2653   1637 07                          pop es                         ;restore UDA
2654   1638 C606E81800                  mov sts_locked,false
2655   163D C3                          ret
2656
2657                                  ;*************************************************************************
2658                                  ;*                                                                       *
2659                                  ;*                        List Device Routines                           *
2660                                  ;*                                                                       *
2661                                  ;*************************************************************************
```

```
2662
2663
2664                            ;=========
2665                            io_listst:                              ;list devices are on poll table
2666                            ;=========
2667                            ;       entry:  DL =    device
2668                            ;       exit:   AL =    0 if not ready
2669                            ;                       0ffh if ready
2670                            ;               BX =    device# * 2
2671                            ;               ALL SEGMENT REGISTERS PRESERVED:
2672                            ;               CS,DS,ES,SS must be preserved though call
2673
2674  163E 32F6D1E2                     xor dh,dh ! shl dx,1
2675  1642 8BDA                         mov bx,dx                             ;device # * 2
2676                                    ;jmps listst
2677
2678                            listst:                                 ;called from IO_POLL:
2679                            ;------                                 ;and IO_LISTST: above
2680                            ;       entry:  BX = device# * 2
2681                            ;               DL = device#
2682                            ;       exit:   AL = 0 if not ready
2683                            ;                       0ffh if ready
2684                            ;               BX = device# * 2
2685                            ;               DL presereved
2686
2687  1644 52                           push dx                               ;save device #
2688  1645 33C9                         xor cx,cx                             ;CX=0
2689  1647 8B97DF0F                     mov dx,list_ports[bx]                 ;port for list device 0
2690                                                                          ;parallel status
2691  1648 42EC                         inc dx ! in al,dx                     ;DX=status register, get status
2692  164D A880                         test al,80h
2693  164F 7401               1652      jz par_ret
2694  1651 49                             dec cx                              ;ready: CX=0ffffh
2695
2696                                par_ret:
2697  1652 8BC15A                       mov ax,cx ! pop dx
2698  1655 C3                           ret
2699
2700                            ;=======
2701                            io_list:
2702                            ;=======
2703                            ;       entry:  CL = character
2704                            ;               DL = device
2705                            ;       exit:   character sent
2706                            ;               ALL SEGMENT REGISTERS PRESERVED:
2707                            ;               CS,DS,ES,SS must be preserved though call
2708
2709  1656 5152                         push cx ! push dx                     ;save the character and device
2710  1658 E828F6             0C83      call poll                             ;wait for device to be ready
2711  165B 5A59                         pop dx ! pop cx                       ;device and character
2712                                                                          ;BX=device# * 2
2713  165D 8AC1                         mov al,cl                             ;AL = character
2714  165F 8B97DF0F                     mov dx,list_ports[bx]                 ;get the port for this device
```

```
2715
2716   1663 8532                    test dx,dx                      ;0 if no such device
2717   1665 7409          1670      jz lst_done
2718   1667 EE                        out dx,al                     ;send character
2719   1668 4242                      inc dx ! inc dx               ;reset it
2720   166A B00D                      mov al,0dh                    ;printer strobe on
2721   166C EE                        out dx,al
2722   166D B00C                      mov al,0ch                    ;printer strobe off
2723   166F EE                        out dx,al
2724                        lst_done:
2725   1670 C3                        ret
2726
2727                        ;*************************************************************************
2728                        ;*                                                                       *
2729                        ;*                        IO_AUXIN and IO_AUXOUT                         *
2730                        ;*                                                                       *
2731                        ;*************************************************************************
2732
2733                        ;========
2734                        io_auxin:
2735                        ;========
2736                        ;       entry:  none
2737                        ;       exit:   AL = character input
2738                        ;               ALL SEGMENT REGISTERS PRESERVED:
2739                        ;               CS,DS,ES,SS must be preserved though call
2740
2741   1671 C3                        ret
2742
2743                        ;=========
2744                        io_auxout:
2745                        ;=========
2746                        ;       entry:  CL character to output
2747                        ;       exit:   none
2748                        ;               ALL SEGMENT REGISTERS PRESERVED:
2749                        ;               CS,DS,ES,SS must be preserved though call
2750
2751   1672 C3                        ret
2752
```

```
2753
2754                            eject
2755                            ;                       Character I/O Data
2756                            ;                       ------------------
2757
2758                            ;***************************************************************
2759                            ;*                                                             *
2760                            ;*                      IO_CONIN DATA                          *
2761                            ;*                                                             *
2762                            ;***************************************************************
2763
2764     1673                   SL15    EQU     offset $
2765                            DSEG
2766                            ORG     SL15
2767
2768   1673 00                  scan_code       db      0               ;code returned by keyboard
2769
2770   1674 00                  down_bits       db      0               ;bit vector of
2771                                                                    ;action keys currently pressed
2772                                                                    ;bits representing toggle keys-
2773                                                                    ;numlock and capslock-are
2774                                                                    ;kept in the screen structures
2775                            ;       Action key scan codes
2776
2777     001D                   ctrl            equ     29              ;live action keys
2778     002A                   shft_left       equ     42
2779     0036                   shft_right      equ     54
2780     0038                   alt             equ     56              ;keys greater than ALT are
2781     003A                   capslock        equ     58              ;toggle action keys
2782     0045                   numlock         equ     69
2783
2784                            ;       Bit masks for special function keys,
2785                            ;       these values are masked into the DOWN_BITS bytes.
2786
2787     0001                   ctrl_bit        equ     001h
2788     0002                   shft_bit        equ     002h
2789     0008                   alt_bit         equ     008h
2790
2791     0006                   num_action_keys equ     6               ;number of entries in table
2792                                                                    ;below
2793                            action_key_table:
2794   1675 1D                          db      ctrl                    ;action key look up table
2795   1676 36                          db      shft_right
2796   1677 2A                          db      shft_left
2797   1678 38                          db      alt
2798   1679 45                          db      numlock
2799   167A 3A                          db      capslock
2800
2801                            action_key_masks:                       ;look up table for mask bits
2802   167B 01                          db      ctrl_bit                ;to use in DOWN_BITS or
2803   167C 02                          db      shft_bit                ;the SS_MODE byte in the
2804   167D 02                          db      shft_bit                ;foreground screen structure
2805   167E 08                          db      alt_bit
```

```
2806
2807   167F 10                          db      ssm_numlock             ;see screen structure definition
2808   1680 20                          db      ssm_capslock            ; "
2809
2810   1681 3B3C3D3E3F40        pfk_id_table    db      ';<=>?@abcdghjkmopqrs'
2811        616263646768
2812        69686D6F7071
2813        7273
2814
2815                            ;*************************************************************************
2816                            ;*                                                                       *
2817                            ;*              Keyboard Translation Tables                              *
2818                            ;*                                                                       *
2819                            ;*************************************************************************
2820
2821                            ;       Three keyboard translation tables follow.  The IBM PC returns
2822                            ;       a "scan code" from the keyboard which is used as an index into
2823                            ;       the following tables.
2824                            ;       KEY_TABLE contains ASCII for keys that have no other keys held down
2825                            ;       simultaneously.  The SHIFT_TABLE is for keys depressed when
2826                            ;       the shift, capslocks, or numslock keys are also down
2827                            ;       The CTRL_TABLE is for keys depressed when the CTRL
2828                            ;       is down.
2829
2830                            ;       0FFH in the translation table designates an illegal key code.
2831                            ;       The most significant bit is set for keys that are programmable
2832                            ;       or are fixed function keys, and also for the switch screen keys.
2833
2834                            key_table:
2835                                    ;       translation             keyboard scan code
2836                                    ;       -----------             ------------------
2837
2838   1695 FF                          db      0ffH                    ;0 - doesn't exist
2839   1696 1B                          db      esc                     ;1
2840   1697 313233343536                db      '1234567890-='          ;2-13 (1st row)
2841        37383930302D3D
2842   16A5 0809                        db      bs,ht                   ;14-15 (backspace, horizontal tab)
2843   16A7 717765727479                db      'qwertyuiop[]'          ;16-27 (2nd row)
2844        75696F705B5D
2845   16B5 0DFF                        db      cr, 0ffh                ;28-29 (carriage return)
2846   16B7 617364666768                db      'asdfghjkl;''`'         ;30-41 (3rd row)
2847        6A6B6C3B2760
2848   16C5 FF                          db      0ffh                    ;42 (left shift)
2849   16C6 5C7A78637662                db      '\zxcvbnm,./'           ;43-53 (4th row)
2850        6E6D2C2E2F
2851   16D1 FF                          db      0ffh                    ;54 (right shift)
2852   16D2 2A                          db      '*'                     ;55
2853   16D3 FF                          db      0ffh                    ;56 (alt)
2854   16D4 20                          db      ' '                     ;57 (space bar)
2855   16D5 FF                          db      0ffh                    ;58 (caps lock)
2856
2857                                                                    ;-function keys-
2858                                                                    ;-programmable-
```

```
2859
2860   16D0 80818283              db      80h,81h,82h,83h         ;59-62 (fucntion keys f1,f2,f3,f4)
2861   16D4 84858687              db      84h,85h,86h,87h         ;63-66 (function keys f5,f6,f7,f8)
2862   16D8 8889                  db      88h,89h                 ;67-68 (function keys f9,f10)
2863
2864   16DA FF                    db      0ffh                    ;69 (num lock)
2865   16DB 13                    db      dc3                     ;70 (scroll lock)
2866
2867                                                              ;- key pad -
2868                                                              ;-programmable-
2869   16DC 8A                    db      8ah                     ;71 (home)
2870   16DD 8B                    db      8bh                     ;72 (up arrow)
2871   16DE 8C                    db      8ch                     ;73 (Pg Up)
2872   16DF 2D                    db      '-'                     ;74
2873   16E0 8D                    db      8dh                     ;75 (left arrow)
2874   16E1 FF                    db      0ffh                    ;76
2875   16E2 8E                    db      8eh                     ;77 (right arror)
2876   16E3 2B                    db      '+'                     ;78
2877   16E4 8F                    db      8fh                     ;79 (end)
2878   16E5 90                    db      90h                     ;80 (down arrow)
2879   16E6 91                    db      91h                     ;81 (Pg Dn)
2880   16E7 92                    db      92h                     ;82 (Ins)
2881   16E8 7F                    db      7fh                     ;83 (Del)
2882
2883                        shift_table:
2884                               ;       translation             keyboard scan code
2885                               ;       -----------             ------------------
2886
2887   16E9 FF                    db      0ffH                    ;0 - doesn't exist
2888   16EA 1B                    db      esc                     ;1
2889   16EB 21402324255E          db      '!@#$%^&*()_+'          ;2-13 (1st row)
2890        262A28295F2B
2891   16F7 080C                  db      bs,ff                   ;14-15 (backspace, form feed)
2892   16F9 515745525459          db      'QWERTYUIOP{}'          ;16-27 (2nd row)
2893        55494F507B7D
2894   1705 0D                    db      cr                      ;28 (carriage return)
2895   1706 FF                    db      0ffh                    ;29 (ctrl)
2896   1707 415344464748          db      'ASDFGHJKL:"~'          ;30-41 (3rd row)
2897        4A4B4C3A227E
2898   1713 FF                    db      0ffh                    ;42 (left shift)
2899   1714 7C5A584356 42         db      '|ZXCVBNM<>?'           ;43-53 (4th row)
2900        4E4D3C3E3F
2901   171F FF                    db      0ffh                    ;54 (right shift)
2902   1720 10                    db      dle                     ;55 (Prt Sc - ^P)
2903   1721 FF                    db      0ffh                    ;56 (Alt)
2904   1722 20                    db      ' '                     ;57 (space bar)
2905   1723 FF                    db      0ffh                    ;58 (Caps Lock)
2906
2907                                                              ;-function keys-
2908                                                              ;-return fixed strings-
2909   1724 CBCCCDCE              db      203,204,205,206         ;59-62 (f1,f2,f3,f4)
2910   1728 CFD0D1D2              db      207,208,209,210         ;63-67 (f5,f6,f7,f8)
2911   172C D3D4                  db      211,212                 ;67-68 (f9,f10)
```

```
2912
2913   172E FF                          db      0ffh                    ;69 (Num Lock)
2914   172F 11                          db      dc1                     ;70 (Scroll Lock - ^Q)
2915
2916                                                                     ;-key pad-          .
2917   1730 3738392D3435                db      "789-456+1230."         ;71-83 (ascii values on keys)
2918        3628313233302E
2919        2E
2920
2921                            control_table:
2922                                    ;       translation             keyboard scan code
2923                                    ;       -----------             ------------------
2924
2925   173D FF                          db      0ffh                    ;0 (no such key code)
2926   173E 1B                          db      ESC                     ;1
2927   173F FF                          db      0ffh                    ;2
2928   1740 00                          db      NUL                     ;3 (ctrl 2)
2929   1741 FFFFFF                      db      0ffh,0ffh,0ffh          ;4-6
2930   1744 1E                          db      RDS                     ;7 (ctrl ^)
2931   1745 FFFFFFFF                    db      0ffh,0ffh,0ffh,0ffh     ;8-11
2932   1749 1F                          db      US                      ;12 (ctrl _)
2933   174A FF                          db      0ffh                    ;13
2934   174B 7F                          db      DEL                     ;14 (left arrow)
2935   174C FF                          db      0ffh                    ;15
2936   174D 11170512                    db      DC1,ETB,ENQ,DC2         ;16-19 (ctrl q,w,e,r)
2937   1751 14191509                    db      DC4,EM,NAK,HT           ;20-23 (ctrl t,y,u,i)
2938   1755 0F101B1D                    db      SHI,DLE,ESC,GS          ;24-27 (ctrl o,p,[,])
2939   1759 0D                          db      cr                      ;28 (ctrl carriage return)
2940   175A FF                          db      0ffh                    ;29
2941   175B 01130406                    db      SOH,DC3,EOT,ACK         ;30-33 (ctrl a,s,d,f)
2942   175F 07080A0B                    db      BEL,BS,LF,VT            ;34-37 (ctrl g,h,j,k)
2943   1763 0C                          db      FF                      ;38 (ctrl l)
2944   1764 FFFFFFFF                    db      0ffh,0ffh,0ffh,0ffh     ;39-42
2945   1768 1C1A1803                    db      FS,SUBB,CAN,ETX         ;43-47 (ctrl \,z,x,c)
2946   176C 16020E0D                    db      SYN,STX,SO,CR           ;47-50 (ctrl v,b,n,m)
2947   1770 FFFFFFFF                    db      0ffh,0ffh,0ffh,0ffh     ;51-54
2948   1774 10                          db      DLE                     ;55 (Prt Sc = ctrl P)
2949   1775 FF20FF                      db      0ffh," ",0ffh           ;56-58
2950
2951                                                                     ;-function keys
2952   1778 C1C2C3C4                    db      193,194,195,196         ;59-62 (f1,f2,f3,f4)
2953   177C C5C6C7C8                    db      197,198,199,200         ;63-66 (f5,f6,f7,f8)
2954   1780 C9CA                        db      201,202                 ;66-68 (f9,f10)
2955   1782 FF                          db      0ffh                    ;69
2956   1783 03                          db      ETX                     ;70 break (^scroll lock = ctrl c)
2957
2958                                                                     ;-key pad-
2959                                                                     ;-returns fixed escape sequences-
2960   1784 F7F8F9FF                    db      247,248,249,0ffh        ;7,8,9,-
2961   1788 F4F5F6FF                    db      244,245,246,0ffh        ;4,5,6,+
2962   178C F1F2F3                      db      241,242,243             ;1,2,3
2963   178F F0FF                        db      240,0ffh                ;0,.
2964
```

```
                        ;************************************************************
                        ;*                                                          *
                        ;*              Special Output Characters Tables            *
                        ;*                                                          *
                        ;************************************************************

1791 04                 special_char_tab        db      4               ;number of special chars
1792 0D0A0818                                   db      cr,lf,bs,esc    ;the special chars

1796 8711               special_func_tab        dw      carriage_return ;the special char routines
1798 9311                                       dw      line_feed
179A A611                                       dw      back_space
179C CD11                                       dw      escape

                        ;************************************************************
                        ;*                                                          *
                        ;*              Escape Sequence Tables                      *
                        ;*                                                          *
                        ;************************************************************

179E 1C                 esc_tab1        db      28              ;number of escape sequences
                                                                ;supported
179F 4843444241                         db      'H','C','D','B','A'    ;the second char
17A4 496A6B5945                         db      'I','j','k','Y','E'    ;in the escape
17A9 624A6C6F4B                         db      'b','J','l','o','K'    ;sequence
17AE 4C4D4E7071                         db      'L','M','N','p','q'
17B3 7275737465                         db      'r','u','s','t','e'
17B8 667677                             db      'f','v','w'
                                                                ;the escape sequence
                                                                ;routines
17BB F211               esc_func_tab1   dw      home            ;H
17BD FF11                               dw      right           ;C
17BF 0F12                               dw      left            ;D
17C1 1F12                               dw      down            ;B
17C3 3012                               dw      up              ;A

17C5 4112                               dw      up_with_scroll  ;T
17C7 5312                               dw      save            ;j
17C9 5F12                               dw      restore         ;k
17CB 6D12                               dw      x_and_y         ;Y - row and column
17CD E111                               dw      erase_screen    ;E

17CF 9E12                               dw      erase_begin     ;b
17D1 B212                               dw      erase_end       ;J
17D3 C812                               dw      erase_line      ;l
17D5 E112                               dw      erase_bol       ;o - BOL = begin of line
17D7 FE12                               dw      erase_eol       ;K - EOL = end of line

17D9 1113                               dw      insert_line     ;L
17DB 2A13                               dw      delete_line     ;M
17DD 4413                               dw      delete_char     ;N
17DF 6213                               dw      enter_reverse   ;p
```

```
3018
3019   17E1 8013                        dw      exit_reverse    ;q
3020
3021   17E3 9613                        dw      enter_bright    ;r
3022   17E5 9B13                        dw      exit_bright     ;u
3023   17E7 8C13                        dw      enter_blink     ;s
3024   17E9 9113                        dw      exit_blink      ;t
3025   17EB A013                        dw      enable_cursor   ;e
3026
3027   17ED A813                        dw      disable_cursor  ;f
3028   17EF B913                        dw      enable_wrap     ;v
3029   17F1 C013                        dw      disable_wrap    ;w
3030
3031                        ;*************************************************************************
3032                        ;*                                                                       *
3033                        ;*                        Screen Structures                              *
3034                        ;*                                                                       *
3035                        ;*************************************************************************
3036
3037                        ;       Each virtual console has a structure of the following
3038                        ;       format associated with it.  (SS = Screen Structure)
3039                        ;       The data in this structure is dependent on the type of screen
3040                        ;       supported and any escape sequence handling in IO_CONOUT.
3041                        ;       Note: SS_CURSOR is the byte offset of the cursor relative
3042                        ;       to the screen segment.  SS_ROW, SS_COLUMN are relative to 0
3043                        ;       and are word offsets, i.e., if SS_ROW is 0 and SS_COLUMN is 1,
3044                        ;       they refer to bytes 2 and 3 in the video RAM or a background
3045                        ;       screen's data area.
3046
3047     0000               ss_cursor       equ     word ptr 0                      ;points at data/attrib
3048     0002               ss_oldcursor    equ     word ptr ss_cursor + word
3049     0004               ss_escape       equ     word ptr ss_oldcursor + word    ;escape routine to return to
3050     0006               ss_screen_seg   equ     word ptr ss_escape + word       ;data for screen image
3051     0008               ss_xy           equ     word ptr ss_screen_seg + word   ;overlay row, col
3052     0008               ss_row          equ     byte ptr ss_xy                  ;current row
3053     0009               ss_column       equ     byte ptr ss_row + byte          ;current col
3054     000A               ss_oldxy        equ     word ptr ss_xy + word           ;overlay old row, col
3055     000A               ss_oldrow       equ     byte ptr ss_oldxy               ;old row
3056     000B               ss_oldcolumn    equ     byte ptr ss_oldxy + byte        ;old col
3057     000C               ss_attribute    equ     byte ptr ss_oldxy + word
3058     000D               ss_mode         equ     byte ptr ss_attribute + byte
3059     000E               ss_len          equ     ss_mode + byte
3060
3061                        ;       ss_mode byte bit values
3062
3063     0001               ssm_reverse     equ     00000001B       ;1 = if reverse video
3064                                                                ;0 = in normal
3065     0002               ssm_no_wrap     equ     00000010B       ;0 = wrap at EOL, 1 = discard
3066     0004               ssm_no_cursor   equ     00000100B       ;0 = cursor, 1 = cursor
3067     0008               ssm_log         equ     00001000B       ;0 = normal, 1 = data log
3068     0010               ssm_numlock     equ     00010000B       ;1 = numlock on
3069     0020               ssm_capslock    equ     00100000B       ;1 = capslock on
3070
```

```
3071
3072  17F3                    screen_structures       rb      0
3073
3074  17F3 00000000           ss0             dw      0,0     ;cursor, old cursor
3075  17F7 1411                               dw      offset co_no_escape
3076                                                    ;initially not in an escape sequence
3077  17F9 0000                               dw      0       ;screen_seg - set by INIT:
3078  17FB 0000                               db      0,0     ;row,column
3079  17FD 0000                               db      0,0     ;old row, old column
3080  17FF 07                                 db      07h     ;attribute
3081  1800 00                                 db      0       ;mode - initialize wrap,cursor,reverse off,
3082                                                    ;no data logging
3083
3084  1801 00000000           ss1             dw      0,0     ;cursor, old cursor
3085  1805 1411                               dw      offset co_no_escape
3086                                                    ;initially not in an escape sequence
3087  1807 0000                               dw      0       ;screen_seg - set by INIT:
3088  1809 0000                               db      0,0     ;row,column
3089  180B 0000                               db      0,0     ;old row, old column
3090  180D 07                                 db      07h     ;attribute
3091  180E 00                                 db      0       ;mode - initialize wrap,cursor,reverse off,
3092                                                    ;no data logging
3093
3094  180F 00000000           ss2             dw      0,0     ;cursor, old cursor
3095  1813 1411                               dw      offset co_no_escape
3096                                                    ;initially not in an escape sequence
3097  1815 0000                               dw      0       ;screen_seg - set by INIT:
3098  1817 0000                               db      0,0     ;row,column
3099  1819 0000                               db      0,0     ;old row, old column
3100  181B 07                                 db      07h     ;attribute
3101  181C 00                                 db      0       ;mode - initialize wrap,cursor,reverse off
3102                                                    ;no data logging
3103
3104  181D 00000000           ss3             dw      0,0     ;cursor, old cursor
3105  1821 1411                               dw      offset co_no_escape
3106                                                    ;initially not in an escape sequence
3107  1823 0000                               dw      0       ;screen_seg - set by INIT:
3108  1325 0000                               db      0,0     ;row,column
3109  1827 0000                               db      0,0     ;old row, old column
3110  1829 07                                 db      07h     ;attribute
3111  182A 00                                 db      0       ;mode - initialize wrap,cursor,reverse off
3112                                                    ;no data logging
3113
3114  182B F317               screen_struct_addrs     dw      offset ss0
3115  182D 0118                               dw      offset ss1
3116  182F 0F18                               dw      offset ss2
3117  1831 1D18                               dw      offset ss3
3118
3119                          ;*****************************************************************************
3120                          ;*                                                                           *
3121                          ;*                  Console Control Blocks                                   *
3122                          ;*                                                                           *
3123                          ;*****************************************************************************
```

```
3124
3125
3126   1833 00                  foreground_screen       db      0               ;console 0 is initial
3127   1834 3818                foreground_ccb          dw      offset ccb0     ;foreground console
3128   1836 F317                foreground_ss           dw      offset ss0
3129
3130   1838                     ccb_tab         rb      0
3131   1838 0000                ccb0            dw      0               ;owner
3132   183A 000000000000                        dw      0,0,0
3133   1840 FFFF                                db      0ffh, 0ffh      ;mimic, msource
3134   1842 00                                  db      0
3135   1843 00                                  db      0               ;virtual console number
3136   1844 0000                                dw      0
3137   1846 0000                                dw      0               ;foreground and dynamic
3138                                                                     ;be the foreground console
3139   1848 1000                                dw      10h             ;max buffer file size
3140   184A 000000000000                        dw      0,0,0,0,0,0
3141        000000000000
3142   1856 000000000000                        dw      0,0,0,0,0,0
3143        000000000000
3144   1862 0000                                dw      0
3145
3146   1864 0000                ccb1            dw      0               ;owner
3147   1866 000000000000                        dw      0,0,0
3148   186C FFFF                                db      0ffh, 0ffh      ;mimic, msource
3149   186E 00                                  db      0
3150   186F 01                                  db      1               ;virtual console number
3151   1870 0000                                dw      0
3152   1872 0200                                dw      csm_background  ;background and dynamic
3153   1874 1000                                dw      10h             ;max buffer file size
3154   1876 000000000000                        dw      0,0,0,0,0,0
3155        000000000000
3156   1882 000000000000                        dw      0,0,0,0,0,0
3157        000000000000
3158   188E 0000                                dw      0
3159
3160   1890 0000                ccb2            dw      0               ;owner
3161   1892 000000000000                        dw      0,0,0
3162   1898 FFFF                                db      0ffh, 0ffh      ;mimic, msource
3163   189A 00                                  db      0
3164   189B 02                                  db      2               ;virtual console number
3165   189C 0000                                dw      0
3166   189E 0200                                dw      csm_background  ;background and dynamic
3167   18A0 1000                                dw      10h             ;max buffer file size
3168   18A2 000000000000                        dw      0,0,0,0,0,0
3169        000000000000
3170   18AE 000000000000                        dw      0,0,0,0,0,0
3171        000000000000
3172   18BA 0000                                dw      0
3173
3174   18BC 0000                ccb3            dw      0               ;owner
3175   18BE 000000000000                        dw      0,0,0
3176   18C4 FFFF                                db      0ffh, 0ffh      ;mimic, msource
```

```
3177
3178   18C6 00                              db      0
3179   18C7 03                              db      3               ;virtual console number
3180   18C8 0000                            dw      0
3181   18CA 0200                            dw      csm_background  ;background and dynamic
3182   18CC 1000                            dw      1000            ;max buffer file size
3183   18CE 000000000000                    dw      0,0,0,0,0,0
3184        000000000000
3185   18DA 000000000000                    dw      0,0,0,0,0,0
3186        000000000000
3187   18E6 0000                            dw      0
3188
3189                                                                 ;non virtual console CCB
3190                        ;ccb4           dw      0               ;would look like this:
3191                        ;               dw      0,0,0
3192                        ;               db      0
3193                        ;               db      0
3194                        ;               db      0
3195                        ;               dw      0
3196                        ;               dw      0
3197                        ;               dw      0
3198                        ;               dw      0,0,0,0,0,0
3199                        ;               dw      0,0,0,0,0,0
3200                        ;               dw      0
3201
3202                        ;*****************************************************************************
3203                        ;*                                                                           *
3204                        ;*                          Status Line Data                                 *
3205                        ;*                                                                           *
3206                        ;*****************************************************************************
3207
3208   18E8 00              sts_locked      db      false   ;semaphore for status line code
3209   18E9 00              sts_special     db      false   ;is a special status line being displayed ?
3210
3211                        ;       The format of the status line is:
3212
3213                        ;Console=1 Buffered GENCCPM  ADDER ^S^P=0 Printer=2 CapsLock NumLock Wrap
3214                        ;                                  ^O
3215
3216   18EA                 status_msg_start        rb      0
3217   18EA                 smsg_con                rb      8               ;0-7 (Console=)
3218   18F2                 smsg_cnum               rb      1               ;8
3219   18F3 20                                      db      ' '             ;9
3220   18F4                 smsg_mode               rb      8               ;10-17
3221   18FC 20                                      db      ' '             ;18
3222   18FD                 smsg_pd                 rb      8               ;19-26
3223   1905 20                                      db      ' '             ;27
3224   1906                 smsg_openvec            rb      6               ;28-33
3225   190C 20                                      db      ' '             ;34
3226   190D                 smsg_ctrlS              rw      1               ;35-36
3227   190F                 smsg_ctrlO              rw      0               ;overlays ctrlS
3228   190F 20                                      db      ' '             ;37
3229   1910                 smsg_ctrlP              rw      1               ;38-39
```

```
3230
3231  1912                      smsg_ctrlP_num          rw      1               ;40-41
3232  1914 20                                           db      ' '             ;42
3233  1915                      smsg_prn                rb      8               ;43-50 (Printer=)
3234  191D                      smsg_pnum               rb      1               ;51
3235  191E 20                                           db      ' '             ;52
3236  191F                      smsg_capslock           rb      3               ;53-60
3237  1927 20                                           db      ' '             ;61
3238  1928                      smsg_numlock            rb      7               ;62-68
3239  192F 20                                           db      ' '             ;69
3240  1930                      smsg_wrap               rb      4               ;70-73
3241  1934                                              rb      6               ;74-79
3242
3243                            ;       String constants for status line
3244
3245  193A 44796E616D69         dynstr                  db      'Dynamic '      ;these
3246       6320
3247  1942 427566666572         bufstr                  db      'Buffered'      ;messages must
3248       6564
3249  194A 50757267696E         purstr                  db      'Purging '      ;be the same length
3250       6720
3251  1952 4E6F53776974         nosstr                  db      'NoSwitch'
3252       6368
3253
3254  195A 436F6E736F6C         constr                  db      'Console='
3255       653D
3256  1962 5072696E7465         prnstr                  db      'Printer='
3257       723D
3258  196A 436170734C6F         capstr                  db      'CapsLock'
3259       636B
3260  1972 4E756D4C6F63         numstr                  db      'NumLock'
3261       6B
3262  1979 57726170             wrpstr                  db      'Wrap'
3263
3264
3265                            ;*****************************************************************************
3266                            ;*                                                                           *
3267                            ;*                          List Control Blocks                              *
3268                            ;*                                                                           *
3269                            ;*****************************************************************************
3270
3271  197D                      lcb_tab         rb      0
3272  197D 000000000000        lcb0            dw      0,0,0,0         ;each LCB is 10 bytes long
3273       0000
3274  1985 00                                   db      0
3275  1986 FF                                   db      0ffh            ;msource
3276  1987 000000000000        lcb1            dw      0,0,0,0
3277       0000
3278  198F 00                                   db      0
3279  1990 FF                                   db      0ffh            ;msource
3280
```

```
3281
3282                        eject
3283                        ;                       DISK I/O
3284                        ;                       --------
3285
3286                        ;***************************************************************
3287                        ;*                                                             *
3288                        ;*                      IBM PC Disk Equates                    *
3289                        ;*                                                             *
3290                        ;***************************************************************
3291
3292                        ;       The following equates are set to the size of a double density,
3293                        ;       single sided 5 & 1/4" floppy.
3294
3295     0200               bytes_per_sector        equ     512
3296     0008               sectors_per_track       equ     8               ;1 to 8
3297     1000               bytes_per_track         equ     sectors_per_track * bytes_per_sector
3298     0028               tracks_per_disk         equ     40              ;0 to 39
3299     8000               bytes_per_disk          equ     tracks_per_disk * bytes_per_track
3300
3301                        ;       Memory disk support, drive M: is a RAM disk.
3302
3303     C000               m_disk_segment  equ     0C000h  ;segment base of mdisk RAM on IBM PC
3304
3305                        ;***************************************************************
3306                        ;*                                                             *
3307                        ;*              Intel 8272 FDC Equates                         *
3308                        ;*                                                             *
3309                        ;***************************************************************
3310
3311     03F4               fdc_stat                equ     03f4h           ;status port for the disk controller
3312     03F5               fdc_data                equ     fdc_stat+1      ;data port for the disk controller
3313     03F2               fdc_port                equ     03f2h           ;all bits clear on channel reset
3314
3315                                        ;7 6 5 4 3 2 1 0
3316                                        ;| | | | | | \_/
3317                                        ;| | | | | |  |
3318                                        ;| | | | | |  drive select 00=a,01=b,10=c,11=d
3319                                        ;| | | | | fdc reset*
3320                                        ;| | | | int & dmarq enable
3321                                        ;d c b a motor on
3322
3323     000C               fdc_on          equ     00001100b       ;mask to keep the 8272 unreset
3324     00FC               fdc_no_motor    equ     11111100b       ;mask for no motors
3325
3326     0066               fdc_read_cmd    equ     01100110b       ;mfm, skip deleted data, read
3327     0045               fdc_write_cmd   equ     01000101b       ;mfm, write
3328     004D               fdc_format_cmd  equ     01001101b       ;mfm, format
3329     000F               fdc_seek_cmd    equ     00001111b       ;seek
3330     0007               fdc_recal_cmd   equ     00000111b       ;home to track 0
3331     0008               fdc_si_cmd      equ     00001000b       ;sense interupt status
3332     0003               fdc_spec_cmd    equ     00000011b       ;specify
3333
```

```
3334                    ;
3335    0080            fdc_ready       equ     10000000b       ;mask for transfer ready
3336
3337    00CF            fdc_spec_1      equ     11001111b       ;srt=0c, hd unload=0f first specify byte
3338    0003            fdc_spec_2      equ     00000011b       ;hd load=1, mode=DMA second specify byte
3339
3340    0002            f_bytes         equ     2               ;magic number for 512 bytes per sector
3341    0008            f_sectors       equ     8               ;sectors per track
3342    003A            f_gap           equ     03ah            ;magic number for format gap
3343    00E5            f_filler        equ     0e5h            ;fill character
3344
3345    0002            r_bytes         equ     2               ;magic number for 512 bytes
3346    0008            r_sectors       equ     8
3347    002A            r_gap           equ     02ah
3348    00FF            r_dtl           equ     0ffh
3349
3350                    ;*****************************************************************
3351                    ;*                                                               *
3352                    ;*              d237 DMA Controller Port and Commands            *
3353                    ;*                                                               *
3354                    ;*****************************************************************
3355
3356    0000            dma_c0_address  equ     000h            ;8237 channel 0 address         rw
3357    0001            dma_c0_count    equ     001h            ;8237 channel 0 transfer count  rw
3358    0002            dma_c1_address  equ     002h            ;8237 channel 1 address         rw
3359    0003            dma_c1_count    equ     003h            ;8237 channel 1 transfer count  rw
3360    0004            dma_c2_address  equ     004h            ;8237 channel 2 address         rw
3361    0005            dma_c2_count    equ     005h            ;8237 channel 2 transfer count  rw
3362    0006            dma_c3_address  equ     006h            ;8237 channel 3 address         rw
3363    0007            dma_c3_count    equ     007h            ;8237 channel 3 transfer count  rw
3364    0008            dma_stat_reg    equ     008h            ;8237 status register           ro
3365    0008            dma_cmd_reg     equ     dma_stat_reg    ;8237 command register          wo
3366    0009            dma_requ_reg    equ     dma_stat_reg+1  ;8237 software dma request      wo
3367    000A            dma_bmsk_reg    equ     dma_stat_reg+2  ;8237 binary channel mask       wo
3368    000B            dma_mode_reg    equ     dma_stat_reg+3  ;8237 mode register             wo
3369    000C            dma_cbpf        equ     dma_stat_reg+4  ;8237 clear byte pointer f/f    wo
3370    000D            dma_temp_reg    equ     dma_stat_reg+5  ;8237 temporary register        ro
3371    000D            dma_clear       equ     dma_stat_reg+5  ;8237 master clear              wo
3372    000F            dma_mask_reg    equ     dma_stat_reg+7  ;8237 linear channel mask       wo
3373
3374    0080            dma_page_c1     equ     080h            ;a16 to a20 for channel 1
3375    0081            dma_page_fdc    equ     081h            ;a16 to a20 for channel 2
3376    0082            dma_page_c3     equ     082h            ;a16 to a20 for channel 3
3377
3378                    ;       The following labels define single mode, address increment
3379                    ;       auto-initialization disable, read or write using channel 2
3380
3381    004A            dma_mode_write_fdc      equ     01001010b
3382    0046            dma_mode_read_fdc       equ     01000110b
3383
3384    0002            dma_bmsk_fdc            equ     00000010b       ;binary channel mask for disk
3385
3386                    ;       DMA channel assignments
```

```
3387
3388
3389                    ;channel 0 dynamic memory refresh
3390                    ;channel 1
3391                    ;channel 2 floppy disk controller
3392                    ;channel 3
3393
3394                    ;*****************************************************************
3395                    ;*                                                               *
3396                    ;*              CCP/M Disk I/O Equates                           *
3397                    ;*                                                               *
3398                    ;*****************************************************************
3399
3400                    ;       Equates for parameter passing for read and write
3401                    ;       requests from the BDOS.
3402
3403                    ;       At the disk read and write function entries,
3404                    ;       all disk I/O parameters are on the stack.
3405                    ;       The stack at these entries appears as
3406                    ;       follows:
3407                    ;
3408                    ;           +-------+-------+
3409                    ;       +14 |  DRV  | MCNT  |   Drive and Multi sector count
3410                    ;           +-------+-------+
3411                    ;       +12 |     TRACK     |   Track number
3412                    ;           +-------+-------+
3413                    ;       +10 |    SECTOR     |   Physical sector number
3414                    ;           +-------+-------+
3415                    ;        +8 |    DMA_SEG    |   DMA segment
3416                    ;           +-------+-------+
3417                    ;        +6 |    DMA_OFF    |   DMA offset
3418                    ;           +-------+-------+
3419                    ;        +4 |    RET_SEG    |   BDOS return segment
3420                    ;           +-------+-------+
3421                    ;        +2 |    RET_OFF    |   BDOS return offset
3422                    ;           +-------+-------+
3423                    ;      SP+0 |    RET_ADR    |   Return address to XIOS ENTRY routine
3424                    ;           +-------+-------+
3425                    ;
3426                    ;       These parameters may be indexed and modified
3427                    ;       directly on the stack by the XIOS read and write rotines
3428                    ;       They will be removed by the BDOS when the XIOS completes
3429                    ;       the read/write function and returns to the BDOS.
3430
3431  000E              drive   equ     byte ptr 14[bp]
3432  000F              mcnt    equ     byte ptr 15[bp]
3433  000C              track   equ     word ptr 12[bp]
3434  000A              sector  equ     word ptr 10[bp]
3435  0008              dma_seg equ     word ptr 8[bp]
3436  0006              dma_off equ     word ptr 6[bp]
3437
3438                    ;       Some equtes in the Disk Parameter Header (DPH)
3439                    ;       and the Disk Parameter Block.
```

```
3440
3441
3442     0000                   xlt     equ     0       ;translation table offset in DPH
3443     0008                   dpb     equ     8       ;disk parameter block offset in DPH
3444     0000                   spt     equ     0       ;sectors per track offset in DPB
3445     000F                   psh     equ     15      ;physical shift factor offset in DPB
3446
3447                            ;*****************************************************************
3448                            ;*                                                               *
3449                            ;*                          IO_SELDSK                            *
3450                            ;*                                                               *
3451                            ;*****************************************************************
3452
3453     1991                   SL16    equ     offset $
3454                            CSEG
3455                            ORG     SL16
3456
3457                            ;=========
3458                            io_seldsk:          ; Function 7:  Select Disk
3459                            ;=========
3460                            ;       entry:  CL = disk to be selected
3461                            ;               DL = 00h if disk has not been previously selected
3462                            ;                  = 01h if disk has been previously selected
3463                            ;       exit:   AX = 0 if illegal disk
3464                            ;                  = offset of DPH relative from
3465                            ;                       XIOS Data Segment
3466                            ;               ALL SEGMENT REGISTERS PRESERVED:
3467                            ;               CS,DS,ES,SS must be preserved though call
3468
3469   1991 33DB                        xor bx,bx                       ;get ready for error
3470   1993 80F90F                      cmp cl,15                       ;is it a valid drive ?
3471   1996 7708            19A0        ja sel_ret                      ;if not just exit
3472   1998 8AD9                          mov bl,cl
3473   199A D1E3                          shl bx,1                        ;index into the DPH's
3474   199C 8B87180C                      mov ax,dph_tbl[bx]              ;get DPH address
3475                            sel_ret:
3476   19A0 C3                          ret
3477
3478                            ;*****************************************************************
3479                            ;*                                                               *
3480                            ;*                          IO_READ                              *
3481                            ;*                                                               *
3482                            ;*****************************************************************
3483
3484                            ;=======
3485                            io_read:            ; Function 11: Read sector
3486                            ;=======
3487                            ; Reads the sector on the current disk, track and
3488                            ; sector into the current DMA buffer.
3489
3490                            ;       entry:  parameters on stack
3491                            ;       exit:   AL = 00 if no error occured
3492                            ;               AL = 01 if an error occured
```

```
3493
3494                            ;               AL = 0ffh if density change detected
3495                            ;               ALL SEGMENT REGISTERS PRESERVED:
3496                            ;               CS,DS,ES,SS must be preserved though call
3497
3498   19A1 88EC                        mov bp,sp
3499   19A3 8A460E                      mov al,drive
3500   19A6 32E4                        xor ah,ah                      ;index into the physical driver
3501   19A8 8BF0                        mov si,ax
3502   19AA D1E6                        shl si,1
3503   19AC FFA4A91E                    jmp read_tbl[si]               ;jump to physical driver read routine
3504
3505                            read_m_disk:
3506                            ;-----------
3507   19B0 E81D00       19D0           call mdisk_calc                ;calculate byte address;
3508   19B3 06                          push es                        ;save UDA
3509   19B4 C47E06                      les di,dword ptr dmaoff        ;load destination DMA offset
3510                                                                   ;and segment
3511   19B7 33F6                        xor si,si                      ;setup source DMA address
3512   19B9 1E                          push ds                        ;save current DS
3513   19BA 8EDB                        mov ds,bx                      ;load pointer to sector in memory
3514   19BC F3A5                        rep movsw                      ;execute move of 128 bytes....
3515   19BE 1F                          pop ds                         ;then restore user DS register
3516   19BF 07                          pop es                         ;restore UDA
3517   19C0 33C0                        xor ax,ax                      ;return with good return code
3518   19C2 C3                          ret
3519
3520                            read_sd_floppy:
3521                            ;--------------
3522   19C3 C6069F1C46                  mov dma_mode_storage,dma_mode_read_fdc
3523   19C8 C606A01C66                  mov flp_rw_command,fdc_read_cmd
3524                                                                   ;set up for read before going
3525   19CD E94D00       1A1D           jmp flp_io                     ;to common read/write code
3526
3527                            mdisk_calc:
3528                            ;----------
3529                            ;       exit:   BX = sector paragraph address
3530                            ;               CX = length in words to transfer
3531                            ;       Assume MDISK OPB describes a disk with a physical
3532                            ;       sector size of 128, 8 sectors to a 1K track.
3533                            ;       Avoid deblocking by setting the logical sector size (128)
3534                            ;       equal to the physical sector size.
3535
3536   19D0 8B5E0C                      mov bx,track                   ;pickup track number
3537   19D3 B103                        mov cl,3                       ;times eight for relative sector
3538                                                                   ;number
3539   19D5 D3E3                        shl bx,cl
3540   19D7 8B4E0A                      mov cx,sector                  ;plus sector
3541   19DA 03D9                        add bx,cx                      ;gives relative sector number
3542   19DC B103                        mov cl,3                       ;times eight for paragraph of
3543                                                                   ;sector start
3544   19DE D3E3                        shl bx,cl
3545   19E0 81C300C0                    add bx,m_disk_segment          ;plus base address of disk in memory
```

```
3546
3547   19E4 8A460F                  mov al,mcnt
3548   19E7 32E4                    xor ah,ah                    ;multiply by 64, length of sector
3549   19E9 B106                    mov cl,6                     ;in words
3550   19EB D3E0                    shl ax,cl                    ;length * multi sector count
3551   19ED 8BC8                    mov cx,ax
3552   19EF FC                      cld
3553   19F0 C3                      ret
3554
3555                        ;*************************************************************************
3556                        ;*                                                                       *
3557                        ;*                              IO_WRITE                                 *
3558                        ;*                                                                       *
3559                        ;*************************************************************************
3560
3561                        ;========
3562                        io_write:               ; Function 12: Write disk
3563                        ;========
3564                        ; write the sector in the current Dma buffer to the current disk on the current
3565                        ; track in the current sector.
3566
3567                        ;       entry:  CL = 0 - Defered Writes
3568                        ;               CL = 1 - non-defered writes
3569                        ;               CL = 2 - def-wrt 1st sect unalloc blk
3570                        ;       exit:   AL = 00H if no error occured
3571                        ;               AL = 01H if error occured
3572                        ;               AL = 02H if read only disk
3573                        ;               AL = 0ffh if density change detected
3574                        ;               ALL SEGMENT REGISTERS PRESERVED:
3575                        ;               CS,DS,ES,SS must be preserved though call
3576
3577   19F1 8BEC                    mov bp,sp
3578   19F3 8A460E                  mov al,drive
3579   19F6 32E4                    xor ah,ah                    ;index into the physical driver
3580   19F8 8BF0                    mov si,ax
3581   19FA D1E6                    shl si,1
3582   19FC FFA4C91E                jmp write_tbl[si]            ;jump to physical driver write routine
3583
3584                        write_m_disk:
3585                        ;------------
3586   1A00 E8CDFF        19D0      call mdisk_calc              ;calculate byte address
3587   1A03 06                      push es                      ;save UDA
3588   1A04 8EC3                    mov es,bx                    ;setup destination DMA segment
3589   1A06 33FF                    xor di,di                    ;destination offset
3590   1A08 1E                      push ds                      ;save user segment register
3591   1A09 C57606                  lds si,dword ptr dmaoff      ;load source DMA offset
3592                                                             ;and segment
3593   1A0C F3A5                    rep movsw                    ;move from user to disk in memory
3594   1A0E 1F                      pop ds                       ;restore user segment pointer
3595   1A0F 07                      pop es                       ;restore UDA
3596   1A10 33C0                    xor ax,ax                    ;return no error
3597   1A12 C3                      ret
3598
```

```
3599
3600                            write_sd_floppy:
3601                            ;---------------
3602   1A13 C6069F1C4A                  mov dma_mode_storage,dma_mode_write_fdc
3603   1A18 C6065A01C45                 mov flp_rw_command,fdc_write_cmd
3604                                    ;jmps flp_io                    ;set up for write before
3605                                                                    ;going to common read/write code
3606
3607                            flp_io:
3608                            ;------
3609                            ;       entry:  parameters on stack
3610                            ;       exit:   zero flag set if successful
3611
3612                            ;       Common code for floppy read and write
3613                            ;       Note the parameters on the stack, SECTOR, TRACK are
3614                            ;       0 relative, and NCNT, SECTORS_PER_TRACK are relative to 1
3615
3616   1A1D C606A71CFF                  mov disk_busy,true              ;keep I_TICK from turning off motor
3617   1A22 B80800                      mov ax,sectors_per_track        ;max sectors - starting sector =
3618   1A25 28450A                      sub ax,sector                   ;sectors left on track
3619   1A28 32FF                        xor bh,bh
3620   1A2A 8A5E0F                      mov bl,mcnt                     ;multi sector count is byte variable
3621   1A2D 38C3                        cmp ax,bx                       ;sectors left on track, sectors
3622                                                                    ;requested
3623   1A2F 7603              1A34      jbe flp_track                   ;transfer to end of track
3624   1A31 8A460F                        mov al,mcnt                   ;transfer to before end of track
3625
3626                            flp_track:
3627   1A34 28460F                      sub mcnt,al                     ;AL = # sectors to R/W on this
3628                                                                    ;iteration
3629                                                                    ;through FLP_IO,
3630   1A37 A2A21C                      mov track_mcnt,al               ;sectors to R/W on current track
3631
3632                                                                    ;check for 64K page overlap
3633   1A3A 8AE0                        mov ah,al                       ;shl al,8 (* 256)
3634   1A3C 32C0                        xor al,al
3635   1A3E D0E4                        shl ah,1                        ;* 512
3636   1A40 50                          push ax                         ;how many bytes to R/W on cur trk
3637
3638   1A41 8B4508                      mov ax,dma_seg                  ;compute new 20 bit DMA addr
3639   1A44 8B5E06                      mov bx,dma_off
3640   1A47 E8C300            1B0D      call comp_dma                   ;returns AX = low 16 bits of 20 bit adr
3641   1A4A F7D0                        not ax                          ;how many bytes left in this 64K page
3642
3643   1A4C 5B                          pop bx                          ;BX=bytes to R/W on current track
3644   1A4D 38C3                        cmp ax,bx                       ;does this transfer fit in 64K page
3645   1A4F 7203              1A54      jb flp_end_64K
3646   1A51 E98700            1ADB        jmp flp_pg_ok
3647
3648                                flp_end_64K:
3649                                                                    ;read to end of 64K page and then
3650                                                                    ;read spanning sector locally
3651   1A54 8AC4                          mov al,ah                    ;how many sectors fit in this page ?
```

```
3652
3653  1A56 32E4                          xor ah,ah                       ;bytes left in 64K page
3654  1A58 D1E8                          shr ax,1                        ;divided by 512
3655  1A5A 85C0                          test ax,ax                      ;if 0, no sectors fit
3656  1A5C 741E          1A7C            jz flp_rwlocal                  ;in the rest of this 64K page
3657
3658  1A5E A2A11C                          mov num_sec,al                ;sectors that fit in end 64K page
3659  1A61 2806A21C                        sub track_mcnt,al             ;track_mcnt always > AL
3660  1A65 E8B000        1B25              call flp_phys_io              ;read/write them, DMA already computed
3661  1A68 7403          1A6D              jz flp_end64k_ok
3662  1A6A E99A00        1B07                jmp flp_ret                 ;zero flag is reset for error return
3663
3664                               flp_end64K_ok:
3665  1A6D 33C0                            xor ax,ax
3666  1A6F A0A11C                          mov al,num_sec                ;compute new DMA offset
3667  1A72 01460A                          add sector,ax                 ;still on the same track
3668  1A75 86E0                            xchg ah,al                    ;shl ax,8 (* 256)
3669  1A77 D0E4                            shl ah,1                      ;* bytes_per_sector (512)
3670  1A79 014606                          add dma_off,ax                ;64K wrap around is legal
3671
3672                               flp_rwlocal:                          ;read into XIOS data segment
3673                                                                     ;the spanning
3674  1A7C BBA91C                        mov bx,offset local_buf         ;sector
3675  1A7F 8CD8                          mov ax,ds                       ;compute 20 bit local DMA addr
3676  1A81 E88900        1B0D            call comp_dma
3677  1A84 803EA01C66                    cmp flp_rw_command,fdc_read_cmd
3678                                                                     ;reading or writing ?
3679  1A89 741C          1AA7            je flp_local
3680  1A8B 8B7606                          mov si,dma_off                ;get the sector to write from local
3681  1A8E BFA91C                          mov di,offset local_buf       ;buffer
3682  1A91 061E07                          push es ! push ds ! pop es
3683  1A94 8B4608ED8                       mov ax,dma_seg ! mov ds,ax
3684  1A99 B90001                          mov cx,bytes_per_sector/2
3685  1A9C F3A5                            rep movsw
3686  1A9E 072E8E1E060C                    pop es ! mov ds,sysdat
3687  1AA4 897606                          mov dma_off,si                ;update DMA offset
3688
3689                               flp_local:
3690  1AA7 C606A11C01                    mov num_sec,1                   ;read/write one sector
3691  1AAC E87600        1B25            call flp_phys_io                ;XIOS data cannot span 64K page
3692  1AAF 7556          1B07            jnz flp_ret                     ;return error
3693  1AB1 FF450AFE0EA2                    inc sector ! dec track_mcnt
3694       1C
3695
3696  1AB8 803EA01C66                      cmp flp_rw_command,fdc_read_cmd
3697  1ABD 7513          1AD2              jne flp_local_done            ;reading or writing ?
3698  1ABF 8B7E06                            mov di,dma_off              ;move the sector to user's area
3699  1AC2 BEA91C                            mov si,offset local_buf     ;if reading
3700  1AC5 068E4608                          push es ! mov es,dma_seg
3701  1AC9 B90001                            mov cx,bytes_per_sector/2
3702  1ACC F3A5                              rep movsw
3703  1ACE 897E06                            mov dma_off,di              ;update DMA offset
3704  1AD1 07                                pop es
```

```
3705
3706
3707                            flp_local_done:
3708   1AD2 8B4608                      mov ax,dma_seg          ;compute new 20 bit DMA addr
3709   1AD5 8B5E06                      mov bx,dma_off          ;for next FDC read/write
3710   1AD8 E83200        1B0D          call comp_dma
3711
3712                            flp_pg_ok:                       ;read will not cross 64K boundary
3713   1ADB A0A21C                      mov al,track_mcnt       ;could be 0 if we just read locally
3714   1ADE 84C07413      1AF5          test al,al ! jz nxt_track
3715   1AE2 A2A11C                        mov num_sec,al        ;read the rest from this track
3716   1AE5 E83D00        1B25            call flp_phys_io      ;DMA is already computed
3717   1AE8 751D          1B07            jnz flp_ret           ;return error, zero flag reset
3718   1AEA 33C0                            xor ax,ax
3719   1AEC 8A26A11C                        mov ah,num_sec      ;shl num_sec,8 (* 256)
3720   1AF0 D0E4                            shl ah,1            ;* 512
3721   1AF2 014606                          add dma_off,ax
3722
3723                            nxt_track:
3724   1AF5 32C0                        xor al,al
3725   1AF7 38460F                      cmp mcnt,al
3726   1AFA 7408          1B07          jz flp_ret              ;sucessful return with zero flag on
3727   1AFC FF460C                        inc track
3728   1AFF C7460A0000                    mov sector,0
3729   1B04 E916FF        1A1D            jmp flp_io            ;more IO to do on next track
3730
3731                            flp_ret:
3732   1B07 C606A71C00                  mov disk_busy,false
3733   1B0C C3                          ret
3734
3735                            comp_dma:       ;Compute 20 bit address from offset, segment
3736                            ;--------
3737
3738                            ;       entry:  AX =    segment
3739                            ;               BX =    offset
3740                            ;       exit:   AX =    low 16 bits
3741                            ;               CH =    highest 4 bits of address, always less than 16 -
3742                            ;                       no megabyte wrap around
3743                            ;
3744                            ;               The XIOS variables DMA_LOW16 and DMA_HIGH4 are
3745                            ;               set by this routine.  These variables are transferred
3746                            ;               to the floppy disk controller by the routine DMA_SET_UP.
3747
3748   1B0D B104D3C0                    mov cl,4 ! rol ax,cl    ;make paragraphs into bytes
3749   1B11 8AE824F0                    mov ch,al ! and al,0f0h ;save high 4 bits, 0 low 4 bits
3750   1B15 03C3                        add ax,bx               ;add byte offset
3751   1B17 80D50080E50F                adc ch,0 ! and ch,0fh   ;add in the carry, page is less than
3752   1B1D A3A31C                      mov dma_low16,ax        ;16
3753   1B20 882EA51C                    mov dma_high4,ch
3754   1B24 C3                          ret
3755
3756                            flp_phys_io:
3757                            ;----------
```

```
3758
3759                            ;       entry:  num_sec = number of sectors to read in this
3760                            ;               operation
3761                            ;               disk parameters on the stack
3762                            ;       exit:   zero flag set if ok
3763
3764                            ;       Perform physical actions to read/write floppy diskette
3765
3766   1825 C6068F1C02                  mov d_a_number,r_bytes
3767   182A C606901C08                  mov d_a_eot,r_sectors
3768   182F C606911C2A                  mov d_a_gpl,r_gap
3769   1834 C606921CFF                  mov d_a_dtl,r_dtl
3770   1839 E82700          1863        call motor_on                   ;make sure motor is on
3771   183C C606881C0A                  mov recals,recal_max
3772                               recal_loop:
3773   1841 C606871C05                  mov retries,retry_max
3774                               retry_loop:
3775   1846 E86600          18AF        call seek
3776   1849 E8A000          18EC        call dma_setup
3777   184C E8C600          1C15        call fdc_read_write
3778   184F 7411            1862        jz phys_ret                     ;if errors
3779   1851 FE0E871C                      dec retries                     ;attempt retries
3780   1855 75EF            1846        jnz retry_loop
3781   1857 E83900          1893          call recal                    ;if retries fail
3782   185A FE0E881C                      dec recals                      ;recalabrate the drive
3783   185E 75E1            1841          jnz recal_loop
3784   1860 0C01                            or al,1                       ;give up and return error
3785                               phys_ret:
3786   1862 C3                          ret
3787
3788                               motor_on:
3789                               ;--------
3790                            ;       entry:  none
3791                            ;       exit:   none
3792
3793                            ;       Turn on the motor if it is off.
3794
3795   1863 C606A81CFF                  mov motor_off_counter,0ffh      ;set for T_TICK timeout
3796   1868 B010                        mov al,010h                     ;pick up a bit for motor a
3797   186A 8A4E0E                      mov cl,drive                    ;fetch the binary drive code
3798   186D D2E0                        shl al,cl                       ;make it a bit for motor x
3799   186F 8A26A61C                    mov ah,motor_flags              ;fetch the motor bits
3800   1873 84E0                        test ah,al                      ;check to see if its on
3801   1875 7518            1892        jnz motor_on_done               ;yes then leap
3802   1877 0C0C                          or al,fdc_on                  ;mask in the no reset,enable interupt
3803   1879 0A460E                        or al,drive                   ;mask in the drive
3804   187C A2A61C                        mov motor_flags,al            ;save for later
3805   187F BAF203                        mov dx,fdc_port               ;point to the port number
3806   1882 EE                            out dx,al                     ;select & motor on
3807   1883 803EA01C66                    cmp flp_rw_command,fdc_read_cmd
3808   1888 7408            1892          je motor_on_done
3809   188A BA0700                          mov dx,ticks_per_second/8   ;wait for motor on 1/8 second
3810   188D B18D                            mov cl,p_delay              ;when writing
```

```
3811
3812   1B8F E8DFF0       0C71              call supif
3813
3814                                                          ;When initially debugging
3815                           ;          xor cx,cx          ;do not use the P_DELAY function
3816                           ;m_wait:                      ;until P_TICK is implemented;
3817                           ;          loop m_wait        ;use a delay loop like this.
3818
3819                           motor_on_done:
3820   1B92 C3                           ret
3821
3822                           recal:
3823                           ;-----
3824                           ;       entry:  none
3825                           ;       exit:   none
3826
3827                           ;       Move the head to home on the selected disk drive.
3828
3829   1B93 C606891C02                   mov disk_arguments,2          ;specify number of arguments
3830   1B98 C6068A1C07                   mov d_a_command,fdc_recal_cmd
3831   1B9D 8A460E                       mov al,drive                  ;get current disk
3832   1BA0 A28B1C                       mov d_a_drive,al              ;set up the command block
3833   1BA3 E8A700       1C4D            call fdc_command_put          ;go send the command block
3834   1BA6 8A0400                       mov dx,fdc_flag
3835   1BA9 E8C3F0       0C6F            call waitflag                 ;wait for FDC interrupt
3836                           rc_end:
3837   1BAC E98800       1C37            jmp sense_interrupt           ;required after RECAL command
3838                                     ;ret
3839
3840                           seek:
3841                           ;----
3842                           ;       entry:  TRACK to seek to on the stack relative to BP
3843                           ;       exit:   none
3844
3845   1BAF C605891C03                   mov disk_arguments,3          ;request 3 byte command transfer
3846   1BB4 C6068A1C0F                   mov d_a_command,fdc_seek_cmd
3847   1BB9 32DB                         xor bl,bl                     ;select head 0
3848   1BBB 8B450C                       mov ax,track                  ;get track off stack
3849
3850                                                          ;code for double sided diskettes
3851                                                          ;would go here ...
3852                           ;       cmp al,40             ;test for off side 1
3853                           ;       jb side_ok            ;if single sided then done
3854                           ;         mov bl,1            ;else specify head 1
3855                           ;         mov ah,79           ;compute track on second side
3856                           ;         sub ah,al
3857                           ;         xchg ah,al
3858
3859                           side_ok:
3860   1BBE 881E8D1C                     mov d_a_head,bl               ;set up the head
3861   1BC2 02DB                         add bl,bl                     ;multiply by 2
3862   1BC4 02DB                         add bl,bl                     ;multiply by 4
3863   1BC6 A28C1C                       mov d_a_cylinder,al           ;set up the track number
```

```
3864
3865   18C9 0A5E0E                    or bl,drive                     ;bits 0,1 are drive number
3866   18CC 881E8B1C                  mov d_a_drive,bl                ;set up disk number
3867   18D0 8B460A                    mov ax,sector                   ;get sector off stack
3868   18D3 40                        inc ax                          ;increment it
3869   18D4 A28E1C                    mov d_a_record,al               ;and set it up
3870   18D7 0206A11C                  add al,num_sec                  ;compute new end of track
3871   18DB FEC8                      dec al
3872   18DD A2901C                    mov d_a_eot,al                  ;put last sector in params
3873   18E0 E86A00       1C40         call fdc_command_put            ;send the command block
3874   18E3 BA0400                    mov dx,fdc_flag
3875   18E6 E886F0       0C6F         call waitflag                   ;wait for FDC interrupt
3876
3877                          st_end:
3878   18E9 E94800       1C37         jmp sense_interrupt             ;required after SEEK command
3879                                  ;ret
3880
3881                          dma_setup:
3882                          ;---------
3883                          ;       entry:  DMA_MODE_STORAGE, DMA_LOW16, DMA_HIGH4 set up
3884                          ;       exit:   none
3885
3886                          ;       Set the DMA device up for a read/write operation.
3887                          ;       The current DMA command word must in DMA_MODE_STORAGE.
3888                          ;       DMA_LOW16 and DMA_HIGH4 are the twenty bit starting address.
3889                          ;       The read/write operation cannot cross a physical 64K boundary.
3890
3891   18EC E60C                      out dma_cbpf,al                 ;reset the byte pointer
3892
3893   18EE A09F1C                    mov al,dma_mode_storage         ;get the mode byte
3894   18F1 E60B                      out dma_mode_reg,al             ;set the mode
3895   18F3 A1A31C                    mov ax,dma_low16                ;low 16 bits of 20 bit DMA address
3896   18F6 E604                      out dma_c2_address,al           ;send low 8 bits
3897   18F8 8AC4                      mov al,ah
3898   18FA E604                      out dma_c2_address,al           ;send next 8 bits
3899   18FC A0A51C                    mov al,dma_high4                ;high 4 bits of 20 bit DMA address
3900   18FF E681                      out dma_page_fdc,al
3901   1C01 33C0                      xor ax,ax
3902   1C03 8A26A11C                  mov ah,num_sec                  ;shl num_sec,8 (* 256)
3903   1C07 D0E4                      shl ah,1                        ;*512
3904   1C09 48                        dec ax                          ;0 relative
3905   1C0A E605                      out dma_c2_count,al             ;set up the low byte
3906   1C0C 8AC4                      mov al,ah                       ;get the low byte
3907   1C0E E605                      out dma_c2_count,al             ;and the high byte
3908   1C10 B002                      mov al,dma_bmsk_fdc             ;get the binary channel mask
3909   1C12 E60A                      out dma_bmsk_reg,al             ;enable the disk channel
3910   1C14 C3                        ret
3911
3912                          fdc_read_write:
3913                          ;--------------
3914                          ;       entry:  DMA device set up, head positioned on correct
3915                          ;               track
3916                          ;       exit:   zero flag set if successful
```

```
3917
3918
3919                            ;       Send read or write command to 8272 FDC
3920
3921   1C15 C606891C09                  mov disk_arguments,9            ;9 byte command for read or write
3922   1C1A A0A01C                      mov al,flp_rw_command           ;get read or write command
3923   1C1D A28A1C                      mov d_a_command,al              ;put it in the command string
3924   1C20 E82A00        1C4D          call fdc_command_put            ;send the command to the FDC
3925   1C23 BA0400                      mov dx,fdc_flag
3926   1C26 E846F0        0C6F          call waitflag                   ;wait for FDC interrupt
3927   1C29 C606931C07                  mov disk_results,7              ;7 byte result transfer
3928   1C2E E83400        1C65          call fdc_status_get             ;get the result bytes
3929   1C31 F606941CC0                  test d_r_st0,0C0H               ;test status register 0
3930   1C36 C3                          ret                             ;return zero flag set on success
3931
3932                                sense_interrupt:
3933                                ;---------------
3934                                ;       Sense interrupt command on the FDC.  It is called
3935                                ;       after a recal or a seek to test for seek complete.
3936
3937   1C37 C606891C01                  mov disk_arguments,1            ;only one byte of command
3938   1C3C C6058A1C08                  mov d_a_command,fdc_si_cmd      ;sense interrupt command
3939   1C41 E80900        1C4D          call fdc_command_put            ;send the command to the FDC
3940   1C44 C606931C02                  mov disk_results,2              ;2 bytes are returned
3941   1C49 E81900        1C65          call fdc_status_get             ;get the 2 result bytes
3942   1C4C C3                          ret
3943
3944                                fdc_command_put:
3945                                ;---------------
3946                                ;       entry:  DISK_ARGUMENT array set up
3947                                ;       exit:   none
3948
3949                                ;       Send the command block in the DISK_ARGUMENTS table to
3950                                ;       the 8272 FDC.
3951                                ;       The number of commands to write to the FDC is the
3952                                ;       first item in the table.
3953
3954   1C4D BAF403                      mov dx,fdc_stat                 ;point to the i/o port
3955   1C50 BE891C                      mov si,offset disk_arguments    ;point to the table of arguments
3956   1C53 FC                          cld                             ;make sure we go forward
3957   1C54 AC                          lodsb                           ;get the length of the arguments table
3958   1C55 8AC8                        mov cl,al                       ;get it into the count register
3959   1C57 2AED                        sub ch,ch                       ;zero the high byte
3960
3961                                fdc_command_loop:
3962   1C59 EC                          in al,dx                        ;get the current control byte
3963   1C5A A880                        test al,fdc_ready               ;if not ok to send next byte
3964   1C5C 74FB          1C59          jz fdc_command_loop             ;then loop waiting
3965   1C5E 42                            inc dx                        ;point at the data port
3966   1C5F AC                            lodsb                         ;else get the byte to send
3967   1C60 EE                            out dx,al                     ;send it
3968   1C61 4A                            dec dx                        ;point back at the status port
3969   1C62 E2F5          1C59            loop fdc_command_loop         ;if not last byte then loop
```

```
3970
3971   1C64 C3                          ret                             ;else we're all done
3972
3973                            fdc_status_get:
3974                            ;--------------
3975                            ;       entry:  number of results in 1st byte of DISK_RESULTS array
3976                            ;       exit:   none
3977
3978                            ;       Get the status information from the 8272
3979                            ;       FDC and place them in the table at DISK_RESULTS.
3980                            ;       The first byte in the table is the number of results
3981                            ;       to read from the FDC
3982
3983   1C65 06                          push es                         ;save UDA
3984   1C66 8AF403                      mov dx,fdc_stat                 ;point at the status port
3985   1C69 8CD8                        mov ax,ds                       ;get our data segment
3986   1C6B 8EC0                        mov es,ax                       ;into the extra segment
3987   1C6D BF941C                      mov di,offset disk_results + 1  ;point to where to put the data
3988   1C70 FC                          cld                             ;make sure we go forward
3989   1C71 8A0E931C                    mov cl,disk_results             ;fetch the number of expected results
3990   1C75 2AED                        sub ch,ch                       ;zero the high byte
3991
3992                            fdc_status_loop:
3993   1C77 EC                          in al,dx                        ;get the current control byte
3994   1C78 A880                        test al,fdc_ready               ;if not ok to read next byte
3995   1C7A 74FB          1C77          jz fdc_status_loop              ;then loop waiting
3996   1C7C 42                            inc dx                        ;point at the data port
3997   1C7D EC                            in al,dx                      ;get the byte
3998   1C7E AA                            stosb                         ;put it in the structure
3999   1C7F 4A                            dec dx                        ;point back at the status port
4000   1C80 E2F5          1C77            loop fdc_status_loop          ;if not last then loop
4001   1C82 07                          pop es                          ;restore UDA
4002   1C83 C3                          ret                             ;else return
4003
4004                            ;*****************************************************************************
4005                            ;*                                                                           *
4006                            ;*                          IO_FLUSHBUF                                      *
4007                            ;*                                                                           *
4008                            ;*****************************************************************************
4009
4010                            ;===========
4011                            io_flushbuf:                    ;Flush Buffer
4012                            ;===========
4013                            ;       entry:  None
4014                            ;       exit:   AL = 00h if no error occurs
4015                            ;                  = 01h if error occurs
4016                            ;                  = 02h if read/only disk
4017                            ;               ALL SEGMENT REGISTERS PRESERVED:
4018                            ;               CS,DS,ES,SS must be preserved though call
4019
4020
4021   1C84 32C0                        xor al,al       ;no need to flush buffer with
4022   1C86 C3                          ret             ;no blocking/deblocking in XIOS
```

```
4023
4024
4025                    ;************************************************************************
4026                    ;*                                                                      *
4027                    ;*                        Disk I/O Data                                 *
4028                    ;*                                                                      *
4029                    ;************************************************************************
4030
4031    1C87            SL17    equ     offset $
4032                    DSEG
4033                    ORG     SL17
4034
4035    0005            retry_max       equ             5
4036    000A            recal_max       equ             10
4037    1C87 00         retries         db              0
4038    1C88 00         recals          db              0
4039
4040                    ;       The following tables are used to issue commands to,
4041                    ;       and read results from the 8272 FDC.  The first entry
4042                    ;       in each table is the number of bytes to send or receive
4043                    ;       from the device
4044
4045    1C89 00         disk_arguments  db      0               ;number of arguments to send
4046    1C8A 00         d_a_command     db      0               ;command read/write
4047    1C8B 00         d_a_drive       db      0               ;drive select & head select
4048    1C8C 00         d_a_cylinder    db      0               ;cylinder to read/write
4049    1C8D 00         d_a_head        db      0               ;head
4050    1C8E 00         d_a_record      db      0               ;sector
4051    1C8F 02         d_a_number      db      2               ;magic number for 512 bytes/sector
4052    1C90 08         d_a_eot         db      sectors_per_track       ;end of track sector number
4053    1C91 2A         d_a_gpl         db      02ah            ;inter sector gap length
4054    1C92 FF         d_a_dtl         db      0ffh            ;data length
4055
4056    1C93 00         disk_results    db      0               ;number of bytes to read
4057    1C94 00         d_r_st0         db      0               ;status byte 0
4058    1C95 00         d_r_st1         db      0               ;status byte 1
4059    1C96 00         d_r_st2         db      0               ;status byte 2
4060    1C97 00         d_r_cylinder    db      0               ;cylinder we are on now
4061    1C98 00         d_r_head        db      0
4062    1C99 00         d_r_record      db      0
4063    1C9A 00         d_r_number      db      0               ;number of sectors read
4064
4065    1C9B 00         f_a_bytes       db      0
4066    1C9C 00         f_a_sectors     db      0
4067    1C9D 00         f_a_gap         db      0
4068    1C9E 00         f_a_filler      db      0
4069
4070    1C9F 46         dma_mode_storage        db      dma_mode_read_fdc       ;current DMA mode
4071    1CA0 66         flp_rw_command          db      fdc_read_cmd            ;reading or writing ?
4072
4073    1CA1 01         num_sec                 db      1               ;num sectors to read/write
4074                                                                    ;in one call to FDC
4075
```

```
4076
4077  1CA2                    track_mcnt              rb      1               ;multi sector count on
4078                                                                          ;current track
4079  1CA3                    dma_low16               rw      1               ;20 bit address storage
4080  1CA5                    dma_high4               rb      1
4081
4082  1CA6 00                 motor_flags             db      0               ;last state of the motor bits
4083  1CA7 00                 disk_busy               db      false
4084  1CA8 00                 motor_off_counter       db      0               ;time out counter to
4085                                                                          ;turn off motor
4086
4087                          ;       Sector buffer used by read/write routines when requested
4088                          ;       multi sector I/O operation crosses a 64K page boundary.
4089                          ;       This buffer cannot cross 64K page boundary on the IBM PC.
4090
4091  1CA9                    local_buf       rb      bytes_per_sector
4092
4093                          ;       Jump table for disk I/O read and write routines.
4094                          ;       Expand for hard disks and other types of floppies.
4095
4096  1EA9 C319               read_tbl        dw      offset read_sd_floppy
4097  1EAB C319                               dw      offset read_sd_floppy
4098  1EAD 000000000000                       dw      0,0,0,0         ;C,D,E,F
4099       0000
4100  1EB5 000000000000                       dw      0,0,0,0         ;G,H,I,J
4101       0000
4102  1EBD 00000000                           dw      0,0             ;K,L
4103  1EC1 8019                               dw      offset read_m_disk
4104  1EC3 000000000000                       dw      0,0,0           ;N,O,P
4105
4106  1EC9 131A               write_tbl       dw      offset write_sd_floppy
4107  1ECB 131A                               dw      offset write_sd_floppy
4108  1ECD 000000000000                       dw      0,0,0,0         ;C,D,E,F
4109       0000
4110  1ED5 000000000000                       dw      0,0,0,0         ;G,H,I,J
4111       0000
4112  1EDD 00000000                           dw      0,0             ;K,L
4113  1EE1 001A                               dw      offset write_m_disk
4114  1EE3 000000000000                       dw      0,0,0           ;N,O,P
4115
4116                          ;       Disk parameter headers
4117
4118                                                                  ;Floppy A:
4119  1EE9 00000000           dph_A   dw      xlt0,0000h              ;translate table
4120  1EED 00000000                   dw      0000h,0000h             ;scratch area
4121  1EF1 251F                       dw      dpb_flp                 ;dsk parm block
4122  1EF3 FFFF                       dw      0ffffh                  ;check
4123  1EF5 FFFF                       dw      0ffffh                  ;alloc vectors
4124  1EF7 FFFF                       dw      0ffffh                  ;dir buff cntrl blk
4125  1EF9 FFFF                       dw      0ffffh                  ;data buff cntrl blk
4126  1EFB FFFF                       dw      0ffffh                  ;hash table segment
4127
4128                                                                  ;Floppy d:
```

```
4129
4130  1EFD 00000000          dph_B   dw      xlt1,0000h      ;translate table
4131  1F01 00000000                  dw      0000h,0000h     ;scratch area
4132  1F05 251F                      dw      dpb_flp         ;dsk parm block
4133  1F07 FFFF                      dw      0ffffh          ;check
4134  1F09 FFFF                      dw      0ffffh          ;alloc vectors
4135  1F0B FFFF                      dw      0ffffh          ;dir buff cntrl blk
4136  1F0D FFFF                      dw      0ffffh          ;data buff cntrl blk
4137  1F0F FFFF                      dw      0ffffh          ;hash table segment
4138
4139                                                         ;Mdisk:
4140  1F11 00000000          dph_M   dw      0000h,0000h     ;translate table
4141  1F15 00000000                  dw      0000h,0000h     ;scratch area
4142  1F19 361F                      dw      dpb_m           ;dsk parm block
4143  1F1B 0000                      dw      0000h           ;check
4144  1F1D FFFF                      dw      0ffffh          ;alloc vectors
4145  1F1F FFFF                      dw      0ffffh          ;dir buff cntrl blk
4146  1F21 0000                      dw      0               ;data buff cntrl blk
4147  1F23 0000                      dw      0               ;hash table segment
4148
4149                        ;       Disk Parameter Blocks
4150
4151                                                         ;For both floppy drives
4152  1F25 0800          dpb_flp         dw      8           ;sectors per track
4153  1F27 03                        db      3               ;block shift
4154  1F28 07                        db      7               ;block mask
4155  1F29 00                        db      0               ;extnt mask
4156  1F2A 9800                      dw      155             ;disk size in 1k blocks
4157                                                         ;less offset track(s)
4158  1F2C 3F00                      dw      63              ;directory max
4159  1F2E C0                        db      11000000b       ;alloc0
4160  1F2F 00                        db      0               ;alloc1
4161  1F30 1000                      dw      16              ;check size
4162  1F32 0100                      dw      1               ;offset
4163  1F34 02                        db      2               ;phys sec shift
4164  1F35 03                        db      3               ;phys sec mask
4165
4166                                                         ;For Mdisk:
4167  1F36 0800          dpb_M           dw      8           ;sectors per track
4168  1F38 03                        db      3               ;block shift
4169  1F39 07                        db      7               ;block mask
4170  1F3A 00                        db      0               ;extnt mask
4171  1F3B BF00          dpb_m_dsm       dw      191         ;max disk size in 1k blocks
4172                                                         ;less offset track(s)
4173                                                         ;reset set by INIT: after memory test,
4174                                                         ;set max here so GENCCPM
4175                                                         ;can reserve the Allocation Vector
4176  1F3D 3F00                      dw      63              ;directory max
4177  1F3F C0                        db      11000000b       ;alloc0
4178  1F40 00                        db      0               ;alloc1
4179  1F41 0000                      dw      0               ;check size
4180  1F43 0000                      dw      0               ;offset
4181  1F45 00                        db      0               ;phys sec shift
```

```
4182
4183  1F46 00                          db      0       ;phys sec mask
4184
4185    0000                  xlt0    equ     0       ;no translate table
4186    0000                  xlt1    equ     xlt0    ;no translate table
4187
```

```
                        eject
                        ;                               Init
                        ;                               ----

                        ;***************************************************************************
                        ;*                                                                         *
                        ;*                      XIOS Initialization                                *
                        ;*                                                                         *
                        ;***************************************************************************

                        ;       The following routine is used to initialize any required
                        ;       data areas and alter any peripheral chip programming when
                        ;       starting up CCP/M-86.  This code is called once from the
                        ;       SUP(ERVISOR) after calling the SUP has called the RTM,
                        ;       RTM, CIO, MEM, BDOS initialization routines and before the
                        ;       SUP has created the RSP processes.
                        ;       This code can be placed in an XIOS data area if the XIOS is
                        ;       8080 model (mixed code and data).  Usually, overlaying the
                        ;       initialization code with a data area is done after the XIOS
                        ;       has been debugged.

1F47                    SL18    equ     offset $
                        CSEG
                        ORG     SL18

                        ;====
                        ;====
                        init:                                   ;arrive here from the JMP
                        ;====                                   ;at 0 in XIOS code segment
                        ;====

1F47 FAFC                       cli ! cld                       ;Supervisor restores DS,ES and int
                                                                ;224 after on INIT call
1F49 06                         push es                         ;save the UDA

                        ;       Memory initialization

1F4A CD12                       int mem_size_int                ;get memory size in K bytes
1F4C 50                         push ax                         ;save the memory size
1F4D 1E07                       push ds ! pop es                ;ES=SYSDAT
1F4F BFF721                     mov di,offset sign_mem          ;fill in sign on message
1F52 B364                       mov bl,100
1F54 F6F3                       div bl
1F56 0430                       add al,030h                     ;put in hundreds digit
1F58 AA                         stosb
1F59 8AC4                       mov al,ah                       ;get remainder
1F5B 32E4                       xor ah,ah                       ;make sure we don't overflow
1F5D B30A                       mov bl,10
1F5F F6F3                       div bl
1F61 0430                       add al,030h                     ;put in tens digit
1F63 AA                         stosb
1F64 8AC4                       mov al,ah                       ;get remainder
```

```
4241
4242  1F66 0430                     add al,030h
4243  1F68 AA                       stosb                           ;put in ones digit
4244  1F69 58                       pop ax                          ;restore memory size in K bytes
4245
4246                        ;       Trim the Memory Free List to the actual memory size.
4247                        ;       This code checks the MFL to be in the bounds of
4248                        ;       the address found by the INT MEMORY_SIZE call.
4249                        ;       The first Memory Descriptor and all those following
4250                        ;       MDs that represent partitions that extend past
4251                        ;       the end of memory, are placed on the Memory Descriptor
4252                        ;       Unused List.  The O.S. uses MDs from the
4253                        ;       MDUL when memory is allocated to processes.
4254
4255                        ;       This code can be expanded to adjust the MD_LENGTH
4256                        ;       field of the MD that overlaps the end of memory,
4257                        ;       instead of just placing it on the MDUL and potentially
4258                        ;       wasting some memory.  However, if the partitions
4259                        ;       are created by GENCCPM with starting addresses
4260                        ;       that correspond to the RAM card options there is no
4261                        ;       waste.  For the IBM PC, partitions starting
4262                        ;       at 64K and every 32K boundary thereafter will be
4263                        ;       trimmed by the following code with no wasted RAM.
4264
4265  1F6A B106                     mov cl,6                        ;make number of K into
4266  1F6C D3E0                     shl ax,cl                       ;number of paragraphs
4267  1F6E 8BC8                     mov cx,ax
4268  1F70 BB5A00                   mov bx,offset mfl - md_link
4269                          next_mfl:
4270  1F73 8BF3                     mov si,bx                       ;save previous link
4271  1F75 8B1F                     mov bx,md_link[bx]
4272  1F77 85DB                     test bx,bx                      ;GENCCPM sorts memory partitions
4273  1F79 7421             1F9C    jz mfl_done                     ;in accending order of starting
4274  1F7B 8B4702                     mov ax,md_start[bx]           ;paragraphs.  Assume memory is
4275  1F7E 034704                     add ax,md_length[bx]          ;contiguous from 0 on IBM PC.
4276  1F81 3BC1                       cmp ax,cx                     ;AX=end of partition, CX=
4277  1F83 76EE             1F73      jbe next_mfl                  ;end of memory
4278  1F85 C7040000                     mov md_link[si],0           ;terminate MFL
4279  1F89 8BF3                         mov si,bx                   ;save beginning of severed list
4280                          next_mdul:                            ;recycle MDs from MFL
4281  1F8B 8BF8                         mov di,bx                   ;save last link
4282  1F8D 8B1F                         mov bx,md_link[bx]          ;to Memory Descriptor
4283  1F8F 85DB                         test bx,bx                  ;Unused List
4284  1F91 75F8             1F8B        jnz next_mdul               ;look for end
4285  1F93 A15800                       mov ax,mdul                 ;save MDUL list
4286  1F96 89365800                     mov mdul,si                 ;attach trimmed MDs to MDUL
4287  1F9A 8905                         mov md_link[di],ax          ;re-attach original MDUL to
4288                          mfl_done:                             ;new end of MDUL
4289
4290                        ;       Initialize the Memory Disk
4291
4292                          test_for_mdisk:
4293  1F9C 8A00C0                   mov dx,m_disk_segment           ;DX is base of 64K Mdisk segments
```

```
4294
4295   1F9F 1E                          push ds                         ;save SYSDAT
4296   1FA0 B90300                      mov cx,3                        ;try three continuous 64K areas
4297
4298                            next64K:
4299   1FA3 8EC28FDA                    mov es,dx ! mov ds,dx           ;ES=DS=mdisk area
4300   1FA7 BEFEFF                      mov si,0fffeh                   ;last word of 64K segment
4301   1FAA 88FE                        mov di,si
4302   1FAC B855AA                      mov ax,0aa55h                   ;bit pattern to write
4303   1FAF 8BD8                        mov bx,ax                       ;keep for compare
4304   1FB1 AB                          stosw                           ;write it
4305   1FB2 AD                          lodsw                           ;read it
4306   1FB3 3BC3                        cmp ax,bx
4307   1FB5 7512              1FC9      jne m_test_done                 ;not equal-then no more memory
4308   1FB7 51                            push cx                       ;set Mdisk to E5's, blank disk
4309   1FB8 B8E5E5                        mov ax,0E5E5h                 ;parity of memory at C0000H is not
4310   1FBB 33FF                          xor di,di                     ;set by IBM PC ROM
4311   1FBD B90080                        mov cx,8000H                  ;32K words = 64K bytes
4312   1FC0 F3AB                          rep stosw
4313   1FC2 59                            pop cx                        ;restore loop count
4314   1FC3 81C20010                      add dx,1000H                  ;try next 64K
4315   1FC7 E2DA              1FA3        loop next64k
4316
4317                            m_test_done:
4318   1FC9 1F                          pop ds                          ;restore DS
4319   1FCA 83F903                      cmp cx,3                        ;if equal no mdisk
4320   1FCD 7508              1FD7      jne m_exists
4321   1FCF C706300C0000                  mov dpn_m_adr,0               ;IO_SELDSK will return
4322   1FD5 EB41              2018        jmps m_disk_done              ;an error on Mdisk select
4323
4324                            m_exists:
4325   1FD7 C606FC210D                    mov sign_m_disk,cr            ;print mdisk message
4326   1FDC 83F902                        cmp cx,2                      ;select to work
4327   1FDF 750E              1FEF        jne more_than_64K
4328   1FE1 C7061F223634                    mov sign_md_size1,'46'      ;64 - low byte high byte storage
4329   1FE7 C7063B1F3F00                    mov dpb_m_dsm,63            ;number of 1K blocks - 1
4330   1FED EB29              2018          jmps mdisk_done
4331
4332                            more_than_64K:
4333   1FEF 83F901                          cmp cx,1
4334   1FF2 7513              2007          jne more_than_128K
4335   1FF4 C6061E2231                        mov sign_md_size0,'1'
4336   1FF9 C7061F223238                      mov sign_md_size1,'82'
4337   1FFF C7063B1F7F00                      mov dpb_m_dsm,127           ;number of 1K blocks - 1
4338   2005 EB11              2018            jmps mdisk_done
4339
4340                            more_than_128K:                          ;it is 192K
4341   2007 C6061E2231                          mov sign_md_size0,'1'
4342   200C C7061F223932                        mov sign_md_size1,'29'  ;low byte high byte
4343   2012 C7063B1FBF00                        mov dpb_m_dsm,191       ;number of 1K blocks - 1
4344                                            ;jmps m_disk_done
4345
4346                            m_disk_done:
```

```
4347
4348
4349                            ;       Disk I/O Initialization
4350
4351                            ;       Initialize DPH Hash Table Segment Entry.
4352                            ;       Commented out since we let GENCCPM perform this
4353                            ;       initialization.  Use this code if you build your own
4354                            ;       Buffer Control Blocks, Hash Table Segment, Allocation
4355                            ;       and Check Sum Vectors.
4356
4357                            ;       mov cx,16
4358                            ;inith0:        push cx
4359                            ;       dec cl
4360                            ;       call io_seldsk                  ;get disk param header offset
4361                            ;       or bx,bx ! jz initn1            ;if not 0, BX = DPH
4362                            ;           mov ax,18[bx]               ;AX = Hash Tbl  Offset
4363                            ;           or ax,ax ! jz inith1        ;if 0, No Hash Tbl
4364                            ;               mov cl,4
4365                            ;               shr ax,cl
4366                            ;               mov dx,ds
4367                            ;               add ax,dx               ;AX = Hash Tbl Seg
4368                            ;               mov 18[bx],ax
4369                            ;inith1:
4370                            ;       pop cx
4371                            ;       loop inith0
4372                            ;
4373                            ;       Initialize data BCB segment addresses
4374                            ;       all drives share the same set of data buffers
4375                            ;
4376                            ;       xor cl,cl                       ;get disk param header address
4377                            ;       call io_seldsk                  ;BX=DPH address for drive A:
4378                            ;       mov si,16[bx]                   ;SI=data BCB root address
4379                            ;       mov si,[si]
4380                            ;initd1:
4381                            ;       mov ax,10[si]                   ;AX=BCB buffer offset
4382                            ;       mov cl,4
4383                            ;       shr ax,cl
4384                            ;       mov dx,ds
4385                            ;       add ax,dx
4386                            ;       mov 10[si],ax                   ;AX=BCB buffer segment
4387                            ;       mov si,12[si]                   ;SI=next BCB
4388                            ;       or si,si ! jnz initd1           ;0 if end of linked list
4389                            ;
4390                            ;end of disk init
4391
4392                            set_up_interrupts:
4393  2018 B08C                         mov al,10111100b                ;enable diskette, keyboard,
4394  201A E621                         out pic_odd_port,al             ;timer interrrupt
4395                                                                    ;and mask off the rest
4396
4397  201C C606891C03                   mov disk_arguments,3            ;send specify command to the
4398  2021 C6068A1C03                   mov d_a_command,fdc_spec_cmd    ;FDC (Intel 8272 or NEC 765)
4399  2026 C6068B1CDF                   mov d_a_drive,0dfh              ;step rate=3ms, head unload=240ms
```

```
4400
4401  202B C6068C1C02                mov d_a_cylinder,02h          ;head load=2ms, DMA mode true
4402  2030 E81AFC       1C4D         call fdc_command_put          ;head unload and head load times
4403                                                               ;are meaningless on mini floppies
4404                                                               ;where the head is loaded
4405                                                               ;when the motor is turned on
4406  2033 B8AE4D                    mov ax,timer_60_hz
4407  2036 E640                      out timer_0_reg,al
4408  2038 86E0                      xchg ah,al
4409  203A E640                      out timer_0_reg,al
4410
4411  203C B036                      mov al,bell_cmd               ;set up the bell frequency
4412  203E E643                      out timer_cmd_reg,al          ;send the command
4413  2040 B83305                    mov ax,timer_1000_hz          ;get the constant
4414  2043 E642                      out timer_2_reg,al
4415  2045 86E0                      xchg ah,al
4416  2047 E642                      out timer_2_reg,al
4417
4418                          ;       Paint the software interrupt vectors.
4419
4420                          ;       When debugging leave the single step, one byte, debugger and
4421                          ;       serial interrupt vectors pointing at DDT86 and CP/M-86.
4422
4423                          ;       To debug CCP/M-86 under CP/M-86, DDT86 or SID86 must be
4424                          ;       able to perform console input, console input status and
4425                          ;       console output through the CP/M-86 BIOS.
4426                          ;       This is usually accomplished by BIOS through a serial port.
4427                          ;       Often the BIOS routines AUXIN and AUXOUT, are used for
4428                          ;       serial device support.  If the BIOS supports the IOBYTE,
4429                          ;       it may be possible to redirect the CP/M console to a
4430                          ;       serial port using the STAT transient.
4431
4432                          ;       When debugging is finished, all interrupts unused
4433                          ;       by the XIOS should be initialized in XIOS INIT to
4434                          ;       point at the XIOS I_UNEXPECTED interrupt handler routine.
4435
4436  2049 33D2                      xor dx,dx
4437  204B 803E3C0CFF                cmp debug,true                ;debug flag: see HEADER
4438  2050 7526         2078         jne no_save
4439  2052 1E07                        push ds ! pop es            ;ES=SYSDAT
4440  2054 8EDA                        mov ds,dx                   ;DS=0
4441  2056 BB0200                      mov bx,2                    ;count of 2 words
4442  2059 8BCB                        mov cx,bx
4443  205B BF3D21                      mov di,offset iv_save       ;local save area
4444  205E BE0400                      mov si,iv_sstep             ;single step interrupt
4445  2061 F3A5                        rep movsw
4446  2063 8BCB                        mov cx,bx                   ;CX=2
4447  2065 BE0C00                      mov si,iv_one_byte          ;one byte INT instruction
4448  2068 F3A5                        rep movsw
4449  206A 8BCB                        mov cx,bx
4450  206C BE5000                      mov si,iv_rs232             ;serial port
4451  206F F3A5                        rep movsw
4452  2071 8BCB                        mov cx,bx                   ;CX=2
```

```
4453
4454   2073 8E8403                      mov si,iv_debug              ;debugger interrupt
4455   2076 F3A5                        rep movsw
4456
4457                            no_save:                             ;paint all the interrupt vectors
4458   2078 8ED4                        mov ss,dx                    ;to point at T_UNEXPECTED:
4459   207A 8EC2                        mov es,dx                    ;ES=0, SS=0
4460   207C C706000006A0F               mov word ptr .0,offset i_unexpected
4461   2082 8C0E0200                    mov word ptr .2,cs
4462   2086 B9FE01                      mov cx,255*2
4463   2089 BE00003F0400                mov si,0 ! mov di,4
4464   208F F3A5                        rep movsw
4465
4466                            int_offsets:                         ;interrupt vectors used by XIOS
4467   2091 B89F0C                      mov ax,offset i_tick         ;DS=0,ES=0
4468   2094 A32000                      mov .iv_tick,ax              ;tick interrupt
4469   2097 B8B50D                      mov ax,offset i_keyboard
4470   209A A32400                      mov .iv_key,ax
4471   209D B8C00E                      mov ax,offset i_disk
4472   20A0 A33800                      mov .iv_disk,ax
4473
4474   20A3 2E8E1E060C                  mov ds,cs:sysdat             ;in debug environment ?
4475   20A8 803E3C0CFF                  cmp debug,true
4476   20AD 751F          20CE          jne no_restore
4477                                                                 ;DS=SYSDAT,ES=0
4478   20AF 8BCB                          mov cx,bx                  ;BX=2
4479   20B1 BE3D21                        mov si,offset iv_save      ;local save area
4480   20B4 BF0400                        mov di,offset iv_s_step    ;single step interrupt
4481   20B7 F3A5                          rep movsw
4482   20B9 8BCB                          mov cx,bx                  ;CX=2
4483   20BB BF0C00                        mov di,offset iv_one_byte  ;one byte INT instruction
4484   20BE F3A5                          rep movsw
4485   20C0 8BCB                          mov cx,bx                  ;CX=2
4486   20C2 BF5000                        mov di,offset iv_rs232     ;serial port
4487   20C5 F3A5                          rep movsw
4488   20C7 8BCB                          mov cx,bx
4489   20C9 BF8403                        mov di,offset iv_debug     ;debugger interrupt
4490   20CC F3A5                          rep movsw
4491
4492                            no_restore:
4493   20CE 1E                          push ds
4494   20CF 07                          pop es                       ;ES=SYSDAT
4495
4496                            init_parallel:
4497   20D0 BA8C03                      mov dx,3bch                  ;DW parallel port
4498   20D3 83C202                      add dx,2
4499   20D6 B008                        mov al,08h                   ;printer init byte
4500   20D8 EE                          out dx,al
4501   20D9 B80010                      mov ax,1000h                 ;delay count
4502
4503                            parallel_wait:
4504   20DC 48                          dec ax
4505   20DD 75FD          20DC          jnz parallel_wait
```

```
4506
4507  20DF B00C                         mov al,0Ch                      ;printer strobe off
4508  20E1 EE                           out dx,al
4509
4510                              set_up_video:
4511  20E2 BA8403                       mov dx,bw_card                  ;get the video chip port
4512  20E5 BE2D21                       mov si,offset bw_table          ;initialization commands
4513  20E8 B91000                       mov cx,length bw_table          ;how many commands
4514  20EB E82B00       2119            call video_init                 ;send commands to port
4515                                                                      ;color board is similar ...
4516
4517                              ;       Set up the virtual screen structures (one per virtual console)
4518                              ;       and blank their screen save areas.
4519
4520  20EE 8B16380C                     mov dx,genccpm_buf              ;paragraph address of buffer
4521                                                                      ;space allocated by GENCCPM.
4522                                                                      ;this area is not part of the
4523                                                                      ;CCPM.SYS file, and is located
4524                                                                      ;with GENCCPM allocated disk
4525                                                                      ;data buffers at the end of
4526                                                                      ;the system image.
4527  20F2 32E4                         xor ah,ah                       ;CX = screen we are initializing
4528                                                                      ;and param to erase_screen
4529  20F4 BBF317                       mov bx,offset screen_structures
4530
4531                              init_screen_structures:
4532  20F7 895706                       mov ss_screen_seg[bx],dx        ;RAM storage segment for this virtual
4533                                                                      ;console
4534  20FA 5052                         push ax ! push dx               ;save screen #
4535
4536                              ;       test ah,ah                      ;to keep the LOADER messages,
4537                              ;       jz no_erase                     ;don't erase screen 0
4538  20FC E8E2F0       11E1            call erase_screen
4539                              ;no_erase:
4540  20FF 5A58                         pop dx ! pop ax                 ;screen #
4541  2101 81C2F000                     add dx,(screen_siz*2 + 15)/16   ;next virtual screen storage area
4542  2105 83C30E                       add bx,ss_len                   ;next screen_structure
4543  2108 FEC43A26110C                 inc ah ! cmp ah,nvcns           ;do for each virtual console
4544  210E 72E7         20F7            jb init_screen_structures
4545
4546  2110 BE5121                       mov si,offset sign_on
4547  2113 E80E01       2224            call print_msg
4548  2116 07                           pop es                          ;restore the UDA
4549  2117 FB                           sti
4550  2118 CB                           retf                            ;initializaiton done
4551
4552                              video_init:
4553                              ;----------
4554                              ;       entry:  DX = video chip port
4555                              ;               SI = table of commands to send to the port
4556                              ;               CX = number of commands to send
4557                              ;       exit:   none
4558
```

```
4559
4560   2119 32DB                        xor bl,bl
4561
4562                            video_init_1:
4563   211B 8AC3FEC3                    mov al,bl ! inc bl
4564   211F EE42                        out dx,al ! inc dx
4565   2121 ACEE                        lodsb ! out dx,al
4566   2123 4A                          dec dx !
4567   2124 E2F5           2116         loop video_init_1
4568
4569   2126 83C204                      add dx,4
4570   2129 B029                        mov al,video_on
4571   212B EE                          out dx,al
4572   212C C3                          ret
4573
4574                            ;*************************************************************************
4575                            ;*                                                                       *
4576                            ;*                          INIT Data                                    *
4577                            ;*                                                                       *
4578                            ;*************************************************************************
4579
4580     212D                   SL19     equ      offset $
4581                            DSEG
4582                            ORG      SL19
4583
4584   212D 6150520F1906        bw_table          db       61h,50h,52h,0fh,19h,06h,19h,19h,02h,0dh,05h,0ch,0,0,0,0
4585        19190200080C
4586        00000000
4587
4588   213D                     iv_save           rw       10              ;space for 5 interrupt vectors
4589
4590   2151                     sign_on rb        0
4591   2151 45786160706C                          db 'Example XIOS Version of '
4592        652058494F53
4593        205665727369
4594        6F6E206F6620
4595   2169 342F31352F38                          db '4/15/83'
4596        33
4597   2170 0D0A0D0A                              db cr,lf,cr,lf
4598   2174 486172647761                          db 'Hardware Supported :',cr,lf
4599        726520537570
4600        706F72746564
4601        203A0D0A
4602   218A 202020202020                          db '                    Diskette(s) :   '
4603        202020202020
4604        202020202020
4605        204469736B65
4606        747465287329
4607        203A202020
4608   21AD 320D0A              sign_d            db '2',cr,lf
4609   21B0 202020202020                          db '      Parallel Printer Port(s) :   '
4610        506172616C6C
4611        656C20507269
```

```
4612
4613          6E7465722050
4614          6F7274287329
4615          203A202020
4616   21D3   31000A              sign_sp         db '1',cr,lf
4617   21D6   202020202020                        db '                Main Memory (Kb) : '
4618          202020202020
4619          20204061696E
4620          2040656D6F72
4621          792028486229
4622          203A20
4623   21F7   2020200D0A          sign_mem        db '    ',cr,lf
4624   21FC   00                  sign_mdisk      db 0                                    ;end msg here
4625   21FD   202020202020                        db '                    M:Disk (Kb) : '  ;unless mdisk
4626          202020202020
4627          202020202020
4628          20403A446973
4629          682028486229
4630          203A20
4631   221E                       sign_md_size0   rb 1                                     ;set to 64,128,192
4632   221F                       sign_md_size1   rw 1
4633   2221   0D0A00                              db cr,lf,0
4634
```

```
4635
4636                            eject
4637                            ;                       Miscellaneous Routines
4638                            ;                       ----------------------
4639
4640                            ;************************************************************************
4641                            ;*                                                                      *
4642                            ;*                              RESET                                   *
4643                            ;*                                                                      *
4644                            ;************************************************************************
4645
4646     2224                   SL20    equ     offset $
4647                            CSEG
4648                            ORG     SL20
4649
4650                            ;---------
4651                            print_msg:
4652                            ;---------
4653                            ;       entry:  SI = address of string to print until 0 byte
4654                            ;                       on foreground console
4655                            ;       exit:   none
4656
4657     2224 8A163318                  mov dl,foreground_screen
4658                            p_msg_l:
4659     2228 8A0C                      mov cl,[si]
4660     222A 84C9                      test cl,cl
4661     222C 740A          2238        jz p_done
4662     222E 5256                        push dx ! push si
4663     2230 E8C5EE        10F8          call io_conout
4664     2233 5E5A                        pop si ! pop dx
4665     2235 46                          inc si
4666     2236 EBF0          2228          jmps p_msg_l
4667                            p_done:
4668     2238 C3                        ret
4669
4670                            ;-----
4671                            reset:
4672                            ;-----
4673                            ;       Reset function - reboot from floppy
4674
4675     2239 B84000                      mov ax,40H
4676     223C 8ED8                        mov ds,ax                      ;ROM data segment
4677     223E C70672003412                mov reset_flag,1234H           ;parameter to reset routine
4678     2244 EA0000FFFF                  jmpf rom_reset                 ;don't perform diagnostics
4679
4680       FFFF                 CSEG    0ffffh                         ;address in ROM monitor
4681                            ORG     0                              ;of reset routine
4682
4683                            rom_reset:
4684
4685       0000                 DSEG    0
4686                            ORG     72h
4687
```

```
4688
4689   0072                    reset_flag      rw      1
4690
4691
4692  END OF ASSEMBLY.  NUMBER OF ERRORS:  0.  USE FACTOR: 45%
```

```
ACK                0006 N   485# 2941
ACTIONKEY          10D3 L  1470  1533#
ACTIONKEYDONE      10F6 L  1561  1564  1568  1573#
ACTIONKEYMASKS     1673 L  1468  2801#
ACTIONKEYTABLE     1675 L  1460  1466  2793#
ACTIONMAKE         10E6 L  1558  1562#
ALT                0038 N  1554  2780# 2797
ALTBIT             0008 N  1436  1517  2789# 2805
BACKSPACE          11A6 L  1752# 2977
BEL                0007 N   486# 1638  2942
BELLCMD            00B6 N   801# 4411
BELLFLAG           0006 N   470#  471   870  1648
BELLOFF            00FC N   803#  877
BELLON             0003 N   802# 1645
BELLTICKS          0D52 V   866   868   874   944# 1642  1647
BS                 0008 N   487# 2842  2891  2942  2973
BSRET              11CC L  1759  1767  1774#
BSROW              11B7 L  1762  1765#
BSSETCURSOR        11C4 L  1764  1770#
BUFSTR             1942 V  2585  3247#
BVERNUM            007A V   263#
BWCARD             03B4 N  1330# 2231  2372  4511
BWTABLE            212D V  4512  4513  4584#
BWVIDEOSEG         3000 N  1303# 2393  2434  2447  2641
BWVIDEOSTATUSLIN   0F00 N  1305# 2640
BYTESPERDISK       8000 N  3299#
BYTESPERSECTOR     0200 N  3295# 3297  3684  3701  4091
BYTESPERTRACK      1000 N  3297# 3299
CAN                0018 N   504# 2945
CAPSLOCK           003A N  2781# 2799
CAPSTR             196A V  2616  2618  3258#
CARRIAGERETURN     1187 L  1675  1724# 2104  2975
CCB                0C14 V   556# 2423
CCB0               1838 V  3127  3131#
CCB1               1864 V  3146#
CCB2               1890 V  3160#
CCB3               18BC V  3174#
CCBLEN             002C N   403# 2422
CCBTAB             1838 V   556  3130#
CHECKBELLOFF       0CE2 L   869   873#
CHECKMOTOROFF      0CEF L   867   872   875   880#
CHECKSECONDS       0D0C L   882   885   888   894#
CIHAVEONE          0FFD L  1382  1389#
CIHAVEONE2         0FFE L  1388  1392#
CIPTROK            1013 L  1399  1401#
CIWAIT             0FEF L  1383#
CMAXBUFSIZ         0010 N   402#
CMIMIC             0008 N   400# 2574
COBACK             1149 L  1663#
COLOOKUP           1166 L  1626  1684# 1798
COLUMNSPERSCREEN   0050 N  1296# 1297  1671  1769  1836  1869  1883  1951  1953  2018
                           2040  2062  2119  2280  2283  2292  2315  2319  2327  2340
                           2535  2644
```

```
COMPDMA          180D L  3640  3676  3710  3735#
COMPUTECURSOR    142C L  1956  2336#
CONINDONE        102C L  1410  1420#
CONOESCAPE       1114 L  1623# 1794  1959  3075  3085  3095  3105
CONOWRAP         1162 L  1674  1677#
CONSTR           195A V  2538  2540  3254#
CONTROLTABLE     1730 L  1481  2921#
COOK             1103 L  1603  1607#
COSIMPLECHAR     1120 L  1627  1632#
COWNER           0000 N   399# 2553
CR               000D N   492# 1273  1278  2845  2894  2939  2946  2973  4325  4597
                        4597  4598  4608  4616  4623  4633
CRDONE           1190 L  1733#
CS               SREG V   931  1062  1170  4461  4474
CSMBACKGROUND    0002 N   408# 3152  3166  3181
CSMBUFFERED      0001 N   407# 2584
CSMCTRLD         0100 N   412# 2569
CSMCTRLP         0200 N   413# 2572
CSMCTRLS         0080 N   411# 2565
CSMNOSWITCH      0008 N   410# 2582
CSMPURGING       0004 N   409# 2586
CSTATE           000E N   401# 2564
CTRL             001D N  2777# 2794
CTRLBIT          0001 N  1434  1479  2787# 2802
CURSUREND        000B N  1336#
CURSORHI         000E N  1340# 2377
CURSORLOW        000F N  1341# 2373
CURSORONOFF      13AE L  2219  2230#
CURSORSTART      000A N  1335# 2232
DACOMMAND        1C8A V  3830  3846  3923  3938  4046# 4396
DACYLINDER       1C8C V  3863  4048# 4401
DADRIVE          1C8B V  3832  3866  4047# 4399
DADTL            1C92 V  3769  4054#
DAEOT            1C90 V  3767  3872  4052#
DAGPL            1C91 V  3768  4053#
DAHEAD           1C8D V  3860  4049#
DANUMBER         1C8F V  3766  4051#
DARECORD         1C8E V  3869  4050#
DC1              0011 N   496#  514  2914  2936
DC2              0012 N   497# 2936
DC3              0013 N   499#  515  2865  2941
DC4              0014 N   500# 2937
DDOWN            1226 L  1744  1865  1868#
DEBUG            0C3C V   582# 1227  4437  4475
DEBUGINT         00E1 N   750#  759
DEL              007F N   512# 2934
DELETECHAR       1344 L  2107# 3016
DELETELINE       132A L  2090# 3015
DEVPOLL          0083 N   289#  714
DEVSETFLAG       0085 N   291#  656
DEVWAITFLAG      0084 N   290#  665
DISABLECURSOR    13A8 L  2222# 2462  3027
DISABLEWRAP      13C0 L  2251# 3029
```

```
DISKARGUMENTS    1C89 V  3829  3845  3921  3937  3955  4045# 4397
DISKBUSY         1CA7 V   881  3616  3732  4083#
DISKCHANNEL      0006 N   785#
DISKINT          000E N   744#  757
DISKRESULTS      1C93 V  3927  3940  3987  3989  4056#
DISKSP           0F68 V  1135  1164  1207#
DISKSS           0F66 V  1134  1163  1206#
DISPATCHER       003A V   243#  931  1062  1170
DISPLAYSTARTHI   000C N  1338#
DISPLAYSTARTLOW  000D N  1339#
DLE              0010 N   495# 2902  2938  2948
DMABMSKFDC       0002 N  3384# 3908
DMABMSKREG       000A N  3367# 3909
DMAC0ADDRESS     0000 N  3356#
DMAC0COUNT       0001 N  3357#
DMAC1ADDRESS     0002 N  3358#
DMAC1COUNT       0003 N  3359#
DMAC2ADDRESS     0004 N  3360# 3896  3898
DMAC2COUNT       0005 N  3361# 3905  3907
DMAC3ADDRESS     0006 N  3362#
DMAC3COUNT       0007 N  3363#
DMACBPF          000C N  3369# 3891
DMACLEAR         000D N  3371#
DMACMDREG        0008 N  3365#
DMAHIGH4         1CA5 V  3753  3899  4080#
DMALOW16         1CA3 V  3752  3895  4079#
DMAMASKREG       000F N  3372#
DMAMODEREADFDC   0046 N  3382# 3522  4070
DMAMODEREG       0008 N  3368# 3894
DMAMODESTORAGE   1C9F V  3522  3602  3893  4070#
DMAMODEWRITEFDC  004A N  3381# 3602
DMAOFF           0006 V  3436# 3509  3591  3639  3670  3680  3687  3698  3705  3709
                         3721
DMAPAGEC1        0080 N  3374#
DMAPAGEC3        0082 N  3376#
DMAPAGEFDC       0081 N  3375# 3900
DMAREQUREG       0009 N  3366#
DMASEG           0008 V  3435# 3638  3683  3700  3708
DMASETUP         1BEC L  3776  3881#
DMASTATREG       0008 N  3364# 3365  3366  3367  3368  3369  3370  3371  3372
DMATEMPREG       000D N  3370#
DOCASECHANGE     10BD L  1508  1513#
DOOR             0C0E V   549#
DOWN             121F L  1859# 2999
DOWNBITS         1674 V  1434  1436  1479  1488  1494  1517  1560  1563  2770#
DPB              0008 N  3443#
DPBFLP           1F25 V  4121  4132  4152#
DPBM             1F36 V  4142  4167#

DPHA             1EE9 V   560  4119#
DPHB             1EFD V   561  4130#
DPHM             1F11 V   566  4140#
DPHMADR          0C30 V   566# 4321
```

```
DPHTBL          0C18 V   560# 3474
DRCYLINDER      1C97 V  4060#
DRHEAD          1C98 V  4061#
DRIVE           000E V  3431# 3499  3578  3797  3803  3831  3865
DRNUMBER        1C96 V  4063#
DRRECORD        1C99 V  4062#
DRST0           1C94 V  3929  4057#
DRST1           1C95 V  4058#
DRST2           1C96 V  4059#
DS              SREG V   829   830   927   995   995  1058  1133  1133  1166  1233
                       1404  1697  2122  2122  2125  2295  2295  2297  2330  2330
                       2332  2435  2436  2438  2451  2452  2454  2522  2523  2532
                       2638  2650  3512  3513  3515  3590  3594  3675  3682  3683
                       3686  3985  4228  4295  4299  4318  4439  4440  4458  4474
                       4493  4676
DUP             1237 L  1880  1882#
DYNSTR          193A V  2579  2583  3245#
EM              0019 N   505# 2937
ENABLECURSOR    13A0 L  2212# 2459  3025
ENABLEWRAP      1389 L  2241# 3028
ENDSEG          0044 V   247#
ENQ             0005 N   484# 2936
ENTERBLINK      138C L  2175# 3023
ENTERBRIGHT     1396 L  2194# 3021
ENTERREVERSE    1362 L  2129# 3017
ENTRY           0C3D L   532   597#
EOT             0004 N   483# 2941
EQUIPINT        0011 N   745#
ERASEBEGIN      129E L  1963# 3008
ERASEBOL        12E1 L  2030# 3011
ERASEEND        1282 L  1986# 3009
ERASEEOL        12FE L  2053# 3012
ERASELINE       12C8 L  1749  1898  2008# 2081  2101  3010
ERASESCREEN     11E1 L  1801# 3006  4538
ES              SREG V   846   909  1006  1038  1144  1150  1234  1255  1403  1404
                       1406  1660  1668  1697  1698  1703  1808  1815  1970  1982
                       1993  2004  2013  2025  2035  2049  2058  2067  2112  2122
                       2125  2274  2295  2297  2308  2330  2332  2394  2399  2419
                       2432  2448  2454  2507  2532  2642  2653  3508  3516  3587
                       3588  3595  3682  3682  3686  3700  3700  3704  3983  3986
                       4001  4222  4228  4299  4439  4459  4494  4548
ESC             001B N   507# 2839  2888  2926  2938  2973
ESCAPE          11CD L  1783# 2978
ESCAPE1         11D3 L  1790  1793#
ESCFUNCTAB1     17BB V  1797  2996#
ESCTAB1         179E V  1796  2986#
ETB             0017 N   503# 2936
ETX             0003 N   482# 2945  2956
EXITBLINK       1391 L  2184# 3024
EXITBRIGHT      1398 L  2203# 3022
EXITREVERSE     1380 L  2164# 3019
FABYTES         1C9B V  4065#
FAFILLER        1C9E V  4068#
```

```
FAGAP              1C9D V  4067#
FALSE              0000 N   451#  547   549   582   902  1027  1031  1111  2512  2654
                            3208  3209  3732  4083
FASECTORS          1C9C V  4066#
FBYTES             0002 N  3340#
FDCCOMMANDLOOP     1C59 L  3961# 3964  3969
FDCCOMMANDPUT      1C4D L  3833  3873  3924  3939  3944# 4402
FDCDATA            03F5 N  3312#
FDCFLAG            0004 N   468# 1146  3834  3874  3925
FDCFORMATCMD       004D N  3328#
FDCNOMOTOR         00FC N  3324#
FDCON              000C N   890  3323# 3802
FDCPORT            03F2 N   889  3313# 3805
FDCREADCMD         0066 N  3326# 3523  3677  3696  3807  4071
FDCREADWRITE       1C15 L  3777  3912#
FDCREADY           0080 N  3335# 3963  3994
FDCRECALCMD        0007 N  3330# 3830
FDCSEEKCMD         000F N  3329# 3846
FDCSIGMD           0008 N  3331# 3938
FDCSPEC1           00CF N  3337#
FDCSPEC2           0003 N  3338#
FDCSPECCMD         0003 N  3332# 4398
FDCSTAT            03F4 N  3311# 3312  3954  3984
FDCSTATUSGET       1C65 L  3928  3941  3973#
FDCSTATUSLOOP      1C77 L  3992# 3995  4000
FDCWRITECMD        0045 N  3327# 3603
FF                 000C N   491# 2891  2943
FFILLER            00E5 N  3343#
FGAP               003A N  3342#
FLPEND64K          1A54 L  3645  3648#
FLPEND64KOK        1A6D L  3661  3664#
FLPIO              1A1D L  3525  3607# 3729
FLPLOCAL           1AA7 L  3679  3689#
FLPLOCALDONE       1AD2 L  3697  3707#
FLPPGOK            1ADB L  3646  3712#
FLPPHYSIO          1B25 L  3660  3691  3715  3756#
FLPRET             1B07 L  3662  3692  3717  3726  3731#
FLPRWCOMMAND       1CA0 V  3523  3603  3677  3696  3807  3922  4071#
FLPRWLOCAL         1A7C L  3656  3672#
FLPTRACK           1A34 L  3623  3626#
FOREGROUNDCCB      1834 V  2424  2552  3127#
FOREGROUNDSCREEN   1833 V  2368  2395  2426  2440  2455  2548  3126# 4657
FOREGROUNDSS       1836 V  1459  1640  2445  2613  3128#
FS                 001C N   508# 2945
FSECTORS           0008 N  3341#
FUNCTIONTABLE      0C4F L   617   630#
GENCCPMBUF         0C38 V   575# 4520
GS                 001D N   509# 2938
HOME               11F2 L  1819# 2996
HT                 0009 N   488# 2842  2937
IDDISP             0F06 L  1167  1169#
IDISK              0EC0 L  1128# 4471
IDISKSTACK         0F66 V  1137  1204#
```

```
IGNOREBELL          1142 L   1641  1643  1652#
IKDISP              0E37 L   1059  1061#
IKEYBOARD           0D35 L    989# 4469
IKEYBOARDSP         0E98 V    997  1055  1100#
IKEYBOARDSS         0E96 V    996  1054  1098#
IKEYBOARDSTACK      0E96 V    999  1096#
IKFULL              0DFC L   1013  1026#
IKNOFLAG            0E0E L   1028  1033#
IKPTROK             0DF8 L   1021  1023#
ILLFUNC             0C4E L    614   620#
INCCOL              1163 L   1672  1680#
INIT                1F47 L    531  4216#
INITPARALLEL        20D0 L   4496#
INITSCREENSTRUCT    20F7 L   4531# 4544
INSERTLINE          1311 L   2071# 3014
INICNT              0D54 V    841   926   946# 1001  1057  1139  1165
INIOFFSETS          2091 L   4466#
IOAUXIN             1671 L    636  2734#
IOAUXOUT            1672 L    637  2744#
IOCONIN             0FE6 L    632  1372# 1414  1416
IOCONOUT            10F8 L    633  1584# 4663
IOCONST             0FE3 L    631  1354#
IOFLUSHBUF          1C84 L    643  4011#
IOLIST              1656 L    635  2701#
IOLISTST            163E L    634  2665#
IOUCONOUT           10FC L   1597  1600#
IOPOLL              0C79 L    644   685#  709
IOREAD              19A1 L    641  3485#
IOSELDSK            1991 L    640  3458#
IOSTATLINE          14E2 L    639  2480#
IOSWITCH            1472 L    638  2410#
IOWRITE             19F1 L    642  3562#
ITDISP              0D4D L    928   930#
ITICK               0C9F L    824# 4467
ITICKEXIT           0D2A L    903   908#
ITICKSP             0DB2 V    835   923   971#
ITICKSS             0DB0 V    833   922   970#
ITICKSTACK          0DB0 V    837   968#
IUABORT             0F9F L   1250  1253#
IUNEXPECTED         0F6A L   1220# 4460
IUNOPD              0F90 L   1238  1244#
IVDEBUG             0384 N    759# 4454  4489
IVDISK              0038 N    757# 4472
IVKEY               0024 N    756# 4470
IVNMI               0008 N    753#
IVONEBYTE           000C N    754# 4447  4483
IVRS232             0050 N    758# 4450  4486
IVSAVE              213D V   4443  4479  4588#
IVSSTEP             0004 N    752# 4444  4480
IVTICK              0020 N    755# 4468
KBBUF               0E9A V   1020  1022  1107# 1108  1109  1398  1400
KBCNT               0EBE V   1011  1015  1110# 1381  1393
KBDCONTROL          0061 N    768# 1034  1042  1044
```

```
KBDDATA          0060 N   767# 1008
KBIN             0EBC V  1016  1024  1109#
KBLEN            0020 N  1012  1020  1105# 1107  1398
KBOUT            0EBA V  1108# 1394  1402
KBWAIT           0EBF V  1027  1031  1111# 1384
KEYBOARDCHANNEL  0001 N   784#
KEYFLAG          0005 N   469# 1029  1386
KEYINT           0009 N   743#  756
KEYMAKE          1073 L  1474  1477#
KEYTABLE         1695 L   580  1478  2834#
LASTFLAG         0006 N   471#
LCB              0C16 V   557#
LCB0             197D V  3272#
LCB1             1987 V  3276#
LCBTAB           197D V   557  3271#
LEFT             120F L  1845# 2998
LF               000A N   489# 1273  1278  2942  2973  4597  4597  4598  4608  4616
                         4623  4633
LFSCROLL         119C L  1743  1745#
LINEFEED         1193 L  1676  1737# 2976
LISTPORTS        0FDF V  1314# 2689  2714
LISTST           1644 L   723   724  2678#
LLEFT            1216 L  1851  1853#
LOCALBUF         1CA9 V  3674  3681  3699  4091#
LSIDONE          1670 L  2717  2724#
LUFUNC           117A L  1704  1707#
MASKRELEASE      1045 L  1433  1435  1437  1440#
MCNT             000F V  3432# 3547  3620  3624  3627  3725
MDISKCALC        1900 L  3507  3527# 3586
MDISKDONE        2018 L  4322  4330  4338  4346#
MDISKSEGMENT     C000 N  3303# 3545  4293
MDLEN            0008 N   443#
MDLENGTH         0004 N   442#  443  4275
MDLINK           0000 N   439#  441  4268  4271  4278  4282  4287
MDSTART          0002 N   441#  442  4274
MDUL             0058 V   253# 4285  4286
MEMSIZEINT       0012 N   746# 4226
MEXISTS          1FD7 L  4320  4324#
MFL              005A V   254# 4268
MFLDONE          1F9C L  4273  4288#
MINFLAG          0003 N   465#
MORETHAN128K     2007 L  4334  4340#
MORETHAN64K      1FEF L  4327  4332#
MOTORFLAGS       1CA6 V   891  3799  3804  4082#
MOTOROFFCOUNTER  1CA8 V   884   886  3795  4084#
MOTORON          1863 L  3770  3788#
MOTORONDONE      1892 L  3801  3808  3819#
MTESTDONE        1FC9 L  4307  4317#
NAK              0015 N   501# 2937
NCCB             0C12 V   553#
NEXT64K          1FA3 L  4298# 4315
NEXTMDUL         1F8B L  4280# 4284
NEXTMFL          1F73 L  4269# 4277
```

```
NLCB                0C13 V   554#
NMIINT              0002 N   740#  755
NOBELL              1143 L  1639  1658#
NOKEY               10CE L  1444  1475  1524# 1574
NORESTORE           20CE L  4476  4492#
NOSAVE              2078 L  4438  4457#
NOSECONDFLAG        001D L   896   901#
NOSSTR              1952 V  2581  3251#
NOSTACKRESTORE      0D45 L   921   925#
NOTACTIONKEY        1069 L  1464  1472#
NUL                 0000 N   479# 2928
NUMACTIONKEYS       0006 N  1462  2791#
NUMLOCK             0045 N  2782# 2798
NUMSEC              1CA1 V  3658  3666  3690  3715  3719  3870  3902  4073#
NUMSTR              1972 V  2623  2625  3260#
NUMVIRCONS          0004 N   534#  552  1300
NVCNS               0C11 V   552# 1601  4543
NXTTRACK            1AF5 L  3714  3723#
ONEBYTEINT          0003 N   741#  754
OPENNXT             15DE L  2601# 2610
OPENVEC             0088 V   270# 2598
OSINT               00E0 N   749#
OSSEG               0040 V   245#
OSVERNUM            007C V   264#
PARALLELWAIT        20DC L  4503# 4505
PARRET              1652 L  2693  2696#
PCONS               0020 N   352#  353
PDELAY              008D N   282# 2500  3810
PDISK               0012 N   348#  349
PDISPATCH           008E N   283#
PDLEN               0030 N   355#
PDONE               2238 L  4661  4667#
PFKIDTABLE          1681 V  2810#
PFLAG               0006 N   345#  346  1254
PHYSRET             1B62 L  3778  3785#
PICEVENPORT         0020 N   777#  916  1050  1159
PICNSEOI            0020 N   780#  915  1049  1158
PICODDPORT          0021 N   778# 4394
PLINK               0000 N   341#  342
PLIST               0024 N   353# 2555
PMEM                0016 N   350#  351
PMSGL               2228 L  4658# 4666
PNAME               0008 N   346#  347  1239  2558
POLL                0C83 L   702# 2710
POLLTABLE           0C93 V   699   723#
PORTA               0060 N   805#
PORTB               0061 N   806#  876   878  1644  1646
PORTC               0062 N   807#
POS                 0C8B L   710   712#
PPARENT             001E N   351#  352
PPDADR              009C N   284#
PPRIOR              0005 N   344#  345
PRINTABLE           10CA L  1518  1520#
```

```
PRINTMSG          2224 L  1246  4547  4651#
PRNSTR            1962 V  2543  2545  3255#
PSH               000F N  3449#
PSTAT             0004 N   343#  344
PTERM             008F N   286#  1256
PTHREAD           0002 N   342#  343
PUDA              0010 N   347#  348  1255
PURSTR            194A V  2587  3249#
PUSER             0013 N   349#  350
QLR               0074 V   259#
RBYTES            0002 N  3345#  3766
RCEND             13AC L  3836#
RDS               001E N   510#  2930
RDTL              00FF N  3348#  3769
READHDISK         1980 L  3505#  4103
READSDFLOPPY      19C3 L  3520#  4096  4097
READTBL           1EA9 V  3503  4096#
RECAL             1893 L  3781  3822#
RECALLOOP         1841 L  3772#  3783
RECALMAX          000A N  3771  4036#
RECALS            1C88 V  3771  3782  4038#
REPORT            1252 L  1902#
RESET             2239 L  1438  4671#
RESETFLAG         0072 V  4677  4689#
RESTORE           125F L  1921#  3004
RETRIES           1C87 V  3773  3779  4037#
RETRYLOOP         1846 L  3774#  3780
RETRYMAX          0005 N  3773  4035#
REVDONE           137F L  2146  2160#  2170
REVSWAP           136C L  2150#  2172
RGAP              002A N  3347#  3768
RIGHT             11FF L  1831#  2997
RLR               0068 V   256#  1236  1248
RUMPRINTERINT     0017 N   748#
RUMRESET          0000 L  4678  4683#
RUMRS232INT       0014 N   747#  758
ROWOK             127E L  1943  1945#
ROWSPERSCREEN     0018 N  1295#  1297  1742  1747  1864  1896  1942  2078  2097  2100
RRIGHT            1206 L  1681  1837  1839#
RSECTORS          0008 N  3346#  3767
SAVE              1253 L  1907#  3003
SCANCODE          1673 V  1432  1473  1555  2768#
SCANDONE          10D2 L  1522  1530#
SCANTRANS         102D L  1405  1423#
SCAPSLOCK         15E6 L  2606  2612#
SCREENSIZ         0780 N  1297#  1300  1305  1810  1999  2431  2449  4541
SCREENSSIZE       03C0 N   575  1300#
SCREENSTRUCTADDR  1826 V  1618  2430  2444  3114#
SCREENSTRUCTURES  17F3 V  3072#  4529
SCROLLDOWN        13F6 L  1897  2080  2300#
SCROLLUP          13C7 L  1748  2098  2266#
SCTRLO            1588 L  2565  2568#
SCTRLP            1596 L  2566  2569  2571#
```

```
SCTRLS                157B L  2563#
SDISPLAY              161F L  2630  2636#
SDLOOP                1630 L  2646# 2649
SECFLAG               0002 N   464#  898
SECTOR                000A V  3434# 3540  3618  3667  3693  3728  3867
SECTORSPERTRACK       0008 N  3296# 3297  3617  4052
SEEK                  1BAF L  3775  3840#
SELRET                19A0 L  3471  3475#
SENSEINTERRUPT        1C37 L  3837  3878  3932#
SETFLAG               0C6B L   651#  871   899   905  1030  1147
SETPHYSICALCURSO      1444 L  1734  1773  1828  1842  1856  1871  1885  1930  1957  2087
                              2358# 2465
SETR                  1471 L  2396  2400#
SETUPES               1461 L  1661  1809  1971  1994  2014  2036  2059  2113  2275  2309
                              2384#
SETUPINTERRUPTS       2018 L  4392#
SETUPVIDEO            20F2 L  4510#
SEXIT                 1637 L  2518  2530  2652#
SGLTOPENVEC           15D0 L  2592#
SHFTBIT               0002 N  1488  1494  2788# 2803  2804
SHFTLEFT              002A N  2778# 2796
SHFTRIGHT             0036 N  2779# 2795
SHI                   000F N   494# 2938
SHIFTTABLE            16E9 L  1490  1497  2883#
SIDEOK                18BE L  3859#
SIGND                 21AD V  4608#
SIGNMDISK             21FC V  4325  4624#
SIGNMDSIZE0           221E V  4335  4341  4631#
SIGNMDSIZE1           221F V  4328  4336  4342  4632#
SIGNMEM               21F7 V  4229  4623#
SIGNON                2151 V  4546  4590#
SIGNSP                2103 V  4516#
SL1                   0C0C N   541#  543
SL10                  0EC0 N  1123# 1125
SL11                  0F0B N  1178# 1180
SL12                  0F6A N  1215# 1217
SL13                  0FB0 N  1269# 1271
SL14                  0FE3 N  1349# 1351
SL15                  1673 N  2764# 2766
SL16                  1991 N  3453# 3455
SL17                  1C87 N  4031# 4033
SL18                  1F47 N  4210# 4212
SL19                  212D N  4580# 4582
SL2                   0C3D N   591#  593
SL20                  2224 N  4646# 4648
SL3                   0C4F N   624#  626
SL4                   0C6B N   646#  648
SL5                   0C93 N   719#  721
SL6                   0C9F N   819#  821
SL7                   0D52 N   939#  942
SL8                   0DB5 N   984#  986
SL9                   0E3C N  1070# 1072
SMODE                 15AD L  2572  2577#
```

```
SSPECIAL            1508 L  2511  2514#
SSPECIALOK          1518 L  2516  2520#
SSROW               0008 N  1742  1768  1826  1864  1870  1876  1884  1893  1946  2017
                            2039  2077  2096  2100  2347  3052# 3053
SSSCREENSEG         0006 N  2398  2432  2452  3050# 3051  4532
SSTEPINT            0001 N   739#  752
SSXY                0008 N  1913  1927  2099  2102  3051# 3052  3054
STATUSMSGSTART      18EA V  2533  2637  3216#
STEND               18E9 L  3877#
STISLOCKED          18E8 V  2497  2654  3208#
STISSPECIAL         18E9 V  2512  2515  2521  2527  3209#
STX                 0002 N   481# 2946
SUBB                001A N   506# 2945
SUPERVISOR          0C08 V   539#  681
SUPERVISORD         0C08 V   537#  539
SUPERVISORS         0C0A V   538#
SUPIF               0C71 L   657   669#  715  1259  2502  3812
SUPMOD              0000 V   241#
SWAIT               14E4 L  2495# 2504
SWDONE              14DB L  2460  2463#
SWNOCURSOR          14D8 L  2458  2461#
SWRAP               1610 L  2622  2628#
SYN                 0016 N   502# 2946
SYSDAT              0C06 V   536#  830   836   995   998  1133  1136  1232  3686  4474
SYSDISK             0048 V   249#
TCNT                0D84 V   831   839   920   972#
TEMPDISK            0050 V   251#
TESTFORALT          106F L  1504  1506  1510  1512  1515#
TESTFORKEYPAD       1082 L  1480  1484#
TESTFORMDISK        1F9C L  4292#
TESTFORSHIFT        1098 L  1485  1487  1493#
THRDRT              0072 V   258#
TICK                0C0C V   547#  902
TICKCOUNTER         0D53 V   895   897   945#
TICKFLAG            0001 N   463#  904
TICKINT             0008 N   742#  755
TICKSPERSECOND      003C N   545#  548   897   945  3809
TICKSSEC            0C0D V   548#
TIMER0REG           0040 N   793# 4407  4409
TIMER1000HZ         0533 N   799# 4413
TIMER1REG           0041 N   794#
TIMER2REG           0042 N   795# 4414  4416
TIMER60HZ           4DAE N   798# 4406
TIMERCHANNEL        0000 N   785#
TIMERCMDREG         0043 N   796# 4412
TNOSWITCH           0CBC L   832   838#
TODDAY              007E V   265#
TODHOUR             0080 V   266#
TODMIN              0081 V   267#
TODSEC              0082 V   268#
TOGGLEACTION        10EC L  1556  1566#
TRACK               000C V  3433# 3536  3727  3848
TRACKMCNT           1CA2 V  3630  3659  3693  3713  4077#
```

```
TRACKSPERDISK     0028 N   3298# 3299
TRANSLATE         1012 L   1482  1489  1491  1495  1499#
TRANSTABLE        0C3A V    580#
TRUE              FFFF N    450#  881  1227  1384  2496  2498  2515  2521  2527  3614
                           4437  4475
TSECONDS          009B N    294#
UNEXMSG           0FB0 V   1245  1273#
UNEXMSGPD         0FBE V   1235  1240  1276#
UNOTD             0F72 L   1228  1230#
UP                1230 L   1874# 1894  3000
UPDATESTATUS      14E0 L   1570  2248  2258  2475#
UPWITHSCROLL      1241 L   1888# 3002
US                001F N    511# 2932
VALIDKEY          104E L   1442  1458#
VERSION           0078 V    261#
VIDEOINIT         2119 L   4514  4552#
VIDEOINITL        2116 L   4562# 4567
VIDEOOFF          0021 N   1333#
VIDEOON           0029 N   1332# 4570
VT                0008 N    490# 2942
WAITFLAG          0C6F L    660# 1387  1649  3835  3875  3926
WRITEMDISK        1A00 L   3534# 4113
WRITESDFLOPPY     1A13 L   3600# 4106  4107
WRITETBL          1EC9 V   3582  4106#
WRPSTR            1979 V   2631  2633  3262#
XANDY             126D L   1933# 3005
XIOSFUNCS         000E N    613   628#
XLT               0000 N   3442#
XLT0              0000 N   4119  4185# 4186
XLT1              0000 N   4130  4186#
XOFF              0013 N    515#
XON               0011 N    514#
XYCOL             1287 L   1947  1949#
XYERR             1298 L   1944  1958#
XYROW             1273 L   1938  1940#
XYSETCOL          128F L   1952  1954#
```