Versalent

USBM232A Mouse-Serial Converter Manual

Version 2.06

Revised Mar 1, 2026

General Description:

The Versalent USBM232A is a small adapter which converts a standard *wired or wireless* USB mouse to an RS232 mouse for use with non-USB systems. RS232 messages are generated periodically so the mouse position and button states 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. USBM232A 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.

The main differences between this 'A' version, and the original USBM232 are:

1) The original USBM232 worked with wired mice only (USBM232A also allows wireless)
2) The serial commands that the 'A' version accepts contain more characters – to better protect the device from inadvertent configuration changes
3) E-baud is a standard feature (baud rate changed electronically, without opening the case)

Normal mouse usage requires some type of graphic interface which allows the operator to see a moving cursor. USBM232/USBM232A provide the incremental position data needed to navigate the cursor on a non-USB system screen. If a mouse will operate with a standard PC without loading any special drivers, it will operate with the USBM232A. Some mice offer more functions than the standard two-button/scroll wheel mouse, however USBM232A cannot provide any of those advanced features since it cannot load unique drivers. It provides information on X-Y movement, left and right button clicks, and scroll wheel rotation. USBM232A will accept a standard wired USB mouse, or a composite dongle (which typically also supports a wireless keyboard). The USBM232A does NOT talk to the keyboard .. it provides the mouse interface only.

When the USBM232A is initially powered on, the GREEN LED flashes very quickly until it finds a wired/wireless USB mouse – then flashes noticeably slower indicating the enumeration (USB connection). It then begins sending RS232 position & button messages as it detects mouse movements and clicks. The output format of the RS232 messages is one of three user-selectable formats.

1) MS Serial Mouse format – original 3-byte-per-message for X/Y positions and clicks.
2) Custom format - this is similar to the MS Serial format in that it is tightly packed , and adds a 4th byte to transfer scroll wheel data.
3) ASCII format – this is a Versalent-defined ASCII-character format which is more human-readable that the MS or Custom formats. There are no standard drivers for it, and it requires higher baud rates that the other formats.

More details on these 3 formats follow.

Microsoft still includes its original serial mouse driver in it's latest operating systems (up to WIN10) which means a standard Windows computer can still use a serial mouse (a USB mouse with the USBM232A). This driver requires that when the serial Request-To-Send signal is toggled, the mouse responds with a 1200 Baud/No Parity 'M' character, so the USBM232A implements this RTS feature when in MS Mouse mode. **When using MS Mouse mode, the RTS signal MUST either be driven by your host's serial port, or tied to an inactive low level (0v to -25v**). The USBM232A does tie this signal weakly to ground, however if a long wire is connected to the USBM232A's DB9 pin and is left unconnected, cable crosstalk can randomly inject noise that acts like an RTS signal disrupting mouse communications.

Alternatively the USBM232A can be ordered with the 'RR5' option which disconnects the RTS signal from the DB9 connector. MS Mouse mode will still operate however the auto-find feature of the Windows driver will not operate so the mouse will not be automatically detected by a Windows/DOS computer.

If your WIN7 or greater computer fails to communicate with a serial mouse, it may be necessary to issue the following command 'sc config sermouse start=demand' in a DOS command-line window with administrator privileges.

Models Available:

To accommodate various power options and RS232 configurations, the USBM232A is offered in the following models:

Model#	RS232 Type	Power Input	Defining Features
USBM232A-014	DCE	+5VDC	Power input on DB9 Pin 4
USBM232A-016	DCE	+5VDC	Power input on DB9 Pin 6
USBM232A-034	DTE	+5VDC	Power input on DB9 Pin 4
USBM232A-036	DTE	+5VDC	Power input on DB9 Pin 6
USBM232A-024	DCE	6-12VDC	Power input on DB9 Pin 4, or power jack/ext wall brick
USBM232A-026	DCE	6-12VDC	Power input on DB9 Pin 6, or power jack/ext wall brick
USBM232A-044	DTE	6-12VDC	Power input on DB9 Pin 4, or power jack/ext wall brick
USBM232A-046	DTE	6-12VDC	Power input on DB9 Pin 6, or power jack/ext wall brick
USBM232A-124	DCE	4-30VDC	Power input on DB9 Pin 4, or power jack/ext wall brick
USBM232A-126	DCE	4-30VDC	Power input on DB9 Pin 6, or power jack/ext wall brick
USBM232A-144	DTE	4-30VDC	Power input on DB9 Pin 4, or power jack/ext wall brick
USBM232A-146	DTE	4-30VDC	Power input on DB9 Pin 6, or power jack/ext wall brick

The 'RR5' suffix can be added to any of the above models. This suffix disables the RTS signal and is intended for use with devices NOT operating the Microsoft Serial Protocol. MS protocol uses RTS to auto-identify the COM port to which a serial mouse is connected .. and is generally required for that protocol. If the user configures the USBM232A to operate a different protocol, it is possible that an unused/undriven RTS signal can inject noise and interfere with USBM232A operation.

USBM232/USBM232A Configuration Application:

A Windows USBM232 Configuration utility is available for download at www.versalent.biz/dl.htm which allows you to configure and test both USBM232/USBM232A devices. It offers the command set below for unit setup and testing.

Wireless Mouse Pairing:

When using a wireless mouse, the mouse must have been previously paired with its dongle-receiver. This process may be as simple as pressing a button on the mouse, or in the case of Logitech it may require running a connection utility. Before using a new mouse its dongle should be plugged into a PC to confirm that it has been paired and works.

An unpaired dongle will be recognized by the USBM232A and will slow the LED flashing to indicate that a USB connection has been established. However this does not confirm that the dongle is paired with, and communicating with the mouse. Follow the manufacturer's directions to accomplish the pairing – typically done with a PC so the mouse can be seen moving the cursor on the screen as a confirmation of successful pairing. The dongle is then ready to be plugged into the USBM232A.

USBM232A 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 (some characters are not printable). 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. 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.

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 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.

3) **ASCII Mode** -- this unique Versalent ASCII mode format is the least efficient of the 3 user 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 across 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 details of this data format are contained in the Command Set description .

XON/XOFF Data Management:

The XON/XOFF protocol allows a serial host to pause the USBM232A's messages if it cannot process them. The USB232A 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. 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. The XON character is 0x11 also referred to as DC1, and XOFF is 0x13 also referred to as DC3.

USBM232A provides a fail-safe communications re-start timeout of 3 seconds .. if the USBM232A has been left in the XOFF state for longer than this, it will automatically negate the XOFF command and resume sending position messages. This prevents the USBM232A from unintentionally being disabled for more than a short period.

Power and LED Operation:

When the USBM232A is powered on either by providing power through the DB9 connector, or with the optional wall-brick, the GREEN LED flashes very quickly with 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 detected and initialized (enumerated). It issues no power-up messages to the RS232 host and begins sending mouse position messages in the selected output format. This allows the fastest and most seamless return to normal operation from a power interruption, or other USBM232A reset. The host can at any time confirm its connection to a USBM232A by sending the '/V' command – USBM232A returns a firmware version (ASCII) string.

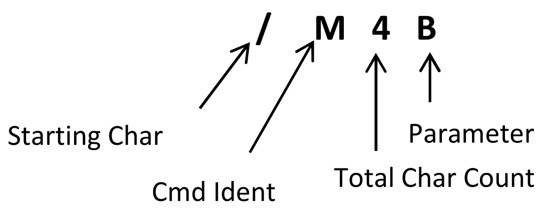
Command set:

USBM232A executes the following command set. These commands and responses are all ASCII characters and can be executed at any time during operation. It is not uncommon for a host to generate a few random RS232 characters at power-up or on reset and these could potentially corrupt a device setting. So the 'A' version commands contain more characters than the original USBM232's one-character commands making them less susceptible to unintended execution.

Command Structure:

1) First character of each command is '/' .
2) 2nd character is the ASCII command identifier .
3) 3rd character is a ASCII count of the characters in the command, '0' – '9'.
4) Any optional ASCII parameters follow.

Example: The 'M' command to set the mouse operating mode to MS Serial is:



Commands can return data, or simply an 'ACK' or 'NAK' character. If data is returned, it is an ASCII representation of a value with no formatting. If the command returns no data and is accepted, it returns 'ACK'. If the command is formatted incorrectly it will return 'NAK'.

Command Identifier	**Description**
'M'	Set /Get the operating mode. If issued with no parameter it is a request for the current operating mode. Valid modes are 'A' = ASCII, 'B' = Binary (MS Serial) or 'C' = Custom and the response will be one of these. If the command is issued with a parameter, that parameter must be 'A', 'B', or 'C' and the response will be 'ACK'. On success the value is stored in non-volatile memory for automatic recall at power up. The response is delayed by the 40ms required to write to the non-volatile memory.
'Y'	Set/Get the sensitivity. To accommodate various mice and adjust for user preferences, the mouse can be made more or less responsive by setting this level from '1' to '9'. The default is '4'. If issued with no parameter it is a request for the current sensitivity and the response will be '1' – '9'. To change the sensitivity it must be issued with one of these values, and the response will be 'ACK'. On success the value is stored in non-

volatile memory for automatic recall at power up. The response is delayed by the 40ms required to write to the non-volatile memory.

‘B’ Set the baud rate. This command requires two parameters:

Baud Rate	Parameter1	Parity	Parameter2
600	‘0’		
1200	‘1’	NONE	‘0’
4800	‘2’		
9600	‘3’	EVEN	‘1’
19200	‘4’		
38400	‘5’	ODD	‘2’
57600	‘6’		
115.2k	‘7’		
Use Jumper Setting	‘8’	Use Jumper Setting	Any character

The response to this command is ‘ACK’ which is issued at the original baud rate. After the response has completed transmission, the baud rate is changed. On success the value is stored in non-volatile memory for automatic recall at power up. The response is delayed by the 40ms required to write to the non-volatile memory.

‘S’ Stop mouse position output. Since the USBM232A generates mouse position output whenever the mouse is moved, this data must be halted when commands are sent to the device in order to interpret command responses. The Stop command prevents the USBM232A from generating position output during command interactions. If a serial host inadvertently leaves a device in this state there will be no mouse output. To prevent this, the USBM232A automatically removes the ‘stop’ state 3 seconds after it was commanded. The command format is /S3 .

‘R’ Resume mouse position output. A serial host should issue this command after a previous ‘S’ command , and a device-setting command (Mode, RTS Enable, Sensitivity etc) has been issued. The USBM232A includes automatic recovery as described above in the ‘S’ command. The command format is /R3 .

‘T’ Get/Set RTS DISABLE. As described previously, the MS Serial (Binary) mode uses the RS232 RTS signal to auto-identify a connected serial mouse and attach it to the Microsoft driver. It is possible to use the MS Serial format without using the Microsoft driver so the RTS signal may not be needed. As described above, when using the MS Serial mode, the RTS signal should not be left ‘floating’ or it could inject noise into the mouse data stream. This command provides another way (instead of ordering the RR5 option) to disable the RTS signal and prevent data corruption. This setting is only in effect for the MS Serial mode.

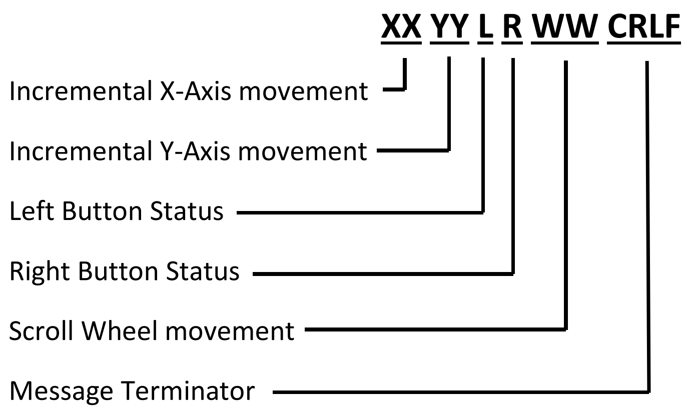
To retrieve the current state: /T3 (returns ‘1’ for RTS ENABLED, ‘0’ for DISABLED)

To set the RTS Enable: /T41 Enables RTS, /T40 Disables RTS

‘V’ Get the firmware version. This firmware string is similar to ‘USBM232A 2.04’ .This command is unique in that it is issued with no command count. So the entire command is ‘/V’ . This simplified command is used to find and identify a device connected to the USBM232A Configuration Program.

<u>ASCII Format Description:</u>

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".

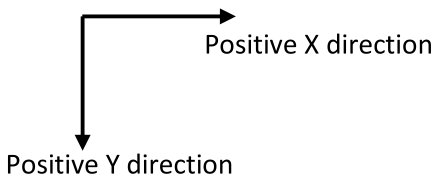
L represents a single character indicating the current state of the Left Mouse Button . When it is pressed L= "1" and when released this value is "0". A press or release of the button (with or without any x/y movement) will cause a new RS232 message to be generated.

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 9600k 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.

MS Serial Format Description:

When this mode is set, the USBM232A 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 ~ 40ms 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	1	1	LB	RB	Y7	Y6	X7	X6
2nd byte	1	0	X5	X4	X3	X2	X1	X0
3rd byte	1	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 1. 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.

Special Notes for Win98/DOS and other old Microsoft-Driver Users:

This mode is compatible with the Microsoft drivers, however to use those drivers you MUST configure USBM232S to operate at 1200 baud/No Parity even if your COM port shows that it is set to some other speed. When the mouse is detected and the driver 'takes over' the COM port it internally changes the baud rate to 1200 but in some cases does NOT update the System Device Manager to show this change. Online documents describe the MS Serial protocol as using 7-bit characters (bits 0-6) which infers that bit 7 should be inactive (set to 0).. however as shown above the protocol actually expects that Bit 7 of each character = 1.. This mode can be used at faster baud rates with a user-provided driver however the standard MS driver only operates at 1200 baud regardless of the setting of the COM port.

Custom Mode Description:

When this mode is set, the USBM232A 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 ~ 40ms 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 and can be used by the receiver to synchronize itself to the message stream.

Mouse Responsiveness:

Transferring mouse data through a low-baud serial channel requires balancing message count and mouse responsiveness. Serial messages require processing so they should be minimized, but they need to be transferred frequently enough that the on-screen cursor can track mouse movements closely. Long ago Microsoft determined that a 25ms reporting period could track a mouse well enough to keep up with human perception so mice were designed to aggregate mouse movements and report on this interval. USBM232A mimics this aggregation period which is especially important for USB mice. USBM232A uses a 10ms aggregation period for baud rates faster than 1200 to provide an even more responsive mouse yet still limit the number of messages required for good tracking.

Mouse sensitivity also affects perceived responsiveness – the number of mouse 'counts' delivered for a given motion. USBM232A provides a sensitivity setting to allow the user to adjust the speed of the perceived motion similar to the setting available in the Windows Control Panel. Sensitivities can be set from 1 to 9 (lowest to highest) with the factory default value set to 4. Note that changing the sensitivity does not affect mouse resolution because small motions of a few counts are not altered by the sensitivity algorithms.

Reliability Features:

USBM232A 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.

Baud Rate/Parity Control:

Baud rate and parity can be set using the 'B' command (E-baud) or the 5 internal jumpers as shown below. To open the snap-together plastic case, 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.

USBM232A can be set to any of the baud rates/parity settings shown in TABLE 1. To configure it the case is opened, and 5 internal shunt-block-jumpers are moved to the positions shown. Note that the 3 shunts 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. The unit will them power-up with these settings.   All baud settings implement 1 stop bit.

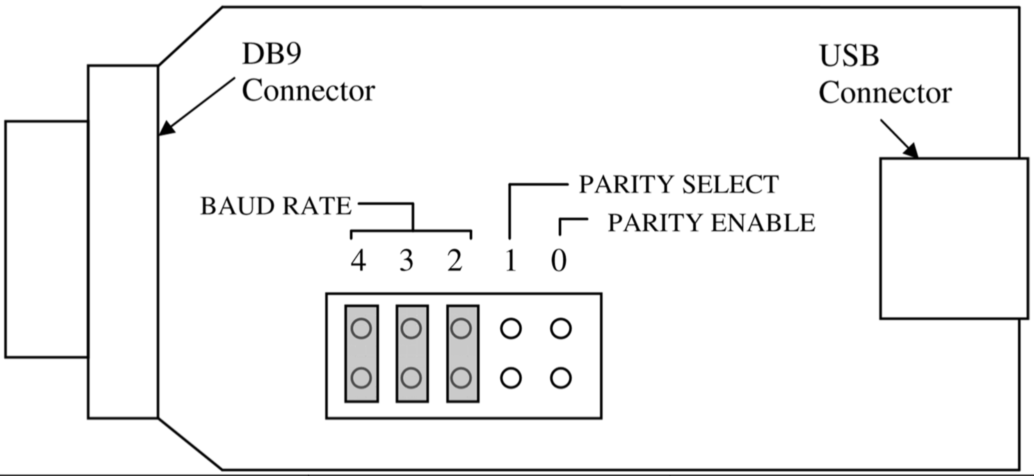
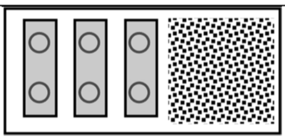
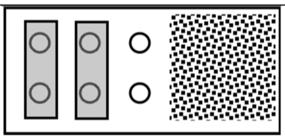
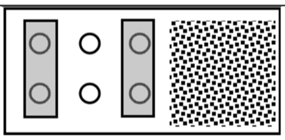
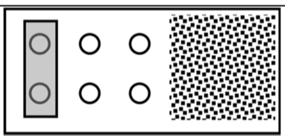
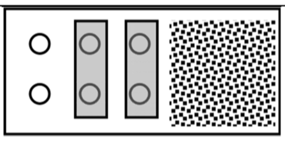
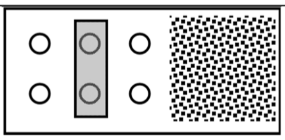
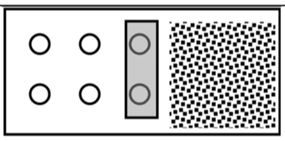
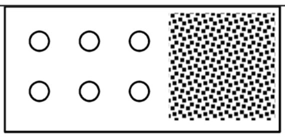


DIAGRAM	BAUD RATE	PARITY
	600 bps	See below
	1200 bps	See below
	4800 bps	See below
	9600 bps	See below
	19.2k bps	See below
	38.4k bps	See below
	57.6k bps	See below
	115k bps	See below

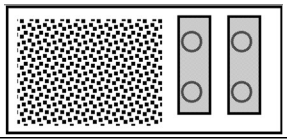
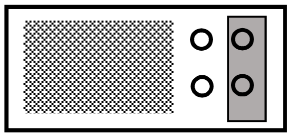
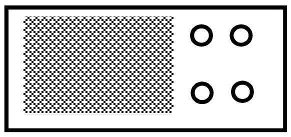
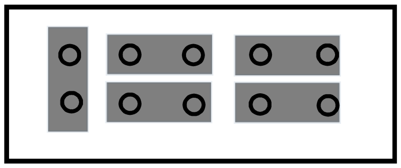
	See above	Parity Enabled, ODD Parity
	See above	Parity Enabled, EVEN Parity
	See above	Parity Disabled

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 .. 9600 baud, no parity. (All the horizontal blocks have no effect; the only effective one is the vertical one in the middle – 9600 setting shown.)



Electrical Parameters:

Absolute Maximum Input Voltage 5V models: -014,-016,-034,-036 6-12V models: -024,-026,-044,-046 4-30V models: -124,-126,-144,-146	 +5.25VDC (35mA no mouse attached) +15VDC (40mA no mouse attached) +31VDC (90mA @4V, 15mA @ 30V typical)
Nominal Voltage Input:	+5VDC , 6-12VDC, 4-30VDC
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

Environmental:

Max Operating Temperature:	+70^{o}C
Min Operating Temperature:	0^{o}C
Max Storage Temperature:	+80^{o}C
Min Storage Temperature:	-20^{o}C
Humidity:	Non-condensing at all temperatures

Physical:

Size:	2.9” 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:	5.5 mm X 2.1 mm (male, Ctr Positive)

Document Revision Record:

Revision #	Revision Date	Description
V2.04	July 24, 2025	Initial Release/modified from USBM232 manual
V2.05	Aug 17, 2025	Fix a few type-o’s
V2.06	Mar 1, 2026	Add section on mouse pairing