

BIOS ASM

By Hicks

Date: Wed, 13 Jan 88 04:26 PST
From: <MULTI%TRUIMFCL.BITNET@CUNYVM.CUNY.EDU>
Subject: BIOS.ASM update INFO-IBMPC librarys
To: hicks@walker-emh.arpa

Page 80,132

Title BIOS - For Intel 8088 or NEC "V20" turbo motherboards. Use MASM 4.0

```
;  
; This bios will work on IBM-PC/xt and many other compatibles that share a  
; similar design concept. You do not need to have a turbo motherboard to  
; use this bios, but if you do, then use the following key sequence  
;                               CTRL ALT -  
; to toggle the computer speed between fast and slow (=IBM compatible)  
;  
; This BIOS can produce the following error messages at IPL time  
;  
ER_BIOS equ      01h          ; Bad ROM bios checksum, patch last byte  
ER_RAM  equ      02h          ; Bad RAM in main memory, replace  
ER_CRT  equ      04h          ; Bad RAM in video card, replace  
ER_MEM  equ      10h          ; Bad RAM in vector area, replace  
ER_ROM  equ      20h          ; Bad ROM in expansion area, bad checksum  
;  
; The last two bytes have to be patched with DEBUG as follows  
;  
;   FFFF 00.xx          ( avoid ER_BIOS on bootstrap ) -----  
;   FFFE 00.FE          ( leaves IBM-PC/xt signature ) -----  
;  
; where "xx" results in a zero checksum for the whole BIOS rom, for ex  
;  
;           masm BIOS;          ( Assemble BIOS source code)  
;           link BIOS;         ( Link the BIOS object code)  
;           debug BIOS.EXE      ( Exe2bin  BIOS binary code)  
;           -nBIOS.BIN          ( Name of the output binary)  
;           -eCS:FFFE          ( Opens BIOS signature byte)  
;           .FE                 ( Leave IBM-PC/xt signature) <--  
;           -eCS:FFFF          ( Opens BIOS checksum  byte)  
;;  -----> .DC                ( Force ROM checksum = zero) <-----  
;;           -rBX              ( Opens hi order byte count)  
;;           :0                 ( ... must be 0 bytes long)  
;;           -rCX              ( Opens lo order byte count)  
;;           :2000              ( ... BIOS 2000 bytes long)  
;;           -wCS:E000          ( Output to BIOS.BIN  file)  
;;           -q  
;;  
;; You must correct the checksum by manually patching the last byte so as  
the  
;; the entire 2764-2 eprom sums to zero. I wish DEBUG could checksum  
blocks.  
;  
; *****Miscellaneous definitions*****  
;  
*
```

BIOS ASM

By Hicks

```
MAX_MEMORY      =704                ; Maximum kilobytes of memory allowed      *
;SLOW_FLOPPY    =1                  ; Define to run floppy always at 4.77 mHz  *
;
; *****Miscellaneous definitions*****
;
entry    macro    x
        pad      =BANNER - $ + x - 0E000h
        if pad LT 0
            .err
            %out  'No room for ENTRY point'
        endif
        if pad GT 0
            db    pad DUP(0FFh)
        endif
    endm
;
jmpf     macro    x,y
        db      0EAh;
        dw      y,x
    endm
;
retf     macro    x
        ifb     <x>
            db   0CBh
        else
            db   0CAh
            dw   x
        endif
    endm
;
LF       equ     0Ah
CR       equ     0Dh
;
        .SALL                                ; Suppress Macro Expansions
        .LFCOND                               ; List False Conditionals
;
ASSUME   DS:code, SS:code, CS:code, ES:code
data     SEGMENT at 40h                      ; IBM compatible data
structure
        dw     4 dup(?)                      ; 40:00                ; RS232 com. ports - up to
four
        dw     4 dup(?)                      ; 40:08                ; Printer ports      - up to
four
        dw     ?                             ; 40:10                ; Equipment present word
                                                ; + (1 iff floppies) *
1.
                                                ; + (# 64K sys ram ) *
4.
                                                ; + (init crt mode ) *
16.
                                                ; + (# of floppies ) *
64.
```

BIOS ASM

By Hicks

```

; + (# serial ports) *
512.
; + (1 iff toy port) *
4096.
; + (# parallel LPT) *
16384.
us      db      ?          ; 40:12      ; MFG test flags, unused by
      dw      ?          ; 40:13      ; Memory size, kilobytes
      db      ?          ; 40:15      ; IPL errors<-
table/scratchpad
      db      ?          ; ...unused
;-----[Keyboard data area]-----;
flags  db      ?,?       ; 40:17      ; Shift/Alt/etc. keyboard
here   db      ?          ; 40:19      ; Alt-KEYPAD char. goes
      dw      ?          ; 40:1A      ; --> keyboard buffer head
      dw      ?          ; 40:1C      ; --> keyboard buffer tail
      dw      16 dup(?)   ; 40:1E      ; Keyboard Buffer
(Scan,Value)
;-----[Diskette data area]-----;
- 3    db      ?          ; 40:3E      ; Drive Calibration bits 0
      db      ?          ; 40:3F      ; Drive Motor(s) on 0-
3,7=write
off    db      ?          ; 40:40      ; Ticks (18/sec) til motor
byte   db      ?          ; 40:41      ; Floppy return code stat
req.   ; 1 = bad ic 765 command
      ; 2 = address mark not
found  ; 3 = write to protected
disk   ; 4 = sector not found
      ; 8 = data late (DMA
overrun) ; 9 = DMA failed 64K page
end    ; 16 = bad CRC on floppy
read   ; 32 = bad NEC 765
controller
failed ; 64 = seek operation
      ;128 = disk drive timed out
      db      7 dup(?)   ; 40:42      ; Status bytes from NEC 765
;-----[Video display area]-----;
      db      ?          ; 40:49      ; Current CRT mode
(software)
```

BIOS ASM

By Hicks

```
color) ; 0 = 40 x 25 text (no
color) ; 1 = 40 x 25 text (16
color) ; 2 = 80 x 25 text (no
color) ; 3 = 80 x 25 text (16
color) ; 4 = 320 x 200 grafix 4
color ; 5 = 320 x 200 grafix 0
color ; 6 = 640 x 200 grafix 0
color ; 7 = 80 x 25 text (mono
card)
    dw    ? ; 40:4A ; Columns on CRT screen
    dw    ? ; 40:4C ; Bytes in the regen region
    dw    ? ; 40:4E ; Byte offset in regen
region
    dw    8 dup(?) ; 40:50 ; Cursor pos for up to 8
pages
    dw    ? ; 40:60 ; Current cursor mode
setting
    db    ? ; 40:62 ; Current page on display
    dw    ? ; 40:63 ; Base addres (B000h or
B800h)
    db    ? ; 40:65 ; ic 6845 mode reg.
(hardware)
    db    ? ; 40:66 ; Current CGA palette
;-----[Used to setup ROM]-----;
    dw    ?,? ; 40:67 ; Eprom base Offset,Segment
    db    ? ; 40:6B ; Last spurious interrupt
IRQ
;-----[Timer data area]-----;
    dw    ? ; 40:6C ; Ticks since midnite (lo)
    dw    ? ; 40:6E ; Ticks since midnite (hi)
    db    ? ; 40:70 ; Non-zero if new day
;-----[System data area]-----;
    db    ? ; 40:71 ; Sign bit set iff break
    dw    ? ; 40:72 ; Warm boot iff 1234h value
;-----[Hard disk scratchpad]-----;
    dw    ?,? ; 40:74 ;
;-----[Timout areas/PRT/LPT]-----;
    db    4 dup(?) ; 40:78 ; Ticks for LPT 1-4
timeouts
    db    4 dup(?) ; 40:7C ; Ticks for COM 1-4
timeouts
;-----[Keyboard buf start/nd]-----;
    dw    ? ; 40:80 ; Contains 1Eh, buffer
start
    dw    ? ; 40:82 ; Contains 3Eh, buffer end
```

BIOS ASM

By Hicks

```
data      ENDS

dosdir    SEGMENT at 50h                ; Boot disk directory from
IPL
xerox     label    byte                ; 0 if Print Screen idle
                                           ; 1 if PrtSc xeroxing

screen
                                           ;255 if PrtSc error in
xerox
                                           ; ...non-grafix PrtSc in
bios
        db      200h dup(?)            ; PC-DOS bootstrap
procedure
                                           ; ...IBMBIO.COM buffers
the
                                           ; ...directory of the boot
                                           ; ...device here at IPL
time
                                           ; ...when locating the
guts
                                           ; ...of the operating
system
                                           ; ...filename "IBMDOS.COM"
dosdir    ends

dosseg    SEGMENT at 70h                ; "Kernel" of PC-DOS op sys
;IBMBIO.COM file loaded by boot block. Device Drivers/Bootstrap.
CONTIGUOUS<---
;IBMDOS.COM operating system nucleus immediately follows IBMBIO.COM and
|
; doesn't have to be contiguous. The IBMDOS operating system nucleus
|
; binary image is loaded by transient code in IBMBIO binary image
|
dosseg    ends                          ;
|
iplseg    SEGMENT at 0h                ; Segment for boot block
|
;The following boot block is loaded with 512. bytes on the first sector of
|
;the bootable device by code resident in the ROM-resident bios. Control is
|
;then transferred to the first word 0000:7C00 of the disk-resident
bootstrap |
        ORG      07C00h                ; ..offset for boot block
|
boot      db      200h dup(?)            ; ..start disk resident
boot--
iplseg    ends

code      SEGMENT
        ORG      0E000h
```

BIOS ASM

By Hicks

```
BANNER db ' Generic Turbo XT Bios 1987',CR,LF
db ' for 8088 or V20 cpu',CR,LF
db ' (c)Anonymous',CR,LF
db LF,0

LPTRS dw 03BCh,0378h,0278h ; Possible line printer
ports

ENTRY 0E05Bh ; IBM restart entry point

COLD: MOV AX,40h ; Entered by POWER_ON/RESET
MOV DS,AX
MOV Word ptr DS:72h,0 ; Show data areas not init

WARM: CLI ; Begin FLAG test of CPU
XOR AX,AX
JB HALT
JO HALT
JS HALT
JNZ HALT
JPO HALT
ADD AX,1
JZ HALT
JPE HALT
SUB AX,8002h
JS HALT
INC AX
JNO HALT
SHL AX,1
JNB HALT
JNZ HALT
SHL AX,1
JB HALT

MOV BX,0101010101010101b ; Begin REGISTER test of
CPU
CPUTST: MOV BP,BX
MOV CX,BP
MOV SP,CX
MOV DX,SP
MOV SS,DX
MOV SI,SS
MOV ES,SI
MOV DI,ES
MOV DS,DI
MOV AX,DS
CMP AX,0101010101010101b
JNZ CPU1
NOT AX
MOV BX,AX
JMP CPUTST
```

BIOS ASM

By Hicks

```
CPU1:   XOR     AX,1010101010101010b
        JZ      CPU_OK

HALT:   HLT

CPU_OK: CLD
        MOV     AL,0                ; Prepare to initialize
        OUT    0A0h,AL              ; ...no NMI interrupts
        MOV     DX,3D8h             ; Load Color Graphic port
        OUT    DX,AL                ; ...no video display
        MOV     DX,3B8h             ; Load Monochrome port
        INC     AL                  ; ...no video display
        OUT    DX,AL                ; ...write it out
        MOV     AL,10011001b        ; Program 8255 PIA chip
        OUT    63h,AL               ; ...Ports A & C, inputs
        MOV     AL,10100101b        ; Set (non)turbo mode
        OUT    61h,AL               ; ...on main board

        MOV     AL,01010100b        ; ic 8253 inits memory
refresh OUT    43h,AL                ; ...chan 1 pulses ic 8237
to      MOV     AL,12h               ; ...dma every 12h clock
ticks  OUT    41h,AL                ; ...64K done in 1
millisecond
clock  MOV     AL,01000000b          ; Latch value 12h in 8253
counter OUT    43h,AL                ; ...chip channel 1

IC8237: MOV     AL,0                ; Do some initialization
        OUT    81h,AL               ; ...dma page reg, chan 2
        OUT    82h,AL               ; ...dma page reg, chan 3
        OUT    83h,AL               ; ...dma page reg, chan
0,1    OUT    0Dh,AL                ; Stop DMA on 8237 chip
read   MOV     AL,01011000b          ; Refresh auto-init dummy
chip   OUT    0Bh,AL                ; ...on channel 0 of DMA
        MOV     AL,01000001b        ; Block verify
chip   OUT    0Bh,AL                ; ...on channel 1 of DMA
        MOV     AL,01000010b        ; Block verify
chip   OUT    0Bh,AL                ; ...on channel 2 of DMA
        MOV     AL,01000011b        ; Block verify
chip   OUT    0Bh,AL                ; ...on channel 3 of DMA
        MOV     AL,0FFh              ; Refresh byte count
```

BIOS ASM

By Hicks

```

        OUT     1,AL           ; ...send lo order
        OUT     1,AL           ; ...send hi order
        MOV     AL,0           ; Initialize 8237 command
reg
        OUT     8,AL           ; ...with zero
        OUT     0Ah,AL         ; Enable DMA on all
channels
        MOV     AL,00110110b   ; Set up 8253 timer chip
        OUT     43h,AL         ; ...chan 0 is time of
day
        MOV     AL,0           ; Request a divide by
        OUT     40h,AL         ; ...65536 decimal
        OUT     40h,AL         ; ...0000h or 18.2
tick/sec
        MOV     DX,213h        ; Expansion unit port
        MOV     AL,1           ; ...enable it
        OUT     DX,AL          ; ...do the enable
        MOV     AX,40h         ; Get bios impure segment
        MOV     DS,AX          ; ...into DS register
        MOV     SI,DS:72h      ; Save reset flag in SI reg
        XOR     AX,AX          ; ...cause memory check
        MOV     BP,AX          ; ...will clobber the flag
        MOV     BX,AX          ; Start at segment 0000h
        MOV     DX,55AAh       ; ...get pattern
        CLD                    ; Strings auto-increment

MEMSIZ: XOR     DI,DI           ; Location XXXX:0
        MOV     ES,BX           ; ...load segment
        MOV     ES:[DI],DX      ; ...write pattern
        CMP     DX,ES:[DI]      ; ...compare
        JNZ     MEM_ND          ; ...failed, memory end
        MOV     CX,2000h        ; Else zero 16 kilobytes
        REPZ   STOSW            ; ...with instruction
        ADD     BH,4            ; ...get next 16K bytes
ifdef   MAX_MEMORY
        CMP     BH,MAX_MEMORY SHR 2 ; Found max legal user ram?
else
        CMP     BH,0A0h         ; Found max legal IBM ram?
endif
        JNZ     MEMSIZ          ; ...no, then check more

MEM_ND: MOV     DS:72h,SI       ; Save pointer
        XOR     AX,AX
        MOV     ES,AX           ; ES = vector segment
        MOV     AX,80h
        MOV     SS,AX           ; Set up temporary stack at
        MOV     SP,100h         ; 0080:0100 for memory
check
        PUSH   BP
        PUSH   BX
        MOV    BP,2
```

BIOS ASM

By Hicks

```
CALL      MEMTST                ; Memory check ES:0 -
ES:0400
POP       AX
MOV       CL,6
SHR      AX,CL
MOV      DS:13h,AX
POP      AX
JNB      MEM_01
OR       AL,ER_MEM              ; Show vector area bad

MEM_01:  MOV      DS:15h,AL      ; Save IPL error code
        XOR      AX,AX
        PUSH     AX
        PUSH     AX
        PUSH     AX
        PUSH     AX
        MOV      AX,30h         ; Set up IBM-compatible
stack
        MOV      SS,AX          ; ...segment 0030h
        MOV      SP,100h       ; ...offset 0100h
        PUSH     DS
        MOV      BX,0E000h     ; Check BIOS eprom
        PUSH     CS
        POP      DS            ; ...at F000:E000
        MOV      AH,1
        CALL     CHKSUM        ; ...for valid checksum
        POP      DS            ; ...restore impure<-DS
        JZ       IC8259
        OR       Byte ptr DS:15h,ER_BIOS ; Checksum error BIOS eprom

IC8259:  CLI                  ; Init interrupt controller
        MOV      AL,13h
        OUT     20h,AL
        MOV      AL,8
        OUT     21h,AL
        MOV      AL,9
        OUT     21h,AL
        MOV      AL,0FFh
        OUT     21h,AL
        PUSH     DS
        XOR      AX,AX         ; 8 nonsense vectors begin
table
        MOV      ES,AX         ; ...at segment 0000h
        PUSH     CS
        POP      DS
        MOV      CX,8          ; Vectors 00h - 07h unused
        XOR      DI,DI        ; ...we start at vec 00h

LO_VEC:  MOV      AX,offset IGNORE ; Nonsense interrupt from
RSX
        STOSW
```

BIOS ASM

By Hicks

```

        MOV     AX,CS                ; ...bios ROM segment
        STOSW
        LOOP   LO_VEC

table   MOV     SI,offset VECTORS    ; SI --> Vector address

busy   MOV     CX,18h                ; ... vectors 08h - 1Fh

HI_VEC: MOVSW                       ; Get INTERRUPT bios ROM
offset

segment MOV     AX,CS                ; ...INTERRUPT bios ROM
        STOSW
        LOOP   HI_VEC

        MOV     AX,0F600h           ; AX --> Rom basic segment
        MOV     DS,AX               ; DS --> " " "
        XOR     BX,BX               ; BX = Rom basic offset
        MOV     AH,4                ; Four basic roms to check

        MOV     BP,SP               ; Save the stack pointer
        PUSH   CS                   ; ...push code segment
        MOV     DX,offset SKIP      ; Save the code offset
        PUSH   DX                   ; ...for RAM_PATCH

subroutine
        MOV     DX,0EA90h           ; Mov DX,'NOP,JMP_FAR'
        PUSH   DX                   ; ...save it on stack
        MOV     DX,0178Bh          ; Mov DX,'MOV DX,[BX]'
        PUSH   DX                   ; ...save it on stack
        PUSH   SS                   ; Save stack segment
        MOV     DX,SP               ; ...get the stack offset
        ADD    DX,02h               ; ...calculate xfer addr.
        PUSH   DX                   ; ...save it on the stack
;
        RETF                         ; Test for BASIC rom
;
; ////////////////////////////////////////
; MOV     DX,[BX]                    ; Executes off the stack ;
; JMPF    0F000h,SKIP                ; ...in RAM space ;
; ////////////////////////////////////////
SKIP:   MOV     SP,BP                ; Restore the stack pointer
        CMP    DL,DH                ; ...compare 1st and 2nd
byte
        JE     kosher                ; ...perfection. No
piracy

B_ROM:  CALL   CHKSUM                ; Scan for BASIC roms
        JNZ   kosher                ; ...bad basic rom
        DEC   AH                     ; Continue
        JNZ   B_ROM                 ; ...yes, more

        POP   DS                     ; Else valid basic
```

BIOS ASM

By Hicks

```
MOV     DI,60h                ; ...install basic

XOR     AX,AX                ; ...zero BASIC interrupt
STOSW                      ; ...offset
MOV     AX,0F600h            ; ...F600h BASIC interrupt
STOSW                      ; ...segment

kosher: PUSH     DS
POP     DS                ; Setup special low vectors
MOV     Word ptr ES:8,offset int_2 ; ...NMI interrupt
MOV     Word ptr ES:14h,offset int_5 ; ...print screen

interrupt
MOV     Word ptr ES:7Ch,0    ; No special graphics chars.
MOV     Word ptr ES:7Eh,0    ; ...so zero vector 1Fh
MOV     DX,61h
IN      AL,DX              ; Read machine flags
OR      AL,00110000b        ; ...clear old parity

error
OUT     DX,AL              ; Write them back to reset
AND     AL,11001111b        ; ...enable parity
OUT     DX,AL              ; Write back, parity

enabled
MOV     AL,80h              ; ...allow NMI interrupts
OUT     0A0h,AL
MOV     AX,0000000000110000b ; Assume monochrome video
MOV     DS:10h,AX          ; ...card has been

installed
INT     10h                ; ...initialize if present
MOV     AX,0000000000100000b ; Assume color/graphics

video
MOV     DS:10h,AX          ; ...card has been

installed
INT     10h                ; ...initialize if present
IN      AL,62h              ; Get memory size (64K

bytes)
AND     AL,00001111b        ; ...in bits 2,3 lo nibble
MOV     AH,AL              ; Save memory size nibble
MOV     AL,10101101b
OUT     61h,AL
IN      AL,62h              ; Get no. of floppies (0-3)
MOV     CL,4                ; ...and init. video mode
SHL     AL,CL              ; ...shift in hi nibble
OR      AL,AH
MOV     AH,0
MOV     DS:10h,AX          ; Start building Equipment

Flag
AND     AL,00110000b        ; ...if video card, mode

set
JNZ     LE232              ; ...found video interface
MOV     AX,offset DUMMY    ; No hardware, DUMMY:

becomes
MOV     ES:40h,AX          ; ...INT_10 video service
```

BIOS ASM

By Hicks

```

        JMP     short   LE235

LE232:  CALL    V_INIT           ; Setup video

LE235:  MOV     AL,00001000b     ; Read low switches
        OUT    61h,AL
        MOV    CX,2956h

WAIT_1: LOOP    WAIT_1
        MOV    AL,11001000b     ; Keyboard acknowledge
        OUT    61h,AL           ; ...send the request
        XOR    AL,10000000b     ; Toggle to enable
        OUT    61h,AL           ; ...send key enable
        MOV    AX,1Eh           ; Offset to buffer start
        MOV    DS:1Ah,AX        ; Buffer head pointer
        MOV    DS:1Ch,AX        ; Buffer tail pointer
        MOV    DS:80h,AX        ; Buffer start
        ADD    AX,20h           ; ...size
        MOV    DS:82h,AX        ; Buffer end
        JMP    short   V_CONT

FAO:    MOV     DL,AL           ; Formatted ascii output

FAO_1:  MOV     AX,BX           ; Get position for
        CALL   LOCATE          ; ...cursor routine
        PUSH   SI              ; Get string address
        CALL   PRINT           ; ...print string
        MOV    AX,ES:[BP+0]     ; Get port # to print
        CALL   BIGNUM          ; ...four digits
        POP    SI              ; Restore string address
        INC    BP              ; ...Address of port
        INC    BP              ; ...is two bytes long
        INC    BH              ; ...down one line
        DEC    DL              ; Decrement device count
        JNZ   FAO_1            ; ...back for more
        RET

K_BYTE: CLC                   ; Say no error
        MOV    AL,DL           ; ...size "checked"
        INC    AL              ; ...show more
        DAA
        MOV    DL,AL
        JNB   KBY_01
        MOV    AL,DH           ; ...do carry
        ADC    AL,0
        DAA
        MOV    DH,AL

KBY_01: MOV    AL,DH
        CALL   DIGIT           ; Print hex digit
        MOV    AL,DL
        MOV    CL,4
```

BIOS ASM

By Hicks

```
ROR    AL,CL
CALL   DIGIT           ; Print hex digit
MOV    AL,DL
CALL   DIGIT           ; Print hex digit
RET

TIMER: MOV    DX,241h   ; Check for timer #2 port
      CLI
      IN     AL,DX     ; ..read BCD seconds/100
      STI
      CMP    AL,99h    ; Are BCD digits in range?
      JBE    SER_01    ; ...yes, port exists
;
      MOV    DX,341h   ; Check for timer #1 port
      CLI
      IN     AL,DX     ; ..read BCD seconds/100
      STI
      CMP    AL,99h    ; Are BCD digits in range?
      JBE    SER_01    ; ...yes, port exists
;
      STC
      RET           ; No hardware, ports 0FFh

SER_01: CLC           ; Found timer(s) answering
      RET

V_CONT: MOV    BP,4    ; Assume monochrome, 4K
memory
      MOV    BX,0B000h ; ...segment in BX
      MOV    AL,DS:49h ; Get the video mode
      CMP    AL,7      ; ...was it mono?
      JZ     M_SEG     ; ...yes, skip
      MOV    BP,10h    ; Else CGA, has 16K memory
      MOV    BX,0B800h ; ...segment in BX

M_SEG:  PUSH   BX      ; Load video seg in ES
      POP    ES
      MOV    AL,DS:65h ; Get CRT hardware mode
      AND    AL,11110111b ; ...disable video
      MOV    DX,DS:63h ; Get 6845 index port
      ADD    DX,4      ; ...add offset for
      OUT    DX,AL     ; 6845 controller port

CRTRAM: CALL   MEMTST  ; Memory check ES:0 -
ES:0400
      DEC    BP
      JNZ   CRTRAM    ; Loop until CRT RAM
checked
      JNB   LE2F5
      OR    Byte ptr DS:15h,ER_CRT ; Set CRT RAM error in
status
```

BIOS ASM

By Hicks

```
LE2F5:  CALL    V_INIT
        MOV     AX,1414h                ; Time-out value seconds
        MOV     DS:78h,AX              ; ...LPT1
        MOV     DS:7Ah,AX              ; ...LPT2
        MOV     AX,101h                ; Time-out value seconds
        MOV     DS:7Ch,AX              ; ...COM1
        MOV     DS:7Eh,AX              ; ...COM2
        MOV     SI,offset LPTRS        ; SI --> LPTR port table
        XOR     DI,DI                  ; ...offset into data seg
        MOV     CX,3                   ; ...number of printers

NXTPRT: MOV     DX,CS:[SI]              ; Get LPTR port
        MOV     AL,10101010b           ; ...write value
        OUT     DX,AL                  ; ...to the LPTR
        MOV     AL,11111111b           ; Dummy data value
        OUT     0C0h,AL                ; ...on the bus
        IN      AL,DX                  ; Read code back
        CMP     AL,10101010b           ; ...check code
        JNZ     NO_LPT                 ; ...no printer found
        MOV     [DI+8],DX              ; Save printer port
        INC     DI
        INC     DI

NO_LPT: INC     SI
        INC     SI
        LOOP    NXTPRT
        MOV     AX,DI                  ; Number of printers * 2
        MOV     CL,3                   ; ...get shift count
        ROR     AL,CL                  ; ...divide by eight
        MOV     DS:11h,AL              ; ...save in equip. flag

        XOR     DI,DI                  ; com port(s) at 40:00
(hex)

COM_1:  MOV     DX,3FBh                 ; COM #1 line control reg.
        MOV     AL,00011010b           ; ...7 bits, even parity
        OUT     DX,AL                  ; Reset COM #1 line cont.
reg
        MOV     AL,11111111b           ; ...noise pattern
        OUT     0C0h,AL                ; Write pattern on data
buss
        IN      AL,DX                  ; ...read result from COM
#1
        CMP     AL,00011010b           ; Check if serial port
exists
        JNZ     COM_2                  ; ...skip if no COM #1
port
        MOV     Word ptr [DI],3F8h     ; Else save port # in
impure
        INC     DI                      ; ...potential COM #2 port
        INC     DI                      ; ...is at 40:02 (hex)
```

BIOS ASM

By Hicks

```
COM_2:  MOV     DX,2FBh           ; COM #2 line control reg
        MOV     AL,00011010b    ; ...7 bits, even parity
        OUT     DX,AL           ; Reset COM #2 line cont.
reg
        MOV     AL,11111111b    ; ...noise pattern
        OUT     0C0h,AL         ; Write pattern on data
buss
        IN      AL,DX           ; ...read results from COM
#2
        CMP     AL,00011010b    ; Check if serial port
exists  JNZ     COM_CT         ; ...skip if no COM #2
port
        MOV     word ptr [DI],2F8h ; Else save port # in
impure  INC     DI              ; ...total number of
serial  INC     DI              ; ...interfaces times two
COM_CT: MOV     AX,DI           ; Get serial interface
count  OR      DS:11h,AL        ; ...equip. flag
        MOV     DX,201h
        IN      AL,DX          ; Read game controller
        TEST    AL,0Fh         ; ...anything there?
        JNZ     NOGAME        ; ...yes, invalid
        OR      Byte ptr DS:11h,00010000b ; Else game port present
NOGAME: MOV     DX,0C000h      ; ROM segment start
        PUSH    DS
FNDROM: MOV     DS,DX          ; Load ROM segment
        XOR     BX,BX          ; ...ID offset
        MOV     AX,[BX]        ; Read the ROM id
        CMP     AX,0AA55h
        JNZ     NXTROM        ; ...not valid ROM
        MOV     AX,40h
        MOV     ES,AX
        MOV     AH,0
        MOV     AL,[BX+2]      ; Get ROM size (bytes *
512)
        MOV     CL,5
        SHL     AX,CL          ; Now ROM size in segments
        ADD     DX,AX          ; ...add base segment
        MOV     CL,4
        SHL     AX,CL          ; ROM address in bytes
        MOV     CX,AX          ; ...checksum requires CX
        CALL    CHK_01         ; Find ROM checksum
        JNZ     BADROM        ; ...bad ROM
        PUSH    DX
        MOV     Word ptr ES:67h,3 ; Offset for ROM being
setup
```

BIOS ASM

By Hicks

```

        MOV     ES:69h,DS           ; Segment for ROM being
setup
        CALL   Dword ptr ES:67h    ; ...call ROM
initialization
        POP    DX
        JMP    short FND_01

BADROM: OR     Byte ptr ES:15h,ER_ROM ; ROM present, bad checksum

NXTROM: ADD    DX,80h              ; Segment for next ROM

FND_01: CMP    DX,0F600h           ; End of ROM space
        JL     FNDROM              ; ...no, continue
        POP    DS
        IN     AL,21h              ; Read ic 8259 interrupt
mask
        AND    AL,10111100b        ; ...enable IRQ (0,1,6)
ints
        OUT    21h,AL              ;
(tod_clock,key,floppy_disk)

        MOV    AH,1
        MOV    CH,0F0h
        INT    10h                  ; Set cursor type
        CALL   BLANK                ; ...clear display
        PUSH   DS
        PUSH   CS
        POP    DS
        POP    ES
        TEST   Byte ptr ES:10h,1    ; Floppy disk present?
        JZ     FND_02              ; ...no
        CMP    Word ptr ES:72h,1234h ; Bios setup before?
        JNZ    CONFIG              ; ...no
FND_02: JMP    RESET               ; Else skip memory check

CONFIG: MOV    AX,41Ah              ; Where to move cursor
        MOV    SI,offset STUF       ; ...equipment message
        CALL   LOCATE              ; ...position cursor
        CALL   PRINT               ; ...and print string
        MOV    AX,51Bh              ; New cursor position
        MOV    SI,offset STUF_1     ; ...CR/LF
        CALL   Locate              ; ...position cursor
        CALL   PRINT               ; ...and print string
        TEST   Byte ptr ES:15h,11111111b ; Any error so far?
        JZ     VALID               ; ...no, skip
        CALL   PRINT               ; Print string
        MOV    AL,ES:15h            ; ...get error number
        CALL   NUMBER              ; ...print hex value
        CALL   PRINT               ; ...print prompt
        MOV    BL,4                 ; ...long beep
        CALL   BEEP
        CALL   GETCH                ; Wait for keypress
```

BIOS ASM

By Hicks

```

        PUSH    AX                ; ...save answer
        CALL    OUTCHR            ; ...echo answer
        POP     AX                ; ...get answer
        CMP     AL,'Y'            ; Was it "Y"
        JZ      FND_02           ; ...ok, continue
        CMP     AL,'y'            ; Was it "y"
        JZ      FND_02           ; ...ok, continue
        JMPF    0F000h,COLD      ; Else cold reset

VALID:  MOV     SI,offset STUF_2  ; No errors found, load
banner
        CALL    PRINT            ; ...and print string
        MOV     AX,81Ch          ; Where to move cursor
        CALL    LOCATE           ; ...position cursor
        CALL    PRINT            ; ...and print string
        MOV     AX,91Ch          ; Where to move cursor
        CALL    LOCATE           ; ...position cursor
        MOV     BL,17h           ; Character count

FENCE:  MOV     AL,'-'           ; Load ascii minus
        CALL    OUTCHR            ; ...and print it
        DEC     BL
        JNZ     FENCE
        MOV     AX,0A21h         ; Where to move cursor
        CALL    LOCATE           ; ...position cursor
        MOV     AL,ES:49h        ; Get CRT mode
        CMP     AL,7
        JZ      FEN_01           ; ...monochrome
        MOV     SI,offset STUF_3 ; ...color/graphics

FEN_01: CALL    PRINT            ; Print the string
        MOV     BX,0B21h         ;
        MOV     AL,ES:11h        ; Get equipment byte
        PUSH    AX
        MOV     CL,6
        ROR    AL,CL
        AND     AL,3             ; Number of printers
        JZ      FEN_02
        MOV     BP,8
        MOV     SI,offset STUF_4
        CALL    FAO              ; Formatted ascii output

FEN_02: POP     AX                ; Equipment byte restore
        MOV     SI,offset STUF_5 ; ...game controller
        PUSH    AX              ; Save a copy of equip.
byte
        TEST    AL,00010000b
        JZ      NO_TOY          ; Jump if no game
controller
        MOV     AX,BX
        CALL    LOCATE           ; Position cursor
        CALL    PRINT            ; ...and print string
```

BIOS ASM

By Hicks

```

        INC     BH                ; ...scroll line

NO_TOY: CALL    TIMER            ; Timer devices?
        JB     NO_TIM           ; ...skip if none
        MOV    AX,BX
        CALL   LOCATE           ; Position cursor
        INC    BH
        MOV    SI,offset STUF_8
        CALL   PRINT

NO_TIM: POP     AX
        MOV    SI,offset STUF_6
        ROR   AL,1              ; Check for COM port
        AND   AL,3
        JZ    NO_COM           ; ...skip if no com
        XOR   BP,BP
        CALL   FAO             ; Formatted ascii output

NO_COM: MOV     AX,121Ch        ; Where to position cursor
        CALL   LOCATE           ; ...position cursor
        MOV    SI,offset STUF_7 ; Memory size string
        CALL   PRINT           ; ...print string
        PUSH  ES
        MOV    BP,ES:13h       ; Memory size (1 K blocks)
        DEC   BP
        DEC   BP
        MOV    SI,2
        MOV    DX,SI
        MOV    AX,80h
        MOV    ES,AX

CUTE:   MOV     AX,122Bh        ; Cursory check of memory
        CALL   LOCATE           ; ...position cursor
        CALL   K_BYTE           ; ...print size in K
        CALL   MEMTST          ; Memory check ES:0 -
ES:0400
        JB     BADRAM          ; ...bad RAM found
(How ???)
        DEC   BP
        JNZ   CUTE
        POP   ES

RESET:  MOV     BL,2           ; Do a warm boot
        CALL   BEEP            ; ...short beep
        CALL   BLANK           ; ...clear display
        MOV    Word ptr ES:72h,1234h ; Show cold start done
        MOV    AH,1
        MOV    CX,607h         ; Set underline cursor
        INT   10h
        MOV    SI,offset BANNER ; Load banner address
        CALL   PRINT           ; ...and print string
        INT   19h             ; Boot the machine
```

BIOS ASM

By Hicks

```
BADRAM: POP      ES
          OR      Byte ptr ES:15h,ER_RAM      ; Show "Bad Ram" error
          JMP     CONFIG

STUF      db      ' Generic Turbo XT Bios 1987',0
STUF_1    db      CR,LF,0,'System error #',0,',', 'Continue?',0
STUF_2    db      ' ',0,'Interface card list',0,'Monochrome',0
STUF_3    db      'Color/Graphics',0
STUF_4    db      'Printer #',0
STUF_5    db      'Game controller',0
STUF_6    db      'Async. commu. #',0
STUF_7    db      'RAM Testing .. 000 KB',0
STUF_8    db      'Timer',0

          ENTRY   0E600h                      ; Not necessary to IPL
here..

IPL:      STI                                ; Called to reboot computer
          XOR     AX,AX
          MOV     DS,AX
          MOV     Word ptr DS:78h,offset INT_1E ; Get disk parameter table
          MOV     DS:7Ah,CS                    ; ...save segment
          MOV     AX,4                          ; Try up to four times

RETRY:    PUSH   AX                          ; Save retry count
          MOV     AH,0                          ; ...reset
          INT     13h                          ; ...floppy
          JB     FAILED
          MOV     AL,1                          ; One sector
          MOV     AH,2                          ; ...read
          XOR     DX,DX                          ; ...from drive 0, head 0
          MOV     ES,DX                          ; ...segment 0
          MOV     BX,7C00h                       ; ...offset 7C00
          MOV     CL,1                          ; ...sector 1
          MOV     CH,0                          ; ...track 0
          INT     13h                          ; ...floppy
          JB     FAILED
          JMPF    0000h,7C00h                   ; Call the boot block
;
FAILED:   POP     AX                          ; Get retries
          DEC     AL                          ; ...one less
          JNZ    RETRY

NODISK:   OR      AH,AH                      ; Disk present?
          JNZ    DERROR                        ; ...yes
          CALL   BLANK                          ; Clear display
          PUSH   CS
          POP    DS
          MOV     SI,offset DSKMSG              ; Load disk message
          CALL   PRINT                          ; ...and print string
          CALL   GETCH                          ; ...wait for keypress
```

BIOS ASM

By Hicks

```
CALL    BLANK                ; ...clear display
MOV     AX,0FF04h           ; Reset retry count
JMP     RETRY                ; ...and retry

DERROR: XOR     AX,AX        ; Error from NEC 765
MOV     DS,AX
LES     AX,Dword ptr DS:60h ; ROM basic vector ES:AX
MOV     BX,ES                ; ...get ROM basic segment
CMP     AX,0
MOV     AX,0
JNZ     NODISK                ; No ROM basic found
CMP     BX,0F600h
JNZ     NODISK                ; Invalid ROM basic segment
INT     18h                  ; ...else call ROM basic

DSKMSG  db      'Insert diskette in DRIVE A.',CR,LF
        db      '    Press any key.',0

        ENTRY   0E6F2h       ; IBM entry point for INT
19h

INT_19: JMP     IPL          ; Warm boot

        ENTRY   0E729h       ; IBM entry point for INT
14h

BAUD    dw      0417h        ; 110 baud clock divisor
        dw      0300h        ; 150 baud clock divisor
        dw      0180h        ; 300 baud clock divisor
        dw      00C0h        ; 600 baud clock divisor
        dw      0060h        ; 1200 baud clock divisor
        dw      0030h        ; 2400 baud clock divisor
        dw      0018h        ; 4800 baud clock divisor
        dw      000Ch        ; 9600 baud clock divisor

INT_14: STI                ; Serial com. RS232
services
        PUSH    DS          ; ...thru IC 8250 uart
(ugh)
        PUSH    DX          ; ...DX = COM device (0 -
3)
        PUSH    SI
        PUSH    DI
        PUSH    CX
        PUSH    BX
        MOV     BX,40h
        MOV     DS,BX
        MOV     DI,DX
        MOV     BX,DX      ; RS232 serial COM index
(0-3)
        SHL     BX,1        ; ...index by bytes
```

BIOS ASM

By Hicks

```

number      MOV     DX,[BX]                ; Convert index to port
            OR      DX,DX                ; ...by indexing 40:0
            JZ      COM_ND              ; ...no such COM device,
exit        OR      AH,AH                ; Init on AH=0
            JZ      COMINI
            DEC     AH
            JZ      COMSND              ; Send on AH=1
            DEC     AH
            JZ      COMGET              ; Rcvd on AH=2
            DEC     AH
            JZ      COMSTS              ; Stat on AH=3
COM_ND:     POP     BX                    ; End of COM service
            POP     CX
            POP     DI
            POP     SI
            POP     DX
            POP     DS
            IRET
COMINI:     PUSH    AX                    ; Init COM port.  AL has
data        ; = (Word Length in Bits -
5)          ; +(1 iff two stop bits) *
4           ; +(1 iff parity enable) *
8           ; +(1 iff parity even ) *
16          ; +(BAUD: select 0-7 ) *
32
            MOV     BL,AL
            ADD     DX,3                  ; Line Control Register
(LCR)
            MOV     AL,80h                ; ...index RS232_BASE + 3
baud        OUT     DX,AL                ; Tell LCR to set (latch)
            MOV     CL,4
            ROL     BL,CL                ; Baud rate selects by
words
            AND     BX,00001110b         ; ...mask off extraneous
            MOV     AX,Word ptr CS:[BX+BAUD] ; Clock divisor in AX
            SUB     DX,3                  ; Load in lo order baud
rate
            OUT     DX,AL                ; ...index RS232_BASE + 0
            INC     DX                    ; Load in hi order baud
rate
            MOV     AL,AH
            OUT     DX,AL                ; ...index RS232_BASE + 1
```

BIOS ASM

By Hicks

```

        POP     AX
        INC     DX                ; Find Line Control
Register
        INC     DX                ; ...index RS232_BASE + 3
        AND     AL,00011111b      ; Mask out the baud rate
        OUT     DX,AL            ; ...set (censored) init
stat
        MOV     AL,0
        DEC     DX                ; Interrupt Enable Reg.
(IER)
        DEC     DX                ; ...index RS232_BASE + 1
        OUT     DX,AL            ; Interrupt is disabled
        DEC     DX
        JMP     short COMSTS      ; Return current status

COMSND: PUSH    AX                ; Send AL thru COM port
        MOV     AL,3
        MOV     BH,00110000b      ;(Data Set Ready,Clear To
Send)
        MOV     BL,00100000b      ; ..(Data Terminal Ready)
wait
        CALL    WAITFR           ; Wait for transmitter to
idle
        JNZ     HUNG             ; ...time-out error
        SUB     DX,5             ; ... (xmit) index
RS232_BASE
        POP     CX                ; Restore char to CL
register
        MOV     AL,CL            ; ...get copy to load in
uart
        OUT     DX,AL            ; ...transmit char to IC
8250
        JMP     COM_ND           ; ...AH register has
status
HUNG:   POP     CX                ; Transmit error, restore
char
        MOV     AL,CL            ; ...in AL for
compatibility
        ; ...fall thru to gen.
error
HUNGG:  OR      AH,80h           ; Set error (=sign) bit in
AH
        JMP     COM_ND           ; ...common exit

COMGET: MOV     AL,1             ; Get char. from COM port
        MOV     BH,00100000b      ; Wait on DSR (Data Set
Ready)
        MOV     BL,00000001b      ; Wait on DTR (Data
Term.Ready)
        CALL    WAITFR           ; ...wait for character
        JNZ     HUNGG           ; ...time-out error
```

BIOS ASM

By Hicks

```

        AND     AH,00011110b      ; Mask AH for error bits
        SUB     DX,5              ; ... (rcvr) index
RS232_BASE
        IN      AL,DX             ; Read the character
        JMP     COM_ND           ; ...AH register has
status

COMSTS: ADD     DX,5              ; Calculate line control
stat
        IN      AL,DX            ; ...index RS232_BASE + 5
        MOV     AH,AL           ; ...save high order
status
        INC     DX              ; Calculate modem stat. reg.
        IN      AL,DX           ; ...index RS232_BASE + 6
        JMP     COM_ND         ; ...save low order
status

1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
idle )*16384
error)*32768

POLL:   MOV     BL,byte ptr [DI+7Ch] ; Wait on BH in status or
error
```

BIOS ASM

By Hicks

```
POLL_1: SUB      CX,CX          ; Outer delay loop
POLL_2: IN       AL,DX         ; ... inner loop
      MOV      AH,AL
      AND      AL,BH          ; And status with user BH
mask
      CMP      AL,BH
      JZ       POLLXT        ; ... jump if mask set
      LOOP    POLL_2        ; Else try again
      DEC     BL
      JNZ     POLL_1
      OR      BH,BH          ; Clear mask to show
timeout
POLLXT: RET                ; Exit AH reg. Z flag
status
WAITFR: ADD     DX,4          ; Reset the Modem Control
Reg.      OUT   DX,AL         ; ...index RS232_BASE + 4
      INC     DX             ; Calculate Modem Status
Reg.      INC   DX           ; ...index RS232_BASE + 6
      PUSH   BX             ; Save masks
(BH=MSR,BL=LSR)
      CALL  POLL            ; ...wait on MSR modem
status
      POP   BX              ; ...restore wait masks
BH,BL
      JNZ  WAITF1           ; ..."Error Somewhere" by
DEC
      DEC   DX              ; Calculate Line Status Reg.
      MOV  BH,BL            ; ...index RS232_BASE + 5
      CALL POLL            ; ...wait on LSR line
status
WAITF1: RET                ; Status in AH reg. and Z
flag
      ENTRY 0E82Eh         ; IBM entry, key bios
service
INT_16: STI                ; Keyboard bios services
      PUSH  DS
      PUSH  BX
      MOV   BX,40h
      MOV   DS,BX          ; Load work segment
      OR    AH,AH
      JZ    KPD_RD         ; Read keyboard buffer,
AH=0
      DEC   AH
```

BIOS ASM

By Hicks

```

AH=1      JZ      KPD_WT      ; Set Z if char ready,
          DEC      AH
          JZ      KPD_SH      ; Return shift in AL ,
AH=2

KPD_XT:   POP      BX      ; Exit INT_16 keypad
service   POP      DS
          IRET

KPD_RD:   CLI      ; No interrupts, alters
buffer    MOV      BX,DS:1Ah ; ...point to buffer head
          CMP      BX,DS:1Ch ; If not equal to buffer
tail      JNZ      KPD_R1    ; ...char waiting to be
read      STI      ; Else allow interrupts
          JMP      KPD_RD    ; ...wait for him to type

KPD_R1:   MOV      AX,[BX]   ; Fetch the character
character INC      BX      ; ...point to next
shift     INC      BX      ; ...char = scan code +
          MOV      DS:1Ah,BX ; Save position in head
          CMP      BX,DS:82h ; ...buffer overflowed?
          JNZ      KPD_XT    ; ...no, done
          MOV      BX,DS:80h ; Else reset to point at
start     MOV      DS:1Ah,BX ; ...and correct head
position  JMP      KPD_XT

KPD_WT:   CLI      ; No interrupts, critical
code      MOV      BX,DS:1Ah ; ...point to buffer head
          CMP      BX,DS:1Ch ; ...equal buffer tail?
          MOV      AX,[BX]   ; (fetch, look ahead)
          STI      ; Enable interrupts
          POP      BX
          POP      DS
          RETF      2      ; Do IRET, preserve flags

KPD_SH:   MOV      AL,DS:17h ; Read keypad shift status
          JMP      KPD_XT

place     ENTRY    0E885h    ; Align INT_9 at correct

```

BIOS ASM

By Hicks

```
ASCII    db      000h,037h,02Eh,020h          ; Scan -> Ascii.  Sign bit
set
        db      02Fh,030h,031h,021h          ; ...if further work
needed
        db      032h,033h,034h,035h
        db      022h,036h,038h,03Eh
        db      011h,017h,005h,012h
        db      014h,019h,015h,009h
        db      00Fh,010h,039h,03Ah
        db      03Bh,084h,001h,013h
        db      004h,006h,007h,008h
        db      00Ah,00Bh,00Ch,03Fh
        db      040h,041h,082h,03Ch
        db      01Ah,018h,003h,016h
        db      002h,00Eh,00Dh,042h
        db      043h,044h,081h,03Dh
        db      088h,02Dh,0C0h,023h
        db      024h,025h,026h,027h
        db      028h,029h,02Ah,02Bh
        db      02Ch,0A0h,090h

NOALFA   db      032h,036h,02Dh,0BBh          ; Non-Alphabetic secondary
        db      0BCh,0BDh,0BEh,0BFh          ; ...translation table
        db      0C0h,0C1h,0C2h,0C3h
        db      0C4h,020h,031h,033h
        db      034h,035h,037h,038h
        db      039h,030h,03Dh,01Bh
        db      008h,05Bh,05Dh,00Dh
        db      05Ch,02Ah,009h,03Bh
        db      027h,060h,02Ch,02Eh
        db      02Fh

CTRLUP   db      040h,05Eh,05Fh,0D4h          ; CTRL uppercase secondary
        db      0D5h,0D6h,0D7h,0D8h          ; ...translation table
        db      0D9h,0DAh,0DBh,0DCh          ; ...for non-ASCII control
        db      0DDh,020h,021h,023h
        db      024h,025h,026h,02Ah
        db      028h,029h,02Bh,01Bh
        db      008h,07Bh,07Dh,00Dh
        db      07Ch,005h,08Fh,03Ah
        db      022h,07Eh,03Ch,03Eh
        db      03Fh

CTRLLO   db      003h,01Eh,01Fh,0DEh          ; CTRL lowercase secondary
        db      0DFh,0E0h,0E1h,0E2h          ; ...translation table
        db      0E3h,0E4h,0E5h,0E6h          ; ...for non-ASCII control
        db      0E7h,020h,005h,005h
        db      005h,005h,005h,005h
        db      005h,005h,005h,01Bh
        db      07Fh,01Bh,01Dh,00Ah
        db      01Ch,0F2h,005h,005h
        db      005h,005h,005h,005h
```

BIOS ASM

By Hicks

```

        db          005h

ALTKEY  db          0F9h,0FDh,002h,0E8h          ; ALT key secondary
        db          0E9h,0EAh,0EBh,0ECh          ; ...translation table
        db          0EDh,0EEh,0EFh,0F0h
        db          0F1h,020h,0F8h,0FAh
        db          0FBh,0FCh,0FEh,0FFh
        db          000h,001h,003h,005h
        db          005h,005h,005h,005h
        db          005h,005h,005h,005h
        db          005h,005h,005h,005h
        db          005h

NUMPAD  db          '789-456+1230.'              ; Keypad secondary
tralsator

NUMCTR  db          0F7h,005h,004h,005h          ; Numeric keypad CTRL sec.
        db          0F3h,005h,0F4h,005h          ; ...translation table
        db          0F5h,005h,0F6h,005h
        db          005h

NUMUPP  db          0C7h,0C8h,0C9h,02Dh          ; Numeric keypad SHIFT sec.
        db          0CBh,005h,0CDh,02Bh          ; ...translation table
        db          0CFh,0D0h,0D1h,0D2h
        db          0D3h

INT_9:  STI                          ; Key press hardware
interrupt

        PUSH       AX
        PUSH       BX
        PUSH       CX
        PUSH       DX
        PUSH       SI
        PUSH       DI
        PUSH       DS
        PUSH       ES
        CLD
        MOV        AX,40h
        MOV        DS,AX
        IN         AL,60h                ; Read the scan code data
        PUSH       AX                    ; ...save it
        IN         AL,61h                ; Get control port status
        PUSH       AX                    ; ...save it
        OR         AL,10000000b          ; Set "latch" bit to
        OUT        61h,AL                ; ...acknowledge data
        POP        AX                    ; Restore control status
        OUT        61h,AL                ; ...to enable keyboard
        POP        AX                    ; ...restore scan code
        MOV        AH,AL                 ; Save copy of scan code
        CMP        AL,11111111b          ; ...check for overrun
        JNZ        KY_01                 ; ...no, OK
        JMP        KY_BEP                ; Else beep bell on overrun

```

BIOS ASM

By Hicks

```
KY_EOI: MOV     AL,20h           ; Send end_of_interrupt
code
        OUT     20h,AL         ; ...to 8259 interrupt
chip

KY_XIT: POP     ES             ; Exit the interrupt
        POP     DS
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        IRET

KY_01:  AND     AL,01111111b    ; Valid scan code, no break
        CMP     AL,46h
        JBE     KY_02
        JMP     KY_CT8

KY_02:  MOV     BX,offset ASCII ; Table for ESC thru Scroll
Lck
        XLAT   CS:[BX]         ; ...translate to Ascii
        OR     AL,AL          ; Sign flags "Shift" type
key
        JS     KY_FLG         ; ...shift,caps,num,scroll
etc
        OR     AH,AH          ; Invalid scan code?
        JS     KY_EOI         ; ...exit if so
        JMP    short  KY_ASC   ; Else normal character

KY_FLG: AND     AL,01111111b    ; Remove sign flag bit
        OR     AH,AH          ; ...check scan code
        JS     KY_SUP         ; ...negative, key
released
        CMP     AL,10h        ; Is it a "toggle" type
key?
        JNB    KY_TOG         ; ...yes
        OR     DS:17h,AL      ; Else set bit in "flag"
byte
        JMP     KY_EOI        ; ...and exit

KY_TOG: TEST    Byte ptr DS:17h,00000100b ; Control key pressed?
        JNZ    KY_ASC         ; ...yes, skip
        TEST   AL,DS:18h     ; Else check "CAPS, NUM,
SCRL"
        JNZ    KY_EOI         ; ...set, invalid, exit
        OR     DS:18h,AL     ; Show set in "flag_1" byte
        XOR    DS:17h,AL     ; ...flip bits in "flag"
byte
        JMP     KY_EOI
```

BIOS ASM

By Hicks

```
KY_SUP: CMP      AL,10h                ; Released - is it "toggle"
key
      JNB      KY_TUP                ; ...skip if so
      NOT      AL                    ; Else form two's
complement
      AND      DS:17h,AL              ; ...to do BIT_CLEAR
"flags"
      CMP      AL,11110111b           ; ALT key release special
case
      JNZ      KY_EOI                ; ...no, exit
      MOV      AL,DS:19h              ; Else get ALT-keypad
character
      MOV      AH,0                  ; ...pretend null scan
code
      MOV      DS:19h,AH              ; ...zero ALT-keypad
character
      CMP      AL,AH                  ; Was there a valid ALT-
keypad?
      JZ       KY_EOI                ; ...no, ignore, exit
      JMP      KY_NUL                ; Else stuff it in ASCII
buffer

KY_TUP: NOT      AL                    ; Form complement of toggle
key
      AND      DS:18h,AL              ; ...to do BIT_CLEAR
"flag_1"
      JMP      KY_EOI

KY_ASC: TEST     Byte ptr DS:18h,00001000b ; Scroll lock pressed?
      JZ       KY_NLK                ; ...no
      CMP      AH,45h                ; Is this a NUM LOCK
character?
      JZ       KY_03                  ; ...no
      AND      Byte ptr DS:18h,11110111b ; Else clear bits in
"flag_1"

KY_03:  JMP      KY_EOI                ; ...and exit

KY_NLK: TEST     Byte ptr DS:17h,00001000b ; ALT key pressed?
      JNZ      KY_ALT                ; ...yes
      TEST     Byte ptr DS:17h,00000100b ; CTRL key pressed?
      JNZ      KY_CTL                ; ...yes
      TEST     Byte ptr DS:17h,00000011b ; Either shift key pressed?
      JNZ      KSHIFT                ; ...yes

KY_LC:  CMP      AL,1Ah                ; Alphabetic character?
      JA       KY_LC1                ; ...no
      ADD      AL,'a'-1                ; Else add lower case base
      JMP      KY_COM

KY_LC1: MOV      BX,offset NOALFA      ; Non-alphabetic character
```

BIOS ASM

By Hicks

```

        SUB     AL,20h
        XLAT   CS:[BX]           ; ...do the xlate
        JMP     KY_COM

KY_ALT:  CMP     AL,1Ah           ; Control key pressed?
        JA     KY_AGN           ; ...no, skip
        MOV     AL,0             ; Else illegal key press
        JMP     KY_BFR

KY_AGN:  MOV     BX,offset ALTKEY ; Load ALT key translation
        SUB     AL,20h           ; ...bias to printing char.
        XLAT   CS:[BX]           ; ...do the translation
        JMP     KY_COM

KY_CTL:  CMP     AH,46h           ; Scroll lock key?
        JNZ    KY_CT1           ; ...no, skip
        MOV     Byte ptr DS:71h,10000000b ; Else CTRL-"Scroll" =
break
        MOV     AX,DS:80h        ; ...get key buffer start
        MOV     DS:1Ch,AX        ; ...get key tail to start
        MOV     DS:1Ah,AX        ; ...get key head to start
        INT     1Bh              ; Issue a "Break" interrupt
        SUB     AX,AX
        JMP     KY_CO2

KY_CT1:  CMP     AH,45h           ; Num lock key?
        JNZ    KY_CT2           ; ...no, skip
        OR     Byte ptr DS:18h,00001000b ; Else show scroll lock
        MOV     AL,20h           ; ...send end_of_interrupt
        OUT    20h,AL           ; ...to 8259 int.
controller
        CMP     Byte ptr DS:49h,7 ; Monochrome monitor?
        JZ     KY_POL           ; ...yes, skip
        MOV     DX,3D8h          ; Else reset mode
        MOV     AL,DS:65h        ; ...for the
        OUT    DX,AL            ; ...CGA color card

KY_POL:  TEST    Byte ptr DS:18h,00001000b ; Wait for him to type
        JNZ    KY_POL           ; ...not yet
        JMP     KY_XIT

KY_CT2:  CMP     AH,3            ; Is it a Control @
        (null) ?
        JNZ    KY_CT3           ; ...no
        MOV     AL,0             ; Else force a null

KY_CT4:  JMP     KY_BFR           ; ...save in buffer

KY_CT3:  CMP     AL,1Ah           ; Is it a control
character?
        JBE    KY_CT4           ; ...yes
        MOV     BX,offset CTRLLO ; Else non-ascii control
```

BIOS ASM

By Hicks

```

        SUB     AL,20h                ; ...lower case
        XLAT   CS:[BX]               ; ...translation
        JMP     KY_COM

KSHIFT: CMP     AH,37h                ; Print_Screen pressed?
        JNZ    KY_CT5
        MOV    AL,20h                ; Yes, send
end_of_interrupt
        OUT    20h,AL                ; ...to 8259 interrupt
chip
        INT    5                     ; Request print_screen
service
        JMP    KY_XIT                ; ...and exit key service

KY_CT5: CMP     AL,1Ah                ; Alphabetic char?
        JA     KY_CT6                ; ...no
        ADD    AL,'A'-1              ; Yes, add base for
alphabet
        JMP    KY_COM

KY_CT6: MOV     BX,offset CTRLUP      ; Non-ascii control
        SUB    AL,20h                ; ...upper case
        XLAT   CS:[BX]               ; ...translation
        JMP    KY_COM

KY_CT8: SUB     AL,47h                ; Keypad key, convert
origin
        MOV    BL,DS:17h             ; ...get "flag" byte
        TEST   BL,00001000b          ; Look for ALT keypad entry
        JNZ    KB_NUM                ; ...do special entry
thing
        TEST   BL,00000100b          ; CTRL key pressed?
        JNZ    KY_CTR                ; ...skip if so
        TEST   BL,00100000b          ; Toggle "Num Lock" ?
        JZ     KY_CT9                ; ...no, continue
        TEST   BL,00000011b          ; Shift keys hit?
        JNZ    KY_CTA                ; ...no, check "INS"
        JMP    KY_CTD                ; Else xlat keypad char.

KY_CT9: TEST    BL,00000011b          ; Shift keys hit?
        JZ     KY_CTA                ; ...no, check "INS" key
        JMP    KY_CTD                ; Else xlat keypad char.

KB_NUM: OR      AH,AH                ; ALT-keypad entry, scan
code
        JS     KY_EO1                ; ...out of range
        TEST   Byte ptr DS:17h,00000100b ; Else check CTRL state
        JZ     KY_PAD                ; ...not pressed, ALT
keypad

KY_PAT: CMP     AH,53h                ; Patch for CTRL ALT -
toggle
```

BIOS ASM

By Hicks

```

        JNZ     KY_PA1                ; ...not a DEL (reset)
        MOV     Word ptr DS:72h,1234h ; Ctrl-Alt-Del, set init
flag
        JMP     WARM                  ; ...do a warm reboot

KY_PA1: CMP     AH,4Ah                ; Is it a keypad "-" ?
        JNZ     KY_PAD                ; ...no, skip
        PUSH   AX
        PUSH   BX
        PUSH   CX
        IN     AL,61h                 ; Read equipment flags
        XOR    AL,00001100b          ; ...toggle speed
        OUT    61h,AL                 ; Write new flags back

        MOV     AH,1                  ; Video func=Set cursor
type
        MOV     CX,607h               ; ...start at 6, end at 7
        AND    AL,4                   ; Is turbo mode set?
        JZ     KY_CUR                ; ...no, keep big cursor
        MOV     CH,0                  ; Else set tiny cursor

KY_CUR: INT     10h                  ; Set cursor type service
        MOV     BX,DS:80h            ; ...get start of key buf
        MOV     DS:1Ah,BX            ; ...set head to start
        MOV     DS:1Ch,BX            ; ...set tail to start
        POP    CX
        POP    BX
        POP    AX

KY_PAD: MOV     BX,offset NUMPAD      ; Get keypad translation
table
        XLAT   CS:[BX]               ; ...convert to number
        CMP    AL,'0'                ; Is it a valid ASCII
digit?
        JB     KY_EO1                ; ...no, ignore it
        SUB    AL,30h                 ; Else convert to number
        MOV    BL,AL                  ; ...save a copy
        MOV    AL,DS:19h              ; Get partial ALT-keypad
sum
        MOV    AH,0Ah                 ; ...times 10 (decimal)
        MUL   AH
        ADD   AL,BL                   ; Add in new digit to sum
        MOV   DS:19h,AL               ; ...save as new ALT entry

KY_EO1: JMP     KY_EOI                ; End_of_interrupt, exit

KY_CTR: OR     AH,AH                 ; Key released?
        JS     KY_EO1                ; ...ignore if so
        MOV    BX,offset NUMCTR      ; Else Numeric Keypad
Control
        XLAT   CS:[BX]               ; ...secondary translate
        JMP    short KY_COM           ; ...and save it
```

BIOS ASM

By Hicks

```
KY_CTA:  CMP     AH,0D2h           ; Was "INS" key released?
        JNZ     KY_CTB
        AND     Byte ptr DS:18h,01111111b ; Yes, clear "INS" in
"FLAG_1"
        JMP     short  KY_EO1

KY_CTB:  OR      AH,AH           ; Key released?
        JS      KY_EO1          ; ...ignore if so
        CMP     AH,52h         ; Else check for "INS"
press
        JNZ     KY_CTC          ; ...not "INS" press
        TEST    Byte ptr DS:18h,10000000b ; Was INS key in effect?
        JNZ     KY_EO1          ; ...yes, ignore
        XOR     Byte ptr DS:17h,10000000b ; Else tog "INS" in "FLAG"
byte
        OR      Byte ptr DS:18h,10000000b ; ...set "INS" in "FLAG_1"
byte

KY_CTC:  MOV     BX,offset NUMUPP ; Numeric Keypad Upper Case
        XLAT    CS:[BX]         ; ...secondary translation
        JMP     short  KY_COM

KY_CTD:  OR      AH,AH           ; Was the key released?
        JS      KY_EO1          ; ...yes, ignore
        MOV     BX,offset NUMPAD ; Load translation table
        XLAT    CS:[BX]         ; ...do translate
        JMP     short  KY_COM

KY_COM:  CMP     AL,5            ; Common entry, char in AL
        JZ      KY_EO2          ; ...Control E, ignore
        CMP     AL,4            ;
        JA      KY_CO1          ; Above Control D

        OR      AL,10000000b     ; Else set sign flag
        JMP     short  KY_CO2

KY_CO1:  TEST    AL,10000000b     ; Is sign bit set?
        JZ      KY_CO3          ; ...skip if so
        AND     AL,01111111b     ; Else mask sign off

KY_CO2:  MOV     AH,AL           ; Save in high order byte
        MOV     AL,0            ; ...set scan code to zero

KY_CO3:  TEST    Byte ptr DS:17h,01000000b ; Test for "CAPS LOCK"
state
        JZ      KY_BFR          ; ...no, skip
        TEST    Byte ptr DS:17h,00000011b ; Test for SHIFT key
        JZ      KY_CO4          ; ...skip if no shift
        CMP     AL,'A'          ; Check for alphabetic key
        JB      KY_BFR          ; ...not SHIFT_able
        CMP     AL,'Z'          ; Check for alphabetic key
```

BIOS ASM

By Hicks

```

        JA      KY_BFR                ; ...not SHIFT_able
        ADD     AL,20h                ; Else do the shift
        JMP     short   KY_BFR

KY_CO4: CMP     AL,'a'                ; Check for alphabetic key
        JB     KY_BFR                ; ...not SHIFT_able
        CMP     AL,'z'                ; Check for Alphabetic key
        JA     KY_BFR                ; ...not SHIFT_able
        SUB     AL,20h                ; Else do the shift

KY_BFR: MOV     BX,DS:1Ch             ; BX = tail of buffer
        MOV     DI,BX                ; ...save it
        INC     BX                    ; ...advance
        INC     BX                    ; ...by word
        CMP     BX,DS:82h            ; End of buffer reached?
        JNZ     KY_CHK               ; ...no, skip
        MOV     BX,DS:80h            ; Else BX = beginning of
buffer

KY_CHK: CMP     BX,DS:1Ah            ; BX = Buffer Head ?
        JNZ     KY_STF               ; ...no, OK
        JMP     short   KY_BEP       ; Else buffer overrun, beep

KY_STF: MOV     [DI],AX              ; Stuff scan code, char in
bfr
        MOV     DS:1Ch,BX            ; ...and update bfr tail

KY_EO2: JMP     KY_EOI

KY_BEP: MOV     AL,20h                ; Keyboard beeper routine
        OUT     20h,AL               ; ...send end_of_interrupt
        MOV     BX,80h                ; Cycles in beep
        IN     AL,61h                ; ...get status
        PUSH    AX                    ; ...save copy

KY_BE1: AND     AL,11111100b         ; Mask off speaker bits
        OUT     61h,AL               ; ...disable speaker
KY_BE2: MOV     CX,64h                ; Constant for pitch
KY_BE3: LOOP    KY_BE3               ; ...delay, speaker off
        XOR     AL,00000010b         ; Toggle speaker position
        OUT     61h,AL               ; Full cycle done yet?
        TEST    AL,00000010b         ; ...no, do other half
cycle
        JZ     KY_BE2

        DEC     BX                    ; Else show cycle sent
        JNZ     KY_BE1               ; ...more cycles to send
        POP     AX
        OUT     61h,AL               ; Restore flags
        MOV     CX,32h                ; Silence counter
KY_BE4: LOOP    KY_BE4               ; Send nothing for while
        JMP     KY_XIT
```

BIOS ASM

By Hicks

```
KY_NUL: MOV     AH,38h           ; ALT key pressed, released
        JMP     KY_BFR         ; ...for no logical reason

        ENTRY   0EC59h        ; IBM entry point for
floppy

INT_13: STI                     ; Floppy disk services
        PUSH    BP
        PUSH    SI
        PUSH    DI
        PUSH    DS
        PUSH    ES
        PUSH    BX
        MOV     DI,AX          ; Request type in DI, for
index
        XOR     AX,AX
        MOV     DS,AX
        LES     SI,Dword ptr DS:78h ; Get disk parameter table
        MOV     AX,40h
        MOV     DS,AX
        MOV     BX,5
        MOV     AX,ES:[BX+SI]    ; Get (Gap Length, DTL) in
AX
        PUSH    AX            ; ...save it
        DEC     BX
        DEC     BX
        MOV     AX,ES:[BX+SI]  ; Get (Bytes/sector,EOT) in
AX
        PUSH    AX            ; ...save it
        XCHG   CL,DH
        XCHG   DL,CL
        PUSH    DX            ; Push (Head,Drive) swapped
        PUSH    CX
        PUSH    DI
        MOV     BP,SP         ; Mark bottom of stack
frame
ifdef   SLOW_FLOPPY
        CALL    FD_SPD        ; ...execute request lo
speed
else
        CALL    FD_XQT        ; ...execute at current
speed
endif
        MOV     AH,ES:[SI+2]   ; Get new motor count
        MOV     DS:40h,AH     ; ...and save it
        MOV     AH,DS:41h    ; Get completion status
        CMP     AH,1         ; ...check for write
protect
        CMC                     ; ...was write protect
error
        POP     BX
        POP     CX
```

BIOS ASM

By Hicks

```
POP      DX
XCHG    DL,CL
XCHG    CL,DH
POP      BX          ; Clean
POP      BX          ; ...up
POP      BX          ; ...stack
POP      ES
POP      DS
POP      DI
POP      SI
POP      BP
RETF    2

FD_XQT:  MOV     AL,[BP+1]          ; Get floppy service number
OR      AL,AL
JZ      FD_RST          ; ...reset, AH=0
DEC     AL
JZ      FD_XQ3          ; ...read status, AH=1
CMP     Byte ptr [BP+2],3        ; For track number above 3?
JA      FD_XQ1          ; ...yes
CMP     AL,5            ; Service within range?
JBE     FD_XQ2          ; ...yes

FD_XQ1:  MOV     Byte ptr DS:41h,1 ; Say write protect error
RET

FD_XQ2:  JMP     FD_001          ; Execute legal service

FD_XQ3:  MOV     AL,DS:41h        ; Return NEC status byte
RET

FD_RST:  MOV     DX,3F2h         ; Reset the floppy disk
system

CLI
AND     Byte ptr DS:3Fh,00001111b ; Clear "write in progress"
MOV     AL,DS:3Fh            ; ...find out busy drives
MOV     CL,4
SHL     AL,CL
TEST    AL,00100000b
JNZ     FD_RS1              ; Drive #1 active
TEST    AL,01000000b
JNZ     FD_RS2              ; Drive #2 active
TEST    AL,10000000b
JZ      FD_RS0              ; Drive #3 idle

FD_RS3:  INC     AL
FD_RS2:  INC     AL
FD_RS1:  INC     AL

FD_RS0:  MOV     Byte ptr DS:3Eh,0 ; All drives need
recalibrate
MOV     Byte ptr DS:41h,0        ; ...no completion status
```

BIOS ASM

By Hicks

```

        OR      AL,00001000b          ; Interrupt ON in command
word
        OUT     DX,AL                 ; ...send word to
controller
        OR      AL,00000100b          ; "Reset" in command word
        OUT     DX,AL                 ; ...send word to
controller
        STI
        CALL    NC_BSY                 ; Wait for completion
        CALL    NC_STS                 ; ...read result block
        MOV     AL,DS:42h
        CMP     AL,0C0h                ; Did the reset work
        JZ      FD_RS4                 ; ...yes
        MOV     Byte ptr DS:41h,20h    ; Else set controller error
        JMP     short  FD_RS5          ; ...return

FD_RS4: MOV     AL,3                   ; Specify command to NEC
        CALL    NEC765                 ; ...send it
        MOV     AL,ES:[SI]             ; First byte in param block
        CALL    NEC765                 ; ...send it
        MOV     AL,ES:[SI+1]          ; Secnd byte in param block
        CALL    NEC765                 ; ...send it

FD_RS5: RET

NECFUN  db      003h,000h,0E6h,0C5h,0E6h,04Dh ; NEC function table lookup
NECDMA  db      000h,000h,046h,04Ah,042h,04Ah ; DMA modes for 8237
NECWRT  db      000h,000h,000h,080h,000h,080h ; Write flag table lookup
NECDRV  db      1,2,4,8                ; Drive number table lookup
NECERR  db      80h,20h,10h,4,2,1      ; Error code table lookup
NECSTS  db      04h,10h,08h,04h,03h,02h,20h ; Disk status table lookup

FD_001: CLI                             ; Normal (non-reset)
commands
        MOV     Byte ptr DS:41h,0      ; ...reset status
        MOV     AL,[BP+1]              ; Get command word
        MOV     AH,0
        MOV     DI,AX                  ; Save copy, zero-extended
        OUT     0Ch,AL                 ; ...diddle LSB/MSB flip-
flop
        MOV     AL,CS:[DI+NECDMA]      ; Fetch DMA mode
        OUT     0Bh,AL                 ; ...send it to IC8237
        MOV     AX,[BP+0Ch]            ; Get segment address
        MOV     CL,4                   ; ...convert
        ROL     AX,CL                  ; ...to (offset, 64K page
no)
        MOV     CH,AL                  ; Extract page number (0-
15.)
        AND     CH,00001111b           ; ...for 8237 dma
controller
        AND     AL,11110000b           ; Extract implicit page
offset
```

BIOS ASM

By Hicks

```

        ADD     AX,[BP+0Ah]           ; ...add explicit user
offset
        ADC     CH,0                 ; ...(page number
overflowed)
        MOV     DX,AX                ; Now save lo 16 bits of
addr.
        OUT     4,AL                 ; ...send lowest 8 bits "
"
        MOV     AL,AH
        OUT     4,AL                 ; ...send next 8 bits "
"
        MOV     AL,CH
        OUT     81h,AL               ; 64K page no to DMA page
reg
        MOV     AH,[BP+0]
        MOV     AL,0
        SHR     AX,1                 ; Sector cnt * 128
        MOV     CL,[BP+6]           ; Track count
        SHL     AX,CL                ; * sector count
        DEC     AX                   ; - 1
        OUT     5,AL                 ; Send 1/2 of the word
count
        XCHG    AL,AH
        OUT     5,AL                 ; Send 2/2 of the word
count
        XCHG    AL,AH
        ADD     AX,DX                ; Compute final address
        JNB     FD_002              ; ...ok
        STI
        MOV     Byte ptr DS:41h,9h  ; Else wrapped around 64K
byte
        JMP     FD_64K              ; ...page register

FD_002: MOV     AL,2                 ; Disable floppy disk dma
        OUT     0Ah,AL
        MOV     Byte ptr DS:40h,0FFh ; Set large motor timeout
        MOV     BL,[BP+2]           ; ...get drive number
        MOV     BH,0
        MOV     AL,CS:[BX+NECDRV]   ; Table lookup bit position
        MOV     CH,AL               ; ...save mask
        MOV     CL,4
        SHL     AL,CL                ; Shift mask into place
        OR     AL,BL                 ; ...or in drive select
        OR     AL,0Ch                ; ...or in DMA and NO
RESET
        MOV     DX,3F2h
        OUT     DX,AL                ; Send to floppy control
port
        STI
        MOV     AL,CS:[DI+NECWRT]   ; Table lookup for write
flag
```

BIOS ASM

By Hicks

```

active    OR      DS:3Fh,AL          ; ...set write flag if
         OR      AL,AL
         JNS     FD_003             ; ...skip if non-write
         MOV     AH,ES:[SI+0Ah]    ; Motor start from param
blk
         OR      AH,AH
         JZ      FD_003             ; ...none specified
         TEST    CH,DS:3Fh        ; Was this drive motor
running?  JNZ     FD_003             ; ...skip if so
         CALL    FD_WT1            ; Else delay for motor
start
FD_003:  OR      DS:3Fh,CH          ; Show this motor is
running  TEST    CH,DS:3Eh        ; Drive recalibration
needed?  JNZ     FD_004             ; ...no, skip
         OR      DS:3Eh,CH        ; Else show recalibrated
         MOV     AL,7              ; Send RECAL command
         CALL    NEC765            ; ...to NEC 765 chip
         MOV     AL,BL
         CALL    NEC765            ; ...drive number
         CALL    NC_BSY            ; Wait for completion of
RECAL    CALL    NEC_04            ; ...dummy call to RET
FD_004:  MOV     AL,0Fh            ; Request a seek
         CALL    NEC765            ; ...from the NEC 765
         MOV     AL,BL
         CALL    NEC765            ; Drive number
         MOV     AL,[BP+3]
         CALL    NEC765            ; Cylinder number
         CALL    NC_BSY            ; ...wait for completion
         CALL    NC_STS            ; ...read results
         MOV     AL,ES:[SI+9]     ; Get head settle time
         OR      AL,AL             ; ...none specified?
         JZ      FD_005            ; ...if none, skip
FD_STL:  MOV     CX,226h           ; Delay time for head
settle
FD_STZ:  LOOP    FD_STZ            ; ...timed wait
         DEC     AL                ; ...delay in millisec
         JNZ    FD_STL            ; ...wait some more
FD_005:  MOV     AL,CS:[DI+NECFUN] ; Translate user service,
then     CALL    NEC765            ; ...and send as NEC func
         MOV     AL,[BP+4]
         AND     AL,1
```

BIOS ASM

By Hicks

```
        SHL     AL,1
        SHL     AL,1
        OR      AL,BL
        CALL    NEC765
        CMP     Byte ptr [BP+1],5           ; Is this a format request?
        JNZ     FD_006                     ; ...skip if not
        MOV     AL,[BP+6]                 ; Else use user
bytes/sector
        CALL    NEC765
        MOV     AL,[BP+7]                 ; ... user EOT
        CALL    NEC765
        MOV     AL,ES:[SI+7]             ; Disk table format gap
length
        CALL    NEC765
        MOV     AL,ES:[SI+8]             ; Disk table format fill
byte
        CALL    NEC765
        JMP     short   FD_008

FD_006: MOV     CX,7                       ; Else lookup bytes *
512/sec
        MOV     DI,3                       ; ...from disk table

FD_007: MOV     AL,[BP+DI]                 ; AL has bytes/sector * 512
        CALL    NEC765
        INC     DI                         ; ...get next item for
table
        LOOP   FD_007                     ; ...also (EOT,GAP,DTL...)

FD_008: CALL    NC_BSY                     ; Wait on floppy i/o
completion
        CALL    NC_ST1                     ; ...get NEC status
        MOV     AL,DS:42h                  ; ...into AL
        AND     AL,11000000b              ; Isolate errors
        JZ      FD_012                     ; ...no errors
        CMP     AL,40h                     ; Test direction bit
        JZ      FD_ERR                    ;
        MOV     Byte ptr DS:41h,20h       ; Set if bad controller
        JMP     short   FD_012             ; ...return error

FD_ERR: MOV     AL,DS:43h                  ; Read return code from
block
        MOV     CX,6                       ; ...number of error types
        XOR     BX,BX                       ; Start at error type 0

FD_009: TEST    AL,CS:[BX+NECERR]         ; Has error type BX
occured?
        JNZ     FD_010                     ; ...yes
        INC     BX                         ; Else try next error type
        LOOP   FD_009                     ; ...until done
```

BIOS ASM

By Hicks

```
FD_010: MOV     AL,CS:[BX+NECSTS]           ; Translate error code
again
        MOV     DS:41h,AL                 ; ...store it as disk
status

FD_012: MOV     AL,DS:45h                 ; Get bytes read
        CMP     AL,[BP+3]                 ; ...compare with
requested
        MOV     AL,DS:47h                 ; Read sectors requested
        JZ      FD_013                    ; ...return if all read
        MOV     AL,[BP+7]                 ; Else read sectors
requested
        INC     AL                         ; ...add one for luck

FD_013: SUB     AL,[BP+5]                 ; Subtract stectors read
        RET

FD_64K: MOV     AL,0                     ; Overflowed 64K page
boundary
        RET                               ; ...show no sectors read

NC_BSY: STI                                         ; Wait for operation to
finish
        XOR     CX,CX                       ; ...zero lo order delay
        MOV     AL,2                         ; Load hi order delay

NC_BS1: TEST    Byte ptr DS:3Eh,10000000b        ; Has interrupt set the
flag?
        CLC                                 ; ...hack to slow CPU
        JNZ    NC_BS2                       ; ...yes
        LOOP   NC_BS1                       ; Else back for more
        DEC     AL
        JNZ    NC_BS1

        MOV     Byte ptr DS:41h,80h         ; Time-out, say it
completed
        POP     AX
        MOV     AL,0                         ; ...return time out code
        STC
        RET

NC_BS2: AND     Byte ptr DS:3Eh,01111111b        ; Mask off completion
status
        RET                               ; ...return carry clear

NC_RDY: PUSH    CX                               ; Wait for NEC ready for
comand
        XOR     CX,CX
        MOV     DX,3F4h                       ; ...NEC status port

NC_RD1: IN      AL,DX                           ; Read status of NEC 765
chip
```

BIOS ASM

By Hicks

```

        OR     AL,AL
        JS     NC_RD2                ; ...able to accept
command
        LOOP   NC_RD1
        MOV    Byte ptr DS:41h,80h  ; Else show timeout error
        JMP    short NC_RD3

NC_RD2: TEST    AL,01000000b        ; Test the direction bit
        JNZ   NC_RD4
        MOV    Byte ptr DS:41h,20h  ; ...clear iff controller
err

NC_RD3: POP     CX
        STC
        RET

NC_RD4: INC     DX                ; Load NEC data port
        IN     AL,DX              ; ...read it
        PUSH   AX

        MOV    CX,0Ah              ; Short delay
NC_RD5: LOOP   NC_RD5

        DEC    DX                ; Load NEC status port
        IN     AL,DX              ; ...read status
        TEST   AL,00010000b        ; ...set Z flag if done
        CLC                                ; ...return success
        POP    AX
        POP    CX
        RET

FD_WT1: PUSH   CX                ; Millisecond delay in AH
FD_WT2: XOR    CX,CX
FD_WT3: LOOP   FD_WT3
        DEC    AH
        JNZ   FD_WT2
        POP    CX
        RET

ifdef   SLOW_FLOPPY                ; Run floppy at SLOWEST
speed

FD_SPD: IN     AL,61h              ; Toggle speed on Floppy
Disk
        PUSH   AX                ; ...save old clock rate
        AND    AL,11110011b        ; ...load slowest clock
rate
        OUT    61h,AL              ; ...slow down to 4.77 mHz
        CALL   FD_XQT              ; Execute the i/o request
        POP    AX                ; ...restore old clock
rate
        OUT    61h,AL              ; ...from saved clock byte
```

BIOS ASM

By Hicks

```

                                RET
endif

                                ENTRY    0EF57h                ; Disk interrupt entry

INT_E:  STI                    ; Floppy disk attention
        PUSH    DS
        PUSH    AX
        MOV     AX,40h
        MOV     DS,AX
        OR     Byte ptr DS:3Eh,10000000b    ; Raise "attention" flag
        MOV     AL,20h                    ; Send end_of_interrupt

code
chip    OUT     20h,AL                    ; ...to 8259 interrupt

        POP     AX
        POP     DS
        IRET

NC_STS: MOV     AL,8                    ; Send a "Request status"
        CALL    NEC765                    ; ...to the NEC 765 chip

NC_ST1: PUSH    BX                    ; Alternate entry point
        PUSH    CX
        MOV     CX,7
        XOR    BX,BX

NC_ST2: CALL    NC_RDY                    ; Wait for NEC 765 ready
        JB     NC_ST3                    ; ...NEC 765 error
        MOV     [BX+42h],AL                ; Save status in BIOS block
        JZ     NC_ST4                    ; ...NEC 765 ready
        INC    BX                        ; Count more
        LOOP   NC_ST2
        MOV     Byte ptr DS:41h,20h        ; NEC 765 controller error

NC_ST3: STC                    ; Set error condition
        POP    CX
        POP    BX
        POP    AX
        MOV    AL,0
        RET

NC_ST4: POP     CX                    ; Successful return
        POP    BX
        RET

NEC765: chip PUSH    CX                    ; Send control to NEC 765
        PUSH    DX
        PUSH    AX
        XOR    CX,CX
        MOV    DX,3F4h                    ; Load NEC 765 status port
```

BIOS ASM

By Hicks

```
NEC_01: IN      AL,DX          ; Read NEC 765 status
        OR      AL,AL
        JS      NEC_02        ; ...done
        LOOP   NEC_01
        MOV     Byte ptr DS:41h,80h ; Set time out status
        JMP     short NEC_05

NEC_02: TEST    AL,40h        ; Check data direction
        JZ      NEC_03
        MOV     Byte ptr DS:41h,20h ; ...NEC 765 is gimped
        JMP     short NEC_05

NEC_03: INC     DX            ; Load NEC 765 data port
        POP     AX
        OUT     DX,AL        ; ...write user's
parameter
        CLC
        POP     DX
        POP     CX
NEC_04: RET

NEC_05: POP     AX            ; Common error return
        POP     DX
        POP     CX
        POP     AX
        MOV     AL,0
        STC
        RET

        ENTRY   0EFC7h      ; IBM entry for disk param

INT_1E: db      11001111b    ; Disk parameter table
        db      2
        db      25h
        db      2
        db      8
        db      2Ah
        db      0FFh
        db      50h
        db      0F6h
        db      19h
        db      4

        ENTRY   0EFD2h      ; IBM entry for parallel
LPT

INT_17: STI            ; Parallel printer services
        PUSH    DS
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

BIOS ASM

By Hicks

```

MOV     BX,40h
MOV     DS,BX
MOV     BX,DX                                ; DX is printer index (0 -
3)
SHL     BX,1                                ; ...word index
MOV     DX,[BX+8]                            ; Load printer port
OR      DX,DX
JZ      LP_01                                ; Goes to black hole
OR      AH,AH
JZ      LP_02                                ; Function is print, AH=0
DEC     AH
JZ      LP_INIT                              ; Function is init , AH=1
DEC     AH
JZ      LP_STS                                ; Get the status , AH=2

LP_01:  POP     DX
        POP     CX
        POP     BX
        POP     DS
        IRET

LP_02:  OUT     DX,AL                          ; Char --> data lines 0-7
        INC     DX                            ; Printer status port
        MOV     BH,[BX+78h]                   ; Load time out parameter
        MOV     AH,AL

LP_05:  XOR     CX,CX                          ; Clear lo order time out

LP_POL: IN     AL,DX                          ; Get line printer status
        OR      AL,AL                          ; ...ready?
        JS      LP_DON                        ; ...done if so
        LOOP   LP_POL                         ; Decrement hi order time
out
        JNZ    LP_05

        OR     AL,00000001b                   ; Set timeout in Status
Byte
        AND    AL,11111001b                   ; ...bits returned to
caller
        JMP    short LP_TOG

LP_DON: INC     DX                            ; Printer control port
        MOV     AL,00001101b                   ; Set output strobe hi
        OUT     DX,AL                          ; ...data lines 0-7 valid

LP_STR: MOV     AL,00001100b                   ; Set output strobe lo
        OUT     DX,AL                          ; ...data lines 0-7 ?????
        DEC     DX                            ; Printer status port
        JMP     short LP_ST1                   ; ...get line printer
status
```

BIOS ASM

By Hicks

```
LP_STS: MOV     AH,AL           ; Save copy of character
        INC     DX             ; Printer status port

LP_ST1: IN      AL,DX          ; Read printer status
        AND     AL,11111000b   ; ...bits returned to
caller

LP_TOG: XOR     AL,01001000b   ; ...toggle
ERROR,ACKNOWLEDGE
        XCHG    AL,AH
        JMP     LP_01          ; Exit,
AH=Status,AL=character

LP_INI: MOV     AH,AL           ; Initialize the line
printer
        INC     DX
        INC     DX
        MOV     AL,00001000b
        OUT    DX,AL           ; Request initialize
        MOV     CX,5DCh        ; ...delay
LP_DLY: LOOP    LP_DLY
        JMP     LP_STR         ; Strobe the line printer

        ENTRY   0F045h        ; IBM entry point for table

V_TABLE dw     CRT_0           ; Set mode
        dw     CRT_1           ; Set cursor type
        dw     CRT_2           ; Set cursor position
        dw     CRT_3           ; Get cursor position
        dw     CRT_4           ; Read light pen position
        dw     CRT_5           ; Set active display page
        dw     CRT_6           ; Scroll active page up
        dw     CRT_7           ; Scroll active page down
        dw     CRT_8           ; Read attribute/character
        dw     CRT_9           ; Write attribute/character
        dw     CRT_10          ; Read character only
        dw     CRT_11          ; Set color
        dw     CRT_12          ; Write pixel
        dw     CRT_13          ; Read pixel
        dw     CRT_14          ; Write teletype
        dw     CRT_15          ; Return current video
state

        ENTRY   0F065h        ; IBM entry, video bios
service

INT_10: STI           ; Video bios service AH=(0-
15.)
        CLD           ; ...strings auto-
increment
        PUSH    BP
        PUSH    ES
```

BIOS ASM

By Hicks

```
PUSH    DS
PUSH    SI
PUSH    DI
PUSH    DX
PUSH    CX
PUSH    BX
PUSH    AX
MOV     BX,40h
MOV     DS,BX
MOV     BL,DS:10h           ; Get equipment byte
AND     BL,00110000b       ; ...isolate video mode
CMP     BL,00110000b       ; Check for monochrome card
MOV     BX,0B800h
JNZ     C_01               ; ...not there, BX --> CGA
MOV     BX,0B000h         ; Else           BX --> MONO

C_01:   PUSH    BX           ; Save video buffer address
        MOV     BP,SP       ; ...start of stack frame
        CALL   C_02         ; ...then do the function
        POP     SI
        POP     AX
        POP     BX
        POP     CX
        POP     DX
        POP     DI
        POP     SI
        POP     DS
        POP     ES
        POP     BP
        IRET

MAPBYT: PUSH    DX           ; Mul AL by BX, CX --> buf
        MOV     AH,0
        MUL    BX           ; Position in AX
        POP     DX
        MOV     CX,[BP+0]   ; CX --> video buffer
        RET

tables  ENTRY    0F0A4h     ; IBM entry, SET_MODE

INT_1D: db     38h,28h,2Dh,0Ah,1Fh,6,19h ; Init string for 40 x 25
        db     1Ch,2,7,6,7
        db     0,0,0,0

col     db     71h,50h,5Ah,0Ah,1Fh,6,19h ; Init string for 80 x 25
        db     1Ch,2,7,6,7
        db     0,0,0,0

        db     38h,28h,2Dh,0Ah,7Fh,6,64h ; Init string for GRAPHIX
        db     70h,2,1,6,7
```

BIOS ASM

By Hicks

```

        db      0,0,0,0

b/w    db      61h,50h,52h,0Fh,19h,6,19h      ; Init string for 80 x 25
        db      19h,2,0Dh,0Bh,0Ch
        db      0,0,0,0

REGENL dw      0800h      ; Regen len, 40 x 25
        dw      1000h      ;           80 x 25
        dw      4000h      ;           GRAPHIX
        dw      4000h

MAXCOL db      28h,28h,50h,50h,28h,28h,50h,50h ; Maximum columns

MODES  db      2Ch,28h,2Dh,29h,2Ah,2Eh,1Eh,29h ; Table of mode sets

TABMUL db      00h,00h,10h,10h,20h,20h,20h,30h ; Table lookup for multiply

C_02:  CMP      AH,0Fh      ; Is AH a legal video
command?
        JBE     C_03
        RET     ; ...error return if not

C_03:  SHL     AH,1      ; Make word value
        MOV     BL,AH      ; ...then set up BX
        MOV     BH,0
        JMP     Word ptr CS:[BX+V_TABLE] ; ...vector to routines

CRT_0: MOV     AL,DS:10h    ; Set mode of CRT
        MOV     DX,3B4h    ; ...mono port
        AND     AL,00110000b ; ...get display type
        CMP     AL,00110000b ; ...equal if mono
        MOV     AL,1      ; Assume mono display
        MOV     BL,7      ; ...mode is 7
        JZ      C0_01     ; ...Skip if mono, else

CGA
AL)    MOV     BL,[BP+2]    ; BL = mode number (user
        MOV     DL,0D4h    ; 3D4 is CGA port
        DEC     AL

C0_01: MOV     DS:63h,DX    ; Save cur. CRT display
port
        ADD     DL,4
        OUT     DX,AL      ; Reset the video
        MOV     DS:49h,BL  ; ...save cur. CRT mode
        PUSH    DS
        XOR     AX,AX
        MOV     DS,AX
        LES     SI,Dword ptr DS:74h ; SI --> INT_1D video param
        POP     DS
        MOV     BH,0
```

BIOS ASM

By Hicks

```

        PUSH    BX
        MOV     BL,CS:[BX+TABMUL]           ; Get BL for index into
INT_1D
        ADD     SI,BX
        MOV     CX,10h                     ; Sixteen values to send

C0_02:  MOV     AL,ES:[SI]                 ; Value to send in SI
        CALL   SENDAX                     ; ...send it
        INC    AH                          ; ...bump count
        INC    SI                          ; ...point to next
        LOOP   C0_02                      ; ...loop until done

        MOV     BX,[BP+0]                  ; BX --> regen buffer
        MOV     ES,BX                      ; ...into ES segment
        XOR     DI,DI
        CALL   MODCHK                     ; Set flags acc. to mode
        MOV     CX,2000h                   ; ...assume CGA
        MOV     AX,0                       ; ...and graphics
        JB     C0_04                       ; ...do graphics fill
        JNZ    C0_03                       ; ...Alphanumeric fill
        MOV     CX,800h                    ; ...mono card
C0_03:  MOV     AX,7*100h+' '              ; Word for text fill
C0_04:  REPZ   STOSW                       ; ...fill regen buffer

        MOV     DX,DS:63h                  ; Get the port
        ADD     DL,4
        POP     BX
        MOV     AL,CS:[BX+MODES]           ; Load data to set for mode
        OUT    DX,AL                       ; ...and send it
        MOV     DS:65h,AL                  ; ...then save active data
        INC    DX
        MOV     AL,30h                     ; Assume not 640 x 200 b/w
        CMP    BL,6                        ; ...correct?
        JNZ    C0_05                       ;
        MOV     AL,3Fh                      ; Palette for 640 x 200 b/w

C0_05:  MOV     DS:66h,AL                  ; ...save palette
        OUT    DX,AL                       ; ...send palette
        XOR    AX,AX
        MOV     DS:4Eh,AX                   ; Start at beg. of 1st page
        MOV     DS:62h,AL                  ; ...active page=page 0
        MOV     CX,8                       ; Do 8 pages of cursor data
        MOV     DI,50h                     ; Page cursor data at 40:50

C0_06:  MOV     [DI],AX                    ; Cursor at upper left of
page
        INC    DI                          ; ...next page
        LOOP   C0_06
        MOV     Word ptr DS:60h,0607h      ; Cursor: Line 6 thru Line
7
        MOV     AL,CS:[BX+MAXCOL]          ; Get display width
        MOV     DS:4Ah,AX                  ; ...save it

```

BIOS ASM

By Hicks

```

        AND     BL,11111110b
        MOV     AX,Word ptr CS:[BX+REGENL]    ; Get video regen length
        MOV     DS:4Ch,AX                    ; ...save it
        RET

CRT_1:  MOV     CX,[BP+6]                    ; Set cursor type, from CX
        MOV     DS:60h,CX                    ; ...save it
        MOV     AH,0Ah                       ; CRT index register 0Ah
        CALL    OT6845                        ; ...send CH,CL to CRT reg
        RET

CRT_2:  MOV     BL,[BP+5]                    ; Set cursor position, page
        BH
        SHL     BL,1                          ; ...(our BL)
        MOV     BH,0
        MOV     AX,[BP+8]                    ; Position in user DX (our
AX)
        MOV     [BX+50h],AX                  ; ...remember cursor
position
        JMP     SETCUR                        ; ...set 6845 cursor
hardware

CRT_3:  MOV     BL,[BP+5]                    ; Get cursor position, page
        BH
        SHL     BL,1
        MOV     BH,0
        MOV     AX,[BX+50h]
        MOV     [BP+8],AX                    ; ...return position in
user DX
        MOV     AX,DS:60h                    ; Get cursor mode
        MOV     [BP+6],AX                    ; ...return in user CX
        RET

PENOFF: db     3,3,5,5,3,3,3,4              ; Light pen offset table

CRT_4:  MOV     DX,DS:63h                    ; Read light pen position
        ADD     DL,6
        MOV     Byte ptr [BP+3],0           ; AH=0, assume not
triggered
        IN      AL,DX
        TEST    AL,00000100b
        JZ     C4_05                          ; Skip, reset if pen not
set
        TEST    AL,00000010b
        JNZ    C4_01                          ; Skip if pen triggered
        RET                                     ; ...return, do not reset

C4_01:  MOV     AH,10h                       ; Offset to pen port is 10h
        CALL    PENXY                          ; ...read into CH,CL
        MOV     BL,DS:49h                     ; Get CRT mode data word
        MOV     CL,BL
        MOV     BH,0
```

BIOS ASM

By Hicks

```

        MOV     BL,Byte ptr CS:[BX+PENOFF]      ; Load offset for
subtraction
        SUB     CX,BX
        JNS     C4_02                          ; ...did not overflow
        XOR     AX,AX                          ; Else fudge a zero

C4_02:  CALL    MODCHK                          ; Set flags on display type
        JNB     C4_03                          ; ...text mode, skip
        MOV     CH,28h
        DIV     DL
        MOV     BL,AH
        MOV     BH,0
        MOV     CL,3
        SHL     BX,CL
        MOV     CH,AL
        SHL     CH,1
        MOV     DL,AH
        MOV     DH,AL
        SHR     DH,1
        SHR     DH,1
        CMP     Byte ptr DS:49h,6              ; Mode 640 x 200 b/w?
        JNZ     C4_04                          ; ...no, skip
        SHL     DL,1
        SHL     BX,1
        JMP     short C4_04

C4_03:  DIV     Byte ptr DS:4Ah                ; Divide by columns in
screen
        XCHG    AL,AH                          ; ...as this is text mode
        MOV     DX,AX
        MOV     CL,3
        SHL     AH,CL
        MOV     CH,AH
        MOV     BL,AL
        MOV     BH,0
        SHL     BX,CL

C4_04:  MOV     Byte ptr [BP+3],1              ; Return AH=1, light pen
read
        MOV     [BP+8],DX                      ; ...row, column in user
DX
        MOV     [BP+4],BX                      ; ...pixel column in user
BX
        MOV     [BP+7],CH                     ; ...raster line in user
CH

C4_05:  MOV     DX,DS:63h                      ; Get port of active CRT
card
        ADD     DX,7
        OUT     DX,AL                          ; ...reset the light pen
        RET
```

BIOS ASM

By Hicks

```
CRT_5:  MOV     AL,[BP+2]           ; Set active display page
to AL
        MOV     DS:62h,AL       ; ...save new active page
        MOV     AH,0           ; ...clear hi order
        PUSH    AX
        MOV     BX,DS:4Ch      ; Get size of regen. buffer
        MUL     BX             ; ...times number of pages
        MOV     DS:4Eh,AX      ; Now AX = CRT offset, save
        SHR     AX,1           ; ...now word offset
        MOV     CX,AX          ; ...save a copy
        MOV     AH,0Ch         ; CRT index register 0Ch
        CALL    OT6845         ; ...send CH,CL thru CRT
reg
        POP     BX
        CALL    MOVCUR         ; Save new parameters
        RET

CRT_6:
CRT_7:  CALL    MODCHK         ; Scroll active page up
        JNB    SCR_01         ; Scroll active page down
        JMP    SCG_01         ; Graphics scroll

SCR_01: CLD                   ; Strings go upward
        CMP    Byte ptr DS:49h,2
        JB    SCR_03         ; ...no retrace wait
needed
        CMP    Byte ptr DS:49h,3
        JA    SCR_03         ; ...no retrace wait
needed
        MOV    DX,3DAh        ; Else 80 x 25, do the
kludge

SCR_02: IN     AL,DX           ; Read CGA status register
        TEST   AL,00001000b   ; ...vertical retrace?
        JZ    SCR_02         ; ...wait until it is
        MOV    DX,3D8h        ; Then go and
        MOV    AL,25h         ; ...turn the display
        OUT   DX,AL          ; ...off to avoid snow

SCR_03: MOV    AX,[BP+8]      ; Get row,column of upper
left
        PUSH   AX
        CMP    Byte ptr [BP+3],7
        JZ    SCR_04         ; Check for scroll down
        ; ...yes, skip if so
        MOV    AX,[BP+6]     ; Get row,column of lowr
right

SCR_04: CALL   RC2COL        ; Get byte offset in CRT
buf
        ADD    AX,DS:4Eh     ; ...add base for CRT
buf
        MOV    SI,AX
```

BIOS ASM

By Hicks

```

        MOV     DI,AX
        POP     DX
        SUB     DX,[BP+6]           ; Subtract (row,col) lwr
rhgt
        ADD     DX,101h           ; ...width of one char
        MOV     BX,DS:4Ah         ; Get columns in display
        SHL     BX,1             ; ...bytes in row of
display
        PUSH    DS
        MOV     AL,[BP+2]         ; Get scroll fill character
        CALL   MAPBYT           ; ...calculate offset
        MOV     ES,CX            ; CX --> byte in buffer
        MOV     DS,CX
        CMP     Byte ptr [BP+3],6 ; Scroll up?
        JZ     SCR_05           ; ...skip if so
        NEG     AX
        NEG     BX
        STD
        ; Else start at top of page
SCR_05: MOV     CL,[BP+2]         ; Get count of lines to
scroll
        OR     CL,CL
        JZ     SCR_07           ; ...nothing to do
        ADD     SI,AX
        SUB     DH,[BP+2]
SCR_06: MOV     CH,0             ; Clear hi order word count
        MOV     CL,DL           ; ...load lo order word
count
        PUSH    DI
        PUSH    SI
        REPZ   MOVSW           ; Do the scroll
        POP     SI
        POP     DI
        ADD     SI,BX
        ; Move one line in
direction
        ADD     DI,BX
        ; " "
        DEC     DH             ; One less line to scroll
        JNZ    SCR_06
        MOV     DH,[BP+2]       ; Now get number of rows
SCR_07: MOV     CH,0             ; Clear hi order word count
        MOV     AH,[BP+5]       ; ...get fill attribute
        MOV     AL,' '          ; ...fill character
SCR_08: MOV     CL,DL           ; Get characters to scroll
        PUSH    DI
        REPZ   STOSW           ; ...store fill attr/char
        POP     DI
        ADD     DI,BX
        ; Show row was filled
        DEC     DH
        JNZ    SCR_08
        ; ...more rows are left
```

BIOS ASM

By Hicks

```

        POP     DS
        CALL    MODCHK           ; Check for monochrome card
        JZ      SCR_09          ; ...skip if so
        MOV     AL,DS:65h       ; Get the mode data byte
        MOV     DX,3D8h        ; ...load active CRT card
port
        OUT     DX,AL           ; ...and unblank the
screen
SCR_09: RET

SCG_01: CLD                     ; Assume GRAFIX scroll up
        MOV     AX,[BP+8]       ; (Row,Col) of lower right
        PUSH    AX
        CMP     Byte ptr [BP+3],7 ; Scroll down?
        JZ      SCG_02         ; ...skip if so
        MOV     AX,[BP+6]       ; (Row,Col) of upper left

SCG_02: CALL    GRAMAP          ; Convert (Row,Col) ->
Chars
        MOV     DI,AX
        POP     DX
        SUB     DX,[BP+6]       ; Chars to copy over
        ADD     DX,101h        ; ...width of one char
        SHL     DH,1
        SHL     DH,1
        MOV     AL,[BP+3]       ; Get command type
        CMP     Byte ptr DS:49h,6 ; ...is this 640 x 200?
        JZ      SCG_03         ; ...skip if so
        SHL     DL,1           ; Else bigger characters
        SHL     DI,1
        CMP     AL,7           ; Is this scroll down?
        JNZ     SCG_03         ; ...skip if not so
        INC     DI

SCG_03: CMP     AL,7           ; Is this scroll down?
        JNZ     SCG_04         ; ...skip if not so
        ADD     DI,0F0h

SCG_04: MOV     BL,[BP+2]       ; Number of rows to blank
        SHL     BL,1
        SHL     BL,1
        PUSH    BX
        SUB     DH,BL           ; Subtract from row count
        MOV     AL,50h
        MUL     BL
        MOV     BX,1FB0h
        CMP     Byte ptr [BP+3],6 ; Is this scroll up?
        JZ      SCG_05         ; ...skip if so
        NEG     AX             ; Else do it
        MOV     BX,2050h
        STD     BX             ; ...in reverse
```

BIOS ASM

By Hicks

```
SCG_05: MOV     SI,DI           ; End of area
        ADD     SI,AX           ; ...start
        POP     AX
        OR      AL,AL
        MOV     CX,[BP+0]
        MOV     DS,CX
        MOV     ES,CX
        JZ      SCG_07         ; No rows to scroll
        PUSH    AX

SCG_06: MOV     CH,0           ; Zero hi order byte count
        MOV     CL,DL           ; ...bytes in row
        PUSH    SI
        PUSH    DI
        REPZ    MOVSB          ; Copy one plane
        POP     DI
        POP     SI
        ADD     SI,2000h        ; Load other grafix
        ADD     DI,2000h        ; ...video plane
        MOV     CL,DL
        PUSH    SI
        PUSH    DI
        REPZ    MOVSB          ; Copy other plane
        POP     DI
        POP     SI
        SUB     SI,BX
        SUB     DI,BX
        DEC     DH              ; One less row to scroll
        JNZ     SCG_06         ; ...loop if more to do
        POP     AX
        MOV     DH,AL          ; Load rows to blank

SCG_07: MOV     AL,[BP+5]      ; Get fill attribute
        MOV     CH,0

SCG_08: MOV     CL,DL          ; Get bytes per row
        PUSH    DI
        REPZ    STOSB          ; Load row with fill attr.
        POP     DI
        ADD     DI,2000h        ; Do other grafix video
plane
        MOV     CL,DL
        PUSH    DI
        REPZ    STOSB          ; Load row with fill attr.
        POP     DI
        SUB     DI,BX
        DEC     DH              ; Show one less row to
blank
        JNZ     SCG_08         ; ...loop if more to do
        RET
```

BIOS ASM

By Hicks

```
CRT_8: ; Read attribute/character
CRT_9: ; Write attribute/character
CRT_10: CALL MODCHK ; Write character only
        JB CG8_01 ; ... graphics operation
        MOV BL,[BP+5] ; Get the display page
        MOV BH,0
        PUSH BX
        CALL MPRC2C ; Convert Row,Col,Page ->
Col
        MOV DI,AX ; ...offset in DI
        POP AX
        MUL Word ptr DS:4Ch ; Page length X page number
        ADD DI,AX ; ...current char.
position
        MOV SI,DI ; ...move into si
        MOV DX,DS:63h ; Display port into DX
        ADD DX,6 ; ...get status port
        PUSH DS
        MOV BX,[BP+0] ; BX --> regen. buffer
        MOV DS,BX
        MOV ES,BX
        MOV AL,[BP+3] ; Get user (AH) func
request
        CMP AL,8
        JNZ C9_01 ; ...skip if not read attr

C8_01: IN AL,DX ; Read CRT display status
        TEST AL,00000001b ; ...test for hor. retrace
        JNZ C8_01 ; Yes, wait for display on
        CLI ; ...no interrupts now

C8_02: IN AL,DX ; Read CRT display status
        TEST AL,00000001b ; ...test for hor. retrace
        JZ C8_02 ; ...not yet, wait for it

        LODSW ; Read character/attribute
        POP DS
        MOV [BP+2],AL ; Return character
        MOV [BP+3],AH ; ..and attribute
        RET

C9_01: MOV BL,[BP+2] ; Get char. to write
        MOV BH,[BP+4] ; ...attribute
        MOV CX,[BP+6] ; ...character count
        CMP AL,0Ah ; Write char. only?
        JZ CA_01 ; ...skip if so

C9_02: IN AL,DX ; Read CRT display status
        TEST AL,00000001b ; ...test for hor. retrace
        JNZ C9_02 ; Yes, wait for display on
        CLI ; ...no interrupts now
```

BIOS ASM

By Hicks

```
C9_03:  IN      AL,DX          ; Read CRT display status
        TEST   AL,00000001b ; ...test for hor. retrace
        JZ     C9_03        ; ...not yet, wait for it

        MOV    AX,BX        ; Get char/attribute
        STOSW                ; ...write it
        LOOP   C9_02        ; ...loop for char. count
        POP    DS
        RET

CA_01:  IN      AL,DX          ; Read CRT display status
        TEST   AL,00000001b ; ...test for hor. retrace
        JNZ   CA_01        ; ...not yet, wait for it
        CLI                                ; ...no interrupts now

CA_02:  IN      AL,DX          ; Read CRT display status
        TEST   AL,00000001b ; ...test for hor. retrace
        JZ     CA_02        ; ...not yet, wait for it

        MOV    AL,BL        ; Get character
        STOSB                ; ...write it
        INC    DI            ; ...skip attribute
        LOOP   CA_01        ; ...loop for char. count
        POP    DS
        RET

CG8_01: CMP     Byte ptr [BP+3],8 ; Read graphics
char/attr. ?
        JNZ   CG9_01        ; ...no, must be write
        JMP   CGR_01        ; Else read char/attr.

CG9_01: MOV     AX,DS:50h      ; Get cursor position
        CALL  GRAMAP         ; ...convert (row,col) ->
col
        MOV    DI,AX         ; Save in displacement
register
        PUSH  DS
        MOV   AL,[BP+2]      ; Get character to write
        MOV   AH,0
        OR    AL,AL         ; Is it user character set?
        JS    CG9_02        ; ...skip if so
        MOV   DX,CS         ; Else use ROM character
set
        MOV   SI,offset GRAFIX ; ...offset GRAFIX into SI
        JMP  short CG9_03

CG9_02: AND     AL,7Fh        ; Origin to zero
        XOR    BX,BX        ; ...then go load
        MOV    DS,BX        ; ...user grafix
        LDS   SI,Dword ptr DS:7Ch ; ...vector, offset in SI
        MOV   DX,DS        ; ...segment into DX
```

BIOS ASM

By Hicks

```
CG9_03: POP      DS          ; Restore data segment
        MOV      CL,3        ; ...char 8 pixels wide
        SHL      AX,CL
        ADD      SI,AX       ; Add regen. buffer base
addr.
        MOV      AX,[BP+0]   ; ...get regen buffer addr.
        MOV      ES,AX       ; ...into ES
        MOV      CX,[BP+6]   ; ...load char. count
        CMP      Byte ptr DS:49h,6 ; Is the mode 640 x 200
b/w?
        PUSH     DS
        MOV      DS,DX
        JZ       CG8_02      ; ...skip if so
        SHL      DI,1
        MOV      AL,[BP+4]   ; Get char. attribute
        AND      AX,3
        MOV      BX,5555h
        MUL     BX
        MOV      DX,AX
        MOV      BL,[BP+4]

CG9_04: MOV      BH,8        ; Char 8 pixels wide
        PUSH     DI
        PUSH     SI

CG9_05: LODSB          ; Read the screen
        PUSH     CX
        PUSH     BX
        XOR      BX,BX
        MOV      CX,8

CG9_06: SHR      AL,1        ; Shift bits thru byte
        RCR      BX,1
        SAR      BX,1
        LOOP    CG9_06

        MOV      AX,BX      ; Result into AX
        POP      BX
        POP      CX
        AND      AX,DX
        XCHG    AH,AL
        OR       BL,BL
        JNS     CG9_07
        XOR     AX,ES:[DI]

CG9_07: MOV      ES:[DI],AX  ; Write new word
        XOR     DI,2000h
        TEST    DI,2000h    ; Is this other plane?
        JNZ     CG9_08     ; ...nope
        ADD     DI,50h      ; Else advance character

CG9_08: DEC      BH        ; Show another char written
```

BIOS ASM

By Hicks

```
JNZ      CG9_05                ; ...more to go
POP      SI
POP      DI
INC      DI
INC      DI
LOOP     CG9_04
POP      DS
RET

CG8_02:  MOV     BL,[BP+4]      ; Get display page
        MOV     DX,2000h      ; ...size of grafix plane

CG8_03:  MOV     BH,8          ; Pixel count to write
        PUSH   DI
        PUSH   SI

CG8_04:  LODSB                ; Read from one plane
        OR     BL,BL          ; ...done both planes?
        JNS    CG8_05        ; ...skip if not
        XOR    AL,ES:[DI]    ; Else load attribute

CG8_05:  MOV     ES:[DI],AL    ; Write out attribute
        XOR    DI,DX          ; ...get other plane
        TEST   DI,DX          ; Done both planes?
        JNZ    CG8_06        ; ...skip if not
        ADD    DI,50h        ; Else position for now
char

CG8_06:  DEC     BH            ; Show row of pixels read
        JNZ    CG8_04        ; ...not done all of them
        POP    SI
        POP    DI
        INC    DI
        LOOP   CG8_03
        POP    DS
        RET

CGR_01:  CLD                ; Increment upwards
        MOV    AX,DS:50h     ; ...get cursor position
        CALL   GRAMAP       ; Convert (row,col) ->
columns
        MOV    SI,AX         ; ...save in SI
        SUB    SP,8          ; Grab 8 bytes temp storage
        MOV    DI,SP         ; ...save base in DI
        CMP    Byte ptr DS:49h,6 ; Mode 640 x 200 b/w?
        MOV    AX,[BP+0]    ; ...AX --> CRT regen
buffer
        PUSH   DS
        PUSH   DI
        MOV    DS,AX
        JZ     CGR_06        ; Mode is 640 x 200 b/w -
skip
```

BIOS ASM

By Hicks

```

        MOV     DH,8                ; Eight pixels high/char
        SHL     SI,1
        MOV     BX,2000h            ; Bytes per video plane

CGR_02: MOV     AX,[SI]              ; Read existing word
        XCHG   AH,AL
        MOV     CX,0C000h          ; Attributes to scan for
        MOV     DL,0

CGR_03: TEST    AX,CX                ; Look for attributes
        CLC
        JZ     CGR_04              ; ...set, skip
        STC                          ; Else show not set

CGR_04: RCL     DL,1
        SHR     CX,1
        SHR     CX,1
        JNB    CGR_03              ; ...more shifts to go
        MOV     SS:[DI],DL
        INC     DI
        XOR     SI,BX              ; Do other video plane
        TEST    SI,BX              ; ...done both planes?
        JNZ    CGR_05              ; ...no, skip
        ADD     SI,50h             ; Else advance pointer

CGR_05: DEC     DH                  ; Show another pixel row
done
        JNZ    CGR_02              ; ...more rows to do
        JMP    short CGR_08

CGR_06: MOV     DH,4                ; Mode 640 x 200 b/w -
special

CGR_07: MOV     AH,[SI]              ; Read pixels from one
plane
        MOV     SS:[DI],AH          ; ...save on stack
        INC     DI                  ; ...advance
        MOV     AH,[SI+2000h]       ; Read pixels from other
plane
        MOV     SS:[DI],AH          ; Save pixels on stack
        INC     DI                  ; ...advance
        ADD     SI,50h              ; Total pixels in char
        DEC     DH                  ; ...another row processed
        JNZ    CGR_07              ; ...more to do

CGR_08: MOV     DX,CS               ; Load segment of grafix
char
        MOV     DI,offset GRAFIX    ; ...and offset
        MOV     ES,DX               ; ...save offset in ES
        MOV     DX,SS
        MOV     DS,DX
        POP     SI
```

BIOS ASM

By Hicks

```

        MOV     AL,0
CGR_09: MOV     DX,80h           ; Number of char. in grafix
set
CGR_10: PUSH   SI
        PUSH   DI
        MOV    CX,8             ; Bytes to compare for char
        REPZ  CMPSB            ; ...do compare
        POP    DI
        POP    SI
        JZ    CGR_11           ; Found grafix character
        INC   AL               ; ...else show another
char
        ADD   DI,8             ; ...advance one row
        DEC   DX               ; ...one less char to
scan
        JNZ   CGR_10           ; Loop if more char left
        OR    AL,AL            ; User grafix character
set?
        JZ    CGR_11           ; ...no, not found
        XOR   BX,BX
        MOV   DS,BX
char    LES   DI,Dword ptr DS:7Ch ; Else load user grafix
        MOV   BX,ES
        OR    BX,DI
        JZ    CGR_11           ; ...not found
char    JMP   short CGR_09      ; Try using user grafix
CGR_11: MOV    [BP+2],AL       ; Return char in user AL
        POP   DS
        ADD   SP,8             ; ...return temp storage
        RET
CRT_11: MOV    DX,DS:63h      ; Set color, get CGA card
port
        ADD   DX,5             ; ...color select register
        MOV   AL,DS:66h       ; Get CRT palette
        MOV   AH,[BP+5]       ; ...new palette ID, user
BH
        OR    AH,AH
        MOV   AH,[BP+4]       ; ...new palette color,
user BL
        JNZ   C_PAL1          ; Palette ID specified,
skip
        AND   AL,0E0h
        AND   AH,1Fh          ; Null ID = ID 01Fh
        OR    AL,AH           ; ...set in color
        JMP   short C_PAL2
```

BIOS ASM

By Hicks

```
C_PAL1: AND     AL,0DFh
        TEST    AH,1
        JZ      C_PAL2
        OR      AL,20h

C_PAL2: MOV     DS:66h,AL           ; Save new palette
        OUT    DX,AL               ; ...tell CGA about it
        RET

CRT_12: MOV     AX,[BP+0]           ; Write pixel
        MOV    ES,AX
        MOV    DX,[BP+8]           ; Load row from user DX
        MOV    CX,[BP+6]           ; ... col from user CX
        CALL   LOCDOT              ; Find dot offset
        JNZ    WD_01               ; ...valid
        MOV    AL,[BP+2]           ; Load user color
        MOV    BL,AL
        AND    AL,1
        ROR    AL,1
        MOV    AH,7Fh
        JMP    short   WD_02

WD_01:  SHL    CL,1
        MOV    AL,[BP+2]
        MOV    BL,AL
        AND    AL,3
        ROR    AL,1
        ROR    AL,1
        MOV    AH,3Fh

WD_02:  ROR    AH,CL
        SHR    AL,CL
        MOV    CL,ES:[SI]         ; Read the char with the
dot
        OR     BL,BL
        JNS    WD_03
        XOR    CL,AL              ; Exclusive or existing
color
        JMP    short   WD_04

WD_03:  AND    CL,AH               ; Set new color for dot
        OR     CL,AL

WD_04:  MOV    ES:[SI],CL         ; Write out char with the
dot
        RET

CRT_13: MOV    AX,[BP+0]           ; AX --> video regen buffer
        MOV    ES,AX              ; ...into ES segment
        MOV    DX,[BP+8]           ; Load row from user DX
        MOV    CX,[BP+6]           ; ... col from user CX
```

BIOS ASM

By Hicks

```
CALL    LOCDOT                ; Calculate dot offset
MOV     AL,ES:[SI]            ; ...read dot
JNZ     RD_01                 ; ...was there
SHL     AL,CL
ROL     AL,1
AND     AL,1
JMP     short    RD_02

RD_01:  SHL     CL,1           ; Calculate offset in char
        SHL     AL,CL
        ROL     AL,1
        ROL     AL,1
        AND     AL,3

RD_02:  MOV     [BP+2],AL      ; Return dot pos in user AL
        RET

CRT_14: MOV     BL,DS:62h     ; Get active video page (0-
7)
        SHL     BL,1         ; ...as word index
        MOV     BH,0         ; ...clear hi order
        MOV     DX,[BX+50h]  ; Index into cursor
position

        MOV     AL,[BP+2]    ; Get char. to write
        CMP     AL,8         ; ...back space?
        JZ      TTY_BS      ; ...skip if so
        CMP     AL,LF       ; Is it a carriage return
        JZ      TTY_LF     ; ...skip if so
        CMP     AL,7        ; Print a bell?
        JZ      BLIP       ; ...do beep
        CMP     AL,CR       ; Is it a line feed?
        JZ      TTY_CR     ; ...skip if so
        MOV     BL,[BP+4]    ; Else write at cur pos
        MOV     AH,0Ah
        MOV     CX,1        ; ...one time
        INT     10h
        INC     DL          ; Advance cursor
        CMP     DL,DS:4Ah   ; ...check for line
overflow
        JNZ     TTYPOS
        MOV     DL,0        ; Overflowed, then fake
        JMP     short    TTY_LF ; ...new line

TTY_BS: CMP     DL,0        ; At start of line?
        JZ      TTYPOS     ; ...skip if so
        DEC     DL         ; Else back up
        JMP     short    TTYPOS ; ...join common code

BLIP:   MOV     BL,2        ; Do a short
        CALL    BEEP       ; ...beep
        RET
```

BIOS ASM

By Hicks

```
TTY_CR: MOV     DL,0           ; Position to start of line
;        JMP     short  TTYPOS

TTYPOS: MOV     BL,DS:62h     ; Get active video page (0-
7)
        SHL     BL,1         ; ...as word index
        MOV     BH,0         ; ...clear hi order
        MOV     [BX+50h],DX   ; Remember the cursor
position
        JMP     SETCUR      ; ...set 6845 cursor
hardware

TTY_LF: CMP     DH,18h       ; Done all 24 lines on
page?
        JZ      TTY_L1      ; ...yes, scroll
        INC     DH          ; Else advance line
        JNZ     TTYPOS

TTY_L1: MOV     AH,2         ; Position cursor at line
start
        INT     10h
        CALL    MODCHK      ; Is this text mode?
        MOV     BH,0
        JB      TTY_L2      ; Skip if text mode
        MOV     AH,8
        INT     10h         ; ...else read attribute
        MOV     BH,AH

TTY_L2: MOV     AH,6         ; Now prepare to
        MOV     AL,1         ; ...scroll
        XOR     CX,CX       ; ...the
        MOV     DH,18h     ; ...page
        MOV     DL,DS:4Ah   ; ...up
        DEC     DL
        INT     10h
        RET

CRT_15: MOV     AL,DS:4Ah    ; Get current video state
        MOV     [BP+3],AL   ; ...columns
        MOV     AL,DS:49h   ;
        MOV     [BP+2],AL   ; ...mode
        MOV     AL,DS:62h   ;
        MOV     [BP+5],AL   ; ...page
        RET

MODCHK: PUSH    AX          ; Set flags acc. to cur.
mode
        MOV     AL,DS:49h   ; ...get mode
        CMP     AL,7        ; ...EQU if mono
        JZ      MODCH1
        CMP     AL,4
```

BIOS ASM

By Hicks

```

        CMC
        JNB     MODCH1      ; ...carry set on graphix
        SBB     AL,AL
        STC

MODCH1: POP     AX
        RET

LOCDOT: MOV     AL,50h      ; Dots in char. position
        XOR     SI,SI
        SHR     DL,1        ; Two bytes/char. position
        JNB     LOCD01     ; ...not overflow
        MOV     SI,2000h    ; Else on other video plane

LOCD01: MUL     DL          ; Multiply position by row
        ADD     SI,AX       ; ...add in column
position
        MOV     DX,CX      ; Copy column position
        MOV     CX,302h    ; ...regular char size
        CMP     Byte ptr DS:49h,6 ; Mode 640 x 200, b/w?
        PUSHF
        JNZ     LOCD02     ; ...skip if not
        MOV     CX,703h    ; Else special char. size

LOCD02: AND     CH,DL
        SHR     DX,CL
        ADD     SI,DX
        XCHG   CL,CH
        POPF
        RET

PENXY:  CALL    PENXY1     ; Read light pen position
HI
        MOV     CH,AL      ; ...save in CH
        INC     AH
        CALL    PENXY1     ; Read light pen position
LO
        MOV     CL,AL      ; ...save in CL
        RET

PENXY1: PUSH    DX          ; Read CRT register offset
AL
        MOV     DX,DS:63h  ; ...get active CRT port
        XCHG   AL,AH
        OUT     DX,AL      ; Send initialization byte
        INC     DL          ; ...increment
        IN      AL,DX      ; Read pen position byte
back
        POP     DX
        RET
```

BIOS ASM

By Hicks

```
MPRC2C: MOV      BH,0                ; Convert Row,Col,Page ->
Col                                           ;
        SHL      BX,1                ; ...two bytes/column
        MOV      AX,[BX+50h]         ; Get page number in AX
                                           ; ...join common code
RC2COL: PUSH     BX                  ; Map (AH=row,AL=COL) to
COL                                           ;
        MOV      BL,AL
        MOV      AL,AH
        MUL      Byte ptr DS:4Ah     ; Multiply ROW x
(Row/Column)                                ;
        MOV      BH,0
        ADD      AX,BX                ; ...add in existing COL
        SHL      AX,1                ; ...times 2 cause 2
bytes/col
        POP      BX
        RET

GRAMAP: PUSH     BX                  ; Convert (row,col) -> col
        MOV      BL,AL                ; ...save column
        MOV      AL,AH                ; ...get row
        MUL      Byte ptr DS:4Ah     ; Multiply by columns/row
        SHL      AX,1
        SHL      AX,1
        MOV      BH,0
        ADD      AX,BX                ; Add in columns
        POP      BX
        RET

SETCUR: SHR      BL,1                ; Sets 6845 cursor position
        CMP      DS:62h,BL           ; ...is this page visible?
        JNZ      SEND01              ; No, do nothing in
hardware

MOVCUR: CALL     MPRC2C               ; Map row,col,page to col
        ADD      AX,DS:4Eh           ; + byte offset, regen reg.
        SHR      AX,1
        MOV      CX,AX
        MOV      AH,0Eh              ; Tell 6845 video
controller
                                           ; ...to position the
cursor

OT6845: MOV      AL,CH                ; Send CH,CL thru CRT reg
AH                                           ;
        CALL     SENDAX              ; ...send CH
        INC      AH                  ; ...increment
        MOV      AL,CL                ; ...send CL

SENDAX: PUSH     DX
        MOV      DX,DS:63h           ; Load active video port
        XCHG    AL,AH
```

BIOS ASM

By Hicks

```

        OUT     DX,AL           ; Send hi order
        XCHG   AL,AH
        INC    DL
        OUT     DX,AL           ; ... lo order
        POP    DX

SEND01: RET

        ENTRY   0F841h         ; IBM entry for memory size

INT_12: STI                   ; Kbytes of memory present
        PUSH   DS
        MOV    AX,40h
        MOV    DS,AX
        MOV    AX,DS:13h      ; AX = memory size,
kilobytes
        POP    DS
        IRET

        ENTRY   0F84Dh         ; IBM entry for equipment
check

INT_11: STI                   ; Equipment present
        PUSH   DS
        MOV    AX,40h
        MOV    DS,AX
        MOV    AX,DS:10h     ; AX = equipment byte
contents
        POP    DS
        IRET

        ENTRY   0F859h         ; IBM entry for cassette
int.

INT_15: STC                   ; Cassette service (error
ret)
        MOV    AH,86h
        RETF   2

        ENTRY   0F85Fh         ; IBM non-maskable int.
entry

INT_2:  PUSH   AX             ; Non-maskable interrupt
        IN     AL,62h
        TEST  AL,11000000b    ; Get cause of interrupt
        JNZ   PAR_01         ; ...parity error
        JMP   PAR_07         ; ...math coprocessor (?)

PAR_01: PUSH   BX             ; Parity error bomb
        PUSH   CX
        PUSH   DX
        PUSH   SI
```

BIOS ASM

By Hicks

```
PUSH    DI
PUSH    BP
PUSH    DS
PUSH    ES
MOV     AX,40h                ; Load data segment
MOV     DS,AX
CALL    V_INIT                ; ...clear/init screen
PUSH    DS
PUSH    CS                    ; Point DS at ROM
POP     DS
MOV     SI,offset BOMB_1      ; SI --> Parity message
CALL    PRINT                 ; ...print
POP     DS                    ; ...restore DS
MOV     AX,11h                ; Back cursor over ? marks
CALL    LOCATE                ; ...with call
MOV     AL,0
OUT     0A0h,AL              ; ...disable NMI

interrupts
MOV     DX,61h
IN      AL,DX                 ; Get machine flags
OR      AL,00110000b          ; ...disable parity int.
OUT     DX,AL                 ; Put out new flags
AND     AL,11001111b          ; ...enable parity int.
OUT     DX,AL                 ; Put out new flags
MOV     CL,6
MOV     BX,DS:13h            ; Get memory size (K bytes)
SHL     BX,CL
INC     DX                    ; ...now paragraphs
XOR     AX,AX
MOV     DS,AX

PAR_02: MOV     CX,10h        ; Iterations to check
        XOR     SI,SI

PAR_03: MOV     AH,[SI]      ; Read the byte (dummy)
        IN      AL,DX        ; ...and read status
        TEST    AL,11000000b ; ...to see what happened
        JNZ     PAR_04      ; Read caused parity error
        INC     SI          ; ...else advance pointer
        LOOP   PAR_03      ; ...and try next byte

        MOV     AX,DS
        INC     AX          ; ...next paragraph
        MOV     DS,AX
        CMP     AX,BX
        JNZ     PAR_02      ; More paragraphs to check
        JMP     short PAR_05 ; ...else flakey error

PAR_04: MOV     [SI],AH     ; Save offset in paragraph
        MOV     AX,DS
        CALL    BIGNUM      ; Print segment
        MOV     AX,SI
```

BIOS ASM

By Hicks

```
CALL DIGIT ; Print offset

PAR_05: MOV AX,16h ; Where to position cursor
CALL LOCATE ; ...position cursor
PUSH DS
PUSH CS
POP DS
MOV SI,offset BOMB_2 ; Continue ?
CALL PRINT ; ...ask the user
POP DS
IN AL,21h ; Get interrupt masks
PUSH AX ; ...save them
MOV AL,11111100b
OUT 21h,AL ; Disable all but keyboard
STI ; ...enable interrupt

system
CALL GETCH ; Get keyboard character
PUSH AX ; ...save it
CALL OUTCHR ; Print ascii character
POP AX ; ...restore
CMP AL,'Y' ; User wants to continue
JZ PAR_06 ; ...stupid answer
CMP AL,'y' ; Look for little case "y"
JZ PAR_06 ; ...stupid answer
JMP COLD ; Retry on cold reboot

PAR_06: CALL BLANK ; Clear display
POP AX
OUT 21h,AL ; Restore interrupt system

state
MOV DX,61h ; Dismiss the NMI interrupt
IN AL,DX ; ...read in machine

flags
OR AL,00110000b
OUT DX,AL ; Write out, parity

disabled
AND AL,11001111b ; ...clears parity error
OUT DX,AL ; Write out, parity enabled
MOV AL,80h
OUT 0A0h,AL ; Enable NMI interrupts
POP ES
POP DS
POP BP
POP DI
POP SI
POP DX
POP CX
POP BX

PAR_07: POP AX
IRET
```

BIOS ASM

By Hicks

```
BOMB_1 db 'Parity error at: ?????',0
BOMB_2 db ' Cont?',0

NUMBER: PUSH AX ; Save number
        MOV CL,4
        SHR AL,CL
        CALL DIGIT ; Out first digit
        POP AX
        CALL DIGIT ; Out second digit
        RET

BIGNUM: PUSH AX ; Unsigned word
        MOV AL,AH
        CALL NUMBER
        POP AX
        CALL NUMBER
        RET

OUTCHR: PUSH BX
        PUSH AX
        MOV AH,0Eh ; Teletype print service
        MOV BL,7 ; ...normal intensity
        INT 10h
        POP AX
        POP BX
        RET

DIGIT: PUSH AX ; Print hex digit in AL
        AND AL,0Fh
        CMP AL,9
        JBE D_01
        ADD AL,'A'-'9'-1

D_01: ADD AL,'0' ; Make ascii digit
        CALL OUTCHR ; ...print it
        POP AX

        MOV AL,CR ; Print carriage return
        CALL OUTCHR ; ...on screen
        MOV AL,LF ; Print line feed
        CALL OUTCHR ; ...on screen
        RET

GETCH: MOV AH,0 ; Read keyboard key
        INT 16h
        RET

PRINT: LODSB ; Print zero terminated
string
        OR AL,AL
        JNZ PRINT1 ; ...not terminator in AX
```

BIOS ASM

By Hicks

```
RET

PRINT1: CALL    OUTCHR    ; Print character in AX
        JMP     PRINT    ; ...back for more

BEEP:   PUSH    AX
        PUSH    CX
        MOV     AL,10110110b ; Timer ic 8253 square
waves
        OUT     43h,AL    ; ...channel 2, speaker
        MOV     AX,528h   ; Get countdown constant
word
        OUT     42h,AL    ; ...send lo order
        MOV     AL,AH     ; ...load hi order
        OUT     42h,AL    ; ...send hi order
        IN      AL,61h    ; Read ic 8255 machine
status
        PUSH    AX
        OR     AL,00000011b
        OUT     61h,AL    ; Turn speaker on
        XOR    CX,CX

BEEP_1: LOOP    BEEP_1
        DEC    BL
        JNZ    BEEP_1
        POP    AX
        OUT     61h,AL    ; Turn speaker off
        POP    CX
        POP    AX
        RET

V_INIT: MOV     AH,DS:10h ; Get equipment byte
        AND    AH,00110000b ; ...extract CRT
        MOV    AL,0      ; ...null lo
        CMP    AH,00110000b ; Monochrome?
        JZ     LF9D9     ; ...yes
        MOV    AL,1      ; CGA 40 x 25?
        CMP    AH,00010000b ; ...yes
        JZ     LF9D9     ; CGA 80 x 25?
        MOV    AL,3      ; ...yes

LF9D9: MOV     AH,0      ; Setup subfunction
        INT    10h     ; ...to video
        RET

BLANK:  MOV     DX,184Fh ; Lower right corner of
scroll
        XOR    CX,CX    ; Upper left corner of
scroll
        MOV    AX,600h  ; Blank entire window
        MOV    BH,7     ; Set regular cursor
        INT    10h     ; Call video service scroll
```

BIOS ASM

By Hicks

```

        MOV     AH,2                ; Set cursor position
        XOR     DX,DX              ; ...upper left corner
        MOV     BH,0              ; ...page 0
        INT     10h               ; ...call video service
        RET

LOCATE:  PUSH   DX
        PUSH   BX
        MOV    DX,AX              ; Get position for cursor
        MOV    AH,2
        MOV    BH,0              ; ...page 0
        INT    10h
        POP    BX
        POP    DX
        RET

CHKSUM:  MOV    CX,2000h          ; Bytes in 2764 eprom

CHK_01:  MOV    AL,0             ; ...zero checksum

ADDBYT:  ADD    AL,[BX]          ; Add byte to checksum
        INC    BX                ; ...BX --> next byte
        LOOP   ADDBYT           ; ...loop until done
        OR     AL,AL            ; Set condition codes
        RET                     ; ...and return

MEMTST:  MOV    BX,0400h         ; Load bytes to test
        MOV    AL,55h
;
PAT_1:   XOR    DI,DI           ; Pattern #1, 55h bytes
        MOV    CX,BX
        REPZ   STOSB            ; Fill memory, pattern #1
        XOR    DI,DI
        MOV    CX,BX
        REPZ   SCASB           ; Scan memory for NOT
pattern #1
        JCXZ   PAT_2
        STC                     ; ...flunked
        RET

PAT_2:   XOR    DI,DI           ; Pattern #2 - 0AAh bytes
        MOV    CX,BX
        NOT    AL
        REPZ   STOSB            ; Fill memory, pattern #2
        XOR    DI,DI
        MOV    CX,BX
        REPZ   SCASB           ; Scan memory for NOT
pattern #2
        JCXZ   PAT_3
        STC                     ; ...flunked
        RET
```

BIOS ASM

By Hicks

```
PAT_3:  XOR    DI,DI                ; Pattern #3 - 01h bytes
        MOV    CX,BX
        MOV    AL,1
        REPZ  STOSB                ; Fill memory, pattern #3
        XOR    DI,DI
        MOV    CX,BX
        REPZ  SCASB                ; Scan memory for NOT
pattern #3
        JCXZ  PAT_4
        STC
        RET

PAT_4:  XOR    DI,DI                ; Pattern #4 - 0h bytes
        MOV    CX,BX
        DEC    AL
        REPZ  STOSB                ; Fill memory, pattern #4
        XOR    DI,DI
        MOV    CX,BX
        REPZ  SCASB                ; Scan memory for NOT
pattern #4
        JCXZ  LFA59
        STC
        RET

LFA59:  MOV    AX,ES
        ADD    AX,40h                ; Add 40h to segment number
        MOV    ES,AX
        RET                        ; ...passed

        ENTRY 0FA6Eh                ; IBM graphics char set
entry

GRAFIX db    000h,000h,000h,000h    ; Graphics character set
        db    000h,000h,000h,000h
        db    07Eh,081h,0A5h,081h
        db    0BDh,099h,081h,07Eh
        db    07Eh,0FFh,0DBh,0FFh
        db    0C3h,0E7h,0FFh,07Eh
        db    06Ch,0FEh,0FEh,0FEh
        db    07Ch,038h,010h,000h

        db    010h,038h,07Ch,0FEh
        db    07Ch,038h,010h,000h
        db    038h,07Ch,038h,0FEh
        db    0FEh,07Ch,038h,07Ch
        db    010h,010h,038h,07Ch
        db    0FEh,07Ch,038h,07Ch
        db    000h,000h,018h,03Ch
        db    03Ch,018h,000h,000h

        db    0FFh,0FFh,0E7h,0C3h
        db    0C3h,0E7h,0FFh,0FFh
```

BIOS ASM

By Hicks

```
db      000h,03Ch,066h,042h
db      042h,066h,03Ch,000h
db      0FFh,0C3h,099h,0BDh
db      0BDh,099h,0C3h,0FFh
db      00Fh,007h,00Fh,07Dh
db      0CCh,0CCh,0CCh,078h

db      03Ch,066h,066h,066h
db      03Ch,018h,07Eh,018h
db      03Fh,033h,03Fh,030h
db      030h,070h,0F0h,0E0h
db      07Fh,063h,07Fh,063h
db      063h,067h,0E6h,0C0h
db      099h,05Ah,03Ch,0E7h
db      0E7h,03Ch,05Ah,099h

db      080h,0E0h,0F8h,0FEh
db      0F8h,0E0h,080h,000h
db      002h,00Eh,03Eh,0FEh
db      03Eh,00Eh,002h,000h
db      018h,03Ch,07Eh,018h
db      018h,07Eh,03Ch,018h
db      066h,066h,066h,066h
db      066h,000h,066h,000h

db      07Fh,0DBh,0DBh,07Bh
db      01Bh,01Bh,01Bh,000h
db      03Eh,063h,038h,06Ch
db      06Ch,038h,0CCh,078h
db      000h,000h,000h,000h
db      07Eh,07Eh,07Eh,000h
db      018h,03Ch,07Eh,018h
db      07Eh,03Ch,018h,0FFh

db      018h,03Ch,07Eh,018h
db      018h,018h,018h,000h
db      018h,018h,018h,018h
db      07Eh,03Ch,018h,000h
db      000h,018h,00Ch,0FEh
db      00Ch,018h,000h,000h
db      000h,030h,060h,0FEh
db      060h,030h,000h,000h

db      000h,000h,0C0h,0C0h
db      0C0h,0FEh,000h,000h
db      000h,024h,066h,0FFh
db      066h,024h,000h,000h
db      000h,018h,03Ch,07Eh
db      0FFh,0FFh,000h,000h
db      000h,0FFh,0FFh,07Eh
db      03Ch,018h,000h,000h
```

BIOS ASM

By Hicks

```
db      000h,000h,000h,000h
db      000h,000h,000h,000h
db      030h,078h,078h,030h
db      030h,000h,030h,000h
db      06Ch,06Ch,06Ch,000h
db      000h,000h,000h,000h
db      06Ch,06Ch,0FEh,06Ch
db      0FEh,06Ch,06Ch,000h
```

```
db      030h,07Ch,0C0h,078h
db      00Ch,0F8h,030h,000h
db      000h,0C6h,0CCh,018h
db      030h,066h,0C6h,000h
db      038h,06Ch,038h,076h
db      0DCh,0CCh,076h,000h
db      060h,060h,0C0h,000h
db      000h,000h,000h,000h
```

```
db      018h,030h,060h,060h
db      060h,030h,018h,000h
db      060h,030h,018h,018h
db      018h,030h,060h,000h
db      000h,066h,03Ch,0FFh
db      03Ch,066h,000h,000h
db      000h,030h,030h,0FCh
db      030h,030h,000h,000h
```

```
db      000h,000h,000h,000h
db      000h,030h,030h,060h
db      000h,000h,000h,0FCh
db      000h,000h,000h,000h
db      000h,000h,000h,000h
db      000h,030h,030h,000h
db      006h,00Ch,018h,030h
db      060h,0C0h,080h,000h
```

```
db      07Ch,0C6h,0CEh,0DEh
db      0F6h,0E6h,07Ch,000h
db      030h,070h,030h,030h
db      030h,030h,0FCh,000h
db      078h,0CCh,00Ch,038h
db      060h,0CCh,0FCh,000h
db      078h,0CCh,00Ch,038h
db      00Ch,0CCh,078h,000h
```

```
db      01Ch,03Ch,06Ch,0CCh
db      0FEh,00Ch,01Eh,000h
db      0FCh,0C0h,0F8h,00Ch
db      00Ch,0CCh,078h,000h
db      038h,060h,0C0h,0F8h
db      0CCh,0CCh,078h,000h
db      0FCh,0CCh,00Ch,018h
```

BIOS ASM

By Hicks

```
db      030h,030h,030h,000h

db      078h,0CCh,0CCh,078h
db      0CCh,0CCh,078h,000h
db      078h,0CCh,0CCh,07Ch
db      00Ch,018h,070h,000h
db      000h,030h,030h,000h
db      000h,030h,030h,000h
db      000h,030h,030h,000h
db      000h,030h,030h,060h

db      018h,030h,060h,0C0h
db      060h,030h,018h,000h
db      000h,000h,0FCh,000h
db      000h,0FCh,000h,000h
db      060h,030h,018h,00Ch
db      018h,030h,060h,000h
db      078h,0CCh,00Ch,018h
db      030h,000h,030h,000h

db      07Ch,0C6h,0DEh,0DEh
db      0DEh,0C0h,078h,000h
db      030h,078h,0CCh,0CCh
db      0FCh,0CCh,0CCh,000h
db      0FCh,066h,066h,07Ch
db      066h,066h,0FCh,000h
db      03Ch,066h,0C0h,0C0h
db      0C0h,066h,03Ch,000h

db      0F8h,06Ch,066h,066h
db      066h,06Ch,0F8h,000h
db      0FEh,062h,068h,078h
db      068h,062h,0FEh,000h
db      0FEh,062h,068h,078h
db      068h,060h,0F0h,000h
db      03Ch,066h,0C0h,0C0h
db      0CEh,066h,03Eh,000h

db      0CCh,0CCh,0CCh,0FCh
db      0CCh,0CCh,0CCh,000h
db      078h,030h,030h,030h
db      030h,030h,078h,000h
db      01Eh,00Ch,00Ch,00Ch
db      0CCh,0CCh,078h,000h
db      0E6h,066h,06Ch,078h
db      06Ch,066h,0E6h,000h

db      0F0h,060h,060h,060h
db      062h,066h,0FEh,000h
db      0C6h,0EEh,0FEh,0FEh
db      0D6h,0C6h,0C6h,000h
db      0C6h,0E6h,0F6h,0DEh
```

BIOS ASM

By Hicks

```
db      0CEh,0C6h,0C6h,000h
db      038h,06Ch,0C6h,0C6h
db      0C6h,06Ch,038h,000h

db      0FCh,066h,066h,07Ch
db      060h,060h,0F0h,000h
db      078h,0CCh,0CCh,0CCh
db      0DCh,078h,01Ch,000h
db      0FCh,066h,066h,07Ch
db      06Ch,066h,0E6h,000h
db      078h,0CCh,0E0h,070h
db      01Ch,0CCh,078h,000h

db      0FCh,0B4h,030h,030h
db      030h,030h,078h,000h
db      0CCh,0CCh,0CCh,0CCh
db      0CCh,0CCh,0FCh,000h
db      0CCh,0CCh,0CCh,0CCh
db      0CCh,078h,030h,000h
db      0C6h,0C6h,0C6h,0D6h
db      0FEh,0EEh,0C6h,000h

db      0C6h,0C6h,06Ch,038h
db      038h,06Ch,0C6h,000h
db      0CCh,0CCh,0CCh,078h
db      030h,030h,078h,000h
db      0FEh,0C6h,08Ch,018h
db      032h,066h,0FEh,000h
db      078h,060h,060h,060h
db      060h,060h,078h,000h

db      0C0h,060h,030h,018h
db      00Ch,006h,002h,000h
db      078h,018h,018h,018h
db      018h,018h,078h,000h
db      010h,038h,06Ch,0C6h
db      000h,000h,000h,000h
db      000h,000h,000h,000h
db      000h,000h,000h,0FFh

db      030h,030h,018h,000h
db      000h,000h,000h,000h
db      000h,000h,078h,00Ch
db      07Ch,0CCh,076h,000h
db      0E0h,060h,060h,07Ch
db      066h,066h,0DCh,000h
db      000h,000h,078h,0CCh
db      0C0h,0CCh,078h,000h

db      01Ch,00Ch,00Ch,07Ch
db      0CCh,0CCh,076h,000h
db      000h,000h,078h,0CCh
```

BIOS ASM

By Hicks

db 0FCh,0C0h,078h,000h
db 038h,06Ch,060h,0F0h
db 060h,060h,0F0h,000h
db 000h,000h,076h,0CCh
db 0CCh,07Ch,00Ch,0F8h

db 0E0h,060h,06Ch,076h
db 066h,066h,0E6h,000h
db 030h,000h,070h,030h
db 030h,030h,078h,000h
db 00Ch,000h,00Ch,00Ch
db 00Ch,0CCh,0CCh,078h
db 0E0h,060h,066h,06Ch
db 078h,06Ch,0E6h,000h

db 070h,030h,030h,030h
db 030h,030h,078h,000h
db 000h,000h,0CCh,0FEh
db 0FEh,0D6h,0C6h,000h
db 000h,000h,0F8h,0CCh
db 0CCh,0CCh,0CCh,000h
db 000h,000h,078h,0CCh
db 0CCh,0CCh,078h,000h

db 000h,000h,0DCh,066h
db 066h,07Ch,060h,0F0h
db 000h,000h,076h,0CCh
db 0CCh,07Ch,00Ch,01Eh
db 000h,000h,0DCh,076h
db 066h,060h,0F0h,000h
db 000h,000h,07Ch,0C0h
db 078h,00Ch,0F8h,000h

db 010h,030h,07Ch,030h
db 030h,034h,018h,000h
db 000h,000h,0CCh,0CCh
db 0CCh,0CCh,076h,000h
db 000h,000h,0CCh,0CCh
db 0CCh,078h,030h,000h
db 000h,000h,0C6h,0D6h
db 0FEh,0FEh,06Ch,000h

db 000h,000h,0C6h,06Ch
db 038h,06Ch,0C6h,000h
db 000h,000h,0CCh,0CCh
db 0CCh,07Ch,00Ch,0F8h
db 000h,000h,0FCh,098h
db 030h,064h,0FCh,000h
db 01Ch,030h,030h,0E0h
db 030h,030h,01Ch,000h

db 018h,018h,018h,000h

BIOS ASM

By Hicks

```
db      018h,018h,018h,000h
db      0E0h,030h,030h,01Ch
db      030h,030h,0E0h,000h
db      076h,0DCh,000h,000h
db      000h,000h,000h,000h
db      000h,010h,038h,06Ch
db      0C6h,0C6h,0FEh,000h

clock   ENTRY    0FE6Eh                ; IBM entry, time_of_day

INT_1A: STI                        ; User time_of_day bios
service

        PUSH     DS
        PUSH     AX
        MOV      AX,40h
        MOV      DS,AX
        POP      AX                ; Get request type
        CLI                        ; ...freeze clock
        OR       AH,AH
        JZ       TD_01            ; Read time, AH=0
        DEC     AH
        JNZ     TD_02            ; ...invalid request
        MOV     DS:6Ch,DX        ; Set time, AH=1
        MOV     DS:6Eh,CX        ; ...set time hi
        MOV     Byte ptr DS:70h,0 ; ...not a new day
        JMP     short TD_02

TD_01:  MOV     CX,DS:6Eh        ; Read lo order time
        MOV     DX,DS:6Ch        ; ... hi order time
        CALL    TD_03            ; Read resets overflow

TD_02:  STI                        ; Unfreeze clock
        POP     DS
        IRET

TD_03:  MOV     AL,DS:70h        ; Zero the overflow and
return  XOR     DS:70h,AL        ; ...previous status in
flags   RET

        ENTRY    0FEA5h                ; IBM entry, hardware clock

INT_8:  STI                        ; Routine services clock
tick

        PUSH     DS
        PUSH     DX
        PUSH     AX
        MOV     AX,40h
        MOV     DS,AX
        DEC     Byte ptr DS:40h    ; Decrement motor count
```

BIOS ASM

By Hicks

```
JNZ      TI_01                ; ...not time to shut off
AND      Byte ptr DS:3Fh,11110000b ; Else show motor off
MOV      AL,0Ch                ; ...send motor off
MOV      DX,3F2h               ; ...to the floppy
OUT      DX,AL                 ; ...disk controller

TI_01:   INC      Word ptr DS:6Ch   ; Bump lo order time of day
        JNZ      TI_02                ; ...no carry
        INC      Word ptr DS:6Eh   ; Bump hi order time of day

TI_02:   CMP      Word ptr DS:6Eh,18h ; Is it midnight yet?
        JNZ      TI_03                ; ...no
        CMP      Word ptr DS:6Ch,0B0h ; Possibly, check lo order
        JNZ      TI_03                ; ...not midnight
        MOV      Word ptr DS:6Eh,0   ; Midnight, reset hi order
        MOV      Word ptr DS:6Ch,0   ; ...lo order ticks
        MOV      Byte ptr DS:70h,1   ; Show new day since last
read

TI_03:   INT      1Ch                ; Execute user clock
service
        MOV      AL,20h               ; ...send end_of_interrupt
        OUT      20h,AL               ; ...to 8259 interrupt
chip
        POP      AX
        POP      DX
        POP      DS
        IRET

        ENTRY    0FEF3h               ; IBM entry, time_of_day
clock

VECTORS  dw      int_8                ; Timer tick
        dw      int_9                ; Key attention
        dw      IGNORE               ; Reserved
        dw      IGNORE               ; Reserved for COM2 serial
i/o
        dw      IGNORE               ; Reserved for COM1 serial
i/o
        dw      IGNORE               ; Reserved for hard disk
attn.
        dw      int_e                ; Floppy disk attention
        dw      IGNORE               ; Reserved for parallel
printer
        dw      int_10               ; Video bios services
        dw      int_11               ; Equipment present
        dw      int_12               ; Memories present
        dw      int_13               ; Disk bios services
        dw      int_14               ; Serial com. services
        dw      int_15               ; Cassette bios services
        dw      int_16               ; Keyboard bios services
        dw      int_17               ; Parallel printer services
```

BIOS ASM

By Hicks

```

        dw      IGNORE                ; rom Basic (setup later)
        dw      int_19                 ; Bootstrap
        dw      int_1a                 ; Timer bios services
        dw      DUMMY                  ; Keyboard break user
service
        dw      DUMMY                  ; System tick user service
        dw      int_1d                 ; Video parameter table
        dw      int_1e                 ; Disk parameter table
        dw      ?                      ; Graphic charactr table
ptr
        ENTRY   0FF23h                 ; IBM entry, nonsense
interrupt
IGNORE: PUSH    DS                    ; Unexpected interrupts go
here
        PUSH    DX
        PUSH    AX
        MOV     AX,40h
        MOV     DS,AX
        MOV     AL,0Bh                 ; What IRQ caused this?
        OUT    20h,AL
        NOP
        IN     AL,20h                 ; ... (read IRQ level)
        MOV    AH,AL
        OR     AL,AL
        JNZ    DU_1
        MOV    AL,0FFh               ; Not hardware, say 0FFh
IRQ
        JMP     short    DU_2

DU_1:   IN     AL,21h                 ; Clear the IRQ
        OR     AL,AH
        OUT    21h,AL
        MOV    AL,20h                 ; Send end_of_interrupt
code
        OUT    20h,AL                 ; ...to 8259 interrupt
chip
DU_2:   MOV    DS:6Bh,AH              ; Save last nonsense
interrupt
        POP    AX
        POP    DX
        POP    DS
        IRET

        ENTRY   0FF53h                 ; IBM entry, dummy
interrupts

;INT_1B:
service
```

BIOS ASM

By Hicks

```
;INT_1C:                                ; Clock    tick    user
service
DUMMY:  IRET

        ENTRY    0FF54h                ; IBM entry, print screen

INT_5:  STI                                ; Print screen service
        PUSH     DS
        PUSH     AX
        PUSH     BX
        PUSH     CX
        PUSH     DX
        MOV      AX,40h
        MOV      DS,AX
        CMP      Byte ptr DS:100h,1    ; Print screen in progress?
        JZ       PS_5                    ; ...yes, ignore
        MOV      Byte ptr DS:100h,1    ; Flag print screen in
progress
        CALL     P_CRLF                    ; ...begin new line
        MOV      AH,0Fh
        INT      10h                      ; Get current video state
        PUSH     AX                        ; ...save it
        MOV      AH,3
        INT      10h                      ; Read cursor position
        POP      AX                        ; ...retrieve video state
        PUSH     DX                        ; ...save cursor position
        MOV      CH,19h                    ; Do 25 rows
        MOV      CL,AH                    ; ...columns in current
mode
        XOR      DX,DX                    ; Start printing from (0,0)

PS_1:   MOV      AH,2                    ; Set cursor to position
        INT      10h
        MOV      AH,8                    ; ...and read character
        INT      10h
        OR       AL,AL                    ; Nulls are special case
        JNZ     PS_2
        MOV      AL,' '                    ; ...convert to spaces

PS_2:   PUSH     DX
        XOR      DX,DX
        MOV      AH,DL                    ; Function=Print character
        INT      17h
        POP      DX
        TEST     AH,00100101b            ; Successful print
        JZ       PS_3
        MOV      Byte ptr DS:100h,0FFh    ; No, error in Print Screen
        JMP      short    PS_4

PS_3:   INC      DL                    ; Increment column count
        CMP      CL,DL
        JNZ     PS_1                    ; ...in range, continue
```

BIOS ASM

By Hicks

```

        MOV     DL,0
        CALL    P_CRLF                ; Else print new line
        INC     DH                    ; ...add another row
        CMP     DH,CH                ; Done all 25 rows?
        JNZ     PS_1                  ; ...no, continue
        MOV     Byte ptr DS:100h,0    ; Show done Print Screen OK

PS_4:   POP     DX                    ; Get saved cursor position
        MOV     AH,2
        INT     10h                  ; ...restore it

PS_5:   POP     DX
        POP     CX
        POP     BX
        POP     AX
        POP     DS
        IRET

        ENTRY   0FFCBh                ; IBM entry, display CR, LF

P_CRLF: PUSH    DX                    ; Print CR, LF, on line
printer
        XOR     DX,DX
        MOV     AH,DL                ; Function=print
        MOV     AL,LF                ; LF
        INT     17h
        MOV     AH,0
        MOV     AL,CR                ; CR
        INT     17h
        POP     DX
        RET

;*****
****
        ENTRY   0FFF0h                ; Hardware power reset
entry *
        PUBLIC  POWER                ; ...ic "8088" or "V20"
*
POWER:   JMPF   0F000h,COLD           ; ...begins here on power
up *
;*****
****

        ENTRY   0FFF5h                ; Release date, Yankee
style
        db     "08/23/87"            ; ...MM/DD/YY (not
logical)

        ENTRY   0FFFEh
        db     0FEh                  ; Computer type (XT)
;        db     ?                    ; Checksum byte
code     ENDS
```

BIOS ASM
By Hicks

;
END