UG496: BT122 Project Configuration User's Guide

This document walks you through how to start a software project for your BT122 Bluetooth Dual Mode module, how to include the necessary resources in the project and also how to configure the hardware interface settings for the Bluetooth modules.

KEY FEATURES

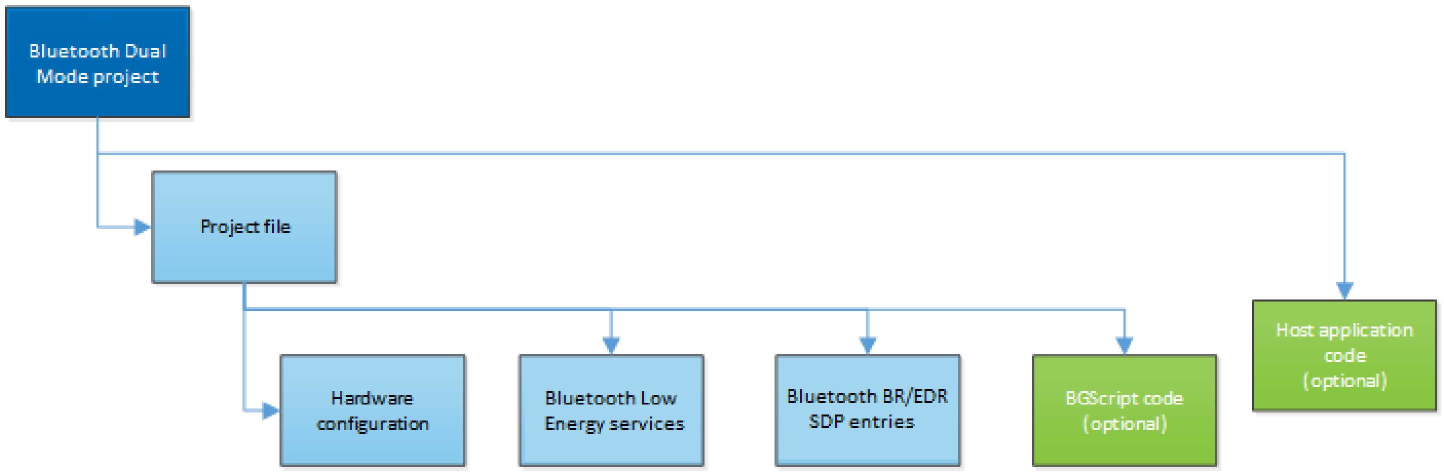
- BT122 Project structure
- Precise description of attributes and syntax used inside project files.

Table of Contents

1. Project Structure

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

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


Bluetooth Dual Mode project
Project file
Hardware configuration
Bluetooth Low Energy services
Bluetooth BR/EDR SDP entries
BGScript code (optional)
Host application code (optional)

1.1 Project File

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

1.2 Hardware Configuration

The hardware configuration file defines the host and peripheral interfaces (like UART, I^2C and GPIO) used by the application and their settings.

1.3 Bluetooth Low Energy 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™ — an XML-based description language

1.4 Bluetooth BR/EDR Profile SDP Entries

The SDP entries define the content of the Service Discovery Profile database for *Bluetooth BR/EDR* profiles like Serial Port Profile, Human Interface Device Profile, Apple iAP2 profile, or Device Information Profile.

1.5 BGScript Application Code

BGScript is a basic-style application scripting language, which allows simple applications to be embedded into the *BT122 Bluetooth Dual Mode* module. When BGScript is used to implement the application logic, the source file needs to be included in the Bluetooth project file.

1.6 Host Application Code

An alternative way to implement the application is to use an additional host (typically an MCU) and use the Bluetooth module as a modem. In this case, the application code runs outside the module and the 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.

2. Project File Syntax

The project file (typically **project.xml**) describes all the components included in your *Bluetooth Dual Mode* project. Typically, these files are named as follows:

- **hardware.xml** – Hardware configuration file for interfaces like UART and I^{2}C
- **GATT.xml** – GATT database file for Bluetooth Low Energy services and characteristics configuration
- **DID.xml, SPP.xml, HID.XML, etc.** – SDP entry file(s) for possible Bluetooth profiles
- **script.bgs** – Optional BGScript application source code

The project file also defines other settings 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.

2.1 <project>

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

Parameter	Description
device	This parameter defines the hardware type this project is used for. **Options**: • **bt122**
Example: Defining the project configuration file including device type	

```
<project device="bt122">
  ...
</project>
```

2.2 <hardware>

The XML attribute <*hardware*> with its *in* parameterdefines the hardware configuration of the device.

Parameter	Description
in	This parameter points to the XML file which contains the hardware configuration definition for your *Bluetooth Dual Mode* device.
Example: Defining the hardware configuration file	

```
<hardware in=”hardware.xml”/>
```

2.3 <gatt>

The XML attribute <*gatt*> with its *in* parameter defines the GATT database file.

Note: The GATT definition can also be placed inside Project XML file.

Parameter	Description
in	This parameter points to the XML file which contains the GATT database defining the *Bluetooth Low Energy* services and characteristics.
Example: Defining the GATT database file	

```
<gatt in=”gatt.xml” />
```

2.4 <script>

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

Parameter	Description
in	This parameter points to the BGScript file that contains the BGScript source code for your standalone *Bluetooth Dual Mode* application. It is also possible to use the BGAPI protocol over UART at the same time by the host system to control the module and a BGScript for additional standalone functionality. In this case, the commands executed in the script will generate respons-es and events which will be sent through UART in BGAPI messages format. To allow such approach, the user should make sure that the host is ready to receive messages like that.
stack	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 expecting to receive large buffers (higher than 150 bytes) into the script. However, by doing so, the amount of memory available for other operations will be reduced and might limit for instance the maximum number of simultaneous connections that the module can handle. The default stack size is 256 bytes. Note: Events having payload of more than 256 bytes are not sent to script.

Example: Defining the BGScript file

```
<scripting>
  <script in=”bgdemo.bgs” />
</scripting>
```

2.5 <image>

The XML attribute <*image*> and its *in* parameter define the firmware binary output file.

Parameter	Description
out	This parameter defines the name of the binary firmware output file which the compiler will generate. This parameter will generate a *.bin* file that can be uploaded to the *Bluetooth Dual Mode* module.

Example: Defining the binary output file for the compiler

```
<image out=”BT122_BGDemo.bin” />
```

2.6 <entry>

The XML attribute <*entry*> and its parameters are used to define the actual XML files for each of the Bluetooth BR/EDR 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 Discovery Profile (SDP) entries or Bluetooth profiles used by the project.

Parameter	Description
file	This parameter defines the name of the XML file containing a single SDP entry.
autoload	This parameter defines whether the SDP entry should be loaded automatically when the *Bluetooth* stack starts. **Values:** • **true**:SDP entry is automatically loaded into the SDP database. • **false**:SDP entry is **not** automatically loaded into the SDP database.
id	This parameter defines a unique ID for the SDP entry that can later be used by the application to manually load the SDP record. Note: If autoload is used then *id* is not allowed.

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>
```

2.7 <library>

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

Parameter	Description
in	This parameter points to the base firmware that we want the *Bluetooth Dual Mode* firmware image for the *BT122* to be based upon. Possible firmware files are: • **bt122**: default, no HID functionality. • **bt122_hid**: includes HID functionality, but no support for le_serial commands.

Example: Selecting the firmware variant where the HID functionality is enabled

```
<software>
  <library in=”bt122_hid” />
</software>
```

2.8 Examples

2.8.1 Basic Project File for BT122 Bluetooth Dual Mode Module without a BGScript Application:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <!-- Project configuration including BT122 device type -->

  <!-- XML file containing GATT service and characteristic definitions both for BLE and GATT over BR/EDR-->
  <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="BT122_BGDemo.bin"/>
</project>
```

2.8.2 Basic Project File for BT122 Bluetooth Dual Mode Module Including a BGScript Application

```
<?xml version="1.0" encoding="UTF-8" ?>
  <!-- Project configuration including BT122 device type -->
  <project device="bt122">

  <!-- 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="BT122_BGDemo.bin"/>
</project>
```

3. Hardware Configuration File

The hardware configuration file is used to configure the hardware features such as TX power, UART, hardware timers, and GPIO setting of your Silicon Labs *Bluetooth Dual Mode* device.

3.1 <adc>

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

Parameter	Description
inputs	This bitmask defines which ADC channels are in use. **Values:** • **bit 0**: AIN0 • **bit 1**: AIN1 • **bit 2**: AIN2
vdd	**Options**: • **true**: enable VDD and junction temperature measurement. • **false**: disable VDD and junction temperature measurement. Note: VDD and junction temperature measurement is always enabled if any of ADC channels is in use.

Example: Enable all available ADCs

```
<adc inputs="0x07"/>
```

Example: Enable only VDD and junction temperature measurement

```
<adc inputs="0x0" vdd="true" />
```

3.2 <sleep>

The XML attribute *<sleep>* can be used to allow or prevent the use of deeper sleep modes. In order to enable deep sleep modes, please make sure that this option is enabled together with *<controller_sleep>* attribute. Otherwise, no sleep will be enabled.

Parameter	Description
enabled	**Options**: • **true**: all power modes are enabled unless the *<controller_sleep>* is also enabled. The firmware will select the best power-saving mode automatically to achieve the lowest possible power consumption. Currently, the lowest mode is power mode EM2. • **false**: prevents the module from entering deep sleep mode. **Default**: **false**

Example: Allow power saving

```
<sleep enabled="true" />
```

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

3.3 <controller sleep>

The XML attribute <*sleep*> can be used to allow or prevent the use of sleep modes.

Parameter	Description
enabled	**Options**: • **true**: allows the radio controller to enter sleep modes. The radio chip automatically asks the upper layers to enter sleep after being idle for certain amount of time. • **false**: prevents the module from entering any of the sleep modes. **Default**: **false**

Example: Allow hardware's sleep modes

```
<controller_sleep enabled=”true” />
```

3.4 <wakeup pin>

The XML attribute <*wakeup_pin*> can be used to define an input GPIO that wakes the module from a sleep mode or alternatively prevents the Bluetooth module from entering a sleep mode. If sleep modes have been enabled and the UART interface is used for communication with the module, this feature must 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 if it is asserted. Normal GPIO interrupts will wake the module form 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 Manual 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.

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

Parameter	Description
port	Defines the port into which the wake-up pin is to be assigned. **Options**: • **0**: Port A • **1**: Port C • **2**: Port F
pin	Defines the pin of the defined port which the wake-up pin is to be assigned to. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
state	Logic state for the assigned wake-up pin. **Options**: • **up** • **down** **Default**: **up**
Example: Enabling wake-up pin on PF3 (BTN0 on BT122 Development Board) and defining the state to "up"	

```
<wakeup_pin port=”2” pin=”3” state=”up” />
```

Note: When this pin is pulled up, the *Bluetooth Dual Mode* module does not enter any of the sleep modes which increases power consumption.

3.5 <port>

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

The parameters are described in the table below.

Parameter	Description
index	Index of a port to configure. **Range**: • **0**: Port A • **1**: Port C • **2**: Port F
output	Bit mask to configure which port's pins are outputs. Output pins are set in push-pull mode. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
input	Bit mask to configure which port's pins are inputs. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
value	Bit mask to configure the status (level) of port's output pins after boot. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
pullup	Pull-up configuration (bit mask) for port's input pins. Pins which are not set with this or the below "pull-down" option are left floating. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
pulldown	Pull-down configuration (bit mask) for port's input pins. Pins which are not set with this or the above "pull-up" option are left floating. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
interrupts_rising	Rising interrupt configuration (bit mask) for pins in this port. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5

Parameter	Description
interrupts_falling	Falling interrupt configuration (bit mask) for pins in this port. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5

Example: Set PF0 and PF1 as outputs and enable interrupts on PF2 and PF3

```
<port index=”2” output=”0x0003” />
```

```
<port index=”2” input=”0x000C” />
```

```
<port index=”2” interrupts_rising=”0x000C” />
```

Example: Set PF3 and input and configure pull-down on this input

```
<port index=”2” input=”0x0008” pulldown=”0x0008” />
```

3.6 <uart>

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

The parameters are explained in the table below.

Parameter	Description
baud	UART baud rate. **Range**: • **1200-3000000** **Default**: **115200**
stopbits	Number of stop bits. **Options**: • **0.5** • **1** • **1.5** • **2** **Default**: **1**
parity	Parity bit setting. **Values**: • **odd**: use odd parity bit • **even**: use even parity bit • **none**: no parity bit **Default**: **none**
flowcontrol	UART flow control setting. **Options**: • **true**: Hardware flow control (RTS and CTS) enabled • **false**: Hardware flow control (RTS and CTS) disabled **Default**: **false**
bgapi	Defines if UART is used for BGAPI protocol or BGScript application. **Values**: • **true**: UART is used for BGAPI protocol • **false**: UART is used for BGScript application data **Default**: **false** Note: When this is set to *true*, 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.
timeout	This parameter configures the timeout (in milliseconds) that the module waits between two consecutive bytes received over UART. If UART is in transparent mode (bgapi="false") and this timeout is reached, then the bytes are sent forward to a BGScript application or to the destination endpoint. If UART is in BGAPI mode (bgapi="true") and this timeout is reached and unless a full BGAPI command has been re-ceived, the module sends a syntax error event to the host and clears the UART buffer. **Range**: • **1-4000** **Default**: **1** (when bgapi="false") or **1000** (when bgapi="true")

Parameter	Description
mode	USART operation mode. **Values**: • **uart**: USART operation mode • **spi_master**: USART is configured as SPI master **Default**: **uart**
polarity	Clock polarity. **Values**: • **negative**: clock is Low when inactive (CPOL = 0) • **positive**: clock is High when inactive (CPOL = 1) **Default**: **negative**
phase	Clock phase. **Values**: • **0**: data is valid on clock Leading edge (CPHA = 0) • **1**: data is valid on clock Trailing edge (CPHA = 1) **Default**: **0**
endianness	Transmission bit order. **Values**: • **msb**: most significant bit first (MSB) • **lsb**: least significant bit first (LSB) **Default**: **msb**

Example: Enabling BGAPI to use UART on BT122 @ 115200bps with RTS/CTS flow control

```
<uart baud=”115200” flowcontrol=”true” bgapi=”true” />
```

Example: Enabling BGScript application to use UART on BT122 @ 115200bps with RTS/CTS flow control

```
<uart baud=”115200” flowcontrol=”true” bgapi=”false” />
```

Example: Enabling UART "spi master" mode, 115200bps, clock is low when inactive, data is valid on clock leading edge, most significant bit first

```
<uart mode="spi_master" polarity="negative" phase="0" endianness="msb" baud="115200" />
```

3.7 <i2c>

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

Parameter	Description
bitrate	Sets I2C bitrate. **Range**: • **7800 - 1048576 [b/s]** **Default**: **100000** **Note**: Recommended values are: 100000 b/s, 400000 b/s and 1000000 b/s.

Example: Enabling I2C

```
<i2c bitrate=”100000” />
```

3.8 <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 a host. Host wake up pin is guaranteed to stay up if there are more events to be sent to a host but not the end of the event.

Parameter	Description
port	Defines the port to which the wake-up pin is to be assigned. **Options**: • **0:** Port A • **1:** Port C • **2:** Port F
pin	Defines the pin of the defined port to which the wake-up pin is to be assigned. **Range**: • **port A**: pins 0-1 • **port C**: pins 8-9 • **port F**: pins 0-5
Example: Configuring wake-up pin on PF2 (BTN1 on BT122 Development Board) `<host_wakeup port="2" pin="2"/>`	

3.9 <bridging>

The XML attribute <*bridging*> Bluetooth Low Energy can be used to allow or prevent the possibility of using bridging between serial connection and *BR/EDR* RFCOMM connection.

Parameter	Description
enabled	**Options**: • **true:** bridging is possible. • **false:** bridging is turned off. **Default**: **false**
Example: Allow bridging between LE Serial and RFCOMM connections `<bridging enabled="true" />`	

Note: Enabling bridging with this attribute only allows or prevents the possibility of using this feature. To make sure it is going to work properly you still need to use proper endpoint routing in BGScript, and you need to run the API command at the BLE side to set the maximum MTU parameter value to 50 using gatt_set_max_mtu(50).

3.10 <lfxo>

The XML attribute <*lfxo*> can be used to configure external Low Frequency Crystal Oscillator.

Parameter	Description
tune	This parameter can be used to adjust load capacitance ladder to achieve the best performance of clocking source. **Range: 0x00-0x7F**
Example: Setting custom tuning value to 0x34 `<lfxo tune="0x34" />`	

Note: If custom value is not set, device uses tuning value from flash, written at manufacturing process.

4. 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

`<ServiceClassIDList>`
`  <ServiceClass uuid128=”1101” />`
`</ServiceClassIDList>`
	This defines the UUID of the *Bluetooth* profile. For *Bluetooth* Serial Port Profile, the UUID must be 1101.

`<BrowseGroupList>`
`  <UUID16 value=”1002”/>`
`</BrowseGroupList>`
	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 dis-able the browse group visibility.

`<ProtocolDescriptorList>`
`  <Protocol>`
`    <UUID16 value=”0100”/>`
`  </Protcol>`
`  <Protocol>`
`    <UUID16 value=”03”/>`
`    <UINT8 value=”05”/>`
`  </Protcol>`
`</ProtocolDescriptorList>`
	Value=”0100” means this profile is based on top of RFCOMM.
	
	Value=”03” means the next parameter defines the assigned RFCOMM channel.
	
	Value=”05” defines the RFCOMM channel assigned for the profile.
	
	Note: You can only change the RFCOMM channel number and keep rest un-changed.

`<ServiceName text=”Bluetooth Serial Port” language_id=”0100” />`	This defines the service name for the given UUID. If you want to rename the serv-ice, you can modify the **Bluetooth Serial Port**to contain something else.

5. DID Configuration File

This mandatory SDP entry defines the **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 configuration file.

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

Contents	Description

`<UINT16 value=”0200” />`
`<UINT16 value=”0103” />`
	<u>This MUST not be changed!</u>

`<UINT16 value=”0201” />`
`<UINT16 value=”0047” />`
	Value **0201** refers to vendor ID parameter and you can change the **0047** to your own vendor ID if you have one assigned from USB Implementers Forum or *Bluetooth* SIG.
	
	If you do not have your own vendor ID you can keep using **0047** **<u>unless you are making MFI compliant devices in which case you must have your own ID.</u>**

`<UINT16 value=”0202” />`
`<UINT16 value=”1234” />`
	Value **0202** 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 **1234** to something else.
	
	In case you are using the default vendor ID, this value must not be changed.

`<UINT16 value=”0203” />`
`<UINT16 value=”0000” />`
	Value **0203** refers to product version and you can replace the value **0000** with your own version number.

`<UINT16 value=”0204” />`
`<UINT16 value=”1” />`
	<u>This MUST not be changed!</u>

`<UINT16 value=”0205” />`
`<UINT16 value=”0001” />`
	Value **0205** refers to the source of the vendor ID, and it must tell if your own vendor ID is form *Bluetooth* SIG or USB Implementers Forum.
	
	0000: Source of vendor ID is USB Implementers Forum
	
	0001: Source of vendor ID is *Bluetooth* SIG

6. 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.

Note: Attribute “autoload” cannot be used in the *project.xml* with HID SDP records.

Contents	Description

`<ServiceClassIDList>`
`  <ServiceClass uuid128=”1124” />`
`</ServiceClassIDList>`
	This defines the UUID of the *Bluetooth* profile. For the HID profile the UUID must be **1124**.
	
	Note: This configuration should not be changed.

`<BrowseGroupList>`
`  <UUID16 value=”1002”/>`
`</BrowseGroupList>`
	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.

`<ProtocolDescriptorList>`
`  <Protocol>`
`    <UUID16 value=”0100”/>`
`    <UUID16 value=”0011”/>`
`  </Protcol>`
`  <Protocol>`
`    <UUID16 value=”0011”/>`
`  </Protcol>`
`</ProtocolDescriptorList>`
	Value **0100** means this profile is based on top of L2CAP.
	
	First value **0011** refers to the PSM for HID Control.
	
	Second value **0011** refers to the Protocol Identifier’s UUID.
	
	Note: This configuration should not be changed.

`<ServiceName text=”BT122 Mouse"`
`language_id=”0100” />`
	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.

`<LanguageBaseAttributeIDList>`
`  <UINT16 value=”656e”/>`
`  <UINT16 value=”006a”/>`
`  <UINT16 value=”0100”/>`
`</LanguageBaseAttributeIDList>`
	Value **656e** is for “en” – English.
	
	Value **006a** is for UTF-8 encoding.
	
	Value **0100** is to define PrimaryLangaugeBaseId = 0

`<BluetoothProfileDescriptorList>`
`  <Profile>`
`    <UUID16 value=”1124”/>`
`    <UINT16 value=”0100”/>`
`  </Profile>`
`</BluetoothProfileDescriptorList>`
	Value **1124** refers to the Service UUID for the HID profile.
	
	Value **0100** is to define the version of HID specification v1.0.

`<AdditionalProtocolDescriptorLists>`
`  <AdditionalProtocolDescriptorList>`
`    <Protocol>`
`      <UINT16 value=”0100”/>`
`      <UINT16 value=”0013”/>`
`    </Protocol>`
`    <Protocol>`
`      <UUID16 value=”0011”/>`
`    </Protocol>`
`  </AdditionalProtocolDescriptorList>`
`</AdditionalProtocolDescriptorLists>`
	Value **0100** means this profile is based on top of L2CAP.
	
	Value **0013** refers to the PSM for HID Interrupt.
	
	Value **0011** refers to the Protocol Identifier’s UUID for the HID profile.
	
	Note: This configuration should not be changed.

`<HIDParserVersion value=”0111” />`	The current *Bluetooth* HID specification fixes this value to 0x0111. **Note**: This configuration should not be changed.
`<HIDDeviceSubclass value=”80” />`	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. Common values are 0x40 for a keyboard and 0x80 for pointing device. For a comprehensive list see of values see tables 9-10 (Minor Device Class field – Peripheral Major Class) at bluetooth.com

Contents	Description
`<HIDCountryCode value="0" />`	Indicates a country code value. Set to 0x00 for non-localized devices. For localized devices such as keyboards, see the section 6.2.1 in USB country code list (usb.org) For example for localized US keyboard set the value to **21**.
`<HIDVirtualCable value="1" />`	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. Enabling this means your device should never store the pairing information of more than one host at a time. **Note**: If enabled, your device MUST also support either HIDReconnectInitiate or HIDNormallyConnectable
`<HIDReconnectInitiate value="1" />`	This value indicates whether the HID device can reconnect to the HID Host. **Values**: • **0**: reconnection not possible • **1**: reconnection possible

`<HIDDescriptorList>`
`  <HIDClassDescriptor>`
`    <UINT8 value="22" />`
`    <HIDUSBDescriptor file="mouse.txt" />`
`  </HIDClassDescriptor>`
`</HIDDescriptorList>`
	This is used to define the actual HID descriptor.
	
	Value **22** indicates a Report Descriptor.
	
	File **mouse.txt** contains the actual HID descriptor. The descriptors are predefined in the USB HID specification and are not listed in the HIDDescriptorList.
	
	Note: The easiest way to create and validate HID descriptor is to use the HID Descriptor Tool (usb.org) and export the descriptor into a TXT file.
	
	Note: Format of descriptor TXT file should be like below:
	
	[comments][{TAB} [hex1 hex2 hex3 …] [EOL]]
	• Use UTF-8 or ASCII encoding.
	• Keep in mind that everything on a line preceding a TAB character will be ignored.
	• Lines without TAB character will be ignored.

`<HIDLANGIDBaseList>`
`  <HIDLANGIDBase>`
`    <UINT16 value="0409"/>`
`    <UINT16 value="0100"/>`
`  </HIDLANGIDBase>`
`</HIDLANGIDBaseList>`
	Value **0409** is for en-US.
	
	Value **0100** is the Bluetooth String Offset.

`<HIDBatteryPower value="1" />`	Indicates if the device is battery powered or not. **Values:** • **0**: device is not battery powered • **1**: device is battery powered

Contents	Description
`<HIDRemoteWake value=”1” />`	Indicates if the device can wake up the host from suspend if it is supported by host. This requires two following things: • HID device can send and Exit Suspend command upon user input in cases where the Host does not disconnect the Bluetooth link while in Suspend mode. • Reconnect upon user input in cases where the Host disconnects the Bluetooth link when entering Suspend mode. **Values:** • **0**: device cannot wake up the Host • **1**: device can wake up the Host
`<HIDNormallyConnectable value=”0” />`	Indicates whether the device normally accepts incoming connections from the Host. Generally, for battery-powered devices this should be false because scanning for paging increases power consumption. **Values:** • **0**: device normally does not accept incoming connections • **1**: device normally accepts incoming connections
`<HIDBootDevice value=”1” />`	Indicates whether the HID device implements either the Boot Keyboard or Boot Mouse, or both. This is mandatory to support keyboards and mice devices. **Values:** • **0**: device does not implement Boot Keyboard or Boot Mouse • **1**: device implements Boot Keyboard or Boot Mouse
`<HIDProfileVersion value=”0100” />`	Indicates a version number of used *Bluetooth* HID specification that the device was designed to. Value **0100**shows that device was designed for HID v1.0.0 specification.
`<HIDDeviceReleaseNumber value=”0100” />`	Indicates a device release number. This attribute is meant to differentiate between versions of products with identical Vendor IDs and Product IDs. The value of the field is JJMN for version JJ.M.N (JJ – major version number, M – minor version number, N – sub-minor version number).

7. Revision History

Revision 1.3

October, 2023

- Updated the I2C bit rate range.

Revision 1.2

July, 2022

- Added new <uart> element attributes.

Revision 1.1

October, 2021

- Added <lfxo> option in hardware.xml
- Updated <library> options with HCI mode firmware example

Revision 1.0

- Initial release

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com