Versalent

May 2020

USBM232 Mouse-Serial Converter Manual

Version 1.02

Revised May 20, 2020

General Description:

The Versalent USBM232 is a small adapter which converts a standard USB *wired* mouse to an RS232 mouse for use with non-USB systems. RS232 messages are generated for each mouse movement and/or mouse click so the mouse position and state can be tracked by computer systems which do not provide a USB interface. There is no single RS232 mouse standard for representing mouse movements. Microsoft, Logitech, Sun, and MouseSystems all had slightly different standards when serial mice were prevalent. They supported different numbers of buttons, and typically no scroll wheels. USBM232 offers selectable output data formats including the original Microsoft Serial Mouse Format (no scroll wheel), a Custom Format which extends the Microsoft format with scroll wheel data, and a unique Versalent ASCII Format which consists entirely of ASCII viewable characters.

Normal mouse usage requires some type of graphic interface which allows the operator to see a mouse cursor so that appropriate actions can be taken when the mouse is in particular positions, and the USBM232 provides the incremental position data needed to navigate the cursor on a non-USB system screen. The initial release (v1.xx) of USBM232 operates with ***wired mice only***. If a mouse will operate with a standard PC without loading any special drivers, it will operate with the USBM232. Some mice offer more functions than the standard two-button mouse, however USBM232 cannot provide any of those advanced features since it has no way to load unique drivers. It provides information on X-Y movement, left and right button clicks, and scroll wheel rotation. Future versions may add operation with composite devices (mouse/keyboard combos) which will allow for wireless mice.

When the USBM232 is powered on either by providing power through the DB9 connector, or with the optional wall-brick, it sends a short string containing the USBM232 product name and its firmware version. The GREEN LED flashes very quickly when there is no USB mouse attached, then flashes noticeably slower once it recognizes an attached mouse. It then begins sending ASCII or binary messages as it detects mouse movements.

USBM232 Output Message Formats:

There are three selectable formats for the serial output of mouse data. Refer to the command set below to set the desired format. Once set, this value is retained in non-volatile storage so the device powers up and/or resets to this mode.

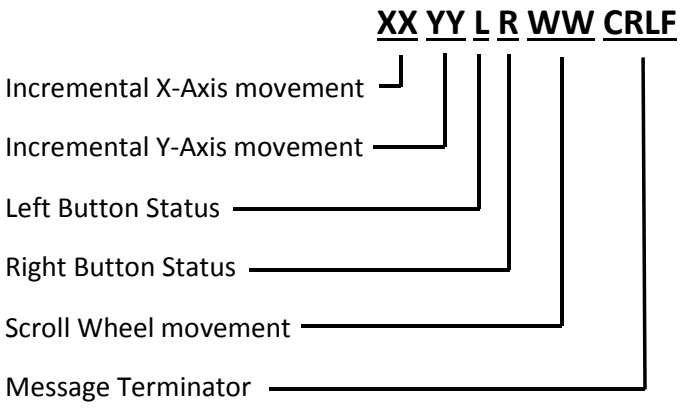
1) **Microsoft Serial Mouse Format** -- this is a binary format (meaning that some characters are not printable/viewable). It is the most efficient of the three available formats in that each message contains only 3 characters (bytes) of data. Because of the small message size it is most suitable for low baud rate applications, although when it was devised, mice did not have scroll wheels so it contains no field to transfer that information. This format transfers data for X-Y movement and L/R mouse buttons only. The details of the data format are contained below in the Command Set description that follows.

2) **Custom Binary Mode** -- this binary mode is similar to the Microsoft Serial Mouse Format, except that it adds another byte for the scroll wheel and the message length is therefore 4 characters. This is a unique format created by Versalent to maintain the transfer efficiency of the Microsoft format while adding scroll wheel data. Characters are not printable/viewable but it allows for efficient transfer of X-Y movement, L/R mouse buttons, and scroll wheel movement. The details of this data format are contained in the Command Set description that follows.

3) **ASCII Mode** -- this unique Versalent ASCII mode format is the least efficient of the 3 modes, however provides for human-readable messages. It uses 10 characters per movement message and displays neatly formatted messages on a terminal-emulator screen. It can be used for diagnostics and validation, or for applications that can provide higher baud rates (typically > 9600). Data analysis by the receiving application is easier since X-Y data and button data is not mixed within bytes as it is in the two binary formats. The receiver's serial buffer may have to be larger than for the binary modes since there are significantly more characters transferred. The detailed description of this format follows:

ASCII Format Description:

For each incremental mouse movement or click, a short RS232 (ASCII) message is output which has the following format:



XX represents two characters which describe the x-axis incremental movement since the last message. These two ASCII hexadecimal digits form a ***signed*** (two's complement) number ranging from "00" to "7F" for forward x-axis motion, and "FF" down to "80" for reverse x-axis motion. So a single increment of motion in the positive direction is represented by "01" and a single increment of motion in the reverse direction is represented by "FF".

YY represents two characters which describe the y-axis incremental movement since the last message. These two ASCII hexadecimal digits form a ***signed*** (two's complement) number ranging from "00" to "7F" for forward y-axis motion, and "FF" down to "80" for reverse y-axis motion. . Again a single increment of motion in the positive direction is represented by "01" and a single increment of motion in the reverse direction is represented by "FF".

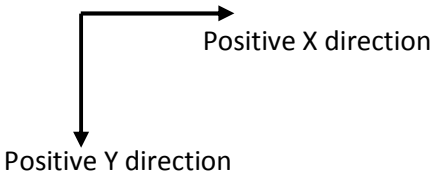
L represents a single character indicating the current state of the Left Mouse Button . When it is pressed L= "1" and when released this value is "0". A press or release of the button (with or without any x/y movement) will cause a new RS232 message to be generated.

R represents a single character indicating the current state of the Right Mouse Button . When it is pressed R = "1" and when released this value is "0". A press or release of the button (with or without any x/y movement) will cause a new RS232 message to be generated.

WW represents two characters which describe the incremental scroll wheel movement since the last message. These two ASCII hexadecimal digits form a ***signed*** (two's complement) number ranging from "00" to "7F" for forward y-axis motion, and "FF" down to "80" for reverse y-axis motion.

CRLF represents the standard RS232 Carriage Return and Line Feed Characters which mark the end of the message.

The sign of the mouse motion is given by the mouse standard below:



Download sample C# message parsing code at
https://www.versalent.biz/downloads/usbmsamplecode.txt

ASCII Format Messaging Rate:

This message format provides human readability at the expense of message size over the binary formats. Quick mouse movements can transfer a lot of characters -- fast movements can generate approximately:

(10 chars/message) X (250 messages/0.25 second) = 2500 characters per ¼ second.

Since each movement message consists of 10 characters, it is recommended that the USBM232 be operated at a minimum of 9600 baud corresponding to a message transmission time of approximately 10ms -- higher rates are recommended to minimize transmission latencies and keep the mouse responsive. The USBM232 supports up to 115k baud and is shipped with a default 56k baud setting.

The above numbers are typical for a fast mouse movement of approximately one foot, so the host should have a receive buffer of 2k characters or more. Even this may not be enough if the host does not process these received messages quickly, or it pauses processing while performing other tasks. To help prevent the host from being overrun, the USBM232 implements the XON/XOFF protocol which allows the host to stop the USBM232 from transmitting new messages if it is in danger of being overrun. When the host sends the RS232 XOFF character, the USBM232 immediately stops sending messages until it receives an XON character. If the mouse continues to move and generate data, that data is discarded.

XON/XOFF Data Management:

The USB232 will only accept an XOFF if there is currently a USB mouse connected, and enumerated. This means that the LED flash rate has slowed to its lower rate of about 1 flash per second with equal on and off times. Once the XOFF is received, the flash ***rate*** remains the same, however the led on time becomes very short so the light pulse changes noticeably. This generally will not be seen because the duration of an XOFF state is typically very short .. the host usually sendd an XON following an XOFF within a fraction of a second to resume communications. However if the host leaves the USBM232 in the XOFF state for an extended period, this change in flash pattern will be evident. The XON character is 0x11 also referred to as DC1, and XOFF is 0x13 also referred to as DC3.

USBM232 provides a fail-safe communications re-start timeout of 10 seconds .. if the USBM232 has been left in the XOFF state for more than 10 seconds, it will automatically negate the XOFF command and resume sending position messages . This prevents the USBM232 from being permanently disabled.

Power and LED Operation:

When the USBM232 is powered on either by providing power through the DB9 connector, or with the optional wall-brick, the GREEN LED flashes very quickly when there is no USB mouse attached. The flash rate slows to 1 flash per second once it recognizes an attached mouse. This visual indication provides a confirmation that the mouse was recognized and initialized (or 'enumerated' in USB terms). It issues no power-up messages to the RS232 host and begins sending mouse position messages in the

last output format selected. This allows the fastest and most seamless return to normal operation from a power interruption, or other USBM232 reset. The host can at any time confirm its connection to a USBM232 by sending the 'V' command.

Command set:

The USBM executes the following command set. These commands and responses are all ASCII characters and can be executed at any time during operation. Commands are all single characters and none require additional parameters as described below.

1) **V** -- Return the firmware version number. The USBM232 flushes any mouse position data in its buffer, and returns "USBM232 vX.XX" where X.XX i.e. 1.02 contains major, followed by minor version numbers.

2) **[XON]** -- This is the single ASCII character (0x11) which re-enables communications after a previous [XOFF] command had paused communications to prevent host overrun. There is no response to this command. USBM resumes outputting serial characters.

3) **[XOFF]** – This is the single ASCII character (0x13) which temporarily disables USBM232 serial output to prevent host overrun. There is no response to this command. USBM232 remains in this disabled state awaiting an XON command to resume communications. If no XON is received within 10 seconds, USBM232 automatically resumes communications.

A -- Set the output mode to (Versalent) ASCII mode. This is the default, as-shipped output mode, and uses a more human readable output format which can be viewed directly with a terminal emulation program. You will notice that the binary formats mix X-Y data and button in the 1st byte, so viewing this data via a terminal emulator is difficult. However instead of 3/4 bytes of binary data this format uses 10 bytes of hexadecimal formatted digits. It is less efficient and requires higher baud rates to prevent transmission latency in the mouse system. When received the USBM232 flushes its output buffer, saves the setting in non-volatile memory and responds with ASCII [ACK] if successful or [NAK} if unsuccessful. The [ACK] will be delayed by the 10ms required to write to flash memory. Mouse data generation and transmission then resumes.

The non-volatile setting causes the USBM232 to then powers up (or reset) in this mode so if the USBM232 undergoes brown-out or watchdog resets which the host may be unaware of, it resumes operation in this last-set mode.

4) **B** -- Set the output mode to Microsoft Serial Mouse Format. When received the USBM232 flushes its output buffer, saves the setting in non-volatile memory and responds with ASCII [ACK] if successful or [NAK} if unsuccessful. . The response may be delayed by up to 50ms as required to erase/re-write flash memory. In this binary format mode only 3 bytes are sent per mouse movement as shown below. Notice that no scroll wheel data is available in this format. This format is very efficient can be very useful for low baud rate systems.

	D7	D6	D5	D4	D3	D2	D1	D0
1st byte	0	1	LB	RB	Y7	Y6	X7	X6
2nd byte	0	0	X5	X4	X3	X2	X1	X0
3rd byte	0	0	Y5	Y4	Y3	Y2	Y1	Y0

```
LB is the state of the left  button, 1 = pressed, 0 = released.
RB is the state of the right button, 1 = pressed, 0 = released
X0-7 is movement of the mouse in the X direction since the
     last packet.  Positive movement is toward the right.
Y0-7 is movement of the mouse in the Y direction since the
     last packet.  Positive movement is back, toward the user.
```

Note that these are 8-bit characters with the upper bit always 0. And as these messages can be packed, the host can synchronize itself and identify the first byte/character of a message by D6 = '1' in the first byte of a message.

The non-volatile setting causes the USBM232 to then powers up (or reset) in this mode so if the USBM232 undergoes brown-out or watchdog resets which the host may be unaware of, it resumes operation in this last-set mode.

5) **C** -- Set the output mode to Custom Binary Mode. When received the USBM232 flushes its output buffer, saves the setting in non-volatile memory and responds with ASCII [ACK] if successful or [NAK} if unsuccessful. This binary mode is similar to the Microsoft Serial Mouse format, except that it adds another byte for the scroll wheel. The message length is therefore 4 bytes. The response may be delayed by up to 50ms as required to erase/re-write flash memory. Mouse data generation and transmission then resumes. . This format can be very useful for low baud rate systems.

	D7	D6	D5	D4	D3	D2	D1	D0
1st byte	1	0	LB	RB	Y7	Y6	X7	X6
2nd byte	0	S7	X5	X4	X3	X2	X1	X0
3rd byte	0	S6	Y5	Y4	Y3	Y2	Y1	Y0
4th byte	0	0	S5	S4	S3	S2	S1	S0

```
LB is the state of the left  button, 1 = pressed, 0 = released.
RB is the state of the right button, 1 = pressed, 0 = released
X0-7 is movement of the mouse in the X direction since the
     last packet.  Positive movement is toward the right.
Y0-7 is movement of the mouse in the Y direction since the
     last packet.  Positive movement is back, toward the user.
```

S0-7 is the movement of the scroll wheel. Positive is a roll forward, negative is a roll back. The user can decide whether to add this movement to the x-axis, or y-axis position. Note that the first byte of each message has D7=1.

The non-volatile setting causes the USBM232 to then powers up (or reset) in this mode so if the USBM232 undergoes brown-out or watchdog resets which the host may be unaware of, it resumes operation in this last-set mode.

6) **D** – Set the output to Diagnostic Mode. All three outputs (ASCII, MS Serial Mouse, and Custom Binary) are output for each mouse movement. The USBM232 responds with an ASCII [ACK]. This mode is generally used only for factory testing. Each mouse movement is output to a single line of a terminal emulator which must be capable of displaying hexadecimal values for non-ASCII characters. The operator can view the ASCII output, and the associated binary bytes of those output formats. Note that his mode is **NOT** saved in non-volatile memory upon command receipt as are the other output modes. The USBM232 stays in this mode until a different output mode command is received, or the device is power cycled/reset.

7) **M** -- Return the current Binary or ASCII mode of the USBM232. . When received, the USBM232 stops transmitting, flushes any mouse data in its output buffer, and responds with an 'A' , 'B' , 'C' or 'D' indicating whether it is in ASCII mode, or Microsoft Binary mode, Custom Binary mode, or Diagnostic mode.

8) **S** -- Stop generating mouse position data (temporarily). When received, the USBM232 flushes any mouse data queued for transmission and does not generate any more mouse data. The USBM232 will respond to other commands like (V)ersion, and (M) ode etc which return responses (since the transmitter is not disabled as with XOFF). For safety, the STOP state is exited automatically after 2 seconds, or immediately after receipt of the (R)esume command. This command is generally used during factory testing, however is made available to the user.

9) **R** – Resume generating mouse data after an (S)top command was issued. If no Stop was issued, or a previous Stop has timed out, this command has no effect.

Reliability Features:

USBM232 implements an internal watchdog timer, and an internal brown-out detector. Should its microcontroller get disrupted through static discharge or other temporary interference, the watchdog will automatically reset the unit so that normal operation resumes with no user intervention. Or should power droop below an operational threshold the brown-out detector will suspend operation until power is restored to a normal level. At that point it will resume operation from a reset state again with no user intervention. Resumption of operation following any kind of reset occurs in approximately 1 second.

Baud Rate/Parity Control:

Baud and parity are set using the 5 internal shorting jumpers as shown below. To open the snap-together plastic case and access them, there are 2 'screw-driver slots' on each side of the case at the seam. Gently pry the case open using a small flat-blade screwdriver at any one of these slots with a little inward pressure while twisting the blade. After the first internal latch releases, the others release even easier and the two halves of the case separate.

USBM232 can be set to any of the baud rates/parity settings shown in TABLE 1. To configure it the plastic case is opened, and 5 internal shunt blocks are moved to the positions shown. Note that the 3 shunt blocks on the left control the baud rate and the right two select parity. The table shows shaded blocks where shunts are to be installed to select the specified settings. After changing these settings the USBM232 must be power cycled to set the new serial settings. All baud settings implement 1 stop bit.

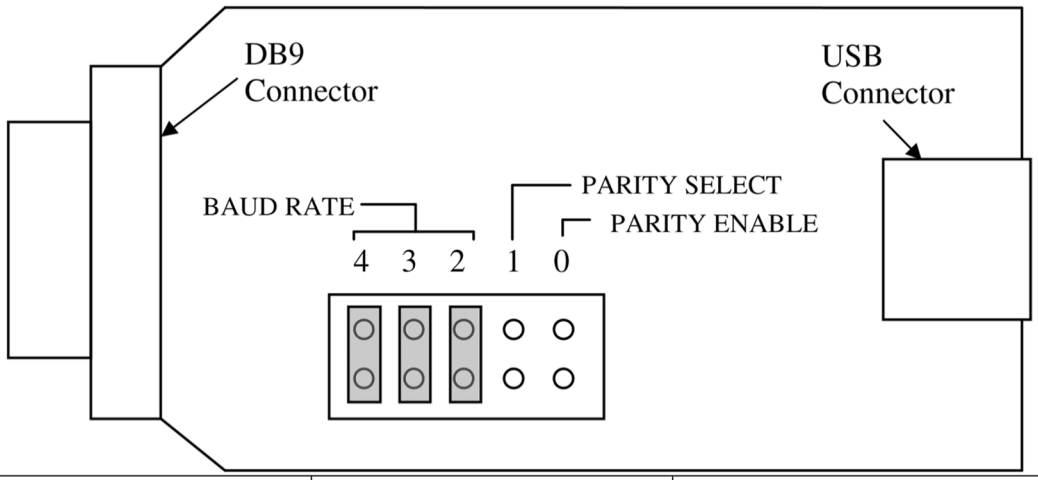
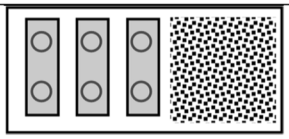
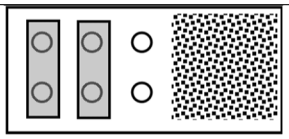
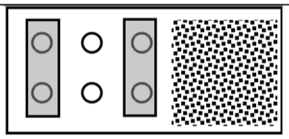
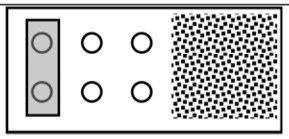
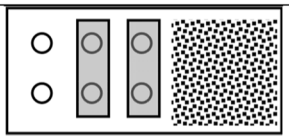
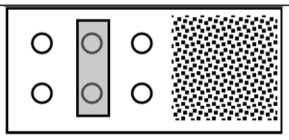
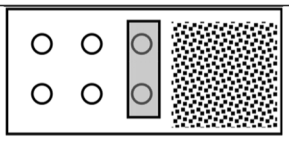
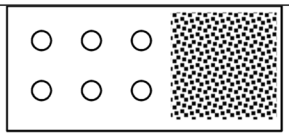


DIAGRAM	BAUD RATE	PARITY
	600 bps	See below
	1200 bps	See below
	4800 bps	See below
	9600 bps	See below
	19.2k bps	See below
	38.4k bps	See below
	57.6k bps	See below
	115k bps	See below

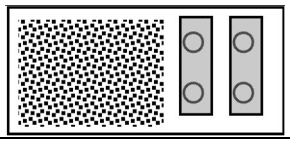
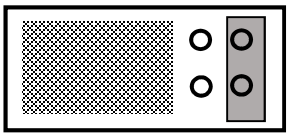
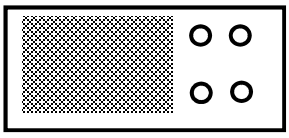
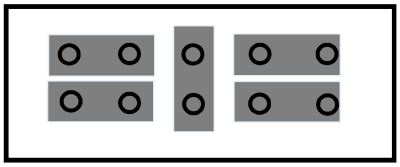
	See above	Parity Enabled, ODD Parity
	See above	Parity Enabled, EVEN Parity
	See above	Parity Disabled

TABLE 1

.

Note that any shunt blocks in a horizontal orientation have no effect on serial settings and are typically used for storing the shunts. USBM232 devices are shipped in the following default configuration .. 57.6k baud, no parity. (All the horizontal blocks have no effect; the only effective one is the vertical one in the middle – 57.6k setting shown.)



Electrical Parameters:

Absolute Maximum Input Voltage	+15VDC (Internal Regulator Models only)*
Nominal Voltage Input:	+5VDC +/- 5% or 6-12VDC (per model number)
Current Input:	35 mA maximum (no mouse attached)
Baud Rate:	600 to 115k selectable via shorting blocks
Parity:	Selectable ODD/EVEN/NONE
DB9 Connector Configuration:	DTE or DCE (per model number)
Handshake Supported	XON/XOFF

*Includes Models -024, -026, -044, -046

Environmental:

Max Operating Temperature:	+70°C
Min Operating Temperature:	0°C
Max Storage Temperature:	+80°C
Min Storage Temperature:	-20°C
Humidity:	Non-condensing at all temperatures

Physical:

Size:	2.6" X 1.7" X 0.8"
Weight:	1.4 oz
Mouse Connector:	Standard USB Type A
Host Connector:	Standard DB9 (female)
Case Color:	Black (Ivory available special order)
Power Connector:	5mm X 2.1 mm (male, Ctr Positive)

Document Revision Record:

Revision #	Revision Date	Description
V1.00	May 14, 2020	Modified from the Preliminary manual
V1.01	May 17, 2020	Add binary output modes
V1.02	May 20, 2020	Add hyperlink to sample code