

```

;*****;
;*          C E B H A N D          *;
;*-----*
;* Task      : Forms the basic structure of an assembler *;
;*            program, in which the DOS Ctrl-Break and *;
;*            Critical Error interrupt are captured. *;
;*-----*
;* Author    : Michael Tischer *;
;* Developed on : 09/05/88 *;
;* Last update : 04/07/95 *;
;*-----*
;* Assembly  : MASM CEBHAND; *;
;*            LINK CEBHAND; *;
;*            or *;
;*            TASM CEBHAND *;
;*            TLINK CEBHAND *;
;*-----*
;* call      : CEBHAND *;
;*            (Please leave the disk drive open so that a *;
;*            Critical Error occurs.) *;
;*****;

;== Constants =====

;== Stack =====

stackseg segment para stack 'STACK' ;Define stack segment
dw 256 dup (?) ;The stack is 256 words
stackseg ends ;End of stack segment

;== Data =====

data segment para 'DATA' ;Define data segment

cr_err db 0 ;Becomes 1 if a critical error occurs
;during access to a peripheral device
;(floppy, hard disk, or printer)
cr_typ db 0 ;Error number of the critical error

cr_mes db "Critical error! (A)bort or (R)etry: $"
next_line db 13,10,"$"
end_mes db "Program ended normally.$"
brk_mes db "Program aborted.$"

dat_nam db "A:TEST.DAT",0 ;Name of the test file

data ends ;End of data segment

;== Code =====

code segment para 'CODE' ;Define CODE segment
assume cs:code, ds:data, ss:stackseg

start proc far

;-- Install both interrupt handlers -----

push cs ;Push CS on the stack
pop ds ;and return as DS
mov ax,2523h ;Funct.no.: Set Ctrl-Break handler
mov dx,offset cbreak ;DS:DX now contains handler address
int 21h ;Call DOS interrupt

mov al,24h ;Set interrupt 24H
mov dx,offset cerror ;DS:DX contains new handler address
int 21h ;Call DOS interrupt

mov ax,data ;Move data segment's segment address
mov ds,ax ;to the DS register

;-- You can add your program here -----
;
;
;
;-- For a demonstration, try to open a file -----
;-- on the opened disk drive -----

```

```

dat_open: mov  ah,3dh                ;Function number: Open file
          mov  al,0                  ;File mode: Read-only
          mov  dx,offset dat_nam     ;DS:DX = address of the filename
          int  21h                  ;Call DOS interrupt 21H
          jnc  exit                  ;No error? No --> Jump to exit
          cmp  cr_err,0              ;Critical error?
          je   exit                  ;No --> Jump to exit
          call crit_err              ;A critical error occurred
          jmp  dat_open              ;CRIT_ERR returns only if the
                                   ;operation should be retried
                                   ;(IGNORE is not possible)

          ;-- The handler must not be re-installed before the -----
          ;-- end of the program, since this is done by DOS -----

exit:     mov  ah,9                  ;Function number: Pass string
          mov  dx,offset end_mes     ;DS:DX = address of the message
          int  21h                  ;Call DOS interrupt
          mov  ax,4C00h              ;Funct. no.: End program (ERRCODE=0)
          int  21h                  ;Call DOS interrupt 21H
                                   ;and end the program

start     endp

;-- CRIT_ERR: Called within the program after discovery of a -----
;--          critical error -----

crit_err  proc near

          ;-- Display message and ask for user input -----

ask:      mov  ah,9                  ;Function number: Display string
          mov  dx,offset cr_mes     ;DS:DX = address of the message
          int  21h                  ;Call DOS interrupt
          mov  ah,1                  ;Function number: Input character
          int  21h                  ;Call DOS interrupt
          push ax                    ;Note the input
          mov  ah,9                  ;Function number: Display string
          mov  dx,offset next_line ;DS:DX = address of the message
          int  21h                  ;Call DOS interrupt

          ;-- Interpret the user's input -----

          pop  ax                    ;Retrieve the input
          cmp  al,"A"                ;Abort?
          je   end_up                ;Go to "end_up" procedure
          cmp  al,"a"                ;Abort?
          je   end_up                ;Go to "end_up" procedure
          cmp  al,"r"                ;Retry?
          je   crend                 ;Go to end of procedure
          cmp  al,"R"                ;Retry?
          jne  ask                   ;No --> Ask again

crend:    ret                        ;Return to caller

crit_err  endp

;-- END_UP: Executes a "clean" ending -----

end_up    proc near
          ;-- All opened files can be closed and the system -----
          ;-- memory allocated by the program can be freed here -----
          ;
          ;
          ;
          mov  ah,9                  ;Function number: Display string
          mov  dx,offset brk_mes     ;DS:DX = address of the message
          int  21h                  ;Call DOS interrupt
          mov  ax,4C00h              ;End the program normally with the
          int  21h                  ;4CH function

end_up    endp

;-- CBREAK: The new Ctrl-Break handler -----

cbreak    proc far

```

```

;-- All registers altered within this routine (excluding ----
;-- the Flag register) have to be secured on the stack ----

push ds
mov  ax,data           ;Move the segment address of the
mov  ds,ax             ;data segment in the DS-Register

;-- For example, you can open a window here in which the ----
;-- user is asked if he really wants to end the program ----
;
;
;
jmp  go_on             ;Don't end program

;-- If the user decides to end the program, a routine with --
;-- which the program can be ended can be started here --
jmp  end_up            ;Prepare to end program

;-- The program should not be aborted, continue normal ----
;-- execution ----

go_on:  pop  ds         ;Restore saved register
        iret           ;Back to DOS, where the interrupted
                        ;function continues
cbreak  endp

;-- CERROR: the new Critical Error handler -----
cerror  proc far
;-- Each of the registers (SS, SP, DX, ES, DX, CX und BX) --
;-- altered within this routine must be saved on the stack --

        sti           ;Enable interrupts
        push ds

        mov  ax,data   ;Load data segment's segment address
        mov  ds,ax     ;in the DS register

        mov  cr_err,1  ;Point to critical error
        mov  ax,di     ;Move error number to AX
        mov  cr_typ,al ;Note error number
        mov  al,3      ;End function call with error

        pop  ds        ;Pop DS from stack
        iret

cerror  endp

;-----
code    ends           ;End of CODE segment
        end  start     ;Start program execution with
                        ;the START procedure

```