

```

;*****
;*                               M A C R O K E Y                               *
;*-----*
;* Task      : Demonstrates the definition of macro keys by *
;*            programming a custom handler using BIOS *
;*            keyboard interrupt 16H. *
;*-----*
;* Author    : Michael Tischer *
;* Developed on : 01/03/92 *
;* Last update : 01/29/92 *
;*-----*
;* Assembly  : MASM MACROKEY *
;*            LINK MACROKEY *
;*            EXE2BIN MACROKEY MACROKEY.COM *
;*            *
;*            or *
;*            *
;*            TASM MACROKEY *
;*            TLINK /T MACROKEY *
;*-----*
;* Call      : MACROKEY *
;*****

```

== Program begins here ==

```

code      segment para 'CODE'      ;Define CODE segment

          org 100h

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

start:    jmp  mcinit              ;Call initialization routine

== Data (keep in memory) ==
olderint  equ this dword          ;Old interrupt vector 16H
intoldoff dw (?)                  ;Offset address interrupt vector 16H
intoldseg dw (?)                  ;Segment address interrupt vector 16H

mkey      dw 3100h                ;Macro keys: ALT + N
mstart    db "'PC Intern' published by Abacus"
          ;Macro text

mend      equ this byte

mofs      dw 0FFFFh               ;Pointer to next character from the
          ;macro text

```

== New BIOS keyboard interrupt 16H (keep in memory) ==

```

newil6    proc far

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

          sti                      ;Re-enable interrupts
          jmp  short nil

          db "MT"                  ;Program identifier

nil:      ;-- First determine whether the new handler applies to the
          ;-- function being called

          or   ah,ah               ;Function 00H or 10H?
          je   fct0
          cmp  ah,10h
          je   fct0

          cmp  ah,01h             ;Function 01H or 11H?
          je   fct1
          cmp  ah,11h
          je   fct1

          jmp  cs:[olderint]       ;If not, then let function be
          ;called by older handler

fct0:     ;-- Function 00H/10H -----

```

```

        cmp  mofs,offset mend      ;Macro already duplicated?
        jae  check0                ;No --> Check for key

        ;-- If macro is already duplicated, get next key -----

fct0p:  push bx                    ;Change BX
        mov  bx,mofs                ;Yes --> Pass next character
        mov  al,cs:[bx]            ;Load ASCII code
        xor  ah,ah                  ;0 as scan code
        inc  mofs                  ;MOFS to next character
        pop  bx                    ;Pop BX from stack

niret0:  iret                      ;Return to caller

check0:  ;-- Check whether macro keys were pressed -----

        pushf                      ;Call old handler
        call cs:[olderint]
        cmp  ax,mkey                ;Macro keys pressed?
        jne  niret0                ;No --> Return to caller

        ;-- Macro keys detected, play the macro -----

        mov  mofs,offset mstart    ;Macro pointer to start of macro
        jmp  fct0p                ;Return first character

fct1:    ;-- Function 01H/11H -----

        cmp  mofs,offset mend      ;Macro already duplicated?
        jae  check1                ;No --> Check for key

        ;-- Macro already duplicated, get next character -----

fct1p:  push bx                    ;Change BX
        mov  bx,mofs                ;Yes --> Pass next character
        mov  al,cs:[bx]            ;Load ASCII code
        xor  ah,ah                  ;0 as scan code
        pop  bx                    ;Pop BX from stack
        cmp  ah,1                  ;Zero flag at 0: Key available

niret1:  ret 2                      ;Return to caller, but
        ;pop flags from stack

check1:  ;-- Check whether macro key has been pressed -----

        pushf                      ;Call old handler
        call cs:[olderint]
        je   niret1                ;No keys in keyboard buffer

        cmp  ax,mkey                ;Macro keys pressed?
        je   check1a               ;Yes --> Play macro

        cmp  ax,0                  ;Zero flag at 0: Character available
        ret 2                      ;Return to caller, but
        ;pop flags from stack

check1a: ;-- Macro key implemented, play macro -----

        xor  ah,ah                  ;Remove macro keys first
        pushf
        call cs:[olderint]

        mov  mofs,offset mstart    ;Move macro pointer to start
        jmp  fct1p                ;Get first character

newil6  endp

;== End of memory resident section =====

instend  equ this byte

;== Data (can be overwritten by DOS) =====

installm db "MACROKEY - (c) 1992 by Michael Tischer", 13, 10, 13, 10
          db "MACROKEY now installed."
          db 13,10, "Call MACROKEY again to de-install.", 13, 10, "$"

```

```

removepg db "MACROKEY de-installed.", 13, 10, "$"

;== Program (can be overwritten by DOS) =====

mcinit label near ;Initialization

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

;-- First determine whether program is already installed ----

mov ax,3516h ;Get contents of interrupt vector 16H
int 21h ;Call DOS function
cmp word ptr es:[bx+3], "TM" ;Test for MACROKEY
jne mcinstall ;Not installed --> Do it now

;-- If program is already installed, stop and remove it ----
;-- from memory

mov dx,es:intoldoff ;Offset address of interrupt 16H
mov ax,es:intoldseg ;Segment address of interrupt 16H
mov ds,ax ;Move to DS
mov ax,2516h ;Put contents of interrupt vector
int 21h ;16H in old routine

;-- Release memory allocated for old MACROKEY -----

mov bx,es ;Mark segment address of program
mov es,es:[2Ch] ;Get environment seg. addr. from PSP
mov ah,49h ;Release old
int 21h ;environment memory

mov es,bx ;Memory for old MACROKEY
mov ah,49h ;Release using DOS
int 21h ;function 49H

push cs ;Store CS on stack
pop ds ;Pop DS from stack

mov ah,09h ;Message: Program removed
mov dx,offset removepg
int 21h

mov ax,4C00h ;End program in normal manner
int 21h

mcinstall: ;-- Install the program -----

mov intoldseg,es ;Pass segment and offset addresses
mov intoldoff,bx ;of interrupt vector 16H

mov dx,offset newil6 ;Offset addr.: New interrupt routine
mov ax,2516h ;Hide contents of interrupt vector
int 21h ;16H in its own routine

mov dx,offset installm ;Message: Program installed
mov ah,09h ;Display function number for string
int 21h ;Call DOS function

;-- PSP, new interrupt routine and corresponding -----
;-- data need be resident

mov dx,offset instend ;Compute the number of 16-byte
add dx,15 ;paragraphs into which the
mov cl,4 ;program should be inserted
shr dx,cl
mov ax,3100h ;End program with end code 0
int 21h ;(O.K.) but keep resident

;== End =====

code ends ;End of CODE segment
end start

```