AN4486
Application note

BlueNRG, BlueNRG-MS over-the-air bootloader

Introduction

This application note describes the BlueNRG and BlueNRG-MS over-the-air (OTA) bootloader running on top of Bluetooth low energy (BLE), provided with the BlueNRG and BlueNRG-MS development kits from STMicroelectronics. It begins with some concepts related to the OTA bootloading process, and then walks the user through all the steps required to run some bootloading sessions.

Note: The application note content is valid for both BlueNRG and BlueNRG-MS devices. Any reference to BlueNRG device is also valid for the BlueNRG-MS device. Any specific difference is highlighted wherever relevant.

Contents

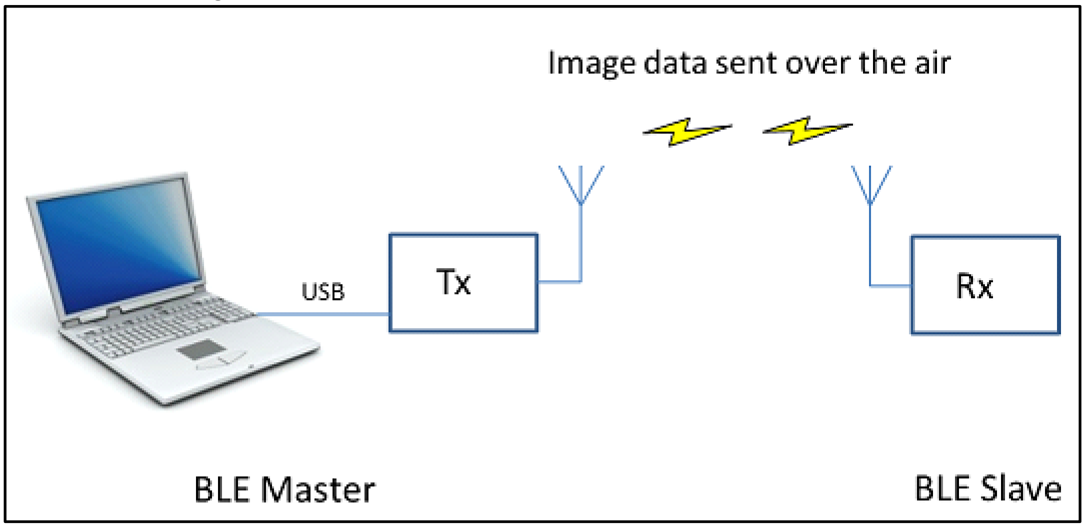
List of tables

List of figures

1 The concept of OTA bootloading

The OTA bootloader is essentially a tool that allows a slave device to receive a firmware image over-the-air from a master device and write it into Flash memory. To put things in the context of Bluetooth low energy, the OTA bootloader is a service offering its own characteristics and can coexist with other services used by any given application running on the BLE stack. In our implementation, the BLE master will be a combined system made up of BlueNRG and BlueNRG-MS kits platforms connected to a PC through USB. The BlueNRG or BlueNRG-MS platform is driven by the BlueNRG GUI running on the connected PC. This choice allows us to benefit from numerous resources available from the PC, specifically with regard to compilers for firmware image generation and the memory space required to store images before they are sent over-the-air for bootloading.

Figure 1: BlueNRG, BlueNRG-MS OTA master & slave devices



Note: For more information related to the BlueNRG and BlueNRG-MS kit platforms, refer to the related user manuals available on ST BlueNRG web pages (see *Section 6: "Related documentation and references"*).

Table 1: "BlueNRG and BlueNRG-MS platform naming convention" describes the naming convention used for referring to the BlueNRG and BlueNRG-MS platform LEDs and buttons.

Table 1: BlueNRG and BlueNRG-MS platform naming convention

Resource name	BlueNRG, BlueNRG-MS development platforms	BlueNRG, BlueNRG-MS USB dongles
LED_D1	DL1 (yellow LED)	D2 (red LED)
LED_D2	DL2 (orange LED)	D3 (orange LED)
BUTTON_1	Push_Button	SW1
BUTTON_2	Joystick SEL	SW2

2 OTA bootloader service description

The OTA bootloader service is addressed through the files btl.[ch] provided within the BlueNRG DK software which supports both BlueNRG and BlueNRG-MS devices.

The following provides a brief description of the OTA bootloader service and related characteristics:

- Bootloader OTA service: the bootloader service
  - aci_gatt_add_serv(UUID_TYPE_128, UUID, PRIMARY_SERVICE, 10, &btlServHandle)
- Btl Image characteristic: contains information related to the lower and higher bounds of free memory as suggested by the current application that includes the OTA bootloader service
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, UUID, 8, CHAR_PROP_READ, ATTR_PERMISSION_NONE, GATT_INTIMATE_APPL_WHEN_READ_N_WAIT,16, 0, &btlImageCharHandle)
- Btl new image characteristic: contains the base address and the size of the image that the master wants to send over-the-air
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, UUID, 9, CHAR_PROP_READ|CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP, ATTR_PERMISSION_NONE, GATT_INTIMATE_APPL_WHEN_READ_N_WAIT | GATT_SERVER_ATTR_WRITE,16, 0, &btlNewImageCharHandle)
- Btl new image content characteristic: contains a 16-byte block of firmware image data sent by the master (through a characteristic write command) along with some control information such as block sequence number (2 bytes) and checksum for integrity check (1 byte)
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, UUID, 20, CHAR_PROP_READ|CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP, ATTR_PERMISSION_NONE, GATT_INTIMATE_APPL_WHEN_READ_N_WAIT | GATT_SERVER_ATTR_WRITE,16, 0, &btlNewImageTUContentCharHandle)
- Btl expected image sequence number characteristic, through which the slave device notifies the master about the next block it expects
  - aci_gatt_add_char(btlServHandle, UUID_TYPE_128, UUID, 4, CHAR_PROP_NOTIFY|CHAR_PROP_READ, ATTR_PERMISSION_NONE, GATT_INTIMATE_APPL_WHEN_READ_N_WAIT, 16, 0, &btlExpectedImageTUSeqNumberCharHandle)

Note: OTA bootloader service and proprietary 128-bit UUID characteristics are defined within the file btl.c.

2.1 Bootloading transactions

This section walks through the steps for OTA bootloading:

1. Once the master and slave running the OTA bootloader service are set, a discovery procedure needs to be done in order for the two devices to connect. Discovery is achieved by listening to advertisements coming from devices within the radio range (active scan) and selecting the ones containing the OTA BTL service UUID 128-bits within the scan response.
2. The name of the selected device will be read from the advertising message to be utilized by the master in order to enhance the slave identification procedure.

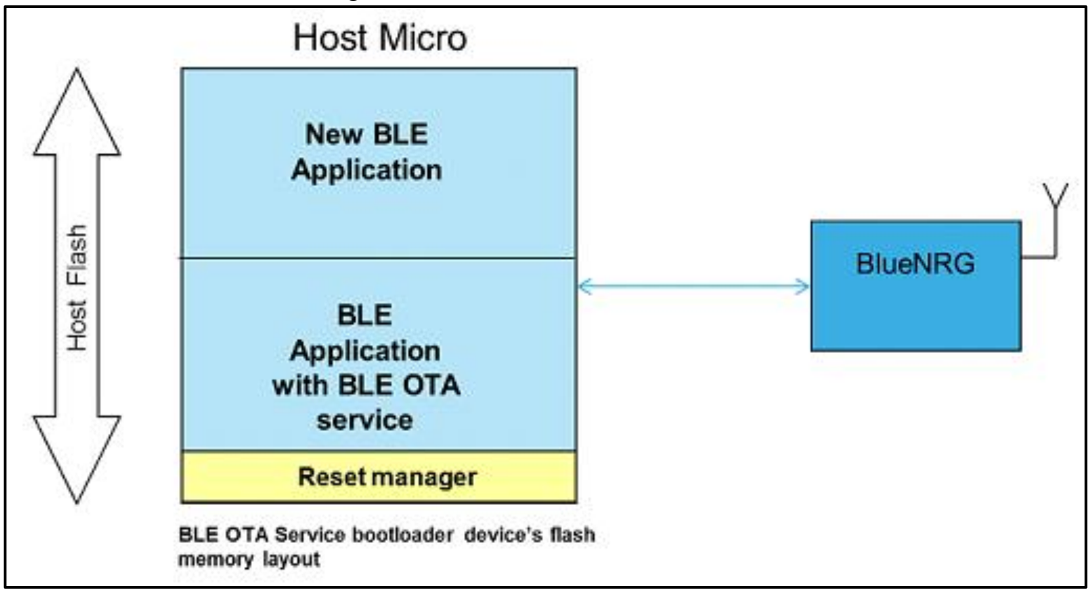
3. After connection, the master will send “GATT DISC CHARAC BY UUID” commands in order to read all OTA bootloader characteristic handles.
4. The master will read the Btl Image characteristic through a “GATT READ CHARACTERISTIC VALUE” command to be aware of free space on the target slave device's Flash memory.
5. Based on the information in the previous step, the master will select an appropriate image to send over-the-air. The candidate image (residing somewhere on the master as a *.bin file) must fit within the target's free Flash range and must be 512-byte aligned as requested by STM32L interrupt vector constraints. Once the selection is made, the master sends a “GATT WRITE CHARACTERISTIC VALUE” in order to write the image base address and size into the Btl new image characteristic and read it back through “GATT READ CHARACTERISTIC VALUE” to verify it.
6. The master writes into the Btl expected image sequence number characteristic descriptor to enable slave notifications for image block sequence numbers and errors. Once the slave receives this command it also erases pages within the Flash memory range written within the Btl new image characteristic by the master, and sends back a notification.
7. The image transfer begins. The master will send the image in block of 16 bytes through a sequence of “GATT WRITE WITHOUT RESPONSE” commands (one for each 16-byte block). Each time a new WRITE command lands on the target slave it will write the new block of data within the Btl new image content characteristic. Each 16-byte block comes along with a 2-byte long sequence number and one-byte long checksum field in order to check for sequencing and message integrity at destination. Every time the slave's internal buffer is filled up by 16 byte blocks, it will download into Flash memory. Once the slave has completed the management of the current 16-byte block, it will send a notification message back to the master providing the block number of the next expected block. It might notify Flash write errors and Flash verify errors as well, in which case the bootloading session should stop under the assumption that we are dealing with issues on the destination device Flash.
8. If the number of bytes downloaded successfully on the destination device Flash is equal to the initially provided image size information, the bootloader will write the base address and end address of the new image at the last two locations of EEPROM memory (at address 0x08080FF8) and transfer control to the location containing the firmware of the new application.

2.2 BLE OTA service bootloader architecture

The BLE OTA service bootloader architecture includes the following components:

- Reset manager
  - This should start after reset and pass control to the latest updated & valid BLE application.
  - BLE application with BLE OTA service This is the running BLE application which allows to receive new application image packets over-the-air and stores them in a specific Flash section.
  - New BLE application
    - this is the application image downloaded through RF link
    - To take full advantage of the BLE OTA service approach, the new BLE application can be built with the BLE service in order to provide itself with OTA service bootloader capability.

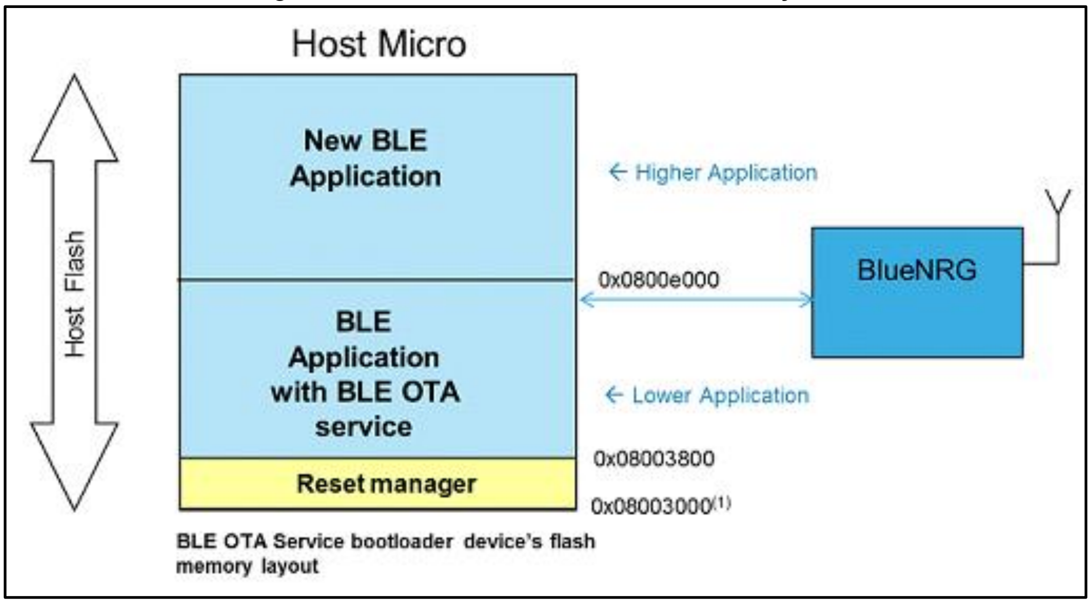
Figure 2: BLE OTA service bootloader


Host Micro
Host Flash
New BLE Application
BLE Application with BLE OTA service
Reset manager
BlueNRG
BLE OTA Service bootloader device's flash memory layout

2.2.1 Reset Manager

The Flash base for STM32L1 starts at 0x8000000, meaning that at device reset, the microcontroller will start executing application images sitting on that boundary. All the BlueNRG and BlueNRG-MS kits platforms are preprogrammed with a DFU application allowing the download of binary images into the internal Flash of the STM32L. Once the OTA bootloader writes a new image starting from a base address that is above the Flash base + DFU size (0x3000), we need a way to allow transfer of control towards the new application every time we reset. For that purpose, every device on which we plan to include the OTA bootloader service will, at first, need to be loaded with a Reset Manager starting from the Flash base address + DFU size. At device reset, the Reset Manager will take care of jumping to the location of the last image that was successfully loaded by the OTA bootloader, as indicated in the higher 8 bytes within the EEPROM memory range of the device (starting at address 0x08080FF8). To summarize, *Figure 3: "BLE OTA service bootloader Flash layout"* shows a typical Flash memory layout for a device with OTA bootloader service on-board, with an STM32L as the host microcontroller. At device initial setup, we will utilize the BlueNRG GUI in order to upload in Flash the application into the STM32L at address 0x8003800 (Reset Manager upper end), then we will do the same in order to upload the Reset Manager itself at address 0x8003000. Actions must be taken to erase the relevant EEPROM locations for proper Reset Manager initial setup as well. On the BlueNRG and BlueNRG-MS kits platform, this can be achieved by keeping BUTTON_2 pressed while pushing the reset (LED_D1 will light up for five seconds). This will cause EEPROM location erase, which means an “all 0” read by the Reset Manager after next pushing of the reset button. As a consequence, by default the Reset Manager assumes that it must jump to 0x8003800, which is exactly the reference base address for the application to run at first setup.

Figure 3: BLE OTA service bootloader Flash layout



(1) It assumes DFU.hex application is on Flash base 0x08000000.

Note: OTA_ResetManager IAR project with related header & source files is provided within the BlueNRG DK software package, Project folder. Also, a prebuilt image is provided within the folder Project/OTA_prebuilt_images. This IAR project is valid for both BlueNRG and BlueNRG-MS devices.

2.2.2 BLE application with BLE OTA service

In order to add the BLE OTA service provided within the file btl.c, follow these steps:

1. On EWARM workspace, add ST_OTA_BTL preprocessor & linker option
2. On EWARM workspace, use the reference stm32l1xx_flash.icf linker file for matching the Flash layout described in *Figure 3: "BLE OTA service bootloader Flash layout"* (it is provided within the BlueNRG SW package, Projects\Project folder)
3. On EWARM workspace, add reference to file btl.c and add path related to btl.h file
4. On BLE user application, include the header file btl.h
5. On BLE user application, call the btl.c function Add_Btl_Service() for adding the OTA BLE service and related characteristics
6. On BLE user application, add OTA service UUID to scan response: hci_le_set_scan_resp_data(18,BTLServiceUUID4Scan)
7. On BLE user application, HCI event handler (HCI_Event_CB), EVT_BLUE_GATT_ATTRIBUTE_MODIFIED event, add the call to the OTA_Write_Request_CB() API defined within btl.c file
8. On BLE user application, HCI event handler (HCI_Event_CB), add the EVT_BLUE_GATT_READ_PERMIT_REQ event case and call to aci_gatt_allow_read() API (for allowing BLE stack to send a response)

Note: In order to easily identify the IAR BlueNRG demo applications built with the OTA bootloader service, the platform LEDs are utilized based on the schemes in *Table 2: "LEDs* *for BlueNRG demo lower applications using BLE OTA bootloader service"* and *Table 3: "LEDs for BlueNRG demo higher applications using BLE OTA bootloader service"*.

Table 2: LEDs for BlueNRG demo lower applications using BLE OTA bootloader service

BlueNRG, BlueNRG-MS development platforms (lower application with OTA bootloader service)	BlueNRG, BlueNRG-MS USB dongles (lower application with OTA bootloader service)
DL1 (yellow) is always blinking[(1)]	D2 (red) is always blinking [(1)]
DL2 (orange) is ON during OTA upgrade	D3 (orange) is ON during OTA upgrade

Notes:

[(1)]When device is connected, the LED blinks faster.

Table 3: LEDs for BlueNRG demo higher applications using BLE OTA bootloader service

BlueNRG, BlueNRG-MS development platforms (higher Application with OTA bootloader service)	BlueNRG, BlueNRG-MS USB dongles (higher application with OTA bootloader service)
DL2 (orange) is always blinking [(1)]	D3 (orange) is always blinking [(1)]
DL1 (yellow) is ON during OTA upgrade	D2 (red) is ON during OTA upgrade

Notes:

[(1)]When device is connected, the LED blinks faster.

2.2.3 New BLE application

New BLE application can be built for supporting the BLE OTA bootloader service in order to offer the OTA bootloading capability (beyond its specific BLE application scenario which involves using other BLE services and related characteristics). Users can follow the same steps as described on *Section 2.2.2: "BLE application with BLE OTA service"*.

If the BLE application is built to reside on the higher Flash (refer to higher application in *Figure 3: "BLE OTA service bootloader Flash layout"*), the HIGHER_APP preprocessor & linker options must be used. BlueNRG demo applications built with OTA bootloader service support are tailored for using 0x0800e000 as reference base address for higher Application. The related vector table base offset 0xe000 is defined, within file btl.h, as follows:

VECTOR_TABLE_BASE_DFU_OFFSET +
VECTOR_TABLE_BASE_HIGHER_APP_OFFSET

where:

VECTOR_TABLE_BASE_DFU_OFFSET = 0x3000 (DFU size).

VECTOR_TABLE_BASE_HIGHER_APP_OFFSET = 0xB000 (User defined location).

The user can customize it by modifying the linker file and base address VECTOR_TABLE_BASE_HIGHER_APP_OFFSET on file btl.h.

Note: If HIGHER_APP preprocessor & linker options are not used, the BLE application is built, by default, for residing at default base address: 0x08003800.

The related vector table base offset 0x3800 is defined, within file btl.h, as follows:

VECTOR_TABLE_BASE_DFU_OFFSET +
VECTOR_TABLE_BASE_RESET_MANAGER_OFFSET

Where:

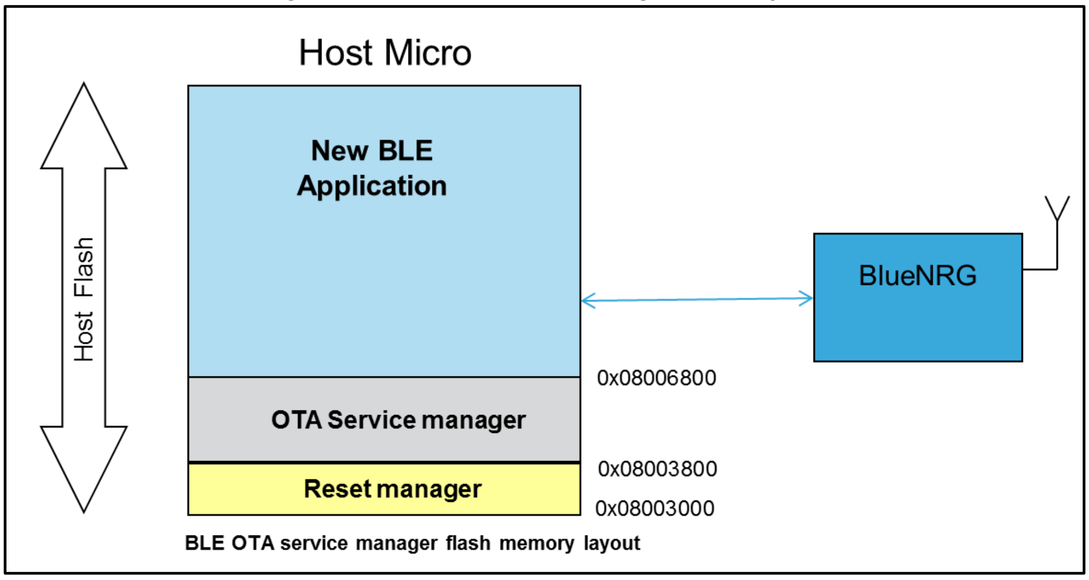
VECTOR_TABLE_BASE_DFU_OFFSET = 0x3000 (DFU size).

VECTOR_TABLE_BASE_RESET_MANAGER_OFFSET = 0x800 (Reset Manager size).

2.3 OTA bootloader service manager (standalone OTA bootloader application)

A simpler approach coming from the BLE OTA service architecture described in *Figure 2: "BLE OTA service bootloader"*, consist of using a basic OTA Service Manager application which only includes the BLE OTA service & characteristics.

Figure 4: BLE OTA Service Manager Flash layout



[1] It assumes DFU.hex application is on Flash base address 0x8000000.

The BLE OTA Service Manager architecture includes the following components:

- Reset manager
  - This should start after reset and pass control to the latest updated & valid BLE application.
- OTA Service Manager
  - This contains only the BLE OTA service & characteristics and provides the OTA bootloader capability to any BLE application.
- New BLE application
  - This is the application image downloaded through the OTA Service Manager.
  - It does not require that the OTA bootloader service be included.
  - In order to use this capability, the BLE application must activate the OTA Service Manager (a specific API Switch_To_OTA_Service_Manager_Application() function is provided within btl.c file).
  - It is always placed at fixed Flash address: 0x8006800 .

2.3.1 Reset Manager

Refer to *Section 2.2.1: "Reset Manager"*.

2.3.2 OTA Service Manager

The OTA Service Manager acts as a standalone OTA bootloader providing the BLE OTA service to any BLE application which wants to use bootloader features without the need to include any OTA capability.

The OTA Service Manager application IAR projects + header & source files are provided within the BlueNRG SW package on Project folder, as follows:

1. OTA_ServiceManager\EWARM\OTA_ServiceManager.eww for the BlueNRG device.
2. OTA_ServiceManager\EWARM_BlueNRG-MS\OTA_ServiceManager.eww for the BlueNRG-MS device.

The OTA Service Manager application uses the platform LEDs as follows:

Table 4: LEDs for BlueNRG OTA Service Manager

BlueNRG, BlueNRG-MS development platforms	BlueNRG, BlueNRG-MS USB dongles
DL1 (yellow) is always blinking [(1)]	D2 (red) is always blinking [(1)]
DL2 (orange) is ON during OTA upgrade	D3 (orange) is ON during OTA upgrade

Notes:

[(1)]When device is connected, the LED blinks faster.

2.3.3 New BLE application

A BLE application which wants to use the OTA Service Manager capability is only required to address these basic steps:

1. On EWARM workspace, add ST_OTA_BASIC_APPLICATION preprocessor & linker option
2. On EWARM workspace, use the reference stm32l1xx_flash.icf linker file for matching the Flash layout described in *Figure 4: "BLE OTA Service Manager Flash layout"* (it is provided within the BlueNRG SW package, demo applications folder)
3. On EWARM workspace, add reference to file btl.c and add path related to btl.h file
4. On BLE user application, include the header file btl.h
5. Add some user code for allowing BLE application to launch the OTA Service Manager application. A simple button could be used for controlling the call to Switch_To_OTA_Service_Manager_Application (this API is provided within btl.c file).

Note: When using the OTA Service Manager, the BLE application does not need to include any OTA bootloader service. Also, the application can always be placed at fixed base address 0x08006200. The related vector table base offset 0x6800 is defined within file btl.h, as follows:

VECTOR_TABLE_BASE_DFU_OFFSET +
VECTOR_TABLE_BASE_RESET_MANAGER_OFFSET +
VECTOR_TABLE_BASE_ADDRESS_OTA_BASIC_APP

where:

VECTOR_TABLE_BASE_DFU_OFFSET = 0x3000 (DFU size).

VECTOR_TABLE_BASE_RESET_MANAGER_OFFSET = 0x800 (Reset Manager size).

VECTOR_TABLE_BASE_ADDRESS_OTA_BASIC_APP = 0x3000 (OTA Service Manager size).

Note: In order to easily identify the BlueNRG demo applications using the OTA Service Manager, the platform LEDs are utilized based on the scheme in *Table 5: "LEDs for BlueNRG demo applications using BLE OTA Service Manager"*.

Table 5: LEDs for BlueNRG demo applications using BLE OTA Service Manager

BlueNRG, BlueNRG-MS development platforms	BlueNRG, BlueNRG-MS USB dongles
DL2 (orange) is always blinking(1)	D3 (orange) is always blinking (1)
DL1 (yellow) is ON during OTA upgrade	D2 (red) is ON during OTA upgrade

Notes:

(1)When device is connected, the LED blinks faster.

3 Hardware and software resources

Before we explain how to set up for the OTA bootloader demos, we will first provide the list of required hardware and software resources:

Table 6: BlueNRG OTA bootloader HW & SW resources

Status	Description
MANDATORY	2 BlueNRG or BlueNRG-MS kits platforms and related USB cables (only needed for development platforms).
	Reset manager image & prebuilt application images including OTA bootloader service
	BlueNRG graphical user interface v1.4.0 or later
	PC with the following resources: Windows® XP SP3 or Windows® Vista or Windows® 7 At least 128 MB of RAM 2 x USB port 40 MB of hard disk space available Adobe Reader 6.0 or later.
OPTIONAL BUT RECOMMENDED	IAR embedded workbench 6.60. This allows access to the code to learn from the sample application provided
	STLink Flash programmer
	STLink utility: allows to perform Flash management operations useful for testing and verification of bootloading sessions with the STLink Flash programmer

4 Running the OTA bootloader GUI & BlueNRG SW package demo applications

The IAR projects addressing the BlueNRG SW package demo applications Sensor Demo and BLE Chat come with specific pre-configured workspaces allowing use of both approaches:

1. BLE application built to include OTA bootloader service
2. BLE application built to use BLE OTA Service Manager

For both Sensor Demo and BLE Chat demo applications, two IAR projects, under EWARM and EWARM_BlueNRG-MS folders, are available for addressing, respectively, BlueNRG and BlueNRG-MS devices.

Table 7: "Sensor Demo IAR workspaces with OTA bootloader capabilities" gives a simple description of the available IAR workspaces with OTA bootloader features for the Sensor Demo BLE application:

Table 7: Sensor Demo IAR workspaces with OTA bootloader capabilities

Sensor Demo application IAR workspace	Description
LowPower_LowerApp_OTA	LowPower configuration including the OTA bootloader service with "Lower Application" memory layout (refer to *Figure 3: "BLE OTA service bootloader Flash layout"* and *Section 2.2.2: "BLE application with BLE OTA service"* and *Section 2.2.3: "New BLE application"*)
LowPower_HigherApp_OTA	LowPower configuration including the OTA bootloader service with "Higher Application" memory layout (refer to *Figure 3: "BLE OTA service bootloader Flash layout"* and *Section 2.2.2: "BLE application with BLE OTA service"* and *Section 2.2.3: "New BLE application"*)
USB_LowerApp_OTA	USB configuration including the OTA bootloader service with "Lower Application" memory layout (refer to *Figure 3: "BLE OTA service bootloader Flash layout"* and *Section 2.2.2: "BLE application with BLE OTA service"* and *Section 2.2.3: "New BLE application"*)
USB_HigherApp_OTA	USB configuration including the OTA bootloader service with "Higher Application" memory layout (refer to *Figure 3: "BLE OTA service bootloader Flash layout"* and *Section 2.2.2: "BLE application with BLE OTA service"* and *Section 2.2.3: "New BLE application"*)
OTA_Basic	LowPower configuration using the OTA Service Manager (refer to *Figure 4: "BLE OTA Service Manager Flash layout"* and *Section 2.3.2: "OTA Service Manager"* and *Section 2.3.3: "New BLE application"*)

Below is a simple description of the available IAR workspaces with OTA features for the BLE Chat application:

Table 8: BLE_Chat IAR workspaces with OTA bootloader capabilities

BLE_Chat application IAR workspace	Description
Server_LowerApp_OTA	Server configuration including the OTA bootloader service with “Lower Application” memory layout (refer to *Figure 3: "BLE OTA service bootloader Flash layout"* and *Section 2.2.2: "BLE application with BLE OTA service"* and *Section 2.2.3: "New BLE application"*)
Server_HigherApp_OTA	Server configuration including the OTA bootloader service with “Higher Application” memory layout (refer to *Figure 3: "BLE OTA service bootloader Flash layout"* and *Section 2.2.2: "BLE application with BLE OTA service"* and *Section 2.2.3: "New BLE application"*)
Server_OTA_Basic	Server configuration using the OTA Service Manager (refer to *Figure 4: "BLE OTA Service Manager Flash layout"* and and *Section 2.3.2: "OTA Service Manager"* and *Section 2.3.3: "New BLE application"*)

Note: The Sensor Demo application is only supported on the BlueNRG, BlueNRG-MS development platforms.

4.1 BLE application built to use the OTA bootloader service

The following is a step-by-step description of how to run the BlueNRG GUI OTA bootloader using the Sensor Demo applications built with OTA service support and two development platforms:

1. Plug two development platforms into the USB connectors on a PC.
2. Use the BlueNRG GUI to program one of the two platforms kits to be used as OTA bootloader slave. This requires plugging the board into the USB connector on the PC running the GUI, then pressing the platform reset button while keeping the BUTTON_1 pressed in order to start the DFU bootloader (LED_D2 should be blinking at this point).
   a. From Tools->Flash Motherboard FW, select the Reset Manager hex prebuilt image (STM32L\Project\OTA_prebuilt_images\ DFU_ResetManager.hex) to upload the Reset Manager on board (hex format implicitly takes into account the proper application base address which, for the case of the application, will be 0x8003000).
   b. The same actions, starting from DFU bootloader activation, must be taken to upload the initial OTA bootloader application on the board (STM32L\Project\OTA_prebuilt_images\ BlueNRG_DFU_OTA_Lower_SensorDemo.hex for a BlueNRG device or BlueNRG-MS_DFU_OTA_Lower_SensorDemo.hex for a BlueNRG-MS device); (hex format implicitly takes into account the proper application base address which, for the case of the application, will be 0x8003800).
   c. After Reset Manager loading, a reset while keeping BUTTON_2 pressed needs to be performed in order to erase obsolete data from the OTA bootloader's EEPROM and guide proper Reset Manager behavior at first startup (5 seconds LED_D1 turned on signals successful run of the process).
   d. (Optional) Get a “BlueNRG app for smartphones (IoS or Android)” to discover,

connect and play with the Sensor Demo application just loaded on the development platform at step 2.b. Just to verify that an ordinary BLE application has services coexisting with the OTA bootloader service which will run next.

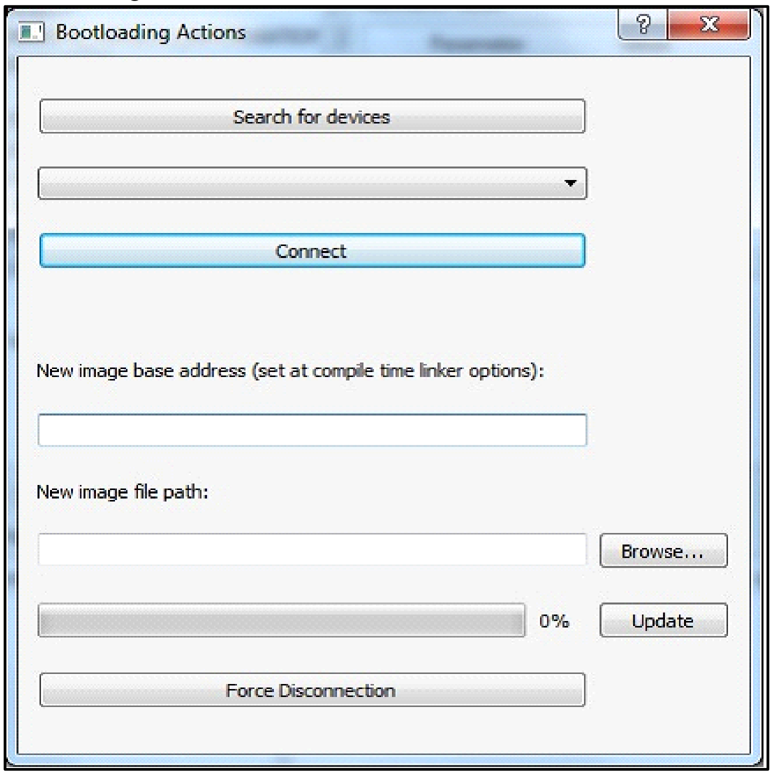
e. (Optional) If actions at step 2.d were taken, disconnect the "BlueNRG app for smart phones (IoS or Android)" enabled device from the development kit platform. The Sensor Demo application will be disconnected and it will start advertising. This will make this application a potential candidate for a BLE OTA upgrade through the BlueNRG GUI.

3 Program the second platform with the required application to be used with the BlueNRG GUI (BlueNRG_VCOM_x_x.hex file available on the BlueNRG DK SW package, on Firmware\STM32L1_prebuilt_images folder).

4 Open the BlueNRG GUI on the PC and select the COM port related to the BlueNRG platform configured in step 3, through the drop-down "Port" and press "Open".

5 Select "Tools" -> "OTA bootloader" to open the dialog box containing the OTA bootloader actions and press "Search for devices".

Figure 5: BlueNRG GUI: OTA Bootloader window



New image base address (set at compile time linker options):

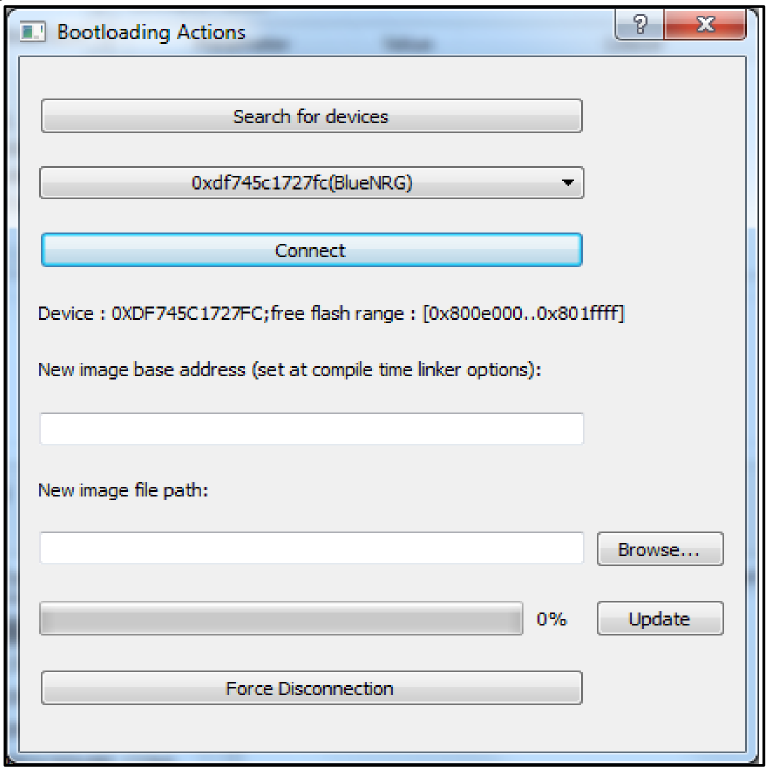
New image file path:

6 After "Search for devices", the GUI will start the discovery process and return information related to the address and application names of devices running the OTA bootloader service within the radio range. Once the previous process ends, the device list can be opened through the combo box arrow below the "Search for devices" button, and the user is free to choose the device to connect for the bootloader process using the "Connect" button (application address & names are displayed). In scenarios where device address and application running are not enough to identify the right device, we recommend to implement a connection active indicator on slave devices running the

OTA bootloader service. As an example, in our demo platform we use an LED blinking at higher frequency when connection is established, and this can be used as visual feedback to obtain confirmation related to the device being addressed. If we realize we have connected the wrong device, we can simply press the 'Force Disconnection' button and return to device selection within the combo box. After device selection and connection through the “Connect” button, we can read the related free memory range that it advertises, which constrains us to provide an image file compiled with a base address and size within the expected range on the slave device.

a. As a next step, we need to provide the “Base address” and press the browse button to select the location of the file containing the image that we plan to send over-the-air to the slave device. For that purpose, the user can open the BlueNRG IAR workspace SensorDemo, LowPower_HigherApp_OTA and build the related DFU_OTA_Higher_SensorDemo.bin image (available in the SensorDemo\EWARM\LowPower_HigherApp_OTA\Exe folder) containing a version of the Sensor Demo that is compiled with base address equal to 0x800e000. Apart from the memory offset, the only difference from LowPower_LowerApp_OTA is the color of the flashing LEDs (LowerApp blinks LED_D1, Higher_App blinks LED_D2). Alternatively, user can open the BlueNRG-MS IAR workspace SensorDemo, LowPower_HigherApp_OTA and build the related DFU_OTA_Higher_SensorDemo.bin image (available in the SensorDemo\EWARM_BlueNRG-MS\LowPower_HigherApp_OTA\Exe folder).

Figure 6: BlueNRG GUI: OTA bootloader search for devices and connect



Device : 0XDF745C1727FC;free flash range : [0x800e000..0x801ffff]

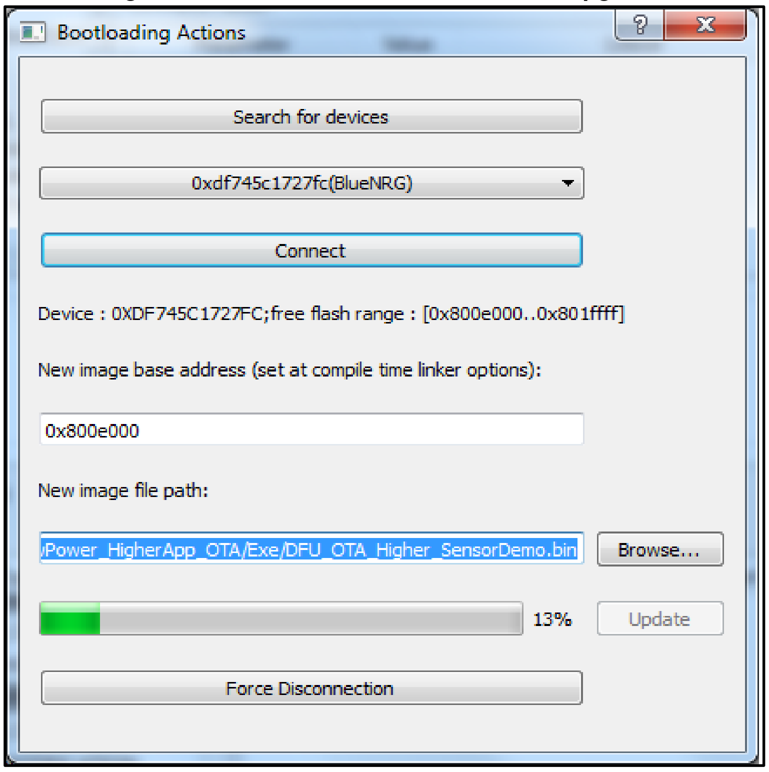
New image base address (set at compile time linker options):

New image file path:

7 Once we have browsed to the application image (SensorDemo\EWARM\LowPower_HigherApp_OTA\Exe\DFU_OTA_Higher_SensorDemo.bin or SensorDemo\EWARM_BlueNRG-MS\LowPower_HigherApp_OTA\Exe\DFU_OTA_Higher_SensorDemo.bin) built with

the proper base address (0x800e000), we can press the “Update” button to start the OTA bootloading process. A progress bar will provide an indication of the time needed for process completion.

Figure 7: BlueNRG GUI: OTA bootloader upgrade



Device : 0XDF745C1727FC;free flash range : [0x800e000..0x801ffff]

New image base address (set at compile time linker options):

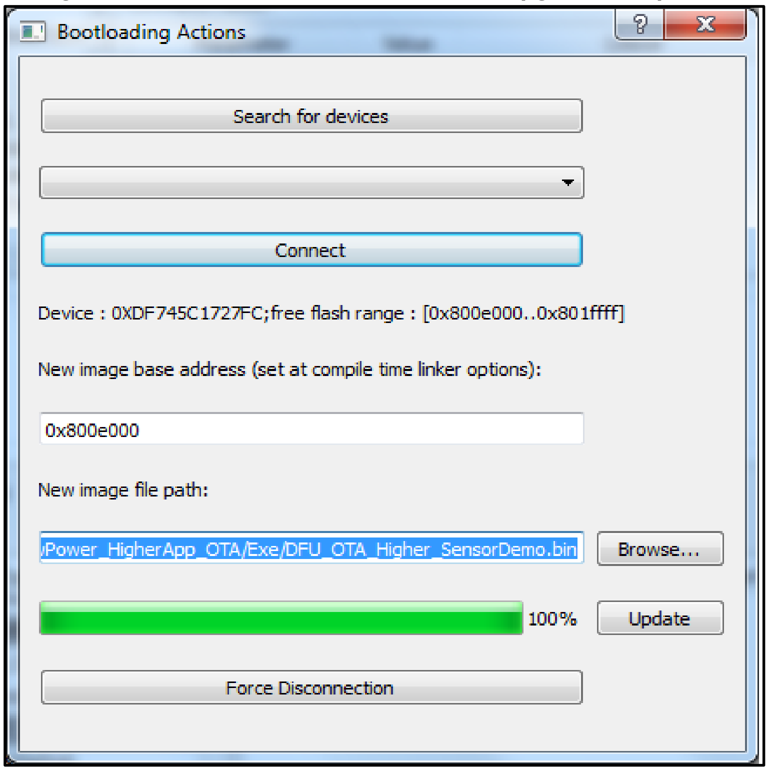
0x800e000

New image file path:

\Power_HigherApp_OTA/Exe/DFU_OTA_Higher_SensorDemo.bin

8 The process is complete when the status indicator reaches 100%, as shown below.

Figure 8: BlueNRG GUI: OTA Bootloader upgrade completed



Device : 0XDF745C1727FC;free flash range : [0x800e000..0x801ffff]

New image base address (set at compile time linker options):

0x800e000

New image file path:

/Power_HigherApp_OTA/Exe/DFU_OTA_Higher_SensorDemo.bin

100%

If we observe the slave board, we will find a change of color in the flashing LEDs notifying us that a new application is running. Tools such as the STM32 STLink utility can be used to check downloaded Flash memory regions on upgraded devices against the image within the selected binary files. After disconnection from the master, we can re-check the new version of the Sensor Demo functions against a "BlueNRG app for smartphones (IoS or Android)" as performed in step 2, optional items.

Note: A similar procedure can be used for the BlueNRG BLE_Chat demo application in order to verify the BLE OTA bootloader service functions (refer to the BLE_Chat\EWARM\BLE_Chat.eww, Server_LowerApp_OTA, Server_HigherApp_OTA workspaces for BlueNRG device and to BLE_Chat\EWARM_BlueNRG-MS\BLE_Chat.eww, Server_LowerApp_OTA, Server_HigherApp_OTA workspaces for BlueNRG-MS device).

4.2 BLE application built to use the BLE OTA Service Manager

The following is a step-by-step description of how to run the BlueNRG GUI OTA bootloader using the BlueNRG OTA Service Manager and Sensor Demo, OTA_Basic workspace:

1. Plug the two developments platforms into the USB connectors on a PC.
2. Use the BlueNRG GUI to program one of the two platforms kits to be used as OTA bootloader slave. This requires plugging the board into the USB connector on the PC running the GUI, then pressing the platform reset button while keeping the BUTTON_1 pressed in order to start the DFU bootloader (LED_D2 should be blinking at this point).

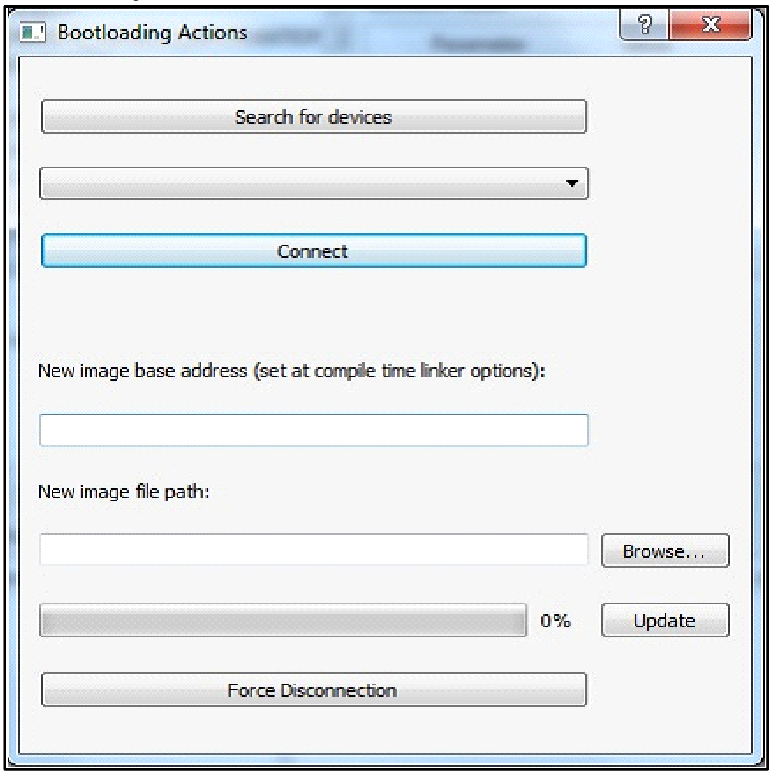
a. From Tools->Flash Motherboard FW, select the Reset Manager hex prebuilt image (STM32L\Project\OTA_prebuilt_images\DFU_ResetManager.hex) to upload the Reset Manager on board (hex format implicitly takes into account the proper application base address which, for the case of the application, will be 0x8003000).
b. After Reset Manager loading, a reset while keeping BUTTON_2 pressed needs to be performed in order to erase obsolete data from the OTA bootloader's EEPROM and guide proper Reset Manager behavior at first startup (5 seconds LED_D1 turned on signals a successful run of the process).
c. The same actions, starting from DFU bootloader activation, must be taken to upload the OTA Service Manager application built with the IAR workspace: OTA_ServiceManager\EWARM\OTA_ServiceManager.eww for BlueNRG device or OTA_ServiceManager\EWARM_BlueNRG-MS\OTA_ServiceManager.eww for BlueNRG-MS device (prebuilt images BlueNRG_OTA_ServiceManager_App.hex and BlueNRG-MS_OTA_ServiceManager_App.hex are also in OTA_prebuilt_images folder).
d. Similar actions, starting from DFU bootloader activation, must be taken to upload the SensorDemo BLE application (done through SensorDemo, OTA_Basic workspace) and built to coexist with the OTA Service Manager (hex format implicitly takes into account the proper application base address which, for the case of the application, will be 0x8006800).
e. (Optional) Get a “BlueNRG app for smartphones (IoS or Android)” to discover, connect and play with the Sensor Demo application just loaded on the development platform in step 2.d. just to verify that an ordinary BLE application has services coexisting with the OTA bootloader service, which will run next.
f. In order to activate the OTA Service Manager, the user can simply press button BUTTON_1: this invokes a specific API handling the SW jumping to the OTA Service Manager (same button pressed on OTA Service Manager will allow to jump to the current valid BLE application).

3 Program the second platform with the required application to be used with the BlueNRG GUI (BlueNRG_VCOM_x_x.hex file available on the BlueNRG DK SW package, on Firmware\STM32L1_prebuilt_images folder).

4 Open the BlueNRG GUI on the PC and select the COM port related to the BlueNRG platform configured in step 3, through the drop-down “Port” and press “Open”.

5 Select “Tools” -> “OTA bootloader” to open the dialog box containing the OTA bootloader actions and press “Search for devices”.

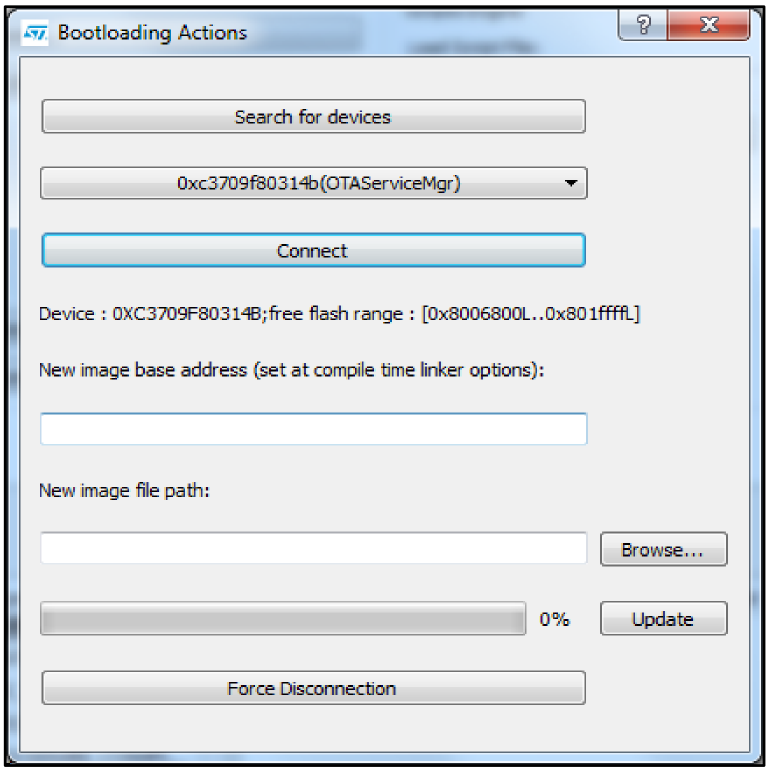
Figure 9: BlueNRG GUI: OTA bootloader window



6 After “Search for devices”, the GUI will start the discovery process and return information related to the address and application names of devices running the OTA bootloader service within the radio range. Once the previous process ends, the device list can be opened through the combo box arrow below the “Search for devices” button and the user can find the device running the OTA Service Manager and press “Connect”. If we realize we have connected the wrong device, we can just press the “Force Disconnection” button and return to device selection within the combo box. After device selection and connection through the “Connect” button, we can read the related free memory range that it advertises, which constrains us to provide an image file compiled with a base address and size within the expected range on the BLE device.

a. As the next step, we need to provide the “Base address: 0x8006800” and press the browse button to select the location of the file containing the image that we plan to send over-the-air towards the BLE device. For that purpose, the user can open the BlueNRG IAR workspace SensorDemo, OTA_Basic and build the related SensorDemo_OTA_Basic.bin image (available in SensorDemo\EWARM\OTA_Basic\Exe folder) containing a version of the Sensor Demo that is compiled with base address equal to 0x8006800. Alternatively, the user can open the BlueNRG IAR workspace BLE_Chat, Server_OTA_Basic and build the related Server_OTA_Basic.bin image (available in BLE_Chat\EWARM\Server_OTA_Basic\Exe folder) containing a version of the BLE_Chat, server application that is compiled with base address equal to 0x8006800. Similar steps must be followed when using a BlueNRG-MS device: user can open the workspace SensorDemo, OTA_Basic and build the related SensorDemo_OTA_Basic.bin image (available in SensorDemo\EWARM_BlueNRG-MS\OTA_Basic\Exe folder) containing a version of the Sensor Demo that is compiled with base address equal to 0x8006800. Alternatively, the user can open the IAR workspace BLE_Chat, Server_OTA_Basic and build the related Server_OTA_Basic.bin image (available in BLE_Chat\EWARM_BlueNRG-MS\Server_OTA_Basic\Exe folder) containing a version of the BLE_Chat, server application that is compiled with base address equal to 0x8006800.

Figure 10: BlueNRG GUI: OTA bootloader search for devices and connect



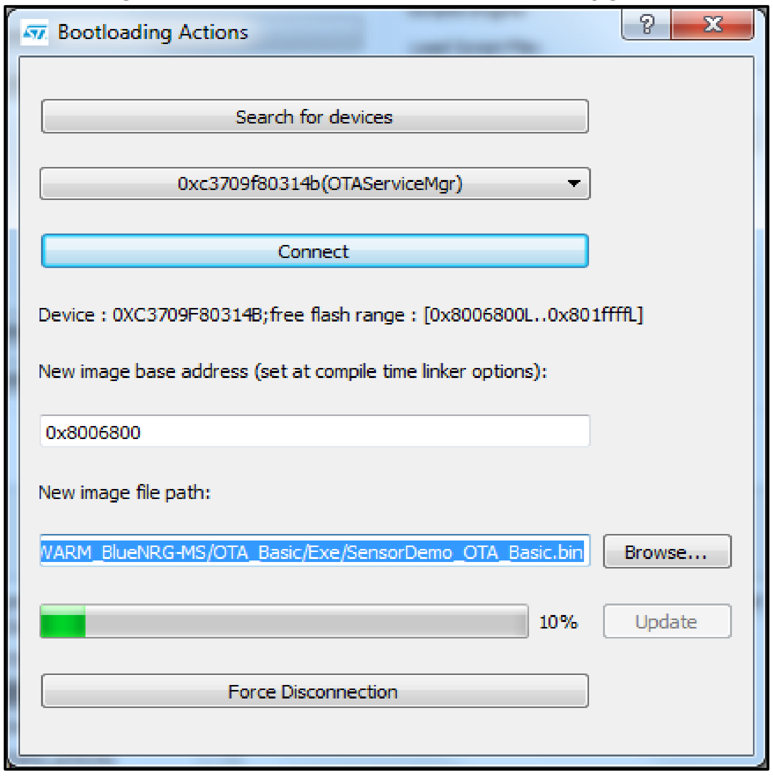
Device : 0XC3709F80314B;free flash range : [0x8006800L..0x801ffffL]

New image base address (set at compile time linker options):

New image file path:

7 Once we have browsed to the application image built with the proper base address (0x8006800), we can press the “Update” button to start the OTA bootloading process. A progress bar will provide an indication of the time needed until process completion:

Figure 11: BlueNRG GUI: OTA bootloader upgrade



Device : 0XC3709F80314B;free flash range : [0x8006800L..0x801ffffL]

New image base address (set at compile time linker options):

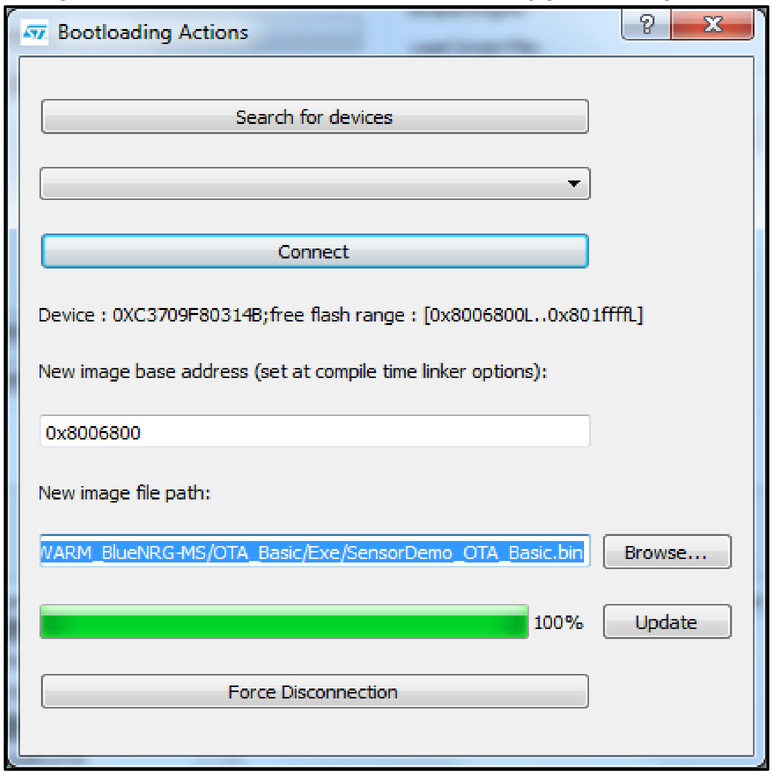
0x8006800

New image file path:

\ARM_BlueNRG-MS/OTA_Basic/Exe/SensorDemo_OTA_Basic.bin

8 The process is complete when the status indicator reaches 100%, as shown in *Figure 12: "BlueNRG GUI: OTA bootloader upgrade completed"*

Figure 12: BlueNRG GUI: OTA bootloader upgrade completed



Device : 0XC3709F80314B;free flash range : [0x8006800L..0x801ffffL]

New image base address (set at compile time linker options):

0x8006800

New image file path:

/ARM_BlueNRG-MS/OTA_Basic/Exe/SensorDemo_OTA_Basic.bin

100%

If we observe the BLE platform, we will find a change of color in the flashing LEDs notifying us that a new application is running.

5 List of acronyms

Table 9: Acronyms used in this document

Acronym	Meaning
BLE	Bluetooth low energy
BTL	Bootloader
IFR	Information register
OTA	Over-the-air
USB	Universal serial bus

6 Related documentation and references

Table 10: Related documentation

Term	Description
BlueNRG, BlueNRG-MS DK	BlueNRG development kit software package
UM1686	BlueNRG development kits user manual

7 Revision history

Table 11: Document revision history

Date	Revision	Changes
30-May-2014	1	Initial release.
22-Apr-2015	2	Document adapted to refer to both the BlueNRG and BlueNRG-MS devices.