

```

;*****;
;*                T Y P M P A                *;
;*-----*
;*   Task          : Assembler routine for use with a Turbo Pascal *;
;*                  program, which sets the key repeat rate of    *;
;*                  the MF II keyboard.                            *;
;*-----*
;*   Author        : Michael Tischer                               *;
;*   Developed on   : 08/27/88                                       *;
;*   Last update    : 01/22/92                                       *;
;*-----*
;*   Assembly      : MASM TYPMPA;                                    *;
;*                  LINK TYPMPA;                                    *;
;*                  EXE2BIN TYPMPA TYPMPA.BIN                       *;
;*                  ... convert to INLINE statements                *;
;*****;

```

```

;== Constants =====

```

```

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

OB_FULL        equ 1            ;Bit 0 in the 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_TPEM       equ 0f3h         ;Set key repeat code

MAX_TRY        equ 3            ;Number of retries

```



```

        mov  ah,byte ptr frame.trate0    ;Get TYPRATE variable addr.
        call send_kb                     ;Send to the controller
        jne  error                       ;Error? Yes --> Error

        inc  dl                          ;Everything O.K., return TRUE

error:    sti                            ;Enable interrupts
        mov  [bp-1],dl                   ;Put error static there
        pop  bp                          ;Pop BP off of stack
        jmp  ende                        ;Return to Turbo Pascal
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 intended for use only within this
;--           module

send_kb  proc near

        push cx                          ;Push all registers used in this
        push bx                          ;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 the 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        ;Answer in the output buffer?
          loope skb_3            ;No --> SKB_3

          ;-- Get reply from controller and evaluate -----

          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            ;Retries left?
                                   ;Yes --> SKB_2

          or    bl,1             ;NO --> Set zero flag to 0,
                                   ;indicating an error

skb_end:  pop    bx              ;Pop registers off of stack
          pop    cx
          ret                   ;Return to caller

send_kb   endp

```

```
;-----  
ende          label near  
;== End =====  
  
code          ends          ;End code segment  
end  set_tym
```