```
IBM. PC. ROM. BIOS.

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************************
;                                                                       *
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.
                                        ;  + (# serial ports) *   512.
                                        ;  + (1 iff toy port) *  4096.
                                        ;  + (# parallel LPT) * 16384.
        db      ?               ; 40:12         ; MFG test flags, unused by us
        dw      ?               ; 40:13         ; Memory size, kilobytes
        db      ?               ; 40:15         ; IPL errors<-table/scratchpad
        db      ?                               ;  ...unused
```

```
;---------------[Keyboard data area]------------;
        db      ?,?             ; 40:17         ; Shift/Alt/etc. keyboard flags
        db      ?               ; 40:19         ; Alt-KEYPAD char. goes here
        dw      ?               ; 40:1A         ;  --> keyboard buffer head
        dw      ?               ; 40:1C         ;  --> keyboard buffer tail
        dw      16 dup(?)       ; 40:1E         ; Keyboard Buffer (Scan,Value)
;---------------[Diskette data area]------------;
        db      ?               ; 40:3E         ; Drive Calibration bits 0 - 3
        db      ?               ; 40:3F         ; Drive Motor(s) on 0-3,7=write
        db      ?               ; 40:40         ; Ticks (18/sec) til motor off
        db      ?               ; 40:41         ; Floppy return code stat byte
                                                ;  1 = bad ic 765 command req.
                                                ;  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
                                                ; 64 = seek operation failed
                                                ;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)
                                                ;  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
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

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

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
       MOV    AL,01000000b              ; Latch value 12h in 8253 clock
       OUT    43h,AL                    ;  ...chip channel 1 counter

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
       MOV    AL,01011000b              ; Refresh auto-init dummy read
       OUT    0Bh,AL                    ;   ...on channel 0 of DMA chip
       MOV    AL,01000001b              ; Block verify
       OUT    0Bh,AL                    ;   ...on channel 1 of DMA chip
       MOV    AL,01000010b              ; Block verify
       OUT    0Bh,AL                    ;   ...on channel 2 of DMA chip
       MOV    AL,01000011b              ; Block verify
       OUT    0Bh,AL                    ;   ...on channel 3 of DMA chip
       MOV    AL,0FFh                   ; Refresh byte count
       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
        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
        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
        MOV     AX,CS                       ;  ...bios ROM segment
        STOSW
        LOOP    LO_VEC

        MOV     SI,offset VECTORS           ; SI --> Vector address table
        MOV     CX,18h                      ;  ... vectors 08h - 1Fh busy

HI_VEC: MOVSW                               ; Get INTERRUPT bios ROM offset
        MOV     AX,CS
        STOSW                               ;  ...INTERRUPT bios ROM segment
        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
        MOV     DI,60h                  ;  ...install basic

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

        PUSH    DS
kosher: 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
        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
        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                             ; No hardware, ports 0FFh
        RET

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

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)

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
        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
        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,81Eh                 ; 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
        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

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
        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
        MOV     DX,[BX]                 ; Convert index to port number
        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
        OUT     DX,AL                   ; Tell LCR to set (latch) baud
        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
        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
        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
                                ;AX=(DEL Clear_To_Send) *    1
                                ;   (DEL Data_Set_ready)*    2
                                ;   (Trailing_Ring_Det.)*    4
                                ;   (DEL Carrier_Detect)*    8
                                ;   (    Clear_To_Send )*   16
                                ;   (    Data_Set_Ready)*   32
```

```
                                   ;  (    Ring_Indicator)*   64
                                   ;  (    Carrier_Detect)*  128
                                   ;       **************
                                   ;  (    Char  received)*  256
                                   ;  (    Char smothered)*  512
                                   ;  (    Parity error  )* 1024
                                   ;  (    Framing error )* 2048
                                   ;  (    Break detected)* 4096
                                   ;  (    Able to xmit  )* 8192
                                   ;  (    Transmit idle )*16384
                                   ;  (    Time out error)*32768

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

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
        JZ      KPD_WT                  ; Set Z if char  ready, AH=1
        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
        INC     BX                      ;  ...point to next character
        INC     BX                      ;  ...char = scan code + shift
        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

        ENTRY   0E885h                  ; Align INT_9 at correct place

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
        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

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

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
        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
```