Logitech C7 Mouse (260-1199)   Programming   Faxback Doc. # 5445

USING LOGIMENU

LOGIMENU lets you use the mouse with software that was not originally designed for mouse use.

You can use LOGIMENU to create "pop up menus" that you can call to your screen while you're using your application. A LOGITECH Mouse pop up menu will give you options that you can select with the mouse.

You can also use LOGIMENU to define precisely how the mouse moves and how you want the mouse buttons to function.

The process is easy to follow. We provide several working examples of LOGIMENU files that you can either copy and alter for your own purposes, or use as prototypes for menus that more closely meet your needs.

Once you define a menu (with a word processor or editor), giving it an extension name of .DEF, you compile that file with the NEWMENU Compiler.

This creates a new file with a format that can be recognized by the CLICK Mouse Control system; this file has an extension of .MNU.

MENUS SUPPLIED BY LOGITECH

A number of useful menus are already supplied on the LOGITECH C7 Mouse Standard Package diskette. These are listed in the directory with .DEF and .MNU extensions.

<XXX>.DEF files are additional LOGIMENU programs or menus that need to be compiled before you use them. See the next section for instructions on how to create and compile LOGIMENU programs and menus.

<XXX>.MNU files are menus that have already been compiled. If they are not already in your CLICK.SRC file they can be loaded and run by typing:

```
    MENU<XXX> [ENTER]
```

For example, to load the menu for VP-Planner from Paperback Software, type:

```
    MENU VP [ENTER]
```

You will see a message telling you that the Pop Up Menu is loaded. Now you can proceed to load VP-Planner by typing:

```
    VP [ENTER]
```

NOTE ON MICROSOFT MENU COMPATIBILITY:

Microsoft Menu Source files can be compiled by NEWMENU. Do not use MENU directly with a .MNU file made by Microsoft's MAKEMENU utility.

HOW TO BUILD YOUR OWN MENUS

Creating LOGIMENU programs involves simple programming. LOGIMENU has been designed to make this process simple and straightforward, whether you are a beginner or an experienced programmer.

You can create a LOGIMENU program or menu by writing a small program that tells the mouse what to do. This chapter shows you how to write these simple programs. To create LOGITECH Mouse programs and menus, you'll need to perform the following steps:

1. Write a LOGIMENU program <XXX>.DEF, where <XXX> is the name of the .EXE or .COM file for the application. You can do this with EDLIN or with the nondocument mode of a word processing program like WordStar.

2. Save your <XXX>.DEF file.

3. At the prompt, type:

   ```
   NEWMENU <XXX> [ENTER]
   ```

   NEWMENU compiles your menu so that you can use it.

4. If NEWMENU finds any errors in your program, it will not compile it. If this happens, refer to Appendix E for an explanation of the error message. Return to the <XXX>.DEF file and make the corrections. Then start over at Step 3.

If there are no errors, your menu is compiled and ready to use.

NOTE ON MICROSOFT MENU SOURCE FILES:

When compiling a .DEF file that was intended for a two-button mouse, the both buttons option is replaced by the right button. If you prefer to have it replaced by the middle button, simply add the /lrm option to the NEWMENU command, like this:

```
NEWMENU<XXX> /lrm [ENTER]
```

Once a LOGIMENU program or menu has been compiled, it can be called up by typing:

```
MENU<XXX> [ENTER]
```

where <XXX> is the name of the .EXE or .COM file for the application. See Chapter 4 for instructions on how to modify CLICK so that applications menus will be loaded automatically by the system.

NOTE: For very large Menu Files: The menu driver has a default capacity of 5000 bytes.

To produce a larger .MNU file, you need to load MENU with a larger capacity. To do so, start MENU as indicated in Chapter 2, but add to it "/<n>" where <n> is a number of thousand bytes, up to 20. For Example:

```
MENU /11 [ENTER]
```

reserves 11,000 bytes (11 kb) for the menus to be used.

USING LOGIMENU TUTORIAL

This tutorial shows you how to create six common LOGIMENU programs and menus. It explains the LOGIMENU language and how each program works. Key terms in Logimenu language are underlined the first time they are mentioned, and they are defined in Section 5.4.

Before beginning the tutorial, perform the following steps:

NOTE: Your LOGITECH C7 Mouse Standard Package diskette contains the files for all programs and menus covered in this chapter.

IF YOU ARE USING A DUAL FLOPPY DISK COMPUTER:

Copy NEWMENU.EXE to a blank, formatted diskette. This will be your "practice diskette" for the tutorial.

IF YOU ARE USING ONE FLOPPY DISK AND A HARD DISK:

Create a new working directory. The NEWMENU.EXE file should be in your DOS PATH. Your new directory will be where you will do your practice work throughout the tutorial. If you need more information on directories and PATH specifications, refer to your DOS manual.

THE EXTYPE PROGRAM

EXTYPE programs the LOGITECH Mouse buttons to perform functions like TIME, DATE, and DOS VERsion. To create EXTYPE:

1. Open a file named EXTYPE.DEF.

2. On line 1, type: BEGIN LeftB, MidB, RightB Press [ENTER].

3. Skip a line. Press [ENTER].

4. On line 3, type: LeftB: TYPE "TIME", ENTER, ENTER Press [ENTER].

5. On line 4, type: ; This calls the TIME command without changing the time Press [ENTER].

6. Skip a line. Press [ENTER].

7. On line 6, type: MidB: TYPE "DATE", ENTER, ENTER   Press [ENTER].

8. On line 7, type: ; This calls the DATE command without changing the date. Press [ENTER].

9. Skip a line. Press [ENTER].

10. On line 9, type: RightB: TYPE "VER", ENTER   Press [ENTER].

11. On line 10, type: ; The VER command gives you the DOS version number. Press [ENTER].

12. Your screen should match Figure 5-1. Make corrections, if necessary.

13. Save the file on your practice diskette or in your working directory.

14. To compile the EXTYPE program from your EXTYPE.DEF file, type:

```
    NEWMENU EXTYPE [ENTER]
```

15. If NEWMENU finds any error in your program, it will not compile it. If this happens, refer to Appendix E for an explanation of the error message. Then repeat Steps 12 through 14.

    If NEWMENU finds no errors, it compiles your program. The program is ready to use. Now click each mouse button in turn.

16. Observe the function performed by each mouse button.

USING LOGIMENU

How the EXTYPE Program Works

EXTYPE starts with a BEGIN statement:

```
    BEGIN LeftB, MidB, RightB
```

The BEGIN statement assigns LABELS to mouse buttons:

```
    LeftB (left button)
    MidB (middle button)
    RightB (right button)
```

These same labels are also listed near the left margin of your program.

When you press a mouse button, your program looks for the label that matches it. Then it performs the action listed next to that label.

Suppose you press the left button. The program looks for the label LeftB and performs the action next to it. The action here is a TYPE statement: TYPE "TIME" ENTER, ENTER. This causes the program to type the word TIME and to "press" the [ENTER] key twice. This causes DOS to display the time.

In a TYPE statement, you must put a word in quotation marks if you want it to be typed as a word. Because ENTER is not in quotation marks, the program interprets it as the [ENTER] key.

Three lines of your program start with a semicolon. Those lines are comments. A COMMENT is a note you make to yourself that makes the program clearer when you go back and reread it later. LOGIMENU ignores any information following a semicolon up to the end of the line.

THE EXEBEGIN PROGRAM

EXBEGIN programs the mouse buttons to perform specific functions and determines how mouse movement affects the cursor. Here, you will make the mouse buttons; simulate [ENTER], [CTRL] [C], and [ESC]. Your program will make the cursor move when the mouse moves 50/200" horizontally or 100/200" vertically. To create EXBEGIN:

1. Open a file named EXBEGIN.DEF.

2. On line 1, enter:

```
    BEGIN LeftB, MidB, RightB, LeftB, LeftM, UpM, DownM, 50, 100
```

3. Skip a line.

4. On lines 3 through 5 enter:

```
    LeftB:    TYPE ENTER
    MidB:     TYPE 3    ;3 IS CTRL-C
    RightB:   TYPE ESC
```

5. Skip a line.

6. On lines 7 through 10, enter:

```
    LeftM:    TYPE 0, 75   ; 0, 75 is Left Arrow Key
    RightM:   TYPE 0, 77   ; 0, 77 is Right Arrow Key
    UpM:      TYPE 0, 72   ; 0, 72 is Up Arrow Key
    DownM:    TYPE 0, 80   ; 0, 80 is Down Arrow Key
```

7. Your screen should match Figure 5.2. Make corrections, if necessary.

8. Save the file on your practice diskette or in your working directory.

9. To compile EXBEGIN from your EXBEGIN.DEF file, type:

```
    NEWMENU EXBEGIN [ENTER]
```

10. If NEWMENU finds any error in your program, it will not compile it. If this happens, refer to Appendix E for an explanation of the error message. Then repeat Steps 7 through 9.

    If NEWMENU find no error, your program is compiled and ready to use. Now move the mouse and click each mouse button in turn.

11. See how the mouse controls movement of the cursor on the screen. Observe the function performed by each mouse button.

HOW THE EXBEGIN PROGRAM WORKS

The EXBEGIN program starts with a BEGIN statement.

```
    BEGIN LeftB, MidB, RightB, LeftM, RightM, UpM, DownM, 50, 100
```

EXBEGIN starts by labeling the mouse buttons. The BEGIN statement also labels mouse movements:

    LeftM (left movement),
    RightM (right movement),
    UpM (up movement), and
    DownM (down movement).

Seven labels are listed at the left-hand side of your program. Each click of a mouse button or movement of the mouse causes the program to look for the label the corresponds to that mouse button or that movement.

NOTE: Any label can be used to name a mouse button and its movement, as long as matching labels are listed down the left-hand side of the program.

In a BEGIN statement, LOGIMENU always recognizes these labels:

Item one, left button
Item two, middle button
Item three, right button
Item four, movement left
Item five, movement right
Item six, movement up
Item seven, movement down.

This means that when you press the left button, your program first looks for the label LeftB. Then it performs the action listed next to that label. The EXEBEGIN program executes an [ENTER].

When you move the mouse to the right, a similar chain of events occurs. Your program looks for the label RightM. Then it performs the action listed next to that label: TYPE 0, 77. the numbers 0, 77 are ASCII code for the [RIGHT] arrow cursor key. When the mouse moves to the right, your program executes "right arrow key" and moves to the right. (More ASCII codes are listed in Section 5.5.1).

The eighth and ninth items in the BEGIN statement refer to sensitivity of the mouse, or how far the cursor will move when you move the mouse on the work surface. the eighth item indicates the mouse's horizontal sensitivity. the ninth item indicates the mouse's vertical sensitivity.

In EXBEGIN, the number 50 indicates that when you move the mouse 50/200" horizontally, LOGIMENU registers a move to the left or right.

The number 100 indicates that when you move the mouse 100/200" vertically, LOGIMENU registers a movement up or down.

In other words, if you move the mouse less than 50/200" horizontally or 100/200" horizontally or 100/200" vertically, no action is registered.

NOTE: If items eight and nine are not included in the BEGIN statement, the LOGITECH Mouse will be set to its default values: 4, 8.

THE EXEXECUT PROGRAM

EXEXECUT programs the LOGITECH Mouse buttons. Here, you will make them call for the currently logged directory, and the directories of drives A and B. EXEXECUT demonstrates how to "chain" a sequence of commands, something you may need to do if your program contains long statements. To create EXEXECUT:

1. Create the file EXEXECUT.DEF exactly as it appears in Figure 5-3.

2. Save the file on your practice diskette or in your working directory.

3. To compile the EXEXECUT.DEF program, type:

   ```
   NEWMENU EXEXECUT [ENTER]
   ```

4. If NEWMENU finds errors in your program, it will not compile it. If this happens, refer to Appendix E for an explanation of the error message. The correct the file and repeat Steps 2 and 3.

If NEWMENU finds no errors, then your program is compiles and ready to use. Click each mouse button in turn.

5. Observe the function performed by each button.

```
BEGIN LeftB, MidB, RightB

LeftB:        EXECUTE TDIR, TRETURN
MidB:         EXECUTE TDIR, TSPACE, TA, TRETURN
RightB:       EXECUTE TDIR, TSPACE, TB, TRETURN

TDIR:         TYPE "DIR"
TSPACE:       TYPE " "
TA:           TYPE "A:"
TB:           TYPE "B:"
TRETURN:      TYPE ENTER
```

FIGURE 5-3 THE EXEXECUT PROGRAM

HOW THE EXEXECUT PROGRAM WORKS

EXEXECUT starts with a BEGIN statement that labels the mouse buttons. A mouse button click makes the program look for the appropriate label and then perform the action listed next to it.

In the EXEXECUT program, the action is an EXECUTE statement that sends the program to a combination to two or more additional labels. Here, the labels are: TDIR, TSPACE, TA, TB, and TRETURN. Each of these labels triggers a TYPE statement. The combinations create different final messages.

For example, click the middle button. EXEXECUT finds the label MidB and performs the EXECUTE statement next to it. The EXECUTE statement send the program to a sequence of four labels: TDIR, TSPACE, TA and TRETURN. This sequence triggers four TYPE statements that come together as:

```
DIR A: [ENTER].
```

THE EXCHORDS PROGRAM

EXCHORDS sets combinations of mouse buttons to perform certain functions. These combinations (left-middle, left-right, middle-right, and all-buttons) are called chords. Here, each chord causes a line of type identifying the chord to appear on the screen, after which the chord will simulate [CTRL] [C], or cancel. To create the EXCHORDS program:

1. Create the nondocument file EXCHORDS.DEF as it appears in Figure 5-4.

2. Save the file on your practice diskette or in your working directory.

3. To compile EXCHORDS.DEF, type:

```
NEWMENU EXCHORDS [ENTER]
```

4. When NEWMENU finds no errors in EXCHORDS.DEF, it is compiled and ready to use.

5. Observe the function performed by each combination of mouse buttons.

```
; 3 stands for CTRL-C, canceling the input

LMB:          TYPE "Left and Middle Buttons Pressed", 3
LRB:          TYPE "Left and Right Buttons Pressed", 3
MRB:          TYPE "Middle and Right Buttons Pressed", 3
ALLB:         TYPE "All Three Buttons Pressed", 3
```

FIGURE 5-4 THE EXCHORDS PROGRAM

HOW THE EXCHORDS PROGRAM WORKS

The EXCHORDS program starts with an empty BEGIN statement and a CHORDS statement that labels the mouse button chords:

    LMB (left and middle button),
    LRB (left and right button),
    MRB (middle and right button), and
    ALLB (all buttons).

When you click one of these combinations, EXCHORDS looks in the program for the appropriate label and then performs the action next to it.

Suppose you press the left and middle button simultaneously. the program finds the label LMB and performs the action next to it.

Suppose you press the left and middle button simultaneously. The program finds the label LMB and performs the TYPE statement next to it:

```
TYPE "LEFT AND MIDDLE BUTTONS PRESSED", 3
```

This types

```
LEFT AND MIDDLE BUTTONS PRESSED
```

on the screen.

Then it types the ASCII code

```
3
```

which causes the program to execute [CTRL] [C], or Cancel.

In this sample program, the final effect of each chord is to execute a [CTRL] [C]. You can easily modify the program by substituting other ASCII codes from the table in Section 5.5.1.

THE EXMENU PROGRAM

EXMENU; creates a menu on your screen and then programs mouse buttons and mouse button chords to perform certain functions. Here, a click of any mouse button or chord causes the menu to be displayed. By moving the mouse, you can highlight any of the options in the menu. A click of any mouse button selects the highlighted option, while a chord cancels the menu. To create the EXMENU program:

1. Create EXMENU.DEF as it appears in Figure 5-5.

2. Save the file on your practice diskette or in your working directory.

3. Compile the EXMENU.DEF program by typing:

```
NEWMENU EXMENU [ENTER]
```

4. When NEWENU finds no errors in EXMENU.DEF, it is compiled and ready to use. Click any mouse button or mouse button chord to display the menu and click any chord to remove the menu from the screen.

5. With the menu on the screen, move the mouse to highlight your selection. Click any mouse button to select a highlighted menu option.

```
BEGIN menu1, menu1, menu1
CHORDS menu1, menu1, menu1 menu1

menu1: MENU "title", 8, 30, BOLD
        OPTION "DIR - LIST DIRECTORY", dir
        OPTION "PRINT file(s)", print
        OPTION "CHKDSK - check disk", chkdsk
        OPTION "RENAME file", rename
        MEND

dir:    TYPE "Dir", ENTER
print:  TYPE "PRINT"
chkdsk: TYPE "CHKDSK", ENTER
rename: TYPE "RENAME"
```

FIGURE 5-5 THE EXMENU PROGRAM

HOW THE EXMENU PROGRAM WORKS

EXMENU starts with a BEGIN statement:

```
BEGIN menu1, menu1, menu1
```

This makes each mouse button call for label menu1.

This CHORDS statement,

```
CHORDS menu1, menu1, menu1, menu1
```

Also programs each chord, or mouse button combination, to call the label menu1.

The line next to the label menu 1 defines the menu.

```
    menu1: MENU "Title", 8, 30, BOLD
```

Any title in quotation marks appears on the screen as the title of the menu. The first number, 8, defines the line number at which the menu appears. The second number, 30, defines the column number at which the menu appears. The word BOLD defines the character properties of the menu. The type is bold-faced. You could also choose NORMAL or INVERSE, or a number to indicate a color code (refer to your graphics adapter manual).

The next four lines are OPTION statements. The words in quotation marks appear as options in the menu: DIR - LIST DIRECTORY, PRINT file(s), CHKDSK, and RENAME file. The word at the end of each line is the label for that option. MEND -- the MEND statement -- ends the menu definition.

EXMENU works like this: When you click a mouse button to select an option, you also select a label. The program goes to that label and performs the action next to it.

Suppose, for example, that you select CHKDSK -- check disk. This selects the label chkdsk. The program finds this label and performs the action next to it, ie. it executes CHKDSK [ENTER].

THE EXPOPUP PROGRAM

EXPOPUP creates a pop-up menu on your screen. It is similar to EXMENU, but allows you much more flexibility in design of the menu. Menu items may be positioned anywhere on the screen, not just one below the other.

EXPOPUP also programs the mouse buttons and chords to perform certain functions. A click calls the menu. A click also selects a highlighted option. To create the EXPOPUP program:

1. Create the file EXPOPUP.DEF exactly as it appears in Figure 5-6.

2. Save the file on your practice diskette or in your working directory.

3. To compile the EXPOPUP.DEF program, type:

```
    NEWMENU EXPOPUP [ENTER]
```

4. When NEWMENU finds no errors in the EXPOPUP.DEF, it is compiled. EXPOPUP.MNU is now generated and ready to use. Click any mouse button or mouse button chord to display the pop-up menu and click any chord to remove the menu from the screen.

5. With the pop-up menu on the screen, move the mouse to highlight an option. Click a mouse button to select the highlighted option.

```
BEGIN popup1, popup1, popup1
CHORDSpopup1, popup1, popup1, popup1

popup1:       POPUP 12, 20, NORMAL
;             WARNING: the following line is a comment
;                     1234567890        1234567890        1234567890
              TEXT "  DIR - LIST DIRECTORY   PRINT file(s)       "
TEXT "  CHKDSK - check disk    RENAME file         "
SELECT  1, 3, 20 dir
SELECT  1, 26, 13 print
SELECT  2, 3, 19 chkdsk
SELECT  2, 26, 11 rename
PEND

dir:          TYPE "DIR", ENTER
print:        TYPE "PRINT"
chkdsk:       TYPE "CHKDSK", ENTER
rename:       TYPE "RENAME"
```

FIGURE 5-6 THE EXPOPUP PROGRAM

HOW THE EXPOPUP PROGRAM WORKS

EXPOPUP starts with a BEGIN statement and a CHORDS statement that causes any click to call the menu with the label popup1.

On line four, the line next to the label popup1 defines the menu you create:

```
    popup1: POPUP 12, 20, NORMAL
```

The numbers 12 and 20 define where the pop-up menu appears -- at line 12, column 20. NORMAL defines the properties of the menu characters as having normal type.

The lines that begin with semicolons are comments. The program does not read these. The numbers help you count characters and spaces so that you can enter other information.

The next two lines of the EXPOPUP program are TEXT statements. The words in quotation marks are the words that appear as options in the menu. they appear in the menu exactly as they are arranged and spaced here. They appear in the menu exactly as they are arranged and spaced here. The quotation marks indicate the outer edge of the menu.

The four SELECT statements are lines that tell the program what to do when you select an option. Each SELECT statement consists of three numbers and a label.

The first number is the menu line on which the option appears. The option DIR - list directory appears on line 1 of the menu, so the number is 1.

The second number is the column of the menu at which the option begins. This is where the numbers you used in the comment line help. The option DIR - list directory begins at column 3, so the number is 3.

The third number is the length of the option, counting both characters and spaces. For DIR - list directory, that number is 20.

PEND -- the PEND statement -- ends the pop-up menu definition.

When you highlight an area of the menu and click a selection, the program matches the parameters of that selection with these numbers.

Selecting DIR - list directory, for example, sends the program to SELECT 1, 3, 20, dir.

The program then goes to the label dir listed at the left and performs the action next to it. In this case, it executes:

```
    DIR [ENTER]
```

This concludes the LOGIMENU tutorial. Useful reference material and two sample menus are contained in the next sections of this chapter.

(dkh-07/30/93)