



Silicon Labs Gecko Bootloader

MAY 2017

Multi-platform, Unified Bootloader



Gecko Bootloader: Platform Support

- The Gecko Bootloader (BTL) has been developed as a single BTL for all stacks
- Gecko BTL is available now for
 - Zigbee
 - Thread
 - Bluetooth
 - Flex
- Gecko BTL supports
 - Various communications modes (UART XMODEM, SPI, BGAPI UART DFU, EZSP)
 - Internal & External SPI Storage
 - A variety of utilities including crypto libraries, CRC, tokens, etc.
 - Legacy .ebl files as well as the new .gbl format
 - Debug prints and asserts
 - Signed firmware, encrypted firmware and secure boot
 - GPIO “Bootloader Recovery Mode”

Gecko Bootloader: Platform Support

- Pre-xG12 EFR32 ICs support **both** legacy and Gecko Bootloader
 - EFR32xG1 (including stacked flash variants, e.g. EFR32MG1P632x/732x)
 - Prebuilt bootloaders still included for legacy configurations
 - Intent is to deprecate legacy bootloader soon
- All new chips (EFR32xG1n) will **require** the new Gecko Bootloader
 - Any legacy bootloader support from xG12 alpha is discontinued
 - This applies to EFR32xG12 and beyond
 - Prebuilt bootloader binaries + AppBuilder samples
 - Provided to replicate functionality of legacy bootloader configurations
 - Binaries only provided for typical use cases on dev kit variants of radio boards
 - SoC and UART NCP bootloaders for Bluetooth and mesh – no security features
 - Only a few MG12/BG12 board variants found in new BLE or mesh dev kits

Gecko Bootloader: Two-stage Design

- Enables in-field upgrades of the bootloader
- First stage
 - Contains just enough code to be able to update the main bootloader (<2kB)
- Main bootloader
 - Contains everything necessary to update the application (~12-13kB)
 - EBL/GBL file parser
 - SPI flash storage driver
 - Internal flash storage driver
 - NCP protocol parsers
 - Security features

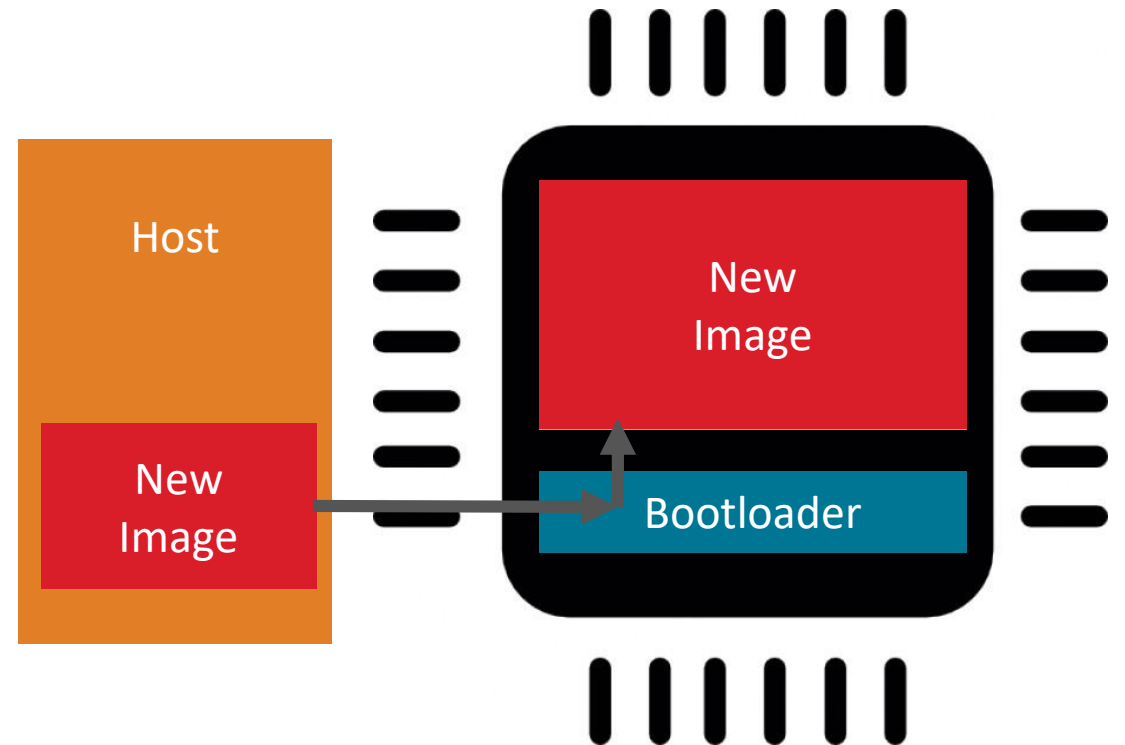


Gecko Bootloader: Device Memory Map

- On EFR32xG1, bootloader resides in main flash
 - First stage bootloader @ 0x0
 - Main bootloader @ 0x800
 - Application @ 0x4000
- On newer devices (xG12 & beyond), bootloader resides in bootloader area
 - Application @ 0x0
 - First stage bootloader @ 0x0FE10000
 - Main bootloader @ 0x0FE10800
 - Bootloader flash is *not* erased with commander device masserase or debug-unlock
 - To remove the bootloader, a new dummy bootloader has to be flashed (these are included)
 - Crypto keys are stored in Lockbits page, which *is* erased
- Firmware update content can reside in upper flash or ext. Dataflash
- Storage Slot concept allows for >1 image stored in download area
 - Facilitates multi-boot use cases

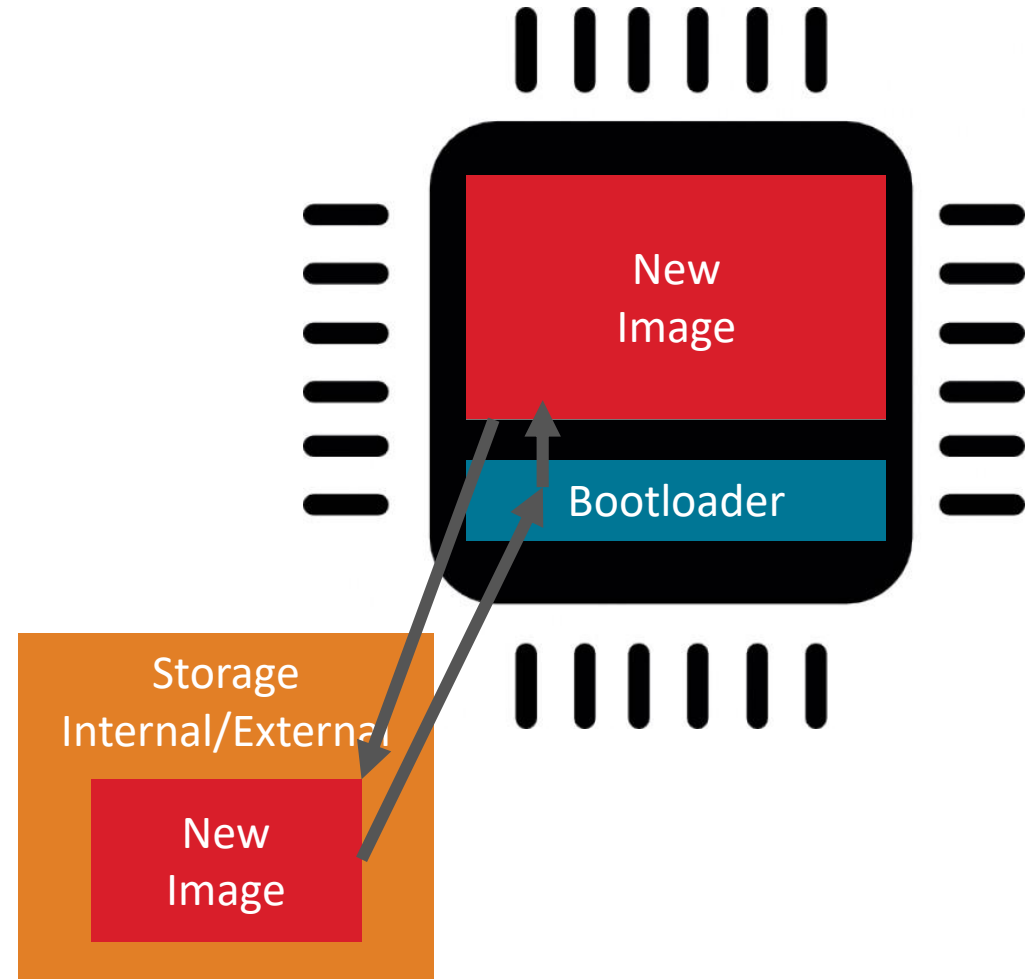
Gecko Bootloader: NCP Bootloader Operation

- The device reboots into the bootloader.
- A GBL file containing an application image is transmitted from the host to the device via UART or SPI
- The bootloader applies the application upgrade from the GBL upgrade file on-the fly into App space (i.e. “standalone bootload” use case).
- The device boots into the application. Application upgrade is complete.



Gecko Bootloader: SoC Operation

- GBL file downloaded onto the storage medium of the device (e.g. int. flash or ext. SPI flash) via App or pre-programming.
- At application's discretion, downloaded image is verified and asked to be installed (i.e. "application bootload" use case).
- The device reboots into the bootloader.
- The bootloader applies the application upgrade from the GBL upgrade file into App space.
- The device boots into the application. Application upgrade is complete.



Some Details about the Gecko Bootloader

- Legacy Ember bootloaders used EBL (Ember BootLoad) version 2
 - Can be optionally supported by Gecko Bootloader for reuse of files across generations
 - Gecko Bootloader not compatible with encrypted EBLv2 files
 - Legacy bootloader does not use older .ebl security
- Some changes to add security features, MCU compatibility, etc
- New Simplicity Commander syntax (not image builder) for app and btl
- App: `commander gbl create <newFile.gbl> --app <myapp.s37>`
- AppBuilder Gecko Bootloader build creates:
 - Bootloader .s37 file
 - Bootloader .gbl file
 - Bootloader “-combined.s37” file with both first- and second-stages in one file
 - Once a chip has a first-stage bootloader, only the second-stage needs to be loaded when changed
- `commander gbl create <newFile.gbl> --bootloader <mybtl.s37> --device EFR32...`
 - Using the `--bootloader` flag with `--device` ensures positioning in bootloader location for the part

Gecko Bootloader: Self-Upgrade Mechanism

- Main bootloader is field upgradeable
 - Allows for critical bug fixes, changes in bootloader features
 - Contains logic for acquiring/verifying images, knowledge of file format, etc.
- First stage is very simple and only knows how to upgrade the second stage
 - Only contains basic flash read/write/verify code for main flash controller
 - Not upgradeable, but fewer pieces to 'go wrong'
- Security of the bootloader upgrade image is provided by signing, encrypting of the GBL
- Can't field-upgrade from legacy bootloader to Gecko Bootloader

Gecko Bootloader: Security Features

- Secure Firmware Update
 - Signed GBL file: ECDSA-P256 → ensures only accepting images trusted by vendor
 - Encrypted GBL file: AES-128 → ensures encryption during transfer, image storage
 - Verification of new image before bootloading
- Secure Boot
 - Verification of image in flash before running
 - Signed application image: ECDSA-P256
- In-field upgradeable
 - 2 stage bootloader allows for security or other features to be added after deployment
- Encryption is hardware accelerated

Gecko Bootloader: Security Features

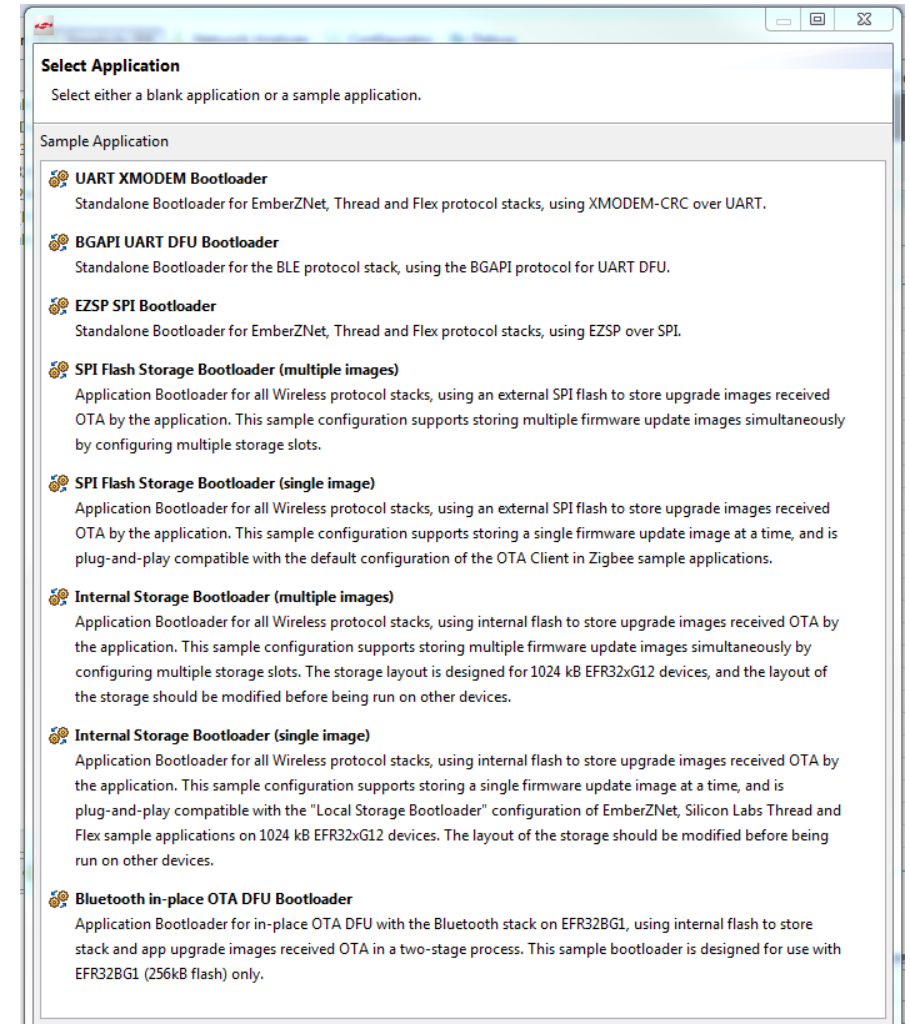
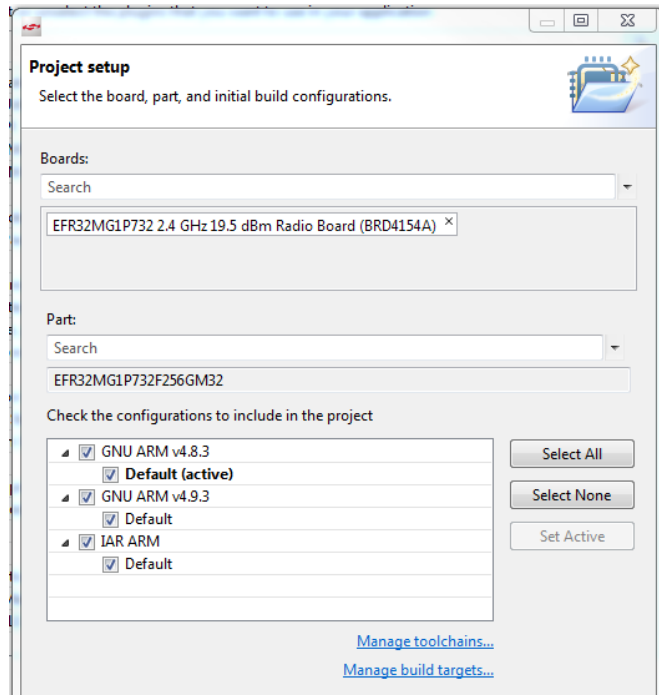
- **GBL File Signing using Public Key Cryptography Verification of new image before bootloading**
 - Manufacturer installs a public key in each device
 - Private Key is held only by the manufacturer
 - GBLs are signed using this key at production time
 - Devices in the field validate the signature before installing
 - Signing is supported using ECC and the P-256 curve (NIST approved)
 - Message Digest is calculated with SHA-256

Gecko Bootloader: App Builder Integration

- EM3xx Bootloader and prior EFR32 Bootloaders were provided as binaries
 - Designed to work with Dev Kits
 - EFR32 BTLs could be modified manually with header edits and recompiled
- Gecko Bootloader has full App Builder support
 - Most features selectable in interface
 - Features are provided as plugins which can be edited by customers if necessary

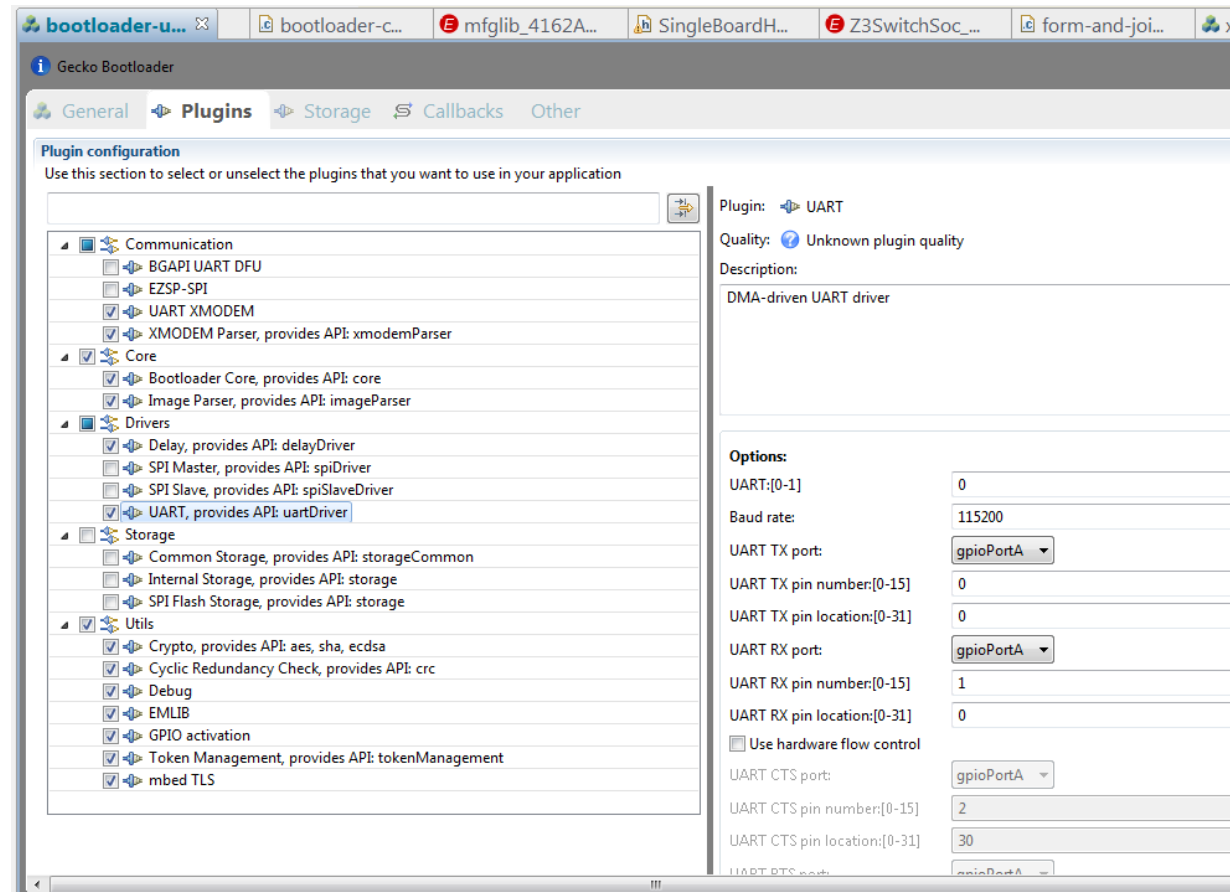
Gecko Bootloader: App Builder Interfaces

- Gecko Bootloaders are presented like Sample Apps
- Specific to the stack and target part/board
- Can be compiled with either IAR or GCC



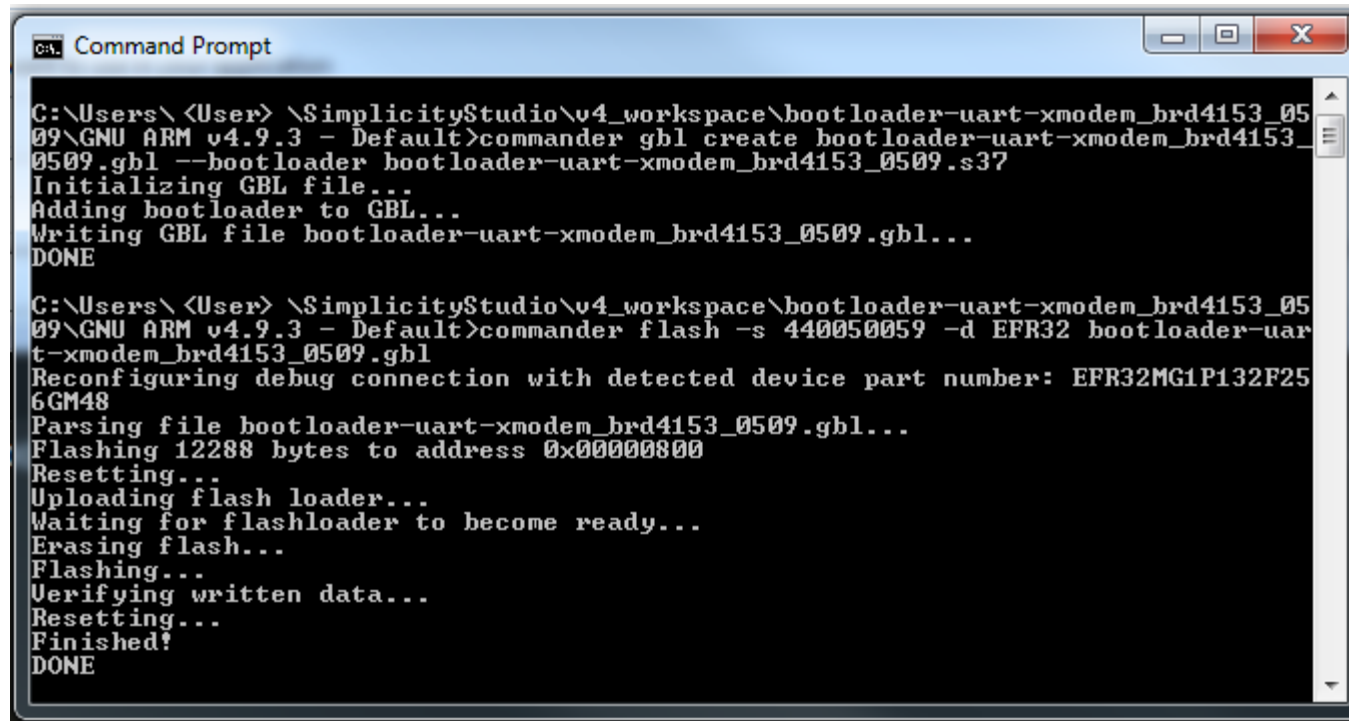
Gecko Bootloader: App Builder Interfaces

- Plugins are used to select options
- Pinouts, speeds, flash memory parts, etc., are made available graphically, avoiding header edits



Gecko Bootloader: Simplicity Commander

- Bootloaders are compiled like other apps and result in .s37 binaries
- Compile builds Second Stage as .s37; post-build script adds -combined.s37 with both stages
- Simplicity Commander is used to convert to .gbl for bootloader OTA (2nd Stage only)
- Commander can be used to upload bootloader file, as can other tools



```
ca. Command Prompt
C:\Users\
```

Thank you

