AN1203: HTTP(S) Gecko Bootloader

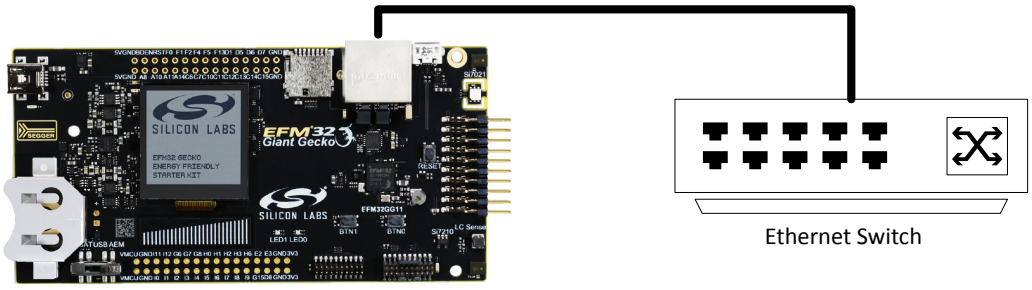
This application note is intended for users of the HTTP(S) bootloader. The HTTP(S) bootloader allows user to program the EFM32GG11 device's firmware without using a debugger.

The HTTP(S) bootloader offers application and bootloader updates using HTTP(S) protocol. It is an application bootloader that requests an GBL upgrade image from the server, which is then programmed to flash by the Gecko Bootloader. The Gecko Bootloader uses the information on the GBL file to determine if the update is for the application, the bootloader or both. This type of bootloader is currently only supported in EFM32GG11 since it requires an ethernet connection. The Gecko Bootloader resides in a reserved memory in flash so there is no destructive write mode for this bootloader. The bootloader can be disabled using the bit in the lock page.

A full GBL image is required in the bootloader area of the device prior to using the HTTP(S) bootloader.

KEY POINTS

- This bootloader can perform firmware upgrade using HTTP(S) protocol.
- GBL file is requested by the application and is responded by the server
- No destructive write is offered in this bootloader, however, reprogramming the bootloader is allowed.
- A full GBL image must be present in the bootloader area of the device prior to using the HTTP(S) bootloader.
- This bootloader only works on EFM32GG11 currently.



1. Device Compatibility

This application note supports the following MCU Series 1 device families.

MCU Series 1:

- EFM32 Giant Gecko (EFM32GG11)

2. Bootloader Documentation and Software Modules

The documents in Table 2.1 Bootloader Documentation on page 3 are available on https://www.silabs.com/support/resources.

Application Notes and User's Guides can also be accessed in Simplicity Studio using the [**Application Notes**] and [**User's Guides**] arrows under [**Documentation**] tab of selected device.

Table 2.1. Bootloader Documentation

Application Note/User's Guide	Device	Description
AN0003	MCU Series 0 MCU Series 1	*UART Bootloader* — All EFM32 devices are pre-programmed with UART bootloader, it is overwritten by Gecko Bootloader in this application note.
AN0042	USB enabled MCU Series 0	*USB/UART Bootloader* — For EFM32 Series 1, the USB Device Bootloader is implemented by USB Device MSD Gecko Bootloader loader in *AN1204: USB Device/Host MSD Gecko Bootloader Loader*.
AN0052	USB enabled MCU Series 0	*USB MSD Host Bootloader* — For EFM32 Series 1, the USB MSD Host Bootloader is implemented by USB Host MSD Gecko Bootloader loader in *AN1204: USB Device/Host MSD Gecko Bootloader Loader*.
UG103.6	MCU Series 1 Wireless SoC Series 1 Wireless SoC Series 2	*Bootloader Fundamentals* — This document introduces bootloading for Silicon Labs networking devices.
UG162	MCU Series 0 MCU Series 1 Wireless MCU Series 0 Wireless SoC Series 1 Wireless SoC Series 2	*Simplicity Commander Reference Guide* — This document describes how and when how to use the Command-Line Interface (CLI) of Simplicity Commander.
UG266	MCU Series 1 Wireless SoC Series 1 Wireless SoC Series 2	*Silicon Labs Gecko Bootloader User's Guide* — This document describes the high-level implementation of the Silicon Labs Gecko Bootloader for EFM32 and EFR32 Series 1 and Series 2 microcontrollers, SoCs (System on Chips) and NCPs (Network Co-Processors), and provides information on different aspects of configuring the Gecko Bootloader.

Note:
1. These application notes provides fundamental knowledge on bootloader.

The relevant software modules are found under the Simplicity Studio installation path. The default locations on Windows are shown in Table 2.2 Relevant Software Modules for the HTTP(S) Bootloader on page 3 (where vX.Y is the Gecko SDK version number).

Table 2.2. Relevant Software Modules for the HTTP(S) Bootloader

Software Module	Default Location on Windows and API Documentation Link
Gecko Bootloader	`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\platform\bootloader` — https://docs.silabs.com/mcu-bootloader/latest/
MicriumOS	`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\platform\micrium_os` — https://doc.micrium.com/display/OSUM50600
mbedTLS	`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\util\third_party\mbedtls` —http://devtools.silabs.com/dl/documentation/doxygen/5.6/mbedtls/html/index.html

3. Gecko Bootloader Overview

The Gecko Bootloader is a program stored in reserved flash memory that can initialize a device, update firmware images and possibly perform some integrity checks. Firmware image update occurs on demand, either by serial communication or over the air. Devices without a bootloader requires external hardware such as a Debug Adapter or third-party SerialWire/JTAG programming device to update the firmware. The Gecko Bootloader offers non-destructive write (i.e, the user cannot replace bootloader area with application code). The bootloader area is used to program bootloader only. For EFM32 and EFR32 Series 1 devices, the Gecko Bootloader is consisted of two stages, where a minimal first stage bootloader is used to update the main bootloader, and the main bootloader is used to perform firmware upgrade on applications. The Gecko Bootloader contains two different modes: standalone and application. 3.1 Standalone Bootloader and 3.2 Application Bootloader discuss these two modes in detail.

Note: EFM32GG11 is shipped with the *AN0003: UART Bootloader* from the factory. Users must flash the Gecko Bootloader on EFM32GG11 in order to use the HTTP(S) Bootloader. For more details refer to 3.3 Programming the Gecko Bootloader and 4. HTTP/HTTPS Bootloader Overview

3.1 Standalone Bootloader

A standalone bootloader is a program that uses an external communication interface, such as UART or SPI, to get an application image. Standalone firmware update is a single-stage process that allows the application image to be placed into flash memory, overwriting the existing application image, without the participation of the application itself. For more information, refer to *UG103.6: Bootloader Fundamentals* and *UG266: Silicon Labs Gecko Bootloader User's Guide*

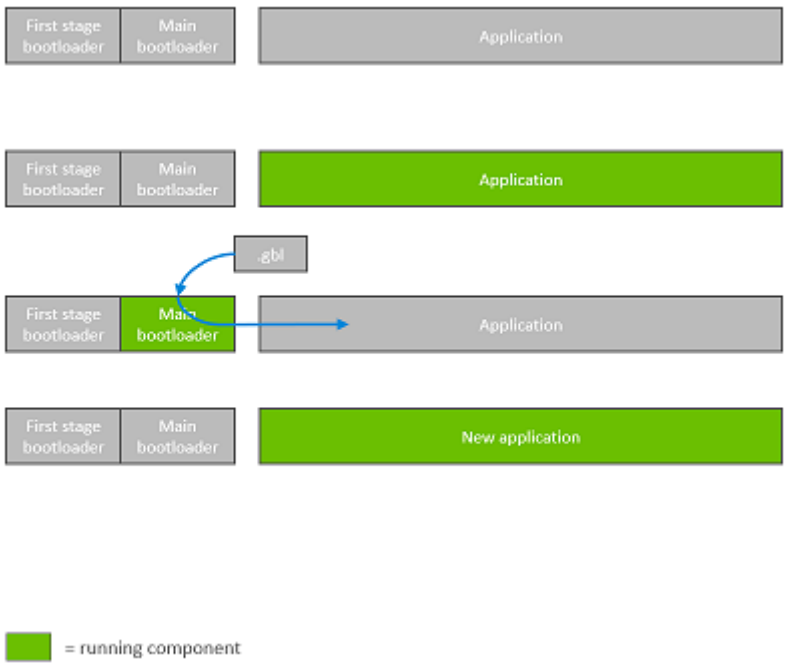


Figure 3.1. Standlone Bootloader Firmware Update Process

3.2 Application Bootloader

An application bootloader begins the firmware update process after the running application has completely downloaded the update image file. The application bootloader expects that the image either lives in external memory accessible by the bootloader or in a portion of main flash memory (internal storage). Both internal and external storage can have single or multiple storages. For the purpose of this application note, only the single internal storage mode is discussed. For information on the other types of application bootloader, please refer to *UG103.6: Bootloader Fundamentals* and *UG266: Silicon Labs Gecko Bootloader User's Guide*.

The HTTP(S) bootloader is an application bootloader. The application requests a GBL upgrade file through the HTTP(S) protocol and stores it in internal flash configurable by the user. After the application verifies the GBL image, the application reboots into bootloader and performs the upgrade. Figure 3.2 Application Bootloader Firmware Upgrade Process for Single Storage on page 5 illustrates the firmware upgrade process of the application bootloader both for a single image/single storage slot.

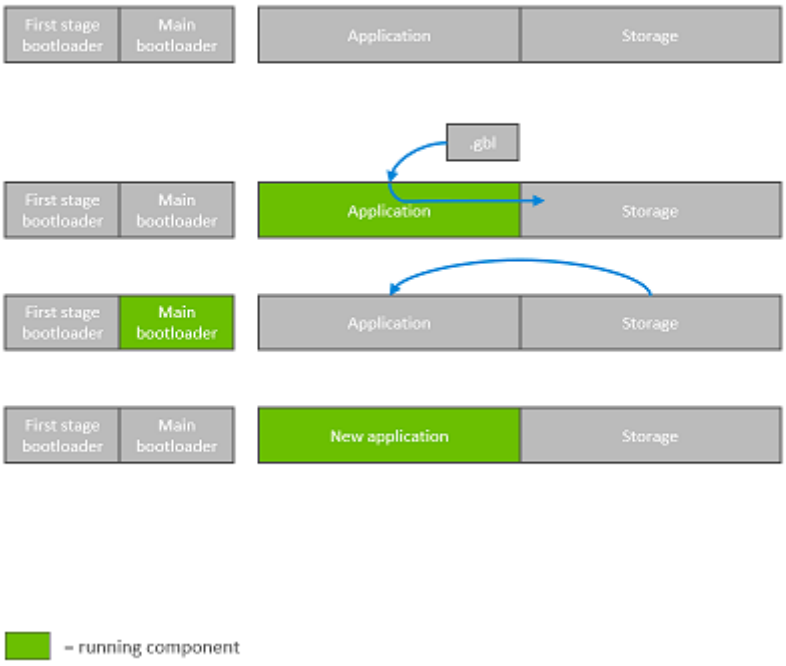


Figure 3.2. Application Bootloader Firmware Upgrade Process for Single Storage

3.3 Programming the Gecko Bootloader

This section shows user how to program a signal internal storage Gecko Bootloader onto EFM32GG11:

1. Connect a EFM32GG11 STK to a computer using the debugger port.
2. Open Simplicity Studio, navigate to the [**Debug Adapters**] tab. locate the J-Link connection drop down menu. Click on the drop down menu and select EFM32GG11 Giant Gecko Starter Kit board.

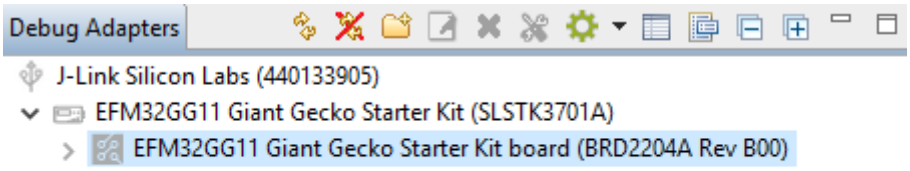


Figure 3.3. Debug Adapter Window

3. After ensuring that the latest Gecko SDK Suite has been installed, click on [**Gecko Bootloader Examples**] under [**Software Examples**] in the [**Getting Started**] tab. Select the [**Internal Storage Bootloader**] example.

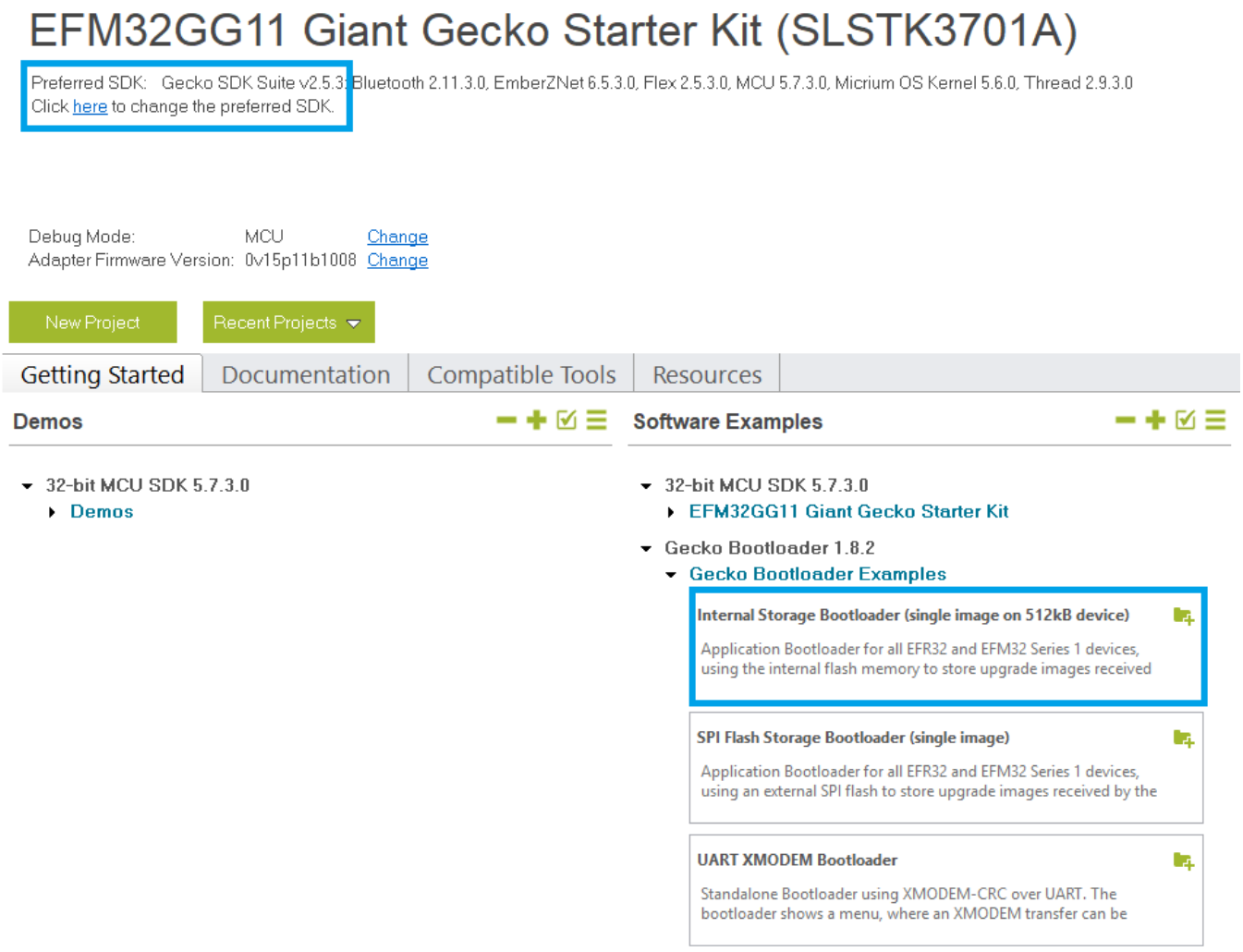


Figure 3.4. Internal Storage Bootloader Example

4. Click this example and navigate to the Simplicity IDE perspective. Open `bootloader-storage-internal-single-512k.isc` opened. Select the [**Storage**] tab and change both the start address and size (bytes) to 1048576. This allocates 1MB for bootloader storage area. Go to step 6 if Gecko Bootloader security feature is not required.

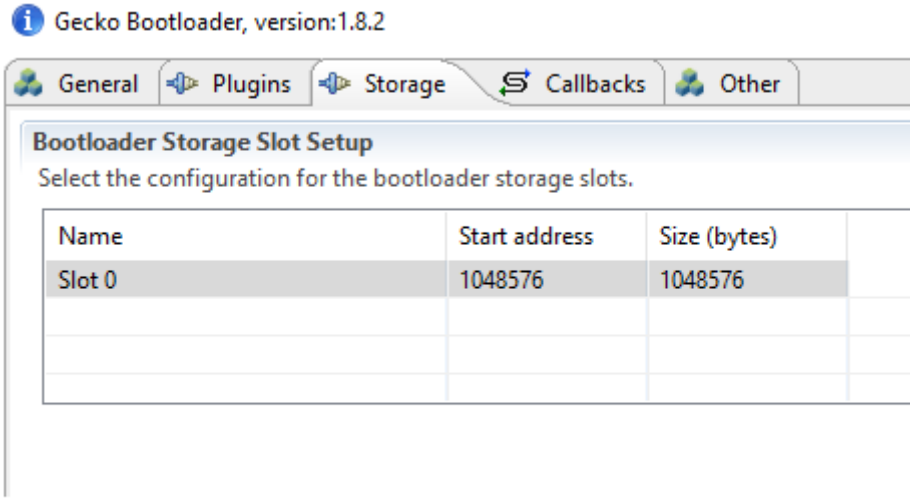
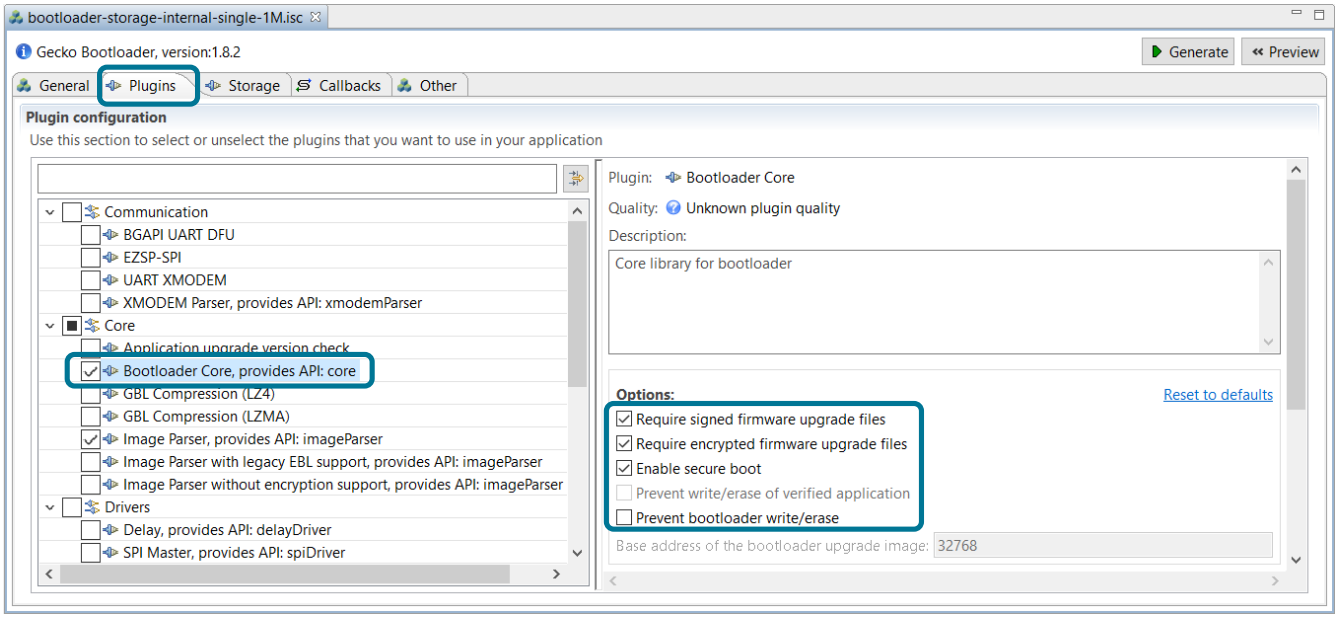


Figure 3.5. Gecko Bootloader Storage Configuration

Note: The user may choose example file of different sizes according to their device specification.

5. Click [**Plugins**] tab, select [**Core**] > [**Bootloader Core, provides API: core**], check all except [**Prevent bootloader write/erase**] option.



Note: This application notes does not cover the security feature of the Gecko Bootloader in the demo section. If the user wants to use the security feature in the Bootloader, then the user must provide their own securely signed GBL image and put it on their own server.

6. Save the project and click on [**Generate**] at the top rightmost corner. The [**Generation successful**] dialog should appear.

7. Select this project in the [**Project Explorer**] perspective, click the build icon ( )to build the project.

8. Once the project finished building, navigate to [**GNU ARM v7.2.1 - Debug**] folder and search for the `bootloader-storage-internal-single-512k-combined.s37` file. This is the file must be used to flash the full Gecko Bootloader image on EFM32GG11.

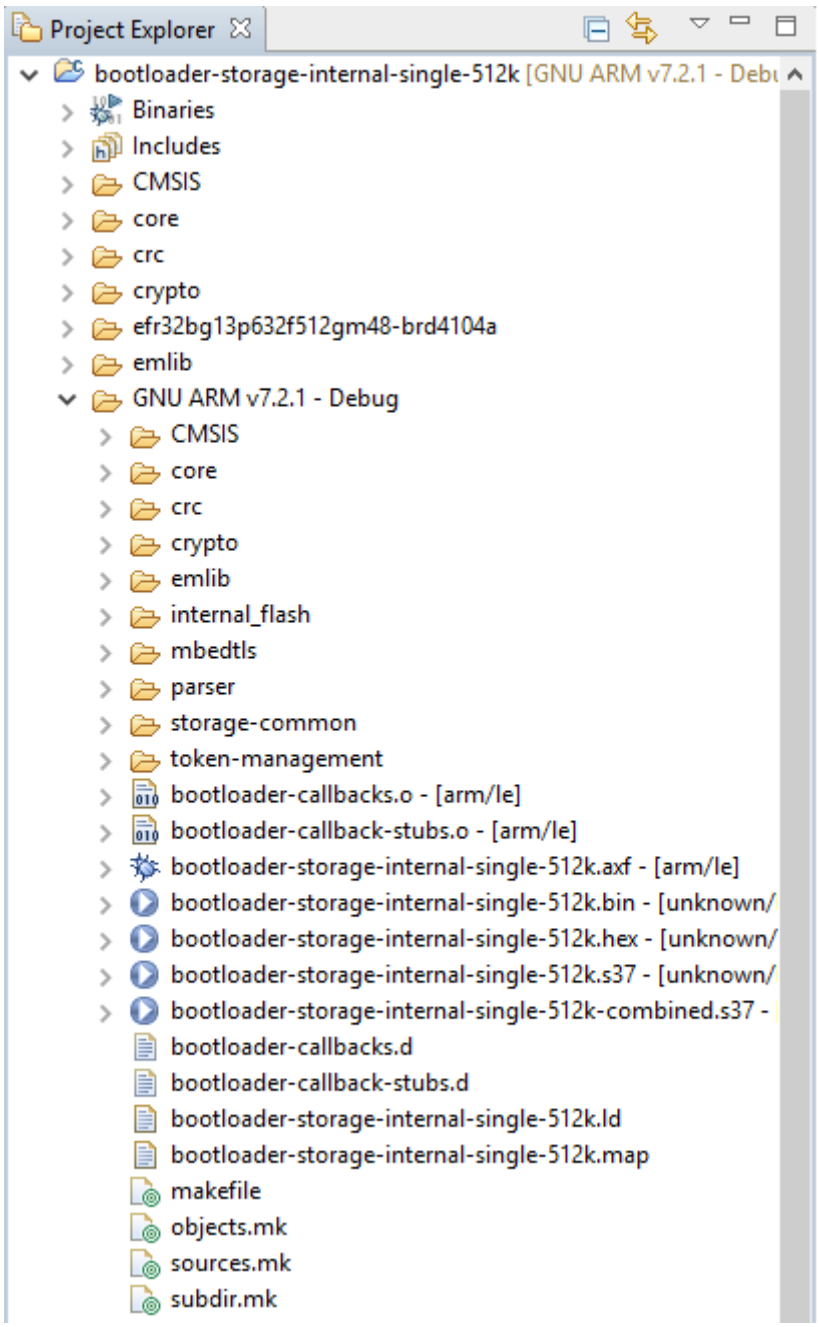


Figure 3.6. Combined Bootloader s37 file

9. Right click on the file and select [**Flash to Device**].

Note: The file `bootloader-storage-internal-single-512k-combined.s37` is the combined first stage + main bootloader image. The first time a device is programmed, the combined image is required. For subsequent programming, when a first stage bootloader is already present in the device, only the main bootloader image is required, which is called `bootloader-storage-internal-single-512k.s37`.

4. HTTP/HTTPS Bootloader Overview

HTTP(S) bootloader is an application bootloader based on the Gecko Bootloader. It provides firmware updates through HTTP(S) protocol. The embedded application in the EFM32 Giant Gecko 11 Starter Kit acts as an HTTP-client and connects to the web server which can be modified through a command line interface:

Figure 4.1. Application Command Line Interface

The command line appears upon start up. The URL can be configured by typing "url" followed by the web server that the user would like to configure. After setting the URL, the user can enter "upgrade" and the EFM32GG11 will try to connect to the web server. Once connected, the application sends a GET request to the server and in return retrieves a firmware update file, which is in Gecko Bootload File (GBL) format. Once the file is downloaded, the application places the file into the bootloader storage space configured by the user. The application validates the upgrade file, reboots into the bootloader and upgrades either application, bootloader or both based on the file specification.

4.1 Gecko Bootloader Image

Since the HTTP(S) bootloader is based on the Gecko Bootloader, users must make sure that a full Gecko Bootloader image, consisting of first and main stage bootloader binaries, is present in the bootloader area of the device. In EFM32GG11, the size of the bootloader area is 32 kB. Out of the 32 kB, the first 4 kB is assigned to the first stage bootloader and the remaining 28 kB is allocated to the main stage bootloader. The address range of the first stage of the bootloader is **0xFE10000 - 0xFE10FFF**. The address range of the main stage of the bootloader is **0xFE11000 - 0xFE17FFF**. This application note uses the Internal Storage Bootloader provided by the Gecko Software Development Kit. This example bootloader provides a full image Gecko Bootloader with a single internal storage. Details on setting up this bootloader is discussed in 3.3 Programming the Gecko Bootloader .

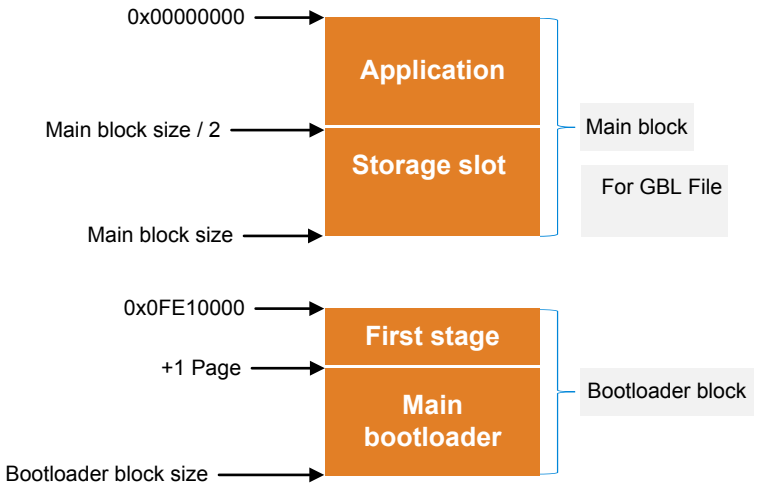


Figure 4.2. Bootloader Memory Layout for EFM32GG11

4.2 Hardware Requirements

The HTTP(S) bootloader firmware upgrade process requires the following set of hardware components:

- EFM32 Giant Gecko 11 Starter Kit
- Ethernet Cable
- Power Supply
- Internet Access
- An adapter for the Ethernet Cable—Router, Modem, Switch or an Ethernet Wall Outlet

4.3 Software Components

The software components that HTTP(S) bootloader firmware upgrade needs are:

- The Gecko Bootloader image
- The embedded application

The embedded application is responsible for the following:

- Providing the command line interface, which has the following functionality:
  - Modify the web server which contains the new application software
  - Initialize EFM32GG11 and try to make connection to the web server
  - Obtain the current application version
  - Acqurie bootloader information: storage type, address and size
  - Query the current time from the network
- Downloading and validating the firmware update image using the bootloader interface API
- Reboot into main stage bootloader to perform the upgrade

This Application Note uses the following software components provided in the Gecko SDK Suite:

- Internal single storage Gecko Bootloader
- Micrium OS
- mbedTLS

4.4 Firmware Upgrade Process using HTTP(S) Bootloader covers the flow of the firmware update process and references these ccomponents.

The application uses the bootloader API to verify the bootloader and the bootloader storage space. It also verifies the GBL image in the storage space and reboots into the bootloader when verification is complete. Micrium OS is used to establish the network interface and retrieve the update image from the server. mbedTLS is used for communication security and provide SSL/TLS communication channel. The embedded application uses these software components to perform the firmware upgrade process.

4.4 Firmware Upgrade Process using HTTP(S) Bootloader

The HTTP(S) bootloader firmware upgrade process consists of five phases:

- Bootloader initialization and setup
- Network interface setup
- Client-server secure communication,
- Download and verify upgrade image
- Reprogram flash with the upgrade image

Note: This example uses the Micrium OS and mbedTLS to set up network interface and establish secure connection. Users may choose their perferred stack to perform this task.

The program flow is illustrated in Figure 4.3 HTTP(S) Bootloader Firmware Upgrade Flow on page 12.

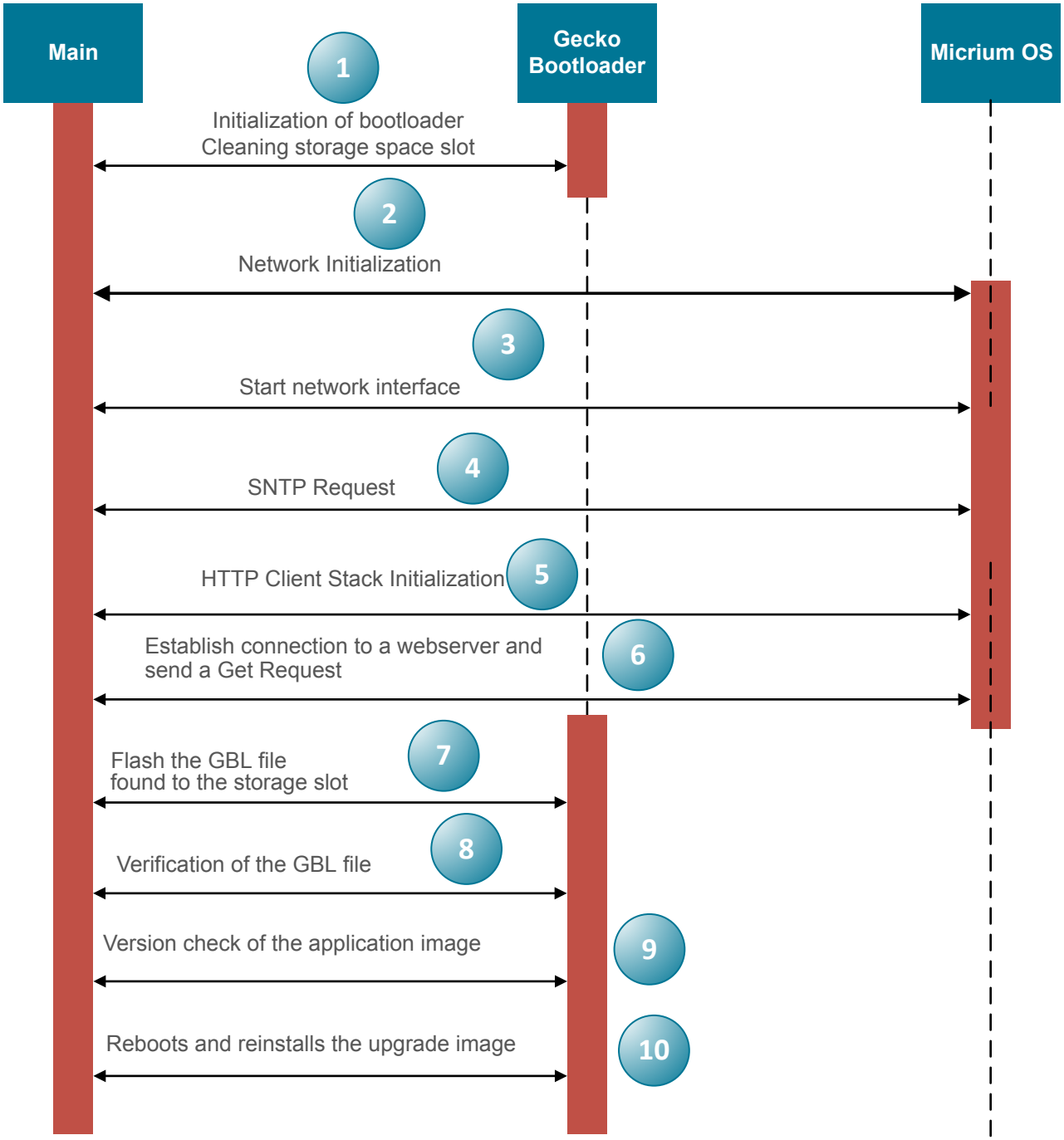


Figure 4.3. HTTP(S) Bootloader Firmware Upgrade Flow

1. Bootloader initialization and setup:
   - Device boots into the application
   - Application checks for the presence of the Gecko Bootloader using bootloader API
   - Bootloader information – storage space base-address and its size
   - Bootloader API displays current application version on terminal
   - Clean storage space slot – MSC erase starting from base address of the storage area
   - **Failure case**: Gecko Bootloader is missing

2. Network Interface:
   - Initialize the network TCP-IP core and add an Ethernet interface
   - Start network interface – waits for the network interface to be ready
   - Send SNTP request to **0.pool.ntp.org** – used to find the elapsed time since 1970, required for HTTPS protocol
   - Initialize the HTTP Client stack
   - **Failure cases**:
     - Ethernet connection could not be found
     - SNTP Timeout - This is caused by internal server firewall blocking communication to public NTP Server
3. Client-server communication:
   - Prepare connection to a web server using HTTP/HTTPS
   - Assign SSL root certificate
   - Establish a connection to the server and send GET request
     - Using the URL given
     - Establish a secure connection using mbedTLS
     - Building on top of the network interface, mbedTLS provides an abstraction layer for secure communication
   - **Failure case**: HTTP/HTTPS response error
4. Download and verify upgrade image:
   - Flash the contents of the file received from the webserver to the bootloader storage space on-the-fly
   - Verification of the GBL file
   - Check the version of the upgrade image stored in the GBL file
   - **Failure case**:
     - Invalid GBL file – Storage slot is cleaned and system resets
     - Version of the running application has higher or equal version number as the application stored in the image file – Storage slot is cleaned and system resets
5. Reprogram flash with the upgrade image:
   - Reboots into main bootloader
   - Extracts the upgrade image from the GBL file
   - Apply upgrades

5. HTTP(S) Bootloader Demo

This chapter presents the user with a walkthrough of the HTTP(S) bootloader. It provides a step-by-step guide on how to flash a GBL image onto EFM32GG11 using the HTTP(S) bootloader.

Note: This demo does not incorporate the security features that can be used with the bootloader. The URL link provided in the demo is not securely signed with keys and therefore is only intended to be used on Gecko Bootloader without security feature enabled.

5.1 Create Micriumos_httpcloader Example

Once a full Gecko Bootloader image is flashed onto EFM32GG11, the user can proceed to program the HTTP(S) embedded application. This Application Note uses `SLSTK3701A_micriumos_httpcloader` example, with the default location in `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\micriumos_httpcloader`, where vX.Y is the version of the SDK.

1. Connect a EFM32GG11 STK to a computer using debug port, select EFM32GG11 Giant Gecko Starter Kit board in the Debug Adapters tab.
2. From the [**Launcher**] Perspective, click [**EFM32GG11 Giant Gecko Starter Kit**] under [**Software Examples**] of [**Getting Started**] tab to browse target sample application.
3. Click the `SLSTK3701A_micriumos_httpcloader` sample application to create the example project in Simplicity IDE.

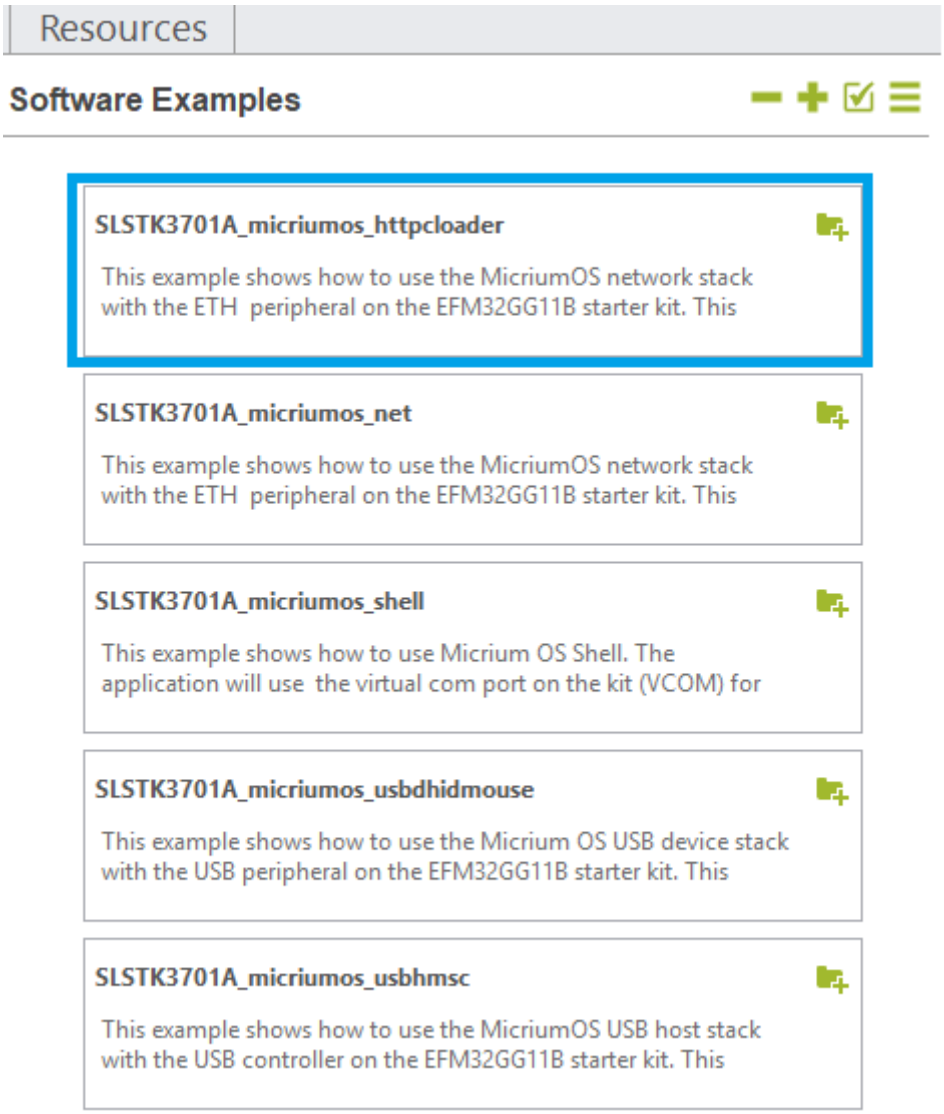


Figure 5.1. micriumos_httpcloader example

4. The project opens in the Simplicity IDE perspective.

5.2 Obtain Certificate Authority(CA) of www.silabs.com

This section describes how to find the correct root Certificate Authority (CA) of www.silabs.com.

1. Navigate to https://www.silabs.com using a web browser.
2. Click on [**View site information**] button (lock icon). Find the certificate information by clicking on [**Certificate**] tab.

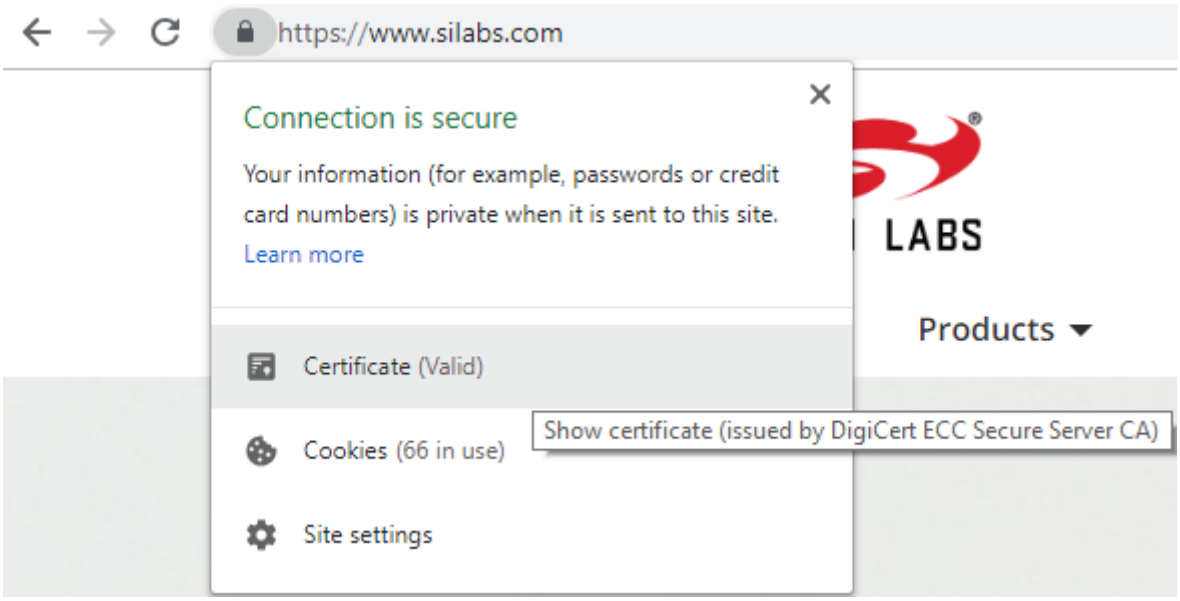


Figure 5.2. Silabs Certificate Location

3. Click on [**Certificate Path**] tab and double click on [**DigiCert**]. Click on [**General**] tab to display certificate information. This is used to establish secure connection between the server and the client.

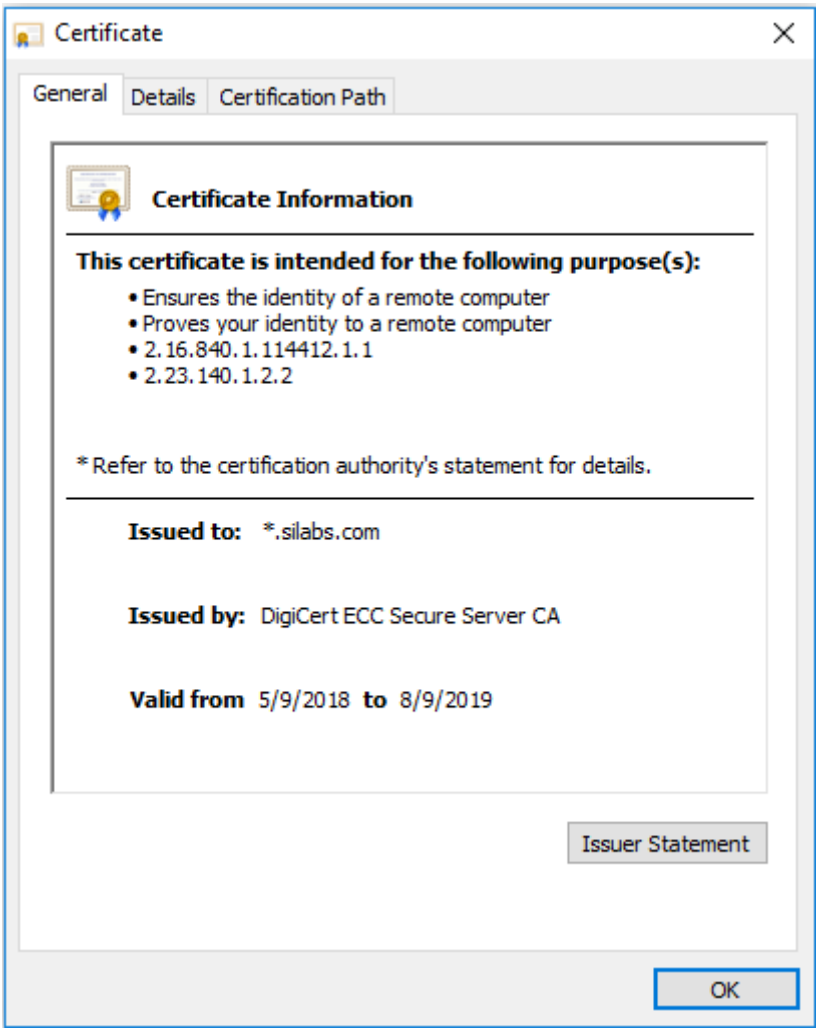


Figure 5.3. Silabs Certificate Window

4. Obtain the Privacy Enhanced Mail(PEM) version of DigiCert Global Root CA certificate from https://global-root-ca.chain-demos.digicert.com/info/index.html. This the root CA used in `ssl_certificate.c`, and it is also part of the client-server connection process.

```
Certificate:
-----BEGIN CERTIFICATE-----
MIIDrzCCApegAwIBAgIQCDvgVpBCRrGhdWrJWZHHSjANBgkqhkiG9w0BAQUFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBD
QTAeFw0wNjExMTAwMDAwMDBaFw0zMTExMTAwMDAwMDBaMGExCzAJBgNVBAYTAlVT
MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxGTAXBgNVBAsTEHd3dy5kaWdpY2VydC5j
b20xIDAeBgNVBAMTF0RpZ2lDZXJ0IEdsb2JhbCBSb290IENBMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4jvhEXLeqKTTo1eqUKKPC3eQyaKl7hLOllsB
CSDMAZOnTjC3U/dDxGkAV53ijSLdhwZAAIEJzs4bg7/fzTtxRuLWZscFs3YnFo97
nh6Vfe63SKMI2tavegw5BmV/Sl0fvBf4q77uKNd0f3p4mVmFaG5cIzJLv07A6Fpt
43C/dxC//AH2hdmoRBBYMql1GNXRor5H4idq9Joz+EkIYIvUX7Q6hL+hqkpMfT7P
T19sdl6gSzeRntwi5m3OFBqOasv+zbMUZBfHWymeMr/y7vrTC0LUq7dBMtoM1O/4
gdW7jVg/tRvoSSiicNoxBN33shbyTApOB6jtSj1etX+jkMOvJwIDAQABo2MwYTAO
BgNVHQ8BAf8EBAMCAYYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUA95QNVbR
TLtm8KPiGxvDl7I90VUwHwYDVR0jBBgwFoAUA95QNVbRTLtm8KPiGxvDl7I90VUw
DQYJKoZIhvcNAQEFBQADggEBAMucN6pIExIK+t1EnE9SsPTfrgT1eXkIoyQY/Esr
hMAtudXH/vTBH1jLuG2cenTnmCmrEbXjcKChzUyImZOMkXDiqw8cvpOp/2PV5Adg
06O/nVsJ8dWO41P0jmP6P6fbtGbfYmbW0W5BjfIttep3Sp+dWOIrWcBAI+0tKIJF
PnlUkiaY4IBIqDfv8NZ5YBberOgOzW6sRBc4L0na4UU+Krk2U886UAb3LujEV0ls
YSEY1QSteDwsOoBrp+uvFRTp2InBuThs4pFsiv9kuXclVzDAGySj4dzp30d8tbQk
CAUw7C29C79Fv1C5qfPrmAESrciIxpg0X40KPMbp1ZWVbd4=
-----END CERTIFICATE-----
```

Figure 5.4. DigiCert Global Root CA

5. Open `ssl_certificate.c` from the `SLSTK3701A_micriumos_httpcloader` example in Simplicity Studio. The root CA should have already been included in the project.

```
/* DigiCert Global Root CA */
"-----BEGIN CERTIFICATE-----\r\n"
"MIIDrzCCApegAwIBAgIQCDvgVpBCRrGhdWrJWZHHSjANBgkqhkiG9w0BAQUFADBh\r\n"
"MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3\r\n"
"d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBD\r\n"
"QTAeFw0wNjExMTAwMDAwMDBaFw0zMTExMTAwMDAwMDBaMGExCzAJBgNVBAYTAlVT\r\n"
"MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxGTAXBgNVBAsTEHd3dy5kaWdpY2VydC5j\r\n"
"b20xIDAeBgNVBAMTF0RpZ2lDZXJ0IEdsb2JhbCBSb290IENBMIIBIjANBgkqhkiG\r\n"
"9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4jvhEXLeqKTTo1eqUKKPC3eQyaKl7hLOllsB\r\n"
"CSDMAZOnTjC3U/dDxGkAV53ijSLdhwZAAIEJzs4bg7/fzTtxRuLWZscFs3YnFo97\r\n"
"nh6Vfe63SKMI2tavegw5BmV/Sl0fvBf4q77uKNd0f3p4mVmFaG5cIzJLv07A6Fpt\r\n"
"43C/dxC//AH2hdmoRBBYMql1GNXRor5H4idq9Joz+EkIYIvUX7Q6hL+hqkpMfT7P\r\n"
"T19sdl6gSzeRntwi5m3OFBqOasv+zbMUZBfHWymeMr/y7vrTC0LUq7dBMtoM1O/4\r\n"
"gdW7jVg/tRvoSSiicNoxBN33shbyTApOB6jtSj1etX+jkMOvJwIDAQABo2MwYTAO\r\n"
"BgNVHQ8BAf8EBAMCAYYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUA95QNVbR\r\n"
"TLtm8KPiGxvDl7I90VUwHwYDVR0jBBgwFoAUA95QNVbRTLtm8KPiGxvDl7I90VUw\r\n"
"DQYJKoZIhvcNAQEFBQADggEBAMucN6pIExIK+t1EnE9SsPTfrgT1eXkIoyQY/Esr\r\n"
"hMAtudXH/vTBH1jLuG2cenTnmCmrEbXjcKChzUyImZOMkXDiqw8cvpOp/2PV5Adg\r\n"
"06O/nVsJ8dWO41P0jmP6P6fbtGbfYmbW0W5BjfIttep3Sp+dWOIrWcBAI+0tKIJF\r\n"
"PnlUkiaY4IBIqDfv8NZ5YBberOgOzW6sRBc4L0na4UU+Krk2U886UAb3LujEV0ls\r\n"
"YSEY1QSteDwsOoBrp+uvFRTp2InBuThs4pFsiv9kuXclVzDAGySj4dzp30d8tbQk\r\n"
"CAUw7C29C79Fv1C5qfPrmAESrciIxpg0X40KPMbp1ZWVbd4=\r\n"
"-----END CERTIFICATE-----\r\n";
```

Figure 5.5. DigiCert Global Root CA in Example

5.3 Update Secure Socket Layer(SSL) Header File with www.silabs.com Certificate Information

This section shows how to update `config-ssl-httpcloader.h` using certificate information from https://www.silabs.com.

1. Navigate to silabs.com using a web browser. Click on [**View site information**] button ( ). Find the certificate information by clicking on [**Certificate**] tab.
2. Click on [**Certificate Path**] tab, double click on [**DigiCert**] tab. Find [**Public Key**] information from [**Details**] tab.

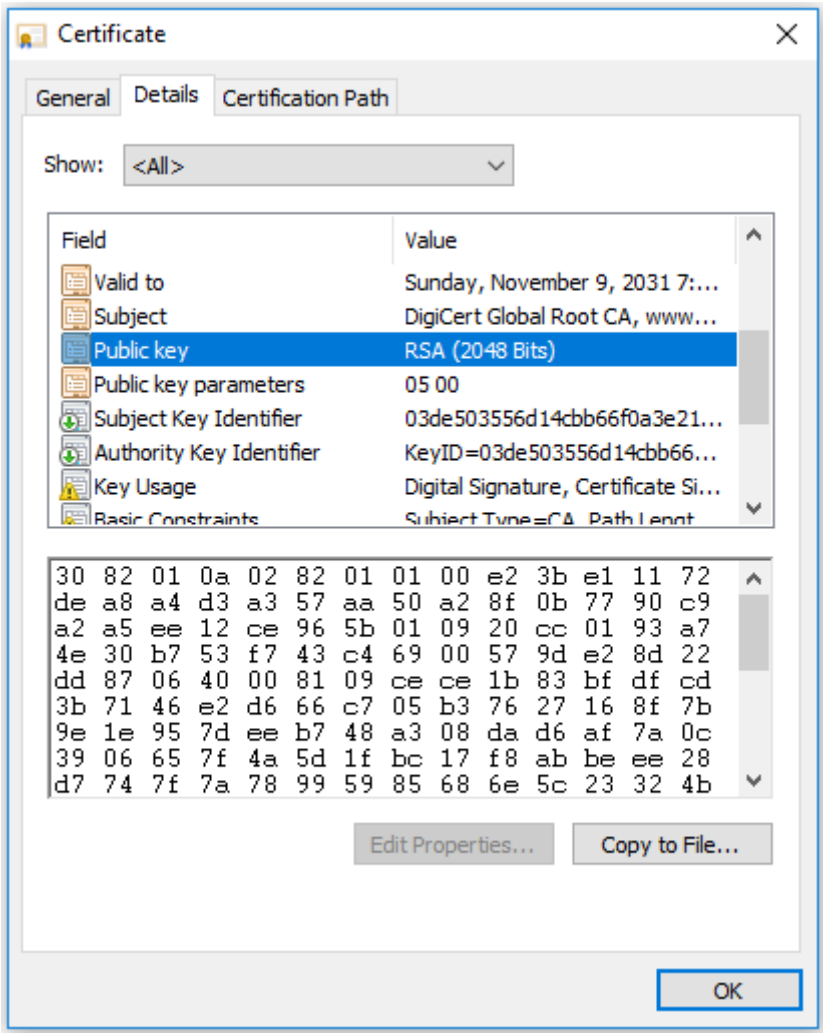


Figure 5.6. Digicert Public Key

3. Open `config-ssl-httpcloader.h` from the `SLSTK3701A_micriumos_httpcloader` project in Simplicty Studio. The header file can be found under the project include files as shown in Figure 5.7 config-ssl-httpcloader.h file location on page 20. The default location for this file is `C:/SiliconLabs/SimplicityStudio/v4/developer/sdks/gecko_sdk_suite/vX.Y/app/mcu_example/SLSTK3701A_EFM32GG11/micriumos_httpcloader` if Studio is installed in the default location, where vX.Y is the version number of the SDK.

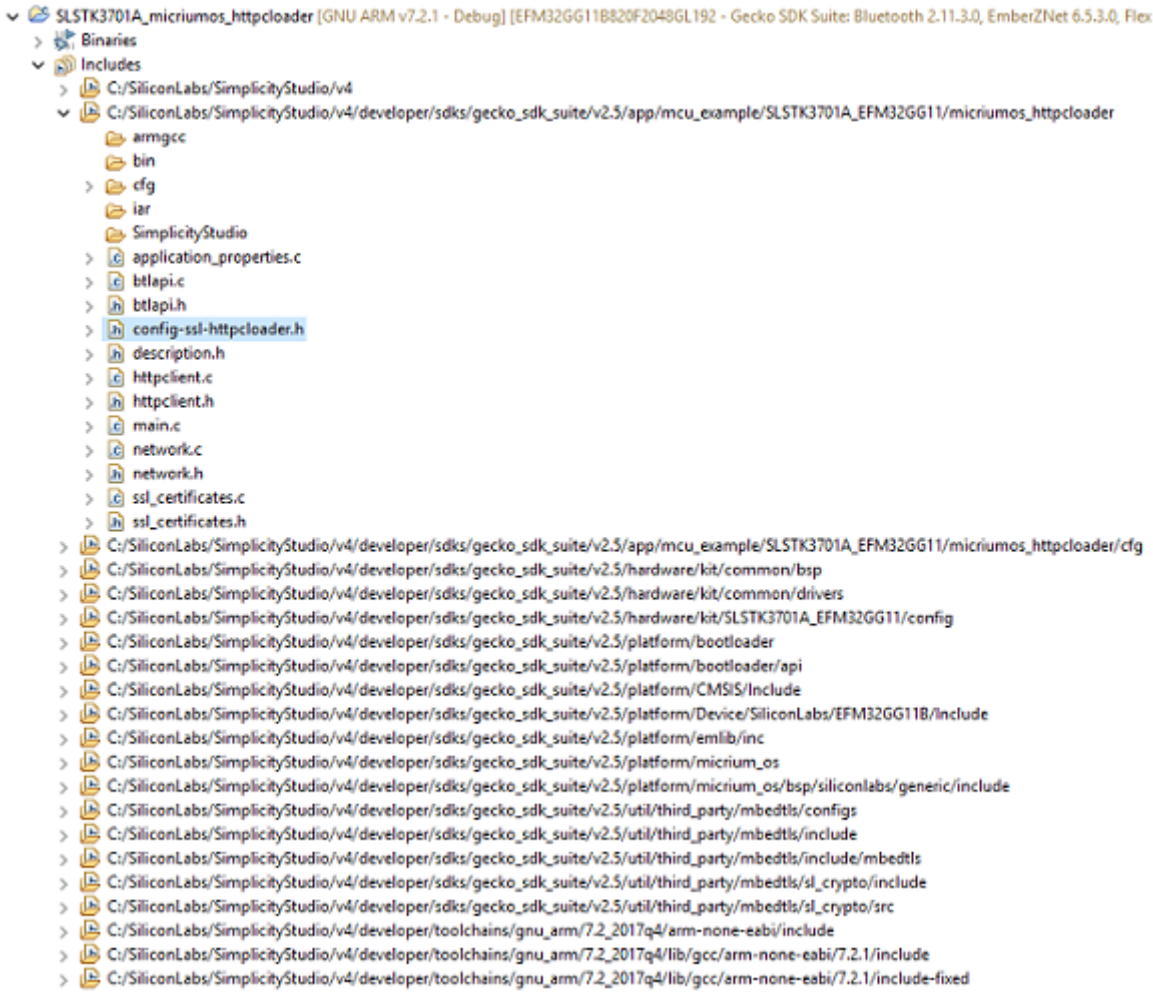


Figure 5.7. config-ssl-httpcloader.h file location

4. Find [**MBEDTLS_MPI_MAX_SIZE**] field in `config-ssl-httpcloader.h`, change it to match the RSA key size (256 bytes). If a warning message pops up, click [**Make a Copy**] button.

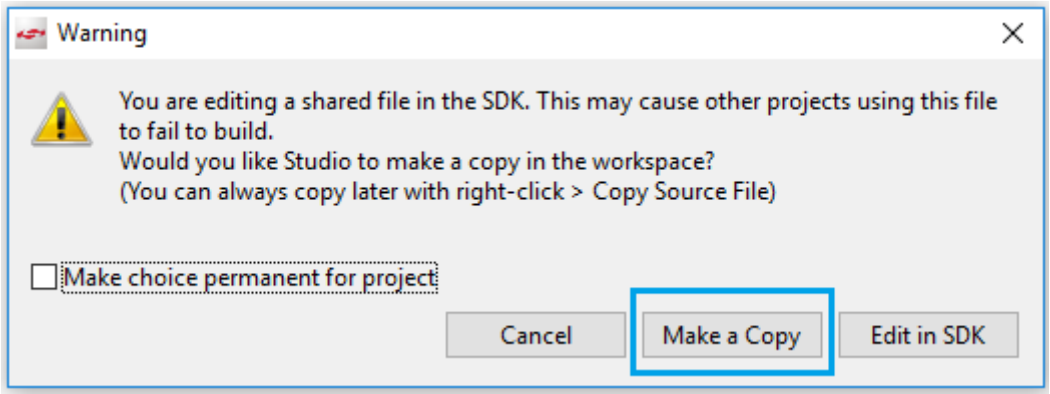


Figure 5.8. Warning Message

5. Navigate to the web brower, repeat steps 1 and 2 and locate the [**Certificate Path**] tab on the Certificate window. Double click on [**DigiCert ECC Secure Server CA**], click on [**Detail**] tab to find [**Public Key**] information.

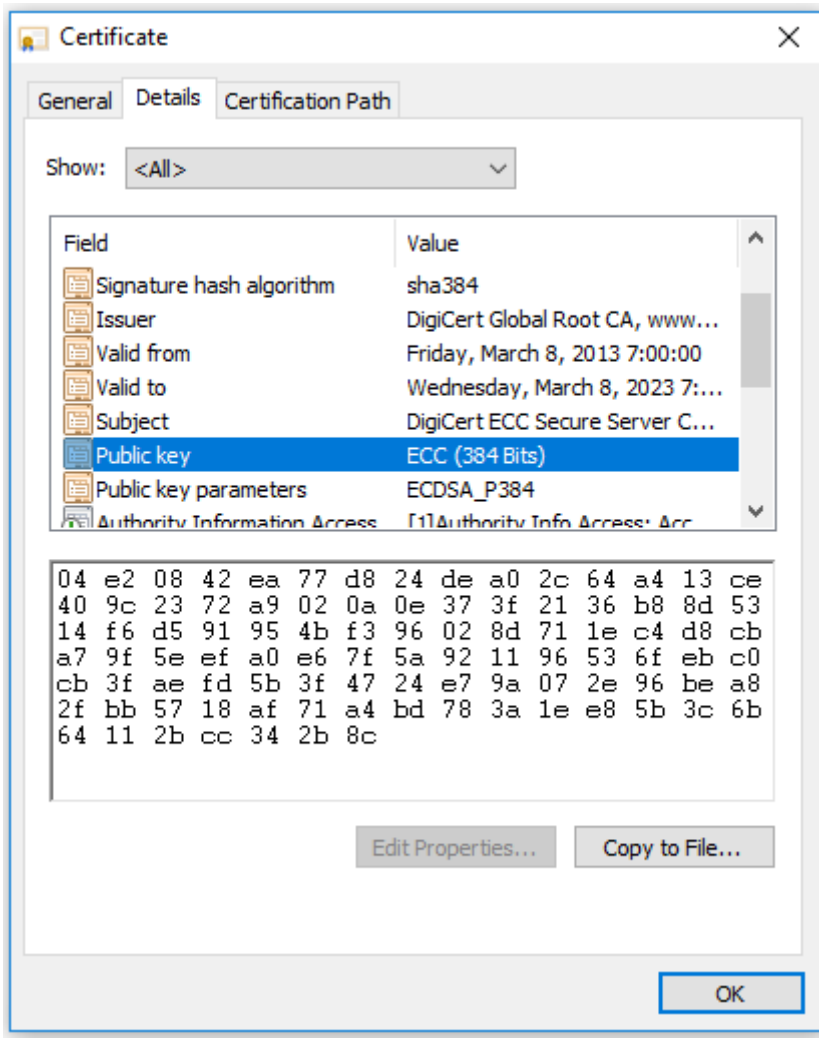


Figure 5.9. DigiCert ECC Public Key

6. Navigate to `config-ssl-httpcloader.h` and look for [**MBEDTLS_ECP_MAX_BITS**] field, change it to match the ECC key size (384 bits).

5.4 Request GBL file and upgrade application

This section shows how to request a GBL file and how to check that the application has been upgraded successfully.

1. Connect EFM32GG11 to Ethernet using a Ethernet cable.
2. Open a terminal program (e.g. Tera Term). Create a new connection and select Serial with the [**Jlink CDC UART Port**].

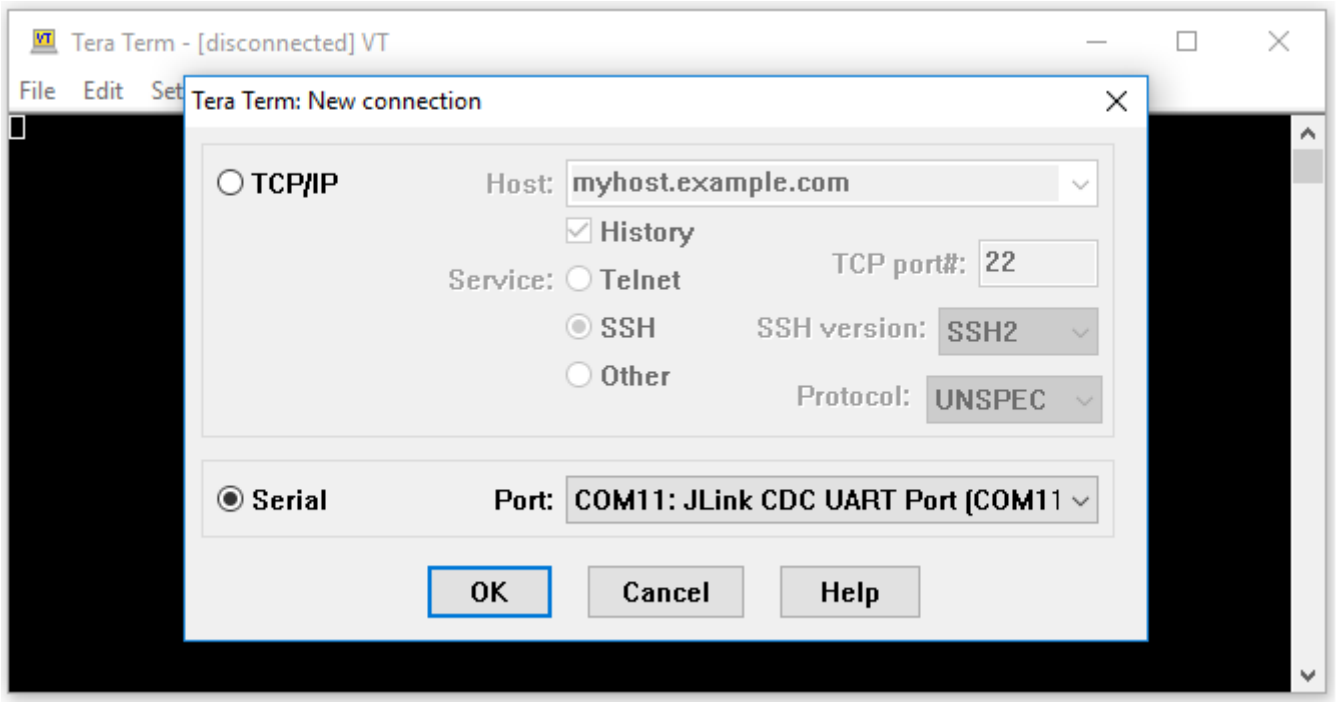


Figure 5.10. Terminal Initialization

3. In Simplicity Studio, select `SLSTK3701A_micriumos_httpcloader` project and click the Debug button( ).

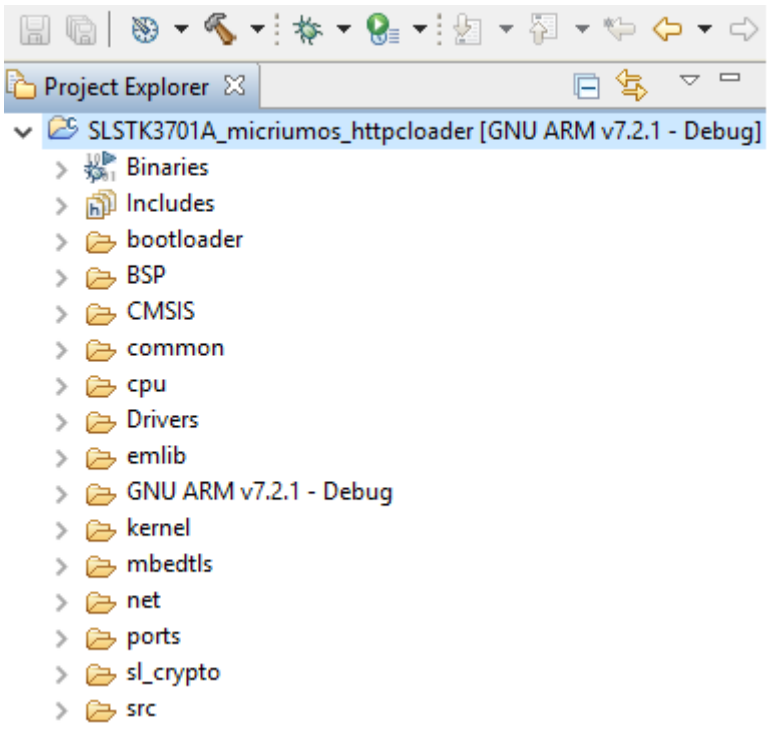


Figure 5.11. Micriumos httpcloader example build

4. From the [**Debug**] perspective, click the [**Resume**] button( ) . Open the terminal, i.e., TeraTerm or Putty, the output terminal should look similar to Figure 5.12 Commander Interface on page 23.

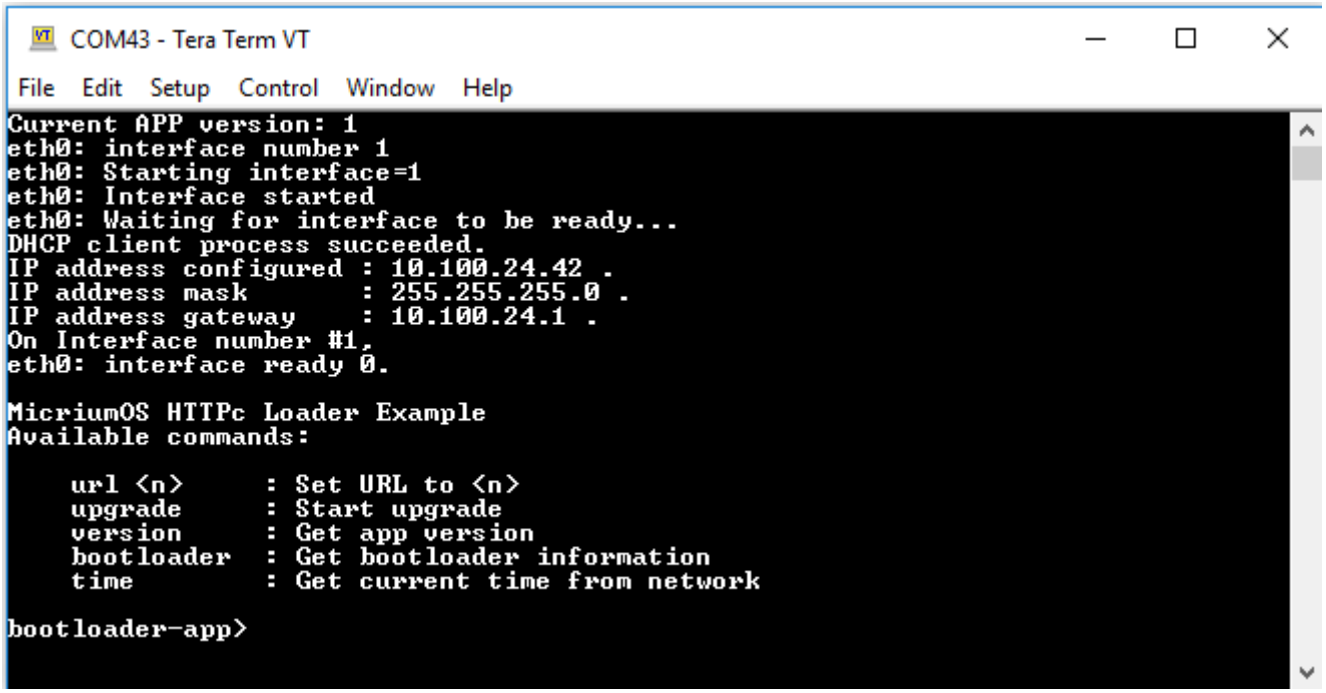


Figure 5.12. Commander Interface

5. Type in URL: https://www.silabs.com/content/dam/siliconlabs/documents/referenced/gbls/httpcloader.gbl

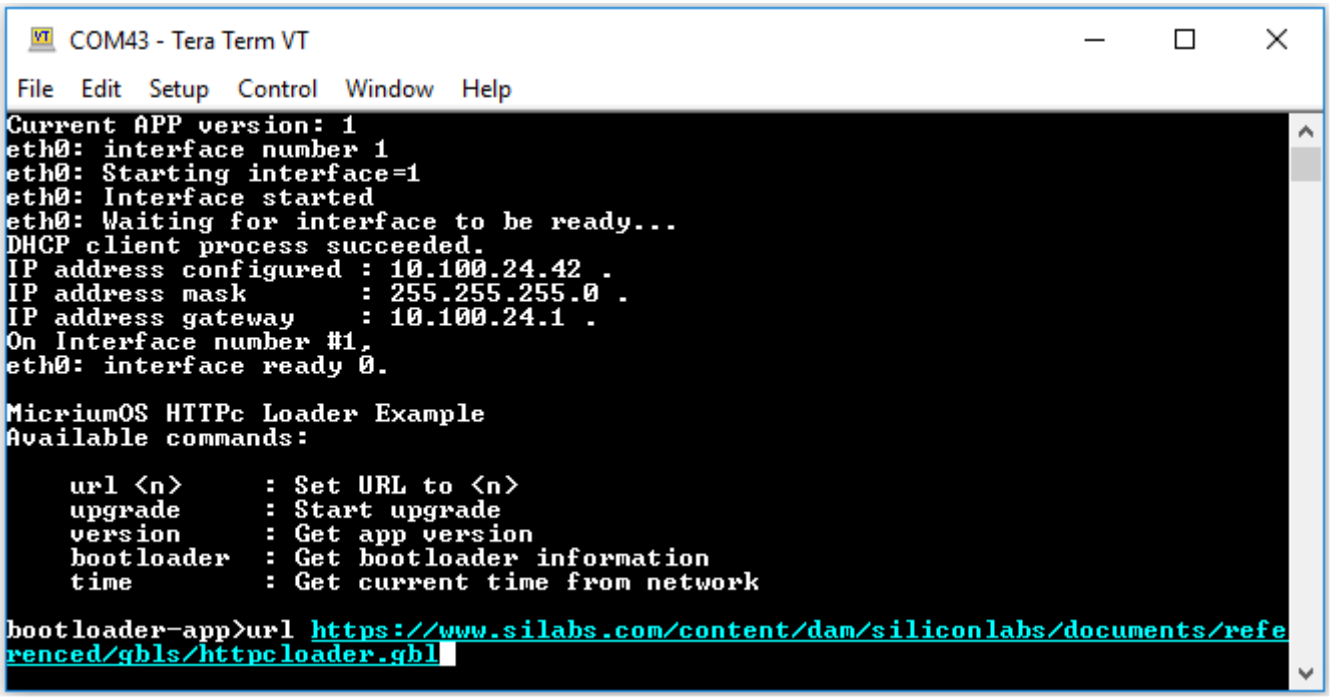


Figure 5.13. Enter URL

6. Type *upgrade* and press enter. The device will try to connect to the web server that has been configured, it will then download and verify the upgrade image. The terminal output upon successful completion should look similar to Figure 5.14 Terminal Output Upon Completion on page 24

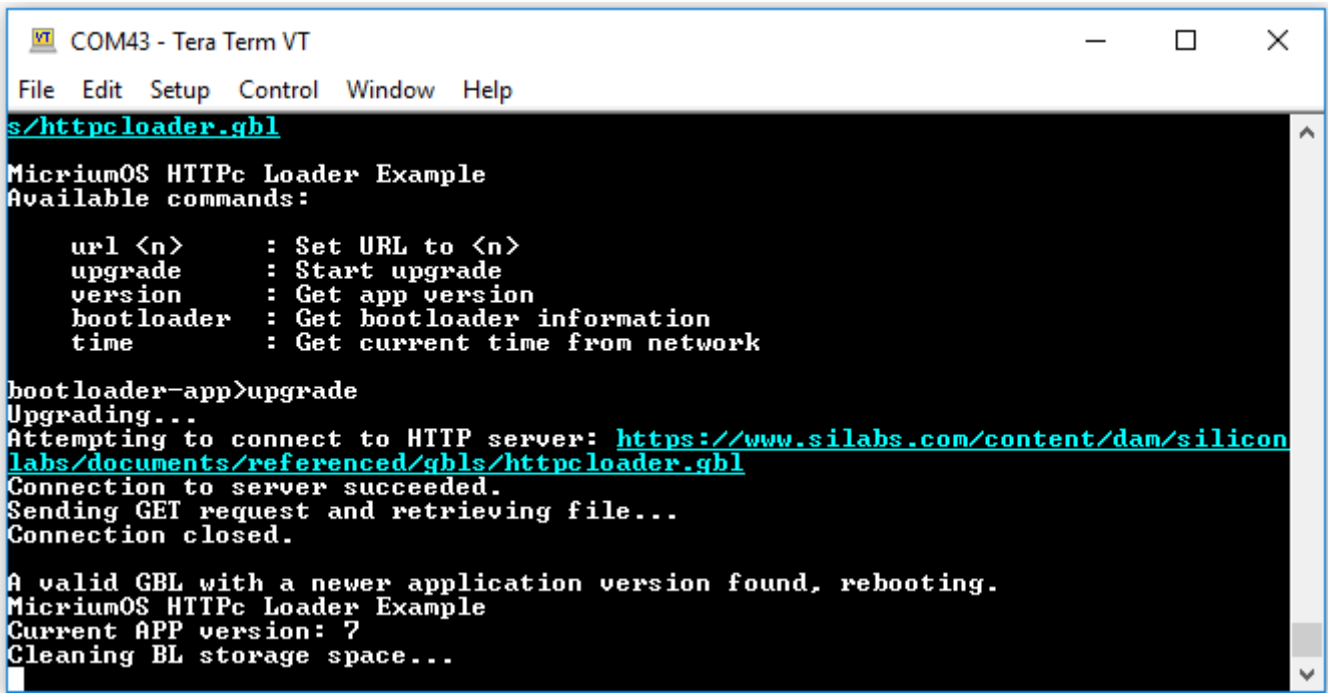


Figure 5.14. Terminal Output Upon Completion

Note: Step 7-11 provided steps to securely sign the application image. The information is here for user's reference and are not part of this demo

7. Execute the following command in Simplicity Commander to generate a key-pair for signing (`httpdevice-signing-key`).

```
commander gbl keygen --type ecc-p256 --outfile httpdevice-signing-key
```

8. Execute the following command in Simplicity Commander to generate an encryption key (`httpdevice-encryption-key`).

```
commander gbl keygen --type aes-ccm --outfile httpdevice-encryption-key
```

9. Execute the following command in Simplicity Commander to write the public signing key (`httpdevice-signing-key-tokens.txt`) and encryption key (`httpdevice-encryption-key`) to the EFM32GG11.

```
commander flash --tokengroup znet --tokenfile httpdevice-encryption-key --tokenfile httpdevice-signing-key-tokens.txt
```

10. Execute the following command in Simplicity Commander to sign the application image (`SLSTK3701A_micriumos_httpcloader.s37`) to enable secure boot of the application image (`SLSTK3701A_micriumos_httpcloader_signed.s37`). The application image is signed using ECDSA-P256 and the signature is verified on every boot.

```
commander convert SLSTK3701A_micriumos_httpcloader.s37 --secureboot --keyfile httpdevice-signing-key --outfile SLSTK3701A_micriumos_httpcloader_signed.s37
```

11. Execute the following command to flash the secure application on the device.

```
commander flash SLSTK3701A_micriumos_httpcloader_signed.s37
```

Note: If Gecko Bootloader security feature is enabled. The GBL image must also be secure signed, otherwise it is an invalid image.

5.5 Create a GBL File

This section describes how to generate a normal or secure Gecko Bootloader (GBL) file containing an upgrade image with a higher verion number of the application than the running application.

1. Connect EFM32GG11 STK to a computer and start Simplicity Studio.
2. Navigate to the previously created example `SLSTK3701A_micriumos_httpcloader`
3. Open `application_properties.c` file in **Project Explorer** and increase `APP_PROPERTIES_VERSION` by 1 (e.g. from 1 to 2).

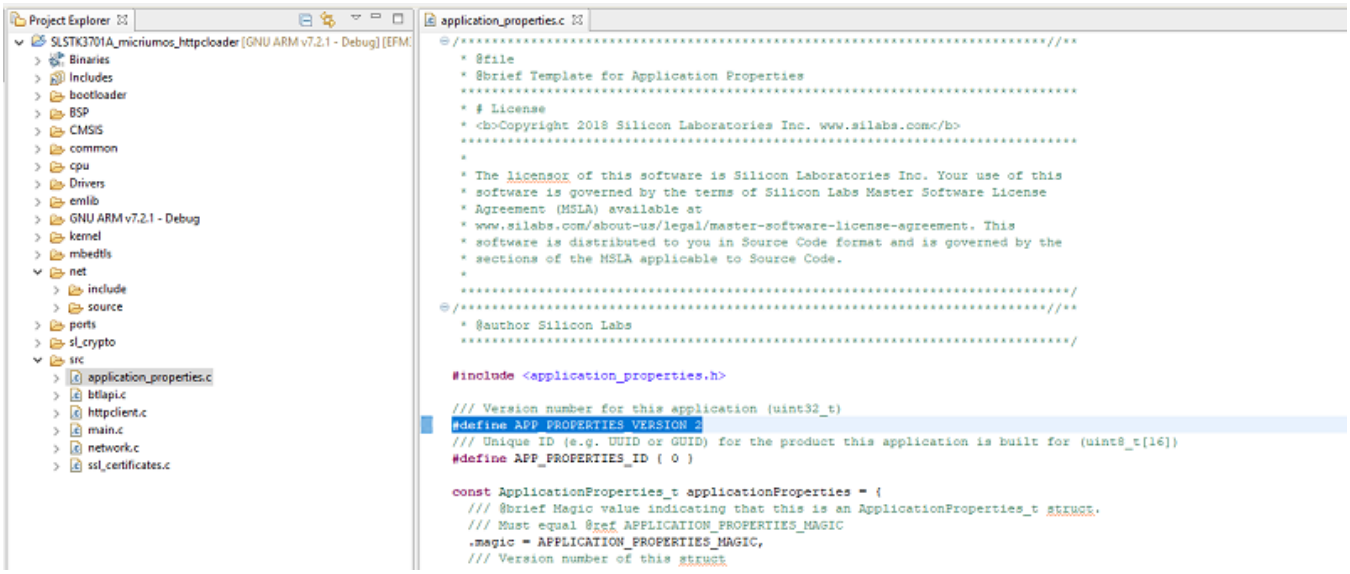


Figure 5.15. Application Properties Version

4. Click the [**Build**] icon ( ). Copy the `SLSTK3701A_micriumos_httpcloader.s37` file from the build directory to Simplicity Commander folder (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\adapter_packs\commander`).
5. Execute the following command in Simplicity Commander to create an upgrade GBL file (`SLSTK3701A_micriumos_httpcloader.gbl`). Go to step 7.

```
commander gbl create SLSTK3701A_micriumos_httpcloader.gbl --app SLSTK3701A_micriumos_httpcloader.s37
```

6. Execute the following command in Simplicity Commander to create a signed and encrypted upgrade GBL file (`SLSTK3701A_micriumos_httpcloader_secure.gbl`). The firmware upgrade GBL file is ECDSA-P256 signed and AES-CTR-128 encrypted.

```
commander gbl create SLSTK3701A_micriumos_httpcloader_secure.gbl --app
SLSTK3701A_micriumos_httpcloader_signed.s37 --sign httpdevice-signing-key --encrypt httpdevice-encryption-
key
```

Note: The signing (`httpdevice-signing-key`) and encryption (`httpdevice-encryption-key`) keys are generated in 5.4 Request GBL file and upgrade application steps 7 and 8

7. Upload the GBL image to the server indicated by the user.
8. Repeat the same procedure in 5.4 Request GBL file and upgrade application to upgrade the application image. For step 2, use the URL field that contains the new uploaded GBL image. Users should get similar terminal output as Figure 5.12 Commander Interface on page 23.
9. The application version number should update to the version number in `application_properties.c` that user configured in step 3 of this section.

6. Pre-Programmed Device

Silicon Labs offers pre-programmed devices for custom Gecko Bootloader and application. To do this, the binary or hex file must be provided. This option is subject to minimum order quantities (MOQ) and an additional cost. For this option, contact your local sales representative (http://www.silabs.com/buysample/pages/contact-sales.aspx?view=map).

7. Revision History

Revision 0.2

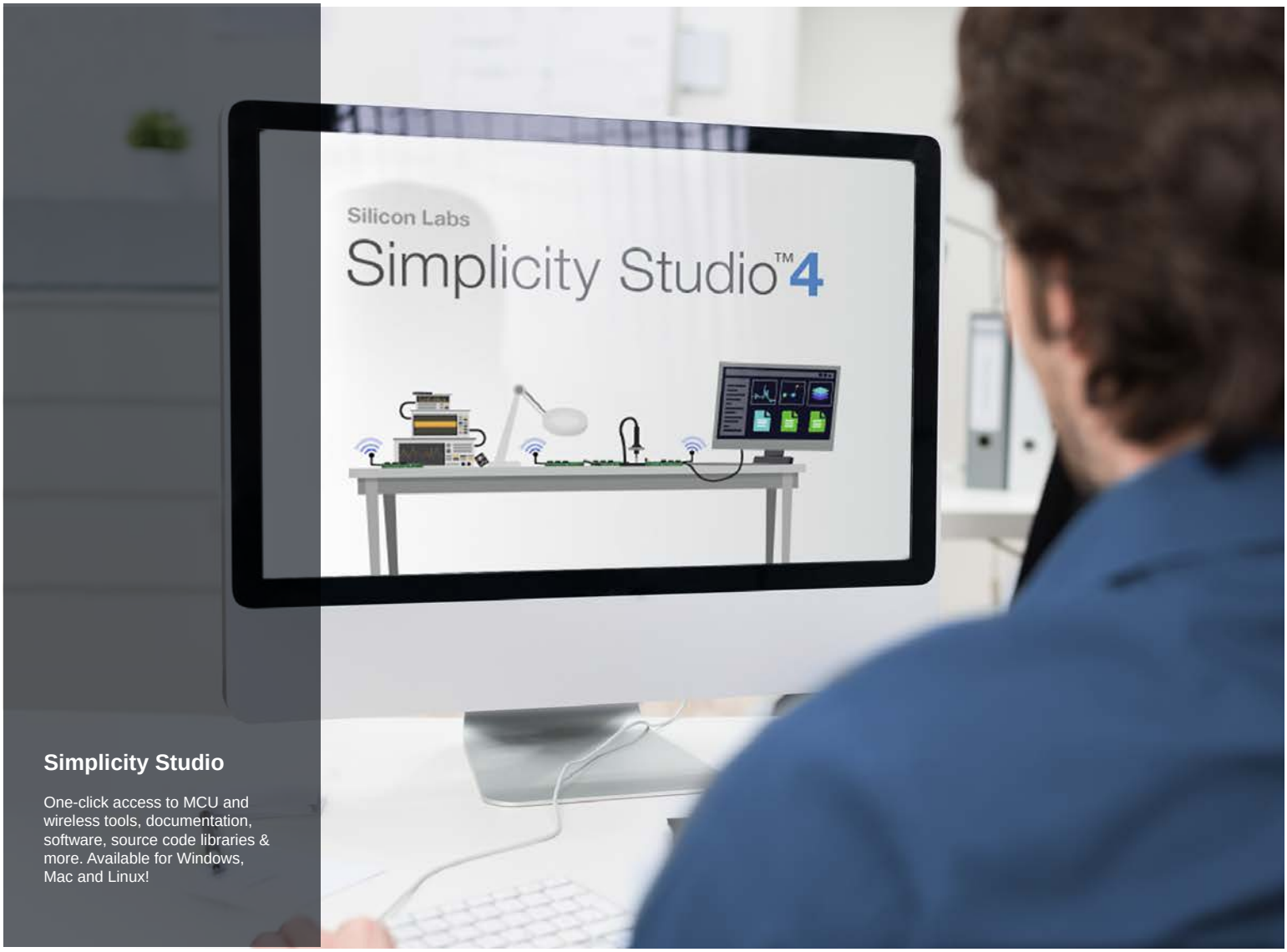
Sepetember, 2019

- Updated security feature option in the bootloader configurator
- Updated HTTP Bootloader Demo
- Updated instructions on how to secure boot/upgrade device

Revision 0.1

May, 2019

- Initial Revision



Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com