```
; ****************************************************************************
; DSECTPM.S (22/02/2015) - DISPLAY DISK SECTORS IN PROTECTED MODE
;                          Retro UNIX 386 v1 - DISK I/O test
; UNIX386.ASM (RETRO UNIX 386 Kernel) - v0.2.0.6 (21/02/2015)
; ----------------------------------------------------------------------------
; NASM version 2.11 (dsectpm.s, unix386.s)
;
; RETRO UNIX 386 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.2) by ERDOGAN TAN (Beginning: 24/12/2013)
;
; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
; (v0.1 - Beginning: 11/07/2012)
;
; [ Last Modification: 28/02/2015 ]
;
; Derived from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (1971-1972)
; <Bell Laboratories (17/3/1972)>
; <Preliminary Release of UNIX Implementation Document>
;
; Derived from 'UNIX v7/x86' source code by Robert Nordier (1999)
; UNIX V7/x86 source code: see www.nordier.com/v7x86 for details.
;
; ****************************************************************************

; 24/12/2013

; Entering protected mode:
; Derived from 'simple_asm.txt' source code file and
; 'The world of Protected mode' tutorial/article by Gregor Brunmar (2003)
; (gregor.brunmar@home.se)
; http://www.osdever.net/tutorials/view/the-world-of-protected-mode
;

; "The Real, Protected, Long mode assembly tutorial for PCs"
; by Michael Chourdakis (2009)
; http://www.codeproject.com/Articles/45788/
; http://www.michaelchourdakis.com
;

; Global Descriptor Table:
; Derived from 'head.s" source code of Linux v1.0 kernel
; by Linus Torvalds (1991-1992)
;

KLOAD   equ 10000h ; Kernel loading address
        ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h
KCODE   equ 08h ; Code segment descriptor (ring 0)
KDATA   equ 10h ; Data segment descriptor (ring 0)
;UCODE  equ 1Bh ; 18h + 3h  (ring 3)
;UDATA  equ 23h ; 20h + 3h  (ring 3)
; 27/12/2013
KEND    equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)

; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)
;--------- CMOS TABLE LOCATION ADDRESS'S -----------------------------------
CMOS_SECONDS    EQU     00H             ; SECONDS (BCD)
CMOS_MINUTES    EQU     02H             ; MINUTES (BCD)
CMOS_HOURS      EQU     04H             ; HOURS (BCD)
CMOS_DAY_WEEK   EQU     06H             ; DAY OF THE WEEK  (BCD)
CMOS_DAY_MONTH  EQU     07H             ; DAY OF THE MONTH (BCD)
CMOS_MONTH      EQU     08H             ; MONTH (BCD)
CMOS_YEAR       EQU     09H             ; YEAR (TWO DIGITS) (BCD)
CMOS_CENTURY    EQU     32H             ; DATE CENTURY BYTE (BCD)
```

```
CMOS_REG_A      EQU     0AH             ; STATUS REGISTER A
CMOS_REG_D      EQU     0DH             ; STATUS REGISTER D  BATTERY
;---------------------------------------
;       CMOS EQUATES FOR THIS SYSTEM    ;
;-----------------------------------------------------------------------------
CMOS_PORT       EQU     070H            ; I/O ADDRESS OF CMOS ADDRESS PORT
CMOS_DATA       EQU     071H            ; I/O ADDRESS OF CMOS DATA PORT
NMI             EQU     10000000B       ; DISABLE NMI INTERRUPTS MASK -
                                        ; HIGH BIT OF CMOS LOCATION ADDRESS

; Memory Allocation Table Address
; 05/11/2014
; 31/10/2014
MEM_ALLOC_TBL   equ     100000h         ; Memory Allocation Table at the end of
                                        ; the 1st 1 MB memory space.
                                        ; (This address must be aligned
                                        ;  on 128 KB boundary, if it will be
                                        ;  changed later.)
                                        ; ((lower 17 bits of 32 bit M.A.T.
                                        ;   address must be ZERO)).
                                        ; ((((Reason: 32 bit allocation
                                        ;     instructions, dword steps)))
                                        ; (((byte >> 12 --> page >> 5)))
;04/11/2014
PDE_A_PRESENT   equ     1               ; Present flag for PDE
PDE_A_WRITABLE  equ     2               ; Writable (write permission) flag
PDE_A_USER      equ     4               ; User (non-system/kernel) page flag
;
PTE_A_PRESENT   equ     1               ; Present flag for PTE
PTE_A_WRITABLE  equ     2               ; Writable (write permission) flag
PTE_A_USER      equ     4               ; User (non-system/kernel) page flag


; 17/02/2015 (unix386.s)
; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
DPT_SEGM equ 09000h  ; FDPT segment (EDD v1.1, EDD v3)
;
HD0_DPT  equ 0      ; Disk parameter table address for hd0
HD1_DPT  equ 32     ; Disk parameter table address for hd1
HD2_DPT  equ 64     ; Disk parameter table address for hd2
HD3_DPT  equ 96     ; Disk parameter table address for hd3


; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
;      (HDPT: Programmer's Guide to the AMIBIOS, 1993)
;
FDPT_CYLS       equ 0 ; 1 word, number of cylinders
FDPT_HDS        equ 2 ; 1 byte, number of heads
FDPT_TT         equ 3 ; 1 byte, A0h = translated FDPT with logical values
                      ; otherwise it is standard FDPT with physical values
FDPT_PCMP       equ 5 ; 1 word, starting write precompensation cylinder
                      ; (obsolete for IDE/ATA drives)
FDPT_CB         equ 8 ; 1 byte, drive control byte
                        ; Bits 7-6 : Enable or disable retries (00h = enable)
                        ; Bit 5 : 1 = Defect map is located at last cyl. + 1
                        ; Bit 4 : Reserved. Always 0
                        ; Bit 3 : Set to 1 if more than 8 heads
                        ; Bit 2-0 : Reserved. Alsways 0
FDPT_LZ         equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
FDPT_SPT        equ 14 ; 1 byte, sectors per track

; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
; (11 bytes long) will be used by diskette handler/bios
; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).
```

```
[BITS 16]       ; We need 16-bit intructions for Real mode

[ORG 0]
        ; 12/11/2014
        ; Save boot drive number (that is default root drive)
        mov     [boot_drv], dl ; physical drv number

        ; Determine installed memory
        ; 31/10/2014
        ;
        mov     ax, 0E801h ; Get memory size
        int     15h        ; for large configurations
        jnc     short chk_ms
        mov     ah, 88h    ; Get extended memory size
        int     15h
        ;
        ;mov    al, 17h ; Extended memory (1K blocks) low byte
        ;out    70h, al ; select CMOS register
        ;in     al, 71h ; read data (1 byte)
        ;mov    cl, al
        ;mov    al, 18h ; Extended memory (1K blocks) high byte
        ;out    70h, al ; select CMOS register
        ;in     al, 71h ; read data (1 byte)
        ;mov    ch, al
        ;
        mov     cx, ax
        xor     dx, dx
chk_ms:
        mov     [mem_1m_1k], cx
        mov     [mem_16m_64k], dx
        ; 05/11/2014
        ;and    dx, dx
        ;jz     short L2
        cmp     cx, 1024
        jnb     short L0
                 ; insufficient memory_error
                 ; Minimum 2 MB memory is needed...
        ; 05/11/2014
        ; (real mode error printing)
        sti
        mov     si, msg_out_of_memory
        mov     bx, 7
        mov     ah, 0Eh ; write tty
oom_1:
        lodsb
        or      al, al
        jz      short oom_2
        int     10h
        jmp     short oom_1
oom_2:
        hlt
        jmp     short oom_2

; 05/11/2014
msg_out_of_memory:
        db      07h, 0Dh, 0Ah
        db      'Insufficient memory ! (Minimum 2 MB memory is needed.)'
        db      0Dh, 0Ah, 0
        ;
; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
```

```
; ///////// DISK I/O SYSTEM STRUCTURE INITIALIZATION ///////////////

        ; 10/12/2014 - 02/02/2015 - dsectrm2.s
L0:
        ; 22/02/2015 - dsectpm.s
        mov     si, prg_msg
        mov     bx, 7
        mov     ah, 0eh
startmsgl:
        lodsb
        and     al, al
        jz      short startmsg_ok
        int     10h
        jmp     short startmsgl
startmsg_ok:
        ;
        ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
        ; Detecting disk drives... (by help of ROM-BIOS)
        mov     dx, 7Fh
L1:
        inc     dl
        mov     ah, 41h ; Check extensions present
                        ; Phoenix EDD v1.1 - EDD v3
        mov     bx, 55AAh
        int     13h
        jc      short L2
        cmp     bx, 0AA55h
        jne     short L2
        inc     byte [hdc]      ; count of hard disks (EDD present)
        mov     [last_drv], dl  ; last hard disk number
        mov     bx, hd0_type - 80h
        add     bx, dx
        mov     [bx], cl ; Interface support bit map in CX
                         ; Bit 0 - 1, Fixed disk access subset ready
                         ; Bit 1 - 1, Drv locking and ejecting ready
                         ; Bit 2 - 1, Enhanced Disk Drive Support
                         ;            (EDD) ready (DPTE ready)
                         ; Bit 3 - 1, 64bit extensions are present
                         ;            (EDD-3)
                         ; Bit 4 to 15 - 0, Reserved
        cmp     dl, 83h  ; drive number < 83h
        jb      short L1
L2:
        ; 23/11/2014
        ; 19/11/2014
        xor     dl, dl  ; 0
        mov     esi, fd0_type
L3:
        ; 14/01/2015
        mov     [drv], dl
        ;
        mov     ah, 08h ; Return drive parameters
        int     13h
        jc      short L4
                ; BL = drive type (for floppy drives)
                ; DL = number of floppy drives
                ;
                ; ES:DI = Address of DPT from BIOS
                ;
        mov     [esi], bl ;  Drive type
                          ; 4 = 1.44 MB, 80 track, 3 1/2"
        ; 14/01/2015
        call    set_disk_parms
```

```
        ; 10/12/2014
        cmp     esi, fd0_type
        ja      short L4
        inc     esi ; fd1_type
        mov     dl, 1
        jmp     short L3
L4:
        ; Older BIOS (INT 13h, AH = 48h is not available)
        mov     dl, 7Fh
        ; 24/12/2014 (Temporary)
        cmp     byte [hdc], 0 ; EDD present or not ?
        ja      L10       ; yes, all fixed disk operations
                          ; will be performed according to
                          ; present EDD specification
L6:
        inc     dl
        mov     [drv], dl
        mov     [last_drv], dl ; 14/01/2015
        mov     ah, 08h ; Return drive parameters
        int     13h     ; (conventional function)
        jc      L13     ; fixed disk drive not ready
        mov     [hdc], dl ; number of drives
        ;; 14/01/2013
        ;;push  cx
        call    set_disk_parms
        ;;pop   cx
        ;
        ;;and   cl, 3Fh  ; sectors per track (bits 0-6)
        mov     dl, [drv]
        mov     bx, 65*4 ; hd0 parameters table (INT 41h)
        cmp     dl, 80h
        jna     short L7
        add     bx, 5*4  ; hd1 parameters table (INT 46h)
L7:
        xor     ax, ax
        mov     ds, ax
        mov     si, [bx]
        mov     ax, [bx+2]
        mov     ds, ax
        cmp     cl, [si+FDPT_SPT] ; sectors per track
        jne     L12 ; invalid FDPT
        mov     di, HD0_DPT
        cmp     dl, 80h
        jna     short L8
        mov     di, HD1_DPT
L8:
        ; 30/12/2014
        mov     ax, DPT_SEGM
        mov     es, ax
        ; 24/12/2014
        mov     cx, 8
        rep     movsw  ; copy 16 bytes to the kernel's DPT location
        mov     ax, cs
        mov     ds, ax
        ; 02/02/2015
        mov     cl, [drv]
        mov     bl, cl
        mov     ax, 1F0h
        and     bl, 1
        jz      short L9
        shl     bl, 4
        sub     ax, 1F0h-170h
L9:
        stosw   ; I/O PORT Base Address (1F0h, 170h)
```

```
        add     ax, 206h
        stosw   ; CONTROL PORT Address (3F6h, 376h)
        mov     al, bl
        add     al, 0A0h
        stosb   ; Device/Head Register upper nibble
        ;
        inc     byte [drv]
        mov     bx, hd0_type - 80h
        add     bx, cx
        or      byte [bx], 80h  ; present sign (when lower nibble is 0)
        mov     al, [hdc]
        dec     al
        jz      L13
        cmp     dl, 80h
        jna     L6
        jmp     L13
L10:
        inc     dl
        ; 25/12/2014
        mov     [drv], dl
        mov     ah, 08h ; Return drive parameters
        int     13h     ; (conventional function)
        jc      L13
        ; 14/01/2015
        mov     dl, [drv]
        push    dx
        push    cx
        call    set_disk_parms
        pop     cx
        pop     dx
        ;
        mov     esi, _end ; 30 byte temporary buffer address
                          ; at the '_end' of kernel.
        mov     word [esi], 30
        mov     ah, 48h  ; Get drive parameters (EDD function)
        int     13h
        jc      L13
        ; 14/01/2015
        sub     ebx, ebx
        mov     bl, dl
        sub     bl, 80h
        add     ebx, hd0_type
        mov     al, [ebx]
        or      al, 80h
        mov     [ebx], al
        sub     ebx, hd0_type - 2 ; 15/01/2015
        add     ebx, drv_status
        mov     [ebx], al
        mov     eax, [esi+16]
        ; 28/02/2015
        and     eax, eax
        jz      short L10_A0h
                        ; 'CHS only' disks on EDD system
                        ;  are reported with ZERO disk size
        sub     ebx, drv_status
        shl     ebx, 2
        add     ebx, disk_size
        mov     [ebx], eax
L10_A0h: ; Jump here to fix a ZERO (LBA) disk size problem
         ; for CHS disks (28/02/2015)
        ; 30/12/2014
        mov     di, HD0_DPT
        mov     al, dl
        and     ax, 3
```

```
        shl     al, 5 ; *32
        add     di, ax
        mov     ax, DPT_SEGM
        mov     es, ax
        ;
        mov     al, ch  ; max. cylinder number (bits 0-7)
        mov     ah, cl
        shr     ah, 6   ; max. cylinder number (bits 8-9)
        inc     ax      ; logical cylinders (limit 1024)
        stosw
        mov     al, dh  ; max. head number
        inc     al
        stosb           ; logical heads (limits 256)
        mov     al, 0A0h ; Indicates translated table
        stosb
        mov     al, [si+12]
        stosb           ; physical sectors per track
        xor     ax, ax
        ;dec    ax      ; 02/01/2015
        stosw           ; precompensation (obsolete)
        ;xor    al, al  ; 02/01/2015
        stosb           ; reserved
        mov     al, 8   ; drive control byte
                        ; (do not disable retries,
                        ; more than 8 heads)
        stosb
        mov     ax, [si+4]
        stosw           ; physical number of cylinders
        ;push   ax      ; 02/01/2015
        mov     al, [si+8]
        stosb           ; physical num. of heads (limit 16)
        sub     ax, ax
        ;pop    ax      ; 02/01/2015
        stosw           ; landing zone (obsolete)
        mov     al, cl  ; logical sectors per track (limit 63)
        and     al, 3Fh
        stosb
        ;sub    al, al  ; checksum
        ;stosb
        ;
        add     si, 26  ; (BIOS) DPTE address pointer
        lodsw
        push    ax      ; (BIOS) DPTE offset
        lodsw
        push    ax      ; (BIOS) DPTE segment
        ;
        ; checksum calculation
        mov     si, di
        push    es
        pop     ds
        ;mov    cx, 16
        mov     cx, 15
        sub     si, cx
        xor     ah, ah
        ;del    cl
L11:
        lodsb
        add     ah, al
        loop    L11
        ;
        mov     al, ah
        neg     al      ; -x+x = 0
        stosb           ; put checksum in byte 15 of the tbl
        ;
```

```
        pop     ds      ; (BIOS) DPTE segment
        pop     si      ; (BIOS) DPTE offset
        ;
        ; 23/02/2015
        push    di
        ; ES:DI points to DPTE (FDPTE) location
        ;mov    cx, 8
        mov     cl, 8
        rep     movsw
        ;
        ; 23/02/2015
        ; (P)ATA drive and LBA validation
        ; (invalidating SATA drives and setting
        ; CHS type I/O for old type fixed disks)
        pop     bx
        mov     ax, cs
        mov     ds, ax
        mov     ax, [es:bx]
        cmp     ax, 1F0h
        je      short L11a
        cmp     ax, 170h
        je      short L11a
        ; invalidation
        ; (because base port addres is not 1F0h or 170h)
        xor     bh, bh
        mov     bl, dl
        sub     bl, 80h
        mov     byte [bx+hd0_type], 0 ; not a valid disk drive !
        ;or     [bx+drv_status+2], 0F0h ; (failure sign)
        jmp     short L11b
L11a:
        ; LBA validation
        mov     al, [es:bx+4] ; Head register upper nibble
        test    al, 40h ; LBA bit (bit 6)
        jnz     short L11b ; LBA type I/O is OK! (E0h or F0h)
        ; force CHS type I/O for this drive (A0h or B0h)
        sub     bh, bh
        mov     bl, dl
        sub     bl, 80h ; 26/02/2015
        and     byte [bx+drv_status+2], 0FEh ; clear bit 0
                                ; bit 0 = LBA ready bit
        ; 'read_disk_sector' procedure will check this bit !
L11b:
        cmp     dl, [last_drv] ; 25/12/2014
        jnb     short L13
        jmp     L10
L12:
        ; Restore data registers
        mov     ax, cs
        mov     ds, ax
L13:
        ; 13/12/2014
        push    cs
        pop     es
L14:
        mov     ah, 11h
        int     16h
        jz      short L15
        mov     al, 10h
        int     16h
        jmp     short L14
L15:
; //////
        ; 24/11/2014
```

```
        ; 19/11/2014
        ; 14/11/2014
        ; Temporary code for disk searching code check
        ;
        ; This code will show existing (usable) drives and also
        ; will show EDD interface support status for hard disks
        ; (If status bit 7 is 1, Identify Device info is ready,
        ; no need to get it again in protected mode...)
        ;
        ; 13/11/2014
        mov     bx, 7
        mov     ah, 0Eh
        mov     al, [fd0_type]
        and     al, al
        jz      short L15a
        mov     dl, al
        mov     al, 'F'
        int     10h
        mov     al, 'D'
        int     10h
        mov     al, '0'
        int     10h
        mov     al, ' '
        int     10h
        call    L15c
        mov     al, ' '
        int     10h
        ;
        mov     al, [fd1_type]
        and     al, al
        jz      short L15a
        mov     dl, al
        mov     al, 'F'
        int     10h
        mov     al, 'D'
        int     10h
        mov     al, '1'
        int     10h
        mov     al, ' '
        int     10h
        call    L15c
        mov     al, ' '
        int     10h
        mov     al, ' '
        int     10h
L15a:
        mov     al, [hd0_type]
        and     al, al
        jz      short L15b
        mov     dl, al
        mov     al, 'H'
        int     10h
        mov     al, 'D'
        int     10h
        mov     al, '0'
        int     10h
        mov     al, ' '
        int     10h
        call    L15c
        mov     al, ' '
        int     10h
        ;
        mov     al, [hd1_type]
        and     al, al
```

```
        jz      short L15b
        mov     dl, al
        mov     al, 'H'
        int     10h
        mov     al, 'D'
        int     10h
        mov     al, '1'
        int     10h
        mov     al, ' '
        int     10h
        call    L15c
        mov     al, ' '
        int     10h
        ;
        mov     al, [hd2_type]
        and     al, al
        jz      short L15b
        mov     dl, al
        mov     al, 'H'
        int     10h
        mov     al, 'D'
        int     10h
        mov     al, '2'
        int     10h
        mov     al, ' '
        int     10h
        call    L15c
        mov     al, ' '
        int     10h
        ;
        mov     al, [hd3_type]
        and     al, al
        jz      short L15b
        mov     dl, al
        mov     al, 'H'
        int     10h
        mov     al, 'D'
        int     10h
        mov     al, '3'
        int     10h
        mov     al, ' '
        int     10h
        call    L15c
        mov     al, ' '
        int     10h
        ;
L15b:
        mov     al, 0Dh
        int     10h
        mov     al, 0Ah
        int     10h
        ;
        xor     ah, ah
        int     16h
        ;
        ;jmp     short L16
        jmp     L16
        ;
L15c:
        mov     dh, dl
        shr     dh, 4
        add     dh, 30h
        and     dl, 15
        add     dl, 30h
```

```
        mov     al, dh
        int     10h
        mov     al, dl
        int     10h
        retn
        ;
        ; end of temporary code for disk searching code check

; //////

set_disk_parms:
        ; 14/01/2015
        ;push   ebx
        sub     ebx, ebx
        mov     bl, [drv]
        cmp     bl, 80h
        jb      short sdp0
        sub     bl, 7Eh
sdp0:
        add     ebx, drv_status
        mov     byte [ebx], 80h ; 'Present' flag
        ;
        mov     al, ch ; last cylinder (bits 0-7)
        mov     ah, cl ;
        shr     ah, 6  ; last cylinder (bits 8-9)
        sub     ebx, drv_status
        shl     bl, 1
        add     ebx, cylinders
        inc     ax  ; convert max. cyl number to cyl count
        mov     [ebx], ax
        push    ax ; ** cylinders
        sub     ebx, cylinders
        add     ebx, heads
        xor     ah, ah
        mov     al, dh ; heads
        inc     ax
        mov     [ebx], ax
        sub     ebx, heads
        add     ebx, spt
        xor     ch, ch
        and     cl, 3Fh ; sectors (bits 0-6)
        mov     [ebx], cx
        sub     ebx, spt
        shl     ebx, 1
        add     ebx, disk_size
        ; LBA size = cylinders * heads * secpertrack
        mul     cx
        mov     dx, ax  ; heads*spt
        pop     ax ; ** cylinders
        dec     ax ; 1 cylinder reserved (!?)
        mul     dx ; cylinders * (heads*spt)
        mov     [ebx], ax
        mov     [ebx+2], dx
        ;
        ;pop    ebx
        retn

; End Of DISK I/O SYSTEM STRUCTURE INITIALIZATION /// 06/02/2015

L16:
_L:     ; 12/02/2015 (unix386.s)
        ; ------------------------------------------------
        ; 28/11/2014 (dsectrm2.s)
        xor     bh, bh
```

```
        mov     ah, 03h ; get cursor position and shape
        int     10h
        ;mov    [cursor_posb], dx ; position ;; 16/02/2015
        mov     [cursor_shp], cx ; shape
        ; ------------------------------------------------
        ;
        ; 10/11/2014
        cli     ; Disable interrupts (clear interrupt flag)
                ; Reset Interrupt MASK Registers (Master&Slave)
        ;mov    al, 0FFh        ; mask off all interrupts
        ;out    21h, al         ; on master PIC (8259)
        ; jmp   $+2  ; (delay)
        ;out    0A1h, al        ; on slave PIC (8259)
        ;
        ; Disable NMI
        mov     al, 80h
        out     70h, al         ; set bit 7 to 1 for disabling NMI
         ;23/02/2015
        nop
        in      al, 71h         ; read in 71h just after writing out to 70h
                                ; for preventing unknown state (!?)
        ; 20/08/2014
        ; Moving the kernel 64 KB back (to physical address 0)
        ; DS = CS = 1000h
        ; 05/11/2014
        xor     ax, ax
        mov     es, ax ; ES = 0
        ;
        mov     cx, (KEND - KLOAD)/4
        xor     si, si
        xor     di, di
        rep     movsd
        ;
        push    es ; 0
        push    L17
        retf
        ;
L17:
        ; Turn off the floppy drive motor
        mov     dx, 3F2h
        out     dx, al ; 0 ; 31/12/2013

        ; Enable access to memory above one megabyte
L18:
        in      al, 64h
        test    al, 2
        jnz     short L18
        mov     al, 0D1h        ; Write output port
        out     64h, al
L19:
        in      al, 64h
        test    al, 2
        jnz     short L19
        mov     al, 0DFh        ; Enable A20 line
        out     60h, al
;L20:
        ;
        ; Load global descriptor table register

        ;mov     ax, cs
        ;mov     ds, ax

        lgdt    [cs:gdtd]
```

```
        mov     eax, cr0
        ; or    eax, 1
        inc     ax
        mov     cr0, eax

        ; Jump to 32 bit code

        db 66h                  ; Prefix for 32-bit
        db 0EAh                 ; Opcode for far jump
        dd StartPM              ; Offset to start, 32-bit
                                ; (1000h:StartPM = StartPM + 10000h)
        dw KCODE                ; This is the selector for CODE32_DESCRIPTOR,
                                ; assuming that StartPM resides in code32

[BITS 32]

StartPM:
        ; Kernel Base Address = 0 (Temporary) ; 30/12/2013
        mov ax, KDATA           ; Save data segment identifier
        mov ds, ax              ; Move a valid data segment into DS register
        mov es, ax              ; Move data segment into ES register
        mov fs, ax              ; Move data segment into FS register
        mov gs, ax              ; Move data segment into GS register
        mov ss, ax              ; Move data segment into SS register
        mov esp, 90000h         ; Move the stack pointer to 090000h

memory_init:
        ; Initialize memory allocation table and page tables
        ; 16/11/2014
        ; 15/11/2014
        ; 07/11/2014
        ; 06/11/2014
        ; 05/11/2014
        ; 04/11/2014
        ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
        ;
        xor     eax, eax
        xor     ecx, ecx
        mov     cl, 8
        mov     edi, MEM_ALLOC_TBL
        rep     stosd               ; clear Memory Allocation Table
                                    ; for the first 1 MB memory
        ;
        mov     cx, [mem_1m_1k]     ; Number of contiguous KB between
                                    ; 1 and 16 MB, max. 3C00h = 15 MB.
        shr     cx, 2               ; convert 1 KB count to 4 KB count
        mov     [free_pages], ecx
        mov     dx, [mem_16m_64k]   ; Number of contiguous 64 KB blocks
                                    ; between 16 MB and 4 GB.
        or      dx, dx
        jz      short mi_0
        ;
        mov     ax, dx
        shl     eax, 4              ; 64 KB -> 4 KB (page count)
        add     [free_pages], eax
        add     eax, 4096           ; 16 MB = 4096 pages
        jmp     short mi_1
mi_0:
        mov     ax, cx
        add     ax, 256             ; add 256 pages for the first 1 MB

mi_1:
        mov     [memory_size], eax ; Total available memory in pages
                                    ; 1 alloc. tbl. bit = 1 memory page
```

```
                                ; 32 allocation bits = 32 mem. pages
        ;
        add     eax, 32767      ; 32768 memory pages per 1 M.A.T. page
        shr     eax, 15         ; ((32768 * x) + y) pages (y < 32768)
                                ;  --> x + 1 M.A.T. pages, if y > 0
                                ;  --> x M.A.T. pages, if y = 0
        mov     [mat_size], ax  ; Memory Alloc. Table Size in pages
        shl     eax, 12         ; 1 M.A.T. page = 4096 bytes
        ;                       ; Max. 32 M.A.T. pages (4 GB memory)
        mov     ebx, eax        ; M.A.T. size in bytes
        ; Set/Calculate Kernel's Page Directory Address
        add     ebx, MEM_ALLOC_TBL
        mov     [k_page_dir], ebx  ; Kernel's Page Directory address
                                ; just after the last M.A.T. page
        ;
        sub     eax, 4          ; convert M.A.T. size to offset value
        mov     [last_page], eax   ; last page ofset in the M.A.T.
        ;                       ; (allocation status search must be
                                ; stopped after here)
        xor     eax, eax
        dec     eax             ; FFFFFFFFh (set all bits to 1)
        push    cx
        shr     ecx, 5          ; convert 1 - 16 MB page count to
                                ; count of 32 allocation bits
        rep     stosd
        pop     cx
        inc     eax             ; 0
        and     cl, 31          ; remain bits
        jz      short mi_4
        mov     [edi], eax      ; reset
mi_2:
        bts     [edi], eax      ; 06/11/2014
        dec     cl
        jz      short mi_3
        inc     al
        jmp     short mi_2
mi_3:
        sub     al, al          ; 0
        add     edi, 4          ; 15/11/2014
mi_4:
        or      dx, dx          ; check 16M to 4G memory space
        jz      short mi_6      ; max. 16 MB memory, no more...
        ;
        mov     ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
        ;
        sub     ecx, edi        ; displacement (to end of 16 MB)
        jz      short mi_5      ; jump if EDI points to
                                ;         end of first 16 MB
        shr     ecx, 1          ; convert to dword count
        shr     ecx, 1          ; (shift 2 bits right)
        rep     stosd           ; reset all bits for reserved pages
                                ; (memory hole under 16 MB)
mi_5:
        mov     cx, dx          ; count of 64 KB memory blocks
        shr     ecx, 1          ; 1 alloc. dword per 128 KB memory
        pushf                   ; 16/11/2014
        dec     eax             ; FFFFFFFFh (set all bits to 1)
        rep     stosd
        inc     eax             ; 0
        popf                    ; 16/11/2014
        jnc     short mi_6
        dec     ax              ; eax = 0000FFFFh
        stosd
        inc     ax              ; 0
```

```
mi_6:
        cmp     edi, ebx          ; check if EDI points to
        jnb     short mi_7        ; end of memory allocation table
        ;                         ; (>= MEM_ALLOC_TBL + 4906)
        mov     ecx, ebx          ; end of memory allocation table
        sub     ecx, edi          ; convert displacement/offset
        shr     ecx, 1            ; to dword count
        shr     ecx, 1            ; (shift 2 bits right)
        rep     stosd             ; reset all remain M.A.T. bits
mi_7:
        ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)
        mov     edx, MEM_ALLOC_TBL
        ;sub    ebx, edx          ; Mem. Alloc. Tbl. size in bytes
        ;shr    ebx, 12           ; Mem. Alloc. Tbl. size in pages
        mov     cx, [mat_size]    ; Mem. Alloc. Tbl. size in pages
        mov     edi, edx
        shr     edi, 15           ; convert M.A.T. address to
                                  ; byte offset in M.A.T.
                                  ; (1 M.A.T. byte points to
                                  ;           32768 bytes)
                                  ; Note: MEM_ALLOC_TBL address
                                  ; must be aligned on 128 KB
                                  ; boundary!
        add     edi, edx          ; points to M.A.T.'s itself
        ; eax = 0
        sub     [free_pages], ecx ; 07/11/2014
mi_8:
        btr     [edi], eax        ; clear bit 0 to bit x (1 to 31)
        ;dec    bl
        dec     cl
        jz      short mi_9
        inc     al
        jmp     short mi_8
mi_9:
        ;
        ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
        ;               (allocate pages for system page tables)

        ; edx = MEM_ALLOC_TBL
        mov     ecx, [memory_size] ; memory size in pages (PTEs)
        add     ecx, 1023         ; round up (1024 PTEs per table)
        shr     ecx, 10           ; convert memory page count to
                                  ; page table count (PDE count)
        ;
        push    ecx               ; (**) PDE count (<= 1024)
        ;
        inc     ecx               ; +1 for kernel page directory
        ;
        sub     [free_pages], ecx ; 07/11/2014
        ;
        mov     esi, [k_page_dir] ; Kernel's Page Directory address
        shr     esi, 12           ; convert to page number
mi_10:
        mov     eax, esi          ; allocation bit offset
        mov     ebx, eax
        shr     ebx, 3            ; convert to alloc. byte offset
        and     bl,  0FCh         ; clear bit 0 and bit 1
                                  ;   to align on dword boundary
        and     eax, 31           ; set allocation bit position
                                  ;  (bit 0 to bit 31)
        ;
        add     ebx, edx          ; offset in M.A.T. + M.A.T. address
        ;
        btr     [ebx], eax        ; reset relevant bit (0 to 31)
```

```
        ;
        inc     esi             ; next page table
        loop    mi_10           ; allocate next kernel page table
                                ; (ecx = page table count + 1)
        ;
        pop     ecx             ; (**) PDE count (= pg. tbl. count)
        ;
        ; Initialize Kernel Page Directory and Kernel Page Tables
        ;
        ; Initialize Kernel's Page Directory
        mov     edi, [k_page_dir]
        mov     eax, edi
        or      al, PDE_A_PRESENT + PDE_A_WRITABLE
                            ; supervisor + read&write + present
        mov     edx, ecx        ; (**) PDE count (= pg. tbl. count)
mi_11:
        add     eax, 4096       ; Add page size (PGSZ)
                                ; EAX points to next page table
        stosd
        loop    mi_11
        sub     eax, eax        ; Empty PDE
        mov     cx, 1024        ; Entry count (PGSZ/4)
        sub     ecx, edx
        jz      short mi_12
        rep     stosd           ; clear remain (empty) PDEs
        ;
        ; Initialization of Kernel's Page Directory is OK, here.
mi_12:
        ; Initialize Kernel's Page Tables
        ;
        ; (EDI points to address of page table 0)
        ; eax = 0
        mov     ecx, [memory_size] ; memory size in pages
        mov     edx, ecx        ; (***)
        mov     al, PTE_A_PRESENT + PTE_A_WRITABLE
                           ; supervisor + r214ead&write + present
mi_13:
        stosd
        add     eax, 4096
        loop    mi_13
        and     dx, 1023        ; (***)
        jz      short mi_14
        mov     cx, 1024
        sub     cx, dx          ; from dx (<= 1023) to 1024
        xor     eax, eax
        rep     stosd           ; clear remain (empty) PTEs
                                ; of the last page table
mi_14:
        ;  Initialization of Kernel's Page Tables is OK, here.
        ;
        mov     eax, edi        ; end of the last page table page
                                ; (beginging of user space pages)
        shr     eax, 15         ; convert to M.A.T. byte offset
        and     al, 0FCh        ; clear bit 0 and bit 1 for
                                ; aligning on dword boundary

        mov     [first_page], eax
        mov     [next_page], eax ; The first free page pointer
                                 ; for user programs
                                 ; (Offset in Mem. Alloc. Tbl.)
        ;
        ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
        ;
```

```
        ; Enable paging
        ;
        mov     eax, [k_page_dir]
        mov     cr3, eax
        mov     eax, cr0
        or      eax, 80000000h  ; set paging bit (bit 31)
        mov     cr0, eax
        ;jmp    KCODE:StartPMP

        db 0EAh                 ; Opcode for far jump
        dd StartPMP             ; 32 bit offset
        dw KCODE                ; kernel code segment descriptor


StartPMP:
        ; 06/11//2014
        ; Clear video page 0
        ;
        ; Temporary Code
        ;
        mov     ecx, 80*25/2
        mov     edi, 0B8000h
        xor     eax, eax        ; black background, black fore color
        rep     stosd

        ; 19/08/2014
        ; Kernel Base Address = 0
        ; It is mapped to (physically) 0 in the page table.
        ; So, here is exactly 'StartPMP' address.
        ;
        mov     ah, 4Eh ; Red background, yellow forecolor
        mov     esi, msgPM
        mov     edi, 0B8000h ; 27/08/2014
        ; 20/08/2014
        call    printk

        ; 'UNIX v7/x86' source code by Robert Nordier (1999)
        ; // Set IRQ offsets
        ;
        ;  Linux (v0.12) source code by Linus Torvalds (1991)
        ;
                                        ;; ICW1
        mov     al, 11h                 ; Initialization sequence
        out     20h, al                 ;       8259A-1
        ; jmp   $+2
        out     0A0h, al                ;       8259A-2
                                        ;; ICW2
        mov     al, 20h                 ; Start of hardware ints (20h)
        out     21h, al                 ;       for 8259A-1
        ; jmp   $+2
        mov     al, 28h                 ; Start of hardware ints (28h)
        out     0A1h, al                ;       for 8259A-2
                                        ;
        mov     al, 04h                 ;; ICW3
        out     21h, al                 ;       IRQ2 of 8259A-1 (master)
        ; jmp   $+2
        mov     al, 02h                 ;       is 8259A-2 (slave)
        out     0A1h, al                ;
                                        ;; ICW4
        mov     al, 01h                 ;
        out     21h, al                 ;       8086 mode, normal EOI
        ; jmp   $+2
        out     0A1h, al                ;       for both chips.
```

```
        ;mov    al, 0FFh        ; mask off all interrupts for now
        ;out    21h, al
        ;; jmp  $+2
        ;out    0A1h, al

        ;
        ;; Linux (v0.12) source code by Linus Torvalds (1991)
        ;  setup_idt:
        ;
        ;; 16/02/2015
        ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
        ; 21/08/2014 (timer_int)
        mov     esi, ilist
        lea     edi, [idt]
        mov     ecx, 64         ; 64 interrupts (INT 0 to INT 3Fh)
rp_sidt1:
        lodsd
        or      eax, eax
        jz      short rp_sidt2
        mov     edx, eax
        mov     ebx, 80000h
        mov     bx, dx          ; /* selector = 0x0008 = cs */
        mov     dx, 8E00h       ; /* interrupt gate - dpl=0, present */
rp_sidt2:
        mov     [edi], ebx
        mov     [edi+4], edx
        add     edi, 8
        dec     ecx
        jz      short sidt_OK
        and     eax, eax
        jz      short rp_sidt2
        jmp     short rp_sidt1
sidt_OK:
        lidt    [idtd]
        ;
        ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
        ;
        ;cli    ; Disable interrupts
        ;    (CPU will not handle hardware interrupts, except NMI!)
        ;
        xor     al, al          ; Enable all hardware interrupts!
        out     21h, al         ; (IBM PC-AT compatibility)
        jmp     $+2             ; (All conventional PC-AT hardware
        out     0A1h, al        ;  interrupts will be in use.)
                                ; (Even if related hardware component
                                ;  does not exist!)
        ; Enable NMI
        mov     al, 7Fh         ; Clear bit 7 to enable NMI (again)
        out     70h, al
        ; 23/02/2015
        nop
        in      al, 71h         ; read in 71h just after writing out to 70h
                                ; for preventing unknown state (!?)
        ;
        ; Only a NMI can occur here... (Before a 'STI' instruction)
        ;
        ; 02/09/2014
        xor     bx, bx
        mov     dx, 0300h       ; Row 3, column 0
        call    set_cpos
        ;
        ; 06/11/2014
        ; Temporary code
        ;
```

```
        call    memory_info
        ;
        call    getch   ; 28/02/2015
drv_init:
        sti     ; Enable Interrupts
        ; 28/02/2015
        mov     ax, [cursor_posn] ; cursor pos. for video page 0
        mov     [cursor_posb], ax ; cursor position backup
        ;
        ; 06/02/2015
        mov     edx, [hd0_type] ; hd0, hd1, hd2, hd3
        mov     bx, [fd0_type] ; fd0, fd1
        ; 22/02/2015
        and     bx, bx
        jnz     short di1
        ;
        or      edx, edx
        jnz     short di2
        ;
setup_error:
        mov     esi, setup_error_msg
psem:
        lodsb
        or      al, al
        ;jz     short hang ; 22/02/2015
        jz      short di3
        push    esi
        xor     ebx, ebx ; 0
                         ; Video page 0 (bl=0)
        mov     ah, 07h ; Black background,
                         ; light gray forecolor
        call    write_tty
        pop     esi
        jmp     short psem

setup_error_msg:
        db 0Dh, 0Ah
        db 'Disk Setup Error!'
        db 0Dh, 0Ah,0
di1:
        ; supress 'jmp short T6'
        ;  (activate fdc motor control code)
        mov     word [T5], 9090h ; nop
        ;
        ;mov    ax, int_0Eh     ; IRQ 6 handler
        ;mov    di, 0Eh*4       ; IRQ 6 vector
        ;stosw
        ;mov    ax, cs
        ;stosw
        ;; 16/02/2015
        ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
        ;
        CALL    DSKETTE_SETUP   ; Initialize Floppy Disks
        ;
        or      edx, edx
        jz      short di3
di2:
        call    DISK_SETUP      ; Initialize Fixed Disks
        jc      short setup_error
di3:
        call    setup_rtc_int   ; 22/05/2015 (dsectrpm.s)
        ;
        ;call   dsectpm  ; 06/02/2015 - 21/02/2015
        call    display_sectors ; 22/02/2015
```

```
hang:
        call    getch ; 22/02/2015
        ;
        jmp     cpu_reset ; 22/02/2015

        ; 27/08/2014
        ; 20/08/2014
printk:
        ;mov    edi, dword [scr_row]
pkl:
        lodsb
        or      al, al
        jz      short pkr
        stosw
        jmp     short pkl
pkr:
        retn

        ; 21/08/2014
ilist:
        ;times  32 dd cpu_except ; INT 0 to INT 1Fh
        ;
        ; Exception list
        ; 25/08/2014
        dd      exc0    ; 0h,  Divide-by-zero Error
        dd      exc1
        dd      exc2
        dd      exc3
        dd      exc4
        dd      exc5
        dd      exc6    ; 06h,  Invalid Opcode
        dd      exc7
        dd      exc8
        dd      exc9
        dd      exc10
        dd      exc11
        dd      exc12
        dd      exc13   ; 0Dh, General Protection Fault
        dd      exc14   ; 0Eh, Page Fault
        dd      exc15
        dd      exc16
        dd      exc17
        dd      exc18
        dd      exc19
        dd      exc20
        dd      exc21
        dd      exc22
        dd      exc23
        dd      exc24
        dd      exc25
        dd      exc26
        dd      exc27
        dd      exc28
        dd      exc29
        dd      exc30
        dd      exc31
        ; Interrupt list
        dd      timer_int       ; INT 20h
                ;dd     irq0
        ;dd     keyb_int        ; 27/08/2014
        dd      kb_int          ; 22/02/2015 - dsectpm.s
                ;dd     irq1
        dd      irq2
        dd      irq3
```

```
        dd      irq4
        dd      irq5
;DISKETTE_INT: ;06/02/2015
        dd      fdc_int         ; 16/02/2015, IRQ 6 handler
                ;dd     irq6
; Default IRQ 7 handler against spurious IRQs (from master PIC)
; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
        dd      default_irq7    ; 25/02/2015
                ;dd     irq7
; Real Time Clock Interrupt
        dd      rtc_int         ; 22/02/2015 - IRQ 8 handler - dsectpm.s
                ;dd     irq8    ; INT 28h
        dd      irq9
        dd      irq10
        dd      irq11
        dd      irq12
        dd      irq13
;HDISK_INT1:  ;06/02/2015
        dd      hdc1_int        ; 21/02/2015, IRQ 14 handler
                ;dd     irq14
;HDISK_INT2:  ;06/02/2015
        dd      hdc2_int        ; 21/02/2015, IRQ 15 handler
                ;dd     irq15   ; INT 2Fh
        dd      ignore_int
        dd      0

; 17/02/2015
; 06/02/2015 (unix386.s)
; 11/12/2014 - 22/12/2014 (dsectrm2.s)
;
; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
;
;-- HARDWARE INT  08 H - ( IRQ LEVEL 0 ) ----------------------------------------
;       THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF        :
;       THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
;       IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
;                                                                              :
;       THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE    :
;       POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.             :
;       THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)  :
;       OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE                :
;       DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.                  :
;       THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH          :
;       INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A                :
;       ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.             :
;------------------------------------------------------------------------------
;

timer_int:      ; IRQ 0
;int_08h:       ; Timer
        STI                             ; INTERRUPTS BACK ON
        ;PUSH   DS
        ;PUSH   AX
        ;PUSH   DX                      ; SAVE MACHINE STATE
        ;xor    ax,ax ; real mode (dsectrm2.s)
        ;mov    ds,ax ; real mode (dsectrm2.s)
        ;
        push    eax
        push    edx
        push    ecx
        push    ebx
        push    ds
        push    es
        mov     eax, KDATA
```

```
        mov     ds, ax
        mov     es, ax
        ;
        INC     word [TIMER_LOW]        ; INCREMENT TIME
        JNZ     short T4                ; GO TO TEST_DAY
        INC     word [TIMER_HIGH]       ; INCREMENT HIGH WORD OF TIME
T4:                                     ; TEST_DAY
        CMP     word [TIMER_HIGH],018H  ; TEST FOR COUNT EQUALING 24 HOURS
        JNZ     short T5                ; GO TO DISKETTE_CTL
        CMP     word [TIMER_LOW],0B0H
        JNZ     short T5                ; GO TO DISKETTE_CTL

;-----  TIMER HAS GONE 24 HOURS
        ;;SUB   AX,AX
        ;MOV    word [TIMER_HIGH],AX
        ;MOV    word [TIMER_LOW],AX
        sub     eax, eax
        mov     dword [TIMER_LH], eax
        ;
        MOV     byte [TIMER_OFL],1

;-----  TEST FOR DISKETTE TIME OUT

T5:
        ; 23/12/2014
        jmp     short T6               ; will be replaced with nop, nop
                                       ; (9090h) if a floppy disk
                                       ; is detected.
        ;mov    al,[CS:MOTOR_COUNT]
        mov     al, [MOTOR_COUNT]
        dec     al
        ;mov    [CS:MOTOR_COUNT], al    ; DECREMENT DISKETTE MOTOR CONTROL
        mov     [MOTOR_COUNT], al
        ;mov    [ORG_MOTOR_COUNT], al
        JNZ     short T6                ; RETURN IF COUNT NOT OUT
        mov     al,0F0h
        ;AND    [CS:MOTOR_STATUS],al    ; TURN OFF MOTOR RUNNING BITS
        and     [MOTOR_STATUS], al
        ;and    [ORG_MOTOR_STATUS], al
        MOV     AL,0CH                  ; bit 3 = enable IRQ & DMA,
                                        ; bit 2 = enable controller
                                        ;       1 = normal operation
                                        ;       0 = reset
                                        ; bit 0, 1 = drive select
                                        ; bit 4-7 = motor running bits
        MOV     DX,03F2H                ; FDC CTL PORT
        OUT     DX,AL                   ; TURN OFF THE MOTOR
T6:
        ;inc    word [CS:wait_count]    ; 22/12/2014 (byte -> word)
                                        ; TIMER TICK INTERRUPT
        inc     word [wait_count]
        ;
        ;; DS = 0 (DS <> CS) !
        ;INT    1CH                     ; TRANSFER CONTROL TO A USER ROUTINE
        ;;;;cli ; 17/02/2015
        ;; 22/02/2015 (dsectpm.s --> cancel 'u_timer' call)
        ;;call  u_timer                 ; TRANSFER CONTROL TO A USER ROUTINE
        ;
        ;POP    DX                      ; RESTORE (DX)
        MOV     AL,EOI                  ; GET END OF INTERRUPT MASK
        CLI                             ; DISABLE INTERRUPTS TILL STACK CLEARED
        OUT     INTA00,AL               ; END OF INTERRUPT TO 8259 - 1
        ;POP    AX
        ;POP    DS                      ; RESTORE MACHINE STATE
```

```
        ;
        pop     es
        pop     ds
        pop     ebx
        pop     ecx
        pop     edx
        pop     eax
        ;
        IRETd                           ; RETURN FROM INTERRUPT

; ///////////////

        ; 23/02/2015
        ; 06/02/2015
        ; 07/09/2014
        ; 21/08/2014
;u_timer:
;timer_int:     ; IRQ 0
        ; 06/02/2015
        ;push   eax
        ;push   edx
        ;push   ecx
        ;push   ebx
        ;push   ds
        ;push   es
        ;mov    eax, KDATA
        ;mov    ds, ax
        ;mov    es, ax
;       inc     dword [tcount]
;       mov     ebx, tcountstr + 4
;       mov     ax, word [tcount]
;       mov     ecx, 10
;rp_divtcnt:
;       xor     edx, edx
;       div     ecx
;       add     dl, 30h
;       mov     byte [ebx], dl
;       or      ax, ax
;       jz      short print_lzero
;       dec     ebx
;       jmp     short rp_divtcnt
;print_lzero:
;       cmp     ebx, tcountstr
;       jna     short print_tcount
;       dec     ebx
 ;      mov     byte [ebx], 30h
;       jmp     short print_lzero
;print_tcount:
;       push    esi
;       push    edi
;       mov     esi, timer_msg ; Timer interrupt message
        ; 07/09/2014
;       mov     bx, 1           ; Video page 1
;ptmsg:
;       lodsb
;       or      al, al
;       jz      short ptmsg_ok
;       push    esi
;       push    bx
;       mov     ah,  2Fh ; Green background, white forecolor
;       call    write_tty
;       pop     bx
;       pop     esi
;       jmp     short ptmsg
```

```
        ;; 27/08/2014
        ;mov     edi, 0B8000h + 0A0h ; Row 1
        ;call   printk
        ;
;ptmsg_ok:
        ; 07/09/2014
;       xor     dx, dx          ; column 0, row 0
;       call    set_cpos        ; set cursor position to 0,0
        ;; 23/02/2015
        ; 25/08/2014
;;      mov     ebx, scounter           ; (seconds counter)
;;      dec     byte [ebx+1]            ; (for reading real time clock)
;;;     jns     short timer_eoi         ; 0 -> 0FFh ?
;;      jns     short u_timer_retn
;;      mov     byte [ebx+1], 18        ; (18.2 timer ticks per second)
;;      dec     byte [ebx]              ; 19+18+18+18+18 (5)
;;;     jnz     short timer_eoi         ; (109 timer ticks in 5 seconds)
;;      jnz     short u_timer_retn ; 06/02/2015
;;      mov     byte [ebx], 5
;;      inc     byte [ebx+1] ; 19
;;;timer_eoi:
;;;     mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
;;;     out     20h, al ; 8259 PORT
;;      ;
;;u_timer_retn:  ; 06/02/2015
;       pop     edi
;       pop     esi
;;      ;pop    es
;;      ;pop    ds
;;      ;pop    ebx
;;      ;pop    ecx
;;      ;pop    edx
;;      ;pop    eax
;;      ;iret
;       retn    ; 06/02/2015

        ; 28/08/2014
irq0:
        push    dword 0
        jmp     short which_irq
irq1:
        push    dword 1
        jmp     short which_irq
irq2:
        push    dword 2
        jmp     short which_irq
irq3:
        push    dword 3
        jmp     short which_irq
irq4:
        push    dword 4
        jmp     short which_irq
irq5:
        push    dword 5
        jmp     short which_irq
irq6:
        push    dword 6
        jmp     short which_irq
irq7:
        push    dword 7
        jmp     short which_irq
irq8:
        push    dword 8
        jmp     short which_irq
```

```
irq9:
        push    dword 9
        jmp     short which_irq
irq10:
        push    dword 10
        jmp     short which_irq
irq11:
        push    dword 11
        jmp     short which_irq
irq12:
        push    dword 12
        jmp     short which_irq
irq13:
        push    dword 13
        jmp     short which_irq
irq14:
        push    dword 14
        jmp     short which_irq
irq15:
        push    dword 15
        ;jmp    short which_irq


        ; 29/08/2014
        ; 21/08/2014
which_irq:
        xchg    eax, [esp]  ; 28/08/2014
        push    ebx
        push    esi
        push    edi
        push    ds
        push    es
        ;
        mov     bl, al
        ;
        mov     eax, KDATA
        mov     ds, ax
        mov     es, ax

        ; 27/08/2014
        add     dword [scr_row], 0A0h
        ;
        mov     ah, 17h ; blue (1) background,
                        ; light gray (7) forecolor
        mov     edi, [scr_row]
        mov     al, 'I'
        stosw
        mov     al, 'R'
        stosw
        mov     al, 'Q'
        stosw
        mov     al, ' '
        stosw
        mov     al, bl
        cmp     al, 10
        jb      short iix
        mov     al, '1'
        stosw
        mov     al, bl
        sub     al, 10
iix:
        add     al, '0'
        stosw
        mov     al, ' '
```

```
        stosw
        mov     al, '!'
        stosw
        mov     al, ' '
        stosw
        ; 23/02/2015
        cmp     bl, 7 ; check for IRQ 8 to IRQ 15
        jna     iiret
        mov     al, 20h  ; END OF INTERRUPT COMMAND TO
        out     0A0h, al ; the 2nd 8259
        jmp     iiret
        ;
        ; 22/08/2014
        ;mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
        ;out    20h, al ; 8259 PORT
        ;
        ;pop    es
        ;pop    ds
        ;pop    edi
        ;pop    esi
        ;pop    ebx
        ;pop    eax
        ;iret

        ; 25/08/2014
exc0:
        push    dword 0
        jmp     cpu_except
exc1:
        push    dword 1
        jmp     cpu_except
exc2:
        push    dword 2
        jmp     cpu_except
exc3:
        push    dword 3
        jmp     cpu_except
exc4:
        push    dword 4
        jmp     cpu_except
exc5:
        push    dword 5
        jmp     cpu_except
exc6:
        push    dword 6
        jmp     cpu_except
exc7:
        push    dword 7
        jmp     cpu_except
exc8:
        ; [esp] = Error code
        push    dword 8
        jmp     cpu_except_en
exc9:
        push    dword 9
        jmp     cpu_except
exc10:
        ; [esp] = Error code
        push    dword 10
        jmp     cpu_except_en
exc11:
        ; [esp] = Error code
        push    dword 11
        jmp     cpu_except_en
```

```
exc12:
        ; [esp] = Error code
        push    dword 12
        jmp     cpu_except_en
exc13:
        ; [esp] = Error code
        push    dword 13
        jmp     cpu_except_en
exc14:
        ; [esp] = Error code
        push    dword 14
        jmp     short cpu_except_en
exc15:
        push    dword 15
        jmp     cpu_except
exc16:
        push    dword 16
        jmp     cpu_except
exc17:
        ; [esp] = Error code
        push    dword 17
        jmp     short cpu_except_en
exc18:
        push    dword 18
        jmp     short cpu_except
exc19:
        push    dword 19
        jmp     short cpu_except
exc20:
        push    dword 20
        jmp     short cpu_except
exc21:
        push    dword 21
        jmp     short cpu_except
exc22:
        push    dword 22
        jmp     short cpu_except
exc23:
        push    dword 23
        jmp     short cpu_except
exc24:
        push    dword 24
        jmp     short cpu_except
exc25:
        push    dword 25
        jmp     short cpu_except
exc26:
        push    dword 26
        jmp     short cpu_except
exc27:
        push    dword 27
        jmp     short cpu_except
exc28:
        push    dword 28
        jmp     short cpu_except
exc29:
        push    dword 29
        jmp     short cpu_except
exc30:
        push    dword 30
        jmp     short cpu_except
exc31:
        push    dword 31
        ;jmp     short cpu_except
```

```
        ; 28/08/2014
cpu_except_en:
        xchg    eax, [esp+4] ; Error code
        push    ds
        push    ax
        mov     ax, KDATA
        mov     ds, ax
        pop     ax
        mov     dword [error_code], eax
        pop     ds
        mov     eax, [esp] ; Exception number
        add     esp, 4
        xchg    eax, [esp]
                ; eax = eax before exception
                ; [esp] -> exception number
                ; [esp+4] -> EIP to return
        ; 29/08/2014
        ; 28/08/2014
        ; 25/08/2014
        ; 21/08/2014
cpu_except:             ; CPU Exceptions
        cld
        xchg    eax, [esp]
                        ; eax = Exception number
                        ; [esp] = eax (before exception)
        push    ebx
        mov     ebx, hang
        xchg    ebx, [esp+8]
                        ; EIP (points to instruction which faults)
                        ; New EIP (hang)
        push    esi
        push    edi
        push    ds
        push    es
        ;
        push    ax      ; Exception number
        mov     ax, KDATA
        mov     ds, ax
        mov     es, ax
        pop     ax
        ;
        mov     [FaultOffset], ebx
        ;
        mov     ah, al
        and     al, 0Fh
        cmp     al, 9
        jna     short h1ok
        add     al, 'A'-':'
h1ok:
        shr     ah, 1
        shr     ah, 1
        shr     ah, 1
        shr     ah, 1
        cmp     ah, 9
        jna     short h2ok
        add     ah, 'A'-':'
h2ok:
        xchg    ah, al
        add     ax, '00'
        mov     word [excnstr], ax
        ;
        ; 29/08/2014
        mov     eax, dword [FaultOffset]
```

```
        push    ecx
        push    edx
        mov     ebx, esp
        mov     ecx, 10     ; divisor to convert
                            ; binary number to decimal string
b2d1:
        xor     edx, edx
        div     ecx
        push    dx
        cmp     eax, ecx
        jnb     short b2d1
        mov     edi, EIPstr ; EIP value
                            ; points to instruction which faults
b2d2:
        add     al, '0'
        stosb               ; write decimal digit to its place
        cmp     ebx, esp
        jna     short b2d3
        pop     ax
        jmp     short b2d2
b2d3:
        mov     al, 20h     ; space
        stosb
        xor     al, al      ; to do it an ASCIIZ string
        stosb
        ;
        pop     edx
        pop     ecx
        ;
        mov     ah, 4Fh ; red (4) background,
                        ; white (F) forecolor
        mov     esi, exc_msg ; message offset
        ;
        jmp     short piemsg
        ;
        ;add    dword [scr_row], 0A0h
        ;mov    edi, [scr_row]
        ;
        ;call   printk
        ;
        ;mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
        ;out    20h, al ; 8259 PORT
        ;
        ;pop    es
        ;pop    ds
        ;pop    edi
        ;pop    esi
        ;pop    eax
        ;iret

        ; 23/02/2015
        ; 20/08/2014
ignore_int:
        push    eax
        push    ebx ; 23/02/2015
        push    esi
        push    edi
        push    ds
        push    es
        ;
        mov     eax, KDATA
        mov     ds, ax
        mov     es, ax
        ;
```

```
        mov     ah, 67h ; brown (6) background,
                        ; light gray (7) forecolor
        mov     esi, int_msg ; message offset
piemsg:
        ; 27/08/2014
        add     dword [scr_row], 0A0h
        mov     edi, [scr_row]
        ;
        call    printk
        ; 23/02/2015
        mov     al, 20h  ; END OF INTERRUPT COMMAND TO
        out     0A0h, al ; the 2nd 8259
        ;
iiret:
        ; 22/08/2014
        mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
        out     20h, al ; 8259 PORT
        ;
        pop     es
        pop     ds
        pop     edi
        pop     esi
        pop     ebx ; 29/08/2014
        pop     eax
        iret

        ; 22/02/2015 (dsectpm.s)
        ; 22/08/2014 - 07/09/2014 (unix386.s)
rtc_int:        ; Real Time Clock Interrupt (IRQ 8)
        ; 22/08/2014
        push    eax
        push    ebx ; 29/08/2014
        push    esi
        push    edi
        push    ds
        push    es
        ;
        mov     eax, KDATA
        mov     ds, ax
        mov     es, ax
        ;
        ; 22/02/2015
        mov     edi, 0B8000h+0A0h+04Ch ; Row 1, Column 38
        cmp     byte [edi+1], 3Fh ; cyan (3) Background
                        ; white (F) forecolor
                        ; (display disk sector frame)
        jne     short rtc_np
        ; 25/08/2014
        call    rtc_p
rtc_np:
        ; 22/02/2015 - dsectpm.s
        ; [ source: http://wiki.osdev.org/RTC ]
        ; read status register C to complete procedure
        ;(it is needed to get a next IRQ 8)
        mov     al, 0Ch ;
        out     70h, al ; select register C
        nop
        in      al, 71h ; just throw away contents
        ; 22/02/2015
        MOV     AL,EOI          ; END OF INTERRUPT
        OUT     INTB00,AL       ; FOR CONTROLLER #2
        ;
        jmp     short iiret
```

```
        ; 22/08/2014
        ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
        ; (INT 1Ah)
        ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
time_of_day:
        call    UPD_IPR                 ; WAIT TILL UPDATE NOT IN PROGRESS
        jc      short rtc_retn
        mov     al, CMOS_SECONDS
        call    CMOS_READ
        mov     byte [time_seconds], al
        mov     al, CMOS_MINUTES
        call    CMOS_READ
        mov     byte [time_minutes], al
        mov     al, CMOS_HOURS
        call    CMOS_READ
        mov     byte [time_hours], al
        mov     al, CMOS_DAY_WEEK
        call    CMOS_READ
        mov     byte [date_wday], al
        mov     al, CMOS_DAY_MONTH
        call    CMOS_READ
        mov     byte [date_day], al
        mov     al, CMOS_MONTH
        call    CMOS_READ
        mov     byte [date_month], al
        mov     al, CMOS_YEAR
        call    CMOS_READ
        mov     byte [date_year], al
        mov     al, CMOS_CENTURY
        call    CMOS_READ
        mov     byte [date_century], al
        ;
        mov     al, CMOS_SECONDS
        call    CMOS_READ
        cmp     al, byte [time_seconds]
        jne     short time_of_day

rtc_retn:
        retn

rtc_p:
        ; 22/02/2015 dsectpm.s
        ; 25/08/2014 - 07/09/2014 unix386.s
        ; Print Real Time Clock content
        ;
        call    time_of_day
        jc      short rtc_retn
        ;
        cmp     al, byte [ptime_seconds]
        je      short rtc_retn ; 29/08/2014
        ;
        mov     byte [ptime_seconds], al
        ;
        mov     al, byte [date_century]
        call    bcd_to_ascii
        mov     word [datestr+6], ax
        mov     al, byte [date_year]
        call    bcd_to_ascii
        mov     word [datestr+8], ax
        mov     al, byte [date_month]
        call    bcd_to_ascii
        mov     word [datestr+3], ax
        mov     al, byte [date_day]
        call    bcd_to_ascii
```

```
        mov     word [datestr], ax
        ;
        xor     ebx, ebx
        mov     bl, byte [date_wday]
        shl     bl, 2
        add     ebx, daytmp
        mov     eax, dword [ebx]
        mov     dword [daystr], eax
        ;
        mov     al, byte [time_hours]
        call    bcd_to_ascii
        mov     word [timestr], ax
        mov     al, byte [time_minutes]
        call    bcd_to_ascii
        mov     word [timestr+3], ax
        mov     al, byte [time_seconds]
        call    bcd_to_ascii
        mov     word [timestr+6], ax
        ;
        mov     esi, rtc_msg ; message offset
        ; 22/02/2015
        ;mov    edi, 0B8000h+0A0h+04Ch ; Row 1, Column 38
        ;mov    ah, [edi+1]
        ;cmp    ah, 3Fh ; cyan (3) Background
                        ; white (F) forecolor
                        ; (display disk sector frame)
        ;jne    short prtcmsg_ok
prtcmsg:
        lodsb
        or      al, al
        jz      short prtcmsg_ok
        stosb
        inc     edi
        jmp     short prtcmsg
prtcmsg_ok:
        retn

; Default IRQ 7 handler against spurious IRQs (from master PIC)
; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
default_irq7:
        push    ax
        mov     al, 0Bh  ; In-Service register
        out     20h, al
        jmp short $+2
        jmp short $+2
        in      al, 20h
        and     al, 80h ; bit 7 (is it real IRQ 7 or fake?)
        jz      short irq7_iret ; Fake (spurious) IRQ, do not send EOI
        mov     al, 20h ; EOI
        out     20h, al
irq7_iret:
        pop     ax
        iretd

        ; 22/08/2014
        ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
CMOS_READ:
        pushf           ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
        rol     al, 1   ; MOVE NMI BIT TO LOW POSITION
        stc             ; FORCE NMI BIT ON IN CARRY FLAG
        rcr     al, 1   ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
        cli             ; DISABLE INTERRUPTS
        out     CMOS_PORT, al   ; ADDRESS LOCATION AND DISABLE NMI
        nop             ; I/O DELAY
```

```
        in      al, CMOS_DATA   ; READ THE REQUESTED CMOS LOCATION
        push    ax      ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
        mov     al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
        rcr     al, 1   ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
        out     CMOS_PORT, al   ; SET DEFAULT TO READ ONLY REGISTER
        pop     ax      ; RESTORE (AH) AND (AL), CMOS BYTE
        popf
        retn            ; RETURN WITH FLAGS RESTORED

        ; 22/08/2014
        ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
UPD_IPR:                                ; WAIT TILL UPDATE NOT IN PROGRESS
        push    ecx
        mov     ecx, 65535              ; SET TIMEOUT LOOP COUNT (= 800)
                ; mov cx, 800
UPD_10:
        mov     al, CMOS_REG_A          ; ADDRESS STATUS REGISTER A
        cli                             ; NO TIMER INTERRUPTS DURING UPDATES
        call    CMOS_READ               ; READ UPDATE IN PROCESS FLAG
        test    al, 80h                 ; IF UIP BIT IS ON ( CANNOT READ TIME )
        jz      short UPD_90            ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
        sti                             ; ALLOW INTERRUPTS WHILE WAITING
        loop    UPD_10                  ; LOOP TILL READY OR TIMEOUT
        xor     eax, eax                ; CLEAR RESULTS IF ERROR
                ; xor ax, ax
        stc                             ; SET CARRY FOR ERROR
UPD_90:
        pop     ecx                     ; RESTORE CALLERS REGISTER
        cli                             ; INTERRUPTS OFF DURING SET
        retn                            ; RETURN WITH CY FLAG SET

bcd_to_ascii:
        ; 25/08/2014
        ; INPUT ->
        ;       al = Packed BCD number
        ; OUTPUT ->
        ;       ax = ASCII word/number
        ;
        ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
        ;
        db 0D4h,10h                     ; Undocumented inst. AAM
                                        ; AH = AL / 10h
                                        ; AL = AL MOD 10h
        or ax,'00'                      ; Make it ASCII based

        xchg ah, al

        retn

; 27/08/2014
scr_row:
        dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
scr_col:
        dd 0

ALIGN 8

gdt:    ; Global Descriptor Table (29/12/2013)
        dw 0, 0, 0, 0           ; NULL descriptor
        ; 18/08/2014
                        ; 8h kernel code segment, base = 00000000h
        dw 0FFFFh, 0, 9A00h
        dw 00C3h                ; KCODE
                        ; 10h kernel data segment, base = 00000000h
```

```
        dw 0FFFFh, 0, 9200h
        dw 00C3h                ; KDATA
                        ; 1Bh user code segment, base address = 0
        ;dw 0FFFFh, 0, 0FA00h, 0CBh ; UCODE
                        ; 23h user data segment, base address = 0
        ;dw 0FFFFh, 0, 0F200h, 0CBh ; UDATA

gdt_end:
        ;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                                        ;; Type= 1 (code)/C=0/R=1/A=0
                ; P= Present, DPL=0=ring 0,  1= user (0= system)
                ; 1= Code C= non-Conforming, R= Readable, A = Accessed

        ;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                                        ;; Type= 0 (data)/E=0/W=1/A=0
                ; P= Present, DPL=0=ring 0,  1= user (0= system)
                ; 0= Data E= Expansion direction (1= down, 0= up)
                ; W= Writable, A= Accessed

        ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
                                        ;; Type= 1 (code)/C=0/R=1/A=0
                ; P= Present, DPL=3=ring 3,  1= user (0= system)
                ; 1= Code C= non-Conforming, R= Readable, A = Accessed

        ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
                                        ;; Type= 0 (data)/E=0/W=1/A=0
                ; P= Present, DPL=3=ring 3,  1= user (0= system)
                ; 0= Data E= Expansion direction (1= down, 0= up)

        ;; C3h = 1100 0011b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=0011b (3)
                ;; Limit = 3FFFFh (=> 3FFFFh+1= 40000h)
                ;        = 40000h * 1000h (G=1) = 1GB
                ; G= Granularity (1= 4KB), B=Big (32 bit),
                ; AVL= Available to programmers
        ;; CBh = 1100 1011b (GDT byte 6) G=1/DB=1/0/AVL=0, Limit=1011b (3)
                ;; Limit = BFFFFh (=> BFFFFh+1= C0000h)
                ;        = C0000h * 1000h (G=1) = 3GB
                ; G= Granularity (1= 4KB),
                ; D=Default operand size (32 bit),
                ; AVL= Available to programmers


        ; Interrupt Descriptor Table (20/08/2014)
idt:
        times   64*8 db 0   ; INT 0 to INT 3Fh
idt_end:


gdtd:
        dw gdt_end - gdt - 1    ; Limit (size)
        dd gdt                  ; Address of the GDT

        ; 20/08/2014
idtd:
        dw idt_end - idt - 1    ; Limit (size)
        dd idt                  ; Address of the IDT



db 0
Align 2

msgPM:
        db "Protected mode and paging are ENABLED ... ", 0
```

```
Align 2

; 20/08/2014
  ; /* This is the default interrupt "handler" :-) */
  ; Linux v0.12 (head.s)
int_msg:
        db "Unknown interrupt ! ", 0

Align 2

; 21/08/2014
timer_msg:
        db "IRQ 0 (INT 20h) ! Timer Interrupt : "
tcountstr:
        db "00000 "
        db 0

Align 2
        ; 21/08/2014
exc_msg:
        db "CPU exception ! "
excnstr:                ; 25/08/2014
        db "??h", "  EIP : "
        ; 29/08/2014
EIPstr: times 12 db 0


Align 4

tcount:
        dd 0

; 22/08/2014 (RTC)
; (Packed BCD)
time_seconds: db 0
time_minutes: db 0
time_hours:   db 0
date_wday:    db 0
date_day:     db 0
date_month:   db 0
date_year:    db 0
date_century: db 0

rtc_msg:
        db "Real Time Clock - "
datestr:
        db "00/00/0000"
        db " "
daystr:
        db "DAY "
timestr:
        db "00:00:00"
        db " "
        db 0
daytmp:
        ; 28/02/2015
        db "??? SUN MON TUE WED THU FRI SAT "

ptime_seconds: db 0FFh

        ;;23/02/2015
        ; 25/08/2014
;scounter:
```

```
;       db 5
;       db 19

; 20/02/2015
; 03/12/2014
; 07/09/2014
; KEYBOARD INTERRUPT HANDLER
; (kb_int - Retro UNIX 8086 v1 - U0.ASM, 30/06/2014)

keyb_int:
        ; 25/02/2015
        ; 20/02/2015
        ; 03/12/2014 (getc_int - INT 16h modifications)
        ; 07/09/2014 - Retro UNIX 386 v1
        ; 30/06/2014
        ; 10/05/2013
        ; Retro Unix 8086 v1 feature only!
        ; 03/03/2014

        push    ds
        push    ebx
        push    eax
        ;
        mov     ax, KDATA
        mov     ds, ax
        ;
        pushfd
        push    cs
        call    kb_int
        ;
        mov     ah, 11h  ; 03/12/2014
        call    getc
        jz      short keyb_int4
        ;
        mov     ah, 10h  ; 03/12/2014
        call    getc
        ;
        ; 20/02/2015
        xor     ebx, ebx
        mov     bl, [ptty]  ; active_page
        ;
        and     al, al
        jnz     short keyb_int1
        ;
        cmp     ah, 68h ; ALT + F1 key
        jb      short keyb_int1
        cmp     ah, 6Fh  ; ALT + F8 key
        ja      short keyb_int1
        ;
        mov     bh, bl
        add     bh, 68h
        cmp     bh, ah
        je      short keyb_int0
        mov     al, ah
        sub     al, 68h
        call    tty_sw
        xor     ax, ax
keyb_int0: ; 25/02/2015
        xor     ebx, ebx
        mov     bl, [ptty]  ; active_page
keyb_int1:
        shl     bl, 1
        add     ebx, ttychr
        ;
```

```
        ;or     ax, ax
        ;jz     short keyb_int2
        ;
        ;cmp    word [ebx], 0
        ;ja     short kb_int_3
keyb_int2:
        mov     word [ebx], ax  ; Save ascii code
                                ; and scan code of the character
                                ; for current tty (or last tty
                                ; just before tty switch).
keyb_int3:
        ;mov     al, byte [ptty]
        ;call   wakeup
        ;
keyb_int4:
        pop     eax
        pop     ebx
        pop     ds
        iret

; 18/02/2015
; REMINDER: Only 'keyb_int' (IRQ 9) must call getc.
; 'keyb_int' always handles 'getc' at 1st and puts the
; scancode and ascii code of the character
; in the tty input (ttychr) buffer.
; Test procedures must call 'getch' for tty input
; otherwise, 'getc' will not be able to return to the caller
; due to infinite (key press) waiting loop.
;
; 03/12/2014
; 26/08/2014
; KEYBOARD I/O
; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)

;NOTE: 'k0' to 'k7' are name of OPMASK registers.
;       (The reason of using '_k' labels!!!) (27/08/2014)
;NOTE: 'NOT' keyword is '~' unary operator in NASM.
;       ('NOT LC_HC' --> '~LC_HC') (bit reversing operator)

getc:
        pushfd  ; 28/08/2014
        push    cs
        call    getc_int
        retn

getc_int:
        ; 28/02/2015
        ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
        ;             instead of pc-at bios - 1985-)
        ; 28/08/2014 (_k1d)
        ; 30/06/2014
        ; 03/03/2014
        ; 28/02/2014
        ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
        ; rombios source code (21/04/1986)
        ;        'keybd.asm', INT 16H, KEYBOARD_IO
        ;
        ; KYBD --- 03/06/86  KEYBOARD BIOS
        ;
        ;--- INT 16 H
----------------------------------------------------------------
        ; KEYBOARD I/O
     :
        ;       THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT
```

```
      :
         ; INPUT
      :
         ;       (AH)= 00H  READ THE NEXT ASCII CHARACTER ENTERED FROM THE
KEYBOARD,   :
         ;                  RETURN THE RESULT IN (AL), SCAN CODE IN (AH).
      :
         ;                  THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO
THE   :
         ;                  STANDARD PC OR PCAT KEYBOARD
      :

;--------------------------------------------------------------------------:
         ;       (AH)= 01H  SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER
IS    :
         ;                  AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.
      :
         ;                  (ZF)= 1 -- NO CODE AVAILABLE
      :
         ;                  (ZF)= 0 -- CODE IS AVAILABLE  (AX)= CHARACTER
      :
         ;                  IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE
READ IS :
         ;                  IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.
      :
         ;                  THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE
CODES    :

;--------------------------------------------------------------------------:
         ;       (AH)= 02H  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
      :
         ;                  THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
      :
         ;                  EQUATES FOR @KB_FLAG
      :

;--------------------------------------------------------------------------:
         ;       (AH)= 03H  SET TYPAMATIC RATE AND DELAY
      :
         ;             (AL) = 05H
      :
         ;             (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)
      :
         ;
      :
         ;                     REGISTER     RATE      REGISTER     RATE
      :
         ;                      VALUE     SELECTED     VALUE     SELECTED
      :
         ;                     ----------------------------------------
      :
         ;                       00H        30.0        10H        7.5
      :
         ;                       01H        26.7        11H        6.7
      :
         ;                       02H        24.0        12H        6.0
      :
         ;                       03H        21.8        13H        5.5
      :
         ;                       04H        20.0        14H        5.0
      :
         ;                       05H        18.5        15H        4.6
      :
         ;                       06H        17.1        16H        4.3
```

```
:
   ;                        07H        16.0        17H        4.0
:
   ;                        08H        15.0        18H        3.7
:
   ;                        09H        13.3        19H        3.3
:
   ;                        0AH        12.0        1AH        3.0
:
   ;                        0BH        10.9        1BH        2.7
:
   ;                        0CH        10.0        1CH        2.5
:
   ;                        0DH         9.2        1DH        2.3
:
   ;                        0EH         8.6        1EH        2.1
:
   ;                        0FH         8.0        1FH        2.0
:
   ;
:
   ;              (BH) = TYPAMATIC DELAY  (BITS 2 - 7 MUST BE RESET TO 0)
:
   ;
:
   ;                       REGISTER     DELAY
:
   ;                        VALUE       VALUE
:
   ;                       ------------------
:
   ;                        00H        250 ms
:
   ;                        01H        500 ms
:
   ;                        02H        750 ms
:
   ;                        03H       1000 ms
:

;-------------------------------------------------------------------------------:
   ;       (AH)= 05H  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN
KEYBOARD    :
   ;                  BUFFER AS IF STRUCK FROM KEYBOARD
:
   ;                  ENTRY:  (CL) = ASCII CHARACTER
:
   ;                          (CH) = SCAN CODE
:
   ;                  EXIT:   (AH) = 00H = SUCCESSFUL OPERATION
:
   ;                          (AL) = 01H = UNSUCCESSFUL - BUFFER FULL
:
   ;                  FLAGS:  CARRY IF ERROR
:

;-------------------------------------------------------------------------------:
   ;       (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,
:
   ;                  OTHERWISE SAME AS FUNCTION AH=0
:

;-------------------------------------------------------------------------------:
```

```
        ;       (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,
     :
        ;                  OTHERWISE SAME AS FUNCTION AH=1
     :

;--------------------------------------------------------------------------:
        ;       (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER
     :
        ;                  AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT
     :
        ;                  CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3
     :
        ; OUTPUT
     :
        ;       AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED
     :
        ;       ALL REGISTERS RETAINED
     :

;--------------------------------------------------------------------------

        sti                             ; INTERRUPTS BACK ON
        push    ds                      ; SAVE CURRENT DS
        push    ebx                     ; SAVE BX TEMPORARILY
        ;push   ecx                     ; SAVE CX TEMPORARILY
        mov     bx, KDATA
        mov     ds, bx                  ; PUT SEGMENT VALUE OF DATA AREA INTO DS
        or      ah, ah                  ; CHECK FOR (AH)= 00H
        jz      short _K1               ; ASCII_READ
        dec     ah                      ; CHECK FOR (AH)= 01H
        jz      short _K2               ; ASCII_STATUS
        dec     ah                      ; CHECK FOR (AH)= 02H
        jz      _K3                     ; SHIFT STATUS
        dec     ah                      ; CHECK FOR (AH)= 03H
        jz      _K300                   ; SET TYPAMATIC RATE/DELAY
        sub     ah, 2                   ; CHECK FOR (AH)= 05H
        jz      _K500                   ; KEYBOARD WRITE
_KIO1:
        sub     ah, 11                  ; AH =  10H
        jz      short _K1E              ; EXTENDED ASCII READ
        dec     ah                      ; CHECK FOR (AH)= 11H
        jz      short _K2E              ; EXTENDED_ASCII_STATUS
        dec     ah                      ; CHECK FOR (AH)= 12H
        jz      short _K3E              ; EXTENDED_SHIFT_STATUS
_KIO_EXIT:
        ;pop    ecx                     ; RECOVER REGISTER
        pop     ebx                     ; RECOVER REGISTER
        pop     ds                      ; RECOVER SEGMENT
        iretd                           ; INVALID COMMAND, EXIT

        ;-----  ASCII CHARACTER
_K1E:
        call    _K1S                    ; GET A CHARACTER FROM THE BUFFER
(EXTENDED)
        call    _KIO_E_XLAT             ; ROUTINE TO XLATE FOR EXTENDED CALLS
        jmp     short _KIO_EXIT         ; GIVE IT TO THE CALLER
_K1:
        call    _K1S                    ; GET A CHARACTER FROM THE BUFFER
        call    _KIO_S_XLAT             ; ROUTINE TO XLATE FOR STANDARD CALLS
        jc      short _K1               ; CARRY SET MEANS TROW CODE AWAY
_K1A:
        jmp     short _KIO_EXIT         ; RETURN TO CALLER

        ;-----  ASCII STATUS
```

```
_K2E:
        call    _K2S                    ; TEST FOR CHARACTER IN BUFFER
(EXTENDED)
        jz      short _K2B              ; RETURN IF BUFFER EMPTY
        pushf                           ; SAVE ZF FROM TEST
        call    _KIO_E_XLAT             ; ROUTINE TO XLATE FOR EXTENDED CALLS
        jmp     short _K2A              ; GIVE IT TO THE CALLER
_K2:
        call    _K2S                    ; TEST FOR CHARACTER IN BUFFER
        jz      short _K2B              ; RETURN IF BUFFER EMPTY
        pushf                           ; SAVE ZF FROM TEST
        call    _KIO_S_XLAT             ; ROUTINE TO XLATE FOR STANDARD CALLS
        jnc     short _K2A              ; CARRY CLEAR MEANS PASS VALID CODE
        popf                            ; INVALID CODE FOR THIS TYPE OF CALL
        call    _K1S                    ; THROW THE CHARACTER AWAY
        jmp     short _K2               ; GO LOOK FOR NEXT CHAR, IF ANY
_K2A:
        popf                            ; RESTORE ZF FROM TEST
_K2B:
        ;pop    ecx                     ; RECOVER REGISTER
        pop     ebx                     ; RECOVER REGISTER
        pop     ds                      ; RECOVER SEGMENT
        retf    4                       ; THROW AWAY (e)FLAGS

        ;-----  SHIFT STATUS
_K3E:                                   ; GET THE EXTENDED SHIFT STATUS FLAGS
        mov     ah, [KB_FLAG_1]         ; GET SYSTEM SHIFT KEY STATUS
        and     ah, SYS_SHIFT           ; MASK ALL BUT SYS KEY BIT
        ;mov    cl, 5                   ; SHIFT THEW SYSTEMKEY BIT OVER TO
        ;shl    ah, cl                  ; BIT 7 POSITION
        shl     ah, 5
        mov     al, [KB_FLAG_1]         ; GET SYSTEM SHIFT STATES BACK
        and     al, 01110011b           ; ELIMINATE SYS SHIFT, HOLD_STATE AND
INS_SHIFT
        or      ah, al                  ; MERGE REMAINING BITS INTO AH
        mov     al, [KB_FLAG_3]         ; GET RIGHT CTL AND ALT
        and     al, 00001100b           ; ELIMINATE LC_E0 AND LC_E1
        or      ah, al                  ; OR THE SHIFT FLAGS TOGETHER
_K3:
        mov     al, [KB_FLAG]           ; GET THE SHIFT STATUS FLAGS
        jmp     short _KIO_EXIT         ; RETURN TO CALLER

        ;-----  SET TYPAMATIC RATE AND DELAY
_K300:
        cmp     al, 5                   ; CORRECT FUNCTION CALL?
        jne     short _KIO_EXIT         ; NO, RETURN
        test    bl, 0E0h                ; TEST FOR OUT-OF-RANGE RATE
        jnz     short _KIO_EXIT         ; RETURN IF SO
        test    BH, 0FCh                ; TEST FOR OUT-OF-RANGE DELAY
        jnz     short _KIO_EXIT         ; RETURN IF SO
        mov     al, KB_TYPA_RD          ; COMMAND FOR TYPAMATIC RATE/DELAY

        call    SND_DATA                ; SEND TO KEYBOARD
        ;mov    cx, 5                   ; SHIFT COUNT
        ;shl    bh, cl                  ; SHIFT DELAY OVER
        shl     bh, 5
        mov     al, bl                  ; PUT IN RATE
        or      al, bh                  ; AND DELAY
        call    SND_DATA                ; SEND TO KEYBOARD
        jmp     _KIO_EXIT               ; RETURN TO CALLER

        ;-----  WRITE TO KEYBOARD BUFFER
_K500:
        push    esi                     ; SAVE SI (esi)
```

```
        cli                             ;
        mov     ebx, [BUFFER_TAIL]      ; GET THE 'IN TO' POINTER TO THE BUFFER
        mov     esi, ebx                ; SAVE A COPY IN CASE BUFFER NOT FULL
        call    _K4                     ; BUMP THE POINTER TO SEE IF BUFFER IS
FULL
        cmp     ebx, [BUFFER_HEAD]      ; WILL THE BUFFER OVERRUN IF WE STORE
THIS?
        je      short _K502             ; YES - INFORM CALLER OF ERROR
        mov     [esi], cx               ; NO - PUT ASCII/SCAN CODE INTO BUFFER
        mov     [BUFFER_TAIL], ebx      ; ADJUST 'IN TO' POINTER TO REFLECT
CHANGE
        sub     al, al                  ; TELL CALLER THAT OPERATION WAS
SUCCESSFUL
        jmp     short _K504             ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
_K502:
        mov     al, 01h                 ; BUFFER FULL INDICATION
_K504:
        sti
        pop     esi                     ; RECOVER SI (esi)
        jmp     _KIO_EXIT               ; RETURN TO CALLER WITH STATUS IN AL

        ;-----  READ THE KEY TO FIGURE OUT WHAT TO DO -----
_K1S:
        cli     ; 03/12/2014
        mov     ebx, [BUFFER_HEAD]      ; GET POINTER TO HEAD OF BUFFER
        cmp     ebx, [BUFFER_TAIL]      ; TEST END OF BUFFER
        ;jne    short _K1U              ; IF ANYTHING IN BUFFER SKIP INTERRUPT
        jne     short _k1x ; 03/12/2014
        ;
        ; 03/12/2014
        ; 28/08/2014
        ; PERFORM OTHER FUNCTION ?? here !
        ;; MOV  AX, 9002h               ; MOVE IN WAIT CODE & TYPE
        ;; INT  15H                    ; PERFORM OTHER FUNCTION
_K1T:                                   ; ASCII READ
        sti                             ; INTERRUPTS BACK ON DURING LOOP
        nop                             ; ALLOW AN INTERRUPT TO OCCUR
_K1U:
        cli                             ; INTERRUPTS BACK OFF
        mov     ebx, [BUFFER_HEAD]      ; GET POINTER TO HEAD OF BUFFER
        cmp     ebx, [BUFFER_TAIL]      ; TEST END OF BUFFER
_k1x:
        push    ebx                     ; SAVE ADDRESS
        pushf                           ; SAVE FLAGS
        call    MAKE_LED                ; GO GET MODE INDICATOR DATA BYTE
        mov     bl, [KB_FLAG_2]         ; GET PREVIOUS BITS
        xor     bl, al                  ; SEE IF ANY DIFFERENT
        and     bl, 07h ; KB_LEDS       ; ISOLATE INDICATOR BITS
        jz      short _K1V              ; IF NO CHANGE BYPASS UPDATE
        call    SND_LED1
        cli                             ; DISABLE INTERRUPTS
_K1V:
        popf                            ; RESTORE FLAGS
        pop     ebx                     ; RESTORE ADDRESS
        je      short _K1T              ; LOOP UNTIL SOMETHING IN BUFFER
        ;
        mov     ax, [ebx]               ; GET SCAN CODE AND ASCII CODE
        call    _K4                     ; MOVE POINTER TO NEXT POSITION
        mov     [BUFFER_HEAD], ebx      ; STORE VALUE IN VARIABLE
        retn                            ; RETURN

        ;-----  READ THE KEY TO SEE IF ONE IS PRESENT -----
_K2S:
        cli                             ; INTERRUPTS OFF
```

```
        mov     ebx, [BUFFER_HEAD]      ; GET HEAD POINTER
        cmp     ebx, [BUFFER_TAIL]      ; IF EQUAL (Z=1) THEN NOTHING THERE
        mov     ax, [ebx]
        pushf                           ; SAVE FLAGS
        push    ax                      ; SAVE CODE
        call    MAKE_LED                ; GO GET MODE INDICATOR DATA BYTE
        mov     bl, [KB_FLAG_2]         ; GET PREVIOUS BITS
        xor     bl, al                  ; SEE IF ANY DIFFERENT
        and     bl, 07h ; KB_LEDS       ; ISOLATE INDICATOR BITS
        jz      short _K2T              ; IF NO CHANGE BYPASS UPDATE
        call    SND_LED                 ; GO TURN ON MODE INDICATORS
_K2T:
        pop     ax                      ; RESTORE CODE
        popf                            ; RESTORE FLAGS
        sti                             ; INTERRUPTS BACK ON
        retn                            ; RETURN

        ;-----  ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
_KIO_E_XLAT:
        cmp     al, 0F0h                ; IS IT ONE OF THE FILL-INs?
        jne     short _KIO_E_RET        ; NO, PASS IT ON
        or      ah, ah                  ; AH = 0 IS SPECIAL CASE
        jz      short _KIO_E_RET        ; PASS THIS ON UNCHANGED
        xor     al, al                  ; OTHERWISE SET AL = 0
_KIO_E_RET:
        retn                            ; GO BACK

        ;-----  ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
_KIO_S_XLAT:
        cmp     ah, 0E0h                ; IS IT KEYPAD ENTER OR / ?
        jne     short _KIO_S2           ; NO, CONTINUE
        cmp     al, 0Dh                 ; KEYPAD ENTER CODE?
        je      short _KIO_S1           ; YES, MASSAGE A BIT
        cmp     al, 0Ah                 ; CTRL KEYPAD ENTER CODE?
        je      short _KIO_S1           ; YES, MASSAGE THE SAME
        mov     ah, 35h                 ; NO, MUST BE KEYPAD /
_kio_ret: ; 03/12/2014
        clc
        retn
        ;jmp    short _KIO_USE          ; GIVE TO CALLER
_KIO_S1:
        mov     ah, 1Ch                 ; CONVERT TO COMPATIBLE OUTPUT
        ;jmp    short _KIO_USE          ; GIVE TO CALLER
        retn
_KIO_S2:
        cmp     ah, 84h                 ; IS IT ONE OF EXTENDED ONES?
        ja      short _KIO_DIS          ; YES, THROW AWAY AND GET ANOTHER CHAR
        cmp     al, 0F0h                ; IS IT ONE OF THE FILL-INs?
        jne     short _KIO_S3           ; NO, TRY LAST TEST
        or      ah, ah                  ; AH = 0 IS SPECIAL CASE
        jz      short _KIO_USE          ; PASS THIS ON UNCHANGED
        jmp     short _KIO_DIS          ; THROW AWAY THE REST
_KIO_S3:
        cmp     al, 0E0h                ; IS IT AN EXTENSION OF A PREVIOUS ONE?
        ;jne    short _KIO_USE          ; NO, MUST BE A STANDARD CODE
        jne     short _kio_ret
        or      ah, ah                  ; AH = 0 IS SPECIAL CASE
        jz      short _KIO_USE          ; JUMP IF AH = 0
        xor     al, al                  ; CONVERT TO COMPATIBLE OUTPUT
        ;jmp    short _KIO_USE          ; PASS IT ON TO CALLER
_KIO_USE:
        ;clc                            ; CLEAR CARRY TO INDICATE GOOD CODE
        retn                            ; RETURN
_KIO_DIS:
```

```
        stc                             ; SET CARRY TO INDICATE DISCARD CODE
        retn                            ; RETURN

        ;-----  INCREMENT BUFFER POINTER ROUTINE -----
_K4:
        inc     ebx
        inc     ebx                     ; MOVE TO NEXT WORD IN LIST
        cmp     ebx, [BUFFER_END]       ; AT END OF BUFFER?
        ;jne    short _K5               ; NO, CONTINUE
        jb      short _K5
        mov     ebx, [BUFFER_START]     ; YES, RESET TO BUFFER BEGINNING
_K5:
        retn

; 20/02/2015
; 05/12/2014
; 26/08/2014
; KEYBOARD (HARDWARE) INTERRUPT -  IRQ LEVEL 1
; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
;
; Derived from "KB_INT_1" procedure of IBM "pc-at"
; rombios source code (06/10/1985)
; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)

;--------- 8042 COMMANDS -----------------------------------------------------
ENA_KBD         equ     0AEh    ; ENABLE KEYBOARD COMMAND
DIS_KBD         equ     0ADh    ; DISABLE KEYBOARD COMMAND
SHUT_CMD        equ     0FEh    ; CAUSE A SHUTDOWN COMMAND
;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
STATUS_PORT     equ     064h    ; 8042 STATUS PORT
INPT_BUF_FULL   equ     00000010b ; 1 = +INPUT BUFFER FULL
PORT_A          equ     060h    ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
;---------- 8042 KEYBOARD RESPONSE ---------------------------------------------
KB_ACK          equ     0FAh    ; ACKNOWLEDGE PROM TRANSMISSION
KB_RESEND       equ     0FEh    ; RESEND REQUEST
KB_OVER_RUN     equ     0FFh    ; OVER RUN SCAN CODE
;---------- KEYBOARD/LED COMMANDS ----------------------------------------------
KB_ENABLE       equ     0F4h            ; KEYBOARD ENABLE
LED_CMD         equ     0EDh            ; LED WRITE COMMAND
KB_TYPA_RD      equ     0F3h            ; TYPAMATIC RATE/DELAY COMMAND
;---------- KEYBOARD SCAN CODES ------------------------------------------------
NUM_KEY         equ     69              ; SCAN CODE FOR  NUMBER LOCK KEY
SCROLL_KEY      equ     70              ; SCAN CODE FOR  SCROLL LOCK KEY
ALT_KEY         equ     56              ; SCAN CODE FOR  ALTERNATE SHIFT KEY
CTL_KEY         equ     29              ; SCAN CODE FOR  CONTROL KEY
CAPS_KEY        equ     58              ; SCAN CODE FOR  SHIFT LOCK KEY
DEL_KEY         equ     83              ; SCAN CODE FOR  DELETE KEY
INS_KEY         equ     82              ; SCAN CODE FOR  INSERT KEY
LEFT_KEY        equ     42              ; SCAN CODE FOR  LEFT SHIFT
RIGHT_KEY       equ     54              ; SCAN CODE FOR  RIGHT SHIFT
SYS_KEY         equ     84              ; SCAN CODE FOR  SYSTEM KEY
;---------- ENHANCED KEYBOARD SCAN CODES ---------------------------------------
ID_1            equ     0ABh            ; 1ST ID CHARACTER FOR KBX
ID_2            equ     041h            ; 2ND ID CHARACTER FOR KBX
ID_2A           equ     054h            ; ALTERNATE 2ND ID CHARACTER FOR KBX
F11_M           equ     87              ; F11 KEY MAKE
F12_M           equ     88              ; F12 KEY MAKE
MC_E0           equ     224             ; GENERAL MARKER CODE
MC_E1           equ     225             ; PAUSE KEY MARKER CODE
;---------- FLAG EQUATES WITHIN @KB_FLAG----------------------------------------
RIGHT_SHIFT     equ     00000001b       ; RIGHT SHIFT KEY DEPRESSED
LEFT_SHIFT      equ     00000010b       ; LEFT SHIFT KEY DEPRESSED
CTL_SHIFT       equ     00000100b       ; CONTROL SHIFT KEY DEPRESSED
ALT_SHIFT       equ     00001000b       ; ALTERNATE SHIFT KEY DEPRESSED
```

```
SCROLL_STATE    equ     00010000b       ; SCROLL LOCK STATE IS ACTIVE
NUM_STATE       equ     00100000b       ; NUM LOCK STATE IS ACTIVE
CAPS_STATE      equ     01000000b       ; CAPS LOCK STATE IS ACTIVE
INS_STATE       equ     10000000b       ; INSERT STATE IS ACTIVE
;---------- FLAG EQUATES WITHIN @KB_FLAG_1 -----------------------------------
L_CTL_SHIFT     equ     00000001b       ; LEFT CTL KEY DOWN
L_ALT_SHIFT     equ     00000010b       ; LEFT ALT KEY DOWN
SYS_SHIFT       equ     00000100b       ; SYSTEM KEY DEPRESSED AND HELD
HOLD_STATE      equ     00001000b       ; SUSPEND KEY HAS BEEN TOGGLED
SCROLL_SHIFT    equ     00010000b       ; SCROLL LOCK KEY IS DEPRESSED
NUM_SHIFT       equ     00100000b       ; NUM LOCK KEY IS DEPRESSED
CAPS_SHIFT      equ     01000000b       ; CAPS LOCK KEY IS DEPRE55ED
INS_SHIFT       equ     10000000b       ; INSERT KEY IS DEPRESSED
;---------- FLAGS EQUATES WITHIN @KB_FLAG_2 ----------------------------------
KB_LEDS         equ     00000111b       ; KEYBOARD LED STATE BITS
;               equ     00000001b       ; SCROLL LOCK INDICATOR
;               equ     00000010b       ; NUM LOCK INDICATOR
;               equ     00000100b       ; CAPS LOCK INDICATOR
;               equ     00001000b       ; RESERVED (MUST BE ZERO)
KB_FA           equ     00010000b       ; ACKNOWLEDGMENT RECEIVED
KB_FE           equ     00100000b       ; RESEND RECEIVED FLAG
KB_PR_LED       equ     01000000b       ; MODE INDICATOR UPDATE
KB_ERR          equ     10000000b       ; KEYBOARD TRANSMIT ERROR FLAG
;----------- FLAGS EQUATES WITHIN @KB_FLAG_3 -----------------------------------
LC_E1           equ     00000001b       ; LAST CODE WAS THE E1 HIDDEN CODE
LC_E0           equ     00000010b       ; LAST CODE WAS THE E0 HIDDEN CODE
R_CTL_SHIFT     equ     00000100b       ; RIGHT CTL KEY DOWN
R_ALT_SHIFT     equ     00001000b       ; RIGHT ALT KEY DOWN
GRAPH_ON        equ     00001000b       ; ALT GRAPHICS KEY DOWN (WT ONLY)
KBX             equ     00010000b       ; ENHANCED KEYBOARD INSTALLED
SET_NUM_LK      equ     00100000b       ; FORCE NUM LOCK IF READ ID AND KBX
LC_AB           equ     01000000b       ; LAST CHARACTER WAS FIRST ID CHARACTER
RD_ID           equ     10000000b       ; DOING A READ ID (MUST BE BIT0)
;
;----------- INTERRUPT EQUATES -------------------------------------------------
EOI             equ     020h            ; END OF INTERRUPT COMMAND TO 8259
INTA00          equ     020h            ; 8259 PORT


kb_int:

; 05/12/2014
; 04/12/2014 (derivation from pc-xt-286 bios source code -1986-,
        ;             instead of pc-at bios - 1985-)
; 26/08/2014
;
; 03/06/86  KEYBOARD BIOS
;
;--- HARDWARE INT 09H -- (IRQ LEVEL 1)
----------------------------------------
;
;
;       KEYBOARD INTERRUPT ROUTINE
;
;
;
;--------------------------------------------------------------------------------
-

KB_INT_1:
        sti                             ; ENABLE INTERRUPTS
        ;push   ebp
        push    eax
        push    ebx
```

```
        push    ecx
        push    edx
        push    esi
        push    edi
        push    ds
        push    es
        cld                             ; FORWARD DIRECTION
        mov     ax, KDATA
        mov     ds, ax
        mov     es, ax
        ;
        ;-----  WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
        mov     al, DIS_KBD             ; DISABLE THE KEYBOARD COMMAND
        call    SHIP_IT                 ; EXECUTE DISABLE
        cli                             ; DISABLE INTERRUPTS
        mov     ecx, 10000h             ; SET MAXIMUM TIMEOUT
KB_INT_01:
        in      al, STATUS_PORT         ; READ ADAPTER STATUS
        test    al, INPT_BUF_FULL       ; CHECK INPUT BUFFER FULL STATUS BIT
        loopnz  KB_INT_01               ; WAIT FOR COMMAND TO BE ACCEPTED
        ;
        ;-----  READ CHARACTER FROM KEYBOARD INTERFACE
        in      al, PORT_A              ; READ IN THE CHARACTER
        ;
        ;-----  SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
        ;MOV    AH, 04FH                ; SYSTEM INTERCEPT - KEY CODE FUNCTION
        ;STC                            ; SET CY=1 (IN CASE OF IRET)
        ;INT    15H                     ; CASETTE CALL (AL)=KEY SCAN CODE
        ;                               ; RETURNS CY=1 FOR INVALID FUNCTION
        ;JC     KB_INT_02               ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
        ;JMP    K26                     ; EXIT IF SYSTEM HANDLES SCAN CODE
        ;                               ; EXIT HANDLES HARDWARE EOI AND ENABLE

        ;
        ;-----  CHECK FOR A RESEND COMMAND TO KEYBOARD
KB_INT_02:                              ;         (AL)= SCAN CODE
        sti                             ; ENABLE INTERRUPTS AGAIN
        cmp     al, KB_RESEND           ; IS THE INPUT A RESEND
        je      short KB_INT_4          ; GO IF RESEND
        ;
        ;-----  CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
        cmp     al, KB_ACK              ; IS THE INPUT AN ACKNOWLEDGE
        jne     short KB_INT_2          ; GO IF NOT
        ;
        ;-----  A COMMAND TO THE KEYBOARD WAS ISSUED
        cli                             ; DISABLE INTERRUPTS
        or      byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
        jmp     K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)
        ;
        ;-----  RESEND THE LAST BYTE
KB_INT_4:
        cli                             ; DISABLE INTERRUPTS
        or      byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
        jmp     K26                     ; RETURN IF NOT ACK RETURNED FOR DATA)
        ;
;-----  UPDATE MODE INDICATORS IF CHANGE IN STATE
KB_INT_2:
        push    ax                      ; SAVE DATA IN
        call    MAKE_LED                ; GO GET MODE INDICATOR DATA BYTE
        mov     bl, [KB_FLAG_2]         ; GET PREVIOUS BITS
        xor     bl, al                  ; SEE IF ANY DIFFERENT
        and     bl, KB_LEDS             ; ISOLATE INDICATOR BITS
        jz      short UP0               ; IF NO CHANGE BYPASS UPDATE
        call    SND_LED                 ; GO TURN ON MODE INDICATORS
```

```
UP0:
        pop     ax                      ; RESTORE DATA IN
;--------------------------------------------------------------------------
;       START OF KEY PROCESSING                                           ;
;--------------------------------------------------------------------------
        mov     ah, al                  ; SAVE SCAN CODE IN AH ALSO
        ;
        ;-----  TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
        cmp     al, KB_OVER_RUN         ; IS THIS AN OVERRUN CHAR
        je      K62                     ; BUFFER_FULL_BEEP
        ;
K16:
        mov     bh, [KB_FLAG_3]         ; LOAD FLAGS FOR TESTING
        ;
        ;-----  TEST TO SEE IF A READ_ID IS IN PROGRESS
        test    bh, RD_ID+LC_AB         ; ARE WE DOING A READ ID?
        jz      short NOT_ID            ; CONTINUE IF NOT
        jns     short TST_ID_2          ; IS THE RD_ID FLAG ON?
        cmp     al, ID_1                ; IS THIS THE 1ST ID CHARACTER?
        jne     short RST_RD_ID
        or      byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
RST_RD_ID:
        and     byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
        ;jmp    short ID_EX             ; AND EXIT
        jmp     K26
        ;
TST_ID_2:
        and     byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
        cmp     al, ID_2A               ; IS THIS THE 2ND ID CHARACTER?
        je      short KX_BIT            ; JUMP IF SO
        cmp     al, ID_2                ; IS THIS THE 2ND ID CHARACTER?
        ;jne    short ID_EX             ; LEAVE IF NOT
        jne     K26
        ;
        ;-----  A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
        test    bh, SET_NUM_LK          ; SHOULD WE SET NUM LOCK?
        jz      short KX_BIT            ; EXIT IF NOT
        or      byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
        call    SND_LED                 ; GO SET THE NUM LOCK INDICATOR
KX_BIT:
        or      byte [KB_FLAG_3], KBX   ; INDICATE ENHANCED KEYBOARD WAS FOUND
ID_EX:  jmp     K26                     ; EXIT
        ;
NOT_ID:
        cmp     al, MC_E0               ; IS THIS THE GENERAL MARKER CODE?
        jne     short TEST_E1
        or      byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
        ;jmp    short EXIT              ; THROW AWAY THIS CODE
        jmp     K26A
TEST_E1:
        cmp     al, MC_E1               ; IS THIS THE PAUSE KEY?
        jne     short NOT_HC
        or      byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
EXIT:   jmp     K26A                    ; THROW AWAY THIS CODE
        ;
NOT_HC:
        and     al, 07Fh                ; TURN OFF THE BREAK BIT
        test    bh, LC_E0               ; LAST CODE THE E0 MARKER CODE
        jz      short NOT_LC_E0         ; JUMP IF NOT
        ;
        mov     edi, _K6+6              ; IS THIS A SHIFT KEY?
        scasb
        ;je     short K16B              ; YES, THROW AWAY & RESET FLAG
        je      K26
```

```
        scasb
        jne     short K16A              ; NO, CONTINUE KEY PROCESSING
        ;jmp    short K16B
        jmp     K26
        ;
NOT_LC_E0:
        test    bh, LC_E1               ; LAST CODE THE E1 MARKER CODE?
        jz      short T_SYS_KEY         ; JUMP IF NOT
        mov     ecx, 4                  ; LENGHT OF SEARCH
        mov     edi, _K6+4              ; IS THIS AN ALT, CTL, OR SHIFT?
        repne   scasb                   ; CHECK IT
        ;je     short EXIT              ; THROW AWAY IF SO
        je      K26A
        ;
        cmp     al, NUM_KEY             ; IS IT THE PAUSE KEY?
        ;jne    short K16B              ; NO, THROW AWAY & RESET FLAG
        jne     K26
        test    ah, 80h                 ; YES, IS IT THE BREAK OF THE KEY?
        ;jnz    short K16B              ; YES, THROW THIS AWAY, TOO
        jnz     K26
        ; 20/02/2015
        test    byte [KB_FLAG_1],HOLD_STATE ;  NO, ARE WE PAUSED ALREADY?
        ;jnz    short K16B              ;  YES, THROW AWAY
        jnz     K26
        jmp     K39P                    ; NO, THIS IS THE REAL PAUSE STATE
        ;
        ;-----  TEST FOR SYSTEM KEY
T_SYS_KEY:
        cmp     al, SYS_KEY             ; IS IT THE SYSTEM KEY?
        jnz     short K16A              ; CONTINUE IF NOT
        ;
        test    ah, 80h                 ; CHECK IF THIS A BREAK CODE
        jnz     short K16C              ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
        ;
        test    byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
        ;jnz    short K16B              ; IF YES, DO NOT PROCESS SYSTEM
INDICATOR
        jnz     K26
        ;
        or      byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
        mov     al, EOI                 ; END OF INTERRUPT COMMAND
        out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                        ; INTERRUPT-RETURN-NO-EOI
        mov     al, ENA_KBD             ; INSURE KEYBOARD IS ENABLED
        call    SHIP_IT                 ; EXECUTE ENABLE
        ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
        ;MOV    AL, 8500H               ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
        ;STI                            ; MAKE SURE INTERRUPTS ENABLED
        ;INT    15H                     ; USER INTERRUPT
        jmp     K27A                    ; END PROCESSING
        ;
;K16B:  jmp     K26                     ; IGNORE SYSTEM KEY
        ;
K16C:
        and     byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
        mov     al, EOI                 ; END OF INTERRUPT COMMAND
        out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
                                        ; INTERRUPT-RETURN-NO-EOI
        ;MOV    AL, ENA_KBD             ; INSURE KEYBOARD IS ENABLED
        ;CALL   SHIP_IT                 ; EXECUTE ENABLE
        ;
        ;MOV    AX, 8501H               ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
        ;STI                            ; MAKE SURE INTERRUPTS ENABLED
        ;INT    15H                     ; USER INTERRUPT
```

```
        ;JMP    K27A                    ; INGONRE SYSTEM KEY

        ;
        jmp     K27                     ; IGNORE SYSTEM KEY
        ;
        ;-----  TEST FOR SHIFT KEYS
K16A:
        mov     bl, [KB_FLAG]           ; PUT STATE FLAGS IN BL
        mov     edi, _K6                ; SHIFT KEY TABLE offset
        mov     ecx, _K6L               ; LENGTH
        repne   scasb                   ; LOOK THROUGH THE TABLE FOR A MATCH
        mov     al, ah                  ; RECOVER SCAN CODE
        jne     K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND
        ;
        ;------ SHIFT KEY FOUND
K17:
        sub     edi, _K6+1              ; ADJUST PTR TO SCAN CODE MATCH
        mov     ah, [edi+_K7]           ; GET MASK INTO AH
        mov     cl, 2                   ; SETUP COUNT FOR FLAG SHIFTS
        test    al, 80h                 ; TEST FOR BREAK KEY
        jnz     K23                     ; JUMP OF BREAK
        ;
        ;-----  SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
K17C:
        cmp     ah, SCROLL_SHIFT
        jae     short K18               ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
        ;
        ;-----  PLAIN SHIFT KEY, SET SHIFT ON
        or      [KB_FLAG], ah           ; TURN ON SHIFT BIT
        test    al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
        ;jnz    short K17D              ; YES, MORE FLAGS TO SET
        jz      K26                     ; NO, INTERRUPT RETURN
K17D:
        test    bh, LC_E0               ; IS THIS ONE OF NEW KEYS?
        jz      short K17E              ; NO, JUMP
        or      [KB_FLAG_3], ah         ; SET BITS FOR RIGHT CTRL, ALT
        jmp     K26                     ; INTERRUPT RETURN
K17E:
        shr     ah, cl                  ; MOVE FLAG BITS TWO POSITIONS
        or      [KB_FLAG_1], ah         ; SET BITS FOR LEFT CTRL, ALT
        jmp     K26
        ;
        ;-----  TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
K18:                                    ; SHIFT-TOGGLE
        test    bl, CTL_SHIFT           ; CHECK CTL SHIFT STATE
        ;jz     short K18A              ; JUMP IF NOT CTL STATE
        jnz     K25                     ; JUMP IF CTL STATE
K18A:
        cmp     al, INS_KEY             ; CHECK FOR INSERT KEY
        jne     short K22               ; JUMP IF NOT INSERT KEY
        test    bl, ALT_SHIFT           ; CHECK FOR ALTERNATE SHIFT
        ;jz     short K18B              ; JUMP IF NOT ALTERNATE SHIFT
        jnz     K25                     ; JUMP IF ALTERNATE SHIFT
K18B:
        test    bh, LC_E0 ;20/02/2015   ; IS THIS NEW INSERT KEY?
        jnz     short K22               ; YES, THIS ONE'S NEVER A '0'
K19:
        test    bl, NUM_STATE           ; CHECK FOR BASE STATE
        jnz     short K21               ; JUMP IF NUM LOCK IS ON
        test    bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
        jz      short K22               ; JUMP IF BASE STATE
K20:                                    ; NUMERIC ZERO, NOT INSERT KEY
        mov     ah, al                  ; PUT SCAN CODE BACK IN AH
        jmp     K25                     ; NUMERAL '0', STNDRD. PROCESSING
```

```
K21:                                    ; MIGHT BE NUMERIC
        test    bl, LEFT_SHIFT+RIGHT_SHIFT
        jz      short K20               ; IS NUMERIC, STD. PROC.
        ;
K22:                                    ; SHIFT TOGGLE KEY HIT; PROCESS IT
        test    ah, [KB_FLAG_1]         ; IS KEY ALREADY DEPRESSED
        jnz     K26                     ; JUMP IF KEY ALREADY DEPRESSED
K22A:
        or      [KB_FLAG_1], ah         ; INDICATE THAT THE KEY IS DEPRESSED
        xor     [KB_FLAG], ah           ; TOGGLE THE SHIFT STATE
        ;
        ;-----  TOGGLE LED IF CAPS, NUM  OR SCROLL KEY DEPRESSED
        test    ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
        jz      short K22B              ; GO IF NOT
        ;
        push    ax                      ; SAVE SCAN CODE AND SHIFT MASK
        call    SND_LED                 ; GO TURN MODE INDICATORS ON
        pop     ax                      ; RESTORE SCAN CODE
K22B:
        cmp     al, INS_KEY             ; TEST FOR 1ST MAKE OF INSERT KEY
        jne     K26                     ; JUMP IF NOT INSERT KEY
        mov     ah, al                  ; SCAN CODE IN BOTH HALVES OF AX
        jmp     K28                     ; FLAGS UPDATED, PROC. FOR BUFFER
        ;
        ;-----  BREAK SHIFT FOUND
K23:                                    ; BREAK-SHIFT-FOUND
        cmp     ah, SCROLL_SHIFT        ; IS THIS A TOGGLE KEY
        not     ah                      ; INVERT MASK
        jae     short K24               ; YES, HANDLE BREAK TOGGLE
        and     [KB_FLAG], ah           ; TURN OFF SHIFT BIT
        cmp     ah, ~CTL_SHIFT          ; IS THIS ALT OR CTL?
        ja      short K23D              ; NO, ALL DONE
        ;
        test    bh, LC_E0               ; 2ND ALT OR CTL?
        jz      short K23A              ; NO, HANSLE NORMALLY
        and     [KB_FLAG_3], ah         ; RESET BIT FOR RIGHT ALT OR CTL
        jmp     short K23B              ; CONTINUE
K23A:
        sar     ah, cl                  ; MOVE THE MASK BIT TWO POSITIONS
        and     [KB_FLAG_1], ah         ; RESET BIT FOR LEFT ALT AND CTL
K23B:
        mov     ah, al                  ; SAVE SCAN CODE
        mov     al, [KB_FLAG_3]         ; GET RIGHT ALT & CTRL FLAGS
        shr     al, cl                  ; MOVE TO BITS 1 & 0
        or      al, [KB_FLAG_1]         ; PUT IN LEFT ALŞT & CTL FLAGS
        shl     al, cl                  ; MOVE BACK TO BITS 3 & 2
        and     al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
        or      [KB_FLAG], al           ; PUT RESULT IN THE REAL FLAGS
        mov     al, ah
K23D:
        cmp     al, ALT_KEY+80h         ; IS THIS ALTERNATE SHIFT RELEASE
        jne     short K26               ; INTERRUPT RETURN
        ;
        ;-----  ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
        mov     al, [ALT_INPUT]
        mov     ah, 0                   ; SCAN CODE OF 0
        mov     [ALT_INPUT], ah         ; ZERO OUT THE FIELD
        cmp     al, 0                   ; WAS THE INPUT = 0?
        je      short K26               ; INTERRUPT_RETURN
        jmp     K61                     ; IT WASN'T, SO PUT IN BUFFER
        ;
K24:                                    ; BREAK-TOGGLE
        and     [KB_FLAG_1], ah         ; INDICATE NO LONGER DEPRESSED
        jmp     short K26               ; INTERRUPT_RETURN
```

```
        ;
        ;-----  TEST FOR HOLD STATE
                                        ; AL, AH = SCAN CODE
K25:                                    ; NO-SHIFT-FOUND
        cmp     al, 80h                 ; TEST FOR BREAK KEY
        jae     short K26               ; NOTHING FOR BREAK CHARS FROM HERE ON
        test    byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
        jz      short K28               ; BRANCH AROUND TEST IF NOT
        cmp     al, NUM_KEY
        je      short K26               ; CAN'T END HOLD ON NUM_LOCK
        and     byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
        ;
K26:
        and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
K26A:                                   ; INTERRUPT-RETURN
        cli                             ; TURN OFF INTERRUPTS
        mov     al, EOI                 ; END OF INTERRUPT COMMAND
        out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
K27:                                    ; INTERRUPT-RETURN-NO-EOI
        mov     al, ENA_KBD             ; INSURE KEYBOARD IS ENABLED
        call    SHIP_IT                 ; EXECUTE ENABLE
K27A:
        cli                             ; DISABLE INTERRUPTS
        pop     es                      ; RESTORE REGISTERS
        pop     ds
        pop     edi
        pop     esi
        pop     edx
        pop     ecx
        pop     ebx
        pop     eax
        ;pop    ebp
        iret                            ; RETURN

        ;-----  NOT IN  HOLD STATE
K28:                                    ; NO-HOLD-STATE
        cmp     al, 88                  ; TEST FOR OUT-OF-RANGE SCAN CODES
        ja      short K26               ; IGNORE IF OUT-OF-RANGE
        ;
        test    bl, ALT_SHIFT           ; ARE WE IN ALTERNATE SHIFT
        ;jz     short K28A              ; IF NOT ALTERNATE
        jz      K38
        ;
        test    bh, KBX                 ; IS THIS THE ENCHANCED KEYBOARD?
        ;jz     short K29               ; NO, ALT STATE IS REAL
        jnz     K38                     ; YES, THIS IS PHONY ALT STATE
        ;                               ; DUE TO PRESSING SYSREQ
;K28A:  jmp     short K38
        ;
        ;-----  TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
K29:                                    ; TEST-RESET
        test    bl, CTL_SHIFT           ; ARE WE IN CONTROL SHIFT ALSO?
        jz      short K31               ; NO_RESET
        cmp     al, DEL_KEY             ; CTL-ALT STATE, TEST FOR DELETE KEY
        jne     short K31               ; NO_RESET, IGNORE
        ;
        ;-----  CTL-ALT-DEL HAS BEEN FOUND
        ; 26/08/2014
cpu_reset:
        ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
        ; Send FEh (system reset command) to the keyboard controller.
        mov     al, SHUT_CMD            ; SHUTDOWN COMMAND
        out     STATUS_PORT, al         ; SEND TO KEYBOARD CONTROL PORT
khere:
```

```
        hlt                             ; WAIT FOR 80286 RESET
        jmp     short khere             ; INSURE HALT

        ;
        ;-----  IN ALTERNATE SHIFT, RESET NOT FOUND
K31:                                    ; NO-RESET
        cmp     al, 57                  ; TEST FOR SPACE KEY
        jne     short K311              ; NOT THERE
        mov     al, ' '                 ; SET SPACE CHAR
        jmp     K57                     ; BUFFER_FILL
K311:
        cmp     al, 15                  ; TEST FOR TAB KEY
        jne     short K312              ; NOT THERE
        mov     ax, 0A500h              ; SET SPECIAL CODE FOR ALT-TAB
        jmp     K57                     ; BUFFER_FILL
K312:
        cmp     al, 74                  ; TEST FOR KEY PAD -
        je      K37B                    ; GO PROCESS
        cmp     al, 78                  ; TEST FOR KEY PAD +
        je      K37B                    ; GO PROCESS
        ;
        ;-----  LOOK FOR KEY PAD ENTRY
K32:                                    ; ALT-KEY-PAD
        mov     edi, K30                ; ALT-INPUT-TABLE offset
        mov     ecx, 10                 ; LOOK FOR ENTRY USING KEYPAD
        repne   scasb                   ; LOOK FOR MATCH
        jne     short K33               ; NO_ALT_KEYPAD
        test    bh, LC_E0               ; IS THIS ONE OF THE NEW KEYS?
        jnz     K37C                    ; YES, JUMP, NOT NUMPAD KEY
        sub     edi, K30+1              ; DI NOW HAS ENTRY VALUE
        mov     al, [ALT_INPUT]         ; GET THE CURRENT BYTE
        mov     ah, 10                  ; MULTIPLY BY 10
        mul     ah
        add     ax, di                  ; ADD IN THE LATEST ENTRY
        mov     [ALT_INPUT], al         ; STORE IT AWAY
;K32A:
        jmp     K26                     ; THROW AWAY THAT KEYSTROKE
        ;
        ;-----  LOOK FOR SUPERSHIFT ENTRY
K33:                                    ; NO-ALT-KEYPAD
        mov     byte [ALT_INPUT], 0     ; ZERO ANY PREVIOUS ENTRY INTO INPUT
        mov     ecx, 26                 ; (DI),(ES) ALREADY POINTING
        repne   scasb                   ; LOOK FOR MATCH IN ALPHABET
        je      short K37A              ; MATCH FOUND, GO FILLL THE BUFFER
        ;
        ;-----  LOOK FOR TOP ROW OF ALTERNATE SHIFT
K34:                                    ; ALT-TOP-ROW
        cmp     al, 2                   ; KEY WITH '1' ON IT
        jb      short K37B              ; MUST BE ESCAPE
        cmp     al, 13                  ; IS IT IN THE REGION
        ja      short K35               ; NO, ALT SOMETHING ELSE
        add     ah, 118                 ; CONVERT PSEUDO SCAN CODE TO RANGE
        jmp     short K37A              ; GO FILL THE BUFFER
        ;
        ;-----  TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
K35:                                    ; ALT-FUNCTION
        cmp     al, F11_M               ; IS IT F11?
        jb      short K35A ; 20/02/2015 ; NO, BRANCH
        cmp     al, F12_M               ; IS IT F12?
        ja      short K35A ; 20/02/2015 ; NO, BRANCH
        add     ah, 52                  ; CONVERT TO PSEUDO SCAN CODE
        jmp     short K37A              ; GO FILL THE BUFFER
K35A:
        test    bh, LC_E0               ; DO WE HAVE ONE OF THE NEW KEYS?
```

```
        jz      short K37               ; NO, JUMP
        cmp     al, 28                  ; TEST FOR KEYPAD ENTER
        jne     short K35B              ; NOT THERE
        mov     ax, 0A600h              ; SPECIAL CODE
        jmp     K57                     ; BUFFER FILL
K35B:
        cmp     al, 83                  ; TEST FOR DELETE KEY
        je      short K37C              ; HANDLE WITH OTHER EDIT KEYS
        cmp     al, 53                  ; TEST FOR KEYPAD /
        ;jne    short K32A              ; NOT THERE, NO OTHER E0 SPECIALS
        jne     K26
        mov     ax, 0A400h              ; SPECIAL CODE
        jmp     K57                     ; BUFFER FILL
K37:
        cmp     al, 59                  ; TEST FOR FUNCTION KEYS (F1)
        jb      short K37B              ; NO FN, HANDLE W/OTHER EXTENDED
        cmp     al, 68                  ; IN KEYPAD REGION?
        ;ja     short K32A              ; IF SO, IGNORE
        ja      K26
        add     ah, 45                  ; CONVERT TO PSEUDO SCAN CODE
K37A:
        mov     al, 0                   ; ASCII CODE OF ZERO
        jmp     K57                     ; PUT IT IN THE BUFFER
K37B:
        mov     al, 0F0h                ; USE SPECIAL ASCII CODE
        jmp     K57                     ; PUT IT IN THE BUFFER
K37C:
        add     al, 80                  ; CONVERT SCAN CODE (EDIT KEYS)
        mov     ah, al                  ; (SCAN CODE NOT IN AH FOR INSERT)
        jmp     short K37A              ; PUT IT IN THE BUFFER
        ;
        ;-----  NOT IN ALTERNATE SHIFT
K38:                                    ; NOT-ALT-SHIFT
                                        ; BL STILL HAS SHIFT FLAGS
        test    bl, CTL_SHIFT           ; ARE WE IN CONTROL SHIFT?
        ;jnz    short K38A              ; YES, START PROCESSING
        jz      K44                     ; NOT-CTL-SHIFT
        ;
        ;-----  CONTROL SHIFT, TEST SPECIAL CHARACTERS
        ;-----  TEST FOR BREAK
K38A:
        cmp     al, SCROLL_KEY          ; TEST FOR BREAK
        jne     short K39               ; JUMP, NO-BREAK
        test    bh, KBX                 ; IS THIS THE ENHANCED KEYBOARD?
        jz      short K38B              ; NO, BREAK IS VALID
        test    bh, LC_E0               ; YES, WAS LAST CODE AN E0?
        jz      short K39               ; NO-BREAK, TEST FOR PAUSE
K38B:
        mov     ebx, [BUFFER_HEAD]      ; RESET BUFFER TO EMPTY
        mov     [BUFFER_TAIL], ebx
        mov     byte [BIOS_BREAK], 80h  ; TURN ON BIOS_BREAK BIT
        ;
        ;-----  ENABLE KEYBOARD
        mov     al, ENA_KBD             ; ENABLE KEYBOARD
        call    SHIP_IT                 ; EXECUTE ENABLE
        ;
        ; CTRL+BREAK code here !!!
        ;INT    1BH                     ; BREAK INTERRUPT VECTOR
        ;
        sub     ax, ax                  ; PUT OUT DUMMY CHARACTER
        jmp     K57                     ; BUFFER_FILL
        ;
        ;-----  TEST FOR PAUSE
K39:                                    ; NO_BREAK
```

```
        test    bh, KBX                 ; IS THIS THE ENHANCED KEYBOARD?
        jnz     short K41               ; YES, THEN THIS CAN'T BE PAUSE
        cmp     al, NUM_KEY             ; LOOK FOR PAUSE KEY
        jne     short K41               ; NO-PAUSE
K39P:
        or      byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
        ;
        ;-----  ENABLE KEYBOARD
        mov     al, ENA_KBD             ; ENABLE KEYBOARD
        call    SHIP_IT                 ; EXECUTE ENABLE
K39A:
        mov     al, EOI                 ; END OF INTERRUPT TO CONTROL PORT
        out     20h, al ;out INTA00, al ; ALLOW FURTHER KEYSTROKE INTERRUPTS
        ;
        ;-----  DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
        cmp     byte [CRT_MODE], 7      ; IS THIS BLACK AND WHITE CARD
        je      short K40               ; YES, NOTHING TO DO
        mov     dx, 03D8h               ; PORT FOR COLOR CARD
        mov     al, [CRT_MODE_SET]      ; GET THE VALUE OF THE CURRENT MODE
        out     dx, al                  ; SET THE CRT MODE, SO THAT CRT IS ON
        ;
K40:                                    ; PAUSE-LOOP
        test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
        jnz     short K40               ; LOOP UNTIL FLAG TURNED OFF
        ;
        jmp     K27                     ; INTERRUPT_RETURN_NO_EOI
        ;
        ;-----  TEST SPECIAL CASE KEY 55
K41:                                    ; NO-PAUSE
        cmp     al, 55                  ; TEST FOR */PRTSC KEY
        jne     short K42               ; NOT-KEY-55
        test    bh, KBX                 ; IS THIS THE ENHANCED KEYBOARD?
        jz      short K41A              ; NO, CTL-PRTSC IS VALID
        test    bh, LC_E0               ; YES, WAS LAST CODE AN E0?
        jz      short K42B              ; NO, TRANSLATE TO A FUNCTION
K41A:
        mov     ax, 114*256             ; START/STOP PRINTING SWITCH
        jmp     K57                     ; BUFFER_FILL
        ;
        ;-----  SET UP TO TRANSLATE CONTROL SHIFT
K42:                                    ; NOT-KEY-55
        cmp     al, 15                  ; IS IT THE TAB KEY?
        je      short K42B              ; YES, XLATE TO FUNCTION CODE
        cmp     al, 53                  ; IS IT THE / KEY?
        jne     short K42A              ; NO, NO MORE SPECIAL CASES
        test    bh, LC_E0               ; YES, IS IT FROM THE KEY PAD?
        jz      short K42A              ; NO, JUST TRANSLATE
        mov     ax, 9500h               ; YES, SPECIAL CODE FOR THIS ONE
        jmp     K57                     ; BUFFER FILL
K42A:
        ;;mov   ebx, _K8                ; SET UP TO TRANSLATE CTL
        cmp     al, 59                  ; IS IT IN CHARACTER TABLE?
        ;jb     short K45F              ; YES, GO TRANSLATE CHAR
        ;;jb    K56 ; 20/02/2015
        ;;jmp   K64 ; 20/02/2015
K42B:
        mov     ebx, _K8                ; SET UP TO TRANSLATE CTL
        jb      K56 ;; 20/02/2015
        jmp     K64
        ;
        ;-----  NOT IN CONTROL SHIFT
K44:                                    ; NOT-CTL-SHIFT
        cmp     al, 55                  ; PRINT SCREEN KEY?
        jne     short K45               ; NOT PRINT SCREEN
```

```
        test    bh, KBX                 ; IS THIS ENHANCED KEYBOARD?
        jz      short K44A              ; NO, TEST FOR SHIFT STATE
        test    bh, LC_E0               ; YES, LAST CODE A MARKER?
        jnz     short K44B              ; YES, IS PRINT SCREEN
        jmp     short K45C              ; NO, TRANSLATE TO '*' CHARACTER
K44A:
        test    bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
        jz      short K45C              ; NO, TRANSLATE TO '*' CHARACTER
        ;
        ;-----  ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
K44B:
        mov     al, ENA_KBD             ; INSURE KEYBOARD IS ENABLED
        call    SHIP_IT                 ; EXECUTE ENABLE
        mov     al, EOI                 ; END OF CURRENT INTERRUPT
        out     20h, al ;out INTA00, al ; SO FURTHER THINGS CAN HAPPEN
        ; Print Screen !!!              ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
        ;PUSH   BP                      ; SAVE POINTER
        ;INT    5H                      ; ISSUE PRINT SCREEN INTERRUPT
        ;POP    BP                      ; RESTORE POINTER
        and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
        jmp     K27                     ; GO BACK WITHOUT EOI OCCURRING
        ;
        ;-----  HANDLE IN-CORE KEYS
K45:                                    ; NOT-PRINT-SCREEN
        cmp     al, 58                  ; TEST FOR IN-CORE AREA
        ja      short K46               ; JUMP IF NOT
        cmp     al, 53                  ; IS THIS THE '/' KEY?
        jne     short K45A              ; NO, JUMP
        test    bh, LC_E0               ; WAS THE LAST CODE THE MARKER?
        jnz     short K45C              ; YES, TRANSLATE TO CHARACTER
K45A:
        mov     ecx, 26                 ; LENGHT OF SEARCH
        mov     edi, K30+10             ; POINT TO TABLE OF A-Z CHARS
        repne   scasb                   ; IS THIS A LETTER KEY?
                ; 20/02/2015
        jne     short K45B              ; NO, SYMBOL KEY
        ;
        test    bl, CAPS_STATE          ; ARE WE IN CAPS_LOCK?
        jnz     short K45D              ; TEST FOR SURE
K45B:
        test    bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
        jnz     short K45E              ; YES, UPPERCASE
                                        ; NO, LOWERCASE
K45C:
        mov     ebx, K10                ; TRANSLATE TO LOWERCASE LETTERS
        jmp     short K56
K45D:                                   ; ALMOST-CAPS-STATE
        test    bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
        jnz     short K45C              ; SHIFTED TEMP OUT OF CAPS STATE
K45E:
        mov     ebx, K11                ; TRANSLATE TO UPPER CASE LETTERS
K45F:   jmp     short K56
        ;
        ;-----  TEST FOR KEYS F1 - F10
K46:                                    ; NOT IN-CORE AREA
        cmp     al, 68                  ; TEST FOR F1 - F10
        ;ja     short K47               ; JUMP IF NOT
        ;jmp    short K53               ; YES, GO DO FN KEY PROCESS

        jna     short K53
        ;
        ;-----  HANDLE THE NUMERIC PAD KEYS
K47:                                    ; NOT F1 - F10
        cmp     al, 83                  ; TEST NUMPAD KEYS
```

```
        ja      short K52               ; JUMP IF NOT
        ;
        ;-----  KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
K48:
        cmp     al , 74                 ; SPECIAL CASE FOR MINUS
        je      short K45E              ; GO TRANSLATE
        cmp     al , 78                 ; SPECIAL CASE FOR PLUS
        je      short K45E              ; GO TRANSLATE
        test    bh, LC_E0               ; IS THIS ONE OFTHE NEW KEYS?
        jnz     short K49               ; YES, TRANSLATE TO BASE STATE
        ;
        test    bl, NUM_STATE           ; ARE WE IN NUM LOCK
        jnz     short K50               ; TEST FOR SURE
        test    bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
        ;jnz    short K51               ; IF SHIFTED, REALLY NUM STATE
        jnz     short K45E
        ;
        ;-----  BASE CASE FOR KEYPAD
K49:
        cmp     al, 76                  ; SPECIAL CASE FOR BASE STATE 5
        jne     short K49A              ; CONTINUE IF NOT KEYPAD 5
        mov     al, 0F0h                ; SPECIAL ASCII CODE
        jmp     short K57               ; BUFFER FILL
K49A:
        mov     ebx, K10                ; BASE CASE TABLE
        jmp     short K64               ; CONVERT TO PSEUDO SCAN
        ;
        ;-----  MIGHT BE NUM LOCK, TEST SHIFT STATUS
K50:                                    ; ALMOST-NUM-STATE
        test    bl, LEFT_SHIFT+RIGHT_SHIFT
        jnz     short K49               ; SHIFTED TEMP OUT OF NUM STATE
K51:    jmp     short K45E              ; REALLY NUM STATE
        ;
        ;-----  TEST FOR THE NEW KEYS ON WT KEYBOARDS
K52:                                    ; NOT A NUMPAD KEY
        cmp     al, 86                  ; IS IT THE NEW WT KEY?
        ;jne    short K53               ; JUMP IF NOT
        ;jmp    short K45B              ; HANDLE WITH REST OF LETTER KEYS
        je      short K45B
        ;
        ;-----  MUST BE F11 OR F12
K53:                                    ; F1 - F10 COME HERE, TOO
        test    bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
        jz      short K49               ; JUMP, LOWER CASE PSEUDO SC'S
                ; 20/02/2015
        mov     ebx, K11                ; UPPER CASE PSEUDO SCAN CODES
        jmp     short K64               ; TRANSLATE SCAN
        ;
        ;-----  TRANSLATE THE CHARACTER
K56:                                    ; TRANSLATE-CHAR
        dec     al                      ; CONVERT ORIGIN
        xlat                            ; CONVERT THE SCAN CODE TO ASCII
        test    byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
        jz      short K57               ; NO, GO FILL BUFFER
        mov     ah, MC_E0               ; YES, PUT SPECIAL MARKER IN AH
        jmp     short K57               ; PUT IT INTO THE BUFFER
        ;
        ;-----  TRANSLATE SCAN FOR PSEUDO SCAN CODES
K64:                                    ; TRANSLATE-SCAN-ORGD
        dec     al                      ; CONVERT ORIGIN
        xlat                            ; CTL TABLE SCAN
        mov     ah, al                  ; PUT VALUE INTO AH
        mov     al, 0                   ; ZERO ASCII CODE
        test    byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
```

```
        jz      short K57               ; NO, GO FILL BUFFER
        mov     al, MC_E0               ; YES, PUT SPECIAL MARKER IN AL
        ;
        ;-----  PUT CHARACTER INTO BUFFER
K57:                                    ; BUFFER_FILL
        cmp     al, -1                  ; IS THIS AN IGNORE CHAR
        ;je     short K59               ; YES, DO NOTHING WITH IT
        je      K26                     ; YES, DO NOTHING WITH IT
        cmp     ah, -1                  ; LOOK FOR -1 PSEUDO SCAN
        ;jne    short K61               ; NEAR_INTEERUPT_RETURN
        je      K26                     ; INTERRUPT_RETURN
;K59:                                   ; NEAR_INTEERUPT_RETURN
;       jmp     K26                     ; INTERRUPT_RETURN
K61:                                    ; NOT-CAPS-STATE
        mov     ebx, [BUFFER_TAIL]      ; GET THE END POINTER TO THE BUFFER
        mov     esi, ebx                ; SAVE THE VALUE
        call    _K4                     ; ADVANCE THE TAIL
        cmp     ebx, [BUFFER_HEAD]      ; HAS THE BUFFER WRAPPED AROUND
        je      short K62               ; BUFFER_FULL_BEEP
        mov     [esi], ax               ; STORE THE VALUE
        mov     [BUFFER_TAIL], ebx      ; MOVE THE POINTER UP
        jmp     K26
        ;;cli                           ; TURN OFF INTERRUPTS
        ;;mov   al, EOI                 ; END OF INTERRUPT COMMAND
        ;;out   INTA00, al              ; SEND COMMAND TO INTERRUPT CONTROL PORT
        ;MOV    AL, ENA_KBD             ; INSURE KEYBOARD IS ENABLED
        ;CALL   SHIP_IT                 ; EXECUTE ENABLE
        ;MOV    AX, 9102H               ; MOVE IN POST CODE & TYPE
        ;INT    15H                     ; PERFORM OTHER FUNCTION
        ;;and   byte [KB_FLAG_3],~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
        ;JMP    K27A                    ; INTERRUPT_RETURN
        ;;jmp   K27
        ;
        ;-----  BUFFER IS FULL SOUND THE BEEPER
K62:
        mov     al, EOI                 ; ENABLE INTERRUPT CONTROLLER CHIP
        out     INTA00, al
        mov     cx, 678                 ; DIVISOR FOR 1760 HZ
        mov     bl, 4                   ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
        call    beep                    ; GO TO COMMON BEEP HANDLER
        jmp     K27                     ; EXIT

SHIP_IT:

;--------------------------------------------------------------------------------
--
        ; SHIP_IT
        ;       THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
        ;       TO THE KEYBOARD CONTROLLER.

;--------------------------------------------------------------------------------
--
        ;
        push    ax                      ; SAVE DATA TO SEND

        ;-----  WAIT FOR COMMAND TO ACCEPTED
        cli                             ; DISABLE INTERRUPTS TILL DATA SENT
        ; xor   ecx, ecx                ; CLEAR TIMEOUT COUNTER
        mov     ecx, 10000h
S10:
        in      al, STATUS_PORT         ; READ KEYBOARD CONTROLLER STATUS
        test    al, INPT_BUF_FULL       ; CHECK FOR ITS INPUT BUFFER BUSY
        loopnz  S10                     ; WAIT FOR COMMAND TO BE ACCEPTED
```

```
        pop     ax                      ; GET DATA TO SEND
        out     STATUS_PORT, al         ; SEND TO KEYBOARD CONTROLLER
        sti                             ; ENABLE INTERRUPTS AGAIN
        retn                            ; RETURN TO CALLER

SND_DATA:
        ;
--------------------------------------------------------------------------------
        ; SND_DATA
        ;       THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
        ;       TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
        ;       HANDLES ANY RETRIES IF REQUIRED
        ;
--------------------------------------------------------------------------------
        ;
        push    ax                      ; SAVE REGISTERS
        push    bx
        push    ecx
        mov     bh, al                  ; SAVE TRANSMITTED BYTE FOR RETRIES
        mov     bl, 3                   ; LOAD RETRY COUNT
SD0:
        cli                             ; DISABLE INTERRUPTS
        and     byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
        ;
        ;-----  WAIT FOR COMMAND TO BE ACCEPTED
        mov     ecx, 10000h             ; MAXIMUM WAIT COUNT
SD5:
        in      al, STATUS_PORT         ; READ KEYBOARD PROCESSOR STATUS PORT
        test    al, INPT_BUF_FULL       ; CHECK FOR ANY PENDING COMMAND
        loopnz  SD5                     ; WAIT FOR COMMAND TO BE ACCEPTED
        ;
        mov     al, bh                  ; REESTABLISH BYTE TO TRANSMIT
        out     PORT_A, al              ; SEND BYTE
        sti                             ; ENABLE INTERRUPTS
        ;mov    cx, 01A00h              ; LOAD COUNT FOR 10 ms+
        mov     ecx, 0FFFFh
SD1:
        test    byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
        jnz     short SD3               ; IF SET, SOMETHING RECEIVED GO PROCESS
        loop    SD1                     ; OTHERWISE WAIT
SD2:
        dec     bl                      ; DECREMENT RETRY COUNT
        jnz     short SD0               ; RETRY TRANSMISSION
        or      byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
        jmp     short SD4               ; RETRIES EXHAUSTED FORGET TRANSMISSION
SD3:
        test    byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
        jz      short SD2               ; IF NOT, GO RESEND
SD4:
        pop     ecx                     ; RESTORE REGISTERS
        pop     bx
        pop     ax
        retn                            ; RETURN, GOOD TRANSMISSION

SND_LED:
        ;
--------------------------------------------------------------------------------
        ; SND_LED
        ;       THIS ROUTINES TURNS ON THE MODE INDICATORS.
        ;
```

```
;--------------------------------------------------------------------------
---
        ;
        cli                             ; TURN OFF INTERRUPTS
        test    byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
        jnz     short SL1               ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
        ;
        or      byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
        mov     al, EOI                 ; END OF INTERRUPT COMMAND
        out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
        jmp     short SL0               ; GO SEND MODE INDICATOR COMMAND
SND_LED1:
        cli                             ; TURN OFF INTERRUPTS
        test    byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
        jnz     short SL1               ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
        ;
        or      byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
SL0:
        mov     al, LED_CMD             ; LED CMD BYTE
        call    SND_DATA                ; SEND DATA TO KEYBOARD
        cli
        call    MAKE_LED                ; GO FORM INDICATOR DATA BYTE
        and     byte [KB_FLAG_2], 0F8h  ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
        or      byte [KB_FLAG_2], al    ; SAVE PRESENT INDICATORS FOR NEXT TIME
        test    byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
        jnz     short SL2               ; IF SO, BYPASS SECOND BYTE TRANSMISSION
        ;
        call    SND_DATA                ; SEND DATA TO KEYBOARD
        cli                             ; TURN OFF INTERRUPTS
        test    byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
        jz      short SL3               ; IF NOT, DON'T SEND AN ENABLE COMMAND
SL2:
        mov     al, KB_ENABLE           ; GET KEYBOARD CSA ENABLE COMMAND
        call    SND_DATA                ; SEND DATA TO KEYBOARD
        cli                             ; TURN OFF INTERRUPTS
SL3:
        and     byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
SL1:                                    ; UPDATE AND TRANSMIT ERROR FLAG
        sti                             ; ENABLE INTERRUPTS
        retn                            ; RETURN TO CALLER

MAKE_LED:

;--------------------------------------------------------------------------
--
        ; MAKE_LED
        ;       THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
        ;       THE MODE INDICATORS.

;--------------------------------------------------------------------------
--
        ;
        ;push   cx                      ; SAVE CX
        mov     al, byte [KB_FLAG]      ; GET CAPS & NUM LOCK INDICATORS
        and     al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
        ;mov    cl, 4                   ; SHIFT COUNT
        ;rol    al, cl                  ; SHIFT BITS OVER TO TURN ON INDICATORS
        rol     al, 4 ; 20/02/2015
        and     al, 07h                 ; MAKE SURE ONLY MODE BITS ON
        ;pop    cx
        retn                            ; RETURN TO CALLER

;--------------------------------------------------------------------------
--
```

```
;       KEY IDENTIFICATION SCAN TABLES
;-------------------------------------------------------------------------------
--

;-----  TABLES FOR ALT CASE ------------
;-----  ALT-INPUT-TABLE
K30:    db      82,79,80,81,75
        db      76,77,71,72,73          ; 10 NUMBER ON KEYPAD
;-----  SUPER-SHIFT-TABLE
        db      16,17,18,19,20,21       ; A-Z TYPEWRITER CHARS
        db      22,23,24,25,30,31
        db      32,33,34,35,36,37
        db      38,44,45,46,47,48
        db      49,50

;-----  TABLE OF SHIFT KEYS AND MASK VALUES
;-----  KEY_TABLE
_K6:    db      INS_KEY                 ; INSERT KEY
        db      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
        db      LEFT_KEY,RIGHT_KEY
_K6L    equ     $-_K6

;-----  MASK_TABLE
_K7:    db      INS_SHIFT               ; INSERT MODE SHIFT
        db      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
        db      LEFT_SHIFT,RIGHT_SHIFT

;-----  TABLES FOR CTRL CASE            ;---- CHARACTERS ------
_K8:    db      27,-1,0,-1,-1,-1        ; Esc, 1, 2, 3, 4, 5
        db      30,-1,-1,-1,-1,31       ; 6, 7, 8, 9, 0, -
        db      -1,127,-1,17,23,5       ; =, Bksp, Tab, Q, W, E
        db      18,20,25,21,9,15        ; R, T, Y, U, I, O
        db      16,27,29,10,-1,1        ; P, [, ], Enter, Ctrl, A
        db      19,4,6,7,8,10           ; S, D, F, G, H, J
        db      11,12,-1,-1,-1,-1       ; K, L, :, ', `, LShift
        db      28,26,24,3,22,2         ; Bkslash, Z, X, C, V, B
        db      14,13,-1,-1,-1,-1       ; N, M, ,, ., /, RShift
        db      150,-1,' ',-1           ; *, ALT, Spc, CL
        ;                               ;----- FUNCTIONS ------
        db      94,95,96,97,98,99       ; F1 - F6
        db      100,101,102,103,-1,-1   ; F7 - F10, NL, SL
        db      119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
        db      116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
        db      147,-1,-1,-1,137,138    ; Del, SysReq, Undef, WT, F11, F12

;-----  TABLES FOR LOWER CASE ----------
K10:    db      27,'1234567890-=',8,9
        db      'qwertyuiop[]',13,-1,'asdfghjkl;',39
        db      96,-1,92,'zxcvbnm,./',-1,'*',-1,' ',-1
;-----  LC TABLE SCAN
        db      59,60,61,62,63          ; BASE STATE OF F1 - F10
        db      64,65,66,67,68
        db      -1,-1                   ; NL, SL

;-----  KEYPAD TABLE
K15:    db      71,72,73,-1,75,-1       ; BASE STATE OF KEYPAD KEYS
        db      77,-1,79,80,81,82,83
        db      -1,-1,92,133,134        ; SysRq, Undef, WT, F11, F12

;-----  TABLES FOR UPPER CASE ----------
K11:    db      27,'!@#$%',94,'&*()_+',8,0
        db      'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'
        db      126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
;-----  UC TABLE SCAN
```

```
K12:    db      84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
        db      89,90,91,92,93
        db      -1,-1                   ; NL, SL

;-----  NUM STATE TABLE
K14:    db      '789-456+1230.'         ; NUMLOCK STATE OF KEYPAD KEYS
        ;
        db      -1,-1,124,135,136       ; SysRq, Undef, WT, F11, F12

Align   4
;--------------------------------------
;       VIDEO DISPLAY DATA AREA         ;
;--------------------------------------
CRT_MODE        db      3       ; CURRENT DISPLAY MODE (TYPE)
CRT_MODE_SET    db      29h     ; CURRENT SETTING OF THE 3X8 REGISTER
                                ; (29h default setting for video mode 3)
                                ; Mode Select register Bits
                                ;   BIT 0 - 80x25 (1), 40x25 (0)
                                ;   BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
                                ;   BIT 2 - COLOR (0), BW (1)
                                ;   BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
                                ;   BIT 4 - 640x200 B&W Graphics Mode (1)
                                ;   BIT 5 - ALPHA mode BLINKING (1)
                                ;   BIT 6, 7 - Not Used

; Mode 0 - 2Ch = 101100b        ; 40x25 text, 16 gray colors
; Mode 1 - 28h = 101000b        ; 40x25 text, 16 fore colors, 8 back colors
; Mode 2 - 2Dh = 101101b        ; 80x25 text, 16 gray colors
; MODE 3 - 29h = 101001b        ; 80x25 text, 16 fore color, 8 back color
; Mode 4 - 2Ah = 101010b        ; 320x200 graphics, 4 colors
; Mode 5 - 2Eh = 101110b        ; 320x200 graphics, 4 gray colors
; Mode 6 - 1Eh = 011110b        ; 640x200 graphics, 2 colors
; Mode 7 - 29h = 101001b        ; 80x25 text, black & white colors
; Mode & 37h = Video signal OFF


; 26/08/2014
; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
; Derived from IBM "pc-at"
; rombios source code (06/10/1985)
; 'dseg.inc'

;--------------------------------------;
;       SYSTEM DATA AREA                ;
;--------------------------------------
BIOS_BREAK      db      0               ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED

;--------------------------------------
;       KEYBOARD DATA AREAS             ;
;--------------------------------------

KB_FLAG         db      0               ; KEYBOARD SHIFT STATE AND STATUS FLAGS
KB_FLAG_1       db      0               ; SECOND BYTE OF KEYBOARD STATUS
KB_FLAG_2       db      0               ; KEYBOARD LED FLAGS
KB_FLAG_3       db      0               ; KEYBOARD MODE STATE AND TYPE FLAGS
ALT_INPUT       db      0               ; STORAGE FOR ALTERNATE KEY PAD ENTRY
BUFFER_START    dd      KB_BUFFER       ; OFFSET OF KEYBOARD BUFFER START
BUFFER_END      dd      KB_BUFFER + 32  ; OFFSET OF END OF BUFFER
BUFFER_HEAD     dd      KB_BUFFER       ; POINTER TO HEAD OF KEYBOARD BUFFER
BUFFER_TAIL     dd      KB_BUFFER       ; POINTER TO TAIL OF KEYBOARD BUFFER
; ------        HEAD = TAIL     INDICATES THAT THE BUFFER IS EMPTY
KB_BUFFER       times   16 dw 0         ; ROOM FOR 16 SCAN CODE ENTRIES

; 28/08/2014
```

```
error_code: dd 0
; 29/08/2014
FaultOffset: dd 0

; 02/09/2014
; 30/08/2014
; VIDEO FUNCTIONS
; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)

write_tty:
        ; 02/09/2014
        ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
        ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
        ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
        ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
        ;
        ; INPUT -> AH = Color (Forecolor, Backcolor)
        ;          AL = Character to be written
        ;          EBX = Video Page (0 to 7)
        ;          (BH = 0 --> Video Mode 3)

RVRT    equ     00001000b       ; VIDEO VERTICAL RETRACE BIT
RHRZ    equ     00000001b       ; VIDEO HORIZONTAL RETRACE BIT

; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
;
; 06/10/85  VIDEO DISPLAY BIOS
;
;--- WRITE_TTY
--------------------------------------------------------------------
;
:
;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
:
;   VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
:
;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
:
;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
:
;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
:
;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
:
;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
:
;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
:
;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
:
;   LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
:
;   THE 0 COLOR IS USED.
:
;   ENTRY --
:
;     (AH) = CURRENT CRT MODE
:
;     (AL) = CHARACTER TO BE WRITTEN
:
;           NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE
:
;           HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS
```

```
;
;     (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE
;
;   EXIT --
;
;     ALL REGISTERS SAVED
;
;--------------------------------------------------------------------------------
-

        cli
        ;
        ; READ CURSOR (04/12/2013)
        ; Retro UNIX 386 v1 Modifications: 30/08/2014
        or      bh, bh
        jnz     beeper
        ; 01/09/2014
        cmp     byte [CRT_MODE], 3
        je      short m3
        ;
        call    set_mode
m3:
        xor     esi, esi
        mov     si, bx
        shl     si, 1
        add     esi, cursor_posn
        mov     dx, word [esi]
        ;
        ; dx now has the current cursor position
        ;
        cmp     al, 0Dh         ; is it carriage return or control character
        jbe     short u8
        ;
        ; write the char to the screen
u0:
        ; ah = attribute/color
        ; al = character
        ; bl = video page number (0 to 7)
        ; bh = 0
        ;
        call    write_c_current
        ;
        ; position the cursor for next char
        inc     dl              ; next column
        ;cmp    dl, byte [CRT_COLS]
        cmp     dl, 80          ; test for column overflow
        jne     set_cpos
        mov     dl, 0           ; column = 0
u10:                            ; (line feed found)
        cmp     dh, 25-1        ; check for last row
        jb      short u6
        ;
        ; scroll required
u1:
        ; SET CURSOR POSITION (04/12/2013)
        call    set_cpos
        ;
        ; determine value to fill with during scroll
u2:
        ; READ_AC_CURRENT               :
        ;   THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER
        ;    AT THE CURRENT CURSOR POSITION
        ;
        ; INPUT
```

```
        ;       (AH) = CURRENT CRT MODE
        ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
        ;       (DS) = DATA SEGMENT
        ;       (ES) = REGEN SEGMENT
        ; OUTPUT
        ;       (AL) = CHARACTER READ
        ;       (AH) = ATTRIBUTE READ
        ;
        ; mov   ah, byte [CRT_MODE] ; move current mode into ah
        ;
        ; bl = video page number
        ;
        call    find_position   ; get regen location and port address
        ; dx = status port
        ; esi = cursor location/address
p11:
        sti                     ; enable interrupts
        nop                     ; allow for small interupts window
        cli                     ; blocks interrupts for single loop
        in      al, dx          ; get status from adapter
        test    al, RHRZ        ; is horizontal retrace low
        jnz     short p11       ; wait until it is
p12:                            ; now wait for either retrace high
        in      al, dx          ; get status
        test    al, RVRT+RHRZ   ; is horizontal or vertical retrace high
        jz      short p12       ; wait until either is active
p13:
        add     esi, 0B8000h    ; 30/08/2014 (Retro UNIX 386 v1)
        mov     ax, word [esi]  ; get the character and attribute
        ;
        ; al = character, ah = attribute
        ;
        sti
        ; bl = video page number
u3:
        ;;mov   ax, 0601h       ; scroll one line
        ;;sub   cx, cx          ; upper left corner
        ;;mov   dh, 25-1        ; lower right row
        ;;;mov  dl, byte [CRT_COLS]
        ;mov    dl, 80          ; lower right column
        ;;dec   dl
        ;;mov   dl, 79

        ;;call  scroll_up       ; 04/12/2013
        ; 02/09/2014
        mov     cx, word [crt_ulc] ; Upper left corner  (0000h)
        mov     dx, word [crt_lrc] ; Lower right corner (2479h)
        ;
        mov     al, 1           ; scroll 1 line up
                ; ah = attribute
        jmp     scroll_up
;u4:
        ;;int   10h             ; video-call return
                                ; scroll up the screen
                                ; tty return
;u5:
        ;retn                   ; return to the caller

u6:                             ; set-cursor-inc
        inc     dh              ; next row
                                ; set cursor
;u7:
        ;;mov   ah, 02h
        ;;jmp   short u4        ; establish the new cursor
```

```
        ;call   set_cpos
        ;jmp    short u5
        jmp     set_cpos

        ; check for control characters
u8:
        je      short u9
        cmp     al, 0Ah         ; is it a line feed (0Ah)
        je      short u10
        cmp     al, 07h         ; is it a bell
        je      short u11
        cmp     al, 08h         ; is it a backspace
        ;jne    short u0
        je      short bs        ; 12/12/2013
        ; 12/12/2013 (tab stop)
        cmp     al, 09h         ; is it a tab stop
        jne     short u0
        mov     al, dl
        cbw
        mov     cl, 8
        div     cl
        sub     cl, ah
ts:
        ; 02/09/2014
        ; 01/09/2014
        mov     al, 20h
tsloop:
        push    cx
        push    ax
        xor     bh, bh
        ;mov    bl, [active_page]
        call    m3
        pop     ax  ; ah = attribute/color
        pop     cx
        dec     cl
        jnz     short tsloop
        retn
bs:
        ; back space found

        or      dl, dl          ; is it already at start of line
        ;je     short u7        ; set_cursor
        jz      short set_cpos
        dec     dx              ; no -- just move it back
        ;jmp    short u7
        jmp     short set_cpos

        ; carriage return found
u9:
        mov     dl, 0           ; move to first column
        ;jmp    short u7
        jmp     short set_cpos

        ; line feed found
;u10:
;       cmp     dh, 25-1        ; bottom of screen
;       jne     short u6        ; no, just set the cursor
;       jmp     u1              ; yes, scroll the screen

beeper:
        ; 30/08/2014 (Retro UNIX 386 v1)
        ; 18/01/2014
        ; 03/12/2013
        ; bell found
```

```
u11:
        sti
        cmp     bl, byte [active_page]
        jne     short u12       ; Do not sound the beep
                                ; if it is not written on the active page
        mov     cx, 1331        ; divisor for 896 hz tone
        mov     bl, 31          ; set count for 31/64 second for beep
        ;call   beep            ; sound the pod bell
        ;jmp    short u5        ; tty_return
        ;retn

TIMER   equ     040h            ; 8254 TIMER - BASE ADDRESS
PORT_B  equ     061h            ; PORT B READ/WRITE DIAGNOSTIC REGISTER
GATE2   equ     00000001b       ; TIMER 2 INPUT CATE CLOCK BIT
SPK2    equ     00000010b       ; SPEAKER OUTPUT DATA ENABLE BIT

beep:
        ; 07/02/2015
        ; 30/08/2014 (Retro UNIX 386 v1)
        ; 18/01/2014
        ; 03/12/2013
        ;
        ; TEST4.ASM - 06/10/85  POST AND BIOS UTILITY ROUTINES
        ;
        ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
        ;
        ; ENTRY:
        ;    (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
        ;    (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
        ; EXIT:                         :
        ;    (AX),(BL),(CX) MODIFIED.

        pushf  ; 18/01/2014     ; save interrupt status
        cli                     ; block interrupts during update
        mov     al, 10110110b   ; select timer 2, lsb, msb binary
        out     TIMER+3, al     ; write timer mode register
        jmp     $+2             ; I/O delay
        mov     al, cl          ; divisor for hz (low)
        out     TIMER+2,AL      ; write timer 2 count - lsb
        jmp     $+2             ; I/O delay
        mov     al, ch          ; divisor for hz (high)
        out     TIMER+2, al     ; write timer 2 count - msb
        in      al, PORT_B      ; get current setting of port
        mov     ah, al          ; save that setting
        or      al, GATE2+SPK2  ; gate timer 2 and turn speaker on
        out     PORT_B, al      ; and restore interrupt status
        ;popf   ; 18/01/2014
        sti
g7:                             ; 1/64 second per count (bl)
        mov     ecx, 1035       ; delay count for 1/64 of a second
        call    waitf           ; go to beep delay 1/64 count
        dec     bl              ; (bl) length count expired?
        jnz     short g7        ; no - continue beeping speaker
        ;
        ;pushf                  ; save interrupt status
        cli     ; 18/01/2014    ; block interrupts during update
        in      al, PORT_B      ; get current port value
        ;or      al, not (GATE2+SPK2) ; isolate current speaker bits in case
        or      al, ~(GATE2+SPK2)
        and     ah, al          ; someone turned them off during beep
        mov     al, ah          ; recover value of port
        ;or      al, not (GATE2+SPK2) ; force speaker data off
        or      al, ~(GATE2+SPK2) ; isolate current speaker bits in case
        out     PORT_B, al      ; and stop speaker timer
```

```
        ;popf                   ; restore interrupt flag state
        sti
        mov     ecx, 1035       ; force 1/64 second delay (short)
        call    waitf           ; minimum delay between all beeps
        ;pushf                  ; save interrupt status
        cli                     ; block interrupts during update
        in      al, PORT_B      ; get current port value in case
        and     al, GATE2+SPK2  ; someone turned them on
        or      al, ah          ; recover value of port_b
        out     PORT_B, al      ; restore speaker status
        popf                    ; restore interrupt flag state
u12:
        retn

REFRESH_BIT equ 00010000b       ; REFRESH TEST BIT

WAITF:
waitf:
        ; 30/08/2014 (Retro UNIX 386 v1)
        ; 03/12/2013
        ;
;       push ax                 ; save work register (ah)
;waitf1:
                                ; use timer 1 output bits
;       in      al, PORT_B      ; read current counter output status
;       and     al, REFRESH_BIT ; mask for refresh determine bit
;       cmp     al, ah          ; did it just change
;       je      short waitf1    ; wait for a change in output line
;       ;
;       mov     ah, al          ; save new lflag state
;       loop    waitf1          ; decrement half cycles till count end
;       ;
;       pop     ax              ; restore (ah)
;       retn                    ; return (cx)=0

; 06/02/2015 (unix386.s <-- dsectrm2.s)
; 17/12/2014 (dsectrm2.s)
; WAITF
; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
;
;---WAITF--------------------------------------------------------------------
;       FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
; ENTRY:
;       (CX) =  COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
;               MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
; EXIT:
;               AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
;       (CX) = 0
;----------------------------------------------------------------------------

; Refresh period: 30 micro seconds (15-80 us)
; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT_REFRESH)

;WAITF:                                 ; DELAY FOR (CX)*15.085737 US
        PUSH    AX                      ; SAVE WORK REGISTER (AH)
        ; 16/12/2014
        ;shr    cx, 1                   ; convert to count of 30 micro seconds
        shr     ecx, 1  ; 21/02/2015
;17/12/2014
;WAITF1:
;       IN      AL, PORT_B   ;061h      ; READ CURRENT COUNTER OUTPUT STATUS
;       AND     AL, REFRESH_BIT ;00010000b ; MASK FOR REFRESH DETERMINE BIT
;       CMP     AL, AH                  ; DID IT JUST CHANGE
;       JE      short WAITF1            ; WAIT FOR A CHANGE IN OUTPUT LINE
```

```
;       MOV     AH, AL                  ; SAVE NEW FLAG STATE
;       LOOP    WAITF1                  ; DECREMENT HALF CYCLES TILL COUNT END

        ;
        ; 17/12/2014
        ;
        ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
        ;
;WAIT_REFRESH:  Uses port 61, bit 4 to have CPU speed independent waiting.
;       INPUT:  CX = number of refresh periods to wait
;              (refresh periods = 1 per 30 microseconds on most machines)
WR_STATE_0:
        IN      AL,PORT_B               ; IN AL,SYS1
        TEST    AL,010H
        JZ      SHORT WR_STATE_0
WR_STATE_1:
        IN      AL,PORT_B               ; IN AL,SYS1
        TEST    AL,010H
        JNZ     SHORT WR_STATE_1
        LOOP    WR_STATE_0
        ;
        POP     AX                      ; RESTORE (AH)
        RETn                            ; (CX) = 0

set_cpos:
        ; 01/09/2014
        ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
        ;
        ; 12/12/2013 (Retro UNIX 8086 v1 - last update)
        ; 04/12/2013 (Retro UNIX 8086 v1 - beginning)
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
        ;
        ; SET_CPOS
        ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
        ;       NEW X-Y VALUES PASSED
        ; INPUT
        ;       DX - ROW,COLUMN OF NEW CURSOR
        ;       BH - DISPLAY PAGE OF CURSOR
        ; OUTPUT
        ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
        ;
        xor     eax, eax
        mov     al, bl  ; BL = video page number
        shl     al, 1   ; word offset
        mov     esi, cursor_posn
        add     esi, eax
        mov     word [esi], dx ; save the pointer
        cmp     byte [active_page], bl
        jne     short m17
        ;call   m18     ; CURSOR SET
;m17:                   ; SET_CPOS_RETURN
        ; 01/09/2014
;       retn
                ; DX  = row/column
m18:
        call    position ; determine location in regen buffer
        mov     cx, word [CRT_START]
        add     cx, ax  ; add char position in regen buffer
                        ; to the start address (offset) for this page
        shr     cx, 1   ; divide by 2 for char only count
        mov     ah, 14  ; register number for cursor
        ;call   m16     ; output value to the 6845
        ;retn
```

```
        ;-----  THIS ROUTINE OUTPUTS THE CX REGISTER
        ;       TO THE 6845 REGISTERS NAMED IN (AH)
m16:
        cli
        ;mov    dx, word [addr_6845] ; address register
        mov     dx, 03D4h ; I/O address of color card
        mov     al, ah  ; get value
        out     dx, al  ; register set
        inc     dx      ; data register
        jmp     $+2     ; i/o delay
        mov     al, ch  ; data
        out     dx, al
        dec     dx
        mov     al, ah
        inc     al      ; point to other data register
        out     dx, al  ; set for second register
        inc     dx
        jmp     $+2     ; i/o delay
        mov     al, cl  ; second data value
        out     dx, al
        sti
m17:
        retn


set_ctype:
        ; 02/09/2014 (Retro UNIX 386 v1)
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS

;       CH) = BITS 4-0 = START LINE FOR CURSOR
;       ** HARDWARE WILL ALWAYS CAUSE BLINK
;       ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
;          OR NO CURSOR AT ALL
;       (CL) = BITS 4-0 = END LINE FOR CURSOR

;-----------------------------------------------
; SET_CTYPE
;       THIS ROUTINE SETS THE CURSOR VALUE
; INPUT
;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
; OUTPUT
;       NONE
;-----------------------------------------------

        mov     ah, 10  ; 6845 register for cursor set
        ;mov    word [CURSOR_MODE], cx ; save in data area
        ;call   m16     ; output cx register
        ;retn
        jmp     m16


position:
        ; 02/09/2014
        ; 30/08/2014 (Retro UNIX 386 v1)
        ; 04/12/2013 (Retro UNIX 8086 v1)
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
        ;
        ; POSITION
        ;       THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
        ;       OF A CHARACTER IN THE ALPHA MODE
        ; INPUT
```

```
        ;       AX = ROW, COLUMN POSITION
        ; OUTPUT
        ;       AX = OFFSET OF CHAR POSITION IN REGEN BUFFER

                ; DX = ROW, COLUMN POSITION
        xor     eax, eax ; 02/09/2014
        ;mov    al, byte [CRT_COLS]
        mov     al, 80   ; determine bytes to row
        mul     dh ;     row value
        xor     dh, dh   ; 0
        add     ax, dx   ; add column value to the result
        shl     ax, 1   ; * 2 for attribute bytes
                ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
        retn

find_position:
        ; 07/09/2014
        ; 02/09/2014
        ; 30/08/2014 (Retro UNIX 386 v1)
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
        xor     ecx, ecx
        mov     cl, bl ; video page number
        mov     esi, ecx
        shl     si, 1
        mov     dx, word [esi + cursor_posn]
        jz      short p21
        xor     si, si
p20:
        ;add    esi, word [CRT_LEN]
        add     si, 80*25*2 ; add length of buffer for one page
        loop    p20
p21:
        and     dx, dx
        jz      short p22
        call    position ; determine location in regen in page
        add     esi, eax ; add location to start of regen page
p22:
        ;mov    dx, word [addr_6845] ; get base address of active display

        ;mov    dx, 03D4h ; I/O address of color card
        ;add    dx, 6   ; point at status port
        mov     dx, 03DAh ; status port
        ; cx = 0
        retn

scroll_up:
        ; 07/09/2014
        ; 02/09/2014
        ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
        ; 04/04/2014
        ; 04/12/2013
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
        ;
        ; SCROLL UP
        ;       THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
        ;       ON THE SCREEN
        ; INPUT
        ;       (AH) = CURRENT CRT MODE
        ;       (AL) = NUMBER OF ROWS TO SCROLL
        ;       (CX) = ROW/COLUMN OF UPPER LEFT CORNER
        ;       (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
        ;       (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
        ;       (DS) = DATA SEGMENT
```

```
        ;       (ES) = REGEN BUFFER SEGMENT
        ; OUTPUT
        ;       NONE -- THE REGEN BUFFER IS MODIFIED
        ;
        ;       bh = 0  (02/09/2014)
        ;
        ; ((ah = 3))
        ; cl = left upper column
        ; ch = left upper row
        ; dl = righ lower column
        ; dh = righ lower row
        ;
        ; al = line count
        ; ah = attribute to be used on blanked line
        ; bl = video page number (0 to 7)
        ;

        ; Test  Line Count
        or      al, al
        jz      short al_set
        mov     bh, dh  ; subtract lower row from upper row
        sub     bh, ch
        inc     bh      ; adjust difference by 1
        cmp     bh, al  ; line count = amount of rows in window?
        jne     short al_set ; if not the we're all set
        xor     al, al  ; otherwise set al to zero
al_set:
        xor     bh, bh  ; 0
        push    ax
        ;mov    esi, [crt_base]
        mov     esi, 0B8000h
        cmp     bl, [active_page]
        jne     short n0
        ;
        mov     ax, [CRT_START]
        add     si, ax
        jmp     short n1
n0:
        and     bl, bl
        jz      short n1
        mov     al, bl
n0x:
        ;add    si, word [CRT_LEN]
        ;add    esi, 80*25*2
        add     si, 80*25*2
        dec     al
        jnz     short n0x
n1:
        ;Scroll position
        push    dx
        mov     dx, cx  ; now, upper left position in DX
        call    position
        add     esi, eax
        mov     edi, esi
        pop     dx      ; lower right position in DX
        sub     dx, cx
        inc     dh      ; dh = #rows
        inc     dl      ; dl = #cols in block
        pop     ax      ; al = line count, ah = attribute
        xor     ecx, ecx
        mov     cx, ax
        ;mov    ah, byte [CRT_COLS]
        mov     ah, 80
        mul     ah      ; determine offset to from address
```

```
        add     ax, ax  ; *2 for attribute byte
        ;
        push    ax      ; offset
        push    dx
        ;
        ; 04/04/2014
        mov     dx, 3DAh ; guaranteed to be color card here
n8:                     ; wait_display_enable
        in      al, dx   ; get port
        test    al, RVRT ; wait for vertical retrace
        jz      short n8 ; wait_display_enable
        mov     al, 25h
        mov     dl, 0D8h ; address control port
        out     dx, al  ; turn off video during vertical retrace
        pop     dx      ; #rows, #cols
        pop     ax      ; offset
        xchg    ax, cx  ;
        ; ecx = offset, al = line count, ah = attribute
;n9:
        or      al, al
        jz      short n3
        add     esi, ecx ; from address for scroll
        mov     bh, dh  ; #rows in block
        sub     bh, al  ; #rows to be moved
n2:
        ; Move rows
        mov     cl, dl  ; get # of cols to move
        push    esi
        push    edi     ; save start address
n10:
        movsw           ; move that line on screen
        dec     cl
        jnz     short n10
        pop     edi
        pop     esi     ; recover addresses
        ;mov    cl, byte [CRT_COLS]
        ;add    cl, cl
        ;mov    ecx, 80*2
        mov     cx, 80*2
        add     esi, ecx  ; next line
        add     edi, ecx
        dec     bh       ; count of lines to move
        jnz     short n2 ; row loop
        ; bh = 0
        mov     dh, al   ; #rows
n3:
        ; attribute in ah
        mov     al, ' '  ; fill with blanks
        ; Clear rows
                ; dh =  #rows
        mov     cl, dl  ; get # of cols to clear
        push    edi     ; save address
n11:
        stosw           ; store fill character
        dec     cl
        jnz     short n11
        pop     edi     ; recover address
        ;mov    cl, byte [CRT_COLS]
        ;add    cl, cl
        ;mov    ecx, 80*2
        mov     cl, 80*2
        add     esi, ecx  ; next line
        add     edi, ecx
        dec     dh
```

```
        jnz     short n3
        ;
        cmp     bl, byte [active_page]
        jne     short n6
        ;mov    al, byte [CRT_MODE_SET] ; get the value of mode set
        mov     al, 29h ; (ORGS.ASM), M7 mode set table value for mode 3
        mov     dx, 03D8h ; always set color card port
        out     dx, al
n6:
        retn


write_c_current:
        ; 30/08/2014 (Retro UNIX 386 v1)
        ; 18/01/2014
        ; 04/12/2013
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
        ;
        ; WRITE_C_CURRENT
        ;       THIS ROUTINE WRITES THE CHARACTER AT
        ;       THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
        ; INPUT
        ;       (AH) = CURRENT CRT MODE
        ;       (BH) = DISPLAY PAGE
        ;       (CX) = COUNT OF CHARACTERS TO WRITE
        ;       (AL) = CHAR TO WRITE
        ;       (DS) = DATA SEGMENT
        ;       (ES) = REGEN SEGMENT
        ; OUTPUT
        ;       DISPLAY REGEN BUFFER UPDATED

        cli
        ; bl = video page
        ; al = character
        ; ah = color/attribute
        push    dx
        push    ax      ; save character & attribute/color
        call    find_position  ; get regen location and port address
        ; esi = regen location
        ; dx = status port
        ;
        ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE
        ;
p41:                    ; wait for horizontal retrace is low or vertical
        sti             ; enable interrupts first
        cmp     bl, byte [active_page]
        jne     short p44
        cli             ; block interrupts for single loop
        in      al, dx  ; get status from the adapter
        test    al, RVRT ; check for vertical retrace first
        jnz     short p43 ; Do fast write now if vertical retrace
        test    al, RHRZ ; is horizontal retrace low
        jnz     short p41 ; wait until it is
p42:                    ;  wait for either retrace high
        in      al, dx ; get status again
        test    al, RVRT+RHRZ ; is horizontal or vertical retrace high
        jz      short p42 ; wait until either retrace active
p43:
        sti
p44:
        pop     ax      ; restore the character (al) & attribute (ah)
        add     esi, 0B8000h ; 30/08/2014 (crt_base)
                             ; Retro UNIX 386 v1 feature only!
```

```
        mov     word [esi], ax
        pop     dx
        retn

set_mode:
        ; 02/09/2014 (Retro UNIX 386 v1)
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS

;--------------------------------------------------
; SET MODE                                        :
;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO :
;       THE SELECTED MODE, THE SCREEN IS BLANKED.  :
; INPUT                                           :
;       (AL) - MODE SELECTED (RANGE 0-7)          :
; OUTPUT                                          :
;       NONE                                      :
;--------------------------------------------------

        push    ebx
        push    edx
        push    eax

        ;mov    dx, 03D4h       ; address or color card
        mov     al, 3
;M8:
        mov     byte [CRT_MODE], al ; save mode in global variable
        mov     al, 29h
        ;mov    byte [CRT_MODE_SET], al ; save the mode set value
        and     al, 037h        ; video off, save high resolution bit
        ;push   dx              ; save port value
        ;add    dx, 4           ; point to control register
        mov     dx, 3D8h
        out     dx, al          ; reset video to off to suppress rolling
        ;pop    dx
;M9:
        xor     ah, ah
        mov     ebx, video_params ; initialization table
        ;mov    ax, word [ebx+10] ; get the cursor mode from the table
        ;xchg   ah, al
        ;mov    word [CURSOR_MODE], ax ; save cursor mode
        xor     ah, ah            ; ah is register number during loop

;-----  LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE

M10:                    ;  initialization loop
        mov     al, ah  ; get 6845 register number
        out     dx, al
        inc     dx      ; point to data port
        inc     ah      ; next register value
        mov     al, [ebx] ; get table value
        out     dx, al  ; out to chip
        inc     ebx     ; next in table
        dec     dx      ; back to pointer register
        loop    M10     ; do the whole table

;-----  FILL REGEN AREA WITH BLANK
        ;xor    ax, ax
        ;mov    word [CRT_START], ax ; start address saved in global
        ;mov    byte [ACTIVE_PAGE], al ; 0 ; (re)set page value
        ;mov    ecx, 8192 ; number of words in color card
        ; black background, light gray characeter color, space character
        ;mov    ax, 0720h ; fill char for alpha - attribute
;M13:                    ; clear buffer
```

```
        ;add    edi, 0B8000h ; [crt_base]
        ;rep    stosw   ; FILL THE REGEN BUFFER WITH BLANKS

;-----  ENABLE VIDEO AND CORRECT PORT SETTING
        ;mov    dx, 3D4h ; mov dx, word [ADDR_6845]
                         ; prepare to output to video enable port
        ;add    dx,4     ; point to the mode control gerister
        mov     dx, 3D8h
        ;mov    al, byte [CRT_MODE_SET] ; get the mode set value
        mov     al, 29h
        out     dx, al   ; set video enable port

;-----  DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
;-----  AND THE NUMBER TO BE USED FOR TTY INTERFACE
        ;
        ;mov byte [CRT_COLS], 80h ; initialize number of columns count
        ;
;-----  SET CURSOR POSITIONS
        push    edi
        ;mov    word [CRT_LEN], 80*25*2
        push    ecx
        mov     edi, cursor_posn
        mov     ecx, 4  ; clear all cursor positions (16 bytes)
        xor     eax, eax
        rep     stosd   ; fill with zeroes
        pop     ecx
        pop     edi

;-----  SET UP OVERSCAN REGISTER
        inc     dx      ; set overscan port to a default
        mov     al, 30h ; 30H valuye for all modes except 640X200 bw
;M14:
        out     dx, al  ; output the correct value to 3D9 port
        ;mov    byte [CRT_PALETTE], al ; save the value for future use

;-----  NORMAL RETURN FROM ALL VIDEO RETURNS
        ;
        pop     eax
        pop     edx
        pop     ebx
        retn

video_params:
        ; 02/09/2014 (Retro UNIX 386 v1)
        ;ORGS.ASM ----- 06/10/85   COMPATIBILITY MODULE
        ; VIDEO MODE 3
        db      71h,50h,5Ah,0Ah,1Fh,6,19h       ; SET UP FOR 80X25
        db      1Ch,2,7,6,7     ; cursor start = 6, cursor stop = 7
        db      0,0,0,0

tty_sw:
        ; 07/09/2014
        ; 02/09/2014 (Retro UNIX 386 v1 - beginning)
        ;
        ;mov     byte [u.quant], 0  ; 04/03/2014
        ;
;act_disp_page:
        ; 04/03/2014  (act_disp_page --> tty_sw)
        ; 10/12/2013
        ; 04/12/2013
        ;
        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
        ;
        ; ACT_DISP_PAGE
```

```
        ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
        ;       THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
        ; INPUT
        ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
        ; OUTPUT
        ;       THE 6845 IS RESET TO DISPLAY THAT PAGE

        ;cli

        push    si ; 10/12/2013
        ;push   ebx
        push    cx
        push    dx
        ;
        mov     byte [active_page], al ; save active page value ; [ptty]
        ;mov    cx, word [CRT_LEN] ; get saved length of regen buffer
        mov     cx, 25*80*2
        ; 01/09/2014
        xor     ebx, ebx
        mov     bl, al
        ;
        cbw     ; 07/09/2014 (ah=0)
        mul     cx      ; display page times regen length
        ; 10/12/2013
        mov     word [CRT_START], ax ; save start address for later
        mov     cx, ax ; start address to cx
        ;sar    cx, 1
        shr     cx, 1   ; divide by 2 for 6845 handling
        mov     ah, 12  ; 6845 register for start address
        call    m16
        ;sal    bx, 1
        ; 01/09/2014
        shl     bl, 1   ; *2 for word offset
        add     ebx, cursor_posn
        mov     dx, word [ebx] ; get cursor for this page
        call    m18
        ;
        pop     dx
        pop     cx
        ;pop    ebx
        pop     si ; 10/12/2013
        ;
        ;sti
        ;
        retn


; Write memory information
; Temporay Code
; 06/11/2014
;
memory_info:
        mov     eax, [memory_size] ; in pages
        push    eax
        shl     eax, 12            ; in bytes
        mov     ebx, 10
        mov     ecx, ebx           ; 10
        mov     esi, mem_total_b_str
        call    bintdstr
        pop     eax
        mov     cl, 7
        mov     esi, mem_total_p_str
        call    bintdstr
        mov     eax, [free_pages] ; in pages
```

```
        push    eax
        shl     eax, 12           ; in bytes
        mov     cl, 10
        mov     esi, free_mem_b_str
        call    bintdstr
        pop     eax
        mov     cl, 7
        mov     esi, free_mem_p_str
        call    bintdstr
        mov     esi, [next_page]
        mov     eax, esi
        shl     eax, 3 ; 1 byte = 8 pages
        add     esi, MEM_ALLOC_TBL
        mov     ecx, [esi]
        and     ecx, ecx
        jz      short mim1
mim0:
        shr     ecx, 1
        jc      short mim1
        inc     eax
        jmp     short mim0
mim1:
        mov     ecx, 7
        mov     esi, next_mem_p_str
        call    bintdstr
        mov     eax, [mat_size]
        mov     cl, 4
        mov     esi, mat_p_str
        call    bintdstr
        mov     eax, [memory_size]
        add     eax, 1023
        shr     eax, 10 ; Page table count
        mov     cl, 4
        mov     esi, pt_c_str
        call    bintdstr
        mov     ax, [mem_1m_1k]
        mov     cl, 5
        mov     esi, mem_1m_1k_str
        call    bintdstr
        mov     ax, [mem_16m_64k]
        mov     cl, 5
        mov     esi, mem_16m_64k_str
        call    bintdstr
        ;
        call    calc_free_mem
        mov     eax, edx ; free memory in pages
        ; ecx = 0
        mov     cl, 7
        mov     esi, free_mem_c_str
        call    bintdstr
        ;
        mov     esi, msg_memory_info
pmim:
        lodsb
        or      al, al
        jz      short pmim_ok
        push    esi
        xor     ebx, ebx ; 0
                         ; Video page 0 (bl=0)
        mov     ah, 07h ; Black background,
                         ; light gray forecolor
        call    write_tty
        pop     esi
        jmp     short pmim
```

```
pmim_ok:
        retn

; Convert binary number to decimal/numeric string
; 06/11/2014
; Temporay Code
;

bintdstr:
        ; EAX = binary number
        ; ESI = decimal/numeric string address
        ; EBX = divisor (10)
        ; ECX = string length (<=10)
        add     esi, ecx
btdstr0:
        dec     esi
        xor     edx, edx
        div     ebx
        add     dl, 30h
        mov     byte [esi], dl
        dec     cl
        jz      btdstr2
        or      eax, eax
        jnz     short btdstr0
btdstr1:
        dec     esi
        mov     byte [esi], 20h ; blank space
        dec     cl
        jnz     short btdstr1
btdstr2:
        retn

; Calculate free memory pages on M.A.T.
; 06/11/2014
; Temporary Code
;

calc_free_mem:
        xor     edx, edx
        ;xor    ecx, ecx
        mov     cx, [mat_size] ; in pages
        shl     ecx, 10 ; 1024 dwords per page
        mov     esi, MEM_ALLOC_TBL
cfm0:
        lodsd
        push    ecx
        mov     ecx, 32
cfm1:
        shr     eax, 1
        jnc     short cfm2
        inc     edx
cfm2:
        loop    cfm1
        pop     ecx
        loop    cfm0
        retn

; 06/02/2015
diskette_io:
        pushfd
        push    cs
        call    DISKETTE_IO_1
        retn
```

```
;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;
;//////////////////////////////////////////////////////

; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
; 20/02/2015
; 06/02/2015 (unix386.s)
; 16/12/2014 - 02/01/2015 (dsectrm2.s)
;
; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
;
; ADISK.EQU

;----- Wait control constants

;amount of time to wait while RESET is active.

WAITCPU_RESET_ON        EQU     21              ;Reset on must last at least
14us
                                                ;at 250 KBS xfer rate.
                                                ;see INTEL MCS, 1985, pg. 5-456

WAITCPU_FOR_STATUS      EQU     100             ;allow 30 microseconds for
                                                ;status register to become valid
                                                ;before re-reading.

;After sending a byte to NEC, status register may remain
;incorrectly set for 24 us.

WAITCPU_RQM_LOW         EQU     24              ;number of loops to check for
                                                ;RQM low.

; COMMON.MAC
;
;       Timing macros
;

%macro          SIODELAY 0                      ; SHORT IODELAY
                jmp short $+2
%endmacro

%macro          IODELAY  0                      ; NORMAL IODELAY
                jmp short $+2
                jmp short $+2
%endmacro

%macro          NEWIODELAY 0
                out     0ebh,al
%endmacro

; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
;;; WAIT_FOR_MEM
WAIT_FDU_INT_LO         equ     017798          ; 2.5 secs in 30 micro units.
WAIT_FDU_INT_HI         equ     1
;;; WAIT_FOR_PORT
WAIT_FDU_SEND_LO        equ     16667           ; .5 secons in 30 us units.
WAIT_FDU_SEND_HI        equ     0
;Time to wait while waiting for each byte of NEC results = .5
;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
WAIT_FDU_RESULTS_LO     equ     16667           ; .5 seconds in 30 micro units.
WAIT_FDU_RESULTS_HI     equ     0
;;; WAIT_REFRESH
;amount of time to wait for head settle, per unit in parameter
;table = 1 ms.
WAIT_FDU_HEAD_SETTLE    equ     33              ; 1 ms in 30 micro units.
```

```
; /////////////// DISKETTE I/O ///////////////

; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)

;--------------------------------------
;       EQUATES USED BY POST AND BIOS   :
;--------------------------------------

;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
;PORT_A         EQU     060H            ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
;PORT_B         EQU     061H            ; PORT B READ/WRITE DIAGNOSTIC REGISTER
;REFRESH_BIT    EQU     00010000B       ; REFRESH TEST BIT

;--------------------------------------
;       CMOS EQUATES FOR THIS SYSTEM    :
;--------------------------------------------------------------------------------
;CMOS_PORT      EQU     070H            ; I/O ADDRESS OF CMOS ADDRESS PORT
;CMOS_DATA      EQU     071H            ; I/O ADDRESS OF CMOS DATA PORT
;NMI            EQU     10000000B       ; DISABLE NMI INTERRUPTS MASK -
                                        ;  HIGH BIT OF CMOS LOCATION ADDRESS

;---------- CMOS TABLE LOCATION ADDRESS'S ## ----------------------------------
CMOS_DISKETTE   EQU     010H            ; DISKETTE DRIVE TYPE BYTE            ;
;               EQU     011H            ; - RESERVED                          ;C
CMOS_DISK       EQU     012H            ; FIXED DISK TYPE BYTE                ;H
;               EQU     013H            ; - RESERVED                          ;E
CMOS_EQUIP      EQU     014H            ; EQUIPMENT WORD LOW BYTE             ;C

;---------- DISKETTE EQUATES --------------------------------------------------
INT_FLAG        EQU     10000000B       ; INTERRUPT OCCURRENCE FLAG
DSK_CHG         EQU     10000000B       ; DISKETTE CHANGE FLAG MASK BIT
DETERMINED      EQU     00010000B       ; SET STATE DETERMINED IN STATE BITS
HOME            EQU     00010000B       ; TRACK 0 MASK
SENSE_DRV_ST    EQU     00000100B       ; SENSE DRIVE STATUS COMMAND
TRK_SLAP        EQU     030H            ; CRASH STOP (48 TPI DRIVES)
QUIET_SEEK      EQU     00AH            ; SEEK TO TRACK 10
;MAX_DRV        EQU     2               ; MAX NUMBER OF DRIVES
HD12_SETTLE     EQU     15              ; 1.2 M HEAD SETTLE TIME
HD320_SETTLE    EQU     20              ; 320 K HEAD SETTLE TIME
MOTOR_WAIT      EQU     37              ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF

;---------- DISKETTE ERRORS ---------------------------------------------------
;TIME_OUT       EQU     080H            ; ATTACHMENT FAILED TO RESPOND
;BAD_SEEK       EQU     040H            ; SEEK OPERATION FAILED
BAD_NEC         EQU     020H            ; DISKETTE CONTROLLER HAS FAILED
BAD_CRC         EQU     010H            ; BAD CRC ON DISKETTE READ
MED_NOT_FND     EQU     00CH            ; MEDIA TYPE NOT FOUND
DMA_BOUNDARY    EQU     009H            ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
BAD_DMA         EQU     008H            ; DMA OVERRUN ON OPERATION
MEDIA_CHANGE    EQU     006H            ; MEDIA REMOVED ON DUAL ATTACH CARD
RECORD_NOT_FND  EQU     004H            ; REQUESTED SECTOR NOT FOUND
WRITE_PROTECT   EQU     003H            ; WRITE ATTEMPTED ON WRITE PROTECT DISK
BAD_ADDR_MARK   EQU     002H            ; ADDRESS MARK NOT FOUND
BAD_CMD         EQU     001H            ; BAD COMMAND PASSED TO DISKETTE I/O

;---------- DISK CHANGE LINE EQUATES ------------------------------------------
NOCHGLN         EQU     001H            ; NO DISK CHANGE LINE AVAILABLE
CHGLN           EQU     002H            ; DISK CHANGE LINE AVAILABLE

;---------- MEDIA/DRIVE STATE INDICATORS --------------------------------------
TRK_CAPA        EQU     00000001B       ; 80 TRACK CAPABILITY
FMT_CAPA        EQU     00000010B       ; MULTIPLE FORMAT CAPABILITY (1.2M)
```

```
DRV_DET         EQU     00000100B       ; DRIVE DETERMINED
MED_DET         EQU     00010000B       ; MEDIA DETERMINED BIT
DBL_STEP        EQU     00100000B       ; DOUBLE STEP BIT
RATE_MSK        EQU     11000000B       ; MASK FOR CLEARING ALL BUT RATE
RATE_500        EQU     00000000B       ; 500 KBS DATA RATE
RATE_300        EQU     01000000B       ; 300 KBS DATA RATE
RATE_250        EQU     10000000B       ; 250 KBS DATA RATE
STRT_MSK        EQU     00001100B       ; OPERATION START RATE MASK
SEND_MSK        EQU     11000000B       ; MASK FOR SEND RATE BITS

;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY ------------------------
M3D3U           EQU     00000000B       ; 360 MEDIA/DRIVE NOT ESTABLISHED
M3D1U           EQU     00000001B       ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
M1D1U           EQU     00000010B       ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
MED_UNK         EQU     00000111B       ; NONE OF THE ABOVE

;---------- INTERRUPT EQUATES -------------------------------------------------
;EOI            EQU     020H            ; END OF INTERRUPT COMMAND TO 8259
;INTA00         EQU     020H            ; 8259 PORT
INTA01          EQU     021H            ; 8259 PORT
INTB00          EQU     0A0H            ; 2ND 8259
INTB01          EQU     0A1H            ;

;-----------------------------------------------------------------------------
DMA08           EQU     008H            ; DMA STATUS REGISTER PORT ADDRESS
DMA             EQU     000H            ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
DMA18           EQU     0D0H            ; 2ND DMA STATUS PORT ADDRESS
DMA1            EQU     0C0H            ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
;-----------------------------------------------------------------------------
;TIMER          EQU     040H            ; 8254 TIMER - BASE ADDRESS

;-----------------------------------------------------------------------------
DMA_PAGE        EQU     081H            ; START OF DMA PAGE REGISTERS

; 06/02/2015 (unix386.s, protected mode modifications)
; (unix386.s <-- dsectrm2.s)
; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)

;---------------------------------------
;       80286 INTERRUPT LOCATIONS       :
;       REFERENCED BY POST & BIOS       :
;---------------------------------------

DISK_POINTER:   dd      MD_TBL6         ; Pointer to Diskette Parameter Table

; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
;-------------------------------------------------------------
; DISK_BASE                                                   :
;       THIS IS THE SET OF PARAMETERS REQUIRED FOR            :
;       DISKETTE OPERATION. THEY ARE POINTED AT BY THE        :
;       DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS,  :
;       BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT         :
;-------------------------------------------------------------

;DISK_BASE:
;       DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
;       DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
;       DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
;       DB      2               ; 512 BYTES/SECTOR
;       ;DB     15              ; EOT (LAST SECTOR ON TRACK)
;       db      18              ; (EOT for 1.44MB diskette)
;       DB      01BH            ; GAP LENGTH
;       DB      0FFH            ; DTL
;       ;DB     054H            ; GAP LENGTH FOR FORMAT
```

```
;       db      06ch            ; (for 1.44MB dsikette)
;       DB      0F6H            ; FILL BYTE FOR FORMAT
;       DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
;       DB      8               ; MOTOR START TIME (1/8 SECONDS)

;---------------------------------------
;       ROM BIOS DATA AREAS             :
;---------------------------------------

;DATA           SEGMENT AT 40H          ; ADDRESS= 0040:0000

;@EQUIP_FLAG    DW      ?               ; INSTALLED HARDWARE FLAGS

;---------------------------------------
;       DISKETTE DATA AREAS             :
;---------------------------------------

;@SEEK_STATUS   DB      ?               ; DRIVE RECALIBRATION STATUS
;                                       ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
;                                       ; BEFORE NEXT SEEK IF BIT IS = 0
;@MOTOR_STATUS  DB      ?               ; MOTOR STATUS
;                                       ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
;                                       ; BIT 7 = CURRENT OPERATION IS A WRITE
;@MOTOR_COUNT   DB      ?               ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
;@DSKETTE_STATUS DB     ?               ; RETURN CODE STATUS BYTE
;                                       ; CMD_BLOCK  IN STACK FOR DISK OPERATION
;@NEC_STATUS    DB      7 DUP(?)        ; STATUS BYTES FROM DISKETTE OPERATION

SEEK_STATUS:    db      0
MOTOR_STATUS:   db      0
MOTOR_COUNT:    db      0
DSKETTE_STATUS: db      0
NEC_STATUS:     db      0,0,0,0,0,0,0

;---------------------------------------
;       POST AND BIOS WORK DATA AREA    :
;---------------------------------------

;@INTR_FLAG     DB      ?               ; FLAG INDICATING AN INTERRUPT HAPPENED

;---------------------------------------
;       TIMER DATA AREA                 :
;---------------------------------------

TIMER_LH:       ; 16/02/205
TIMER_LOW:      DW      0               ; LOW WORD OF TIMER COUNT
TIMER_HIGH:     DW      0               ; HIGH WORD OF TIMER COUNT
TIMER_OFL:      DB      0               ; TIMER HAS ROLLED OVER SINCE LAST READ

; 17/12/2014  (IRQ 0 - INT 08H)
;TIMER_LOW      equ     46Ch            ; Timer ticks (counter)  @ 40h:006Ch
;TIMER_HIGH     equ     46Eh            ; (18.2 timer ticks per second)
;TIMER_OFL      equ     470h            ; Timer - 24 hours flag  @ 40h:0070h

;---------------------------------------
;       ADDITIONAL MEDIA DATA           :
;---------------------------------------

;@LASTRATE      DB      ?               ; LAST DISKETTE DATA RATE SELECTED
;@DSK_STATE     DB      ?               ; DRIVE 0 MEDIA STATE
;               DB      ?               ; DRIVE 1 MEDIA STATE
;               DB      ?               ; DRIVE 0 OPERATION START STATE
;               DB      ?               ; DRIVE 1 OPERATION START STATE
;@DSK_TRK       DB      ?               ; DRIVE 0 PRESENT CYLINDER
```

```
;               DB      ?               ; DRIVE 1 PRESENT CYLINDER

LASTRATE:       db      0
;HF_STATUS:     db      0
;HF_ERROR:      db      0
;HF_INT_FLAG:   db      0
;HF_CNTRL:      db      0
DSK_STATE:      db      0,0,0,0
DSK_TRK:        db      0,0

;DATA           ENDS                    ; END OF BIOS DATA SEGMENT


; 17/12/2014 (mov ax, [cfd])
; 11/12/2014
cfd:            db 0                    ; current floppy drive (for GET_PARM)
; 17/12/2014                            ; instead of 'DISK_POINTER'
pfd:            db 1                    ; previous floppy drive (for GET_PARM)
                                        ; (initial value of 'pfd
                                        ; must be different then 'cfd' value
                                        ; to force updating/initializing
                                        ; current drive parameters)
; 10/12/2014
;
;int40h:
;       pushf
;       push    cs
;       ;cli
;       call    DISKETTE_IO_1
;       retn

; DSKETTE ----- 04/21/86 DISKETTE BIOS
; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
;

;-- INT13H ------------------------------------------------------------------
; DISKETTE I/O
;       THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
;       1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
; INPUT
;       (AH) =  00H RESET DISKETTE SYSTEM
;               HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
;               ON ALL DRIVES
;-----------------------------------------------------------------------------

;       (AH)= 01H  READ THE STATUS OF THE SYSTEM INTO (AH)
;               @DISKETTE_STATUS FROM LAST OPERATION IS USED
;-----------------------------------------------------------------------------
;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
;       (CH) - TRACK NUMBER (NOT VALUE CHECKED)
;               MEDIA   DRIVE   TRACK NUMBER
;               320/360 320/360     0-39
;               320/360 1.2M        0-39
;               1.2M    1.2M        0-79
;               720K    720K        0-79
;               1.44M   1.44M       0-79
;       (CL) -  SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
;               MEDIA   DRIVE   SECTOR NUMBER
;               320/360 320/360      1-8/9
;               320/360 1.2M         1-8/9
;               1.2M    1.2M         1-15
;               720K    720K         1-9
```

```
;               1.44M   1.44M        1-18
;       (AL)    NUMBER OF SECTORS (NOT VALUE CHECKED)
;               MEDIA   DRIVE   MAX NUMBER OF SECTORS
;               320/360 320/360         8/9
;               320/360 1.2M            8/9
;               1.2M    1.2M            15
;               720K    720K            9
;               1.44M   1.44M           18
;
;       (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
;
;-------------------------------------------------------------------------------
;       (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY
;-------------------------------------------------------------------------------
;       (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY
;-------------------------------------------------------------------------------
;       (AH)= 04H  VERIFY THE DESIRED SECTORS
;-------------------------------------------------------------------------------
;       (AH)= 05H  FORMAT THE DESIRED TRACK
;               (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
;               FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
;               WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
;               N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
;               THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
;               THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
;               READ/WRITE ACCESS.
;               PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
;               ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
;               THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
;               (INT 13H, AH =  18H) MUST BE CALLED TO SET THE DISKETTE TYPE
;               THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
;               IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
;               MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
;
;               THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
;               FORMAT THE FOLLOWING MEDIAS:
;               ------------------------------------------
;               : MEDIA  :     DRIVE      : PARM 1 : PARM 2 :
;               ------------------------------------------
;               : 320K   : 320K/360K/1.2M :  50H   :   8    :
;               : 360K   : 320K/360K/1.2M :  50H   :   9    :
;               : 1.2M   : 1.2M           :  54H   :  15    :
;               : 720K   : 720K/1.44M     :  50H   :   9    :
;               : 1.44M  : 1.44M          :  6CH   :  18    :
;               ------------------------------------------
;               NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
;                      - PARM 2 = EOT (LAST SECTOR ON TRACK)
;                      - DISK BASE IS POINTED BY DISK POINTER LOCATED
;                        AT ABSOLUTE ADDRESS 0:78.
;                      - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
;                        SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.

;-------------------------------------------------------------------------------
;       (AH) = 08H READ DRIVE PARAMETERS
;       REGISTERS
;         INPUT
;           (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;         OUTPUT
;           (ES:DI) POINTS TO DRIVE PARAMETER TABLE
;           (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
;           (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
;                  BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
;           (DH) - MAXIMUM HEAD NUMBER
;           (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
```

```
;           (BH) - 0
;           (BL) - BITS 7 THRU 4 - 0
;                  BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
;           (AX) - 0
;         UNDER THE FOLLOWING CIRCUMSTANCES:
;           (1) THE DRIVE NUMBER IS INVALID,
;           (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
;           (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
;           (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
;           THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
;           IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
;           @DISKETTE_STATUS = 0 AND CY IS RESET.
;--------------------------------------------------------------------------------
;       (AH)= 15H  READ DASD TYPE
;       OUTPUT REGISTERS
;       (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
;               00 - DRIVE NOT PRESENT
;               01 - DISKETTE, NO CHANGE LINE AVAILABLE
;               02 - DISKETTE, CHANGE LINE AVAILABLE
;               03 - RESERVED (FIXED DISK)
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;--------------------------------------------------------------------------------
;       (AH)= 16H  DISK CHANGE LINE STATUS
;       OUTPUT REGISTERS
;       (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
;              06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
;--------------------------------------------------------------------------------
;       (AH)= 17H  SET DASD TYPE FOR FORMAT
;       INPUT REGISTERS
;       (AL) -  00 - NOT USED
;               01 - DISKETTE 320/360K IN 360K DRIVE
;               02 - DISKETTE 360K IN 1.2M DRIVE
;               03 - DISKETTE 1.2M IN 1.2M DRIVE
;               04 - DISKETTE 720K IN 720K DRIVE
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
;              (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
;--------------------------------------------------------------------------------
;       (AH)= 18H  SET MEDIA TYPE FOR FORMAT
;       INPUT REGISTERS
;       (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
;       (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
;              BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
;       (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
;       OUTPUT REGISTERS:
;       (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
;                 UNCHANGED IF (AH) IS NON-ZERO
;       (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
;            - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
;            - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
;            - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
;--------------------------------------------------------------------------------
;       DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
;       THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
;       ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
;               ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
;               IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
;               CHANGE ERROR CODE
;               IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
;               TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
;       IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
;
; DATA VARIABLE -- @DISK_POINTER
;       DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
```

```
;--------------------------------------------------------------------------
; OUTPUT FOR ALL FUNCTIONS
;       AH = STATUS OF OPERATION
;               STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
;               VARIABLE IN THE DATA SEGMENT OF THIS MODULE
;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
;               TYPE AH=(15)).
;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
;       FOR READ/WRITE/VERIFY
;               DS,BX,DX,CX PRESERVED
;       NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
;               ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
;               ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
;               THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
;               PROBLEM IS NOT DUE TO MOTOR START-UP.
;--------------------------------------------------------------------------
;
; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
;
;   ---------------------------------------------------------------
;   |       |       |       |       |       |       |       |       |
;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
;   |       |       |       |       |       |       |       |       |
;   ---------------------------------------------------------------
;       |       |       |       |       |       |       |       |
;       |       |       |       |       |       -----------------
;       |       |       |       |       |               |
;       |       |       |       |   RESERVED            |
;       |       |       |       |                 PRESENT STATE
;       |       |       |       |       000: 360K IN 360K DRIVE UNESTABLISHED
;       |       |       |       |       001: 360K IN 1.2M DRIVE UNESTABLISHED
;       |       |       |       |       010: 1.2M IN 1.2M DRIVE UNESTABLISHED
;       |       |       |       |       011: 360K IN 360K DRIVE ESTABLISHED
;       |       |       |       |       100: 360K IN 1.2M DRIVE ESTABLISHED
;       |       |       |       |       101: 1.2M IN 1.2M DRIVE ESTABLISHED
;       |       |       |       |       110: RESERVED
;       |       |       |       |       111: NONE OF THE ABOVE
;       |       |       |       |
;       |       |       |       ------> MEDIA/DRIVE ESTABLISHED
;       |       |       |
;       |       |       --------------> DOUBLE STEPPING REQUIRED (360K IN 1.2M
;       |       |                       DRIVE)
;       |       |
;       ------------------------------> DATA TRANSFER RATE FOR THIS DRIVE:
;
;                                               00: 500 KBS
;                                               01: 300 KBS
;                                               10: 250 KBS
;                                               11: RESERVED
;
;
;--------------------------------------------------------------------------
; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
;--------------------------------------------------------------------------
; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
;--------------------------------------------------------------------------

struc MD
        .SPEC1          resb    1       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
        .SPEC2          resb    1       ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        .OFF_TIM        resb    1       ; WAIT TIME AFTER OPERATION TILL MOTOR
OFF
        .BYT_SEC        resb    1       ; 512 BYTES/SECTOR
        .SEC_TRK        resb    1       ; EOT (LAST SECTOR ON TRACK)
```

```
        .GAP            resb    1       ; GAP LENGTH
        .DTL            resb    1       ; DTL
        .GAP3           resb    1       ; GAP LENGTH FOR FORMAT
        .FIL_BYT        resb    1       ; FILL BYTE FOR FORMAT
        .HD_TIM         resb    1       ; HEAD SETTLE TIME (MILLISECONDS)
        .STR_TIM        resb    1       ; MOTOR START TIME (1/8 SECONDS)
        .MAX_TRK        resb    1       ; MAX. TRACK NUMBER
        .RATE           resb    1       ; DATA TRANSFER RATE
endstruc

BIT7OFF EQU     7FH
BIT7ON  EQU     80H

;--------------------------------------------------------
;       DRIVE TYPE TABLE                                :
;--------------------------------------------------------
                ; 16/02/2015 (unix386.s, 32 bit modifications)
DR_TYPE:
                DB      01              ;DRIVE TYPE, MEDIA TABLE
                ;DW      MD_TBL1
                dd      MD_TBL1
                DB      02+BIT7ON
                ;DW      MD_TBL2
                dd      MD_TBL2
DR_DEFAULT:     DB      02
                ;DW      MD_TBL3
                dd      MD_TBL3
                DB      03
                ;DW      MD_TBL4
                dd      MD_TBL4
                DB      04+BIT7ON
                ;DW      MD_TBL5
                dd      MD_TBL5
                DB      04
                ;DW      MD_TBL6
                dd      MD_TBL6
DR_TYPE_E       equ $                   ; END OF TABLE
;DR_CNT         EQU     (DR_TYPE_E-DR_TYPE)/3
DR_CNT          equ     (DR_TYPE_E-DR_TYPE)/5
;--------------------------------------------------------
;       MEDIA/DRIVE PARAMETER TABLES                    :
;--------------------------------------------------------
;--------------------------------------------------------
;       360 KB MEDIA IN 360 KB DRIVE                    :
;--------------------------------------------------------
MD_TBL1:
        DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
        DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
        DB      2               ; 512 BYTES/SECTOR
        DB      09              ; EOT (LAST SECTOR ON TRACK)
        DB      02AH            ; GAP LENGTH
        DB      0FFH            ; DTL
        DB      050H            ; GAP LENGTH FOR FORMAT
        DB      0F6H            ; FILL BYTE FOR FORMAT
        DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
        DB      8               ; MOTOR START TIME (1/8 SECONDS)
        DB      39              ; MAX. TRACK NUMBER
        DB      RATE_250        ; DATA TRANSFER RATE
;--------------------------------------------------------
;       360 KB MEDIA IN 1.2 MB DRIVE                    :
;--------------------------------------------------------
MD_TBL2:
        DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
```

```
        DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
        DB      2               ; 512 BYTES/SECTOR
        DB      09              ; EOT (LAST SECTOR ON TRACK)
        DB      02AH            ; GAP LENGTH
        DB      0FFH            ; DTL
        DB      050H            ; GAP LENGTH FOR FORMAT
        DB      0F6H            ; FILL BYTE FOR FORMAT
        DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
        DB      8               ; MOTOR START TIME (1/8 SECONDS)
        DB      39              ; MAX. TRACK NUMBER
        DB      RATE_300        ; DATA TRANSFER RATE
;-------------------------------------------------------
;       1.2 MB MEDIA IN 1.2 MB DRIVE                    :
;-------------------------------------------------------
MD_TBL3:
        DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
        DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
        DB      2               ; 512 BYTES/SECTOR
        DB      15              ; EOT (LAST SECTOR ON TRACK)
        DB      01BH            ; GAP LENGTH
        DB      0FFH            ; DTL
        DB      054H            ; GAP LENGTH FOR FORMAT
        DB      0F6H            ; FILL BYTE FOR FORMAT
        DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
        DB      8               ; MOTOR START TIME (1/8 SECONDS)
        DB      79              ; MAX. TRACK NUMBER
        DB      RATE_500        ; DATA TRANSFER RATE
;-------------------------------------------------------
;       720 KB MEDIA IN 720 KB DRIVE                    :
;-------------------------------------------------------
MD_TBL4:
        DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
        DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
        DB      2               ; 512 BYTES/SECTOR
        DB      09              ; EOT (LAST SECTOR ON TRACK)
        DB      02AH            ; GAP LENGTH
        DB      0FFH            ; DTL
        DB      050H            ; GAP LENGTH FOR FORMAT
        DB      0F6H            ; FILL BYTE FOR FORMAT
        DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
        DB      8               ; MOTOR START TIME (1/8 SECONDS)
        DB      79              ; MAX. TRACK NUMBER
        DB      RATE_250        ; DATA TRANSFER RATE
;-------------------------------------------------------
;       720 KB MEDIA IN 1.44 MB DRIVE                   :
;-------------------------------------------------------
MD_TBL5:
        DB      11011111B       ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
        DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
        DB      2               ; 512 BYTES/SECTOR
        DB      09              ; EOT (LAST SECTOR ON TRACK)
        DB      02AH            ; GAP LENGTH
        DB      0FFH            ; DTL
        DB      050H            ; GAP LENGTH FOR FORMAT
        DB      0F6H            ; FILL BYTE FOR FORMAT
        DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
        DB      8               ; MOTOR START TIME (1/8 SECONDS)
        DB      79              ; MAX. TRACK NUMBER
        DB      RATE_250        ; DATA TRANSFER RATE
;-------------------------------------------------------
```

```
;       1.44 MB MEDIA IN 1.44 MB DRIVE                  :
;-------------------------------------------------------
MD_TBL6:
        DB      10101111B       ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
        DB      2               ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
        DB      MOTOR_WAIT      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
        DB      2               ; 512 BYTES/SECTOR
        DB      18              ; EOT (LAST SECTOR ON TRACK)
        DB      01BH            ; GAP LENGTH
        DB      0FFH            ; DTL
        DB      06CH            ; GAP LENGTH FOR FORMAT
        DB      0F6H            ; FILL BYTE FOR FORMAT
        DB      15              ; HEAD SETTLE TIME (MILLISECONDS)
        DB      8               ; MOTOR START TIME (1/8 SECONDS)
        DB      79              ; MAX. TRACK NUMBER
        DB      RATE_500        ; DATA TRANSFER RATE

;;int13h: ; 16/02/2015
;; 16/02/2015 - 21/02/2015
int40h:
        pushfd
        push    cs
        call    DISKETTE_IO_1
        retn

DISKETTE_IO_1:

        STI                             ; INTERRUPTS BACK ON
        PUSH    eBP                     ; USER REGISTER
        PUSH    eDI                     ; USER REGISTER
        PUSH    eDX                     ; HEAD #, DRIVE # OR USER REGISTER
        PUSH    eBX                     ; BUFFER OFFSET PARAMETER OR REGISTER
        PUSH    eCX                     ; TRACK #-SECTOR # OR USER REGISTER
        MOV     eBP,eSP                 ; BP     => PARAMETER LIST DEP. ON AH
                                        ; [BP]   = SECTOR #
                                        ; [BP+1] = TRACK #
                                        ; [BP+2] = BUFFER OFFSET
                                        ; FOR RETURN OF DRIVE PARAMETERS:
                                        ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
                                        ;           BITS 0-5 MAX SECTORS/TRACK
                                        ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
                                        ; BL/[BP+2] = BITS 7-4 = 0
                                        ;             BITS 3-0 = VALID CMOS TYPE
                                        ; BH/[BP+3] = 0
                                        ; DL/[BP+4] = # DRIVES INSTALLED
                                        ; DH/[BP+5] = MAX HEAD #
                                        ; DI/[BP+6] = OFFSET TO DISK BASE
        push    es ; 06/02/2015
        PUSH    DS                      ; BUFFER SEGMENT PARM OR USER REGISTER
        PUSH    eSI                     ; USER REGISTERS
        ;CALL   DDS                     ; SEGMENT OF BIOS DATA AREA TO DS
        ;mov    cx, cs
        ;mov    ds, cx
        mov     cx, KDATA
        mov     ds, cx
        mov     es, cx

        ;CMP    AH,(FNC_TAE-FNC_TAB)/2  ; CHECK FOR > LARGEST FUNCTION
        cmp     ah,(FNC_TAE-FNC_TAB)/4 ; 18/02/2015
        JB      short OK_FUNC           ; FUNCTION OK
        MOV     AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION
OK_FUNC:
        CMP     AH,1                    ; RESET OR STATUS ?
        JBE     short OK_DRV            ; IF RESET OR STATUS DRIVE ALWAYS OK
```

```
        CMP     AH,8                    ; READ DRIVE PARMS ?
        JZ      short OK_DRV            ; IF SO DRIVE CHECKED LATER
        CMP     DL,1                    ; DRIVES 0 AND 1 OK
        JBE     short OK_DRV            ; IF 0 OR 1 THEN JUMP
        MOV     AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION
OK_DRV:
        xor     ecx, ecx
        ;mov    esi, ecx ; 08/02/2015
        mov     edi, ecx ; 08/02/2015
        MOV     CL,AH                   ; CL = FUNCTION
        ;XOR    CH,CH                   ; CX = FUNCTION
        ;SHL    CL, 1                   ; FUNCTION TIMES 2
        SHL     CL, 2 ; 20/02/2015      ; FUNCTION TIMES 4 (for 32 bit offset)
        MOV     eBX,FNC_TAB             ; LOAD START OF FUNCTION TABLE
        ADD     eBX,eCX                 ; ADD OFFSET INTO TABLE => ROUTINE
        MOV     AH,DH                   ; AX = HEAD #,# OF SECTORS OR DASD TYPE
        XOR     DH,DH                   ; DX = DRIVE #
        MOV     SI,AX                   ; SI = HEAD #,# OF SECTORS OR DASD TYPE
        MOV     DI,DX                   ; DI = DRIVE #
        ;
        ; 11/12/2014
        mov     [cfd], dl               ; current floppy drive (for 'GET_PARM')

        ;
        MOV     AH, [DSKETTE_STATUS]    ; LOAD STATUS TO AH FOR STATUS FUNCTION
        MOV     byte [DSKETTE_STATUS],0 ; INITIALIZE FOR ALL OTHERS

;       THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
;       THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
;       FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
;
;               DI      : DRIVE #
;               SI-HI   : HEAD #
;               SI-LOW  : # OF SECTORS OR DASD TYPE FOR FORMAT
;               ES      : BUFFER SEGMENT
;               [BP]    : SECTOR #
;               [BP+1]  : TRACK #
;               [BP+2]  : BUFFER OFFSET
;
;       ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
;       SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
;       CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
;       SPECIFIC ERROR CODE.
;
                                        ; (AH) = @DSKETTE_STATUS
        CALL    dWORD [eBX]             ; CALL THE REQUESTED FUNCTION
        POP     eSI                     ; RESTORE ALL REGISTERS
        POP     DS
        pop     es      ; 06/02/2015
        POP     eCX
        POP     eBX
        POP     eDX
        POP     eDI
        MOV     eBP, eSP
        PUSH    eAX
        PUSHFd
        POP     eAX
        ;MOV    [BP+6], AX
        mov     [ebp+12], eax  ; 18/02/2015, flags
        POP     eAX
        POP     eBP
        IRETd

;-----------------------------------------------------------------------------
```

```
; DW --> dd (06/02/2015)
FNC_TAB dd      DSK_RESET               ; AH = 00H; RESET
        dd      DSK_STATUS              ; AH = 01H; STATUS
        dd      DSK_READ                ; AH = 02H; READ
        dd      DSK_WRITE               ; AH = 03H; WRITE
        dd      DSK_VERF                ; AH = 04H; VERIFY
        dd      DSK_FORMAT              ; AH = 05H; FORMAT
        dd      FNC_ERR                 ; AH = 06H; INVALID
        dd      FNC_ERR                 ; AH = 07H; INVALID
        dd      DSK_PARMS               ; AH = 08H; READ DRIVE PARAMETERS
        dd      FNC_ERR                 ; AH = 09H; INVALID
        dd      FNC_ERR                 ; AH = 0AH; INVALID
        dd      FNC_ERR                 ; AH = 0BH; INVALID
        dd      FNC_ERR                 ; AH = 0CH; INVALID
        dd      FNC_ERR                 ; AH = 0DH; INVALID
        dd      FNC_ERR                 ; AH = 0EH; INVALID
        dd      FNC_ERR                 ; AH = 0FH; INVALID
        dd      FNC_ERR                 ; AH = 10H; INVALID
        dd      FNC_ERR                 ; AH = 11H; INVALID
        dd      FNC_ERR                 ; AH = 12H; INVALID
        dd      FNC_ERR                 ; AH = 13H; INVALID
        dd      FNC_ERR                 ; AH = 14H; INVALID
        dd      DSK_TYPE                ; AH = 15H; READ DASD TYPE
        dd      DSK_CHANGE              ; AH = 16H; CHANGE STATUS
        dd      FORMAT_SET              ; AH = 17H; SET DASD TYPE
        dd      SET_MEDIA               ; AH = 18H; SET MEDIA TYPE
FNC_TAE EQU     $                       ; END

;--------------------------------------------------------------------------------
; DISK_RESET    (AH = 00H)
;               RESET THE DISKETTE SYSTEM.
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
DSK_RESET:
        MOV     DX,03F2H                ; ADAPTER CONTROL PORT
        CLI                             ; NO INTERRUPTS
        MOV     AL,[MOTOR_STATUS]       ; GET DIGITAL OUTPUT REGISTER REFLECTION
        AND     AL,00111111B            ; KEEP SELECTED AND MOTOR ON BITS
        ROL     AL,4                    ; MOTOR VALUE TO HIGH NIBBLE
                                        ; DRIVE SELECT TO LOW NIBBLE
        OR      AL,00001000B            ; TURN ON INTERRUPT ENABLE
        OUT     DX,AL                   ; RESET THE ADAPTER
        MOV     byte [SEEK_STATUS],0    ; SET RECALIBRATE REQUIRED ON ALL DRIVES
        ;JMP    $+2                     ; WAIT FOR I/O
        ;JMP    $+2                     ; WAIT FOR I/O (TO INSURE MINIMUM
                                        ;      PULSE WIDTH)
        ; 19/12/2014
        NEWIODELAY

        ; 17/12/2014
        ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
;       mov     cx, WAITCPU_RESET_ON    ; cx = 21 -- Min. 14 micro seconds !?
;wdw1:
;       NEWIODELAY
;       loop    wdw1
        ;
        OR      AL,00000100B            ; TURN OFF RESET BIT
        OUT     DX,AL                   ; RESET THE ADAPTER
        ; 16/12/2014
        IODELAY
        ;
        ;STI                            ; ENABLE THE INTERRUPTS
        CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
```

```
        JC      short DR_ERR            ; IF ERROR, RETURN IT
        MOV     CX,11000000B            ; CL = EXPECTED @NEC_STATUS
NXT_DRV:
        PUSH    CX                      ; SAVE FOR CALL
        MOV     eAX, DR_POP_ERR         ; LOAD NEC_OUTPUT ERROR ADDRESS
        PUSH    eAX                     ; "
        MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
        CALL    NEC_OUTPUT
        POP     eAX                     ; THROW AWAY ERROR RETURN
        CALL    RESULTS                 ; READ IN THE RESULTS
        POP     CX                      ; RESTORE AFTER CALL
        JC      short DR_ERR            ; ERROR RETURN
        CMP     CL, [NEC_STATUS]        ; TEST FOR DRIVE READY TRANSITION
        JNZ     short DR_ERR            ; EVERYTHING OK
        INC     CL                      ; NEXT EXPECTED @NEC_STATUS
        CMP     CL,11000011B            ; ALL POSSIBLE DRIVES CLEARED
        JBE     short NXT_DRV           ; FALL THRU IF 11000100B OR >
        ;
        CALL    SEND_SPEC               ; SEND SPECIFY COMMAND TO NEC
RESBAC:
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        MOV     BX,SI                   ; GET SAVED AL TO BL
        MOV     AL,BL                   ; PUT BACK FOR RETURN
        RETn
DR_POP_ERR:
        POP     CX                      ; CLEAR STACK
DR_ERR:
        OR      byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
        JMP     SHORT RESBAC            ; RETURN FROM RESET

;--------------------------------------------------------------------------
; DISK_STATUS   (AH = 01H)
;       DISKETTE STATUS.
;
; ON ENTRY:     AH : STATUS OF PREVIOUS OPERATION
;
; ON EXIT:      AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
;--------------------------------------------------------------------------
DSK_STATUS:
        MOV     [DSKETTE_STATUS],AH     ; PUT BACK FOR SETUP END
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        MOV     BX,SI                   ; GET SAVED AL TO BL
        MOV     AL,BL                   ; PUT BACK FOR RETURN
        RETn

;--------------------------------------------------------------------------
; DISK_READ     (AH = 02H)
;       DISKETTE READ.
;
; ON ENTRY:     DI      : DRIVE #
;               SI-HI   : HEAD #
;               SI-LOW  : # OF SECTORS
;               ES      : BUFFER SEGMENT
;               [BP]    : SECTOR #
;               [BP+1]  : TRACK #
;               [BP+2]  : BUFFER OFFSET
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------

; 06/02/2015, ES:BX -> EBX (unix386.s)

DSK_READ:
        AND     byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
```

```
        MOV     AX,0E646H               ; AX = NEC COMMAND, DMA COMMAND
        CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
        RETn

;--------------------------------------------------------------------------------
; DISK_WRITE    (AH = 03H)
;       DISKETTE WRITE.
;
; ON ENTRY:     DI      : DRIVE #
;               SI-HI   : HEAD #
;               SI-LOW  : # OF SECTORS
;               ES      : BUFFER SEGMENT
;               [BP]    : SECTOR #
;               [BP+1]  : TRACK #
;               [BP+2]  : BUFFER OFFSET
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------

; 06/02/2015, ES:BX -> EBX (unix386.s)

DSK_WRITE:
        MOV     AX,0C54AH               ; AX = NEC COMMAND, DMA COMMAND
        OR      byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
        CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
        RETn

;--------------------------------------------------------------------------------
; DISK_VERF     (AH = 04H)
;       DISKETTE VERIFY.
;
; ON ENTRY:     DI      : DRIVE #
;               SI-HI   : HEAD #
;               SI-LOW  : # OF SECTORS
;               ES      : BUFFER SEGMENT
;               [BP]    : SECTOR #
;               [BP+1]  : TRACK #
;               [BP+2]  : BUFFER OFFSET
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
DSK_VERF:
        AND     byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
        MOV     AX,0E642H               ; AX = NEC COMMAND, DMA COMMAND
        CALL    RD_WR_VF                ; COMMON READ/WRITE/VERIFY
        RETn

;--------------------------------------------------------------------------------
; DISK_FORMAT   (AH = 05H)
;       DISKETTE FORMAT.
;
; ON ENTRY:     DI      : DRIVE #
;               SI-HI   : HEAD #
;               SI-LOW  : # OF SECTORS
;               ES      : BUFFER SEGMENT
;               [BP]    : SECTOR #
;               [BP+1]  : TRACK #
;               [BP+2]  : BUFFER OFFSET
;               @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
DSK_FORMAT:
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
```

```
        CALL    FMT_INIT                ; ESTABLISH STATE IF UNESTABLISHED
        OR      byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
        CALL    MED_CHANGE              ; CHECK MEDIA CHANGE AND RESET IF SO
        JC      short FM_DON            ; MEDIA CHANGED, SKIP
        CALL    SEND_SPEC               ; SEND SPECIFY COMMAND TO NEC
        CALL    CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
        JZ      short FM_WR             ; YES, SKIP SPECIFY COMMAND
        CALL    SEND_RATE               ; SEND DATA RATE TO CONTROLLER
FM_WR:
        CALL    FMTDMA_SET              ; SET UP THE DMA FOR FORMAT
        JC      short FM_DON            ; RETURN WITH ERROR
        MOV     AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
        CALL    NEC_INIT                ; INITIALIZE THE NEC
        JC      short FM_DON            ; ERROR - EXIT
        MOV     eAX, FM_DON             ; LOAD ERROR ADDRESS
        PUSH    eAX                     ; PUSH NEC_OUT ERROR RETURN
        MOV     DL,3                    ; BYTES/SECTOR VALUE TO NEC
        CALL    GET_PARM
        CALL    NEC_OUTPUT
        MOV     DL,4                    ; SECTORS/TRACK VALUE TO NEC
        CALL    GET_PARM
        CALL    NEC_OUTPUT
        MOV     DL,7                    ; GAP LENGTH VALUE TO NEC
        CALL    GET_PARM
        CALL    NEC_OUTPUT
        MOV     DL,8                    ; FILLER BYTE TO NEC
        CALL    GET_PARM
        CALL    NEC_OUTPUT
        POP     eAX                     ; THROW AWAY ERROR
        CALL    NEC_TERM                ; TERMINATE, RECEIVE STATUS, ETC,
FM_DON:
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        MOV     BX,SI                   ; GET SAVED AL TO BL
        MOV     AL,BL                   ; PUT BACK FOR RETURN
        RETn

;--------------------------------------------------------------------------
; FNC_ERR
;       INVALID FUNCTION REQUESTED OR INVALID DRIVE:
;       SET BAD COMMAND IN STATUS.
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------
FNC_ERR:                                ; INVALID FUNCTION REQUEST
        MOV     AX,SI                   ; RESTORE AL
        MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
        MOV     [DSKETTE_STATUS],AH     ; STORE IN DATA AREA
        STC                             ; SET CARRY INDICATING ERROR
        RETn

;--------------------------------------------------------------------------
; DISK_PARMS    (AH = 08H)
;       READ DRIVE PARAMETERS.
;
; ON ENTRY:     DI : DRIVE #
;
; ON EXIT:      CL/[BP]   = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
;                           BITS 0-5 MAX SECTORS/TRACK
;               CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
;               BL/[BP+2] = BITS 7-4 = 0
;                           BITS 3-0 = VALID CMOS DRIVE TYPE
;               BH/[BP+3] = 0
;               DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
```

```
;               DH/[BP+5] = MAX HEAD #
;               DI/[BP+6] = OFFSET TO DISK_BASE
;               ES        = SEGMENT OF DISK_BASE
;               AX        = 0
;
;               NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
;                      THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
;                      INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
;                      CALLER.
;--------------------------------------------------------------------------------
DSK_PARMS:
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH,
     ;  MOV     WORD [BP+2],0           ; DRIVE TYPE = 0
        sub     edx, edx ; 20/02/2015
        mov     [ebp+4], edx ; 20/02/2015
     ;  MOV     AX, [EQUIP_FLAG]        ; LOAD EQUIPMENT FLAG FOR # DISKETTES
     ;  AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
     ;  MOV     DL,2                    ; DISKETTE DRIVES = 2
     ;  CMP     AL,01000001B            ; 2 DRIVES INSTALLED ?
     ;  JZ      short STO_DL            ; IF YES JUMP
     ;  DEC     DL                      ; DISKETTE DRIVES = 1
     ;  CMP     AL,00000001B            ; 1 DRIVE INSTALLED ?
     ;  JNZ     short NON_DRV           ; IF NO JUMP
        ;sub    edx, edx
        mov     ax, [fd0_type]
        and     ax, ax
        jz      short NON_DRV
        inc     dl
        and     ah, ah
        jz      short STO_DL
        inc     dl
STO_DL:
        ;MOV    [BP+4],DL               ; STORE NUMBER OF DRIVES
        mov     [ebp+8], edx ; 20/02/2015
        CMP     DI,1                    ; CHECK FOR VALID DRIVE
        JA      short NON_DRV1          ; DRIVE INVALID
        ;MOV    BYTE [BP+5],1           ; MAXIMUM HEAD NUMBER = 1
        mov     byte [ebp+9], 1  ; 20/02/2015
        CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL
        ;;20/02/2015
        ;;JC    short CHK_EST           ; IF CMOS BAD CHECKSUM ESTABLISHED
        ;;OR    AL,AL                   ; TEST FOR NO DRIVE TYPE
        JZ      short CHK_EST           ; JUMP IF SO
        CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
        JC      short CHK_EST           ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
        ;MOV    [BP+2],AL               ; STORE VALID CMOS DRIVE TYPE
        mov     [ebp+4], al ; 06/02/2015
        MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
        MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
        JMP     SHORT STO_CX            ; CMOS GOOD, USE CMOS
CHK_EST:
        MOV     AH, [DSK_STATE+eDI]     ; LOAD STATE FOR THIS DRIVE
        TEST    AH,MED_DET              ; CHECK FOR ESTABLISHED STATE
        JZ      short NON_DRV1          ; CMOS BAD/INVALID OR UNESTABLISHED
USE_EST:
        AND     AH,RATE_MSK             ; ISOLATE STATE
        CMP     AH,RATE_250             ; RATE 250 ?
        JNE     short USE_EST2          ; NO, GO CHECK OTHER RATE

;-----  DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST

        MOV     AL,01                   ; DRIVE TYPE 1 (360KB)
        CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
        MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
```

```
        MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
        TEST    byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
        JZ      short STO_CX            ; MUST BE 360KB DRIVE

;-----  IT IS 1.44 MB DRIVE

PARM144:
        MOV     AL,04                   ; DRIVE TYPE 4 (1.44MB)
        CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
        MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
        MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
STO_CX:
        MOV     [eBP],eCX               ; SAVE POINTER IN STACK FOR RETURN
ES_DI:
        ;MOV    [BP+6],BX               ; ADDRESS OF MEDIA/DRIVE PARM TABLE
        mov     [ebp+12], ebx ; 06/02/2015
        ;MOV    AX,CS                   ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
        ;MOV    ES,AX                   ; ES IS SEGMENT OF TABLE
DP_OUT:
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        XOR     AX,AX                   ; CLEAR
        CLC
        RETn

;-----  NO DRIYE PRESENT HANDLER

NON_DRV:
        ;MOV    BYTE [BP+4],0           ; CLEAR NUMBER OF DRIVES
        mov     [ebp+8], edx ; 0 ; 20/02/2015
NON_DRV1:
        CMP     DI,80H                  ; CHECK FOR FIXED MEDIA TYPE REQUEST
        JB      short NON_DRV2          ; CONTINUE IF NOT REQUEST FALL THROUGH

;-----  FIXED DISK REQUEST FALL THROUGH ERROR

        CALL    XLAT_OLD                ; ELSE TRANSLATE TO COMPATIBLE MODE
        MOV     AX,SI                   ; RESTORE AL
        MOV     AH,BAD_CMD              ; SET BAD COMMAND ERROR
        STC
        RETn

NON_DRV2:
        ;XOR    AX,AX                   ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
        xor     eax, eax
        MOV     [eBP],AX                ; TRACKS, SECTORS/TRACK = 0
        ;MOV    [BP+5],AH               ; HEAD = 0
        mov     [ebp+9], ah ; 06/02/2015
        ;MOV    [BP+6],AX               ; OFFSET TO DISK_BASE = 0
        mov     [ebp+12], eax
        ;MOV    ES,AX                   ; ES IS SEGMENT OF TABLE
        JMP     SHORT DP_OUT

;-----  DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST

USE_EST2:
        MOV     AL,02                   ; DRIVE TYPE 2 (1.2MB)
        CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
        MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
        MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
        CMP     AH,RATE_300             ; RATE 300 ?
        JZ      short STO_CX            ; MUST BE 1.2MB DRIVE
        JMP     SHORT PARM144           ; ELSE, IT IS 1.44MB DRIVE

;--------------------------------------------------------------------------
```

```
; DISK_TYPE (AH = 15H)
;       THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
;
;  ON ENTRY:    DI = DRIVE #
;
;  ON EXIT:     AH = DRIVE TYPE, CY=0
;--------------------------------------------------------------------------------
DSK_TYPE:
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
        MOV     AL, [DSK_STATE+eDI]     ; GET PRESENT STATE INFORMATION
        OR      AL,AL                   ; CHECK FOR NO DRIVE
        JZ      short NO_DRV
        MOV     AH,NOCHGLN              ; NO CHANGE LINE FOR 40 TRACK DRIVE
        TEST    AL,TRK_CAPA             ; IS THIS DRIVE AN 80 TRACK DRIVE?
        JZ      short DT_BACK                   ; IF NO JUMP
        MOV     AH,CHGLN                ; CHANGE LINE FOR 80 TRACK DRIVE
DT_BACK:
        PUSH    AX                      ; SAVE RETURN VALUE
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        POP     AX                      ; RESTORE RETURN VALUE
        CLC                             ; NO ERROR
        MOV     BX,SI                   ; GET SAVED AL TO BL
        MOV     AL,BL                   ; PUT BACK FOR RETURN
        RETn
NO_DRV:
        XOR     AH,AH                   ; NO DRIVE PRESENT OR UNKNOWN
        JMP     SHORT DT_BACK

;--------------------------------------------------------------------------------
; DISK_CHANGE   (AH = 16H)
;       THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
;
; ON ENTRY:     DI = DRIVE #
;
; ON EXIT:      AH = @DSKETTE_STATUS
;                    00 - DISK CHANGE LINE INACTIVE, CY = 0
;                    06 - DISK CHANGE LINE ACTIVE, CY = 1
;--------------------------------------------------------------------------------
DSK_CHANGE:
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
        MOV     AL, [DSK_STATE+eDI]     ; GET MEDIA STATE INFORMATION
        OR      AL,AL                   ; DRIVE PRESENT ?
        JZ      short DC_NON            ; JUMP IF NO DRIVE
        TEST    AL,TRK_CAPA             ; 80 TRACK DRIVE ?
        JZ      short SETIT             ; IF SO , CHECK CHANGE LINE
DC0:
        CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
        JZ      short FINIS             ; CHANGE LINE NOT ACTIVE

SETIT:  MOV     byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED

FINIS:  CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        MOV     BX,SI                   ; GET SAVED AL TO BL
        MOV     AL,BL                   ; PUT BACK FOR RETURN
        RETn
DC_NON:
        OR      byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
        JMP     SHORT FINIS

;--------------------------------------------------------------------------------
; FORMAT_SET    (AH = 17H)
;       THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
;       FOR THE FOLLOWING FORMAT OPERATION.
```

```
;
; ON ENTRY:     SI LOW = DASD TYPE FOR FORMAT
;               DI     = DRIVE #
;
; ON EXIT:      @DSKETTE_STATUS REFLECTS STATUS
;               AH = @DSKETTE_STATUS
;               CY = 1 IF ERROR
;--------------------------------------------------------------------------------
FORMAT_SET:
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
        PUSH    SI                      ; SAVE DASD TYPE
        MOV     AX,SI                   ; AH = ? , AL , DASD TYPE
        XOR     AH,AH                   ; AH , 0 , AL , DASD TYPE
        MOV     SI,AX                   ; SI = DASD TYPE
        AND     byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
        DEC     SI                      ; CHECK FOR 320/360K MEDIA & DRIVE
        JNZ     short NOT_320           ; BYPASS IF NOT
        OR      byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
        JMP     SHORT S0

NOT_320:
        CALL    MED_CHANGE              ; CHECK FOR TIME_OUT
        CMP     byte [DSKETTE_STATUS], TIME_OUT
        JZ      short S0                ; IF TIME OUT TELL CALLER
S3:
        DEC     SI                      ; CHECK FOR 320/360K IN 1.2M DRIVE
        JNZ     short NOT_320_12        ; BYPASS IF NOT
        OR      byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300 ; SET STATE
        JMP     SHORT S0

NOT_320_12:
        DEC     SI                      ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
        JNZ     short NOT_12            ; BYPASS IF NOT
        OR      byte [DSK_STATE+eDI], MED_DET+RATE_500 ; SET STATE VARIABLE
        JMP     SHORT S0                ; RETURN TO CALLER

NOT_12:
        DEC     SI                      ; CHECK FOR SET DASD TYPE 04
        JNZ     short FS_ERR            ; BAD COMMAND EXIT IF NOT VALID TYPE

        TEST    byte [DSK_STATE+eDI], DRV_DET ; DRIVE DETERMINED ?
        JZ      short ASSUME            ; IF STILL NOT DETERMINED ASSUME
        MOV     AL,MED_DET+RATE_300
        TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
        JNZ     short OR_IT_IN          ; IF 1.2 M THEN DATA RATE 300

ASSUME:
        MOV     AL,MED_DET+RATE_250     ; SET UP

OR_IT_IN:
        OR      byte [DSK_STATE+eDI], AL        ; OR IN THE CORRECT STATE
S0:
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        POP     BX                      ; GET SAVED AL TO BL
        MOV     AL,BL                   ; PUT BACK FOR RETURN
        RETn

FS_ERR:
        MOV     byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
        JMP     SHORT S0

;--------------------------------------------------------------------------------
; SET_MEDIA     (AH = 18H)
```

```
;       THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
;       TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
;
; ON ENTRY:
;       [BP]    = SECTOR PER TRACK
;       [BP+1]  = TRACK #
;       DI      = DRIVE #
;
; ON EXIT:
;       @DSKETTE_STATUS REFLECTS STATUS
;       IF NO ERROR:
;               AH = 0
;               CY = 0
;               ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
;               DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
;       IF ERROR:
;               AH = @DSKETTE_STATUS
;               CY = 1
;--------------------------------------------------------------------------------
SET_MEDIA:
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
        TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
        JZ      short SM_CMOS           ; JUMP IF 40 TRACK DRIVE
        CALL    MED_CHANGE              ; RESET CHANGE LINE
        CMP     byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
        JE      short SM_RTN
        MOV     byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
SM_CMOS:
        CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
        ;;20/02/2015
        ;;JC    short MD_NOT_FND        ; ERROR IN CMOS
        ;;OR    AL,AL                   ; TEST FOR NO DRIVE
        JZ      short SM_RTN            ; RETURN IF SO
        CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
        JC      short MD_NOT_FND        ; TYPE NOT IN TABLE (BAD CMOS)
        PUSH    eDI                     ; SAVE REG.
        XOR     eBX,eBX                 ; BX = INDEX TO DR. TYPE TABLE
        MOV     eCX,DR_CNT              ; CX = LOOP COUNT
DR_SEARCH:
        MOV     AH, [DR_TYPE+eBX]       ; GET DRIVE TYPE
        AND     AH,BIT7OFF              ; MASK OUT MSB
        CMP     AL,AH                   ; DRIVE TYPE MATCH ?
        JNE     short NXT_MD            ; NO, CHECK NEXT DRIVE TYPE
DR_FND:
        MOV     eDI, [DR_TYPE+eBX+1]    ; DI = MEDIA/DRIVE PARAM TABLE
MD_SEARCH:
        MOV     AH, [eDI+MD.SEC_TRK]    ; GET SECTOR/TRACK
        CMP     [eBP],AH                ; MATCH?
        JNE     short NXT_MD            ; NO, CHECK NEXT MEDIA
        MOV     AH, [eDI+MD.MAX_TRK]    ; GET MAX. TRACK #
        CMP     [eBP+1],AH              ; MATCH?
        JE      short MD_FND            ; YES, GO GET RATE
NXT_MD:
        ;ADD    BX,3                    ; CHECK NEXT DRIVE TYPE
        add     ebx, 5 ; 18/02/2015
        LOOP    DR_SEARCH
        POP     eDI                     ; RESTORE REG.
MD_NOT_FND:
        MOV     byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
        JMP     SHORT SM_RTN            ; RETURN
MD_FND:
        MOV     AL, [eDI+MD.RATE]       ; GET RATE
        CMP     AL,RATE_300             ; DOUBLE STEP REQUIRED FOR RATE 300
        JNE     short MD_SET
```

```
        OR      AL,DBL_STEP
MD_SET:
        ;MOV    [BP+6],DI               ; SAVE TABLE POINTER IN STACK
        mov     [ebp+12], edi ; 18/02/2015
        OR      AL,MED_DET              ; SET MEDIA ESTABLISHED
        POP     eDI
        AND     byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
        OR      byte [DSK_STATE+eDI], AL
        ;MOV    AX, CS                  ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
        ;MOV    ES, AX                  ; ES IS SEGMENT OF TABLE
SM_RTN:
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        RETn

;--------------------------------------------------------------
; DR_TYPE_CHECK                                                :
;       CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)         :
;       IS SUPPORTED IN BIOS DRIVE TYPE TABLE                  :
; ON ENTRY:                                                    :
;       AL = DRIVE TYPE                                        :
; ON EXIT:                                                     :
;       CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE)        :
;       CY = 0  DRIVE TYPE SUPPORTED                           :
;            BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE        :
;       CY = 1  DRIVE TYPE NOT SUPPORTED                       :
; REGISTERS ALTERED: BX                                        :
;--------------------------------------------------------------
DR_TYPE_CHECK:
        PUSH    AX
        PUSH    eCX
        XOR     eBX,eBX                 ; BX = INDEX TO DR_TYPE TABLE
        MOV     eCX,DR_CNT              ; CX = LOOP COUNT
TYPE_CHK:
        MOV     AH,[DR_TYPE+eBX]        ; GET DRIVE TYPE
        CMP     AL,AH                   ; DRIVE TYPE MATCH?
        JE      short DR_TYPE_VALID     ; YES, RETURN WITH CARRY RESET
        ;ADD    BX,3                    ; CHECK NEXT DRIVE TYPE
        add     ebx, 5  ; 16/02/2015 (32 bit address modification)
        LOOP    TYPE_CHK
        ;
        mov     ebx, MD_TBL6            ; 1.44MB fd parameter table
                                        ; Default for GET_PARM (11/12/2014)
        ;
        STC                             ; DRIVE TYPE NOT FOUND IN TABLE
        JMP     SHORT TYPE_RTN
DR_TYPE_VALID:
        MOV     eBX,[DR_TYPE+eBX+1]     ; BX = MEDIA TABLE
TYPE_RTN:
        POP     eCX
        POP     AX
        RETn

;--------------------------------------------------------------
; SEND_SPEC                                                    :
;       SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM  :
;       THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER      :
; ON ENTRY:     @DISK_POINTER = DRIVE PARAMETER TABLE          :
; ON EXIT:      NONE                                           :
; REGISTERS ALTERED: CX, DX                                    :
;--------------------------------------------------------------
SEND_SPEC:
        PUSH    eAX                     ; SAVE AX
        MOV     eAX, SPECBAC            ; LOAD ERROR ADDRESS
```

```
        PUSH    eAX                     ; PUSH NEC_OUT ERROR RETURN
        MOV     AH,03H                  ; SPECIFY COMMAND
        CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
        SUB     DL,DL                   ; FIRST SPECIFY BYTE
        CALL    GET_PARM                ; GET PARAMETER TO AH
        CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
        MOV     DL,1                    ; SECOND SPECIFY BYTE
        CALL    GET_PARM                ; GET PARAMETER TO AH
        CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
        POP     eAX                     ; POP ERROR RETURN
SPECBAC:
        POP     eAX                     ; RESTORE ORIGINAL AX VALUE
        RETn

;--------------------------------------------------------------
; SEND_SPEC_MD                                                 :
;       SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM  :
;       THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)      :
; ON ENTRY:     CS:BX = MEDIA/DRIVE PARAMETER TABLE             :
; ON EXIT:      NONE                                            :
; REGISTERS ALTERED: AX                                         :
;--------------------------------------------------------------
SEND_SPEC_MD:
        PUSH    eAX                     ; SAVE RATE DATA
        MOV     eAX, SPEC_ESBAC         ; LOAD ERROR ADDRESS
        PUSH    eAX                     ; PUSH NEC_OUT ERROR RETURN
        MOV     AH,03H                  ; SPECIFY COMMAND
        CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
        MOV     AH, [eBX+MD.SPEC1]      ; GET 1ST SPECIFY BYTE
        CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
        MOV     AH, [eBX+MD.SPEC2]      ; GET SECOND SPECIFY BYTE
        CALL    NEC_OUTPUT              ; OUTPUT THE COMMAND
        POP     eAX                     ; POP ERROR RETURN
SPEC_ESBAC:
        POP     eAX                     ; RESTORE ORIGINAL AX VALUE
        RETn

;-----------------------------------------------------------------------------
; XLAT_NEW
;       TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
;       MODE TO NEW ARCHITECTURE.
;
; ON ENTRY:     DI = DRIVE #
;-----------------------------------------------------------------------------
XLAT_NEW:
        CMP     eDI,1                           ; VALID DRIVE
        JA      short XN_OUT                    ; IF INVALID BACK
        CMP     byte [DSK_STATE+eDI], 0         ; NO DRIVE ?
        JZ      short DO_DET                    ; IF NO DRIVE ATTEMPT DETERMINE
        MOV     CX,DI                           ; CX = DRIVE NUMBER
        SHL     CL,2                            ; CL = SHIFT COUNT, A=0, B=4
        MOV     AL, [HF_CNTRL]                  ; DRIVE INFORMATION
        ROR     AL,CL                           ; TO LOW NIBBLE
        AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA    ; KEEP DRIVE BITS
        AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
        OR      [DSK_STATE+eDI], AL             ; UPDATE DRIVE STATE
XN_OUT:
        RETn
DO_DET:
        CALL    DRIVE_DET                       ; TRY TO DETERMINE
        RETn

;-----------------------------------------------------------------------------
; XLAT_OLD
```

```
;       TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
;       ARCHITECTURE TO COMPATIBLE MODE.
;
; ON ENTRY:     DI = DRIVE
;--------------------------------------------------------------------------------
XLAT_OLD:
        CMP     eDI,1                   ; VALID DRIVE ?
        ;JA     short XO_OUT            ; IF INVALID BACK
        ja      XO_OUT
        CMP     byte [DSK_STATE+eDI],0  ; NO DRIVE ?
        JZ      short XO_OUT            ; IF NO DRIVE TRANSLATE DONE

;-----  TEST FOR SAVED DRIVE INFORMATION ALREADY SET

        MOV     CX,DI                   ; CX = DRIVE NUMBER
        SHL     CL,2                    ; CL = SHIFT COUNT, A=0, B=4
        MOV     AH,FMT_CAPA             ; LOAD MULTIPLE DATA RATE BIT MASK
        ROR     AH,CL                   ; ROTATE BY MASK
        TEST    [HF_CNTRL], AH          ; MULTIPLE-DATA RATE DETERMINED ?
        JNZ     short SAVE_SET          ; IF SO, NO NEED TO RE-SAVE

;-----  ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE

        MOV     AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
        ROR     AH,CL                   ; FIX MASK TO KEEP
        NOT     AH                      ; TRANSLATE MASK
        AND     [HF_CNTRL], AH          ; KEEP BITS FROM OTHER DRIVE INTACT

;-----  ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL

        MOV     AL, [DSK_STATE+eDI]     ; ACCESS STATE
        AND     AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
        ROR     AL,CL                   ; FIX FOR THIS DRIVE
        OR      [HF_CNTRL], AL          ; UPDATE SAVED DRIVE STATE

;-----  TRANSLATE TO COMPATIBILITY MODE

SAVE_SET:
        MOV     AH, [DSK_STATE+eDI]     ; ACCESS STATE
        MOV     BH,AH                   ; TO BH FOR LATER
        AND     AH,RATE_MSK             ; KEEP ONLY RATE
        CMP     AH,RATE_500             ; RATE 500 ?
        JZ      short CHK_144           ; YES 1.2/1.2 OR 1.44/1.44
        MOV     AL,M3D1U                ; AL = 360 IN 1.2 UNESTABLISHED
        CMP     AH,RATE_300             ; RATE 300 ?
        JNZ     short CHK_250           ; NO, 360/360, 720/720 OR 720/1.44
        TEST    BH,DBL_STEP             ; CHECK FOR DOUBLE STEP
        JNZ     short TST_DET           ; MUST BE 360 IN 1.2
UNKNO:
        MOV     AL,MED_UNK              ; NONE OF THE ABOVE
        JMP     SHORT AL_SET            ; PROCESS COMPLETE
CHK_144:
        CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
        ;;20/02/2015
        ;;JC    short UNKNO             ; ERROR, SET 'NONE OF ABOVE'
        jz      short UNKNO ;; 20/02/2015
        CMP     AL,2                    ; 1.2MB DRIVE ?
        JNE     short UNKNO             ; NO, GO SET 'NONE OF ABOVE'
        MOV     AL,M1D1U                ; AL = 1.2 IN 1.2 UNESTABLISHED
        JMP     SHORT TST_DET
CHK_250:
        MOV     AL,M3D3U                ; AL = 360 IN 360 UNESTABLISHED
        CMP     AH,RATE_250             ; RATE 250 ?
        JNZ     short UNKNO             ; IF SO FALL IHRU
```

```
        TEST    BH,TRK_CAPA             ; 80 TRACK CAPABILITY ?
        JNZ     short UNKNO             ; IF SO JUMP, FALL THRU TEST DET
TST_DET:
        TEST    BH,MED_DET              ; DETERMINED ?
        JZ      short AL_SET            ; IF NOT THEN SET
        ADD     AL,3                    ; MAKE DETERMINED/ESTABLISHED
AL_SET:
        AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
        OR      [DSK_STATE+eDI], AL     ; REPLACE WITH COMPATIBLE MODE
XO_OUT:
        RETn

;--------------------------------------------------------------------------------
; RD_WR_VF
;       COMMON READ, WRITE AND VERIFY:
;       MAIN LOOP FOR STATE RETRIES.
;
; ON ENTRY:     AH = READ/WRITE/VERIFY NEC PARAMETER
;               AL = READ/WRITE/VERIFY DMA PARAMETER
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
RD_WR_VF:
        PUSH    AX                      ; SAVE DMA, NEC PARAMETERS
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH.
        CALL    SETUP_STATE             ; INITIALIZE START AND END RATE
        POP     AX                      ; RESTORE READ/WRITE/VERIFY
DO_AGAIN:
        PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER
        CALL    MED_CHANGE              ; MEDIA CHANGE AND RESET IF CHANGED
        POP     AX                      ; RESTORE READ/WRITE/VERIFY
        JC      RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
RWV:
        PUSH    AX                      ; SAVE READ/WRITE/VERIFY PARAMETER
        MOV     DH, [DSK_STATE+eDI]     ; GET RATE STATE OF THIS DRIVE
        AND     DH,RATE_MSK             ; KEEP ONLY RATE
        CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL (AL)
        ;;20/02/2015
        ;;JC    short RWV_ASSUME        ; ERROR IN CMOS
        jz      short RWV_ASSUME ; 20/02/2015
        CMP     AL,1                    ; 40 TRACK DRIVE?
        JNE     short RWV_1             ; NO, BYPASS CMOS VALIDITY CHECK
        TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
        JZ      short RWV_2             ; YES, CMOS IS CORRECT
        MOV     AL,2                    ; CHANGE TO 1.2M
        JMP     SHORT RWV_2
RWV_1:
        JB      short RWV_2             ; NO DRIVE SPECIFIED, CONTINUE
        TEST    byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
        JNZ     short RWV_2             ; NO, 80 TRACK
        MOV     AL,1                    ; IT IS 40 TRACK, FIX CMOS VALUE
        jmp     short rwv_3
RWV_2:
        OR      AL,AL                   ; TEST FOR NO DRIVE
        JZ      short RWV_ASSUME        ; ASSUME TYPE, USE MAX TRACK
rwv_3:
        CALL    DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
        JC      short RWV_ASSUME        ; TYPE NOT IN TABLE (BAD CMOS)

;-----  SEARCH FOR MEDIA/DRIVE PARAMETER TABLE

        PUSH    eDI                     ; SAVE DRIVE #
        XOR     eBX,eBX                 ; BX = INDEX TO DR_TYPE TABLE
        MOV     eCX,DR_CNT              ; CX = LOOP COUNT
```

```
RWV_DR_SEARCH:
        MOV     AH, [DR_TYPE+eBX]       ; GET DRIVE TYPE
        AND     AH,BIT7OFF              ; MASK OUT MSB
        CMP     AL,AH                   ; DRIVE TYPE MATCH?
        JNE     short RWV_NXT_MD        ; NO, CHECK NEXT DRIVE TYPE
RWV_DR_FND:
        MOV     eDI, [DR_TYPE+eBX+1]    ; DI = MEDIA/DRIVE PARAMETER TABLE
RWV_MD_SEARH:
        CMP     DH, [eDI+MD.RATE]       ; MATCH?
        JE      short RWV_MD_FND        ; YES, GO GET 1ST SPECIFY BYTE
RWV_NXT_MD:
        ;ADD    BX,3                    ; CHECK NEXT DRIVE TYPE
        add     eBX, 5
        LOOP    RWV_DR_SEARCH
        POP     eDI                     ; RESTORE DRIVE #

;-----  ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED

RWV_ASSUME:
        MOV     eBX, MD_TBL1            ; POINT TO 40 TRACK 250 KBS
        TEST    byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
        JZ      short RWV_MD_FND1       ; MUST BE 40 TRACK
        MOV     eBX, MD_TBL3            ; POINT TO 80 TRACK 500 KBS
        JMP     short RWV_MD_FND1       ; GO SPECIFY PARAMTERS

;-----  CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE

RWV_MD_FND:
        MOV     eBX,eDI                 ; BX = MEDIA/DRIVE PARAMETER TABLE
        POP     eDI                     ; RESTORE DRIVE #

;-----  SEND THE SPECIFY COMMAND TO THE CONTROLLER

RWV_MD_FND1:
        CALL    SEND_SPEC_MD
        CALL    CHK_LASTRATE            ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
        JZ      short RWV_DBL           ; YES,SKIP SEND RATE COMMAND
        CALL    SEND_RATE               ; SEND DATA RATE TO NEC
RWV_DBL:
        PUSH    eBX                     ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
        CALL    SETUP_DBL               ; CHECK FOR DOUBLE STEP
        POP     eBX                     ; RESTORE ADDRESS
        JC      short CHK_RET           ; ERROR FROM READ ID, POSSIBLE RETRY
        POP     AX                      ; RESTORE NEC, DMA COMMAND
        PUSH    AX                      ; SAVE NEC COMMAND
        PUSH    eBX                     ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
        CALL    DMA_SETUP               ; SET UP THE DMA
        POP     eBX
        POP     AX                      ; RESTORE NEC COMMAND
        JC      short RWV_BAC           ; CHECK FOR DMA BOUNDARY ERROR
        PUSH    AX                      ; SAVE NEC COMMAND
        PUSH    eBX                     ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
        CALL    NEC_INIT                ; INITIALIZE NEC
        POP     eBX                     ; RESTORE ADDRESS
        JC      short CHK_RET           ; ERROR - EXIT
        CALL    RWV_COM                 ; OP CODE COMMON TO READ/WRITE/VERIFY
        JC      short CHK_RET           ; ERROR - EXIT
        CALL    NEC_TERM                ; TERMINATE, GET STATUS, ETC.
CHK_RET:
        CALL    RETRY                   ; CHECK FOR, SETUP RETRY
        POP     AX                      ; RESTORE READ/WRITE/VERIFY PARAMETER
        JNC     short RWV_END           ; CY = 0 NO RETRY
        JMP     DO_AGAIN                ; CY = 1 MEANS RETRY
RWV_END:
```

```
        CALL    DSTATE                  ; ESTABLISH STATE IF SUCCESSFUL
        CALL    NUM_TRANS               ; AL = NUMBER TRANSFERRED
RWV_BAC:                                ; BAD DMA ERROR ENTRY
        PUSH    AX                      ; SAVE NUMBER TRANSFERRED
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        POP     AX                      ; RESTORE NUMBER TRANSFERRED
        CALL    SETUP_END               ; VARIOUS CLEANUPS
        RETn

;--------------------------------------------------------------------------
; SETUP_STATE:  INITIALIZES START AND END RATES.
;--------------------------------------------------------------------------
SETUP_STATE:
        TEST    byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
        JNZ     short J1C               ; NO STATES IF DETERMINED
        MOV     AX,(RATE_500*256)+RATE_300  ; AH = START RATE, AL = END RATE
        TEST    byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
        JZ      short AX_SET            ; DO NOT KNOW DRIVE
        TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
        JNZ     short AX_SET            ; JUMP IF YES
        MOV     AX,RATE_250*257         ; START A END RATE 250 FOR 360 DRIVE
AX_SET:
        AND     byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
        OR      [DSK_STATE+eDI], AH     ; RATE FIRST TO TRY
        AND     byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
        ROR     AL,4                    ; TO OPERATION LAST RATE LOCATION
        OR      [LASTRATE], AL          ; LAST RATE
J1C:
        RETn

;--------------------------------------------------------------------------
;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
;--------------------------------------------------------------------------
FMT_INIT:
        TEST    byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
        JNZ     short F1_OUT            ; IF SO RETURN
        CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN AL
        ;; 20/02/2015
        ;;JC    short CL_DRV            ; ERROR IN CMOS ASSUME NO DRIVE
        jz      short CL_DRV ;; 20/02/2015
        DEC     AL                      ; MAKE ZERO ORIGIN
        ;;JS    short CL_DRV            ; NO DRIVE IF AL 0
        MOV     AH, [DSK_STATE+eDI]     ; AH = CURRENT STATE
        AND     AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
        OR      AL,AL                   ; CHECK FOR 360
        JNZ     short N_360             ; IF 360 WILL BE 0
        OR      AH,MED_DET+RATE_250     ; ESTABLISH MEDIA
        JMP     SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING
N_360:
        DEC     AL                      ; 1.2 M DRIVE
        JNZ     short N_12              ; JUMP IF NOT
F1_RATE:
        OR      AH,MED_DET+RATE_500     ; SET FORMAT RATE
        JMP     SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING
N_12:
        DEC     AL                      ; CHECK FOR TYPE 3
        JNZ     short N_720             ; JUMP IF NOT
        TEST    AH,DRV_DET              ; IS DRIVE DETERMINED
        JZ      short ISNT_12           ; TREAT AS NON 1.2 DRIVE
        TEST    AH,FMT_CAPA             ; IS 1.2M
        JZ      short ISNT_12           ; JUMP IF NOT
        OR      AH,MED_DET+RATE_300     ; RATE 300
        JMP     SHORT SKP_STATE         ; CONTINUE
N_720:
```

```
        DEC     AL                      ; CHECK FOR TYPE 4
        JNZ     short CL_DRV            ; NO DRIVE, CMOS BAD
        JMP     SHORT F1_RATE
ISNT_12:
        OR      AH,MED_DET+RATE_250     ; MUST BE RATE 250

SKP_STATE:
        MOV     [DSK_STATE+eDI], AH     ; STORE AWAY
F1_OUT:
        RETn
CL_DRV:
        XOR     AH,AH                   ; CLEAR STATE
        JMP     SHORT SKP_STATE         ; SAVE IT

;--------------------------------------------------------------------------------
; MED_CHANGE
;       CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
;       CHECKS MEDIA CHANGE AGAIN.
;
; ON EXIT:      CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
;               @DSKETTE_STATUS = ERROR CODE
;--------------------------------------------------------------------------------
MED_CHANGE:
        CALL    READ_DSKCHNG            ; READ DISK CHANCE LINE STATE
        JZ      short MC_OUT            ; BYPASS HANDLING DISK CHANGE LINE
        AND     byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE

;       THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
;       ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
;       BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).

        MOV     CX,DI                   ; CL = DRIVE 0
        MOV     AL,1                    ; MOTOR ON BIT MASK
        SHL     AL,CL                   ; TO APPROPRIATE POSITION
        NOT     AL                      ; KEEP ALL BUT MOTOR ON
        CLI                             ; NO INTERRUPTS
        AND     [MOTOR_STATUS], AL      ; TURN MOTOR OFF INDICATOR
        STI                             ; INTERRUPTS ENABLED
        CALL    MOTOR_ON                ; TURN MOTOR ON

;-----  THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL

        CALL    DSK_RESET               ; RESET NEC
        MOV     CH,01H                  ; MOVE TO CYLINDER 1
        CALL    SEEK                    ; ISSUE SEEK
        XOR     CH,CH                   ; MOVE TO CYLINDER 0
        CALL    SEEK                    ; ISSUE SEEK
        MOV     byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
OK1:
        CALL    READ_DSKCHNG            ; CHECK MEDIA CHANGED AGAIN
        JZ      short OK2               ; IF ACTIVE, NO DISKETTE, TIMEOUT
OK4:
        MOV     byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
OK2:
        STC                             ; MEDIA CHANGED, SET CY
        RETn
MC_OUT:
        CLC                             ; NO MEDIA CHANGED, CLEAR CY
        RETn

;--------------------------------------------------------------------------------
; SEND_RATE
;       SENDS DATA RATE COMMAND TO NEC
; ON ENTRY:     DI = DRIVE #
```

```
; ON EXIT:      NONE
; REGISTERS ALTERED: DX
;--------------------------------------------------------------------------
SEND_RATE:
        PUSH    AX                      ; SAVE REG.
        AND     byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
        MOV     AL, [DSK_STATE+eDI]     ; GET RATE STATE OF THIS DRIVE
        AND     AL,SEND_MSK             ; KEEP ONLY RATE BITS
        OR      [LASTRATE], AL          ; SAVE NEW RATE FOR NEXT CHECK
        ROL     AL,2                    ; MOVE TO BIT OUTPUT POSITIONS
        MOV     DX,03F7H                ; OUTPUT NEW DATA RATE
        OUT     DX,AL
        POP     AX                      ; RESTORE REG.
        RETn

;--------------------------------------------------------------------------
; CHK_LASTRATE
;       CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.
; ON ENTRY:
;       DI = DRIVE #
; ON EXIT:
;       ZF =  1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC
;       ZF =  0 DATA RATE IS DIFFERENT FROM LAST RATE
; REGISTERS ALTERED: DX
;--------------------------------------------------------------------------
CHK_LASTRATE:
        PUSH    AX                      ; SAVE REG
        AND     AH, [LASTRATE]          ; GET LAST DATA RATE SELECTED
        MOV     AL, [DSK_STATE+eDI]     ; GET RATE STATE OF THIS DRIVE
        AND     AX, SEND_MSK*257        ; KEEP ONLY RATE BITS OF BOTH
        CMP     AL, AH                  ; COMPARE TO PREVIOUSLY TRIED
                                        ; ZF = 1 RATE IS THE SAME
        POP     AX                      ; RESTORE REG.
        RETn

;--------------------------------------------------------------------------
; DMA_SETUP
;       THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
;
; ON ENTRY:     AL = DMA COMMAND
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------

; SI = Head #, # of Sectors or DASD Type

; 08/02/2015 - Protected Mode Modification
; 06/02/2015 - 07/02/2015
; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
; (DMA Addres = Physical Address)
; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)
;

; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
; 16/12/2014 (IODELAY)

DMA_SETUP:
;; 20/02/2015
        mov     edx, [ebp+4]            ; Buffer address
        test    edx, 0FFF00000h         ; 16 MB limit
        jnz     short dma_bnd_err_stc
        ;
        push    ax                      ; DMA command
        push    dx                      ; *
```

```
        mov     dl, 3                   ; GET BYTES/SECTOR PARAMETER
        call    GET_PARM                ;
        mov     cl, ah                  ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
        mov     ax, si                  ; Sector count
        mov     ah, al                  ; AH =  # OF SECTORS
        sub     al, al                  ; AL = 0, AX = # SECTORS * 256
        shr     ax, 1                   ; AX = # SECTORS * 128
        shl     ax, cl                  ; SHIFT BY PARAMETER VALUE
        dec     ax                      ; -1 FOR DMA VALUE
        mov     cx, ax
        pop     dx                      ; *
        pop     ax
        cmp     al, 42h
        jne     short NOT_VERF
        mov     edx, 0FF0000h
        jmp     short J33
NOT_VERF:
        add     dx, cx                  ; check for overflow
        jc      short dma_bnd_err
        ;
        sub     dx, cx                  ; Restore start address
J33:
        CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
        OUT     DMA+12,AL               ; SET THE FIRST/LA5T F/F
        IODELAY                         ; WAIT FOR I/O
        OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
        mov     eax, edx                ; Buffer address
        OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
        IODELAY                         ; WAIT FOR I/O
        MOV     AL,AH
        OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
        shr     eax, 16
        IODELAY                         ; I/O WAIT STATE
        OUT     081H,AL                 ; OUTPUT highest BITS TO PAGE REGISTER
        IODELAY
        mov     ax, cx                  ; Byte count - 1
        OUT     DMA+5,AL                ; LOW BYTE OF COUNT
        IODELAY                         ; WAIT FOR I/O
        MOV     AL, AH
        OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
        IODELAY
        STI                             ; RE-ENABLE INTERRUPTS
        MOV     AL, 2                   ; MODE FOR 8237
        OUT     DMA+10, AL              ; INITIALIZE THE DISKETTE CHANNEL
        retn
dma_bnd_err_stc:
        stc
dma_bnd_err:
        MOV     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
        RETn                            ; CY SET BY ABOVE IF ERROR

;; 16/12/2014
;;      CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
;;      OUT     DMA+12,AL               ; SET THE FIRST/LA5T F/F
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      IODELAY
;;      OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
;;      ;SIODELAY
;;      ;CMP    AL, 42H                 ; DMA VERIFY COMMAND
;;      ;JNE    short NOT_VERF          ; NO
;;      ;XOR    AX, AX                  ; START ADDRESS
;;      ;JMP    SHORT J33
;;;NOT_VERF:
;;      ;MOV    AX,ES                   ; GET THE ES VALUE
```

```
;;      ;ROL    AX,4                    ; ROTATE LEFT
;;      ;MOV    CH,AL                   ; GET HIGHEST NIBBLE OF ES TO CH
;;      ;AND    AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
;;      ;ADD    AX,[BP+2]               ; TEST FOR CARRY FROM ADDITION
;;      mov     eax, [ebp+4] ; 06/02/2015
;;      ;JNC    short J33
;;      ;INC    CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
;;;J33:
;;      PUSH    eAX                     ; SAVE START ADDRESS
;;      OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      IODELAY
;;      MOV     AL,AH
;;      OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
;;      shr     eax, 16      ; 07/02/2015
;;      ;MOV    AL,CH                   ; GET HIGH 4 BITS
;;      ;JMP    $+2                     ; I/O WAIT STATE
;;      IODELAY
;;      ;AND    AL,00001111B
;;      OUT     081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
;;      ;SIODELAY
;;
;;;----- DETERMINE COUNT
;;      sub     eax, eax ; 08/02/2015
;;      MOV     AX, SI                  ; AL =  # OF SECTORS
;;      XCHG    AL, AH                  ; AH =  # OF SECTORS
;;      SUB     AL, AL                  ; AL = 0, AX = # SECTORS * 256
;;      SHR     AX, 1                   ; AX = # SECTORS * 128
;;      PUSH    AX                      ; SAVE # OF SECTORS * 128
;;      MOV     DL, 3                   ; GET BYTES/SECTOR PARAMETER
;;      CALL    GET_PARM                ; "
;;      MOV     CL,AH                   ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
;;      POP     AX                      ; AX = # SECTORS * 128
;;      SHL     AX,CL                   ; SHIFT BY PARAMETER VALUE
;;      DEC     AX                      ; -1 FOR DMA VALUE
;;      PUSH    eAX  ; 08/02/2015       ; SAVE COUNT VALUE
;;      OUT     DMA+5,AL                ; LOW BYTE OF COUNT
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      IODELAY
;;      MOV     AL, AH
;;      OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
;;      ;IODELAY
;;      STI                             ; RE-ENABLE INTERRUPTS
;;      POP     eCX  ; 08/02/2015       ; RECOVER COUNT VALUE
;;      POP     eAX  ; 08/02/2015       ; RECOVER ADDRESS VALUE
;;      ;ADD    AX, CX                  ; ADD, TEST FOR 64K OVERFLOW
;;      add     ecx, eax ; 08/02/2015
;;      MOV     AL, 2                   ; MODE FOR 8237
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      SIODELAY
;;      OUT     DMA+10, AL              ; INITIALIZE THE DISKETTE CHANNEL
;;      ;JNC    short NO_BAD            ; CHECK FOR ERROR
;;      jc      short dma_bnd_err ; 08/02/2015
;;      and     ecx, 0FFF00000h ; 16 MB limit
;;      jz      short NO_BAD
;;dma_bnd_err:
;;      MOV     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
;;NO_BAD:
;;      RETn                            ; CY SET BY ABOVE IF ERROR

;--------------------------------------------------------------------------
; FMTDMA_SET
;       THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
;
```

```
; ON ENTRY:     NOTHING REQUIRED
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;-----------------------------------------------------------------------------

FMTDMA_SET:
;; 20/02/2015 modification
        mov     edx, [ebp+4]            ; Buffer address
        test    edx, 0FFF00000h         ; 16 MB limit
        jnz     short dma_bnd_err_stc
        ;
        push    dx                      ; *
        mov     DL, 4                   ; SECTORS/TRACK VALUE IN PARM TABLE
        call    GET_PARM                ; "
        mov     al, ah                  ; AL = SECTORS/TRACK VALUE
        sub     ah, ah                  ; AX = SECTORS/TRACK VALUE
        shl     ax, 2                   ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
        dec     ax                      ; -1 FOR DMA VALUE
        mov     cx, ax
        pop     dx                      ; *
        add     dx, cx                  ; check for overflow
        jc      short dma_bnd_err
        ;
        sub     dx, cx                  ; Restore start address
        ;
        MOV     AL, 04AH                ; WILL WRITE TO THE DISKETTE
        CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
        OUT     DMA+12,AL               ; SET THE FIRST/LA5T F/F
        IODELAY                         ; WAIT FOR I/O
        OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
        mov     eax, edx                ; Buffer address
        OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
        IODELAY                         ; WAIT FOR I/O
        MOV     AL,AH
        OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
        shr     eax, 16
        IODELAY                         ; I/O WAIT STATE
        OUT     081H,AL                 ; OUTPUT highest BITS TO PAGE REGISTER
        IODELAY
        mov     ax, cx                  ; Byte count - 1
        OUT     DMA+5,AL                ; LOW BYTE OF COUNT
        IODELAY                         ; WAIT FOR I/O
        MOV     AL, AH
        OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
        IODELAY
        STI                             ; RE-ENABLE INTERRUPTS
        MOV     AL, 2                   ; MODE FOR 8237
        OUT     DMA+10, AL              ; INITIALIZE THE DISKETTE CHANNEL
        retn

;; 08/02/2015 - Protected Mode Modification
;;      MOV     AL, 04AH                ; WILL WRITE TO THE DISKETTE
;;      CLI                             ; DISABLE INTERRUPTS DURING DMA SET-UP
;;      OUT     DMA+12,AL               ; SET THE FIRST/LA5T F/F
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      IODELAY
;;      OUT     DMA+11,AL               ; OUTPUT THE MODE BYTE
;;      ;MOV    AX,ES                   ; GET THE ES VALUE
;;      ;ROL    AX,4                    ; ROTATE LEFT
;;      ;MOV    CH,AL                   ; GET HIGHEST NIBBLE OF ES TO CH
;;      ;AND    AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
;;      ;ADD    AX,[BP+2]               ; TEST FOR CARRY FROM ADDITION
;;      ;JNC    short J33A
;;      ;INC    CH                      ; CARRY MEANS HIGH 4 BITS MUST BE INC
```

```
;;      mov     eax, [ebp+4] ; 08/02/2015
;;;J33A:
;;      PUSH    eAX ; 08/02/2015        ; SAVE START ADDRESS
;;      OUT     DMA+4,AL                ; OUTPUT LOW ADDRESS
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      IODELAY
;;      MOV     AL,AH
;;      OUT     DMA+4,AL                ; OUTPUT HIGH ADDRESS
;;      shr     eax, 16 ; 08/02/2015
;;      ;MOV    AL,CH                   ; GET HIGH 4 BITS
;;      ;JMP    $+2                     ; I/O WAIT STATE
;;      IODELAY
;;      ;AND    AL,00001111B
;;      OUT     081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
;;
;;;----- DETERMINE COUNT
;;      sub     eax, eax ; 08/02/2015
;;      MOV     DL, 4                   ; SECTORS/TRACK VALUE IN PARM TABLE
;;      CALL    GET_PARM                ; "
;;      XCHG    AL, AH                  ; AL = SECTORS/TRACK VALUE
;;      SUB     AH, AH                  ; AX = SECTORS/TRACK VALUE
;;      SHL     AX, 2                   ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
;;      DEC     AX                      ; -1 FOR DMA VALUE
;;      PUSH    eAX     ; 08/02/2015    ; SAVE # OF BYTES TO BE TRANSFERED
;;      OUT     DMA+5,AL                ; LOW BYTE OF COUNT
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      IODELAY
;;      MOV     AL, AH
;;      OUT     DMA+5,AL                ; HIGH BYTE OF COUNT
;;      STI                             ; RE-ENABLE INTERRUPTS
;;      POP     eCX     ; 08/02/2015    ; RECOVER COUNT VALUE
;;      POP     eAX     ; 08/02/2015    ; RECOVER ADDRESS VALUE
;;      ;ADD    AX, CX                  ; ADD, TEST FOR 64K OVERFLOW
;;      add     ecx, eax ; 08/02/2015
;;      MOV     AL, 2                   ; MODE FOR 8237
;;      ;JMP    $+2                     ; WAIT FOR I/O
;;      SIODELAY
;;      OUT     DMA+10, AL              ; INITIALIZE THE DISKETTE CHANNEL
;;      ;JNC    short FMTDMA_OK         ; CHECK FOR ERROR
;;      jc      short fmtdma_bnd_err ; 08/02/2015
;;      and     ecx, 0FFF00000h  ; 16 MB limit
;;      jz      short FMTDMA_OK
;;      stc     ; 20/02/2015
;;fmtdma_bnd_err:
;;      MOV     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
;;FMTDMA_OK:
;;      RETn                            ; CY SET BY ABOVE IF ERROR

;--------------------------------------------------------------------------
; NEC_INIT
;       THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
;       THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
;
; ON ENTRY:     AH = NEC COMMAND TO BE PERFORMED
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------
NEC_INIT:
        PUSH    AX                      ; SAVE NEC COMMAND
        CALL    MOTOR_ON                ; TURN MOTOR ON FOR SPECIFIC DRIVE

;-----  DO THE SEEK OPERATION

        MOV     CH,[eBP+1]              ; CH = TRACK #
```

```
        CALL    SEEK                    ; MOVE TO CORRECT TRACK
        POP     AX                      ; RECOVER COMMAND
        JC      short ER_1              ; ERROR ON SEEK
        MOV     eBX, ER_1               ; LOAD ERROR ADDRESS
        PUSH    eBX                     ; PUSH NEC_OUT ERROR RETURN

;-----  SEND OUT THE PARAMETERS TO THE CONTROLLER

        CALL    NEC_OUTPUT              ; OUTPUT THE OPERATION COMMAND
        MOV     AX,SI                   ; AH = HEAD #
        MOV     eBX,eDI                 ; BL = DRIVE #
        SAL     AH,2                    ; MOVE IT TO BIT 2
        AND     AH,00000100B            ; ISOLATE THAT BIT
        OR      AH,BL                   ; OR IN THE DRIVE NUMBER
        CALL    NEC_OUTPUT              ; FALL THRU CY SET IF ERROR
        POP     eBX                     ; THROW AWAY ERROR RETURN
ER_1:
        RETn

;--------------------------------------------------------------------------------
; RWV_COM
;       THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
;       READ/WRITE/VERIFY OPERATIONS.
;
; ON ENTRY:     CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
RWV_COM:
        MOV     eAX, ER_2               ; LOAD ERROR ADDRESS
        PUSH    eAX                     ; PUSH NEC_OUT ERROR RETURN
        MOV     AH,[eBP+1]              ; OUTPUT TRACK #
        CALL    NEC_OUTPUT
        MOV     AX,SI                   ; OUTPUT HEAD #
        CALL    NEC_OUTPUT
        MOV     AH,[eBP]                ; OUTPUT SECTOR #
        CALL    NEC_OUTPUT
        MOV     DL,3                    ; BYTES/SECTOR PARAMETER FROM BLOCK
        CALL    GET_PARM                ; ... TO THE NEC
        CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
        MOV     DL,4                    ; EOT PARAMETER FROM BLOCK
        CALL    GET_PARM                ; ... TO THE NEC
        CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
        MOV     AH, [eBX+MD.GAP]        ; GET GAP LENGTH
_R15:
        CALL    NEC_OUTPUT
        MOV     DL,6                    ; DTL PARAMETER PROM BLOCK
        CALL    GET_PARM                ;  TO THE NEC
        CALL    NEC_OUTPUT              ; OUTPUT TO CONTROLLER
        POP     eAX                     ; THROW AWAY ERROR EXIT
ER_2:
        RETn

;--------------------------------------------------------------------------------
; NEC_TERM
;       THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
;       FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
NEC_TERM:

;-----  LET THE OPERATION HAPPEN

        PUSH    eSI                     ; SAVE HEAD #, # OF SECTORS
```

```
        CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
        PUSHF
        CALL    RESULTS                 ; GET THE NEC STATUS
        JC      short SET_END_POP
        POPF
        JC      short SET_END           ; LOOK FOR ERROR

;-----  CHECK THE RESULTS RETURNED BY THE CONTROLLER

        CLD                             ; SET THE CORRECT DIRECTION
        MOV     eSI, NEC_STATUS         ; POINT TO STATUS FIELD
        lodsb                           ; GET ST0
        AND     AL,11000000B            ; TEST FOR NORMAL TERMINATION
        JZ      short SET_END
        CMP     AL,01000000B            ; TEST FOR ABNORMAL TERMINATION
        JNZ     short J18               ; NOT ABNORMAL, BAD NEC

;-----  ABNORMAL TERMINATION, FIND OUT WHY

        lodsb                           ; GET ST1
        SAL     AL,1                    ; TEST FOR EDT FOUND
        MOV     AH,RECORD_NOT_FND
        JC      short J19
        SAL     AL,2
        MOV     AH,BAD_CRC
        JC      short J19
        SAL     AL,1                    ; TEST FOR DMA OVERRUN
        MOV     AH,BAD_DMA
        JC      short J19
        SAL     AL,2                    ; TEST FOR RECORD NOT FOUND
        MOV     AH,RECORD_NOT_FND
        JC      short J19
        SAL     AL,1
        MOV     AH,WRITE_PROTECT        ; TEST FOR WRITE_PROTECT
        JC      short J19
        SAL     AL,1                    ; TEST MISSING ADDRESS MARK
        MOV     AH,BAD_ADDR_MARK
        JC      short J19

;-----  NEC MUST HAVE FAILED
J18:
        MOV     AH,BAD_NEC
J19:
        OR      [DSKETTE_STATUS], AH
SET_END:
        CMP     byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
        CMC
        POP     eSI
        RETn                            ; RESTORE HEAD #, # OF SECTORS

SET_END_POP:
        POPF
        JMP     SHORT SET_END

;--------------------------------------------------------------------------
; DSTATE:       ESTABLISH STATE UPON SUCCESSFUL OPERATION.
;--------------------------------------------------------------------------
DSTATE:
        CMP     byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
        JNZ     short SETBAC                ; IF ERROR JUMP
        OR      byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS
DETERMINED
        TEST    byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
        JNZ     short SETBAC            ; IF DETERMINED NO TRY TO DETERMINE
```

```
        MOV     AL,[DSK_STATE+eDI]      ; LOAD STATE
        AND     AL,RATE_MSK             ; KEEP ONLY RATE
        CMP     AL,RATE_250             ; RATE 250 ?
        JNE     short M_12              ; NO, MUST BE 1.2M OR 1.44M DRIVE

;-----  CHECK IF IT IS 1.44M

        CALL    CMOS_TYPE               ; RETURN DRIVE TYPE IN (AL)
        ;;20/02/2015
        ;;JC    short M_12              ; CMOS BAD
        jz      short M_12 ;; 20/02/2015
        CMP     AL, 4                   ; 1.44MB DRIVE ?
        JE      short M_12              ; YES
M_720:
        AND     byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
        OR      byte [DSK_STATE+eDI],DRV_DET  ; MARK DRIVE DETERMINED
        JMP     SHORT SETBAC            ; BACK
M_12:
        OR      byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
                                        ; TURN ON DETERMINED & FMT CAPA
SETBAC:
        RETn

;--------------------------------------------------------------------------
; RETRY
;       DETERMINES WHETHER A RETRY IS NECESSARY.
;       IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
;
; ON EXIT:      CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
;--------------------------------------------------------------------------
RETRY:
        CMP     byte [DSKETTE_STATUS],0 ; GET STATUS OF OPERATION
        JZ      short NO_RETRY          ; SUCCESSFUL OPERATION
        CMP     byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
        JZ      short NO_RETRY
        MOV     AH,[DSK_STATE+eDI]      ; GET MEDIA STATE OF DRIVE
        TEST    AH,MED_DET              ; ESTABLISHED/DETERMINED ?
        JNZ     short NO_RETRY          ; IF ESTABLISHED STATE THEN TRUE ERROR
        AND     AH,RATE_MSK             ; ISOLATE RATE
        MOV     CH,[LASTRATE]           ; GET START OPERATION STATE
        ROL     CH,4                    ; TO CORRESPONDING BITS
        AND     CH,RATE_MSK             ; ISOLATE RATE BITS
        CMP     CH,AH                   ; ALL RATES TRIED
        JE      short NO_RETRY          ; IF YES, THEN TRUE ERROR

;       SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
;        00000000B (500) -> 10000000B   (250)
;        10000000B (250) -> 01000000B   (300)
;        01000000B (300) -> 00000000B   (500)

        CMP     AH,RATE_500+1           ; SET CY FOR RATE 500
        RCR     AH,1                    ; TO NEXT STATE
        AND     AH,RATE_MSK             ; KEEP ONLY RATE BITS
        AND     byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
                                        ; RATE, DBL STEP OFF
        OR      [DSK_STATE+eDI],AH      ; TURN ON NEW RATE
        MOV     byte [DSKETTE_STATUS],0 ; RESET STATUS FOR RETRY
        STC                             ; SET CARRY FOR RETRY
        RETn                            ; RETRY RETURN

NO_RETRY:
        CLC                             ; CLEAR CARRY NO RETRY
        RETn                            ; NO RETRY RETURN
```

```
;--------------------------------------------------------------------------------
; NUM_TRANS
;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
;       ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
;
; ON ENTRY:     [BP+1] = TRACK
;               SI-HI  = HEAD
;               [BP]   = START SECTOR
;
; ON EXIT:      AL = NUMBER ACTUALLY TRANSFERRED
;--------------------------------------------------------------------------------
NUM_TRANS:
        XOR     AL,AL                   ; CLEAR FOR ERROR
        CMP     byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
        JNZ     NT_OUT                  ; IF ERROR 0 TRANSFERRED
        MOV     DL,4                    ; SECTORS/TRACK OFFSET TO DL
        CALL    GET_PARM                ; AH = SECTORS/TRACK
        MOV     BL, [NEC_STATUS+5]      ; GET ENDING SECTOR
        MOV     CX,SI                   ; CH = HEAD # STARTED
        CMP     CH, [NEC_STATUS+4]      ; GET HEAD ENDED UP ON
        JNZ     DIF_HD                  ; IF ON SAME HEAD, THEN NO ADJUST
        MOV     CH, [NEC_STATUS+3]      ; GET TRACK ENDED UP ON
        CMP     CH,[eBP+1]              ; IS IT ASKED FOR TRACK
        JZ      short SAME_TRK          ; IF SAME TRACK NO INCREASE
        ADD     BL,AH                   ; ADD SECTORS/TRACK
DIF_HD:
        ADD     BL,AH                   ; ADD SECTORS/TRACK
SAME_TRK:
        SUB     BL,[eBP]                ; SUBTRACT START FROM END
        MOV     AL,BL                   ; TO AL
NT_OUT:
        RETn

;--------------------------------------------------------------------------------
; SETUP_END
;       RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
;       AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
;
; ON EXIT:
;       AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------------
SETUP_END:
        MOV     DL,2                    ; GET THE MOTOR WAIT PARAMETER
        PUSH    AX                      ; SAVE NUMBER TRANSFERRED
        CALL    GET_PARM
        MOV     [MOTOR_COUNT],AH        ; STORE UPON RETURN
        POP     AX                      ; RESTORE NUMBER TRANSFERRED
        MOV     AH, [DSKETTE_STATUS]    ; GET STATUS OF OPERATION
        OR      AH,AH                   ; CHECK FOR ERROR
        JZ      short NUN_ERR                   ; NO ERROR
        XOR     AL,AL                   ; CLEAR NUMBER RETURNED
NUN_ERR:
        CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
        CMC                             ; SUCCESS OR FAILURE
        RETn

;--------------------------------------------------------------------------------
; SETUP_DBL
;       CHECK DOUBLE STEP.
;
; ON ENTRY :    DI = DRIVE
;
; ON EXIT :     CY = 1 MEANS ERROR
;--------------------------------------------------------------------------------
```

```
SETUP_DBL:
        MOV     AH, [DSK_STATE+eDI]     ; ACCESS STATE
        TEST    AH,MED_DET              ; ESTABLISHED STATE ?
        JNZ     short NO_DBL                    ; IF ESTABLISHED THEN DOUBLE
DONE

;-----  CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE

        MOV     byte [SEEK_STATUS],0    ; SET RECALIBRATE REQUIRED ON ALL DRIVES
        CALL    MOTOR_ON                ; ENSURE MOTOR STAY ON
        MOV     CH,0                    ; LOAD TRACK 0
        CALL    SEEK                    ; SEEK TO TRACK 0
        CALL    READ_ID                 ; READ ID FUNCTION
        JC      short SD_ERR            ; IF ERROR NO TRACK 0

;-----  INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)

        MOV     CX,0450H                ; START, MAX TRACKS
        TEST    byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
        JZ      short CNT_OK            ; IF NOT COUNT IS SETUP
        MOV     CL,0A0H                 ; MAXIMUM TRACK 1.2 MB

;       ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
;       MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
;       THEN SET DOUBLE STEP ON.

CNT_OK:
        MOV     byte [MOTOR_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
        PUSH    CX                      ; SAVE TRACK, COUNT
        MOV     byte [DSKETTE_STATUS],0 ; CLEAR STATUS, EXPECT ERRORS
        XOR     AX,AX                   ; CLEAR AX
        SHR     CH,1                    ; HALVE TRACK, CY = HEAD
        RCL     AL,3                    ; AX = HEAD IN CORRECT BIT
        PUSH    AX                      ; SAVE HEAD
        CALL    SEEK                    ; SEEK TO TRACK
        POP     AX                      ; RESTORE HEAD
        OR      DI,AX                   ; DI = HEAD OR'ED DRIVE
        CALL    READ_ID                 ; READ ID HEAD 0
        PUSHF                           ; SAVE RETURN FROM READ_ID
        AND     DI,11111011B            ; TURN OFF HEAD 1 BIT
        POPF                            ; RESTORE ERROR RETURN
        POP     CX                      ; RESTORE COUNT
        JNC     short DO_CHK            ; IF OK, ASKED = RETURNED TRACK ?
        INC     CH                      ; INC FOR NEXT TRACK
        CMP     CH,CL                   ; REACHED MAXIMUM YET
        JNZ     short CNT_OK            ; CONTINUE TILL ALL TRIED

;-----  FALL THRU, READ ID FAILED FOR ALL TRACKS

SD_ERR:
        STC                             ; SET CARRY FOR ERROR
        RETn                            ; SETUP_DBL ERROR EXIT

DO_CHK:
        MOV     CL, [NEC_STATUS+3]      ; LOAD RETURNED TRACK
        MOV     [DSK_TRK+eDI], CL       ; STORE TRACK NUMBER
        SHR     CH,1                    ; HALVE TRACK
        CMP     CH,CL                   ; IS IT THE SAME AS ASKED FOR TRACK
        JZ      short NO_DBL            ; IF SAME THEN NO DOUBLE STEP
        OR      byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
NO_DBL:
        CLC                             ; CLEAR ERROR FLAG
        RETn
```

```
;--------------------------------------------------------------------------
; READ_ID
;       READ ID FUNCTION.
;
; ON ENTRY:     DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
;
; ON EXIT:      DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
;               @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
;--------------------------------------------------------------------------
READ_ID:
        MOV     eAX, ER_3               ; MOVE NEC OUTPUT ERROR ADDRESS
        PUSH    eAX
        MOV     AH,4AH                  ; READ ID COMMAND
        CALL    NEC_OUTPUT              ; TO CONTROLLER
        MOV     AX,DI                   ; DRIVE # TO AH, HEAD 0
        MOV     AH,AL
        CALL    NEC_OUTPUT              ; TO CONTROLLER
        CALL    NEC_TERM                ; WAIT FOR OPERATION, GET STATUS
        POP     eAX                     ; THROW AWAY ERROR ADDRESS
ER_3:
        RETn

;--------------------------------------------------------------------------
; CMOS_TYPE
;       RETURNS DISKETTE TYPE FROM CMOS
;
; ON ENTRY:     DI = DRIVE #
;
; ON EXIT:      AL = TYPE; CY REFLECTS STATUS
;--------------------------------------------------------------------------

CMOS_TYPE: ; 11/12/2014
mov     al, [eDI+fd0_type]
and     al, al ; 18/12/2014
retn

;CMOS_TYPE:
;       MOV     AL, CMOS_DIAG           ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
;       CALL    CMOS_READ               ; GET CMOS STATUS
;       TEST    AL,BAD_BAT+BAD_CKSUM    ; BATTERY GOOD AND CHECKSUM VALID
;       STC                             ; SET CY = 1 INDICATING ERROR FOR RETURN
;       JNZ     short BAD_CM            ; ERROR IF EITHER BIT ON
;       MOV     AL,CMOS_DISKETTE        ; ADDRESS OF DISKETTE BYTE IN CMOS
;       CALL    CMOS_READ               ; GET DISKETTE BYTE
;       OR      DI,DI                   ; SEE WHICH DRIVE IN QUESTION
;       JNZ     short TB                ; IF DRIVE 1, DATA IN LOW NIBBLE
;       ROR     AL,4                    ; EXCHANGE NIBBLES IF SECOND DRIVE
;TB:
;       AND     AL,0FH                  ; KEEP ONLY DRIVE DATA, RESET CY, 0
;BAD_CM:
;       RETn                            ; CY, STATUS OF READ

;--------------------------------------------------------------------------
; GET_PARM
;       THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
;       BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
;       THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
;       THE PARAMETER IN DL.
;
; ON ENTRY:     DL = INDEX OF BYTE TO BE FETCHED
;
; ON EXIT:      AH = THAT BYTE FROM BLOCK
;               AL,DH DESTROYED
;--------------------------------------------------------------------------
```

```
GET_PARM:
        ;PUSH   DS
        PUSH    eSI
        ;SUB    AX,AX                   ; DS = 0, BIOS DATA AREA
        ;MOV    DS,AX
        ;;mov   ax, cs
        ;;mov   ds, ax
        ; 08/02/2015 (protected mode modifications, bx -> ebx)
        XCHG    eDX,eBX                 ; BL = INDEX
        ;SUB    BH,BH                   ; BX = INDEX
        and     ebx, 0FFh
        ;LDS    SI, [DISK_POINTER]      ; POINT TO BLOCK
        ;
        ; 17/12/2014
        mov     ax, [cfd] ; current (AL) and previous fd (AH)
        cmp     al, ah
        je      short gpndc
        mov     [pfd], al ; current drive -> previous drive
        push    ebx ; 08/02/2015
        mov     bl, al
        ; 11/12/2014
        mov     al, [eBX+fd0_type]      ; Drive type (0,1,2,3,4)
        ; 18/12/2014
        and     al, al
        jnz     short gpdtc
        mov     ebx, MD_TBL6            ; 1.44 MB param. tbl. (default)
        jmp     short gpdpu
gpdtc:
        call    DR_TYPE_CHECK
        ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
gpdpu:
        mov     [DISK_POINTER], ebx
        pop     ebx
gpndc:
        mov     esi, [DISK_POINTER] ; 08/02/2015, si -> esi
        MOV     AH, [eSI+eBX]           ; GET THE WORD
        XCHG    eDX,eBX                 ; RESTORE BX
        POP     eSI
        ;POP    DS
        RETn

;--------------------------------------------------------------------------
; MOTOR_ON
;       TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
;       IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
;       THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
;       MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
;       (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
;       THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
;       FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
;       HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
;       THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
;       NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
;       PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
;       IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
;       WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
;
; ON ENTRY:     DI = DRIVE #
; ON EXIT:      AX,CX,DX DESTROYED
;--------------------------------------------------------------------------
MOTOR_ON:
        PUSH    eBX                     ; SAVE REG.
        CALL    TURN_ON                 ; TURN ON MOTOR
        JC      short MOT_IS_ON         ; IF CY=1 NO WAIT
```

```
        CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
        CALL    XLAT_NEW                ; TRANSLATE STATE TO PRESENT ARCH,
        ;CALL   TURN_ON                 ; CHECK AGAIN IF MOTOR ON
        ;JC     MOT_IS_ON               ; IF NO WAIT MEANS IT IS ON
M_WAIT:
        MOV     DL,10                   ; GET THE MOTOR WAIT PARAMETER
        CALL    GET_PARM
        ;MOV    AL,AH                   ; AL = MOTOR WAIT PARAMETER
        ;XOR    AH,AH                   ; AX = MOTOR WAIT PARAMETER
        ;CMP    AL,8                    ; SEE IF AT LEAST A SECOND IS SPECIFIED
        cmp     ah, 8
        ;JAE    short GP2               ; IF YES, CONTINUE
        ja      short J13
        ;MOV    AL,8                    ; ONE SECOND WAIT FOR MOTOR START UP
        mov     ah, 8

;-----  AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
GP2:
;-----  FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
J13:                                    ; WAIT FOR 1/8 SECOND PER (AL)
        MOV     eCX,8286                ; COUNT FOR 1/8 SECOND AT 15.085737 US
        CALL    WAITF                   ; GO TO FIXED WAIT ROUTINE
        ;DEC    AL                      ; DECREMENT TIME VALUE
        dec     ah
        JNZ     short J13               ; ARE WE DONE YET
MOT_IS_ON:
        POP     eBX                     ; RESTORE REG.
        RETn

;--------------------------------------------------------------------------------
; TURN_ON
;       TURN MOTOR ON AND RETURN WAIT STATE.
;
; ON ENTRY:     DI = DRIVE #
;
; ON EXIT:      CY = 0 MEANS WAIT REQUIRED
;               CY = 1 MEANS NO WAIT REQUIRED
;               AX,BX,CX,DX DESTROYED
;--------------------------------------------------------------------------------
TURN_ON:
        MOV     eBX,eDI                 ; BX = DRIVE #
        MOV     CL,BL                   ; CL = DRIVE #
        ROL     BL,4                    ; BL = DRIVE SELECT
        CLI                             ; NO INTERRUPTS WHILE DETERMINING STATUS
        MOV     byte [MOTOR_COUNT],0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
        MOV     AL, [MOTOR_STATUS]      ; GET DIGITAL OUTPUT REGISTER REFLECTION
        AND     AL,00110000B            ; KEEP ONLY DRIVE SELECT BITS
        MOV     AH,1                    ; MASK FOR DETERMINING MOTOR BIT
        SHL     AH,CL                   ; AH = MOTOR ON, A=00000001, B=00000010

;  AL = DRIVE SELECT FROM @MOTOR_STATUS
;  BL = DRIVE SELECT DESIRED
;  AH = MOTOR ON MASK DESIRED

        CMP     AL,BL                   ; REQUESTED DRIVE ALREADY SELECTED ?
        JNZ     short TURN_IT_ON        ; IF NOT SELECTED JUMP
        TEST    AH, [MOTOR_STATUS]      ; TEST MOTOR ON BIT
        JNZ     short NO_MOT_WAIT       ; JUMP IF MOTOR ON AND SELECTED

TURN_IT_ON:
        OR      AH,BL                   ; AH = DRIVE SELECT AND MOTOR ON
        MOV     BH, [MOTOR_STATUS]      ; SAVE COPY OF @MOTOR_STATUS BEFORE
        AND     BH,00001111B            ; KEEP ONLY MOTOR BITS
        AND     byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
```

```
        OR      byte [MOTOR_STATUS],AH  ; OR IN DRIVE SELECTED AND MOTOR ON
        MOV     AL, [MOTOR_STATUS]      ; GET DIGITAL OUTPUT REGISTER REFLECTION
        MOV     BL,AL                   ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
        AND     BL,00001111B            ; KEEP ONLY MOTOR BITS
        STI                             ; ENABLE INTERRUPTS AGAIN
        AND     AL,00111111B            ; STRIP AWAY UNWANTED BITS
        ROL     AL,4                    ; PUT BITS IN DESIRED POSITIONS
        OR      AL,00001100B            ; NO RESET, ENABLE DMA/INTERRUPT
        MOV     DX,03F2H                ; SELECT DRIVE AND TURN ON MOTOR
        OUT     DX,AL
        CMP     BL,BH                   ; NEW MOTOR TURNED ON ?
        JZ      short NO_MOT_WAIT       ; NO WAIT REQUIRED IF JUST SELECT
        CLC                             ; (re)SET CARRY MEANING WAIT
        RETn

NO_MOT_WAIT:
        STC                             ; SET NO WAIT REQUIRED
        STI                             ; INTERRUPTS BACK ON
        RETn

;--------------------------------------------------------------------------------
; HD_WAIT
;       WAIT FOR HEAD SETTLE TIME.
;
; ON ENTRY:     DI = DRIVE #
;
; ON EXIT:      AX,BX,CX,DX DESTROYED
;--------------------------------------------------------------------------------
HD_WAIT:
        MOV     DL,9                    ; GET HEAD SETTLE PARAMETER
        CALL    GET_PARM
        or      ah, ah  ; 17/12/2014
        jnz     short DO_WAT
        TEST    byte [MOTOR_STATUS],10000000B ; SEE IF A WRITE OPERATION
        ;JZ     short ISNT_WRITE        ; IF NOT, DO NOT ENFORCE ANY VALUES
        ;OR     AH,AH                   ; CHECK FOR ANY WAIT?
        ;JNZ    short DO_WAT            ; IF THERE DO NOT ENFORCE
        jz      short HW_DONE
        MOV     AH,HD12_SETTLE          ; LOAD 1.2M HEAD SETTLE MINIMUM
        MOV     AL,[DSK_STATE+eDI]      ; LOAD STATE
        AND     AL,RATE_MSK             ; KEEP ONLY RATE
        CMP     AL,RATE_250             ; 1.2 M DRIVE ?
        JNZ     short DO_WAT            ; DEFAULT HEAD SETTLE LOADED
;GP3:
        MOV     AH,HD320_SETTLE         ; USE 320/360 HEAD SETTLE
;       JMP     SHORT DO_WAT

;ISNT_WRITE:
;       OR      AH,AH                   ; CHECK FOR NO WAIT
;       JZ      short HW_DONE           ; IF NOT WRITE AND 0 ITS OK

;-----  AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
DO_WAT:
;       MOV     AL,AH                   ; AL = # MILLISECONDS
;       ;XOR    AH,AH                   ; AX = # MILLISECONDS
J29:                                    ;       1 MILLISECOND LOOP
        ;mov    cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
        MOV     eCX,66                  ; COUNT AT 15.085737 US PER COUNT
        CALL    WAITF                   ; DELAY FOR 1 MILLISECOND
        ;DEC    AL                      ; DECREMENT THE COUNT
        dec     ah
        JNZ     short J29               ; DO AL MILLISECOND # OF TIMES
HW_DONE:
        RETn
```

```
;--------------------------------------------------------------------------------
; NEC_OUTPUT
;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
;       FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
;       TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
;       OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
;
; ON ENTRY:     AH = BYTE TO BE OUTPUT
;
; ON EXIT:      CY = 0  SUCCESS
;               CY = 1  FAILURE -- DISKETTE STATUS UPDATED
;                       IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
;                       HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE
;                       REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
;               AX,CX,DX DESTROYED
;--------------------------------------------------------------------------------

; 09/12/2014 [Erdogan Tan]
;       (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
; Diskette Drive Controller Status Register (3F4h)
;       This read only register facilitates the transfer of data between
;       the system microprocessor and the controller.
; Bit 7 - When set to 1, the Data register is ready to transfer data
;         with the system micrprocessor.
; Bit 6 - The direction of data transfer. If this bit is set to 0,
;         the transfer is to the controller.
; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
; Bit 3 - Reserved.
; Bit 2 - Reserved.
; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.
; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.

; Data Register (3F5h)
; This read/write register passes data, commands and parameters, and provides
; diskette status information.

NEC_OUTPUT:
        ;PUSH   BX                      ; SAVE REG.
        MOV     DX,03F4H                ; STATUS PORT
        ;MOV    BL,2                    ; HIGH ORDER COUNTER
        ;XOR    CX,CX                   ; COUNT FOR TIME OUT
        ; 16/12/2014
        ; waiting for (max.) 0.5 seconds
        mov     byte [wait_count], 0
        ;
        ; 17/12/2014
        ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
        ;
        ;WAIT_FOR_PORT: Waits for a bit at a port pointed to by DX to
        ;               go on.
        ;INPUT:
        ;       AH=Mask for isolation bits.
        ;       AL=pattern to look for.
        ;       DX=Port to test for
        ;       BH:CX=Number of memory refresh periods to delay.
        ;            (normally 30 microseconds per period.)
        ;
        ;WFP_SHORT:
        ;       Wait for port if refresh cycle is short (15-80 Us range).
        ;

;       mov     bl, WAIT_FDU_SEND_HI+1  ; 0+1
```

```
;       mov     cx, WAIT_FDU_SEND_LO    ; 16667
;
;WFPS_OUTER_LP:
;       ;
;WFPS_CHECK_PORT:
;;J23:
;       IN      AL,DX                   ; GET STATUS
;       AND     AL,11000000B            ; KEEP STATUS AND DIRECTION
;       CMP     AL,10000000B            ; STATUS 1 AND DIRECTION 0 ?
;       JZ      short J27               ; STATUS AND DIRECTION OK
;WFPS_HI:
;       IN      AL, PORT_B      ;061h   ; SYS1  ; wait for hi to lo
;       TEST    AL,010H                 ; transition on memory
;       JNZ     SHORT WFPS_HI           ; refresh.
;WFPS_LO:
;       IN      AL, PORT_B              ; SYS1
;       TEST    AL,010H
;       JZ      SHORT WFPS_LO
;       LOOP    SHORT WFPS_CHECK_PORT
;       ;
;       dec     bl
;       jnz     short WFPS_OUTER_LP
;       jmp     short WFPS_TIMEOUT      ; fail

J23:
        IN      AL,DX                   ; GET STATUS
        AND     AL,11000000B            ; KEEP STATUS AND DIRECTION
        CMP     AL,10000000B            ; STATUS 1 AND DIRECTION 0 ?
        JZ      short J27               ; STATUS AND DIRECTION OK
        ;LOOP   J23                     ; CONTINUE TILL CX EXHAUSTED
        ;DEC    BL                      ; DECREMENT COUNTER
        ;JNZ    short J23               ; REPEAT TILL DELAY FINISHED, CX = 0

        ;16/12/2014
        cmp     byte [wait_count], 10   ; (10/18.2 seconds)
        jb      short J23

;WFPS_TIMEOUT:

;-----  FALL THRU TO ERROR RETURN

        OR      byte [DSKETTE_STATUS],TIME_OUT
        ;POP    BX                      ; RESTORE REG.
        POP     eAX ; 08/02/2015        ; DISCARD THE RETURN ADDRESS
        STC                             ; INDICATE ERROR TO CALLER
        RETn

;-----  DIRECTION AND STATUS OK; OUTPUT BYTE

J27:
        MOV     AL,AH                   ; GET BYTE TO OUTPUT
        INC     DX                      ; DATA PORT = STATUS PORT + 1
        OUT     DX,AL                   ; OUTPUT THE BYTE
        IODELAY
        ;
        ;PUSHF                          ; SAVE FLAGS
        ;MOV    CX, 3                   ; 30 TO 45 MICROSECONDS WAIT FOR
        ;CALL   WAITF                   ; NEC FLAGS UPDATE CYCLE
        ;POPF                           ; RESTORE FLAGS FOR EXIT
        ;POP    BX                      ; RESTORE REG
        RETn                            ; CY = 0 FROM TEST INSTRUCTION

;--------------------------------------------------------------------------------
; SEEK
```

```
;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
;       TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
;       RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
;
; ON ENTRY:     DI = DRIVE #
;               CH = TRACK #
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
;               AX,BX,CX DX DESTROYED
;--------------------------------------------------------------------------------
SEEK:
        MOV     eBX,eDI                 ; BX = DRIVE #
        MOV     AL,1                    ; ESTABLISH MASK FOR RECALIBRATE TEST
        XCHG    CL,BL                   ; SET DRIVE VALULE INTO CL
        ROL     AL,CL                   ; SHIFT MASK BY THE DRIVE VALUE
        XCHG    CL,BL                   ; RECOVER TRACK VALUE
        TEST    AL,[SEEK_STATUS]        ; TEST FOR RECALIBRATE REQUIRED
        JNZ     short J28A              ; JUMP IF RECALIBRATE NOT REQUIRED

        OR      [SEEK_STATUS],AL        ; TURN ON THE NO RECALIBRATE BIT IN FLAG
        CALL    RECAL                   ; RECALIBRATE DRIVE
        JNC     short AFT_RECAL         ; RECALIBRATE DONE

;-----  ISSUE RECALIBRATE FOR 80 TRACK DISKETTES

        MOV     byte [DSKETTE_STATUS],0 ; CLEAR OUT INVALID STATUS
        CALL    RECAL                   ; RECALIBRATE DRIVE
        JC      short RB                ; IF RECALIBRATE FAILS TWICE THEN ERROR

AFT_RECAL:
        MOV     byte [DSK_TRK+eDI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
        OR      CH,CH                   ; CHECK FOR SEEK TO TRACK 0
        JZ      short DO_WAIT           ; HEAD SETTLE, CY = 0 IF JUMP

;-----  DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK

J28A:   TEST    byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
        JZ      short R7                ; SINGLE STEP REQUIRED BYPASS DOUBLE
        SHL     CH,1                    ; DOUBLE NUMBER OF STEP TO TAKE

R7:     CMP     CH, [DSK_TRK+eDI]       ; SEE IF ALREADY AT THE DESIRED TRACK
        JE      short RB                ; IF YES, DO NOT NEED TO SEEK

        MOV     eDX, NEC_ERR            ; LOAD RETURN ADDRESS
        PUSH    eDX ; (*)               ; ON STACK FOR NEC OUTPUT ERROR
        MOV     [DSK_TRK+eDI],CH        ; SAVE NEW CYLINDER AS PRESENT POSITION
        MOV     AH,0FH                  ; SEEK COMMAND TO NEC
        CALL    NEC_OUTPUT
        MOV     eBX,eDI                 ; BX = DRIVE #
        MOV     AH,BL                   ; OUTPUT DRIVE NUMBER
        CALL    NEC_OUTPUT
        MOV     AH, [DSK_TRK+eDI]       ; GET CYLINDER NUMBER
        CALL    NEC_OUTPUT
        CALL    CHK_STAT_2              ; ENDING INTERRUPT AND SENSE STATUS

;-----  WAIT FOR HEAD SETTLE

DO_WAIT:
        PUSHF                           ; SAVE STATUS
        CALL    HD_WAIT                 ; WAIT FOR HEAD SETTLE TIME
        POPF                            ; RESTORE STATUS
RB:
NEC_ERR:
        ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
```

```
        ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
        RETn                            ; RETURN TO CALLER

;--------------------------------------------------------------------------------
; RECAL
;       RECALIBRATE DRIVE
;
; ON ENTRY:     DI = DRIVE #
;
; ON EXIT:      CY REFLECTS STATUS OF OPERATION.
;--------------------------------------------------------------------------------
RECAL:
        PUSH    CX
        MOV     eAX, RC_BACK            ; LOAD NEC_OUTPUT ERROR
        PUSH    eAX
        MOV     AH,07H                  ; RECALIBRATE COMMAND
        CALL    NEC_OUTPUT
        MOV     eBX,eDI                 ; BX = DRIVE #
        MOV     AH,BL
        CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
        CALL    CHK_STAT_2              ; GET THE INTERRUPT AND SENSE INT STATUS
        POP     eAX                     ; THROW AWAY ERROR
RC_BACK:
        POP     CX
        RETn

;--------------------------------------------------------------------------------
; CHK_STAT_2
;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
;       OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
;       INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
;--------------------------------------------------------------------------------
CHK_STAT_2:
        MOV     eAX, CS_BACK            ; LOAD NEC_OUTPUT ERROR ADDRESS
        PUSH    eAX
        CALL    WAIT_INT                ; WAIT FOR THE INTERRUPT
        JC      short J34               ; IF ERROR, RETURN IT
        MOV     AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
        CALL    NEC_OUTPUT
        CALL    RESULTS                 ; READ IN THE RESULTS
        JC      short J34
        MOV     AL,[NEC_STATUS]         ; GET THE FIRST STATUS BYTE
        AND     AL,01100000B            ; ISOLATE THE BITS
        CMP     AL,01100000B            ; TEST FOR CORRECT VALUE
        JZ      short J35               ; IF ERROR, GO MARK IT
        CLC                             ; GOOD RETURN
J34:
        POP     eAX                     ; THROW AWAY ERROR RETURN
CS_BACK:
        RETn
J35:
        OR      byte [DSKETTE_STATUS], BAD_SEEK
        STC                             ; ERROR RETURN CODE
        JMP     SHORT J34

;--------------------------------------------------------------------------------
; WAIT_INT
;       THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
;       TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
;       IF THE DRIVE IS NOT READY.
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
```

```
;--------------------------------------------------------------------------------

; 17/12/2014
; 2.5 seconds waiting !
;(AWARD BIOS - 1999, WAIT_FDU_INT_LOW, WAIT_FDU_INT_HI)
; amount of time to wait for completion interrupt from NEC.


WAIT_INT:
        STI                             ; TURN ON INTERRUPTS, JUST IN CASE
        CLC                             ; CLEAR TIMEOUT INDICATOR
       ;MOV     BL,10                   ; CLEAR THE COUNTERS
       ;XOR     CX,CX                   ; FOR 2 SECOND WAIT

        ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
        ;
        ;WAIT_FOR_MEM:
        ;       Waits for a bit at a specified memory location pointed
        ;       to by ES:[DI] to become set.
        ;INPUT:
        ;       AH=Mask to test with.
        ;       ES:[DI] = memory location to watch.
        ;       BH:CX=Number of memory refresh periods to delay.
        ;            (normally 30 microseconds per period.)

        ; waiting for (max.) 2.5 secs in 30 micro units.
;       mov cx, WAIT_FDU_INT_LO         ; 017798
;;      mov bl, WAIT_FDU_INT_HI
;       mov bl, WAIT_FDU_INT_HI + 1

;WFMS_CHECK_MEM:
;       test    byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
;       jnz     shortr J37
;WFMS_HI:
;       IN      AL,PORT_B  ; 061h       ; SYS1, wait for lo to hi
;       TEST    AL,010H                 ; transition on memory
;       JNZ     SHORT WFMS_HI           ; refresh.
;WFMS_LO:
;       IN      AL,PORT_B               ;SYS1
;       TEST    AL,010H
;       JZ      SHORT WFMS_LO
;       LOOP    SHORT WFMS_CHECK_MEM
;WFMS_OUTER_LP:
;;      or      bl, bl                  ; check outer counter
;;      jz      short J36A              ; WFMS_TIMEOUT
;       dec     bl
;       jz      short J36A
;       jmp     short WFMS_CHECK_MEM

        ;17/12/2014
        ;16/12/2014
        mov     byte [wait_count], 0    ; Reset (INT 08H) counter
J36:
        TEST    byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
        JNZ     short J37
        ;16/12/2014
        ;LOOP   J36                     ; COUNT DOWN WHILE WAITING
        ;DEC    BL                      ; SECOND LEVEL COUNTER
        ;JNZ    short J36
        cmp     byte [wait_count], 46   ; (46/18.2 seconds)
        jb      short J36

;WFMS_TIMEOUT:
J36A:
```

```
        OR      byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
        STC                             ; ERROR RETURN
J37:
        PUSHF                           ; SAVE CURRENT CARRY
        AND     byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
        POPF                            ; RECOVER CARRY
        RETn                            ; GOOD RETURN CODE

;--------------------------------------------------------------------------------
; RESULTS
;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
;       FOLLOWING AN INTERRUPT.
;
; ON EXIT:      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
;               AX,BX,CX,DX DESTROYED
;--------------------------------------------------------------------------------
RESULTS:
        PUSH    eDI
        MOV     eDI, NEC_STATUS         ; POINTER TO DATA AREA
        MOV     BL,7                    ; MAX STATUS BYTES
        MOV     DX,03F4H                ; STATUS PORT

;-----  WAIT FOR REQUEST FOR MASTER

_R10:
        ; 16/12/2014
        ; wait for (max) 0.5 seconds
        ;MOV    BH,2                    ; HIGH ORDER COUNTER
        ;XOR    CX,CX                   ; COUNTER

        ;Time to wait while waiting for each byte of NEC results = .5
        ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
        mov     cx, WAIT_FDU_RESULTS_LO ; 16667
        mov     bh, WAIT_FDU_RESULTS_HI+1 ; 0+1

WFPSR_OUTER_LP:
        ;
WFPSR_CHECK_PORT:
J39:                                    ; WAIT FOR MASTER
        IN      AL,DX                   ; GET STATUS
        AND     AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
        CMP     AL,11000000B            ; STATUS 1 AND DIRECTION 1 ?
        JZ      short J42               ; STATUS AND DIRECTION OK
WFPSR_HI:
        IN      AL, PORT_B      ;061h   ; SYS1  ; wait for hi to lo
        TEST    AL,010H                 ; transition on memory
        JNZ     SHORT WFPSR_HI          ; refresh.
WFPSR_LO:
        IN      AL, PORT_B              ; SYS1
        TEST    AL,010H
        JZ      SHORT WFPSR_LO
        LOOP    WFPSR_CHECK_PORT
        ;
        dec     bh
        jnz     short WFPSR_OUTER_LP
        ;jmp    short WFPSR_TIMEOUT     ; fail

        ;;mov   byte [wait_count], 0
;J39:                                   ; WAIT FOR MASTER
;       IN      AL,DX                   ; GET STATUS
;       AND     AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
;       CMP     AL,11000000B            ; STATUS 1 AND DIRECTION 1 ?
;       JZ      short J42               ; STATUS AND DIRECTION OK
        ;LOOP   J39                     ; LOOP TILL TIMEOUT
```

```
        ;DEC    BH                      ; DECREMENT HIGH ORDER COUNTER
        ;JNZ    short J39               ; REPEAT TILL DELAY DONE
        ;
        ;;cmp   byte [wait_count], 10  ; (10/18.2 seconds)
        ;;jb    short J39

;WFPSR_TIMEOUT:
        OR      byte [DSKETTE_STATUS],TIME_OUT
        STC                             ; SET ERROR RETURN
        JMP     SHORT POPRES            ; POP REGISTERS AND RETURN

;-----  READ IN THE STATUS

J42:
        ;JMP    $+2                     ; I/O DELAY
        INC     DX                      ; POINT AT DATA PORT
        IN      AL,DX                   ; GET THE DATA
        ; 16/12/2014
        NEWIODELAY
        MOV     [eDI],AL                ; STORE THE BYTE
        INC     eDI                     ; INCREMENT THE POINTER
        ; 16/12/2014
;       push    cx
;       mov     cx, 30
;wdw2:
;       NEWIODELAY
;       loop    wdw2
;       pop     cx

        MOV     eCX,3                   ; MINIMUM 24 MICROSECONDS FOR NEC
        CALL    WAITF                   ; WAIT 30 TO 45 MICROSECONDS
        DEC     DX                      ; POINT AT STATUS PORT
        IN      AL,DX                   ; GET STATUS
        ; 16/12/2014
        NEWIODELAY
        ;
        TEST    AL,00010000B            ; TEST FOR NEC STILL BUSY
        JZ      short POPRES            ; RESULTS DONE ?

        DEC     BL                      ; DECREMENT THE STATUS COUNTER
        JNZ     short _R10              ; GO BACK FOR MORE
        OR      byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
        STC                             ; SET ERROR FLAG

;-----  RESULT OPERATION IS DONE
POPRES:
        POP     eDI
        RETn                            ; RETURN WITH CARRY SET

;--------------------------------------------------------------------------
; READ_DSKCHNG
;       READS THE STATE OF THE DISK CHANGE LINE.
;
; ON ENTRY:     DI = DRIVE #
;
; ON EXIT:      DI = DRIVE #
;               ZF = 0 : DISK CHANGE LINE INACTIVE
;               ZF = 1 : DISK CHANGE LINE ACTIVE
;               AX,CX,DX DESTROYED
;--------------------------------------------------------------------------
READ_DSKCHNG:
        CALL    MOTOR_ON                ; TURN ON THE MOTOR IF OFF
        MOV     DX,03F7H                ; ADDRESS DIGITAL INPUT REGISTER
        IN      AL,DX                   ; INPUT DIGITAL INPUT REGISTER
```

```
        TEST    AL,DSK_CHG              ; CHECK FOR DISK CHANGE LINE ACTIVE
        RETn                            ; RETURN TO CALLER WITH ZERO FLAG SET

;--------------------------------------------------------------------------------
; DRIVE_DET
;       DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
;       UPDATES STATE INFORMATION ACCORDINGLY.
;üğpol*ğj       56789*87111"; ON ENTRY: DI = DRIVE #
;--------------------------------------------------------------------------------
DRIVE_DET:
        CALL    MOTOR_ON                ; TURN ON MOTOR IF NOT ALREADY ON
        CALL    RECAL                   ; RECALIBRATE DRIVE
        JC      short DD_BAC            ; ASSUME NO DRIVE PRESENT
        MOV     CH,TRK_SLAP             ; SEEK TO TRACK 48
        CALL    SEEK
        JC      short DD_BAC            ; ERROR NO DRIVE
        MOV     CH,QUIET_SEEK+1         ; SEEK TO TRACK 10
SK_GIN:
        DEC     CH                      ; DECREMENT TO NEXT TRACK
        PUSH    CX                      ; SAVE TRACK
        CALL    SEEK
        JC      short POP_BAC           ; POP AND RETURN
        MOV     eAX, POP_BAC            ; LOAD NEC OUTPUT ERROR ADDRESS
        PUSH    eAX
        MOV     AH,SENSE_DRV_ST         ; SENSE DRIVE STATUS COMMAND BYTE
        CALL    NEC_OUTPUT              ; OUTPUT TO NEC
        MOV     AX,DI                   ; AL = DRIVE
        MOV     AH,AL                   ; AH = DRIVE
        CALL    NEC_OUTPUT              ; OUTPUT TO NEC
        CALL    RESULTS                 ; GO GET STATUS
        POP     eAX                     ; THROW AWAY ERROR ADDRESS
        POP     CX                      ; RESTORE TRACK
        TEST    byte [NEC_STATUS], HOME ; TRACK 0 ?
        JZ      short SK_GIN            ; GO TILL TRACK 0
        OR      CH,CH                   ; IS HOME AT TRACK 0
        JZ      short IS_80             ; MUST BE 80 TRACK DRIVE

;       DRIVE IS A 360; SET DRIVE TO DETERMINED;
;       SET MEDIA TO DETERMINED AT RATE 250.

        OR      byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
        RETn                            ; ALL INFORMATION SET
IS_80:
        OR      byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
DD_BAC:
        RETn
POP_BAC:
        POP     CX                      ; THROW AWAY
        RETn

fdc_int:  ; 16/02/2015
;int_0Eh: ; 11/12/2014

;--- HARDWARE INT 0EH -- ( IRQ LEVEL  6 ) -------------------------------------
; DISK_INT
;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
;
; ON EXIT:      THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
;--------------------------------------------------------------------------------
DISK_INT_1:

        PUSH    AX                      ; SAVE WORK REGISTER
        ;OR      byte [CS:SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
        or      byte [SEEK_STATUS], INT_FLAG ; 08/02/2015
```

```
        MOV     AL,EOI                  ; END OF INTERRUPT MARKER
        OUT     INTA00,AL               ; INTERRUPT CONTROL PORT
        POP     AX                      ; RECOVER REGISTER
        IRET                            ; RETURN FROM INTERRUPT

;--------------------------------------------------------------------------------
; DSKETTE_SETUP
;       THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
;       DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
;--------------------------------------------------------------------------------
DSKETTE_SETUP:
        ;PUSH   AX                      ; SAVE REGISTERS
        ;PUSH   BX
        ;PUSH   CX
        PUSH    eDX
        ;PUSH   DI
        ;;PUSH  DS
        ; 14/12/2014
        ;mov    word [DISK_POINTER], MD_TBL6
        ;mov    word [DISK_POINTER+2], cs
        ;
        ;OR     byte [RTC_WAIT_FLAG], 1 ; NO RTC WAIT, FORCE USE OF LOOP
        XOR     eDI,eDI                 ; INITIALIZE DRIVE POINTER
        MOV     WORD [DSK_STATE],0      ; INITIALIZE STATES
        AND     byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
        OR      byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
        MOV     byte [SEEK_STATUS],0    ; INDICATE RECALIBRATE NEEDED
        MOV     byte [MOTOR_COUNT],0    ; INITIALIZE MOTOR COUNT
        MOV     byte [MOTOR_STATUS],0   ; INITIALIZE DRIVES TO OFF STATE
        MOV     byte [DSKETTE_STATUS],0 ; NO ERRORS

        ; 28/02/2015
        ;mov    word [cfd], 100h
        call    DSK_RESET
        pop     edx
        retn

;SUP0:
;       CALL    DRIVE_DET               ; DETERMINE DRIVE
;       CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
;       ; 02/01/2015
;       ;INC    DI                      ; POINT TO NEXT DRIVE
;       ;CMP    DI,MAX_DRV              ; SEE IF DONE
;       ;JNZ    short SUP0              ; REPEAT FOR EACH ORIVE
;       cmp     byte [fd1_type], 0
;       jna     short sup1
;       or      di, di
;       jnz     short sup1
;       inc     di
;       jmp     short SUP0
;sup1:
;       MOV     byte [SEEK_STATUS],0    ; FORCE RECALIBRATE
;       ;AND    byte [RTC_WAIT_FLAG],0FEH ; ALLOW FOR RTC WAIT
;       CALL    SETUP_END               ; VARIOUS CLEANUPS
;       ;;POP   DS                      ; RESTORE CALLERS REGISTERS
;       ;POP    DI
;       POP     eDX
;       ;POP    CX
;       ;POP    BX
;       ;POP    AX
;       RETn

;////////////////////////////////////////////////
;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;

int13h: ; 21/02/2015
        pushfd
        push    cs
        call    DISK_IO
        retn

;;;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
;////////////////////////////////////////////////////////////////////////

; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
; 23/02/2015
; 21/02/2015 (unix386.s)
; 22/12/2014 - 14/02/2015 (dsectrm2.s)
;
; Original Source Code:
; DISK ----- 09/25/85 FIXED DISK BIOS
; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
;
; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
;                Source Code - ATORGS.ASM, AHDSK.ASM
;


;The wait for controller to be not busy is 10 seconds.
;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
;;WAIT_HDU_CTLR_BUSY_LO equ     1615h
;;WAIT_HDU_CTLR_BUSY_HI equ       05h
WAIT_HDU_CTRL_BUSY_LH   equ     51615h   ;21/02/2015

;The wait for controller to issue completion interrupt is 10 seconds.
;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
;;WAIT_HDU_INT_LO       equ     1615h
;;WAIT_HDU_INT_HI       equ       05h
WAIT_HDU_INT_LH         equ     51615h  ; 21/02/2015

;The wait for Data request on read and write longs is
;2000 us. (?)
;;WAIT_HDU_DRQ_LO       equ     1000    ; 03E8h
;;WAIT_HDU_DRQ_HI       equ     0
WAIT_HDU_DRQ_LH         equ     1000    ; 21/02/2015

; Port 61h (PORT_B)
SYS1            equ     61h     ; PORT_B  (diskette.inc)

; 23/12/2014
%define CMD_BLOCK       eBP-8  ; 21/02/2015


; << diskette.inc >>
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;---------------------------------------
;       ROM BIOS DATA AREAS             :
;---------------------------------------

;DATA           SEGMENT AT 40H          ; ADDRESS= 0040:0000

;---------------------------------------
;       FIXED DISK DATA AREAS           :
;---------------------------------------

DISK_STATUS1:   DB      0               ; FIXED DISK STATUS
```

```
HF_NUM:         DB      0               ; COUNT OF FIXED DISK DRIVES
CONTROL_BYTE:   DB      0               ; HEAD CONTROL BYTE
;@PORT_OFF      DB      ?               ;  RESERVED (PORT OFFSET)

;---------------------------------------
;       ADDITIONAL MEDIA DATA           :
;---------------------------------------

;@LASTRATE      DB      ?               ; LAST DISKETTE DATA RATE SELECTED
HF_STATUS       DB      0               ; STATUS REGISTER
HF_ERROR        DB      0               ; ERROR REGISTER
HF_INT_FLAG     DB      0               ; FIXED DISK INTERRUPT FLAG
HF_CNTRL        DB      0               ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
;@DSK_STATE     DB      ?               ; DRIVE 0 MEDIA STATE
;               DB      ?               ; DRIVE 1 MEDIA STATE
;               DB      ?               ; DRIVE 0 OPERATION START STATE
;               DB      ?               ; DRIVE 1 OPERATION START STATE
;@DSK_TRK       DB      ?               ; DRIVE 0 PRESENT CYLINDER
;               DB      ?               ; DRIVE 1 PRESENT CYLINDER

;DATA           ENDS                    ; END OF BIOS DATA SEGMENT
;
; ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


;--- INT 13H ----------------------------------------------------------------
;                                                                            :
; FIXED DISK I/O INTERFACE                                                   :
;                                                                            :
;       THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH         :
;       THE IBM FIXED DISK CONTROLLER.                                       :
;                                                                            :
;       THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH          :
;       SOFTWARE  INTERRUPTS  ONLY.    ANY  ADDRESSES  PRESENT  IN           :
;       THESE  LISTINGS  ARE  INCLUDED   ONLY   FOR  COMPLETENESS,           :
;       NOT  FOR  REFERENCE.  APPLICATIONS   WHICH  REFERENCE  ANY           :
;       ABSOLUTE  ADDRESSES  WITHIN  THE  CODE  SEGMENTS  OF  BIOS           :
;       VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                      :
;                                                                            :
;----------------------------------------------------------------------------:
;                                                                            :
; INPUT  (AH)= HEX COMMAND VALUE                                             :
;                                                                            :
;       (AH)= 00H  RESET DISK (DL = 80H,81H) / DISKETTE                      :
;       (AH)= 01H  READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)      :
;                   NOTE: DL < 80H - DISKETTE                                :
;                         DL > 80H - DISK                                    :
;       (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY                      :
;       (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY                     :
;       (AH)= 04H  VERIFY THE DESIRED SECTORS                                :
;       (AH)= 05H  FORMAT THE DESIRED TRACK                                  :
;       (AH)= 06H  UNUSED                                                    :
;       (AH)= 07H  UNUSED                                                    :
;       (AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                       :
;       (AH)= 09H  INITIALIZE DRIVE PAIR CHARACTERISTICS                     :
;                   INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0            :
;                   INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1            :
;       (AH)= 0AH  READ LONG                                                 :
;       (AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
;       (AH)= 0CH  SEEK                                                      :
;       (AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                             :
;       (AH)= 0EH  UNUSED                                                    :
;       (AH)= 0FH  UNUSED                                                    :
;       (AH)= 10H  TEST DRIVE READY                                          :
```

```
;       (AH)= 11H  RECALIBRATE                                                 :
;       (AH)= 12H  UNUSED                                                      :
;       (AH)= 13H  UNUSED                                                      :
;       (AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                              :
;       (AH)= 15H  READ DASD TYPE                                              :
;                                                                              :
;-------------------------------------------------------------------------------
;                                                                              :
;       REGISTERS USED FOR FIXED DISK OPERATIONS                               :
;                                                                              :
;               (DL)    -  DRIVE NUMBER     (80H-81H FOR DISK. VALUE CHECKED)  :
;               (DH)    -  HEAD NUMBER      (0-15 ALLOWED, NOT VALUE CHECKED)  :
;               (CH)    -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL):
;               (CL)    -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)          :
;                                                                              :
;                          NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED     :
;                                IN THE HIGH 2 BITS OF THE CL REGISTER         :
;                                (10 BITS TOTAL)                               :
;                                                                              :
;               (AL)    -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,    :
;                                             FOR READ/WRITE LONG 1-79H)       :
;                                                                              :
;               (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,             :
;                          (NOT REQUIRED FOR VERIFY)                           :
;                                                                              :
;               FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST     :
;                          2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
;                          F = 00H FOR A GOOD SECTOR                           :
;                              80H FOR A BAD SECTOR                            :
;                          N = SECTOR NUMBER                                   :
;                          FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK         :
;                          THE TABLE SHOULD BE:                                :
;                                                                              :
;                  DB      00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH     :
;                  DB      00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH     :
;                  DB      00H,07H,00H,10H,00H,08H,00H,11H,00H,09H             :
;                                                                              :
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; OUTPUT                                                                       :
;       AH = STATUS OF CURRENT OPERATION                                       :
;            STATUS BITS ARE DEFINED IN THE EQUATES BELOW                      :
;       CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)                          :
;       CY = 1  FAILED OPERATION (AH HAS ERROR REASON)                         :
;                                                                              :
;       NOTE:   ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE      :
;               ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA       :
;               IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN      :
;               ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE       :
;               FOR ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF  THE DATA IS      :
;               REWRITTEN.                                                     :
;                                                                              :
;       IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),                        :
;          INPUT:                                                              :
;            (DL) = DRIVE NUMBER                                               :
;          OUTPUT:                                                             :
;            (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)  :
;                   (CONTROLLER CARD ZERO TALLY ONLY)                          :
;            (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER                      :
;            (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER                  :
;            (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER                    :
;                   AND CYLINDER NUMBER HIGH BITS                              :
;                                                                              :
```

```
;       IF READ DASD TYPE WAS REQUESTED,                                       :
;                                                                              :
;       AH = 0 - NOT PRESENT                                                   :
;            1 - DISKETTE - NO CHANGE LINE AVAILABLE                           :
;            2 - DISKETTE - CHANGE LINE AVAILABLE                              :
;            3 - FIXED DISK                                                    :
;                                                                              :
;       CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                          :
;                                                                              :
;       REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN        :
;       INFORMATION.                                                           :
;                                                                              :
;       NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE        :
;               ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.         :
;                                                                              :
;-------------------------------------------------------------------------------

SENSE_FAIL      EQU     0FFH            ; NOT IMPLEMENTED
NO_ERR          EQU     0E0H            ; STATUS ERROR/ERROR REGISTER=0
WRITE_FAULT     EQU     0CCH            ; WRITE FAULT ON SELECTED DRIVE
UNDEF_ERR       EQU     0BBH            ; UNDEFINED ERROR OCCURRED
NOT_RDY         EQU     0AAH            ; DRIVE NOT READY
TIME_OUT        EQU     80H             ; ATTACHMENT FAILED TO RESPOND
BAD_SEEK        EQU     40H             ; SEEK OPERATION FAILED
BAD_CNTLR       EQU     20H             ; CONTROLLER HAS FAILED
DATA_CORRECTED  EQU     11H             ; ECC CORRECTED DATA ERROR
BAD_ECC         EQU     10H             ; BAD ECC ON DISK READ
BAD_TRACK       EQU     0BH             ; NOT IMPLEMENTED
BAD_SECTOR      EQU     0AH             ; BAD SECTOR FLAG DETECTED
;DMA_BOUNDARY   EQU     09H             ; DATA EXTENDS TOO FAR
INIT_FAIL       EQU     07H             ; DRIVE PARAMETER ACTIVITY FAILED
BAD_RESET       EQU     05H             ; RESET FAILED
;RECORD_NOT_FND EQU     04H             ; REQUESTED SECTOR NOT FOUND
;BAD_ADDR_MARK  EQU     02H             ; ADDRESS MARK NOT FOUND
;BAD_CMD        EQU     01H             ; BAD COMMAND PASSED TO DISK I/O

;--------------------------------------------------------
;                                                       :
; FIXED DISK PARAMETER TABLE                            :
;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:      :
;                                                       :
;  +0   (1 WORD) - MAXIMUM NUMBER OF CYLINDERS          :
;  +2   (1 BYTE) - MAXIMUM NUMBER OF HEADS              :
;  +3   (1 WORD) - NOT USED/SEE PC-XT                   :
;  +5   (1 WORD) - STARTING WRITE PRECOMPENSATION CYL   :
;  +7   (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH        :
;  +8   (1 BYTE) - CONTROL BYTE                         :
;                  BIT    7 DISABLE RETRIES -OR-        :
;                  BIT    6 DISABLE RETRIES             :
;                  BIT    3 MORE THAN 8 HEADS           :
;  +9   (3 BYTES)- NOT USED/SEE PC-XT                   :
; +12   (1 WORD) - LANDING ZONE                         :
; +14   (1 BYTE) - NUMBER OF SECTORS/TRACK              :
; +15   (1 BYTE) - RESERVED FOR FUTURE USE              :
;                                                       :
;        - TO DYNAMICALLY DEFINE A SET OF PARAMETERS    :
;          BUILD A TABLE FOR UP TO 15 TYPES AND PLACE   :
;          THE CORRESPONDING VECTOR INTO INTERRUPT 41   :
;          FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.    :
;                                                       :
;--------------------------------------------------------

;--------------------------------------------------------
;                                                       :
```

```
; HARDWARE SPECIFIC VALUES                                      :
;                                                               :
;  -  CONTROLLER I/O PORT                                       :
;                                                               :
;     > WHEN READ FROM:                                         :
;       HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)          :
;       HF_PORT+1 - GET ERROR REGISTER                          :
;       HF_PORT+2 - GET SECTOR COUNT                            :
;       HF_PORT+3 - GET SECTOR NUMBER                           :
;       HF_PORT+4 - GET CYLINDER LOW                            :
;       HF_PORT+5 - GET CYLINDER HIGH (2 BITS)                  :
;       HF_PORT+6 - GET SIZE/DRIVE/HEAD                         :
;       HF_PORT+7 - GET STATUS REGISTER                         :
;                                                               :
;     > WHEN WRITTEN TO:                                        :
;       HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER)         :
;       HF_PORT+1 - SET PRECOMPENSATION CYLINDER                :
;       HF_PORT+2 - SET SECTOR COUNT                            :
;       HF_PORT+3 - SET SECTOR NUMBER                           :
;       HF_PORT+4 - SET CYLINDER LOW                            :
;       HF_PORT+5 - SET CYLINDER HIGH (2 BITS)                  :
;       HF_PORT+6 - SET SIZE/DRIVE/HEAD                         :
;       HF_PORT+7 - SET COMMAND REGISTER                        :
;                                                               :
;---------------------------------------------------------------

;HF_PORT        EQU     01F0H   ; DISK PORT
;HF1_PORT       equ     0170h
;HF_REG_PORT    EQU     03F6H
;HF1_REG_PORT   equ     0376h

align 2

HF_PORT:        dw      1F0h  ; Default = 1F0h
                              ; (170h)
HF_REG_PORT:    dw      3F6h  ; HF_PORT + 206h

HDC1_BASEPORT   equ     1F0h
HDC2_BASEPORT   equ     170h

; 05/01/2015
hf_m_s:         db      0     ; (0 = Master, 1 = Slave)

align 2

;-----          STATUS REGISTER

ST_ERROR        EQU     00000001B       ;
ST_INDEX        EQU     00000010B       ;
ST_CORRCTD      EQU     00000100B       ; ECC CORRECTION SUCCESSFUL
ST_DRQ          EQU     00001000B       ;
ST_SEEK_COMPL   EQU     00010000B       ; SEEK COMPLETE
ST_WRT_FLT      EQU     00100000B       ; WRITE FAULT
ST_READY        EQU     01000000B       ;
ST_BUSY         EQU     10000000B       ;

;-----          ERROR REGISTER

ERR_DAM         EQU     00000001B       ; DATA ADDRESS MARK NOT FOUND
ERR_TRK_0       EQU     00000010B       ; TRACK 0 NOT FOUND ON RECAL
ERR_ABORT       EQU     00000100B       ; ABORTED COMMAND
;               EQU     00001000B       ; NOT USED
ERR_ID          EQU     00010000B       ; ID NOT FOUND
;               EQU     00100000B       ; NOT USED
```

```
ERR_DATA_ECC    EQU     01000000B
ERR_BAD_BLOCK   EQU     10000000B


RECAL_CMD       EQU     00010000B       ; DRIVE RECAL   (10H)
READ_CMD        EQU     00100000B       ;       READ    (20H)
WRITE_CMD       EQU     00110000B       ;       WRITE   (30H)
VERIFY_CMD      EQU     01000000B       ;       VERIFY  (40H)
FMTTRK_CMD      EQU     01010000B       ; FORMAT TRACK  (50H)
INIT_CMD        EQU     01100000B       ;   INITIALIZE  (60H)
SEEK_CMD        EQU     01110000B       ;       SEEK    (70H)
DIAG_CMD        EQU     10010000B       ; DIAGNOSTIC    (90H)
SET_PARM_CMD    EQU     10010001B       ; DRIVE PARMS   (91H)
NO_RETRIES      EQU     00000001B       ; CHD MODIFIER  (01H)
ECC_MODE        EQU     00000010B       ; CMD MODIFIER  (02H)
BUFFER_MODE     EQU     00001000B       ; CMD MODIFIER  (08H)

;MAX_FILE       EQU     2
;S_MAX_FILE     EQU     2
MAX_FILE        equ     4               ; 22/12/2014
S_MAX_FILE      equ     4               ; 22/12/2014

DELAY_1         EQU     25H             ; DELAY FOR OPERATION COMPLETE
DELAY_2         EQU     0600H           ; DELAY FOR READY
DELAY_3         EQU     0100H           ; DELAY FOR DATA REQUEST

HF_FAIL         EQU     08H             ; CMOS FLAG IN BYTE 0EH

;-----          COMMAND BLOCK REFERENCE

;CMD_BLOCK      EQU     BP-8            ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
                                        ;  (BP) POINTS TO COMMAND BLOCK TAIL
                                        ;       AS DEFINED BY THE "ENTER" PARMS
; 19/12/2014
ORG_VECTOR      equ     4*13h           ; INT 13h vector
DISK_VECTOR     equ     4*40h           ; INT 40h vector (for floppy disks)
;HDISK_INT      equ     4*76h           ; Primary HDC - Hardware interrupt
(IRQ14)
;HDISK_INT1     equ     4*76h           ; Primary HDC - Hardware interrupt
(IRQ14)
;HDISK_INT2     equ     4*77h           ; Secondary HDC - Hardware interrupt
(IRQ15)
;HF_TBL_VEC     equ     4*41h           ; Pointer to 1st fixed disk parameter
table
;HF1_TBL_VEC    equ     4*46h           ; Pointer to 2nd fixed disk parameter
table
;
;HF_TBL_VEC:    dd      0               ; Primary master disk param. tbl.
pointer
;HF1_TBL_VEC:   dd      0               ; Primary slave disk param. tbl. pointer
HF_TBL_VEC: ; 22/12/2014
HDPM_TBL_VEC:   dd      0               ; Primary master disk param. tbl.
pointer
HDPS_TBL_VEC:   dd      0               ; Primary slave disk param. tbl. pointer
HDSM_TBL_VEC:   dd      0               ; Secondary master disk param. tbl.
pointer
HDSS_TBL_VEC:   dd      0               ; Secondary slave disk param. tbl.
pointer

;---------------------------------------
;       FIXED DISK DATA AREAS           :
;---------------------------------------

;@DISK_STATUS1  DB      ?               ; FIXED DISK STATUS
```

```
;@HF_NUM        DB      ?               ; COUNT OF FIXED DISK DRIVES
;@CONTROL_BYTE  DB      ?               ; HEAD CONTROL BYTE
;@PORT_OFF      DB      ?               ;  RESERVED (PORT OFFSET)
;
;port1_off      db      0               ; Hard disk controller 1 - port offset
;port2_off      db      0               ; Hard idsk controller 2 - port offset

align 2

;--------------------------------------------------------------
; FIXED DISK I/O SETUP                                         :
;                                                              :
;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK            :
;  -  PERFORM POWER ON DIAGNOSTICS                             :
;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED      :
;                                                              :
;--------------------------------------------------------------

DISK_SETUP:
        ;CLI
        ;;MOV   AX,ABS0                          ; GET ABSOLUTE SEGMENT
        ;xor    ax,ax
        ;MOV    DS,AX                           ; SET SEGMENT REGISTER
        ;MOV    AX, word [ORG_VECTOR]           ; GET DISKETTE VECTOR
        ;MOV    word [DISK_VECTOR],AX           ;  INTO INT 40H
        ;MOV    AX, word [ORG_VECTOR+2]
        ;MOV    word [DISK_VECTOR+2],AX
        ;MOV    word [ORG_VECTOR],DISK_IO       ; FIXED DISK HANDLER
        ;MOV    word [ORG_VECTOR+2],CS
        ; 1st controller (primary master, slave)   - IRQ 14
        ;;MOV   word [HDISK_INT],HD_INT         ; FIXED DISK INTERRUPT
        ;mov    word [HDISK_INT1],HD_INT        ;
        ;;MOV   word [HDISK_INT+2],CS
        ;mov    word [HDISK_INT1+2],CS
        ; 2nd controller (secondary master, slave) - IRQ 15
        ;mov    word [HDISK_INT2],HD1_INT       ;
        ;mov    word [HDISK_INT2+2],CS
        ;
        ;;MOV   word [HF_TBL_VEC],HD0_DPT       ; PARM TABLE DRIVE 80
        ;;MOV   word [HF_TBL_VEC+2],DPT_SEGM
        ;;MOV   word [HF1_TBL_VEC],HD1_DPT      ; PARM TABLE DRIVE 81
        ;;MOV   word [HF1_TBL_VEC+2],DPT_SEGM
        ;push   cs
        ;pop    ds
        ;mov    word [HDPM_TBL_VEC],HD0_DPT     ; PARM TABLE DRIVE 80h
        ;mov    word [HDPM_TBL_VEC+2],DPT_SEGM
        mov     dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
        ;mov    word [HDPS_TBL_VEC],HD1_DPT     ; PARM TABLE DRIVE 81h
        ;mov    word [HDPS_TBL_VEC+2],DPT_SEGM
        mov     dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
        ;mov    word [HDSM_TBL_VEC],HD2_DPT     ; PARM TABLE DRIVE 82h
        ;mov    word [HDSM_TBL_VEC+2],DPT_SEGM
        mov     dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
        ;mov    word [HDSS_TBL_VEC],HD3_DPT     ; PARM TABLE DRIVE 83h
        ;mov    word [HDSS_TBL_VEC+2],DPT_SEGM
        mov     dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
        ;
        ;;IN    AL,INTB01               ; TURN ON SECOND INTERRUPT CHIP
        ;;;AND  AL,0BFH
        ;;and   al, 3Fh                 ; enable IRQ 14 and IRQ 15
        ;;;JMP  $+2
        ;;IODELAY
        ;;OUT   INTB01,AL
        ;;IODELAY
```

```
        ;;IN    AL,INTA01               ; LET INTERRUPTS PASS THRU TO
        ;;AND   AL,0FBH                 ;  SECOND CHIP
        ;;;JMP  $+2
        ;;IODELAY
        ;;OUT   INTA01,AL
        ;
        ;STI
        ;;PUSH  DS                       ; MOVE ABS0 POINTER TO
        ;;POP   ES                       ; EXTRA SEGMENT POINTER
        ;;;CALL DDS                      ; ESTABLISH DATA SEGMENT
        ;;MOV   byte [DISK_STATUS1],0    ; RESET THE STATUS INDICATOR
        ;;MOV   byte [HF_NUM],0          ; ZERO NUMBER OF FIXED DISKS
        ;;MOV   byte [CONTROL_BYTE],0
        ;;MOV   byte [PORT_OFF],0        ; ZERO CARD OFFSET
        ; 20/12/2014 - private code by Erdogan Tan
                     ; (out of original PC-AT, PC-XT BIOS code)
        ;mov    si, hd0_type
        mov     esi, hd0_type
        ;mov    cx, 4
        mov     ecx, 4
hde_l:
        lodsb
        cmp     al, 80h                 ; 8?h = existing
        jb      short _L4
        inc     byte [HF_NUM]           ; + 1 hard (fixed) disk drives
        loop    hde_l
_L4:                                    ; 0 <= [HF_NUM] =< 4
;L4:
        ;
        ;; 31/12/2014 - cancel controller diagnostics here
        ;;;mov  cx, 3  ; 26/12/2014 (Award BIOS 1999)
        ;;mov   cl, 3
        ;;
        ;;MOV   DL,80H                   ; CHECK THE CONTROLLER
;;hdc_dl:
        ;;MOV   AH,14H                   ; USE CONTROLLER DIAGNOSTIC COMMAND
        ;;INT   13H                      ; CALL BIOS WITH DIAGNOSTIC COMMAND
        ;;;JC   short CTL_ERRX           ; DISPLAY ERROR MESSAGE IF BAD RETURN
        ;;;jc   short POD_DONE ;22/12/2014
        ;;jnc   short hdc_reset0
        ;;loop  hdc_dl
        ;;; 27/12/2014
        ;;stc
        ;;retn
        ;
;;hdc_reset0:
        ; 18/01/2015
        mov     cl, [HF_NUM]
        and     cl, cl
        jz      short POD_DONE
        ;
        mov     dl, 7Fh
hdc_reset1:
        inc     dl
        ;; 31/12/2015
        ;;push  dx
        ;;push  cx
        ;;push  ds
        ;;sub   ax, ax
        ;;mov   ds, ax
        ;;MOV   AX, [TIMER_LOW]          ; GET START TIMER COUNTS
        ;;pop   ds
        ;;MOV   BX,AX
        ;;ADD   AX,6*182                 ; 60 SECONDS* 18.2
```

```
        ;;MOV   CX,AX
        ;;mov   word [wait_count], 0    ; 22/12/2014 (reset wait counter)
        ;;
        ;; 31/12/2014 - cancel HD_RESET_1
        ;;CALL  HD_RESET_1              ; SET UP DRIVE 0, (1,2,3)
        ;;pop   cx
        ;;pop   dx
        ;;
        ; 18/01/2015
        mov     ah, 0Dh ; ALTERNATE RESET
        ;int    13h
        call    int13h
        loop    hdc_reset1
POD_DONE:
        RETn

;;----- POD_ERROR

;;CTL_ERRX:
;       ;MOV    SI,OFFSET F1782         ; CONTROLLER ERROR
;       ;CALL   SET_FAIL                ; DO NOT IPL FROM DISK
;       ;CALL   E_MSG                   ; DISPLAY ERROR AND SET (BP) ERROR FLAG
;       ;JMP    short POD_DONE

;;HD_RESET_1:
;;      ;PUSH   BX                      ; SAVE TIMER LIMITS
;;      ;PUSH   CX
;;RES_1: MOV    AH,09H                  ; SET DRIVE PARAMETERS
;;      INT     13H
;;      JC      short RES_2
;;      MOV     AH,11H                  ; RECALIBRATE DRIVE
;;      INT     13H
;;      JNC     short RES_CK            ; DRIVE OK
;;RES_2: ;CALL  POD_TCHK                ; CHECK TIME OUT
;;      cmp     word [wait_count], 6*182 ; waiting time (in timer ticks)
;;                                      ; (30 seconds)
;;      ;cmc
;;      ;JNC    short RES_1
;;      jb      short RES_1
;;;RES_FL: ;MOV SI,OFFSET F1781         ; INDICATE DISK 1 FAILURE;
;;      ;TEST   DL,1
;;      ;JNZ    RES_E1
;;      ;MOV    SI,OFFSET F1780         ; INDICATE DISK 0 FAILURE
;;      ;CALL   SET_FAIL                ; DO NOT TRY TO IPL DISK 0
;;      ;JMP    SHORT RES_E1
;;RES_ER: ; 22/12/2014
;;RES_OK:
;;      ;POP    CX                      ; RESTORE TIMER LIMITS
;;      ;POP    BX
;;      RETn
;;
;;RES_RS: MOV   AH,00H                  ; RESET THE DRIVE
;;      INT     13H
;;RES_CK: MOV   AH,08H                  ; GET MAX CYLINDER,HEAD,SECTOR
;;      MOV     BL,DL                   ; SAVE DRIVE CODE
;;      INT     13H
;;      JC      short RES_ER
;;      MOV     word [NEC_STATUS],CX    ; SAVE MAX CYLINDER, SECTOR
;;      MOV     DL,BL                   ; RESTORE DRIVE CODE
;;RES_3: MOV    AX,0401H                ; VERIFY THE LAST SECTOR
;;      INT     13H
;;      JNC     short RES_OK            ; VERIFY OK
;;      CMP     AH,BAD_SECTOR           ; OK ALSO IF JUST ID READ
;;      JE      short RES_OK
```

```
;;      CMP     AH,DATA_CORRECTED
;;      JE      short RES_OK
;;      CMP     AH,BAD_ECC
;;      JE      short RES_OK
;;      ;CALL   POD_TCHK                ; CHECK FOR TIME OUT
;;      cmp     word [wait_count], 6*182 ; waiting time (in timer ticks)
;;                                      ; (60 seconds)
;;      cmc
;;      JC      short RES_ER            ; FAILED
;;      MOV     CX,word [NEC_STATUS]    ; GET SECTOR ADDRESS, AND CYLINDER
;;      MOV     AL,CL                   ; SEPARATE OUT SECTOR NUMBER
;;      AND     AL,3FH
;;      DEC     AL                      ; TRY PREVIOUS ONE
;;      JZ      short RES_RS            ; WE'VE TRIED ALL SECTORS ON TRACK
;;      AND     CL,0C0H                 ; KEEP CYLINDER BITS
;;      OR      CL,AL                   ; MERGE SECTOR WITH CYLINDER BITS
;;      MOV     word [NEC_STATUS],CX    ; SAVE CYLINDER, NEW SECTOR NUMBER
;;      JMP     short RES_3             ; TRY AGAIN
;;;RES_ER: MOV  SI,OFFSET F1791         ; INDICATE DISK 1 ERROR
;;      ;TEST   DL,1
;;      ;JNZ    short RES_E1
;;      ;MOV    SI,OFFSET F1790         ; INDICATE DISK 0 ERROR
;;;RES_E1:
;;      ;CALL   E_MSG                   ; DISPLAY ERROR AND SET (BP) ERROR FLAG
;;;RES_OK:
;;      ;POP    CX                      ; RESTORE TIMER LIMITS
;;      ;POP    BX
;;      ;RETn
;
;;SET_FAIL:
;       ;MOV    AX,X*(CMOS_DIAG+NMI)    ; GET CMOS ERROR BYTE
;       ;CALL   CMOS_READ
;       ;OR     AL,HF_FAIL              ; SET DO NOT IPL FROM DISK FLAG
;       ;XCHG   AH,AL                   ; SAVE IT
;       ;CALL   CMOS_WRITE              ; PUT IT OUT
;       ;RETn
;
;;POD_TCHK:                              ; CHECK FOR 30 SECOND TIME OUT
;       ;POP    AX                      ; SAVE RETURN
;       ;POP    CX                      ; GET TIME OUT LIMITS
;       ;POP    BX
;       ;PUSH   BX                      ; AND SAVE THEM AGAIN
;       ;PUSH   CX
;       ;PUSH   AX
;       ;push   ds
;       ;xor    ax, ax
;       ;mov    ds, ax                  ; RESTORE RETURN
;       ;MOV    AX, [TIMER_LOW]         ; AX = CURRENT TIME
;       ;                               ; BX = START TIME
;       ;                               ; CX = END TIME
;       ;pop    ds
;       ;CMP    BX,CX
;       ;JB     short TCHK1             ; START < END
;       ;CMP    BX,AX
;       ;JB     short TCHKG             ; END < START < CURRENT
;       ;JMP    SHORT TCHK2             ; END, CURRENT < START
;;TCHK1: CMP    AX,BX
;;      JB      short TCHKNG            ; CURRENT < START < END
;;TCHK2: CMP    AX,CX
;;      JB      short TCHKG             ; START < CURRENT < END
;;                                      ; OR CURRENT < END < START
;;TCHKNG: STC                           ; CARRY SET INDICATES TIME OUT
;;      RETn
;;TCHKG: CLC                            ; INDICATE STILL TIME
```

```
;;      RETn
;;
;;int_13h:

;--------------------------------------
;       FIXED DISK BIOS ENTRY POINT     :
;--------------------------------------

DISK_IO:
        CMP     DL,80H                  ; TEST FOR FIXED DISK DRIVE
        ;JAE    short A1                ; YES, HANDLE HERE
        ;;;INT  40H                     ; DISKETTE HANDLER
        ;;call  int40h
        jb      DISKETTE_IO_1
;RET_2:
        ;RETf   2                       ; BACK TO CALLER
;       retf    4
A1:
        STI                             ; ENABLE INTERRUPTS
        ;; 04/01/2015
        ;;OR    AH,AH
        ;;JNZ   short A2
        ;;INT   40H                     ; RESET NEC WHEN AH=0
        ;;SUB   AH,AH
        CMP     DL,(80H + S_MAX_FILE - 1)
        JA      short RET_2
        ; 18/01/2015
        or      ah,ah
        jz      short A4
        cmp     ah, 0Dh ; Alternate reset
        jne     short A2
        sub     ah,ah   ; Reset
        jmp     short A4
A2:
        CMP     AH,08H                  ; GET PARAMETERS IS A SPECIAL CASE
        ;JNZ    short A3
        ;JMP    GET_PARM_N
        je      GET_PARM_N
A3:     CMP     AH,15H                  ; READ DASD TYPE IS ALSO
        ;JNZ    short A4
        ;JMP    READ_DASD_TYPE
        je      READ_DASD_TYPE
        ; 02/02/2015
        cmp     ah, 1Dh                 ;(Temporary for Retro UNIX 386 v1)
        ; 12/01/2015
        cmc
        jnc     short A4
        ; 30/01/2015
        ;mov     byte [CS:DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
        mov     byte [DISK_STATUS1], BAD_CMD
        ;jmp    short RET_2
RET_2:
        retf    4
A4:                                     ; SAVE REGISTERS DURING OPERATION
        ENTER   8,0                     ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
        PUSH    eBX                     ;  IN THE STACK, THE COMMAND BLOCK IS:
        PUSH    eCX                     ;   @CMD_BLOCK == BYTE PTR [BP]-8
        PUSH    eDX
        PUSH    DS
        PUSH    ES
        PUSH    eSI
        PUSH    eDI
        ;;04/01/2015
        ;;OR    AH,AH                   ; CHECK FOR RESET
```

```
        ;;JNZ   short A5
        ;;MOV   DL,80H                  ; FORCE DRIVE 80 FOR RESET
;;A5:
        ;push   cs
        ;pop    ds
        ; 21/02/2015
        push    ax
        mov     ax, KDATA
        mov     ds, ax
        mov     es, ax
        pop     ax
        CALL    DISK_IO_CONT            ; PERFORM THE OPERATION
        ;;CALL  DDS                     ; ESTABLISH SEGMENT
        MOV     AH,[DISK_STATUS1]       ; GET STATUS FROM OPERATION
        CMP     AH,1                    ; SET THE CARRY FLAG TO INDICATE
        CMC                             ; SUCCESS OR FAILURE
        POP     eDI                     ; RESTORE REGISTERS
        POP     eSI
        POP     ES
        POP     DS
        POP     eDX
        POP     eCX
        POP     eBX
        LEAVE                           ; ADJUST (SP) AND RESTORE (BP)
        ;RETf   2                       ; THROW AWAY SAVED FLAGS
        retf    4
; 21/02/2015
;       dw --> dd
M1:                                     ; FUNCTION TRANSFER TABLE
        dd      DISK_RESET              ; 000H
        dd      RETURN_STATUS           ; 001H
        dd      DISK_READ               ; 002H
        dd      DISK_WRITE              ; 003H
        dd      DISK_VERF               ; 004H
        dd      FMT_TRK                 ; 005H
        dd      BAD_COMMAND             ; 006H  FORMAT BAD SECTORS
        dd      BAD_COMMAND             ; 007H  FORMAT DRIVE
        dd      BAD_COMMAND             ; 008H  RETURN PARAMETERS
        dd      INIT_DRV                ; 009H
        dd      RD_LONG                 ; 00AH
        dd      WR_LONG                 ; 00BH
        dd      DISK_SEEK               ; 00CH
        dd      DISK_RESET              ; 00DH
        dd      BAD_COMMAND             ; 00EH  READ BUFFER
        dd      BAD_COMMAND             ; 00FH  WRITE BUFFER
        dd      TST_RDY                 ; 010H
        dd      HDISK_RECAL             ; 011H
        dd      BAD_COMMAND             ; 012H  MEMORY DIAGNOSTIC
        dd      BAD_COMMAND             ; 013H  DRIVE DIAGNOSTIC
        dd      CTLR_DIAGNOSTIC         ; 014H  CONTROLLER DIAGNOSTIC
        ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
        dd      BAD_COMMAND             ; 015h
        dd      BAD_COMMAND             ; 016h
        dd      BAD_COMMAND             ; 017h
        dd      BAD_COMMAND             ; 018h
        dd      BAD_COMMAND             ; 019h
        dd      BAD_COMMAND             ; 01Ah
        dd      DISK_READ               ; 01Bh ; LBA read
        dd      DISK_WRITE              ; 01Ch ; LBA write
M1L     EQU    $-M1

DISK_IO_CONT:
        ;;CALL  DDS                     ; ESTABLISH SEGMENT
        CMP     AH,01H                  ; RETURN STATUS
```

```
        ;;JNZ   short SU0
        ;;JMP    RETURN_STATUS
        je      RETURN_STATUS
SU0:
        MOV     byte [DISK_STATUS1],0   ; RESET THE STATUS INDICATOR
        ;;PUSH  BX                      ; SAVE DATA ADDRESS
        ;mov    si, bx ;; 14/02/2015
        mov     esi, ebx ; 21/02/2015
        MOV     BL,[HF_NUM]             ; GET NUMBER OF DRIVES
        ;; 04/01/2015
        ;;PUSH  AX
        AND     DL,7FH                  ; GET DRIVE AS 0 OR 1
                                        ; (get drive number as 0 to 3)
        CMP     BL,DL
        ;;JBE   BAD_COMMAND_POP         ; INVALID DRIVE
        jbe     BAD_COMMAND ;; 14/02/2015
        ;
        ;;03/01/2015
        sub     ebx, ebx
        mov     bl, dl
        ;sub    bh, bh
        mov     [LBAMode], bh   ; 0
        ;;test  byte [bx+hd0_type], 1   ; LBA ready ?
        ;test   byte [ebx+hd0_type], 1
        ;jz     short su1               ; no
        ;inc    byte [LBAMode]
;su1:
        ; 21/02/2015 (32 bit modification)
        ;04/01/2015
        push    ax ; ***
        ;PUSH   ES ; **
        PUSH    DX ; *
        push    ax
        CALL    GET_VEC                 ; GET DISK PARAMETERS
        ; 02/02/2015
        ;mov    ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
        mov     ax, [ebx+16]
        mov     [HF_PORT], ax
        ;mov    dx, [ES:BX+18] ; control port address (3F6h, 376h)
        mov     dx, [ebx+18]
        mov     [HF_REG_PORT], dx
        ;mov    al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
        mov     al, [ebx+20]
        ; 23/02/2015
        test    al, 40h  ; LBA bit (bit 6)
        jz      short su1
        inc     byte [LBAMode] ; 1
su1:
        shr     al, 4
        and     al, 1
        mov     [hf_m_s], al
        ;
        ; 03/01/2015
        ;MOV    AL,byte [ES:BX+8]       ; GET CONTROL BYTE MODIFIER
        mov     al, [ebx+8]
        ;MOV    DX,[HF_REG_PORT]        ; Device Control register
        OUT     DX,AL                   ; SET EXTRA HEAD OPTION
                                        ; Control Byte:  (= 08h, here)
                                        ; bit 0 - 0
                                        ; bit 1 - nIEN (1 = disable irq)
                                        ; bit 2 - SRST (software RESET)
                                        ; bit 3 - use extra heads (8 to 15)
                                        ;         -always set to 1-
                                        ; (bits 3 to 7 are reserved
```

```
                                        ;          for ATA devices)
        MOV     AH,[CONTROL_BYTE]       ; SET EXTRA HEAD OPTION IN
        AND     AH,0C0H                 ; CONTROL BYTE
        OR      AH,AL
        MOV     [CONTROL_BYTE],AH
        ; 04/01/2015
        pop     ax
        pop     dx ; * ;; 14/02/2015
        and     ah, ah  ; Reset function ?
        jnz     short su2
        ;;pop   dx ; * ;; 14/02/2015
        ;pop    es ; **
        pop     ax ; ***
        ;;pop   bx
        jmp     DISK_RESET
su2:
        cmp     byte [LBAMode], 0
        jna     short su3
        ;
        ; 02/02/2015 (LBA read/write function calls)
        cmp     ah, 1Bh
        jb      short lbarw1
        cmp     ah, 1Ch
        ja      short invldfnc
        ;;pop   dx ; * ; 14/02/2015
        ;mov    ax, cx ; Lower word of LBA address (bits 0-15)
        mov     eax, ecx ; LBA address (21/02/2015)
        ;; 14/02/2015
        mov     cl, dl ; 14/02/2015
        ;;mov   dx, bx
        ;mov    dx, si ; higher word of LBA address (bits 16-23)
        ;;mov   bx, di
        ;mov    si, di ; Buffer offset
        jmp     short lbarw2
lbarw1:
        ; convert CHS to LBA
        ;
        ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
        ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
        ;       + Sector - 1
        push    dx ; * ;; 14/02/2015
        ;xor    dh, dh
        xor     edx, edx
        ;mov    dl, [ES:BX+14]  ; sectors per track (logical)
        mov     dl, [ebx+14]
        ;xor    ah, ah
        xor     eax, eax
        ;mov    al, [ES:BX+2]   ; heads (logical)
        mov     al, [ebx+2]
        dec     al
        inc     ax              ; 0 =  256
        mul     dx
                ; AX = # of Heads" * Sectors/Track
        mov     dx, cx
        ;and    cx, 3Fh  ; sector  (1 to 63)
        and     ecx, 3fh
        xchg    dl, dh
        shr     dh, 6
                ; DX = cylinder (0 to 1023)
        ;mul    dx
                ; DX:AX = # of Heads" * Sectors/Track * Cylinder
        mul     edx
        dec     cl  ; sector - 1
        ;add    ax, cx
```

```
        ;adc    dx, 0
                ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
        add     eax, ecx
        pop     cx ; * ; ch = head, cl = drive number (zero based)
        ;push   dx
        ;push   ax
        push    eax
        ;mov    al, [ES:BX+14]  ; sectors per track (logical)
        mov     al, [ebx+14]
        mul     ch
                ;  AX = Head * Sectors/Track
        cwd
        ;pop    dx
        pop     edx
        ;add    ax, dx
        ;pop    dx
        ;adc    dx, 0 ; add carry bit
        add     eax, edx
lbarw2:
        sub     edx, edx ; 21/02/2015
        mov     dl, cl ; 21/02/2015
        mov     byte [CMD_BLOCK], 0 ; Features Register
                                ; NOTE: Features register (1F1h, 171h)
                                ; is not used for ATA device R/W functions.
                                ; It is old/obsolete 'write precompensation'
                                ; register and error register
                                ; for old ATA/IDE devices.
        ; 18/01/2014
        ;mov    ch, [hf_m_s]    ; Drive 0 (master) or 1 (slave)
        mov     cl, [hf_m_s]
        ;shl    ch, 4           ; bit 4 (drive bit)
        ;or     ch, 0E0h        ; bit 5 = 1
                                ; bit 6 = 1 = LBA mode
                                ; bit 7 = 1
        or      cl, 0Eh ; 1110b
        ;and    dh, 0Fh         ; LBA byte 4 (bits 24 to 27)
        and     eax, 0FFFFFFFh
        shl     ecx, 28 ; 21/02/2015
        ;or     dh, ch
        or      eax, ecx
        ;;mov   [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
                                 ; (Sector Number Register)
        ;;mov   [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
                                 ; (Cylinder Low Register)
        ;mov    [CMD_BLOCK+2], ax ; LBA byte 1, 2
        ;mov    [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
                                 ; (Cylinder High Register)
        ;;mov   [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
                                 ; (Drive/Head Register)

        ;mov    [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
        mov     [CMD_BLOCK+2], eax ; 21/02/2015
        ;14/02/2015
        ;mov    dl, cl ; Drive number (INIT_DRV)
        jmp     short su4
su3:
        ; 02/02/2015
        ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
        cmp     ah, 14h
        jna     short chsfnc
invldfnc:
        ; 14/02/2015
        ;pop    es ; **
        pop     ax ; ***
```

```
        ;jmp     short BAD_COMMAND_POP
        jmp     short BAD_COMMAND
chsfnc:
        ;MOV    AX,[ES:BX+5]            ; GET WRITE PRE-COMPENSATION CYLINDER
        mov     ax, [ebx+5]
        SHR     AX,2
        MOV     [CMD_BLOCK],AL
        ;;MOV   AL,[ES:BX+8]            ; GET CONTROL BYTE MODIFIER
        ;;PUSH  DX
        ;;MOV   DX,[HF_REG_PORT]
        ;;OUT   DX,AL                   ; SET EXTRA HEAD OPTION
        ;;POP   DX ; *
        ;;POP   ES ; **
        ;;MOV   AH,[CONTROL_BYTE]       ; SET EXTRA HEAD OPTION IN
        ;;AND   AH,0C0H                 ; CONTROL BYTE
        ;;OR    AH,AL
        ;;MOV   [CONTROL_BYTE],AH
        ;
        MOV     AL,CL                   ; GET SECTOR NUMBER
        AND     AL,3FH
        MOV     [CMD_BLOCK+2],AL
        MOV     [CMD_BLOCK+3],CH        ; GET CYLINDER NUMBER
        MOV     AL,CL
        SHR     AL,6
        MOV     [CMD_BLOCK+4],AL        ; CYLINDER HIGH ORDER 2 BITS
        ;;05/01/2015
        ;;MOV   AL,DL                   ; DRIVE NUMBER
        mov     al, [hf_m_s]
        SHL     AL,4
        AND     DH,0FH                  ; HEAD NUMBER
        OR      AL,DH
        ;OR     AL,80H or 20H
        OR      AL,80h+20h              ; ECC AND 512 BYTE SECTORS
        MOV     [CMD_BLOCK+5],AL        ; ECC/SIZE/DRIVE/HEAD
su4:
        ;POP    ES ; **
        ;; 14/02/2015
        ;;POP   AX
        ;;MOV   [CMD_BLOCK+1],AL        ; SECTOR COUNT
        ;;PUSH  AX
        ;;MOV   AL,AH                   ; GET INTO LOW BYTE
        ;;XOR   AH,AH                   ; ZERO HIGH BYTE
        ;;SAL   AX,1                    ; *2 FOR TABLE LOOKUP
        pop     ax ; ***
        mov     [CMD_BLOCK+1], al
        sub     ebx, ebx
        mov     bl, ah
        ;xor     bh, bh
        ;sal     bx, 1
        sal     bx, 2   ; 32 bit offset (21/02/2015)
        ;;MOV   SI,AX                   ; PUT INTO SI FOR BRANCH
        ;;CMP   AX,M1L                  ; TEST WITHIN RANGE
        ;;JNB   short BAD_COMMAND_POP
        ;cmp     bx, M1L
        cmp     ebx, M1L
        jnb     short BAD_COMMAND
        ;xchg    bx, si
        xchg    ebx, esi
        ;;;POP  AX                     ; RESTORE AX
        ;;;POP  BX                     ; AND DATA ADDRESS

        ;;PUSH  CX
        ;;PUSH  AX                     ; ADJUST ES:BX
        ;MOV    CX,BX                  ; GET 3 HIGH ORDER NIBBLES OF BX
```

```
        ;SHR    CX,4
        ;MOV    AX,ES
        ;ADD    AX,CX
        ;MOV    ES,AX
        ;AND    BX,000FH                ; ES:BX CHANGED TO ES:000X
        ;;POP   AX
        ;;POP   CX
        ;;JMP   word [CS:SI+M1]
        ;jmp    word [SI+M1]
        jmp     dword [esi+M1]
;;BAD_COMMAND_POP:
;;      POP     AX
;;      POP     BX
BAD_COMMAND:
        MOV     byte [DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
        MOV     AL,0
        RETn

;---------------------------------------
;       RESET THE DISK SYSTEM  (AH=00H) :
;---------------------------------------

; 18-1-2015 : one controller reset (not other one)

DISK_RESET:
        CLI
        IN      AL,INTB01               ; GET THE MASK REGISTER
        ;JMP    $+2
        IODELAY
        ;AND    AL,0BFH                 ; ENABLE FIXED DISK INTERRUPT
        and     al,3Fh                  ; 22/12/2014 (IRQ 14 & IRQ 15)
        OUT     INTB01,AL
        STI                             ; START INTERRUPTS
        ; 14/02/2015
        mov     di, dx
        ; 04/01/2015
        ;xor    di,di
drst0:
        MOV     AL,04H  ; bit 2 - SRST
        ;MOV    DX,HF_REG_PORT
        MOV     DX,[HF_REG_PORT]
        OUT     DX,AL                   ; RESET
;       MOV     CX,10                   ; DELAY COUNT
;DRD:   DEC     CX
;       JNZ     short DRD               ; WAIT 4.8 MICRO-SEC
        ;mov    cx,2                    ; wait for 30 micro seconds
        mov     ecx, 2 ; 21/02/2015
        call    WAITF                   ; (Award Bios 1999 - WAIT_REFRESH,
                                        ; 40 micro seconds)
        mov     al,[CONTROL_BYTE]
        AND     AL,0FH                  ; SET HEAD OPTION
        OUT     DX,AL                   ; TURN RESET OFF
        CALL    NOT_BUSY
        JNZ     short DRERR             ; TIME OUT ON RESET
        MOV     DX,[HF_PORT]
        inc     dl  ; HF_PORT+1
        ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
        ;mov     cl, 10
        mov     ecx, 10 ; 21/02/2015
drst1:
        IN      AL,DX                   ; GET RESET STATUS
        CMP     AL,1
        ; 04/01/2015
        jz      short drst2
```

```
        ;JNZ    short DRERR             ; BAD RESET STATUS
                ; Drive/Head Register - bit 4
        loop    drst1
DRERR:
        MOV     byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
        RETn
drst2:
        ; 14/02/2015
        mov     dx,di
;drst3:
;       ; 05/01/2015
;       shl     di,1
;       ; 04/01/2015
;       mov     ax,[di+hd_cports]
;       cmp     ax,[HF_REG_PORT]
;       je      short drst4
;       mov     [HF_REG_PORT], ax
;       ; 03/01/2015
;       mov     ax,[di+hd_ports]
;       mov     [HF_PORT], ax
;       ; 05/01/2014
;       shr     di,1
;       ; 04/01/2015
;       jmp     short drst0     ; reset other controller
;drst4:
;       ; 05/01/2015
;       shr     di,1
;       mov     al,[di+hd_dregs]
;       and     al,10h ; bit 4 only
;       shr     al,4 ; bit 4  -> bit 0
;       mov     [hf_m_s], al ; (0 = master, 1 = slave)
        ;
        mov     al, [hf_m_s] ; 18/01/2015
        test    al,1
;       jnz     short drst6
        jnz     short drst4
        AND     byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
;drst5:
drst3:
        CALL    INIT_DRV                ; SET MAX HEADS
        ;mov    dx,di
        CALL    HDISK_RECAL             ; RECAL TO RESET SEEK SPEED
        ; 04/01/2014
;       inc     di
;       mov     dx,di
;       cmp     dl,[HF_NUM]
;       jb      short drst3
;DRE:
        MOV     byte [DISK_STATUS1],0   ; IGNORE ANY SET UP ERRORS
        RETn
;drst6:
drst4:          ; Drive/Head Register - bit 4
        OR      byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
        ;jmp    short drst5
        jmp     short drst3

;---------------------------------------
;       DISK STATUS ROUTINE  (AH = 01H) :
;---------------------------------------

RETURN_STATUS:
        MOV     AL,[DISK_STATUS1]       ; OBTAIN PREVIOUS STATUS
        MOV     byte [DISK_STATUS1],0   ; RESET STATUS
        RETn
```

```
;---------------------------------------
;       DISK READ ROUTINE    (AH = 02H) :
;---------------------------------------

DISK_READ:
        MOV     byte [CMD_BLOCK+6],READ_CMD
        JMP     COMMANDI

;---------------------------------------
;       DISK WRITE ROUTINE   (AH = 03H) :
;---------------------------------------

DISK_WRITE:
        MOV     byte [CMD_BLOCK+6],WRITE_CMD
        JMP     COMMANDO

;---------------------------------------
;       DISK VERIFY          (AH = 04H) :
;---------------------------------------

DISK_VERF:
        MOV     byte [CMD_BLOCK+6],VERIFY_CMD
        CALL    COMMAND
        JNZ     short VERF_EXIT         ; CONTROLLER STILL BUSY
        CALL    _WAIT                   ; (Original: CALL WAIT)
        JNZ     short VERF_EXIT         ; TIME OUT
        CALL    CHECK_STATUS
VERF_EXIT:
        RETn

;---------------------------------------
;       FORMATTING           (AH = 05H) :
;---------------------------------------

FMT_TRK:                                ; FORMAT TRACK  (AH = 005H)
        MOV     byte [CMD_BLOCK+6],FMTTRK_CMD
        ;PUSH   ES
        ;PUSH   BX
        push    ebx
        CALL    GET_VEC                 ; GET DISK PARAMETERS ADDRESS
        ;MOV    AL,[ES:BX+14]           ; GET SECTORS/TRACK
        mov     al, [ebx+14]
        MOV     [CMD_BLOCK+1],AL        ; SET SECTOR COUNT IN COMMAND
        pop     ebx
        ;POP    BX
        ;POP    ES
        JMP     CMD_OF                  ; GO EXECUTE THE COMMAND

;---------------------------------------
;       READ DASD TYPE       (AH = 15H) :
;---------------------------------------

READ_DASD_TYPE:
READ_D_T:                               ; GET DRIVE PARAMETERS
        PUSH    DS                      ; SAVE REGISTERS
        ;PUSH   ES
        PUSH    eBX
        ;CALL   DDS                     ; ESTABLISH ADDRESSING
        ;push   cs
        ;pop    ds
        mov     bx, KDATA
        mov     ds, bx
        ;mov    es, bx
```

```
        MOV     byte [DISK_STATUS1],0
        MOV     BL,[HF_NUM]             ; GET NUMBER OF DRIVES
        AND     DL,7FH                  ; GET DRIVE NUMBER
        CMP     BL,DL
        JBE     short RDT_NOT_PRESENT   ; RETURN DRIVE NOT PRESENT
        CALL    GET_VEC                 ; GET DISK PARAMETER ADDRESS
        ;MOV    AL,[ES:BX+2]            ; HEADS
        mov     al, [ebx+2]
        ;MOV    CL,[ES:BX+14]
        mov     cl, [ebx+14]
        IMUL    CL                      ; * NUMBER OF SECTORS
        ;MOV    CX,[ES:BX]              ; MAX NUMBER OF CYLINDERS
        mov     cx ,[ebx]
        ;
        ; 02/01/2015
        ; ** leave the last cylinder as reserved for diagnostics **
        ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
        DEC     CX                      ; LEAVE ONE FOR DIAGNOSTICS
        ;
        IMUL    CX                      ; NUMBER OF SECTORS
        MOV     CX,DX                   ; HIGH ORDER HALF
        MOV     DX,AX                   ; LOW ORDER HALF
        ;SUB    AX,AX
        sub     al, al
        MOV     AH,03H                  ; INDICATE FIXED DISK
RDT2:   POP     eBX                     ; RESTORE REGISTERS
        ;POP    ES
        POP     DS
        CLC                             ; CLEAR CARRY
        ;RETf   2
        retf    4
RDT_NOT_PRESENT:
        SUB     AX,AX                   ; DRIVE NOT PRESENT RETURN
        MOV     CX,AX                   ; ZERO BLOCK COUNT
        MOV     DX,AX
        JMP     short RDT2

;----------------------------------------
;       GET PARAMETERS       (AH = 08H) :
;----------------------------------------

GET_PARM_N:
;GET_PARM:                              ; GET DRIVE PARAMETERS
        PUSH    DS                      ; SAVE REGISTERS
        ;PUSH   ES
        PUSH    eBX
        ;MOV    AX,ABS0                 ; ESTABLISH ADDRESSING
        ;MOV    DS,AX
        ;TEST   DL,1                    ; CHECK FOR DRIVE 1
        ;JZ     short G0
        ;LES    BX,@HF1_TBL_VEC
        ;JMP    SHORT G1
;G0:    LES     BX,@HF_TBL_VEC
;G1:
        ;CALL   DDS                     ; ESTABLISH SEGMENT
        ; 22/12/2014
        ;push   cs
        ;pop    ds
        mov     bx, KDATA
        mov     ds, bx
        ;mov    es, bx
        ;
        SUB     DL,80H
        CMP     DL,MAX_FILE             ; TEST WITHIN RANGE
```

```
        JAE     short G4
        ;
        xor     ebx, ebx ; 21/02/2015
        ; 22/12/2014
        mov     bl, dl
        ;xor    bh, bh
        shl     bl, 2                   ; convert index to offset
        ;add    bx, HF_TBL_VEC
        add     ebx, HF_TBL_VEC
        ;mov    ax, [bx+2]
        ;mov    es, ax                  ; dpt segment
        ;mov    bx, [bx]                ; dpt offset
        mov     ebx, [ebx] ; 32 bit offset

        MOV     byte [DISK_STATUS1],0
        ;MOV     AX,[ES:BX]              ; MAX NUMBER OF CYLINDERS
        mov     ax, [ebx]
        ;;SUB   AX,2                    ; ADJUST FOR 0-N
        dec     ax                      ; max. cylinder number
        MOV     CH,AL
        AND     AX,0300H                ; HIGH TWO BITS OF CYLINDER
        SHR     AX,1
        SHR     AX,1
        ;OR     AL,[ES:BX+14]           ; SECTORS
        or      al, [ebx+14]
        MOV     CL,AL
        ;MOV    DH,[ES:BX+2]            ; HEADS
        mov     dh, [ebx+2]
        DEC     DH                      ; 0-N RANGE
        MOV     DL,[HF_NUM]             ; DRIVE COUNT
        SUB     AX,AX
        ;27/12/2014
        ; ES:DI = Address of disk parameter table from BIOS
        ;(Programmer's Guide to the AMIBIOS - 1993)
        ;mov    di, bx                  ; HDPT offset
        mov     edi, ebx
G5:
        POP     eBX                     ; RESTORE REGISTERS
        ;POP    ES
        POP     DS
        ;RETf   2
        retf    4
G4:
        MOV     byte [DISK_STATUS1],INIT_FAIL ; OPERATION FAILED
        MOV     AH,INIT_FAIL
        SUB     AL,AL
        SUB     DX,DX
        SUB     CX,CX
        STC                             ; SET ERROR FLAG
        JMP     short G5

;---------------------------------------
;       INITIALIZE DRIVE     (AH = 09H) :
;---------------------------------------
        ; 03/01/2015
        ; According to ATA-ATAPI specification v2.0 to v5.0
        ; logical sector per logical track
        ; and logical heads - 1 would be set but
        ; it is seen as it will be good
        ; if physical parameters will be set here
        ; because, number of heads <= 16.
        ; (logical heads usually more than 16)
        ; NOTE: ATA logical parameters (software C, H, S)
        ;       == INT 13h physical parameters
```

```
;INIT_DRV:
;       MOV     byte [CMD_BLOCK+6],SET_PARM_CMD
;       CALL    GET_VEC                 ; ES:BX -> PARAMETER BLOCK
;       MOV     AL,[ES:BX+2]            ; GET NUMBER OF HEADS
;       DEC     AL                      ; CONVERT TO 0-INDEX
;       MOV     AH,[CMD_BLOCK+5]        ; GET SDH REGISTER
;       AND     AH,0F0H                 ; CHANGE HEAD NUMBER
;       OR      AH,AL                   ; TO MAX HEAD
;       MOV     [CMD_BLOCK+5],AH
;       MOV     AL,[ES:BX+14]           ; MAX SECTOR NUMBER
;       MOV     [CMD_BLOCK+1],AL
;       SUB     AX,AX
;       MOV     [CMD_BLOCK+3],AL        ; ZERO FLAGS
;       CALL    COMMAND                 ; TELL CONTROLLER
;       JNZ     short INIT_EXIT         ; CONTROLLER BUSY ERROR
;       CALL    NOT_BUSY                ; WAIT FOR IT TO BE DONE
;       JNZ     short INIT_EXIT         ; TIME OUT
;       CALL    CHECK_STATUS
;INIT_EXIT:
;       RETn

; 04/01/2015
; 02/01/2015 - Derived from from AWARD BIOS 1999
;                                AHDSK.ASM - INIT_DRIVE
INIT_DRV:
        ;xor    ah,ah
        xor     eax, eax ; 21/02/2015
        mov     al,11 ; Physical heads from translated HDPT
        cmp     [LBAMode], ah   ; 0
        ja      short idrv0
        mov     al,2  ; Physical heads from standard HDPT
idrv0:
        ; DL = drive number (0 based)
        call    GET_VEC
        ;push   bx
        push    ebx ; 21/02/2015
        ;add    bx,ax
        add     ebx, eax
        ;; 05/01/2015
        mov     ah, [hf_m_s] ; drive number (0= master, 1= slave)
        ;;and   ah,1
        shl     ah,4
        or      ah,0A0h  ; Drive/Head register - 10100000b (A0h)
        ;mov    al,[es:bx]
        mov     al, [ebx] ; 21/02/2015
        dec     al       ; last head number
        ;and    al,0Fh
        or      al,ah    ; lower 4 bits for head number
        ;
        mov     byte [CMD_BLOCK+6],SET_PARM_CMD
        mov     [CMD_BLOCK+5],al
        ;pop    bx
        pop     ebx
        sub     eax, eax ; 21/02/2015
        mov     al,4 ; Physical sec per track from translated HDPT
        cmp     byte [LBAMode], 0
        ja      short idrv1
        mov     al,14 ; Physical sec per track from standard HDPT
idrv1:
        ;xor    ah,ah
        ;add    bx,ax
        add     ebx, eax ; 21/02/2015
        ;mov    al,[es:bx]
```

```
                        ; sector number
        mov     al, [ebx]
        mov     [CMD_BLOCK+1],al
        sub     al,al
        mov     [CMD_BLOCK+3],al  ; ZERO FLAGS
        call    COMMAND           ; TELL CONTROLLER
        jnz     short INIT_EXIT   ; CONTROLLER BUSY ERROR
        call    NOT_BUSY          ; WAIT FOR IT TO BE DONE
        jnz     short INIT_EXIT   ; TIME OUT
        call    CHECK_STATUS
INIT_EXIT:
        RETn

;--------------------------------------
;       READ LONG            (AH = 0AH) :
;--------------------------------------

RD_LONG:
        ;MOV    @CMD_BLOCK+6,READ_CMD OR ECC_MODE
        mov     byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
        JMP     COMMANDI

;--------------------------------------
;       WRITE LONG           (AH = 0BH) :
;--------------------------------------

WR_LONG:
        ;MOV    @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
        MOV     byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
        JMP     COMMANDO

;--------------------------------------
;       SEEK                 (AH = 0CH) :
;--------------------------------------

DISK_SEEK:
        MOV     byte [CMD_BLOCK+6],SEEK_CMD
        CALL    COMMAND
        JNZ     short DS_EXIT           ; CONTROLLER BUSY ERROR
        CALL    _WAIT
        JNZ     DS_EXIT                 ; TIME OUT ON SEEK
        CALL    CHECK_STATUS
        CMP     byte [DISK_STATUS1],BAD_SEEK
        JNE     short DS_EXIT
        MOV     byte [DISK_STATUS1],0
DS_EXIT:
        RETn

;--------------------------------------
;       TEST DISK READY      (AH = 10H) :
;--------------------------------------

TST_RDY:                                ; WAIT FOR CONTROLLER
        CALL    NOT_BUSY
        JNZ     short TR_EX
        MOV     AL,[CMD_BLOCK+5]        ; SELECT DRIVE
        MOV     DX,[HF_PORT]
        add     dl,6
        OUT     DX,AL
        CALL    CHECK_ST                ; CHECK STATUS ONLY
        JNZ     short TR_EX
        MOV     byte [DISK_STATUS1],0   ; WIPE OUT DATA CORRECTED ERROR
TR_EX:
        RETn
```

```
;---------------------------------------
;       RECALIBRATE          (AH = 11H) :
;---------------------------------------

HDISK_RECAL:
        MOV     byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
        CALL    COMMAND                 ; START THE OPERATION
        JNZ     short RECAL_EXIT        ; ERROR
        CALL    _WAIT                   ; WAIT FOR COMPLETION
        JZ      short RECAL_X           ; TIME OUT ONE OK ?
        CALL    _WAIT                   ; WAIT FOR COMPLETION LONGER
        JNZ     short RECAL_EXIT        ; TIME OUT TWO TIMES IS ERROR
RECAL_X:
        CALL    CHECK_STATUS
        CMP     byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
        JNE     short RECAL_EXIT        ; IS OK
        MOV     byte [DISK_STATUS1],0
RECAL_EXIT:
        CMP     byte [DISK_STATUS1],0
        RETn

;---------------------------------------
;      CONTROLLER DIAGNOSTIC (AH = 14H) :
;---------------------------------------

CTLR_DIAGNOSTIC:
        CLI                             ; DISABLE INTERRUPTS WHILE CHANGING MASK
        IN      AL,INTB01               ; TURN ON SECOND INTERRUPT CHIP
        ;AND    AL,0BFH
        and     al, 3Fh                 ; enable IRQ 14 & IRQ 15
        ;JMP    $+2
        IODELAY
        OUT     INTB01,AL
        IODELAY
        IN      AL,INTA01               ; LET INTERRUPTS PASS THRU TO
        AND     AL,0FBH                 ;  SECOND CHIP
        ;JMP    $+2
        IODELAY
        OUT     INTA01,AL
        STI
        CALL    NOT_BUSY                ; WAIT FOR CARD
        JNZ     short CD_ERR            ; BAD CARD
        ;MOV    DX, HF_PORT+7
        mov     dx, [HF_PORT]
        add     dl, 7
        MOV     AL,DIAG_CMD             ; START DIAGNOSE
        OUT     DX,AL
        CALL    NOT_BUSY                ; WAIT FOR IT TO COMPLETE
        MOV     AH,TIME_OUT
        JNZ     short CD_EXIT           ; TIME OUT ON DIAGNOSTIC
        ;MOV    DX,HF_PORT+1            ; GET ERROR REGISTER
        mov     dx, [HF_PORT]
        inc     dl
        IN      AL,DX
        MOV     [HF_ERROR],AL           ; SAVE IT
        MOV     AH,0
        CMP     AL,1                    ; CHECK FOR ALL OK
        JE      SHORT CD_EXIT
CD_ERR: MOV     AH,BAD_CNTLR
CD_EXIT:
        MOV     [DISK_STATUS1],AH
        RETn
```

```
;---------------------------------------
; COMMANDI                              :
;       REPEATEDLY INPUTS DATA TILL     :
;       NSECTOR RETURNS ZERO            :
;---------------------------------------
COMMANDI:
        CALL    CHECK_DMA               ; CHECK 64K BOUNDARY ERROR
        JC      short CMD_ABORT
        ;MOV    DI,BX
        mov     edi, ebx ; 21/02/2015
        CALL    COMMAND                 ; OUTPUT COMMAND
        JNZ     short CMD_ABORT
CMD_I1:
        CALL    _WAIT                   ; WAIT FOR DATA REQUEST INTERRUPT
        JNZ     short TM_OUT            ; TIME OUT
        ;MOV    CX,256                  ; SECTOR SIZE IN WORDS
        mov     ecx, 256 ; 21/02/2015
        ;MOV    DX,HF_PORT
        mov     dx,[HF_PORT]
        CLI
        CLD
        REP     INSW                    ; GET THE SECTOR
        STI
        TEST    byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL INPUT
        JZ      CMD_I3
        CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
        JC      short TM_OUT
        ;MOV    DX,HF_PORT
        mov     dx,[HF_PORT]
        ;MOV    CX,4                    ; GET ECC BYTES
        mov     ecx, 4 ; mov cx, 4
CMD_I2: IN      AL,DX
        ;MOV    [ES:DI],AL              ; GO SLOW FOR BOARD
        mov     [edi], al ; 21/02/2015
        INC     eDI
        LOOP    CMD_I2
CMD_I3: CALL    CHECK_STATUS
        JNZ     short CMD_ABORT         ; ERROR RETURNED
        DEC     byte [CMD_BLOCK+1]      ; CHECK FOR MORE
        JNZ     SHORT CMD_I1
CMD_ABORT:
TM_OUT: RETn

;---------------------------------------
; COMMANDO                              :
;       REPEATEDLY OUTPUTS DATA TILL    :
;       NSECTOR RETURNS ZERO            :
;---------------------------------------
COMMANDO:
        CALL    CHECK_DMA               ; CHECK 64K BOUNDARY ERROR
        JC      short CMD_ABORT
CMD_OF: MOV     eSI,eBX ; 21/02/2015
        CALL    COMMAND                 ; OUTPUT COMMAND
        JNZ     short CMD_ABORT
        CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
        JC      short TM_OUT                    ; TOO LONG
CMD_O1: ;PUSH   DS
        ;PUSH   ES                      ; MOVE ES TO DS
        ;POP    DS
        ;MOV    CX,256                  ; PUT THE DATA OUT TO THE CARD
        ;MOV    DX,HF_PORT
        ; 01/02/2015
        mov     dx, [HF_PORT]
        ;push   es
```

```
        ;pop    ds
        ;mov    cx, 256
        mov     ecx, 256 ; 21/02/2015
        CLI
        CLD
        REP     OUTSW
        STI
        ;POP    DS                      ; RESTORE DS
        TEST    byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
        JZ      short CMD_O3
        CALL    WAIT_DRQ                ; WAIT FOR DATA REQUEST
        JC      short TM_OUT
        ;MOV    DX,HF_PORT
        mov     dx, [HF_PORT]
        ;MOV    CX,4                    ; OUTPUT THE ECC BYTES
        mov     ecx, 4  ; mov cx, 4
CMD_O2: ;MOV    AL,[ES:SI]
        mov     al, [esi]
        OUT     DX,AL
        INC     eSI
        LOOP    CMD_O2
CMD_O3:
        CALL    _WAIT                   ; WAIT FOR SECTOR COMPLETE INTERRUPT
        JNZ     short TM_OUT            ; ERROR RETURNED
        CALL    CHECK_STATUS
        JNZ     short CMD_ABORT
        TEST    byte [HF_STATUS],ST_DRQ ; CHECK FOR MORE
        JNZ     SHORT CMD_O1
        ;MOV    DX,HF_PORT+2            ; CHECK RESIDUAL SECTOR COUNT
        mov     dx, [HF_PORT]
        ;add    dl, 2
        inc     dl
        inc     dl
        IN      AL,DX                   ;
        TEST    AL,0FFH                 ;
        JZ      short CMD_O4                    ; COUNT = 0  OK
        MOV     byte [DISK_STATUS1],UNDEF_ERR
                                        ; OPERATION ABORTED - PARTIAL TRANSFER
CMD_O4:
        RETn

;--------------------------------------------------------
; COMMAND                                               :
;       THIS ROUTINE OUTPUTS THE COMMAND BLOCK          :
; OUTPUT                                                :
;       BL = STATUS                                     :
;       BH = ERROR REGISTER                             :
;--------------------------------------------------------

COMMAND:
        PUSH    eBX                     ; WAIT FOR SEEK COMPLETE AND READY
        ;;MOV   CX,DELAY_2              ; SET INITIAL DELAY BEFORE TEST
COMMAND1:
        ;;PUSH  CX                      ; SAVE LOOP COUNT
        CALL    TST_RDY                 ; CHECK DRIVE READY
        ;;POP   CX
        JZ      short COMMAND2          ; DRIVE IS READY
        CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
        ;JZ     short CMD_TIMEOUT
        ;;LOOP  COMMAND1                ; KEEP TRYING FOR A WHILE
        ;JMP    SHORT COMMAND4          ; ITS NOT GOING TO GET READY
        jne     short COMMAND4
CMD_TIMEOUT:
        MOV     byte [DISK_STATUS1],BAD_CNTLR
```

```
COMMAND4:
        POP     eBX
        CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
        RETn
COMMAND2:
        POP     eBX
        PUSH    eDI
        MOV     byte [HF_INT_FLAG],0    ; RESET INTERRUPT FLAG
        CLI                             ; INHIBIT INTERRUPTS WHILE CHANGING MASK
        IN      AL,INTB01               ; TURN ON SECOND INTERRUPT CHIP
        ;AND    AL,0BFH
        and     al, 3Fh                 ; Enable IRQ 14 & 15
        ;JMP    $+2
        IODELAY
        OUT     INTB01,AL
        IN      AL,INTA01               ; LET INTERRUPTS PASS THRU TO
        AND     AL,0FBH                 ;  SECOND CHIP
        ;JMP    $+2
        IODELAY
        OUT     INTA01,AL
        STI
        XOR     eDI,eDI                 ; INDEX THE COMMAND TABLE
        ;MOV    DX,HF_PORT+1            ; DISK ADDRESS
        mov     dx, [HF_PORT]
        inc     dl
        TEST    byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
        JZ      short COMMAND3
        MOV     AL, [CMD_BLOCK+6]       ; YES-GET OPERATION CODE
        AND     AL,0F0H                 ; GET RID OF MODIFIERS
        CMP     AL,20H                  ; 20H-40H IS READ, WRITE, VERIFY
        JB      short COMMAND3
        CMP     AL,40H
        JA      short COMMAND3
        OR      byte [CMD_BLOCK+6],NO_RETRIES
                                        ; VALID OPERATION FOR RETRY SUPPRESS
COMMAND3:
        MOV     AL,[CMD_BLOCK+eDI]      ; GET THE COMMAND STRING BYTE
        OUT     DX,AL                   ; GIVE IT TO CONTROLLER
        IODELAY
        INC     eDI                     ; NEXT BYTE IN COMMAND BLOCK
        INC     DX                      ; NEXT DISK ADAPTER REGISTER
        cmp     di, 7   ; 1/1/2015      ; ALL DONE?
        JNZ     short COMMAND3          ; NO--GO DO NEXT ONE
        POP     eDI
        RETn                            ; ZERO FLAG IS SET

;CMD_TIMEOUT:
;       MOV     byte [DISK_STATUS1],BAD_CNTLR
;COMMAND4:
;       POP     BX
;       CMP     [DISK_STATUS1],0        ; SET CONDITION CODE FOR CALLER
;       RETn

;---------------------------------------
;       WAIT FOR INTERRUPT              :
;---------------------------------------
;WAIT:
_WAIT:
        STI                             ; MAKE SURE INTERRUPTS ARE ON
        ;SUB    CX,CX                   ; SET INITIAL DELAY BEFORE TEST
        ;CLC
        ;MOV    AX,9000H                ; DEVICE WAIT INTERRUPT
        ;INT    15H
        ;JC     WT2                     ; DEVICE TIMED OUT
```

```
        ;MOV    BL,DELAY_1              ; SET DELAY COUNT

        ;mov    bl, WAIT_HDU_INT_HI
        ;; 21/02/2015
        ;;mov   bl, WAIT_HDU_INT_HI + 1
        ;;mov   cx, WAIT_HDU_INT_LO
        mov     ecx, WAIT_HDU_INT_LH
                                        ; (AWARD BIOS -> WAIT_FOR_MEM)
;-----  WAIT LOOP

WT1:
        ;TEST   byte [HF_INT_FLAG],80H  ; TEST FOR INTERRUPT
        test    byte [HF_INT_FLAG],0C0h
        ;LOOPZ  WT1
        JNZ     short WT3               ; INTERRUPT--LETS GO
        ;DEC    BL
        ;JNZ    short WT1               ; KEEP TRYING FOR A WHILE

WT1_hi:
        in      al, SYS1 ; 61h (PORT_B) ; wait for lo to hi
        test    al, 10h                 ; transition on memory
        jnz     short WT1_hi            ; refresh.
WT1_lo:
        in      al, SYS1                ; 061h (PORT_B)
        test    al, 10h
        jz      short WT1_lo
        loop    WT1
        ;;or    bl, bl
        ;;jz    short WT2
        ;;dec   bl
        ;;jmp   short WT1
        ;dec    bl
        ;jnz    short WT1

WT2:    MOV     byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
        JMP     SHORT WT4
WT3:    MOV     byte [DISK_STATUS1],0
        MOV     byte [HF_INT_FLAG],0
WT4:    CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
        RETn

;--------------------------------------
;       WAIT FOR CONTROLLER NOT BUSY    :
;--------------------------------------
NOT_BUSY:
        STI                             ; MAKE SURE INTERRUPTS ARE ON
        ;PUSH   eBX
        ;SUB    CX,CX                   ; SET INITIAL DELAY BEFORE TEST
        mov     DX, [HF_PORT]
        add     dl, 7                   ; Status port (HF_PORT+7)
        ;MOV    BL,DELAY_1
                                        ; wait for 10 seconds
        ;mov    cx, WAIT_HDU_INT_LO     ; 1615h
        ;;mov   bl, WAIT_HDU_INT_HI     ;   05h
        ;mov    bl, WAIT_HDU_INT_HI + 1
        mov     ecx, WAIT_HDU_INT_LH  ; 21/02/2015
        ;
;;      mov     byte [wait_count], 0    ; Reset wait counter
NB1:
        IN      AL,DX                   ; CHECK STATUS
        ;TEST   AL,ST_BUSY
        and     al, ST_BUSY
        ;LOOPNZ NB1
        JZ      short NB2               ; NOT BUSY--LETS GO
```

```
        ;DEC    BL
        ;JNZ    short NB1               ; KEEP TRYING FOR A WHILE

NB1_hi: IN      AL,SYS1                 ; wait for hi to lo
        TEST    AL,010H                 ; transition on memory
        JNZ     SHORT NB1_hi            ; refresh.
NB1_lo: IN      AL,SYS1
        TEST    AL,010H
        JZ      short NB1_lo
        LOOP    NB1
        ;dec    bl
        ;jnz    short NB1
        ;
;;      cmp     byte [wait_count], 182  ; 10 seconds (182 timer ticks)
;;      jb      short NB1
        ;
        ;MOV    [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
        ;JMP    SHORT NB3
        mov     al, TIME_OUT
NB2:
        ;MOV    byte [DISK_STATUS1],0
;NB3:
        ;POP    eBX
        mov     [DISK_STATUS1], al      ;;; will be set after return
        ;CMP    byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
        or      al, al                  ; (zf = 0 --> timeout)
        RETn

;---------------------------------------
;       WAIT FOR DATA REQUEST           :
;---------------------------------------
WAIT_DRQ:
        ;MOV    CX,DELAY_3
        ;MOV    DX,HF_PORT+7
        mov     dx, [HF_PORT]
        add     dl, 7
        ;;MOV   bl, WAIT_HDU_DRQ_HI     ; 0
        ;MOV    cx, WAIT_HDU_DRQ_LO     ; 1000 (30 milli seconds)
                                        ; (but it is written as 2000
                                        ; micro seconds in ATORGS.ASM file
                                        ; of Award Bios - 1999, D1A0622)
        mov     ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
WQ_1:   IN      AL,DX                   ; GET STATUS
        TEST    AL,ST_DRQ               ; WAIT FOR DRQ
        JNZ     short WQ_OK
        ;LOOP   WQ_1                    ; KEEP TRYING FOR A SHORT WHILE
WQ_hi:
        IN      AL,SYS1                 ; wait for hi to lo
        TEST    AL,010H                 ; transition on memory
        JNZ     SHORT WQ_hi             ; refresh.
WQ_lo:  IN      AL,SYS1
        TEST    AL,010H
        JZ      SHORT WQ_lo
        LOOP    WQ_1

        MOV     byte [DISK_STATUS1],TIME_OUT ; ERROR
        STC
WQ_OK:
        RETn
;WQ_OK: ;CLC
;       RETn

;---------------------------------------
;       CHECK FIXED DISK STATUS         :
```

```
;----------------------------------------
CHECK_STATUS:
        CALL    CHECK_ST                ; CHECK THE STATUS BYTE
        JNZ     short CHECK_S1          ; AN ERROR WAS FOUND
        TEST    AL,ST_ERROR             ; WERE THERE ANY OTHER ERRORS
        JZ      short CHECK_S1          ; NO ERROR REPORTED
        CALL    CHECK_ER                ; ERROR REPORTED
CHECK_S1:
        CMP     byte [DISK_STATUS1],0   ; SET STATUS FOR CALLER
        RETn

;----------------------------------------
;       CHECK FIXED DISK STATUS BYTE    :
;----------------------------------------
CHECK_ST:
        ;MOV    DX,HF_PORT+7            ; GET THE STATUS
        mov     dx, [HF_PORT]
        add     dl, 7
        IN      AL,DX
        MOV     [HF_STATUS],AL
        MOV     AH,0
        TEST    AL,ST_BUSY              ; IF STILL BUSY
        JNZ     short CKST_EXIT         ;  REPORT OK
        MOV     AH,WRITE_FAULT
        TEST    AL,ST_WRT_FLT           ; CHECK FOR WRITE FAULT
        JNZ     short CKST_EXIT
        MOV     AH,NOT_RDY
        TEST    AL,ST_READY             ; CHECK FOR NOT READY
        JZ      short CKST_EXIT
        MOV     AH,BAD_SEEK
        TEST    AL,ST_SEEK_COMPL        ; CHECK FOR SEEK NOT COMPLETE
        JZ      short CKST_EXIT
        MOV     AH,DATA_CORRECTED
        TEST    AL,ST_CORRCTD           ; CHECK FOR CORRECTED ECC
        JNZ     short CKST_EXIT
        MOV     AH,0
CKST_EXIT:
        MOV     [DISK_STATUS1],AH       ; SET ERROR FLAG
        CMP     AH,DATA_CORRECTED       ; KEEP GOING WITH DATA CORRECTED
        JZ      short CKST_EX1
        CMP     AH,0
CKST_EX1:
        RETn

;----------------------------------------
;       CHECK FIXED DISK ERROR REGISTER :
;----------------------------------------
CHECK_ER:
        ;MOV    DX, HF_PORT+1           ; GET THE ERROR REGISTER
        mov     dx, [HF_PORT]           ;
        inc     dl
        IN      AL,DX
        MOV     [HF_ERROR],AL
        PUSH    eBX ; 21/02/2015
        MOV     eCX,8                   ; TEST ALL 8 BITS
CK1:    SHL     AL,1                    ; MOVE NEXT ERROR BIT TO CARRY
        JC      short CK2               ; FOUND THE ERROR
        LOOP    CK1                     ; KEEP TRYING
CK2:    MOV     eBX, ERR_TBL            ; COMPUTE ADDRESS OF
        ADD     eBX,eCX                 ; ERROR CODE
        ;;MOV   AH,BYTE [CS:BX]         ; GET ERROR CODE
        ;mov    ah, byte [bx]
        mov     ah, byte [ebx] ; 21/02/2015
CKEX:   MOV     [DISK_STATUS1],AH       ; SAVE ERROR CODE
```

```
        POP     eBX
        CMP     AH,0
        RETn
ERR_TBL:
        DB      NO_ERR
        DB      BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
        DB      RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR

;--------------------------------------------------------
; CHECK_DMA                                             :
;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
;   FIT WITHOUT SEGMENT OVERFLOW.                        :
;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X       :
;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)    :
;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H)      :
;  -ERROR OTHERWISE                                      :
;--------------------------------------------------------
CHECK_DMA:
        PUSH    AX                      ; SAVE REGISTERS
        MOV     AX,8000H                ; AH = MAX # SECTORS AL = MAX OFFSET
        TEST    byte [CMD_BLOCK+6],ECC_MODE
        JZ      short CKD1
        MOV     AX,7F04H                ; ECC IS 4 MORE BYTES
CKD1:   CMP     AH, [CMD_BLOCK+1]       ; NUMBER OF SECTORS
        JA      short CKDOK             ; IT WILL FIT
        JB      short CKDERR            ; TOO MANY
        CMP     AL,BL                   ; CHECK OFFSET ON MAX SECTORS
        JB      short CKDERR            ; ERROR
CKDOK:  CLC                             ; CLEAR CARRY
        POP     AX
        RETn                            ; NORMAL RETURN
CKDERR: STC                             ; INDICATE ERROR
        MOV     byte [DISK_STATUS1],DMA_BOUNDARY
        POP     AX
        RETn

;----------------------------------------
;       SET UP ES:BX-> DISK PARMS       :
;----------------------------------------

; INPUT -> DL = 0 based drive number
; OUTPUT -> ES:BX = disk parameter table address

GET_VEC:
        ;SUB    AX,AX                   ; GET DISK PARAMETER ADDRESS
        ;MOV    ES,AX
        ;TEST   DL,1
        ;JZ     short GV_0
;       LES     BX,[HF1_TBL_VEC]        ; ES:BX -> DRIVE PARAMETERS
;       JMP     SHORT GV_EXIT
;GV_0:
;       LES     BX,[HF_TBL_VEC]         ; ES:BX -> DRIVE PARAMETERS
;
        ;xor    bh, bh
        xor     ebx, ebx
        mov     bl, dl
        ;;02/01/2015
        ;;shl   bl, 1                   ; port address offset
        ;;mov   ax, [bx+hd_ports]       ; Base port address (1F0h, 170h)
        ;;shl   bl, 1                   ; dpt pointer offset
        shl     bl, 2   ;;
        ;add    bx, HF_TBL_VEC          ; Disk parameter table pointer
        add     ebx, HF_TBL_VEC ; 21/02/2015
        ;push   word [bx+2]             ; dpt segment
```

```
        ;pop    es
        ;mov    bx, [bx]                ; dpt offset
        mov     ebx, [ebx]
;GV_EXIT:
        RETn

hdc1_int: ; 21/02/2015
;--- HARDWARE INT 76H -- ( IRQ LEVEL  14 ) ----------------------
;                                                                :
;       FIXED DISK INTERRUPT ROUTINE                             :
;                                                                :
;-----------------------------------------------------------------

; 22/12/2014
; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
;        '11/15/85'
; AWARD BIOS 1999 (D1A0622)
;       Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)

;int_76h:
HD_INT:
        PUSH    AX
        PUSH    DS
        ;CALL   DDS
        ; 21/02/2015 (32 bit, 386 pm modification)
        mov     ax, KDATA
        mov     ds, ax
        ;
        ;;MOV   @HF_INT_FLAG,0FFH       ; ALL DONE
        ;mov     byte [CS:HF_INT_FLAG], 0FFh
        mov     byte [HF_INT_FLAG], 0FFh
        ;
        push    dx
        mov     dx, HDC1_BASEPORT+7     ; Status Register (1F7h)
                                        ; Clear Controller
Clear_IRQ1415:                          ; (Award BIOS - 1999)
        in      al, dx                  ;
        pop     dx
        NEWIODELAY
        ;
        MOV     AL,EOI                  ; NON-SPECIFIC END OF INTERRUPT
        OUT     INTB00,AL               ; FOR CONTROLLER #2
        ;JMP    $+2                     ; WAIT
        NEWIODELAY
        OUT     INTA00,AL               ; FOR CONTROLLER #1
        POP     DS
        ;STI                            ; RE-ENABLE INTERRUPTS
        ;MOV    AX,9100H                ; DEVICE POST
        ;INT    15H                     ;  INTERRUPT
irq15_iret: ; 25/02/2015
        POP     AX
        IRETd                           ; RETURN FROM INTERRUPT

hdc2_int: ; 21/02/2015
;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) ++++++++++++++++++++
;                                                                :
;       FIXED DISK INTERRUPT ROUTINE                             :
;                                                                :
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;int_77h:
HD1_INT:
        PUSH    AX
        ; Check if that is a spurious IRQ (from slave PIC)
```

```
        ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
        mov     al, 0Bh  ; In-Service Register
        out     0A0h, al
        jmp short $+2
        jmp short $+2
        in      al, 0A0h
        and     al, 80h ; bit 7 (is it real IRQ 15 or fake?)
        jz      short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
        ;
        PUSH    DS
        ;CALL   DDS
        ; 21/02/2015 (32 bit, 386 pm modification)
        mov     ax, KDATA
        mov     ds, ax
        ;
        ;;MOV   @HF_INT_FLAG,0FFH       ; ALL DONE
        ;or      byte [CS:HF_INT_FLAG],0C0h
        or      byte [HF_INT_FLAG], 0C0h
        ;
        push    dx
        mov     dx, HDC2_BASEPORT+7     ; Status Register (177h)
                                        ; Clear Controller (Award BIOS 1999)
        jmp     short Clear_IRQ1415


;////////////////////////////////////////////////////////////////
;; END OF DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


; Beginning of DSECTPM (Display Disk Sectors in Protected Mode) code
;; ////////////////////////////////////////////////////////////
;, 12/02/2015
;; Display Disk Sectors in protected mode (12/02/2015, unix386.s)
;; Retro UNIX 386 v1 - DISK I/O Test


; display disk sector data [ Retro Unix 386 v1 - test ]
; modified from 'DSECTRM2.S' (02/02/2015)
; by Erdogan Tan
;

;; 22/02/2015
;, Enable/Setup Real Time Clock Interrupt (dsectpm.s)
;;
setup_rtc_int:
; source: http://wiki.osdev.org/RTC
        cli             ; disable interrupts
        ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b
or 6)
        ; in order to change this ...
        ; frequency  = 32768 >> (rate-1) --> 32768 >> 5 = 1024
        ; (rate must be above 2 and not over 15)
        ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
        mov     al, 8Ah
        out     70h, al ; set index to register A, disable NMI
        nop
        in      al, 71h ; get initial value of register A
        mov     ah, al
        and     ah, 0F0h
        mov     al, 8Ah
        out     70h, al ; reset index to register A
        mov     al, ah
        or      al, 0Fh ; new rate (0Fh -> 15)
        out     71h, al ; write only our rate to A. Note, rate is the bottom 4
```

```
bits.
        ; enable RTC interrupt
        mov     al, 8Bh ;
        out     70h, al ; select register B and disable NMI
        nop
        in      al, 71h ; read the current value of register B
        mov     ah, al  ;
        mov     al, 8Bh ;
        out     70h, al ; set the index again (a read will reset the index to
register B)
        mov     al, ah  ;
        or      al, 40h ;
        out     71h, al ; write the previous value ORed with 0x40. This turns on
bit 6 of register B
        sti
        retn

; 25/11/2014

ESCKey   equ 1Bh    ;27
ENTERKey equ 0Dh    ;13
SPACEKey equ 20h    ;32
BACKSPC  equ 08h    ; 8
DELKey   equ 53E0h
F1Key    equ 3B00h
F2Key    equ 3C00h
F3Key    equ 3D00h ; 19/12/2014
HOMEKey  equ 47E0h
ENDKey   equ 4FE0h
PgUpKey  equ 49E0h
PgDnKey  equ 51E0h

; 30/12/2014 (0B000h -> 9000h)
; 10/12/2014
;DPT_SEGM equ 9000h  ; FDPT segment (EDD v1.1, EDD v3)
;
;HD0_DPT  equ 0     ; Disk parameter table address for hd0
;HD1_DPT  equ 32    ; Disk parameter table address for hd1
;HD2_DPT  equ 64    ; Disk parameter table address for hd2
;HD3_DPT  equ 96    ; Disk parameter table address for hd3

; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
;      (HDPT: Programmer's Guide to the AMIBIOS, 1993)
;
;FDPT_CYLS equ 0  ; 1 word, number of cylinders
;FDPT_HDS  equ 2  ; 1 byte, number of heads
;FDPT_TT          equ 3  ; 1 byte, A0h = translated FDPT with logical values
                 ; otherwise it is standard FDPT with physical values
;FDPT_PCMP equ 5  ; 1 word, starting write precompensation cylinder
                 ; (obsolete for IDE/ATA drives)
;FDPT_CB   equ 8  ; 1 byte, drive control byte
                  ; Bits 7-6 : Enable or disable retries (00h = enable)
                  ; Bit 5    : 1 = Defect map is located at last cyl. + 1
                  ; Bit 4 : Reserved. Always 0
                  ; Bit 3 : Set to 1 if more than 8 heads
                  ; Bit 2-0 : Reserved. Alsways 0
;FDPT_LZ   equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
;FDPT_SPT  equ 14 ; 1 byte, sectors per track

dsectpm: ; 16/02/2015
        ;mov    word [ttychr], 0
display_sectors:
        ; 20/02/2015
        ; 12/02/2015 (protected mode modification, unix386.s)
```

```
        ; 26/11/2014 - 04/12/2014 (dsectrm2.s)
        ;
        call    hide_cursor
        ; Save video page (before displaying sector)

        mov     esi, 0B8000h
        mov     edi, esi
        add     edi, 5DC0h  ; 4000*6 (video page 6)
        mov     ecx, 1000
        rep     movsd

        ; Save cursor position
        ;mov    ax, [cursor_posn] ; cursor pos.
                                  ; for video page 0.
        ;mov    [cursor_posb], ax
        call    clear_frame
        jmp     short dscl_0
dscl_esc:
        call    restore_video_page
dscl_getc:
        call    getch
        ;
        cmp     al, ESCKey
        je      dscl_exit
        mov     byte [dscmd], 0
        cmp     ax, F1Key
        je      short dscl_0
        ; 19/12/2014
        inc     byte [dscmd]
        cmp     ax, F3Key
        jne     dscl_5
        inc     byte [dscmd] ; Display disk params.
dscl_0:
        call    save_video_page
        mov     esi, F1_ib ; F1 (Change drive)
                           ; Inputbox address
dscl_ib:
        call    inputbox
                ; cursor position in DX
        call    show_cursor
                ; cursor blinks at current position
dscl_3:
        call    getch
        cmp     al, ESCKey
        jne     short dscl_27
        call    hide_cursor
        jmp     short dscl_esc
dscl_27:
        cmp     al, SPACEKey
        je      short dscl_4
        cmp     al, ENTERKey
        je      short dscl_4
        ;
        xor     ebx, ebx ; 20/02/2015
        cmp     byte [dscmd], 1
        je      short dscl_12
        ;
        cmp     al, '0'
        jb      short dscl_3
        cmp     al, '5'
        ja      short dscl_3
        mov     edi, [current_txtpos]
        stosb
        sub     al, '0'
```

```
        mov     dl, al
        xor     dh, dh
        mov     bl, al
        shl     bl, 2  ; *4
        add     ebx, ds_sec ; current_sector
        mov     ecx, [ebx]
        mov     esi, buffer
        jmp     short dscl_3
dscl_4:
        cmp     byte [inds],  0 ; display other half or not ?
        ja      dscl_oh         ; other half
        push    ecx
        ; 17/02/2015
        ; save regs (ESI, EAX, DX)
        push    dx
        call    hide_cursor
        ; restore regs (ESI, EAX, DX)
        pop     dx
        pop     eax
        ; 19/12/2014
        cmp     byte [dscmd], 1 ; Requested function ?
        je      dscl_17         ; Change sector (F2)
        jb      dscl_ns         ; Change drive (F1)
        ; Display disk parameters (dscmd = 2)
        ; 31/12/2014
        cmp     dl, 2
        jb      short dscl_28
        add     dl, 7Eh
dscl_28:
        call    dskprm
        jmp     dscl_esc
dscl_12:
        cmp     ax, DELKey      ; DEL key
        je      short dscl_bs
        cmp     al, BACKSPC     ; Backspace key
        jne     short dscl_13
dscl_bs:
        cmp     byte [txtposoff], 0
        jna     dscl_3
        dec     byte [txtposoff]
        dec     byte [cursor_posn]
        ;mov    dx, [cursor_posn]
        call    set_cposn
        xor     ebx, ebx
        mov     bl, [txtposoff]
        dec     byte [txtposoff]
        dec     byte [cursor_posn]
        mov     al, 20h
        jmp     short dscl_14
dscl_13:
        mov     bl, [txtposoff]
        cmp     bl, 8
        jnb     dscl_3
        ;
        cmp     al, '0'
        jb      dscl_3
        cmp     al, '9'
        ja      short dscl_15
dscl_14:
        shl     bl, 1
        mov     esi, [current_txtpos]
        add     ebx, esi
        mov     [ebx], al
        ;
```

```
        cmp     byte [txtposoff], 8
        jge     dscl_3 ; JGE !
        inc     byte [txtposoff]
        inc     byte [cursor_posn]
        ;mov    dx, [cursor_posn]
        call    set_cposn
        jmp     dscl_3
dscl_15:
        cmp     al, 'A'
        jb      dscl_3
        cmp     al, 'F'
        jna     short dscl_14
dscl_16:
        cmp     al, 'a'
        jb      dscl_3
        cmp     al, 'f'
        ja      dscl_3
        sub     al, 'a' - 'A'
        jmp     short dscl_14
        ;
dscl_17:
        mov     esi, [current_txtpos]
        xor     eax, eax
        mov     byte [txtposoff], al ; 0
        push    eax  ; sector value (reset)
dscl_18:
        lodsw
        cmp     al, '0'
        jb      short dscl_22
dscl_19:
        sub     ecx, ecx
        mov     ebx, hexchrs
dscl_20:
        cmp     al, [ebx]
        je      short dscl_21
        ;cmp    cl, 15
        ;jnb    short dscl_22
        inc     cl
        inc     ebx
        jmp     short dscl_20
dscl_21:
        pop     eax
        shl     eax, 4  ; * 16
        add     eax, ecx
        push    eax
        jmp     short dscl_18
dscl_22:
        mov     dl, [ds_drv]
        xor     dh, dh
        pop     eax
        mov     esi, buffer
        jmp     short dscl_ns
dscl_oh:
        mov     dl, [ds_drv]
        xor     ebx, ebx
        mov     bl, dl
        shl     bl, 2
        add     ebx, ds_sec
        mov     eax, [ebx]
        mov     esi, buffer
        ;
        mov     dh, [ds_drv+1]
        or      dh, dh
        jz      short dscl_nh ; second half of sector (0->1)
```

```
        xor     dh, dh        ; reset (0)
        jmp     short dscl_nx
dscl_nh:
        add     esi, 256
        inc     dh
dscl_nx:
        mov     [ds_drv+1], dh
        jmp     dscl_25
dscl_ns:
        mov     [ds_drv+1], dh
        xor     ebx, ebx
        mov     bl, dl
        shl     bl, 2
        add     ebx, ds_sec
        cmp     dl, [ds_drv]
        jne     short dscl_23
        cmp     eax, [ebx]
        je      short dscl_25
dscl_23:
        mov     [ds_drv], dl
dscl_26:
        mov     ecx, [ebx]
        mov     [prev_sec], ecx
        mov     [ebx], eax
        call    read_disk_sector
        jnc     short dscl_24
dscl_rd_err:
        ;
        ;; Temporary, 15/12/2014
        mov     al, ah  ; error code
        mov     di, err_code_str
        call    write_hex
        ;
        mov     esi, dskr_err ; drive not ready or read error
        call    inputbox
        call    getch
        call    restore_video_page
        sub     ebx, ebx
        mov     bl, [ds_drv]
        shl     bl, 2
        add     ebx, ds_sec
        mov     eax, [prev_sec]
        mov     [ebx], eax
        jmp     dscl_getc
dscl_24:
        mov     dx, [ds_drv]
        xor     ebx, ebx
        mov     bl, dl
        shl     bl, 2
        add     ebx, ds_sec
        mov     eax, [ebx]
        mov     esi, buffer
dscl_25:
        call    display_sector
        call    save_video_page
        jmp     dscl_getc
dscl_11:
        mov     esi, buffer
        mov     dl, [ds_drv]
        sub     dh, dh   ; 0 = first half of sector
        jmp     dscl_ns
dscl_5:
        cmp     ax, F2Key
        jne     short dscl_6
```

```
        call    save_video_page
        mov     esi, F2_ib ; F2 (Change sector)
                           ; Inputbox address
        ;mov    byte [dscmd], 1
        jmp     dscl_ib
dscl_6:
        cmp     al, SPACEKey
        je      dscl_oh
        cmp     al, ENTERKey
        je      dscl_oh
        ;
        cmp     ax, HOMEKey
        jne     short dscl_7
        xor     eax, eax
        jmp     short dscl_11
dscl_7:
        cmp     ax, ENDKey
        jne     short dscl_8
        xor     ebx, ebx
        mov     bl, [ds_drv]
        shl     bl, 2
        add     ebx, disk_size
        mov     eax, [ebx]
        dec     eax
        jmp     short dscl_11
dscl_8:
        cmp     ax, PgDnKey
        jne     short dscl_10
        call    dscl_9
        inc     eax
        cmp     eax, ecx ; last sector
        jna     dscl_ns
        xor     eax, eax
        jmp     dscl_26
dscl_9:
        mov     dl, [ds_drv]
        xor     dh, dh
        sub     ebx, ebx
        mov     bl, dl
        shl     bl, 2  ; *4
        add     ebx, disk_size
        mov     ecx, [ebx]
        dec     ecx
        sub     ebx, disk_size
        add     ebx, ds_sec ; current sector
        mov     eax, [ebx]
        mov     esi, buffer
        retn
dscl_10:
        cmp     ax, PgUpKey
        jne     dscl_getc
        call    dscl_9
        dec     eax
        cmp     eax, ecx ; last sector
        jna     dscl_ns
        mov     eax, ecx
        jmp     dscl_26

dscl_exit:
        ; 12/02/2015 (unix386.s)
        ; 11/12/2014 (dsectrm2.s)
        ;
        ; Restore video page (before displaying sector)
        mov     edi, 0B8000h
```

```
        mov     esi, edi
        add     esi, 5DC0h  ; 4000*6 (video page 6)
        mov     ecx, 1000
        rep     movsd
        ; Restore cursor position
        mov     dx, [cursor_posb] ; cursor position backup
        mov     [cursor_posn], dx ; for video page 0.
        ;
        ; Set cursor position
        xor     bl, bl  ; Video page 0
        call    set_cpos
        ; Show standard blinking text cursor
        mov     cx, [cursor_shp]
        call    set_ctype
;terminate:
        ;
        ;jmp    cpu_reset
        ;
        retn
;hltsys:
;       hlt
;       jmp     short hltsys

getch:
        ; 22/02/2015 - dsectpm
        mov     ah, 10h ; enhanced keyboard, read function
        jmp     getc    ; jump to getchar
                        ; (getc_int, modified interrupt) routine
;getch:
        ; 18/02/2015 (unix386.s)
        ; This routine will be replaced with Retro UNIX 386
        ; version of Retro UNIX 8086 getch (tty input)
        ; routine, later... (multi tasking ability)
        ;push   esi
        ;push   ebx
        ;xor    ebx, ebx
        ;mov    bl, [ptty]  ; active_page
        ;mov    esi, ebx
        ;shl    si, 1
        ;add    esi, ttychr
;getch_1:
        ;mov    ax, word [esi]
;       mov     ax, [ttychr] ; video page 0 (tty0)
;       and     ax, ax
;       jz      short getch_2
;       mov     word [ttychr], 0
        ;mov    word [esi], 0
        ;pop    ebx
        ;pop    esi
;       retn
;getch_2:
        ;sti
;       hlt     ; not proper for multi tasking!
                ; (temporary halt for now)
                ; 'sleep' on tty
                ; will (must) be located here
;       nop
;       jmp     short getch_1

save_video_page:
        ; 12/02/2015 (unix386.s)
        ; 26/11/2014 (dsectrm2.s)
        ;
        ; Save video page
```

```
        mov     esi, 0B8000h
        mov     edi, esi
        add     edi, 6D60h  ; 4000*7 (video page 7)
        mov     ecx, 1000
        rep     movsd
        retn

restore_video_page:
        ; 12/02/2015 (unix386.s)
        ; 26/11/2014 (dsectrm2.s)
        mov     esi, 0B8000h
        mov     edi, esi
        add     esi, 6D60h  ; 4000*7 (video page 7)
        mov     ecx, 1000
        rep     movsd
        retn

display_sector:
        ; 19/02/2015
        ; 12/02/2015 (unix386.s)
        ; 6/11/2014 - 04/12/2014 (dsectrm2.s)
        ;
        ; display disk sector data (on tty0)
        ;
        ; INPUT ->
        ;       ESI = sector buffer offset
        ;             (sector size: 512 bytes)
        ;       EAX = sector number
        ;       DL = drive number (0,1,2,3,4,5,6)
        ;       DH = portion control byte
        ;                (0= first half of the sector,
        ;                >0= second half of the sector)
        ; OUTPUT ->
        ;       Video page 0 (0B8000h) will be filled
        ;       with sector data
        ;       (ESI points to byte 256 of the buffer
        ;       or end of the buffer)
        ;
        ; Modified registers: eax, edx, ecx, ebx, esi, edi
        ;
        ;
        ;xor    ecx, ecx ; reset for cx loop counts
        mov     byte [inds], 1 ; for ENTER key handling
        ;
        push    eax
        call    clear_frame
        pop     eax
dsfh:
        xor     ebx, ebx
        or      dh, dh
        jz      short dsfh1
        mov     bl, 10h
dsfh1:
        mov     [paragr], bl    ; Paragraph (16 bytes)
        ;
        mov     bl, dl
        shl     bl, 2   ; *4
        add     ebx, drv_names
        mov     edx, [ebx]
        mov     [drv_name], edx
        call    dwordtohex
        mov     [sector_num], edx
        mov     [sector_num+4], eax
        mov     al, 1
```

```
        mov     ah, 3Fh ; cyan background, white forecolor
        mov     ebx, dpheader
        call    print_line
        mov     al, 21
        ;mov    ah, 3Fh ; cyan background, white forecolor
        mov     ebx, dpfooter1
        call    print_line
        mov     al, 22
        ;mov    ah, 3Fh ; cyan background, white forecolor
        mov     ebx, dpfooter2
        call    print_line
ds1:
        mov     ecx, 16
ds2:
        mov     al, [paragr]
        call    bytetohex
        mov     [sdline_1], ax
        ;
        push    ecx
        mov     cl, 16
        mov     edi, sdline_2
ds3:
        lodsb
        call    bytetohex
        stosw
        inc     edi
        loop    ds3
        sub     esi, 16
        inc     edi
        mov     cl, 16
        rep     movsb
        pop     ecx
        mov     al, 19  ; line (row) 3 to 24
        sub     al, cl
        mov     ah, 07h ; Black background, light gray forecolor
        mov     ebx, sdline
        call    print_line_80 ; 04/12/2014
        loop    ds4
        retn
ds4:
        inc     byte [paragr]
        jmp     short ds2

; Convert binary number to hexadecimal string
; 01/12/2014
; 25/11/2014
;
bytetohex:
        ; INPUT ->
        ;       AL = byte (binary number)
        ; OUTPUT ->
        ;       AX = hexadecimal string
        ;
        push    ebx
        xor     ebx, ebx
        mov     bl, al
        shr     bl, 4
        mov     bl, [ebx+hexchrs]
        xchg    bl, al
        and     bl, 0Fh
        mov     ah, [ebx+hexchrs]
        pop     ebx
        retn
```

```
wordtohex:
        ; INPUT ->
        ;       AX = word (binary number)
        ; OUTPUT ->
        ;       EAX = hexadecimal string
        ;
        push    ebx
        xor     ebx, ebx
        xchg    ah, al
        push    ax
        mov     bl, ah
        shr     bl, 4
        mov     al, [ebx+hexchrs]
        mov     bl, ah
        and     bl, 0Fh
        mov     ah, [ebx+hexchrs]
        shl     eax, 16
        pop     ax
        pop     ebx
        jmp     short bytetohex
        ;mov    bl, al
        ;shr    bl, 4
        ;mov    bl, [ebx+hexchrs]
        ;xchg   bl, al
        ;and    bl, 0Fh
        ;mov    ah, [ebx+hexchrs]
        ;pop    ebx
        ;retn

dwordtohex:
        ; INPUT ->
        ;       EAX = dword (binary number)
        ; OUTPUT ->
        ;       EDX:EAX = hexadecimal string
        ;
        push    eax
        shr     eax, 16
        call    wordtohex
        mov     edx, eax
        pop     eax
        call    wordtohex
        retn

print_line_80:
        ; 04/12/2014
        ; al = line (0 to 24)
        ; ah = color attributes
        ; ebx = 80 chars string address
        call    get_lpos
        push    ecx
        mov     ecx, 80
pl80:
        mov     al, [ebx]
        inc     ebx
        stosw
        loop    pl80
        pop     ecx
        retn

print_line:
        ; al = line (0 to 24)
        ; ah = color attributes
        ; ebx = ASCIIZ string address
        call    get_lpos
```

```
        push    esi
        mov     esi,ebx
prl1:
        lodsb
        and     al, al
        jz      short prl2
        stosw
        jmp     short prl1
prl2:
        pop     esi
        retn

clear_frame:
        xor     al, al ; Line 0
        call    clear_line
        mov     al, 1
        mov     ah, 3Fh ; cyan background, white forecolor
        call    fill_color
        mov     al, 1
dscf0:
        inc     ax
        push    ax
        call    clear_line
        pop     ax
        cmp     al, 19
        jb      short dscf0
        ;inc    al ; line 20
        mov     ah, 3Fh
dscf1:
        inc     al
        push    ax
        call    fill_color
        pop     ax
        cmp     al, 23
        jb      short dscf1
        inc     al
        call    clear_line
        retn

clear_line:
        xor     ah, ah ; blank
fill_color:
        ; al = line (0 to 24)
        ; ah = color attributes
        call    get_lpos
        mov     ecx, 80 ; 23/02/2015
        mov     al, 20h ; space/blank
        rep     stosw
        retn

get_lpos:  ; Get line position in video buffer
        push    ax
        xor     edi, edi
        mov     ah, 80*2
        mul     ah
        mov     di, ax
        pop     ax
        add     edi, 0B8000h ; video page 0
        retn

inputbox:
        ; 12/02/2015 (unix386.s)
        ; 27/11/2014 (dsectrm2.s)
        ; Show an input box for user/keyboard input
```

```
        ; INPUT ->
        ;       ESI = input structure address
        ; OUTPUT ->
        ;       DX =  cursor position for input
        ;       input box will be displayed (on tty0)
        ;
        ; Modified registers: ax, ebx, ecx, dx, esi, edi
        ;

        mov     byte [inds], 0 ; for ENTER key handling
        xor     ecx, ecx
        ;xor    cx, cx   ; ecx = 0
        mov     ebx, 0B8000h
        mov     ax, 5018h ; 80, 24
        mov     dx, [esi] ; box width (dl)
                          ; box height (dh)
        sub     al, dh
        shr     al, 1
        mov     [ibcp+1], al ; row
        mul     ah
        shl     ax, 1  ; char + attribute
        ;mov    bx, ax
        add     bx, ax
        mov     al, 80
        sub     al, dl
        shr     al, 1
        mov     [ibcp], al ; column
        shl     al, 1  ; char + attribute
        sub     ah, ah
        add     bx, ax
        mov     ah, [esi+5] ; color attributes
        mov     al, 20h ; space/blank
        mov     cl, dh ; height
ib0:
        push    cx
        mov     cl, dl
        mov     edi, ebx
        rep     stosw
        pop     cx
        add     bx, 80*2 ; number of columns * 2
        loop    ib0
        ;
        mov     edi, 0B8000h
        mov     al, [ibcp+1] ; row position
        add     al, [esi+2] ; label offset (row)
        mov     [ibcp+1], al
        mov     ah, 80*2
        mul     ah
        add     di, ax
        mov     al, [ibcp] ; column position
        add     al, [esi+3] ; label offset (column)
        mov     [ibcp], al
        xor     ah, ah
        shl     al, 1
        add     di, ax
        mov     ebx, esi
        add     esi, 6 ; Label offset
ib2:
        lodsb
        or      al, al
        jz      short ib3
        stosb
        inc     edi
        inc     cl
```

```
        jmp     short ib2
ib3:
        add     [ibcp], cl ; column position
        mov     [current_txtpos], edi
        ;
        mov     cl, [ebx+4] ; input char count
        or      cl, cl
        jz      short ib5 ; message box (no input)
        mov     al, 20h
        mov     ah, 07h ; black background
                        ; light gray fore color
ib4:
        rep     stosw
        ;
        mov     dx, [ibcp] ; cursor position
ib5:
        retn

hide_cursor:
        ; 12/02/2015 (unix386.s)
        ; 01/12/2014 (dsectrm2.s)
        ;CH = cursor start line (bits 0-4)
        ;       and options (bits 5-7).
        ;CL = bottom cursor line (bits 0-4).
        ; when bit 5 of CH is set to 0, the cursor is visible.
        ; when bit 5 is 1, the cursor is not visible.
        ; hide blinking text cursor:
        mov     ch, 32
        jmp     short hc_sc

show_cursor:
        ; 12/02/2015 (unix386.s)
        ; 01/12/2014 - 13/12/2014 (dsectrm2.s)
        ; dh = row
        ; dl = column
        mov     [cursor_posn], dx
        sub     bl, bl  ; video page 0
        call    set_cpos
        ;
        ;show blinking text cursor
        mov     ch, 13 ; 16/02/2015
hc_sc:
        mov     cl, 15 ; 16/02/2015
        jmp     set_ctype

read_disk_sector:
        ; 21/02/2015
        ; 16/02/2015 (unix386.s)
        ; 01/12/2014 - 18/01/2015 (dsectrm2.s)
        xor     ebx, ebx
        mov     bl, [ds_drv]
        mov     esi, ebx ; 26/02/2015
        mov     dl, bl
        cmp     dl, 2
        jb      short rd0
        add     dl, 7Eh  ; 80h, 81h, 82h, 83h
rd0:
        mov     [drv], dl
        add     ebx, drv_status
        mov     dh, [ebx]
rd1:
        cmp     dh, 0F0h ; 18/1/2015
        cmc
        jc      rd_fails
```

```
        ;
        ;xor    ebx, ebx
        ;mov    bl, [ds_drv]
        ;;mov   esi, ebx
        mov     ebx, esi ; 26/02/2015
        shl     bl, 2
        add     ebx, ds_sec
        mov     eax, [ebx]
        sub     ebx, ds_sec
        add     ebx, disk_size
        cmp     eax, [ebx] ; Last sector + 1 (number of secs.)
        cmc
        jc      short rd_lba_fails
        ; 02/02/2015
        test    dh, 1 ; LBA ready ?
        jz      short rd_chs
rd_lba:
        ; LBA read (with temporary function)
        ;((Retro UNIX 386 v1 - DISK I/O Test))
        add     esi, drv_status
        and     byte [esi], 8Fh ; clear error bits
        ;
        ;mov    ebx, eax
        ;mov    cx, bx
        ;shr    ebx, 16
        ;;mov   di, cs
        ;;mov   es, di
        ;mov    edi, buffer
        ; 21/02/2015 (unix386.s, 32 bit modification)
        mov     ecx, eax
        mov     ebx, buffer
        mov     dl, [drv]
        mov     byte [retry_count], 4
rd_lba_retry:
        mov     ah, 1Bh ; Temporary/Private function
        mov     al, 1
        ;int    13h
        call    int13h
        jnc     short rd_lba_ok
        cmp     ah, 80h ; time out ?
        je      short rd_lba_rfails
        dec     byte [retry_count]
        jz      short rd_lba_rfails
        mov     ah, 0Dh ; Alternate reset
        ;int    13h
        call    int13h
        jnc     short rd_lba_retry
        or      byte [esi], 0F0h ; drive not ready !
rd_lba_rfails:
        stc
rd_lba_fails:
rd_lba_ok:
        retn
        ;
        ; CHS read (convert LBA address to CHS values)  ;
rd_chs:
        shl     esi, 1
        mov     ebx, esi
        xor     edx, edx ; 0
        sub     ecx, ecx
        add     ebx, spt
        mov     cx, [ebx] ; sector per track
                ; EDX:EAX = LBA
        div     ecx
```

```
        mov     cl, dl  ; sector number - 1
        inc     cl      ; sector number (1 based)
        push    cx
        mov     ebx, esi
        add     ebx, heads
        mov     cx, [ebx] ; heads
        xor     edx, edx
                ; EAX = cylinders * heads + head
        div     ecx
        pop     cx      ; sector number
        mov     dh, dl ; head number
        mov     dl, [drv]
        mov     ch, al ; cylinder (bits 0-7)
        shl     ah, 6
        or      cl, ah ; cylinder (bits 8-9)
                       ; sector (bits 0-7)
        ;;push  cs
        ;;pop   es
        mov     ebx, buffer
                ; CL = sector (bits 0-6)
                ;      cylinder (bits 7-8 -> bits 8-9)
                ; CH = cylinder (bits 0-7)
                ; DH = head
                ; DL = drive
        ; 02/02/2015
        shr     esi, 1 ; drive index (byte alignment)
        add     esi, drv_status
        and     byte [esi], 8Fh ; clear error bits
        ;
        mov     byte [retry_count], 4
rd_retry:
        mov     ah, 02h ; read sectors
        mov     al, 1 ; sector count
        ;int    13h
        call    int13h
        jnc     short rd_ok
        cmp     ah, 80h ; time out ?
        je      short rd_rfails
        dec     byte [retry_count]
        jnz     short rd_reset
rd_rfails:
        stc
rd_fails:
        retn
rd_reset:
        ; 02/02/2015
        sub     ah, ah
        cmp     dl, 80h
        jb      rd_fd_reset
        mov     ah, 0Dh ; Alternate reset
rd_fd_reset:
        ;int    13h
        call    int13h
        jnc     short rd_retry
        or      byte [esi], 0F0h ; drive not ready !
        stc
rd_ok:
        retn

retry_count: ; 16/12/2014
        db      0
;rd_lba:
;       mov     [lba], eax
;       mov     ax, cs
```

```
;       mov     [tbuff+2], ax
;       mov     esi, disk_pkt
;       ;mov    dl, [drv]
;       mov     ah, 42h ; Extended read
;       int     13h
;       retn

; 16/02/2015 (unix386.s)
; 19/12/2014 (dsectrm2.s)
clear_screen:
        ;;mov ah, 0Fh ; get video mode
        ;;int 10h
        ;;; al = video mode
        ;;mov ah, 0   ; set video mode (clears screen)
        ;;int 10h
        ;;retn
        ; 19/12/2014
        ;push   es
        push    edi
        ;mov    di, 0B800h
        ;mov    es, di
        ;sub    di, di
        mov     edi, 0B8000h
        mov     ecx, 1000
        ;mov cx, 2000
        ;mov    ax, 0720h ; light gray char space (blank)
        mov     eax,07200720h
        ;rep    stosw
        rep     stosd
        pop     edi
        ;pop    es
        xor     dx, dx    ; column 0, row 0
        sub     bl, bl ; 17/02/2015 - video page 0
        jmp     set_cpos ; set cursor position

rfdp_err:
        ;23/12/2014
        retn

dskprm:
        ; 20/02/2015
        ; 16/02/2015 (unix386.s)
        ; 19/12/2014 - 23/12/2014 (dsectrm2.s)
        ; DISPLAY DISK PARAMETERS TABLE
        ;
        ; INPUT -> DL = Disk/Drive #
        ;
        mov     byte [drv], dl  ; 0,1,80h,81h,82h,83h
        ;
        mov     ah, 08h ; Return drive parameters (conventional)
        ;int    13h
        call    int13h
        jc      short rfdp_err
        ;
        push    bx ; 20/02/2015
        ; 23/12/2014
        call    clear_screen       ; clear video page 0
        pop     bx ; 20/02/2015
        ;
        test    byte [drv], 80h
        jnz     print_hdpt
        add     bl, 30h ; '0'
        mov     byte [flpdtype], bl
                                   ; floppy disk drive type
```

```
                                        ; (1=360K, 2=1.2M, 3=720K, 4=1.44M)
print_flpdpt:
        ; 16/02/2015 (unix386.s)
        ; Writing the Diskette Parameter Table on screen
        ;mov    si, di
        mov     esi, edi
        ;mov    ax, es
        ;mov    ds, ax
        ;mov    ax, cs
        ;mov    es, ax
        lodsb   ; bits 0-3: SRT step rate time
                ; bits 4-7: head unload time
        mov     edi, rSrtHdUnld
        call    write_hex
        lodsb   ; bit 0: 1=use DMA
                ; bits 2-7: head load time
        mov     edi, rDmaHdLd
        call    write_hex
        lodsb   ; 55-ms increments
                ; before turning disk motor off
        mov     edi, bMotorOff
        call    write_hex
        lodsb   ; sector size
                ; (0=128, 1=256, 2=512, 3=1024)
        mov     edi, bSectSize
        call    write_hex
        lodsb   ; EOT (last sector on a track)
        mov     edi, bLastTrack
        call    write_hex
        lodsb   ; gap length
                ; for read/write operations
        mov     edi, bGapLen
        call    write_hex
        lodsb   ; DTL (Data Transfer Length)
                ; max transfer when length not set
        mov     edi, bDTL
        call    write_hex
        lodsb   ; gap length for format operation
        mov     edi, bGapFmt
        call    write_hex
        lodsb   ; fill character for format
                ; (normally F6H)
        mov     edi, bFillChar
        call    write_hex
        lodsb   ; head-settle time
                ; (in milliseconds)
        mov     edi, bHdSettle
        call    write_hex
        lodsb   ; motor-startup time
                ; (in 1/8th-second intervals)
        mov     edi, bMotorOn
        call    write_hex
        ; 19/12/2014
        ; (extension, not in original bios function)
        lodsb   ; Max. track number
        mov     edi, bMaxTrack
        call    write_hex
        lodsb   ; Data transfer rate
        mov     edi, bDataRate
        call    write_hex
        ;
        ;mov    ax, cs
        ;mov    ds, ax
       ;;mov     al, byte [ds_drv]
```

```
        mov     al, [drv] ; 11/01/2015
        add     al, 30h ; '0'
        mov     byte [flpdnum], al
        mov     esi, FLPDPT
        call    print_msg
        retn

write_dhex:
        ; 16/02/2015 (unix386.s)
        mov     bl, ah
        shr     bl, 1
        shr     bl, 1
        shr     bl, 1
        shr     bl, 1
        call    dhgd
        mov     bl, ah
        call    dhgd
write_hex:
        mov     bl, al
        shr     bl, 1
        shr     bl, 1
        shr     bl, 1
        shr     bl, 1
        call    dhgd
        mov     bl, al
        ;call   dhgd
        ;retn
dhgd:
        push    ax
        ; 16/02/2015 (unix386.s, 32 bit modifications)
        ;and    bx, 0Fh
        ;add    bx, hex_digits
        ;mov    al, byte [CS:BX]
        and     ebx, 0Fh
        add     ebx, hex_digits
        mov     al, [ebx]
        stosb
        pop     ax
        retn

print_hdpt:
        ; 20/02/2015
        ; 16/02/2015 (unix386.s)
        ; 23/12/2014 - 11/01/2015 (dsectrm2.s)
        ;; ES:DI = DPT address
        ;
        ;;mov   si, HD0_DPT
        mov     esi, (DPT_SEGM*16)+HD0_DPT
        mov     bl, [drv] ; 20/02/2015
        and     ebx, 3
        mov     al, bl
        add     al, 2
        mov     [ds_drv], al
        ;
        shl     bl, 5 ; * 32
        add     esi, ebx
        ;;mov   ax, DPT_SEGM
        ;;mov   ds, ax
        ;;mov   ax, cs
        ;;mov   es, ax
        cmp     byte [esi+3], 0A0h ; Translated table
        je      print_thdpt       ; indicator
        ;
        ; Writing Fixed Disk Parameter Table on screen
```

```
lodsw   ; Number of Cylinders
mov     edi, cylnum
call    write_dhex
lodsb   ; Number of Heads
mov     edi, headnum
call    write_hex
lodsb   ; Reserved
mov     edi, rsvd3
call    write_hex
lodsb   ; Reserved
mov     edi, rsvd4
call    write_hex
lodsw   ; Precompensation (Obsolete)
mov     edi, pcompnum
call    write_dhex
lodsb   ; Reserved
mov     edi, rsvd7
call    write_hex
lodsb   ; Drive Control Byte
mov     edi, dcbnum
call    write_hex
lodsw   ; Reserved
mov     edi, rsvd9
call    write_dhex
lodsb   ; Reserved
mov     edi, rsvd11
call    write_hex
lodsw   ; Landing Zone (Obsolete)
mov     edi, lzonenum
call    write_dhex
lodsb   ; Sectors per Track
mov     edi, psptnum
call    write_hex
lodsb   ; Reserved
mov     edi, rsvd15
call    write_hex
;
; (extension, not in original bios function)
lodsw   ; I/O Port Base Address
mov     edi, bPortAddr
call    write_dhex
lodsw   ; Control Port Address
mov     edi, cPortAddr
call    write_dhex
lodsb   ; Head Register Upper Nibble
mov     edi, hregupnib
call    write_hex
;
;;mov   ax, cs
;;mov   ds, ax
;sub    ebx, ebx
mov     al, [ds_drv]
mov     bl, al
add     al, '0'
mov     [dsknum], al
sub     bh, bh
shl     bl, 2
mov     esi, ebx
add     esi, disk_size
mov     ax, [esi+2]
mov     edi, disksize
call    write_dhex
mov     ax, [esi]
mov     edi, disksize+4
```

```
        call    write_dhex
        ;
        mov     esi, HDPT
        call    print_msg
        retn

print_thdpt:
        ; 16/02/2015 (unix386.s)
        ; 25/12/2014 (dsectrm2.s)
        ; Writing the Translated FDPT on screen
        ; (PHOENIX - EDD specification v1.1)
        lodsw   ; Logical Numbers of Cylinders, Limit 1024
        mov     edi, lcylnum
        call    write_dhex
        lodsb   ; Logical Numbers of Heads, Limit 256
        mov     edi, lheadnum
        call    write_hex
        lodsb   ; A0h signature, indicates translated table
        mov     edi, tsignum
        call    write_hex
        lodsb   ; Physical Sectors per Track
        mov     edi, tpsptnum
        call    write_hex
        lodsw   ; Precompensation (Obsolete)
        mov     edi, tpcompnum
        call    write_dhex
        lodsb   ; Reserved
        mov     edi, trsvd7
        call    write_hex
        lodsb   ; Drive Control Byte
        mov     edi, tdcbnum
        call    write_hex
        lodsw   ; Physical Cylinders, limit 65536
        mov     edi, tpcylnum
        call    write_dhex
        lodsb   ; Physical Heads, limit 16
        mov     edi, tpheadnum
        call    write_hex
        lodsw   ; Landing Zone (Obsolete)
        mov     edi, tlzonenum
        call    write_dhex
        lodsb   ; Logical Sectors per Track, Limit 63
        mov     edi, lsptnum
        call    write_hex
        lodsb   ; Checksum for translated FDPT
        mov     edi, checksum
        call    write_hex
        ;
        ; (extension, not in original bios function)
        lodsw   ; I/O Port Base Address
        mov     edi, tbPortAddr
        call    write_dhex
        lodsw   ; Control Port Address
        mov     edi, tcPortAddr
        call    write_dhex
        lodsb   ; Head Register Upper Nibble
        mov     edi, thregupnib
        call    write_hex
        ;
        ;;mov   ax, cs
        ;;mov   ds, ax
        ;sub    ebx, ebx
        mov     al, [ds_drv]
        mov     bl, al
```

```
        add     al, '0'
        mov     [tdsknum], al
        sub     bh, bh
        shl     bl, 2
        mov     esi, ebx
        add     esi, disk_size
        mov     ax, [esi+2]
        mov     edi, tdisksize
        call    write_dhex
        mov     ax, [esi]
        mov     edi, tdisksize+4
        call    write_dhex
        ;
        mov     esi, THDPT
        ;call   print_msg
        ;retn

print_msg:
        ;mov    bx, 7
        ;mov    ah, 0Eh
pmsg_loop:
        lodsb
        and     al, al
        jz      short pmsg_ok
        ;int    10h
        ; 16/02/2015
        push    esi
        xor     ebx, ebx ; video page 0
        mov     ah,  07h ; Black background, light gray forecolor
        call    write_tty
        pop     esi
        jmp     short pmsg_loop
pmsg_ok:
        ;mov    ah, 10h ; Getchar
        ;int    16h
        call    getch ; 16/02/2015
        retn
;
FLPDPT:
        db 07h
        db 0Dh, 0Ah
        db 'Disk '
flpdnum:
        db 'X - '
        db 'DISKETTE PARAMETER TABLE'
        db 0Dh, 0Ah, 0DH, 0Ah
        db 'Type                : '
flpdtype:
        db 'X   '
        db '[ 1 = 360K, 2 = 1.2M, 3 = 720K, 4 = 1.44M ]'
        db 0Dh, 0Ah, 0DH, 0Ah
        db 'SRT - Head Unld Time : '
rSrtHdUnld:
        db 'XXh (bits 0-3: SRT, bits 4-7: head unload time)'
        db 0Dh, 0Ah
        db 'DMA - Head Load Time : '
rDmaHdLd:
        db 'XXh (bit 0: 1 = DMA, bits 2-7: head load time)'
        db 0Dh, 0Ah
        db 'Motor Off Count     : '
bMotorOff:
        db 'XXh (with 55ms icrements before turning off)'
        db 0Dh, 0Ah
        db 'Sector Size         : '
```

```
bSectSize:
        db 'XXh (2 = 512 bytes)'
        db 0Dh, 0Ah
        db 'Last Sect on a Track : '
bLastTrack:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Gap Length  (R/W)    : '
bGapLen:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Data Transfer Length : '
bDTL:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Gap Length (Format)  : '
bGapFmt:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Fill Char for format : '
bFillChar:
        db 'XXh (normally F6h)'
        db 0Dh, 0Ah
        db 'Head Settle Time     : '
bHdSettle:
        db 'XXh milliseconds'
        db 0Dh, 0Ah
        db 'Motor Startup Time   : '
bMotorOn:
        db 'XXh (in 1/8th second intervals)'
        db 0Dh, 0Ah
        ; 19/12/2014
        db 0Dh, 0Ah
        db 'Maximum Track Number : '
bMaxTrack:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Data Transfer Rate   : '
bDataRate:
        db 'XXh (00h = 500KBS, 40h = 300KBS, 80H = 250KBS)'
        db 0Dh, 0Ah
        db 0Dh, 0Ah, 0

; 23/12/2014
HDPT:
        db 07h
        db 0Dh, 0Ah
        db 'Disk '
dsknum:
        db 'X - '
        db 'FIXED DISK PARAMETER TABLE'
        db 0Dh, 0Ah, 0DH, 0Ah
        db 'Number of Cylinders : '
cylnum:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Number of Heads     : '
headnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Reserved            : '
rsvd3:
        db 'XXh'
        db 0Dh, 0Ah
```

```
        db 'Reserved            : '
rsvd4:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Precompensation     : '
pcompnum:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Reserved            : '
rsvd7:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Drive Control Byte  : '
dcbnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Reserved            : '
rsvd9:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Reserved            : '
rsvd11:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Landing Zone        : '
lzonenum:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Sectors per Track   : '
psptnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Reserved            : '
rsvd15:
        db 'XXh'
        db 0Dh, 0Ah
        db 0Dh, 0Ah
        db 'I/O Port Base Addr  : '
bPortAddr:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Control Port Addr   : '
cPortAddr:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Head Reg Upp Nibb   : '
hregupnib:
        db 'XXh'
        db 0Dh, 0Ah
        db 0Dh, 0Ah
        db 'Size (in sectors)   : '
disksize:
        db 'XXXXXXXXh'
        db 0Dh, 0Ah
        db 0Dh, 0Ah, 0

THDPT:
        db 07h
        db 0Dh, 0Ah
        db 'Disk '
tdsknum:
        db 'X - '
        db 'TRANSLATED FIXED DISK PARAMETER TABLE'
        db 0Dh, 0Ah, 0DH, 0Ah
```

```
        db 'Logical Cylinders   : '
lcylnum:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Logical Heads       : '
lheadnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Signature           : '
tsignum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Phy Sec per Track   : '
tpsptnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Precompensation     : '
tpcompnum:
        db 'XXXXh  (Obsolete)'
        db 0Dh, 0Ah
        db 'Reserved            : '
trsvd7:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Drive Control Byte  : '
tdcbnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Physical Cylinders  : '
tpcylnum:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Physical Heads      : '
tpheadnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Landing Zone        : '
tlzonenum:
        db 'XXXXh  (Obsolete)'
        db 0Dh, 0Ah
        db 'Logic Sec per Trk   : '
lsptnum:
        db 'XXh'
        db 0Dh, 0Ah
        db 'Checksum            : '
checksum:
        db 'XXh'
        db 0Dh, 0Ah
        db 0Dh, 0Ah
        db 'I/O Port Base Addr  : '
tbPortAddr:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Control Port Addr   : '
tcPortAddr:
        db 'XXXXh'
        db 0Dh, 0Ah
        db 'Head Reg Upp Nibb   : '
thregupnib:
        db 'XXh'
        db 0Dh, 0Ah
        db 0Dh, 0Ah
        db 'Size (in sectors)   : '
tdisksize:
```

```
        db 'XXXXXXXXh'
        db 0Dh, 0Ah
        db 0Dh, 0Ah, 0

hex_digits:
hexchrs:
        db '0123456789ABCDEF'

inds:
        db 0
ds_drv:
        db 0FFh ; Current drive (on display)
        db 0    ; Current half (0 or >0)
ds_sec:
        dd 0   ; Current sector (on display), drv 0
        dd 0   ; Current sector (on display), drv 1
        dd 0   ; Current sector (on display), drv 2
        dd 0   ; Current sector (on display), drv 3
        dd 0   ; Current sector (on display), drv 4
        dd 0   ; Current sector (on display), drv 5
paragr:
        db 0

drv_names:
        db 'fd0 fd1 hd0 hd1 hd2 hd3 '

dpheader:
        db ' Drive : '
drv_name:
        db '000  '
        db  'Sector : '
sector_num:
        db  'FFFFFFFFh'
        db 0

current_txtpos:
        dd 0
txtposoff:
        db 0 ; txtpos offset for sector number input
dscmd:
        db 0   ; 0 = change drive
               ; 1 = change sector
               ; 2 = display disk parameters
sdline:
        db ' Byte '
sdline_1:
        db '000h'
        db ' -  '
sdline_2:
        db '00 00 00 00 00 00 00 00 '
        db '00 00 00 00 00 00 00 00 '
        db ' '
sdline_3:
        db '................'
        db 20h

dpfooter1:
        db ' F1 = Change Drive  '
        db 'Home = First Sector '
        db 'PgUp = Previous Sector '
        db 'ESC = EXIT'
        db 0
dpfooter2:
        db ' F2 = Change Sector '
```

```
        db 'End = Last Sector   '
        db 'PgDown = Next Sector   '
        db 'ENTER = Prv/Nxt'
        db 0
ibcp:
        db 0    ; input box - row position
        db 0    ; input box - column position
F1_ib:
        db 16   ; box width (columns)
        db 3    ; box height (rows)
        db 1    ; label offset (vertical)
        db 1    ; label offset (horizontal)
        db 1    ; text (input) size
        db 4Eh  ; box color
        db 'Drive: '  ; Label
        db 0

F2_ib:
        db 20   ; box width (columns)
        db 3    ; box height (rows)
        db 1    ; label offset (vertical)
        db 1    ; label offset (horizontal)
        db 8    ; text (input) size
        db 4Eh  ; box color
        db 'Sector : '  ; Label
        db 0

dskr_err:
        ;db 33  ; box width (columns)
        db 17
        db 3    ; box height (rows)
        db 1    ; label offset (vertical)
        db 1    ; label offset (horizontal)
        db 0    ; text (input) size
        db 4Eh  ; box color
        ;db 'Drive not ready or read error !'  ; Label
        db ' Error : '
err_code_str:
        db '00h ! '
        db 0

; Additional functions, variables/pointers for
; Real Mode adaption (out of unix386.s) variables/pointers

;28/11/2014
; (set_cpos in dsectrm2.s)
set_cposn:
        sub     bl, bl ; video page 0
        mov     dx, [cursor_posn] ; dh = row, dl = column
        jmp     set_cpos

align 2

; 16/02/2015
cursor_posb: dw 0
; 01/12/2014
cursor_shp: dw 0

;; 10/12/2014
;; 24/11/2014 (Retro UNIX 386 v1)
; Physical drive type & flags
;fd0_type:   db 0  ; floppy drive type
;fd1_type:   db 0  ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
                  ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
```

```
                ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
                ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
                ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
;hd0_type:   db 0  ; EDD status for hd0 (bit 7 = present flag)
;hd1_type:   db 0  ; EDD status for hd1 (bit 7 = present flag)
;hd2_type:   db 0  ; EDD status for hd2 (bit 7 = present flag)
;hd3_type:   db 0  ; EDD status for hd3 (bit 7 = present flag)
                ; bit 0 - Fixed disk access subset supported
                ; bit 1 - Drive locking and ejecting
                ; bit 2 - Enhanced disk drive support
                ; bit 3 = Reserved (64 bit EDD support)
                ; (If bit 0 is '1' Retro UNIX 386 v1
                ; will interpret it as 'LBA ready'!)
; 01/12/2014
drv_status: dw 0  ; fd0, fd1 (FFh = failure, 80h = existing)
            dd 0  ; hd0, hd1 hd2, hd3 (FFh =failure)
                  ;                   (80h - 87h = exiting)
                  ;                   (bit 0 = 1 : LBA ready)
; 10/12/2014
;hdc:   db 0      ; Hard (Fixed) disk count
;drv:   db 0      ; physical drive number (0, 1, 80h, 81h, 82h, 83h)

; 23/12/2014 (dw)
; 16/12/2014 (db)
wait_count: dw 0  ; INT 08h timer ticks for disk functions

; 02/12/2014
prev_sec:  dd 0   ; previous sector (before reading)
last_drv:  db 0   ; 25/12/2014 - physical drv num of last hard disk

; 03/01/2015
LBAMode: db 0

prg_msg:
        db 0Dh, 0Ah, 07h
        db 'Display Disk Sectors in 386 Protected Mode - Retro UNIX 386 v1 Disk
I/O test.'
        db 0Dh, 0Ah
        db 'by Erdogan Tan  [28/02/2015]'
        db 0Dh, 0Ah, 0Dh, 0Ah
        db '(Press any key to continue...)'
        db 0Dh, 0Ah, 0

; 16/12/2014
;drv_not_ready:
;       db 07h, 0Dh, 0Ah
;       db 'Drive not ready !'  ; Temporary
;       db 0Dh, 0Ah, 0

align 2

cylinders :  dw 0, 0, 0, 0, 0, 0
heads     :  dw 0, 0, 0, 0, 0, 0
spt       :  dw 0, 0, 0, 0, 0, 0
disk_size :  dd 0, 0, 0, 0, 0, 0

disk_pkt:
        db 16 ; Packet size
        db 0
        db 1  ; Transfer count (1 sector)
        db 0
tbuff:
        dw buffer ; offset
        dw 0 ; segment
```

```
lba:
        dd 0
        dd 0

; End of DSECTPM (Display Disk Sectors in Protected Mode) code

; ////////////////////////////////////////////////////////////////////////


Align 2

; 12/11/2014 (Retro UNIX 386 v1)
boot_drv:    db 0  ; boot drive number (physical)
; 24/11/2014
drv:         db 0  ; last drive number for hdd
hdc:         db 0  ; number of hard disk drives
                   ; (present/detected)
Align 2
;
; 24/11/2014 (Retro UNIX 386 v1)
; Physical drive type & flags
fd0_type:    db 0  ; floppy drive type
fd1_type:    db 0  ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
                   ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
                   ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
                   ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
                   ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
hd0_type:    db 0  ; EDD status for hd0 (bit 7 = present flag)
hd1_type:    db 0  ; EDD status for hd1 (bit 7 = present flag)
hd2_type:    db 0  ; EDD status for hd2 (bit 7 = present flag)
hd3_type:    db 0  ; EDD status for hd3 (bit 7 = present flag)
                   ; bit 0 - Fixed disk access subset supported
                   ; bit 1 - Drive locking and ejecting
                   ; bit 2 - Enhanced disk drive support
                   ; bit 3 = Reserved (64 bit EDD support)
                   ; (If bit 0 is '1' Retro UNIX 386 v1
                   ; will interpret it as 'LBA ready'!)
;
; 04/11/2014 (Retro UNIX 386 v1)
mem_1m_1k:   dw 0 ; Number of contiguous KB between
                  ;   1 and 16 MB, max. 3C00h = 15 MB.
mem_16m_64k: dw 0 ; Number of contiguous 64 KB blocks
                  ;   between 16 MB and 4 GB.
k_page_dir:  dd 0 ; Kernel's (System) Page Directory address
                  ;   (Physical address = Virtual address)
memory_size: dd 0 ; memory size in pages
free_pages:  dd 0 ; number of free pages
next_page:   dd 0 ; offset value in M.A.T. for
                  ;   first free page search
last_page:   dd 0 ; offset value in M.A.T. which
                  ;   next free page search will be
                  ; stopped after it. (end of M.A.T.)
first_page:  dd 0 ;   offset value in M.A.T. which
                  ; first free page search
                  ;   will be started on it. (for user)
mat_size:    dw 0 ; Memory Allocation Table size in pages


; 02/09/2014 (Retro UNIX 386 v1)
; 04/12/2013 (Retro UNIX 8086 v1)
CRT_START:   dw 0         ; starting address in regen buffer
                          ; NOTE: active page only
cursor_posn: times 8 dw 0 ; cursor positions for video pages
active_page:
```

```
ptty:       db 0         ; current tty
db 0
; 02/09/2014 (Retro UNIX 386 v1)
crt_ulc : db 0 ; upper left column (for scroll)
          db 0 ; upper left row (for scroll)

crt_lrc : db 79 ; lower right column (for scroll)
          db 24 ; lower right row (for scroll)

; 07/09/2014
ttychr: times 8 dw 0    ; Character buffer (multiscreen)

; 06/11/2014 (Temporary data)
; Memory Information message
msg_memory_info:
        db      "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
        db      "Total memory  : "
mem_total_b_str: ; 10 digits
        db      "0000000000 bytes", 0Dh, 0Ah
        db      "                ", 20h, 20h, 20h
mem_total_p_str: ; 7 digits
        db      "0000000 pages", 0Dh, 0Ah
        db      0Dh, 0Ah
        db      "Memory 1M-16M :   "
mem_1m_1k_str:   ; 5 digits
        db      "00000  1K blocks", 0Dh, 0Ah
        db      "Memory 16M-4G :   "
mem_16m_64k_str: ; 5 digits
        db      "00000 64K blocks", 0Dh, 0Ah
        db      0Dh, 0Ah
        db      "Free memory : "
free_mem_b_str:  ; 10 digits
        db      "0000000000 bytes", 0Dh, 0Ah
        db      "              ", 20h, 20h, 20h
free_mem_p_str:  ; 7 digits
        db      "0000000 pages", 0Dh, 0Ah
        db      0Dh, 0Ah
        db      "First free page : "
next_mem_p_str:  ; 7 digits
        db      "0000000", 0Dh, 0Ah
        DB      0Dh, 0Ah
        db      "M.A.T. size: "
mat_p_str:       ; 4 digits
        db      "0000 pages", 0Dh, 0Ah
        db      "Kernel page tables : "
pt_c_str:        ; 4 digits
        db      "0000 + 1", 0Dh, 0Ah
        db      0Dh, 0Ah
        db      "Calculated free memory : "
free_mem_c_str:  ; 7 digits
        db      "0000000 pages", 0Dh, 0Ah
        db      0Dh, 0Ah, 0

align 2

; 12/02/2015
buffer:
; 27/12/2013
_end:  ; end of kernel code (and read only data, just before bss)
```