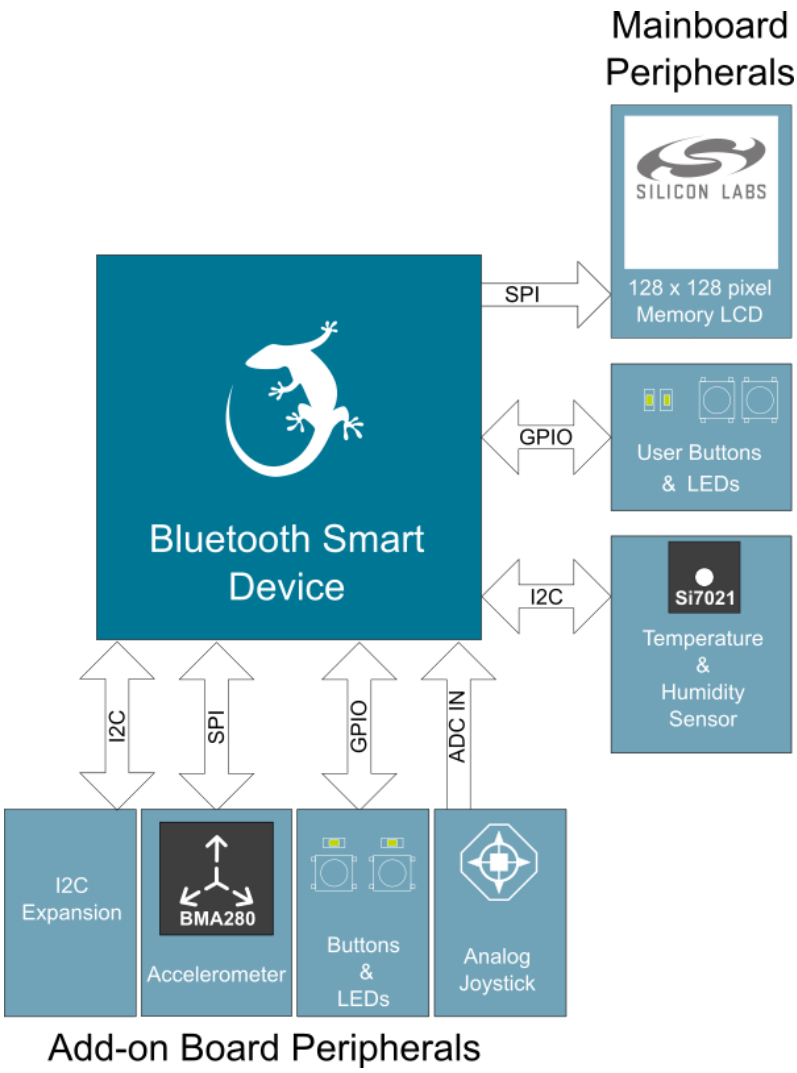
UG119: Blue Gecko *Bluetooth*® Smart Device Configuration Guide

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

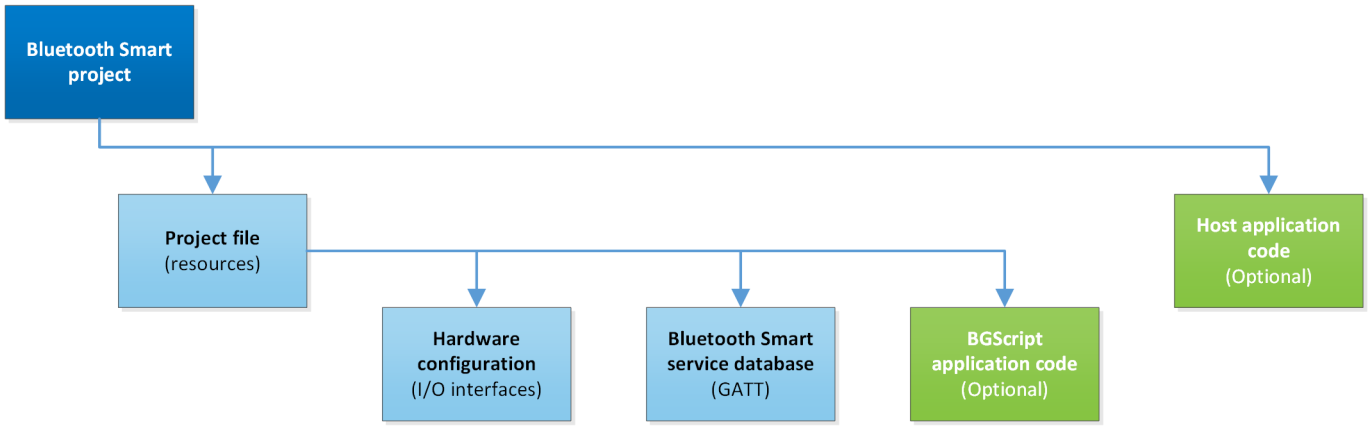
KEY POINTS
- Starting a software project
- Including necessary resources
- Configuring hardware interface settings



1. Project Structure

The flowchart below illustrates the structure of a *Bluetooth* Smart 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* Smart 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.

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 Smart Service Database

The *Bluetooth* Smart 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* Smart 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 the *UG118: Blue Gecko Bluetooth Smart 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* Smart devices. In case 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 way to using BGScript based application code is to use an external host (typically an MCU), and use the *Bluetooth* Smart device as a modem or in so-called 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* Smart project. When BGScript code is not defined in the project file the *Bluetooth* Smart device will be put into the 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* Smart 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* Smart 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 selected boot loader. 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 and also includes as a parameter the hardware device type the project is intended for. All the other definitions need to be inside the **<project>** and **</project>** tags.

Parameter	Description
device	This parameter and its value define the hardware device type the project is to compiled for. **Values** **bgm111**: Blue Gecko BGM111 *Bluetooth* Smart module **bgm113**: Blue Gecko BGM113 *Bluetooth* Smart module **bluegecko** : A Blue Gecko *Bluetooth* Smart SoC

Example: A project file definition for BGM111 *Bluetooth* Smart module

```
<project device="bgm111">
…
</project>
```

Example: A project file definition for EFR32 *Bluetooth* Smart SoC

```
<project device="bluegecko">
…
</project>
```

2.2 <build>

The XML attribute **<build>** is be used to instruct the compiler to build two files which are needed when doing a DFU update of firmware. The parameter *dfu* and its string value are used to define the prefix prepended to the file names.

Created files

- *app.dfu* - this file contains only the application
- *full.dfu* - this file contains the application and the full stack

Parameter	Description
dfu	This parameter defines the prefix prepended to the created file names. **Value** **String**: defines the prefix to be appended to name of both files **Default**: empty string

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

```
<!-- Define prepended string for app.dfu and full.dfu file names -->
<build dfu="wstk_bgapi" />
```

The above example code will produce the following file names:

- *wstk_bgapi_app.dfu*
- *wstk_bgapi_full.dfu*

2.3 <hardware>

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

Parameter	Description
in	This parameter and its value are used to define the XML file which contains the hardware configuration for your *Bluetooth* Smart device.

Example: Defining the hardware configuration file

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

2.4 <gatt>

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

Parameter	Description
in	This parameter and its value point to the XML file which contains the *Bluetooth* Smart GATT database definition.

Example: Defining the GATT database definition file

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

2.5 <scripting>

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

Parameter	Description
-	-

Example: Grouping of <*script*> attributes within <*scripting*> XML attribute.

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

2.5.1 <script>

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

Parameter	Description
in	This parameter 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 ise used.

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 leave the<*script*> attribute out from the project file.

2.6 <software>

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

Parameter	Description
-	-

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

```
<software>
   <max connections value="8" />
   ...
</software>
```

2.6.1 <max_connections>

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

Parameter	Description
value	This parameter defines the maximum number of simultaneously allowed BLE connections. **Range**: 0 - 8 **Default**: 4

Example: Setting the maximum number of simultaneosly allowed BLE connections to 8.

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

2.7 <image>

The XML attribute **<image>** defines the name of the firmware file output by the BGBuild compiler.

Parameter	Description
out	This parameter and its value define the name of the firmware output file.

Example: Naming the firmware output file

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

2.8 Project File Examples

The examples shown in this section are written for BGM111.

Example: Project file for a Blue Gecko device (BGM111) with application code implemented using BGScript scripting language

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

<!-Project for BGM111 Bluetooth Smart Module -->
<project device="bgm111">

  <!-- 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 -->
  <image out="bgm111demo.bin" />

</project>
```

Example: Project file creation for a Blue Gecko device (BGM111) with application code running on an external host (NCP mode)

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

<!-Project for BGM111 Bluetooth Smart Module -->
<project device="bgm111">

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

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

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

</project>
```

Example: Project file for an EFR32BG with application code running on an external host (NCP mode).

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

<!-Project for BGM111 Bluetooth Smart Module -->
<project device="bluegecko">

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

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

  <!-- Firmware output file -->
  <image out="pcb4100_wstl_bgapi.bin" />

</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. 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.

3.1 <sleep>

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

Parameter	Description
enable	This parameter 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 attribute **<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 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.

Note: The parameter ranges defined in the table below relate to BGM111. Ranges may differ for other devices. For more details regarding available Ports and pins please see the relevant devices data sheet.

Parameter	Description
port	Defines the port of the wake-up pin **Range**: A - F **Default**: -
pin	Defines the pin of the wake-up pin **Range**: 0 - 15 **Default**: -
state	Defines the I/O state when wake-up occurs **up**: wake-up occurs when pin is high **down**: wake-up occurs when pin is low **Default**: up

Example: Enabling EM2 and wake-up pin configured to PB1

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

3.3 <host_wake_up>

The XML attribute **<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 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.

Note: The parameter ranges defined in the table below relate to BGM111. Ranges may differ for other devices. For more details regarding available Ports and pins please see the relevant devices data sheet.

Parameter	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. **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 is high **down**: wake-up occurs when pin is low **Default**: up

Example: Configuring host wake-up pin to PF6. In WSTK Development Kit SLWSTK6101B and for the Radio Board 43## 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 attribute <uart> and its parameters are used to configure the UART interface settings.

Note: The parameter ranges defined in the table below relate to BGM111 and EFR32BG. Ranges may differ for other devices. For more details regarding available Ports and pins please see the relevant devices data sheet.

Parameter	Description
index	This parameter is used to select the UART to configure. **Values:** **0:** UART 0 **1:** UART 1 **Default:** 0
baud	This parameter defines the UART baud rate. **Range**: 600 – 6000000 **Default**: 115200
rx_pin	This parameter defines the UART RX pin location. **Range**: PA0 – PF7 **Default**: PA1
tx_pin	This parameter defines the UART TX pin location. **Range**: PA0 – PF7 **Default**: PA0
rts_pin	This parameter defines the UART RTS pin location. **Range**: PA0 – PF7 **Default**: PA5
cts_pin	This parameter defines the UART CTS pin location. **Range**: PA0 – PF7 **Default**: PA4
flowcontrol	This parameter 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 parameter defines the parity setting for the defined UART. **Values:** **odd**: Odd parity enabled **even**: Even parity enabled **none**: Parity disabled **Default**: none

Parameter	Description
databits	This parameter defines the number of databits for the defined UART. **Values:** **7**: Do not use (not supported) **8**: Number of databits is 8 **Default**: 8
stopbits	This parameter 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 parameter 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 Smart 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 attribute **<i2c>** and its parameters are used to configure the I^2C interface settings.

Note: The parameter ranges defined in the table below relate to BGM111 and EFR32BG. Ranges may differ for other devices. For more details regarding available Ports and pins please see the relevant devices data sheet.

Parameter	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 that on the SLWSTK6101B Main Board the RHT sensor uses these pins.

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

3.6 <gpio>

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

Note: The parameter ranges defined in the table below relate to BGM111 and EFR32BG. Ranges may differ for other devices. For more details regarding available Ports and pins please see the relevant devices data sheet.

Parameter	Description
port	This parameter is used to select the port to configure. **Range:** A – F
pin	This parameter is used to select the pin within the defined port to configure. **Range:** 0 – 15
out	This parameter 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 parameter is used to configure the GPIO mode of the pin to configure. **Values:** **DISABLED** Input disabled. Pull-up if DOUT is set. **INPUT** Input enabled. Filter if DOUT is set. **INPUTPULL** Input enabled. DOUT determines pull direction. **INPUTPULLFILTER** Input enabled with filter. DOUT 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.

Parameter	Description
interrupt	This parameter 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 UART (UART-to-USB bridge) on the WSTK Main Board requires that PA5 is configured as an output and PA3 as an input. The following GPIO configuration must be used with WSTK if UART functionality is needed.

```
<gpio port="A" pin="6" mode="pushpull" out="1" />
<gpio port="A" pin="3" mode="pushpull" out="0" />
```

Example: Configuring both LED0 (PF6) and LED1 (PF7) 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 attribute **<adc>** and its parameters are used for configuring the ADC settings.

Parameter	Description
enable	This parameter 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 attribute **<ctune>** can be used to configure the crystal (XTAL) frequency tuning value.

The crystal tuning value must be set correctly for every design, which is especially important when making your own EFR32BG based hardware.

Parameter	Description
value	This parameter value configures the crystal tune value. **Range**: 0 - 511 Required or recommended settings per device type and version: **BGM111A256V1**: 0 (required value) **BGM111A256V1.1**: 270 (recommended value) **BGM111A256V2**: 270 (recommended value) **BGM113**: 260 (recommended value) **EFR32BG Radio Board**: Measure and determine optimal value for each design specifically.

Example: Setting crystal tune value for BGM111A256V1

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

3.9 <lfxo>

The XML attribute **<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 the value of this XML attribute must be set to true (which is also the default value), but with the 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 used.

Parameter	Description
enable	This parameter is used to define if LFXO is present or not. **Values:** **true**: LFXO is present **false**: LFXO is not present **Default**: true

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 3.8V.

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

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

Note: The parameter ranges defined in the table below relate to BGM111. Ranges may differ for other devices. For more details regarding available Ports and pins please see the relevant devices data sheet.

Parameter	Description
enable	This parameter defines if packet trace feature is enabled or disabled. **Values** **true**: Packet Trace is enabled **false**: Packet Trace is enabled **Default**: false
mode	This parameter 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 parameter configures the PTI baud rate in Hz. **Range**: 9600 - 2000000 **Default**: 1600000
data	This parameter selects the data output pin **Range**: PA0 - PA7 **Default**: PA4
clock	This parameter selects the clock output pin **Range**: PA0 - PA7 **Default**: PB11
frame	This parameter selects the frame output pin **Range**: PA0 - PA7 **Default**: PB13

Example: Enabling packet trace with default settings

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

3.12 <obssel>

The XML attribute **<obssel>** is used to enable or disable the pins which indicate that the device is either transmitting or receiving data via RF interface.

Note: TX and RX pins must be physically different pins, you can not use the same pin for both indications.

Parameter	Description
rx_obs_pin	This parameter is used to define the GPIO pin used for indicating that the device is receiving data via RF interface. **Range:** BGM111 **6 - 11**: Defines the pin of Port C used as the RX indication pin. **Values:**: BGM113 **PC10**: Use pin 10 of Port C as the RX indication pin. **PC11**: Use pin 11 of Port C as the RX indication pin. **Note:** RX activity is indicated by high logic level.
tx_obs_pin	This parameter is used to define the GPIO pin used for indicating that the module is transmitting data via RF interface. **Range:**BGM111 **6 - 11**: Defines the pin of Port C used as the TX indication pin. **Values:**: BGM113 **PC10**: Use pin 10 of Port C as the TX indication pin. **PC11**: Use pin 11 of Port C as the TX indication pin. **Note:** TX activity is indicated by high logic level.

Example: Setting RX and TX indication pins for BGM111 as RX=PC6 and TX=PC7.

```
<!-- Enable TX and RX indication pins -->
<obssel rx_obs_pin="5" tx_obs_pin="6" />
```

3.13 Hardware Configuration File Examples

The example below (written for BGM111) shows how to create a hardware configuration file for a Blue Gecko *Bluetooth* Smart device, where the application is implemented with BGScript scripting language and both UART and I2C interfaces are enabled and controlled by the script application.

Example: Hardware configuration with both UART and I2C 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 with BGAPI serial protocol enabled

```
<?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. Verifying Configuration from BGBuild Compiler Output

When you compile the project file with the BGBuild compiler it will output the project configuration, BGScript, BGAPI and hardware interface settings, so you can verify the correctness of your configuration files.

An example of a compiler output is shown in the image below.

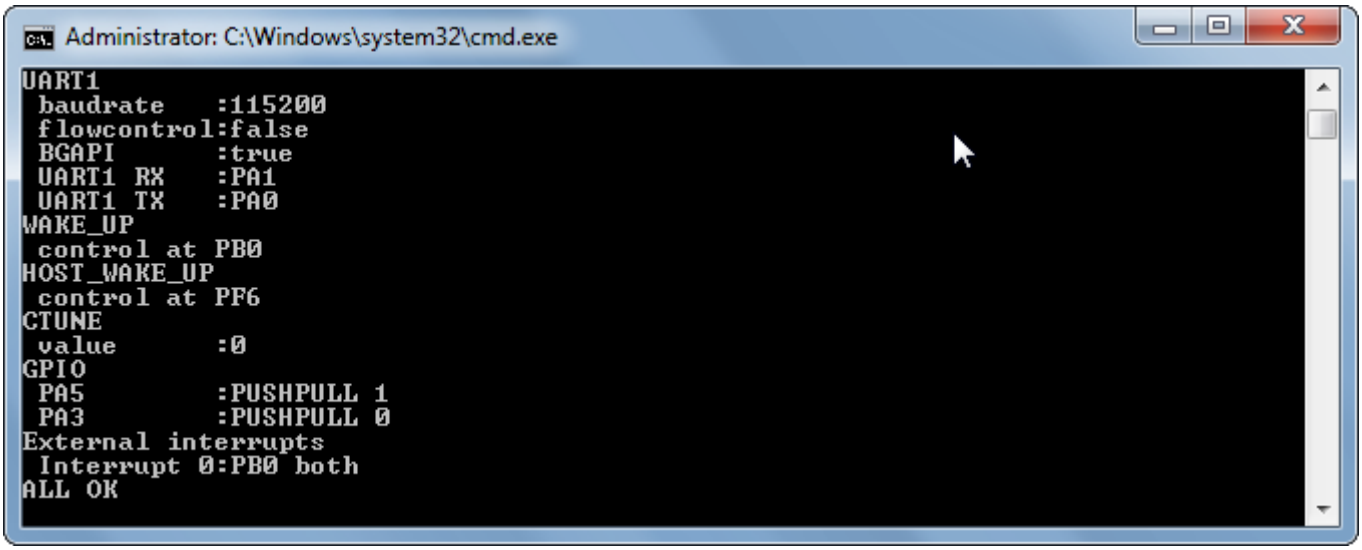


Figure 4.1. BGBuild Compiler Output

The following table lists possible output parameters which may be listed in the BGBuild compiler output screen. Some parameters might be not 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 which contains the IDs and attribute handles defined in the GATT database .
variables	Indicates 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 **BGAPI:** BGAPI serial protocol configuration **UART RX:** UART RX pin **UART TX:** UART TX pin **UART CTS:** UART CTS pin **UART RTX:** UART RTS pin
WAKE_UP	Port and pin used as the EM2 wake-up. (Shown only if enabled.)
HOST_WAKE_UP	Pin used for host wake-up pin. (Shown only if enabled.)
I2C0	I2C interface settings **Values** **I2C0 SDA:** I2C SDA pin **I2C0 SCL:** I2C SCL pin
ADC	Indicates whether the device's ADC is enabled or disabled.
CTUNE	Configured crystal tune value.
GPIO	GPIO configuration and pin assignment check result.
External interrupts	Shows the GPIO pins that have interrupts enabled.

5. Factory Default Configuration

The factory default configuration of the BGM111 and BGM113 modules depend on whether the module was delivered as part of a reel order or as part of the WSTK, which refers to Blue Gecko *Bluetooth* Smart Module Wireless Starter Kit SLWSTK6101A (contains only the Radio Board BRD4300A with BGM111) or SLWSTK6101B (contains Radio Boards BRD4300A with BGM111 and BRD4301A with BGM113).

Factory default values for both cases are described in more detail below.

Reel delivered BGM111 Modules

The BGM111 *Bluetooth* Smart modules supplied in reels have a factory installed firmware created from a project named **"production"**, which is also stored in the **example\production** folder in the SDK. This project file has the hardware configuration listed in the following table.

Note: In case DFU is required, the UART settings are the same as in the table below, determined in the *hardware* section of the **"production"** project.

Feature	Setting	Value	Notes
BGAPI UART	USART	1	This UART gives access to BGAPI serial protocol, which can be used to control the 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	
Note: In addition the parameter *bgapi* in *hardware.xml* file must be set to **"true"**.			

The corresponding hardware configuration is defined in the project definition file as shown below.

```
<hardware>
  <uart index="1" baud="115200" flowcontrol="true" rts_pin="PA3" cts_pin="PA2" bgapi="true"/>
</hardware>
```

WSTK delivered BGM111 and BGM113 Modules

The BGM111 Bluetooth Smart Modules supplied as part of the Blue Gecko Bluetooth Smart Module Wireless Starter Kit SLWSTK6100A (WSTK) have a factory installed firmware created from the example project **sensor**, which is also stored in the **example\sensor** folder in the SDK. This project file has the hardware configuration listed in the following table.

Note: In case DFU is required the UART settings are the same as in the table below, determined in the *hardware* section of the **"sensor"** project.

Note: If you have already loaded another example project or a project of your own the UART settings might differ from the ones listed below in the table. In such cases the Module will be configured according to the parameter values determined in the said project files hardware section. If those values are not known, DFU cannot be used. The only way to flash the firmware then is to use the J-Link debugger.

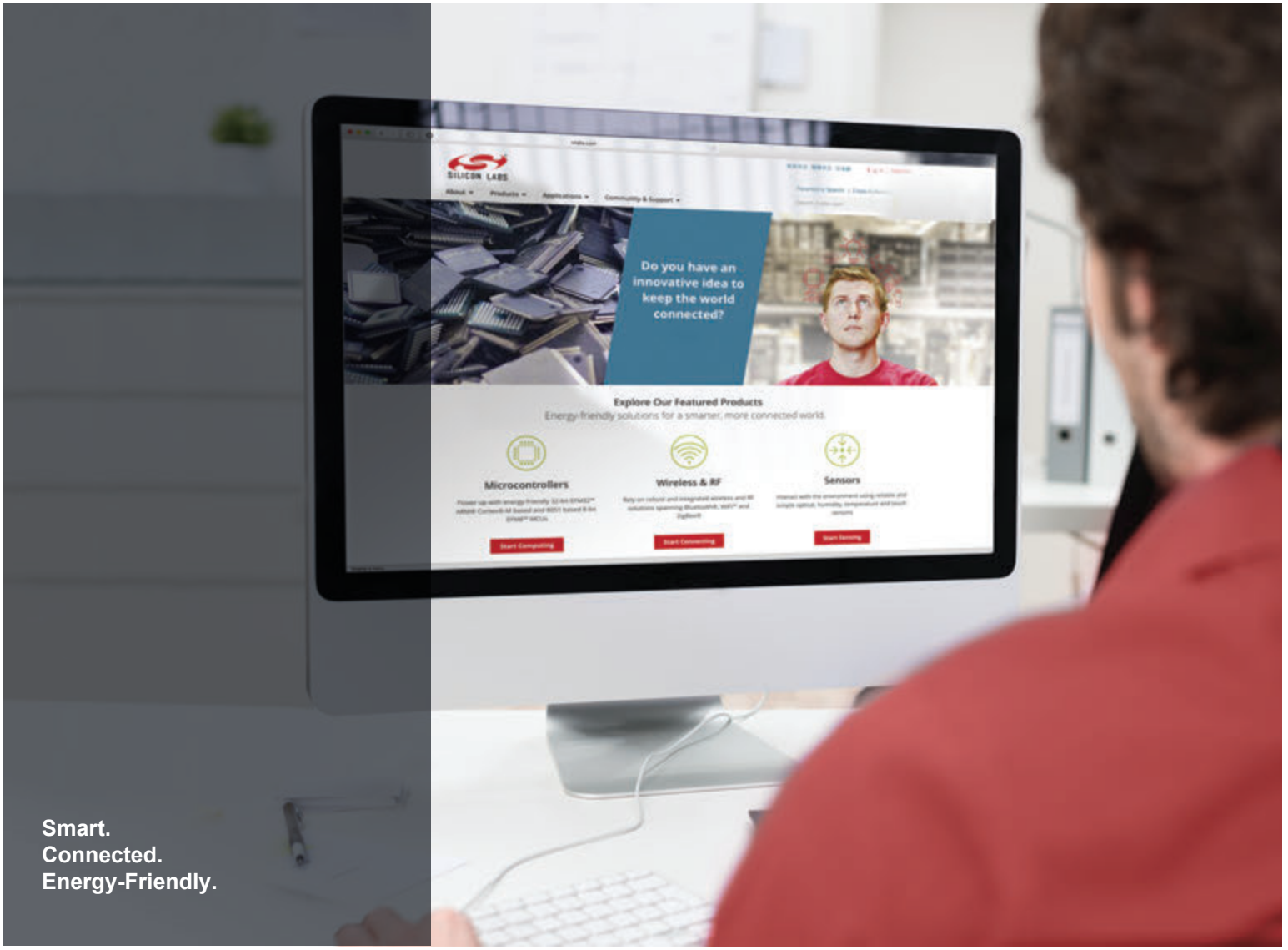
Feature	Setting	Value	Notes
RHT sensor	I2C	enabled	This I2C is used to communicate with the RHT sensor on the WSTK main board.
	Pins	• **SCL** PC11 • **SDA** PC10	
WSTK	vcom	enabled	Virtual sensor communication support enabled, BGAPI UART is available via the USB CDC interface on the WSTK main board.
	sensor	enabled	RHT sensor support enabled.
Note: The UART is disabled in the WSTK default demo.			

The corresponding hardware configuration is defined in the project definition file as shown below.

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

The UART is disabled in the demo application and BGScript is used to define GPIO pins PF6 and PF7 as output and input correspondingly in the *peripheral.bgs* file as shown below.

```
# Configure GPIO as push-pull output
call hardware_configure_gpio(5,6,hardware_gpio_mode_push_pull,1)
# Configure GPIO as push-pull input
call hardware_configure_gpio(5,7,hardware_gpio_mode_input_pull,1)
```



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