

Developer Guide

Security Starter Kit with GG11, XBee3 and OPTIGA™ Trust M

Date: December 1, 2020 | Version 1.0

FINAL



The Solutions People



Confidentiality Notice

Copyright (c) 2020 eInfochips. - All rights reserved

This document is authored by eInfochips and is eInfochips intellectual property, including the copyrights in all countries in the world. This document is provided under a license to use only with all other rights, including ownership rights, being retained by eInfochips. This file may not be distributed, copied, or reproduced in any manner, electronic or otherwise, without the express written consent of eInfochips.

CONTENTS

1. INTRODUCTION.....	4
1.1. Purpose of the Document	4
1.2. Architecture	4
1.3. SSK Suit Package Contents.....	5
2. SETTING UP THE DEVELOPMENT ENVIRONMENT	6
2.1. Hardware setup GG11-LTE-M-SSK kit.....	6
2.2. Set up LTE-M expansion module SLEXP8021A	6
2.3. Connect Optiga™ Trust-M	7
2.4. Software Setup	8
2.4.1. Simplicity Studio	8
2.4.2. Simplicity Commander Tool.....	8
2.4.3. OpenSSL	8
2.4.4. AWS CLI.....	8
2.4.5. Terminal emulator.....	8
2.5. AWS configuration	9
2.5.1. AWS IoT Thing creation.....	9
2.5.2. OTA configuration	11
2.6. Building the bootloader source	11
2.6.1. Importing Source.....	11
2.6.2. Configuring source	15
2.6.3. Building and Flashing	18
3. AMAZON FREERTOS APPLICATIONS	20
3.1. AWS Source	20
3.2. OPTIGA™ Trust M Source	20
3.3. LTE-M Xbee®3 module source	21
4. BUILDING AMAZON FREERTOS SOURCE	22
4.1. Import the Source	22
4.2. Configure the source	22
4.3. Application Configuration.....	23
4.4. Sim card activation.....	23
4.5. Building and Flashing.....	24
4.6. Acquire Device Certificate	25
4.7. Verify XBee®3 LTE-M Module's info.....	26
5. SAMPLE LOGS	27
5.1. MQTT Demo Logs	27
5.2. OTA Demo Logs	27
6. AWS DEMOS.....	28
6.1. General description	28
6.2. MQTT demo.....	28
6.2.1. Source config	28

6.2.2.	Run MQTT Demo.....	28
6.3.	OTA demo.....	30
6.3.1.	Source config	30
6.3.2.	Run OTA Demo	31
7.	MODIFYING SOURCE.....	36
7.1.	Source files	36
7.2.	Configuration macros in IDE Settings	36
7.3.	Enable logging	36
8.	TROUBLESHOOTING	38
9.	REFERENCES	39
9.1.	Reference Documents	39

DEFINITION, ACRONYMS AND ABBREVIATIONS

Definition/Acronym/Abbreviation	Description
MCU	Microcontroller Unit
LTE-M	Long Term Evolution for Machines
AWS	Amazon Web Services
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
OTA	Over the Air
ARN	Amazon Resource name
IP	Internet Protocol
SIM	Subscriber Identity Module
CLI	Command Line Interface
IMEI	International Mobile Equipment Identity
SSK	Security Starter Kit

1. INTRODUCTION

1.1. Purpose of the Document

This document describes how to get started implementing Amazon FreeRTOS port on for GG11-LTE-M-SSK ; i.e. SLSTK3701A EFM32 Giant Gecko S1 with SLEXP8021A LTE-M kit.

Many of the instructions may be replicate of Amazon AWS FreeRTOS web documentation. This document will point to the web pages and does not reproduce them This document will facilitate with tips to use the instructions from Amazon. It also describes how to port OPTIGA™ Trust M on FreeRTOS based MCU device.

1.2. Architecture

The EFM32 Giant Gecko 11 Starter Kit (SLSTK3701A) is connected to the XBee3 LTE-M expansion kit (SLEXP8021A) with operational SIM card and UFL Antenna. The LTE-M expansion kit provides cellular connectivity to access AWS services over the internet. The MQTT protocol is mainly used for communication purpose.

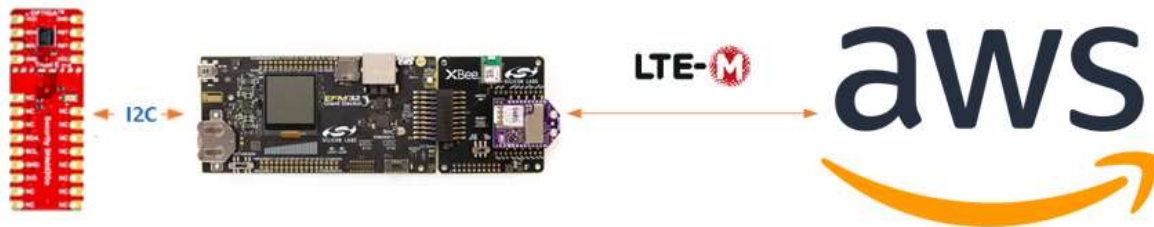


Figure 1: Hardware Setup

1.3. SSK Suit Package Contents

Download the complete SSK Suit package. The Firmware and source code can be found in the github repository; <https://github.com/ArrowElectronics/Security-Starter-Kits>

The Developer and Quick Start Guides can be found at the following link;
<https://www.arrow.com/en/products/gg11-lte-m-ssk/arrow-development-tools>

- Developer_Guide_GG11-LTE-M-SSK
- Quick_Start_Guide_GG11-LTE-M-SSK
- Firmware Image
- Source Code
- Releasenote.txt
- SSK_Cert_And_Config

2. SETTING UP THE DEVELOPMENT ENVIRONMENT

2.1. Hardware setup GG11-LTE-M-SSK kit

The following hardware will be needed for this project:

- GG11-LTE-M-SSK kit contents:
 - Silicon Labs Giant Gecko 11 Starter Kit (SLEXP8021A)
 - Silicon Labs LTE-M Expansion Kit (SLSTK3701A)
 - Cellular SIM card (Hologram SIM – not shown)
 - LTE Antenna (not shown)
 - Infineon Shield2Go Cloud Security OPTIGA™ Trust M
 - Custom cable to connect OPTIGA™ Trust M board with the Giant Gecko 11 Starter Kit
 - Micro USB cable (required for power and communication with the Host PC)
- Development Host PC running Linux

NOTE: Make sure to perform hardware set up in an electrostatic environment

2.2. Set up LTE-M expansion module SLEXP8021A

The Xbee®3 LTE-M cellular expansion module is used for communication medium

1. Connect the included patch antenna to the module's u.FL connector labeled 'CELL' port.
2. Make sure to have XBXC-11415 firmware version for Xbee®3 LTE-M module (refer section 4.7).

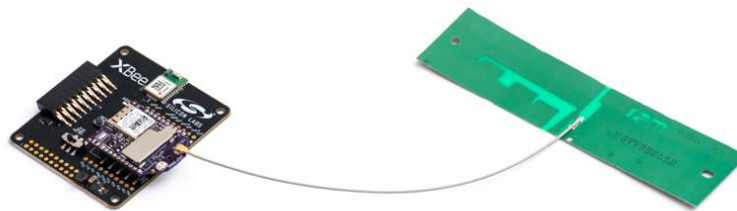


Figure 2: Patch Antenna Setup

3. After attaching Antenna insert 4FF Nano LTE-M enabled SIM card into provided slot.



Figure 3: Sim card in LTE-M module

4. Assemble the LTE-M expansion board to the Giant Gecko 11 Starter Kit
5. Set switch on SLSTK3701A to AEM (Advanced Energy Monitor) mode
6. Set switch on SLEXP8021A to high power (dc-dc) mode

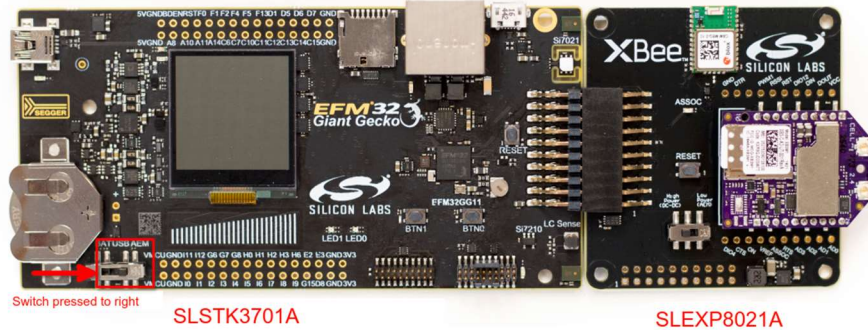


Figure 4: LTE-M kit Assembling

2.3. Connect Optiga™ Trust-M

1. Connect Optiga™ Trust-M shield2go board with SLEXP8021A Expansion kit using the custom cable as shown below;

Connections No	OPTIGA™ TrustM	LTE-M EXP-Header
1(Red)	VCC	20 (+3v3)
2(Black)	GND	1 (GND)
3(Green)	RST	3 (PA12)
4(White)	SCL	15 (PC1)
5(Orange)	SDA	16 (PC0)

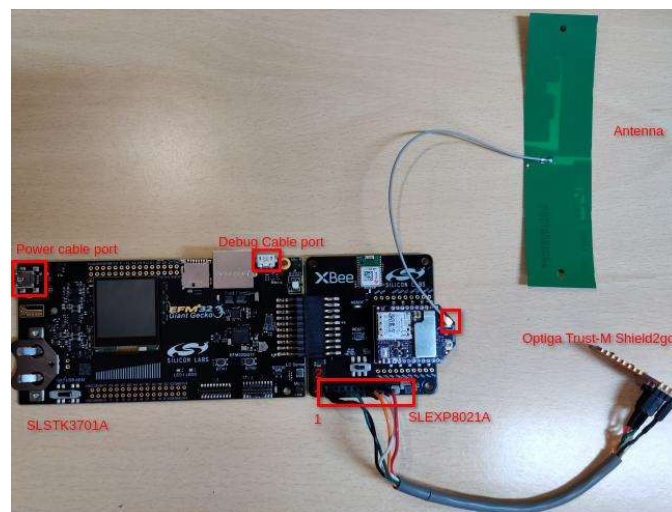


Figure 5: LTE-M Connections

2.4. Software Setup

2.4.1. Simplicity Studio

Simplicity Studio Version: SV4.1.13.4 (or later) is required to compile the software with the GUI

- Install Simplicity Studio Silicon labs site:
<https://www.silabs.com/products/development-tools/software/simplicity-studio>
- SDK version 2.7
- Gecko bootloader version 1.10.3
- GNU ARM v7.2.1 GCC compiler

Note: Java Runtime Environment is needed for Simplicity studio. See installation instructions on Ubuntu (18.04) PC

2.4.2. Simplicity Commander Tool

The latest version of Simplicity Commander 1v10p0b810 (or later) is required to perform secure boot and encrypted binary operations.

- Install Simplicity Commander from Silicon labs site:
<https://www.silabs.com/products/mcu/programming-options>.

2.4.3. OpenSSL

OpenSSL is needed for certificate creation if you want to do OTA update.

2.4.4. AWS CLI

AWS CLI is needed to automate various tasks and import OTA certificate in AWS.

2.4.5. Terminal emulator

A serial terminal software utility like Tera Term, putty or Minicom is needed to interact with Amazon FreeRTOS application running on SLSTK3701A Giant gecko kit (using USB virtual COM port).

2.5. AWS configuration

2.5.1. AWS IoT Thing creation

1. Signing in to AWS console from <https://aws.amazon.com/console/>
2. Go to Services -> IoT Core
3. Select Manage -> Things and choose **Create**

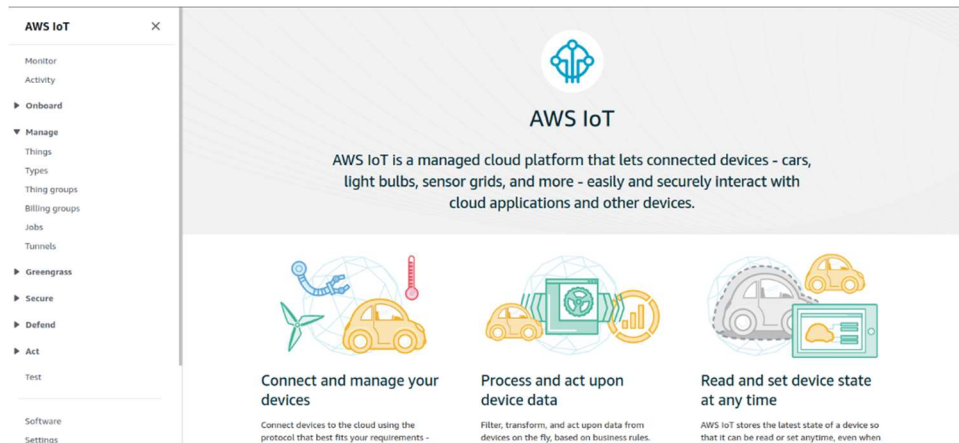


Figure 6: AWS Console

4. Choose to create single thing

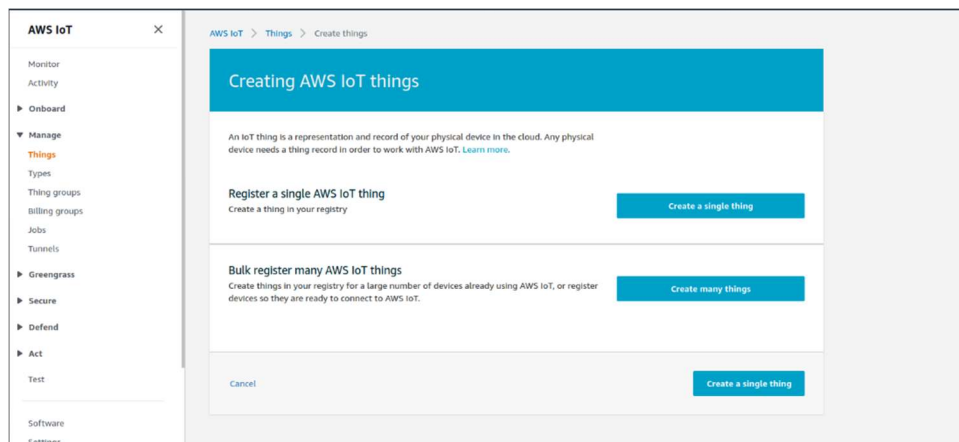


Figure 7: AWS Create Thing

5. In **Add your device to the thing registry** add name and click Next
6. In **Add a certificate for your thing** choose **Skip certificate and create thing**
7. Go to Secure -> Certificates from left navigation pane and click Create

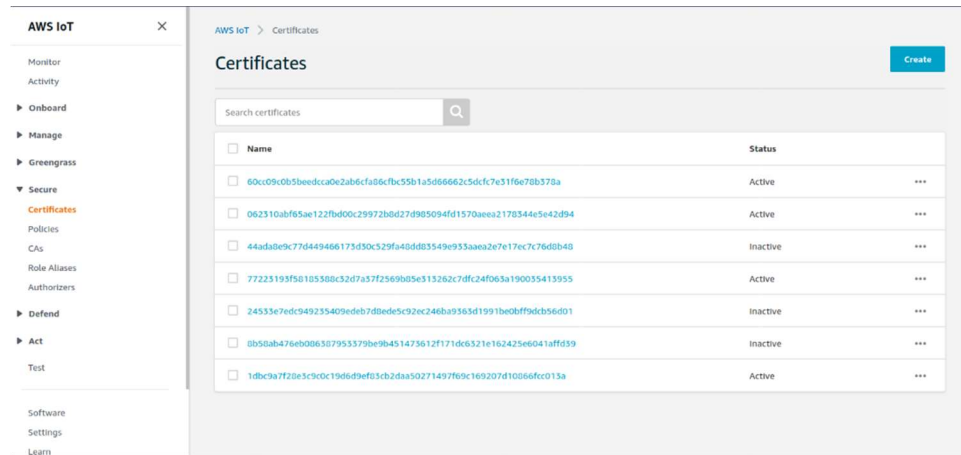


Figure 8: AWS Certificate creation

8. In **Create a certificate** select **Use my certificate**
9. In **Select a CA** click **Next**
10. In **Register existing device certificates** click **Select certificates**
11. Attach deviceCert.pem certificate generated from 4.6 here and check active all and click **Register certificates**
12. Go to **Secure -> Policies** from left navigation pane and click **Create**
13. In **Create a policy** provide name and click **Advanced mode** from right side
14. Paste below policy and click **Next**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

15. Go to **Secure -> Certificates** from left navigation pane
16. Verify recently added certificate number and click on the **certificate number**
17. Go to **Actions** and click to open drop down

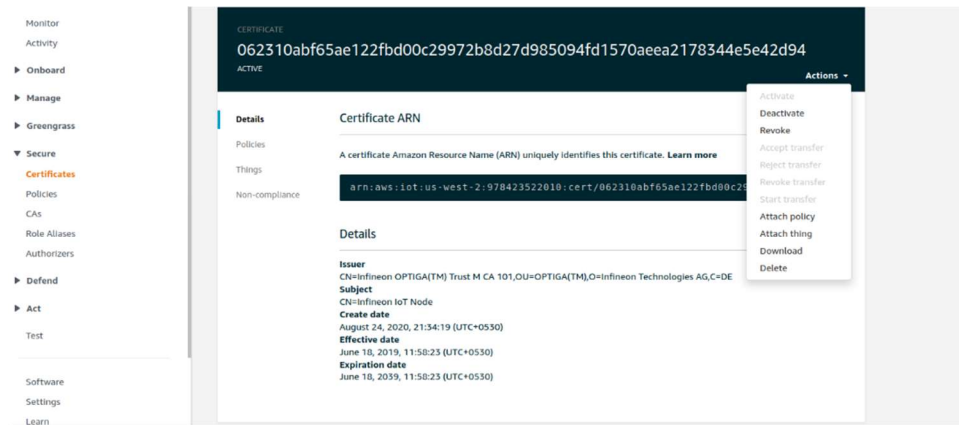


Figure 9: Thing, policy and certificate attachment

18. Select **Attach thing** and add recently added thing (as per point 6)
19. Select **attach policy** and add recently added policy (as per point 14)

2.5.2. OTA configuration

To perform OTA, user should have to configure the below listed items on [AWS console](#)

- Check the [Prerequisites](#) for OTA updates using MQTT.
- Create an Amazon S3 bucket to store your update.
- Create an OTA Update service role.
- Create an OTA user policy.
- Create a code-signing certificate.
- Grant access to code signing for AWS IoT
- Detailed configuration and steps available on AWS document.

<https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html>

2.6. Building the bootloader source

2.6.1. Importing Source

- Open Simplicity Studio IDE
- Go to **File -> New -> Project**

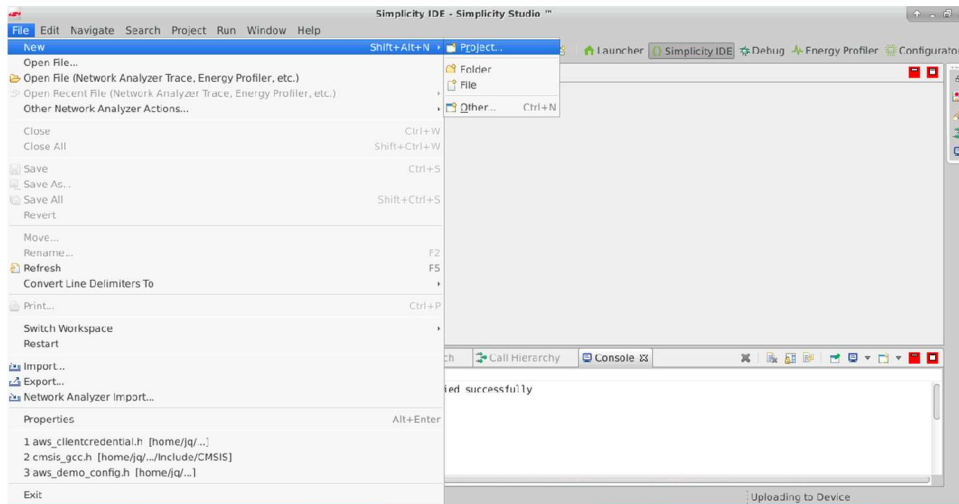


Figure 10: Create New project

- In New Project Window Choose **Silicon Labs App Builder** Project and select **Next**

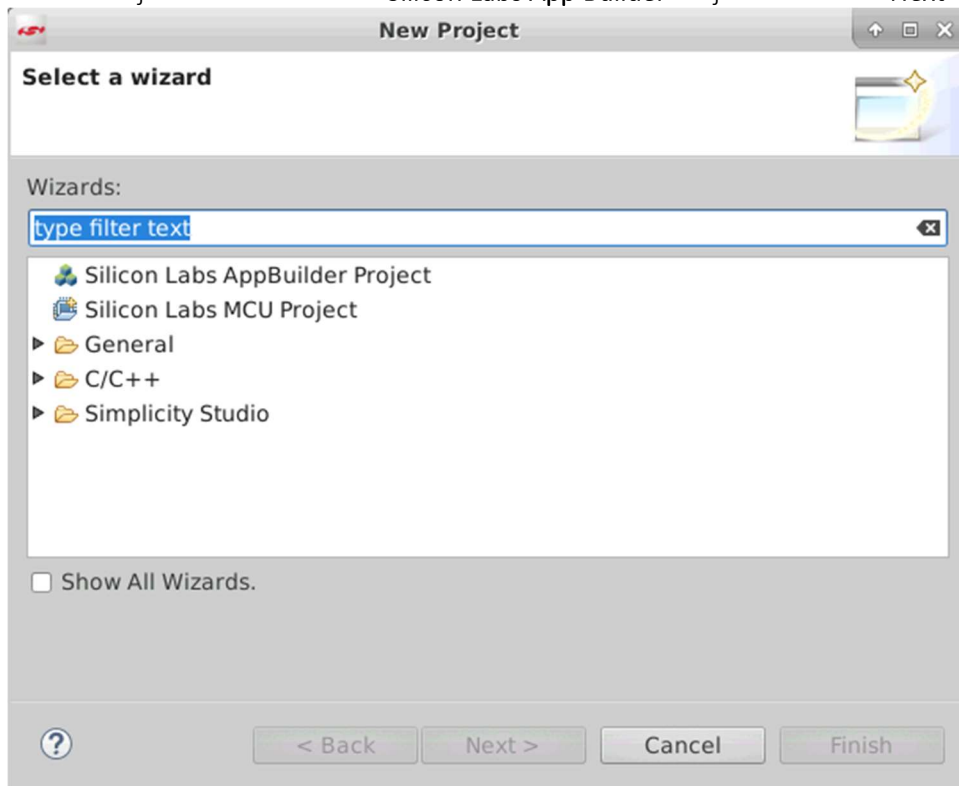


Figure 11: Select AppBuilder Project

- Choose **Gecko Bootloader** and select **Next**

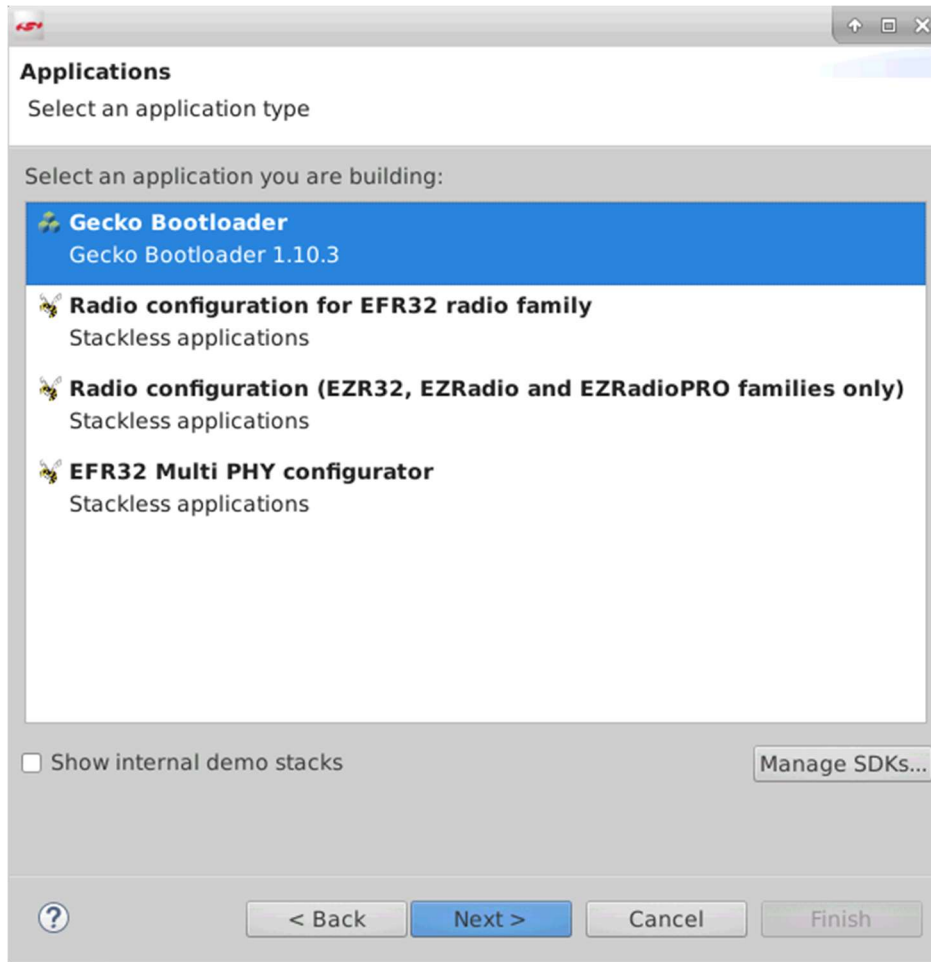


Figure 12: Selecting Gecko Bootloader

- Choose **Gecko Bootloader 1.10.3** and select **Next**
- Choose **Internal Storage Bootloader (Single image on 2MB device)** and select **Next**

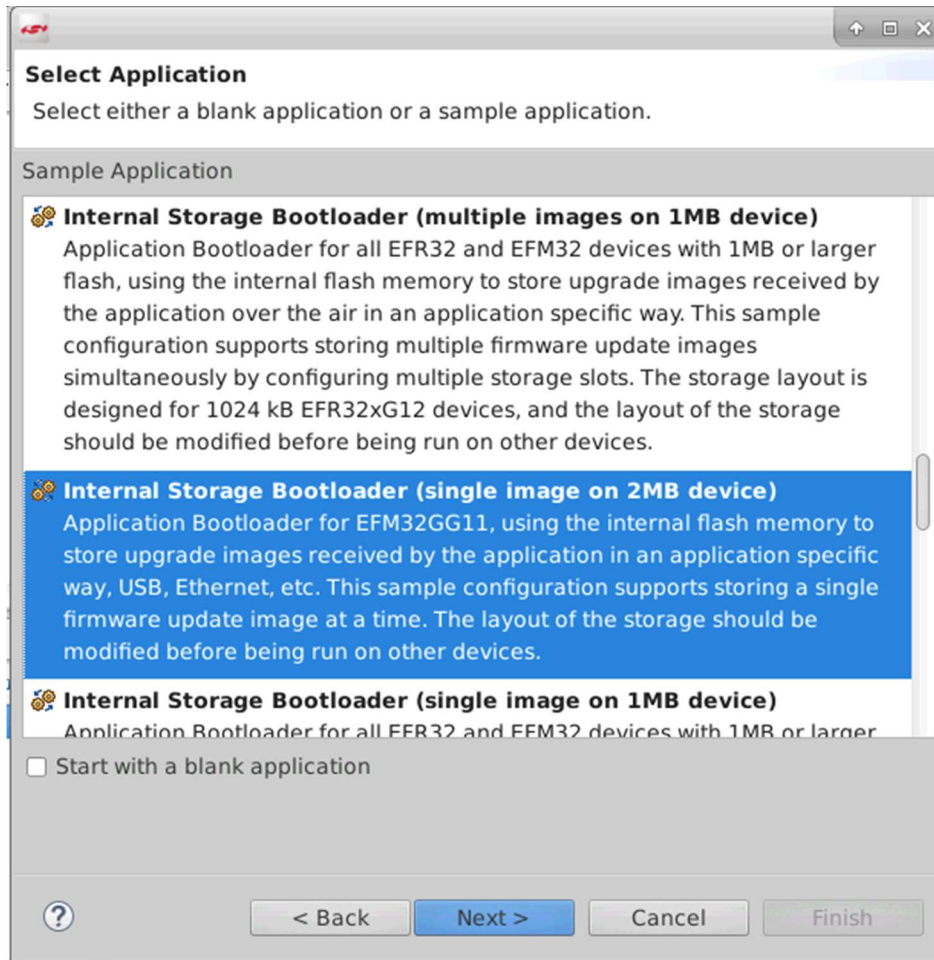


Figure 13: Selecting Internal-Storage Bootloader

- Use default location and select next
- Use **EFM32GG11B820F2048GL192** part number and select **finish**

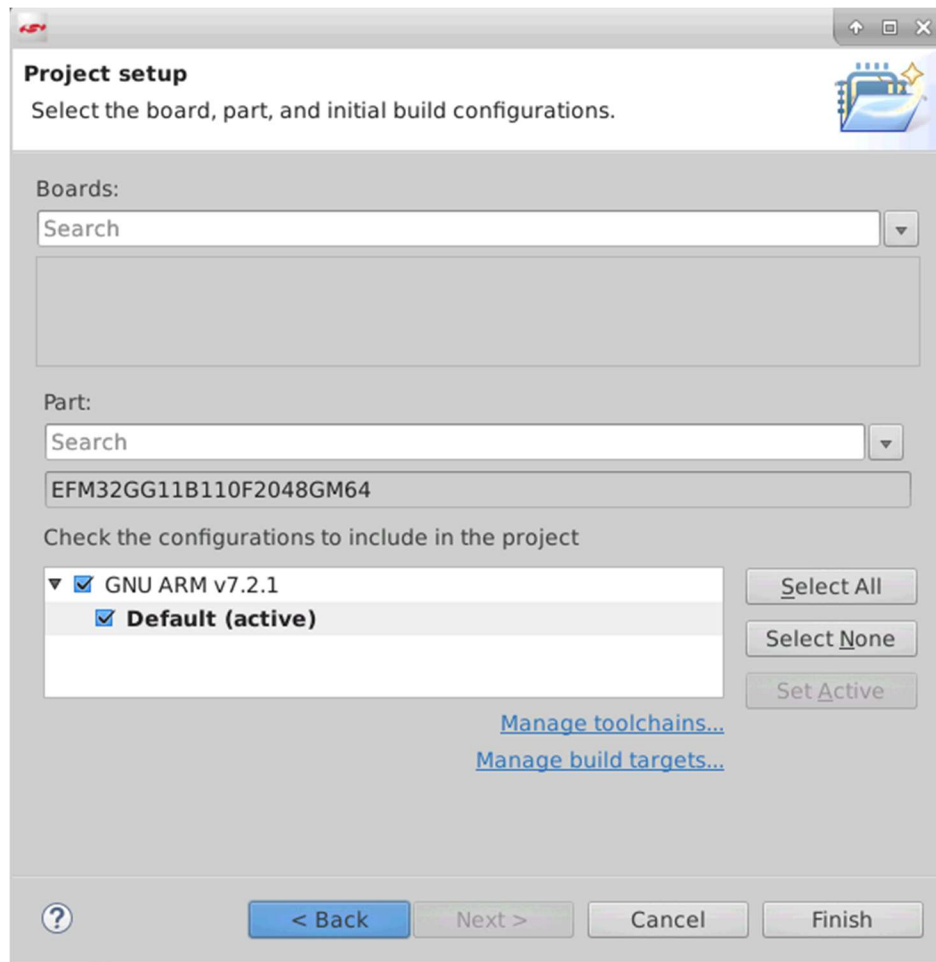


Figure 14: Import Finish

2.6.2. Configuring source

1. Configure bootloader source using [bootloader-storage-internal-single-2048k.isc](#)
2. Open file in [Simplicity studio](#) after importing by double clicking on .isc file from file explorer tab
3. Verify MCU details by referring to General tab in configuration tab

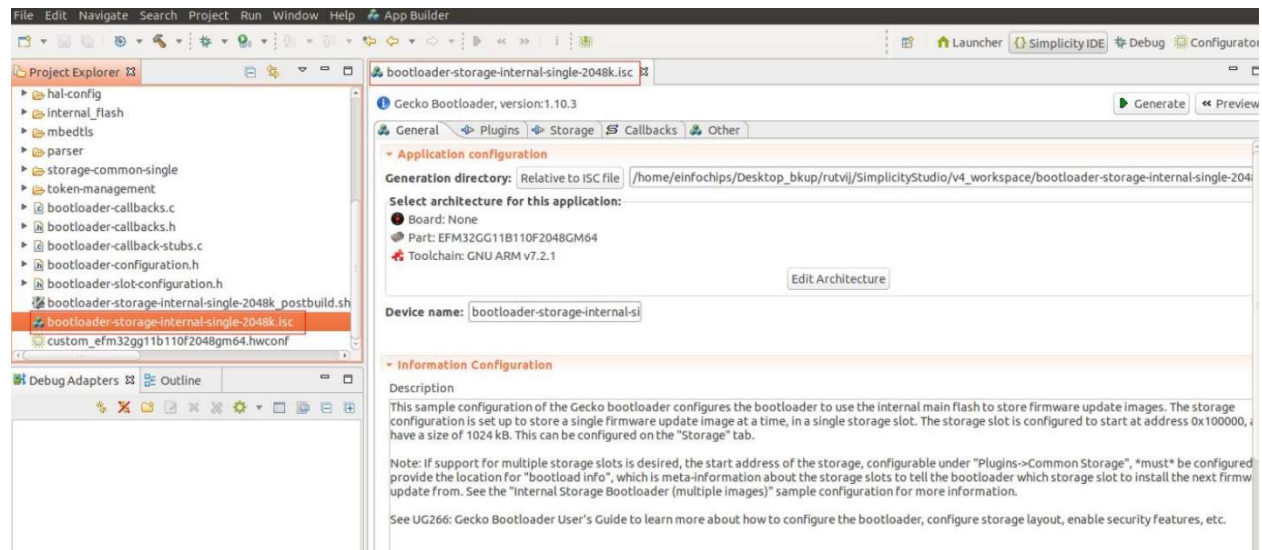


Figure 15: Generate - Bootloader

4. Go to **Plugins** -> Select **Bootloader Core**, provides API: **Core** -> **Options** -> Select below options
 - Require signed firmware upgrade files
 - Require encrypted firmware upgrade files
 - Enable secure boot option

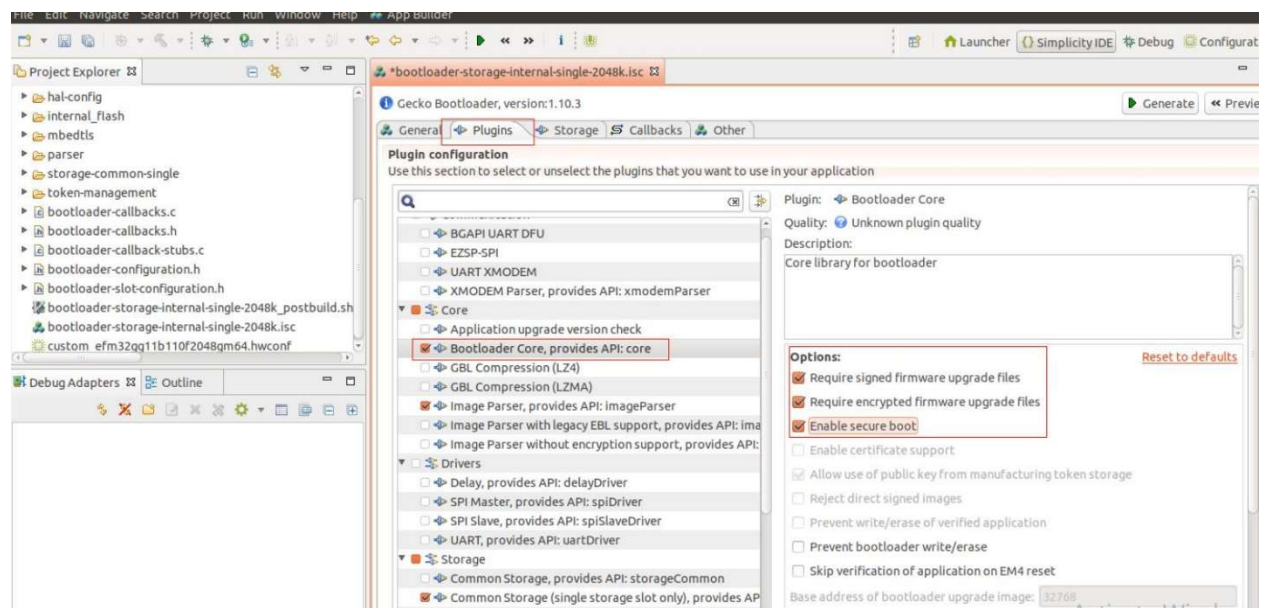


Figure 16: Plugin - Bootloader

5. Go to **Storage** tab and verify **slot 0** with **starting address 1048576**

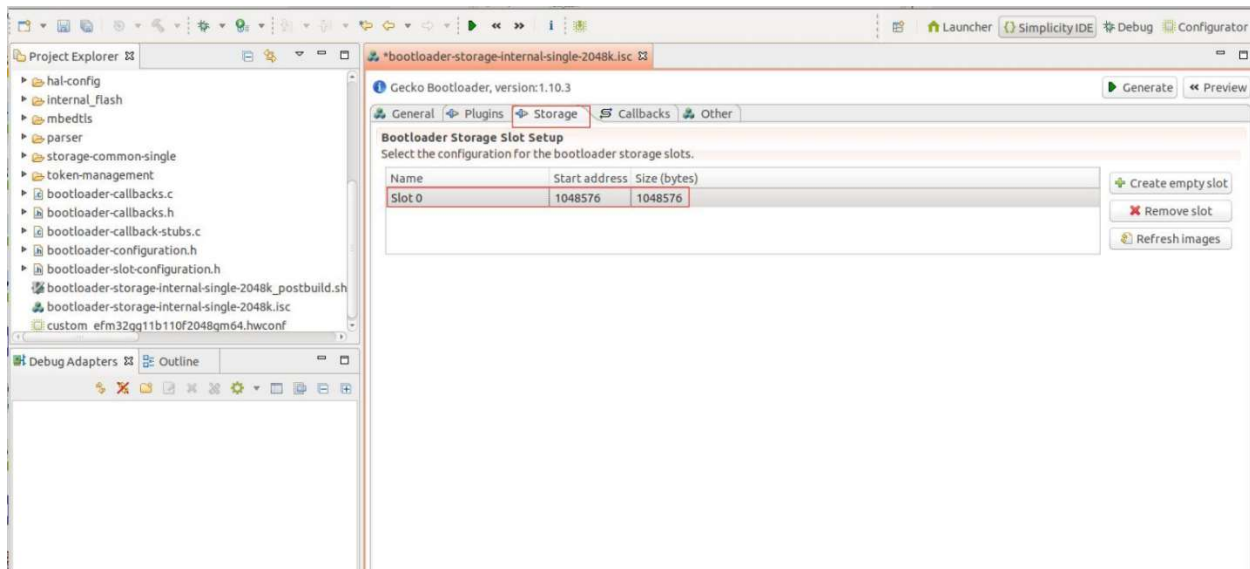


Figure 17: Storage configuration - Bootloader

6. Click **Green button with Generate from** right side top

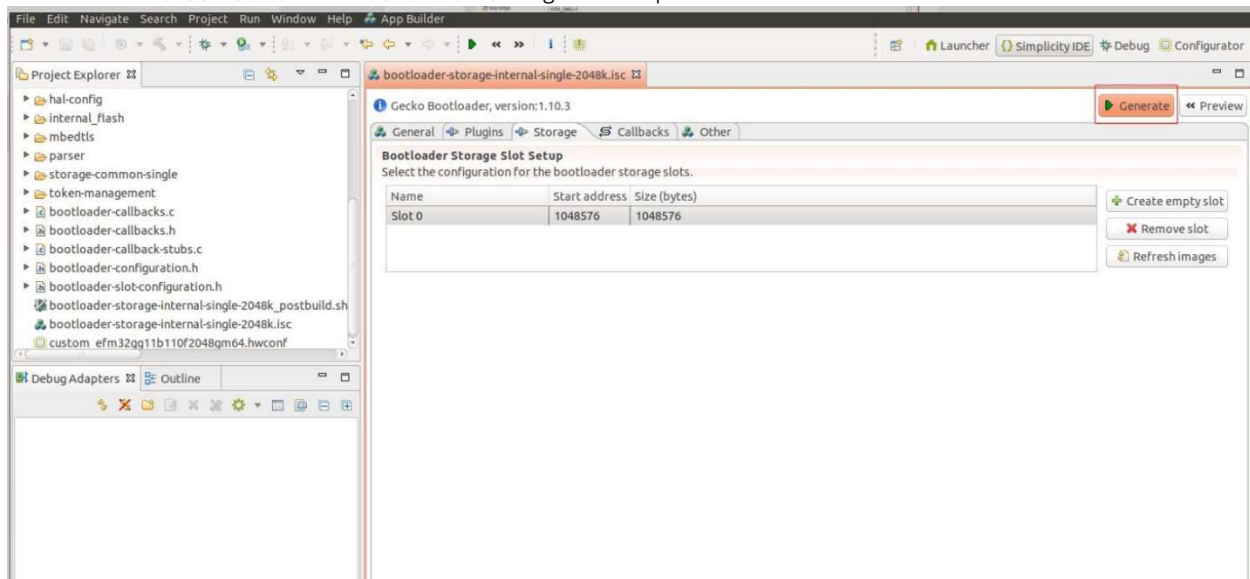



Figure 18: Generate the Bootloader .isc file

7. Verify generation successful

2.6.3. Building and Flashing

1. Build project by selecting 'Hammer'  option from tools panel
2. Check Console for successful build with file named ***bootloader-project-name-*combined.s37**

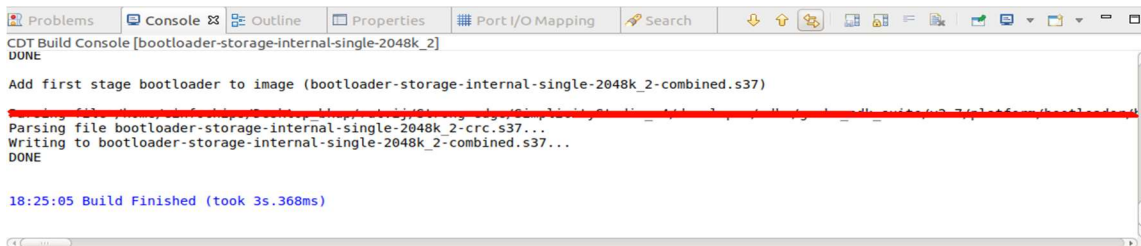


Figure 19: Gecko Bootloader compilation

3. From **project explorer** panel go to **Binaries** folder and verify binary generated

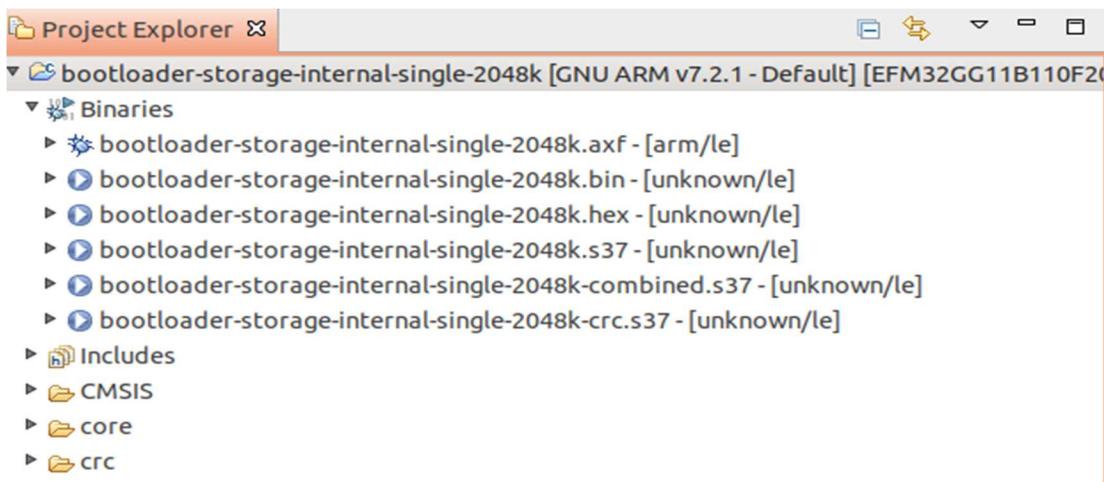


Figure 20: Gecko Bootloader binaries

4. Right click on binary named ***bootloader-project-name*-combined.s37** -> **Flash to Device...**
5. Select **Program** button from Flash Programmer window after verifying info. to flash bins into device

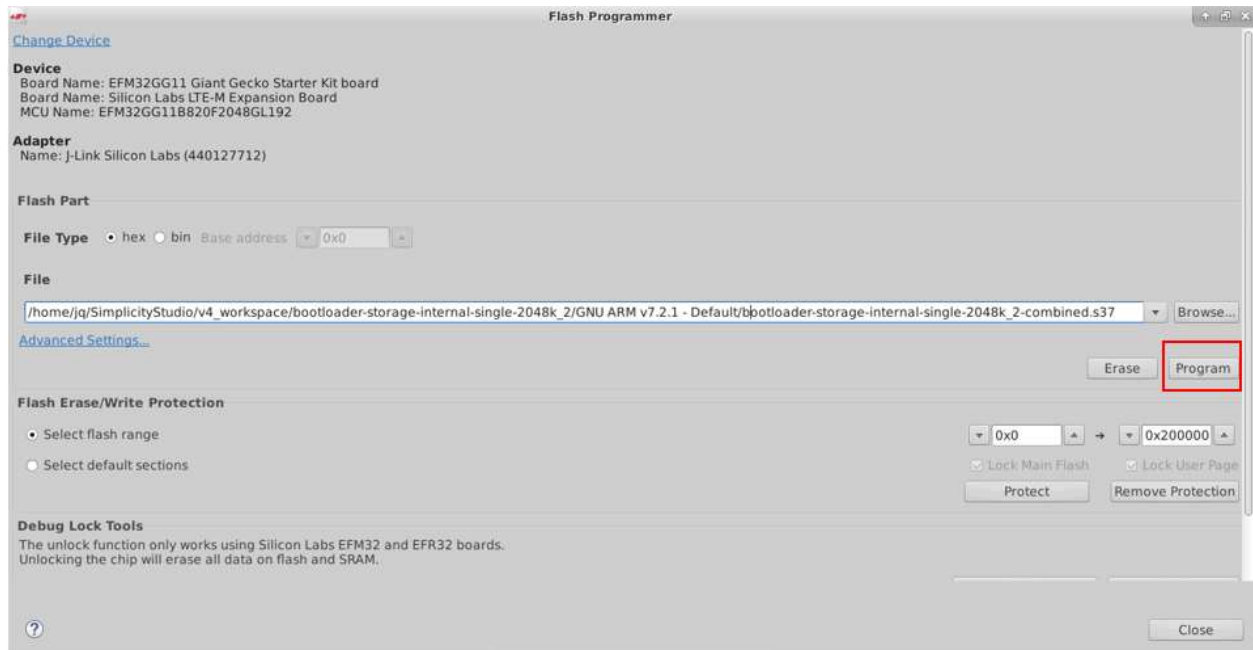


Figure 21: Flash Gecko Bootloader binaries

3. AMAZON FREERTOS APPLICATIONS

3.1. AWS Source

The Amazon FreeRTOS for GG11-LTE-M-SSK software package is an extension of Amazon FreeRTOS available at:

<https://github.com/aws/amazon-freertos>

It is provided as a Silabs_LTE-M_SSK_AWS_Rel01.zip with mandatory license acceptance. Two Amazon FreeRTOS applications are provided

[Note: aws_demos: provide various demos (MQTT, OTA ...) selected by compilation flags]

3.2. OPTIGA™ Trust M Source

The OPTIGA™ Trust M is a high-end security solution that provides an anchor of trust for connecting IoT devices to the cloud, giving every IoT device its own unique identity. This pre-personalized turnkey solution offers secured, zero-touch onboarding and the high performance needed for quick cloud access.

In provided package, OPTIGA™ Trust M security chip is configured. Software Framework is available at below location.

<https://github.com/Infineon/OPTIGA-trust-m>

To port software framework one needs to implement below file based on your Hardware platform.

- pal_ifx_i2c_config.c
- pal_i2c.c
- pal_gpio.c
- pal_os_timer.c
- pal_os_event.c

Refer below link for reference

<https://github.com/Infineon/OPTIGA-trust-m/wiki/Porting-Guide>

3.3. LTE-M Xbee®3 module source

The project is a collection of portable ANSI C code for communicating with Digi International's [XBee](#) wireless radio modules in API mode. You will typically compile/link the files directly into your application, instead of compiling them into a shared/dynamic-link library. Because of this design, most of the library configuration takes place at compile time.

In provided package, software C libraries framework is available at below location.

https://github.com/digidotcom/xbee_ansic_library/

xbee_ansic_library/ports/efm32/ has provided port for Silabs EFM32 starter kit with below files

1. platform_config.h
2. xbee_platform_efm32.c
3. xbee_serial_config_efm32.h
4. xbee_serial_efm32.c

4. BUILDING AMAZON FREERTOS SOURCE

4.1. Import the Source

1. Unzip Silabs_LTE-M_SSK_AWS_Rel01.zip file from downloaded in section 1.5
2. Open Simplicity Studio IDE
3. Go to **File -> Import...**
4. In **New project** search window browse, the following path and select **Next**
 - o <Zip location>/amazon-freertos/projects/silabs/efm32gg11_slstk3701a/SimplicityStudio_v4/aws_demo
5. Verify details in **configuration** tabs and click **Next**
6. Use default location and click **Finish**

4.2. Configure the source

1. Go to **Project explorer -> Select aws_demo -> Right click -> Properties... -> C/C++ Build -> Tools Settings tab -> GNU ARM C Compiler -> Symbols**

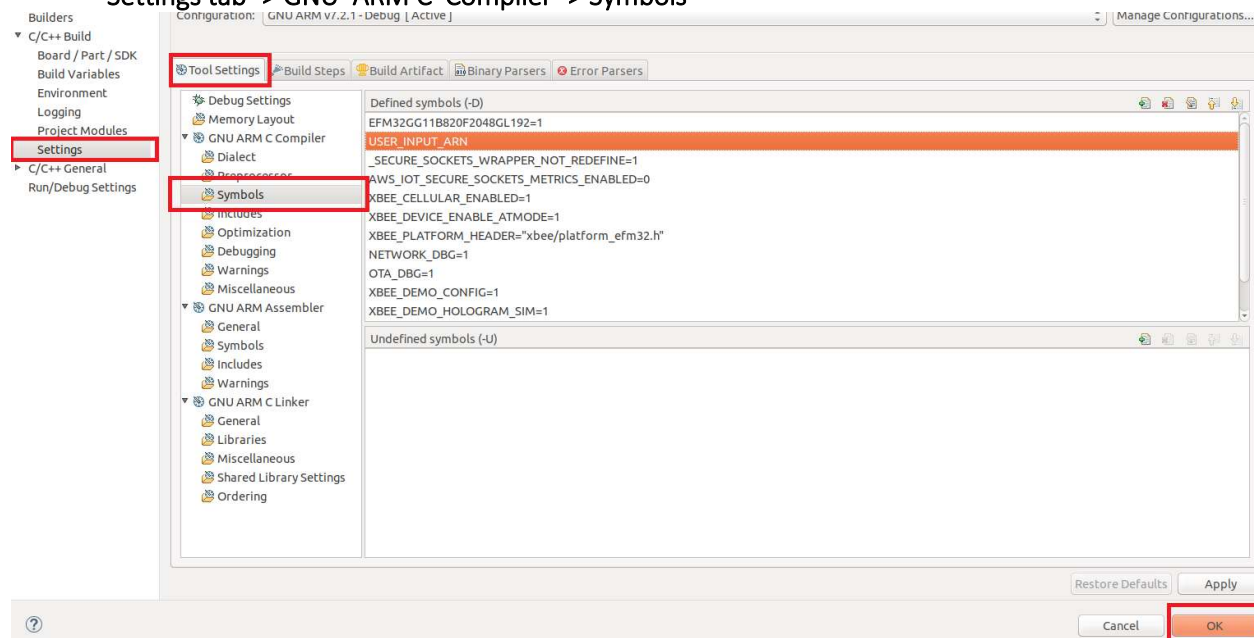


Figure 22: Symbol - Properties

2. Add/Modify Symbols from below references (Refer 7.1.3)

```
EFM32GG11B820F2048GL192=1
USER_INPUT_ARN
_SECURE_SOCKETS_WRAPPER_NOT_REDEFINE=1
AWS_IOT_SECURE_SOCKETS_METRICS_ENABLED=0
XBEE_CELLULAR_ENABLED=1
XBEE_DEVICE_ENABLE_ATMODE=1
XBEE_PLATFORM_HEADER="xbee/platform_efm32.h"
NETWORK_DBG=1
OTA_DBG=1
```

```

RETARGET_VCOM=1
XBEE_DEMO_CONFIG=1
XBEE_DEMO_HOLOGRAM_SIM=1
XBEE_CHANGE_APN=1
XBEE_TARGET_APN="hologram"

```

Figure 23: aws_demo project config.

4.3. Application Configuration

Set the AWS configurations in sources:

Log on to AWS web site (<https://aws.amazon.com/console/>), browse to the AWS IoT Core service.


1. In the left navigation pane, choose **Settings**. Your AWS IoT endpoint is displayed in **Endpoint**. It should be like `*<accountID>*-ats.iot.*<region>*.amazonaws.com`. Record the endpoint. This is required when you start the board for first time.
2. In the left navigation pane, choose **Manage > Things**. Your device should have an AWS IoT thing name. Record this name.
3. Use your AWS IoT thing name to open `demos/include/aws_clientcredential.h` in IDE, and specify values for the following #define constant:
4. **clientcredentialIOT_THING_NAME** The AWS IoT thing name of your board. Ex: `Silabs_Xbee3_LTE` (created in selection 2.4.1)
5. For OTA: Replace **signingcredentialSIGNING_CERTIFICATE_PEM** variable value in `demos/include/aws_ota_codesigner_certificate.h` with created certificate content (beginning with `-----BEGIN CERTIFICATE-----`);
6. To create your code-signing certificate refer below link.
<https://docs.aws.amazon.com/freertos/latest/userguide/ota-code-sign-cert-win.html>
7. Configure Demo for either MQTT or OTA by defining either of below macro
#define CONFIG_MQTT_DEMO_ENABLED
Or
#define CONFIG_OTA_UPDATE_DEMO_ENABLED
By editing `vendors/silab/boards/efm32gg11_slstk3701a/aws_demos/config_files/aws_demo_config.h`

4.4. Sim card activation

1. Global Hologram SIM card is used with AT&T connectivity carrier,

2. To activate and use Hologram SIM card refer [Quick_Start_Guide_Silabs_LTE-M_SSK_v01.docx](#)
Section 2.2.3 Activate the included SIM card

4.5. Building and Flashing

1. Once done with configuration, build the aws_demos project using **build**  symbol.
2. Go to [amazon-freertos/projects/silabs/efm32gg11_slstk3701a/SimplicityStudio_v4/aws_demo/GNU ARM v7.2.1 - Debug](#)
3. Verify generated binary **aws_demo.s37**
4. Go to the directory/location where commander is installed and use below commands to sign and encrypt generated binaries
 - a. `$./commander gbl keygen --type ecc-p256 --outfile signing-key`
 - b. `$./commander gbl keygen --type aes-ccm --outfile encryption-key`
 - c. `$./commander flash --tokengroup znet --tokenfile encryption-key --tokenfile signing-key-tokens.txt`
 - d. `$./commander convert aws_demo.s37 --secureboot --keyfile signing-key --outfile aws_demo-signed.s37`
 - e. `$./commander gbl create aws_demo.gbl --app aws_demo-signed.s37 --sign signing-key --encrypt encryption-key`

NOTE: Commander is not able to parse .gbl while flashing so use signed image, gbl file can be used for OTA operation

5. Go to the directory/location where commander is installed and use below command
 - o `$./commander`
 - o Go to Simplicity commander GUI verify part number in **j-link Device section** I.e. S/N <Part number of j-link>
 - o Click on **connect** in adapter section
 - o Once connected - verify MCU hardware information and select Connect in **Target** section with SWD Freq. 8000 KHz
 - o Click on **Flash** GUI button



Figure 24: Commander Flash button

6. Go to **Flash MCU -> Binary File -> browse...** -> select the **aws_demo-signed.s37** file from binary location and **flash** the binary with shown config enabled.

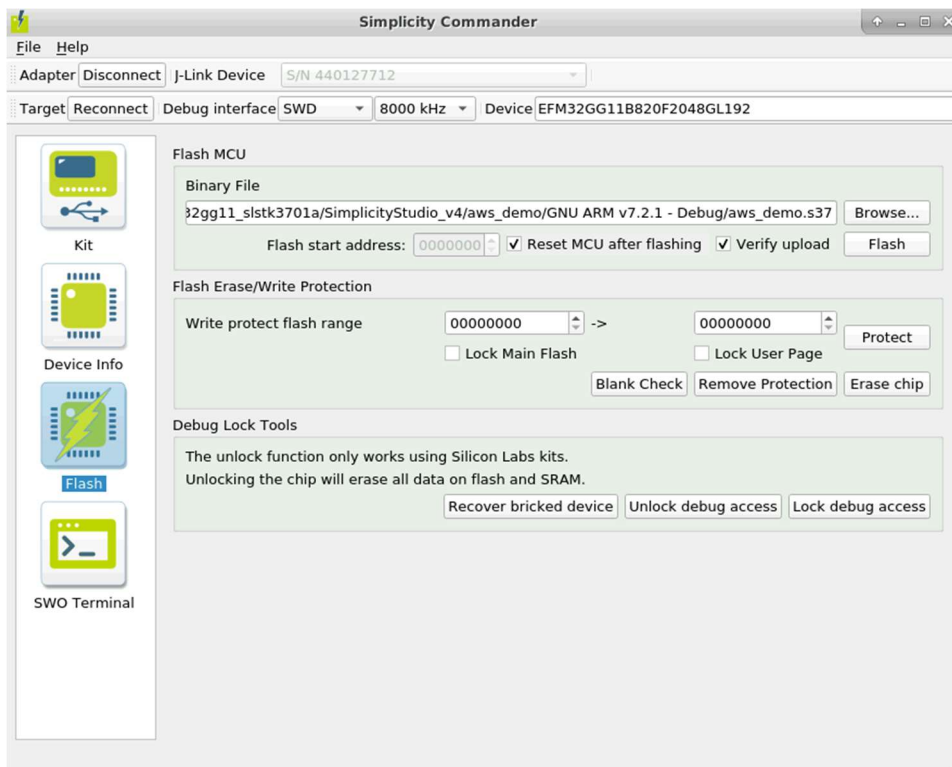


Figure 25: Commander Flash tool

4.6. Acquire Device Certificate

1. Open putty/minicom terminal of GG11-LTE-M-SSK
2. Press User button 0 and power on the board
3. Run the application demo as mentioned in 4.1.5
4. Copy terminal text starting from -----BEGIN CERTIFICATE----- to -----END CERTIFICATE----- in text file as deviceCert.pem file

```

Wait for cell signal
.....
.....
.....
.....
cell signal available
-----BEGIN CERTIFICATE-----
MIT2226...WIBAgIEbfa1qDAKBggqhkJOPQQDAjByMQswCQYDVQQGEWJERTEh
MB8GA1UECgwYSW5maW5lb24gVG9jaG5vbG9naWVzIEFHMRMwEQYDVQQL
R0EoVE0phsSwRQ10VXG9naWVzIEFHMRMwEQYDVQQLIE0gQ0Eg
MTAxMB4XDTE5MDYxODA2MjgyM1oXDTM5MDYxODA2MjgyM1owHDEaMBG
jA1UEAwR
SW5maW5lb24gSW9UIE5vZGUwWTATBgqhkJOPQMBBwCAASz1W99
FiKdtwISAEUdz9+610NcUllqt3c3...CxhwbFRqW0AKwq9NHUIdbr
i6DoYV72Nl
BeVwZRp...AUBgNVHQ8BAf8EBAMCAIAwDAYDVR0TAH/BAIwADAV
BgnVHSAEDjAMMAoGCCqCFABEARQ...GA1UdIwQYMBaAFDwwjFzV
jIjXTKA5FSD
sv/Nhk0jMAoGCCqGSM49BAMCA0GAMEUCIQ...5V72Nl1qSlKM
0Y6oArSwyZr
nZ7LzUrqp7LoYwIgaeIKz82jhKr0YLDpnu0PquDSR558FkdAD...MscF/M=
-----END CERTIFICATE-----

----- DEMO RUNNER -----

```

Figure 26: OPTIGA™ Trust M certificate reading

4.7. Verify XBee®3 LTE-M Module's info

Run MQTT or OTA demo and check on terminal console for logs and check print for **Xbee3** info.

```
Silabs Xbee3 LTE-M AWSFreeRTOS demo 2020 version 0.0.1
Please Enter Your Endpoint ARN
e.g- xxxxxxxxxxxx-ats.iot.xx-xxx-1.amazonaws.com
Endpoint ARN_URL = a3vwfgdm06d0xb-ats.iot.ap-south-1.amazonaws.com

Xbee LTE-M init done !

Initializing
.....

Successful initialization of Xbee Done

Attempting to apply
the new APN
The APN was already
set correctly!

Xbee3 info : Hardware Version:4b42, Firmware Version:11415, Baud Rate:115200, Connection:1.
```

Figure 27: Xbee3 info

5. SAMPLE LOGS

5.1. MQTT Demo Logs

```
[INFO ][MQTT][lu] (MQTT connection 0x2000a3d0) MQTT PUBLISH operation queued.[INFO ][DEMO][lu] Acknowledgment message for PUBLISH 3 will be sent.
Incoming PUBLISH received:
Subscription topic filter: LTEiotdemo/topic/3
Publish topic name: LTEiotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: LTE-M MQTT Message Count 2

[INFO ][MQTT][lu] (MQTT connection 0x2000a3d0) MQTT PUBLISH operation queued.[INFO ][DEMO][lu] Acknowledgment message for PUBLISH 2 will be sent.
Incoming PUBLISH received:
Subscription topic filter: LTEiotdemo/topic/1
Publish topic name: LTEiotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: LTE-M MQTT Message Count 4

[INFO ][MQTT][lu] (MQTT connection 0x2000a3d0) MQTT PUBLISH operation queued.[INFO ][DEMO][lu] Acknowledgment message for PUBLISH 4 will be sent.
Incoming PUBLISH received:
Subscription topic filter: LTEiotdemo/topic/2
Publish topic name: LTEiotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: LTE-M MQTT Message Count 5
```

Figure 28: MQTT Logs

5.2. OTA Demo Logs

```
[INFO ][MQTT][lu] (MQTT connection 0x2000a948, SUBSCRIBE operation 0x2000d4c0) Wait complete with r[prvSubscribeToJobNotificationTopics] OK: $aws/things/LTE_Thing/jobs/
notify-next
[prvRequestJob_Mqtt] Request #0
[INFO ][MQTT][lu] (MQTT connection 0x2000a948) MQTT PUBLISH operation queued.[INFO ][MQTT][lu] (MQTT connection 0x2000a948, PUBLISH operation 0x2000d4c0) Waiting for op
eration [INFO ][DEMO][lu] State: RequestingJob Received: 0 Queued: 0 Processed: 0 Dropped: 0
[INFO ][MQTT][lu] (MQTT connection 0x2000a948, PUBLISH operation 0x2000d4c0) Wait complete with res[prvOTAAgentTask] Called handler. Current State [RequestingJob] Event
[RequestJobDocument] New stat[INFO ][MQTT][lu] subscription callback 460[prvParseJobDoc] Size of OTA_FileContext_t [64]
[prvParseJSONbyModel] Extracted parameter [ clientToken: 0:LTE_Thing ]
[prvParseJSONbyModel] Extracted parameter [ jobId: AFR_OTA-aws_demo_1_2_0 ]
[prvParseJSONbyModel] Extracted parameter [ protocols: ["MQTT"] ]
[prvParseJSONbyModel] Extracted parameter [ streamname: AFR_OTA-aaade69e-d7db-4c28-a676-5086840d7f8[prvParseJSONbyModel] Extracted parameter [ filepath: firmware.bin ]
[prvParseJSONbyModel] Extracted parameter [ filesize: 235064 ]
[prvParseJSONbyModel] Extracted parameter [ fileId: 0 ]
[prvParseJSONbyModel] Extracted parameter [ certfile: /home/desktop/bins ]
[prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: MEUCIF9idfJhpMnJgIWGRo2kLf+2WVf/... ][prvParseJobDoc] Job was accepted. Attempting to start transfer.

Current Silabs APP version: 1
```

Figure 29: OTA Logs

6. AWS DEMOS

6.1. General description

AWS endpoint and device name

File: demos/include/aws_clientcredential.h

Macro: #define clientcredentialIOT_THING_NAME "Silabs_Xbee3_LTE"

Aws_demos is a multi-demo application: MQTT, HTTP, and OTA Configuration files:

See <https://docs.aws.amazon.com/freertos/latest/userguide/freertos-configure.html>

Example demo is configured in

File: vendors/silab/boards/efm32gg11_slstk3701a/aws_demos/config_files/aws_demo_config.h

Example:

#define CONFIG_MQTT_DEMO_ENABLED

6.2. MQTT demo

6.2.1. Source config

Make sure aws_demos are configured for MQTT demo in config_files/aws_demo_config.h in Simplicity Studio (or vendors/silab/boards/efm32gg11_slstk3701a/aws_demos/config_files/aws_demo_config.h):

#define CONFIG_MQTT_DEMO_ENABLED

[**Note:** In addition, all the other **CONFIG_*_ENABLED** flags are commented]

6.2.2. Run MQTT Demo

1. If you want to modify the Endpoint URL, then follow the step 2 otherwise Reset the board and jump to step 3 as shown below figure.
2. Press the USER 0 button and Reset the board it will ask to enter endpoint URL on Serial console screen as below figure

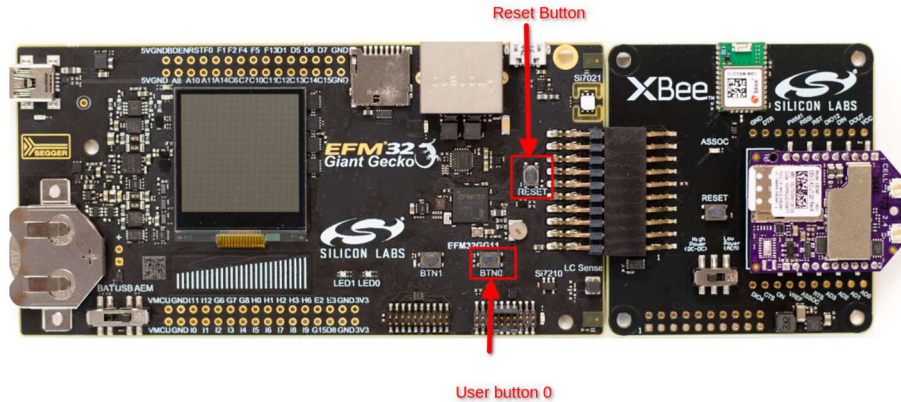


Figure 30: Reset Button and User Button

- Paste the Endpoint URL i.e. xxxxxx-ats.iot.xx.xxxx-x.amazonaws.com here and press the “Enter” key on PC keyboard. (Note: Default Echo for terminal is OFF, so cannot see your Endpoint URL value until turned ON for your terminal)
- So, to verify if correct endpoint URL was entered, on the serial console it will display the URL as shown above (with yellow marked).

```
Silabs Xbee3 LTE-M AWSFreeRTOS demo 2020 version 0.0.1
Please Enter Your Endpoint ARN
e.g- xxxxxxxxxxxx-ats.iot.xx-xxx-1.amazonaws.com
Endpoint ARN_URL = arn:aws:iot:us-east-2:amazonaws.com
```

- To see the MQTT message on AWS console, Open AWS Console >> IoT Service >> Test page and Subscribe the Topic: **LTEiotdemo/topic/#**

Figure 31: MQTT Logs on AWS console

- MQTT Message will display on the screen i.e. **LTE MQTT Message Count 2**

```
[INFO ][MQTT][lu] (MQTT connection 0x2000a3d0) MQTT PUBLISH operation queued.[INFO ][DEMO][lu] Acknowledgment message for PUBLISH 3 will be sent.
Incoming PUBLISH received:
Subscription topic filter: LTEIotdemo/topic/3
Publish topic name: LTEIotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: LTE-M MQTT Message Count 2

[INFO ][MQTT][lu] (MQTT connection 0x2000a3d0) MQTT PUBLISH operation queued.[INFO ][DEMO][lu] Acknowledgment message for PUBLISH 2 will be sent.
Incoming PUBLISH received:
Subscription topic filter: LTEIotdemo/topic/1
Publish topic name: LTEIotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: LTE-M MQTT Message Count 4

[INFO ][MQTT][lu] (MQTT connection 0x2000a3d0) MQTT PUBLISH operation queued.[INFO ][DEMO][lu] Acknowledgment message for PUBLISH 4 will be sent.
Incoming PUBLISH received:
Subscription topic filter: LTEIotdemo/topic/2
Publish topic name: LTEIotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: LTE-M MQTT Message Count 5
```

Figure 32: MQTT logs on pc terminal

6.3. OTA demo

6.3.1. Source config

1. The aws_demos application configured for OTA demonstrates an Over the Air update of the application firmware. The new firmware is downloaded via LTE-M connectivity to the SLSTK3701A Giant gecko device and installed securely with the Giant gecko bootloader.
2. Make sure aws_demos are configured for OTA demo in config_files/aws_demo_config.h in Simplicity Studio IDE
or
3. vendors/silab/boards/efm32gg11_slstk3701a/aws_demos/config_files/aws_demo_config.h:
 - `#define CONFIG_OTA_UPDATE_DEMO_ENABLED`

[Note: In addition, all the other `CONFIG_*_ENABLED` flags are commented]

4. Please ensure you have configured the aws_demos (especially the code signing certificate in demos/include/aws_ota_codesigner_certificate.h).
5. To create the code-signing certificate refer below link.

<https://docs.aws.amazon.com/freertos/latest/userguide/ota-code-sign-cert-win.html>
6. To do an OTA update, a new version of the application must be compiled in a .gbl file and uploaded on S3 file server.
7. Flash the current version of the application to the SLSTK3701A Giant gecko board. Then increase the Amazon FreeRTOS application version in `demos/include/aws_application_version.h` and build a new binary with Simplicity studio. The installable .gbl firmware is under GNU ARM v7.2.1 - Debug/ directory (GNU ARM v7.2.1 - Debug/aws_demos.gbl file). It must be copied to the S3 bucket.
8. The aws_demos project must be running on SLSTK3701A Giant gecko board and be connected cellular.

6.3.2. Run OTA Demo

1. On AWS console, create a FreeRTOS OTA update job (IoT Core / Manage / Jobs / Create):

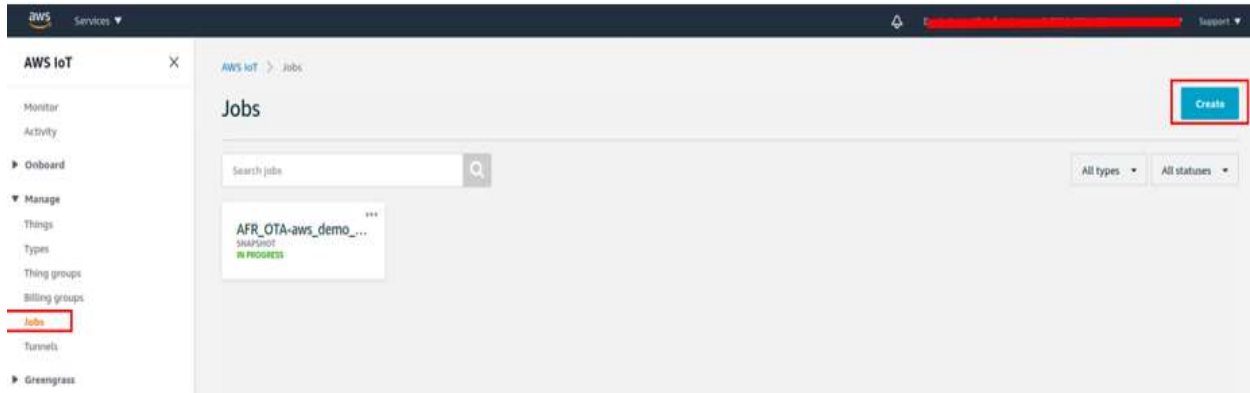


Figure 33: OTA Job creation

2. Select the **create OTA update job**

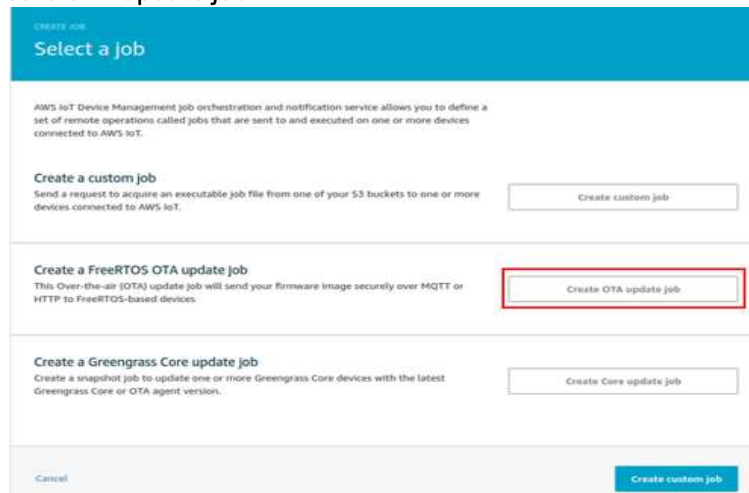


Figure 34: OTA job selection

3. Select the 'thing' to update. Ex: Silabs_Xbee3_LTE

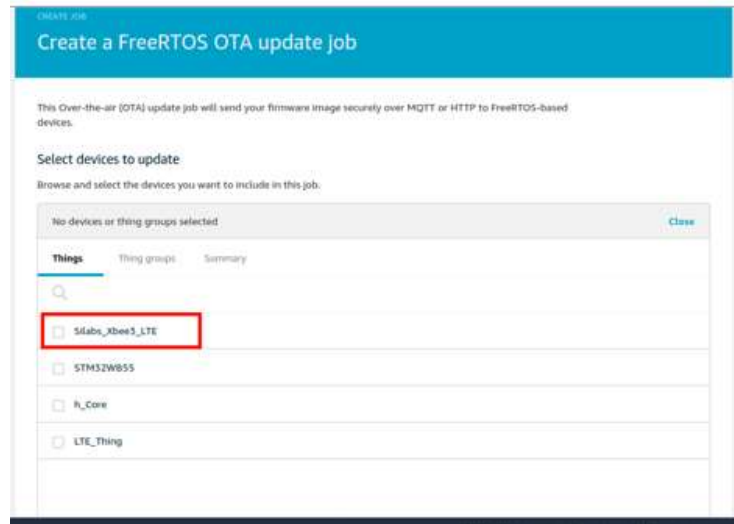


Figure 35: OTA Selecting Thing

4. The MQTT protocol for image transfer, (HTTP is not supported)
5. "Sign a new firmware for me",
6. Your .gbl firmware file, referenced from the S3 bucket where you have uploaded it, any non-empty destination pathname (e.g. firmware.bin)

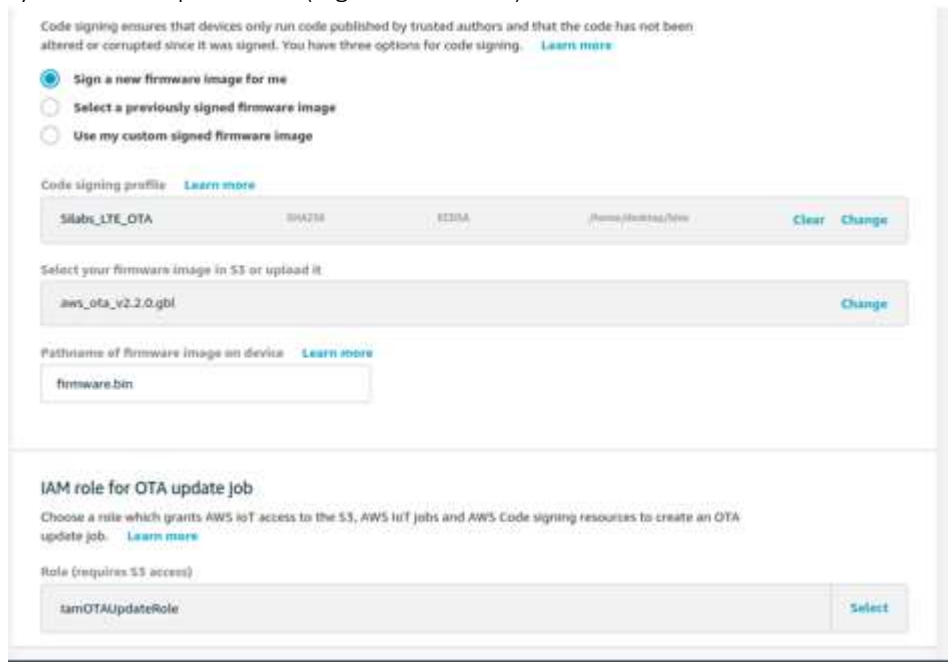



Figure 36: OTA configuring role

7. Select the IAM role for OTA update Job which was created in section 2.4.2 and click on **Next**
8. Finally choose a unique job ID and click **create** button

CREATE JOB

Create a FreeRTOS OTA update job

ID

Update_ver3 

Description (optional)

Give your job a helpful description

Job type

A job can run on the devices and/or groups selected, or remain open, and apply to devices later added to a group.

☒ Your job will complete after deploying to the selected devices/groups (snapshot)

☐ Your job will continue deploying to any devices added to the selected groups (continuous)

Job executions rollout configuration - optional

Specify how quickly devices will be notified of a pending job execution. [Info](#)

☒ Constant rate

☐ Exponential rate

Maximum per minute (1-1000)

1000

Figure 37: OTA job naming

9. Create code signing profile, Enter Profile Name (e.g. ssk_code_sign) , Device Hardware Platform select Windows Simulator, code signing certificate press “Import” and browse the created code signing certificate and private key , Enter the pathname of code signing certificate on device (e.g. ecdsa.crt).

Create a code signing profile

Profile name
ssk_code_sign

Device hardware platform
Windows Simulator SHA256 ECDSA [Change](#)

Code signing certificate
AWS Certificate Manager (ACM) handles the complexity of creating and managing or importing SSL/TLS certificates. You use ACM to create an ACM Certificate or import a third-party certificate that you use for signing. You must have a certificate to sign code.

No certificate selected [Close](#)

Select Certificate
[Browse...](#) ecdsasigner.crt

Select Certificate private key
[Browse...](#) ecdsasigner.key

Select Certificate chain (optional)
[Browse...](#) No file selected.

[Import](#)

Pathname of code signing certificate on device
This is the platform-specific location and name of the certificate used by the FreeRTOS device firmware to perform OTA image signature verification.
ecdsa.crt

[Cancel](#) [Create](#)

Figure 38: OTA Code signing certificate adding

10. Successful message will be displayed.
11. If you want to modify the Endpoint URL, then follow the step 2 otherwise Reset the board and jump to step 3.
12. Press the USER 0 button and Reset the board It will ask to enter endpoint URL on Serial console screen as below.
13. Paste the Endpoint URL here and press the “Enter” key on keyboard. (Note: Default Echo is OFF, so cannot see your Endpoint URL value.)
14. So, to verify that the correct endpoint URL was entered, on the serial console it will display the URL as shown above (with yellow marked).

```
Silabs Xbee3 LTE-M AWSFreeRTOS demo 2020 version 0.0.1
Please Enter Your Endpoint ARN
e.g- xxxxxxxxxxx-ats.iot.xx-xxx-1.amazonaws.com
Endpoint ARN_URL = arn:aws:iot:us-west-2:123456789012:certificate/12345678-9012-3456-7890-123456789012-ats.iot.us-west-2.amazonaws.com
```

15. OTA will start and will be downloading new firmware

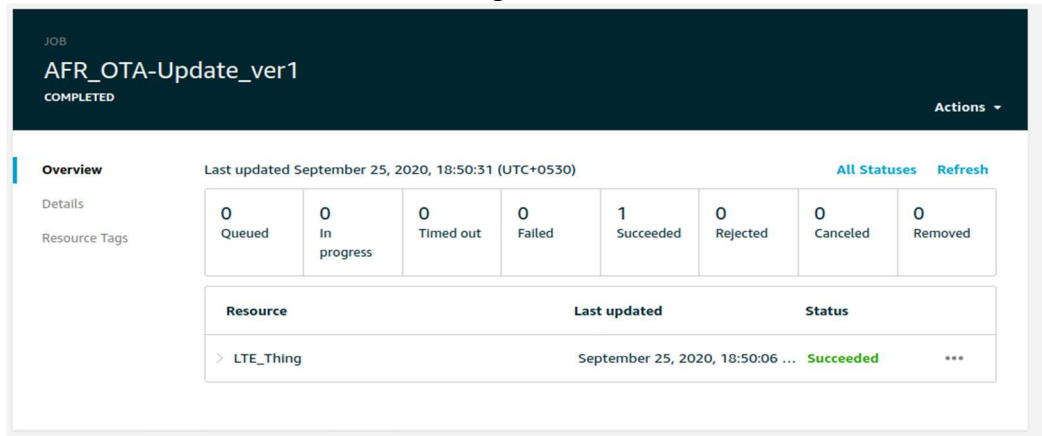


Figure 39: OTA Successful log on AWS console

16. OTA Job and its packets are as shown in the screen below:

```
[prvParseJSONbyModel] Extracted parameter [ clientToken: 0:LTE_Thing ]
[prvParseJSONbyModel] Extracted parameter [ jobId: AFR_OTA-aws_demo_1_2_0 ]
[prvParseJSONbyModel] Extracted parameter [ protocols: ["MQTT"] ]
[prvParseJSONbyModel] Extracted parameter [ streamname: AFR_OTA-aaade69e-d7db-4c28-a676-5086840d7f8[prvParseJSONbyModel] Extracted parameter [ filepath: firmware.bin ]
[prvParseJSONbyModel] Extracted parameter [ filesize: 235064 ]
[prvParseJSONbyModel] Extracted parameter [ fileId: 0 ]
[prvParseJSONbyModel] Extracted parameter [ certfile: /home/desktop/bins ]
[prvParseJSONbyModel] Extracted parameter [ sig-sha256-ecdsa: MEUCIF9idfJhpMmJgIWGRo2kLf+2WVf/... ][prvParseJobDoc] Job was accepted. Attempting to start transfer.
Current Silabs APP version: 1
```

Figure 40: OTA Logs on pc terminal

17. Verify the received Firmware and its signature as shown in screen below

```
OTA Signature DONE !!!
[prvIngestDataBlock] File receive complete and signature is valid.
[prvStopRequestTimer] Stopping request timer.
[prvUpdateJobStatus_Mqtt] Msg: {"status":"IN_PROGRESS","statusDetails":{"self_test":"ready","update[INFO ][MQTT][Lu] (MQTT connection 0x2000a948) MQTT PUBLISH operation
```

Figure 41: OTA Verification

18. On Next reboot - Board will power up with newly upgraded firmware

7. MODIFYING SOURCE

7.1. Source files

1. amazon-FreRTOS/demos/mqtt/iot_demo_mqtt.c
 - IOT_DEMO_MQTT_TOPIC_PREFIX – can modify AWS MQTT topic for subscription and publish
 - publishInfo.pPayload - can be used to send customized messages over MQTT to AWS
2. amazon-FreRTOS/demos/demo_runner/iot_demo_freertos.c
 - In runDemoTask() credentials can be set to compile time certifications instead of using OPTIGA™ Trust M pre provisioned certificates
 - In above modification amazon-FreRTOS/demos/include/aws_clientcredential_keys.h also needs to supply proper certificates
 - **NOTE:** OPTIGA™ Trust M still be used in pkcs11
3. amazon-FreRTOS/demos/include/aws_application_version.h
 - #define APP_VERSION_* - Macro specifying application version
4. amazon-freertos/libraries/3rdparty/optiga-trust-m/pal/EFM32GG11/pal*.c
 - Platform layer for OPTIGA™ Trust M communication
5. amazon-freertos/libraries/3rdparty/xbee/ports/efm32/xbee_serial_config_efm32.h
 - #define macros to modify communication and configuration of USART interface which communicates with XBee®3 LTE-M

7.2. Configuration macros in IDE Settings

1. USER_INPUT_ARN - if defined , user can provide Endpoint of AWS from minicom/putty console at runtime else clientcredentialMQTT_BROKER_ENDPOINT from aws_clientcredentials.h
2. XBEE_DEMO_CONFIG, XBEE_DEMO_HOLOGRAM_SIM, XBEE_CHANGE_APN, XBEE_TARGET_APN – These macros will set APN name " hologram " for cellular LTE-M connectivity, for changing APN XBEE_TARGET_APN can be modified to provide string APN.

7.3. Enable logging

From IDE Settings

- NETWORK_DBG - define and set 1 to enable iot secure socket APIs logs
- OTA_DBG – define and set 1 to enable MQTT AWS OTA logs

From Source file

- Enable MBEDTLS_DEBUG_C macro from mbedtls/config.h for TLS logs

AWS Logging

- Include iot_logging_setup.h in source file where logging is required

- Before including `iot_logging_setup.h`, the constants `LIBRARY_LOG_LEVEL` and `LIBRARY_LOG_NAME` must be defined.
- After including `iot_logging_setup.h`, the logging functions can be used as per below

```
lotLogError( "Error." );  
lotLogWarn( "Warning. " );  
lotLogInfo( "Info." );  
lotLogDebug( "Debug." );
```

8. TROUBLESHOOTING

- Stuck at **Initializing ...** - Xbee®3 SLSTK3701A kit not connected properly with MCU
- Stuck at **Wait for cell signal ...** - Antenna not connected, Sim card not connected, Sim card not activated, Low coverage
- Stuck at **Resolving FQDN ...** - Invalid Endpoint ARN supplied, Low coverage
- While performing OTA Stops receiving data - Low signal or coverage, reinitiate OTA
- XBEE_TX_DELIVERY=0x89 - LTE-M socket resource busy, retry
- OTA status not updating - update APP_VERSION_* macros from aws_application_version.h properly
- Timeout waiting on semaphore 0xxxxx error received - too much traffic in MQTT publishing, reconfigure in `iot_demo_mqtt.c`
- Stuck at while (OPTIGA_LIB_BUSY == optiga_lib_status) loop - `pal_os_event_register_callback_oneshot` not receiving properly, check MbedTLS version compatibility with OPTIGA™ Trust M
- Stuck at `xDestroyProvidedObjects()` or cannot read OPTIGA™ Trust M at any OID - check metadata while performing any `optiga_util_write_metadata()` operation
- Stuck randomly after Enabling every possible log – memory error, verify `FreeRTOSConfig.h` file or lower the logs

9. REFERENCES

9.1. Reference Documents

FreeRTOS Porting Guide	AWS FreeRTOS porting Guide Latest
FreeRTOS User Guide	AWS FreeRTOS User Guide Latest
OPTIGA™ Trust-M porting guide	Trust-M Porting Guide Latest
XBee® User Guide	Cellular User Guide- Revision 90002253
QSG156: :LTE-M Quick Start Guide	Rev. 0.3
UG266: Silicon Labs Gecko Bootloader User's Guide	Rev. 1.3
UG162: Simplicity Commander Reference Guide	Rev. 1.8
QSG149: EFM32GG11-SLSTK3701A Quick-Start Guide	Rev. 0.1

[1] <https://aws.amazon.com/freertos/>

[2] <https://github.com/Infineon/OPTIGA-trust-m>

[3] <https://www.infineon.com/cms/en/product/security-smart-card-solutions/OPTIGA-embedded-security-solutions/OPTIGA-trust/OPTIGA-trust-m-sls32aia/>

[4] <https://www.silabs.com/development-tools/mcu/32-bit/efm32gg11-starter-kit>

[5] <https://docs.aws.amazon.com/freertos/latest/lib-ref/c-sdk/logging/index.html>