AN14657

Getting Started with Secure Boot on MCX W23

Rev. 1.0 — 5 August 2025 Application note

Document information

Information	Content
Keywords	MCX W23, Secure boot, MCXW23, MCX W23, Health care, BOD, shelf, privileged, TrustZone, NHS5204, Ultralow power, Small footprint, Bluetooth Low Energy, Integrated flash, Security, IoT, Coin battery, Small Body-worn device
Abstract	This application note covers the design of the bootloader on MCX W23 and how to use its all features.

1 Introduction

This application note covers the design of the bootloader ROM code that NXP has developed on the MCX W23, and how to use all its features. There is a particular focus on booting applications securely. Application refers to a piece of code that the ROM bootloader of NXP loads. It can be an OEM secondary bootloader or an OEM application. Securely means only executing code if it is verifiably coming from a trusted source (authenticity) and only after a check has been done to ensure that it has not been modified (integrity).

During the application development phase, the bootloader covers the possibility to boot the application temporarily without the additional layer of security. However, its final purpose remains to boot an application securely when its development has been finalized.

The use of the bootloader features is illustrated with an example, using the secure provisioning tool (SEC tool) that NXP has created, for an intuitive, visual experience.

This application note refers to the security primitives described in a white paper, referred to with the abbreviation "SEC_PRIM". For more information, see Security Primitives: Requirements in (I)IoT Systems.

2 Bootloader functionality

The bootloader code in the ROM is the first piece of software to be run on the processor after a reset event. It checks and configures the platform components using available settings in the protected flash region (PFR). Then it lets the user choose between three operation modes:

- To enter the debug mailbox after receiving a specific command over the serial wire debug (SWD) connection within a specific time window. This option is extensively discussed in the MCX W23 Reference manual (Reference 6).
  Note: *For more information, contact your local Field Application Emgineer (FAE) or NXP representative.*
  One of the features of the debug mailbox is to enable the sending of commands over the SWD connection and so manage the platform memory, if it is allowed in the debug settings.[1]
- To enter the in-system programming (ISP) mode, a special bootloader mode that enables external communication with the bootloader ROM API to:
  - Manage the content of memories, and manually launch the execution of an application in memory without running through the bootloader again.
  - Read immutable platform properties.
  - Read the platform life-cycle state and application boot status.
  - Read and change a few mutable platform properties.
  - Read and change boot configurations of the platform stored in CMPA.[2] and CFPA[3]
- To search at several locations in memory for a valid OEM application code (image) to run. It is called "boot". If the image type allows it, booting the image can happen securely. That is, using additional mechanisms that:
  - Get the platform itself to a known secure state during start-up, before jumping to an image code.
  - Verify that the image code has not been altered (either maliciously or accidentally) and that its content and origin are still genuine and traceable (concepts of "authenticity" and "integrity" of the code executed).

  This feature is commonly called "secure boot". A formal definition is given in the *Section 4.11 of the SEC_PRIM white paper*. For more information, see Security Primitives: Requirements in (I)IoT Systems.
  If booting a valid image is impossible and no critical errors have bricked the part, the bootloader falls through to the ISP mode. It can be disabled in the platform security settings.

1 Happens when a developer clicks "Program" or "Debug" or "Erase" in an IDE.
2 Customer manufacturing programmable area, containing customer-defined configurations.
3 Customer in-field programmable area, containing updatable customer-defined configurations.

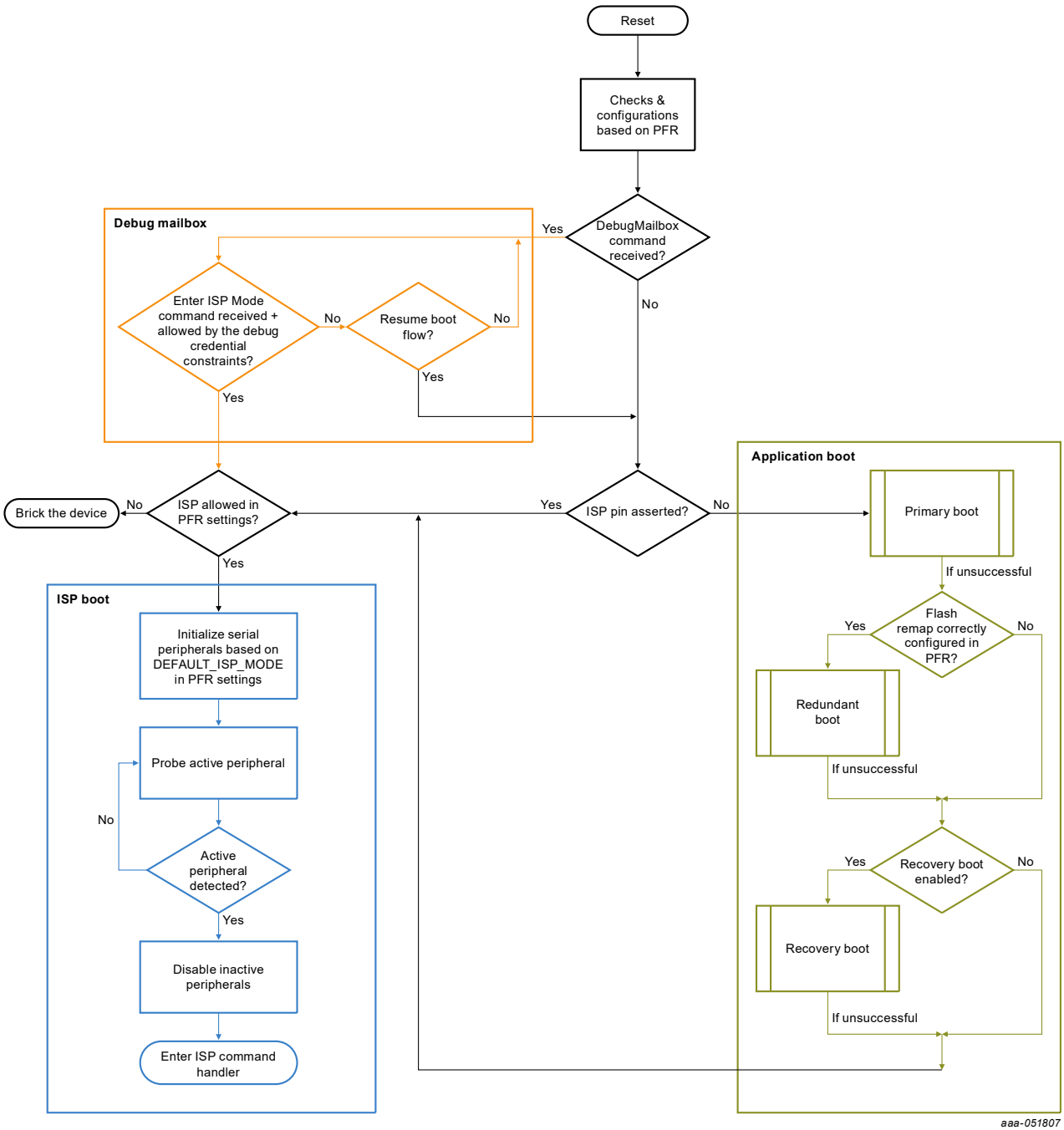


Figure 1. Illustration of the three main boot tracks (debug mode, ISP mode, and OEM application boot mode)

3 In-system programming (ISP) mode

The ISP mode is a bootloader mode that enables a customer to download application images, to set up platform configurations, and to inject or generate platform keys that are stored in a secure way.

The ISP mode can be reached in several ways:

1. When the reset signal is released and the ISP signal is asserted at the same time. This method is used to recover a part programmed with a corrupted image, which the ROM does not detect. This feature depends on the value of the DEFAULT_ISP_MODE bits in the BOOT_CFG word in CMPA.
2. The bootloader tries to boot a valid image but does not succeed, and falls through to the ISP mode. This feature also depends on the value of the DEFAULT_ISP_MODE bits in the BOOT_CFG word in CMPA.
3. The application code issues a ROM API call that resets the platform to ISP mode. The ISP mode can then be used to implement an in-field update.
4. The debug mailbox is used and a command to enter ISP mode has been issued there. This solution is used if the product does not have an ISP pin. It works with the SWD pin instead. It is not sensitive to the values of the DEFAULT_ISP_MODE bits. Nevertheless, the value of the debug configuration in CMPA and CFPA (fields starting with DCFG_CC_SOCU, at bit position 6, tagged as ISP_CMD_EN) conditions its availability.

Note: *Third and fourth methods are out-of-scope for this document.*

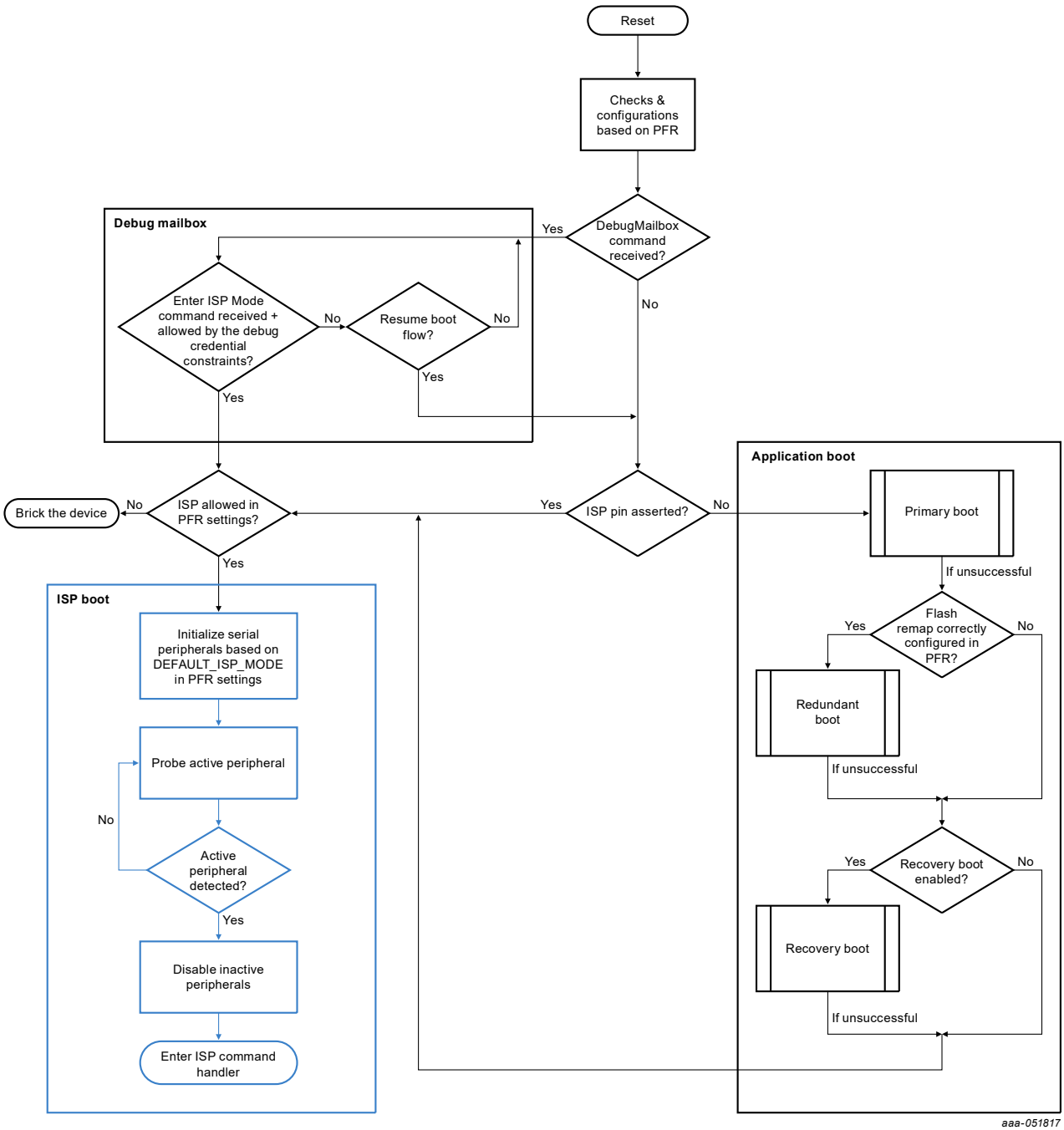


Figure 2. ISP mode flow

4 Application boot

In this section, the following is described:

- How the application images can be protected.
- Where the images can be located in memory (internal or external).
- The protected flash region (PFR) fields that impact the bootloader flow.

4.1 Ways of protecting the image on-chip

The MCX W23 platform supports booting:

- Unsigned images (sometimes also called "plain images")
- Unsigned images with a CRC checksum for a simple integrity check
- Images signed with RSA private keys for integrity and authenticity check

The former two are only used during the development phase, since they do not verify image authenticity. They enable the developer to take away some complexity while developing and debugging their application.

When the developed application is stable enough, use only signed images. For the signature to be checked, the secure boot functionality must be enabled in the SECURE_BOOT_CFG field of the CMPA (otherwise, the signature of a signed image is ignored).

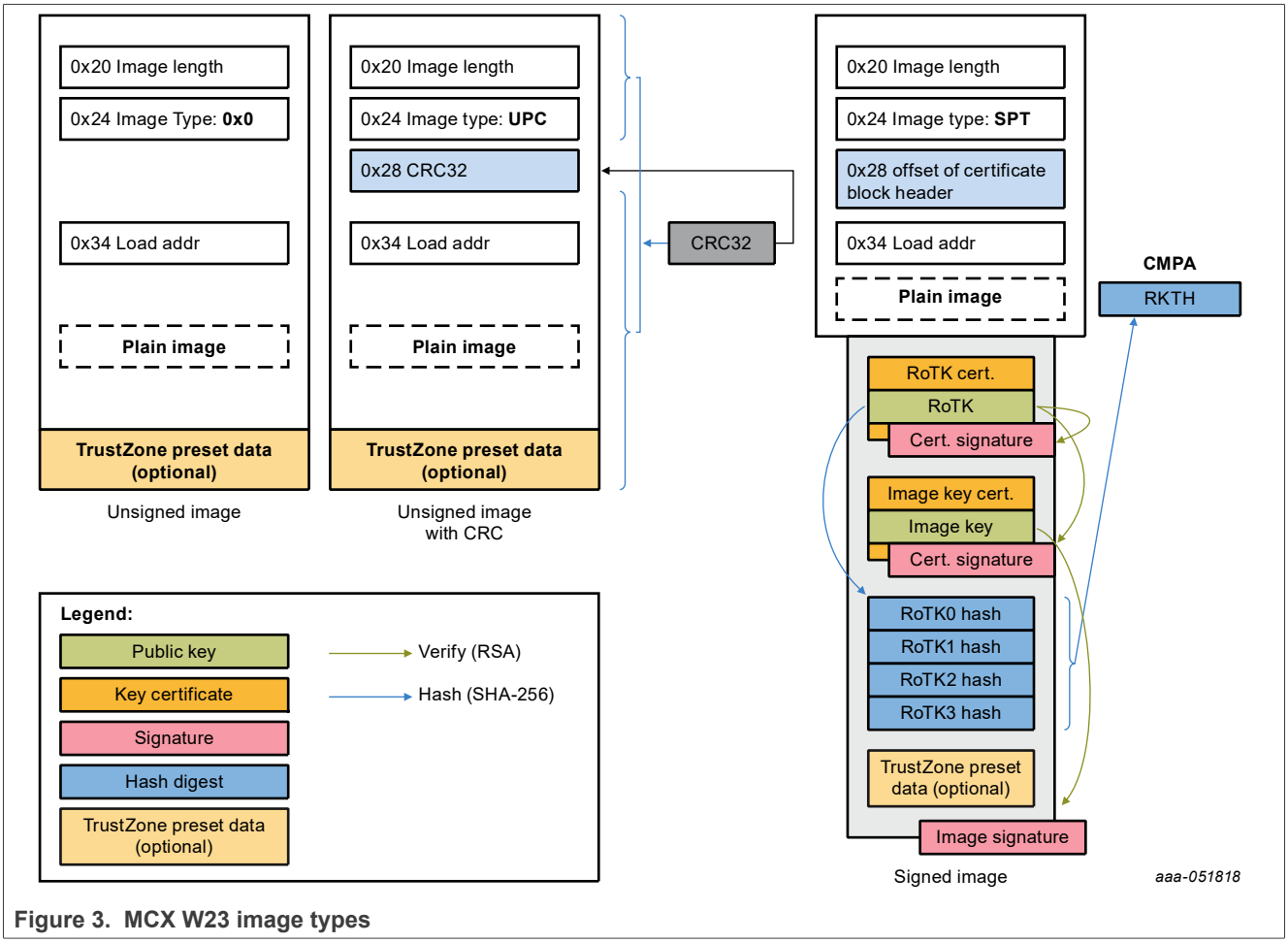


Figure 3. MCX W23 image types

4.1.1 Unsigned image

Unsigned binary images are the most basic type of image. Any compiler or IDE can generate them. It is the type of image that is sent to the platform when using the programming functionality of any IDE.

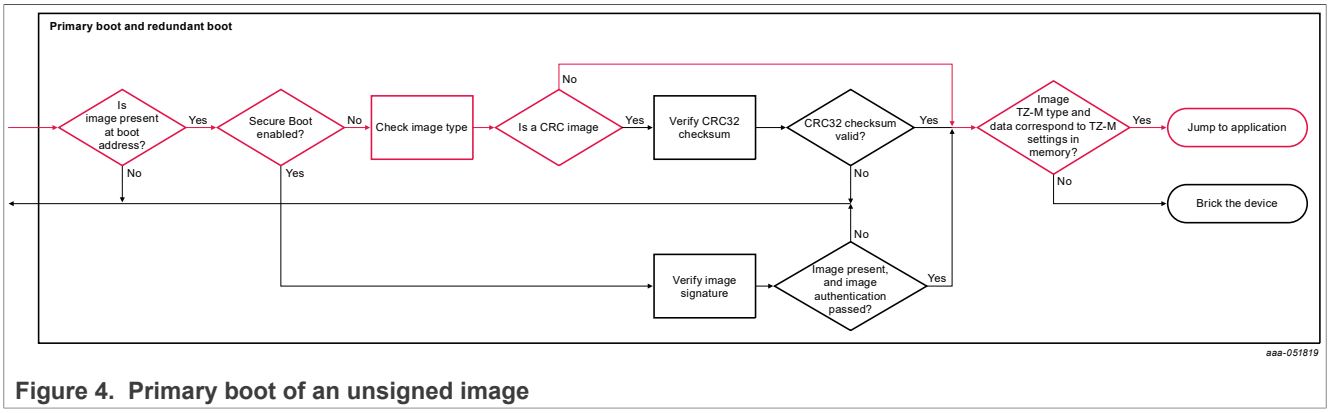


Figure 4. Primary boot of an unsigned image

4.1.2 Unsigned image with CRC

Unsigned images with CRC contain a CRC32 checksum value in their header, computed on the rest of the image (header included).

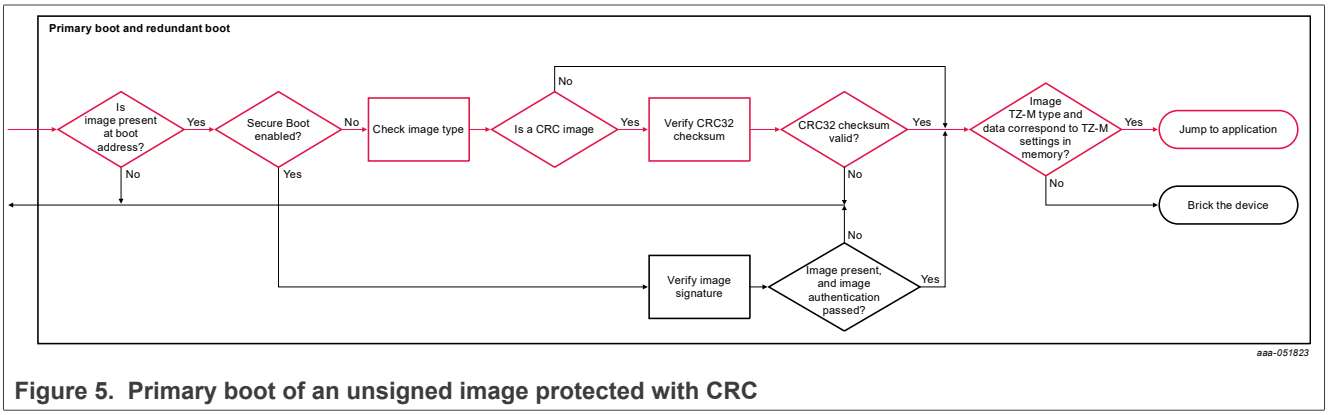


Figure 5. Primary boot of an unsigned image protected with CRC

4.1.3 Signed image

MCX W23 devices support booting of RSA signed. 2048-bit and 4096-bit RSA keys are supported[4] [5].

4 Uses the RSASSA-PKCS1-v1_5 scheme of the PKCS #1 standard. For more information, see RFC 3447.

5 The FIPS 140-3 security standard advises against 1024-bit RSA keys. The 3072-bit RSA key is also allowed. However, the debug authentication protocol of this platform does not support RoT keys of this size.

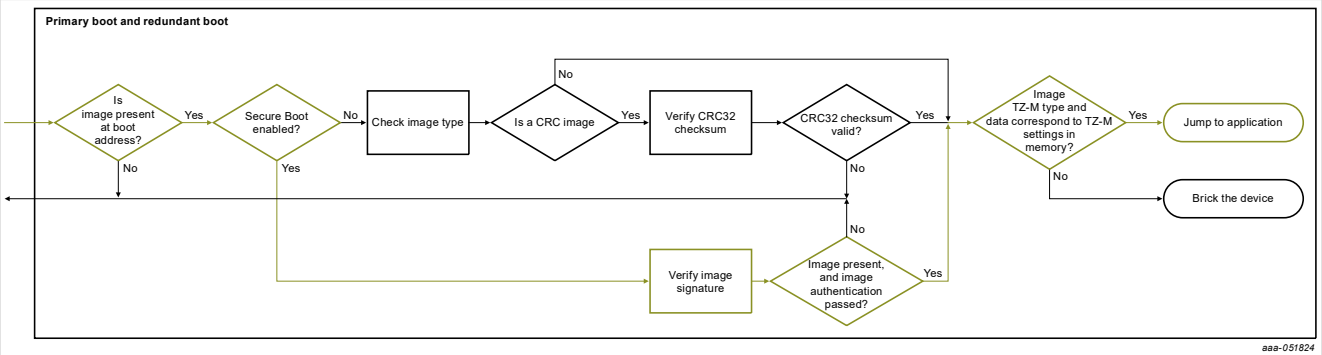


Figure 6. Primary boot of a signed image

4.1.3.1 Certificates and public-key infrastructure (PKI)

To verify the authenticity of the RSA keys used for signing, X.509 V3 certificates are used.

They follow the concept of public-key infrastructure (PKI). Every public key is bound to a specific identity of an entity or "subject" ("key-to-owner binding"), such as a person or an organization. This action happens using an (ownership) certificate that a certificate authority (CA, sometimes also called trusted third party) registers, issues, and manages throughout its lifetime. So, acceptance of the key-to-owner binding depends on the trust in the CA by the subject (owner) of the certificate and the party needing the certificate for signature verification.

This PKI is the principle behind the "chain of trust" used for signed image authentication. For more information, see *LPC55Sxx Debug Authentication* (document AN13037).[6]

Each X.509 V3 certificate contains:

- Public key of the subject (which is used to verify a signature generated using the corresponding private key of the subject)
- Signature algorithm used
- Issuer and subject names
- Validity period
- Serial Number field (which contains a revocation identifier compared against the image revocation field in CFPA; see Section 5.1).

It is the minimal certificate profile. Additional information on the issuer and subject is optional. If the user chooses to generate their certificates with non-NXP tools, such as OpenSSL, the additional information can be added as an "extension".

4.1.3.2 Image validation

Image validation for a signed image is a two-step process (establishing the trust in the public key and then using that public key to verify the signature of that image):

1. The first step is the validation of the chain of X.509 V3 certificates inserted in the image (in DER format). This example presents a chain of two certificates, the Root-of-Trust certificate and the image certificate. This chain is the PKI, which is then linked via a hashing strategy to a trusted value stored safely on the platform.
   a. The signature of the image public key certificate is verified against the RoT public key that is present in the RoT certificate.
   b. The signature of the RoT public key certificate is verified against the RoT public key that is present in the RoT certificate (since it is self-signed).

6 This document is an example from a similar platform, as there is no specific document yet for this product.

c. The RoT public key is hashed (SHA-256) and its hash digest must match with one of the four[7] RoT key hashes present in the image header (called as the RoT key hash table).
d. Finally, the hash (SHA-256) of the RoT key hash table must match the RoT key hash table hash (abbreviated RKTH) that has been defined in CMPA (see Figure 11).

So, a "chain of trust" is established between the RKTH value stored in CMPA, which is the trusted value on the platform, and the image certificate, which is the last certificate in the chain.

2. The image public key is used in a second step to validate the signature on the entire image (including the certificates).

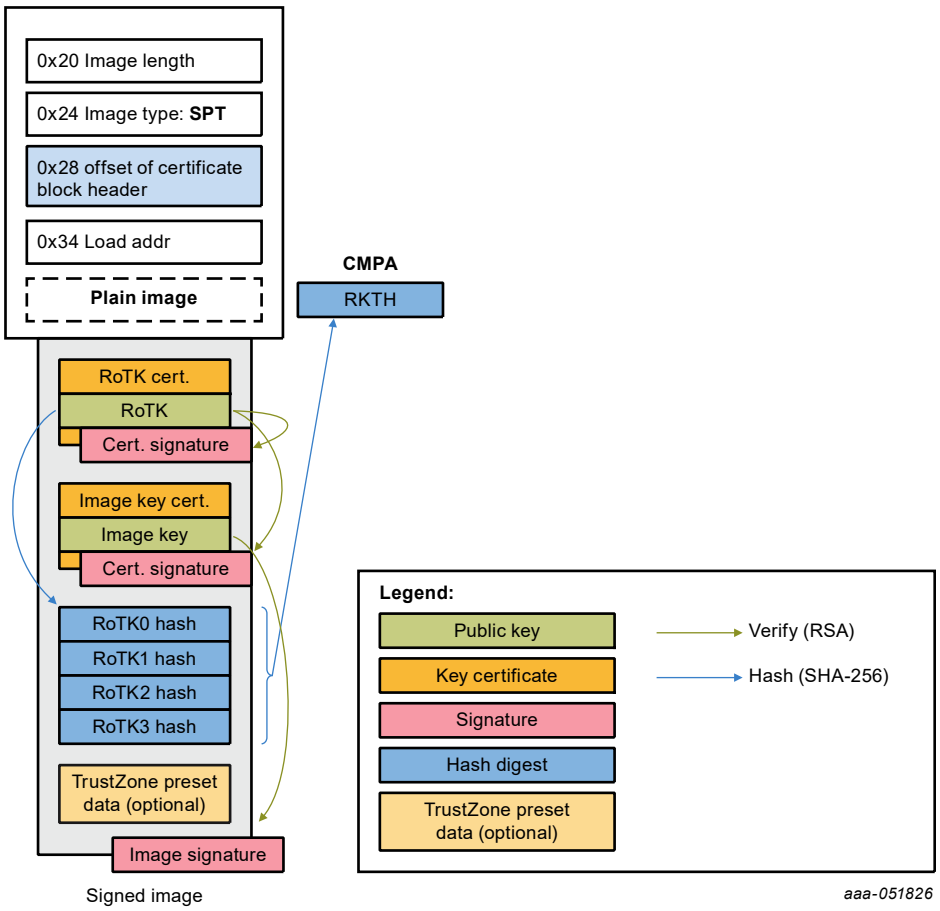


Figure 7. Signed image

The validation of the image is only enforced when the SEC_BOOT_EN bits are enabled in the SECURE_BOOT_CFG field of the CMPA. Otherwise, the signature of a signed image is ignored.

The RSA4K bits in SECURE_BOOT_CFG in CMPA determine if only 4096-bit RSA keys can be used or not. With the default settings, the platform accepts 2048-bit and 4096-bit RSA keys. [8]

7 The platform can accept up to four RoT keys. If one RoT key has been compromised, the platform remains usable with the other RoT keys. For more information, see Section 5.1.

8 The FIPS does not recommend using RSA keys shorter than 3072 bits beyond 2030. So, use only 4096-bit keys to set the RSA4K bits to 0b01. However, when there are intermediate certificates in the chain of trust between the RoT key certificate and the image key certificate, using the 4096-bit keys is impossible. In this case, the developer must use 2048-bit RSA keys for each of the keys involved in the chain.

4.2 Three application boot modes

The bootloader can boot images from three locations in memory. So, it can make up to three boot attempts.

The bootloader tries the following boot attempts sequentially:

1. Primary boot
2. Redundant boot, if configured in CMPA
3. Recovery boot, if an external SPI NOR flash is connected to the platform and the option is enabled in CMPA

4.2.1 Primary boot

By default, the processor automatically boots images that are placed at location 0x0 in internal flash memory, which is called primary boot.

If secure boot was enabled, the signature on this primary image is verified.

If no valid image is found at address 0x0, the bootloader proceeds with another boot option.

This option is illustrated in Section 4.1 for each image protection level.

4.2.2 Redundant boot

If the fields FLASH_REMAP_OFFSET and FLASH_REMAP_SIZE are both filled with non-zero integer values respecting the conditions explained in Section 4.3.1.4, the redundant boot ("flash remap") feature can be used to boot an image at another location. If conditions are not met, the part simply proceeds with a recovery boot or fall-through.

If secure boot was enabled, the signature on this fallback image is verified too.

Section 4.1 illustrates this option for each image protection level.

4.2.3 Recovery boot

If the three conditions mentioned below are met, an image on the external memory can be booted (see Figure 8):

- Booting from the main flash (primary boot and redundant boot) was not successful.
- A serial NOR flash memory is coupled to the MCX W23 device.
- The SPI_RECOVERY_BOOT_EN setting is enabled in CMPA.

In recovery boot, the bootloader copies the application to RAM and executes it there.

If secure boot was enforced, the image must be wrapped in a valid SB2.1 container (a container format that NXP has developed, wrapping the signed image).

Figure 8 shows the recovery application boot for a signed image.

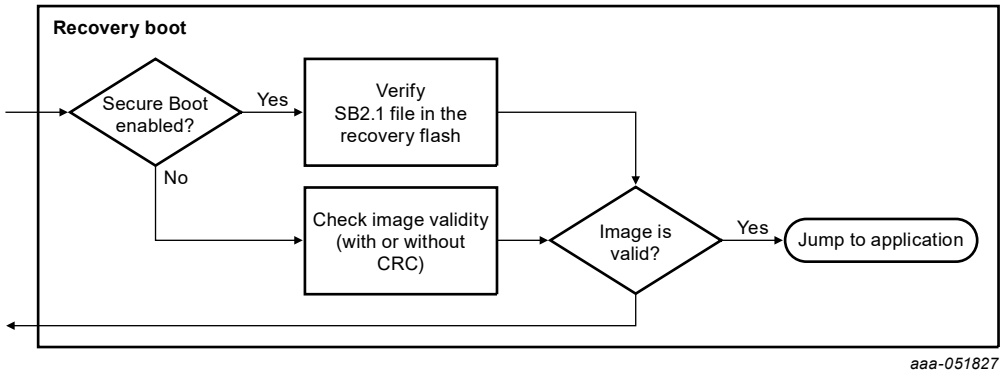


Figure 8. Recovery boot

4.2.4 Fall-through

Finally, if none of these options works, the platform enters the ISP mode if the CMPA settings are set to allow it. Otherwise, the part bricks.

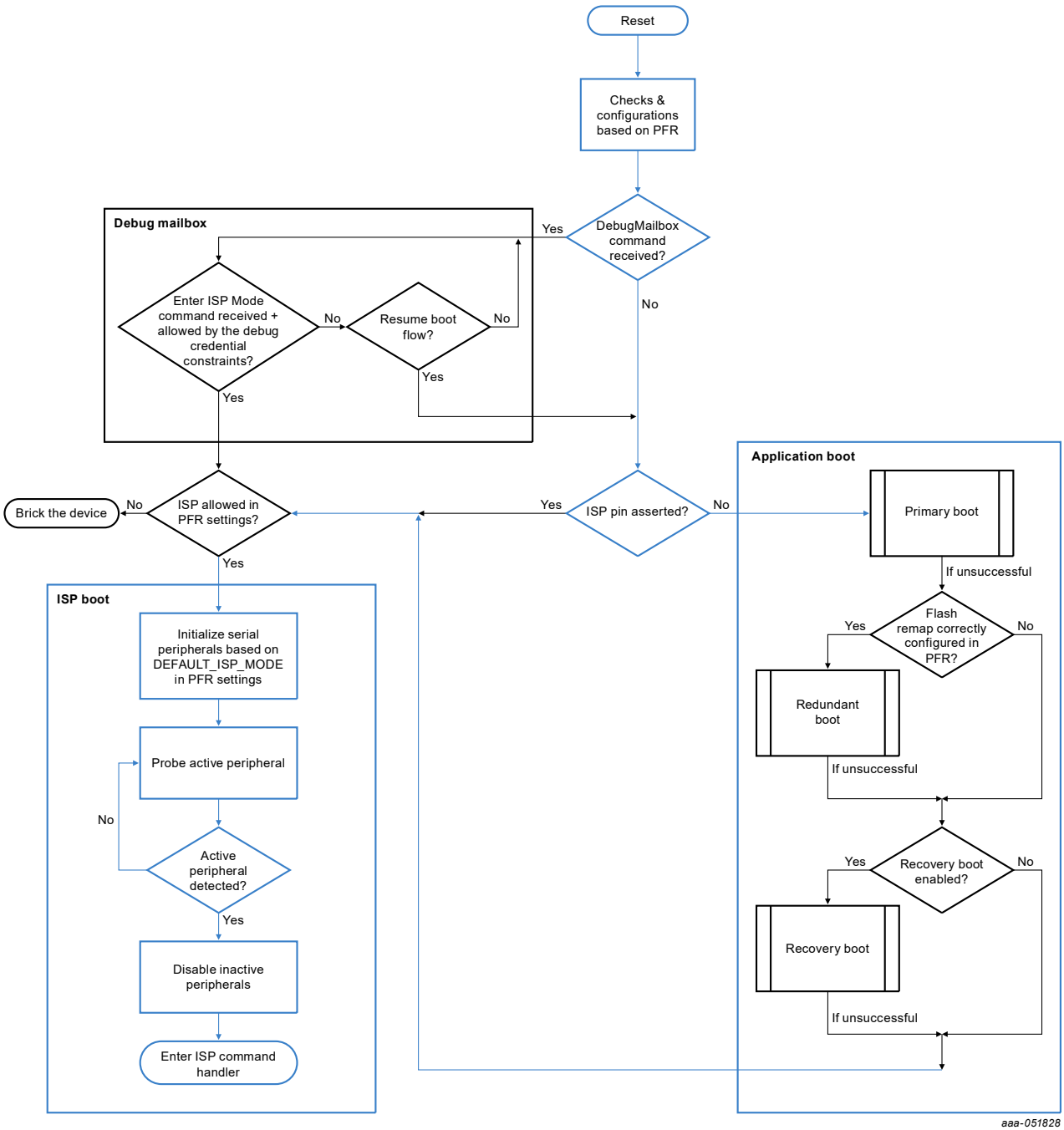


Figure 9. Fall-through to ISP mode

4.3 Protected flash region (PFR)

The protected flash region (PFR) is a special flash memory region to which access is protected. It stores configuration options for the bootloader and platform operation.

The PFR is made up of different parts:

- CFPA (three pages):
  - CFPA scratch
  - CFPA ping
  - CFPA pong
- CMPA
- Key store area (KSA)

In this application note, only the fields that have an impact on the bootloader actions are considered.

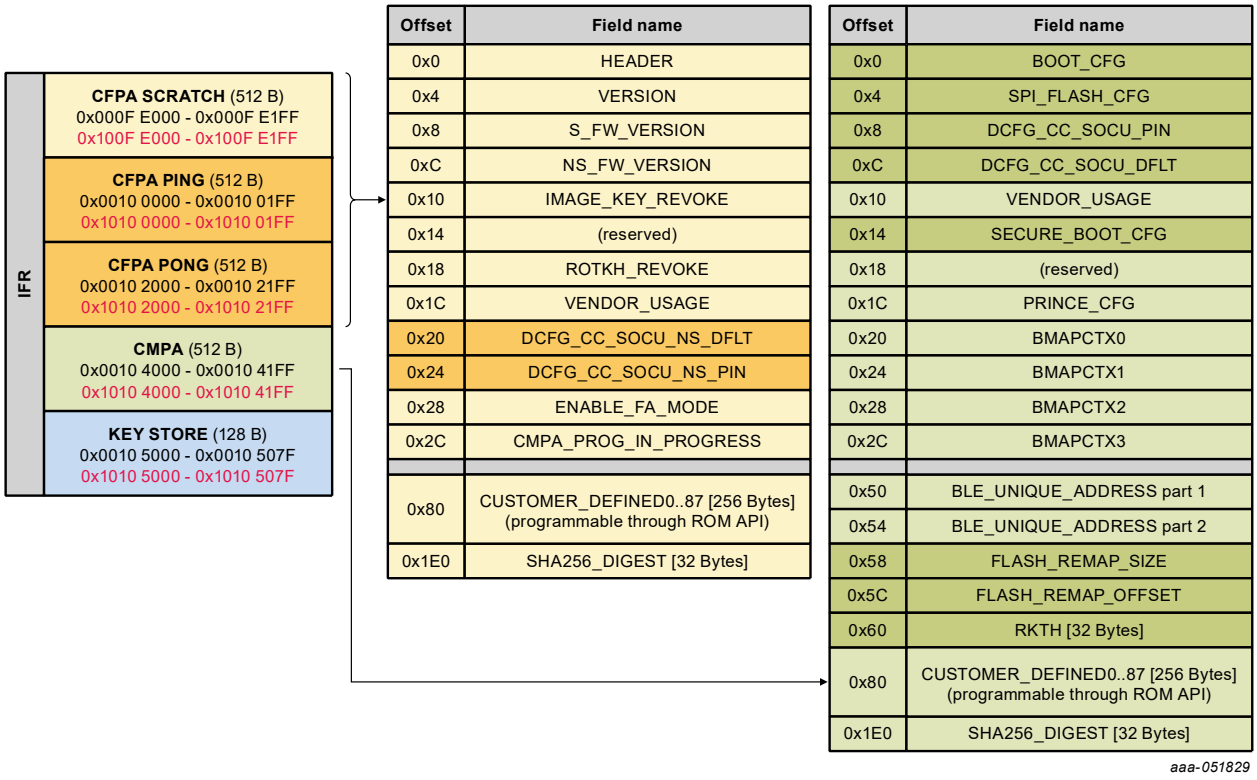


Figure 10. PFR sections impacting the bootloader

4.3.1 CMPA

The customer manufacturing programmable area (CMPA) contains settings that the customer provides.

As long as the platform must remain in development mode, the SHA-256 digest field at the end of the page must remain blank. When switching to deployment mode, a non-zero value must be written to this SHA-256 digest field. The platform replaces this value by the real SHA-256 digest value computed on the CMPA page. From this moment on, the CMPA page and the key store area (KSA) of the customer can no longer be modified.

4.3.1.1 BOOT_CFG

The DEFAULT_ISP_MODE bits, at bits [6:4] in BOOT_CFG, define which communication protocols are allowed or expected in ISP mode.

Table 1. ISP download mode based on DEFAULT_ISP_MODE bits (BOOT_CFG[6:4] in CMPA)

ISP boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Description
Auto ISP	0	0	0	The MCX W23 detects the active peripheral from one of the serial interfaces below: • UART2 • I2C1 • SPI0
Disable ISP	0	0	1	disable ISP mode
UART ISP	0	1	0	UART is used to download image
SPI slave ISP	0	1	1	I^2C slave is used to download image
Disable ISP	1	0	1	disable ISP mode
Disable ISP	1	1	0	disable ISP mode
Disable ISP	1	1	1	disable ISP mode

4.3.1.2 SPI_FLASH_CFG

If a value other than zero is written to bits [4:0] of SPI_RECOVERY_BOOT_EN, using the external flash device is considered in the boot flow.

4.3.1.3 SECURE_BOOT_CFG

The bit fields RSA4K and SEC_BOOT_EN are the most important fields for the boot flow in SECURE_BOOT_CFG. Other bit fields are reserved or irrelevant for this application note. They are not shown below.

Table 2. RSA4K bits (SECURE_BOOT_CFG[1:0] in CMPA)

SECURE_BOOT_CFG[1]	SECURE_BOOT_CFG[0]	Description
0	0	Allow RSA2048, RSA3072, and RSA4096 keys
0	1	Allow RSA4096 keys only
1	0	
1	1	

Table 3. SEC_BOOT_EN bits (SECURE_BOOT_CFG[31:30] in CMPA)

SECURE_BOOT_CFG[31]	SECURE_BOOT_CFG[30]	Description
0	0	Check on signature and chain of trust is not enforced. Booting an unsigned image from an internal or external flash is possible. Signed images are booted too without checking the signature and chain of trust.
0	1	Check on signature and chain of trust is enforced. Booting only RSA-signed images in internal flash and SB capsules in external flash.
1	0	
1	1	

4.3.1.4 FLASH_REMAP_SIZE and FLASH_REMAP_OFFSET

These fields enable the use of redundant boot, as explained in Section 4.2.2. They define the location and size of a redundant boot image. This image is booted if no valid image is present at address 0 of the internal flash.

In the FLASH_REMAP_SIZE field, any multiple of 0x1000 can be entered, since the flash granularity is 4 kB. This FLASH_REMAP_SIZE value must be greater than the size of the image put at the remap location.

In the FLASH_REMAP_OFFSET field, any multiple of 0x1000 can be entered. And this condition must be met:

FLASH_REMAP_OFFSET ≥ FLASH_REMAP_SIZE

4.3.1.5 DCFG_CC_SOCU_PIN and DCFG_CC_SOCU_DFLT

The only bit in both fields important for the boot flow is bit 6, ISP_CMD_EN. Other DCFG_CC_SOCU. [9]For more information, see *LPC55Sxx Debug Authentication* (Reference 4[10]).

Table 4. DCFG_CC_SOCU fields for ISP_CMD_EN

DCFG_CC_SOCU_PIN[6]	DCFG_CC_SOCU_DFLT[6]	Result
1	0	This setting offers the highest level of restriction. Entering ISP mode over debug is never possible.
0	0	This setting allows the debug authentication process to define access rights to enter ISP mode.
0	1	Illegal setting. If this setting is selected, the part may brick.
1	1	This setting is the lowest security level. Entering ISP mode over debug is always possible.

The command that can be sent to the debug mailbox to enter ISP mode is Enter ISP mode (Table 5).

9 DCFG_CC_SOCU stands for "Device Configuration Credential Constraints for System-on-Chip Unit"

10 This document is an example from a similar platform, as there is no specific document yet for this product.

Table 5. DM-AP commands

Name	Command code	Parameters	Response	Description
Enter ISP mode	0x05	dataWordCount: 0x1 • data[0]: ISP mode enum – 0xFFFF_FFFF - auto detection – 0x1 - UART – 0x2 - I^2C – 0x4 - SPI – Others - reserved	32-bit status	Enter the specified ISP mode By default, the state of the ISP boot selection pins at reset time determines ISP mode entry. Before field deployment, this functionality is disabled via the CMPA configuration. This command can be used to enter ISP mode in those situations or when ISP boot selection pins are used for some other board function. Support of this command can be restricted through the DCFG_CC_SOCU(_NS) fields in the PFR, as described in the LPC55Sxx debug authentication application note (Reference 4).

4.3.1.6 Root key table hash (RKTH)

The value stored in this field is the foundation of the "chain of trust" within the device, to which the PKI is attached. It is used for image verification at start-up and for verification of SB containers.

RKTH is a 32-byte SHA-256 hash of SHA-256 hashes of up to four RoT public keys. Multiple RoT public keys are supported, such that RoT key revocation is possible (see Section 5.1).

Figure 11 shows the structure of this table.

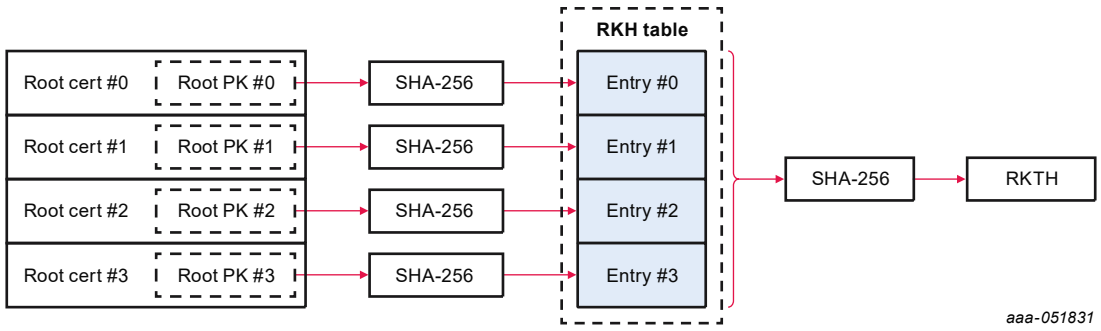


Figure 11. RKTH generation process

The number of hash digests of keys in the RKH table must be from at least 1 up to a maximum of 4. Unused table entries must be set to all-0 bytes.

4.3.2 CFPA

The customer in-field programmable area (CFPA) contains settings that the customer provides, which, after the product has been deployed to end users, can still be updated.

4.3.2.1 DCFG_CC_SOCU_NS_PIN and DCFG_CC_SOCU_NS_DFLT

The debug settings in CFPA shown below can only close the part further. They never weaken the settings established in CMPA.

For configuration of the bits in DCFG_CC_SOCU_NS_PIN and DCFG_CC_SOCU_NS_DFLT, see Section 4.3.1.5.

4.3.3 KSA

The key store area in CMPA contains two keys on this platform:

- SBKEK
- USERKEK

Only the SBKEK is of interest in the scope of this document.

4.3.3.1 SBKEK

To load an image to a device that is in the deployment state, it must be wrapped in an NXP-designed container that is encrypted and signed. This container is called the SB2.1 container.

The SB key encryption key (SBKEK) is the AES-256 secret key that is required to decrypt an encrypted SB2.1 container, so the signed application image stored within it can be obtained.

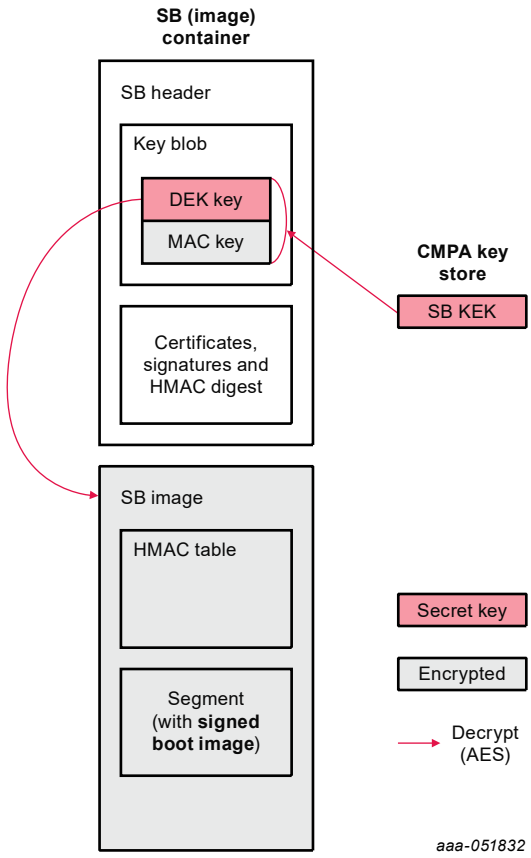


Figure 12. SB2.1 container decryption by SBKEK (arrows)

If the use of recovery boot is desired while secure boot is enforced, the SBKEK is required.

Before storing the SBKEK in the KSA, the physical unclonable function (PUF) must wrap it. The PUF is a peripheral that enables to store keys safely in an unencrypted part of memory, without the requirement for a hidden root encryption key on the device. The key protection is derived from the physical characteristics of the device, such that the keys can only be retrieved on the device where they were "wrapped". The result of tampering the device is that the keys become unretrievable. For more information, see the LPC55Sxx usage of the PUF and Hash Crypt to AES coding application note (Reference 5).

When the SEC tool is used, the wrapping and storing happen together and transparently.

5 Additional security mechanisms

The security mechanisms described next can be used to increase the security of the device. Security is about layers of protection in hardware and software. They can be used independently of the secure boot feature. However, they only really make sense when combined with a secure boot.

- Key revocation prevents that the platform boots application images signed with deprecated keys.
- Using PRINCE encryption on internal flash memory addresses the requirement for confidentiality of the contents stored there.
- Designing the application image using TrustZone-M allows the isolation of critical software and sensitive information from other parts of software and the threats that can come with them. Because of the temporal isolation that TrustZone-M offers, a single CPU can be used instead of two parallel CPUs. So, it reduces the attack surface of critical components.

Figure 13 shows where these additional security settings are located in PFR.

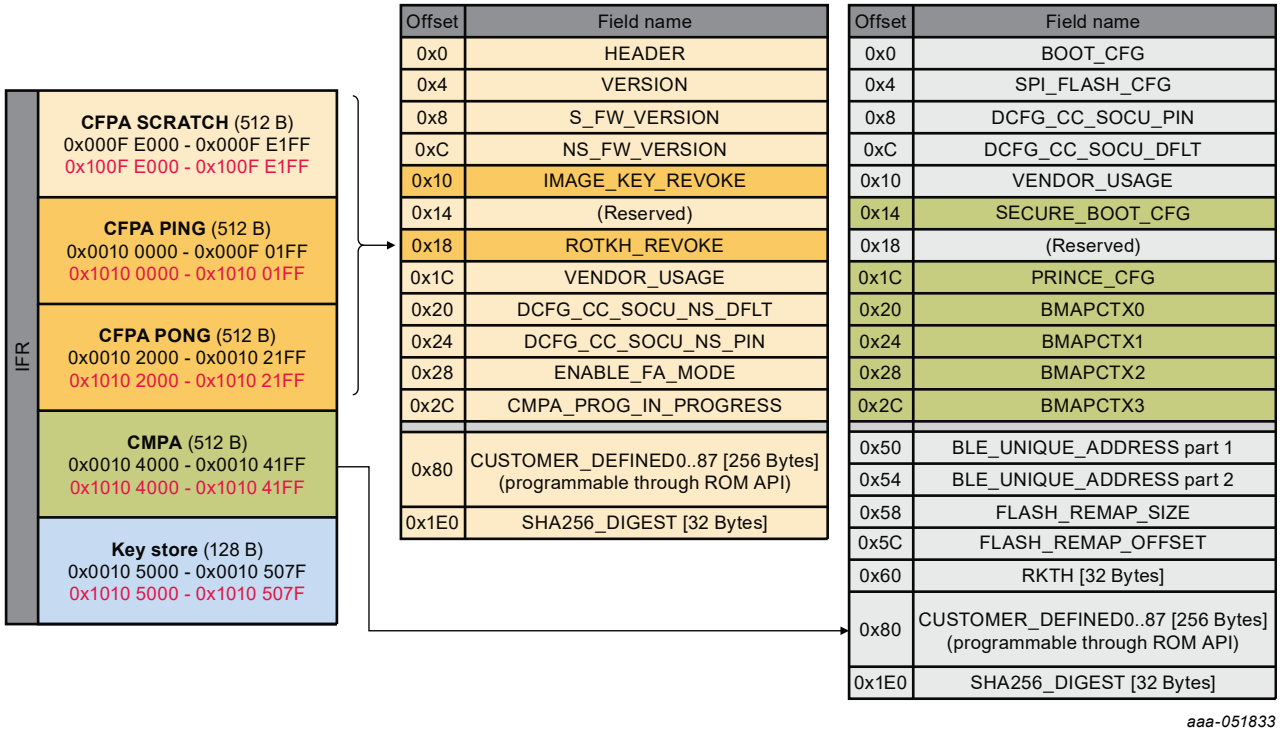


Figure 13. Additional security settings in PFR

5.1 Key revocation

The MCX W23 supports four root-of-trust (RoT) keys (see section 4.19 of SEC_PRIM; Reference 1) and up to 16 image keys over its lifetime.

The platform allows up to 4 RoT keys. Any of the allowed RoT keys can be used in the certificate chain of an application image. However, as long as it has not been compromised, it is recommended to stick to the use of only one of them.

On the other hand, the platform allows only one image key at a particular point in time. A change in image key is irreversible. Starting to use a new image key means that the previous image key cannot be used anymore. The previous image key is revoked. Its public key is flagged to become invalid for image validation. This action is necessary to prevent rollback to image keys whose private key has been compromised.

This implementation of revocation is a lightweight alternative to certificate revocation lists (CRLs), which are the common way to revoke certificates, used in the X.509 standard.

5.1.1 On the platform

The bits indicating which RoT keys are allowed and which have been revoked are in the ROTKH_REVOKE field in CFPA. Revoking a RoT key is irreversible.

Table 6. ROTKH_REVOKE table bit field description

Bits	Name	Description
1:0	RoTK0_EN	RoT key 0 enable 2'b00: invalid 2'b01: RoT Key 0 is enabled 2'b10: RoT Key 0 is revoked 2'b11: RoT Key 0 is revoked
3:2	RoTK1_EN	RoT key 1 enable 2'b00: invalid 2'b01: RoT Key 1 is enabled 2'b10: RoT Key 1 is revoked 2'b11: RoT Key 1 is revoked
5:4	RoTK2_EN	RoT key 2 enable 2'b00: invalid 2'b01: RoT Key 2 is enabled 2'b10: RoT Key 2 is revoked 2'b11: RoT Key 2 is revoked
7:6	RoTK3_EN	RoT key 3 enable 2'b00: invalid 2'b01: RoT Key 3 is enabled 2'b10: RoT Key 3 is revoked 2'b11: RoT Key 3 is revoked
31:8	reserved	reserved To be filled in with zeros

The 16 bits of the IMAGE_KEY_REVOKE field in CFPA indicate how many image keys have been revoked. Each bit represents a possible image key. The bits must be enabled from the right to the left, like a monotonic bit counter, such that only 17 revocation IDs are possible: 0x0, 0x1, 0x3, 0x7, 0xF, 0x1F, 0x3F, 0x7F, 0xFF, and so on, up to 0xFFFF. When updating the CFPA, the value of the IMAGE_KEY_REVOKE field cannot decrease (protection against rollback, see section 4.22 of SEC_PRIM; Reference 1).

5.1.2 In the certificate chain of the image

Each image contains a reference to its image key version. Which image key version is used can be found in the serial number field of the certificate in the image (which the signature of the image protects).

The serial number field has a length between 4 bytes and 20 bytes and looks like this:

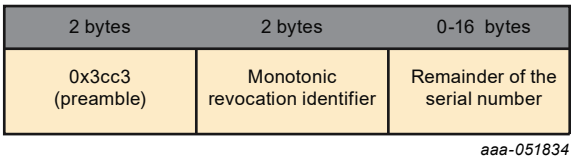


Figure 14. Serial number structure

To allow an image to be booted, the image key version in the image certificate must be equal to the one in CFPA. Or to be the next monotonic value (a so-called roll-forward; allowed to avoid bricking the device if power loss happens after the FW image has been updated and before the field of the CFPA is updated).

For instance, if the CFPA indicates 0x3FF, certificates with image key version 0x3FF or 0x7FF are accepted, while certificates with other image key versions are rejected.

5.2 PRINCE encryption on internal memory

In addition to the protection of the integrity of the stored image with a CRC or its integrity and authenticity with a cryptographic signature, the image confidentiality can be protected against physical attacks by activating on-the-fly PRINCE encryption on the internal flash region where the image is stored. It allows efficient protection of intellectual property without any delay cost (see section 4.15 of SEC_PRIM; Reference 1).

The MCX W23 supports 4 PRINCE regions, such that different images can coexist in internal memory without sharing the encryption domain. Each PRINCE region has a secret key, which is supplied from the on-chip SRAM PUF via a secret bus interface (not accessible to SW).

The PRINCE settings that define the encrypted areas can be locked using the GLK sticky bit, at position 13 in the PRINCE_CFG word in CMPA.

5.3 TrustZone-M for core process separation

The MCX W23 looks at the TrustZone-M setting in CMPA to know what type of image it must accept, with or without the TrustZone-M configuration. If the CMPA setting does not conflict with the image type indicated in the image header, the image boots. Otherwise, the part bricks (fatal mode).

A TrustZone preconfiguration can be added at the end of the image, such that it is applied before jumping to the image, while the TrustZone configuration compiled within the image is applied after having jumped to the image.

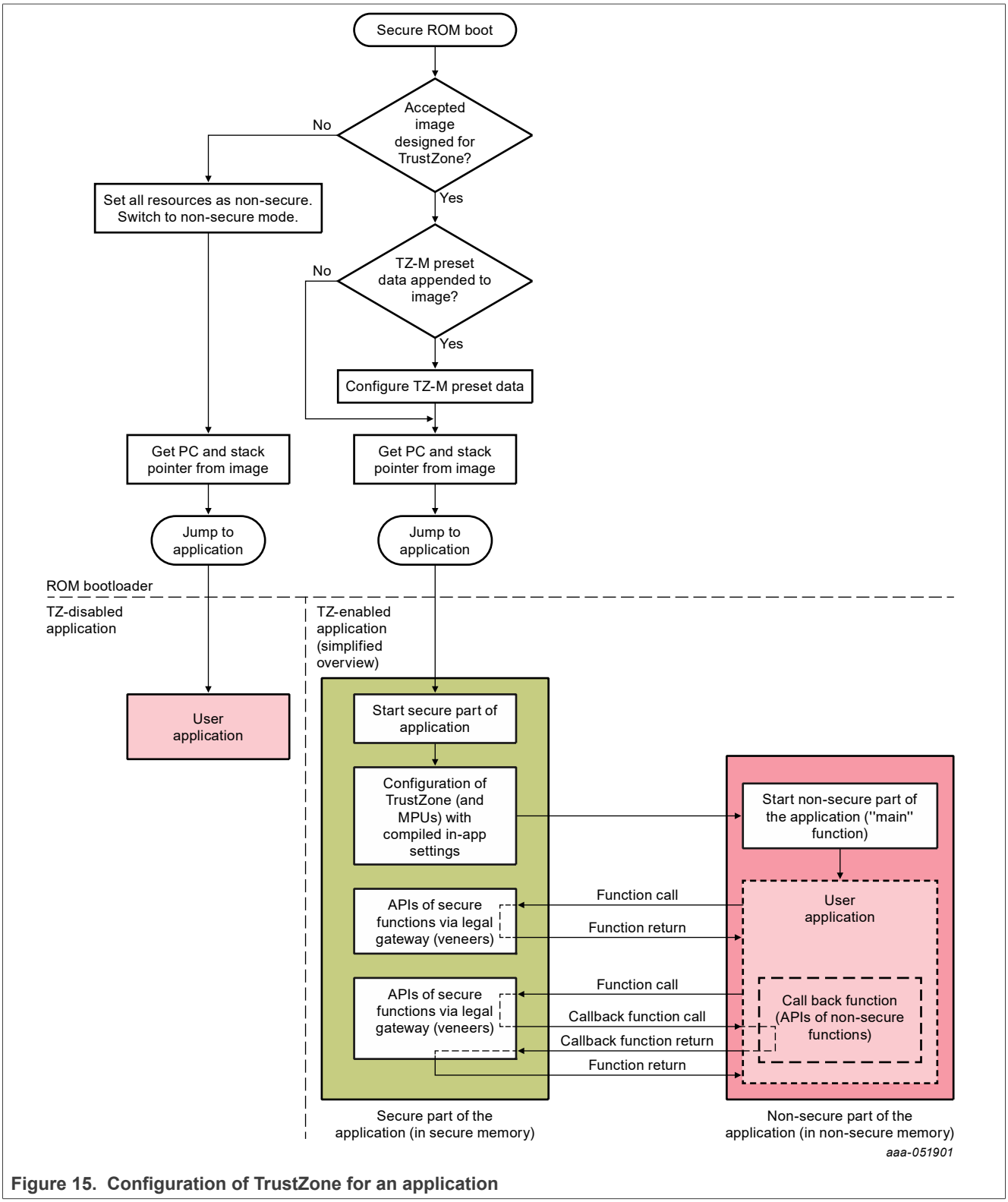


Figure 15. Configuration of TrustZone for an application

Table 7. TZM_IMAGE_TYPE bits (SECURE_BOOT[9:8] in CMPA)

TZM_IMAGE_TYPE[9]	TZM_IMAGE_TYPE[8]	Description
0	0	The TZ-M mode of the application image is taken from its header. How the CPU acts depend on it. See the other TZM_IMAGE_TYPE bit combinations.
0	1	An application image with TZ-M disabled is required (consists of only one part). The CPU configures minimal trusted execution environment (TEE) settings and jumps to the application in the non-secure state.
1	0	An application image with TZ-M enabled is required[1]. The CPU jumps to the secure part of the application in a secure state. It configures the TEE settings compiled with the application in that state.[2]
1	1	An application image with TZ-M enabled[1] and with TZ-M preset data appended to it is required. Before the CPU jumps to the secure part of the application in secure state, the CPU preconfigures the TEE using this TZ-M preset data. After the jump, the CPU configures the additional TEE settings compiled with the application. These in-app TEE settings can only make the isolation requirements stronger. So, weaker in-app TEE settings are ignored.

[1] Consists of two parts, "secure" and "non-secure".
[2] In TZ-M, the "secure" part must be understood as the part where the "trusted code" executes security-critical actions and where security-sensitive information is processed and stored. Here, "secure" does not point to protection using cryptography but to protection using flow and context isolation in hardware.

The TrustZone-M and MPU settings can be easily configured using the TEE tool in the MCU Config Tools program of NXP, available at www.nxp.com.

Note: *TrustZone-M is the architecture that Arm has developed. TEE is the trustworthy environment ("Secure World") that the methodology of this architecture creates. The TEE is separated from the normal environment, which is deemed untrustworthy. Even if the CPU is not involved (for example, if a DMA transfer occurs), the TEE can encompass interactions involving secure masters and secure slaves (memory or peripherals) on the AHB bus. In that case, it is not the TrustZone-M technology in and around the processor (the so-called "attribution units") that manages them, but a system security controller and security gates on the bus.*

6 Configuring boot on MCX W23 using SEC tool

The SEC tool is a user-friendly interface designed to configure the platform security settings correctly and to create a valid signed image to be loaded to the platform. It has been built on top of the SPSDK, a Python library designed for scripting the whole configuration process. The SEC tool outputs scripts using SPSDK commands to ensure that the whole process can be easily integrated into a manufacturing flow.

For more information, see the following webpages:

- MCUXpresso Secure Provisioning Tool: https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-secure-provisioning-tool:MCUXPRESSO-SECURE-PROVISIONING
- Secure Provisioning SDK (SPSDK): https://www.nxp.com/design/design-center/software/development-software/secure-provisioning-sdk-spsdk:SPSDK

6.1 Assumptions and prerequisites

6.1.1 Assumptions

In this section, we assume that:

- There is a certain familiarity with building binaries for the targeted platform (MCX W23).
- The previous concepts of this document itself are clear.
- There is a familiarity with the currently available evaluation board and its jumpers/buttons.

6.1.2 Prerequisites

Ensure that you meet the following prerequisites:

- The unprotected binary is developed in a toolchain supported by the SDK.
- The SEC tool (version 25.03 and up) is installed. To ensure that potential bugs are patched, always use the latest version of the tool.

For more information, refer to the following MCUXPRESSO documents:

- Secure provisioning tool
- User Guide (document MCUXSPTUG)

6.2 Setting up the workspace

The properties of a workspace within the SEC tool are different from the properties of a workspace in MCUXpresso. For example, the SEC tool workspace acts more as a project. So, the workspace is processor-specific and not generic. It means that once the user creates a workspace for a specific device, a certain set of features/options are available, which could differ from processor to processor.

6.2.1 Creating a workspace in the SEC tool

Start by opening the sec tool. Depending on if the tool was used previously or was newly installed, the process is different.

- Previously used:
  A workspace, which has been generated previously, opens.
  – On the top left, click "File" > "New Workspace …" once
- Newly installed:
  A window pops up, prompting the user to create a new workspace (see Figure 16).

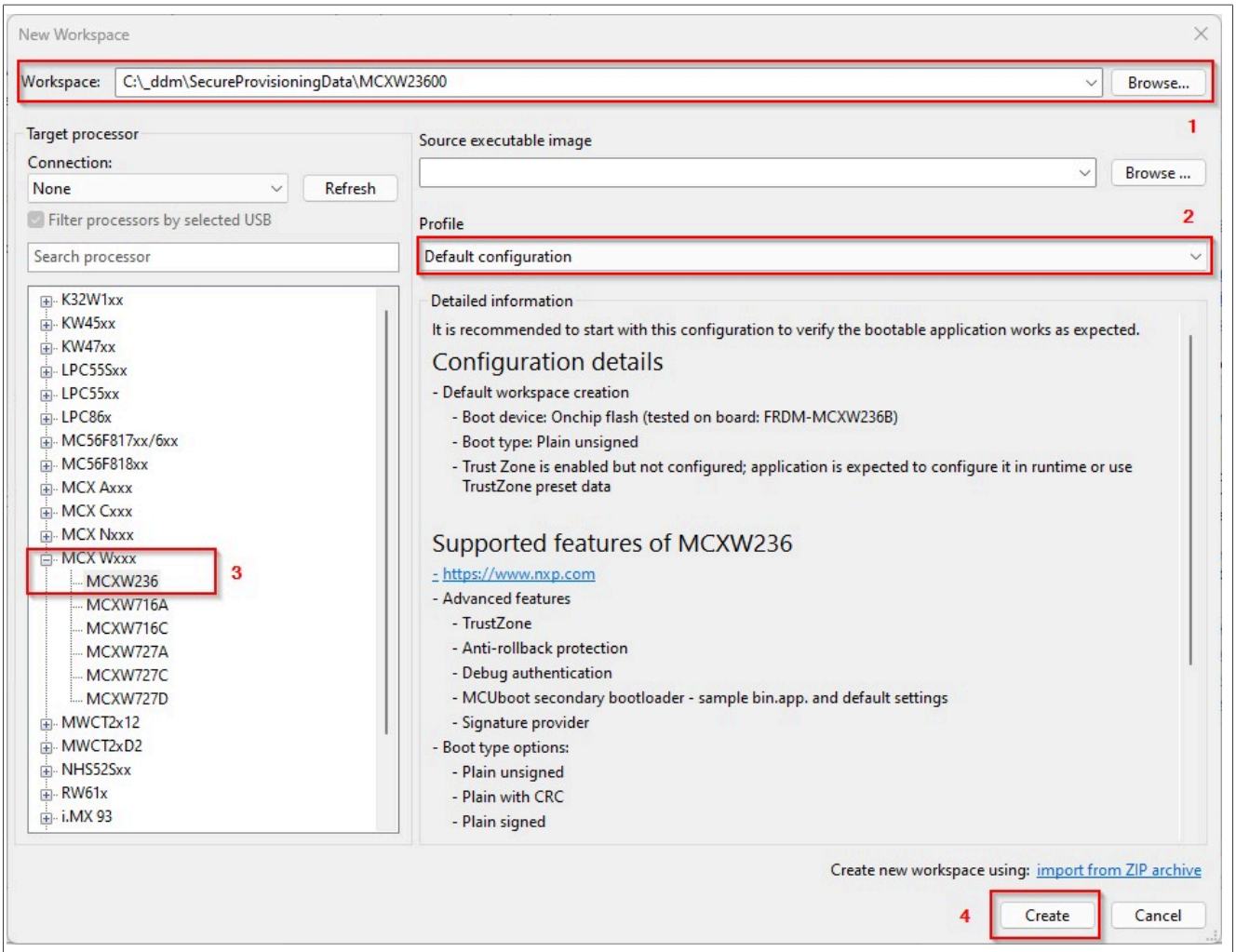


Figure 16. SEC tool - creating a new workspace

1. Click **Browse** to select the appropriate location to store the workspace.
2. Select **Default Configuration** as Profile.
3. Select **MCX Wxxx/MCXW236** from the list.
4. Click the **Create** button to generate the workspace and its artifacts.

For more information, see the doucment "MCUXpresso Secure Provisioning Tool User Guide" (Ref. 3).

6.3 Device connection

On the FRDM-MCX W236B board, the default jumper configuration sets the UART to FLEXCOMM 0. To enter ISP mode, UART must be changed to FLEXCOMM 2.

6.3.1 ISP mode on the EVK board

To change the UART from Flexcomm 0 to Flexcomm 2 on the FRDM-MCX W236B, move the installed J13 and J14 jumpers from 1-2 towards 2-3 jumpers over to the right (see Figure 17).

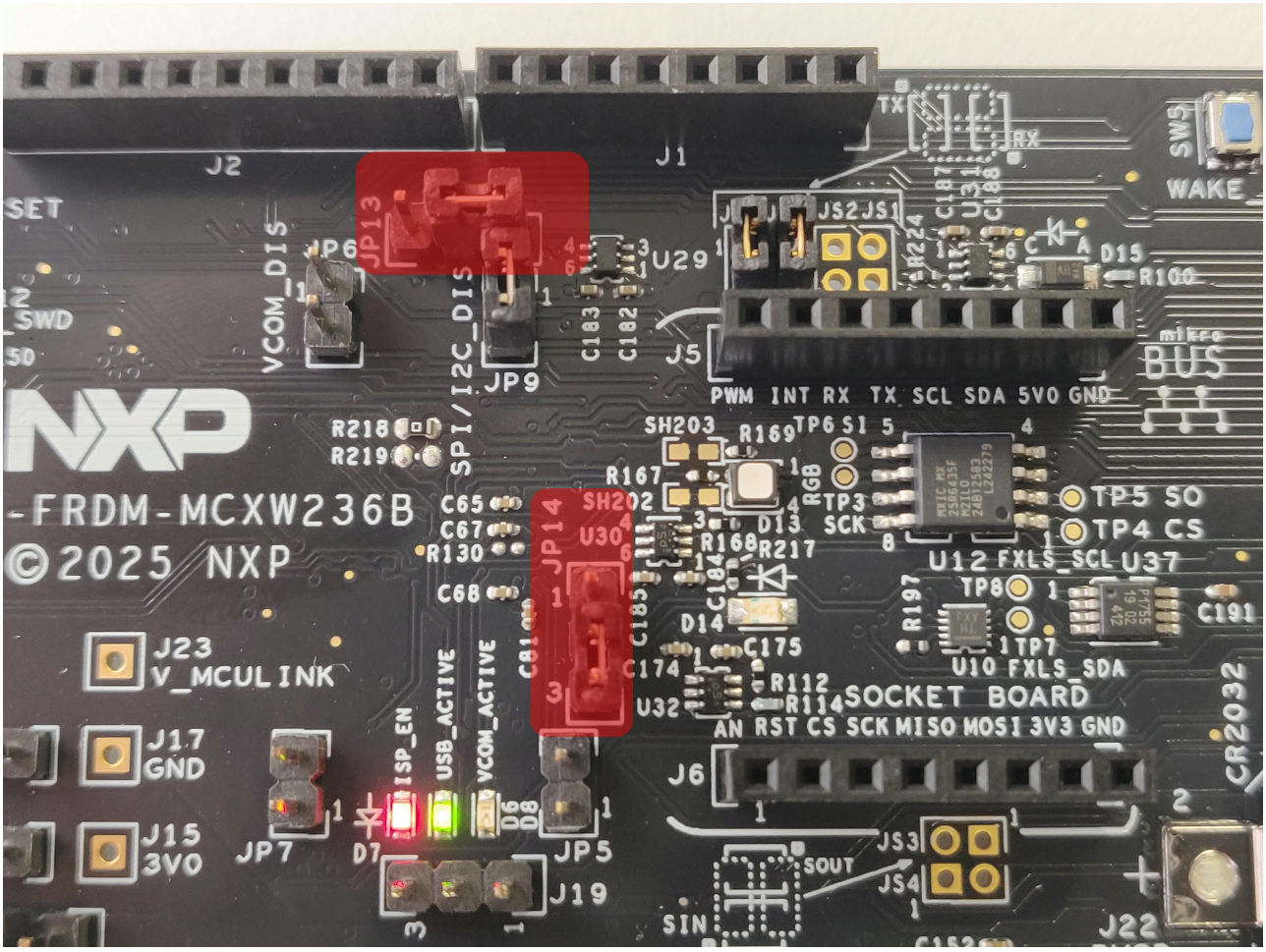


Figure 17. Setting the UART to Flexcomm 2

The next step is to connect the micro-USB cable to the board. It boots the board, but it must not be in ISP mode yet. To boot in ISP mode, hold down the SW3 "ISP" switch, proceed to press the SW1 "RESET" switch, and finally release the SW3 "ISP" switch.

6.3.2 Checking the connection in the SEC tool

In the SEC tool, we test to see if we successfully booted into ISP mode. On the top left, click "Target" > "Connection …". It opens a new window where the user can test the connection.

1. Ensure that UART is selected and put in the right parameters. The default baud rate works. The port differs from PC to PC.
2. When the right parameters have been selected, the user can click the "Test connection" button. A set of status values and a green OK result pop-up.

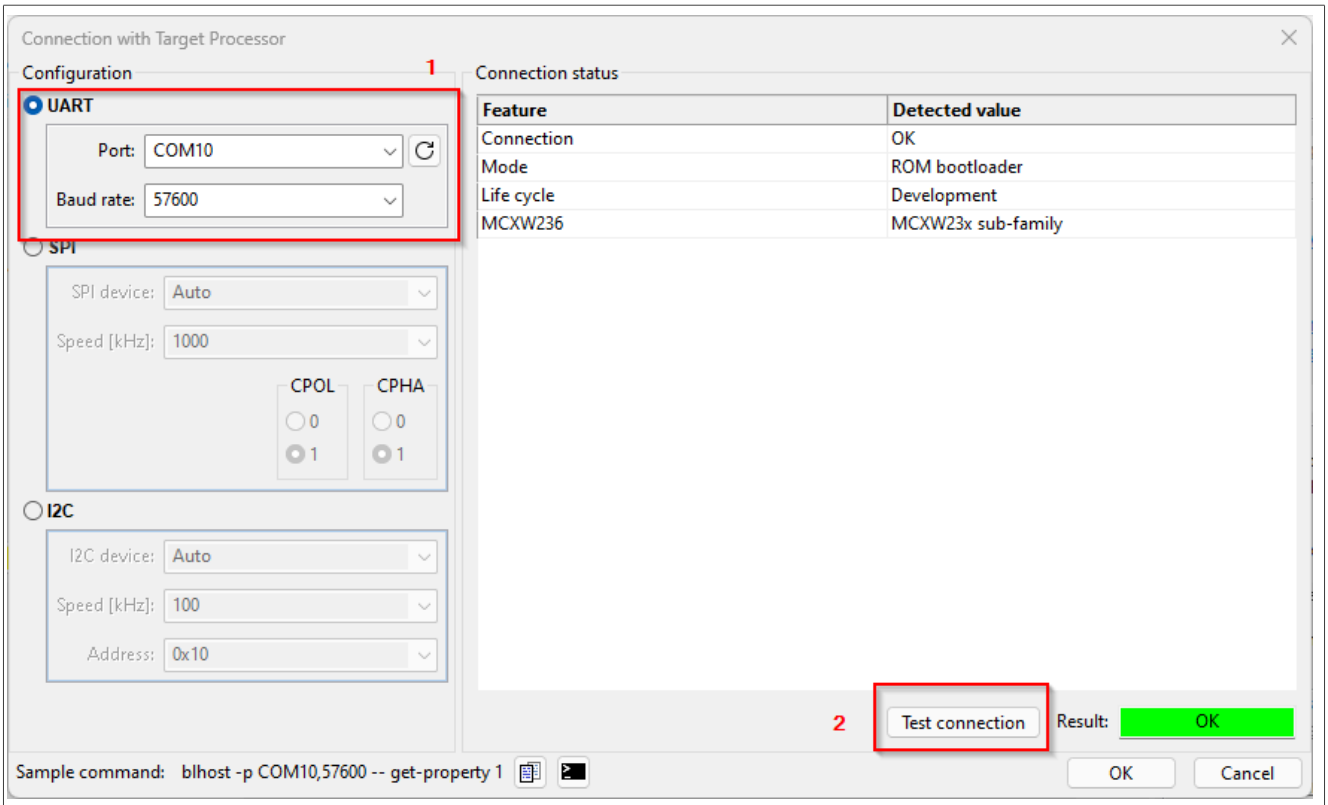


Figure 18. Configuration with target processor

This configuration is persistent and is used when flashing images and configuration.

For more information, see the document "MCUXpresso Secure Provisioning Tool User Guide" (Ref. 3).

6.4 SEC tool layout explanation

For the following steps, we first introduce the main components present in the general view of the tool. The application has multiple views, which are explained in the following sections. By default the "Build image" view opens. The different views have consistent areas, which are part of all views.

6.4.1 Consistent areas

The two consistent areas across the different views are the header and the log. The header contains three layers, one to change tool preferences, a second one to configure the boot of the device and a third to switch between the views.

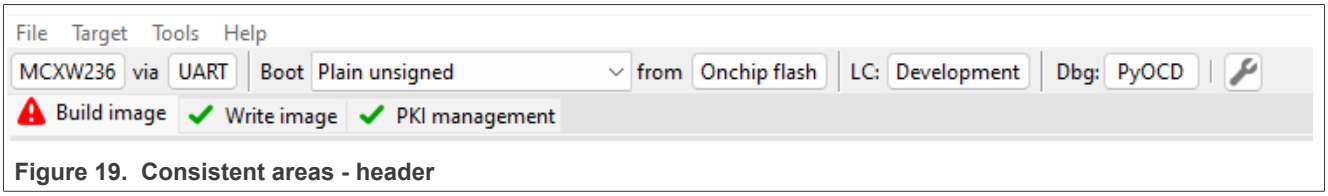
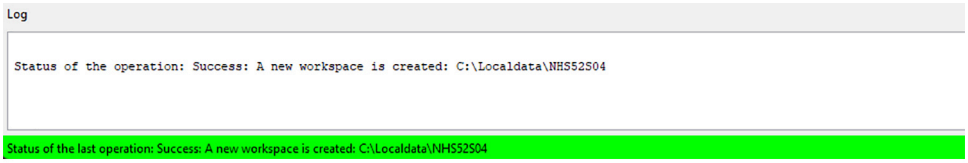


Figure 19. Consistent areas - header

The log area contains all the CLI commands, which are run when operations are executed. So, they can be easily scripted into a manufacturing phase.



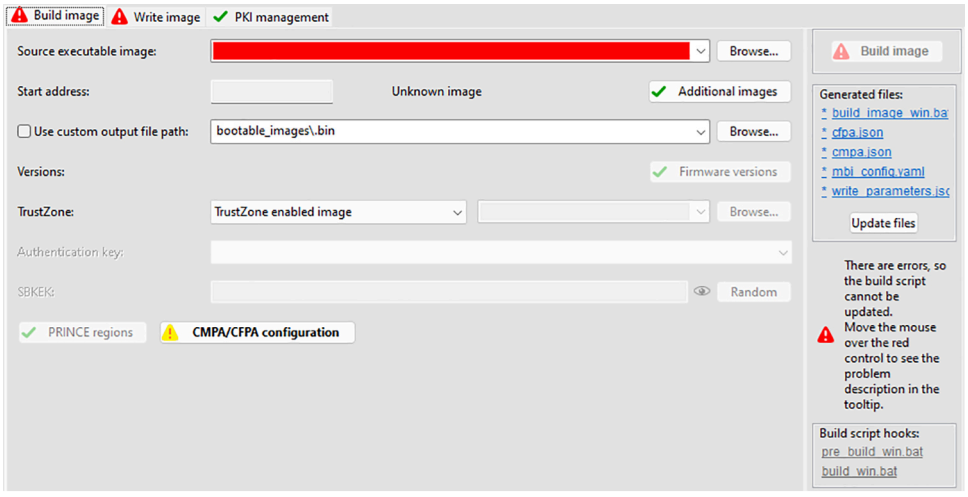
aaa-057232

Figure 20. Consistent areas - Log

6.4.2 Build image view

In this view, parameters of the executable image, like the firmware version, are set if the image is TrustZone enabled or not. This view also includes the settings of the PRINCE memory encryption if that has been enabled in the Customer Manufacturing Programmable Area (CMPA) device and the Customer Field Programmable Area (CFPA) configuration.

After a build, these settings result in the setup files of the device and the SB2.1 container with the encapsulated image.



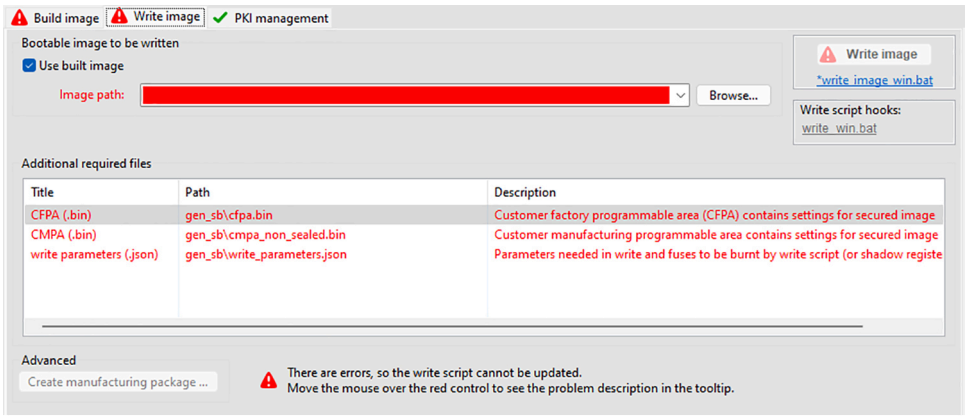
aaa-057233

Figure 21. Build image view

6.4.3 Write image view

This view consists of the selection of the SB2.1 container, which has previously been explained in Section 4; a list of all built artifacts, which are written to the device.

The write button sets up the device with the setup from the "Build image" view.



aaa-057234

Figure 22. Write image view

6.4.4 PKI management view

This view allows the setup of all the key material used in the device, like the image authentication credentials and the debug authentication credentials. The right-hand side of the view has multiple buttons to generate those specific key materials.

aaa-057235

Figure 23. PKI management view

6.4.5 Flow of development

The normal flow of development is to first choose a boot mode for the device. The boot mode can be selected in the header of the consistent area, so it is found on all views. After configuring the right header parameters, the right key material must be created within the "PKI management" view. After completing the key material generation, they can be used to build a secure configuration in the "Build image" view. When the image and its artifacts are built, they can be applied to the device using the "Write image" view.

6.5 Example setup

6.5.1 Boot mode: Encrypted and signed

Selecting one of the available types of boot depends on what the user is trying to protect.

- Signed binaries are verified during boot, hardening the device against booting malicious code.
- Encrypted images are stored encrypted in flash and the PRINCE engine decrypts them on-the-fly, when required.

For this example, we try to protect both by using the "Encrypted (PRINCE) and signed" boot mode.

Figure 19 show what the header looks like after selecting the right boot mode. If information about other boot modes is required, see the previous sections of this application note or the "MCUXpresso Secure Provisioning Tool User Guide v.9" (Ref. 3).

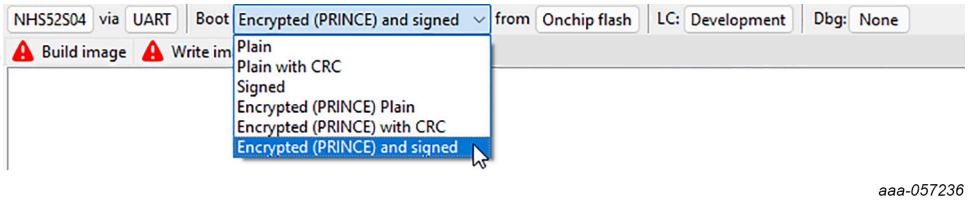


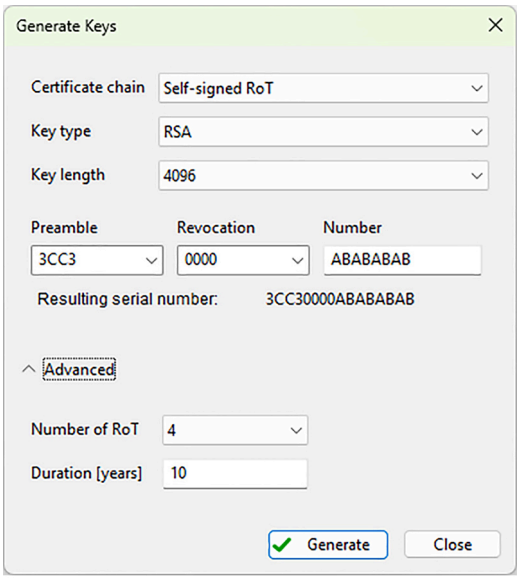
Figure 24. Encrypted PRINCE and signed boot mode

If the "UART" button is still red, see Section 6.3.

6.5.2 PKI management

In the PKI management view, we just set authentication keys as a starting point. At a later stage, we try debug authentication and show how it can be set up.

To generate authentication keys, clicking "Generate keys…" in the "PKI management" view opens a window, which allows the configuring of the certificate chain setup. To ease the development effort, the choice to use self-signed RoT was made. It is possible that it is not the right configuration that the user wants for the development.



aaa-057237

Figure 25. Generate keys

The following setup was made:

- Certificate chain: "Self-signed RoT"
- Key type: RSA
  The MCX W23 only uses RSA.
- Key length: 4096
  Comes with a time and power penalty. It can be changed but the recommended minimum for self-signed is 3072.
- Serial number construction: default
  For other setup possibilities, see Section 5.1.2.
- Number of RoT: 4
  Number of certificates chains to generate.
- Duration [years]: 10
  Certificate validity period.

After clicking the "Generate" button, the SEC tool generates the private keys to sign the images and the certificated and public keys to be hashed and put into the device. While we make no recommendations on how to protect private key material, storing them on a generic office PC could not fit the risk assessment of the user. Consider using the generated scripts as a starting point to replicate the key generation procedure on a well-secured machine such as an hardware security module (HSM).

The "PKI management" now looks something like this:

aaa-057238

Figure 26. PKI management authentication keys

6.5.3 Image configuration

After the PKI management, we can start configuring the image setup in the "Build image" view. In this section, we assume that you have your unprotected binary ready and at hand.

- **Source executable image**: You select the unprotected binary here.
  1. Click the "Browse…" button next to the dropdown field.
  2. Select the ".bin" or ".axf" file that you want to use.
- **Start address**: This field is filled when an .afx file is used.
  If the field is not filled, fill in the "Start address" field with the value of the start address of the binary.
- **Use custom output file path**: This field is left default for this example.
- **Versions**: This field is an anti-rollback countermeasure. The device has implemented a monotonic counter, which can be configured to prevent the rollback of the firmware.

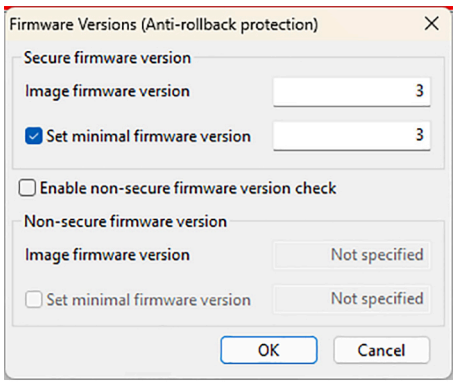


Figure 27. Firmware versions

1. Click the "Firmware versions" button. It opens a window where you can configure your own image versioning.
   We only use the "Secure firmware version" section for this example.
2. Set "Image firmware version" to 3.
3. Select the "Set minimal firmware version" checkbox and also set it to 3.
   This last 3 is reflected within the CFPA.
4. Click the OK button to finish the configuration.

- **Trustzone**: This field can be left default (Trustzone disabled image). We are not using it for this example.
- **Authentication key**: This field is used for image authentication.
  1. Click to open the dropdown menu and select ROT1.
- **SBKEK**: This field is a key encryption key used for encrypting key material in the SB2.1 container. For more information, see Section 4.3.3.1.
  1. Click the “Random” button a couple of times to generate a random 32-byte HEX string.
     You can view the random string by clicking the Eye icon.

6.5.4 PRINCE regions and CMPA/CFPA configuration

For these configurations, we stay in the "Build image" view and use the corresponding buttons.

6.5.4.1 PRINCE regions setup

Click the "PRINCE regions" button. The "PRINCE Region Configuration" window opens.

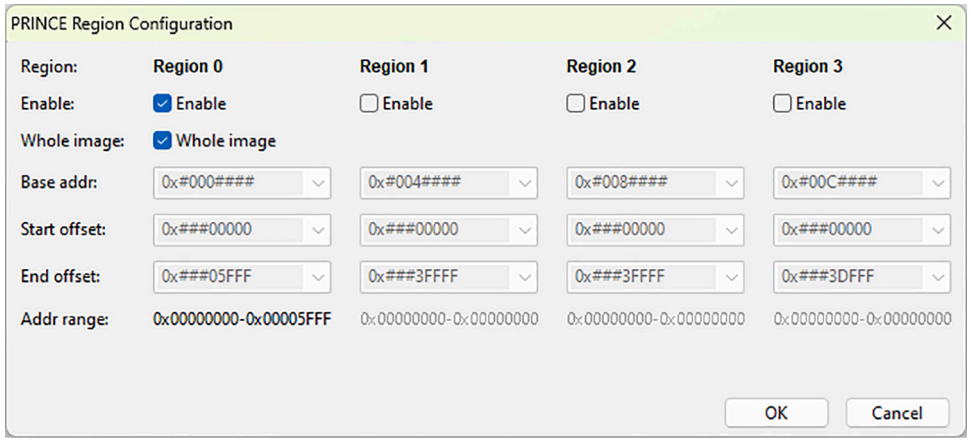


Figure 28. PRINCE regions configuration

Region 0 is enabled and "Whole image" is selected by default.

Each region is 256 kB in size, meaning that if the binary used fits within this size, the "whole image" option works. When the size of the binary exceeds 256 KB, a warning is shown notifying the user to enable sequential regions to cover the whole image.

The point of the different regions is that each region can use a separate key for its encryption/decryption. So, you can in turn create certain areas of the binary to be encrypted with different keys and leave certain areas unencrypted to reduce the power/compute penalty that PRINCE causes.

An example of different non-sequential regions is when using TrustZone.

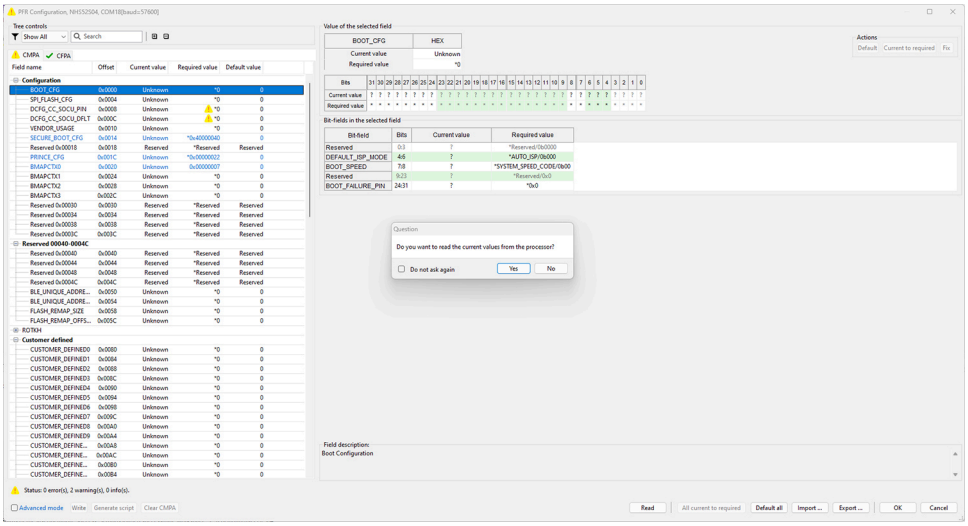
- The secure application can start at address 0x#000_0000 and fits within 256 KB, region 0 can be used with key 0.
- The non-secure application can start at address 0x#008_0000 and fits within 256 KB, region 2 can be used with key 2.

You can experiment with the regions and ranges of encryption. However, to keep things as simple as possible, default settings are used.

More information on PRINCE can be found in section 41.6 of the "MCX W23 Reference manual (Reference 6).

6.5.4.2 CMPA/CFPA configuration

Click the "CMPA/CFPA configuration" button. The "PFR configuration" window opens.



aaa-057245

Figure 29. PFR configuration

First, the window asks if you want to read the currently active values in the device. It is a good way to compare the values, which the tool has already set and the new values from the device.

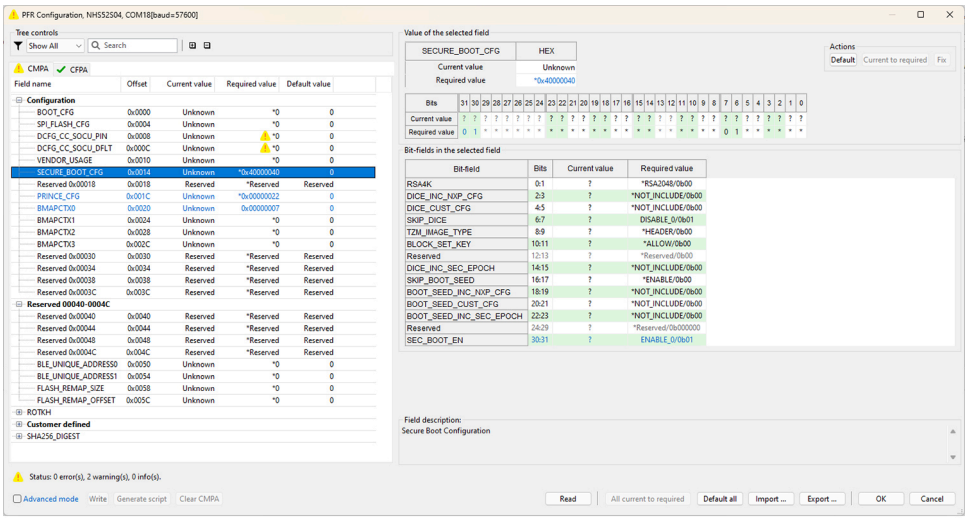
As mentioned before, the SEC tool has already set some of the values while going through the normal setup flow:

- In CMPA, the "SECURE_BOOT_CFG", "PRINCE_CFG", and "BMAPCTX0" words have been changed.
- In CFPA, the "S_FW_Version" and "ROTKH_REVOKE" words have been changed.

These values are highlighted in blue in the tool.

We consider the exact details of what has been set. As an example, click the "SECURE_BOOT_CFG" word in CMPA. The right-hand side of the window now shows the bit representation of the word itself.

In this "bit-editor", the exact value that has changed since last time is also highlighted in blue.

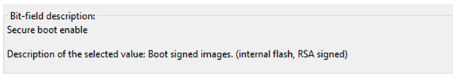


aaa-057246

Figure 30. PFR configuration + "bit editor"

Looking at the example, we check what has been set. In the word "SECURE_BOOT_CFG" of the CMPA, the bits of "SEC_BOOT_EN" have been set to "01" which denotes the enabling of secure boot.

Clicking the specific bit opens the "Field description" box with an explanation of what the bit does exactly.



aaa-057247

Figure 31. Field description box

To know exactly what all the words in the CMPA and the CFPA mean, see Section 4.3.

The default way in which the words "DCFG_CC_SOCU_PIN" and "DCFG_CC_SOCU_DFLT" are set up in CMPA can cause the device to enter a state that is unfavorable for the user. So, the user must change these words. The bit values within these words depend on the required outcome. Table 8 can be used as a guideline to set up the words for a "n"-specific bit.

Table 8. "DCFG_CC_SOCU_PIN" and "DCFG_CC_SOCU_DFLRT" in CMPA

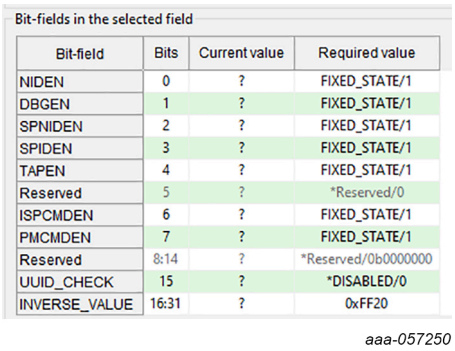
Security level	DCFG_CC_SOCU_PIN[n]	DCFG_CC_SOCU_DFLT[n]
restriction LV0	bit value = 1	bit value = 1
restriction LV1	bit value = 0	bit value = 0
invalid	bit value = 0	bit value = 1
restriction LV3	bit value = 1	bit value = 0

The security levels can be explained in the following way:

- LV0: Access to the "n" specific bit is always enabled.
  For example, bit 6 ISPCMDEN (ISP boot command enable) access is available for anyone to use.
- LV1: Access to the "n" specific bit is disabled at boot-up.
  It can be changed by using the right debug credential (when debug authentication is enabled). For example, bit 6 ISPCMDEN (ISP boot command enable) access is available to users who are authenticated and use the correct debug credential.
- LV3: Access to the "n" specific bit is always disabled and cannot be reenabled.Figure 32

For example, bit 6 ISPCMDEN (ISP boot command enable) access is never available.

For this example, we set them all to LV0 in CMPA. So, all the bits of the DCFG_CC_SOCU_PIN register must be set to 1. The ones of the DCFG_CC_SOCU_DFLT register must be set to 1 as well. shows what it looks like.



Bit-fields in the selected field

Bit-field	Bits	Current value	Required value
NIDEN	0	?	FIXED_STATE/1
DBGEN	1	?	FIXED_STATE/1
SPNIDEN	2	?	FIXED_STATE/1
SPIDEN	3	?	FIXED_STATE/1
TAPEN	4	?	FIXED_STATE/1
Reserved	5	?	*Reserved/0
ISPCMDEN	6	?	FIXED_STATE/1
PMCMDEN	7	?	FIXED_STATE/1
Reserved	8:14	?	*Reserved/0b0000000
UUID_CHECK	15	?	*DISABLED/0
INVERSE_VALUE	16:31	?	0xFF20

aaa-057250

Figure 32. CMPA - bit field descriptions

"INVERSE_VALUE" must also be set for "PIN" and "DFLT". It is not similar. For "PIN", the inverse can be calculated as follows:

- Bit representation: 0000 0000 1101 1111
- Inverse bit representation: 1111 1111 0010 0000
- Hexadecimal representation of inverse: 0xFF20

For "DFLT", the inverse can be calculated as follows:

- Bit representation: 0000 0000 1101 1111
- Inverse bit representation: 1111 1111 0010 0000
- Hexadecimal representation of inverse: 0xFF20

As this setup allows anyone to use the debug freely, it is not secure and must be secured for the production-ready setup using either LV1 or LV2.

Exiting out can be done by canceling (if no changes have been made) or clicking OK to apply the changes, which are written for the next step then.

6.5.5 Build image

While staying on the "Build image" view, you can now click the "Build image" button on the right side to generate the artifacts, which are written to the device.

6.5.6 Write image

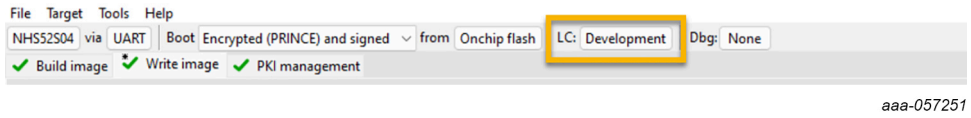
After the build, everything must come together and be flashed into the device, which can be done in the "Write image" view.

When the device is connected, click the "Write image" button on the right side and the device is set up with the right configuration.

6.5.7 Locking configuration

The previous steps do not lock the device yet. Everything from the previous steps can be reversed. It means that we can roll back to a previous firmware version, undo the signature check, and update the CMPA and the CFPA.

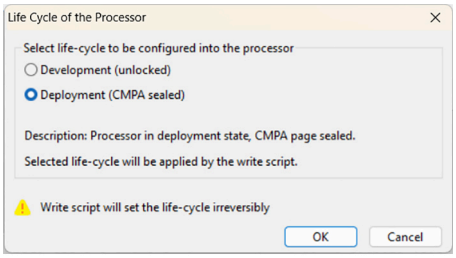
To configure the device to its final (setup) state, we must advance the life cycle. Advancing the life cycle can be done by sealing the CMPA, which must be done through the tool using the "Deployment" life-cycle state.



aaa-057251

Figure 33. deployment life cycle - development button

Click the "Development" button. A pop-up window with the selection to go to "Deployment" appears. It must be used with caution and only if the CMPA configuration is ready to be written.



aaa-057252

Figure 34. Life cycle of the Processor - Deployment

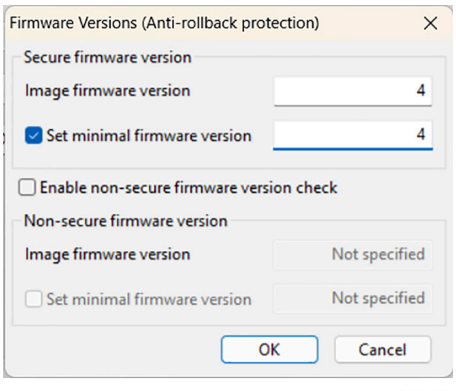
The life cycle advancing is not reversible. So, if executed, changes to any value in CMPA are blocked.

After checking the "Deployment (CMPA sealed)" radio button, click OK to make the tool seal the CMPA at the next image-write operation.

6.5.8 Image updating

After a device has been sealed (CMPA sealed), the CFPA and the firmware can still be updated. However, it can only be updated if the CMPA does not restrict it from happening by completely locking down or requiring a debug credential to do so. It depends on the previously set security level (see Table 8).

We will also update the firmware version to 4 and enable rollback protection to firmware version 3.



aaa-057253

Figure 35. Upping the firmware version

In the "Build image" view:

1. Select the new firmware image in the "Source executable image" field. open the "Firmware Versions" window.
2. Update the "Image firmware" field to 4.
3. Update the "Set minimal firmware version" field to 4.
   If we keep the "minimal firmware version" at 3, it still works. However, it allows anyone with the version 3 SB2.1 container to reverse the device to a previous version. If we only update the "minimal firmware version" and keep the "image firmware version" set to 3", the image does not boot because we only allow version 4 or higher.
4. When all is set, click OK to close the "firmware versions" window, saving the new settings.

The CFPA can now be updated. We can also use "Build image" and "Write image" to update the firmware.

7 Things to consider before deploying a secure product

Before switching the life cycle of the processor to "Deployment", ensure that the actions mentioned below have been taken.

Keys

- Safeguard your SBKEK and RoT private keys (and USERKEK, if applicable).
- Fill in the RKTH field with the necessary value derived from your RoT keys. So, the necessary RoT keys are enabled in ROTKH_REVOKE while the others are disabled.
- Provision an SBKEK to the platform and generate the PRINCE keys on the platform.
  - To block the possibility to add a key to the PUF when the product is deployed (recommended), an AES key and/or a USERKEK key must be manually provisioned to the PUF as well (see the MCX W23 for Reference manual Reference 6).

Settings

- Enable the mandatory check on image signatures in the toolbar of the SEC tool or directly in CMPA.
- If you design an application based on TrustZone (recommended), enable the mandatory use of it.
- Regardless of the previous point, ensure that the TrustZone setting in CMPA does not conflict with the image type that is downloaded to and booted on the platform. It bricks the platform.
- Enable PRINCE on-the-fly encryption on the internal flash memory regions containing sensitive information that must be protected (securing confidentiality).
- Apply the global lock (GLK) on the PRINCE settings.
- Close off all debug functionality from unauthenticated access (no debug feature must be enabled all the time unconditionally, that is, have "FIXED_STATE/1" and “ENABLED/1” together). Secure debug rules must be enforced (see section 4.6 of SEC_PRIM; Reference 1).
  - Test if a port is not open all the time by using the "Open debug port…" tool in the PKI management tab. If the connection is rejected automatically, or debug credentials are required, that debug functionality was successfully closed off from open access. For information on the use of debug authentication, see the LPC55Sxx debug authentication application note (Reference 4).
- Disable the ISP mode as an additional layer of protection. It can be done by changing the DEFAULT_ISP_MODE bits in the BOOT_CFG word in CMPA to 0b111.

Finally

- Verify all the values that you have entered in CMPA, including the inverse values of the debug fields, as they are final.

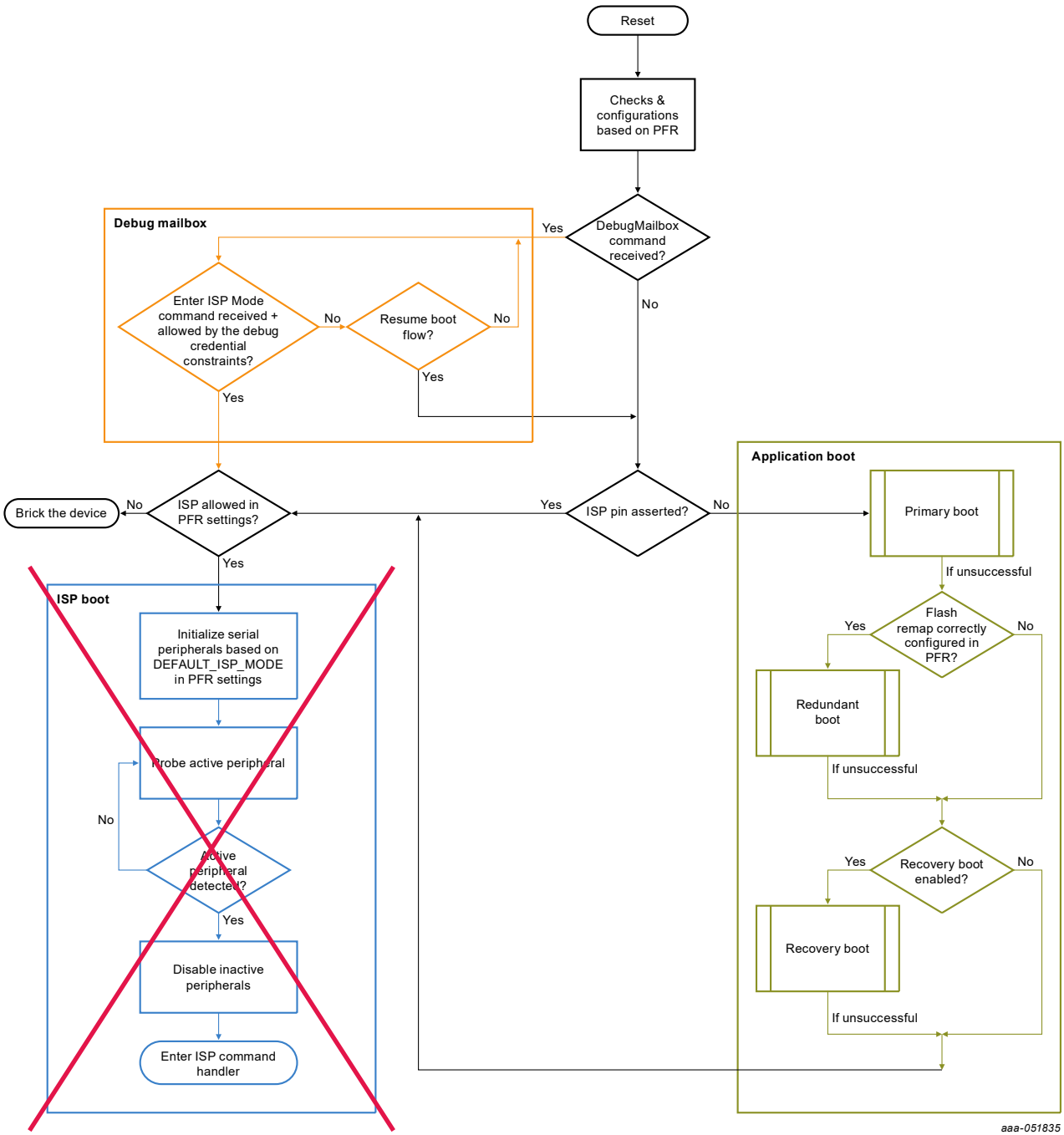


Figure 36. Boot flow in deployment mode

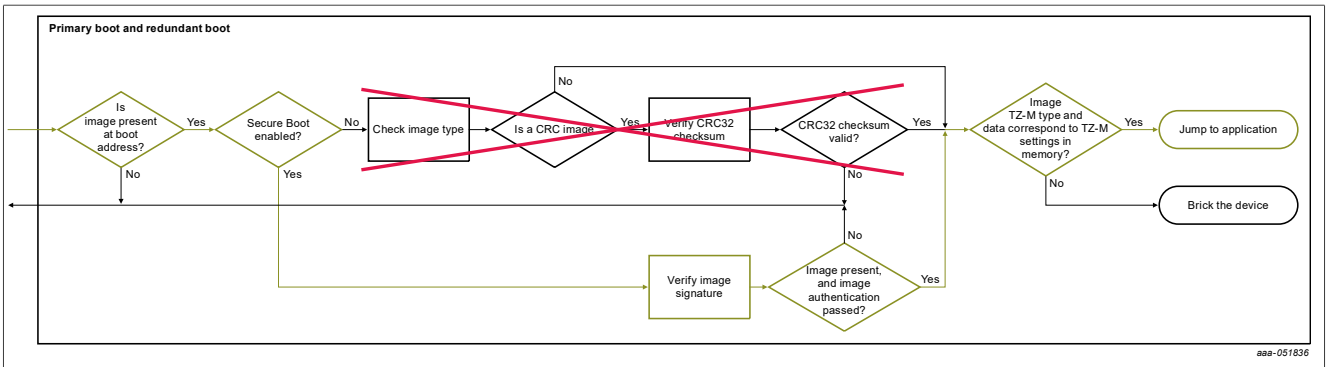


Figure 37. Boot flow in deployment mode - primary/redundant boot

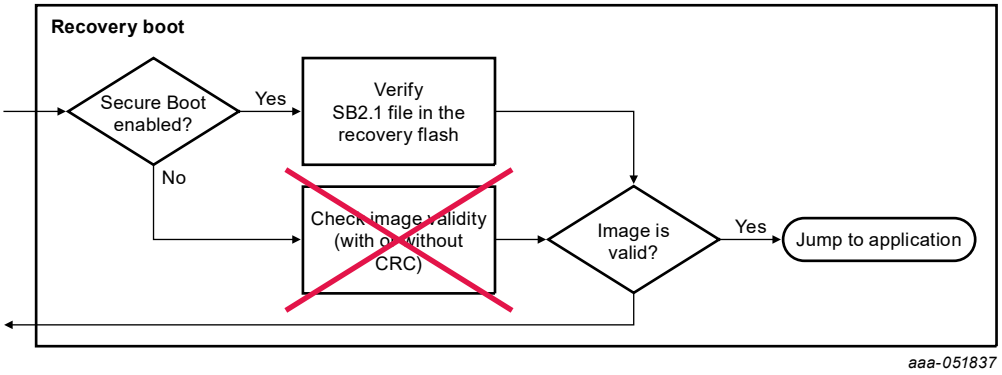


Figure 38. Boot flow in deployment mode - recovery boot

8 Verifying that secure boot is enabled

If SEC_BOOT_EN is enabled, an image that is not signed is rejected. An image that does not pass the authentication checks is also rejected.

Get-property LifeCycleState (0x11) can verify the sealing of CMPA (and so, the deployment mode). The unavailability of certain ISP commands such as read-memory or write-memory can also verify it.

9 Actions that can brick an MCX W23 chip

The following actions can cause an MCX W23 chip to become unusable:

- Enabling field analysis (FA)[11] mode through the dedicated CFPA field or through the dedicated debug mailbox command (if the DCFG_CC_SOCU settings allow it).
- Triggering fatal mode through:
  - Faulty debug configurations:
    - When changing the debug access restriction of any debug domain in the DCFG_CC_SOCU fields in CMPA, forgetting to write the corresponding inverse value to the upper halfword.
    - When changing the debug access restriction of any debug domain in the DCFG_CC_SOCU_NS fields in CFPA, forgetting to write the corresponding inverse value to the upper halfword.
    - Changing debug configurations in the DCFG_CC_SOCU_NS fields in CFPA while leaving the DCFG_CC_SOCU fields in CMPA in their default, all-zero value.
  - Erasing a contiguous PRINCE-encrypted area and then writing content to it that does not fill up the whole encrypted area.
  - Loading an image whose TrustZone-M image type in the image header conflicts with the required TrustZone-M image type found in the TZM_IMAGE_TYPE bits at position [9:8] in SECURE_BOOT_CFG in CMPA.

11 Also called return material analysis (RMA), used to "retire" the platform and send it over to NXP for analysis.

10 Abbreviations

Table 9. Abbreviations

Acronym	Description
AES	advanced encryption standard
API	application programming interface
CA	certificate authority
CC	credential constraint
CFG	configuration
CFPA	customer in-field programmable area
CMD	command
CMPA	customer manufacturing programmable area
CRC	cyclic redundancy check
CRL	certificate revocation list
DCFG	device configuration
DER	distinguished encoding rules
EVK	evaluation kit board
FA	field analysis
FIPS	federal information processing standards
HSM	hardware security mudule
IDE	integrated development environment
I^2C	inter-integrated circuit
ISP	in-system programming
KEK	key encryption key (= key wrapping key)
KSA	key store area
NS	non-secure
PC	program counter
PFR	protected flash region
PKCS	public-key cryptography standard
PKI	public-key infrastructure
PUF	physical unclonable function
RAM	random access memory
RKTH	root key table hash
RMA	return material analysis
ROM	read-only memory
RoT	root-of-trust
RFC	request for comments
RSA	Rivest-Shamir-Adleman

Table 9. Abbreviations...*continued*

Acronym	Description
SB	secure binary
SHA	secure hash algorithm
SOCU	system-on-chip unit
SPI	serial peripheral interface
SRAM	static random-access memory
SWD	serial wire debug
TEE	trusted execution environment
TF-M	TrustedFirmware for Arm Cortex-M processors
TZM	TrustZone for Arm Cortex-M processors
UART	universal asynchronous receiver-transmitter

11 References

Table 10 lists the references used to supplement this document.

Table 10. Related documentation

[1]	**SECPRIMWPA4 REV 1**	—	White paper on secure primitives; 2023, NXP Semiconductors (document Security Primitives: Requirements in (I)IoT Systems)
[2]	**SEC tool**	—	NXP's Secure Provisioning Tool
[3]	**MCUXSPTUG**	—	MCUXpresso Secure Provisioning Tool User Guide (document MCUXSPTUG)
[4]	**AN13037 Application note**	—	LPC55Sxx debug authentication (document AN13037)[1]
[5]	**AN12324 Application note**	—	LPC55Sxx usage of the PUF and Hash Crypt to AES coding (document AN12324)[1]
[6]	**Reference manual**	—	MCX W23 Reference Manual (document MCXW23RM)

[1] This document is an example from a similar platform, as there is no specific document yet for this product.

12 Revision history

Table 11 summarizes the revisions done to this document.

Table 11. Revision history

Document ID	Release date	Description
AN14657 v.1.0	05 August 2025	Initial pulic 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.

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.

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.

Suitability for use in industrial applications (functional safety) — This NXP product has been qualified for use in industrial applications. It has been developed in accordance with IEC 61508, and has been SIL-classified accordingly. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Tables

Tab. 1. ISP download mode based on DEFAULT_ ISP_MODE bits (BOOT_CFG[6:4] in CMPA) 15
Tab. 2. RSA4K bits (SECURE_BOOT_CFG[1:0] in CMPA) 15
Tab. 3. SEC_BOOT_EN bits (SECURE_BOOT_ CFG[31:30] in CMPA) 15
Tab. 4. DCFG_CC_SOCU fields for ISP_CMD_EN 16
Tab. 5. DM-AP commands 17
Tab. 6. ROTKH_REVOKE table bit field description 20
Tab. 7. TZM_IMAGE_TYPE bits (SECURE_ BOOT[9:8] in CMPA) 23
Tab. 8. "DCFG_CC_SOCU_PIN" and "DCFG_CC_ SOCU_DFLRT" in CMPA 35
Tab. 9. Abbreviations 44
Tab. 10. Related documentation 46
Tab. 11. Revision history 47

Figures

Fig. 1. Illustration of the three main boot tracks (debug mode, ISP mode, and OEM application boot mode) 4
Fig. 2. ISP mode flow 6
Fig. 3. MCX W23 image types 7
Fig. 4. Primary boot of an unsigned image 8
Fig. 5. Primary boot of an unsigned image protected with CRC 8
Fig. 6. Primary boot of a signed image 9
Fig. 7. Signed image 10
Fig. 8. Recovery boot 12
Fig. 9. Fall-through to ISP mode 13
Fig. 10. PFR sections impacting the bootloader 14
Fig. 11. RKTH generation process 17
Fig. 12. SB2.1 container decryption by SBKEK (arrows) 18
Fig. 13. Additional security settings in PFR 19
Fig. 14. Serial number structure 21
Fig. 15. Configuration of TrustZone for an application 22
Fig. 16. SEC tool - creating a new workspace 25
Fig. 17. Setting the UART to Flexcomm 2 26
Fig. 18. Configuration with target processor 27
Fig. 19. Consistent areas - header 27
Fig. 20. Consistent areas - Log 28
Fig. 21. Build image view 28
Fig. 22. Write image view 29
Fig. 23. PKI management view 29
Fig. 24. Encrypted PRINCE and signed boot mode 30
Fig. 25. Generate keys 31
Fig. 26. PKI management authentication keys 31
Fig. 27. Firmware versions 32
Fig. 28. PRINCE regions configuration 33
Fig. 29. PFR configuration 34
Fig. 30. PFR configuration + "bit editor" 35
Fig. 31. Field description box 35
Fig. 32. CMPA - bit field descriptions 36
Fig. 33. deployment life cycle - development button 37
Fig. 34. Life cycle of the Processor - Deployment 37
Fig. 35. Upping the firmware version 37
Fig. 36. Boot flow in deployment mode 40
Fig. 37. Boot flow in deployment mode - primary/ redundant boot 41
Fig. 38. Boot flow in deployment mode - recovery boot 41

Contents

1 Introduction 2
2 Bootloader functionality 3
3 In-system programming (ISP) mode 5
4 Application boot 7
4.1 Ways of protecting the image on-chip 7
4.1.1 Unsigned image 8
4.1.2 Unsigned image with CRC 8
4.1.3 Signed image 8
4.1.3.1 Certificates and public-key infrastructure (PKI) 9
4.1.3.2 Image validation 9
4.2 Three application boot modes 11
4.2.1 Primary boot 11
4.2.2 Redundant boot 11
4.2.3 Recovery boot 11
4.2.4 Fall-through 12
4.3 Protected flash region (PFR) 14
4.3.1 CMPA 14
4.3.1.1 BOOT_CFG 15
4.3.1.2 SPI_FLASH_CFG 15
4.3.1.3 SECURE_BOOT_CFG 15
4.3.1.4 FLASH_REMAP_SIZE and FLASH_REMAP_OFFSET 16
4.3.1.5 DCFG_CC_SOCU_PIN and DCFG_CC_SOCU_DFLT 16
4.3.1.6 Root key table hash (RKTH) 17
4.3.2 CFPA 17
4.3.2.1 DCFG_CC_SOCU_NS_PIN and DCFG_CC_SOCU_NS_DFLT 17
4.3.3 KSA 18
4.3.3.1 SBKEK 18
5 Additional security mechanisms 19
5.1 Key revocation 19
5.1.1 On the platform 20
5.1.2 In the certificate chain of the image 20
5.2 PRINCE encryption on internal memory 21
5.3 TrustZone-M for core process separation 21
6 Configuring boot on MCX W23 using SEC tool 24
6.1 Assumptions and prerequisites 24
6.1.1 Assumptions 24
6.1.2 Prerequisites 24
6.2 Setting up the workspace 24
6.2.1 Creating a workspace in the SEC tool 24
6.3 Device connection 25
6.3.1 ISP mode on the EVK board 25
6.3.2 Checking the connection in the SEC tool 26
6.4 SEC tool layout explanation 27
6.4.1 Consistent areas 27
6.4.2 Build image view 28
6.4.3 Write image view 28
6.4.4 PKI management view 29
6.4.5 Flow of development 29
6.5 Example setup 30
6.5.1 Boot mode: Encrypted and signed 30
6.5.2 PKI management 30
6.5.3 Image configuration 32
6.5.4 PRINCE regions and CMPA/CFPA configuration 33
6.5.4.1 PRINCE regions setup 33
6.5.4.2 CMPA/CFPA configuration 34
6.5.5 Build image 36
6.5.6 Write image 36
6.5.7 Locking configuration 36
6.5.8 Image updating 37
7 Things to consider before deploying a secure product 39
8 Verifying that secure boot is enabled 42
9 Actions that can brick an MCX W23 chip 43
10 Abbreviations 44
11 References 46
12 Revision history 47
Legal information 48