

## **USB.**

### ***Basic information:***

USB 2.0, Full Speed (12Mbps).

### ***Interfaces:***

#### **Firmware 0.97 and later:**

HID-compliant keyboard (Endpoint 1),

HID-compliant mouse (Endpoint 2),

HID-compliant consumer control and gamepad device (Endpoint 3).

USB HID interface provides USB host with HID reports. Each interface has polling interval set to 4ms. Polling process is controlled by USB host and there is no guarantee that each interface will be polled exactly at this rate.

## USB Report Buffers:

InputStick provides FIFO buffer for each interface. Each buffer can store up to 32 HID reports of respective type.

InputStick will periodically send status update messages (more less every 100 ms). Such message contains information whether buffer is currently empty (at the time message was generated) and how many reports were received by USB host (and removed from the buffer) since last update.

When queuing reports to a respective report buffer, there must be enough free space to put ALL\* reports from packet into the buffer. If there is not enough free space to add ALL reports delivered in a single packet, no data will be added at all.

\*Exception from this rule is *CMD Queue mixed reports (0x25)* introduced in v0.98 firmware release. It is highly possible that this will be changed in next firmware version.

### Example:

| Event:  | Remaining buffer capacity: |         |            |
|---|----------------------------|---------|------------|
|   | (Keyboard)                 | (Mouse) | (Consumer) |
| InputStick initialization   | 32                         | 32      | 32         |
| Sent:<br>CMD Queue keyboard reports (0x21)<br>PARAM = 3   | 29                         | 32      | 32         |
| Sent:<br>CMD Queue keyboard reports (0x21)<br>PARAM = 6   | 23                         | 32      | 32         |
| Received:<br>CMD USB HID Status Update (0x2F)<br>Keyboard reports sent to host = 5<br>Mouse reports sent to host = 0<br>Consumer reports sent to host = 0 | 28                         | 32      | 32         |
| Sent:<br>CMD Queue consumer reports (0x22)<br>PARAM = 2   | 28                         | 32      | 30         |
| Sent:<br>CMD Queue mouse reports (0x23)<br>PARAM = 8  | 28                         | 24      | 30         |
| Sent:<br>CMD Queue keyboard reports (0x21)<br>PARAM = 3   | 25                         | 24      | 30         |
| Received:   | 32                         | 32      | 32         |

|   |  |  |  |
|---|--|--|--|
| <i>CMD</i> USB HID Status Update (0x2F)<br>Keyboard reports sent to host = 7<br>Mouse reports sent to host = 8<br>Consumer reports sent to host = 2 |  |  |  |
|---|--|--|--|

Remaining buffer capacity: at given moment, this is maximal number of HID reports that can be still added to the buffer.

If you want to keep your implementation as simple as possible, there are some ways to skip buffer management. Please keep in mind that this is generally not recommended and relies on the fact that USB host should poll each interface every 4 ms (what is not guaranteed, but should work in most cases).

1. Make each queue *CMD* at least 64 bytes long (*LENGTH* = 4,  $4 * 16B = 64B$ ) by adding dummy bytes (0x00s, remember to modify packet header, *INFO\_BYTE*). This will guarantee that no more than one *CMD* will be received during 4 ms period, since receiving 64B takes approximately 5 ms.  
You can adjust number of HID reports in a single packet and total packet length, for example: 3 HID reports per packet, total length 192B.
2. You can skip buffer management if your application won't send more than 32 reports of each kind, during each 128 ms period ( $32 * 4 \text{ ms}$ ). However due to Bluetooth latency, you should consider increasing this period to at least 1000 ms, preferably 2000 ms.
3. Set *RESP* flag. Check *RESP\_CODE* provided in response. If there was not enough free space and reports were dropped, sent the packet again.

All these methods will increase latency, typing speed will be reduced, mouse movements won't be smooth.