# iWRAP4

USER GUIDE

MONDAY, 02 JANUARY 2012

Version 4.1

**blue**giga

## VERSION HISTORY

| Version | Comment |
|---------|---------|
| 1.0 | First draft |
| 1.1 | PBAP documentation added |
| 1.2 | Added PBAP to CALL command |
| 1.3 | Added OBEX OPP and FTP examples |
| 1.4 | Improved PBAP description and added OBEX AUTH event |
| 1.5 | Added OBEX headers into PBAP documentation |
| 1.6 | iWRAP3 merge started |
| 1.7 | More commands documented |
| 1.8 | Minor fixes to links and styles |
| 1.9 | Events, errors and known issues added |
| 2.0 | Examples added |
| 2.1 | Known issues updated |
| 2.2 | HFP-AG events updated |
| 2.3 | SET CONTROL PREAMP documentation added, CLOCK event fixed |
| 2.4 | Updated SET CONTROL BIND documentation |
| 2.5 | Fixed the number of maximum simultaneous connections |
| 2.6 | Chapter 5.1 updated |
| 2.7 | SET CONTROL READY added |
| 2.8 | SET CONTROL BATTERY is only for WT32 |
| 2.9 | iWRAP4 command updates |
| 3.0 | SET CONTROL CODEC description added |
| 3.1 | SSP section added |

| 3.2 | iWRAP 4.0 QDID added |
|-----|----------------------|
| 3.3 | iWRAP4 release changes and updates |
| 3.4 | CONNAUTH command and event added. Small corrections. Chapter 9 improved. |
| 3.5 | SSP requirement added to CONNAUTH documentation. |
| 3.6 | @ command documented |
| 3.7 | SET RESET added. Minor fixes. |
| 3.8 | Removed HID example |
| 3.9 | HID mouse report fixed |
| 4.0 | PBAP command description improved |
| 4.1 | SSP COMPLETE event added and SET BT BDADDR clarified |

# TABLE OF CONTENTS

# 1 Introduction

iWRAP is an embedded firmware running entirely on the RISC processor of WT12, WT11, WT41 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.
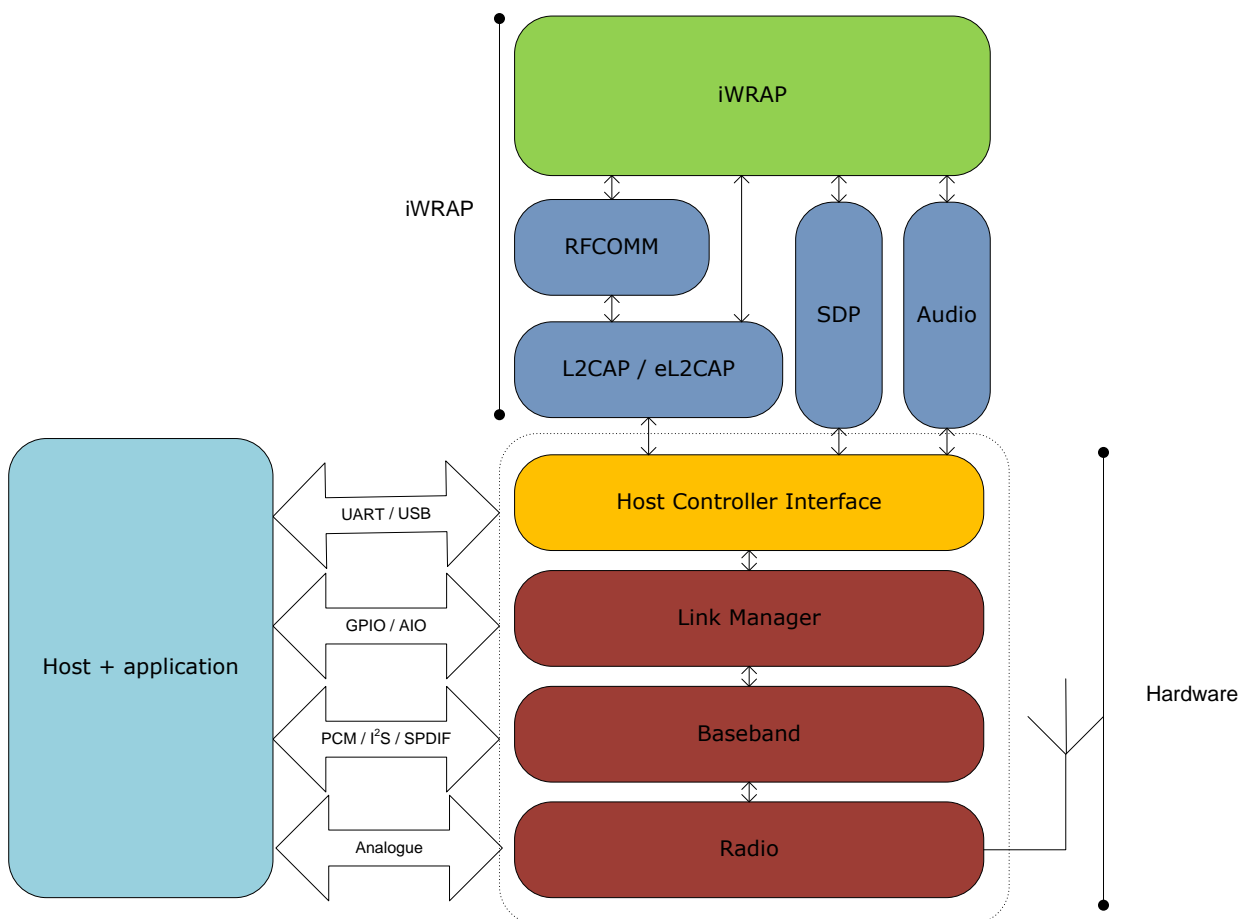
**Figure 1: iWRAP Bluetooth stack**

In the figure above, a 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 Bluegiga's Bluetooth 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.

# 2  Getting started

To start using iWRAP firmware, you can use, for example, terminal software such as HyperTerminal. When using the terminal software, make sure that the Bluetooth module is connected to your PC's serial port. By default, iWRAP uses the following UART settings:

- Baud rate:            115200bps

- Data bits:            8

- Stop bits:            1

- Parity bit:           No parity

- HW Flow Control:      Enabled

When you power up your Bluetooth module or evaluation kit, you can see the boot prompt appear on the screen of the terminal software. After the "**READY**." event iWRAP firmware is ready to be used.
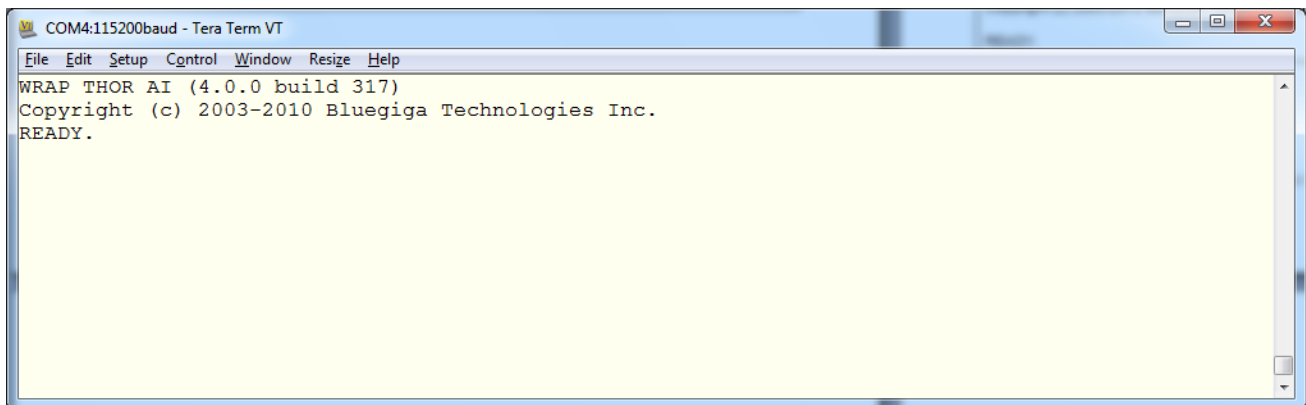


**Figure 2: iWRAP boot prompt**

If no READY. event is received the possible reasons are:

- The Bluetooth module is not equipped with iWRAP firmware, but HCI firmware

- The UART logic levels are incorrect

- Boot prompt is disabled with "**SET CONTROL ECHO 0**" setting

## 2.1 First course to iWRAP

A few very basic iWRAP usage examples are presented below. Just a few very basic use cases are shown and more detailed examples will be presented later in this user guide.

**AT** command can be sent to iWRAP to test that the firmware is operational. An OK response tells that iWRAP is functional.

```
AT
OK
```

**SET** command displays the settings of the local Bluetooth device.

```
SET
SET BT BDADDR 00:07:80:ff:ff:f1
SET BT NAME WT32-A
SET BT CLASS 001f00
SET BT IDENT BT:47 f000 4.0.0 Bluegiga iWRAP
SET BT LAP 9e8b33
SET BT PAGEMODE 4 2000 1
SET BT POWER 0 0 0
SET BT ROLE 0 f 7d00
SET BT SNIFF 0 20 1 8
SET BT MTU 667
SET CONTROL BAUD 115200,8n1
SET CONTROL CD 00 0
SET CONTROL ECHO 7
SET CONTROL ESCAPE 43 00 1
SET CONTROL GAIN 8 8
SET CONTROL MSC DTE 00 00 00 00 00 00
SET CONTROL READY 00
SET PROFILE SPP Bluetooth Serial Port
SET
```

**INQUIRY** command can be used to discover other visible Bluetooth devices in the range. An **INQUIRY_PARTIAL** event is generated as soon as a device is discovered and finally is summary is displayed.

```
INQUIRY 5
INQUIRY_PARTIAL 00:21:86:35:c9:c8 02010c
INQUIRY_PARTIAL 00:07:80:93:d7:66 240408
INQUIRY_PARTIAL a8:7b:39:c3:ca:99 5a020c
INQUIRY 3
INQUIRY 00:21:86:35:c9:c8 02010c
INQUIRY 00:07:80:93:d7:66 240408
INQUIRY a8:7b:39:c3:ca:99 5a020c
```

**SET** commands can be used to modify the settings of the local Bluetooth device. In the example below Bluetooth PIN code required for pairing is set to "0000" and also the Secure Simple Pairing (SSP) "just works" mode is enabled. The settings are stored on a local non-volatile memory so they need to be configured only once. **With iWRAP4 it is strongly recommended that SSP is enabled by default in all applications.**

```
SET BT AUTH * 0000

SET BT SSP 3 0
```

A Bluetooth connection is opened with a **CALL** command. A **CALL** event indicates that a connection establishment is in progress and a **CONNECT** event indicates a successful connection.

```
CALL 00:07:80:93:d7:66 1101 RFCOMM

CALL 0

CONNECT 0 RFCOMM 1
```

A **SET RESET** command can be used to return the factory level settings. iWRAP is reset as indicated by the boot prompt.

```
SET RESET

WRAP THOR AI (4.0.0 build 317)

Copyright (c) 2003-2010 Bluegiga Technologies Inc.

READY.
```

Bluegiga Technologies Oy

# 3  iWRAP modes

iWRAP has two basic operational modes, **command mode** and **data mode**. In command mode, ASCII commands can be given to iWRAP firmware to perform various actions or to change configuration settings. Command mode is the default mode when there are no Bluetooth connections. Data mode, on the other hand, is used to transmit and receive data over a Bluetooth link. Data mode is only available if there is a Bluetooth connection. It is possible to switch between modes at any time assuming the conditions for data mode are fulfilled. The mode transitions are illustrated below.



- CONNECT event
- RING event
- Escape sequence
- SELECT command

- NO CARRIER event
- Escape sequence
- DTR switch
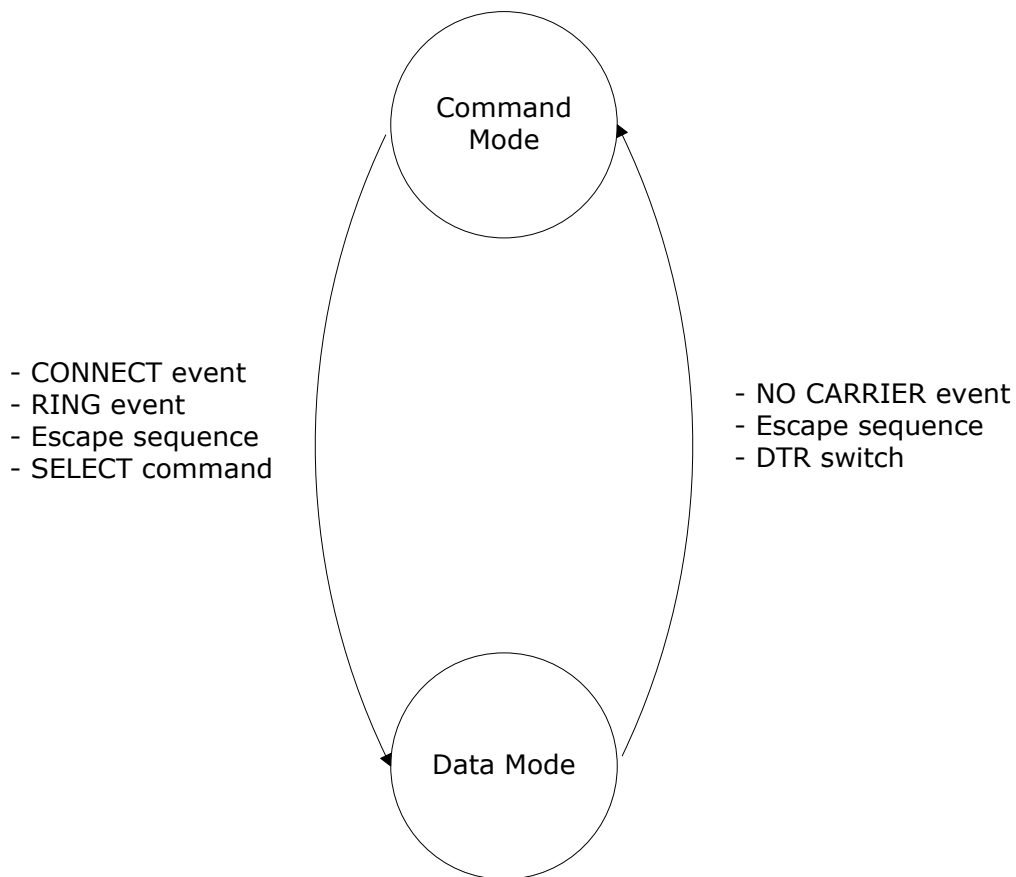
**Figure 3: Mode transitions**

| Initial mode | Target mode | Requirements for state transition |
|---|---|---|
| Command Mode (no Bluetooth connections)<br><br>In this mode, ASCII commands can be given to iWRAP. | Data Mode | A connection is successfully created by using the **CALL** command and **CONNECT** event indicating that a successful connection is received.<br><br>A remote device opens a Bluetooth connection to iWRAP. A **RING** event indicating that a connection is received.<br><br>If iWRAP events are disabled the **carrier detect** (CD) pin can also be used to indicate data or command mode. |
| Data Mode<br><br>In this mode, all data is sent transparently from UART interface to Bluetooth connection. | Command Mode | The user switches mode by sending an **escape sequence** to iWRAP firmware or by toggling the DTR pin.<br><br>A link is terminated (closed by the remote device or by link loss) and **NO CARRIER** event is received. |
| Command Mode (active connection)<br><br>In this mode, ASCII commands can be given to iWRAP. | Data Mode | User switches the mode either by sending an escape sequence to iWRAP firmware or by using the **SELECT** command. |

**Table 1: iWRAP mode transitions explained**

## 3.1 The escape sequence

The escape sequence causes the iWRAP firmware to toggle between command mode and data mode. The escape sequence consists of three (3) escape characters that are defined by the **SET CONTROL ESCAPE** command. By default, the escape character is '+'.

Do not enter any character before and/or after the escape sequence for a guard time, which is 1 second. Furthermore, send the escape characters individually, not as a string.

With default settings, the escape sequence is:

**< 1 second sleep> +++ < 1 second sleep>**

When a successful state transition from data mode to command mode is made, iWRAP sends a "**READY.**" event to indicate that it is ready to receive commands.

The same escape sequence or the **SELECT** command can be used to return to data mode.

## 3.2  Command mode

The command mode is the default mode when iWRAP is powered up. In command mode, ASCII commands can be entered to iWRAP to perform various functions.

**Notes:**

- In command mode, if there are active Bluetooth connections, the data from remote devices is buffered into iWRAP buffers.

- Because of the embedded nature of iWRAP, buffering capabilities are low and only small amounts of data can be received to buffers. The amount of data which can be buffered depends on the firmware version and the state of iWRAP. Usually, it is around 1000 bytes, but may vary radically.

- **The LIST** command shows active connections and the amount of buffered data.

## 3.3  Data mode

Data mode is the default mode when there are one or more Bluetooth connections. In data mode, all data is sent transparently from UART interface to the Bluetooth link and vice versa.

**Notes:**

- When iWRAP enters command mode from data mode, a "READY" event occurs, unless events are masked away by using the "SET CONTROL ECHO" command.

- The DTR pin can be used instead of the escape sequence to switch from data mode to command mode. This allows much faster mode switching and no guard time is needed. The DTR pin can be enabled by using the "**SET CONTROL ESCAPE**" command.

- When enabled, the DTR line can be configured also for closing the active connection or for a reset.

- The Carrier Detect (CD) pin can be used to indicate either a Bluetooth connection or data mode. The CD pin can be enabled and configured by using the "**SET CONTROL CD**" command.

- The "**SET CONTROL BIND**" command can be used in conjunction with the "**SET CONTROL ESCAPE**" command to allow data-command-data mode switches with the same GPIO line; consider in fact the following commands together: "SET CONTROL ESCAPE - 20 1" and "SET CONTROL BIND 0 20 F SELECT 0"

## 3.4  Multiplexing mode

In iWRAP version 2.1.0 and newer, there is a special mode called multiplexing mode. In this mode, iWRAP does not have separate commands or data modes, but data, commands and events are all handled in one single mode. There is, however, a special protocol to separate commands and events from the actual data. This protocol must be used between the host system and iWRAP firmware.

The advantage of multiplexing mode is that several Bluetooth connections can be handled simultaneously and there is no need to do time consuming data-command-data mode switching. However, the downside is that the performance of iWRAP is reduced, since the firmware needs to handle the multiplexing protocol and it causes overhead.

To learn more about multiplexing mode, see the description of the "**SET CONTROL MUX**" command.

## 3.5  HFP and HSP modes

iWRAP 2.2.0 and newer support Bluetooth Hands-Free (v.1.5) profile. This profile includes a lot of control messaging and events, which are handled in command mode. In other words, when a HFP connection is opened or received no state transition occurs, but iWRAP stays in command mode, where all HFP messaging is done. Refer to HFP profile usage for more information.

## 3.6  OBEX mode

IWRAP4 and newer versions support Bluetooth Object Push Profile (OPP) or File Transfer Protocol (FTP) modes. The operation in this mode is quite similar to HFP mode. For example, there are no separate command and data modes, but iWRAP always stays in command mode. Refer to OPP and FTP profile usage for more information.

## 3.7  A2DP mode

As of iWRAP3, Bluetooth Advanced Audio Distribution Profile (A2DP) is supported. This profile also includes control messaging and events, which are handled in command mode. In other words, when an A2DP connection is opened or received no state transition occurs, but iWRAP stays in command mode, where all A2DP messaging is done.

## 3.8  AVRCP mode

As of IWRAP3, Bluetooth Audio/Video Remote Control Profile (AVRCP) is supported. This profile also includes control messaging and events, which are handled in command mode.  In other words, when an AVRCP connection is opened or received no state transition occurs, but iWRAP stays in command mode, where all AVRCP messaging is done.

## 3.9  PBAP mode

As of IWRAP4, Bluetooth Phone Book Access Profile (PBAP) is supported. This profile also includes control messaging and events, which are handled in command mode.  In other words, when a PBAP connection is opened or received no state transition occurs, but iWRAP stays in command mode, where all PBAP messaging is done.

# 4 Technical details

| Feature | Value |
|---|---|
| MAX simultaneous ACL connections | 7 |
| MAX simultaneous SCO connections | 1 (2 with WT32) |
| MAX data rate | 550 kbps (WTxx to BT2.0 USB dongle)<br>500 kbps (WTxx to WTxx)<br>450 kbps (WTxx to BT1.1-BT1.2 device)<br>N/A (MUX data rate)<br>50 kbps (OBEX transfer) |
| MAX UART baud rate | 1800000 bps |
| Typical data transmission delay | 10-15ms |
| Minimum data transmission delay | 5-10ms |
| Typical SCO delay | 30-40ms |
| Typical A2DP delay (* | 150-200ms |
| A2DP coding/encoding methods | SBC, APT-x**, FastStream**, MP3** and AAC** |
| PIN code length | Configurable from 0 to 16 characters. |
| Encryption length | From 0 to 128** bits |
| MAX simultaneous pairings | 16 |
| MAX Friendly name length | Configurable up to 248 characters |
| RFCOMM Packet size | Configurable from 21 to 1008 |
| Supported Bluetooth profiles (iWRAP4) | GAP, SPP, HFP (v.1.5), HSP (v.1.2) A2DP, AVRCP, HID, DUN, DI, OPP, FTP, HDP*** and PBAP. |
| Supported power saving modes | Sniff and deep sleep |
| Bluetooth QD ID | iWRAP 4.0:  B016540<br>iWRAP 3.0:  B014328<br>iWRAP 2.2.0: B012647 |
| Secure Simple Pairing modes | Just works mode |

| | Man-in-the-middle protection (MITM) |
| --- | --- |
| | Out-of-Band (OOB) pairing |
| Echo canceling and noise reduction | Clear Voice Capture (cVc) algorithm. A licensable 3$^{rd}$ party product. |

**Table 2: Technical details**

*) Alternative coding methods (APT-x, FastStream) exist to reduce the delay to 40-90ms or to improve audio quality.

**) Custom firmware needs to be request from *support@bluegiga.com*

***) Health Device Profile is not included in the standard iWRAP4 firmware release, but is available separately.

Bluegiga Technologies Oy

# 5 iWRAP command reference

iWRAP can be used and controlled from the host system by sending ASCII commands through the UART interface to iWRAP.

This section explains the iWRAP commands and their syntax. Some simple usage examples and tips are also given.

**NOTES:**

- The parser is not case sensitive!

- iWRAP commands must end with a line feed **"\n"** character.

- By default iWRAP does not print OK to indicate that the command has been executed, but this feature can be separately enabled with **SET CONTROL CONFIG** command.

## 5.1  Command listings

All the available iWRAP commands are listed and briefly described in the tables below. The detailed description of each command can be found later.

| Command: | iWRAP version: | HW version: | Short description |
|---|---|---|---|
| AUTH | iWRAP 2.2.0 | ALL | Authenticates Bluetooth pairing |
| BER | iWRAP 2.2.0 | ALL | Reads Bit Error Rate |
| CALL | iWRAP 2.1.0 | ALL | Opens Bluetooth connections |
| CLOCK | iWRAP 3.0 | ALL | Reads Piconet clock |
| CLOSE | iWRAP 2.1.0 | ALL | Closes Bluetooth connections |
| CONNAUTH | iWRAP 4.0.0. | ALL | Authenticate incoming connections |
| CONNECT | iWRAP 3.0 | ALL | Connects Bluetooth links |
| ECHO | iWRAP 2.2.0 | ALL | Echoes data to Bluetooth connection |
| IC | iWRAP 2.2.0 | ALL | Inquiry cancel |
| IDENT | iWRAP 3.0 | ALL | Identifies a Bluetooth device |
| INQUIRY | iWRAP 2.1.0 | ALL | Searches other Bluetooth devices |
| KILL | iWRAP 3.0 | ALL | Kills Bluetooth connections |
| L2CAP | iWRAP 3.0 | ALL | Sets up L2CAP psm |
| LIST | iWRAP 2.1.0 | ALL | Lists Bluetooth connections |
| NAME | iWRAP 2.2.0 | ALL | Does friendly name discovery |
| PAIR | iWRAP 3.0 | ALL | Pairs with a Bluetooth device |
| PING | iWRAP 2.2.0 | ALL | Pings a Bluetooth connection |
| RFCOMM | iWRAP 3.0 | ALL | Sets up RFCOMM channels |
| RSSI | iWRAP 2.2.0 | ALL | Reads RSSI of a connection |

| SCO ENABLE | iWRAP 2.2.0 | ALL | Enables SCO connections |
|---|---|---|---|
| SCO OPEN | iWRAP 2.2.0 | ALL | Opens SCO connection |
| SDP | iWRAP 2.2.0 | ALL | Browse SDP records |
| SDP ADD | iWRAP 2.2.0 | ALL | Create SDP entries |
| SELECT | iWRAP 2.1.0 | ALL | Selects a Bluetooth connection |
| TEST | iWRAP 2.2.0 | ALL | Enables self test modes |
| TESTMODE | iWRAP 2.2.0 | ALL | Enables Bluetooth test mode |
| TXPOWER | iWRAP 2.2.0 | ALL | Reads TX power level |

**Table 3: Commands related to Bluetooth actions**

| Command: | iWRAP version: | HW version: | Short description |
|---|---|---|---|
| @ | iWRAP 4.0.0. | ALL | Shortcut for "SET {link_id} SELECT" |
| AIO | iWRAP 4.0.0 | ALL | Read AIO values |
| A2DP | iWRAP3.0 | WT32 | A2DP streaming control |
| AT | iWRAP 2.1.0 | ALL | Attention |
| BATTERY | iWRAP 3.0 | WT32 | Reads battery level |
| BCSP_ENABLE | iWRAP 3.0 | ALL | Enables BCSP mode |
| BOOT | iWRAP 2.2.0 | ALL | Boots module into different modes |
| BYPASSUART | iWRAP 3.0 | ALL | Enables UART bypass |
| DEFRAG | iWRAP 3.0 | ALL | Defrags PS key storage |
| HELP | iWRAP 2.2.0 | ALL | Prints help |
| INFO | iWRAP 2.2.0 | ALL | Prints firmware information |
| PIO | iWRAP 3.0 | ALL | Reads & Writes PIO statuses |
| RESET | iWRAP 2.1.0 | ALL | Does a software reset |
| SET | iWRAP 2.1.0 | ALL | Lists iWRAP configuration |
| SET RESET | iWRAP 3.0.0 | ALL | Restores factory settings |
| SLEEP | iWRAP 2.2.0 | ALL | Enables deep sleep |
| TEMP | iWRAP 3.0 | ALL | Reads internal temperature sensor |
| VOLUME | iWRAP 3.0 | ALL | Changes volume level |

**Table 4: Generic commands**

| Command: | iWRAP version: | HW version: | Short description |
|---|---|---|---|
| SET BT OPP | iWRAP 2.2.0 | ALL | Enable OPP profile |
| SET BT AUTH | iWRAP 2.1.0 | ALL | Set PIN code |
| SET BT BDADDR | iWRAP 2.1.0 | ALL | Not a command |
| SET BT CLASS | iWRAP 2.1.0 | ALL | Set Class-of-Device |
| SET BT IDENT | iWRAP 3.0 | ALL | Set DI profile data |
| SET BT LAP | iWRAP 2.2.0 | ALL | Set inquiry access code |
| SET BT MTU | iWRAP 4.0.0 | ALL | Configure Bluetooth connection MTU |
| SET BT NAME | iWRAP 2.1.0 | ALL | Change friendly name |
| SET BT PAGEMODE | iWRAP 2.1.0 | ALL | Set page mode and timeout |
| SET BT PAIR | iWRAP 2.1.0 | ALL | Manage pairings |
| SET BT PAIRCOUNT | iWRAP 4.0.0 | ALL | Limit the number of stored pairings |
| SET BT POWER | iWRAP 2.2.0 | ALL | Set TX power levels |
| SET BT ROLE | iWRAP 2.1.0 | ALL | Set role and supervision timeout |
| SET BT SNIFF | iWRAP 2.2.0 | ALL | Manage automatic sniff mode |

**Table 5: Bluetooth settings related SET commands**

| Command: | iWRAP version: | HW version: | Short description |
|---|---|---|---|
| SET CONTROL AUTOCALL | iWRAP 2.1.0 | ALL | Manage automatic connection control |
| SET CONTROL BATTERY | iWRAP 4.0.0. | WT32 | Change battery configuration |
| SET CONTROL BAUD | iWRAP 2.1.0 | ALL | Change UART baud rate |
| SET CONTROL BIND | iWRAP 2.2.0 | ALL | Manage GPIO bindings |
| SET CONTROL CD | iWRAP 2.1.0 | ALL | Manage Carrier Detect (CD) signal |
| SET CONTROL CODEC | iWRAP 4.0.0 | WT32 | Configures the internal audio codec |
| SET CONTROL CONFIG | iWRAP 2.1.0 | ALL | Manage configuration bits |
| SET CONTROL ECHO | iWRAP 2.1.0 | ALL | Manage echo mode |
| SET CONTROL GAIN | iWRAP 3.0 | WT32 | Manage ADC and DAC gains |
| SET CONTROL INIT | iWRAP 2.1.0 | ALL | Manage start-up command |
| SET CONTROL MICBIAS | iWRAP 3,0 | WT32 | Control MIC bias settings |
| SET CONTROL MSC | iWRAP 2.2.0 | ALL | Manage MSC functionality |
| SET CONTROL MUX | iWRAP 2.2.0 | ALL | Manage MUX mode |
| SET CONTROL PCM | iWRAP 3.0 | ALL | Manage PCM settings |
| SET CONTROL PREAMP | iWRAP 4.0 | WT32 | Enable/disable 20dB preamplifier |
| SET CONTROL RINGTONE | iWRAP 4.0 | All | Set HFP/HSP ringtone |
| SET CONTROL READY | iWRAP 4.0 | All | Tells when iWRAP firmware is ready |
| SET CONTROL VREGEN | iWRAP 3.0 | WT32 | Manage VREG_EN functionality |

**Table 6: Module configuration related SET commands**

| Command: | iWRAP version: | HW version: | Short description |
|---|---|---|---|
| SET {link_id} ACTIVE | iWRAP 2.1.0 | ALL | Disable Bluetooth link power saving |
| SET {link_id} MASTER | iWRAP 2.1.0 | ALL | Set Bluetooth link to master |
| SET {link_id} MSC | iWRAP 2.2.0 | ALL | Set Bluetooth link MSC status |
| SET {link_id} PARK | only iWRAP 2.2.0 | ALL | Enable Park state on a Bluetooth link |
| SET {link_id} SELECT | iWRAP 3.0 | ALL | Set Bluetooth link to active status |
| SET {link_id} SLAVE | iWRAP 2.1.0 | ALL | Set Bluetooth link to slave |
| SET {link_id} SNIFF | iWRAP 2.1.0 | ALL | Enable Sniff mode on a Bluetooth link |

**Table 7: Bluetooth connection related SET commands**

| Command: | iWRAP version: | HW version: | Short description |
|---|---|---|---|
| SET PROFILE A2DP | iWRAP 3.0.0 | WT32 | Enable / disable A2DP profile |
| SET PROFILE BGIO | iWRAP 4.0.0. | ALL | Enable / disable BGIO profile |
| SET PROFILE HDP* | iWRAP 4.0.0 | ALL | Enable / disable HDP profile |
| SET PROFILE HFP | iWRAP 2.1.0 | ALL | Enable / disable HFP profile |
| SET PROFILE HFP-AG | iWRAP 2.1.0 | ALL | Enable / disable HFP profile (AG) |
| SET PROFILE HID | iWRAP 3.0 | ALL | Enable / disable HID profile |
| SET PROFILE HSP | iWRAP 4.0.0 | ALL | Enable / disable HSP profile |
| SET PROFILE OPP | iWRAP 3.0.0 | ALL | Enable / disable OPP profile |
| SET PROFILE OTA | iWRAP 3.0.0 | ALL | Enable / disable OTA profile |
| SET PROFILE PBAP | iWRAP 4.0.0 | ALL | Enable / disable PBAP profile |
| SET PROFILE SPP | iWRAP 2.1.0 | ALL | Enable / disable SPP profile |

**Table 8: Supported Bluetooth profile commands**

*) HDP capable firmware only

## 5.2  Typographical conventions

The ASCII commands and their usage are further described in this chapter.

Commands and their output synopsis are presented as follows:

| Synopsis |
|---|
| **COMMAND** *{required parameter}* **[***optional parameter***]** **STATIC TEXT [***2ND OPTIONAL PARAMETER***]** |

Command parameters, on the other hand, are described like this:

| Description | |
|---|---|
| ***parameter*** | Description |

Responses to the command are described as shown in the table below:

| Response | |
|---|---|
| **RESPONSE {***parameters***}** | |
| ***parameter*** | Description |

Events generated by commands or actions are described as follows:

| Event | |
|---|---|
| <u>**EVENT**</u> | Description |

The list format shows how the current command configuration appears after the SET command is issued:

| List format |
|---|
| **COMMAND {***required parameter***} [***optional parameter***]** |

Finally, examples shown are described like this:

| |
|---|
| **iWRAP COMMAND** |
| iWRAP COMMAND RESPONSE(S) |

## 5.3 @

Command **@** can be used to read send commands to a dedicated profile parser like Hands-Free Profile's AT-command parser.

### 5.3.1 Syntax

| Synopsis: |
| --- |
| **@ {*link_id*} {*command*}** |

| Description: | |
| --- | --- |
| *link_id* | Numeric connection identifier |
| *{command}* | Command to send to the parser |

| Response: |
| --- |
| None. |

### 5.3.2 Examples

**CALL a8:7b:39:c3:ca:99 111F HFP          (HFP connection establishment)**

CALL 0

CONNECT 0 HFP 3

HFP 0 BSRF 491

HFP 0 STATUS "battchg" 5

HFP 0 STATUS "signal" 5

HFP 0 STATUS "service" 1

HFP 0 STATUS "call" 0

HFP 0 STATUS "callsetup" 0

HFP 0 STATUS "callheld" 0

HFP 0 STATUS "roam" 0

HFP 0 READY

RING 1 a8:7b:39:c3:ca:99 SCO

Bluegiga Technologies Oy

HFP 0 VOLUME 5

HFP 0 VOLUME 5

HFP 0 VOLUME 5

HFP 0 VOLUME 5

HFP 0 VOLUME 5

HFP 0 NETWORK "elisa"

NO CARRIER 1 ERROR 113 HCI_ERROR_OETC_USER

**@0 ATD777;**                                    **("ATD777;" sent to link ID 0)**

HFP 0 OK

HFP 0 STATUS "callsetup" 2

RING 1 a8:7b:39:c3:ca:99 SCO

HFP 0 VOLUME 6

HFP 0 VOLUME 5

HFP 0 STATUS "callsetup" 3

The above example shows how @ command can be used to send an AT command to the HFP profile parser. **@** command replaces "**SET** *{link_id}* **SELECT"** command and simplifies the software implementation in multi-profile use cases.

## 5.4 AIO

Command **AIO** can be used to read the value of ADC converters.

### 5.4.1 Syntax

| Synopsis: |
|---|
| **AIO {*source*}** |

| Description: | |
|---|---|
| ***source*** | Source AIO to read.<br>Valid values:  0 = AIO0 on WT32<br>                   1 = AIO1 on WT32<br>                   2 = Internal voltage reference on WT32 |

| Response: | |
|---|---|
| **AIO {*source*} {*value*}** | |
| ***source*** | Source AIO to read |
| ***value*** | Value of the AIO |

### 5.4.2 Examples

| |
|---|
| **AIO 0**<br>AIO 0 0015 |

## 5.5  AT

Command **AT**, "attention", can be used to check that iWRAP is functional and in command mode.

### 5.5.1  Syntax

| Synopsis |
|---|
| **AT** |

| Response |
|---|
| **OK** |

### 5.5.2  Examples

| |
|---|
| **AT**<br>OK |

**Tip:**

- In iWRAP3 or older version iWRAP commands do not produce replies telling that command was successful or execution has finished. AT command can be used to provide this functionality, but appending AT into the end of other iWRAP commands.

**Appending AT after "SET BT AUTH" command:**

| |
|---|
| **SET BT AUTH * 4564\r\nAT\r\n**<br>OK |

## 5.6  AUTH

**AUTH** command can be used to reply to **AUTH** event to perform interactive pairing. **AUTH** event is only displayed if **SET CONTROL CONFIG** bit 11 is set.

### 5.6.1  Syntax

| Synopsis: |
| --- |
| **AUTH {***bd_addr***} [***pin_code]* |

| Description: | |
| --- | --- |
| ***bd_addr*** | Bluetooth device address of the remote device |
| ***pin_code*** | Bluetooth pin code |

| Response: |
| --- |
| No response |

| Events: | |
| --- | --- |
| **PAIR           {bd_addr} {link_key}** | This event occurs if **PAIR** event is enabled with **SET CONTROL CONFIG** and pairing is successful. |

### 5.6.2  Examples

Interactive pairing with AUTH command, initiated from remote device.

AUTH 00:07:80:81:66:8c?

**AUTH 00:07:80:81:66:8c 6666**

Declining pairing with AUTH command.

AUTH 00:07:80:81:66:8c?

**AUTH 00:07:80:81:66:8c**

Pairing with AUTH command and with **PAIR** event enabled.

AUTH 00:07:80:81:66:8c?

**AUTH 00:07:80:81:66:8c 6666**

PAIR 00:07:80:81:66:8c 0 16b9515e878c39ed785ba4499322079e

Bluegiga Technologies Oy

## 5.7 AVRCP PDU

**AVRCP PDU** command is used by the AVRCP Controller to send metadata request Protocol Data Units to the Target.

### 5.7.1 Syntax

| Synopsis |
|---|
| **AVRCP PDU {***PDU_ID***} [***parameters***]** |

| Description | |
|---|---|
| *10* | Get capabilities command. Query for events or Company_ID's the Target supports.<br><br>Parameters:<br><br>**2**<br><br>       Query supported Company_ID's.<br><br>**3**<br><br>       Query supported events. |
| *11* | List player application settings. No parameters. |
| *12* | List possible values for a player application setting.<br><br>Parameters:<br><br>**{setting_id}**<br><br>       See list at the end of this command's description. |
| *13* | Get current values of player application settings.<br><br>Parameters:<br><br>**{number of settings}**<br><br>       Number of following parameters.<br><br>Followed by:<br><br>**{setting_id}**<br><br>       See list at the end of this command's description. |
| *14* | Set current values of player application settings.<br><br>Parameters:<br><br>**{number of settings}**<br><br>       Number of setting_id-value-pairs that follow. |

Bluegiga Technologies Oy

| | |
|---|---|
| | Followed by: <br><br> **{setting_id} {value}** <br><br> See list at the end of this command's description. |
| *20* | Get attributes of the currently playing track. <br><br> Parameters: <br><br> **{number of attributes}** <br><br> Number of attributes that follow. If zero, list all available information. <br><br> Followed by (unless number of attributes is zero): <br><br> **[attribute_id]** <br><br> See list at the end of this command's description. |
| *30* | Get the playing status, length and position of the current track. No parameters. |
| *31* | Register notification of events. This will request the Target to notify us when a track is changed for instance. <br><br> Parameters: <br><br> **{event_id}** <br><br> See list at the end of this command's description. |

| Events |
|---|
| **AVRCP {*PDU_ID name*}_RSP [*parsed data*]** <br><br> **AVRCP RSP PDU_ID {*PDU_ID*}, data: [*unparsed data*]** <br><br> **AVRCP {*PDU_ID name*}_RSP REJ** |

## 5.7.2 Examples

Ask the Target which events it supports.

---

**AVRCP PDU 10 3**

AVRCP GET_CAPABILITIES_RSP EVENT COUNT 3 PLAYBACK_STATUS_CHANGED TRACK_CHANG
ED PLAYBACK_POSITION_CHANGED

---

Ask the Target about its player application settings, their possible values and change a value.

---

**AVRCP PDU 11**

AVRCP LIST_APPLICATION_SETTING_ATTRIBUTES_RSP COUNT 2 REPEAT SHUFFLE

**AVRCP PDU 12 2**

AVRCP LIST_APPLICATION_SETTING_VALUES_RSP COUNT 3 1 2 3

**AVRCP PDU 13 1 2**

AVRCP GET_APPLICATION_SETTING_VALUE_RSP COUNT 1 REPEAT OFF

**AVRCP PDU 14 1 2 2**

AVRCP SET_APPLICATION_SETTING_VALUE_RSP

**AVRCP PDU 13 1 2**

AVRCP GET_APPLICATION_SETTING_VALUE_RSP COUNT 1 REPEAT SINGLE_TRACK

---

Ask the Target about the title and artist of the song that is currently playing and ask it to notify us if the playback status changes.

---

**AVRCP PDU 20 2 1 2**

**AVRCP GET_ELEMENT_ATTRIBUTES_RSP COUNT 2 TITLE "Cold Women and Warm Beer" ARTIST "The Black League"**

**AVRCP PDU 31 1 1**

AVRCP REGISTER_NOTIFICATION_RSP INTERIM PLAYBACK_STATUS_CHANGED PLAYING

(the interim response is received right after the request to confirm we were registered for notification)

AVRCP REGISTER_NOTIFICATION_RSP CHANGED PLAYBACK_STATUS_CHANGED PAUSED

(the changed response is received when the playing status changes)

---

## 5.8  BATTERY

Command **BATTERY** is used to read the current voltage of the module battery. Works only with WT32.

### 5.8.1  Syntax

| Synopsis: |
| --- |
| **BATTERY** |

| Description: |
| --- |
| None |

| Response: |
| --- |
| None |

| Events: |
| --- |
| **BATTERY {mv}** | Current battery voltage in millivolts. |

### 5.8.2  Examples

Reading battery voltage.

| **BATTERY** |
| --- |
| BATTERY 3673 |

## 5.9  BCSP_ENABLE

Command **BCSP_ENABLE** is used to boot the device and enter BCSP mode; it is an alias for **BOOT 1**.
See the documentation of **BOOT** command for a detailed explanation of iWRAP boot modes.

### 5.9.1  Syntax

| Synopsis: |
|---|
| **BCSP_ENABLE** |

| Description: |
|---|
| None |

| Response: |
|---|
| No response |

| Events: |
|---|
| None |

### 5.9.2  Examples

Switching iWRAP into BCSP mode. BCSP link establishment packets are sent after command has been executed.

| |
|---|
| **BCSP_ENABLE** |
| À |
|     ?¯WWUo`À |
|          ?¯WWUo`À |
|             ?¯WWUo`À |
|               ?¯WWUo`À |

## 5.10 BER

The **BER** command returns the Bit Error Rate of the given link ID.

### 5.10.1 Syntax

| Synopsis: |
|---|
| **BER {*link_id*}** |

| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier |

| Response: | |
|---|---|
| **BER {*bd_addr*} {*ber*}** | |
| ***bd_addr*** | Bluetooth address of the remote device |
| ***ber*** | Average Bit Error Rate on the link. Possible values are from 0.0000 to 100.0000. |

| Events: |
|---|
| None |

### 5.10.2 Examples

Checking the Bit Error Rate of an active connection.

| |
|---|
| **LIST**<br>LIST 1<br>LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE MASTER PLAIN<br><br>**BER 0**<br>BER 00:60:57:a6:56:49 0.0103          (Bit Error Rate is 0.0103 per cent) |

**Note:**

- Works only for BDR links.

## 5.11 BOOT

The **BOOT** command is used to change the iWRAP's boot mode. After issuing this command, the module will enter the selected boot mode. After resetting the module, it will boot in iWRAP mode again.

The boot mode change can be made permanent by writing the boot mode to PS-key: "Initial device bootmode".

### 5.11.1    Syntax

| Synopsis: |
| --- |
| **BOOT {*boot_mode*}** |

| Description: | |
| --- | --- |
| *boot_mode* | **0000** <br><br> iWRAP <br><br> **0001** <br><br> HCI, BCSP, 115800,8e1 <br><br> **0003** <br><br> HCI, USB <br><br> **0004** <br><br> HCI, H4, 115200,8n1 |

| Response: |
| --- |
| No response |

### 5.11.2    Examples

Boot to BCSP mode. Same as issuing **BCSP_ENABLE** command.

| |
| --- |
| **BOOT 1** <br> • Ò• ¯WWUo`À <br>   • Ò• ¯WWUo`À <br>     • Ò• ¯WWUo`À <br>       • Ò• ¯WWUo`À <br>         • Ò• ¯WWUo |

## 5.12 BYPASSUART

**BYPASSUART** command enables the UART bypass mode, in which the UART traffic is relayed to GPIO pins instead of iWRAP. Please refer to the modules data sheet for more information. A physical reset is needed to return to normal operation mode.

### 5.12.1 Syntax

| Synopsis: |
|---|
| **BYPASSUART** |


| Response: |
|---|
| No response |


| Events: |
|---|
| No event is raised |

## 5.13 CALL

The **CALL** command is used to initiate Bluetooth connections to the remote devices. Connections are closed by using command **CLOSE**. Currently open connections can be viewed by using command **LIST**.

### 5.13.1 Syntax

| Synopsis |
| --- |
| CALL {*address*} {*target*} {*connect_mode*} [MTU {*packet size*}] |

| Description | |
| --- | --- |
| *address* | Bluetooth address of the remote device |
| *target* | RFCOMM, HFP or HFP-AG, HID or A2DP target for the connection. The target can be one of the following: **channel**<br><br>    RFCOMM channel number<br>    HFP channel number<br>    HFP-AG channel number<br>    Format: xx (hex)<br><br>**uuid16**<br><br>    16-bit UUID for searching channel<br>    Format: xxxx (hex)<br><br>**uuid32**<br><br>    32-bit UUID for searching channel<br>    Format: xxxxxxxx (hex)<br><br>**uuid128**<br><br>    128-bit UUID for searching channel<br>    Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (hex)<br><br>**L2CAP psm**<br><br>    16-bit L2CAP psm. Must be an odd value.<br>    Format: xxxx (hex) |
| *connect_mode* | Defines the connection mode to be established.<br>Possible modes are: |

| | |
|---|---|
| | **RFCOMM** |
| |  Normal RFCOMM connection |
| | **HFP** |
| |  Opens a connection in the Hands Free device mode. |
| | **HFP-AG** |
| |  Opens a connection in the Hands Free Audio Gateway mode. |
| | **A2DP** |
| |  Opens a connection in the Advanced Audio Distribution Profile (A2DP) mode. L2CAP psm for A2DP is 19. |
| | **AVRCP** |
| |  Opens a connection in the Audio Video Remote Control Profile (AVRCP) mode. L2CAP psm for AVRCP is 17. |
| | **HID** |
| |  Opens a connection in the HID keyboard mode or HID mouse mode. L2CAP psm for HID is 11. |
| | **L2CAP** |
| |  Opens a generic L2CAP connection. |
| | **PBAP** |
| |  Opens a Phone Book Access Profile connection. |
| | **OPP** |
| |  Opens an OBEX Object Push Profile connection. |
| | **FTP** |
| |  Opens an OBEX File Transfer Profile connection. |
| | **HSP** |
| |  Opens a Bluetooth Headset Profile connection |
| | **HSP-AG** |
| |  Opens a Bluetooth Headset Profile Audio Gateway connection |
| | **HDP** |
| |  Opens a Bluetooth Health Device Profile connection |
| **MTU** | Optional static text to indicate that the packet size parameter is used. |
| *packet size* | Packet size to use (in bytes) Range: 21 to 1008 |

| Response | |
|---|---|
| **CALL {*link_id*}** | |
| ***link_id*** | Numeric connection identifier |

| Events | |
|---|---|
| **<u>CONNECT</u>** | Delivered if the **CALL** command is successful. |
| **<u>NO CARRIER</u>** | Delivered if the **CALL** command fails. |
| **<u>PAIR</u>** | If the **<u>PAIR</u>** event is enabled by using "**SET CONTROL CONFIG**", it will be displayed during the call if paring has to be done. |
| **<u>CLOCK</u>** | If piconet clock event is enabled, **<u>CLOCK</u>** event will be displayed. |
| **<u>AUTH</u>** | If interactive pairing mode is enabled and no paring exists, **<u>AUTH</u>** event will be displayed. |

## 5.13.2    Examples

Creating a successful connection to 00:07:80:80:52:27 using Serial Port Profile.

(UUID16 SPP = 1101)

---
**CALL 00:07:80:80:52:27 1101 RFCOMM**

CALL 0

CONNECT 0 RFCOMM 1

---

Creating a successful connection to 00:07:80:80:52:27 using RFCOMM channel 1.

---
**CALL 00:07:80:80:52:27 1 RFCOMM**

CALL 0

CONNECT 0 RFCOMM 1

---

Unsuccessful SPP connection attempt to 00:07:80:80:52:26.

---
**CALL 00:07:80:80:52:26 1101 RFCOMM**

CALL 0

NO CARRIER 0 ERROR 406 RFC_CONNECTION_FAILED

---

Creating a successful connection to 00:07:80:80:52:27 with MTU 600.

**CALL 00:07:80:80:52:27 1 RFCOMM MTU 600**

CALL 0

CONNECT 0 RFCOMM 1

Creating a successful A2DP connection

**CALL 00:07:80:80:52:27 19 A2DP**

CALL 0

CONNECT 0 A2DP 25

CONNECT 1 A2DP 25

Creating a successful AVRCP connection

**CALL 00:07:80:80:52:27 17 AVRCP**

CALL 0

CONNECT 0 AVRCP 23

Creating a successful HID connection

**CALL 00:07:80:80:52:27 11 HID**

CALL 0

CONNECT 0 HID 17

CONNECT 1 HID 19

Creating a successful PBAP connection

**CALL 00:07:80:80:52:27 112F PBAP**

CALL 0

CONNECT 0 PBAP 5

Creating a successful OBEX OPP connection

**CALL 00:07:80:80:52:27 1105 OPP**

CALL 0

CONNECT 0 OPP 2

Bluegiga Technologies Oy

Creating a successful Health Device Profile MCAP Communications Link (MCL)

---

**CALL 00:07:80:80:52:27 1001 HDP**

CALL 0

CONNECT 0 HDP 4097

---

Opening a HSP connection from iWRAP to Headset Audio Gateway (phone).

---

**CALL 00:07:80:80:52:27 1008 HSP**

CALL 0

CONNECT 0 HSP 5

---

Opening a HSP connection from iWRAP (HSP-AG) to Headset.

---

**CALL 00:07:80:80:52:26 1008 HSP-AG**

CALL 0

CONNECT 0 HSP-AG 5

---

**Note:**

- If **CALL** is used with **CHANNEL** instead of **UUID**, it will be on average around 300ms faster, since there is no need to do service discovery. However when calling directly with RFCOMM channel you need to be sure that the profile you want to connect to is always in that RFCOMM channel. RFCOMM channel assignments are manufacturer specific and vary between different Bluetooth devices.

## 5.14 CLOCK

**CLOCK** command can be used to read the Bluetooth Piconet clock value. This is useful if time synchronization between different Piconet devices needs to be achieved.

### 5.14.1    Syntax

| Synopsis: |
|---|
| **CLOCK {*link_id*}** |

| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier |

| Response: |
|---|
| No response |

| Events: | |
|---|---|
| **CLOCK        {bd_addr} {clock}** | CLOCK event occurs, if valid ***link_id*** is used. |
| **SYNTAX ERROR** | If incorrect parameters are given. |

### 5.14.2    Examples

Reading Piconet clock value:

| |
|---|
| **CLOCK 0**<br>CLOCK 00:07:80:12:34:56 3bb630 |

**Note:**

- Piconet clock is extremely useful when time needs to be synchronized between Piconet slaves. All the slaves in the Piconet are synchronized to master's clock and they share the same clock value.

- Accuracy is 625us, but it also takes some time for the iWRAP to perform the **CLOCK** command and display the result. This time can not be unambiguously defined as it depends on the state of iWRAP.

## 5.15 CLOSE

Command **CLOSE** is used to terminate a Bluetooth connection.

### 5.15.1    Syntax

| Synopsis: |
|---|
| **CLOSE {*link_id*}** |

| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier from a previously used command **CALL** or from event **RING**. |

| Response: |
|---|
| No response |

| Events: | |
|---|---|
| **NO CARRIER** | This event is delivered after the link has been closed. |

### 5.15.2    Examples

Closing an active connection:

| |
|---|
| **CALL 00:60:57:a6:56:49 1103 RFCOMM**<br>CALL 0<br>CONNECT 0 RFCOMM 1<br>*[+++]*                               (data mode -> command mode transition)<br>READY.<br><br>**CLOSE 0**<br>NO CARRIER 0 ERROR 0 |

2.1.2012

## 5.16 CONNAUTH

**CONNAUTH** command can be used to authenticate incoming Bluetooth connections. It is used to reply to **CONNAUTH** event.

### 5.16.1 Syntax

| Synopsis: |
|---|
| **CONNAUTH {*bd_addr*} {*protocol_id* } {*channel_id*} [OK]** |

| Description: | |
|---|---|
| *bd_addr* | Bluetooth device address of the remote device trying to connect. |
| *protocol_id* | Protocol ID of the incoming connection<br>**0**<br>**1**<br>**2** |
| *channel_id* | Channel number of the incoming connection |
| *OK* | Optional flag, which decides if the connection is accepted or not. If the flag is used the connection is accepted and if it is not used the connection is declined. |

| Response: |
|---|
| None |

| Events: |
|---|
| None |

**Note:**

- For **CONNAUTH** feature to work properly the Secure Simple Pairing mode MUST be enabled in iWRAP. If SSP is not enabled it may not be able to open a  Bluetooth connection.

## 5.17 CONNECT

iWRAP3 and newer can act as a repeater / range extender for RFCOMM connections by using the **CONNECT** command which will transparently link two ongoing connections together as a connection between the two remote devices.

### 5.17.1 Syntax

| Synopsis: |
| --- |
| **CONNECT {*link_id1*} {*link_id2*}** |

| Description: | |
| --- | --- |
| ***link_id_1*** | Numeric connection identifier as displayed by the LIST command. |
| ***link_id_2*** | Numeric connection identifier as displayed by the LIST command.. |

| Response: |
| --- |
| None |

| Events: |
| --- |
| None |

## 5.17.2 Examples

Piping two RFCOMM connections.

---

**SET BT PAGEMODE 3**

RING 0 00:07:80:87:69:2f 1 RFCOMM

RING 1 00:07:80:87:68:ec 1 RFCOMM

**+++**                                        (Data to command mode transition)

READY.

**LIST**                                       (List active connections)

LIST 2

LIST 0 CONNECTED RFCOMM 320 0 0 33 8d 1 00:07:80:87:69:2f 1 INCOMING ACTIVE SLAVE PLAIN 0

LIST 1 CONNECTED RFCOMM 320 0 0 31 8d 8d 00:07:80:87:68:ec 1 INCOMING ACTIVE MASTER PLAIN 0

**CONNECT 0 1**

---

First the page mode is set to 3 so that iWRAP is able to receive 2 connections. Second **LIST** command is issued to show that two connections exist. Finally the connections are piped with **CONNECT** command. After this has been done iWRAP transparently sends all data from 1st connection to the 2nd one and vice versa.

## 5.18 ECHO

The **ECHO** command sends a specified string of characters to the active link specified by the '*link_id*' parameter. This command can be used, for example, with command **SET CONTROL BIND** to send an indication of activity over a Bluetooth link.

### 5.18.1   Syntax

| Synopsis: |
|---|
| **ECHO {*link_id*} [*string*]** |

| Description: | |
|---|---|
| *link_id* | Numeric connection identifier |
| *string* | User-determined string of characters |

| Response: |
|---|
| No response |

| Events: |
|---|
| None |

### 5.18.2   Examples

| |
|---|
| **ECHO 0 DATA**          **(Sends "DATA" to link with ID 0)** |

## 5.19 DEFRAG

This command defragments persistent store memory. The command will reset iWRAP.

### 5.19.1 Syntax

| Synopsis: |
|---|
| **DEFRAG** |

| Description: |
|---|
| None |

| Response: |
|---|
| No response |

| Events: |
|---|
| None |

## 5.20 INQUIRY

Command **INQUIRY** is used to find other Bluetooth devices in the area i.e. to make a device discovery.

### 5.20.1    Syntax

| Synopsis: |
|---|
| **INQUIRY {*timeout*} [NAME] [LAP {*lap*}]** |

| Description: | |
|---|---|
| *timeout* | The maximum amount of time (in units of 1.28 seconds) before the inquiry process is halted.<br><br>Range: 1-48 |
| **NAME** | Optional flag to automatically request the friendly name for found devices. See command **NAME** for more information about the remote name request. |
| **LAP** | Optional flag for specifying that inquiry access code will be used. |
| *lap* | Value for inquiry access code. The following values are possible:<br><br>**9E8B33**<br><br>General/Unlimited Inquiry Access Code (GIAC). This is the default value unless "**SET BT LAP**" is used.<br><br>**9E8B00**<br><br>Limited Dedicated Inquiry Access Code (LIAC).<br><br>**9E8B01-9E8B32** and **9E8B34-9E8B3F**<br><br>Reserved for future use. |

| Response: | |
|---|---|
| **INQUIRY {*num_of_devices*}** <br> and <br> **INQUIRY {*addr*} {*class_of_device*}** | |
| ***num_of_devices*** | The number of found devices |
| ***addr*** | Bluetooth device address |
| ***class_of_device*** | Bluetooth Class of Device |

| Events: | |
|---|---|
| **INQUIRY_PARTIAL** | These events are delivered as devices are found. |
| **INQUIRY_EXTENDED** | These events are delivered when Bluetooth 2.1 + EDR devices are found that support Extended Inquiry Response (EIR) |
| **NAME** | These events are delivered after **INQUIRY** if the ***NAME*** flag is present. |
| **NAME_ERROR** | These events are delivered after **INQUIRY** if the ***NAME*** flag is present and the name discover fails. |

## 5.20.2    Examples

Basic INQUIRY command example

```
INQUIRY 1
INQUIRY_PARTIAL 00:14:a4:8b:76:9e 72010c
INQUIRY_PARTIAL 00:10:c6:62:bb:9b 1e010c
INQUIRY 2
INQUIRY 00:14:a4:8b:76:9e 72010c
INQUIRY 00:10:c6:62:bb:9b 1e010c
```

An INQUIRY with NAME resolution

**INQUIRY 1 NAME**

INQUIRY_PARTIAL 00:14:a4:8b:76:9e 72010c

INQUIRY 1

INQUIRY 00:14:a4:8b:76:9e 72010c

NAME 00:14:a4:8b:76:9e "SWLTMIKKO_3"

An INQUIRY command with LIAC in use

**INQUIRY 1 LAP 9E8B00**

INQUIRY_PARTIAL 00:07:80:80:52:15 111111

INQUIRY_PARTIAL 00:07:80:80:52:27 111111

INQUIRY 2

INQUIRY 00:07:80:80:52:15 111111

INQUIRY 00:07:80:80:52:27 111111

An INQUIRY command with RSSI enabled

**INQUIRY 1**

INQUIRY_PARTIAL 00:14:a4:8b:76:9e 72010c "" -71

INQUIRY_PARTIAL 00:10:c6:62:bb:9b 1e010c "" -73

INQUIRY 2

INQUIRY 00:14:a4:8b:76:9e 72010c

INQUIRY 00:10:c6:62:bb:9b 1e010c

An INQUIRY command with EIR responses

**INQUIRY 2**

INQUIRY_PARTIAL 00:18:42:f1:a5:be 5a020c "" -92

INQUIRY_PARTIAL 00:17:e4:ef:f9:01 50020c "" -92

INQUIRY_EXTENDED 00:07:80:87:68:ec RAW 0909575433322d53616d020a0800

INQUIRY_PARTIAL 00:07:80:87:68:ec 200428 "WT32-Sam" -73

INQUIRY 3

INQUIRY 00:18:42:f1:a5:be 5a020c

INQUIRY 00:17:e4:ef:f9:01 50020c

INQUIRY 00:07:80:87:68:ec 200428

## 5.21 IC

The **IC** (inquiry cancel) command can be used to stop an on-going inquiry.

### 5.21.1    Syntax

| Synopsis: |
|---|
| **IC** |

| Description: |
|---|
| No Description |

| Response: | |
|---|---|
| **INQUIRY {*num_of_devices*}** | |
| **INQUIRY {*addr*} {*class_of_device*}** | |
| ***num_of_devices*** | The number of found devices |
| ***addr*** | Bluetooth address of a found device |
| ***class_of_device*** | Bluetooth Class of Device of a found device |

| Events: |
|---|
| None |

### 5.21.2    Examples

**INQUIRY 5**
INQUIRY_PARTIAL 00:14:a4:8b:76:9e 72010c "" -71
INQUIRY_PARTIAL 00:10:c6:62:bb:9b 1e010c "" -73

**IC**
INQUIRY 2
INQUIRY 00:14:a4:8b:76:9e 72010c
INQUIRY 00:10:c6:62:bb:9b 1e010c

**Note:**

- IC command cancels the inquiry only if issued before the "**INQUIRY {num_of_devices}**" message. The name resolution process can not be cancelled with **IC.**

## 5.22 IDENT

**IDENT** command can be used to identify a remote Bluetooth device with the Bluetooth Device ID profile.

### 5.22.1    Syntax

| Synopsis: |
|---|
| **IDENT {*bd_addr*}** |

| Description: | |
|---|---|
| ***bd_addr*** | Bluetooth device address of the remote device |

| Response: |
|---|
| No response |

| Events: | |
|---|---|
| **IDENT** | **IDENT** event is raised if a successful response is received |
| **IDENT ERROR** | **IDENT ERROR** event is raised if identification fails |

### 5.22.2    Examples

Successful IDENT of a remote Bluetooth device.

| |
|---|
| **IDENT 00:07:80:00:a5:a5**<br>IDENT 00:07:80:00:a5:a5 BT:47 f000 3.0.0 "Bluegiga iWRAP"<br><br>**IDENT 00:07:80:82:42:d8**<br>IDENT 00:07:80:82:42:d8 BT:47 b00b 3.2.0 "Bluegiga Access Server" |

Using IDENT to try to identify a remote Bluetooth device without success.

| |
|---|
| **IDENT 00:07:80:00:48:84**<br><br>**IDENT ERROR 2 00:07:80:00:48:84 NOT_SUPPORTED_BY_REMOTE** |

## 5.23 INFO

**INFO** displays information about iWRAP version and features.

### 5.23.1    Syntax

| Synopsis: |
|---|
| **INFO [*CONFIG | BOOTMODE*]** |

| Description: | |
|---|---|
| ***CONFIG*** | Optional flag that displays more detailed information about the firmware for example changed parameters. |
| ***BOOTMODE*** | Displays bootmode parameters |

| Response: |
|---|
| Information about iWRAP version and features. |

| Events: |
|---|
| None. |

## 5.23.2 Examples

---

**INFO**

WRAP THOR AI (4.0.0 build 313)

Copyright (c) 2003-2010 Bluegiga Technologies Inc.

Compiled on Apr  8 2010 11:23:45, running on WT32-A module, psr v26

    A2DP AVRCP BATTERY FTP MAP MICBIAS PBAP PIO=0x07ff SSP SUBRATE VOLUME

    - BOCK3 version 313 (Apr  8 2010 11:23:38) (max acl/sco 7/1)

    - Bluetooth version 2.1, Power class 3

    - Loader 6302, firmware 6302 (56-bit encryption), native execution mode

    - up 0 days, 01:12, 0 connections (pool 2)

READY.

---

Detailed information display:

---

**INFO CONFIG**

WRAP THOR AI (4.0.0 build 313)

Copyright (c) 2003-2010 Bluegiga Technologies Inc.

Compiled on Apr  8 2010 11:23:45, running on WT32-A module, psr v26

    A2DP AVRCP BATTERY FTP MAP MICBIAS PBAP PIO=0x07ff SSP SUBRATE VOLUME

    - BOCK3 version 313 (Apr  8 2010 11:23:38) (max acl/sco 7/1)

    - Bluetooth version 2.1, Power class 3

    - Loader 6302, firmware 6302 (56-bit encryption), native execution mode

    - up 0 days, 00:00, 0 connections (pool 1)

    - User configuration:

&02ac = 0000 0000 002b 0000 0000 0000 0000 0000 0000 0000 0042 0000 0000 0000 0010 0000 0000 0000 0000 029b 0000 0000 0000 0000

&02ad = 5457 3233 412d

&02b1 = 0000 0000 0000

READY.

---

**Note:**

- When requesting a custom firmware configuration from Bluegiga, it useful to attach output of "**INFO CONFIG**" to the request.

## 5.24 KILL

Command **KILL** is used to explicitly terminate all ACL connections between two devices.

### 5.24.1 Syntax

| Synopsis: |
| --- |
| **KILL {*bd_addr*} [*reason*]** |

| Description: | |
| --- | --- |
| ***bd_addr*** | Bluetooth address of the connected remote device. |
| ***reason*** | Reason for disconnecting the ACL link; see Chapter 9 for a listing of possible error codes. The default value is 0x115: HCI_ERROR_OETC_POWERING_OFF, device is about to power off. |
| | All existing RFCOMM connections will disconnect with reason RFC_ABNORMAL_DISCONNECT. |

| Response: |
| --- |
| None |

| Events: | |
| --- | --- |
| **NO CARRIER** | This event is delivered after the link is closed. |

## 5.25 L2CAP

Command **L2CAP** is used to create a L2CAP psm for L2CAP connections to the local device.

### 5.25.1    Syntax

| Synopsis: |
|---|
| **L2CAP {*psm*}** |

| Description: | |
|---|---|
| ***psm*** | L2CAP psm; must be an odd number. |

| Response: |
|---|
| No response |

| Events: | |
|---|---|
| <u>**SYNTAX ERROR**</u> | If an invalid UUID is given. |

### 5.25.2    Examples

Making an L2CAP call between two iWRAPs:

| | |
|---|---|
| **L2CAP 37** | (Creates L2CAP psm 37 on the local device) |
| **CALL 00:07:80:12:34:56 37 L2CAP** | (Opening L2CAP connection to a remote device) |
| CALL 0 | |
| CONNECT 0 L2CAP 37 | |

## 5.26 LIST

Command **LIST** shows the count of active connection and detailed information about each connection.

### 5.26.1 Syntax

| Synopsis: |
|---|
| **LIST** |

| Description: |
|---|
| No description |

| Response: |
|---|
| **LIST {*num_of_connections*}**<br><br>**LIST {*link_id*} CONNECTED {*mode*} {*blocksize*} 0 0 {*elapsed_time*} {*local_msc*} {*remote_msc*} {*addr*} {*channel*} {*direction*} {*powermode*} {*role*} {*crypt*} {*buffer*} [ERETX]** |

| | |
|---|---|
| ***num_of_connections*** | Number of active connections. Possible values range from 0 to 7. |
| ***link_id*** | Numeric connection identifier |
| ***mode*** | **RFCOMM**<br><br>Connection type is RFCOMM<br><br>**L2CAP**<br><br>Connection type is L2CAP<br><br>**SCO**<br><br>Connection type is SCO |
| ***blocksize*** | RFCOMM, L2CAP or SCO data packet size, that is, how many bytes of data can be sent in one packet |
| ***elapse_time*** | Link life time in seconds |
| ***local_msc*** | Local serial port modem status control (MSC) bits. |
| ***remote_msc*** | Remote serial port modem status control (MSC) bits. |
| ***addr*** | Bluetooth device address of the remote device |

Bluegiga Technologies Oy

| | |
|---|---|
| *channel* | RFCOMM channel or L2CAP psm number at remote device |
| *direction* | Direction of the link. The possible values are:<br><br>**OUTGOING**<br><br>    The connection has been initiated by the local device.<br><br>**INCOMING**<br><br>    The connection has been initiated by the remote device |
| *powermode* | Power mode for the link. The possible values are:<br><br>**ACTIVE**<br><br>    Connection is in active mode, no power saving in use<br><br>**SNIFF**<br><br>    Connection is in sniff mode<br><br>**HOLD**<br><br>    Connection is in hold mode<br><br>**PARK**<br><br>Connection is in park mode |
| *role* | Role of the link. The possible values are:<br><br>**MASTER**<br><br>    iWRAP is the master device of this connection<br><br>**SLAVE**<br><br>iWRAP is the slave device of this connection |
| *crypt* | Encryption state of the connection. The possible values are:<br><br>**PLAIN**<br><br>    Connection is not encrypted<br><br>**ENCRYPTED**<br><br>Connection is encrypted |
| *buffer* | Tells the amount of data (in bytes) that is stored in the incoming data buffer. |
| *ERETX* | This flag is visible is enhanced retransmission mode is in use. At the moment only used with HDP connections. |

| **Events:** |
|---|
| No events raised |

## 5.26.2    Examples

Listing active connections:

**LIST**

LIST 1

LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE MASTER PLAIN 0

## 5.27 NAME

Command **NAME** can be used to perform a friendly name discovery.

### 5.27.1 Syntax

| Synopsis: |
|---|
| **NAME {*bd_addr*}** |

| Description: | |
|---|---|
| ***bd_addr*** | Bluetooth address of the connected remote device. |

| Response: | |
|---|---|
| **NAME {*bd_addr*} "{*name*}"** <br><br> or <br><br> **NAME ERROR {*error_code*} {*bd_addr*} {*reason*}** | |
| ***bd_addr*** | Bluetooth address of the connected remote device. |
| ***name*** | Friendly name of the remote device |
| ***error_code*** | Error code |
| ***reason*** | ASCII description of the reason |

| Events: |
|---|
| None. |

Bluegiga Technologies Oy

## 5.27.2　Examples

Making a successful name discovery

---

**NAME 00:07:80:FF:FF:F1**

NAME 00:07:80:FF:FF:F1 "WT32-A"

---

Name discovery error because of page timeout

---

**NAME 00:07:80:FF:FF:F2**

NAME ERROR 0x104 00:07:80:FF:FF:F2 HCI_ERROR_PAGE_TIMEOUT

---

## 5.28 PAIR

Command **PAIR** can be used to pair with other Bluetooth devices. Pairing mode can be traditional or Secure Simple Pairing.

### 5.28.1    Syntax

| Synopsis |
|---|
| **PAIR  {*bd_addr*}** |

| Description | |
|---|---|
| ***bd_addr*** | Bluetooth device address of the device remote device |

| Response | |
|---|---|
| **PAIR {*bd_addr*} {*result*}** | |
| ***bd_addr*** | Bluetooth device address of the device remote device |
| ***result*** | **OK**<br><br>    Pairing successful<br><br>**FAIL**<br><br>    Pairing failed |

| Events | |
|---|---|
| **PAIR      {bd_addr} {status}** | This event occurs if PAIR event is enabled with "**SET CONTROL CONFIG**" and pairing is successful. |
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |
| **AUTH** | This event occurs if interactive pairing is enabled with "**SET CONTROL CONFIG**". |

**Note:**

- In iWRAP4 if pin codes are not set PAIR will return "**PAIR  {*bd_addr*} FAIL**" since the link keys can not be generated.

- In iWRAP3 and older version and similar situation iWRAP returned "**PAIR  {*bd_addr*} OK"**.

Bluegiga Technologies Oy

**Bluetooth 2.1 + EDR specification mandates:**

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt1. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

## 5.28.2    Conventional pairing examples

Successful pairing with a remote device when pin code is enabled with **SET BT AUTH** (no SSP).

---

**PAIR 00:07:80:12:34:56**

PAIR  00:07:80:12:34:56 OK

---

Unsuccessful pairing with a remote device when pin code is enabled with SET BT AUTH (no SSP).

---

**PAIR 00:07:80:12:34:56**

PAIR  00:07:80:12:34:56 FAIL

---

## 5.28.3    Secure Simple Pairing examples

Successful Secure Simple Pairing with "Just Works" mode (SET BT SSP 1 0). With the "just works" mode users do not need to use any PIN code, but it is automatically generated and exchanged by the Bluetooth devices.

---

**PAIR 00:07:80:12:34:56**

PAIR  00:07:80:12:34:56 OK

---

Secure Simple Pairing with Man-in-the-Middle (MITM) protection enabled.

Device 1:

> BD_ADDR: 00:07:80:81:66:8c
>
> SSP mode: "SET BT SSP 0 1" (display only, MITM enabled)

Device 2:

> BD_ADDR 00:07:80:89:a4:85
>
> SSP mode: "SET BT SSP 2 1" (keyboard only, MITM enabled)

Device 1:

---

**PAIR 00:07:80:93:d7:66**

SSP PASSKEY 00:07:80:93:d7:66 633237

PAIR 00:07:80:89:a4:85 OK

---

Device 2:

---

SSP PASSKEY 00:07:80:ff:ff:f1 ?

**SSP PASSKEY 00:07:80:81:66:8c 633237**

---

1.  First **PAIR** command is issued on device 1.
2.  Then **SSP PASSKEY** event is displayed on device 1 and a 6 digit number is displayed for numeric comparison.
3.  A **SSP PASSKEY** event is displayed on device 2 to indicate that numeric comparison needs to be made.
4.  The numeric comparison is responded with **SSP PASSKEY** command on device 2 and the 6 digit  number is given as a parameter.
5.  If the number is correct paring is successful and this is indicated on the device that initiated pairing.

Successful SSP pairing with Man-in-the-Middle (MITM) protection enabled.

Device 1:

> BD_ADDR: 00:07:80:81:66:8c
>
> SSP mode: "SET BT SSP 1 1" (display + yes/no button, MITM enabled)

Device 2:

> BD_ADDR 00:07:80:89:a4:85
>
> SSP mode: "SET BT SSP 1 1" (display + yes/no button, MITM enabled)

Device 1:

---

**PAIR 00:07:80:89:a4:85**

SSP CONFIRM 00:07:80:89:a4:85 951521 ?

**SSP CONFIRM 00:07:80:89:a4:85 OK**

PAIR 00:07:80:89:a4:85 OK

---

Device 2:

---

SSP CONFIRM 00:07:80:81:66:8c 951521 ?

**SSP CONFIRM 00:07:80:81:66:8c OK**

---

1. First **PAIR** command is issued on device 1.

2. Then **SSP CONFIRM** event is displayed on device 1 and a number is displayed for numeric comparison.

3. A **SSP CONFIRM** event is displayed on device 2 to indicate that numeric comparison needs to be made.

4. Both devices need to acknowledge that the number displayed on both devices is the same. This is done with **SSP CONFIRM** command.

5. If the number is correct paring is successful and this is indicated on the device that initiated pairing.

## 5.29 PING

The **PING** command sends a Bluetooth test packet to the other device, which sends the packet back and the round trip time of the packet is shown.

### 5.29.1 Syntax

| Synopsis: |
|---|
| **PING {*link_id*}** |

| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier |

| Response: | |
|---|---|
| **RSSI {*bd_addr*} {*round trip time*}** | |
| ***bd_addr*** | Bluetooth address of the remote device |
| ***round trip time*** | Round trip time of the packet |

| Events: |
|---|
| None |

### 5.29.2 Examples

Checking the round trip time:

| |
|---|
| **PING 0** |
| PING 00:07:80:80:c3:4a 42   (Round trip time is 42ms.) |

## 5.30 PIO

The command **PIO** is used to get and set PIO states and directions.

### 5.30.1　Syntax

| Synopsis |
| --- |
| **PIO {***command***} [***mask] [states***]** |

| Description | |
| --- | --- |
| *command* | **GET**<br><br>Read the contents of the PIO register. Bits that are set denote pins that are pulled up.<br><br>**GETDIR**<br><br>Read the contents of the PIO direction register. Bits that are set denote output pins; others are input pins.<br><br>**GETBIAS**<br><br>Read the contents of the PIO bias register. Bits that are set denote pins that are pulled up/down strongly, others are pulled up/down weakly.<br><br>**SET {***mask***} {***states***}**<br><br>Set the contents of the PIO register; the first parameter is the bit mask for deciding which PIOs are affected, the second parameter is the bits to set/unset.<br><br>**SETDIR {***mask***} {***states***}**<br><br>Set the contents of the PIO direction register. By default, only bit 8 (PIO7) is set, thus only it can be controlled locally with PIO SET, and all others are input pins.<br><br>**SETBIAS {***mask***} {***states***}**<br><br>Set the contents of the PIO bias register. By default, all pins are pulled up/down weakly.<br><br>**RESET**<br><br>Set the registers to iWRAP defaults. |
| *mask* | The hexadecimal bitmask that defines which PIOs are affected. |
| *states* | The hexadecimal bitmask that defines the states of the PIOs specified by *mask*. |

| Response | |
|---|---|
| | None for set commands. |
| **PIO GET {*state*}** | Response for PIO GET; displays PIO register value. |
| **PIO GETDIR {*state*}** | Response for PIO GETDIR. |
| **PIO GETBIAS {*state*}** | Response for PIO GETBIAS. |

| Events |
|---|
| None |

## 5.30.2    Examples

Playing with PIO7

| | |
|---|---|
| **PIO GET** | (Read PIO statues) |
| **PIO GET 0** | |
| **PIO SETDIR 80 80** | **(Sets PIO7 to output)** |
| **PIO SET 80 80** | (Sets PIO7 high) |
| **PIO GETDIR** | (Reads PIO directions) |
| PIO GETDIR 80 | |
| **PIO GET** | (Reads PIO statuses) |
| PIO GET 80 | |
| **PIO RESET** | (Reset PIOs) |
| **PIO GETDIR** | |
| PIO GETDIR 0 | |
| **PIO GET** | |
| PIO GET 0 | |

**Note:**

- There are 6 usable IO pins (PIO2-PIO7) on the WT11/12/41 modules and 11 GPIO lines (PIO0-PIO10) on the WT32. Therefore the range for the mask and state parameters for the WT11/12 is 4-FF and for the WT32 it is 0-07FF.

- The default values for the PIO registers are all zero; except for the WT11-A/E the direction register is set so that PIO0 and PIO1 are outputs.

- Switches on the evaluation kits can also affect PIO values. For instance, if on the WT32 evaluation kit PIO8 is routed to USB and the USB charger is in place, PIO8 will be high.

## 5.31 PLAY

Command **PLAY** is used to generate tones or beeps.

### 5.31.1 Syntax

| Synopsis: |
|---|
| **PLAY {*string*}** |

| Description: | |
|---|---|
| ***string*** | String of tones to play |
| | If empty string is given iWRAP stops playing the previous ringtone. |
| | **\*** |
| | WHOLENOTE |
| | **+** |
| | HALFNOTE |
| | **-** |
| | QUARTERNOTE |
| | **;** |
| | EIGHTHNOTE |
| | **:** |
| | SIXTEENTHNOTE |
| | **,** |
| | THIRTYSECONDNOTE |
| | **.** |
| | SIXTYFOURTHNOTE |
| | **a-g** |
| | notes |
| | **0-9** |
| | selects octave for the following notes, 4 by default |
| | **_** |
| | rest |
| | **!** |

| | | timbre sine(default) |
|---|---|---|
| | **"** | timbre square |
| | **#** | timbre saw |
| | **%** | timbre triangle |
| | **&** | timbre triangle2 |
| | **/** | timbre clipped sine |
| | **(** | timbre plucked |

| Response: | |
|---|---|
| **PLAY OK** | Returned when play command has finished |
| **PLAY BUSY** | Returned if previous play command is still being executed |

| Events: |
|---|
| None. |

Modern desk phone ring:

**PLAY 6,gfgfgf__gfgfgf_____gfgfgf__gfgfgf**
PLAY OK

## 5.32 RFCOMM

Command **RFCOMM** is used to create a RFCOMM channel for general RFCOMM connections.

### 5.32.1 Syntax

| Synopsis: |
|---|
| **RFCOMM {*action*}** |

| Description: | |
|---|---|
| *action* | **CREATE** |
| | Creates a generic RFCOMM channel. |

| Response: | |
|---|---|
| **RFCOMM {*channel*}** | |
| *channel* | RFCOMM channel number |

| Events: |
|---|
| None |

### 5.32.2 Examples

Creating a generic RFCOMM channel.

| **RFCOMM CREATE** |
|---|
| RFCOMM 2 |

## 5.33 RESET

Command **RESET** is used to perform a software reset.

### 5.33.1    Syntax

| Synopsis: |
|---|
| **RESET** |

| Description: |
|---|
| No description |

| Response: |
|---|
| No response |

## 5.34 RSSI

The **RSSI** command returns the Receiver Signal Strength Indication of the link given as a parameter.

### 5.34.1    Syntax

| Synopsis: |
|---|
| **RSSI {*link_id*}** |

| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier |

| Response: | |
|---|---|
| **RSSI {*bd_addr*} {*rssi*}** | |
| ***bd_addr*** | Bluetooth address of the remote device |
| ***rssi*** | Receiver Signal Strength Indication. Possible values are from +20 to  -128.<br><br>20 = Good link<br><br>-128 = Poor link |

| Events: |
|---|
| None |

### 5.34.2    Examples

Checking the RSSI of an active connection:

| |
|---|
| **LIST**<br><br>LIST 1<br><br>LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE MASTER PLAIN<br><br>**RSSI 0**<br>RSSI 00:60:57:a6:56:49 -10                          (RSSI is -10) |

Bluegiga Technologies Oy

## 5.35 SCO ENABLE

The **SCO ENABLE** command enables support for SCO (audio) connections. This command is needed if SCO connections are used none of the audio profiles (HFP or HSP) are enabled.

### 5.35.1    Syntax

| Synopsis: |
|---|
| **SCO ENABLE** |

| Description: |
|---|
| None |

| Response: |
|---|
| None |

| Events: |
|---|
| None |

**Note:**

- The SCO ENABLE command must be given every time after reset; it is not stored on flash memory.

- "SET CONTROL INIT" can be used to automatically issue one "SCO ENABLE" command.

- IF HFP or HSP profiles are enabled SCO ENBLED command is not needed.

## 5.36 SCO OPEN

The **SCO OPEN** command is used to open a SCO connection on top of an existing RFCOMM link.

### 5.36.1    Syntax

| Synopsis: |
|---|
| **SCO OPEN {*link_id*}** |

| Description: | |
|---|---|
| *link_id* | Numeric connection identifier |

| Response: |
|---|
| None |

| Response: |
|---|
| None |

| Events: | |
|---|---|
| **CONNECT** | If SCO connection was opened successfully |
| **NO_CARRIER** | If connection opening failed |

**Note:**

- The SCO ENABLE command must be given before the SCO OPEN command can be used.

### 5.36.2    Examples

Creating an SCO connection to another iWRAP device:

| |
|---|
| **SCO ENABLE**<br><br>**CALL 00:07:80:80:52:27 1 RFCOMM**<br>CALL 0<br>CONNECT 0 RFCOMM 1<br>**[+++]**                        (Command to data mode transition) |

Bluegiga Technologies Oy

**SCO OPEN 0**            (SCO is opened on top of the existing RFCOMM link with ID 0)

CONNECT 1 SCO

## 5.37 SDP

The **SDP** command can be used to browse the available services on other Bluetooth devices.

### 5.37.1    Syntax

| Synopsis: |
|---|
| **SDP {*bd_addr*} {*uuid*} [*ALL*]** |

| Description: | |
|---|---|
| ***bd_addr*** | Bluetooth address of the remote device |
| ***uuid*** | Service to look for<br>UUID "1002" stands for root and returns all the services the remote device supports. |
| ***ALL*** | Optional flag to read all the SDP information from the remote device. |

| Response: | |
|---|---|
| SDP      {***bd_addr***}     <     I     SERVICENAME     S     "**service_name**"     ><br>< I PROTOCOLDESCRIPTORLIST < < U L2CAP ***psm***> < U RFCOMM I ***channel*** > > ><br><br>**SDP** | |
| ***bd_addr*** | Bluetooth address of the remote device |
| ***service name*** | Name of the service. For example "Serial Port Profile" |
| ***psm*** | L2CAP psm of the profile (if L2CAP based profile) |
| ***channel*** | RFCOMM channel of the profile (if RFCOMM based profile) |

| Events: |
|---|
| None |

## 5.37.2    Examples

Browsing the SDP root record to retrieve all SDP entries

---

**SDP 00:07:80:89:a4:85 1002**

SDP  00:07:80:89:a4:85  <  I  SERVICENAME  S  "Bluetooth  Serial  Port"  >  <  I PROTOCOLDESCRIPTORLIST < < U L2CAP > < U RFCOMM I 01 > > >

SDP 00:07:80:89:a4:85 < I SERVICENAME S "Stereo headset" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP I 19 > < U 0019 I 100 > > >

SDP

---

Searching for SPP profile

---

**SDP 00:07:80:93:d7:66 1101**

SDP  00:07:80:93:d7:66  <  I  SERVICENAME  S  "Bluetooth  Serial  Port"  >  <  I PROTOCOLDESCRIPTORLIST < < U L2CAP > < U RFCOMM I 01 > > >

SDP

---

Searching for SPP profile using the ALL flag.

---

**SDP 00:07:80:93:d7:66 1101 ALL**

SDP 00:07:80:93:d7:66 < I 0 I 10000 > < I 1 < U 00001101-0000-1000-8000-00805f9b34fb > > < I PROTOCOLDESCRIPTORLIST < < U L2CAP > < U RFCOMM I 01 > > > < I 5 < U BROWSE > > < I 6 < I 656e I 6a I 100 > > < I SERVICENAME S "Bluetooth Serial Port" >

SDP

---

Some devices return the protocol descriptions using 128-bit format and older iWRAP version could not parse them correctly. The response might therefore look like this. iWRAP4 can parse 128-bit protocol description lists and display them correctly.

---

**SDP 00:17:4b:67:a8:c3 1101**

SDP 00:17:4b:67:a8:c3 < I SERVICENAME S "Bluetooth SPP" > < I PROTOCOLDESCRIPTORLIST < < U 00000100-0000-1000-8000-00805f9b34fb > < U 00000003-0000-1000-8000-00805f9b34fb I 19 > > >

---

According to the Bluetooth specification:

00000100-0000-1000-8000-00805f9b34fb  =                                                                                    L2CAP
00000003-0000-1000-8000-00805f9b34fb  = RFCOMM

## 5.38 SDP ADD

The **SDP ADD** command can be used to modify a local service record to add new RFCOMM based services. This is useful if one wants to implement a Bluetooth profile iWRAP itself does not support.

### 5.38.1    Syntax

| Synopsis: |
|---|
| **SDP ADD {*uuid*} {*name*}** |


| Description: | |
|---|---|
| ***uuid*** | Identifier of the service<br><br>**uuid16**<br><br>   16-bit UUID<br>   Format: xxxx (hex)<br><br>**uuid32**<br><br>   32-bit UUID<br>   Format: xxxxxxxx (hex)<br><br>**uuid128**<br><br>   128-bit UUID<br>   Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (hex) |
| ***name*** | Name of the service |


| Response: | |
|---|---|
| **SDP {*channel*}** | |
| ***channel*** | RFCOMM channel where the service is bound to |


| Events: |
|---|
| None |

## 5.38.2    Examples

Adding a Dial-Up Networking profile

---
**SDP ADD 1103 Dial-Up Networking**

SDP 2

---

**Note:**

- The service record will be cleared when a reset is made, so **SDP ADD** command(s) must be given every time after a reset.

- "**SET CONTROL INIT**" can be used to automatically run "**SDP ADD**" command after a reset.

## 5.39 SELECT

Command **SELECT** can be used to switch from command mode to data mode.

### 5.39.1 Syntax

| Synopsis: |
|---|
| **SELECT {*link_id*}** |

| Description: | |
|---|---|
| *link_id* | Numeric connection identifier |

| Response: |
|---|
| No response if a valid link is selected. iWRAP goes to data mode of the link *link_id*. |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if an invalid *link_id* is given |

### 5.39.2 Examples

Changing between links:

| |
|---|
| **LIST** |
| LIST 2 |
| LIST 0 CONNECTED RFCOMM 668 0 0 243 8d 8d 00:07:80:80:38:77 1 OUTGOING ACTIVE MASTER ENCRYPTED |
| LIST 1 CONNECTED RFCOMM 668 0 0 419 8d 8d 00:07:80:80:36:85 1 OUTGOING ACTIVE MASTER ENCRYPTED |
| **SELECT 1**                    (iWRAP switches  to data mode with link ID 1) |

## 5.40 SET

With the **SET** command, you can display or configure different iWRAP configuration values.

### 5.40.1    Syntax of SET Commands

| Synopsis: |
| --- |
| **SET [{*category*} {*option*} {*value*}]** |

| Description: |
| --- |
| Without any parameters, **SET** displays the current configuration. |

| | |
| --- | --- |
| *category* | Category of setting<br><br>**BT**<br><br>    Changes different Bluetooth related settings. See **SET BT** commands for more information about options.<br><br>**CONTROL**<br><br>    Changes different iWRAP settings. See **SET CONTROL** commands for more information about options.<br><br>**PROFILE**<br><br>    Activates or deactivates Bluetooth profiles.<br><br>**link_id**<br><br>    This command is used to control the various settings related to Bluetooth links in iWRAP. These are, for example, master, slave and power save modes (SNIFF and ACTIVE). |
| *option* | Option name, which depends on the category. See the following sections for more information. |
| *value* | Value for the option. See the following sections for more information. |

| **Response:** | |
|---|---|
| None if issued with parameters | |
| **SET {*category*} {*option*} {*value*}** | If no parameters given displays current iWRAP settings. |

| **Events:** |
|---|
| None |

## 5.40.2    Examples

Listing current settings:

```
SET
SET BT BDADDR 00:07:80:80:c2:37
SET BT NAME WT12
SET BT CLASS 50020c
SET BT AUTH * 9078
SET BT LAP 9e8b33
SET BT PAGEMODE 4 2000 1
SET BT PAIR 00:07:cf:51:f6:8d 9c4e70d929a83812a00badba7379d7c2
SET BT PAIR 00:14:a4:8b:76:9e 90357318b33817002c5c13b62ac6507f
SET BT PAIR 00:60:57:a6:56:49 3b41ca4f42401ca64ab3ca3303d8ccdc
SET BT ROLE 0 f 7d00
SET BT SNIFF 0 20 1 8
SET CONTROL BAUD 115200,8n1
SET CONTROL CD 80 0
SET CONTROL ECHO 7
SET CONTROL ESCAPE 43 00 1
SET
```

## 5.41 SET BT AUTH

**SET BT AUTH** shows or sets the local device's PIN code.

### 5.41.1    Syntax

| Synopsis: |
|---|
| **SET BT AUTH {*mode*} {*pin_code*}** |

| Description: | |
|---|---|
| *mode* | Pin code usage mode:<br><br>*<br><br>      Pin code will be displayed by "**SET**" command.<br><br>-<br><br>      Pin code will NOT be displayed by "**SET**" command. |
| *pin_code* | PIN code for authorized connections. Authorization is required if this option is present. The PIN code can be from 0 to 16 characters. |

| Response: |
|---|
| No response |

| Events: |
|---|
| None |

| List format: | |
|---|---|
| | If PIN code is not set "**SET BT AUTH \***" is not displayed |
| **SET BT AUTH \* {*pin_code*}** | If PIN code is set |
| **SET BT AUTH \*** | If pin code set with "**SET BT AUTH –**" |

**Note:**

If command "SET BT AUTH *" is given, PIN code will be disabled and no encryption can be used during Bluetooth connections.

Bluegiga Technologies Oy

## 5.42 SET BT BDADDR

The string **SET BT BDADDR** is displayed when the **SET** command is issued. It is meant to show the Bluetooth address of the local device. This is not a command, so it cannot be used to edit the module's MAC address or to read the address as an alternative to using the SET command.

### 5.42.1 Syntax

| Synopsis: |
| --- |
| None |

| Description: |
| --- |
| None |

| Response: |
| --- |
| None |

| Events: |
| --- |
| None |

| List format: |
| --- |
| **SET BT BDADDR {*bd_addr*}** |
| **_bd_addr_**        Bluetooth device address of the local device |

## 5.43 SET BT CLASS

**SET BT CLASS** sets the local device's Bluetooth Class-of-Device (CoD). Class of device is a parameter, which is received during the device discovery procedure, indicating the type of device and which services are supported.

### 5.43.1 Syntax

| Synopsis: |
|---|
| **SET BT CLASS {*class_of_device*}** |

| Description: | |
|---|---|
| ***class_of_device*** | Bluetooth Class-of-Device of the local device<br><br>**AUTO**<br><br>        If this flag is used iWRAP automatically sets the class of device during boot time. |

| Response: |
|---|
| None |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| List format: |
|---|
| **SET BT CLASS {*class_of_device*}** |

**Note:**

- The class-of-device parameter should reflect the features and supported profiles of a Bluetooth device. Refer to the Bluetooth specification for more information.

- A useful tool to work out Class of Device can be found from:

  http://bluetooth-pentest.narod.ru/software/bluetooth_class_of_device-service_generator.html

Bluegiga Technologies Oy

## 5.44 SET BT IDENT

This command changes the device identification information. Only the freeform description can be changed; the first four parameters exist for the sake of conformity. A reset is needed for the setting to take place.

### 5.44.1    Syntax

| Synopsis: |
|---|
| **SET BT IDENT {*src*}:{*vendor_id*} {*product_id*} {*version*} [*descr*]** |

| Description: | |
|---|---|
| *src* | This attribute indicates which organization assigned the VendorID attribute. There are two possible values: BT for the Bluetooth Special Interest Group (SIG) or USB for the USB Implementer's Forum. |
| *vendor_id* | Intended to uniquely identify the vendor of the device. The Bluetooth SIG or the USB IF assigns VendorIDs. Bluegiga's VendorID is 47. |
| *product_id* | Intended to distinguish between different products made by the vendor in question. These IDs are managed by the vendors themselves, and should be changed when new features are added to the device. |
| *version* | Vendor-assigned version string indicating device version number. This is given in the form of major.minor.revision, for example "3.0.0". |
| *descr* | Optional freeform product description string. |

| Respone: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| Events: |
|---|
| None |

| List format: |
|---|
| **SET BT IDENT {*src*}:{*vendor_id*} {*product_id*} {*version*} [*descr*]** |

## 5.44.2      Examples

Changing the description string:

**SET BT IDENT BT:47 f000 3.0.0 My Description String**

**RESET**

## 5.45 SET BT LAP

This command configures the Inquiry Access code (IAC) that iWRAP uses. IAC is used in inquiries and inquiry responses.

### 5.45.1 Syntax

| Synopsis: |
|---|
| **SET BT LAP {*iac*}** |

| Description: | |
|---|---|
| ***iac*** | Value for the inquiry access code. The following values are possible: |
| | **9e8b33** |
| | General/Unlimited Inquiry Access Code (GIAC). This is the default value. |
| | **9e8b00** |
| | Limited Dedicated Inquiry Access Code (LIAC). |
| | **9e8b01 - 9e8b32** and **9e8b34-9e8b3f** |
| | Reserved for future use. |

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| Events: |
|---|
| None. |

| List format: |
|---|
| **SET BT LAP {*iac*}** |

**Note:**

- IAC is very useful in cases where the module needs to be visible in the inquiry but only for dedicated devices, such as other iWRAP modules, but not for standard devices like PCs or mobile phones. When the value of IAC is left to default value "**9E8B33"** (GIAC**)** iWRAP will be visible for all devices capable of making an inquiry. On the other, hand when IAC is set to **9E8B00** (LIAC), only devices capable of making limited inquiry will be able to discover iWRAP. Using LIAC will usually speed up the inquiry process since standard Bluetooth device like mobile phones and PC will normally not respond to inquiry.

## 5.46 SET BT MTU

**SET BT MTU** configures the Maximum Transfer Unit (packet size) for Bluetooth RFCOMM connections. iWRAP tries to use this MTU by default for all the Bluetooth connections.

### 5.46.1 Syntax

| Synopsis: |
|---|
| **SET BT MTU {*mtu*}** |

| Description: | |
|---|---|
| *mtu* | Maximum Transfer Unit. Valid range: 22 to 1009 |

| Response: |
|---|
| None |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| List format: |
|---|
| **SET BT NAME {*mtu*}** |

**Note:**

- The remote device may not accept as large MTU as iWRAP wants to use and MTU may be limited to a smaller value.

### 5.46.2 Examples

Changing the default MTU 1000 bytes.

| **SET BT MTU 1000** |
|---|

Bluegiga Technologies Oy

## 5.47 SET BT NAME

**SET BT NAME** configures the local device's friendly name.

### 5.47.1 Syntax

| Synopsis: |
| --- |
| **SET BT NAME {*friendly_name*}** |

| Description: | |
| --- | --- |
| *friendly_name* | Friendly name of the local device |

| Response: |
| --- |
| None |

| Events: | |
| --- | --- |
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| List format: |
| --- |
| **SET BT NAME {*friendly_name*}** |

**Note:**

- The maximum length of a friendly name is 16 characters in iWRAP 2.0.2 and older. In iWRAP 2.1.0 and newer versions, the maximum length is 256 characters.

- If *friendly_name* is left empty, some devices (like PCs or PDAs) may have problems showing the device in the inquiry.

## 5.48 SET BT PAIRCOUNT

This command can be used to set the maximum amount of pairings iWRAP will accept.

### 5.48.1 Syntax

| Synopsis |
| --- |
| **SET BT PAIRCOUNT {*max_pairings*}** |

| Description | |
| --- | --- |
| ***max_pairings*** | **0-16** <br><br> Valid values are 0-16 (decimal). Values 0 and 16 disable pair count limiting. |

| Response |
| --- |
| None |

| Events | |
| --- | --- |
| **PAIR                    {bd_addr} ERR_MAX_PAIRCOUNT** | This event will be received when the maximum number of pairings already exists, and the pair event config bit is set, and the automatically delete old pairings config bit is not set. |

| List format |
| --- |
| **SET BT PAIRCOUNT {*max_pairings*}** |

**Note:**

It is highly recommended that config bit 12 (automatically make room for new pairings) is set, because if the maximum pair count is reached and a remote party wishes to pair to us, they may see a successful pairing followed by a failed connection attempt, because we have no room to store the new link key – while at the same time they have stored it.

Also, **SET BT PAIRCOUNT** should never be issued before all the pairings are cleared, because it may not parse partially filled pairing tables correctly. When using **SET BT PAIRCOUNT**, you should set it only once. If you need to change the pairing count, delete all old pairings before doing it.

## 5.49 SET BT PAGEMODE

**SET BT PAGEMODE** configures or displays the local device's page mode.

Page mode controls whether iWRAP can be seen in the inquiry and whether it can be connected. This command can also be used to change the page timeout.

### 5.49.1    Syntax

| Synopsis: |
|---|
| SET BT PAGEMODE {*page_mode*} {*page_timeout*} {*page_scan_mode*} |


| Description: | |
|---|---|
| *page_mode* | This parameter defines the Bluetooth page mode.<br><br>**0**<br><br>iWRAP is NOT visible in the inquiry and does NOT answers calls<br><br>**1**<br><br>iWRAP is visible in the inquiry but does NOT answers calls<br><br>**2**<br><br>iWRAP is NOT visible in the inquiry but answers calls<br><br>**3**<br><br>iWRAP is visible in the inquiry and answers calls<br><br>**4**<br><br>Just like mode 3 if there are NO connections. If there are connections, it is like mode 0. (default value) |
| *page_timeout* | **0001 – FFFF**<br><br>Page timeout defines how long the connection establishment can take before an error occurs. Page timeout is denoted as a hexadecimal number (HEX) and calculated as in the example below:<br><br>2000 (HEX) equals 8192 (DEC). Multiply it by 0.625 and you get the page timeout in milliseconds. In this case, it is 5120 ms (8192 * 0,625ms). |
| *page_scan_mode* | This parameter configures the Bluetooth page scan mode. The possible values are:<br><br>**0**<br><br>Mode R0 means that iWRAP IS connectable all the time. High current consumption! Since iWRAP is all the time connectable, it will not be visible in the inquiry, no matter what the page mode configuration is. |

| | 1 | |
|---|---|---|
| | | Mode R1 means that iWRAP is connectable every 1.28 sec (the default value) |
| | 2 | |
| | | Mode R2 means that iWRAP is connectable every 2.56 sec (lowest power consumption) |

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| Events: |
|---|
| None |

| List format: |
|---|
| **SET BT PAGEMODE {*page_mode*} {*page_timeout*} {*page_scan_mode*}** |

**Note:**

- If page scan mode is set to 0 iWRAP will be visible even if page mode is set to 1.
- Command "**SET BT PAGEMODE**" returns default values.

## 5.50 SET BT PAIR

**SET BT PAIR** displays or configures the local device's pairing information.

### 5.50.1 Syntax

| Synopsis: |
|---|
| **SET BT PAIR {*bd_addr*} {*link_key*}** |


| Description: | |
|---|---|
| ***bd_addr*** | Bluetooth address of the paired device |
| ***link_key*** | Link key shared between the local and the paired device. If this value is empty, pairing for the given Bluetooth address will be removed. Link key is 32hex values long. |


| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |


| Events: |
|---|
| None |


| List format: | |
|---|---|
| | SET BT PAIR is not displayed if there are no pairings |
| **SET BT PAIR {*bd_addr*} {*link_key*}** | One line per pairing is displayed |

**Note:**

- iWRAP supports up to 16 simultaneous pairings. If 16 devices have been already paired, new pairings will not be stored.

- If command "**SET BT PAIR \***" is given, all pairings will be removed.

## 5.51 SET BT POWER

This command changes the TX power parameters of the Bluetooth module. Notice that **SET BT POWER** will automatically round the powers levels to the closest value which exists in a so called radio power table.

### 5.51.1    Syntax

| Synopsis: |
|---|
| **SET BT POWER [RESET] | [{*default*} {*maximum*} [*inquiry*]]** |

| Description: | |
|---|---|
| | If no parameters are given, displays current TX power settings. |
| *RESET* | Returns default TX power values and resets iWRAP |
| *default* | Default TX power in dBm. The default TX power used for **CALL** and **NAME** operations and when responding to inquiries and connection requests. |
| *maximum* | Maximum TX power in dBm. Bluetooth power control may raise the TX power up to this value. |
| *inquiry* | Transmit power in dBm used for **INQUIRY** operation. This is an optional parameter introduced in iWRAP version 3.0.0: if not given, inquiry power is unchanged; by default is equal to the default TX power. |

| Respone: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| Events: |
|---|
| None |

| List format: |
|---|
| **SET BT POWER {*default*} {*maximum*} {*inquiry*}** |

## 5.51.2    Examples

Change the default TX power to 0, maximum TX power to 4 and inquiry power to 0.

---

**SET BT POWER 0 4 0**

---

**Note:**

Please see the table below for the Bluetooth power classes:

| Power class: | Max. TX power: | Nominal TX power: | Minimum TX power: |
|:---:|:---:|:---:|:---:|
| 1 | 20 dBm | N/A | 0dBm |
| 2 | 4dBm | 0dBm | -6 dBm |
| 3 | 0dbm | N/A | N/A |

**Table 9: Power TX power classes as defined in Bluetooth specification**

- The values passed with "**SET BT POWER**" will always be rounded to the next available value in the radio power table.

- If possible, always use default values!

## 5.52 SET BT ROLE

This command configures or displays the local device's role configuration. With the "**SET BT ROLE**" command, iWRAP's master-slave behavior can be configured. This command can also be used to set the supervision timeout and link policy.

### 5.52.1 Syntax

| Synopsis: |
| --- |
| **SET BT ROLE {*ms_policy*} {*link_policy*} {*supervision_timeout*}** |


| Description: | |
| --- | --- |
| *ms_policy* | This parameter defines how the master-slave policy works. |
| | **0** |
| | This value allows master-slave switch when calling, but iWRAP does not request it when answering (default value). |
| | **1** |
| | This value allows master-slave switch when calling, and iWRAP requests it when answering. |
| | **2** |
| | If this value is set, master-slave switch is not allowed when calling, but it is requested for when answering. |
| | This bitmask controls the link policy modes. It is represented in a hexadecimal format. |
| *link_policy* | **Bit 1** |
| | If this bit is set, Role switch is enabled |
| | **Bit 2** |
| | If this bit is set, Hold mode is enabled |
| | **Bit 3** |
| | If this bit is set, Sniff mode is enabled |
| | **Bit 4** |
| | If this bit is set, Park state is enabled |
| | **F** |
| | This value enables all of the above modes (the default value) |
| | **0** |

| | |
|---|---|
| | This value disables all of the above modes |
| *supervision_timeout* | **0001 – FFFF**<br><br>Supervision timeout controls how long a Bluetooth link is kept open if the remote end does not answer. Supervision timeout is denoted as a hexadecimal number (HEX) and is calculated as in the example below:<br><br>12C0 (HEX) is 4800 (DEC). Multiply it by 0,625 and you get the supervision timeout in milliseconds. In this case, it is 3000 ms (4800 * 0,625ms).<br><br>In other words, the remote end can be silent for three seconds until the connection is closed. |

| **Response:** |
|---|
| None |

| **Events:** | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| **List format:** |
|---|
| **SET BT ROLE {*ms_policy*} {*link_policy*} {*supervision_timeout*}** |

**Note:**

- If a master-slave switch occurs during the connection setup the supervision timeout set with **SET BT ROLE** in the master device will not be used, unless **SET CONTROL CONFIG** bit 3 in config block 1 is set. This forces iWRAP to update the supervision timeout after a master-slave switch.
- Command **"SET BT ROLE"** restores default values.

## 5.53 SET BT SNIFF

This command enables or disables automatic sniff mode for Bluetooth connections. Notice that remote devices may not support sniff.

### 5.53.1      Syntax

| Synopsis: |
|---|
| **SET BT SNIFF {*max*} {*min*} [{*attempt*} {*timeout*}]**<br>or<br>**SET BT SNIFF 0** |

| Description: | |
|---|---|
| *max* | Maximum acceptable interval in milliseconds<br>Range: **0004** to **FFFE**; only even values are valid<br>Time = *max* * 0.625 msec<br>Time Range: 2.5 ms to 40959 ms |
| *min* | Minimum acceptable interval in milliseconds<br>Range: **0002** to **FFFE**; only even values, up to *max*, are valid<br>Time = *min* * 0.625 ms<br>Time Range: 1.25 ms to 40959 ms |
| *attempt* | Number of Baseband receive slots for sniff attempt.<br>Length = N* 1.25 ms<br>Range for N: **0001 – 7FFF**<br>Time Range: 0.625ms - 40959 ms |
| *timeout* | Number of Baseband receive slots for sniff timeout.<br>Length = N * 1.25 ms<br>Range for N: **0000 – 7FFF**<br>Time Range: 0 ms - 40959 ms |

| Response: |
|---|
| None |

| Events: |
|---|

| SYNTAX ERROR | This event occurs if incorrect parameters are given |
|---|---|

**List format:**

**SET BT SNIFF {*max*} {*min*} {*attempt*} {*timeout*}**

**Note:**

- Supervisor timeout set with "**SET BT ROLE**" must be longer than maximum acceptable sniff interval.

- "**SET BT SNIFF 0**" disables automatic sniff mode (this is the default, shown in the output of **SET** command at line "**SET BT SNIFF 0 20 1 8**").

- You can not change sniff mode on the fly with "**SET BT SNIFF**", but you need to close all active Bluetooth connections, then change the sniff setting and reopen the connections. If you want to be able to control the sniff mode and keep the connections active, disable "**SET BT SNIFF**" use command "**SET {link_id} SNIFF**" instead.

## 5.54 SET BT SSP

This command enabled or disables the Bluetooth 2.1 + EDR compliant Secure Simple Pairing mode.

### 5.54.1    Syntax

| Synopsis |
|---|
| **SET BT SSP {*capabilities*} {*mitm*}** |

| Description | | |
|---|---|---|
| *capabilities* | **0** | Display only |
| | **1** | Display + yes/no button |
| | **2** | Keyboard only |
| | **3** | None |
| *mitm* | **0** | Man-in-the-middle protection disabled |
| | **1** | Man-in-the-middle protection enabled |

| Response |
|---|
| None |

| Events |
|---|
| None |

| List format |
|---|
| **SET BT SSP {*capabilities*} {*mitm*}** |

Bluegiga Technologies Oy

**Note:**

- According to the Bluetooth 2.1 + EDR specification SSP pairing must always be enabled.

The Bluegiga recommendation is that that the "just works" mode is enabled and to support older devices with out SSP also the PIN code is enabled. If the remote device will not support SSP iWRAP will fall back to PIN code. Recommended configuration is therefore

**"SET BT SSP 3 0"**

**"SET BT AUTH * {*pin*}"**

- Man in the middle protection does not work if either end claims to be a "display only" device while the other end is "display with buttons".

| Initiator A / B Responder | Display Only | DisplayYesNo | KeyboardOnly | NoInputNoOutput |
|---|---|---|---|---|
| **DisplayOnly** | Numeric Comparison with automatic confirmation on both devices. <br><br> Unauthenticated | Numeric Comparison with automatic confirmation on device B only. <br><br> Unauthenticated | Passkey Entry: Responder Display, Initiator Input. <br><br> Authenticated | Numeric Comparison with automatic confirmation on both devices. <br><br> Unauthenticated |
| **DisplayYesNo** | Numeric Comparison with automatic confirmation on device A only. <br><br> Unauthenticated | Numeric Comparison: Both Display, Both Confirm. <br><br> Authenticated | Passkey Entry: Responder Display, Initiator Input. <br><br> Authenticated | Numeric Comparison with automatic confirmation on device A only. <br><br> Unauthenticated |
| **Keyboard Only** | Passkey Entry: Initiator Display, Responder Input. <br><br> Authenticated | Passkey Entry: Initiator Display, Responder Input. <br><br> Authenticated | Passkey Entry: Initiator and Responder Input. <br><br> Authenticated | Numeric Comparison with automatic confirmation on both devices. <br><br> Unauthenticated |
| **NoInputNoOutput** | Numeric Comparison with automatic confirmation on both devices. <br><br> Unauthenticated | Numeric Comparison with automatic confirmation on device B only. <br><br> Unauthenticated | Numeric Comparison with automatic confirmation on both devices. <br><br> Unauthenticated | Numeric Comparison with automatic confirmation on both devices. <br><br> Unauthenticated |

**Figure 4: IO capability mapping to authentication stage 1**

## 5.55 SET CONTROL AUDIO

This command controls the physical interface routing of audio on WT32. The command is also used to enable or disable multiple SCO support.

### 5.55.1 Syntax

| Synopsis |
| --- |
| **SET CONTROL AUDIO {*sco_routing*} {*a2dp_routing*} [MULTISCO] [EVENT]** |

| Description | |
| --- | --- |
| ***sco_routing***<br><br>***a2dp_routing*** | **INTERNAL**<br><br>    Routes SCO/A2DP input and output through the internal PCM codec to the analogue input and output.<br><br>**PCM**<br><br>    Routes SCO/A2DP to the PCM pins (see the WT32 schematic.)<br><br>**I2S**<br><br>    Routes SCO/A2DP to the PCM pins with the module acting as I2S master.<br><br>**I2S_SLAVE**<br><br>    Same as above, but acting as Slave.<br><br>**SPDIF**<br><br>    Routes SCO/A2DP to the PCM pins in S/PDIF encoding. Using the S/PDIF interface is not recommended for A2DP audio. Please see the WT32 datasheet for more information. |
| MULTISCO | Issue this to enable initiating two SCO connections at the same time. The first connection takes the left channel, the second the right. This is an experimental feature and may not always work correctly. |
| EVENT | Issue this to receive audio routing events (and DSP codec events in WT32.) |

| Response |
| --- |
| None |

| Events | |
|---|---|
| **AUDIO ROUTE {link_id} {type} {channels}** | This event occurs when the audio routings are changed; e.g. when an A2DP or SCO connection is started or closed. |
| | **{link_id}** indicates the link ID of the connection where the audio is received or sent. |
| | **{type}** indicates the type of the audio (SCO, A2DP or TUNE). |
| | **{channels}** indicates which channels are used for the audio: LEFT, RIGHT, or both. |
| **A2DP CODEC {codec} {channel_mode} {rate} BITPOOL 2-250** | This event occurs with a WT32 when a codec is loaded into the DSP, e.g. when an A2DP starts or resumes after SCO is disconnected. |
| | **{codec}** indicates which codec is used. Only SBC is included in the standard iWRAP. APT-X is integrated into a special version of iWRAP, which can be evaluated on demand; please contact our Sales department for further information. MP3 is also available on request. |
| | **{channel_mode}** can be JOINT_STEREO, STEREO, DUAL_CHANNEL or MONO. |
| | **{rate}** is the sampling rate in Hz. |

| List format |
|---|
| **SET CONTROL AUDIO {*sco_routing*} {*a2dp_routing*} [MULTISCO] [EVENT]** |

## 5.56 SET CONTROL AUTOCALL

**SET CONTROL AUTOCALL** enables or disables the AUTOCALL functionality in iWRAP.

When the AUTOCALL feature is enabled, iWRAP tries to form a connection with a paired (see **"SET BT PAIR"**) device until the connection is established. If the connection is lost or closed, iWRAP tries to reopen it.

If there are several paired devices in iWRAP memory, an inquiry (transparent to the user) is made and the first paired device found is connected.

### 5.56.1 Syntax

| Synopsis: |
|---|
| **SET CONTROL AUTOCALL {*target*} {*timeout*} {*profile*}** |

| Description: | |
|---|---|
| *target* | RFCOMM, HFP or HFP-AG, HID or A2DP target for the connection. The target can be one of the following:<br><br>**channel**<br><br>      RFCOMM channel number<br>      HFP channel number<br>      HFP-AG channel number<br>      Format: xx (hex)<br><br>**uuid16**<br><br>      16-bit UUID for searching channel<br>      Format: xxxx (hex)<br><br>**uuid32**<br><br>      32-bit UUID for searching channel<br>      Format: xxxxxxxx (hex)<br><br>**uuid128**<br><br>      128-bit UUID for searching channel<br>      Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (hex)<br><br>**L2CAP psm**<br><br>      16-bit L2CAP psm<br>      Format: xxxx (hex) |
| *timeout* | Timeout between calls (in milliseconds) |
| *profile* | Defines the connection mode to be established.<br>Possible modes are: |

Bluegiga Technologies Oy

**RFCOMM**

Normal RFCOMM connection

**HFP**

Opens a connection in the Hands Free device mode.

**HFP-AG**

Opens a connection in the Hands Free Audio Gateway mode.

**A2DP**

Opens a connection in the Advanced Audio Distribution Profile (A2DP) mode or Audio Video Remote Control Profile (AVRCP) mode. L2CAP psm for A2DP is 19 and for AVRCP 17.

**HID**

Opens a connection in the HID keyboard mode or HID mouse mode. L2CAP psm for HID is 11.

**L2CAP**

Opens a generic L2CAP connection.

**"Any other profile"**

Any other type of profile can also be used.

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| Events: |
|---|
| None |

| List format: | |
|---|---|
| | If AUTOCALL is not enabled, **"SET CONTROL AUTOCALL"** will not be displayed |
| **SET CONTROL AUTOCALL {*target*} {*timeout*} {*profile*}** | When AUTOCALL is enabled |

**Note:**

- If AUTOCALL is enabled no manual "CALL" commands should be given to iWRAP.

- INQUIRY commands may fail when AUTOCALL is enabled, because AUTOCALL makes inquiries (transparent to the user) if multiple devices are paired.

## 5.57 SET CONTROL BATTERY

This command enables low battery indication and automatic shutdown. This command is only for WT32 module.

### 5.57.1    Syntax

| Synopsis |
| --- |
| **SET CONTROL BATTERY {***low***} {***shutdown***} {***full***} {***mask***}** |


| Description | |
| --- | --- |
| *low* | When battery voltage drops below this level, iWRAP will start sending low battery warning events, and drives high the PIO(s) according to **mask**. Maximum value is 3700 (millivolts). |
| *shutdown* | When battery voltage drops below this level, iWRAP will automatically shut itself down to prevent the battery from completely draining. Maximum value is 3300 (millivolts). |
| *full* | When battery voltage rises above this level, the low battery warnings cease and the low battery indicator PIO(s) is/are driven low. |
| *mask* | Hexadecimal PIO mask to select the PIO(s) used to indicate low battery status. |


| Response |
| --- |
| None |


| Events | |
| --- | --- |
| **BATTERY LOW {voltage}** | This event indicates that the battery is low. iWRAP will keep sending these events until the battery is full.<br><br>**{voltage}** current battery voltage in millivolts. |
| **BATTERY SHUTDOWN {voltage}** | This event indicates that the battery voltage has fallen below the shutdown threshold. The module will shut down immediately.<br><br>**{voltage}** current battery voltage in millivolts. |
| **BATTERY FULL {voltage}** | This event indicates that the battery is full and low battery warnings will cease.<br><br>**{voltage}** current battery voltage in millivolts. |

Bluegiga Technologies Oy

| List format |
|---|
| **SET CONTROL BATTERY {*low*} {*shutdown*} {*full*} {*mask*}** |

## 5.58 SET CONTROL BAUD

This command changes the local device's UART settings.

### 5.58.1    Syntax

| Synopsis: |
|---|
| **SET CONTROL BAUD {*baud_rate*},8{*parity*}{*stop_bits*}** |

| Description: | |
|---|---|
| ***baud_rate*** | UART baud rate in bps. See modules data sheet for suitable values. |
| ***parity*** | UART parity setting<br><br>**n**<br><br>  No parity<br><br>**e**<br><br>  Even parity<br><br>**o**<br><br>  Odd parity |
| ***stop_bits*** | Number of stop bits in UART communications<br><br>**1**<br><br>  One stop bit<br><br>**2**<br><br>  Two stop bits |

| Response: |
|---|
| None |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

Bluegiga Technologies Oy

| List format: |
|---|
| **SET CONTROL BAUD {*baud_rate*},8{*parity*}{*stop_bits*}** |

## 5.58.2    Examples

Configuring local UART to 9600bps, 8 data bits, no parity and 1 stop bit

| |
|---|
| **SET CONTROL BAUD 9600,8N1** |

**Note:**

- If you enter an incorrect or invalid baud rate and can not access iWRAP any more, the only way to recover the module is via the SPI interface by deleting the value of PS-key : PSKEY_USR26. Please see chapter 9.1 PS-keys and how to change them for information how to change the PS-keys.

## 5.59 SET CONTROL BIND

With **SET CONTROL BIND**, it is possible to bind iWRAP commands to GPIO pins.

### 5.59.1 Syntax

| Synopsis: |
| --- |
| SET CONTROL BIND {*pri*} [*io_mask*] [*direction*] [*command*] |

| Description: | |
| --- | --- |
| *pri* | Command priority. Determines the order in which the commands bound to a PIO are executed (lowest *pri* is executed first). <br><br> *pri* range is 0-7 <br><br> *pri* is an absolute value for all bindings, so there cannot be two or more similar *pri* values (also across SET CONTROL BIND commands with different *io_mask*) <br><br> If only *pri* parameter is given, then the current bind will be removed. |
| *io_mask* | Determines which PIO is to be bind. <br><br> In WT12, WT11 and WT41 possible PIOs are PIO2 to PIO7 <br><br> In WT32 possible PIOs are PIO0 to PIO10 <br><br> This is a hexadecimal value. <br><br> Example: PIO5 is referred to by 100000bin (5th bit is one) = 20hex |
| *direction* | Determines whether PIO is triggered on rising, falling, or on both edges of the signal. <br><br> Possible values are: <br><br> **RISE** <br><br>     Command is executed on rising edge. <br><br> **FALL** <br><br>     Command is executed on falling edge. <br><br> **CHANGE** <br><br>     Command is executed on rising and falling edge. |
| *command* | Standard iWRAP command or string to be sent to the active Bluetooth link. <br><br> The maximum length for *command* is 31 characters. |

| Response: |
|---|
| No response |

| List format: | |
|---|---|
| | If no binding exists, **"SET CONTROL BIND"** will not be displayed |
| **SET CONTROL BIND** {*pri*} [*io_mask*] [*direction*] [*command*] | When a binding exists |

## 5.59.2    Examples

Example of binding PIO5 to close the connection with ID 0 and delete all pairings after PIO5 has fallen.

```
SET CONTROL BIND 0 20 FALL CLOSE 0
SET CONTROL BIND 1 20 FALL SET BT PAIR *
```

## 5.60 SET CONTROL CD

This command enables or disables the carrier detect signal (CD) in iWRAP.

Carrier detect signal can be used to indicate that iWRAP has an active Bluetooth connection. With "**SET CONTROL CD**" command, one PIO line can be configured to act as a CD signal.

### 5.60.1    Syntax

| Synopsis: |
|---|
| **SET CONTROL CD {*cd_mask*} {*datamode*}** |

| Description: | |
|---|---|
| *cd_mask* | This is a bit mask, which defines the GPIO lines used for CD signaling<br><br>For example, value 20 (HEX) must be used for PIO5.<br><br>20 (HEX) = 100000 (BIN)<br><br>For PIO6, the value is 40<br><br>40 (HEX) = 1000000 (BIN) |
| *datamode* | This parameter defines how the carrier detect signal works.<br><br>**0**<br><br>      CD signal is driven high if there are one or more connections.<br><br>**1**<br><br>      CD signal is driven high only in data mode. |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| List format: |
|---|
| **SET CONTROL CD {*cd_mask*} {*datamode*}** |

## 5.61 SET CONTROL CODEC

This command controls the preference of A2DP audio codecs, channel modes and sampling rates for A2DP. In A2DP connections it's always the device establishing the connection, who decides which parameters to use and the device receiving the connection needs to adapt to those parameters despite the configuration set with **SET CONTROL CODEC**.

### 5.61.1    Syntax

| Synopsis |
|---|
| **SET CONTROL CODEC {*codec*} {*channel_mode*} {*sampling_rate*} {*priority*}** |

| Description | |
|---|---|
| *codec* | Which codec to configure. Standard iWRAP supports only SBC. APT-X and FastSream codecs are available upon request. |
| *channel_mode* | Valid channel modes are JOINT_STEREO, STEREO, DUAL_CHANNEL or MONO. |
| *sampling_rate* | Valid sampling rates are 48000 (A2DP sink only), 44100, 32000 and 16000. |
| *priority* | This is used to determine which codec to use in case both the module and the remote end support multiple codecs. Lower priority number means higher preference. |

| Response |
|---|
| None |

| Events |
|---|
| None |

| List format |
|---|
| **SET CONTROL CODEC {*codec*} {*channel_mode*} {*sampling_rate*} {*priority*}** |

Bluegiga Technologies Oy

## 5.61.2    Examples

Configuring the SBC codec for joint stereo 44.1kHz sampling rate and highest priority.

**SET CONTROL CODEC JOINT_STEREO 44100 0**

**Note:**

- Use the EVENT parameter of the SET CONTROL AUDIO configuration command to display a message related to the codec being loaded into the DSP.

## 5.62 SET CONTROL CONFIG

This command enables or disables various functional features in iWRAP. These features are described below.

### 5.62.1 Syntax

| Synopsis |
| --- |
| **SET CONTROL CONFIG *[[[optional_block_2*] *optional_block_1*] *config_block* | LIST]** |

| Description | |
| --- | --- |
| | If no parameters are given, lists the current configuration values. |
| *list* | Same as above, but additionally prints short descriptions of active configuration bits. |
| *config_block* | Hexadecimal number that specifies configuration bits, with bit 0 being the least significant (right hand) bit.<br><br>**Bit 0**<br><br>If this bit is set, the RSSI value will be visible in the inquiry results<br><br>**Bit 1**<br><br>"Bluetooth clock caching". If this bit is set, iWRAP will store the clock states of devices discovered in inquiry. This may speed up connection establishment if the connected device has responded to inquiry.<br><br>**Bit 2**<br><br>"Interlaced inquiry scan". If this bit is set, interlaced inquiry will be used. As a rule, interlaced inquiry is a little bit faster than regular inquiry.<br><br>**Bit 3**<br><br>"Interlaced page scan". If this bit is set, interlaced page (call) will be used. As a rule, interlaced page is a little bit faster than regular page.<br><br>**Bit 4**<br><br>"Deep sleep enabled". If this bit is set, 'Deep sleep' power saving mode will be used. Deep sleep is an aggressive power saving mode used when there are no connections.<br><br>**Bit 5**<br><br>"Bluetooth address in CONNECT". If this bit is set, the Bluetooth address of the remote end will be displayed on the CONNECT event.<br><br>**Bit 6**<br><br>Not used. Must be set to 0. |

Bluegiga Technologies Oy

| | |
|---|---|
| | **Bit 7**<br><br>Displays the **PAIR** event after successful pairing.<br><br>**Bit 8**<br><br>Enables SCO links. This bit must be 1 if you use audio profiles. *Note: this is always set unless bit 2 of optional block 1 is set.*<br><br>**Bit 9**<br><br>Must be set to 0.<br><br>**Bit 10**<br><br>Must be set to 0.<br><br>**Bit 11**<br><br>Enables interactive pairing mode. Where pin code is prompted rather then pin code set with "SET BT AUTH" used.<br><br>**Bit 12**<br><br>If this bit is set iWRAP randomly replaces one of the existing pairings, when the maximum number of pairings (16 unless a lower limit is specified by SET BT PAIRCOUNT) is exceeded. If this bit is not set and the pairing count is exceeded, the pairing will fail.<br><br>**Bit 13**<br><br>If this bit is set CLOCK event will be displayed on CONNECT and RING events.<br><br>**Bit 14**<br><br>If this bit is set UART will be optimized for low latency instead of throughput.<br><br>**Bit 15**<br><br>If this bit is set low inquiry priority is used. This feature reduces inquiry priority and number of inquiry responses but improves simultaneous data transfer performance. |
| ***optional_block_1*** | Hexadecimal number that specifies additional configuration options.<br><br>**Bit 0**<br><br>If this bit is set. All changing iWRAP configuration with SET commands will be disabled. The only way to enable SET commands are by deleting PS-key: "user configuration data 30"<br><br>**Bit 1**<br><br>"Enhanced Inquiry Response (EIR)". If this bit is set, iWRAP will display **INQUIRY_EXTENDED** reports during inquiry. There is a known issue regarding EIR; please see issue #478 in the known issues section.<br><br>**Bit 2**<br><br>Disables automatic setting of config block bit 8. |

Bluegiga Technologies Oy

| | |
|---|---|
| | **Bit 3**<br><br>If this bit is set, iWRAP will always set the link supervision timeout after a Master/Slave switch.<br><br>**Bit 4**<br><br>If this bit is set, iWRAP will display the **CONNAUTH** event before accepting an incoming connection. This allows the user to accept or reject each connection individually.<br><br>**Bit 5**<br><br>If this bit is set, iWRAP will not automatically enter data mode when an RFCOMM connection is opened.<br><br>**Bit 6**<br><br>If this bit is set, iWRAP will display "OK." after each successful command. The message is printed synchronously, e.g. once iWRAP receives the command, no other messages can be printed in between the command's normal output and the "OK." confirmation. Please note that some commands, such as BATTERY and A2DP STREAMING START/STOP, may appear to trigger a synchronous response, but in reality request an event that, while quick to appear, will appear after "OK." is printed.<br><br>**Bit 7**<br><br>If this bit is set, the iWRAP Hands-Free Profile handler will not automatically send an error reply to AT commands it does not understand. This is useful when the user wants to implement their own proprietary commands. Note that the user must implement their own error message sending if this bit is set, since it is mandatory to reply even to unknown commands.<br><br>**Bits 8-15**<br><br>These are temporary configuration bits, for internal use only. They cannot be set or unset. |
| *optional_block_2* | **Bits 0-2**<br><br>These are temporary configuration bits, for internal use only. They cannot be set or unset.<br><br>**Bits 3-15**<br><br>These are reserved for future additions, and cannot be set or unset. |

| **Response** | |
|---|---|
| None | If any configuration blocks are given. |
| **SET CONTROL CONFIG {optional_block_2 optional_block_1 config_block}** | If no parameters are given. |

| **SET CONTROL CONFIG {optional_block_2 optional_block_1 config_block} {descriptions}** | If LIST was issued. |
|---|---|

| **Events** |
|---|
| None |

| List format |
|---|
| None |

## 5.62.2     Examples

Setting optional block 2 to zero (no effect), setting bits 4 and 6 (connection authorization, command confirmation) of optional block 1, setting bits 0 and 5 (inquiry with RSSI and show Bluetooth address in connect events) of config block.

---

**SET CONTROL CONFIG 0000 0050 0021**

**SET CONTROL CONFIG LIST**

SET CONTROL CONFIG 0000 0050 0121 INQUIRY_WITH_RSSI CONN_BD KLUDGE AUTHORISE_REQ PRINT_OK

OK.

---

## 5.63 SET CONTROL ECHO

This command changes the echo mode of iWRAP.

### 5.63.1 Syntax

| Synopsis: |
|---|
| **SET CONTROL ECHO {*echo_mask*}** |

| Description: | |
|---|---|
| ***Echo_mask*** | Bit mask for controlling the display of echo and events |
| | **Bit 0** |
| | If this bit is set, the start-up banner is visible. |
| | **Bit 1** |
| | If this bit is set, characters are echoed back to client in command mode. |
| | **Bit 2** |
| | This bit indicates if set events are displayed in command mode. |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| List format: |
|---|
| **SET CONTROL ECHO  {*echo_mask*}** |

**Warning!**

If every bit is set off (value 0), it is quite impossible to know the iWRAP status.

If Bit 2 is set off, it is very hard to detect whether iWRAP is in command mode or in data mode. This can, however, be solved if one IO is used to indicate that iWRAP is in data mode (**"SET CONTROL CD"**).

# 5.64 SET CONTROL ESCAPE

This command is used to select the escape character used to switch between command and data modes. This command also enables, sets and disables DTR signaling over a selectable GPIO line.

## 5.64.1    Syntax

| Synopsis: |
|---|
| SET CONTROL ESCAPE *{esc_char} {dtr_mask} {dtr_mode}* |

| Description: | |
|---|---|
| *esc_char* | Decimal ASCII value defining the escape character to be used in the escape sequence. Use "-" to disable escape sequence. The default value is 43, which corresponds to "+" |
| *dtr_mask* | Bit mask for selecting the digital I/O pins used for DTR. For example, for I/O 5, the bit mask is **00100000** and **dtr_mask** is then **20** (HEX). |
| *dtr_mode* | **0**<br>DTR Disabled.<br><br>**1**<br>Return to command mode when DTR line transitions from low to high. (It happens, for instance, when pressing the DSR button on the evaluation board, which is linked to pin number 5, after configuring the firmware according to example below.)<br><br>**2**<br>Close the active connection when DTR line transitions from low to high.<br><br>**3**<br>Soft reset iWRAP when DTR line transitions from low to high. |

| Events: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given |

| List format: |
|---|
| SET CONTROL ESCAPE *{esc_char} {dtr_mask} {dtr_mode}* |

## 5.64.2    Example

Disable default escape character "+" and set DTR to GPIO5 for escaping from data to command mode:

Bluegiga Technologies Oy

**SET CONTROL ESCAPE - 20 1**

## 5.65 SET CONTROL GAIN

In WT32, the **SET CONTROL GAIN** command is used to control the internal codec's input and output gain. In WT11, WT12 and WT41, when PCM frame is configured for 13-bit samples with padding in 16-bit slots, this command is meant to control the 3-bit audio attenuation used by some Motorola codecs and other compatible codecs.

### 5.65.1    Syntax

| Synopsis: |
|---|
| SET CONTROL GAIN [{*input*} {*output*} [*DEFAULT*]] |


| Description: | |
|---|---|
|  | If no parameters are given, returns the input and output gain ranges. |
| *input* | Input gain. Range: WT32: 0-16 (hex) , others: must be set to 0 |
| *output* | Output gain. Range: WT32: 0-16 (hex) , others: 0-7 (hex) |
| *DEFAULT* | If given, configures given input and output gain as default values and save them in the persistent store. |


| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |


| Events: |
|---|
| None: |


| List format: |
|---|
| SET CONTROL GAIN {*default input*} {*default output*} |


**Note:**

- When issuing the SET command, **SET CONTROL GAIN** always shows the default input/output gain levels, not the currently active ones.

- On A2DP **source** the input gain should be set to a low value, otherwise the A2DP audio quality will suffer radically.

- Listed below are the different parameter values and their corresponding approximate gains for the WT32.

| Parameter value | Gain (dB) |
|---|---|
| 0 | -24 |
| 1 | -21 |
| 2 | -18 |
| 3 | -15 |
| 4 | -12 |
| 5 | -9 |
| 6 | -6 |
| 7 | -3 |
| 8 | 0 |
| 9 | 3 |
| a | 6 |
| b | 9 |
| c | 12 |
| d | 15 |
| e | 18 |
| f | 21 |
| 10 | 24 |
| 11 | 27 |
| 12 | 30 |
| 13 | 33 |
| 14 | 36 |
| 15 | 39 |
| 16 | 42 |

# 5.66 SET CONTROL INIT

**SET CONTROL INIT** lists or changes the initialization command in iWRAP. This command is run when iWRAP is started or reset.

## 5.66.1 Syntax

| Synopsis: |
| --- |
| **SET CONTROL INIT [*command*]** |

| Description: | |
| --- | --- |
| | If no command is given, will erase the initialization command. |
| ***command*** | Any of the available iWRAP commands. |
| | This command is automatically executed every time iWRAP starts (after power-on, RESET or watchdog event) |

| Events: |
| --- |
| None |

| List format: |
| --- |
| **SET CONTROL INIT {*command*}** |

## 5.66.2 Examples

To remove all pairings after reset:

| |
| --- |
| **SET CONTROL INIT SET BT PAIR *** |

To change baud rate to 115200 bps after reset:

| |
| --- |
| **SET CONTROL INIT SET CONTROL BAUD 115200,8n1** |

**Warning!**

Issuing **SET CONTROL INIT RESET** will cause iWRAP to enter an infinite reset loop, rendering it unusable until the persistent store user key #27 is removed by hand.

## 5.67 SET CONTROL MICBIAS

**SET CONTROL MICBIAS** controls the linear regulator that drives current through the dedicated mic bias pin.

### 5.67.1    Syntax

| Synopsis: |
|---|
| **SET CONTROL MICBIAS [{*voltage*} {*current*}]** |

| Description: | |
|---|---|
| | If no parameters are given, returns current mic bias settings. |
| ***voltage*** | Voltage driven through the mic bias pin. Range 0-F (hex). |
| ***current*** | Current driven through the mic bias pin. Range: 0-F (hex). The setting values and their corresponding typical voltage and current ranges are in the table below. |

| Value | Voltage (V) | Current (mA) |
|---|---|---|
| 0 | 1.61 - 1.80 | 0.237 - 0.394 |
| 1 | 1.65 - 1.86 | 0.296 - 0.492 |
| 2 | 1.71 - 1.93 | 0.354 - 0.589 |
| 3 | 1.76 - 1.98 | 0.412 - 0.687 |
| 4 | 1.84 - 2.06 | 0.471 - 0.785 |
| 5 | 1.89 - 2.14 | 0.530 - 0.883 |
| 6 | 1.97 - 2.23 | 0.589 - 0.980 |
| 7 | 2.04 - 2.32 | 0.647 - 1.078 |
| 8 | 2.18 - 2.46 | 0.706 - 1.176 |
| 9 | 2.27 - 2.58 | 0.764 - 1.273 |
| 10 | 2.39 - 2.72 | 0.823 - 1.371 |
| 11 | 2.50 - 2.87 | 0.882 - 1.469 |
| 12 | 2.70 - 3.09 | 0.940 - 1.566 |
| 13 | 2.85 - 3.29 | 0.998 - 1.664 |
| 14 | 3.07 - 3.56 | 1.057 - 1.762 |
| 15 | 3.28 - 3.84 | 1.116 - 1.859 |

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |

| Events: |
|---|
| None |

| List format: |
|---|
| **SET CONTROL MICBIAS {*voltage*} {*current*}** |

**Note:**

- With WT32 you should not use the mic biasing directly, but only as a digital enable disable signal. The mic bias line suffers from a noise and the recommendation is to use an external mic biasing.

## 5.68 SET CONTROL MUX

**SET CONTROL MUX** can be used to enable or disable the multiplexing mode. This chapter describes the usage of the command as well as the operation of multiplexing mode.

### 5.68.1    Syntax

| Synopsis: |
| --- |
| **SET CONTROL MUX {*mode*}** |

| Description: | |
| --- | --- |
| *mode* | Multiplexing mode<br><br>**0**<br><br>　　　　Multiplexing mode disabled. Normal (data-command) mode enabled<br><br>**1**<br><br>　　　　Multiplexing mode enabled. Multiplexing protocol must be used to talk to iWRAP. |

| Events: | |
| --- | --- |
| **READY** | READY event occurs after a successful mode change. |

| List format: | |
| --- | --- |
|  | Nothing is displayed when multiplexing mode is disabled. |
| **SET CONTROL MUX 1** | This string is displayed when multiplexing mode is enabled. |

### 5.68.2    Examples

To enable multiplexing mode:

| |
| --- |
| **SET CONTROL MUX 1**<br>¿READY. |

To disable multiplexing mode

---

**BF FF 00 11 53 45 54 20 43 4f 4e 54 52 4f 4c 20 4d 55 58 20 30 00**

READY

---

The command is "**SET CONTROL MUX 0**" in the frame format used by MUX mode. The command must be sent in hex format, not in ASCII format.

**Note:**

- When multiplexing mode is enabled, no ASCII commands can be given to iWRAP but the multiplexing protocol must be used. Multiplexing mode can be disabled by deleting PSKEY_USR3 with PSTool.

- ASCII commands do not need to end with"\r" when multiplexing mode is used.

## 5.68.3    Using Multiplexing Mode

The multiplexing protocol format is presented below:

| Length: | Name: | Description: | Value: |
|---|---|---|---|
| 8 bits | SOF | Start of frame | 0xBF |
| 8 bits | LINK | Link ID | 0x00 - 0x08 or 0xFF (control) |
| 6 bits | FLAGS | Frame flags | 0x00 |
| 10 bits | LENGTH | Size of data field in bytes | - |
| 0-1023 Bytes | DATA | Data | - |
| 8 bits | nLINK | {LINK} XOR 0xFF | - |

**Table 10: Multiplexing frame format**

When multiplexing mode is enabled, all the commands and data sent from host to iWRAP must be sent by using the frame format described above instead of plain ASCII commands. Also, the responses and data coming from iWRAP to the host are sent using the same format. iWRAP firmware autonomously processes the frames and decides whether they contain control commands or data which should be forwarded to its destination.

The advantage of multiplexing mode is that there is no need to do special command-data –command mode switching since data and commands are transmitted in the same mode. This saves a lot of time especially in multipoint scenarios, where - in the worst case - switching from data mode to command mode can take more than two seconds.

Also in scenarios where there are several connections, receiving data simultaneously from several devices is difficult if multiplexing mode is not used. In normal (data/command) mode, only one connection can be active (in data mode) at a time, and it can only be used to transmit or receive data. If there is any data received from the other connection during normal mode, the data is stored to small iWRAP buffers and received when the connections become active (data mode of the connection enabled).

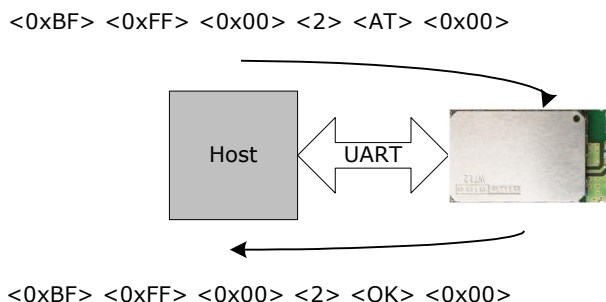The next figure illustrates the host-iWRAP-host communications in multiplexing mode.

<0xBF> <0xFF> <0x00> <2> <AT> <0x00>



<0xBF> <0xFF> <0x00> <2> <OK> <0x00>

**Figure 5: Host-iWRAP-Host communication**

The figure below illustrates host-iWRAP-remote device communication when multiplexing mode is in use. The key thing is that the remote device does not need to know anything about the multiplexing communication and frame format, but it sees the connection as a standard Bluetooth connection.
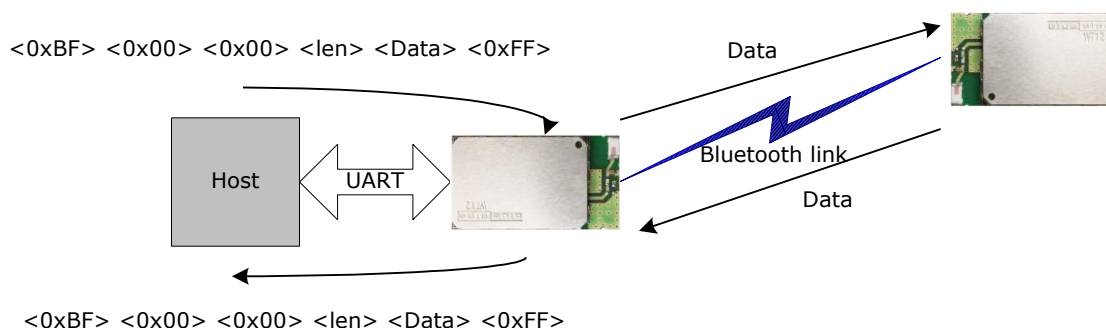
<0xBF> <0x00> <0x00> <len> <Data> <0xFF>



<0xBF> <0x00> <0x00> <len> <Data> <0xFF>

**Figure 6: Host-iWRAP-remote device communications**

At the moment, seven (7) simultaneous connections can be used in multiplexing mode.

**Tips:**

In MUX mode the processor of the module is highly utilized and on the edge of its performance. This may be seen as a instability of Bluetooth connections, especially if 3 or more connections are used or data rate is high. There are however a few tricks how the stability of the Bluetooth connections can be improved:

1. Use SNIFF mode: Using sniff mode reduces the rate the master device needs to poll the active connections are leaves more time for the processor to parse or generate the multiplexing protocol. Therefore as aggressive as possible sniff mode should be used.

2. Optimize Bluetooth packet size by using MTU option in CALL command: Using smaller Bluetooth packet size improves the multiplexing performance.

On the next page, there is a simple C-code example on how to create a simple multiplexing frame containing an iWRAP command.

```
//HOW TO CREATE A SIMPLE FRAME

char outbuf[128];                //Buffer for frame

char* cmd = "SET";              //ASCII command

int link = 0xff, pos=0;          //0xFF for control channel

int len = strlen(cmd);           //Calc. length of ASCII command

//Generate packet

outbuf[pos++]=0xbf;              //SOF

outbuf[pos++]=link;              //Link  (0xFF=Control, 0x00 = connection 1,
etc.)

outbuf[pos++]=0;                 //Flags

outbuf[pos++]=len;               //Length

//Insert data into correct position in the frame

memmove(outbuf+pos cmd, len);

pos += len;                      //Move to correct position

outbuf[pos++]=link^0xff;         //nlink
```

## 5.69 SET CONTROL MSC

With iWRAP firmware, it is possible to transmit all the UART modem signals over the SPP (Serial Port Profile) Bluetooth link. The signals DSR, DTR, RTS, CTS, RI and DCD can be specified to GPIO pins on the WRAP THOR modules. The **SET CONTROL MSC** command is used to do this.

### 5.69.1 Syntax

| Synopsis: |
| --- |
| SET CONTROL MSC [[*mode*] [[*DSR*] [[*DTR*] [[*RTS*] [[*CTS*] [[*RI*] [*DCD*]]]]]]] |

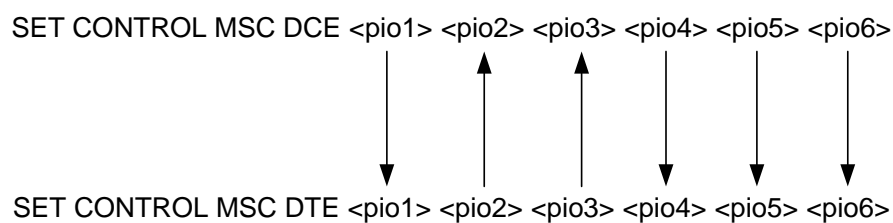| Description: | |
| --- | --- |
| mode | Mode of the device iWRAP connects to. <br><br> The mode can be: <br> **DTE** or **nDTE** <br> and <br><br> **DCE or nDCE** <br><br> **NOTE:** <br><br> DTE means that remote Bluetooth device is DTE (so iWRAP is DCE and device connected to iWRAP is DTE). nDTE and nDCE means that the signals are active low, not active high. |
| *DSR* | Data Set Ready. Select PIO with a bitmask. See the note below on how to select the PIO. |
| *DTR* | Data Terminal Ready. See the note below on how to select the PIO. |
| *RTS* | Request To Send. See the note below on how to select the PIO. |
| *CTS* | Clear To Send. See the note below on how to select the PIO. |
| *RI* | Ring Indicator. See the note below on how to select the PIO. |
| *DCD* | Data Carrier Detect. See the note below on how to select the PIO. |

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |

| Events: |
|---|
| None |

**Note:**

- The PIO pin is selected with a bit mask. For example, if you want to use PIO3, you will then have a bit mask where the third bit is 1, that is, 1000. This bit mask value is then given in the command in hexadecimal format. 1000(bin) = 8(hex).

- If MUX mode is in use physical PIO statuses do not change even if SET CONTROL MSC is used, since in MUX mode it would be hard tell which of the connections defines the MSC signal statuses.

- When the connection is closed the status of MSC signals are not automatically reset, but they are left to the last known state.

SET CONTROL MSC DCE <pio1> <pio2> <pio3> <pio4> <pio5> <pio6>

SET CONTROL MSC DTE <pio1> <pio2> <pio3> <pio4> <pio5> <pio6>

**Figure 7: MSC signal directions**

## 5.70 SET CONTROL PCM

This command configures the physical PCM hardware interface settings and PCM data format.

### 5.70.1    Syntax

| Synopsis: |
| --- |
| **SET CONTROL PCM {*config*} {*data*}** |

| Description: | |
| --- | --- |
| | If no parameters are given lists the current settings. |
| *config* | Value for the PCM interface configuration. Corresponds to PS-key: PSKEY_PCM_CONFIG32 |
| *data* | Value for PCM data format. Corresponds to PS-key: PSKEY_PCM_FORMAT |

| Response: | |
| --- | --- |
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |

| Events: |
| --- |
| None: |

| List format: |
| --- |
| This configuration command is not listed when the SET command is issued |

**Note:**

- Use PCM configuration tool (Excel file) available at http://techforum.bluegiga.com to determine correct value for *config*. You can also find a description of the PSKEY_PCM_CONFIG32 from your modules data sheet and you can calculate the value manually.

- The value for the *data* is calculated according to the Figure 8: HCI Write Voice Setting. Usually it should not be changed, but left to the default value.

| Value | Parameter Description |
|---|---|
| 00XXXXXXXX | Input Coding: Linear |
| 01XXXXXXXX | Input Coding: µ-law Input Coding |
| 10XXXXXXXX | Input Coding: A-law Input Coding |
| 11XXXXXXXX | Reserved for Future Use |
| XX00XXXXXX | Input Data Format: 1's complement |
| XX01XXXXXX | Input Data Format: 2's complement |
| XX10XXXXXX | Input Data Format: Sign-Magnitude |
| XX11XXXXXX | Input Data Format: Unsigned |
| XXXX0XXXXX | Input Sample Size: 8-bit (only for linear PCM) |
| XXXX1XXXXX | Input Sample Size: 16-bit (only for linear PCM) |
| XXXXXnnnXX | Linear_PCM_Bit_Pos: # bit positions that MSB of sample is away from starting at MSB (only for Linear PCM). |
| XXXXXXXX00 | Air Coding Format: CVSD |
| XXXXXXXX01 | Air Coding Format: µ-law |
| XXXXXXXX10 | Air Coding Format: A-law |
| XXXXXXXX11 | Air Coding Format: Transparent Data |

**Figure 8: HCI Write Voice Setting**

## 5.71 SET CONTROL PREAMP

This command enables of disables the 20dB microphone preamplifier on WT32.

### 5.71.1    Syntax

| Synopsis: |
|---|
| **SET CONTROL PREAMP {*left*} {*right*}** |


| Description: | | |
|---|---|---|
| *left* | **1** | |
| | | 20dB preamplifier is enabled for left channel |
| | **0** | |
| | | 20dB preamplifier is disabled for left channel |
| *right* | **1** | |
| | | 20dB preamplifier is enabled for right channel |
| | **0** | |
| | | 20dB preamplifier is disabled for right channel |


| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |


| Events: |
|---|
| None: |


| List format: |
|---|
| **SET CONTROL PREAMP {*left*} {*right*}** |

2.1.2012

## 5.72 SET CONTROL RINTONE

Configures a ring tone which is used with HFP or HSP profile if the Audio Gateway does not provide an in-band ring tone.

### 5.72.1 Syntax

| Synopsis: |
|---|
| **SET CONTROL RINGTONE {*ringtone*}** |

| Description: | |
|---|---|
| | If no parameters are given lists the current settings. |
| ***ringtone*** | Ring tone to play. See the description of PLAY command for more details. |

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |

| Events: |
|---|
| None: |

| List format: | |
|---|---|
| | Nothing is displayed ring tone is not enabled. |
| **SET CONTROL RINGTONE {*ringtone*}** | This string is displayed when ring tone is enabled. |

## 5.73 SET CONTROL READY

This command can be used to dedicate a GPIO pin to indicate that the iWRAP firmware is ready to be used. A typical use case is to indicate a reset condition.

### 5.73.1 Syntax

| Synopsis: |
| --- |
| SET CONTROL READY {*piomask*} |

| Description: | |
| --- | --- |
| *piomask* | A piomask to indicate which GPIOs are used for the signal. Value **0** disables the feature. |

| Response: | |
| --- | --- |
| SYNTAX ERROR | This event occurs if incorrect parameters are given. |

| Events: |
| --- |
| None: |

| List format: | |
| --- | --- |
| SET CONTROL READY {*piomask*} | This string is displayed when ring tone is enabled. |

### 5.73.2 Examples

Using PIO7 to indicate iWRAP ready state.

| SET CONTROL READY 80 |
| --- |

## 5.74 SET CONTROL VREGEN

**SET CONTROL VREGEN** is used to set the behavior of the internal software-controlled regulator on the module. The PIO's specified by the *PIO mask* parameter will be pulled high by the regulator as specified by the *mode* parameter.

### 5.74.1    Syntax

| Synopsis: |
|---|
| **SET CONTROL VREGEN {*mode*} {*PIO mask*}** |

| Description: | |
|---|---|
| *mode* | **0** <br><br> Regulator is enabled on rising edge when its input voltage (VREG_ENA) rises past around 1V and will hold the PIO's high. <br><br> **Warning:** using this mode may or may not, depending on your setup, keep the module powered on until its power source is disconnected or regulator mode is switched! <br><br> **1** <br><br> Regulator is enabled on rising edge (of VREG_ENA) and holds the PIO voltage up until a falling edge is encountered, at which point the regulator will pull the voltage down. <br><br> **2** <br><br> Regulator is enabled on rising edge and holds the voltage up until another rising edge followed by a falling edge is encountered, at which point the regulator will bring the voltage down. |
| *PIO mask* | Bit mask used to specify which PIO's are held up by the regulator. This parameter is given in hexadecimal format. |

| Response: | |
|---|---|
| **SYNTAX ERROR** | This event occurs if incorrect parameters are given. |

| Events: |
|---|
| None |

| List format: |
|---|

Bluegiga Technologies Oy

SET CONTROL VREGEN {*mode*} {*PIO mask*}

## 5.74.2 Examples

Building a setting in which a switch is toggled to power on the module, and keep it powered on until the switch is first toggled back, then toggled up and down again (rising edge followed by falling edge); PIO2 is pulled high to hold up an external regulator

**SET CONTROL VREGEN 2 4** (4 in hexadecimal is 100 in binary)

**Note:**

- Valid for WT32 only

- See the WT32 Design Guide located at http://techforum.bluegiga.com for further information on the design of the module and regulator**.**

## 5.75 SET {link_id} ACTIVE

This command disables all the power save modes for the defined, active Bluetooth link and sets it into active mode.

### 5.75.1　Syntax

| Synopsis: |
| --- |
| SET {*link_id*} ACTIVE |

| Description: | |
| --- | --- |
| *link_id* | Numeric connection identifier |

| Events: |
| --- |
| None |

### 5.75.2　Examples

Changing from SNIFF to active:

**LIST**

**LIST 1**

**LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING SNIFF MASTER PLAIN**

**SET 0 ACTIVE**

**LIST**

LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE MASTER PLAIN

## 5.76 SET {link_id} MASTER

This command attempts to switch the link to Piconet master. Notice that this may not be allowed by the remote end.

### 5.76.1 Syntax

| Synopsis: |
|---|
| **SET {*link_id*} MASTER** |

| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier |

| Events: |
|---|
| None |

### 5.76.2 Examples

Changing from slave to master:

| |
|---|
| **LIST** |
| **LIST 1** |
| **LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE SLAVE PLAIN** |
| **SET 0 MASTER** |
| **LIST** |
| LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE MASTER PLAIN |

## 5.77 SET {link_id} SLAVE

This command attempts to switch the link to Piconet slave. Notice that this may not be allowed by the remote end.

### 5.77.1 Syntax

| Synopsis: |
|---|
| SET {*link_id*} SLAVE |

| Description: | |
|---|---|
| *link_id* | Numeric connection identifier |

| Events: |
|---|
| None |

## 5.78 SET {link_id} SNIFF

This command attempts to enable SNIFF mode for the defined Bluetooth link. Whether this command is successful or not, depends on if the remote end allows sniff to be used.

### 5.78.1    Syntax

| Synopsis: |
| --- |
| SET {*link_id*} SNIFF {max} {min} [{attempt} {timeout}]<br>or<br>SET {*link_id*} SNIFF {avg} |

| Description: | |
| --- | --- |
| *max* | Maximum acceptable interval in milliseconds<br>Range: **0004** to **FFFE**; only even values are valid<br>Time = *max* * 0.625 msec<br>Time range: 2.5 ms to 40959 ms |
| *min* | Minimum acceptable interval in milliseconds<br>Range: **0002** to **FFFE**; only even values, up to *max*, are valid<br>Time = *min* * 0.625 ms<br>Time range: 1.25 ms to 40959 ms |
| *attempt* | Number of Baseband receive slots for sniff attempt.<br>Length = N* 1.25 ms<br>Range for N: **0001 – 7FFF**<br>Time range: 0.625ms - 40959 ms |
| *timeout* | Number of Baseband receive slots for sniff timeout.<br>Length = N * 1.25 ms<br>Range for N: **0000 – 7FFF**<br>Time range: 0 ms - 40959 ms |
| *avg* | Shortcut, sets *max* to 1.5 * *avg*, *min* to 0.67 * *avg*, *attempt* to 1 and *timeout* to 8. |

| Events: |
| --- |
| None |

**Note:**

- Supervisor timeout set with "**SET BT ROLE**" must be longer than maximum acceptable sniff interval.

# 5.79 SET {link_id} SUBRATE

This command attempts to enable sniff subrating mode for the defined Bluetooth link. Whether this command is successful or not, depends on if the remote end allows sniff subrate mode to be used.

Sniff sub-rating (SSR) provides a means to further reduce power consumed by link management. SSR allows either device to increase the time between sniff anchor points. While this change will reduce the responsiveness of the link, it also reduces the number of packets that are exchanged to maintain the link and thus reduces power consumption.

SSR is particularly useful for devices that have periods of activity separated by long periods of inactivity. A computer mouse would be a good example. A user working with a word processor might use the mouse to open a document and position the cursor. Once that is accomplished the user might use only the keyboard for an extended time, never touching the mouse. During these periods of inactivity, the mouse can move into SSR mode and reduce its power consumption.

## 5.79.1      Syntax

| Synopsis: |
|---|
| SET {*link_id*} SUBRATE {*remote_latency*} {*remote_timeout*} {*local_latency*} |


| Description: | |
|---|---|
| ***link_id*** | Numeric connection identifier |
| ***remote_latency*** | The value is specified in units of baseband slots (625 µs). This value is used by the link manager (LM) layer to calculate the value of max_sniff_subrate, which is sent as a parameter of the LMP command used to start sniff subrating. |
| ***remote_timeout*** | The value Is specified in units of baseband slots (625 µs). This value is used by the link manager (LM) layer to determine when to transition a device from sniff mode to sniff sub-rating mode. |
| ***local_latency*** | The value is specified in units of baseband slots (625 µs). This value is used by the link manager (LM) layer to calculate the value of max_sniff_subrate, which is sent as a parameter of the LMP command used to start sniff subrating. |


| Events: |
|---|
| None |

**Note:**

Refer to the Bluetooth specification for more information.

## 5.80 SET {link_id} MSC

With this command, it is possible to send 07.10 Modem Status Command to the remote device without having the signals actually connected to the module. However the status of local GPIO pins will also affect the status of the MSC signals.

### 5.80.1 Syntax

| Synopsis: |
|---|
| SET {*link_id*} MSC {*status*} |

| Description: | |
|---|---|
| *link_id* | Numeric connection identifier of the link where the modem status is to be sent. |
| *status* | Status of the signals according to 07.10 standards. |

| Response: |
|---|
| No response |

### 5.80.2 Examples

Example usage of sending MSC:

| |
|---|
| **SET  0 MSC 8D** |
| ***Normal MSC status was sent.*** |

**Note:**

- Bit 0 can not be modified, it's always set to 1
- Bit 1 can not be changed remotely, only locally
-

## 5.81 SET {link_id} SELECT

With this command, you can define the active Bluetooth connection for the iWRAP command wrapped. This command is useful for example when two simultaneous Hands-Free connections or one Hands-Free connection and one A2DP connection is used

### 5.81.1　　Syntax

| Synopsis: |
| --- |
| **SET {*link_id*} SELECT** |

| Description: | |
| --- | --- |
| ***link_id*** | Numeric connection identifier of the link where the modem status is to be sent. |

| Response: |
| --- |
| No response |

**Note:**

- iWRAP uses an internal command parser/wrapped with some Bluetooth profiles like Hands-Free profile. The internal parser handles commands like HANGUP, VOLUME etc. and transfers them into AT-commands defined in the Hands Free profile specification. To be able to send the commands you need to have correct Bluetooth link / parser selected and this can be done with **SET {link_id} SELECT** command.

## 5.82 SET PROFILE

The **SET PROFILE** command can be used to enable or disable the available Bluetooth profiles: SPP, OPP, HFP and HFP-AG, A2DP, AVRCP and HID.

### 5.82.1 Syntax

| Synopsis: |
|---|
| SET PROFILE {*profile_name*} [*SDP_name*] |

| Description: |
|---|

| *profile_name* | Specify the profile to be enabled or disabled. Possible profile acronyms are: |
|---|---|
| | **SPP** |
| | Serial Port Profile |
| | **HFP** |
| | Hands Free Profile |
| | **HFP-AG** |
| | Hands Free Profile Audio Gateway |
| | **OPP** |
| | Object Push Profile |
| | **A2DP SINK** |
| | Advanced Audio Distribution Profile Sink mode. SDP name can not be changed with A2DP sink. Profile is disabled with "**SET PROFILE A2DP**" |
| | **A2DP SOURCE** |
| | Advanced Audio Distribution Profile Source mode. SDP name can not be changed with A2DP sink. Profile is disabled with "**SET PROFILE A2DP**" |
| | **AVRCP CONTROLLER** |
| | A/V Remote Control Profile controller mode. No *SDP_name* can be given. |
| | **AVRCP TARGET** |
| | A/V Remote Control Profile target mode. No *SDP_name* can be given. |
| | **HID** |
| | HID keyboard and mouse emulation |

Bluegiga Technologies Oy

| | |
|---|---|
| | **HSP**<br><br>    HSP profile in Headset mode. No ***SDP_name*** can be given.<br><br>**HSP-AG**<br><br>    HSP Audio Gateway mode. No ***SDP_name*** can be given.<br><br>**HDP SINK {mdep}**<br><br>    Health Device Profile sink. ***mdep*** defines the IEEE device data type.<br><br>**HDP SOURCE {mdep}**<br><br>    Health Device Profile source. ***mdep*** defines the IEEE device data type.<br><br>**PBAP**<br><br>    Phone Book Access Profile client. No ***SDP_name*** can be given.x<br><br>**BGIO**<br><br>    Bluegiga IO Profile sensor mode<br><br>**OTA**<br><br>    Bluegiga OTA profile |

| | |
|---|---|
| *SDP_name* | **ON** |
| |      Enables the profile with default SDP name. |
| | ***<string>*** |
| |      Enables the profile with ***string*** used as SDP name. Maximum length is 49 characters. Notice that SDP name can ne be set for the following profiles: A2DP, HSP and HID |
| | If this parameter is not given, the profile will be disabled. |

| Response: |
|---|
| No response |

## 5.82.2    Examples

Example of enabling HFP profile.

---

**SET PROFILE HFP My Hands-Free**

**SET**

SET BT BDADDR 00:07:80:80:c2:37

SET BT NAME WT12

SET BT CLASS 001f00

SET BT AUTH * 6666

SET BT LAP 9e8b33

SET BT PAGEMODE 4 2000 1

SET BT ROLE 0 f 7d00

SET BT SNIFF 0 20 1 8

SET CONTROL BAUD 115200,8n1

SET CONTROL CD 80 0

SET CONTROL ECHO 7

SET CONTROL ESCAPE 43 00 1

SET CONTROL MSC DTE 00 00 00 00 00 00

SET PROFILE HFP My Hands-Free

SET PROFILE SPP Bluetooth Serial Port

SET

**RESET**

---

Example of enabling OTA profile. The **password** is needed to connect the OTA profile.

**SET PROFILE OTA** *password*

**SET**

SET BT BDADDR 00:07:80:80:c2:37

SET BT NAME WT12

SET BT CLASS 001f00

SET BT AUTH * 6666

SET BT LAP 9e8b33

SET BT PAGEMODE 4 2000 1

SET BT ROLE 0 f 7d00

SET BT SNIFF 0 20 1 8

SET CONTROL BAUD 115200,8n1

SET CONTROL CD 80 0

SET CONTROL ECHO 7

SET CONTROL ESCAPE 43 00 1

SET CONTROL MSC DTE 00 00 00 00 00 00

SET PROFILE OTA

SET

**RESET**

**Note:**

- iWRAP must be reset for the profile to be activated or deactivated.

- With PBAP profile no SDP name can be given. The only possible *SDP_name* is **ON**.

- If you want to use the HFP or HFP-AG audio profiles, enable also the support for SCO links, by setting "**SET CONTROL CONFIG**" bit 8 to 1. This is "**SET CONTROL CONFIG  100**" if no other configuration bits are enabled. This is only required with iWRAP 2.2.0.

## 5.83 SET RESET

The **SET RESET** command returns the factory settings of the module.

### 5.83.1 Syntax

| Synopsis: |
|---|
| **SET RESET** |

| Description: |
|---|
| None. |

| Response: |
|---|
| iWRAP resets. |

| Events: |
|---|
| None |

**Note:**

- **SET RESET** does not clear the pairings. They must be reset with "**SET BT PAIR \***".

## 5.84 SLEEP

The **SLEEP** command will force deep sleep on. After issuing this command, the module will enter deep sleep until a Bluetooth connection is received or something is received from the UART interface in command mode. The SLEEP command will also work when there are one or more active connections and iWRAP is in command mode and sniff power saving mode is used for the connections.

Deep sleep mode puts the processor into a reduced duty cycle mode.

### 5.84.1    Syntax

| Synopsis: |
|---|
| **SLEEP** |

| Description: |
|---|
| None. |

| Response: |
|---|
| None |

| Events: |
|---|
| None |

**Note:**

- If UART data is used to wake up the module from the deep sleep, the first byte sent to UART is "lost" to wake the module up.

- Refer to power consumption documents for more information about power consumption in deep sleep mode.

- Deep sleep might sometimes be used even if there are active Bluetooth connections. However all the connections need to ne in aggressive sniff power saving mode.

## 5.85 SSP CONFIRM

**SSP CONFIRM** command is used to confirm or cancel SSP requests from other Bluetooth devices.

### 5.85.1 Syntax

| Synopsis |
| --- |
| **SSP CONFIRM** {*bd_addr*} [*OK*] |

| Description | |
| --- | --- |
| *bd_addr* | Bluetooth device address of the device initiating SSP request |
| *OK* | OK flag confirms the SSP request. If left empty the request is denied. |

| Response |
| --- |
| None |

| Events |
| --- |
| None |

## 5.86 SSP PASSKEY

**SSP PASSKEY** command is used to confirm or cancel SSP PASSKEY requests from other Bluetooth devices.

### 5.86.1    Syntax

| Synopsis |
|---|
| **SSP PASSKEY** {*bd_addr*} {*pass_key*} |

| Description | |
|---|---|
| ***bd_addr*** | Bluetooth device address of the device initiating SSP request |
| ***pass_key*** | Common pass key used for authentication |

| Response |
|---|
| None |

| Events |
|---|
| None |

## 5.87 SSP GETOOB

This command can be used to retrieve an Out-of-Band pairing key from iWRAP.

### 5.87.1    Syntax

| Synopsis |
|---|
| **SSP GETOOB** |

| Description |
|---|
| None |

| Response |
|---|
| **SSP SETOOB H:{key1} R:{key2}** |

| | |
|---|---|
| **key1** | 128-bit security key |
| *key2* | 123-bit security key |

| Events |
|---|
| None |

Get OOB-keys from iWRAP

| |
|---|
| **SSP GETOOB**<br>SSP SETOOB H:fc3e453f7f3f6ff0bf226e26385ec538 R:bbca555c64244fe6696c004c9be61ac4 |

## 5.88 SSP SETOOB

This command can be used to set the Out-of-Band pairing key into iWRAP.

### 5.88.1      Syntax

| Synopsis |
|---|
| **SSP SETOOB H:{key1] R:{key2}** |

| Description |
|---|
| None |

| Response |
|---|
| None |

| Events |
|---|
| None |

Set OOB-keys into iWRAP

| |
|---|
| **SSP SETOOB H:fc3e453f7f3f6ff0bf226e26385ec538 R:bbca555c64244fe6696c004c9be61ac4** |

## 5.89 TEMP

This command reads the value of internal temperature sensor. This value should not be considered very reliable. The value can be compensated by modifying PS-key PSKEY_TEMPERATURE_CALIBRATION.

### 5.89.1 Syntax

| Synopsis: |
| --- |
| **TEMP** |

| Description: |
| --- |
| None. |

| Response: | |
| --- | --- |
| **TEMP {*temp*}** | |
| *temp* | Temperature in Celsius |

| Events: |
| --- |
| None. |

### 5.89.2 Examples

Reading the value of internal temperature sensor.

| **TEMP** |
| --- |
| TEMP 31 |

**Note:**

- The refresh rate of the temperature sensor is not very high.

## 5.90 TEST

The **TEST** command is used to give radio test commands to iWRAP. The commands are the same that can be given by using CSR BlueTest software. TEST commands must only be used for testing purposes, not for application functionality.

### 5.90.1    Syntax

| Synopsis: |
| --- |
| **TEST {*mode*} [*mode_specific_parameters*]** |

| Description: | |
| --- | --- |
| ***mode &***<br><br>***mode_specific_parameters*** | RF Test mode<br><br>Supported test modes are:<br><br>**PAUSE**<br><br>    Pause halts the current test and stops any radio activity.<br><br>**TXSTART {lo_freq} {level} {mod_freq}**<br><br>    Enables the transmitter in continuous transmission at a designated frequency (**lo_freq**) with a designated output power (**level**) and designated tone modulation frequency (**mod_freq**).<br><br>    **lo_freq:** range 2402 – 2480 (MHz)<br><br>    **level**: 0xYYZZ where YY corresponds to Radio Power Table's "Basic Ext PA" and can range from 00 to FF (that is, 0 to 255 in decimal, as seen in the Radio Power Table's view of PSTool) while ZZ corresponds to "Basic Int PA" and can range from 00 to 3F (0 to 63)<br><br>    **mod_freq:** range 0 – 32767 (recommended values 0 or 256)<br><br>**TXDATA1 {lo_freq} {level}**<br><br>    Enables the transmitter with a designated frequency (**lo_freq**) and output power (**level)**. Payload is PRBS9 data. In this mode, the receiver is not operating.<br><br>**TXDATA2 {cc} {level}**<br><br>    Enables the transmitter with a simplified hop sequence designated by country code **{cc}** and output power **{level}**. Payload is PRBS9 data. In this mode, the receiver is not operating.<br><br>    Related test spec name: **TRM/CA/01/C** (output power), **TRM/CA/02/C** (power density).<br><br>    **cc** range: 0 – 3 (default = 0)<br><br>**RXSTART {lo_freq} {highside} {attn}**<br><br>    Enables the receiver in continuous reception at a |

Bluegiga Technologies Oy

designated frequency (**lo_freq**) with a choice of low or high side modulation (**highside**) and with designated attenuation setting (**attn**).

**highside** range: 0 or 1 (default = false = 0)

**attn**: range: 0 – 15

## DEEPSLEEP

Puts the module into deep-sleep after a delay of half a second until woken by a reset or activity on UART.

## PCMLB {pcm_mode}

Sets the PCM to loop back mode, where the data read from PCM input is output again on the PCM output.

If **pcm_mode** = 0, module is slave in normal 4-wire configuration

If **pcm_mode** = 1, module is master in normal 4-wire configuration

If **pcm_mode** = 2, module is master in Manchester encoded 2-wire configuration

## PCMEXTLB {pcm_mode}

Sets the PCM to external loop back mode, whereby the data written to PCM output is read again on the input. Check is made that the data read back is the same as that written.

The external loop back may be a simple wire.

Modes are save as above.

## LOOPBACK {lo_freq} {level}

Receives data on set frequency **lo_freq** for data packets and then retransmits this data on the same channel at output power **level**.

## CFGXTALFTRIM {xtal_ftrim}

This command can be used to set the crystal frequency trim value directly from iWRAP. This is not a permanent setting!

**xtal_ftrim** range: 0 – 63

## PCMTONE {freq} {ampl} {dc}

Plays a constant tone on the PCM port.

**freq** range: 0 – 5

**ampl** range : 0-8

**dc**: 0 – 60096 (set to 0)

## SETPIO {mask} {bits}

Sets PIO high or low according to given parameters.

NOTE: This command sets the PIO regardless of other usage!

**mask**: Bit mask specifying the PIOs that are to be set

**bits**: the bit values

| | If you use hexadecimals, put 0x in front of the value, otherwise they are interpreted as decimals. |
| --- | --- |
| | **GETPIO** |
| | Gets the status of all the PIO lines. |
| | **AUDIOLOOPBACK** |
| | On WT32 loops the audio via the built-in codec. |

| Response: |
| --- |
| **OK** for successful execution |
| **ERROR** for unsuccessful execution |

## 5.90.2   Examples

| | |
| --- | --- |
| **TEST TXSTART 2441 0xFF3F 0**<br>OK | (Enables      carrier      wave      @      2441Mhz) |
| **TEST PCMTONE 1 5 0**<br>OK | (Enables      PCM      tone      signal) |

**Note:**

- Always consult Bluegiga Technologies about the right parameters for RF testing. The TX power parameters are unique for each module: WT11, WT12 and WT32.

- If **TEST** command is used a reset should be made before returning to normal operation.

## 5.91 TESTMODE

The **TESTMODE** command is used to put the iWRAP into a Bluetooth test mode, where a Bluetooth tester can control the hardware. Reset must be done to recover normal operation.

### 5.91.1    Syntax

| Synopsis: |
|---|
| **TESTMODE** |

| Description: |
|---|
| No description. |

| Response: |
|---|
| **TEST 0** |

| Events: |
|---|
| None |

## 5.92 TXPOWER

The **TXPOWER** command can be used check the TX output power level of an active Bluetooth link.

### 5.92.1 Syntax

| Synopsis: |
| --- |
| **TXPOWER {*link_id*}** |

| Description: | |
| --- | --- |
| ***link_id*** | Numeric connection identifier |

| Response: | |
| --- | --- |
| **TXPOWER {*bd_addr*} {*txpower*}** | |
| ***bd_addr*** | Bluetooth address of the remote device |
| ***txpower*** | User TX power level in dBm |

| Events: |
| --- |
| None |

### 5.92.2 Examples

Checking the TX power level of an active connection:

| |
| --- |
| **LIST**<br>LIST 1<br>LIST 0 CONNECTED RFCOMM 320 0 0 3 8d 8d 00:60:57:a6:56:49 1 OUTGOING ACTIVE MASTER PLAIN<br><br>**TXPOWER 0**<br><br>**TXPOWER 00:60:57:a6:56:49 3**          **(TX power level is 3 dBm)** |

## 5.93 PBAP

**PBAP** command is used to retrieve phone book entries or call history from a PBAP PSE device.

### 5.93.1 Syntax

| Synopsis |
| --- |
| PBAP {*path*} {*count*} [*offset*] [*filter*] [*format*] |

| Description | | |
| --- | --- | --- |
| *path* | **Left HEX**<br><br>Store to retrieve data from.<br><br>**0**<br><br>Phone<br><br>**1**<br><br>SIM card | **Right HEX**<br><br>Phone book or call history to read.<br><br>**0**<br><br>Phonebook<br><br>**1**<br><br>Incoming call history<br><br>**2**<br><br>Outgoing call history<br><br>**3**<br><br>Missed call history<br><br>**4**<br><br>Combined call history |
| *count* | Number of entries to be retrieved.<br><br>**0**<br><br>Returns phone book size<br><br>**FFFF**<br><br>Retrieves all entries | |
| *offset* | Offsets from which to start the retrieve from. | |
| *filter* | This is a bit mask to filter the response. If this is left to 0 all fields will be returned.<br><br>Mandatory attributes for vCard 2.1 are VERSION ,N and TEL and they are returned always.<br><br>Mandatory attributes for vCard 3.0 are VERSION, N, FN and TEL  and they are also returned always.<br><br>**bit 0** | |

| | | VERSION  vCard Version |
| --- | --- | --- |
| | **Bit 1** | |
| | | FN  Formatted pbap Name |
| | **bit 2** | |
| | | N  Structured Presentation of Name |
| | **bit 3** | |
| | | PHOTO  Associated Image or Photo |
| | **bit 4** | |
| | | BDAY  Birthday |
| | **bit 5** | |
| | | ADR  Delivery Address |
| | **bit 6** | |
| | | LABEL  Delivery |
| | **bit 7** | |
| | | TEL  Telephone Number |
| | **bit 8** | |
| | | EMAIL  Electronic Mail Address |
| | **bit 9** | |
| | | MAILER  Electronic Mail |
| | **bit 10** | |
| | | TZ  Time Zone |
| | **bit 11** | |
| | | GEO  Geographic Position |
| | **bit 12** | |
| | | TITLE  Job |
| | **bit 13** | |
| | | ROLE  Role within the Organization |
| | **bit 14** | |
| | | LOGO  Organization Logo |
| | **bit 15** | |
| | | AGENT  vCard of Person Representing |
| | **bit 16** | |
| | | ORG  Name of Organization |

| | | |
|---|---|---|
| | **bit 17** | |
| | | NOTE  Comments |
| | **bit 18** | |
| | | REV  Revision |
| | **bit 19** | |
| | | SOUND  Pronunciation of Name |
| | **bit 20** | |
| | | URL  Uniform Resource Locator |
| | **bit 21** | |
| | | UID  Unique ID |
| | **bit 22** | |
| | | KEY  Public Encryption Key |
| | **bit 23** | |
| | | NICKNAME  Nickname |
| | **bit 24** | |
| | | CATEGORIES  Categories |
| | **bit 25** | |
| | | PROID  Product ID |
| | **bit 26** | |
| | | CLASS  Class information |
| | **bit 27** | |
| | | SORT-STRING  String used for sorting operations |
| | **bit 28** | |
| | | X-IRMC-CALL-DATETIME  Time stamp |
| format | **1** | |
| | | Return vcard 3.0 |
| | **0** | |
| | | Return vcard 2.1 |

| Response |
|---|
| **{OBEX header} {vCARD}** |

| **OBEX header** | OBEX header. See the header descriptions below. |
|---|---|
| ***vCARD*** | vCARD data |

| Length | Name | Description | Value |
|---|---|---|---|
| 8 bits | Begin | Start of OBEX frame. | 0xFC (last frame) <br> or 0xFB (more frames to follow) |
| 8 bits | Length | Length of full frame | 0x00 – 0xFF |
| 8 bits | Length | Length of full frame | 0x00 – 0xFF |
| 8 bits | Body | Body or end-of-body. | 0x49 (last frame) <br> 0x48 (more frames to follow) |
| 8 bits | Length of data | Length of data field | 0x00 – 0xFF |
| 8 bits | Length of data | Length of data field | 0x00 – 0xFF |

**Table 11: OBEX header**

| Events |
|---|
| **OBEX AUTH** | This event occurs if the server requires authentication |

## 5.93.2    Examples

This example will retrieve the first 2 phone book entries from the phone memory.

NOTE: OBEX frame in brackets.

---

**PBAP 00 2**

**{0xFC 0x00 0xD8 0x4} 0x01 0xD2} BEGIN:VCARD**

VERSION:2.1

N:

TEL:

END:VCARD

BEGIN:VCARD

VERSION:2.1

REV:20090209T230141Z

UID:33e2c83138e30149-00e1403769bb27b9-389

N:Emergency;Number;;;

X-CLASS:private

TEL;VOICE;WORK:+112

END:VCARD

---

This example read first six placed calls from the phone.

NOTE: OBEX frames bold and in brackets.

---

**PBAP 02 6**

**{0xFB 0x01 0xFF 0x48 0x01 0xFC}BEGIN:VCARD**

VERSION:2.1

N:

TEL:+1234567896

X-IRMC-CALL-DATETIME;DIALED:20090414T082710Z

END:VCARD

BEGIN:VCARD

VERSION:2.1

N:

TEL:+1234567895

X-IRMC-CALL-DATETIME;DIALED:20090414T082707Z

END:VCARD

BEGIN:VCARD

VERSION:2.1

---

N:

TEL:+1234567894

X-IRMC-CALL-DATETIME;DIALED:20090414T082704Z

END:VCARD

BEGIN:VCARD

VERSION:2.1

N:

TEL:+1234567893

X-IRMC-CALL-DATETIME;DIALED:20090414T082623Z

END:VCARD

BEGIN:VCARD

VERSION:2.1

N:

TEL:+1234567892

X-IRMC-CALL-DATETIME;DIALED:20090414T08262**{0xFC 0x00 0x7D 0x49 0x00 0x7A}**0Z

END:VCARD

BEGIN:VCARD

VERSION:2.1

N:

TEL:+1234567891

X-IRMC-CALL-DATETIME;DIALED:20090414T082613Z

END:VCARD

## 5.94 VOLUME

Command **VOLUME** is used to modify and read the module's line out volume level. The command also reports the volume level to HFP-AG in case HFP connection is active.

### 5.94.1 Syntax

| Synopsis: |
|---|
| **VOLUME [{*vol*}]** |


| Description: | |
|---|---|
| *vol* | New volume level value; leave blank to read current volume level. <br><br> **0…15** <br><br>          Sets volume level: Range 0-15. In iWRAP 3.0.0 the range is 0-9. <br><br> **down** <br><br>          Decreases volume level by one. <br><br> **up** <br> Increases volume level by one. |


| Response: |
|---|
| None |


| Events: | |
|---|---|
| **VOLUME {vol}** | Current volume level. |

# 6  iWRAP Events

Events are a mechanism that iWRAP uses to notify the user for completed commands, incoming connections etc.

**Note:**

- If iWRAP is in data mode (data is being transmitted and no multiplexing mode is used) the only possible event is <u>**NO CARRIER**</u> indicating that connection was closed or lost.

- iWRAP is designed so that unwanted events can be safely ignored. Events <u>**CONNECT**</u>, <u>**NO CARRIER**</u> and <u>**RING**</u> change the mode of operation and therefore they cannot be ignored.

- Events can be masked away by removing **Bit 2** from command **SET CONTROL ECHO**.

## 6.1 AUTH

__AUTH__ event indicates that someone is trying to pair with iWRAP.

### 6.1.1 Syntax

| Synopsis: |
|---|
| **AUTH {*bd_addr*}?** |

| Description: | |
|---|---|
| ***bd_addr*** | Bluetooth device address of the remote device |

**Note:**

- The __AUTH__ event occurs only if interactive pairing is enabled with "**SET CONTROL CONFIG**" command.

Bluegiga Technologies Oy

## 6.2 BATTERY

The **<u>BATTERY</u>** event is used to report the current battery voltage to the user.

### 6.2.1 Syntax

| Synopsis: |
|---|
| **BATTERY {*mv*}** |

| Description: | |
|---|---|
| *mv* | Current battery voltage in millivolts. |

## 6.3  CONNECT

The **CONNECT** event is used to notify the user for a successful link establishment.

### 6.3.1  Syntax

| Synopsis: |
| --- |
| **CONNECT {*link_id*} {SCO \| RFCOMM \| A2DP \| HID \| HFP \| HFP-AG {*target*} [*address*]}** |

| Description: | |
| --- | --- |
| *link_id* | Numeric connection identifier |
| *target* | Connected RFCOMM channel number or L2CAP psm |
| *address* | Address of the remote end. This is displayed only if bit 5 is set in "**SET CONTROL CONFIG**". |

**Note:**

iWRAP automatically enters data mode after the **CONNECT** event if multiplexing mode is disabled.

## 6.4 CONNAUTH

The **CONNAUTH** event indicates an incoming Bluetooth connection, which needs to be authorized with the **CONNAUTH** command.

### 6.4.1 Syntax

| Synopsis: |
| --- |
| **CONNAUTH {*bd_addr*} {*protocol_id* } {*channel_id*}** |

| Description: | |
| --- | --- |
| ***bd_addr*** | Bluetooth device address of the remote device |
| ***protocol_id*** | Protocol ID of the incoming connection<br>**0**<br>RFCOMM<br>**1**<br>TBA<br>**2**<br>TBA |
| ***channel_id*** | Channel number of the incoming connection |

| Response: |
| --- |
| None |

| Events: |
| --- |
| None. |

**Note**:

- By default the connections are autokorized automatically. **CONNAUTH** event needs to be separately enabled with **SET CONTROL CONFIG** command.

Bluegiga Technologies Oy

## 6.5  CLOCK

**CLOCK** event indicates the Piconet clock value for a specific Bluetooth connection.

### 6.5.1  Syntax

| Synopsis: |
|---|
| **CLOCK {*bd_addr*} {*clock*}** |

| Description: | |
|---|---|
| ***bd_addr*** | Bluetooth device address of the remote device |
| ***clock*** | Piconet clock value |

All the devices in a Bluetooth Piconet are synchronized to a same clock (master clock). The CLOCK event displays the clock value and it can for example be used for time synchronization of the Piconet slaves and master. The accuracy of the Piconet clock is 625us.

## 6.6  IDENT

**IDENT** event informs that a remote Bluetooth device has been identified by using the Device ID profile and reports the identification data sent by the remote device.

### 6.6.1  Syntax

| Synopsis: |
| --- |
| **IDENT {*src*}:{*vendor_id*} {*product_id*} {*version*} "[*descr*]"** |


| Description: | |
| --- | --- |
| *src* | This attribute indicates which organization assigned the VendorID attribute. There are two possible values: BT for the Bluetooth Special Interest Group (SIG) or USB for the USB Implementer's Forum. |
| *vendor_id* | Intended to uniquely identify the vendor of the device. The Bluetooth SIG or the USB IF assigns VendorIDs. Bluegiga's VendorID is 47. |
| *product_id* | Intended to distinguish between different products made by the vendor in question. These IDs are managed by the vendors themselves, and should be changed when new features are added to the device. |
| *version* | Vendor-assigned version string indicating device version number. |
| *descr* | Optional freeform product description string. |

## 6.7 IDENT ERROR

**IDENT ERROR** event informs that a remote Bluetooth could not be identified by the Device ID profile.

## 6.7.1 Syntax

| Synopsis: |
|---|
| **IDENT ERROR {*error_code*} {*address*} [*message*]** |

| Description: | |
|---|---|
| *error_code* | Code describing the error |
| *address* | Bluetooth address of the device |
| *message* | Optional verbose error message |

## 6.8  INQUIRY_PARTIAL

The **INQUIRY_PARTIAL** event is used to notify the user for a found Bluetooth device. This event precedes response for the **INQUIRY** command.

### 6.8.1  Syntax

| Synopsis: |
| --- |
| **INQUIRY_PARTIAL {*address*} {*class_of_device*} [{*cahced_name*} {*rssi*}]** |

| Description: | |
| --- | --- |
| ***address*** | Bluetooth address of the found device |
| ***class_of_device*** | C Bluetooth Class of Device of the found device |
| ***cached_name*** | User friendly name of the found device if already known |
| ***rssi\**** | Received Signal Strength of the found device |

*) RSSI is a value between -128 and 0. The lower the value, the lower the signal strength.

**Note:**

***cached_name*** **and** *rssi* **are only visible if "Inquiry with RSSI" is enabled with "SET CONTROL CONFIG".**

## 6.9 NO CARRIER

The **NO CARRIER** event is used to notify the user for a link loss or, alternatively, a failure in the link establishment.

### 6.9.1 Syntax

| Synopsis: |
|---|
| **NO CARRIER {*link_id*} ERROR {*error_code*} [*message*]** |


| Description: | |
|---|---|
| *link_id* | Numeric connection identifier |
| *error_code* | Code describing the error |
| *message* | Optional verbose error message |

## 6.10 NAME

The **NAME** event is used to notify the user for a successful lookup for Bluetooth friendly name of the remote device.

### 6.10.1 Syntax

| Synopsis: |
|---|
| **NAME {*address*} {"*friendly_name*"}** |

| Description: | |
|---|---|
| ***address*** | Bluetooth device address of the device |
| ***friendly_name*** | Friendly name of the device |

## 6.11 NAME ERROR

The **NAME ERROR** event is used to notify the user for a Bluetooth friendly name lookup failure.

### 6.11.1    Syntax

| Synopsis: |
|---|
| **NAME ERROR {*error_code*} {*address*} [*message*]** |

| Description: | |
|---|---|
| *error_code* | Code describing the error |
| *address* | Bluetooth address of the device |
| *message* | Optional verbose error message |

## 6.12 OBEX AUTH

The **OBEX AUTH** event is used to notify that the PBAP server device requires authentication.

### 6.12.1 Syntax

| Synopsis |  |
|---|---|
| **OBEX AUTH [USERID:<*userid*> [READONLY]] [REALM:<*realm*>]?** | |
| ***userid*** | TBD |
| ***realm*** | TBD |

| Description |
|---|
| TBD <br> UserID must be given in authentication response <br> READONLY: only readonly access allowed <br> realm: authentication realm <br> first byte is encoding: <br>        0 ASCII <br>        1 ISO-8859-1 <br>        2 ISO-8859-2 <br>        3 ISO-8859-3 <br>        4 ISO-8859-4 <br>        5 ISO-8859-5 <br>        6 ISO-8859-6 <br>        7 ISO-8859-7 <br>        8 ISO-8859-8 <br>        9 ISO-8859-9 <br> 0xFF = 255 UNICODE <br> OBEX AUTH <pincode> |

Bluegiga Technologies Oy

## 6.13 PAIR

The **PAIR** event is used to notify the user for a successful pairing.

### 6.13.1    Syntax

| Synopsis: |
|---|
| **PAIR {*address*} {*key_type*} {*link_key*}** |

| Description: | |
|---|---|
| *address* | Bluetooth device address of the paired device |
| *key_type* | Type of link key<br><br>**0**<br><br>        Combination key<br><br>**1**<br><br>        Local unit key<br><br>**2**<br><br>        Remote unit key<br><br>**3**<br><br>        Debug combination key<br><br>**4**<br><br>        Unauthenticated combination key<br><br>**5**<br><br>        Authenticated combination key<br><br>**5**<br><br>        Changed combination key<br><br>**0x07-0xFF**<br><br>        Reserved |
| *link_key* | Link key shared between the local and the paired device |

**Note:**

- The **PAIR** event is enabled or disabled with the "**SET CONTROL CONFIG**" command.
- If the **PAIR** event is enabled the event will also be shown during the **CALL** procedure and also before the **RING** event, if pairing occurs.

## 6.14 READY

The **READY** event is used to notify the user for switching to command mode or to indicate that iWRAP is ready to be used after a reset or after a successful switch between normal or multiplexing mode has been done.

### 6.14.1    Syntax

| Synopsis: |
|---|
| READY. |

| Description: |
|---|
| None |

## 6.15 RING

The **RING** event is used to notify the user for an incoming connection.

## 6.15.1    Syntax

| Synopsis: |
|---|
| RING {*link_id*} {*address*} {SCO | {*channel*} {profile}} |


| Description: | |
|---|---|
| *link_id* | Numeric connection identifier |
| *address* | Bluetooth device address of the device |
| *channel* | Local RFCOMM channel, L2CAP psm or SCO channel |
| *profile* | Profile or protocol indicator. Indicates the profile or protocol type. For example: RFCOMM or L2CAP HFP, HSP, A2DP, AVRCP, OBEX etc. |

## 6.16 SSP COMPLETE

The **SSP COMPLETE** event is used to notify the completion of SSP pairing.

### 6.16.1 Syntax

| Synopsis: |
|---|
| **SSP COMPLETE** |

| Description: |
|---|
|  |
|  |
|  |
|  |

# 7 iWRAP Error Messages

This chapter briefly presents the iWRAP error messages.

## 7.1 HCI Errors

HCI errors start with code:    **0x100**

| ERROR MESSAGE | CODE | Explanation |
|---|---|---|
| HCI_SUCCESS | 0x00 | Success |
| HCI_ERROR_ILLEGAL_COMMAND | 0x01 | Unknown HCI command |
| HCI_ERROR_NO_CONNECTION | 0x02 | Unknown connection identifier |
| HCI_ERROR_HARDWARE_FAIL | 0x03 | Hardware Failure |
| HCI_ERROR_PAGE_TIMEOUT | 0x04 | Page timeout |
| HCI_ERROR_AUTH_FAIL | 0x05 | Authentication failure |
| HCI_ERROR_KEY_MISSING | 0x06 | PIN or key missing |
| HCI_ERROR_MEMORY_FULL | 0x07 | Memory capacity exceeded |
| HCI_ERROR_CONN_TIMEOUT | 0x08 | Connection timeout |
| HCI_ERROR_MAX_NR_OF_CONNS | 0x09 | Connection Limit Exceeded |
| HCI_ERROR_MAX_NR_OF_SCO | 0x0a | Synchronous connection limit to a device exceeded |
| HCI_ERROR_MAX_NR_OF_ACL | 0x0b | ACL Connection Already Exists |
| HCI_ERROR_COMMAND_DISALLOWED | 0x0c | Command Disallowed |
| HCI_ERROR_REJ_BY_REMOTE_NO_RES | 0x0d | Connection Rejected due to Limited Resources |
| HCI_ERROR_REJ_BY_REMOTE_SEC | 0x0e | Connection Rejected Due To Security Reasons |
| HCI_ERROR_REJ_BY_REMOTE_PERS | 0x0f | Connection Rejected due to Unacceptable BD_ADDR |
| HCI_ERROR_HOST_TIMEOUT | 0x10 | Connection Accept Timeout Exceeded |
| HCI_ERROR_UNSUPPORTED_FEATURE | 0x11 | Unsupported Feature or Parameter Value |
| HCI_ERROR_ILLEGAL_FORMAT | 0x12 | Invalid HCI Command Parameter |
| HCI_ERROR_OETC_USER | 0x13 | Remote User Terminated Connection |
| HCI_ERROR_OETC_LOW_RESOURCE | 0x14 | Remote Device Terminated Connection due to Low Resources |

| HCI_ERROR_OETC_POWERING_OFF | 0x15 | Remote Device Terminated Connection due to Power Off |
|---|---|---|
| HCI_ERROR_CONN_TERM_LOCAL_HOST | 0x16 | Connection Terminated By Local Host |
| HCI_ERROR_AUTH_REPEATED | 0x17 | Repeated Attempts |
| HCI_ERROR_PAIRING_NOT_ALLOWED | 0x18 | Pairing Not Allowed |
| HCI_ERROR_UNKNOWN_LMP_PDU | 0x19 | Unknown LMP PDU |
| HCI_ERROR_UNSUPPORTED_REM_FEATURE | 0x1a | Unsupported Remote Feature / Unsupported LMP Feature |
| HCI_ERROR_SCO_OFFSET_REJECTED | 0x1b | SCO Offset Rejected |
| HCI_ERROR_SCO_INTERVAL_REJECTED | 0x1c | SCO Interval Rejected |
| HCI_ERROR_SCO_AIR_MODE_REJECTED | 0x1d | SCO Air Mode Rejected |
| HCI_ERROR_INVALID_LMP_PARAMETERS | 0x1e | Invalid LMP Parameters |
| HCI_ERROR_UNSPECIFIED | 0x1f | Unspecified Error |
| HCI_ERROR_UNSUPP_LMP_PARAM | 0x20 | Unsupported LMP Parameter Value |
| HCI_ERROR_ROLE_CHANGE_NOT_ALLOWED | 0x21 | Role Change Not Allowed |
| HCI_ERROR_LMP_RESPONSE_TIMEOUT | 0x22 | LMP Response Timeout |
| HCI_ERROR_LMP_TRANSACTION_COLLISION | 0x23 | LMP Error Transaction Collision |
| HCI_ERROR_LMP_PDU_NOT_ALLOWED | 0x24 | LMP PDU Not Allowed |
| HCI_ERROR_ENC_MODE_NOT_ACCEPTABLE | 0x25 | Encryption Mode Not Acceptable |
| HCI_ERROR_UNIT_KEY_USED | 0x26 | Link Key Can Not be Changed |
| HCI_ERROR_QOS_NOT_SUPPORTED | 0x27 | Requested QoS Not Supported |
| HCI_ERROR_INSTANT_PASSED | 0x28 | Instant Passed |
| HCI_ERROR_PAIR_UNIT_KEY_NO_SUPPORT | 0x29 | Pairing With Unit Key Not Supported |

**Table 12: HCI errors**

Please see Bluetooth 2.0+EDR core specification page 493 for more information about error codes.

## 7.2  SDP Errors

SDP errors start with code:                    **0x300**

| ERROR MESSAGE | CODE | Explanation |
|---|---|---|
| SDC_OK | 0x00 | - |
| SDC_OPEN_SEARCH_BUSY | 0x01 | SDP search is busy |
| SDC_OPEN_SEARCH_FAILED | 0x02 | SDP search failed |
| SDC_OPEN_SEARCH_OPEN | 0x03 | - |
| SDC_OPEN_DISCONNECTED | 0x04 | - |
| SDC_OPEN_SEARCH_FAILED_PAGE_TIMEOUT | 0x05 | SDP search failed due page timeout |
| SDC_OPEN_SEARCH_FAILED_REJ_PS | 0x06 | - |
| SDC_OPEN_SEARCH_FAILED_REJ_SECURITY | 0x07 | SDP search failed because of security |
| SDC_OPEN_SEARCH_FAILED_REJ_RESOURCES | 0x08 | SDP search failed because of insufficient resources |
| SDC_OPEN_SEARCH_FAILED_SIGNAL_TIMEOUT | 0x09 | - |
| SDC_ERROR_RESPONSE_PDU | 0x10 | - |
| SDC_NO_RESPONSE_DATA | 0x11 | Empty response - no results |
| SDC_CON_DISCONNECTED | 0x12 | Remote device disconnected |
| SDC_CONNECTION_ERROR | 0x13 | Remote device refused connection |
| SDC_CONFIGURE_ERROR | 0x14 | L2CAP config failed |
| SDC_SEARCH_DATA_ERROR | 0x15 | Search data is invalid |
| SDC_DATA_CFM_ERROR | 0x16 | Failed to transmit PDU |
| SDC_SEARCH_BUSY | 0x17 | Search is busy |
| SDC_RESPONSE_PDU_HEADER_ERROR | 0x18 | The response had a header error |
| SDC_RESPONSE_PDU_SIZE_ERROR | 0x19 | The response had a size error |
| SDC_RESPONSE_TIMEOUT_ERROR | 0x1a | The response has timed out |
| SDC_SEARCH_SIZE_TOO_BIG | 0x1b | The size of the search will not fit into the L2CAP packet |
| SDC_RESPONSE_OUT_OF_MEMORY | 0x1c | |
| SDC_RESPONSE_TERMINATED | 0x1d | |

| SDC_OPEN_SEARCH_FAILED_PAGE_TIMEOUT | 305 | SDP search failed because of page timeout |
| --- | --- | --- |
| SDC_OPEN_SEARCH_FAILED_REJ_TIMEOUT | 305 | SDP search failed because of page timeout |

**Table 13: SDP errors**

## 7.3 RFCOMM Errors

RFCOMM errors start with code: **0x400**

| ERROR MESSAGE | CODE | Explanation |
|---|---|---|
| RFC_OK | 0x00 | |
| RFC_CONNECTION_PENDING | 0x01 | |
| RFC_CONNECTION_REJ_PSM | 0x02 | |
| RFC_CONNECTION_REJ_SECURITY | 0x03 | |
| RFC_CONNECTION_REJ_RESOURCES | 0x04 | |
| RFC_CONNECTION_REJ_NOT_READY | 0x05 | |
| RFC_CONNECTION_FAILED | 0x06 | |
| RFC_CONNECTION_TIMEOUT | 0x07 | |
| RFC_NORMAL_DISCONNECT | 0x08 | |
| RFC_ABNORMAL_DISCONNECT | 0x09 | |
| RFC_CONFIG_UNACCEPTABLE | 0x0a | |
| RFC_CONFIG_REJECTED | 0x0b | |
| RFC_CONFIG_INVALID_CID | 0x0c | |
| RFC_CONFIG_UNKNOWN | 0x0d | |
| RFC_CONFIG_REJECTED_LOCALLY | 0x0e | |
| RFC_CONFIG_TIMEOUT | 0x0f | |
| RFC_REMOTE_REFUSAL | 0x11 | |
| RFC_RACE_CONDITION_DETECTED | 0x12 | |
| RFC_INSUFFICIENT_RESOURCES | 0x13 | |
| RFC_CANNOT_CHANGE_FLOW_CONTROL_MECHANISM | 0x14 | |
| RFC_DLC_ALREADY_EXISTS | 0x15 | |
| RFC_DLC_REJ_SECURITY | 0x16 | |
| RFC_GENERIC_REFUSAL | 0x1f | |
| RFC_UNEXPECTED_PRIMITIVE | 0x20 | |
| RFC_INVALID_SERVER_CHANNEL | 0x21 | |
| RFC_UNKNOWN_MUX_ID | 0x22 | |

| RFC_LOCAL_ENTITY_TERMINATED_CONNECTION | 0x23 | |
|---|---|---|
| RFC_UNKNOWN_PRIMITIVE | 0x24 | |
| RFC_MAX_PAYLOAD_EXCEEDED | 0x25 | |
| RFC_INCONSISTENT_PARAMETERS | 0x26 | |
| RFC_INSUFFICIENT_CREDITS | 0x27 | |
| RFC_CREDIT_FLOW_CONTROL_PROTOCOL_VIOLATION | 0x28 | |
| RFC_RES_ACK_TIMEOUT | 0x30 | |
| RFC_CONNECTION_REJ_SSP_AUTH_FAIL | - | RFCOMM connection failed because of SSP authentication failure |
| L2CAP_CONNECTION_SSP_AUTH_FAIL | - | L2CAP connection failed because of SSP authentication failure |

**Table 14: RFCOMM errors**

This section explains the iWRAP events and their syntax.

# 8  Supported Bluetooth Profiles

## 8.1  RFCOMM with TS07.10

The RFCOMM protocol emulates the serial cable line settings and status of an RS-232 serial port and is used for providing serial data transfer. RFCOMM connects to the lower layers of the Bluetooth protocol stack through the L2CAP layer.

By providing serial-port emulation, RFCOMM supports legacy serial-port applications while also supporting the OBEX protocol among others. RFCOMM is a subset of the ETSI TS 07.10 standard, along with some Bluetooth specific adaptations.

The RFCOMM protocol supports up to 60 simultaneous connections between two Bluetooth devices. The number of connections that can be used simultaneously in a Bluetooth device is implementation-specific.

For the purposes of RFCOMM, a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them. The figure above shows the complete communication path. (In this context, the term application may mean other things than end-user application; e.g. higher layer protocols or other services acting on behalf of end-user applications.)

RFCOMM is intended to cover applications that make use of the serial ports of the devices in which they reside. In the simple configuration, the communication segment is a Bluetooth link from one device to another (direct connect), see the figure to the left. Where the communication segment is another network, Bluetooth wireless technology is used for the path between the device and a network connection device like a modem. RFCOMM is only concerned with the connection between the devices in the direct connect case, or between the device and a modem in the network case.

RFCOMM can support other configurations, such as modules that communicate via Bluetooth wireless technology on one side and provide a wired interface on the other side, as shown in the figure below. These devices are not really modems but offer a similar service. They are therefore not explicitly discussed here.

Basically two device types exist that RFCOMM must accommodate. Type 1 devices are communication end points such as computers and printers. Type 2 devices are those that are part of the communication segment; e.g. modems. Though RFCOMM does not make a distinction between these two device types in the protocol, accommodating both types of devices impacts the RFCOMM protocol.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/RFCOMM.htm

## 8.2  Service Discovery Protocol (SDP)

SDAP describes how an application should use SDP to discover services on a remote device. It illustrates several approaches to managing the device discovery via Inquiry and Inquiry Scan and service discovery via SDP. The ideas contained in the SDAP specification augment the basic specifications provided in GAP, SDP, and the basic processes of device discovery. The use cases for SDAP are intended to encompass the majority of service discovery scenarios associated with all profiles and devices.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/SDAP.htm

## 8.3  Serial Port Profile (SPP)

A scenario would be using two devices, such as PCs or laptops, as virtual serial ports and then connecting the two devices via Bluetooth technology.

The SPP defines two roles, Device A and Device B.

1. Device A – This is the device that takes initiative to form a connection to another device (initiator).

2. Device B – This is the device that waits for another device to take initiative to connect (acceptor).

The applications on both sides are typically legacy applications, able and wanting to communicate over a serial cable (which in this case is emulated). But legacy applications cannot know about Bluetooth procedures for setting up emulated serial cables, which is why they need help from some sort of Bluetooth aware helper application on both sides. (These issues are not explicitly addressed in this profile; the major concern here is for Bluetooth interoperability.)

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/SPP.htm

## 8.4 Headset Profile (HSP)

The HSP describes how a Bluetooth enabled headset should communicate with a computer or other Bluetooth enabled device such as a mobile phone.

HSP defines two roles, that of an Audio Gateway (AG) and a Headset (HS):

3. Audio Gateway (AG) – This is the device that is the gateway of the audio, both for input and output, typically a mobile phone or PC.

4. Headset (HS) – This is the device acting as the Audio Gateway's remote audio input and output mechanism.

The Baseband, LMP and L2CAP are the OSI layer 1 and 2 Bluetooth protocols. RFCOMM is the Bluetooth adaptation of GSM TS 07.10. SDP is the Bluetooth Service Discovery Protocol. Headset Control is the entity responsible for headset-specific control signalling; this signalling is AT command based.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/HSP.htm

## 8.5 Hands-Free Profile (HFP)

HFP describes how a gateway device can be used to place and receive calls for a hand-free device.

The HFP defines two roles, that of an Audio Gateway (AG) and a Hands-Free unit (HF):

- Audio Gateway (AG) – This is the device that is the gateway of the audio, both for input and output, typically a mobile phone.

- Hands-Free Unit (HF) – This is the device acting as the Audio Gateway's remote audio input and output mechanism. It also provides some remote control means.

Hands-Free control is the entity responsible for Hands-Free unit specific control signaling; this signaling is AT command based.

Although not shown in the model to the left, it is assumed by this profile that Hands-Free Control has access to some lower layer procedures (for example, Synchronous Connection establishment).

The audio port emulation layer shown in the figure to the left is the entity emulating the audio port on the Audio Gateway, and the audio driver is the driver software in the Hands-Free unit.

For the shaded protocols/entities in the figure to the left, the Serial Port Profile is used as the base standard. For these protocols, all mandatory requirements stated in the Serial Port Profile apply except in those cases where this specification explicitly states deviations.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/HFP.htm

## 8.6 Dial-up Networking Profile (DUN)

DUN provides a standard to access the Internet and other dial-up services over Bluetooth technology. The most common scenario is accessing the Internet from a laptop by dialing up on a mobile phone wirelessly. It is based on SPP and provides for relatively easy conversion of existing products, through the many features that it has in common with the existing wired serial protocols for the same task. These include the AT command set specified in ETSI 07.07 and PPP.

Like other profiles built on top of SPP, the virtual serial link created by the lower layers of the Bluetooth protocol stack is transparent to applications using the DUN profile. Thus, the modem driver on the data-terminal device is unaware that it is communicating over Bluetooth technology. The application on the data-terminal device is similarly unaware that it is not connected to the gateway device by a cable.

DUN describes two roles, the gateway and terminal devices. The gateway device provides network access for the terminal device. A typical configuration consists of a mobile phone acting as the gateway device for a personal computer acting as the terminal role.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/DUN.htm

## 8.7 OBEX Object Push Profile (OPP)

OPP defines the roles of push server and push client. These roles are analogous to and must interoperate with the server and client device roles that GOEP defines. It is called push because the transfers are always instigated by the sender (client), not the receiver (server). OPP focuses on a narrow range of object formats to maximize interoperability. The most common acceptable format is the vCard. OPP may also be used for sending objects such as pictures or appointment details.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/OPP.htm

## 8.8 OBEX File Transfer Profile (FTP)

FTP defines how folders and files on a server device can be browsed by a client device. Once a file or location is found by the client, a file can be pulled from the server to the client, or pushed from the client to the server using GOEP.

The FTP defines two roles, that of a Client and a Server:

- Client – The Client device initiates the operation, which pushes and pulls objects to and from the Server.

- Server – The Server device is the target remote Bluetooth device that provides an object exchange server and folder browsing capability using the OBEX Folder Listing format.

The Baseband, LMP and L2CAP are the OSI layer 1 and 2 Bluetooth protocols. RFCOMM is the Bluetooth adaptation of GSM TS 07.10. SDP is the Bluetooth Service Discovery Protocol. OBEX is the Bluetooth adaptation of IrOBEX.

The RFCOMM, L2CAP, LMP, and Baseband interoperability requirements are defined in GOEP.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/FTP.htm

## 8.9  Advanced Audio Distribution Profile (A2DP)

A2DP describes how stereo-quality audio can be streamed from a media source to a sink.

The profile defines two roles of an audio device: source and sink.

- Source (SRC) – A device is the SRC when it acts as a source of a digital audio stream that is delivered to the SNK of the Piconet.

- Sink (SNK) – A device is the SNK when it acts as a sink of a digital audio stream delivered from the SRC on the same Piconet.

A2DP defines the protocols and procedures that realize distribution of audio content of high-quality in mono or stereo on ACL channels. The term "advanced audio," therefore, should be distinguished from "Bluetooth audio," which indicates distribution of narrow band voice on SCO channels as defined in the baseband specification.

This profile relies on GAVDP. It includes mandatory support for low complexity subband codec (SBC) and supports optionally MPEG-1,2 Audio, MPEG-2,4 AAC and ATRAC.

The audio data is compressed in a proper format for efficient use of the limited bandwidth. Surround sound distribution is not included in the scope of this profile.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/A2DP.htm

## 8.10 Audio Video Remote Control Profile (AVRCP)

AVRCP is designed to provide a standard interface to control TVs, hi-fi equipment, or others to allow a single remote control (or other device) to control all the A/V equipment to which a user has access. It may be used in concert with A2DP or VDP.

The AVRCP defines two roles, that of a controller and target device.

- Controller – The controller is typically considered the remote control device.

- Target – The target device is the one whose characteristics are being altered.

This protocol specifies the scope of the AV/C Digital Interface Command Set (AV/C command set, defined by the 1394 trade association) to be applied, realizing simple implementation and easy operability. This protocol adopts the AV/C device model and command format for control messages and those messages are transported by the Audio/Video Control Transport Protocol (AVCTP).

In AVRCP, the controller translates the detected user action to the A/V control signal, and then transmits it to a remote Bluetooth enabled device. The functions available for a conventional infrared remote controller can be realized in this protocol. The remote control described in this protocol is designed specifically for A/V control only.

**Source: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/AVRCP.htm

## 8.11 Human Interface Device Profile (HID)

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: Bluetooth SIG, URL:**

http://www.bluetooth.com/Bluetooth/Technology/Works/HID.htm

# 8.12 Phone Book Access Profile (PBAP)

Phone Book Access Profile (PBAP) is a profile that allows exchange of Phone Book Objects between devices. It can be used for example between a car kit and a mobile phone to:

1. Allow the car kit to display the name of the incoming caller;

2. Allow the car kit to download the phone book so the user can initiate a call from the car display

The PBAP defines two roles:

- Phone Book Server Equipment (PSE)**:** this role is for the device that contains the source phone-book objects; for example, a mobile phone.

- Phone Book Client Equipment (PCE) role**:** this role is for the device that retrieves phone-book objects from the PSE device; for example, a portable navigation device (PND).

iWRAP firmware supports PCE role.

# 8.13 Health Device Profile (HDP)

The Bluetooth Health Device Profile (HDP) allows the transmission of health and medical related data between Bluetooth devices. The typical uses cases are wireless blood pressure monitors, weight scales, blood glucose meters and ECG transmitters. The HDP profile offers unique features and extra reliability not included in the other Bluetooth profiles. A key feature in the HDP profile is also the application level interoperability defined by a set of IEEE 11073-xxxxx standards.

he HDP defines two roles:

- HDP sink**:** this role is for the device that receives the data from one or several medical sensors and processes it or relays it to other services like Personal Health Records.

- HDP source**:** this role is for the device that is used to make the measurements and transmit them over Bluetooth connection for future processing, for example a blood pressure meter.

# 8.14 Device Identification Profile (DI)

TDB

## 8.15 Bluegiga Proprietary Profiles

### 8.15.1    Bluegiga IO Profile (BGIO)

The BGIO profile is a Bluegiga proprietary profile based on the Bluetooth RFCOMM. BGIO allows one to read/write the status of GPIO and AIO lines of Bluegiga's Bluetooth modules. The controlling is made using a Bluegiga proprietary binary protocol over the Bluetooth RFCOMM connection.

Like in the Serial Port Profile there are two roles in the BGIO profile:

1. Device A – This is the device that takes initiative to form a connection to another device (initiator).

2. Device B – This is the device that waits for another device to take initiative to connect (acceptor).

### 8.15.2    Over-the-Air Profile (OTA)

The OTA profile is a 2nd Bluegiga proprietary profile based on the Bluetooth RFCOMM. OTA profile allows one to wirelessly configure the iWRAP firmware settings of Bluegiga's Bluetooth modules. The controlling is made using ASCII based iWRAP commands over the Bluetooth RFCOMM connection. OTA profile also includes a second level of authentication and does not only rely on Bluetooth pairing.

The OTA profile only contains one role:

- Device A – This is the device that takes initiative to form a connection to another device (initiator).

The connecting device can be any Bluetooth device supporting the Bluetooth Serial Port Profile.

## 8.16 UUIDs of Bluetooth profiles

The table below lists the UUIDs of different Bluetooth profiles.

| UUID: | *Bluetooth* Profile: |
|---|---|
| 0001 | SDP |
| 0003 | RFCOMM |
| 0008 | OBEX |
| 000C | HTTP |
| 000F | BNEP |
| 0100 | L2CAP |
| 1000 | Service Discovery Server Service ClassID |
| 1001 | Browse Group Descriptor Service ClassID |
| 1002 | Public Browse Group |
| 1101 | Serial Port Profile |
| 1102 | LAN Access Using PPP |
| 1103 | Dial up Networking |
| 1104 | IrMC Sync |
| 1105 | OBEX Object Push Profile |
| 1106 | OBEX File Transfer Profile |
| 1107 | IrMC Sync Command |
| 1108 | Headset |
| 1109 | Cordless Telephony |
| 110A | Audio Source |
| 110B | Audio Sink |
| 110C | A/V_Remote Control Target |
| 110D | Advanced Audio Distribution Profile (A2DP) |
| 110E | A/V_Remote Control |
| 110F | Video Conferencing |

| 1110 | Intercom |
|------|----------|
| 1111 | Fax |
| 1112 | Headset Audio Gateway |
| 1113 | WAP |
| 1114 | WAP_CLIENT |
| 1115 | Personal Area Networking User |
| 1115 | PANU |
| 1116 | Network Access Point |
| 1116 | NAP |
| 1117 | Group Network |
| 1117 | GN |
| 1118 | Direct Printing |
| 1119 | Reference Printing |
| 111A | Imaging |
| 111B | Imaging Responder |
| 111C | Imaging Automatic Archive |
| 111D | Imaging Referenced Objects |
| 111E | Hands-Free |
| 111F | Hands-Free Audio Gateway |
| 1120 | Direct Printing Reference Objects Service |
| 1121 | ReflectedUI |
| 1122 | Basic Printing |
| 1123 | Printing Status |
| 1124 | Human Interface Device Service |
| 1125 | Hardcopy Cable Replacement |
| 1126 | HCR_Print |
| 1127 | HCR_Scan |

| 1128 | Common_ISDN_Access |
|------|--------------------|
| 1129 | Video Conferencing GW |
| 112A | UDI_MT |
| 112B | UDI_TA |
| 112C | Audio/Video |
| 112D | SIM_Access |
| 112E | Phonebook Access - PCE |
| 112F | Phonebook Access - PSE |
| 1130 | Phonebook Access |
| 1200 | PnP Information |
| 1201 | Generic Networking |
| 1202 | Generic File Transfer |
| 1203 | Generic Audio |
| 1204 | Generic Telephony |
| 1205 | UPNP_Service |
| 1206 | UPNP_IP_Service |
| 1300 | ESDP_UPNP_IP_PAN |
| 1301 | ESDP_UPNP_IP_LAP |
| 1302 | ESDP_UPNP_L2CAP |
| 1303 | Video Source |
| 1304 | Video Sink |
| 1305 | Video Distribution |

**Table 15: UUIDs and Profiles**

For more information, please go to (login required):

- https://programs.Bluetooth.org/apps/content/?doc_id=49709

# 9 Useful Information

This chapter contains useful information about iWRAP and its usage.

## 9.1 PS-keys and how to change them

The Bluegiga Bluetooth modules use Cambridge Silicon Radio's (CSR) Bluetooth chips. The CSR chips contain a set of low level parameters called PS-keys that can be used to change the behaviour of the Bluetooth chips. These parameters can also be changed on the Bluegiga Bluetooth modules. Usually they do not need to be modified as they are optimized by Bluegiga Technologies for each module, but if necessary it should be done carefully and understanding what the parameters will do.

Software called PSTool allow the user to change the PS-keys of the module. PSTool is part of the Bluesuite software package, which can be downloaded from Bluegiga's Tech Forum.

The parameters can be changed over the UART or SPI interfaces. With UART a typical level sifter can be used but SPI interface requires a special programming cable is required. The programming cable is included in all the evaluation kits and the schematics are also available in the Tech Forum.

To get access to the PS-keys over UART interface the following steps need to be done:

1. Connect the Bluetooth module via RS232 to a Windows PC

2. Power up the Bluetooth module

3. Open a terminal connection to iWRAP and issue command: "**BOOT 1**" to switch to BCSP mode.

   Alternatively "**BOOT 4**" can be used to switch to H4 mode.

4. Close the terminal software and Open PSTool application

5. Use connection settings: **BCSP**, **COMn** and **115200** (or H4 if you switched to H4 mode)

6. Change the necessary PS-keys.

   Remember to press **SET** in PSTool after every parameter change.

7. Close PSTool and reset your module.

With SPI interface are exactly the same, except that you can skip step 3 and you also need to select SPI as the connection method.

**NOTE:**

- When using BCSP or H4, the UART baud rate does NOT depend on the iWRAP's "**SET CONTROL BAUD**" setting, but is defined by using PS key "PSKEY_UART_BAUD RATE". By default, the parameter has value 115200 bps.

- It is possible to configure the module in a way that the UART interface does not respond to BCSP or H4. In this situation the SPI interface is the only way to connect to the module and restore the factory settings.

- You can always recover the factory settings by reinstalling the firmware with the **iWRAP Update Client** available in the Bluegiga's Tech Forum.

## 9.2  BlueTest radio test utility

BlueTest is software, which can be used to perform several built-in radio tests, such as Bit Error Rate (BER) measurements, TX power measurements and RX measurements. It is usually required to enable various radio test modes for certification purposes.

Just like PSTool BlueTest can be used over UART and SPI interfaces and BlueTest also supports BCSP and H4 protocols.   uses the BCSP protocol to talk to the module and can be used in a similar way as PSTool. i

To use BlueTest over UART interface the following steps need to be done:

1. Connect the Bluetooth module via RS232 to a Windows PC

2. Power up the Bluetooth module

3. Open a terminal connection to iWRAP and issue command: "**BOOT 1**" to switch to BCSP mode.

4. Alternatively "**BOOT** 4" can be used to switch to H4 mode.

5. Close the terminal software and Open BlueTest application

6. Use connection settings: **BCSP**, **COMn** and **115200** (or H4 if you switched to H4 mode)

7. Perform the necessary radio tests

8. Close BlueTest and reset your module.

With SPI interface are exactly the same, except that you can skip step 3 and you also need to select SPI as the connection method.

Always consult Bluegiga Technologies before performing any radio tests for certification purposes. Many radio tests require the configuration of radio power parameters and they are unique to each module type. Using incorrect parameters may lead to failures in certification testing.

Some of the BlueTest radio tests can also be enabled with iWRAP commands. Please see the documentation of **TEST** command.

## 9.3  Switching between iWRAP and HCI firmware

The iWRAP firmware build also contains the lower level Host Controller Interface (HCI) firmware. The user can quite easily switch the firmware configuration between iWRAP and HCI. The firmware mode is controlled with a single PS-key called: PSKEY_INITIAL_BOOTMODE and by changing its value the firmware mode can be switched easily.

The values for PSKEY_INITIAL_BOOTMODE are exactly the same as the parameters for the **BOOT** command. They were:

- 0000: iWRAP mode

- 0001: HCI, BCSP protocol          (default: 115200,8e1)

- 0003: HCI, USB          (default: bus powered, design identical to USB on evaluation kits)

- 0004: HCI, H4 protocol          (default: 115200,8n1)

If the interface parameters for the BCSP and H4 modes need to be changed, additional PS-keys need to be modified. The PS-keys are:

- PSKEY_UART_BAUDRATE          : Configures the UART baud rate

- PSKEY_UART_CONFIG_H4          : Configures the H4 interface

- PSKEY_UART_CONFIG_BCSP          : Configures the BCSP interface

- PSKEY_USB_XXXX          : Several keys exists for USB configuration.

    Please refer to the USB design guide for more information.

An alternative way to switch the firmware configuration is to reinstall the firmware with the **iWRAP Update Client** available in the Bluegiga's Tech Forum.

## 9.4  Firmware updates

### 9.4.1  Firmware update over SPI

The SPI interface is dedicated only for firmware updates. The Onboard Installation Kit (SPI programming cable) and a Windows software called iWRAP update client (or BlueTest) can be used to update or restore the firmware over SPI interface.

The iWRAP update client always restores the firmware to a factory status and settings. BlueFlash software, which is part of BlueTest package can be used to make dedicated firmware updates, which for example leave the user configuration (iWRAP settings and PS-keys) intact.

iWRAP update client is an easier and the suggested way to do the firmware upgrade instead of BlueFlash. iWRAP update client can recognize the hardware and software version of the module and reflash correct firmware and parameters into the module, and the user just needs to select the firmware version. With BlueFlash it is possible to install incorrect firmware to the module damage its operation.

### 9.4.2  Firmware update over UART

The firmware can also be updated over the UART interface. A protocol called Device Firmware Upgrade (DFU) is available for this purpose. BlueSuite software package contains a tool called DFUWizard tool, which allows the updates to be made from a Windows based PC in a similar way as with iWRAP update client. Bluegiga has also a SerialDFU tool available, which can be used for the same purpose. The difference however is that SerialDFU is written in Python and can be used on other operating systems and it can also write PS-keys into the module during the update.

The DFU protocol is and open protocol can be integrated into various systems for example to perform on the field updates. Please contact Bluegiga Technologies by sending email to support@bluegiga.com for the DFU protocol description and sample code.

**Typical DFU firmware update files are:**

- iWRAP version update :              200-300kB

- iWRAP + *Bluetooth* stack:         70-900kB

- Full update:                       ~1MB

**Note:**

- More information and instructions about firmware updates can be found from *Firmware & Parameters User Guide*, which is available in Tech Forum.

## 9.5  UART hardware flow control

Hardware flow control is enabled by default and it should not be disabled unless mandatory, because without the hardware flow control the data transmission may not be reliable. However if the hardware flow control must be disabled it can be done by changing the value of PS-key: PSKEY_UART_CONFIG_XXX.

(XXX = USR, H4, H5 or BCSP).

With iWRAP, the PS key is PSKEY_UART_CONFIG_USR.

- If PSKEY_UART_CONFIG_USR is **08a8**, HW flow control is enabled

- If PSKEY_UART_CONFIG_USR is **08a0**, HW flow control is disabled

Hardware flow control can be disabled also with a proper hardware design. If the hardware flow control is enabled from PS-keys, but no flow control is used, the following steps should be implemented in the hardware design:

- CTS pin must be grounded

- RTS pin must be left floating

**WARNING:**

- If hardware flow control is disabled and iWRAP buffers are filled (in command or data mode), the firmware may hang and needs a physical reset to survive. Therefore, hardware flow control should be used whenever possible to avoid this situation.

- However, if hardware flow control must be disabled, the host system should be designed in a way that it can recognize that the firmware has hung and is able to survive it.

## 9.6  RS232 connections diagram
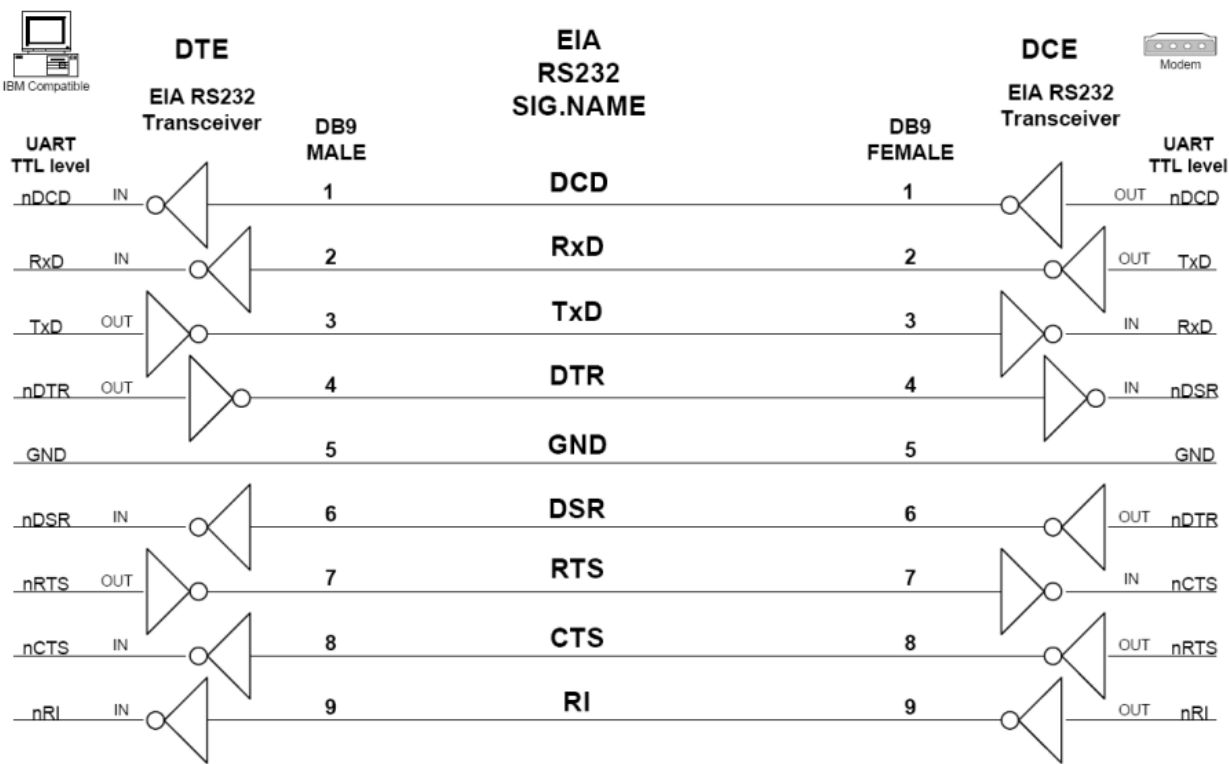


**Figure 9:** RS232 connections

# 10 General Bluetooth Information

## 10.1 Secure Simple Pairing (SSP) Overview

The primary goal of Secure Simple Pairing is to simplify the pairing procedure for the user. Secondary goals are to maintain or improve the security in Bluetooth wireless technology. Since high levels of security and ease-of-use are often at opposite ends of the spectrum in many technologies and products, much care has been taken to maximize security while minimizing complexity from the end user's point of view.

### 10.1.1 SECURITY GOALS

Secure Simple Pairing has two security goals: protection against passive eavesdropping and protection against man-in-the-middle (MITM) attacks (active eavesdropping). It is a goal of Secure Simple Pairing to exceed the maximum security level provided by the use of a 16 alphanumeric PIN with the pairing algorithm used in Bluetooth Core Specification version 2.0 + EDR and earlier versions. Note that many Bluetooth devices compliant with Bluetooth Core Specification 2.0 + EDR and earlier versions use a 4-digit PIN or a fixed PIN of commonly known values significantly limiting the security on the link.

### 10.1.2 PASSIVE EAVESDROPPING PROTECTION

A strong link key coupled with a strong encryption algorithm is necessary to give the user protection against passive eavesdropping. The strength of the link key is based on the amount of entropy (or randomness) in its generation process which would not be known by an attacker. Using legacy pairing, the only source of entropy is the PIN which, in many use cases, is typically four digits either selected by the user or fixed for a given product. Therefore, if the pairing procedure and one authentication exchange is recorded one can run an

exhaustive search to find the PIN in a very short amount of time on commonly available computing hardware. With Secure Simple Pairing, the recording attack becomes much harder as the attacker must have solved a hard problem in public key cryptography in order to derive the link key from the recorded information. This protection is independent of the length of the passkey or other numeric values that the user must handle. Secure Simple Pairing gives the same resistance against the recording and passive eavesdropping attacks even when the user is not required to do anything.

Secure Simple Pairing uses Elliptic Curve Diffie Hellman (ECDH) public key cryptography as a means to thwart passive eavesdropping attacks. ECDH provides a very high degree of strength against passive eavesdropping attacks but it may be subject to MITM attacks, which however, are much harder to perform in practice than the passive eavesdropping attack.

Using the security protocols in the Bluetooth Core Specification version 2.0 + EDR and earlier with a 16 numeric digit PIN achieves about 53 bits of entropy whereas a 16 character alphanumeric, case sensitive PIN yields about 95 bits of entropy when the entire 62 character set is used ([0, … 9, 'A', … 'Z', 'a', … 'z']). Secure Simple Pairing has approximately 95 bits of entropy using the FIPS approved P192 elliptic curve which is at least as good as the entropy in Bluetooth Core Specification 2.0 + EDR and earlier using a 16 character, alphanumeric, case sensitive PIN. Secure Simple Pairing, therefore, exceeds the security requirements of the Bluetooth SIM Access Profile (SAP) which is the profile with the most stringent security requirements. ECDH cryptography was selected over standard Diffie Hellman (often referred to as DH76) since it is computationally less complex and less likely to exceed the low computational capacity in common Bluetooth Controllers.

### 10.1.3 MAN-IN-THE-MIDDLE PROTECTION

A man-in-the-middle (MITM) attack occurs when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to a third (attacking) device that plays the role of the device they are attempting to pair with. The third device then relays information between the two devices giving the illusion that they are directly connected. The attacking device may even eavesdrop on communication between the two devices (known as active eavesdropping) and is able to insert and modify information on the connection. In this type of attack, all of the information exchanged between the two devices are compromised and the attacker may inject commands and information into each of the devices thus

potentially damaging the function of the devices. Devices falling victim to the attack are capable of communicating only when the attacker is present. If the attacker is not active or out range, the two victim devices will not be able to communicate directly with each other and the user will notice it.

To prevent MITM attacks, Secure Simple Pairing offers two user assisted numeric methods: numerical comparison or passkey entry. If Secure Simple Pairing would use 16 decimal digit numbers, then the usability would be the same as using legacy pairing with 16 decimal digit PIN. The chance for a MITM to succeed inserting its own link keys in this case is a 1 in $10^{16} = 2^{53}$ pairing instances, which is an unnecessarily low probability.

Secure Simple Pairing protects the user from MITM attacks with a goal of offering a 1 in 1,000,000 chance that a MITM could mount a successful attack. The strength of the MITM protections was selected to minimize the user impact by using a six digit number for numerical comparison and Passkey entry. This level of MITM protection was selected since, in most cases, users can be alerted to the potential presence of a MITM attacker when the connection process fails as a result of a failed MITM attack. While most users feel that provided that they have not compromised their passkey, a 4-digit key is sufficient for authentication (i.e. bank card PIN codes), the use of six digits allows Secure Simple Pairing to be FIPS compliant and this was deemed to have little perceivable usability impact.

# 10.1.4 ASSOCIATION MODELS

Secure Simple Pairing uses four association models referred to as Numeric Comparison, Just Works, Out Of Band, and Passkey Entry. Each of these association models are described in more detail in the following sections.

The association model used is deterministic based on the I/O capabilities of the two devices.

## 10.1.4.1 Numeric Comparison

The Numeric Comparison association model is designed for scenarios where both devices are capable of displaying a six digit number and both are capable of having the user enter "yes" or "no". A good example of this model is the cell phone / PC scenario.

The user is shown a six digit number (from "000000" to "999999") on both displays and then asked whether the numbers are the same on both devices. If "yes" is entered on both devices, the pairing is successful.

The numeric comparison serves two purposes. First, since many devices do not have unique names, it provides confirmation to the user that the correct devices are connected with each other. Second, the numeric comparison provides protection against MITM attacks.

Note that there is a significant difference from a cryptographic point of view between Numeric Comparison and the PIN entry model used by Bluetooth Core Specification and earlier versions. In the Numeric Comparison association model, the six digit number is an artifact of the security algorithm and not an input to it, as is the case in the Bluetooth security model. Knowing the displayed number is of no benefit in decrypting the encoded data exchanged

## 10.1.4.2 Just Works

The Just Works association model is primarily designed for scenarios where at least one of the devices does not have a display capable of displaying a six digit number nor does it have a keyboard capable of entering six decimal digits. A good example of this model is the cell phone/mono headset scenario where most headsets do not have a display.

The Just Works association model uses the Numeric Comparison protocol but the user is never shown a number and the application may simply ask the user to accept the connection (exact implementation is up to the end product manufacturer). The Just Works association model provides the same protection as the Numeric Comparison association model against passive eavesdropping but offers no protection against the MITM attack.

When compared against today's experience of a headset with a fixed PIN, the security level of the Just Works association model is considerably higher since a high degree of protection against passive eavesdropping is realized.

### 10.1.4.3    Out of Band

The Out of Band (OOB) association model is primarily designed for scenarios where an Out of Band mechanism is used to both discover the devices as well as to exchange or transfer cryptographic numbers used in the pairing process. In order to be effective from a security point of view, the Out of Band channel should provide different properties in terms of security compared to the Bluetooth radio channel. The Out of Band channel should be resistant to MITM attacks. If it is not, security may be compromised during authentication.

The user's experience differs a bit depending on the Out of Band mechanism. As an example, with a Near Field Communication (NFC) solution, the user(s) will initially touch the two devices together, and is given the option to pair the first device with the other device. If "yes" is entered, the pairing is successful. This is a single touch experience where the exchanged information is used in both devices. The information exchanged includes discovery information (such as the Bluetooth Device Address) as well as cryptographic information. One of the devices will use a Bluetooth Device Address to establish a connection with the other device. The rest of the exchanged information is used during authentication.

The OOB mechanism may be implemented as either read only or read/write. If one side is read only, a one-way authentication is performed. If both sides are read/write, a two-way authentication is performed.

The OOB protocol is selected only when the pairing process has been activated by previous OOB exchange of information and one (or both) of the device(s) gives OOB as the IO capabilities. The protocol uses the information which has been exchanged and simply asks the user to confirm connection. The OOB association model supports any OOB mechanism where cryptographic information and the Bluetooth Device Address can be exchanged. The OOB association model does not support a solution where the user has activated a Bluetooth connection and would like to use OOB for authentication only.

### 10.1.4.4    Passkey Entry

The Passkey Entry association model is primarily designed for the scenario where one device has input capability but does not have the capability to display six digits and the other device has output capabilities. A good example of this model is the PC and keyboard scenario.

The user is shown a six digit number (from "000000" to "999999") on the device. If the value entered on the second device is correct, the pairing is successful. Note that there is a significant difference from a cryptographic point of view between Passkey Entry and the PIN entry model used by Bluetooth Core Specification 2.0 + EDR and earlier versions. In the Passkey Entry association model, the six digit number is independent of the security algorithm and not an input to it, as is the case in the 2.0 + EDR security model. Knowing the entered number is of no benefit in decrypting the encoded data exchanged between the two devices.

**Source**: Specification of The Bluetooth System  Version 2.1 + EDR, The Bluetooth SIG, 26 July 2007

## 10.2 Sniff power saving mode

In Sniff mode, the duty cycle of the slave's activity in the piconet may be reduced. If a slave is in active mode on an ACL logical transport, it shall listen in every ACL slot to the master traffic, unless that link is being treated as a scatternet link or is absent due to hold mode. With sniff mode, the time slots when a slave is listening are reduced, so the master shall only transmit to a slave in specified time slots. The sniff anchor points are spaced regularly with an interval of Tsniff.
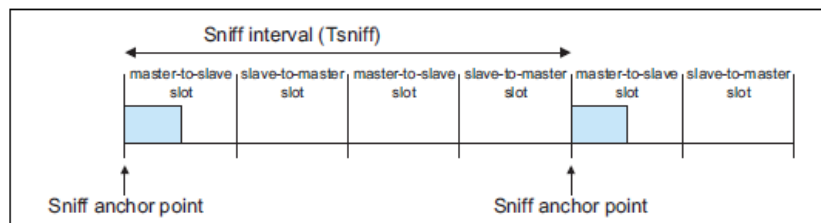


**Figure 10: Sniff anchor points**

The slave listens in master-to-slave transmission slots starting at the sniff anchor point. It shall use the following rules to determine whether to continue listening:

- If fewer than Nsniff attempt master-to-slave transmission slots have elapsed since the sniff anchor point then the slave shall continue listening.

- If the slave has received a packet with a matching LT_ADDR that contains ACL data (DM, DH, DV, or AUX1 packets) in the preceding Nsniff timeout master- to-slave transmission slots then it shall continue listening.

- If the slave has transmitted a packet containing ACL data (DM, DH, DV, or AUX1 packets) in the preceding Nsniff timeout slave-to-master transmission slots then it shall continue listening.

- If the slave has received any packet with a matching LT_ADDR in the preceding Nsniff timeout master-to-slave transmission slots then it may continue listening.

- A device may override the rules above and stop listening prior to Nsniff timeout or the remaining Nsniff attempt slots if it has activity in another piconet.

It is possible that activity from one sniff timeout may extend to the next sniff anchor point. Any activity from a previous sniff timeout shall not affect activity after the next sniff anchor point. So in the above rules, only the slots since the last sniff anchor point are considered.

Note that Nsniff attempt =1 and Nsniff timeout =0 cause the slave to listen only at the slot beginning at the sniff anchor point, irrespective of packets received from the master.

Nsniff attempt =0 shall not be used.

Sniff mode only applies to asynchronous logical transports and their associated LT_ADDR.Sniff mode shall not apply to synchronous logical transports, therefore, both masters and slaves shall still respect the reserved slots and retransmission windows of synchronous links.

To enter sniff mode, the master or slave shall issue a sniff command via the LM protocol. This message includes the sniff interval Tsniff and an offset Dsniff. In addition, an initialization flag indicates whether initialization procedure 1 or 2 shall be used. The device shall use initialization 1 when the MSB of the current master clock (CLK27) is 0; it shall use initialization 2 when the MSB of the current master clock (CLK27) is 1. The slave shall apply the initialization method as indicated by the initialization flag irrespective of its clock bit value CLK27. The sniff anchor point determined by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

CLK27-1 mod Tsniff = Dsniff for initialization 1

(CLK27,CLK26-1) mod Tsniff = Dsniff for initialization 2

this implies that Dsniff must be even

After initialization, the clock value CLK(k+1) for the next sniff anchor point shall be derived by adding the fixed interval Tsniff to the clock value of the current sniff anchor point:

$$CLK(k+1) = CLK(k) + Tsniff$$

**Source**: Specification of The Bluetooth System  Version 2.1 + EDR, The Bluetooth SIG, 26 July 2007

# 11 Known Issues

| Issue | | Explanation |
|---|---|---|
| - | Using multiple DLCs can crash iWRAP | Opening several connections to iWRAP using the same channel may crash the firmware. UUID should be used instead of a channel. This is a bug in the CSR firmware. |
| - | Listing remote SDP record may run out of memory | When a service discovery is made by using the SDP command and if root mode is used and remote device supports many services, iWRAP may run out of memory and reset. To overcome this, only a specific service should be searched for instead of using root mode. |
| - | Do not force sniff | If sniff is enabled by using the 'SET BT SNIFF' command, iWRAP cannot unsniff if remote end requests for it. |
| - | Frame mode flow control hangs | In multiplexing mode, the firmware will hang if data length is longer than 100 bytes. A physical reset is needed. This is a bug in the CSR firmware. |
| - | Inquiry RSSI and clock caching | If RSSI in the inquiry and clock offset caching are enabled, connections can not be opened. This is a bug in the CSR firmware. |
| - | HW flow control | If HW flow control is not used and iWRAP buffers are filled either in data or command mode, the firmware will hang and needs a physical reset. This is a bug in the CSR firmware. |
| - | Simultaneous connection between two iWRAPs | Two simultaneous ACL connections can not be opened between two iWRAPs. |
| - | SET CONTROL INIT RESET | Issuing SET CONTROL INIT RESET will result in an infinite reset loop. PSKEY_USR_27 must be deleted to survive this condition. |
| 238 | SELECT [link_id] with MUX mode | SELECT [link_id] can be issued in MUX mode and it puts iWRAP into normal data mode. |
| 387 | Interactive pairing | Pin code should be enabled even if interactive pairing mode is used. If pin code is disabled with "SET BT PAIR *", Bluetooth security mode 0 is used and interactive pairing does not produce "AUTH" vent. |
| 337 | Link Ids get mixed | If an existing Bluetooth connection is lost while opening a new connection and exactly between "CALL [link_id]" and "CONNECT [link_id]" events, the Link IDs the mixed. Example:<br><br>CALL 00:00:00:00:00:00 1101 rfcomm<br>CALL 1<br>NO CARRIER 0 ERROR 0<br>CONNECT 0 RFCOMM 1 |
| 670 | Outgoing connections | Outgoing HFP, HID, A2DP etc. connections can be made |

| | | even if the profile is not locally enabled. This may cause strange behavior, usually disconnections, since if the profiles are not enabled iWRAP will not behave according to the profile specification. |
|---|---|---|
| 711 | PIO bindings | Binding SET CONTROL ESCAPE or CD or MSC to a PIO and triggering the PIO event causes iWRAP to crash. |
| 717 | Switching between MUX and normal mode when connections are already in place | iWRAP's data and command modes get mixed when an RFCOMM connection is active and the user switches from MUX to normal mode or from normal mode to MUX mode. In the former case iWRAP will not receive any data from the other end; in the latter case any data received from the other end will be output to the UART even though iWRAP is in command mode. |
| 478 | Inquiry with EIR enabled may crash the chip | In an environment with many Bluetooth devices, using EIR may crash the chip, because the chip's memory simply runs out. This is a CSR issue. |
| 746 | Sending long commands too quickly will freeze iWRAP | Sending a 512+ bytes long command will hang the iWRAP command parser. Also, sending multiple long commands (say, 200 bytes long, 10 times in a row) in quick succession may cause this. The UART buffers fill, the chip asserts the CTS pin and will not accept any more data from the UART. Only a reset will recover iWRAP. The workaround is simple: use the OK flag (see **SET CONTROL CONFIG**) and wait until each command is processed before sending a new command, and don't send commands which exceed 511 bytes (including carriage return and line feed). |
| 736 | A2DP STREAMING STOP and A2DP STREAMING START do not work with iPhone. | The iPhone's A2DP implementation is flawed, and it will malfunction whenever an A2DP sink wishes to pause and resume the stream. It will accept the pause and resume commands, but will not resume audio unless the Bluetooth connection is restarted. This effectively renders iWRAP's **PLAY** command unusable with the iPhone, because it will pause the stream while it is playing the tone, and resume streaming afterwards. |
| 677 | HFP in MUX mode | If MUX mode is used and data is sent directly to the HFP link iWRAP hangs and a physical reset is needed to recover. One should not send data directly to the HFP link in MUX mode. |
| 742 | Long SET CONTROL BIND commands | **SET CONTROL BIND has** a limitation of 30 characters in the command. |
| 763 | A2DP and sniff mode | Using sniff mode with A2DP and then issuing "A2DP STREAMING STOP" on A2DP sink terminates the connection. However this only occurs if "**SET {link_id} SNIFF 4 4**" or "**SET {link_id} SNIFF 4 2**" values are used. |
| 773 | iPhone | If SPP profile is enabled iPhone can not open connection to iWRAP. This is an issue in iPhone's Bluetooth |

| | | implementation rather then in iWRAP. |
|---|---|---|
| 501 | Page timeout / supervision timeout | Page timeout MUST BE smaller then supervision timeout, or otherwise a failed page will close all active connections. |
| 681 | Binding multiple actions to same PIO | Multiple actions (SET CONTROL ESCAPE, SET CONTROL CD etc.) can be bound to same PIO and iWRAP does not give SYNTAX ERROR. In practice this setup does not work however |
| 680 | SET CONTROL READY does not check if PIO is valid | An invalid PIO mask can be given to SET CONTROL READY command. |
| 544 | Page scan mode 0 | If page scan mode is set to 0 iWRAP will be visible even if page mode is set to 1. |
| 586 | Bluesoleil (v. 6.2.227.11) and HFP | iWRAP sends AT+COPS command during HFP connection negotiation, which is not answered by Bluesoleil. This causes iWRAP to close the HFP connection after a 5 second timeout. This is an issue of the Bluesoleil Bluetooth stack |
| 712 | SET CONTROL ESCAPE | SET CONTROL ESCAPE accepts hex number as a first parameter although it should be a decimal number. |
| 723 | RING event received in data mode | If you make two Bluetooth connections to iWRAP, the RING event of the 2nd connection is received while in data mode. This can be avoided by using the MUX mode. |
| 827 | Closing connection after ECHO command | With iWRAP4 if the connection is closed very quickly after data has been sent with ECHO command the data may be lost. Some sleep should be added between ECHO and CLOSE commands. |
| 833 | CALL issued too quickly after CLOSE | If a CALL command is issued right after CLOSE command before the NO CARRIER event is received the new connection will not open and iWRAP state machine goes into a wrong state. Solution: Wait for NO CARRIER before issuing a new CALL command. |
| 835 | Two concurrent CALLs | iWRAP can not make two or more concurrent calls but they need to executed one at a time and a response (CONNECT or NO CARRIER) must be received before a new outgoing connection attempt can be made. |
| 851 | SDP during inquiry | If SDP query is made during inquiry process, iWRAP will crash. |
| 899 | SET CONTROL PREAMP | Set control preamp does not have any effect on WT12/WT11 or WT41 although the command can be executed. |
| 877 | Very small supervision timeout | Using 1-3 as a supervision timeout will cause the Bluetooth connection to be closed so quickly that the RING event is not received and only NO CARRIER is displayed. |

| 826 | PLAY with insufficient parameters | If PLAY command is issued with insufficient parameters f.ex "**PLAY 8**" iWRAP will play random noise until the ringtone is stopped with "**PLAY**" |

**Table 16: Known issues in iWRAP**

# 12 iWRAP Usage Examples

This section contains various iWRAP configuration and usage examples. Most of the examples are now available as separate application notes are not any more included in the iWRAP user guide.

## 12.1 Serial Port Profile

Please see a Serial Port Profile application note.

## 12.2 Dial-up Networking

The Dial-Up Networking (DUN) profile allows you for example to connect to phone phones and control their GSM modem with AT commands. The most common use cases for DUN are sending SMS messages or connecting to Internet via GPRS or 3G. The simple below shows how to open a Dial-Up Networking connection to a phone and how to send an AT command to the phone.
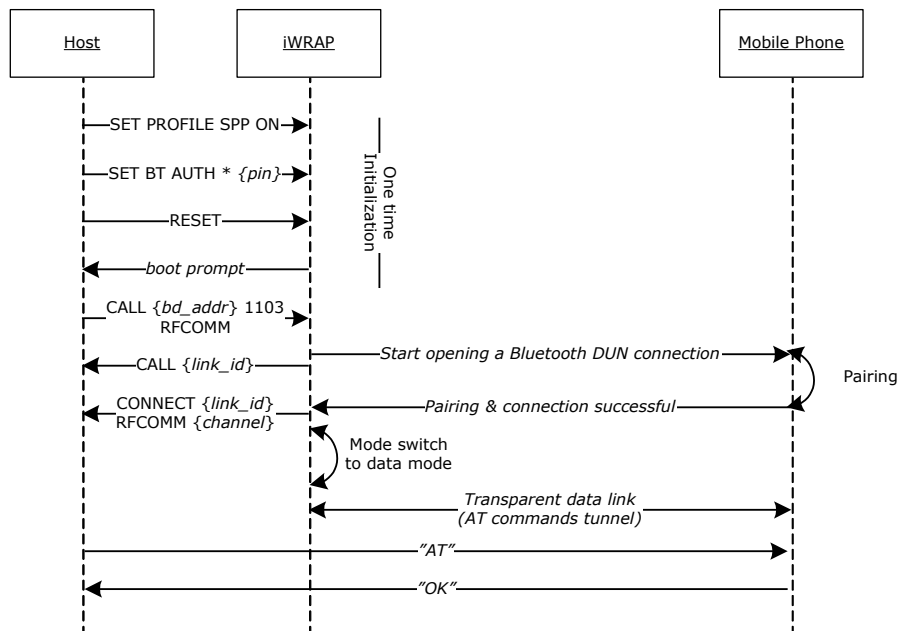


**Figure 11: How to open a DUN connection to a mobile phone**

In iWRAP the Bluetooth code must be set, since most of the mobile phones always require the PIN code authentication, before allowing the Dial-Up Networking connection.

It may be wise to do the pairing from the mobile phone and make the iWRAP module 'trusted'. Once this is done, the phone does not ask for the PIN code every time the connection is opened.

Notice that not all the mobile phones support the same AT commands, since some of the commands are optional and some mandatory.

Refer to the following AT command specifications for more information and examples: 3GPP TS 27.005 and 3GPP TS 07.07.

## 12.3 Hands-Free Audio Gateway Connection to a Headset Device

Please see HFP and HSP profiles application note.

## 12.4 Hands-Free connection to a Mobile Phone

Please see HFP and HSP profiles application note.

## 12.5 Human Interface Device profile example

Please see HID Profile application note.

## 12.6 Wireless IO Replacement

iWRAPs can be used to do wireless IO replacement, that is, to transmit the status of GPIO PINs over the SPP link. This means that if the status of the local IO changes, so does the status of the remote IO. This functionality can be accomplished by using the MSC (Modem Status Control) feature in iWRAP.
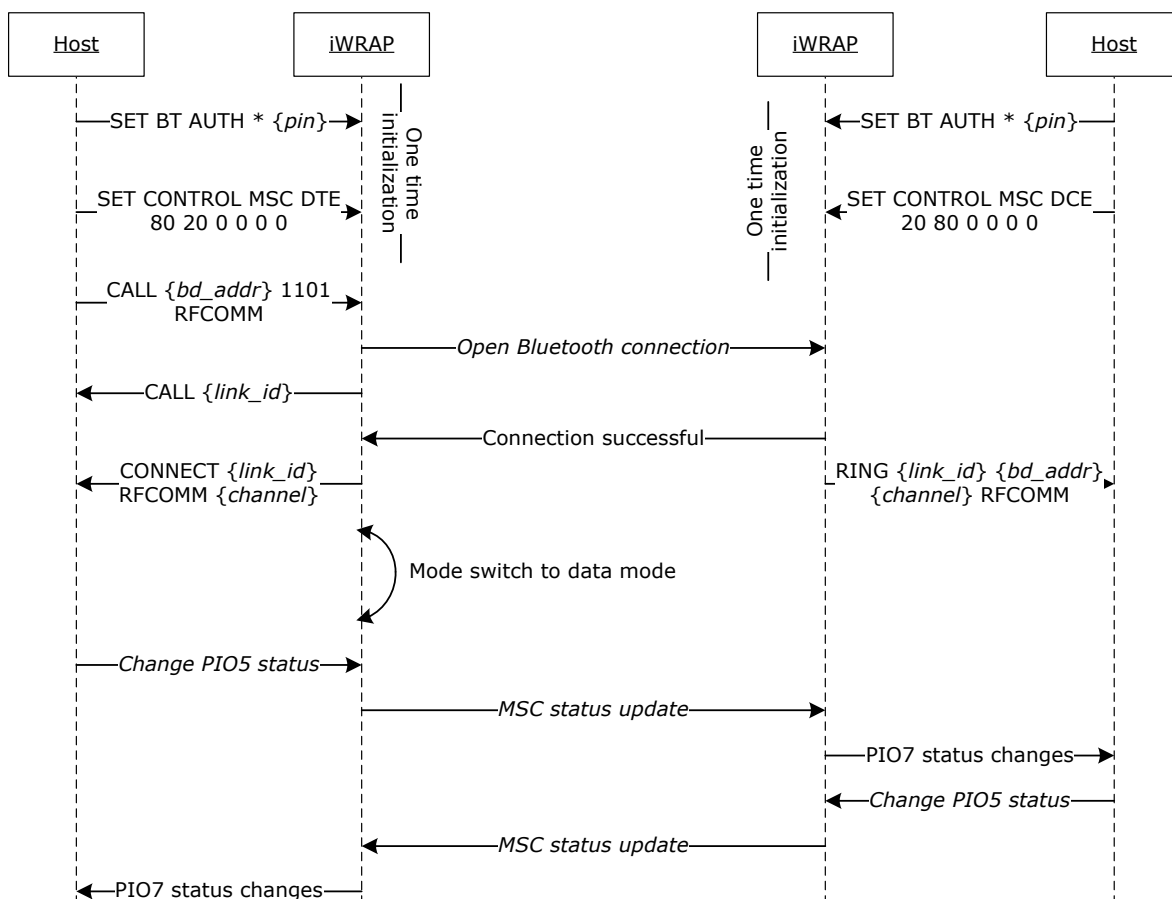


**Figure 12: Wireless IO replacement connection**

The example above was done with WT12 evaluation kits. In the evaluation kit, there is a DSR button in PIO5 and a LED in PIO7. Parameter 80 matches with PIO7 and parameter 20 with PIO5. So whenever DSR button is pressed in the local device, the LED status changes in the remote end.

**NOTE:**

- Switching the IO status very rapidly may reset iWRAP as the GPIO interrupts are handled with low priority. Therefore MSC feature is not feasible for radio GPIO sampling application.

- There is also a delay when transmitting the MSC status over the Bluetooth link. Without power saving in use, this delay is roughly 20ms and if power saving is in use, the delay depends on SNIFF mode parameters.
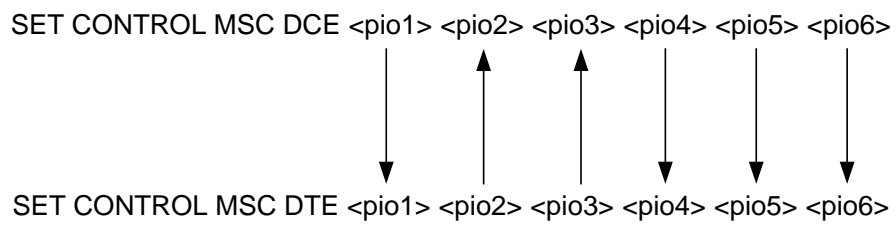
SET CONTROL MSC DCE <pio1> <pio2> <pio3> <pio4> <pio5> <pio6>

SET CONTROL MSC DTE <pio1> <pio2> <pio3> <pio4> <pio5> <pio6>

**Figure 13: MSC signal directions**

## 12.7 A2DP Sink

Please see A2DP and AVRCP application note.

## 12.8 A2DP Source

Please see A2DP and AVRCP application note.

## 12.9 AVRCP Connection

Please see A2DP and AVRCP application note.

## 12.10     Over-the-Air Configuration

iWRAP3 has Over-the-Air (OTA) configuration interface, which allows one to configure iWRAP settings over a *Bluetooth* SPP connection. OTA gives one access to standard iWRAP commands which also available over UART interface. This example shows how OTA interface can be accessed from another iWRAP device.
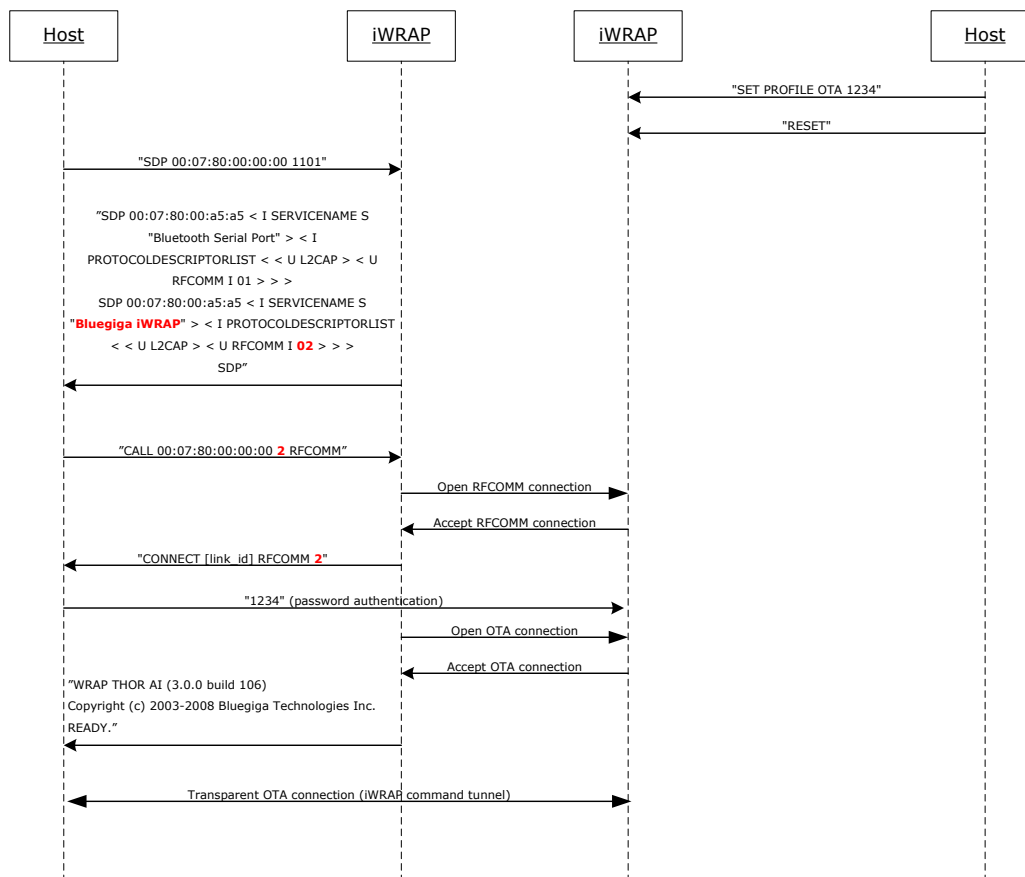


**Figure 14: Over-the-Air connection example**

On a remote iWRAP OTA is simply activated by issuing iWRAP command: **SET PROFILE OTA {*password*}** and by performing a reset.

In the Bluetooth interface OTA is seen as a standard Bluetooth Serial Port Profile service with a fixed service name "***Bluegiga iWRAP***".

When OTA connection is opened the first thing that needs to be done is to send the password from the controlling device to the controlled iWRAP. If the password is correct iWRAP boot prompt will be displayed, otherwise the connection will be closed.

There is a special use case for OTA to remotely read/write the GPIO pins of the iWRAP under control.

# 13 Technical support

- For technical questions and problems, please contact: support@bluegiga.com

- Firmware, parameters, tools and documentation can be downloaded from: http://techforum.bluegiga.com

## 13.1 Sending email to technical support

In case you facing problems with iWRAP firmware, always remember to include the output of "**INFO CONFIG**" command to your email. This way we can replicate the exact setup that you have and solve the problems faster.



```
COM2:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Resize  Help

WRAP THOR AI (3.1.0 build 257)
Copyright (c) 2003-2009 Bluegiga Technologies Inc.
Compiled on Jul 31 2009 10:50:22, running on WT32-A module, psr v255
        A2DP AVRCP BATTERY FTP MICBIAS PBAP PIO=0x07ff SSP SUBRATE VOLUME
        - BOCK3 version 257 (Jul 31 2009 10:50:14) (max acl/sco 7/1)
        - Bluetooth version 2.1, Power class 3
        - Loader 5946, firmware 5946 (56-bit encryption)
        - up 0 days, 00:18, 0 connections (pool 1)
        - User configuration:
&0299 = 0006 0000 0021 0000 0000 0000 0000 0000 0000 0000
&02ac = 0000 0000 002b 0000 0000 0000 0000 0000 0000 0000 0042 0000 0000 0000 0010 0000 0
000 0000 0000 0000 0000 0000 0000 0000
&02ad = 5457 3233 412d 4d2d 6b69 6f6b 312d
READY.
```

**Table 17: INFO CONFIG output**

# 14 Contact information

**Sales:**                    sales@bluegiga.com

**Technical support:**      support@bluegiga.com

http://techforum.bluegiga.com

**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