WF121 CONFIGURATION

DEVELOPER GUIDE

Thursday, 24 November 2016

Version 3.0

Table of Contents

1 Version History

Version	Comments
1.0	Updated to match v.1.0 software
1.1	Sleep mode configuration documentation improved
1.2	HTTP server and other configurations added
1.3	WF121 factory default configuration added
1.4	Minor improvements
1.5	Added new Wi-Fi software profile
1.6	Added firmware compilation and installation instructions
1.7	Updated to match v.1.2 software
1.8	Updates for Wi-FI software v. 1.2.2 and editorial changes Watchdog timer (WDTPS) configuration added
1.9	Ethernet configuration added
2.0	Updated for 1.3 SW compatibility with the following changes: • <certificates> configuration added for including server certificates • <sdhc> configuration added for including an external SD card • External crystal, e.g., for host communication (USB, UART) configurable under <bootloader> • <software> configuration removed as SW profile not configurable anymore • Embedded HTTP server chapter removed as from WiFi 1.3 SW onwards, the embedded web and HTTP server design not tied to specific files and PS keys
2.1	Updated to match v1.3 software • SPI DFU boot loader added under <bootloader>
2.2	Updated to match v1.3 release software • Removed X509 certificate content • Added warning of SPI data mode unreliability
3.0	Updated for v1.4 release software • Added advice of UART flow control usage • Added description of Microchip IPE usage • Minor updates and corrections • Text spelling and formatting corrections • Changes in description of ADC pin configuration

2 Project configuration file

The project configuration file (typically **project.xml**) is the file that describes all the components included in a Wi-Fi software and hardware project. Typically the project configuration file contains at least the following files:

- Hardware configuration file: **hardware.xml**
- Optional BGScript application code: **script.bgs**
- Optional HTTP server files

The project configuration file also defines names of generated firmware files and included bootloader file.

The project file itself is a simple XML file with just few tags on it, which are described below.

2.1 <hardware>

This tag is used to define the XML file for the hardware (such as UART or USB) configuration.

XML attribute	Description
hardware	Describes the XML file, which contains the hardware configuration of your Wi-Fi device. **Example:** *<hardware in="hardware.xml" />* Hardware configuration can also be directly defined in the **project.xml** file. **Example:** *<hardware>* *<uart channel="1" baud="5000000" api="true" handshake="true" />* *</hardware>*

2.2 <scripting>

This optional tag is used to define the file which contains the BGScript application code.

XML attribute	Description
script	Describes the BGScript file, which contains the BGScript code of your standalone Wi-Fi application. If you use BGAPI protocol and a separate host and do not use BGScript code, this attribute can be left out. **Example:** *<scripting>* *<script in="wifi.bgs" />* *</scripting>*

2.3 <image>

This tag is used to define the firmware (HEX or DFU) output file.

XML attribute	Description

XML attribute	Description
image	Describes the firmware output files for the compiler. Both the .dfu and .hex files can be given as parameters **Default:** If this tag does not exists the firmware output files will be named **wifi.hex** and **wifi.dfu.** **Example:** *<image out="buttons.dfu" out_hex="buttons.hex" />*

2.4 <bootloader>

This tag is used to define the binary bootloader configuration (optional).

XML attribute	Description
in	Defines the bootloader used in the Wi-Fi software project. Default bootloader is *boot.juo* which enables DFU upgrades over the BGAPI interface via UART or USB interfaces. In order to use DFU via SPI interface, *spi_boot.juo* bootloader must be used instead. **Default:** *<bootloader in="../../fw/boot.juo" />* **Example:** *<bootloader in="../../fw/boot.juo" />*
WDTPS	Watchdog timer postscaler. Controls maximum sleep time allowed for WF121. Sleep time calculation: 1ms * 2^{WDTPS} **Range:** 0-20 **Default:** <bootloader WDTPS="13" /> **Example:** <bootloader WDTPS="20" /> Module sleeps at maximum of 1ms*2^{20}=1048.576s

XML attribute	Description
oscillator	Frequency of external crystal in MHz **Possible values:** 4, 8, 12, 16, 20, 24 **Default:** *<bootloader oscillator="8" />* *Default value is required when using the internal 8MHz RC oscillator. If an 8MHz external crystal is used, the module will automatically detect and use it instead of the internal oscillator.* **Example:** *<bootloader oscillator="16" />* *Defines module to use an external 16MHz crystal. If an external crystal of 16MHz is not present the module will not work due to wrong divider for the internal 8MHz RC oscillator which would be used instead.*

2.5 <files>

This optional tag is used to include additional files such as HTML and CSS files in the WF121 firmware image.

The **<files>** tag is used to define the individual files such as HTML pages which will be included in the compiled firmware.

XML attribute	Description
file	Defines the files to be included in the firmware and used by the embedded HTTP server. **path** attribute is used to define the path and name of file. Optional **include_path** defines if complete path and name should be used when accessing the HTML page via the embedded web server. **Example:** *<files>* *<file path="web/index.html" include_path="true"/>* *<file path="page.html"/>* *<file path="image.jpg"/>* *<file path="style.css"/>* *</files>* Below is a list of files that are recognized by the HTTP server, with the content type they map to. All other files can be used, but they are taken as binaries while still downloadable. "html" - text/html "htm" - text/html "shtml" - text/html Expires... "shtm" - text/html Expires... "ssi" - text/html Expires... "gif" - image/gif "png" - image/png "jpg" - image/jpeg "bmp" - image/bmp "ico" - image/x-icon "class" - application/octet-stream "cls" - application/octet-stream "js" - application/x-javascript "ram" - application/x-javascript "css" - text/css "swf" - application/x-shockwave-flash "xml" - text/xml "pdf" - application/pdf

2.6 Examples

Below is an example of a project file for WF121 Wi-Fi Module using BGAPI over UART interface:

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<project>
    <hardware>
         <uart channel="0" baud="115200" api="true" handshake="True" />
    </hardware>
    <image out="WF121_BGAPI_UART2_115k.dfu" out_hex="WF121_BGAPI_UART2_115k.hex" />
    <bootloader in="../../fw/boot.juo" />
</project>
```

Below is an example of a project file for WF121 Wi-Fi Module using BGAPI over USB interface:

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<project>
    <hardware>
        <usb descriptor="cdc.xml" api="true" />
    </hardware>
    <image out="WF121_BGAPI_USBCDC.dfu" out_hex="WF121_BGAPI_USBCDC.hex" />
    <bootloader in="../../fw/boot.juo" />
</project>
```

Below is an example of a project file for WF121 Wi-Fi Module using a BGScript application.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<project>
    <hardware in="hardware.xml" />
    <scripting>
        <script in="wifi.bgs" />
    </scripting>
    <image out="WF121_enhanced_BGAPI_UART2_115k.dfu" out_hex="WF121_enhanced_BGAPI_UART2_115k.hex"/
>
    <bootloader in="../../fw/boot.juo" />
</project>
```

3 Hardware configuration file

The hardware configuration file (***hardware.xml***) is used to configure hardware interfaces such as UART, SPI, I2C, Ethernet, ADC and GPIO of you Wi-Fi module.

3.1 <adc>

This tag is used to configure the A/D Converter (ADC) settings:

Attribute	Value - Description
enabled_pins	Bitmask to choose the ADC pins to use. **Values:** **Bit 0**: AN0 **Bit 1**: AN1 **Bit 5**: AN5 **Bit 8**: AN8 **Bit 10**: AN10 **Bit 11**: AN11 **Bit 12**: AN12 **Bit 13**: AN13 **Bit 14**: AN14 **Bit 15**: AN15 **Example to enable all ADCs:** *enabled_pins="0xFD23"*

3.2 <i2c>

This tag is used to configure I2C (Inter-Integrated Circuit) settings.

⚠ Currently WF121 only operates as I2C master.

Attribute	Value - Description
channel	I2C channel to configure **Values:** **1:** I2C channel 1 **3:** I2C channel 3 **5:** I2C channel 5 **Example:** *channel="1"*

Attribute	Value - Description
brg	Baud rate generator setting to use. Divides peripheral clock (40MHz) to generate baud rate for I2C. **Range:** 2-4095 **Default:** 44 (40MHz/44=~909kHz) **Example:** To set 250kHz. (40MHz/250kHz=160) *brg="160"*

3.3 <notify>

This tag is used to configure the GPIO pin used to notify host when BGAPI events should be read over SPI interface. The notify pin indicates there is data in the SPI buffer that should be read by the host.

The pin goes high when WF121 has set BGAPI message to outgoing buffer. The pin goes low when host starts to read message from WF121.

Attribute	Value - Description
port	Port index to use. **Range:** 1-5 **1:** Port B **2:** Port C **3:** Port D **4:** Port E **5:** Port F **Example:** *port="1"*
bit	IO (pin) index to use for host notification **Range:** 0-16 (decimal) **Example:** *bit="0" (PIO0)* *bit="1" (PIO1)*

3.4 <port>

This tag is used to define I/O port configuration settings.

Attribute	Value - Description
index	Port index to configure **Range: 1-6** **1:** Port B **2:** Port C **3:** Port D **4:** Port E **5:** Port F **6:** Port G **Example:** *index="1"*
tristate	Tri-state configuration bit mask. **1:** the corresponding port pin is input. **0:** the corresponding port pin is output. **Default:** 0xFFFF **Example:** *tristate="0x1234"*
open_drain	Open-Drain configuration bit mask **1:** The pin acts as Open-Drain output **Default:** 0x0000 **Example:** *open_drain="0x0000"*
latch	Latched value for port **Default:** 0x0000 **Example:** *latch="0x20"*

3.5 <sleep>

This tag is used to configure sleep mode.

Attribute	Value - Description
delay	Delay in milliseconds for the CPU and Wi-Fi chip to both go from idle to sleep mode. This delay tells when the module is allowed to enter the max power saving state (see *set_max_power_saving_state* from the BGAPI documentation). **Range:** **0 - 4294967295** (in ms) **Example:** *delay="2000"* ⚠ The module cannot go to the max power saving state if there is a timer running in the BGScript or if the USB interface is enabled.
interrupt	Mask to select the wake-up pins connected to external interrupts used to wake-up the module **Values:** **0x1** :INT0 (WF121 pin 38) **0x4** : INT2 (WF121 pin 44) **0x8** : INT3 (WF121 pin 46) **0x10** : INT4 (WF121 pin 37) **Example:** *interrupt="0x1"*

3.6 <spi>

This tag is used to configure SPI settings.

Attribute	Value - Description
channel	SPI channel index to configure **Values:** **3:** SPI channel 3 **4:** SPI channel 4 **Example:** *channel="3"*

Attribute	Value - Description
divisor	SPI divisor to define the clock in master mode. Bit rate is 20MHz / (*divisor + 1)* **Range:** 0-511 **Default:** 15 **Example:** *divisor="15"* *Actual bit rate would be 20MHz/(15+1)=1.25Mbit/s, also confirmed by the log of compiler.*
mode	Use SPI in master mode **Values:** **master:** Use SPI as master **slave:** Use SPI as slave **Default:** master **Example:** *mode="master"*
api	SPI is used for BGAPI protocol (only supported in SPI slave mode) **Values:** **true:** SPI is used for BGAPI protocol **false:** SPI is used for application data **Default:** false **Example:** *api="true"* ⚠ Data transfer reliability in application data mode (value: false) is not guaranteed.
slave_select	Use SPI slave select (SS) pin **Values:** **true:** Use SPI slave select (SS) pin **false:** Do not use SPI slave select (SS) pin **Default:** false **Example:** *slave_select="true"*

Attribute	Value - Description
clock_idle_polarity	SPI clock idle selection **Values:** **low:** Idle state for clock is a low level; active state is a high level. **high:** Idle state for clock is a high level; active state is a low level. **Default:** low **Example:** *clock_idle_polarity="low"*
clock_edge	SPI clock edge selection. **Values:** **0:** Serial output data changes on transition from idle clock state to active clock state. **1:** Serial output data changes on transition from active clock state to idle clock state. **Default:** 0 **Example:** *clock_edge="0"*

3.7 <timer>

This tag is used to configure timer settings, which are described below. For more details see the PIC32 reference manual [1].

This configuration must be set if Output Compare command is used.

Attribute	Value - Description
index	Index of timer to configure **Values:** **2:** Timer 2 **3:** Timer 3 **Example:** *index="2"*
prescale	Prescale value to use **Values:** **1,2,4,8,16,32,64,256** **Default:** 1 **Example:** *prescale="16"*
bits	Width of the timer **Values:** **16:** Timer uses 16 bits **32:** Timer2 uses Timer3 to get 32bits and Timer3 is disabled **Default:** 16 **Example:** *bits="32"*
period	Timer period to use. Timer counts from 0 to period. **Range:** **0x0 - 0xFFFF:** for 16bit timer **0x0 - 0xFFFFFFFF:** for 32bit timer

3.8 <uart>

This tag is used to configure UART settings.

Attribute	Value - Description
channel	UART channel to configure **Values:** **0:** UART channel 1 **1:** UART channel 2 **Example:** *channel="0"* ⚠ Channel 0 refers to USART1 and channel 1 refers to USART2.
baud	UART baud rate Notice that not all baud rates are possible. BGBuild compiler tries to find best match with least amount of error and outputs the result. **Range:** 0-10000000 **Some supported baud rates**: 10000000, 5000000, 2500000, 2000000, 1000000, 115200, 57600 **Default:** 57600 **Example:** *baud="115200"*
stopbits	Stop bits to use **Range:** 1-2 **Default:** 1 **Example:** *stopbits="2"*
parity	Parity to use **Values:** **odd:** use odd parity bit **even:** use even parity bit **none:** no parity bit **Default:** *none* **Example:** *parity:"odd"*

Attribute	Value - Description
handshake	RTS/CTS for data flow control **Values:** **true:** RTS/CTS flow control is used **false:** RTS/CTS flow control is not used* **Default:** false **Example:** *handshake="true"* ⚠ **Streaming mode** When using UART in streaming mode (*api="false"*), it is highly recommend to use RTS /CTS data flow control (*handshake="true"*) to ensure reliable data transfer.
api	UART is used for BGAPI protocol **Values:** **true:** UART is used for BGAPI protocol **false:** UART is used for application data **Default:** *false* **Example:** *api="true"* ⚠ When *true*, there should be an application receiving the BGAPI responses and events at the host, otherwise the module might get stuck.

3.9 <usb>

This tag is used for configuring USB settings.

Attribute	Value - Description
descriptor	The XML file containing the USB descriptors **Example:** *descriptor="cdc.xml"* See the example USB descriptor in the Wi-Fi SDK's **examples** folder.
api	USB is used for BGAPI protocol **Values:** **true:** USB is used for BGAPI protocol **false:** USB is used for application data **Default:** false **Example:** *api="true"*

3.10 <ethernet>

This tag is used for configuring Ethernet settings.

Attribute	Value - Description
enable	Enable/disable Ethernet interface **Values:** **1:** Ethernet interface enabled **0:** Ethernet interface disabled **Default:** 0 **Example:** *enable="1"*

3.11 <sdhc>

This tag is used for configuring SD/SDHC memory card settings.

SD cards with the following specifications are supported:

- SD or microSD card
- FAT32 or FAT16 file system
- Capacity from 2GB to 32GB

Attribute	Value - Description
enable	Enable/disable memory card usage **Values:** **1:** memory card in use **0:** memory card not in use **Default:** 0 **Example:** *enable="1"*
spi_port	SPI channel where memory card is connected **Values:** **3:** SPI channel 3 **4:** SPI channel 4 **Default:** 3 **Example:** *spi_port="3"*

Attribute	Value - Description
cs_port	Memory card chip select I/O port **Values:** **1:** port B **2:** port C **3:** port D **4:** port E **5:** port F **6:** port G **Default:** 3 **Example:** *cs_port="3"*
cs_pin	I/O pin of memory card chip select port **Values:** **0:** pin 0 **1:** pin 1 .... **15:** pin 15 **Default:** 0 **Example:** *cs_pin="6"*

3.12 Example configurations

3.12.1 Using BGAPI over UART interface

The example below shows how to configure BGAPI to be used over UART2 interface with 115200 bps baud rate and hardware flow control.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
    <uart channel="1" baud="115200" api="true" handshake="True" />
</hardware>
```

3.12.2 Using BGAPI over USB/CDC interface

The example below shows how to configure BGAPI to be used over USB/CDC (virtual COM) interface.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
    <usb descriptor="cdc.xml" api="true" />
</hardware>
```

3.12.3 Using BGAPI over SPI slave interface

The example below shows how to configure BGAPI to be used over SPI slave interface. Port D pin 5 is used to indicate when data is available and should be read via the SPI.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
        <spi channel="3" mode="slave" slave_select="true" api="true"/>
        <notify port="3" bit="4"/>
</hardware>
```

3.12.4 Using BGAPI over UART and enabling I2C peripheral interface

The example below shows how to configure BGAPI to be used over UART interface and at the same time I2C interface is enabled for peripheral access.

This configuration is used for example in the thermometer example, where a I2C interface based thermometer is connected directly to WF121 module and controlled by a BGScript application. BGAPI interface is enabled for example for DFU firmware update purposes.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
    <uart channel="1" baud="115200" api="True" handshake="True" />
    <i2c channel="3"/>
</hardware>
```

3.12.5 Using BGAPI over UART2 and enabling UART1 to BGScript application

The example below shows how to configure BGAPI to be used over UART2 interface and UART1 interface access is given to a BGScript application.

This is used in multiple example applications delivered with the Bluegiga Wi-Fi Software. The purpose is for example to use UART1 for BGScript debugging and UART2 for DFU firmware updates.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
    <uart channel="0" baud="115200" api="false" handshake="false" />
    <uart channel="1" baud="115200" api="True" handshake="True" />
</hardware>
```

3.12.6 Using Ethernet interface

The example below shows how to configure BGAPI to be used over UART1 and enabling Ethernet interface.

WF121 Project

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
    <uart channel="1" baud="115200" api="true" handshake="True" />
    <ethernet enable="1"/>
</hardware>
```

4 WF121 Factory Configuration

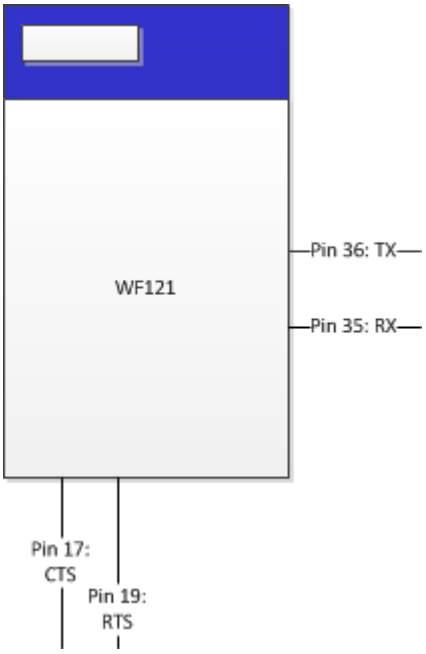
WF121 Modules are factory shipped with settings listed below.

Feature	Value	Notes
BGAPI UART	UART2	This UART gives access to BGAPI protocol, which can be used to control the WF121 Module from a separate host. **Pin 17:** CTS **Pin 19:** RTS **Pin 35:** RX **Pin 36:** TX
Baud rate	115200 bps	
Data bits	8	
Parity bit	none	
Stop bit (s)	1	
RTS /CTS	enabled	RTS and CTS must be connected for the BGAPI communication to work properly.

⚠ UART1 is activated, but not accessed in the factory configuration.

The corresponding hardware configuration is shown below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<hardware>
  <uart channel="0" baud="115200" api="false" />
  <uart channel="1" baud="115200" api="true" handshake="True" />
</hardware>
```



⚠ The example project named "**wf121**" in the SDK's **example** folder matches the default factory configuration.

5 Compiling a Project into a Firmware Image

The project is compiled with the **bgbuild.exe** compiler and this can for example be done either using Bluegiga WiFiGUI software or using the Windows Command Line Prompt (**cmd.exe).** The example below shows how to compile a Wi-Fi Software project with the BGBuild compiler and cmd.exe to a firmware image which can be installed to a Wi-Fi module.

To compile the firmware binary:

- Open for example Windows command prompt
- Navigate (using '**cd**') to the folder where your project is
- Run the ***bgbuild.exe*** compiler as shown below, giving the project file as a parameter
- The syntax for the bgbuild compiler is : **bgbuild.exe <*project_file*>**

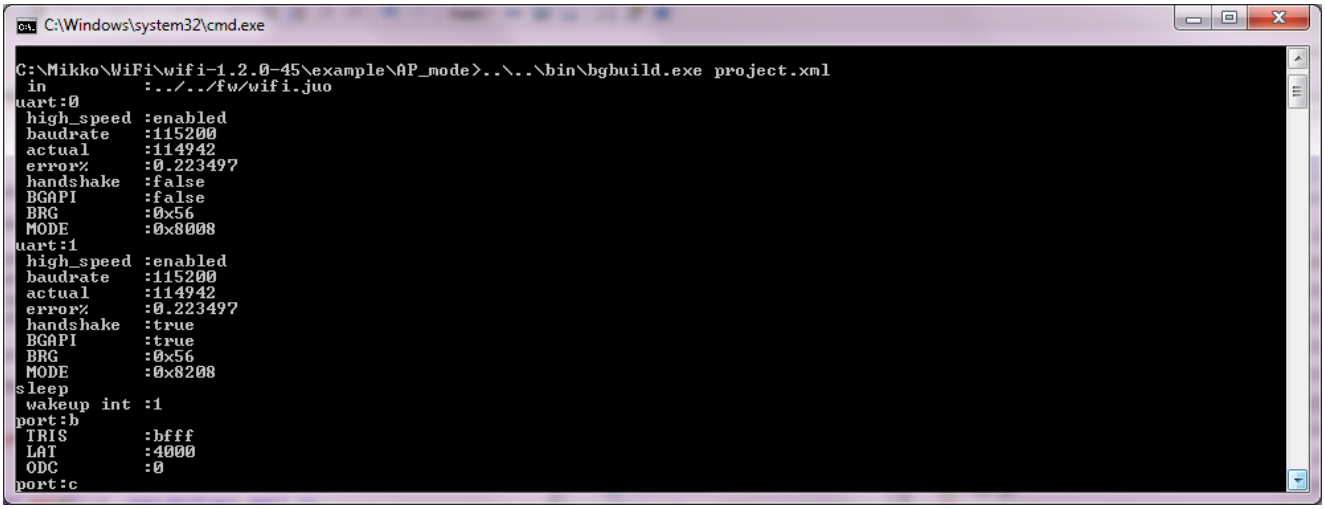


Figure: Compiling the project with BGBuild compiler

Based on the settings in the **project.xml** file the compiler will output .HEX and/or .DFU files to be installed into the Wi-Fi module with the PICkit 3 programmer or alternatively using the DFU update method.

The BGBuild compiler will output the following information.

Feature	Output	Explanation
uart:0	high_speed : enabled baudrate :115200 actual :114942 error% :0.223497 handshake :false BGAPI :false BRG :0x56MODE : 0x8008	This shows UART1 interface is enabled at 115200 bps baud rate. RTS/CTS handshaking is disabled. BGAPI protocol for this UART is disabled. The endpoint is allocated with ID 0 (shown in **uart:0**) and this ID can be used by the BGScript application or BGAPI commands to send data to it or to route for example UART endpoint to TCP endpoint.
uart:1	high_speed : enabled baudrate :5000000 actual :5000000 error% :0 handshake :true BGAPI :true BRG :0x1MODE : 0x8008	This shows UART2 interface is enabled at 5000000 bps baud rate. RTS/CTS handshaking is enabled. BGAPI protocol for this UART is enabled.
sleep	wakeup int :1	This message tells interrupt INT0 is enabled (pin 38).
port:N	TRIS :ffff LAT :0ODC :0	Shows the default configuration for Port N (B to G) and the setting for tri-state configuration bit mask, open drain configuration, and latched value for the port
script	compiler :c:/WiFi /wifi-1.2.0-42/wifi/bin /script_compiler.exe script :APMode.bgs api :../../api/wifiapi. xmlstack :512	Shows the directory where the bgbuild compiler is located.Shows the BGScript source code file.
Stack size	SW : 359044 HW :130 USB :0 Script:163 Free :152661 /512000(30%)	**SW** shows the flash usage (in B) of the firmware image. **HW** shows the size of the hardware configuration. **USB** shows the size of the USB interface descriptor. **Script** shows the size of BGScript code. **Free** shows the total size of the software and how much flash space is left in the device.

6 Installing the Firmware

The firmware can be installed either using the DFU protocol over UART or USB or via the debug interface using the Microchip **PICkit 3 In-Circuit Debugger/Programme**r and **PICkit3** software or **Integrated Programming Environment (IPE)** software from Microchip.

6.1 Using PICkit 3

- As **PICkit 3** will erase the full flash, please write down the MAC (IEEE) address of your device or change MAC address of the firmware to install with changemac.exe
- Download and install **PICkit 3** software from Silicon Laboratories web site (silabs.com).
- Connect the **PICkit 3** to the debug interface of your WF121 (named ICSP on WF121 development kit) and connect the **PICkit 3** to your PC via USB interface.
- Start **PICkit 3** software
    - From **Device Family** select **PIC32**
    - From **Device** drop down list select model : **PIC32MX695F512H**
    - Verify the **STATUS** led on the PICkit 3 device turns **green**
    - From **File** select **Import Hex**
    - Choose the **.hex** file output by the **BGBuild** compiler
    - Press **Write**
- Wait for the programming to be successfully finalized

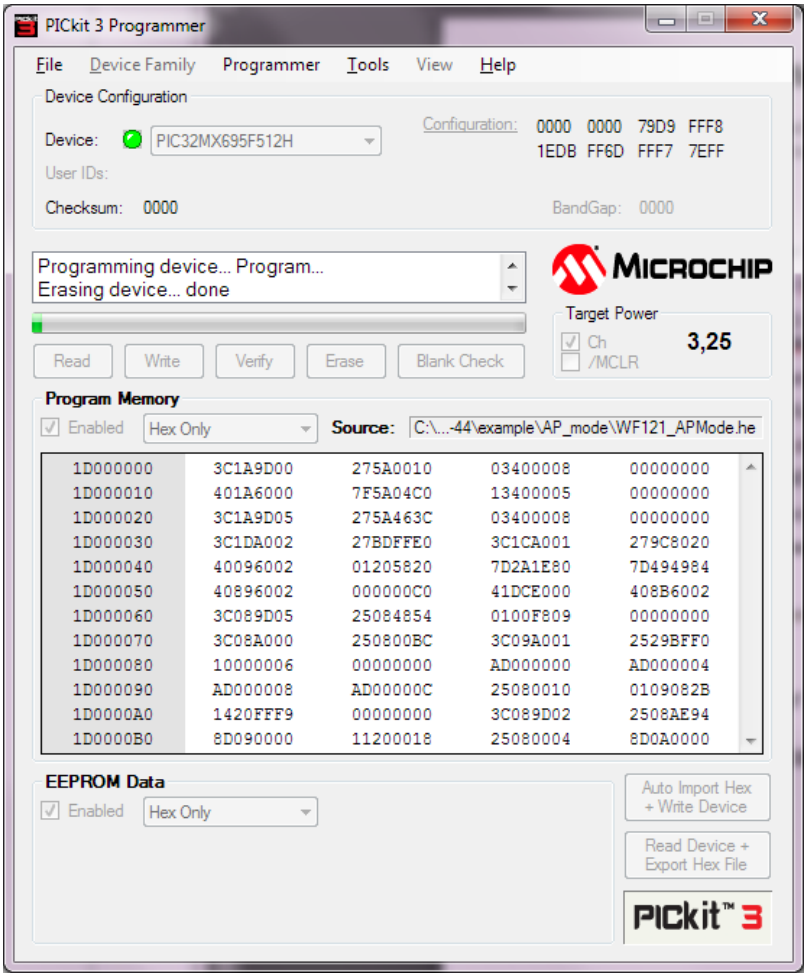


Figure: Programming firmware using PICkit 3

6.2 Using Microchip IPE

Microchip IPE is part of MPLAB-X IDE and can be downloaded from www.microchip.com.

- connect **PICkit 3 In-Circuit Debugger/Programmer** to module debug interface and PC as above
- start **MPLAB IPE**
- select device **PIC32MX695F512H**
- select **Connect** button
- browse hex file, produced with **BGBuild** compiler, as **Source:**
- select **Program** button
- wait for programming finalized and check **Pass** and **Fail** Counts

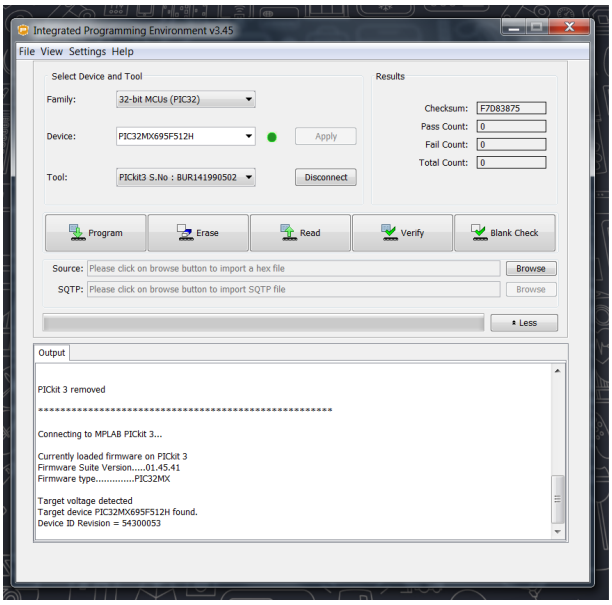


Figure: Programming firmware using MPLAB IPE

> The MAC address can be restored with WIFIGUI software and by typing the original MAC address to the MAC address field not the Network page.
>
> The default MAC address of the firmware can be changed with command line tool changemac.exe, included in SDK. The use of program is *changemac.exe <hex file> <mac address in format xx:xx.xx:xx:xx:xx>*

6.3 Using DFU over UART or USB

In order to install the firmware using DFU protocol, please do the following steps:

- Connect the WF121 Wi-Fi Module to a PC via UART or USB. Selection is done in **project.xml** of the firmware to generate by enabling api on UART
  *<uart channel="1" baud="115200" **api="True"** handshake="True" />*
  or USB
  *<usb descriptor="cdc.xml" **api="True"** />*
- Start **WiFiGUI** software
- Select the correct COM port and baud rate
  - Verify the communication works for example by pressing **Retrieve info** button
- Go to **Firmware** update subpage
  - Press **Boot in DFU** mode button
    - A successful DFU mode is indicated with the event : **wifi_evt_dfu_boot**
  - Select the correct .DFU file using the **Browse**... button
  - Press **Upload**
- Make sure the firmware is uploaded correctly and the device boots normally
  - A successful DFU upload is indicated with event: **wifi_rsp_dfu_flash_upload_finish result: 0**
  - A successful boot is indicated with event: **wifi_evt_system_boot**

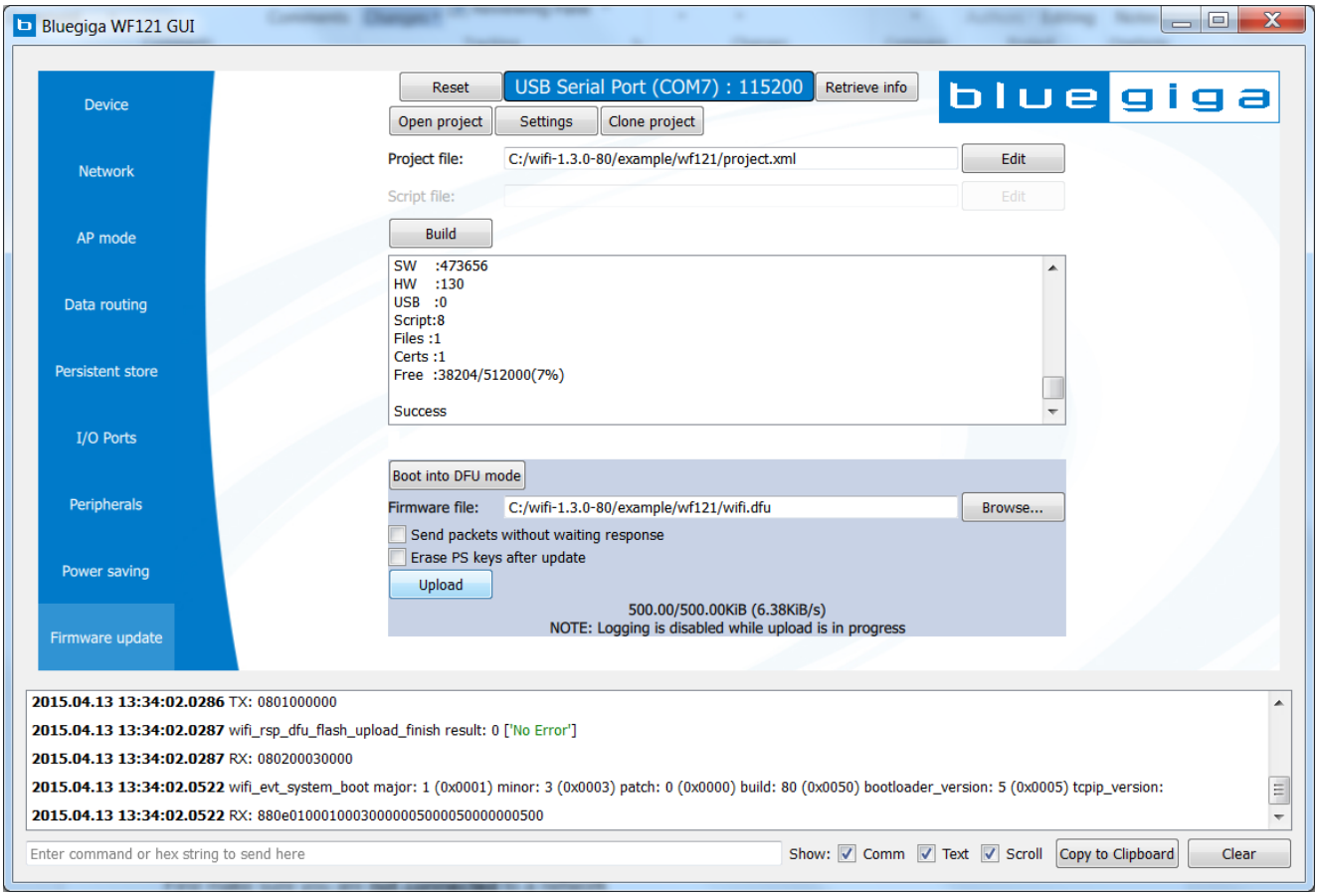
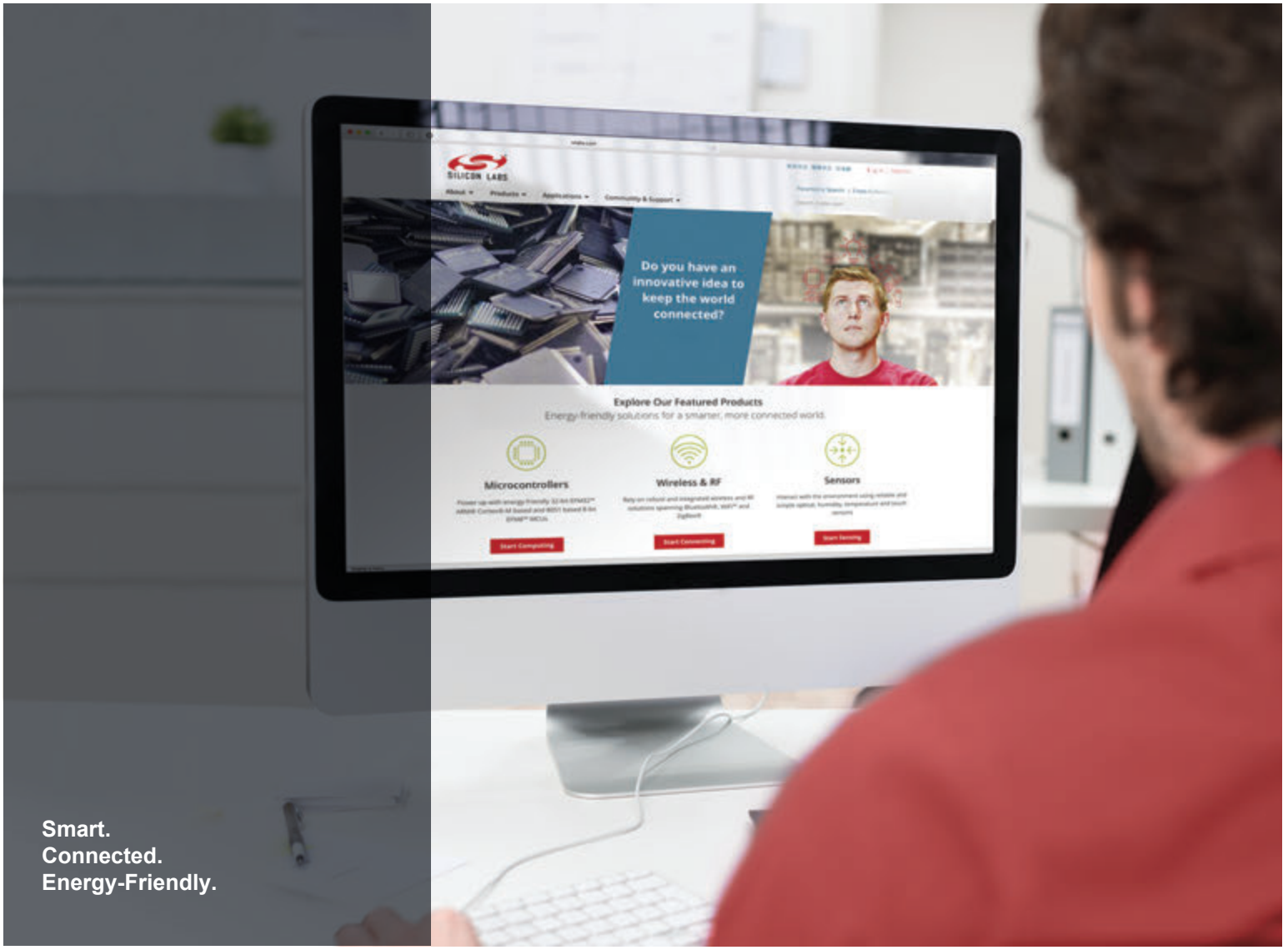


Figure: Updating firmware via DFU

> ⚠ If the DFU load is interrupted, the module hardware configuration can be lost. In this situation UART2 is set as fallback DFU loading port with configuration 115200-8-N-1.

7 References

[1] PIC32 (PIC32MX695F512H) Reference Manual, see Microchip website (www.microchip.com)



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