

```

*****
;*                               P L I N K P A . A S M                               *;
;*-----*
;* Task      : Assembly language module for PLINKP program. *;
;*           : This module contains an interrupt handler, *;
;*           : as well as routines for fast port access. *;
;*-----*
;* Author    : Michael Tischer *;
;* Developed on : 10/10/91 *;
;* Last update  : 01/30/92 *;
;*-----*
;* Assembly   : TASM PLINKPA *;
;*           : ... Link to PLINKP *;
;*****
;

;== Constants =====
KB_PORT    = 60h                ;Keyboard port
INT_CTR    = 20h                ;Interrupt controller port
EOI         = 20h                ;End of Interrupt instruction
ESCAPE     = 1                  ;Scan code for Escape

;== Data =====
DATA      segment word public    ;Turbo data segment

extrn     InpPort                ;Input port as Turbo variable
extrn     OutPort                ;Output port as Turbo variable

DATA      ends                  ;End data segment

;== Program =====
CODE      segment byte public    ;Turbo CODE segment
          assume cs:CODE, ds:DATA, es:nothing, ss:nothing

;-- Public declarations of internal functions -----
public    intr_install           ;Allows call from Turbo program
public    intr_remove
public    escapedirect
public    getb
public    putb

;-- Variables for interrupt handler -----
;-- (accessible from code segment only) -----
key_ptr   dd 0                  ;Pointer to variable for ESCAPE
tout_ptr  dd 0                  ;Pointer to time out counter
escdirect db 0                  ;No time out occurs on escape

;-- The following variables refer to the old interrupt handler --
;-- addresses, which are replaced by the new interrupt handler --
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

int1C_ptr  equ this dword        ;Old interrupt vector 1CH
int1C_ofs  dw 0                  ;Offset address of old handler
int1C_seg  dw 0                  ;Segment address of old handler

;-----
;-- GETB : Reads a byte from the input port
;-- Call from Turbo: getb : BYTE;

getb       proc near

          mov  dx,InpPort         ;Move port address to DX
          in   al,dx              ;Read from port
          and  al,0F8h            ;Mask bits 0-2

          ret                     ;Return - result of function in AL

getb       endp

;-----
;-- PUTB : Writes a byte to the output port
;-- Call from Turbo: putb( TpVar : BYTE );

putb       proc near

TpVar      equ byte ptr [bp+4]    ;Variable passed from Turbo

          push bp                  ;Enable access to arguments

```

```

mov     bp, sp
mov     al, TpVar           ;Move TpVar to al
mov     dx, OutPort         ;Pass OutPort to DX
out     dx, al              ;Pass value to port

pop     bp                  ;Restore BP register
ret     2                   ;Release stack and return

putb    endp

;-----
;-- INTR_INSTALL: Installs the interrupt handler      ---
;-- Call from Turbo: intr_install( escape_flag, timeout_count : ptr );

intr_install proc near

sframe0 struc              ;Access structure on the stack
bp0      dw ?              ;Sets BP
ret_adr0 dw ?              ;Return address
toptr    dd ?              ;FAR pointer to the time out counter
escptr    dd ?             ;FAR pointer to the ESCAPE flag
sframe0 ends              ;End of structure

frame    equ [ bp - bp0 ]

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

;-- Get arguments from stack and process them -----

les     si,frame.escptr     ;Load pointer to ESCAPE flag
mov     word ptr key_ptr,si ;and place in the CODE segment
mov     word ptr key_ptr+2,es ;variables

les     si,frame.toptr      ;Load pointer to time out
mov     word ptr tout_ptr,si ;counter and place in the CODE
mov     word ptr tout_ptr+2,es ;segment variables

;-- Get addresses of the replacement interrupt handler ----

mov     ax,3509h            ;Get interrupt vector 9H
int     21h                ;Call DOS interrupt
mov     int9_ofs,bx         ;Place handler addresses
mov     int9_seg,es         ;in corresponding variables

mov     ax,351Ch            ;Get interrupt vector 1CH
int     21h                ;Call DOS interrupt
mov     int1C_ofs,bx        ;Place handler addresses
mov     int1C_seg,es        ;in corresponding variables

;-- Install new interrupt handler -----

push     ds                 ;Mark data segment
mov     ax,cs               ;Place DS on CS
mov     ds,ax

mov     ax,2509h            ;Func. no.: Set interrupt 9H
mov     dx,offset int09      ;DS:DX receives the handler address
int     21h                ;Call DOS interrupt

mov     ax,251Ch            ;Func. no.: Set interrupt 1CH
mov     dx,offset int1C      ;DS:DX receives the handler address
int     21h                ;Call DOS interrupt

pop     ds                  ;Pop DS from stack
pop     es                  ;Pop ES from stack
pop     bp                  ;Pop BP from stack
ret     8                   ;Return to caller

intr_install endp

;-----
;-- INTR_REMOVE: Disables the interrupt handler
;-- Call from Turbo: intr_remove;

intr_remove proc near

cli                     ;Disable interrupts
push     ds             ;Push DS onto stack

mov     ax,2509h        ;Func. no.: Set INT 9H handler
mov     ds,int9_seg     ;Segment address of old handler
mov     dx,int9_ofs     ;Offset address of old handler
int     21h            ;Re-install old handler

```

```

        mov ax,251Ch          ;Func. no.: Set INT 1CH handler
        mov ds,int1C_seg      ;Segment address of old handler
        mov dx,int1C_ofs      ;Offset address of old handler
        int 21h               ;Re-install old handler

        pop ds                ;Pop DS from stack
        sti                   ;Re-enable interrupts

        ret                   ;Return to caller

intr_remove endp              ;End of procedure

;-----
;-- Escapedirect: Determines whether a time out should occur on -----
;-- an escape
;-- Call from Turbo: procedure Escapedirect( Disconnect : boolean );

Escapedirect proc near

sframe1 struct                ;Access structure on the stack
bpl      dw ?                 ;Set BP
ret_adr1 dw ?                 ;Return address
escflag  dw ?                 ;TRUE or FALSE
sframe1  ends                 ;End of structure

frame    equ [ bp - bpl ]

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

        mov al,byte ptr frame.escflag ;Load flag and
        mov escdirect,al       ;place in CS variable

        pop bp
        ret 2                  ;Return to caller, pop
                                ;arguments from stack

Escapedirect endp

;-----
;-- New interrupt handler follows -----
;-----

        assume CS:CODE, DS:nothing, ES:nothing, SS:nothing

;-- New interrupt 9H handler -----

int09    proc far

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

        cmp al,128             ;Release code?
        jae i9_end             ;Yes --> Don't test first

        cmp al,ESCAPE          ;No --> Is it ESCAPE?
        jne i9_end             ;No --> No, revert to old handler

        ;-- User presses <Esc> -----

        push ds                ;Push DS and SI onto stack
        push si
        lds si,key_ptr         ;Load pointer to ESCAPE
        mov word ptr [si],1    ;Set ESCAPE flag to 1
        cmp escdirect,0        ;Clear time out flag?
        je i9_1                ;No ---> I9_1

        lds si,tout_ptr        ;Yes --> Load time out flag counter
        mov word ptr [si],0    ;Set counter to 0

i9_1:    pop si                 ;Restore DS and SI
        pop ds

        mov al,EOI              ;Display end of interrupt
        out INT_CTR,al

        pop ax                  ;Pop AX from stack
        iret                   ;Return to interrupted program

i9_end:  pop ax                 ;Pop AX from stack
        jmp cs:[int9_ptr]       ;Jump to old handler

int09    endp

;-- New interrupt 1CH handler -----

```

```

int1C      proc far

            push ds                ;Push DS and SI onto stack
            push si
            lds si,tout_ptr        ;Load pointer to time out counter
            cmp word ptr [si],0    ;Counter already set to 0?
            je  no_decr            ;Yes --> No more decrementing

            dec word ptr [si]      ;No --> Decrement

no_decr:    pop si                 ;Pop DS and SI from stack
            pop ds

            jmp cs:[int1C_ptr]     ;Revert to old interrupt handler

int1C      endp

;-----
CODE       ends                  ;End of CODE segment
           end                  ;End of program

```