UG161: WGM110 Wi-Fi ® Module Configuration Guide

This document describes how to define and configure software projects for the Silicon Labs Wizard Gecko WGM110 Wi-Fi® Module. It shows how to include the necessary resources in the project, how to configure hardware interface settings, and how to build projects and flash the built image files into the WGM110 Module using either the BGBuild command-line tool or the GUI based BGTool.

KEY POINTS
- Starting a software project
- Including the necessary resources
- Configuring hardware interface settings
- Building projects
- Flashing image files

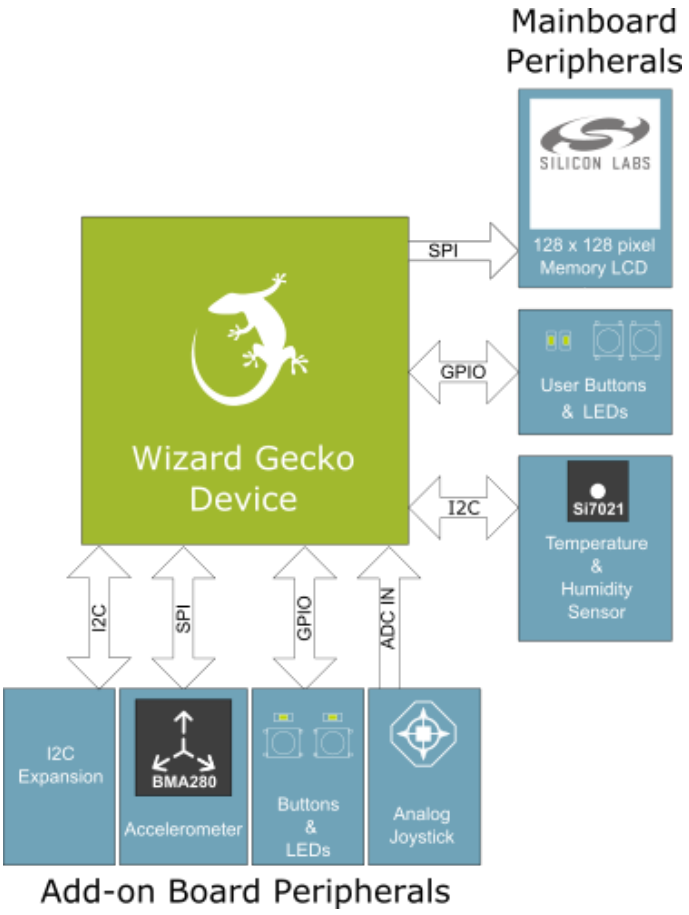


Table of Contents

1. Project Structure

The flowchart below illustrates the Wi-Fi software project structure and the mandatory and optional resources required. The structure is relatively simple and consists of the following components:

- Project file
- Hardware configuration file
- HTTP server files (optional HTML, CSS, etc. files for the built-in HTTP server)
- X.509 certificates (optional certificates to be used with TLS and/or WPA enterprise)
- BGScript application code (optional)

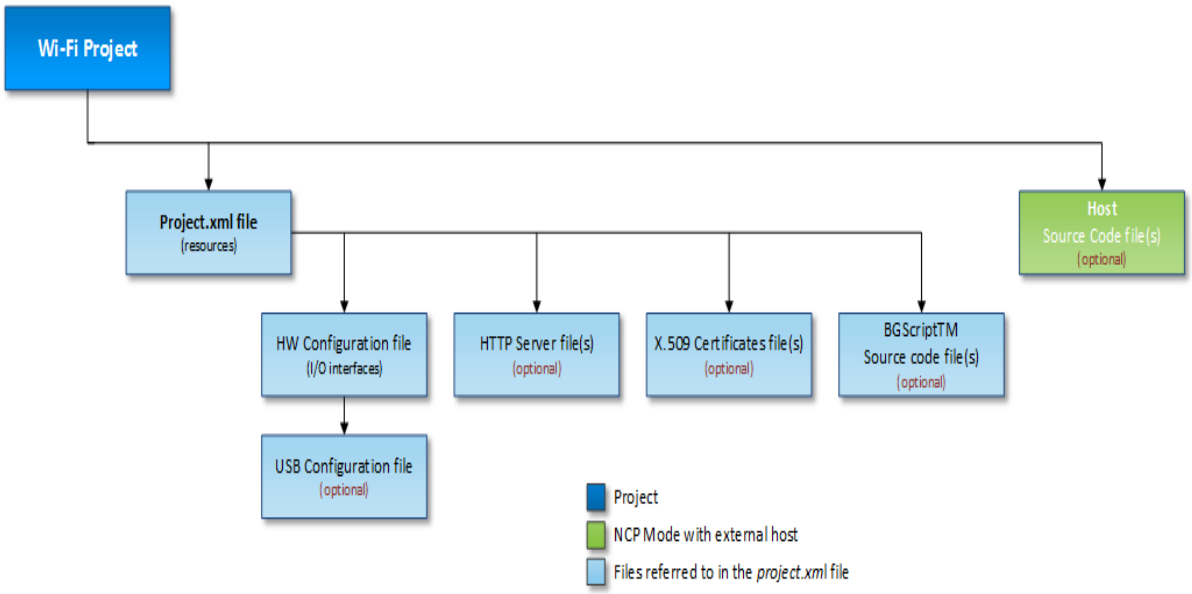


Figure 1.1. Wizard Gecko Module Project Definition

1.1 Project File

The Project file 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 (UART, SPI, I2C, and GPIO used by the project) and defines their physical locations (pins) and their settings.

1.2.1 USB Configuration File

This optional file is used to configure the USB port of the WGM110 Module. For further details see the 3.8 <usb> section.

1.3 HTTP Server Files

The Wi-Fi software stack contains a built-in HTTP server, which can be used to host and display content such as HTML, CSS, JavaScript and other types of files. These files can be compiled directly into the Wi-Fi firmware image. In case you want to do this, you need to define the files in the project file and also include the desired content in the actual project contents.

1.4 Certificates

The Wi-Fi software stack contains a built-in TLS/SSL client and support for WPA enterprise security. Both technologies use X.509 certificates for authentication, and it is possible to include the certificates in the Wi-Fi firmware. In case you want to do this, you need to define the certificates in the project file and include the actual X.509 certificate files in the actual project contents.

1.5 BGScript Application Code

BGScript is an event-driven, application programming language, which allows simple applications to be embedded into Silicon Labs' Wi-Fi devices. When BGScript is used to implement the application logic, the source files need to be included in the project file.

For more information about the BGScript scripting language, go to the Silicon Labs web page at http://www.silabs.com/start-wgm and download the *UG170: Wizard Gecko BGScript™ User's Guide*.

2. Project File Syntax

The project file (typically **project.xml** or **project.bgproj**) is the file that describes all the components included in your Wi-Fi project. This section describes the syntax and the options which can be used in the project file.

2.1 <project>

The XML element **<project>** is the root element of the project definition. It includes the hardware device type the project is intended for as an attribute. All the other definitions need to be inside the **<project>** and **</project>** tags.

Attribute	Type	Description
device	string	This attribute defines the hardware device type the project is intended for. **Values:** **wgm110**: Wizard Gecko WGM110 Wi-Fi Module

Example: A project definition for WGM110 Wi-Fi Module

```
<project device="wgm110">
 …
</project>
```

2.2 <hardware>

The XML element **<hardware>** defines the location of the hardware configuration file. Hardware configuration can also be directly defined in the project definition file, in which case this element acts as the parent element.

Attribute	Type	Description
in	string	This attribute defines the XML file which contains the hardware configuration of the project.

Example: Defining hardware configuration file

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

Example: Defining hardware configuration directly in the project definition

```
<hardware>
...
</hardware>
```

2.3 <scripting>

The XML elements **<scripting>** and <**script**> are used to define the location of the BGScript application code file.

Attribute	Type	Description
in	string	This attribute defines the .bgs file which contains the BGScript application code. This definition is only needed for projects where BGScript applications are used.

Example: Defining BGScript file to include in the project

```
<scripting>
<script in="wgm110demo.bgs"/>
</scripting>
```

2.4 <image>

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

Attribute	Type	Description
out	string	This attribute defines the name of the firmware output file (.bin).
out_hex	string	This attribute defines the name of the firmware output file (.hex).

Example: Naming the firmware output file

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

2.5 <bootloader>

The XML element **<bootloader>** defines the location of the bootloader to include in the project.

Attribute	Type	Description
in	string	This attribute defines the bootloader binary to include in the project. **Default:** fw/boot.juo

Example: Defining the bootloader to use

```
<bootloader in="fw/boot.juo"/>
```

2.6 <files>

The XML element **<files>** is the parent element of all the files included in the project.

Attribute	Type	Description
-	-	-

Example: Defining files to include in the project

```
<files>
...
</files>
```

2.6.1 <file>

The XML element **<file>** defines a file to include in the project. The file is only visible to the built-in HTTP server. All the **<file>** elements must be defined inside the **<files>** parent element.

Attribute	Type	Description
path	string	This attribute defines the file to include in the project.
include_path	boolean	This attribute defines whether the path to the file is added to the file name. **Values:** **true:** path is added to the file name **false:** path is discarded **Default:** false

Example: Including a file in the project

```
<files>
<file path="index.html"/>
</files>
```

Example: Including a file from a subdirectory, path added to the file name

```
<files>
<file path="img/image.jpg" include_path="true"/>
</files>
```

2.7 <certificates>

The XML element **<certificates>** is the parent element of all the X.509 certificates included in the project either for TLS/SSL or WPA Enterprise.

Note: All certificates are loaded to RAM memory when TLS or WPA Enterprise connection is established. Memory consumption of the certificates can affect module operation.

Attribute	Type	Description
-	-	-

Example: Defining certificates to include in the project

```
<certificates>
...
</certificates>
```

2.7.1 <certificate>

The XML element **<certificate>** defines an X.509 certificate file to include in the project. All the **<certificate>** elements must be defined inside **<certificates>** parent element.

Attribute	Type	Description
path	string	This attribute defines the X.509 certificate file to include in the project.
format	string	This attribute defines the format of the certificate file. **Values:** **pem:** certificate is in PEM format **der:** certificate is in DER format **Default:** pem

Example: Including an X.509 certificate in PEM format

```
<certificates>
<certificate path="certificate.pem" format="pem"/>
</certificates>
```

2.8 Project File Examples for WGM110

The example below shows how to create a project file for WGM110 Wizard Gecko Wi-Fi module when the application code is implemented with BGScript scripting language.

Example 1: A BGScript-Based Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<project device="wgm110">
  <scripting>
    <script in="wifi.bgs"/>
  </scripting>
  <hardware in="hardware.xml"/>
  <image out="wifi.bin" out_hex="wifi.hex"/>
</project>
```

The example below, on the other hand, shows how to create a project where the application runs on an external host.

Example 2: A Host-Based Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<project device="wgm110">
  <hardware in="hardware.xml"/>
  <image out="wifi.bin" out_hex="wifi.hex"/>
</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.

Interfaces lacking the "enable" attribute in their tag are enabled only when the tag itself exists in the hardware configuration.

3.1 <hardware>

The XML element **<hardware>** is the parent element of all the hardware configuration elements in the project.

Example: Defining project hardware configuration

```
<hardware>
...
</hardware>
```

3.2 <adc>

The XML element **<adc>** and its attributes are used for enabling ADC and configuring the ADC settings.

Attribute	Type	Description
reference	integer	This attribute defines ADC reference voltage configuration. **Values:** **0:** internal 1.25 V reference **1:** internal 2.5 V reference **2:** buffered VDD **3:** internal differential 5 V reference **4:** single ended external reference connected to ADC CH6 pad **5:** differential external reference connected to ADC CH6 & CH7 pads **6:** unbuffered 2 x VDD **Default:** 0

Example: Enabling ADC using internal 1.25 V reference voltage

```
<adc reference="0" />
```

3.3 <i2c>

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

Attribute	Type	Description
channel	integer	This attribute defines the I^2C module used. **Range**: 0 - 1 **Default**: 0
baud	integer	This attribute defines the I^2C bit rate. **Range**: 1 - 100000 **Default**: 100000
location	integer	This attribute defines the location of the I^2C pads. **Range:** 0 - 6 **Default:** 0

Example: Enabling I2C1 at Location 2

```
<i2c channel="1" location="2"/>
```

Note: The RHT (Relative Humidity and Temperature) sensor on the WSTK main board uses the pads defined by Channel 1 Location 2 used in the example above.

3.4 <gpio>

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

Attribute	Type	Description
port	integer	This attribute defines the GPIO port to configure. Values: **0:** PA **1:** PB **2:** PC **3:** PD **4:** PE **5:** PF **Default:** 0
pin	integer	This attribute defines the GPIO pad within the defined port to configure. **Range:** 0 – 15 **Default:** 0
mode	integer	This attribute defines the GPIO mode of the pad. **Values:** **1:** input with optional glitch filter **2:** input with pull-down or pull-up **3:** input with glitch filter and pull-down or pull-up **4:** output **Default:** 1
out	integer	This attribute defines the default function of the pad. The function depends on the mode attribute used. **Values in mode 1:** **0:** glitch filter disabled **1:** glitch filter enabled **Values in mode 2:** **0:** pull-down enabled **1:** pull-up enabled **Values mode 3:** **0:** pull-down enabled **1:** pull-up enabled **Values mode 4:** **0:** output low **1:** output high **Default:** 0

Attribute	Type	Description
interrupt	string	This attribute defines whether interrupt is enabled on the pad and select rising, falling or both edge triggering of the pad. **Values**: **none**: interrupt disabled **rising**: interrupt generated on rising edge **falling**: interrupt generated on falling edge **both**: interrupt generated on both rising and falling edges **Default**: none
Example: Using PC1 as an input with glitch filter enabled, interrupt enabled on the falling edge. <gpio port="**2**" pin="**1**" mode="**1**" out="**1**" interrupt="**falling**"/>		
Example: Using PD5 as an output with the output set high. <gpio port="**3**" pin="**5**" mode="**4**" out="**1**"/>		

3.5 <spi>

The XML element **<spi>** and its attributes are used to configure the SPI interface settings.

Attribute	Type	Description
channel	integer	This attribute defines the USART module used. **Range:** 0 - 1 **Default:** 0
baud	integer	This attribute defines the bit rate of the SPI master. When the interface is configured as an SPI slave, this attribute has no effect. **Range**: 9600 – 6000000 **Default**: 1000000
api	boolean	This attribute defines whether this interface is used as a BGAPI host interface or as a transparent data interface. BGAPI can be enabled only for an SPI slave. **Values:** **true:** BGAPI is enabled on the interface **false:** transparent data is enabled on the interface **Default:** false
mode	string	This attribute defines whether this interface is an SPI master or an SPI slave. **Values:** **master:** SPI master mode **slave:** SPI slave mode **Default**: slave
slave_select	boolean	This attribute defines whether slave select is enabled. **Values:** **true:** slave select is enabled **false:** slave select is disabled **Default**: false
clock_idle_polarity	string	This attribute defines the polarity of the SPI clock signal. **Values:** **low:** idle state for the clock signal is low **high:** idle state for the clock signal is high **Default**: low
clock_edge	integer	This attribute defines when the SPI data lines are sampled and set up. **0**: sampled when the clock signal changes from active to idle, set up when the clock signal changes from idle to active **1**: sampled when the clock signal changes from idle to active, set up when the clock signal changes active to idle **default**: 0

Attribute	Type	Description
notify	boolean	This attribute defines whether a GPIO pad is used to notify the SPI master the SPI slave has buffered data for the master. This attribute is valid only when the interface has been configured as a BGAPI host interface. **Values:** **true:** notify support is enabled **false:** notify support is disabled **Default**: false
notify_port	integer	This attribute defines the GPIO port used for the notify support. **Values:** **0:** PA **1:** PB **2:** PC **3:** PD **4:** PE **5:** PF **Default:** 0
notify_pin	integer	This attribute defines the GPIO pad used for the notify support. **Range:** 0 - 15 **Default:** 0
location	integer	This attribute defines location of the USART pads. **Range:** 0 - 6 **Default:** 0

Example: Using USART0 at Location 0 as a BGAPI host interface without slave select, notify support enabled at PD6

```
<spi index="0" location="0" api="true" notify="true" notify_port="3" notify_pin="6"/>
```

Example: Using USART1 at Location 1 as an SPI master with an external sensor, bitrate 100000 bps, slave select enabled

```
<spi index="1" location="1" baud="100000" mode="master" slave_select="true"/>
```

SPI Modes:

SPI Mode	*clock_idle_polarity*	*clock_edge*
0	low	1
1	low	0
2	high	1
3	high	0

3.6 <timer>

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

Attribute	Type	Description
index	integer	This attribute defines the timer being configured. **Range:** 0 - 1 **Default:** 0
location	integer	This attribute defines location of the timer pads. **Range:** 0 - 6 **Default:** 0
prescale	integer	This attribute defines the prescale factor used to derive the timer clock signal from the peripheral clock signal. The timer counter is increased by one at every clock cycle. Values: **1:** corresponds to 48 MHz **2:** corresponds to 24 MHz **4:** corresponds to 12 MHz **8:** corresponds to 6 MHz **16:** corresponds to 3 MHz **32:** corresponds to 1500 kHz **64:** corresponds to 750 kHz **128:** corresponds to 375 kHz **256:** corresponds to 187.5 kHz **512:** corresponds to 93.75 kHz **1024:** corresponds to 46.875 kHz **Default:** 1
top_value	integer	This attribute defines the top value of the timer counter. When the timer has counted to this value, it wraps to around to zero and continues counting. **Range:** 0 - 65535 **Default:** 65535

Example: Enabling TIMER0 at location 0 with prescale factor 512, top value 32767

```
<timer index="0" location="0" prescale="512" top_value="32767"/>
```

3.7 <uart>

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

When using WGM110 UART in transparent mode (api="false"), we highly recommend to use flow control to avoid the possibility of losing data. Furthermore, we recommend to use DMA on the receiving host, because the SW emulated flow control used in the WGM110 is not as robust as HW flow control and therefore when CTS is raised, the module will most likely still send at least one extra byte. Also using lower baud rates (< 1Mbps) can help.

Attribute	Type	Description
channel	integer	This attribute defines the USART module used. **Range:** 0 - 1 **Default:** 0
baud	integer	This attribute defines the UART bit rate. **Range**: 9600 – 6000000 **Default**: 115200
stopbits	integer	This attribute defines the number of stop bits used. **Values:** **1:** one stop bit **2:** two stop bits **Default:** 1
location	integer	This attribute defines the location of the USART pads. **Range:** 0 - 6 **Default:** 0
handshake	boolean	This attribute defines whether RTS/CTS is enabled. **Values:** **true:** RTS/CTS is enabled **false:** RTS/CTS is disabled **Default:** false
api	boolean	This attribute defines whether this interface is used as a BGAPI host interface or as a transparent data interface. **Values:** **true:** BGAPI is enabled on the interface **false:** transparent data is enabled on the interface **Default:** false
parity	string	This attribute defines usage of the parity bit. **Values:** **none:** no parity bit **odd:** odd parity bit **even:** even parity bit **Default:** none

Example: Using USART0 at Location 0 as a BGAPI host interface without RTS/CTS, bit rate 115200 bps

```
<uart channel="0" location="0" baud="115200" api="true" handshake="false"/>
```

Example: Using USART1 at Location 1 for transparent data, bitrate 6000000 bps, RTS/CTS enabled

```
<uart channel="0" location="1" baud="6000000" api="false" handshake="true">
```

3.8 <usb>

The XML element **<usb>** and its attributes are used to configure the USB interface settings.

Attribute	Type	Description
api	boolean	This attribute defines whether this interface is used as a BGAPI host interface or as a transparent data interface. **Values:** **true:** BGAPI is enabled on the interface **false:** transparent data is enabled on the interface **Default:** false
descriptor	string	This attribute defines the XML file that contains the USB descriptor definition for this interface. See the *usbcdc* example in the SDK on how to define the descriptor. **Default:** This is a mandatory attribute

Example: Using USB as a BGAPI host interface, USB descriptor defined in *cdc.xml*

```
<usb api="true" descriptor="cdc.xml"/>
```

3.9 <sdhc>

The XML element **<sdhc>** and its attributes are used to configure the SD card settings.

Attribute	Type	Description
enable	boolean	This attribute defines whether SD card support is enabled. **Values:** **true:** SD card support is enabled **false:** SD card support is disabled **Default:** false
usart	integer	This attribute defines the USART module used. **Range**: 0 - 1 **Default**: 1
usart_loc	integer	This attribute defines location of the USART pads. **Range**: 0 – 6 **Default**: 1
cs_port	integer	This attribute defines the GPIO port assigned to SD card CS line. Values: **0:** PA **1:** PB **2:** PC **3:** PD **4:** PE **5:** PF **Default**: 3
cs_pin	integer	This attribute defines the GPIO pad assigned to SD card CS line. **Range**: 0 – 15 **Default**: 6

Example: Enabling SD card, using USART1 at Location 1 with CS line connected to PD6

```
<sdhc enable="true" usart="1" usart_loc="1" cs_port="3" cs_pin="6"/>
```

3.10 <kit>

The XML element **<kit>** and its attributes are used to configure WSTK specific functionality. Including this element in the hardware file configures the module to work correctly with the WSTK main board.

Attribute	Type	Description
vcom	boolean	This attribute defines whether WSTK virtual serial communication support is enabled. If enabled, USART0 at location 0 will be routed to the WSTK main board. UART has to be configured separately. **Values:** **true:** enable virtual serial communication support **false:** disable virtual serial communication support **Default:** false
sensor	boolean	This attribute defines whether WSTK RHT sensor support is enabled. If enabled, I2C1 at location 2 will route to the WSTK main board. I2C has to be configured separately. **Values:** **true:** enable RHT sensor support **false:** disable RHT sensor support **Default:** false
lcd	boolean	This attribute defines whether WSTK LCD support is enabled. If enabled, USART1 at location 1 will route to the WSTK main board. SPI has to be configured separately. **Values:** **true:** enable LCD support **false:** disable LCD support **Default:** false

Example: Enabling the virtual serial communication and the RHT sensor support

```
<kit vcom="true " sensor="true"/>
```

Note: The above example applies for WGM110 and the SLWSTK6120A main board.

3.11 Hardware Configuration File Examples

The example below shows how to create a hardware configuration file for a Wizard Gecko WGM110, 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 3: Hardware Configuration with both UART and I^2C Enabled

```
<hardware>
  <uart channel="0" baud="115200" api="false" handshake="false"/>
  <i2c channel="1" location = "2"/>
</hardware>
```

The example below on the other hand shows how to create a hardware configuration where the application runs on an external host.

Example 4: A Host-based Project with BGAPI Serial Protocol Enabled

```
<hardware>
  <uart channel="0" baud="115200" api="true" handshake="false"/
</hardware>
```

3.12 Hardware Configuration File and DFU

The use of DFU (Device Field Upgrade) functionality requires that at least one of the available host interface options (UART, SPI, or USB) is defined in the hardware section of the Project and that the *api* parameter is set to **"TRUE"** which will define that BGAPI is enabled on the corresponding interface. In such cases DFU will be usable through the defined host interface with settings declared with the parameters in the hardware section. In case more than one host interface is defined for BGAPI use, DFU functionality will be usable through the host interface in the following order: UART - SPI - USB.

Example: Configuring to allow DFU

```
<hardware>>
    <!--UART-->
    <uart channel="0" baud="115200" api="true" handshake="false"/>

    <!--SPI-->
    <spi channel="1" location="1" mode="slave" slave_select="true" api="true"
    notify="true" notify_port="1" notify_pin="11"/>

    <!--USB-->
    <usb descriptor="cdc.xml" api="true"/>
</hardware>
```

Note: If the hardware section does not contain any host interface definition, the only way to flash the Module is to use the **J-Link Debugger**.

4. WGM110 Factory Default Configuration

The factory default configuration of the WGM110 Module depends on whether the WGM110 Module was delivered as part of a reel order or as part of the Wizard Gecko Wi-Fi Module Wireless Starter Kit SLWSTK6120A (WSTK), in which case the WGM110 Module is pre-installed on the Wizard Gecko WGM110 Wi-Fi Module Radio Board BRD4320A PCB.

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

Reel delivered WGM110 Modules

The WGM110 Wi-Fi 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	0	This UART gives access to BGAPI serial protocol, which can be used to control the the module from an external host.
	location	0	
	baud rate	115200	
	data bits	8	
	stop bits	1	
	RTS/CTS	enabled	

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

```
<hardware>
  <uart channel="0" baud="115200" api="true" handshake="true"/>
</hardware>
```

WSTK delivered WGM110 Modules

The WGM110 Wi-Fi Modules supplied as part of the Wizard Gecko Wi-Fi Module Wireless Starter Kit SLWSTK6120A (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
BGAPI UART	USART	0	This UART gives access to BGAPI serial protocol, which can be used to control the module from an external host.
	location	0	
	baud rate	115200	
	data bits	8	
	stop bits	1	
	RTS/CTS	disabled	
RHT sensor	I2C	1	This I2C is used to communicate with the RHT sensor on the WSTK main board.
	location	2	
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.

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

```
<hardware>
  <uart channel="0" baud="115200" api="true" handshake="false"/>
  <i2c channel="1" location="2"/>
  <kit vcom="true" sensor="true"/>
</hardware>
```

5. Building a Project

The project is built into a firmware image using the **bgbuild.exe** compiler. The compiler can be executed using BGTool software or directly from the command-line using Windows Command Line Prompt (**cmd.exe**). The examples below demonstrate how to build a project.

5.1 Building a Project with BGBuild

To build the project from the command-line using BGBuild:

1. Open Windows Command Prompt.
2. Navigate (using the command **'cd'**) to the folder where the project is located.
3. Run the **bgbuild.exe** compiler as shown in the figure below, giving the project file as a parameter. The syntax for the bgbuild compiler is: **bgbuild.exe *<project file>***.
4. If the image file was built successfully, bgbuild will output the message **"ALL OK"**.

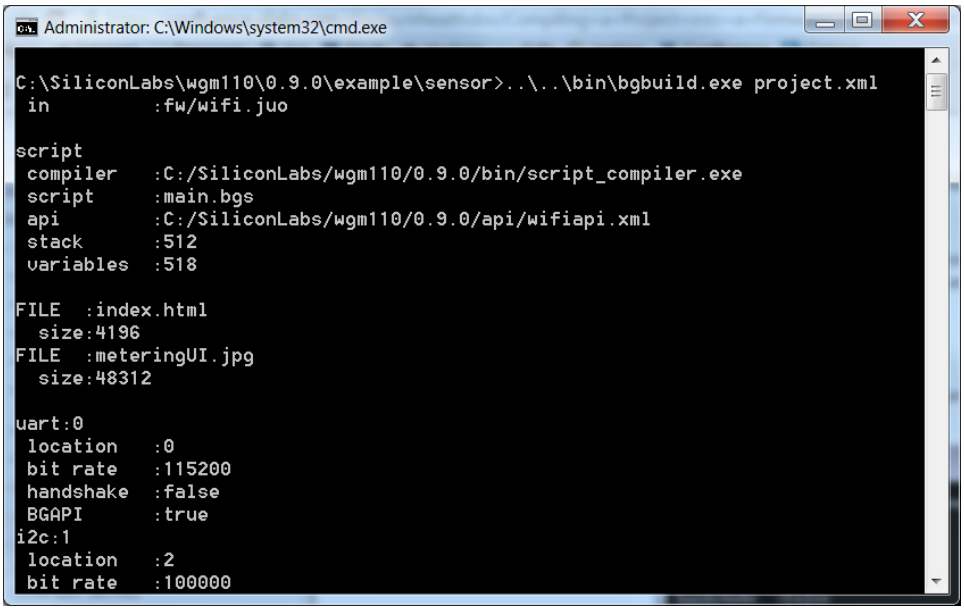


Figure 5.1. Building the Project with bgbuild.exe

5.2 Building a Project with BGTool

You can use BGTool to build a project into a ***.bin*** file.

To build the project using BGTool:

1. Launch BGTool.
2. Click the [**Upload tool**] tab.
3. Click [**Browse**] and navigate to your project.
4. Click the [**Build**] button.
5. If the project was built successfully, BGTool will show **"Build has been completed successfully"** and the **BGBuild Output Window** will output **"ALL OK"** (you may have to scroll the output window to see the message).

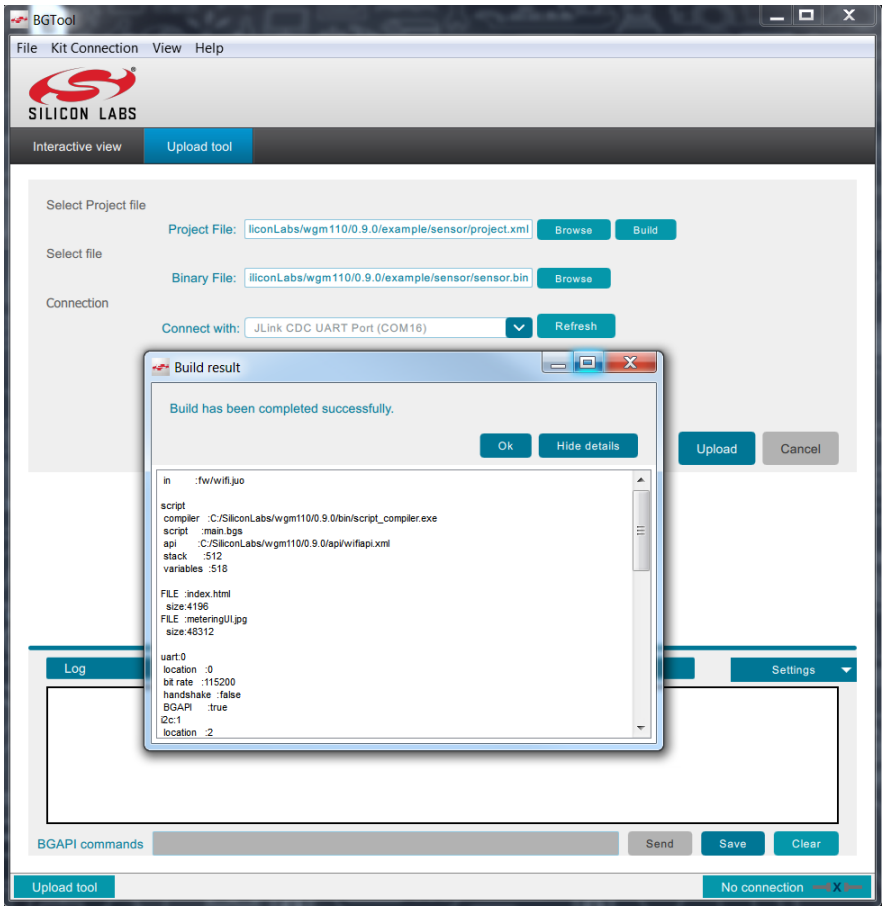


Figure 5.2. Building the Project with BGTool

For more information on how to use BGTool, go to Silicon Labs web page at http://www.silabs.com/start-wgm and download the *UG160: Wizard Gecko BGTool™ User's Guide*.

5.3 Verifying Configuration from BGBuild or BGTool Compiler Output

When you have built the project using either the command-line BGBuild (bgbuild.exe) or the BGTool, the compiler will output the project configuration, list the BGScript and BGAPI files and the hardware interface settings. You can use this information to verify the correctness of your configuration files and how you have configured the hardware settings for the Module.

The table below defines the compiler output details as listed either in the output of the BGBuild or BGTool.

Table 5.1. BGBuild and BGTool Compiler Output Details

Output	Description
compiler	Location of the BGScript compiler used in the compilation.
script	Main BGScript source code file. This is displayed ONLY if BGScript is defined in the project configuration file.
api	Location of the used API definition file.
stack	Amount of stack space allocated for BGScript applications in bytes.
variables	Indicates the amount of memory reserved by variables in bytes.
FILE	Name and size in bytes of a file included in the project.
CERT	Name, size in bytes and fingerprint of an X.509 certificate included in the project.
uart	UART interface settings **index:** USART ID **location:** location of the USART pads **bit rate:** UART baud rate **handshake:** RTS/CTS flow control configuration **BGAPI:** BGAPI serial protocol configuration
spi	SPI interface settings **index:** USART ID **location:** location of the USART pads **bit rate:** SPI bit rate **BGAPI:** BGAPI serial protocol configuration
usb	USB interface settings **BGAPI:** BGAPI serial protocol configuration
i2c	I2C interface settings **index:** I2C ID **location:** location of the I2C pads **bit rate:** I2C bit rate
adc	ADC settings **reference:** ADC reference voltage
timer	TIMER settings **index:** TIMER ID **location:** location of the TIMER pads **precale:** clock signal prescale factor **top value:** top value of the timer
sdhc	SD card settings **usart:** USART ID **location:** location of the USART pads

6. Flashing Project Image Files into the Module

After the project and its configuration files have succesfully been compiled, the created image file needs to be flashed into the Module. Flashing can be done using either the command-line tool `wifi_dfu` or the BGTool. The following sections describe both methods.

6.1 Flashing a Project with wifi_dfu

When you have compiled the project file with the BGBuild program (***bgbuild.exe***) using the command-line, you can flash the module with the firmware image.

To flash the module using ***wifi_dfu*** tool:

1. Use the command-line to run ***wifi_dfu.exe <com port> <port bitrate> [nortscts] <dfu image.bin>***. The com port is presented in Windows as ***\\.\COM8***. The port bitrate is the same as it was before the flashing.

   Parameter ***nortscts*** must be set in case the communication port is not using hardware handshaking.

   Note: Files with filename extension ***.hex*** can't be loaded using ***wifi_dfu***.

   Note: Don't set **address** in command line parameters

2. After the image file has been flashed into the Module, the ***wifi_dfu*** program is automatically terminated. After a successful loading ***wifi_dfu*** outputs in command line view **"booting in Application"** and if the flashing has failed **"booting in DFU"**.

If the flashing of the Module was succesful, BGBuild will output a window as shown in the figure below:

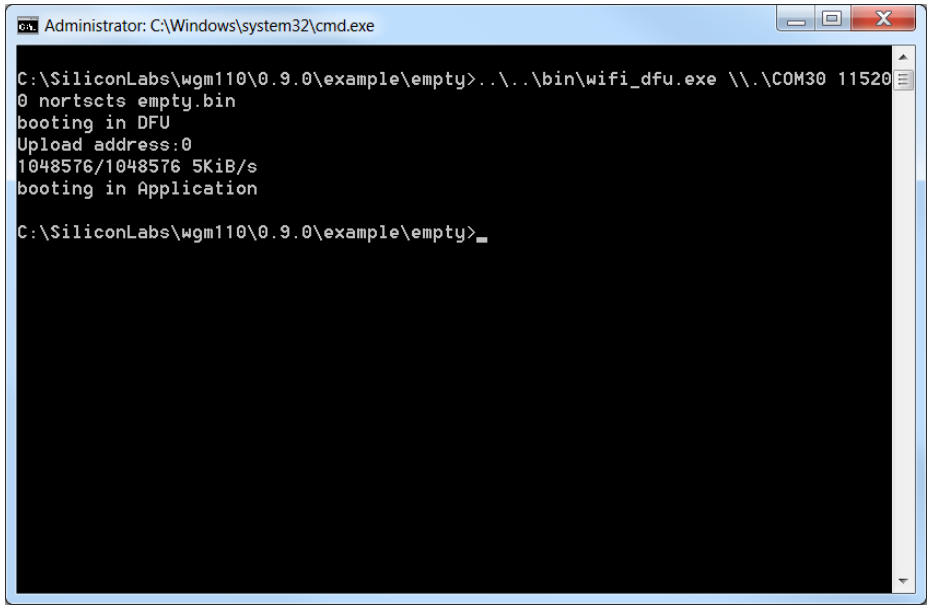


Figure 6.1. Succesful flashing using BGBuild

6.2 Flashing a Project With BGTool

When you have compiled the project file with BGTool, you can use BGTool to flash the Module with the ***.bin*** file.

To flash the Module using BGTool follow one of two scenarios:

- J-Link Upload:
  1. Select Upload Method: *J-Link Upload* (default selection).
  2. Select the ***.bin*** file you want to flash into the Module.
  3. Click the [**Upload**] button.
  4. Wait for the update to finish. After a succesful update, BGTool will output a window as shown in the figure below.

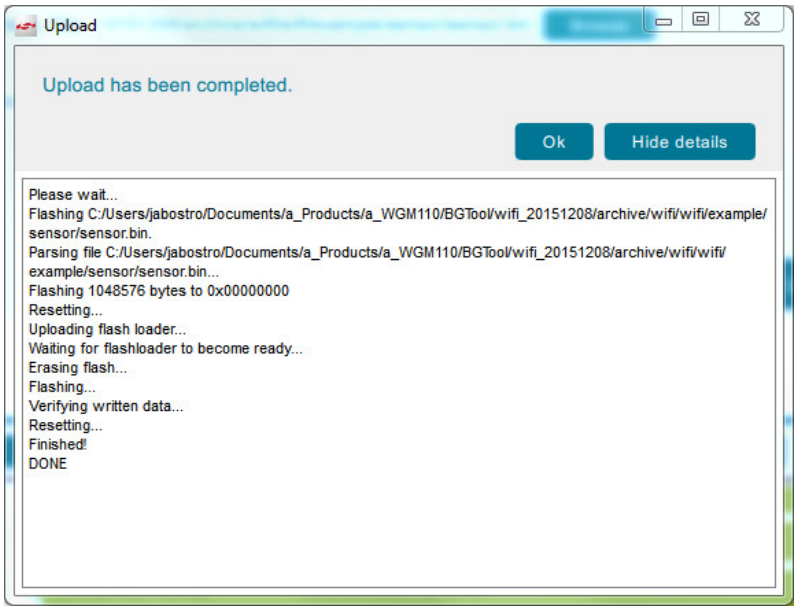


Figure 6.2. Succesful Flashing Using BGTool (J-Link Upload)

- UART DFU Upload:
  1. Select Upload Method: *UART DFU Upload*.
  2. Select the ***.bin*** file you want to flash into the Module.
  3. Select COM port.
  4. Select baud rate.
  5. *Optional:* Select *No Rts/Cts*.
  6. Click the [**UART DFU Upload**] button.
  7. Wait for the update to finish. After a succesful update, BGTool will output a window as shown in the figure below.

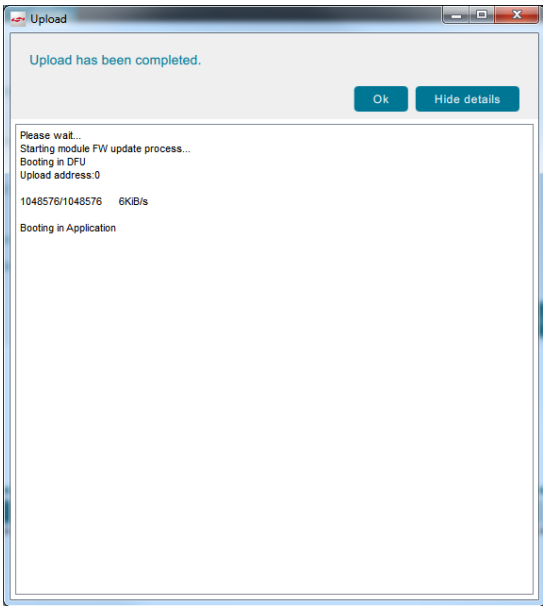


Figure 6.3. Succesful Flashing Using BGTool (UART DFU Upload)

7. Revision History

7.1 Revision 1.3

March 31st, 2017

Changes:

- Updated hardware configuration to allow GPIO settings, added SPI Modes table in section 3.5.
- Updated Table 5.1. BGBuild and BGTool Compiler Output Details.
- Updated 6.2 Flashing a Project With BGTool chapter.
- Note added into Certificates section for RAM consumption.

7.2 Revision 1.2

September 16th, 2016

Changes: Updated UART RTS/CTS to be enabled as the default factory setting.

7.3 Revision 1.1

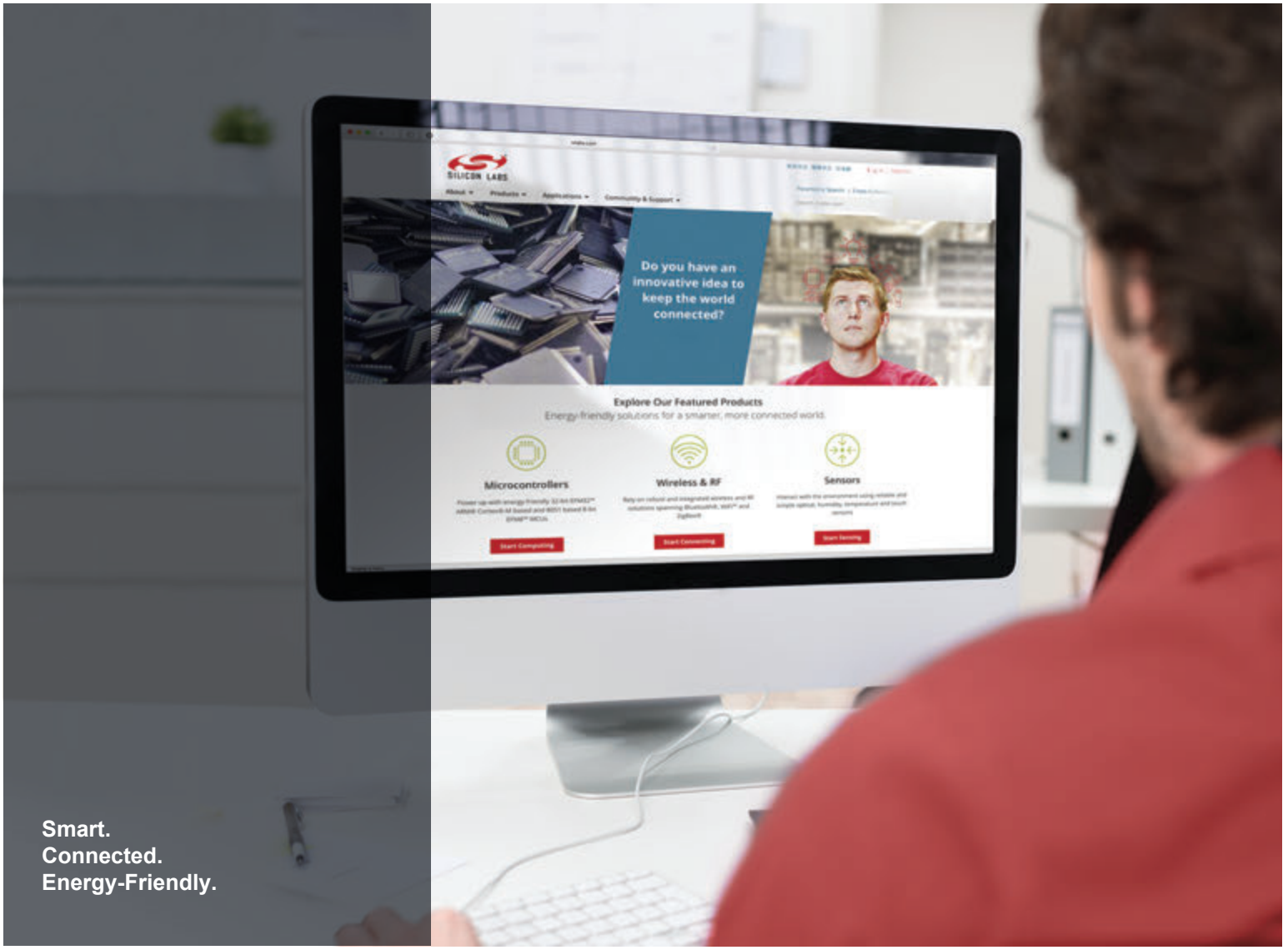
May 23rd, 2016

Changes: Figures updated and table layouts changed, added *"SPI master mode"* value to table in Section 5.3.

7.4 Revision 1.0

February 22nd, 2016

Initial release.



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