

```

{
*****
*                               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     : 07/29/1990
*****
}

```

```

program HMAP;

```

```

uses Crt;                                { for ClnScr }

```

```

{
*****
* HMAAvail : Check for 80286 or higher processor and if
*           at least 64 KB extended 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    >ÄÄÄÄÄÄ¿ }
    $B4/$88/        { mov    ah,88h          }
    $CD/$15/        { int     15h            }
    $3D/$40/$00/    { cmp    ax,64            }
    $72/$05/        { jb     no_hma    >ÄÄÄÄÄÄ´ }
    $B8/$01/$00/    { mov    ax,0001h          }
    $EB/$02/        { jmp     ende            }
    $33/$C0/        { xor    ax,ax    <ÄÄÄÄÄÄÄÜ }
    $88/$46/$FF     { mov8   [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 line is free
*               since this is only possible on machine with an ISA bus
*****
}

```

```

function GateA20( FREE: boolean ) : boolean;

```

```

begin
  inline (
    $B4/$DD/      { mov    ah,11011101b      }
    $83/$7E/$04/$00/ { cmp    FREE,0          }
    $74/$02/      { je     g1    ÄÄÄÄÄÄÄÄÄÄ¿ }
    $B4/$DF/      { mov    ah,11011111b      }
    $33/$C9/      { xor    cx,cx    <ÄÄÄÄÄÄÄÜ }
    $FA/          { cli                      }
    $E4/$64/      { in     al,64    <ÄÄÄÄÄÄÄ¿ }
    $A8/$02/      { test   al,02          }
    $E0/$FA/      { loopnz ÄÄÄÄÄÄÄÄÄÄÄÜ }
    $75/$1D/      { jne    gerr    ÄÄÄÄÄÄÄÄÄÄ>Ä¿ }
    $B0/$D1/      { mov    al,WO_COMMAND    }
    $E6/$64/      { out     KB_COMMAND,al    }
    $E4/$64/      { in     al,64    <ÄÄÄÄÄÄÄ¿ }
    $A8/$02/      { test   al,02          }
    $E0/$FA/      { loopnz ÄÄÄÄÄÄÄÄÄÄÄÜ }
    $75/$11/      { jne    gerr    ÄÄÄÄÄÄÄÄÄÄ>Ä´ }
    $8A/$C4/      { mov    al,ah          }
    $E6/$60/      { out     KB_DATA,al      }
    $E4/$64/      { in     al,64    <ÄÄÄÄÄÄÄ¿ }
    $A8/$02/      { test   al,02          }
    $E0/$FA/      { loopnz ÄÄÄÄÄÄÄÄÄÄÄÜ }
    $75/$05/      { jne    gerr    ÄÄÄÄÄÄÄÄÄÄ>Ä´ }
    $B8/$01/$00/  { mov    ax,0001h          }
  );
end;

```

```

        $EB/$02/          { jmp     ende3 }
        $33/$C0/          { xor     ax,ax <ÄÄÄÄÄÄÄÄÄÜ }
        $FB/              { sti     }
        $88/$46/$FF       { mov     [bp-1],al }
    );

end;

{*****
* Isa200n : Check, is address line A20 available
*-----*
* Input   : none
* Output  : TRUE, when the line is free, else FALSE
*-----*}

function Isa200n : 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 ÄÄÄÄÄÄÄÄ }
        $B8/$01/$00/      { mov     ax,0001h3 }
        $EB/$02/          { jmp     ende3 }
        $33/$C0/          { xor     ax,ax <ÄÄÄÄÄÄÄÜ }
        $88/$46/$FF       { mov     [bp-1],al }
    );

end;

{*****
* HMAtest : Demonstration of accessing the HMA.
*-----*
* Input   : none
*-----*}

procedure HMAtest;

type HMAR      = array [1..65520] of BYTE;
    HMAPTR = ^HMAR;

var hmap      : HMAPTR;
    i,
    err       : word;
    dummy     : boolean;

{ the HMA-Array }
{ Pointer to the HMA-Array }
{ Pointer of the HMA }
{ loop counter }
{ Number of error of HMA access }

begin
    if ( Isa200n ) then
        writeln( 'The address line A20 is already switched on!' )
    else
        if ( GateA20( TRUE ) = FALSE ) or ( Isa200n = FALSE ) then
            begin
                writeln( 'Note! Address line A20 can not be switched' +
                    'on.' );
                exit;
            end
        else
            writeln( 'The access to the HMA is switched on.' );

hmap := HMAPTR(Ptr( $FFFF, $0010 ));

err := 0;
for i := 1 to 65520 do
    begin
        write( #13, 'Memory location: ', i );
        hmap^[i] := i mod 256;
        if ( hmap^[i] <> i mod 256 ) then
            begin
                writeln( ' ERROR!' );
                inc( err );
            end
        end;
    end;

writeln( #13 );
if ( err = 0 ) then
    writeln( 'HMA ok, no defective memory locations.' )
else
    writeln( 'NOTE!', err, ' Defective memory location in ' +

```

```

        'the HMA found!');
dummy := GateA20( FALSE );
end;
{ Address line release }

{*****}
*      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;
writeln;
end
else
writeln( 'No access to HMA possible.' );
end.
{ HMA test }

```