

PSOC™ Control C3 – Security

Customer training workshop (CTW)

February 2026

Target products

Target product list for this training material

Family category	Series	Code flash memory size
PSOC™ Control C3 Entry Line	PSC3P2xD	Up to 256 KB
PSOC™ Control C3 Entry Line	PSC3M3xD	Up to 256 KB
PSOC™ Control C3 Main Line	PSC3P5xD	Up to 256 KB
PSOC™ Control C3 Main Line	PSC3M5xD	Up to 256 KB

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 Overview 4
2 Key features and capabilities 6
3 Basic boot flow 8
4 Life cycle stage (LCS) 10
5 Device provisioning 15
6 Edge protect tool 18
7 Boot configurations 20
8 Firmware upgrade 28
9 Protected firmware 33
10 CryptoLite 36
11 TrustZone® and system security architecture 42
12 Additional security protection units 53
13 Application architecture 61
14 Code example (CE) 65

Copyright © Infineon Technologies AG 2026. All rights reserved.

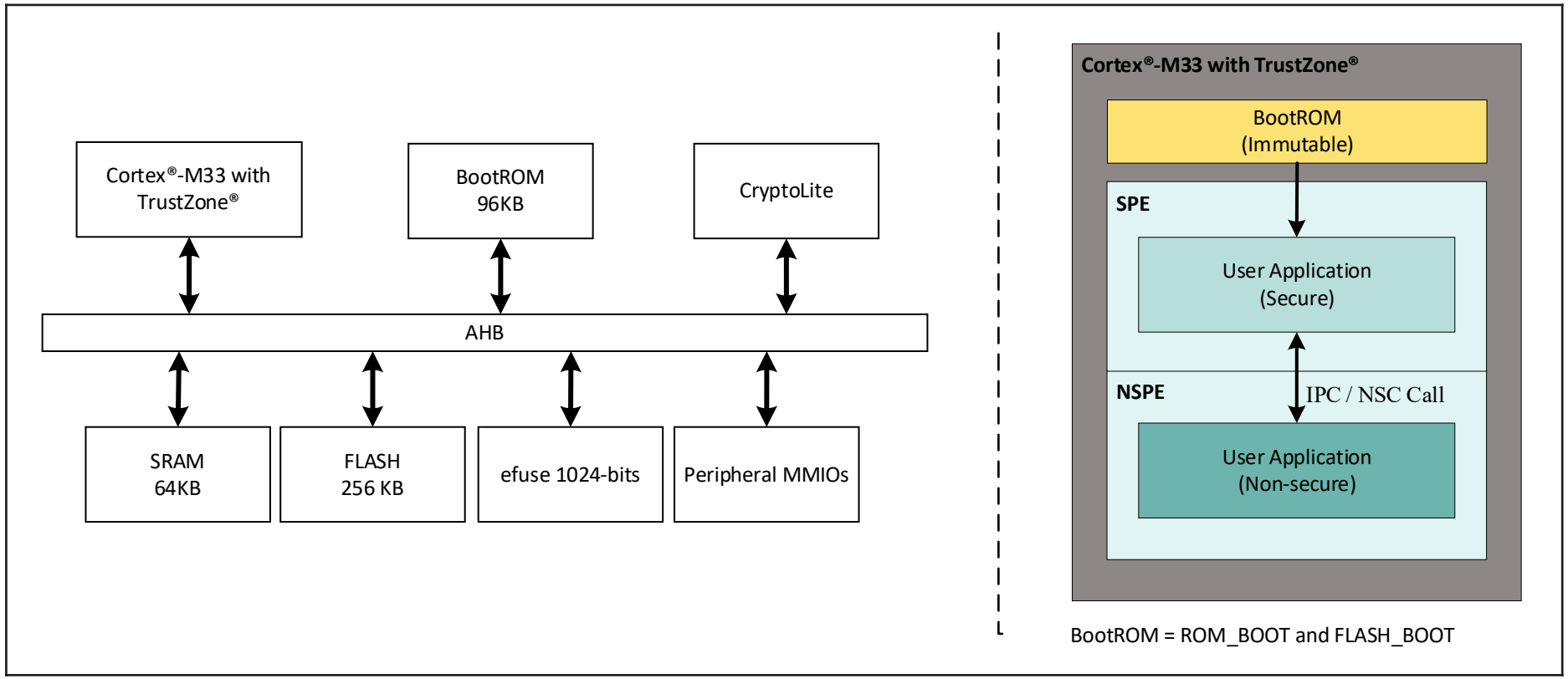
Table of contents

1	**Overview**	**4**
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Overview

The PSOC™ Control C3 MCU is a single-core device that contains an Arm® TrustZone® enabled Cortex®-M33 CPU, SONOS flash, a range of peripherals, including UART, SPI, I2C, CAN-FD, CRYPTO accelerators (CryptoLite, TRNG), and several other blocks.



Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1	Overview	4
2	**Key features and capabilities**	**6**
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Key features and capabilities

- Secure boot
  - Immutable BootROM provides a chain of trust that includes the first OEM Application code
  - Supports Fast Secure Boot – HASH verification less than 10 ms
- Arm® Cortex®-M33 with TrustZone® technology
  - Provides isolation between the secure world and the non-secure world
- Secure debug
  - Enforces debug control based on the debug policy that supports an authenticated debug token
  - The debug token is signed by the OEM_KEY provisioned by the OEM
- Edge Protect Bootloader support
  - Optionally, support MCUboot-based bootloader
- Secure firmware upgrade
  - Only authenticated OEM firmware images are allowed to boot
  - Secure firmware upgrade through UART, I2C, SPI, and CAN-FD
- Multi-image authentication
  - BootROM supports authentication up to three images
- Factory programming (DFU) with serial interface
- Infineon-signed RAM applications support device provisioning
- Dual-bank support
  - Live firmware upgrades
- PSA Level 2 certified device

Copyright © Infineon Technologies AG 2026. All rights reserved.

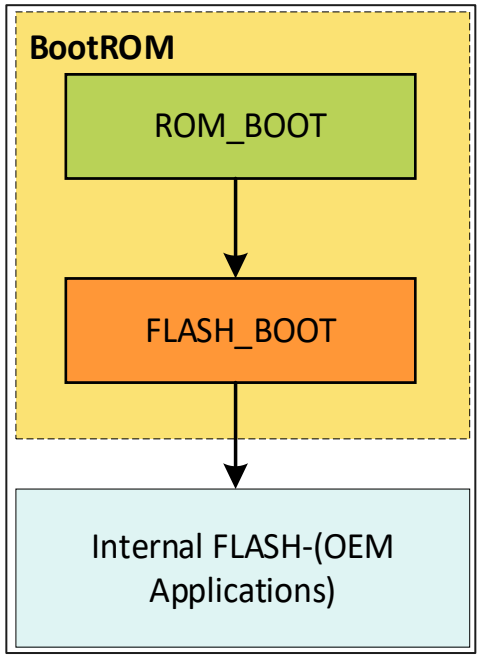
Table of contents

1	Overview	4
2	Key features and capabilities	6
3	**Basic boot flow**	**8**
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Basic boot flow

- The immutable BootROM is the first code that executes from power-on reset (POR) until the system launches the OEM application
  - BootROM consists of ROM_BOOT + FLASH_BOOT
  - ROM_BOOT authenticate FLASH_BOOT in SFLASH
  - FLASH_BOOT extends the ROM_BOOT functionality
  - FLASH_BOOT authenticates the first OEM application using the OEM-provided OEM_ROT_KEY
- Provisioning policy allows OEM to configure boot policy, in such a way that BootROM will launch OEM application with or without authentication



STEP	SW Asset	Authenticated by	Using
1	ROM	N/A Immutable	N/A Immutable
2	SFLASH_BOOT	ROM_BOOT	ASSET_HASH
3	OEM Keys and Provisioned Data	SFLASH_BOOT	SECURE HASH
4	EdgeProtectBootloader	SFLASH_BOOT	OEM Key
5	OEM Application	SFLASH_BOOT	OEM Key

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 Overview 4
2 Key features and capabilities 6
3 Basic boot flow 8
4 **Life cycle stage (LCS)** **10**
5 Device provisioning 15
6 Edge protect tool 18
7 Boot configurations 20
8 Firmware upgrade 28
9 Protected firmware 33
10 CryptoLite 36
11 TrustZone® and system security architecture 42
12 Additional security protection units 53
13 Application architecture 61
14 Code example (CE) 65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Life cycle stage (LCS) (1/4)

Three device life cycle stages (LCS) are important to the product developer:

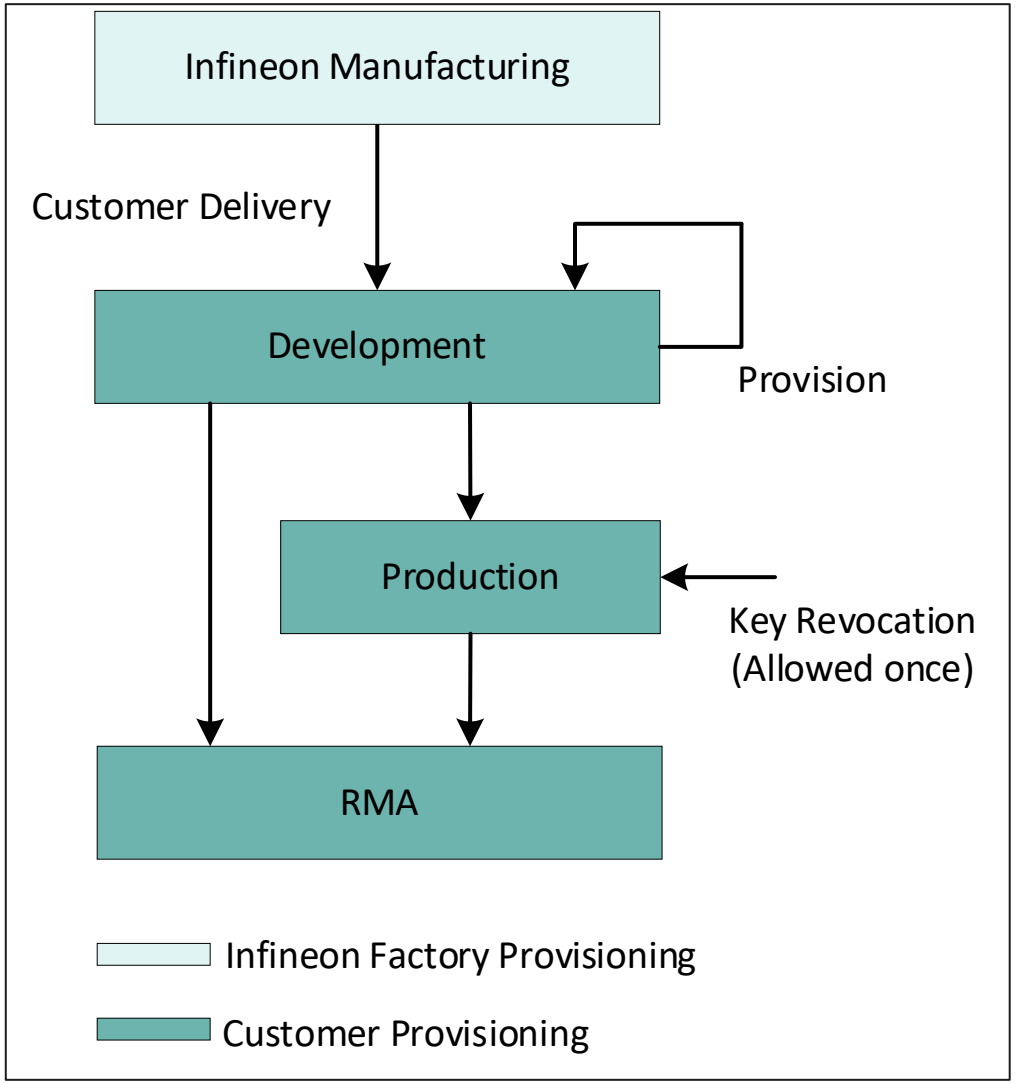
- Development
- Production
- Return merchandise authorisation (RMA)

- The BootROM execution flow depends on the life cycle stage (LCS) at which the MCU is running. The LCS stage is dictated by reading the efuse bit values from the device

- During provisioning, the device LCS stage is transitioned to the next LCS, as shown in the diagram

- Infineon-signed RAM applications are used to provision the assets into the device and transition the device to the LCS

- The RAM applications and the associated input parameters are loaded into the device SRAM through a SWD or DFU serial interface. The RAM applications are authenticated by BootROM before they execute from device SRAM



Copyright © Infineon Technologies AG 2026. All rights reserved.

Life cycle stage (LCS) (2/4)

- The LCS stages are sequential, and it cannot be reverted to the previous stage, since the LCS stage is maintained in the eFuse (one-time programmable) memory
- Devices are shipped in the development stage, which allows the device to be provisioned and programmed as many times as needed to develop the final application
- The devices are shipped with a default policy that defines all debug ports as open and allows application execution without code authorisation. This allows any user to program, evaluate, and debug the parts before any knowledge of security or code signing
- During the development stage, the OEM can take ownership of the device. This allows the owner to replace the default policy file with a custom policy

Copyright © Infineon Technologies AG 2026. All rights reserved.

Life cycle stage (LCS) (3/4)

Development LCS

- By default, the device is shipped to the customer in the Development LCS
- OEM policy assets are provisioned in this LCS stage
- Transition allowed in this stage is Production or RMA
- RAM Application will be used to do the device LCS transition from development to production
- Device can be provisioned several times as needed

Production LCS

- This is a lifecycle stage of a secured device, and LCS is mandated for the secure device
- Except for the BOOT_SIMPLE_APP configuration, all the rest of the configurations support production LCS
- OEM policy assets (for example, SECURE_HASH) are provisioned before moving to this LCS
- The device will launch the user application only when authentication is successful
- The only transition allowed in this stage is RMA LCS
- Once the device transitions into Production LCS, the protection state is set to secure, and DAP access restriction will be deployed based on OEM policy

Copyright © Infineon Technologies AG 2026. All rights reserved.

Life cycle stage (LCS) (4/4)

RMA LCS

The RMA stage is used only if a hardware issue is detected, and the device must be shipped to Infineon for failure analysis.

- RMA LCS enables Infineon to perform failure analysis (FA)
- To initiate FA, the customer transitions the device from the LCS to RMA and ships the device to Infineon
- A device is allowed to transition to the RMA LCS when it is currently in Development or Production LCS
- RMA transition in production LCS requires a certificate signed with the OEM_ROT_KEY
- During the RMA transition below list of assets will be cleared from the device
  - Hardware unique key- HUK
  - Individual authentication key- IAK
  - OEM policies (in SFLASH row 6)
  - PROT_FW policy (in SFLASH row 4)
  - IFX policies (in SFLASH row 4) - L1_HASH, L2_HASH, PROT_FW_HASH
    (**Note**: L1/L2 would be OEM images)
  - SFLASH backup row (row 63)

Copyright © Infineon Technologies AG 2026. All rights reserved.

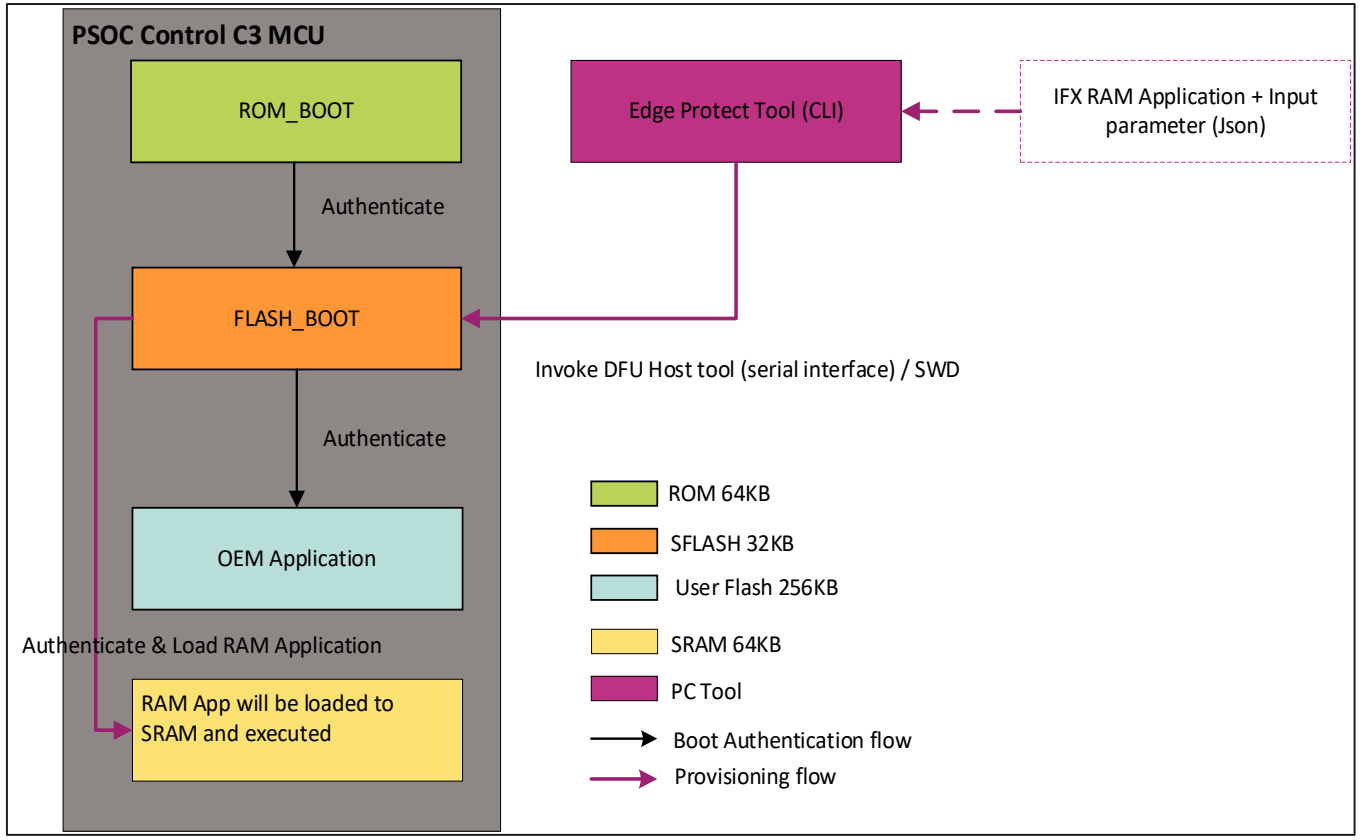
Table of contents

1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	**Device provisioning**	**15**
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Device provisioning

- Provisioning is an act of configuring a device with an authorized set of keys, parameters, certificates and firmware images
- Infineon signed RAM Applications are used to provision the device
- Policy json files are the input parameters for the RAM Application, where user can configure the parameters in policy file
- The RAM Application and the input parameters are converted as DLM (Downloadable module) package and send to device SRAM by Edge protect tools



Copyright © Infineon Technologies AG 2026. All rights reserved.

Device provisioning

- Edge Protect tool invokes SWD or DFU serial interface to send the DLM package to the device
- After the DLM package is loaded into SRAM, the BootROM authenticates the RAM application and transfers control to the RAM application. Additionally, the RAM application provisions the device based on policy inputs
- On the next reset, the BootROM execute based on device provisioning

Sl.No	RAM Application	Description
1	cyapp_entrance_exam	Checks the device integrity and whether the device has original factory configuration.
2	cyapp_prov_oem	The application injects OEM assets and provision the device
3	cyapp_reprovisioning	The application intended for OEM and/or ICV key revocation.
4	cyapp_to_rma	The application advance LCS to RMA and clear OEM provisioned assets.

```
{
  "image": {
    "path": "cyapp_prov_oem_signed_ifx0.bin",     # RAM App path
    "address": "0x34003000"                       # Address where the RAM App to be programmed
  },
  "in_params": {
    "path": "in_params.bin",                      # Path where the input parameters bin is generated
    "address": "0x3400D000"                       # Address where the input parameters is programmed
  },
  "dlm": {
    "path": "prov_oem_dlm.hex"                    # Path where the DLM path is generated
  },
  "allowed_lcs": [
    "NORMAL_PROVISIONED"                          # List of LCS in which the application is allowed to be run
  ]
}
```


Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	**Edge protect tool**	**18**
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Edge Protect tool

Edge Protect tool is an Infineon-developed software package (command-line interface) that comes along with the Early Access (EA) pack. It can be used as a standalone application. The executable can be found in the tools/edgeprotecttools/bin directory of the EA installation or within the Edge Protect Security Suite.

Key features

- **Keys creation**
  - Creates a private-public key pair
- **Integrity exam**
  - Passing an integrity exam before provisioning a device is an option to ensure that the device is a genuine Infineon device, and it is in a known state
- **Provisioning**
- Provisioning is the act of configuring a device with an authorised set of keys, policies, and optionally certificates. Supports SWD and serial interface
- **Signing user applications**
  - Signs a user application with a private key locally or using a hardware security module
- **Debug token**
  - The debug token is used to enable CM33 and/or the system access port when it is temporarily disabled

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

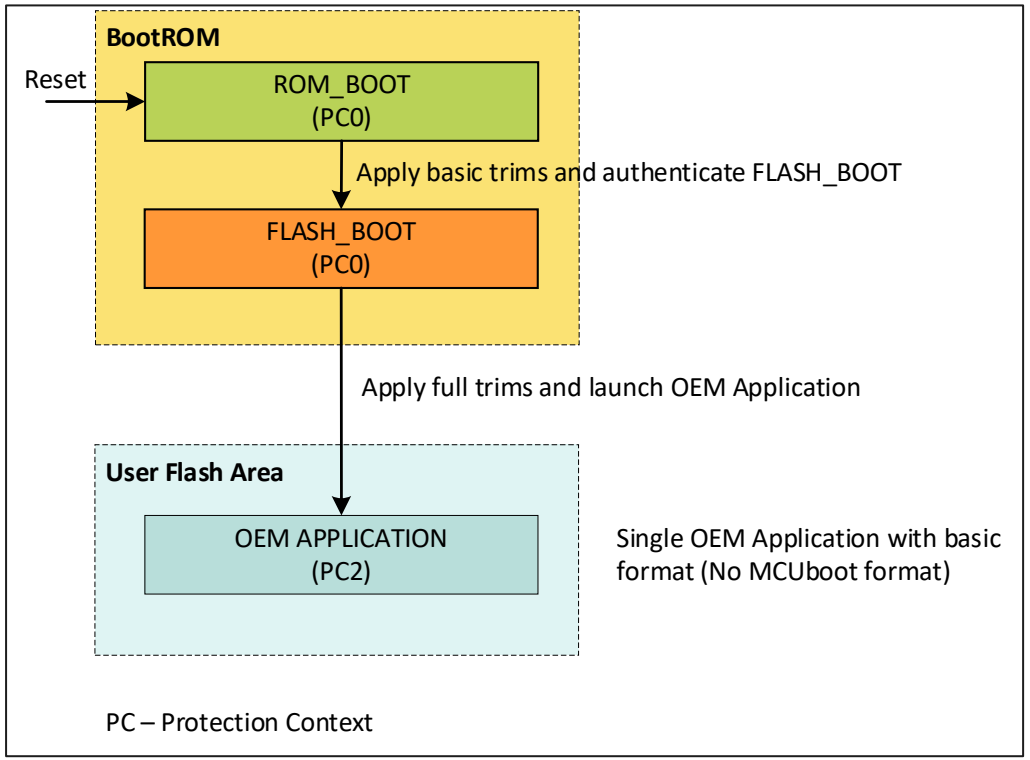
1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	**Boot configurations**	**20**
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Boot configurations

Boot simple application

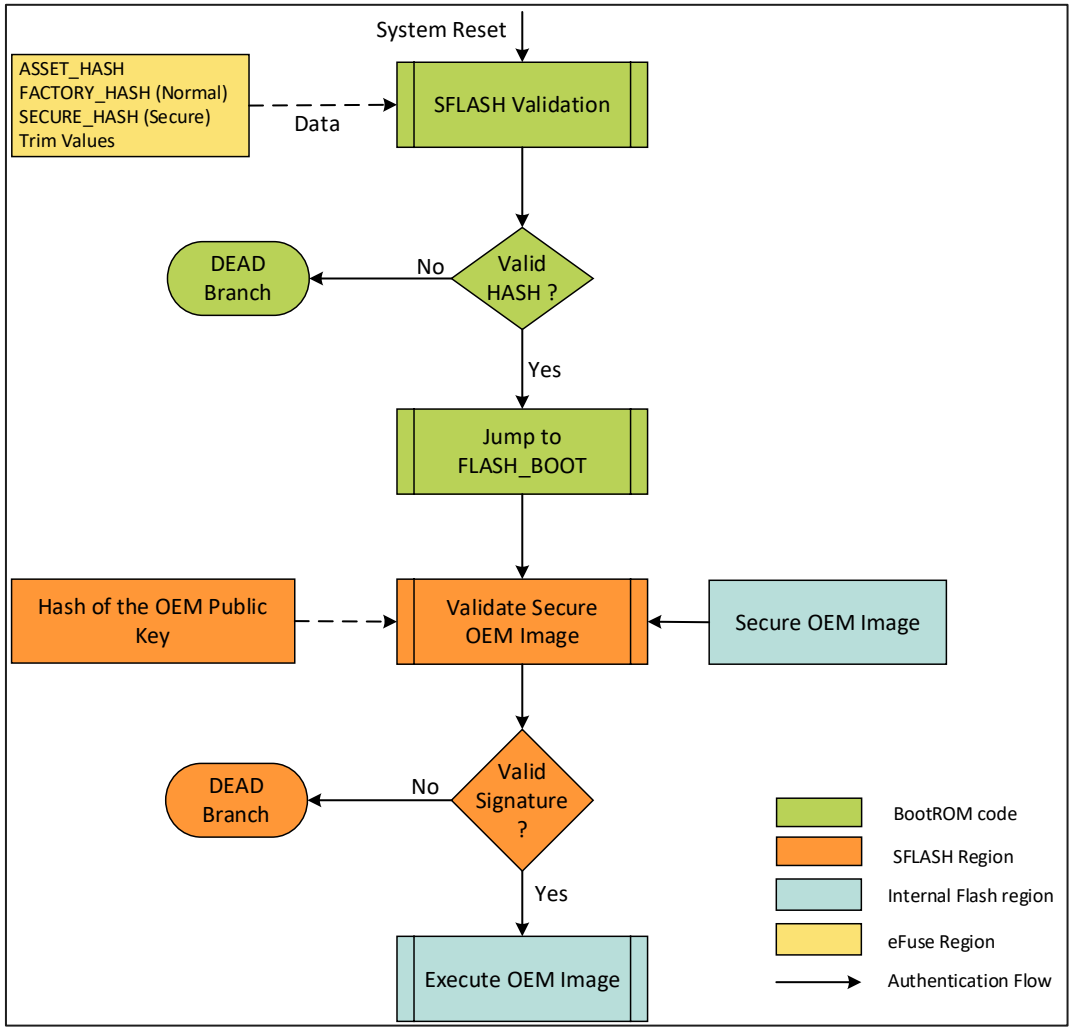
- Boot simple application is the default BootROM configuration
- This configuration is intended for customers who want to develop code without authentication
- The OEM app must be in basic application format, only the interrupt table and code (i.e., no signed format)
- The BootROM launches the OEM application (SPE) without authentication
- The boot simple app configuration does not support the transition to the production LCS



Copyright © Infineon Technologies AG 2026. All rights reserved.

Secure boot sequence

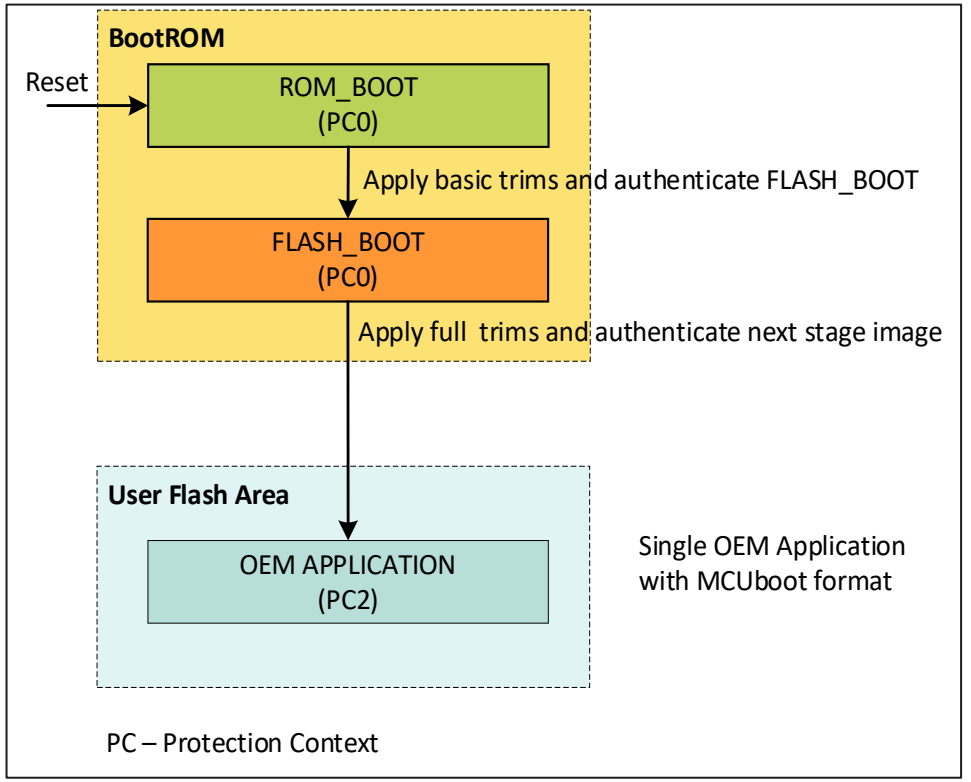
- After reset, ROM_BOOT applies basic trim, configures clocks, and validates the SFLASH
- On successful validation, ROM_BOOT transfers execution to SFLASH_BOOT
- SFLASH_BOOT authenticates the next-stage application (OEM)
- Upon successful authentication, SFLASH_BOOT prepares the environment and launches the OEM application
- If authentication fails, the device transitions to the **Dead** branch



Copyright © Infineon Technologies AG 2026. All rights reserved.

Single-slot boot configuration

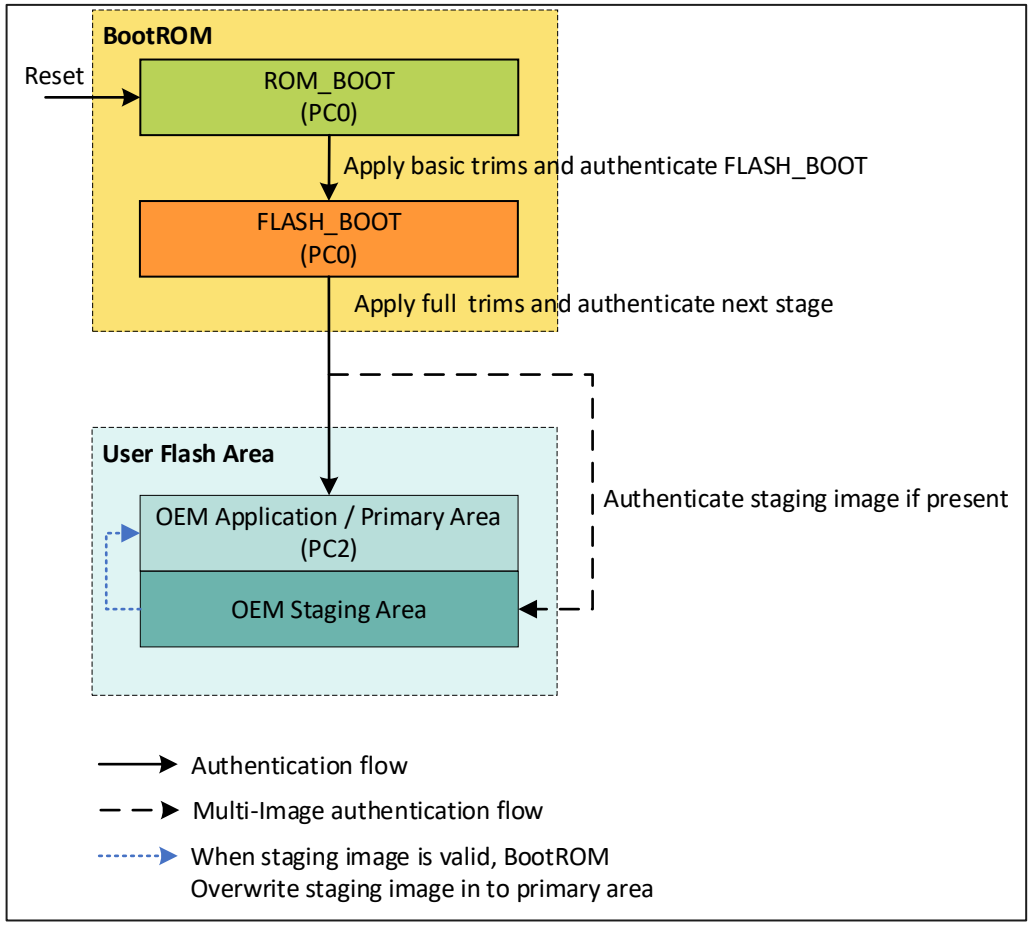
- One OEM application must be present in the user flash region as a signed image
- The BootROM will launch the OEM app when authentication succeeds
- The OEM application start address and size for the application must be specified in the policy during device provisioning
- In the configuration, the BootROM needs to authenticate only one image
- The BootROM authenticates the image before launch, irrespective of whether the device is in development or production LCS



Copyright © Infineon Technologies AG 2026. All rights reserved.

Dual-slot boot configuration

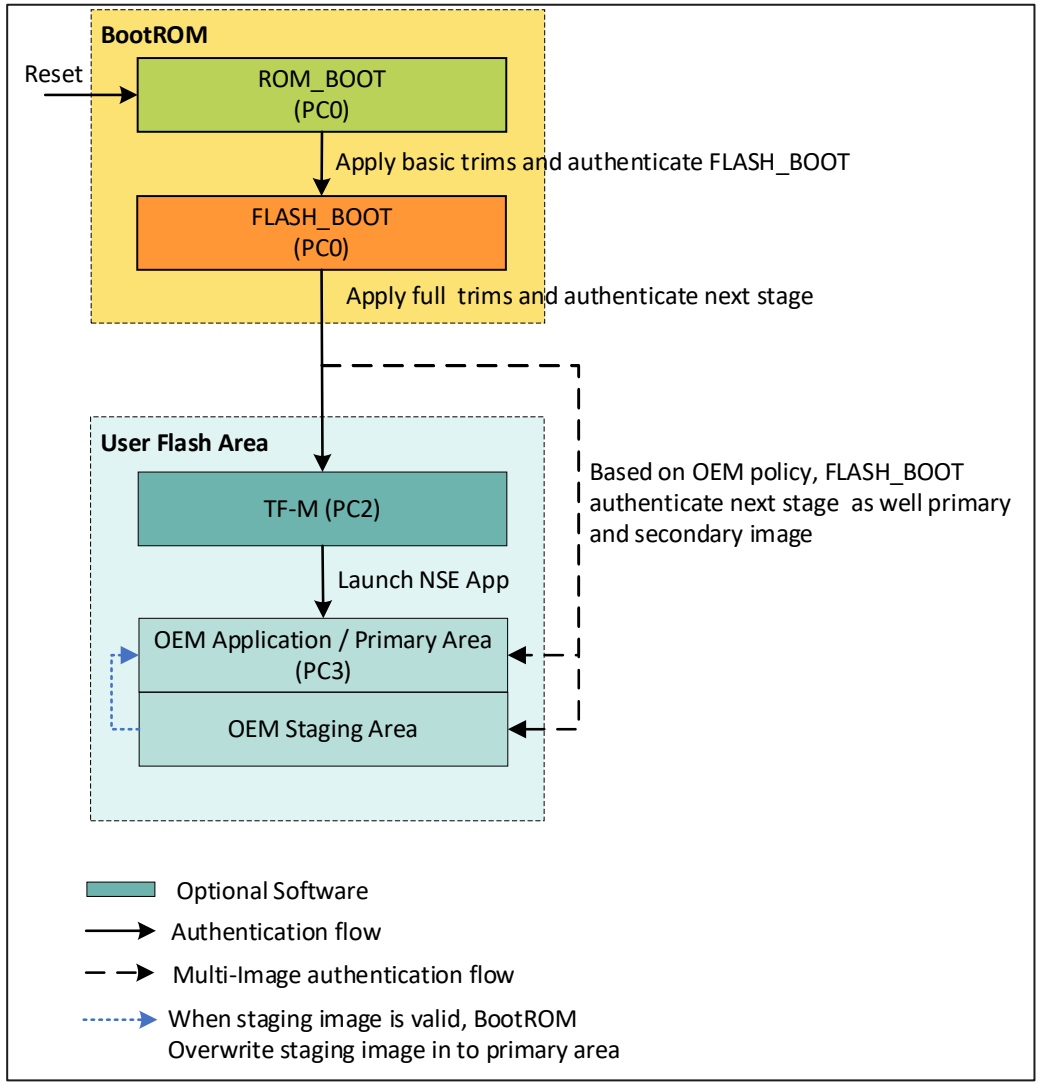
- Two OEM applications must be present in the user flash region
- During device provisioning, the start addresses and sizes of both the primary and secondary OEM application slots must be specified in the provisioning policy
- The primary slot is for boot, and the secondary slot is used for updating the image
- The BootROM authenticates the secondary image (if present) and overwrites it into the primary slot on successful authentication
- Additionally, the BootROM launches the OEM application from the primary location



Copyright © Infineon Technologies AG 2026. All rights reserved.

Three-slot boot configuration

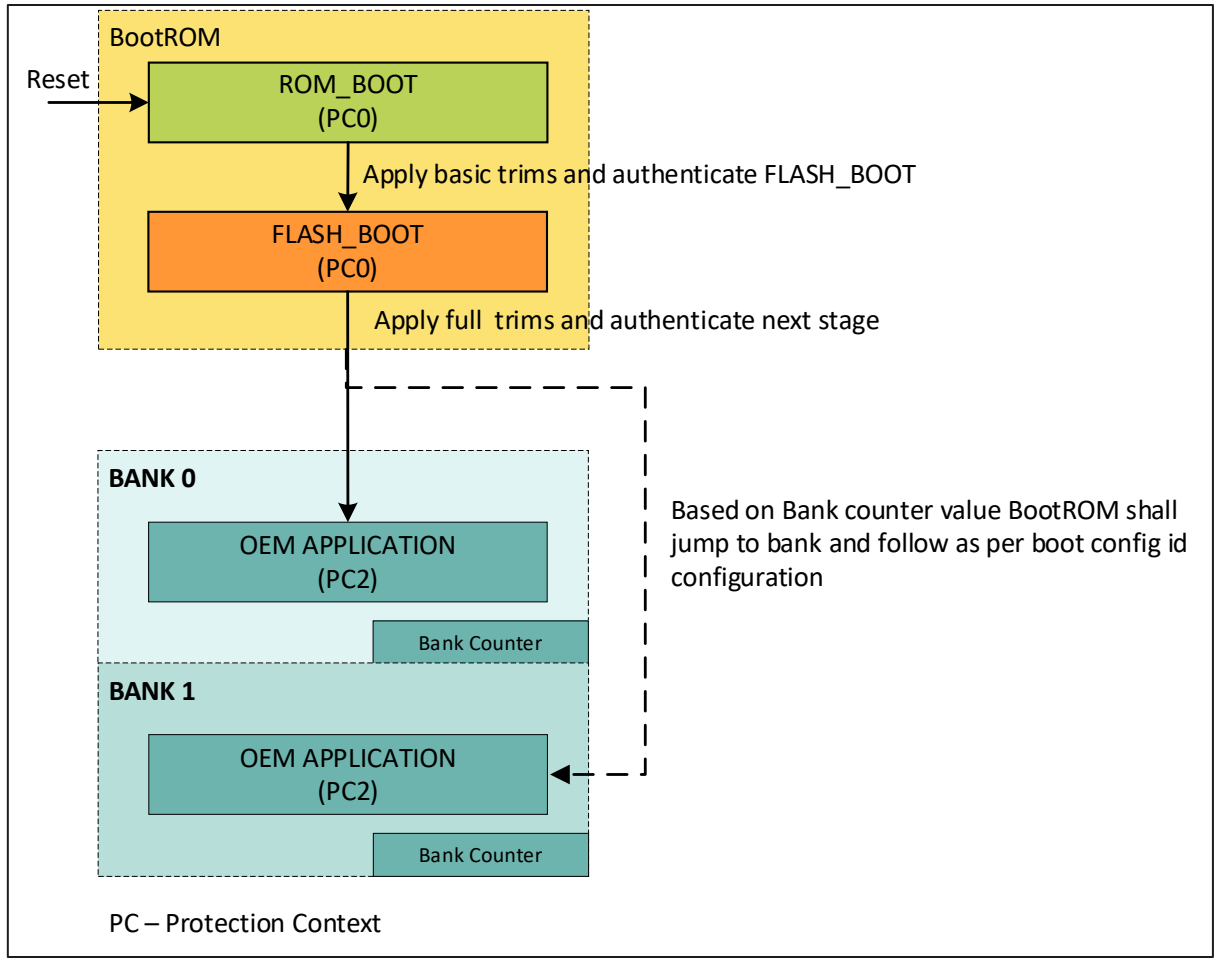
- Three OEM applications must be present in the user flash region
- All three OEM application primary and secondary start addresses and size for the application must be specified in the policy during device provisioning
- The first application (L1) can be either TF-M Secure firmware or a custom secure FW
- Using the multi-image authentication feature, the BootROM will authenticate each stage of the application and launch the first OEM image
- If the L1 image is invalid, or if both the primary and secondary images are invalid, the device transitions into the **Dead** branches



Copyright © Infineon Technologies AG 2026. All rights reserved.

Dual-bank configuration

- The user flash regions are divided into two equal memory regions (128 KB each). Both regions contain a dual-bank counter offset and an OEM application
- The BootROM reads the dual-bank counter offset value and identifies the active bank region
- The BootROM selects the active bank region and executes the application as per the boot policy
- In a normal case, during the firmware upgrade, the new image is loaded to the staging area and moved to the primary area. When the dual bank feature is enabled, this step is not needed, and the BootROM launches the active bank image directly



Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	**Firmware upgrade**	**28**
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

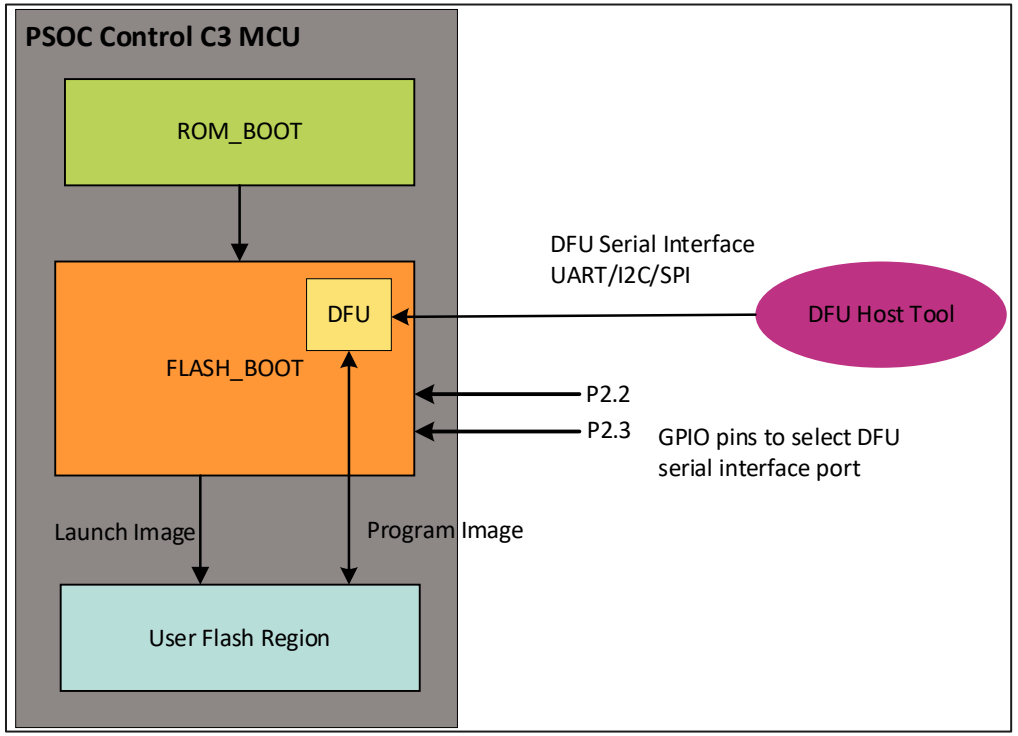
Copyright © Infineon Technologies AG 2026. All rights reserved.

Firmware upgrade

Firmware upgrade

PSOC™ Control C3 series MCU supports two ways of firmware upgrade.

- The BootROM DFU serial interface - disabled by default
- The first method is recommended for factory production programming and upgrades
- The immutable BootROM integrates an in-built Infineon custom serial DFU protocol for firmware upgrade
- The DFU serial interface supports UART, I2C, and SPI ports. The selection of the DFU serial interface port is based on two GPIO pin states: P2.2 and P2.3
- These GPIO pins are scanned during power cycle, reset and not in runtime. BootROM launches DFU when it is enabled through the OEM policy, and based on P2.2, P2.3 GPIO pins state

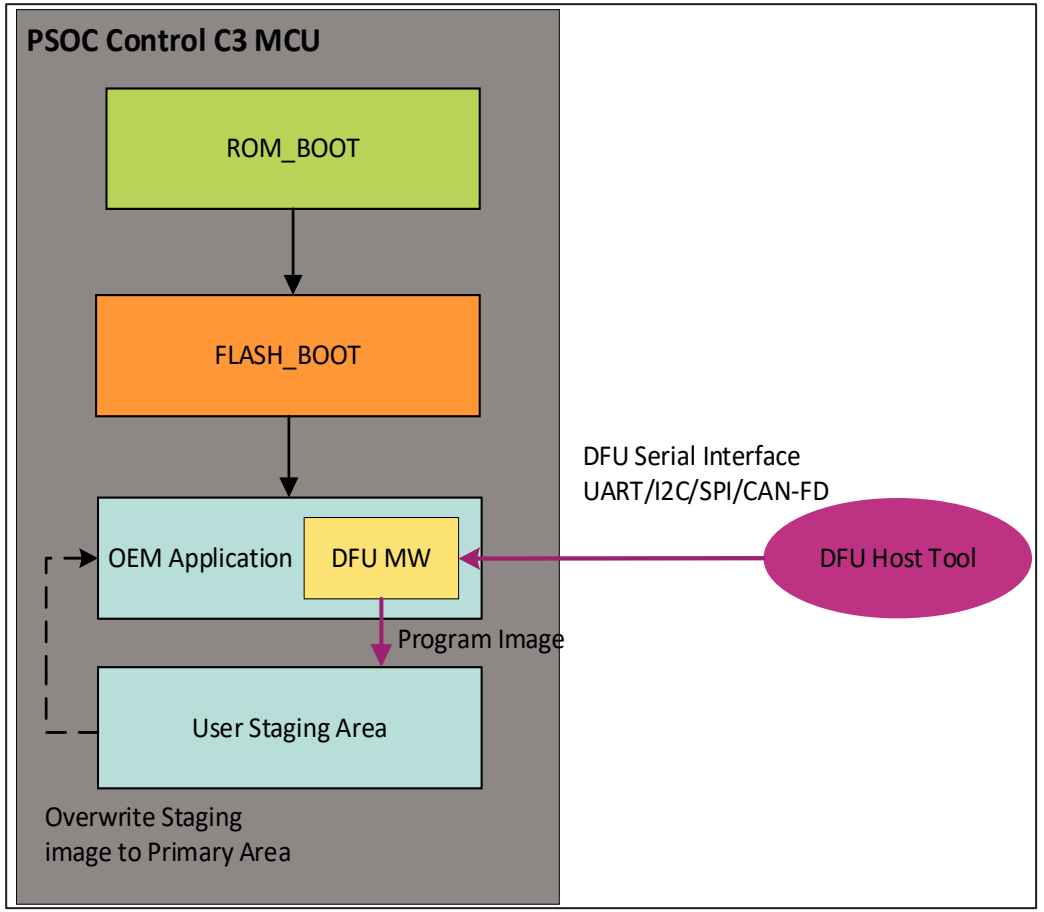


	P2.3 = Vss	P2.3 = Vdd	P2.3 = High-Z
P2.2 = Vss	Serial Disable	UART	Serial Disable
P2.2 = Vdd	I2C	SPI	Serial Disable
P2.2 = High-Z	UART	Serial Disable	Serial Disable

Copyright © Infineon Technologies AG 2026. All rights reserved.

Secure firmware upgrade through DFU middleware

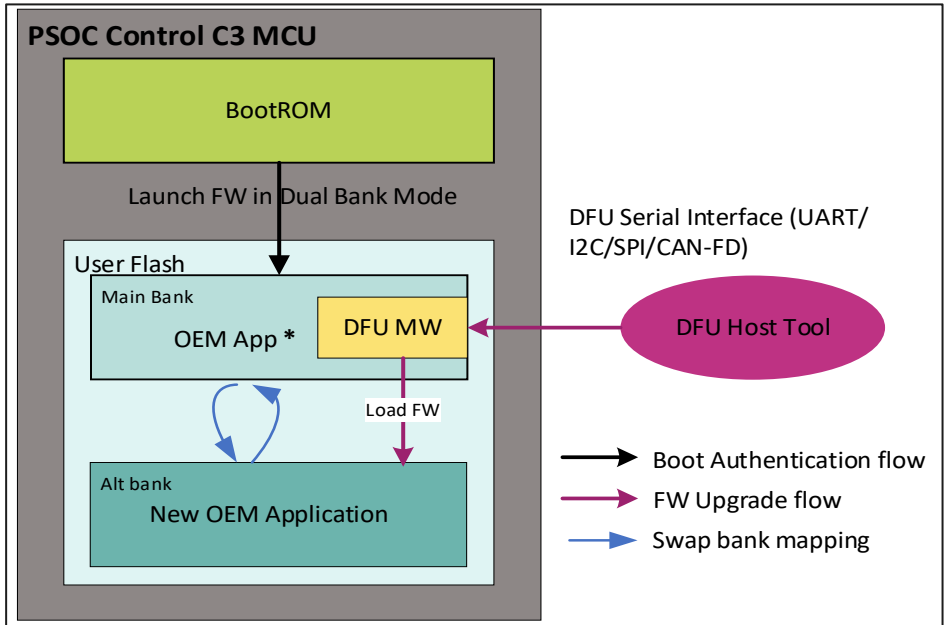
- User application (DFU MW) serial interface
- This method is recommended for field firmware upgrades
- Using this approach, the user can either:
  - Develop their application using the DFU middleware (DFU MW) library provided by Infineon through the ModusToolbox™ library manager
  - Implement their own custom protocol to perform a firmware upgrade using the DFU MW framework
- The DFU middleware supports UART, SPI, I2C, and CAN-FD interfaces
- The user can perform **a live firmware upgrade** using both the dual-bank and DFU middleware features. This means without a reset/less time, the user launches an upgraded new application from another bank



Copyright © Infineon Technologies AG 2026. All rights reserved.

Live firmware upgrade

- PSOC™ C3's flash supports dual bank mode of operation
- In dual-bank mode, the flash is divided into two banks (bank 0 and bank 1) of equal size
- Two mapping modes are available: mapping A and mapping B
  - Mapping A: bank 0 at lower address (main bank) and bank1 at higher (alternate bank) address
  - Mapping B: bank 0 at a higher address (main bank) and bank 1 at a lower address (alternate bank)
- The flash address mapping changes in dual bank mode are as follows:
  - Main bank: 0x1200_0000 – 0x1202_0000 (128 KB)
  - Alternate bank: 0x1280_0000 – 0x1282_0000 (128 KB)
- The dual bank mode is enabled from the OEM policy, and the bank mapping is controlled with a bank counter
- The firmware in the main bank downloads and authenticates the new firmware in the alternate bank, switches mapping mode and launches the new firmware



***OEM App** consists of DFU MW and mbedTLS Lib(for authentication). It does the following operation:

1. Receive new FW image via DFU serial interface and Load in to alternate bank.

2. Validate the signed image using mbedTLS.

3. Change the Flash mapping with SRAM function. Now Alt Bank become new bank.

4. Launch new FW image without reset.

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

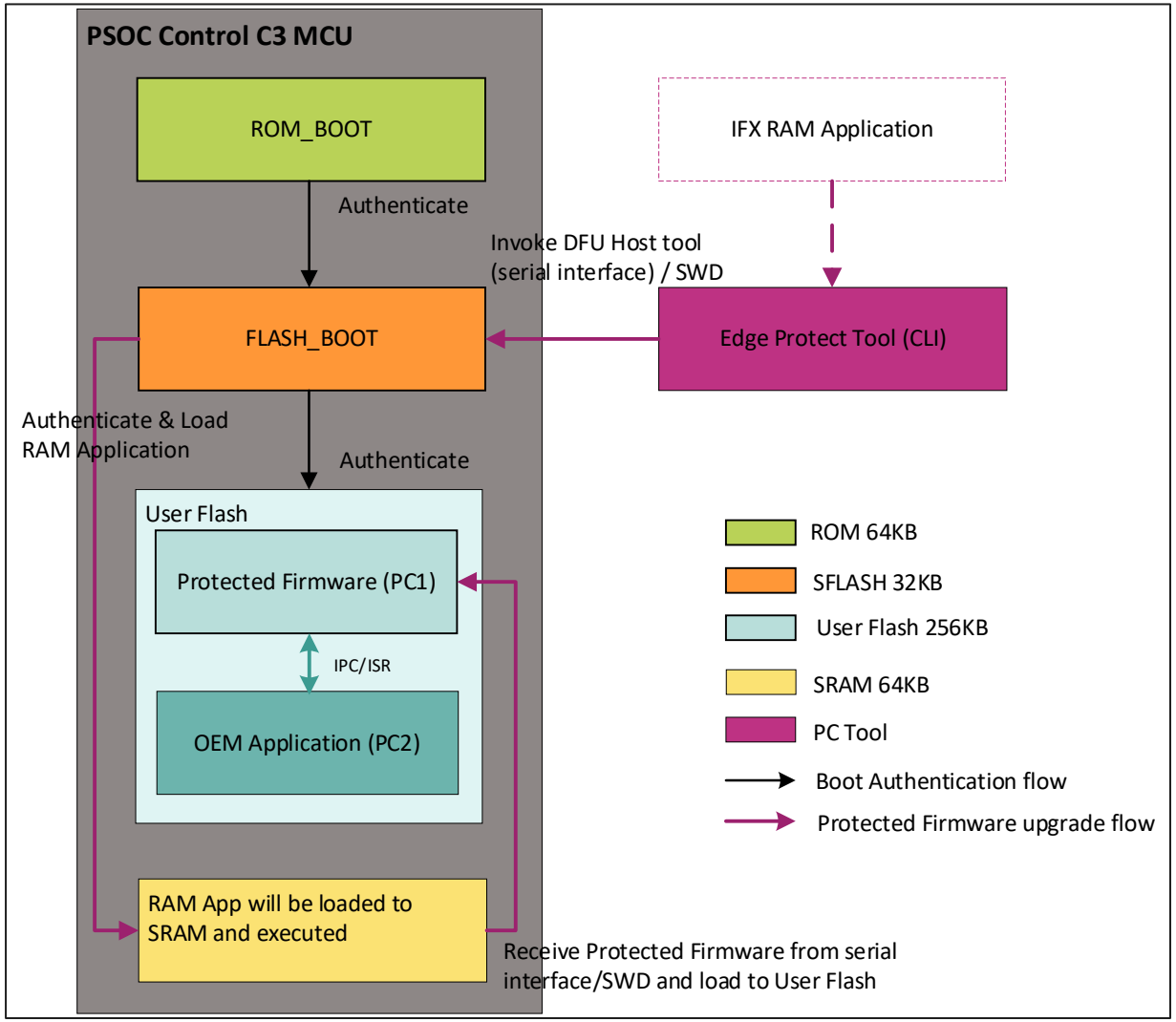
1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	**Protected firmware**	**33**
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Protected firmware

Protected firmware

- Protected firmware (PROT_FW) is developed by Infineon/third-party to do a dedicated algorithm/service based on customer requirements
- PROT_FW flashed in the internal flash, and it is not readable by the customers. It is only accessible through a defined NSC (Non-Secure Callable) interface
- PROT_FW is upgradable in the field with a RAM app that is signed by Infineon
- PROT_FW is authenticated by BootROM and launched in PC=1 SPE context
- OEM application will not have RD/WR access to PROT_FW



Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	**CryptoLite**	**36**
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

CryptoLite

Overview

Key features

- Advanced encryption standard (AES)-128
- Secure hash algorithm (SHA)-256
- True random number generator (TRNG)
- Vector unit (VU) to support asymmetric key cryptography, such as RSA and elliptic curve cryptography (ECC)

Benefits

- Provides hardware implementation and acceleration of cryptographic functions
  - Implementation in hardware takes less time and energy than the equivalent firmware implementation
  - For example: speeds up secure boot digital signature authentication (DSA or ECDSA) through its SHA-256 and VU HW acceleration

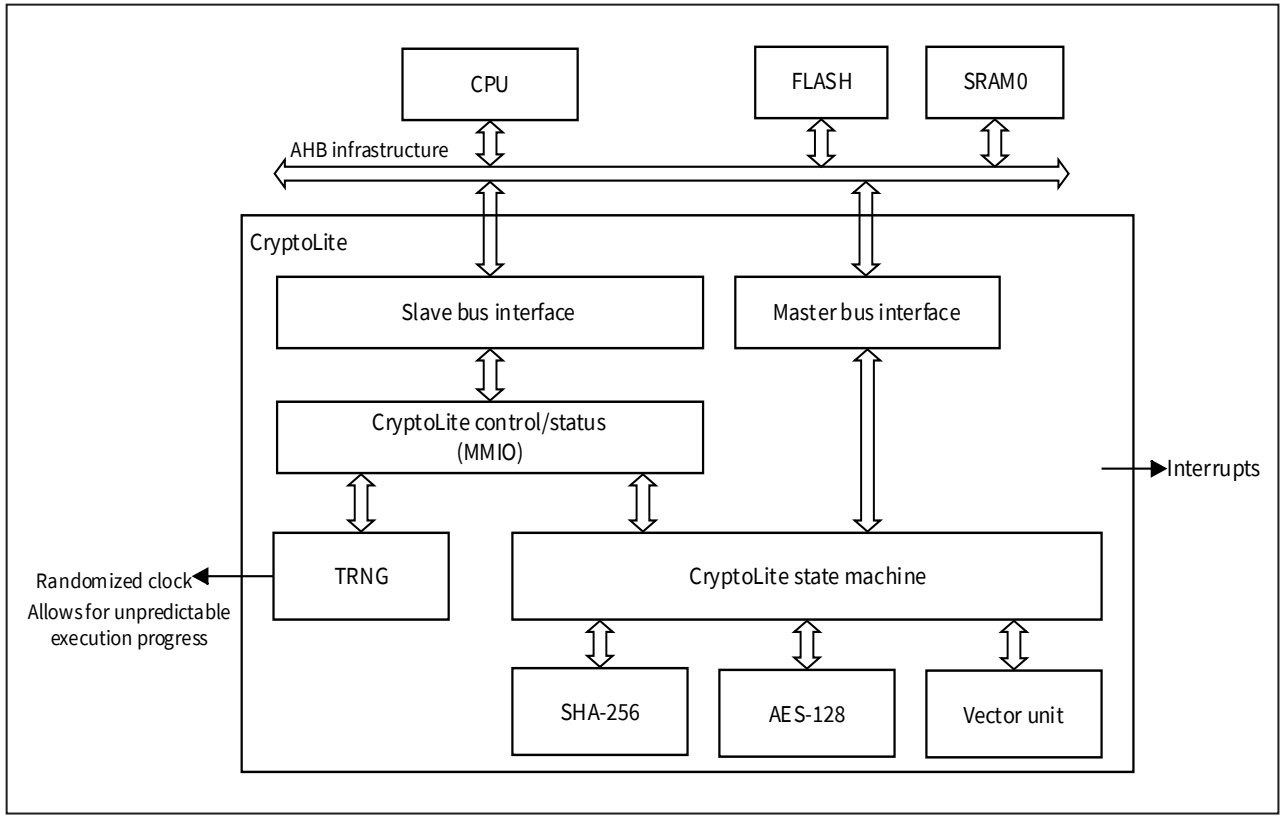
Copyright © Infineon Technologies AG 2026. All rights reserved.

Block diagram

Includes four components

- AES-128 component
  - To accelerate block cipher functionality
  - Encrypt a 128-bit block P (plain-text) with a 128-bit SK (secret key) into a 128-bit block C (cipher-text)

- SHA-256 component
  - To accelerate hash functionality
  - Process function: SHA-256 processing of a 256-bit hash state by performing 64 rounds of a compression function
  - The process inputs are a 256-bit hash state and a message schedule. The process output is an updated 256-bit hash state



Copyright © Infineon Technologies AG 2026. All rights reserved.

Block diagram (contd.)

Vector unit (VU) component

- To accelerate asymmetric key cryptography (for example, RSA and ECC)
- The VU supports long integer operations that are useful for asymmetric key cryptography, supports the following long integer operations: MUL, XMUL, ADD, SUB, XOR, MOV, LSR1, LSL1, LSR, COND_SUB

TRNG component

- Based on a set of ring oscillators, the TRNG includes a hardware health monitor. This component generates true random numbers using up to six ring oscillators
- TRNG generates fixed 32-bit true random numbers
- Has a built-in health monitor that performs tests on the digitised noise source to detect deviations from the intended behaviour

Interrupts

- TRNG interrupt: This interrupt is used to report TRNG behaviour
- Error interrupt: This interrupt is generated for an AHB bus error on the AHB master interface

Copyright © Infineon Technologies AG 2026. All rights reserved.

ModusToolbox™ software – CRYPTO PDL driver

The PDL CryptoLite driver provides APIs to perform hardware-accelerated cryptographic calculations. All functions and declarations for this driver are in cy_cryptolite.h

- PDL AES
  - Supports 128-bit AES encryption in ECB (Electronic Codebook) mode using native CryptoLite hardware
  - Supports 128-bit AES encryption in CBC (Cipher Block Chaining) mode
  - Supports 128-bit encryption and decryption for CFB (Cipher feedback) and CTR (Counter) modes using a combination of software routines and hardware-accelerated ECB mode
- PDL SHA-256
  - Calculates a 256-bit hash and only supports SHA-256 mode
- PDL HMAC-SHA256
  - Calculates a 256-bit hash and only supports SHA-256 mode
- PDL Vector unit (VU)
  - Supports long integer arithmetic and logical operations
  - No native support for RSA and ECC in CryptoLite hardware
  - PDL is implemented by software using VU acceleration. APIs block until the hardware completes the operation, and the buffer/pointer is updated with the result
- PDL RSA
  - Supports up to 4096-bit modulus size; PDL ECDSA support only P256, and P384 curves
- PDL TRNG
  - Supports 32-bit random number generation

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

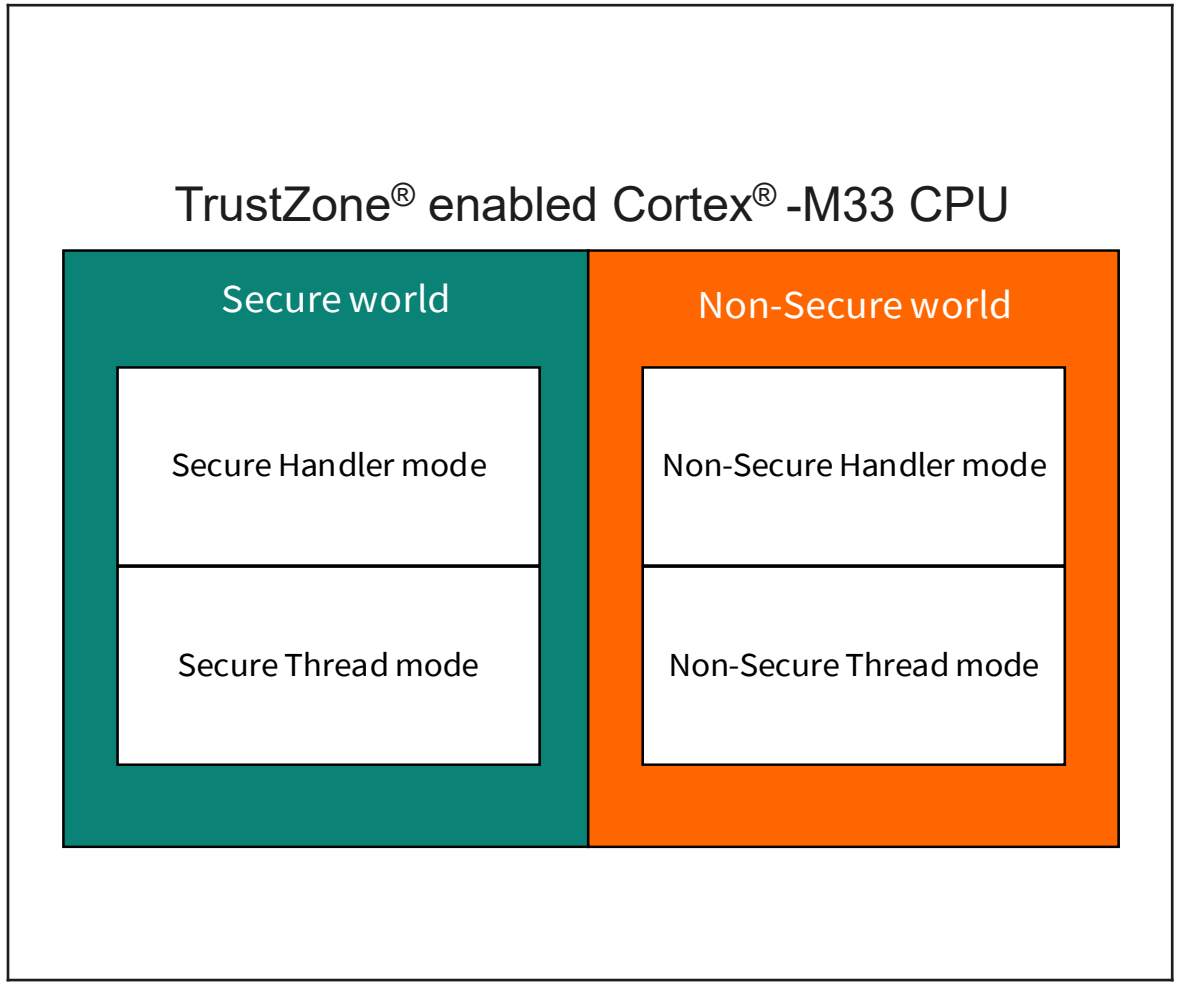
1 Overview 4
2 Key features and capabilities 6
3 Basic boot flow 8
4 Life cycle stage (LCS) 10
5 Device provisioning 15
6 Edge protect tool 18
7 Boot configurations 20
8 Firmware upgrade 28
9 Protected firmware 33
10 CryptoLite 36
11 **TrustZone® and system security architecture** **42**
12 Additional security protection units 53
13 Application architecture 61
14 Code example (CE) 65

Copyright © Infineon Technologies AG 2026. All rights reserved.

TrustZone® and system security architecture

Overview

- Arm® TrustZone® technology is a security extension designed for Armv8-M
- Implements system-wide security by partitioning the system resources into:
  - Secure world (SPE)
  - Non-secure world (NSPE)
- Introduces two more states to the CPU
  - Secure state
  - Non-secure state
- These security states are orthogonal to the existing Thread and handler modes, resulting in four modes:
  - Secure handler
  - Secure thread
  - Non-secure handler
  - Non-secure thread



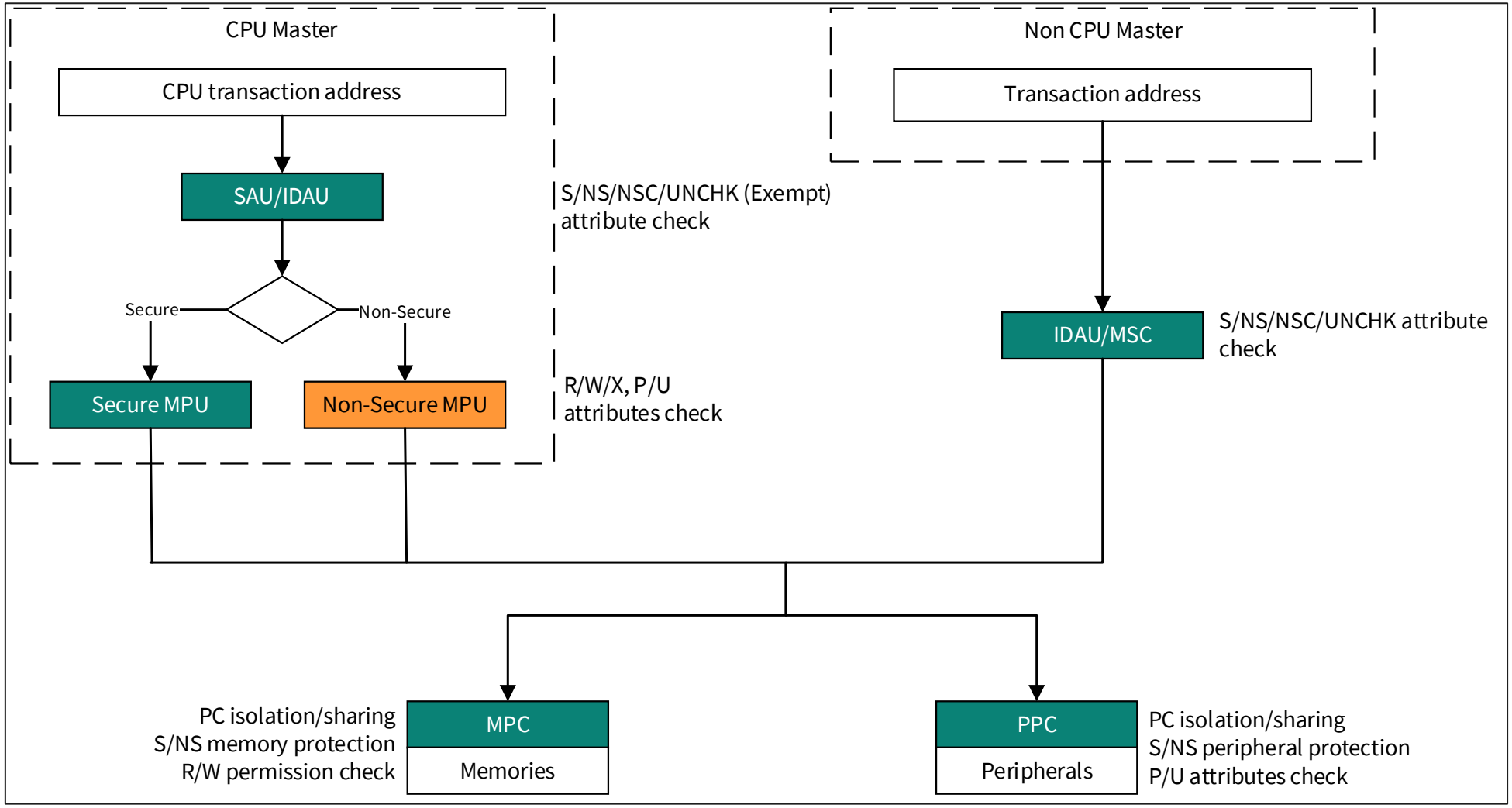
Copyright © Infineon Technologies AG 2026. All rights reserved.

Overview (contd.)

- TrustZone® isolation is based on address space partitioning
  - Attribution units (SAU and IDAU) are used for partitioning the address space
- SAU and IDAU work in conjunction to partition the address space into the following regions:
  - Secure (S)
    - Only accessible in secure software
    - Used for secure code execution and secure peripheral access
  - Non-secure callable (NSC)
    - A special type of secure region used for state transition from non-secure to Secure.
  - Non-secure (NS)
    - Accessible to both secure and non-secure software (read/write access)
    - Used for non-secure code execution and non-secure peripheral access

Copyright © Infineon Technologies AG 2026. All rights reserved.

Security architecture of PSOC™ Control C3



Copyright © Infineon Technologies AG 2026. All rights reserved.

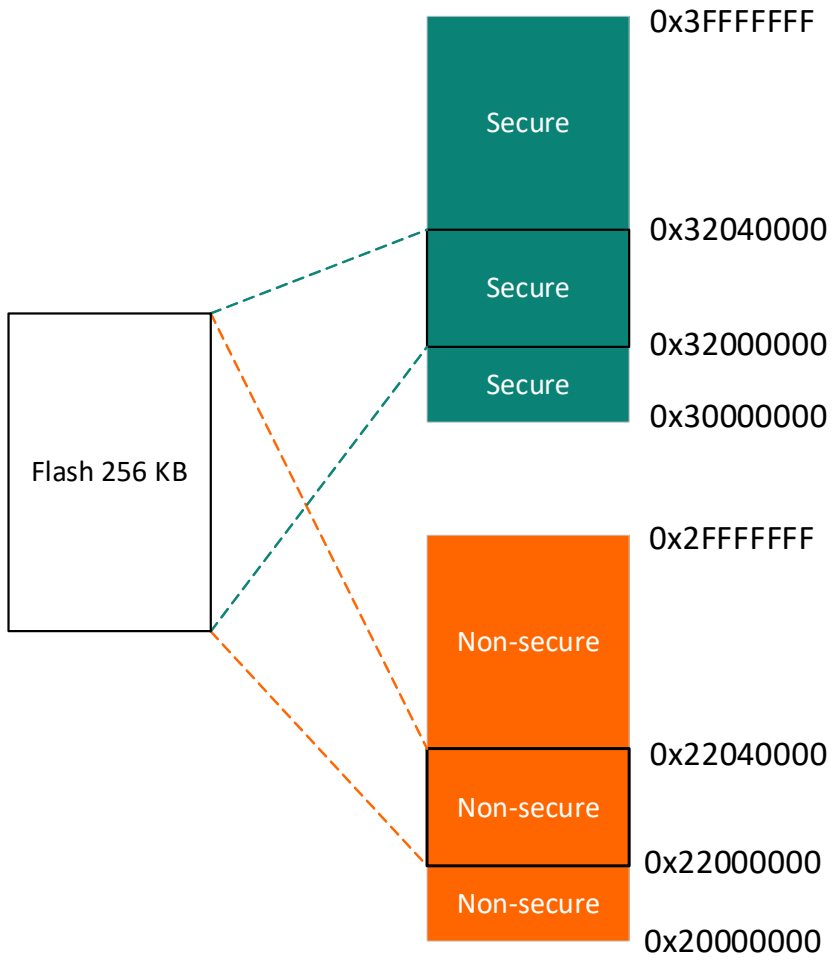
Address space aliasing

- Arm® memory map divides the 4 GB address into multiple regions.
- The memory regions are at 512 MB (code execution, SRAM, peripheral, and so on)
- Each of the 512 MB regions is aliased at the midway, such that the lower 256 MB provides an NS view and the upper 256 MB provides an S view of the same 256 MB region
- For example,
  - 0x3200_0000 is used by S SW for S access to flash
  - 0x2200_0000 is used by NS SW for NS access to flash

Address	Region
0x00000000 - 0x1FFFFFFF	Code
0x20000000 - 0x3FFFFFFF	SRAM
0x40000000 - 0x5FFFFFFF	Peripheral
0x60000000 - 0x7FFFFFFF	RAM
0x80000000 - 0x9FFFFFFF	RAM
0xA0000000 - 0xBFFFFFFF	Device
0xC0000000 - 0xDFFFFFFF	Device
0xE0000000 - 0xE00FFFFF	System PPB
0xE0100000 - 0xFFFFFFFF	System Vendor_SYS

Copyright © Infineon Technologies AG 2026. All rights reserved.

Address space aliasing (contd.)



Copyright © Infineon Technologies AG 2026. All rights reserved.

Attribution units

- The Address space is partitioned by two attribution units
  - Implementation-defined attribution unit (IDAU)
  - Secure attribution unit (SAU)
  - The attribution units indicate if an address is Secure or Non-secure. The higher security level among the two is used to indicate an address is S or NS

IDAU	SAU	Result
S	X	S
X	S	S
NSC or NS	NSC	NSC
NSC	NSC or NS	NSC
NS	NS	NS

Copyright © Infineon Technologies AG 2026. All rights reserved.

IDAU and SAU

Implementation-defined attribution unit (IDAU)

- IDAU provides a fixed address partitioning scheme
- IDAU uses bit 28 of the address to indicate whether the address is S or NS, which aligns with 256 MB aliasing boundaries
  - If bit 28 is set to ‘0’, the address is NS
  - If bit 28 is set to ‘1’, the address is S
- The system address region (0xE000_0000 – 0xE010_0000) is marked as exempt from security check by IDAU
- We have three IDAUS. By default, the implementation is by Arm®

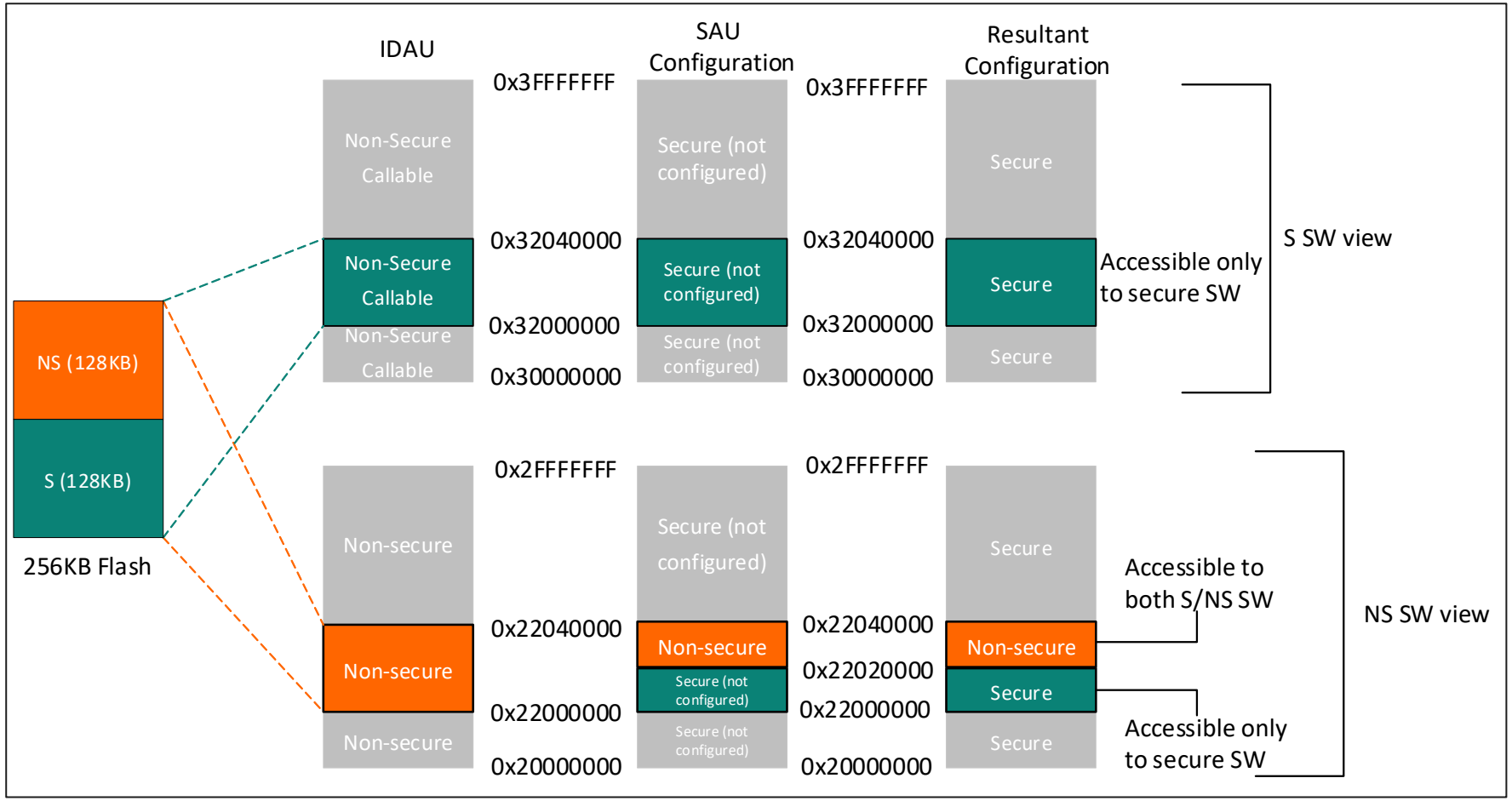
Secure attribution unit (SAU)

- SAU is a programmable attribution unit and part of the CPU
- Eight programmable SAU regions are available; each region can be configured independently
- By default, a region not covered by SAU is marked as **Secure**
- The region start/end address and the NS/NSC attribute can be programmed only by secure software

Copyright © Infineon Technologies AG 2026. All rights reserved.

IDAU and SAU (contd.)

For example,



Copyright © Infineon Technologies AG 2026. All rights reserved.

State transition

– The state transition is possible in following three scenarios:

S SW calls NS SW

– A CPU in secure state can execute instructions only from secure SW

– If the S SW calls NS SW with special BXNS or BLXNS branch instruction, the CPU state switches to NS

NS SW calls S SW

– A CPU in NS state can execute instruction only from non-secure SW

– A transition to secure state is possible through entry points in NSC regions

– If SG instruction is the first instruction executed in NSC region, the CPU state changes to S

Exception/Interrupt

– If CPU is running NS SW and a S exception takes place, the state changes to S

– If a NS exception occurs while running S SW, the CPU state changes to NS

– If CPU state and exception target state is same, the CPU state does not change

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	**Additional security protection units**	**53**
13	Application architecture	61
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Additional security protection units

Protection context

- Protection context (PC) is an Infineon proprietary additional security layer for isolation
- PC adds pseudo states to bus masters, an additional attribute on top of S/NS and Privi/Unpriv
- PSOC™ C3 has eight PCs. Each PC is isolated from one another
- To isolate the FW layers on PSOC™ C3, they execute in a different PC

PC	FW layer
PC0	Integrated Boot ROM
PC1	Protected FW
PC2	S SW/EPB/TF-M (SPE) + NSPE
PC3-PC6	User defined
PC7	Debugger (Sys - AP)

Copyright © Infineon Technologies AG 2026. All rights reserved.

Memory protection controller (MPC)

- If an application requires memory to be partitioned into multiple sections, SAU alone might not be enough
- MPCs are protection units that allow memories to be partitioned at a more granular level
- MPC divides the SRAM and flash into fixed-sized 2 KB blocks
- For each block, independent read and write permission can be controlled by restricting the access to this block using S/NS and PC access attributes
- If the region is configured as S, NS transactions are blocked. If the region is configured as NS, S transactions are blocked
- Similarly, if the PC of bus masters does not match with blocks attribute, the transactions are blocked.
- If an attribute mismatch (s/ns and pc) is observed by MPC, it triggers an error response. The error response can be configured as read zero write ignore (RZWI) or bus fault
- In PSOC™ C3, the MPC is configured through the OEM policy, and the configuration is applied by the integrated boot ROM
- The MPC config cannot be configured at runtime

Copyright © Infineon Technologies AG 2026. All rights reserved.

Example for memory access

CPU state	Data/ execute	IDAU	SAU	MPC			PC	Resultant security
				R	W	NS		
S	Data or execute	S	S	1	1	S	All PCs	S: RW access allowed for all PCs
NS	Data	NS	NS	1	0	NS	PC2,3	NS: Read access allowed only for PC2,3. NS: Write access blocked for PC2,3
S	Execute	NS	NS	1	1	NS	PC2,3	All NS R/W access is blocked for all PCs by TrustZone®
S	Data or Execute	S	S	1	1	NS	PC2,3	S: R/W access is blocked by MPC for PC2,3.
NS	Data	NSC	S (or NSC)	1	1	S	All PCs	NS: R/W access is blocked for all PCs by TrustZone®
NS	Execute	NSC	NSC	1	1	S	All PCs	NS: Executes only SG instruction for all PCs

Copyright © Infineon Technologies AG 2026. All rights reserved.

Peripheral protection controller (PPC)

- The PPC is a TrustZone® compliant protection unit that helps in isolating peripherals between secure and non-secure worlds
- It divides the peripheral MMIO info fixed sized regions (PPC region)
- A PPC region consists of peripheral MMIO registers of the same peripheral
- There are two types of PPC regions
  - Security Aware
  - Security Unaware
- For security-unaware regions, if a PPC region is configured as S, NS transactions are blocked. If the region is configured as NS, S transactions are blocked
- For security-aware regions, if a PPC region is configured as S, NS transactions are blocked. If the region is configured as NS, both S and NS are allowed
- If the PC of the bus masters does not match the PPC region attribute, the transactions are blocked
- Additionally, Priv/Unpriv attributes of the bus master and the PPC region are also verified by PPC
- If an attribute mismatch between the bus master and region is observed by PPC, an error response is triggered. The error response can be configured as RZWI or a bus fault
- Unlike MPC, PPC is configured at run-time through the secure code

Copyright © Infineon Technologies AG 2026. All rights reserved.

Example for peripheral access

CPU state	CPU Priv	Security Aware	IDAU	SAU	PPC			Resultant security
					S/NS	PC Mask	Priv	
S	0	No	S	S	S	all PCs	1	S: Access allowed for all PCs
NS	1	No	NS	NS	NS	PC2,3	0	NS: Access blocked due to CPU in Unprivileged State
S	0	No	NS	NS	NS	PC2,3	1	NS: Access allowed only for PC2,3
S	0	No	S	S	NS	PC2,3	1	S: Access blocked for PC2,3
S	0	Yes	S	S	NS	PC2,3	1	S: Access allowed for PC2,3
NS	1	Yes/No	NS	NS	NS	PC2,3	1	NS: Access allowed for PC2,3

Copyright © Infineon Technologies AG 2026. All rights reserved.

MPU and MSC

Memory protection unit (MPU)

- The MPU is a programmable protection unit, and it is part of the CPU
- MPU controls the execute/read/write permission for privileged and unprivileged software
- In PSOC™ C3, the Cortex® -M33 CPU has S and NS MPUs. Each MPU has eight programmable regions
- The secure and non-secure MPUs are banked between security states
- Access violation detected by MPU triggers MemManage Fault

Master security controller (MSC)

- The MSC is an adapter used to add TrustZone® capability to masters without a security extension
- The MSC is connected to IDAU and performs the security check
- If an attribute mismatch is found, it can result in RZWI or MSC interrupt

Copyright © Infineon Technologies AG 2026. All rights reserved.

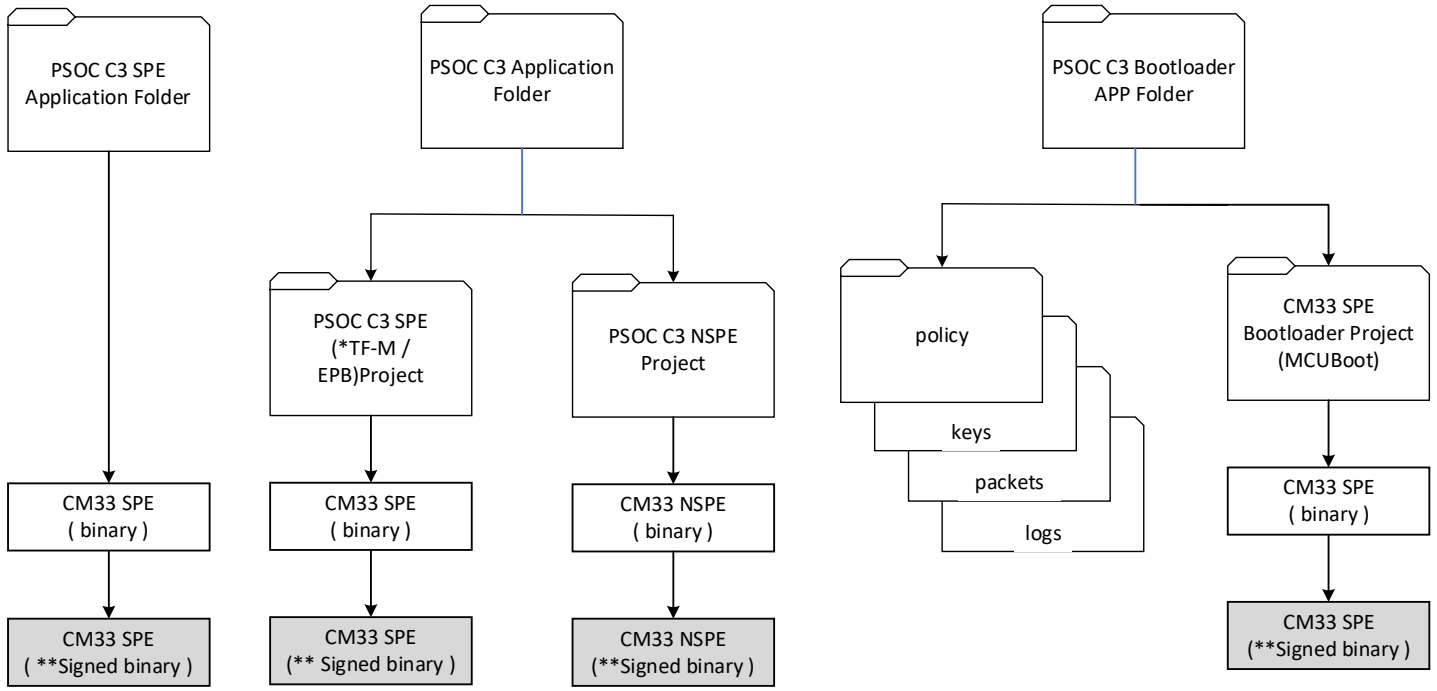
Table of contents

1	Overview	4
2	Key features and capabilities	6
3	Basic boot flow	8
4	Life cycle stage (LCS)	10
5	Device provisioning	15
6	Edge protect tool	18
7	Boot configurations	20
8	Firmware upgrade	28
9	Protected firmware	33
10	CryptoLite	36
11	TrustZone® and system security architecture	42
12	Additional security protection units	53
13	**Application architecture**	**61**
14	Code example (CE)	65

Copyright © Infineon Technologies AG 2026. All rights reserved.

Application architecture

Application structure



(a) - Default App structure

(b) SPE + NSPE App structure

(c) Edge Protect Bootloader App structure

EPB - Edge Protect Bootloader
SPE - Secure Processing Environment
NSPE - Non-Secure Processing Environment

** - Project signing is optional
* - TF-M / PROT_FW are the optional application which can be reside in SPE directory based on provisioning policy configuration

Note:
- If EPB not present, policy file dictate the user application signing.
- If EPB preset, it shall decide the application signing

Copyright © Infineon Technologies AG 2026. All rights reserved.

Application structure (contd.)

- This section will show a typical folder structure for an application
  - The term application refers to one or more projects that run on this CPU

- Two projects associated with it:
  - A Secure Processing Environment (SPE)
  - Non-Secure Processing Environment (NSPE) project
  - It is optional to support NSPE project if needed

- Each application may contain four additional folders that are generated when the Edge Protect Tools are used to initialize the application for provisioning. These folders include:
  - policy
  - Keys
  - Packets
  - logs
  - Folder contains the default policy files, the generated or stored keys, packets generated by the tools, and log files generated each time you execute the Edge Protect Tools

Copyright © Infineon Technologies AG 2026. All rights reserved.

Table of contents

1 Overview 4
2 Key features and capabilities 6
3 Basic boot flow 8
4 Life cycle stage (LCS) 10
5 Device provisioning 15
6 Edge protect tool 18
7 Boot configurations 20
8 Firmware upgrade 28
9 Protected firmware 33
10 CryptoLite 36
11 TrustZone® and system security architecture 42
12 Additional security protection units 53
13 Application architecture 61
14 **Code example (CE)** **65**

Copyright © Infineon Technologies AG 2026. All rights reserved.

Code example (CE)

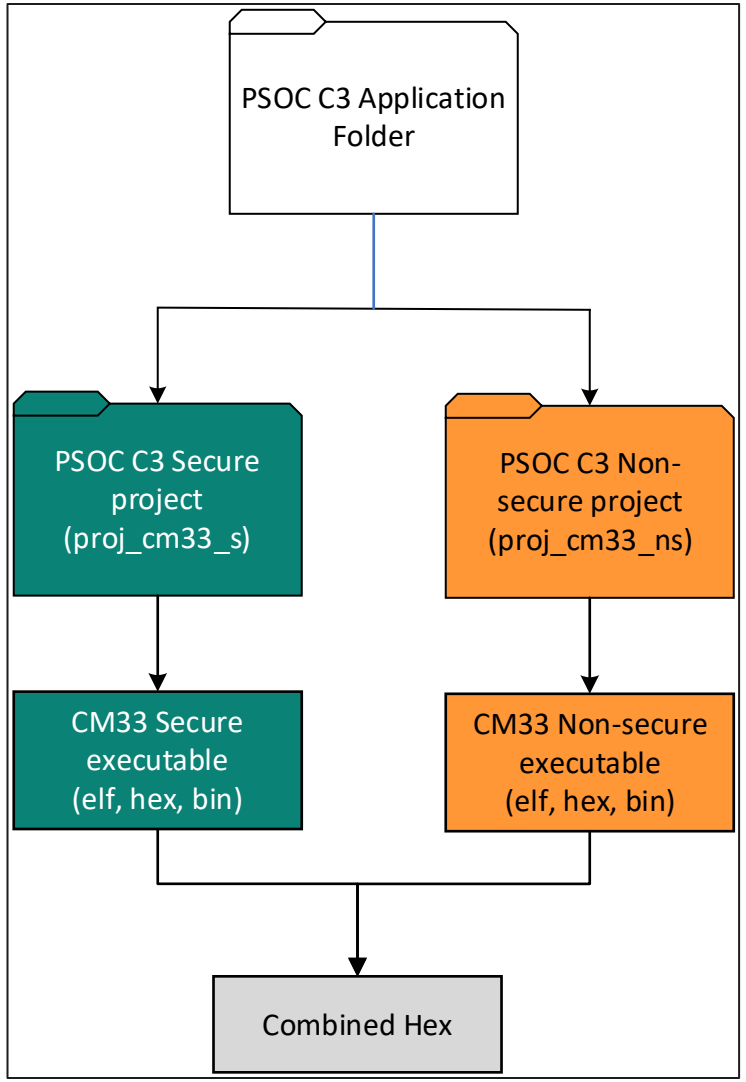
Non-secure app template

- A basic starter S+NS template application is available in Early Access (EA) pack
- On reset, the integrated boot ROM executes first and launches the Secure project
- This secure application sets up the device security and launches the NS project
- The NS project is the starting point for developing any Non-secure code example
- Scope of code example
  - Use ModusToolbox™ (v3.3) to build a basic secure and non-secure project
  - Provision the device with OEM policy
  - Program and Debug the S+NS application

Copyright © Infineon Technologies AG 2026. All rights reserved.

Application structure

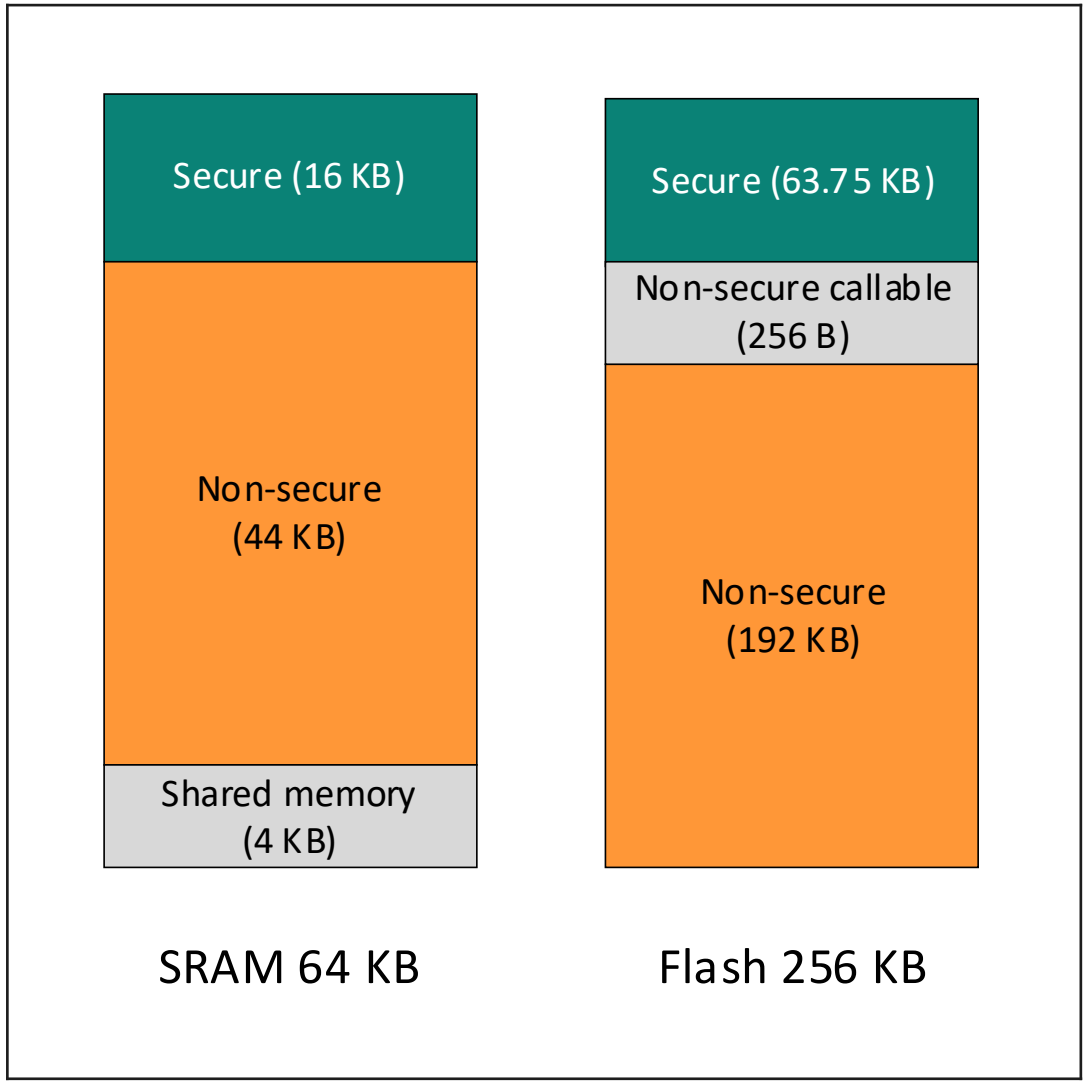
- The application consists of two projects
  - proj_cm33_s project
  - proj_cm33_ns project
- The proj_cm33_s project runs in secure PC2 state while the proj_cm33_ns project runs in Non-secure PC3 state
- Each project is built independently to generate secure and non-secure executable files
- ModusToolbox™ combines the hex files of both the projects to generate a single hex file containing both projects which can be programmed in the device



Copyright © Infineon Technologies AG 2026. All rights reserved.

Memory map

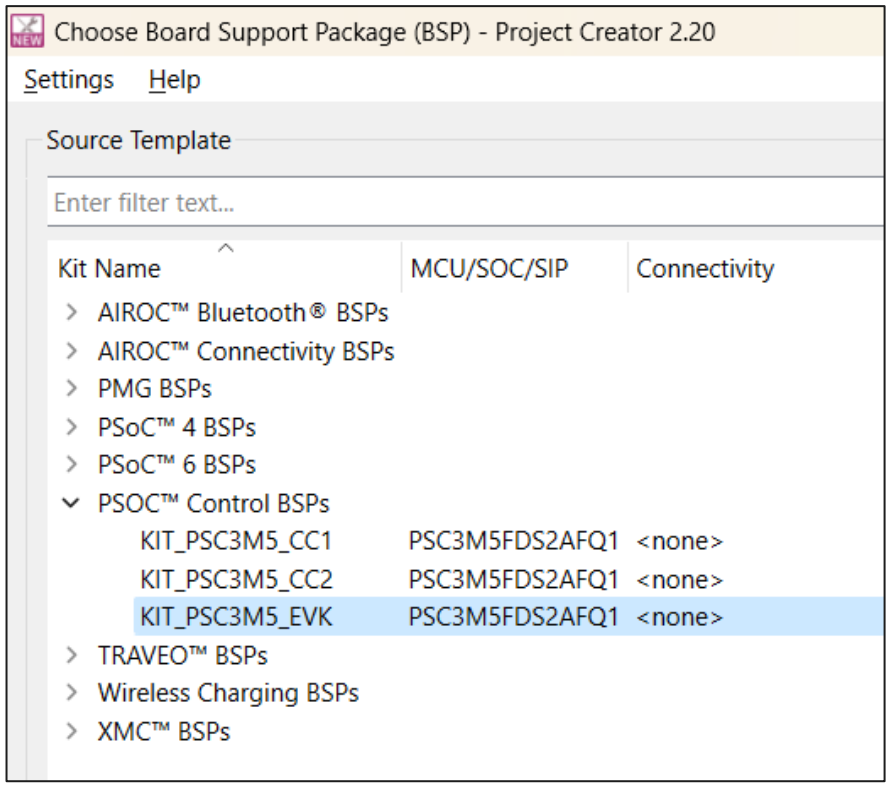
- The 64 KB RAM is divided into three regions
  - Secure – 16 KB
  - Non-secure – 44 KB
  - Shared memory between Secure and Non-secure projects – 4 KB
- The 256 KB flash is partitioned into three regions
  - Secure – 63.75 KB
  - Non-secure Callable – 256 Bytes
  - Non-secure – 192 KB
- The SAU and MPC configuration must be set up by in such a way that it aligns with this memory map

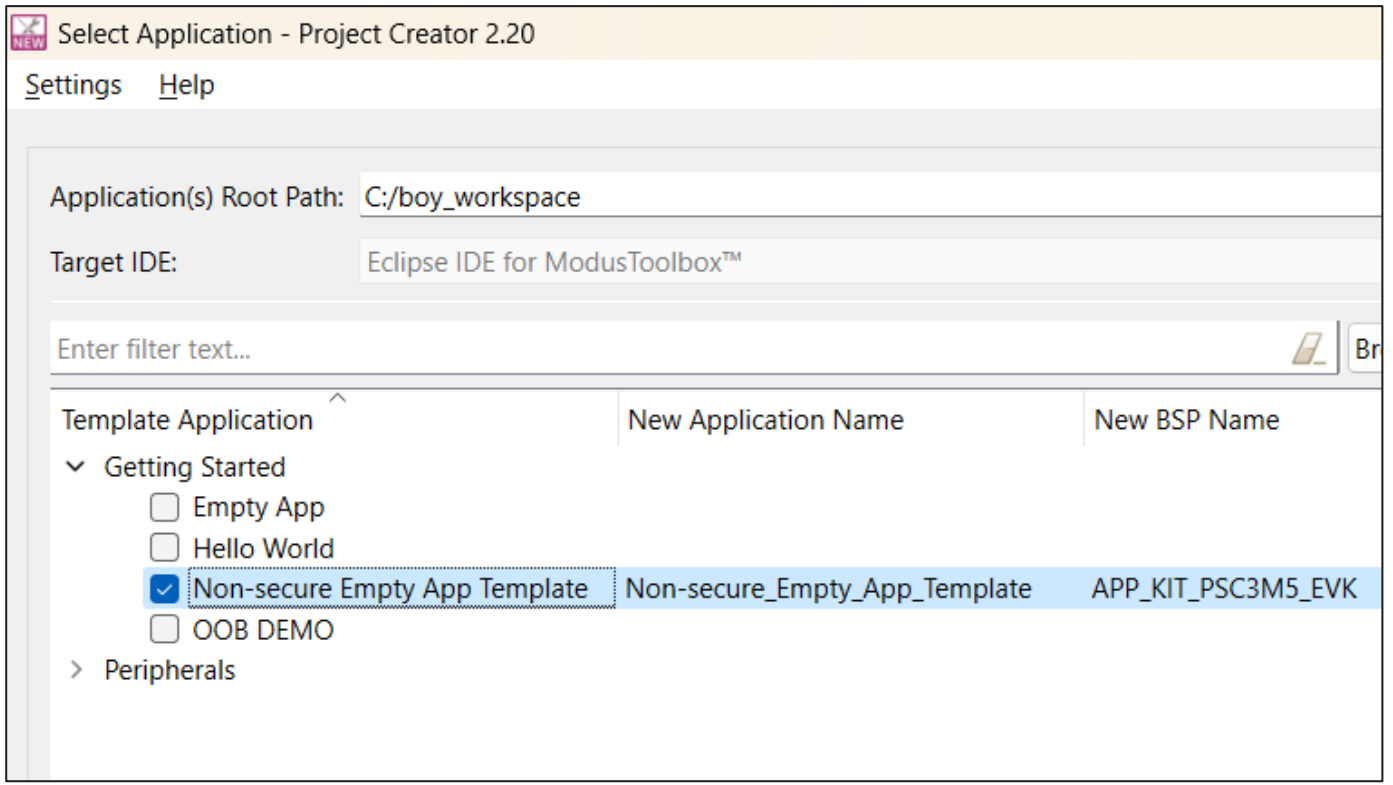


Copyright © Infineon Technologies AG 2026. All rights reserved.

Create a new application

- Open ModusToolbox™ software, select a workspace of your choice
- Select the KIT_PSC3M5_EVK board support package (BSP)
- Choose the Non-secure empty app Template application





Copyright © Infineon Technologies AG 2026. All rights reserved.

Device provisioning

- To program and execute a non-secure project, the MPC needs to be configured such that it aligns with the memory map of code example. Refer to the code example (CE) memory map mentioned in the previous slide
- The MPC can be configured by provisioning the device with OEM policy
- Follow the below steps to provisioning the device:
  - Add EdgeProtectTools executable in the *PATH* environment variable. It is located at

C:\Users\<USERNAME>\ModusToolbox\packs\ModusToolbox-PSoC-C3-Early-Access-Pack\tools\edgeprotecttools\bin

  - Open command prompt or ModusShell and go to the S + NS Application directory *(<Workspace>\Non-secure_Empty_App_Template* directory)
  - An OEM policy which has MPC config aligned with code example memory map is provided with code example in *Non-secure_Empty_App_Template\ns_policy* directory
  - Run the following command to provision the device with this policy

```
- edgeprotecttools -t psoc_c3 init
- edgeprotecttools create-key --key-type ECDSA-P256 -o keys/oem_rot_priv_key_0.pem keys/oem_rot_pub_key_0.pem
- edgeprotecttools create-key --key-type ECDSA-P256 -o keys/oem_rot_priv_key_1.pem keys/oem_rot_pub_key_1.pem
- edgeprotecttools -t psoc_c3 provision-device -p ns_policy/policy_oem_provisioning.json
```


Copyright © Infineon Technologies AG 2026. All rights reserved.

MPC configuration in OEM policy

1. Set SRAM from 16 KB to 60 KB as NS for PC2,3
2. Set Last 4 KB of SRAM as shared memory, i.e., secure for PC2 and Non-secure for PC3
3. Set flash from 64 KB to 256 KB as non-secure

Note: Regions not covered by this configuration are set as secure by default

– SRAM MPC config

```
"mpc_struct_ram": {
    "description": "RAM MPC structures. The total number of RAM and Flash MPC structures must be less than 32",
    "value": [
        {
            "description": "Offset and size in KB. The values must be multiple of 2 (e.g. 0 KB, 2 KB, 4 KB, 8 KB, ...)",
            "offset": "16 KB",
            "size": "44 KB",
            "attr": {
                "description": "The MPC attributes for this memory region",
                "comment_1": "WRN allow write, read, and NS only access",
                "comment_2": "-RN allow read, NS-only access",
                "comment_3": "--- and --N, no access",
                "comment_4": "WR- read and write access, but NS=0 only",
                "value": {
                    "pc0": "WR-",
                    "pc1": "WR-",
                    "pc2": "WRN",
                    "pc3": "WRN",
                    "pc4": "--N",
                    "pc5": "--N",
                    "pc6": "--N",
                    "pc7": "--N"
                }
            }
        },
```

– 1

```
        {
            "description": "Offset and size are indexes where 1 equals 2 KB (e.g. 3 = 6 KB)",
            "offset": "60 KB",
            "size": "4 KB",
            "attr": {
                "description": "The MPC attributes for this memory region",
                "comment_1": "RWN allow read, write, and NS only access",
                "comment_2": "R-N allow read, NS-only access",
                "comment_3": "--- and --N, no access",
                "comment_4": "RW- read and write access, but NS=0 only",
                "value": {
                    "pc0": "WR-",
                    "pc1": "WR-",
                    "pc2": "WR-",
                    "pc3": "WRN",
                    "pc4": "--N",
                    "pc5": "--N",
                    "pc6": "--N",
                    "pc7": "--N"
                }
            }
        }
```

– 2

– Flash MPC config

```
"mpc_struct_flash": {
    "description": "Flash MPC structures. The total number of RAM and Flash MPC structures must be less than 32",
    "value": [
        {
            "description": "Offset and size in KB. The values must be multiple of 2 (e.g. 0 KB, 2 KB, 4 KB, 8 KB, ...)",
            "offset": "64 KB",
            "size": "192 KB",
            "attr": {
                "description": "The MPC attributes for this memory region",
                "comment_1": "WRN allow read, write, and NS only access",
                "comment_2": "-RN allow read, NS-only access",
                "comment_3": "--- and --N, no access",
                "comment_4": "WR- read and write access, but NS=0 only",
                "value": {
                    "pc0": "WR-",
                    "pc1": "WR-",
                    "pc2": "WRN",
                    "pc3": "WRN",
                    "pc4": "--N",
                    "pc5": "--N",
                    "pc6": "--N",
                    "pc7": "--N"
                }
            }
        }
    ]
}
```

– 3

Copyright © Infineon Technologies AG 2026. All rights reserved.

Secure project code

The following are the steps to be followed:

1. Setup SAU for entire address space
2. Configure PPC
3. Set the NS stack pointer before launching NS image
4. Get the function pointer of NS image Reset Handler
5. Change the PC from 2 to 3
6. Start NS project by calling its Reset Handler

```
int main(void)
{
    cy_rslt_t result = 0xFFFFFFFF;
    uint32_t ns_stack = 0;
    funcptr_void NonSecure_ResetHandler = NULL;
    cy_en_ms_ctl_status_t ms_ctl_status = CY_MS_CTL_FAILURE;
    cy_en_ppc_status_t ppc_status = CY_PPC_FAILURE;

    /* Initialize the device and board peripherals */
    result = cybsp_init();
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Configure SAU for SRAM, Flash and Peri MMIO */
    TZ_SAU_Setup();                                                    1

    /* Configure PPC */
    ppc_status = config_ppc();                                         2
    if (CY_PPC_SUCCESS != ppc_status)
    {
        CY_ASSERT(0);
    }

    /* Enable global interrupts */
    __enable_irq();

    ns_stack = (uint32_t)(*((uint32_t*)CM33_SP_STORE));                3
    __TZ_set_MSP_NS(ns_stack);

    NonSecure_ResetHandler = (funcptr_void)(*((uint32_t*)CM33_RESET_HANDLER_STORE));
                                                                       4
    /* change pc value for cm33-ns */
    ms_ctl_status = config_set_cm33_ns_pc();                           5
    if (CY_MS_CTL_SUCCESS != ms_ctl_status)
    {
        CY_ASSERT(0);
    }

    /* Start non-secure application */
    NonSecure_ResetHandler();                                          6
```


Copyright © Infineon Technologies AG 2026. All rights reserved.

Non-secure project code

```c
int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init();

    /* Board init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Enable global interrupts */
    __enable_irq();

    /* Initialize the User LED */
    result = cyhal_gpio_init(CYBSP_USER_LED, CYHAL_GPIO_DIR_OUTPUT,
                             CYHAL_GPIO_DRIVE_STRONG, CYBSP_LED_STATE_OFF);

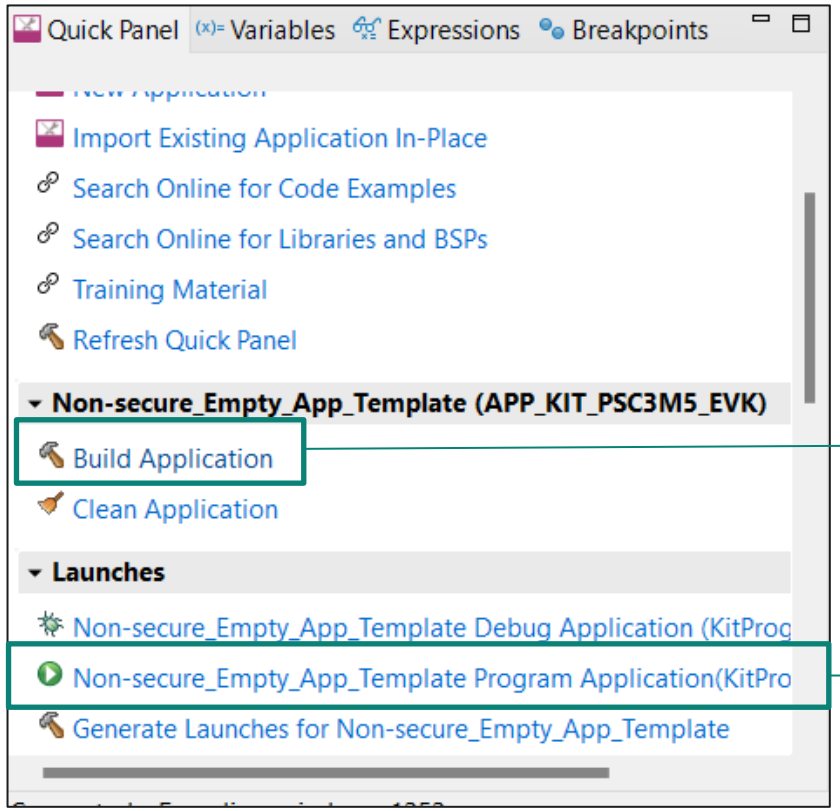
    /* GPIO init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    for (;;)
    {
        cyhal_gpio_toggle(CYBSP_USER_LED);
        cyhal_system_delay_ms(BLINKY_LED_DELAY_MSEC);
    }
}
```


Copyright © Infineon Technologies AG 2026. All rights reserved.

Build and program the application

- After the device is provisioned successfully, build and program the application
- After successfully programming the application, you will observe a **blinking LED** on the EVK. This LED is toggled by the non-secure application



Build the application

Program the application

Copyright © Infineon Technologies AG 2026. All rights reserved.

Revision history

Document revision	Date	Description of change
**	2026-02-03	Initial release

Copyright © Infineon Technologies AG 2026. All rights reserved.


Infineon