

# Assembler listing: GETSCAN.ASM

```

;*****;
;*                G E T S C A N                *;
;*-----*
;* Task          : Displays keyboard scan codes before they can *;
;*                be filtered and changed by a keyboard driver. *;
;*-----*
;* Author        : Michael Tischer                    *;
;* Developed on   : 12/08/90                            *;
;* Last update    : 04/07/95                            *;
;*-----*
;* Assembly      : MASM GETSCAN                        *;
;*                LINK GETSCAN                          *;
;*                EXE2BIN GETSCAN GETSCAN.COM           *;
;*                or                                     *;
;*                TASM GETSCAN                          *;
;*                TLINK /T GETSCAN                      *;
;*-----*
;* Call          : GETSCAN                            *;
;*****;

```

== Constants and structures =====

```

INT_CTR   = 20h                ;Interrupt controller port
EOI        = 20h                ;End of interrupt instruction
KB_PORT    = 60h                ;Keyboard port

```

== Program starts here =====

```

code      segment para 'CODE'    ;Define CODE segment

```

```

        org 100h

        assume cs:code, ds:code, es:code, ss:code

start:    jmp  getscan                ;Jump to actual start of program

;== Data =====

int9_ptr  equ this dword              ;Old interrupt vector 9H
int9_ofs  dw 0                       ;Offset address of old handler
int9_seg  dw 0                       ;Segment address of old handler

allscan   db 0                       ;Display release codes as well?

scanbuf   db 32 dup (0)              ;Scan code buffer
scanend   equ this byte

scannext  dw offset scanbuf          ;Next character in scan buffer
scanlast  dw offset scanbuf          ;Last character in scan buffer

copyr     db  "GETSCAN - (c) 1990, 92 by MICHAEL TISCHER",13,10,13,10
          db  "Press any key to see its scan code,",13,10
          db  "or press <Enter> to end the program."
          db  13,10,13,10, "$"

hexdigits db  "0123456789ABCDEF" ;Digits converted to hex numbers

scanmes   db  "Scan code: "          ;Scan code message
scandec   db  "000 ("
scanhex   db  "xx)", 13, 10, "$"

```

```

;== Program code =====

getscan:  mov  ah,09h           ;Display copyright message
          mov  dx,offset copyr
          int  21h

          ;-- /R switch instructs program to display all codes -----

          cmp  word ptr ds:[130], "R" * 256 + "/"
          je   gsall
          cmp  word ptr ds:[130], "r" * 256 + "/"
          jne  gsnall

gsall:    mov  allscan,1        ;Display release codes as well

gsnall:   ;-- Install new keyboard interrupt handler -----

          mov  ax,3509h         ;Get contents of interrupt vector 9H
          int  21h              ;Call DOS interrupt
          mov  int9_seg,es      ;Pass segment and offset addresses
          mov  int9_ofs,bx      ;of interrupt vector 9H

          mov  dx,offset newi9  ;Offset addr. New interrupt routine
          mov  ax,2509h         ;Place interrupt vector 9H
          int  21h              ;in its own routine

gs1:      ;-- Read loop -----

          mov  ah,01h           ;Determine whether a
          int  16h              ;key is ready
          je   gs2              ;No --> GS2

```

```

xor     ah,ah           ;Yes --> Read character
int     16h
cmp     al,13           ;Character = <Enter>?
je      gs4             ;Yes --> End program

gs2:    mov     di,scannext ;Move pointer to next scan code
        cmp     di,scanlast ;Buffer empty?
        je      gs1       ;Yes --> Back to start of loop

        ;== Convert scan code to ASCII and display it =====

        mov     word ptr scandec, 32 shl 8 + 32
        mov     byte ptr scandec+2, 32

        mov     si,offset scandec+2;Make SI the last number in buffer
        mov     al,[di]           ;Load scan code into AL
        mov     bl,10             ;Divisor is always 10

gs3:    xor     ah,ah           ;Divide high byte by 10
        div     bl              ;Divide AX by 10
        or      ah,'0'         ;Change AH to ASCII format
        mov     [si],ah        ;Place in buffer
        dec     si             ;Address next character
        or      al,al          ;Remainder from division?
        jne     gs3            ;Yes --> Next digit

        ;-- Display code in hexadecimal format -----

        mov     bx,offset hexdigits ;Move BX to table with hex numbers
        mov     al,[di]           ;Isolate low nibble of scan code
        and     al,15
        xlat                     ;Get hex number from table

```

```

mov     ah,al                ;Change low digit into high byte

mov     al,[di]              ;Isolate high nibble of scan code
mov     cl,4
shr     al,cl
xlat                     ;Get hex number from table

mov     word ptr scanhex,ax ;Place both digits in buffer

;-- Advance pointer in scan buffer

inc     di
cmp     di,offset scanend    ;Overflow?
jne     gsnowwrap            ;No --> Mark it

mov     di,offset scanbuf    ;Yes --> Back to the start

gsnowrap: mov     scannext,di ;Mark next character position

;-- Display message -----

mov     ah,09h               ;Display string
mov     dx,offset scanmes
int     21h

jmp     gsl                  ;Jump to start of loop

gs4:    ;-- Prepare end of program -----

lds     dx,int9_ptr          ;Back to installing old
mov     ax,2509h             ;keyboard interrupt handler
int     21h

```

```

        mov  ax,4C00h           ;Everything is O.K., end program
        int  21h

;== Interrupt handler 09H (keyboard) =====

newi9    proc far

        assume cs:code, ds:nothing, es:nothing, ss:nothing

        push ax                ;Push AX onto stack
        in   al,KB_PORT        ;Get scan code from keyboard port

        cmp  al,128            ;Release code?
        jnb i9marks           ;No --> Mark it

        cmp  allscan,0         ;Yes --> Marked?
        jne  i9end             ;No --> Return

i9marks:  ;-- Place scan code in buffer -----

        push di                ;Push DI onto stack
        mov  di,scanlast       ;Move DI to next buffer position
        mov  cs:[di],al        ;Place scan code at that location
        inc  di                ;Increment DI to next position
        cmp  di,offset scanend ;Overflow?
        jne  i9nowrap         ;No ---> Mark it

        mov  di,offset scanbuf ;Yes --> Go to beginning

i9nowrap: mov  scanlast,di      ;Mark next character position
        pop  di                ;Pop DI from stack

```

```
i9end:    pop  ax                ;Pass scan code to old  
          jmp  [int9_ptr]        ;keyboard handler
```

```
newi9     endp
```

```
== End =====
```

```
code      ends                ;End of CODE segment  
end       start
```