

```

;*****;
;*                H M A C A . A S M                *;
;*-----*
;* Task          : Provides extended memory routines for HMAC.C, *;
;*               as well as control for address line A20.      *;
;*-----*
;* Author        : Michael Tischer                          *;
;* Developed on   : 07/28/90                                  *;
;* Last update    : 03/04/92                                  *;
;*-----*
;* Assembly      : MASM HAMCA; or TASM HMACA;                *;
;*               Link to HMAC.C                               *;
;*****;

```

```

;== Constants =====

```

```

KB_COMMAND equ 64h           ;Keyboard command port
KB_STATUS  equ 64h           ;is also status port
KB_DATA    equ 60h           ;Keyboard data port
IB_FREE    equ 2             ;Mask for testing free input buffer
WO_COMMAND equ 0D1h          ;Writes to output port

```

```

GATE_ON    equ 11011111b     ;A20 line free
GATE_OFF   equ 11011101b     ;A20 line busy

```

```

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

```

```

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

CONST segment word public 'CONST';Readable constants

```

```

CONST  ends

_BSS   segment word public 'BSS'  ;Un-initialized static variables
_BSS   ends

_DATA  segment word public 'DATA' ;Initialized static and global vars.

_DATA  ends

;== Program =====

_TEXT  segment byte public 'CODE' ;Program

public      _HMAAvail
public      _GateA20
public      _IsA20On

;-----
;-- HMAAvail : Determines whether an HMA is available
;-- Call from C: BOOL HMAAvail( void )
;-- Return val. : TRUE if HMA is available, otherwise FALSE

_HMAAvail  proc near

    ;-- Test for AT or 386 -----

    xor     ax,ax                ;Check for the existence of
    push    ax                  ;8088 or 8086 processor
    popf
    pushf
    pop     ax

```

```

and    ax,0F000h
cmp    ax,0F000h
je     nonhma           ;Normal PC ---> nonhma

;-- AT or 386, but does it have extended memory? -----

mov    ah,88h           ;Read extended memeory size using
int     15h             ;BIOS
cmp    ax,64            ;> 64K of extended memory?
jb     nonhma           ;No --> No HMA

mov    ax,0FFFFh        ;Yes --> HMA ready
ret                     ;Return TRUE

nonhma:  xor    ax,ax     ;Return 0 FALSE
        ret            ;Return to caller

_HMAAvail  endp

;-----
;-- GateA20 : Enables or disables A20 line
;-- Call from C: BOOL GateA20( BOOL free )
;-- Return val. : TRUE if operation successful, otherwise FALSE

_GateA20  proc near

;-- Structure for easy access to parameters -----

sframe   struc          ;Structure for stack access
bp0      dw ?           ;Gets BP
ret_adr  dw ?           ;Return address to caller
frei     dw ?           ;Enable or disable line

```

```

sframe      ends                      ;End structure

frame       equ [ bp - bp0 ]          ;Addresses structure elements

;-- Macro for reading KB controller status -----

kbc_ready   macro                      ;Keyboard controller ready?
local      notready

notready:   in      al,KB_STATUS        ;Read status port
            test    al,IB_FREE          ;Input buffer free?
            loopne  notready            ;No, and CX still not null -->

            endm

;-- Procedure code begins here -----

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

            mov     ah,11011101b        ;Free line
            cmp     frame.frei,0        ;Line freed?
            je      g1                  ;Yes --> Code is O.K.

            mov     ah,11011111b        ;No --> Load Line free code

g1:          xor     cx,cx               ;Time out counter
            cli                      ;Disable interrupts
            kbc_ready                  ;Wait for keyboard controller
            jne     gerr                ;Time out? Yes --> GERR

            mov     al,WO_COMMAND        ;Send code for output port access

```

```

        out     KB_COMMAND,al    ;to command port

        kbc_ready                ;Wait for keyboard controller
        jne     gerr             ;Time out? Yes --> GERR

        mov     al,ah            ;Send command to
        out     KB_DATA,al       ;data port

        kbc_ready                ;Wait for keyboard controller
        sti                     ;Enable interrupts
        jne     gerr             ;Time out? Yes --> GERR

        mov     ax,0FFFFh        ;No time out, O.K.
        pop     bp               ;Pop BP from stack
        ret

gerr:    xor     ax,ax           ;No toggle possible
        pop     bp               ;Pop BP from stack
        ret

```

```

_GateA20    endp

```

```

;-----
;-- IsA20On : Determines whether A20 is free
;-- Call from C: BOOL IsA20( void )
;-- Return val. : TRUE if line is free, otherwise FALSE

```

```

ramptr    dd 00000000h          ;Pointer to conventional RAM
extptr    dd 0FFFF0010h        ;Pointer to extended memory

```

```

_IsA20On  proc near

```

```

        push    ds
        push    es

        lds     si,cs:ramptr    ;Line shows identical memory locations
        les     di,cs:extptr    ;on both lines

        mov     cx,64           ;Compare 128 bytes
        cld                     ;Increment on string inst.
        repe    cmpsw          ;Compare ranges

        pop     es              ;Pop registers
        pop     ds
        jcxz    a20off         ;CX = 0 --> Ranges identical

        mov     ax,0FFFFh      ;Range unequal --> A20 active
        ret

a20off:  xor     ax,ax          ;Range equal --> A20 disabled
        ret

_IsA20On  endp

;-----

_text    ends                  ;End code segment
        end                ;End program

```