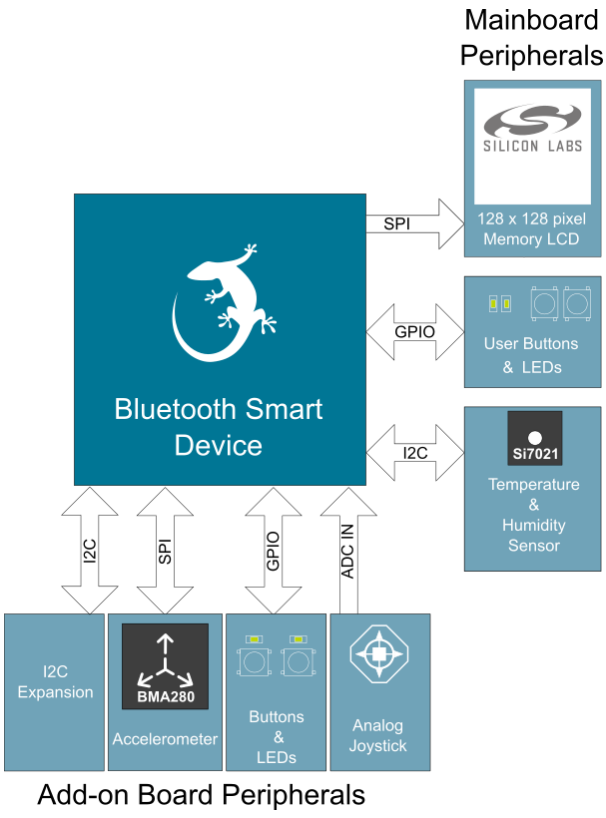
UG119: Blue Gecko *Bluetooth*® Device Configuration Guide

This document describes how to start a software project for your Blue Gecko Bluetooth devices, how to include the necessary resources in to the project, and also how to configure the hardware interface settings for the Blue Gecko Bluetooth devices.

This document is valid only for BGScript™ and Network Co-Processor (NCP) developers. Developers who write Bluetooth C applications for the Wireless Geckos should refer to the C developer documentation.

Note: Beginning with Bluetooth SDK 2.4.0, released in June 2017, BGScript is no longer supported. All applications must be developed using C. Please refer to *UG136: Silicon Labs Bluetooth® C Application Developers Guide* for further info and to *AN1042: Using the Silicon Labs Bluetooth® Stack in Network Co-Processor Mode* for specific details on configuring NCP applications

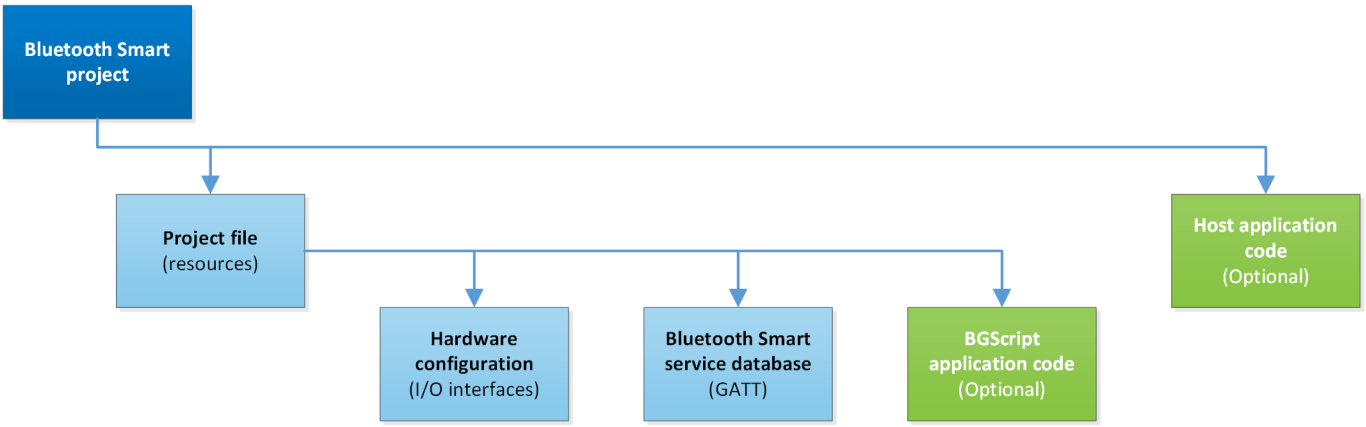
KEY POINTS
- Starting a BGScript or NCP software project
- Including necessary resources
- Configuring hardware interface settings



1. Project Structure

The flowchart below illustrates the structure of a Bluetooth software project, including the required mandatory and optional resources. The Project structure is relatively simple and consists of the following components:

- Project file
- Hardware configuration file
- Bluetooth service and characteristics database (GATT database)
- BGScript application code (optional and exclusive to host application code)
- Host application code (optional and exclusive to BGScript code)



1.1 Project File

The project file simply defines the resources included in the project and their physical locations. The project file may also be used to define the maximum number of simultaneous connections and to define the name and type of the generated firmware files etc.

1.2 Hardware Configuration File

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

1.3 Bluetooth Service Database

The Bluetooth service database (GATT database) file is an XML-based file prepared by the user (e.g. by using XML editor or Bluetooth Developer Studio) and defines the content and structure of the Bluetooth GATT services and characteristics implemented by the application. The Bluetooth SDK compiles the contents of this XML file and embeds the result into the firmware image.

Guidelines and syntax for defining the GATT database using XML can be found in *UG118: Blue Gecko Bluetooth Profile Toolkit Developer's Guide*.

1.4 BGScript™ Application Code

BGScript is a BASIC-style event-driven application programming language, which allows simple applications to be embedded into Bluetooth devices. If BGScript is used to implement the application logic, the source files need to be included in the project file.

1.5 Host Application Code

An alternative to using BGScript-based application code is to use an external host (typically an MCU), and use the Bluetooth device as a modem or in Network Co-Processor (NCP) mode. In this case the application code runs outside the Bluetooth device, and the BGScript source code files do not need to be included in the Bluetooth project. When BGScript code is not defined in the project file the Bluetooth device will be put into NCP mode.

2. Project File Syntax

The project file (typically named *project.xml* or *project.bgproj*) is the file that describes all the components included in your Bluetooth project.

Typically these files are named as follows:

- **Hardware.xml** — Hardware configuration file for interfaces like UART, SPI, I^2C and GPIO
- **GATT.xml** — GATT database file for Bluetooth services and characteristics
- **Script.bgs** — Optional BGScript application source code

The project file also defines other features of the project like the hardware version, firmware output files, and the bootloader. The project file itself is a simple XML file with only a few attributes on it, which are described in the following sections.

2.1 <project>

The XML attribute **<project>** is used to mark the beginning of the definition of a project file. All the other definitions need to be inside the **<project>** and **</project>** tags.

Example: A project file definition.

```
<project>
…
</project>
```

2.2 <hardware>

The XML element **<hardware>** is used to define the location of the hardware configuration file.

Attribute	Description
in	This attribute and its value are used to define the XML file, which contains the hardware configuration for your Bluetooth device.

Example: Defining the hardware configuration file.

```
<hardware in="hardware.xml" />
```

2.3 <gatt>

The XML element **<gatt>** defines the location of the GATT database definition file.

Note: For more information on the GATT database see *UG118: Blue Gecko Bluetooth® Profile Toolkit Developer's Guide*.

Attribute	Description
in	This attribute and its value point to the XML file, which contains the Bluetooth GATT database definition.

Example: Defining the GATT database definition file.

```
<gatt in="gatt.xml" />
```

2.4 <scripting>

The XML element **<scripting>** is used to group the location definition of an optional BGScript application code as defined by the XML element **<script>** (see the next subsection).

Attribute	Description
-	-

Example: Grouping of <*script*> elements the <*scripting*> XML element.

```
<scripting>
   <script in="bgm111demo.bgs" />
</scripting>
```

2.4.1 <script>

The XML element **<script>** defines the location of the (optional) BGScript application code file.

Attribute	Description
in	This attribute and its value point to the .BGS file, which contains the BGScript application code. **Note:** This definition is required only for projects in which a BGScript application is used.
stack	This attribute sets the size of the stack. **Range**: 1 - limit set by available memory **Default**: 256

Example: Defining the location of the optional BGScript file.

```
<scripting>
   <script in="bgm111demo.bgs" />
</scripting>
```

Note: If you are making a project using NCP mode simply omit the<*script*> element from the project file.

2.5 <software>

The XML element **<software>** is used to group software related definitions together.

Attribute	Description
-	-

Example: Grouping of software related definitions using the <*software*> XML element.

```
<software>
   <max_connections value="8" />
   ...
</software>
```

2.5.1 <max_connections>

The XML element **<max_connections>** defines the maximum number of simultaneous Bluetooth connections allowed by the Bluetooth stack.

Attribute	Description
value	This attribute defines the maximum number of simultaneously allowed Bluetooth connections. **Range**: 1 - 8 **Default**: 4

Example: Setting the maximum number of simultaneously allowed Bluetooth connections to 8.

```
<software>
   <max_connections value="8" />
</software>
```

2.6 <image>

The XML element **<image>** defines the name of the .BIN firmware file output by the BGBuild compiler. BIN files can be flashed with the Simplicity Commander application.

Attribute	Description
out	This attribute and its value define the name of the firmware output file.

Example: Naming the firmware output file.

```
<image out="firmware.bin"/>
```

2.7 <build>

The XML element **<build>** defines the name of the .HEX and .EBL firmware files output by the BGBuild compiler. HEX files can be flashed with the Simplicity Commander application and EBL files are used for OTA and NCP (UART) firmware updates.

Created files

- *stack.ebl* - this file contains only the stock. Used only with OTA updates
- *app.ebl* - this file contains only the application and excludes the *Bluetooth* stack. Used for both OTA and NCP updates
- *full.ebl* - this file contains the *Bluetooth* stack and the application. Used only for NCP updates.
- *.hex* - this file contains the *Bluetooth* stack and the application in HEX format to be used with Simplicity Commander application.

Attribute	Description
hexout	This attribute defines the name for the .HEX output files **Value** **String**: defines the name of the file **Default**: empty string
ebl	This attribute defines the name for the .EBL output files **Value** **String**: defines the prefix to be appended to name of the files **Default**: empty string

Example: Defining the prepended string for *app.dfu* and *full.dfu* file names as *"wstk_bgapi"*.

```
<!-- Define .hex and .ebl firmware output files -->
<build hexout="wstk_bgapi.hex" ebl="wstk_bgapi" />
```

The above example code will produce the following file names:

- *wstk_bgapi.hex* - Both Bluetooth stack and application for Simplicity Commander
- *wstk_bgapi_stack.ebl* - Bluetooth stack for OTA
- *wstk_bgapi_app.ebl* - Application for OTA
- *wstk_bgapi_full.ebl* - Both Bluetooth stack and application for NCP

2.8 <bootloader>

The XML element **<bootloader>** can be used to include the bootloader in the firmware image.

For the UART bootloader the interface settings (pins, baud rate etc.) for the bootloader are defined with the <uart> element in the hardware configuration file.

Attribute	Description
type	This attribute defines the bootloader type to be included in the project. **Values:** **uart** UART (NCP) bootloader **ota** Bluetooth Over-the-Air bootloader **Default:** ota
wstk	This attribute defines the UART bootloader is used with Silican Labs' WSTK development kit or not. **Values:** **true:** Bootloader is used with WSTK development kit and two GPIOs are set by the bootloader to enable the VCOM on the WSTK **false:** Bootloader is NOT used with WSTK and GPIOs are not set. **Default:** false

Example: Enabling UART bootloader in the firmware.

```
<!-- Include Bootloader with UART DFU support -->
<bootloader type="uart"/>
```

Example: Enabling UART bootloader in the firmware for the WSTK development kit.

```
<!-- Include Bootloader with UART DFU support -->
<bootloader type="uart" wstk="true"/>
```

2.9 Project File Examples

Example: Project file for a Blue Gecko device with application code implemented using BGScript scripting language.

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

<!-- Project for Blue Gecko -->
<project>

  <!-- GATT service database -->
  <gatt in="gatt.xml" />

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

  <!-- BGScript source code -->
  <scripting>
     <script in="bgm111demo.bgs" />
  </scripting>

  <!-- Firmware output file (BIN) -->
  <image out="bgm111demo.bin" />

  <!-- .HEX and .EBL firmware output files -->
  <build hexout="bgm111demo.hex" ebl="bgm111demo" />

</project>
```

Example: Project file creation for a Blue Gecko device with application code running on an external host (NCP mode).

Note: The UART must be enabled in the *hardware.xml* file.

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

<!-- Project for Blue Gecko -->
<project>

  <!-- GATT service database -->
  <gatt in="gatt.xml" />

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

  <!-- Firmware output file -->
  <image out="WSTK_BGAPI.bin" />

  <!-- .HEX and .EBL firmware output files -->
  <build hexout="WSTK_BGAPI.hex" ebl="WSTK_BGAPI" />

</project>
```

3. Hardware Configuration File Syntax

The hardware configuration file (typically named *hardware.xml*) is the file that describes the hardware interface configuration and settings for both the host and the peripheral interfaces.

The hardware configuration file is a simple XML file. The syntax is described in the following subsections.

Note: The examples shown in this section assume the device is BGM111 unless otherwise noted. When using other devices it should be noted that they may have different GPIO pinouts and that some of the examples will need to be modified accordingly.

Note: Check available GPIO pins from the device's data sheet and note that the indicated ranges may be a subset of the ones indicated in the tables in this document.

3.1 <sleep>

The XML element **<sleep>** can be used to allow or disallow the use of **EM2** sleep mode.

Note: The device wakes up when the wake-up pin changes state to the defined active state and stays wake as long as the wake-up is held in active state.

Attribute	Description
enable	This attribute is used to allow or disallow the use of **EM2** sleep mode. **true**: **EM2** sleep mode is allowed **false**: **EM2** sleep mode is not allowed **Default**: false

Example: Allowing the use of EM2 sleep mode.

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

3.2 <wake_up>

The XML element **<wake_up>** can be used to configure an I/O pin to wake the device up from **EM2** mode. This is needed if the device is used in NCP mode and **EM2** power save mode is enabled with <**sleep**>.

When the device is in **EM2** mode the host UART will be disabled and the host MUST use the wake-up pin to wake the device up before sending any BGAPI commands to it.

Attribute	Description
port	This attribute defines the port of the wake-up pin. **Range**: A - F **Default**: -
pin	This attribute defines the pin of the wake-up pin. **Range**: 0 - 15 **Default**: 0
state	This attribute defines the polarity of the wake-up pin in its active state. **up**: wake-up occurs when pin changes to high state and remains wake as long as pin is held high **down**: wake-up occurs when pin changes to low state and remains wake as long as pin is held low **Default**: up

Example: Enabling **EM2** mode with wake-up pin configured to port B pin 0.

```
<wake_up port="b" pin="0" state="up" />
```

3.3 <host_wake_up>

The XML element **<host_wake_up>** can be used to configure an I/O pin, which is used to wake up the host MCU when the Bluetooth device sends data or events over the host interface. The external host must then use UART flow control signals (or wake immediately) so that the Bluetooth device can send data to it. This feature is needed if the device is used in NCP mode and the host MCU is sleeping and needs to be woken up when the data or events are received over the host interface.

Notice that the host wake-up pin is only meant to wake up the host from sleep and it does not necessarily remain active during the UART transmission. The host therefore should not go back to sleep after the host wake-up pin is de-asserted, but only after all the expected data has been received over UART.

Attribute	Description
port	Defines the port of the host wake-up pin. **Range**: A - F **Default**: -
pin	Defines the pin of the defined host wake-up pin port. **Range**: 0 - 15 **Default**: 0
state	Defines the I/O state of the defined host wake-up pin when wake-up occurs. **Values:** **up:** wake-up occurs when pin changes to high state and remains wake as long as pin is held high **down**: wake-up occurs when pin changes to low state and remains wake as long as pin is held low **Default**: up

Example: Configuring host wake-up pin to port F pin 6. In WSTK Development Kit SLWSTK6101B and for the Radio Board 4300A, this pin connects the host wake-up pin to LED0 on the WSTK Main Board.

```
<host_wake_up port="f" pin="6" state="down" />
```

3.4 <uart>

The XML element <uart> and its attributes are used to configure the UART interface settings.

Attribute	Description
index	This attribute is used to select the UART to configure. **Values:** **0:** UART 0 **1:** UART 1 **Default:** 0
baud	This attribute defines the UART baud rate. **Range**: 600 – 6000000 **Default**: 115200
rx_pin	This attribute defines the UART RX pin location. **Range**: PA0 – PF7 **Default**: PA1
tx_pin	This attribute defines the UART TX pin location. **Range**: PA0 – PF7 **Default**: PA0
rts_pin	This attribute defines the UART RTS pin location. **Range**: PA0 – PF7 **Default**: PA5
cts_pin	This attribute defines the UART CTS pin location. **Range**: PA0 – PF7 **Default**: PA4
flowcontrol	This attribute defines the RTS/CTS flow control state as enabled or disabled. RTS/CTS flow control is recommend to be set to enabled whenever possible to ensure reliable data transmission over the UART interface. **Values:** **true**: RTS/CTS enabled **false**: RTS/CTS disabled **Default**: false
parity	This attribute defines the parity setting for the defined UART. **Values:** **odd**: Odd parity enabled **even**: Even parity enabled **none**: Parity disabled **Default**: none
databits	This attribute defines the number of databits for the defined UART. **Values:** **8**: Number of databits is 8 **Default**: 8

Attribute	Description
stopbits	This attribute defines the number of stopbits for the defined UART. **Values:** **1**: Number of stopbits is 1 **2**: Number of stopbits is 2 **Default**: 1
bgapi	This attribute defines whether the BGAPI serial protocol (NCP mode) is enabled or disabled over the UART interface. When an external host is used to control the Bluetooth module over UART, the BGAPI serial protocol must be enabled. When a BGScript application runs in the device the BGAPI serial protocol should be disabled. **true**: BGAPI serial protocol is enabled over UART **false**: BGAPI serial protocol is disabled for UART **Default**: false

Example: Enabling UART at 115200 bps, disabling RTS/CTS flow control and disabling BGAPI serial protocol. Default pin mappings are used. BGScript application has control over UART.

```
<uart index="1" baud="115200" flowcontrol="false" bgapi="false" />
```

Example: Enabling UART at 460800 bps, enabling RTS/CTS flow control and enabling BGAPI serial protocol. Default pin mappings are used.

```
<uart index="1" baud="460800" flowcontrol="true" bgapi="true" />
```

3.5 <i2c>

The XML element **<i2c>** and its attributes are used to configure the I^2C interface settings.

Attribute	Description
scl_pin	This parameter defines the I^2C SCL pin location. **Range**: PA0 – PF7 **Default**: PC11
sda_pin	This parameter defines the I^2C SDA pin location. **Range**: PA0 – PF7 **Default**: PC10

Example: Enabling I^2C into pins PC10 (SDA) and PC11 (SCL).

Note: On the SLWSTK6101B Main Board the RHT sensor uses these pins.

```
<i2c scl_pin="PC11" sda_pin="PC10" />
```

3.6 <gpio>

The XML element **<gpio>** and its attributes are used to configure the GPIO pin settings.

Attribute	Description
port	This attribute is used to select the port to configure. **Range:** A – F
pin	This attribute is used to select the pin of the defined port to configure. **Range:** 0 – 15
out	This attribute configures the default value of the data out register (DOUT). **Values:** **0:** DOUT register value is 0 **1:** DOUT register value is 1 **Default:** 0
mode	This attribute is used to configure the GPIO mode of the defined port pin. **Values:** **DISABLED:** Input disabled. Pull-up if DOUT is set (see above). **INPUT:** Input enabled. Filter if DOUT is set (see above). **INPUTPULL:** Input enabled. DOUT (see above) determines pull direction. **INPUTPULLFILTER:** Input enabled with filter. DOUT (see above) determines pull direction. **PUSHPULL:** Push-pull output. **PUSHPULLALT:** Push-pull using alternate control. **WIREDOR:** Wired-or output. **WIREDORPULLDOWN:** Wired-or output with pull-down. **WIREDAND:** Open-drain output. **WIREDANDFILTER:** Open-drain output with filter. **WIREDANDPULLUP:** Open-drain output with pull-up. **WIREDANDPULLUPFILTER:** Open-drain output with filter and pull-up. **WIREDANDALT:** Open-drain output using alternate control. **WIREDANDALTFILTER:** Open-drain output using alternate control with filter. **WIREDANDALTPULLUP:** Open-drain output using alternate control with pull-up. **WIREDANDALTPULLUPFILTER:** Open-drain output using alternate control with filter and pull-up. **Default:** PUSHPULL

Attribute	Description
interrupt	This attribute is used to enable or disable interrupts and select rising, falling or both edge triggering of the pin to configure. **Values**: **none**: Interrupts are disabled **rising**: Interrupts generated on rising edge **falling**: Interrupts generated on falling edge **both**: Interrupts generated on both rising and falling edges **Default**: none

The examples below apply for BGM111 and WSTK Main Board.

Example: Enabling the built-in VCOM (UART-to-USB bridge) on the WSTK Main Board requires that Port A pin 5 and and Port A pin 3 are configured as an outputs. The following GPIO configuration must be used with WSTK if UART functionality is needed. Port A pin 5 acts as the VCOM_ENABLE and Port A pin 3 acts as the CTS signal that is set to a constant value '0'.

```
<gpio port="A" pin="5" mode="pushpull" out="1" />
<gpio port="A" pin="3" mode="pushpull" out="0" />
```

Example: Configuring both LED0 (Port F pin 6) and LED1 (Port F pin 7) on the SLWSTK6101B Main Board with default state "0", which means the LEDs are on (active low).

```
<!-- LED0 -->
<gpio port="F" pin="6" mode="pushpull" out="0" />
<!-- LED1 -->
<gpio port="F" pin="7" mode="pushpull" out="0" />
```

Pin Configuration

Table 3.1. Pin Configuration

MODEn	Input	Out-put	DOUT	Pull-down	Pull-up	Alt. strength	Input Filter	Description
DISABLED	Disa-bled	Disa-bled	0					Input disabled
			1		On			Input disabled with pull-up
INPUT	Ena-bled		0					Input enabled
			1				On	Input enabled with filter
INPUTPULL			0	On				Input enabled with pull-down
			1		On			Input enabled with pull-up
INPUTPULLFILTER			0	On			On	Input enabled with pull-down and filter
			1		On		On	Input enabled with pull-up and filter
PUSHPULL		Push-pull	x					Push-pull
PUSHPULLALT			x			On		Push-pull with alt. drive strength
WIREDOR		Open Sourc e (Wired -OR)	x					Open-source
WIREDORPULLDOWN			x	On				Open-source with pull-down
WIREDAND		Open Drain (Wired -AND)	x					Open-drain
WIREDANDFILTER			x				On	Open-drain with filter
WIREDANDPULLUP			x		On			Open-drain with pull-up
WIREDANDPULLUPFILTER			x		On		On	Open-drain with pull-up and filter
WIREDANDALT			x			On		Open-drain with alt. drive strength
WIREDANDALTFILTER			x			On	On	Open-drain with alt. drive strength and filter
WIREDANDALTPULLUP			x		On	On		Open-drain with alt. drive strength and pull-up
WIREDANDALTPULLUPFILTER			x		On	On	On	Open-drain with alt. drive strength, pull-up and filter

Note: x = Don't care

3.7 <adc>

The XML element **<adc>** is used for configuring the ADC settings.

Attribute	Description
enable	This attribute is used to enable or disable the ADC functionality. **Values:** **true:** ADC is enabled **false:** ADC is disabled **Default:** false

Example: Enabling ADC.

```
<adc enable="true" />
```

3.8 <ctune>

The XML element **<ctune>** can be used to configure the crystal High-Frequency Crystal (HFXO) frequency tuning value.

Normally this value should not be changed and the default value can be used. However, on EFR32 SoC designs the ctune value may need to be adjusted based on the hardware design.

Note: For more information on the selection of the **<ctune>** value for an SoC, see the related SoC's Reference Manual.

Attribute	Description
value	This parameter value configures the crystal tune value. **Range**: 0 - 511 **Default**: 322 Required settings: **BGM111A256V1**: 0 With other BGM modules and EFR32BG or MG radio boards do not use the ctune setting.

Example: Setting crystal tune value for BGM111A256V1.

```
<ctune value="0" / >
```

3.9 <lfxo>

The XML element **<lfxo>** can be used to define whether the Low Frequency Crystal (LFXO) is present in the system or not.

All Blue Gecko modules have the LFXO built in, which means that the value of the attribute of this XML element must be set to true (which is also the default value). With Blue Gecko SoC designs you can decide not to use the LFXO to optimize BoM cost.

Note: EM2 or more aggressive sleep modes are not available if LFXO is not present.

Note: LFXO must be defined if **EM2** mode (sleep mode) is required. If LFXO is not defined Low Frequency RC Oscillator (LFRC) is used, which prevents the use of **EM2** sleep mode.

Attribute	Description
enable	This attribute is used to define if LFXO is present or not. **Values:** **true**: LFXO is present **false**: LFXO is not present **Default**: true
tuning	This attribute is the ***ctune*** attribute for the LFXO element. **Range:** 0 - 1023 **Default:** 0

Example: Disabling LXFO.

```
<lfxo enable="false" />
```

3.10 <dcdc>

The XML attribute **<dcdc>** is used to enable or disable the DC-DC bypass mode.

Note: The data below applies for BGM111 and BGM113. For other device types consult the device's datasheet.

When DC/DC is enabled, supply voltage range is 2.4 to 3.8 V.

When DC/DC is disabled, supply voltage range is 1.85 to 3.8 V.

Attribute	Description
enable	This parameter is used to enable or disable the built-in DC-DC converter. **Values:** **true:** DC-DC is enabled **false:** DC-DC is disabled **Default**: enabled

Example: Enabling the built-in DC/DC.

```
<!-- Enable DC/DC converter -->
<dcdc enable="true" />
```

Example: Disabling the built-in DC/DC.

```
<!-- Disable DC/DC converter -->
<dcdc enable="false" />
```

3.11 <pti>

The XML element **<pti>** can be used to enable or disable the packet trace feature on the Blue Gecko Bluetooth hardware.
Packet trace outputs the Bluetooth packets sent and received by the Blue Gecko hardware via a special hardware interface. They can be viewed in the Simplicity Studio's Network Analyzer tool. The packet trace feature is useful for debugging the Bluetooth communications and error situations.

Attribute	Description
enable	This attribute defines if packet trace feature is enabled or disabled. **Values** **true**: Packet Trace is enabled **false**: Packet Trace is disabled **Default**: false
mode	This attribute configures the PTI format. **Values** **uart**: UART format: TX + frame **spi**: SPI format: Data + Clock + Frame **onewire**: one wire format: Data only **Default**: uart
baud	This attribute configures the PTI baud rate in Hz. **Range**: 9600 - 2000000 **Default**: 1600000
data	This attribute selects the data output pin. **Range**: PA0 - PF7 **Default**: PA4
clock	This attribute selects the clock output pin. **Range**: PA0 - PF7 **Default**: PB11
frame	This attribute selects the frame output pin. **Range**: PA0 - PF7 **Default**: PB13

Example: Enabling packet trace with default settings.

```
<pti enable="true" />
```

3.12 <obssel>

The XML element **<obssel>** is used to enable or disable the pins indicating that the device is either transmitting (TX) or receiving (RX) data via RF interface.

Note: TX and RX pins must be physically different pins. You cannot use the same pin for both indications.

Attribute	Description
rx_obs_pin	This attribute is used to define the GPIO pin used to indicate that the device is receiving data via RF interface. **Range:** **6 - 11**: Defines the pin of Port C used as the RX indication pin **Default**: - **Note:** RX activity is indicated by high logic level.
tx_obs_pin	This parameter is used to define the GPIO pin used to indicate that the device is transmitting data via RF interface. **Range:** **6 - 11**: Defines the pin of Port C used as the TX indication pin **Default**: - **Note:** TX activity is indicated by high logic level.

Example: Setting RX and TX indication pins as RX = Port C pin 6 and TX = Port C pin 7.

```
<!-- Enable TX and RX indication pins -->
<obssel rx_obs_pin="6" tx_obs_pin="7" />
```

3.13 <ota>

The XML element **<ota>** is used to configure the OTA options for OTA DFU mode.

Attribute	Description
authenticated_write	This attribute defines if writing to the OTA characteristics requires authentication. In order to write the OTA characteristics the remote device has to be bonded with MITM protection and the connection must be encrypted. **Values:** **true**: authentication is required **false:** authentication is not required **Default**: false
bonded_write	This attribute defines if writing to the OTA characteristics requires bonding. In order to write to OTA characteristics the remote device has to be bonded at least with Just Works pairing. **Values:** **true**: bonding is required **false:** bonding is not equired **Default**: false
encrypted_write	This attribute defines if writing to the OTA characteristics requires an encrypted link. In order to write to OTA characteristics the remote device has to encrypt the link. With iOS 9.1 and newer versions the device must also be bonded with at least Just Works pairing. **Values:** **true**: encryption is required **false:** encryption is not equired **Default**: false
device_name	Defines the device name to be used in OTA DFU mode.

Example: Setting the OTA device name to "My OTA" with authenticated write protection.

```
<ota authenticated_write="true" device_name="My OTA" />
```

3.14 Hardware Configuration File Examples

The example below (written for BGM111) shows how to create a hardware configuration file for a Blue Gecko Bluetooth device, where the application is implemented with BGScript scripting language and both UART and I^2C interfaces are enabled and controlled by the script application.

Example: Hardware configuration with both UART and I^2C enabled .

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

<hardware>

   <!-- UART configuration -->
   <!-- Settings: UART 1 enabled, @115200bps, no RTS/CTS and BGAPI serial protocol is disabled -->
   <uart index="1" baud="115200" flowcontrol="false" bgapi="false" />

   <!-- I2C configuration -->
   <!-- Settings: SCL pin PC11 and SDA pin PC10 -->
   <i2c scl_pin="PC11" sda_pin="PC10" />

</hardware>
```

Example: A host-based project (NCP) with BGAPI serial protocol enabled for WSTK VCOM.

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

<hardware>

   <!-- UART configuration -->
   <!-- Settings: UART 1 enabled, @115200bps, no RTS/CTS and BGAPI serial protocol enabled -->
   <uart index="1" baud="115200" flowcontrol="false" bgapi="true" />

   <!-- Enable UART on WSTK -->
   <!-- PA5 as output, PA3 as output -->
   <gpio port="A" pin="5" mode="pushpull" out="1" />
   <gpio port="A" pin="3" mode="pushpull" out="0" />

</hardware>
```

4. Compiling the Project and Verifying Configuration

The BGScript or NCP project can be either compiled from the command line using the BGBuild compiler or from Simplicity Studio using a BGTool application.

To compile the project from the command line run bgbuild.exe, which is delivered in the SDK.

Usage:

```
Usage: ..\..\..\..\protocol\bluetooth_dev\bin\bgbuild.exe [options] input
BGBuild tool for Blue Gecko

Options:
  -?, -h, --help                    Displays this help.
  -v, --version                     Displays version information.
  -g, --gattonly                    Only create GATT c-file.
  -r, --root <buildtools>           Override default build tools location
  -s, --scompiler <scriptcompiler>  Path to script compiler
  -c, --cc <path>                   Path to C compiler
  -p, --partID <partID>             PartID of connected device
  -f, --flash                       Flash resulting image to device
  -w, --hwonly                      Only create HW configuration c-file.
  -n, --nopaths                     Remove relative paths from generated
                                    c-files.

Arguments:
  input                             Project file to build.
```

The most common options are highlighted below:

Parameter	Description
-c	GCC compiler path. Typically: **C:\SiliconLabs\SimplicityStudio\v4\developer\toolchains\gnu_arm\4.9_2015q3** The GCC path can be also set to PATH environment variable, but then it must include \bin at the end.
-f	Flash the firmware after compilation. Optional to use.
-p	EFR32 part number such as "efr32bg1b232f256gm48" (must be lower case). If not given, defaults to EFR32BG1P (Blue Gecko Performance line).

Example: Compiling empty.bgproj BGScript project with BGBuild compiler.

```
bgbuild.exe -c C:\SiliconLabs\SimplicityStudio\v4\developer\toolchains\gnu_arm\4.9_2015q3 empty.bgproj
```

Example: Compiling and flashing a BGscript project.

```
bgbuild.exe -f -c C:\SiliconLabs\SimplicityStudio\v4\developer\toolchains\gnu_arm\4.9_2015q3 empty.bgproj
```

Note: bgbuild.exe is located in folder **..\..\..\..\protocol\bluetooth_dev\bin** in relation to **examples_bgscript** folder.

In order to compile the project with BGTool in Simplicity Studio.

1. Run Simplicity Studio and select the connected device form the devices view on the left.
2. Start the BGTool application and go to the Upload tool page.
3. Select the project file and click [**Build**].

The following table lists possible output parameters that may be listed in the *Build result* output window. Some parameters might not be listed, depending on the content and settings defined in the project file.

Output	Description
compiler	Location of the BGBuild compiler used in the compilation. Shown only if BGScript is used.
script	Main BGScript source code file. Shown only if BGScript is used.
api	Location of the used API definition file. Shown only if BGScript is used.
stack	Available stack allocated for BGScript applications.
constants	Name of the file containing the IDs and attribute handles defined in the GATT database.
variables	The amount of memory reserved by variables in bytes.
UART0 ***or*** ***UART1***	UART interface settings **Values** **baudrate**: UART baud rate **flowcontrol**: RTS/CTS flow control configuration **parity**: **data bits**: **stop bits**: **BGAPI:** BGAPI serial protocol configuration **UART RX:** UART RX Port and pin **UART TX:** UART TX Port and pin **UART CTS:** UART CTS Port and pin **UART RTX:** UART RTS Port and pin
WAKE_UP	Port and pin used as the EM2 wake-up (shown only if enabled).
HOST_WAKE_UP	Port and pin used for host wake-up pin (shown only if enabled).
I2C0	I2C interface settings. **Values** **I2C0 SDA:** I2C SDA port and pin **I2C0 SCL:** I2C SCL port and pin
ADC	Whether the device's ADC is enabled or disabled.
CTUNE	Configured crystal tune value.
GPIO	GPIO configuration and pin assignment check result.
External interrupts	The GPIO pins that have interrupts enabled.
Total	Total size of image in bytes.
MAX Flash	Total size of flash memory on the device in bytes.
Free	Free part of the flash memory on the device in bytes and also in percentage.
RAM	Random Access Memory size in bytes.

5. Factory Default Configuration

The BGM111 and BGM113 modules are delivered with a pre-installed firmware from the factory, which enables the Network Co-Processor (NCP) over UART interface. In this mode the Bluetooth modules can be controlled with the BGAPI serial protocol over the UART interface.

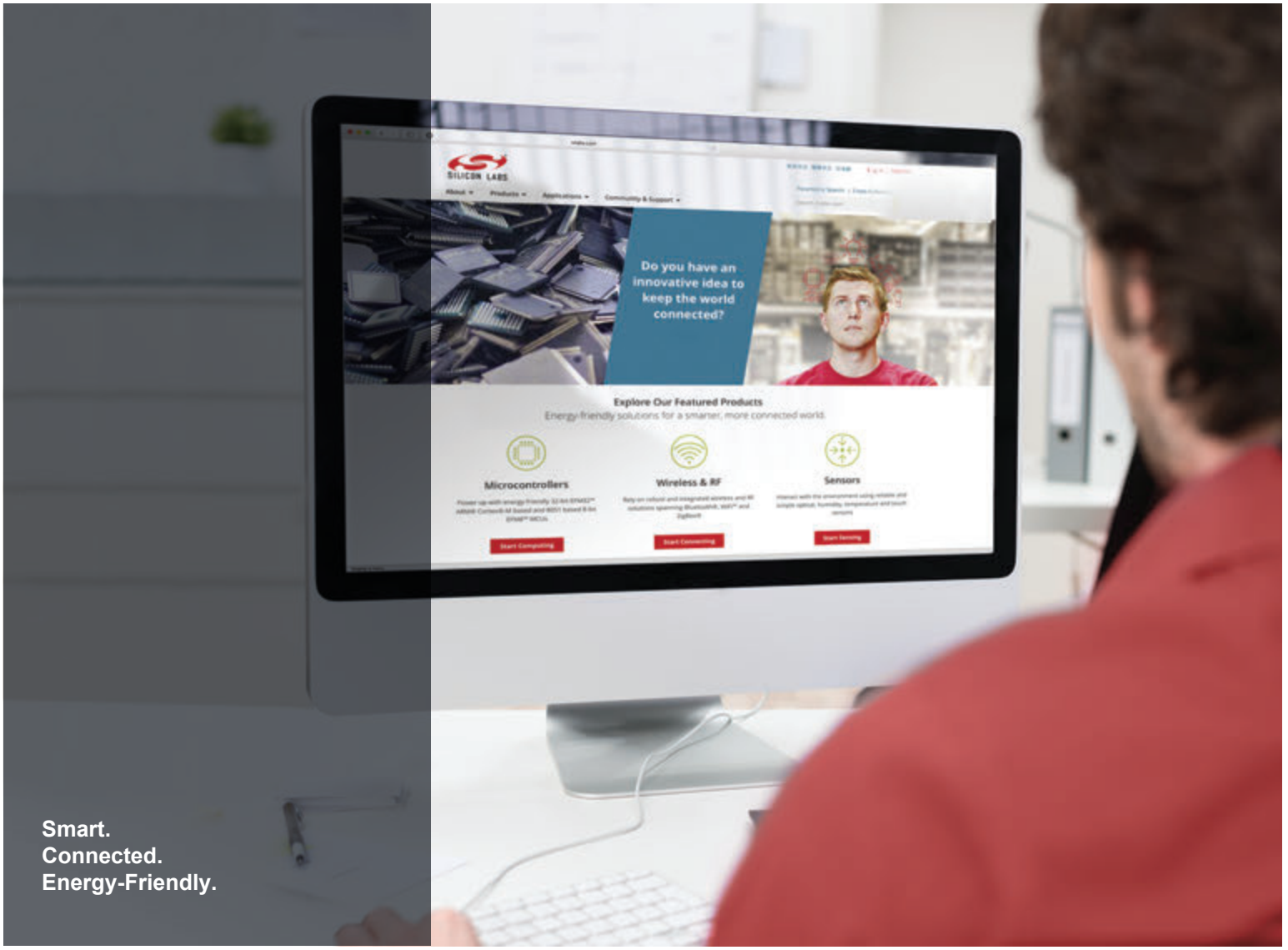
The factory default settings are described below.

Factory Default Configuration for BGM111

Feature	Setting	Value	Notes
BGAPI serial protocol over UART	USART	1	This UART gives access to BGAPI serial protocol, which can be used to control the module from an external host.
	Pins	• **TX** PA0 • **RX** PA1 • **CTS** PA2 • **RTS** PA3	
	baud rate	115200	
	data bits	8	
	stop bits	1	
	RTS/CTS	enabled	

Factory Default Configuration for BGM113

Feature	Setting	Value	Notes
BGAPI serial protocol over UART	USART	1	This UART gives access to BGAPI serial protocol, which can be used to control the module from an external host.
	Pins	• **TX** PC10 • **RX** PC11 • **CTS** PF3 • **RTS** PF2	
	baud rate	115200	
	data bits	8	
	stop bits	1	
	RTS/CTS	enabled	



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

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

http://www.silabs.com