

# **BLUETOOTH SMART READY**

CONFIGURATION GUIDE

Tuesday, 28 August 2018

Version 2.0



**Copyright © Silicon Labs**

All rights reserved.

Silicon Labs assumes no liability or responsibility for any errors, mistakes or inaccuracies in content. Silicon Labs reserves the right to change products or specifications without notice, and does not make any commitment to update the information herein.

Silicon Labs' products are not authorized for use as critical components in life support devices or systems.

The Bluetooth® word mark and logos are registered trademarks owned by the Bluetooth® SIG, Inc. USA.

All other trademarks and trade names listed herein belong to their respective owners.

Information is subject to change without notice.

# Table of Contents

|       |                                       |    |
|-------|---------------------------------------|----|
| 1     | Version History                       | 4  |
| 2     | Introduction                          | 5  |
| 2.1   | Project structure                     | 5  |
| 2.1.1 | Project file                          | 5  |
| 2.1.2 | Hardware configuration                | 5  |
| 2.1.3 | Bluetooth Smart service database      | 5  |
| 2.1.4 | Bluetooth classic profile SDP entries | 5  |
| 2.1.5 | BGScript application code             | 5  |
| 2.1.6 | Host application code                 | 6  |
| 3     | Project File Syntax                   | 7  |
| 3.1   | <project>                             | 7  |
| 3.2   | <hardware>                            | 7  |
| 3.3   | <gatt>                                | 8  |
| 3.4   | <script>                              | 8  |
| 3.5   | <image>                               | 8  |
| 3.6   | <entry>                               | 9  |
| 3.7   | <library>                             | 10 |
| 3.8   | Examples                              | 10 |
| 4     | Hardware Configuration file           | 12 |
| 4.1   | <adc>                                 | 12 |
| 4.2   | <sleep>                               | 12 |
| 4.3   | <controller_sleep>                    | 13 |
| 4.4   | <wakeup_pin>                          | 13 |
| 4.5   | <port>                                | 14 |
| 4.6   | <uart>                                | 16 |
| 4.7   | <spi>                                 | 18 |
| 4.8   | <i2c>                                 | 20 |
| 4.9   | <host_wakeup>                         | 21 |
| 5     | SPP Configuration file                | 22 |
| 6     | DID Configuration file                | 23 |
| 7     | HID Configuration file                | 24 |

# 1 Version History

| Version | Comments  |
|---------|---|
| 1.0     | First version   |
| 1.1     | Improved hardware.xml syntax documentation  |
| 1.2     | ADC / Sleep pin / I2C parts edited and general edits  |
| 1.3     | SPP and DID sections added  |
| 1.4     | Minor changes   |
| 1.5     | Updated SPP SDP entry documentation   |
| 1.6     | Slave select description removed<br>Host wake-up pin description added<br>Some terminology corrections  |
| 1.7     | Minor updates   |
| 1.8     | HID descriptor file documentation added<br>Project file configuration updated to match 1.1.0 software   |
| 1.9     | UART configuration updated  |
| 1.9.1   | Added "stack" parameter for XML attribute <script><br>Added information that junction temperature measurement is also enabled with the "vdd" parameter of the XML attribute <adc><br>XML attribute <i2c> - Parameters clarified, default values adjusted, and "pullup" parameter added. |
| 2.0     | Added HID descriptor file format info<br><sleep> attribute description update<br>Added <controller_sleep> attribute<br><wakeup_pin> attribute example update<br><host_wakeup> clarification<br>Fixed available pins for attributes  |

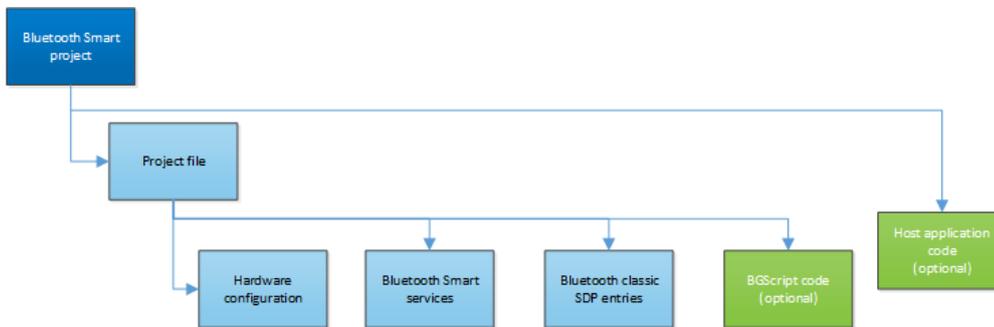
## 2 Introduction

This document walks you through how to start a software project for your BT121 *Bluetooth* Smart Ready module, how to include the necessary resources in the project and also how to do configure the hardware interface settings for the Bluetooth modules.

### 2.1 Project structure

The figure below illustrates the Bluetooth software project structure and the mandatory and optional resource. The structure is relatively simple and consists of the following components:

1. Project file
2. Hardware configuration file
3. Bluetooth Smart service and characteristics database (GATT database)
4. Bluetooth classic profile SDP entries
5. BGScript application source code (optional)
6. Host application source code (optional and exclusive to BGScript code)



#### 2.1.1 Project file

Project file simply defines the resources included in the project and their physical locations.

#### 2.1.2 Hardware configuration

The hardware configuration file defines the host and peripheral interfaces like UART, SPI, I2C and GPIO used by the application and their physical locations (pins) and the settings.

#### 2.1.3 Bluetooth Smart service database

The service database (GATT database) defines the contents and structure of the Bluetooth GATT services and characteristics implemented by the application. The GATT database is defined with the Profile Toolkit™ XML based description language included the Bluetooth SDK.

#### 2.1.4 Bluetooth classic profile SDP entries

The SDP entries defines the contents of the Service Discovery Profile database for Bluetooth classic profiles like Serial Port Profile, Human Interface Device Profile, Apple iAP2 profile or Device Information profile.

#### 2.1.5 BGScript application code

BGScript is a basic-style application scripting language, which allows simple applications to be embedded into the BT121 Bluetooth module. In case BGScript is used to implement the application logic, the source files need to be included in the Bluetooth project file.

## 2.1.6 Host application code

An alternative way to implement the application is to use an additional host (typically a MCU) and use the Bluetooth module as a modem. In this case the application code runs outside the module and source code files do not need to be included in the Bluetooth project, but the architecture selection needs to be defined in the project file.

## 3 Project File Syntax

The project file (typically *project.xml* or *project.bgproj*) is the file that describes all the components included in your *Bluetooth* Smart Ready project. Typically these files are named as follows:

- **hardware.xml** - Hardware configuration file for interfaces like UART and SPI
- **GATT.xml** - GATT database file for *Bluetooth* Smart services and characteristics
- **DID.xml, SPP.xml, HID.xml** etc. - SDP entry file(s) for supported *Bluetooth* profiles
- **script.bgs** - Optional BGScript application source code

The project file also defines other features of the project like the hardware version or the firmware output files.

The project file itself is a simple XML file with only a few elements in it, which are described below.

### 3.1 <project>

The XML attribute *<project>* starts the definition of the project file and also includes the hardware device type the project is meant for. All the other definitions need to be inside the project attribute.

| Parameter  | Description  |
|--|--|
| <i>device</i>  | This parameter defines the hardware type this project is used for.<br>Options:<br><b>bt121</b> |
| <b>Example: Defining the hardware configuration file</b><br><pre>&lt;project device="bt121"&gt;<br/>...<br/>&lt;/project&gt;</pre> |  |

### 3.2 <hardware>

The XML attribute *<hardware>* and its parameter *in* are used to define the hardware configuration file for the device.

| Parameter   | Description  |
|---|--|
| <i>in</i>   | This parameter points to the XML file which contains the hardware configuration definition for your <i>Bluetooth</i> Smart Ready device. |
| <b>Example: Defining the hardware configuration file</b><br><pre>&lt;hardware in="hardware.xml" /&gt;</pre> |  |

### 3.3 <gatt>

The XML attribute <gatt> and its parameter *in* are used to define the GATT database file.

**NOTE!** The GATT definition can also be placed inside the Project XML file.

| Parameter                                       | Description   |
|---|---|
| <i>in</i>                                       | This parameter points to the XML file that contains the GATT database defining the <i>Bluetooth</i> Smart services and characteristics. |
| <b>Example: Defining the GATT database file</b> |   |
| <pre>&lt;gatt in="GATT.xml" /&gt;</pre>         |   |

### 3.4 <script>

The optional XML attribute <script> and its parameter *in* are used to define the BGScript source code file. This XML attribute is placed within the XML attribute pair <scripting> </scripting>

| Parameter  | Description   |
|--|---|
| <i>in</i>  | <p>This parameter points to the BGScript file that contains the BGScript source code for your standalone <i>Bluetooth</i> Smart Ready application.</p> <p>You are allowed also to use at the same time the BGAPI protocol over UART for a host system to control the module and a BGScript for additional standalone functionality. In this case, the commands in the script will generate responses and events which are sent out of the UART as well as BGAPI messages, so for the script to run make sure that the host is ready to receiving such messages.</p> |
| <b>stack</b>   | <p>This parameter sets the size of the script stack. Increasing this value from its default might be necessary to extend the stack if you are receiving large buffers (&gt;150 bytes) into the script. Please note that events having payload of more than 256 bytes are not sent to script. Increasing the stack size reduces the amount of memory available for other operations and might limit for instance the amount of simultaneous connections that the module can handle.</p> <p>Default stack size is 256 bytes.</p>                                      |
| <b>Example: Defining the BGScript file</b>                                       |   |
| <pre>&lt;scripting&gt; &lt;script in="bgdemo.bgs" /&gt; &lt;/scripting&gt;</pre> |   |

### 3.5 <image>

The XML attribute <image> and its parameter *out* are used to define the firmware binary output files.

| Parameter  | Description   |
|------------|---|
| <i>out</i> | <p>This parameter defines the name of the binary firmware output file which the compiler will generate.</p> <p>This parameter will generate a <i>.bin</i> file which can be uploaded to the <i>Bluetooth</i> Smart Ready Module.</p> <p>In newer versions of the firmware a <i>.bootdfu</i> binary file is also created: it contains the new bootloader that might have to be uploaded first when upgrading from older version.</p> |

| Parameter   | Description |
|---|-------------|
| <b>Example: Defining the binary and HEX output files for the compiler</b> |             |
| <code>&lt;image out="BT121_BGDemo.bin" /&gt;</code>                       |             |

### 3.6 <entry>

The XML attributes `<entry>` and their parameters are used to define the actual XML files for each of the *Bluetooth* Classic's SDP static records that you want to include in the firmware.

These XML attributes are grouped within an XML attribute pair `<sdp> </sdp>` which is used to define the *Bluetooth* BR/EDR Service Discover Profile (SDP) entries or *Bluetooth* profiles used by the project.

| Parameter       | Description  |
|-----------------|--|
| <i>file</i>     | This parameter defines the name of the XML file containing a single SDP entry.   |
| <i>autoload</i> | This parameter defines whether the SDP entry should be loaded automatically when the <i>Bluetooth</i> stack starts.<br><br><b>Values:</b><br><br><b>true:</b> SDP entry is automatically loaded into the SDP database<br><br><b>false:</b> SDP entry is NOT automatically loaded into the SDP database   |
| <i>id</i>       | This parameter defines a unique ID for the SDP entry that can then later be used by the application to manually load the SDP record<br><br><div style="border: 1px solid yellow; background-color: #ffffcc; padding: 5px;"> <p> <b>autoload</b></p> <p>If autoload is used then id is not allowed.</p> </div> |

**Example: Defining an SDP entry for the project, which is auto-loaded and another SDP entry with a unique ID = 2**

```
<sdp>
<entry file="DID.xml" autoload="true"/>
<entry file="SPP.xml" id="2"/>
</sdp>
```

### 3.7 <library>

The optional XML attribute `<library>` and its parameter `in` are used to select which variant of the software is to be built. This XML attribute is placed within the XML attribute pair `<software>` `</software>`. This tag is not mandatory, if omitted default image will be produced.

| Parameter   | Description   |
|---|---|
| <i>in</i>   | <p>This parameter points to the base firmware that we want the Smart Ready firmware image for the BT121 to be based upon.</p> <p>Possible firmware files are:</p> <p><b>bt121</b> : default, no HID functionality</p> <p><b>bt121_hid</b> : includes HID functionality, but no support for le_serial commands</p> <p><b>bt121_tiny</b> : no HID functionality and no support for le_serial commands</p> |
| <p><b>Example: Selecting the firmware variant where the HID functionality is enabled</b></p> <pre>&lt;software&gt; &lt;library in="bt121_hid" /&gt; &lt;/software&gt;</pre> |   |

### 3.8 Examples

Typical example:

```
BT121 Project

<?xml version="1.0" encoding="UTF-8" ?>

<!-- Project configuration including BT121 device type -->
<project device="bt121">

  <!-- XML file containing GATT service and characteristic
  definitions both for BLE and GATT over BR -->
  <gatt in="gatt.xml" />

  <!-- Local hardware interfaces configuration file -->
  <hardware in="hardware.xml" />

  <!-- Local SDP entries for Bluetooth BR/EDR -->
  <sdp>
    <entry file="DID.xml" autoload="true"/>
    <entry file="SPP.xml" id="2"/>
  </sdp>

  <!-- Firmware output files -->
  <image out="BT121_BGDemo.bin" />

</project>
```

Below is an example of a project file for BT121 *Bluetooth* Smart Module including a BGScript application:

```
<?xml version="1.0" encoding="UTF-8" ?>

<!-- Project configuration including BT121 device type -->
<project device="bt121">

  <!-- XML file containing GATT service and characteristic
  definitions both for BLE and GATT over BR -->
  <gatt in="gatt.xml" />

  <!-- Local hardware interfaces configuration file -->
  <hardware in="hardware.xml" />

  <!-- Local SDP entries for Bluetooth BR/EDR -->
  <sdp>
    <entry file="DID.xml" autoload="true"/>
    <entry file="SPP.xml" id="2"/>
  </sdp>

  <!-- BGScript source code file -->
  <scripting>
    <script in="bgdemo.bgs" />
  </scripting>

  <!-- Firmware output files -->
  <image out="BT121_BGDemo.bin" />

</project>
```

## 4 Hardware Configuration file

The hardware configuration file is used to configure the hardware features such as TX power, UART, SPI, hardware timers, and GPIO settings of your Bluegiga *Bluetooth* Ready Smart device.

### 4.1 <adc>

The XML attribute <adc> is used to configure the module's ADC (Analog Digital Converter) settings. ADC reference is always VDD.

| Parameter   | Description  |
|---|--|
| <i>inputs</i>   | This bit mask defines which ADC channels are in use.<br><br><b>Values:</b><br><br><b>Bit 4:</b> AIN4<br><br><b>Bit 5:</b> AIN5<br><br><b>Bit 6:</b> AIN6<br><br><b>Bit 7:</b> AIN7   |
| <i>vdd</i>  | Options:<br><br><b>true:</b> Enable Vdd and junction temperature measurement<br><br><b>false:</b> Disable Vdd and junction temperature measurement<br><br><b>Note:</b> Vdd and junction temperature measurement is enabled always if any ADC channel is in use |
| <b>Example to enable all ADCs:</b><br><code>&lt;adc inputs="0xf0" /&gt;</code>  |  |
| <b>Example to enable only Vdd and junction temperature measurement:</b><br><code>&lt;adc inputs="0x0" vdd="true" /&gt;</code> |  |

### 4.2 <sleep>

The XML attribute <sleep> can be used to allow or prevent the use of sleep modes. To use any sleep modes of the module, enable this option both with the <controller\_sleep> attribute.

| Parameter      | Description   |
|----------------|---|
| <i>enabled</i> | <b>Options:</b><br><br><b>true:</b> All power modes can be enabled. Selection of power modes is done automatically by the firmware. Firmware will select the best power saving mode automatically to achieve lowest possible power consumption. Currently the lowest mode is power mode 2.<br><br><b>false:</b> Use this to prevent the firmware from entering any of the sleep modes.<br><br><b>Default:</b><br><br><b>false</b> |

| Parameter                                   | Description |
|---|-------------|
| <b>Example : Allow power saving</b>         |             |
| <code>&lt;sleep enabled="true" /&gt;</code> |             |

#### **sleep**

If you enable the `<sleep>` feature and use UART to communicate with the module you must also enable the `<wakeup_pin>` feature.

### 4.3 `<controller_sleep>`

The XML attribute `<controller_sleep>` can be used to allow or prevent the use of sleep modes of hardware (including using radio module's depth sleep). To use any sleep modes of the module, enable this option both with the `<sleep>` attribute.

| Parameter  | Description   |
|--|---|
| <b><i>enabled</i></b>                                  | <p><b>Options:</b></p> <p><b>true:</b> All hardware's power modes can be enabled. Entering into some power save mode is done automatically by the firmware (same after allowing sleep modes via the <code>&lt;sleep&gt;</code> attribute).</p> <p><b>false:</b> Use this to prevent the hardware from entering any of the sleep modes.</p> <p><b>Default:</b></p> <p><b>false</b></p> |
| <b>Example : Allow hardware's sleep modes</b>          |   |
| <code>&lt;controller_sleep enabled="true" /&gt;</code> |   |

### 4.4 `<wakeup_pin>`

The XML attribute `<wakeup_pin>` can be used to define an input GPIO pin which wakes the module up from a sleep mode or alternatively prevents to *Bluetooth* module from entering a sleep mode. If you have enabled the sleep modes and use UART to communicate with the module, then this feature must also be enabled.

The wake-up pin functionality can only be assigned to a single GPIO, but you can still assign normal GPIO interrupts to other pins. The difference between the wake-up pin and normal GPIO interrupt is that the wake-up pin will not only generate the interrupt which wakes up the module from sleep, but will also keep the module awake as long as it is asserted. Normal GPIO interrupts will wake the module from any state but after the interrupt event handler completes the module will return to sleep.

How to use the wake-up pin:

1. Assert the wake-up pin from an external host and keep it asserted
2. Process the **dumo\_evt\_hardware\_interrupt** event generated by the module (see the API Reference for more details).
3. Send the desired BGAPI command(s) to the module.
4. Wait until you receive the full BGAPI response(s) back from the module
5. De-assert the wake-up pin
6. The module enters sleep mode

### **wake-up pin**

Steps 2 and 4 are critical and must be implemented correctly or otherwise data loss might occur.

| Parameter    | Description   |
|--------------|---|
| <i>port</i>  | Defines the port into which the wake-up pin is to be assigned to.<br><b>Options:</b><br><b>0:</b> Port A<br><b>1:</b> Port B  |
| <i>pin</i>   | Defines the pin of the defined port to which the wake-up pin is to be assigned to.<br><b>Range:</b><br><b>Port A:</b> pins 4-7 and 9-14<br><b>Port B:</b> pins 3-10 and 12-15 |
| <i>state</i> | Logic state for the assigned wake-up pin.<br><b>Options:</b><br><b>up</b><br><b>down</b><br><b>Default:</b><br><b>up</b>  |

**Example: Enabling wake-up pin on PB12 (BTN4 on DKBT Development kit) and defining the state to "up".**

```
<wakeup_pin port="1" pin="12" state="up" />
```

 When this pin is pulled, the *Bluetooth* Smart Ready module does not enter any sleep modes which increases power consumption.

## 4.5 <port>

The XML attribute <port> can be used to define the settings for I/O ports A and B.

The parameters are described in the table below.

| Parameter    | Description  |
|--------------|--|
| <i>index</i> | Index of port to configure.<br><b>Range:</b><br><b>0:</b> Port A<br><b>1:</b> Port B |

| Parameter   | Description  |
|---|--|
| <i>output</i>   | <p>Bit mask to configure which port's pins are outputs. Output pins are set in push-pull mode.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p>  |
| <i>input</i>  | <p>Bit mask to configure which port's pins are inputs.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p>  |
| <i>value</i>  | <p>Bit mask to configure the status (level) of port's output pins after boot.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p>   |
| <i>pullup</i>   | <p>Pull-up configuration (bit mask) for port's input pins. Pins which are not set with this or the below "pulldown" option are left floating.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p> |
| <i>pulldown</i>   | <p>Pull-down configuration (bit mask) for port's input pins. Pins which are not set with this or the above "pullup" option are left floating.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p> |
| <i>interrupts_rising</i>  | <p>Rising interrupt configuration (bit mask) for pins in this port.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p>   |
| <i>interrupts_falling</i>   | <p>Falling interrupt configuration (bit mask) for pins in this port.</p> <p><b>Range:</b></p> <p><b>Port A: pins 4-7 and 9-14</b></p> <p><b>Port B: pins 3-10 and 12-15</b></p>  |
| <p><b>Example : Set PB8 and PB9 as outputs and enable interrupts on PB10 and PB13</b></p> <pre>&lt;port index="1" output="0x300" /&gt; &lt;port index="1" input="0x2400"&gt; &lt;port index="1" interrupts_rising="0x2400"/&gt;</pre> |  |

| Parameter  | Description |
|--|-------------|
| <b>Example : Set PB12 as input and configure pull-down on this input</b> |             |
| <code>&lt;port index="1" input="0x1000" pulldown="0x1000" /&gt;</code>   |             |

## 4.6 <uart>

The XML attribute `<uart>` can be used to define the UART interface settings.

The parameters are explained in the table below.

| Parameter              | Description   |
|------------------------|---|
| <i><b>baud</b></i>     | UART baud rate.<br><br><b>Range:</b><br><b>1200 - 4000000</b><br><br><b>Default:</b><br><b>115200</b>   |
| <i><b>stopbits</b></i> | Number of stop bits.<br><br><b>Options:</b><br>1<br>1.5<br>2<br><br><b>Default:</b><br>1  |
| <i><b>parity</b></i>   | Parity bit setting.<br><br><b>Values:</b><br><b>odd:</b> use odd parity bit<br><b>even:</b> use even parity bit<br><b>none:</b> no parity bit<br><br><b>Default:</b><br><b>none</b><br><br><b>Example:</b><br><code>parity="odd"</code> |

| Parameter          | Description  |
|--------------------|--|
| <i>flowcontrol</i> | <p>UART flow control setting.</p> <p><b>Options:</b></p> <p><b>true:</b> Hardware flow control (RTS and CTS) enabled</p> <p><b>false:</b> Hardware flow control (RTS and CTS) disabled</p> <p><b>Default:</b></p> <p><b>false</b></p> <div style="border: 1px solid #ccc; background-color: #ffffcc; padding: 10px; margin-top: 10px;"> <p> <b>RTS/CTS</b></p> <p>STM32F071 Controller used in BT121 does not have FIFO in the UART and hence will de-assert RTS immediately when SW is not ready to receive more data. In practice the RTS toggles between every byte. Any byte sent while RTS is de-asserted will be lost. See STm document RM0091 chapter 25.5.16.</p> <p><a href="http://www.st.com/content/ccc/resource/technical/document/reference_manual/c2/f8/8a/f2/18/e6/43/96/DM00031936.pdf/files/DM00031936.pdf/jcr:content/translations/en.DM00031936.pdf">http://www.st.com/content/ccc/resource/technical/document/reference_manual/c2/f8/8a/f2/18/e6/43/96/DM00031936.pdf/files/DM00031936.pdf/jcr:content/translations/en.DM00031936.pdf</a></p> </div> |
| <i>bgapi</i>       | <p>Defines if UART is used for BGAPI protocol or BGScript application</p> <p><b>Values:</b></p> <p><b>true:</b> UART is used for BGAPI protocol</p> <p><b>false:</b> UART is used for BGScript application data</p> <p><b>Default:</b></p> <p><b>false</b></p> <p><b>Example:</b></p> <p><i>bgapi="true"</i></p> <div style="border: 1px solid #ccc; background-color: #ffffcc; padding: 10px; margin-top: 10px;"> <p> <b>bgapi</b></p> <p>When this is set to <i>true</i>, there should be an application listening to the UART data. If the UART buffer fills up the firmware execution is halted on the Bluetooth module. If RTS/CTS flow control is not used, then there is no need to read the data from the UART.</p> </div>  |

| Parameter   | Description   |
|---|---|
| <b><i>timeout</i></b>   | <p>This parameter configures the time-out in milliseconds that the module waits between two consecutive bytes received over UART.</p> <p>If UART is in transparent mode (<i>bgapi="false"</i>) and this time-out is reached, then the bytes are sent forward to a BGScript application or to the destination endpoint.</p> <p>If UART is in BGAPI mode (<i>bgapi="true"</i>) and this time-out is reached and unless a full BGAPI command has been received, the module returns a syntax error event to the host and clears the UART buffer.</p> <p><b>Range:</b></p> <p><b>1 - 4000</b></p> <p><b>Default:</b></p> <p>1 (when <i>bgapi=false</i>)</p> <p>1000 (when <i>bgapi=true</i>)</p> |
| <p><b>Example : Enabling BGAPI over UART on BT121 @ 115200bps and with RTS/CTS flow control</b></p> <pre>&lt;uart baud="115200" flowcontrol="true" bgapi="true" /&gt;</pre> |   |
| <p><b>Example : Enabling BGAPI over UART on BT121 @ 115200bps for BGScript usage</b></p> <pre>&lt;uart baud="115200" flowcontrol="true" bgapi="false" /&gt;</pre>           |   |

## 4.7 <spi>

The XML attribute <spi> can be used to define the module's SPI configuration settings.

The parameters are explained in the table below.

| Parameter               | Description  |
|-------------------------|--|
| <b><i>channel</i></b>   | <p>Defines the SPI channel to configure.</p> <p><b>Values:</b></p> <p>1: SPI channel 1</p> <p>2: SPI channel 2</p> <p><b>Example:</b></p> <pre>channel="2"</pre>   |
| <b><i>alternate</i></b> | <p>Defines the alternate pin configuration option for SPI.</p> <p><b>Values:</b></p> <p>1: Alternative configuration 1 (Alt 1) (see data sheet for details)</p> <p>2: Alternative configuration 2 (Alt 2) (see data sheet for details)</p> <p><b>Default:</b></p> <p>1</p> |

| Parameter                  | Description   |
|----------------------------|---|
| <i>divisor</i>             | <p>Defines the SPI divisor used for the clock in master mode.</p> <p>Bitrate is 48MHz / (<i>divisor</i>)</p> <p><b>Values:</b></p> <p><b>2,4,8,16,32,64,128,256</b></p> <p><b>Default:</b></p> <p><b>2</b></p> <p><b>Example:</b></p> <p><i>divisor="16"</i></p>  |
| <i>mode</i>                | <p>Defines the SPI mode as master or slave.</p> <p><b>Values:</b></p> <p><b>master:</b> Use SPI as master</p> <p><b>slave:</b> Use SPI as slave</p> <p><b>Default:</b></p> <p><b>master</b></p> <p><b>Example:</b></p> <p><i>mode="master"</i></p>  |
| <i>clock_idle_polarity</i> | <p>Defines the logic level used when SPI clock is in idle state.</p> <p><b>Values:</b></p> <p><b>low:</b> Idle state for clock is a low level; active state is a high level.</p> <p><b>high:</b> Idle state for clock is a high level; active state is a low level.</p> <p><b>Default:</b></p> <p><b>low</b></p> <p><b>Example:</b></p> <p><i>clock_idle_polarity="low"</i></p> |
| <i>clock_edge</i>          | <p>Defines the SPI clock edge.</p> <p><b>Values:</b></p> <p><b>0:</b> Serial output data changes on transition from idle clock state to active clock state.</p> <p><b>1:</b> Serial output data changes on transition from active clock state to idle clock state.</p> <p><b>Default:</b></p> <p><b>0</b></p> <p><b>Example:</b></p> <p><i>clock_edge="0"</i></p>               |

| Parameter         | Description  |
|-------------------|--|
| <i>endianness</i> | Defines the SPI bit order.<br><br><b>Options:</b><br><br><b>msb:</b> most significant bit<br><br><b>lsb:</b> least significant bit |

**Example: Configure SPI interface settings for the display on the DKBT Development kit:**

```
<spi channel="1" alternate="2" clock_idle_polarity="high" clock_edge="1" endianness="msb" divisor="256" />
```

## 4.8 <i2c>

The XML attribute <i2c> can be used to define the module's I2C (Inter-Integrated Circuit) interface configuration.

 Bitrate is calculated as  **$8MHz/prescaler/divider = bitrate$** .

*This bitrate is not accurate due clock syncing etc. Please see processor's reference manual for details (ST RM0091)*

 If you select I2C *channel 2* you may only use *Alt 2* setting, *Alt 1* setting is not allowed. For details see module data sheet.

| Parameter        | Description   |
|------------------|---|
| <i>channel</i>   | Defines and enables the I2C channel to configure.<br><br><b>Values:</b><br><br>1: I2C channel 1<br><br>2: I2C channel 2   |
| <i>alternate</i> | Defines the alternate configuration option for I2C.<br><br><b>Options:</b><br><br>1 : Alternative configuration 1 (Alt 1) (see data sheet for details)<br><br>2 : Alternative configuration 2 (Alt 2) (see data sheet for details)<br><br><b>Default:</b><br><br>For channel 1 alternate default is 1<br><br>For channel 2 alternate default is 2 and only possible value |
| <i>prescaler</i> | Defines the pre-scaler for baud rate generator.<br><br><b>Range:</b><br><br>1-16<br><br><b>Default:</b><br><br>2  |

| Parameter   | Description   |
|---|---|
| <i>divider</i>  | Defines the divider for baud rate generator.<br><br><b>Range:</b><br><b>1-256</b><br><br><b>Default:</b><br><b>40</b> |
| <b>pullup</b>   | Pull-up configuration for the SDA,<br><br><b>Default:</b><br><b>True</b>  |
| <b>Example: Enabling I2C</b><br><code>&lt;i2c channel="1" alternate="1" prescaler="2" divider="40" /&gt;</code> |   |

## 4.9 <host\_wakeup>

This XML element `<host_wakeup>` can be used to wake up the host processor when the module is about to send events or data over the UART to host. Host wake up pin is guaranteed to stay up as long as there are more events to be sent to host but not to the end of the event.

| Parameter   | Description   |
|---|---|
| <i>port</i>   | Defines the port to which the wake-up pin is to be assigned to.<br><br><b>Options:</b><br><b>0: Port A</b><br><b>1: Port B</b>  |
| <i>pin</i>  | Defines the pin of the defined port to which the wake-up pin is to be assigned to.<br><br><b>Range:</b><br><b>Port A: pins 4-7 and 9-14</b><br><b>Port B: pins 3-10 and 12-15</b> |
| <b>Example: Configuring wake-up pin on PB13 (BTN5 on DKBT Development kit).</b><br><code>&lt;host_wakeup port="1" pin="13" /&gt;</code> |   |

## 5 SPP Configuration file

For *Bluetooth* BR/EDR profile the SDP entries also need to be configured so the profiles are properly advertised to remote devices. The SDP entries for all desired profiles must be defined in the project configuration file.

In addition the one XML file per profile must also be included in actual project and these XML files are used to configure profile based settings.

Below is an example showing the user configurable options for the Serial Port Profile (SPP) XML file.

| Contents  | Description   |
|---|---|
| <pre>&lt;ServiceClassIDList&gt; &lt;ServiceClass uuid128="1101"/&gt; &lt;/ServiceClassIDList&gt;</pre>  | <p>This defines the UUID of the <i>Bluetooth</i> profile. For <i>Bluetooth</i> Serial Port Profile the UUID must be 1101 and <b>should not be changed</b>.</p>  |
| <pre>&lt;BrowseGroupList&gt; &lt;UUID16 value="1002"/&gt; &lt;/BrowseGroupList&gt;</pre>  | <p>This section defines if this SDP entry is visible in the SDP browse group. Typically you should not change this, but for some special applications you might want to disable the browse group visibility.</p>  |
| <pre>&lt;ProtocolDescriptorList&gt; &lt;Protocol&gt; &lt;UUID16 value="0100"/&gt; &lt;/Protocol&gt; &lt;Protocol&gt; &lt;UUID16 value="03"/&gt; &lt;UINT8 value="05"/&gt; &lt;/Protocol&gt; &lt;/ProtocolDescriptorList&gt;</pre> | <p>value="0100" means this profile is based on top of RFCOMM.<br/> value="03" means the next parameter defines the assigned RFCOMM channel<br/> value="05" defines the RFCOMM channel assigned for the profile</p> <div style="border: 1px solid black; background-color: #ffffcc; padding: 5px; margin-top: 10px;"> <p> You can only change the RFCOMM channel number and keep rest unchanged.</p> </div> |
| <pre>&lt;ServiceName&gt; text=" <b>Bluetooth Serial Port</b>" language_id="0100" &lt;/ServiceName&gt;</pre>   | <p>This defines the service name for the given UUID. If you want to rename the service you can modify the <i>Bluetooth Serial Port</i> to contain something else.</p>   |

## 6 DID Configuration file

This mandatory SDP entry defines the so called **Device Information Profile**, the attributes of which describe certain characteristics of the module such as **Vendor ID**, **Product ID**, **Version** etc. For the **Device Information Profile** there is a corresponding XML file, named *DID.xml* in the project configuration file.

The DID configuration file itself is a simple XML file consisting of nested structured elements, their attributes and attribute values.

| Contents  | Description   |
|---|---|
| <pre>&lt;UINT16 value=" 0200"/&gt;  &lt;UINT16 value=" 0103"/&gt;</pre> | <p><b><u>This MUST not be changed.</u></b></p>  |
| <pre>&lt;UINT16 value=" 0201"/&gt; &lt;UINT16 value=" 0047"/&gt;</pre>  | <p><b>0201</b> refers to vendor ID parameter and you can change the <b>0047</b> to your own vendor ID if you have one assigned from USB Implementers Forum or <i>Bluetooth</i> SIG.</p> <p>If you do not have your own vendor ID you can keep using <b>0047</b> <b><u>unless you are making MFI compliant devices in which can you must have your own ID.</u></b></p> |
| <pre>&lt;UINT16 value=" 0202"/&gt; &lt;UINT16 value=" 1234"/&gt;</pre>  | <p><b>0202</b> refers to product ID parameter and if you have decided to use your own vendor ID you can also use your own product ID as well and change <b>1234</b> to something else.</p> <p>If case you are using the default vendor ID, this value must not be changed.</p>  |
| <pre>&lt;UINT16 value=" 0203"/&gt; &lt;UINT16 value=" 0000"/&gt;</pre>  | <p><b>0202</b> refers to product version and you can replace the value <b>0000</b> with your own version number.</p>  |
| <pre>&lt;UINT16 value=" 0204"/&gt; &lt;UINT16 value="1"/&gt;</pre>      | <p><b><u>This MUST not be changed.</u></b></p>  |
| <pre>&lt;UINT16 value=" 0205"/&gt; &lt;UINT16 value=" 0001"/&gt;</pre>  | <p><b>0205</b> refers to the source of the vendor ID and it must tell if your own vendor ID is from <i>Bluetooth</i> SIG or USB Implementers Forum</p> <p><b>0000</b>: Source of vendor ID is USB Implementers Forum<br/> <b>0001</b>: Source of vendor ID is <i>Bluetooth</i> SIG</p>  |

## 7 HID Configuration file

Below is an example showing the user configurable options for the Human Interface Devices (HID) XML file.

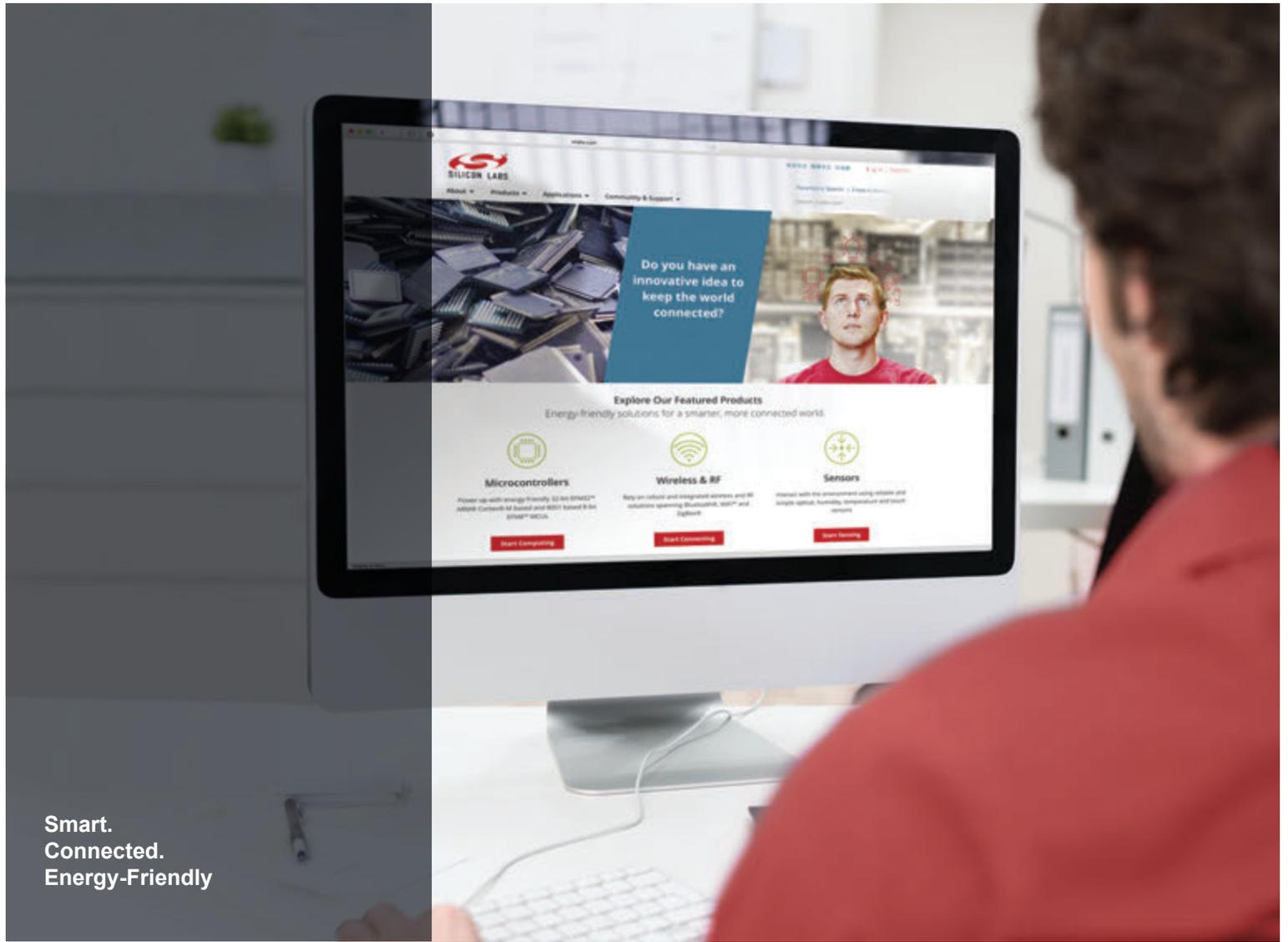
Note that the "autoload" attribute cannot be used in the project.xml with HID SDP records.

| Contents   | Description  |
|--|--|
| <pre>&lt;ServiceClassIDList&gt; &lt;ServiceClass uuid128="1124"/&gt; &lt;/ServiceClassIDList&gt;</pre>   | <p>This defines the UUID of the <i>Bluetooth</i> profile. For the HID profile the UUID must be 1124.</p> <div style="border: 1px solid black; background-color: #ffffcc; padding: 5px;">  This configuration should not be changed.         </div>  |
| <pre>&lt;BrowseGroupList&gt; &lt;UUID16 value="1002"/&gt; &lt;/BrowseGroupList&gt;</pre>   | <p>This section defines if this SDP entry is visible in the SDP browse group. Typically you should not change this, but for some special applications you might want to disable the browse group visibility.</p>   |
| <pre>&lt;ProtocolDescriptorList&gt; &lt;Protocol&gt; &lt;UUID16 value="0100"/&gt; &lt;UINT16 value="0011"/&gt; &lt;/Protocol&gt; &lt;Protocol&gt; &lt;UUID16 value="0011"/&gt; &lt;/Protocol&gt; &lt;/ProtocolDescriptorList&gt;</pre> | <p>value="0100" means this profile is based on top of L2CAP</p> <p>The first value="0011" refers to the PSM for HID Control</p> <p>The second value="0011" refers to the Protocol Identifier's UUID</p> <div style="border: 1px solid black; background-color: #ffffcc; padding: 5px;">  This configuration should not be changed.         </div> |
| <pre>&lt;ServiceName text="BT121 Mouse" language_id="0100"/&gt;</pre>  | <p>This entry defines the service name for the SDP record. If you want to rename the service you can modify the value of the text= attribute</p>   |
| <pre>&lt;LanguageBaseAttributeIDList&gt; &lt;UINT16 value="656e"/&gt; &lt;UINT16 value="006a"/&gt; &lt;UINT16 value="0100"/&gt; &lt;/LanguageBaseAttributeIDList&gt;</pre>   | <p>value="656e" is for "en" - English</p> <p>value="006a" is for UTF-8 encoding</p> <p>value="0100" is to define PrimaryLanguageBaselid = 0</p>  |
| <pre>&lt;BluetoothProfileDescriptorList&gt; &lt;Profile&gt; &lt;UUID16 value="0011"/&gt; &lt;UINT16 value="0101"/&gt; &lt;/Profile&gt; &lt;/BluetoothProfileDescriptorList&gt;</pre>   | <p>value="0011" refers to the Protocol Identifier's UUID for the HID profile</p> <p>value="0101" is to define the version to 1.1</p> <div style="border: 1px solid black; background-color: #ffffcc; padding: 5px;">  This configuration should not be changed.         </div>  |

| Contents   | Description   |
|--|---|
| <pre> &lt;AdditionalProtocolDescriptorLists&gt; &lt;AdditionalProtocolDescriptorList&gt; &lt;Protocol&gt; &lt;UUID16 value="0100"/&gt; &lt;UINT16 value="0013"/&gt; &lt;/Protocol&gt; &lt;Protocol&gt; &lt;UUID16 value="0011"/&gt; &lt;/Protocol&gt; &lt;/AdditionalProtocolDescriptorList&gt; &lt; /AdditionalProtocolDescriptorLists&gt; </pre> | <p>value="0100" means this profile is based on top of L2CAP</p> <p>value="0013" refers to the PSM for HID Interrupt</p> <p>value="0011" refers to the Protocol Identifier's UUID for the HID profile</p> <div data-bbox="598 376 1453 465" style="border: 1px solid black; background-color: #ffffcc; padding: 5px;">  This configuration should not be changed. </div>                            |
| <pre> &lt;HIDParserVersion value="0111"/&gt; </pre>  | <p>The current Bluetooth HID specification fixes this value to 0x0111.</p> <div data-bbox="598 929 1453 1019" style="border: 1px solid black; background-color: #ffffcc; padding: 5px;">  This configuration should not be changed. </div>   |
| <pre> &lt;HIDDeviceSubclass value="80"/&gt; </pre>   | <p>The value of this field must match the bits 2 to 7 in the Bluetooth Class of Device. Bits 0 and 1 must be set to zero.</p> <p>Common values are 0x40 for a keyboard and 0x80 for a pointing device.</p> <p>For a comprehensive list see of values see tables 9-10 (Minor Device Class field - Peripheral Major Class) at <a href="https://www.bluetooth.com/specifications/assigned-numbers/baseband">https://www.bluetooth.com/specifications/assigned-numbers/baseband</a></p> |
| <pre> &lt;HIDCountryCode value="0"/&gt; </pre>   | <p>Country code value.</p> <p>Set to 0x00 for non-localized devices.</p> <p>For localized devices such as keyboards, see the section 6.2.1 in USB country code list at <a href="http://www.usb.org/developers/hidpage/HID1_11.pdf">http://www.usb.org/developers/hidpage/HID1_11.pdf</a>.</p> <p>For example for localized US keyboard set the the value to:<br/> <i>&lt;HIDCountryCode value="21"/&gt;</i></p>   |
| <pre> &lt;HIDVirtualCable value="1"/&gt; </pre>  | <p>This value indicates whether the HID device should be associated with only one host at a time, like a wired keyboard can be connected to only one computer.</p> <p>Enabling this means your device should never store the pairing information of more than one host at a time</p> <p>* If enabled, your device MUST also support either HIDReconnectInitiate or HIDNormallyConnectable</p>   |

| Contents   | Description  |
|--|--|
| <HIDReconnectInitiate value="1"/>  | <p>This value indicates whether the HID Device can reconnect to the HID Host.</p> <p><b>Values:</b></p> <p><b>1:</b> Reconnection possible</p> <p><b>0:</b> Reconnection not possible</p>  |
| <pre>&lt;HIDDescriptorList&gt; &lt;HIDClassDescriptor&gt; &lt;UINT8 value="22"/&gt; &lt;HIDUSBDescriptor file="mouse.txt" /&gt; &lt;/HIDClassDescriptor&gt; &lt;/HIDDescriptorList&gt;</pre> | <p>This is used to define the actual HID descriptor.</p> <p>value="22" indicates a Report Descriptor</p> <p>mouse.txt contains the actual HID descriptor. The descriptors are pre-defined in the USB HID specification and are not listed in the HIDDescriptorList.</p> <div data-bbox="598 705 1452 891" style="border: 1px solid #c8e6c9; padding: 10px; margin: 10px 0;"> <p> The easiest way to create and validate HID descriptor is to use the USB HID Descriptor Tool (<a href="http://www.usb.org/developers/hidpage#HID%20Descriptor%20Tool">http://www.usb.org/developers/hidpage#HID%20Descriptor%20Tool</a>) and export the descriptor into a TXT file.</p> </div> <div data-bbox="598 929 1452 1288" style="border: 1px solid #c8e6c9; padding: 10px; margin: 10px 0;"> <p> Format of descriptor TXT file should be like below:</p> <p>[comments] [{TAB} [hex1 hex2 hex3 ...] [EOL]]</p> <ul style="list-style-type: none"> <li>- Use UTF-8 or ASCII encoding.</li> <li>- Keep in mind that everything on a line preceding a TAB character will be ignored.</li> <li>- Lines without a TAB character will be ignored.</li> </ul> </div> |
| <pre>&lt;HIDLANGIDBaseList&gt; &lt;HIDLANGIDBase&gt; &lt;UINT16 value="0409"/&gt; &lt;UINT16 value="0100"/&gt; &lt;/HIDLANGIDBase&gt; &lt;/HIDLANGIDBaseList&gt;</pre>                       | <p>HIDLANGIDBaseList</p> <p>value="0409" is for en-US</p> <p>value="0100" is the Bluetooth String Offset</p>   |
| <HIDBatteryPower value="1"/>   | <p>Indicates if the device is battery powered or not.</p> <p><b>Values:</b></p> <p><b>1:</b> Device is battery powered</p> <p><b>0:</b> Device is not battery powered</p>  |

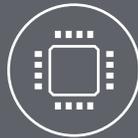
| Contents   | Description  |
|--|--|
| <code>&lt;HIDRemoteWake value="1"/&gt;</code>          | <p>Indicates if the device can wake up the host from suspend if it is supported by the host. This requires two things:</p> <ol style="list-style-type: none"> <li>1. The HID Device can send an Exit Suspend command upon user input, if the Host doesn't disconnect the Bluetooth link while in Suspend mode.</li> <li>2. Reconnect upon user input, if the Host disconnects the Bluetooth link when entering Suspend mode.</li> </ol> <p><b>Values:</b></p> <p><b>1:</b> Device can wake up the host</p> <p><b>0:</b> Device cannot wake up the host</p> |
| <code>&lt;HIDNormallyConnectable value="0"/&gt;</code> | <p>Indicates whether the device normally accepts incoming connections from the Host. Generally for battery-powered devices this should be false, because scanning for paging consumes battery power.</p> <p><b>Values:</b></p> <p><b>1:</b> Device normally accepts incoming connections</p> <p><b>0:</b> Device normally does not accept incoming connections</p>   |
| <code>&lt;HIDBootDevice value="1"/&gt;</code>          | <p>Indicates whether the HID device implements either the Boot Keyboard or Boot Mouse, or both. This is mandatory to support for keyboards and mice devices.</p> <p><b>Values:</b></p> <p><b>1:</b> Device implements Boot Keyboard or Boot Mouse, or both.</p> <p><b>0:</b> Device does not implement Boot Keyboard or Boot Mouse.</p>  |



Smart.  
Connected.  
Energy-Friendly



**Products**  
[www.silabs.com/products](http://www.silabs.com/products)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support and Community**  
[community.silabs.com](http://community.silabs.com)

**Disclaimer**  
Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are not designed or authorized for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

**Trademark Information**  
Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR®, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOModem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.  
400 West Cesar Chavez  
Austin, TX 78701  
USA

<http://www.silabs.com>