

Experiment N° 11**Using the Mouse****Introduction:**

The mouse is an I/O device that replaces the arrow keys on the keyboard for graphical and text style programs. This experiment shows how to add the mouse to applications through a series of macros that enable and allow the mouse to function.

Objectives:

- 1- Develop macros that detect the mouse and enable it for applications.
- 2- Develop macros that track the mouse position and test button status.
- 3- Use the mouse in simple programs.

References:

1. Barry B. Brey, "Programming the 80286, 80386, 80486, and Pentium-Based Personal Computer", Prentice Hall, (1996).

INT 33H:

The mouse is controlled through INT 33H function call instructions. There are actually more than 50 functions for mouse control. However, we will limit ourselves to the most commonly used functions. These functions are listed in Table 11. .

Function	Description	Entry	Exit
00	Reset Mouse	AH = 00H	BX = Number of Mouse Buttons
01	Show Mouse Cursor	AH = 01H	Displays the mouse cursor
02	Hide Mouse Cursor	AH = 02H	Hides the mouse cursor
03	Read Mouse Status on the fly	AH = 03H	BX = Button Status CX= Horizontal Cursor Position DX= Vertical Cursor Position
04	Set Mouse Cursor Position	AH = 04H CX= Horizontal Cursor Position DX= Vertical Cursor Position	
05	Get Button Press Information	AH = 05H BX= Desired button 0 for left and 1 for right	AX = Button Status BX = Number of presses CX= Horizontal Position of Last Press DX= Vertical Position of Last Press

Table 11. 1:Mouse (INT 33H) Functions

Test the mouse:

To be able to use the mouse it must be first tested whether it is present or not. To detect the presence of the mouse and be able to use it, the following steps are to be followed.

Step 1: Test Interrupt Vector 33H to see if it contains a value other than zero. A zero indicates that the mouse driver has not been installed yet.

Step 2: If the vector is not zero, check if it points to an IRET (value CFH) instruction. For some operating systems, an IRET indicates that the vector is unused.

Step 3: If the vector is neither zero nor does it point to an IRET instruction, then use the following code to test for the presence of the mouse.

```
MOV AX, 0000
INT 33H
```

If a zero is returned in AX, there is no mouse otherwise, the mouse is present.

The following MACRO tests for the presence of the mouse:

```
MP    MACRO                                ;Is mouse present?
      LOCAL M1, M2, M3
      PUSH ES
      MOV AX, 3533H                        ;Read vector 33H
      INT 21H
      MOV AX, ES
      OR AX, BX                            ;Test for ES:BX= 00
      JZ M2
      CMP BYTE PTR[BX], 0CFH              ;Test for 0CFH
      JZ M2                                ;If not, end macro
      MOV AX, 0000                        ;Start mouse
      INT 33H
      OR AX, AX
      JZ M2                                ;No mouse
      CLC                                  ;If mouse, Carry = 0
      JMP M3
M1    DB 13, 10, '*** MOUSE PRESENT ***'

M2:   PUSH DS
      MOV AX, CS
      MOV DS, AX
      ;DISPLAY M1                          ;Show no mouse
      POP DS
      STC                                  ;If no mouse, carry =1
M3:   POP ES
ENDM
```

Enabling the Mouse:

The presence of a mouse does not mean that it can be used, unless it is enabled. The mouse cursor is enabled with INT 33H function number 01H, and disabled with function number 02H. Neither of these functions returns any information to the caller. The following macros (see below) turn the cursor ON and OFF. The mouse cursor is off until the mouse driver is enabled. If the mouse cursor is enabled and data are displayed to the screen, the computer places a copy of the mouse pointer on the screen. If n items are displayed on the screen, the mouse pointer is also displayed n times. To avoid this problem, the mouse pointer should always be turned OFF before updating the video information, and then turned ON after the update is complete.

```

MON  MACRO                ;Enable Mouse Pointer
      MOV AX, 0001H
      INT 33H
ENDM

```

```

MOFF MACRO                ;Disable Mouse Pointer
      MOV AX, 0002H
      INT 33H
ENDM

```

Tracking the Mouse Button:

Mouse INT 33H function number 5 returns the button information and position of the last press. When called AX = 5 and BX = the button being tested = 0, 1 and 4 for respectively left, right and middle in case of a three button mouse. On return from function 5, AX gives the button status, i.e. if a button is being pressed.

```

Bit 0  = 1 for the left button
Bit 1  = 1 for the right button
Bit 2  = 1 for the middle button
BX     = number of times the button has been pressed, since the last
        time this function was called.
CX     = horizontal position
DX     = vertical position

```

The following Macro is used for the above purpose:

```

MBUT  MACRO NUM           ;Read Button
      MOV AX, 0005H       ;NUM = 0 for left
      MOV BX, NUM         ; NUM = 1 for right
      INT 33H            ; NUM = 4 for middle
ENDM

```

Tracking the Mouse Position:

In the 80x25-text mode, the values in CX range from 0 to 632 and the values in DX range from 0 to 192 by increments of 8. As an example line 1 position 3 returns CX = 8 and DX = 24. Function 5 returns the mouse cursor position at the most recent button press, whereas function 3 returns the mouse position on the fly, i.e. in real-time, as it occurs. The following macro is used for that purpose.

```

MRTIME    MACRO NUM           ;Read Mouse Status
          MOV AX, 0003H
          INT 33H
ENDM

```

The Mouse in Graphics Mode:

To have a good understanding of how the mouse works in video mode, it is of benefit to try program 11.2.

To move the mouse cursor to position X (horizontal) and Y (vertical), use INT 10H function 02H.

Function	Description	Entry
02H	Move Mouse Cursor	AH = 02 DH = Line Number DL = Column Number

Table 11. 2: Move Cursor Function

Pre Lab Work:

- 1- Write all macros given in the manual, and add them to your MACROS.INC file.
- 2- Write a program that tests the presence of the mouse using the macros given in the text.
- 3- Write a program that displays the word LEFT if the left button is pressed and RIGHT if the right button is pressed. Exit the program if AX indicates that both left and right buttons are pressed together. Do not forget to turn off the mouse pointer before displaying LEFT or RIGHT, and turn it back on afterwards.
- 4- Bring your work to the lab.

Lab Work:

- 1- Write, link and run program 11-1. Compare the mouse pointer generated in graphics mode with the pointer generated in text mode.
- 2- Modify Program 11.1, so that it displays the mouse position on the top right corner of the screen. Call this program 11.3

Assignment:

Write a program that displays a green square on the middle of the screen. Use the mouse, so that when the mouse enters the square, the color of the square changes to red. Modify the above program, so that when the mouse is inside the square and you want to leave the square red just press the left button.

```

TITLE "Program 11-1"
INCLUDE MACROS.INC
.MODEL SMALL
.STACK 200H
.DATA
.CODE
.STARTUP
    MOV AX, 12H                ;Switch to mode 12H
    INT 10H
    MP                        ;Test for mouse
    JC MAIN2                  ;If no mouse
    MON                       ;Enable mouse pointer

MAIN1:    MRTIME                ;Read Mouse Status on-the-fly
          CMP BX, 3            ;Test for left and right buttons
          JNE MAIN1           ;If both not pressed repeat

MAIN2:    MOFF                  ;Disable mouse pointer
          MOV AX, 03H         ;Switch to mode 3
          INT 10H

.EXIT
END

```

```

TITLE "Program 11-2"
;a program that displays the mouse pointer and its X and Y
;position in text mode.
;
    .MODEL SMALL
    .DATA
MES    DB    13, 'X Position = '
MX     DB    '      '
      DB    'Y Position = '
MY     DB    '      $'
X      DW    ?                ;X position
Y      DW    ?                ;Y position
    .CODE
    .STARTUP
        CALL TM_ON            ;enable mouse
        JC    MAIN4           ;if no mouse
MAIN1:  MOV    AX, 3            ;get mouse status
        INT    33H
        CMP    BX, 1
        JE    MAIN3           ;if left button pressed

        CMP    CX, X
        JNE    MAIN2           ;if X position changed
        CMP    DX, Y
        JE    MAIN1           ;if Y position did not change
MAIN2:  MOV    X, CX            ;save new position
        MOV    Y, DX
        MOV    DI, OFFSET MX
        MOV    AX, CX
        CALL  PLACE            ;store ASCII X
        MOV    DI

```

```

,OFFSET MY
    MOV    AX,Y
    CALL  PLACE        ;store ASCII Y
    MOV    AX,2
    INT   33H          ;hide mouse pointer

    MOV    AH,9
    MOV    DX,OFFSET MES
    INT   21H          ;display position

    MOV    AX,1
    INT   33H          ;show mouse pointer

    JMP   MAIN1        ;do again
MAIN3:
    MOV    AX,0        ;reset mouse
    INT   33H
MAIN4:
    .EXIT
;
;procedure that tests for the presence of a mouse driver
;***Output parameters***
;Carry = 1, if no mouse present
;Carry = 0, if mouse is present
;
CHKM PROC NEAR
    MOV    AX,3533H    ;get INT 33H vector
    INT   21H          ;returns vector in ES:BX
    MOV    AX,ES
    OR     AX,BX       ;test for 0000:0000
    STC
    JZ     CHKM1       ;if no mouse driver
    CMP    BYTE PTR ES:[BX],0CFH
    STC
    JE     CHKM1       ;if no mouse driver
    MOV    AX,0
    INT   33H          ;reset mouse
    CMP    AX,0
    STC
    JZ     CHKM1       ;if no mouse
    CLC
CHKM1:
    RET
CHKM ENDP

;the TM_ON procedure tests for the presence of a mouse
;and enables mouse pointer.
;uses the CHKM (check for mouse) procedure
;***output parameters***
;Carry = 0, if mouse is present pointer enabled
;Carry = 1, if no mouse present

TM_ON PROC NEAR
    CALL  CHKM         ;test for mouse
    JC    TM_ON1
    MOV    AX,1        ;show mouse pointer
    INT   33H
    CLC
TM_ON1:
    RET
TM_ON ENDP

```

```
;The PLACE procedure converts the contents of AX into a
;decimal ASCII coded number stored at the memory location
;addressed by DS:DI
;***input parameters***
;AX = number to be converted to decimal ASCII code
;DS:DI = address where number is stored
;
PLACE PROC NEAR

    MOV    CX,0        ;clear count
    MOV    BX,10       ;set divisor
PLACE1:
    MOV    DX,0        ;clear DX
    DIV    BX          ;divide by 10
    PUSH  DX
    INC    CX
    CMP    AX,0
    JNE    PLACE1     ;repeat until quotient 0
PLACE2:
    MOV    BX,5
    SUB    BX,CX
PLACE3:
    POP    DX
    ADD    DL,30H      ;convert to ASCII
    MOV    [DI],DL    ;store digit
    INC    DI
    LOOP  PLACE3
    CMP    BX,0
    JE     PLACE5
    MOV    CX,BX
PLACE4:
    MOV    BYTE PTR [DI],20H
    INC    DI
    LOOP  PLACE4
PLACE5:
    RET

PLACE ENDP
END
```