AN14525

MCXA14x/15x Secure Boot Based on MCUBoot

Rev. 1.0 — 10 January 2025 Application note

Document information

Information	Content
Keywords	AN14525, MCXA14, MCXA15, MCUboot, secure boot
Abstract	This document is based on the MCUboot and OTA application inside the MCUXpresso SDK for MCXA14x/A15x.

1 Introduction

MCXA14 and MCXA15 microcontrollers, featuring an Arm Cortex-M33 core, operate at speeds of up to 96 MHz with high levels of integration and analog. They address a wide range of applications with scalable device options. The low-power and intelligent peripherals include timers that generate three complementary PWM pairs with deadband insertion, 16-bit ADC with hardware windowing and averaging features. The innovative power architecture is designed to support high utilization of I/Os and power efficiency with a simple supply circuit in a smaller footprint.

MCXA14 and MCXA15 contain ROM and extended bootloader, but does not support secure boot, therefore, this document uses MCUboot to implement the secure boot function.

MCUboot is a secure bootloader for 32-bits microcontrollers. MCUboot defines a common infrastructure for the bootloader and the system flash layout on microcontroller systems. It also provides a secure bootloader that enables easy software upgrade.

To achieve secure boot, use secure features (MBC, GLIKEY, UUID, Code Watchdog, and so on) and ROP mechanisms in MCX A, based on the open source MCUboot.

1.1 MCUboot

MCUboot is an open source project with Apache-2.0 license.

MCUXpresso SDK for MCXA14x/A15x supports MCUboot with secure boot and an example application is provided to demonstrate the OTA process.

MCUboot supports either swap-based or overwrite-based image upgrades. The overwrite upgrade strategy is substantially simpler to implement than the image swapping strategy. It does not require an extra area (scratch area or extra sector for swapping). The MCXA14x/A15x features a small flash/memory footprint prompting the SDK to choose the overwrite upgrade strategy as the default approach.

For more information on the top-level design of MCUboot, see mcuboot.

This document is based on the MCUboot and OTA application inside the MCUXpresso SDK for MCXA14x/A15x. The software in this document adds the hardware security features of the MCX A to the SDK MCUboot example and uses the SDK original OTA example.

1.2 MCXA14x/A15x security features

MCXA14x/A15x offers a range of basic security features and secure boot can benefit from these features as follows:

- Memory block checker (MBC): In the example, the bootloader (MCUboot) configures the MBC to hide itself and configures the permissions for the rest of the flash.
- GLIKEY: GLIKEY provides state machine protection for writing of security-sensitive registers, which helps protect the registers from accidental or malicious modification and glitching.
- Code Watchdog (CDOG): In the example, the application (OTA example) uses CDOG to protect the debug unlock function.
- Read-out protection (ROP): ROP allows the users to enable different levels of protection in the system.
- UUID: 128-bit universal unique identifier (UUID).

The following sections briefly describe these features. For detailed information, see the *Reference Manual*, and the accompanying example source code.

1.2.1 Memory block checker

For MCXA14x/15x main flash, MBC implements access controls for on-chip flash based on a fixed-sized block format. The block size is 8 kB. MBC provides hardware access control for system bus references targeted at on-chip memory spaces.

To control the read/write/execute/lock access to the flash block, apply the functionality of MBC. Therefore, flash can be divided into different partitions with different permissions.

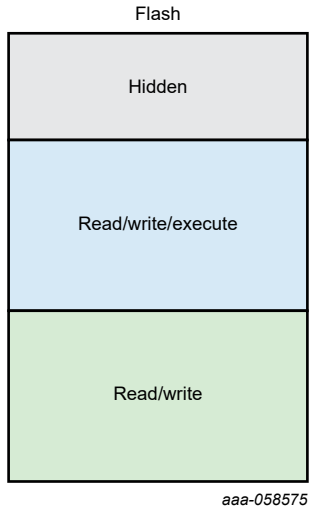

Figure 1. Example of MBC configuration

Figure 1 is an example of MBC configuration. In this example:

- A portion of the flash is configured to be "Hidden", that is, not allowed to be read, written, or executed.
- A portion of the flash is configured as "Read/write/execute", that is, read/write and execution are allowed.
- A portion of the flash is configured as "Read/write", that is, read/write is allowed, however, execution is not allowed.

1.2.2 GLIKEY

GLIKEY controls the write enables and resets for the security-sensitive registers through FSM. Software has to follow a strict procedure to enable the write access and resets for security-sensitive registers.

- It provides integrity protection of security-sensitive registers during write and at rest against power glitch attacks.
- It provides integrity protection of security-sensitive registers at rest against privilege escalation.
- GLIKEY also supports error management with dedicated interrupt and dedicated integrity protection.

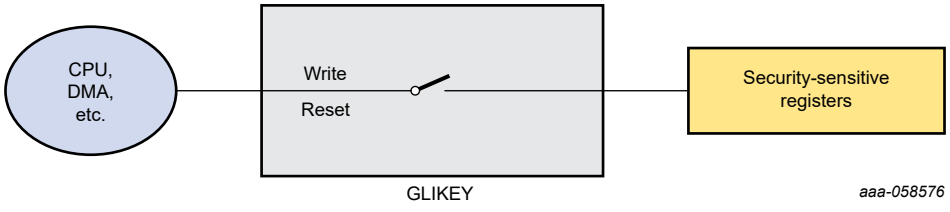

Figure 2. GLIKEY

1.2.3 Code Watchdog

CDOG helps protect the integrity of software by detecting unexpected changes (faults) in code execution flow. This module can be configured to reset or interrupt the processor core when it detects a fault.

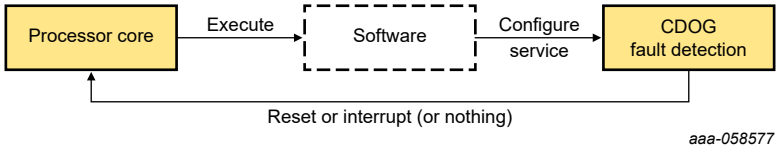

Figure 3. CDOG overview

CDOG includes the following code flow and data integrity checking:

- Faults and flags are configurable to generate a system reset, interrupts, or nothing
- Counters for statistics on code behavior patterns for fault types

CDOG provides the following two primary mechanisms for detecting low-cost fault attacks and the execution of unexpected instruction sequences:

1. Secure counter: It is a 32-bit accumulator that holds a dynamically changing value that can periodically be evaluated to determine if a program is executing as expected. If a mismatch is detected, a fault is generated.
2. Instruction timer (INSTRUCTION_TIMER): It is a 32-bit countdown timer that an application uses to set the number of instructions that software expects to execute. The instruction timer counts off the instructions as they are executed (clocks). If the number is exhausted before the CDOG is serviced, a fault is generated.

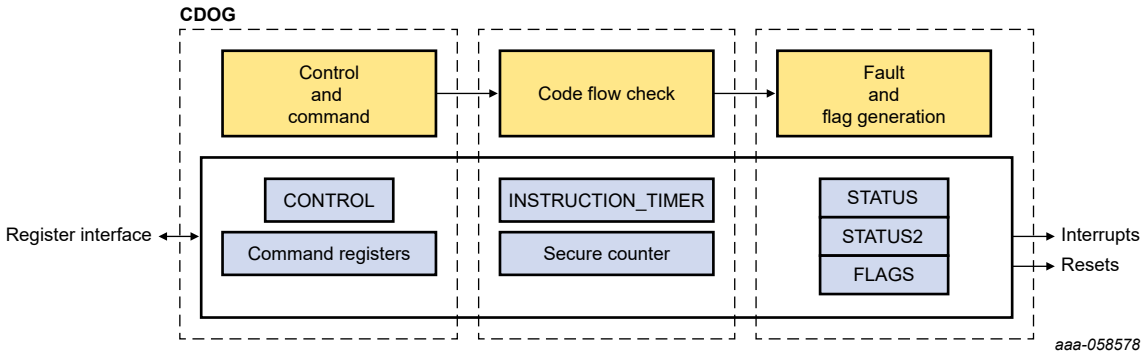

Figure 4. Block diagram of CDOG

1.2.4 Read-out protection

MCXA14x/A15x supports the life cycle states given in Table 1.

Table 1. Life cycle states

Life cycle	Life cycle type	Description
Develop	Customer	Initial customer development state after leaving NXP manufacturing
In-field ROP1/ROP2	Customer	In-field application state with ROP protection
In-field ROP3	Customer	In-field application state with ROP protection and prevents use of field return
Bricked	End-of-life	Bricked state to prevent device use

Based on this information, it supports four levels of ROP, also called ROP_STATE.

This ROP is a mechanism that allows the user to enable different levels of protection in the system. It is a 32-bit field stored in IFR0. Customers can program it.

Figure 5 shows the different debug permissions, ISP commands, and SWD DM-AP commands supported under different life cycles.

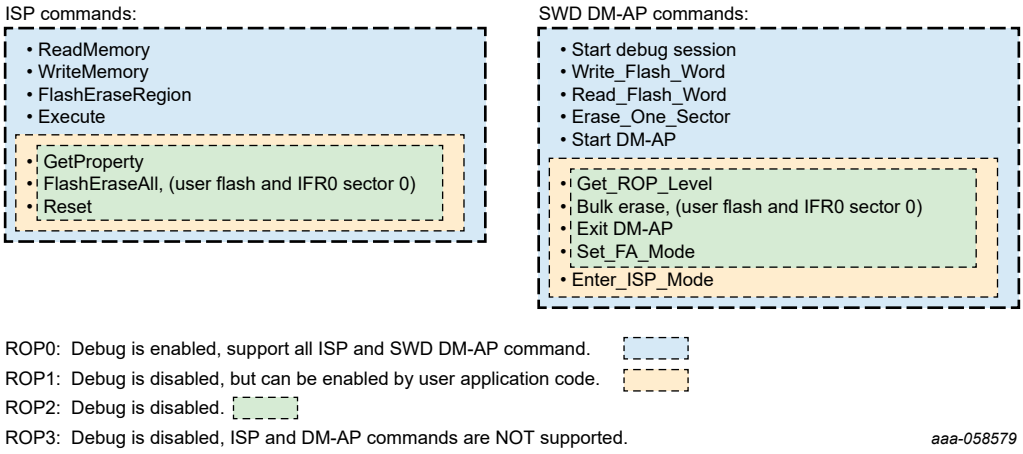

Figure 5. ROP debug/ISP/SWD DM-AP

1.2.5 Universal unique identifier

MCX A devices support a 128-bit UUID per device in accordance with IETF's RFC4122 version 5 specification. In this example, the UUID is used for managing debugging permissions, see Section 2.3.1.

Note: *On the early FRDM-MCXA153 boards featuring early engineering chips, the first line of the package marking begins with "**P**MCXA". The UUID of all these chips is 0xFF("FFFF....FF"). In contrast, officially ordered chips have package markings starting with "MCXA" and contain the correct UUID, eliminating this issue.*

2 MCXA14x/A15x secure boot based on MCUboot

This chapter introduces the overall boot flow, architecture, and the hardware functions of the MCX A series used by the MCUboot.

2.1 MCX A14x/A15x boot flow

After any reset, including POR reset and warm reset sequence, the CPU always runs to boot ROM. The main functions of the ROM are as follows:

- Check life cycle
- Process mailbox request through debug access port
- Configure the GLIKEY and MBC
- Run image integrity check based on wake-up source
- Before exiting to extended bootloader (on IFR0), hides critical sections
- ROM provides a flash driver API to the user

Then, the ROM jumps to the extended bootloader. The extended bootloader is the secondary bootloader, which is located in IFR0, sector 1 ~ 3, total 24 kB, and provisioned during the NXP manufacturing process. It checks the configuration, enables the booting interfaces, performs ISP command following the NXP bootloader protocol, and other tasks.

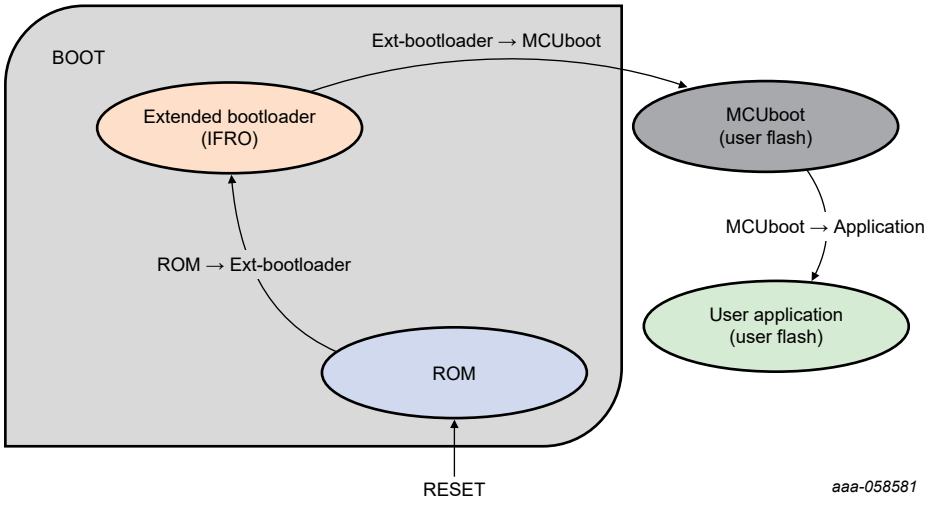

Figure 6. Boot flow

Extended bootloader jumps to user flash after execution, however, in this example, it jumps to MCUboot. If the application and the signature are valid, MCUboot jumps to the application.

2.2 Software architecture

The software package contains two MCUXpresso IDE projects:

- `frdmmcxa153_mcuboot_opensource`: It is the bootloader based on MCUboot, and must be downloaded to the "MCUboot" region shown in Figure 7.
- `frdmmcxa153_ota_mcuboot_basic`: It is an example of an application that provides several commands used to demonstrate the OTA abilities. It must be downloaded to the "Primary slot" region shown in Figure 7.

Figure 7 shows the flash layout of the example:

- The MCUboot region is used to store MCUboot.
- The primary slot is used to store the application.
- The secondary slot is used to store the upgrade firmware during the upgrade.
- The optional user data region is used to store parameters and other data.

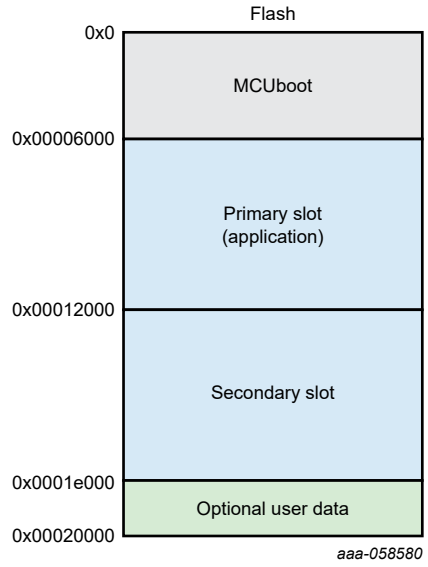

Figure 7. Flash layout

2.3 Life cycle and ROP

In this example, the chip must be set to In-field ROP1. Under In-field ROP1, ISP commands and DM-AP commands are limited, and the ROM disables the debug after booting. Therefore, the contents of the flash cannot be read via ISP or SWD. This example enables the debug when the password is passed in the application.

Note: *In practice, the level of ROP protection can be adjusted as needed. For higher security requirements, ROP2 or ROP3 can be used.*

2.3.1 Debug control

Under In-field ROP1, the debug is disabled by default. DEBUG_FEATURES/DEBUG_FEATURES_DP and SWD_ACCESS_CPU0 in SYSCON can be used to re-enable the debug.

In this example, the application project provides a command to re-enable the debug. For details, see `shellCmd_swd` in `frdmmcxa153_ota_mcuboot_basic/source/ota_mcuboot_basic.c`.

This command uses the UUID of the chip as the password to enable the debug.

Warning: *As UUID can be obtained through ISP command `GetProperty` under In-field ROP1, do not use it as a password in the real product. Use other confidential information such as the unique serial number of the product defined by the OEM.*

Warning: *To avoid storing the unlocked password as plaintext on the device, the HASH of the password must be stored. During runtime, the HASH of the entered password is calculated and compared with the stored HASH to verify its authenticity.*

The registers related to debug control are protected by GLIKEY. For information on how to operate the GLIKEY to modify these registers, see the source code.

This command also uses CDOG for protecting code execution flow and data integrity.

Note: *In this example, only `shellCmd_swd` command is protected by CDOG. The protection capabilities provided by CDOG can be extended to other functions, such as checking of signed images in MCUboot as required.*

2.4 Flash access control capability

Take advantage of the flash access control capability provided by MBC to implement an immutable bootloader.

In this example, FLASH_ACL_x_x fields in IFR0/CMPA are configured to provide the flash access permission to read/execute, after booting from ROM/extended bootloader. Then, MCUboot configures the other flash regions and hide itself to read/write for OTA demonstration. After verifying the application, MCUboot hides itself and lock the configuration before jumping to the application.

Note: *Lock means until reset, its permission cannot be modified.*

Figure 8 shows the configuration flow.

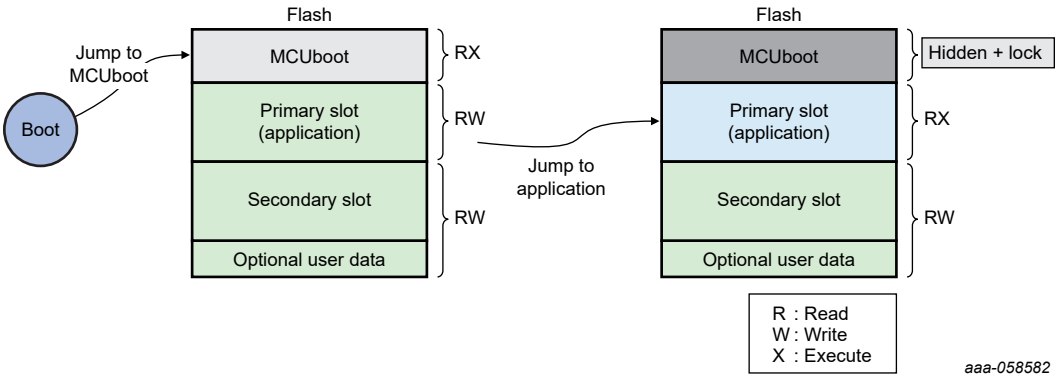

Figure 8. Flash access control

The flash region, which stores MCUboot, has no write access during the entire execution period. Therefore, it can be assumed that the MCUboot flash region is immutable.

3 Demonstration

This chapter demonstrates how to provision the chip and use the accompanying software package.

3.1 Prerequisite

It is assumed that the user is aware of the basic operation of the MCUXpresso IDE and FRDM development board, see Getting Started with FRDM-MCXA153.

3.1.1 Hardware and software requirements

Table 2 describes the hardware and software requirements for using the software pack.

Table 2. Hardware and software details

Category	Description
Hardware	• NXP FRDM-MCXA153 board shown in Figure 9 • Type-C cable • Personal computer – Use a USB cable to establish a connection between the FRDM-MCXA153 board and the computer. • J15 is an onboard-debugger USB port of MCU-Link. • J8 is FS USB port of MCXA153 and is used for USB ISP communication.

Table 2. Hardware and software details...*continued*

Category	Description
Software	• AN14525.zip folder: – `frdmmcxa153_mcuboot_opensource` – `frdmmcxa153_ota_mcuboot_basic` – Secure Provisioning Tool workspace • MCUXpresso IDE 11.10.0 or later • MCUXpresso Secure Provisioning Tool V9 or later

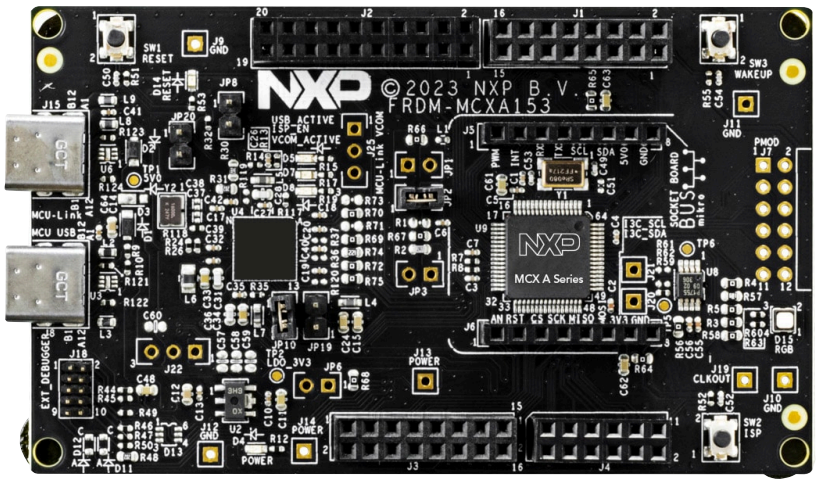

Figure 9. FRDM-MCXA153

3.2 Step-by-step implementation

To implement the method of this document, perform the following steps:

1. Import and compile MCUXpresso IDE projects:
   To generate the firmware, import the `frdmmcxa153_mcuboot_opensource` and `frdmmcxa153_ota_mcuboot_basic` projects to the MCUXpresso IDE, and then compile the projects.
2. Provision the device, sign the application, and program the firmware (MCUboot (bootloader) and signed application):
   MCUXpresso Secure Provisioning Tool is used to provision the device, for example, the flash access control in IFR0/CMPA. Also, the software facilitates the generation of MCUboot signed images.
   The provisioning process and downloading MCUboot and signed application image can be completed in a single step.
   a. Open MCUXpresso Secure Provisioning Tool and select the workspace, as shown in Figure 10.

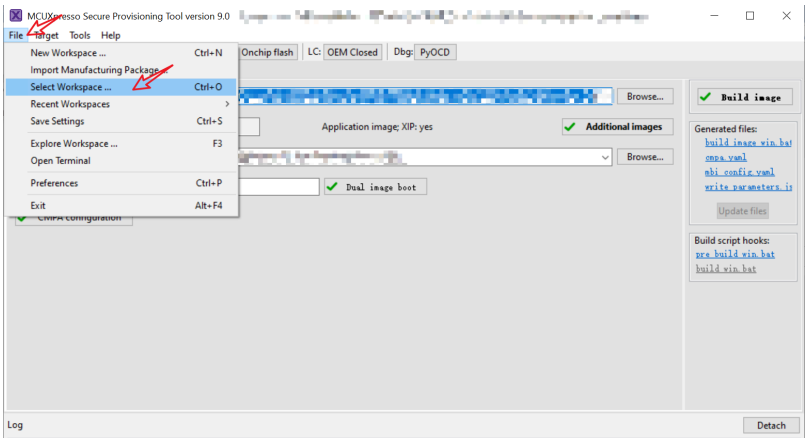

Figure 10. Select the workspace

b. Select the `frdmmcxa153_mcuboot_opensource.axf` (generated by compiling the `frdmmcxa153_mcuboot_opensource` project) as the source executable image.

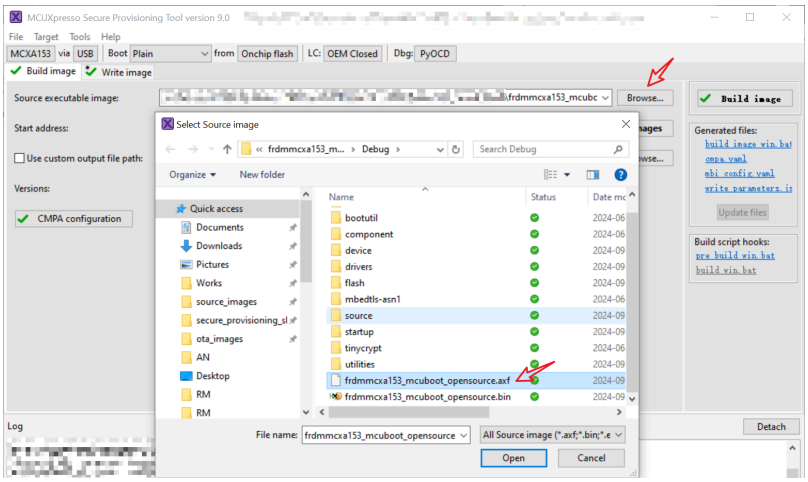

Figure 11. Select the source executable image

c. Open the MCUboot configuration window.

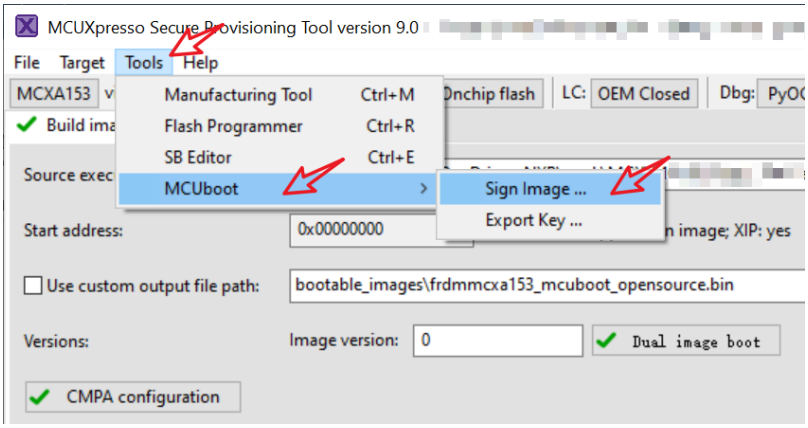

Figure 12. Open the MCUboot configuration window

d. Select the input image and the key. Select `frdmmcxa153_ota_mcuboot_basic.axf` as "Input Image", it is the application for this demo and generated by compiling the `frdmmcxa153_ota_mcuboot_basic` project. Select `sign-ecdsa-p256-priv.pem`, which is inside

the `frdmmcxa153_mcuboot_opensource` project as the "Signing priv.key". Then, click **Sign** to sign the application image.
The signature is in accordance with the MCUboot specification. For further details, see mcuboot.
Note: *The keys in the attached project and workspace are for demonstration purposes only, do not use these keys for real project.*

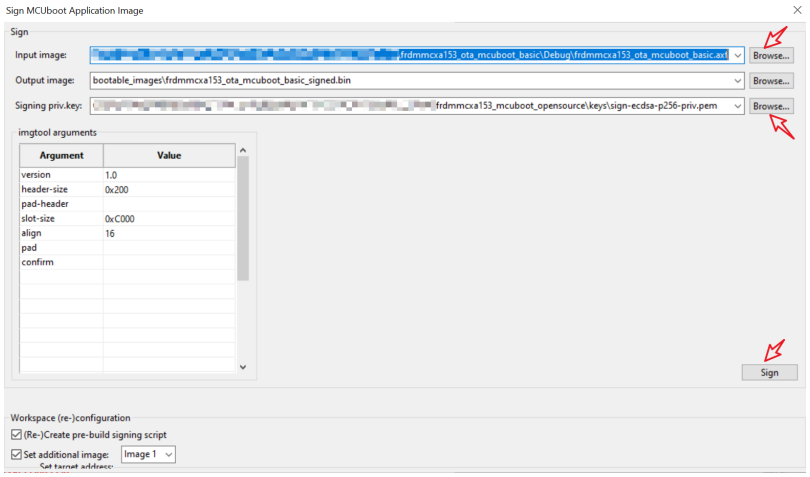

Figure 13. SEC MCUboot configuration

e. To check the CMPA, click the **CMPA configuration** button, as shown in Figure 14. The flash access control and ROP configuration must be as shown in Figure 15.

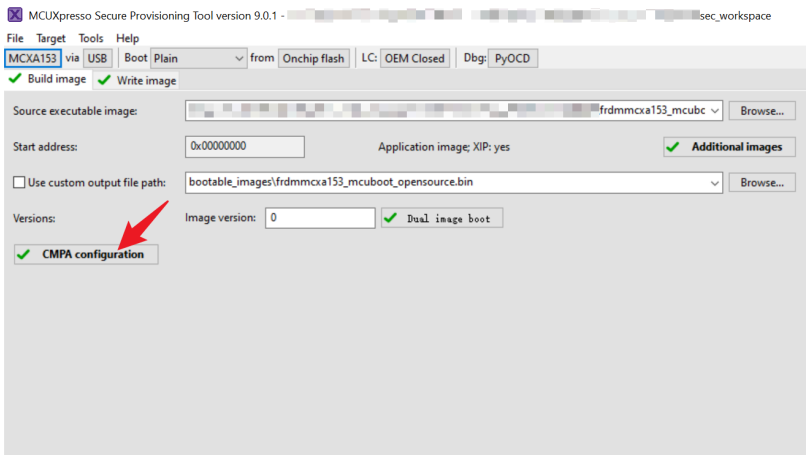

Figure 14. Click CMPA configuration

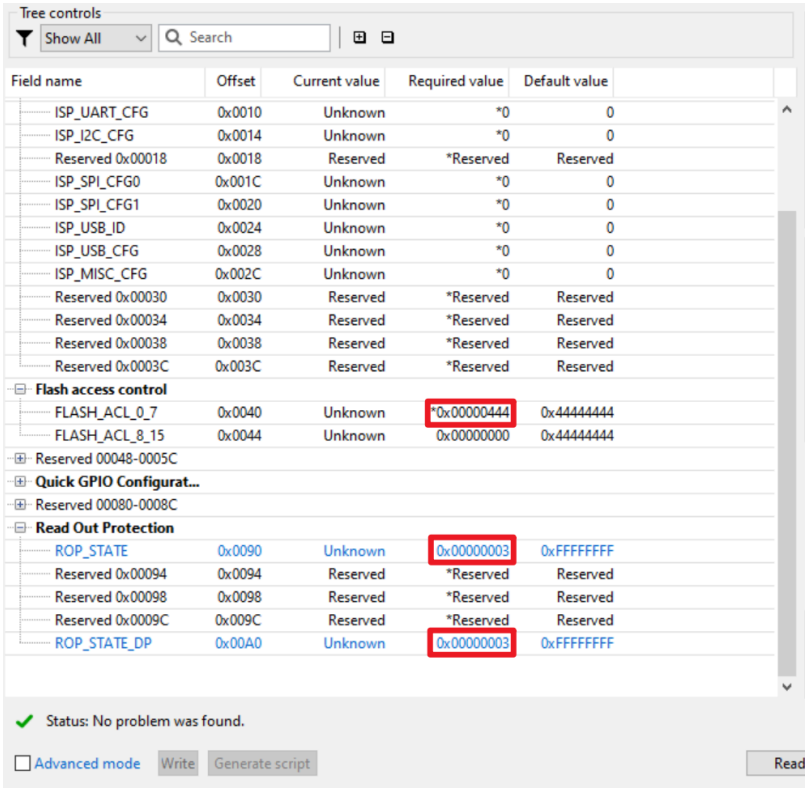

Figure 15. CMPA configuration

f. Select the signed binary file of the application. To check the configuration, click the **Additional images**, and then click **Build image**, as shown in Figure 16.

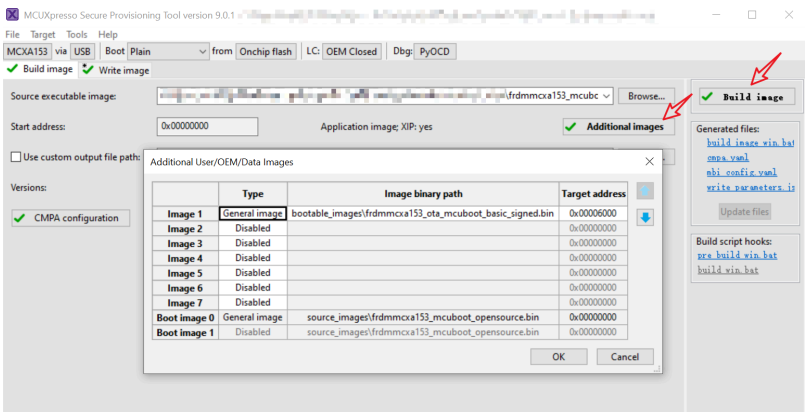

Figure 16. Build image

g. Use a USB cable to connect the J8 USB port to the PC. Hold the ISP button (SW2) and press the reset button (SW1). To make the chip enter ISP mode, release the reset button (SW1) first, then the ISP button (SW2).
Click **Write image** to write the CMPA, MCUboot (bootloader), and signed application together.

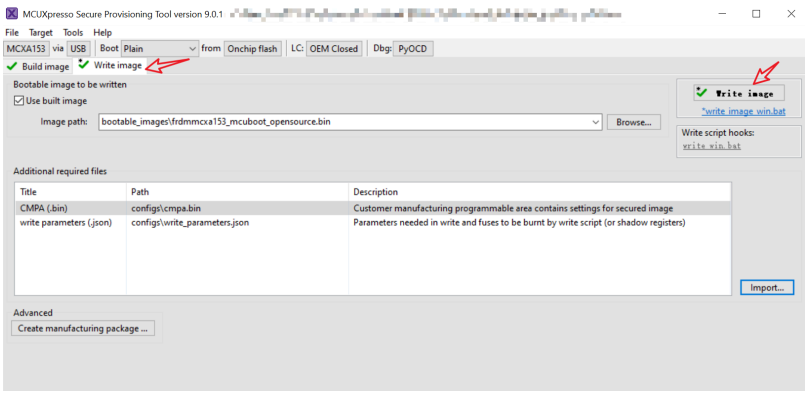

Figure 17. Write image

h. Reset the board, using terminal software (for example, PuTTy) to check the UART log as shown in Figure 18. It supports several commands.
   - `mem` command can be used to check if the flash region has stored MCUboot as "hidden" as expected or not.
   - As the life cycle is set to In-field ROP1, the debug is disabled by default. The `swd` command can be used to re-enable the debug features.
     After unlocking, do not reset the chip to avoid the debugging port being disabled after the reset. Therefore, use the "attach" function of the debugger to debug without resetting.

```
************************************
* Basic MCUBoot application example *
************************************

Built Jul  2 2024 18:33:18
Please use ISP blhost with 'get-property 18' to get uuid
--- !Please DO NOT use UUID as password in REAL products! ---
$
help
help

"help": List all the registered commands

"exit": Exit program

"image [info]"          : Print image information
"image test [imgNum]"   : Mark candidate slot of given image number as ready for test
"image accept [imgNum]" : Mark active slot of given image number as accepted
"image erase [imgNum]"  : Erase candidate slot of given image number

"xmodem [imgNum]": Start receiving with XMODEM-CRC

"mem read addr [size]" : Read memory at given address
"mem erase addr "      : Erase sector containing given address

"reboot": Triggers software reset

"swd unlock [key/password]" : Unlock the SWD debug port
"swd lock "                 : Lock the SWD debug port
$
```

Figure 18. Terminal log

3. Restore the chip to its initial state:
   - After completing all the experiments, the chip can be restored to its initial state, which allows the user to make more attempts during the development process.
   - Use the SEC tool to erase the chip (including the CMPA).
   - Click the **Erase** button to perform the `DM-AP Bulk Erase` command.

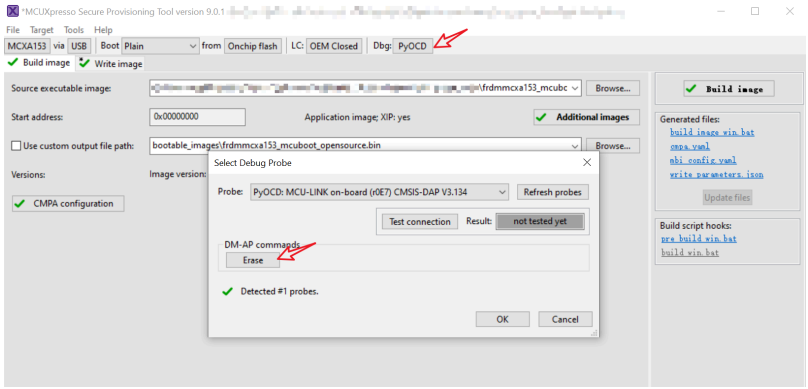

Figure 19. DM-AP Erase

4 Revision history

Table 3 summarizes the revisions to this document.

Table 3. Revision history

Document ID	Release date	Description
AN14525 v.1.0	10 January 2025	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents