User's Guide

Microsoft® Mouse

(Serial Version)
for the IBM® Personal Computer

Installation and Operation Manual

Microsoft Corporation

If you have questions about this documentation or enclosed software, complete the Problem Report at the back of this manual and return it to Microsoft.

Document No. 8809-300-02p
Part No. 039-099-027

Microsoft Mouse (Serial Version) Addendum

Version 3.0 of the mouse driver software (the version included in this package) and later versions will run correctly if you are using an IBM Color Graphics Adapter, IBM Monochrome Adapter, Hercules Graphics Card, IBM Enhanced Graphics Adapter, or IBM All Points Addressable Graphics Adapter.

Demonstration Programs

In this section in Chapter 3 of the Mouse Installation and Operation Manual, *page 25, change the Note to the following:*

Note

Piano does not run on an IBM 3270 Personal Computer with an All Points Addressable Graphics Adapter.

The Game of Life runs only on an IBM Personal Computer with a Color/Graphics Monitor Adapter or an Enhanced Graphics Adapter.

Making Mouse System Calls

In this section in Chapter 4 of the Mouse Installation and Operation Manual, *page 49, change the first sentence in the first paragraph to read:*

This section describes how to make mouse system calls from the BASIC interpreter, from assembly language programs, and from programs in high-level languages such as FORTRAN, Pascal, and BASIC.

Making Calls from High-Level Languages

In this section in Chapter 4 of the Mouse Installation and Operation Manual, *page 52, change the first sentence in the first paragraph to read:*

You can make calls from compiled FORTRAN, Pascal, and BASIC language programs.

Preface

In this section of the Microsoft Notepad Manual, *page iii (unnumbered), add the following text as the last paragraph under the* "Important" *heading:*

Notepad does not run on an IBM 3270 Personal Computer with an All Points Addressable Graphics Adapter.

What You Need

In this section in Chapter 1 of the PC Paintbrush User's Guide, *page 4, add the following note after the last paragraph:*

Note

PC Paintbrush does not run if you are using an IBM Monochrome Adapter or an IBM 3270 Personal Computer with an All Points Addressable Graphics Adapter.

Introduction

Add the following to the "Introduction" section on p. 9 of the PC Paintbrush User's Guide:

Do not erase files from the original Microsoft Mouse Program Disk. Before beginning the PC Paintbrush installation, make certain you are using a copy of the Program Disk. Follow these steps to erase the PBRUSH.EXE or PBRUSH1.EXE file:

1. Insert the PC Paintbrush disk into disk drive A.
2. If you are using an IBM Enhanced Graphics Adapter, type the following at the A> prompt:

   erase PBRUSH.EXE
3. If you are not using an IBM Enhanced Graphics Adapter, type the following at the A> prompt:

   erase PBRUSH1.EXE

Display Adapters

Add the following to p. 70 in the "Display Adapters" section of the PC Paintbrush User's Guide:

If you are using the IBM Enhanced Graphics Adapter, please note the following:

- To use PC Paintbrush with the IBM Enhanced Graphics Adapter, your disk (or directory) must include all the PC Paintbrush files plus the EGA.DEV file.
- The See Clip option on the Page Menu is not available if you are using an IBM Enhanced Graphics Adapter with PC Paintbrush.

Local Undo

Add the following to p. 19 of the PC Paintbrush User's Guide:

- You can use the Backspace key to perform a Local Undo when using any tool except the Hand and Text tools. Place the cursor on the area to be undone and press the Backspace key.
- When using the Show Screen, Adjust Palette, or ZOOM-in/zoom-in options, you can cancel an action by pressing both mouse buttons at the same time.

README.DOC

Please be sure to read the README.DOC file on your System Disk before installing and using the Microsoft Mouse. The README.DOC file contains information that became available after this manual was printed.

Preface

This manual explains how to use the Microsoft Mouse and accompanying software. In particular, it contains information about:

- How to install the mouse
- How to use the mouse in programs
- How to make system calls to the mouse software from your application programs

---

Important

This manual assumes that you have read Section 2, "Setup," and Section 3, "Operation," in the IBM Personal Computer *Guide to Operations* manual. It also assumes that you are familiar with Chapter 3, "DOS Commands," in the IBM Personal Computer *Disk Operating System* manual.

Contents

Chapter 1
Introduction

The Microsoft® Mouse is a hand-held pointing device that controls the motion of the cursor on the CRT screen of the IBM® Personal Computer. Designed to be used with a variety of screen-oriented programs, the Microsoft Mouse frees you from dependence on directional and text keys to move the cursor and to make command selections.

By sliding the Microsoft Mouse across a flat surface, you can guide the cursor to the words and symbols on the screen that represent the commands of a program. By pressing the buttons on the mouse, you can select the commands that you want to be performed. You use the mouse instead of the keyboard to direct the action of the program.

The Microsoft Mouse includes all the software and hardware necessary to use the mouse with your computer. The Microsoft Mouse is simple to use and easy to learn. This manual shows how to install the mouse and how to use it.

Hardware Features

The Microsoft Mouse hardware consists of a mouse pointing unit, attached cable, and a 25-pin "female" connector. The mouse has two buttons for flexible command selection and a removable ball for easy cleaning.

The mouse can be used with any serial communications interface board capable of RS-232 standard asynchronous data communications. The only other requirement is either a 25-pin male connector or an adapter connector for the mouse cable. All power for the mouse is supplied by your computer; no additional power is required.

Software Features

The Microsoft Mouse software consists of the driver program for implementing the mouse and demonstration programs that will help you learn how to use the mouse.

No special configuration is required to operate the mouse. Simply load the mouse software into memory and you are ready to use the mouse in any program that uses the Microsoft Mouse.

The mouse driver programs make the necessary modifications to the operating system to run the mouse. Two driver programs are provided for the different versions of DOS. The instructions on using the driver programs are given in the "Software Installation" section of Chapter 2.

The three demonstration programs, Doodle, Piano, and Life, are designed to teach you the basic techniques of the Microsoft Mouse. If you read the short tutorial in Chapter 3 and experiment with the programs, you will soon be an expert at using the mouse.

The applications software is a set of programs called Microsoft Mouse Menu. These are programs that provide mouse support and permit you to use "popup menus" and the Microsoft Mouse for application programs that currently have no built-in mouse support. Included are prewritten mouse menus for some of the most popular applications and a program to create your own mouse menus. For more information, see the *Mouse Menu* manual.

Also included in the Microsoft Mouse package are mouse functions to incorporate the Microsoft Mouse into your BASIC, assembly, and high-level language application programs. Mouse functions allow a program to access the mouse and control the cursor. You can call these functions from your application programs by using the system calls described in Chapter 4.

Differences Between Mouse Versions

There are two operational versions of the Microsoft Mouse: Bus version and Serial version. The versions differ in system requirements and the hardware included in the mouse package. The software is the same for either version.

The Serial version mouse is for the IBM PC equipped with a serial communications interface board capable of RS-232 standard asynchronous communication. The board must have a 25-pin male connector or a 25-pin male-to-female adapter. Serial version packages include the mouse pointing unit, a connecting cable, and a 25-pin female connector.

The Bus version mouse includes the mouse pointing unit, connecting cable, a 9-pin female connector, and an interface circuit board. The Bus version is for the IBM PC that either doesn't have a serial communications interface board installed or whose serial communications port is already used for something else.

How to Use This Manual

This manual provides ready information for both first-time mouse users and experienced programmers.

- Chapter 1 introduces the Microsoft Mouse hardware and software.
- Chapter 2 lists detailed instructions for installing the Microsoft Mouse on your computer.
- Chapter 3 gives the operating instructions for the Microsoft Mouse and teaches the basic techniques of the mouse with a simple tutorial.
- Chapter 4 discusses the interface between the Microsoft Mouse and your computer, and describes the mouse functions that can be called by your application programs with the mouse system calls.
- Appendix A explains how to clean the Microsoft Mouse.
- Appendix B is the BASIC program listing of the demonstration program Piano.
- Appendix C lists eight sample cursors that you may incorporate into your own application programs.

Chapter 2
Installation

This chapter gives instructions for installing the Microsoft Mouse and mouse software. Before installing the mouse, read the following preliminary procedures.

Preliminary Procedures

First, make sure that your system meets all the criteria set forth in the following "System Requirements" section. The IBM Personal Computer should be set up and operational as specified in the IBM *Guide to Operations* manual.

System Requirements

To use your Microsoft Mouse successfully, you will need the following:

1. An IBM Personal Computer
2. An IBM Personal Computer Disk Operating System (DOS), any version
3. A disk drive
4. A screen monitor
5. An interface board capable of RS-232 standard serial communications with either a 25-pin female connector or an adapter cable for connecting to the mouse 25-pin male connector
6. The Microsoft Mouse
7. The Microsoft Mouse software disk
8. A high-resolution screen graphics interface board. (The IBM Color/Graphics Adapter board is recommended.)

Unpacking

Upon receipt of your Microsoft Mouse package, check carefully for shipping damage. If any item is damaged or missing, report it to your computer dealer for further action.

Your Microsoft Mouse package should include the following:

The Microsoft Mouse

The Microsoft Mouse software disk with the following files:

123.DEF	MENU.COM	MPMS.MNU
123.MNU	MOUSE.COM	PIANO.BAS
AUTOEXEC.BAT	MOUSE.LIB	PIANO.EXE
DOODLE.EXE	MOUSE.SYS	VC.DEF
HOKUSAI	MPIBM.DEF	VC.MNU
LIFE.EXE	MPIBM.MNU	WS.DEF
MAKEMENU.EXE	MPMS.DEF	WS.MNU

Note

Because the Microsoft Mouse is packaged with other Microsoft products, there may be additional files listed on the disk, or the above files may be contained on another disk.

The *Microsoft Mouse Installation and Operation Manual*

A Microsoft Customer Service Plan

If reshipment of the mouse or the software disk is necessary, contact Microsoft Corporation as instructed in the Customer Service Plan before returning it.

Removing the Foam Ring

To protect the mouse during shipping, a soft foam ring is placed between the ball-retaining cover on the bottom of the mouse and the ball inside. Before installing the mouse, you will need to remove the ring. Follow these steps:

1. Hold the mouse upside down and locate the foam ring. (It should be clearly visible in the center of the mouse.)

2. Grasp the ring with your fingers.

3. Carefully pull the ring out from under the retaining cover and free the ring from the mouse.

If you have trouble, follow the procedure in Appendix A, "Cleaning Instructions," to remove the ball-retaining cover. Then remove the ring and replace the ball-retaining cover as instructed.

Mouse Installation Procedure

Use the following procedure to connect the Microsoft Mouse to a serial communications port.

Note

Once you connect the mouse to the selected serial port and run the mouse software, the port can only be used for mouse input. That is, the mouse software modifies DOS so that the port is no longer available for application programs or languages such as BASIC. To change ports, you must move the mouse cable and change the software as described in the "Software Installation Procedures" section of this chapter.

---

1. Set the power switch to OFF unless the system unit is already off.

2. Locate a serial communications port on the back of the system unit. See the IBM *Guide to Operations* manual to determine how the ports are numbered.

3. Connect the mouse cable to the serial port connector.

Note

Some computer or serial interface boards may have a different type of connector. If this is the case, ask your computer dealer or the computer manufacturer how to obtain an adapter cable.

The mouse should now be installed. The next step is installing the software. The following section will tell you how.

Software Installation Procedures

This section gives instructions for loading the Microsoft Mouse software into the IBM PC memory. The software can be loaded manually after each system reset or automatically from a system disk. The following subsections describe the ways to load the mouse software.

Using a system disk for automatic installation is the most convenient way to load the mouse software. It requires a few preliminary steps to copy the appropriate mouse files onto each system disk you plan to use. When you complete these steps, you will not have to load the mouse software after each system reset.

Manual Installation

Use the following procedure if you wish to install the mouse software manually each time you load the Disk Operating System (DOS). The DOS is loaded after you turn on the computer and after a system reset (pressing the DEL key while holding down the CTRL and ALT keys).

1. Insert your system disk into drive A and turn on the power to the computer, or if power is already on, perform a system reset. When the computer has finished its self-test, it will ask you to change the time and date. After providing the needed information or just pressing the RETURN key, you will see the DOS banner and the A> prompt.

2. If you have a multiple drive system, insert the mouse disk into drive B. Type,

   B:

   and press the RETURN key.

   If you have a single drive system, remove the system disk from drive A and insert the mouse disk.

3. If the mouse is connected to the first serial port, type,

   MOUSE /1

   and press the RETURN key. If the mouse is connected to the second serial port, type,

   MOUSE /2

   and press the RETURN key.

   The computer responds by running the MOUSE.COM program to load the mouse software into memory. If successful, the screen will display the message:

   Mouse driver installed

   You can now use the Microsoft Mouse. Remember that when loading the mouse software manually, steps 2 and 3 should be repeated after each system reset.

Unsuccessful Installation

If the software installation was unsuccessful, the screen will display one of two messages. If the screen displays,

```
Bad command or filename
```

it indicates that the MOUSE.COM program is either missing or cannot be run. To prevent this problem, make sure MOUSE.COM is on the disk you are using. Also make sure you are installing the mouse software from the active disk drive. Then, repeat steps 2 and 3. If you receive the message again, the problem is possibly a damaged disk.

If the mouse is improperly installed or there are other hardware problems related to the serial port, the screen will display:

```
Mouse: Serial card not found
```

If you receive this message, check to see that the serial communications interface board is correctly installed and working. See the board's technical manual for instructions on testing the interface board. If the board is not functioning properly, contact your computer dealer for help.

Automatic Installation

Automatic installation permits you to bypass the steps required to load the mouse software manually. Automatic installation consists of a system disk containing the DOS and a special file that directs the computer to load the mouse software each time the operating system is loaded into memory.

Creating a system disk for automatic installation of the mouse software consists of the following three steps:

1. Copy the appropriate mouse files onto each system disk you plan to use with the mouse.

2. Add the appropriate command line to either of the following: AUTOEXEC.BAT file (DOS versions 1.x) or CONFIG.SYS file (DOS versions 2.x).

3. Test the system disk by performing a system reset.

The following procedures show how to add the mouse software to system disks. Use the procedure that is appropriate for your DOS version. If necessary, refer to the operating system manual for help with the utilities used in this procedure.

DOS Versions 1.x Installation Procedure

1. Insert the desired system disk into drive A. You will use this disk to automatically install the mouse software when DOS is loaded into memory.

2. Turn on the power to the computer, or if the power is already on, perform a system reset. When the computer has completed its self-test, enter the time and date (or just press the RETURN key for each prompt) until you see the A> prompt.

3. Insert the mouse disk into drive B. If you have a single-drive system, remove the system disk and insert the mouse disk into drive A. Follow the instructions on the screen for changing disks. If you are unsure as to how to copy files with a single-drive system, see the *IBM DOS* manual for instructions.

4. Copy the MOUSE.COM file onto the system disk.

Note

If you don't have an AUTOEXEC.BAT file on your system disk, copy AUTOEXEC.BAT from the mouse disk to your system disk.

---

5. Remove the mouse disk from drive B and load a word processing program or text editor into memory.

6. If the mouse is connected to the first serial port, use a word processing program or text editor to insert the following command line into the AUTOEXEC.BAT file:

```
MOUSE /1
```

If the mouse is connected to the second serial port, substitute MOUSE /2 for MOUSE /1.

7. Repeat Steps 1 through 6 for each system disk that you plan to use with the mouse.

8. Test the AUTOEXEC.BAT file by performing a system reset. If the mouse software is loaded successfully, you will see the message:

   ```
   Mouse driver installed
   ```

   If unsuccessful, check the AUTOEXEC.BAT file for errors and try the procedure again. If unsuccessful again, see "Unsuccessful Installation" in the preceding section.

You now have a system disk that will automatically install the mouse software each time you load the operating system.

DOS Versions 2.x Installation Procedure

1. Insert the desired DOS 2.x system disk into drive A. You will use this disk to automatically install the mouse software when DOS is loaded into memory.

2. Turn on the power to the computer, or if the power is already on, perform a system reset. When the computer has completed its self-test, enter the time and date (or just press the RETURN key for each prompt) until you see the A> prompt.

3. Insert the mouse disk into drive B. If you have a single drive system, remove the system disk and insert the mouse disk into drive A. Follow the instructions on the screen for changing disks. If you are unsure as to how to copy files with a single drive system, see the *IBM DOS* manual for instructions.

4. Copy the MOUSE.SYS file onto the system disk.

5. Remove the mouse software disk from drive B and load a word processing program or text editor into memory.

6. If the mouse is connected to the first serial port, use the word processing program or text editor to insert the following command line into the CONFIG.SYS file:

   ```
   DEVICE = MOUSE.SYS /1
   ```

   If the mouse is connected to the second serial port, substitute MOUSE.SYS /2 for MOUSE.SYS /1.

7. Repeat the steps 1 through 6 for each system disk that you want to use with the mouse.

8. Test the CONFIG.SYS file by performing a system reset. If the mouse software is loaded successfully, you will see the message:

   ```
   Mouse driver installed
   ```

   If unsuccessful, check the CONFIG.SYS file for errors, and repeat the procedure. If unsuccessful again, see "Unsuccessful Installation" in the preceding section.

You now have a system disk that will automatically install the mouse software each time you load the operating system.

Backup Copies

It is always good practice to make a backup copy of a disk. If you do not have a copy of the Microsoft Mouse disk, use the following procedure to make a backup copy of the disk:

1. Use the FORMAT program to format a new disk.

2. Use the DISKCOPY program to copy the files on the mouse disk to the new disk.

3. Label the new disk "Microsoft Mouse."

To use the FORMAT and DISKCOPY programs, follow the instructions in the IBM *Disk Operating System* manual.

Store the original mouse disk in a cool, dry place away from direct sunlight.

Chapter 3
How to Use the Mouse

In this chapter, you will learn the basic techniques of using the mouse, and practice these techniques using two demonstration programs, Piano and The Game of Life.

Basic Techniques

The Microsoft Mouse takes just a few minutes to learn. If you have never used a mouse, the following sections will help you get acquainted. If you have used a mouse before, these sections will indicate differences between the Microsoft Mouse and other mouse devices.

Mouse Anatomy

Before using the Microsoft Mouse, it is a good idea to examine its working parts. Using Figure 3.1, locate the left and right buttons. The buttons permit you to make selections when presented with choices by a program. When you press and release a button, the mouse passes this information to the program.

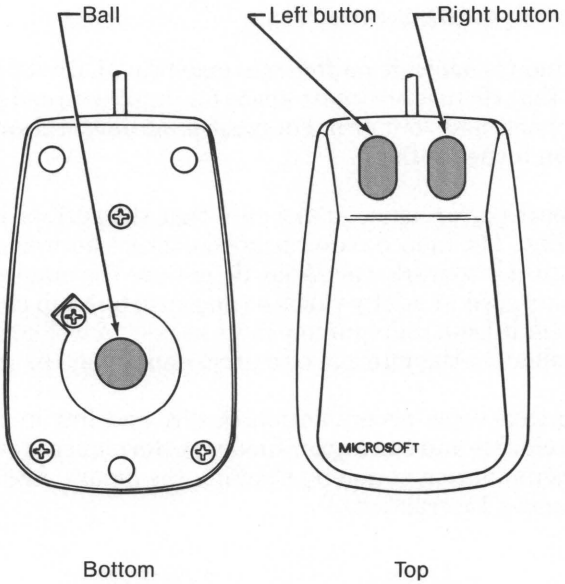


Figure 3.1 Mouse Ball and Buttons

A button's function depends on how the current program has defined it. Just like the programmable keys on a keyboard, the function of a button can change from program to program.

Again using Figure 3.1, turn the mouse upside down and locate the ball. The ball controls the movement of the cursor on the screen. When you hold the mouse top side up and slide it across a hard, flat surface, the ball rolls within its socket. The mouse translates this rolling into directional data and passes it to the mouse software to move the cursor on the screen.

The Microsoft Mouse is enclosed in a tough and durable plastic case. Although it can survive a fall from a desk top, take care to safeguard against falls and other accidents that might shorten its operating life.

Mouse Surface Requirements

Use the mouse on any hard, flat surface, such as a desk. We recommend placing the mouse right beside the keyboard since most programs that use the Microsoft Mouse combine both the mouse and the keyboard for input. But any configuration which is comfortable for you and any surface which is hard and flat is fine.

The mouse depends on free movement in all directions, so make sure that there is adequate space for uninterrupted movement of the mouse and your arm. For most programs, a clear space of ten by ten inches is fine.

For best performance, make sure that the surface is free of dirt and lint. The mouse requires good contact between the ball and the surface to work well. Also, do not use the mouse on sticky or wet surfaces. A sticky surface can cause buildup on the ball and prevent it from rolling freely in its socket. A wet surface can lead to a short in the internal circuitry, damaging the mouse.

Note that some accumulation of dirt and lint in the mouse is unavoidable and can impair mouse performance. You can remove an accumulation of dirt by cleaning the mouse. See Appendix A, "Cleaning Instructions."

Holding the Mouse

The Microsoft Mouse is designed to be easily held in either the right or the left hand. To hold the mouse:

1. Place your palm over the Microsoft logo on the mouse body.

2. Hold your index and middle fingers over the buttons.

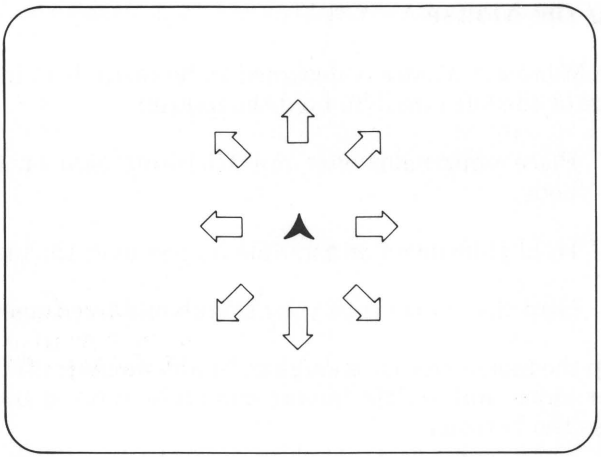
3. Grip the mouse with your thumb and free fingers.

Grip the mouse tightly enough to be able to lift it off the surface. Your index and middle fingers should be relaxed but ready to press the buttons.

Moving the Mouse/Controlling the Cursor

The Microsoft Mouse controls the motion of the cursor when a program that uses the mouse is running. The shape of the cursor and its starting point on the screen depend on the program, but the way the cursor moves on the screen is the same for all programs.

To move the cursor, simply move the mouse in the direction that you wish the cursor to move. All motions of the cursor are relative to the front of the mouse (where the buttons are located). Pushing the mouse to the front moves the cursor to the top of the screen. Pulling in the opposite direction moves the cursor to the bottom of the screen. Unlike the direction keys on the keyboard, the mouse permits you to move the cursor in any direction, even diagonally. Figure 3.2 illustrates ways that you can move the mouse and the corresponding cursor motions.



Movement of cursor on screen corresponds to movement of mouse on table or desk

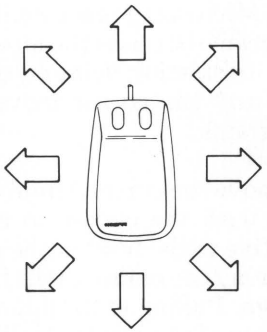


Figure 3.2 Mouse and Cursor Motions

The cursor moves only when the mouse moves across the surface. The location of the mouse on the surface is immaterial. This means that you can lift the mouse off the surface and return it to its starting point without returning the cursor to its starting point. This feature is useful when you are moving the cursor all the way across the screen. The move can be an accumulation of short strokes instead of one long stroke.

Demonstration Programs

The three demonstration programs, Doodle, Piano, and The Game of Life, are designed to let you practice and master the basic techniques of the Microsoft Mouse. Piano and Life are explained in the following sections.

Doodle is a graphics demonstration program for sketching different shapes on your computer screen. A special file (HOKUSAI) has been included on the mouse software disk to show what can be created with Doodle. For instructions on how to use Doodle, see the Doodle manual which is part of the Microsoft Mouse package.

To load a demonstration program, type the filename at the keyboard and press the RETURN key. The filenames of these programs are:

```
DOODLE
PIANO
LIFE
```

Remember, if the Microsoft Mouse software disk is not in the default drive, you must precede the filename with a drive name.

Note

The Game of Life runs only on an IBM Personal Computer that has a Color/Graphics Monitor Adapter.

When the program is loaded and execution has begun, you will see the game screen and the mouse cursor. We recommend trying Piano first to learn how to move and control the cursor. Then try The Game of Life to learn how to use the Microsoft Mouse to select commands in a program.

Program 1: Piano

Piano lets you play music at a video keyboard. The game screen consists of a keyboard (21 white keys and 15 black keys) and a "quit" box in the lower right corner.

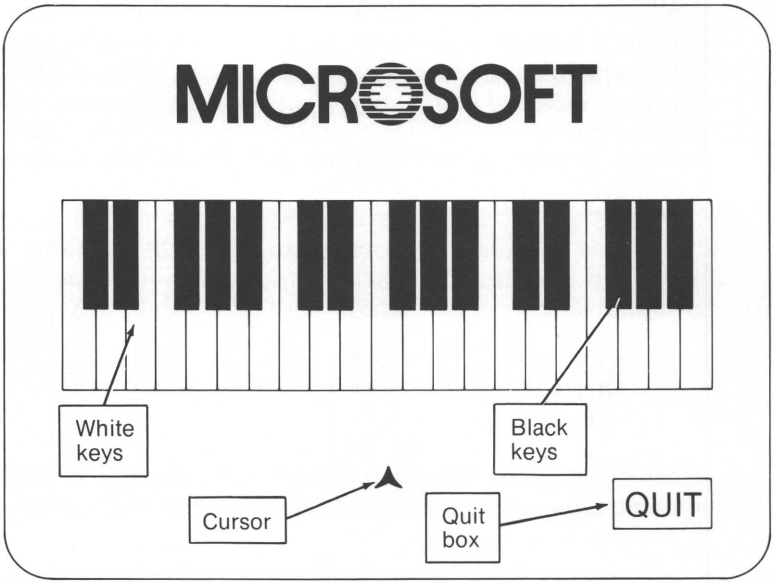


Figure 3.3 Piano Screen

The cursor is in the middle of the screen just below the keyboard. Practice moving the cursor by moving the mouse from side to side. Notice how just a small motion of the mouse moves the cursor quickly and accurately. With just a little practice you can pinpoint even the smallest objects on the screen.

Don't be afraid to move the cursor to the edge of the screen. The screen edge forms a boundary beyond which the cursor will not pass.

The notes of the white keys range from low C on the left to high B on the right. The black keys are the sharps and flats between these notes. To play a note, use the mouse to move the cursor over the key that you want and press the left button. Note that the tip of the cursor must be within the boundaries of the key.

For example, to play middle C move the cursor to the eighth white key from the left and press the left button.

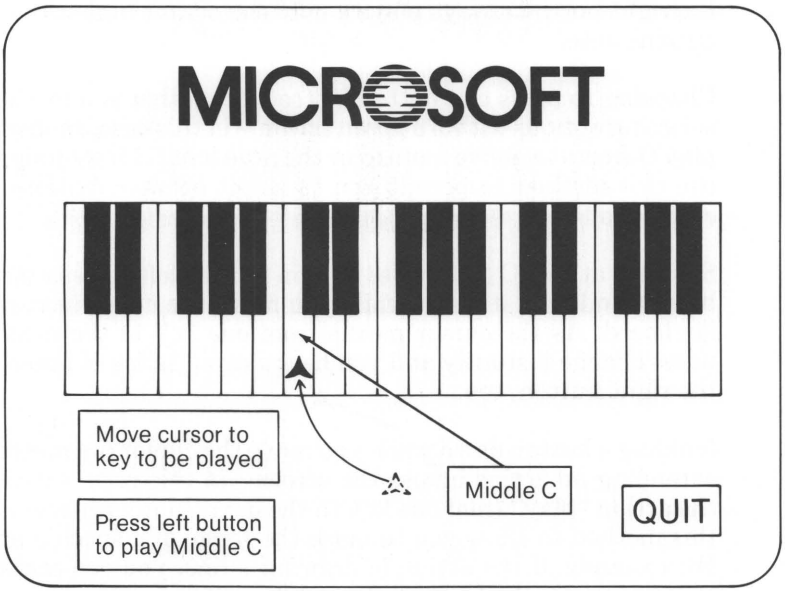


Figure 3.4 Playing the Piano

The computer responds by playing a middle C. The note plays as long as you hold the button down and stops as soon as you release the button.

Play another note by moving the cursor to another key and pressing the left button. Each note you choose plays as long as you hold the button down. If you move the cursor off the keyboard and press the button, no note will play.

Moving the cursor to a key and pressing the button is a method of selection. Many programs use this method to allow you to choose a program action from a menu of commands. You simply move the cursor to the word, command, or symbol that represents the action and press the button. This method is faster and easier than typing command letters or names at the keyboard.

Now return the cursor to middle C. To play an octave higher, you can either move eight white keys to the right, or leave the cursor where it is and press the mouse's right button. In Piano, the right button always plays a note one octave higher than the current note.

Choosing to press one button instead of another is a method of selecting options within a given action—in this case, choosing to play the octave above instead of the note itself. Many programs use this method to permit you to select between options in a command.

Starting at low C (the white key on the far left), press the left button and hold it down while you move the cursor across the keyboard. As the cursor moves from one key to the next, the notes change instantly and you hear a rapid series of notes. Try the right button, too.

Holding a button down while you move the cursor is a method of extending an action across the screen—in this case, extending the action "play" from one key to the next. Many programs use this method to allow you to mark the range of a specific action. For example, if the action is drawing a line, you can mark the starting point, the line's path, and its ending point.

When you have finished playing Piano, move the cursor to the quit box and press the left button. The computer responds by leaving Piano and displaying the system prompt.

Program 2: The Game of Life

The Game of Life lets you simulate the growth and death of cultures of living cells. The game consists of a 20- by 39-line grid, a command line, and a message line.

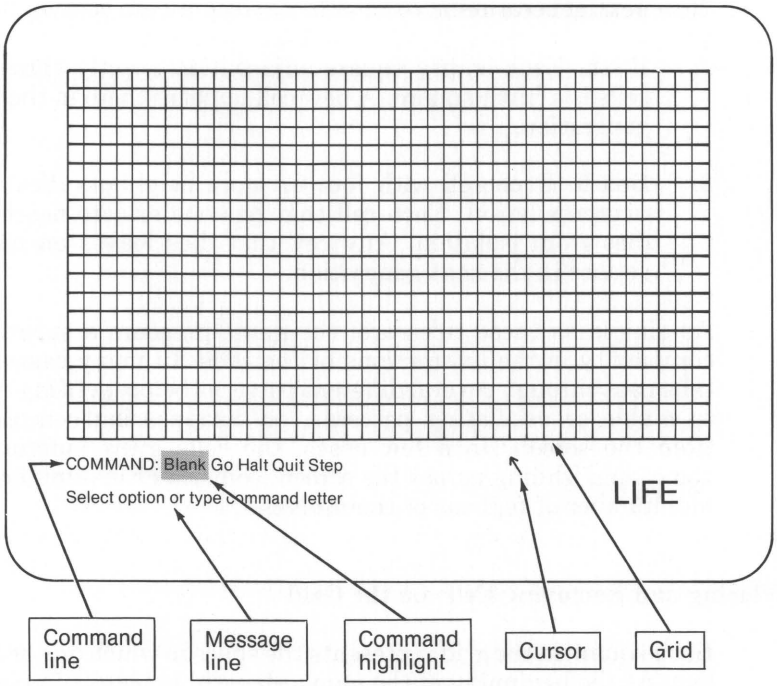


Figure 3.5 Life Screen

The object of the game is to place cells in the squares of the grid and watch how they interact generation after generation.

The Game of Life has three simple rules:

1. Survival. Every cell with two or three adjacent neighbors (vertically, horizontally, or diagonally) survives for the next generation.

2. Birth. Each empty square adjacent to exactly three live cells is a birth square. A new cell appears there in the next generation.

3. Death. Each cell with four or more neighbors dies from overpopulation. Each cell that has one or zero neighbors dies from isolation. Any cell that dies leaves an empty square in the next generation.

As simple as these rules are, the game presents a surprising complexity in the interactions of the cells. In many cases, the cells pass through several different patterns before settling down to stable or oscillating patterns, or disappearing completely from the screen. In a few cases, the cells grow indefinitely, sometimes gliding across the screen, sometimes making an infinite number of replicas of themselves.

Placing and Removing Cells on the Grid

Each square in the grid represents the space in which one cell can live. At the beginning of the game, the grid is blank. To place a cell in a square, touch the tip of the cursor to the square and press the left button. (In The Game of Life the right button has no effect.)

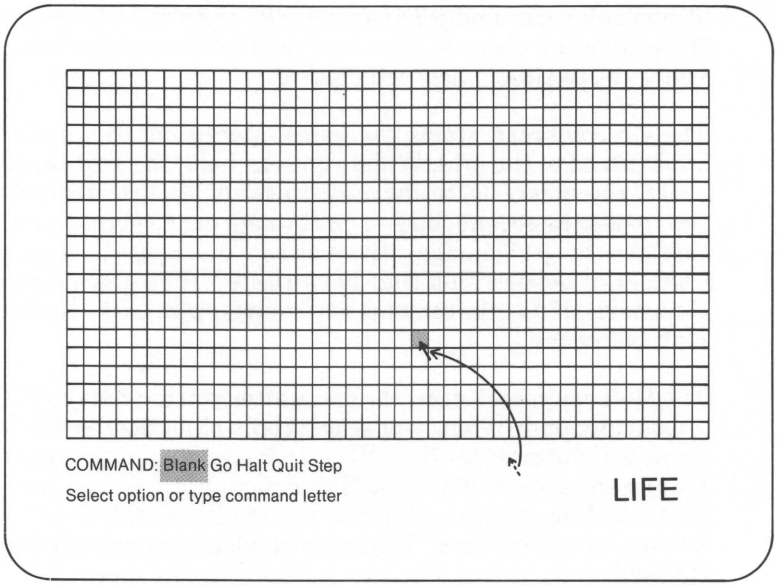


Figure 3.6 Placing a Cell

If you want to place many cells in the grid at once, move the cursor across the grid while holding down the left button. Be sure to start with the cursor in an empty cell. Cells will continue to be placed in the grid until you release the button.

Cells can only live and grow within the boundaries of the grid. If you try to place a cell outside of the grid, a warning tone sounds.

If you want to remove a cell from a square, touch the tip of the cursor to the cell and press the left button. To remove many cells at once, move the cursor over cells in the grid while holding down the left button. Be sure to start with the cursor in a square that has a cell. Cells will continue to be removed until you release the button.

The Commands

The Game of Life has five commands that control the action of the game. Each command has a command word that appears in the command menu below the grid.

The Blank command removes all cells from the grid and resets the generation count to zero. Use the Blank command whenever you want to start a new cell culture.

The Go command starts the game, giving life to the cells you have placed in the grid. When you select Go, the grid changes to reflect the survivals, births, and deaths of the first generation of cells, then the second, and so on through each successive generation. As the grid changes, the message line keeps a count of the number of generations that have passed. The generations continue (even if no cells remain) until you stop Life by selecting the Halt command.

The Halt command stops the game after you have started it with a Go command. When you select Halt, all action on the screen stops and the message line tells you the number of the generation now on the screen. After the Halt command, you may use Go to continue Life, Step to continue one more generation of Life, or Blank to clear the grid. You can also place or remove cells from the grid.

The Step command starts the game like the Go command but sustains it for one generation only. When you select Step, the grid changes to reflect the survivals, births, and deaths of one generation and then all action stops. The message line displays the number of the generation now on the screen.

The Quit command ends the game. When you select Quit, the message "Enter Y to confirm" appears in the message line. Type *Y* to end The Game of Life; press the CANCEL (ESC) key to continue.

Selecting and Entering Commands

There are three ways to select and enter commands:

1. Move the cursor to the word in the command menu and press and release the left button on the mouse,

2. Type the command letter (first letter of the command word), or

3. Use the space bar or TAB key to move the menu highlight to the command word and then press the RETURN key.

The most convenient method is to move the cursor to the word in the command menu and press and release the left button.

To practice this method, select the Go command. After placing cells in the grid, move the cursor so that its tip touches the word "Go" in the command menu, then press the left button. Be careful to touch the tip to the word; if it does not touch you will get a warning tone.

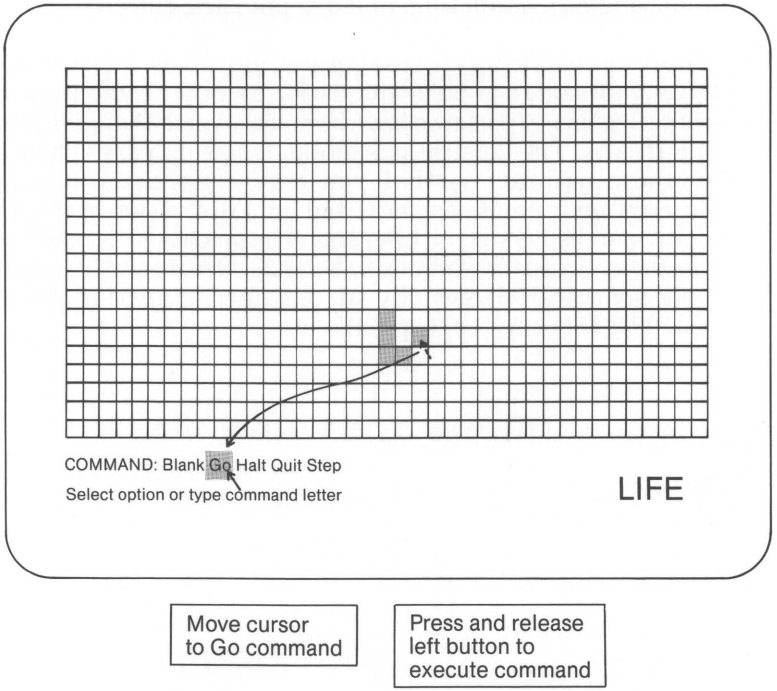


Figure 3.7 Selecting the Go Command

To stop the action, use the cursor to select the Halt command.

Try selecting the Go command using another method. The effect is the same, but you might find that using the keyboard is cumbersome compared to using the mouse.

Playing the Game

The Game of Life is a game of discovery. Since in most cases the outcome is unpredictable, a large part of the charm of the game is discovering just what a certain pattern will do.

When playing the Game of Life for the first time, we recommend that you start small—with three cells—and combine them in as many patterns as possible before moving on to more cells. In this way, you will discover the basic patterns that develop into stable (unchanging), oscillating, and disappearing cultures.

The Game of Life was first suggested in 1970 by John Conway. It quickly became a popular simulation game for computers. Since that time, many patterns have been discovered that give interesting results. The illustrations on the following pages show some of these patterns so that you may try them yourself. For descriptions of more patterns and for a discussion of the mathematics of The Game of Life, see "Mathematical Games" by M. Gardner, *Scientific American*, Vol. 223, October 1970, p.120 and "Mathematical Games" by M. Gardner, *Scientific American*, Vol. 224, February 1971, p. 112.

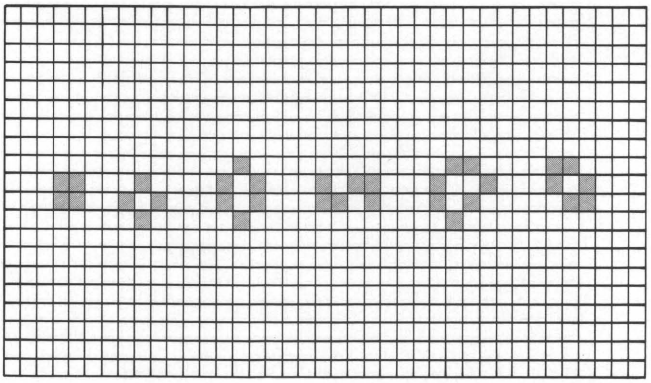


Figure 3.8 Stable Patterns

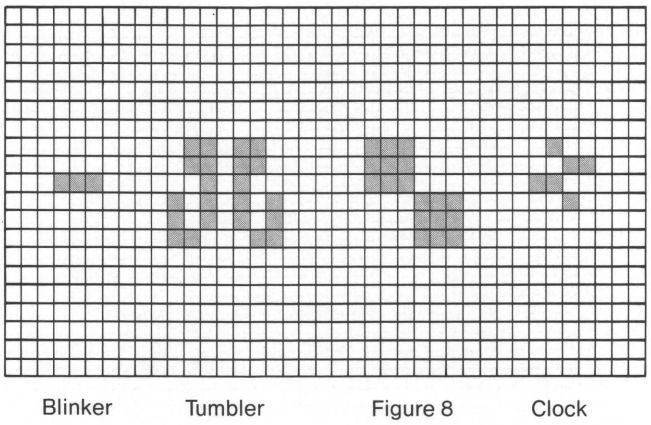
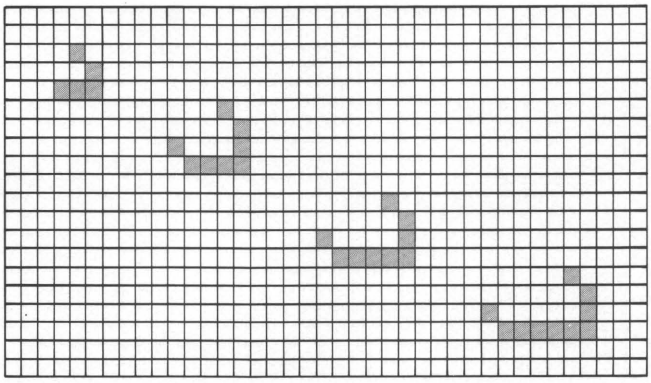
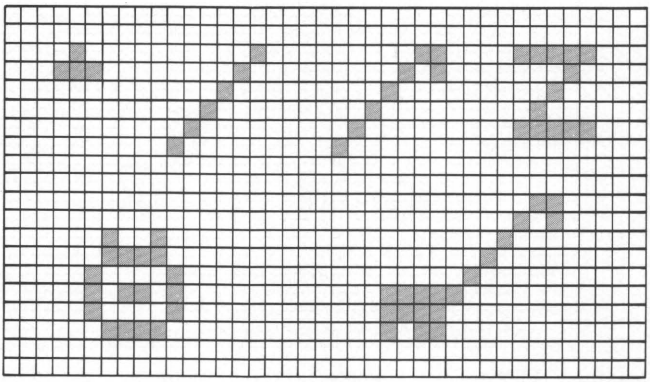


Figure 3.9 Oscillating Patterns



Glider | Lightweight spaceship | Mediumweight spaceship | Heavyweight spaceship

Figure 3.10 Glider and Space Ships



TOP ROW: Traffic lights | Fuse | Capped fuse | Z

BOTTOM ROW: Cheshire cat (watch for grin generation 6) | Harvester

Figure 3.11 Patterns with Interesting Lives

Chapter 4
Mouse Interface and System Calls

This chapter provides the information you need to incorporate the Microsoft Mouse into your application programs. It describes:

The interface between your computer's screen and Microsoft Mouse software.

The steps required to make mouse system calls from BASIC, assembly, and high-level language programs.

The input, output, and operation of the mouse function system calls.

A sample BASIC program illustrating the use of the mouse functions.

Mouse system calls are easy to incorporate in new and existing application programs. To make full use of the system calls, we recommend that you carefully read each section and pay close attention to the sample program described in Appendix B of this manual.

Mouse Interface

This section describes the interface between the mouse software and the IBM Personal Computer. In particular, it describes how the mouse software uses the resources of the personal computer to create a cursor on the screen and control its movement. This section defines:

1. The screen modes
2. The virtual screen
3. The graphics and text cursors
4. The buttons
5. The mouse unit of distance: the mickey
6. The internal cursor flag

Since many of the mouse functions make use of the interface, it is important that you understand it before using the functions. Read the following sections carefully before trying to use the functions in your application programs.

The Screen Modes

The mouse software works with all four screen modes of the IBM Personal Computer. The screen modes define the number of pixels (points of light) on the screen and the types of objects you can see on the screen. The screen modes are:

1. High-resolution graphics mode
2. Four-color graphics mode
3. 80-column text mode
4. 40-column text mode

In high-resolution graphics mode, there are 128,000 pixels on the screen, each capable of being either white or black. You can control the color of each individual pixel; that is, you can change the color of a pixel without affecting its neighbors. This allows you to create any image on the screen that you want.

In four-color (medium resolution) graphics mode, there are only 64,000 pixels on the screen. Each pixel can be individually controlled, and each is capable of being one of four colors. This allows you to create images with color.

In 80-column text mode, the screen is divided into 25 lines of text characters, with 80 characters in each line. Although there are 128,000 pixels on the screen, the pixels cannot be individually controlled, as in the graphics modes. The only way to change the screen is by changing the characters on the screen. Note that each 8- by 8-pixel group forms a single character.

In 40-column text mode, the screen is divided into 25 lines of text characters, with 40 characters in each line. There are 64,000 pixels on the screen, and the pixels cannot be individually controlled. To change the screen you must change the characters on the screen. In this mode, each 16- by 8-pixel group forms a single character.

The screen modes available on your IBM Personal Computer depend on your adapter. If your computer has the Color/Graphics Monitor Adapter, all four screen modes are available. If you have the Monochrome Display and Printer Adapter, only the 40- and 80-column text modes are available.

The Virtual Screen

The mouse software operates on the IBM Personal Computer screen as if it were a "virtual screen" of 128,000 individual points arranged in a matrix of 640 horizontal by 200 vertical points. Whenever it refers to the location of an object on the screen (for example, a pixel or a character), no matter what the mode, it gives that object's location as a pair of coordinates on the virtual screen. For example, a pixel in the upper left corner of the screen has the coordinates (0,0) and a character at the center of the screen has the coordinates (320,100). Each pair of coordinates defines a point on the virtual screen. The horizontal coordinate is given first.

In high-resolution graphics mode each point in the virtual screen has a one-to-one correspondence with each pixel on the screen. In this mode, the full range of coordinates, from (0,0) to (639,199), is permitted.

In four-color graphics mode, there are half the number of pixels on the screen as in high-resolution graphics mode. To compensate, the mouse software uses even-numbered horizontal coordinates only. This means every other point in the virtual screen corresponds to a pixel.

In 80-column text mode, only characters are permitted on the screen. There are still 128,000 pixels on the screen, each corresponding one-to-one with the points in the virtual screen, but because the individual pixels in a character cannot be accessed, the mouse software uses just one pair of coordinates to refer to the location of a character. The coordinates used are the coordinates of the pixel in the upper left corner of the character. For example, the character in the upper left corner of the screen has the coordinates (0,0), the next character horizontally has the coordinates (8,0), and so on. Since each character in this mode is an 8- by 8-pixel group, both the horizontal and vertical coordinates are multiples of eight.

In 40-column text mode, the mouse software again uses the coordinates of just one pixel in a character to refer to the location of the character. But there are half the number of pixels as in 80-column text mode. To compensate, the mouse software uses horizontal coordinates that are multiples of sixteen. For example, the character in the upper left corner of the screen still has the coordinates (0,0), but the character next to it has the coordinates (16,0).

There are many mouse functions that take coordinates as input or return coordinates as output. Whenever you make a reference to a pixel or character in a function, make sure that the horizontal and vertical coordinates are correct values for the given screen mode. If you supply an incorrect value, the function will round the value down before continuing. The mouse functions always return correct values for the given screen mode.

The Graphics and Text Cursors

The mouse has three different cursors: a graphics cursor, and two text cursors. The graphics cursor is a shape (for example, an arrow) that moves over the images on the screen. The software text cursor is a character attribute (for example, an underscore) that moves from character to character on the screen. The hardware text cursor is a flashing block, half-block, or underscore that also moves from character to character on the screen. Only one cursor can be on the screen at any given time. You can choose which cursor is on the screen, and even switch back and forth between cursors.

Functions 9 and 10 of the mouse system calls permit you to define the characteristics of these cursors. You may define the characteristics yourself, or use the characteristics of the sample cursors listed in Appendix C. The following subsections describe the cursors in detail.

The Graphics Cursor

The graphics cursor is the cursor used when the IBM Personal Computer is in high-resolution or four-color graphics mode. The graphics cursor is a block of pixels. In high-resolution graphics mode, it is 256 pixels in a 16- by 16-pixel square. In four-color graphics mode, it is 128 pixels in an 8- by 16-pixel square. As you move the mouse, the block moves over the screen and interacts with the pixels directly under it. This interaction creates the cursor shape and background.

The interaction between the cursor points and screen pixels is defined by two 16- by 16-bit arrays called the screen mask and the cursor mask. In high-resolution mode, each bit in a mask corresponds to one pixel in the cursor block. In four-color graphics mode, each pair of bits corresponds to a pixel. The screen mask determines whether the cursor pixel is part of the shape or background. The cursor mask determines how the pixel under the cursor contributes to the color of the cursor.

To create the cursor, the mouse software operates on the data in the IBM Personal Computer's screen memory that defines the color of each pixel on the screen. First, the software logically ANDs the screen mask with the 256 bits of data that define the pixels under the cursor. Then, it logically XORs the cursor mask with the result of the AND operation. Table 4.1 shows how these operations affect the individual screen bits.

Table 4.1

Mask Bit Values and Screen Result

if the screen mask bit is:	if the cursor mask bit is:	the resulting screen bit is:
0	0	0
0	1	1
1	0	unchanged
1	1	inverted

In high-resolution mode, each screen bit defines the color of a single pixel, so just one bit in the screen mask and one bit in the cursor mask define the pixel's color when the cursor is over it. For example, if the first bit in the screen mask is 1 and the first bit in the cursor mask is 0, then the upper left corner of the cursor block is transparent.

In four-color graphics mode, each pair of screen bits defines the color of a pixel, so a pair of bits in the screen mask and a pair in the cursor mask define a pixel's color.

You can define the screen mask and the cursor mask values explicitly by defining the masks as arrays in your program and passing them as parameters in a call to Function 9 of the mouse system calls. By assigning the appropriate values to the screen mask and cursor mask, you can define any shape to the cursor that you wish. For an example, see the description of Function 9 later in this chapter or the sample program in Appendix B.

The Graphics Cursor "Hot Spot"

Whenever a mouse function refers to the graphics cursor location, it gives the point on the virtual screen that lies directly under the cursor's "hot spot." The hot spot is the point in the cursor block that the mouse software uses to determine the cursor coordinates.

You can define which point in the cursor block will be the hot spot by passing the horizontal and vertical coordinates of the point to Function 9. In both high-resolution and four-color graphics mode, the coordinates must be within the range –16 to 16, but in four-color graphics mode, the horizontal coordinate must be an even number. In both cases, the coordinates are relative to the upper left corner of the cursor block.

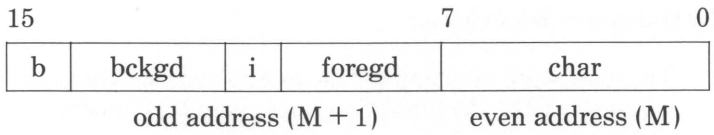
The Software Text Cursor

The software text cursor is the cursor used when the IBM Personal Computer is in 40- or 80-column text mode. The text cursor affects the appearance of the characters on the screen. Unlike the graphics cursor, the text cursor usually does not have a shape of its own; instead, it changes character attributes such as foreground and background color, intensity, and underscoring of the character directly under it. If the cursor does have a shape of its own, it is one of the 256 ASCII characters in the IBM Personal Computer character set.

The effect of the text cursor on the character under it is defined by two 16-bit values called the screen mask and the cursor mask. The screen mask determines which attributes of the character on the screen are to be preserved. The cursor mask determines how these attributes are to be altered to yield the cursor.

To create the cursor, the mouse software operates on the data that defines each character on the screen. The software first logically ANDs the screen mask and the 16 bits of screen data for the character under the cursor. It then logically XORs the cursor mask and the result of the AND operation.

In both 40-column and 80-column text modes, the 16 bits of screen data for each character take the following form:



where:

b	sets blinking or nonblinking character
bckgd	sets the background color
i	sets high intensity or medium intensity
foregd	sets the foreground color
char	is the ASCII value of the character

The range of values for each field depends on the display adapter in your computer. See the IBM *Technical Reference* manual for details.

The screen and cursor masks are divided into the same fields as shown above, so the value of these fields in the screen and cursor masks defines the new attributes of the character when the cursor is over it. For example, to invert the foreground and background colors, the screen mask and cursor mask should have the following values:

	b	bckgd	i	foregd	char	=
screen mask	0	111	0	111	11111111	&H77FF
cursor mask	0	111	0	111	00000000	&H7700

You can define the values of the screen mask and cursor mask by passing their values as parameters in Function 10 of the mouse system calls. For an example, see the description of Function 10 later in this chapter.

Whenever a mouse function refers to the text cursor location, it gives the coordinates of the character under the cursor. The text cursor does not have a hot spot.

The Hardware Text Cursor

The hardware text cursor is another cursor that can be used when the IBM Personal Computer is in text mode.

The hardware text cursor is actually the IBM Personal Computer's own cursor (the one you see after the A> prompt on the screen). The mouse software allows you to adapt this cursor for your own use.

The hardware cursor is 8 pixels wide and 8 to 12 pixels tall. Each horizontal set of pixels forms a line, called a "scan line." There are 8 to 12 scan lines. A scan line can be on or off. If a line is on, it appears as a flashing bar on the screen. If a line is off, it has no effect on the screen. You may turn on any number of adjacent scan lines. Gaps between scan lines are not allowed. This gives the hardware text cursor a characteristic box or underscore shape.

You can define which lines are on and which are off by passing the number of the first and last lines in the cursor to Function 10 of the mouse system calls. The number of lines in the hardware cursor depends on the display adapter in your IBM Personal Computer. If your computer has a Color/Graphics Monitor Adapter, the cursor has 8 lines. If your computer has a Monochrome Display and Printer Adapter, the cursor has 12 lines. The lines are numbered from 0 to 7 or from 0 to 11.

The Buttons

The mouse functions read the state of the buttons on the mouse and keep a count of the number of times the buttons are pressed and released.

A button state is "pressed" if the button is down, and "released" if the button is up. When a function returns the state of the buttons, it returns an integer value in which the first 2 bits are set or cleared. Bit 0 represents the state of the left button, and bit 1 represents the state of the right button. If a bit is set (equal to 1), the button is down. If a bit is clear (equal to 0), the button is up.

The mouse software has internal counters to keep track of the number of presses and releases of a button. The software increments a counter each time the corresponding button is pressed or released. The software sets a counter to zero after a reset (Function 0) or after a counter's contents are read (Functions 5 and 6).

The Mouse Unit of Distance: The Mickey

The motion of the mouse is translated into values that express the direction and duration of the motion. The values are given in a unit of distance called a "mickey," which is approximately 1/100 of an inch.

When you slide the mouse across a desk top, the mouse hardware passes the software a horizontal and a vertical "mickey count," i.e., the number of mickeys the mouse ball has rolled in the horizontal and vertical directions. The software uses the mickey count to move the cursor a certain number of pixels on the screen.

The number of pixels moved does not have to correspond one-to-one with the number of mickeys the ball rolled. The mouse software defines a "sensitivity" for the mouse, which is a ratio of the number of mickeys required to move the cursor 8 pixels on the screen. The sensitivity determines the rate the cursor moves on the screen.

You can define the sensitivity of the mouse by passing a mickey count to Function 15 of the mouse system calls. The count can be any value from 1 to 32767. For example, if you send a count of 8, the sensitivity is 8 mickeys per 8 pixels. That is, the cursor will move 1 pixel for each mickey the ball rolls, or one character for every 8 mickeys the ball rolls.

The Internal Cursor Flag

The mouse software maintains an internal flag that determines when the cursor should be displayed on the screen. When the flag is zero, the cursor is displayed. When the flag is any other value, the cursor is hidden.

The flag is not directly accessible to your program. To change the flag's value, you must use Functions 1 and 2 in the mouse system calls. Function 1 increments the flag; Function 2 decrements it. You may use Function 0 to reset the flag to −1. The flag is also reset to −1 when you change screen modes.

Initially, the flag's value is −1, so a call to Function 1 will cause the cursor to be displayed. You can call either function any number of times, but remember each call to one function requires a subsequent call to the other to restore the flag's previous value. For example, if the cursor is on the screen and you make five calls to Function 2, you will have to make five calls to Function 1 to get the cursor back on the screen.

Making Mouse System Calls

This section describes how to make mouse system calls from the BASIC interpreter, from assembly language programs, and from programs in high-level languages such as COBOL, FORTRAN, Pascal, and BASIC. The statements and/or instructions required to make the calls depend on the language of your application program.

You can also let the mouse software call a subroutine in your program whenever a specific condition occurs. When this capability is enabled, the mouse software interrupts whatever process is going on and passes execution control to the subroutine that you have specified in Function 12 of the mouse system calls. For details, see the description of Function 12.

Making Calls From the BASIC Interpreter

To make a mouse system call from a BASIC program running under the BASIC interpreter, you must:

1. Assign the offset and segment address of the mouse software to a pair of integer variables in your program. The mouse entry offset and segment address are in memory. To get these values, insert the following statements into your program:

```
10 DEF SEG=0
20 MSEG =256*PEEK(51*4+3)+PEEK(51*4+2)
30 MOUSE=256*PEEK(51*4+1)+PEEK(51*4)+2
40 IF MSEG OR (MOUSE-2) THEN 60
50 PRINT "Mouse Driver not found": END
60 DEF SEG=MSEG
```

   Be sure that the statements appear before any calls to mouse functions.

2. Use the CALL statement to make the call. The statement should have the form

```
CALL MOUSE(M1%, M2%, M3%, M4%)
```

where MOUSE is the variable containing the entry offset of the mouse software, and M1%, M2%, M3%, and M4% are the names of the integer variables you have chosen for parameters in this call. All four parameters must appear in the CALL statement even if no value is assigned to one or more of them. These must be integer variables. Constants and noninteger variables are not allowed.

To ensure that the variables are integer variables, use the percent sign (%) as the variable name. You may also use the DEFINT statement at the beginning of your program. For example, the statement

```
10 DEFINT A-Z
```

defines all variables as integer. With this statement at the beginning of the program, the percent sign is optional.

Example

Assuming that the variable MOUSE has the mouse software offset, use the following statements to set the cursor position to 320 (horizontal) and 100 (vertical):

```
100  '
200  ' Set cursor position to (320,100)
300  '
400  M1% = 4        ' function number is 4
500  M3% = 320      ' horizontal coordinate
600  M4% = 100      ' vertical coordinate
700  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Making Calls From Assembly Language Programs

To make mouse system calls from an assembly language program, you must:

1. Load the AX, BX, CX, and DX registers with the parameter values.

2. Execute software interrupt 51 (33H).

The AX, BX, CX, and DX registers correspond to the M1%, M2%, M3%, and M4% parameters defined for the BASIC program. Values returned by the mouse functions will be placed in the registers.

Example

Use the following instructions to set the cursor position to 320 (horizontal) and 100 (vertical):

```
;*
;* Set Cursor to Location (320,100)
;*
MOV   AX,  4        ;function #4
MOV   CX,  320      ;set horizontal to 320
MOV   DX,  100      ;set vertical to 100
INT   51            ;interrupt to mouse
```

This call has the same effect as the call from the BASIC program shown in the previous example.

Note

When making a mouse system call in assembly language, Functions 9 and 12 expect a somewhat different value for the fourth parameter than when calling from a BASIC program. See the description of these functions for details.

Making Calls From High-Level Languages

You can make calls from compiled COBOL, FORTRAN, Pascal, and BASIC language programs. The call is included as an ordinary procedure call in the source program. After the program is compiled, it must then be linked with the MOUSE.LIB file on the Microsoft Mouse software disk.

To make a mouse system call from a high-level language, you must:

1. Declare MOUSE or MOUSES as an external procedure. Both procedures expect to be passed the addresses (not the values) of four integer arguments, so be sure to include an appropriate parameter list in the declarations.

2. Use the normal calling conventions to make the calls. Use the procedure MOUSE if the argument addresses are in the same segment as the mouse software (short addresses), and use MOUSES if the arguments are in another segment (long addresses). If your program is in FORTRAN, use MOUSES.

3. Link the compiled program with the mouse library file MOUSE.LIB provided on the Microsoft Mouse software disk.

Example

In a Pascal program with long argument addresses, use the following statement to declare MOUSES as an external procedure:

```
PROCEDURE MOUSES
                (VARS M1,M2,M3,M4:INTEGER);
                EXTRN;
```

Once the procedure has been declared, use the following statements to set the cursor position to 320 (horizontal) and 100 (vertical):

```
M1  :=  4       (* Function number is 4 *)
M3  :=  320     (* Horizontal position *)
M4  :=  100     (* Vertical position *)
MOUSES(M1, M2, M3, M4)
```

The call has the same effect as the calls from the BASIC and assembly language programs shown in the previous examples.

Sample Program

To help you learn how to use the mouse system calls, Appendix B contains the listing of the Piano demonstration program described in Chapter 3. We recommend that you read the listing and refer to this chapter for details on the operation of each function.

The Piano program listing is also in the file PIANO.BAS on the Microsoft Mouse software disk. To see how this program runs under the BASIC interpreter, start the interpreter, load PIANO.BAS, and run it.

Function Descriptions

This section defines the following mouse functions in detail.

Number	Function
0	Mouse Installed Flag and Reset
1	Show Cursor
2	Hide Cursor
3	Get Mouse Position and Button Status
4	Set Mouse Cursor Position
5	Get Button Press Information
6	Get Button Release Information
7	Set Minimum and Maximum Horizontal Position
8	Set Minimum and Maximum Vertical Position
9	Set Graphics Cursor Block
10	Set Text Cursor
11	Read Mouse Motion Counters
12	Set User-Defined Subroutine Input Mask
13	Light Pen Emulation Mode On
14	Light Pen Emulation Mode Off
15	Set Mickey/Pixel Ratio
16	Conditional Off
19	Set Double Speed Threshold

Each description specifies the parameters required to make the call (input) and the expected return values (output), any special considerations to be taken, and an example illustrating how to use the call. All examples show BASIC program segments.

In the function descriptions, the parameter names M1%, M2%, M3%, and M4% are dummy variable names. When making a call, use the names of the variables that you want to pass.

The dummy variable names include the percent sign (%) to emphasize that only integer variables can be used as parameters. Constants, single precision variables, and double precision variables are not allowed.

If the function description does not specify an input for a parameter, you need not supply a value before making the call. If the function description does not specify an output value for a parameter, the parameter's value before and after the call remains unchanged.

Caution

The mouse software does not check input values, so be sure that the values you assign to the parameters before making a call are correct for the given screen mode. If you assign incorrect values, you will get unpredictable results.

Function 0: Mouse Installed Flag and Reset

Input	Output
M1% = 0	M1% = mouse status M2% = number of buttons (always 2)

Returns the current status of the mouse hardware and software.

The mouse status is 0 (false) if the mouse hardware and software are not installed, and is –1 (true) if the hardware and software are installed.

The function also resets the mouse driver to the following default parameters:

Function	Parameter
cursor position	screen center
internal cursor flag	–1
graphics cursor shape/hot spot	arrow/(–1,–1)
text cursor	inverting box
user-defined call mask	all zeroes
light pen emulation mode	enabled
mickey to pixel ratio (horizontal)	8 to 8
mickey to pixel ratio (vertical)	16 to 8
min/max cursor position (horizontal)	0/639
min/max cursor position (vertical)	0/199

Example

To ensure that mouse hardware/software is installed and to reset to default values:

```
000 '
100 '  Is Mouse present?  If not, error.
200 '
300 M1%=0
400 CALL MOUSE(M1%,M2%,M3%,M4%)
500 IF NOT(M1%) THEN PRINT "Mouse not installed.": END
```

Function 1: Show Cursor

Input	Output
M1% = 1	--

Increments the internal cursor flag and, if the flag is 0, displays the cursor on the screen. The cursor tracks the motion of the mouse, changing position as the mouse changes position.

The current value of the internal cursor flag depends on the number of calls that have been made to Function 1 and Function 2. (See the section in this chapter, "The Internal Cursor Flag.") The default flag value is –1, so, after a reset, a call to Function 1 will display the cursor.

Example

```
100  '
200  '  Show the cursor.
300  '
400  M1% = 1
500  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 2: Hide Cursor

Input	Output
M1% = 2	--

Removes the cursor from the screen and decrements the internal cursor flag. The cursor, even though hidden, still tracks the motion of the mouse, changing position as the mouse changes position.

Use this function before modifying any portion of the screen containing the cursor. This prevents the cursor from possibly affecting the data written to the screen.

Remember that each call to this function decrements the internal cursor flag. Each call to this function will require a subsequent call to Function 1 to restore the flag to its previous value. (See the section in this chapter, "The Internal Cursor Flag.")

Example

```
100  '
200  '  Hide the cursor
300  '
400  M1% = 2
500  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 3: Get Mouse Position and Button Status

Input	Output
M1% = 3	M2% = button status M3% = cursor position (horizontal) M4% = cursor position (vertical)

Returns the state of the left and right buttons and the horizontal and vertical positions of the cursor. The button status is a single integer value. Bits 0 and 1 represent the left and right buttons, respectively. A bit is 1 if a button is down, and 0 if up.

The cursor positions are always within the range of minimum and maximum values of the virtual screen. (See the section in this chapter, "The Virtual Screen.")

Example

```
100  '
200  '  Get current cursor positions, check button status.
300  '
400  M1% = 3
500  CALL MOUSE(M1%, M2%, M3%, M4%)
600  IF M2% AND 1 THEN PRINT "Left button down."
700  IF M2% AND 2 THEN PRINT "Right button down."
```

Function 4: Set Mouse Cursor Position

Input	Output
M1% = 4 M3% = (horizontal) new cursor position M4% = (vertical) new cursor position	--

Sets the cursor to the specified horizontal and vertical screen positions. The new values must be in the horizontal and vertical ranges of the virtual screen. If the screen is not in high resolution mode, the values are rounded to the nearest horizontal or vertical values permitted for the current screen mode. (See the section in this chapter, "The Virtual Screen.")

Example

Assume that HMAX and VMAX contain the maximum horizontal and vertical positions values for the virtual screen. To set the cursor to the center of the screen:

```
100  '
200  '  Put cursor in center of screen
300  '
400  M1% = 4
500  M3% = INT( HMAX/2 )
600  M4% = INT( VMAX/2 )
700  CALL MOUSE(M1%,M2%,M3%,M4%)
```

Function 5: Get Button Press Information

Input	Output
M1% = 5	M1% = button status
M2% = button	M2% = count of button presses
	M3% = cursor (horizontal) at last press
	M4% = cursor (vertical) at last press

Returns current button status, a count of button presses since the last call to this function, and the horizontal and vertical position of the cursor at the last press of the button.

The parameter M2% specifies which button is checked. If set to 0, the left button is checked. If 1, the right button is checked.

The button status is a single integer value. Bits 0 and 1 represent the left and right buttons, respectively. A bit is 1 if a button is down, and 0 if up.

The count of button presses is always in the range 0 to 32767; overflow is not detected. The count is set to 0 after the call.

The horizontal and vertical values are in the ranges defined by the virtual screen. Note that these values represent the cursor position when the button was last pressed and do not represent the cursor's current position.

Example

```
100  '
200  '  Get cursor position at last button press.
300  '
400  M1% = 5
500  M2% = 0          '  left button
600  CALL MOUSE(M1%, M2%, M3%, M4%)
700  IF (M1% AND 1) THEN PRINT "Left button down."
```

Function 6: Get Button Release Information

Input	Output
M1% = 6	M1% = button status
M2% = button	M2% = count of button releases
	M3% = cursor (horizontal) at last release
	M4% = cursor (vertical) at last release

Returns current button status, a count of button releases since the last call to this function, and the horizontal and vertical position of the cursor at the last release of the button.

The parameter M2% specifies which button is checked. If set to 0, the left button is checked. If set to 1, the right button is checked.

The button status is a single integer value. Bits 0 and 1 represent the left and right buttons, respectively. A bit is 1 if a button is down, and 0 if up.

The count of button releases is always in the range 0 to 32767; overflow is not detected. The count is set to zero after the call.

The horizontal and vertical values are in the ranges defined by the virtual screen. Note that these values represent the cursor position when the button was last released and do not represent the cursor's current position.

Example

```
100  '
200  '  Get cursor position at last button release.
300  '
400  M1% = 6
500  M2% = 1          ' right button
600  CALL MOUSE(M1%, M2%, M3%, M4%)
700  IF (M1% AND 2) THEN PRINT "Right button down."
```

Function 7: Set Minimum and Maximum Horizontal Position

Input	Output
M1% = 7 M3% = minimum position M4% = maximum position	--

Sets the minimum and maximum horizontal cursor positions on the screen. Subsequent cursor motion is restricted to the specified area. The minimum and maximum values are defined by the virtual screen. (See the section in this chapter, "The Virtual Screen.")

If the cursor is outside the area when the call is made, it moves to just inside the area. If the minimum value is greater than the maximum, the two values are swapped.

Example

```
100  '
200  '  Limit cursor to horizontal positions below 150
300  '
400  M1% = 7
500  M3% = 0
600  M4% = 150
700  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 8: Set Minimum and Maximum Vertical Position

Input	Output
M1% = 8 M3% = minimum position M4% = maximum position	--

Sets the minimum and maximum vertical cursor positions on the screen. Subsequent cursor motion is restricted to the specified area. The minimum and maximum values are defined by the virtual screen. (See the section in this chapter, "The Virtual Screen.")

If the cursor is outside the area when the call is made, it moves to just inside the area. If the minimum value is greater than the maximum, the two values are swapped.

Example

```
100  '
200  '  Limit cursor to vertical positions between 100
300  '  and 150
400  '
500  M1% = 8
600  M3% = 100
700  M4% = 150
800  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 9: Set Graphics Cursor Block

Input	Output
M1% = 9 M2% = cursor hot spot (horizontal) M3% = cursor hot spot (vertical) M4% = pointer to screen and cursor masks	--

Defines the shape, color, and center of the cursor when in graphics mode.

The function uses the values found in the screen mask and cursor mask to build the cursor shape and color. (See the section in this chapter, "The Graphics Cursor.") To pass the screen mask and cursor mask in BASIC, assign their values to an integer array and use the first element of the array as the fourth parameter in the call (see example). To pass the screen and cursor masks in assembly language, assign their values to two contiguous arrays and pass the address of the first array in register DX. Be sure to load the segment address of the arrays in the ES register before making the call.

The cursor hot spot values must define one pixel within the cursor. (See the section in this chapter, "The Graphics Cursor Hot Spot.") The values must be within the range –16 to 16.

Example

To define a cursor in high-resolution graphics mode, first define the values to the cursor array and then make the call:

```
100  '
200  '  Define the screen mask
300  '
400  CURSOR(0,0)=&HFFFF      '1111111111111111
500  CURSOR(1,0)=&HFFFF      '1111111111111111
600  CURSOR(2,0)=&HFFFF      '1111111111111111
700  CURSOR(3,0)=&HFFFF      '1111111111111111
800  CURSOR(4,0)=&HFFFF      '1111111111111111
900  CURSOR(5,0)=&HFFFF      '1111111111111111
1000 CURSOR(6,0)=&HFFFF      '1111111111111111
1100 CURSOR(7,0)=&HFFFF      '1111111111111111
1200 CURSOR(8,0)=&HFFFF      '1111111111111111
1300 CURSOR(9,0)=&HFFFF      '1111111111111111
1400 CURSOR(10,0)=&HFFFF     '1111111111111111
1500 CURSOR(11,0)=&HFFFF     '1111111111111111
1600 CURSOR(12,0)=&HFFFF     '1111111111111111
1700 CURSOR(13,0)=&HFFFF     '1111111111111111
1800 CURSOR(14,0)=&HFFFF     '1111111111111111
1900 CURSOR(15,0)=&HFFFF     '1111111111111111
2000 '
2100 '  Define cursor mask
2200 '
2300 CURSOR(0,1)=&H8000      '1000000000000000
2400 CURSOR(1,1)=&HE000      '1110000000000000
2500 CURSOR(2,1)=&HF800      '1111100000000000
2600 CURSOR(3,1)=&HFE00      '1111111000000000
2700 CURSOR(4,1)=&HD800      '1101100000000000
2800 CURSOR(5,1)=&H0C00      '0000110000000000
2900 CURSOR(6,1)=&H0600      '0000011000000000
3000 CURSOR(7,1)=&H0300      '0000001100000000
3100 CURSOR(8,1)=&H0000      '0000000000000000
3200 CURSOR(9,1)=&H0000      '0000000000000000
3300 CURSOR(10,1)=&H0000     '0000000000000000
3400 CURSOR(11,1)=&H0000     '0000000000000000
3500 CURSOR(12,1)=&H0000     '0000000000000000
3600 CURSOR(13,1)=&H0000     '0000000000000000
3700 CURSOR(14,1)=&H0000     '0000000000000000
3800 CURSOR(15,1)=&H0000     '0000000000000000
3900 '
4000 '  Define cursor shape, color, and center
4100 '
4200 M1% = 9
4300 M2% = 0      '  Horizontal hot spot
4400 M3% = 0      '  Vertical hot spot
4500 CALL MOUSE(M1%, M2%, M3%, CURSOR(0,0))
```

Function 10: Set Text Cursor

Input	Output
M1% = 10 M2% = cursor select M3% = screen mask value/scan line start M4% = cursor mask value/scan line stop	--

Selects the software or hardware text cursor. If the software text cursor is selected, this function defines the character attributes of the cursor when in text mode. If the hardware text cursor is selected, this function defines the first and last scan lines to be shown on the screen.

The value of the parameter M2% selects the cursor type. If the value is 0, the software text cursor is selected. If the value is 1, the hardware text cursor is selected.

If the software text cursor is selected, the parameters M3% and M4% must specify the screen and cursor masks. These masks define the attributes of a character when the cursor is over it. (See the section in this chapter, "The Software Text Cursor.") The mask values depend on the display adapter in your computer.

If the hardware cursor is selected, the parameters M3% and M4% must contain the line numbers of the first and last scan line in the cursor to be show on the screen. (See the section in this chapter, "The Hardware Text Cursor.") The line numbers depend on the display adapter in your computer.

Example

To create a text cursor that inverts the foreground and background colors:

```
100  M1%=10
110  M2%=0              ' select text cursor
120  M3%=&HFFFF         ' screen mask
130  M4%=&H7700         ' cursor mask
140  CALL MOUSE(M1%,M2%,M3%,M4%)
```

If you have installed both a color/graphics board and a monochrome board in your computer, you can change lines 120 and 130 to read as follows:

```
120  M3%=&H77FF
130  M4%=&HEA00
```

Function 11: Read Mouse Motion Counters

Input	Output
M1% = 11	M3% = count (horizontal) M4% = count (vertical)

Returns the horizontal and vertical mickey count since the last call to this function. The mickey count is the distance in 1/100 inch increments that the mouse has moved. (See the section in this chapter, "The Mouse Unit of Distance: The Mickey.")

The mickey count is always within the range –32768 to 32767. A positive horizontal count specifies a motion to the right. A positive vertical count specifies a motion to the bottom of the screen. Overflow is ignored.

The mickey count is set to 0 after the call is completed.

Example

```
100  '
200  '  Get the mickey count
300  '
400  M1% = 11
500  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 12: Set User-Defined Subroutine Input Mask

Input	Output
M1% = 12 M3% = call mask M4% = address offset to subroutine	--

Sets the call mask and subroutine address for the mouse software interrupts. The mouse software interrupts automatically, stops execution of your program and calls the specified subroutine whenever one or more of the conditions defined by the call mask occur. On completion of the subroutine, your program will continue execution at the point of interruption.

The call mask, a single integer value, defines which conditions will cause an interrupt. Each bit in the call mask corresponds to a specific condition as shown here:

Mask bit	Condition
0	cursor position changes
1	left button pressed
2	left button released
3	right button pressed
4	right button released
5-15	not used

To enable an interrupt for a given condition, set the corresponding call mask bit to 1 and pass the mask as parameter M3%. To disable a condition, set the corresponding bit to 0 and pass the mask. All conditions are automatically disabled by Function 0.

When the mouse software makes a call to the subroutine, it loads the following information into the CPU registers:

Register	Information
AX	condition mask (similar to the call mask except a bit is set only if the condition has occurred)
BX	button state
CX	cursor position (horizontal)
DX	cursor position (vertical)

To use this function with the BASIC interpreter, load an assembly language subroutine into memory (use the same segment as the BASIC interpreter), assign the entry address of the subroutine to an integer variable, and pass this variable to Function 12 as the fourth parameter.

To use this function in assembly language, load the ES register with the subroutine's segment address, and load the DX register with the subroutine's offset.

Example

Assuming that a subroutine has been loaded into memory and that the integer variable SKETCH has been assigned the subroutine's entry address, use the following statements to set up calls on any press of the left button.

```
100 '
200 '  Call subroutine SKETCH on left button press
300 '
400 M1% = 12
500 M3% = &H4000
600 M4% = SKETCH
700 CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 13: Light Pen Emulation Mode On

Input	Output
M1% = 13	--

Enables the light pen emulation by the mouse. When the mouse emulates the light pen, calls to the PEN function, described in the IBM *BASIC* manual, will return the cursor position at the last "pen down."

"Pen down" and "pen off the screen" are controlled by the mouse buttons. The pen is down when both buttons are down. The pen is off the screen when both buttons are up.

The mouse software enables light pen emulation mode after each reset (Function 0).

Example

```
100  '
200  '  Enable Light Pen
300  '
400  M1% = 13
500  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 14: Light Pen Emulation Mode Off

Input	Output
M1% = 14	--

Disables the light pen emulation. When light pen emulation is disabled, calls to the PEN function, described in the IBM *BASIC* manual, return information about the light pen only.

Example

```
100  '
200  '     Disable Light Pen
300  '
400  M1% = 14
500  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 15: Set Mickey/Pixel Ratio

Input	Output
M1% = 15	--
M3% = mickey/pixel ratio (horizontal)	
M4% = mickey/pixel ratio (vertical)	

Sets the mickey to pixel ratio for mouse motion. (See the section in this chapter, "The Mouse Unit of Distance: The Mickey.") The horizontal and vertical ratios specify a number of mickeys per 8 pixels. The values must be in the range 1 to 32767.

The default value for the horizontal ratio is 8 mickeys to 8 pixels. The default value for the vertical ratio is 16 mickeys to 8 pixels. In the default setting, it takes 6.4 inches of mouse travel to move the cursor across the screen horizontally, and 4.0 inches of travel to move it vertically.

Example

```
100  '
200  '  Set mickey/pixel ratio at 16 to 8 and 32 to 8
300  '
400  M1% = 15
500  M3% = 16          ' horizontal ratio
600  M4% = 32          ' vertical ratio
700  CALL MOUSE(M1%, M2%, M3%, M4%)
```

Function 16: Conditional Off

Input	Output
AX = 16 CX = upper x screen coordinate value DX = upper y screen coordinate value SI = lower x screen coordinate value DI = lower y screen coordinate value	--

Function 16 defines a region on the screen for updating. If the mouse pointer is in or moves into the defined region, Function 16 hides the mouse cursor while the region is being updated. After calling Function 16, a subsequent call to Function 1 (Show Cursor) is needed to show the cursor again.

The region is defined by placing the screen coordinate values in the four parameter registers. The CX and DX registers define the upper left corner of the region while the SI and DI registers define the lower right corner.

Function 16 is similar to Function 2 (Hide Cursor) but is for advanced applications that require quicker screen updates. Because of the number of parameters required, Function 16 can only be used in assembly language routines. To use this function from high-level language programs, see the language reference manual on how to call assembly language routines.

Example

```
;Define screen region for conditional off
;
MOV     AX,   16    ;Load Function 16
MOV     CX,   10    ;Define upper x value of region
MOV     DX,   30    ;Define upper y value of region
MOV     DI,   40    ;Define lower y value of region
MOV     SI,   80    ;Define lower x value of region
INT     51          ;Interrupt to mouse function
.
.                   ;Screen update routine
.
MOV     AX,   1     ;Load Function 1 to show cursor
INT     51          ;Interrupt to mouse function
```

Function 19: Set Double Speed Threshold

Input	Output
M1% = 19 M4% = Threshold speed in mickeys/second	--

Function 19 sets the threshold speed for doubling the cursor's motion on the screen. Using Function 19 will make it easier to point at images widely separate on the screen.

Parameter M4% defines the threshold speed of the mouse. If no value is given, a preset value of 64 mickeys per second is assigned. If the mouse movement speed exceeds the value in M4%, cursor motion doubles in speed. The threshold speed is set until Function 19 is called again.

The speed doubling feature is not turned off in the same sense as a switch is turned off. It can be disabled by setting the value of M4% to a speed sufficiently higher than the mouse can obtain (10000 for example) and then calling Function 19.

Example

```
100  '
110  '     Set threshold to 32 mickeys/sec
120  '
130  M1% = 19
140  M4% = 32   'mickeys/seconds
150  CALL MOUSE (M1%, M2%, M3%, M4%)
.
.
.
1000 '     Turn off speed doubling
1010 M1% = 19
1020 M4% = 10000   'mickeys/seconds
1030 CALL MOUSE (M1%, M2%, M3%, M4%)
```

Appendix A
Cleaning Instructions

Even in the best of environments, the Microsoft Mouse will pick up dirt and lint which can build up inside the ball's socket and eventually retard mouse performance. Use the following procedure to clean the Microsoft Mouse whenever free motion of the ball is restricted. To clean the mouse, you will need a medium Phillips head screwdriver, a clean, dry cloth, and a toothpick.

1. Turn the power to your computer off.

2. Unplug the Microsoft Mouse connector from the serial port.

3. Turn the mouse upside down so that the ball is face up. See Figure A.1.

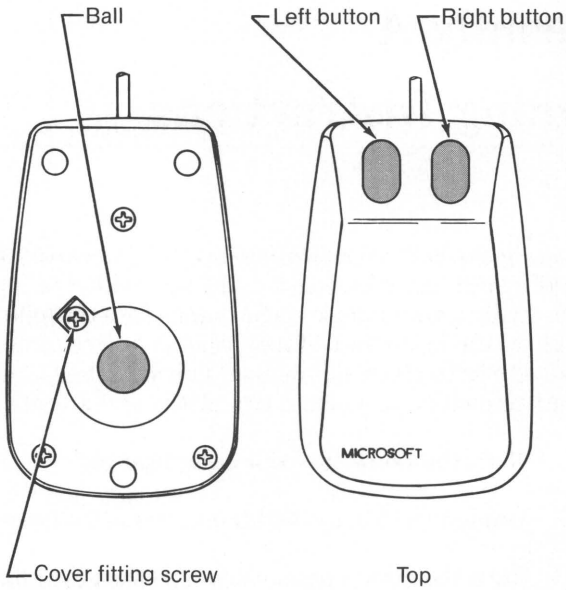


Figure A.1 Ball and Cover Fitting Screw

4. Locate the ball retaining screw. Use the Phillips screwdriver to remove the screw by turning counterclockwise (CCW). Set the screw aside in a safe place.

5. Holding your hand over the ball and retaining bracket, turn the mouse rightside up. The ball and retaining bracket should drop into your hand. If not, shake the ball loose.

Caution

Do not tap the mouse against a hard surface to loosen the ball. It may damage the internal circuitry.

---

6. Once the ball is free, wipe it with a clean cloth.

7. Wipe away any dirt or lint inside the socket. If any lint is clogged inside the socket, use a toothpick to loosen it. Do not use any sharp, metallic object.

8. Return the ball to its socket and place the retaining bracket over it. Insert the retaining screw and tighten.

9. Reconnect the Microsoft Mouse connector and turn power to your computer back on.

The Microsoft Mouse cleaning procedure is now complete.

Appendix B
Piano Program Listing

This appendix presents the complete source to the Piano demonstration program. The program is written in BASIC for the IBM Personal Computer's BASIC interpreter. The following is an explanation of the program details:

Line Numbers	Comments
1000-1090	Copyright message.
1100-1160	Set up music, clear graphics screen to blue.
1170-1250	Read in the frequencies for the various piano keys.
1260-1380	As explained in Chapter 4, these statements link the mouse software and the program.
1390-1430	Function 15 sets the mouse sensitivity. With this setting, a horizontal movement of 3.2 inches moves the cursor across the entire screen. This relatively high sensitivity permits songs to be played rapidly. Accuracy is no problem since the piano keys are large.
1440-1620	The integer array CURSOR contains the screen mask and the cursor mask. The masks define the shape and color of the cursor. These statements define the screen mask; the mask is set to all ones. The mask will be logically ANDed with screen under the cursor.

1630-1810	These statements define the cursor mask. The values will be exclusively ORed with the result of the AND operation to create the cursor shape and color. In this case, the cursor shape is a north-pointing arrowhead. Its color is the inverse of whatever is below it.
1820-1860	Function 9 sets the cursor shape. It also defines the cursor hot spot. In this case, the hot spot is the tip of the arrowhead. The mouse software will automatically prevent the cursor hot spot from leaving the screen.
1870-1930	These statements simply read in the Microsoft logo from precalculated data. The statements place the data on the screen.
1940-2150	These statements draw the white and black piano keys.
2160-2200	These statements draw the "quit" box in the lower right corner.
2210-2240	Function 4 centers the cursor to just under the piano keys.
2250	Function 1 turns the cursor on. The cursor appears on the screen and can be moved by using the mouse.

2260-2290	Function 3 gives the status of the two mouse buttons and the location of the cursor. This is probably the most common mouse function used in applications.
2300-2370	Some decision making is performed. If both mouse buttons are up, or if the mouse is not on the piano keyboard, then any sound that might be playing is turned off.
2380-2430	At this point, the mouse button is down over the quit box. The program turns off the mouse cursor, clears the screen, then quits.
2440-2510	The program has determined a button is down over the piano keyboard. These statements determine which key the mouse cursor is over.
2520-2570	The note is played by the SOUND statement set with the correct frequency. This note is played in the background as the program loops back to line 2090.
2580-2630	This data contains the correct frequency to play the musical notes.
2640-3050	Data to draw the Microsoft logo using the PUT statement.

```
1000 '
1010 '                    THE VIRTUAL PIANO
1020 '
1030 '  COPYRIGHT (C) 1983 BY MICROSOFT CORPORATION
1040 '                 WRITTEN BY CHRIS PETERS
1050 '
1060 '-------------------------------------------------
1070 '
1080 '  INITIALIZE
1090 '
1100 DEFINT A-Z
1110 DIM CURSOR(15,1),FREQ(27,2),MICROSOFT(839)
1120 KEY OFF
1130 PLAY"MF"
1140 SCREEN 1
1150 COLOR 1,1
1160 CLS
1170 '
1180 ' Read in the flat, normal, and sharp note frequencies
1190 '
1200 FOR J=0 TO 2
1210 FOR I=0 TO 6
1220 READ K
1230 FREQ(I,J)=K : FREQ(I+7,J)=K*2 :
     FREQ(I+14,J)=K*4 : FREQ(I+21,J)=K*8
1240 NEXT
1250 NEXT
1260 '
1270 ' Determine mouse driver location, if not found, quit.
1280 '
1290 DEF SEG=0
1300 MSEG=256*PEEK(51*4+3)+PEEK(51*4+2)   'Get mouse
                                           segment
1310 MOUSE=256*PEEK(51*4+1)+PEEK(51*4)+2 'Get mouse
                                           offset
1320 IF MSEG OR MOUSE THEN 1370
1330 PRINT"Mouse driver not found"
1340 PRINT                                 'Not found
                                           so print error.
1350 PRINT"Press any key to return to system"
1360 I$=INKEY$ : IF I$=" " THEN 1360 ELSE SYSTEM
1370 DEF SEG=MSEG                          'Set mouse segment
1380 M1=0 : CALL MOUSE(M1,M2,M3,M4)        'Initialize the mouse
1390 '
1400 ' Set mouse sensitivity
1410 '
```

```
1420 M1 = 15 : M3=4 : M4=8
1430 CALL MOUSE(M1,M2,M3,M4)
1440 '
1450 ' Define the "logical and" cursor mask
1460 '
1470 CURSOR( 0,0)=&HFFFF        ' Binary 1111111111111111
1480 CURSOR( 1,0)=&HFFFF        ' Binary 1111111111111111
1490 CURSOR( 2,0)=&HFFFF        ' Binary 1111111111111111
1500 CURSOR( 3,0)=&HFFFF        ' Binary 1111111111111111
1510 CURSOR( 4,0)=&HFFFF        ' Binary 1111111111111111
1520 CURSOR( 5,0)=&HFFFF        ' Binary 1111111111111111
1530 CURSOR( 6,0)=&HFFFF        ' Binary 1111111111111111
1540 CURSOR( 7,0)=&HFFFF        ' Binary 1111111111111111
1550 CURSOR( 8,0)=&HFFFF        ' Binary 1111111111111111
1560 CURSOR( 9,0)=&HFFFF        ' Binary 1111111111111111
1570 CURSOR(10,0)=&HFFFF        ' Binary 1111111111111111
1580 CURSOR(11,0)=&HFFFF        ' Binary 1111111111111111
1590 CURSOR(12,0)=&HFFFF        ' Binary 1111111111111111
1600 CURSOR(13,0)=&HFFFF        ' Binary 1111111111111111
1610 CURSOR(14,0)=&HFFFF        ' Binary 1111111111111111
1620 CURSOR(15,0)=&HFFFF        ' Binary 1111111111111111
1630 '
1640 ' Define the "exclusive or" cursor mask
1650 '
1660 CURSOR( 0,1)=&H0300        ' Binary 0000001100000000
1670 CURSOR( 1,1)=&H0300        ' Binary 0000001100000000
1680 CURSOR( 2,1)=&H0FC0        ' Binary 0000111111000000
1690 CURSOR( 3,1)=&H0FC0        ' Binary 0000111111000000
1700 CURSOR( 4,1)=&H3FF0        ' Binary 0011111111110000
1710 CURSOR( 5,1)=&H3FF0        ' Binary 0011111111110000
1720 CURSOR( 6,1)=&HFCFC        ' Binary 1111110011111100
1730 CURSOR( 7,1)=&HC00C        ' Binary 1100000000001100
1740 CURSOR( 8,1)=&H0000        ' Binary 0000000000000000
1750 CURSOR( 9,1)=&H0000        ' Binary 0000000000000000
1760 CURSOR(10,1)=&H0000        ' Binary 0000000000000000
1770 CURSOR(11,1)=&H0000        ' Binary 0000000000000000
1780 CURSOR(12,1)=&H0000        ' Binary 0000000000000000
1790 CURSOR(13,1)=&H0000        ' Binary 0000000000000000
1800 CURSOR(14,1)=&H0000        ' Binary 0000000000000000
1810 CURSOR(15,1)=&H0000        ' Binary 0000000000000000
1820 '
1830 ' Set the mouse cursor shape
1840 '
1850 M1 = 9 : M2 = 6 : M3 = 0
1860 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
1870 '
1880 ' Draw the MICROSOFT logo from pre-calculated data
1890 '
```

```
1900 FOR I=0 TO 779
1910 READ MICROSOFT(I)
1920 NEXT
1930 PUT(62,0),MICROSOFT,PSET
1940 '
1950 ' Initialize keyboard size parameters
1960 '
1970 YL = 60 : WKL = 80 : BKL = 45 : KW = 15 : WKN = 21
1980 XL = 320-KW*WKN : YH = YL + WKL : XH = 319 : BKW2=KW3
1990 QX = 272 : QY = 176
2000 '
2010 ' Draw the white keys
2020 '
2030 LINE (XL,YL)-(XH,YH),3,BF
2040 FOR I=XL TO XH STEP KW
2050 LINE (I,YL)-(I,YH),0
2060 NEXT
2070 '
2080 ' Draw the black keys
2090 '
2100 C=6
2110 FOR X=XL TO XH STEP KW
2120 C=C+1 : IF C=7 THEN C=0
2130 IF C=0 OR C=3 THEN 2150
2140 LINE(X-BKW2,YL)-(X+BKW2,YL+BKL),2,BF
2150 NEXT
2160 '
2170 ' Draw the quit box
2180 '
2190 LINE(QX,QY)-(319,199),3,B
2200 LOCATE 24,36 : PRINT"Quit";
2210 '
2220 ' Set mouse cursor location, then turn on cursor
2230 '
2240 M1 = 4 : M3 = 320 : M4 = 160 : CALL MOUSE(M1,M2,M3,M4)
2250 M1 = 1 : CALL MOUSE(M1,M2,M3,M4)
2260 '
2270 '     MAIN LOOP
2280 '
2290 M1=3:CALL MOUSE(M1,BT,MX,MY)        'Get mouse location
                                              and button status
2300 IF (BT AND 2) THEN OTV=7 : GOTO 2340
                        ' If right button down, set high octave
2310 IF (BT AND 1) THEN OTV=0 : GOTO 2340
                        ' If left button down, set lower octave
2320 SOUND 442,0    ' If both buttons up, turn off sound
2330 GOTO 2290      ' Keep looping...
2340 MX = MX2       ' Correct for medium resolution screen
```

```
2350 IF MX <= XL OR MY < YL THEN 2320    'If above keyboard,
                                           turn off sound
2360 IF MY <= YH THEN 2470               ' If on keyboard,
                                           play sound
2370 IF MY < QY OR MX < QX THEN 2320     ' If above quit box,
                                           turn off sound
2380 '
2390 ' Button down inside the quit box
2400 '
2410 M1=2 : CALL MOUSE(M1,M2,M3,M4)      ' Turn off mouse
                                           cursor
2420 CLS                                 ' Clear screen
2430 END                                 ' Quit
2440 '
2450 ' Button down over keyboard, determine which key
2460 '
2470 WKY = (MX-XL)KW+OTV : R = 1         ' Get which white
                                           key cursor is over
2480 IF MY > YL+BKL THEN 2560            ' Is it lower than
                                           the black keys?
2490 MK=(MX-XL) MOD KW                   ' No, get which
                                           side of key
2500 IF MK <= BKW2 THEN R=0 : GOTO 2560  ' Is it the left
                                           black key?
2510 IF MK >= KW-BKW2 THEN R=2           ' Is it the right
                                           black key?
2520 '
2530 ' Play the note. For BASIC interpreter duration = 2
2540 '                For BASIC compiler   duration = 1
2550 '
2560 SOUND FREQ(WKY,R),2
2570 GOTO 2290                           'Continue looping
2580 '
2590 ' Musical note frequencies
2600 '
2610 DATA 131,139,156,175,185,208,233
2620 DATA 131,147,165,175,196,220,247
2630 DATA 139,156,165,185,208,233,247
```

```
2640 '
2650 ' Data to draw the MICROSOFT logo
2660 '
2670 DATA 462,28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2680 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2690 DATA 0,0,0,-193,240,0,0,0,0,0,0,0,0,0,0,0,0,0
2700 DATA 0,0,0,0,0,768,-1,0,0,0,0,3840,-1,-16129,0,
          -253,0,0,-193,240
2710 DATA 0,0,0,0,0,0,0,0,0,-193,0,16128,4095,252,16128,
          -1,240,-256,-769,0
2720 DATA 0,0,0,0,-193,240,768,-1,255,768,-1,1023,-1,-1,
          240,0,0,0,-193,192
2730 DATA -256,4095,252,-253,-1,255,-256,-1,240,-253,-1,
          -1,768,-1,255,16128,-1,-3841,768,-1
2740 DATA 1023,-1,-1,240,0,0,0,-193,192,-256,4095,252,
          -193,-1,-3841,-256,-1,252,-1009,0
2750 DATA -256,4032,-1,-16129,-253,-1,-1,768,-1,1023,-1,
          -1,240,0,0,0,-193,240,-253,4095
2760 DATA 252,-3841,0,-961,-256,-1,255,0,0,0,3840,-1,
          -16129,-241,0,-253,960,-1,1023,-1
2770 DATA -1,240,0,0,0,-193,240,-253,4095,1020,255,0,
          -253,-256,4032,-16129,-1,-1,-1,4092
2780 DATA 4095,-16129,-4033,0,16128,1008, -1,1023,-1,-1,
          240,0,0,0,-193,252,-241,4095,1020,252
2790 DATA 0,-256,-256,960,-15361,252,0,0,4095,1023,-16129,
          -16321,0,3840,1008,255,0,3840,252,0
2800 DATA 0,0,0,-193,252,-241,4095,4092,240,0,16128,-64,
          192,-16129,0,0,0,3840,255,0
2810 DATA 255,0,768,1020,255,0,3840,252,0,0,0,0,-193,255,
          -193,4095,4092,240,0,16128
2820 DATA -64,192,-12289,-1,192,-241,-12289,-3841,0,255,0,
          768,1020,255,0,3840,252,0,0,0
2830 DATA 0,-193,255,-193,4095,16380,192,0,3840,-16,960,
          -12289,240,0,0,-15553,-1,768,252,0
2840 DATA 0,1023,255,0,3840,252,0,0,0,0,-193,-16129,-1,
          4095,16380,192,0,0,-256,4032
2850 DATA -16129,0,0,0,768,-1,1008,252,0,0,1023,-1,255,
          3840,252,0,0,0,0,-3265
2860 DATA -16129,-3073,4095,16380,192,0,0,-256,-1,4095,
          -1,0,-253,-16129,-1,1020,252,0,0,1023
2870 DATA -1,255,3840,252,0,0,0,0,-3265,-3073,-3073,4095,
          16380,192,0,0,-256,-1,4095,240
2880 DATA 0,0,-16321,-241,1023,252,0,0,1023,-1,255,3840,
          252,0,0,0,0,-4033,-3073,-15361
2890 DATA 4095,16380,192,0,0,-256,-1,252,0,0,0,0,16128,
          -15361,252,0,0,1023,-1,255
2900 DATA 3840,252,0,0,0,0,-4033,-1,-15361,4095,16380,192,
          0,0,-256,-1,4092,240,0,0
```

```
2910 DATA -16321,768,-3073,252,0,0,1023,255,0,3840,252,0,
          0,0,0,-4033,-193,1023,4095,4092
2920 DATA 240,0,0,-256,-64,4092,-1,192,-241,-16129,0,
          -3841,255,0,768,1020,255,0,3840,252
2930 DATA 0,0,0,0,-4033,-193,1023,4095,4092,240,0,16128,
          -64,4032,255,0,0,0,16128,252
2940 DATA -3841,255,0,768,1020,255,0,3840,252,0,0,0,0,
          -4033,-241,1020,4095,1020,252,0
2950 DATA -256,-256,960,1023,252,0,0,16383,1023,-3841,
          -16321,0,3840,1008,255,0,3840,252,0,0
2960 DATA 0,0,-4033,-241,1020,4095,1020,255,0,-253,-256,
          960,-16129,-1,-1,-1,16380,-1,-3841,-4033
2970 DATA 0,16128,1008,255,0,3840,252,0,0,0,0,-4033,-253,
          1008,4095,252,-3841,0,-961,-256
2980 DATA 192,-16129,0,0,0,3840,-1,-16129,-241,0,-253,960,
          255,0,3840,252,0,0,0,0
2990 DATA -4033,-253,1008,4095,252,-193,768,-3841,-256,192,
          -16129,-1009,0,-256,4032,-1,255,-253,240,-193
3000 DATA 768,255,0,3840,252,0,0,0,0,-4033,-256,960,4095,
          252,-253,-1,255,-256,192,-16129
3010 DATA -253,-1,-1,768,-1,252,16128,-1,-3841,768,255,0,
          3840,252,0,0,0,0,-4033,-256
3020 DATA 960,4095,252,16128,-1,240,-256,192,-16129,0,0,
          0,0,-193,192,768,-1,255,768,255
3030 DATA 0,3840,252,0,0,0,0,0,0,0,0,0,768,-1,0,0,0,0,
          3840,-1
3040 DATA -16129,0,0,0,0,-193,240,0,0,0,0,0,0,0,0,0,0,0,0
3050 DATA 0,0,0,0,0,0,0,0,0,-193,240,0,0,0,0,0,0,0,0,0,0
```

Appendix C
Sample Cursors

This appendix describes eight sample graphics cursors. These sample cursors illustrate the wide variety of cursor shapes that can be defined for use in BASIC application programs.

The sample cursors are designed for high-resolution graphics mode. Each cursor is a white shape with a black outline on a transparent field. The shape suggests the type of action you may take with the mouse. For example, an arrow usually means "make a selection by pointing at an item."

To use a sample cursor in your own BASIC program, copy the BASIC statements presented for the cursor directly to your program. Type the statements exactly as shown, using line numbers that are consistent with your program's numbering scheme.

To use a sample cursor in an assembly or high-level language program, define an array in your program and assign the values given for each cursor to the array elements. Assign the values so that their storage order is identical to their storage order in a BASIC program.

The statements in this appendix only define the cursor's shape. It is up to you to define the action associated with a cursor by including the necessary statements in your program.

Standard Cursor Shape

The standard cursor shape is a solid arrow which points up and to the left. The hot spot is just beyond the arrow's tip, so you can point to an item without covering it. The standard cursor is the most convenient shape when using the mouse to choose or select items from the screen.

```
100 '
200 ' Define the screen mask
300 '
400 CURSOR( 0,0)=&H3FFF      'Binary 0011111111111111
500 CURSOR( 1,0)=&H1FFF      'Binary 0001111111111111
600 CURSOR( 2,0)=&H0FFF      'Binary 0000111111111111
700 CURSOR( 3,0)=&H07FF      'Binary 0000011111111111
800 CURSOR( 4,0)=&H03FF      'Binary 0000001111111111
900 CURSOR( 5,0)=&H01FF      'Binary 0000000111111111
1000 CURSOR( 6,0)=&H00FF     'Binary 0000000011111111
1100 CURSOR( 7,0)=&H007F     'Binary 0000000001111111
1200 CURSOR( 8,0)=&H003F     'Binary 0000000000111111
1300 CURSOR( 9,0)=&H001F     'Binary 0000000000011111
1400 CURSOR(10,0)=&H01FF     'Binary 0000000111111111
1500 CURSOR(11,0)=&H10FF     'Binary 0001000011111111
1600 CURSOR(12,0)=&H30FF     'Binary 0011000011111111
1700 CURSOR(13,0)=&HF87F     'Binary 1111100001111111
1800 CURSOR(14,0)=&HF87F     'Binary 1111100001111111
1900 CURSOR(15,0)=&HFC3F     'Binary 1111110000111111
```

```
2000 '
2100 ' Define cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000        'Binary 0000000000000000
2400 CURSOR( 1,1)=&H4000        'Binary 0100000000000000
2500 CURSOR( 2,1)=&H6000        'Binary 0110000000000000
2600 CURSOR( 3,1)=&H7000        'Binary 0111000000000000
2700 CURSOR( 4,1)=&H7800        'Binary 0111100000000000
2800 CURSOR( 5,1)=&H7C00        'Binary 0111110000000000
2900 CURSOR( 6,1)=&H7E00        'Binary 0111111000000000
3000 CURSOR( 7,1)=&H7F00        'Binary 0111111100000000
3100 CURSOR( 8,1)=&H7F80        'Binary 0111111110000000
3200 CURSOR( 9,1)=&H78C0        'Binary 0111111111000000
3300 CURSOR(10,1)=&H7C00        'Binary 0111110000000000
3400 CURSOR(11,1)=&H4600        'Binary 0100011000000000
3500 CURSOR(12,1)=&H0600        'Binary 0000011000000000
3600 CURSOR(13,1)=&H0300        'Binary 0000001100000000
3700 CURSOR(14,1)=&H0300        'Binary 0000001100000000
3800 CURSOR(15,1)=&H0180        'Binary 0000000110000000
3900 '
4000 ' Define cursor shape, color, and hot spot
4100 '
4200 M1% = 9
4300 M2% = -1 ' Horizontal hot spot
4400 M3% = -1 ' Vertical hot spot
4500 CALL MOUSE(M1% , M2%, M3%, CURSOR(0,0))
```

Up Arrow

The up arrow is a solid, up-directed arrow with the hot spot at the tip. This shape is useful when directing a motion on the screen with the mouse.

```
100 '
200 ' Define the screen mask
300 '
400 CURSOR( 0,0)=&HF9FF      ' Binary 1111100111111111
500 CURSOR( 1,0)=&HF0FF      ' Binary 1111000011111111
600 CURSOR( 2,0)=&HE07F      ' Binary 1110000001111111
700 CURSOR( 3,0)=&HE07F      ' Binary 1110000001111111
800 CURSOR( 4,0)=&HC03F      ' Binary 1100000000111111
900 CURSOR( 5,0)=&HC03F      ' Binary 1100000000111111
1000 CURSOR( 6,0)=&H801F     ' Binary 1000000000011111
1100 CURSOR( 7,0)=&H801F     ' Binary 1000000000011111
1200 CURSOR( 8,0)=&H000F     ' Binary 0000000000001111
1300 CURSOR( 9,0)=&H000F     ' Binary 0000000000001111
1400 CURSOR(10,0)=&HF0FF     ' Binary 1111000011111111
1500 CURSOR(11,0)=&HF0FF     ' Binary 1111000011111111
1600 CURSOR(12,0)=&HF0FF     ' Binary 1111000011111111
1700 CURSOR(13,0)=&HF0FF     ' Binary 1111000011111111
1800 CURSOR(14,0)=&HF0FF     ' Binary 1111000011111111
1900 CURSOR(15,0)=&HF0FF     ' Binary 1111000011111111
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000        ' Binary 0000000000000000
2400 CURSOR( 1,1)=&H0600        ' Binary 0000011000000000
2500 CURSOR( 2,1)=&H0F00        ' Binary 0000111100000000
2600 CURSOR( 3,1)=&H0F00        ' Binary 0000111100000000
2700 CURSOR( 4,1)=&H1F80        ' Binary 0001111110000000
2800 CURSOR( 5,1)=&H1F80        ' Binary 0001111110000000
2900 CURSOR( 6,1)=&H3FC0        ' Binary 0011111111000000
3000 CURSOR( 7,1)=&H3FC0        ' Binary 0011111111000000
3100 CURSOR( 8,1)=&H7FE0        ' Binary 0111111111100000
3200 CURSOR( 9,1)=&H0600        ' Binary 0000011000000000
3300 CURSOR(10,1)=&H0600        ' Binary 0000011000000000
3400 CURSOR(11,1)=&H0600        ' Binary 0000011000000000
3500 CURSOR(12,1)=&H0600        ' Binary 0000011000000000
3600 CURSOR(13,1)=&H0600        ' Binary 0000011000000000
3700 CURSOR(14,1)=&H0600        ' Binary 0000011000000000
3800 CURSOR(15,1)=&H0000        ' Binary 0000000000000000
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 5 ' horizontal hot spot
4400 M3 = 0 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Left Arrow

The left arrow is a solid, left-directed arrow with the hot spot at the tip. This shape is useful when directing a motion on the screen with the mouse. To generate a right arrow, just reverse the binary bit pattern for each array element and move the hot spot to the new tip. For example, the first element, Binary 1111111-000011111 (&HFE1F), becomes Binary 1111100001111111 (&HF87F).

```
100 '
200 ' Define the screen mask
300 '
400 CURSOR( 0,0)=&HFE1F        ' Binary 1111111000011111
500 CURSOR( 1,0)=&HF01F        ' Binary 1111000000011111
600 CURSOR( 2,0)=&H0000        ' Binary 0000000000000000
700 CURSOR( 3,0)=&H0000        ' Binary 0000000000000000
800 CURSOR( 4,0)=&H0000        ' Binary 0000000000000000
900 CURSOR( 5,0)=&HF01F        ' Binary 1111000000011111
1000 CURSOR( 6,0)=&HFE1F       ' Binary 1111111000011111
1100 CURSOR( 7,0)=&HFFFF       ' Binary 1111111111111111
1200 CURSOR( 8,0)=&HFFFF       ' Binary 1111111111111111
1300 CURSOR( 9,0)=&HFFFF       ' Binary 1111111111111111
1400 CURSOR(10,0)=&HFFFF       ' Binary 1111111111111111
1500 CURSOR(11,0)=&HFFFF       ' Binary 1111111111111111
1600 CURSOR(12,0)=&HFFFF       ' Binary 1111111111111111
1700 CURSOR(13,0)=&HFFFF       ' Binary 1111111111111111
1800 CURSOR(14,0)=&HFFFF       ' Binary 1111111111111111
1900 CURSOR(15,0)=&HFFFF       ' Binary 1111111111111111
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000        ' Binary 0000000000000000
2400 CURSOR( 1,1)=&H00C0        ' Binary 0000000011000000
2500 CURSOR( 2,1)=&H07C0        ' Binary 0000011111000000
2600 CURSOR( 3,1)=&H7FFE        ' Binary 0111111111111110
2700 CURSOR( 4,1)=&H07C0        ' Binary 0000011111000000
2800 CURSOR( 5,1)=&H07C0        ' Binary 0000000011000000
2900 CURSOR( 6,1)=&H0000        ' Binary 0000000000000000
3000 CURSOR( 7,1)=&H0000        ' Binary 0000000000000000
3100 CURSOR( 8,1)=&H0000        ' Binary 0000000000000000
3200 CURSOR( 9,1)=&H0000        ' Binary 0000000000000000
3300 CURSOR(10,1)=&H0000        ' Binary 0000000000000000
3400 CURSOR(11,1)=&H0000        ' Binary 0000000000000000
3500 CURSOR(12,1)=&H0000        ' Binary 0000000000000000
3600 CURSOR(13,1)=&H0000        ' Binary 0000000000000000
3700 CURSOR(14,1)=&H0000        ' Binary 0000000000000000
3800 CURSOR(15,1)=&H0000        ' Binary 0000000000000000
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 0 ' horizontal hot spot
4400 M3 = 3 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Check Mark

The check mark is a solid figure with the hot spot in the center of the "V" formed by the check. The shape can be used when checking off items from a list with the mouse or while a program is checking some aspect of its operation.

```
100 '
200 ' Define the screen mask
300 '
400 CURSOR( 0,0)=&HFFF0        ' Binary 1111111111110000
500 CURSOR( 1,0)=&HFFE0        ' Binary 1111111111100000
600 CURSOR( 2,0)=&HFFC0        ' Binary 1111111111000000
700 CURSOR( 3,0)=&HFF81        ' Binary 1111111110000001
800 CURSOR( 4,0)=&HFF03        ' Binary 1111111100000011
900 CURSOR( 5,0)=&H0607        ' Binary 0000011000000111
1000 CURSOR( 6,0)=&H000F       ' Binary 0000000000001111
1100 CURSOR( 7,0)=&H001F       ' Binary 0000000000011111
1200 CURSOR( 8,0)=&HC03F       ' Binary 1100000000111111
1300 CURSOR( 9,0)=&HF07F       ' Binary 1111000001111111
1400 CURSOR(10,0)=&HFFFF       ' Binary 1111111111111111
1500 CURSOR(11,0)=&HFFFF       ' Binary 1111111111111111
1600 CURSOR(12,0)=&HFFFF       ' Binary 1111111111111111
1700 CURSOR(13,0)=&HFFFF       ' Binary 1111111111111111
1800 CURSOR(14,0)=&HFFFF       ' Binary 1111111111111111
1900 CURSOR(15,0)=&HFFFF       ' Binary 1111111111111111
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000        ' Binary 0000000000000000
2400 CURSOR( 1,1)=&H0006        ' Binary 0000000000000110
2500 CURSOR( 2,1)=&H000C        ' Binary 0000000000001100
2600 CURSOR( 3,1)=&H0018        ' Binary 0000000000011000
2700 CURSOR( 4,1)=&H0030        ' Binary 0000000000110000
2800 CURSOR( 5,1)=&H0060        ' Binary 0000000001100000
2900 CURSOR( 6,1)=&H70C0        ' Binary 0111000011000000
3000 CURSOR( 7,1)=&H1D80        ' Binary 0001110110000000
3100 CURSOR( 8,1)=&H0700        ' Binary 0000011100000000
3200 CURSOR( 9,1)=&H0000        ' Binary 0000000000000000
3300 CURSOR(10,1)=&H0000        ' Binary 0000000000000000
3400 CURSOR(11,1)=&H0000        ' Binary 0000000000000000
3500 CURSOR(12,1)=&H0000        ' Binary 0000000000000000
3600 CURSOR(13,1)=&H0000        ' Binary 0000000000000000
3700 CURSOR(14,1)=&H0000        ' Binary 0000000000000000
3800 CURSOR(15,1)=&H0000        ' Binary 0000000000000000
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 6 ' horizontal hot spot
4400 M3 = 7 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Pointing Hand

The pointing hand is a solid figure with the hot spot at the tip of the extended finger. The pointing hand is another convenient shape to use when choosing or selecting items from the screen, especially if the items are represented by icons or symbols such as the keys of a piano keyboard or a calculator.

```
100 '
200 ' Define the screen mask
300 '
400 CURSOR( 0,0)=&HE1FF       ' Binary 1110000111111111
500 CURSOR( 1,0)=&HE1FF       ' Binary 1110000111111111
600 CURSOR( 2,0)=&HE1FF       ' Binary 1110000111111111
700 CURSOR( 3,0)=&HE1FF       ' Binary 1110000111111111
800 CURSOR( 4,0)=&HE1FF       ' Binary 1110000111111111
900 CURSOR( 5,0)=&HE000       ' Binary 1110000000000000
1000 CURSOR( 6,0)=&HE000      ' Binary 1110000000000000
1100 CURSOR( 7,0)=&HE000      ' Binary 1110000000000000
1200 CURSOR( 8,0)=&H0000      ' Binary 0000000000000000
1300 CURSOR( 9,0)=&H0000      ' Binary 0000000000000000
1400 CURSOR(10,0)=&H0000      ' Binary 0000000000000000
1500 CURSOR(11,0)=&H0000      ' Binary 0000000000000000
1600 CURSOR(12,0)=&H0000      ' Binary 0000000000000000
1700 CURSOR(13,0)=&H0000      ' Binary 0000000000000000
1800 CURSOR(14,0)=&H0000      ' Binary 0000000000000000
1900 CURSOR(15,0)=&H0000      ' Binary 0000000000000000
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H1E00        ' Binary 0001111000000000
2400 CURSOR( 1,1)=&H1200        ' Binary 0001001000000000
2500 CURSOR( 2,1)=&H1200        ' Binary 0001001000000000
2600 CURSOR( 3,1)=&H1200        ' Binary 0001001000000000
2700 CURSOR( 4,1)=&H1200        ' Binary 0001001000000000
2800 CURSOR( 5,1)=&H13FF        ' Binary 0001001111111111
2900 CURSOR( 6,1)=&H1249        ' Binary 0001001001001001
3000 CURSOR( 7,1)=&H1249        ' Binary 0001001001001001
3100 CURSOR( 8,1)=&HF249        ' Binary 1111001001001001
3200 CURSOR( 9,1)=&H9001        ' Binary 1001000000000001
3300 CURSOR(10,1)=&H9001        ' Binary 1001000000000001
3400 CURSOR(11,1)=&H9001        ' Binary 1001000000000001
3500 CURSOR(12,1)=&H8001        ' Binary 1000000000000001
3600 CURSOR(13,1)=&H8001        ' Binary 1000000000000001
3700 CURSOR(14,1)=&H8001        ' Binary 1000000000000001
3800 CURSOR(15,1)=&HFFFF        ' Binary 1111111111111111
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 5 ' horizontal hot spot
4400 M3 = 0 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Diagonal Cross

The diagonal cross is a solid figure with the hot spot at the center of the cross. The shape is useful as a pointer in a game, or when canceling an operation or deleting an item from a list.

```
100 '
200 ' Define the screen mask
300 '
400 CURSOR( 0,0)=&H07E0          ' Binary 0000011111100000
500 CURSOR( 1,0)=&H0180          ' Binary 0000000110000000
600 CURSOR( 2,0)=&H0000          ' Binary 0000000000000000
700 CURSOR( 3,0)=&HC003          ' Binary 1100000000000011
800 CURSOR( 4,0)=&HF00F          ' Binary 1111000000001111
900 CURSOR( 5,0)=&HC003          ' Binary 1100000000000011
1000 CURSOR( 6,0)=&H0000         ' Binary 0000000000000000
1100 CURSOR( 7,0)=&H0180         ' Binary 0000000110000000
1200 CURSOR( 8,0)=&H07E0         ' Binary 0000011111100000
1300 CURSOR( 9,0)=&HFFFF         ' Binary 1111111111111111
1400 CURSOR(10,0)=&HFFFF         ' Binary 1111111111111111
1500 CURSOR(11,0)=&HFFFF         ' Binary 1111111111111111
1600 CURSOR(12,0)=&HFFFF         ' Binary 1111111111111111
1700 CURSOR(13,0)=&HFFFF         ' Binary 1111111111111111
1800 CURSOR(14,0)=&HFFFF         ' Binary 1111111111111111
1900 CURSOR(15,0)=&HFFFF         ' Binary 1111111111111111
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000          ' Binary 0000000000000000
2400 CURSOR( 1,1)=&H700E          ' Binary 0111000000001110
2500 CURSOR( 2,1)=&H1C38          ' Binary 0001110000111000
2600 CURSOR( 3,1)=&H0660          ' Binary 0000011001100000
2700 CURSOR( 4,1)=&H03C0          ' Binary 0000001111000000
2800 CURSOR( 5,1)=&H0660          ' Binary 0000011001100000
2900 CURSOR( 6,1)=&H1C38          ' Binary 0001110000111000
3000 CURSOR( 7,1)=&H700E          ' Binary 0111000000001110
3100 CURSOR( 8,1)=&H0000          ' Binary 0000000000000000
3200 CURSOR( 9,1)=&H0000          ' Binary 0000000000000000
3300 CURSOR(10,1)=&H0000          ' Binary 0000000000000000
3400 CURSOR(11,1)=&H0000          ' Binary 0000000000000000
3500 CURSOR(12,1)=&H0000          ' Binary 0000000000000000
3600 CURSOR(13,1)=&H0000          ' Binary 0000000000000000
3700 CURSOR(14,1)=&H0000          ' Binary 0000000000000000
3800 CURSOR(15,1)=&H0000          ' Binary 0000000000000000
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 7 ' horizontal hot spot
4400 M3 = 4 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Rectangular Cross

The rectangular cross is a solid figure with the hot spot at the center of the cross. The shape is useful as a pointer in a game, or when inserting items into a list.

```
 100 '
 200 ' Define the screen mask
 300 '
 400 CURSOR( 0,0)=&HFC3F        ' Binary 1111110000111111
 500 CURSOR( 1,0)=&HFC3F        ' Binary 1111110000111111
 600 CURSOR( 2,0)=&HFC3F        ' Binary 1111110000111111
 700 CURSOR( 3,0)=&H0000        ' Binary 0000000000000000
 800 CURSOR( 4,0)=&H0000        ' Binary 0000000000000000
 900 CURSOR( 5,0)=&H0000        ' Binary 0000000000000000
1000 CURSOR( 6,0)=&HFC3F        ' Binary 1111110000111111
1100 CURSOR( 7,0)=&HFC3F        ' Binary 1111110000111111
1200 CURSOR( 8,0)=&HFC3F        ' Binary 1111110000111111
1300 CURSOR( 9,0)=&HFFFF        ' Binary 1111111111111111
1400 CURSOR(10,0)=&HFFFF        ' Binary 1111111111111111
1500 CURSOR(11,0)=&HFFFF        ' Binary 1111111111111111
1600 CURSOR(12,0)=&HFFFF        ' Binary 1111111111111111
1700 CURSOR(13,0)=&HFFFF        ' Binary 1111111111111111
1800 CURSOR(14,0)=&HFFFF        ' Binary 1111111111111111
1900 CURSOR(15,0)=&HFFFF        ' Binary 1111111111111111
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000        ' Binary 0000000000000000
2400 CURSOR( 1,1)=&H0180        ' Binary 0000000110000000
2500 CURSOR( 2,1)=&H0180        ' Binary 0000000110000000
2600 CURSOR( 3,1)=&H0180        ' Binary 0000000110000000
2700 CURSOR( 4,1)=&H7FFE        ' Binary 0111111111111110
2800 CURSOR( 5,1)=&H0180        ' Binary 0000000110000000
2900 CURSOR( 6,1)=&H0180        ' Binary 0000000110000000
3000 CURSOR( 7,1)=&H0180        ' Binary 0000000110000000
3100 CURSOR( 8,1)=&H0000        ' Binary 0000000000000000
3200 CURSOR( 9,1)=&H0000        ' Binary 0000000000000000
3300 CURSOR(10,1)=&H0000        ' Binary 0000000000000000
3400 CURSOR(11,1)=&H0000        ' Binary 0000000000000000
3500 CURSOR(12,1)=&H0000        ' Binary 0000000000000000
3600 CURSOR(13,1)=&H0000        ' Binary 0000000000000000
3700 CURSOR(14,1)=&H0000        ' Binary 0000000000000000
3800 CURSOR(15,1)=&H0000        ' Binary 0000000000000000
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 7 ' horizontal hot spot
4400 M3 = 4 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Hourglass

The hourglass is a solid figure with the hot spot at the center of the glass. This shape can be used to show that the operation in progress will take some time to complete.

```
100 '
200 ' Define the screen mask
300 '
400 '
500 CURSOR( 1,0)=&H0000        ' Binary 0000000000000000
600 CURSOR( 2,0)=&H0000        ' Binary 0000000000000000
700 CURSOR( 3,0)=&H0000        ' Binary 0000000000000000
800 CURSOR( 4,0)=&H8001        ' Binary 1000000000000001
900 CURSOR( 5,0)=&HC003        ' Binary 1100000000000011
1000 CURSOR( 6,0)=&HE007       ' Binary 1110000000000111
1100 CURSOR( 7,0)=&HF00F       ' Binary 1111000000001111
1200 CURSOR( 8,0)=&HE007       ' Binary 1110000000000111
1300 CURSOR( 9,0)=&HC003       ' Binary 1100000000000011
1400 CURSOR(10,0)=&H8001       ' Binary 1000000000000001
1500 CURSOR(11,0)=&H0000       ' Binary 0000000000000000
1600 CURSOR(12,0)=&H0000       ' Binary 0000000000000000
1700 CURSOR(13,0)=&H0000       ' Binary 0000000000000000
1800 CURSOR(14,0)=&H0000       ' Binary 0000000000000000
1900 CURSOR(15,0)=&HFFFF       ' Binary 1111111111111111
```

```
2000 '
2100 ' Define the cursor mask
2200 '
2300 CURSOR( 0,1)=&H0000        ' Binary 0000000000000000
2400 CURSOR( 1,1)=&H7FFE        ' Binary 0111111111111110
2500 CURSOR( 2,1)=&H6006        ' Binary 0110000000000110
2600 CURSOR( 3,1)=&H300C        ' Binary 0011000000001100
2700 CURSOR( 4,1)=&H1818        ' Binary 0001100000011000
2800 CURSOR( 5,1)=&H0C30        ' Binary 0000110000110000
2900 CURSOR( 6,1)=&H0660        ' Binary 0000011001100000
3000 CURSOR( 7,1)=&H03C0        ' Binary 0000001111000000
3100 CURSOR( 8,1)=&H0660        ' Binary 0000011001100000
3200 CURSOR( 9,1)=&H0C30        ' Binary 0000110000110000
3300 CURSOR(10,1)=&H1998        ' Binary 0001100110011000
3400 CURSOR(11,1)=&H33CC        ' Binary 0011001111001100
3500 CURSOR(12,1)=&H67E6        ' Binary 0110011111100110
3600 CURSOR(13,1)=&H7FFE        ' Binary 0111111111111110
3700 CURSOR(14,1)=&H0000        ' Binary 0000000000000000
3800 CURSOR(15,1)=&H0000        ' Binary 0000000000000000
3900 '
4000 ' Set the mouse cursor shape, color, and hot spot
4100 '
4200 M1 = 9
4300 M2 = 7 ' horizontal hot spot
4400 M3 = 7 ' vertical hot spot
4500 CALL MOUSE(M1,M2,M3,CURSOR(0,0))
```

Index