English | 中文

Table of Contents

1. Overview

This tutorial explains how to perform a Device Firmware Upgrade (DFU) with Bluetooth Over-The-Air (OTA) update. Any chip that has OTA-updates enabled in their GATT profile can have an OTA upgrade. Most of the example applications provided in the Bluetooth SDK already have OTA support built into the code. In these examples, the DFU mode is triggered through the Silicon Labs OTA service that is included as part of the application's GATT database. OTA functionality can be added by installing the OTA DFU software component in your project.

2. Bootloader Introduction

The Silicon Labs Gecko Bootloader is a common bootloader for all the newer MCUs and wireless MCUs from Silicon Labs. The Gecko Bootloader can be configured to perform a variety of bootload functions, from device initialization to firmware upgrades. The Gecko Bootloader uses a proprietary format for its upgrade images, called GBL (Gecko Bootloader file). These images are produced with the file extension ".gbl". Additional information on the GBL file format is provided in UG103.6: Bootloader Fundamentals.

The Gecko Bootloader can be configured to perform firmware upgrades in standalone mode (also called a standalone bootloader) or in application mode (also called an application bootloader), depending on the component/plugin configuration. Components (in GSDK 4.0 and higher) and plugins (in older GSDKs) can be enabled and configured through the Simplicity Studio IDE.

If you need to get more detailed information on Silicon Labs Gecko Bootloader and UART/OTA DFU with Silicon Labs Bluetooth application, please refer to an1086-gecko-bootloader-bluetooth.pdf.

3. Bluetooth OTA Upgrade

This chapter will introduce how to implement device firmware update functionality by AppLoader (In-Place OTA) and the User Application.

3.1. Implementing Device Firmware Update with AppLoader(In-Place OTA)

A Bluetooth application developed with Silicon Labs Bluetooth SDK comprises two parts: AppLoader and the user application. AppLoader is a small standalone application that is required to support in-place OTA updates. AppLoader can run independently of the user application. It contains a minimal version of the Bluetooth stack, including only those features that are necessary to perform the OTA update. Any Bluetooth features that are not necessary to support OTA updates are disabled in AppLoader to minimize the flash footprint.

3.1.1. AppLoader Introduction

The AppLoader features and limitations are summarized below:

• Enables OTA updating of user application.

• The AppLoader itself can also be updated.

• Only one Bluetooth connection is supported, GATT server role only.

• Encryption and other security features such as bonding are not supported.

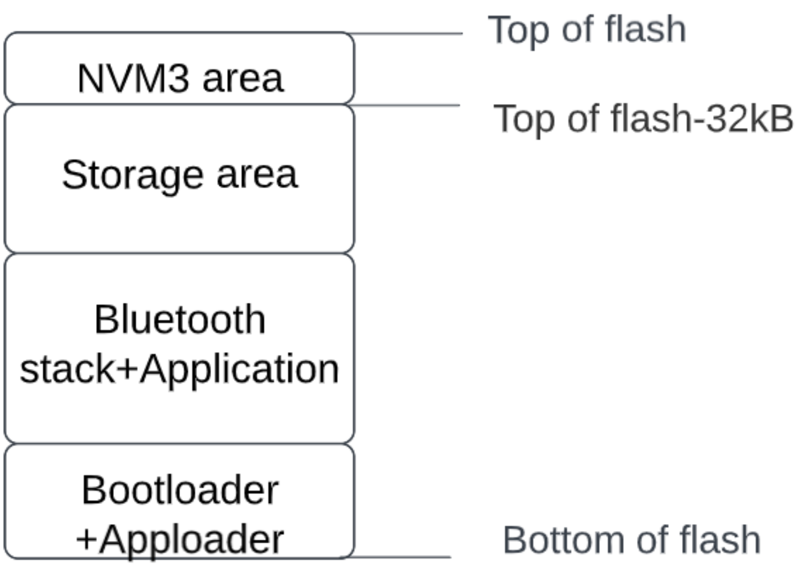
• PTI is not enabled so it is not possible to use the Network Analyzer with the AppLoader.

Note: AppLoader in SDK v3.x or higher requires that the Gecko Bootloader version must be v1.11 or later to support OTA.

The user application is placed in code flash after AppLoader. The default linker script provided in the SDK places the user application so that it starts at the next flash sector following AppLoader. The user application contains a full-featured version of the Bluetooth stack and it can run independently of the AppLoader. If in-place OTA update does not need to be supported then the AppLoader can be removed completely to free up flash for other use (code space or data storage).

3.1.2. Gecko Bootloader Configuration from SDK version 4.0

From SDK version v4.0, the Apploader can also be included in the Bootloader. The following picture shows an example of this new layout:



To use the Apploader from the Bootloader, the start address of the application needs to be changed so there will be enough space for the Bootloader and the Apploader before the application. This is not possible in Series 1 devices, so this new method can only be used for Series 2 devices. For Series 2 devices, this is the new default configuration.

3.1.3. In-Place OTA Process

Most of the OTA functionality is handled autonomously by the AppLoader, which greatly simplifies application development. The minimum requirement for the user application is for a way to trigger a reboot into DFU mode. Rebooting into DFU mode in this context means that after the device is reset, the AppLoader is run instead of the user application. After the upload is complete, AppLoader will reboot the device back into normal mode.

Reboot into DFU mode can be triggered in a variety of ways. It is up to the application developer to decide which is most applicable.The following sections explain in detail how this is done in the user application.

AppLoader supports two types of update:

• Full update: both Bootloader and the user application are updated

• Partial update: only the user application is updated

A full update is always recommended when moving from one SDK version to another. The size of AppLoader can vary depending on the SDK version. This may prevent a partial OTA update if the new application image overlaps with the old AppLoader version.

3.1.3.1. Basic Steps to Partial Update Firmware from the AppLoader

With partial update, it is possible to update the Bluetooth stack and user application. BootLoader is not modified during partial update. The partial update process using AppLoader consists of following steps:

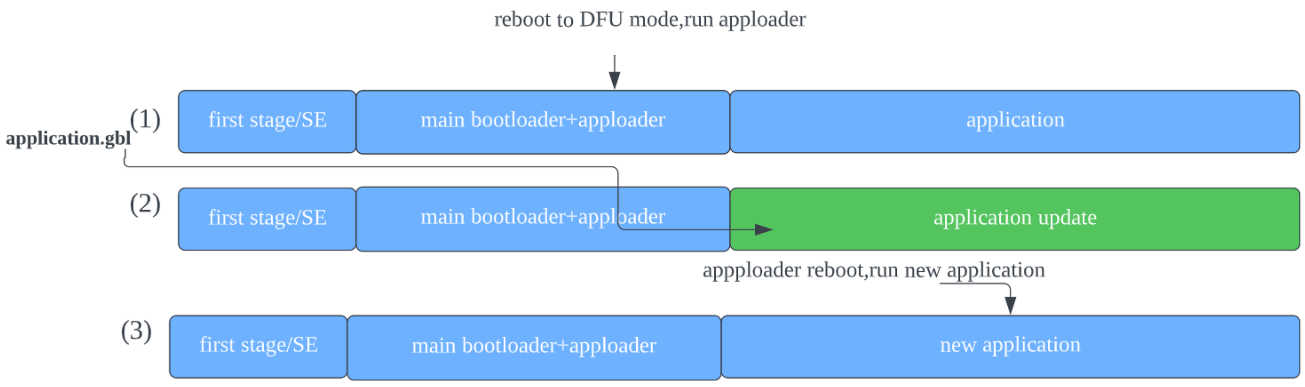
1. • OTA client connects to target device.

   • Client requests target device to reboot into DFU mode.
2. • After reboot, client connects again. During the 2nd connection, target device is running AppLoader which is contained in bootloader image file(not the user application).

   • New firmware image (application.gbl) is uploaded to the target.

   • AppLoader copies the new application on top of the existing application.
3. • When upload is finished and connection closed, AppLoader reboots back to normal mode.

   • Update complete.


reboot to DFU mode,run apploader
(1)
application.gbl
first stage/SE
main bootloader+apploader
application
(2)
first stage/SE
main bootloader+apploader
application update
appploader reboot,run new application
(3)
first stage/SE
main bootloader+apploader
new application

3.1.3.2. Basic Steps to Full Update Firmware from the AppLoader

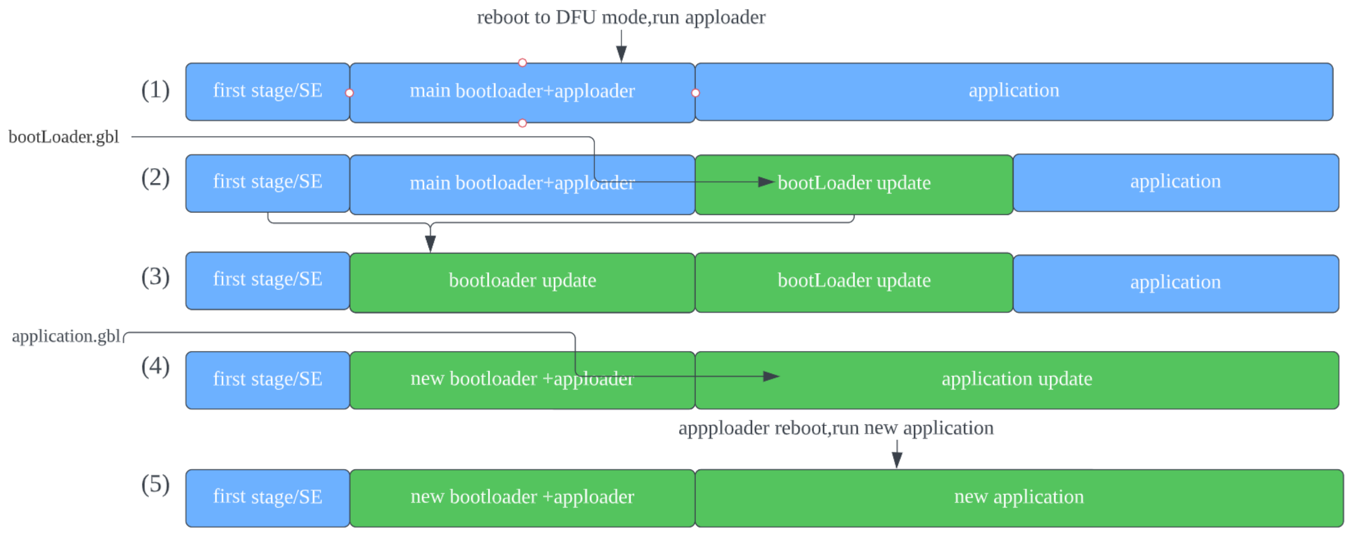
Full update enables updating both the bootloader which contain AppLoader and the user application. Full update is done in two steps. Updating the bootloader always erases the user application and therefore bootloader update must always be followed by application update. The first phase of full update updates the bootloader which contain AppLoader and it consists of following steps:

1. • OTA client connects to target device.

   • Client requests target device to reboot into DFU mode.
2. • After reboot, client connects again. During the 2nd connection, target device is running AppLoader in bootloader image file(not the user application).

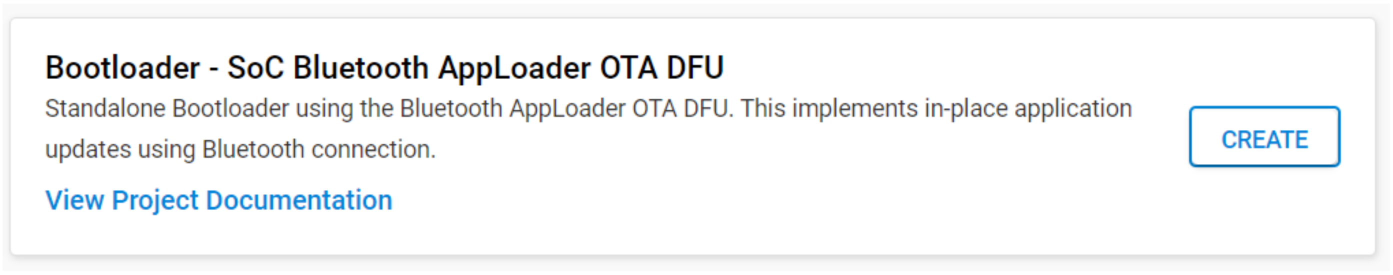
   • New bootloader image (bootloader.gbl) is uploaded to the target. AppLoader copies the image into the download area (specified in Gecko bootloader configuration).
3. • When upload is finished and connection closed, AppLoader reboots and requests the first stage/Secure Element to install the downloaded image.

   • The first stage/Secure Element will overwrite the main bootloader with the new image. After reboot, the new bootloader which contain new apploader is started.



At the end of the bootLoader update, the device does not contain a valid user application and therefore appLoader will remain in DFU mode. To complete the update, a new user application is uploaded following the same sequence of operations that were described for the partial update.

It is recommended to use Gecko SDK 4.1.0 "SoC Bluetooth AppLoader OTA DFU" project that appLoader has been added by default. Create the example bootloader project below,and the component of Bluetooth AppLoader OTA DFU has been added.



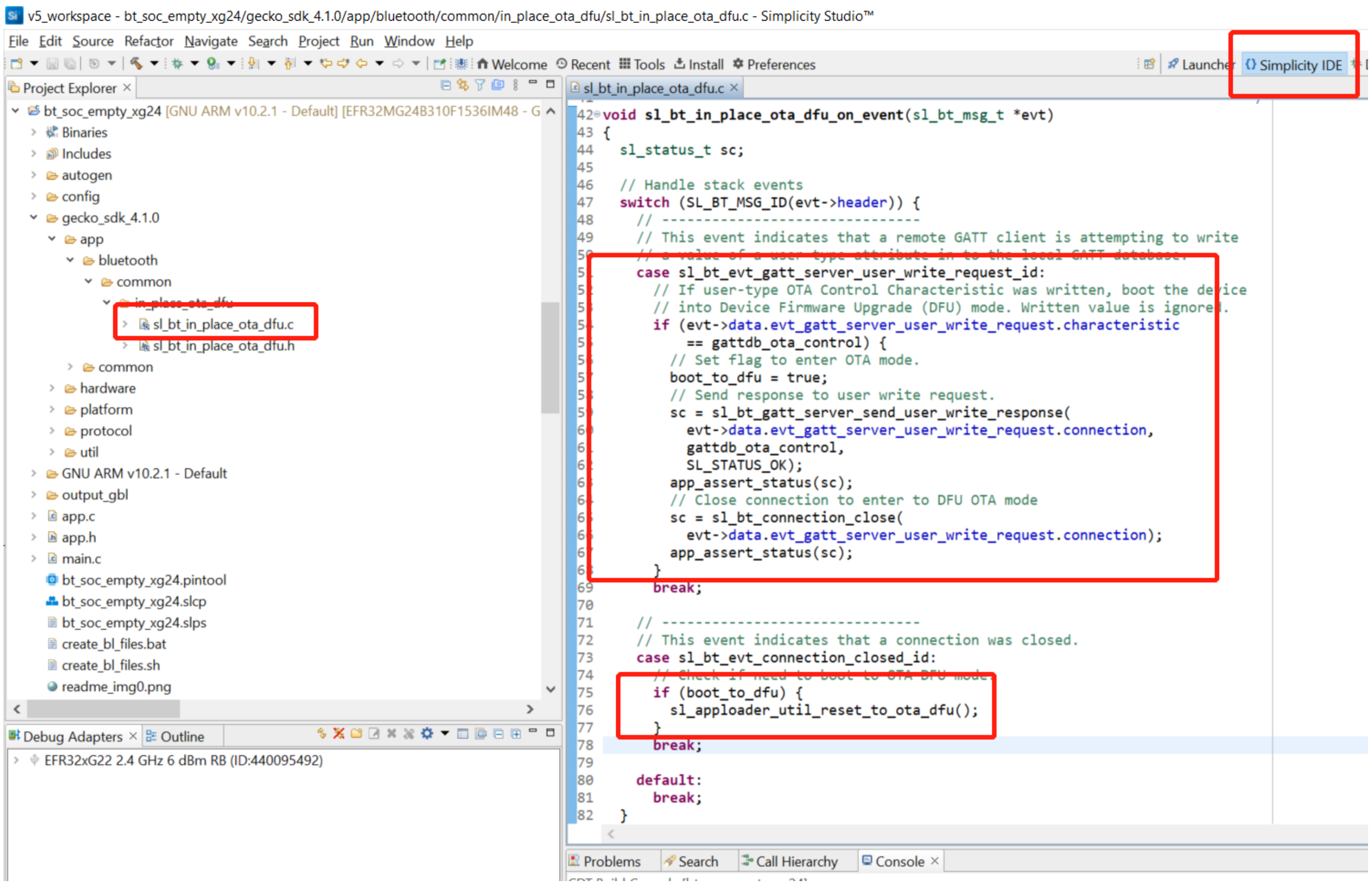
Bootloader - SoC Bluetooth AppLoader OTA DFU

Standalone Bootloader using the Bluetooth AppLoader OTA DFU. This implements in-place application updates using Bluetooth connection.

View Project Documentation

3.1.3.3. Triggering Reboot into DFU Mode from the User Application

The minimum functional requirement to enable OTA in the user application is to implement a "hook" that allows the device to be rebooted into DFU mode. By default, this is done through the Silicon Labs OTA service. The following code snippet is from the SoC empty example supplied with the SDK. The code to enter DFU mode is similar in the other examples.



```
void sl_bt_in_place_ota_dfu_on_event(sl_bt_msg_t *evt)
{
  sl_status_t sc;

  // Handle stack events
  switch (SL_BT_MSG_ID(evt->header)) {
    // -------------------------------
    // This event indicates that a remote GATT client is attempting to write
    // a value of a user type attribute in to the local GATT database.
    case sl_bt_evt_gatt_server_user_write_request_id:
      // If user-type OTA Control Characteristic was written, boot the device
      // into Device Firmware Upgrade (DFU) mode. Written value is ignored.
      if (evt->data.evt_gatt_server_user_write_request.characteristic
          == gattdb_ota_control) {
        // Set flag to enter OTA mode.
        boot_to_dfu = true;
        // Send response to user write request.
        sc = sl_bt_gatt_server_send_user_write_response(
          evt->data.evt_gatt_server_user_write_request.connection,
          gattdb_ota_control,
          SL_STATUS_OK);
        app_assert_status(sc);
        // Close connection to enter to DFU OTA mode
        sc = sl_bt_connection_close(
          evt->data.evt_gatt_server_user_write_request.connection);
        app_assert_status(sc);
      }
      break;

    // -------------------------------
    // This event indicates that a connection was closed.
    case sl_bt_evt_connection_closed_id:
      // Check if need to boot to OTA DFU mode.
      if (boot_to_dfu) {
        sl_apploader_util_reset_to_ota_dfu();
      }
      break;

    default:
      break;
  }
```

v5_workspace - bt_soc_empty_xg24/gecko_sdk_4.1.0/app/bluetooth/common/in_place_ota_dfu/sl_bt_in_place_ota_dfu.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Project Explorer
bt_soc_empty_xg24 [GNU ARM v10.2.1 - Default] [EFR32MG24B310F1536IM48 - G
Binaries
Includes
autogen
config
gecko_sdk_4.1.0
app
bluetooth
common
in_place_ota_dfu
sl_bt_in_place_ota_dfu.c
sl_bt_in_place_ota_dfu.h
common
hardware
platform
protocol
util
GNU ARM v10.2.1 - Default
output_gbl
app.c
app.h
main.c
bt_soc_empty_xg24.pintool
bt_soc_empty_xg24.slcp
bt_soc_empty_xg24.slps
create_bl_files.bat
create_bl_files.sh
readme_img0.png
Debug Adapters
Outline
EFR32xG22 2.4 GHz 6 dBm RB (ID:440095492)
sl_bt_in_place_ota_dfu.c
Problems
Search
Call Hierarchy
Console

3.2. Implementing Device Firmware Update in the User Application

It is possible to implement the firmware update functionality completely in the user application. This makes it possible to use a custom GATT service instead of the Silicon Labs OTA service. To use this update mechanism, any application bootloader configuration may be used, using internal or external storage. At least one download area must be defined and the area must be large enough to fit the full GBL file. The download area must not overlap with the user application.

3.2.1. Basic Steps to Update Firmware from the User Application

The general firmware upgrade sequence is explained in UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher and UG266: Silicon Labs Gecko Bootloader User's Guide for GSDK 3.2 and Lower.

The basic steps are summarized below.

1. • Application initializes the Gecko bootloader by calling `bootloader_init()`.

   • Get bootlader information by `bootloader_getInfo`.Storage slot information can be got by `bootloader_getStorageSlotInfo`.

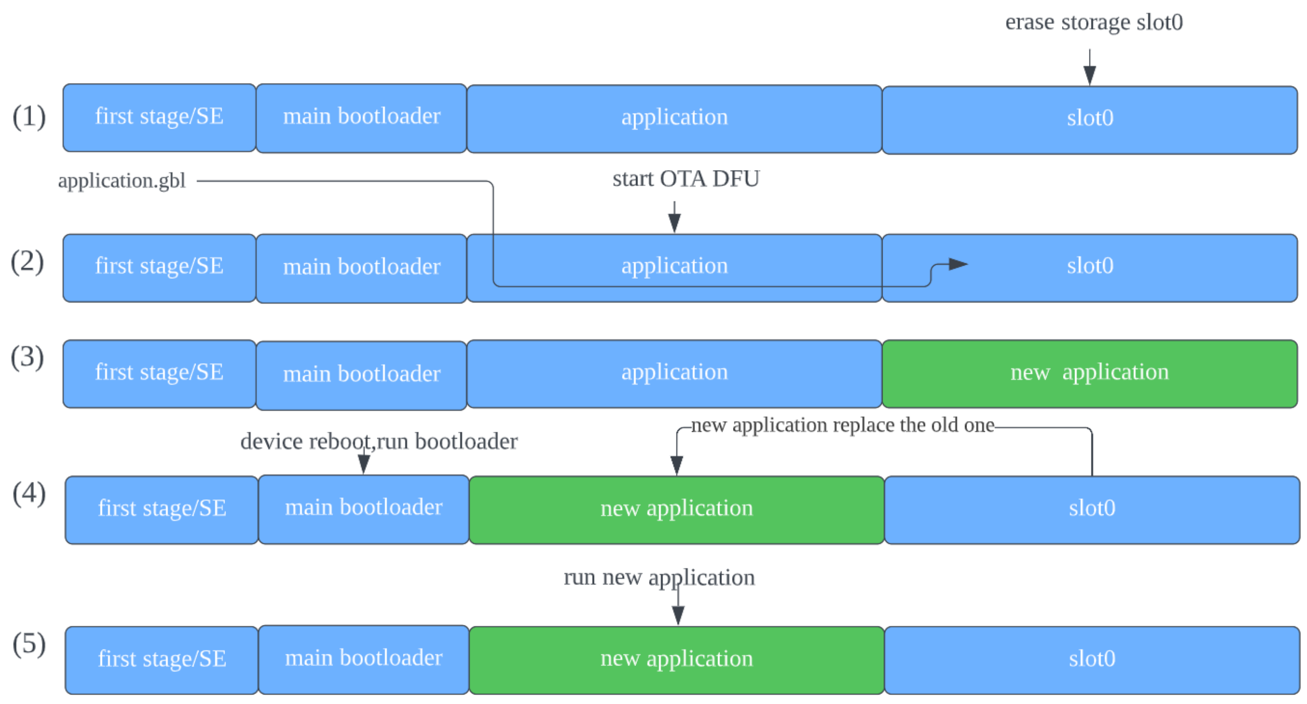
   • Run through the full storage and read the context in chunks by `calling bootloader_readStorage`.

   • The download area is erased by calling `bootloader_eraseStorageSlot(0)`.
2. • OTA client connects to target device.Client requests target device into DFU mode.

• The update image (full GBL file) is received over-the-air, application writes the received bytes to the download area by calling `bootloader_writeStorage()`.

3. • (optional) Application can verify the integrity of the received GBL file by calling `bootloader_verifyImage()`.
4. • Before rebooting, call `bootloader_setAppImageToBootload(0)` to specify the slot ID where new image is stored.

   • Reboot and instruct Gecko bootloader to perform the update by calling `bootloader_rebootAndInstall()`.
5. • Update complete,run new application.



It is assumed here that only one download area is configured and therefore the slot index in the above function calls is set to 0.Note that the erase procedure in step (2) above takes several seconds to complete. If the new image is downloaded over a Bluetooth connection then the supervision timeout must be set long enough to avoid connection drops. Alternatively, the download area can be erased in advance, before the Bluetooth connection is opened. A third alternative is to erase the download area one flash page at a time while the writing progresses. This can be done using `bootloader_eraseRawStorage()`.

3.2.2. Enabling Gecko Bootloader API

Gecko bootloader has an application interface exposed through a function table in the bootloader. To be able to call Gecko bootloader functions from your Bluetooth application, the following source files must be added into the project:

btl_interface.c (common interface)

btl_interface_storage.c (interface to storage functionality)

These files are found in the Gecko SDK suite in the following directory (exact path depends on installed SDK version):

`\gecko_sdk_suite\<version>\platform\bootloader\api\`

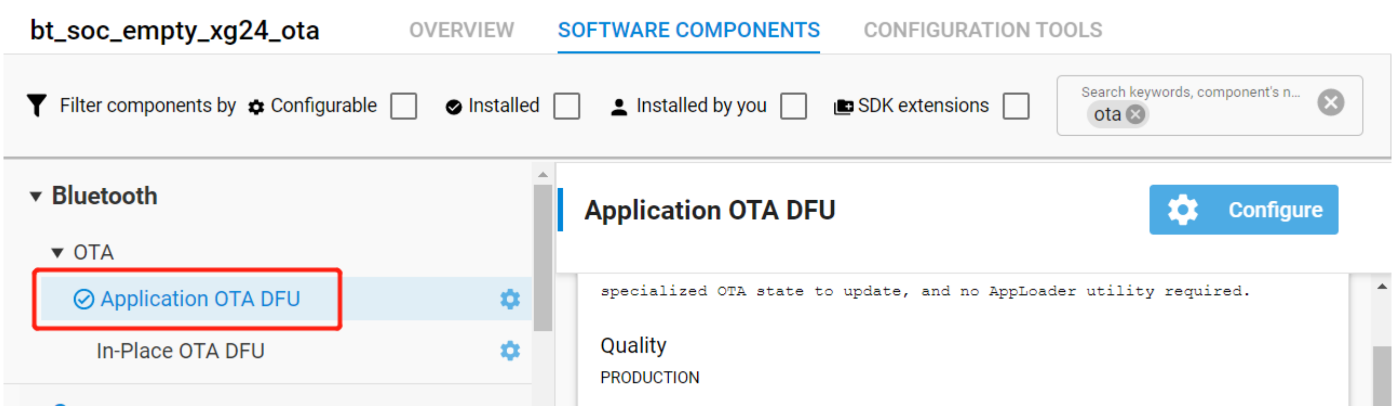
Starting with SDK v3.0, these files are copied to the sample projects by default but the corresponding include files must be added to the source file(s) that call any Gecko bootloader functions:

`#include "btl_interface.h"`

`#include "btl_interface_storage.h"`

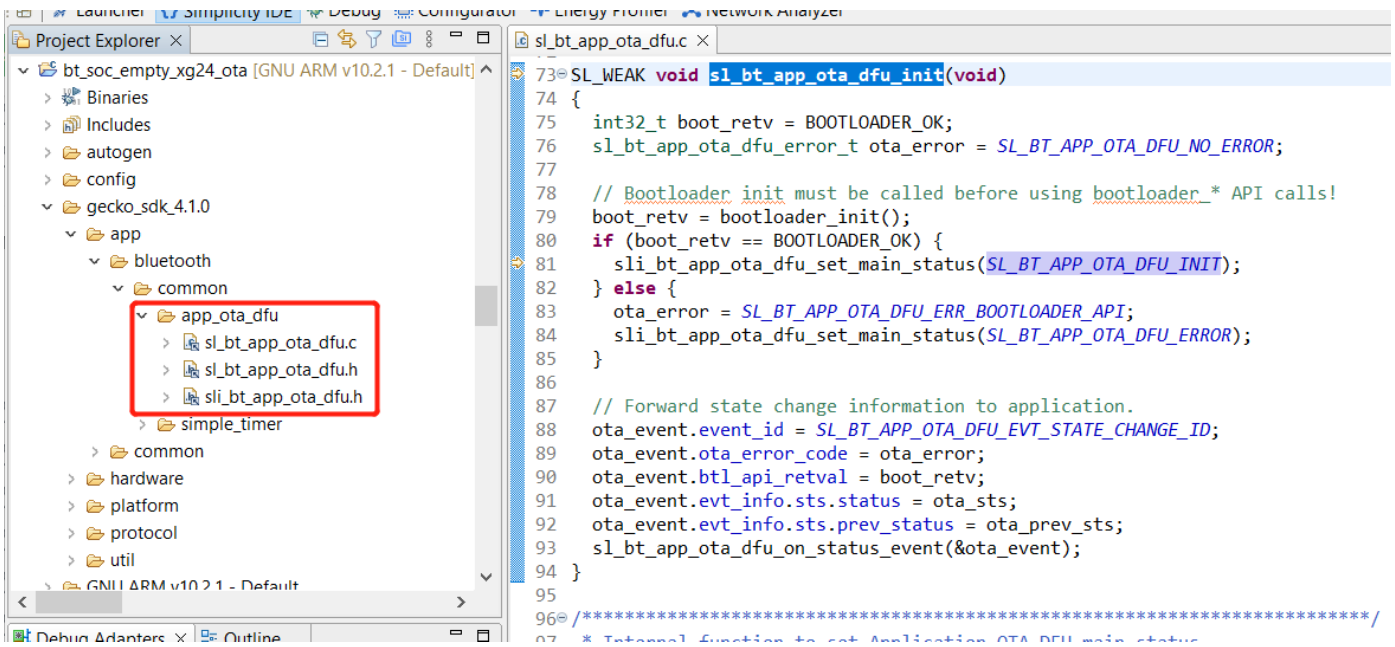
3.2.3. Example Implementation of Bluetooth OTA Update under Application Control

An example implementation of application-controlled OTA update is provided as a Software Component. To use it, simply add the "Application OTA DFU" component to the project:



specialized OTA state to update, and no AppLoader utility required.

After add "Application OTA DFU" component,"app_ota_dfu" folder will be automatically generated.The code of "app_ota_dfu" folder follow the section 3.1.3.2 to implement device firmware update in the user application.The gatt_configuration.btconf file will automatically add "Silicon Labs OTA Data Characteristics" in the project tree.

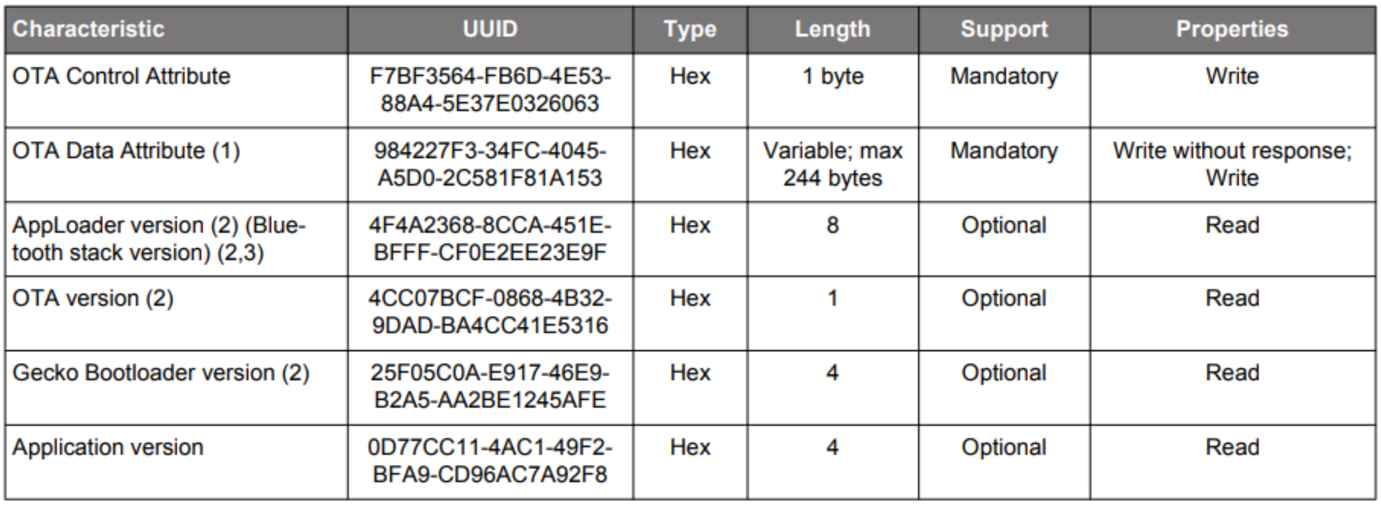


```
SL_WEAK void sl_bt_app_ota_dfu_init(void)
{
  int32_t boot_retv = BOOTLOADER_OK;
  sl_bt_app_ota_dfu_error_t ota_error = SL_BT_APP_OTA_DFU_NO_ERROR;

  // Bootloader init must be called before using bootloader_* API calls!
  boot_retv = bootloader_init();
  if (boot_retv == BOOTLOADER_OK) {
    sli_bt_app_ota_dfu_set_main_status(SL_BT_APP_OTA_DFU_INIT);
  } else {
    ota_error = SL_BT_APP_OTA_DFU_ERR_BOOTLOADER_API;
    sli_bt_app_ota_dfu_set_main_status(SL_BT_APP_OTA_DFU_ERROR);
  }

  // Forward state change information to application.
  ota_event.event_id = SL_BT_APP_OTA_DFU_EVT_STATE_CHANGE_ID;
  ota_event.ota_error_code = ota_error;
  ota_event.btl_api_retval = boot_retv;
  ota_event.evt_info.sts.status = ota_sts;
  ota_event.evt_info.sts.prev_status = ota_prev_sts;
  sl_bt_app_ota_dfu_on_status_event(&ota_event);
}

/***************************************************************************//**
 * Internal function to set Application OTA DFU main status
```

3.3. Silicon Labs OTA GATT Service

The XML representation in the project defines the Silicon Labs OTA service. It is a custom service using 128-bit UUID values. The service content and the UUID values are fixed and must not be changed.

The OTA service characteristics are described in the following table. The UUID value of the service itself is `1d14d6eefd63-4fa1-bfa4-8f47b42119f0`.

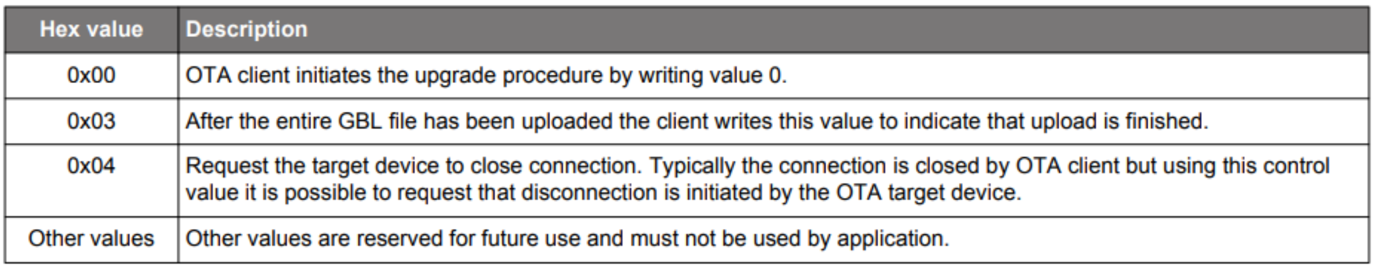


Characteristic	UUID	Type	Length	Support	Properties
OTA Control Attribute	F7BF3564-FB6D-4E53-88A4-5E37E0326063	Hex	1 byte	Mandatory	Write
OTA Data Attribute (1)	984227F3-34FC-4045-A5D0-2C581F81A153	Hex	Variable; max 244 bytes	Mandatory	Write without response; Write
AppLoader version (2) (Bluetooth stack version) (2,3)	4F4A2368-8CCA-451E-BFFF-CF0E2EE23E9F	Hex	8	Optional	Read
OTA version (2)	4CC07BCF-0868-4B32-9DAD-BA4CC41E5316	Hex	1	Optional	Read
Gecko Bootloader version (2)	25F05C0A-E917-46E9-B2A5-AA2BE1245AFE	Hex	4	Optional	Read
Application version	0D77CC11-4AC1-49F2-BFA9-CD96AC7A92F8	Hex	4	Optional	Read

Figure 3-3-1 Silicon Labs OTA Service Characteristics

Notes:

- (1) This characteristic is excluded from the user application GATT database.
- (2) Version information is automatically added by AppLoader when running in DFU mode. These are optional in the application GATT database.
- (3) This characteristic exposes AppLoader version starting from SDK 2.7.0; was stack version in earlier versions.



Hex value	Description
0x00	OTA client initiates the upgrade procedure by writing value 0.
0x03	After the entire GBL file has been uploaded the client writes this value to indicate that upload is finished.
0x04	Request the target device to close connection. Typically the connection is closed by OTA client but using this control value it is possible to request that disconnection is initiated by the OTA target device.
Other values	Other values are reserved for future use and must not be used by application.

Figure 3-3-2 Possible Control Words Written to the OTA Control Characteristic

In DFU mode, AppLoader uses the full OTA service described above. This allows a remote Bluetooth device to upload a new firmware image, as described later in this chapter. The GATT database of the user application includes only a subset of the full OTA service. The application must not include the OTA data characteristic in its GATT database.

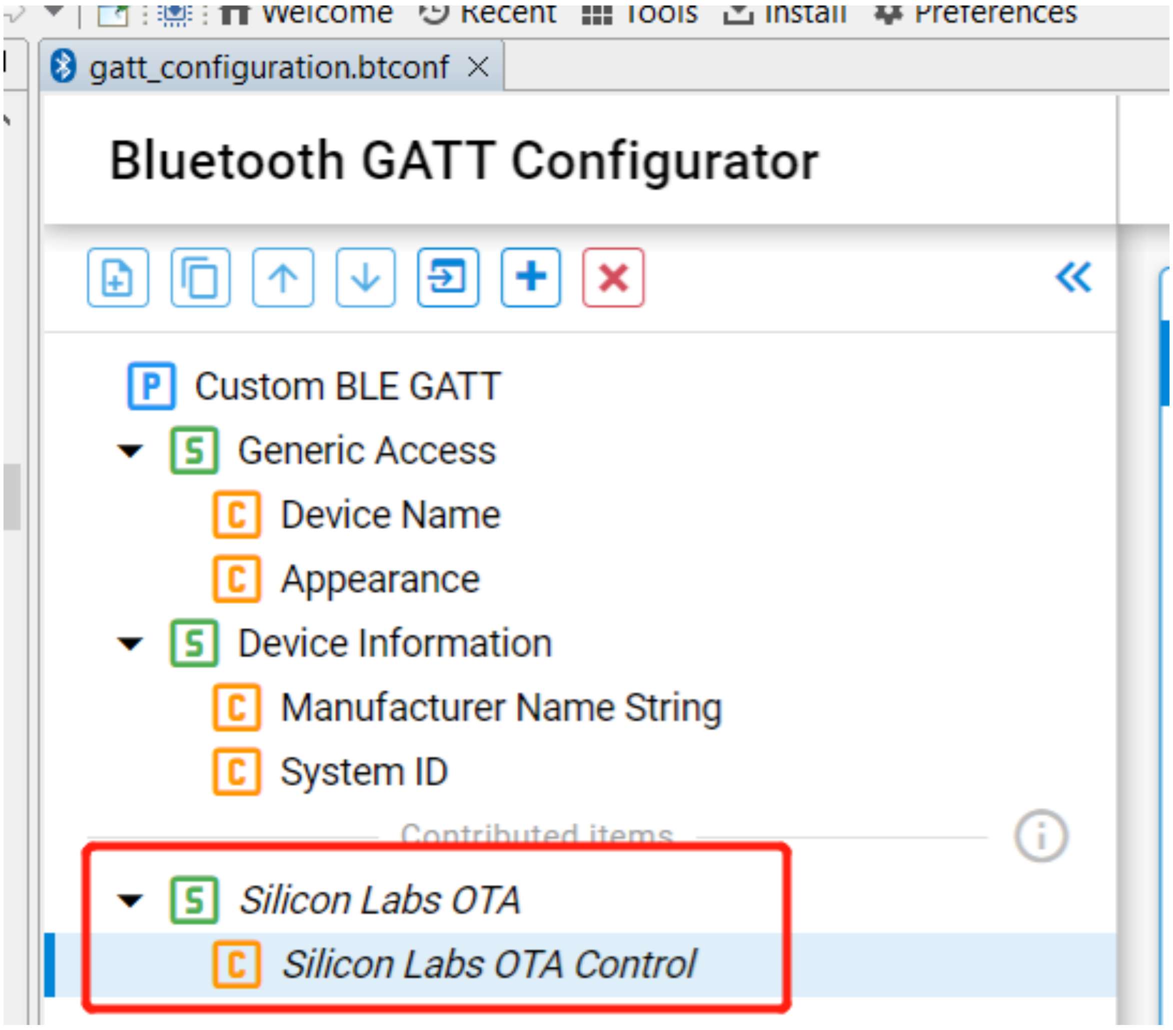
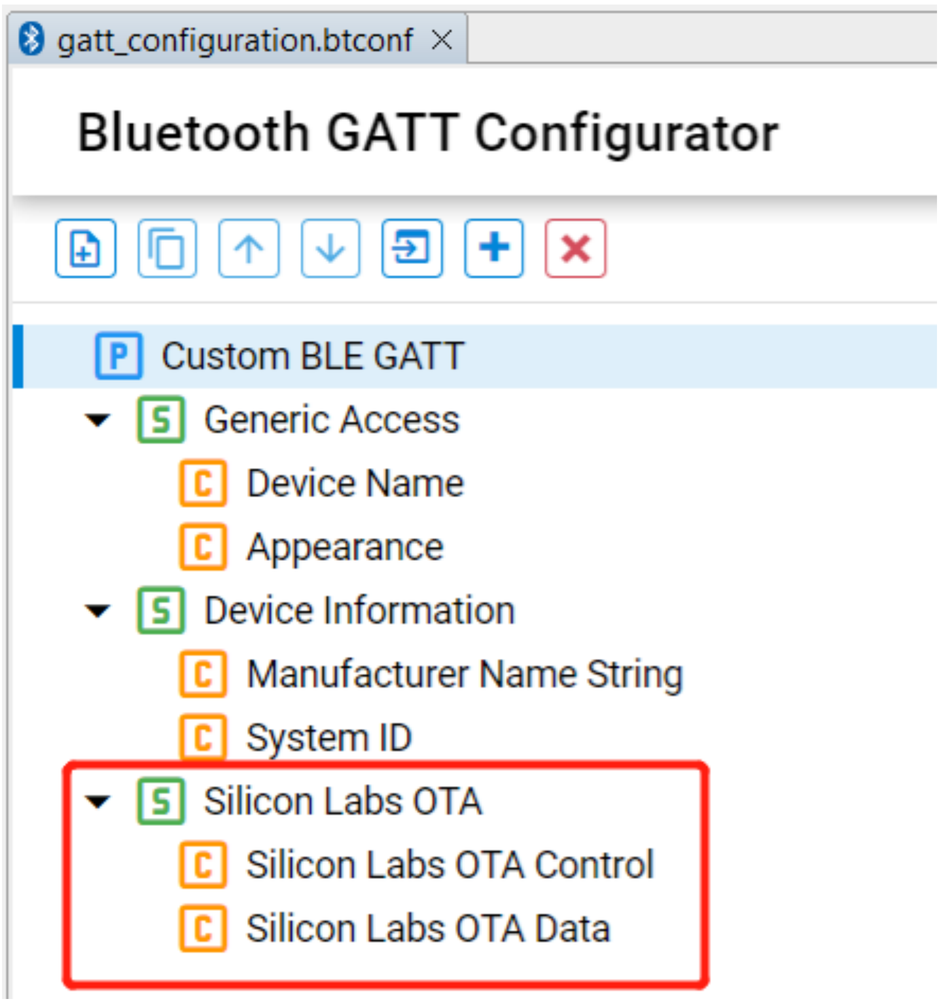


Figure 3-3-3 The OTA Control Characteristic in the Application

If the OTA update is implemented fully in application code, as described in section 3.2. Implementing Device Firmware Update in the User Application. The minimum application requirement is to include the OTA control and the OTA Data characteristic. For more detailed OTA service information, please refer to section 4.4. gatt-configuration.btconf file.

From the user application viewpoint, only the OTA control attribute is relevant. In the OTA host example reference implementation that is included in the SDK, the OTA procedure is triggered when the client writes value 0 to the OTA control attribute. The user application does not handle any data transfers related to OTA upgrades and therefore the OTA Data Attribute is excluded from the user application’s GATT.


gatt_configuration.btconf

Bluetooth GATT Configurator

- Custom BLE GATT
  - Generic Access
    - Device Name
    - Appearance
  - Device Information
    - Manufacturer Name String
    - System ID
  - Silicon Labs OTA
    - Silicon Labs OTA Control
    - Silicon Labs OTA Data

Figure 3-3-4 The OTA Control Characteristic and the OTA Data characteristic in the Application

It is also possible to use an application-specific trigger to enter OTA mode, and therefore it is not absolutely necessary to include the OTA control attribute in the application's GATT database. If reboot into DFU mode is handled using some other mechanism, then it is possible to exclude the whole OTA service from the application GATT. However, it should be noted that to be compatible with the OTA host example from the SDK or the EFR Connect smartphone app the OTA trigger must be implemented as described above.

4. Lab

4.1. Lab Introduction

For tutorial purposes, the Bluetooth - SoC Empty SDK example application will be upgraded to the Bluetooth - SoC Thermometer example application in section 4.3 In-Place OTA Lab Step. In section 4.4 User Application DFU Lab Steps,the Bluetooth - SoC Empty SDK example application will be changed device name as "OTA test". So users can easily check that the functionality of the user application has changed.

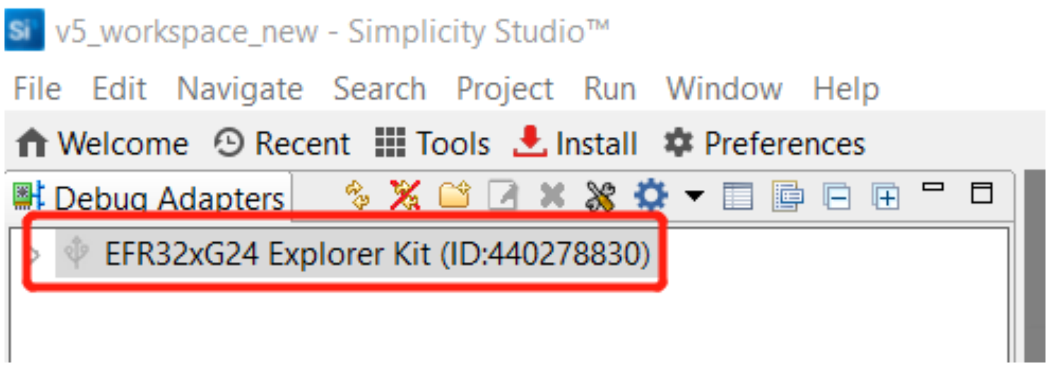
4.2. Lab Requirements

- Bluetooth-capable radio board（EFR32MG24-BRD2703A)
- Simplicity Studio 5
- Android or iOS mobile device

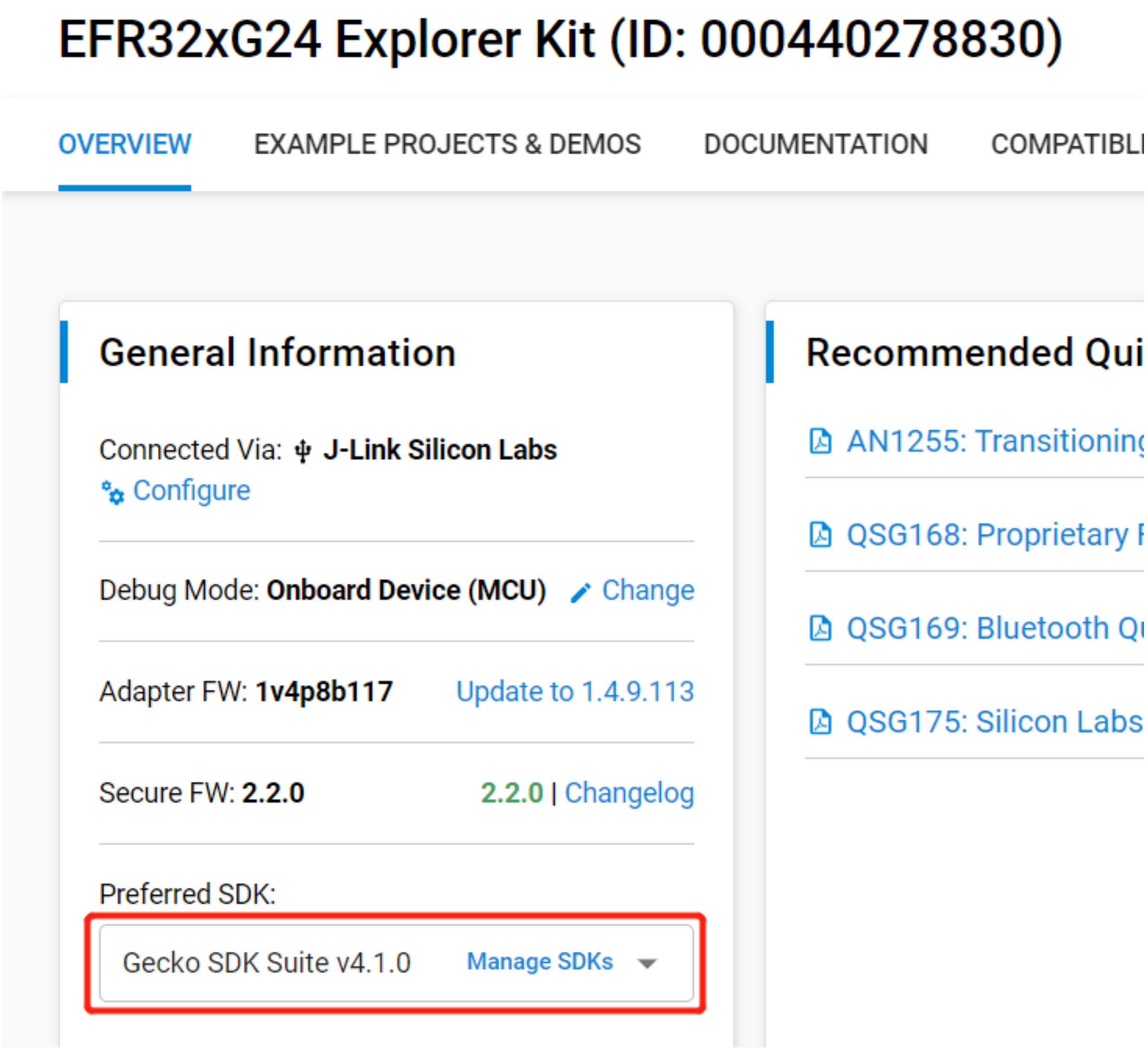
- Download EFR Connect mobile app, version 2.4 or newer(Android / iOS),and the source is available on GitHub

4.3. In-Place OTA Lab Steps

1. Connect the kit to your computer and select it in Simplicity Studio.


v5_workspace_new - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Debug Adapters
EFR32xG24 Explorer Kit (ID:440278830)

2. Select Gecko SDK Suite v4.1.0 in Preferred SDK.



EFR32xG24 Explorer Kit (ID: 000440278830)

General Information

Connected Via: J-Link Silicon Labs
Configure

Debug Mode: Onboard Device (MCU) Change

Adapter FW: 1v4p8b117 Update to 1.4.9.113

Secure FW: 2.2.0 2.2.0 | Changelog

Preferred SDK:
Gecko SDK Suite v4.1.0 Manage SDKs

Recommended Qui

AN1255: Transitioning

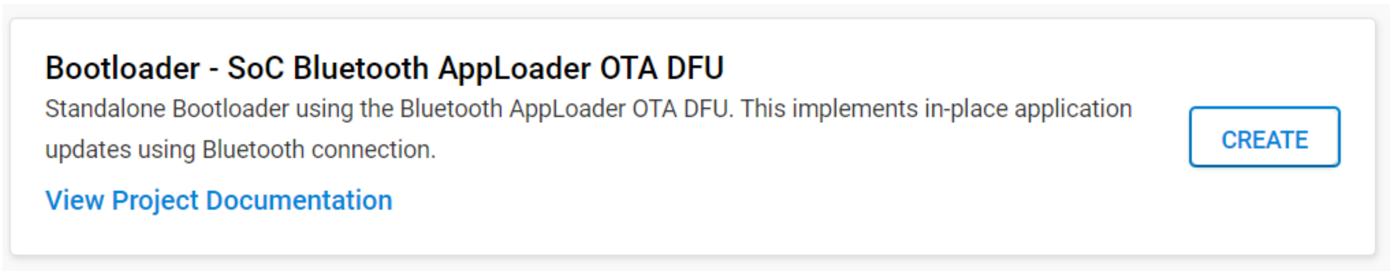
QSG168: Proprietary F

QSG169: Bluetooth Qu

QSG175: Silicon Labs

3. Create the `Bootloader - SoC Bluetooth Apploader OTA DFU` example project from Simplicity Studio

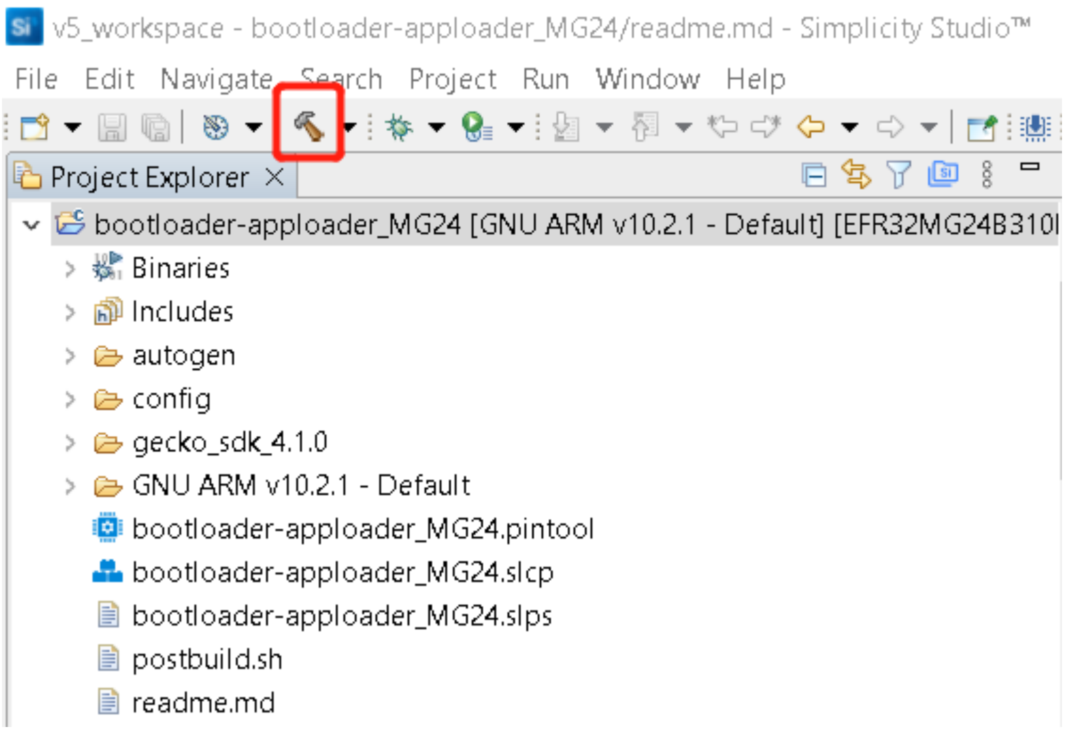
Launcher.

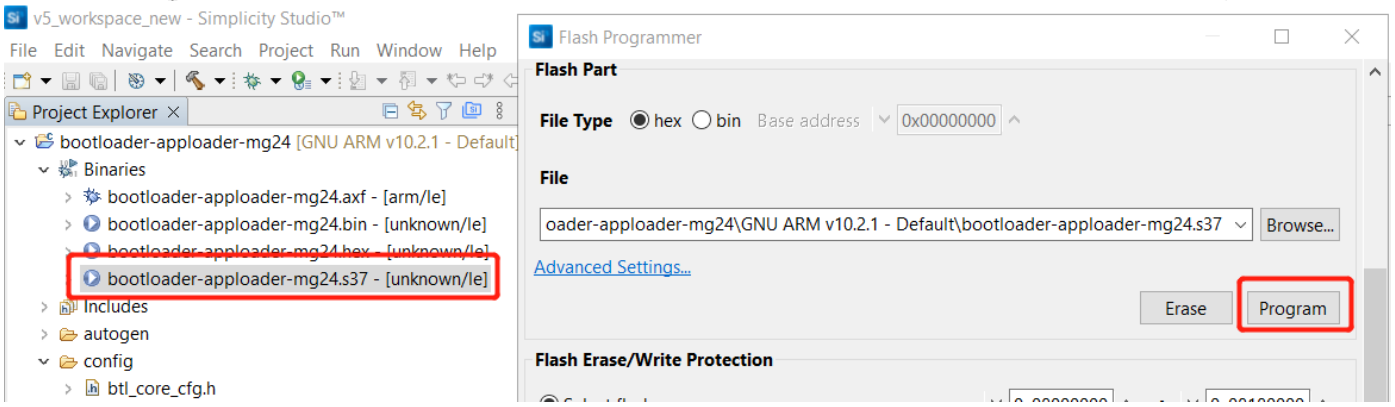

Bootloader - SoC Bluetooth AppLoader OTA DFU

Standalone Bootloader using the Bluetooth AppLoader OTA DFU. This implements in-place application updates using Bluetooth connection.

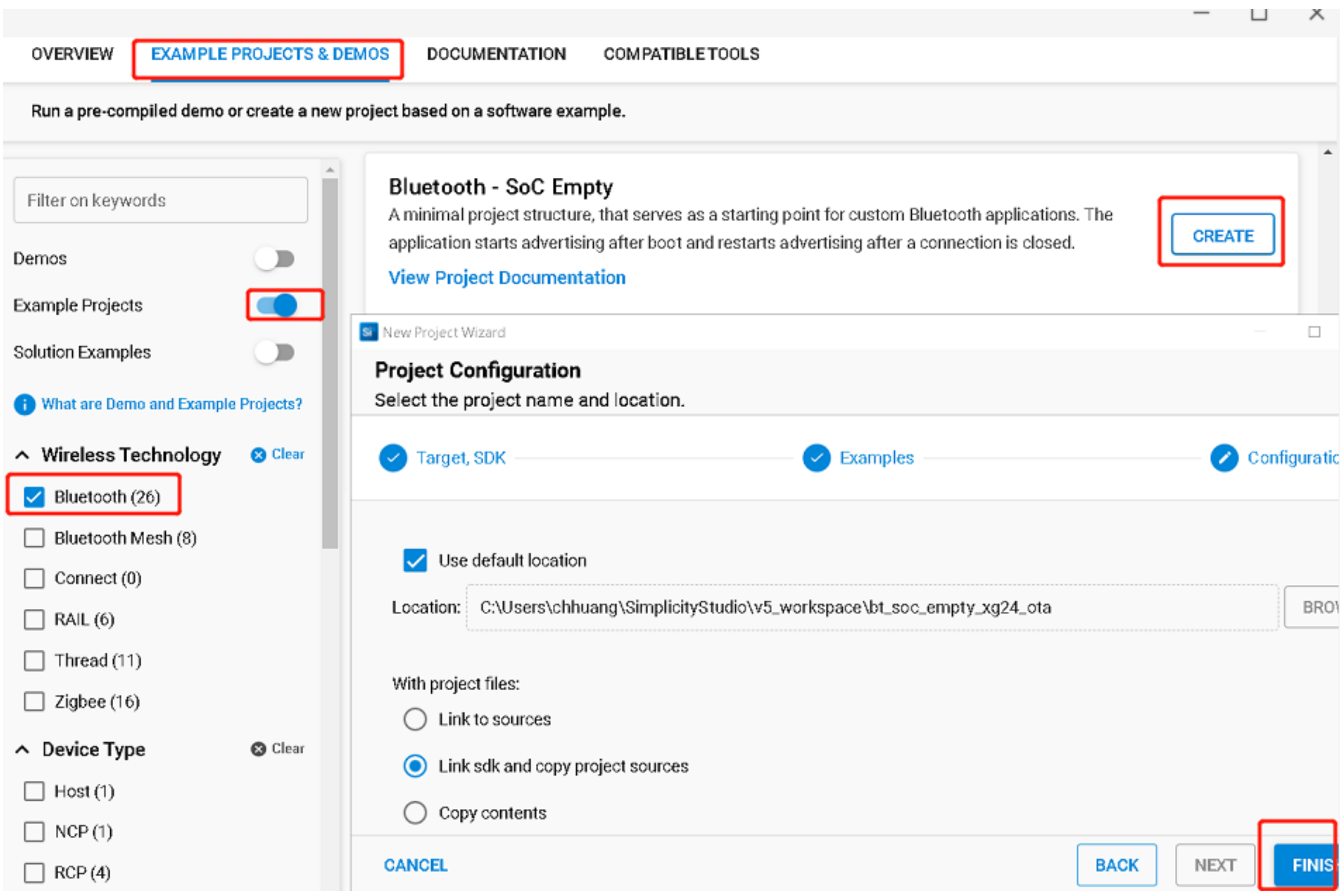
View Project Documentation

4. The created project will display on the "Project Explorer", then click the hammer icon to build the project. After building the bootloader image, right click on the "bootloader-apploader-mg24.s37" and select "Flash to Device" for programming the image to the device.


v5_workspace - bootloader-apploader_MG24/readme.md - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Project Explorer
bootloader-apploader_MG24 [GNU ARM v10.2.1 - Default] [EFR32MG24B310
Binaries
Includes
autogen
config
gecko_sdk_4.1.0
GNU ARM v10.2.1 - Default
bootloader-apploader_MG24.pintool
bootloader-apploader_MG24.slcp
bootloader-apploader_MG24.slps
postbuild.sh
readme.md


v5_workspace_new - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Project Explorer
bootloader-apploader-mg24 [GNU ARM v10.2.1 - Default]
Binaries
bootloader-apploader-mg24.axf - [arm/le]
bootloader-apploader-mg24.bin - [unknown/le]
bootloader-apploader-mg24.hex - [unknown/le]
bootloader-apploader-mg24.s37 - [unknown/le]
Includes
autogen
config
btl_core_cfg.h
Flash Programmer
Flash Part
File Type ● hex ○ bin Base address 0x00000000
File
oader-apploader-mg24\GNU ARM v10.2.1 - Default\bootloader-apploader-mg24.s37
Browse...
Advanced Settings...
Erase
Program
Flash Erase/Write Protection

5. Create the `Bluetooth - SoC Empty example` project from Simplicity Studio Launcher.



Run a pre-compiled demo or create a new project based on a software example.

Bluetooth - SoC Empty

A minimal project structure, that serves as a starting point for custom Bluetooth applications. The application starts advertising after boot and restarts advertising after a connection is closed.

Project Configuration

Select the project name and location.

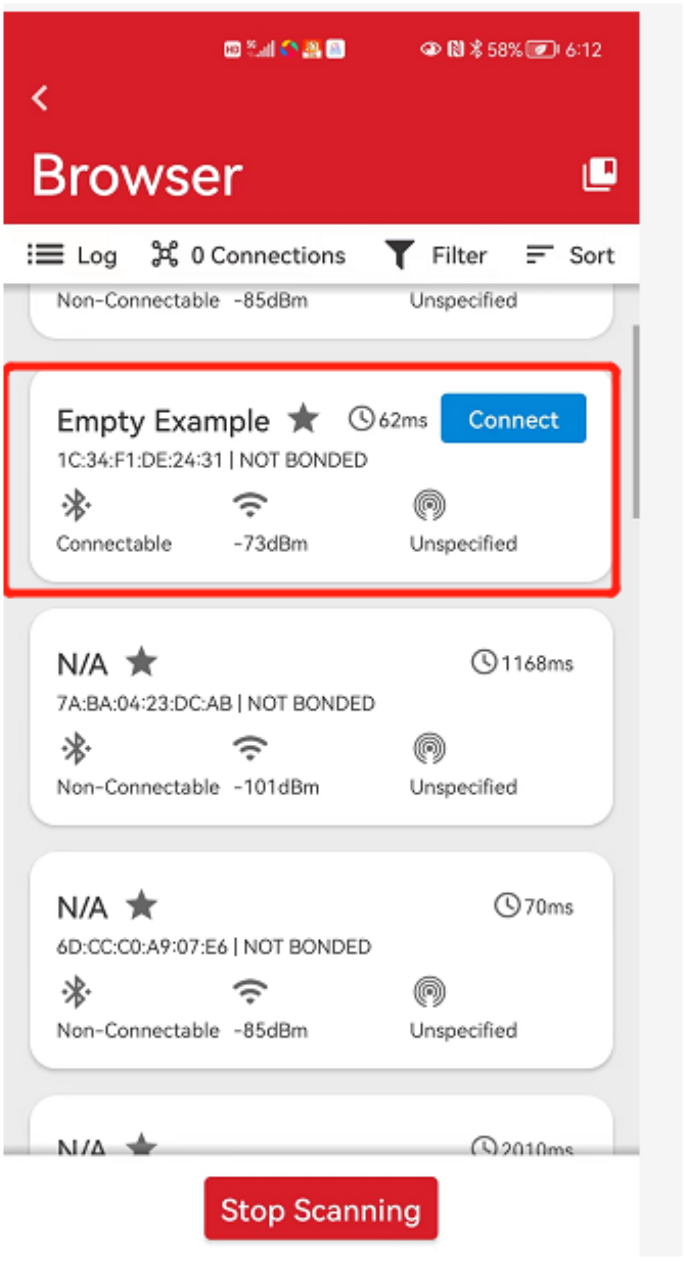
[✓] Use default location

Location: C:\Users\chhuang\SimplicityStudio\v5_workspace\bt_soc_empty_xg24_ota

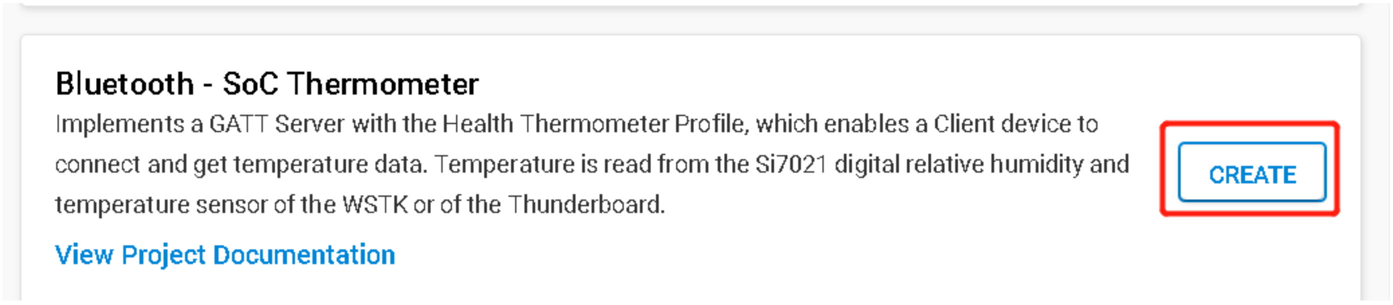
With project files:

Link to sources [✓ Link sdk and copy project sources] Copy contents

6. Build the `Bluetooth - SoC Empty` project and flash the firmware image to the device, then EFR connect can discover the device named "Empty Example".



7. Create the `Bluetooth - SoC Thermometer` example project.



Bluetooth - SoC Thermometer

Implements a GATT Server with the Health Thermometer Profile, which enables a Client device to connect and get temperature data. Temperature is read from the Si7021 digital relative humidity and temperature sensor of the WSTK or of the Thunderboard.

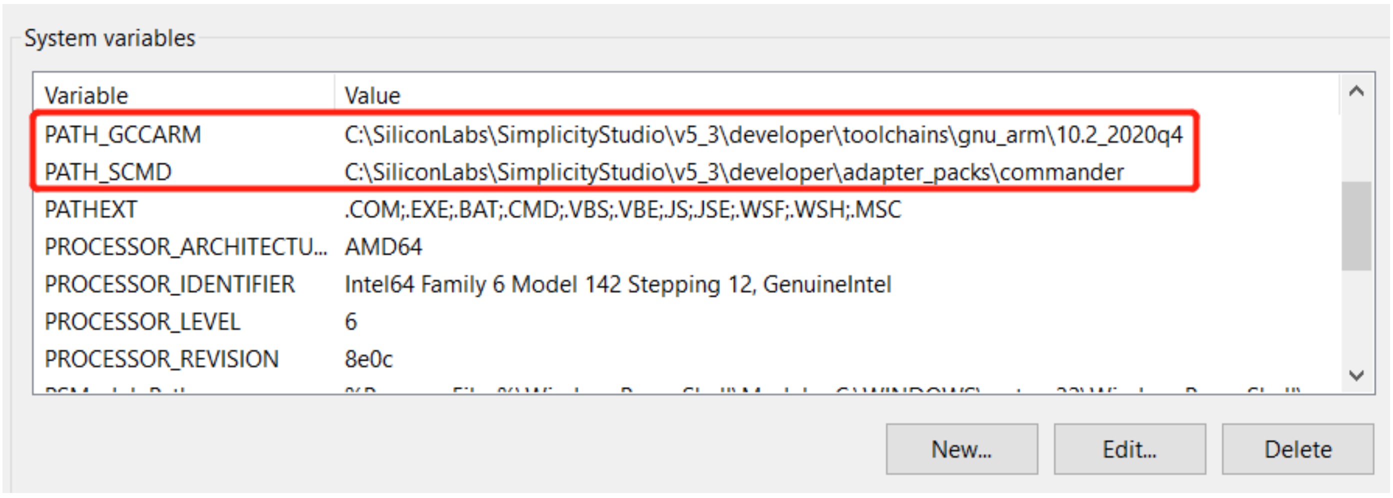
View Project Documentation

8. Build the `Bluetooth - SoC Thermometer project` and double click on the create_bl_files.bat script in the project tree. You may need to define two environment variables `PATH_SCMD` and `PATH_GCCARM` before running the script. The script creates a folder named output_gbl under your project and .gbl upgrade image files in this folder:

- application.gbl: user application (including full Bluetooth stack)

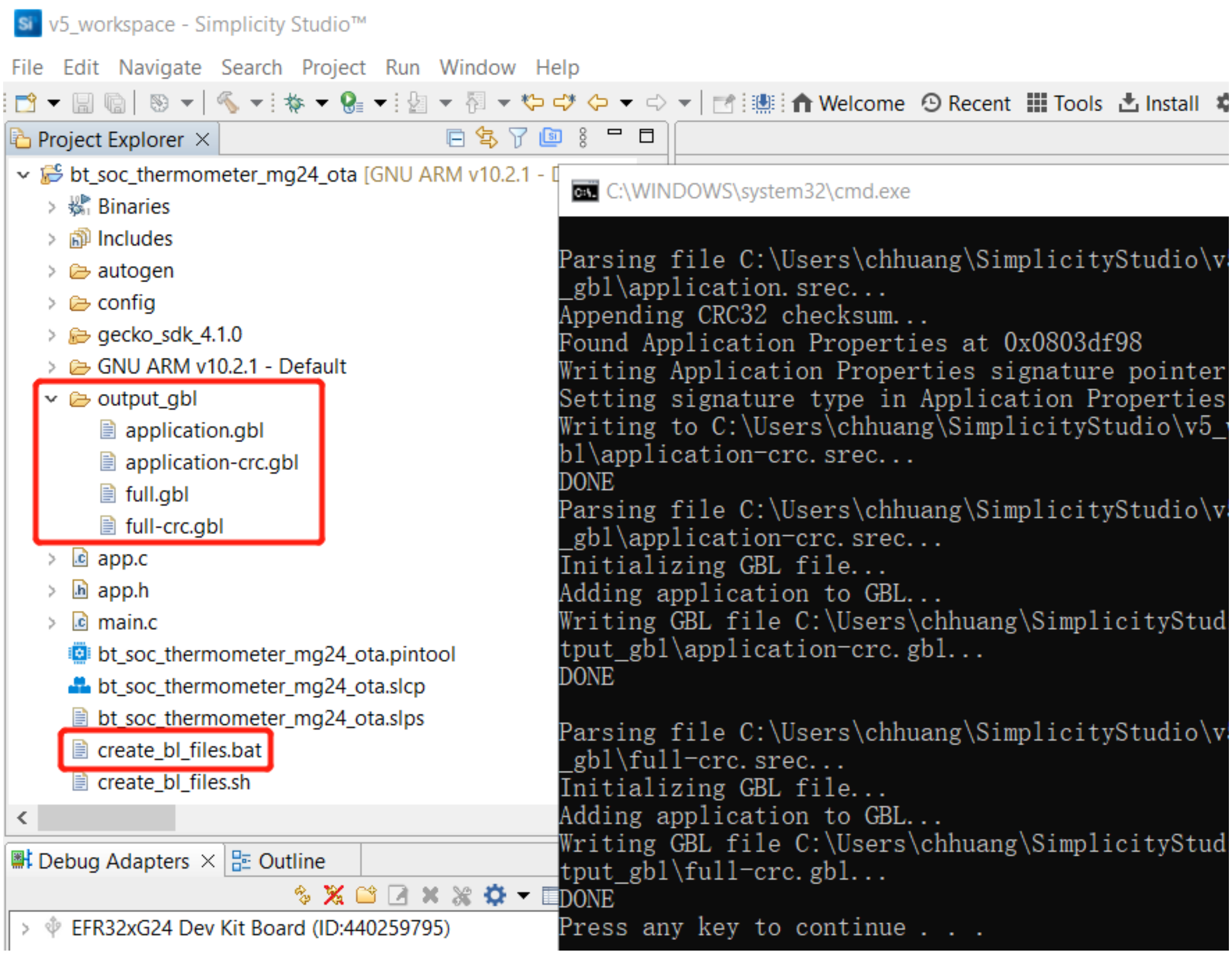
- application-crc.gbl: user application with a CRC32 checksum

Transfer the "application.gbl" file in the output_gbl folder to the phone.



System variables

Variable	Value
PATH_GCCARM	C:\SiliconLabs\SimplicityStudio\v5_3\developer\toolchains\gnu_arm\10.2_2020q4
PATH_SCMD	C:\SiliconLabs\SimplicityStudio\v5_3\developer\adapter_packs\commander
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e0c



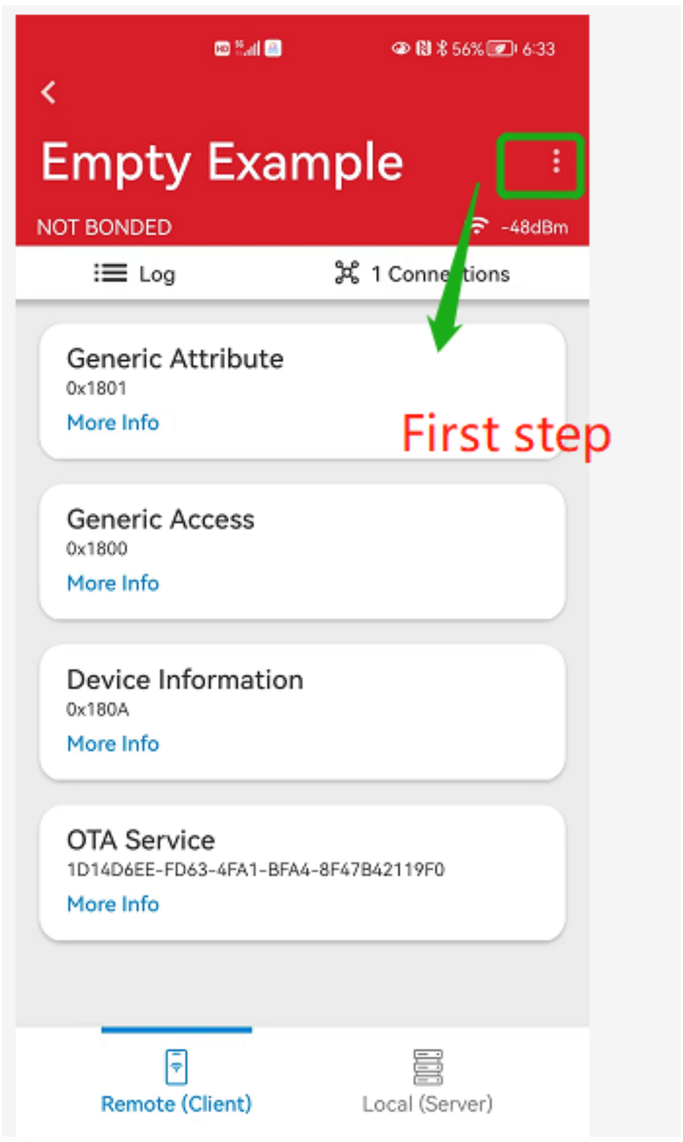
v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install
Project Explorer
Debug Adapters
Outline
C:\WINDOWS\system32\cmd.exe

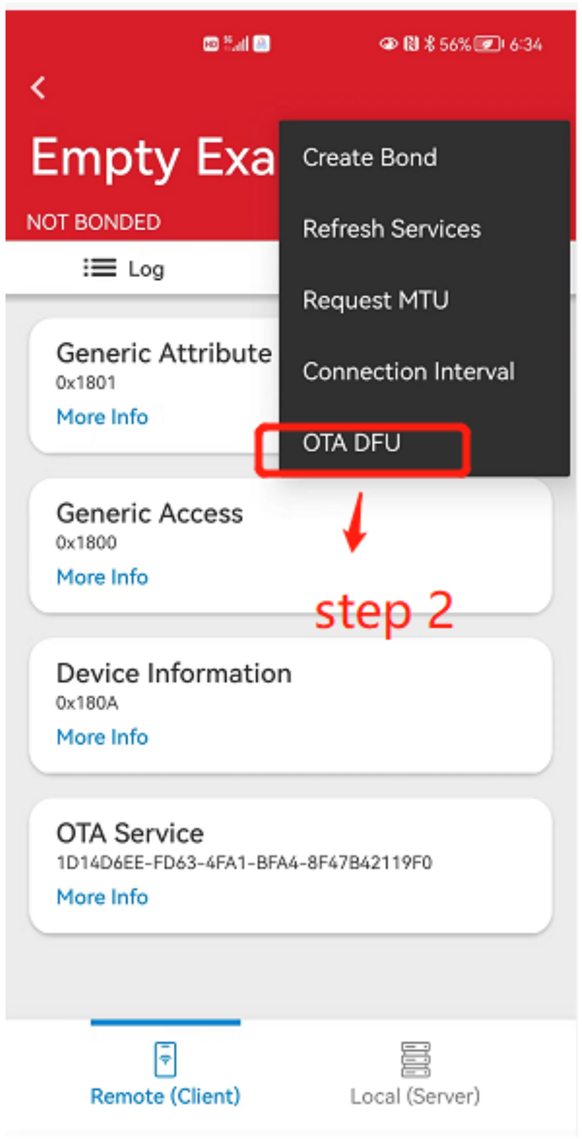
```
bt_soc_thermometer_mg24_ota [GNU ARM v10.2.1 - D
  Binaries
  Includes
  autogen
  config
  gecko_sdk_4.1.0
  GNU ARM v10.2.1 - Default
  output_gbl
    application.gbl
    application-crc.gbl
    full.gbl
    full-crc.gbl
  app.c
  app.h
  main.c
  bt_soc_thermometer_mg24_ota.pintool
  bt_soc_thermometer_mg24_ota.slcp
  bt_soc_thermometer_mg24_ota.slps
  create_bl_files.bat
  create_bl_files.sh
```

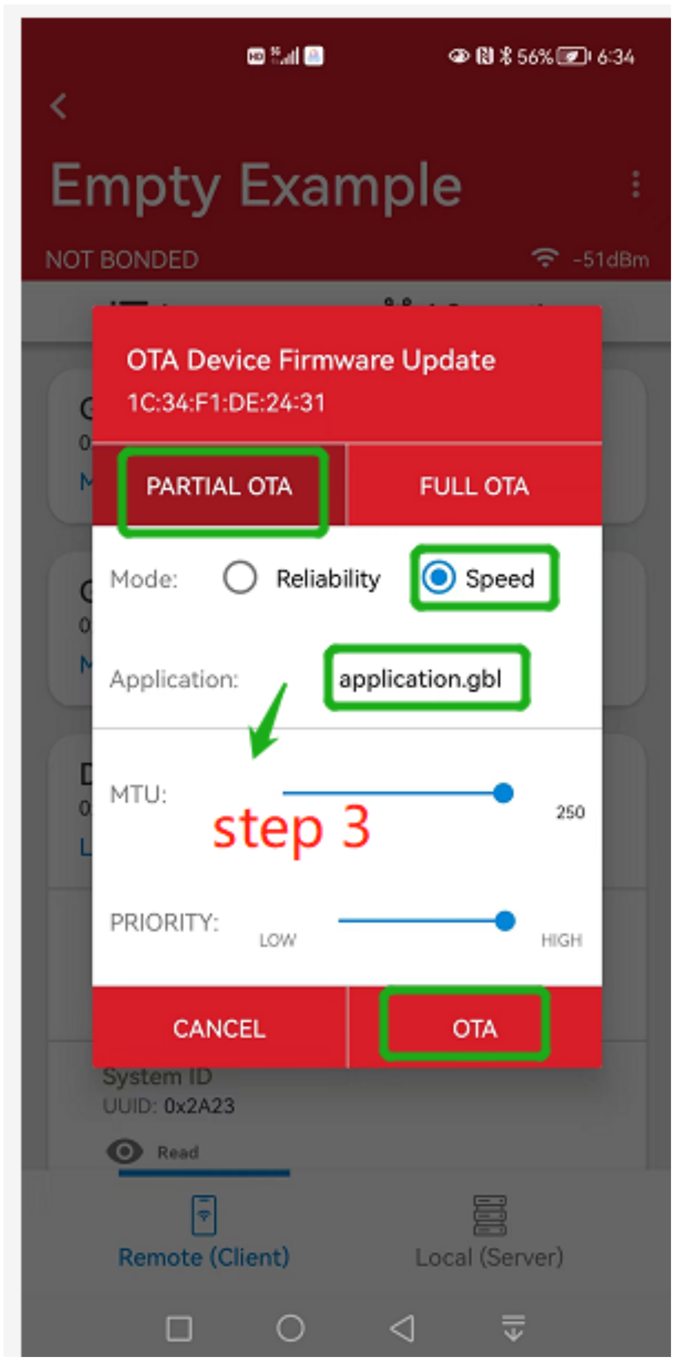
EFR32xG24 Dev Kit Board (ID:440259795)

```
Parsing file C:\Users\chhuang\SimplicityStudio\v
_gbl\application.srec...
Appending CRC32 checksum...
Found Application Properties at 0x0803df98
Writing Application Properties signature pointer
Setting signature type in Application Properties
Writing to C:\Users\chhuang\SimplicityStudio\v5_
bl\application-crc.srec...
DONE
Parsing file C:\Users\chhuang\SimplicityStudio\v
_gbl\application-crc.srec...
Initializing GBL file...
Adding application to GBL...
Writing GBL file C:\Users\chhuang\SimplicityStud
tput_gbl\application-crc.gbl...
DONE

Parsing file C:\Users\chhuang\SimplicityStudio\v
_gbl\full-crc.srec...
Initializing GBL file...
Adding application to GBL...
Writing GBL file C:\Users\chhuang\SimplicityStud
tput_gbl\full-crc.gbl...
DONE
Press any key to continue . . .
```

9. Open the EFR Connect app on your phone and follow up the picture below. Frist find the "OTA DFU" button, and select the "PARTIAL OTA" with "Speed" configuration. Find the application.gbl which is created by step 8. Then click the "OTA". Eventually, you'll search Thermometer Example in your "EFR Connect" app.


6:33
Empty Example
NOT BONDED
-48dBm
Log
1 Connections
Generic Attribute
0x1801
More Info
First step
Generic Access
0x1800
More Info
Device Information
0x180A
More Info
OTA Service
1D14D6EE-FD63-4FA1-BFA4-8F47B42119F0
More Info
Remote (Client)
Local (Server)


6:34
Empty Exa
NOT BONDED
Log
Create Bond
Refresh Services
Request MTU
Connection Interval
OTA DFU
step 2
Generic Attribute
0x1801
More Info
Generic Access
0x1800
More Info
Device Information
0x180A
More Info
OTA Service
1D14D6EE-FD63-4FA1-BFA4-8F47B42119F0
More Info
Remote (Client)
Local (Server)


Empty Example

NOT BONDED -51dBm

OTA Device Firmware Update
1C:34:F1:DE:24:31

[✓ PARTIAL OTA] FULL OTA

Mode: Reliability [✓ Speed]

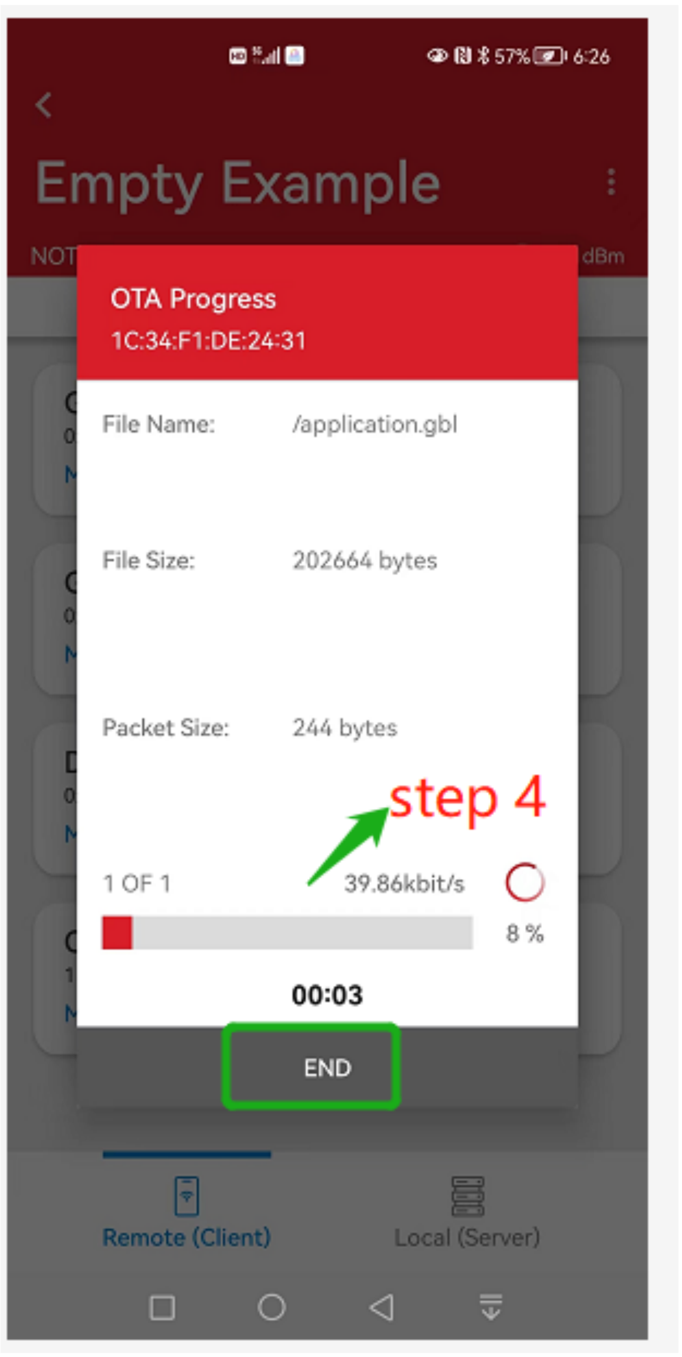
Application: application.gbl

MTU: 250

PRIORITY: LOW HIGH

CANCEL OTA

System ID
UUID: 0x2A23


Empty Example

OTA Progress
1C:34:F1:DE:24:31

File Name: /application.gbl

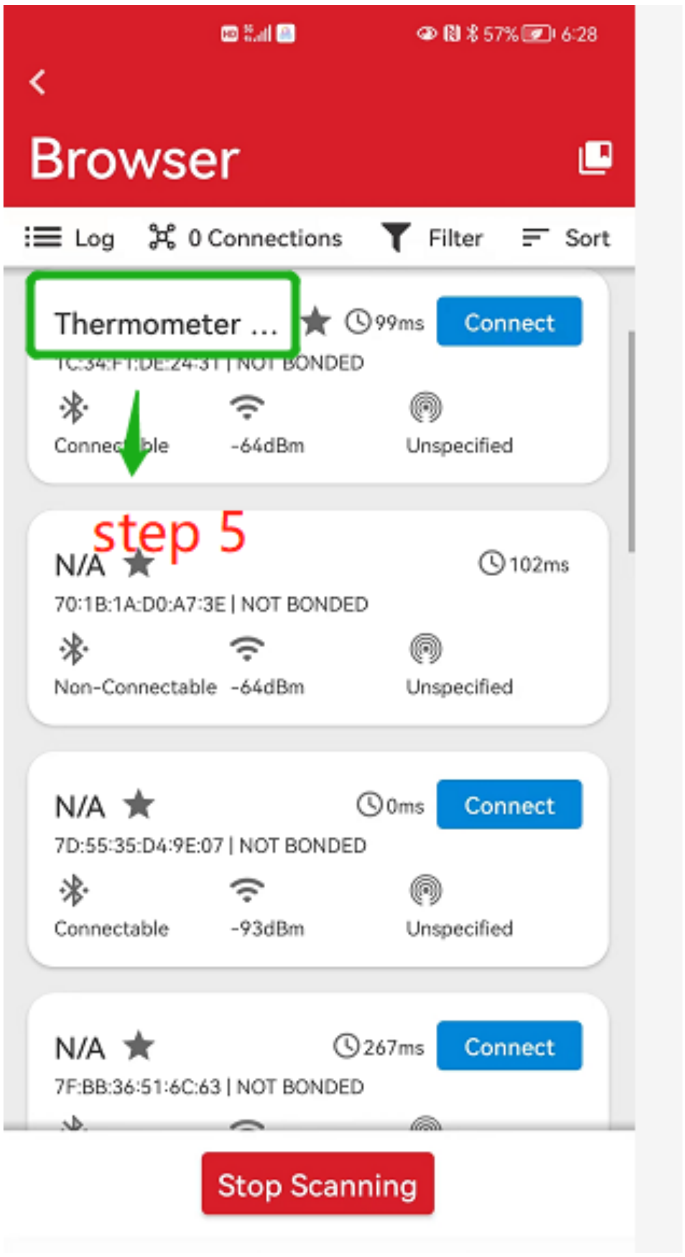
File Size: 202664 bytes

Packet Size: 244 bytes

1 OF 1 39.86kbit/s 8 %

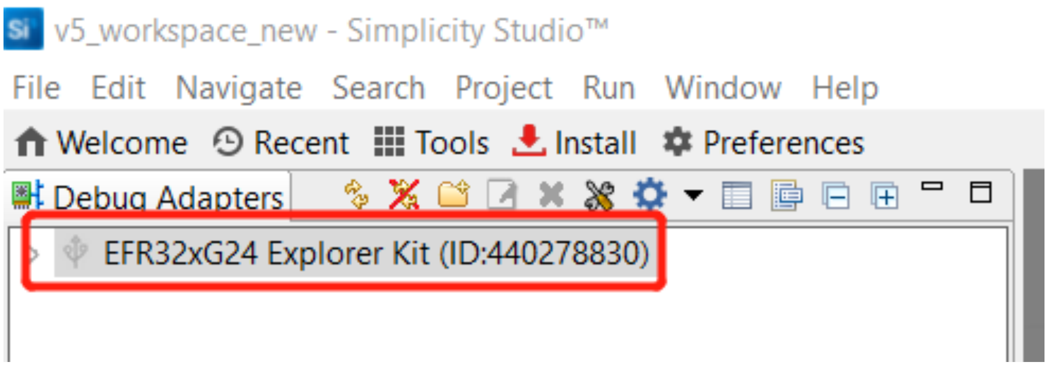
00:03

END

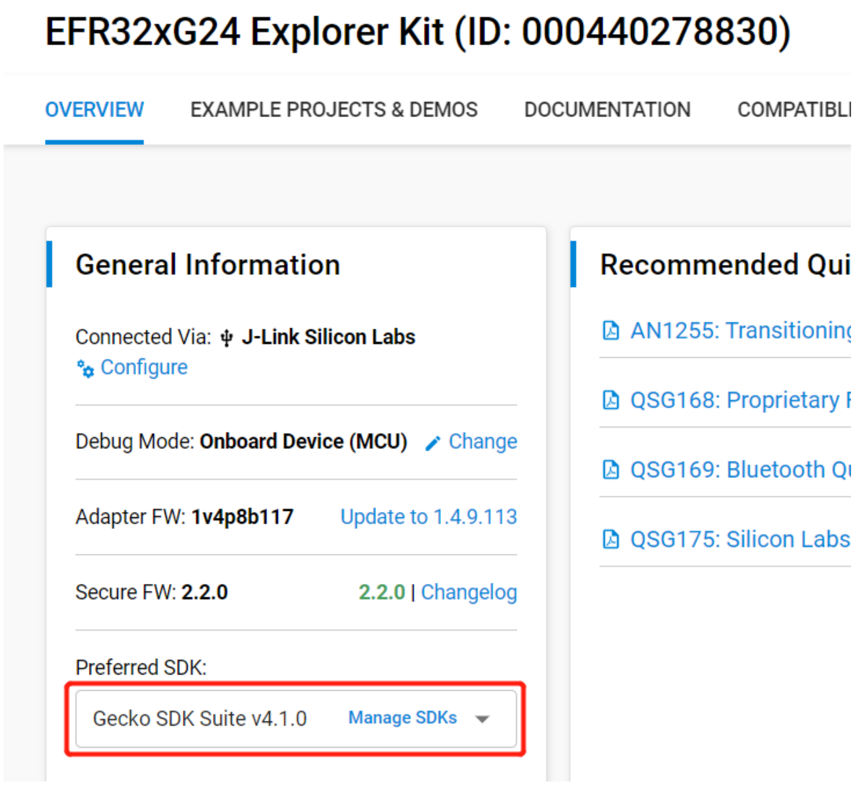


4.4. User Application DFU Lab Steps

1. Connect the kit to your computer and select it in Simplicity Studio.


v5_workspace_new - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Debug Adapters
EFR32xG24 Explorer Kit (ID:440278830)

2. Select Gecko SDK Suite v4.1.0 in Preferred SDK.


EFR32xG24 Explorer Kit (ID: 000440278830)

General Information

Connected Via: J-Link Silicon Labs
Configure

Debug Mode: Onboard Device (MCU) Change

Adapter FW: 1v4p8b117 Update to 1.4.9.113

Secure FW: 2.2.0 2.2.0 | Changelog

Preferred SDK:
Gecko SDK Suite v4.1.0 Manage SDKs

Recommended Qui

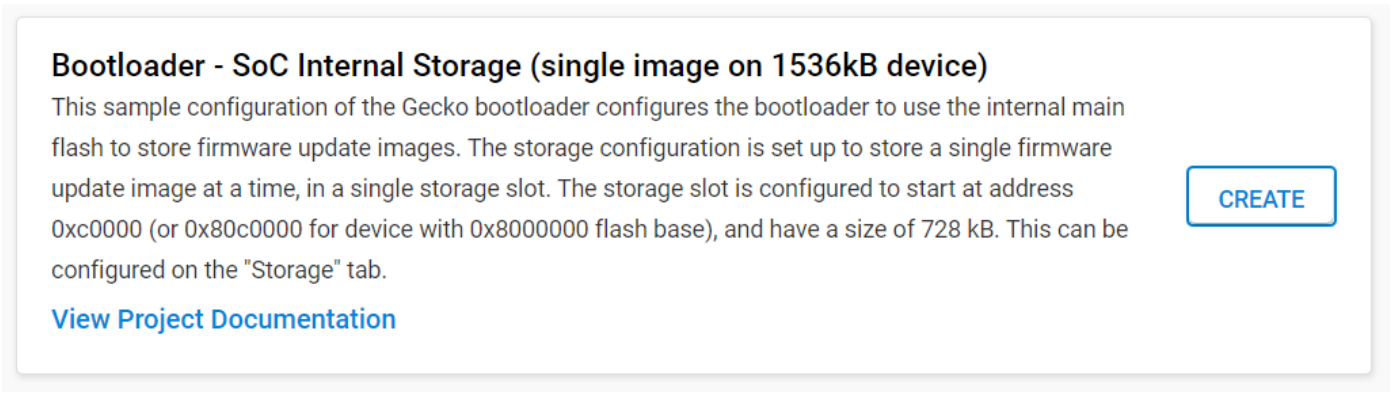
AN1255: Transitioning

QSG168: Proprietary F

QSG169: Bluetooth Qu

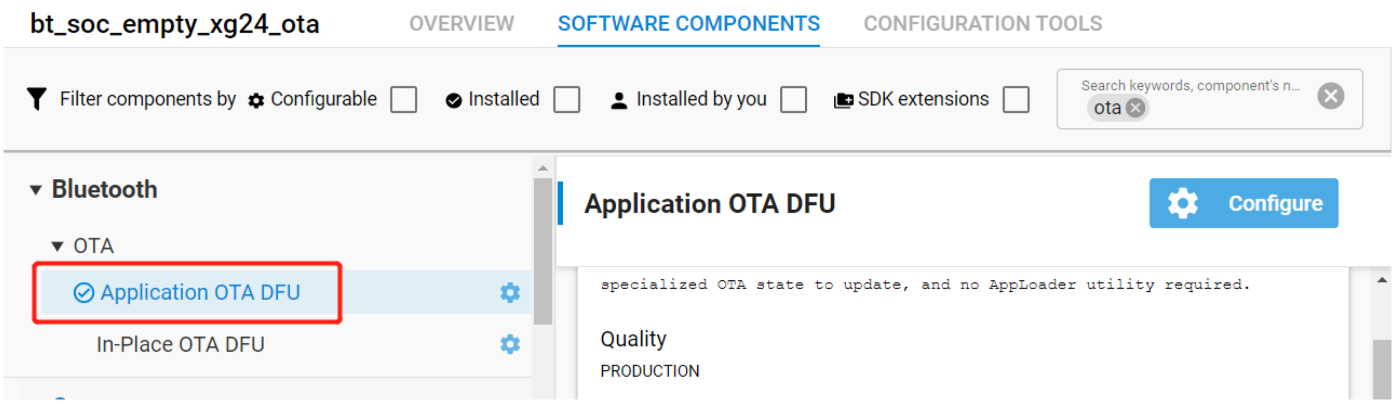
QSG175: Silicon Labs

3. Create the `Bootloader - Soc Internal Storage` example project named bootloader-storage-internal-single-1536k-xg24 from Simplicity Studio Launcher.

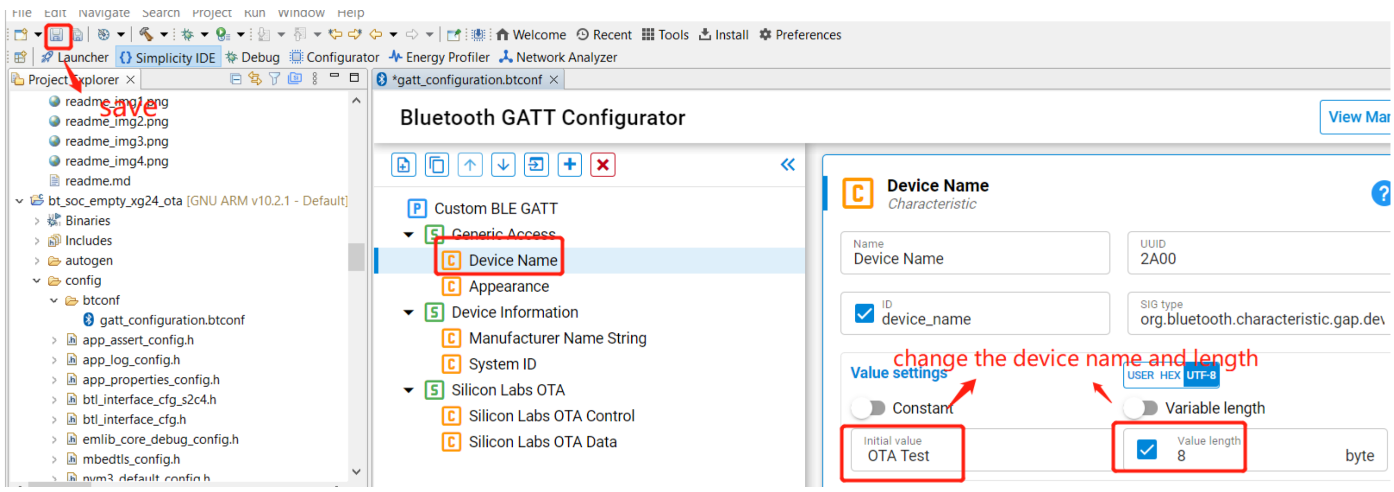

Bootloader - SoC Internal Storage (single image on 1536kB device)

This sample configuration of the Gecko bootloader configures the bootloader to use the internal main flash to store firmware update images. The storage configuration is set up to store a single firmware update image at a time, in a single storage slot. The storage slot is configured to start at address 0xc0000 (or 0x80c0000 for device with 0x8000000 flash base), and have a size of 728 kB. This can be configured on the "Storage" tab.

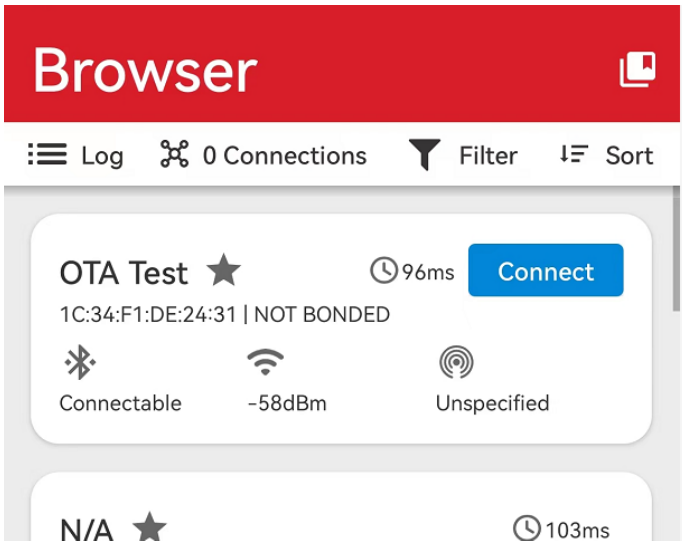
4. The created project will display on the "Project Explorer", then click the hammer icon to build the project. After building the bootloader image, right click on the "bootloader-storage-internal-single-1536k-xg24.s37" and select "Flash to Device" for programming the image to the device.
5. Create the `Bluetooth - SoC Empty` example project from Simplicity Studio Launcher named bt_soc_empty_xg24_ota.
6. Double click on the bt_soc_empty_xg24_ota.slcp file in the project tree. Then remove "In-Place OTA DFU" component and add the "Application OTA DFU" component according to the section 3.2.3.



7. Build the "bt_soc_empty_xg24_ota" project and flash the firmware image to the device, then EFR connect can discover the device named "Empty Example"
8. Double click on the gatt_configuration.btconf file in the project tree. Select the "Device Name", then set the "Value settings" as "OTA test" and "8" byte. Build the "bt_soc_empty_xg24_ota" project, and double click on the create_bl_files.bat script in the project tree. Transfer the "application.gbl" file in the output_gbl folder to the phone.



9. Open the EFR Connect app on your phone and follow up the picture in section 4.3 step 9. Frist find the OTA DFU button, and select the PARTIAL OTA with "Speed" configuration. Find the application.gbl which is created by step 9. Then click the OTA. Eventually, you'll search "OTA test" in your "EFR Connect" app.



Reference

UG103.6: Bootloader Fundamentals
an1086-gecko-bootloader-bluetooth
UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher
UG266: Silicon Labs Gecko Bootloader User's Guide for GSDK 3.2 and Lower
How to use bootloader and ota in your project