1. **DISCLAIMER**

**THIS SPECIFICATION IS LICENSED AND PROVIDED BY LOGITECH "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL LOGITECH OR ANY OF ITS AFFILIATED COMPANIES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.**

1. **INTRODUCTION**

## **Hidpp10 vs Hidpp20**

Both protocols use the same transport. From a host SW perspective the API used to send/receive already crafted hidpp10 and hidpp20 messages should be the same.

Hidpp20 enumeration process is described elsewhere.

This document exposes a subset of the hidpp10 specification that targets the Logitech Unifying receiver. Among other things, device connect/disconnect, pairing/unpairing is covered by this document.

The Logitech Unifying receiver carries the USB VID 046d and PID 0xc52b

## **Definitions**

A "HID++ device" can refer to any type: mouse, keyboard, but also embedded receiver.

## **HID++ Definition**

The HID++ protocol is a Logitech specific protocol with the following main characteristics:

* laying on top of the HID class, requires only a data pipe, no control pipe
* can use USB or BT (Bluetooth) as underlying transport
* uses the interrupt channel
* declared as vendor specific HID reports in the HID descriptor
* uses the Report ID to identify the HID++ commands of fixed length
* uses very few commands to access an address/register structure in the device
* a value at an address in this structure can represent an action, a state or a stored value and is defined in the address map of the device
* use handshake mechanism and error reporting
* in one direction, only 1 command can be pending at a time

## **HID++ Transfer Concept & Rules**

* The HID++ protocol allows to transport data or commands from or to a HID device.
* Commands are always sent by the host to the device.
* HID++ information is encapsulated into a specific fixed-length format, with two possible length: short or long.
* Commands are transported in short or long packets, and can result in short or long responses, depending on the command. The length of the response is not linked to the length of the command, so a long command can result in a short response, or the opposite.
* Data can be sent by the host to the device as part of a command, or by the device to the host, either spontaneously or as a response to a command.
* Commands and data are tagged by a Device Index field, identifying the destination or the source of the HID++ packet.

## **Information from Device to PC**

### ***Spontaneous Information Delivery***

The following information is sent by the device to the PC at any time when connected:

* HID reports (keyboard, mouse, etc.)
* Battery status, F-Lock status, etc.
* Receiver messages (device arrival, departure, etc.)

The HID specifications define a way for spontaneous information delivery (HID reports). As an extension of that, the HID++ specifications define HID++ reports as a means of spontaneous information delivery of Logitech device-specific content.

## **HID++ Message Format**

The HID++ messages are sent as vendor specific HID reports with the following format:

|  |  |  |  |
| --- | --- | --- | --- |
| Report ID | Device Index | Sub ID | Parameters |

Where the fields are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Length** | **Content** | **Description** |
| Report ID | 8 bit | 0x10 (short)  0x11 (long) | The Report ID and the size of the corresponding messages are declared in the HID descriptor. |
| Device Index | 8 bit | 1 to 6 for Unifying devices.  0xFF for the receiver/corded or bluetooth | Defines the origin or destination device of the message. This may be an embedded receiver or any of its connected devices. |
| Sub ID | 8 bit | Report or message Sub ID | Sub ID of the message to follow. |

## **Report ID**

The report ID (as declared in the HID descriptor) defines the length of the HID++ message. Two message lengths are defined:

* Short messages (7 Bytes) use report ID 0x10
* Long messages (20 Bytes) use report ID 0x11

## **Device Index**

The device index tells the receiver of the message which device originated the message or which device is the destination of the message. It allows a receiver to route through messages from and to a specific device.

The receiver always uses device index 0xff.

Devices use 0x01 to 0x0fe and the device index may be statically or dynamically allocated by the receiver. The receiver sends connection messages to the host to indicate what device is connected on which device index.

## **Message Sub ID**

The following HID++ message Sub ID ranges are defined:

|  |  |  |
| --- | --- | --- |
| **Report ID** | **Sub ID** | **Description** |
| 0x10 | 0x00 – 0x7f | HID++ Reports & Notifications |
| 0x10 | 0x80 – 0xff | Register Access |

The register address map is unique to each device. Any device may contain up to 256 registers each one containing either 3bytes (short) or 16bytes (long). These registers are accessed using their address as identifier in the request. Values are sent in a LSB-first byte ordering.

### ***0x80 - SET\_REGISTER\_REQ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x80 | Address (8bit) | Value (3 Bytes) |

The SET\_REGISTER\_REQ message sets the given value to the register at the given address. Valid addresses and values are defined for each device in the register map.

Upon reception of a SET\_REGISTER\_REQ message, a SET\_REGISTER\_RSP message is returned if the value could be successfully written. Otherwise, an ERROR\_MSG is returned.

### ***0x80 - SET\_REGISTER\_RSP***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x80 | Address (8bit) | 0 (3 Bytes) |

The SET\_REGISTER\_RSP is returned on a successful SET\_REGISTER operation.

### ***0x81 - GET\_REGISTER\_REQ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x81 | Address (8bit) | Parameters (up to 3 Bytes) |

The GET\_REGISTER\_REQ message asks for a register value at the given address to be returned to the host. Valid addresses and values are defined for each device in the register map.

Upon reception of a GET\_REGISTER\_REQ message, a GET\_REGISTER\_RSP message is returned containing the register value if the value could be successfully read. Otherwise, an ERROR\_MSG is returned.

### ***0x81 - GET\_REGISTER\_RSP***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x81 | Address (8bit) | Value (3 Bytes) |

The GET\_REGISTER\_RSP message is the response message to a successful GET\_REGISTER operation. It contains the value at the requested address.

### ***0x82 - SET\_LONG\_REGISTER\_REQ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x11 | Device Index | Sub ID=0x82 | Address (8bit) | String (16 Bytes) |

The SET\_LONG\_REGISTER\_REQ message sets the given value to the register at the given address. Valid addresses and values are defined for each device in the register map.

Upon reception of a SET\_LONG\_REGISTER\_REQ message, a SET\_LONG\_REGISTER\_RSP message is returned, if the write operation was successful. Otherwise, an ERROR\_MSG is returned.

### ***0x82 - SET\_LONG\_REGISTER\_RSP***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x82 | Address (8bit) | 0 (3 Bytes) |

The SET\_LONG\_REGISTER\_RSP is returned on a successful SET\_LONG\_REGISTER operation.

### ***0x83 - GET\_LONG\_REGISTER\_REQ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x83 | Address (8bit) | Parameters (up to 3 Bytes) |

The GET\_LONG\_REGISTER\_REQ message asks for register content at the given address to be returned to the host. Valid addresses and defined for each device in the register map.

Upon reception of a GET\_LONG\_REGISTER\_REQ message, a GET\_LONG\_REGISTER\_RSP message is returned if the read operation was successful. Otherwise, an ERROR\_MSG is returned.

### ***0x83 - GET\_LONG\_REGISTER\_RSP***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Report ID=0x11 | Device Index | Sub ID=0x83 | Address (8bit) | String (16 Bytes) |

The GET\_LONG\_REGISTER\_RSP message is the response message to a successful GET\_LONG\_REGISTER operation. It contains the value at the requested address.

### ***0x8F - ERROR\_MSG***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Report ID=0x10 | Device Index | Sub ID=0x8F | Sub ID (8bit) | Address (8bit) | Error code (8bit) | 0 (1 Byte) |

An ERROR\_MSG is returned as a response to a detected error in a request or an unknown request. The message contains the SubID of the command that caused the error, the HID++ address that was faulty (or 0 if not applicable) and an error code (see §2.11).

## **Error response**

Unless otherwise specified, any command can be answered by an Error response. The error response is always short (report ID 0x10). The message contains the SubID of the command that caused the error, the HID++ address that was faulty (or 0 if not applicable) and an error code.

|  |  |  |
| --- | --- | --- |
| **Write register command (Long or Short)** | | |
| 11 ix cm rg p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 pa pb pc pd pe pf  or  10 ix cm rg p0 p1 p2 | ix | Index |
| cm | Command SubID  (for instance: 82 or 83 for long, 80 or 81 for short) |
| rg | Register |
| p0..pf | Command Parameters |
| **Error response to Command** | | |
| 10 ix 8F cm rg er 00 | ix | Index (same as command) |
| cm | Command SubID (same as command) |
| rg | Register (same as command) |
| er | Error code (see §4.4 "HID++ Error Codes") |

## **HIDPP 1.0 Error Codes**

.

**Error codes defined:**

|  |  |  |
| --- | --- | --- |
| **Code (hex)** | **Name** | **Description** |
| 0 | ERR\_SUCCESS | No error / undefined |
| 1 | ERR\_INVALID\_SUBID | Invalid SubID / command |
| 2 | ERR\_INVALID\_ADDRESS | Invalid address |
| 3 \* | ERR\_INVALID\_VALUE | Invalid value |
| 4 | ERR\_CONNECT\_FAIL | Connection request failed (Receiver) |
| 5 | ERR\_TOO\_MANY\_DEVICES | Too many devices connected (Receiver) |
| 6 | ERR\_ALREADY\_EXISTS | Already exists (Receiver) |
| 7 | ERR\_BUSY | Busy (Receiver) |
| 8 | ERR\_UNKNOWN\_DEVICE | Unknown device (Receiver) |
| 9 | ERR\_RESOURCE\_ERROR | Resource error (Receiver) |
| A | ERR\_REQUEST\_UNAVAILABLE | "Request not valid in current context" error |
| B | ERR\_INVALID\_PARAM\_VALUE | Request parameter has unsupported value |
| C | ERR\_WRONG\_PIN\_CODE | the PIN code entered on the device was wrong |
| D – FF | Reserved |  |

See section Unifying receiver errors for a more detailed explanation of error codes.

1. **UNIFYING RECEIVER NOTIFICATIONS**

## **0x40 - Device Disconnection**

This notification is sent by a receiver to the host SW to report the disconnection of a device, but only if the related reporting flag in register 0x00 “**Wireless notifications**” is enabled by the host. The Unifying devices are free to spontaneously disconnect from the host after several minutes without user interaction. Once the device is disconnected it cannot receive hidpp commands from the host.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 40 r0 00 00 00 | ix | Index |
| r0 | Disconnection type  0x00 = Reserved  0x01 = Reserved  0x02 =Device disconnected  0x03..0xFF = Reserved |
|  |  |
|  |  |
|  |  |  |

## **0x41 - Device Connection**

#### This notification is sent by a receiver to the host SW to report the connection of a device, but only if the related reporting flag in register 0x00 “**Wireless notifications**” is enabled by the host. A disconnected Unifying device will spontaneously reconnect when the user triggers activity. A connected Unifying device can send and receive hidpp commands.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 41 r0 r1 r2 r3 | ix | Index |
| r0 | bits [0..2] Protocol type  0x04 = Unifying  bits [3..7] Reserved |
| r1 | Device Info  bit0..3 = Device Type  0x00 = Unknown  0x01 = Keyboard  0x02 = Mouse  0x03 = Numpad  0x04 = Presenter  0x05 = Reserved for future  0x06 = Reserved for future  0x07 =Reserved for future  0x08 =Trackball  0x09 =Touchpad  0x0A..0x0F = Reserved  bit4 = Software Present flag  reflects flag in register 0x00, r1, bit 3  bit5 = Encryption Status  0 = Link not encrypted  1 = link encrypted  bit6 = Link Status  0 = Link established (in range)  1 = Link not established (out of range)  bit7 = Connection reason  0 = packet without payload  1 = packet with payload |
| r2 | Wireless PID LSB |
| r3 | Wireless PID MSB |

#### 

## **0x4A - Unifying Receiver Locking Change information**

Provides information about locking change (open/close lock) of the Unifying receiver.

This notification is sent by the receiver after a device has been paired successfully or an error happened during the pairing sequence. A receiver with an open lock can accept new pairings.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 4A r0 r1 r2 r3 | ix | Index 0xFF: Transceiver |
| r0 | Locking Info  bit0 = locking state  0 = locking closed  1 = locking open  bit1..7 = Reserved |
| r1 | Error Type  0x00 = no error  0x01 = timeout  0x02 = unsupported device  0x03 = too many devices  0x04 = Reserved  0x05 = Reserved  0x06 = connection sequence timeout  0x07..0xFF: Rerserved |
| r2 | Reserved |
| r3 | Reserved |

1. **UNIFYING RECEIVER REGISTERS**

## **0x00 – Enable HID++ Notifications**

This register defines a number of flags that allow the host to turn on or off individual spontaneous HID++ reports. Not setting a flag means default reporting. See the table below for more details on each flag.

For all bits: 0 = disabled (default value at power-up), 1 = enabled.

|  |  |  |
| --- | --- | --- |
| **Read short register command** | | |
| 10 ix 81 00 00 00 00 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| **Response to Read command (success)** | | |
| 10 ix 81 00 r0 r1 r2 | ix | Index (same as command) |
| r0 | HID++ Reporting Flags (Devices)  bit 0: Not applicable  bit 1: Not applicable  bit 2: Not applicable  bit 3: Not applicable  bit 4: Not applicable  bit 5: Not applicable  bit 6: Not applicable  bit 7: Not applicable |
| r1 | HID++ Reporting Flags (Receiver)  bit 0: **Wireless notifications**  bit 1: Reserved  bit 2: Reserved  bit 3: **Software Present**  bit 4…7: Reserved |
| r2 | HID++ Reporting Flags, Cont'd (Devices)  bit 0: Not applicable  bit 1: Not applicable  bit 2: Not applicable  bit 3..7: Not applicable |

|  |  |  |
| --- | --- | --- |
| **Write short register command** | | |
| 10 ix 80 00 p0 p1 p2 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| p0 | HID++ Reporting Flags (Devices)  (same format as above) |
| p1 | HID++ Reporting Flags (Receiver)  (same format as above) |
| p2 | HID++ Reporting Flags, Cont'd (Devices)  (same format as above) |
| **Response to Write command (success)** | | |
| 10 ix 80 00 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

|  |  |  |
| --- | --- | --- |
| **Flag name** | **Action if enabled** | **Action if disabled** |
| Wireless notifications | Device arrival, removal, are reported by HID++ notif. 0x40, 0x41 | Device arrival, removal, infos are not reported |

## **0x02 – Connection State**

This register allows the SW to take an action on the connection state (writing) or getting information about the connection state (reading). The fake device arrival command triggers device connection notifications for all paired devices. This can be used to enumerate the devices paired to a receiver.

|  |  |  |
| --- | --- | --- |
| **Read short register command** | | |
| 10 ix 81 02 00 00 00 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| **Response to Read command (success)** | | |
| 10 ix 81 02 00 r1 r2 | ix | Index (same as command) |
|  |  |
| r1 | Number of Connected Devices  bit 0..7: Number of connected devices (receivers only) |
| r2 | Number of Remaining Pairing Slots  bit 0..7: Number of remaining pairing slots for receivers with limited number of pairings.  255 = no remaining pairing slots  0 = no limit to remaining pairing slots |

|  |  |  |
| --- | --- | --- |
| **Write short register command** | | |
| 10 ix 80 02 p0 00 00 | ix | Index 0xFF: Transceiver |
| p0 | Action on connection state  bit 0: Reserved  bit 1: Fake Device Arrival  bit 2…7: Reserved |
| **Response to Write command (success)** | | |
| 10 ix 80 02 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

## **0xB2 – Device Connection and Disconnection (Pairing)**

Perform a Unifying device connection setup (device pairing)

|  |
| --- |
| **Read short register command: Not Applicable** |

|  |  |  |
| --- | --- | --- |
| **Write short register command** | | |
| 10 ix 80 B2 p0 p1 p2 | ix | Index 0xFF: Transceiver |
| p0 | Connect Devices  0 = No change  1 = Open Lock (RCV)  2 = Close Lock (RCV)  3 = Disconnect (unplug) (DEV)  4…255 = Reserved |
| p1 | Device number  Same value as device index transmitted in 0x41 notification |
| p2 | Open lock timeout  0 = use default value (30s)  1..255 = timeout in [s] |
| **Response to Write command (success)** | | |
| 10 ix 80 B2 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

## **0xB3 – Device Activity**

This register reports the current value of up to 16 device activity counters. The receiver increments each counter when the corresponding device sends any non-empty report. When the software needs activity information, it polls this register at regular intervals and subtracts the previous counter values from the current ones to get the number of non-empty reports received during the interval.

|  |  |  |
| --- | --- | --- |
| **Read long register command** | | |
| 10 ix 83 B3 00 00 00 | ix | Index 0xFF: Transceiver |
| **Response to Read command (success)** | | |
| 11 ix 83 B3 r0 r1 r2 r3 r4 r5 r6 r7 r8 r9 ra rb rc rd re rf | ix | Index (same as command) |
| r0 | Activity counter for device #1 |
| r1 | Activity counter for device #2 |
| r2 | Activity counter for device #3 |
| r3 | Activity counter for device #4 |
| r4 | Activity counter for device #5 |
| r5 | Activity counter for device #6 |
| r6-rf | Reserved for future extensions |

|  |
| --- |
| **Write long register command: Not Applicable** |

## **0xB5 Pairing information**

### ***0x20..0x2F - Unifying Device pairing information***

|  |  |  |
| --- | --- | --- |
| **Read long register command** | | |
| 10 ix 83 B5 nn 00 00 | ix | Index 0xFF: Transceiver |
| nn | 0x20 Device 1  0x21 Device 2  0x22 Device 3  0x23 Device 4  0x24 Device 5  0x25 Device 6  0x26..0x2F Reserved for future extensions |
| **Response to Read command (success)** | | |
| 11 ix 83 B5 nn r1 r2 r3 r4 r5 r6 r7 r8 r9 ra rb rc rd 00 00 | ix | Index (same as command) |
| nn | (same format as above) |
| r1 | Destination ID |
| r2 | Default report interval [ms]  0x00..0x07 Reserved (not supported)  0x08 8 ms  0x09..0x13 Reserved (not qualified)  0x14 20 ms  0x15..0xFF Reserved (not qualified) |
| r3 | Device Wireless PID MSB |
| r4 | Device Wireless PID LSB |
| r5 | Reserved |
| r6 | Reserved |
| r7 | Unifying device type  0x00 = Unknown  0x01 = Keyboard  0x02 = Mouse  0x03 = Numpad  0x04 = Presenter  0x05 = Reserved for future  0x06 = Reserved for future  0x07 =Reserved for future  0x08 =Trackball  0x09 =Touchpad  0x0A..0xFF = Reserved |
| r8 | Reserved |
| r9 | Reserved |
| ra | Reserved |
| rb | Reserved |
| rc | Reserved |
| rd | Reserved |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Write long register command** | | |
| 11 ix 82 B5 nn p1 p2 p3 p4 p5 p6 p7 p8 p9 pa pb pc pd 00 00 | ix | Index 0xFF: Transceiver |
| nn | (same format as above) |
| p1..pd | (same format as above) |
| **Response to Write command (success)** | | |
| 10 ix 82 B5 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

### ***0x30..0x3F - Unifying Device extended pairing info***

|  |  |  |
| --- | --- | --- |
| **Read long register command** | | |
| 10 ix 83 B5 nn 00 00 | ix | Index 0xFF: Transceiver |
| nn | 0x30 Device 1  0x31 Device 2  0x32 Device 3  0x33 Device 4  0x34 Device 5  0x35 Device 6  0x36..0x3F Reserved for future extensions |
| **Response to Read command (success)** | | |
| 11 ix 83 B5 nn r1 r2 r3 r4 r5 r6 r7 r8 r9 00 00 00 00 00 00 | ix | Index (same as command) |
| nn | (same format as above) |
| r1-r4 | Serial Number (r1 = MSB) |
| r5-r8 | Report types (r5 = MSB)  Used by Unifying Bus Enumerator |
| r9 | Usability info  bit 0..3 location of power switch  0x0 = Reserved  0x1 = on the base  0x2 = on the top case  0x3 = on the edge of the top right corner  0x4 = other  0x5 = on the top left corner  0x6 = on the bottom left corner  0x7 = on the top right corner  0x8 = on the bottom right corner  0x9 = on the top edge  0xA = on the right edge  0xB = on the left edge  0xC = on the bottom edge  0xD..0xF = Reserved  bit 4..7 Reserved |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Write long register command** | | |
| 11 ix 82 B5 nn p1 p2 p3 p4 p5 p6 p7 p8 p9 00 00 00 00 00 00 | ix | Index 0xFF: Transceiver |
| nn | (same format as above) |
| p1..p9 | (same format as above) |
| **Response to Write command (success)** | | |
| 10 ix 82 B5 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

### ***0x40..0x4F - Unifying Device name***

|  |  |  |
| --- | --- | --- |
| **Read long register command** | | |
| 10 ix 83 B5 nn 00 00 | ix | Index 0xFF: Transceiver |
| nn | 0x40 Device 1  0x41 Device 2  0x42 Device 3  0x43 Device 4  0x44 Device 5  0x45 Device 6  0x46..0x4F Reserved for future extensions |
| **Response to Read command (success)** | | |
| 11 ix 83 B5 nn r1 r2 r3 r4 r5 r6 r7 r8 r9 ra rb rc rd re rf | ix | Index (same as command) |
| nn | (same format as above) |
| r1 | Segment length (in bytes) |
| r2-rf | Name string (up to 14 bytes, UTF8 encoding) |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Write long register command** | | |
| 11 ix 82 B5 nn p1 p2 p3 p4 p5 p6 p7 p8 p9 pa pb pc pd pe pf | ix | Index 0xFF: Transceiver |
| nn | (same format as above) |
| p1..pf | (same format as above) |
| **Response to Write command (success)** | | |
| 10 ix 82 B5 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

1. **UNIFIYING RECEIVER ERRORS**

## **ERR\_CONNECT\_FAIL (0x04)**

The pairing process failed.

## **ERR\_TOO\_MANY\_DEVICES (0x05)**

Cannot pair more than six devices per Unifiying receiver

## **ERR\_ALREADY\_EXISTS (0x06)**

## **ERR\_BUSY (0x07)**

The receiver is currently handling a downstream (to device) message and cannot process a second one.

## **ERR\_UNKNOWN\_DEVICE (0x08)**

Trying to send a message to a device (device index) where there is no device paired.

## **ERR\_RESOURCE\_ERROR (0x09)**

This error is returned by the receiver when an hidpp command has been sent to a device that is in disconnected mode. When a device is in disconnected mode it cannot receive commands from the host until it reconnects. A device reconnects when the user interacts with it. In most cases, a device disconnects after several minutes of inactivity.

1. **UNIFYING DEVICES REGISTERS**

## **0x00 Enable HID++ Notifications**

This register defines a number of bits that allow the host to turn on or off individual spontaneous HID++ reports. Not setting a bit means default (i. e. HID if applicable) reporting. See the table below for more details on each flag. Not all devices support all flags, writing to an unsupported flag has no effect.

For all bits: 0 = disabled (default value at power-up), 1 = enabled.

|  |  |  |
| --- | --- | --- |
| **Read short register command** | | |
| 10 ix 81 00 00 00 00 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| **Response to Read command (success)** | | |
| 10 ix 81 00 r0 r1 r2 | ix | Index (same as command) |
| r0 | HID++ Reporting Flags (Devices)  bit 0: Consumer & Vendor Specific Control  bit 1: Power Keys  bit 2: Roller V  bit 3: Mouse Extra Buttons  bit 4: Battery Status  bit 5: Roller H  bit 6: F-Lock Status  bit 7: Numpad Numeric Keys |
| r1 | HID++ Reporting Flags (Receiver)  bit 0: Not applicable  bit 1: Not applicable  bit 2: Not applicable  bit 3: Not applicable  bit 4…7: Not applicable |
| r2 | HID++ Reporting Flags, Cont'd (Devices)  bit 0: 3D Gesture (MX Air)  bit 1: Reserved  bit 2: Reserved  bit 3..7: Reserved |

|  |  |  |
| --- | --- | --- |
| **Write short register command** | | |
| 10 ix 80 00 p0 p1 p2 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| p0 | HID++ Reporting Flags (Devices)  (same format as above) |
| p1 | HID++ Reporting Flags (Receiver)  (same format as above) |
| p2 | HID++ Reporting Flags, Cont'd (Devices)  (same format as above) |
| **Response to Write command (success)** | | |
| 10 ix 80 00 zz zz zz | ix | Index (same as command) |
| zz | (don't care, recommended to return 0) |

|  |  |  |
| --- | --- | --- |
| **Flag name** | **Action if enabled** | **Action if disabled** |
| Consumer & Vendor Specific Control | Multimedia and MS vendor specific keys are reported by HID++ notif. 0x03 | Multimedia and MS vendor specific keys are reported by normal HID reports |
| Power Keys | Power keys are reported by HID++ notif. 0x04 | Power keys are reported by normal HID reports |
| Roller V | Vertical scroll is reported by HID++ notif. 0x05 | Vertical scroll is reported by normal HID reports |
| Mouse Extra Buttons | Mouse buttons not available in the standard HID mouse report are reported by HID++ notif. 0x06 | Mouse buttons not available in the standard HID mouse report are not reported |
| Battery Status | Battery state or mileage are reported by HID++ notif. 0x07 or 0x0D (depending on the device) | Battery state or mileage are not reported |
| Roller H | Horizontal scroll is reported by HID++ notif. 0x05 | Horizontal scroll is reported by normal HID reports |
| F-Lock Status | F-Lock status is reported by HID++ notif. 0x09 | F-Lock status is not reported |
| Numpad numeric keys | Numpad keys are reported as buttons by HID++ notif. 0x03 | Numpad numeric keys are reported as normal keys |
| 3D Gesture | 3D gestures are reported by HID++ notif. 0x65 | 3D gestures are not reported |

## **0x01 – Enable Individual Features**

This register allows changing the operating mode of several device’s features. Default settings are underlined. Not all devices support all functions, setting an unsupported bit has no effect.

|  |  |  |
| --- | --- | --- |
| **Read short register command** | | |
| 10 ix 81 01 00 00 00 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| **Response to Read command (success)** | | |
| 10 ix 81 01 r0 r1 r2 | ix | Index (same as command) |
| r0 | Feature Bits  bit 0: Reserved  bit 1: Special button function **(see note)**  0 = buttons work as buttons  1 = buttons have special function  bit 2: Enhanced Key Usage  0 = normal key usage  1 = enhanced key usage  bit 3: Fast Forward/Rewind  0 = disabled, 1 = enabled  bit 4: Reserved  bit 5: Reserved  bit 6: Scrolling acceleration  0 = disabled, 1 = enabled  bit 7: buttons control the resolution  0 = work as buttons  1 = control the resolution |
| r1 | Not applicable |
| r2 | Feature Bits (Cont’d)  For all bits, unless specified: 0 = disabled, 1= enabled  bit 0: Inhibit lock key sound (1= no sound)  bit 1: Reserved  bit 2: 3D engine (1 = Enable) (MX Air)  bit 3: Host SW controls LEDs  bit 4: Reserved  bit 5…7: Reserved |

|  |  |  |
| --- | --- | --- |
| **Bit name** | **Action if '0'** | **Action if '1'** |
| Special button function | Buttons are reported as buttons, so that SW can take control over them | Buttons have a special function (multimedia, local setting, etc.) |
| Enhanced Key Usage | Fn+AlphaNumKey acts as a normal key | Fn+AlphaNumKey is reported as button + HID++ notification 0x0C |
| Fast Forward/Rewind | Disabled | A long press on the next/previous track keys acts as Fast Forward/Rewind |
| Scrolling Acceleration | Disabled | Enabled, device handles scrolling acceleration |
| buttons control the resolution | This set of buttons are reported as buttons, so that SW can take control over them | Buttons control the resolution locally (i.e. without SW intervention) |
| Inhibit lock key sound | Key sound is active | Key sound is inhibited (no sound) |
| 3D engine | 3D engine disabled | 3D engine enabled |
| SW controls LEDs | LEDs are exclusively under FW control | LEDs are controlled by SW using reg. 0x51 |

## **0xF1 – Device firmware information**

A FW name belongs to a specific product or group of products.

A different FW version implies some user visible feature differences for the same product.

|  |  |  |
| --- | --- | --- |
| **Read short register command** | | |
| 10 ix 81 F1 p0 00 00 | ix | Index 0x0n: Device #n  0xFF: Transceiver |
| p0 | Firmware information item  0x01 MCU1 - FW name (MSB) and version (LSB)  0x02 MCU1 - FW build number  0x03 MCU1 - HW version  0x04 MCU1 - Boot loader version  0x05..0x0F Reserved - MCU 1  0x10 MCU2 Reserved  0x11 MCU2 FW name (MSB) and version (LSB)  0x12 MCU2 FW build number  0x13 MCU2 HW version  0x14 MCU2 – Boot loader version  0x15..0x1F Reserved - MCU2  0x20 MCU3 Reserved  0x21 MCU3 - FW version  0x22 etc...  0x30..0xFF Reserved |
| **Response to Read command (success)** | | |
| 10 ix 81 F1 r0 r1 r2 | ix | Index (same as command) |
| r0 | Firmware information item  (same as p0 in command) |
| r1 | Requested information (MSB) |
| r2 | Requested information (LSB) |

**Examples:**

**Command Returns Explanations**

10 FF 81 F1 01 00 00 10 FF 81 F1 01 25 16 Returns FW version and release (RR25.16)

10 FF 81 F1 02 00 00 10 FF 81 F1 02 12 03 Returns FW build (build 1203)

1. **UNIFYING DEVICES NOTIFICATIONS**

## **0x03 - Consumer & Vendor Specific Control Keys**

This notification replaces “standard” HID reports, but only if the related reporting flag (Consumer & Vendor Specific control key) in register 0x00 is enabled by the host

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 03 r0 r1 r2 r3 | ix | Index |
| r0 | Keycode 1, LSB |
| r1 | Keycode 1, MSB |
| r2 | Keycode 2, LSB |
| r3 | Keycode 2, MSB |

Key code from 0x0000 to 0x0FFF represent standard HID Usages from the Consumer Control Page (0x0C).

Key code from 0x1000 to 0x1FFF represent button usages from the Button Page (0x09):  
0x1000 = reserved, 0x1001 = button 1, 0x1002 = button 2, etc.

Key code from 0x2000 to 0x2FFF represent key usages from the Keyboard/Keypad Page (0x07):

0x2000..0x2003 = reserved, 0x2004 ="A", 0x2005 = "B", etc.

Key code from 0x3000 to 0x3FFF represent button usages from the vendor-specific page used for Microsoft’s Media Center remote-control buttons (0xFFBC):

0x300D ="Media Center Start (green) Button", 0x3024 = "DVD Menu", 0x3025 = "Live TV", 0x3027 = "Zoom", 0x3028 = "DVD Eject", 0x302B = "Closed Captioning", 0x302C = "Network Selection", 0x302D = "Sub Audio", 0x302E = "DVD Sub-Title Change Track", 0x302F = "DVD Audio Track", etc.

## 

## **0x04 - Power Keys**

This notification replaces the “standard” System Power Keys report, but only if the related reporting flag (power key) in register 0x00 is enabled by the host.

The power key code uses the standard HID System Control Usages in the Generic Desktop Usage Page (0x01).

For instance, a "sleep" key report will typically have r0=0x82, and for a "power down" r0=0x81.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 04 r0 00 00 00 | ix | Index |
| r0 | key HID usage (as in HID Generic Desktop page 0x01) |
|  |  |
|  |  |

## **0x05 Roller**

This notification replaces the standard HID mouse roller report, but only if the related reporting flag (Roller V and Roller H) in register 0x00 is enabled by the host.

The report includes one byte (signed int8) for the vertical roller (also called Roller V, Wheel, vertical scroll or iNav) and one for the horizontal roller (also called Roller H, horizontal scroll).

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 05 r0 r1 00 00 | ix | Index |
| r0 | Vertical roller |
| r1 | Horizontal roller |
|  |  |

## **0x06 - Mouse Extra Buttons**

This notification is used only in case the device can report more buttons than what can be reported to the host over regular HID reports. For example: the HID mouse collection declares 8 buttons, but the device actually features 12 buttons. The HID++ report 0x06 is required to transport the 4 buttons missing in the normal mouse report.

"Mouse Extra Buttons" are reported only if the related reporting flag in register 0x00 is enabled by the host.

All 16 buttons are mapped in this notification, but the mouse buttons that can be transported in the regular HID mouse report are always reported as '0' in notification 0x06.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 06 r0 r1 00 00 | ix | Index |
| r0 | Buttons 1..8 |
| r1 | Buttons 9..16 |
|  |  |

## **0x07 - Battery/Charging Level and Status**

The battery/charging level and status is reported only if the related reporting flag in register 0x00 is enabled by the host. The "Battery/Charging Level" byte indicates the battery level if the "Charging State" indicates 0x00 ("Not Charging"). If "Charging State" indicates 0x21 to 0x23 ("Charging"), the "Battery/Charging Level" byte indicates the level of charging.

Notification 0x0D is very similar, except that the information returned is the battery/charging level is returned as the capacity (mileage) in %. A given device supports either notification 0x07, or 0x0D; never both at the same time.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 07 r0 r1 r2 00 | ix | Index |
| r0 | Battery/Charging Level  0x00 = Reserved/Unknown  0x01 = Critical  0x02 = Critical (legacy value, don't use)  0x03 = Low  0x04 = Low (legacy value, don't use)  0x05 = Good  0x06 = Good (legacy value, don't use)  0x07 = Full  0x08..0xFF = Reserved |
| r1 | Charging state  0x00 = Not charging  0x01..0x1F = Reserved (not charging)  0x20 = Unknown charging state  0x21 = Charging  0x22 = Charging complete  0x23 = Charging error  0x24..0xFF = Reserved |
| r2 | Low batt. threshold  0x00 = Not available  0x01 = Warning below 5%  0x02 = Warning below 10%  0x03 = Warning below 15%  0x04 = Warning below 20%  0x05 = Warning below 25%  0x06 = Warning below 30%  0x07..0xFF = Reserved |

## **0x09 - F-Lock Status**

An F-Lock status notification is sent whenever F-Lock status changes, but only if the related reporting flag in register 0x00 is enabled by the host.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 09 r0 00 00 00 | ix | Index |
| r0 | F-Lock status  bit0 = F-Lock (0=off, 1=on)  bit1..7 = Reserved |
|  |  |

## **0x0C – Fn + key**

When used, this notification is sent whenever any arbitrary key (but usually an alphanumeric key), which does not have any alternate function, is pressed in conjunction with the alternate-function (Fn) key.

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 0C r0 r1 r2 00 | ix | Index |
| r0 | HID usage page |
| r1 | HID usage number, LSB |
| r2 | HID usage number, MSB |

## **0x0D – Battery mileage**

Same as notification 0x07, except that the information returned is the battery/charging level is returned as the capacity (mileage) in %. A given device supports either notification 0x07, or 0x0D; never both at the same time.

The battery/charging mileage and status is reported only if the related reporting flag in register 0x00 is enabled by the host. The "Battery/Charging Level" byte indicates the battery level if the "Charging State" indicates 0x00 ("Not Charging"). If "Charging State" indicates 0x21 to 0x23 ("Charging"), the "Battery/Charging Level" byte indicates the level of charging.

INIT\_CONDITION is a 12bit constant, representing the initial battery mileage condition. The value is split between byte r1 (8 LSbits) and r2 (4 MSbits).

If on cradle (r2 byte, bits 6..7 != 0x00), this constant corresponds to the initial condition Battery life for 100%. The possible range is: 0..4096 sec/min/hr/day, the unit depending on bits 4..5 in byte r2.

If not charging (r2 byte, bits 6..7 == 0x00) , this constant corresponds to the max battery charge time (from 0% up to 100%).

|  |  |  |
| --- | --- | --- |
| **Notification** | | |
| 10 ix 0D r0 r1 r2 00 | ix | Index |
| r0 | Battery/Charging Level  bit0..6 = Battery/Charging Level , in %  0x00..0x64 = 0..100%  0x65..0x7E = Reserved  0x7F = Battery unknown  bit7 = Reserved (0) |
| r1 | INIT\_CONDITION constant value, LSB |
| r2 | Bit0..3: INIT\_CONDITION constant value, MSB  Bit 4..5 = INIT\_CONDITION units  0x00 = seconds  0x01 = minutes  0x02 = hours  0x03 = days  Bit 6..7 = Charging state  0x00 = not charging  0x01 = charging  0x02 = charging complete  0x03 = charging error |

1. **DESCRIPTORS**

## **Short Messages (Output and Input)**

/\* HID++ short messages \*/

0x06, 0x00, 0xff, // USAGE\_PAGE (Vendor Page 1)

0x09, 0x01, // USAGE (Vendor Usage 1)

0xa1, 0x01, // COLLECTION (Application)

0x85, 0x10, // REPORT\_ID (16)

0x75, 0x08, // REPORT\_SIZE (8)

0x95, 0x06, // REPORT\_COUNT (6)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x26, 0xff, 0x00, // LOGICAL\_MAXIMUM (255)

0x09, 0x01, // USAGE (Vendor Usage 1)

0x81, 0x00, // INPUT (Data,Ary,Abs)

0x09, 0x01, // USAGE (Vendor Usage 1)

0x91, 0x00, // OUTPUT (Data,Ary,Abs)

0xc0, // END\_COLLECTION

|  |  |
| --- | --- |
| Byte |  |
| 0 | Report ID = 0x10 |
| 1 | Device Index |
| 2 | Sub ID |
| 3 | Address |
| 4 | Value 0 |
| 5 | Value 1 |
| 6 | Value 2 |

## 

## **Long Messages (Output and Input)**

/\* HID++ long messages \*/

0x06, 0x00, 0xff, // USAGE\_PAGE (Vendor Page 1)

0x09, 0x02, // USAGE (Vendor Usage 2)

0xa1, 0x01, // COLLECTION (Application)

0x85, 0x11, // REPORT\_ID (17)

0x75, 0x08, // REPORT\_SIZE (8)

0x95, 0x13, // REPORT\_COUNT (19)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x26, 0xff, 0x00, // LOGICAL\_MAXIMUM (255)

0x09, 0x02, // USAGE (Vendor Usage 2)

0x81, 0x00, // INPUT (Data,Ary,Abs)

0x09, 0x02, // USAGE (Vendor Usage 2)

0x91, 0x00, // OUTPUT (Data,Ary,Abs)

0xc0, // END\_COLLECTION

|  |  |
| --- | --- |
| Byte |  |
| 0 | Report ID = 0x11 |
| 1 | Device Index |
| 2 | Sub ID |
| 3 | Address |
| 4 | Value0 |
|  | … |
| 19 | Value15 |