UG103.6: Bootloader Fundamentals

This document introduces bootloading for Silicon Labs networking devices. It describes the concepts of standalone and application bootloaders and discusses their relative strengths and weaknesses. In addition, it looks at design and implementation details for each method. Finally, it describes the bootloader file format.

Silicon Labs' *Fundamentals* series covers topics that project managers, application designers, and developers should understand before beginning to work on an embedded networking solution using Silicon Labs chips, networking stacks such as EmberZNet PRO or Silicon Labs *Bluetooth*®, and associated development tools. The documents can be used as a starting place for anyone needing an introduction to developing wireless networking applications, or who is new to the Silicon Labs development environment.

KEY POINTS

- Introduces the Gecko Bootloader.
- Summarizes the key features the bootloaders support and the design decisions associated with selecting a bootloader.
- Describes bootloader file formats.

1. Introduction

The 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. Production-level programming is typically done during the product manufacturing process yet it is desirable to be able to reprogram the system after production is complete. More importantly, it is valuable to be able to update the device's firmware with new features and bug fixes after deployment. The firmware image update capability makes that possible.

Silicon Labs supports devices that do not use a bootloader, but this requires external hardware such as a Debug Adapter (Silicon Labs ISA3 or Wireless Starter Kit (WSTK)) or third-party SerialWire/JTAG programming device to update the firmware. Devices without a bootloader have no supported way of updating the firmware once they are deployed, which is why Silicon Labs strongly advocates implementing a bootloader.

In March of 2017, Silicon Labs introduced the Gecko Bootloader, a code library configurable through Simplicity Studio's IDE to generate bootloaders that can be used with a variety of Silicon Labs protocol stacks. The Gecko Bootloader can be used with EFM32 and EFR32 Series 1 and later devices. The Gecko Bootloader was restructured into a component-based design and released with Gecko SDK Suite (GSDK) 4.0 in December of 2021. This new version is documented in *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*, along with other documents. Documentation for older versions is installed with their respective SDKs.

The Gecko Bootloader uses a customized update image file format. The update image file consumed by a Gecko Bootloader-generated application bootloader is a GBL (Gecko BootLoader) file, and is described in *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*.

Bootloading a firmware update image can be accomplished in two ways. The first is Over-The-Air (OTA), that is, through the wireless network, as shown in the following figure.

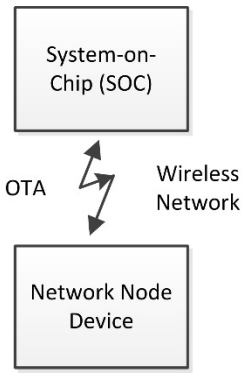

Figure 1.1. OTA Bootloading Use Case

The second is through a hardwired link to the device. The following figure represents the serial bootloader use cases for SoCs (System on Chips) using either a UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Protocol Interface), or USB (Universal Serial Bus) interface, and for NCPs (Network Coprocessors) using either UART or SPI.

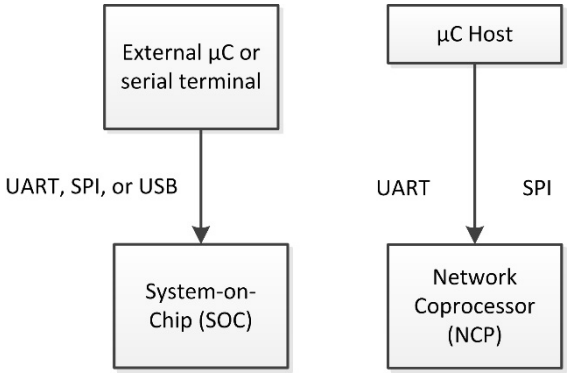

Figure 1.2. Serial Bootloading Use Cases

Silicon Labs networking devices use bootloaders that perform firmware updates in two different modes: standalone (also called standalone bootloaders) and application (also called application bootloaders). Application bootloaders are further divided into those that use external storage for the download update image, and those that use local storage. These bootloader types are discussed in the next two sections.

The firmware update situations described in this document assume that the source node (the device sending the firmware image to the target through a serial or OTA link) acquires the new firmware through some other means. For example, if a device on the local Zigbee network has an Ethernet gateway attached, this device could get or receive these firmware updates over the Internet. This necessary part of the firmware update process is system-dependent and beyond the scope of this document.

1.1 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. Very little interaction occurs between the standalone bootloader and the application running in flash. In general, the only time that the application interacts with the bootloader is when it requests a reboot into the bootloader. Once the bootloader is running, it receives firmware update packets containing the (new) firmware image either by physical connections such as UART or SPI, or by the radio (over-the-air).

When the firmware update process is initiated, the new code overwrites the existing stack and application code. If any errors occur during this process, the application cannot be recovered, and the process must start over. For information about configuring the Gecko Bootloader as a standalone bootloader, see *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*.

1.2 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 (if the chip has sufficient memory to support this local storage model).

The application bootloader relies on the application to acquire the new firmware image. This image can be downloaded by the application in any way that is convenient (UART, over-the-air, etc.) but it must be stored into a region referred to as the download space. The download space is typically an external memory device such as an EEPROM or dataflash, but it can also be a section of the chip's internal flash when using a local storage variant of the application bootloader. Once the new image has been stored, the application bootloader is then called to validate the new image and copy it from the download space to flash.

Since the application bootloader does not participate in acquiring the image, and the entire image is downloaded before the firmware update process is started, download errors do not adversely affect the running image. The download process can be restarted or paused to acquire the image over time. The integrity of the downloaded update image can be verified before initiating the firmware update process, to prevent a corrupt or non-functional image from being applied.

The Gecko Bootloader can be configured to accept a list of multiple upgrade images to attempt to verify and apply. This allows the Gecko Bootloader to store what is in effect a backup copy of the update image, which it can access if the first image is corrupt.

Note that the EmberZNet NCP platform does not utilize an application bootloader because the application code resides on the host rather than on the NCP directly. Instead a device acting as a serial coprocessor would utilize a standalone bootloader designed to accept code over the same serial interface as the expected NCP firmware uses. However, the host application (residing on a separate MCU from the NCP) can utilize whatever bootloading scheme is appropriate.

For more information on application bootloaders, see *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*.

2. About the Gecko Bootloader

The Silicon Labs Gecko Bootloader is a configurable code library that can be used with all the newer Silicon Labs Gecko MCUs and wireless MCUs. It uses a specially-formatted update image file called a GBL file. The Gecko Bootloader has a two-stage design on Series 1 devices, where a minimal first stage bootloader is used to update the main bootloader. On Series 2 devices, the first stage bootloader is replaced by a Secure Engine and the Gecko Bootloader consists only of the main bootloader. The Secure Engine may be hardware-based, or virtual (software). If hardware-based, the implementation may be either with or without Secure Vault. Throughout this document, the following conventions will be used.

- HSE - Hardware Secure Engine, either with or without Secure Vault if not specified
- VSE - Virtual Secure Engine
- SE - Secure Engine (either HSE or VSE, in general)

Having a first stage bootloader or SE allows for field updates of the main bootloader, including adding new capabilities, changing communication protocols, adding new security features and fixes, and so on. The Gecko Bootloader consists of three component parts:

Core: The bootloader core contains the main function of both bootloader stages. It also contains functionality to write to the internal main flash, to perform a bootloader update, and to reset into the application flagging applicable reset reasons.

Driver: Different bootloading applications require different hardware drivers for use by the other components of the bootloader.

Component/Plugin: All parts of the main bootloader that are either optional or selectable for different configurations are implemented as components (in GSDK 4.0 and higher) or previously in plugins. Each component/plugin has a generic header file, and one or more implementations. The current release contains components for functionality like UART and SPI communication protocols, SPI flash storage, internal flash storage, and different cryptographic operations.

2.1 Features

Gecko Bootloader features include:

- Field-updateable
- Secure boot
- Signed GBL firmware update image file
- Encrypted GBL firmware update image file

These features are summarized in the following sections and described in more detail in *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*. Protocol-specific information about using the Gecko Bootloader may be found in the following documents:

- *AN1084: Using the Gecko Bootloader with EmberZNet*
- *UG235.06: Bootloading and OTA with Silicon Labs Connect SDK v2.x*
- *UG435.06: Bootloading and OTA with Silicon Labs Connect SDK v3.x*
- *AN1086: Using the Gecko Bootloader with Silicon Labs Bluetooth Applications*

2.1.1 Field-Updateable

Series 1

On EFM32 and EFR32 Series 1 devices, field update capability for the Gecko bootloader is provided by a two-stage design where the bootloader has a first stage and a main stage. The minimal first stage of the bootloader is not field updateable, and can only update the main bootloader by reading from and writing to fixed addresses in internal flash memory. To perform a main bootloader update, the running main bootloader verifies the integrity and authenticity of the bootloader update image, writes it to internal flash, and issues a reboot into the first stage bootloader. The first stage bootloader verifies the integrity of the main bootloader update image before copying it to the main bootloader location, completing the update

Series 2

On Series 2 devices, field update capability for the Gecko bootloader is provided by the SE. To perform a main bootloader update, the running main bootloader verifies the integrity and authenticity of the bootloader update image, writes it to internal flash, and requests that the SE installs the update. The SE optionally verifies the authenticity of the main bootloader update image before copying it to the main bootloader location, completing the update. The same mechanism can be used to update the SE itself.

2.1.2 Secure Boot

Secure boot is designed to prevent an untrusted application from running on the device. When Secure Boot is enabled, the bootloader enforces cryptographic signature verification of the application image on every boot using asymmetric cryptography. The signature algorithm used is ECDSA-P256-SHA256. The public key is written to the device during manufacturing, while the private key is kept secret. This ensures that the application was created and signed by a trusted party.

2.1.3 Signed GBL Update Image File

The Gecko Bootloader supports enforcing cryptographic signature verification of the update image file in addition to Secure Boot. This allows the bootloader and application to verify that the application or bootloader update comes from a trusted source before starting the update process. The signature algorithm used is ECDSA-P256-SHA256. The public key is the same key as for secure boot, written to the device during manufacturing, while the private key is never distributed. This ensures that the GBL file was created and signed by a trusted party.

2.1.4 Encrypted GBL Update File

The GBL update file can also be encrypted, to prevent eavesdroppers from getting hold of the plaintext firmware image. The encryption algorithm used is AES-CTR-128, and the encryption key is written to the device during manufacturing.

3. Memory Space for Bootloading

The first stage of the Gecko Bootloader on Series 1 devices takes up a single flash page. On devices with 2 kB flash pages, like EFR32MG1, this means that the first stage takes 2 kB.

The size of the main bootloader is dependent on the functionality required. With a typical bootloader configuration, the main bootloader for Series 1 devices takes up 14 kB of flash, bringing the total bootloader size to 16 kB.

Silicon Labs recommends reserving 16 kB for the bootloader for Series 1 and EFR32xG21 devices and 24 kB for EFR32xG22 devices.

On EFR32xG1 devices (Mighty Gecko, Flex Gecko, and Blue Gecko families), the bootloader resides in main flash.

- First stage bootloader @ 0x0
- Main bootloader @ 0x800
- Application @ 0x4000

On EFR32xG12 and later Series 1 devices, the bootloader resides in the bootloader area in the Information Block.

- Application @ 0x0
- First stage bootloader @ 0x0FE10000
- Main bootloader @ 0x0FE10800

On EFR32xG21, the main bootloader resides in main flash:

- Main bootloader @ 0x0
- Application @ 0x4000

On EFR32xG22, the main bootloader resides in main flash:

- Main bootloader @ 0x0
- Application @ 0x6000

On EFR32xG23, the main bootloader resides in main flash:

- Main bootloader @ 0x08000000
- Application @ 0x08006000

On EFR32xG24, the main bootloader resides in main flash:

- Main bootloader @ 0x08000000
- Application @ 0x08006000

4. Design Decisions

The decision of what bootloader type to deploy depends on many factors. Note that the platform type and available flash memory may limit bootloader choices.

Some questions related to this are:

- Where does the device get the new update image? Is this over-the-air via the networking protocol? Using a separate interface connected to the Internet?
- Will the device have an external memory chip to store a new update image? If not, is there enough internal flash memory to store both a current and a newly downloaded copy of the largest expected application image?
- If the device receives the new image over-the-air, will it be multiple hops away from the server holding the download image?
- What kind of image security is needed?
- What communications driver will be used (in the single protocol case)?
- Does the use case require more than one protocol?

The configurable design of the Gecko Bootloader platform means that developers can create bootloaders to fit almost any design choice. See *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher* for more details.

Smart. Connected. Energy-Friendly.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com