AN1326: Transitioning to the Updated Gecko Bootloader in GSDK 4.0 and Higher

Gecko Bootloader v2.x, introduced in GSDK 4.0, contains a number of changes compared to Gecko Bootloader v1.x. Many of these changes are due to an underlying framework redesign that results in an improved developer experience in Simplicity Studio 5 as well as enhanced compatibility within the GSDK. This document describes the differences between the versions, including how to configure the new Gecko Bootloader within Simplicity Studio 5.

KEY POINTS

- Gecko Bootloader project structure changes
- Gecko Bootloader configuration changes
- Gecko Bootloader component configurator
- Configuring storage slots in the Gecko Bootloader v2.x
- Adding postbuild steps for Gecko Bootloader v2.x
- New linker file generation system

1 Introduction

Gecko Software Development Kit suite (GSDK) version 3 (GSDK v3.0) introduced a new underlying Gecko Platform architecture based on components. Beginning with GSDK 4.0, the Gecko Bootloader now uses this underlying architecture. With Simplicity Studio 5 (SSv5) and GSDK 4.0, developers working with Gecko Bootloader will benefit from the following component-based project configuration features:

- Search and filter to find and discover software components that work with the target device
- Automatically pull in all component dependencies and initialization code
- Configurable software components including peripheral inits, drivers, middleware, and stacks
- All configuration settings are in C header files for usage outside of Simplicity Studio
- Configuration validation to alert developers to errors or issues
- Easily manage all project source via git or other SCM tools
- Managed migration to future component and SDK versions
- Simplified transitions from Silicon Labs development kits to custom hardware

Other features of the SSv5/GSDK 4.x development environment include:

- Project source management options (link to SDK sources or copy all contents to user folder)
- Graphical pin configuration through the Pin Tool
- Redesigned Radio Configurator with a fresh UI that’s more intuitive for single- and multi-PHY customization
- Iterative development (configure components, edit sources, compile, debug) using SSv5 configuration tools and third-party IDEs
- GNU makefiles as a build option

This document summarizes the differences between the Gecko Bootloader v2.x in GSDK 4.0 and earlier AppBuilder-based versions. These differences include:

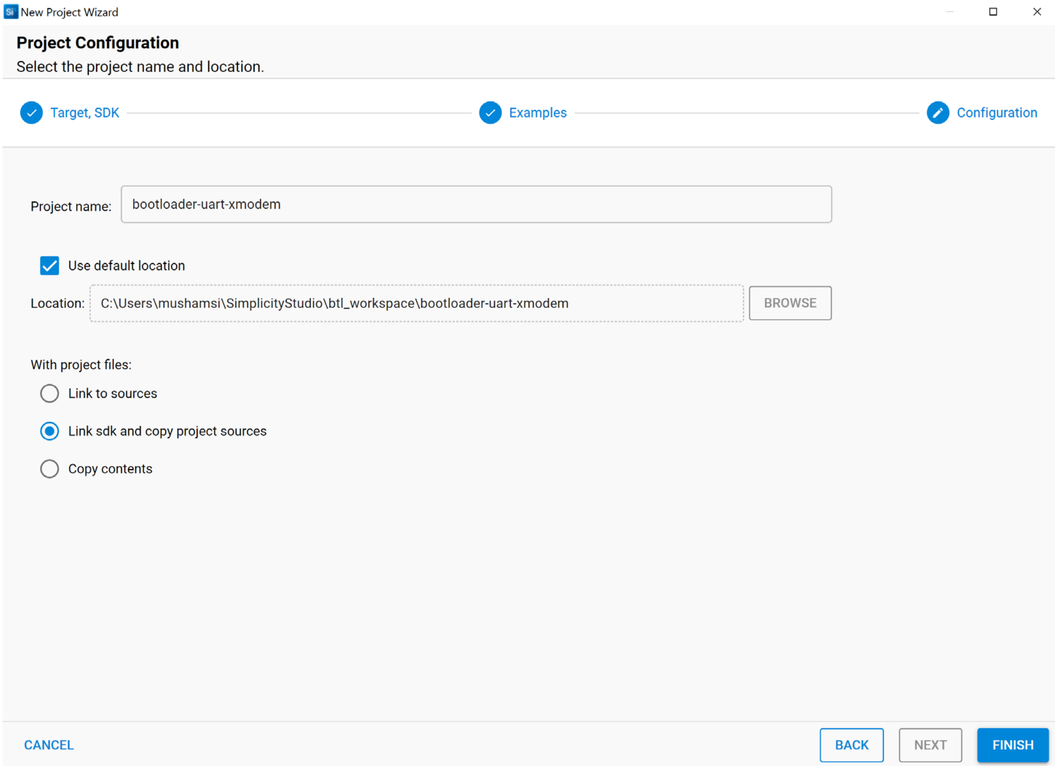
- Differences between AppBuilder and the new component-based Project Configurator
- Comparison of the new Project Configurator components with AppBuilder plugins
- Features of the Pin Tool, now used instead of Hardware Configurator
- Other differences including:
  - Linker file
  - Additional Macros
  - Postbuild steps
  - Main bootloader in main flash
  - Callbacks
  - Storage Slots
  - App Properties

2 About Projects

2.1 Creating a Project

If you have used earlier versions of SSv5, initial project creation is unchanged. If you are unfamiliar with project creation in SSv5, see the Simplicity Studio 5 User's Guide. This chapter highlights the differences with AppBuilder project creation and configuration.

Once you have selected an example project, an initial configuration dialog opens. Note that the Project Location, which in AppBuilder is shown on the General tab, is defined here.



Project Configuration

Select the project name and location.

Project name: bootloader-uart-xmodem

☑ Use default location

Location: C:\Users\mushamsi\SimplicityStudio\btl_workspace\bootloader-uart-xmodem

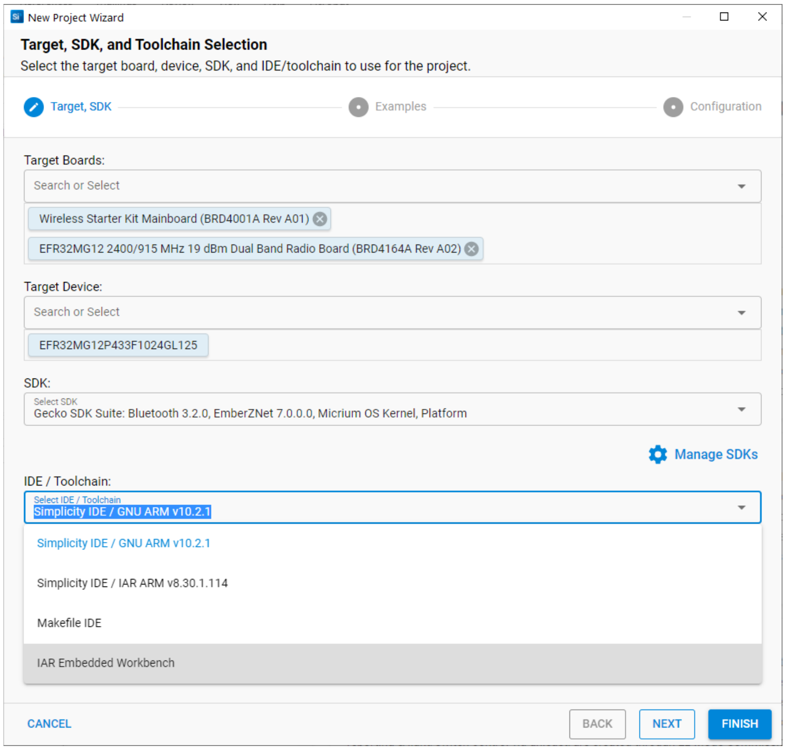
With project files:

○ Link to sources

● Link sdk and copy project sources

○ Copy contents

In SSv5, the compiler to use is specified when you create a project and, after that, is difficult to change (see Simplicity Studio 5 User's Guide Tips & Tricks for a procedure to convert an existing project from GCC to IAR). If you do not want to use the default compiler (GCC unless otherwise specified), click **BACK**, and change it on the first Project Configuration dialog.



New Project Wizard

Target, SDK, and Toolchain Selection

Select the target board, device, SDK, and IDE/toolchain to use for the project.

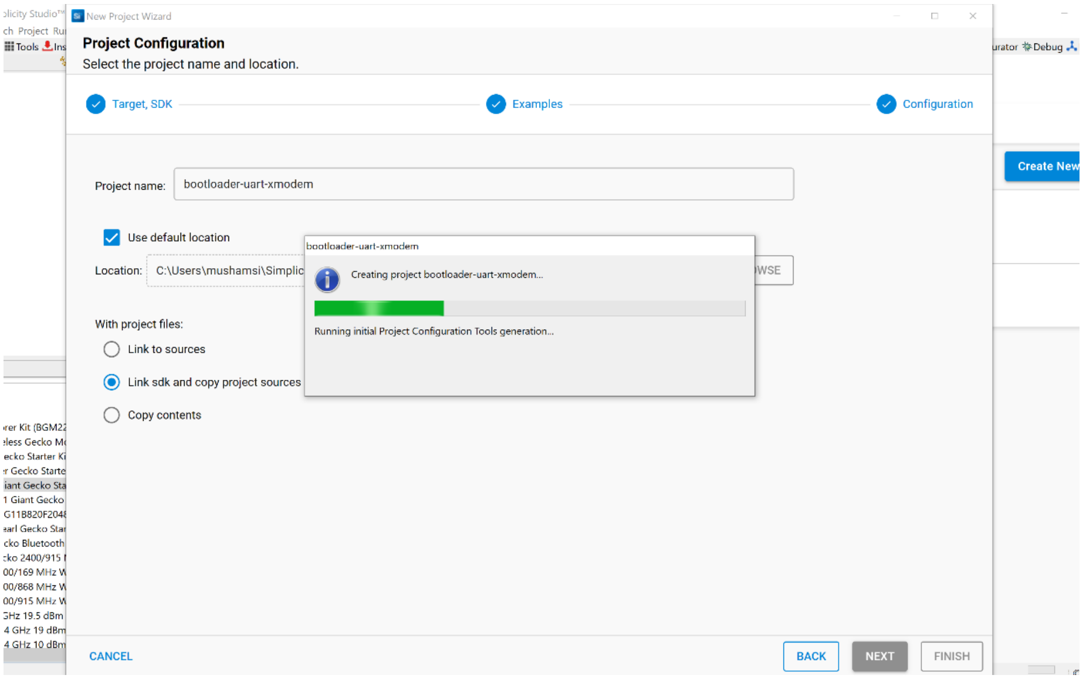
Target Boards: Search or Select
- Wireless Starter Kit Mainboard (BRD4001A Rev A01)
- EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board (BRD4164A Rev A02)

Target Device: Search or Select
- EFR32MG12P433F1024GL125

SDK: Select SDK
Gecko SDK Suite: Bluetooth 3.2.0, EmberZNet 7.0.0.0, Micrium OS Kernel, Platform

IDE / Toolchain: Select IDE / Toolchain
Simplicity IDE / GNU ARM v10.2.1
- Simplicity IDE / GNU ARM v10.2.1
- Simplicity IDE / IAR ARM v8.30.1.114
- Makefile IDE
- IAR Embedded Workbench

Click **FINISH** and the project will be generated.



New Project Wizard

Project Configuration

Select the project name and location.

Project name: bootloader-uart-xmodem

☑ Use default location

Location: C:\Users\mushamsi\Simplic

With project files:
- ○ Link to sources
- ● Link sdk and copy project sources
- ○ Copy contents

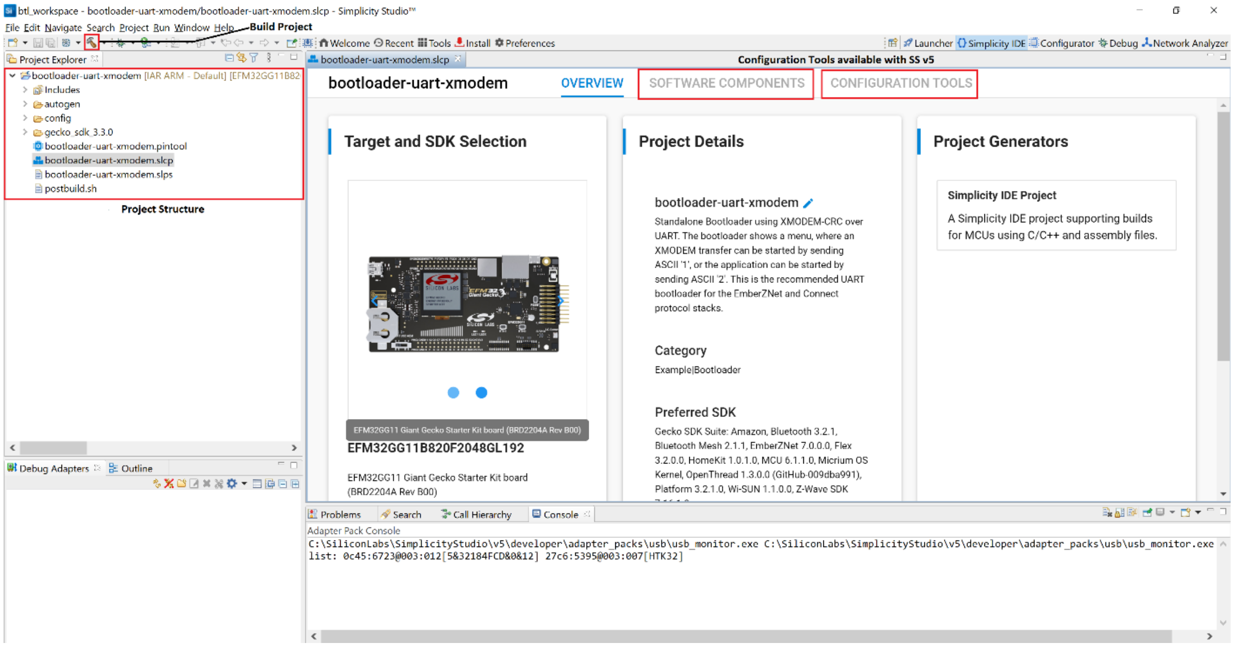
bootloader-uart-xmodem

Creating project bootloader-uart-xmodem...

Running initial Project Configuration Tools generation...

2.2 Project File Structure

On project generation, the required project files and folders are created.


btl_workspace - bootloader-uart-xmodem/bootloader-uart-xmodem.slcp - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Build Project
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Configurator Debug Network Analyzer
Project Explorer
bootloader-uart-xmodem.slcp
Configuration Tools available with SS v5
OVERVIEW
SOFTWARE COMPONENTS
CONFIGURATION TOOLS
Project Structure
Debug Adapters
Outline
Problems
Search
Call Hierarchy
Console
Adapter Pack Console

bootloader-uart-xmodem [IAR ARM - Default] [EFM32GG11B82
- Includes
- autogen
- config
- gecko_sdk_3.3.0
- bootloader-uart-xmodem.pintool
- bootloader-uart-xmodem.slcp
- bootloader-uart-xmodem.slps
- postbuild.sh

bootloader-uart-xmodem

Target and SDK Selection

EFM32GG11 Giant Gecko Starter Kit board (BRD2204A Rev B00)

EFM32GG11B820F2048GL192

EFM32GG11 Giant Gecko Starter Kit board (BRD2204A Rev B00)

Project Details

bootloader-uart-xmodem

Standalone Bootloader using XMODEM-CRC over UART. The bootloader shows a menu, where an XMODEM transfer can be started by sending ASCII '1', or the application can be started by sending ASCII '2'. This is the recommended UART bootloader for the EmberZNet and Connect protocol stacks.

Category

Example|Bootloader

Preferred SDK

Gecko SDK Suite: Amazon, Bluetooth 3.2.1, Bluetooth Mesh 2.1.1, EmberZNet 7.0.0.0, Flex 3.2.0.0, HomeKit 1.0.1.0, MCU 6.1.1.0, Micrium OS Kernel, OpenThread 1.3.0.0 (GitHub-009dba991), Platform 3.2.1.0, Wi-SUN 1.1.0.0, Z-Wave SDK

Project Generators

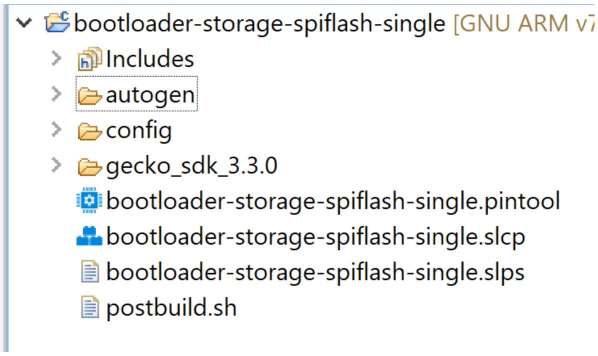
Simplicity IDE Project

A Simplicity IDE project supporting builds for MCUs using C/C++ and assembly files.

```
C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\usb\usb_monitor.exe C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\usb\usb_monitor.exe
list: 0c45:6723@003:012[5&32184FCD&0&12] 27c6:5395@003:007[HTK32]
```

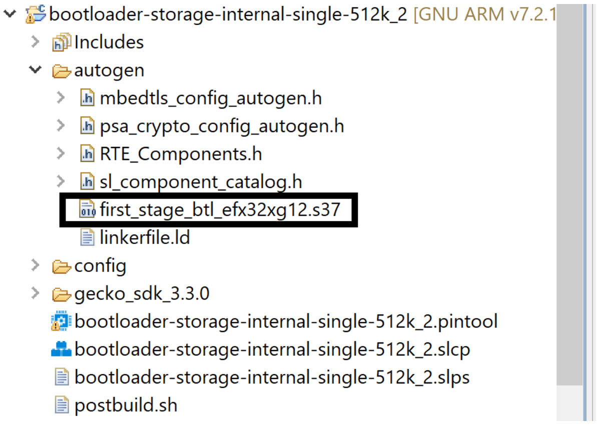
Projects always include the following parts:

- **autogen** folder: Only the autogen folder includes generated code. It includes configuration (mbedtls_config_autogen.h), init code, the linker script, and other generated code used by components, like the command descriptors for the CLI interface.
- **config** folder: Component configuration headers are located in this folder. These can be edited with the Simplicity IDE Component Editor, but directly editing the header file is also possible. The Component Editor is available through the Project Configurator's **Configure** control, available only for configurable components.
- **gecko_sdk** folder (with version number): Contains source and binary files added by components.
- **files in the root folder**: Only the application specific files should be in the root folder, including source files, the project configurator (*.slcp*) file and the Pin Tool *(.pintool*) file. For more details, see *UG489: Silicon Labs Gecko Bootloader User's Guide for GDSK 4.0 and Higher*.



bootloader-storage-spiflash-single [GNU ARM v7
- Includes
- autogen
- config
- gecko_sdk_3.3.0
- bootloader-storage-spiflash-single.pintool
- bootloader-storage-spiflash-single.slcp
- bootloader-storage-spiflash-single.slps
- postbuild.sh

The first stage bootloader binary file is copied to the **autogen** folder at the time of project generation. The correct first stage binary file is located in accordance with the target hardware and placed in the **autogen,** which is later used by the postbuild script to generate a combined bootloader binary. The **autogen** folder is shown below with the first stage bootloader binary.



```
bootloader-storage-internal-single-512k_2 [GNU ARM v7.2.1
  Includes
  autogen
    mbedtls_config_autogen.h
    psa_crypto_config_autogen.h
    RTE_Components.h
    sl_component_catalog.h
    first_stage_btl_efx32xg12.s37
    linkerfile.ld
  config
  gecko_sdk_3.3.0
  bootloader-storage-internal-single-512k_2.pintool
  bootloader-storage-internal-single-512k_2.slcp
  bootloader-storage-internal-single-512k_2.slps
  postbuild.sh
```

2.3 Configuring a Project

Most project configuration can be done through tools in Simplicity Studio. These tools provide the functionality previously provided by AppBuilder.

- **Project Configurator** – the top-level project configuration tool through which you install and uninstall components and change other project parameters. It also provides access to other configuration tools.
- **Component Editor** – provides access to the configurable parameters of an individual component.
- **Pin Tool** – configures peripherals.

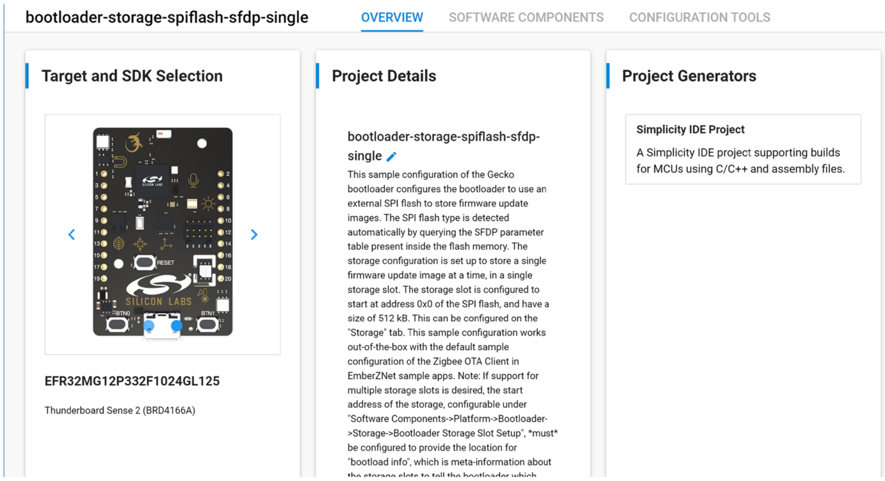
Once you create the project, the Project Configurator tabbed interface is presented, including:

- Overview
- Software Components
- Configuration Tools

You can always re-open the Project Configurator by double-clicking the <projectname>.slcp file. For details on using the Project Configurator and Component Editor, see the Simplicity Studio 5 User's Guide.

2.3.1 OVERVIEW Tab

The OVERVIEW tab, like AppBuilder's General tab, shows the target part information, and a description of the project.



bootloader-storage-spiflash-sfdp-single

EFR32MG12P332F1024GL125

Thunderboard Sense 2 (BRD4166A)

bootloader-storage-spiflash-sfdp-single

This sample configuration of the Gecko bootloader configures the bootloader to use an external SPI flash to store firmware update images. The SPI flash type is detected automatically by querying the SFDP parameter table present inside the flash memory. The storage configuration is set up to store a single firmware update image at a time, in a single storage slot. The storage slot is configured to start at address 0x0 of the SPI flash, and have a size of 512 kB. This can be configured on the "Storage" tab. This sample configuration works out-of-the-box with the default sample configuration of the Zigbee OTA Client in EmberZNet sample apps. Note: If support for multiple storage slots is desired, the start address of the storage, configurable under "Software Components->Platform->Bootloader->Storage->Bootloader Storage Slot Setup", *must* be configured to provide the location for "bootload info", which is meta-information about the storage slots to tell the bootloader which

Simplicity IDE Project

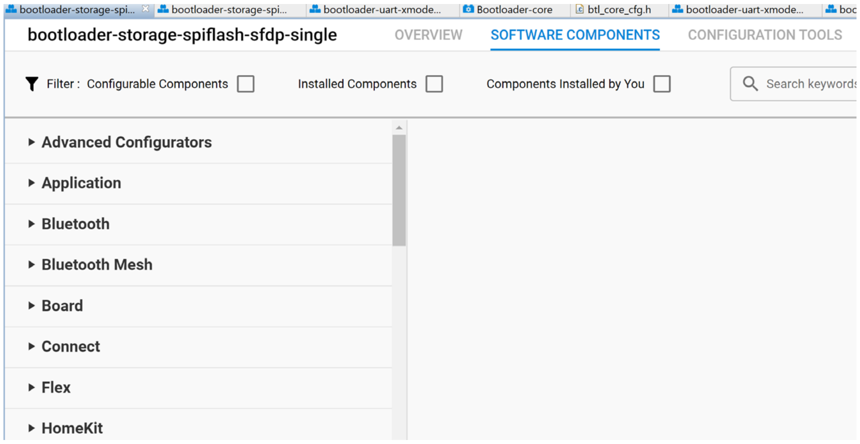
A Simplicity IDE project supporting builds for MCUs using C/C++ and assembly files.

Note that there is no **Generate** control. Changes made through Project Configurator are auto-saved and the project files are auto-generated. The Force Generation control on the Project Details card is provided for use only in the event of a system problem, where auto-generation fails.

The Project Generators interface controls generation of additional files for import into other IDEs. It does not change the compiler used to build the application image.

2.3.2 SOFTWARE COMPONENTS Tab

The SOFTWARE COMPONENTS tab shows the available components and those that are already installed in the example code.



bootloader-storage-spiflash-sfdp-single

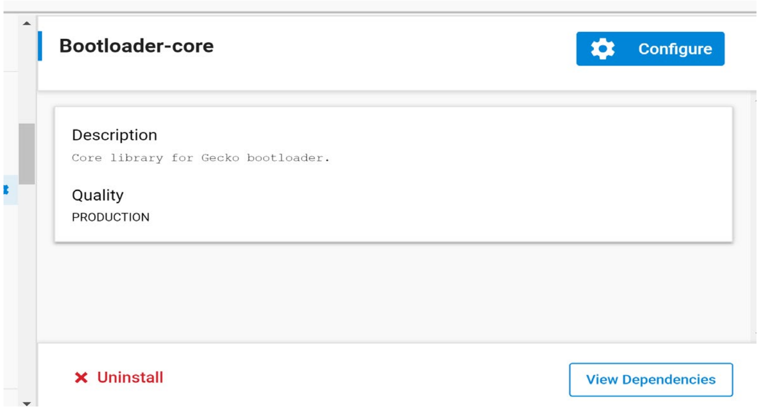
Filter : Configurable Components □ Installed Components □ Components Installed by You □

- Advanced Configurators
- Application
- Bluetooth
- Bluetooth Mesh
- Board
- Connect
- Flex
- HomeKit

Component groupings are subject to change in SDK releases, as new components are added and organization is updated. The most reliable way to find a specific component is to search for it. You can also filter the displayed components using the checkboxes at the top.

Installed components have a circled checkmark to the left. Configurable components have a gear icon to the right.

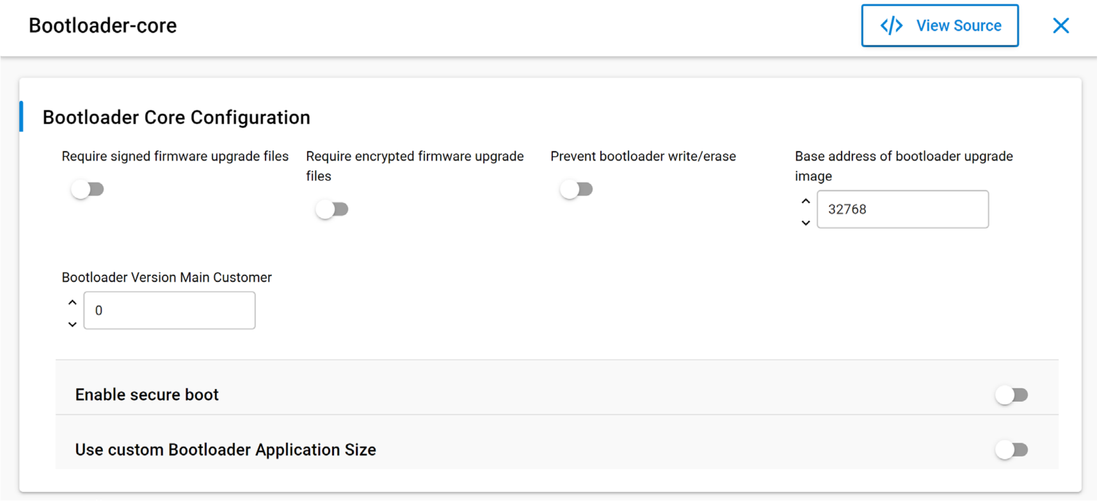
Select a component to see details about it. Click **Install** to install the component or **Uninstall** to remove it. If it is configurable, a **Configure** control is shown in the upper right.



Description
Core library for Gecko bootloader.

Quality
PRODUCTION

Click that control or the gear icon next to an installed component name to open the Component Editor in a new tab.

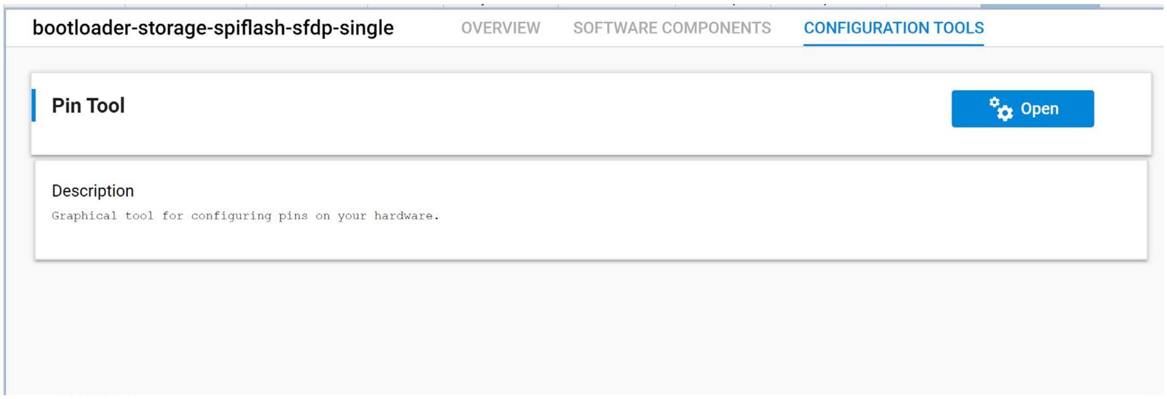


Require signed firmware upgrade files: off
Require encrypted firmware upgrade files: off
Prevent bootloader write/erase: off
Base address of bootloader upgrade image: 32768
Bootloader Version Main Customer: 0
Enable secure boot: off
Use custom Bootloader Application Size: off

Save any changes to trigger file generation.

2.3.3 CONFIGURATION TOOLS Tab

The CONFIGURATION TOOLS tab is a quick way of opening useful tools for project development.



Description
Graphical tool for configuring pins on your hardware.

3 Components

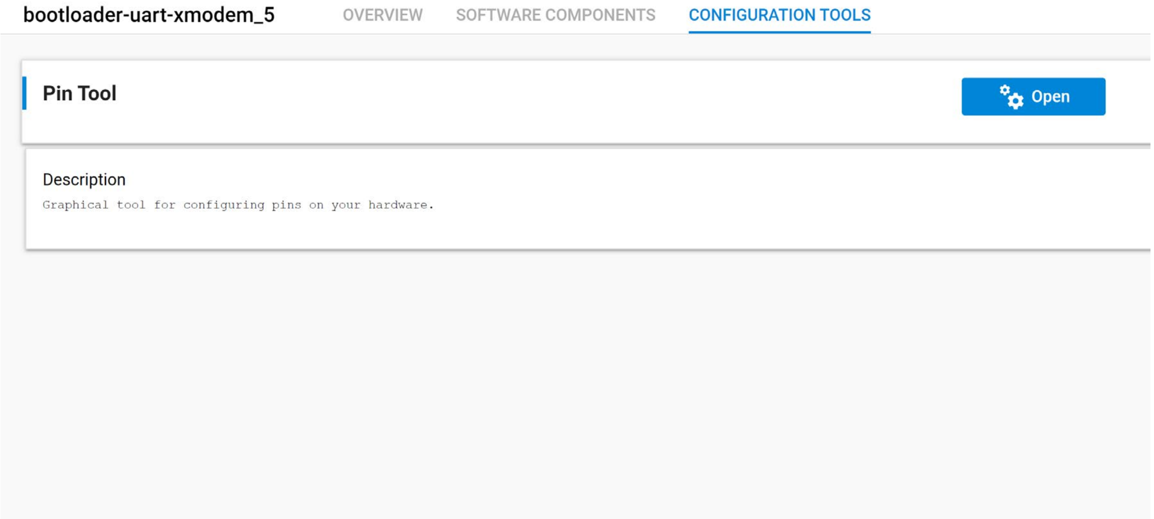
This section maps the various Gecko Bootloader components to the previously available plugins (in the AppBuilder workflow). Although the basic subgroups might remain the same, specific component locations are subject to change across various GSDK versions. Therefore, it is good practice to search for individual components by their names rather than the specific subgroups that they belong to.

AppBuilder Plugin	Simplicity Studio v5 Component	Additional Comments
Communication		
BGAPI UART DFU	BGAPI UART DFU	
EZSP-SPI	EZSP-SPI	
UART XMODEM	UART XMODEM	
XMODEM Parser	XMODEM Parser	
Core		
Application Upgrade Version Check	Application Upgrade Version Check	
Bootloader Core	Bootloader Core	
GBL Compression (LZ4)	GBL Compression (LZ4)	
GBL Compression (LZMA)	GBL Compression (LZMA)	
Image Parser	Image Parser	
Image Parser with legacy EBL support	Image Parser with legacy EBL support	
Image Parser without encryption support	Image Parser without encryption support	
“	Bootloader Include Parser	New component that includes common Image Parser header files
“	Bootloader in Main Flash	New component that enables the user to place bootloader in main flash for xG13 and xG14 devices
Drivers		
Delay	Bootloader Delay Driver	
SPI Master	Bootloader SPI Controller USART Driver	
SPI Slave	Bootloader SPI Peripheral USART Driver	
UART	Bootloader UART Driver	
EUART	Bootloader EUART Driver	
“	Bootloader SPI Controller EUSART Driver	New component supporting SPI EUSART driver. This component can be used with devices supporting EUSART interface.
“	Bootloader SPI Peripheral EUSART Driver	New component supporting SPI EUSART driver. This component can be used with devices supporting EUSART interface.
Storage		
Common Storage	Common Storage	
Common Storage (single storage slot only)	Common Storage (single storage slot only)	
Internal Storage	Internal Storage	
SPI Flash Storage	SPI Flash Storage	
“	Bootloader Storage Slot Setup	New component that facilitates configuring storage slots for Internal and SPI Flash based bootloaders
Utils		
Crypto	Crypto	
Cyclic Redundancy Check	Cyclic Redundancy Check	

Debug	Debug	
EMLIB	EMLIB Peripheral HAL	This component is no longer part of the Bootloader module. EMLIB component(s) can be found under Platform > Peripheral
EZSP GPIO Activation	EZSP GPIO Activation	
GPIO Activation	GPIO Activation	
SE Manager	SE Manager	This component is now available under Platform > Security. This component is only applicable for Series 2 devices.
Token Management	Token Management	
mbed TLS	Mbed TLS …	These components are available under Platform > Security
Bootloader Interface		
	App Properties	A new component to configure application properties (version), which is used during creation of a GBL file using Simplicity Commander.
Bootloader-interface	Bootloader Application Interface	

4 Pin Tool

In contrast to the AppBuilder workflow, Simplicity Studio v5 offers the **Pin Tool** configuration tool, which can be used to configure the various pins of the MCU used in the project. Pin Tool can be used in standalone mode through the tile on the **Configuration Tools** tab.



Description
Graphical tool for configuring pins on your hardware.

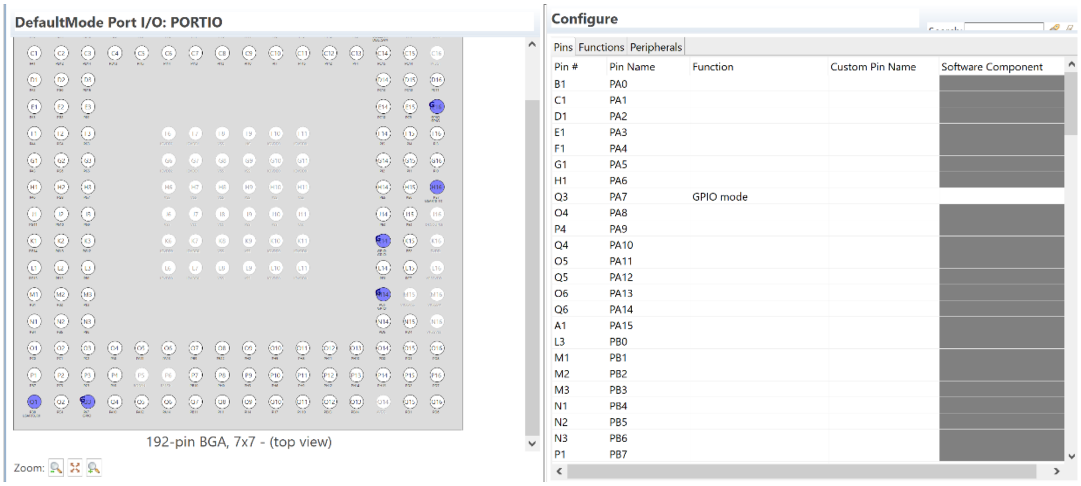
Pin Tool-related configuration is also available through the Component Editor of all applicable configurable components, making it a one-stop solution for configuring pin details and setting user configuration details. Following is the list of configurable components that also include Pin Tool configuration (wherever applicable) in their configuration wizard.

Configurable Component	Configuration Options
UART XMODEM	Menu Idle Timeout
Bootloader Core	All options that were available for the bootloader plugin are now available in the Bootloader Core component. Options include: Enable Secure Boot, Enable certificate support, Prevent bootloader write/erase, etc
Bootloader UART/EUART Driver	Configuration options include: USART settings, USART peripheral, Serial VCOM Enable/Disable
Bootloader SPI Controller USART/EUSART Driver	Configuration options include: Frequency, Peripheral and SPI Port Settings
Bootloader SPI Peripheral USART/EUSART Driver	Configuration options include: Tx Buffer Size, Rx Buffer Size, Peripheral and Port Settings
Bootloader Storage Slot Setup	Supports configuration of storage slot (start addresses and slot lengths) up to a maximum of 3 slots
Common Storage/Common Storage (single storage slot only)	Configuration options include: Start address of bootload info
Internal Storage	Configuration options include: Enable DMA based MSC Write, DMA channel to reserve
SPI Flash Storage	Enables configuration of the list of SPI flash devices that must be enabled in the bootloader image
Debug	Enable/disable Debug prints/asserts
EZSP GPIO Activation	Configuration options include: Properties of SPI NCP, WAKE INT Pin, HOST INT Pin
GPIO Activation	Configuration options include: Properties of Bootloader entry, Button Pin configuration

The configuration options of any installed configurable component can be accessed by clicking **Configure**. Refer to section 2 Project Creation for more details on the Component Editor.

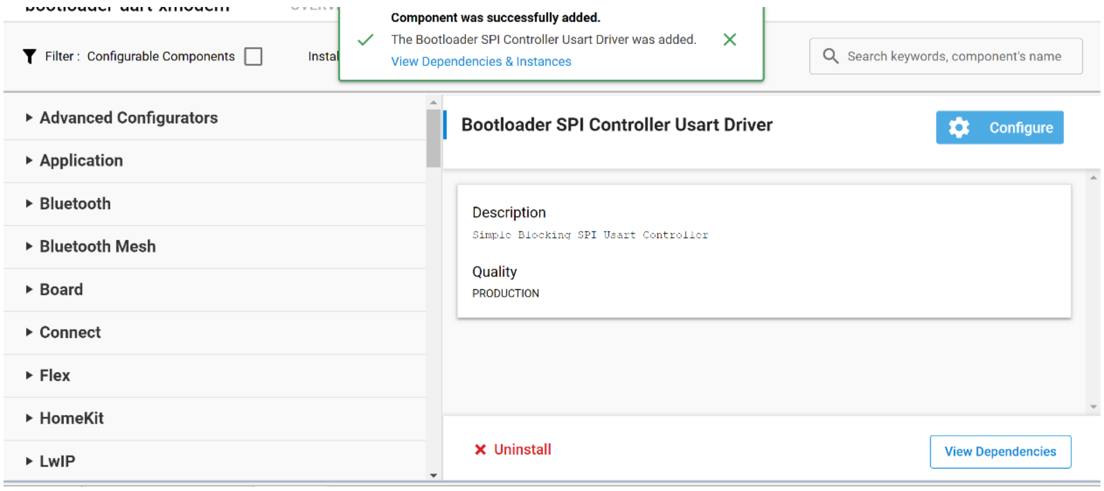
As an example, in the following procedure the Bootloader SPI Controller USART Driver is configured using the Component Editor and the pin configuration is verified using the Pin Tool.

1. Pin Tool configuration before installing the SPI Controller component. The device used for this example is the EFM32GG11 Giant Gecko Starter Kit board (BRD2204A Rev B00).



Pin #	Pin Name	Function	Custom Pin Name	Software Component
B1	PA0			
C1	PA1			
D1	PA2			
E1	PA3			
F1	PA4			
G1	PA5			
H1	PA6			
Q3	PA7	GPIO mode		
O4	PA8			
P4	PA9			
Q4	PA10			
O5	PA11			
Q5	PA12			
O6	PA13			
Q6	PA14			
A1	PA15			
L3	PB0			
M1	PB1			
M2	PB2			
M3	PB3			
N1	PB4			
N2	PB5			
N3	PB6			
P1	PB7			

2. Install the Bootloader SPI Controller USART driver component.



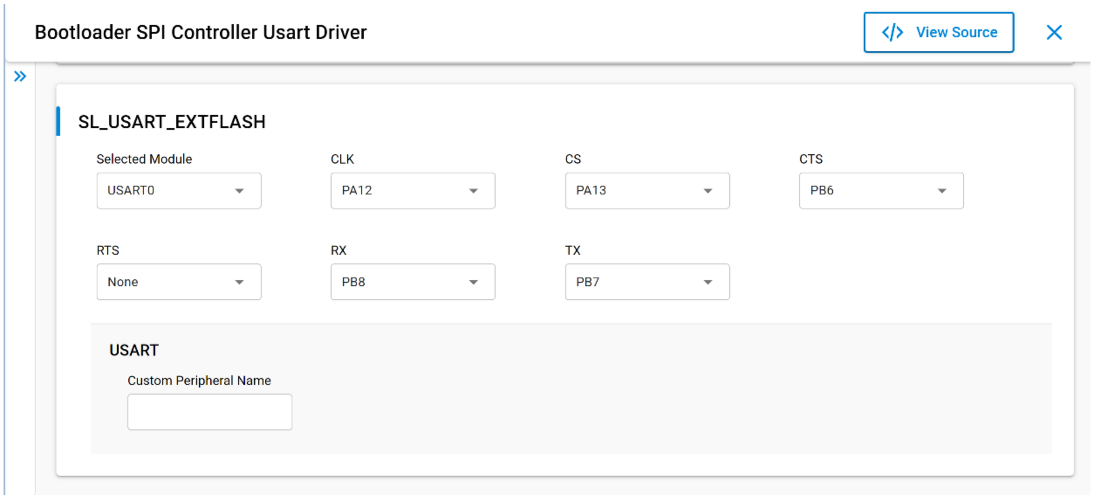
Component was successfully added.
The Bootloader SPI Controller Usart Driver was added.
View Dependencies & Instances

Bootloader SPI Controller Usart Driver

Description
`Simple Blocking SPI Usart Controller`

Quality
PRODUCTION

3. Click **Configure** to open the Component Editor and configure the USART peripheral as shown.



Bootloader SPI Controller Usart Driver

SL_USART_EXTFLASH

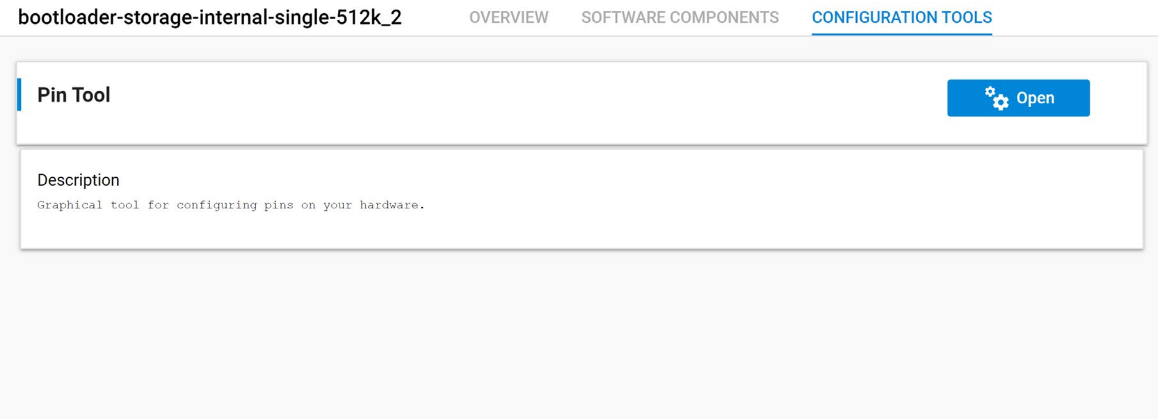
Selected Module	CLK	CS	CTS
USART0	PA12	PA13	PB6

RTS	RX	TX
None	PB8	PB7

USART

Custom Peripheral Name

4. Open the Pin Tool.



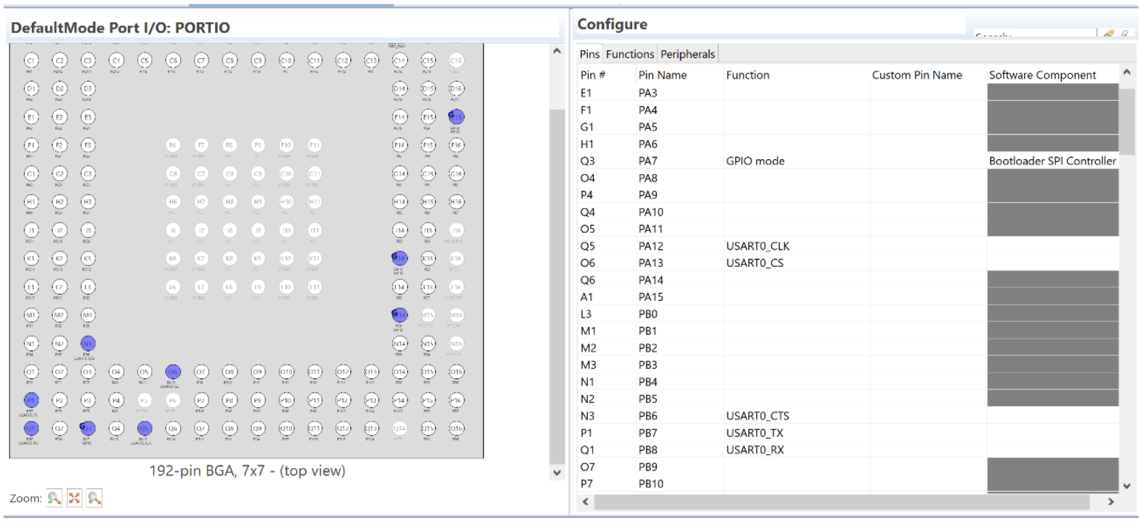
bootloader-storage-internal-single-512k_2

Pin Tool

Description

Graphical tool for configuring pins on your hardware.

5. The pin configurations are shown. Notice that pins PA12, PA13, PB6, PB7, and PB8 are reserved for USART0.



DefaultMode Port I/O: PORTIO

192-pin BGA, 7x7 - (top view)

Configure

Pin #	Pin Name	Function	Custom Pin Name	Software Component
E1	PA3			
F1	PA4			
G1	PA5			
H1	PA6			
Q3	PA7	GPIO mode		Bootloader SPI Controller
O4	PA8			
P4	PA9			
Q4	PA10			
O5	PA11			
Q5	PA12	USART0_CLK		
O6	PA13	USART0_CS		
Q6	PA14			
A1	PA15			
L3	PB0			
M1	PB1			
M2	PB2			
M3	PB3			
N1	PB4			
N2	PB5			
N3	PB6	USART0_CTS		
P1	PB7	USART0_TX		
Q1	PB8	USART0_RX		
O7	PB9			
P7	PB10			

5 Linker File

The linker files for GCC and IAR are now generated during the project generation step. The linker files for the bootloader are generated from template files using the Jinja template engine. The template files are included with theGecko SDK Suite (GSDK), which is installed through Simplicity Studio. The jinja template files are located at **<path-to-simplicity-studio-installation>\v5\developer\sdks\gecko_sdk_suite\<gecko_sdk_version>\platform\common\toolchain\**.This path contains folders for both **gcc** and **iar**.

6 Additional Macros

In the AppBuilder workflow, the user could define additional compiler macros used during compilation of the bootloader application. In Gecko Bootloader v2.x, these additional macros are now part of the various components. These components have these macros defined internally, and those macros which require user intervention are available to the user as configuration options through the Component Editor.

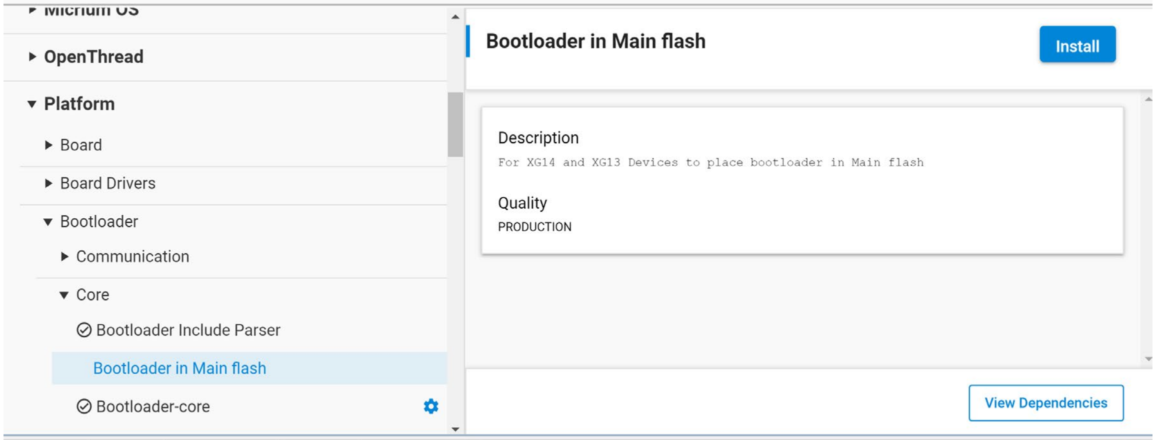
7 Postbuild Steps

For Series 1 devices, the main stage bootloader needs to be combined with the first stage bootloader, which results in a combined bootloader binary. To accomplish this, a postbuild script is made available with the GSDK, which combines the appropriate first stage bootloader with the main stage bootloader to produce a combined bootloader binary. A postbuild step must be explicitly added through the Project's settings so that the postbuild script can be run after the bootloader build is complete. For more details, see *UG489: Silicon Labs Gecko Bootloader User's Guide for GDSK 4.0 and Higher*.

Note: Adding the postbuild steps is mandatory when building bootloader binaries for Series 1 devices. If the postbuild step is not configured correctly, Simplicity Studio cannot build a combined bootloader binary image.

8 Main Bootloader in Main Flash

For xG13 and xG14 devices, the entire main stage bootloader might not fit into the bootloader flash if the user installs some extra components. In such scenarios, the main stage bootloader can be placed in the main flash by installing the **Bootloader in Main Flash** core component in the bootloader project.



Bootloader in Main flash

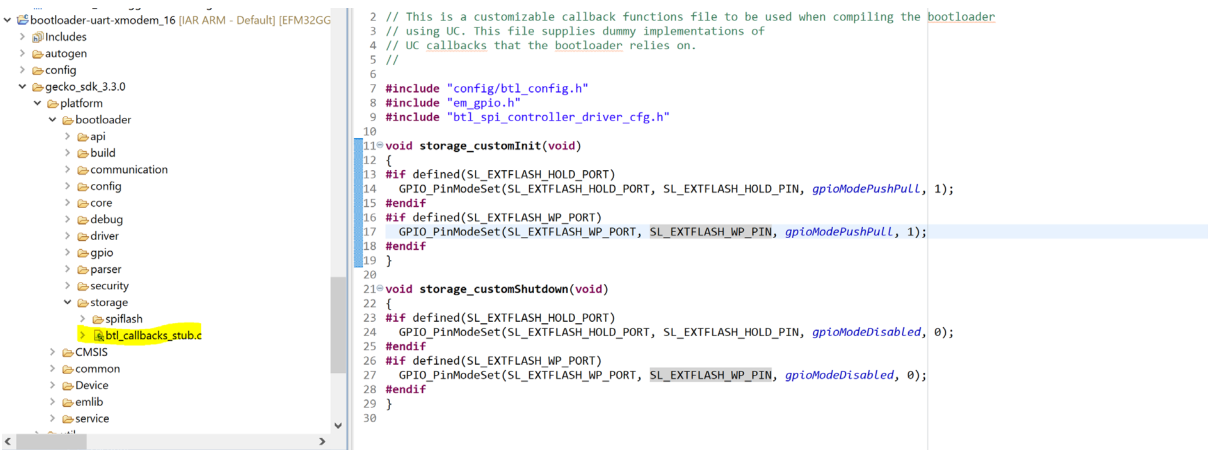
Description
For XG14 and XG13 Devices to place bootloader in Main flash

Quality
PRODUCTION

For more information on this component, see *UG489: Silicon Labs Gecko Bootloader User’s Guide for GDSK 4.0 and Higher*.

9 Callbacks

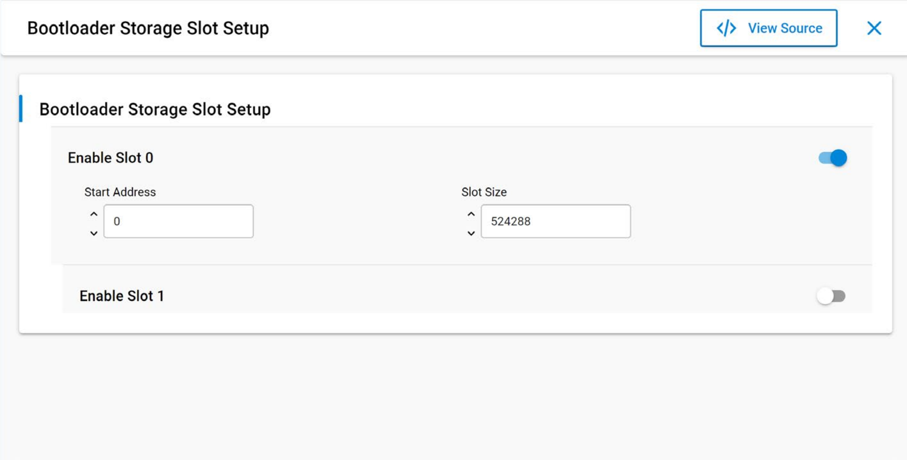
In contrast to the AppBuilder workflow, callbacks are now part of **btl_callbacks_stub.c**. This file is added to the project when the **SPI Flash Storage** component is installed. This file contains dummy implementation of callbacks that the bootloader relies on. This file can be found in **gecko_sdk_<version> > platform > bootloader > storage > btl_callbacks_stub.c** as shown.



```
// This is a customizable callback functions file to be used when compiling the bootloader
// using UC. This file supplies dummy implementations of
// UC callbacks that the bootloader relies on.
//

#include "config/btl_config.h"
#include "em_gpio.h"
#include "btl_spi_controller_driver_cfg.h"

void storage_customInit(void)
{
#if defined(SL_EXTFLASH_HOLD_PORT)
  GPIO_PinModeSet(SL_EXTFLASH_HOLD_PORT, SL_EXTFLASH_HOLD_PIN, gpioModePushPull, 1);
#endif
#if defined(SL_EXTFLASH_WP_PORT)
  GPIO_PinModeSet(SL_EXTFLASH_WP_PORT, SL_EXTFLASH_WP_PIN, gpioModePushPull, 1);
#endif
}

void storage_customShutdown(void)
{
#if defined(SL_EXTFLASH_HOLD_PORT)
  GPIO_PinModeSet(SL_EXTFLASH_HOLD_PORT, SL_EXTFLASH_HOLD_PIN, gpioModeDisabled, 0);
#endif
#if defined(SL_EXTFLASH_WP_PORT)
  GPIO_PinModeSet(SL_EXTFLASH_WP_PORT, SL_EXTFLASH_WP_PIN, gpioModeDisabled, 0);
#endif
}
```

10 Storage Slots

Gecko Bootloader supports the Application Bootloader mode. This mode supports single/multiple storage slots that can be configured at compile time. In Gecko Bootloader v2.x, the storage slots can now be configured in the **Bootloader Storage Slot Setup** component using the Component Editor. A maximum of 3 storage slots can be configured.



Bootloader Storage Slot Setup

Bootloader Storage Slot Setup

Enable Slot 0 ●

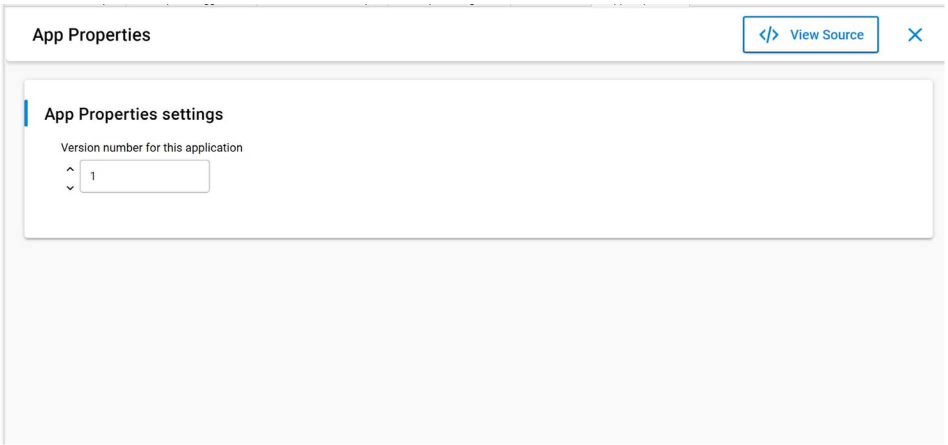
Start Address: 0

Slot Size: 524288

Enable Slot 1 ○

11 App Properties

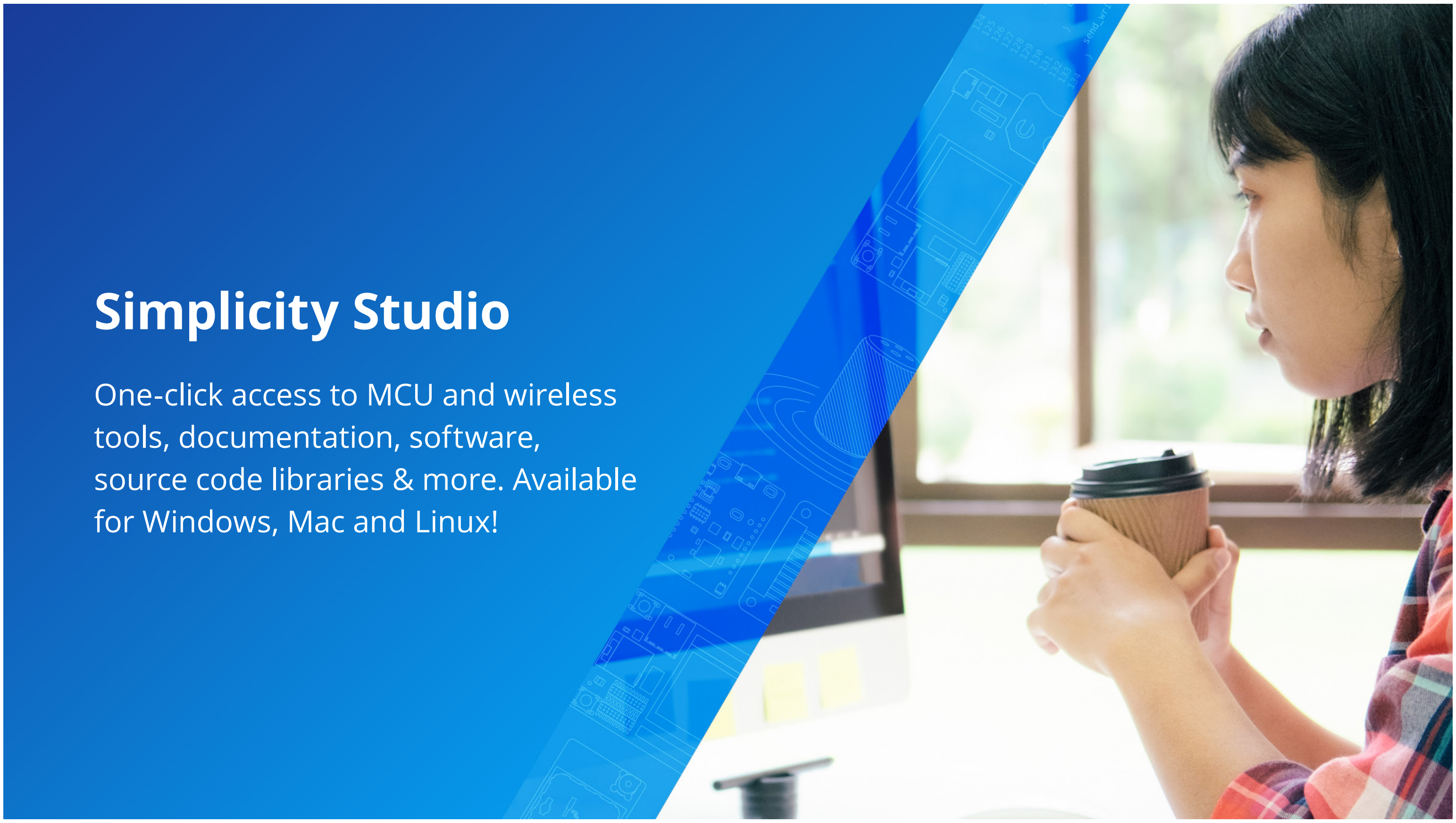
Previously, in the AppBuilder workflow, a variable of type **AppProperties_t** had to be configured to add application properties to the application. This was then later used by Simplicity Commander during gbl file creation. Beginning with Gecko Bootloader version 2.x, a new configurable component named **App Properties** can be used to configure the application properties. This component can be installed using the Component editor which is available as part of Simplicity Studio. It is also installed automatically as a dependency on installing the **Bootloader Application Interface** component in the project. The component allows the user to configure the application version using Simplicity Studio's Component Editor as shown below.


App Properties

App Properties settings

Version number for this application: 1

For more details on how to configure this component, see *UG489: Silicon Labs Gecko Bootloader User's Guide for GDSK 4.0 and Higher*.



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 & Community
www.silabs.com/community

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. **Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project**

Trademark Information

SILICON LABS

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

www.silabs.com