

```

;*****;
;*                T Y P M C A                *;
;*-----*
;*   Description   : Assembler routine for setting the key repeat *;
;*                  rate on an AT keyboard. For linking with a   *;
;*                  C program.                                   *;
;*-----*
;*   Author        : Michael Tischer                             *;
;*   Developed on   : 08/27/88                                     *;
;*   Last update    : 01/22/92                                     *;
;*-----*
;*   Assembly      : MASM TYPMCA;                                 *;
;*                  ... link with a C program                     *;
;*****;

```

```

;== Constants =====

```

```

KB_STATUS_P    equ 64h           ;Keyboard status port
KB_DATA_P      equ 60h           ;Keyboard data port

OB_FULL        equ 1             ;Bit 0 in keyboard status port
                                   ;A character in the output buffer
IB_FULL        equ 2             ;Bit 1 in the keyboard status port
                                   ;A character in the input buffer

ACK_SIGNAL     equ 0fah          ;Keyboard acknowledge signal
SET_TYPMEM     equ 0f3h          ;Set typematic rate code

MAX_TRY        equ 3             ;Number of retries allowed

```

```

;== Segment declarations for the C program =====

```

```

IGROUP group _text                ;Combination of the program segments
DGROUP group const,_bss, _data    ;Combination of the data segments
    assume CS:IGROUP, DS:DGROUP, ES:DGROUP, SS:DGROUP

CONST segment word public 'CONST';This segment stores all of the
CONST ends                        ;read-only constants

_BSS segment word public 'BSS'    ;This segment stores all of the
_BSS ends                        ;uninitialized static variables

_DATA segment word public 'DATA' ;This segment stores all of the ini-
_DATA ends                       ;tialized global and static variables

;== Program =====

_TEXT segment byte public 'CODE' ;CODE segment

public    _set_ttypm
;-----
;-- SET_TYPM: Sends the key repeat rate to the keyboard controller ----
;-- Call from C : bool set_ttypm( byte trate );
;-- Return value: TRUE if the repeat rate was set
;--              FALSE if an error occurred

_set_ttypm proc near

sframe0 struct                    ;Structure for accessing the stack
bp0      dw ?                    ;Stores BP
ret_adr0 dw ?                    ;Return address to caller
trate0   dw ?                    ;Repeat rate to be set
sframe0 ends                     ;End of structure

```

```

frame      equ [ bp - bp0 ]      ;Address structure elements

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

xor  dx,dx     ;Assume transfer fails
mov  ah,SET_TYPEM ;Set command code for rep rate
cli      ;Disable interrupts
call send_kb   ;Send to the controller
jne  error     ;Error? Yes --> Error routine

mov  ah,byte ptr frame.trate0    ;Get key repeat rate
call send_kb   ;Send to the controller
jne  error     ;Error? Yes --> Error routine

inc  dl        ;Everything O.K., return TRUE

error:      sti      ;Enable interrupts
mov  ax,dx     ;Return value to AX
pop  bp        ;Pop BP from the stack
ret          ;Return to the C program

```

```
_set_ttypm  endp
```

```

;-----
;-- SEND_KB: Sends a byte to the keyboard controller -----
;-- Input      : AH = the byte to be sent
;-- Output     : Zero flag=0: Error
;--             Zero flag=1: O.K.
;-- Registers: AX and FLAGS are affected

```

```

;-- Info      : This routine is to be called only within the module

send_kb  proc near

    push cx                ;Push all registers changed by
    push bx                ;in this routine onto the stack

    mov  bl,MAX_TRY        ;Maximum of MAX_TRY retries

    ;-- Wait until the controller is ready to receive data -----

skb_1:    xor  cx,cx                ;Maximum of 65536 loop passes
skb_2:    in   al,KB_STATUS_P      ;Read contents of status port
    test al,IB_FULLL          ;Still a char in the input buffer?
    loopne skb_2              ;Yes --> SKB_2

    ;-- Send character to the controller -----

    mov  al,ah              ;Get character in AL
    out  KB_DATA_P,al        ;Send character to the data port
skb_3:    in   al,KB_STATUS_P      ;Read contents of the status port
    test al,OB_FULLL        ;Reply in output buffer?
    loope skb_3              ;No --> SKB_3

    ;-- Get and process reply from controller -----

    in   al,KB_DATA_P        ;Read reply from data port
    cmp  al,ACK_SIGNAL        ;Was the character accepted?
    je   skb_end              ;Yes --> Everything O.K.

    ;-- The character was not accepted -----

```

```

        dec    bl                ;Decrement error counter
        jne    skb_2            ;Still retries left?
                                   ;Yes --> SKB_2

        or     bl,1             ;No --> Set zero flag to 0 to
                                   ;         indicate the error

skb_end: pop    bx                ;Pop registers from the stack
        pop    cx
        ret                    ;Return to caller

send_kb    endp

;-----

_text      ends                ;End code segment
        end                ;End program

```