AN998: WF121 Standalone Wi-Fi Access Point and HTTP Server

APPLICATION NOTE

Tuesday, 28 January 2014

Version 1.1

VERSION HISTORY

Version	Comment
0.5	Draft
0.6	Started to write BGBuild instructions
0.7	HTTP server instructions added
0.8	Improved AP mode configuration
0.9	Firmware update instructions added
0.95	Test instructions added
0.96	Name changed
0.97	BGScript documentation started
0.99	BGScript documentation added
1.0	Updates to BGScript documentation to match the latest code
1.1	BGScript documentation updated to match with the HTTP example of 1.2.2 SDK SW version.

Contents

1 Introduction

This application note describes the configuration and usage of a standalone Wi-Fi Access Point (AP) mode and built-in HTTP server using the Bluegiga WF121 Wi-Fi Module. The Access Point mode can be used to configure the WF121 Wi-Fi module to act as a regular Wi-Fi Access Point and to be connectable from Wi-Fi clients such as smart phones and tablets. The HTTP server can on the other hand be used to display HTML pages and configure the settings of the Access Point, and also reconfigure the Wi-Fi module to a Wi-Fi client mode and connect any standard Wi-Fi network.

The application note describes the following steps:

1. WF121 project configuration
2. WF121 hardware configuration
3. HTTP server configuration and HTML code structure
4. BGScript code that implements the application level functionality

In the Wi-Fi Access Point mode the WF121 Wi-Fi Module acts just like a regular Access Point and broadcasts it's SSID and ESSID so that regular Wi-Fi stations (STAs) can connect to it.

The figure below simply describes how the WF121 Access Point mode works.

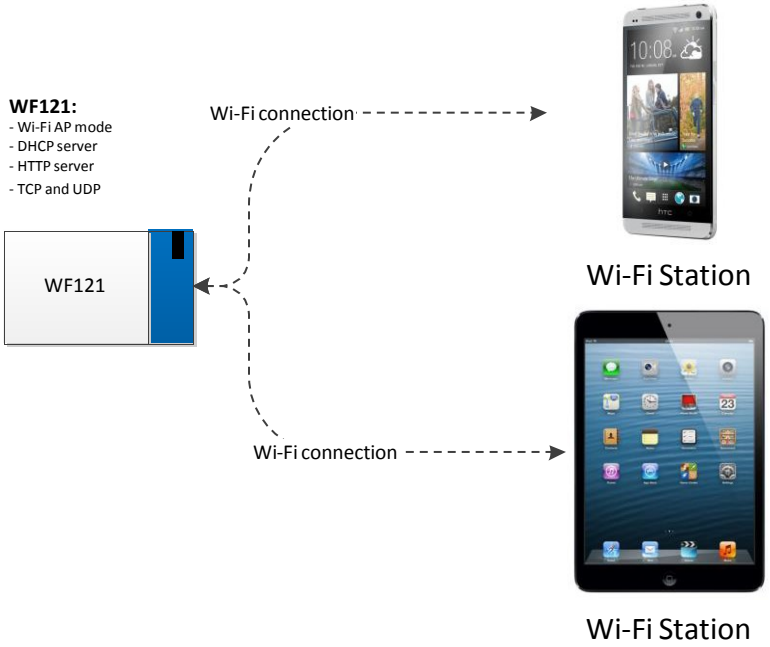


Figure 1: WF121 Wi-Fi Access Point

Once the Wi-Fi connection has been established one can access the built-in embedded HTTP server and browse the built-in HTML pages which can be used to see the current settings, modify them and reconfigure the device into a Wi-Fi client mode.

This application is intended as an example implementation showing the usage of Wi-Fi AP mode and built-in HTTP server. The code follows the WF121 Software Development Kit's HTTP server example.

Feature	Notes
Security	WPA2, WPA and WEP
DHCP	Client and Server
Maximum client connections	5
HTTP server	Supported
Throughput over TCP	3+ Mbps (with a single Wi-Fi connection)

Figure 2: WF121 AP mode features

2 Bluegiga Wi-Fi Software

The Bluegiga Wi-Fi Software is an embedded 802.11 MAC and IPv4 stack targeted for Bluegiga's WF121 Wi-Fi module. The software implements full 802.11 functionality, WPA2, WPA and WEP and WPS security and various IP based protocols such as TCP, UDP, DHCP, DNS, ICMP and HTTP server.

The Bluegiga Wi-Fi software provides powerful, low overhead and easy-to-use Bluegiga BGAPI™ binary API that can be used over UART, USB or SPI interfaces and it provides functions such as Access Point discovery, Access Point associations and encryption and connection establishment. The Wi-Fi Software also supports the 802.11 access point and HTTP server functionality for the easy configurations and direct connections with phones, tablets and PCs. To simplify the software development the Bluegiga Wi-Fi software also includes a Bluegiga BGLib™ C-library that implements the BGAPI protocol parser for various embedded systems.

For standalone applications the Bluegiga Wi-Fi Software also provides a simple BGScript™ scripting language and VM environment, which enables the users to write simple applications for the WF121 Wi-Fi module. This enables lower cost and smaller designs to be made as there is no need to use an external MCU.

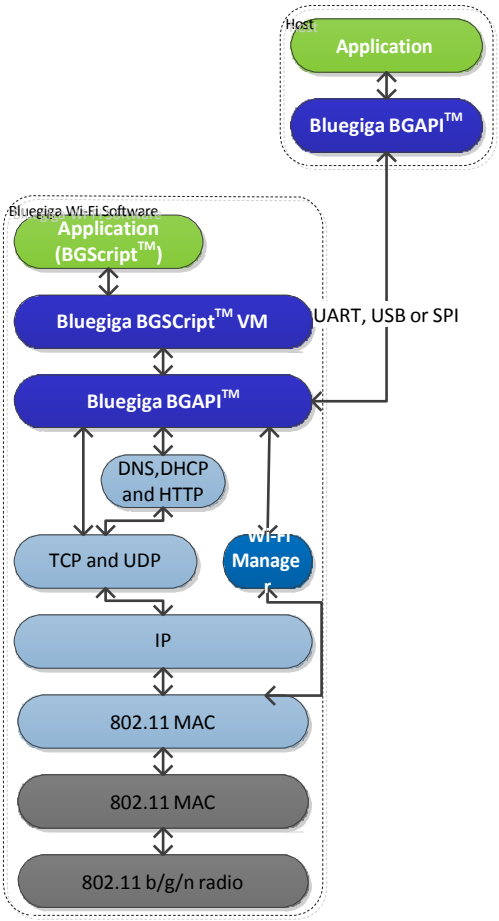


Figure 3: Bluegiga Wi-Fi Software Architecture

Key features

- A fully embedded 802.11 MAC stack
  - STA (client) mode
  - Access Point mode up to five clients
  - WPA2, WPA, WEP and WPS security
  - Up to 3.5 Mbps throughput
- An embedded IPv4 stack
  - TCP and UDP
  - DHCP, DNS client
  - ICMP
  - HTTP server
- Programming API
  - Binary BGAPI™ protocol over UART, USB or SPI
  - Bluegiga BGScript™ scripting language for stand-alone applications
- Small memory requirements
  - 64kB RAM
  - 256kB Flash

Benefits

- A fully embedded 802.11 MAC with the client and access point modes
- Built-in TCP, UDP, DHCP, DNS and HTTP server protocols
- Supports WPA2, WPA, WEP and WPS security
- Simple and low overhead BGAPI™ protocol over UART, USB or SPI
- Standalone applications can be created with Bluegiga BGScript™
- High throughput over TCP and UDP

3 Configuring Wi-Fi AP Mode

This section describes how the WF121 Wi-Fi module is configured to operate in a Wi-Fi Access Point mode and how the built-in HTTP server is enabled.

3.1 Project Configuration: Project.xml

This section describes how to configure the project options for the Wi-Fi Access Point example. The AP mode application implementation is started by first creating a project file (**project.xml**), which defines the resources use by the project and the firmware output file.

- **<hardware>** Defines the XML-file or XML tags containing the WF121 hardware configuration.
- **<script>** Defines the BGScript-file which contains the BGScript application code.
- **<binary>** Defines the WF121 low level firmware image used in the project. With the embedded HTTP server a software profile called **wifi_http_serv** must be used.

  For details of different options see the software profiles chapter in the ***WF121 configuration guide***.
- **<image>** Defines the DFU and HEX output file (firmware file) names for the project
- **<image>** Defines the boot loader file for the project. The default boot loader enables DFU firmware updates over UART or USB
- **<files>** Defines the HTML, CSS and GIF files used by the HTTP server

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<project>
    <scripting>
        <script in="main.bgs" />
    </scripting>
    <software>
        <binary in="fw/wifi_http_serv.juo" />
    </software>

    <hardware>
        <uart channel="0" baud="5000000" api="false" handshake="True" />
        <uart channel="1" baud="5000000" api="true" handshake="True" />
        <sleep interrupts="0x1"/>
    </hardware>
    <files>
        <file path="404.html"/>
        <file path="main.html"/>
        <file path="ap.html"/>
        <file path="appl.html"/>
        <file path="client.html"/>
        <file path="index.html"/>
        <file path="logo.gif"/>
        <file path="style.css"/>
    </files>
</project>
```

Figure 4: Project configuration file

3.2 Hardware Configuration

The hardware configuration for WF121 Wi-Fi Module device may be part of the project file or it may be stored in a separate file. It describes which interfaces and functions are in used and their properties.

- **<uart>** UART1 and UART2 interfaces are enabled at the 5 Mbps baud rate and with hardware flow control enabled. BGAPI is enabled for the UART2 interface, so the WF121 can be controlled with BGAPI commands over UART2 and this also enables DFU updates over the UART.
- **<sleep>** Enables interrupt listener for INT0 (pin 38).

3.3 HTTP Server Configuration

This section describes how to configure the necessary HTML pages, CSS style sheet and picture files for the built-in HTTP server.

File	Description	Mandatory
Index.html	Basic layout page and frame structure	**Yes**
client.html	Wi-Fi client mode configuration page	No
ap.html	Wi-Fi Access Point mode configuration page	No
main.html	Landing page	**Yes**
appl.html	Application data configuration page	No
404.html	HTTP 404 error page	No
style.css	CSS Style sheet for HTML pages	**Yes**
logo.gif	Image files used in header	No

Figure 5: HTTP server files

Note:

At the moment the HTTP server can only host the above HTML pages out of which the ones mentioned mandatory must always be present and in addition there are a few optional pages.

3.3.1 index.html

The **index.html** is the basic layout page that defines structure for the web server. The default index.html contains a header (named **header**) frame on the top of the page, a simple navigation frame on the left side (named **sidebar**) and a main content frame in the centre (named **middle**).

The CSS style sheet is applied to index.html as shown below.

```
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Bluegiga WF121 Wi-Fi Module</title>
    <link rel="stylesheet" href="style.css" type="text/css" media="screen, projection" />
</head>
```

Figure 6: Applying CSS style sheet

The header frame and logo.gif is defined as shown below:

```
<div id="header">
    <img src="img/logo.gif" class=img>
</div>
```

Figure 7: Defining top header and inserting logo.gif into it

The navigation frame on the left on the other hand is constructed as show below.

```
<div class="sidebar" id="sideLeft">
    <form>
        <a href='network.cgi?Main' target='fr' class='btn'>Main page</a><br>
        <a href='network.cgi?Ap' target='fr' class='btn'>Wi-Fi AP Mode</a><br>
        <a href='network.cgi?Client' target='fr' class='btn'>Wi-Fi Client Mode</a><br>
        <a href='network.cgi?Appl' target='fr' class='btn'>Application Config</a><br>
    </form>
</div>
```

Figure 8: Navigation frame and buttons

The navigation frame contains four buttons (**Main page**, **Wi-Fi AP Mode**, **Wi-Fi Client mode** and **Application Config)** which will load the specific configuration pages for the Wi-Fi Access Point, Client and Application data.

The ***network.cgi*** is a fixed CGI script that will cause a HTML page load in the main content frame and the HTML page name is passed as a parameter to the CGI script such as **Client** which will cause the **client.html** page to be loaded.

3.3.2 main.html

This HTML page will show information about the current state of the device or other desired data from PS keys. In this example, the following information will be dynamically printed on the HTML page.

- **Services**: AP Mode, Client mode or Both
- **SSID**: SSID used in the Access Point mode
- **Channel**: Wi-Fi channel used by the device
- **Security**: The Wi-Fi security mode used

The information is presented using the HTML label type as shown is the HTML source in Figure 9.

```
<table>
    <tr><td width="50%"><b>Services:</b></label></td><td><label id='5'></label></td></tr>
    <tr><td><label id='52'><b>SSID:</b></label></td><td><label id='20'></label></td></tr>
    <tr><td><label id='53'><b>Channel:</b></label></td><td><label id='21'></label></td></tr>
    <tr><td><label id='54'><b>Security:</b></label></td><td><label id='24'></label></td></tr>
</table>
```

Figure 9: main.html HTML source code

The syntax of the HTML page is described below:

- **<label id='52'>** This will cause the label text to be loaded from PS-key 52. If the PS-key does not contain any data the default text (f.ex. "**SSID:**") will be shown instead.
- **<label id='20'>** This will cause the label text to be loaded from PS-key 20.

The HTML code shown above will load the device information from the dedicated PS keys and populate the information on the web page. The following PS keys are used in the **main.html** page as the data inputs:

PS key ID	PS key type	Type on web page	Description
5	TBD	Text	PS key indicating the service running on the device **Values:** Access Point Client Both
20	32 byte array	Text	Access Point mode SSID
21	4 byte integer	Number	Access point channel
24	4 byte integer	Text	Access point security mode

Table 1: PS keys and used on the main page

Note:

Additional data can also be displayed on the main page by using PS keys 50 to 59, which are reserved for labels.

3.3.3 client.html

The **client.html** page can be used to configure the device into a Wi-Fi client mode and connect your Wi-Fi module to an existing infrastructure.

When the Wi-Fi module is booted it will perform an access point scan and scan for available Wi-Fi networks in its range. It will then store ten of the strongest (based on RSSI) networks into the PS keys and the **client.html** page will use these PS keys to show the available networks.

To configure the device into a client mode, the user can select one of the available networks, type in the password and press connect button to activate the configuration. If the network uses hidden SSID or for some reason the SSID is not displayed, it also can be manually typed.

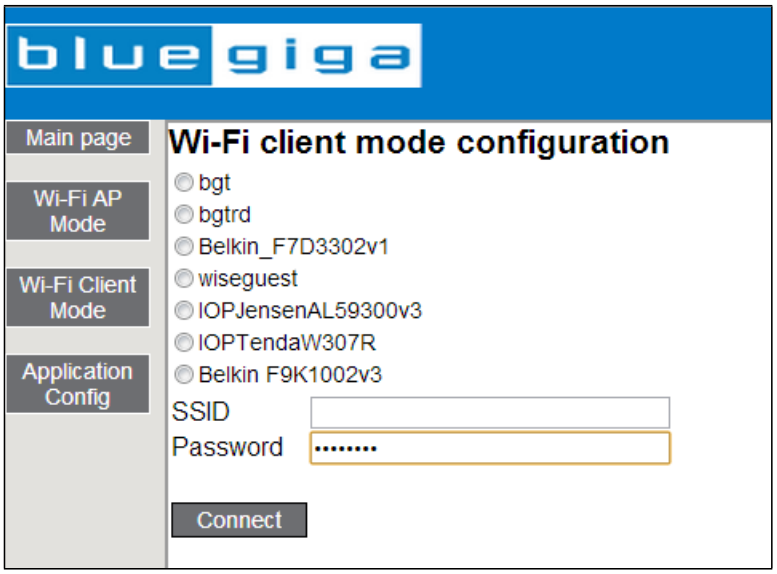


Figure 10: Wi-Fi client mode configuration

Note:

The HTML source code of the **client.html** page should not be edited. As mentioned, this page is optional and if the end-device is not expected to support the Wi-Fi client mode, this page can be left out.

3.3.4 ap.html

The **ap.html** page shows how the Access Point mode settings can be configured. The page allows the following settings to be modified:

- Access Point **SSID** — Configures the Access Point SSID (name)
- Access Point **Channel** — Configures the Access Point channel
- Access Point **Security mode** — Configures the Access Point security mode
- Access Point **Security key** — Sets the Access Point security key
- **Wi-Fi N-mode** (on/off) — Enables or disables Wi-Fi N-mode
- **DHCP server** (on/off) — Enables or disables DHCP server
- **1st client address** — Configures the starting IP address for DHCP clients
- **DNS Server** (on/off) — Enables or disables the DNS server
- **DNS URL** — The DNS URL of the device.
- **Any URL** (on/off) — If this is enabled all DNS requests will be forwarded to the URL defined in "**DNS URL**" key

The settings are stored into PS-keys, which are described in the table below:

PS key ID	PS key type	Type on web page	Values	Description
20	32 byte array	Text	User defined	Access Point mode SSID
21	4 byte integer	Number	**1-11**	Access Point channel
24	4 byte integer	Text	**0:** Open **1:** WPA **2:** WPA2 **3:** WEP	Access Point security mode
22	Max 64 byte array	password	User defined	Access Point security key
23	4 byte integer	checkbox	**1:** Enabled **0:** Disabled	Access Point 802.11 N-mode enabled/disabled
30	4 byte integer	checkbox	**1:** Enabled **0:** Disabled	DNS server enabled/disabled
31	4 byte integer	text	User defined	First address of DHSP server IPv4 address space in a dotted string mode
35	4 byte integer	checkbox	**1:** Enabled **0:** Disabled	DNS server enabled/disabled

36	Max 64 byte array	text	User defined	URL of DNS server
37	4 byte integer	checkbox	**1:** Enabled **0:** Disabled	DNS reply to any URL enabled/disabled

Table 2: PS-keys used to configure the Access Point mode

The HTML code to generate the page, load and save the values looks like the example below. The HTML code uses a form and **save.cgi** script to load and store the values into PS-keys.

Two buttons "**Save**" and "**Activate**" are also which can be used to Save the configuration or to save and activate the configuration, which will reset the Wi-Fi module.

```
  <form action="/save.cgi" method="get">
      <table>
    <tr><td width="110px">SSID</td><td> <input type="text" name="20" size="32"  id="20"></td></tr>
    <tr><td>Channel</td><td><input type="text" name="21" size="2" id="21"></td></tr>
    <tr><td>Security mode</td><td>
        <select name="24" id="24">
         <option value="0">Open</option>
         <option value="1">WPA</option>
         <option value="2">WPA2</option>
         <option value="3">WEP</option>
       </select>
       </td></tr>
    <tr><td>Security Key</td><td witdh="10px"><input type="password" name="22" size="20" id="22"></td></tr>
    </table>
    <table>
   <tr><td width="110px">Wi-Fi N-mode</td><td width="35px" align="left" ><input type="checkbox" name="23" id="23" ></td></tr
   <tr><td>DHCP Server</td><td><input type="checkbox" name="30" id="30"><input type="hidden" name="30"></td><td>1st client ac
    <tr><td>DNS Server</td><td><input type="checkbox" name="35" id="35"><input type="hidden" name="35"></td><td>DNS URL</td>
    </table>
    <INPUT TYPE="submit"  NAME="Save" VALUE="Save" CLASS="btn_body">  <INPUT TYPE="submit"  NAME="Activate" VALUE="Activate"
      </form>
    </body>
</html>
```

Figure 11: HTML code for AP mode configuration

Note:

The code of **ap.html** should not be modified other than removing the undesired configuration options.

3.3.5 appl.html

The **appl.html** page shows how the HTTP server can be used to display application data and interact with the BGScript application. The page can be used to read, display and store data from any user specific PS keys.

Buttons can also be used to interact with the BGScript application.

The following PS-keys are available for the user specific data:

PS key ID	PS key type	Type on web page	Values	Description
10 - 13	4 byte integer	number	User defined	Integer type appliction parameters
14 - 17	Max 64 byte array	text	User defined	String type application parameters
50 - 59	Max 64 byte array	text	User defined	String type application parameters for HTML page labels

Figure 12: PS keys for user specific data

The example below shows how to use HTML input field to display and read the user specific PS keys and how to implement buttons.

```
<body>
    <h2><label id='18'>Application Page</label></h2><br>
  <form action="/app.cgi" method="get">
   <table>
   <tr><td><label for="attr1">Integer 1</label></td><td><input type="text" name="10" size="11"  id="10"></td></tr>
   <tr><td><label for="attr2">Integer 2</label></td><td><input type="text" name="11" size="11"  id="11"></td></tr>
   <tr><td><label for="attr3">Integer 3</label></td><td><input type="text" name="12" size="11"  id="12"></td></tr>
   <tr><td><label for="attr4">Integer 4</label></td><td><input type="text" name="13" size="11"  id="13"></td></tr>
      <tr><td><label for="attr5">String 1</label></td><td><input type="text" name="14" size="64"  id="14"></td></tr>
      <tr><td><label for="attr5">String 2</label></td><td><input type="text" name="15" size="64"  id="15"></td></tr>
      <tr><td><label for="attr5">String 3</label></td><td><input type="text" name="16" size="64"  id="16"></td></tr>
      <tr><td><label for="attr5">String 5</label></td><td><input type="text" name="17" size="64"  id="17"></td></tr>
   <tr><td><input type="submit"  name="Action" value="Save" class="btn_body"></td></tr>
   </table>
   </form>
```

Figure 13: appl.html source code

A HTML form needs to be used to collect and display user data and the CGI script used for this purpose is called **app.cgi**. The example above displays the integer data from PS keys 10 to 13 and string data from PS keys 14 to 17.

The example below on the other hand uses a second HTML form and **button.cgi** script to demonstrate how to listen for button presses and pass them to BGScript at which **https_button** event is to be listened.

```
    <form action="/button.cgi" method="get">
    <Input type="submit"  name="1" value="Button1" class="btn_body">
    <Input type="submit"  name="2" value="Button2" class="btn_body">
    <Input type="submit"  name="3" value="Button3" class="btn_body">
    </form>
   </body>
```

Figure 14: Example how to use buttons

3.3.6 404.html

This is the regular **404** or **Not Found** error page indicating the client was able to communicate with the HTTP server, but the requested page was not found. E.g., this is shown when an URL cannot be found from the in-built webserver.

3.3.7 style.css

Cascading Style Sheets (CSS) file that is used for describing the look and feel of the HTML pages.

3.3.8 logo.gif

A single GIF image can be used with the web server and it can be up to 2kB in size.

3.4 BGScript Code

This section of the document goes through the most important parts of the BGScript code that is used to implement the standalone Access Point mode and explains how the code works. However not all the code parts are discussed here and more detailed information can be found from the source code documentation. In order to improve readability and demonstrate how a project can utilize multiple source code files, the example has been split into three different files: **sta.bgs** containing Wi-Fi client functionality, **ap.bgs** for Wi-Fi Access Point functionality and **main.bgs** containing the common code and shared variables.

3.4.1 System Boot Event

All BGScript applications more or less start by catching the **system_boot** event. This event is generated when the device is power on and the firmware starts.

The boot event code is shown in Figure 15 and it will perform the following actions:

- Call the initialization procedures defined in Wi-Fi Client and Wi-Fi Access Point files. See Figure 16 and Figure 17 for details.
- Initialize the default operating mode into Wi-Fi Access Point.
- Then read the operating mode from the PS keys in case it has been changed by the user earlier and set the **operating_mode** parameter based on the flash contents.
- Next the BGScript will call an API function that sets the operating mode to Wi-Fi Access Point or Wi-Fi client depending on the PS key configuration.
- Next the BGScript code will call the API function to read the **config_get_mac()** to read the MAC address and append it to the default SSID name
- Finally the **sme_wifi_on()** function is called that will turn on the 802.11 radio

```
# Event received when the system has been successfully started up.
event system_boot(major, minor, patch, build, bootloader_version, tcpip_version, hw)
  # Initialise variables
  call init_sta_mode()
  call init_ap_mode()
  operating_mode = MODE_AP
  ps_label_scan_result = FLASH_PS_KEY_SCAN_RESULT_START

  # Read the current operating mode from PS.
  call flash_ps_load(FLASH_PS_KEY_MODULE_SERVICE)(result, value_len, value_data(0:value_len))
  if result = 0
    operating_mode = value_data(0:value_len)
  end if

  # Set the correct operating mode.
  call sme_set_operating_mode(operating_mode)

  # Read the MAC address, this call will trigger config_mac_address.
  call config_get_mac(0)(result, value_len)
  if result != 0
    # No MAC address was found. This happens after flashing with PICKit. Proceed with startup,
    # a default MAC address will be used.

    # Enable Wi-Fi, this call will trigger sme_wifi_is_on() event.
    call sme_wifi_on()
  end if
end
```

Figure 15: system_boot event

The initialization procedure for Wi-Fi Client mode simply initializes the variable used for client mode reconnection logic.

```
# Procedure for initialising STA mode variables.
export procedure init_sta_mode()
  # Initialise variables
  reconnect_count = 0
end
```

Figure 16: init_sta_mode procedure

Wi-Fi Access Point initialization procedure performs the following actions:

- Set the default Wi-Fi Access Point configuration to SSID **"Bluegiga"** and open security.
- Initialize variables that later on enable HTTP server, enable DHCP server and disable DNS server

```
# Procedure for initialising AP mode variables.
export procedure init_ap_mode()
  # Default operating mode is AP. If no AP parameters are stored in PS,
  # channel 1, no security and SSID "Bluegiga" are used.
  default_ap_channel = 1
  default_ap_security = 0
  default_ap_ssid_len = 8
  default_ap_ssid(0:default_ap_ssid_len) = "Bluegiga"

  # Default DNS name is "bluegiga.com" unless overwritten by PS.
  default_ap_dns_len = 12
  default_ap_dns(0:default_ap_dns_len) = "bluegiga.com"

  # HTTP server and DHCP server are enabled, DNS server is disabled
  # unless overwritten by PS.
  enable_ap_https = 1
  enable_ap_dhcps = 1
  enable_ap_dnss = 0
end
```

Figure 17: init_ap_mode procedure

After the Wi-Fi has been turned on a API event **sme_wifi_is_on** is generated. It is caught by the BGScript code.

In the Access Point mode the BGScript code simply starts a status notification by calling a BGScript function **start_status_notification,** which in this state blinks a led every 1000ms until the AP mode is ready. On WF121 development kit the status is indicated via RD6 pin and the led connected to it.

```
# Event received after Wi-Fi has been switched on.
event sme_wifi_is_on(state)
  if operating_mode = MODE_STA then
    # Indicate connecting state via the status notification service.
    call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_STA, STATUS_INTERVAL_CONNECTING)

    # Start station mode.
    call start_sta_mode()
  else
    # Indicate connecting state via the status notification service.
    call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_AP, STATUS_INTERVAL_CONNECTING)

    # Initiate a scan. This needs to be done before starting AP mode.
    # This call will trigger sme_scanned() event once done.
    call sme_start_scan(HANDLE_WIFI, 0, 0)
  end if
end
```

Figure 18: sme_wifi_is_on event handled

Once the AP mode has been started another API event is generated and it's also caught by the BGScript code.

The code below starts the HTTP server and changes the status notification mode and turns the RD6 (or the led) connected to it permanently on.

```
# Event received after AP mode has been started.
event sme_ap_mode_started(hw_interface)
  # Read AP DNS name from PS.
  call flash_ps_load(FLASH_PS_KEY_DNSS_URL)(result, value_len, value_data(0:value_len))
  if result != 0
    # PS has no valid parameter, use the default value.
    call flash_ps_save(FLASH_PS_KEY_DNSS_URL, default_ap_dns_len, default_ap_dns(0:default_ap_dns_len))
  end if

  # Enable/disable HTTP/DHCP/DNS server.
  call https_enable(enable_ap_https, enable_ap_dhcps, enable_ap_dnss)

  # Indicate connected state via the status notification service.
  call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_AP, STATUS_INTERVAL_CONNECTED)
end
```

Figure 19: Catching event indicating AP mode has been started

In case the AP mode is not properly started an error event is generated and the status notification mode changed.

```
# Event received after AP mode startup fails.
event sme_ap_mode_failed(reason, hw_interface)
  # Indicate error state via the status notification service.
  call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_AP, STATUS_INTERVAL_ERROR)
end
```

Figure 20: Catching event indicating AP mode has been started

3.4.2 Starting Wi-Fi Client Mode

The following BGScript function will start the device in the Wi-Fi client (STA) mode and it will do the following actions:

- By default the example application enables DHCP client and it assume the network it connects to has a DHCP server.
- Also in client mode HTTP, DHCP and DNS servers are disabled. If you modify the code pay extra attention to this as having two DHCP servers in the same network may result it a complete network failure.
- Next the BGScript code will read the Wi-Fi security key from the PS keys as well the SSID of the Access Point that it's supposed to connect to.
    - If no SSID was found from the PS keys the **start_status_notification** BGScript function is used to indicate an error.
- However if a valid SSID was found a connection attempt to the AP is made using the API call **sme_connect_ssid.**

```
# Procedure for starting STA mode.
export procedure start_sta_mode()
  # Enable DHCP.
  call tcpip_configure($00000000, $00000000, $00000000, 1)

  # Disable HTTP, DHCP and DNS servers
  call https_enable(0, 0, 0)

  # Read STA passphrase from PS and set it. Ignored if not found from PS.
  call flash_ps_load(FLASH_PS_KEY_CLIENT_PW)(result, value_len, value_data(0:value_len))
  if result = 0
    call sme_set_password(value_len, value_data(0:value_len))
  end if

  # Read STA SSID from PS and initiate a connection attempt.
  call flash_ps_load(FLASH_PS_KEY_CLIENT_SSID)(result, value_len, value_data(0:value_len))
  if result = 0
    ssid_len = value_len
    memcpy(ssid(0), value_data(0), ssid_len)

    # This call will trigger either sme_connected() event if the attempt
    # succeeds or sme_connect_failed() if it fails.
    call sme_connect_ssid(ssid_len, ssid(0:ssid_len))
  else
    # We don't have a valid SSID defined in PS. Indicate error state via the status notification service.
    call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_STA, STATUS_INTERVAL_ERROR)
  end if
end
```

Figure 21: Starting Wi-Fi client (STA) mode

3.4.2.1 Reconnection Management

In client mode the BGScript code implements a simple reconnection logic that attempts to reconnect the Access Point in case a single connection attempt fails.

- A failed Wi-Fi connection attempt will generate an API event **sme_connect_failed** which is caught by the BGScript code
- The event handled then increases the reconnection counter until it reaches the maximum value
- A single shot software timer is then created in order to make another connection attempt
- Once the maximum number of reconnection attempts is reached and if the connection has not been successful the BGScript code indicates an error with the **start_status_notification** function

```
# Event received after a connection attempt fails.
event sme_connect_failed(reason, hw_interface)
  reconnect_count = reconnect_count + 1
  if(reconnect_count < MAX_RECONNECTS)
    # Set a timer for a reconnection attempt.
    call hardware_set_soft_timer(RECONNECTION_DELAY, TIMER_RECONNECTION, 1)
  else
    # Indicate error state via the status notification service.
    call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_STA, STATUS_INTERVAL_ERROR)
  end if
end
```

Figure 22: Reconnection management

When the software timer expires a BGScript event is generated, which can be caught with the script.

When the timer event in generated and if the timer handle is a reconnection handle the script simply issues an API call **sme_connect_ssid()** which tries to open a Wi-Fi connection to an Access Point.

```
# Event received when a timer is triggered.
event hardware_soft_timer(handle)
  if handle = TIMER_RECONNECTION
    # Try reconnecting to the network.
    call sme_connect_ssid(ssid_len, ssid(0:ssid_len))
  end if
  if handle = TIMER_STATUS_PIN
    # Handle status notification.
    call status_notification_next()
  end if
end
```

Figure 23: Reconnection implemented with a software timer

3.4.3 Starting Wi-Fi Access Point Mode

The next BGScript function will start the device in the Wi-Fi Access Point (AP) by performing the following actions:

- As the very first step the BGScript code will set a static IP address for the device, in this case: **192.168.1.1**
- Then the BGScript code will load the AP mode configuration from the PS-keys including:
  - DHCP server enable bit
  - DNS server enable bit
- Power saving is then disabled as the AP mode currently does not support power saving
- Next the other configuration values are loaded from PS keys, including:
  - Access Point security key
  - Access Point channel
  - Access Point security mode
  - Access Point SSID
- Finally the AP more is started with the API call **sme_start_ap_mode**

Note:

If the values are not from PS-keys, default values are used instead, which are initialized in the **system_boot** event handler.

```
# Procedure for starting AP mode.
export procedure start_ap_mode()
  # Set a static IP address (192.168.1.1).
  call tcpip_configure(192.168.1.1, 255.255.255.0, 192.168.1.1, 0)

  # Check whether DHCP server and DNS server should be enabled.
  call flash_ps_load(FLASH_PS_KEY_DHCPS_ENABLE)(result, value_len, value_data(0:value_len))
  if result = 0
    enable_ap_dhcps = value_data(0:value_len)
  end if
  call flash_ps_load(FLASH_PS_KEY_DNSS_ENABLE)(result, value_len, value_data(0:value_len))
  if result = 0
    enable_ap_dnss = value_data(0:value_len)
  end if

  # Disable all power save functionality.
  call system_set_max_power_saving_state(0)

  # Read AP channel from PS.
  call flash_ps_load(FLASH_PS_KEY_AP_CHANNEL)(result, value_len, value_data(0:value_len))
  if result = 0
    ap_channel = value_data(0:value_len)
  else
    # PS has no valid parameter, use the default value.
    ap_channel = default_ap_channel
  end if

  # Read AP security mode from PS.
  call flash_ps_load(FLASH_PS_KEY_AP_SECURITY)(result, value_len, value_data(0:value_len))
  if result = 0
    ap_security = value_data(0:value_len)
  else
    # PS has no valid parameter, use the default value.
    ap_security = default_ap_security
  end if

  # Read AP SSID from PS.
  call flash_ps_load(FLASH_PS_KEY_AP_SSID)(result, value_len, value_data(0:value_len))
  if result = 0
    ssid_len = value_len
    memcpy(ssid(0), value_data(0), ssid_len)
  else
    # PS has no valid parameter, use the default value.
    ssid_len = default_ap_ssid_len
    memcpy(ssid(0), default_ap_ssid(0), ssid_len)
  end if

  # Read AP passphrase from PS.
  call flash_ps_load(FLASH_PS_KEY_AP_PW)(result, value_len, value_data(0:value_len))
  if result = 0
    # Set the passphrase.
    call sme_set_ap_password(value_len, value_data(0:value_len))

    # Start the AP mode. This call will trigger sme_ap_mode_started() event on success
    # and sme_ap_mode_failed on failure.
    call sme_start_ap_mode(ap_channel, ap_security, ssid_len, ssid(0:ssid_len))
  else
    # No passphrase in PS. If not using security, proceed with connection. Otherwise
    # indicate error state via the status notification service.
    if ap_security = 0
      # Start the AP mode. This call will trigger sme_ap_mode_started() event on success
      # and sme_ap_mode_failed on failure.
      call sme_start_ap_mode(ap_channel, ap_security, ssid_len, ssid(0:ssid_len))
    else
      call start_status_notification(STATUS_PIN_INDEX, STATUS_PIN_AP, STATUS_INTERVAL_ERROR)
    end if
  end if
end
```

Figure 24: BGScript code to start Wi-Fi AP mode

3.4.4 PS Key Management

The BGScript code contains a few functions for generic PS key management, which are shortly explained below.

The following BGScript function saves the Access Point operating mode to the PS store

```
# Procedure for falling back to AP mode. This will reset the operating
# mode setting into PS store.
procedure reset_operating_mode()
    call flash_ps_save(FLASH_PS_KEY_MODULE_SERVICE, 4, MODE_AP)
end
```

Figure 25: Resetting operating mode

The following BGScript function stores the scan results (discovered Access Points) to the PS store.

```
# Procedure for storing a scan list entry into PS store.
procedure store_scan_result(ssid_len, ssid_data())
    if ps_label_scan_result <= FLASH_PS_KEY_SCAN_RESULT_MAX
        call flash_ps_save(ps_label_scan_result, ssid_len, ssid_data(0:ssid_len))
        ps_label_scan_result = ps_label_scan_result + 1
    end if
end
```

Figure 26: Storing discovered Access Points

The following function on the other hand on the other hand deletes the unused scan results from the PS store.

```
# Procedure for deleting unused scan list entries from PS store.
procedure delete_unused_scan_entries()
    while ps_label_scan_result <= FLASH_PS_KEY_SCAN_RESULT_MAX
        call flash_ps_erase(ps_label_scan_result)
        ps_label_scan_result = ps_label_scan_result + 1
    end while
end
```

Figure 27: Deleting unused Access Points from PS store

3.4.5 Restoring Default Configuration

The demo application implements functionality that an interrupt (for example a button press on DKWF121) can be used to restore the Wi-Fi module to the default configuration. In the BGScript code this is implemented with a simple code handling the hardware interrupts. If an interrupt is cached the default operating mode is restored with the following BGScript code.

The code will perform the following actions:

- Removes all Wi-Fi Access Point settings from PS store so that the default values will be used
- Calls BGScript function **reset_operating_mode**
- And finally performs a system reset with API call **system_reset**

```
#In case of button connected to INT0 is pressed, set default settings and reset module
event hardware_external_interrupt(irq,timestamp)
    if irq = 0 then
        call reset_ap_mode()
        call reset_operating_mode()
        call system_reset(0)
    end if
end
```

Figure 28: Restoring default operating mode

```
# Procecure for restoring AP mode to factory settings.
export procedure reset_ap_mode()
  call flash_ps_erase(FLASH_PS_KEY_AP_SSID)
  call flash_ps_erase(FLASH_PS_KEY_AP_CHANNEL)
  call flash_ps_erase(FLASH_PS_KEY_AP_SECURITY)
  call flash_ps_erase(FLASH_PS_KEY_AP_PW)
  call flash_ps_erase(FLASH_PS_KEY_DHCPS_ENABLE)
  call flash_ps_erase(FLASH_PS_KEY_DHCPS_SPACE)
  call flash_ps_erase(FLASH_PS_KEY_DNSS_ENABLE)
  call flash_ps_erase(FLASH_PS_KEY_DNSS_URL)
  call flash_ps_erase(FLASH_PS_KEY_DNSS_ANY_URL)
end
```

Figure 29: Removing AP mode settings

4 Compiling the Firmware

The firmware is compiled with the **bgbuild.exe** and this can for example be done from the Windows Command Line Prompt (**cmd.exe).** The example below shows how to compile the HTTP server project with the BGBuild compiler.

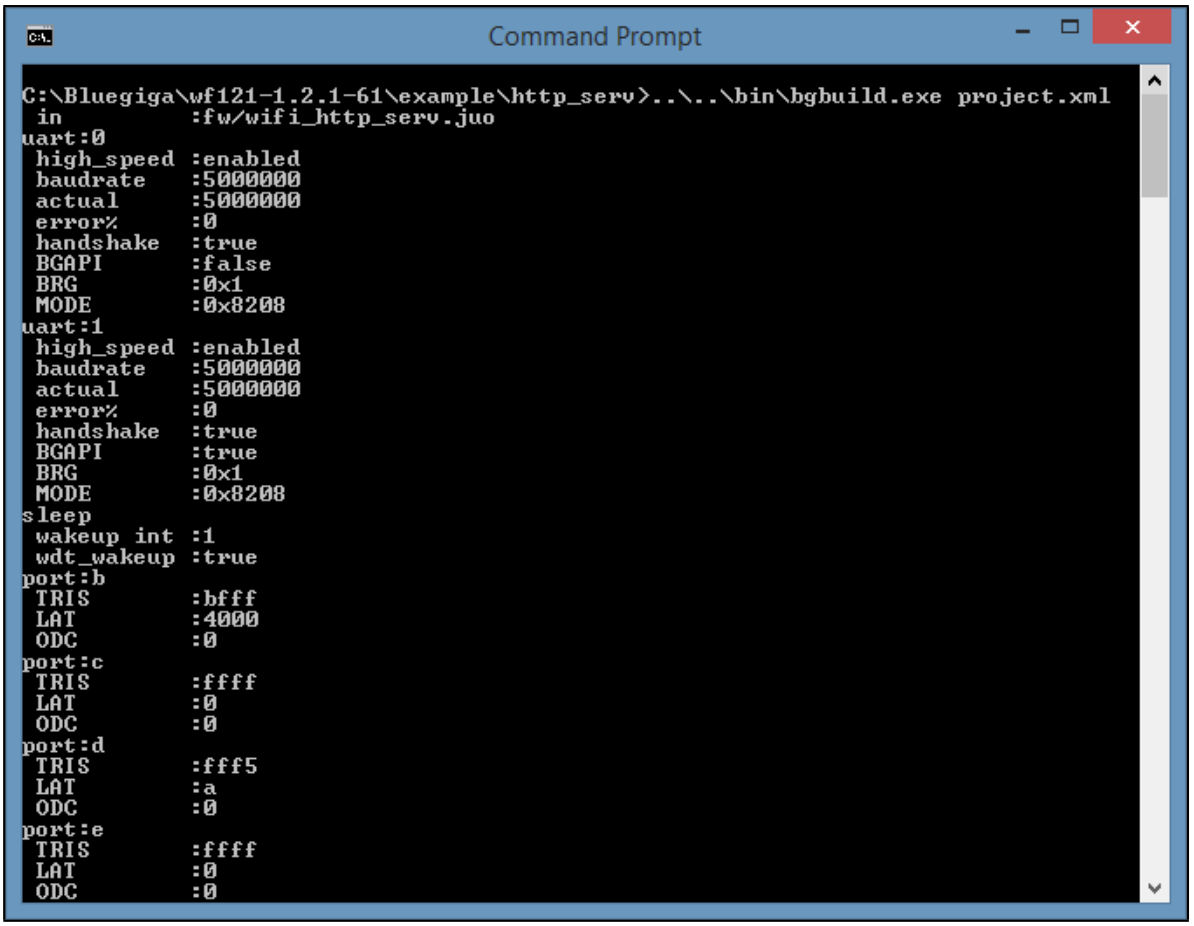


Figure 30: Compiling the project

The compiler output will print the following information:

Feature	Output	Explanation
uart:0	high_speed :enabled baudrate :5000000 actual :5000000 error% :0 handshake :true BGAPI :false BRG :0x1 MODE :0x8208	Shows that UART1 is enabled at 5Mbps baud rate. RTS/CTS handshaking is enabled.
uart:1	high_speed :enabled baudrate :5000000 actual :5000000 error% :0 handshake :true BGAPI :true BRG :0x1 MODE :0x8208	Shows that UART2 is enabled at 5Mbps baud rate. RTS/CTS handshaking is enabled. BGAPI protocol is also ENABLED for UART2.
sleep	wakeup int :1 wdt_wakeup :true	Interrupt INT0 is enabled (pin 38)
port: N	TRIS :ffff LAT 0 ODC 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:/Bluegiga/wf121-1.2.1-61/bin/script_compiler.exe script :main.bgs api : C:/Bluegiga/wf121-1.2.1-61/api/wifiapi.xml stack :512 variables :303	Shows the directory where the bgbuild compiler is located. Shows the BGScript source code file.
HTTP server	FILE :404.html size:202 FILE :main.html size:555 FILE :ap.html size:1622 FILE :appl.html size:1575 FILE :client.html size:2744 FILE :index.html	Shows the HTML pages and their sized that are compiled into the firmware.

	size:1140 FILE :logo.gif size:1011 FILE :style.css size:1255	
Stack size	SW :402104 HW :122 USB :0 Script:1811 Files :10268 Free :97695/512000(19%)	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. Files shows the size if the HTTP server files Free shows the total size of the software and how much space is left in the device.

5 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 PickIT3 tool and software.

5.1 Using PICkit 3

- As **PICkit 3** will erase the full flash, please write down the MAC (IEEE) address of your device
- Download and install **PICkit 3** software from Bluegiga web site
- 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

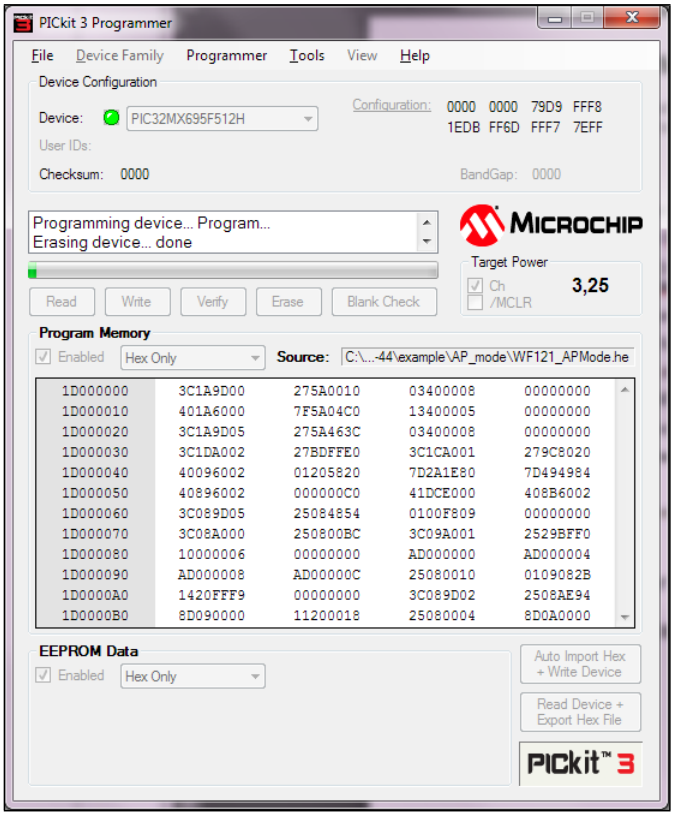


Figure 31: Programming firmware using PICkit 3

5.2 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 PC via UART or USB
- 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 version: 4**
    - 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**

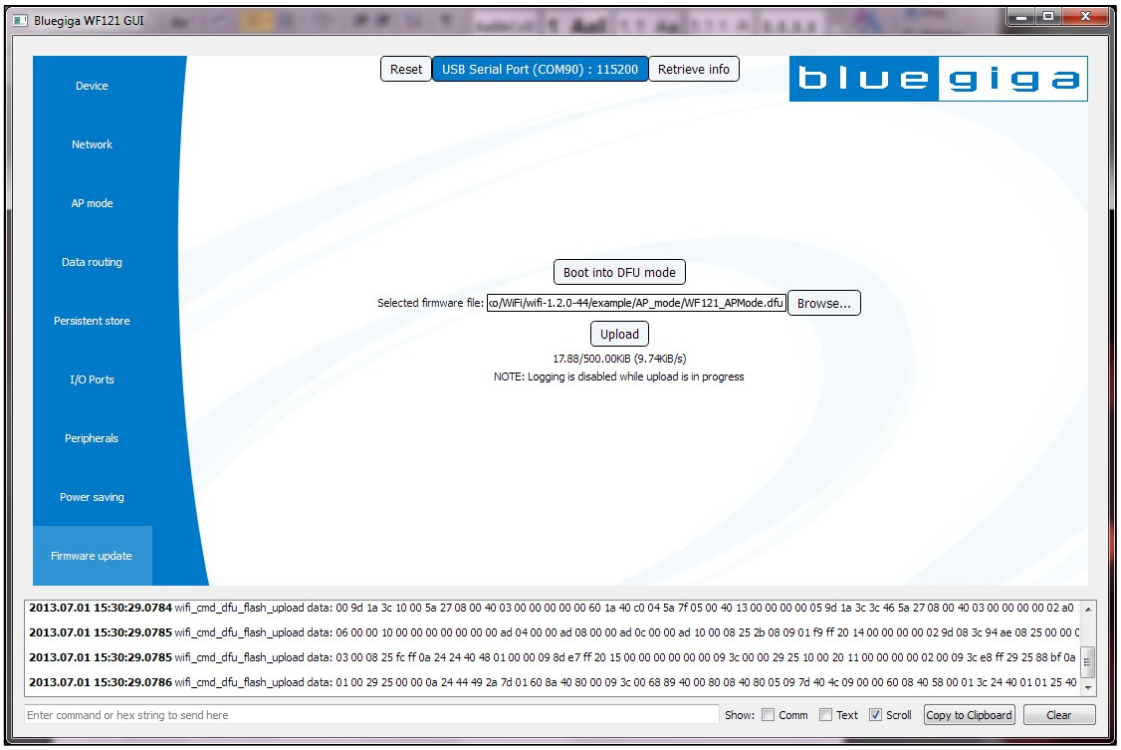


Figure 32: Updating firmware via DFU

Note:

DFU updates allow the change of hardware settings (hardware.xml), so make sure you do not make unwanted changes to your hardware configuration when performing DFU updates.

6 Testing the Access Point Mode

The following instructions briefly show how to test the Access Point mode and built in HTTP server using the WF121 development kit and a PC.

6.1 Test Setup

- WF121 development kit
  - Access Point mode firmware installed and not configured before
  - Connected to a Windows PC via USB (USB-to-Serial)
  - WiFiGUI software
- A Windows PC with a web browser

6.2 Getting Started

- Connect the WF121 development kit to the PC with a micro USB cable (via USB-to-Serial interface)
- Make sure the Windows enumerates two virtual COM ports
  - If you get prompted for a driver, install the driver from the Wi-Fi Software SDK's **driver** folder
- Open WiFiGUI software and select the second COM port (typically larger) at 115200bps baud rate and press **Attach**
- Once the COM port is open, press the **Reset** button on the WF121 development kit to reset the WF121 module and see the boot prompt.

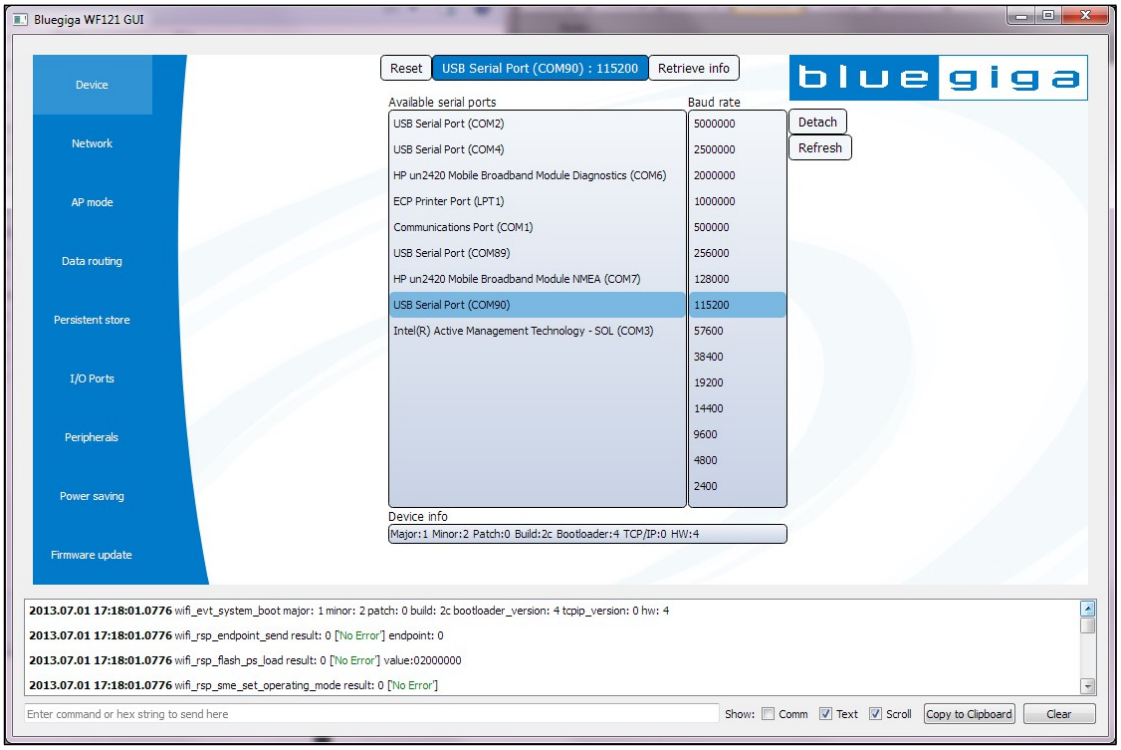


Figure 33: WiFiGUI software

- Make sure you see the **wifi_evt_system_boot** event, indicating the communications work and that firmware boots correctly.

The software will now work as follows:

- It will scan for nearby Access Points and store ten (if visible) of the Access Points with strongest RSSI into the PS keys.
- Once the scan is complete the firmware will switch to a Wi-Fi AP mode, start the HTTP server and wait for an incoming client connection.
- By default the DHCP server is enabled and the device will use IP-address 192.168.1.1 and the first client will get the following IP address 192.168.1.2

Note:

This will typically take around 15-30 seconds and the process can be monitored through WiFiGUI software.

6.3 Connecting to the Access Point

In order to connect to the Access Point please perform the following steps:

- Use the Wi-Fi tool in Windows to scan for the nearby Wi-Fi Access Points
- Look for a device with the SSID : **Bluegiga-WF121**
    - By default no security is used
- Click the **Bluegiga-WF121** and **Connect** button in order to connect it
    - Notice that you might get a warning message, since WF121 will not provide Internet access to the PC

Figure 34: Scanning for Wi-Fi networks in Windows 7

- Once connected open your preferred web browser and type http://192.168.1.1/ into the address bar
    - Notice that it might take a while for the Windows to load the page

- You should now see the main landing page with the following information

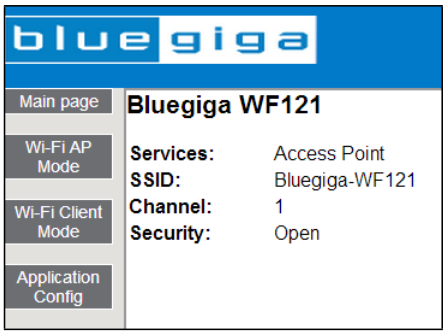


Figure 35: Main landing page

6.4 Changing Access Point Mode Settings

Wi-Fi AP Mode page allows you to change the settings of Access Point such as SSID, security mode etc.

In order to change the settings:

- Navigate to the Wi-Fi AP Mode page
- Change the desired settings
- Press Save to save the settings for the next reboot or alternatively press Activate, which will save the settings and reboot the WF121 module

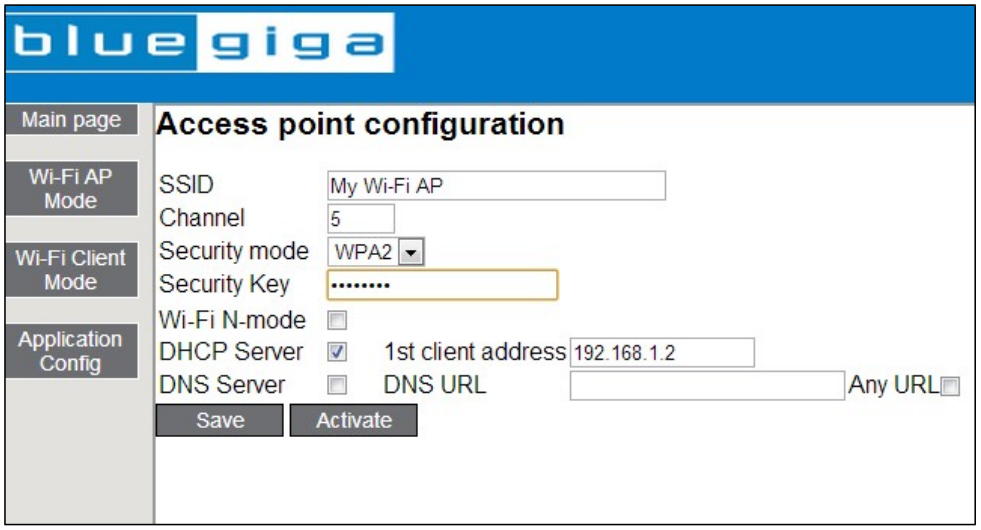


Figure 36: Access Point mode settings

6.5 Changing the Application Parameters (PS Keys)

Application configuration page allows you to read or change the use specific PS keys. At the moment four (4) integer and five (5) string values are supported as well three (3) button interactions, which can be caught via BGScript or BGAPI.

In order to change the parameters:

- Type the integer values (max 11 digits) into Integer 1 – 4 fields
- Type the string values (max 64 bytes) into String 1 – 5 fields
- Press **Save** to store the values into PS keys

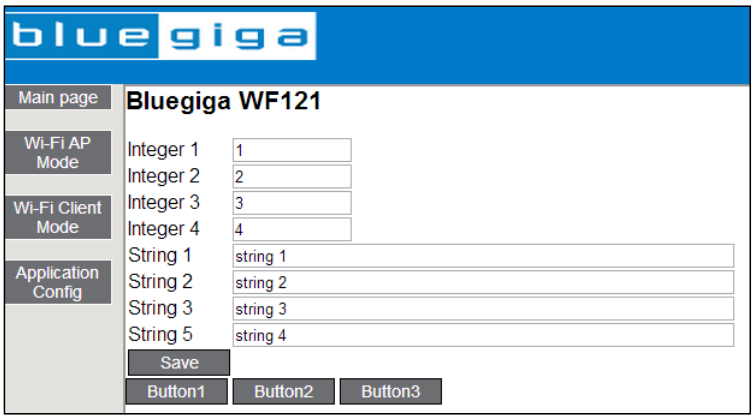


Figure 37: Application configuration page

Pressing Button 1, 2 or 3 will cause **wifi_evt_https_button** events to be generated with, which can be caught with BGScript or BGAPI application.

6.6 Switching to Wi-Fi Client Mode

You can reconfigure the WF121 module into a Wi-Fi client mode and have it connect to your existing Wi-Fi infrastructure as a client. This can be performed from the **Wi-Fi Client Mode** page.

In order to configure your device into a client mode:

- Navigate to the **Wi-Fi Client Mode** page
- Select the network you want to connect to
- Type the security key into the **Security key** box
- Press **Connect** to save the settings and reboot the device

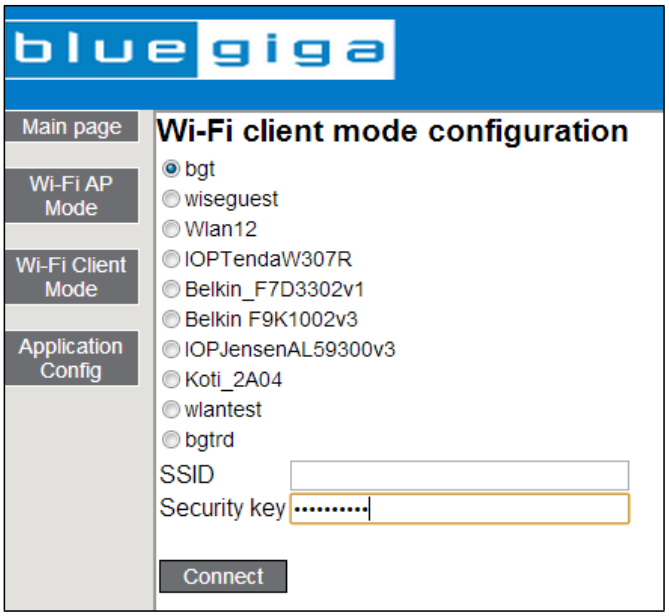


Figure 38: Wi-Fi client mode configuration

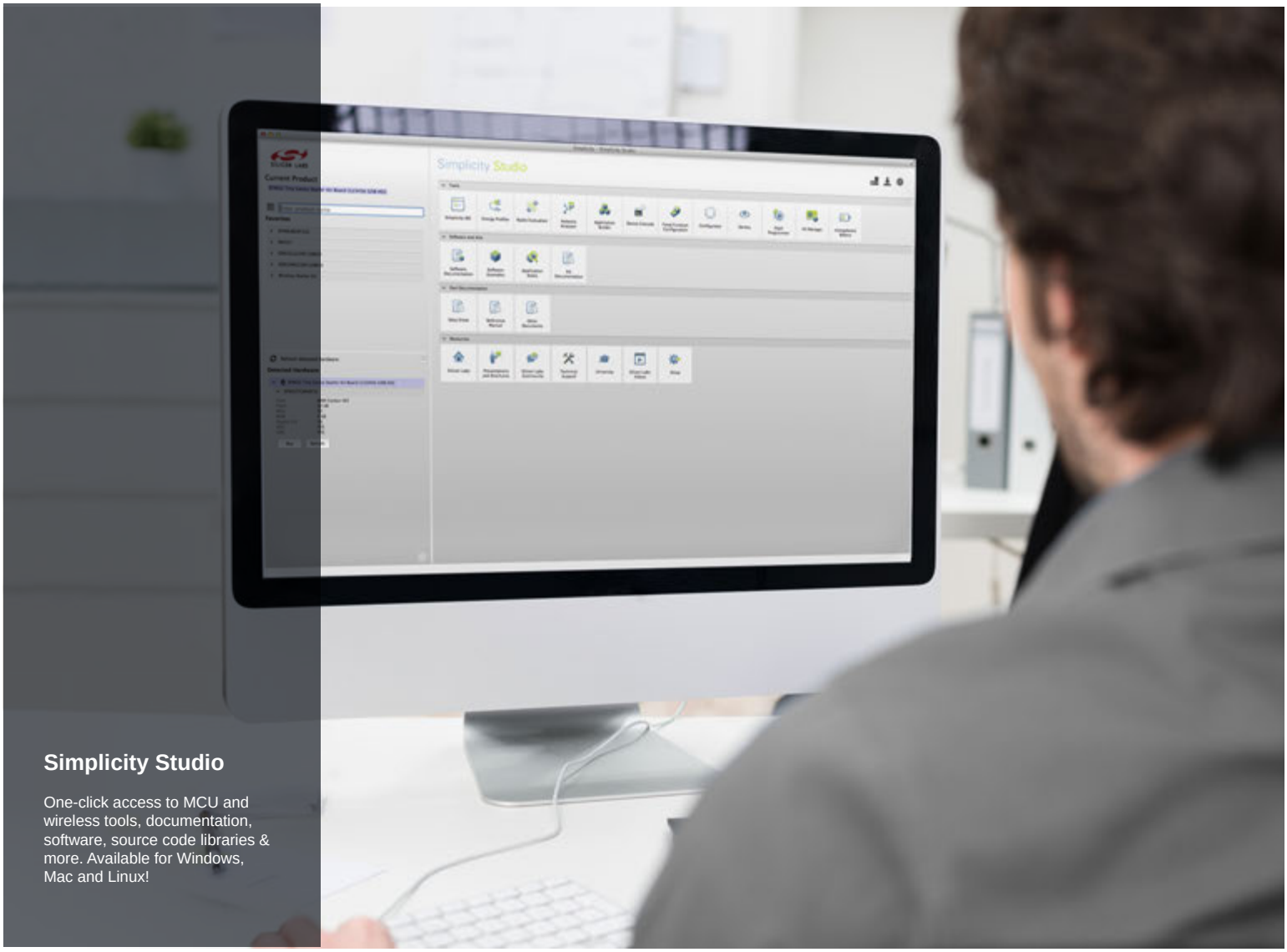
The WF121 module now reboots and tries to connect to the select network as a Wi-Fi client. If the connection is successful the device tries to get an IP address from the DHCP server. Notice that no services (TCP server/client, UDP client/server or HTTP server) are not started but you should still be able to ping the device.

Note:

The WF121 will try to connect the Wi-Fi network three (3) times and if the connection fails it will fall back to the Access Point mode. The reconnection count can be configured in the BGScript code.

6.7 Restoring Default Values

The device can be reset to factory default settings (Access Point mode parameters) via INT0 interrupt, which on WF121 development kit is connected to **RD5** button. Pressing the **RD5** button will cause the device to reset.



Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com