BLUETOOTH DUAL MODE

CONFIGURATION GUIDE

Wednesday, 25 November 2020

Version 2.2

Table of Contents

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 Host wake-up pin description added Some terminology corrections
1.7	Minor updates
1.8	HID descriptor file documentation added 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> Added information that junction temperature measurement is also enabled with the "vdd" parameter of the XML attribute <adc> XML attribute <i2c> - Parameters clarified, default values adjusted, and "pullup" parameter added.
2.0	Added HID descriptor file format info <sleep> attribute description update Added <controller_sleep> attribute <wakeup_pin> attribute example update <host_wakeup> clarification Fixed available pins for attributes
2.1	Added <bridging> that enables/disables possible bridging between leserial and rfcomm
2.2	Renamed "Smart Ready" to "Dual Mode" and "Classic" to "BR/EDR" according to the official Bluetooth SIG nomenclature

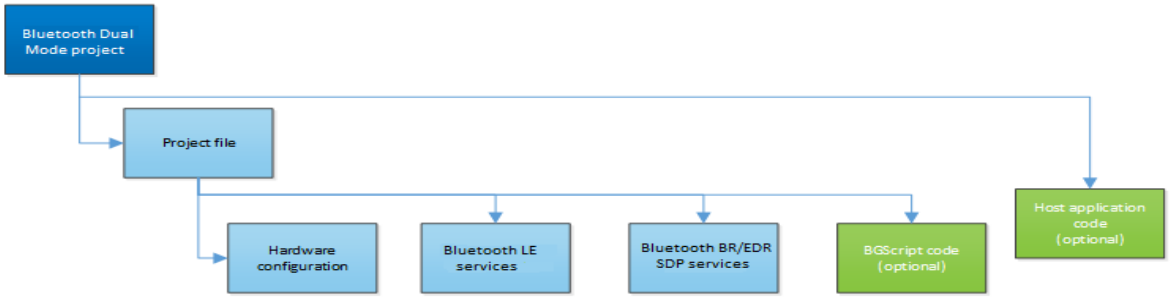
2 Introduction

This document walks you through how to start a software project for your BT121 *Bluetooth Dual Mode* 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 LE 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)



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 LE 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 BR/EDR profile SDP entries

The SDP entries defines the contents 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.

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 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* Dual Mode 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 LE* 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
device	This parameter defines the hardware type this project is used for. Options: **bt121**
Example: Defining the hardware configuration file *<project device="bt121">* *...* *</project>*	

3.2 <hardware>

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

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
in	This parameter points to the XML file that contains the GATT database defining the *Bluetooth LE* services and characteristics.
Example: Defining the GATT database file *<gatt in="GATT.xml" />*	

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
in	This parameter points to the BGScript file that contains the BGScript source code for your standalone *Bluetooth Dual Mode* application. 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.
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 receiving large buffers (>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. Default stack size is 256 bytes.
Example: Defining the BGScript file *<scripting>* *<script in="bgdemo.bgs" />* *</scripting>*	

3.5 <image>

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

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 which can be uploaded to the *Bluetooth* Dual Mode Module. In newer versions of the firmware a .bootdfu binary file is also created: it contains the new bootloader that might have to be uploaded first when upgrading from older version.

Parameter	Description
Example: Defining the binary and HEX output files for the compiler *<image out="BT121_BGDemo.bin" />*	

3.6 <entry>

The XML attributes *<entry>* and their parameters are used to define the actual XML files for each of the *Bluetooth* BR/EDR'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
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 then later be used by the application to manually load the SDP record ⚠ **autoload** 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>*	

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
in	This parameter points to the base firmware that we want the Dual Mode firmware image for the BT121 to be based upon. Possible firmware files are: **bt121** : default, no HID functionality **bt121_hid** : includes HID functionality, but no support for le_serial commands **bt121_tiny** : no HID functionality and no support for le_serial commands

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

<software>
<library in="bt121_hid" />
</software>

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* Dual Mode Module including a BGScript application:

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>

    <!-- 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* Dual Mode 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
inputs	This bit mask defines which ADC channels are in use. **Values:** **Bit 4**: AIN4 **Bit 5**: AIN5 **Bit 6**: AIN6 **Bit 7**: AIN7
vdd	Options: **true:** Enable Vdd and junction temperature measurement **false:** Disable Vdd and junction temperature measurement **Note:** Vdd and junction temperature measurement is enabled always if any ADC channel is in use
Example to enable all ADCs: *<adc inputs="0xf0" />* ***Example to enable only Vdd and junction temperature measurement:*** *<adc inputs="0x0" vdd="true" />*	

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
enabled	**Options:** **true:** 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. **false:** Use this to prevent the firmware from entering any of the sleep modes. **Default:** **false**

Parameter	Description
Example : Allow power saving *<sleep enabled="true" />*	

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
enabled	**Options:** **true:** 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 <sleep> attribute). **false:** Use this to prevent the hardware from entering any of the sleep modes. **Default:** **false**
Example : Allow hardware's sleep modes *<controller_sleep enabled="true" />*	

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
port	Defines the port into which the wake-up pin is to be assigned to. **Options:** **0:** Port A **1:** Port B
pin	Defines the pin of the defined port to which the wake-up pin is to be assigned to. **Range:** **Port A:** pins 4-7 and 9-14 **Port B:** pins 3-10 and 12-15
state	Logic state for the assigned wake-up pin. **Options:** **up** **down** **Default:** **up**
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* Dual Mode 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
index	Index of port to configure. **Range:** **0:** Port A **1:** Port B

Parameter	Description
output	Bit mask to configure which port's pins are outputs. Output pins are set in push-pull mode. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
input	Bit mask to configure which port's pins are inputs. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
value	Bit mask to configure the status (level) of port's output pins after boot. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
pullup	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. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
pulldown	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. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
interrupts_rising	Rising interrupt configuration (bit mask) for pins in this port. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
interrupts_falling	Falling interrupt configuration (bit mask) for pins in this port. **Range:** **Port A: pins 4-7 and 9-14** **Port B: pins 3-10 and 12-15**
Example : Set PB8 and PB9 as outputs and enable interrupts on PB10 and PB13 *<port index="1" output="0x300" /> <port index="1" input="0x2400"/> <port index="1" interrupts_rising="0x2400"/>*	

Parameter	Description
Example : Set PB12 as input and configure pull-down on this input *<port index="1" input="0x1000" pulldown="0x1000" />*	

4.6 <uart>

Th e 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 - 4000000** **Default:** **115200**
stopbits	Number of stop bits. **Options:** **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** **Example:** *parity="odd"*

Parameter	Description
flowcontrol	UART flow control setting. **Options:** **true:** Hardware flow control (RTS and CTS) enabled **false:** Hardware flow control (RTS and CTS) disabled **Default:** **false** **RTS/CTS** STM32F071 Controller used in BT121 does not have FIFO in the UART and hence will de-assert RTS immediatedly 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. 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
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** **Example:** *bgapi="true"* **bgapi** 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.

Parameter	Description
timeout	This parameter configures the time-out in milliseconds that the module waits between two consecutive bytes received over UART. If UART is in transparent mode (*bgapi="false"*) and this time-out 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 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. **Range:** **1 - 4000** **Default:** **1** (when bgapi=false) **1000** (when bgapi=true)
Example : Enabling BGAPI over UART on BT121 @ 115200bps and with RTS/CTS flow control *<uart baud="115200" flowcontrol="true" bgapi="true" />*	
Example : Enabling BGAPI over UART on BT121 @ 115200bps for BGScript usage *<uart baud="115200" flowcontrol="true" bgapi="false" />*	

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
channel	Defines the SPI channel to configure. **Values:** **1:** SPI channel 1 **2:** SPI channel 2 **Example:** *channel="2"*
alternate	Defines the alternate pin configuration option for SPI. **Values:** **1:** Alternative configuration 1 (Alt 1) (see data sheet for details) **2:** Alternative configuration 2 (Alt 2) (see data sheet for details) **Default:** **1**

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

Parameter	Description
endianness	Defines the SPI bit order. **Options:** **msb:** most signigicant bit **lsb:** 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
channel	Defines and enables the I2C channel to configure. **Values:** **1:** I2C channel 1 **2:** I2C channel 2
alternate	Defines the alternate configuration option for I2C. **Options:** **1 :** Alternative configuration 1 (Alt 1) (see data sheet for details) **2 :** Alternative configuration 2 (Alt 2) (see data sheet for details) **Default:** For channel 1 alternate default is 1 For channel 2 alternate default is 2 and only possible value
prescaler	Defines the pre-scaler for baud rate generator. **Range:** **1-16** **Default:** **2**

Parameter	Description
divider	Defines the divider for baud rate generator. **Range:** **1-256** **Default:** **40**
pullup	Pull-up configuration for the SDA, **Default:** **True**
Example: Enabling I2C *<i2c channel="1" alternate="1" prescaler="2" divider="40" />*	

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

4.10 <bridging>

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

Parameter	Description

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" />*	

Enabling bridging with this attribute only allow or prevent the possibility of use of this feature. To make sure it is going to work properly you still need to use proper endpoint routing in BG script and you need to run the API command at the BLE side to set maximum MTU parameter value to 50 - gatt_set_max_mtu(50).

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
<ServiceClassIDList> <ServiceClass uuid128="1101"/> </ServiceClassIDList>	This defines the UUID of the *Bluetooth* profile. For *Bluetooth* Serial Port Profile the UUID must be 1101 and **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" /> </Protocol> <Protocol> <UUID16 value="03"/> <UINT8 value="05"/> </Protocol> </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 ⚠ You can only change the RFCOMM channel number and keep rest unchanged.
<ServiceName> text="***Bluetooth Serial Port***" language_id="0100" </ServiceName>	This defines the service name for the given UUID. If you want to rename the service you can modify the ***Bluetooth Serial Port*** to contain something else.

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
<UINT16 value=" 0200"/> <UINT16 value=" 0103"/>	**This MUST not be changed.**
<UINT16 value=" 0201"/> <UINT16 value=" 0047"/>	**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** **unless you are making MFI compliant devices in which can you must have your own ID.**
<UINT16 value=" 0202"/> <UINT16 value=" 1234"/>	**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. If case you are using the default vendor ID, this value must not be changed.
<UINT16 value=" 0203"/> <UINT16 value=" 0000"/>	**0202** refers to product version and you can replace the value **0000** with your own version number.
<UINT16 value=" 0204"/> <UINT16 value="1"/>	**This MUST not be changed.**
<UINT16 value=" 0205"/> <UINT16 value=" 0001"/>	**0205** refers to the source of the vendor ID and it must tell if your own vendor ID is from *Bluetooth* SIG or USB Implementers Forum **0000**: Source of vendor ID is USB Implementers Forum**0001**: Source of vendor ID is *Bluetooth* SIG

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
<ServiceClassIDList> <ServiceClass uuid128="1124"/> </ServiceClassIDList>	This defines the UUID of the *Bluetooth* profile. For the HID profile the UUID must be 1124. ⚠ 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"/> <UINT16 value="0011"/> </Protocol> <Protocol> <UUID16 value="0011"/> </Protocol> </ProtocolDescriptorList>	value="0100" means this profile is based on top of L2CAP The first value="0011" refers to the PSM for HID Control The second value="0011" refers to the Protocol Identifier's UUID ⚠ This configuration should not be changed.
<ServiceName text="BT121 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 PrimaryLanguageBaseId = 0
<BluetoothProfileDescriptorList> <Profile> <UUID16 value="0011"/> <UINT16 value="0101"/> </Profile> </BluetoothProfileDescriptorList>	value="0011" refers to the Protocol Identifier's UUID for the HID profile value="0101" is to define the version to 1.1 ⚠ This configuration should not be changed.

Contents	Description
<AdditionalProtocolDescriptorLists> <AdditionalProtocolDescriptorList> <Protocol> <UUID16 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 ⚠ This configuration should not be changed.
<HIDParserVersion value="0111"/>	The current Bluetooth HID specification fixes this value to 0x0111. ⚠ 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 a pointing device. For a comprehensive list see of values see tables 9-10 (Minor Device Class field - Peripheral Major Class) at https://www.bluetooth.com/specifications/assigned-numbers/baseband
<HIDCountryCode value="0"/>	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 at http://www.usb.org/developers/hidpage/HID1_11.pdf . For example for localized US keyboard set the the value to: *<HIDCountryCode value="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 * If enabled, your device MUST also support either HIDReconnectInitiate or HIDNormallyConnectable

Contents	Description
<HIDReconnectInitiate value="1"/>	This value indicates whether the HID Device can reconnect to the HID Host. **Values:** **1:** Reconnection possible **0:** Reconnection not 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 mouse.txt contains the actual HID descriptor. The descriptors are pre-defined in the USB HID specification and are not listed in the HIDDescriptorList. The easiest way to create and validate HID descriptor is to use the USB HID Descriptor Tool (http://www.usb.org/developers/hidpage#HID%20Descriptor%20Tool) and export the descriptor into a TXT file. 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 a TAB character will be ignored.
<HIDLANGIDBaseList> <HIDLANGIDBase> <UINT16 value="0409"/> <UINT16 value="0100"/ </HIDLANGIDBase> </HIDLANGIDBaseList>	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:** **1:** Device is battery powered **0:** Device is not battery powered

Contents	Description
<HIDRemoteWake value="1"/>	Indicates if the device can wake up the host from suspend if it is supported by the host. This requires two things: 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. 2. Reconnect upon user input, if the Host disconnects the Bluetooth link when entering Suspend mode. **Values:** **1:** Device can wake up the host **0:** Device cannot 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 consumes battery power. **Values:** **1:** Device normally accepts incoming connections **0:** Device normally does not accept 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 for keyboards and mice devices. **Values:** **1:** Device implements Boot Keyboard or Boot Mouse, or both. **0:** Device does not implement Boot Keyboard or Boot Mouse.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

http://www.silabs.com