S32K3 HSE

CHARLES ZHAO

Created: Nov 2022

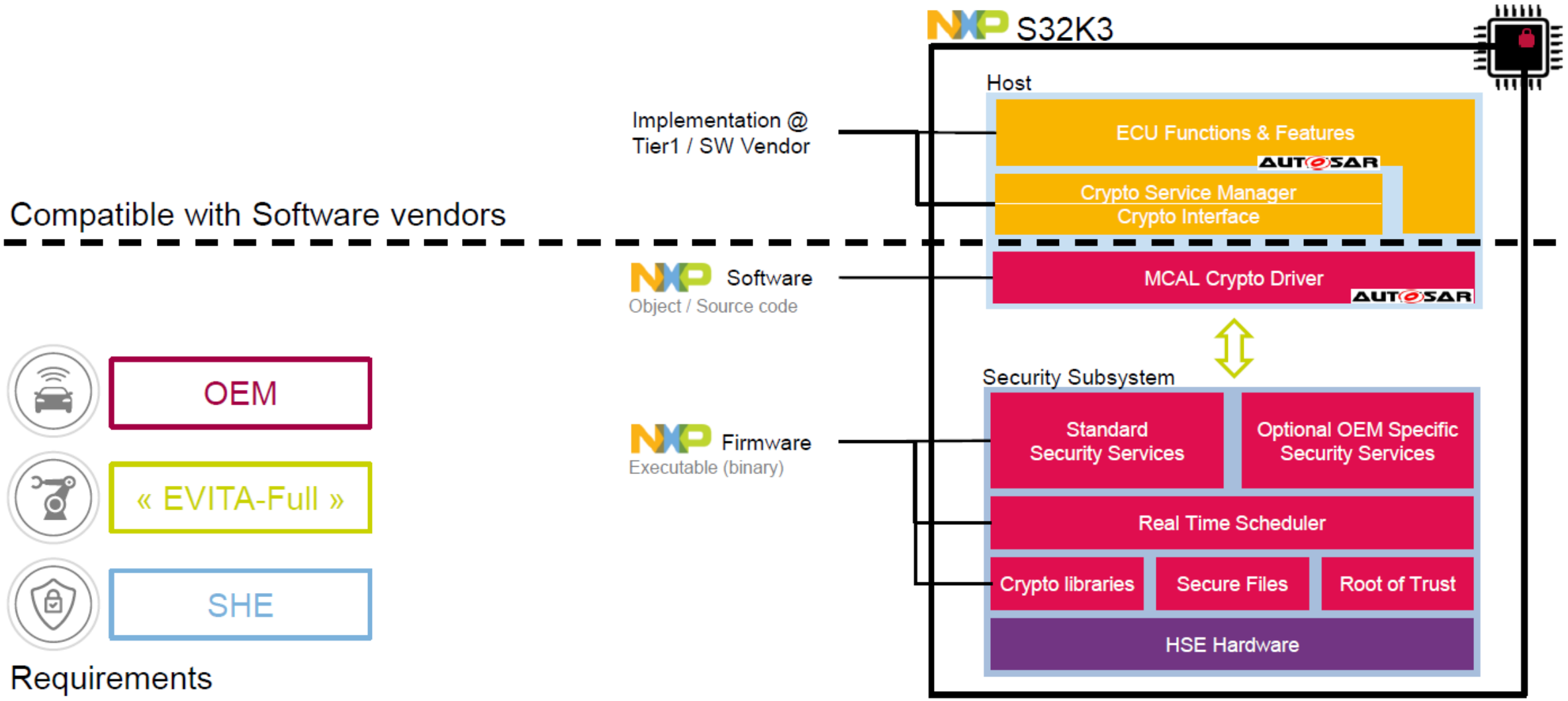
Last Update: Dec 2022

Agenda

- HSE Overview
- HSE FW & Documentation
- HSE Memory Layout
- HSE Firmware Install & Update
- HSE OTA
- HSE Recovery Mode
- HSM FW Handshake
- HSE Key Management & Crypto Services
- HSE Secure Boot
- HSE Lifecycle & Host Debug Access
- Some Important Notes

HSE OVERVIEW

Overview


S32K3
Host
Implementation @
Tier1 / SW Vendor
ECU Functions & Features
AUTOSAR
Crypto Service Manager
Crypto Interface
Compatible with Software vendors
Software
Object / Source code
MCAL Crypto Driver
AUTOSAR
Security Subsystem
Firmware
Executable (binary)
Standard
Security Services
Optional OEM Specific
Security Services
Real Time Scheduler
Crypto libraries
Secure Files
Root of Trust
HSE Hardware
OEM
« EVITA-Full »
SHE
Requirements

HSE NATIVE SECURITY SERVICES

Cryptographic functions
- Encryption / decryption
- MAC generation / verification
- Hashing
- Signature generation / verification

Key management
- Key import & export
- Key generation
- Key derivation
- Key exchange
- SHE specification services

Random number generation
- Pseudo-random numbers based on true random seed

Secure Memory Regions (SMR)
- Memory verification at start-up (secure boot)
- Memory verification at run-time

Monotonic counters
- Incrementing and reading volatile & non-volatile counters

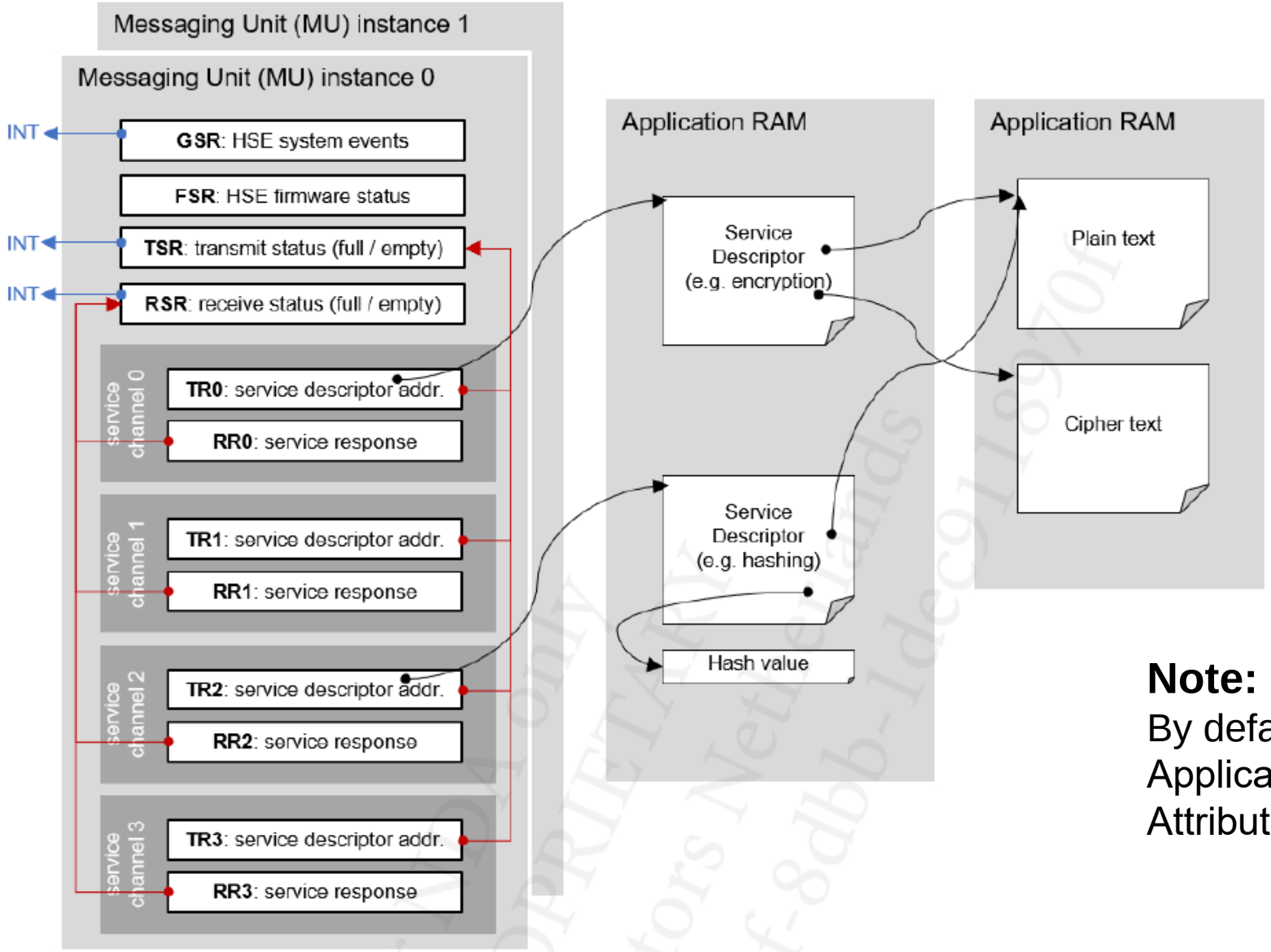
Administration
- System initialization & configuration
- Functional tests
- Security policy manager
- Service updates & extension
- FW Update

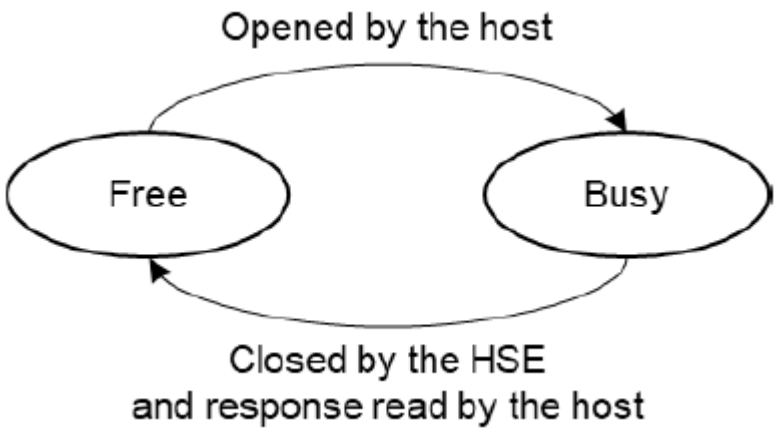
Secure network protocols
- N/A

HSE CRYPTOGRAPHIC SERVICES

Service	Feature	Description
Cryptography	Ciphers	AES: ECB, CBC, CFB, OFB, CTR RSAES: PKCS1-v1_5, OAEP ECIES * Software implementation in S32K3
	Message Authentication Code (MAC)	AES: CMAC, GMAC, HMAC, FAST CMAC, CMAC with Counter
	Hashing	SHA1, SHA224, SHA256, *SHA384, *SHA512 Miyaguchi-Preneel Compression
	Authenticated ciphers	AES: CCM, GCM
	Digital signature generation and verification	RSASSA_PSS RSASSA_PKCS1-v1_5 ECDSA – ECC over GF(p) with all prime standard curve supported BN p256, p638 / ANSI x9p192 to x9p512 / Brainpool P160 to P512 / sec p128 to p512/ TU Darmstadt primeCurve 1 to 35 EdDSA - Ed25519
Key Management	Max key sizes	AES: 256 bits RSA: 4096 bits ECC: 521 bits
	Key generation	Permanent and ephemeral RSA and ECC key pair generation
	Key import	Plain or encrypted form, with optional authentication tag. SHE key update protocol
	Key derivation	various: NIST 800-108, PBKDF2, TLS1.2_PRF, HKDF etc.
	Key exchange	ECDH, ECC Burmester-Desmedt Protocol
	Certificate handling	Key Installation from x.509 and CVC certificates Certificate installation for Root of Trust establishment.
Random Number	Pseudo random generation	Based on a True Random Number AIS31 Class P2 high and FIPS 140-2 compliant (Supported classes: PTG.3, DRG3, DRG.4)

HSE INTERFACE





Note:

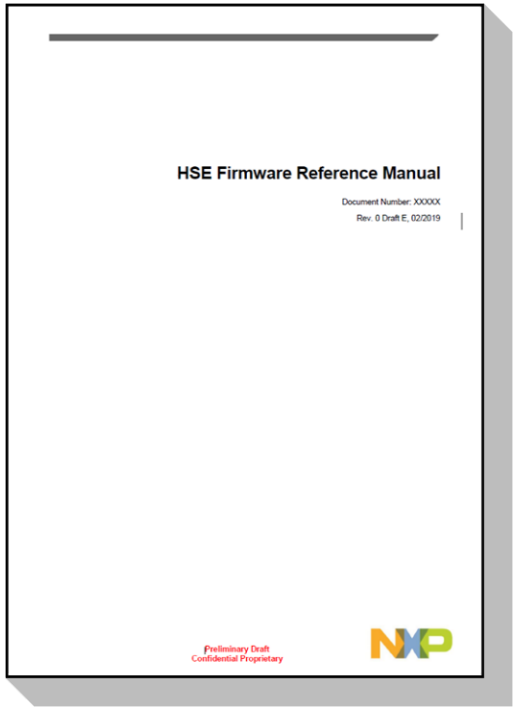
By default, only MU0 is enable, The Application can enable more MU using “Set Attribute” service

HSE FW & DOCUMENTATION

DOC AND HSE-FW

Design : Product Information : Automotive SW - S32K3 - HSE Firmware (flexnetoperations.com)

HSE FW Reference Manual (detailing the HSE configuration & usage)



NXP HSE FW package:
- binary and interface (.h files)
- HSE Service API RM

HSE FW Demo App package:
- Sample code (scripts, readme)
- HSE FW FAQ,
- HSE Demo App installer.

HSE – Related Documents on Docstore

Unknown	Tr744910 - K3_SecurityWorkshop_CryptoDriver_29Mar2022 (1.0)
Unknown	Tr744101 - S32K3XX HSE And OTA Advance Training (0.1)
Unknown	Tr665601 - S32K344_HSE_Training (0.1)
Development Software: Host Device Drivers	Sw745310 - SecureBootAppNoteDemo (1.0)
Development Software: Application Development Tools	Sw744701 - S32K312_HSE_FW_INSTALL_V_0_1_2_1.Zip (0.1)
Unknown	Sw559707 - S32K3 HSE Service API RM (0.7)
Unknown	Si774301 - Boot Performance Numbers On S32K3x4 (0.1)
Unknown	Eb788501 - S32K3 - New HSE FW & SBAF (0.1)
Data sheet	Ds765101 - Cybersecurity Case For S32K344 (0.1)
Data sheet	Ds765001 - NXP Cybersecurity Plan For S32K344 (0.1)
Application note	An781201 - Secure Boot Overview Training (0.1)
Application note	An745220 - S32K3 HSE Traning - Oct2022 (2.0)

DOC store : https://www.docstore.nxp.com/

Application note	An744810 - HSE FW Install For S32K3xx (1.0)
Application note	An744610 - S32K3xx_Memories_Training. (1.0)
Application note	An744511 - Secure Boot Application Note V0.1.1.0 (1.1)
Application note	An744410 - K3_SecurityWorkshop_VKMS__29Mar2022.Pdf (1.0)
Manual	RM758221 - HSE-B Firmware Reference Manual - V2.1 (2.1)

HSE – HSE FW

NXP > Design > **Product Information : Automotive SW - S32K3 - HSE Firmware**

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

FAQ

- Download Help
- Table of Contents
- FAQs

Product Information

Automotive SW - S32K3 - HSE Firmware

Select a version. To access older versions, click on the " Previous " tab

Current | Previous

Version	—	Description	Date Available	
0.2.1.0	—	HSE FW 0.2.1.0 RTM Release This is the HSE standard firmware 0.2.1.0 RTM release targeting the S32K344, S32K324, S32K314 devices.	Jul 8, 2022	Download Log
0.1.2.1	—	HSE FW 0.1.2.1 Hotfix Release This is the HSE Standard FW 0.1.2.1 Hotfix release targeting the S32K312 platform.	Feb 7, 2022	Download Log
0.1.2.0	—	HSE FW 0.1.2.0 BETA Release This is the HSE Standard FW 0.1.2.0 BETA release targeting the S32K312 platform.	Jan 18, 2022	Download Log
0.14.0	—	HSE FW 0.0.14.0 EAR Release This is the HSE Standard FW 0.0.14.0 EAR release targeting the S32K342, S32K322 and S32K341 devices.	Dec 7, 2021	Download Log
0.1.1.0	—	HSE FW 0.1.1.0 RTM Release This is the HSE Standard FW 0.1.1.0 RTM release targeting the S32K3X4 platform.	Oct 13, 2021	Download Log
0.0.11.0	—	HSE FW 0.0.11.0 EAR Release This is the HSE Standard FW 0.0.11.0 EAR release targeting the S32K3X2 platform.	Aug 17, 2021	Download Log

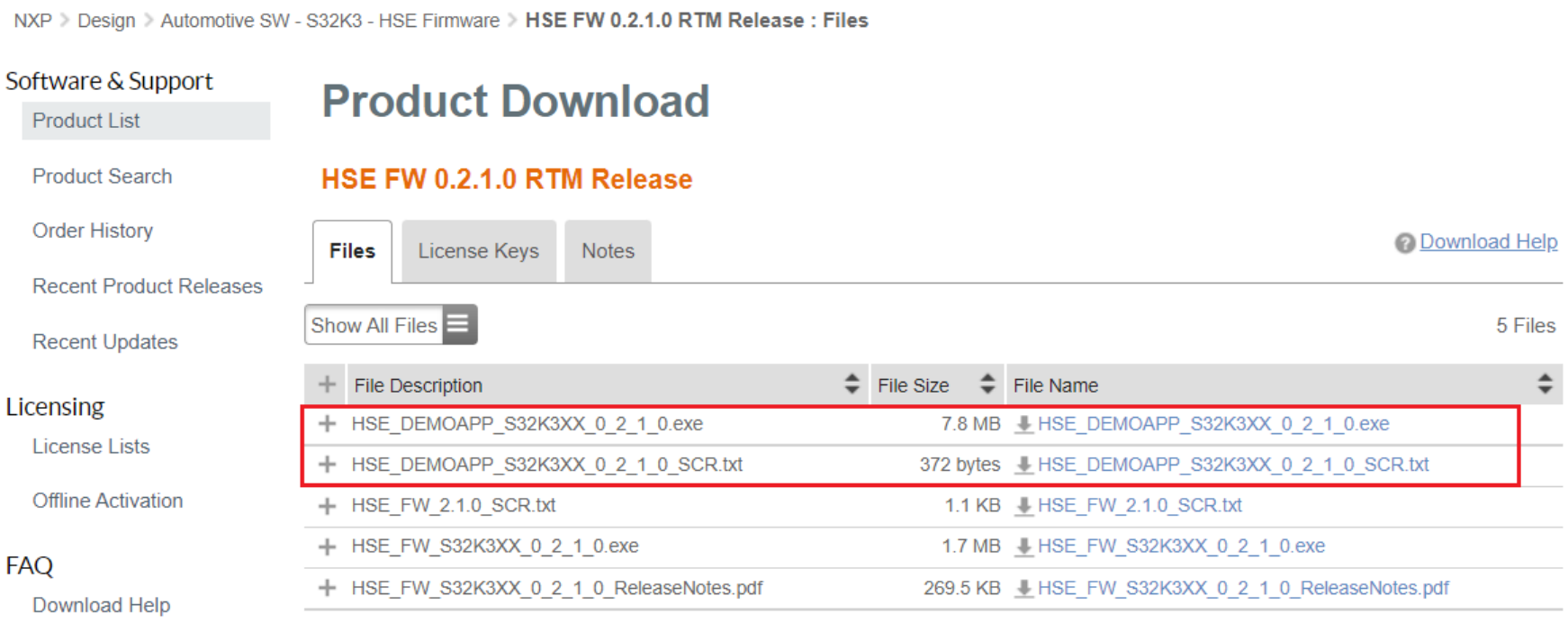
HSE – HSE FW

NXL79504 > OSDisk (C:) > NXP > HSE_FW_S32K3XX_0_2_1_0

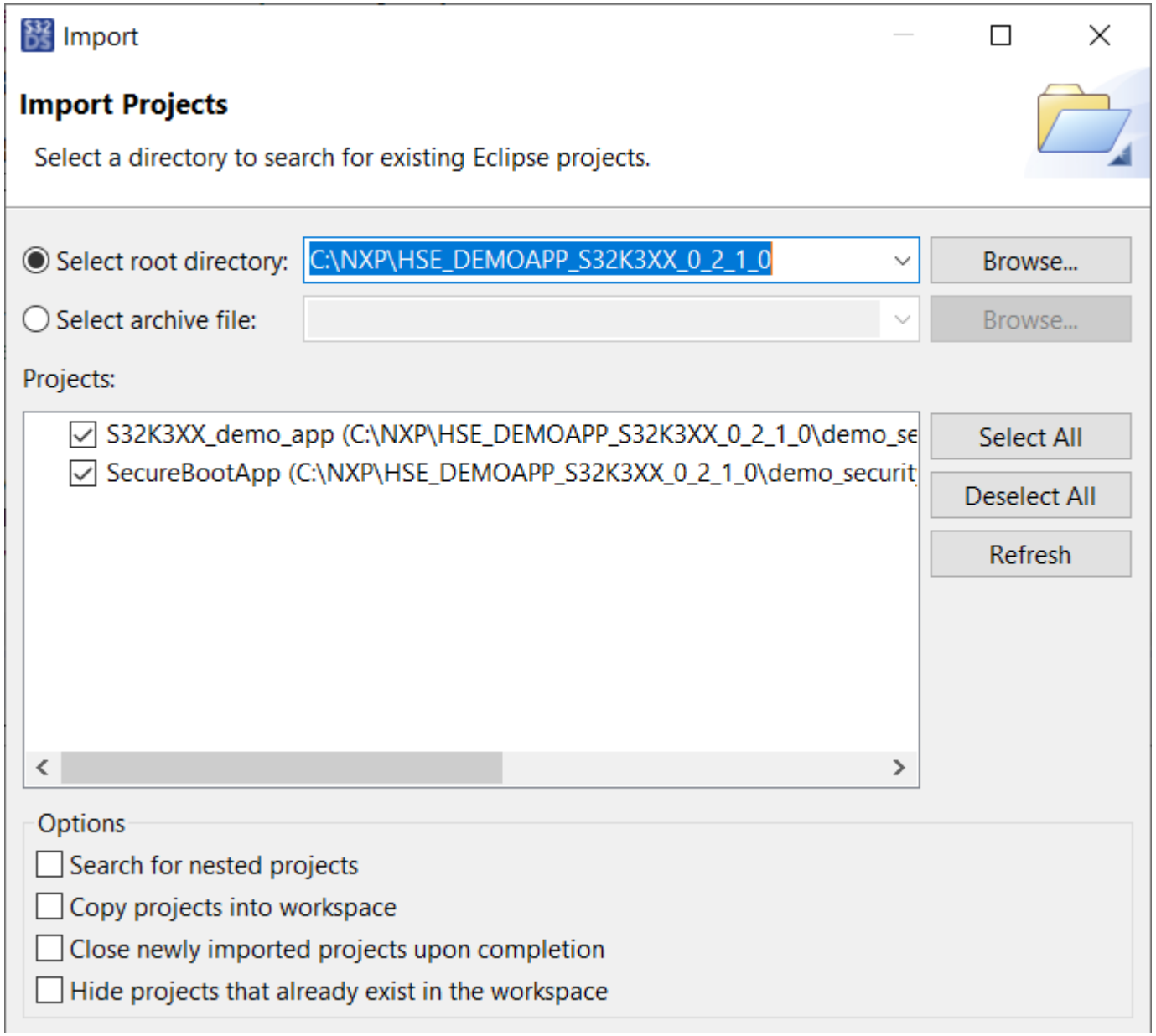
Name	Date modified	Type	Size
docs	2022/7/20 11:37	File folder	
hse_ab_swap	2022/7/20 11:37	File folder	
hse_full_mem	2022/7/20 11:37	File folder	
GettingStarted.html	2022/7/20 11:37	Microsoft Edge HT...	7 KB
HSE_FW_S32K3XX_0_2_1_0_ReleaseNotes...	2022/7/8 20:33	Adobe Acrobat D...	270 KB
license.rtf	2022/7/8 20:35	Rich Text Format	446 KB
uninst.exe	2022/7/20 11:37	Application	80 KB

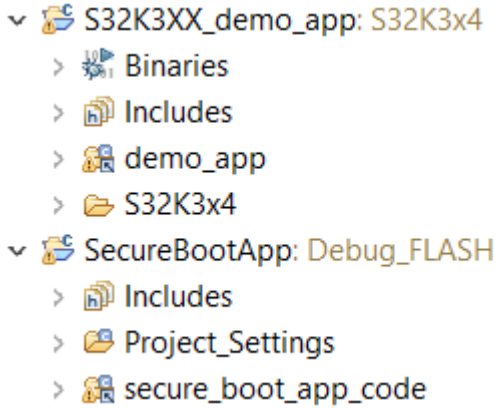
HSE DEMO APP

- ✓ **Focused** approach on HSE service interface
- ✓ Provides bare metal **sample codes** for HSE services for ease of use and understanding
- ✓ Released along with HSE-Firmware
- ✓ **ReadMe** document



HSE Demo APP

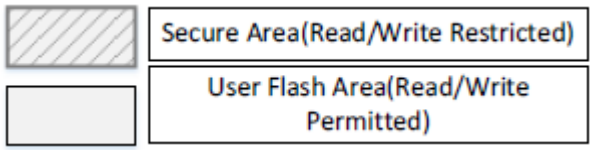


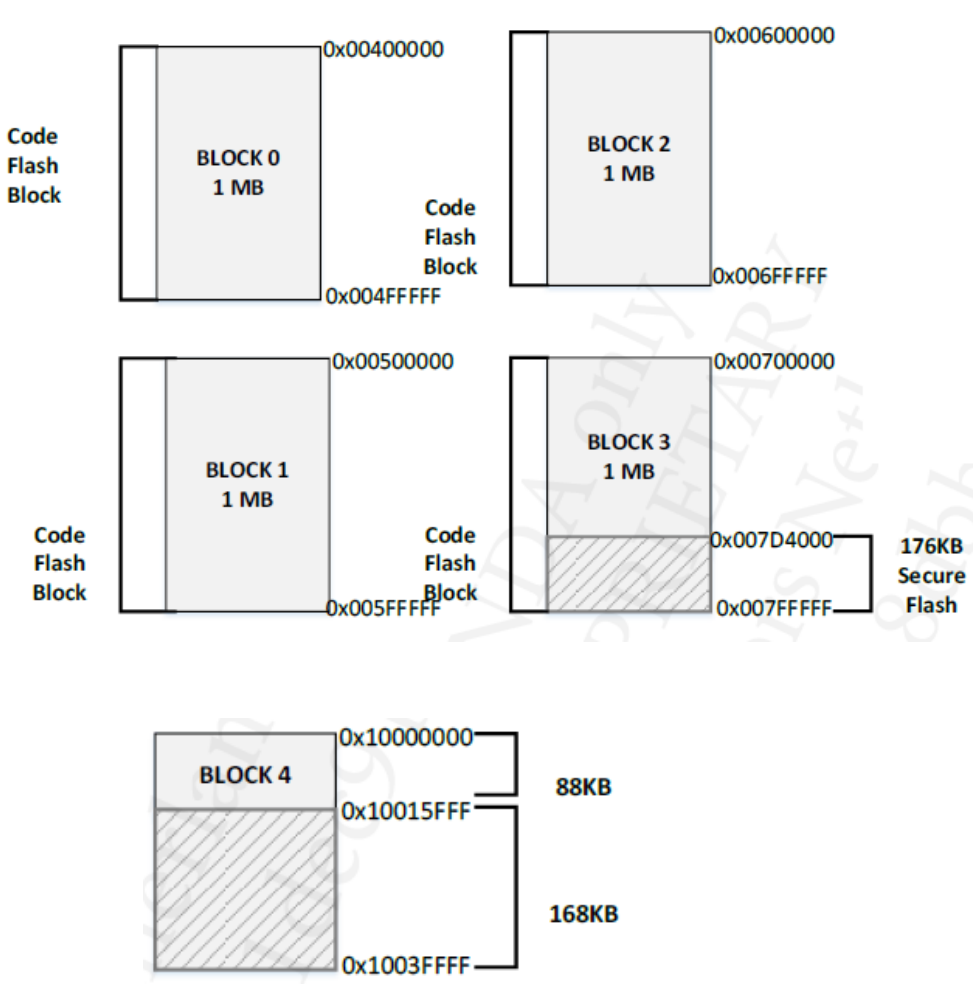


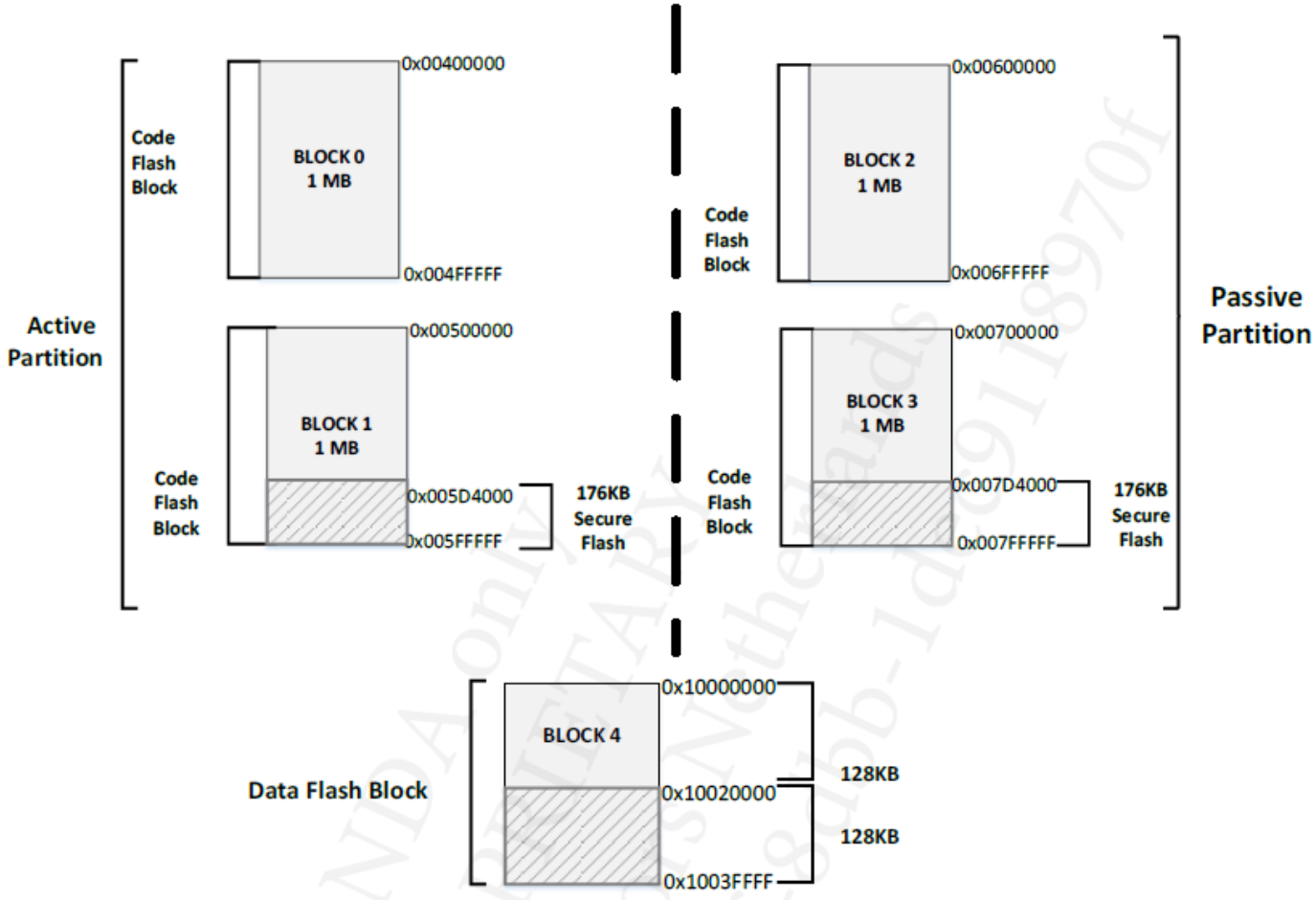

```
arm-none-eabi-size --format=berkeley S32K3XX_demo_app.elf
arm-none-eabi-objdump --source --all-headers --demangle --line-numbers
   text    data     bss     dec     hex filename
  55832    1075   36917   93824   16e80 S32K3XX_demo_app.elf
Finished building: S32K3XX_demo_app.siz
Finished building: S32K3XX_demo_app.srec


Finished building: S32K3XX_demo_app.lst


Invoking: Standard S32DS Create Flash Image
arm-none-eabi-objcopy -O srec secure_boot_app.elf  "secure_boot_app.srec"
Invoking: Standard S32DS Print Size
arm-none-eabi-size --format=berkeley secure_boot_app.elf
   text    data     bss     dec     hex filename
   2158      72    1036    3266     cc2 secure_boot_app.elf
Finished building: secure_boot_app.siz
Finished building: secure_boot_app.srec
```

HSE MEMORY LAYOUT

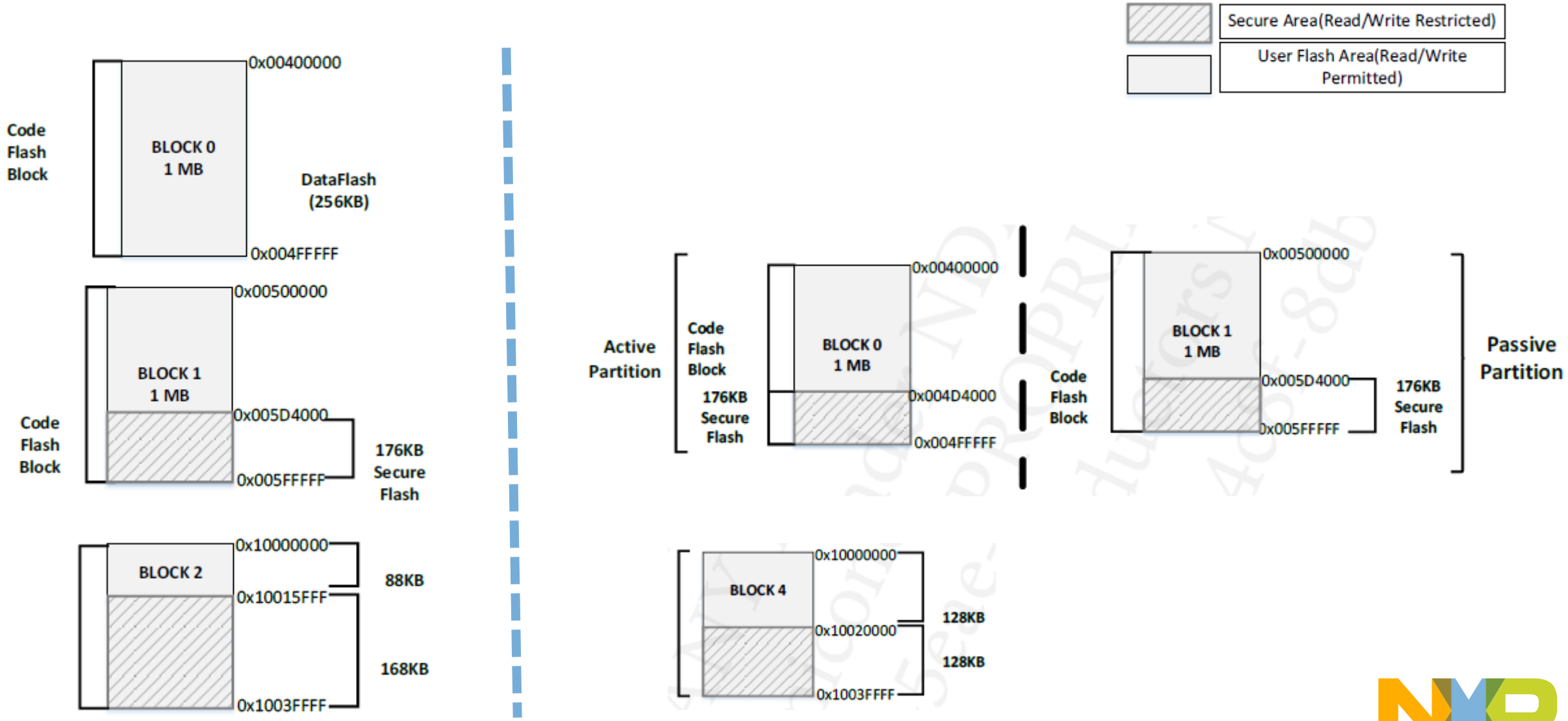
S32K3x4 HSE MEMORY LAYOUT







S32K3x2 HSE MEMORY LAYOUT



S32K3

Table 126: Secure NVM mapping (FULL_MEM)

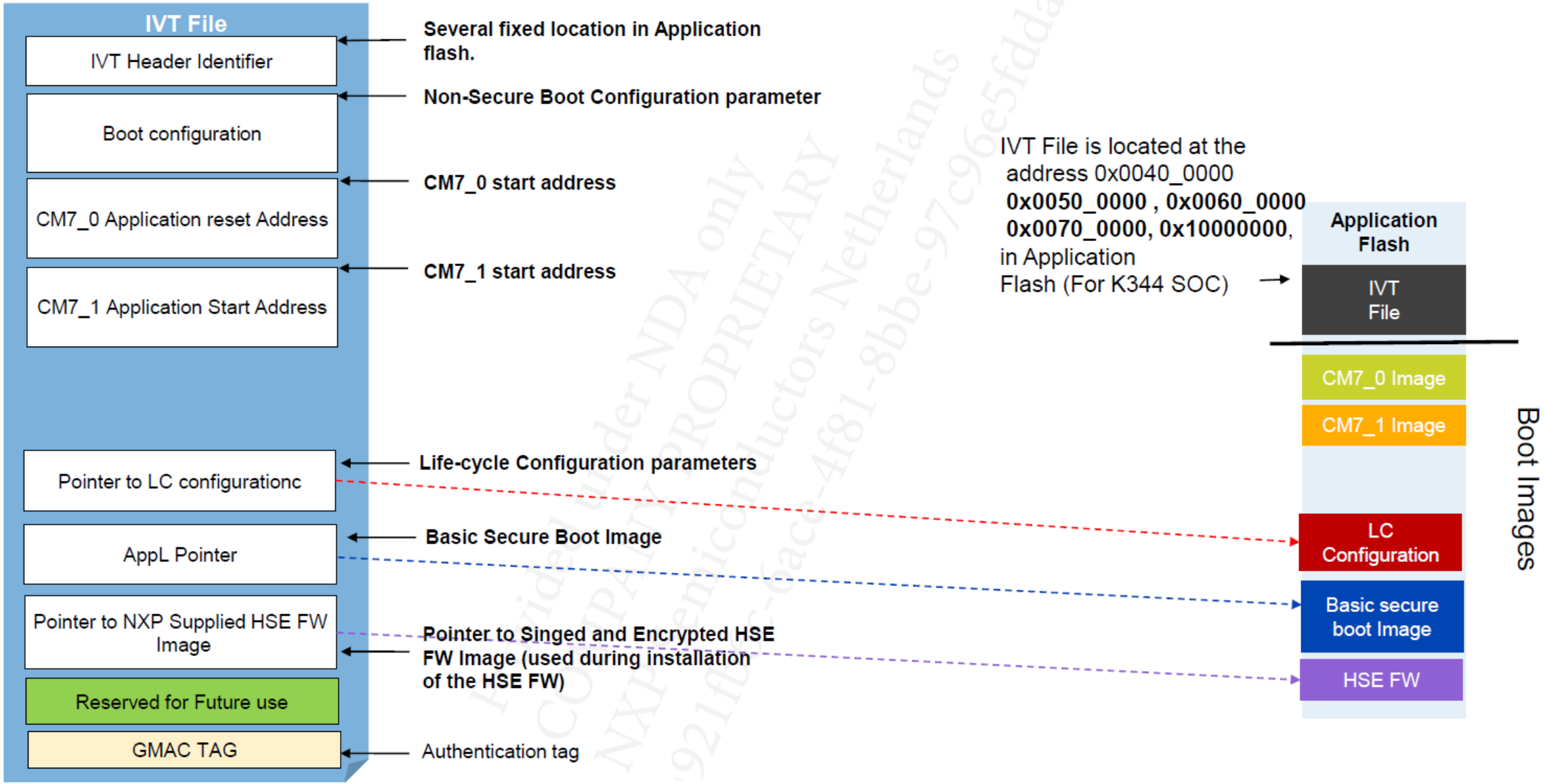
Device	Flash area	Start address	Size
Common	HSE data flash	0x10016000	168KB
	HSE configuration (UTEST)	0x1B000000	8KB
S32K344, S32K324, S32K314	HSE code flash	0x007D4000	176KB
S32K312, S32K342, S32K322, S32K341	HSE code flash	0x005D4000	176KB

Table 127: Secure NVM mapping (AB_SWAP)

Device	Flash area	Start address	Size
Common	HSE data flash	0x10020000	128KB
	HSE configuration (UTEST)	0x1B000000	8KB
S32K344, S32K324, S32K314	HSE code flash (passive area)	0x007D4000	176KB
	HSE code flash (active area)	0x005D4000	176KB
S32K312, S32K342, S32K322, S32K341	HSE code flash (passive area)	0x005D4000	176KB
	HSE code flash (active area)	0x004D4000	176KB

HSE FW INSTALL & UPDATE

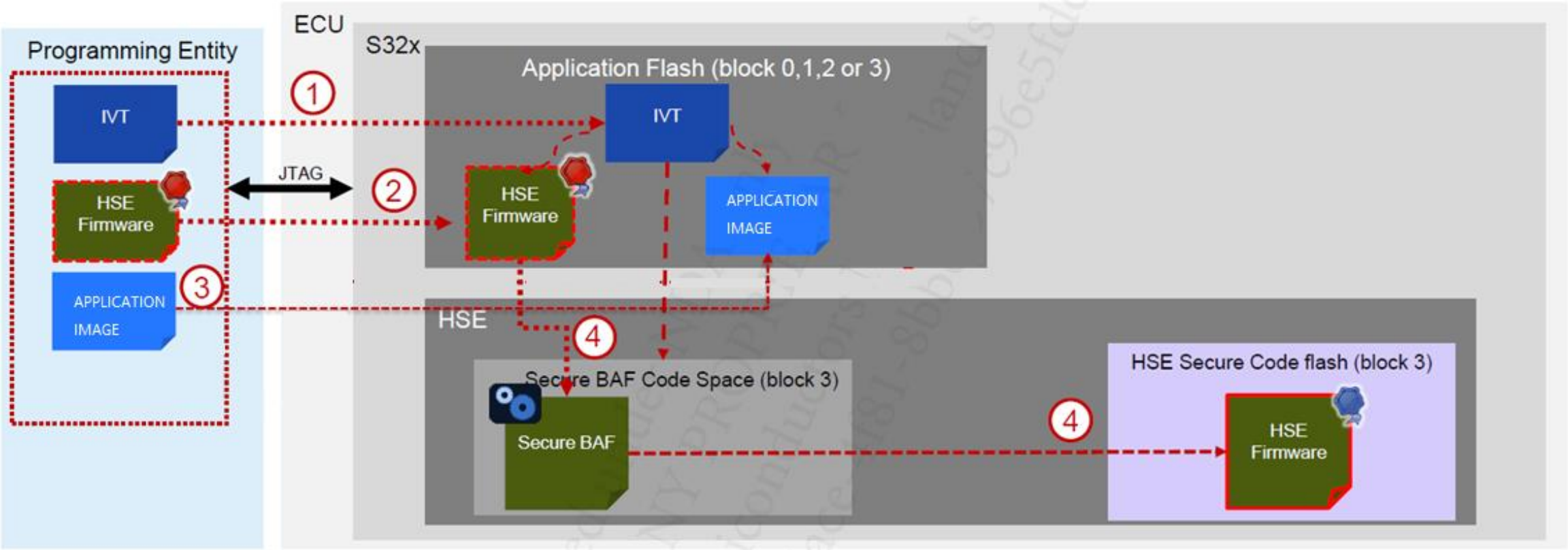
HSE IVT



INSTALL HSE FW METHODS

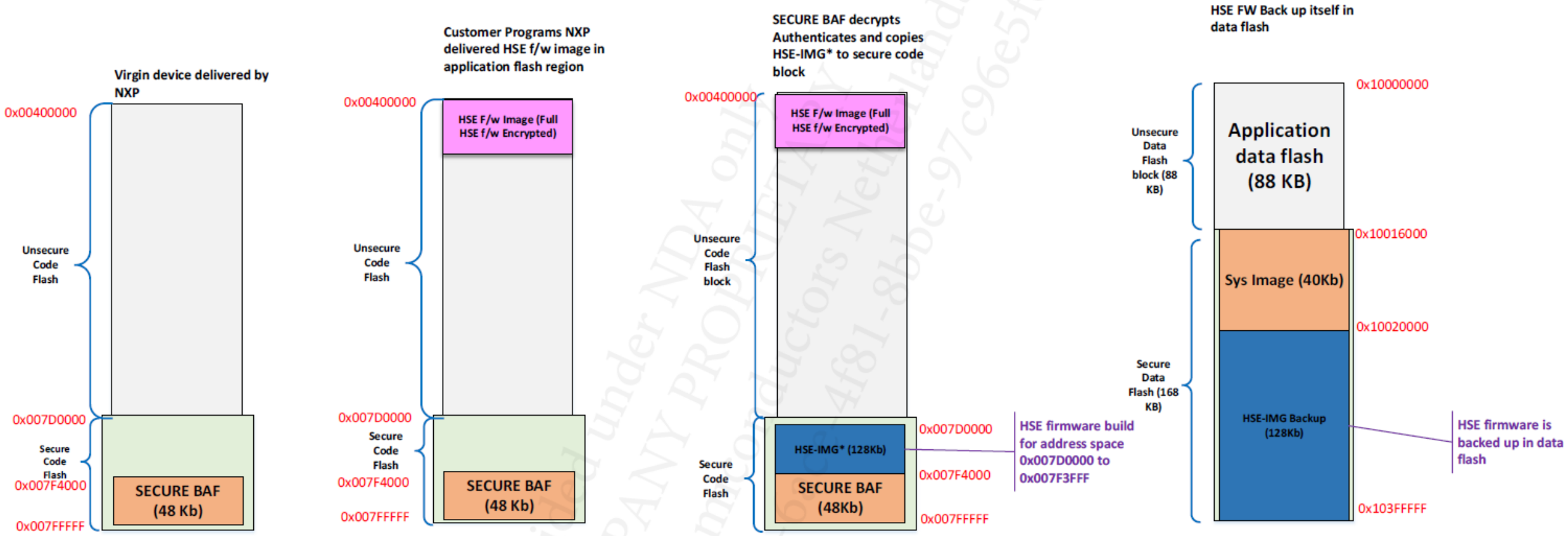
- **Method 1**: Program the encrypted image of HSE FW at start location of code flash area i.e. 0x00400000 and give a reset. SBAF installs the HSE FW after reset.
- **Method 2**: Program the address encrypted image of HSE FW in IVT and program the encrypted HSE FW image at the provided address. After programming, provide a reset.
- **Method 3**: Installing the HSE FW through MU interface. Refer to HSE FW reference manual for more details. The advantage of this approach is that user doesn't need to program the encrypted image in flash. It can be saved in RAM also.

INSTALLATION ON A VIRGIN DEVICE FOR FULL MEM

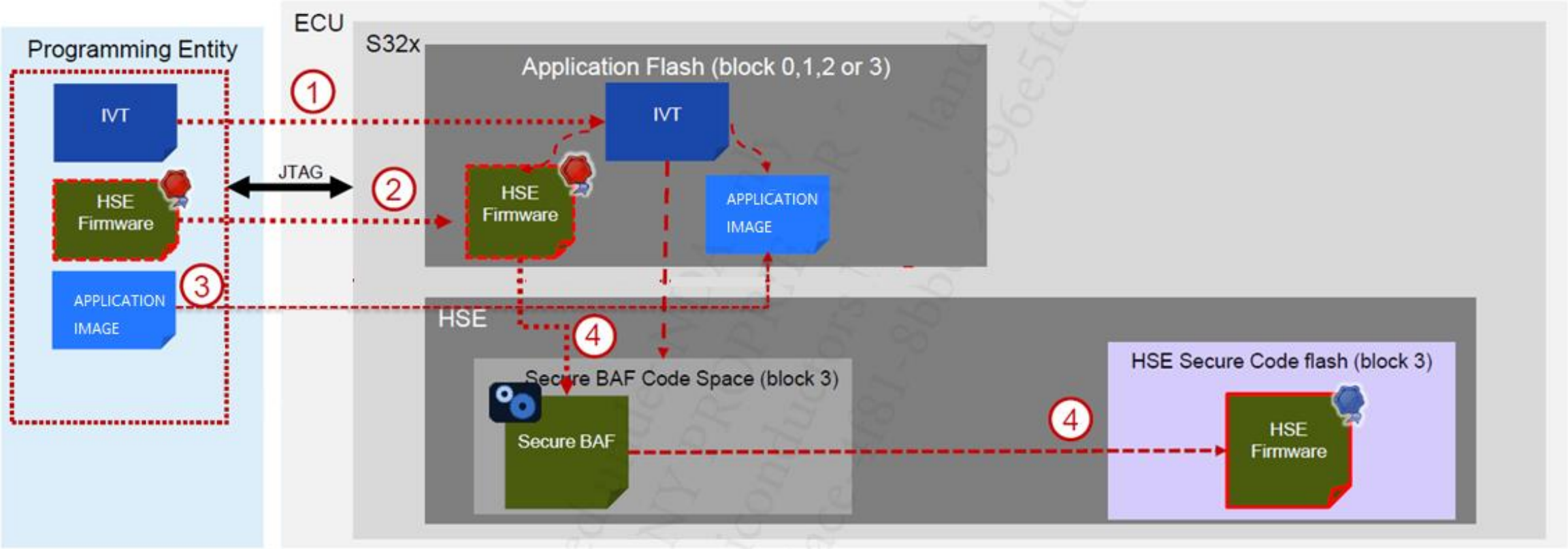


1. The programming entity programs the images IVT in application flash block.
2. The programming entity programs HSE FW Image in application flash block.
3. The programming entity program application image in application flash block and issues a reset.
4. Upon next boot HSE will decrypt, authenticate and program the HSE FW image in secure code area. It will boot the application and HSE FW.

FLASH MEMORY LAYOUT FOR FULL MEM

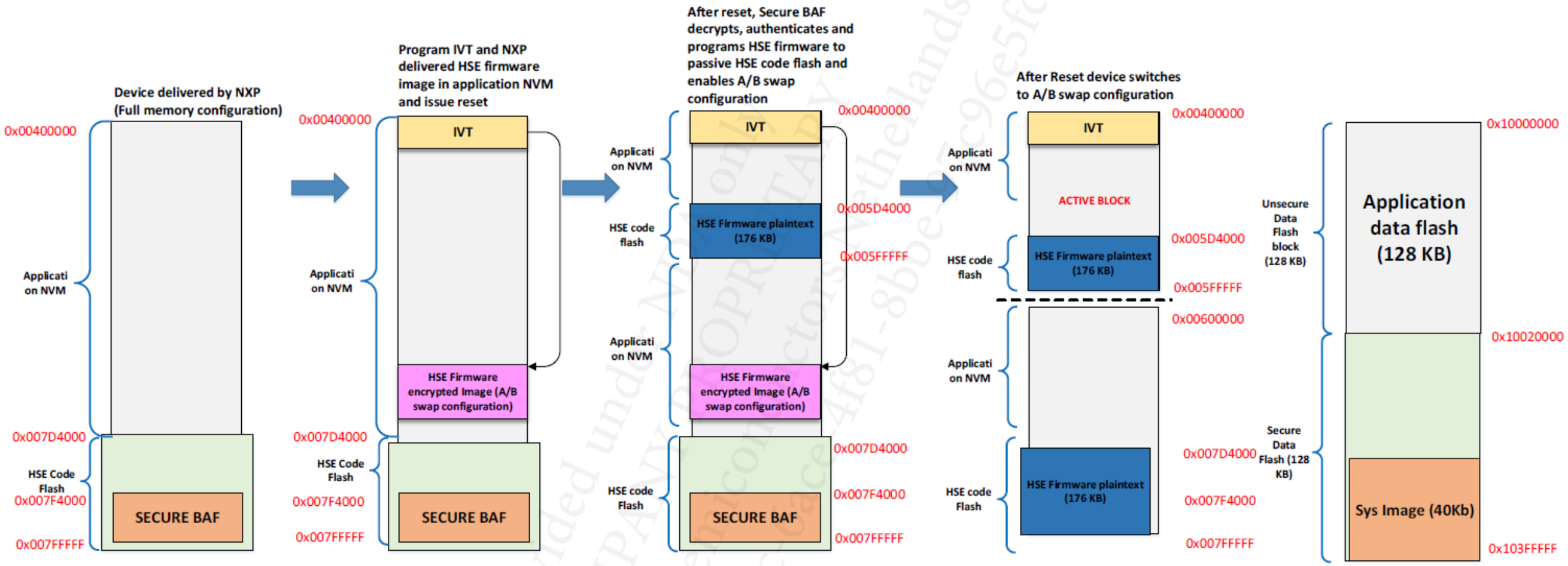


INSTALLATION ON A VIRGIN DEVICE FOR A/B SWAP

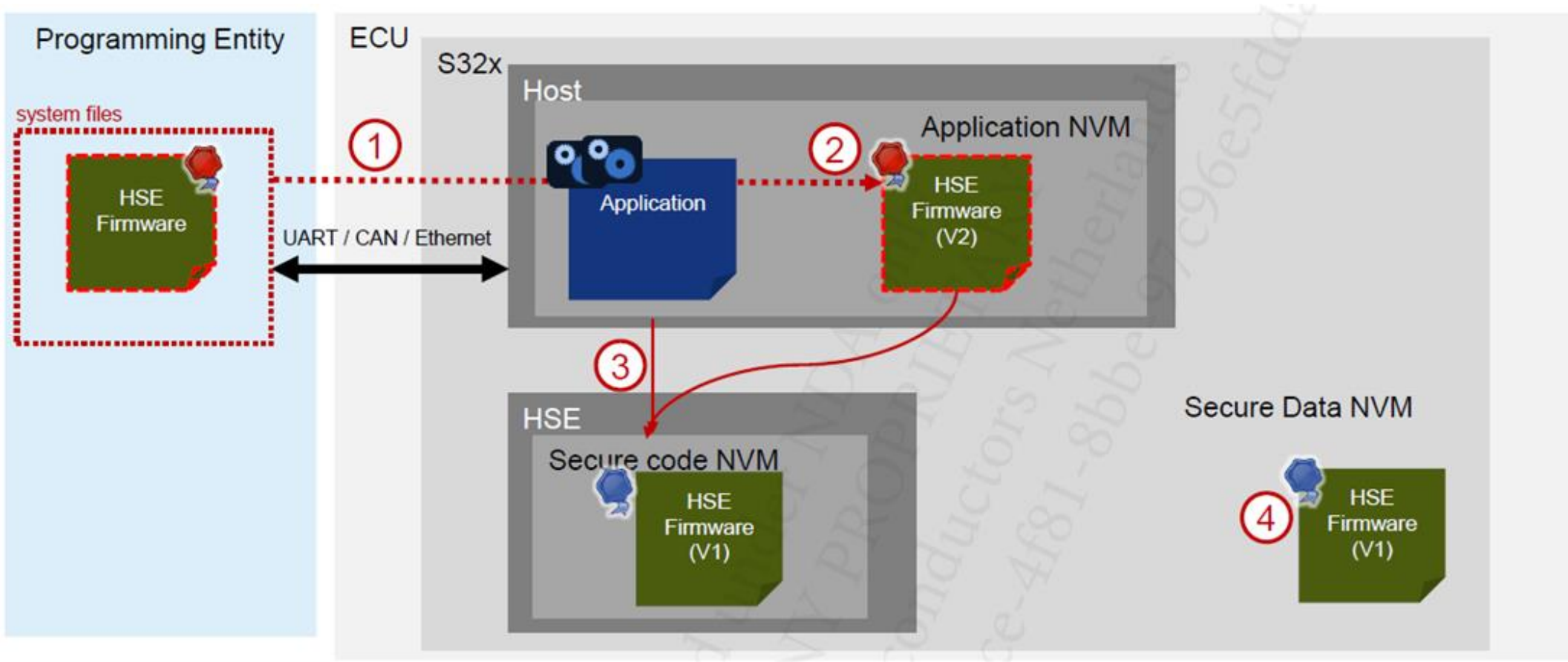


1. The programming entity programs the images IVT in application flash block.
2. The programming entity programs A/B swap HSE FW Image in application flash block.
3. The programming entity program application image in application flash block and issues a reset.
4. Upon next boot HSE will decrypt, authenticate and program the HSE FW image in secure code area. It will program A/B swap enable configuration parameters and issue reset.

FLASH MEMORY LAYOUT FOR A/B SWAP

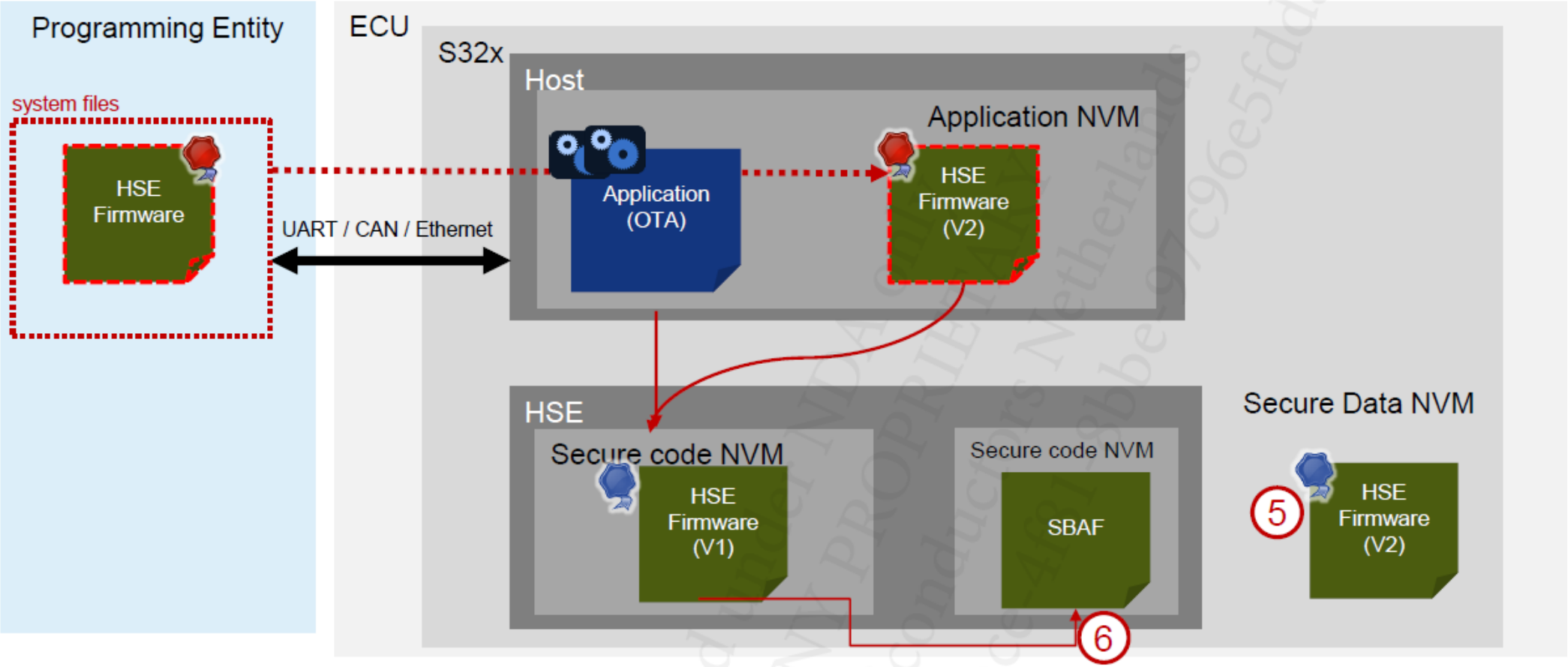


HSE FIRMWARE UPDATE FULL MEM DEVICE



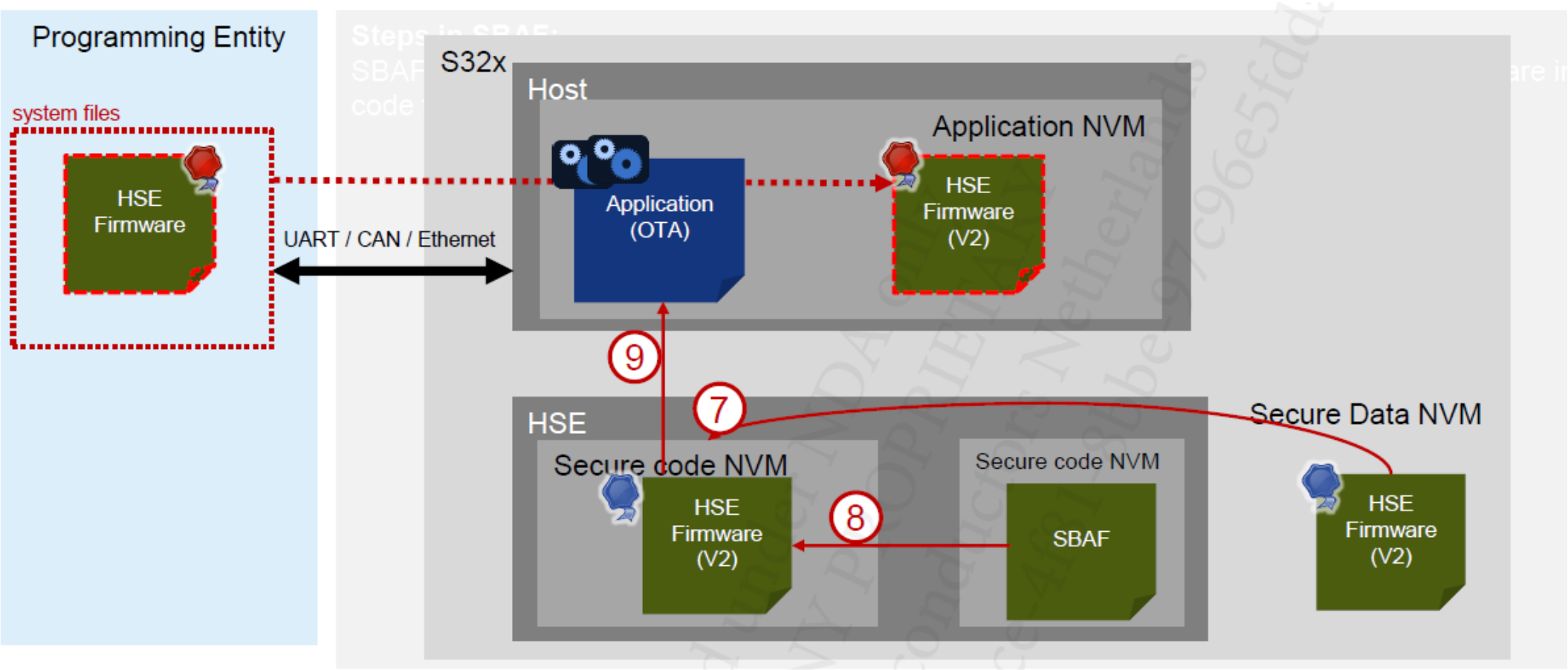
1. The Application receives a new image of HSE Firmware (pink image) from a Programming Entity
2. Application stores the image in application NVM area in case of one-shot mode or in RAM area in case of streaming mode.
3. The Application requests for a firmware update service.
4. The current running firmware after doing the basic sanity check erases the backup firmware from data flash.

HSE FIRMWARE UPDATE FULL MEM DEVICE



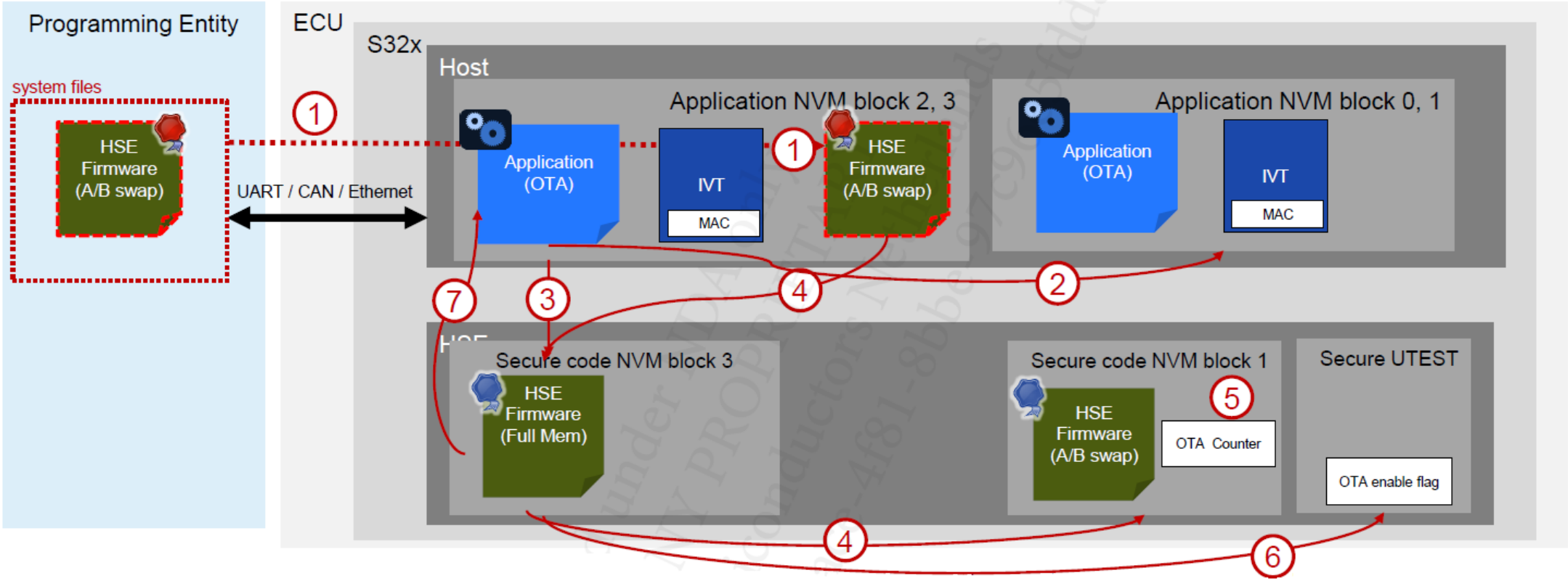
5. The HSE decrypts new firmware, verifies it and programs it in the data flash.
6. After successful programming of the new firmware, it passes the control to SBAF.

HSE FIRMWARE UPDATE FULL MEM DEVICE



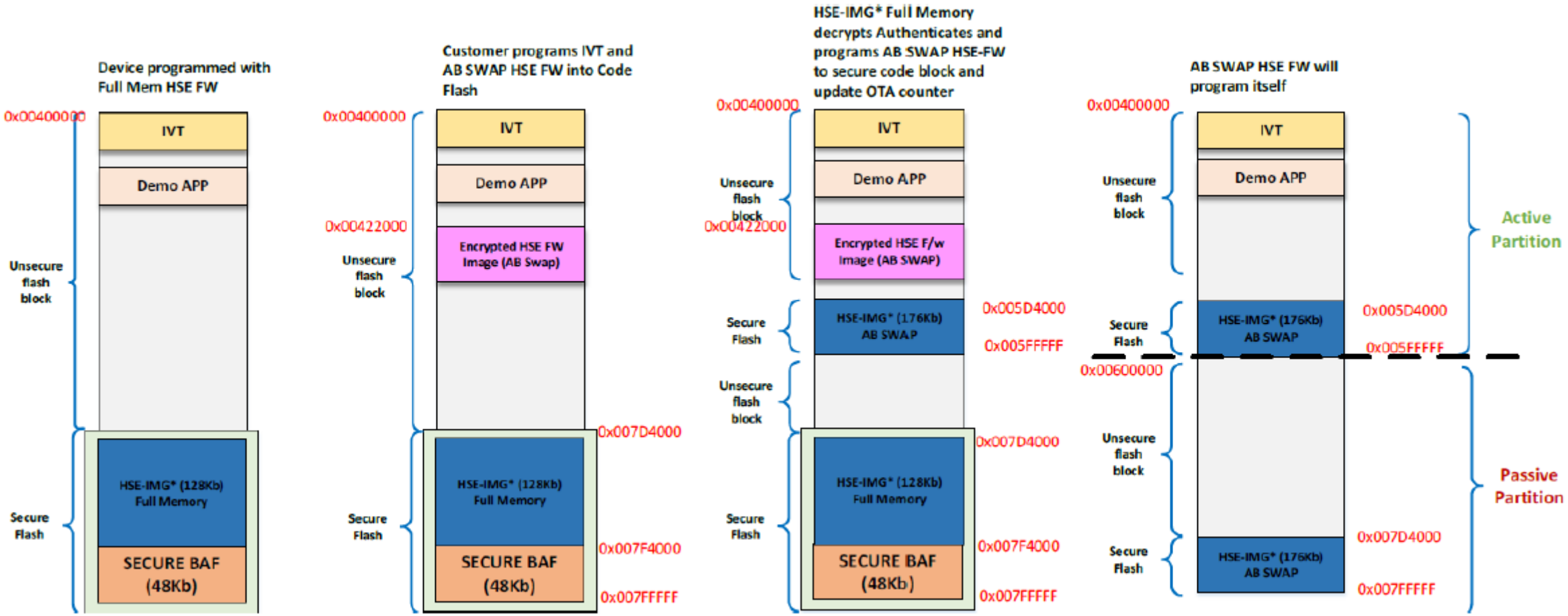
7. SBAF restores the firmware from data flash to code flash.
8. SBAF passes the control to new firmware.
9. New firmware sends the response back to application.

HSE FIRMWARE UPDATE FROM FULL MEM TO A/B SWAP



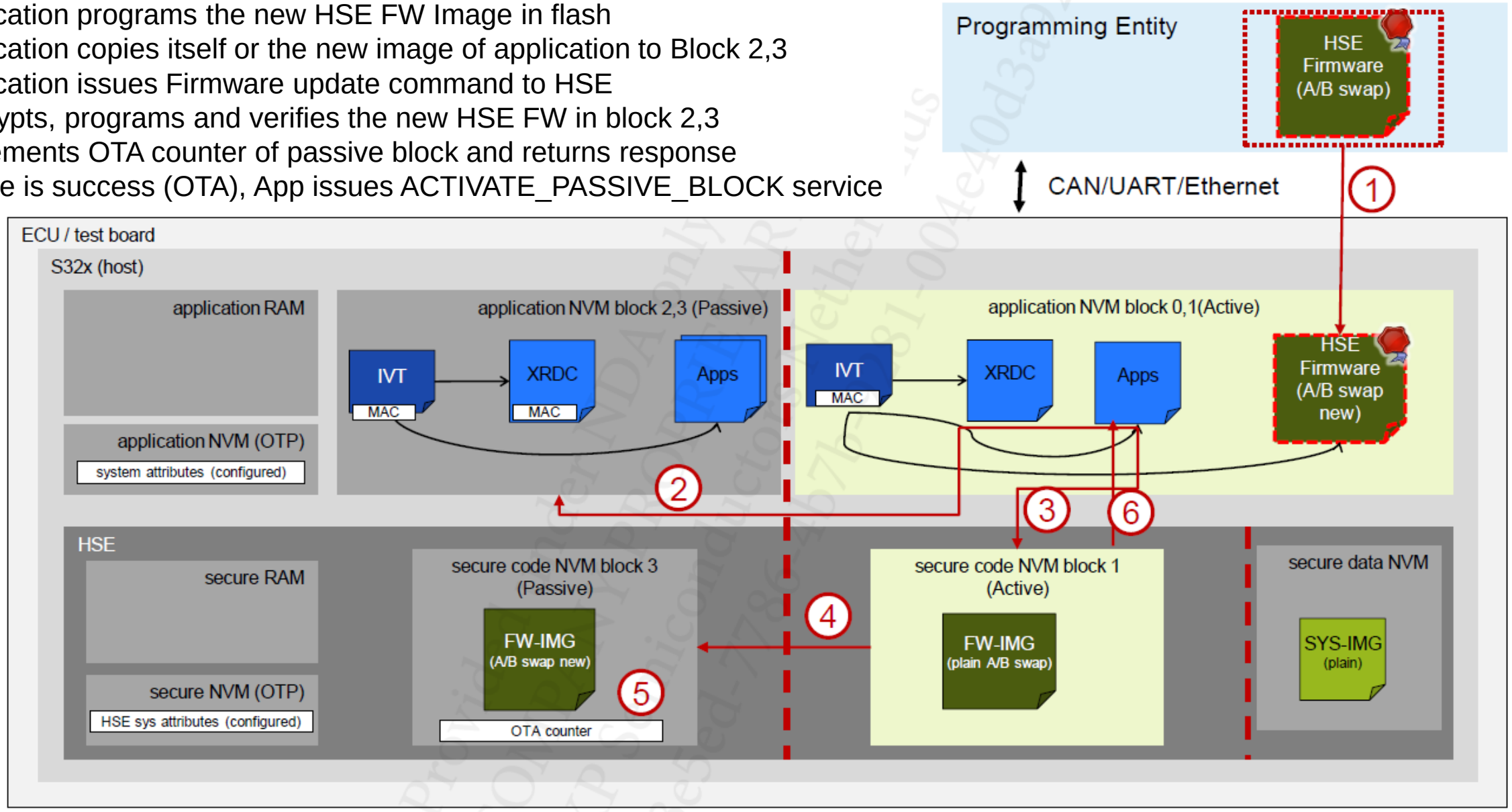
1. The Application receives a new image of A/B swap HSE Firmware (pink image) from a Programming Entity and store it in its NVM
2. The Application copies itself and IVT in block 0,1 NVM
3. The Application issues firmware update service request to HSE
4. The HSE decrypts, verify and copies the A/B swap HSE firmware to block 1 NVM.
5. HSE programs the A/B swap counter value
6. HSE program the A/B Swap enable flag in UTEST
7. HSE FW sends the response to Application.
8. Application issue a reset to switch to Active block.

FROM FULL MEM TO A/B SWAP

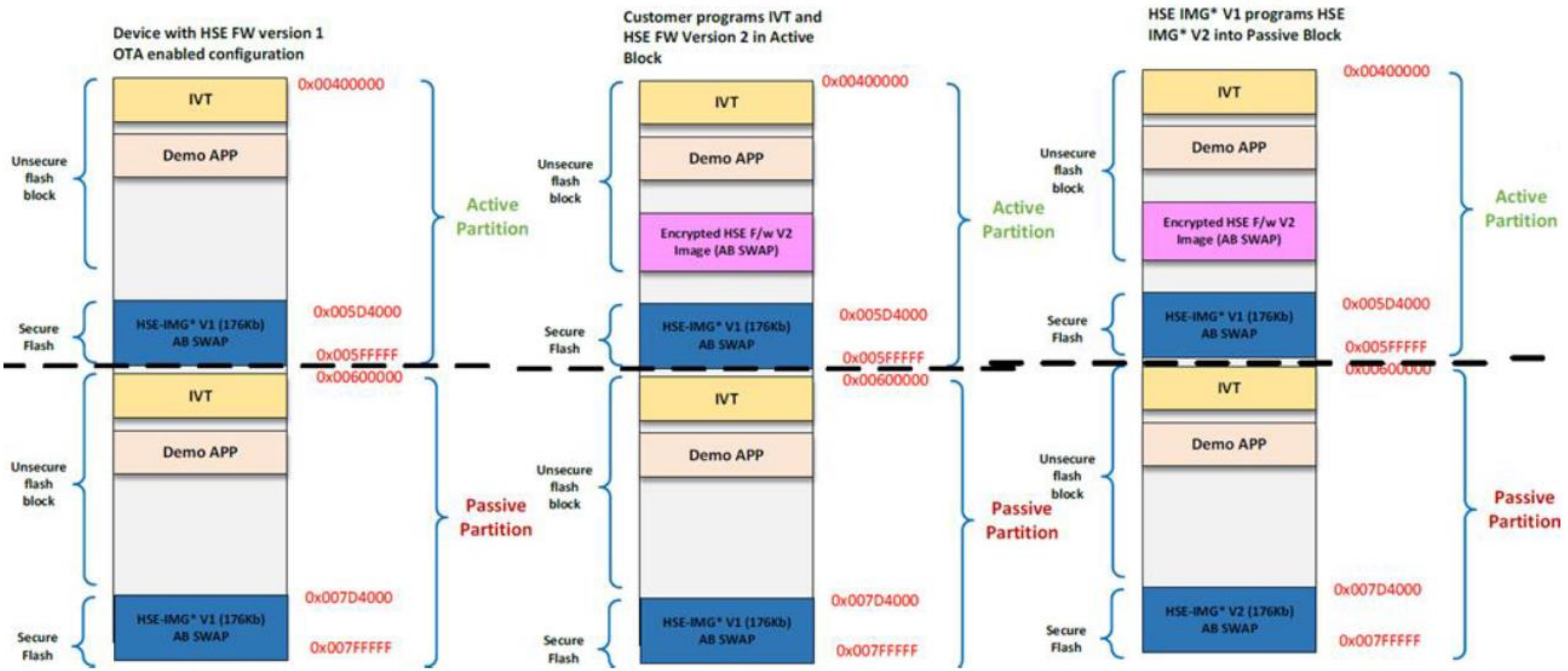


HSE FIRMWARE UPDATE FROM A/B SWAP TO A/B SWAP

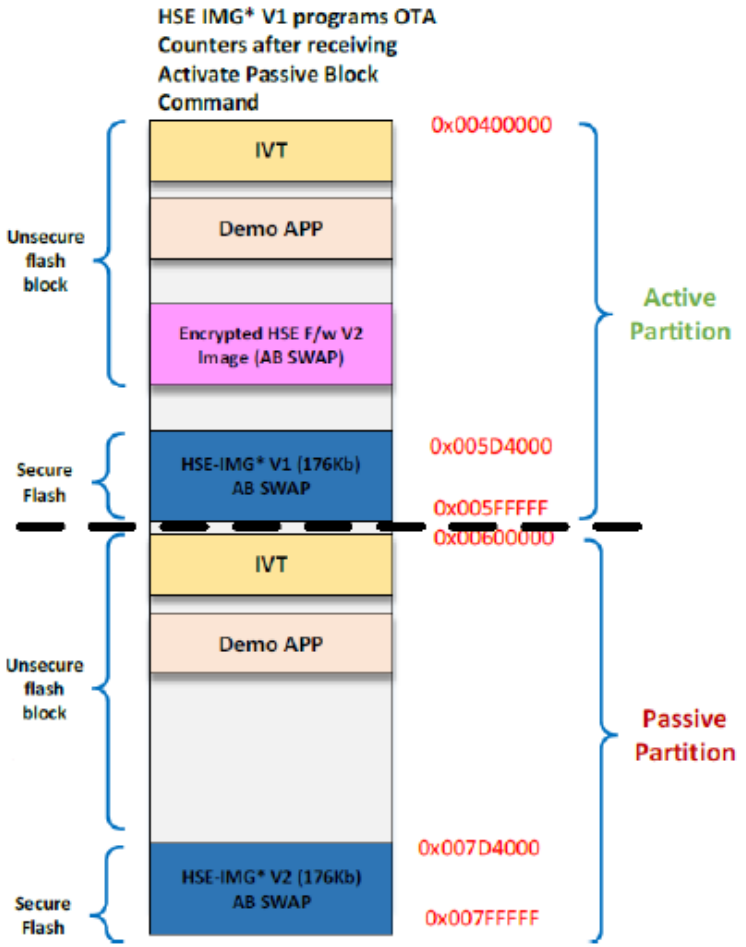
- The Application programs the new HSE FW Image in flash
- The Application copies itself or the new image of application to Block 2,3
- The Application issues Firmware update command to HSE
- HSE decrypts, programs and verifies the new HSE FW in block 2,3
- HSE increments OTA counter of passive block and returns response
- If response is success (OTA), App issues ACTIVATE_PASSIVE_BLOCK service and reset.

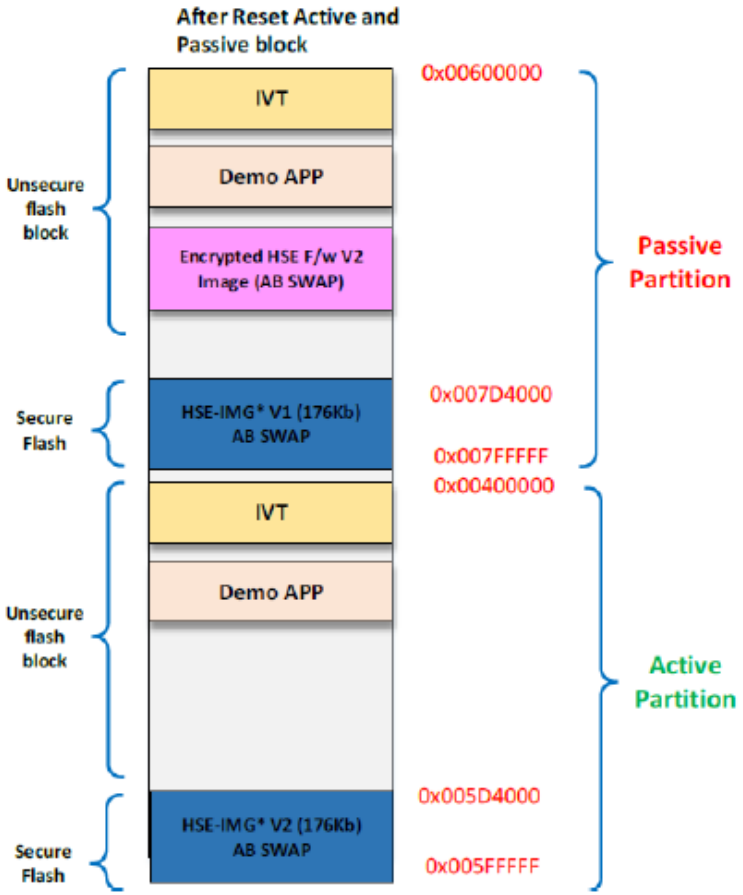


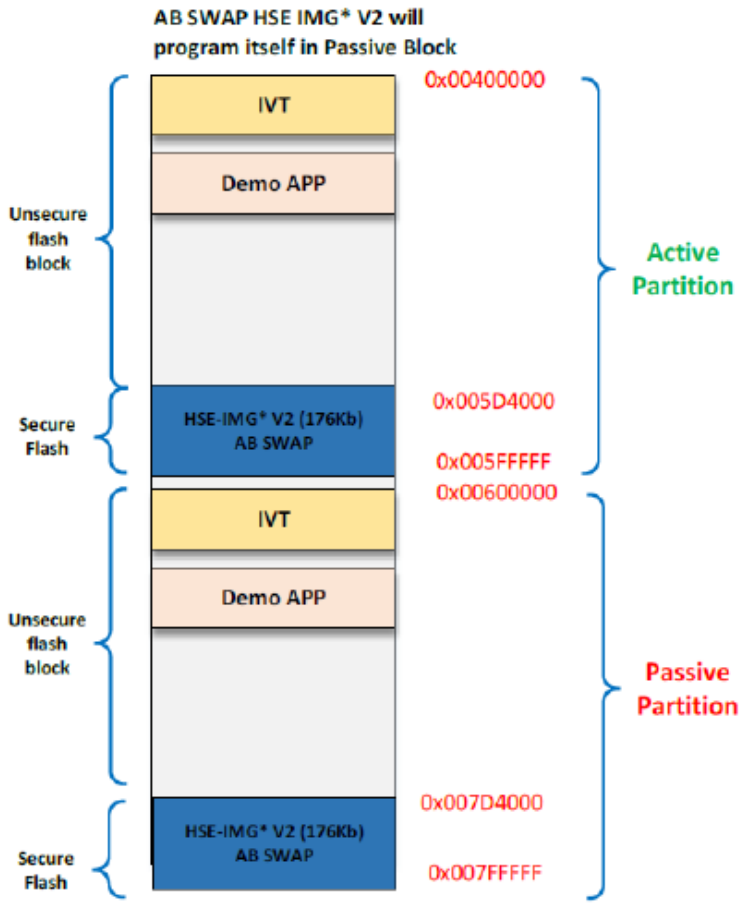
FROM A/B SWAP TO A/B SWAP



FROM A/B SWAP TO A/B SWAP







HSE – Install Firmware Demo – Method2

S32K3x4 Demo

- S32K344_HSE_FW_INSTALL_Demo_V010.zip
- S32K344_HSE_FW_INSTALL_Demo_V010_ABSwap.zip
- S32K344_HSE_FW_INSTALL_Demo_V012.zip
- S32K344_HSE_FW_INSTALL_Demo_V012_ABSwap.zip
- S32K344_HSE_FW_INSTALL_Demo_V110.zip
- S32K344_HSE_FW_INSTALL_Demo_V110_ABSwap.zip

The installation of AB swap FW, update Full mem -> AB swap , is irreversible,

The AB swap device cannot change to Full mem, the OTA enable Flag is in UTEST(OTP- one time program)

hsefwinstallfors32k3.zip

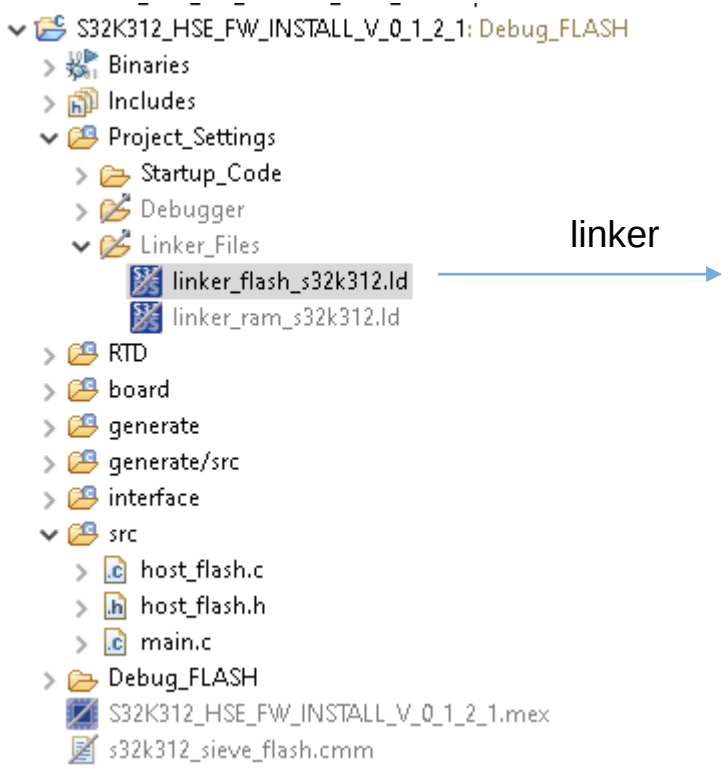
S32K312 Demo

- S32K312_HSE_FW_INSTALL_V_0_1_2_1: Debug_FLASH
  - Binaries
  - Includes
  - Project_Settings
    - Startup_Code
    - Debugger
    - Linker_Files
      - linker_flash_s32k312.ld
      - linker_ram_s32k312.ld
  - RTD
  - board
  - generate
  - generate/src
  - interface
  - src
    - host_flash.c
    - host_flash.h
    - main.c
  - Debug_FLASH
  - S32K312_HSE_FW_INSTALL_V_0_1_2_1.mex
  - s32k312_sieve_flash.cmm

S32K312_HSE_FW_INSTALL_V_0_1_2_1_CUSTOMER.zip

HSE – INSTALL HSE-FW

For K312 install HSE FW
– full mem



Make sure HSE FW path is correct

linker_flash_s32k312.ld | main.c

```
52 }
53
54 /* */
55 TARGET(binary)
56 INPUT (C:\NXP\HSE_FW_S32K3XX_0_1_2_1\hse_full_mem\hse\bin\s32k3x2_hse_fw_0.13.0_1.2.1_pb220205.bin.pink)
57 OUTPUT_FORMAT(default)
58 /* */
59
60 /*
61 TARGET(binary)
62 INPUT (C:\NXP\HSE_FW_S32K3XX_0_1_2_0\hse_full_mem\hse\bin\s32k3x2_hse_fw_0.13.0_1.2.0_pb211228.bin.pink)
63 OUTPUT_FORMAT(default)
64 */
65
66 /*
67 TARGET(binary)
68 INPUT (C:\NXP\HSE_FW_S32K3X2_0_0_11_0\hse_full_mem\hse\bin\s32k3x2_hse_fw_0.13.0_0.11.0_pb210726.bin.pink)
69 OUTPUT_FORMAT(default)
70 */
```

```
72 SECTIONS
73 {
74
75     .hse_bin :
76     {
77         . = ALIGN (0x4);
78           hse_bin_start__ = .;
79
80         /* */
81         C:\NXP\HSE_FW_S32K3XX_0_1_2_1\hse_full_mem\hse\bin\s32k3x2_hse_fw_0.13.0_1.2.1_pb220205.bin.pink (.data)
82         /* */
83
84         /*
85         C:\NXP\HSE_FW_S32K3XX_0_1_2_0\hse_full_mem\hse\bin\s32k3x2_hse_fw_0.13.0_1.2.0_pb211228.bin.pink (.data)
86         */
87
88         /*
89         C:\NXP\HSE_FW_S32K3X2_0_0_11_0\hse_full_mem\hse\bin\s32k3x2_hse_fw_0.13.0_0.11.0_pb210726.bin.pink (.data)
90         */
91         . = ALIGN (0x4);
92         __hse_bin_end__ = .;
93     } > HSE_BINARY
94
95     __HSE_BIN_START = ORIGIN(HSE_BINARY);
96     __HSE_BIN_SIZE = __hse_bin_end__ - __hse_bin_start__;
```

HSE – INSTALL HSE-FW

Build and debug the project

```
    /* enable utest HSE usage, OTP operation */
#if 1
    while ( FALSE == checkHseFwFeatureFlagEnabled())
    {
        /* user has requested to program HSE FW feature flag */
        HseResponse = EnableHSEFWUsage();
    }
#endif

    /* config clock option B , write DCF and FXOSC, OTP operation */
#if 0
    uint8_t isWriteDCF = 0 ;
    if( 1 == isWriteDCF )
    {
        /* enable utest fxosc usage and dcf clock option
         * with pll enable in ivt.beq, secure boot verification can be accelerated */
        EnableFXOSCUsage();

        ChangeDcfClockOption();
    }
#endif

    /* Clocks Configuration */
    if( 1 == isConfigClock )
    {
        Clock_Ip_Init(&Mcu_aClockConfigPB[0]);
    }

    /* get HSE FW version */
    while((HSE_STATUS_INIT_OK & Hse_Ip_GetHseStatus(0)) == 0)
    {

    }

    /*  */
    HSE_GetVersion_Example(&HseFwVersion);
```

OTP – one time program

1. Enable HSE FW usage, OTP operation

2. config clock option B , write DCF and FXOSC, OTP operation

3. Check HSE MU status, stuck here if without HSE FW

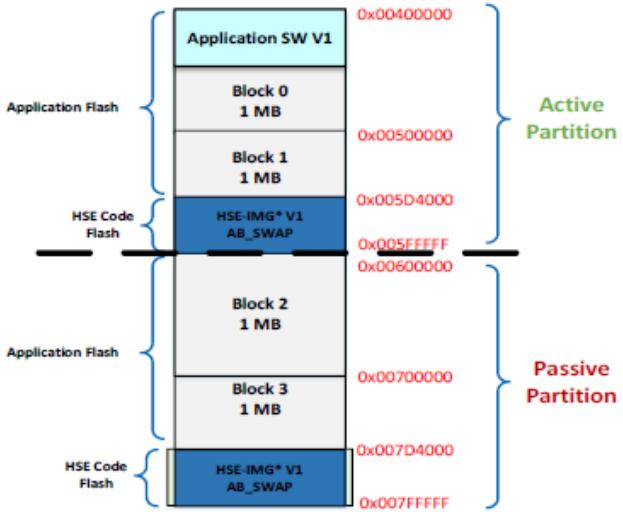
Need to reset for SBAF install the HSE FW

4. After reset and pass the status check, Get the HSE FW version means the HSE FW is already installed

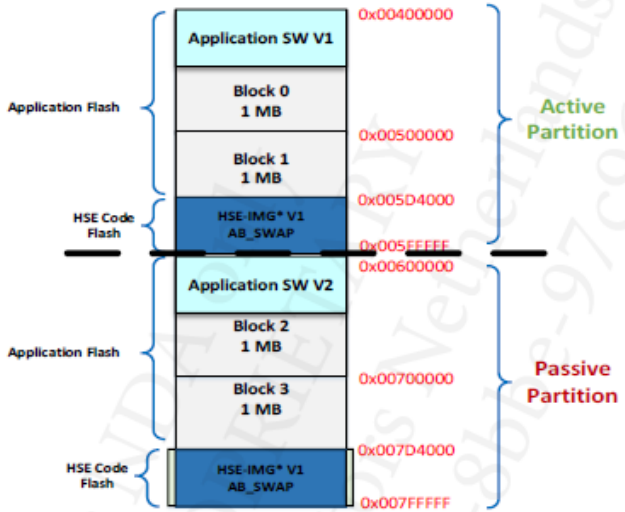
HSE OTA

Update Application for A/B SWAP

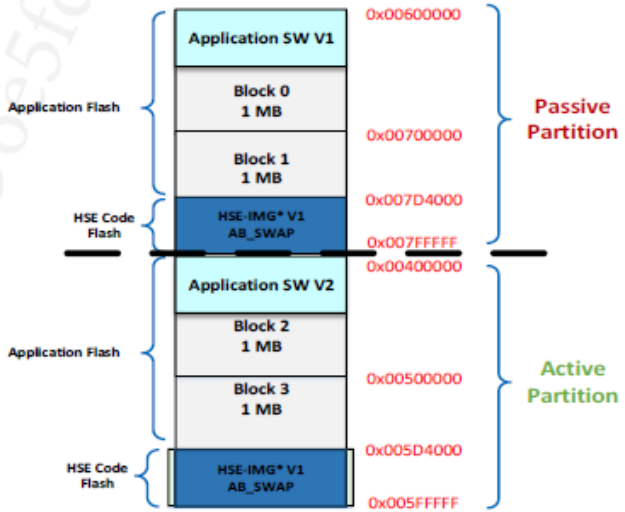
App SW V1 is programmed in Device with Block 0 and 1 is Active Partition



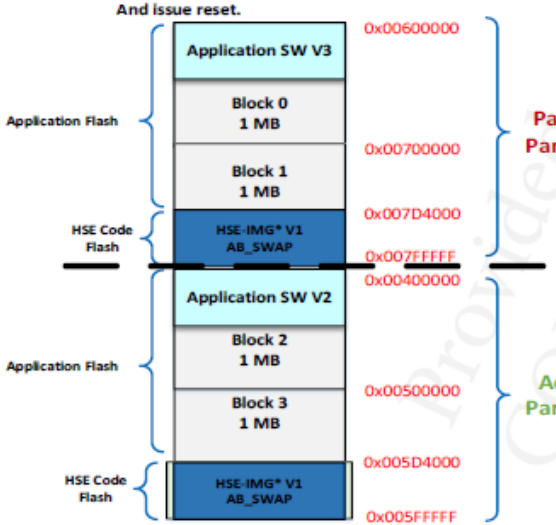
Customer programs Application SW V2 in Passive Partition then call srv_id HSE_SRV_ID_ACTIVATE_PASSIVE_BLOCK, And issue reset.



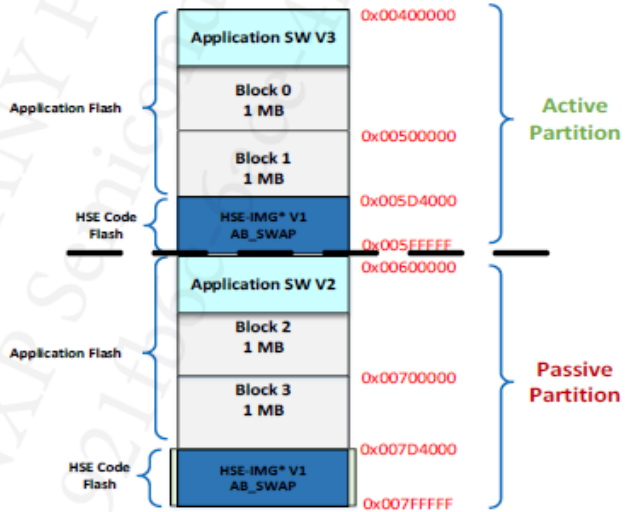
After Reset, Application SW V2 gets control and becomes Active.



Customer programs Application SW V3 in Passive Partition then call srv_id HSE_SRV_ID_ACTIVATE_PASSIVE_BLOCK, And issue reset.



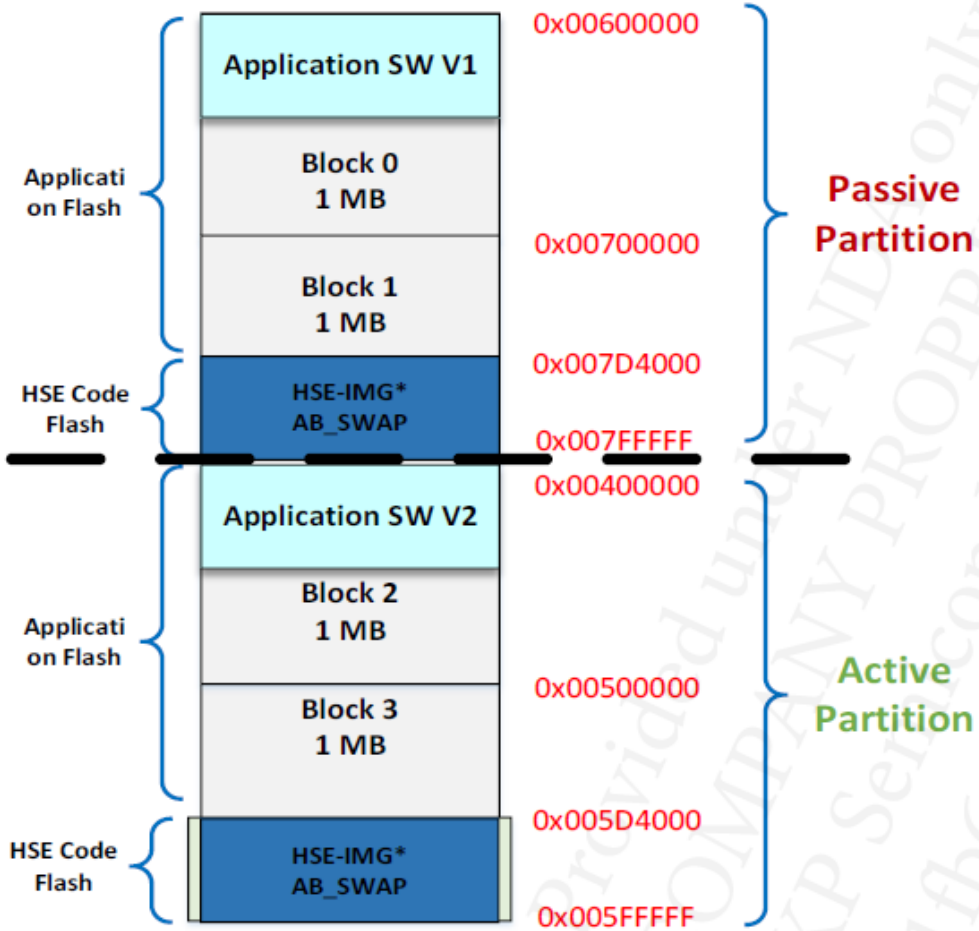
After Reset, Application SW V3 gets control and becomes Active.



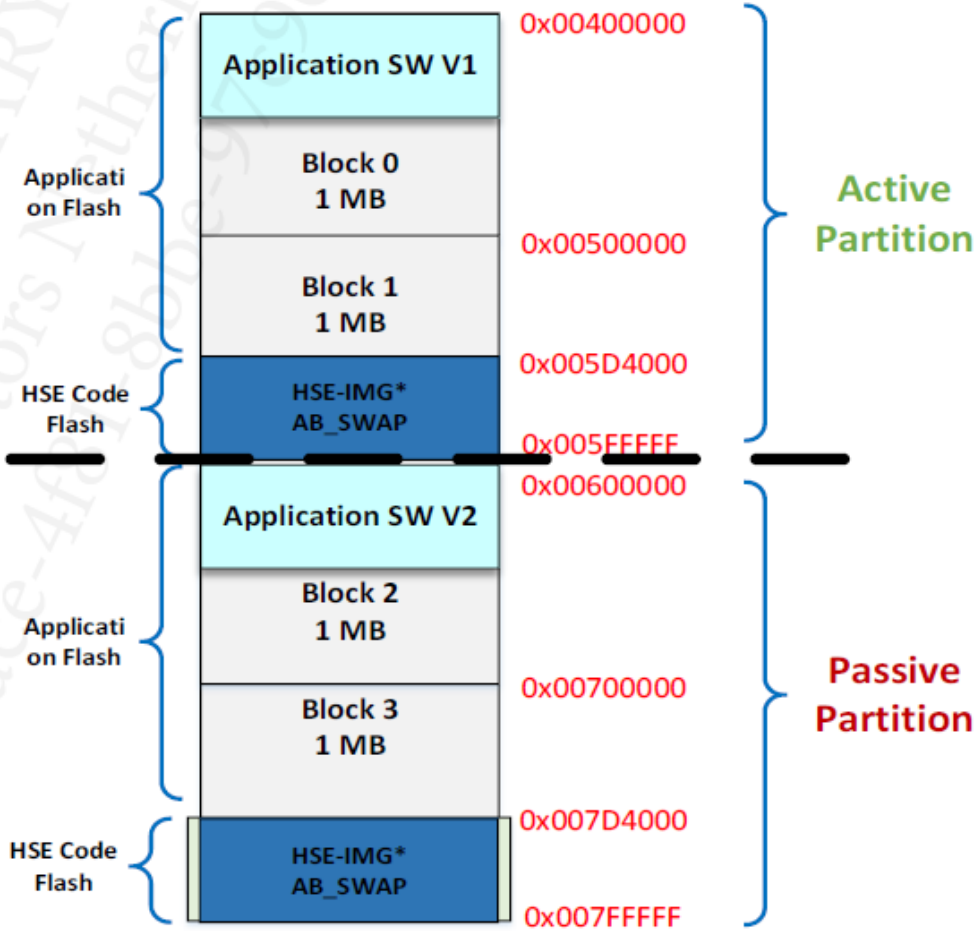
Note:
1. Application code must build for Block 0 and Block 1 address space always.
2. Application must issue Reset after Passive Partition activation command. i.e. Between 2 consecutives Passive Partition activation command reset must be given.

Rollback Application for A/B SWAP

Application SW V2 running on Active Partition[2-3].
Application wants to rollback to V1 so call srv_id
HSE_SRV_ID_ACTIVATE_PASSIVE_BLOCK
and then issues Reset.



After Reset, Application SW V1 gets control as Block[0-1] becomes Active Partition.



SW32K3x4 OTA DEMO

NXP > Design > Automotive SW - S32K3 Reference Software > **SW32K3_OTADEMO_0.8.0_D2203 : Files**

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

Product Download

SW32K3_OTADEMO_0.8.0_D2203

Files | License Keys | Notes

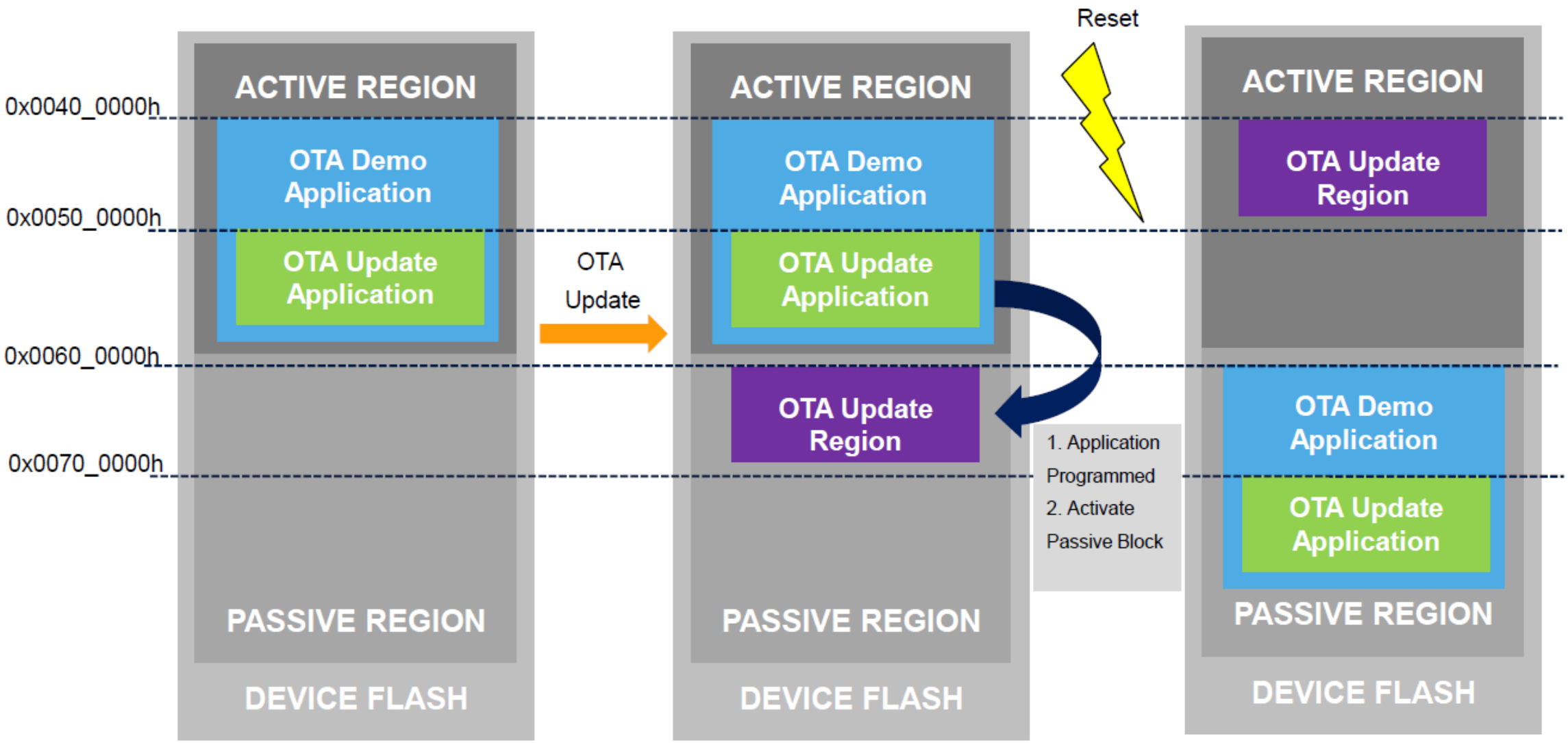
Download Help

Show All Files

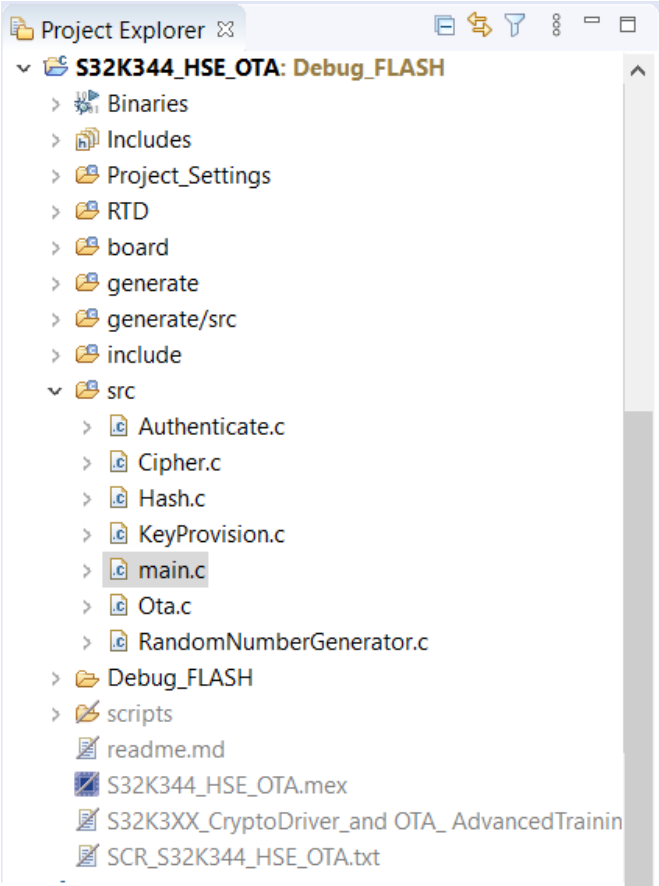
3 Files

File Description	File Size	File Name
SCR_S32K344_HSE_OTA.txt	390 bytes	SCR_S32K344_HSE_OTA.txt
SW32K3_OTADEMO_0.8.0_D2203.exe	1.7 MB	SW32K3_OTADEMO_0.8.0_D2203.exe
SW32K3_OTADEMO_0.8.0_ReleaseNotes.pdf	162.1 KB	SW32K3_OTADEMO_0.8.0_ReleaseNotes.pdf

SW32K3x4 OTA DEMO



SW32K3x4 OTA DEMO



main.c

```
/*    Initialize Crypto Driver
/* ==========================================
Crypto_Init(NULL_PTR);

/* ==========================================
/*    Random Number Generation demo
/* ==========================================
//Demo_RandomNumberGeneration();

/* ==========================================
/*    Key Provisioning  demo
/* ==========================================
//Demo_KeyProvisioning();

/* ==========================================
/*   Authenticate Application demo
/* ==========================================
//Demo_Authentication();

/* ==========================================
/*   Symmetric Cipher demo
/* ==========================================
//Demo_SymmetricCipher();

/* ==========================================
/*   Hash demo
/* ==========================================
Demo_Hash();

/* ==========================================
/*   Application OTA Update demo
/* ==========================================
Demo_OtaUpdate();

/* After next reset the code in passive blocks is execute
```

HSE RECOVERY MODE

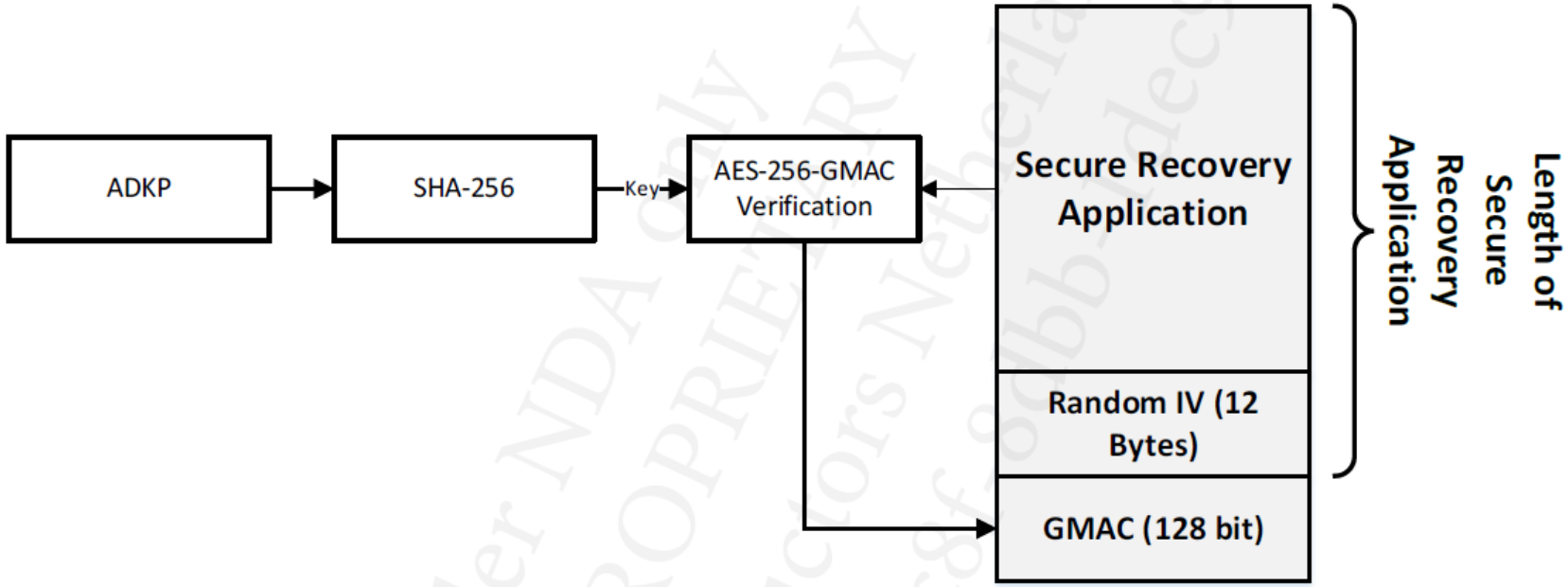
Recovery Mode – Secure Recovery Mode

The recovery mode allows the host to recover from following abnormal situations

- ✓ IVT is not present or corrupted
- ✓ There is 8 consecutive functional or destructive resets.
- ✓ Secure boot authentication of application image failed.
- ✓ Can be disabled by clearing bit number 22 and 23 in register “DCMRWP1”.

Recovery Mode – Secure Recovery Mode

- ✓ In Secure Recovery mode, the HSE subsystem boot the Secure Recovery Application after its authenticity vs. ADKP is confirmed (see the below figure).
- ✓ eeds to be enabled by *HSE_SECURE_RECOVERY_CONFIG_ATTR_ID*.
- ✓ The start address and the size of the secure recovery application must be provided in the IVT and Secure Recovery Application includes random IV.



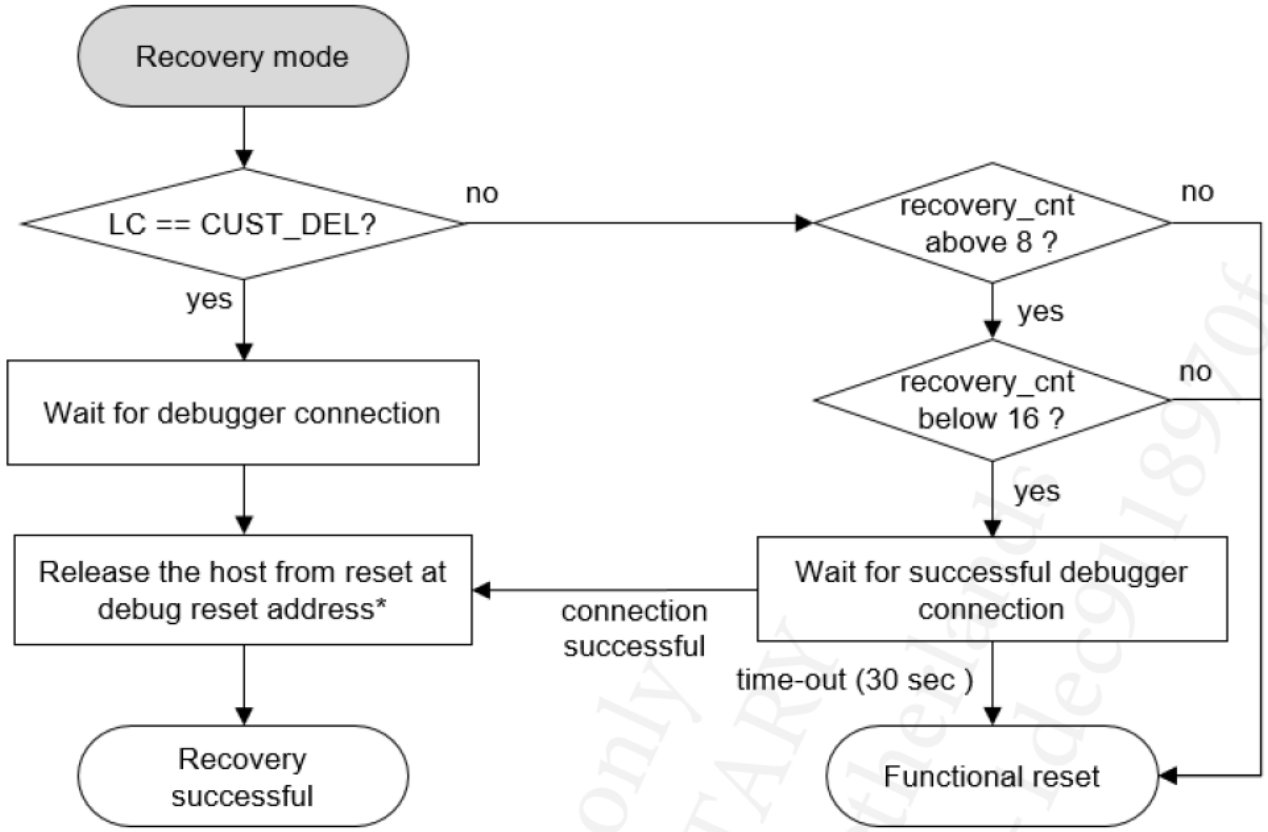
Offset in IVT

0x40	4	Executable	Start Address of Application Core for Secure Recovery mode.	Pointer

Recovery Mode – JTAG Recovery Mode

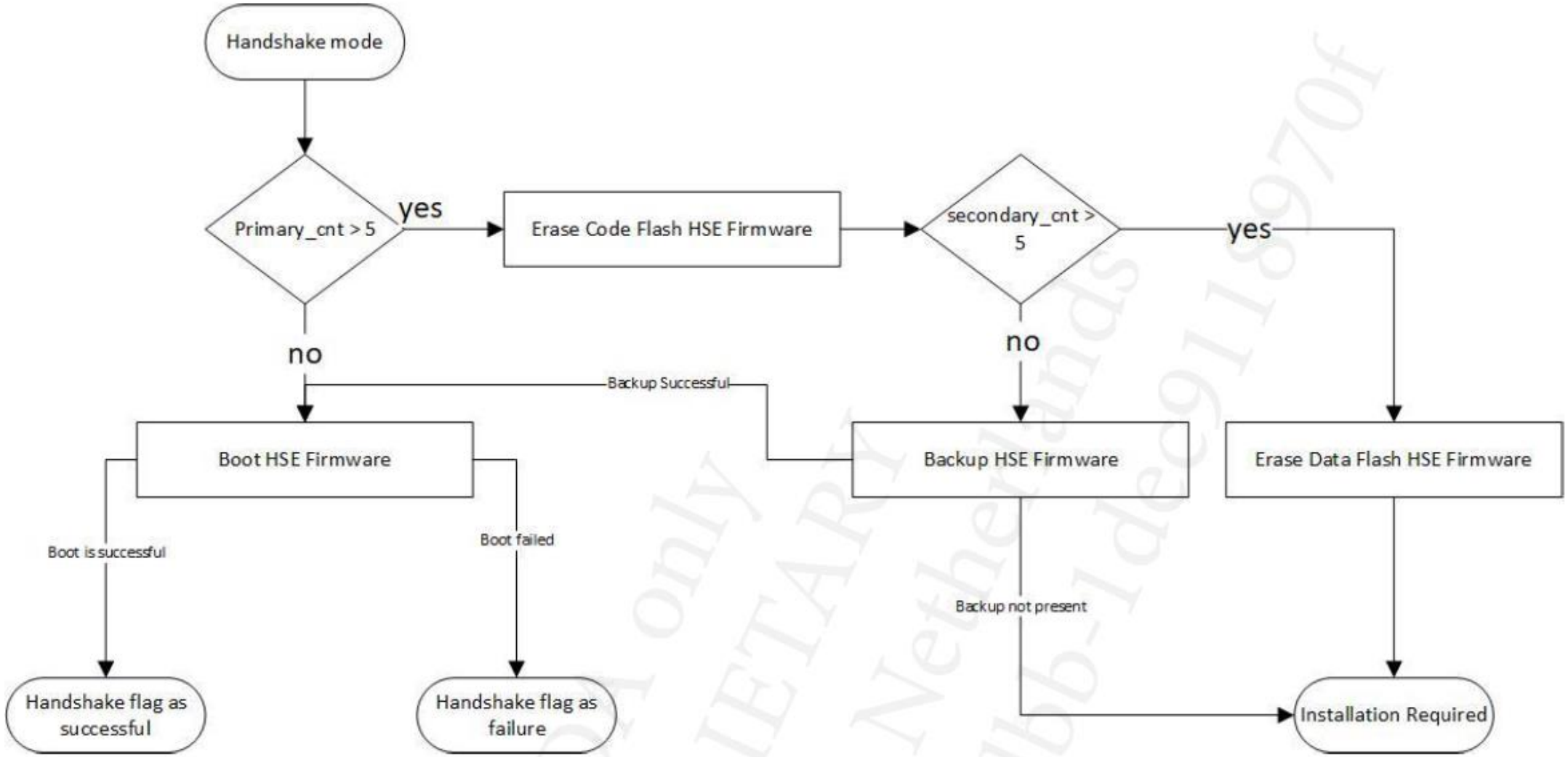
If the Secure Recovery Mode fails, the device enters JTAG Based Recovery Mode.

the HSE subsystem waits for debugger connection (that must be authenticated when LC is OEM_PROD or IN_FIELD) and then releases the host from reset at a predefined address 0x20400100 to a ram application with application core in sleeping mode.



HSE FW HANDSHAKE

HSE FW HANDSHAKE



REINSTLL FW VIA MU IN CASE A/B SWAP

MU Restore FW

1. Root cause

In most cases, HSE FW erased caused by incorrect clock configuration. The proper operation of the HSE subsystem depends on the correct configuration of the clocks CORE_CLK, HSE_CLK, AIPS_SLOW_CLK, AIPS_PLAT_CLK, etc. Therefore, users need to follow the 23.7.2 clock option in the S32Kxx-RM, otherwise the HSE may not operate properly, or even HSE FW will be erased.

But I also found some other case, like clear FES-POR without DCF record config (K312), K312 enable "PLL_ENABLE" in IVT.

More detail can refer to S32K3-RM and HSE-RM, I also write mention this in HSE QSG.

Full mem firmware recovery is similar to AB swap, you can refer to
S32K3 HSE installation using MU Interface - NXP Community

2. Prepare

Need to prepare a HSE FW install project to download the firmware to FLASH, and then add a loop, avoid running the program to another location, click "run" in debug window each time after writing to the MU RR/TR register

3. HSE handshake mechanism

After POR, the MU -FSR reg all "0", the HSE is not working

Check the HSE GPR

14.2.6.2 HSE GPR Register 3

Secure BAF updates status bits on HSE GPR Register 3 (0x4039C028) as explained in below table.

Table 136: Status Bits on HSE GPR Register 3 (0x4039C028)

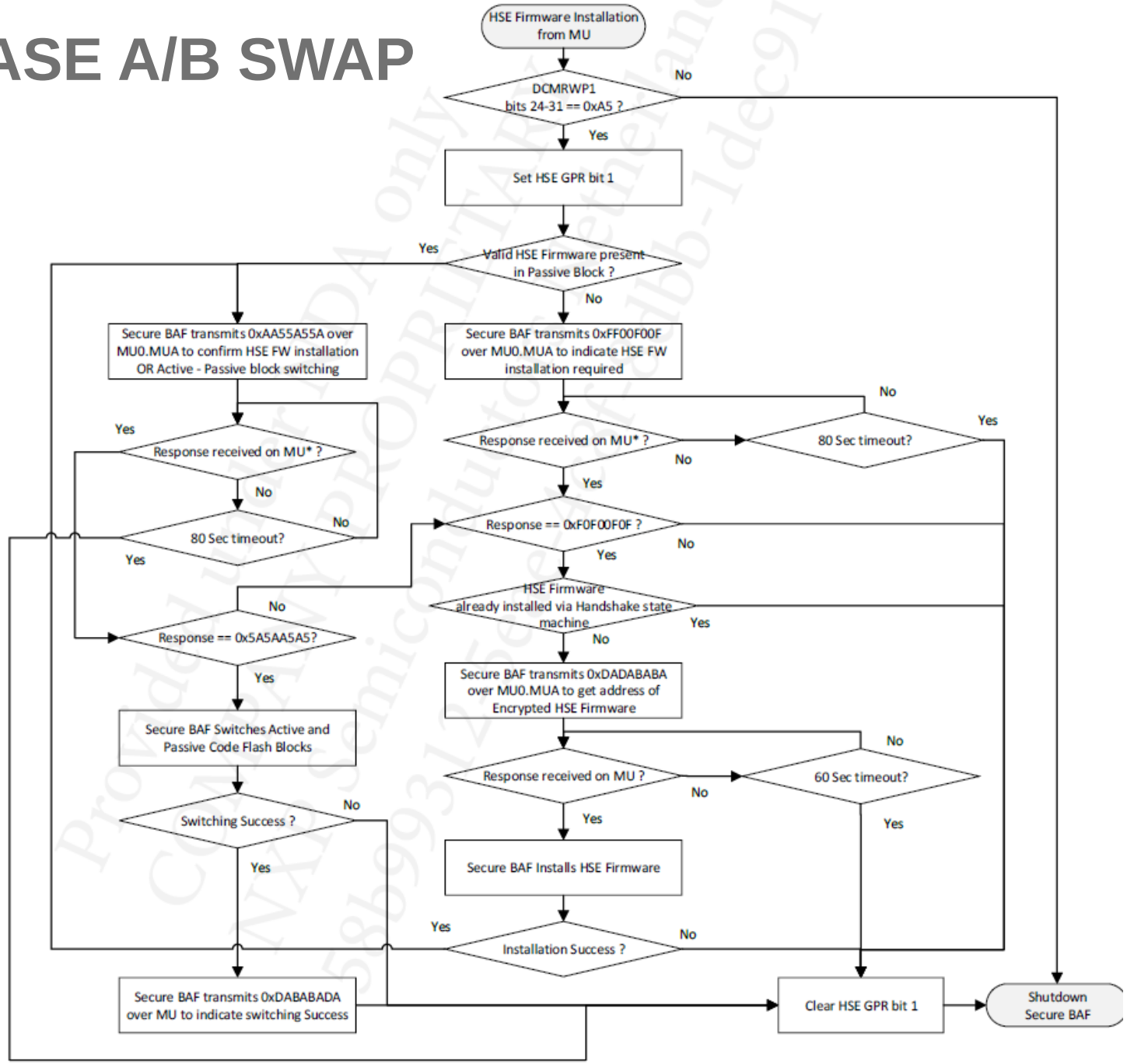
Bit #	Description
31...	Reserved
5	Application cores booted in Recovery mode by SBAF.
4	No HSE Firmware is present in Device due to Erase performed by SBAF Handshake logic. This bit resets on presence of valid HSE Firmware.
3	HSE Firmware from Data flash area is erased by SBAF Handshake logic in current reset cycle.
2	HSE Firmware from code flash area is erased by SBAF Handshake logic in current reset cycle.
1	MU interface is enabled for installation of HSE Firmware.
0	HSE FW is present and SBAF Booted HSE Firmware

HSE_B Firmware Reference Manual, Rev 1.2, 01/2022

NXP Semiconductors 209

And the HSE GPR 3 register bit 0 is also 0 (normal should be 0x00000001), so it is obvious that the firmware is automatically erased by SBAF through the handshake mechanism.

Handshake mechanism in HSE-RM,



KEY MANAGEMENT & CRYPTO SERVICES

HSE KEY ATTRIBUTES & STORAGE LOCATIONS

Key Handle
Uniquely identifies a key for host usage

Storage Location
One-time configuration

Owner
Tier1 or OEM

ROM catalog	NVM catalog	RAM catalog
Device's Root-of-Trust	Long-term application keys	Temporary keys

Type + Size*

Secret key k

AES
SHE
HMAC
SHARED_SECRET

Public key k_{pub}

RSA_PUB
RSA_PUB_EXT
ECC_PUB
ECC_PUB_EXT
DH_PUB

Key pair (k_{priv}, k_{pub})

RSA_PAIR
ECC_PAIR
DH_PAIR

HSE Key

Update Counter
Provides rollback protection

Usage
Restricts the usage to specific operations

ENCRYPT
DECRYPT
SIGN
VERIFY
EXCHANGE
DERIVE
KEY_PROVISION
AUTHORIZATION

SMR verification map
Only usable if e.g. SMR1 and SMR31 are verified

SMR0	SMR1	...	SMR30	SMR31
-	X	...	-	X

Access restrictions
Restricts updates & access possibilities

WRITE_PROT
DEBUG_PROT
EXPORTABLE

HSE KEY CATALOGS

MU0	MU1
-	X

OWNER_CUST
OWNER_OEM
OWNER_ANY (RAM)

AES
SHE
HMAC
SHARED_SECRET
RSA_PAIR
RSA_PUB
ECC_PAIR
ECC_PUB
DH_PAIR
DH_PUB

RSA_PUB_EXT
ECC_PUB_EXT

ECC keys on
- Prime field ($GF(p)$)
- Any Weierstrass form
- Ed25519
- Curve25519

ROM / NVM / RAM Key Catalog

Key Group #0

MU Mask	Key Owner	Key Type	Nb of Slots	Max Key Size

Slot #0	Attributes	Key Values
Slot #1	Attributes	Key Values
⋮	⋮	⋮
Slot #n	Attributes	Key Values

Key Group #1

MU Mask	Key Owner	Key Type	Nb of Slots	Max Key Size

Slot #0	Attributes	Pointers (Pub Key & Cert)+Tag
Slot #1	Attributes	Pointers (Pub Key & Cert)+Tag
⋮	⋮	⋮

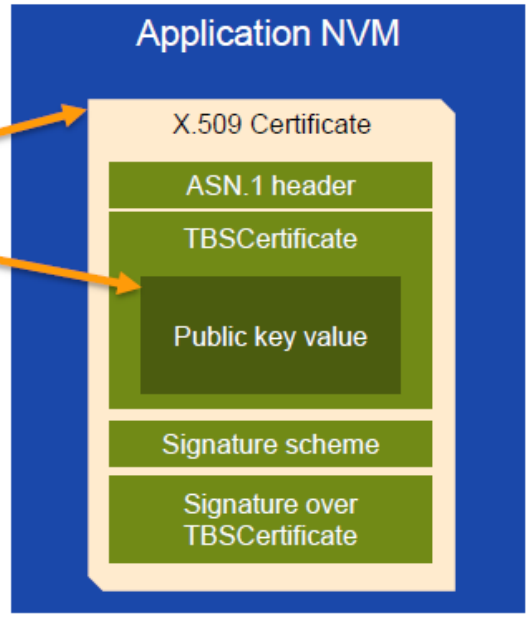
⋮

Key Group #m

Key slots can be erased (Super User only)

- Key Groups are **linked to one or more MUs** (key can be accessed using that MU interface)
- **ROM key catalog**: defined / provisioned by NXP
- **NVM / RAM key catalogs**: one-time configurable
  - Number of groups & key slots limited by HSE memory size
- Key handle = $CONCAT$(0x00, catalog type, group index, slot index)

- Keys can refer to standard / custom **certificates**
  - Pointers to public key values provided during key import
  - Certificate parsing handled by the application
  - Key value modification after import render the key unusable



HSE KEY CATALOG FORMAT EXAMPLE

The key catalog formatting takes as inputs:

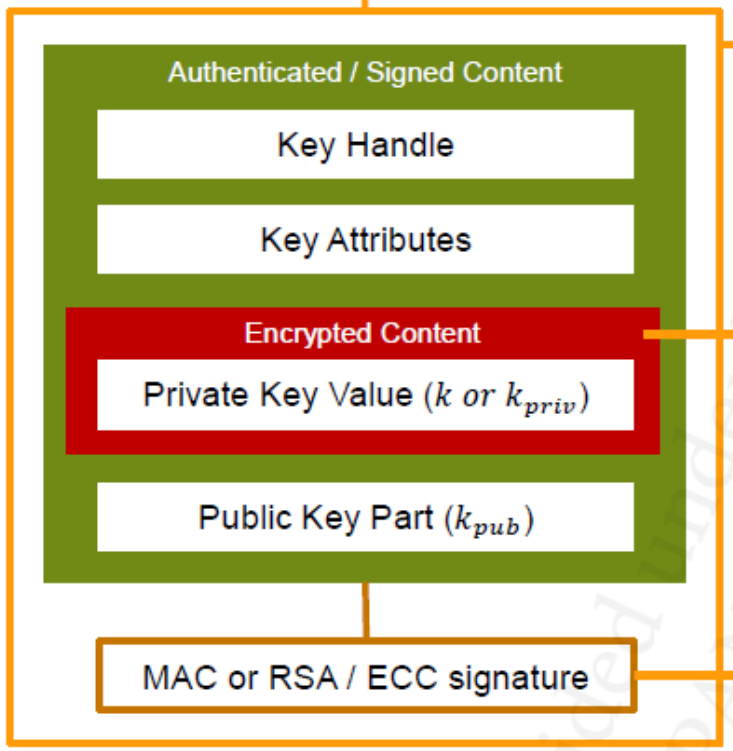
- ✓ A pointer to the NVM key catalog configuration
- ✓ A pointer to the RAM key catalog configuration

```
hseKeyGroupCfgEntry_t my_NVM_key_catalog[] = {
/* AES keys */
    {HSE_MU0_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_AES,        10,  128},
    {HSE_MU0_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_AES,        10,  256},
/* ECC keys */
    {HSE_MU0_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_ECC_PAIR,    2,  256},
    {HSE_MU0_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_ECC_PUB,     5,  256},
/* RSA keys */
    {HSE_MU0_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_RSA_PAIR,    2, 2048},
    {HSE_MU0_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_RSA_PUB,    10, 4096},

    {0, 0, 0, 0, 0}
};

hseKeyGroupCfgEntry_t my_RAM_key_catalog[] = {
/* ECC keys */
    {HSE_MU0_MASK, HSE_KEY_OWNER_ANY, HSE_KEY_TYPE_ECC_PUB,     5,  256},
/* AES keys */
    {HSE_MU0_MASK, HSE_KEY_OWNER_ANY, HSE_KEY_TYPE_AES,        10,  128},
    {HSE_MU0_MASK, HSE_KEY_OWNER_ANY, HSE_KEY_TYPE_AES,        10,  256},
```

KEY Handle

31 ~ 24	23 ~ 16	15 ~ 8	7 ~ 0
0	Key catalog ID	Key group index	Key slot index

KEY IMPORT / EXPORT



Imported / Exported Key
K: (k or k_{priv}) and/or k_{pub}

Access for Import	Access for Export
WRITE_PROT	EXPORTABLE

Encryption Key K_e

Must be provisioned in NVM catalog

Usage for Import	Usage for Export
DECRYPT	ENCRYPT
KEY_PROVISION	KEY_PROVISION

Authentication Key K_a

Must be provisioned in NVM catalog

Usage for Import	Usage for Export
VERIFY	SIGN
KEY_PROVISION	KEY_PROVISION

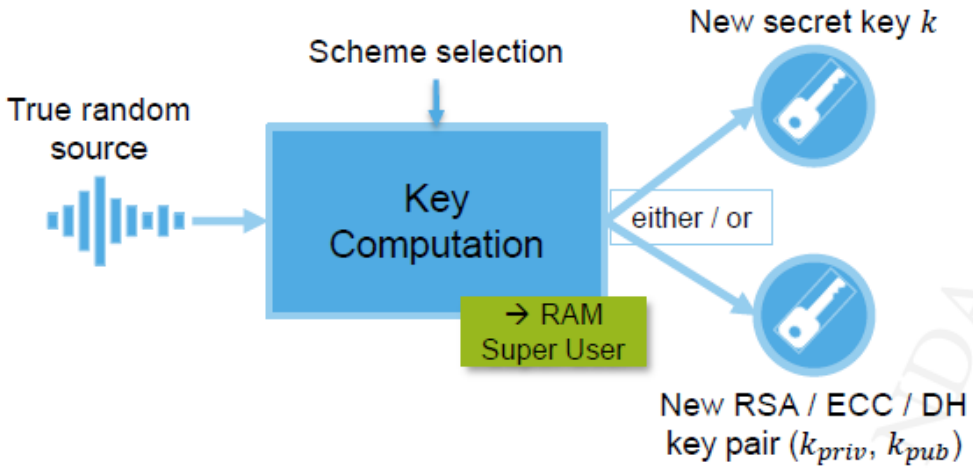
Same Owner
(Tier1 or OEM)

The KEY_PROVISION flag must be set for keys protecting the import/export operations

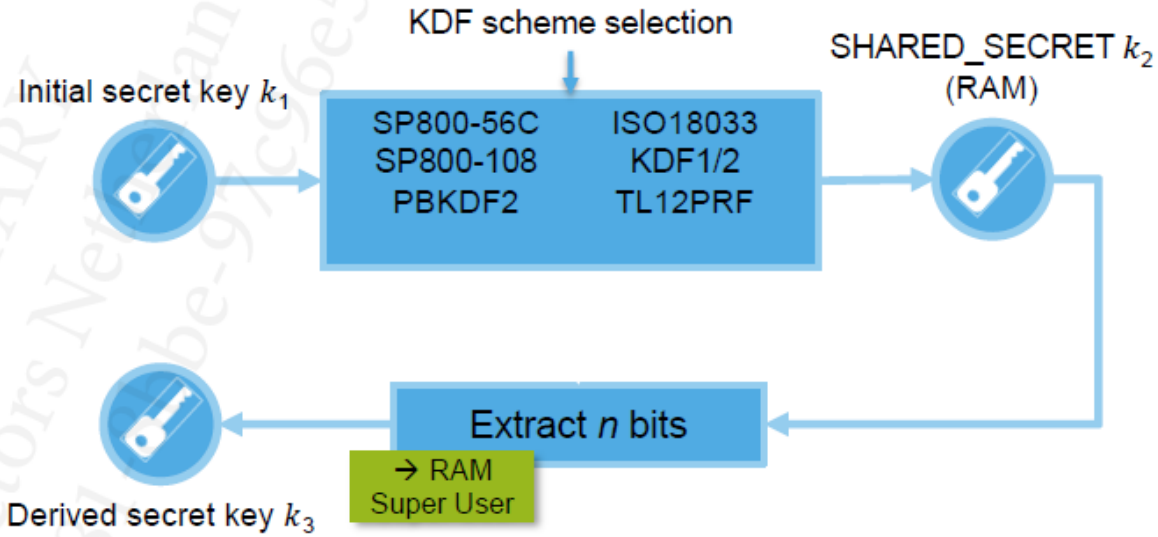
Encryption & Authentication Required Depending on Life Cycle and Host Rights

ADDITIONAL KEY PROVISIONING SERVICES

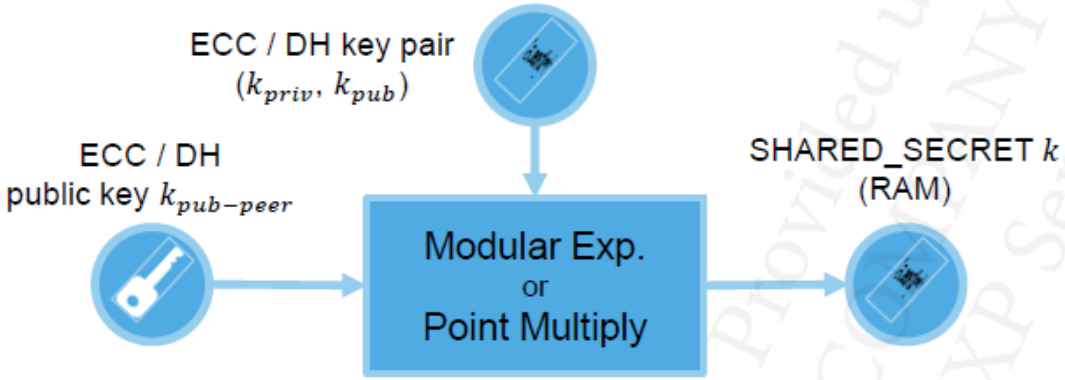
Key Generation



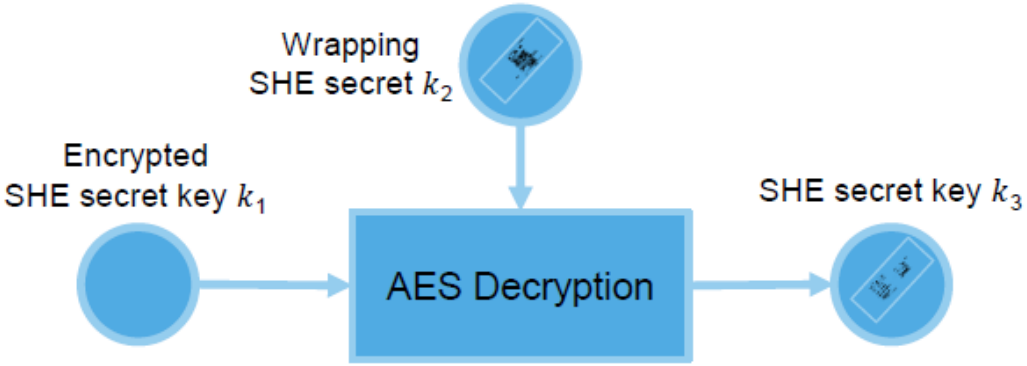
Key Derivation



Key Agreement



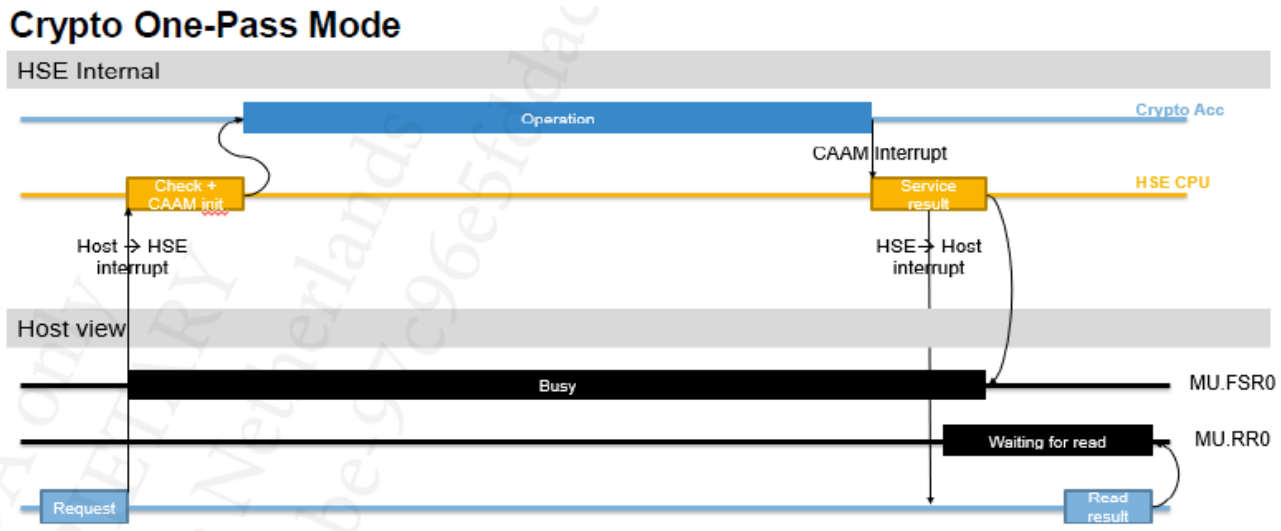
SHE Key Update Protocol

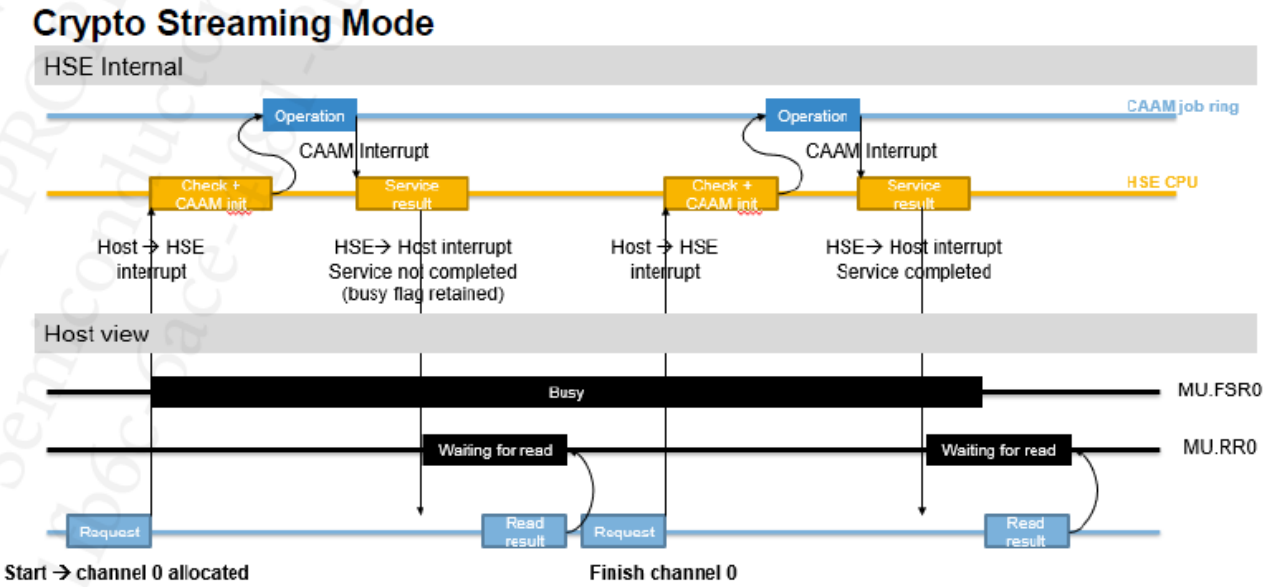


HSE CRYPTO SERVICES

CRYPTO services can be executed in two modes:

- **One Pass Mode**
  - Usually intended to process data in one-shot (e.g. Can messages,)
- **Streaming Mode**
  - Operation has **many phases: Start, Update, Update…, Finish**
  - **Context** is established during start of operation. (Key, IV, etc.). Stream context is identified by a unique ID.
  - Dependence between two consecutive Update operations
  - Used when data is not available, or it has big size (and need to be split in chucks)
  - Streaming context can be exported (encrypted & authenticated) to the host and then imported, when needed.

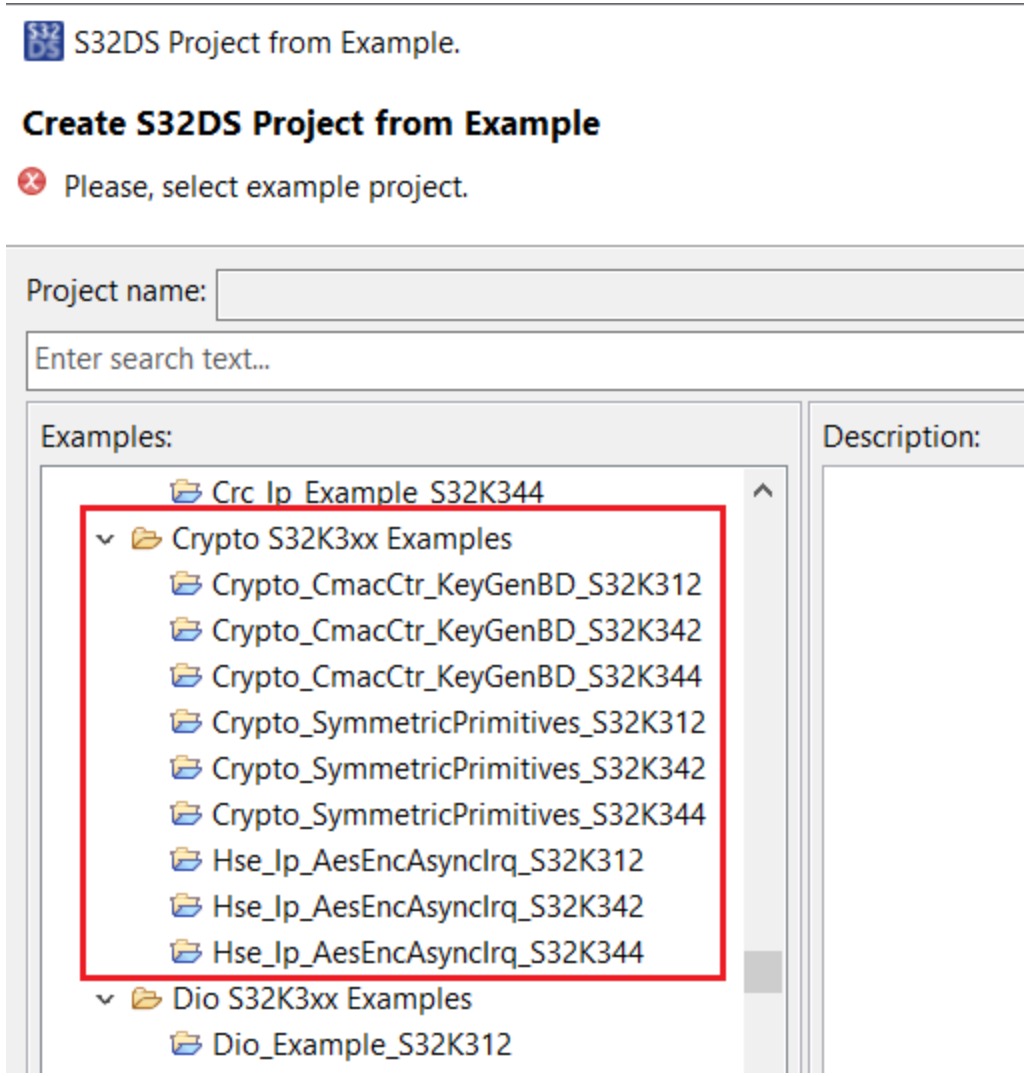




HSE CRYPTO SERVICES

HSE Service ID	HSE Service Data	Description
HSE_SRV_ID_HASH	hseHashSrv_t	Hash service (one-pass and streaming): * **SHA1** * **SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256** * **Miyaguchi-Preneel** compression function (SHE specification support)
HSE_SRV_ID_MAC	hseMacSrv_t	Request to generate/verify a Message Authentication Code (MAC): * **AES-CMAC, AES-GMAC** * **HMAC_(SHA1, all SHA224 and SHA256)**
HSE_SRV_ID_FAST_CMAC	hseFastCMACSrv_t	Low latency, high performance CMAC generate/verify request
HSE_SRV_ID_SYM_CIPHER	hseGetKeyInfoSrv_t	Symmetric encryption/decryption (one-pass and streaming): * **AES-128/-192/-256: ECB, CBC, CTR, OFB, CFB**
HSE_SRV_ID_AEAD	hseAeadSrv_t	AEAD encryption/decryption: * **AES-CCM-128/-192/-256 (one-pass, no streaming support)** * **AES-GCM-128/-192/-256 (one-pass and streaming)**
HSE_SRV_ID_SIGN	hseSignSrv_t	Request a Digital Signature Generation/Verification (one-pass and streaming): * **RSASAA_PSS (1024, 2048, 3072, 4096)** * **RSASAA_PKCS1-v1_5(1024, 2048, 3072, 4096)** * **ECDSA (all supported ECC curves)** * **EDDSA (for ED25519 curve)**
HSE_SRV_ID_RSA_CIPHER	hseRsaCipherSrv_t	RSA encryption/decryption: * **RSAES-PKCS1-v1_5 (1024, 2048, 3072, 4096)** * **RSAES-OEAP (1024, 2048, 3072, 4096)**

HSE CRYPTO DEMO



HSE CRYPTO DEMO

Project Explorer

- Crypto_SymmetricPrimitives_S32K344: Debug_FLASH
  - Binaries
  - Includes
  - Project_Settings
  - RTD
  - board
  - generate
  - generate/include
  - generate/src
  - src
    - main.c
  - Debug_FLASH
  - include
  - Configuration.mex
  - description.txt

main.c

```c
/* ==========================================================================
int main(void)
{
    Std_ReturnType RetVal;

    /* ==========================================================================
    /*    Initialization
    /* ==========================================================================
    /* Initialize OsIf component. It will be used to measure timeouts while
    OsIf_Init(NULL_PTR);
    /* Initialize Crypto driver */
    Crypto_Init(NULL_PTR);

    /* Format HSE key catalogs (RAM + NVM) */
    RetVal = Crypto_Exts_FormatKeyCatalogs();
    App_SetSuccessStatus((Std_ReturnType)E_OK == RetVal);

    /* ==========================================================================
    /*    Run the AES128 encrypt/decrypt example
    /* ==========================================================================
    App_Aes128EncryptDecryptExample();

    /* ==========================================================================
    /*    Run the CMAC generate/verify example
    /* ==========================================================================
    App_CmacGenerateVerifyExample();

    /* ==========================================================================
    /*    Run the HASH example
    /* ==========================================================================
    App_HashExample();

    /* ==========================================================================
    /*    Finish application execution, signaling the status
    /* ==========================================================================
    Exit_Example(App_GetSuccessStatus());
    return (0U);
}
```

HSE CRYPTO DEMO

Project Explorer

- Hse_Ip_AesEncAsyncIrq_S32K344: Debug_FLASH
  - Binaries
  - Includes
  - Project_Settings
  - RTD
  - board
  - generate
  - generate/include
  - generate/src
  - src
    - main.c
  - Debug_FLASH
  - include
  - Configuration.mex
  - description.txt

main.c | main.c

```c
523 int main(void)
524 {
525     hseStatus_t            HseStatus;
526     Hse_Ip_StatusType      HseIpStatus;
527     hseSrvResponse_t       HseResponse;
528     uint8                  u8Counter;
529     static volatile uint8  u8NumSuccessfulHseRequests = 0;
530     static volatile uint32 u32NumFailedOperations     = 0U;
531
532     /* ==========================================================================
533     /*    Initialize OsIf component. It will be used to measure timeouts while wai
534     /* ==========================================================================
535     OsIf_Init(NULL_PTR);
536
537     /* ==========================================================================
538     /*    Install and enable the needed interrupt handlers
539     /* ==========================================================================
540     /* Install ORed RX interrupt for MU-0 */
541     IntCtrl_Ip_InstallHandler(HSE_MU0_RX_IRQn, &Mu_Ip_Mu0_OredRx_Isr, NULL_PTR);
542     /* Enable ORed RX interrupt for MU-0 */
543     IntCtrl_Ip_EnableIrq(HSE_MU0_RX_IRQn);
544
545     /* ==========================================================================
546     /*    Check that HSE is initialized (along with RNG module) by reading the sta
547     /* ==========================================================================
548     HseStatus = Hse_Ip_GetHseStatus(MU0_INSTANCE_U8);
549     App_SetSuccessStatus(0U != (HseStatus & HSE_STATUS_INIT_OK));
550     App_SetSuccessStatus(0U != (HseStatus & HSE_STATUS_RNG_INIT_OK));
551
552     /* ==========================================================================
553     /*    Initialize Hse Ip layer for MU0 instance
554     /* ==========================================================================
555     HseIpStatus = Hse_Ip_Init(MU0_INSTANCE_U8, &HseIp_MuState);
556     App_SetSuccessStatus(HSE_IP_STATUS_SUCCESS == HseIpStatus);
557
```

HSE CRYPTO DEMO

Project Explorer

- Crypto_47478A_440_100: Debug_FLASH
  - Includes
  - Generate
  - Mcal_Plugins
  - Project_Settings
  - inc
  - src
    - HSE_Util.c
    - main.c
    - Task_AesEncDecDemo.c
    - Task_HashDemo.c
    - Task_MacGenVerDemo.c
  - Tresos_Project
  - readme.txt

main.c

```c
    }

    /* ==========================================================
    /*    Initialization
    /* ==========================================================
    /* Initialize OsIf component. It will be used to measure ti
    Std_ReturnType RetVal;
    (void) RetVal;

    /* Initialize Crypto driver */
    Crypto_Init(NULL_PTR);

#if 1
    /* if already format catalog ,erase NVM , new HSE-FW(0.0.12
    if (TRUE == HSE_CheckStatus(HSE_STATUS_INSTALL_OK) )
    {
        HSE_EraseNVM();
    }
#endif

    /* Format HSE key catalogs (RAM + NVM) */
    RetVal = Crypto_Exts_FormatKeyCatalogs();

    Task_AesEncDecDemo();
    Task_MacGenVerDemo();
    Task_HashDemo();

#if 1
    HSE_EraseNVM();
#endif

    return (0);
}
```

Crypto.zip

SECURE BOOT

HSE SECURE BOOT MODE

Basic Secure Boot

- ✓ Only one application core is enabled (specified in IVT)
- ✓ GMAC based Authentication used with a key derived from ADK/P.
- ✓ Can be used only if SMR/CR based Secure Boot is not used.

Advanced Secure Boot

- ✓ Enables single or multiple cores (defined in the Core Rest table)
- ✓ Supports Symmetric Authentication Scheme (AES-CMAC, GMAC, HMAC, XCBC-MAC etc.)
- ✓ Supports RSA and ECC Signature verification schemes.
- ✓ Supports encrypted images (AEAD-GCM, AES-CTR)

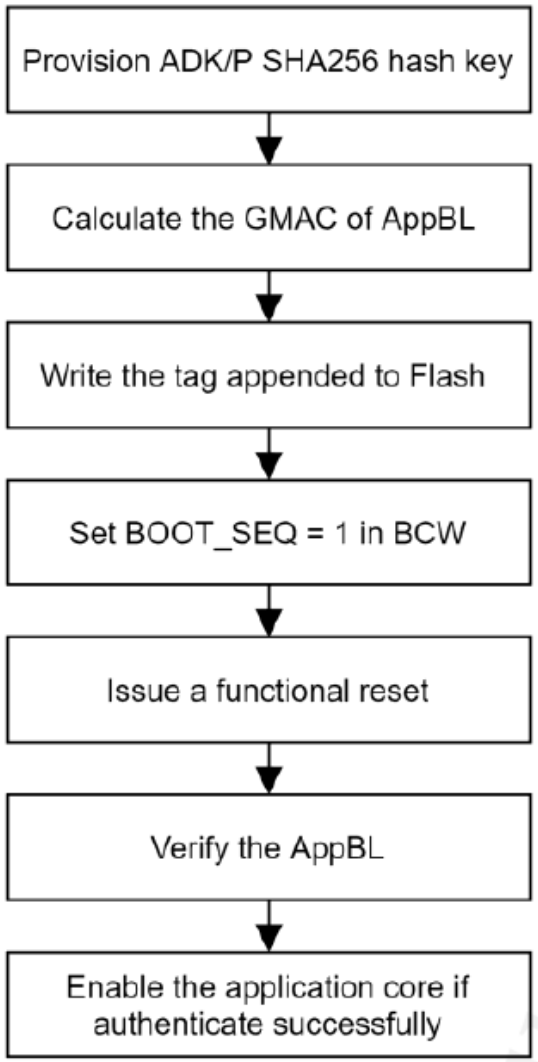
HSE SECURE BOOT MODE

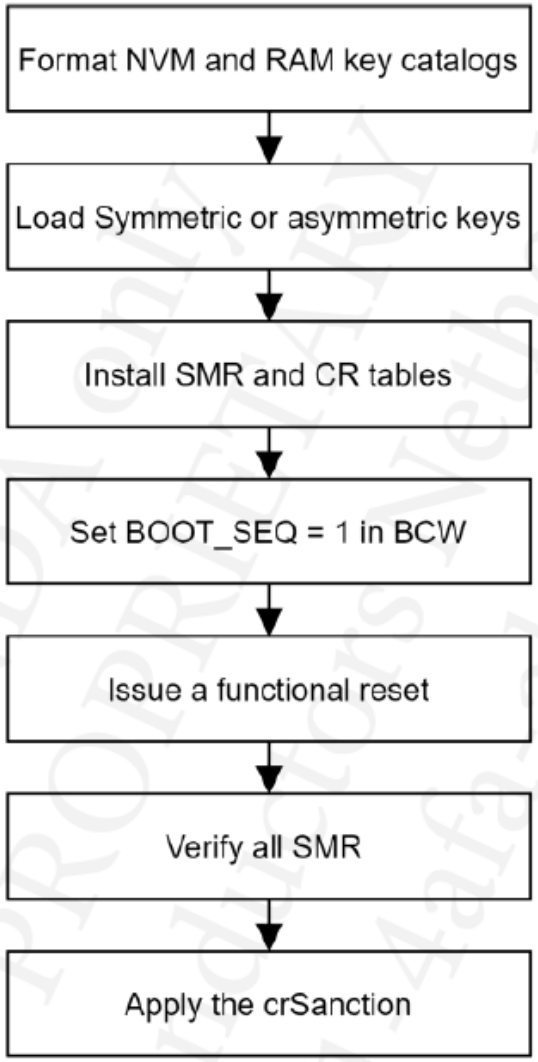
SHE Based Secure Boot

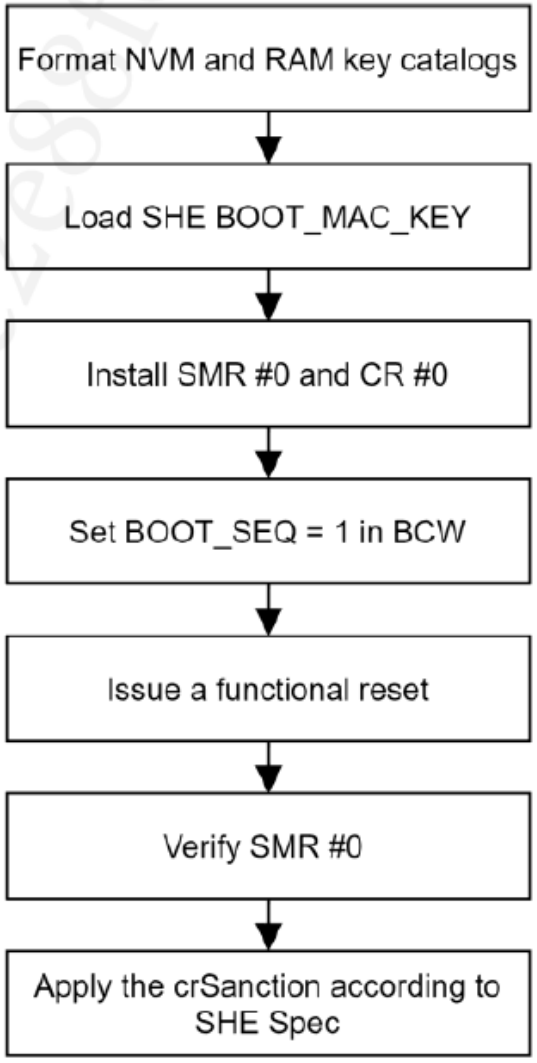
- ✓ Emulates SHE protocol based secure boot operation
- ✓ Can be enabled for any one core. SHE supports only CMAC based authentication scheme with user BOOT_MAC key.
- ✓ It is a variant of SMR Boot - the first entry into SMR (entry at index 0) can be used for SHE secure boot operation. HSE firmware identifies SHE secure boot by reading the Key handle in the SMR. If the key handle at entry index 0, is SHE BOOT_MAC key then HSE Firmware will initiate a SHE secure boot.

Mode	Key	Scheme	SMR use	Number of protect regions	Proof location
BSB	ADKP	GMAC	No	1	Application NVM
ASB	Sym or Asym key	MAC or Sign	Yes	Up to 8	Secure NVM
SHE（ASB）	BOOT_MAC_KEY	CMAC	Yes (only SMR #0)	1	Secure NVM

HSE SECURE BOOT MODE



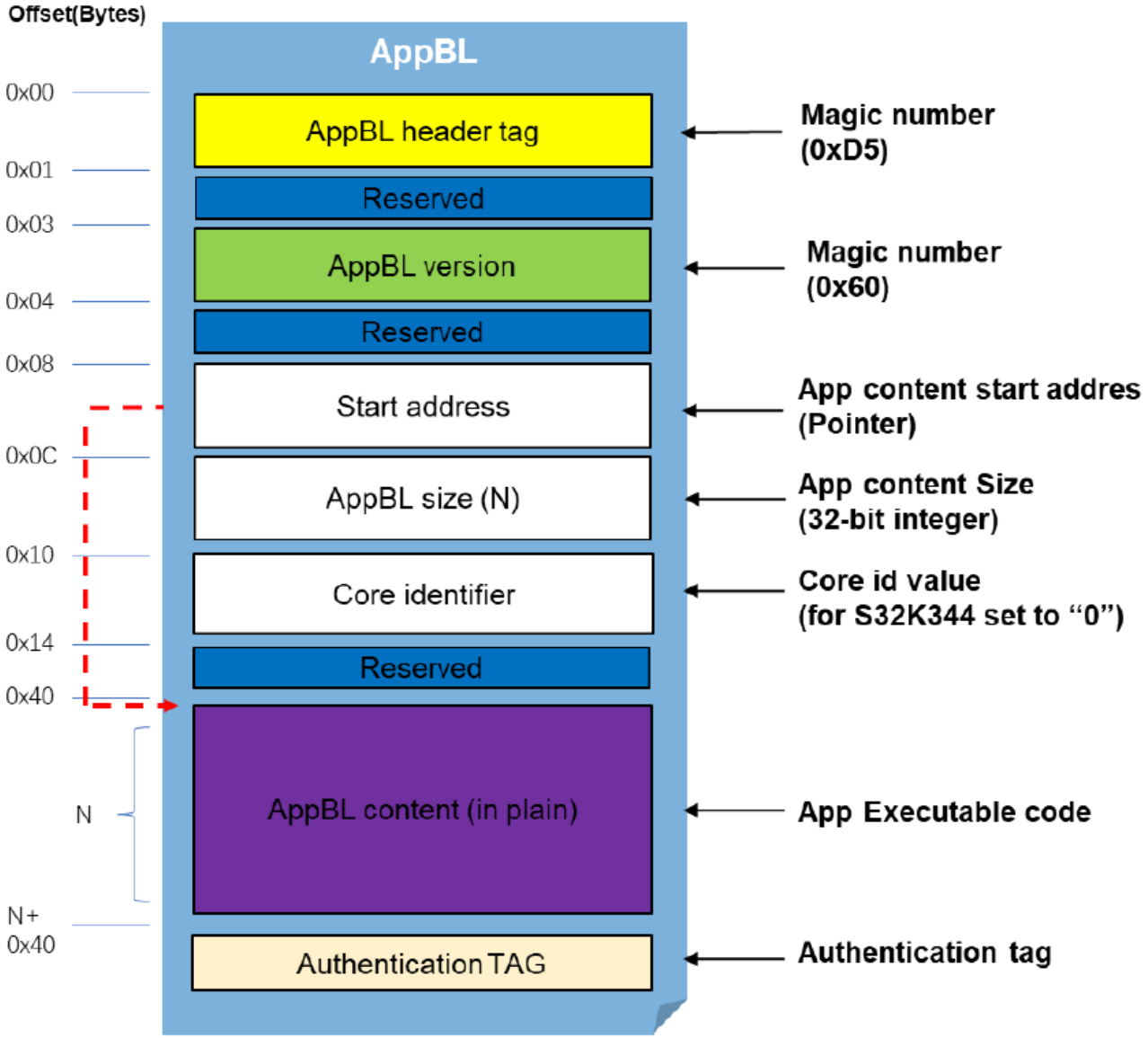




BASIC SECURE BOOT

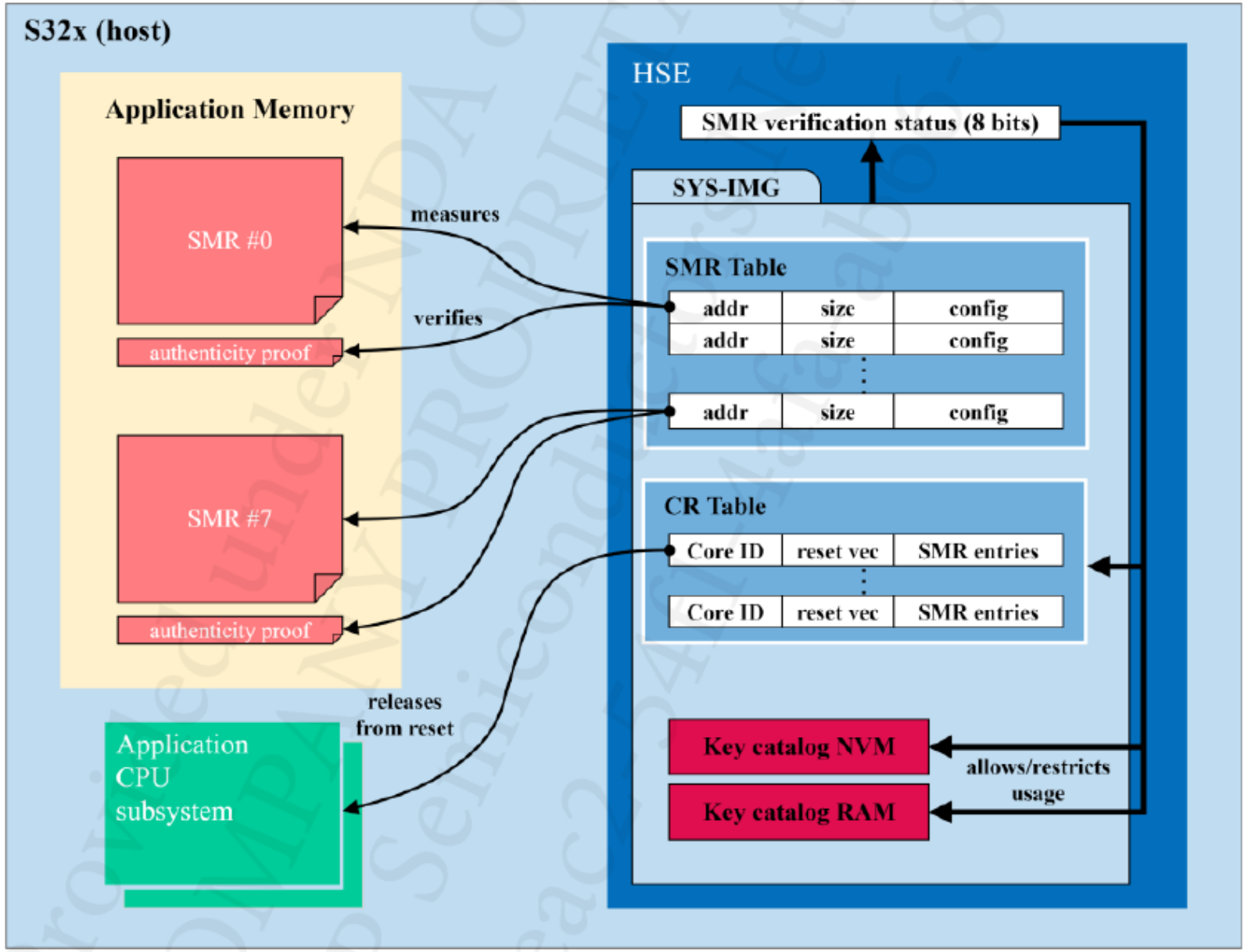
- Application Bootloader **either in non-secure way** or secure way by HSE.
- To support the Basic Secure Boot (BSB) the App Bootloader header includes some **extra configurations** such as Authentication Tag, Core ID
- The App Bootloader is signed by the Host using the “HSE Boot Data Sign” service. The generated tag has to be stored at the end of the header.

NOTE: APPBL will be ignored if SMR/CR installed



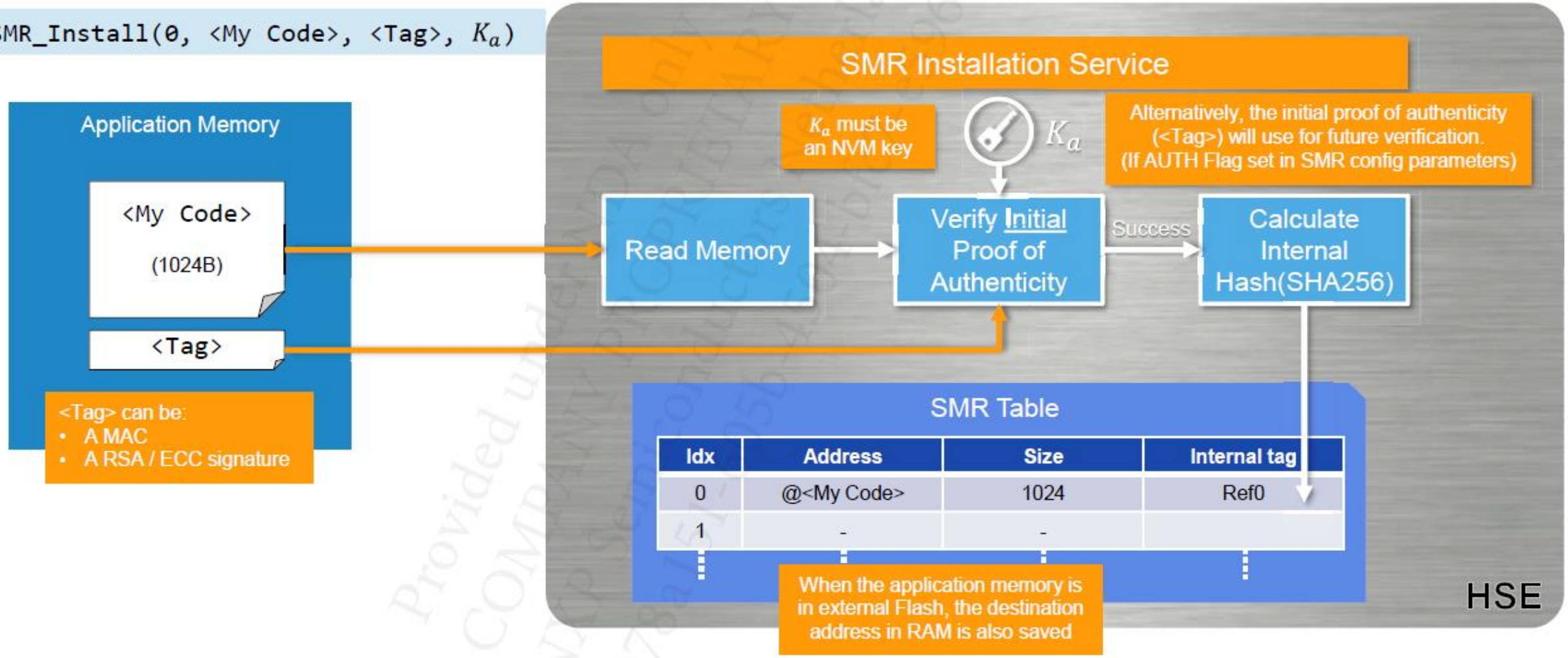
ADVANCED SECURE BOOT

A secure memory region (SMR) is defined by a start address and a size, associated to a proof of authenticity, either a MAC or an RSA/ECC signature, which authenticates the region's content.

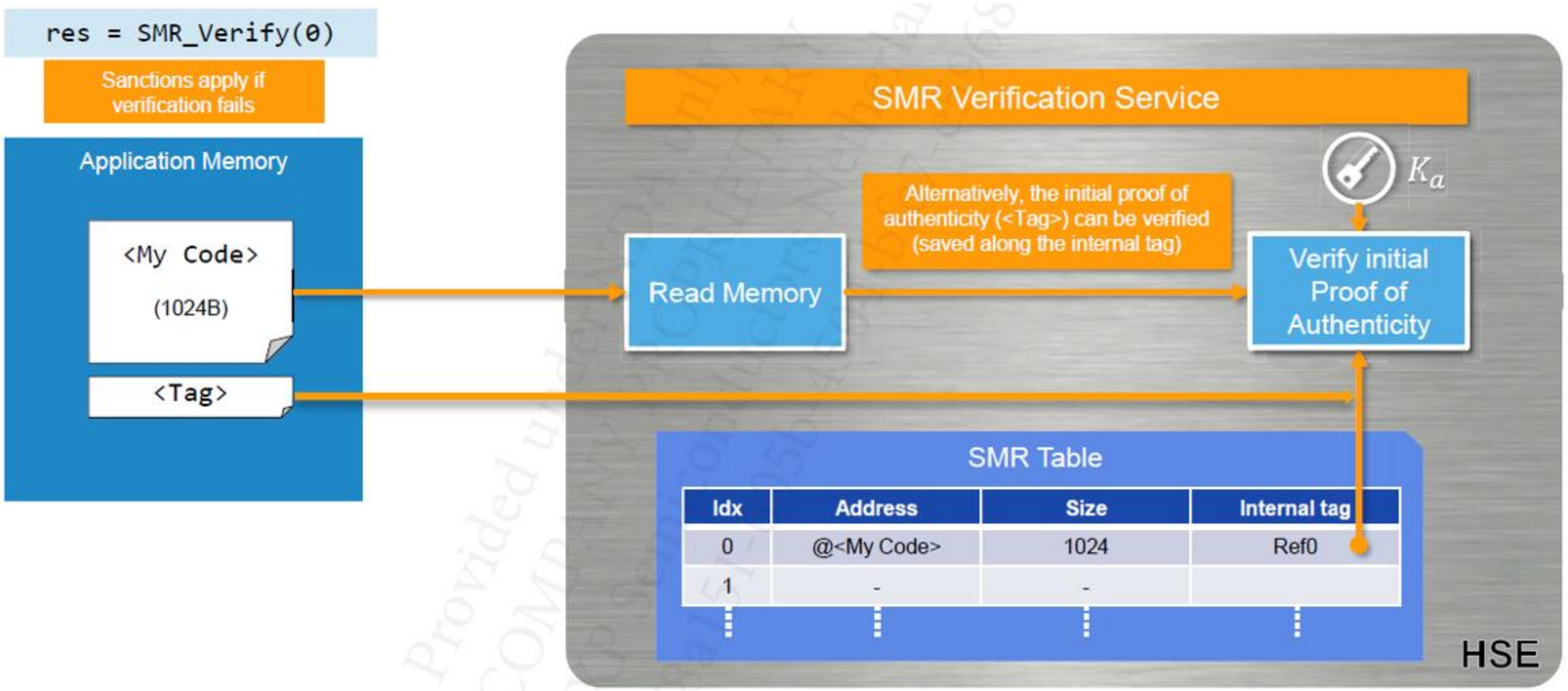


SMR INSTALL

```
res = SMR_Install(0, <My Code>, <Tag>, K_a)
```



SMR VERIFY


res = SMR_Verify(0)
Sanctions apply if verification fails
Application Memory
<My Code>
(1024B)
<Tag>
SMR Verification Service
Read Memory
Alternatively, the initial proof of authenticity (<Tag>) can be verified (saved along the internal tag)
K_a
Verify initial Proof of Authenticity
SMR Table
Idx
Address
Size
Internal tag
0
@<My Code>
1024
Ref0
1
-
-
HSE

SMR PARAMETERS

Attribute	Data field	Description
Source address	pSmrSrc	SMR Source Address (External or internal Flash)
Size	smrSize	SMR Size
Destination address	pSmrDest	SMR destination address (System RAM)
Initial authenticity proof	pInstAuthTag[]	Initial Auth. Tag address (External or internal Flash)
Authentication scheme	authScheme	Verification Scheme (MAC, RSA, ECC)
Authentication key	authKeyHandle	NVM Key
Decryption parameters	smrDecrypt	Reference to SMR decryption values
Verification period	checkPeriod	Define the verification sequence period
SMR configuration flags	configFlags	Configuration flags for memory interface and the authenticity proof
SMR Version	versionOffset	offset in SMR where the image version can be found

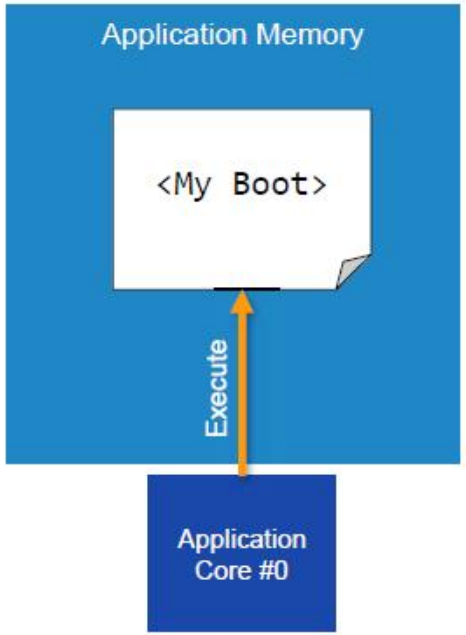
CR PARAMETERS

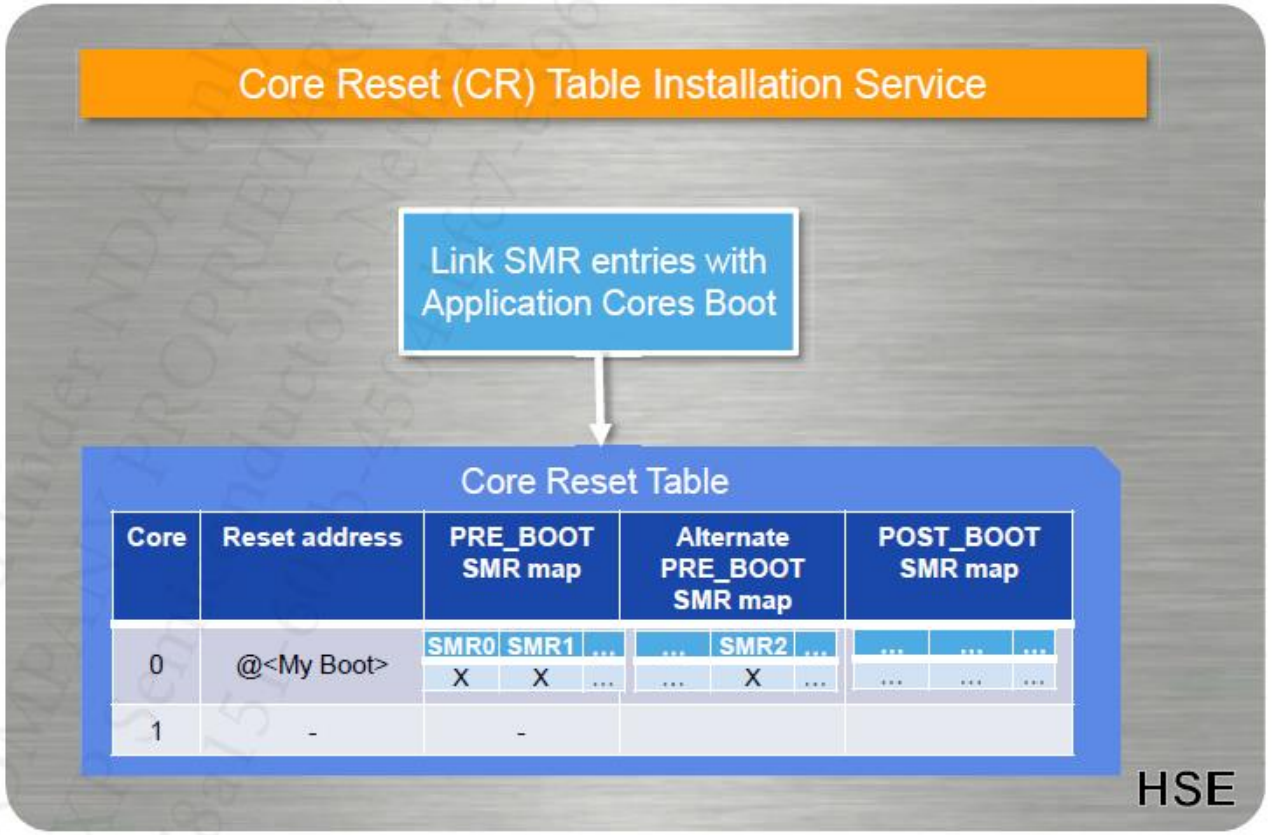
Attribute	Data field	Description
Core identifier	coreId	A unique number that identifies a CPU-driven subsystem
Pre-boot SMR verification map	preBootSmrMap	A set of flags that define which SMR, indexed from 0 to 31(bit #i for SMR #i)
Alternate Pre-boot SMR verification map	altPreBootSmrMap	A set of flags that define which SMR, indexed from 0 to 31 (bit #i for SMR #i)
Post-boot SMR verification map	postBootSmrMap	A set of flags that define which SMR, indexed from 0 to 31 (bit #i for SMR #i)
Reset address	pPassReset	A Value of the VTOR of associated application subsystem
Alternate reset address	pAltReset	Value of the VTOR of associated application subsystem if all the SMR defined in altSmrVerifMap pass the verification
Core boot option	startOption	Specifies whether the core is automatically started by the HSE at boot-time or if the CR entry is used for on-demand booting at run-time.
Sanctions on failed verification	crSanction	The sanction HSE applies for the CR entry if one of the associated SMR fails verification.

BOOT SEQUENCE EXAMPLE

"I want Core #0 to boot only if HSE has successfully verified SMR0 & SMR1"
"If SMR0 & SMR1 does not verify, load the backup from SMR2 and boot the Core #0"

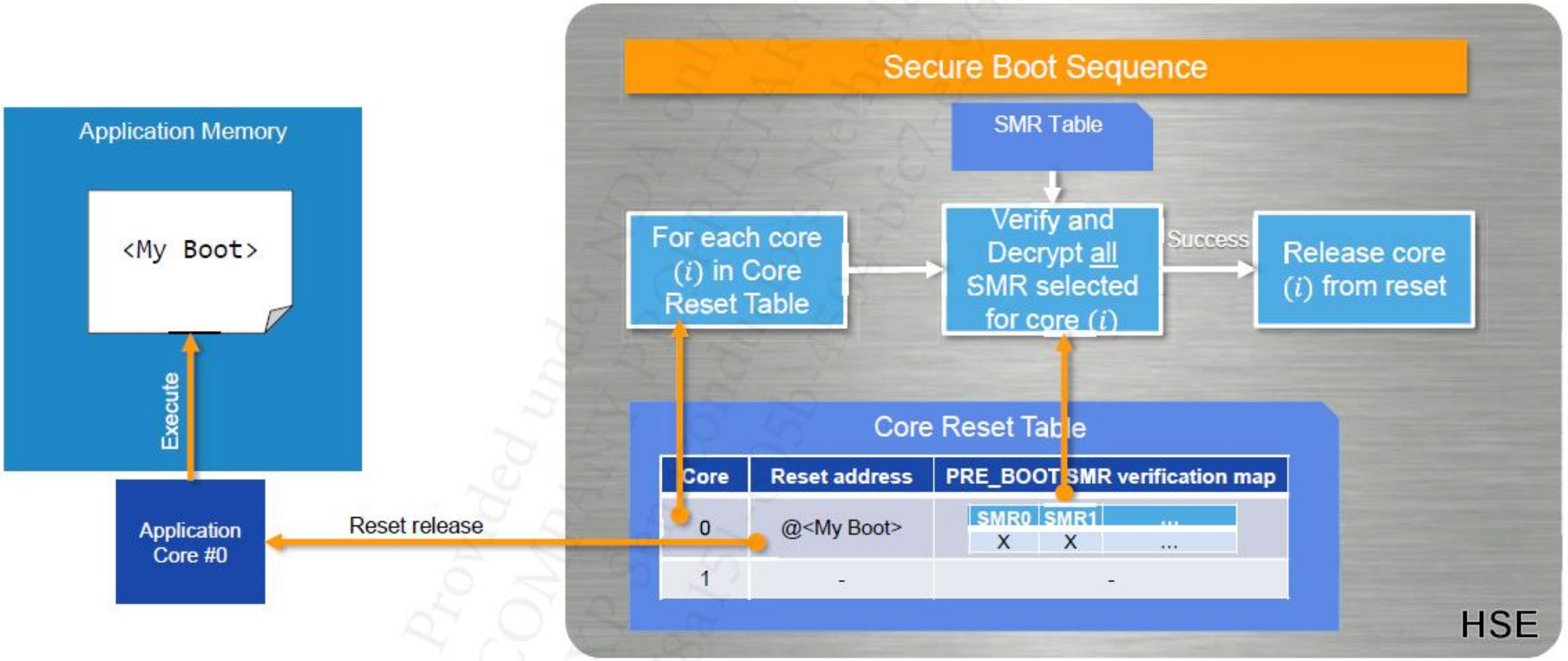
```
res = CR_Install(0, <My Boot>,
PRE_BOOT=SMR0|SMR1, ALT_PRE_BOOT=SMR2)
```



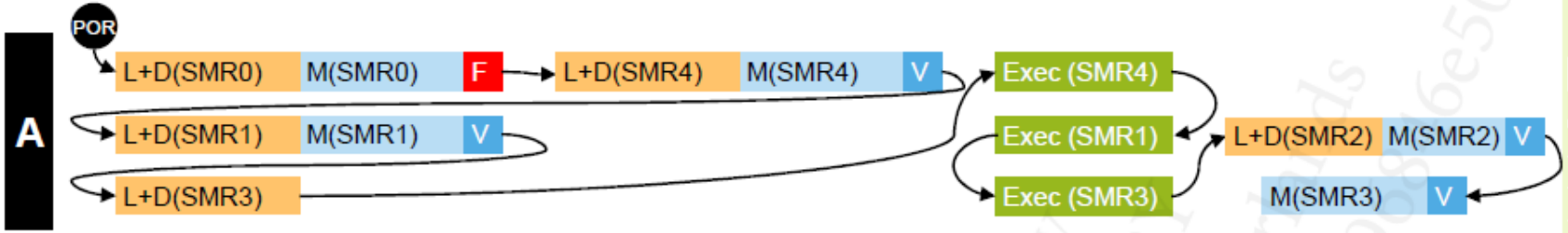


BOOT SEQUENCE EXAMPLE

"Boot securely"



RESET STRATEGIES

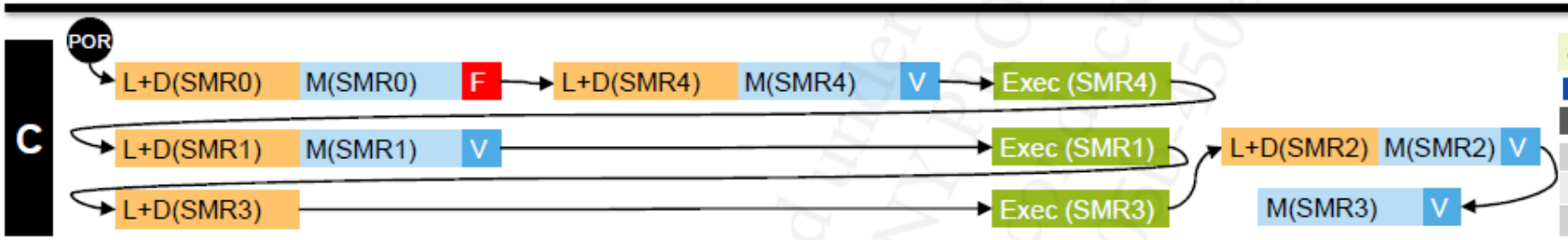


- ALL_AT_ONCE (Mix entries)

SMR map	PRE_BOOT					PRE_BOOT ALT				POST_BOOT				
Core ID	0	1	2	3	...	...	4	5	6	...	0	1	2	3	...
0	X						X								
1		X						X					X		
2														X	

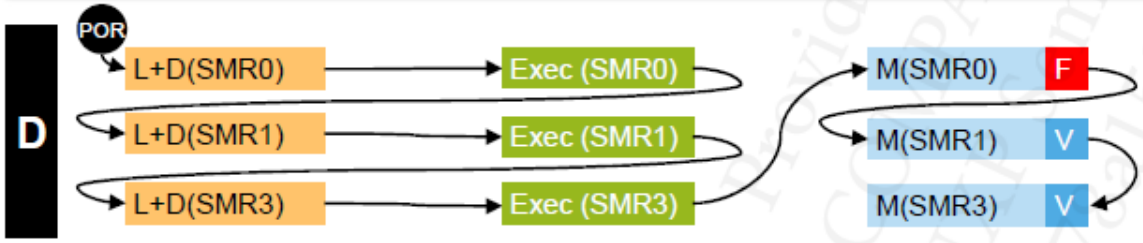
- ALL_AT_ONCE (All SMR in "post-boot")

SMR map	PRE_BOOT					PRE_BOOT ALT				POST_BOOT				
Core ID	0	1	2	3	...	...	4	5	6	...	0	1	2	3	...
0											X				
1												X			
2														X	



- ONE_BY_ONE (Mix entries)

SMR map	PRE_BOOT					PRE_BOOT ALT				POST_BOOT				
Core ID	0	1	2	3	...	...	4	5	6	...	0	1	2	3	...
0	X						X								
1		X						X					X		
2														X	



- ONE_BY_ONE (All SMR in "post-boot")

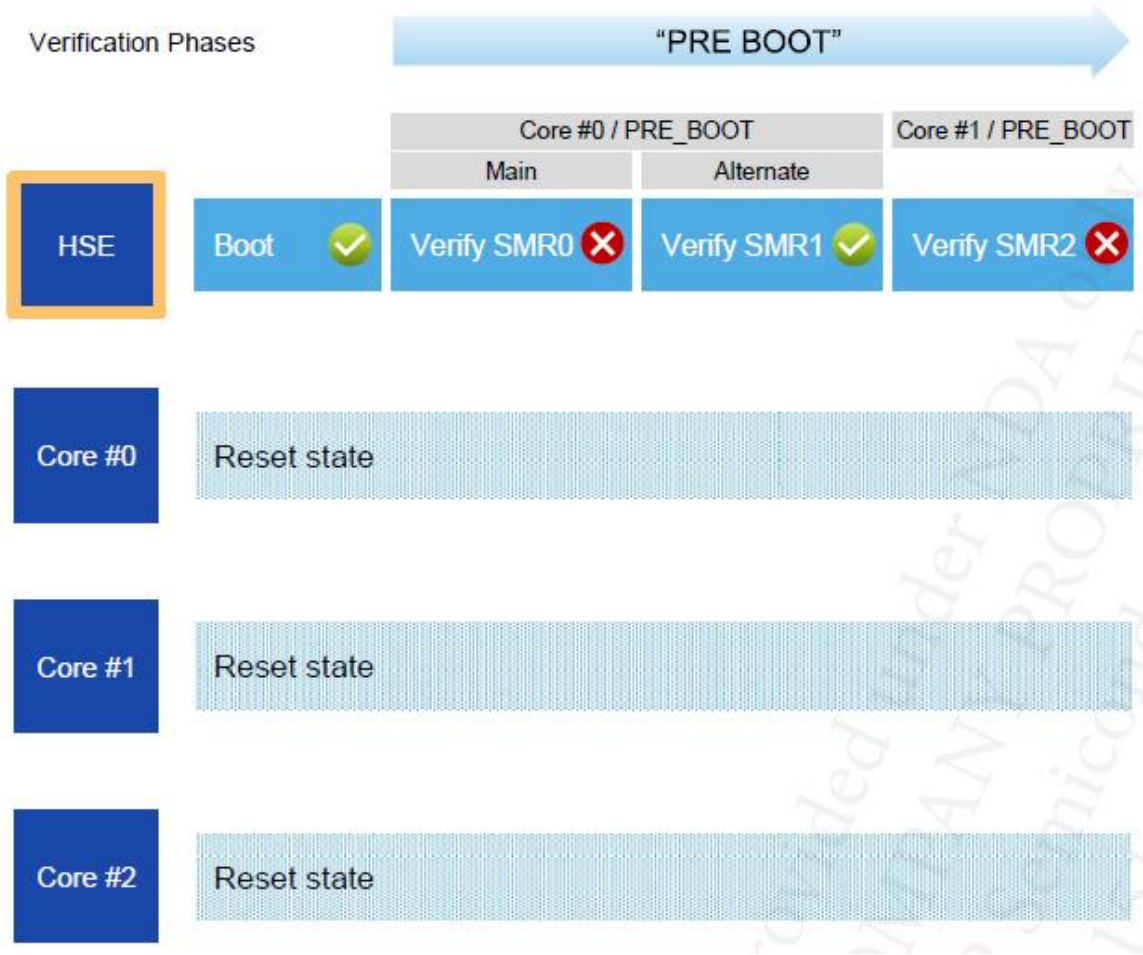
SMR map	PRE_BOOT					PRE_BOOT ALT				POST_BOOT				
Core ID	0	1	2	3	...	...	4	5	6	...	0	1	2	3	...
0											X				
1												X			
2														X	

SMR / CR VERIFICATION & SANCTIONS

SMR verification moment	PRE_BOOT	POST_BOOT	RUN_TIME	ON_DEMAND
Verification trigger	One-time automatic verification at reset		Automatic verification at regular time interval	Verification on demand from host
Target use case	Strict secure boot	Parallel secure boot	Run-time integrity check - periodically	Run-time integrity check – on demand
Application core(s) status	Stop (Reset)	Running (when released in PRE_BOOT)		
Sanctions applying when SMR verification fails (one selectable option) — Defined in CR table	Keep associated core in reset			
	Release core to alternate reset addr. → In this case, the reset address can be defined to be part of a SMR			
	Reset the device			
	All keys disabled			
Sanction always applying when SMR verification fails — Defined in key attributes	Associated keys not usable			

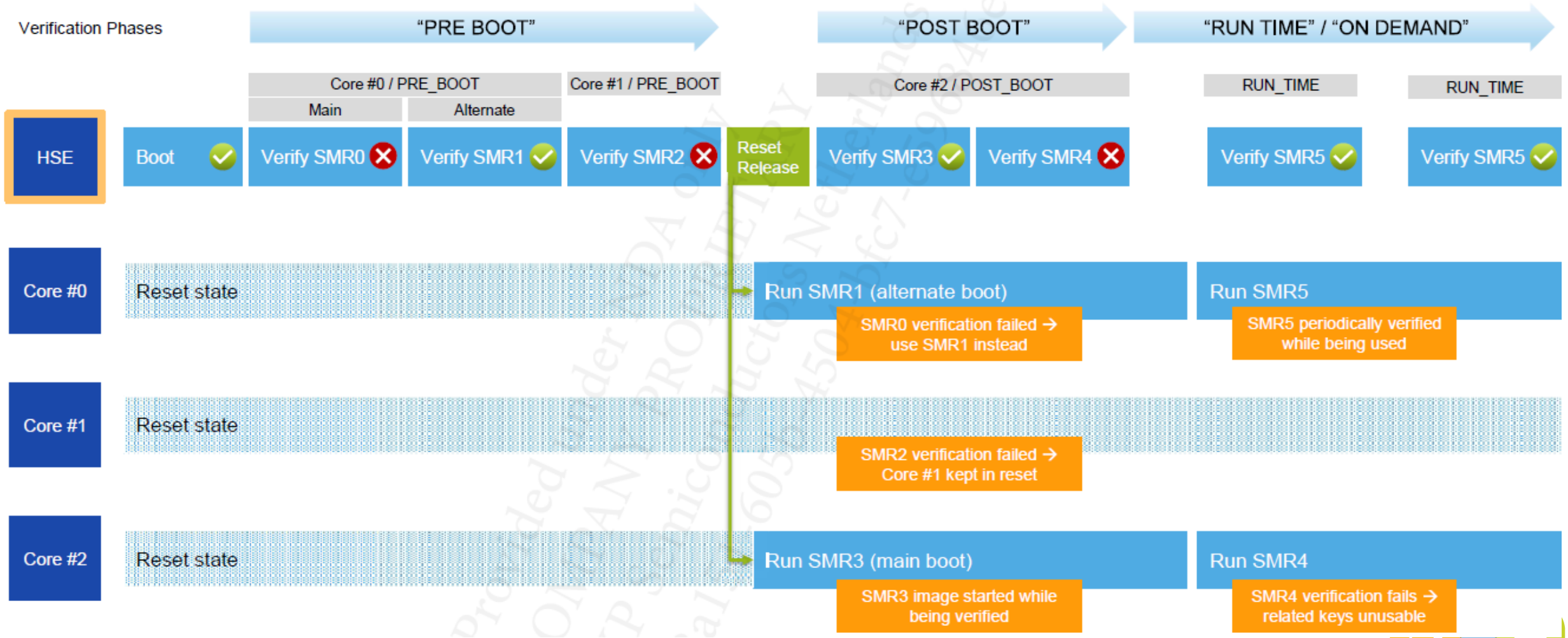
BOOT SEQUENCE EXAMPLE

SMR map	PRE_BOOT					PRE_BOOT ALT				POST_BOOT					RUN TIME	
Core ID	0	1	2	3	...	...	0	1	2	...	...	...	3	4	...	5	...
0	X							X									
1			X														
2													X	X			



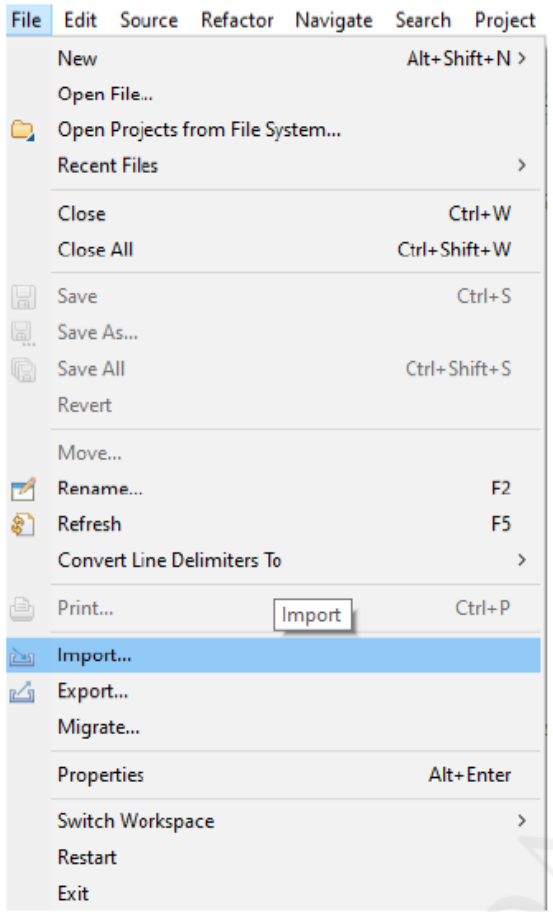
BOOT SEQUENCE EXAMPLE

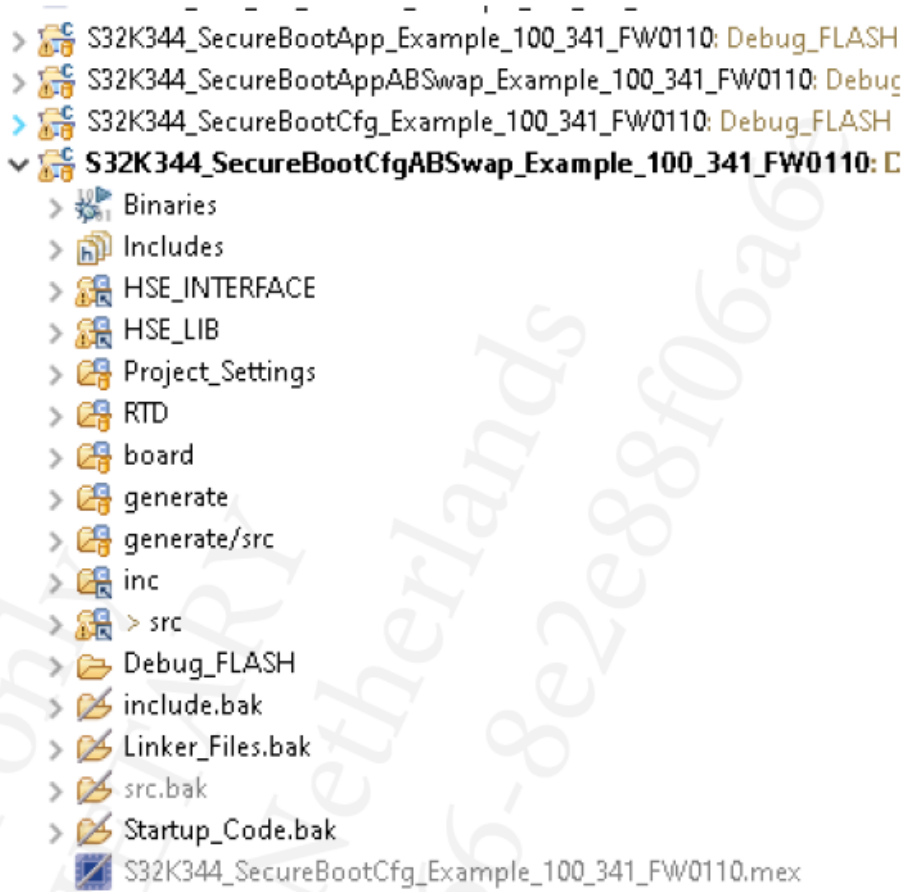
SMR map	PRE_BOOT						PRE_BOOT ALT						POST_BOOT				RUN TIME
Core ID	0	1	2	3	...	...	0	1	2	...	...	...	3	4	...	5	...
0	X							X									
1			X														
2													X	X			

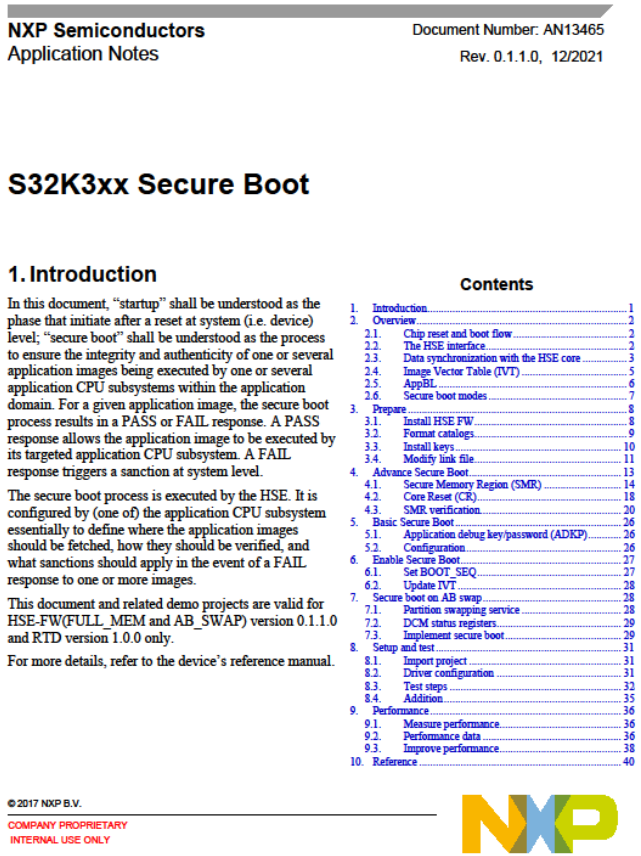


SECURE BOOT DEMO

The Demo includes secure boot cfg and app project.







NXP Semiconductors
Application Notes

Document Number: AN13465
Rev. 0.1.1.0, 12/2021

S32K3xx Secure Boot

1. Introduction

In this document, "startup" shall be understood as the phase that initiate after a reset at system (i.e. device) level; "secure boot" shall be understood as the process to ensure the integrity and authenticity of one or several application images being executed by one or several application CPU subsystems within the application domain. For a given application image, the secure boot process results in a PASS or FAIL response. A PASS response allows the application image to be executed by its targeted application CPU subsystem. A FAIL response triggers a sanction at system level.

The secure boot process is executed by the HSE. It is configured by (one of) the application CPU subsystem essentially to define where the application images should be fetched, how they should be verified, and what sanctions should apply in the event of a FAIL response to one or more images.

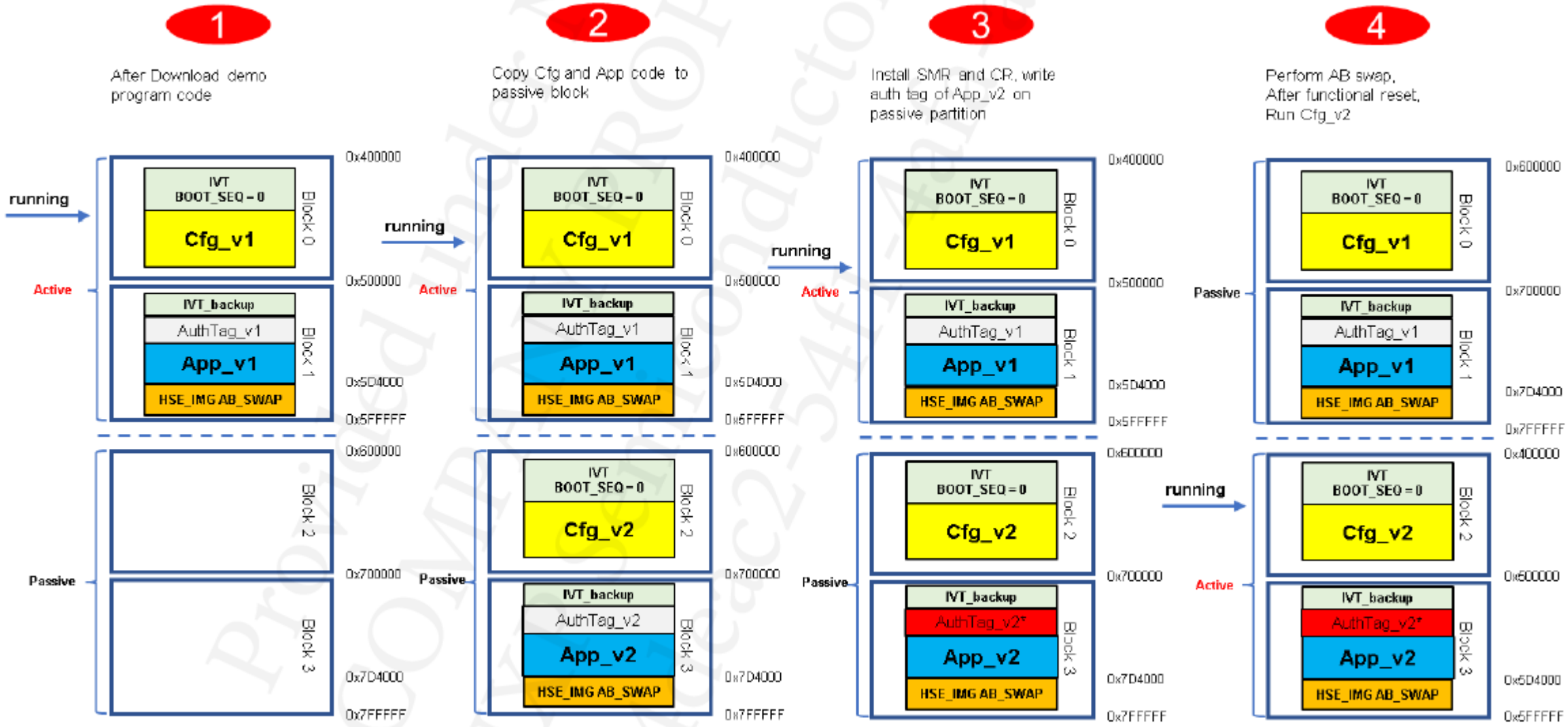
This document and related demo projects are valid for HSE-FW(FULL_MEM and AB_SWAP) version 0.1.1.0 and RTD version 1.0.0 only.

For more details, refer to the device's reference manual.

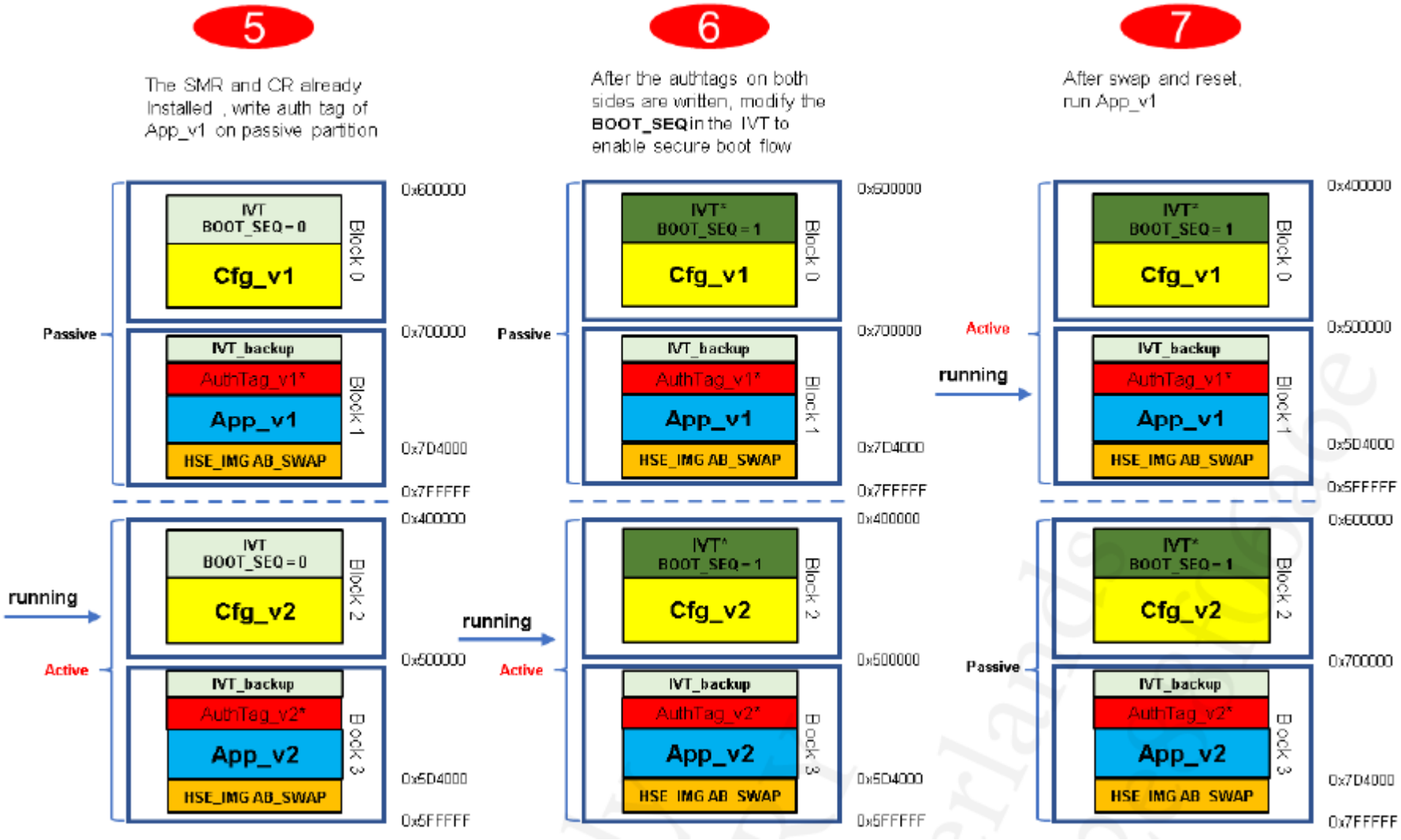
Contents

Secure Boot Demo and AN.zip

SECURE BOOT DEMO – A/B SWAP



SECURE BOOT DEMO – A/B SWAP



LIFE CYCLE & HOST DEBUG ACCESS

LIFE CYCLE INTRODUCTION

The Life Cycle (LC) is an important one-way internal device state closely related to ECU manufacturing, vehicle integration and failure analysis, which restricts access to certain HSE functionalities and debugging options.

LC state	Description
CUST_DEL	Device (i.e., NXP's IC) delivered to system integrator (i.e., NXP's customer) for ECU manufacturing and initial configuration.
OEM_PROD	ECU (device) delivered to the OEM for vehicle integration and final configuration.
IN_FIELD	ECU integrated in the vehicle and operating, this is the state of normal device use (and most secure state).
PRE_FA	Provide capabilities with Failure Analysis.
FA	ECU (device) failure, this is the state for functional testing of the IC.

The below table lists the host debugging capabilities that are available depending on the LC state.

LC state	Host debugging capabilities
CUST_DEL	Host debug open (unrestricted)
OEM_PROD	Host debug protected (with ADKP) or permanently disabled (see DEBUG_DISABLE)
IN_FIELD	Host debug protected (with ADKP) or permanently disabled (see DEBUG_DISABLE)
PRE_FA	Host debug protected (with ADKP) or permanently disabled (see DEBUG_DISABLE)
FA	Host debug open

LIFE CYCLE INTRODUCTION

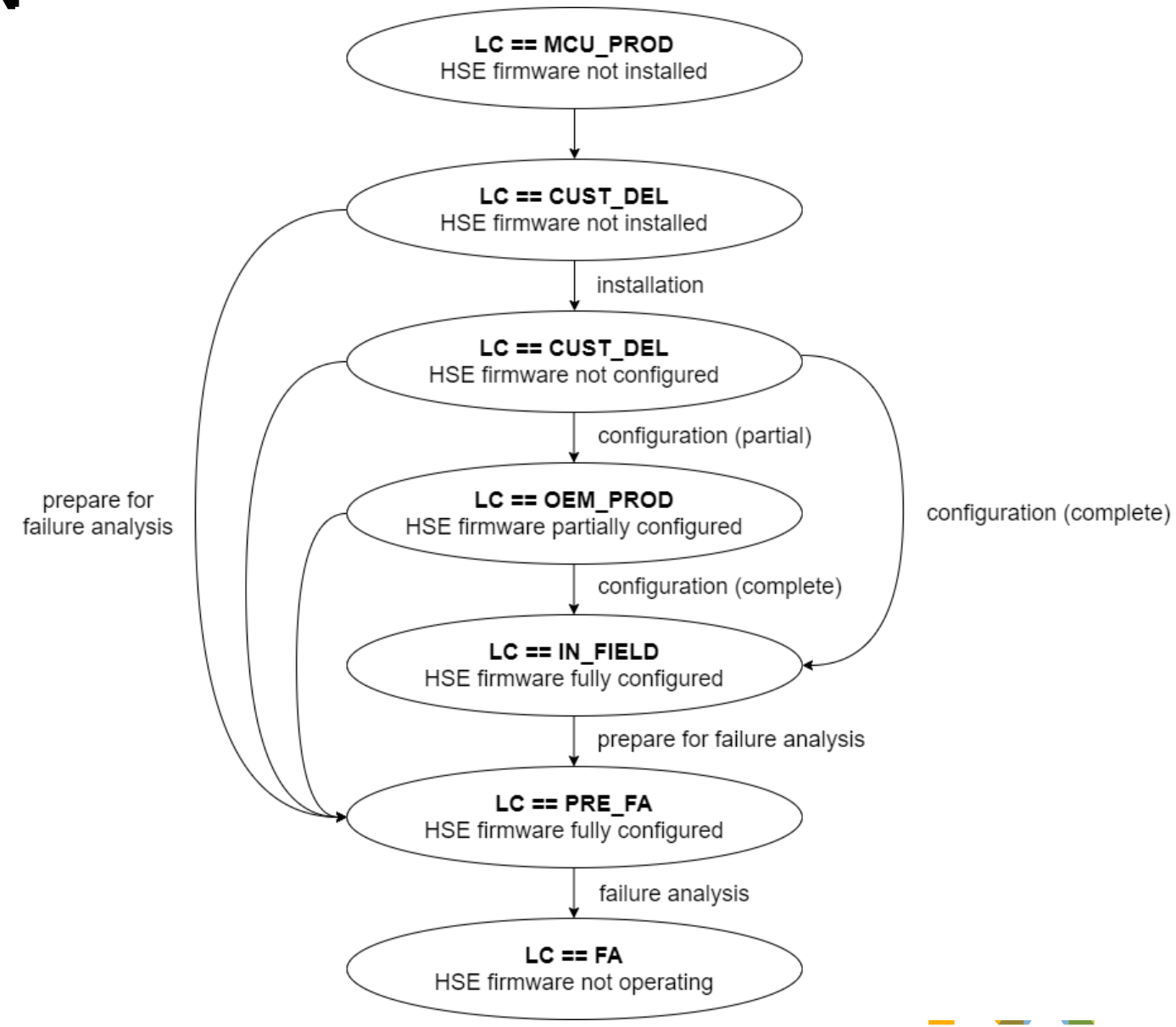
The LC states link with the HSE FW installation and configuration phases.

The LC can be advanced by two methods:

- The LCW in the IVT, changed by SBAF during start-up (reset)
- The HSE system attribute management service, changed by HSE during run-time

Pay special attention to the several points:

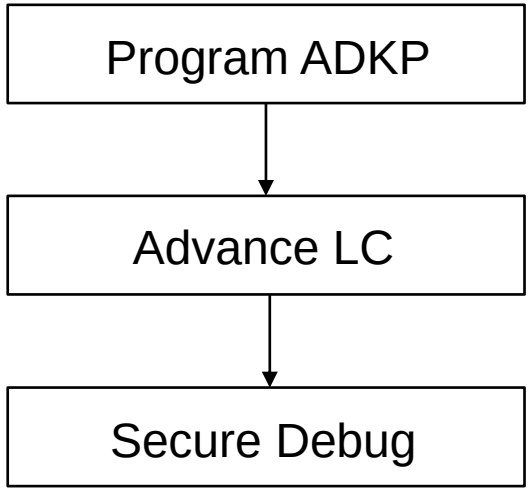
- Revert to a previous LC state is never possible
- The LC transition to PRE_FA|FA is only by NXP
- The LC transition to OEM_PROD|IN_FIELD is only possible when ADKP provisioned
- The LC transition is not possible through LCW in IVT if the user has installed the HSE FW



LIFE CYCLE INTRODUCTION

Some important One Time Programmed (OTP) fields (mainly for JTAG debug protection feature) in UTEST Flash area which were used in this project are shown in the right table.

To realize the JTAG debug protection feature, the users must strictly follow the below work sequence.



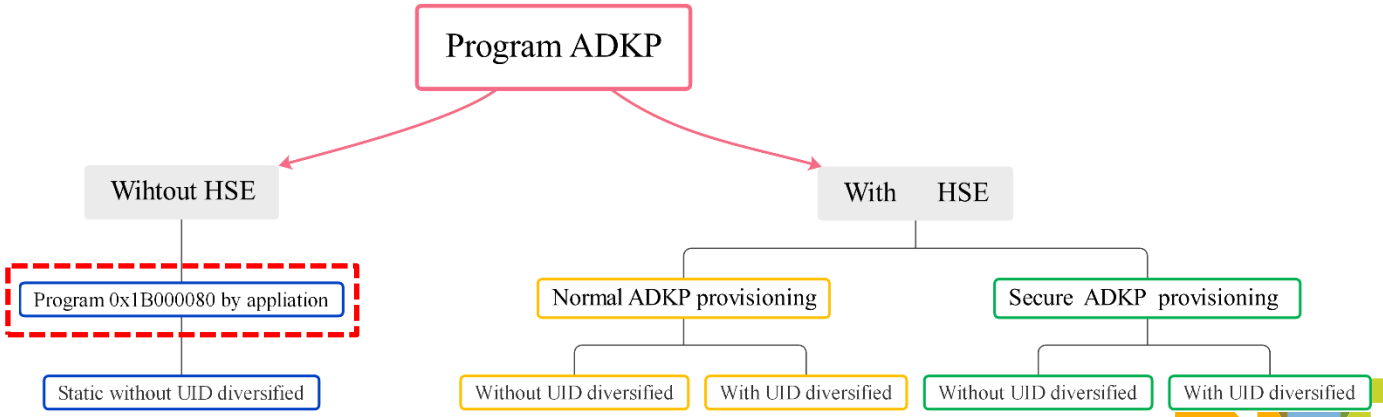
Start (hex)	End (hex)	Description	Details
1B00_0000	1B00_0007	HSE Firmware Feature Usage Flag	This flag indicates whether application intend to use HSE firmware on the device.
1B00_0040	1B00_0047	Unique Unique Identifier (UID)	The UID is provisioned by NXP in application NVM to identify each device from any other.
1B00_0080	1B00_008F	Debug password (CUST_DB_PSWD_A)	When HSE Firmware Feature Flag is disabled, this location stores the password for Password secure debug authentication mode.
1B00_0200	1B00_0207	Debug Auth Flag	This flag indicates whether changing the secure debug authentication mode from Password to Challenge Response (CR).
1B00_0208	1B00_020F	IVT_XRDC_GMAC Flag	This flag indicates whether enabling the IVT_AUTH feature.
1B00_0220	1B00_022F	Lifecycle slot 1: CUST_DEL	DCM determines the LC of the chip by reading these LC slots from the UTEST Flash Memory during reset phase.
1B00_0230	1B00_023F	Lifecycle slot 2: OEM_PROD	DCM determines the LC of the chip by reading these LC slots from the UTEST Flash Memory during reset phase.
1B00_0240	1B00_024F	Lifecycle slot 3: IN_FIELD	DCM determines the LC of the chip by reading these LC slots from the UTEST Flash Memory during reset phase.
1B00_0250	1B00_025F	Lifecycle slot 4: PRE_FA	DCM determines the LC of the chip by reading these LC slots from the UTEST Flash Memory during reset phase.
1B00_0260	1B00_026F	Lifecycle slot 5: FA	DCM determines the LC of the chip by reading these LC slots from the UTEST Flash Memory during reset phase.
1B00_0370	1B00_037F	APP_DBG_PASSWORD	When HSE Firmware Feature Flag is enabled, this location stores the ADKP for Password/CR secure debug authentication mode.
	R/W for any Master in all LCs		
	R/W blocked for any MASTER except HSE (read only) for LC >CUST_DEL		
	Write blocked for any Master except HSE for LC > MCU_PROD		

PROGRAM ADKP

The 128-bit Application Debug Key / Password (ADKP) is a very important OTP HSE system attribute, which closely related to such as secure debug, LC advancement and IVT authentication features.

HSE system attributes	Size	Description
AUTH_MODE	8 bits	- When 0(default): static authentication (password) - When 1: dynamic authentication (challenge / response)
ADKP	128 bits	- If AUTH_MODE equals 0, ADKP is a password - If AUTH_MODE equals 1, ADKP is a cryptographic key
ADKP_MASTER	8 bits	- When 0 (default): the input value is ADKP and is written “as is” in secure NVM - When 1: the input value is considered as a master debug key and is diversified with the device’s UID before being written in secure NVM

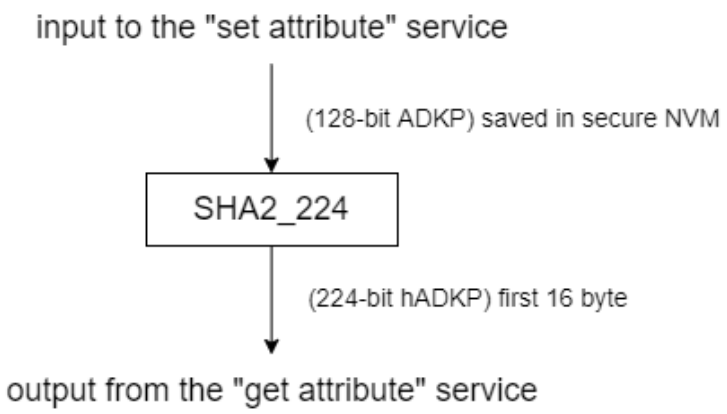
```
/*************************************************************************
 * Function:     ProgramAdkp
 * Description: Function for programming ADK/P request without HSE firmware.
 *              It is a write-once operation.
 *************************************************************************/
hseSrvResponse_t ProgramAdkp(void)
{
    Fls_CheckStatusType write_status = FLS_JOB_FAILED;

    write_status = HostFlash_Program(
                   (uint32_t )UTEST_ADKP_WITHOUT_HSE_ADDRESS,
                   (uint8_t* )applicationDebugKeyPassword,
                   (uint32_t )16U
                   );
    /* check if flash write was successful else will be stuck */
    ASSERT( write_status == FLS_JOB_OK );
    return HSE_SRV_RSP_OK;
}
```



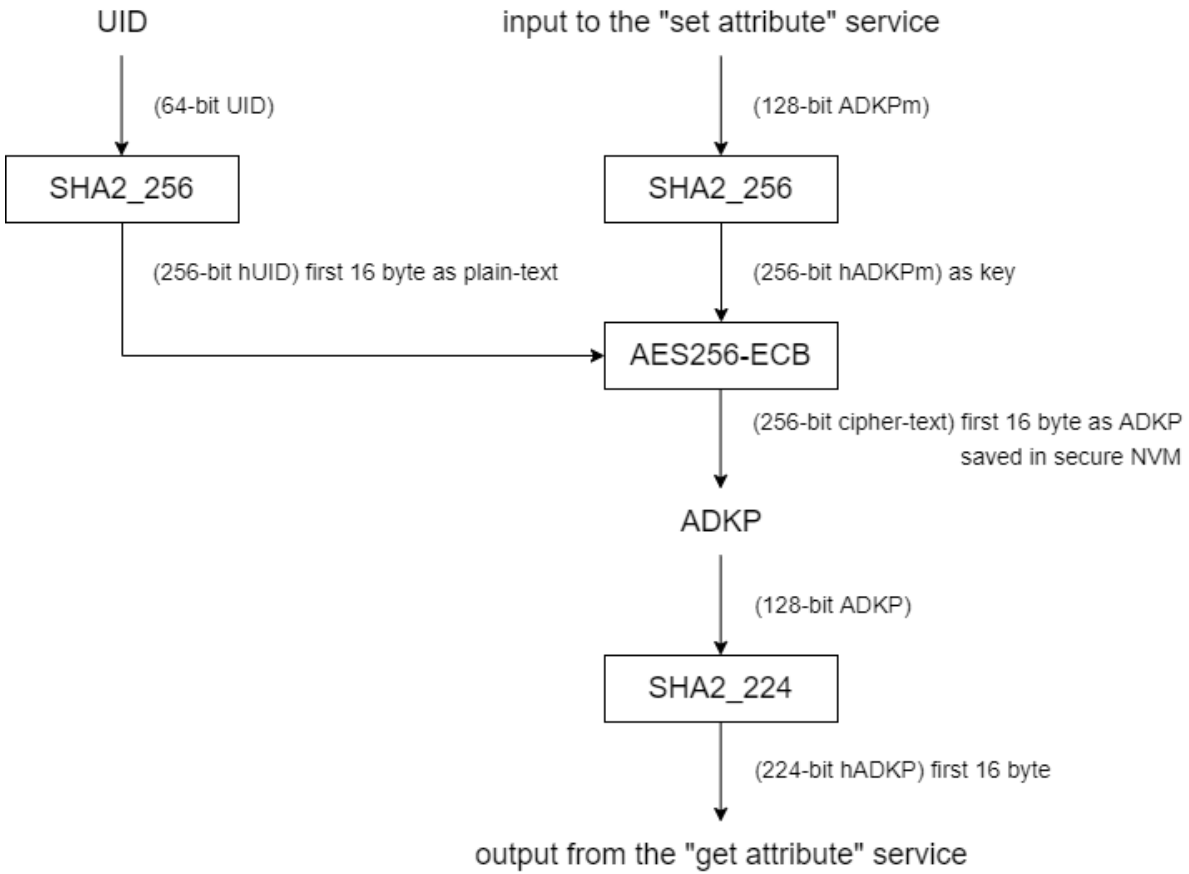
PROGRAM ADKP – NORMAL ADKP PROVISIONING

This is the commonly used ADKP provisioning process (only allowed in CUST_DEL LC with SU rights), which requires the user to provide the pointer to 16 bytes of plain ADKP stored in user space.

- This allows to provision a device-dependent debug key (or password) and use the ADKP as a master debug key
- The device-dependent key can be calculated based on the UID and the knowledge of the master debug key is never shared



Without UID diversified



With UID diversified

SECURE DEBUG – AUTHENTICATION MODE

The host debug is either open or protected (closing the debugger access through the JTAG interface, until the SBAF authenticates the debugger successfully) depending on the LC state.

Authentication mode	Description
Static (Password)	The ADKP is a password which is provided in plain form by the debugger.
Dynamic (Challenge Response)	The ADKP is a key which is used by the debugger to calculate a response to a random challenge.

There are two available secure debug authentication modes, which can be configured by the host via the HSE "set attribute" service (HSE_DEBUG_AUTH_MODE_ATTR_ID), only allowed in CUST_DEL LC stage.

The debugger runs the authentication process through the JTAG interface via two registers:

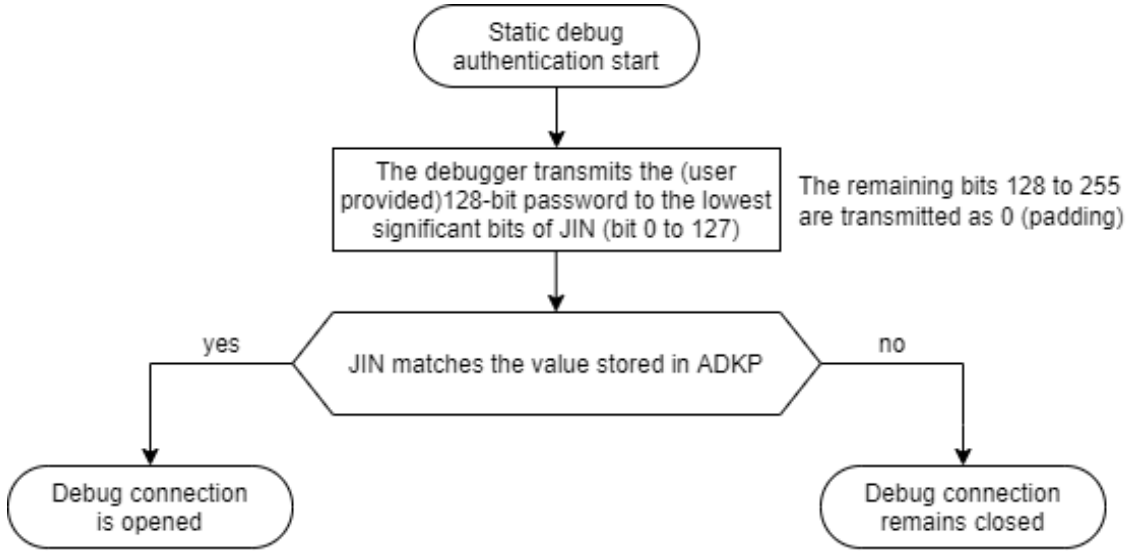
- JIN is a 256-bit input data register (debugger → device)
- JOUT is a 256-bit output data register (device → debugger)

Static or dynamic authentication	Use UID as a diversification parameter
Static	Without UID diversified
Static	With UID diversified
Dynamic	Without UID diversified
Dynamic	With UID diversified

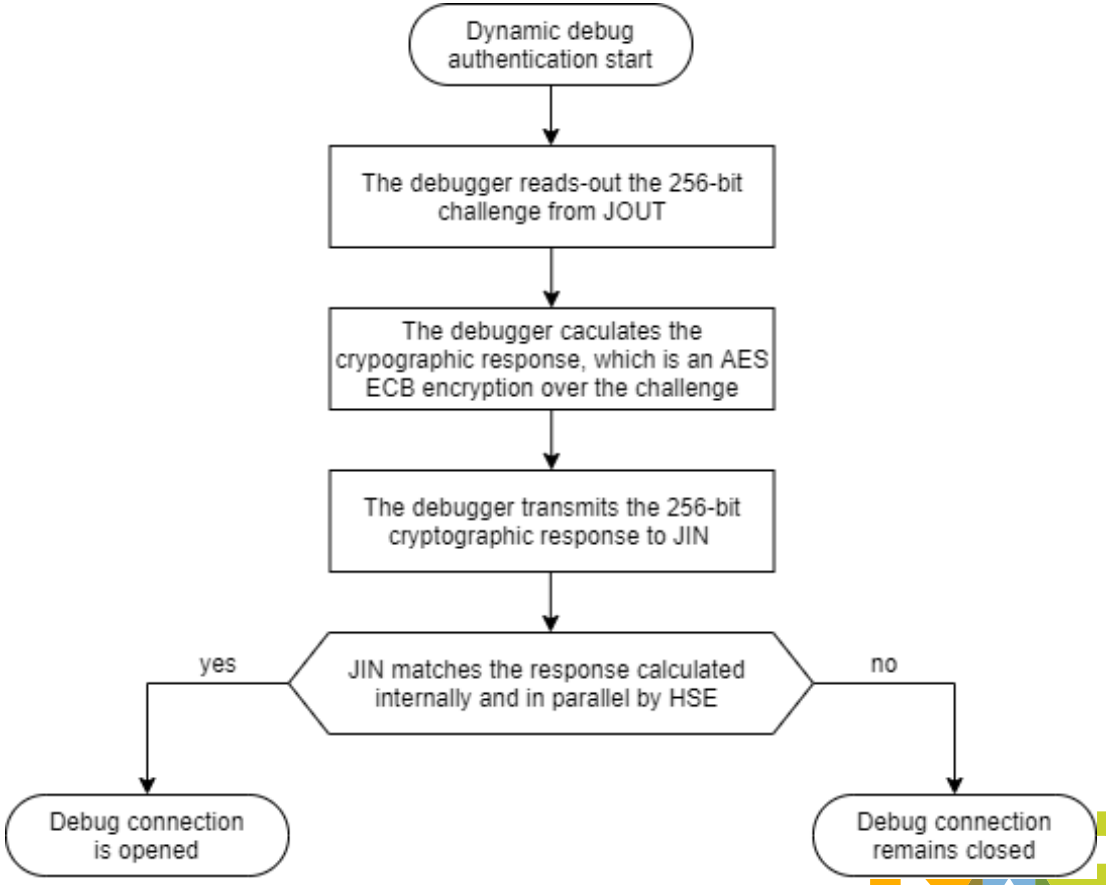
When the debugger authentication fails, the debugger must reset the device before trying to connect again and authenticate itself.

SECURE DEBUG – AUTHENTICATION MODE

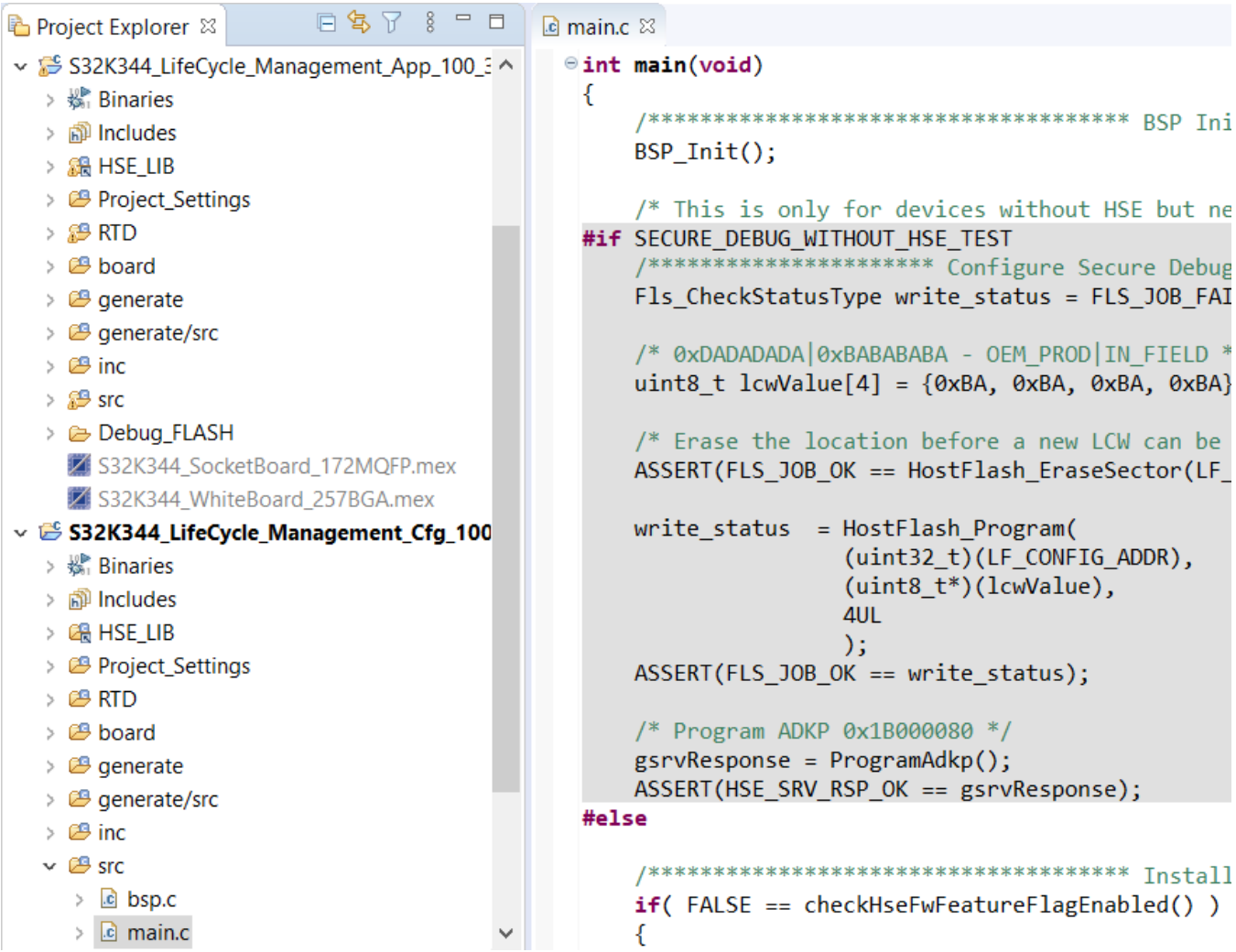
- Static
  - The debugger is expected to provide the same preconfigured password after each reset for debug unlocking.

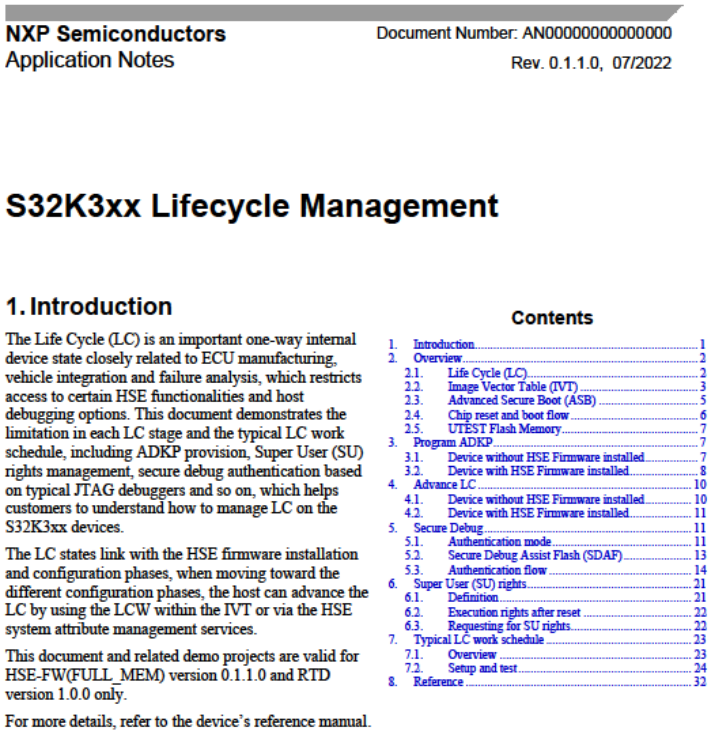


- Dynamic
  - The debugger is expected to write a different response after each reset.



SECURE DEBUG DEMO AND AN





NXP Semiconductors
Application Notes

Document Number: AN00000000000000
Rev. 0.1.1.0, 07/2022

S32K3xx Lifecycle Management

1. Introduction

The Life Cycle (LC) is an important one-way internal device state closely related to ECU manufacturing, vehicle integration and failure analysis, which restricts access to certain HSE functionalities and host debugging options. This document demonstrates the limitation in each LC stage and the typical LC work schedule, including ADKP provision, Super User (SU) rights management, secure debug authentication based on typical JTAG debuggers and so on, which helps customers to understand how to manage LC on the S32K3xx devices.

The LC states link with the HSE firmware installation and configuration phases, when moving toward the different configuration phases, the host can advance the LC by using the LCW within the IVT or via the HSE system attribute management services.

This document and related demo projects are valid for HSE-FW(FULL_MEM) version 0.1.1.0 and RTD version 1.0.0 only.

For more details, refer to the device's reference manual.

Contents

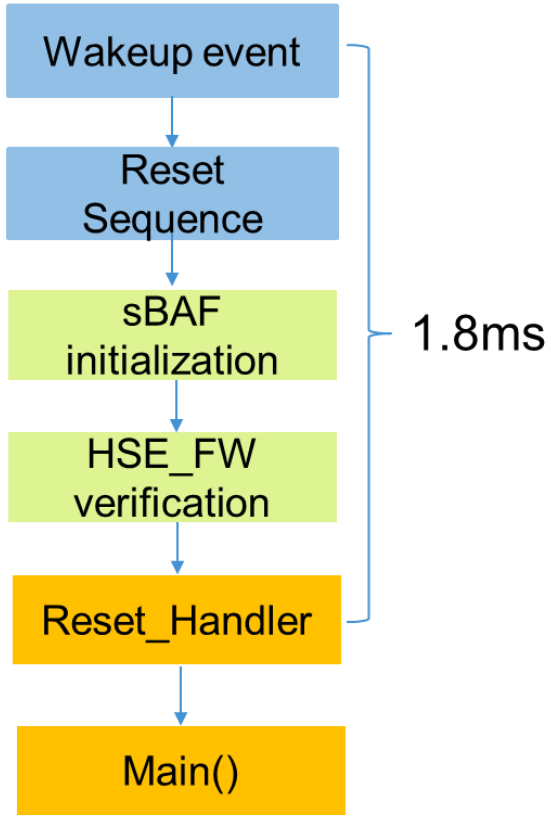
LifeCycle DEMO Project.zip

SOME IMPORTANT NOTES

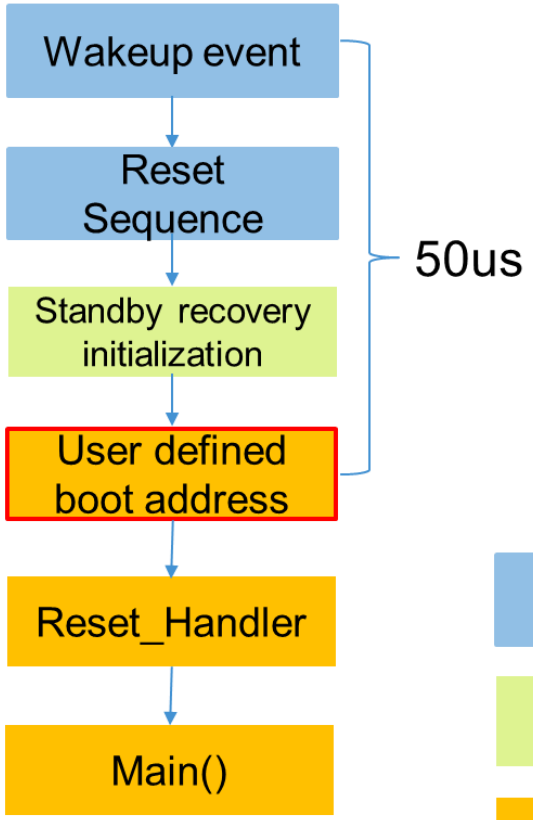
NOTE -- HSE Doesn't WORK AFTER FAST WAKEUP

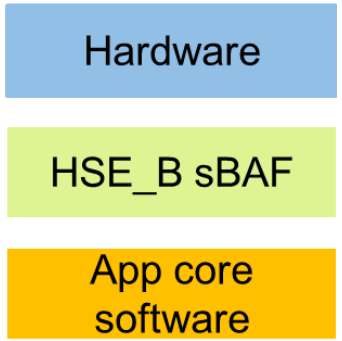
✓ When using fast wakeup, the HSE is not initialized and can not be use after boot

Normal standby exit procedure



Fast standby exit procedure





NOTE -- INCONSISTENT DATA BY CACHE

- ✓ The M7 core and HSE have data synchronization problems. Pay attention to putting some key information in the non-cacheable area.
- ✓ Any variables(hseSrvDescriptor, input/output length/size…) used in the interaction between the core and HSE need to be placed in a non-cacheable area (DTCM, non-cacheable SRAM…).

PROGRAM CODE

```
#define CRYPTO_START_SEC_VAR_CLEARED_32_NO_CACHEABLE
#include "Crypto_MemMap.h"

static uint32 App_u32HashResultSize;

#define CRYPTO_STOP_SEC_VAR_CLEARED_32_NO_CACHEABLE
#include "Crypto_MemMap.h"
```

```
#define CRYPTO_START_SEC_VAR_INIT_32_NO_CACHEABLE
#include "Crypto_MemMap.h"

static uint32 App_u32CmacResultSize = APP_CMAC_TAG_SIZE;

static uint32 App_u32MacStreamResultSize = APP_CMAC_TAG_SIZE;

#define CRYPTO_STOP_SEC_VAR_INIT_32_NO_CACHEABLE
#include "Crypto_MemMap.h"
```

MCAL BSS

MCAL DATA

LINKFILE

```
.non_cacheable_bss (NOLOAD) :
{
    . = ALIGN(16);
    __non_cacheable_bss_start = .;
    *(.mcal_bss_no_cacheable)
    . = ALIGN(4);
    __non_cacheable_bss_end = .;
} > int_sram_no_cacheable

.non_cacheable_data : AT(__non_cacheable_data_rom)
{
    . = ALIGN(4);
    __non_cacheable_data_start__ = .;
    . = ALIGN(4096);
    __interrupts_ram_start = .;
    . += __interrupts_rom_end - __interrupts_rom_start;
    . = ALIGN(4);
    __interrupts_ram_end = .;
    *(.mcal_data_no_cacheable)
    . = ALIGN(4);
    *(.mcal_const_no_cacheable)
    . = ALIGN(4);
    HSE_LOOP_ADDR = .;
    LONG(0x0);
    __non_cacheable_data_end__ = .;
} > int_sram_no_cacheable
```

NOTE -- HSE FW & SBAF COMPATIBILITY

✓ Check with latest HSE RM document to update sBAF to make it full compatibility with HSE FW.

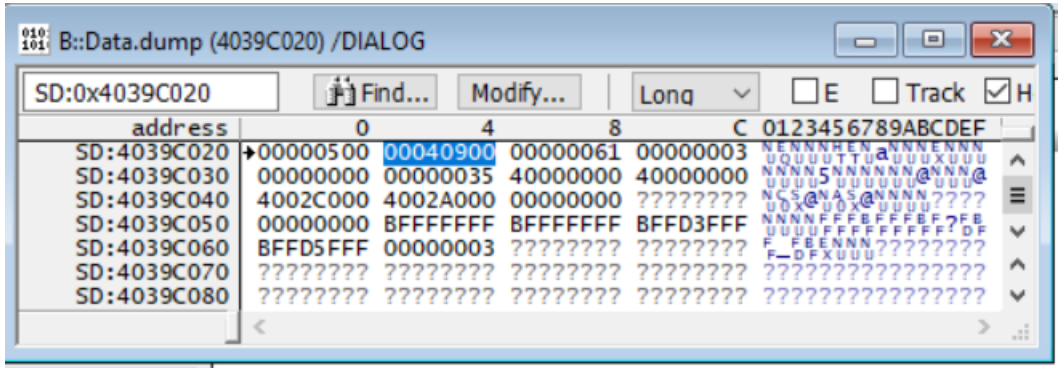
14.3 HSE Firmware and Secure BAF release version compatibility

The below table lists the compatibility between the HSE firmware and the SBAF for S32K344 and S32K342 devices. For all other devices of S32K3 family, all HSE FW versions are compatible with SBAF.

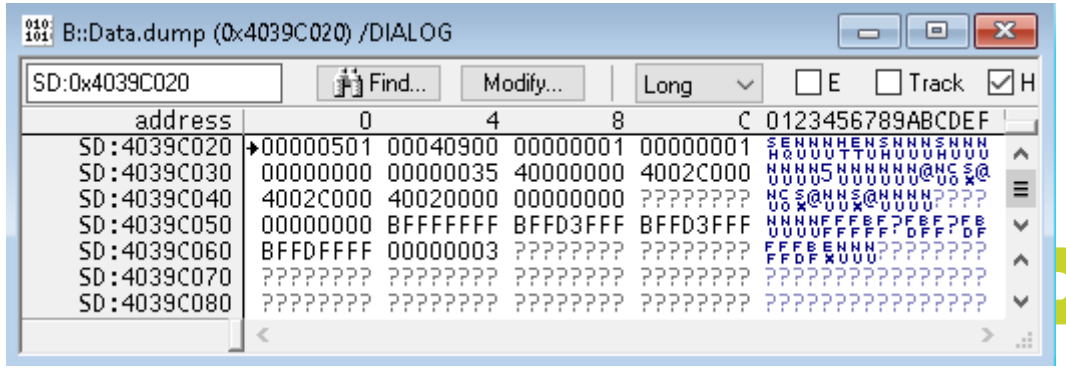
Table 143: HSE Firmware and Secure BAF release version compatibility

Secure BAF version number	HSE FW version number	Remarks
00 05 00 00 00 08 00 11	00 05 00 00 00 08 00 03 or 00 05 00 00 08 08 00 08	Fully Compatible. All functionalities of HSE Firmware supported.
00 05 00 00 00 08 00 11	00 05 00 00 00 **0A** 00 00 or **higher**	All functionality supported except firmware update service HSE_SRV_ID_FIRMWARE_UPDATE is not supported.
00 05 00 00 00 09 04 00	00 05 00 00 00 08 00 03 or 00 05 00 00 08 08 00 08	Not compatible.
00 05 00 00 00 09 04 00	00 05 00 00 00 **0A** 00 00 or **higher**	Fully Compatible. All functionalities of HSE Firmware supported.

Full mem SBAF version 00 05 00 00 00 09 04 00



AB swap SBAF version 01 05 00 00 00 09 04 00



NOTE – CLOCK CONFIGURATION

- ✓ The proper operation of the HSE subsystem depends on the correct configuration of the clocks CORE_CLK, HSE_CLK, AIPS_SLOW_CLK, AIPS_PLAT_CLK, etc. Therefore, users need to follow the **clock option in the S32Kxx-RM**, otherwise the HSE may not operate properly, or even HSE FW will be erased.
- ✓ **Note that K312 cannot be configured as clock option A**

Clocking options	Option A	Option B	Option C	Option D	Option E	Option E2	Option F	Option G
	High-performance mode	Reduced speed mode	Boot (default) Standby configuration (for low dynamic current consumption)	FIRC divider bypassed	Low speed Run mode, clocked by divided FIRC	Very low speed Run mode, clock by divided FIRC	Operation in 1:1 mode with core and AXBS at same speed	PLL providing 48 MHz (test bench use case)
System clock source (SYS_CLK)	PLL_PHI0_CLK	PLL_PHI0_CLK	FIRC_CLK ÷ 2	FIRC_CLK	FIRC_CLK ÷ 2	FIRC_CLK ÷ 16	PLL_PHI0_CLK	PLL_PHI0_CLK
PLL VCO frequency	480 MHz	480 MHz	—	—	—	—	480 MHz	480 MHz
PLL_PHI1_CLK	K344: 240 MHz (VCO ÷ 2) K342: 160 MHz (VCO ÷ 3)	K344: 240 MHz (VCO ÷ 2) K342: 160 MHz (VCO ÷ 3)	—	—	—	—	K344: 240 MHz (VCO/2) K342: 160 MHz (VCO/3)	—
PLL_PHI0_CLK	160 MHz (VCO ÷ 3)	120 MHz (VCO ÷ 4)	—	—	—	—	160 MHz (VCO ÷ 3)	96 MHz (VCO ÷ 5)
CORE_CLK (application cores, AXBS, SRAM, AIPS0,	160 MHz (SYS_CLK)	120 MHz (SYS_CLK)	24 MHz (FIRC_CLK	48 MHz (FIRC)	3 MHz ((FIRC ÷ 2)	187.5 kHz ((FIRC ÷	80 MHz (SYS_CLK	48 MHz (SYS_CLK

The chip supports 1:1 clocking mode, whereby the core(s) are clocked at the same frequency as the slave ports (flash memory, PRAM controller, AIPS controller). Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed supports this requirement.

The frequencies in the table above are maximum frequencies for a specific clock. However, any clock frequency selected must adhere to the same clock divider ratios shown in Clocking use case examples.

NOTE – HSE CLOCK GASKET SETTING FOR S32K312

31.8 Reduced clock mode configuration

If you use clocking option B (Reduced clock mode configuration), the application sets the “dcf_client_utest_misc” DCF record to enable Reduced Clock mode. See the DCF clients file attached to the S32K3xx reference manual for more information on DCF records.

Must write DCF to configure FXOSC frequency and option B

```
#if 1
    uint8_t isWriteDCF = 0 ;
    if( 1 == isWriteDCF )
    {
        /* enable utest fxosc usage and dcf clock option
         * with pll enable in ivt.beq, secure boot
         * verification can be accelerated */
        EnableFXOSCUsage();

        ChangeDcfClockOption();
    }
#endif
```

HSE_CLK_MODE_AND _GSKT_CTRL	30-29	hse Clock mode selection and Bypass HSE IAHB gasket control	These bits control the HSE clock control and HSE_IAHB gasket operation in different clocking options as listed in field descripiton.	The values in different clocking options are as listed below: 00 = Applicable for clocking option A. ***(non_cust: and HSE 60MHz configuration in option B)*** Ratio of 1:2 in between HSE subsystem programming interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK). HSE_IAHB gasket enabled. 01 = Applicable for clocking option C, D, E, E2, and F. Ratio of 1:2 in between HSE subsystem programming interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK). HSE_IAHB gasket bypass. 1x = 10b and 11b both are applicable for clocking option A+ ***(non_cust: for S32K388)*** and B ***(non_cust: HSE 120MHz mode configuration in option B)*** in same way. Ratio of 1:4 in between HSE subsystem programming interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK). HSE_IAHB gasket enabled.

NOTE – HSE GPR

- ✓ The HSE GPR(0x4039C028) provides some information about the internal working status of the HSE. In case of some abnormal conditions that prevent the HSE from working or the firmware is erased, the user should immediately check the HSE GPR value and save it for further analysis.

14.2.6.2 HSE GPR Register 3

Secure BAF updates status bits on HSE GPR Register 3 (0x4039C028) as explained in below table.

Table 136: Status Bits on HSE GPR Register 3 (0x4039C028)

Bit #	Description
31…	Reserved
5	Application cores booted in Recovery mode by SBAF.
4	No HSE Firmware is present in Device due to Erase performed by SBAF Handshake logic. This bit resets on presence of valid HSE Firmware.
3	HSE Firmware from Data flash area is erased by SBAF Handshake logic in current reset cycle.
2	HSE Firmware from code flash area is erased by SBAF Handshake logic in current reset cycle.
1	MU interface is enabled for installation of HSE Firmware.
0	HSE FW is present and SBAF Booted HSE Firmware
6	Indicates that SBAF performs the debug authentication.

NOTE – FLASH synchronization

- ✓ To avoid synchronization issues, HSE Firmware sets the write block whenever it is executing or reading from a block and sets the read block CONFIG_GPR3 at the address (0x4039C028).

Don't neglect not to perform HSE key related operations after operating MCAL-EEEPROM to avoid D-FLASH read/write conflicts

Table 130: HSE_READ_WRITE_LOCK REGISTER (CONFIG_GPR3 address 0x4039C028)

Bit #	Num of bits	Application Access	Description
30-31	2	R	Reserved
29	1	R	Application Flash Read is Blocked for Block 4
28	1	R	Application Flash Read is Blocked for Block 3
27	1	R	Application Flash Read is Blocked for Block 2
26	1	R	Application Flash Read is Blocked for Block 1
25	1	R	Application Flash Read is Blocked for Block 0
24	1	R	Application Flash Read is Blocked for UTEST
22-23	2	R	Reserved
21	1	R	Application Program and Erase Blocked for Block 4
20	1	R	Application Program and Erase Blocked for Block 3
19	1	R	Application Program and Erase Blocked for Block 2
18	1	R	Application Program and Erase Blocked for Block 1
17	1	R	Application Program and Erase Blocked for Block 0
16	1	R	Application Program and Erase Blocked for UTEST
0-15	15	R	Used for other applications

Note :
Data Flash is indicated at different flash blocks for different devices.
For 4MB/8MB devices the data flash is denotes by Block 4.
For 2MB/1MB devices the data flash is denotes by Block 2.
For 6MB devices the data flash is denotes by Block 3.

Application needs to read from CONFIG_GPR3 before read or write on flash so the synchronization issue can be avoided between HSE and application core(s).

Important!

In AB_SWAP configuration, HSE_READ_WRITE_LOCK REGISTER work on physical flash Blocks and not on swapped blocks. In case of higher block being active and firmware is executing from physical block 1 (for 2MB devices) and physical block 3 (for 4MB devices) although the addressing of flash still represents to block 0 and block 1 respectively.

NOTE – FLASH synchronization

- ✓ The FLASH will **continue to be occupied** after the HSE execution service is completed, and if the M7 core performs the FLASH erase or write, the operation will fail.

- ✓ In debug window, found c40 flash - Module Configuration Status (MCRS) - Program and Erase Protection Error (PEP) report an error.

17 PEP	Program and Erase Protection Error PEP provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a locked sector or super sector, or if an erase is selected to a locked sector or super sector. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations (either a Program or Erase) will not be attempted for this request. NOTE If a location has both OTP and Lock protection, the response from the NVM will be PEP=1 only. If PEP is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0b - Program and erase protection errors do not exist 1b - Previous program or erase protection error encountered

```
/* check if HSE is locked flash */
while ( 1 == HostFlash_CheckHseFlashLock() );

SuspendAllInterrupts();
/* Erase sector */
C40_Ip_Status = C40_Ip_MainInterfaceSectorErase(vSector, FLS_MASTER_ID);
HOST_FLASH_ASSERT((STATUS_C40_IP_BUSY == C40_Ip_Status) || (STATUS_C40_IP_SUCCESS == C40_Ip_Status));
do
{
    C40_Ip_Status = STATUS_C40_IP_SUCCESS ;
    C40_Ip_Status = C40_Ip_MainInterfaceSectorEraseStatus();
    HOST_FLASH_ASSERT((STATUS_C40_IP_BUSY == C40_Ip_Status) || (STATUS_C40_IP_SUCCESS == C40_Ip_Status));
}
while (STATUS_C40_IP_BUSY == C40_Ip_Status);
ResumeAllInterrupts();
```

```
/* add check HSE flash lock function before erase or write s */
bool HostFlash_CheckHseFlashLock(void)
{
    volatile uint32_t HseGprVal;
    bool IsHseFlashLock;

    HseGprVal = *( (uint32_t*)HSE_GPR_3_ADDR );
    IsHseFlashLock = (0 != (HseGprVal & HSE_FLASH_LOCK)) ? 1 : 0;

    /* stop if HSE is locked flash */
#if 0
    HOST_FLASH_ASSERT(0 == IsHseFlashBlock);
#endif

    return IsHseFlashLock ;
}
```

NOTE – HSE FIRC divider in RTD 2.0.0 clock init

- ✓ After the HSE FW is installed, the highest 3 bits of the GPR register are set to 0, which means that the application core cannot change the FIRC divider. However, the current RTD 2.0.0 code doesn't check and change it directly, that makes Hardfault.

Field	Function
31-29 APP_CORE_ACC	FIRC Divider NOTE While writing to this register, APP_CORE_ACC is RO and should not be changed from 0b101. 101b - Application core can write this field [FIRC_DIV_SEL] All other values - No access to application core

```
296
297     if (FALSE == TimeoutOccurred)
298     {
299         RegValue = IP_CONFIGURATION_GPR->CONFIG_REG_GPR;
300         if((RegValue & 0xA0000000) == 0xA0000000)
301         {
302             RegValue &= ~CONFIGURATION_GPR_CONFIG_REG_GPR_FIRC_DIV_SEL_MASK;
303             RegValue |= CONFIGURATION_GPR_CONFIG_REG_GPR_FIRC_DIV_SEL(DividerValue);
304             IP_CONFIGURATION_GPR->CONFIG_REG_GPR = RegValue;
305         }
```

Add the FIRC access checking code

main.c | main.c | exceptions.c | system.c | S32K344_MPU.h | Mcu.c | Mcu_Ipw.c | Clock_Ip.c | Clock_Ip_Divider.c

```
254 #ifdef CLOCK_IP_FIRC_DIV_SEL_HSEb_CONFIG_REG_GPR
255 #define CLOCK_IP_WFI_EXECUTED_MC_ME_PRTN0_CORE2_STAT_WFI_MASK
256 static void Clock_Ip_SetFircDivSelHseBConfigRegGpr(Clock_Ip_DividerConfigType const* Config)
257 {
258     uint32 RegValue;
259     uint32 DividerValue = 0U;
260
261     boolean TimeoutOccurred = FALSE;
262     uint32 StartTime;
263     uint32 ElapsedTime;
264     uint32 TimeoutTicks;
265     uint32 WfiStatus;
266
267     switch(Config->Value)
268     {
269         case 1U:
270             DividerValue = 3U;
271             break;
272         case 2U:
273             DividerValue = 1U;
274             break;
275         case 16U:
276             DividerValue = 2U;
277             break;
278         default:
279             /* No option in hardware for this value */
280             break;
281     }
282
283     /* if divider value option from configuration is valid */
284     if (DividerValue != 0U)
285     {
286         /* Before access to CONFIG_REG_GPR register, driver should wait for Secure BAF to go in WFI
287            by reading register PRTN0_CORE2_STAT. Wfi status will be checked. */
288         Clock_Ip_StartTimeout(&StartTime, &ElapsedTime, &TimeoutTicks, CLOCK_IP_TIMEOUT_VALUE_US);
289         /* Wait for acknowledge to be cleared. */
290         do
291         {
292             WfiStatus = (IP_MC_ME->PRTN0_CORE2_STAT & MC_ME_PRTN0_CORE2_STAT_WFI_MASK);
293             TimeoutOccurred = Clock_Ip_TimeoutExpired(&StartTime, &ElapsedTime, TimeoutTicks);
294         }
295         while ((CLOCK_IP_WFI_EXECUTED != WfiStatus) && (FALSE == TimeoutOccurred));
296
297         if (FALSE == TimeoutOccurred)
298         {
299             RegValue = IP_CONFIGURATION_GPR->CONFIG_REG_GPR;
300             RegValue &= ~CONFIGURATION_GPR_CONFIG_REG_GPR_FIRC_DIV_SEL_MASK;
301             RegValue |= CONFIGURATION_GPR_CONFIG_REG_GPR_FIRC_DIV_SEL(DividerValue);
302             IP_CONFIGURATION_GPR->CONFIG_REG_GPR = RegValue;
303         }
304     }
305 }
```

NOTE – S32K324 install HSE FW failed and can’t erase chip

- ✓ The newly shipped chip(probably after June or July of '22.) S32K324 is different from the sample P32K344 and already comes with the latest SBAF, which cannot support the old version of HSE FW installation.

After downloading the HSE FW install project (K3x4 ab swap 0.1.1.0) using PE micro, the chip will enter an unknown error state causing the chip not to be erased.

Need to use Jlink or Lauterbach to write it once to restore it to normal.

NOTE -- SRAM SIZE

- ✓ HSE core has SRAM internal, so there is no need to reserve SRAM memory for HSE. The total SRAM in link file could be used by application, while it is changed in example applications.

```
MEMORY
{
    int_flash               : ORIGIN = 0x00400000, LENGTH = 0x003D4000    /* 4096K - 176K (sBAF + HSE)*/
    int_itcm                : ORIGIN = 0x00000000, LENGTH = 0x00010000    /* 32K */
    int_dtcm                : ORIGIN = 0x20000000, LENGTH = 0x00020000    /* 64K */
    int_sram                : ORIGIN = 0x20400000, LENGTH = 0x0002DF00    /* 183.9K */
    int_sram_fls_rsv        : ORIGIN = 0x2042DF00, LENGTH = 0x00000100    /* 0.1K */
    int_sram_stack_c0       : ORIGIN = 0x2042E000, LENGTH = 0x00001000    /* 4KB  */
    int_sram_stack_c1       : ORIGIN = 0x2042F000, LENGTH = 0x00001000    /* 4KB  */
    int_sram_no_cacheable   : ORIGIN = 0x20430000, LENGTH = 0x0000FF00    /* 64KB, needs to include int_res
    int_sram_results        : ORIGIN = 0x2043FF00, LENGTH = 0x00000100
    int_sram_shareable      : ORIGIN = 0x20440000, LENGTH = 0x00004000    /* 16KB */
    ram_rsvd2               : ORIGIN = 0x20444000, LENGTH = 0             /* End of SRAM */
}
```

NOTE -- S32K3X4 ITCM & DTCM SIZE

Memory region Name	Single Core (Size) (S32K314)	Multi Core Lock Step Enable (Size) (S32K344)	Multi Core Lock Step Disable (Size) (S32K324)
SRAM	0x20400000 - 0x2044FFFF (320 KB)		
ITCM_0	0x01000000 - 0x01007FFF (32 KB)	0x01000000 - 0x0100FFFF (64 KB)	0x01000000 - 0x01007FFF (32 KB)
ITCM_1	N/A	N/A	0x01400000 - 0x01407FFF (32 KB)
DTCM_0	0x20000000 - 0x2000FFFF (64 KB)	0x20000000 - 0x2001FFFF (128 KB)	0x20000000 - 0x2000FFFF (64 KB)
DTCM_1	N/A	N/A	0x20400000 - 0x2040FFFF (64 KB)
ITCM_0 Alternate Address	0x11000000 - 0x11007FFF (32 KB)	0x11000000 - 0x1100FFFF (64 KB)	0x11000000 - 0x11007FFF (32 KB)
ITCM_1 Alternate Address	0x11400000 - 0x11407FFF (32 KB)	N/A	0x11400000 - 0x11407FFF (32 KB)
DTCM_0 Alternate Address	0x21000000 - 0x2100FFFF (64 KB)	0x21000000 - 0x2101FFFF (128 KB)	0x21000000 - 0x2100FFFF (64 KB)
DTCM_1 Alternate Address	0x21400000 - 0x2140FFFF (64 KB)	N/A	0x21400000 - 0x2140FFFF (64 KB)

NOTE – Changes in the new released HSE FW (0.2.1.0)

Rev 2.0	06/2022	1. Updated the description of XRDC configuration at various places. 2. IVT content is modified. 3. HSE GPR registers are updated. 4. Memory map of various K3 variants is updated. 5. Added details about SHE UID 6. Updated the UID usage for **Provisioning a device-dependent ADKP**. 7. Updated the **Secure ADKP Provisioning** section. 8. Added more clarifications in **Debug** section. 9. Update the DH private/public keHDy description 10. Updated the key provisioning usage when importing a key: a key imported in an encrypted format must be always authenticated (**7.2.3.2** and **13.4** sections) 11. Updated **Secure Boot and Memory Verification Services**: add the types of secure boot; updated the SMR entry to include the AAD data 12. Added the section which captures the details of various SBAF and HSE firmware versions.

When using RTD 201, the HSE interface has been replaced with 0.2.1.0(full mem)

If the firmware version used by the K344 is 0.1.1.0, an error will occur when executing the SMR install service in the secure boot demo.

Be careful with the HSE interface, it should be the same as the HSE FW version

Otherwise, some HSE services may not execute properly(e.g. secure boot – SMR install)

The HSE FW in the chip and HSE interface header file version need to be the same

```
Running Secure Boot CFG program
Hse-FW Version : 1.5.0.2.1.0 , AB_swap
using interface version : 0.5.0.2.1.0

AB_SWAP Active State : Active ,
AB_SWAP Active Region : High address, flash block 2,3(K3x4) / 1(K3x2)
```

```
Running Secure Boot CFG program
Hse-FW Version : 1.5.0.2.1.0 , AB_swap
using interface version : 1.5.0.2.1.0

AB_SWAP Active State : Active ,
AB_SWAP Active Region : High address, flash block 2,3(K3x4) / 1(K3x2)
```

SECURE CONNECTIONS
FOR A SMARTER WORLD