BLUETOOTH HID PROFILE

iWRAP APPLICATION NOTE

Tuesday, 09 October 2012

Version 1.7

VERSION HISTORY

Version	Comment
1.0	First version
1.1	iWRAP4 updates
1.2	Minor changes
1.3	Consumer page descriptions added
1.4	Multimedia key examples added
1.5	Complete revamp for iWRAP5
1.6	Restructured example descriptors
1.7	Improved examples

TABLE OF CONTENTS

1 Introduction

This application note discusses Bluetooth Human Interface Device (HID) Profile its advantages and how this profiles can be utilized. Also practical examples are given how the HID is used with the iWRAP firmware.

1.1 Human Interface Device Profile

The HID profile defines the protocols, procedures and features to be used by Bluetooth HID such as keyboards, pointing devices, gaming devices and remote monitoring devices.

The HID defines two roles, that of a Human Interface Device (HID) and a Host:

- Human Interface Device (HID) – The device providing the service of human data input and output to and from the host.

- Host – The device using or requesting the services of a Human Interface Device.

The HID profile uses the universal serial bus (USB) definition of a HID device in order to leverage the existing class drivers for USB HID devices. The HID profile describes how to use the USB HID protocol to discover a HID class device's feature set and how a Bluetooth enabled device can support HID services using the L2CAP layer. The HID profile is designed to enable initialization and control self-describing devices as well as provide a low latency link with low power requirements.

The Bluetooth HID profile is built upon the Generic Access Profile (GAP), specified in the Bluetooth Profiles Document; see Referenced Documents. In order to provide the simplest possible implementation, the HID protocol runs natively on L2CAP and does not reuse Bluetooth protocols other than the Service Discovery Protocol.

Source: [1]

HID device (mouse emulation)

HID device (keyboard emulation)

HID host

Figure 1: Typical HID use case

Bluegiga Technologies Oy

2 iWRAP firmware overview

iWRAP is an embedded firmware running entirely on the RISC processor of WT12, WT12 and WT32 modules. It implements the full *Bluetooth* protocol stack and many *Bluetooth* profiles as well. All software layers, including application software, run on the internal RISC processor in a protected user software execution environment known as a Virtual Machine (VM).

The host system can interface to iWRAP firmware through one or more physical interfaces, which are also shown in the figure below. The most common interfacing is done through the UART interface by using the ASCII commands that iWRAP firmware supports. With these ASCII commands, the host can access *Bluetooth* functionality without paying any attention to the complexity, which lies in the *Bluetooth* protocol stack. GPIO interface can be used for event monitoring and command execution. PCM, SPDIF, I2S or analog interfaces are available for audio. The available interfaces depend on the used hardware.

The user can write application code to the host processor to control iWRAP firmware using ASCII commands or GPIO events. In this way, it is easy to develop *Bluetooth* enabled applications.

On WT32 there is an extra DSP processor available for data/audio processing.
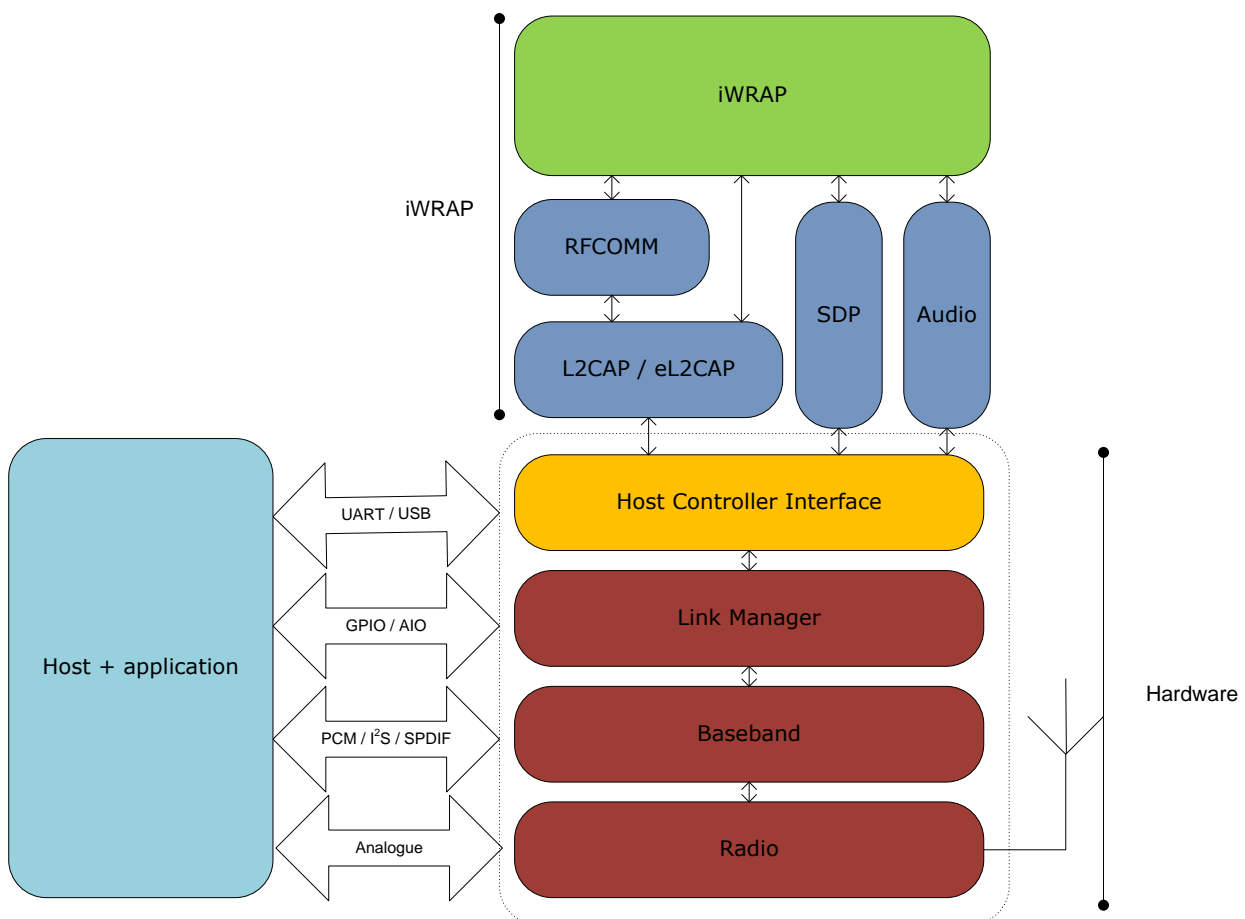
Figure 2: iWRAP Bluetooth stack

Bluegiga Technologies Oy

In the figure above, a Bluegiga *Bluetooth* module with iWRAP firmware could be connected to a host system for example through the UART interface. The options are:

- If the host system has a processor, software can be used to control iWRAP by using ASCII based commands or GPIO events.

- If there is no need to control iWRAP, or the host system does not need a processor, iWRAP can be configured to be totally transparent and autonomous, in which case it only accepts connections or automatically opens them.

- GPIO lines that WRAP THOR modules offer can also be used together with iWRAP to achieve additional functionality, such as Carrier Detect or DTR signaling.

- Audio interfaces can be used to transmit audio over a *Bluetooth* link.

Bluegiga Technologies Oy

3 Using HID with iWRAP

This chapter instructs the HID usage and configuration with the iWRAP firmware. ***Please note that the HID profile API has changed radically since iWRAP4.***

3.1 Profile configuration

HID is enabled and configured with

SET PROFILE HID {features} {subclass} {version} {country_code} {BTlang} {USBlang} {service_name}
where the fields are as follows:

Field	Description
features (uint8)	This bitmask indicates which optional features the HID device supports. **Bit 0**: Virtual Cable support. This essentially means saving generated link keys so pairing does not have to be done upon each connection setup, unless the Virtual Cable Unplug procedure is done. **Bit 1**: Reconnection Initiation support. This indicates to the HID Host that this device is capable of reconnecting to the Host. The Host shall not initiate reconnection if this is set. **Bit 2**: Normally Connectable. This means that the device can be connected to without any user intervention when it is powered on. **Bit 3**: Boot Device support. Setting this bit indicates that the device supports HID Boot Reports. ***Mandatory for keyboards and mice.*** The rest of the bits are reserved and shall be set to zero.
subclass (uint8)	Identifies the type of device. It should be set identical to the lowest 8 bits of the Class of Device, unless the device is a composite device that supports other profiles than just HID. Bits 0-1 must be set to zero.
version (uint16)	The version number of your device in binary-coded decimal (BCD) format. Example: 1203 = 12.0.3.
country_code (uint8)	Country code for localization information, for example the language of the key caps of a keyboard. Set to zero if not localized.
Btlang (char[2])	Language code in Bluetooth SDP format. Two ASCII characters as described by ISO 639:1988. Example: "en" is English.
USBlang (uint16)	Language code in USB HID format, as specified in Universal Serial Bus Language Identifiers (LANGIDs). 0409 is English.
service_name (string)	The service name the HID profile will advertise. The maximum length is 114 characters.

Because iWRAP registers its SDP entries at boot time, a reset is needed before changes made with SET PROFILE HID become effective.

To disable HID, issue **SET PROFILE HID** without any parameters.

3.2 HID descriptors configuration

In addition to the high-level information provided here, the HID descriptors themselves must be defined.

The following command is used to set the HID descriptors:

HID SET {*length*} {*descriptor*}

> **length**   parameter is a uint16 in hexadecimal format.
>
> **descriptor**   is the entire USB HID report descriptor in hexadecimal format.

To read the current HID descriptor the following command can be used:

HID GET

The response to the command is:

HID GET {*length*} {*descriptor*}

> **length**   parameter is a uint16 in hexadecimal format.
>
> **descriptor**   is the entire USB HID report descriptor in hexadecimal format.

3.3 Class-of-Device configuration

The Bluetooth class-of-device also needs to be configured properly. This can be done with the iWRAP command "**SET BT CLASS**". Usually, the Major Device Class must be set to Peripheral (0x000500).

The Minor Device Class field depends on which type of HID device your device is. 0x40 is a keyboard of any sort, 0x80 is a pointing device. If your device includes both types, the field is 0xc0.

Note that the SET PROFILE HID subclass must match the low 8 bits of the CoD (with bits 0-1 set to zero), unless it is a composite device that supports other profiles, which may add their own bits into the low 8 CoD bits.

Some examples:

SET BT CLASS 00540   : A keyboard device

SET BT CLASS 00580   : A pointing device

SET BT CLASS 005C0   : A combined keyboard + pointing device

SET BT CLASS 00500   : Not keyboard / not pointing device

3.4 Security configuration

The third configuration is related to pairing and security. iWRAP5 has Secure Simple Pairing enabled by default, as it is a mandatory feature for all Bluetooth 2.1 (and upwards) devices. iWRAP3 and older do not support SSP and legacy paring needs to be used.

To enable SSP two possible configurations can be used depending on the device type. The configuration is done with iWRAP command "**SET BT SSP**".

For a keyboard one should use setting:

> **SET BT SSP 2 0** **:** Enables SSP pairing for keyboard device, Man-in-the-Middle protection not required but can be used

For a mouse or any other device without a keyboard one should use setting:

> **SET BT SSP 3 0** : Enables SSP just works mode

To support pairing with older devices that do not implement SSP. Two other configurations should also be made. The Bluetooth PIN code should be enabled with "**SET BT AUTH * <pin>**" command and a so called interactive pairing mode should also be enabled with "**SET CONTROL CONFIG**" command.

Below is an example how to configure the security for a keyboard device supporting SSP.

```
SET BT AUTH * 0000
SET BT SSP 2 0
SET CONTROL CONFIG 800
RESET
```

Below is an example how to configure the security for a keyboard device without SSP support.

```
SET BT AUTH * 0000
SET CONTROL CONFIG 800
RESET
```

3.5 *Bluetooth* mouse example

Suppose our device is a Bluetooth mouse. It is capable of storing pairing information and can initiate connections on its own. It also supports Boot Reports, as is mandatory for all keyboards and mice. We do not want to bypass iWRAP parsing, so bits 4 and 5 of the first field are zero.

Its Class of Device must have the Peripheral Major Device Class (0x500) and Pointing Device Minor Device Class (0x80). The whole CoD is then 0x000580, and the HID Device Subclass must duplicate the CoD's Minor Device Class bits, 0x80.

This is the first release, so its product version is 1.0.0. It does not have any specific localization, as it is a mouse.

Our mouse has three buttons and a wheel. The buttons are either up or down, so they have a binary state. The range of the X- and Y-axis relative displacement is between -127 and 127. For the wheel it is between -15 and 15.

For a detailed explanation of the HID descriptor, please see Appendix A.

SET BT CLASS 000580

SET BT SSP 3 0

SET PROFILE HID d 80 100 0 en 409 Bluetooth Mouse

HID SET 3c
05010902a1010901a1008501050919012903150025019503750181020501093815f1250f950175058106050 1093009311581257f750895028106c0c0

RESET

3.6 Service discovery

Bluetooth technology enables wireless service discovery, so you can find out the capabilities the remote device supports. Wireless service discovery uses the Bluetooth Service Discovery Profile (SDP).

With iWRAP the service discovery is performed with command: "**SDP {*bd_addr*} {*uuid*}**".

        bd_addr   Bluetooth device address of the remote device.

        uuid   Universally unique identifier. Refers to the Bluetooth profile to be discovered. For HID the ***uuid*** is 1124.

Below is an example how to perform a service discovery for HID device.

SDP 00:07:80:FF:FF:FF 1124

SDP 00:07:80:ff:ff:ff < I SERVICENAME S "**HID**" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP I **11** > < U 0011 > > >

SDP

HID   = Service name

11   = L2CAP psm for HID profile

3.7 Pairing

The pairing must be initiated by the HID Host device such as a PC or a mobile phone, because the Host must read the HID Device's SDP entry in order to be able to parse its HID reports. The pairing may need actions on iWRAP, depending on which paring mode is used. If SSP pairing is used, no actions are needed on iWRAP, but with the legacy Bluetooth pairing also the interactive pairing mode needs to be enabled, and this requires user interaction.

When a device like a PC starts pairing with a keyboard it usually automatically displays a pin code that the user needs to type with a keyboard. This is the reason why interactive pairing is needed in iWRAP. The following example shows how the pairing procedure is made.

Interactive pairing example:

AUTH 00:21:86:35:c9:c8?	(Pairing is initiated from a PC and iWRAP shows **<u>AUTH</u>** event)
AUTH 00:21:86:35:c9:c8 12476505	(This is responded with AUTH command)

Pairing can also be initiated from iWRAP using the iWRAP command "**PAIR {*bd_addr*}**".

When using SSP, it is recommended to use **SET BT SSP 2 0** for keyboard-like devices (keypad capability, Man-in-the-Middle protection not required, but will be complied to if the remote end requests it), and **SET BT SSP 3 0** (no capabilities, no MITM) for mice, joysticks and similar devices with no display and yes/no button capability.

3.8 Connection establishment

Usually the HID connection is opened by the PC right after pairing. This can be seen by an incoming **RING** event generated by iWRAP.

Below is an example how a HID connection is received

```
RING 0 00:21:86:35:c9:c8 11 HID
RING 1 00:21:86:35:c9:c8 13 HID
```

However if a HID connection needs to be opened from iWRAP it can be done with a **CALL** command:

"**CALL {bd_addr} 11 HID**"

       bd_addr   Bluetooth device address of the remote device.

Below is an example how to set up a HID from iWRAP to a HID host device.

```
CALL 00:07:80:aa:bb:cc 11 HID
CALL 0
CONNECT 0 HID 11
CONNECT 1 HID 13
```

Two separate connections are established: the first is the control channel and the second is the data channel.

3.9 Connection termination

The HID data and control channels should be terminated on iWRAP using the "**DISCONNECT**" or "**CLOSE {link_id}**" command.

If the Virtual Cable Unplug procedure needs to be done, the command "**UNPLUG**" can be issued. It instructs the Host to delete all pairing information of the Device. iWRAP will delete its corresponding pairing information automatically.

HID connection termination:

CLOSE 0

NO CARRIER 1 ERROR 0

NO CARRIER 0 ERROR 0

3.10 Sending and receiving HID reports

HID can be used in either command/data mode or MUX mode. In command mode, HID commands can be issued to either of the HID links as long as they're selected as the active receiver of commands with **SET {link_id} SELECT**. In MUX mode, the link ID must be set to 0xff and either of the HID links must be the active receiver.

To send HID reports in command/data mode, either of the links must be selected with **SELECT {link_id}**. In MUX mode, HID reports can be sent to either of the links.

The escape character is disabled when the HID profile is used and a GPIO pin needs to be used for mode switching. This is because a user might want to transmit the escape (three '+' characters by default) sequence over the HID connection and this might lead to an unwanted iWRAP mode switch.

HID Output Reports sent by the Host will be presented in **HID {link_id} OUTPUT {data_length} {data}** events, unless the Raw Output bit is set in the HID configuration; in that case iWRAP will simply give a raw dump of the data received.

For backwards compatibility and ease of testing, iWRAP has built-in support for the example keyboard (see Appendix). The keyboard support requires that iWRAP's HID descriptor entry contains the example keyboard or a keyboard with a similar report format, or that Boot Reports are supported. The example keyboard reports are identical to Boot Reports, so they can be used even in the case the Host indicates that it wants to use Boot Reports.

Example:

```
RING 0 00:21:86:35:c9:c8 11 HID

RING 1 00:21:86:35:c9:c8 13 HID

HID 0 OUTPUT 01 00 (Host indicates that Caps Lock, Num Lock etc. are off)

SELECT 0

abcdefg
```

3.11 HID raw reports

All HID functionality outside of the example/Boot keyboard must be implemented with raw reports. All raw reports begin with the hexadecimal byte 0x9f followed by the length byte (length excluding the 0x9f and the length byte themselves).

Example keyboard report:

0x9f	0x0a	0xa1	report ID	modifier	0x00	key code 1	key code 2	key code 3	key code 4	key code 5	key code 6

Figure 3: Raw HID keyboard report

Example mouse report:

0x9f	0x05	0xa1	report ID	buttons	x-step	y-step

Figure 4: Raw HID mouse report

Example consumer page report:

0x9f	0x05	0xa1	report ID	bitfield 1	bitfield 2	bitfield 3

Figure 5: Raw HID consumer report (iWRAP 4.1.0 and later)

Bitfield 1:

0x01 Volume Increment
0x02 Volume Decrement
0x04 Mute
0x08 Play/Pause
0x10 Scan Next Track
0x20 Scan Previous Track
0x40 Stop
0x80 Eject

Bitfield 2:

0x01 Email Reader
0x02 Application Control Search
0x04 AC Bookmarks
0x08 AC Home
0x10 AC Back
0x20 AC Forward
0x40 AC Stop
0x80 AC Refresh

Bitfield 3:

0x01 Application Launch Generic Consumer Control
0x02 AL Internet Browser
0x04 AL Calculator
0x08 AL Terminal Lock / Screensaver
0x10 AL Local Machine Browser
0x20 AC Minimize
0x40 Record
0x80 Rewind

Bluegiga Technologies Oy

Full key codes can be found from document USB HID Usage Tables:

http://www.usb.org/developers/devclass_docs/Hut1_11.pdf

Modifier key codes can be found from document HID1_11.pdf (page 56)

http://www.usb.org/developers/devclass_docs/HID1_11.pdf

Example of transmitting a key press-release sequence for 'a' key:

> **0x9f 0x0a 0xa1 0x01 0x00 0x00 0x04 0x00 0x00 0x00 0x00 0x00**
>
> **0x9f 0x0a 0xa1 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00**

Example of transmitting a key press-release sequence for 'a' key while holding left shift down

> **0x9f 0x0a 0xa1 0x01 0x02 0x00 0x04 0x00 0x00 0x00 0x00 0x00**
>
> **0x9f 0x0a 0xa1 0x01 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00**

0x9f is used to indicate that a raw HID report is sent. Then **0x0a** indicates the length of the data to follow (10 bytes) and then the next 10 bytes are the raw report as described in the Keyboard descriptor. Note that the example Keyboard descriptor format is identical to the Boot Keyboard in the USB HID specification.

To indicate a user has simultaneously pressed multiple keys (up to six different keys) you can use the rest of the key codes. **Note:** to send key combinations which include modifier keys (ctrl, alt, shift, GUI), the corresponding modifier bit must be used, not the key code for the modifier key. For example, Left Alt + Tab:

> **0x9f 0x0a 0xa1 0x01 0x04 0x00 0x2b 0x00 0x00 0x00 0x00 0x00**

Launching the calculator application using the Consumer Report descriptor:

> **0x9f 0x05 0xa1 0x02 0x00 0x00 0x04**
>
> **0x9f 0x05 0xa1 0x02 0x00 0x00 0x00**

The first raw frame indicates that the calculator button is pushed down and the second frame indicates that it is released.

Play/pause report:

> **0x9f 0x05 0xa1 0x02 0x08 0x00 0x00**
>
> **0x9f 0x05 0xa1 0x02 0x00 0x00 0x00**

Bluegiga Technologies Oy

Next track:

> **0x9f 0x05 0xa1 0x02 0x10 0x00 0x00**
>
> **0x9f 0x05 0xa1 0x02 0x00 0x00 0x00**

Mouse movement examples

> **\x9f\x05\xa1\x01\x00\x05\x00   move 5px right**
>
> **\x9f\x05\xa1\x01\x00\xff\x00   move 1px left**

The above uses the following report:

9f 05 a1 01 [buttons+wheel] [x] [y]

x and y steps (IIRC):

00-7F corresponds to +0…+127

FF-80 corresponds to -1…-127

3.12 Power saving

iWRAP offers two power saving options. Sniff mode, which can be used to save power for active Bluetooth connections and deep sleep more which puts the internal processor into a reduced duty cycle mode. Please refer to iWRAP user guide for more information about sniff and deep sleep modes.

One should also know that when Bluetooth connections are in active mode i.e. no power saving in use the master device uses 3-4 times less power then a slave device. Therefore for battery powered applications it might be useful to configure the device as a master rather then a slave, eventually considering role switching.

For HID devices the Bluetooth 2.1 + EDR introduced a new sniff mode called sniff sub rating. In brief the sniff sub rate mode increases the sniff interval after a certain period of inactivity and therefore reduces the current consumption. iWRAP4 supports sniff sub rating mode. Please refer into iWRAP4 user guide for more information.

4 Example connection diagram

An example of HID configuration and a simple HID connection setup is illustrated below.
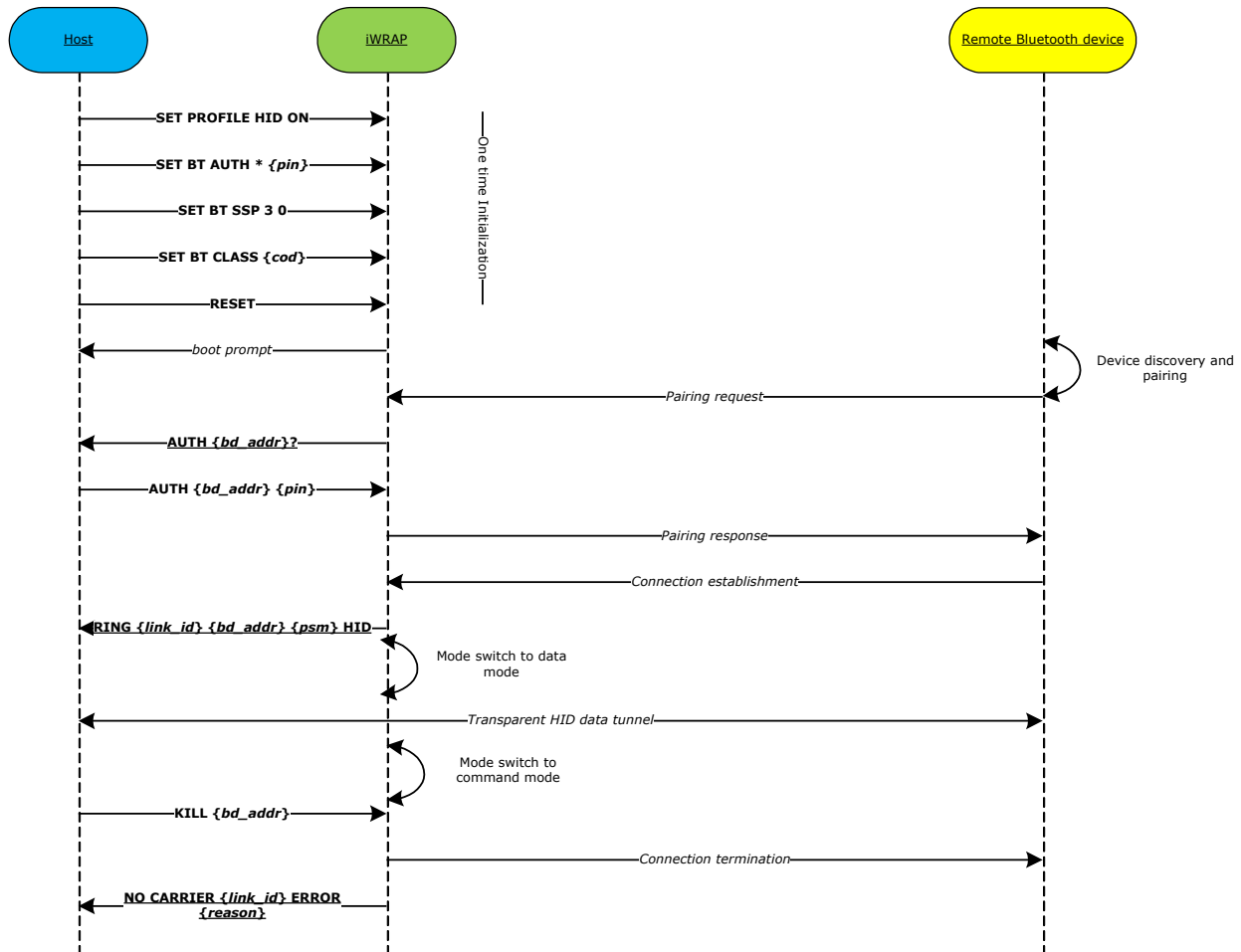
Figure 6: HID connection example

In the above example the **AUTH** event is only seen if a legacy pairing with pin code and interactive paring is made. If the legacy pairing does not take place and SSP pairing is used instead then **AUTH** event is not seen and there is no need to reply to it with the **AUTH** command. With SSP pairing, depending on the SSP mode, however SPP events may be displayed and then need to be responded with correct SSP pairing commands.

Also depending on the remote Bluetooth device one or two **RING** events may been seen during the connection establishment.

Please refer to iWRAP user guide for more information about the iWRAP command and events.

5 Known issues

After pairing with the Windows stack, when the Windows Host initiates first connection, it is possible to lose Input Reports sent from the Device to the Host when sending them very rapidly. A feasible workaround is to either initiate a Role Switch, or disconnect and reconnect the connection so that iWRAP is the master of the connection. Any connection beyond the initial pairing is usually unaffected, since in almost all cases the Device is the one initiating the connection.

6 Appendix

6.1 Example keyboard layout

The example HID keyboard layout is US and the following key codes are supported. These codes are transmitted over the HID data channel to the HID host device such as a PC.

Code	Description
00	Left control + space
01	Left control + a
02	Left control + b
03	Left control + c
04	Left control + d
05	Left control + e
06	Left control + f
07	Left control + g
08	Backspace
09	Tab
0a	Enter
0b	Left control + k
0c	Left control + l
0d	Enter
e0	Left control + n
0f	Left control + o
10	Left control + p

Bluegiga Technologies Oy

11	Left control + q
12	Left control + r
13	Left control + s
14	Left control + t
15	Left control + u
16	Left control + v
17	Left control + w
18	Left control + x
19	Left control + y
1a	Left control + z
1b	Left control + space
1c	Esc
1d-1f	N/A
20-7e	Corresponding ASCII character
7f	backspace
80	Cursor up
81	Cursor right
82	Cursor down
83	Cursor left
84	Insert
85	Delete
86	Home

87	End
88	Page up
89	Page down
8a-9e	-
9f	raw mode*
a0-ff	-

Table 1: Available HID key codes

6.2 Example descriptors

6.2.1 Example keyboard descriptor

```
static const uint8 hid_descriptor_keyboard[] = {
  /****/ 0x05, 0x01,     /* USAGE_PAGE (Generic Desktop) */
  /****/ 0x09, 0x06,     /* USAGE (Keyboard) */
  /****/ 0xa1, 0x01,     /* COLLECTION (Application) */
  /******/ 0x05, 0x07,   /*   USAGE_PAGE (Keyboard) */
  /******/ 0x85, 0x01,   /*   REPORT_ID (1) */

  /* Ctrl, Shift and other modifier keys, 8 in total */
  /******/ 0x19, 0xe0,   /*   USAGE_MINIMUM (kbd LeftControl) */
  /******/ 0x29, 0xe7,   /*   USAGE_MAXIMUM (kbd Right GUI) */
  /******/ 0x15, 0x00,   /*   LOGICAL_MINIMUM (0) */
  /******/ 0x25, 0x01,   /*   LOGICAL_MAXIMUM (1) */
  /******/ 0x75, 0x01,   /*   REPORT_SIZE (1) */
  /******/ 0x95, 0x08,   /*   REPORT_COUNT (8) */
  /******/ 0x81, 0x02,   /*   INPUT (Data,Var,Abs) */

  /* Reserved byte */
  /******/ 0x95, 0x01,   /*   REPORT_COUNT (1) */
  /******/ 0x75, 0x08,   /*   REPORT_SIZE (8) */
  /******/ 0x81, 0x01,   /*   INPUT (Cnst,Ary,Abs) */

  /* LEDs for num lock etc */
  /******/ 0x95, 0x05,   /*   REPORT_COUNT (5) */
  /******/ 0x75, 0x01,   /*   REPORT_SIZE (1) */
  /******/ 0x05, 0x08,   /*   USAGE_PAGE (LEDs) */
  /******/ 0x85, 0x01,   /*   REPORT_ID (1) */
  /******/ 0x19, 0x01,   /*   USAGE_MINIMUM (Num Lock) */
  /******/ 0x29, 0x05,   /*   USAGE_MAXIMUM (Kana) */
  /******/ 0x91, 0x02,   /*   OUTPUT (Data,Var,Abs) */

  /* Reserved 3 bits */
  /******/ 0x95, 0x01,   /*   REPORT_COUNT (1) */
  /******/ 0x75, 0x03,   /*   REPORT_SIZE (3) */
  /******/ 0x91, 0x03,   /*   OUTPUT (Cnst,Var,Abs) */

  /* Slots for 6 keys that can be pressed down at the same time */
  /******/ 0x95, 0x06,   /*   REPORT_COUNT (6) */
  /******/ 0x75, 0x08,   /*   REPORT_SIZE (8) */
  /******/ 0x15, 0x00,   /*   LOGICAL_MINIMUM (0) */
  /******/ 0x25, 0x65,   /*   LOGICAL_MAXIMUM (101) */
```

Bluegiga Technologies Oy

```
  /******/ 0x05, 0x07,   /*   USAGE_PAGE (Keyboard) */
  /******/ 0x19, 0x00,   /*   USAGE_MINIMUM (Reserved (no event indicated)) */
  /******/ 0x29, 0x65,   /*   USAGE_MAXIMUM (Keyboard Application) */
  /******/ 0x81, 0x00,   /*   INPUT (Data,Ary,Abs) */
  /****/ 0xc0,           /* END_COLLECTION */
};
```

6.2.2 Example mouse descriptor

```
static const uint8 hid_descriptor_mouse\[\] = {
/****/ 0x05, 0x01,     /\* USAGE_PAGE (Generic Desktop) \*/
/****/ 0x09, 0x02,     /\* USAGE (Mouse) \*/
/****/ 0xa1, 0x01,     /\* COLLECTION (Application) \*/
/******/ 0x09, 0x01,   /\*   USAGE (Pointer) \*/
/******/ 0xa1, 0x00,   /\*   COLLECTION (Physical) \*/
/********/ 0x85, 0x01, /\*     REPORT_ID (1) \*/
/********/ 0x05, 0x09, /\*     USAGE_PAGE (Button) \*/
/********/ 0x19, 0x01, /\*     USAGE_MINIMUM (Button 1) \*/
/********/ 0x29, 0x03, /\*     USAGE_MAXIMUM (Button 3) \*/
/********/ 0x15, 0x00, /\*     LOGICAL_MINIMUM (0) \*/
/********/ 0x25, 0x01, /\*     LOGICAL_MAXIMUM (1) \*/
/********/ 0x95, 0x03, /\*     REPORT_COUNT (3) \*/
/********/ 0x75, 0x01, /\*     REPORT_SIZE (1) \*/
/********/ 0x81, 0x02, /\*     INPUT (Data,Var,Abs) \*/
/********/ 0x05, 0x01, /\*     USAGE_PAGE (Generic Desktop) \*/
/********/ 0x09, 0x38, /\* USAGE (Wheel) \*/
/********/ 0x15, 0xf1, /\*                 LOGICAL_MINIMUM (-15) \*/
/********/ 0x25, 0x0f, /\* LOGICAL_MAXIMUM (15) \*/
/********/ 0x95, 0x01, /\*     REPORT_COUNT (1) \*/
/********/ 0x75, 0x05, /\*     REPORT_SIZE (5) \*/
/********/ 0x81, 0x06, /\*     INPUT (Data,Var,Rel) \*/
/********/ 0x05, 0x01, /\*     USAGE_PAGE (Generic Desktop) \*/
/********/ 0x09, 0x30, /\*     USAGE (X) \*/
/********/ 0x09, 0x31, /\*     USAGE (Y) \*/
/********/ 0x15, 0x81, /\*     LOGICAL_MINIMUM (-127) \*/
/********/ 0x25, 0x7f, /\*     LOGICAL_MAXIMUM (127) \*/
/********/ 0x75, 0x08, /\*     REPORT_SIZE (8) \*/
/********/ 0x95, 0x02, /\*     REPORT_COUNT (2) \*/
/********/ 0x81, 0x06, /\*     INPUT (Data,Var,Rel) \*/
/******/ 0xc0,         /\*   END_COLLECTION \*/
/****/ 0xc0,           /\* END_COLLECTION \*/
};
```

Bluegiga Technologies Oy

6.2.3 Example consumer page descriptor

```
static const uint8 hid_descriptor_consumer_page[] = {
  /****/ 0x05, 0x0c,                   /* USAGE_PAGE (Consumer) */
  /****/ 0x09, 0x01,                   /* USAGE (Consumer Control) */
  /****/ 0xa1, 0x01,                   /* COLLECTION (Application) */
  /******/ 0x85, 0x02,               /* Report ID 2 */
  /******/ 0x05, 0x0c,               /* USAGE_PAGE (Consumer) */


  /* 8 media player related keys */
  /******/ 0x15, 0x00,     /* Logical Min 0 */
  /******/ 0x25, 0x01,     /* Logical Max 1 */
  /******/ 0x09, 0xe9,     /* Usage (8-bit), Volume Increment */
  /******/ 0x09, 0xea,     /* Usage (8-bit), Volume Decrement */
  /******/ 0x09, 0xe2,     /* Usage (8-bit), Mute */
  /******/ 0x09, 0xcd,     /* Usage (8-bit), Play/Pause */
  /******/ 0x19, 0xb5,     /* Usage Min (Scan Next Track, Scan Previous Track, Stop, Eject) */
  /******/ 0x29, 0xb8,     /* Usage Max */
  /******/ 0x75, 0x01,     /* Report Size */
  /******/ 0x95, 0x08,     /* Report Count */
  /******/ 0x81, 0x02,     /* Input type 2 */


  /* 8 application control keys */
  /******/ 0x0a, 0x8a, 0x01,/* Usage (16-bit), LSB first, e.g. this is 0x018a Email Reader */
  /******/ 0x0a, 0x21, 0x02,/* Usage (16-bit), Generic GUI Application Control Search */
  /******/ 0x0a, 0x2a, 0x02,/* Usage (16-bit), Application Control Bookmarks */
  /******/ 0x1a, 0x23, 0x02,/* Usage Min (16-bit), AC Home, Back, Forward, Stop, Refresh */
  /******/ 0x2a, 0x27, 0x02,/* Usage Max (16-bit) */
  /******/ 0x75, 0x01,     /* Report Size */
  /******/ 0x95, 0x08,     /* Report Count */
  /******/ 0x81, 0x02,             /* Input type 2 */


  /* Application launch keys + record & rewind */
  /******/ 0x0a, 0x83, 0x01,/* Usage (16-bit), Application Launch Generic Consumer Control */
  /******/ 0x0a, 0x96, 0x01,/* Usage (16-bit), AL Internet Browser */
  /******/ 0x0a, 0x92, 0x01,/* Usage (16-bit), AL Calculator */
  /******/ 0x0a, 0x9e, 0x01,/* Usage (16-bit), AL Terminal Lock / Screensaver */
  /******/ 0x0a, 0x94, 0x01,/* Usage (16-bit), AL Local Machine Browser */
  /******/ 0x0a, 0x06, 0x02,/* Usage (16-bit), AC Minimize */
  /******/ 0x09, 0xb2,     /* Usage (8-bit), Record */
  /******/ 0x09, 0xb4,     /* Usage (8-bit), Rewind */
  /******/ 0x75, 0x01,     /* Report Size */
  /******/ 0x95, 0x08,     /* Report Count */
  /******/ 0x81, 0x02,     /* Input type 2 */
```

Bluegiga Technologies Oy

```
  /******/ 0xc0,                                       /* End Collection */
};
```

6.2.4 Example joystick descriptor

```c
static const uint8 hid_descriptor_joystick[] = {
  0x05, 0x01,                    /* USAGE_PAGE (Generic Desktop) */
    0x09, 0x04,                  /* USAGE (Joystick) */
    0x85, 0x01,                       /* Report ID 1 */
    0xa1, 0x01,                  /* COLLECTION (Application) */
    0x05, 0x01,                  /*   USAGE_PAGE (Generic Desktop) */
    0x09, 0x01,                  /*   USAGE (Pointer) */
    0xa1, 0x00,                  /*   COLLECTION (Physical)*/
    0x15, 0x81,                  /*     LOGICAL_MINIMUM (-127)*/
    0x25, 0x7f,                  /*     LOGICAL_MAXIMUM (127)*/
    0x75, 0x08,                  /*     REPORT_SIZE (8)*/
    0x95, 0x02,                  /*     REPORT_COUNT (2)*/
    0x09, 0x30,                  /*     USAGE (X)*/
    0x09, 0x31,                  /*     USAGE (Y)*/
    0x81, 0x02,                  /*     INPUT (Data,Var,Abs)*/
    0x09, 0x39,                  /*     USAGE (Hat switch)*/
    0x15, 0x00,                  /*     LOGICAL_MINIMUM (0)*/
    0x25, 0x03,                  /*     LOGICAL_MAXIMUM (3)*/
    0x35, 0x00,                  /*     PHYSICAL_MINIMUM (0)*/
    0x46, 0x0e, 0x01,            /*     PHYSICAL_MAXIMUM (270)*/
    0x65, 0x14,                  /*     UNIT (Eng Rot:Angular Pos)*/
    0x95, 0x01,                  /*     REPORT_COUNT (1)*/
    0x75, 0x04,                  /*     REPORT_SIZE (4)*/
    0x81, 0x42,                  /*     INPUT (Data,Var,Abs,Null)*/
    0x15, 0x00,                  /*     LOGICAL_MINIMUM (0)*/
    0x25, 0x01,                  /*     LOGICAL_MAXIMUM (1)*/
    0x95, 0x02,                  /*     REPORT_COUNT (2)*/
    0x75, 0x01,                  /*     REPORT_SIZE (1)*/
    0x05, 0x09,                  /*     USAGE_PAGE (Button)*/
    0x19, 0x01,                  /*     USAGE_MINIMUM (Button 1)*/
    0x29, 0x02,                  /*     USAGE_MAXIMUM (Button 2)*/
    0x65, 0x00,                  /*     UNIT (None)*/
    0x81, 0x02,                  /*     INPUT (Data,Var,Abs)*/
    0xc0,                        /*     END_COLLECTION*/
    0x19, 0x03,                  /*   USAGE_MINIMUM (Button 3)*/
    0x29, 0x04,                  /*   USAGE_MAXIMUM (Button 4)*/
    0x81, 0x02,                  /*   INPUT (Data,Var,Abs)*/
    0x05, 0x02,                  /*   USAGE_PAGE (Simulation Controls)*/
    0x09, 0xbb,                  /*   USAGE (Throttle)*/
    0x15, 0x81,                  /*   LOGICAL_MINIMUM (-127)*/
    0x25, 0x7f,                  /*   LOGICAL_MAXIMUM (127)*/
    0x75, 0x08,                  /*   REPORT_SIZE (8)*/
```

Bluegiga Technologies Oy

```
    0x95, 0x01,                   /*   REPORT_COUNT (1)*/
    0x81, 0x02,                   /*   INPUT (Data,Var,Abs)*/
    0xc0,                         /* END_COLLECTION*/
};
```

7 References

[1]   The Bluetooth SIG, Human Interface Device Profile overview, URL:
http://www.bluetooth.com/Bluetooth/Technology/Works/HID.htm

Bluegiga Technologies Oy

8 Contact Information

Sales: sales@bluegiga.com

Technical support: support@bluegiga.com

                        http://www.bluegiga.com/techforum/

Orders: orders@bluegiga.com

Head Office / Finland:

                        Phone: +358-9-4355 060

                        Fax: +358-9-4355 0660

Street Address:

                        Sinikalliontie 5A

                        02630 ESPOO

                        FINLAND

Postal address:

                        P.O. BOX 120

                        02631 ESPOO

                        FINLAND

Sales Office / USA:

                        Phone: (781) 556-1039

                        Bluegiga Technologies, Inc.

                        99 Derby Street, Suite 200 Hingham, MA 02043

Bluegiga Technologies Oy