x1b04 - special keys and mouse buttons

Version 4

SpecialKeysMseButtons

This feature describes nonstandard buttons present on a device and allows them to be diverted to software or mapped to different native functions internally within the device.

[0] **getCount**() → count
[1] **getCidInfo**(index) → cid, tid, flags, pos, group, gmask, additionalflags
[2] **getCidReporting**(cid) → cid, divert, persist, forceRawXY, rawXY, remap, analyticsKeyEvt
[3] **setCidReporting**(cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, avalid, analyticsKeyEvt) → cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, analyticsKeyEvt
[event0] **divertedButtonsEvent** → cid1, cid2, cid3, cid4
[event1] **divertedRawMouseXYEvent** → dx, dy
[event2] **analyticsKeyEvents** → cid1, event_1, cid2, event_2, cid3, event_3, cid4, event_4, cid5, event_5

Overview

Functions are provided to enumerate controls on the device which are interesting to software. This includes any control which can be reprogrammed or should be handled by the software in some special way.

Once enumerated, controls can be configured to perform actions different from their default.

Any control which has a boolean state can be supported by this feature. This usually means buttons, but it is possible that other types of controls may be included. For clarity this document may refer to the state of a control in button terms such as being pressed or released, but these states could also represent the active or inactive state of a non-button control.

Controls are identified by control ID. The control ID value describes what icon if any appears on the control and how the device natively handles the control. See the control ID table for information on specific control IDs.

The control ID is used by software to determine the possible list of options the user may assign to the control.

A particular control ID should not appear more than once in the enumerated list of control IDs. If duplicate control IDs are desired a new control ID should be created to identify the secondary instance of the control.

In its default configuration a control performs some native action on the device when pressed or released. This native action may be reporting the usage in a HID report or some more complex control specific action. The concept of diverting a control is to suppress this native action and instead report any user action on a control to software via the HID++ [event0] **divertedButtonsEvent** → cid1, cid2, cid3, cid4.

When a device lists a control as "divertable" it indicates that it has the capability to divert the control but is not necessarily telling the sofware that it should divert it. The software will decide based on the control ID whether it can provide a better action for the control and may divert it to do so. In the same way, if a control is divertable it may also have the capability of diverting raw mouse xy reports ("rawXY" and "forceRawXY" flags).

The "reprog" flag in the device tells the software that it should allow user configuration of the control. This flag is required for the control to appear as configurable in the software UI.

The "analyticsKeyEvents" flag shows the device capability to send project-dependent notifications related to analytics key events (e.g., just pressed key, just released key, etc). Those notification will help the SW to collect data and perform analytics tasks. Therefore, this flag is completely independent from the "divertable" state of the control IDs.

The task ID value reported by the device tells the software how it should handle the control. If the control is reprogrammable (has the "reprog" flag), this task ID will be used as the default task assigned to the control. If the control is not reprogrammable, the task ID might indicate the software should divert the control and handle it in some special way, or that it should do nothing special with the control and let the firmware natively handle it. See the task ID table for information on specific task IDs.

Functions and Events

[0] getCount() → count

Returns the number of items in the control ID table. This includes physical controls and extra native functions that buttons can be mapped to.

Parameters

none

Returns

count

The number of items in the control ID table.

Table 1. getCount() response packet

byte \ bit	7	6	5	4	3	2	1	0
0	count							

[1] getCidInfo(index) → cid, tid, flags, pos, group, gmask, additionalflags

Returns a row from the control ID table. This table information describes capabilities and desired software handling for all the controls defined in the device.

Parameters

index

The zero based control ID index to retrieve.

Table 2. getCidInfo request packet

byte \ bit	7	6	5	4	3	2	1	0
0	index							

Returns

cid

Control ID for a physical or virtual control. If this row is an extra native function then this is the control ID that would normally produce this native function.

tid

Task ID which specifies how the software should handle this control.

mouse flag

This control is a mouse button.

fkey flag

This control resides on a function key. A function key is a key labeled with a number such as F1 to F16 and is on the function key row.

hotkey flag

This control is a nonstandard key which does not reside on a function key. It may be anywhere on the device including the function key row.

fntog flag

This key is affected by the fnToggle setting. This flag does not indicate whether fn is required to use the key. This flag cannot appear with the mouse flag or hotkey flag.

reprog flag

The software should let the user reprogram this control. If not set, the software should improve handling of the control ID as described by the task ID without allowing the user to reprogram it.

divert flag

This control can be temporarily diverted to software by reporting via [event0] **divertedButtonsEvent** → cid1, cid2, cid3, cid4.

persist flag

This control can be persistently diverted to software by reporting via [event0] **divertedButtonsEvent** → cid1, cid2, cid3, cid4.

virtual flag

This item is a virtual control representing an optional native function (broadcasting raw (dx,dy) data can be considered as such) on the device that can be mapped to a physical control in the same device or another one (e.g. duo-link).

pos

The position of the control on the device.

```
   0 = not specified
1-16 = if fkey, control is located on this F-key number from F1 to F16
```

group

Which mapping group this control ID belongs to.

```
  0 = does not belong to a group
1-8 = belongs to this group number (g1 through g8)
```

gmask (g1-g8)

This control can be remapped to any control ID contained in the specified groups.

raw XY flag

This control has the capability of being programmed as a gesture button, which implies the control can be diverted along with raw mouse xy reports to the SW if this control is currently diverted and pressed. On the other side, the SW will perform gesture detection and gesture task injection.

forceRawXY flag

This control has the capability of being programmed as a gesture button yet the activation of the raw XY function will be initiated by SW without the need of any user action. However, this control will still need to be currently diverted.

analyticsKeyEvents flag

This control has the capability of sending project-dependent analytics key events to the SW. On the other side, the SW will gather these notifications and use them for analytics purposes. This functionality is entirely independent from the divertable state of the control.

Table 3. getCidInfo response packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid msb							
1	cid lsb							
2	tid msb							

byte \ bit	7	6	5	4	3	2	1	0
3	tid lsb							
4	virtual	persist	divert	reprog	fntog	hotkey	fkey	mouse
5	pos							
6	group							
7	g8	g7	g6	g5	g4	g3	g2	g1
8	unused	unused	unused	unused	unused	analyticsKeyEvents	forceRawXY	rawXY

[2] getCidReporting(cid) → cid, divert, persist, forceRawXY, rawXY, remap, analyticsKeyEvt

This returns the current reporting method for a control ID. The current reporting method will have been previously configured via the function [3] **setCidReporting**(cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, avalid, analyticsKeyEvt) → cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, analyticsKeyEvt. See that function for a more detailed description of the parameters.

Parameters

cid

The control ID whose reporting method is being requested.

Table 4. getCidReporting request packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid msb							
1	cid lsb							

Returns

cid

The control ID whose reporting method is being requested.

divert

Flag indicating that the control is being temporarily diverted.

persist

Flag indicating that the control is being persistently diverted.

rawXY

Flag indicating that the control is being temporarily diverted along with mouse xy reports.

forceRawXY

Flag indicating that the control is being force diverted by SW. i.e. no need of user action to send raw XY.

remap

The control ID that this control has been remapped to.

```
  0 = Control is mapped to its own control ID
1-N = Control is remapped to this control ID
```

analyticsKeyEvt

Flag indicating that the control is temporarily reporting project-dependent analytics key events to the SW.

```
0 = disabled analytics key events
1 = enabled analytics key events
```

Table 5. getCidReporting response packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid msb							
1	cid lsb							
2	---	forceRawXY	---	rawXY	---	persist	---	divert
3	remap msb							
4	remap lsb							
5	---	---	---	---	---	---	---	analyticsKeyEvt

[3] setCidReporting(cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, avalid, analyticsKeyEvt) → cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, analyticsKeyEvt

This configures the current reporting method for a control ID. If successful, the request packet is echoed as the response.

Parameters

cid

The control ID whose reporting method is being configured.

divert

Flag which causes this control to temporarily be diverted to software via [event0] **divertedButtonsEvent** → cid1, cid2, cid3, cid4. This flag is ignored by the device if d-valid is not set. The "temporary" means that it will be reset to its default value when a HID++ configuration reset occurs. See feature 0x0020 for more info on reset policy.

```
0 = not diverted
1 = diverted
```

dvalid

Flag which indicates that the divert flag is valid and device should update the temporary divert state of this control ID.

```
0 = divert flag is ignored by device
1 = divert flag affects the divert setting in device
```

persist

Flag which causes this control to be persistently diverted to software via [event0] **divertedButtonsEvent** → cid1, cid2, cid3, cid4. This flag is ignored by the device if p-valid is not set.

```
0 = not diverted
1 = diverted
```

NOTE If either of the divert or persist flags is set, the control will be diverted via HID++ notification.

pvalid

Flag which indicates that the persist flag is valid and device should update the persistant divert state of this control ID.

```
0 = divert flag is ignored by device
1 = divert flag affects the divert setting in device
```

forceRawXY

Flag which indicates that the control is being force diverted by SW

```
0 = not diverted
1 = diverted and ignore user action (i.e. device sends raw XY without the need of user
interaction)
```

fvalid

Flag which indicates that the forceRawXY flag is valid and device should update the persistant forceRawXY state of this control ID.

```
0 = forceRawXY flag is ignored by device
1 = forceRawXY flag affects the divert setting in device
```

rawXY

Flag which causes this control and all raw mouse move reports to temporarily be diverted to software via [event1] **divertedRawMouseXYEvent** → dx, dy. This flag is ignored by the device if r-valid is not set. The "temporary" means that it will be reset to its default value when a HID++ configuration reset occurs. See feature 0x0020 for more info on reset policy.

```
0 = not diverted
1 = diverted
```

rvalid

Flag which indicates that the divert raw xy flag is valid and device should update the temporary divert state of this control ID.

```
0 = divert flag is ignored by device
1 = divert flag affects the divert setting in device
```

remap

The control ID to remap this control to. This can be any entry from the control ID table (physical controls and extra native functions included) that is included in the original control ID's group mask. Setting this will cause this control to perform the native function of the target control ID rather than the normal native function of this control ID. To disable the remapping, remap the control to its own control ID.

```
  0 = Keep the previously configured remap setting (do not change)
1-N = Remap control to this control ID
```

The remapping is temporary and is reset by the device to its own control ID when a HID++ configuration reset occurs.

While a button remains pressed changes to the remap setting on the button do not affect the functionality of the button. The new setting takes effect after the button is released.

If a control is both temporarily diverted and remapped, the control is diverted using the original control ID and the remapping has no effect.

If a control is both persistently diverted and remapped, the native function of the target control ID will be performed and the diversion has no effect. Persistent settings are lower priority than temporary settings.

If the target control ID of a remapping is diverted, the native function of the target control ID will be performed and the diversion has no effect. Remapping is not recursive.

analyticsKeyEvt

Flag which causes this control to temporarily report project-dependent analytics key events to software via [event2] **analyticsKeyEvents** → cid1, event_1, cid2, event_2, cid3, event_3, cid4, event_4, cid5, event_5. This flag is ignored by the device if k-valid is not set. The "temporary" means that the flag will be reset to its default value when a HID++ configuration reset occurs. See feature 0x0020 for more info on reset policy.

```
0 = disable analytics key events
1 = enable analytics key events
```

avalid

Flag which indicates that the analyticsKeyEvt flag of the CId is valid and device should temporarily update it

```
0 = analyticsKeyEvt flag is ignored by device
1 = keyStanalyticsKeyEvtte flag affects the notification setting in device
```

Table 6. setCidReporting request packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid msb							
1	cid lsb							
2	fvalid	forceRaw XY	rvalid	rawXY	pvalid	persist	dvalid	divert
3	remap msb							
4	remap lsb							
5	---	---	---	---	---	---	avalid	analytics KeyEvt

NOTE Calling this command doesn’t immediately modify the reporting method. Instead, the new settings are kept into a buffer and it is to the application to update them when it is safe to do it (via a dedicated API). Tipically, this is performed when no control ID is currently being used.

Returns

The setCidReporting response packet echoes the data from the request packet.

Table 7. setCidReporting response packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid msb							
1	cid lsb							
2	fvalid	forceRaw XY	rvalid	rawXY	pvalid	persist	dvalid	divert
3	remap msb							

byte \ bit	7	6	5	4	3	2	1	0
4	remap lsb							
5	---	---	---	---	---	---	avalid	analytics KeyEvt

[event0] divertedButtonsEvent → cid1, cid2, cid3, cid4

This event reports a list of diverted buttons which are currently pressed. It is sent in response to changes in the pressed state of diverted buttons.

This event reports from zero to four simultaneously pressed keys. Keys are packed from the beginning of the report so that there are no gaps. Note that the position of a pressed key in the report might move if a key preceding it in the report is released. Newly pressed keys are added to the end of the report so that keys appear in the order they were pressed.

Configuration changes such as diverting or undiverting a pressed button does not trigger this report. While a button remains pressed, changes to the divert status are stored in the device memory but do not affect the button’s current divert state. The button’s current divert state is updated only on the event of physical button press.

Table 8. divertedButtonsEvent packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid1 msb							
1	cid1 lsb							
2	cid2 msb							
3	cid2 lsb							
4	cid3 msb							
5	cid3 lsb							
6	cid4 msb							
7	cid4 lsb							

[event1] divertedRawMouseXYEvent → dx, dy

This event reports the mouse XY data when the diverted button is pressed and mouse move happens. The reports are sent in response to pressed state of diverted button and mouse move.

This event does not report button pressed/released information, the button pressed and released events are still sent via event0

Configuration changes such as diverting or undiverting a pressed button does not trigger this report. While a button remains pressed, changes to the divert status are stored in the device memory but do not affect the button’s current divert state. The button’s current divert state is updated only on the event of physical button press.

Table 9. divertedRawMouseXYEvent packet

byte \ bit	7	6	5	4	3	2	1	0
0	dx msb							
1	dx lsb							
2	dy msb							
3	dy lsb							

[event2] analyticsKeyEvents → cid1, event_1, cid2, event_2, cid3, event_3, cid4, event_4, cid5, event_5

This event reports up to five analytics key events associated to the CIds defined in this feature. This event is independent from the divertable state of the controls The analytics key events reported by this notification (labeled as "event_x") are project-dependent and notified to the SW for analitics purposes.

Table 10. divertedButtonsEvent packet

byte \ bit	7	6	5	4	3	2	1	0
0	cid1 msb							
1	cid1 lsb							
2	event_1							
3	cid2 msb							
4	cid2 lsb							
5	event_2							
6	cid3 msb							
7	cid3 lsb							
8	event_3							
9	cid4 msb							
10	cid4 lsb							
11	event_4							
12	cid5 msb							
13	cid5 lsb							
14	event_5							

NOTE This notification reports the events in the same order they have been reported, and it is possible to have more than one event associated to the same cid.

Examples

Here is an example control ID table which demonstrates the group and gmask characteristics.

Table 11. Example Control ID table

index	cid	tid	flags	pos	group	gmask
0	80 = Left	56 = Left Click	0x01	0	1	00000000
1	81 = Right	57 = Right Click	0x01	0	1	00000000
2	82 = Middle	58 = Middle Button	0x31	0	1	00000011
3	83 = Back	60 = BackEx	0x31	0	1	00000001
4	86 = Forward	62 = ForwardEx	0x31	0	1	00000001
5	195 = AppSwitchGesture	156 = GestureButton	0x31	0	2	00000011
6	196 = SmartShift	157 = SmartShift	0x31	0	2	00000011
7	315 = LedToggle	221 = LedToggle	0x80	0	2	00000000

This example is of a mouse with the normal first 5 buttons followed by 2 special buttons and 1 virtual native function that physical buttons can be mapped to.

The Left and Right buttons cannot be remapped.

The buttons Back or Forward can be mapped to any group 1 action including Left, Right, Middle, Back or Forward.

The buttons Middle, ThumbAppSelect, or SmartShift can be mapped to any group 1 or group 2 action including Left, Right, Middle, Back, Forward, AppSwitchGesture (or any other gesture button action supported by software), SmartShift, or LedToggle.

Virtual native functions will always have a pos and gmask of 0 since those fields describe characteristics which only apply to a real physical control. The flags value for virtual function items will always be 0x80.

Proposed extensions

Ideas for extending this feature that are not yet implemented in any device or software. When needed these can be added in a future version of this feature.

Protection for left click button

Since most operating systems do not provide security for HID++ it is possible for third party software to accidentally or maliciously configure our devices. Therefore it is desirable for devices to protect themselves from being configured into an unusable state.

For mice, a functioning left click button is essential for the device to be usable. If this feature allows remapping of the left button, it should be only to swap the left and right buttons. This feature should not allow diverting the left or right buttons.

To accomplish this, the setCidReporting firmmware handler should act as follows:

- For setCidReporting(LEFT_CLICK):
  - if divert != 0, fail with error
  - if remap != 0 and remap != LEFT_CLICK and remap != RIGHT_CLICK, fail with error
  - if remap == RIGHT_CLICK, silently force remap of RIGHT_CLICK to LEFT_CLICK
- For setCidReporting(RIGHT_CLICK):
  - if divert != 0, fail with error
  - if remap != 0 and remap != LEFT_CLICK, silently force remap of LEFT_CLICK to LEFT_CLICK

Mouse button position

For control IDs with the mouse flag set, the position field could describe where on the device the button is located. Something like "top left", "top right" "underside left" and so on and so forth. Software could use this to show the position of the control on the device.

Persistent remapping

To persistently remap a button to a different native function, getCidInfo() could return an additional group mask byte which lists up to 8 groups that the button can be persistently mapped to. To remap a button persistently the software would send a nonzero "persistent" control ID in bytes 3 and 4 of setCidReporting which would also be returned in bytes 3 and 4 of getCidReporting.

ChangeLog

- Version 4: Add "analytics key events" capability
- Version 3: Add force raw XY capability (for 1+1=3)
- Version 2: Add rawXY reporting capability
- Version 1: Add virtual flag, group and gmask to [1] **getCidInfo**(index) → cid, tid, flags, pos, group, gmask, additionalflags. Adds remap to [2] **getCidReporting**(cid) → cid, divert, persist, forceRawXY, rawXY, remap, analyticsKeyEvt and [3] **setCidReporting**(cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, avalid, analyticsKeyEvt) → cid, divert, dvalid, persist, pvalid, forceRawXY, fvalid, rawXY, rvalid, remap, analyticsKeyEvt.
- Version 0: Initial version