

```

;*****;
;*          S O U N D C A          *;
;*-----*
;* Task      : Creates a function suitable for inclusion in *;
;*            C codes, which enables C to play notes in the *;
;*            3rd, 4th and 5th PC musical octave.          *;
;*-----*
;* Author    : Michael Tischer                               *;
;* Developed on : 08/15/87                                    *;
;* Last update  : 02/04/92                                    *;
;*-----*
;* assembly   : MASM SOUNDCA;                                *;
;*****;

IGROUP group _text      ;Merging of program segment
DGROUP group const,_bss, _data ;Merging of data segment
      assume CS:IGROUP, DS:DGROUP, ES:DGROUP, SS:DGROUP

      public _Sound      ;Make function public (accessible to
                          ;other programs)

CONST segment word public 'CONST';This segment denotes all read-only
CONST ends               ;constants

_BSS segment word public 'BSS' ;This segment denotes all static, non-
_BSS ends                ;initialized variables

_DATA segment word public 'DATA' ;This segment contains all initialized
                                ;global and static variables

old_time dw (?),(?)        ;Address of old timer interrupt
s_counter db (?)           ;Counts duration of notes in
                          ;1/18 second increments

s_endit db (?)             ;Indicates whether note already played
tones dw 9121,8609,8126,7670 ;Note values for octave 3
      dw 7239,6833,6449,6087
      dw 5746,5423,5119,4831
      dw 4560,4304,4063,3834 ;Note values for octave 4
      dw 3619,3416,3224,3043
      dw 2873,2711,2559,2415
      dw 2280,2152,2031,1917 ;Note values for octave 5
      dw 1809,1715,1612,1521
      dw 1436,1355,1292,1207

_DATA ends

;== Program =====

_TEXT segment byte public 'CODE' ;Program msegment

;-- SOUND: Plays a note -----
;-- Call from C : Sound((int) Note, (int) Duration);
;-- Output      : none
;-- Info        : Note is the number of the note relative to 3rd octave
;--             C
;--             Duration=duration of the note in 1/18-sec. increments

_Sound proc near

      push bp          ;Push BP onto stack
      mov bp,sp        ;Transfer SP to BP

      ;-- Modify timer interrupt for user application -----
      mov word ptr cs:setds+1,ds ;Store DS for new timer interrupt
      mov ax,351ch             ;Get timer interrupt's address
      int 21h                  ;Call DOS interrupt
      mov old_time,bx          ;Note offset address and segment
      mov old_time+2,es        ;address of old interrupt
      mov word ptr cs:stjump+1,bx ;Save for new timer interrupt
      mov word ptr cs:stjump+3,es ;
      mov bx,ds                ;Place DS in BX
      push cs                  ;Push CS onto stack
      pop ds                   ;and pop off DS
      mov dx,offset sound_ti    ;Offset address of new timer routine
      mov ax,251ch             ;Set new timer routine

```

```

        int    21h                ;Call DOS interrupt
        mov    ds,bx              ;Restore DS

        mov    al,182              ;Get ready to generate tone
        out    43h,al             ;Send value to timer command register

        mov    bx,[bp+4]          ;Get note
        xor    bh,bh              ;BH for addressing of note table = 0
        shl    bx,1               ;Divide note number (for word table)
        mov    ax,[tones+bx]      ;Get note value
        out    42h,al             ;Pass low byte to timer counter register
        mov    al,ah              ;Pass high byte to AL
        out    42h,al             ;and to timer counter register
        in     al,61h             ;Read speaker control bit
        or     al,11b             ;Two lowest bits activate speaker
        mov    s_endit,1          ;Still have to play note
        mov    dl,[bp+6]          ;Get note duration
        mov    s_counter,dl       ;and store it
        out    61h,al             ;Turn on speaker

play:    cmp    s_endit,0          ;Note ended?
        jne    play              ;NO --> wait

        in     al,61h             ;Read speaker control bit
        and    al,11111100b       ;Clear two lowest bits to
        out    61h,al             ;disable speaker

        ;-- re-activate original timer interrupt -----
        mov    cx,ds              ;Note DS
        mov    ax,251ch           ;Set function no. for interrupt vector
        lds    dx,dword ptr old_time ;Load old address into DS:DX
        int    21h               ;Call DOS interrupt
        mov    ds,cx              ;Return DS

        mov    sp,bp              ;Restore stack pointer
        pop    bp                 ;Pop BP off of stack
        ret                      ;Return to calling program

_Sound    endp

;-- new timer interrupt -----

sound_ti  proc far                ;Call this 18 times per second

        push    ax                ;Push AX and DS onto stack
        push    ds
setds:    mov    ax,0000h          ;Transfer C to DS
        mov    ds,ax
        dec    s_counter          ;Decrement time counter
        jne    st_endit          ;If still unequal to 0 then end
        mov    s_endit,0         ;Signal end of note duration
st_endit: pop    ds               ;Pop value off of DS (reset to old value)
        pop    ax                 ;Get AX from stack again

stjump:   db     0EAh,0,0,0,0     ;FAR-JUMP to old timer interrupt

sound_ti  endp

;== End =====

_text    ends                    ;End of program segment
        end                      ;End of assembler source

```