

```

;*****;
;*          P L I N K C A . A S M          *;
;*-----*
;* Task          : Assembly language module for PLINKC program. *;
;*               : This module contains an interrupt handle,   *;
;*               : as well as routines for fast port access.   *;
;*-----*
;* Author        : Michael Tischer                             *;
;* Developed on   : 10/10/91                                    *;
;* Last update    : 01/31/92                                    *;
;*-----*
;* Memory model   : SMALL                                       *;
;*-----*
;* Assembly      : MASM PLINKCA; or TASM PLINKCA               *;
;*               : ... Link to PLINKC                          *;
;*****;

IGROUP group _text          ;Program segment
DGROUP group _bss, _data    ;Data segment
        assume CS:IGROUP, DS:DGROUP, ES:DGROUP, SS:DGROUP

_BSS    segment word public 'BSS' ;This segment contains all un-
_BSS    ends                     ;initialized static variables

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

_DATA   ends

;== Constants =====

```

```

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

;== Program =====

_TEXT segment byte public 'CODE' ;Code segment

;-- Public declarations of internal functions -----

public      _IntrInstall        ;Enables calls from C programs
public      _IntrRemove
public      _EscapeDirect

;-- Interrupt handler variables -----
;-- (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

```

```

;-----
;-- IntrInstall : Installs the interrupt handler ---
;-- Call from C: void IntrInstall( int far * escape_flag,
;--                               word far * timeout_count );

_IntrInstall  proc near

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

frame        equ [ bp - bp0 ]

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

push si

;-- 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 si
pop bp
ret              ;Return to caller
```

```
_IntrInstall  endp
```

```
;-----  
;-- IntrRemove : Disables the interrupt handler  
;-- Call from C: void IntrRemove( void );
```

```
_IntrRemove  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
```

```
_IntrRemove  endp                ;End of procedure
```

```
;-----  
;-- EscapeDirect: Determines whether a time out should occur on -----  
;--                an escape  
;-- Call from C: void EscapeDirect( int Disconnect );
```

_EscapeDirect proc near

```
sframe1    struc                ;Access structure on the stack
bpl        dw ?                 ;Sets 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        ;Move SP to BP

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

pop bp
ret              ;Return to caller
```

_EscapeDirect endp

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

assume CS:IGROUP, 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 --> 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

;-----

_text      ends            ;End of code segment
end        ;End of program

```