Chapter 15

FACILITIES FOR USING THE MOUSE

Objective: To describe the programming requirements for using the mouse.

INTRODUCTION

This chapter describes the use of the mouse: initializing it, displaying and concealing the mouse pointer, setting the pointer's location and limits, and getting button information. Two program examples illustrate the use of mouse handling. The only new instruction introduced is INT 33H for mouse handling.

The mouse is a commonly used pointing device controlled by a software interface known as a device driver that is normally installed by an entry in the CONFIG.SYS or AUTOEXEC.BAT file. The driver must be installed so that a program can recognize and respond to the mouse's actions.

Some basic mouse definitions follow:

- *Pixel:* The smallest addressable element on a screen. For text mode 03, for example, there are eight pixels per byte.
- *Mouse pointer:* In text mode, the pointer is a flashing block in reverse video; in graphics mode, the pointer is an arrowhead.
- *Mickey:* A unit of measure for movement of the mouse, approximately 1/200 of an inch.
- *Mickey count:* The number of mickeys the mouse ball rolls horizontally or vertically. The mouse driver uses the mickey count to move the pointer on the screen a certain number of pixels.
- *Threshold speed:* The speed in mickeys per second that the mouse must move to double the speed of the pointer on the screen. The default is 64 mickeys per second.

All mouse operations within a program are performed by standard INT 33H functions, of the form

```
MOV   AX, function     ;Request mouse function
...                    ;Parameters (if any)
INT   33H              ;Call mouse driver
```

Note that unlike other INT operations that use the AH register, INT 33H functions are loaded in the *full AX register*.

The first mouse instruction that a program issues should be function 00H, which simply initializes the interface between the mouse driver and the program. Typically, you need issue this command just once, at the start of the program. Following function 00H, the program should execute function 01H, which causes the mouse pointer to appear on the screen. After that, you have a choice of a wide range of mouse operations.

The following are the mouse functions available for INT 33H, of which relatively few are commonly used:

00H	Initialize the mouse
01H	Display the mouse pointer
02H	Conceal the mouse pointer
03H	Get button status and pointer location
04H	Set pointer location
05H	Get button-press information
06H	Get button-release information
07H	Set horizontal limits for pointer
08H	Set vertical limits for pointer
09H	Set graphics pointer type
0AH	Set text pointer type
0BH	Read mouse-motion counters
0CH	Install interrupt handler for mouse events
0DH	Turn on light pen emulation
0EH	Turn off light pen emulation
0FH	Set mickey-to-pixel ratio
10H	Set pointer exclusion area
13H	Set double-speed threshold
14H	Swap mouse-event interrupt
15H	Get buffer size for mouse driver state
16H	Save mouse driver state
17H	Restore mouse driver state
18H	Install alternative handler for mouse events
19H	Get address of alternative handler
1AH	Set mouse sensitivity
1BH	Get mouse sensitivity

1CH	Set mouse interrupt rate
1DH	Select display page for pointer
1EH	Get display page for pointer
1FH	Disable mouse driver
20H	Enable mouse driver
21H	Reset mouse driver
22H	Set language for mouse driver messages
23H	Get language number
24H	Get mouse information

BASIC MOUSE OPERATIONS

The following sections describe the basic INT 33H operations required for programs that use a mouse.

Function 00H: Initialize the Mouse. This is the first command that a program issues for handling a mouse and needs to be executed only once. Load AX with function 00H with no other input parameters, and issue INT 33H. The operation returns these values:

- AX = 0000H if no mouse support is available or FFFFH if support is available
- BX = the number of mouse buttons if support is available.

If mouse support is available, the operation initializes the mouse driver as follows:

- Sets the mouse pointer to the center of the screen
- Conceals the mouse pointer if it is visible
- Sets the mouse pointer's display page to zero
- Sets the mouse pointer according to the screen mode: rectangle and inverse color for text or arrow shape for graphics
- Sets the mickey-to-pixel ratio, where horizontal ratio = 8 to 8 and vertical ratio = 16 to 8
- Sets the horizontal and vertical limits for the pointer to their minimum and maximum values
- Sets the double-speed threshold to 64 mickeys per second, which you can change.

Function 01H: Display the Mouse Pointer. This operation, used after function 00H, causes the mouse pointer to be displayed on the screen. The operation requires no input parameters and returns no values.

The mouse driver maintains a *pointer flag* that determines whether or not to display the pointer. It displays the pointer if the flag is 0 and conceals it for any other value. Initially, the value is −1; function 01H increments the flag to 0, thus causing the pointer to be displayed. (See also function 02H.)

Function 02H: Conceal the Mouse Pointer. The standard practice is to issue this function at the end of a program's execution to cause the pointer to be concealed. The operation requires no input parameters and returns no values.

The pointer flag is displayed when it contains a 0 and is concealed for any other value. This function decrements the flag from 0 to −1 to cause it to be concealed.

Function 03H: Get Button Status and Pointer Location. This function requires no input parameters and returns this information about the mouse:

- BX = Status of buttons, according to bit location, as follows:
  - Bit 0 Left button (0 = up, 1 = pressed down)
  - Bit 1 Right button (0 = up, 1 = pressed down)
  - Bit 2 Center button (0 = up, 1 = pressed down)
  - Bits 3–15 Reserved for internal use
- CX = Horizontal (x) coordinate
- DX = Vertical (y) coordinate

The horizontal and vertical coordinates are expressed in terms of *pixels*, even in text mode (eight per byte for video mode 03). The values are always within the minimum and maximum limits for the pointer.

Function 04H: Set Pointer Location. This operation sets the horizontal and vertical coordinates for the mouse pointer on the screen (the values for the location are in terms of pixels—eight per byte for video mode 03):

```
MOV   AX,04H            ;Request set mouse pointer
MOV   CX,horizontal     ;Horizontal location
MOV   DX,vertical       ;Vertical location
INT   33H               ;Call mouse driver
```

The operation sets the pointer at the new location, adjusted as necessary if outside the minimum and maximum limits.

PROGAM: DISPLAYING THE MOUSE LOCATION

The program in Figure 15-1 illustrates the basic mouse operations covered to this point. It displays the horizontal and vertical coordinates of the pointer as a user moves, but does not press, the mouse. The main procedures are:

- A10MAIN initializes the program, calls B10INITZ, C10POINTR, D10CONVRT, and E10DISPLY. When the user presses the left button, the program uses function 02H to hide the pointer and ends processing.
- B10INITZ issues INT 33H function 00H to initialize the mouse (or to indicate that no mouse driver is present) and issues function 01H to cause the mouse pointer to display.

```
TITLE     A15MOUSE (EXE)  Handling the mouse
          .MODEL SMALL
          .STACK 64
          .DATA
LEN_DATA  EQU     14                  ;Display length
XCOORD    DW      0                   ;Binary X coordinate
YCOORD    DW      0                   ;Binary Y coordinate
ASCVAL    DW      ?                   ;ASCII field

DISPDATA  LABEL   BYTE                ;Screen display fields:
XMSG      DB      'X = '              ;X message
XASCII    DW      ?                   ;X ASCII value
          DB      '  '                ;
YMSG      DB      'Y = '              ;Y message
YASCII    DW      ?                   ;Y ASCII value
.386  ; ------------------------------------------------
          .CODE
A10MAIN   PROC    FAR
          MOV     AX,@data            ;Initialize
          MOV     DS,AX               ;  DS and ES
          MOV     ES,AX               ;  addressability
          CALL    Q10CLEAR            ;Clear screen
          CALL    B10INITZ            ;Initialize mouse
          CMP     AX,00               ;Mouse installed?
          JE      A90                 ;  no, exit
A20:      CALL    C10POINTR           ;Get mouse pointer
          CMP     BX,01               ;Button pressed?
          JE      A80                 ;  yes, exit
          MOV     AX,XCOORD           ;Convert
          CALL    D10CONVRT           ;  X to ASCII
          MOV     AX,ASCVAL           ;
          MOV     XASCII,AX           ;
          MOV     AX,YCOORD           ;Convert
          CALL    D10CONVRT           ;  Y to ASCII
          MOV     AX,ASCVAL           ;
          MOV     YASCII,AX           ;Display
          CALL    E10DISPLY           ;  X and Y values
          JMP     A20                 ;Repeat
A80:      MOV     AX,02H              ;Request hide pointer
          INT     33H
A90:      CALL    Q10CLEAR            ;Clear screen
          MOV     AX,4C00H            ;End processing
          INT     21H
A10MAIN   ENDP
;                 Initialize mouse pointer:
;                 -------------------------
B10INITZ  PROC    NEAR                ;Uses AX
          MOV     AX,00H              ;Request initialize
          INT     33H                 ;  mouse
          CMP     AX,00               ;Mouse installed?
          JE      B90                 ;  no, exit
          MOV     AX,01H              ;Show pointer
          INT     33H
B90:      RET                         ;Return to caller
B10INITZ  ENDP
```

Figure 15-1 Using the Mouse

C10POINTR issues function 03H to check and to exit if the user has pressed the left button. If not pressed, the program converts the horizontal and vertical coordinates from pixel values to binary numbers (by shifting the values three bits to the right, effectively dividing by 8). If the location is the same as when it was previously checked, the routine repeats issuing function 03H; if the location has changed, control returns to the calling procedure.

```
;                       Get mouse pointer location:
;                       ---------------------------
C10POINTR PROC    NEAR                  ;Uses AX, BX, CX, DX
C20:      MOV     AX,03H                ;Get pointer location
          INT     33H
          CMP     BX,00000001B          ;Left button pressed?
          JE      C90                   ;  yes, means exit
          SHR     CX,03                 ;Divide pixel
          SHR     DX,03                 ;  coordinates by 8
          CMP     CX,XCOORD             ;Has pointer location
          JNE     C30                   ;  changed?
          CMP     DX,YCOORD             ;
          JE      C20                   ;  no, repeat operation
C30:      MOV     XCOORD,CX             ;  yes, save new locations
          MOV     YCOORD,DX             ;
C90:      RET                           ;Return to caller
C10POINTR ENDP
;                       Convert binary X or Y location to ASCII:
;                       ----------------------------------------
                              ;AX set on entry = binary X or Y
D10CONVRT PROC    NEAR                  ;Uses CX, SI
          MOV     ASCVAL,2020H          ;Clear ASCII field
          MOV     CX,10                 ;Set divide factor
          LEA     SI,ASCVAL+1           ;Load ASCVAL address
          CMP     AX,CX                 ;Compare location to 10
          JB      D20                   ;  lower, bypass
          DIV     CL                    ;  higher, divide by 10
          OR      AH,30H                ;Insert ASCII 3s
          MOV     [SI],AH               ;Store in rightmost byte
          DEC     SI                    ;Decr address of ASCVAL
D20:      OR      AL,30H                ;Insert ASCII 3s
          MOV     [SI],AL               ;Store in leftmost byte
          RET                           ;Return to caller
D10CONVRT ENDP
;                       Display X, Y locations:
;                       -----------------------
E10DISPLY PROC    NEAR                  ;Uses AX, BX, BP, CX, DX
          MOV     AX,1300H              ;Request display
          MOV     BX,0031H              ;Page:attribute
          LEA     BP,DISPDATA           ;Address of string
          MOV     CX,LEN_DATA           ;No. of characters
          MOV     DX,0020H              ;Screen row:column
          INT     10H
          RET                           ;Return to caller
E10DISPLY ENDP
;                       Clear screen, set attribute:
;                       ----------------------------
Q10CLEAR  PROC    NEAR                  ;Uses AX, BH, CX, DX
          MOV     AX,0600H              ;Request clear screen
          MOV     BH,30H                ;Colors
          MOV     CX,00                 ;Full
          MOV     DX,184FH              ;  screen
          INT     10H
          RET                           ;Return to caller
Q10CLEAR  ENDP
          END     A10MAIN
```

Figure 15-1 *Continued*

- D10CONVRT converts the binary values for horizontal and vertical screen locations to displayable ASCII characters. Note that with eight pixels per byte, the horizontal value returned at screen column 79 (the rightmost location) is $79 \times 8 = 632$. The procedure divides this horizontal value by 8 to get, in this case, 79, the maximum value. Consequently, the conversion ensures that values returned are within 0 through 79.

- E10DISPLY displays the horizontal and vertical coordinates at the center of the screen as X = *col* and Y = *row*.

One way to improve this program would be to issue function 0CH to set an interrupt handler. In this way, the required instructions are automatically invoked whenever the mouse is active.

MORE ADVANCED MOUSE OPERATIONS

This section covers the remaining mouse operations and the following section provides another program example.

Function 05H: Get Button-Press Information. This function returns information about button presses. Set BX with the button number, where 0 = left, 1 = right, and 2 = center:

```
MOV   AX,05H          ;Request press information
MOV   BX,button-no    ;Button number
INT   33H             ;Call mouse driver
```

The operation returns the up/down status of all buttons and the press count and location of the requested button:

- AX = Status of buttons, according to bit location, as follows:
  - Bit 0 Left button (0 = up, 1 = pressed down)
  - Bit 1 Right button (0 = up, 1 = pressed down)
  - Bit 2 Center button (0 = up, 1 = pressed down)
  - Bits 3–15 Reserved for internal use
- BX = Button-press counter
- CX = Horizontal (x) coordinate (pixel value) of last button press
- DX = Vertical (y) coordinate (pixel value) of last button press

The operation resets the button-press counter to 0.

Function 06H: Get Button-Release Information. This function returns information about button releases. Set BX with the button number (0 = left, 1 = right, and 2 = center):

```
MOV   AX,06H          ;Request release information
MOV   BX,button-no    ;Button number
INT   33H             ;Call mouse driver
```

The operation returns the up/down status of all buttons and the release count and location of the requested button, as follows:

- AX = Status of buttons, according to bit location, as follows:
  - Bit 0 Left button (0 = up, 1 = pressed down)
  - Bit 1 Right button (0 = up, 1 = pressed down)

Bit 2 Center button (0 = up, 1 = pressed down)

Bits 3–15 Reserved for internal use

- BX = Button release counter
- CX = Horizontal (x) coordinate (pixel value) of last button release
- DX = Vertical (y) coordinate (pixel value) of last button release

The operation resets the button release counter to 0.

Function 07H: Set Horizontal Limits for Pointer. This operation sets the minimum and maximum horizontal limits (pixel values) for the pointer:

```
MOV   AX,07H        ;Request set horizontal limit
MOV   CX,minimum    ;Minimum limit
MOV   DX,maximum    ;Maximum limit
INT   33H           ;Call mouse driver
```

If the minimum value is greater than the maximum, the operation arbitrarily exchanges the values. If the pointer is outside the defined area, the operation moves it inside the area. See also functions 08H and 10H.

Function 08H: Set Vertical Limits for Pointer. This operation sets minimum and maximum vertical limits (pixel values) for the pointer:

```
MOV   AX,08H        ;Request set vertical limit
MOV   CX,minimum    ;Minimum limit
MOV   DX,maximum    ;Maximum limit
INT   33H           ;Call mouse driver
```

If the minimum value is greater than the maximum, the operation arbitrarily exchanges the values. If the pointer is outside the defined area, the operation moves it inside the area. See also functions 07H and 10H.

Function 0BH: Read Mouse-Motion Counters. This operation returns the horizontal and vertical mickey count (within the range −32,768 to +32,767) since the last request to the function. Returned values are:

- CX = Horizontal count (a positive value means travel to the right, negative means to the left)
- DX = Vertical count (a positive value means travel downwards, negative means upwards)

Function 0CH: Install Interrupt Handler for Mouse Events. A program may need to respond automatically when a mouse-related activity (or event) has occurred. The purpose of function 0CH is to provide an *event handler* whereby the mouse software interrupts your program and calls the event handler, which performs its required function and returns to your program's point of execution on completion of the task.

Load CX with an event mask to indicate the actions for which the handler is to respond and ES:DX with the segment:offset address of the interrupt handler routine:

```
MOV   AX,0CH          ;Request interrupt handler
LEA   CX,mask         ;Address of event mask
LEA   DX,handler      ;Address of handler (ES:DX)
INT   33H             ;Call mouse driver
```

Define the event mask with bits set as required:

0 = mouse pointer moved	4 = right button released
1 = left button pressed	5 = center button pressed
2 = left button released	6 = center button released
3 = right button pressed	7–15 = reserved, define as 0

Define the interrupt handler as a FAR procedure. The mouse driver uses a far call to enter the interrupt handler with these registers set:

- AX = The event mask as defined, except that bits are set only if the condition occurred
- BX = Button state (if set, bit 0 means left button down, bit 1 means right button down, and bit 2 means center button down)
- CX = Horizontal (x) coordinate
- DX = Vertical (y) coordinate
- SI = Last vertical mickey count
- DI = Last horizontal mickey count
- DS = Data segment for the mouse driver

On the program's entry into the interrupt handler, push all registers and initialize DS to the address of your data segment. Within the handler, use only *BIOS*, not DOS, interrupts. On exit, pop all registers.

Function 10H: Set Pointer Exclusion Area. This operation defines a screen area in which the pointer is not displayed:

```
MOV   AX,10H          ;Request set exclusion area
MOV   CX,upleft-x     ;Upper left x-coordinate
MOV   DX,upleft-y     ;Upper left y-coordinate
MOV   SI,lowright-x   ;Lower right x-coordinate
MOV   DI,lowright-y   ;Lower right y-coordinate
INT   33H             ;Call mouse driver
```

To replace the exclusion area, call the function again with different parameters, or reissue function 00H or 01H.

Function 13H: Set Double-Speed Threshold. This operation sets the threshold speed at which the pointer motion on the screen is doubled. Load DX with the new value (the default is 64 mickeys per second). (See also function 1AH.)

Function 1AH: Set Mouse Sensitivity. Sensitivity concerns the number of mickeys that the mouse needs to move before the pointer is moved. This function sets the

horizontal and vertical mouse motion in terms of the number of mickeys per eight pixels, as well as the threshold speed at which the pointer motion on the screen is doubled (see also functions 0FH, 13H, and 1BH):

```
MOV   AX,1AH          ;Request set mouse sensitivity
MOV   BX,horizontal   ;Horizontal mickeys (default = 8)
MOV   CX,vertical     ;Vertical mickeys (default = 16)
MOV   DX,threshold    ;Threshold speed (default = 64)
INT   33H             ;Call mouse driver
```

Function 1BH: Get Mouse Sensitivity. This operation returns the horizontal and vertical mouse motion in terms of number of mickeys per eight pixels as well as the threshold speed at which the pointer motion on the screen is doubled. (See function 1AH for the returned registers and values.)

Function 1DH: Select Display Page for Pointer. The page for video display is set with INT 10H function 05H. For mouse operations, set the page number in BX and issue this function.

Function 1EH: Get Display Page for Pointer. This operation returns the current video display page in BX.

Function 24H: Get Mouse Information. This operation returns information about the version and type of mouse that is installed:

BH = Major version number
BL = Minor version number
CH = Mouse type (1 = bus mouse, 2 = serial mouse)

PROGRAM: USING THE MOUSE WITH A MENU

Earlier, the program in Figure 10-2 used the cursor keys for selecting an item from a menu. The program in Figure 15-2 is similar, but now allows the user to move the mouse pointer up and down the menu and to select an entry by pressing the left button. Also, there is now an entry at the bottom of the menu for "Exit Program." The major procedures are the following:

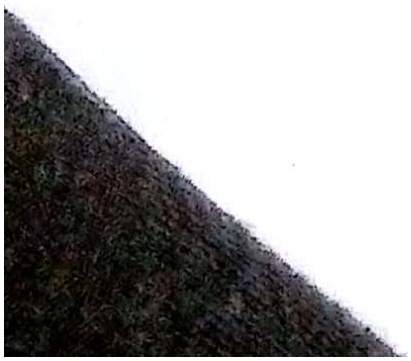
- A10MAIN calls B10INITZ to initialize the mouse, calls C10MENU to display the menu, calls E10DISPLY to highlight the current menu line, calls D10POINTR to respond to mouse actions, and ends processing when the user requests "Exit Program."
- B10INITZ initializes the mouse, displays the pointer, and sets horizontal and vertical limits to the pointer area.
- C10MENU displays the full set of menu selections.
- D10POINTR checks for the left button pressed; if so, calls E10DISPLY to set the old menu line to normal video and the selected line to reverse video.
- E10DISPLY displays menu lines according to given attributes.

```
TITLE     A15SELMU (EXE) Select item from menu
          .MODEL SMALL
          .STACK 64
          .DATA
TOPROW    EQU    08                 ;Top row of menu
BOTROW    EQU    16                 ;Bottom row of menu
LEFTCOL   EQU    26                 ;Left column of menu
LEN_LINE  EQU    19                 ;Length of menu line
ATTRIB    DB     ?                  ;Screen attribute
COL       DB     00                 ;Screen column
ROW       DB     00                 ;Screen row
SHADOW    DB     19 DUP(0DBH)       ;Shadow characters
MENU      DB     0C9H, 17 DUP(0CDH), 0BBH
          DB     0BAH, ' Add records      ', 0BAH
          DB     0BAH, ' Delete records   ', 0BAH
          DB     0BAH, ' Enter orders     ', 0BAH
          DB     0BAH, ' Print report     ', 0BAH
          DB     0BAH, ' Update accounts  ', 0BAH
          DB     0BAH, ' View records     ', 0BAH
          DB     0BAH, ' Exit program     ', 0BAH
          DB     0C8H, 17 DUP(0CDH), 0BCH
PROMPT    DB     'To select an item, press left '
          DB     'button of mouse pointer.'
.386 ; ------------------------------------------------
          .CODE
A10MAIN   PROC   FAR
          MOV    AX,@data           ;Initialize segment
          MOV    DS,AX              ;  registers
          MOV    ES,AX
          CALL   Q10CLEAR           ;Clear screen
          CALL   B10INITZ           ;Initialize mouse
          CMP    AX,00              ;Mouse installed?
          JE     A90                ;  no, exit
          CALL   C10MENU            ;Display menu
A20:
          MOV    ROW,TOPROW+1       ;Set row to top item
          MOV    ATTRIB,16H         ;Set reverse video
          CALL   E10DISPLY          ;Highlight current menu line
          CALL   D10POINTR          ;Call mouse routine
          CMP    DX,BOTROW-1        ;Exit requested?
          JNE    A20                ;  no, continue
          MOV    AX,02H             ;Hide mouse pointer
          INT    33H
          MOV    AX,0600H           ;Clear
          CALL   Q10CLEAR           ;  screen
A90:      MOV    AX,4C00H           ;End of processing
          INT    21H
A10MAIN   ENDP
;                  Initialize mouse pointer, set
;                  horizontal and vertical limits:
;                  ------------------------------
B10INITZ  PROC   NEAR               ;Uses AX, CX, DX
          MOV    AX,00H             ;Request initialize
          INT    33H                ;  mouse
          CMP    AX,00              ;Mouse installed?
          JE     B90                ;  no, exit
```

Figure 15-2 Selecting from a Menu

```
                MOV     AX,01H          ;Show pointer
                INT     33H
                MOV     AX,04H          ;Set pointer
                MOV     CX,256
                MOV     DX,108
                INT     33H
                MOV     AX,07H          ;Horizontal limits
                MOV     CX,LEFTCOL+1    ;  left column
                MOV     DX,LEFTCOL+17   ;  right column
                SHL     CX,03           ;Multiply by 8 for
                SHL     DX,03           ;  pixel value
                INT     33H
                MOV     AX,08H          ;Vertical limits
                MOV     CX,TOPROW+1     ;  top row
                MOV     DX,BOTROW-1     ;  bottom row
                SHL     CX,03           ;Divide by 8
                SHL     DX,03
                INT     33H
B90:            RET
B10INITZ        ENDP
;                       Display shadow box and full menu:
;                       ---------------------------------
C10MENU         PROC    NEAR            ;Uses AX, BP, BX, CX, DX
                MOV     AX,1301H        ;Request display
                MOV     BX,0060H        ;Black on brown
                LEA     BP,SHADOW       ;Address of shadow
                MOV     CX,LEN_LINE     ;Length of line
                MOV     DH,TOPROW+1     ;Screen row
                MOV     DL,LEFTCOL+1    ;  and column
C20:            INT     10H
                INC     DH              ;Next row
                CMP     DH,BOTROW+2     ;All rows displayed?
                JNE     C20             ;  no, repeat
                MOV     ATTRIB,71H      ;Blue on white
                MOV     AX,1300H        ;Request display
                MOV     BH,00           ;Page 0
                MOV     BL,ATTRIB       ;Attribute
                LEA     BP,MENU         ;Address of menu
                MOV     CX,LEN_LINE     ;Length of line
                MOV     DH,TOPROW       ;Screen row,
                MOV     DL,LEFTCOL      ;  column
C30:            INT     10H
                ADD     BP,LEN_LINE     ;Next menu line
                INC     DH              ;Next row
                CMP     DH,BOTROW+1     ;All rows displayed?
                JNE     C30             ;  no, repeat
                MOV     AX,1300H        ;Request display
                MOV     BH,00           ;Page 0
                MOV     BL,ATTRIB       ;Attribute
                LEA     BP,PROMPT       ;Prompt line
                MOV     CX,45           ;Length of prompt
                MOV     DH,BOTROW+4     ;Screen row,
                MOV     DL,15           ;  column
                INT     10H
                RET
C10MENU         ENDP
```

Figure 15-2 *Continued*



```
;                       If left button pressed, set old menu line to
;                       normal video, new line to reverse video:
;                       ----------------------------------------
D10POINTR PROC    NEAR                  ;Uses AX, BX, DX
D20:      MOV     AX,03H                ;Get button status
          INT     33H
          CMP     BX,00000001B          ;Left button pressed?
          JNE     D20                   ;  no, repeat
          SHR     DX,03                 ;Divide vertical by 8
          CMP     DX,BOTROW-1           ;Request for exit?
          JE      D90                   ;  yes, exit
          PUSH    DX                    ;  no, save row
          MOV     ATTRIB,71H            ;Blue on white
          CALL    E10DISPLY             ;Set old line to normal video
          POP     DX                    ;Get row
          MOV     ROW,DL
          MOV     ATTRIB,17H            ,White on blue
          CALL    E10DISPLY             ;Set new line to reverse video
          JMP     D20                   ;Repeat
D90:      RET
D10POINTR ENDP
;                       Set menu line to normal or highlight:
;                       -------------------------------------
E10DISPLY PROC    NEAR                  ;Uses AX, BX, BP, CX, DX
          MOVZX   AX,ROW                ;Row tells which menu line
          SUB     AX,TOPROW
          IMUL    AX,LEN_LINE           ;Multiply by length of line
          LEA     SI,MENU+1             ;  for selected menu line
          ADD     SI,AX
          MOV     AX,1300H              ;Request display
          MOV     BH,00                 ;Page
          MOV     BL,ATTRIB             ;New attribute
          MOV     BP,SI                 ;Menu line
          MOV     CX,LEN_LINE-2         ;Length of string
          MOV     DH,ROW                ;Row
          MOV     DL,LEFTCOL+1          ;Column
          INT     10H
          RET
E10DISPLY ENDP
;                        Clear screen, set attribute:
;                        ----------------------------
Q10CLEAR  PROC    NEAR                  ;Uses AX, BH, CX, DX
          MOV     AX,0600H              ;Request scroll
          MOV     BH,61H                ;Blue on brown
          MOV     CX,0000               ;Full screen
          MOV     DX,184FH
          INT     10H
          RET
Q10CLEAR  ENDP
          END     A10MAIN
```

Figure 15-2 *Continued*

KEY POINTS

- In text mode, the mouse pointer is a flashing block in reverse video; in graphics mode, the pointer is an arrowhead.
- Mouse operations use INT 33H with a function code loaded in the full AX.
- The first mouse operation to execute should be INT 33H function 00H, which initializes the mouse driver.

- INT 33H Function 01H is required to display the mouse pointer, 03H to get the button status, 04H to get the pointer location, 05H to get button-press information, and 06H to get button-release information.
- The horizontal and vertical coordinates for the mouse location are in terms of pixels.

REVIEW QUESTIONS AND EXERCISES

15-1. Explain these terms: (a) mickey, (b) mickey count, (c) mouse pointer.

15-2. Provide the INT 33H function for each of the following mouse operations:
(a) Conceal the mouse pointer
(b) Get button-press information
(c) Set pointer location
(d) Install interrupt handler for mouse events
(e) Get button-release information
(f) Read mouse-motion counters

15-3. Explain the purpose of the mouse pointer flag.

15-4. Code the instructions for the following requirements:
(a) Initialize the mouse
(b) Display the mouse pointer
(c) Get mouse information
(d) Set the mouse pointer on row 22 at the center column
(e) Get mouse sensitivity
(f) Get button status and pointer location
(g) Conceal the mouse pointer.

15-5. Combine the requirements in Question 15-4 into a full program. You can run the program under DEBUG, although at times DEBUG may scroll the pointer off the screen.

15-6. Code instructions for setting the pointer exclusion area to (a) upper left: x = 40, y = 40, (b) lower right: x = 160, y = 80.