

Pascal listing: HMAP.PAS

```
{*****
*                                     H M A P . P A S                                     *
**-----**
* Description      : Demonstration of directly accessing the HMA without*
*                  the assistance of any special drivers .             *
**-----**
* Author          : MICHAEL TISCHER                                     *
* Developed on    : 07/27/1990                                         *
* Last update     : 04/07/1995                                         *
*****}
```

program HMAP;

uses Crt; { for ClrScr }

```
{*****
* HMAAvail : Check for 80286 or higher processor and if               *
*           at least 64 KB exdtended memory exists                   *
**-----**
* Input    : none                                                    *
* Output    : TRUE, when the HMA exists, else FALSE                  *
* Info      : - The call of this function must precede the call of   *
*             all other procedures and function of the program       *
*****}
```

function HMAAvail : boolean;

begin

```
    inline (
        $33/$C0/                { xor    ax,ax                }
        $50/                     { push   ax                    }
```

\$9D/	{ popf	}
\$9C/	{ pushf	}
\$58/	{ pop	ax
\$25/\$00/\$F0/	and	ax,0F000h
\$3D/\$00/\$F0/	cmp	ax,0F000h
\$74/\$0E/	je	no_hma >-----+
\$B4/\$88/	mov	ah,88h
\$CD/\$15/	int	15h
\$3D/\$40/\$00/	cmp	ax,64
\$72/\$05/	jb	no_hma >-----!
\$B8/\$01/\$00/	mov	ax,0001h
\$EB/\$02/	jmp	ende
\$33/\$C0/	xor	ax,ax <-----+
\$88/\$46/\$FF	movb	[bp-1],al

);

end;

```

{*****
* GateA20 : Locks the address line A20 or frees it *
**-----**
* Input   : FREE = TRUE, when the line is free *
* Output  : TRUE, when access to the keyboard controller is desired *
*          else FALSE *
* Info    : - After calling this function, you can with the help of *
*            Function IsA20On test to see if the address sline is fre, *
*            since this is only possible on machine with and ISA-Bus *
*****}

```

```
function GateA20( FREE: boolean ) : boolean;
```

```
begin
  inline (
```

```

$B4/$DD/
$83/$7E/$04/$00/
$74/$02/
$B4/$DF/
$33/$C9/
$FA/
$E4/$64/
$A8/$02/
$E0/$FA/
$75/$1D/
$B0/$D1/
$E6/$64/
$E4/$64/
$A8/$02/
$E0/$FA/
$75/$11/
$8A/$C4/
$E6/$60/
$E4/$64/
$A8/$02/
$E0/$FA/
$75/$05/
$B8/$01/$00/
$EB/$02/
$33/$C0/
$FB/
$88/$46/$FF
);

```

```

{ mov    ah,11011101b      }
{ cmp    FREE,0            }
{ je     g1 -----+      }
{ mov    ah,11011111b      }
{ xor    cx,cx <-----+  }
{ cli                               }
{ in     al,64 <-----+   }
{ test   al,02              }
{ loopnz -----+         }
{ jne    gerr ----->+   }
{ mov    al,WO_COMMAND      }
{ out    KB_COMMAND,al      }
{ in     al,64 <-----+   }
{ test   al,02              }
{ loopnz -----+         }
{ jne    gerr ----->-   }
{ mov    al,ah              }
{ out    KB_DATA,al         }
{ in     al,64 <-----+   }
{ test   al,02              }
{ loopnz -----+         }
{ jne    gerr ----->-   }
{ mov    ax,0001h           }
{ jmp    ende               }
{ xor    ax,ax <-----+  }
{ sti                               }
{ mov    [bp-1],al          }

```

end;

```

{ *****
* IsA20On : Check, is address line A20 available
*

```

```

**-----**
* Input      : none                                     *
* Output     : TRUE, when the line is free, else FALSE *
*****}

```

```
function IsA20On : boolean;
```

```

begin
  inline (
    $1E/                                     { push    ds                                     }
    $06/                                     { push    es                                     }
    $33/$F6/                                { xor     si,si                                 }
    $8E/$DE/                                { mov     ds,si                                 }
    $BF/$10/$00/                             { mov     di,0010                              }
    $B8/$FF/$FF/                             { mov     ax,FFFF                              }
    $8E/$C0/                                { mov     es,ax                                }
    $B9/$40/$00/                             { mov     cx,64                                }
    $FC/                                     { cld                                           }
    $F3/$A7/                                { repe    cmpsw                                }
    $07/                                     { pop     es                                     }
    $1F/                                     { pop     ds                                     }
    $E3/$05/                                { jcxz    a20off -----+                       }
    $B8/$01/$00/                             { mov     ax,0001h                             }
    $EB/$02/                                { jmp     ende                                 }
    $33/$C0/                                { xor     ax,ax <-----+                       }
    $88/$46/$FF                             { mov     [bp-1],al                             }
  );
end;

```

```

{ *****
* HMA Test : Demonstration of accessing the HMA. *
**-----**

```

```

* Input      : none
*****}

procedure HMAtest;

type HMAR      = array [1..65520] of BYTE;           { the HMA-Array }
    HMAPTR = ^HMAR;                                { Pointer to the HMA-Array }

var hmap      : HMAPTR;                             { Pointer of the HMA }
    i,        : word;                               { loop counter }
    err       : word;                               { Number of error of HMA access }
    dummy     : boolean;

begin
    if ( IsA20On ) then
        writeln( 'The address line A20 is already switched on!' )
    else
        if ( GateA20( TRUE ) = FALSE ) or ( IsA20On = FALSE ) then
            begin
                writeln( 'Note! Address line A20 can not be switched' +
                    'on.' );
                exit;
            end
        else
            writeln( 'The access to the H HMA is switched on.' );

    hmap := HMAPTR(Ptr( $FFFF, $0010 ));              { Pointer to HMA }

    err := 0;                                         { we start with no n errors }
    for i := 1 to 65520 do                          { each memory location will be tested }
        begin
            write( #13, 'Memory location: ', i );

```

```

        hmap^[i] := i mod 256;                { Memory location description }
        if ( hmap^[i] <> i mod 256 ) then      { and return selection }
            begin                             { Error! }
                writeln( ' ERROR!' );
                inc( err );
            end;
        end;

    end;

writeln( #13 );
if ( err = 0 ) then                          { Output the test results }
    writeln( 'HMA ok, no defective memory locations.' )
else
    writeln( 'NOTE! ', err, ' Defective memory location in ' +
            'the HMA found! ');

    dummy := GateA20( FALSE );                { Address line release }
end;

{ *****
*                                     M A I N   P R O G R A M                                     *
***** }

begin
    writeln( 'HMAP - HMA-Demo program by MICHAEL TISCHER'#10 );
    if HMAAvail then
        begin
            HMAtest;                          { HMA test }
            writeln;
        end
    else
        writeln( 'No access to HMA possible.' );
    end.

```

