Active PS/2 Multiplexing

Version 1.1

Created:	1.0	October 6, 1998
Revision:	1.1	May 17, 1999
Printed:		June 1, 1999

Synaptics, Inc.
2702 Orchard Parkway
San Jose, California 95134
(408) 434-0110
Fax (408) 434-9819

Phoenix
Insyde
LCS/Telegraphics

1. Introduction

This document defines a method for attaching up to four PS/2 pointing devices to a computer without sacrificing the enhanced capabilities of any of the devices.

Many computers today include two or more PS/2 pointing devices. For example, there may be a Synaptics TouchPad®, a tracking stick, and an external connector for a mouse. But the traditional PC architecture provides for only one keyboard and one other device such as a mouse. Some method is needed to reconcile these multiple PS/2 devices with the "one mouse" model that system software expects.

Early solutions allowed only one pointing device to be active at a time, at great inconvenience to the user. More recent approaches allow all the pointing devices to be active at once, but these methods interfere with enhanced pointing devices such as the Microsoft IntelliMouse® and the Synaptics TouchPad®. In fact, Microsoft, in its Mobile PC '98 specification, strongly discourages or disallows these more recent approaches because they interfere with the IntelliMouse. Microsoft undoubtedly hopes vendors will address the issue by switching to USB, but it is clear that for pointing devices at least, PS/2 will remain much more practical than USB in the foreseeable future.

The "Active Multiplexing" system described here gives the user maximum freedom and functionality with multiple PS/2 devices, is backward compatible with legacy software and hardware, and is fully compliant with the requirements of PC '98.

Section 2 of this document (page 3) reviews the standard mouse interface and surveys some of the existing ways for supporting multiple pointing devices. Section 3 (page 8) describes the Active Multiplexing system in detail.

Many individuals and organizations contributed to this standard, particularly Phoenix, Insyde, LCS/Telegraphics, and IBM. See page 27 for a full list of acknowledgments.

Table of Contents:

2. Background

The next few sections provide useful background information on the existing state of the art in keyboard controllers and PS/2 pointing devices.

2.1. Background: Traditional KBC interface

In a standard PC, the keyboard controller (KBC) is responsible for communicating with the keyboard and with a second "auxiliary" or AUX device. The AUX device is often a mouse or other pointing device but may be any device that obeys the mouse-style PS/2 communication protocol. The port connecting to the AUX device is called the Auxiliary Device Port or ADP. Here for reference is a summary of the host interface that traditional keyboard controllers provide for talking to AUX devices on the ADP.

The host talks to the KBC via I/O port 60h (for data) and I/O port 64h (for commands and status). For reading, input port 64h has the following layout:

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
Parity	Timeout	AUX	—	—	Sys Flag	IBF	OBF

Figure 1. Traditional input port 64h.

The OBF bit is '1' if there is data available in input port 60h. This may be input from the keyboard or the AUX device, or an internally generated response to a KBC command. OBF is '0' if no data is available. OBF is set to '1' when new data arrives, and cleared to '0' automatically when the host reads input port 60h.

The IBF bit is '0' if the KBC is ready to accept a new byte from the host on either output port 60h or output port 64h. IBF is '1' if these ports are still busy. IBF is set to '1' when the host writes to output port 60h or output port 64h, and cleared to '0' when the KBC accepts the command or data.

The AUX bit is '1' if the data on input port 60h came from the AUX port, or '0' if the data came from the keyboard or from the KBC itself. The AUX bit is valid when OBF is '1'. (At other times the AUX bit is "undefined"; in other words, what the AUX bit does when OBF is '0' may vary from one KBC to another. Typically, the KBC will leave the AUX bit alone until shortly before it next sets OBF to '1', but driver software must not count on this.)

The Timeout bit is '1' if a timeout error occurred in the last attempted communication with the keyboard or AUX device; input port 60h typically contains FEh or FFh. The Timeout bit is valid when OBF is '1'. A timeout error occurs when the KBC attempts to send a byte to the device but the device does not respond, either because the device does not respond to the Request to Send condition on the bus, or because the device clocks in the byte but does not return an acknowledgement byte in a timely fashion.

The Parity bit is '1' if a parity error was detected on the last byte received from the keyboard or AUX device; input port 60h typically contains FFh. The Parity bit is valid when OBF is '1'.

Bit 2, the System Flag bit, reports the current state of the System Flag (see Figure 2).

The bits shown as "—" in Figure 1 are not relevant to keyboard or mouse communication.

Flow control on PS/2 communication is automatic. The IBF bit prevents the host from writing bytes to output port 60h faster than they can be transmitted to the device. Similarly, when the device transmits a byte, the KBC holds the device inhibited (with the CLK line low) until the host reads the byte from input port 60h.

The KBC raises IRQ 1 when a byte of data is available from the keyboard (i.e., when OBF is '1' and AUX is '0'). The KBC raises IRQ 12 when a byte of data is available from the AUX port (i.e., when OBF is '1' and AUX is '1'). If both the AUX device and the keyboard try to send bytes, the KBC handles one complete transaction before beginning the next (at least, from the point of view of the host).

Bytes written to output port 64h are interpreted as KBC commands. Bytes written to output port 60h are sent to the keyboard by default, but various KBC commands cause the next byte written to output port 60h to be interpreted as an argument to the KBC command.

The KBC commands of relevance to the AUX port are:

A7h	Disable the AUX port by holding the CLK line low.
A8h	Enable the AUX port by releasing the CLK line.
A9h	Test the AUX port.
D3h	Treat the next byte written to port 60h as if it had been received from the AUX device. (Echo the byte on input port 60h, with the Parity and Timeout bits '0', the OBF and AUX bits '1', and IRQ 12 raised.)
D4h	Send the next byte written to port 60h to the AUX device instead of to the keyboard.

There is also a "KBC command byte" read by command 20h and written by command 60h.

Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
—	—	AUX Dis	—	—	Sys Flag	En IRQ12	—

Figure 2. KBC command byte.

Bit 5 of the command byte is the AUX Disable flag, which controls whether the AUX port is inhibited by holding its CLK line low. The A7h command works by setting AUX Disable to '1'; the A8h command clears AUX Disable to '0'. The AUX port can also be disabled or enabled by using command 60h to write directly to the command byte.

Bit 2 of the command byte is the System Flag, which is reported continuously in bit 2 of input port 64h. In typical usage, this bit is set to '1' during the boot process and is unused after that point. (The System Flag is typically used by the BIOS to distinguish between cold and warm boots. The bit powers up as '0', then is set to '1' once the

system is booted; it is never cleared to ‘0’ except possibly at shutdown time to simulate a cold reset.)

Bit 1 of the command byte controls whether IRQ 12, the AUX port interrupt, is enabled or disabled.

Other bits shown as “—” in Figure 2 are not relevant to this specification.

The AUX port is often used for a mouse or a mouse-compatible pointing device. While the AUX port was intended as a general interface not restricted to pointing devices, its use as a mouse port is so prevalent that some existing KBCs have developed algorithms which tacitly assume the AUX device is a mouse. For example, the algorithms discussed in section 2.2 below assume that the data received from the AUX device takes the form of mouse motion packets.

When a standard mouse is in fact used on the AUX port, its standard data packet is arranged as follows:

	Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
Byte 1	X ovfl	Y ovfl	X sign	Y sign	1	Middle	Right	Left
Byte 2	X delta							
Byte 3	Y delta							

Figure 3. Traditional mouse packet.

The Left, Middle, and Right bits report the state of the mouse buttons. Bit 3 of byte 1 is typically always ‘1’ in every mouse packet; in some mice, this bit is always ‘0’. The remaining bits and bytes encode the amount of X and Y mouse motion.

Some advanced pointing devices support alternate packet formats. The Microsoft IntelliMouse®, for example, can expand the packet to four bytes by adding a “wheel” byte. The Synaptics TouchPad® can switch to Absolute mode with six-byte packets arranged in a completely different format. Some AUX devices are not pointing devices at all; their usage of the AUX port may be very different from that of a mouse.

For more information on the PS/2 protocol used for communication with AUX devices, and on the command set supported by a typical AUX pointing device, consult the *Synaptics TouchPad Interfacing Guide*. This publication is available from Synaptics or via the Internet at http://www.synaptics.com/drivers/.

2.2. Background: Hidden multiplexing

The protocol described in the previous section assumes there is only one AUX mouse in the system. Because this one-mouse architecture has become entrenched in PC drivers and operating systems, new designs that support multiple, simultaneously active pointing devices typically must maintain the fiction of a single mouse. Commands sent to “the mouse” are broadcast to all attached devices; packets received from all devices are merged into a single packet stream from “the mouse.” The various PS/2 devices are effectively “multiplexed” into a single device as seen by host software. In this document

we refer to this method as “hidden multiplexing” since the multiplexing is completely hidden from higher-level driver software.

Existing schemes for hidden multiplexing vary, but they generally operate as follows:

Every byte sent to the mouse using KBC command D4h is broadcast to all PS/2 devices. The response from one preferred device is forwarded back to the host. For example, if the host sends a Status (E9h) command to “the mouse,” only the status response from the preferred device is forwarded to the host.

Every packet received from any device is forwarded to the host. Input is interleaved at the level of three-byte packets, not at the byte level. For example, if one device sends byte 1 of its packet, and then a second device sends byte 1 of its packet, the KBC will not simply send those two bytes to the host as they arrive. If it did, the host would interpret the second device’s byte 1 as byte 2 of a packet from the same mouse. Instead, the KBC either buffers bytes up and forwards whole packets at a time, or it inhibits the other devices when the first device is midway through transmitting its packet.

The KBC must do some limited editing of the packet stream in order to implement hidden multiplexing correctly. Suppose there are two mice attached, one with the Left button held down and the other with all the buttons released. Suppose the user moves both mice simultaneously. The two mice will send packet streams to the host, which the KBC will interleave at the packet level. Because packets from the first mouse, with the Left button bit set, are interleaved with packets from the second mouse, with the Left bit clear, the host will perceive rapid repeated clicking of the Left button. To avoid this, hidden multiplexers typically remember the button bits from the last packet received from each device, and replace the button bits of every packet with the logical OR of all devices’ most recent button bits.

Some hidden multiplexers also edit bit 3 of byte 1 of the packet, for example, forcing this bit to ‘0’ in packets from one device and to ‘1’ in packets from the other device. This information allows a knowledgeable driver to separate the two packet streams in software.

In order to interleave packets and to edit the button bits, hidden-multiplexing KBCs must assume that every packet is exactly three bytes long. These KBCs are confused by the enhanced packets of pointing devices like the Synaptics TouchPad® and the Microsoft IntelliMouse®. Naturally, these KBCs are even more confused by AUX devices which are not pointing devices at all.

In the solution described in this document, all bytes received from the various devices are forwarded to the driver as they arrive, and each byte is tagged to identify its source. The driver can divide this byte stream into packets based on its complete knowledge of the enhanced features and protocol of each device.

2.3. Background: Hot-plugging

The PS/2 mouse specification states that at power-up, a PS/2 mouse must send a special announcement consisting of the bytes AAh and 00h. The mouse then wakes up in the Disabled state, in which it does not spontaneously send packets until the host sends an

F4h (Enable) command. Hence, if the user "hot-plugs" a mouse (i.e., attaches a mouse to the external connector with the computer already running), the mouse will not work until the driver sends an Enable command, which may require restarting the operating system or even power-cycling the computer.

Some KBCs attempt to support hot-plugging by noticing when they receive AAh and 00h preceded and followed by silence. Since normal packets are three bytes long, this input can only mean that a new mouse has been hot-plugged onto the port. The KBC responds by sending a known sequence of PS/2 commands to reinitialize and re-Enable the mouse. The KBC also absorbs the AAh, 00h bytes without passing them on to the host. This scheme allows mice to be hot-plugged even when a legacy driver is used.

(Another method of hot-plugging is to check for AAh, 00h as the first two bytes of a packet; while not impossible, it is extremely unlikely for a legitimate packet to begin with these two bytes. In this method, the KBC must count the received bytes to keep track of packet boundaries, just as in hidden multiplexing. Hence, this method is especially likely to be confused by non-standard packet lengths and formats.)

KBC hot-plugging causes serious problems when an enhanced pointing device with an enhanced driver is used. The KBC attempts to reinitialize the mouse to its previous state, but the KBC probably does not know the non-standard commands to reactivate the device's enhanced mode. In fact, the newly hot-plugged device may not support the same enhanced modes as the previous device. From the point of view of the driver, the device spontaneously and without notice reverts to its non-enhanced mode, possibly changing the size of its packets. The driver continues to interpret the data stream in terms of enhanced-mode packets, with disastrous consequences.

Also, because the KBC has absorbed the AAh, 00h announcement to keep from confusing legacy drivers, enhanced drivers have no way to detect hot-plugging and reinitialize the device properly. The KBC's attempts to provide hot-plugging have actually *prevented* the enhanced driver from handling hot-plugging properly.

A similar problem arises when the KBC attempts to handle suspend/resume events on its own. After a suspend which involved powering down the mouse, the KBC may attempt to reinitialize the mouse to its known state from before the suspend, again with bad consequences for enhanced pointing devices.

In the solution described in this document, when the KBC notes that an enhanced driver is in use, the KBC allows the driver to handle hot-plugging and suspend/resume with no interference from the KBC.

3. Active PS/2 Multiplexing Specification

The solution to all the problems described in the preceding sections is to use "Active Multiplexing." In an Active Multiplexing system, the driver actively participates in operating all the various pointing devices and interpreting the packets from the devices. Because Active Multiplexing is incompatible with legacy drivers, the KBC must also support a legacy or compatibility mode.

The following sections define a method for Active Multiplexing of up to four auxiliary devices on a computer. The four devices are identified as "AUX port #0," "AUX port #1," "AUX port #2," and "AUX port #3." Not all four ports need to be present on every computer. Each AUX port connects to one mouse or other pointing device, or indeed to any device such as a bar code reader or smart card reader which supports the mouse-style PS/2 protocol.

The Active Multiplexing solution is implemented partially in KBC firmware, and partially in driver software. No modifications to the pointing devices themselves are required.

3.1. Overview

Active Multiplexing involves the following features:

- Knowledgeable drivers can switch the KBC into a special Multiplexed mode that is slightly incompatible with the traditional mouse interface.
- Multiplexed mode allows commands to be directed to individual devices using "routing prefixes."
- Multiplexed mode reports input from all devices unedited, with each byte tagged to identify its source.
- Multiplexed mode allows the driver to handle mouse management duties like hot-plugging and suspend/resume.

The net result is that each AUX device communicates with its driver as if it had an independent dedicated channel (subject to a total bandwidth limit).

3.2. Activation and Deactivation sequences

Before using any of the features described in this specification, a mouse driver must first check that the KBC supports these features. To this end, the KBC recognizes a special "Activation sequence" which also serves as a feature identification sequence.

To identify the multiplexing capability, the host writes D3h to output port 64h, then F0h to output port 60h. The result will be that F0h appears on input port 60h as if from a mouse. Then the host writes D3h to port 64h and 56h to port 60h, receiving a 56h back from port 60h. Finally, the host writes D3h to port 64h and A4h to port 60h. If the KBC supports the protocol described here, it replies with a version number byte instead of the expected A4h. The version number byte identifies the version of the Active Multiplexing standard that is implemented; the value *xy*h indicates version *x.y* of this document. For

example, the byte 10h indicates version 1.0. (Version number 10.4, corresponding to the byte A4h, is disallowed.)

> *Rationale: There is no well-established standard method in existing practice for identifying special KBC features. The D3h mechanism outlined here was chosen to have minimal chance of colliding with identification mechanisms of other controllers, or of accidentally triggering non-standard commands of other controllers.*
>
> *The sequence of bytes to be passed to D3h should not coincide with a plausible transmission from a mouse, since software might use D3h to impersonate transmissions from a real mouse. F0h, 56h, A4h was chosen to be impossible as either a Status response or a three-byte excerpt from a packet stream from a standard mouse.*

The rule for parsing the Activation sequence can be expressed more precisely as a state machine, as shown in the following diagram (Figure 4). Each D3h command causes a transition to the next state. KBC commands other than D3h do not affect the state of the parser, with the possible exception of system reset commands like AAh and FEh. The D3h command edits its response to *xy*h only in the situation shown by the heavy arrow in Figure 4; in all other cases, D3h echoes its argument unchanged as in a traditional KBC.

The Activation sequence also causes the KBC to switch into Multiplexed mode, as described in section 3.3.3. If the KBC is already in Multiplexed mode, its mode is unaffected (but the final response is still edited to *xy*h, and the Port Disable flags are still cleared as discussed in section 3.7). It is implementation-defined whether the KBC switches from Legacy to Multiplexed mode before or after the *xy*h byte is sent.

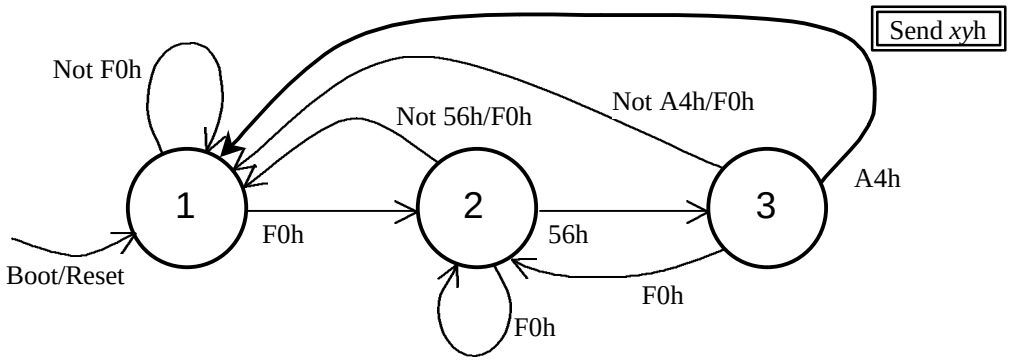

Figure 4. Parsing the Activation sequence.

There is a second sequence, the "Deactivation sequence," which consists of three D3h commands with arguments of F0h, 56h, and A5h, respectively. The KBC edits the response to the final D3h command to the version number *xy*h to acknowledge that the Deactivation command was recognized. The KBC also switches to Legacy mode as described in section 3.3.3. If the KBC is already in Legacy mode, its mode is unaffected (but the final response is still edited to *xy*h).

3.3. KBC modes

The KBC has two operating modes: Legacy mode, and Multiplexed mode.

3.3.1. Legacy mode

Legacy mode is for use with "legacy" mouse drivers, i.e., older drivers which do not understand Active Multiplexing.

In Legacy mode, the KBC operates as described above in section 2.1. This specification does not define the behavior of Legacy mode when multiple pointing devices are attached. Reasonable approaches include, but are not limited to:

1. Support only one preferred port chosen at BIOS setup time; inhibit and ignore all other ports.
2. Choose the highest-priority port with a device attached at boot time; inhibit and ignore all other ports.
3. Choose the highest-priority port dynamically. If the highest-priority mouse is unplugged, or if a new higher-priority mouse is hot-plugged, switch to and initialize the new highest-priority device as described above in section 2.3.
4. Use hidden multiplexing as outlined above in section 2.2, possibly with dynamic hot-plugging as outlined in section 2.3.

While any of these strategies (or any other) is acceptable with respect to this specification, Microsoft appears to recommend strategy #2 and to forbid strategy #4 as default behaviors in the latest PC '98 specifications.

3.3.2. Multiplexed mode

In Multiplexed mode, various aspects of the AUX port interface change as described in the following sections. Briefly, the prefix used to send commands to the AUX device is changed, certain bits of input port 64h are redefined, the concept of enabling/disabling the AUX port is broadened, and the response to hot-plugging may change.

Multiplexed mode affects only the AUX or "mouse" interface. The keyboard interface, including all aspects of ports 60h and 64h that would be used by a keyboard driver, is unchanged in Multiplexed mode.

3.3.3. Transitions between Legacy and Multiplexed modes

The KBC powers up in Legacy mode, and remains in Legacy mode unless/until the host driver issues the Activation sequence.

The KBC enters Multiplexed mode whenever the host issues an Activation sequence (section 3.2).

The KBC reverts to Legacy mode whenever the host issues a Deactivation sequence, and also at system boot time and when the host issues one of the certain KBC commands described in section 3.8. (The latter commands will typically occur when a new legacy

mouse driver uses D4h to send the first command of its initialization sequence to the mouse.)

3.4. Routing prefixes

In a traditional KBC, the command D4h is used to send commands or data to the AUX device; D4h can be viewed as a prefix to a mouse or AUX device command. Active Multiplexing introduces four new prefix commands called "routing prefixes" for talking to AUX devices:

90h Route a command to AUX port #0.

91h Route a command to AUX port #1.

92h Route a command to AUX port #2.

93h Route a command to AUX port #3.

Rationale: The codes 90h, 91h, 92h, and 93h were chosen so as not to coincide with codes already defined in existing KBCs by major vendors such as Phoenix and SystemSoft.

The routing prefix (90h, 91h, 92h, or 93h) is written to output port 64h, followed by any of the following host actions:

Routing prefix + output a byte to port 60h

This command sends the indicated byte to the AUX port selected by the routing prefix. The acknowledgement, timeout error, or other response from the command is passed back to the host by the mechanisms described below in section 3.5.

Note that a routing prefix plus output to port 60h performs the same actions on the selected port as a D4h prefix plus output to port 60h would to the single mouse port in a traditional one-mouse system.

Bytes sent to an AUX port are generally either command codes for the AUX device or arguments to previous commands. While most AUX devices only support a restricted set of commands and arguments, the KBC must forward any 8-bit byte to the AUX device without editing or restriction in Multiplexed mode.

Routing prefix + output A7h to port 64h

This command disables the selected AUX port by holding that port's CLK line low. See section 3.7 below.

Routing prefix + output A8h to port 64h

This command enables the selected AUX port by releasing that port's CLK line. See section 3.7 below.

The effect of a routing prefix *not* immediately followed by one of the host actions listed above is undefined. (A reasonable behavior is for the routing prefix to be ignored.)

3.4.1. Routing prefixes in Legacy mode

In Legacy mode, a routing prefix followed by output to port 60h causes the KBC to reply with FFh as if by a D3h command with an argument of FFh. (Thus, IRQ 12 is raised, input port 60h contains FFh, and, in input port 64h, the Parity and Timeout bits are '0' and the System Flag, AUX, and OBF bits are '1'.) The value written to output port 60h is ignored.

> *Rationale: If the KBC reverts to Legacy mode from Multiplexed mode, this rule allows a multiplexing driver to detect the reversion. The multiplexing driver will interpret bit 2 as an Error bit instead of a System Flag, and the value FFh as a Parity error code. As part of processing this error, the driver can query the KBC to check for Legacy mode. See section 3.9 for details.*
>
> *Another plausible behavior would be for a routing prefix to cause a switch from Legacy mode to Multiplexed mode. Unfortunately, to do so would cause trouble with legacy drivers that query for features of particular KBCs by trying commands specific to those KBCs. For example, if a brand "X" KBC defined a command 91h, then a driver might issue command 91h to check for the presence of a brand "X" KBC. When the driver failed to see the brand "X" response to command 91h, the driver would go on to operate the KBC in the traditional way and would be confused by a switch to Multiplexed mode.*

A routing prefix in Legacy mode followed by an A7h or A8h command may set or clear the selected Port Disable bit (as if the KBC were in Multiplexed mode), or it may set or clear the AUX Disable bit (as if the routing prefix were omitted), or it may have no effect.

> *Rationale: Because commands A7h and A8h elicit no response from the KBC, there is no way to arrange for the driver to detect reversion to Legacy mode on these commands. But most interactions that involve prefixed A7h or A8h commands will also involve prefixed output to port 60h, so the prefixed A7h and A8h commands merely need to behave in a way that is "harmless." All of the allowed behaviors are easy to implement and harmless in this context.*

As in Multiplexed mode, the effect of a routing prefix in Legacy mode followed by any other action is undefined.

Some existing KBCs may already use command codes 90h–93h for unrelated commands. When Active Multiplexing is implemented on these KBCs, it is permissible, but not recommended, for the conflicting commands to retain their original meanings in Legacy mode. However, even in this case commands 90h–93h must produce a response of some kind on input ports 60h and 64h.

3.5. Input from auxiliary devices

3.5.1. Interpretation of port 64h

AUX devices send several kinds of transmissions to the host:

- The AAh, 00h announcement when power is applied to the device.
- The FAh, FEh, or FCh acknowledgement to a command.
- Additional response bytes depending on the command, e.g., three bytes for an E9h (Status) command.
- Streaming (unsolicited) data, e.g., mouse packets in the case of AUX pointing devices.
- Any other kinds of transmissions, especially in the case of AUX devices which are not pointing devices.

In Multiplexed mode, the KBC passes all of these transmissions on to the host without significant delay or editing of any kind. To identify the source of each byte, the KBC redefines certain bits of input port 64h in Multiplexed mode.

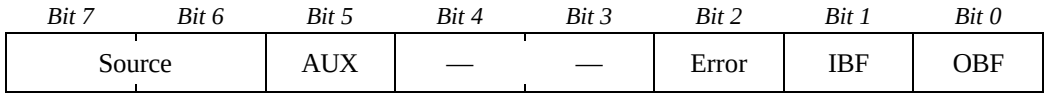

Figure 5. Input port 64h in Multiplexed mode.

When the AUX bit and OBF bit of input port 64h are both '1' in Multiplexed mode, the other bits of that port are redefined as shown in Figure 5. (When the AUX bit and/or the OBF bit of input port 64h are '0', then the other bits of that port are defined in the traditional legacy way as shown in Figure 1, even in Multiplexed mode.)

The original Parity and Timeout bits become a two-bit field called Source. This field identifies which AUX port produced the byte:

Source bits	*Port*
00	Port #0
01	Port #1
10	Port #2
11	Port #3

Figure 6. Source bits of input port 64h.

When several AUX devices wish to send data at once, the order in which the KBC services them is undefined. The KBC should not allow one device to monopolize the port for long periods to the exclusion of other devices which may also have data to transmit.

> *Rationale: The non-monopolization requirement is often satisfied automatically because many AUX devices do not transmit continuously without at least some delay between packets. However, the KBC should*

not count on this fact—for example, two mice reporting at once could keep the port busy continuously, and should not be allowed to exclude packets from a third device.

Bit 2 of input port 64h, originally the System Flag bit, becomes an Error bit when AUX and OBF are both '1'. In Legacy mode, the Error bit is always reported as '1' if the KBC has been in Multiplexed mode since the last system reset or Deactivation sequence. (This rule allows the host to detect reversion to Legacy mode; see section 3.9.1.) The implementation may choose to force the Error bit to '1' at all times in Legacy mode, or it may distinguish between a "true Legacy mode" where bit 2 is the System Flag and a "reverted Legacy mode" where bit 2 is forced to '1'.

In Multiplexed mode, the Error bit works as described in section 3.5.2. Figure 7 summarizes the various interpretations of this bit.

OBF bit	*AUX bit*	*Current mode*	*Contents of bit 2*
0	*Any*	*Any*	System Flag
1	0	*Any*	System Flag
1	1	Legacy	Always '1'
1	1	Multiplexed	Error bit

Figure 7. Bit 2 of input port 64h.

Rationale: Requiring bit 2 of port 64h to be forced to '1' when AUX and OBF are both set, even in Legacy mode, introduces a minor incompatibility. The incompatibility is minor since the traditional usage of bit 2 normally holds this bit at '1' anyway. Requiring the KBC to force this bit to '1' when AUX and OBF are both set allows for a rigorous way to detect reversion to Legacy mode, as described in section 3.9.1.

Note that bit 2 changes from System Flag to Error bit whenever AUX and OBF are both set to '1', even as a result of a D3h command.

Once the host reads the data byte from input port 60h, the OBF bit clears automatically to '0'. After OBF is cleared, the Source, AUX, and Error bits become undefined until the next time OBF is set to '1'. This specification recommends that bit 2 revert to reporting the System Flag when OBF is '0'. Bit 2 does not need to change between System Flag and Error bit instantaneously as OBF changes, but it should change from System Flag to Error bit not long before OBF changes to '1', and it should change from Error bit to System Flag soon after OBF changes to '0'.

Rationale: The phrases "not long before" and "soon after" are deliberately vague. In some KBC hardware it may be impractical to guarantee instantaneous role switching of bit 2, but the switch should be reasonably prompt for the benefit of system software that uses the System Flag. In practice, only boot software uses the System Flag, so the transition does not have to be extremely prompt.

While this specification recommends that bit 2 revert to the System Flag when OBF is '0', this is not required. The behavior of bit 2 when OBF is '0' or AUX is '0' is described above as "undefined" because it is outside the scope of this specification.

If the system can be relied upon never to examine the System Flag after boot time, another acceptable implementation would be to delay the reversion of bit 2 to System Flag until the next time OBF is '1' and AUX is '0'.

3.5.2. Reporting errors

In Multiplexed mode, the old Parity and Timeout bits have been replaced by the Source bits. Instead, the Error bit of input port 64h is used to report parity and timeout errors.

If the Error bit is '0', input port 60h contains valid data received from the port indicated by the Source bits.

If the Error bit is '1', there was a transmission error on the port indicated by the Source bits; input port 60h will contain either FFh to report a Parity error or FEh to report a Timeout error. Input port 60h may also contain FDh to report a hot-removal if supported; see section 3.11. Input port 60h will never contain any value except for FDh, FEh or FFh when Error is '1'.

In circumstances where Legacy mode would report both Timeout = '1' and Parity = '1', the error code is FFh in Multiplexed mode (i.e., the Parity error takes precedence).

Rationale: Even though some drivers ignore the Parity and Timeout bits of input port 64h, it is good to report some kind of Error bit for the benefit of careful drivers.

Of the bits of input port 64h, bit 2 is the only one that is firmware-controllable and not occupied with an important function. The legacy function of bit 2 (System Flag) is not likely to interfere with the new meaning as an Error bit, though there is no guarantee of this. See sections 3.5.1 and 3.9 for further discussion of bit 2.

3.5.3. Responses to commands

Every byte that is sent to an AUX device elicits at least one byte of response, typically beginning with an Acknowledge (FAh) byte. If the device responds properly, this first response byte will be reported to the host just as any other input byte from that port. If the device does not respond, the KBC will signal a timeout by setting the OBF, AUX, and Error bits to '1' and the Source bits to indicate the port number, and by putting FEh in input port 60h. To avoid a timeout, an AUX device must respond to each byte within about 25 milliseconds. The KBC is not required to detect the timeout after exactly 25 milliseconds, but it should be reasonably prompt in reporting timeouts.

While waiting for the acknowledgement or timeout response to a byte sent to a given port, the host must not send another byte to the same port. However, the host may send

bytes to other ports, and acknowledgements and other inputs from the various ports may be interleaved in any fashion.

> *Rationale: This means that all four ports could potentially have pending timeouts at once, presumably detected by four parallel software timers in the KBC. (Note that each port must be able to report a timeout even while other ports are experiencing heavy command activity with no timeouts.)*
>
> *Drivers may wish to poll for an acknowledgement byte without being interrupted by input from other ports. Such drivers can use command A7h with routing prefixes to disable all ports except the one to which they are sending the command, as described in section 3.7.*

3.5.4. Demultiplexers

IRQ 12 is shared among all AUX ports in Multiplexed mode; hence, the operating system must install an interrupt handler on the IRQ 12 vector that understands how to read the Source bits and direct the interrupt accordingly. The easiest approach is for a single driver to operate all the attached AUX devices, using a single handler on IRQ 12. Thus the single-PS/2-mouse model is preserved except within the mouse driver itself. (In this document, AUX drivers are referred to generally as "mouse drivers" even though in practice they may also operate non-mouse AUX devices.)

It is also possible to use a separate demultiplexing IRQ 12 handler which coordinates several distinct PS/2 mouse drivers. Because the demultiplexing handler needs to examine only input port 64h, and not actually read the byte from input port 60h, the demultiplexing handler can jump to a legacy interrupt handler in a legacy driver if desired. This works only if the legacy driver's interrupt handler ignores the Parity and Timeout bits of input port 64h; fortunately, many existing mouse drivers do ignore these bits. (Note, however, that legacy code which sends commands to the mouse is not reusable, since it must be re-expressed in terms of routing prefixes instead of D4h commands.)

3.6. Recommended port assignments

This specification does not require any particular correspondence between AUX port numbers (#0, #1, #2, and #3) and physical types of devices. Also, if only one, two, or three ports are implemented, this specification does not require any particular choice of active port numbers.

However, to encourage consistency among implementations, this specification *recommends* the following assignments for use in typical situations. When one internal and one external pointing device are used:

Port	*Recommended usage*
Port #0	Unused.
Port #1	Internal pointing device.
Port #2	Unused.
Port #3	External pointing device.

Figure 8. Recommended port assignments (two devices).

When two internal devices (e.g., a TouchPad and another type of integrated device) and one external pointing device are used:

Port	*Recommended usage*
Port #0	Unused.
Port #1	Internal TouchPad pointing device.
Port #2	Internal other pointing device.
Port #3	External pointing device.

Figure 9. Recommended port assignments (three devices).

Rationale: Port #0 is left open because the D3h (Force Mouse Output) command simulates input from port #0. Having a real device at this address would create a potentially confusing ambiguity.

Some systems include two external PS/2 ports, one for a keyboard and one for a mouse. These ports could be assigned distinct AUX port numbers in order to allow the user to attach mice or other AUX devices to both ports at once. Or, the system can assume that at most one port will have a mouse attached, reserve a single AUX port number (port #3 in the above examples) for the external mouse, and dynamically assign to that port number the physical port that is observed to hold a mouse.

Some systems also include both an external AUX port and USB ports to which USB mice can be attached. The system must support "legacy" operation of the USB mouse, in which the USB mouse is disguised as a PS/2 mouse for the benefit of operating systems which do not support USB directly. It is possible that a multiplexing-aware driver could run on such an operating system. In this case, two reasonable implementations would be for the legacy USB mouse to be assigned its own distinct AUX port number, or for the USB mouse and external PS/2 mouse to share the same AUX port number (e.g., port #3) on a priority basis. (In practice, it is expected to be rare for a KBC to operate in Multiplexed mode when the operating system does not support USB mice directly.)

3.7. Enabling and disabling ports

In Legacy mode, there is a single AUX Disable flag (bit 5 of the KBC command byte) which enables or disables the AUX port(s). In Multiplexed mode, this flag continues to exist, but there are four additional Port Disable flags for separately enabling or disabling the individual AUX ports.

For example, AUX port #1 is enabled (CLK line free) if AUX Disable is ‘0’ and Port #1 Disable is also ‘0’. AUX port #1 is disabled (CLK line held low) if either AUX Disable is ‘1’ or Port #1 Disable is ‘1’. Likewise for ports #0, #2, and #3, as shown in Figure 10.

Port	*Enabled if…*
Port #0	AUX Disable = 0 **and** Port #0 Disable = 0
Port #1	AUX Disable = 0 **and** Port #1 Disable = 0
Port #2	AUX Disable = 0 **and** Port #2 Disable = 0
Port #3	AUX Disable = 0 **and** Port #3 Disable = 0

Figure 10. Interpretation of AUX and Port Disable flags.

In Multiplexed mode, the A7h and A8h commands preceded by a routing prefix (section 3.4) set the corresponding Port Disable flag to ‘1’ or ‘0’ respectively. An unprefixed A7h or A8h sets the shared AUX Disable flag to ‘1’ or ‘0’ respectively.

Many KBC implementations clear the AUX Disable flag automatically when a D4h prefix is used to send data to the device; this is done in order to allow the KBC to receive the acknowledge byte from the device. In Multiplexed mode, these KBCs would clear both the AUX Disable flag and the appropriate Port Disable flag when a routing prefix is used to send a byte to a device.

When the KBC recognizes the Activation sequence (section 3.2), it initializes all four Port Disable flags to ‘0’. Note that the Port Disable flags are not readable or settable while in Legacy mode, so the KBC may set them all to ‘0’ or to any other convenient values, or not implement them at all, while in Legacy mode.

The operation of the KBC command byte is unchanged in Multiplexed mode; bit 5 continues to access the AUX Disable flag during reads and writes to the command byte. (Also, bit 1 continues to serve as an IRQ 12 enable flag, which is unmultiplexed since all AUX ports share the IRQ 12 vector.)

There is no way for the host to read back the individual Port Disable flags.

KBC command	*Effect on various Disable flags*
A7h	Set AUX Disable = 1
A8h	Set AUX Disable = 0
90h A7h	Set Port #0 Disable = 1
90h A8h	Set Port #0 Disable = 0
91h A7h	Set Port #1 Disable = 1
91h A8h	Set Port #1 Disable = 0
92h A7h	Set Port #2 Disable = 1
92h A8h	Set Port #2 Disable = 0
93h A7h	Set Port #3 Disable = 1
93h A8h	Set Port #3 Disable = 0
20h	Read AUX Disable flag in bit 5

KBC command	*Effect on various Disable flags*
60h	Set AUX Disable according to bit 5
Legacy mode	Set Port #0, #1, #2, #3 Disables arbitrarily
Activation sequence	Set Port #0, #1, #2, #3 Disables = 0

Figure 11. Effects on AUX and Port Disable flags.

Rationale: It would appear to be simpler to dispense with the AUX Disable flag in Multiplexed mode, and simply to have the unprefixed A7h and A8h commands set or clear all Port Disable flags together. However, such a definition would run afoul of certain non-mouse-driver software (such as the keyboard driver) which occasionally disables the AUX port and then re-enables the port before returning control to the mouse driver. It is unreasonable to expect non-mouse drivers to be aware of active mouse multiplexing, but if A8h is defined to re-enable all AUX ports, then these non-mouse drivers could accidentally disrupt the state maintained by the multiplexing mouse driver.

Using the definition here, a non-mouse driver that issues command A7h will disable all AUX ports as intended; then, when the non-mouse driver issues command A8h, only the AUX ports that were enabled before by the multiplexing driver will be re-enabled.

Also, some non-mouse drivers enable and disable the AUX port by writing directly to the KBC command byte. If there were no global AUX Disable flag in Multiplexed mode, then it would be impossible for the KBC to tell whether or not a write to the KBC command byte was meant to change the enable/disable state of the AUX port.

3.8. KBC commands

Most KBC commands are not affected by, and do not affect, Multiplexed mode. But certain KBC commands cause the KBC to revert to Legacy mode if executed in Multiplexed mode. After reverting to Legacy mode, the KBC can process the command in the normal (legacy) manner according to the guidelines of section 3.3.1. Once in Legacy mode, the KBC remains in Legacy mode until the host executes another Activation sequence (section 3.3.3).

This section lists the standard KBC commands which have special interactions with Multiplexed mode.

A6h Enable Password. This command inhibits most KBC operations until a password is typed on the keyboard. In systems with Active Multiplexing, the password inhibits all the AUX devices in the obvious way.

A7h Disable AUX Port: See section 3.7.

A8h Enable AUX Port: See section 3.7.

A9h The Test AUX Port command in Multiplexed mode returns a one-byte response in the range 00h to 04h just as it would in Legacy mode. However, it is undefined whether or not A9h causes the KBC to revert to Legacy mode; if not, it is undefined which port (if any) is actually tested.

Rationale: Since A9h is rarely used, and there is no easy way to define it rigorously in the case of multiple AUX ports, it is easiest to define A9h to revert to Legacy mode.

If the ability to test individual AUX ports is desired, a reasonable KBC extension would be to allow command A9h to be preceded by a routing prefix.

AAh The KBC Self Test command always causes the KBC to revert to Legacy mode before executing the self-test.

Rationale: The AAh command is typically issued by the system BIOS at boot time. On a typical system, having AAh revert to Legacy mode will satisfy the requirement (section 3.3.3) for the KBC to revert to Legacy mode when the system boots.

D3h The Force AUX Output command operates in largely the same way in Multiplexed mode as in Legacy mode. The argument byte is echoed back as if received from AUX port #0, i.e., with Source = '00', AUX = '1', OBF = '1', and Error = '0'.

(Note also the special usage of command D3h in the Activation and Deactivation sequences; see section 3.2.)

Rationale: The D3h command is generally used only by system software that wishes to simulate mouse input. For example, legacy USB mouse support might translate USB packets into PS/2 packets which are then presented to the PS/2 mouse driver using command D3h.

It is possible that non-system software could use D3h to meddle with the mouse, in which case it would seem to be good for D3h to revert to Legacy mode so that the multiplexing driver can reinitialize itself. However, D3h is so useful to system software, and so rarely (if ever) used by non-system software, that it is better overall to define D3h to leave the KBC in Multiplexed mode and to echo the data on a known port.

A typical implementation might be for the system to use D3h to support legacy USB on port #0. In the usual case where there is either no USB mouse, or a USB mouse with native operating system support, the PS/2 mouse driver will perceive port #0 as unattached. In this situation, meddlesome use of command D3h by non-system software can be detected because data

(other than AAh 00h) will appear to arrive unexpectedly on port #0.

Another advantage to having D3h respond with Source = '00' is that it is convenient to implement: The D3h command in Legacy mode would also set those bits to '00'. (Note, however, that the Error bit is set differently depending on whether the device is in Legacy or Multiplexed mode; see also section 3.9.3.)

This specification provides no standard way to simulate input from a source other than port #0; if this feature is desired, a reasonable KBC extension would be to allow command D3h to be preceded by a routing prefix.

D4h This command always causes the KBC to revert to Legacy mode; once in Legacy mode, the D4h command is processed as usual.

Rationale: Legacy drivers will typically issue an FFh or F6h command as their first step in initializing the mouse. This rule ensures that the KBC will revert to Legacy mode if the multiplexing mouse driver is replaced by a legacy driver while the system is running. The KBC must revert to Legacy mode so that the legacy driver will interpret input port 64h properly.

It is best for all D4h commands to revert to Legacy mode in order to detect interference by legacy software. For example, if software uses D4h to send F5h (Disable) and F4h (Enable) commands to the mouse without the knowledge of the multiplexing mouse driver, then the KBC will appear to revert spontaneously to Legacy mode from the point of view of the multiplexing driver. As described in section 3.9, the driver will quickly discover that this has happened by interpreting the bits of input port 64h. The driver can then reinitialize itself as needed.

Some other KBC commands may cause the KBC to revert to Legacy mode when executed in Multiplexed mode. For example, FEh (System Reset) may or may not cause reversion to Legacy mode. However, the commands 20h, 60h, ABh, ADh, AEh, C0h, D0h, D1h, D2h, and D3h must *not* cause the KBC to revert to Legacy mode.

Standard KBC commands not otherwise covered by this document should operate the same in Multiplexed and Legacy modes, whether or not they cause a reversion to Legacy mode. However, the KBC may implement additional non-standard commands whose behavior is affected by Multiplexed mode.

The following table summarizes the various KBC commands and their effects on Multiplexed mode.

KBC command	*In Legacy mode…*	*In Multiplexed mode…*
20h	Read Command Byte	*Same; do not revert*
60h	Write Command Byte	*Same; do not revert*
90h	(See section 3.4.1)	Route to Port #0
91h	(See section 3.4.1)	Route to Port #1
92h	(See section 3.4.1)	Route to Port #2
93h	(See section 3.4.1)	Route to Port #3
A7h	Disable AUX Port	(See section 3.7)
A8h	Enable AUX Port	(See section 3.7)
A9h	Test AUX Port	Poss. revert and Test
AAh	Self Test	Revert and Self Test
ABh	Test Keyboard Port	*Same; do not revert*
ADh	Disable Keyboard	*Same; do not revert*
AEh	Enable Keyboard	*Same; do not revert*
C0h	Read Input Port	*Same; do not revert*
D0h	Read Output Port	*Same; do not revert*
D1h	Write Output Port	*Same; do not revert*
D2h	Force Keyboard Output	*Same; do not revert*
D3h	Force AUX Output	Force Port #0 Output
D4h	Output to AUX	Revert and Output
Other	(Normal function)	*Same; may revert or not*
Activation sequence	Switch to Multiplexed	Remain in Multiplexed
Deactivation sequence	Remain in Legacy	Revert to Legacy

Figure 12. KBC commands in Legacy and Multiplexed modes.

3.9. Reversion to Legacy mode

Software in the PC world is notoriously unregulated. Numerous situations arise when software other than the mouse driver might try to access the AUX port without the mouse driver's knowledge or consent. In the Active Multiplexing system, such interference generally either will leave Multiplexed mode unaffected (e.g., enabling and disabling the keyboard or AUX port), or it will cause the KBC to revert to Legacy mode. When the latter occurs, the KBC will appear to revert "spontaneously" to Legacy mode from the point of view of the multiplexing driver.

There are several ways for the multiplexing driver to discover that the KBC has reverted to Legacy mode. The driver should respond by reinitializing Multiplexed mode, and preferably by completely reinitializing all the pointing devices. (As described in section 2.3, full reinitialization is a good idea since interference by non-mouse drivers may disrupt the state of enhanced pointing devices like the IntelliMouse.)

Rationale: An example scenario where the KBC may revert to Legacy mode is a suspend/resume in Windows NT 4.0, which does not support formal power management events. Another is the use of games or system utilities that try to take over the mouse for a period of time before handing it back to the mouse driver.

Instead of making a futile attempt to head off all such interference, it is best to provide a simple way for the multiplexing driver to detect the interference and reinitialize itself.

There is no way for the KBC to notify the multiplexing driver at the time of reversion to Legacy mode, since the multiplexing driver may well be disconnected from IRQ 12 at that time. Instead, mechanisms exist to allow the driver to detect the reversion the next time it interacts with the KBC.

3.9.1. Detecting reversion when receiving data

Bit 2 of input port 64h is the System Flag bit most of the time, but it becomes an Error bit when AUX input is pending (i.e., when AUX = '1' and OBF = '1'). In Legacy mode, this bit is forced to '1'. In Multiplexed mode, this bit reports input errors and is '0' for normal inputs. If the driver reads AUX = '1', OBF = '1', and Error = '1', it should read an error code from input port 60h. That code should be either FFh (Parity Error), FEh (Timeout Error), or FDh (Hot Removal). If the error code is anything else, then the driver can conclude that the KBC has reverted to Legacy mode and is passing on a normal data byte from the AUX device.

Rationale: Typically the first data to arrive after reversion to Legacy mode will be the first byte of a mouse motion packet. The first byte of a mouse packet is the status byte, which is very unlikely to be in the range FDh–FFh: These values decode as a three-button mouse with the middle button and at least one other button held down, plus an overflowing amount of motion in the negative direction in both X and Y. Even if this occurs, the reversion is guaranteed to be detected on the second byte of the packet, which must be 00h because of the overflow condition indicated in the first byte. The same argument applies to IntelliMouse enhanced packets; Synaptics TouchPad enhanced packets always have a first byte in the range 80h–BFh.

If the multiplexing driver sees a data byte in the range FDh–FFh with the Error bit set to '1', it can poll the KBC as described in section 3.9.3 to check for reversion to Legacy mode as one step of its error processing.

Figure 13 summarizes the method for detecting reversion to Legacy mode when AUX = '1' and OBF = '1'.

Port 64h, bit 2	*Port 60h*	*Interpretation*
0	*any*	KBC is in Multiplexed mode.
1	00h–FCh	KBC has reverted to Legacy mode.
1	FDh–FFh	Error or reversion; poll to distinguish.

Figure 13. Detecting reversion when receiving data.

3.9.2. Detecting reversion in command responses

If the driver attempts to send a command to a device after reversion to Legacy mode, it will receive an FFh Parity error response as described in section 3.4.1. When the driver receives an FFh error response, it can poll the KBC as described in section 3.9.3 to distinguish between legitimate parity errors and reversion to Legacy mode.

3.9.3. Detecting reversion by polling

Host software can use the D3h command to check whether the KBC is in Legacy or Multiplexed mode. The host issues a D3h command with any argument byte (for example, 00h). The response will be the usual IRQ 12 with input port 60h holding that argument byte. The host can examine bit 2 of input port 64h to detect the KBC mode. Bit 2 will report an Error bit of '0' if the KBC is in Multiplexed mode, or it will be forced to '1' if the KBC is in Legacy mode. (See sections 3.5.1 and 3.8.)

3.10. Unattached and unimplemented ports

If a routing command is used to send a data byte to an AUX port which does not have a device attached, the response should be a Timeout error. Specifically, the KBC should set output port 64h to OBF = '1', AUX = '1', Error = '1', and the Source bits to indicate the same port as the routing prefix. Output port 60h should hold the value FEh, with IRQ 12 raised.

If a particular KBC does not implement all four AUX ports, then attempts to route data bytes to an unimplemented port should act as if the bytes were sent to an implemented but unattached port, as described in the preceding paragraph.

Some KBCs allow keyboards to be attached to the same external connector that accepts a mouse. On these KBCs, if a particular port has a keyboard attached instead of a mouse, then the port should act as if it is unattached for purposes of the AUX-oriented operations described in this document.

If an A7h or A8h command is routed to a currently-unattached port, the corresponding Port Disable flag must be set or cleared accordingly even though its effect will not be noticeable until a device is later attached to the port. Note that a multiplexing driver will generally leave unattached ports enabled in order to be able to detect hot-plugging on those ports.

A7h and A8h commands routed to *unimplemented* ports may safely be ignored since the Port Disable flag for an unimplemented port can never affect the KBC's operation in any way, and thus may itself be left unimplemented.

3.11. Hot-plugging

In Multiplexed mode, the KBC must *not* attempt to handle hot-plugging in the ways described in section 2.3. Specifically, the KBC must pass the AAh, 00h sequence on to the host instead of absorbing that sequence, and the KBC must not attempt to initialize a device that is hot-plugged on any AUX port.

(The KBC may track hot-plugging and hot-removal of mice and other AUX devices for its own record-keeping purposes; however, it must refrain from using this knowledge to interfere with the host driver's direct access to the devices.)

Because the PS/2 connector is not designed for hot-plugging or hot-removal, these actions may generate spurious signals which may, for example, appear as spurious bytes preceding the AAh, 00h sequence. The KBC may take steps to filter out these spurious signals, but it must not do so in a way that could interfere with legitimate transmissions from AUX devices, including non-mouse devices.

The KBC may support hot-plugging in Legacy mode if desired.

Some KBCs check for the presence of a mouse at boot time and permanently disable the AUX port if no mouse is present at boot time. KBCs may continue to have this behavior in Legacy mode, but in Multiplexed mode, it must be possible to hot-plug an AUX device to a port even if no device was attached to the port at boot time.

Some KBCs check periodically for hot-removal of a pointing device. The KBC may choose to support this or not in Legacy mode and/or Multiplexed mode. In Multiplexed mode, if the KBC detects hot-removal of a device on a particular port, it can optionally report this event to the driver as an FDh byte from that port with the Error flag set to '1'. The KBC is not required to report hot-removal in this way, nor is the driver required to act on hot-removal information. Note also that some obvious methods for checking for hot-removal, such as sending periodic probing commands, may interfere with the proper operation of some AUX devices, especially non-mouse devices.

3.12. Suspend and resume

In case of a suspend and resume during Multiplexed mode, the system may handle the AUX ports in any of the following ways:

1. The system may leave the AUX device powered during the suspend, so that no special action is necessary on the part of the driver.
2. The system may power down and re-power the AUX device, but it must pass the resulting AAh, 00h announcement on to the host driver. The driver can respond to this announcement just like a hot-plug event.
3. The KBC may revert to Legacy mode; once the KBC is in Legacy mode, the system (i.e., the driver) may reinitialize the AUX devices in any way it sees fit, as discussed above in section 3.3.1.

Rationale: In normal circumstances, the operating system will notify the mouse driver of the suspend/resume, and the mouse driver will reinitialize the pointing devices. However, some operating systems are still in use (Windows 3.1, Windows NT 4.0) which do not support power management. For the benefit of these operating systems, the KBC should give the driver enough information to know to reinitialize the AUX devices when necessary.

4. Summary of Implementation Dependencies

This specification leaves a number of details up to individual implementers.

This document is freely redistributable, but Synaptics requests that it be distributed only in its original form without modifications. KBC implementers may wish to distribute this document with an appendix that describes their choices on each of the following issues. All items apply to the KBC's behavior in Multiplexed mode unless otherwise noted.

3.2A	The set of KBC commands that will reset the parser for Activation and Deactivation sequences.
3.2B	Whether *xy*h response to Activation sequence is sent in Multiplexed mode.
3.2C	Whether *xy*h response to Deactivation sequence is sent in Legacy mode.
3.3.1A	The general behavior of the KBC in Legacy mode.
3.3.1B	Whether the KBC supports hidden multiplexing in Legacy mode.
3.3.1C	Whether the KBC supports hot-plugging in Legacy mode.
3.4A	The effect of commands 90h–93h not followed by either a data byte or command A7h or A8h.
3.4.1A	The effect of prefixed commands A7h and A8h in Legacy mode.
3.5.1A	The behavior of the Source and AUX bits when OBF is '0'.
3.5.1B	The behavior of the System Flag / Error bit when OBF is '0'.
3.5.1C	Whether "true" and "reverted" Legacy modes are treated differently.
3.5.3A	The time limit before which a Timeout error is signaled.
3.6A	The set of port numbers (#0–#3) which are implemented.
3.6B	The types of ports or devices attached to the various port numbers (#0–#3).
3.6C	Whether dual external ports share a port number or use distinct port numbers.
3.6D	Whether the external port and legacy USB device share a port number or use distinct port numbers.
3.8A	Whether or not command A9h reverts to Legacy mode; its behavior if not.

3.8B Whether or not the A9h command accepts a routing prefix.

3.8C Whether or not the D3h command accepts a routing prefix.

3.8D The mechanism by which a legacy USB device is simulated.

3.8E The set of other KBC commands which revert to Legacy mode.

3.10A Whether the KBC recognizes a keyboard attached to the external AUX port.

3.11A Whether the KBC can report hot-removal using code FDh and Error = '1'.

3.12A The method of handling suspend/resume (each port may use method 1 or method 2, or the KBC as a whole may use method 3).

Acknowledgments

This standard could not have been created without the valuable contributions of many pointing device, KBC, driver, and computer manufacturers. Synaptics would particularly like to thank Joe Rutledge and Steve Ihde of IBM; Aleksandr Frid of Phoenix; Anil Mhatre of Insyde; Robert Dezmelyk of LCS/Telegraphics; and David Delisle and Paul Cheok for their contributions.

Revision history

Revision 1.0, created 10/12/1998. Internal review copy, never released as final spec.

Revision 1.1, released 5/17/1999. First official release of spec.

