RS232-PS2 Operations Manual

Revised 6/14/19

GENERAL DESCRIPTION & TERMINOLOGY:

The RS232-PS2 adapter converts RS232 serial characters to PS2 protocol for direct connection to a PC PS2 port. It provides serial emulation of a standard PC-AT 102-key keyboard by translating the received characters into PS2 scan-codes (or sequences of codes). Scan codes are 8-bit codes unique to the PS2 interface, and keyboards generate a code on key-press, and another (or typically two codes) on release. This scheme allows a PC to maintain the pressed status of CTRL, ALT, SHIFT keys which are modifiers of other keys. This document refers to 'key-press' codes and 'key-release' codes, and sometimes 'code sequences' since some keys can generate several codes, and thus a single received RS232 character can generate single or multiple codes.

All this complexity is transparent to the user .. you simply send single, standard ASCII characters and RS232-PS2 handles the press/release sequences and passes the appropriate scan-codes to the PC PS2 port. Or you send one of the 8-bit characters defined in Tables 1 & 2 to send the scan codes for keyboard keys that are not ASCII (Function keys, Arrow keys, CTRL, ALT, Insert, Home, PageUp etc). This allows a non-PS2 serial host to fully control a PC program (with NO changes to that program code). While the RS232-PS2 is powered and idle the red LED indicator flashes briefly every second. It also flashes briefly for every scan code transferred to the PC, so bursts of short flashes occur while communication is occurring.

The RS232-PS2 acts just like a PS2 keyboard. It implements the PS2 power-up handshaking protocol, sends key-press codes and automatically sends the associated key-release sequences – after receiving just a single serial character. The RS232-PS2's extended 8-bit serial character set is necessary to emulate the non-ASCII keys since there are more key-cap values on a keyboard than there are ASCII characters to represent them. [**NOTE:** *Mouse operation is NOT emulated by the RS232-PS2 . That is not practical since mouse cursor positioning requires visual feedback of that position before activating the mouse buttons.*]

DETAILED DESCRIPTION:

As each character arrives, its equivalent PS2 'scan codes' are generated and sent to the PS2 interface. The serial character set (shown below in Table 2) includes all 7-bit ASCII printable characters, some of the non-printable ASCII control characters below 0x20 **, and some 8-bit characters (above the 7-bit ASCII range). This set allows full emulation of all the keys on a standard 102-key PC keyboard, therefore allows a PC program to be fully operated using serial characters to emulate arrow keys, function keys, numeric keys, CAPS and NUMLOCK keys, Control, Alt, Home, PageUP/DN etc… all PS2 keyboard functions.

** *The PS2 scan-code set does not contain special codes for all the ASCII 'control-characters' which are the set of non-printable ASCII characters from 0x00 – 0x1F. A few of them, like BACKSPACE, TAB, ESC, ENTER do have equivalent scan codes, but most do not. So to pass most control characters to a program*

the user must send the Control-Key-Press character (0x01 in Table 1), then send an ASCII printable character, followed by the Control-Key-Release character (0x02 in Table 1). This allows the serial host to pass any of the control characters to a PC program.

EXAMPLE: To send CTRL-C to a program you would send the 3 serial character sequence

0x01 0x43 0x02

[CTRL-key-press ASCII ' C' char CTRL-key-release]

The RS232-PS2 *automatically* sends the release-code for all ASCII characters right after the character so the above sequence does NOT include a character for a 'C' release code, however you DO have to send the release code for the CTRL key (a non-ASCII key). For reference only, the actual sequence of scan-codes sent for the above sequence is:

0x14 **0x21** **0xF0 0x21** **0xF0 0x14**

(CTRL Press 'C' Press 'C' Release (sent automatically) CTRL Release)

RS232 communications can be configured for low to high baud rates (1200 – 115k with or without parity). Because the PS2 protocol transfers its 'scan-codes' at an approximately equivalent 13k baud, it can be slower than the incoming RS232 rate. Additionally, a single RS232 character received can cause several (up to 5) PS2 scan-codes to be transferred to the PC. All this means that RS232 data can arrive faster than the PS2 interface can transfer it to the PC, and the RS232 receive buffer could overflow if characters arrived in tightly packed bursts.

To prevent overflow, RS232-PS2 implements the XON-XOFF protocol. If the internal 24 character RS232 buffer approaches overflow it issues an XOFF to cause the RS232 host to pause and allow the PS2 interface to catch up -- without character loss. The XOFF character (0x11) is issued when there are fewer than 5 character positions available in the receive buffer – which gives the RS232 host 4 character times to recognize that XOFF and stop transmitting. When receive-character processing catches up and there are 10 or more buffer positions available, RS232-PS2 issues XON (0x13) allowing the host to resume transmission. If this XON were to get corrupted, or was not recognized for ANY reason, the Rs232 host system would be left in a permanent XOFF condition and no further communications would occur. For added reliability, if the RS232-PS2 does not receive a character for 5 seconds after it re-enables communications by issuing an XON, it sends another one. The probability of both of these XON characters getting corrupted or ignored is very small so communications can be expected to resume reliably.

Since PC programs are designed to expect and operate with fairly slow PS2 (keyboard/human) input, it is unlikely that serial characters will arrive at a sustained high rate (lots of simulated key-presses) so RS232-PS2 receive-buffer overflow will generally not occur. However for reliability, RS232-PS2 uses the XON/XOFF protocol to prevent the possibility of overflow, and the serial host should implement it to maintain that reliability. If overflow does occur because the host ignores an XOFF, RS232-PS2 sends an Overflow (0x03) character to the serial host.

RS232-PS2 can be operated in (RS232) full duplex mode which causes it to echo each received character back to the serial host. The power-up default for this feature is OFF, so there is no echo unless it is turned on with the command character 0x1D (Table 2). This can be used as a further system-monitoring aid. If the RS232-PS2 (and PC) are currently powered-down or has undergone a reset your serial host would recognize this condition since there would be no echo.

All the printable ASCII characters from 0x20 to 0x7E cause PS2 scan codes to be generated corresponding to a keypress followed by an automatic key release. So the user sends just one character and the whole PS2 press/release sequence is automatically generated. And when the user sends one uppercase character, the entire PS2 sequence of shift-press/character-press/character-release/shift-release is generated automatically so no additional characters are required from the user.

Most non-ASCII PS2 key codes operate the same way .. Function keys, CAPSLOCK, NUMLOCK, SCROLL LOCK, PGUP, PGDN, INSERT, DELETE, HOME, ARROWS etc. Sending the single character from Table 2 causes a press/release PS2 sequence to be automatically generated. The 3 exceptions are SHIFT, CTRL and ALT. To operate these keys, the user has to explicitly turn the feature ON/OFF using the following control characters (< 0x20) as listed below in Table 1. Any serial characters not listed in Table 1 or Table 2 are ignored.

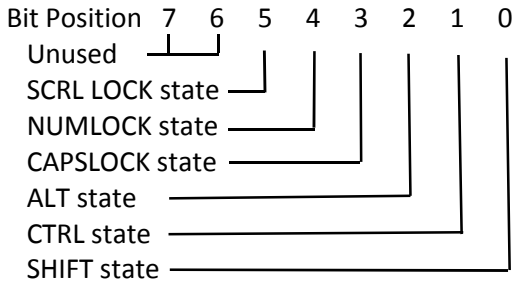
Serial Character	Function	PS2 Scan Codes Generated
0x01	Turn CONTROL mode ON. This is the same as pressing and holding the CTRL key on a PC keyboard. It remains ON until the CONTROL mode OFF character is received. Each subsequent character is processed like a keyboard key pressed with the CTRL-key pressed.	Yes
0x02	Turn CONTROL mode OFF. This is the same as releasing the CTRL key on a PC keyboard.	Yes
0x03	Turn ALT mode ON. This is the same as pressing and holding the ALT key on a PC keyboard. It remains ON until the ALT mode OFF character is received. Each subsequent character is processed like a keyboard key pressed with the ALT-key pressed.	Yes
0x04	Turn ALT mode OFF. This is the same as releasing the ALT key on a PC keyboard.	Yes
0x05	Turn SHIFT mode ON. This is the same as pressing and holding the SHIFT key on a PC keyboard. It remains ON until the SHIFT mode OFF character is received. Each subsequent character is processed like a keyboard key pressed with the SHIFT-key pressed.	Yes
0x06	Turn SHIFT mode OFF. This is the same as releasing the SHIFT key on a PC keyboard.	
0x07	Report states of CTRL key, ALT key, CAPSLOCK, NUMLOCK. A single character is returned with one bit position indicating the state of each item. A '0' in the position indicates that the feature is OFF, while a '1' indicates that the feature is ON. The returned 8-bit character/associated bit positions:  Example: 00010101 (0x15) = SHIFT ON, ALT ON, NUMLOCK ON	No

TABLE 1

Because the SHIFT, CTRL, ALT, NUMLOCK and CAPSLOCK can be set to semi-permanent states, synchronization between the PC, the RS232-PS2 and your RS232 host can be critical to proper program operation. The PC retains its version of these states, the RS232-PS2 retains its version of these states, and your host should maintain its version of these states. If your host alters them from their initial OFF states, and any of these devices undergoes a reset, or is unplugged/re-plugged, synchronization may be lost. To keep them all synchronized, and provide automatic recovery of lost synchronization the following scheme is recommended.

On a regular schedule (perhaps every 10-30 seconds) your RS232 serial host can:

1. Send the character to set the CONTROL mode to the state it expects the PC to be in. If your system thinks the CONTROL mode is ON/ OFF, send 0x01/0x02 (see table above) to set the PC to the same state. If the PC was not in this state, it has just been re-synchronized. If it was in this state, it stays there.
2. Send the character to set the ALT mode to the state it expects the PC to be in. If your system thinks the ALT mode is ON/OFF, send 0x03/0x04 (see table above) to set PC to the same state. The PC state is now synchronized.
3. Send the character to set the SHIFT mode to the state it expects the PC to be in. If your system thinks the SHIFT mode is ON/OFF, send 0x05/ 0x06 (see table above) to set the PC to the same state The PC state is now synchronized.
4. Send the character to toggle the NUMLOCK state (0x8D) .. TWICE. The first character toggles it to its alternate state, and the 2nd character toggles it back to its original state so that it remains unchanged. The effect of this is that the PC will command the RS232-PS2 to 'Update All Keyboard LEDs'. While the RS232-PS2 does not have LEDs to update, it will capture the PC's NUMLOCK, CAPSLOCK, and SCROLL-LOCK states.
5. Send the 0x07 character (see table above) to Report key states. If any of these key states does not match your host, you can send the one character needed that toggles the particular key state and make it agree with your host's state values.

All three devices (your host, the RS232-PS2, and the PC) in the chain are now key-state-synchronized.

Complete Serial Character Set with PS2 Function

Serial Character	PS2 Function
All Printable ASCII	Characters from SPACE (0x20) thru ~ (0x7E) pass thru to PS2 as their equivalent PS2 scan-codes.
0x01	Turn CONTROL mode ON. (PC acts as though CTRL key held pressed)
0x02	Turn CONTROL mode OFF. (PC acts as though CTRL key has been released)
0x03	Turn ALT mode ON. (PC acts as though ALT key held pressed)
0x04	Turn ALT mode OFF. (PC acts as though ALT key has been released)
0x05*	Turn SHIFT mode ON, (PC acts as through SHIFT key held pressed)
0x06*	Turn SHIFT mode OFF, (PC acts as though SHIFT key has been released)
0x07	Returns a single character containing state status (see table 1.0 above)
0x08	BACKSPACE character
0x09	TAB character
0x0A	Page Up
0x0B	Page Down
0x0C	Scroll Lock
0x0D	Enter (Carriage Return)
0x0E	Home
0x0F	End
0x10	Insert
0x12	Delete
0x14	Up Arrow
0x15	Left Arrow
0x16	Down Arrow
0x17	Right Arrow
0x1B	ESCAPE character
0x1C 0x1C **	TWO consecutive 0x1C characters causes the RS232-PS2 to Reset itself
0x1D	Turn ON Full Duplex RS232 (all received characters are echo'd back to host)
0x1E	Turn OFF Full Duplex – RS232 characters are not echo'd (power-up default)
0x80	F1
0x81	F2
0x82	F3
0x83	F4
0x84	F5
0x85	F6
0x86	F7
0x87	F8
0x88	F9
0x89	F10
0x8A	F11
0x8B	F12
0x8C	CAPSLOCK

0x8D	NUMLOCK
0x8E	Keypad Up Arrow (KP 8)
0x8F	Keypad 5 (KP 5)
0x90	Keypad Down Arrow (KP 2)
0x91	Keypad Insert (KP 0)
0x92	Keypad PgUp (KP 9)
0x93	Keypad Right Arrow (KP 6)
0x94	Keypad PgDown (KP 3)
0x95	Keypad Delete (KP .)
0x96	Keypad Home (KP 7)
0x97	Keypad Left Arrow (KP 4)
0x98	Keypad End (KP 1)

Table 2

* When sending RS232 uppercase ASCII characters/keys (including uppercased numerics !@#$%^&*()_+ and {}|:"<>?) you do NOT need to turn SHIFT mode ON/OFF -- RS232-PS2 does that automatically on receipt of an uppercase. The only time you need to explicitly turn SHIFT mode ON/OFF is if you want to send a shifted non-ASCII character, like SHIFT F1, SHIFT INSERT, SHIFT HOME etc. Most PC programs do not use these shifted keys, but you are able to do so if needed.

** A Reset of the RS232-PS2 does not reset the attached PC. To do that you can send CTRL-ALT-DEL, however most PCs then want a confirmation, or logoff confirmation or some additional information before resetting themselves. It is the users responsibility to determine if the PC can be reset with keystrokes alone, or if a mouse is also needed.

Shaded entries are identical to those shown in Table 1.

Differences Between RS232-PS2 and a PC Keyboard:

The RS232-PS2 emulates a PC keyboard, however there are some differences:

1) The most obvious difference is that the RS232-PS2 does not have a cord attached as does a PC keyboard. It provides a female 6-pin mini DIN connector which requires a male-to-male PS2 cable to connect to a standard PC PS2 input.
2) The RS232-PS2 has no (visible) keyboard LEDs. In this regard it is more like a typical wireless keyboard – no LEDs. However it does respond to host commands to set its internal 'led states' to ON/OFF, and those states can be read using the 0x07 character (see Tables 1 & 2 above).
3) RS232-PS2 does not auto-repeat key-presses. If you hold a keyboard key pressed, the keyboard generates repeated key-presses which causes the PC to receive multiples of that key. To make operation as simple as possible for a serial host, when RS232-PS2 receives a serial character, it sends one key-press followed by an automatic key-release (except for CTRL, ALT and SHIFT keys as described previously). The host can send multiple identical characters to simulate key-repeat.

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 there are 2 '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.

RS232-PS2 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 RS232-PS2 must be powered off/on to reset and accept the new serial settings.

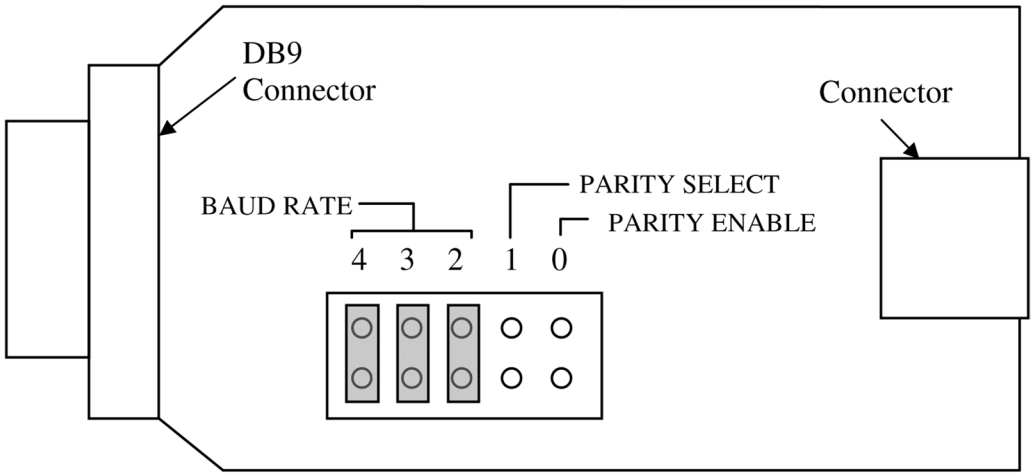
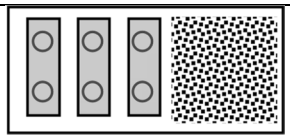
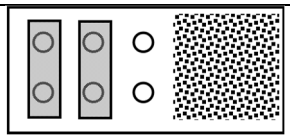
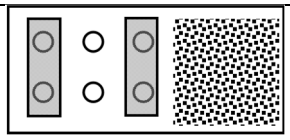


DIAGRAM	BAUD RATE	PARITY
	1200 bps	See below
	2400 bps	See below
	4800 bps	See below

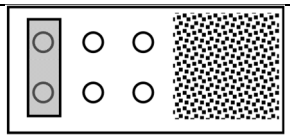
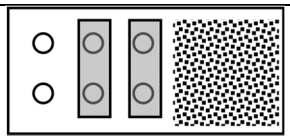
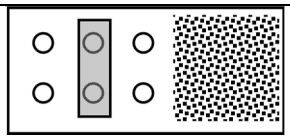
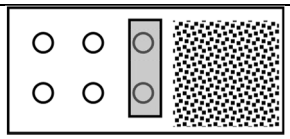
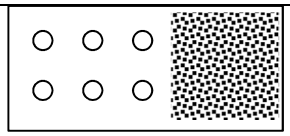
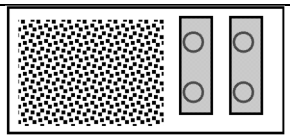
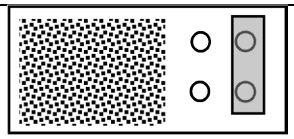
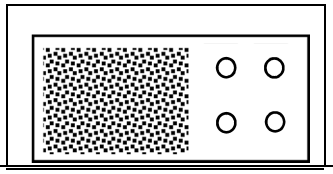
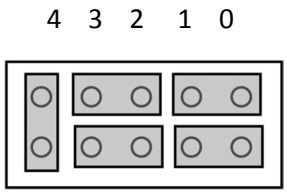
	9600 bps	See below
	19.2k bps	See below
	38.4k bps	See below
	57.6k bps	See below
	115k bps	See below
	See above	Parity Enabled, ODD Parity
	See above	Parity Enabled, EVEN Parity
	See above	Parity Disabled

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. RS232-PS2 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 to the far left.)


4 3 2 1 0