```
                page    58,132
                title   network_virtual_disk_driver
;*****************************************************************************
;*                                                                           *
;*      Virtual Disk Driver                                  April 11, 1983  *
;*          Version 3.0                                      J. Whitnell*
;*                                                                           *
;*      Modified from Zynar's DOS 1.1 virtual disk driver.                   *
;*                                                                           *
;*      This driver has knowledge of the DOS 2.0 drive parameter blocks*
;*      which appear only in Microsoft OEM documentation.  If you are*
;*      using version 3.0 or later, you should check this. (I did, and*
;*      and version 3.0 gives some values we need.  We still use dpbs,*
;*      however   jdw).                                                      *
;*                                                                           *
;*****************************************************************************
;*
;*      History
;*
;*      JDW     5-18-83
;*              Support to fix DOS bug (#572).  If character and block
;*              device drivers are mixed together, DOS allocates tables
;*              for both in the same memory.  To get around the problem,
;*              we must allocate an extra chunk of memory and tell DOS
;*              about it.  Hence, the vdisk is inited first, and calculates
;*              enough space for the drive table that DOS needs.  It
;*              then returns drive_end as its end.  DOS then builds
;*              its tables and inits the other character devices.  They
;*              return the value (from your_end) that vdisk calculates
;*              so that DOS puts its other tables past the end.
;*      JDW     5-28-83
;*              Fixes to remove address table.  We now just store arcnet
;*              address and figure things out from there.  Also added DIM
;*              support, offset drives from 1 instead of 0.  Changed interface
;*              to vdisk_virt_io to accept station # in DH.
;*      cwp     7-31-83
;*              Removed logic in real_build_bpb to optionally copy 2.0 bpb,
;*                now always copies.
;*              Fixed output with verify
;*              Made max_drives and clicking setable options
;*      PLS     8-9-83
;*              Fix to compute length of a DOS 1.1-zformatted volume properly
;*      cwp     8-15-83
;*              Fix to always zap media desc. in bpb, not only if size changes
;*              Fix to keep Media_desc below 128 to keep from looking like ibm
;*      cwp     9-13-83
;*              added our own pointer to ncmd driver header
;*              added encrypted vector to setiobsw
;*              added setiobsw routine and iob switch timeout logic
;*              fixed non-$fe vprints
;*              fixed msdos bug #573. see comment for var media_desc
;*      JDW     9-27-83         Version 2.1
;*              Break driver again by adding mirror volumes.
;*              Also added timeouts and our_drive to header.
```

```
;*              Also added ve_timeout error code so everyone knows we
;*              couldn't talk to the other end.
;*      JDW     11-27-83        Version 3.0
;*              Use cwp's level 4 instead of Zynar's.
;*      JDW     2-16-84         Version 3.1b
;*              Add code to remap drives from floppy/hard disk to us for
;*              boot from net.
;*      JDW     2-22-84         Version 3.1c
;*              Add more rb's.
;*      GIGI    4-9-84          Version 3.1d
;*              Move alloc_more_rbs to l4asm so it gets done for ram version
;*                of level 4 (nettemp.sys).  Call alloc_more_rbs after calling
;*                l4_init.
;*              Use common stack as other drivers.
;*              Change vdisk_init to initialize all bpb's (not just starting
;*                at our_drive).
;*      GIGI    4-20-84
;*              Fix vdl4init to copy dims from Nic ram correctly.
;*              Trace off.
;*  JDW/GIGI    4-23-84
;*              Add remapped3 for drive C (fix for booting virtual C on
;*                floppyless machine).
;*      GIGI    4-24-84         Version 3.1
;*              Change version for FCS.  All trace off.
;*      GIGI    4-25-84
;*              Change start_noise/end_noise to half_click.
;*      GIGI    7-02-84
;*              Remove call to build_bpb in read_disk_info so ?drive would
;*                not clear MediaChanged flag.
;*              Version 3.1.1.
;*      GIGI    7-13-84
;*              Support Non-IBM DOS by checking for OEM_Name ("IBM  2.0") then
;*                check for NIBM_OEM ("IBM  2.0", to be patched) in BPB.
;*              Version 3.1.2.
;*      GIGI    8-1-84
;*              Change version to 3.1.3a to include niccode with int13 fix.
;*      PLS     9-12-84
;*              Add IBM/Denver RPQ drive-remapping logic to FindDrive.  Add
;*                conditional assembly flag "denver".  No change to non-RPQ
;*                versions.  RPQ is version 3.1.4a.
;*      GIGI    12-04-84
;*              Change version to 3.1.3b for virtual printer initialization
;*                fixes (for Ctrl PrtSc from BASIC without parallel printer
;*                interface).
;*      DEBRA   2-25-85
;*              Change version to 3.2a for adding retry on disk errors.
;*                Number of times to retry is set in header block.
;*      DEBRA   3-14-85
;*              Change version to 3.2b for adding alt-222 option on header block.
;*                so that user can disable alt-222 and not to close the virtual
;*                printer from the keyboard. Default is alt-222 on. This option
;*                can be changed by running netconfg.exe.
;*      esp     23 Jul 1985
;*              fix vdiskio bug, alt222 option bug, and added code to download
```

```
                        ;*              dim info into our NIC RAM.  change version to 3.2c
                        ;*      esp     25 Jul 1985
                        ;*              change version to 3.2d.  fix problem with changing fs when
                        ;*              issuing network commands.
                        ;*      jdw     8 Aug 1985      V 3.2e
                        ;*              Add timeout byte to header for virtual printer
                        ;*      jdw     13 Aug 1985     V 3.2f
                        ;*              Set timeout to 15 seconds, add cursor init code.
                        ;*      jdw     26 Aug 1985     V 3.2g
                        ;*              For DOS 3.x, get drive number from init block for FindDrive.
                        ;*              Also don't fix DOS 2.x bug if running DOS 3.x (init_vidsk)
                        ;*      jdw     19 Sept 1985    V 3.2h
                        ;*              Fix bsave in ncmd.
                        ;*      gkn     28 Oct 1985     V 3.2i
                        ;*              Added code by conditional assembly to support Zenith 120.
                        ;*              Added in modules vdisk.asm and vdinit.asm.
                        ;*              Added conditional assembly to support Zenith 120 speaker click
                        ;*              Added speaker click changes in vddef.asm and vdmisc.asm
                        ;*      gkn     31 Dec 1985     V 3.2j
                        ;*              Added to vddef.asm, 'driver_busy'and 'rb_busy'. Also added
                        ;*              notready condition to 'status' macro. Also added 'extern rb_ptr:dword'
                        ;*              Added checking and returning error in vddos.asm if drivers are
                        ;*              busy. Error is returned to DOS in request header status field.
                        ;*              Output status function was added to vddos driver to return
                        ;*              status if RB is free or connected.
                        ;*
                        ;Copyright (c) 1982, 1983 Zynar Ltd.
                        ;Copyright (c) 1983 Nestar Systems, Inc.         Version 2.0
                        ;Copyright (c) 1983, 1984 Nestar Systems, Inc.   Version 3.0
                        ;Copyright (c) 1984 Nestar Systems, Inc.         Version 3.1
                        ;Copyright (c) 1985 Nestar Systems, Inc.         Version 3.2
                                        .sall
                                        name    vdisk_driver

                                        ELSE
                                        %OUT    Pass 2 ...
                                        ENDIF


= 0000                  trace           equ     0
= 0000                  vers3           equ     0
= 0000                  for_z120        equ     0       ; one if to run on Zenith 120
                        ;denver         equ     1       ;denver should be defined only for Denver RPQ

0000                    CSEG            segment public 'CODE'

                                        assume  CS:CSEG,DS:CSEG,ES:NOTHING,SS:NOTHING

                C                       include vddbdcl.asm                     ; Debug definitions
                C
                C
                C       writeln         macro   str
                C                       local   wr1,wr2
                C       if      TRACE
```

```
C                       push    si
C                       lea     si,wr2
C                       call    print_string
C                       pop     si
C                       jmp     wr1
C   wr2:                db      str,0aH,0dH,0H
C   wr1:
C   endif
C                       endm
C   ;
C
C
C   write               macro   str
C                       local   wr1,wr2
C   if      TRACE
C                       push    si
C                       lea     si,wr2
C                       call    print_string
C                       pop     si
C                       jmp     wr1
C   wr2:                db      str,0H
C   wr1:
C   endif
C                       endm
C   ;
C   writeint            macro   int
C   if                  trace
C                       push    ax
C                       mov     ax,int
C                       call    print_word
C                       pop     ax
C   endif
C                       endm
C
C   ;
C   writebyte           macro   byt
C   if                  trace
C                       push    ax
C                       mov     al,byt
C                       call    print_byte
C                       pop     ax
C   endif
C                       endm
C   ;
C   endif   ; TRACE

            endif   ; for_z120
            extrn   print_string:near, print_crlf:near,print_byte:near
            extrn   print_word: near, print_hex : near,print_char:near

                    %OUT    ..........VDISK_DEFINITIONS/RAM
C                   include vdbpb.asm                       ; bpb defintion
C
C   bpb             struc
```

```
0000  ????              C                   BBytesPerSect   dw      ?
0002  ??                C                   BSectPerClust   db      ?
0003  ????              C                   BResvSect       dw      ?
0005  ??                C                   BFATCount       db      ?
0006  ????              C                   BDirEntries     dw      ?
0008  ????              C                   BSectCount      dw      ?
000A  ??                C                   BMediaDesc      db      ?
000B  ????              C                   BFATSize        dw      ?
                        C   ;
000D  ????              C                   BSectsPerTrack  dw      ?
000F  ????              C                   BHeadCount      dw      ?
0011  ????              C                   BHiddenSects    dw      ?
                        C   ; Nestar stuff.
0013  ??                C                   BMediaChanged   db      ?
                        C   ; FS address
0014  ??                C                   BPriFS          db      ?
0015  ??                C                   BSecFS          db      ?
                        C   ; FS state
0016  ??                C                   BPriRead        db      ?
0017  ??                C                   BPriWrite       db      ?
0018  ??                C                   BSecRead        db      ?
0019  ??                C                   BSecWrite       db      ?
                        C   ; IOB error code
001A  ??                C                   BPriErr         db      ?
001B  ??                C                   BSecErr         db      ?
                        C   ; Dim stuff
001C  ??                C                   BFileType       db      ?
001D  ??                C                   BFileSubType    db      ?
001E  ??                C                   BAccess         db      ?
001F  ??                C                   BShr            db      ?
0020                    C   bpb             ends
                        C
= 0052                  C   ReadState       equ     'R'             ; Read from this server
= 0043                  C   CompState       equ     'C'             ; Read and compare to other server
= 0057                  C   WriteState      equ     'W'             ; Write to this server
= 002D                  C   NotUsedState    equ     '-'             ; Something wrong. Don't use this server
                        C
= 0001                  C   BPriDim         equ     1
= 0002                  C   BSecDim         equ     2
                        C
= 0020                  C   NotMountedType  equ     ' '             ; This drive has nothing mounted on it

                        C                   include vddef.asm                       ;ram/definitions
                        C
                        C                   subttl  vdisk_definitions/ram
```

```
                        C                page
                        C
                        C
                        C  ;return values for interface routines:
                        C  ;-----------------------------------
                        C
 = FFFF                 C  TRUE           equ     0FFFFH
 = 0000                 C  FALSE          equ     00000H
 = 0000                 C  NIL            equ     00000H
                        C
                        C
                        C
                        C
                        C  ;macros:
                        C  ;------
                        C
                        C  ;
                        C  ;Set status byte of Request Header
                        C  ;
                        C  status       macro               state,err,rc
                        C               ifidn               <state>,<done>
                        C                or                 es:word ptr srh_sta_fld[bx],0100h
                        C               endif
                        C               ifidn               <state>,<busy>
                        C                or                 es:word ptr srh_sta_fld[bx],0200h
                        C               endif
                        C               ifidn               <err>,<error>
                        C                or                 es:word ptr srh_sta_fld[bx],8000h
                        C               endif
                        C               ifidn               <state>,<notready>
                        C                or                 es:word ptr srh_sta_fld[bx],8002h
                        C               endif
                        C               ifnb                <rc>
                        C                or                 es:word ptr srh_sta_fld[bx],rc
                        C               endif
                        C               endm
                        C
                        C
                        C
                        C  ;Real long jump
                        C  ;--------------
                        C  bnz            macro   to
                        C                 local   b
                        C                 jz      b
                        C                 jmp     to
                        C  b:
                        C                 endm
                        C
                        C  ;
                        C  ;Long call in local segment
                        C  ;--------------------------
                        C  ;
                        C  lcall          macro   longdest        ; Call to long procedure in current seg
                        C                 push    cs              ; CS is on stack for return
```

```
                        C                call    longdest        ; Then do a near call.
                        C                endm
                        C
                        C  ;btye sex along the wire is hi-order first:
                        C  ;----------------------------------------
                        C
                        C  flip          macro   hi,lo
                        C                xchg    hi,lo
                        C                endm
                        C
                        C
                        C  ;*******************************
                        C  ;IOB commands:
                        C  ;------------
                        C
= 0001                  C  IOB_read              equ     1
= 0002                  C  IOB_write             equ     2
= 0003                  C  IOB_special           equ     3               ;special iob's to read execut only
= 0004                  C  IOB_init              equ     4               ;not currently used
                        C
                        C
                        C
                        C  ;IOB error codes:
                        C  ;---------------
                        C
= 0000                  C  ve_ok                 equ     0
= 0001                  C  ve_no_drive           equ     1       ;no drive mounted           => BIOS_timeout
= 0002                  C  ve_illegal_op         equ     2       ;not read/write/init        => BIOS_bad_nec
                        C                                        ;  or not special for execute only
= 0003                  C  ve_bad_machine        equ     3       ;server not know us         => BIOS_bad_nec
= 0004                  C  ve_no_read            equ     4       ;no read_acces              => BIOS_write_protect
= 0005                  C  ve_no_write           equ     5       ;no write access            => BIOS_write_protect
= 0006                  C  ve_bad_block          equ     6       ;                           => BIOS_record_not_fnd
= 0007                  C  ve_no_descr_write     equ     7       ;no write to descriptor     => BIOS_record_not_fnd
= 0008                  C  ve_bad_disk           equ     8       ;peer error (NFS disk)      => BIOS_bad_nec
= 0009                  C  ve_restriction        equ     9       ;implementation restriction => BIOS_bad_nec
= 000A                  C  ve_level4             equ    10       ;Maude level 4 failure      => BIOS_timeout
= 000B                  C  ve_protocol           equ    11       ;peer protocol error        => BIOS_timeout
= 000C                  C  ve_internal           equ    12       ;our error (= client abort) => BIOS_timeout
= 000C                  C  ve_client_abort       equ    12       ;           "               => BIOS_timeout
= 000D                  C  ve_timeout            equ    13       ; our timeout.
                        C
                        C
                        C
                        C  ;interupt vector numbers:
                        C  ;-----------------------
                        C
                        C  ife trace
= 000E                  C  write_tty             equ     0EH             ;write screen function
= 0010                  C  video_call            equ     10H             ;int number for screen action
                        C  endif
                        C
= 0011                  C  equip_call            equ     11H             ;rom bios equipment vector
= 0013                  C  diskio_call           equ     13H             ;rom bios disk i/o vector
```

```
= 0016                  C  kbdio_call              equ     16H             ;rom bios keyboard i/o vector
= 001B                  C  break_call              equ     1BH             ;rom bios break vector
= 00FD                  C  map_int                 equ    0FDH             ;map interrupt vector
                        C
                        C  ;port addresses on motherboard used to make noise:
                        C  ;-------------------------------------------------
                        C
= 0040                  C  timer_zero              equ     40h             ;base of timer chip regs.
= 0042                  C  timer_two               equ     timer_zero+2
= 0043                  C  timer_ctrl              equ     timer_zero+3    ;control reg. for timer
= 0002                  C  spkr_enable             equ     2               ;speaker enable
                        C  else             ; not for_z120
= 0061                  C  spkr_port               equ     61h             ;port that controls speaker
                        C  endif            ; for_z120
                        C
                        C  ;drive and address table definitions:
                        C  ;-----------------------------------
= 001A                  C  def_max_drive           equ     26
                        C
                        C
                        C  ;DOS calls:
                        C  ;-------------
= 0052                  C  GET_IN_VARS             equ     52H             ; Return internal vars ptr in es:bx
= 0030                  C  DOS_VERS                equ     30H             ; Return DOS version number
                        C
                        C
                        C  ;DOS 2.0/2.1 internal variables defintions:
                        C  ;-----------------------------------------
= 0000                  C  dpb_chain               equ     0               ; Chain of drive parameter blocks
= 0010                  C  last_drive              equ     10H             ; Maximum number of drives in sys
= 0017                  C  drivers_chain           equ     17H             ; Chain of all drivers
                        C
                        C
                        C  ;Drive Parameter Block (for DOS 2.0 ONLY!):
                        C  ;-----------------------------------------
                        C  ;       Copied from DOSSYM.ASM in MicroSoft OEM documentation.
                        C  ;       Note: This documentation is marked Caveat Programmer by
                        C  ;       Microsoft.  This structure should be checked in new
                        C  ;       versions of DOS.
                        C  ;
= 0040                  C  DIRSTRLEN       equ     64
                        C
                        C  dpb             struc
0000  ??                C  dpb_drive               db      ?       ; Logical drive number (A=0, B=1)
0001  ??                C  dpb_UNIT                db      ?       ; Drive unit number
0002  ????              C  dpb_sector_size         dw      ?       ; Size of physical sectors in bytes
0004  ??                C  dpb_cluster_mask        db      ?       ; Sectors/cluster - 1
0005  ??                C  dpb_cluster_shift       db      ?       ; Log2 of sectors/cluster
0006  ????              C  dpb_first_FAT           dw      ?       ; Starting record of FATs
0008  ??                C  dpb_FAT_count           db      ?       ; Number of FATs for this drive
0009  ????              C  dpb_root_entries        dw      ?       ; Number of directory entries
000B  ????              C  dpb_first_sector        dw      ?       ; First sector of first cluster
000D  ????              C  dpb_max_cluster         dw      ?       ; Number of clusters on drive + 1
000F  ??                C  dpb_FAT_size            db      ?       ; Number of records occupied by FAT
```

```
0010  ????                C  dpb_dir_sector          dw      ?       ; Starting sector of directory
0012  ????????            C  dpb_driver_addr         dd      ?       ; Pointer to driver
0016  ??                  C  dpb_media               db      ?       ; Media byte
0017  ??                  C  dpb_first_access        db      ?       ; This is initialized to -1 to force
                          C                                          ; a media check the first time this DPB
                          C                                          ; is used
0018  ????????            C  dpb_next_dpb            dd      ?       ; Pointer to next dpb
001C  ????                C  dpb_current_dir         dw      ?       ; Cluster number of start of current directory
                          C                                          ; 0 indicates root, -1 indicates invalid
                          C                                          ; (disk ? changed)
001E  0040[               C  dpb_dir_text            db      DIRSTRLEN dup (?)
        ??
                        ] C
                          C
                          C                                          ; ASCIZ string of current directory
005E                      C  dpb                     ends
                          C
= 005E                    C  dpbsiz                  equ     size dpb
                          C
                          C
                          C  ;Long Pointer Struc
                          C  ;---- ------- -----
                          C  pointer         struc
0000  ????                C  off             dw      ?
0002  ????                C  segp            dw      ?
0004                      C  pointer         ends
                          C
                          C
                          C
                          C  ;miscellaneous constants:
                          C  ;-----------------------
                          C
= 0020                    C  io_limit                equ     020H            ; Max blocks we can read
= 005E                    C  DOS_disk_entry_size     equ     05EH            ; Size of DOS's drive table entry
```

```
                        C               page
                        C
                        C
                        C
                        C  ;the first executable instruction in the VDISK code segment must be a DOS
                        C  ;2.0 header block.
                        C               extrn   stack_top:near, sp_save:word
                        C               extrn   ss_save:word, stack_use:byte
                        C               extrn   nc_head:near,driver_end:near,l4_init:near
                        C               extrn   vp_vol_unit:byte, rb_ptr : dword
                        C               public  your_end, my_end, half_click
                        C               public  our_drive, db_flag, driver_busy, rb_busy
                        C               public  click_on, more_rbs, rty_num, alt222, timeout
                        C               extrn   vdisk_io: near
                        C               extrn   alloc_more_rbs:near, l4_vars:dword
                        C
0000  0000 E            C  next_dev     dw   nc_head       ;pointer to next device
0002  0000              C               dw   0
0004  2000              C  attribute    dw   2000h         ;block device (non-ibm format)
0006  050D R            C  strategy     dw   dev_strategy  ;pointer to device strategy
0008  0518 R            C  interrupt    dw   dev_int       ;pointer to device interrupt handler
000A  ??                C  dev_name     db   ?             ;number of block devices, filled in by init_vdisk
000B  0007[             C               db   7 dup (?)     ;7 bytes of filler
          ??            C
                      ] C
                        C
                        C  ;
                        C  ; **** magic ****
                        C  ;  These constants are patched by NETCONFIG.EXE and must follow the
                        C  ;  device driver header.
                        C  ;
= 1776                  C  secret_bias     equ     1776h                   ;offset bias used to hide vectors
                        C  ;
0012  001A              C  max_drive       dw      def_max_drive           ;soft limit to drives
0014  0002              C  click_on        dw      spkr_enable             ;2 for noise, 0 for silent
0016  0000 E            C                  dw      nc_head                 ;pointer to ncmd header
0018  2414 R            C                  dw      secret_bias + offset setiobsw
001A  01                C  db_flag         db      1               ;one for extra info.
                        C                                          ;zero for the quiet life...
001B  ??                C  our_drive       db      ?               ; Driver number we start at.
                        C  ;
001C  0004              C  more_rbs        dw      4               ; Number of rb's to add in
                        C  ;
001E  0005              C  rty_num         dw      5               ; Number of times to retry when timeouts
                        C  ;
0020  0001              C  alt222          dw      1               ; 1 = alt-222 on, 0 = alt-222 off
0022  010E              C  timeout         dw      15*18           ; Number of ticks (vprn timeout)
                        C  ;
                        C
                        C  ;
0024  56 44 49 53 4B 20 56  C  version                 db      'VDISK V3.2j, 12-31-85,  gkn,jdw,tmd,dsj,cwp,gigi,pls,debra,esp,/alt/cat
      33 2E 32 6A 2C 20 31  C
      32 2D 33 31 2D 38 35  C  and a cast of thousands'
```

```
       2C 20 20 67 6B 6E 2C  C
       6A 64 77 2C 74 6D 64  C
       2C 64 73 6A 2C 63 77  C
       70 2C 67 69 67 69 2C  C
       70 6C 73 2C 64 65 62  C
       72 61 2C 65 73 70 2C  C
       2F 61 6C 74 2F 63 61  C
       74 20 61 6E 64 20 61  C
       20 63 61 73 74 20 6F  C
       66 20 74 68 6F 75 73  C
       61 6E 64 73           C
                             C
                             C  ; *** end of magic ***
                             C
                             C  ;
0083   00                    C  driver_busy             db      0              ;0 if not busy, 1 if network.sys
                             C                                                 ; drivers are busy
0084   00                    C  rb_busy                 db      0              ;0 if not busy, 1 if RB conn_status
                             C                                                 ; is not connected
                             C
0085   ????                  C  rh_off                  dw      ?              ;request header offset
0087   ????                  C  rh_seg                  dw      ?              ;request header segment
                             C
                             C  ;vars used in special iob processing
                             C  ;
0089   00                    C  iob_switch              db      0              ;non-zero for special read iobs
008A   0000                  C  iob_timeout             dw      0              ;in 18.2 ticks/sec.
                             C  ;
008C   ????                  C  end_special_lo          dw      ?              ;expiration time for special iobs.
008E   ????                  C  end_special_hi          dw      ?
                             C  ;
                             C                  public  remapped1              ; For vdl4init
0090   FF                    C  remapped1               db      0FFH           ; Drive remapped by us
0091   FF                    C  remapped2               db      0FFH           ; Drive remapped by us
0092   FF                    C  remapped3               db      0FFH           ; Drive remapped by us
0093   ??                    C  special_drive           db      ?              ;only acceptable drive for
                             C                                                 ;  special iobs, set by setiobsw
                             C  ;
0094   176F                  C  int_1a_1                dw      5999           ;an "int 1a" = (5999+7723) div 2
= 1E2B                       C  int_1a_2                equ     7723
                             C  ;
                             C  ;
0096   ????                  C  your_end                dw      ?              ; Offset to end of driver.
0098   0000 E                C  my_end                  dw      offset driver_end      ; My end of driver.
                             C  ;
009A   0000                  C  control_word            dw      0              ;for NIC_link board,routine
                             C  ;
009C   00 00 00 00           C  old_13_vector           dd      0              ;stash for old diskio vector
00A0   00 00 00 00           C  old_11_vector           dd      0              ;stash for old equipment vector
00A4   00 00 00 00           C  old_1B_vector           dd      0              ;stash for old break vector
                             C                                                 ;vector
                             C  ;
                             C  ;DOS appears to want more than just a disk change flag to cause it to
                             C  ;update it's internal bpb's
```

```
                        C  ;This byte is incremented each time a disk change occurs, we think this
                        C  ;convinces DOS that the disk has really changed
                        C  ;
00A8  01                C  Media_desc              db      1               ; Our current media.
                        C
                        C  ;Locals for various routines
                        C  ;--------------------------
00A9  ??                C  save_drive              db      ?               ; For build_bpb
00AA  ??                C  io_fs                   db      ?               ; File server I/O is sent to.
                        C
                        C  ;Constants stored in memory (Hopefully they don't change)
                        C  ;-------------------------------------------------------
00AB  0020              C  type_bpb                dw      type bpb        ; Idiots couldn't put immediate mode on mul
00AD  0200              C  bytes_per_sector        dw      0200H           ; What is being used
00AF  0200              C  block_size              dw      0200H           ; For non-DOS mounts.
00B1  49 42 4D 20 20 32 2E  C  OEM_Name            db      'IBM  2.0'      ; Our OEM Name (8 bytes)
      30                C
00B9  50 4C 41 4E 34 30 30  C  ID                  db      'PLAN4000/IBMPC/DOS1.1'
      30 2F 49 42 4D 50 43  C
      2F 44 4F 53 31 2E 31  C
                        C
= 0015                  C  ID_len                  equ     $ - offset ID   ; DOS 1.1 id.
00CE  4F 45 4D 2D 3E    C  marker                  db      'OEM->'
00D3  49 42 4D 20 20 32 2E  C  NIBM_OEM            db      'IBM  2.0'      ; alternate OEM Name to be patched
      30                C
                        C
                        C
                        C
                        C
                        C  ;structures:
                        C  ;----------
                        C
                        C  etna_dim        struc
0000  ??                C                  EDCode          db      ?
0001  ??                C                  EDSubCode       db      ?
0002  ????              C                  EDDriveNum      dw      ?
0004  ??                C                  EDFileType      db      ?
0005  ??                C                  EDFileSubType   db      ?
0006  ??                C                  EDAccess        db      ?
0007  ??                C                  EDShr           db      ?
0008  ????????          C                  EDSize          dd      ?
000C  0006[             C                  EDFiller        db      6 dup (?)
          ??            C
                    ]   C
                        C
0012                    C  etna_dim        ends
                        C
= 0000                  C  NopCode         equ     0
= 0001                  C  MountCode       equ     1
= 0002                  C  UnmountCode     equ     2
                        C
                        C
= 000D                  C  bpb_on_disk     equ             0DH             ; Number of bytes actually on disk
                        C
```

```
 = 01C0                 C  Code_Len        equ             448d            ; from Zynar's format routine.
                        C
                        C  Boot_Record     struc
 0000  01C0[            C                  B_Code          db      Code_Len dup (?)
         ??             C
                     ]  C
                        C
 01C0  0015[            C                  B_ID            db      ID_len dup (?)
         ??             C
                     ]  C
                        C
 01D5  ??               C                  B_vpb_spc       db      ?
 01D6  ??               C                  B_vpb_csf       db      ?
 01D7  ????             C                  B_vpb_vol_ss    dw      ?
 01D9  ??               C                  B_vpb_nfats     db      ?
 01DA  ????             C                  B_vpb_nf        dw      ?
 01DC  ????             C                  B_vpb_data_ss   dw      ?
 01DE  ????             C                  B_vpb_ndc       dw      ?
 01E0  ??               C                  B_vpb_spf       db      ?
 01E1  ????             C                  B_vpb_dir_ss    dw      ?
 01E3  001D[            C                  B_filler        db      29d dup (?)
         ??             C
                     ]  C
                        C
 0200                   C  Boot_Record     ends
                        C
                        C  block0          struc
 0000  0003[            C                  BOJump          db      3 dup (?)
         ??             C
                     ]  C
                        C
 0003  0008[            C                  BOOEM           db      8 dup (?)
         ??             C
                     ]  C
                        C
 0008  ????             C                  BOBytesPerSect  dw      ?
 000D  ??               C                  BOSectPerClust  db      ?
 000E  ????             C                  BOResvSect      dw      ?
 0010  ??               C                  BOFATCount      db      ?
 0011  ????             C                  BODirEntries    dw      ?
 0013  ????             C                  BOSectCount     dw      ?
 0015  ??               C                  BOMediaDesc     db      ?
 0016  ????             C                  BOFATSize       dw      ?
                        C  ;
 0018  ????             C                  BOSectsPerTrack dw      ?
 001A  ????             C                  BOHeadCount     dw      ?
 001C  ????             C                  BOHiddenSects   dw      ?
 001E                   C  block0          ends
                        C
                        C  request_header  struc
 0000  ??               C                  RhLength        db      ?
 0001  ??               C                  RhUnit          db      ?
 0002  ??               C                  RhCommandCode   db      ?
 0003  ????             C                  RhStatus        dw      ?
```

```
0005  0008[                C                 RhDOSResv       db      8 dup (?)
          ??                C
                        ]   C
                            C
000D                        C  request_header  ends
                            C
= 0000                      C  dos_init        equ     0         ; Initilize driver
= 0001                      C  dos_media_check equ     1         ; Has media been changed?
= 0002                      C  dos_build_bpb   equ     2
= 0003                      C  dos_ioctl_in    equ     3
= 0004                      C  dos_read        equ     4
= 0005                      C  dos_read_nowait equ     5
= 0006                      C  dos_in_status   equ     6
= 0007                      C  dos_in_flush    equ     7
= 0008                      C  dos_output      equ     8
= 0009                      C  dos_out_verify  equ     9
= 000A                      C  dos_out_status  equ     10
= 000B                      C  dos_out_flush   equ     11
= 000C                      C  dos_ioctl_out   equ     12
                            C
                            C  init_rh         struc
0000  0000[                 C                  IRH             db      type request_header dup (?)
          ??                C
                        ]   C
                            C
000D  ??                    C                  IUnitCount      db      ?
000E  00 00 00 00           C                  IDriverEnd      dd      0
0012  00 00 00 00           C                  IBPBArray       dd      0
0016  00                    C                  IOurDrive       db      0
0017                        C  init_rh         ends
                            C
                            C  build_bpb_rh    struc
0000  0000[                 C                  BPBRH           db      type request_header dup (?)
          ??                C
                        ]   C
                            C
000D  ??                    C                  BPBMediaDesc    db      ?
000E  00 00 00 00           C                  BPBTempBuffer   dd      0
0012  00 00 00 00           C                  BPBBPBPointer   dd      0
0016                        C  build_bpb_rh    ends
                            C
                            C
                            C  ;   Parameter Header Offsets
                            C  ;
= 0000                      C  srh          equ  0              ;static request header start
= 000D                      C  srh_len      equ  13             ;  "      "       "     length
=                           C  srh_len_fld  equ  srh            ;  "      "       "       "    field
= 0001                      C  srh_ucd_fld  equ  srh+1          ;  "      "       "     unit code field
= 0002                      C  srh_ccd_fld  equ  srh+2          ;  "      "       "     command code field
= 0003                      C  srh_sta_fld  equ  srh+3          ;  "      "       "     status field
= 0005                      C  srh_res_fld  equ  srh+5          ;  "      "       "     reserved area field
                            C
                            C  ;
                            C  ;       Input/Output
```

```
                        C  ;
 = 000D                 C  md           equ  srh+srh_len     ;media descriptor byte
 = 0001                 C  md_len       equ  1               ; "     "          "   length
 = 000E                 C  dta          equ  md+md_len       ;disk transfer address
 = 0004                 C  dta_len      equ  4               ; dta length
 = 0012                 C  count        equ  dta+dta_len     ;byte/sector count
 = 0002                 C  count_len    equ  2               ;    "         "  length
 = 0014                 C  ssn          equ  count+count_len ; starting sector number
 = 0002                 C  ssn_len      equ  2               ;  "         "      "    length
                        C
                        C  ;
                        C  ;   media check
                        C  ;
 = 000E                 C  ret_byte     equ  md+md_len       ;byte returned from driver
=-0001                  C  MediaChanged    equ     -1        ; And what we return
 = 0000                 C  MediaDontKnow   equ     0
 = 0001                 C  MediaNotTouched equ     1
                        C
                        C  ;
                        C  ;build bpb
                        C  ;
 = 0012                 C  bpba_ptr     equ  dta+dta_len     ;pointer to bpb
 = 0004                 C  bpba_ptr_len equ  4               ;  "       "  "  length
                        C
                        C  ;
                        C  ;       init
                        C  ;
 = 000D                 C  units        equ  srh+srh_len
 = 0001                 C  units_len    equ  1
 = 000E                 C  br_addr_0    equ  units+units_len
 = 0010                 C  br_addr_1    equ  br_addr_0+2
 = 0004                 C  br_addr_len  equ  4
 = 0012                 C  bpb_ptr_off  equ  br_addr_0+br_addr_len
 = 0014                 C  bpb_ptr_seg  equ  bpb_ptr_off+2
                        C  ;
                        C
                        C  ; drive and address tables:
                        C  ;------------------------
                        C
                        C
 00DB  001A[            C  bpb_table       bpb     def_max_drive dup (<>)
           ????         C
           ??           C
           ????         C
           ??           C
           ????         C
           ????         C
           ??           C
           ????         C
           ????         C
           ????         C
           ????         C
           ??           C
           ??           C
```

```
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                ??              C
                        ]       C
                                C
                                C                               ; We start out life looking like double sided
                                C                               ;   floppies.
                                C
041B  001A[                     C  bpb_ptr_table    dw      def_max_drive dup (?)
        ????                    C
                        ]       C
                                C
                                C
                                C
044F  ????                      C  temp_bpb         bpb     <>              ; Copy of bpb for build_bpb
0451  ??                        C
0452  ????                      C
0454  ??                        C
0455  ????                      C
0457  ????                      C
0459  ??                        C
045A  ????                      C
045C  ????                      C
045E  ????                      C
0460  ????                      C
0462  ??                        C
0463  ??                        C
0464  ??                        C
0465  ??                        C
0466  ??                        C
0467  ??                        C
0468  ??                        C
0469  ??                        C
046A  ??                        C
046B  ??                        C
046C  ??                        C
046D  ??                        C
046E  ??                        C
                                C
                                C
                                C
                                C  ;vdisk tables for volumes without header (1.1 floppy size)
                                C  ;-----------------------------------------------------------
046F  0200                      C  single_8_bpb     bpb     <200H, 1, 1, 2,  64d,   40*8, 0FEH, 1, 8, 1, 0, 0 >
0471  01                        C
0472  0001                      C
```

```
0474  02                  C
0475  0040                C
0477  0140                C
0479  FE                  C
047A  0001                C
047C  0008                C
047E  0001                C
0480  0000                C
0482  00                  C
0483  ??                  C
0484  ??                  C
0485  ??                  C
0486  ??                  C
0487  ??                  C
0488  ??                  C
0489  ??                  C
048A  ??                  C
048B  ??                  C
048C  ??                  C
048D  ??                  C
048E  ??                  C
                          C
                          C  ;double_8_bpb   bpb     <200H, 2, 1, 2, 112d, 2*40*8, 0FFH, 1, 8, 2, 0, 0 >
048F  0200                C  single_9_bpb    bpb     <200H, 1, 1, 2,  64d,   40*9, 0FCH, 1, 9, 1, 0, 0 >
0491  01                  C
0492  0001                C
0494  02                  C
0495  0040                C
0497  0168                C
0499  FC                  C
049A  0001                C
049C  0009                C
049E  0001                C
04A0  0000                C
04A2  00                  C
04A3  ??                  C
04A4  ??                  C
04A5  ??                  C
04A6  ??                  C
04A7  ??                  C
04A8  ??                  C
04A9  ??                  C
04AA  ??                  C
04AB  ??                  C
04AC  ??                  C
04AD  ??                  C
04AE  ??                  C
                          C
04AF  0200                C  double_9_bpb    bpb     <200H, 2, 1, 2, 112d, 2*40*9, 0FDH, 1, 9, 2, 0, 0 >
04B1  02                  C
04B2  0001                C
04B4  02                  C
04B5  0070                C
04B7  02D0                C
```

```
04B9  FD                    C
04BA  0001                  C
04BC  0009                  C
04BE  0002                  C
04C0  0000                  C
04C2  00                    C
04C3  ??                    C
04C4  ??                    C
04C5  ??                    C
04C6  ??                    C
04C7  ??                    C
04C8  ??                    C
04C9  ??                    C
04CA  ??                    C
04CB  ??                    C
04CC  ??                    C
04CD  ??                    C
04CE  ??                    C
                            C
                            C
                            C
                            C
                            C  ;Vdisc -> BIOS error mapping table:
                            C  ;---------------------------------
                            C  ;                    BIOS value/code          <-      Vdisc code
04CF  00                    C  BIOS_error      db      00H     ;ok                      ve_ok
04D0  02                    C                  db      02H     ;device not ready        ve_no_drive
04D1  03                    C                  db      03H     ;unknown command         ve_illegal_op
04D2  0C                    C                  db      0CH     ;general failure         ve_bad_machine
04D3  08                    C                  db      08H     ;read failure            ve_no_read
04D4  00                    C                  db      00H     ;write_protect           ve_no_write
04D5  08                    C                  db      08H     ;sector_not_fnd          ve_bad_block
04D6  08                    C                  db      08H     ;sector_not_fnd          ve_no_descr_write
04D7  07                    C                  db      07H     ;unknowm media           ve_bad_disk
04D8  0C                    C                  db      0CH     ;general failure         ve_restriction
04D9  02                    C                  db      02H     ;device not ready        ve_level4
04DA  02                    C                  db      02H     ;general failure         ve_protocol
04DB  02                    C                  db      02H     ;general failure         ve_internal
04DC  02                    C                  db      02H     ;drive not ready         ve_timeout
                            C
                            C
                            C  ;
                            C  ;Request Header for Build BPB.
                            C  ;----------------------------
04DD  16                    C  rh_len          db      22
04DE  ??                    C  rh_unit_code    db      ?
04DF  02                    C  rh_command_code db      2
04E0  0000                  C  rh_status       dw      0
04E2  0008[                 C  rh_resv         db      8 dup (?)
        ??
                        ]   C
                            C
04EA  00                    C  rh_media_desc   db      0
04EB  ????????              C  rh_buffer       dd      ?
```

```
 04EF  ????????              C  rh_bpb_ptr      dd      ?
                             C

                                                %OUT    ..........VDISK_DOS_INTERFACE
                             C                  include vddos.asm                       ;DOS 2.0 interface
                             C
                             C                   subttl DOS Interface to virtual disk driver
```

```
                        C                page
                        C  ;-----------------------------------------------------------------------
                        C  ;
                        C  ;       DOS interface to virtual disk driver.  This routine interprets
                        C  ;       calls to the driver.
                        C  ;
                        C  ;-----------------------------------------------------------------------
                        C  ;
04F3                    C  vdsk    proc    far
                        C  ;
                        C  ;   function table
                        C  ;
04F3                    C  funtab  label   byte
04F3  0CDC R            C          dw   init_vdisk      ;initialization
04F5  05B3 R            C          dw   media_check     ;media check (block only)
04F7  05E0 R            C          dw   build_bpb       ;build bpb      "     "
04F9  05E5 R            C          dw   ioctl_in        ;ioctl input
04FB  06E5 R            C          dw   input           ;input (read)
04FD  05E5 R            C          dw   nd_input        ;non_destructive input no wait (char only)
04FF  05E5 R            C          dw   in_stat         ;input status                    "   "
0501  05E5 R            C          dw   in_flush        ;input flush                     "   "
0503  060F R            C          dw   output          ;output (*write)
0505  060F R            C          dw   out_verify      ;output (write) with verify
0507  05F9 R            C          dw   out_stat        ;output status                   "   "
0509  05E5 R            C          dw   out_flush       ;output flush                    "   "
050B  05E5 R            C          dw   ioctl_out       ;ioctl output
                        C  ;
                        C  ;  device strategy
                        C  ;
050D                    C  dev_strategy:
050D  2E: 8C 06 0087 R  C             mov    cs:rh_seg,es     ;save segment of request header pointer
0512  2E: 89 1E 0085 R  C             mov    cs:rh_off,bx     ;save offset of    "       "       "
0517  CB                C             ret
                        C
                        C  ;
                        C  ;  device interrupt handler
                        C  ;
0518                    C  dev_int:
                        C  ; preserve machine state on entry
0518  FC                C          cld
0519  1E                C          push   ds
051A  06                C          push   es
051B  50                C          push   ax
051C  53                C          push   bx
051D  51                C          push   cx
051E  52                C          push   dx
051F  57                C          push   di
0520  56                C          push   si
0521  55                C          push   bp
                        C  ;
                        C  ;       Now set DS to be the code segment
                        C  ;
0522  8C C8             C          mov    ax,cs
0524  8E D8             C          mov    ds,ax
```

```
                        C          assume  ds:cseg
                        C  ;
                        C  ;       Set up local stack
                        C  ;
0526  FE 06 0000 E      C          inc     stack_use               ;we know we are first user
052A  89 26 0000 E      C          mov     sp_save,sp
052E  8C 16 0000 E      C          mov     ss_save,ss
0532  FA                C          cli
0533  90                C          nop
0534  BC 0000 E         C          mov     sp,offset stack_top
0537  8E D0              C          mov     ss,ax                           ;still = cs
0539  FE 06 0083 R      C          inc     driver_busy
053D  80 3E 0083 R 01   C          cmp     driver_busy,1           ; are other driver active
0542  74 10             C          je      notbusy
0544  FB                C          sti                             ; enable interrupts
                        C          status  notready,noerror,0      ; send notready error
0551  EB 35 90           C          jmp     exit                    ; return to DOS
                        C
                        C  ;
                        C  ;       do iob switch processing
                        C  ;
0554                    C  notbusy:
0554  FB                C                  sti                                 ;enable interrupts
0555  80 3E 0089 R 00   C                  cmp     iob_switch,0                ;in special iob mode ?
055A  74 1C             C                  je      do_function                 ;skip this if not
                        C  ;I don't think this will work if the protected program is loading across midnight
055C  E8 0C77 R         C                  call    get_tod                     ;get current time
055F  2B 16 008C R      C                  sub     dx,end_special_lo           ;compare against expire time
0563  1B 0E 008E R      C                  sbb     cx,end_special_hi
0567  73 0A             C                  jnc     cancel_special              ;cancel if tod >= end_special
0569  26: 8A 47 01      C                  mov     al,es:srh_ucd_fld[bx]       ;check for correct special drive
056D  3A 06 0093 R      C                  cmp     al,special_drive            ;stay in special iob state if
0571  74 05             C                  je      do_function                 ;  correct drive
0573                    C  cancel_special:
0573  C6 06 0089 R 00   C                  mov     iob_switch,0                ;clear the iob switch
0578                    C  do_function:
                        C  ;
                        C  ;       Do the branch according to the function passed
                        C  ;
0578  26: 8A 47 02      C             mov    al,es:srh_ccd_fld[bx]  ;get function byte
057C  D0 C0             C             rol    al,1                   ;get offset into table
057E  8D 3E 04F3 R      C             lea    di,funtab              ;get address of function table
0582  32 E4             C             xor    ah,ah
                        C          write   'Dos function: '
                        C          writeint ax
                        C          writeln ' '
0584  03 F8             C             add    di,ax
                        C  ;       call    dos_debug
0586  FF 25             C             jmp    word ptr[di]
```

```
                        C           page
                        C   ;*****************************************************************
                        C   ;
                        C   ;       common exit
                        C   ;
 0588                   C   exit:
                        C           writeln 'At exit...'
 0588  E8 OCBA R        C           call    set_end_special         ;remember time of last call to us
                        C           writeln 'after set_end_special'
 058B  2E: 8E 06 0087 R C           mov     es, cs:rh_seg
 0590  2E: 8B 1E 0085 R C           mov     bx, cs:rh_off
                        C           write   'Returning '
                        C           writeint        es:[bx].RhStatus
                        C           writeln ' '
 0595  FA               C           cli
 0596  2E: 8E 16 0000 E C           mov     ss, cs:ss_save          ; Restore stack pointer
 059B  2E: 8B 26 0000 E C           mov     sp, cs:sp_save
 05A0  FE 0E 0000 E     C           dec     stack_use               ; we know we are outer user
 05A4  FE 0E 0083 R     C           dec     driver_busy
 05A8  FB               C           sti
 05A9  5D               C           pop     bp
 05AA  5E               C           pop     si              ;restore all of the registers
 05AB  5F               C           pop     di
 05AC  5A               C           pop     dx
 05AD  59               C           pop     cx
 05AE  5B               C           pop     bx
 05AF  58               C           pop     ax
 05B0  07               C           pop     es
 05B1  1F               C           pop     ds
 05B2  CB               C           ret
                        C   vdsk    endp
```

```
                        C           page
 05B3                   C  vsub     proc    near
                        C  ;***************************************************************
                        C  ;
                        C  ; media check
                        C  ;
 05B3                   C  media_check:                         ; media check (block only)
 05B3  E8 0C66 R        C          call    get_drive
 05B6  B4 00            C          mov     ah,0                 ; Convert to 16 bits
 05B8  2E: F7 26 00AB R C          mul     cs:type_bpb          ; Offset into bpb table
                        C  ;
 05BD  8B F0            C          mov     si,ax                ; To an index register
 05BF  2E: 8A 84 00EE R C          mov     al,cs:bpb_table[si].BMediaChanged
                        C  ;
 05C4  2E: 80 BC 00F7 R 20 C       cmp     cs:bpb_table[si].BFileType, NotMountedType
 05CA  75 02            C          jne     mc1
 05CC  B0 01            C          mov     al, MediaNotTouched
                        C  ;
 05CE                   C  mc1:
 05CE  26: 88 47 0E     C          mov     es:ret_byte[bx],al   ; Return to caller
                        C  ;
                        C          status  done,noerror,0       ;turn on the done bit
 05DE  EB A8            C          jmp     exit
```

```
                        C           page
                        C   ;*****************************************************************
                        C   ;
                        C   ; build bios parameter block
                        C   ;
05E0                    C   build_bpb:
05E0  E8 07F3 R         C           call    real_build_bpb          ; Do the real work
05E3  EB A3             C           jmp     exit
                        C
                        C
                        C   ;*****************************************************************
                        C   ;
                        C   ;the following entries are for not supported by this device
                        C   ;
05E5                    C   ioctl_in:
05E5                    C   ioctl_out:
05E5                    C   nd_input:                           ;non_destructive input no wait (char only)
05E5                    C   in_stat:                            ;input status                   "    "
05E5                    C   in_flush:                           ;input flush
05E5                    C   out_flush:                          ;output flush
                        C           status  done,error,03H
05F7  EB 8F             C           jmp     exit
```

```
                        C           page
                        C   ;*************************************************************
                        C   ;
                        C   ; Output Status
                        C   ;
05F9                    C   out_stat:                               ;output status
05F9  80 3E 0084 R 00   C           cmp     rb_busy,0               ; RB Connected?
05FE  74 0C             C           je      vd_notbusy              ; rb is not busy
                        C           status  busy,noerror,0          ;turn on the busy bit
060C                    C   vd_notbusy:
060C  E9 0588 R         C           jmp     exit
```

```
                        C           page
                        C   ;******************************************************************************
                        C   ;
                        C   ;  disk write
                        C   ;
060F                    C   out_verify:
060F                    C   output:                             ;output (write)
060F  E8 0C66 R         C           call    get_drive
0612  F6 26 00AB R       C           mul     byte ptr type_bpb       ; Convert to offset in bpb table
0616  8B F0              C           mov     si, ax
                        C   ;
0618  B4 05              C           mov     ah, ve_no_write         ; Set default error code
061A  2E: 88 A4 00F5 R   C           mov     cs:bpb_table[si].BPriErr,ah; store in table
061F  2E: 88 A4 00F6 R   C           mov     cs:bpb_table[si].BSecErr,ah
0624  BA 0000            C           mov     dx, 0                   ; Clear length in case no write
                        C   ;
0627  B4 00              C           mov     ah, ve_ok               ; If primary is not being written too,
0629  2E: 80 BC 00F2 R 57 C           cmp     cs:bpb_table[si].BPriWrite, WriteState
062F  75 26              C           jne     o5
                        C   ;
0631  2E: 8A 84 00EF R   C           mov     al, cs:bpb_table[si].BPriFS
0636  B4 03              C           mov     ah, ve_bad_machine
0638  0A C0              C           or      al,al                   ; If 0, then this hasn't been set
063A  74 1B              C           jz      o5
063C  A2 00AA R          C           mov     io_fs,al                        ; Set file server for i/o
                        C   ;
063F  56                 C           push    si
0640  E8 06BF R          C           call    outl
0643  5E                 C           pop     si
                        C   ;
0644  80 FC 01           C           cmp     ah, ve_no_drive
0647  75 0E              C           jne     o5
0649  2E: 80 BC 00F0 R 00 C           cmp     cs:bpb_table[si].BSecFS, 0
064F  75 06              C           jne     o5
0651  2E: C6 84 00F7 R 20 C           mov     cs:bpb_table[si].BFileType, NotMountedType
                        C   ;
0657                    C   o5:
0657  2E: 88 A4 00F5 R   C           mov     cs:bpb_table[si].BPriErr, ah
065C  2E: C6 84 00F6 R 00 C           mov     cs:bpb_table[si].BSecErr, ve_ok
                        C   ;
0662  2E: 80 BC 00F4 R 57 C           cmp     cs:bpb_table[si].BSecWrite, WriteState
0668  75 18              C           jne     o2
066A  2E: 8A 84 00F0 R   C           mov     al, cs:bpb_table[si].BSecFS
066F  B4 03              C           mov     ah, ve_bad_machine
0671  0A C0              C           or      al,al                   ; If 0, then this hasn't been set
0673  74 08              C           jz      o6
0675  A2 00AA R          C           mov     io_fs, al
                        C   ;
0678  56                 C           push    si
0679  E8 06BF R          C           call    outl
067C  5E                 C           pop     si
067D                    C   o6:
067D  2E: 88 A4 00F6 R   C           mov     cs:bpb_table[si].BSecErr, ah
                        C   ;
```

```
0682                    C  o2:
0682  2E: 8A 84 00F5 R   C          mov     al, cs:bpb_table[si].BPriErr
0687  3C 00              C          cmp     al, ve_ok               ; Is primary ok
0689  75 06              C          jne     o3                      ; No, report error
                        C  ;
068B  86 E0              C          xchg    ah, al
068D  3C 00              C          cmp     al, ve_ok               ; Is secondary ok
068F  74 1F              C          je      o4                      ; Yes, no error so return
                        C  ;
0691                    C  o3:
0691  26: 29 57 12       C          sub     es: word ptr [bx].count, dx
0695  B4 00              C          mov     ah, 0                   ; Update count and set error to int
0697  8B F0              C          mov     si, ax
0699  8A 84 04CF R       C          mov     al, BIOS_error[si]      ; translate from IOB error
                        C          status  done, error, ax
06AD  E9 0588 R          C          jmp     exit
                        C  ;
06B0                    C  o4:
                        C          status  done, noerror, 0
06BC  E9 0588 R          C          jmp     exit
                        C
                        C  ;
                        C  ;       outl.  Detirmine iob command and handle write with verify
                        C  ;
06BF                    C  outl:
06BF  C6 06 0089 R 00    C          mov     iob_switch,0            ;clear the iob switch
06C4  B4 02              C          mov     ah,IOB_write
06C6  E8 079A R          C          call    doio                    ; Do the write]
                        C  ;
06C9  80 FC 00           C          cmp     ah, ve_ok               ; Was write ok
06CC  75 16              C          jne     out2                    ; If not, don't bother to verify
                        C  ;
06CE  26: 80 7F 02 09    C          cmp     es: byte ptr srh_ccd_fld[bx], dos_out_verify
06D3  75 0F              C          jne     out2
                        C  ;
06D5  26: C6 47 02 04    C          mov     es: byte ptr srh_ccd_fld[bx], dos_read
06DA  B4 01              C          mov     ah, IOB_read
06DC  E8 079A R          C          call    doio
06DF  26: C6 47 02 09    C          mov     es: byte ptr srh_ccd_fld[bx], dos_out_verify
                        C  ;
06E4                    C  out2:
06E4  C3                 C          ret
```

```
                        C           page
                        C   ;*****************************************************************
                        C   ;*
                        C   ;*      Disk Read.  Read from whichever volume has the read flag set.
                        C   ;*      Primary takes precedence over secondary.
                        C   ;*
 06E5                   C   input:
 06E5  E8 0C66 R        C           call    get_drive
 06E8  2E: F6 26 00AB R C           mul     byte ptr cs:type_bpb            ; Offset in bpb table
 06ED  8B F0            C           mov     si,ax
                        C   ;
 06EF  2E: 80 BC 00F1 R 52  C       cmp     cs:bpb_Table[si].BPriRead,ReadState     ; Read from primary?
 06F5  75 2E            C           jne     i1
                        C   ;
 06F7  2E: 8A 84 00EF R C           mov     al, cs:bpb_table[si].BPriFS     ; Get primary file server
 06FC  B4 03            C           mov     ah, ve_bad_machine              ; If we're not talking
 06FE  0A C0            C           or      al,al                           ; Are we talking?
 0700  74 1B            C           je      i5
                        C   ;
 0702  A2 00AA R        C           mov     io_fs, al                       ; Set up global stn adr
 0705  56               C           push    si
 0706  E8 078C R        C           call    in1                             ; Do it.
 0709  5E               C           pop     si
                        C   ;
 070A  80 FC 01         C           cmp     ah, ve_no_drive
 070D  75 0E            C           jne     i5
 070F  2E: 80 BC 00F0 R 00  C       cmp     cs:bpb_table[si].BSecFS, 0
 0715  75 06            C           jne     i5
 0717  2E: C6 84 00F7 R 20  C       mov     cs:bpb_table[si].BFileType, NotMountedType
                        C   ;
 071D                   C   i5:
 071D  2E: 88 A4 00F5 R C           mov     cs:bpb_table[si].BPriErr, ah    ; Save error code
 0722  EB 33 90         C           jmp     i2                              ; Off we go.
                        C   ;
 0725                   C   i1:
 0725  2E: 80 BC 00F3 R 52  C       cmp     cs:bpb_Table[si].BSecRead,ReadState     ; Read here?
 072B  75 1B            C           jne     i3                              ; Nope, no reads
                        C   ;
 072D  2E: 8A 84 00F0 R C           mov     al, cs:bpb_table[si].BSecFS     ; Get secondary file server
 0732  B4 03            C           mov     ah, ve_bad_machine              ; If we're not talking
 0734  0A C0            C           or      al,al                           ; Are we talking?
 0736  74 08            C           je      i6
                        C   ;
 0738  A2 00AA R        C           mov     io_fs, al                       ; Set up global stn adr
 073B  56               C           push    si
 073C  E8 078C R        C           call    in1                             ; Do it.
 073F  5E               C           pop     si
                        C   ;
 0740                   C   i6:
 0740  2E: 88 A4 00F6 R C           mov     cs:bpb_table[si].BSecErr, ah    ; Save error code
 0745  EB 10 90         C           jmp     i2                              ; Off we go.
                        C   ;
 0748  B4 04            C   i3:     mov     ah, ve_no_read                  ; Come here if no reads allowed
 074A  2E: 88 A4 00F5 R C           mov     cs:bpb_table[si].BPriErr, ah    ; No error on read
```

```
074F  2E: 88 A4 00F6 R      C          mov     cs:bpb_table[si].BSecErr, ah
0754  BA 0000               C          mov     dx, 0                           ; 0 bytes read
0757                        C  i2:
0757  80 FC 00              C          cmp     ah, ve_ok                       ; is everything OK?
075A  74 21                 C          je      i4                              ; Yes
                            C  ;
075C  26: 29 57 12          C          sub     es: word ptr [bx].count, dx     ; Adjust count to show work done
0760  B0 00                 C          mov     al, 0
0762  86 E0                 C          xchg    ah,al                           ; Convert error code to word
0764  8B F0                 C          mov     si, ax
0766  8A 84 04CF R          C          mov     al, BIOS_error[si]      ; translate from IOB error
                            C          status  done, error, ax
077A  E9 0588 R             C          jmp     exit
                            C  ;
077D                        C  i4:
                            C          status  done, noerror, 0
0789  E9 0588 R             C          jmp     exit
                            C
                            C  ;
                            C  ;  Figure out proper iob command and pass it on to doio
                            C  ;
078C                        C  in1:
078C  B4 01                 C          mov     ah,IOB_read                     ;assume normal read
078E  80 3E 0089 R 00       C          cmp     iob_switch,0                    ;special ?
0793  74 02                 C          je      normal_input                    ;jump if not special
0795  B4 03                 C          mov     ah, IOB_special
0797                        C  normal_input:
0797  EB 01 90              C          jmp     doio
```

```
                        C           page
                        C   ;*****************************************************************************
                        C   ;
                        C   ;       doio.  Doio is the command code for input and output.  It windows
                        C   ;       the request to less then 128 blocks (due to sign problems in
                        C   ;       virt_disk_io).  It expects the IOB command in ah and the file
                        C   ;       server station to be stored in the variable io_fs.  It returns
                        C   ;       the error code in ah and the blocks actually done in dx
                        C   ;
 079A                   C   doio:
 079A  E8 0C66 R        C           call    get_drive
                        C   ;
 079D  26: 8B 57 12     C           mov     dx,es:count[bx]         ; Number of sectors
 07A1  83 FA 00         C           cmp     dx,0                    ; Anything to do?
 07A4  74 42            C           je      nothing_to_do
 07A6  26: 8B 4F 14     C           mov     cx,es:ssn[bx]           ; Start sector number
 07AA  26: C4 5F 0E     C           les     bx,es:[bx].BPBTempBuffer ; Pointer to data in es:bx
                        C   ;
                        C   ;       See if we need to window
                        C   ;
 07AE                   C   window:
 07AE  83 FA 20         C           cmp     dx,io_limit
 07B1  76 2C            C           jbe     in_limit
                        C   ;
 07B3  50               C           push    ax
 07B4  53               C           push    bx
 07B5  51               C           push    cx
 07B6  52               C           push    dx
 07B7  06               C           push    es
 07B8  E8 0C61 R        C           call    get_file_server         ; Returned in dh
 07BB  B2 20            C           mov     dl,io_limit
                        C           lcall   vdisk_io
                        C           bnz     window_error
 07C6  07               C           pop     es
 07C7  5A               C           pop     dx
 07C8  59               C           pop     cx
 07C9  5B               C           pop     bx
 07CA  58               C           pop     ax
 07CB  83 EA 20         C           sub     dx,io_limit
 07CE  83 C1 20         C           add     cx,io_limit             ; Update code
 07D1  81 C3 4000       C           add     bx,io_limit*200H        ; and hope we don't wrap around
 07D5  EB D7            C           jmp     window
                        C   ;
 07D7                   C   window_error:
 07D7  07               C           pop     es
 07D8  5A               C           pop     dx
 07D9  59               C           pop     cx
 07DA  5B               C           pop     bx
 07DB  5B               C           pop     bx                      ; Dump old ax
 07DC  EB 0A 90         C           jmp     io_return
                        C   ;
                        C   ;       Come here if less then our limit
                        C   ;
 07DF                   C   in_limit:
```

```
07DF  52                  C          push    dx
07E0  E8 0C61 R           C          call    get_file_server
                          C          lcall   vdisk_io                ; Off we go...
07E7  5A                  C          pop     dx                      ; Restore size of read
                          C  ;
07E8                      C  nothing_to_do:
07E8                      C  io_return:
07E8  2E: 8B 1E 0085 R    C          mov     bx,cs:rh_off
07ED  2E: 8E 06 0087 R    C          mov     es,cs:rh_seg            ; Get back request header
07F2  C3                  C          ret
                          C
                          C  vsub    endp
```

```
                        C           page
                        C   ;*****************************************************************
                        C   ;*
                        C   ;*      Real_build_bpb
                        C   ;*
                        C   ;*      Do the actual work of building a bpb.  ES:BX is the pointer
                        C   ;*      to a dos style request header.
                        C   ;*
 07F3                   C   real_build_bpb  proc    near
 07F3  E8 0C66 R        C           call    get_drive
                        C           write   'drive '
                        C           writebyte       al
                        C           writeln ' '
 07F6  2E: F6 26 00AB R C           mul     byte ptr cs: type_bpb           ; Convert to index in bpb table
 07FB  8B F0            C           mov     si,ax
                        C   ;
 07FD  2E: C6 84 00F5 R 04  C       mov     cs:bpb_table[si].BPriErr, ve_no_read
 0803  2E: C6 84 00F6 R 04  C       mov     cs:bpb_table[si].BSecErr, ve_no_read
                        C   ;
 0809  2E: 80 BC 00F1 R 2D  C       cmp     cs:bpb_table[si].BPriRead, NotUsedState
 080F  75 08            C           jne     rbb1                            ; Primary in use
 0811  2E: 80 BC 00F2 R 2D  C       cmp     cs:bpb_table[si].BPriWrite, NotUsedState
 0817  74 31            C           je      rbb2
                        C   ;
 0819                   C   rbb1:
 0819  2E: 8A 84 00EF R C           mov     al, cs:bpb_table[si].BPriFS     ; Get file server
 081E  B4 03            C           mov     ah, ve_bad_machine
 0820  0A C0            C           or      al,al
 0822  74 1B            C           jz      rbb8
 0824  A2 00AA R        C           mov     io_fs, al                       ;    and set it.
 0827  56               C           push    si
 0828  E8 08FD R        C           call    bpb1
 082B  5E               C           pop     si
                        C           write   'Returned '
                        C           writebyte       ah
                        C           writeln ' '
 082C  80 FC 01         C           cmp     ah, ve_no_drive
 082F  75 0E            C           jne     rbb8
 0831  2E: 80 BC 00F0 R 00  C       cmp     cs:bpb_table[si].BSecFS, 0
 0837  75 06            C           jne     rbb8
 0839  2E: C6 84 00F7 R 20  C       mov     cs:bpb_table[si].BFileType, NotMountedType
                        C   ;
 083F                   C   rbb8:
 083F  2E: 88 A4 00F5 R C           mov     cs:bpb_table[si].BPriErr, ah
 0844  2E: C6 84 00F6 R 00  C       mov     cs:bpb_table[si].BSecErr, ve_ok
                        C   ;
 084A                   C   rbb2:
 084A  2E: 80 BC 00F3 R 2D  C       cmp     cs:bpb_table[si].BSecRead, NotUsedState
 0850  75 08            C           jne     rbb3                            ; Primary in use
 0852  2E: 80 BC 00F4 R 2D  C       cmp     cs:bpb_table[si].BSecWrite, NotUsedState
 0858  74 5F            C           je      rbb4
                        C   ;
 085A                   C   rbb3:
 085A  2E: 80 BC 00F5 R 00  C       cmp     cs:bpb_table[si].BPriErr, ve_ok
```

```
0860  75 10                C          jne     rbb5                            ; Don't copy, its not there
                            C  ;
0862  06                    C          push    es
0863  56                    C          push    si
0864  51                    C          push    cx                              ; Save regs used
                            C  ;
0865  1E                    C          push    ds
0866  07                    C          pop     es
0867  BF 044F R            C          mov     di, offset temp_bpb             ; Stash primary bpb here
086A  B9 0020               C          mov     cx, type bpb                    ; this much
086D  F3/ A4                C          rep     movsb                           ; Shovel as fast as you can
086F  59                    C          pop     cx
0870  5E                    C          pop     si
0871  07                    C          pop     es
                            C  ;
0872                        C  rbb5:
0872  2E: 8A 84 00F0 R      C          mov     al, cs:bpb_table[si].BSecFS     ; Get file server
0877  B4 03                 C          mov     ah, ve_bad_machine
0879  0A C0                 C          or      al,al
087B  74 08                 C          jz      rbb9
087D  A2 00AA R             C          mov     io_fs, al                       ;    and set it.
0880  56                    C          push    si
0881  E8 08FD R             C          call    bpb1
0884  5E                    C          pop     si
                            C  ;
0885                        C  rbb9:
0885  2E: 88 A4 00F6 R      C          mov     cs:bpb_table[si].BSecErr, ah
                            C  ;
088A  80 FC 00              C          cmp     ah, ve_ok                       ; Did secondary finish?
088D  75 2A                 C          jne     rbb4                            ; No, don't compare
088F  2E: 80 BC 00F5 R 00   C          cmp     cs:bpb_table[si].BPriErr, ve_ok ; Did primary finish?
0895  75 22                 C          jne     rbb4                            ; No
                            C  ;
0897  06                    C          push    es
0898  56                    C          push    si
0899  51                    C          push    cx                              ; save some regs
                            C
089A  1E                    C          push    ds
089B  07                    C          pop     es
089C  BF 044F R             C          mov     di, offset temp_bpb
089F  B9 000D               C          mov     cx, bpb_on_disk                 ; Compare only important stuff
08A2  F2/ A6                C          repne   cmpsb
08A4  59                    C          pop     cx
08A5  5E                    C          pop     si
08A6  07                    C          pop     es
08A7  74 10                 C          je      rbb4                            ; Everything same
                            C  ;
08A9  56                    C          push    si
08AA  51                    C          push    cx
08AB  BF 044F R             C          mov     di, offset temp_bpb             ; Restore primary
08AE  B9 000D               C          mov     cx, bpb_on_disk
08B1  F3/ A4                C          rep     movsb
                            C  ;
08B3  B8 0008               C          mov     ax, 8                           ; Media unknown
```

```
08B6  EB 34 90                C           jmp     rbb6                            ; and off we go
                              C   ;
08B9                          C   rbb4:
08B9  2E: 8A 84 00F5 R        C           mov     al, cs:bpb_table[si].BPriErr    ; Get primary error code
08BE  3C 00                   C           cmp     al, ve_ok
08C0  75 1C                   C           jne     rbb7                            ; Not ok, report it
08C2  2E: 8A 84 00F6 R        C           mov     al, cs:bpb_table[si].BSecErr    ; Get secondary error code
08C7  3C 00                   C           cmp     al, ve_ok
08C9  75 13                   C           jne     rbb7                            ; No ok
                              C   ;
08CB  2E: C6 84 00EE R 01     C           mov     cs:bpb_table[si].BMediaChanged, MediaNotTouched
                              C   ;
                              C           status  done, noerror, 0
08DD  C3                      C           ret
                              C   ;
08DE                          C   rbb7:
08DE  2E: C6 84 00EE R FF     C           mov     cs:bpb_table[si].BMediaChanged, MediaChanged
08E4  B4 00                   C           mov     ah, 0                           ; Convert al to word
08E6  8B F0                   C           mov     si, ax
08E8  8A 84 04CF R            C           mov     al, BIOS_error[si]              ; Convert error code
08EC                          C   rbb6:
                              C           status  done, error, ax
08FC  C3                      C           ret
                              C   real_build_bpb  endp
```

```
                        C           page
                        C   ;*****************************************************************
                        C   ;*
                        C   ;*      bpbl.  Build a bpb from one of the file servers.  Expects
                        C   ;*      io_fs to contain the station address of the file server.
                        C   ;*      Note that on errors, the bpb table is assumed to be left
                        C   ;*      unchanged.
                        C   ;*
 08FD                   C   bpbl            proc    near
 08FD  B4 01            C           mov     ah,IOB_read                     ;assume normal read
 08FF  80 3E 0089 R 00  C           cmp     iob_switch,0                    ;special ?
 0904  74 02            C           je      normal_bpb                      ;jump if not special
 0906  B4 03            C           mov     ah, IOB_special
 0908                   C   normal_bpb:
 0908  E8 0C66 R        C           call    get_drive
 090B  2E: A2 00A9 R    C           mov     cs:save_drive, al       ; Save driver number
 090F  E8 0C61 R        C           call    Get_file_server         ; Get servers station in dh
                        C   ;
 0912  B2 01            C           mov     dl,1                    ; of 1 sector
 0914  B9 0000          C           mov     cx,0                    ; from sector 0
 0917  26: C4 5F 0E     C           les     bx,es:[bx].BPBTempBuffer ; into DOS's buffer
                        C           lcall   vdisk_io
                        C           bnz     bad_bpb
                        C   ;
 0924  83 F9 01         C           cmp     cx,1                    ; See if we got the right #
 0927  B4 08            C           mov     ah, ve_bad_disk         ; Read error
                        C           bnz     bad_bpb                 ; Oops, should never happen
                        C   ;
 092E  B4 08            C           mov     ah, ve_bad_disk         ; Unknown Media
                        C   ;
 0930  BE 00B1 R        C           mov     si,offset OEM_name      ; check primary OEM ('IBM  2.0')
 0933  8B FB            C           mov     di,bx
 0935  83 C7 03         C           add     di,3                    ; es:di -> BPB.OEM_name
 0938  B9 0004          C           mov     cx,4                    ; for 4 words
 093B  FC               C           cld
 093C  F3/ A7           C           repz    cmpsw
 093E  74 0F            C           jz      is_20                   ; compared ok
                        C   ;
 0940  BE 00D3 R        C           mov     si,offset NIBM_OEM      ; check alternate OEM
 0943  8B FB            C           mov     di,bx
 0945  83 C7 03         C           add     di,3
 0948  B9 0004          C           mov     cx,4
 094B  F3/ A7           C           repz    cmpsw
 094D  75 57            C           jnz     not_20                  ; not matched either, check for 1.1
                        C   ;
 094F                   C   is_20:
 094F  2E: A0 00A9 R    C           mov     al, cs:save_drive       ; Get back drive number
 0953  B4 00            C           mov     ah,0                    ; to unsigned word
 0955  2E: F7 26 00AB R C           mul     cs:type_bpb             ; Offset in bpb_table
 095A  8B F8            C           mov     di,ax                   ;    in an index reg
 095C  50               C           push    ax
 095D  BE 0000          C           mov     si,0                    ; For copy
 0960  B9 000D          C           mov     cx, bpb_on_disk ; Copy this many bytes
 0963                   C   copy_bpb:
```

```
0963  26: 8A 40 0B          C          mov     al, es:byte ptr [bx+si].BOBytesPerSect
0967  2E: 88 85 00DB R       C          mov     cs:byte ptr bpb_table[di],al
096C  46                    C          inc     si
096D  47                    C          inc     di
096E  E2 F3                 C          loop    copy_bpb
                            C  ;
0970  5E                    C          pop     si
0971  A0 00A8 R             C          mov     al,Media_desc
0974  2E: 88 84 00E5 R      C          mov     cs:bpb_table[si].BMediaDesc,al
0979  FE C0                 C          inc     al
097B  24 7F                 C          and     al,7fh                          ;keep media desc < 128
097D  A2 00A8 R             C          mov     Media_desc,al
0980  56                    C          push    si
0981                        C  made_bpb:
0981  2E: 8B 1E 0085 R      C          mov     bx, cs:rh_off           ; Get back request header pointer
0986  2E: 8E 06 0087 R      C          mov     es, cs:rh_seg
                            C  ;
098B  5E                    C          pop     si                      ; Offset into bpb table
098C  8D 84 00DB R          C          lea     si, cs:bpb_table[si]    ; Address of entry
0990  26: 89 77 12          C          mov     es:bpba_ptr[bx],si      ; Stash offset in request header
0994  26: 8C 4F 14          C          mov     es:bpba_ptr+2[bx],cs    ; Stash segment in request header
                            C  ;
0998  B4 00                 C          mov     ah, ve_ok
099A  C3                    C          ret
                            C  ;
                            C  ;       Come here if bpb is not 2.0 or 1.0 or if we had some error.
                            C  ;       Error code is in ah.
                            C  ;
099B                        C  bad_bpb:
099B  2E: 8B 1E 0085 R      C          mov     bx, cs:rh_off           ; Get back request header pointer
09A0  2E: 8E 06 0087 R      C          mov     es, cs:rh_seg
09A5  C3                    C          ret
                            C  ;
                            C  ;       Check for 1.1 virtual parameter block.  If so, convert to 2.0
                            C  ;
09A6                        C  not_20:
09A6  B9 0015               C          mov     cx,ID_Len               ; Check id
09A9  BE 0000               C          mov     si,0
09AC                        C  IDLoop:
09AC  8A 84 00B9 R          C          mov     al, ID[si]
09B0  26: 3A 80 01C0        C          cmp     al,es:byte ptr[bx+si].B_ID
09B5  74 03                 C          jz      ID2
09B7  E9 0A46 R             C          jmp     check_FAT                ; Not right, try the FAT
09BA                        C  ID2:
09BA  46                    C          inc     si
09BB  E2 EF                 C          loop    IDLoop
                            C  ;
09BD  2E: A0 00A9 R         C          mov     al, cs:save_drive
09C1  B4 00                 C          mov     ah,0                    ; To unsigned word
09C3  2E: F7 26 00AB R      C          mul     cs:type_bpb
09C8  8B F8                 C          mov     di,ax
09CA  50                    C          push    ax
                            C  ;
                            C  ;       It is 1.1, convert to 2.0 format.
```

```
                                C  ;
09CB  2E: C7 85 00DB R 0200 C           mov     cs:bpb_table[di].BBytesPerSect,200h     ; Always 512
09D2  2E: C7 85 00DE R 0001 C           mov     cs:bpb_table[di].BResvSect,1            ; Always 1
                                C  ;
09D9  26: 8A 87 01D5        C           mov     al,es:[bx].B_vpb_spc                    ; Copy sectors per cluster
09DE  FE C0                 C           inc     al                                      ; Zynar's zformat dec's it
09E0  2E: 88 85 00DD R      C           mov     cs:bpb_table[di].BSectPerClust,al
                                C  ;
09E5  26: 8A 87 01D9        C           mov     al,es:[bx].b_vpb_nfats                  ; # of fats
09EA  2E: 88 85 00E0 R      C           mov     cs:bpb_table[di].BFATCount,al
                                C  ;
09EF  26: 8B 87 01DA        C           mov     ax,es:[bx].b_vpb_nf                     ; # of dir entries
09F4  2E: 89 85 00E1 R      C           mov     cs:bpb_table[di].BDirEntries,ax
                                C  ;
                                C  ;The following computation gives an incorrect value for total number of
                                C  ; sectors.  Removed 8/9/83 by Peter L. Stahl, who wants his initials
                                C  ; immortalized in this driver somewhere or other.
                                C  ;       mov     al,es:[bx].b_vpb_spc                    We want # of sectors
                                C  ;       inc     al                                      which is
                                C  ;       mov     ah,0                                    spc * ndc
                                C  ;       mul     es:[bx].b_vpb_ndc
                                C  ;       cmp     dx,0                                    If too many
                                C  ;       je      not_bad_bpb                               error
                                C  ;       mov     ah,07                                      unknwon media
                                C  ;       jmp     bad_bpb
                                C  ;
                                C  ;Here's how it ought to be done...
                                C  ;
                                C  ;Total sectors on volume = ((NDC - 1) * SPC) + DATA_SS
09F9  51                    C           push    cx                                      ; we'll use this
09FA  26: 8B 8F 01DE        C           mov     cx,es:[bx].b_vpb_ndc                    ; # of data clusters
09FF  49                    C           dec     cx                                      ; as per formula above
0A00  26: 8A 87 01D5        C           mov     al,es:[bx].b_vpb_spc                    ; # of sectors per cl.
0A05  FE C0                 C           inc     al                                      ; becuz Zynar dec's it
0A07  B4 00                 C           mov     ah,0                                    ; to get a word
0A09  F7 E1                 C           mul     cx
0A0B  59                    C           pop     cx
0A0C  83 FA 00              C           cmp     dx,0                                    ; If too many sectors,
0A0F  74 04                 C           je      ok_bpb_so_far                           ;   error:
0A11  B4 08                 C           mov     ah,ve_bad_disk                          ;     unknown media
0A13  EB 86                 C           jmp     bad_bpb
                                C  ;
0A15                            C  ok_bpb_so_far:
0A15  26: 03 87 01DC        C           add     ax,es:[bx].b_vpb_data_ss                ; as per formula above
0A1A  73 05                 C           jnc     not_bad_bpb                             ; If too many sectors,
0A1C  B4 08                 C           mov     ah,ve_bad_disk                                ;   error:
0A1E  E9 099B R             C           jmp     bad_bpb                                 ;     unknown media
                                C  ;
0A21                            C  not_bad_bpb:
0A21  50                    C           push    ax
0A22  A0 00A8 R             C           mov     al,Media_desc
0A25  2E: 88 85 00E5 R      C           mov     cs:bpb_table[di].BMediaDesc,al
0A2A  FE C0                 C           inc     al
0A2C  24 7F                 C           and     al,7fh                                  ; keep media desc < 128
```

```
0A2E  A2 00A8 R           C          mov     Media_desc,al
0A31  58                  C          pop     ax
0A32  2E: 89 85 00E3 R    C          mov     cs:bpb_table[di].BSectCount,ax
                          C  ;
0A37  26: 8A 87 01E0      C          mov     al,es:[bx].b_vpb_spf                    ; Sectors per fat
0A3C  B4 00               C          mov     ah,0
0A3E  2E: 89 85 00E6 R    C          mov     cs:bpb_table[di].BFATSize,ax
                          C  ;
0A43  E9 0981 R           C          jmp     made_bpb
                          C  ;
                          C  ;       None of our id's matched.  Check the media descriptor byte in
                          C  ;       the FAT to see if we know the type.  We assume 5 1/4" disks
                          C  ;       when we define the bpb's, 8" and fixed disk should be caught
                          C  ;       above.
                          C  ;
0A46                      C  Check_FAT:
0A46  B4 01               C          mov     ah,IOB_read                     ;assume normal read
0A48  80 3E 0089 R 00     C          cmp     iob_switch,0                    ;special ?
0A4D  74 02               C          je      normal_fat                      ;jump if not special
0A4F  B4 03               C          mov     ah, IOB_special
0A51                      C  normal_fat:
0A51  2E: A0 00A9 R       C          mov     al,cs:save_drive
0A55  E8 0C61 R           C          call    get_file_server         ; returns fs stn addr in dh
0A58  B2 01               C          mov     dl,1                    ; Read one sector
0A5A  B9 0001             C          mov     cx,1                    ; ...from the first FAT
                          C                                          ; es:bx is already set
                          C          lcall   vdisk_io                ; off we go...
                          C          bnz     short_bad_bpb           ; Oops, error, give up.
                          C  ;
0A66  B4 08               C          mov     ah,ve_bad_disk          ; Error is unkown media
0A68  26: 83 7F 01 FF     C          cmp     es:word ptr [bx+1],0FFFFH       ; Bytes 0 and 1 must be FFFF
0A6D  75 1C               C          jne     short_bad_bpb           ; Nope, error
                          C  ;
0A6F  26: 8A 07           C          mov     al,es:byte ptr [bx]     ; Get media descriptor
                          C          ;mov    si,offset double_8_bpb  ; Pointer to bpb
0A72  3C FF               C          cmp     al,0FFH                 ; check the media descriptor
0A74  74 15               C          je      short_bad_bpb           ; Always bad, must be swaped
                          C  ;
0A76  BE 046F R           C          mov     si, offset single_8_bpb
0A79  3C FE               C          cmp     al,0FEH
0A7B  74 11               C          je      Copy_FAT_bpb
                          C  ;
0A7D  BE 04AF R           C          mov     si,offset double_9_bpb
0A80  3C FD               C          cmp     al,0FDH
0A82  74 0A               C          je      Copy_FAT_bpb
                          C  ;
0A84  BE 046F R           C          mov     si,offset single_8_bpb
0A87  B0 FC               C          mov     al,0FCH
0A89  74 03               C          je      Copy_FAT_bpb
0A8B                      C  short_bad_bpb:
0A8B  E9 099B R           C          jmp     bad_bpb
                          C  ;
0A8E                      C  Copy_FAT_bpb:
0A8E  2E: A0 00A9 R       C          mov     al, cs:save_drive       ; Figure out which bpb we modify
```

```
 0A92  B4 00                C          mov     ah,0
 0A94  2E: F7 26 00AB R     C          mul     cs:type_bpb             ; Offset into bpb_table
 0A99  8B F8                C          mov     di,ax                   ; Move it so we can indirect...
 0A9B  50                   C          push    ax                      ; Save for returning to DOS
 0A9C  B9 000D              C          mov     cx, bpb_on_disk         ; Copy this many bytes
 0A9F                       C  Copy_2:                                 ; I'm too lazy to figure the movs
 0A9F  2E: 8A 04            C          mov     al, cs:byte ptr [si]    ; Get byte from our bpb
 0AA2  2E: 88 85 00DB R     C          mov     cs:byte ptr bpb_table[di],al    ; Stash in table
 0AA7  46                   C          inc     si
 0AA8  47                   C          inc     di
 0AA9  E2 F4                C          loop    Copy_2
                            C  ;
 0AAB  E9 0981 R            C          jmp     made_bpb
                            C  bpb1    endp
```

```
                        C           page
                        C   ;*****************************************************************************
                        C   ;*
                        C   ;*      DIM support
                        C   ;*
                        C   ;*      These procs support command channel handleing of DIMs.  Write_disk_info
                        C   ;*      passes us a DIM, a station address and a drive number and we set
                        C   ;*      that driver virtual to that station.  Read_disk_info returns a
                        C   ;*      pointer to bpb table for the drive requested.
                        C   ;*
                        C   ;*      Registers in:
                        C   ;*          AH = station address
                        C   ;*          AL = [BPriDim, BSecDim]
                        C   ;*       DS:SI = Pointer to DIM.
                        C   ;*
                        C   ;*          CY set on drive out of range, clear otherwise
                        C   ;*
                        C           public  write_disk_info
 0AAE                   C   write_disk_info proc    near
                        C           assume  ds: nothing
 0AAE  52               C           push    dx                              ; Save some registers
 0AAF  57               C           push    di
                        C   ;
 0AB0  50               C           push    ax
 0AB1  8A 44 03         C           mov     al,byte ptr [si].EDDriveNum+1   ; See who we are talking about
                        C                                                   ; Stored in 68K integer
 0AB4  2E: 3A 06 0000 E C           cmp     al,vp_vol_unit                  ; allow vp mount as a special case
 0AB9  75 05            C           jne     wdi0                            ; no, normal processing
 0ABB  FE C8            C           dec     al                              ; scale for table
 0ABD  EB 2B 90         C           jmp     wdi3
                        C   ;
 0AC0                   C   wdi0:
 0AC0  FE C8            C           dec     al                              ; FS thinks A: is 1 (in 1 .. 254)
 0AC2  2E: 3A 06 001B R C           cmp     al,our_drive                    ; Is it one of ours
 0AC7  7C 07            C           jl      wdi2
 0AC9  2E: 3A 06 0012 R C           cmp     al,byte ptr max_drive           ; is it in range
 0ACE  7C 1A            C           jl      wdi3
 0AD0                   C   wdi2:
 0AD0  2E: 3A 06 0090 R C           cmp     al, remapped1
 0AD5  74 13            C           je      wdi3                            ; If remapped drive, then ok
 0AD7  2E: 3A 06 0091 R C           cmp     al, remapped2
 0ADC  74 0C            C           je      wdi3
 0ADE  2E: 3A 06 0092 R C           cmp     al, remapped3
 0AE3  74 05            C           je      wdi3
 0AE5  58               C           pop     ax
 0AE6  5F               C           pop     di
 0AE7  5A               C           pop     dx
 0AE8  F9               C           stc
 0AE9  C3               C           ret
 0AEA                   C   wdi3:
 0AEA  B4 00            C           mov     ah,0
 0AEC  2E: F7 26 00AB R C           mul     type_bpb                        ; Offset into table
 0AF1  8B F8            C           mov     di,ax                           ; In an index register
 0AF3  58               C           pop     ax
```

```
                        C  ;
0AF4  80 3C 01          C          cmp     [si].EDCode, MountCode          ; If it's not a mount, then...
                        C          bnz     wdil                            ;   ignore it.
                        C  ;
0AFC  3C 01             C          cmp     al, BPriDim                     ; See if we're primary dim
0AFE  74 61             C          je      wdi4                            ; Yes, off we go.
                        C  ;
0B00  50                C          push    ax
0B01  2E: 88 A5 00F0 R  C          mov     cs: bpb_table[di].BSecFS, ah    ; Set secondary fs
                        C  ;
0B06  8A 44 04          C          mov     al,[si].EDFileType
0B09  2E: 38 85 00F7 R  C          cmp     cs:bpb_table[di].BFileType,al   ; Copy file type.
0B0E  75 4C             C          jne     wderr
                        C  ;
0B10  8A 44 05          C          mov     al,[si].EDFileSubType   ; Copy file sub type
0B13  2E: 38 85 00F8 R  C          cmp     cs:bpb_table[di].BFileSubType,al
0B18  75 42             C          jne     wderr
                        C  ;
0B1A  8A 44 06          C          mov     al,[si].EDAccess                ; Access
0B1D  2E: 38 85 00F9 R  C          cmp     cs:bpb_table[di].BAccess,al
0B22  75 38             C          jne     wderr
                        C  ;
0B24  8A 44 07          C          mov     al,[si].EDShr                   ; share
0B27  2E: 38 85 00FA R  C          cmp     cs:bpb_table[di].BShr,al
0B2C  75 2E             C          jne     wderr
                        C  ;
0B2E  52                C          push    dx                              ; Save for div
0B2F  8B 54 08          C          mov     dx,word ptr [si].EDSize
0B32  86 F2             C          xchg    dh,dl                           ; Bytes are swaped
0B34  8B 44 0A          C          mov     ax,word ptr [si].EDSize+2       ; Get size from DIM
0B37  86 E0             C          xchg    ah,al                           ; Bytes are swaped
0B39  2E: F7 36 00AF R  C          div     block_size                      ; Div by 512 (always for this)
0B3E  2E: 39 85 00E3 R  C          cmp     cs:bpb_table[di].BSectCount,ax  ; Size in blocks.
0B43  5A                C          pop     dx
0B44  75 16             C          jne     wderr
                        C  ;
0B46  2E: C6 85 00EE R FF C        mov     cs:bpb_table[di].BMediaChanged, MediaChanged
0B4C  58                C          pop     ax
0B4D  2E: C6 85 00F3 R 2D C        mov     cs:bpb_table[di].BSecRead, NotUsedState
0B53  2E: C6 85 00F4 R 57 C        mov     cs:bpb_table[di].BSecWrite, WriteState
0B59  E9 0C14 R         C          jmp     wdil0
                        C  ;
0B5C                    C  wderr:
0B5C  58                C          pop     ax
0B5D  5F                C          pop     di
0B5E  5A                C          pop     dx
0B5F  F9                C          stc
0B60  C3                C          ret
                        C  ;
0B61                    C  wdi4:
0B61  50                C          push    ax
0B62  2E: 88 A5 00EF R  C          mov     cs:bpb_table[di].BPriFS,ah      ; Set station address
                        C  ;
0B67  8A 44 04          C          mov     al,[si].EDFileType
```

```
0B6A  2E: 88 85 00F7 R      C          mov     cs:bpb_table[di].BFileType,al   ; Copy file type.
                            C  ;
0B6F  8A 44 05              C          mov     al,[si].EDFileSubType   ; Copy file sub type
0B72  2E: 88 85 00F8 R      C          mov     cs:bpb_table[di].BFileSubType,al
                            C  ;
0B77  8A 44 06              C          mov     al,[si].EDAccess                ; Access
0B7A  2E: 88 85 00F9 R      C          mov     cs:bpb_table[di].BAccess,al
                            C  ;
0B7F  8A 44 07              C          mov     al,[si].EDShr                   ; share
0B82  2E: 88 85 00FA R      C          mov     cs:bpb_table[di].BShr,al
                            C  ;
0B87  52                    C          push    dx                              ; Save for div
0B88  8B 54 08              C          mov     dx,word ptr [si].EDSize
0B8B  86 F2                 C          xchg    dh,dl                           ; Bytes are swaped
0B8D  8B 44 0A              C          mov     ax,word ptr [si].EDSize+2       ; Get size from DIM
0B90  86 E0                 C          xchg    ah,al                           ; Bytes are swaped
0B92  2E: F7 36 00AF R      C          div     block_size                      ; Div by 512 (always for this)
0B97  2E: 89 85 00E3 R      C          mov     cs:bpb_table[di].BSectCount,ax  ; Size in blocks.
0B9C  5A                    C          pop     dx
                            C  ;
0B9D  2E: C6 85 00F0 R 00   C          mov     cs:bpb_table[di].BSecFS, 0      ; Not used for now.
0BA3  2E: C6 85 00EE R FF   C          mov     cs:bpb_table[di].BMediaChanged, MediaChanged
0BA9  2E: C6 85 00F1 R 52   C          mov     cs:bpb_table[di].BPriRead, ReadState
0BAF  2E: C6 85 00F2 R 57   C          mov     cs:bpb_table[di].BPriWrite, WriteState
0BB5  2E: C6 85 00F3 R 2D   C          mov     cs:bpb_table[di].BSecRead, NotUsedState
0BBB  2E: C6 85 00F4 R 2D   C          mov     cs:bpb_table[di].BSecWrite, NotUsedState
0BC1  2E: C6 85 00F5 R 00   C          mov     cs:bpb_table[di].BPriErr, ve_ok
0BC7  2E: C6 85 00F6 R 00   C          mov     cs:bpb_table[di].BSecErr, ve_ok
0BCD  58                    C          pop     ax
0BCE  EB 44 90              C          jmp     wdi10
0BD1                        C  wdi1:
0BD1  80 3C 02              C          cmp     [si].EDCode, UnmountCode        ; Is it unmount
                            C          bnz     wdi11                           ; No, try another
                            C  ;
0BD9  3C 01                 C          cmp     al, BPriDim                     ; Primary file server
0BDB  75 1D                 C          jne     wdi12                           ; No, must be secondary
                            C  ;
0BDD  2E: C6 85 00EF R 00   C          mov     cs:bpb_table[di].BPriFS, 0      ; Trash file server stn addr
0BE3  2E: C6 85 00EE R FF   C          mov     cs:bpb_table[di].BMediaChanged, MediaChanged
0BE9  2E: 80 BD 00F0 R 00   C          cmp     cs:bpb_table[di].BSecFS, 0      ; If 0 then...
0BEF  75 23                 C          jne     wdi10
                            C  ;
0BF1  2E: C6 85 00F7 R 20   C          mov     cs:bpb_table[di].BFileType, ' ' ; Tell world nothing there
0BF7  EB 1B 90              C          jmp     wdi10
                            C  ;
0BFA                        C  wdi12:
0BFA  2E: C6 85 00F0 R 00   C          mov     cs:bpb_table[di].BSecFS, 0      ; Trash file server stn addr
0C00  2E: C6 85 00EE R FF   C          mov     cs:bpb_table[di].BMediaChanged, MediaChanged
0C06  2E: 80 BD 00EF R 00   C          cmp     cs:bpb_table[di].BPriFS, 0      ; If 0 then...
0C0C  75 06                 C          jne     wdi10
                            C  ;
0C0E  2E: C6 85 00F7 R 20   C          mov     cs:bpb_table[di].BFileType, ' ' ; Tell world nothing there
                            C  ;       and fall through into...
0C14                        C  wdi11:                                          ; Next dim type
```

```
0C14                    C  wdi10:                                        ; All done, return
0C14  5F                C          pop     di
0C15  5A                C          pop     dx
0C16  F8                C          clc
0C17  C3                C          ret
                        C  write_disk_info endp
                        C
                        C  ;*
                        C  ;*      read_disk_info
                        C  ;*
                        C  ;*      Registers in
                        C  ;*          AL = disk number (A: = 1)
                        C  ;*
                        C  ;*      Registers out
                        C  ;*          CY = clear if no error, set if error
                        C  ;*       DS:SI = pointer to bpb
                        C  ;*
                        C          public  read_disk_info
0C18                    C  read_disk_info  proc    near
                        C          assume  ds:cseg
0C18  C7 06 04E0 R 0000 C          mov     rh_status,0             ; Clear status just in case
0C1E  52                C          push    dx
0C1F  3A 06 0000 E      C          cmp     al,vp_vol_unit          ; special case vp unit
0C23  75 05             C          jne     rdi0
0C25  FE C8             C          dec     al                      ; scale unit unmber
0C27  EB 27 90          C          jmp     rdi2
                        C  ;
0C2A                    C  rdi0:
0C2A  FE C8             C          dec     al                      ; Everyone else thinks A: is 1.
0C2C  3A 06 001B R      C          cmp     al,our_drive
0C30  7D 15             C          jge     rdi1
                        C  ;
0C32  3A 06 0090 R      C          cmp     al, remapped1
0C36  74 18             C          je      rdi2
0C38  3A 06 0091 R      C          cmp     al, remapped2
0C3C  74 12             C          je      rdi2
0C3E  3A 06 0092 R      C          cmp     al, remapped3
0C42  74 0C             C          je      rdi2
                        C  ;
0C44  5A                C          pop     dx
0C45  F9                C          stc
0C46  C3                C          ret                             ; Not one of ours, tell caller
0C47                    C  rdi1:
0C47  3A 06 0012 R      C          cmp     al,byte ptr max_drive   ; See if above us
0C4B  7C 03             C          jl      rdi2
                        C  ;
0C4D  5A                C          pop     dx
0C4E  F9                C          stc
0C4F  C3                C          ret
                        C  ;
                        C  ;       AL is disk drive with A: = 0
                        C  ;
0C50                    C  rdi2:
0C50  50                C          push    ax
```

```
0C51  B4 00                C          mov     ah,0
0C53  F7 26 00AB R         C          mul     type_bpb
0C57  8B F0                C          mov     si,ax
0C59  58                   C          pop     ax              ; Get it back
                           C
                           C  ;--comment out so that ?drive does not clear MediaChanged flag in BPB.
                           C  ;       push    ax
                           C  ;
                           C  ;       cmp     byte ptr bpb_table[si].BMediaChanged,MediaNotTouched
                           C  ;       jz      rdi4            ; If media not touched then bpb must be ok
                           C  ;
                           C  ;       push    si              ; Save our register
                           C  ;
                           C  ;       cmp     al, our_drive
                           C  ;       jge     rdi5
                           C  ;       add     al, 0C0H
                           C  ;       jmp     short rdi6
                           C  ;rdi5:  sub     al,our_drive
                           C  ;
                           C  ;rdi6:
                           C  ;       mov     rh_unit_code,al ; Set unit number
                           C  ;
                           C  ;       mov     word ptr rh_buffer,offset bpb_buffer
                           C  ;       mov     ax,cs           ; Get segment for buffer
                           C  ;       mov     word ptr rh_buffer+2,ax
                           C  ;
                           C  ;       push    es
                           C  ;
                           C  ;       push    cs
                           C  ;       pop     es
                           C  ;
                           C  ;       push    bx
                           C  ;
                           C  ;       mov     bx,es           ; Set our pointer to the request header
                           C  ;       mov     rh_seg,bx
                           C  ;       mov     bx,offset rh_len
                           C  ;       mov     rh_off,bx
                           C  ;
                           C  ;       push    ds
                           C  ;       push   es
                           C  ;       push   ax
                           C  ;       push   bx
                           C  ;       push   cx
                           C  ;       push   dx
                           C  ;       push   di
                           C  ;       push   si
                           C  ;       push    bp
                           C  ;
                           C  ;       call    real_build_bpb
                           C  ;
                           C  ;       pop     bp
                           C  ;       pop    si              ;restore all of the registers
                           C  ;       pop    di
                           C  ;       pop    dx
```

```
                        C  ;       pop    cx
                        C  ;       pop    bx
                        C  ;       pop    ax
                        C  ;       pop    es
                        C  ;       pop    ds
                        C  ;       pop     bx
                        C  ;       pop     es
                        C  ;       pop     si
                        C  ;
                        C  ;rdi4:
                        C  ;       pop     ax
                        C  ;-------------------------------------------
0C5A  5A                C          pop     dx
0C5B  8D B4 00DB R      C          lea     si,bpb_table[si]
                        C  ;
                        C  ;---don't check for error since real_build_bpb wasn't called
                        C  ;       test    rh_status,8000H         ; An error occured
                        C  ;       jz      rdi3
                        C  ;
                        C  ;       cmp     al,8
                        C  ;       jne     rdi3
                        C  ;       mov     [si].BFileType,' '      ; If nothing mounted then tell 'em
                        C  ;rdi3:
                        C  ;-------------------------------------------
0C5F  F8                C          clc
0C60  C3                C          ret
                        C  read_disk_info  endp
```

```
                        C               page
                        C       ;*****************************************************************************
                        C       ;*
                        C       ;*      Get_file_server
                        C       ;*
                        C       ;*      Get the file server.  Return in dh.  No other regs
                        C       ;*      touched (except flags).
                        C       ;*
 0C61                   C       get_file_server proc    near
 0C61  8A 36 00AA R     C               mov     dh, io_fs
 0C65  C3               C               ret
                        C       get_file_server endp
                        C
                        C
                        C       ;*      get_drive
                        C       ;*      Convert DOS unit number to logical drive number (a=0, b=1, etc)
                        C       ;*      We change to floppy/hard disk look alikes set up by boot program
                        C       ;*      to range C0+logical drive number.  Unit numbers come in the range
                        C       ;*      0 .. n, and are relative to the variable our_drive which is
                        C       ;*      set at init time.
                        C       ;*
 0C66                   C       get_drive       proc    near
 0C66  26: 8A 47 01     C                       mov     al, es:[bx].srh_ucd_fld
 0C6A  3C C0            C                       cmp     al, 0C0H
 0C6C  72 04            C                       jb      gd1
 0C6E  2C C0            C                       sub     al, 0C0H
 0C70  EB 04            C                       jmp     short gd2
 0C72                   C       gd1:
 0C72  02 06 001B R     C                       add     al, our_drive
 0C76                   C       gd2:
 0C76  C3               C                       ret
                        C       get_drive       endp
                        C       ;


                                                %OUT    ..........VDISK_MISCELLANEOUS
                        C                       include vdmisc.asm                      ;miscellaneous routines
                        C
                        C
                        C                       subttl  vdisk_miscellaneous
                        C
```

```
                        C                       page
                        C
 0C77                   C  get_tod              proc    near
 0C77  50               C                       push    ax
 0C78  2E: A1 0094 R    C                       mov     ax,cs:int_1a_1          ;calculate an "int 1a"
 0C7C  05 1E2B          C                       add     ax,int_1a_2
 0C7F  D1 F8            C                       sar     ax,1                    ;int 1a is cd, 1a is lacd is
 0C81  2E: A3 0CD7 R    C                       mov     cs:int_ins,ax           ;  is 6861 is (5999+7723) div 2
 0C85  E8 0CD4 R        C                       call    do_int_1a               ;keep funny code a little out of sight
                        C                                                       ;must have cd:1a in al:ah
 0C88  2E: C7 06 0CD7 R 0A0E C                  mov     cs:int_ins,int_1a_2/3   ;wipe out "int 1a"
 0C8F  58               C                       pop     ax
 0C90  C3               C                       ret
                        C  get_tod              endp
                        C
                        C  ;-------------------------------------------------------------------------
                        C  ;
                        C  ; Produce a half click - changes flags.
                        C  ; V5
                        C  ;-------------------------------------------------------------------------
                        C
                        C                       public  half_click
 0C91                   C  half_click           proc    near            ; toggles speaker port
 0C91  50               C                       push    ax
                        C
                        C                       else            ; not for_z120
 0C92  E4 61            C                       in      al,spkr_port
 0C94  33 06 0014 R     C                       xor     ax,click_on     ; if off, no change to spkr_port
 0C98  24 FE            C                       and     al,0FEh         ; not timer gate
 0C9A  E6 61            C                       out     spkr_port,al
                        C                       endif           ; for_z120
                        C
 0C9C  58               C                       pop     ax
 0C9D  C3               C                       ret
                        C  half_click           endp
                        C
                        C  ;--------- old code -----------------------------------
                        C  ;save_port           db      ?               ; Save status of port here.
                        C  ;
                        C  ;start_noise         proc    near
                        C  ;                    push    ax              ;save needed regs
                        C  ;                    cmp     click_on,0      ;noise or not
                        C  ;                    je      no_click_start  ;jump if no
                        C  ;                    mov     al,10110110b    ;program timer two
                        C  ;                    out     timer_ctrl,al
                        C  ;                    mov     ax,15000        ;divide for approx 80hz
                        C  ;                    out     timer_two,al    ;which is quite discreet
                        C  ;                    mov     al,ah
                        C  ;                    out     timer_two,al
                        C  ;                    in      al,spkr_port    ;remember state of i/o port
                        C  ;                    mov     save_port,al
                        C  ;                    or      al,3            ;enable both speaker gates
                        C  ;                    out     spkr_port,al
                        C  ;no_click_start:
```

```
C  ;                pop     ax
C  ;                ret
C  ;start_noise     endp
C  ;
C  ;end_noise       proc    near
C  ;                push    ax
C  ;                cmp     click_on,0      ;noise or not
C  ;                je      no_click_stop   ;jump if no
C  ;                mov     al,save_port
C  ;                out     spkr_port,al
C  ;no_click_stop:
C  ;                pop     ax
C  ;                ret
C  ;end_noise       endp
```

```
                        C               page
                        C  ;--------------------------------------------------------------------------
                        C  ;
                        C  ; setiobsw enables special iobs for reading from "execute only" volumes
                        C  ;  al:special iob function code to use
                        C  ;  ah:drive letter (in ascii, capital letter) to permit special iob's on
                        C  ;  cx:duration of permision in ticks of 18.2 per second
                        C  ;
 0C9E                   C  setiobsw        proc    far
 0C9E  50               C                  push    ax
 0C9F  80 EC 41         C                  sub     ah,'A'
 0CA2  2E: 2A 26 001B R C                  sub     ah,cs:our_drive
 0CA7  2E: 88 26 0093 R C                  mov     cs:special_drive,ah
 0CAC  2E: A2 0089 R    C                  mov     cs:iob_switch,al
 0CB0  2E: 89 0E 008A R C                  mov     cs:iob_timeout,cx
 0CB5  E8 0CBA R        C                  call    set_end_special
 0CB8  58               C                  pop     ax
 0CB9  CB               C                  ret
                        C  setiobsw        endp
                        C  ;
 0CBA                   C  set_end_special proc    near
 0CBA  51               C                  push    cx                      ;free up some regs.
 0CBB  52               C                  push    dx
 0CBC  E8 0C77 R        C                  call    get_tod                 ;get tod in cx:dx
 0CBF  2E: 03 16 008A R C                  add     dx,cs:iob_timeout       ;calculate expiration time of
 0CC4  83 D1 00         C                  adc     cx,0                    ;  special iob state
 0CC7  2E: 89 16 008C R C                  mov     cs:end_special_lo,dx    ;set low time
 0CCC  2E: 89 0E 008E R C                  mov     cs:end_special_hi,cx    ;set high time
 0CD1  5A               C                  pop     dx
 0CD2  59               C                  pop     cx
 0CD3  C3               C                  ret
                        C  set_end_special endp
                        C  ;
 0CD4                   C  do_int_1a       proc    near
 0CD4  80 C4 E6         C                  add     ah,-1ah                 ;convert the 1a in ah to zero
 0CD7  ????             C  int_ins         dw      ?                       ;an "int 1a" tod call is built here
 0CD9  F7 D0            C                  not     ax                      ;does nothing useful, just adds
                        C                                                  ;  to the confusion
 0CDB  C3               C                  ret
                        C  do_int_1a       endp


                                           %OUT    ..........VDISK_INITIALISATION
                        C                  include vdinit.asm                      ;initialisation
                        C
                        C                  subttl  vdisk_initialization
```

```
                        C               page
                        C
                        C ;*****************************************************************************
                        C ;
                        C ;  init
                        C ;
                        C ;       6/26/85         cwp     Moved allocation of additional rb's to 14_init.
                        C ;      10/24/85         gkn     Added conditional assembly to support Zenith 120
                        C ;      12/31/85         gkn     Fixed SR5 bug for setting the number of drives to
                        C ;                               less than the REAL number.
                        C ;
= 0CDC                  C bpb_buffer    equ     $               ; Start of initilization code.
= 0000                  C Trace   equ   0
                        C ;
0CDC                    C init_vdisk:
0CDC  C7 06 0098 R 0000 E C             mov     my_end, offset driver_end       ; Init this here
                        C ;
                        C ;Here we correct a bug in the pc rom bios.  The bug is that int 10h,
                        C ;function 3 (read cursor position & type) returns the value for a color
                        C ;monitor cursor regardless of whether a color or monochrome monitor is
                        C ;being used, unless the cursor has been previously set using int 10h,
                        C ;function 1.  The fix, of course, is to set the cursor explicity.  So
                        C ;we do an equipment determination, then set the cursor accordingly.
0CE2                    C set_curs:
                        C                       ife     for_z120
0CE2  50                C                       push    ax
0CE3  CD 11             C                       int     11h             ;equipment determination--
                        C                                               ; takes no input parameters
0CE5  24 30             C                       and     al,30h          ;bits 4 & 5 on means monochrome--
0CE7  3C 30             C                       cmp     al,30h          ; anything else is color.
0CE9  75 0F             C                       jnz     color_mon       ;process only if monochrome
0CEB  51                C                       push    cx
0CEC  56                C                       push    si              ;these three destroyed by int 10h
0CED  57                C                       push    di
0CEE  55                C                       push    bp
0CEF  B9 0C0D           C                       mov     cx,0c0dh        ;monochrome cursor
0CF2  B4 01             C                       mov     ah,1            ;set cursor
0CF4  CD 10             C                       int     10h             ;video
0CF6  5D                C                       pop     bp
0CF7  5F                C                       pop     di
0CF8  5E                C                       pop     si
0CF9  59                C                       pop     cx
0CFA                    C color_mon:
0CFA  58                C                       pop     ax
                        C                       endif   ; for_z120
                        C
0CFB  E8 0E33 R         C               call    vdisk_init
0CFE  E8 0D5A R         C               call    FindDrive       ; Find out what outside world thinks we are.
                        C ;
                        C               write   'Return from find drive.  Our drive is '
                        C               writebyte our_drive
                        C               writeln ' '
                        C ;
0D01  E8 0000 E         C               call    14_init         ; Go forth and intilize level 4.
```

```
                        C           writeln 'Back from 14_init'
                        C   ;
                        C   ;
 0DD4  2E: 8E 06 0087 R C           mov     es,cs:rh_seg                    ;restore es:bx to request header
 0DD9  2E: 8B 1E 0085 R C           mov     bx,cs:rh_off
 0DDE  A0 0012 R        C           mov     al,byte ptr max_drive           ;get maximum number of drives
                        C                                                   ;***Start of bug fix (12-31-85)
 0DE1  8A 26 001B R     C           mov     ah, our_drive                   ;get number of drives in system
 0DE5  FE C4            C           inc     ah                              ;drive number starts with 0
 0DE7  3A E0            C           cmp     ah, al                          ;who has more drives ?
 0DE9  76 05            C           jbe     ok_dr                           ;If we do, jump
 0DEB  8A C4            C           mov     al, ah                          ;If DOS does,
 0DED  A2 0012 R        C           mov     byte ptr max_drive, al          ;save it
 0DF0                   C   ok_dr:                                          ;***End of SR5 bug fix (12-31-85)
 0DF0  2A 06 001B R     C           sub     al, our_drive                   ;Any drives left
 0DF4  26: 88 47 0D     C           mov     es:[bx].IUnitCount, al          ;return number of drives
 0DF8  A2 000A R        C           mov     dev_name,al                     ;set number of drives in header
                        C   ;
 0DFB  8D 16 041B R     C           lea     dx, bpb_ptr_table
 0DFF  26: 89 57 12     C           mov     es:bpb_ptr_off[bx],dx           ; Point to bpb pointer table
 0E03  26: 8C 4F 14     C           mov     es:bpb_ptr_seg[bx],cs
                        C   ;
                        C   ;       The following code fixes a bug in DOS 2.x.  When a single file
                        C   ;       contains both block and serial drivers, DOS 2.x does not correctly
                        C   ;       allocation memory.  Hence, we allocate memory for it.
                        C   ;
 0E07  A1 0098 R        C           mov     ax, my_end
 0E0A  05 098C          C           add     ax,(DOS_disk_entry_size*def_max_drive)
 0E0D  A3 0096 R        C           mov     your_end,ax                     ; Offset for other's.
                        C   ;
 0E10  A1 0098 R        C           mov     ax, my_end
 0E13  26: 89 47 0E     C           mov     es:word ptr [bx].IDriverEnd,ax  ; Offset of end of driver
 0E17  26: 8C 4F 10     C           mov     es:word ptr [bx+2].IDriverEnd,cs
                        C   ;
                        C           write   'Drive end is at '
                        C           writeint        cs
                        C           write   ':'
                        C           writeint        ax
                        C           writeln ' '
                        C   ;
                        C           status  done,noerror,0   ;set status word (done,noerror)
 0E27  E9 0588 R        C           jmp     exit
```

```
                        C           page
                        C   ;*****************************************************************************
                        C   ;*
                        C   ;*      FindDrive.  FindDrive first finds the offset of our drive by
                        C   ;*      looking at the drive parameter blocks for each of the drives
                        C   ;*      until we find one that returns not there (OFFH).  This value
                        C   ;*      is stored in variable our_drives.  We then check the int13_loaded
                        C   ;*      flag to see if we just loaded int 13.  If we did, we're done.
                        C   ;*      If not, then we must remap the drives mapped by int13 from
                        C   ;*      the default drivers to our own.
                        C   ;*
                        C   ;*              "Piece of cake, trust me."
                        C   ;*                              S. Dillion, 1984
                        C   ;*
                        C
 0D5A                   C   FindDrive       proc    near
 0D5A  B4 30            C                   mov     ah, DOS_VERS
 0D5C  CD 21            C                   int     21H                     ; See what version we looking at
                        C   ;
 0D5E  3C 03            C                   cmp     al, 3                   ; AL contains major version
 0D60  72 13            C                   jb      fd10                    ; Not DOS 3.x, process old way
                        C   ;
 0D62  06               C                   push    es
 0D63  8E 06 0087 R     C                   mov     es, rh_seg              ; Get request block
 0D67  8B 1E 0085 R     C                   mov     bx, rh_off
 0D6B  26: 8A 57 16     C                   mov     dl, es:[bx].IOurDrive   ; Get unit count
                        C   ;
 0D6F  B4 52            C                   mov     ah,GET_IN_VARS          ; Get pointer to internal vars
 0D71  CD 21            C                   int     21H                     ; Do it.
 0D73  EB 09            C                   jmp     short fd20
                        C
 0D75                   C   fd10:
 0D75  06               C                   push    es
 0D76  B4 52            C                   mov     ah,GET_IN_VARS          ; Get pointer to internal vars
 0D78  CD 21            C                   int     21H                     ; Do it.
                        C   ;
                        C   ;               es:bx now points to DOS variables.  We use two fields:
                        C   ;               last_drive, which at this point in the DOS initilization
                        C   ;               process is the index of the last drive installed by
                        C   ;               the previous driver; and dpb_chain which is the chain
                        C   ;               of disk paramter blocks used by DOS to map from DOS
                        C   ;               drive letter to driver address, driver drive number.
                        C   ;
 0D7A  26: 8B 57 10     C                   mov     dx,es:[bx].last_drive   ; Number of drives in system
 0D7E                   C   fd20:
 0D7E  88 16 001B R     C                   mov     our_drive,dl            ; Stash it
                        C                   write   'Drive at offset '
 0D82  B6 00            C                   mov     dh,0
                        C                   writeint        dx
                        C                   writeln ' '
                        C   ;
                        C   ;               Remap DOS drivers for floppies mapped by boot program.
                        C   ;               We need to check floppies A and B and the hard disk
                        C   ;               whose drive we need to detirmine.  This remapping is
```

```
                        C  ;               done by three steps, first setting the remapped variable
                        C  ;               to the DOS drive number (A:=0), second setting the
                        C  ;               dpb driver address to point to ourseleves, and third
                        C  ;               setting the dpb_UNIT field to the drive number + 0C0H.
                        C  ;               Final values of remapped are:
                        C  ;
                        C  ;                   remapped1 = 0 if A: is mapped virtual, 0FFH if not
                        C  ;                   remapped2 = 1 if B: is mapped virutal, 0FFH if not
                        C  ;                   remapped3 = drive number for hard disk (C:=2) if mapped.
                        C  ;
 0D84  C6 06 0090 R FF  C                   mov     remapped1,0FFH          ; Nothing remapped
 0D89  C6 06 0091 R FF  C                   mov     remapped2,0FFH          ; Nothing remapped
 0D8E  C6 06 0092 R FF  C                   mov     remapped3,0FFH          ; Nothing remapped
                        C  ;
                        C  ;               First check the floppies,  if A: is not mapped, we
                        C  ;               skip B:.  If A: is mapped, we remap it for the driver
                        C  ;               and check B:
                        C  ;
 0D93  B4 F1            C                  mov     ah, 0F1H                ; Get drive mapping
 0D95  B0 00            C                  mov     al, 0                   ; Floppy drive 0
 0D97  CD 13            C                  int     13H                     ; Do it
                        C  ;
 0D99  80 FA C0         C                  cmp     dl,0C0H                 ; Is it mapped to network?
 0D9C  72 40            C                  jb      fd30                    ; No
 0D9E  80 FA FF         C                  cmp     dl, 0FFH                ; Is it not mapped
 0DA1  74 3B            C                  je      fd30                    ; Yes
                        C  ;
 0DA3  06               C                  push    es
 0DA4  53               C                  push    bx
 0DA5  C6 06 0090 R 00  C                  mov     remapped1, 0H
 0DAA  26: C4 37        C                  les     si, es:dword ptr [bx].dpb_chain ; Get pointer to entry for 0
 0DAD  26: C6 44 01 C0  C                  mov     es:[si].dpb_UNIT, 0C0H
 0DB2  26: 8C 4C 14     C                  mov     es:[si].dpb_driver_addr.segp,cs
 0DB6  26: C7 44 12 0000 C                 mov     es:[si].dpb_driver_addr.off, 0
                        C  ;
 0DBC  CD 11            C                  int     11H
 0DBE  A8 01            C                  test    al, 1                   ; Is there a floppy in sys
 0DC0  75 1A            C                  jne     fd25                    ; Yes, go on
                        C  ;
 0DC2  26: C4 74 18     C                  les     si, es:[si].dpb_next_dpb        ; dpb for drive b:
 0DC6  FE C2            C                  inc     dl                      ; For drive b:
 0DC8  C6 06 0091 R 01  C                  mov     remapped2, 1H           ; second remapping
 0DCD  26: C6 44 01 C1  C                  mov     es:[si].dpb_UNIT, 0C1H
 0DD2  26: 8C 4C 14     C                  mov     es:[si].dpb_driver_addr.segp,cs
 0DD6  26: C7 44 12 0000 C                 mov     es:[si].dpb_driver_addr.off, 0
                        C  ;
 0DDC                   C  fd25:
 0DDC  5B               C                  pop     bx
 0DDD  07               C                  pop     es
```

```
                        C               page
                        C  ;
                        C  ;            Remap DOS drivers for hard disk mapped by boot program.
                        C  ;            We first see if drive 80 is mapped virtual.  If it is
                        C  ;            then we need to see what DOS drive the hard disk is mapped
                        C  ;            in as.  Note that we assume the IBM PC DOS ordering of
                        C  ;            floppies followed by hard disks.
                        C  ;
 0DDE                   C  fd30:
                        C               writeln 'Try drive 2'
 0DDE  B4 F1            C               mov     ah, 0F1H                ; Get drive mapping
 0DE0  B0 80            C               mov     al, 80H                 ; Hard disk 0
 0DE2  CD 13            C               int     13H                     ; Do it
                        C  ;
                        C               write   'returned '
                        C               writebyte       dl
                        C               writeln ' '
                        C  ;
 0DE4  80 FA FF         C               cmp     dl, 0FFH                ; Is it not mapped
 0DE7  74 05            C               je      fd33                    ; Yes
 0DE9  80 FA C0         C               cmp     dl,0C0H                 ; Is it mapped to network?
 0DEC  77 03            C               ja      fd35                    ; No
 0DEE                   C  fd33:
 0DEE  EB 41 90         C               jmp     fd80
                        C  ;
                        C  ;            Get the number of floppies from the switches.  Note
                        C  ;            the special rules in the case of 0 and 1 floppies.
                        C  ;
 0DF1                   C  fd35:
 0DF1  CD 11            C               int     11H                     ; Equipment detirmination
                        C               write   'Equip returns '
                        C               writeint        ax
                        C               writeln ' '
                        C  ;
 0DF3  A8 01            C               test    al, 1                   ; Is there no drive?
 0DF5  75 04            C               jne     fd40
 0DF7  B4 02            C               mov     ah,2                    ; DOS thinks minimum 2
 0DF9  EB 12            C               jmp     short fd50
 0DFB                   C  fd40:
 0DFB  8A E0            C               mov     ah,al
 0DFD  80 E4 C0         C               and     ah,0C0H                 ; Get top two bits
 0E00  B1 06            C               mov     cl, 6                   ; 6 bits
 0E02  D2 EC            C               shr     ah,cl
 0E04  FE C4            C               inc     ah
 0E06  80 FC 02         C               cmp     ah,2
 0E09  73 02            C               jae     fd50
 0E0B  B4 02            C               mov     ah,2
 0E0D                   C  fd50:
                        C               write   ' Floppies = '
                        C               writebyte       ah
                        C               writeln ' '
                        C  ;
                        C  ;
                        C  ;            We now walk down the dpb chain to find the dpb for the
```

```
                        C  ;               hard disk.  AH contains the number of floppies.
                        C  ;
 0E0D  26: C4 37        C                  les     si, es:dword ptr [bx].dpb_chain ; Get pointer to entry for 0
 0E10  8A CC            C                  mov     cl, ah
 0E12  32 ED            C                  xor     ch,ch                   ; Convert count to 16 bits
                        C  ;
 0E14                   C  fd60:                                           ; Get hard disk dpb
 0E14  26: C4 74 18     C                  les     si, es:[si].dpb_next_dpb
 0E18  E2 FA            C                  loop    fd60
                        C
                        C                  write   'Dpb = '
                        C                  writeint        es
                        C                  write   ':'
                        C                  writeint        si
                        C                  writeln ' '
                        C  ;
                        C  ;               Finally there, so remap it.  We assume only one hard
                        C  ;               disk is mapped  to be virtual.  A real hard disk
                        C  ;               can still be mapped from 81H to 80H.  AH still contains
                        C  ;               the number of floppies.
                        C  ;
                        C  ;
 0E1A  8A D4            C                  mov     dl, ah
 0E1C  80 C2 C0         C                  add     dl, 0C0H                ; Calculate correct unit number
                        C  ;
 0E1F  26: 88 54 01     C                  mov     es:[si].dpb_UNIT, dl
 0E23  26: 8C 4C 14     C                  mov     es:[si].dpb_driver_addr.segp,cs
 0E27  26: C7 44 12 0000 C                 mov     es:[si].dpb_driver_addr.off, 0
 0E2D  88 26 0092 R     C                  mov     remapped3, ah           ; Drive number of remapped drive
                        C  ;
 0E31                   C  fd80:
 0E31  07               C                  pop     es
 0E32  C3               C                  ret
                        C  FindDrive       endp
```

```
                        C           page
                        C   ;-----------------------------------------------------------------------------
                        C   ;
                        C   ; This is the initialisation code for Vdisk. It is is responsible for :
                        C   ;   1. initialising vdisk's internal drive and address tables, and the internal
                        C   ;      vdisk variables.
                        C   ;
                        C   ;-----------------------------------------------------------------------------
                        C
                        C
 0E33                   C   vdisk_init      proc    near
                        C
 0E33                   C   vdisk_init_label:
                        C
 0E33  50               C                   push    ax
 0E34  53               C                   push    bx
 0E35  51               C                   push    cx
 0E36  56               C                   push    si
 0E37  57               C                   push    di
 0E38  1E               C                   push    ds
 0E39  06               C                   push    es
 0E3A  0E               C                   push    cs                              ;set DS up to VDISC segment
 0E3B  1F               C                   pop     ds
                        C                   assume  ds:cseg
                        C
                        C                   write   'In init '
                        C                   writeint        cs
                        C                   writeln ' '
                        C
                        C   ;
                        C   ;       Initialize the bpb table and the bpb pointer table.
                        C   ;
 0E3C  B9 001A          C                   mov     cx, length bpb_table
 0E3F  BE 0000          C                   mov     si,0                    ; init all bpb's
 0E42  BF 0000          C                   mov     di,0
 0E45                   C   V5_bpb_loop:
                        C   ;
 0E45  8D 84 00DB R     C                   lea     ax,bpb_table[si]        ; Set pointer table
 0E49  89 85 041B R     C                   mov     bpb_ptr_table[di],ax
                        C   ;
                        C   ;       Initlize bpb to look like a double sided floppy.
                        C   ;
 0E4D  C7 84 00DB R 0200 C                  mov     word ptr bpb_table[si].BBytesPerSect,200H
 0E53  C6 84 00DD R 02  C                   mov     byte ptr bpb_table[si].BSectPerClust,2
 0E58  C7 84 00DE R 0001 C                  mov     word ptr bpb_table[si].BResvSect,1
 0E5E  C6 84 00E0 R 02  C                   mov     byte ptr bpb_table[si].BFAtCount,2
 0E63  C7 84 00E1 R 0070 C                  mov     word ptr bpb_table[si].BDirEntries,70H
 0E69  C7 84 00E3 R 02D0 C                  mov     word ptr bpb_table[si].BSectCount,2d0H
 0E6F  C6 84 00E5 R FC  C                   mov     byte ptr bpb_table[si].BMediaDesc,0FCh
 0E74  C7 84 00E6 R 0002 C                  mov     word ptr bpb_table[si].BFATSize,2
 0E7A  C7 84 00E8 R 0009 C                  mov     word ptr bpb_table[si].BSectsPerTrack,9
 0E80  C7 84 00EA R 0002 C                  mov     word ptr bpb_table[si].BHeadCount,2
 0E86  C7 84 00EC R 0000 C                  mov     word ptr bpb_table[si].BHiddenSects,0
 0E8C  C6 84 00EE R FF  C                   mov     byte ptr bpb_table[si].BMediaChanged,LOW MediaChanged
```

```
0E91  C6 84 00EF R FE       C                   mov     byte ptr bpb_table[si].BPriFS, 0FEH
0E96  C6 84 00F0 R 00       C                   mov     byte ptr bpb_table[si].BSecFS, 0
0E9B  C6 84 00F1 R 52       C                   mov     byte ptr bpb_table[si].BPriRead, ReadState
0EA0  C6 84 00F2 R 57       C                   mov     byte ptr bpb_table[si].BPriWrite, WriteState
0EA5  C6 84 00F3 R 2D       C                   mov     byte ptr bpb_table[si].BSecRead, NotUsedState
0EAA  C6 84 00F4 R 2D       C                   mov     byte ptr bpb_table[si].BSecWrite, NotUsedState
0EAF  C6 84 00F5 R 00       C                   mov     byte ptr bpb_table[si].BPriErr, ve_ok
0EB4  C6 84 00F6 R 00       C                   mov     byte ptr bpb_table[si].BSecErr, ve_ok
0EB9  C6 84 00F7 R 20       C                   mov     byte ptr bpb_table[si].BFileType,' '
0EBE  C6 84 00F8 R 20       C                   mov     byte ptr bpb_table[si].BFileSubType,' '
0EC3  C6 84 00F9 R 00       C                   mov     byte ptr bpb_table[si].BAccess,0
0EC8  C6 84 00FA R 00       C                   mov     byte ptr bpb_table[si].BShr,0
                            C  ;
0ECD  83 C6 20              C                   add     si, type bpb
0ED0  83 C7 02              C                   add     di, type bpb_ptr_table
0ED3  49                    C                   dec     cx
0ED4  74 03                 C                   jz      V5_ja
0ED6  E9 0E45 R             C                   jmp     V5_bpb_loop
0ED9                        C  V5_ja:
                            C
                            C
                            C                   writeln 'End init'
                            C
0ED9  07                    C                   pop     es                          ;restore the original registers
0EDA  1F                    C                   pop     ds
0EDB  5F                    C                   pop     di
0EDC  5E                    C                   pop     si
0EDD  59                    C                   pop     cx
0EDE  5B                    C                   pop     bx
0EDF  58                    C                   pop     ax
                            C
0EE0  C3                    C                   ret
                            C
                            C  vdisk_init       endp
                            C
                            C  endif


0EE1                           CSEG             ends


                                                end
```

Macros:

Name	Lines
BNZ	3
FLIP	1
LCALL	2
STATUS	15
WRITE	9
WRITEBYTE	6
WRITEINT	6
WRITELN	9

Structures and Records:

Name	Width / Shift	# fields / Width	Mask	Initial
BLOCKO	001E	000D		
BOJUMP	0000			
BOOEM	0003			
BOBYTESPERSECT	000B			
BOSECTPERCLUST	000D			
BORESVSECT	000E			
BOFATCOUNT	0010			
BODIRENTRIES	0011			
BOSECTCOUNT	0013			
BOMEDIADESC	0015			
BOFATSIZE	0016			
BOSECTSPERTRACK	0018			
BOHEADCOUNT	001A			
BOHIDDENSECTS	001C			
BOOT_RECORD	0200	000C		
B_CODE	0000			
B_ID	01C0			
B_VPB_SPC	01D5			
B_VPB_CSF	01D6			
B_VPB_VOL_SS	01D7			
B_VPB_NFATS	01D9			
B_VPB_NF	01DA			
B_VPB_DATA_SS	01DC			
B_VPB_NDC	01DE			
B_VPB_SPF	01E0			
B_VPB_DIR_SS	01E1			
B_FILLER	01E3			
BPB	0020	0018		
BBYTESPERSECT	0000			
BSECTPERCLUST	0002			
BRESVSECT	0003			
BFATCOUNT	0005			
BDIRENTRIES	0006			
BSECTCOUNT	0008			
BMEDIADESC	000A			
BFATSIZE	000B			

```
  BSECTSPERTRACK . . . . . .      000D
  BHEADCOUNT . . . . . .          000F
  BHIDDENSECTS . . . . . .        0011
  BMEDIACHANGED  . . . . . .      0013
  BPRIFS . . . . . .              0014
  BSECFS . . . . . .              0015
  BPRIREAD . . . . . .            0016
  BPRIWRITE  . . . . . .          0017
  BSECREAD . . . . . .            0018
  BSECWRITE  . . . . . .          0019
  BPRIERR  . . . . . .            001A
  BSECERR  . . . . . .            001B
  BFILETYPE  . . . . . .          001C
  BFILESUBTYPE . . . . . .        001D
  BACCESS  . . . . . .            001E
  BSHR . . . . . .                001F
BUILD_BPB_RH . . . . . .          0016    0004
  BPBRH  . . . . . .              0000
  BPBMEDIADESC . . . . . .        000D
  BPBTEMPBUFFER  . . . . . .      000E
  BPBBPBPOINTER  . . . . . .      0012
DPB  . . . . . .                  005E    0012
  DPB_DRIVE  . . . . . .          0000
  DPB_UNIT . . . . . .            0001
  DPB_SECTOR_SIZE  . . . . . .    0002
  DPB_CLUSTER_MASK . . . . . .    0004
  DPB_CLUSTER_SHIFT  . . . . . .  0005
  DPB_FIRST_FAT  . . . . . .      0006
  DPB_FAT_COUNT  . . . . . .      0008
  DPB_ROOT_ENTRIES . . . . . .    0009
  DPB_FIRST_SECTOR . . . . . .    000B
  DPB_MAX_CLUSTER  . . . . . .    000D
  DPB_FAT_SIZE . . . . . .        000F
  DPB_DIR_SECTOR . . . . . .      0010
  DPB_DRIVER_ADDR  . . . . . .    0012
  DPB_MEDIA  . . . . . .          0016
  DPB_FIRST_ACCESS . . . . . .    0017
  DPB_NEXT_DPB . . . . . .        0018
  DPB_CURRENT_DIR  . . . . . .    001C
  DPB_DIR_TEXT . . . . . .        001E
ETNA_DIM . . . . . .              0012    0009
  EDCODE . . . . . .              0000
  EDSUBCODE  . . . . . .          0001
  EDDRIVENUM . . . . . .          0002
  EDFILETYPE . . . . . .          0004
  EDFILESUBTYPE  . . . . . .      0005
  EDACCESS . . . . . .            0006
  EDSHR  . . . . . .              0007
  EDSIZE . . . . . .              0008
  EDFILLER . . . . . .            000C
INIT_RH  . . . . . .              0017    0005
  IRH  . . . . . .                0000
  IUNITCOUNT . . . . . .          000D
  IDRIVEREND . . . . . .          000E
```

```
  IBPBARRAY . . . . . .  0012
  IOURDRIVE . . . . . .  0016
POINTER . . . . . .  0004  0002
  OFF . . . . . .  0000
  SEGP . . . . . .  0002
REQUEST_HEADER . . . . . .  000D  0005
  RHLENGTH . . . . . .  0000
  RHUNIT . . . . . .  0001
  RHCOMMANDCODE . . . . . .  0002
  RHSTATUS . . . . . .  0003
  RHDOSRESV . . . . . .  0005

Segments and Groups:

                N a m e         Size    Align   Combine Class

CSEG . . . . . .  0EE1  PARA  PUBLIC  'CODE'

Symbols:

                N a m e         Type    Value   Attr

ALLOC_MORE_RBS . . . . . .  L NEAR  0000  CSEG  External
ALT222 . . . . . .  L WORD  0020  CSEG  Global
ATTRIBUTE . . . . . .  L WORD  0004  CSEG

BAD_BPB . . . . . .  L NEAR  099B  CSEG
BIOS_ERROR . . . . . .  L BYTE  04CF  CSEG
BLOCK_SIZE . . . . . .  L WORD  00AF  CSEG
BPB1 . . . . . .  N PROC  08FD  CSEG  Length = 01B1
BPBA_PTR . . . . . .  Number  0012
BPBA_PTR_LEN . . . . . .  Number  0004
BPB_BUFFER . . . . . .  NEAR  0CDC  CSEG
BPB_ON_DISK . . . . . .  Number  000D
BPB_PTR_OFF . . . . . .  Number  0012
BPB_PTR_SEG . . . . . .  Number  0014
BPB_PTR_TABLE . . . . . .  L WORD  041B  CSEG  Length = 001A
BPB_TABLE . . . . . .  L 0020  00DB  CSEG  Length = 001A
BPRIDIM . . . . . .  Number  0001
BREAK_CALL . . . . . .  Number  001B
BR_ADDR_0 . . . . . .  Number  000E
BR_ADDR_1 . . . . . .  Number  0010
BR_ADDR_LEN . . . . . .  Number  0004
BSECDIM . . . . . .  Number  0002
BUILD_BPB . . . . . .  L NEAR  05E0  CSEG
BYTES_PER_SECTOR . . . . . .  L WORD  00AD  CSEG

CANCEL_SPECIAL . . . . . .  L NEAR  0573  CSEG
CHECK_FAT . . . . . .  L NEAR  0A46  CSEG
CLICK_ON . . . . . .  L WORD  0014  CSEG  Global
CODE_LEN . . . . . .  Number  01C0
COLOR_MON . . . . . .  L NEAR  0CFA  CSEG
COMPSTATE . . . . . .  Number  0043
CONTROL_WORD . . . . . .  L WORD  009A  CSEG
```

COPY_2	L NEAR	0A9F	CSEG	
COPY_BPB	L NEAR	0963	CSEG	
COPY_FAT_BPB	L NEAR	0A8E	CSEG	
COUNT	Number	0012		
COUNT_LEN	Number	0002		
DB_FLAG	L BYTE	001A	CSEG	Global
DEF_MAX_DRIVE	Number	001A		
DEV_INT	L NEAR	0518	CSEG	
DEV_NAME	L BYTE	000A	CSEG	
DEV_STRATEGY	L NEAR	050D	CSEG	
DIRSTRLEN	Number	0040		
DISKIO_CALL	Number	0013		
DOIO	L NEAR	079A	CSEG	
DOS_BUILD_BPB	Number	0002		
DOS_DISK_ENTRY_SIZE	Number	005E		
DOS_INIT	Number	0000		
DOS_IN_FLUSH	Number	0007		
DOS_IN_STATUS	Number	0006		
DOS_IOCTL_IN	Number	0003		
DOS_IOCTL_OUT	Number	000C		
DOS_MEDIA_CHECK	Number	0001		
DOS_OUTPUT	Number	0008		
DOS_OUT_FLUSH	Number	000B		
DOS_OUT_STATUS	Number	000A		
DOS_OUT_VERIFY	Number	0009		
DOS_READ	Number	0004		
DOS_READ_NOWAIT	Number	0005		
DOS_VERS	Number	0030		
DOUBLE_9_BPB	L 0020	04AF	CSEG	
DO_FUNCTION	L NEAR	0578	CSEG	
DO_INT_1A	N PROC	0CD4	CSEG	Length = 0008
DPBSIZ	Number	005E		
DPB_CHAIN	Number	0000		
DRIVERS_CHAIN	Number	0017		
DRIVER_BUSY	L BYTE	0083	CSEG	Global
DRIVER_END	L NEAR	0000	CSEG	External
DTA	Number	000E		
DTA_LEN	Number	0004		
END_SPECIAL_HI	L WORD	008E	CSEG	
END_SPECIAL_LO	L WORD	008C	CSEG	
EQUIP_CALL	Number	0011		
EXIT	L NEAR	0588	CSEG	
FALSE	Number	0000		
FD10	L NEAR	0D75	CSEG	
FD20	L NEAR	0D7E	CSEG	
FD25	L NEAR	0DDC	CSEG	
FD30	L NEAR	0DDE	CSEG	
FD33	L NEAR	0DEE	CSEG	
FD35	L NEAR	0DF1	CSEG	
FD40	L NEAR	0DFB	CSEG	
FD50	L NEAR	0E0D	CSEG	

```
FD60 . . . . . .  L NEAR  0E14  CSEG
FD80 . . . . . .  L NEAR  0E31  CSEG
FINDDRIVE  . . . .  N PROC  0D5A  CSEG    Length = 0009
FOR_Z120 . . . . .  Number  0000
FUNTAB . . . . . .  L BYTE  04F3  CSEG

GD1  . . . . . .  L NEAR  0C72  CSEG
GD2  . . . . . .  L NEAR  0C76  CSEG
GET_DRIVE  . . . .  N PROC  0C66  CSEG    Length = 0011
GET_FILE_SERVER  .  N PROC  0C61  CSEG    Length = 0005
GET_IN_VARS  . . .  Number  0052
GET_TOD  . . . . .  N PROC  0C77  CSEG    Length = 001A

HALF_CLICK . . . .  N PROC  0C91  CSEG    Global  Length = 000D

I1 . . . . . .  L NEAR  0725  CSEG
I2 . . . . . .  L NEAR  0757  CSEG
I3 . . . . . .  L NEAR  0748  CSEG
I4 . . . . . .  L NEAR  077D  CSEG
I5 . . . . . .  L NEAR  071D  CSEG
I6 . . . . . .  L NEAR  0740  CSEG
ID . . . . . .  L BYTE  00B9  CSEG
ID2  . . . . . .  L NEAR  09BA  CSEG
IDLOOP . . . . . .  L NEAR  09AC  CSEG
ID_LEN . . . . . .  Number  0015
IN1  . . . . . .  L NEAR  078C  CSEG
INIT_VDISK . . . .  L NEAR  0CDC  CSEG
INPUT  . . . . . .  L NEAR  06E5  CSEG
INTERRUPT  . . . .  L WORD  0008  CSEG
INT_1A_1 . . . . .  L WORD  0094  CSEG
INT_1A_2 . . . . .  Number  1E2B
INT_INS  . . . . .  L WORD  0CD7  CSEG
IN_FLUSH . . . . .  L NEAR  05E5  CSEG
IN_LIMIT . . . . .  L NEAR  07DF  CSEG
IN_STAT  . . . . .  L NEAR  05E5  CSEG
IOB_INIT . . . . .  Number  0004
IOB_READ . . . . .  Number  0001
IOB_SPECIAL  . . .  Number  0003
IOB_SWITCH . . . .  L BYTE  0089  CSEG
IOB_TIMEOUT  . . .  L WORD  008A  CSEG
IOB_WRITE  . . . .  Number  0002
IOCTL_IN . . . . .  L NEAR  05E5  CSEG
IOCTL_OUT  . . . .  L NEAR  05E5  CSEG
IO_FS  . . . . . .  L BYTE  00AA  CSEG
IO_LIMIT . . . . .  Number  0020
IO_RETURN  . . . .  L NEAR  07E8  CSEG
IS_20  . . . . . .  L NEAR  094F  CSEG

KBDIO_CALL . . . .  Number  0016

L4_INIT  . . . . .  L NEAR  0000  CSEG    External
L4_VARS  . . . . .  V DWORD 0000  CSEG    External
LAST_DRIVE . . . .  Number  0010
```

MADE_BPB	L NEAR	0981	CSEG	
MAP_INT	Number	00FD		
MARKER	L BYTE	00CE	CSEG	
MAX_DRIVE	L WORD	0012	CSEG	
MC1	L NEAR	05CE	CSEG	
MD	Number	000D		
MD_LEN	Number	0001		
MEDIACHANGED	Number	-0001		
MEDIADONTKNOW	Number	0000		
MEDIANOTTOUCHED	Number	0001		
MEDIA_CHECK	L NEAR	05B3	CSEG	
MEDIA_DESC	L BYTE	00A8	CSEG	
MORE_RBS	L WORD	001C	CSEG	Global
MOUNTCODE	Number	0001		
MY_END	L WORD	0098	CSEG	Global
NC_HEAD	L NEAR	0000	CSEG	External
ND_INPUT	L NEAR	05E5	CSEG	
NEXT_DEV	L WORD	0000	CSEG	
NIBM_OEM	L BYTE	00D3	CSEG	
NIL	Number	0000		
NOPCODE	Number	0000		
NORMAL_BPB	L NEAR	0908	CSEG	
NORMAL_FAT	L NEAR	0A51	CSEG	
NORMAL_INPUT	L NEAR	0797	CSEG	
NOTBUSY	L NEAR	0554	CSEG	
NOTHING_TO_DO	L NEAR	07E8	CSEG	
NOTMOUNTEDTYPE	Number	002D		
NOTUSEDSTATE	Number	002D		
NOT_20	L NEAR	09A6	CSEG	
NOT_BAD_BPB	L NEAR	0A21	CSEG	
O2	L NEAR	0682	CSEG	
O3	L NEAR	0691	CSEG	
O4	L NEAR	06B0	CSEG	
O5	L NEAR	0657	CSEG	
O6	L NEAR	067D	CSEG	
OEM_NAME	L BYTE	00B1	CSEG	
OK_BPB_SO_FAR	L NEAR	0A15	CSEG	
OK_DR	L NEAR	0D20	CSEG	
OLD_11_VECTOR	L DWORD	00A0	CSEG	
OLD_13_VECTOR	L DWORD	009C	CSEG	
OLD_1B_VECTOR	L DWORD	00A4	CSEG	
OUR_DRIVE	L BYTE	001B	CSEG	Global
OUT1	L NEAR	06BF	CSEG	
OUT2	L NEAR	06E4	CSEG	
OUTPUT	L NEAR	060F	CSEG	
OUT_FLUSH	L NEAR	05E5	CSEG	
OUT_STAT	L NEAR	05F9	CSEG	
OUT_VERIFY	L NEAR	060F	CSEG	
PRINT_BYTE	L NEAR	0000	CSEG	External
PRINT_CHAR	L NEAR	0000	CSEG	External
PRINT_CRLF	L NEAR	0000	CSEG	External

```
PRINT_HEX . . . . . .  L NEAR  0000  CSEG  External
PRINT_STRING . . . . .  L NEAR  0000  CSEG  External
PRINT_WORD . . . . . .  L NEAR  0000  CSEG  External

RBB1 . . . . . .  L NEAR  0819  CSEG
RBB2 . . . . . .  L NEAR  084A  CSEG
RBB3 . . . . . .  L NEAR  085A  CSEG
RBB4 . . . . . .  L NEAR  08B9  CSEG
RBB5 . . . . . .  L NEAR  0872  CSEG
RBB6 . . . . . .  L NEAR  08EC  CSEG
RBB7 . . . . . .  L NEAR  08DE  CSEG
RBB8 . . . . . .  L NEAR  083F  CSEG
RBB9 . . . . . .  L NEAR  0885  CSEG
RB_BUSY . . . . . .  L BYTE  0084  CSEG  Global
RB_PTR . . . . . .  V DWORD 0000  CSEG  External
RDIO . . . . . .  L NEAR  0C2A  CSEG
RDI1 . . . . . .  L NEAR  0C47  CSEG
RDI2 . . . . . .  L NEAR  0C50  CSEG
READSTATE . . . . . .  Number  0052
READ_DISK_INFO . . . . . .  N PROC  0C18  CSEG  Global  Length = 0049
REAL_BUILD_BPB . . . . . .  N PROC  07F3  CSEG  Length = 010A
REMAPPED1 . . . . . .  L BYTE  0090  CSEG  Global
REMAPPED2 . . . . . .  L BYTE  0091  CSEG
REMAPPED3 . . . . . .  L BYTE  0092  CSEG
RET_BYTE . . . . . .  Number  000E
RH_BPB_PTR . . . . . .  L DWORD 04EF  CSEG
RH_BUFFER . . . . . .  L DWORD 04EB  CSEG
RH_COMMAND_CODE . . . . . .  L BYTE  04DF  CSEG
RH_LEN . . . . . .  L BYTE  04DD  CSEG
RH_MEDIA_DESC . . . . . .  L BYTE  04EA  CSEG
RH_OFF . . . . . .  L WORD  0085  CSEG
RH_RESV . . . . . .  L BYTE  04E2  CSEG  Length = 0008
RH_SEG . . . . . .  L WORD  0087  CSEG
RH_STATUS . . . . . .  L WORD  04E0  CSEG
RH_UNIT_CODE . . . . . .  L BYTE  04DE  CSEG
RTY_NUM . . . . . .  L WORD  001E  CSEG  Global

SAVE_DRIVE . . . . . .  L BYTE  00A9  CSEG
SECRET_BIAS . . . . . .  Number  1776
SETIOBSW . . . . . .  F PROC  0C9E  CSEG  Length = 001C
SET_CURS . . . . . .  L NEAR  0CE2  CSEG
SET_END_SPECIAL . . . . . .  N PROC  0CBA  CSEG  Length = 001A
SHORT_BAD_BPB . . . . . .  L NEAR  0A8B  CSEG
SINGLE_8_BPB . . . . . .  L 0020  046F  CSEG
SINGLE_9_BPB . . . . . .  L 0020  048F  CSEG
SPECIAL_DRIVE . . . . . .  L BYTE  0093  CSEG
SPKR_ENABLE . . . . . .  Number  0002
SPKR_PORT . . . . . .  Number  0061
SP_SAVE . . . . . .  V WORD  0000  CSEG  External
SRH . . . . . .  Number  0000
SRH_CCD_FLD . . . . . .  Number  0002
SRH_LEN . . . . . .  Number  000D
SRH_LEN_FLD . . . . . .  Alias  SRH
SRH_RES_FLD . . . . . .  Number  0005
```

```
SRH_STA_FLD . . . . . .  Number  0003
SRH_UCD_FLD . . . . . .  Number  0001
SSN . . . . . .  Number  0014
SSN_LEN . . . . . .  Number  0002
SS_SAVE . . . . . .  V WORD  0000  CSEG  External
STACK_TOP . . . . . .  L NEAR  0000  CSEG  External
STACK_USE . . . . . .  V BYTE  0000  CSEG  External
STRATEGY . . . . . .  L WORD  0006  CSEG

TEMP_BPB . . . . . .  L 0020  044F  CSEG
TIMEOUT . . . . . .  L WORD  0022  CSEG  Global
TIMER_CTRL . . . . . .  Number  0043
TIMER_TWO . . . . . .  Number  0042
TIMER_ZERO . . . . . .  Number  0040
TRACE . . . . . .  Number  0000
TRUE . . . . . .  Number  FFFF
TYPE_BPB . . . . . .  L WORD  00AB  CSEG

UNITS . . . . . .  Number  000D
UNITS_LEN . . . . . .  Number  0001
UNMOUNTCODE . . . . . .  Number  0002

V5_BPB_LOOP . . . . . .  L NEAR  0E45  CSEG
V5_JA . . . . . .  L NEAR  0ED9  CSEG
VDISK_INIT . . . . . .  N PROC  0E33  CSEG  Length = 00AE
VDISK_INIT_LABEL . . . . . .  L NEAR  0E33  CSEG
VDISK_IO . . . . . .  L NEAR  0000  CSEG  External
VDSK . . . . . .  F PROC  04F3  CSEG  Length = 00C0
VD_NOTBUSY . . . . . .  L NEAR  060C  CSEG
VERS3 . . . . . .  Number  0000
VERSION . . . . . .  L BYTE  0024  CSEG
VE_BAD_BLOCK . . . . . .  Number  0006
VE_BAD_DISK . . . . . .  Number  0008
VE_BAD_MACHINE . . . . . .  Number  0003
VE_CLIENT_ABORT . . . . . .  Number  000C
VE_ILLEGAL_OP . . . . . .  Number  0002
VE_INTERNAL . . . . . .  Number  000C
VE_LEVEL4 . . . . . .  Number  000A
VE_NO_DESCR_WRITE . . . . . .  Number  0007
VE_NO_DRIVE . . . . . .  Number  0001
VE_NO_READ . . . . . .  Number  0004
VE_NO_WRITE . . . . . .  Number  0005
VE_OK . . . . . .  Number  0000
VE_PROTOCOL . . . . . .  Number  000B
VE_RESTRICTION . . . . . .  Number  0009
VE_TIMEOUT . . . . . .  Number  000D
VIDEO_CALL . . . . . .  Number  0010
VP_VOL_UNIT . . . . . .  V BYTE  0000  CSEG  External
VSUB . . . . . .  N PROC  05B3  CSEG  Length = 0240

WDERR . . . . . .  L NEAR  0B5C  CSEG
WDIO . . . . . .  L NEAR  0AC0  CSEG
WDI1 . . . . . .  L NEAR  0BD1  CSEG
WDI1O . . . . . .  L NEAR  0C14  CSEG
```

```
WDI11  . . . . . .  L NEAR  0C14  CSEG
WDI12  . . . . . .  L NEAR  0BFA  CSEG
WDI2 . . . . . .  L NEAR  0AD0  CSEG
WDI3 . . . . . .  L NEAR  0AEA  CSEG
WDI4 . . . . . .  L NEAR  0B61  CSEG
WINDOW . . . . . .  L NEAR  07AE  CSEG
WINDOW_ERROR . . . . . .  L NEAR  07D7  CSEG
WRITESTATE . . . . . .  Number  0057
WRITE_DISK_INFO  . . . . . .  N PROC  0AAE  CSEG  Global  Length = 016A
WRITE_TTY  . . . . . .  Number  000E

YOUR_END . . . . . .  L WORD  0096  CSEG  Global

??000C . . . . . .  L NEAR  07C6  CSEG
??0015 . . . . . .  L NEAR  0924  CSEG
??0016 . . . . . .  L NEAR  092E  CSEG
??0017 . . . . . .  L NEAR  0A66  CSEG
??0018 . . . . . .  L NEAR  0AFC  CSEG
??0019 . . . . . .  L NEAR  0BD9  CSEG


   2414 Source  Lines
   2974 Total   Lines
    431 Symbols

  30488 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```