Select another site

Specifications Rev. 1.1 11/2002

PS2++, Extended PS/2 Protocol Specification

Introduction

This document summarizes Logitech specific extensions to PS/2 protocol from IBM, called PS2++.

It will ONLY describe the protocol level, NOT the timings NOR low-level specifications NOR driver identification procedures.

Refer to IBM documentation (PS/2 Mouse Architecture Type 1 and 2) for specifications. Standard PS/2 commands are summarized in the PS/2++ Command Summary document.

The Keyboard PS/2 extended protocol (PS2K+) is described in a separate document.

For reference, the normal PS/2 report format is the following:

Byte	D7	D6	D5	D4	D3	D2	D1	D0	Comment
1	Yover	Xover	Ysign	Xsign	Tag	M	R	L	Overflows, signs & switches
2	X7	X6	X5	X4	X3	X2	X1	X0	X data byte (positive East)
3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y data byte (positive North)

Note: The Overflow bits are not checked in our driver and this also the case for other mouse drivers (i.e. MS). In case of Overflow, sending the maximal signed displacement is good practice.

```
The Tag bit is normally 1; the driver also ignores it.
```

PS/2++ Protocol Overview

The PS2++ protocol is defined to be an extension of the PS/2 protocol. This is achieved by using an additional mode in which the device has to be set when it is identified.

The PS2++ protocol has to include the support for up to two PS/2 devices on the same port: internal (built-in) and external. It means, to address each device by commands issued from the host and receive tagged reports from any device connected. The scheme should work on most hardware implementations (pass-through).

Goals

- Compatible with standard PS/2
- Handle multi-devices
- Fast reaction time

- Implement device attention

Constraints

The constraints in extending current protocol are:

- 3 bytes size.
  The protocol should be set in 3 bytes size (report) to be supported by the BIOS and the VKD (Win3.1). Usually the BIOS is set in 3 bytes report and the VKD is called after every 3 bytes (report completed) before generating an event to the driver (callback installed in the BIOS).
- Pass-through limitations.
  Most laptops have already an integrated pointing device but provide an external PS/2 connection. When plugging a device, it is usually polled and its information combined with the internal one before being sent to the driver. The way of handling the external/internal device varies from makers.
- Extendibility.
  The protocol should be extensible to allow future extensions. To achieve this, the protocol must define the general framework and leave the content open for future extensions (encapsulation of data).
- Multiple devices.
  More than one device on a port should be supported. This implies that there is some sort of device multiplexer (pass-through) that can insert information in the report flow.

PS/2++ Commands

The IBM PS/2 Mouse Architecture specification provides a set of "Vendor Unique Commands" (D1h ... DFh) allowing to set a PS/2 device in a certain mode or to get more information from it. Such commands will be used to detect if the device supports PS2++, and then enable it.

We planned to use D1 xx [yy zz ...] as entry level command, xx being the sub command and yy, zz or more, some parameters.

However, such command is not supported by BIOSes, so a workaround has been defined as below:

The **sliced commands** (sausage-like) or **BIOS compatibility mode**:

Each byte (xx, yy or zz) will be coded 2 bits at the time using standard "Set Resolution" command (E8).

To shorten the writing, the following convention is used:

{E8} 0123 is equivalent to E8 0**0** E8 0**1** E8 0**2** E8 0**3**, the mouse answer (FA) is omitted as part of normal protocol behaviour.

To synchronize the commands, two options exist:

Each command between 00 and 3F will start with an E8 00. So the E8 00 is the synchro. (This is always true for Magic Knock: before entering the PS2++ mode)

Any command is accepted between the E8 xx until the Magic Knock has been received.

If the command is higher than 3F or if it has a parameter, E6 is used to separate every packets of 4 E8 (= every byte), only after the Magic Knock has been received.

Note: some pass-through will generate extra E8 when receiving E6: this is not a 100% fullproof solution.

Whenever possible a command should have each par of successive bits different to avoid pass-through filtering (e.g E8 01 E8 01 ... might be forwarded as only one E8 01)

Sliced command scheme is mandatory.

D1 command scheme is only optional (e.g. to make test modes faster), however we use this mode for command description, as it is clearer and simpler.

Enter the PS2++ Mode (Magic Knock)

Due to the structure of the PS/2 protocol, there is no easy way to extend its format. The PS/2 pointing device needs to be set in this PS2++ protocol, if it supports it, by a command sent from the host. This command is the same for internal or external device.

To allow to detect and configure them separately:

- Internal device will send its answer (P0 packet only) within 20..40 ms.

- External device will send its P0 packet (followed by any other relevant packets) within 50..100ms.

The device will send its P0 packet only if enabled (or polled). If it is polled before the time, a zero-displacement report (PN) shall be returned.

Magic Knock: {E8} 0321 3123 or D1 39 DB

Answer: P0 packet, followed by other relevant packets (that can be interleaved by displacement: **event base**)

Tips:

- Any E8 00 must reset the Magic Knock sequence: this is the synchro.

- To make Magic Knock stronger, any command can come between the E8 xxs, as long as the sequence is correct. Nevertheless some limitation on the E6 might be accepted (E6 is supposed to synchronize the packet of 4 E8): once PS2++ mode is entered, E6 is the byte/packet synchro.

The device is expected to acknowledge the PS2++ mode by sending its Device Type (Packet P0, see below) and any relevant information.

The driver can expect any PS2++ packet, at any time, once it has sent the Magic Knock: P0 is the only one that needs to be waited for.

Reset the PS2++ Mode (Magic Unknock)

It is important to be able to return to normal PS/2 reports when giving hand to another driver or when sending commands to a specific device (in case both internal and external are PS2++ able).

This command is called Magic Unknock and it is different for internal and external device.

- External device Magic Unknock: {E8} 0323 or D1 3B

- Internal device Magic Unknock: {E8} 0321 or D1 39

Reset command (FF) will also reset the PS2++ mode (for both devices) as returning to device's default settings.

Note that the beginning of the Magic Knock is the same as internal Unknock. This should allow sending the Unknock to the internal device at the same time as re-enabling the external one.

Sending a Magic Unknock will only revert to standard PS/2 mode: all settings (done by PS2++ commands) are maintained. Only FF (Reset) command will bring the device to its default settings.

Cordless specific commands: D1 20 to D1 2F

Those commands are valid only once the Magic Knock has been received.

Here is a summary for reference.

D1 20 Reserved (Test PS/2 communications)

Answer: FA + P0

D1 21/22 Open Locking on Receiver1/ Receiver2

Answer: FA + P6 (WaitLk=1)

D1 23/24 Close Locking on Receiver1/ Receiver 2

Answer: FA + P6 (WaitLk=0)

D1 25 Reserved (Test PS/2 communications)

Answer: FA + P0

D1 26/27 nn Set Device Num on Receiver1/ Receiver2

nn = device number for next device locked.

Answer: FA + P0

D1 29 p4 p8 p9 Short ID Write

p4 = [P4 data], p8 = [P8 data], p9 = [P9 data]

Answer: FA + P4 + P8 + P9

D1 2A Reserved (Test PS/2 communications)

Answer: FA + P0

D1 2B/2C dd Set Channel on Receiver1/ Receiver2

dd = channel Nr: 0..7 (3 bits)

Answer: FA + P5

D1 2D/2E Get Channel on Receiver1/ Receiver2

Answer: FA + P5

-

Power Management

Those commands are valid only once the Magic Knock has been received.

D1 36 Suspend (Host entering Suspend mode)

= {E8} 0312

Answer: FA (as normal in PS/2 protocol)

Mouse behavior: all motion lower than n (typ. 16) is trashed.

The first motion greater than n is not reported but cancel this mode (same as Resume).

Note: The last byte sent (36 in D1 36 or 02 in {E8} 0312) is acknowledged (FA) to comply PS/2 protocol, then the transmission is stopped. Sending a packet would resume the host...

D1 37 Resume (Host resuming from Suspend mode)

= {E8} 0313

Answer: FA

Mouse behavior: return to normal: report all motion, etc.

D1 38 Sleep (Host entering Sleep mode)

= {E8} 0320

Answer: FA

Mouse behavior: mouse sleeps and is awakened only on button press.

Note: The last byte sent (38 in D1 36 or 00 in {E8} 0320) is acknowledged (FA) to comply PS/2 protocol, then the transmission is stopped. Sending a packet would resume the host...

Enter/Exit PS2++ mode

Those commands have already been described above, they are repeated here to make a complete list.

D1 39 Disable PS2++ mode for **internal** device

= {E8} 0321

Answer: FA

D1 39 DB Enable PS2++ mode (external & internal devices)

{E8} 0321 3123

Answer: FA + P0 + any other relevant packet

See timing constraints below to differentiate external/internal devices

D1 3B Disable PS2++ mode for **external** device

= {E8} 0323

Answer: FA

D1 80 Enter Test Mode

Answer: FA

The test mode will enable D1 1x or specific commands.

Caution: Sliced mode command {E8} 2000 0000 is **not allowed** to enter Test mode!

PS/2++ Packets: Extended Reports

The PS2++ report format is the following:

Byte D7 D6 D5 D4 D3 D2 D1 D0 Comment

1 E ***1*** T3 T2 *1* M R L Packet Type

2 *? ?* T1 T0 c3 c2 c1 c0 Packet Type & checkbits

3 d7 d6 d5 d4 d3 d2 d1 d0 Data byte

Italic *(red)* bits are constants. Note: if T2=0, Byte2-D7&D6=1 and if T2=1, D7&D6=0 (2's complement: PS2++ detection scheme is ABS(X) > 191)

c3..c0: checkbits. They are the copy of 4 data low bits shifted twice: **d1 d0 1 0**. This should filter jitter of +/-1 on both axis but recommended action, in case of checkbits error, is to ignore the packet.

E is 1 if this packet comes from an external device and 0 if from internal

A PS2++ packet is to be considered the same as any normal report:

- Report rate is the same (time between 2 PS2++ packets is the same as between two normal reports or between PS2++ and normal reports)
- Read Data command (EB) gets the same result as if the device was in stream mode: PS2++ packets are reported the same way, in the same order.
- Resend Command (FE) reacts the same way: will resend last packet, standard or PS2++

Consequences on normal reports format (PN)

Since we use overflow bits in PS2++ packets, those bits should never be set in normal reports.

At the same time, the maximal displacement value of a normal report is ±191 for X axis (because of PS2++ detection scheme), but a maximal value of +127/-128 (7bits) is recommended.

This could be expressed as:

PN: Normal PS/2 report when PS2++ mode enabled:

Byte D7 D6 D5 D4 D3 D2 D1 D0 Comment

1 0 0 Ysign Xsign Tag M R L Signs & switches

2 Xsign X6 X5 X4 X3 X2 X1 X0 X data byte (positive East). Range: -128/+127

3 Ysign Y6 Y5 Y4 Y3 Y2 Y1 Y0 Y data byte (**positive North**). Range: -128/+127

P0: Packet for Device Type:

(First report coming after Magic Knock)

Byte D7 D6 D5 D4 D3 D2 D1 D0 Comment

1 E ***1*** 0 0 *1* M R L Packet Type0= Device Type

2 *1 1* 0 0 DT1 DT0 *1 0*

3 DT7 DT6 DT5 DT4 DT3 DT2 DT1 DT0 Device Type

Note: If the device was enabled before the Magic Knock is sent, P0 comes just after the FA acknowledging the last command.

P0 will also come just after the FA of Enable (F4) command or as an answer to first Read Data (EB) command if polled (and disabled)

P1: Packet for Mouse extra info:

(Coming at any change of its information)

Byte	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**Comment**
1	E	***1***	0	0	*1*	M	R	L	Packet Type1= Mouse Info
2	*1*	*1*	0	1	R1	R0	*1*	*0*	
3	RolH	0	B5	B4	RS	R2	R1	R0	Extra Buttons & Roller (3bit+sign)

RolH: 1 if horizontal roller displacement, 0 if vertical (as in Intellimouse)

Note: This packet should be followed by a normal displacement report (PN) to allow synchronization between switch press and displacement. This is not necessary for roller information. In case of short press, PN might not be sent between P1 for press and the 3 P1 for release; this is acceptable.

It is recommended to send one PN between two P1 if a displacement is detected (send P1 & PN alternatively => 1/2 report rate)**.**

It is also recommended to send at least 3 times P1 when a switch is released, to prevent the case of packet being "eaten" by pass-through. In case of ROM/RAM space limitation, repeating it only twice (or once) is however acceptable.

(See for more explanations)

Summary: Switch down: P1, PN, ...

Switch up: P1, P1, P1, PN, ...

(Short Switch: P1, P1, P1, P1, PN, ...)

Switch & displ.: P1, PN, PN, ..., PN, P1 ,P1, P1, PN, ...

Roller change: P1, ...

Roller & displ.: P1, PN, P1, PN, ...

P4: Packet for wireless status: sub-Device Type & DeviceNr

(coming after Magic Knock sequence or at any change of its information)

(also coming before data of "new" device if using multiple devices on same receiver)

Byte	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**Comment**
1	E	***1***	0	1	*1*	M	R	L	Packet Type4= Device descr.
2	*0*	*0*	0	0	sDT1	sDT0	*1*	*0*	
3	sDT5	Nr1	Nr0	sDT4	sDT3	sDT2	sDT1	sDT0	DeviceNr & sub-Device Type

Nr2 is now replaced by sDT5. C-RA2-DUAL and C-RD3-DUAL use this bit Nr2.

For all coming RF products, keyboard devices will then be numbered from 0 to 3 instead of 4 of 7.

P5: Packet for wireless status: Channel & Battery

(coming after Magic Knock sequence or at any change of its information)

Byte	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**Comment**
1	E	***1***	0	1	*1*	M	R	L	Packet Type5= wireless-Status
2	*0*	*0*	0	1	Bat1	Bat0	*1*	*0*	
3	res	CBtn	Ch2	Ch1	Ch0	Bat2	Bat1	Bat0	Button, Channel & Battery

Bat2..0: *0*: Battery level unknown; *1*: critical; *2-4*: low; *5-7*: good

Ch2..0: RF Channel Nr (0 = channel 1, 1 = channel 2, etc)

CBtn: Sent once at 1 if device connect button pressed (used only at MFG)

res: reserved, default to *0*

P6: Packet for wireless status: Notifications

(coming at any change of its information)

Byte	D7	D6	D5	D4	D3	D2	D1	D0	Comment
1	E	*1*	0	1	*1*	M	R	L	Packet Type6= wireless Notification
2	*0*	*0*	1	0	Susp	OpOk	*1*	*0*	
3	LkReq	PwrUp	R1Tx	UnLk	WtLk	R2Tx	Susp	OpOK	Notifications

LkReq: 1 if device is attempting to lock

PwrUp: 1 if device just powered up

R1Tx: 1 if Mouse receiver (=RCV1) is talking

UnLk: 1 if no device is locked

WtLk: 1 if receiver is waiting to lock new device

R2Tx: 1 if Keyboard receiver (=RCV2) is talking

Susp: 1 if suspend/sleep mode is active (D1 36/38 commands)

OpOK: 1 if when request (slow) operation has been completed

P8&P9: Packets for wireless status: Short-ID

(coming after Magic Knock sequence or at any change of its information)

Byte	D7	D6	D5	D4	D3	D2	D1	D0	Comment
1	E	*1*	1	0	*1*	M	R	L	Packet Type8= Short-ID LSB
2	*1*	*1*	0	0	id1	id0	*1*	*0*	
3	id7	id6	id5	id4	id3	id2	id1	id0	Device Short-ID low byte
1	E	*1*	1	0	*1*	M	R	L	Packet Type9= Short-ID MSB
2	*1*	*1*	0	1	id9	id8	*1*	*0*	
3	0	0	0	0	id11	id10	id9	id8	Device Short-ID high byte

PB: Packet for Mouse Capabilities:

(coming after the Magic Knock and after P0)

Byte	D7	D6	D5	D4	D3	D2	D1	D0	Comment
1	E	*1*	**1**	**0**	*1*	M	R	L	Packet TypeB= Mouse Capabilities
2	*1*	*1*	**1**	**1**	Btn1	Btn0	*1*	*0*	
3	0	Shp6	Shp5	Shp4	Shp3	Shp2	Shp1	Shp0	Shape (= Case Number)
3'	1	res	res	Rol1	Rol0	Btn2	Btn1	Btn0	Roller Types & Nr of Buttons

Btn2..0: *000*: 2 buttons, *001*: 3 buttons, *010*: 4 buttons, *011*: 5 buttons, *1xx*: reserved

Rol1..0: *00*: no roller, *01*: vertical roller, *10*: reserved, *11*: vertical & horizontal roller res: reserved, default to *0*

Examples:

For **M-CW47 and BA47 (same** shape):

DT=80 (at E6E6E6E9 command and P0 packet)

with 2 PB packets: E8 FE 2F, E8 FA 8A with their last byte expanded below:

3 0 0 1 0 1 1 1 1 Shape = 47 (0x2F)

3' 1 0 0 0 1 0 1 0 Vert Roller & 4 Buttons

PC&PD: Packets for remote control data

(coming at any change of its information)

Byte D7 D6 D5 D4 D3 D2 D1 D0 Comment

1 E ***1*** 1 1 *1* M R L Packet TypeC= RC data LSB

2 *0 0* 0 0 K1 K0 *1 0*

3 K7 K6 K5 K4 K3 K2 K1 K0 Scan Code or Key Code LSB

1 E ***1*** 1 1 *1* M R L Packet TypeD= RC data MSB

2 *0 0* 0 1 B0/K8 KeyCd *1 0*

3 D 0 0 0 0 B1/K9 B0/K8 KeyCd KeyDown/Bank/Key Code MSB

PF: Packet reserved for future extensions (raw status transmission)

Byte D7 D6 D5 D4 D3 D2 D1 D0 Comment

1 E ***1*** 1 1 *1* M R L Packet TypeC= RC data LSB

2 *0 0* 0 0 K1 K0 *1 0*

3 S7 S6 S5 S4 S3 S2 S1 S0 Raw status

PS/2++ Timings

By philosophy, PS2++ is not timing dependant.

However, the following points must be respected:

PS2++ packets expected, as a "response" to a PS2++ command shall come before a 100ms delay.

Packets sent as answer to Magic Knock shall not be sent before a 20..40ms delay for internal device and before a 50..100ms delay for an external device (like a mouse or a trackball)

Typical timing values are preferably closer to the lower value: to 20 ms for internal devices and to 50ms for external devices.

Note: To avoid confusion with old PS2++ external devices which would send P0 right after the end of Magic Knock, the driver will wait 50ms before enabling the device, thus allowing an Internal device to send its P0 even if disabled (this is an exception to PS/2 protocol). It is however necessary to implement the 50ms delay in the device to make sure the P0 is not eaten by a polling pass-through.

Annexes

Summary of Packet Types

T3	T2	T1	T0	Packet Type
0	0	0	0	P0: Device Type
0	0	0	1	P1: Roller & Buttons
0	0	1	0	P2: Reserved
0	0	1	1	P3: Reserved
0	1	0	0	P4: sub-Device Type (& Device Description)
0	1	0	1	P5: Channel & Battery
0	1	1	0	P6: wireless Notifications
0	1	1	1	P7: Reserved
1	0	0	0	P8: Short-ID LSB
1	0	0	1	P9: Short-ID MSB
1	0	1	0	PA: Reserved
1	0	1	1	PB: Mouse Capabilities (shape, #buttons, roller)
1	1	0	0	PC: Remote controller data LSB
1	1	0	1	PD: Remote controller data MSB
1	1	1	0	PE: Reserved
1	1	1	1	PF: Next packet(s) definition (sub-Type)

Glossary/Term explanation

Device Type (DT) 7 bits to identify the connected device type

Sub-Device Type (sub-DT) 5 bits to identify the cordless device type

Short-ID 12 bits to identify uniquely one cordless device (packet filter)

(eg. 16'0456 in my mouse and 16'0567 in his mouse)

DeviceNr 3 bits to number one cordless device in a list.

(eg. 1=keyboard, 2=mouse#1, 3=mouse#2)

Home Device PS/2++ PS/2 Command PS/2++ Protocol PS/2 Detection M++ MZ++ Serial PNP Firmware Detection PS2K+ USB Mouse

Last Updated: 07/11/2003 10:53:17 PM

Email: info@dqcs.com

Posted Web Usage - Site Links