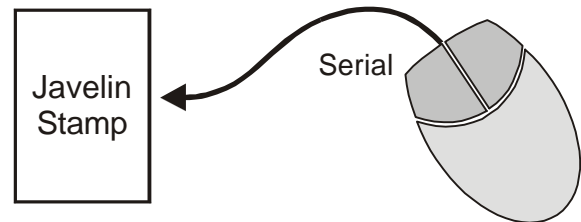




599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office/Tech Support: (916) 624-8333
Fax: (916) 624-8003

Web Site: www.javelinstamp.com
Home Page: www.parallaxinc.com

General: info@parallaxinc.com
Sales: sales@parallaxinc.com
Technical: javelintech@parallaxinc.com



Contents

Introduction to Microsoft's™ 2-Button Serial Mouse	1
Downloads, Parts, and Equipment for a serial mouse	2
The Serial Modem Circuit.....	3
Creating a Mouse Circuit Without A Demo Board	4
Testing the Mouse Circuit.....	5
Program Listing 1.1 – The MS 2 Button Serial Test	8
A Drag and Drop Example.....	9
Program Listing 1.2 – MS2ButtonSerialExample	11
Published Resources – for More Information	14
Javelin Stamp Discussion Forum – Questions and Answers.....	14

Introduction to Microsoft'sä 2-Button Serial Mouse

This application note will demonstrate how to use a Microsoft 2-button serial mouse within your Javelin programs. The Javelin's Demo Board has a built-in COM port connection which makes it very easy to interface a serial mouse to the Javelin.

This class will let you utilize the mouse within your own applications to interact with the *Messages from Javelin* window, various LCDs, and for many other purposes.

This application note includes:

- 1) The mouse library which contains all the methods needed to receive data from the mouse.
- 2) The javadoc for this library which gives you information on all the methods' parameters.
- 3) A test program that will help you verify the unit is working properly.
- 4) A full-featured, interactive mouse example.
- 5) A helpful terminal class that the mouse example uses to interact with the Javelin's IDE message window.

Downloads, Parts, and Equipment for a serial mouse

This application note (AppNote010-MS2ButtonSerial.pdf), the library file (MS2ButtonSerial.java), the library's javadoc file (MS2ButtonSerial.pdf), the test program (MS2ButtonSerialTest.java), and a demonstration program (MS2ButtonSerialExample.java) are all available to you for free download from:

<http://www.javelinstamp.com/Applications.htm>

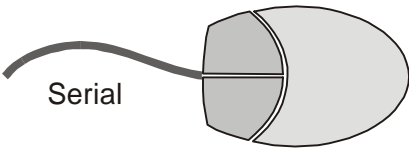
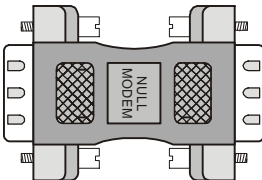
You can use the AppNote010-MS2ButtonSerial.exe, to install the files listed below. These files must be located in specific paths within the Javelin Stamp IDE directory. Although the path to this directory can be different, the default root path is: C:\Program Files\Parallax Inc\Javelin Stamp IDE

The file list below is organized by directory, then by filename, please verify that your file list is organized in the same way.

```
<root path>\doc\AppNote010-MS2ButtonSerial.pdf
<root path>\doc\MS2ButtonSerial.pdf
<root path>\lib\stamp\peripheral\hid\mouse\serial\MS2ButtonSerial.java
<root path>\Projects\examples\peripheral\hid\mouse\serial\MS2ButtonSerialTest.java
<root path>\Projects\examples\peripheral\hid\mouse\serial\MS2ButtonSerialExample.java
<root path>\Projects\examples\peripheral\hid\mouse\serial\MS2ButtonSerialTerminalHelper.java
```

Table 1.1 lists the parts you will need for this application note.

Table 1.1: Parts List

Quantity	Part Ordering Info and Part Description	Schematic Symbol	Pin Map
1	Microsoft Compatible 2-Button Serial Mouse (DB9-F)		
1	Null Modem Adapter (DB9-M to DB9-M) Parallax Part #800-00002		

The equipment used to test this example includes a Javelin Stamp, Javelin Stamp Demo Board, 7.5 V, 1000 mA DC power supply, serial cable, and PC with the Javelin Stamp IDE v2.01.

The Serial Modem Circuit

Configuring the Javelin's Demo board to be used with a mouse is explained below (refer to Figure 1.1). There are only three connections that you will need to make to link the Javelin Stamp to the Demo board's COM Port. These three connections are shown within the magnified circle in Figure 1.1. Once these connections are in place simply plug your *Microsoft compatible 2-button serial mouse* into the *Null modem adapter* and plug the adapter into the Demo board's COM Port. These steps are detailed below:

Tip

This application note is for a *serial* type mouse. These mice should have a 9-pin (DB-9) connector. The Javelin Demo board has two female DB-9 connectors, one is labeled *JIDE Port*, and the other is labeled *COM Port*. The *JIDE Port* connects to your computer for programming the Javelin. The *COM Port* connects to peripherals like a mouse.

- Connect pin 1 of the COM Port to P0 of the Demo board.
- Connect pin 3 of the COM Port to P1 of the Demo board.
- Connect pin 8 of the COM Port to P2 of the Demo board.

- Connect your Microsoft compatible 2-button serial mouse into the null modem adapter. *A null modem cable may be used if available. The mouse should be a DB-9 female, the null modem adapter should be a DB-9 male to DB-9 male. If your adapter or mouse has a gender that is not compatible, you can use an appropriate “straight-through” gender changer.*
- Plug the other end of your null modem adapter into the *COM Port* of the Demo board.

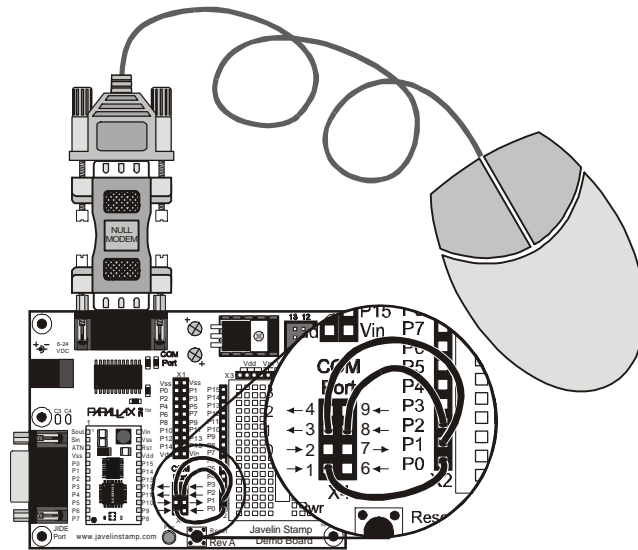


Figure 1.1
Connection diagram for a serial mouse.

Tip

When a peripheral like a mouse is connected to the Demo board, the Demo board takes the roll of a computer for the mouse. When used this way, a *null modem adapter* must be used.

Creating a Mouse Circuit Without A Demo Board

The easiest way to use the mouse is using the built-in COM port on the Javelin's Demo board. If you do not have the Javelin's Demo board, you can build your own COM Port circuit with a SP237, MAX232 or similar chip. Figure 1.2 gives you breakdown of how this connection can be made using an SP237. Since a detailed explanation is beyond the scope of this application note, you will need to refer to your chip's data sheet for more information.

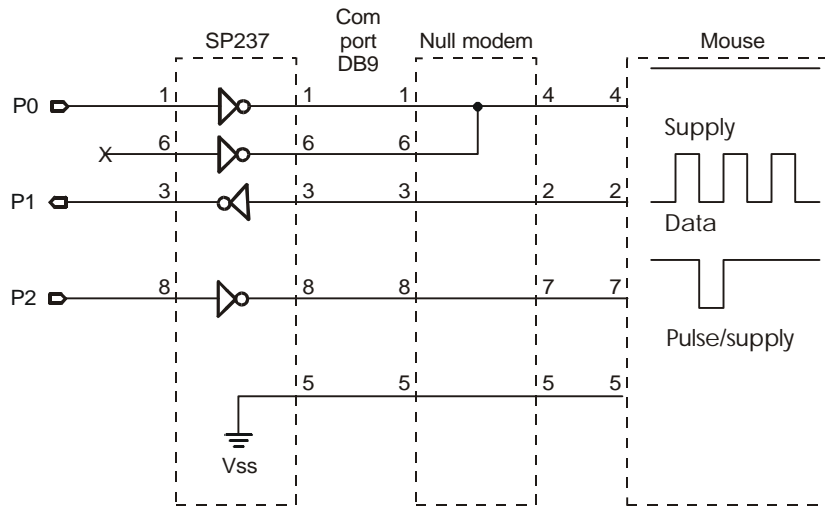


Figure 1.2: Breakdown of the mouse circuit.

Testing the Mouse Circuit

Program Listing 1.1 is a short program that will verify that the circuit in Figure 1.1 is working properly. This program will display the current status of the mouse location and button positions.

The program begins by creating 3 constants, **DTR**, **RX**, and **RTS**, which set the Javelin's I/O pins P0, P1, and P2 to the COM port pins 1, 3 and 8 respectively.

```
final static int DTR = CPU.pin0;           // COM Port 1
final static int RX  = CPU.pin1;           // COM Port 3
final static int RTS = CPU.pin2;           // COM Port 8
```

Next we will be creating a UART which will communicate between the Javelin and the mouse. This UART will be transmitting data to the Javelin without inverting the data, using a 1200 baud rate, with 1 stop bit.

```
static Uart rxUart = new Uart( Uart.dirReceive, RX, Uart.dontInvert,
                               Uart.speed1200, Uart.stop1);
```

Now we can create the mouse object, we do this using the default settings of the constructor.

```
static MS2ButtonSerial mouse = new MS2ButtonSerial(rxUart,DTR,RTS);
```

Next we will create three variables which will store information received from the mouse.

```
static int xDistance, yDistance;      // Store x & y mouse values
static boolean mouseDetect;          // Mouse status
```

Now we enter the main section of the program. We begin by clearing the IDE's output window.

```
System.out.print('\u0010');          // Clear output window.
```

Then we will initialize communication with the mouse, this method will return a TRUE or FALSE and store this value in the **mouseDetect** variable.

```
mouseDetect = mouse.bootSequence();  // Initialize mouse.
```

Now based upon **mouseDetect** we will either enter the **if** statement if a mouse was detected, or drop to the **else** statement if the mouse was not detected. If a mouse was detected, we will print a message stating so, then print a message to the user to move the mouse.

```
System.out.println("Mouse successfully detected.\n");
System.out.println("Move mouse and click buttons to display data.");
```

Here is when we enter a continuous loop where the program will constantly display the status of the mouse.

```
while(true){
```

Next, the program will wait for a mouse event to be stored in the serial buffer.

```
if(mouse.event()){
```

Once an event has been buffered, we will pull this data into the object.

```
mouse.update();
```

This next segment of code will position the cursor to a specific location on the screen, then display the status of each mouse event.

```
System.out.print("\u0001\n\n\n\n\n");

// Display button states.
System.out.print("leftButton    ");
System.out.print(mouse.leftButton);
```

```
System.out.print("      \n");

System.out.print("rightButton  ");
System.out.print(mouse.rightButton);
System.out.print("      \n\n");

// Display distances traveled.
System.out.print("xDistance    ");
System.out.print(mouse.xDistance);
System.out.print("      \n");

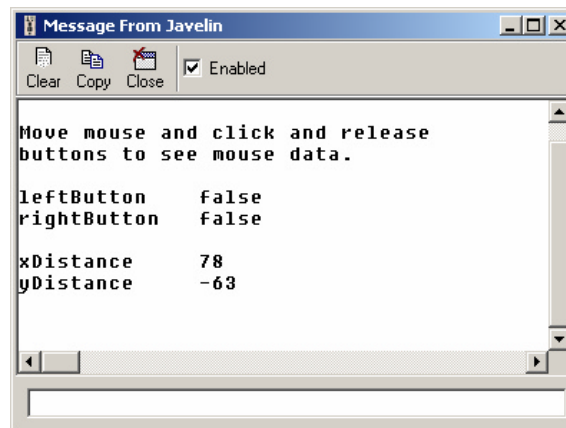
System.out.print("yDistance    ");
System.out.print(mouse.yDistance);
System.out.print("      \n");
```

Finally if the mouse could not be detected, the following segment of code will be executed, and a message will be displayed.

```
else{
    CPU.delay(20000);
    System.out.println("Mouse not detected!");
    System.out.println("Check wiring and try again.");
} // end else
```

The test program (Program Listing 1.1) will display output to the Javelin's IDE message window, similar to Figure 1.3.

Figure 1.3
Output for test
program.



If your output differs from that of Figure 1.3, verify each of your connections from Figure 1.1 are correctly installed.

Program Listing 1.1 – The MS 2 Button Serial Test

```
import stamp.core.*;
import stamp.peripheral.hid.mouse.serial.MS2ButtonSerial;

/* This program will test the mouse for proper configuration
 * by displaying the status of the buttons and the location of the mouse.
 * For more information see application note 10 (an010) from Parallax, Inc.
 *
 * Version 1.0 - 12/27/02
 */
public class MS2ButtonSerialTest {

    // Declare I/O pins connected to Mouse via RS232.
    final static int DTR = CPU.pin0;           // COM Port 1
    final static int RX  = CPU.pin1;           // COM Port 3
    final static int RTS = CPU.pin2;           // COM Port 8

    // Declare receiver Uart for serial mouse connection/data.
    static Uart rxUart = new Uart( Uart.dirReceive, RX, Uart.dontInvert,
                                   Uart.speed1200, Uart.stop1);

    // Create a mouse object with default settings.
    static MS2ButtonSerial mouse = new MS2ButtonSerial(rxUart,DTR,RTS);
    static int xDistance, yDistance;           // Store x & y mouse values
    static boolean mouseDetect;                // Mouse status

    public static void main(){
        System.out.print('\u0010');           // Clear output window.
        mouseDetect = mouse.bootSequence();    // Initialize mouse.

        // If mouse initialization succeeded, display button and distance info.
        if(mouseDetect){
            System.out.println("Mouse successfully detected.\n");
            System.out.println("Move mouse and click buttons to display data.");

            while(true){

                // Wait until the mouse object has captured data before updating display.
                if(mouse.event()){

                    // Move mouse data from serial buffer into the mouse object
                    mouse.update();

                    // Send cursor to home position, then move to sixth line.
                    System.out.print("\u0001\n\n\n\n\n");

                    // Display button states.
```



```
        System.out.print("leftButton    ");
        System.out.print(mouse.leftButton);
        System.out.print("\n");

        System.out.print("rightButton   ");
        System.out.print(mouse.rightButton);
        System.out.print("\n\n");

        // Display distances traveled.
        System.out.print("xDistance    ");
        System.out.print(mouse.xDistance);
        System.out.print("\n");

        System.out.print("yDistance    ");
        System.out.print(mouse.yDistance);
        System.out.print("\n");

    } // end if mouse.event
} // end while
} // end if mouseDetect

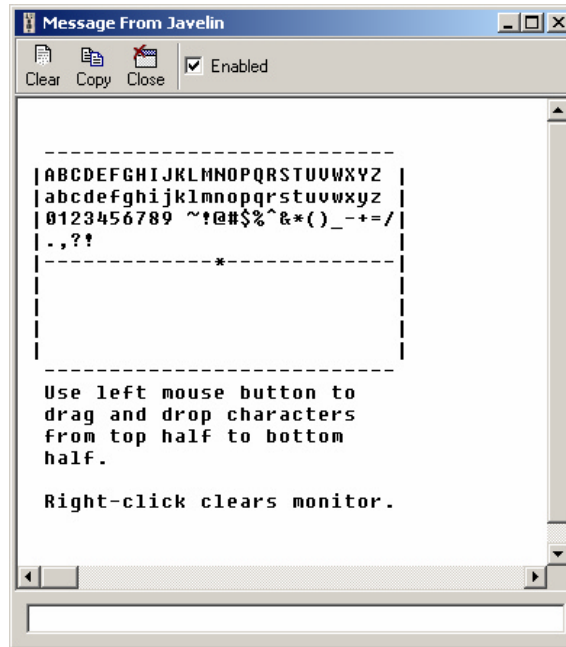
// If mouse initialization did not succeed, display error message.
else{
    CPU.delay(20000);
    System.out.println("Mouse not detected!");
    System.out.println("Check wiring and try again.");
} // end else
} // end main
} // end class
```

A Drag and Drop Example

Program Listing 1.2 allows you to drag and drop characters within a box on the IDE's "Messages from the Javelin" window. The mouse will not move outside of this boxed area and the mouse will pass over the characters present within the box. For this example to function properly all display interactions have been imported from a terminal helper class (MS2ButtonSerialTerminalHelper.java).

This program should only be executed if the test program (Program Listing 1.1) has been successfully executed. The output shown in Figure 1.4 should appear in the Messages from the Javelin window. If your output differs, verify the connections from Figure 1.1.

Figure 1.4
Output for
Program Listing 1.2



Program Listing 1.2 uses the methods below from the **MS2ButtonSerial** library class.

- **bootSequence()**
Emulate PC boot sequence at the Mouse's serial connections.
- **update()**
Processes all information stored in the rxUart buffer and updates distance measurements and button triggered event flags.
- **clearStatus()**
Clears the status of all button triggered event flags

Program Listing 1.2 uses the methods below from the **MS2ButtonSerialTerminalHelper** class.

- **cls()**
Clears all text from the Messages from Javelin window.

- `indexInit()`
Create a list of character arrays. Each element in the list is a copy of an element in the `monitorOutline` string array that has been given a character array reference.
- `indexDisplay()`
Displays the contents of the list in the Messages from Javelin window.
- `placeChar(, ,)`
Prints a character at a particular location in the 'Message From Javelin' window.
- `index(,)`
Get a character at a particular column (x) and row (y) in the list of character arrays.
- `cursorInRegion(, , ,)`
Determines if the cursor has been placed in a particular region of the display. The region is defined by the upper left and lower right coordinates in (x,y) implies (column, row) format.
- `indexUpdate(, ,)`
Change the value of a character at a particular column (x) and row (y) in the list of character arrays.
- `indexReInit()`
Recopy the contents of the `monitorOutline` string array into the list of character arrays created by the `indexInit` method.

Tip

Use the **MS2ButtonSerial** javadoc file (`MS2ButtonSerial.pdf` in your <"Javelin Stamp IDE\doc"> folder) as a reference to find out more about using the methods available to you.

Program Listing 1.2 – MS2ButtonSerialExample

```
import stamp.core.*;
import stamp.peripheral.hid.mouse.serial.MS2ButtonSerial;
import examples.peripheral.hid.mouse.serial.MS2ButtonSerialTerminalHelper;

/* This program will allow you to drag and drop characters within a box on the
 * IDE's "Messages from the Javelin" window.
 * For more information see application note 10 (an010) from Parallax, Inc.
 *
 * Version 1.0 - 12/27/02
 */
public class MS2ButtonSerialExample {
```

```
// Declare I/O pins connected to Mouse via RS232.
final static int DTR = CPU.pin0;           // COM Port 1
final static int RX  = CPU.pin1;           // COM Port 3
final static int RTS = CPU.pin2;           // COM Port 8

// Declare receiver Uart for serial mouse connection/data.
static Uart rxUart = new Uart( Uart.dirReceive, RX, Uart.dontInvert,
                               Uart.speed1200, Uart.stop1);

// Initialize values that will be passed to the mouse object.
// See MS2ButtonSerial javadoc (MS2ButtonSerial.pdf) for details.
static int x=15, y=7, xOld=15, yOld=7, scale=40, xMin=2, yMin=3, xMax=28, yMax=11;

// Use values to create and initialize mouse object.
public static MS2ButtonSerial mouse = new MS2ButtonSerial
(rxUart, DTR, RTS, scale, xMin, yMin, xMax, yMax, x, y );

// Declare a display object. This display is for demonstration purposes only.
static MS2ButtonSerialTerminalHelper display = new MS2ButtonSerialTerminalHelper();

// Declare intermediate variables that store values.
static int xInitial, xFinal, yInitial, yFinal;
static char c;

public static void main(){

    // Initialize display object.
    display.cls();
    display.indexInit();
    display.indexDisplay();

    // Initialize mouse to communicate with the Javelin Stamp.
    mouse.bootSequence();

    // Place display pointer. We'll control the motion of an asterisk
    // using the mouse.
    display.placeChar(x,y,'*');

    // Get initial position information from the mouse object. These
    // positions were set in the mouse object's constructor.
    x = mouse.xDistance/scale;
    y = mouse.yDistance/scale;

    // This infinite loop calls the update method, then processes and displays
    // the resulting information. Distance values are scaled for a small
    // display.
    while(true){

        mouse.update();

        x = mouse.xDistance/scale;
        y = mouse.yDistance/scale;
```

```
// When a drag operation is detected, record the initial values and
// replace the asterisk cursor with the character being dragged.
if(mouse.leftDrag){
    xInitial = mouse.xDragStart/scale;
    yInitial = mouse.yDragStart/scale;
    c = display.index(xInitial,yInitial);
    display.placeChar(x,y,c);
}

// When a drop is detected, copy the character and update the display
// if the character is dropped in the lower portion of the display;
// otherwise discard the action and display the cursor.
else if(mouse.leftDrop){

    // Copy the character from the start of the drag to the location of
    // the drop and update the display object using the indexUpdate method.
    if(!display.cursorInRegion(2,2,30,8)){
        xFinal = mouse.xDragEnd/scale;
        yFinal = mouse.yDragEnd/scale;
        display.indexUpdate(xFinal,yFinal,c);
        display.placeChar(x,y,c);
        mouse.clearStatus();
    }

    // Update the cursor location and forget about the drag and drop.
    else{
        display.placeChar(x,y,'*');
        mouse.clearStatus();
    }
}

// If no drag and drop activity, update the cursor location.
else{
    display.placeChar(x,y,'*');
}

// If right-click and release, clear the display and forget the
// character location date.
if(mouse.rightDrop){
    display.cls();
    display.indexReInit();
    display.indexDisplay();
    x = mouse.xDistance/scale;
    y = mouse.yDistance/scale;
    display.placeChar(x,y,'*');
    mouse.clearStatus();
}

// Place the old character where the cursor is displayed. This has
// two purposes. First, the cursor flashes on/off and displays the
// character under it, and second, when the cursor moves to a new cell,
```

```
// the old cell remains unchanged.  
display.placeChar(xOld,yOld,display.index(xOld,yOld));  
xOld = x;  
yOld = y;  
}  
}
```

Published Resources – for More Information

This class was developed to allow the Javelin to interact with a serial mouse. Much care has been taken in creating this class. Below you will find useful information that was used in creating this document.

“PC Mouse Information”, Web Page, Helsinki University of Technology, www.hut.fi

This document will give you detailed information about the internal workings of a serial mouse.

Javelin Stamp Discussion Forum – Questions and Answers

The Parallax, Inc. Javelin Stamp Discussion Forum is a searchable repository of design questions and answers for the Javelin Stamp. To view the Javelin Stamp Forum, go to www.javelinstamp.com and follow the Discussion link. You can also join this forum and post your own questions. The Parallax technical staff, and Javelin Stamp users who monitor the list, will see your questions and reply with helpful tips, part numbers, pointers to useful web pages, etc.

Copyright © 2002 by Parallax, Inc. All rights reserved. Javelin, Stamp, and PBASIC are trademarks of Parallax, Inc., and BASIC Stamp is a registered trademark of Parallax, Inc. Windows is a registered trademark of Microsoft Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Other brand and product names are trademarks or registered trademarks of their respective holders.

Parallax, Inc. is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, and any costs of recovering, reprogramming, or reproducing any data stored in or used with Parallax products.