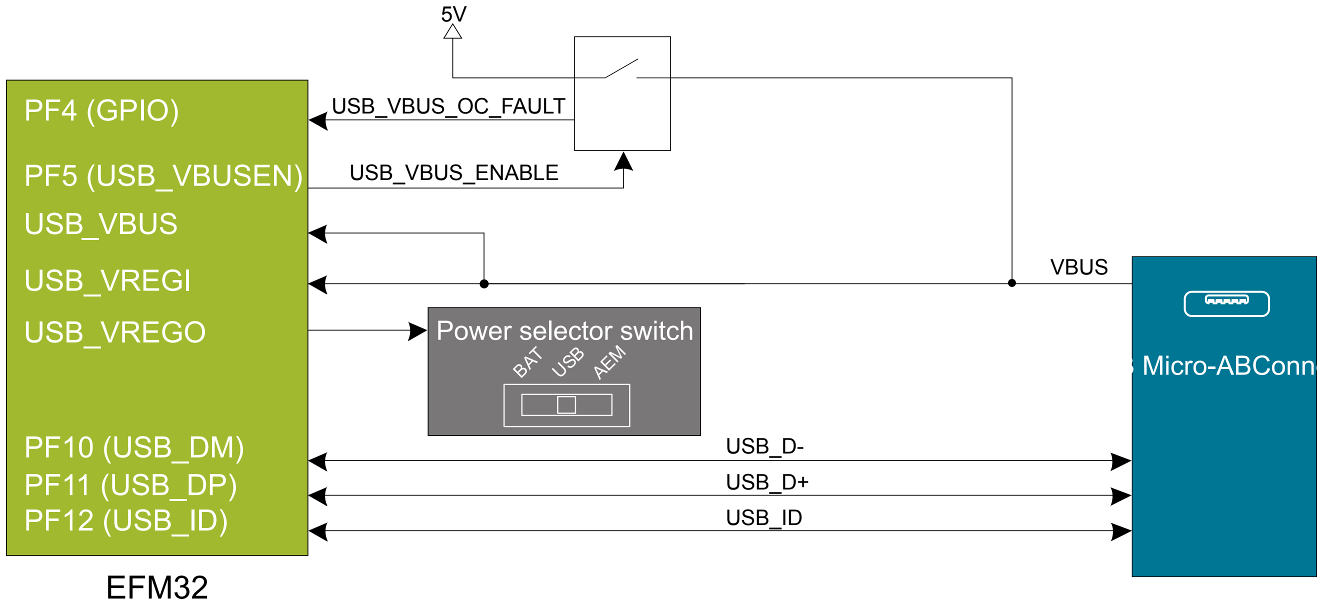
AN1204: USB Device/Host MSD Application Bootloader

This application note uses the EFM32 USB device or host protocol stack to implement a Mass Storage Device (MSD) class loader for Gecko Bootloader (GBL). The GBL image file used for firmware upgrade is stored in the MSD. The USB Device/Host MSD Application Bootloader shows how to customize and add application-level functionality to the Gecko Bootloader.

For more information on using the EFM32 USB and Gecko Bootloader, refer to the 2. Relevant Documentation and Software Modules section in this application note.

KEY POINTS

- MCU Series 1 Gecko Bootloader.
- USB Device MSD Application Bootloader.
  - Makes Start Kit appear as a MSD device
  - MSD is used to store a GBL file for firmware upgrade
- USB Device sample application
- USB Host MSD Application Bootloader.
  - Detects USB MSD device connection
  - Searches valid GBL file in MSD for firmware upgrade
- USB Host sample application



1. Device Compatibility

This application note supports USB enabled MCU Series 1 device families.

MCU Series 1 consists of the following:

- EFM32 Giant Gecko GG11 (EFM32GG11)
- EFM32 Giant Gecko GG12 (EFM32GG12)

2. Relevant Documentation and Software Modules

The documents in Table 2.1 Relevant Documentation for the USB Device/Host and Gecko Bootloader on page 3 are available on https://www.silabs.com/support/resources and https://www.silabs.com/support/resources.ct-manuals_user-guides.p-wireless_bluetooth-low-energy.

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

Table 2.1. Relevant Documentation for the USB Device/Host and Gecko Bootloader

Application Note/ User's Guide	Applicable Device	Description
AN0002.1	MCU Series 1	*EFM32 and EFR32 Wireless Gecko Series 1 Hardware Design Considerations* — The USB sections illustrate several different configurations for connecting and decoupling the USB power, signalling, and control signals.
AN0003	MCU Series 0 MCU Series 1	*UART Bootloader* — All EFM32 devices are pre-programmed with UART bootloader, it will be overwritten by Gecko Bootloader in this application note.
AN0042	USB enabled MCU Series 0	*USB/UART Bootloader* — For MCU Series 1, the USB Device Bootloader is implemented by USB Device MSD Application Bootloader in this application note.
AN0046[1]	USB enabled MCU Series 0	*USB Hardware Design Guide* — This application note gives recommendations on hardware design for implementing USB host and device applications using USB capable EFM32 microcontrollers.
AN0052	USB enabled MCU Series 0	*USB MSD Host Bootloader* — For MCU Series 1, the USB MSD Host Bootloader is implemented by USB Host MSD Application Bootloader in this application note.
AN0065[1]	USB enabled MCU Series 0	*EFM32 as a USB Device* — This application note introduces the EFM32 or EZR32 USB Device stack and explains how to configure the MCU to act as a USB Device.
AN0801[1]	USB enabled MCU Series 0	*EFM32 as USB Host* — This application note introduces the EFM32 USB Host stack and explains how to configure the EFM32 as a USB Host.
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 USB application notes are intended for USB enabled MCU Series 0 but most of the materials can apply to USB enabled MCU Series 1.

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 USB Device/Host MSD Application Bootloader on page 4 (where vX.Y is the Gecko SDK version number).

Table 2.2. Relevant Software Modules for the USB Device/Host MSD Application 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/
Gecko USB	`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\platform\middleware\usb_gecko` https://docs.silabs.com/mcu/latest/efm32gg11/group-USB
FatFs	`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\util\third_party\fatfs`

3. MCU Series 1 Gecko Bootloader

3.1 Memory Layout

The memory layout of the USB Device/Host MSD Application Bootloader is shown in Figure 3.1 USB Device/Host MSD Application Bootloader Memory Layout on page 5, only a single storage slot is supported. The size of the main and bootloader block is device dependent, the main block is mapped to address `0x00000000` and the bootloader block is mapped to address `0x0FE10000`.

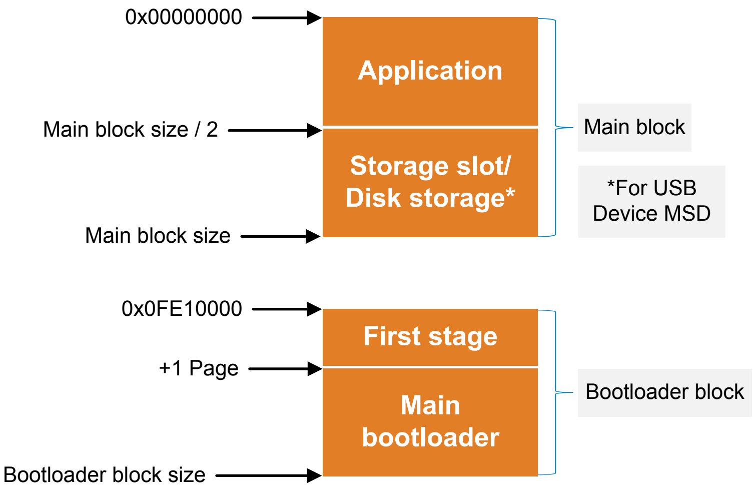


Figure 3.1. USB Device/Host MSD Application Bootloader Memory Layout

Note: The MCU Series 1 devices are shipped with the *AN0003: UART Bootloader* from the factory. Users must program the Gecko Bootloader to bootloader block (3.3 Programming the Gecko Bootloader) on USB enabled MCU Series 1 devices.

3.2 Security Feature

The Gecko Bootloader security features are described in Table 3.1 Gecko Bootloader Security Feature on page 5. Refer to step 8 of 3.3 Programming the Gecko Bootloader to enable or disable the Gecko Bootloader security features.

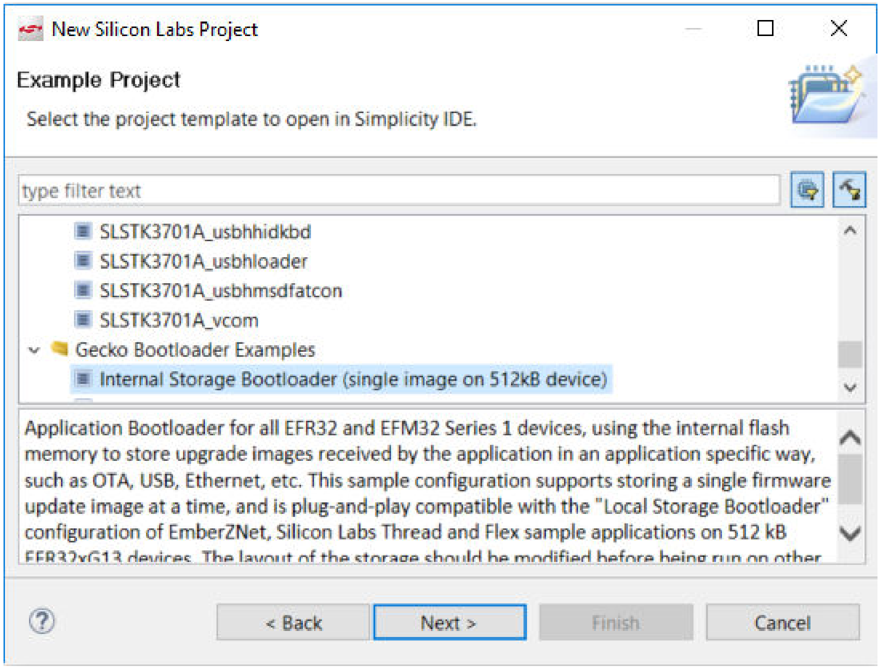
Table 3.1. Gecko Bootloader Security Feature

Security Feature	Advantage	Disadvantage
Require signed firmware upgrade files.	Protect against untrusted application updates.	Added GBL update time and key management complexity.
Require encrypted firmware upgrade files.	Protect software IP or private data within the image.	Added GBL update time and key management complexity.
Enable secure boot.	Prevent execution of untrusted images.	Adds application start-up time.
Prevent bootloader write/erase.	Protects the bootloader itself against malware in the application.	Limitations on debug access (rewrite) of main bootloader. Simplicity Commander or debugger is unable to erase bootloader space.

3.3 Programming the Gecko Bootloader

This section describes how to build and program a Gecko Bootloader from one of the provided EFM32GG11 Giant Gecko Starter Kit (STK) examples. The instructions assume that you have installed the required Silicon Labs Gecko SDK and that you are familiar with generating, compiling, and flashing an example application.

1. Connect EFM32GG11 STK to your computer and start Simplicity Studio.
2. Click the **EFM32GG11 Giant Gecko Starter Kit (SLSTK3701A)** from the [**Debug Adapters**] tab. This will verify that the installation was successful, identify the device on the kit hardware, and automatically configure the software tools for use with your device.
3. From the [**Launcher**] perspective, click [**New Project**].
4. In the [**Example Project**] dialog, select the **Internal Storage Bootloader (single image on 512kB device)** sample application under **Gecko Bootloader Examples** and click [**Next >**].



New Silicon Labs Project

Example Project

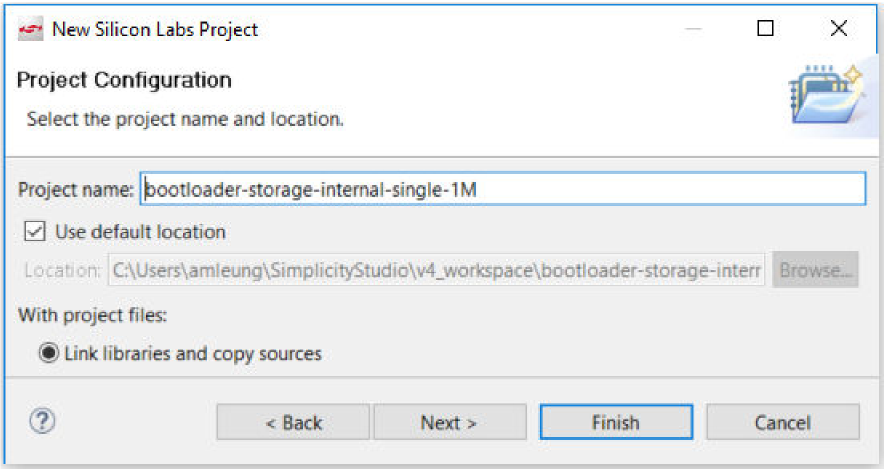
Select the project template to open in Simplicity IDE.

type filter text

- SLSTK3701A_usbhhidkbd
- SLSTK3701A_usbhloader
- SLSTK3701A_usbhmsdfatcon
- SLSTK3701A_vcom
- Gecko Bootloader Examples
  - Internal Storage Bootloader (single image on 512kB device)

Application Bootloader for all EFR32 and EFM32 Series 1 devices, using the internal flash memory to store upgrade images received by the application in an application specific way, such as OTA, USB, Ethernet, etc. This sample configuration supports storing a single firmware update image at a time, and is plug-and-play compatible with the "Local Storage Bootloader" configuration of EmberZNet, Silicon Labs Thread and Flex sample applications on 512 kB EFR32xG13 devices. The layout of the storage should be modified before being run on other

5. In the [**Project Configuration**] dialog, name your project (e.g. `bootloader-storage-internal-single-1M`) and optionally select a different project location. Click [**Next >**].



New Silicon Labs Project

Project Configuration

Select the project name and location.

Project name: bootloader-storage-internal-single-1M

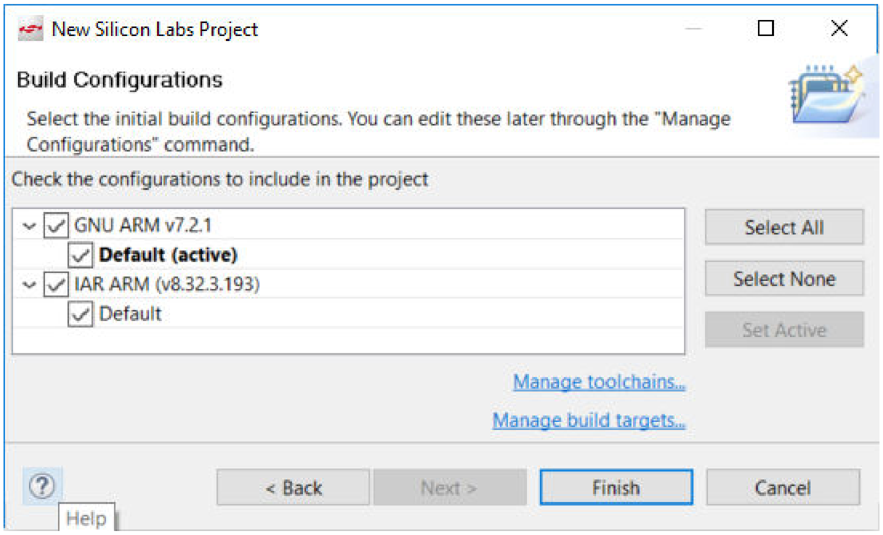
☑ Use default location

Location: C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-interr

With project files:

● Link libraries and copy sources

6. In the [**Build Configurations**] dialog, select your compiler (in general, the same compiler you will use for the application). Click [**Finish**] to create the Gecko Bootloader project in Simplicity IDE and AppBuilder perspective (`bootloader-storage-internal-single-1M.isc`) is displayed.



New Silicon Labs Project

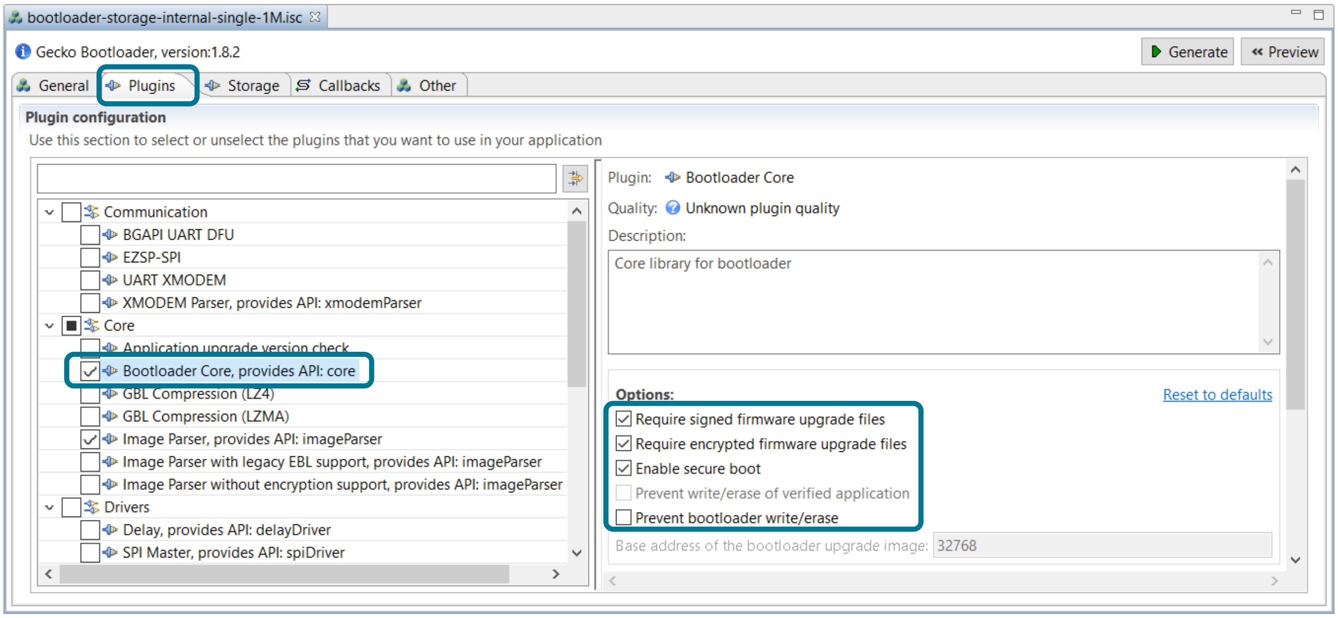
Build Configurations

Select the initial build configurations. You can edit these later through the "Manage Configurations" command.

Check the configurations to include in the project

- ☑ GNU ARM v7.2.1
  - ☑ **Default (active)**
- ☑ IAR ARM (v8.32.3.193)
  - ☑ Default

7. On the [**General**] tab, optionally enter a description. Go to step 9 if Gecko Bootloader security features are not required.
8. Click [**Plugins**] tab, select [**Core**] > [**Bootloader Core, provides API: core**], check all except **Prevent bootloader write/erase** option.



bootloader-storage-internal-single-1M.isc

Gecko Bootloader, version:1.8.2

Plugin configuration

Use this section to select or unselect the plugins that you want to use in your application

- □ Communication
  - □ BGAPI UART DFU
  - □ EZSP-SPI
  - □ UART XMODEM
  - □ XMODEM Parser, provides API: xmodemParser
- ■ Core
  - □ Application upgrade version check
  - ☑ Bootloader Core, provides API: core
  - □ GBL Compression (LZ4)
  - □ GBL Compression (LZMA)
  - ☑ Image Parser, provides API: imageParser
  - □ Image Parser with legacy EBL support, provides API: imageParser
  - □ Image Parser without encryption support, provides API: imageParser
- □ Drivers
  - □ Delay, provides API: delayDriver
  - □ SPI Master, provides API: spiDriver

Plugin: Bootloader Core

Quality: Unknown plugin quality

Description:

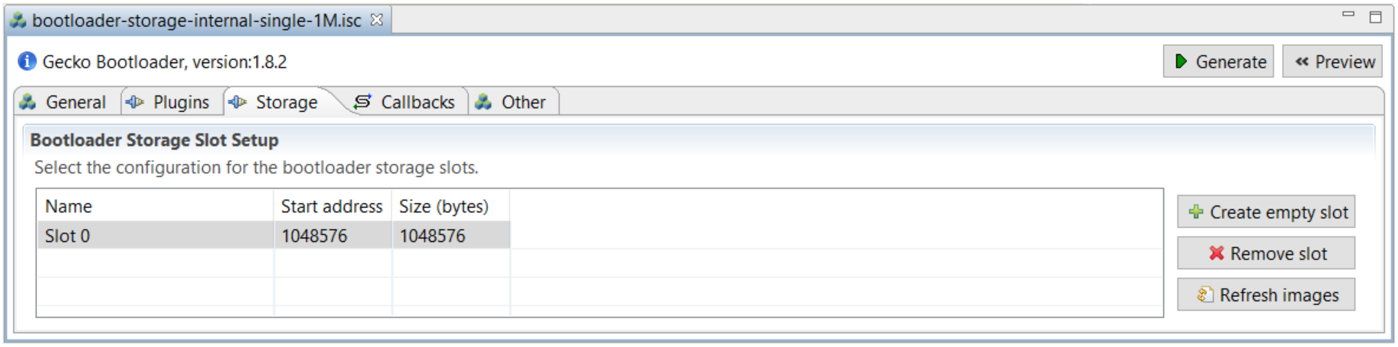
Core library for bootloader

Options:

- ☑ Require signed firmware upgrade files
- ☑ Require encrypted firmware upgrade files
- ☑ Enable secure boot
- □ Prevent write/erase of verified application
- □ Prevent bootloader write/erase

Base address of the bootloader upgrade image: 32768

9. Click on the [**Storage**] tab to modify the default configuration of the bootloader storage slot. Change the **Start address** and **Size (bytes)** to `1048576` (EFM32GG11B820F2048GL192 on Starter Kit has 2 MB main block flash and the storage slot is half of it).

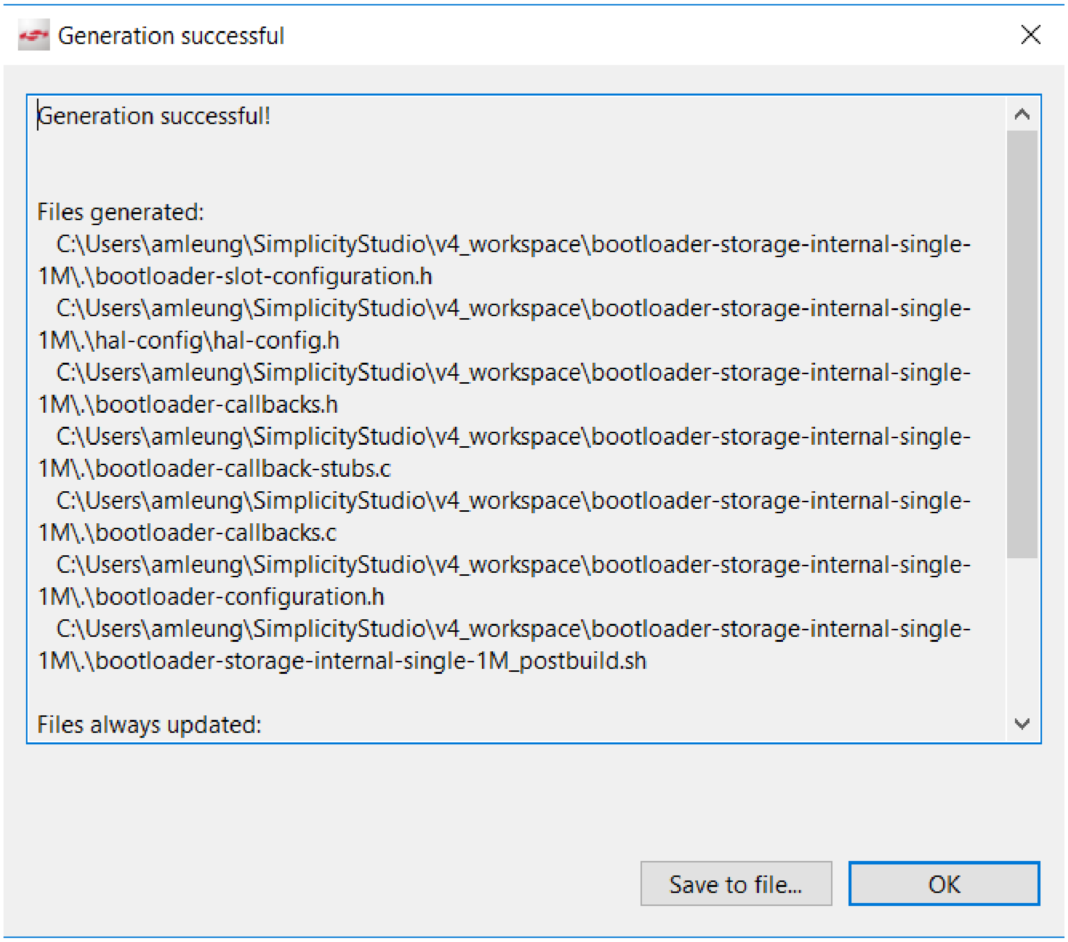


Bootloader Storage Slot Setup

Select the configuration for the bootloader storage slots.

Name	Start address	Size (bytes)
Slot 0	1048576	1048576

10. Click [**Generate**] button at the top rightmost corner.
11. In the [**Generation successful**] dialog, click [**OK**].

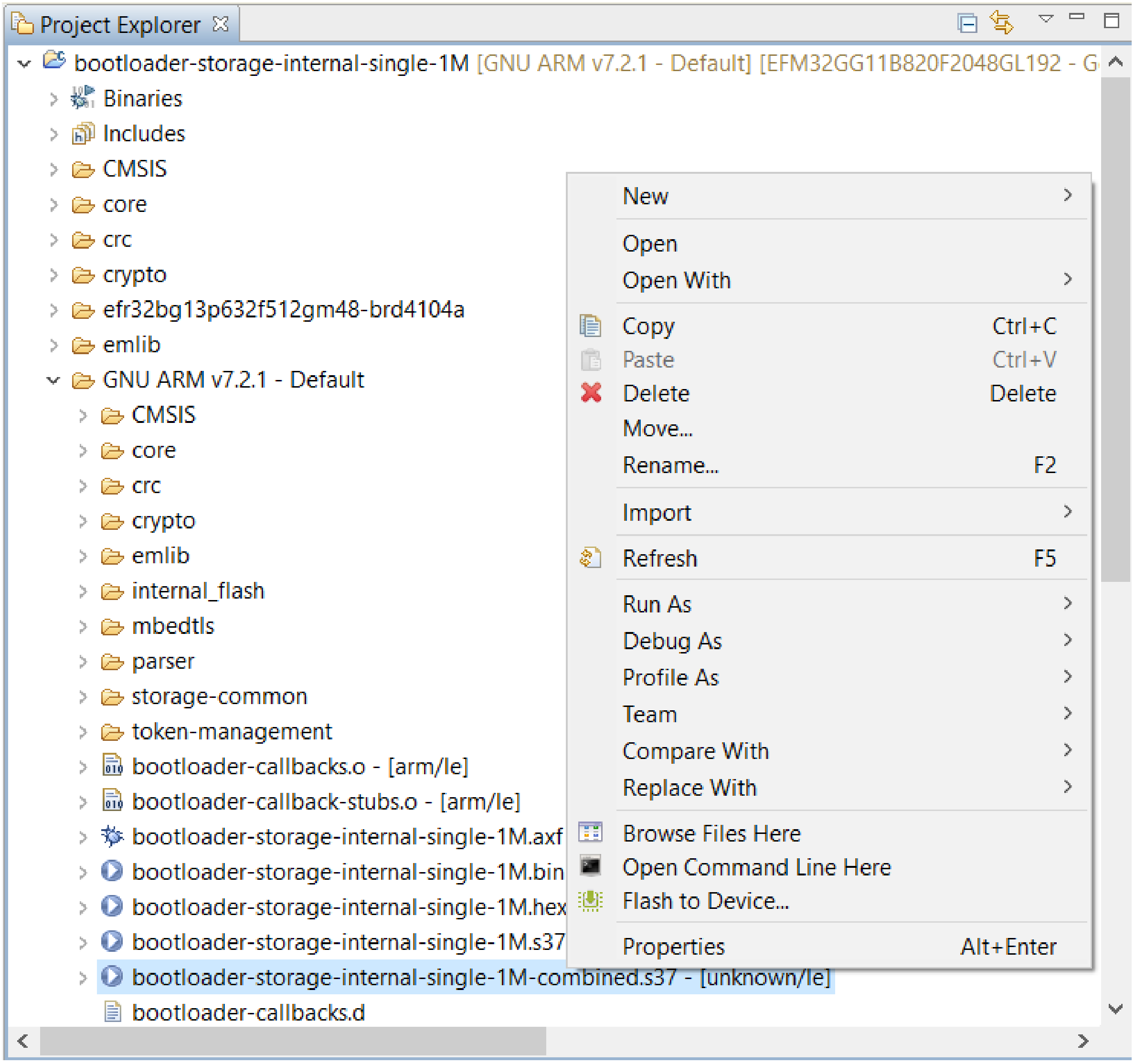


```
Generation successful!


Files generated:
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\bootloader-slot-configuration.h
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\hal-config\hal-config.h
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\bootloader-callbacks.h
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\bootloader-callback-stubs.c
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\bootloader-callbacks.c
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\bootloader-configuration.h
  C:\Users\amleung\SimplicityStudio\v4_workspace\bootloader-storage-internal-single-
1M\.\bootloader-storage-internal-single-1M_postbuild.sh

Files always updated:
```

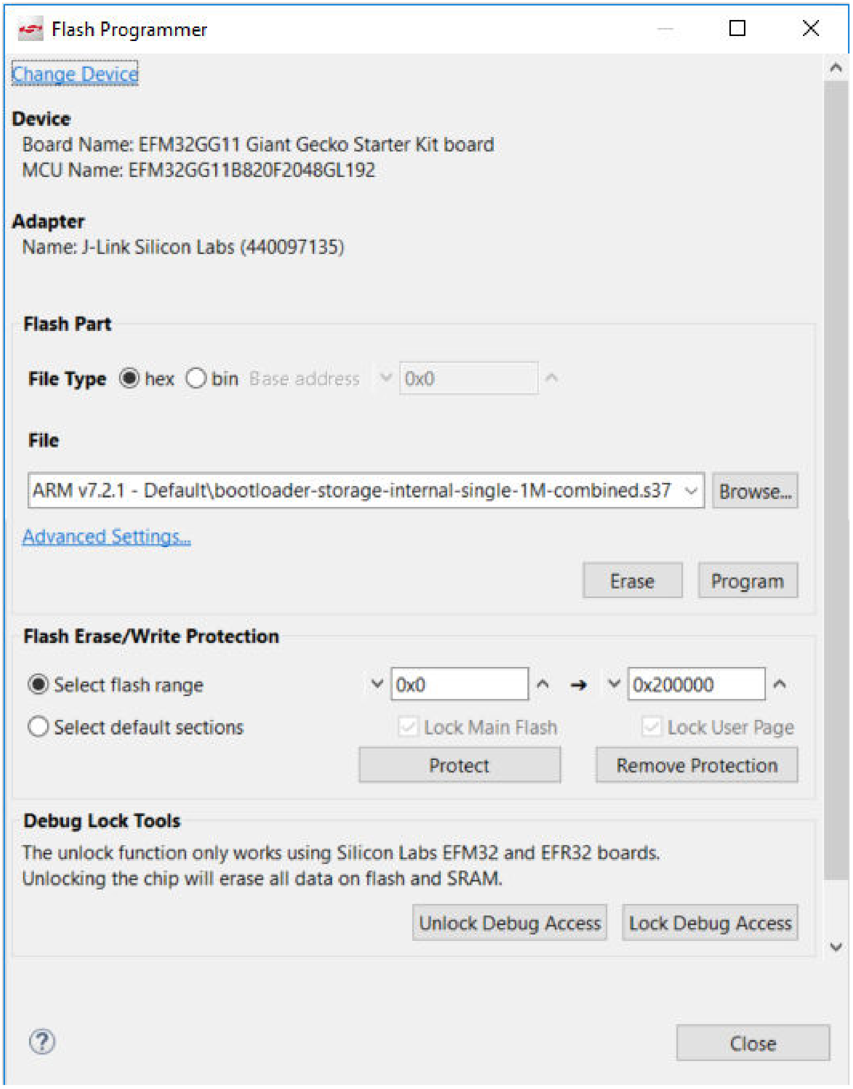
12. Click the [**Build**] icon ( ). On MCU Series 1 devices, two bootloader images are generated into the build directory: a main bootloader and a combined first stage and main bootloader. The main bootloader image is called `bootloader-storage-internal-single-1M.s37`, while the combined first stage + main bootloader image is called `bootloader-storage-internal-single-1M-combined.s37`.

    Note: The first time a device is programmed, whether during development or manufacturing, the combined image needs to be programmed. For subsequent programming, when a first stage bootloader is already present on the device, the image containing only a main bootloader may be used.

13. Right click the `bootloader-storage-internal-single-1M-combined.s37` file under build folder in Project Explorer, click [**Flash to Device...**] in the context menu.


Project Explorer
bootloader-storage-internal-single-1M [GNU ARM v7.2.1 - Default] [EFM32GG11B820F2048GL192 - G
Binaries
Includes
CMSIS
core
crc
crypto
efr32bg13p632f512gm48-brd4104a
emlib
GNU ARM v7.2.1 - Default
CMSIS
core
crc
crypto
emlib
internal_flash
mbedtls
parser
storage-common
token-management
bootloader-callbacks.o - [arm/le]
bootloader-callback-stubs.o - [arm/le]
bootloader-storage-internal-single-1M.axf
bootloader-storage-internal-single-1M.bin
bootloader-storage-internal-single-1M.hex
bootloader-storage-internal-single-1M.s37
bootloader-storage-internal-single-1M-combined.s37 - [unknown/le]
bootloader-callbacks.d
New
Open
Open With
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Move...
Rename... F2
Import
Refresh F5
Run As
Debug As
Profile As
Team
Compare With
Replace With
Browse Files Here
Open Command Line Here
Flash to Device...
Properties Alt+Enter

14. This will open the Flash Programmer. Click [**Program**] button to flash the `bootloader-storage-internal-single-1M-combined.s37` file to the kit.



Device
Board Name: EFM32GG11 Giant Gecko Starter Kit board
MCU Name: EFM32GG11B820F2048GL192

Adapter
Name: J-Link Silicon Labs (440097135)

Flash Part

File Type ● hex ○ bin Base address 0x0

File
ARM v7.2.1 - Default\bootloader-storage-internal-single-1M-combined.s37

Flash Erase/Write Protection

● Select flash range 0x0 ➜ 0x200000
○ Select default sections ☑ Lock Main Flash ☑ Lock User Page

Debug Lock Tools
The unlock function only works using Silicon Labs EFM32 and EFR32 boards.
Unlocking the chip will erase all data on flash and SRAM.

15. The STK is now ready for sample applications on 4.5 USB Device Sample Application and 5.5 USB Host Sample Application.

4. USB Device MSD Application Bootloader

4.1 Hardware Overview

The EFM32GG11 Giant Gecko Start Kit (SLSTK3701A_EFM32GG11) is used as the hardware platform of the USB Device MSD Application Bootloader. Resources of Starter Kit used by the device application bootloader is shown in Table 4.1 Hardware Resources Used by the USB Device MSD Application Bootloader on page 11.

Table 4.1. Hardware Resources Used by the USB Device MSD Application Bootloader

Clock/Peripheral	Description
HFXO	The on-board 50 MHz HFXO crystal is used as HFCLK.
LFRCO	The LFRCO is used as USB Low Energy Mode (LEM) clock.
USHFRCO	For USB device mode, the USB is clocked from its own internal oscillator USHFRCO.
AUXHFRCO	The AUXHFRCO is required on SWO trace output for Energy Profiler in Simplicity Studio. It is optional.
USB	Configures as USB device. • PF10 — USB_DM#0 (USB D- pin) • PF11 — USB_DP#0 (USB D+ pin)
TIMER0	Hardware timer for USB device stack (default is TIMER0).
USART4	The serial port for on-board USB virtual COM port (CDC). • PH4 — US4_TX#4 • PH5 — US4_RX#4 • PE1 — On-board USB virtual COM port enable
GPIO	The on-board LED0 (GPIO PH10) is used as USB activity indicator. It is optional.

4.2 USB Device Configuration

The application must provide a header file named usbconfig.h (default location on Windows is C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\usbdloader) to configure the USB device stack.

Table 4.2. USB Device Configuration for USB Device MSD Application Bootloader

Define (#define)[1]	Parameter/Setting	Description
USB_DEVICE[2]	—/—	Compiles the USB stack for device mode.
USB_CLKSRC_USHFRCO[2]	—/—	Selects clock source to clock the USB peripheral, it must be 48 MHz (2500 ppm).
NUM_EP_USED[2]	number/2	Specifies number of endpoints used (in addition to EP0).
MSD_INTERFACE_NO [2]	number/0	USB interface number for USB Mass Storage Class device.
MSD_BULK_IN[2]	address/0x01	Endpoint address for data transmission.
MSD_BULK_OUT[2]	address/0x08	Endpoint address for data reception.
USB_TIMER	USB_TIMERn/(default USB_TIMER0)	Selects which hardware timer (n = 0, 1, 2, ...) the USB stack is allowed to use.
NUM_APP_TIMERS	number/(default 0)	Specifies number of software timers required by application.
USB_PWRSAVE_MODE	USB_PWRSAVE_MODE_X/(default OFF)	The flags (X = ...) below can be OR'ed together to select the energy saving modes. Default selection is to not use any energy saving modes (X = OFF). • USB_PWRSAVE_MODE_ONSUSPEND • USB_PWRSAVE_MODE_ONVBUSOFF • USB_PWRSAVE_MODE_ENTEREM2
USB_USBC_32kHz_CLK	USB_USBC_32kHz_CLK_X/(default LFXO)	Selects clock source (X = LFRCO or LFXO) when USB peripheral is in energy saving mode.
USB_USBLEM_CLK	USB_USBLEM_CLK_X/(default LFRCO)	Selects clock source (X = LFRCO or LFXO) for Low Energy Mode (LEM), which is activated by default.
BUSPOWERED	—/—	To build bus-powered device; otherwise the device is self-powered.
DEBUG_USB_API	—/—	Turns on API debug diagnostics.
USB_USE_PRINTF	—/—	Enables utility print functions.

Note:

1. If parametric item is not specified in usbconfig.h, default setting is used.
2. These defines are mandatory for USB device.

4.3 Software Component

The following software components are used in USB Device MSD Application Bootloader.

1. Gecko Bootloader
    - Checks presence of the bootloader
    - Gets the bootloader storage space information
    - Reads/writes to the storage space
    - Verifies the GBL image in the storage space
    - Checks the version of the application stored in the storage space
    - Enters the bootloader firmware upgrade mode
2. Gecko USB
    - Initializes MSD device
    - Initializes and starts USB device stack
    - Makes Start Kit appear as a MSD device

4.4 Program Flow

The USB Device MSD Application Bootloader program flow is illustrated in Figure 4.1 USB Device MSD Application Bootloader Program Flow on page 14.

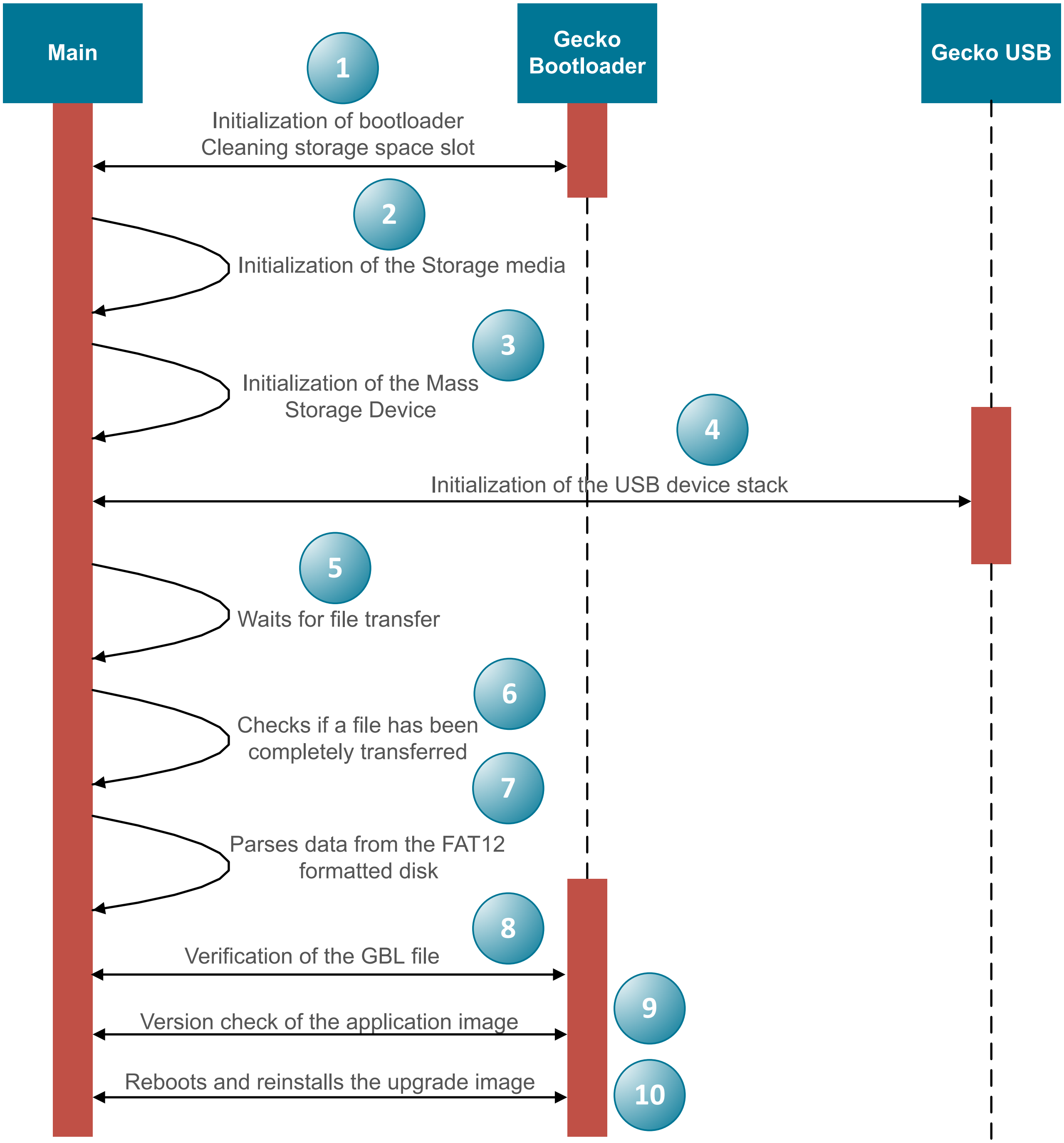

Main
Gecko Bootloader
Gecko USB
1
Initialization of bootloader
Cleaning storage space slot
2
Initialization of the Storage media
3
Initialization of the Mass Storage Device
4
Initialization of the USB device stack
5
Waits for file transfer
6
Checks if a file has been completely transferred
7
Parses data from the FAT12 formatted disk
8
Verification of the GBL file
9
Version check of the application image
10
Reboots and reinstalls the upgrade image

Figure 4.1. USB Device MSD Application Bootloader Program Flow

1. Initialization of bootloader
   - Presence of the bootloader — single internal storage
   - Bootloader information — storage space base-address and its size
   - Version of the running application
   - To clean storage space slot — Memory System Controller (MSC) erase starting from base address of the storage area
   - **Failure case** — Gecko bootloader is missing

2. Initializes the storage media interface
    - Uses the bootloader storage space as the disk space
    - Total number of sectors is calculated based on the bootloader storage area
    - Initializes Logic Block Addressing (LBA) table
    - Prepares FAT12 image (boot block) on the disk
3. Initializes MSD device
4. Initializes and starts Gecko USB device stack
    - Once the Starter Kit is connected to a computer via USB, it appears as a MSD device with a FAT12 formatted disk
5. Waits for a file to be dropped into the disk
    - Polled operation using MSD state machine
6. Checking if file transfer to the disk has been completed
    - Checks Logic Block Addressing (LBA) table
    - Searches through the root directory. Looks for a file with the `.gbl` extension
    - Follows the cluster chain using FAT table and checks for EOF mark
    - **Failure case** — Disk is full
7. Parses the GBL file stored in the FAT12 formatted disk
    - File size from root directory
    - Follows the cluster chain using FAT table
    - Overwrites the data on-the-fly to the storage space
8. Verification of the GBL file
    - **Failure case** — Storage slot is cleaned and system reset if Gecko Bootloader (GBL) file is invalid
9. Checks the version of the upgrade image stored in the GBL file
    - **Failure case** — Storage slot is cleaned and system reset if version of the running application has higher or equal version number as the application stored in the GBL file
10. Reboots and reinstalls the upgrade image
    - Extracts the upgrade image from the GBL file

4.5 USB Device Sample Application

The USB Device MSD Application Bootloader software is found under the Simplicity Studio installation path. The default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\usbdloader`

4.5.1 Flash Image File

This section describes how to flash a normal or secure USB device sample application image file to EFM32GG11 Giant Gecko Starter Kit and check the current version of the application.

1. Connect EFM32GG11 STK to your computer (`bootloader-storage-internal-single-1M-combined.s37` file was already flashed to STK in 3.3 Programming the Gecko Bootloader).
2. Copy the `usbdloader.bin` file from the SLSTK3701A_EFM32GG11 example directory (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\usbdloader\bin`) to Simplicity Commander folder (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\adapter_packs\commander`).
3. Go to step 5 if Gecko Bootloader security features (refer to step 8 in 3.3 Programming the Gecko Bootloader) are enabled.
4. Execute the following command in Simplicity Commander to flash the normal image file (`usbdloader.bin`). Go to step 10.

```
commander flash usbdloader.bin --address 0x0
```

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

```
commander gbl keygen –-type ecc-p256 --outfile usbdevice-signing-key
```

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

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

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

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

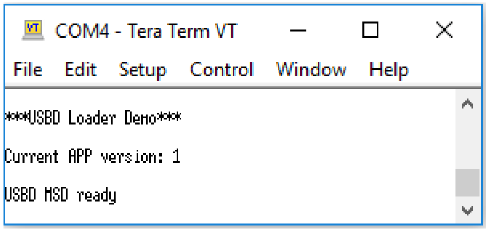
8. Execute the following command in Simplicity Commander to sign the application image (`usbdloader.bin`) to enable secure boot of the application image (`usbdloader_signed.s37`).

```
commander convert usbdloader.bin --secureboot --keyfile usbdevice-signing-key --outfile usbdloader_signed.s37
```

9. Execute the following command in Simplicity Commander to flash the secure image file (`SLSTK3701A_usbdloader_signed.s37`).

```
commander flash usbdloader_signed.s37
```

10. Open a terminal program (e.g. Tera Term) and access the STK Virtual COM port.
11. Press the RESET button on the Starter Kit. The version number of the running application will display on the terminal program.



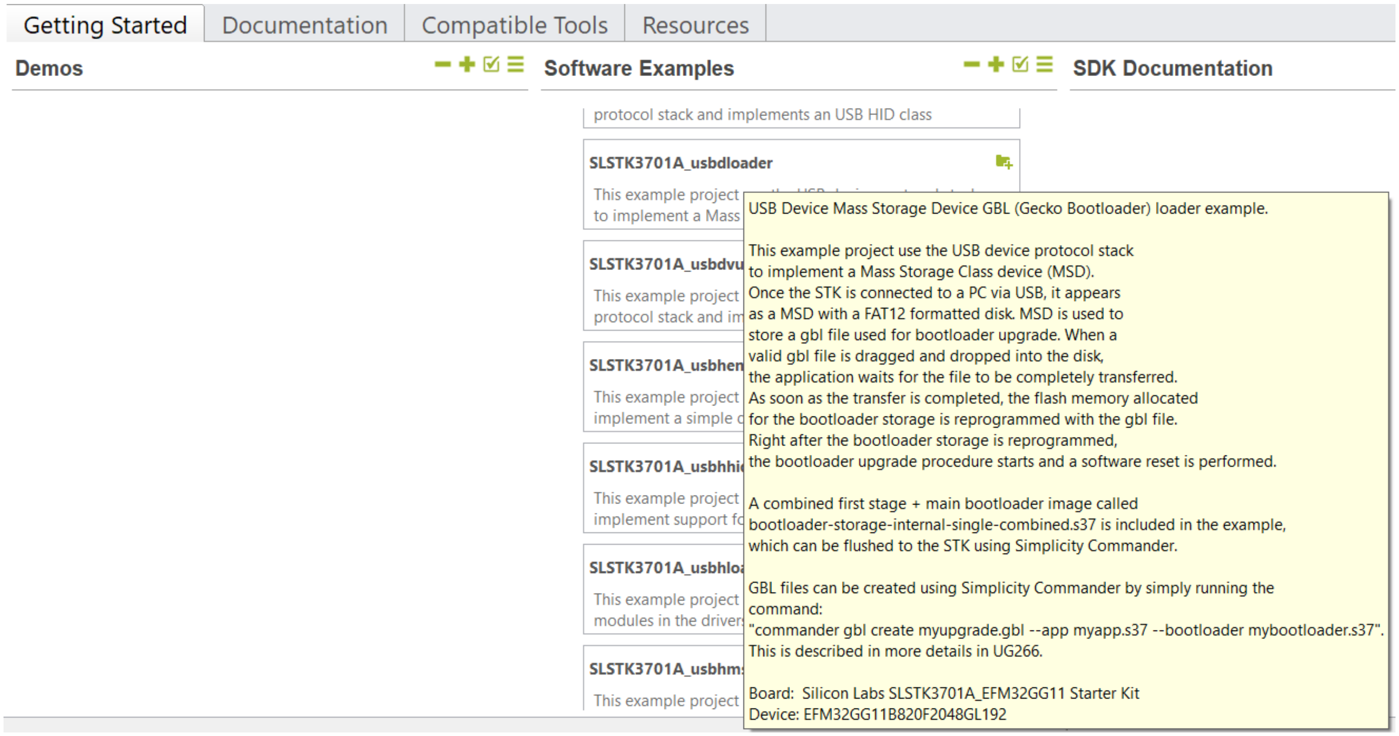
```
***USBD Loader Demo***
Current APP version: 1
USBD MSD ready
```

12. The STK is now ready for firmware upgrade on 4.5.2 Create GBL File.

4.5.2 Create GBL File

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

1. Connect EFM32GG11 STK to your computer and start Simplicity Studio.
2. Click the **EFM32GG11 Giant Gecko Starter Kit (SLSTK3701A)** from the [**Debug Adapters**] tab. This will verify that the installation was successful, identify the device on the kit hardware, and automatically configure the software tools for use with your device.
3. From the [**Launcher**] perspective, click [**EFM32GG11 Giant Gecko Starter Kit**] under [**Software Examples**] of [**Getting Started**] tab to browse target sample application.
4. Click the **SLSTK3701A_usbdloader** sample application to create the example project in Simplicity IDE.



USB Device Mass Storage Device GBL (Gecko Bootloader) loader example.

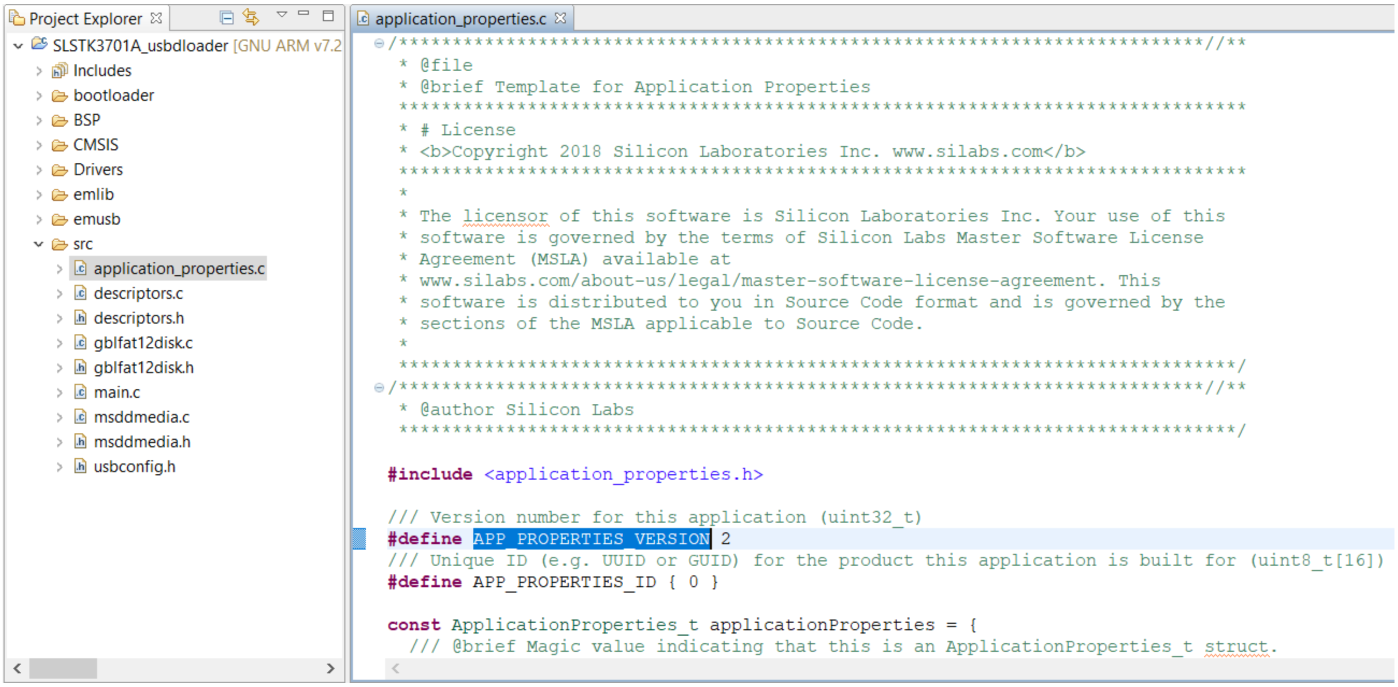
This example project use the USB device protocol stack
to implement a Mass Storage Class device (MSD).
Once the STK is connected to a PC via USB, it appears
as a MSD with a FAT12 formatted disk. MSD is used to
store a gbl file used for bootloader upgrade. When a
valid gbl file is dragged and dropped into the disk,
the application waits for the file to be completely transferred.
As soon as the transfer is completed, the flash memory allocated
for the bootloader storage is reprogrammed with the gbl file.
Right after the bootloader storage is reprogrammed,
the bootloader upgrade procedure starts and a software reset is performed.

A combined first stage + main bootloader image called
bootloader-storage-internal-single-combined.s37 is included in the example,
which can be flushed to the STK using Simplicity Commander.

GBL files can be created using Simplicity Commander by simply running the
command:
"commander gbl create myupgrade.gbl --app myapp.s37 --bootloader mybootloader.s37".
This is described in more details in UG266.

Board: Silicon Labs SLSTK3701A_EFM32GG11 Starter Kit
Device: EFM32GG11B820F2048GL192

5. Open `application_properties.c` file in **Project Explorer** and increase `APP_PROPERTIES_VERSION` by 1 (e.g. from 1 to 2).



```
/***************************************************************************//**
 * @file
 * @brief Template for Application Properties
 *******************************************************************************
 * # License
 * <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
 *******************************************************************************
 *
 * The licensor of this software is Silicon Laboratories Inc. Your use of this
 * software is governed by the terms of Silicon Labs Master Software License
 * Agreement (MSLA) available at
 * www.silabs.com/about-us/legal/master-software-license-agreement. This
 * software is distributed to you in Source Code format and is governed by the
 * sections of the MSLA applicable to Source Code.
 *
 ******************************************************************************/
/***************************************************************************//**
 * @author Silicon Labs
 ******************************************************************************/

#include <application_properties.h>

/// Version number for this application (uint32_t)
#define APP_PROPERTIES_VERSION 2
/// Unique ID (e.g. UUID or GUID) for the product this application is built for (uint8_t[16])
#define APP_PROPERTIES_ID { 0 }

const ApplicationProperties_t applicationProperties = {
  /// @brief Magic value indicating that this is an ApplicationProperties_t struct.
```

6. Click the [**Build**] icon ( ). Copy the `SLSTK3701A_usbdloader.s37` file from the build directory to Simplicity Commander folder (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\adapter_packs\commander`).
7. Go to step 9 if Gecko Bootloader security features (refer to step 8 in 3.3 Programming the Gecko Bootloader) are enabled.
8. Execute the following command in Simplicity Commander to create an upgrade GBL file (`SLSTK3701A_usbdloader.gbl`). Go to step 11 if the signing and encryption are not needed.

```
commander gbl create SLSTK3701A_usbdloader.gbl --app SLSTK3701A_usbdloader.s37
```

9. Execute the following command in Simplicity Commander to sign the application image (`SLSTK3701A_usbdloader.s37`) to enable secure boot of the application image (`SLSTK3701A_usbdloader_signed.s37`). The application image is signed using ECDSA-P256 and the signature is verified on every boot.

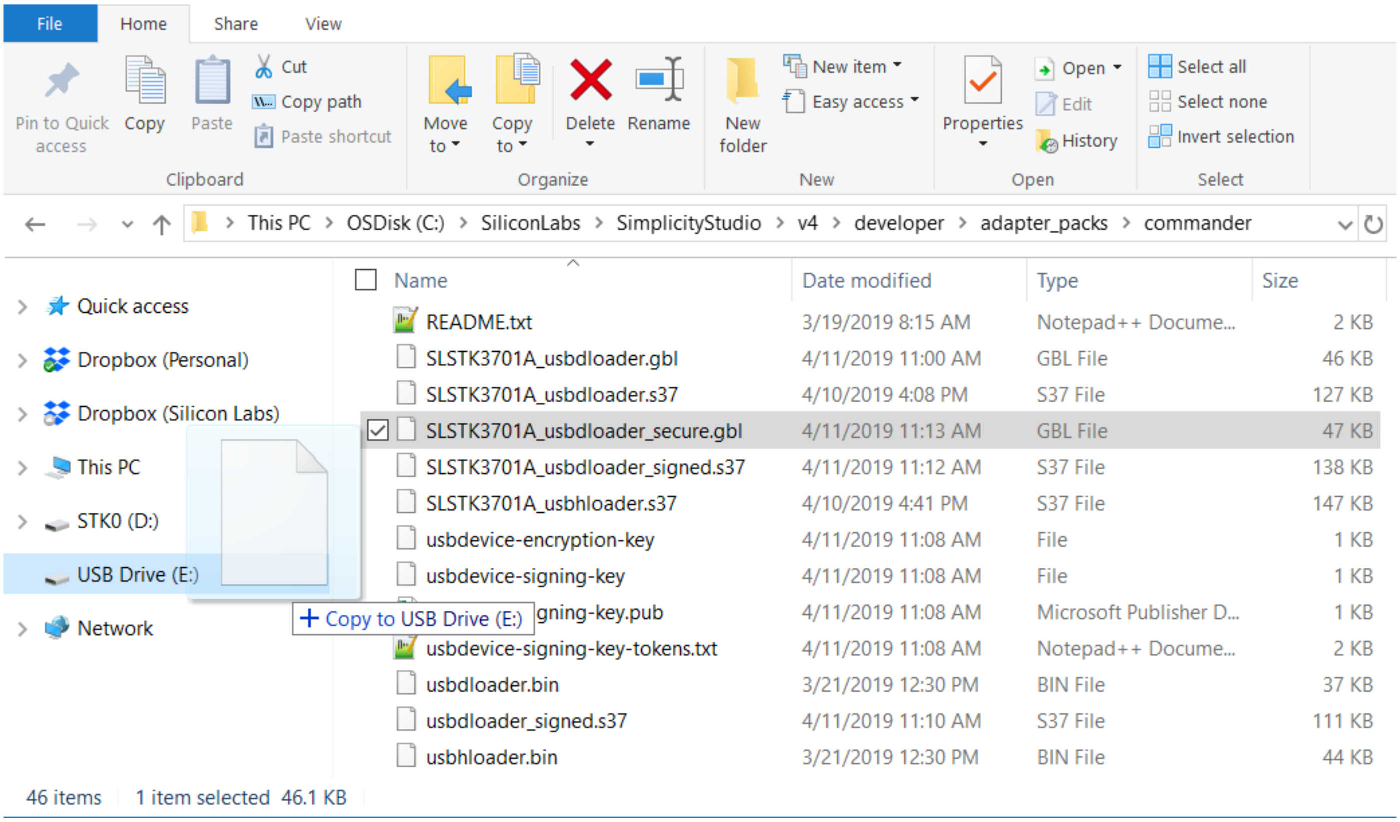
```
commander  convert  SLSTK3701A_usbdloader.s37  --secureboot  --keyfile  usbdevice-signing-key  --outfile
SLSTK3701A_usbdloader_signed.s37
```

10. Execute the following command in Simplicity Commander to create a signed and encrypted upgrade GBL file (`SLSTK3701A_usbdloader_secure.gbl`). The firmware upgrade GBL file is ECDSA-P256 signed and AES-CTR-128 encrypted.

```
commander  gbl  create  SLSTK3701A_usbdloader_secure.gbl  --app  SLSTK3701A_usbdloader_signed.s37  --sign
usbdevice-signing-key --encrypt usbdevice-encryption-key
```

Note: The signing (`usbdevice-signing-key`) and encryption (`usbdevice-encryption-key`) keys are generated in 4.5.1 Flash Image File step 5 and 6.

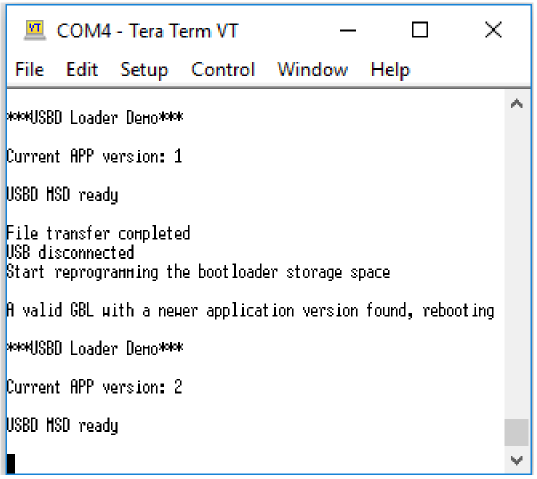
11. Open a terminal program (e.g. Tera Term) and access the STK Virtual COM port.
12. Press the RESET button on the Starter Kit. The version number of the running application will display on the terminal program.
13. Connect EFM32GG11 STK to your computer using USB 2.0 Type A – USB micro cable (USB Micro-AB connector is next to the Ethernet jack).
14. Drag-and-drop the normal `SLSTK3701A_usbdloader.gbl` file on step 8 (if GBL security features are disabled) or secure `SLSTK3701A_usbdloader_secure.gbl` file on step 10 (if GBL security features are enabled) into the STK (appeared as a USB MSD device on your computer).



Name	Date modified	Type	Size
README.txt	3/19/2019 8:15 AM	Notepad++ Docume...	2 KB
SLSTK3701A_usbdloader.gbl	4/11/2019 11:00 AM	GBL File	46 KB
SLSTK3701A_usbdloader.s37	4/10/2019 4:08 PM	S37 File	127 KB
☑ SLSTK3701A_usbdloader_secure.gbl	4/11/2019 11:13 AM	GBL File	47 KB
SLSTK3701A_usbdloader_signed.s37	4/11/2019 11:12 AM	S37 File	138 KB
SLSTK3701A_usbhloader.s37	4/10/2019 4:41 PM	S37 File	147 KB
usbdevice-encryption-key	4/11/2019 11:08 AM	File	1 KB
usbdevice-signing-key	4/11/2019 11:08 AM	File	1 KB
gning-key.pub	4/11/2019 11:08 AM	Microsoft Publisher D...	1 KB
usbdevice-signing-key-tokens.txt	4/11/2019 11:08 AM	Notepad++ Docume...	2 KB
usbdloader.bin	3/21/2019 12:30 PM	BIN File	37 KB
usbdloader_signed.s37	4/11/2019 11:10 AM	S37 File	111 KB
usbhloader.bin	3/21/2019 12:30 PM	BIN File	44 KB

46 items   1 item selected 46.1 KB

15. Use the GBL file to upgrade the application. The version of the application is updated (e.g. from 1 to 2) after the upgrade.



```
***USBD Loader Demo***

Current APP version: 1

USBD MSD ready

File transfer completed
USB disconnected
Start reprogramming the bootloader storage space

A valid GBL with a newer application version found, rebooting

***USBD Loader Demo***

Current APP version: 2

USBD MSD ready
```

5. USB Host MSD Application Bootloader

5.1 Hardware Overview

The EFM32GG11 Giant Gecko Start Kit (SLSTK3701A_EFM32GG11) is used as the hardware platform of the USB Host MSD Application Bootloader. Resources of Starter Kit used by the host application bootloader is shown in Table 5.1 Hardware Resources Used by the USB Host MSD Application Bootloader on page 20.

Table 5.1. Hardware Resources Used by the USB Host MSD Application Bootloader

Clock/Peripheral	Description
HFXO	The on-board 50 MHz HFXO crystal is used as HFCLK and DPLL reference clock.
DPLL	The Digital Phase-Locked Loop (DPLL) uses the HFRCO to generate a clock as a ratio of HFXO (reference clock source).
HFRCO	In USB host mode, a 48 MHz clock (2500ppm or better) is required. The HFRCO output frequency will be generated according to the DPLL configuration, which is 48 MHz for USBCLK.
AUXHFRCO	The AUXHFRCO is required on SWO trace output for Energy Profiler in Simplicity Studio. It is optional.
USB	Configures as USB host. • PF4 — Optional GPIO input pin for detection of VBUS over current or short circuit conditions • PF5 — USB_VBUSEN#0 (USB 5V VBUS enable) • PF10 — USB_DM#0 (USB D- pin) • PF11 — USB_DP#0 (USB D+ pin)
TIMER0	Hardware timer for USB host stack (default is TIMER0).
USART4	The serial port for on-board USB virtual COM port (CDC). • PH4 — US4_TX#4 • PH5 — US4_RX#4 • PE1 — On-board USB virtual COM port enable

5.2 USB Host Configuration

The application must provide a header file named `usbconfig.h` (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\usbhloader`) to configure the USB host stack.

Table 5.2. USB Host Configuration for USB Host MSD Application Bootloader

Define (#define)[1]	Parameter/Setting	Description
`USB_HOST`[2]	—/—	Compiles the USB stack for host mode.

`USB_CLKSRC_HFRCODPLL`[2]	—/—	Selects clock source to clock the USB peripheral, it must be 48 MHz (2500 ppm). Additional defines below are required if `USB_CLKSRC_HFRCODPLL` is selected.
		`// Using DPLL with 50 MHz HFXO as reference clock:`
		`#define USB_DPLL_FREQUENCY    48000000UL`
		`#define USB_DPLL_M            349U`
		`#define USB_DPLL_N            335U`
		`#define USB_DPLL_SRC USB_DPLL_SRC_HFXO`

`NUM_HC_USED`[2]	number/`2`	Specifies number of host channels used (in addition to EP0).
`USB_TIMER`	`USB_TIMERn`/(default `USB_TIMER0`)	Selects which hardware timer (n = `0`, `1`, `2`, ...) the USB stack is allowed to use.
`NUM_APP_TIMERS`	number/(default `0`)	Specifies number of software timers required by application.
`USB_VBUSOVRCUR_PORT`	`gpioPortn`/(default `gpioPortF`)	Specifies which GPIO port (n = `A`, `B`, `C`, ...) for detection of VBUS over current or short circuit conditions. Uses `USB_VBUSOVRCUR_PORT_NONE` as parameter if no over current circuitry in the hardware design.
`USB_VBUSOVRCUR_PIN`	n/(default `4`)	Specifies which GPIO pin (n = `0`, `1`, `2`, ...) for detection of VBUS over current or short circuit conditions.
`USB_VBUSOVRCUR_POLARITY`	`USB_VBUSOVRCUR_POLARITY_X`/(default `LOW`)	Specifies polarity (X = `LOW` or `HIGH`) for over current detection.
`DEBUG_USB_API`	—/—	Turns on API debug diagnostics.
`USB_USE_PRINTF`	—/—	Enables utility print functions.

Note:

1. If parametric item is not specified in `usbconfig.h`, default setting is used.
2. These defines are mandatory for USB host.

5.3 Software Component

The following software components are used in USB Host MSD Application Bootloader.

1. Gecko Bootloader
   - Checks presence of the bootloader
   - Gets the bootloader storage space information
   - Reads/writes to the storage space
   - Verifies the GBL image in the storage space
   - Checks the version of the application stored in the storage space
   - Enters the bootloader firmware upgrade mode
2. Gecko USB
   - Initializes host protocol stack data structures and makes Starter Kit appear as a USB host device
   - Detects USB MSD device connection
3. FatFS (third-party file system)
   - Navigates the memory stick file system
   - Reads the memory stick contents and parses the data stored in it

5.4 Program Flow

The USB Host MSD Application Bootloader program flow is illustrated in Figure 5.1 USB Host MSD Application Bootloader Program Flow on page 23.

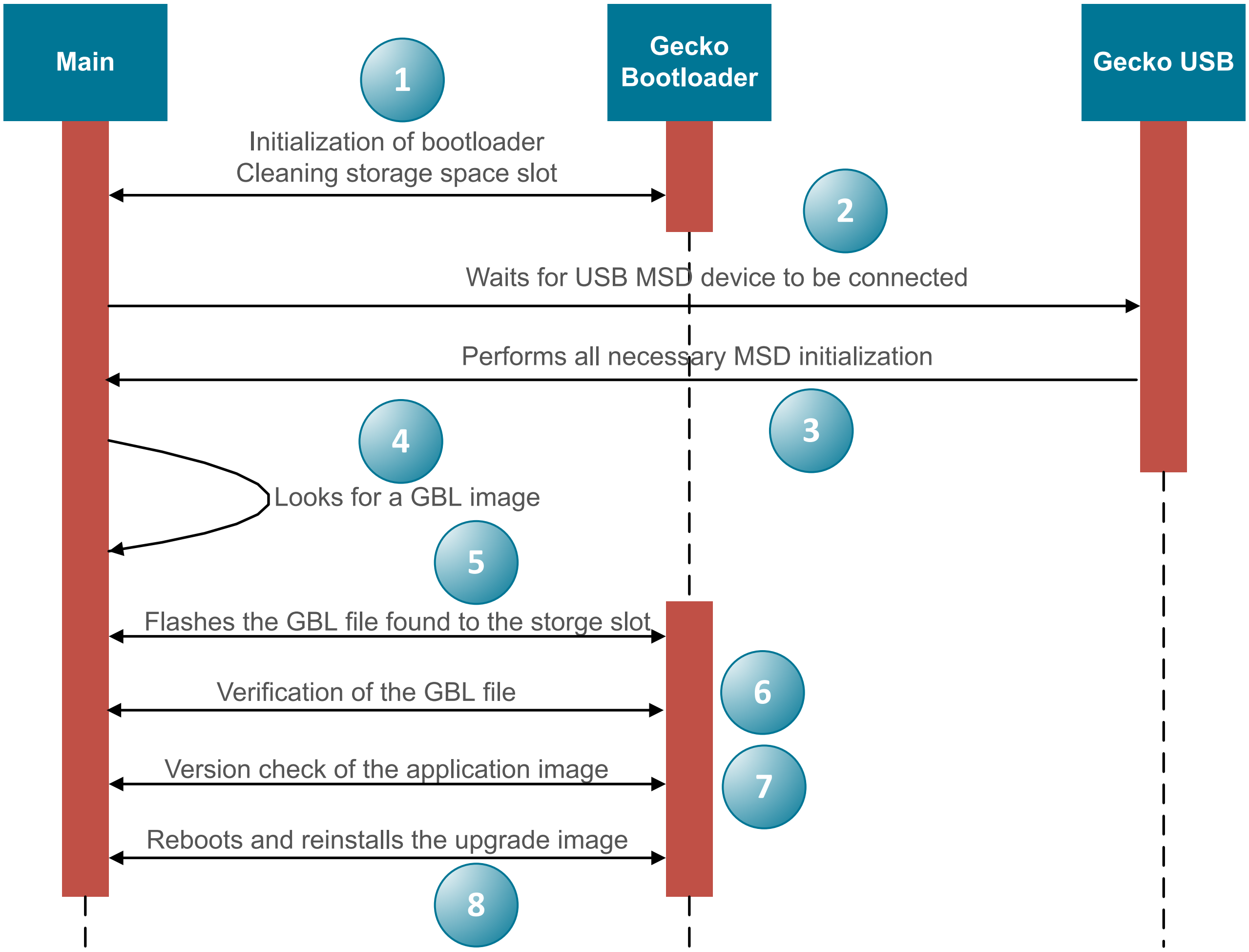

Main
Gecko Bootloader
Gecko USB
1
Initialization of bootloader
Cleaning storage space slot
2
Waits for USB MSD device to be connected
Performs all necessary MSD initialization
3
4
Looks for a GBL image
5
Flashes the GBL file found to the storge slot
Verification of the GBL file
6
Version check of the application image
7
Reboots and reinstalls the upgrade image
8

Figure 5.1. USB Host MSD Application Bootloader Program Flow

1. Initialization of bootloader
    - Presence of the bootloader — single internal storage
    - Bootloader information — storage space base-address and its size
    - Version of the running application
    - To clean storage space slot — Memory System Controller (MSC) erase starting from base address of the storage area
    - **Failure case** — Gecko bootloader is missing
2. Waits for USB MSD device plug-in
    - Polled operation on Gecko USB host stack
3. Initializes USB connected Mass Storage Device
    - Starter Kit appears as an USB MSD Host
4. Looks for a Gecko Bootloader file (third-party file system library FatFS) from the connected USB MSD device
    - **Failure case** — USB MSD device is un-mounted if Gecko Bootloader (GBL) file cannot be found
5. Flashes the GBL file found into the storage space
6. Verification of the GBL file
    - **Failure case** — Storage slot is cleaned and USB MSD device is un-mounted if Gecko Bootloader (GBL) file is invalid

7. Checks the version of the upgrade image stored in the GBL file
   - **Failure case** — Storage slot is cleaned and USB MSD device is un-mounted if version of the running application has higher or equal version number as the application stored in the GBL file
8. Reboots and reinstalls the upgrade image
   - Extracts the upgrade image from the GBL file

5.5 USB Host Sample Application

The USB Host MSD Application Bootloader software is found under the Simplicity Studio installation path. The default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\usbhloader`

5.5.1 Flash Image File

This section describes how to flash a normal or secure USB host sample application image file to EFM32GG11 Giant Gecko Starter Kit and check the current version of the application.

1. Connect EFM32GG11 STK to your computer (`bootloader-storage-internal-single-1M-combined.s37` file was already flashed to STK in 3.3 Programming the Gecko Bootloader).
2. Copy the `usbhloader.bin` file from the SLSTK3701A_EFM32GG11 example directory (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\vX.Y\app\mcu_example\SLSTK3701A_EFM32GG11\usbhloader\bin`) to Simplicity Commander folder (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\adapter_packs\commander`).
3. Go to step 5 if Gecko Bootloader security features (refer to step 8 in 3.3 Programming the Gecko Bootloader) are enabled.
4. Execute the following command in Simplicity Commander to flash the normal image file (`usbhloader.bin`). Go to step 10.

```
commander flash usbhloader.bin --address 0x0
```

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

```
commander gbl keygen –-type ecc-p256 --outfile usbhost-signing-key
```

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

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

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

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

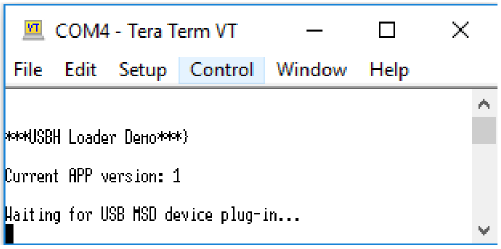
8. Execute the following command in Simplicity Commander to sign the application image (`usbhloader.bin`) to enable secure boot of the application image (`usbhloader_signed.s37`).

```
commander convert usbhloader.bin --secureboot --keyfile usbhost-signing-key --outfile usbhloader_signed.s37
```

9. Execute the following command in Simplicity Commander to flash the secure image file (`SLSTK3701A_usbhloader_signed.s37`).

```
commander flash usbhloader_signed.s37
```

10. Open a terminal program (e.g. Tera Term) and access the STK Virtual COM port.
11. Press the RESET button on the Starter Kit. The version number of the running application will display on the terminal program.



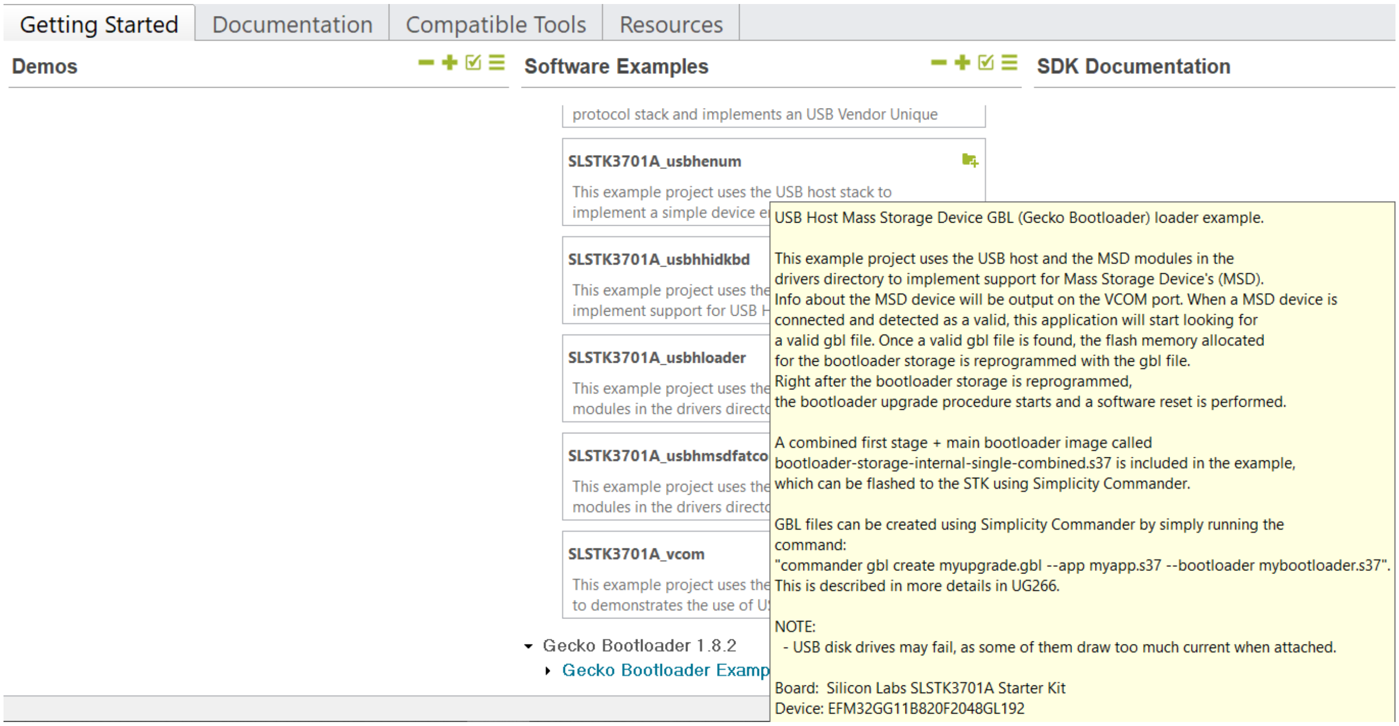
```
***USBH Loader Demo***)

Current APP version: 1

Waiting for USB MSD device plug-in...
```

12. The STK is now ready for firmware upgrade on 5.5.2 Create GBL File.

5.5.2 Create GBL File

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

1. Connect EFM32GG11 STK to your computer and start Simplicity Studio.
2. Click the **EFM32GG11 Giant Gecko Starter Kit (SLSTK3701A)** from the [**Debug Adapters**] tab. This will verify that the installation was successful, identify the device on the kit hardware, and automatically configure the software tools for use with your device.
3. From the [**Launcher**] perspective, click [**EFM32GG11 Giant Gecko Starter Kit**] under [**Software Examples**] of [**Getting Started**] tab to browse target sample application.
4. Click the **SLSTK3701A_usbhloader** sample application to create the example project in Simplicity IDE.


Getting Started
Documentation
Compatible Tools
Resources
Demos
Software Examples
SDK Documentation
protocol stack and implements an USB Vendor Unique
SLSTK3701A_usbhenum
This example project uses the USB host stack to
implement a simple device e
SLSTK3701A_usbhhidkbd
This example project uses the
implement support for USB H
SLSTK3701A_usbhloader
This example project uses the
modules in the drivers directo
SLSTK3701A_usbhmsdfatco
This example project uses the
modules in the drivers directo
SLSTK3701A_vcom
This example project uses the
to demonstrates the use of U
Gecko Bootloader 1.8.2
Gecko Bootloader Examp

USB Host Mass Storage Device GBL (Gecko Bootloader) loader example.

This example project uses the USB host and the MSD modules in the drivers directory to implement support for Mass Storage Device's (MSD). Info about the MSD device will be output on the VCOM port. When a MSD device is connected and detected as a valid, this application will start looking for a valid gbl file. Once a valid gbl file is found, the flash memory allocated for the bootloader storage is reprogrammed with the gbl file. Right after the bootloader storage is reprogrammed, the bootloader upgrade procedure starts and a software reset is performed.

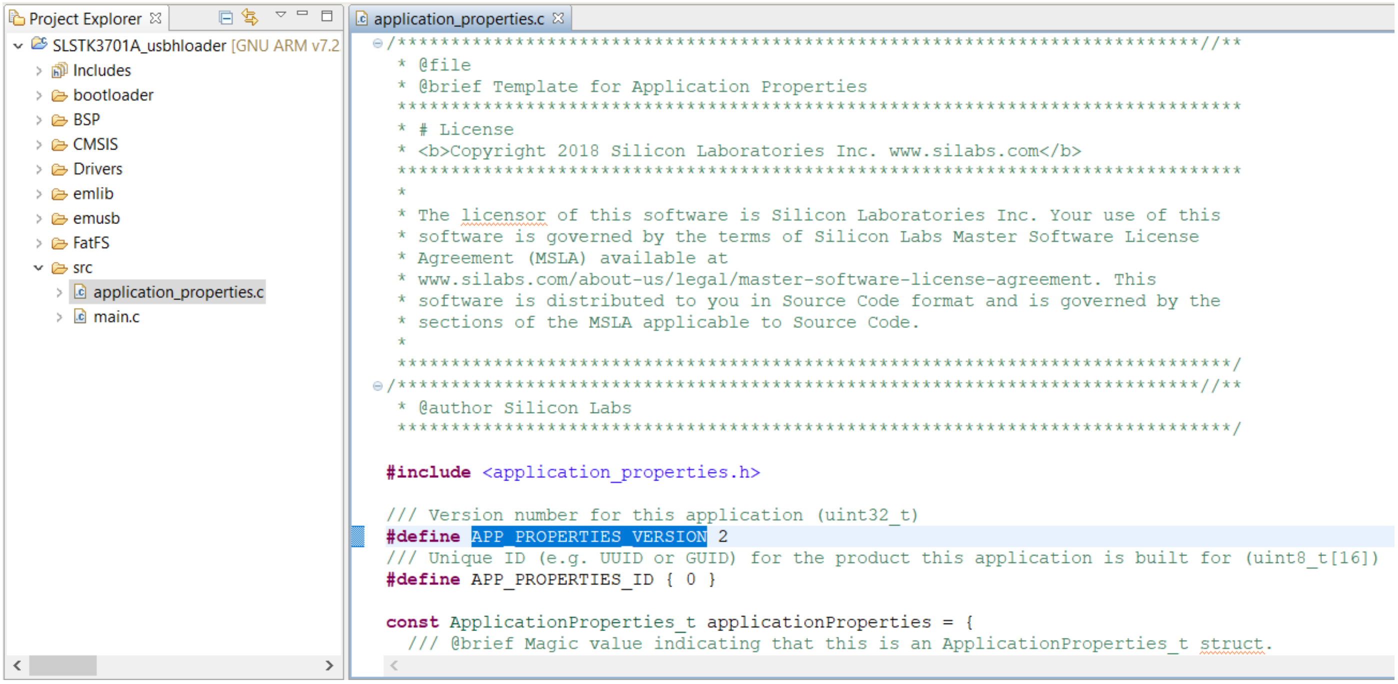
A combined first stage + main bootloader image called bootloader-storage-internal-single-combined.s37 is included in the example, which can be flashed to the STK using Simplicity Commander.

GBL files can be created using Simplicity Commander by simply running the command:
"commander gbl create myupgrade.gbl --app myapp.s37 --bootloader mybootloader.s37".
This is described in more details in UG266.

NOTE:
- USB disk drives may fail, as some of them draw too much current when attached.

Board: Silicon Labs SLSTK3701A Starter Kit
Device: EFM32GG11B820F2048GL192

5. Open `application_properties.c` file in **Project Explorer** and increase `APP_PROPERTIES_VERSION` by one (e.g., from 1 to 2).



```
/***************************************************************************//**
 * @file
 * @brief Template for Application Properties
 *******************************************************************************
 * # License
 * <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
 *******************************************************************************
 *
 * The licensor of this software is Silicon Laboratories Inc. Your use of this
 * software is governed by the terms of Silicon Labs Master Software License
 * Agreement (MSLA) available at
 * www.silabs.com/about-us/legal/master-software-license-agreement. This
 * software is distributed to you in Source Code format and is governed by the
 * sections of the MSLA applicable to Source Code.
 *
 ******************************************************************************/
/***************************************************************************//**
 * @author Silicon Labs
 ******************************************************************************/

#include <application_properties.h>

/// Version number for this application (uint32_t)
#define APP_PROPERTIES_VERSION 2
/// Unique ID (e.g. UUID or GUID) for the product this application is built for (uint8_t[16])
#define APP_PROPERTIES_ID { 0 }

const ApplicationProperties_t applicationProperties = {
  /// @brief Magic value indicating that this is an ApplicationProperties_t struct.
```

6. Click the [**Build**] icon ( ). Copy the `SLSTK3701A_usbhloader.s37` file from the build directory to Simplicity Commander folder (default location on Windows is `C:\SiliconLabs\SimplicityStudio\v4\developer\adapter_packs\commander`).
7. Go to step 9 if Gecko Bootloader security features (refer to step 8 in 3.3 Programming the Gecko Bootloader) are enabled.
8. Execute the following command in Simplicity Commander to create an upgrade GBL file (`SLSTK3701A_usbhloader.gbl`). Go to step 11 if the signing and encryption are not needed.

```
commander gbl create SLSTK3701A_usbhloader.gbl --app SLSTK3701A_usbhloader.s37
```

9. Execute the following command in Simplicity Commander to sign the application image (`SLSTK3701A_usbhloader.s37`) to enable secure boot of the application image (`SLSTK3701A_usbhloader_signed.s37`). The application image is signed using ECDSA-P256 and the signature is verified on every boot.

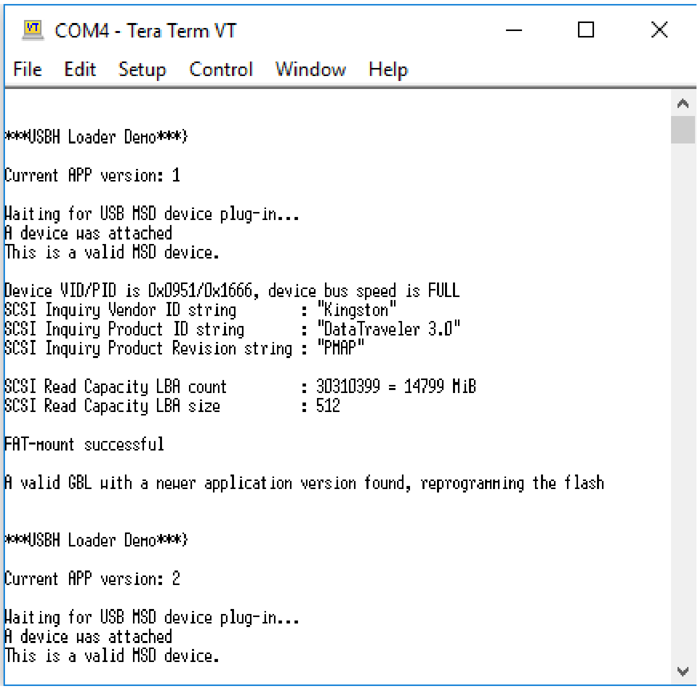
```
commander   convert   SLSTK3701A_usbhloader.s37   --secureboot   --keyfile   usbhost-signing-key   --outfile
SLSTK3701A_usbhloader_signed.s37
```

10. Execute the following command in Simplicity Commander to create a signed and encrypted upgrade GBL file (`SLSTK3701A_usbhloader_secure.gbl`). The firmware upgrade GBL file is ECDSA-P256 signed and AES-CTR-128 encrypted.

```
commander  gbl  create  SLSTK3701A_usbhloader_secure.gbl  --app  SLSTK3701A_usbhloader_signed.s37  -–sign
usbhost-signing-key --encrypt usbhost-encryption-key
```

Note: The signing (`usbhost-signing-key`) and encryption (`usbhost-encryption-key`) keys are generated in 5.5.1 Flash Image File step 5 and 6.

11. Open a terminal program (e.g. Tera Term) and access the STK Virtual COM port.
12. Press the RESET button on the Starter Kit. The version number of the running application will display on the terminal program.
13. Copy the normal `SLSTK3701A_usbhloader.gbl` file on step 8 (if GBL security features are disabled) or secure `SLSTK3701A_usbhloader_secure.gbl` file on step 10 (if GBL security features are enabled) to a USB stick and connect the USB stick to the STK.
14. Use the GBL file to upgrade the application. The version of the application is updated (e.g., from 1 to 2) after the upgrade.



```
***USBH Loader Demo***}

Current APP version: 1

Waiting for USB MSD device plug-in...
A device was attached
This is a valid MSD device.

Device VID/PID is 0x0951/0x1666, device bus speed is FULL
SCSI Inquiry Vendor ID string        : "Kingston"
SCSI Inquiry Product ID string       : "DataTraveler 3.0"
SCSI Inquiry Product Revision string : "PMAP"

SCSI Read Capacity LBA count         : 30310399 = 14799 MiB
SCSI Read Capacity LBA size          : 512

FAT-mount successful

A valid GBL with a newer application version found, reprogramming the flash


***USBH Loader Demo***}

Current APP version: 2

Waiting for USB MSD device plug-in...
A device was attached
This is a valid MSD device.
```

6. Pre-Programmed Device

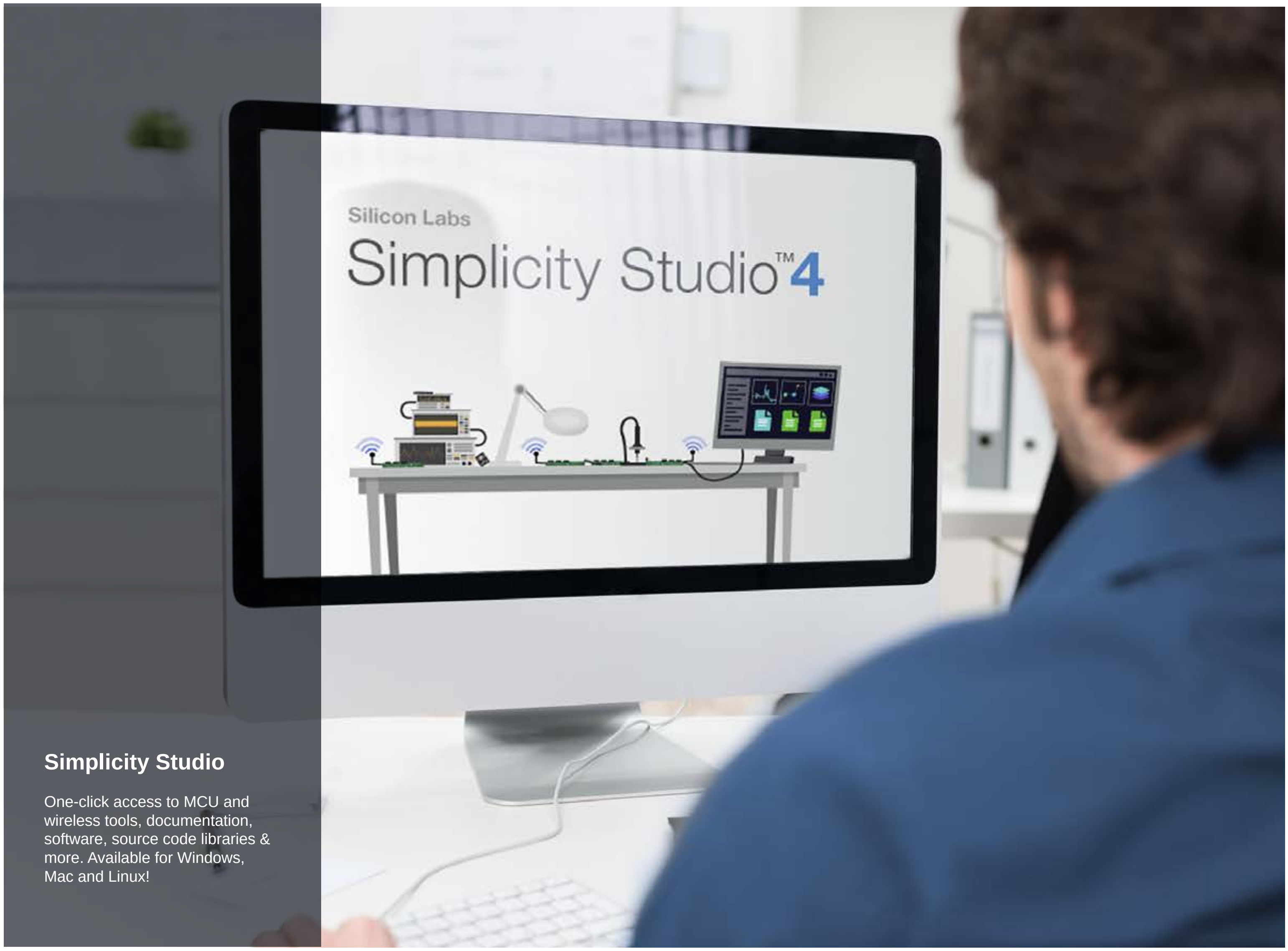
Silicon Labs offers pre-programmed devices for custom Gecko Bootloader and application. To do this, the binary or hex file must be provided. This option is subject to minimum order quantities (MOQ) and an additional cost. For this option, contact your local sales representative (http://www.silabs.com/buysample/pages/contact-sales.aspx?view=map).

7. Revision History

Revision 0.1

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*

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Trademark Information

Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, ISOmodem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® , Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

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

http://www.silabs.com