CAT. NO. 26-5144

TANDY
TRS-80
COMPUTER PRODUCTS

Model 2000

DIGI-Mouse/Clock Controller Board

Installation and Use

READ ME FIRST!

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG AVENUE MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

PRINTED IN U.S.A.

874-9523 10/84-SP

NOTICE

Disregard information related to installation of the LITHIUM BATTERY, as it has already been installed at the factory.

TERMS AND CONDITIONS OF SALE AND LICENSE OF RADIO SHACK AND TANDY
COMPUTER EQUIPMENT AND SOFTWARE PURCHASED FROM A RADIO SHACK
COMPANY-OWNED COMPUTER CENTER, RETAIL STORE OR FROM A RADIO SHACK
FRANCHISEE OR DEALER AT ITS AUTHORIZED LOCATION

LIMITED WARRANTY

I. CUSTOMER OBLIGATIONS

A CUSTOMER assumes full responsibility that this computer hardware purchased (the "Equipment"), and any copies of software included with the Equipment or licensed separately (the "Software") meets the specifications, capacity, capabilities, versatility, and other requirements of CUSTOMER.

B CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation.

II. RADIO SHACK LIMITED WARRANTIES AND CONDITIONS OF SALE

A For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment, RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects. **This warranty is only applicable to purchases of Radio Shack and Tandy Equipment by the original customer from Radio Shack company-owned computer centers, retail stores and from Radio Shack franchisees and dealers at its authorized location.** The warranty is void if the Equipment's case or cabinet has been opened, or if the Equipment or Software has been subjected to improper or abnormal use. If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer for repair, along with a copy of the sales document or lease agreement. The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair, replacement, or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items.

B RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software, except as provided in this paragraph. Software is licensed on an "AS IS" basis, without warranty. The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software. The defective Software shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, participating Radio Shack franchisee or Radio Shack dealer along with the sales document.

C Except as provided herein no employee, agent, franchisee, dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK.

D **Except as provided herein, Radio Shack makes no express warranties, and any implied warranty of merchantability or fitness for a particular purpose is limited in its duration to the duration of the written limited warranties set forth herein.**

E Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.

III. LIMITATION OF LIABILITY

A **Except as provided herein, Radio Shack shall have no liability or responsibility to customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by "Equipment" or "Software" sold, leased, licensed or furnished by Radio Shack, including, but not limited to, any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of the "Equipment" or "Software". In no event shall Radio Shack be liable for loss of profits, or any indirect, special, or consequential damages arising out of any breach of this warranty or in any manner arising out of or connected with the sale, lease, license, use or anticipated use of the "Equipment" or "Software".**

Notwithstanding the above limitations and warranties, Radio Shack's liability hereunder for damages incurred by customer or others shall not exceed the amount paid by customer for the particular "Equipment" or "Software" involved.

B RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software.

C No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software, whichever first occurs.

D Some states do not allow the limitation or exclusion of incidental or consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.

IV. RADIO SHACK SOFTWARE LICENSE

RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the RADIO SHACK Software on **one** computer, subject to the following provisions:

A Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.

B Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER, but not title to the Software.

C CUSTOMER may use Software on one host computer and access that Software through one or more terminals if the Software permits this function.

D CUSTOMER shall not use, make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in this Software License. Customer is expressly prohibited from disassembling the Software.

E CUSTOMER is permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation of **one** computer with the Software, but only to the extent the Software allows a backup copy to be made. However, for TRSDOS Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use.

F CUSTOMER may resell or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed. The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER.

G All copyright notices shall be retained on all copies of the Software.

V. APPLICABILITY OF WARRANTY

A The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby RADIO SHACK sells or conveys such Equipment to a third party for lease to CUSTOMER.

B The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK, the author, owner and/or licensor of the Software and any manufacturer of the Equipment sold by RADIO SHACK.

VI. STATE LAW RIGHTS

The warranties granted herein give the **original** CUSTOMER specific legal rights, and the **original** CUSTOMER may have other rights which vary from state to state.

Model 2000
DIGI™-Mouse Clock Controller Board
Installation and Use:
Copyright 1984 Tandy Corporation.
All Rights Reserved.

MOUSE.SYS:
Copyright 1984 Tandy Corporation and Microsoft Corporation.
All Rights Reserved.

PIANO.BAS:
Copyright 1984 Microsoft Corporation.
All Rights Reserved.
Licensed to Tandy Corporation.

CLOCKGET.EXE and CLOCKSET.EXE:
Copyright 1984 Tandy Corporation.
All Rights Reserved.

Reproduction or use without express written permission from Tandy Corporation of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the information contained herein.

10 9 8 7 6 5 4 3

Index

About the Mouse/Clock Board

Thank you for purchasing the DIGI-Mouse/Clock Controller Board for the Tandy TRS-80® Model 2000. The dual-purpose Mouse/Clock Board includes a controller for the DIGI-Mouse™ hand-held pointing device (Cat. No. 26-1197) and a real-time/date clock.

Designed to be used with a variety of screen-oriented programs, the DIGI-Mouse frees you from having to use the keyboard to move the cursor and to select commands. By sliding the DIGI-Mouse across a desk top, 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 to be performed.

Note: When using the mouse, place a piece of paper between it and the desk top. Doing this keeps the mouse clean and helps to avoid marring the desk top.

```
2000'
2100' Define 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 0000011001100000
2900 CURSOR( 6,1)=&H0660          'Binary 0000001111000000
3000 CURSOR( 7,1)=&H03C0          'Binary 0000011001100000
3100 CURSOR( 8,1)=&H0660          'Binary 0000110000110000
3200 CURSOR( 9,1)=&H0C30          'Binary 0001100110011000
3300 CURSOR(10,1)=&H1998          'Binary 0011001111001100
3400 CURSOR(11,1)=&H33CC          'Binary 0110011111100110
3500 CURSOR(12,1)=&H67E6          'Binary 0111111111111110
3600 CURSOR(13,1)=&H7FFE          'Binary 0000000000000000
3700 CURSOR(14,1)=&H0000          'Binary 0000000000000000
3800 CURSOR(15,1)=&H0000
3900'
4000' Define 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))
```

Hourglass

The hourglass is a solid figure with the hot spot at the center of the glass. You can use this shape to show that the operation in progress takes some time to complete.

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

About this Manual

Chapter 1 of this manual discusses installation of the hardware. **Before installing the Mouse/Clock Board and connecting the mouse, read the entire chapter carefully.**

Chapters 2 through 5 discuss installation of the software and use of the clock features. It is critical that you read all of these chapters before attempting to use the mouse or clock. This will take very little time, as the chapters are short.

Chapter 6 discusses use of the mouse. To get you started, it includes a sample program called ''Piano.''

Chapter 7 discusses the purpose of each available mouse function call. It is for programmers who want to write programs to use the mouse.

Appendix A is a listing of the Piano program.

Appendix B shows how to create eight different cursor shapes for use in your own programs.

Note: For your convenience, this manual uses ***italicized type*** to represent variable information that you must supply.

```
2000'
2100' Define 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' Define 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
```

Contents

```
2000'
2100' Define 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)=&H1C38          'Binary 0000001111000000
2800 CURSOR( 5,1)=&H700E          'Binary 0000011001100000
2900 CURSOR( 6,1)=&H0000          'Binary 0001110000111000
3000 CURSOR( 7,1)=&H0000          'Binary 0111000000011100
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' Define 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))
```

Diagonal Cross

The diagonal cross is a solid figure with the hot spot at the center of the cross. This 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
```

Chapter 1

Installing the Hardware

Be sure you have the following equipment:

- DIGI-Mouse/Clock Controller Board (Cat. No. 26-5144)
- DIGI-Mouse (Cat. No. 26-1197)
- Three-volt Lithium Coin Cell Battery (Part No. 23-163)

Before installing the Mouse/Clock Controller Board and the DIGI-Mouse, read through the instructions below completely, noting all warnings about equipment handling. Then install the equipment, following the instructions exactly.

1. **Warning: Turn off and then disconnect all equipment.** If it is on, you could damage the central processing unit, as well as the controller board.
2. To avoid possible static charge buildup, touch a metal object to ground yourself before you begin.
3. Slide the Lithium Coin Cell battery under the clip on the board, plus (+) side up, and snap it into place.

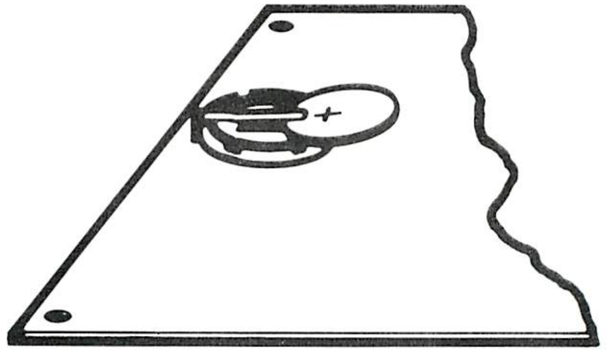

Figure 1

Note: If you install the battery upside-down, it does not make contact.

4. You can install the Mouse/Clock Board in any slot. If, however, you have a Monochrome Graphics Option Board (Cat. No. 26-5140), do not install the memory

board in the lowest slot. The graphics board must go there.

After selecting a slot, "unlock" the panel covering it. Do this by pulling the buttons on either side of the panel. (See Figure 2. Do not try to remove the buttons.) Remove the panel.

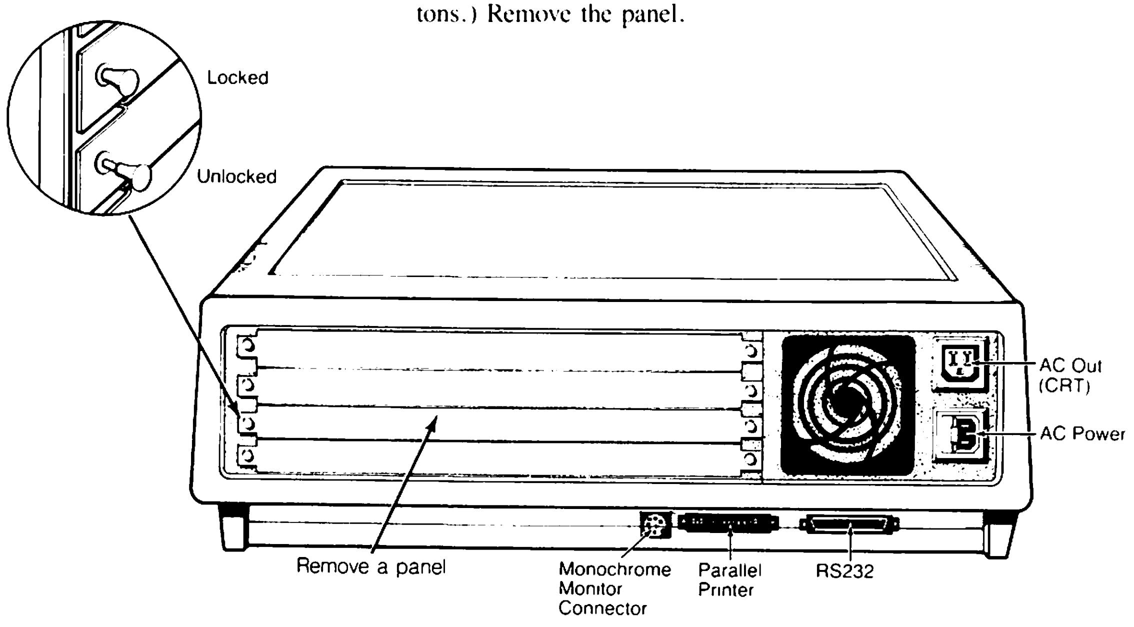

Figure 2

5. Unlock the buttons on the Mouse/Clock Board's panel. (Note: To avoid contaminating the board with oil from your hands, handle the pan containing the board, **not** the board itself.) Slowly slide the pan between the guides along the sides of the card slot until the board meets the connector. (See Figure 3.)

```
2000'
2100' Define 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' Defne 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))
```

Pointing Hand

The upward-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 selecting items from the screen, especially if the items are represented by symbols such as the keys of a piano.

```
 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
```

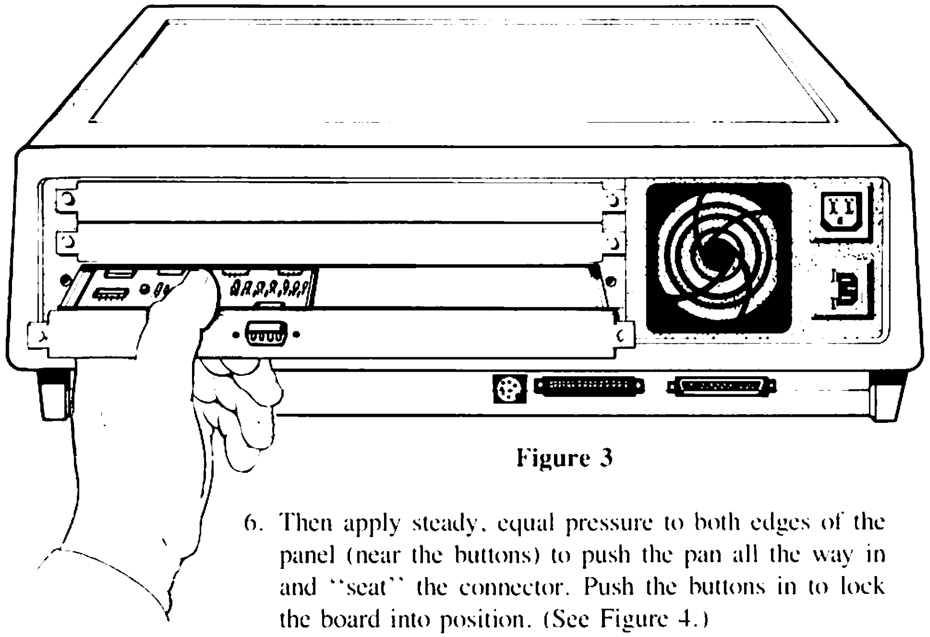
Figure 3

6. Then apply steady, equal pressure to both edges of the panel (near the buttons) to push the pan all the way in and "seat" the connector. Push the buttons in to lock the board into position. (See Figure 4.)

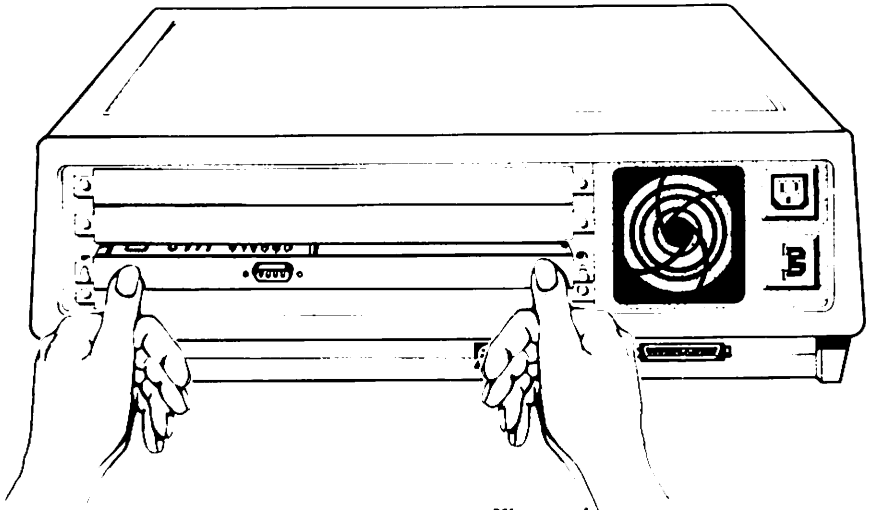
Figure 4

7. Connect the cable from the DIGI-Mouse to the 9-pin connector on the Mouse/Clock Board.

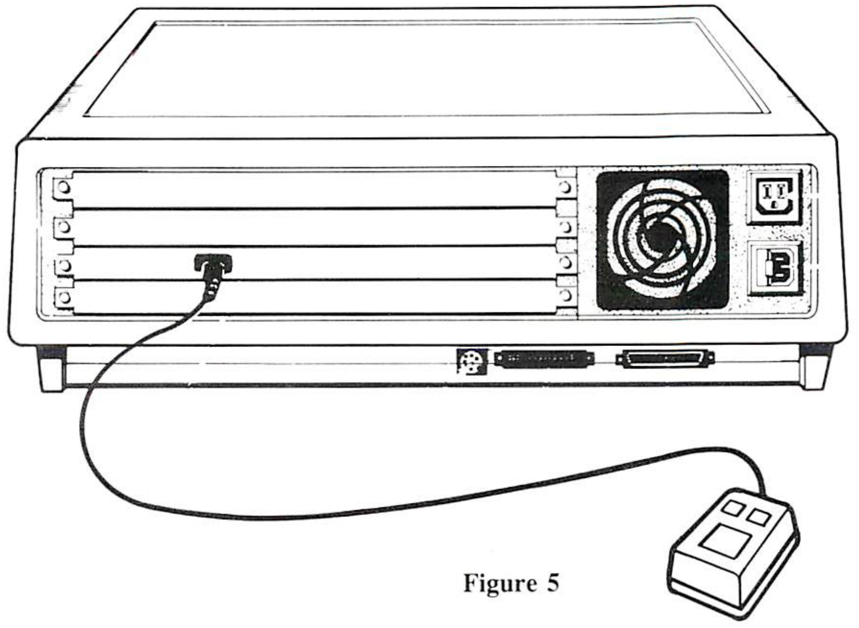

Figure 5

8. Reconnect all cables.

Note: If the clock fails to retain the time when the power is off, the battery is drained. Use a pointed instrument, such as a small screwdriver or an awl, to pry the battery up and out from under the clip. Replace it with another three-volt Lithium Coin Cell Battery.

```
2000'
2100' Define 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
3900'
4000' Define 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))
```

Check Mark

The check mark is a solid figure with the hot spot in the center of the ''V'' formed by the check. It's useful 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
```

Chapter 2

Learning about the Software

The Mouse/Clock Software consists of five files, of which one is optional. Three are provided with the mouse. You provide the other two. Those that are provided on DIGI-Mouse/Clock Diskette are:

- **MOUSE.SYS,** the loadable mouse driver
- **CLOCKSET.EXE,** which sets the clock
- **CLOCKGET.EXE,** which reads the clock

The file **CONFIG.SYS** is a ''configuration'' file — a file that gives the hardware extra information it needs to execute a particular piece of software. In this case, CONFIG.SYS must tell the computer to use the mouse device.

It may be, however, that you have an application program that already has its own CONFIG.SYS file. Because of this, we do not provide CONFIG.SYS on the DIGI-Mouse/Clock Diskette. Instead, we describe how to add the mouse information to an existing CONFIG.SYS or how to create CONFIG.SYS if it does not exist. The procedures, which are given in Chapter 5, take very little time.

The file **AUTOEXEC.BAT** is an optional file that causes your system to read the real-time clock automatically upon each startup. It is not provided on diskette for the same reason that CONFIG.SYS is not: If you have no other file named AUTOEXEC.BAT, you can create the file; if you do have one, you can add to it. See Chapter 6 for instructions.

```
2000'
2100' Define 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)=&H00C0          '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' Define 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))
```

Left Arrow

The left-pointing arrow is a solid 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, reverse the binary bit pattern for each array element and move the hot spot to the new tip. For example, the first element, Binary 1111111000011111 (&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
```

Chapter 3

Copying the Mouse/Clock Software

The procedure for copying the mouse/clock software varies, depending on the kind of system you have. Follow the appropriate steps below.

Floppy Diskette Users

Follow these steps once for each system diskette with which you want to use the mouse and clock.

Note: A system diskette is any diskette that contains the MS-DOS operating system. All Tandy Model 2000 application program diskettes come with the system already on them.

1. Turn on the Model 2000.
2. Insert a backup of the system diskette in Drive A. (First, be sure the diskette's write-protect notch is not covered by a tab.)
3. Insert the DIGI-Mouse/Clock Diskette in Drive B.
4. At the A> prompt, enter these commands:

```
COPY B:MOUSE.SYS A: ENTER
COPY B:CLOCKSET.EXE A: ENTER
COPY B:CLOCKGET.EXE A: ENTER
COPY B:PIANO.BAS A: ENTER
```

Note: The last command copies PIANO.BAS, the demonstration program that you will use in Chapter 7. You need copy this program to Drive A only if you wish to see the demonstration.

The Drive A diskette now contains all mouse/clock files provided. Remove the DIGI-Mouse/Clock Diskette from Drive B and store it in a safe place. Then proceed to the next chapter to learn how to create CONFIG.SYS on the Drive A diskette—or how to add to it, if it already exists—so that you can use the mouse with that diskette whenever you wish.

Hard Disk Users

Follow these steps once to transfer the software to the hard disk, Drive C.

Note: This procedure assumes you have formatted your hard disk and transferred the latest version of the operating system and application programs to it. If you have not done so, see Chapter 6 of *Introduction to the Model 2000* before proceeding.

1. Turn on your Model 2000 and start it up under hard disk control.
2. Insert DIGI-Mouse/Clock Diskette into Drive A.
3. At the C> prompt, enter these commands:

```
COPY A:MOUSE.SYS C: ENTER
COPY A:CLOCKSET.EXE C: ENTER
COPY A:CLOCKGET.EXE C: ENTER
COPY A:PIANO.BAS C: ENTER
```

Note: The last command copies PIANO.BAS, the demonstration program that you will use in Chapter 7. You need copy this only if you wish to see the demonstration.

The hard disk now contains all mouse/clock files provided. Remove the DIGI-Mouse/Clock Diskette from Drive A and store it in a safe place. Then proceed to the next chapter to learn how to create CONFIG.SYS on the hard disk—or how to add to it, if it already exists—so that you can use the mouse with all programs on your hard disk whenever you wish.

```
2000'
2100'  Define 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'  Define 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))
```

Upward Arrow

The upward-pointing arrow is a solid arrow with the hot spot at the tip. It 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
```

Chapter 4

Initializing the Mouse Automatically

To use the mouse and the clock, the current disk must contain a CONFIG.SYS file that has it in this line:

```
DEVICE = MOUSE.SYS
```

If the current disk (Drive C for hard disk users or the diskette currently in Drive A for floppy disk users) already contains CONFIG.SYS, add to the existing file. If it does not, create the file. The procedures for doing both are described below.

Creating CONFIG.SYS

To create CONFIG.SYS on the current disk, type (at the system prompt):

```
COPY CON CONFIG.SYS ENTER
DEVICE = MOUSE.SYS ENTER
CTRL Z ENTER
```

To be sure the file is created and contains the proper command, type:

```
TYPE CONFIG.SYS ENTER
```

MS-DOS displays the commands in the file.

Adding to CONFIG.SYS

To add to an existing CONFIG.SYS file, follow these steps:

1. Type:

   ```
   EDLIN CONFIG.SYS ENTER
   ```

2. MS-DOS displays:

   ```
   End of input file
   *
   ```

   To enter the "insert text" mode, type I ENTER.

   MS-DOS displays the line number and the asterisk prompt:

   ```
   1:*
   ```

3. Type this line:

   DEVICE = MOUSE.SYS (ENTER)

   DEVICE = MOUSE.SYS is the only line required for mouse use. If, however, your application program(s) require other lines not already in the file, insert them too.

4. After typing the line(s), exit the insert mode by pressing (CTRL) (C).

5. Then type **E** (ENTER) to end the file and save all old and new lines.

To be sure the file now contains the proper command(s), type:

TYPE CONFIG.SYS (ENTER)

MS-DOS displays the commands in the file.

After adding to CONFIG.SYS, you must reset the system. Each time you start up from a disk that contains such a CONFIG.SYS and the file MOUSE.SYS, the mouse device driver loads automatically.

Note that the driver requires approximately 3000 bytes of random access memory (RAM). If you have an application program that requires all of RAM and does not require the mouse, delete the line DEVICE = MOUSE.SYS from CONFIG.SYS; then reset the system.

```
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))
```

Standard Cursor

The standard cursor is a solid arrow that points up and to the left. The hot spot is directly 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 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 1111110000111111
1900  CURSOR(15,0) = &HFC3F        'Binary 1111111000011111
```

Chapter 5

Using the System Clock

With the Mouse/Clock Board, you have the option of having the system read the real-time clock automatically upon each startup or reset. This information is used to set the MS-DOS system clock.

Initializing the System Clock Automatically

If you want to use this option, the current disk must contain the CLOCKGET.EXE command in a file called AUTOEXEC.BAT. If the current disk already contains an AUTOEXEC.BAT file, add the command to the existing file. If AUTOEXEC.BAT does not exist, create the file. The procedures for doing both are described below. They are similar to the procedures for creating and adding to CONFIG.SYS.

Creating AUTOEXEC.BAT

To create AUTOEXEC.BAT on the current disk, type (at the system prompt):

```
COPY CON AUTOEXEC.BAT (ENTER)
CLOCKGET.EXE (ENTER)
(CTRL) (Z) (ENTER)
```

Now that AUTOEXEC.BAT contains the CLOCKGET command, the system automatically reads (initializes) the real-time clock upon each startup or reset. Although you need to execute the command only once after each reboot, you can execute it at any time by typing CLOCKGET (ENTER) at the system prompt.

Adding to AUTOEXEC.BAT

To add to an existing AUTOEXEC.BAT file, follow these steps:

1. Type:

   ```
   EDLIN AUTOEXEC.BAT (ENTER)
   ```

2. MS-DOS displays:

   ```
   End of input file
   *
   ```

   Type I (ENTER) to enter the insert mode. Then type the CLOCKGET command line so that your screen looks like this:

   ```
   1:* CLOCKGET.EXE (ENTER)
   ```

   This is the only line required for this real-time clock function. If, however, your application program(s) require other auto commands not already in the file, insert them too.

3. When finished typing the line(s), exit the insert mode by pressing (CTRL) (C).

4. Then type E (ENTER) to end the file and save all old and new lines.

   Note: For more information about batch files such as AUTOEXEC.BAT, see *MS-DOS Commands Reference*.

Setting the System Clock

You may set the real-time clock on the DIGI-Mouse/Clock Board at any time by running CLOCKSET.

To do so, type CLOCKSET (ENTER) at the system prompt.

This causes the current system date and time to be written to the DIGI-Mouse/Clock Board. (See the DATE and TIME commands in *MS-DOS Commands Reference*.) Note that the clock is accurate only to the nearest minute.

The clock cannot save the year; so CLOCKSET writes a file called YEAR.DAT that maintains the year. Whenever you run CLOCKGET, YEAR.DAT should be in the default (current) directory so that CLOCKGET can read it. We suggest, therefore, that you always run CLOCKGET and CLOCKSET from the root directory.

Appendix B
Sample Cursors

This appendix describes eight sample graphics cursors. These sample cursors illustrate the wide variety of cursor shapes that you can define 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 typically suggests the type of action you can 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 in a way that makes their storage order identical to their storage order in a BASIC program.

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

Chapter 6

Using the Mouse

It takes only a few minutes to learn to use the DIGI-Mouse. To help you get started, this chapter includes a demonstration run of Piano, the sample program provided on the DIGI/Mouse Clock Diskette.

Mouse Anatomy

Before using the mouse, examine its working parts.

The buttons permit you to make selections when an application program presents you with a choice. When you press and release a button, the mouse passes this information to the program. A button's definition depends on the current program's definition of it.

When you slide the mouse across a hard, flat surface, the ball on the bottom of the mouse rolls in its socket. The mouse translates this rolling into directional data and passes it to the mouse software to move the cursor on the screen.

Mouse Surface Requirements

Use the mouse on any flat, hard surface, such as a desk. We recommend placing the mouse right beside the keyboard because most programs that use the mouse require a combination of mouse and keyboard input.

The mouse depends on free movement in all directions; so be sure there is adequate space for uninterrupted movement of the mouse and your arm. For most programs, a clear space of 10 by 10 inches is sufficient.

For the best performance, be sure that the surface is free of dirt, moisture, and lint. A sticky surface can prevent the ball from rolling freely. A wet surface can lead to a short in the internal circuitry, damaging the mouse.

Note that some accumulation of dirt and lint is unavoidable. See Chapter 4 of the *DIGI-Mouse Operation Manual* for cleaning instructions. Be sure to turn off your computer and disconnect the mouse before cleaning it.

Moving the Mouse

The mouse lets you move the cursor up, down, left, right and—unlike the keyboard—even diagonally on the screen. See Chapter 3 of the *DIGI-Mouse Operation Manual* for an illustration of how to use the mouse to control cursor movement.

Note that the cursor moves only when the mouse moves. The location of the mouse does not matter. Thus, 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.

Using the Mouse with Piano

If your computer is not already on, turn it on.

Floppy Diskette Users: Insert the system diskette that contains PIANO.BAS into Drive A. (This diskette also contains the Model 2000 BASIC Interpreter, which you need to run the program.)

Hard Disk Users: Be sure Drive C contains the BASIC Interpreter, as well as the operating system and PIANO.BAS, and that the system is operating under hard disk control.

Type:

```
BASIC PIANO (ENTER)
```

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

Note: To run Piano, you must have a Monochrome Graphics Option Board (Cat. No. 26-5140). In addition, if you do not have the Color Graphics Option Kit (Cat. No. 26-5141), your system displays an Illegal Function Call in 1140 message. If this happens, type:

```
LIST 1140 (ENTER)
```

```
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
```

```
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'     M A I N   L O O P
2280'
2290 M1=3:CALL MOUSE(M1,BT,MX,MY)  'Get mouse location
                                     and button status
2995 MY=MY/2
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 IF SC=1 THEN MX=MX\2
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'
```

Then change LINE 1140 to **SC=2:SCREEN SC**, and run the program again.

Piano lets you create 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.

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 even 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 cannot 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; then 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. The computer plays a middle C 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. Notice that if you move the cursor off the piano keyboard and press the button, no note sounds.

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 the cursor to the right eight "white" keys or leave it 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 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 moving 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 the ending point.

When finished playing Piano, move the cursor to the quit box and press either button. The computer exits Piano and displays the system prompt.

```
1780 CURSOR(12,1)=&H0                'Binary 0000000000000000
1790 CURSOR(13,1)=&H0                'Binary 0000000000000000
1800 CURSOR(14,1)=&H0                'Binary 0000000000000000
1810 CURSOR(15,1)=&H0                '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'
1940'
1950' Initialize keyboard size parameters
1960'
1969 IF SC=1 THEN XH=319 ELSE XH=639
1970 YL=60:WKL=80:BKL=45:KW=SC*15:WKN=21
1980 XL=(XH+1)-KW*WKN:YH=YL+WKL:BKW2=KW\3
1990 QX=XH-41*SC: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)-(XH,199),3,B
2200 LOCATE 24,SC*36:PRINT"Quit"
2210'
2220'  Set mouse cursor location, then turn on cursor
```

```
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) = &H300             'Binary 0000001100000000
1670 CURSOR( 1,1) = &H300             'Binary 0000001100000000
1680 CURSOR( 2,1) = &HFC0             'Binary 0000111111000000
1690 CURSOR( 3,1) = &HFC0             '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) = &H0               'Binary 0000000000000000
1750 CURSOR( 9,1) = &H0               'Binary 0000000000000000
1760 CURSOR(10,1) = &H0               'Binary 0000000000000000
1770 CURSOR(11,1) = &H0               'Binary 0000000000000000
```

Chapter 7

Programming for the Mouse

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

- The interface between your computer's screen and the 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 16 functions that make up the mouse system calls.
- A sample BASIC program illustrating the use of the mouse functions.

Mouse Interface

This section discusses the interface between the mouse software and the Model 2000. In particular, it describes the way in which the mouse software uses the resources of the Model 2000 to create a cursor on the screen and to control its movement. This section defines:

- The screen modes
- The virtual screen
- The graphics and text cursors
- The buttons
- The mouse unit of distance: the mickey
- The internal cursor flag

Read the following sections carefully before using any of the mouse functions in your application programs.

The Screen Modes

The mouse software works with all graphics and text ''screen modes'' available with the Model 2000. The screen modes you can use depend on the specific kinds of equipment you have. (See ''Introduction to Graphics'' in Chapter 6 of *BASIC Reference*.)

The Virtual Screen

The mouse software operates on the Model 2000 screen as if it were a ''virtual screen'' of 256,000 individual points arranged in a matrix of 640 horizontal by 400 vertical points, as shown here:

```
0,0 ............................. 639,0
0,1 ............................. 639,1
0,2 ............................. 639,2
 .                                  .
 .                                  .
 .                                  .
 .                                  .
0,399 ......................... 639,399
```

Coordinate System of Virtual Screen

Whenever the software refers to the location as a pair of coordinates on the virtual 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 (320,200). Each pair of coordinates defines a point on the virtual screen. The horizontal coordinate is given first.

In high-resolution (640 by 400) graphics mode (SCREEN 3 or 4), 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 (639,399), is permitted.

In 320 by 200 graphics mode, which is obtained through BASIC SCREEN 1, the number of pixels on the screen is half that 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.

```
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(8,39)
1120 KEY OFF
1130 PLAY"MF"
1140 SC=1:SCREEN SC
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 AND MOUSE THEN 1370
1330 PRINT"Mouse driver not found"          'Not found so print
                                             error.
1340 PRINT
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
```

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.

In 80-column text mode, only characters are permitted on the screen. There are still 256,000 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 16-pixel group, the horizontal coordinate is a multiple of 8 and the vertical coordinate is a multiple of 16.

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 the number of pixels is half that in 80-column text mode. To compensate, the mouse software uses horizontal coordinates that are multiples of 16. 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).

Many mouse functions take coordinates as input or return coordinates as output. Whenever you make a reference to a pixel or character in a function, be sure that the horizontal and vertical coordinates are correct values for the given screen mode. If you supply an incorrect value, the function rounds 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 two different cursors — a graphics cursor and a text cursor. The graphics cursor is a shape (for example, an arrow) that moves over the images on the screen. The text cursor is the normal system cursor. Only one cursor can be on the screen at any given time. You can choose which cursor is on the screen, and you can even switch back and forth between cursors.

Function 9 of the mouse system calls permits you to define the characteristics of the graphics cursor. You may define the characteristics yourself or use the characteristics of the sample cursors listed in Appendix B. The following subsections describe the cursors in detail.

The Graphics Cursor

The graphics cursor is the cursor used when the computer is in graphics mode. It is a block of 256 pixels in a 16- 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.

Note: Some modes, such as BASIC SCREEN 1, are created by software emulation. There is a difference between a logical pixel and an actual screen pixel. The mouse cursor is always defined in terms of actual pixels.

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. Each bit in a mask corresponds to one pixel in the cursor block. 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 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 1 shows how these operations affect the individual screen bits.

Appendix A

Piano Program Listing

This appendix presents the complete source to the Piano Demonstration program. The program is written in BASIC for the Model 2000'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	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 an upward-pointing arrowhead. Its color is different from 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.
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.

Function Code 15

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

Entry Conditions

M1% = 15
M3% = *horizontal mickey/pixel ratio*
    Default = 8/8
M4% = *vertical mickey/pixel ratio*
    Default = 8/8

In the default setting, 6.4 inches of mouse travel moves the cursor across the screen horizontally, and 4.0 inches of travel moves it vertically.

Exit Conditions

None

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%)
```

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

Table 1

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.

The actual color of the cursor is determined by the screen palette. Experienced programmers should know that the cursor is written only to Plane 0.

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 give the cursor any shape 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. The coordinates, which must be within the range -16 to 16, are relative to the upper left corner of the cursor block.

The text cursor is actually the 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 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 9) or after a counter's contents are read (Functions 5 and 6).

The Mouse Unit of Distance: The Mickey

The mouse hardware translates the motion of the ball in the mouse 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" (the number of mickeys the mouse ball

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, first load an assembly-language subroutine into memory. (Use the same segment as the BASIC Interpreter.) Then 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 is loaded into memory and that the integer variable SKETCH is 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%)
```

Set User-Defined Subroutine Input Mask

Function Code 12

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

Entry Conditions

M1% = 12
M3% = *call mask*
M4% = *address offset to subroutine*

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.

Exit Conditions

None

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

has rolled horizontally and vertically). 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 at which 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 moves one 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 must 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 a high-level language 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(5*4+2)
30 MOUSE=256*PEEK(51*4+1)+PEEK(51*4)+2
40 IF MSEG AND MOUSE 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%)
```

   MOUSE is the variable containing the entry offset of the mouse software, and M1%, M2%, M3%, and

Read Mouse Motion Counters

Function Code 11

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 Mouse Unit of Distance: The Mickey" in this chapter.)

Entry Conditions

M1% = 11

Exit Conditions

M3% = *horizontal count*
M4% = *vertical count*

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%)
```

```
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))
```

M4% are the names of the integer variables you have chosen for the 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 examle, this statement defines all variables as integers:

```
10 DEFINT A-Z
```

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 are 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. To do so, follow these steps:

1. Write an assembly-language subroutine to call the mouse driver (as explained in the preceding section).
2. Call the routine from the desired high-level language. (For more information, refer to the manual provided with the high-level language.) The assembly-language routine passes arguments between the high-level language routine and the mouse driver.

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 6. We recommend that you read the listing and refer to this chapter for details on the operation of each function.

Function Code 9

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 Graphics Cursor" in this chapter.) 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 the 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 Graphics Cursor Hot Spot" in this chapter.) The values must be within the range − 16 to 16.

Entry Conditions

M1% = 9
M2% = *horizontal cursor hot spot*
M3% = *vertical cursor hot spot*
M4% = *pointer to screen and cursor masks*

Exit Conditions

None

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
```

Set Minimum and Maximum Vertical Positions

Function Code 8

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 Virtual Screen" in this chapter.)

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.

Entry Conditions

M1% = 8
M3% = *minimum position*
M4% = *maximum position*

Exit Conditions

None

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%)
```

The Piano program listing is also in the file PIANO.BAS on the DIGI-Mouse/Clock Diskette. To see how this program runs under the BASIC Interpreter, type:

```
BASIC PIANO (ENTER)
```

Function Descriptions

This section defines the mouse functions in detail. The following is a list of the functions:

Number	Function
0	Mouse Installed Flag and Reset
1	Show Cursor
2	Hide Cursor
3	Get Mouse Position and Button Status
4	Get 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	Reserved
11	Read Mouse Motion Counters
12	Set User-Defined Subroutine Input Mask
13	Reserved
14	Reserved
15	Set Mickey/Pixel Ratio

Each description specifies the parameters required to make the call (entry conditions) and the expected return values (exit conditions), 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 entry condition for a parameter, you need not supply a value before making the call. If the function description does not spec-

Function Code 7

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 Virtual Screen" in this chapter.)

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.

Entry Conditions

M1% = 7
M3% = *minimum position*
M4% = *maximum position*

Exit Conditions

None

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%)
```

Get Button Release Information

Function Code 6

Returns the 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.

Entry Conditions

```
M1%  =  6
M2%  =  button checked
          M2%  =  0: left button
          M2%  =  1: right button
```

Exit Conditions

M1% = *button status*
M2% = *count of button releases*
M3% = *horizontal position at last press*
M4% = *vertical position at last press*

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."
```

ify an exit condition value for a parameter, the parameter's value after the call is the same as before the call.

Caution: The mouse software does not check entry condition 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 Code Ø

Returns the current installation status of the mouse hardware and software.

Entry Conditions

M1% = Ø

Exit Conditions

```
M1% = mouse status
      M1% =   Ø: not installed
      M1% =  -1: installed
M2% = number of buttons (always 2)
```

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)
User-defined call mask	All zeroes
Horizontal mickey to pixel ratio	8 to 8
Vertical mickey to pixel ratio	16 to 8
Horizontal min./max. cursor position	Ø/639
Vertical min./max. cursor position	Ø/299

Example

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

```
ØØØ '
1ØØ ' Is Mouse present? If not, error.
2ØØ '
3ØØ M1% = Ø
4ØØ CALL MOUSE (M1%,M2%,M3%,M4%)
5ØØ IF NOT(M1%)THEN PRINT "Mouse not
    installed.":END
```

Function Code 5

Returns the 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.

Entry Conditions

```
M1% = 5
M2% = button checked
      M2% = Ø: left button
      M2% = 1: right button
```

Exit Conditions

M1% = *button status*
M2% = *count of button presses*
M3% = *horizontal position at last press*
M4% = *vertical position at last press*

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

The *count of button presses* is always in the range Ø to 32767; overflow is not detected. The count is set to Ø 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

```
1ØØ '
2ØØ ' Get cursor position at last button press.
3ØØ '
4ØØ M1% = 5
5ØØ M2% = Ø  'left button
6ØØ CALL MOUSE (M1%,M2%,M3%,M4%)
7ØØ IF (M1% AND 1) THEN PRINT "Left button
    down."
```

Set Mouse Cursor Position

Function Code 4

Sets the cursor to the specified horizontal and vertical screen positions.

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 Virtual Screen" in this chapter.)

Entry Conditions

M1% = 4
M3% = *new horizontal cursor position*
M4% = *new vertical cursor position*

The new values must be within the horizontal and vertical ranges of the virtual screen.

Exit Conditions

None

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, use these lines:

```
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%)
```

Show Cursor

Function Code 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.

Entry Conditions

M1% = 1

Exit Conditions

None

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

Example

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

Hide Cursor

Function Code 2

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

Entry Conditions

M1% = 2

Exit Conditions

None

Use this function before modifying any position 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 requires a subsequent call to Function 1 to restore the flag to its previous value. (See "The Internal Cursor Flag" in this chapter.)

Example

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

Get Mouse Position and Button Status

Function Code 3

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.

Entry Conditions

M1% = 3

Exit Conditions

M2% = *button status*
M3% = *horizontal cursor position*
M4% = *vertical cursor position*

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

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."
```