

UTest NVM

0 ¹	UTest NVM	1B00_0000	1B00_1FFF	8 KB	S32K358, S32K3x4, S32K342, S32K312, S32K311
----------------	-----------	-----------	-----------	------	---

1. The address region number is the same as block number for all the blocks except this one. Address region is called UTest NVM address region in this case.

31.4.1 Feature configuration in virgin device

Feature	Configuration information	Status	Configurability
OTA functionality	Samples are always shipped in the OTA disabled configuration	Disabled	Yes Can be enabled by application by requesting to HSE firmware or Secure BAF
HSE firmware usage feature flag	This flag indicates whether firmware installation allowed in the device or not. By default, this flag is unprogrammed and Secure BAF assumes that firmware installation is not allowed.	Disabled	Yes Can be enabled by programming in the UTEST location. More details on "UTEST Flag description" section.
Secure BAF firmware	Samples are always shipped with Secure BAF programmed	Programmed	No
HSE Firmware	HSE firmware is always programmed by user/ Encrypted and signed firmware image is always delivered to user.	Not programmed	Yes Can be installed by the application software.
Image Vector Table	No IVT is programmed at any of the IVT locations.	Not programmed	Yes Can be programmed by application software.
SWT0	Application core watchdog SWT0 is disabled.	Disabled	Yes It can be optionally enabled by BAF if SWT bit is configured in boot configuration word.

Boot Sequence	Boot sequence is non-secure boot i.e. application are booted by Secure BAF without any authentication.	Non secure boot	Yes It can be changed to secure boot by programming the BOOT_SEQ bit in IVT.
Life Cycle	Samples are always delivered in LC=CUST_DEL	Customer Delivery	Yes It can be changed to OEM_PROD and IN_FIELD by application through Secure BAF and HSE firmware.
Application Core enablement status	All applications cores are in disable state.	Disabled.	Yes Single or all application cores can be enabled at required address by programming the required fields in the IVT.
Clock Source	Fast Internal Reference clock	FIRC	Yes
FIRC Frequency Value	48 MHz	48 MHz	Yes, application can configures after Secure BAF goes in WFI.
Application debug authorization mode	Debug Authorization mode of application cores is password based. It can be changed to challenge response mode by programming the configuration in UTEST.	Password based approach	When HSE Firmware Feature flag is disabled, debug authorization mode cannot be changed. When HSE Firmware Feature flag is enabled, debug authorization mode can be changed to challenge response mode by requesting to HSE firmware.
Application core debug status	Debug of application cores is enabled in customer delivery lifecycle.	Enabled	No

UTest MAP

Start address (hex) (offset from UTEST Row Base Address)	End address (hex)	Allocated size (bytes)	Description	Accessibility
UTEST flash memory				
0	0000000F	16	Reserved	R/W
00000010	0000001F	16	Test Mode Disable Block Select-1	R/W
00000020	0000002F	16	Reserved for application usage	R/W
00000030	0000003F	16	Test Mode Disable Block Select-2	R/W
00000040	00000047	8	Reserved	R/W
00000048	0000007F	56	Reserved for application usage	R/W
00000080	0000009F	32	Debug password(CUST_DB_PSWD_A)	R/W
000000A0	000000FF	96	Reserved	R/W
00000100	00000103	4	Test Mode Disable Override Passcode	R/W
00000104	0000017F	124	Reserved	R/W
00000180	000001FF	128	Reserved	R/W
00000200	00000207	8	Debug Auth Flag	Read Only
00000208	0000020F	8	IVT_XRDC_GMAC flag	Read Only
00000210	0000022F	32	Reserved	Read Only
00000230	0000023F	16	Lifecycle slot 2: OEM_PROD	Read Only
00000240	0000024F	16	Lifecycle slot 3: IN_FIELD	Read Only
00000250	0000025F	16	Lifecycle slot 4: PRE- FA	Read Only
00000260	0000026F	16	Lifecycle slot 5: FA	Read Only
00000270	0000036F	256	Reserved	Read Only
00000370	0000037F	16	APP_DBG_PASSWORD	Read Only
00000380	0000067F	768	Reserved	NA
00000680	000006FF	128	Reserved	NA
00000700	00000707	8	UTEST- DCF Start Record	R/W
00000708	000018FF	5368	UTEST- DCF Records	R/W
00001C00	00001FFF	1024	Reserved	R/W

UTEST memory usage by Secure BAF

Table 170. UTEST memory location usage by Secure BAF

Start address	End address	Size (bytes)	Description	Programmed by	Write protected
1B00_0000h	1B00_0007h	8	HSE Firmware Feature Usage flag. For more detail, see UTEST flag description	Application	No
1B00_0040h	1B00_0047h	8	Unique Chip Identifier (UID) 0	NXP	No
1B00_0050h	1B00_0057h	8	FXOSC enablement flag For more detail, UTEST flag description	Application	No
1B00_0080h	1B00_009Fh	32	Debug password(CUST_D B_PSWD_A). When HSE FW feature Flag is disabled, this location is used by Secure BAF to run the debug authorization feature. Secure BAF copies this value in application expected response	Application	No
			<p>register and this value is used to derive the HSE expected response register. Size of this register is 16B and 1B00_0090h – 1B00_009h is reserved.</p> <p>When HSE Firmware feature flag is enabled, password is programmed by HSE firmware at different location.</p> <p>Password is scanned by DCM during Reset only. Password to be retained in standby.</p>		

1B00_0220h	1B00_022Fh	16	Lifecycle slot 1: CUST_DEL	NXP	Yes
1B00_0230h	1B00_023Fh	16	Lifecycle slot 2: OEM_PROD	Secure BAF	Yes
1B00_0240h	1B00_024Fh	16	Lifecycle slot 3: IN_FEILD	Secure BAF	Yes
1B00_0250h	1B00_025Fh	16	Lifecycle slot 4: Pre-FA	Secure BAF	Yes
1B00_0260h	1B00_026Fh	16	Lifecycle slot 5: FA	Secure BAF	Yes

Debug password(CUST_DB_PSWD_A).不可见,, , hse fw 能读hash (SHA2-224) over ADKP

Debugging 特性

Table 5: Host debugging capabilities vs. LC

LC state	Host debugging
CUST_DEL	Host debug open (unrestricted)
OEM_PROD 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

Rm

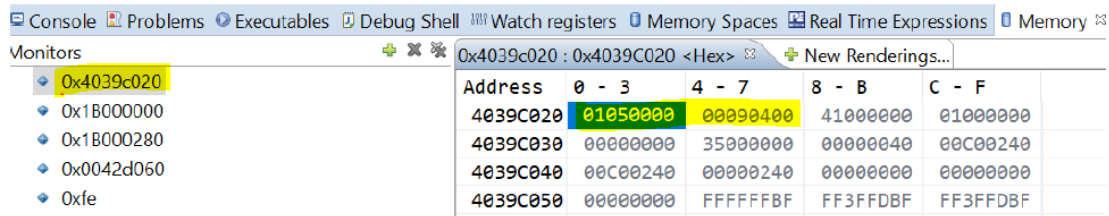
Table 726. Debug access based on LifeCycle and bit configurations

LifeCycle	Configuration bits				Debug access
	DBG_EN_OEM	APP_DIS_OEM	DBG_EN_FLD	APP_DIS_FLD	Appl Core Debug
MCU_PROD	-	-	-	-	Open
	-	-	-	-	Open
CUST_DEL	-	-	-	-	Open
	-	-	-	-	Open
OEM_PROD	0	-	-	-	Closed
	1	0	-	-	Trusted
	1	0	-	-	Trusted
	1	1	-	-	Disabled
IN_FIELD	-	-	0	-	Closed
	-	-	1	0	Trusted
	-	-	1	0	Trusted
	-	-	1	1	Disabled
PRE_FA	-	-	-	-	Trusted
	-	-	-	-	Trusted
FA	-	-	-	-	Open

4. Disabled: Debug has been explicitly disabled via burning of a fuse in that LC, otherwise behaves as Closed

特殊地址

2.再次启动调试,从地址 0x4039c020 读取 SBAF 版本信息,如果为 00050000, 00090400, 则 SBAF 版本以更新到 0.5.0_0.9.4



The screenshot shows a debugger interface with the 'Watch registers' tab selected. A list of monitors is on the left, with '0x4039c020' selected. The main pane displays the register's value in hexadecimal: 01050000 00090400. The address 0x4039c020 is also shown in the title bar of the register pane.

Address	0 - 3	4 - 7	8 - B	C - F
4039C020	01050000	00090400	41000000	01000000
4039C030	00000000	35000000	00000040	00C00240
4039C040	00C00240	00000240	00000000	00000000
4039C050	00000000	FFFFFFBF	FF3FFDBF	FF3FFDBF

MOTOROLA ; majorVersion, minorVersion, patchVersion =16BIT

14.6.4.1 Secure BAF version number

Secure BAF version is a 64bit field. Secure BAF Version can be read by Application from HSE GPR register 0x4039C020. The Version information is explained in below table.

Table 140. Secure BAF version number HSE GPR (0x4039C020)

Bit #	Field Name	Description
56 - 63	RC_NUMBER	Release Candidate Number.
48 - 55	INCREMENTAL_NUMBER	Incremented when new features are added but compatibility kept.
40 - 47	BASELINE_NUMBER	Incremented when the compatibility with the previous version is broken.
32 - 39	RESERVED	Reserved
16 - 31	FW_TYPE	This field identify the FW type: 0 – Standard generic FW targeting all customers 1-7 – Reserved 8 >= Custom1, Custom2....(e.g. Custom1 = customer X's project A, Custom2 = customer Y's project B)
8 – 15	SOC_TYPE_ID	This field Identifies the SoC family* 5 – S32K344, S32K324 and S32K314 devices 12 – S32K311 and S32K310 devices 13 – S32K312, S32K342, S32K322 and S32K341 devices 14 – S32K358, S32K348, S32K338, S32K328 S32K336 and S32K356 devices 15 – S32K396, S32K376, S32K394 and S32K374 devices 16 – S32K388
0 – 7	RESERVED	Reserved

SBAF VERSION

HSE FW 110: s32k3x4_Secure_Baf_0.5.0_0.9.4_pb210708.bin.pink
HSE FW 210: s32k3x4_Secure_Baf_0.5.0_0.10.0_pb220428.bin.pink
HSE FW 260: s32k3x2_Secure_Baf_0.13.0_0.9.0_pb220502.bin.pink

SBAF_S32K312_0_0_15_0_ReleaseNotes.pdf

2 Release Details

This is the HSE Secure-BAF (SBAF) 0.15.0 RTM release for S32K312 for FULL_MEM configuration. The SBAF release of AB_SWAP configuration is part of the HSE firmware AB_SWAP pink image. Both releases can be used for production.

It is strongly recommended that user should update the SBAF in their existing samples if the features provided in this release are important to them.

The SBAF can be updated only in FULL_MEM configuration. In case of AB_SWAP, the SBAF gets updated as part of the HSE firmware update.

This release was developed and tested using:

- Chip : P32K312NGVPBS
- Mini-Module : XS32K3X2CVB-Q172

~ ~ ~ ~ ~

HSE VERSION

```
typedef struct
{
    uint8_t    reserved;        /**< @brief For HSE-B, it is used for OTA Config: 0 = Full Mem Config; 1 = AB Swap Config.
                                For other SOC type: Reserved, expected to be 0 */
    uint8_t    socTypeId;       /**< @brief Identifies the SoC Type ID; same as HSE_PLATFORM from hse_target.h */
    uint16_t   fwTypeId;        /**< @brief Identifies the FW type:
                                - 0 - Standard FW targeting all customers
                                - 1 - Premium FW targeting all customers
                                - 2-7 - Reserved
                                - 8 >= Custom1, Custom2... etc */
    uint8_t    majorVersion;    /**< @brief Major revision
                                - 0 - Pre-stabilization releases
                                - 1 - at first stable interface release, and increased later if breaking changes
    uint8_t    minorVersion;    /**< @brief Minor revision, bumped on new compatible changes added; <br>
                                reset to 0 on majorVersion bump, if majorVersion>0 */
    uint16_t   patchVersion;    /**< @brief Hotfix release (patch version, bug fix releases).<br>
                                After majorVersion>0, reset to 0 on majorVersion or minorVersion bump. */
} hseAttrFwVersion_t;
```

HSE FW 110: s32k3x4_hse_fw_0.5.0_1.1.0_pb211004.bin.pink
 s32k3x4_hse_fw_1.5.0_1.1.0_pb211004.bin.pink

HSE FW 210: s32k3x4_hse_fw_0.5.0_2.1.0_pb220625.bin.pink FULL MEM
 s32k3x4_hse_fw_1.5.0_2.1.0_pb220625.bin.pink

HSE FW 260: s32k3x2_hse_fw_0.13.0_2.6.0_pb221129.bin.pink
 s32k3x2_hse_fw_1.13.0_2.6.0_pb221129.bin.pink

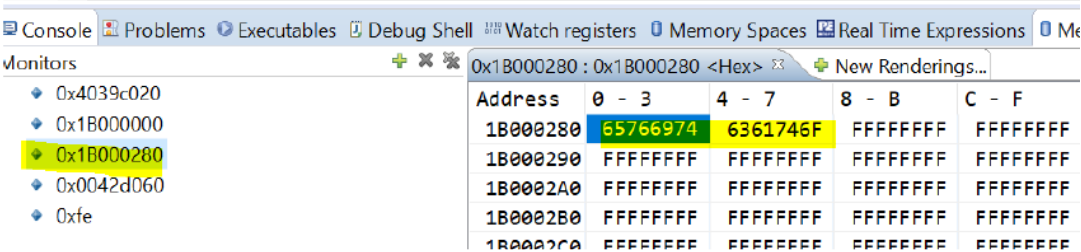
HSE_FW_S32K3XX_0_2_1_0

- S32K344
- S32K324
- S32K314

HSE_FW_S32K3XX_0_2_6_0

- S32K312

读取 OTA ENABLE 地址 0x1B000280,为 65766974 6361746f



```
uint8_t DCMOTAR = 0u; //0 -- low address, 1 -- high address
uint8_t DCMOTAA = 0u; //0 -- inactive, 1 -- active
```

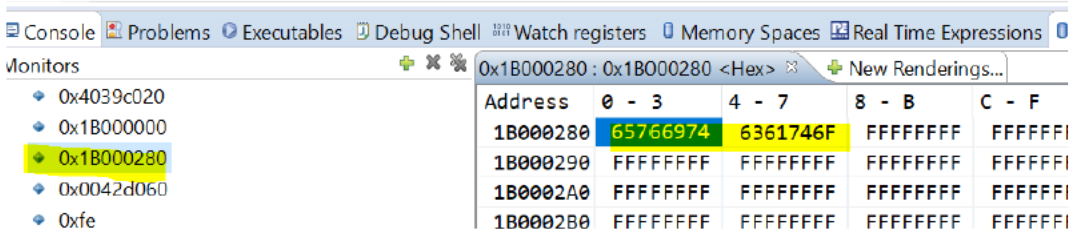
OTA enable

NVM parameters	Block	Address	Length
OTA_INDICATOR_1	Code flash	005F_FBF8h	64-bit
OTA_INDICATOR_2	Code flash	007F_FBF8h	64-bit
OTA_ENABLE	UTest (OTP)	1B00_0160h	64-bit

?? ? 1B000280

如何在 EAR 版本的 SBAF 片子上安装 1_0 之后的 AB SWAP 固件.pdf

读取 OTA ENABLE 地址 0x1B000280,为 65766974 6361746f



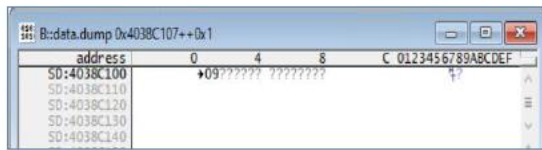
//// HSE_DEMOAPP_S32K3X4_0_1_1_0_ReadMe.pdf

Below figure shows UTEST memory counter value when AB SWAP FW is installed in the device.



Figure 70. UTEST memory Layout when AB swap HSE FW installed

FW install is said to be successful when HSE FW init bit 0 at location 0x4038C107 is set to 1 as shown below. (MU_0)



address	0	4	8	C	0123456789ABCDEF
SD:4038C100	09	??????	????????		
SD:4038C110					
SD:4038C120					
SD:4038C130					
SD:4038C140					

Figure 32. HSE FW install bit

Mu map

MU_0	MU_0_MUB	16	0x4038C000
MU_1	MU_1_MUB	16	0x40390000

HSE status

FSR 0x4038C000+0x104

Table 8: HSE global status bits in FSR

Bit #	Description
31	RFU
30	HSE_STATUS_PUBLISH_NVM_KEYSTORE_RAM_TO_FLASH: signals the application to publish the SYS-IMG to Secure NVM; the host must trigger the service HSE_SRV_ID_PUBLISH_NVM_KEYSTORE_RAM_TO_FLASH
29	HSE_STATUS_FW_UPDATE_IN_PROGRESS; when set to 1, indicates that firmware update is in progress.
28	HSE_STATUS_OEM_SUPER_USER; when set to 1, indicates that SU rights are granted to OWNER_OEM
27	HSE_STATUS_CUST_SUPER_USER; when set to 1, indicates that SU rights are granted to OWNER_CUST
26	HSE_STATUS_BOOT_OK; set to 1 when all the secure boot conditions (pre-boot phase) defined in the HSE successfully pass
25	HSE_STATUS_INSTALL_OK; set to 1 once the key catalogs have been successfully formatted; when cleared to 0, indicates to the host that the key catalogs must be formatted
24	HSE_STATUS_INIT_OK; set to 1 when the HSE initialization is completed; when cleared to 0, no service request can be made to the HSE (MU disabled)
23	HSE_STATUS_HSE_DEBUGGER_ACTIVE; set to 1 when a HSE debug session is active
22	HSE_STATUS_HOST_DEBUGGER_ACTIVE; set to 1 when a host debug session is active
21	HSE_STATUS_RNG_INIT_OK; set to 1 when the RNG initialization is complete; when cleared to 0, any services using random number is unavailable to the host
20	HSE_SHE_STATUS_SECURE_BOOT_OK; set to 1 when SMR #0 successfully verified against BOOT_MAC
19	HSE_SHE_STATUS_SECURE_BOOT_FINISHED; set to 1 when SMR #0 was not successfully verified
18	HSE_SHE_STATUS_SECURE_BOOT_INIT; set to 1 when SMR #0 has been installed and authenticated with BOOT_MAC_KEY
17	HSE_SHE_STATUS_SECURE_BOOT; set to 1 when SMR #0 has been installed and BOOT_SEQ equals 1
16	RFU

15	Set to 1 when service channel #15 execution is in progress
14	Set to 1 when service channel #14 execution is in progress
...	...
2	Set to 1 when service channel #2 execution is in progress
1	Set to 1 when service channel #1 execution is in progress
0	Set to 1 when service channel #0 execution is in progress

```

9
10 &dcm_otaa=(Data.long(SD:0x402ac000)>>10)&0x1
11 &value=Data.byte(SD:0x4038C107)
12 &ivt_header=Data.long(SD:0x00400000)
13
14 Var.Assign \init_status_byte=&value
15 Var.Assign \init_status_byte=(\init_status_byte)&0x1
16
17 VAR.IF ( 0x1 == \init_status_byte )
18 (
19     var.if (0x1 == &dcm_otaa)
20         DIALOG.MESSAGE "AB SWAP HSE FW installed."
21     else
22         DIALOG.MESSAGE "Full HSE FW installed."
23     ;print hse firmware version only when ivt is present
24     VAR.IF (0x5AA55AA5 == &ivt_header)
25     (
26         Var.AddWatch %Hex gHseFwVersion
27     )
28 )
29 VAR.IF (0x0 == \init_status_byte)
30 (
31     var.if (0x1 == &dcm_otaa)
32         DIALOG.MESSAGE "AB SWAP Device.HSE FW not installed."
33     else
34         DIALOG.MESSAGE "Full Mem Device.HSE FW not installed."
35 )
36 data.dump 0x4038C107++0x1
37

```

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.



///S32K2TV_S32K3xx_Training_Section_1-8_rev1.0.pdf

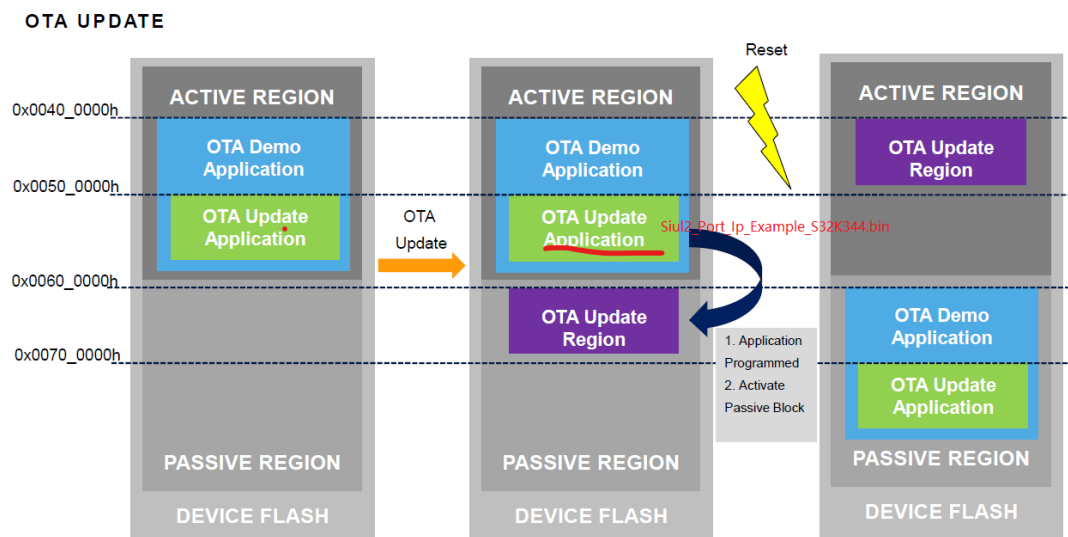
OTA

Ota Demo code

C:\NXP\SW32K3_OTADEMO_0.8.0\S32K344_HSE_OTA

S32K3XX_CryptoDriver_and OTA_AdvancedTraining.pdf

Memory map



Flash

Erasing of selected sector or block erase

8kB sectors in every block

Boot

31.4 IVT (NON secure)

IVT Image Vector Table.

Table 145. IVT

Address offset	Size in bytes	Content	Comments
00h	4	IVT marker	It is a magic number which marks the starting of Image Vector Table location. Its value must be 5AA5_5AA5h.
04h	4	Boot Configuration Word	Configuration word that allows the user to select the various configuration in which device can be booted. See Boot configuration word for details.
08h	4	Reserved	—
0Ch	4	CM7_0 application start address	Boot address of CM7_0 application in code flash area. It must honor core VTOR alignment restrictions. This field will be used by SBAF when BOOT_SEQ bit is 0.
10h	4	Reserved	—
14h	4	CM7_1 application start address	Boot address of the CM7_1 application code in flash memory. It must honor core VTOR alignment restrictions. This field will be used by SBAF only when BOOT_SEQ bit is 0. *This will be ignored if Lockstep configuration is enabled.
18h	8	Reserved	—
20h	4	Reserved	—
24h	4	Address of LC configuration	Address of Configuration word that allows User to Advance LC. Details are mentioned below.
28h	4	Reserved	—
2Ch	4	Reserved	—
30h	192	Reserved	—
F0h	16	Reserved	—

Table 175. Boot configuration register

[illegible]

IVT secure boot

Table 109: IVT structure

Offset	Byte size	Category	Description	Value / Value type
0x00	4	Tag	IVT header tag	Magic number (0x5AA55AA5)
0x04	4	Configuration	BCW	Bit field
0x08	4	Reserved		
0x0C	4	Executable	Apps for BOOT_TARGET bit #0	Pointer
0x10	4	Reserved		
0x14	4	Executable	Apps for BOOT_TARGET bit #1	Pointer ^[1]

Offset	Byte size	Category	Description	Value / Value type
0x18	4	Reserved		
0x1C	4	Executable	Apps for BOOT_TARGET bit #2	Pointer ^{[1] [2]}
0x20	4	Configuration	CFG: XRDC configuration	Pointer
0x24	4	Configuration	LCW	Pointer ^[1]
0x28	4	Reserved		
0x2C	4	Executable	FW-IMG	Pointer ^[1]
0x30	4	Executable	AppBL	Pointer
0x34	12	Reserved		
0x40	4	Executable	Start Address of Application Core for Secure Recovery mode.	Pointer
0x44	4	Length	Length of Recovery Application	32 bits data
0x48	168	Reserved		
0xF0	16	Tag	Authentication tag (GMAC)	Byte array

Table 110: BCW bit mapping

Bit #	Description
31	RFU
30	RFU
...	...
6	DISABLE_SECURE_RECOVERY_MODE
5	SWT0_ENABLE
4	PLL_ENABLE ^{[1] [2]}
3	BOOT_SEQ
2	BOOT_TARGET(CM7_2_ENABLE) ^{[3] [4]}
1	BOOT_TARGET(CM7_1_ENABLE) ^[4]
0	BOOT_TARGET (CM7_0_ENABLE)

^[1] FXOSC enablement flag must be enabled in UTEST area as described in [REF02].

^[2] PLL is configured only when BOOT_SEQ==1

Table 114: PLL Configuration in HSE Firmware

Device	Options Available
S32K312	Option B
S32K342	Option A and Option B
S32K344	Option A and Option B
S32K358	Option A+ and Option B
S32K396	Option A+ and Option B
S32K311	Option B

Table 115: Clock Frequencies for various clocking options in S32K3xx devices (except S32K3X6)

Clock	HSE_CLK	CORE_CLK
Option A	80MHz	160MHz
Option B	120MHz	120MHz
Option A+ (only in S32K358 and its phantoms)	120MHz	240MHz

Only Option A+ is available in S32K3X8 family.

Table 116: Clock Frequencies for various clocking options in S32K3X6 family of devices

Clock	HSE_CLK	CM7_CORE_CLK
Option A+	80MHz	320MHz
Option B	120MHz	240MHz

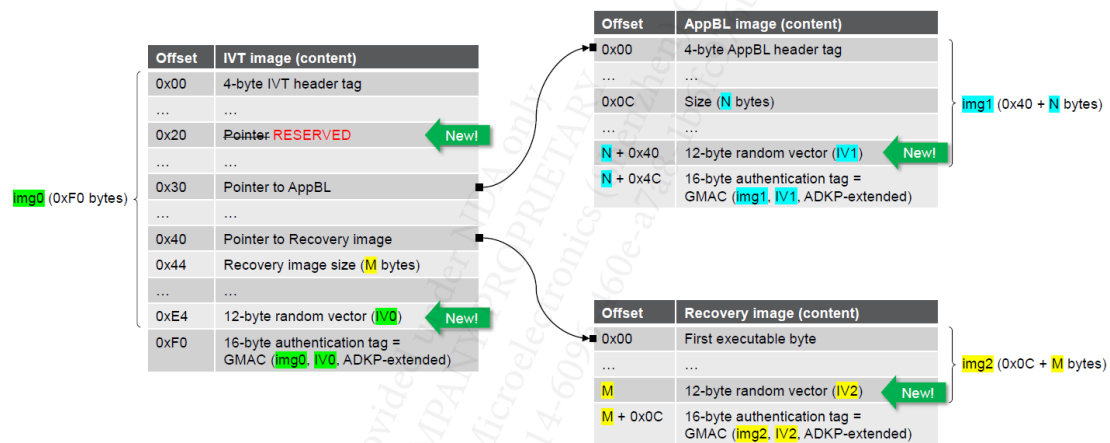
Configuration Name	UTEST Location	Size (in bytes)	Description
HSE Firmware Usage feature flag	0x1B000000	8 bytes	Programming this location enables the security in the device. This flag must be programmed before the HSE firmware installation can be performed. This location can be programmed by any value directly by application in CUST_DEL lifecycle only.
Reserved	0x1B000048	8 bytes	Used by HSE. Must not be programmed by application.
FXOSC configuration	0x1B000050	8 bytes	Oscillator configuration values in case PLL enabled by HSE firmware during secure boot. Refer to "Boot Chapter" in SOC RM for more details.
Partial A/B swap	0x1B000058	8 bytes	Only applicable for S32K3x8 devices. Programming this location with a value "0xDABADABADABADABA" configures the device for partial A/B swap mode when device is converted to A/B swap configuration. For more details refer to "HSE firmware Installation" or "HSE firmware update" section. Note that S32K3x6 devices

			are always configured in Partial A/B swap configuration instead of Normal A/B swap configuration.
JDC clock disable	0x1B000060	8 bytes	Programming this location disables the JDC clock. This helps to save the power consumption of the device. When this feature is enabled, debug authorization is only performed on SDAP interface not on JDC. This feature is supported only for S32K3x6, S32K3x8 and S32K3x1 device family.

HSE FW 0.2.1.0 update

- Changes
 - A random vector is added to IVT, AppBL and recovery images
 - A random vector is returned by the HSE service that computes a GMAC over IVT, AppBL and recovery images
 - The XRDC configuration helper is removed
- Motivation
 - Improved authentication scheme over IVT, AppBL and recovery images
 - Faster boot time by discarding the XRDC configuration

NEW SYSTEM IMAGE STRUCTURES



NEW HSE SERVICE TO CALCULATE AUTHENTICATION TAGS OVER SYSTEM IMAGES

Service structure parameters

```
typedef struct
{
    HOST_ADDR pInImage;
    uint32_t inTagLength;
    HOST_ADDR pOutTagAddr;
} hseBootDataImageSignSrv_t;
```

- Pointer to first byte of IVT, AppBL or Recovery image
 - Size of the image to process
 - Pointer to a buffer of minimum **28** bytes
 After successful execution, the service initializes the buffer with two concatenated arrays as follows:

- A 12-byte random vector New!
- A 16-byte authentication tag

Byte offset in memory	0x00	...	0x0B	0x0C	...	0x1B
Content	First byte of random vector	...	Last byte of random vector	First byte of auth. tag	...	Last byte of auth. tag

Table 112: XRDC configuration structure

Offset	Byte size	Category	Description	Value / Value type
0x00	4	Tag	XRDC configuration header tag	Magic number (0xCC5577CC)
0x04	4	Configuration	MDA_W0_0_DFMT0	Byte array
0x08	4	Configuration	MDA_W0_1_DFMT0	Byte array
0x0C	4	Configuration	MDA_W0_2_DFMT1	Byte array
0x10	4	Configuration	MDA_W0_4_DFMT1	Byte array
0x14	4	Configuration	MDA_W0_5_DFMT1	Byte array
0x18	16	Reserved		
0x20	4	Configuration	PDAC_W0_154	Byte array
0x24	4	Configuration	PDAC_W1_154	Byte array
0x28	4	Configuration	PDAC_W0_158	Byte array
0x2C	4	Configuration	PDAC_W1_158	Byte array
0x30	4	Configuration	PDAC_W0_163	Byte array
0x34	4	Configuration	PDAC_W1_163	Byte array
0x38	4	Configuration	PDAC_W0_175	Byte array
0x3C	4	Configuration	PDAC_W1_175	Byte array
0x40	4	Configuration	PDAC_W0_180	Byte array
0x44	4	Configuration	PDAC_W1_180	Byte array
0x48	4	Configuration	PDAC_W0_181	Byte array
0x4C	4	Configuration	PDAC_W1_181	Byte array
0x50	4	Configuration	PDAC_W0_182	Byte array
0x54	4	Configuration	PDAC_W1_182	Byte array
0x58	4	Configuration	PDAC_W0_183	Byte array
0x5C	4	Configuration	PDAC_W1_183	Byte array
0x60	4	Configuration	PDAC_W0_184	Byte array
0x64	4	Configuration	PDAC_W1_184	Byte array
0x68	4	Configuration	PDAC_W0_186	Byte array
0x6C	4	Configuration	PDAC_W1_186	Byte array
0x70	4	Configuration	PDAC_W0_187	Byte array
0x74	4	Configuration	PDAC_W1_187	Byte array
0x78	4	Configuration	PDAC_W0_223	Byte array
0x7C	4	Configuration	PDAC_W1_223	Byte array
0x80	4	Configuration	PDAC_W0_225	Byte array
0x84	4	Configuration	PDAC_W1_225	Byte array
0x88	4	Configuration	PDAC_W0_280	Byte array
0x8C	4	Configuration	PDAC_W1_280	Byte array
0x90	96	Reserved		
0xF0	16	Tag	Authentication tag (GMAC)	Byte array

LCW is a 32-bit value that specifies the LC state advancement:

- LCW = 0xDADADADA advances LC to OEM_PROD
- LCW = 0xBABABABA advances LC to IN_FIELD

Table 114: AppBL structure

Table 114: AppBL structure

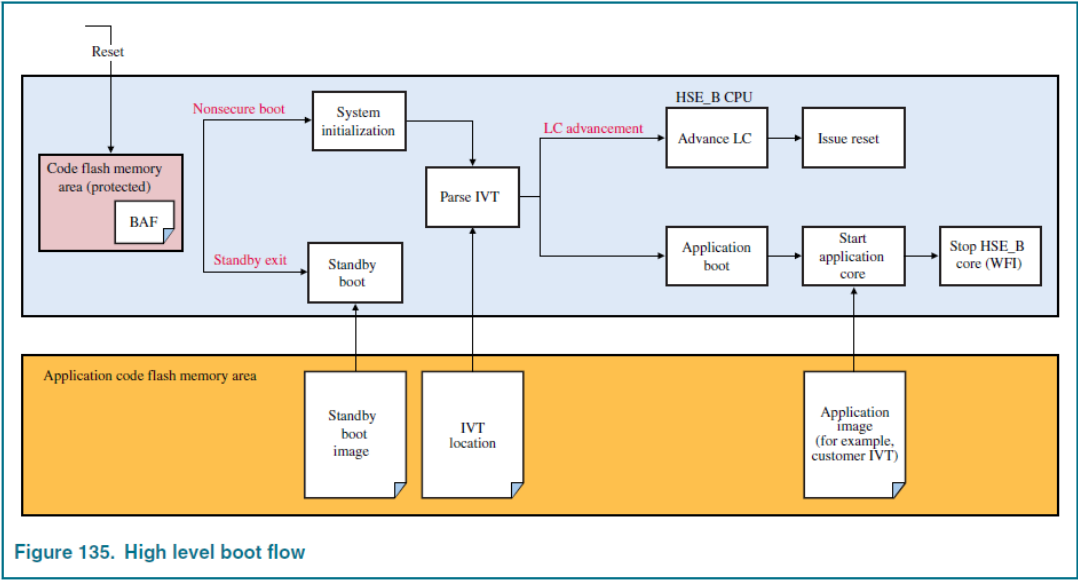
Offset	Byte size	Category	Description	Value / Value type
0x00	1	Tag	AppBL header tag	Magic number (0xD5)
0x01	2	Reserved		
0x03	1	Tag	AppBL version	Magic number (0x60)
0x04	4	Reserved		
0x08	4	Configuration	Start address (in Flash)	Pointer
0x0C	4	Configuration	AppBL size (<i>N</i>)	32-bit integer
0x10	1	Configuration	Core identifier (see Table 120)	Value
0x14	47	Reserved		
0x40	<i>N</i> ⁽¹⁾	Executable	AppBL content (in plain)	Executable
<i>N</i> + 0x40	16	Tag	Authentication tag (GMAC)	Byte array

⁽¹⁾ As defined in offset 0x0C.

Boot flow

31.5 Boot Flow

31.5.1 High level boot flow

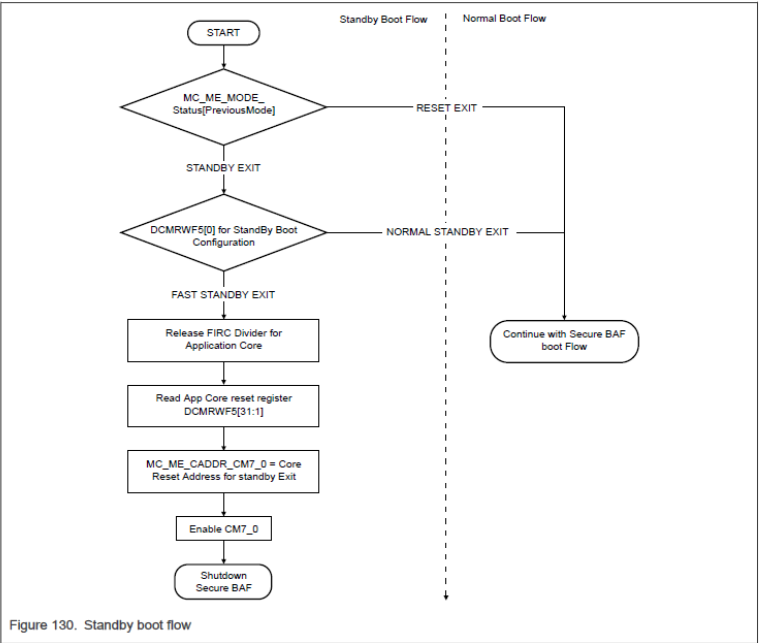


31.7 Standby boot

There are two types of boot mode on exit from standby.

- Fast standby mode
- Normal boot on exit from standby

In Fast standby mode Secure BAF boots CM7_0 and halts the HSE core. The flow of standby boot is explained below:



S32K3_RTD_Training_Startup_Linker_File_and_Customization.pdf

Boot flow diagram in BAF

•
BAF Boot Assist Firmware) is the first codes to run on HSE B secure core after reset, it does necessary system initialization, searches and parses the IVT Image Vector Table), finally stop HSE B, enable SWT and start the application core:

```
__CORE0_VTOR = __interrupts_rom_start;
```

Demo 实际使用, map

```
.flash 0x00400000 0x2253c
*(.boot_header)
.boot_header 0x00400000 0x28 ./Project_Settings/Startup_Code/startup_cm7.o
              0x00401000          . = ALIGN (0x1000)
*fill*       0x00400028 0xfd8
              0x00401000          _text_start = .
              0x00401000          _interrupts_rom_start = .
*(.intc_vector)
.intc_vector 0x00401000 0x380 ./Project_Settings/Startup_Code/Vector_Table.o
              0x00401000          VTABLE
              0x00401380          . = ALIGN (0x4)
              0x00401380          _interrupts_rom_end = .
*(.core_loop)
.core_loop   0x00401380 0xc ./Project_Settings/Startup_Code/startup_cm7.o
              0x00401380          _core_loop
```

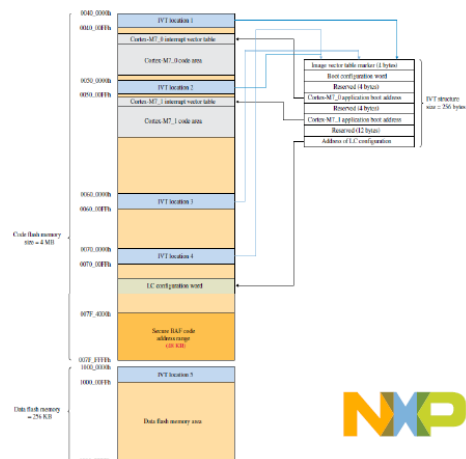
IVT(Image Vector Table) Details

- The IVT location can be the start address (256 bytes) of any available P/D-Flash partition blocks. 6 possible locations for S32K344
- IVT is placed into section ".boot_header" in RTD based project

```
.section ".boot_header", "ax"
.long SBAF_BOOT_MARKER /* IVT marker */
.long (CHT_0_ENABLE << CHT_0_ENABLE_SHIFT) | (CHT_1_ENABLE << CHT_1_ENABLE_SHIFT) /* Boot configuration word */
.long 0 /* Reserved */
.long CHT_0_VTOR_ADDR /* CHT_0 Start address */
.long 0 /* Reserved */
.long CHT_1_VTOR_ADDR /* CHT_1 Start address */
.long 0 /* Reserved */
.long XROC_CONFIG_ADDR /* XROC configuration pointer */
.long IF_CONFIG_ADDR /* Lifecycle configuration pointer */
.long 0 /* Reserved */
```

Table 155. IVT field details

Address offset	Size in bytes	Content	Details
00h	4	IVT marker	Magic number that marks the starting of the IVT location. Its value must be SBAF_SBAF.
04h	4	Boot configuration word	Configuration word that allows you to select the various configuration options available to boot the chip. See <i>Boot configuration word</i> for details.
08h	4	Reserved	—
0Ch	4	Cortex-M7_0 application core start address	Boot address of the Cortex-M7_0 application core in the code flash memory area. It must honor core VTOR alignment restrictions. SBAF uses this field when the BOOT_SEQ field in <i>Boot configuration word</i> is 0.
10h	4	Reserved	—
14h	4	Cortex-M7_1 application core start address	Boot address of the Cortex-M7_1 application core in flash memory. It must honor core VTOR alignment restrictions. SBAF uses this field only when the BOOT_SEQ field in <i>Boot configuration word</i> is 0. This field is ignored if lockstep configuration is enabled.
18h	8	Reserved	—
20h	4	Reserved	—
24h	4	LC configuration address	Configuration word address that allows you to advance the chip's LC to the next stage. See <i>Address LC configuration word</i> for details.
28h	4	Reserved	—
2Ch	4	Reserved	—
30h	192	Reserved	—
F0h	16	Reserved	—



SecureBoot

<https://www2.renesas.cn/us/en/blogs/introduction-about-secure-boot-automotive-mcu-rh850-and-soc-r-car-achieve-root-trust-1>

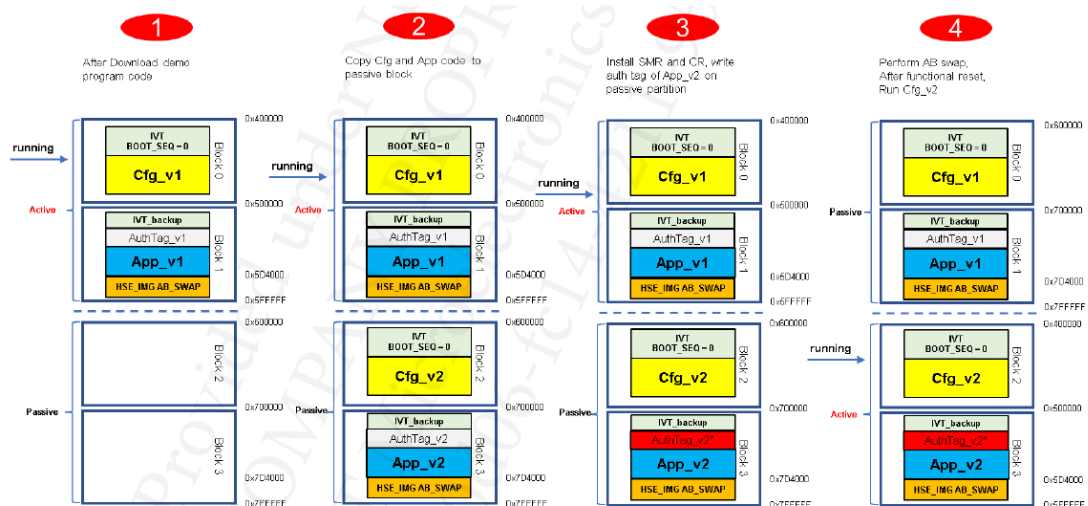
AB SWAP CASE

AN744511-Secure Boot Application note v0.1.1.0(1.1).pdf

7.3. Implement secure boot

To implement secure boot on the OTA enabled device, the users need to know that the **SMR table** is uniquely stored in the S32K3 HSE secure NVM, and the area protected by each SMR and its **auth-tag** will also point to a unique address.

Therefore, it is recommended to store the **auth-tag in the same fixed address** regardless of the active or passive partition, otherwise you need to reinstall the SMR to avoid secure boot failure.



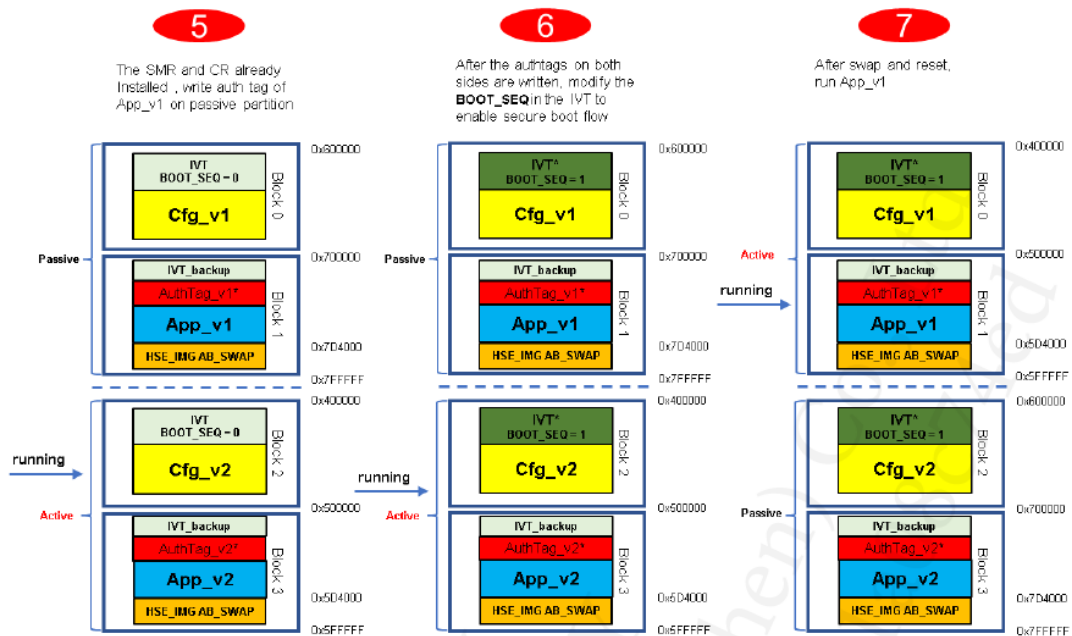
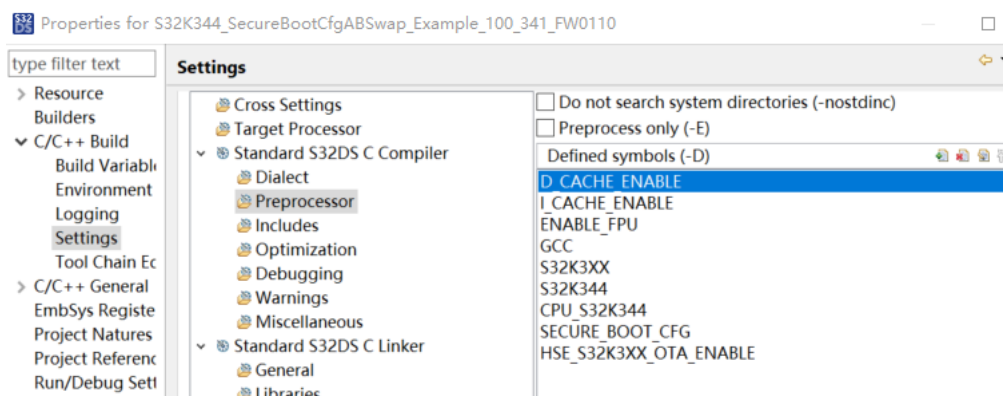


Figure 23. Illustrating secure boot configuration on AB swap device

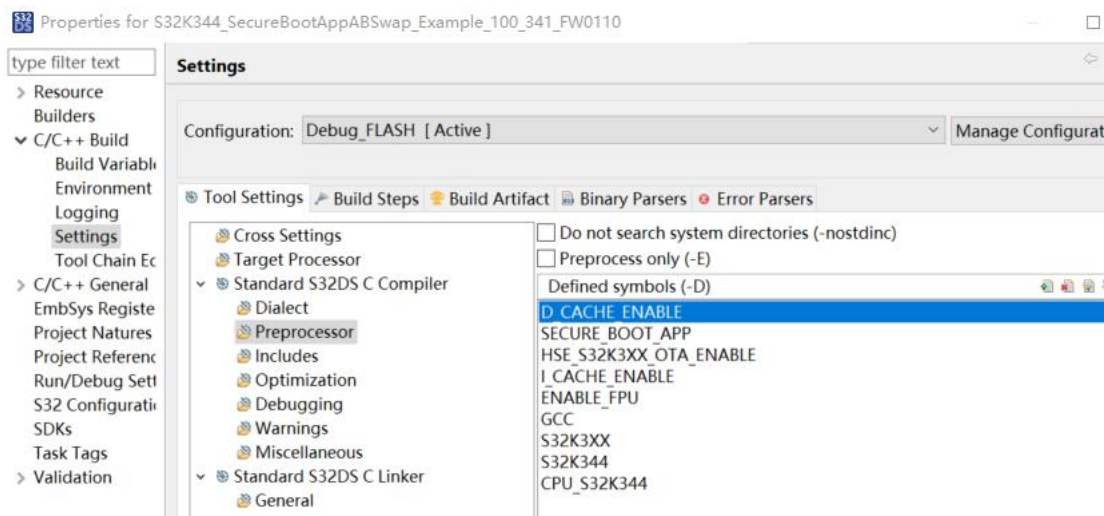
S32K344_SecureBootCfgABSwap_Example_1

00_341_FW0110

Cfg



App



HSE

Terms	Description
Application Image	Binary image containing the user application code. This is stored in plain format and optionally signed if secure boot is enabled.
HSE FW	HSE firmware code: It provides cryptographic security services for the entire platform. HSE FW image delivered is encrypted and signed (see HSE FW Pink image definitions).
HSE FW Pink image	HSE FW image delivered by NXP. This is encrypted and authenticated with keys known by NXP. Authentication (signature generation) is done with an asymmetric algorithm.
HSE FW Image header tag	The first byte of an image header, also known as marker. 0xDA – Full Mem HSE firmware 0xDB – AB SWAP HSE Firmware 0xDC – Encrypted Secure-BAF Image
IVT image	First 256 bytes of data read by HSE firmware after reset. Contains some attributes configured by secure BAF and references to the other images used by secure BAF/HSE FW at boot time.

How do I install HSE FW on a virgin device without secure boot?

Firmware feature flag must be enabled before installing the firmware. HSE FW can be installed in the system using three 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.

6.4. HSE FW Update

HSE-B Firmware Reference Manual

RM559614-HSE-B Firmware Reference Manual -V1.2(1.4).pdf

3.2.4 Host system images

The host system images required to operate in a device with embedded Flash are:

- The Image Vector Table, hereafter referred to as **IVT**.
- Device-specific configuration data / commands, hereafter referred to as **CFG**.
- Various applications images (executable, data, etc.) referred to as **Apps**.
- An (optional) authenticated application image, hereafter referred to as **AppBL** that can run in the host in lieu of the Apps and after the HSE has verified its authenticity.

AppBL 一定是smr验证后运行。

3.3.7 HSE images

3.3.7 HSE images

3.3.7.1 Overview

The HSE images are:

- The HSE firmware executable, here after referred to as **FW-IMG**
- The HSE system image that contains public and private (secret) keys, monotonic counters and configuration data (aka HSE system attributes), hereafter referred to as **SYS-IMG**.

The location, access and update policies that apply to each HSE image depend on the type of host where the HSE integrates.

The HSE_B subsystems have all their images stored in the secure NVM mapping to the embedded Flash:

- FW-IMG is stored in the HSE code Flash area
- FW-IMG backup is stored in the HSE data flash area
- SYS-IMG is stored in the HSE data Flash area

The HSE images are read-out and updated exclusively by the HSE.

SYS IMAGE BREAKUP

Monotonic Counter(8K)
Monotonic Counter and Config data Scratchpad(8K)
HSE Config Data(8K)
Key Store (8K)
Key Store Scratchpad Sector(8K)

Data Type	Description	Size
Monotonic Counter	This data type stores the 16 counters of 64 bits in NVM memory.	8 KB
Monotonic Counter and config data Scratchpad	Once the previous sector of monotonic counter gets full, the updated values of all counters are shifted to scratchpad sector and previous sector becomes the scratchpad. This sector will be used for storing the new version of config data also.	8 KB
HSE Config Data	This data type stores the SMR tables, Core reset table, NVM attributes, User ECC curves	8KB
Key Store	The sector stores all type of user keys which includes symmetric and asymmetric of all key sizes	8KB
Key Data Scratchpad Sector(8K)	This sector acts as passive sector for updating the key and config data. New key file is first saved in this sector and after successful programming this sector becomes the active sector	8KB
Total Size		40KB

3.3.8 Life cycle (LC)

Figure 3: HSE installation / configuration states vs. LC states

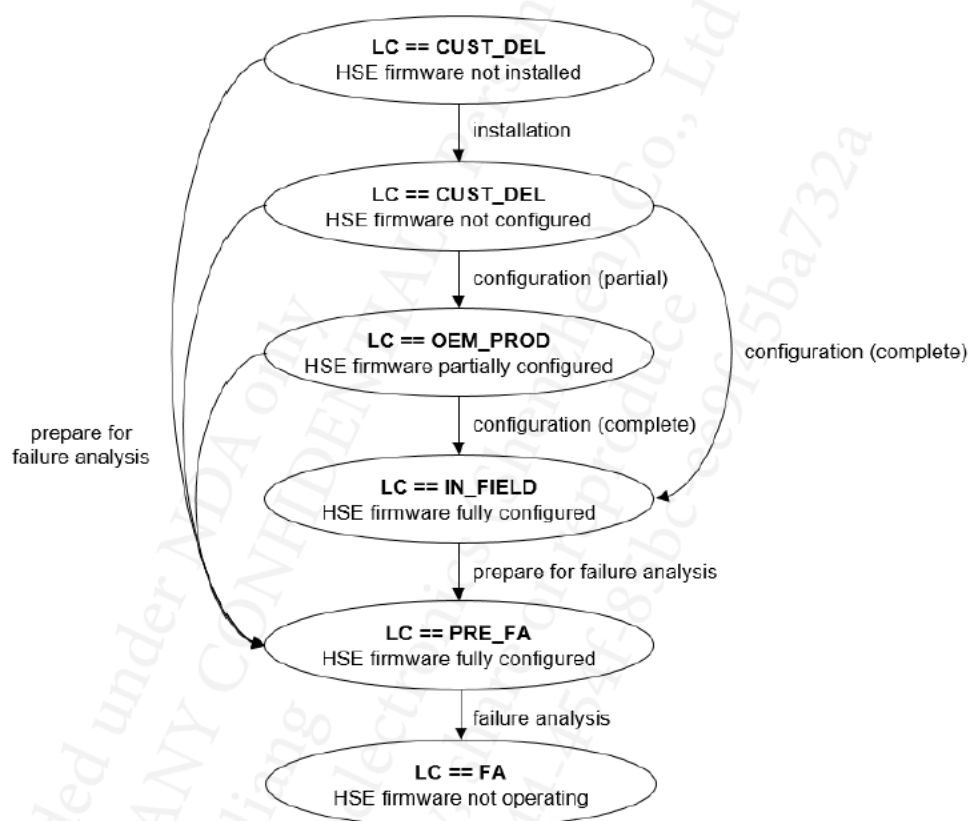


Table 4: HSE Firmware capabilities vs. LC

LC state	HSE firmware capabilities
CUST_DEL	HSE firmware ready for installation and configuration.
OEM_PROD	HSE firmware ready for additional configuration; no restrictions except those implicitly implied by the security policies.

IN_FIELD	Key management capabilities (import, export, etc.) are restricted (most secure state).
PRE_FA	HSE firmware fully configured. It is similar to IN_FIELD.
FA	HSE firmware not operating.

Debug

Table 5: Host debugging capabilities vs. LC

LC state	Host debugging
CUST_DEL	Host debug open (unrestricted)
OEM_PROD 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

3.6.3.3 Host debug permanently disabled

The host debug can be permanently disabled (i.e., closed) in LC states OEM_PROD and IN_FIELD by programming the DCF records in the UTEST area. Refer to [REF02] more details.

Life cycle 推进:

1. 不可逆,
2. 通过 Set HSE system attributes 服务

DEVICE'S LIFE CYCLE AND RELATED SECURITY STATE

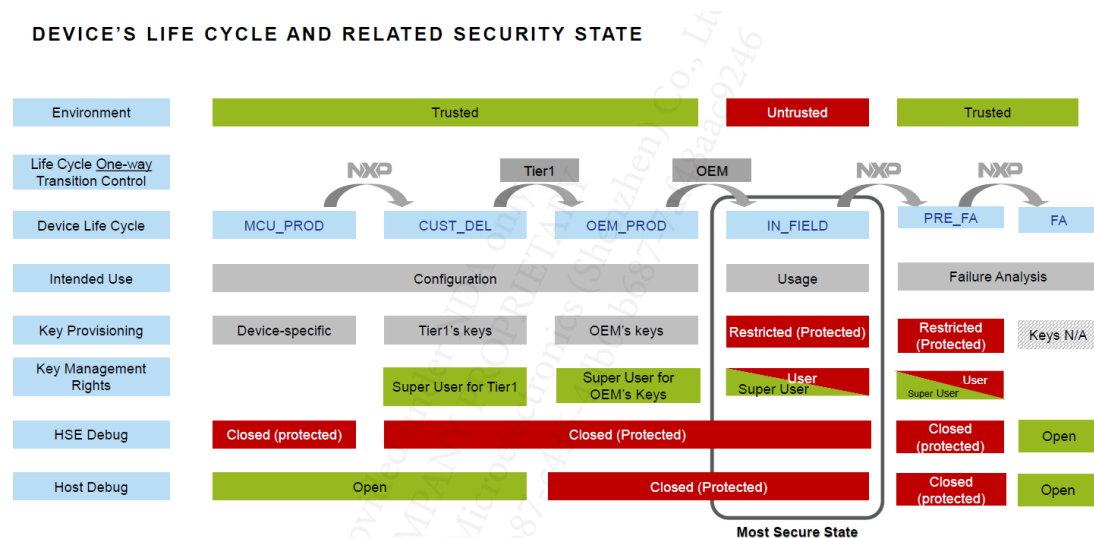
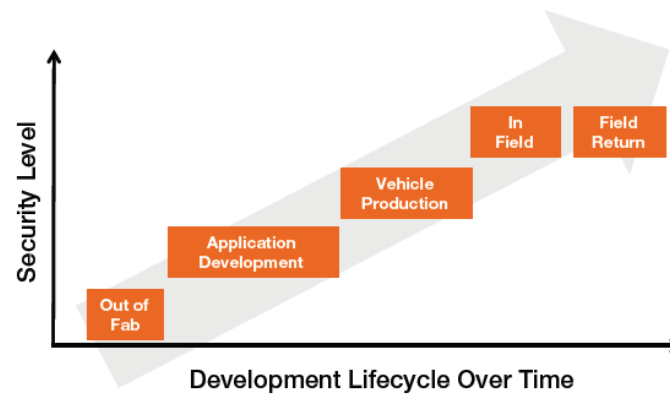


FIGURE 10. Security Level Increases as Product is Developed



3.4 HSE subsystem software components

There are two software components that operate the HSE subsystem:

- SBAF
- HSE Firmware

Secure Boot Assist Flash (aka SBAF)

- HSE firmware installation
- HSE firmware restoration
- Debug authorization
- Partition swapping enablement
- XRDC configuration
- Support in firmware update
- Secure and JTAG based recovery mode

The HSE firmware is the software component that provides a variety of native security services as described below.

- **Administration services** are provided to install, configure and test the HSE.
- **Key management services** are available for the application to manage different sets of keys that are handled by the HSE for example through the cryptographic services.
- **Cryptographic services** provide the application with cryptographic primitives that are used by high-level security stacks in the application.
- **Random number services** generate random streams that can be used in various security protocols.
- **Memory verification services** allow the application to verify different memory areas at start-up (after reset) and during run-time.
- **Monotonic counter services** provide the application with a set of monotonic counters that can be read and only incremented.

3.6.1 Reset (start-up flow)

Table 11: CPU subsystems released from reset by the HSE

Configuration	Identification of the CPU subsystem(s) to release from reset	Release conditions
BOOT_SEQ == 0	BOOT_TARGET (in IVT)	Unconditional
BOOT_SEQ == 1	Core Reset table (in SYS-IMG)	Defined in the Secure Memory Region (SMR) tables

5.2.1 Configurable HSE system attributes

A set of programmable HSE system attributes within the secure NVM can be provisioned by the host via HSE administration services. Some of these attributes, **once set, cannot be updated.**

Table 13: One-time configurable system attributes

Parameter	Size	Description
IVT_AUTH	8 bits	Selects the IVT authentication method: <ul style="list-style-type: none">- When 0 (default): no authentication check- When 1: forces the IVT and CFG authentication check before running the HSE firmware
AUTH_MODE	8 bits	Selects the method to open the host debug protection: <ul style="list-style-type: none">- When 0 (default): static authentication (password)- When 1: dynamic authentication (challenge / response)
ADKP	128 bits	Value of the application debug key or password <ul style="list-style-type: none">- If AUTH_MODE equals 0, ADKP is a password- If AUTH_MODE equals 1, ADKP is a cryptographic key
ADKP_MASTER	8 bits	Selects the method to provision ADKP in secure NVM: <ul style="list-style-type: none">- 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
LC	8 bits	Selects the life cycle: OEM_PROD or IN_FIELD

ADKP_MASTER ----> Extend HSE Security Policies

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

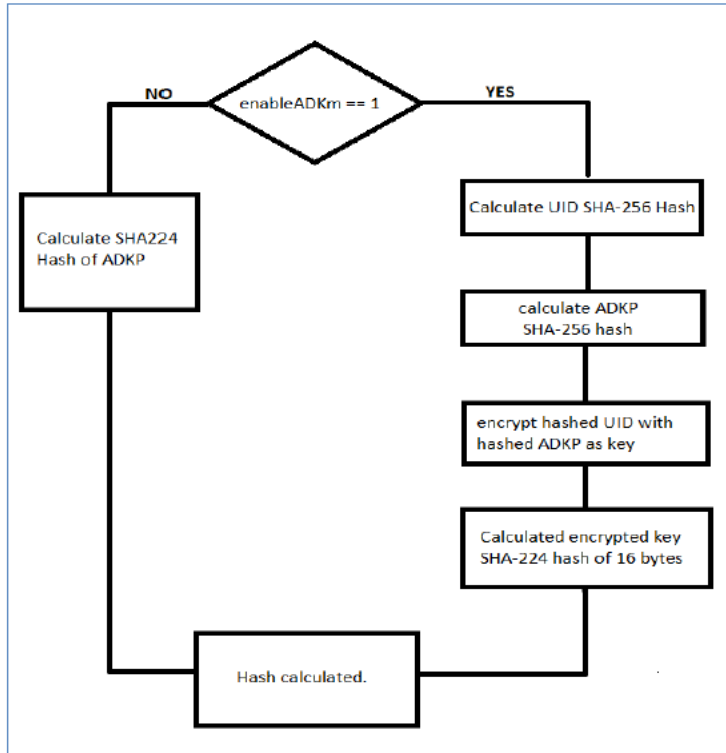


Figure 46. Verify ADKP when enableADKPm=1

If set, the following logic must be used at customer's end for debug-authorization:

- hUID = SHA2_256(UID)
- hADKPm = SHA2_256(ADKPm)
- ADKP {for debugger} = AES256-ECB(hUID(16 bytes..0 to 15)), key = hADKPm; {ADKPm = customer's master key/ password}.

The hash of ADKPm (set using ADKP attribute) will be used as the key in the derivation of the application password.

6.4.2 Execution rights (Super User vs. User)

Table 17: Execution rights and respective limitations in key management

Service		SU rights	User rights
Import a new NVM key (i.e. in an empty key slot)	Encryption	Optional	Mandatory
	Authentication	Optional	Mandatory
NVM key generation (i.e. in an empty key slot)		Possible	Not possible
NVM key deletion		Possible	Not possible
Copy part of a RAM key to an NVM key slot		Possible	Not possible
Load an user defined ECC curve		Possible	Not Possible

Table 18: Execution rights and respective limitations in HSE configuration

Service	SU rights	User rights
Set HSE system attributes	Possible	Not possible (except for SET-ONCE-ATTR attribute types)
Authenticate the host system images (IVT, CFG)	Possible	Not possible
Complete SMR entry update (including key handle)	Possible	Not possible
Update a Core Reset entry	Possible	Not possible
Monotonic counter configuration	Possible	Not possible

HSE AUTHORIZATION DEFAULT STATE

The execution of certain HSE services is conditioned by **the execution rights granted to the host**:

- **SuperUser** (OEM or CUST) rights - high execution privileges and less restrictions on service requests
- **User rights** – restricted execution privileges

After reset, the System Rights are synchronized with Life cycle (LC):

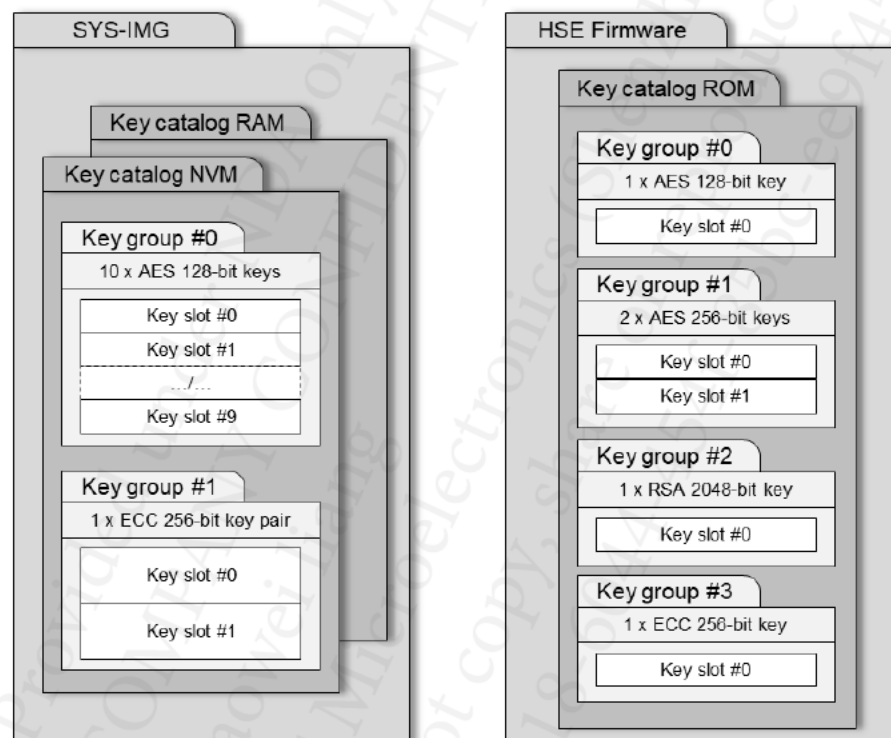
Life Cycle	SYS rights after reset
CUST_DEL	CUST SuperUser rights are granted.
OEM_PROD	OEM SuperUser rights are granted
IN_FIELD	User rights are granted
PRE_FA	User rights are granted

7.1 Cryptographic keys

The keys accessible to the host via the cryptographic services are organized in groups of certain types, within catalogs that are statically configured by the host.

Each key has a value and certain attributes contained into individual key slots, that are dynamically configured by the host via key management services

Figure 32: Illustrating the key catalogs



7.1.2 Key storage

SYS-IMG contains:

- The structure of the RAM and NVM key catalogs
- The NVM key properties
- The NVM key values, or a pointer to the key values stored in the host memory for the key types HSE_KEY_TYPE_RSA_PUB_EXT and HSE_KEY_TYPE_ECC_PUB_EXT (see below)

Key properties and values are updated within SYS-IMG after successful key provisioning operations.

SYS-IMG is saved in secure NVM (i.e., internal Flash) by the host. SYS-IMG is loaded and authenticated by the HSE at start-up.

A **key group** is a set of cryptographic keys of the same type. Each group is identified by an index within the key catalog where it is declared (see next section). The index aligns with the order of declaration in the group: the first group has the index 0, the second group has the index 1, etc.

The below table lists the different key types supported by the HSE.

Table 26: Key types

Key type	Description	Key catalog
HSE_KEY_TYPE_AES	AES key	NVM & RAM
HSE_KEY_TYPE_SHE	AES key used with SHE specific services	NVM & RAM
HSE_KEY_TYPE_HMAC	HMAC key	NVM & RAM
HSE_KEY_TYPE_RSA_PAIR	RSA key pair (public & private)	NVM only
HSE_KEY_TYPE_RSA_PUB	RSA public key	NVM & RAM
HSE_KEY_TYPE_RSA_PUB_EXT	RSA public key, stored in application NVM	NVM & RAM
HSE_KEY_TYPE_ECC_PAIR	ECC key pair (public & private)	NVM & RAM
HSE_KEY_TYPE_ECC_PUB	ECC public key	NVM & RAM
HSE_KEY_TYPE_ECC_PUB_EXT	ECC public key, stored in application NVM	NVM & RAM
HSE_KEY_TYPE_DH_PAIR	DH key pair (public & private)	NVM & RAM
HSE_KEY_TYPE_DH_PUB	DH public key	NVM & RAM
HSE_KEY_TYPE_SHARED_SECRET	Shared secret, can be used to derive a secret key	RAM only

7.1.4 Key slot

A **key slot** is a memory container that holds a single key, with its value(s) and attributes.

HSE Key Format

```
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_MU1_MASK, HSE_KEY_OWNER_CUST, HSE_KEY_TYPE_RSA_PUB, 10, 4096},
{0, 0, 0, 0, 0}
};
```

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

7.2 Key management

Key management services are available to the host to:

- Initialize and update key values and properties
- Export key values and properties
- Generate and derivate key values
- Establish secret keys in a secure manner

7.1.2 Key storage

7.1.2 Key storage

SYS-IMG contains:

- The structure of the RAM and NVM key catalogs
- The NVM key properties
- The NVM key values, or a pointer to the key values stored in the host memory for the key types HSE_KEY_TYPE_RSA_PUB_EXT and HSE_KEY_TYPE_ECC_PUB_EXT (see below)

Key properties and values are updated within SYS-IMG after successful key provisioning operations.

SYS-IMG is saved in secure NVM (i.e., internal Flash) by the host. SYS-IMG is loaded and authenticated by the HSE at start-up.

Key 表示

A key K can be imported by the host to the HSE in plain or encrypted with an encryption key K_e . It can be further authenticated using an authentication key K_a .

Depending on its type, a key K has either a single value (a secret that *can be shared*) or a pair of values made of a private key (a secret that is *never shared*) and a public key (not a secret).

The secret or private value of a key K is noted k_{sec} . The public key value of K is noted k_{pub} .

The properties of K (usage, access restrictions, etc.) are defined by the host and provided along with the key values.

K_e is used to decrypt k_{sec} when it is provided encrypted by the host. k_{pub} (if any) is always provided in plain.

K_a is used to authenticate a container where k_{sec} and k_{pub} are stored among other information (if any).

7.2.11 Key value verification

```
7: #ifndef HSE_SPT_KEY_VERIFY
8: /** @brief The algorithm used for key verification. */
9: typedef uint8_t hseKeyVerAlgo_t;
0: #define HSE_KEY_VER_SHA256 ((hseKeyVerAlgo_t)HSE_HASH_ALGO_SHA2_256) /**< @brief SHA256 */
1: #define HSE_KEY_VER_SHA384 ((hseKeyVerAlgo_t)HSE_HASH_ALGO_SHA2_384) /**< @brief SHA384 */
2: #define HSE_KEY_VER_SHA512 ((hseKeyVerAlgo_t)HSE_HASH_ALGO_SHA2_512) /**< @brief SHA512 */
3: #define HSE_KEY_VER_CMACE ((hseKeyVerAlgo_t)HSE_MAC_ALGO_CMACE) /**< @brief CMACE (AES) */
4:
```

7.2.3 Key import

6.2.3.4 Key properties

The imported key attributes defined in [Key attributes](#) must be provided with the pointer `pKeyInfo` as listed in the below table.

Table 52. Key attribute mapping in `pKeyInfo`

Key attribute	Data field	Possible values
Key bit size	<code>pKeyInfo→keyBitLen</code>	16-bit integer
Key Type	<code>pKeyInfo→keyType</code>	Key type; see Key group and key type
Update counter	<code>pKeyInfo→keyCounter</code>	28-bit integer
Access restriction flags	<code>pKeyInfo→keyFlags</code>	Binary OR combination of <code>HSE_KF_ACCESS_xxx</code> and <code>HSE_KF_USAGE_xxx</code>
Usage flags		
SMR verification map	<code>pKeyInfo→smrFlags</code>	Binary OR combination of <code>HSE_KF_SMR_xxx</code> enumerates
Curve ID (ECC keys only)	<code>pKeyInfo→specific.eccCurveId</code>	See Notes on ECC keys
Public exponent size (RSA keys only)	<code>pKeyInfo→specific.pubExponentSize</code>	The public exponent size in bytes; see Key values
AES Block Mode Mask (AES keys only)	<code>pKeyInfo→aesBlockModeMask</code>	Bit field to declare the block cipher modes that can be used with the key. If it is cleared to 0, any AES cipher mode can be used. See HSE Service API Reference Manual .

7.2.3.5 Key values

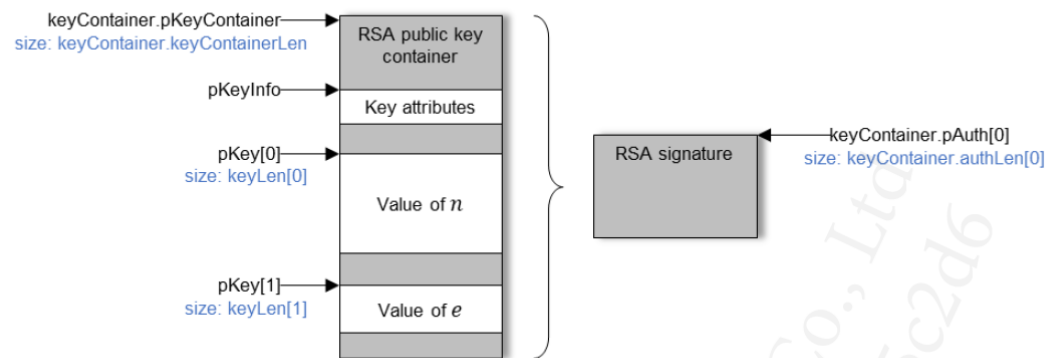
The imported key values are provided via the array of pointers `pKey[]` as listed in the below table.

Table 52: Pointer to provisioning key values vs. key type

Key type	<code>pKey[0]</code>	<code>pKey[1]</code>	<code>pKey[2]</code>
<code>HSE_KEY_TYPE_AES</code>	<i>unused</i>	<i>unused</i>	Value of k
<code>HSE_KEY_TYPE_SHE</code>	Provisioning via SHE services only		
<code>HSE_KEY_TYPE_HMAC</code>	<i>unused</i>	<i>unused</i>	Value of k
<code>HSE_KEY_TYPE_SHARED_SECRET</code>	<i>unused</i>	<i>unused</i>	Value of k
<code>HSE_KEY_TYPE_RSA_PAIR</code>	Value of n	Value of e	Value of d
<code>HSE_KEY_TYPE_RSA_PUB</code>	Value of n	Value of e	<i>Unused</i>
<code>HSE_KEY_TYPE_RSA_PUB_EXT</code>	Value of n	Value of e	<i>Unused</i>
<code>HSE_KEY_TYPE_ECC_PAIR</code>	Value of Q ^[Note]	<i>unused</i>	Value of w
<code>HSE_KEY_TYPE_ECC_PUB</code>	Value of Q ^[Note]	<i>unused</i>	<i>Unused</i>
<code>HSE_KEY_TYPE_ECC_PUB_EXT</code>	Value of Q ^[Note]	<i>unused</i>	<i>Unused</i>
<code>HSE_KEY_TYPE_DH_PAIR</code>	Value of p	Value of T_a	Value of a
<code>HSE_KEY_TYPE_DH_PUB</code>	Value of p	Value of T_a	<i>Unused</i>

[Note] The point coordinates are encoded depending on the curve type (see Table 54)

Figure 35: Illustrating the use of a key container (provisioning a RSA public key)



Key Container import

Table 48: Parameters for an authenticated key import

Data field	Explanation
keyContainer.pKeyContainer	The pointer to the memory container that contains the imported key value(s) and the key properties. The memory location must be accessible by both the host and the HSE.
keyContainer.keyContainerLen	The byte size of the container.
keyContainer.authScheme	The authentication scheme used to authenticate the key container. The proof of authenticity can be an authentication tag (i.e., a Message Authentication Code (MAC)) or a public key signature scheme (i.e., RSA or ECC signature).
keyContainer.authKeyHandle	The key handle to K_a , which must: <ul style="list-style-type: none">- Be declared in a key group owned by the same owner as the target key handle- Refer to a non-empty key slot having its key usage flags HSE_KF_USAGE_KEY_PROVISION and HSE_KF_USAGE_VERIFY set- Refer to a key type that matches with the authentication scheme selected
keyContainer.pAuth[i]	The pointer(s) (i is 0 or 1) to the proof of authenticity, calculated over the key container. The memory location must be accessible by both the host and the HSE. When the proof of authenticity is an authentication tag, pAuth[0] is the pointer to the MAC calculated over the key container and pAuth[1] is not used.

0

50 keyinfo

66

166 key0

166 + 256 + 16= 438 key 1

7.3 Key management: SHE keys

7.3.2 Declaring SHE keys

The AES 128-bit keys specified in [REF40] are declared as HSE_KEY_TYPE_SHE in specific key groups and slots within the key catalogs as described in the below table

Table 63: SHE keys

SHE key name (ID)	Key catalog	Key group index	Key slot index
SECRET_KEY (0x00)	ROM key catalog	N/A	N/A
MASTER_ECU_KEY (0x01)	NVM key catalog	0	0
BOOT_MAC_KEY (0x02)			1
KEY_1 (0x04)			2
KEY_2 (0x05)			3
KEY_3 (0x06)			4
KEY_4 (0x07)			5
KEY_5 (0x08)			6
KEY_6 (0x09)			7
KEY_7 (0x0A)			8
KEY_8 (0x0B)			9
KEY_9 (0x0C)			10
KEY_10 (0x0D)			11
RAM_KEY (0x0E)	RAM key catalog	0	0

BOOT_MAC

The “special” BOOT_MAC value defined in the [REF40] is the CMAC value over the “boot” (and its size) calculated using the key BOOT_MAC_KEY. In the HSE, this “special” key corresponds to the reference authentication tag of the secure memory region (SMR) #0. It is not mapped as a key but can still be updated using the SHE key update protocol.

7.3.6 SHE key provisioning

Table 66: Acronyms used in the SHE key update protocol

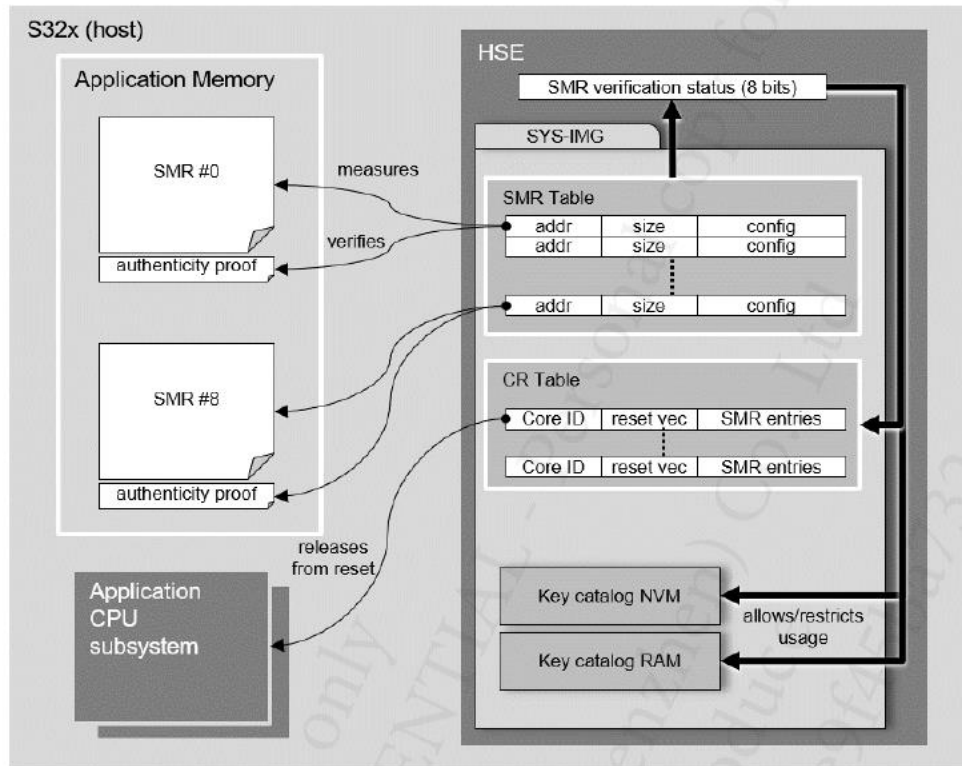
Acronym	Description
ID	Identifier of the key to update K_{ID} (see Table 68)
AuthID	Identifier of the authentication key K_{AuthID} (see Table 68)
K_{ID}'	New key value
K_{AuthID}	Authentication key value
C_{ID}'	New counter value
F_{ID}'	New security flag values, the concatenation WRITE_PROTECTION BOOT_PROTECTION DEBUGGER_PROTECTION KEY_USAGE WILDCARD VERIFY_ONLY
C_{ID}	Current counter value
F_{ID}	Current security flag values
$CMAC_K(M)$	CMAC calculation over M using key K
$ENC_K(M)$	AES CBC-encryption of M using key K (IV = 0)
C_{ENC}	KDF input constant for deriving an encryption key
C_{MAC}	KDF input constant for deriving an authentication key

Table 68: Key ID values in SHE key update protocol

Key to update / Authentication key	ID / AuthID
SECRET_KEY	0x00
MASTER_ECU_KEY	0x01
BOOT_MAC_KEY	0x02
BOOT_MAC	0x03
KEY_1, KEY_11, KEY_21, KEY_31, KEY_41	0x04
KEY_2, KEY_12, KEY_22, KEY_32, KEY_42	0x05
KEY_3, KEY_13, KEY_23, KEY_33, KEY_43	0x06
KEY_4, KEY_14, KEY_24, KEY_34, KEY_44	0x07
KEY_5, KEY_15, KEY_25, KEY_35, KEY_45	0x08
KEY_6, KEY_16, KEY_26, KEY_36, KEY_46	0x09
KEY_7, KEY_17, KEY_27, KEY_37, KEY_47	0x0A
KEY_8, KEY_18, KEY_28, KEY_38, KEY_48	0x0B
KEY_9, KEY_19, KEY_29, KEY_39, KEY_49	0x0C
KEY_10, KEY_20, KEY_30, KEY_40, KEY_50	0x0D
RAM_KEY	0x0E

Chapter 8. Memory Verification Services

Figure 40: Illustrating the memory verification service (SMR)



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

For all SMR that have been defined, the HSE verifies the authenticity of memory contents:

- During the device **start-up phase** (after reset)
- While the application(s) is(are) running on the host side (**during run-time**)

Sanctions:

Unsuccessful verification can **keep** select subsystems on the host side **in reset state**; those subsystems are referenced in the Core Reset (CR) table

- Likewise, failing to verify certain SMR can **render select keys within the HSE unusable**; these restrictions are defined individually for each key via the SMR verification map

memory verification services are made of:

- The SMR **installation** service
- The SMR **verification** service

- The **Core Reset** table installation service

```
#define HSE_SRV_ID_SMR_ENTRY_INSTALL ((hseSrvId_t)(HSE_SRV_VER_0 | 0x00000501UL)) /**< @brief Install a Secure memory region (SMR) table entry. */
#define HSE_SRV_ID_SMR_VERIFY ((hseSrvId_t)(HSE_SRV_VER_0 | 0x00000502UL)) /**< @brief Verify a Secure memory region (SMR) table entry. */
#define HSE_SRV_ID_CORE_RESET_ENTRY_INSTALL ((hseSrvId_t)(HSE_SRV_VER_0 | 0x00000503UL)) /**< @brief Install a Core Reset(CR) table entry. */
#define HSE_SRV_ID_ON_DEMAND_CORE_RESET ((hseSrvId_t)(HSE_SRV_VER_0 | 0x00000504UL)) /**< @brief On demand release a core from reset after loading and verificat
```

The host can define up to **8 SMR clustered** into the SMR table

8.5.4.2 Specific use (SMR #0)

The SMR #0 is the only SMR that can be associated to the SHE AES key **BOOT_MAC_KEY** as the SMR authentication key. In this case, the reference authentication tag is the CMAC value referred to as **BOOT_MAC**.

The **BOOT_MAC** value can be initialized and updated via the SHE key update protocol (see section **SHE key update protocol**).

In addition, when host is granted with SU rights, **BOOT_MAC** can be automatically calculated as described below.

On the first SMR #0 installation using **BOOT_MAC_KEY**, if **BOOT_MAC** is empty (i.e. not initialized) and if **BOOT_MAC_KEY** has been provisioned, the reference authentication tag is calculated by the HSE and saved in **BOOT_MAC**. This specific installation process satisfies the requirement in **[REF40]** referred to as "autonomous bootstrap configuration".

demo_app

- ✓ 6. HSW FW Features
 - > 6.1. HSE Attribute Programming
 - 6.2. HSE Cryptographic Services
 - > 6.3. Secure BAF Update
 - > 6.4. HSE FW Update
 - > 6.5. Secure Boot Demo
 - > 6.6. Monotonic Counter

demo 服 务 范 例 代 码 /
HSE_DEMOAPP_S32K3X4_0_1_1_0\demo_security_installer\src\demo_app\services

子范例:

1. APP_PROGRAM_HSE_ATTRIBUTE

2. APP_RUN_HSE_CRYPTOGRAPHIC_SERVICES 主任务函数 HSE_Crypto()

3. APP_HSE_FW_UPDATE:

1) HSE_FwUpdateExample(gNewHseFwAddress);

2) HSE_ActivatePassiveBlock() abswap

4. APP_SECURE_BOOT_CONFIGURED sbl

Secure Boot Demo

Table 1. Secure boot modes

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

The procedures of configuring these secure boot modes are shown in the [Figure 8](#) below.

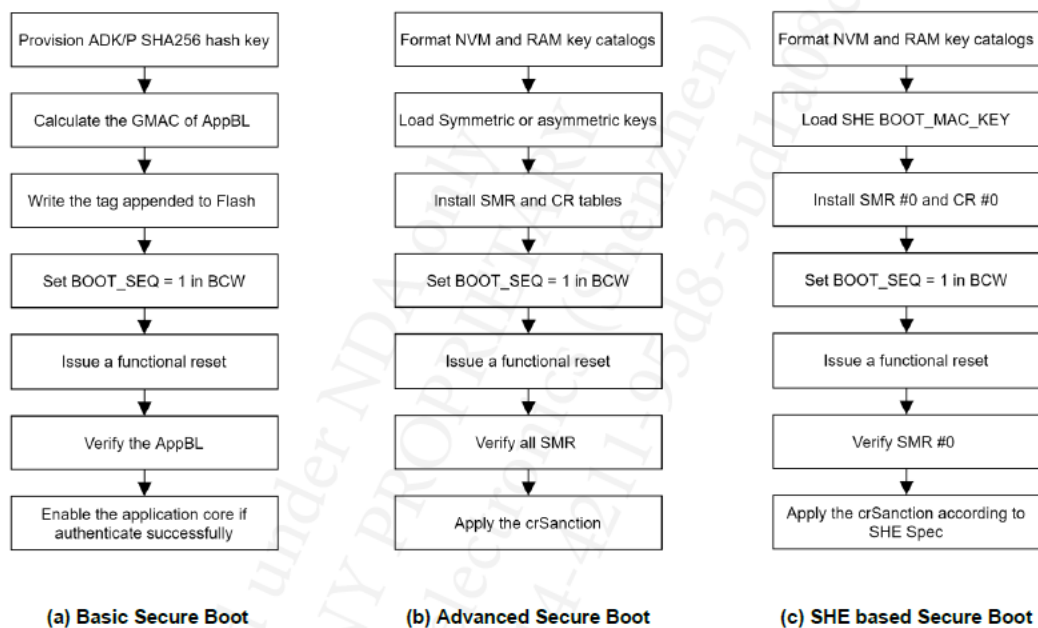


Figure 8. Three Secure Boot modes

2.5. AppBL

The constituent elements of AppBL structure are shown in the [Figure 7](#) below.

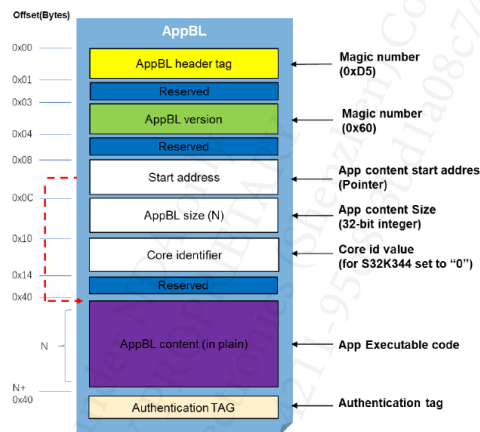
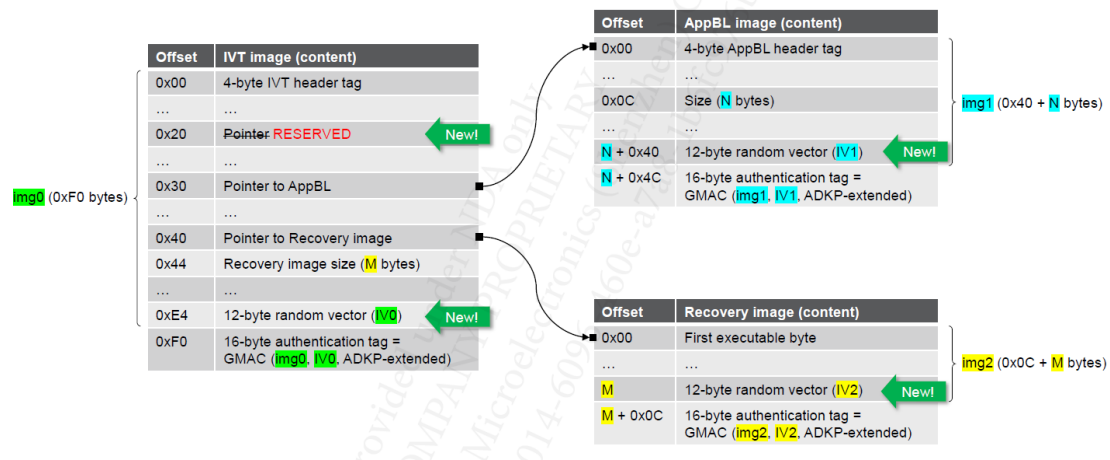


Figure 7. AppBL structure

Fw v210

NEW SYSTEM IMAGE STRUCTURES



Basic Secure Boot (BSB)

Authenticate:

application header : AES-GMAC algorithm

key : ADK/P SHA256 hash key

BSB can boot **only one core** (the booted core can start other cores).

The application image should contain the header that includes all information needed for BSB (i.e. same as in un-secure boot).

The signature should be **appended to the end** of AppBL and generated using a key derived from ADK/P.

Advanced Secure Boot (ASB)

Install and verify **SMR#0 to SMR#4** and CR0 with following options for multiple crypto algorithms in configurable options such as:

- AES CMAC
- AES GMAC
- HMAC
- RSA-PKCS
- ECC

o This demo application can be configured in any cipher combination mentioned above.

o Secure boot application is booted with core 0.

SHE based Secure boot

HSE FW does SHE based boot operation using SMR and CR tables. **Only SMR #0** should be used to implement SHE secure boot.


```
authKeyHandle = NVM_SHE_AES128_BOOT_KEY  
macAlgo      = HSE_MAC_ALGO_CMAC
```

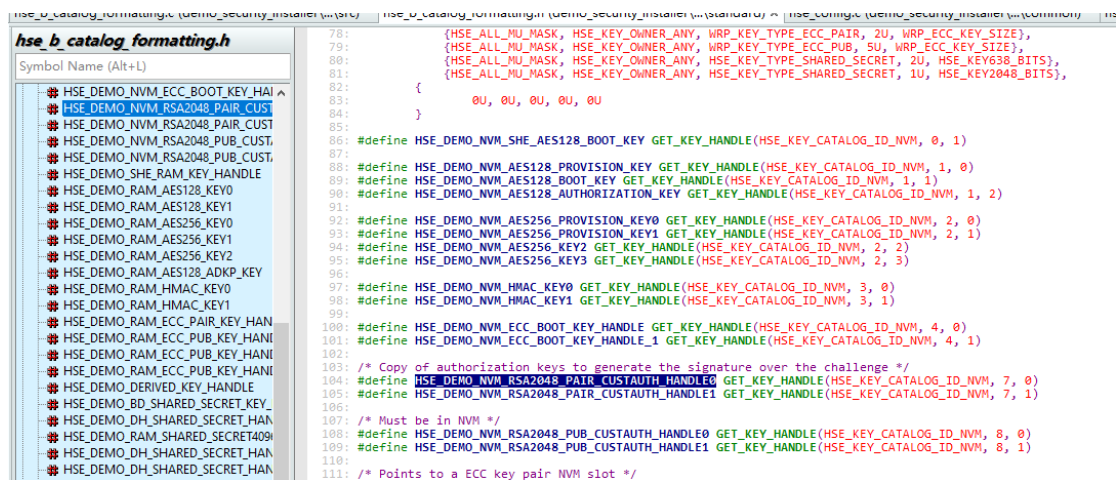
SecureBootTestApp

SHECommandApp

Key formatted v0210

HSE_DEMOAPP_S32K3XX_0_2_1_0 > demo_security_installer > src > demo_app > services > inc > standard

名称	修改日期	类型	大小
 hse_b_catalog_formatting.h	2022/7/20 20:08	H 文件	11 KB



V0110

SE_DEMOAPP_S32K3X4_0_1_1_0 > demo_security_installer > src > demo_app > framework > host_hse > hse_b

名称	修改日期	类型	大小
 hse_default_config.h	2021/10/13 20:25	H 文件	21 KB

Image

HSE FW Pink image	HSE FW image delivered by NXP. This is encrypted and authenticated with keys known by NXP. Authentication (signature generation) is done with an asymmetric algorithm.
HSE FW Image header tag	The first byte of an image header, also known as marker. 0xDA – Full Mem HSE firmware 0xDB – AB SWAP HSE Firmware 0xDC – Encrypted Secure-BAF Image
IVT image	First 256 bytes of data read by HSE firmware after reset. Contains some attributes configured by secure BAF and references to the other images used by secure BAF/HSE FW at boot time.

Table 3. HSE-IMG header for Full Mem HSE firmware configuration

Byte 0	Byte 1	Byte 2	Byte 3
0xDA	0xFF	0xFF	0x60

App = 5AA5 5AA5

Hse 保留资源

secure BAF will reserve
the 176KB area of code flash and
168KB of data flash area for HSE area.
48k ram(??新固件不使用)

RM559614-HSE-B Firmware Reference Manual -V1.2(1.4)

Table 118: 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 119: Secure NVM mapping (AB_SWAP)

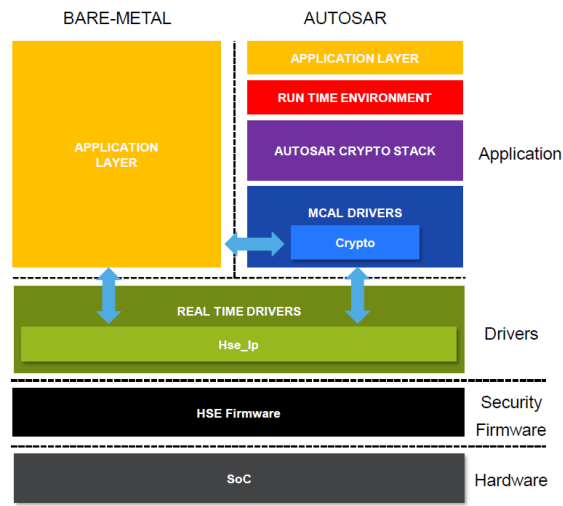
Device	Flash area	Start address	Size
Common	HSE data flash	0x10020000	128KB
	HSE configuration (UTEST)	0x1B000000	8KB
S32K344, S32K324, S32K314	HSE code flash (passive area)	0x007D4000	176KB
	HSE code flash (active area)	0x005D4000	176KB
S32K312, S32K342, S32K322, S32K341	HSE code flash (passive area)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

*The sizes mentioned are only valid for Standard Firmware.

Crypto Driver

S32K3XX_CryptoDriver_and OTA_AdvancedTraining.pdf

SECURITY SOFTWARE LAYERED ARCHITECTURE



• Security Software Drivers

– Crypto Driver

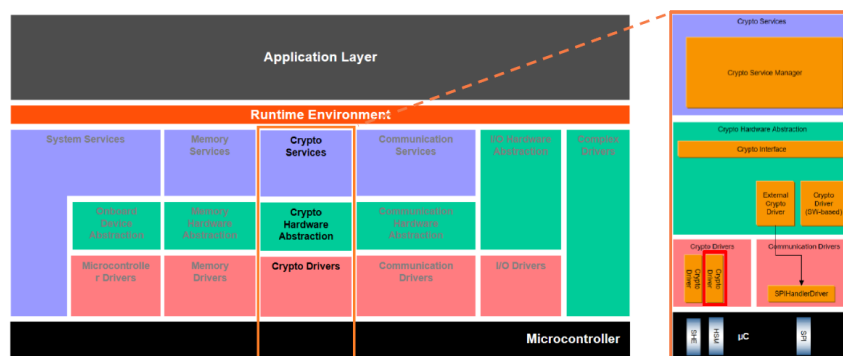
- Crypto Driver Interface for AUTOSAR Crypto Stack

– Hse_Ip Driver

- Bare-Metal Driver for interfacing with Security Firmware

CRYPTO DRIVER AS PART OF THE AUTOSAR STACK

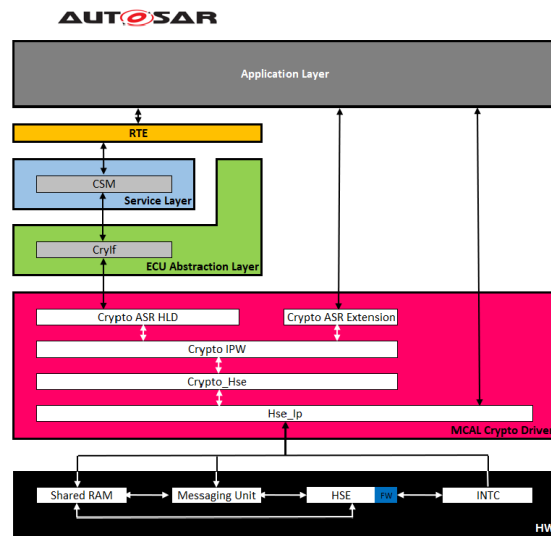
- The Crypto Stack offers standardized access to cryptographic services for applications and system functions.
- The Crypto Driver is a driver for a specific device, abstracting the features supported by the hardware.



NXP CRYPTO STACK STRUCTURE

Crypto Driver is layered as following:

- Crypto Autosar
 - Implements the Autosar APIs
 - Offers Extension APIs to extend the standard APIs
- HSE IP
 - Low level driver that is simple and fast
 - Offers IP APIs to communicate with HSE firmware



CRYPTO DRIVER FEATURES

Features	Standard
Extension Services	SHE Services
AES encryption & decryption	ECB CBC CTR OFB CFB
AES authenticated encryption & decryption	AES-CCM AES-GCM
Hashing	SHA-1 SHA-2 (all digest sizes), Miyagucci Preneel
MAC generation & verification	CMAC FAST_CMAC HMAC GMAC
Signature generation & verification	RSA PKCS-1.5 and PSS ECDSA ^[1] EdDSA ^[2]
RSA encryption & decryption	PKCS-1.5 and OAEP
Key generation	Symmetric AES Keys, RSA & ECC Key Pairs
Key derivation services	PBKDF2 X9.63 TLS1.2
Key Sizes (max key sizes)	AES (up to 256) RSA (up to 4096) HMAC(up to 1152) ECC(up to 521)
Key Import & Export	AES SHE ECC
Shared Secret generation	ECDH
Random Number Generation	DRG.3 DRG.4 PTG.3

^[1] Standard and user configurable Weierstrass & Montgomery curves

^[2] Curve Ed25519, Curve 448

Doc ref

C:\NXP\HSE_DEMOAPP_S32K3X4_0_1_1_0\HSE_DEMOAPP_S32K3X4_0_1_1_0_ReadMe.pdf
安装, demoapp 解释

C:\NXP\HSE_DEMOAPP_S32K3X4_0_1_1_0\demo_security_installer\docs\NXP_SECURE_BAF_
UPDATE_README.pdf

C:\NXP\HSE_FW_S32K3X4_0_1_1_0\docs\HSE Service **API Reference Manual**

HSE Firmware Reference Manual ???

术语和缩写

Authentication: Designed to assure that something is what it claims to be.

Authenticity: Assurance that code is from the source it claims to be.

Digital Signature: An asymmetric key algorithm that associates a
calculated number to both a message and its signer.

Sign/Verify: See Digital Signature

HSE Hardware Security Engine

UID 8bytes in UTest NVM

ADK/P debug key/password (ADK/P) 128bit

ECB Electronic code book

CBC Cipher block chaining

CTR Counter-based block cipher mode

OFB Output feedback based block cipher mode

CFB Cipher feedback mode

Key Provisioning

- Initializes Crypto Driver
- Performs Key Provisioning (Catalog format, loads AES-128 and RSA Key pair (2048bit))
- Performs RSA-2048 Signature Generation and Verification (using RSASSA-PSS & SHA256 hash algorithm)
- Performs AES-128-CTR Encryption and Decryption operation
- Performs Hash generation using SHA256 algorithm
- Performs OTA application update
- After next reset, the new application is executed

基本算法：

1.AES

AES Advanced encryption standard

AES 可以使用大小为 128 Bits、192 Bits 和 256 Bits 的密钥

2. RSA

RSA Rivest–Shamir–Adleman (a public key cryptosystem)

OAEP= Optimal Asymmetric Encryption Padding， 翻译为中文是最优非对称加密填充。

PSS = Probabilistic Signature Scheme， 翻译为中文就是概率签名方案。

RSA 的加密机制有两种方案:一个是 RSAES-OAEP,另一种是 RSAES-PKCS1-v1_5。RSAES 前缀的意思是 **RSA ENCRYPTION SCHEME**。PKCS#1 推荐在新的应用中使用 RSAES-OAEP， 因为其安全性更高， 规范保留对 RSAES-PKCS#1-v1_5 的支持是为了跟老的应用保持兼容。它们两的区别仅仅在于加密前编码的方式不同。而加密前的编码是为了提供了抵抗各种活动的敌对攻击的安全机制。

PKCS#1 的签名机制也有种方案: RSASSA-PSS 和 RSASSA-PKCS1-v1_5。RSASSA 前缀的意思是 **SIGNATURE SCHEMES WITH APPENDIX**。同样， 推荐 RSASSA-PSS 用于新的应用,而 RSASSA-PKCS1-v1_5 只用于兼容老的应用。

3.ECC 椭圆曲线加密法

ECC Elliptic curve cryptography

4. Hashing 哈希

Let's finish up with a side-by-side comparison of SHA1 vs SHA2:

	SHA-1	SHA-2
Years During Which the Algorithm Was Considered the Industry Standard	2011-2015	Since 2016
Other Names	N/A	SHA-256, SHA-256 Bit
How Many Possible Hashing Combinations Does It Have?	2 ¹⁶⁰	2 ²⁵⁶ Possible Combinations
Single Hash or Family of Algorithms Hash Values	Single Hash: 160-bit	Family of Hash Functions – 224, 256, 384, and 512

Hash Algorithm Comparison Table: MD5, SHA-1, SHA-2, SHA-3

Keys for Comparison	MD5	SHA-1	SHA-2 (224 & 256/384 & 512)	SHA-3 (224/256/384/512)
Available Since	1992	1995	2002	2008
Block Size	512 bits	512 bits	512/1024 bits	1152/1088/8 32/576 bits (this is referred to as a Rate [R] for SHA-3 algorithms)
Hash Digest Size (Output)	128 bits (i.e., 16 bytes), or 32 hexadecimal digits	160 bits (i.e., 20 bytes), or 40 hexadecimal digits	256 bits (i.e., 32 bytes), or 64 hexadecimal digits/512 bits (i.e., 64 bytes), or 128 hexadecimal digits	224/256/384/512 bits (i.e., 28/32/48/64 bytes), or 56/64/96/128 hexadecimal digits
Rounds of Operations	64	80 (4 groups of 20 rounds)	64 (for SHA-224 and SHA-256)/80 (SHA0384/SHA-512)	24
Construction	Merkle-Damgård	Merkle-Damgård	Merkle-Damgård	Sponge (Keccak)
Collision Level	High — They can be found in seconds, even using an ordinary home computer.	Cheap and easy to find as demonstrated by a 2019 study .	Low — No known collisions found to date.	Low
Successful Attacks	Many. Researchers showed concrete evidence in 2004.	Yes, many. The first one called SHAttered happened in 2017.	SHA-256 has never been broken.	Few collision type attacks have been demonstrated.
Common Weaknesses	Vulnerable to collisions.	Vulnerable to collisions.	Susceptible to preimage attacks.	Susceptible to: ✓ Practical collision. ✓ Near collision attacks.
Security Level	Low	Low	High	High
ApplicationsApplications	Previously used for data encryption, MD5 is now mostly used for verifying the integrity of files against involuntary corruption.	Previously widely used in TLS and SSL. Still used for HMAC (even if it's recommended to move to a more secure algorithm), and for verifying the integrity of files against involuntary corruption.	Widely used in: ✓ Security applications and protocols (e.g., TLS, SSL, PGP, SSH, S/MIME, IPsec) ✓ Cryptocurrencies transactions validation ✓ Digital certificates ✓ Other applications	Used to replace SHA-2 when necessary (in specific circumstances).
Deprecated?	Yes	Yes	No	No

Miyaguchi-Preneel compression

using AES-ECB with 128-bit key size (SHE spec support).

MAC message authentication codes (MAC)

CMAC Cipher-based message authentication code ---AES

GMAC Galois message authentication code ---AES

HMAC Keyed-hash message authentication code

ECDSA Elliptic curve digital signature algorithm

EdDSA Edwards-curve digital signature algorithm

PKCS1 Public-key cryptography standards. PKCS provides the basic definitions of, and recommendations for implementing the RSA algorithm.

IPSec 互联网安全协议 (Internet Protocol Security, IPSec)

PKCS Public-Key Cryptography Standards (PKCS)

ECIES Elliptic Curve Integrated Encryption Scheme (ECIES)

[X.209-88] CCITT Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1988.

[P1v1.5] Kaliski, B., "PKCS #1: RSA Encryption Version 1.5", RFC 2313, March 1998.

Optimal Asymmetric Encryption Padding (OAEP)

AEAD Authenticated Encryption with Associated Data

Authenticated block ciphering (AEAD)

Algorithms :

CCM, Counter with CBC-MAC (CCM)

GCM. GCM 全称为 Galois/Counter Mode

Root of Trust: Secure foundation (hardware, software, firmware) of a system that cannot be tampered with by malware.

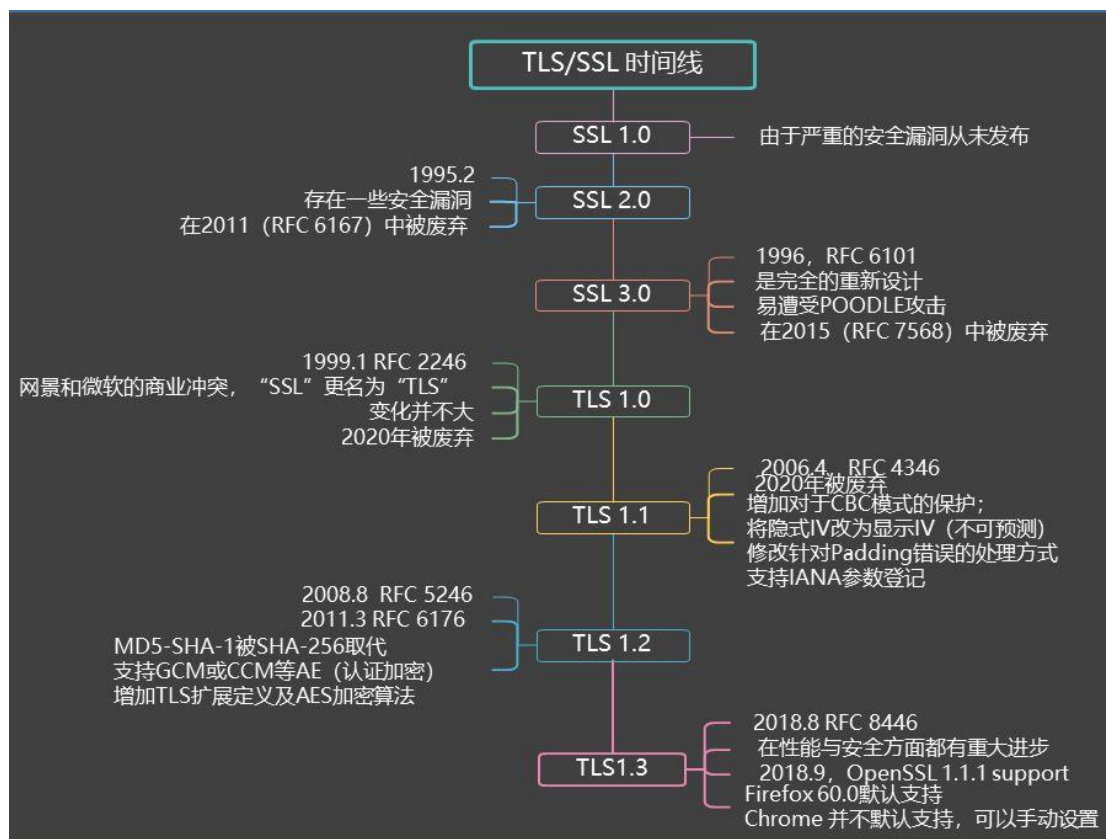
secure memory regions (SMR)

Core Reset (CR)

CSEc vs. HSE – Cryptographic Services		
Secure Subsystem	CSEc	HSE
AES encryption & decryption	ECB CBC	ECB CBC CTR OFB CFB XTS
AES authenticated encryption & decryption	N/A	CCM GCM
Hashing	Miyaguchi-Preneel compression function	SHA-1 SHA-2 (all digest sizes) SHA-3 (all digest sizes) Miyaguchi-Preneel compression function
xMAC generation & verification	CMAC	CMAC XCBC-MAC HMAC GMAC
Signature generation & verification	N/A	RSA PKCS-1.5 + PSS ECDSA ^[1] EdDSA ^[2]
RSA encryption & decryption	N/A	PKCS-1.5 + OAEP
ECC encryption & decryption	N/A	ECIES
Random number generation	TRNG	TRNG AIS-31 Class P2 High + PRNG

TLS

TLS 前身是网景（Netscape）公司开发的 SSL 规范，后来在协议维护移交给 IETF 之后，将其更名为 TLS，



标准及组织

the Secure Hardware Extension (**SHE**) specification developed by Escript for Audi and BMW via the HIS Working Group, with early cooperation from Freescale in 2008, has now been accepted as an open and free standard.

EVITA defines the overall functionality of three different hardware security module approaches: – full, medium and light.

ARM[®] developed its TrustZone[®] security infrastructure

Trusted Computing Group (TCG), which claims to provide

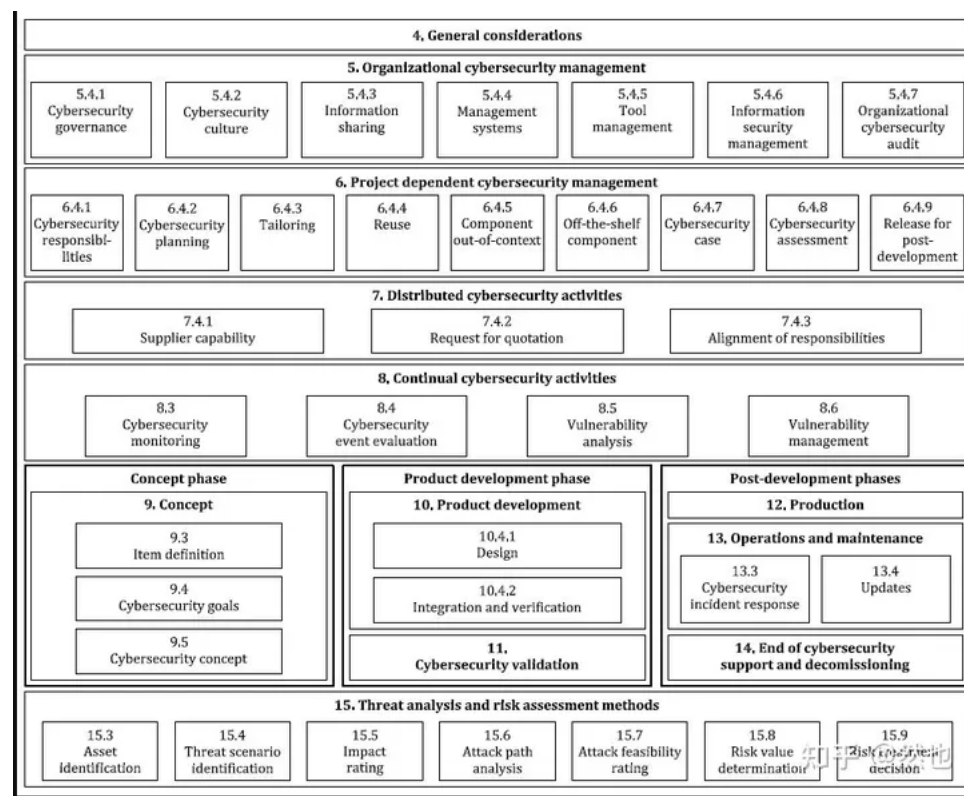
open, interoperable and international standards for trusted computing. One specification

released by this organization is their Trusted Platform Module (TPM)—published as ISO/IEC

11889 Parts 1-4.

ISO/SAE 21434

ISO/SAE 21434:2021 Road vehicles — Cybersecurity engineering



UN R155 Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system

UN R156 Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system

Internet Engineering Task Force (IETF)

<https://www.rfc-editor.org/rfc/rfc8017>

This document represents a republication of PKCS #1 v2.2 from RSA Laboratories' Public-Key Cryptography Standards (PKCS) series. By publishing this RFC, change control is transferred to the IETF.

Unify bootloader

通信設定

```
#define RX_FUN_ID (0x7FFu)    /* can tp rx function ID */
#define RX_PHY_ID (0x784u)    /* can tp rx phy ID */
#define TX_ID (0x7F0u)        /* can tp tx ID */
```

固件更新调试记录

s32k3x4_hse_fw_1.5.0_1.1.0_pb211004.bin.pink

s32k3x4_Secure_Baf_0.5.0_0.9.4_pb210708.bin.pink

(x)= Variables Breakpoints Expressions

Expression	Type	Value	Add
gHseFwVersion	hseAttrFwVersi...	{...}	0x20
reserved	uint8_t	0x1	0x20
socTypeId	uint8_t	0x5	0x20
fwTypeId	uint16_t	0x0	0x20
majorVersion	uint8_t	0x1	0x20
minorVersion	uint8_t	0x1	0x20
patchVersion	uint16_t	0x0	0x20
Add new expression			

Problems Console Search Debugger Console Memory Call Hierarchy

Monitors

0x4039c020 : 0x4039C020 <Hex> New Renderings...

Address	0 - 3	4 - 7	8 - B	C - F
4039C020	01050000	00090400	01000000	01000000
4039C030	00000000	35000000	00000040	00C00240
4039C040	00C00240	00000240	00000000	00000000

更新后

s32k3x4_hse_fw_1.5.0_2.1.0_pb220625.bin.pink

s32k3x4_Secure_Baf_0.5.0_0.10.0_pb220428.bin.pink

1. 升级 hse fw g_isEnableActivePassiveBlock =1,立即 swap active/passive block

Expression

Type

Value

gHseFwVersion

hseAttrFwVersi...

{...}

reserved

uint8_t

0x1

socTypeId

uint8_t

0x5

fwTypeId

uint16_t

0x0

majorVersion

uint8_t

0x2

minorVersion

uint8_t

0x1

patchVersion

uint16_t

0x0

hseSrvResponse

hseSrvRespons...

0xdeadbee

g_isEnableActivePassiveBlo

uint8_t

0x0

Add new expression

Problems Console Search Debugger Console Memory Call Hierarchy

Monitors

0x5d4000
0x400000
0x500000
0x4039c020

0x4039c020 : 0x4039C020 <Hex>

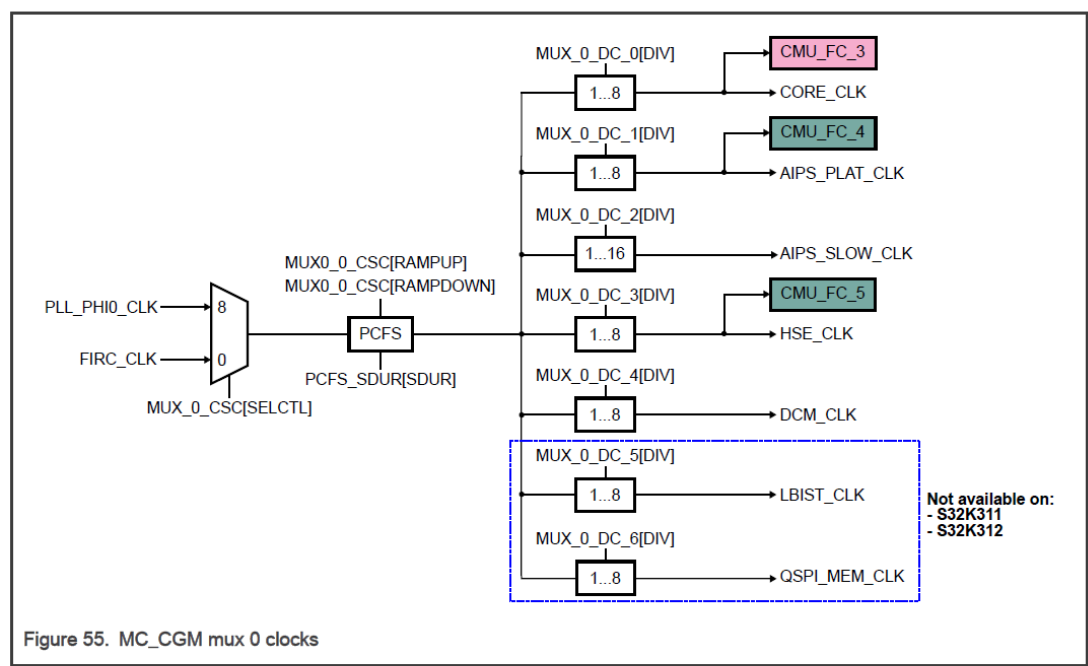
New Renderings...

Address	0 - 3	4 - 7	8 - B	C - F
4039C020	01050000	000A0003	C1000000	01000000
4039C030	00000000	35000000	00000040	00C00240
4039C040	00C00240	00000240	00000000	00000000
4039C050	00000000	FFFFFFBF	FF3FFDBF	FF3FFDBF

问题点

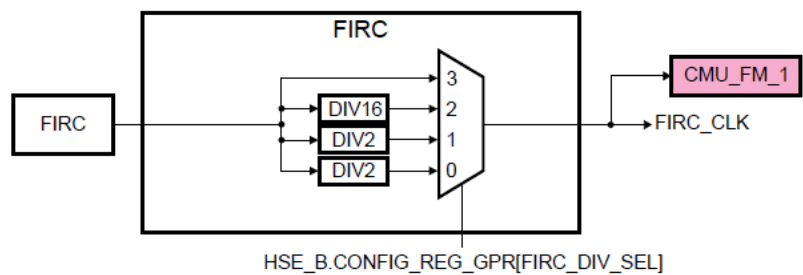
1. hse fw 使能后，， clock 的初始化， hardfault

23.3.2 MC_CGM mux 0 clocks



NOTE

The clock frequency relationship between TCK and HSE_CLK clocks for HSE_B must be a minimum ratio of 1:1.5. For example, if HSE_CLK equals 80 MHz, then TCK must be less than or equal to 53 MHz (80 MHz ÷ 1.5).



1. The FIRC_DIV_SEL is configured by the **sBAF code**. 35.6.13

HSE Configuration REG_GPR (CONFIG_REG_GPR)

Field	Function
31-29 APP_CORE_ACC	FIRC Divider <div>NOTE</div> <div>While writing to this register, APP_CORE_ACC is RO and should not be changed from 0b101.</div> <div>101b - Application core can write this field [FIRC_DIV_SEL]</div> <div>All other values - No access to application core</div>
28-2 —	Reserved
1-0 FIRC_DIV_SEL	FIRC Divider Indicates this chip's FIRC clock division factor. 00b - Divided by 2 01b - Divided by 2 10b - Divided by 16 11b - Undivided

23.6.1.5.1 HSE_B clocking

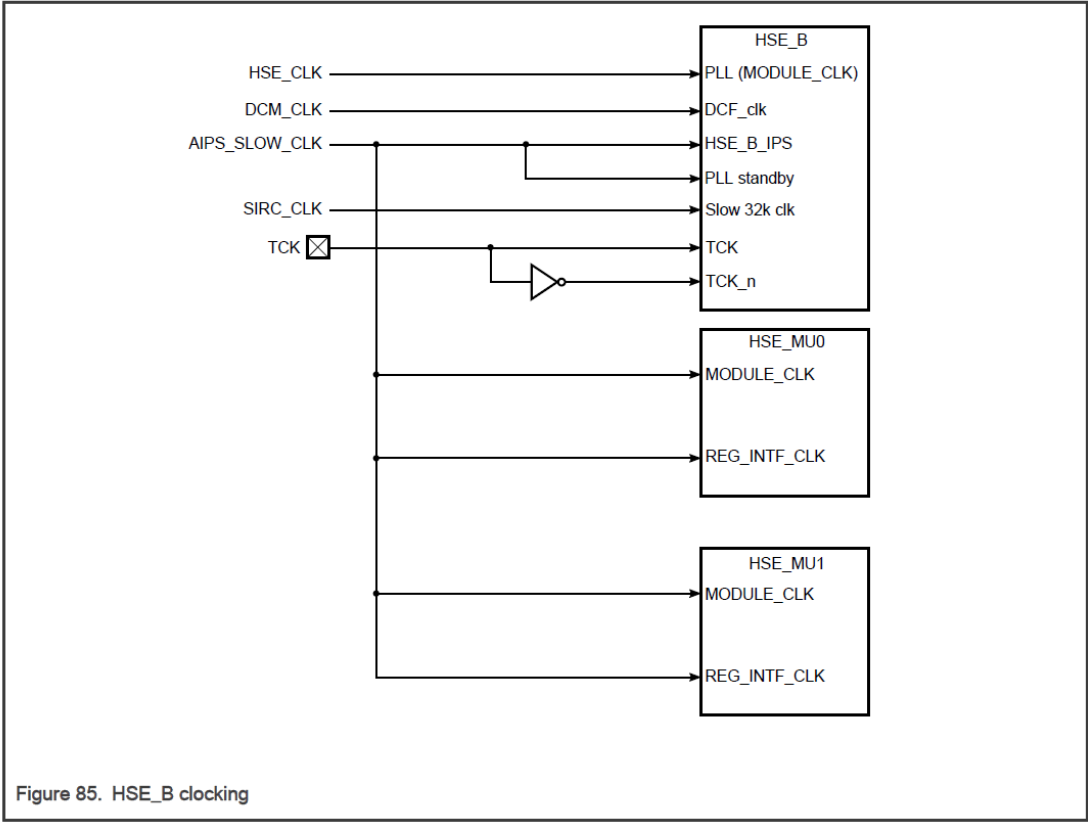


Figure 85. HSE_B clocking

disable 生成代码

```
C Compare Viewer
最新生成: Clock_Ip_Cfg.c
在磁盘上: generate\src\Clock_Ip_Cfg.c

const Clock_Ip_ClockConfigType Clock_Ip_aClockConfig[1U] = {
    /*! @brief User Configuration structure clock_Cfg_0 */
    {
        0U, /* clkConfigId */
        (NULL_PTR), /* Register data if register value c
        1U, /* ircoscsCount */
        2U, /* xoscsCount */
        1U, /* pllsCount */
        13U, /* selectorsCount */
        21U, /* dividersCount */
        1U, /* dividerTriggersCount */
        0U, /* fracDivsCount */
        2U, /* extClksCount */
        101U, /* gatesCount */
        0U, /* pcfsCount */
        4U, /* cmusCount */
        6U, /* configureFrequenciesCount */
    }
};

const Clock_Ip_ClockConfigType Clock_Ip_aClockConfig[1U] = {
    /*! @brief User Configuration structure clock_Cfg_0 */
    {
        0U, /* clkConfigId */
        (NULL_PTR), /* Register data if register va
        2U, /* ircoscsCount */
        2U, /* xoscsCount */
        1U, /* pllsCount */
        13U, /* selectorsCount */
        22U, /* dividersCount */
        1U, /* dividerTriggersCount */
        0U, /* fracDivsCount */
        2U, /* extClksCount */
        101U, /* gatesCount */
        0U, /* pcfsCount */
        4U, /* cmusCount */
        6U, /* configureFrequenciesCount */
    }
};
```

```

/* IRCOSC initialization. */
{
    #if CLOCK_IP_IRCOSCS_NO > 0U
    {
        SIRC_STANDBY_CLK, /* name */
        0U, /* Disabled in standby mode. */
        0U, /* Enable regulator */
        0U, /* Ircosc range */
        0U, /* Ircosc enable in VLP mode */
        0U, /* Ircosc enable in STOP mode */
    },
    #endif
    {
        RESERVED_CLK, /* name */
        0U, /* enable */
        0U, /* Enable regulator */
        0U, /* Ircosc range */
        0U, /* Ircosc enable in VLP mode */
    },
}

/* DIVIDER initialization. */
{
    #if CLOCK_IP_DIVIDERS_NO > 0U
    {
        PLL_POSTDIV_CLK, /* name */
        2U, /* value */
        {
            0U,
        },
    },
    #endif

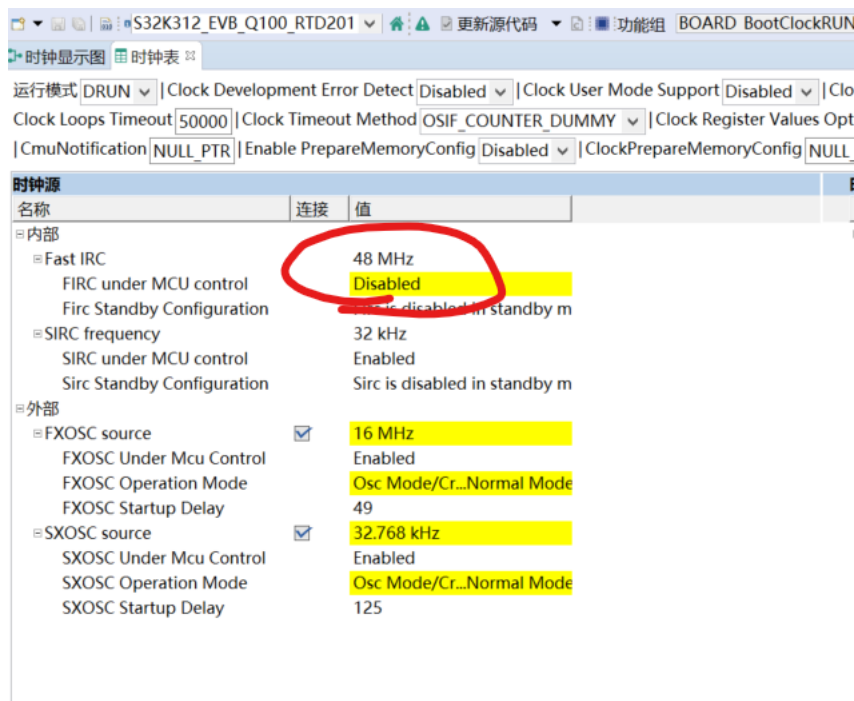
    #if CLOCK_IP_DIVIDERS_NO > 1U
    {
        PLL_PHI0_CLK, /* name */
        3U, /* value */
    },
}

/* IRCOSC initialization. */
{
    #if CLOCK_IP_IRCOSCS_NO > 0U
    {
        FIRC_STANDBY_CLK, /* name */
        0U, /* Disabled in standby */
        0U, /* Enable regulator */
        0U, /* Ircosc range */
        0U, /* Ircosc enable in VL */
        0U, /* Ircosc enable in ST */
    },
    #endif
    #if CLOCK_IP_IRCOSCS_NO > 1U
    {
        SIRC_STANDBY_CLK, /* name */
        0U, /* Disabled in standby */
        0U, /* Enable regulator */
    },
    #endif
}

/* DIVIDER initialization. */
{
    #if CLOCK_IP_DIVIDERS_NO > 0U
    {
        FIRC_POSTDIV_CLK, /* name */
        1U, /* value */
        {
            0U,
        },
    },
    #endif

    #if CLOCK_IP_DIVIDERS_NO > 1U
    {
        PLL_POSTDIV_CLK, /* name */
        2U, /* value */
    },
}

```



S312 CLOCK

<https://community.nxp.com/t5/S32K/HSE-PLL-configuration/m-p/1672534>

a month ago • 236 Views



lukaszdrapa
NXP TechSupport



Hi @DGB

I just got new information. This configuration of HSE_CLK_MODE in DCF record (and also configuration of FXOSC) has no effect unless PLL_ENABLE in IVT is enabled. SBAF will ignore it and FIRC will be used at boot time. And PLL_ENABLE has effect only when BOOT_SEQ is 1 (i.e. secure boot enabled).

I'm sorry, I provided wrong information in previous post. The documentation is little bit confusing here, I will ask to clarify it.

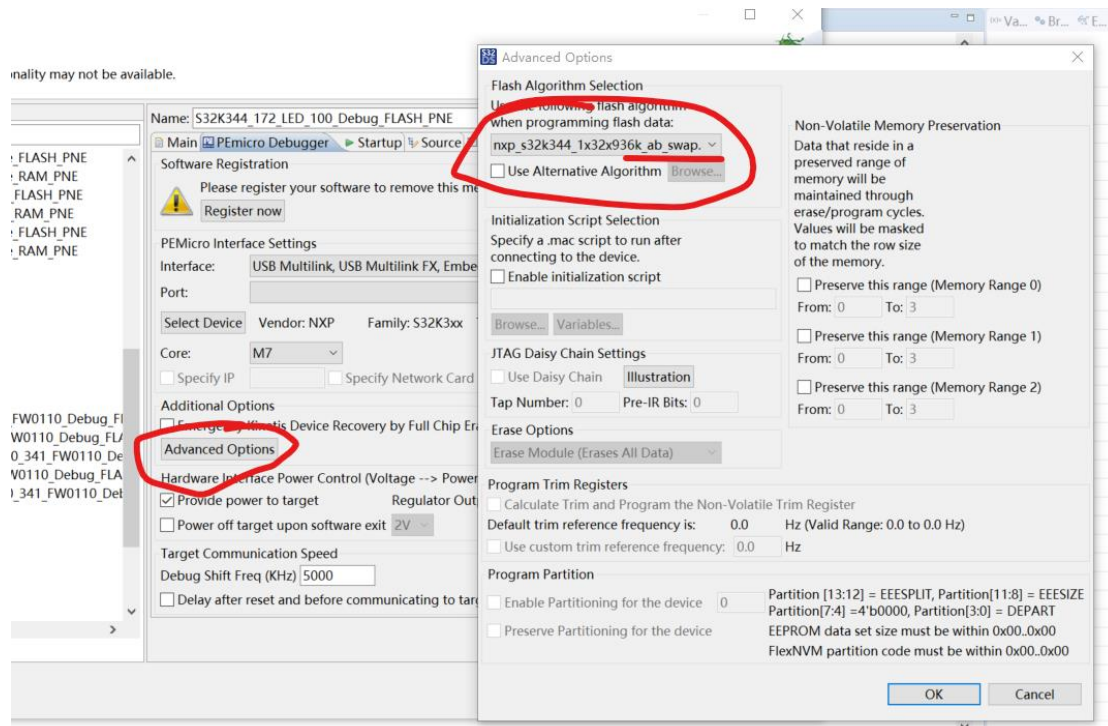
Based on HSE_CLK_MODE_OPTION (if PLL_ENABLE is set), SBAF configures MUX_0_DC_1 and MUX_0_DC_2 dividers. If PLL_ENABLE is not used, it's up to user to configure these dividers later in the code. So, we are getting back to Table 144 in the S32K3 RM. Could you send me your clock configuration? Is it the same as in Table 144?

Regards,

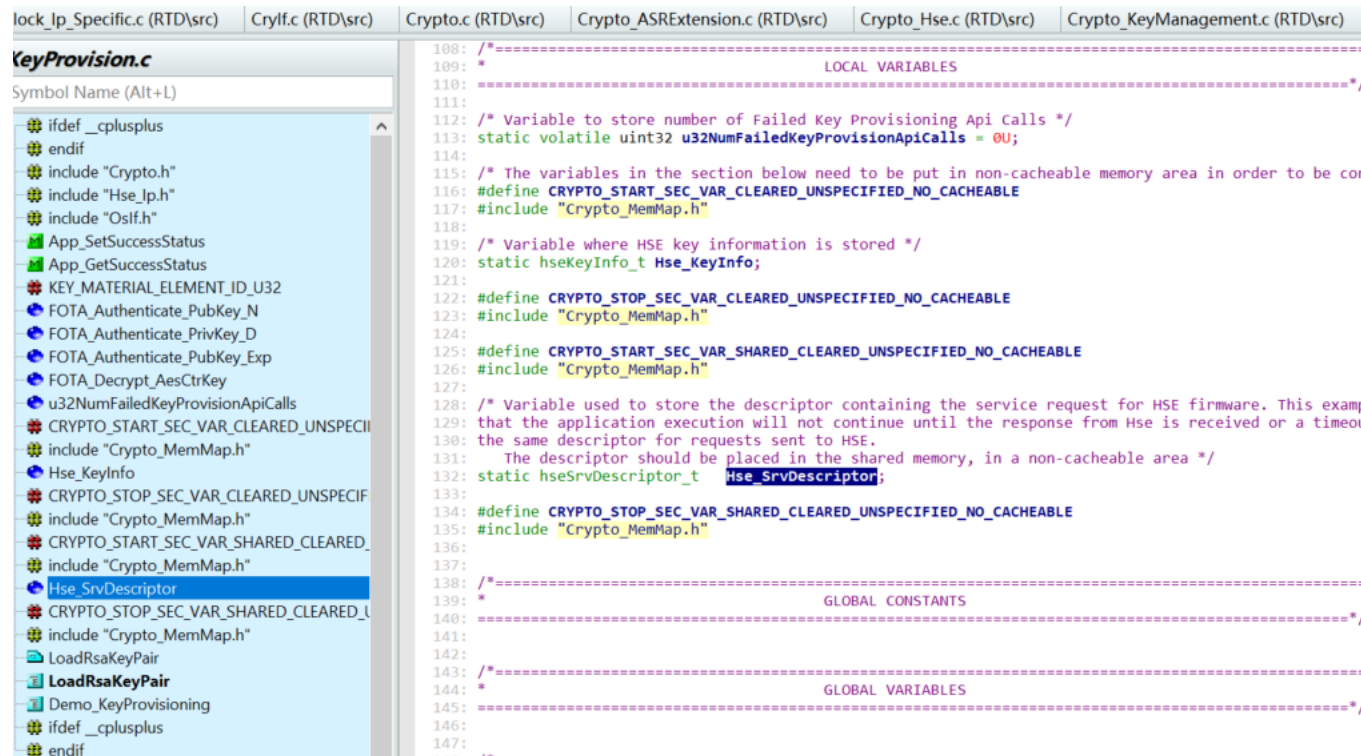
Lukas

2. HSE 使能后，PE multilink 不能擦 application 和调试

根据安装的固件 full mem 或 ab_swap 选择对应的 FLASH algorithm



3. hse 通信用到的发送及接收的变量都要放 non-cache 区域，如下图



详细参考附件简单翻转 demo S32K344_HSE_OTA.7z。

4. Hse 的 demo 都是使用 dtcm memory

地址是 0x21001000 这是出于什么考虑？

,,, no cache 快

5. Hse 的 demo 都没有看到设置 ivt, 代码 rom

从 0x00402000 开始, 但是能进 app? ,,, 中断向量表都是 402000? ?

IVT BIN,,, T32 刷进

6. hse 保留的 data flash ， 应用不可见？

已安装的 smr ， CR Entry ，， 如何清掉？

调用 erase HSE NVM 服务 HSE_SRV_ID_ERASE_HSE_NVM_DATA

7. 高地址 IVT 有效， fw 固件不能安装，

两种方式都是可以的。

第一种方法 把 HSE 的固件放在 0x400000 的位置，是否还需要其他的操作
不需要，， 上电复位 sbaf 识别后安装

第二 方法， IVT

注意 HSE_FW_ADDR 要跟 link file 指定 HSE_BINARY 地址对应， 上电复位 sbaf 识别后安装

装不上，， 可能是客户 调试时 （PE mutilink 只擦下载目标文件用到区域）在 0x500000 放了有效 ivt，， 导致 sbaf 不识别 0x400000

详见

请把 312 可能 ivt 位置 0x500000 ，， 0x10000000 中前 256 byte 擦掉。

https://community.nxp.com/t5/S32K/S32K312-The-HSE-FW-fails-to-be-installed/m-p/1659214/emcs_t/S2h8ZW1haWx8bWVudGlvbl9zdWJzY3JpcHRpb258TEk4V1RIQINDQzBQWDF8MTY1OTIxNHxBVF9NRU5USU9OU3xoSw#M23369