

# S32G2 BOOT

APRIL 2021



SECURE CONNECTIONS  
FOR A SMARTER WORLD

PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.  
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.



## AGENDA

- High Level Boot Overview
- Detailed Overview - Security Installation, Secure Booting
- OTA Updates

# High Level Overview

---



SECURE CONNECTIONS  
FOR A SMARTER WORLD

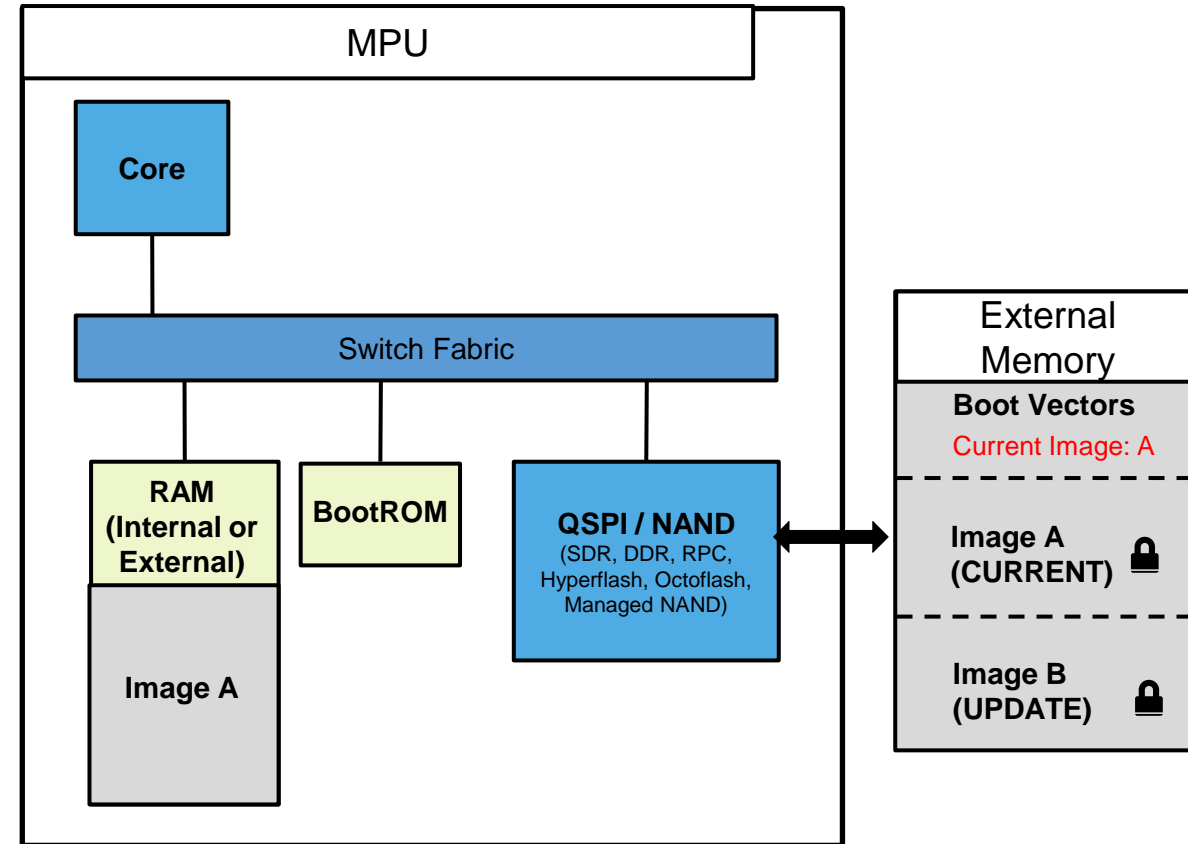
PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.  
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.



# BOOT

- The S32G2 copies code from external flash to internal or external RAM during the boot process.
- This initial copying is carried out by code stored in an internal BootROM.
- The following external flash interfaces are supported:
  - QuadSPI (including DDR octal flash)
  - uSDHC (SD, eSD, MMC, eMMC)



# BOOT SELECTION: TARGET, INTERFACES AND MODES

## Selection:

Boot Pins (BMODE1 & BMODE2)  
FUSE\_SEL Fuse  
Life-Cycle

FUSEs  
RCON  
Serial RCON

IVT Config Data

### Boot Modes

- Pre-test mode (Only when LC == OUT\_OF\_FAB)
- Boot Devices from **Fuses**
- Boot Devices from **RCON** or **Serial RCON (I2C EEPROM)**
- Serial Boot
- Test Mode
- Delayed Reset (only when LC == FA)

### Boot Interfaces

- QSPI Flash
- SD/eSD/MMC/eMMC
- CAN
- UART
- Ethernet (RMII/RGMII/SGMII)

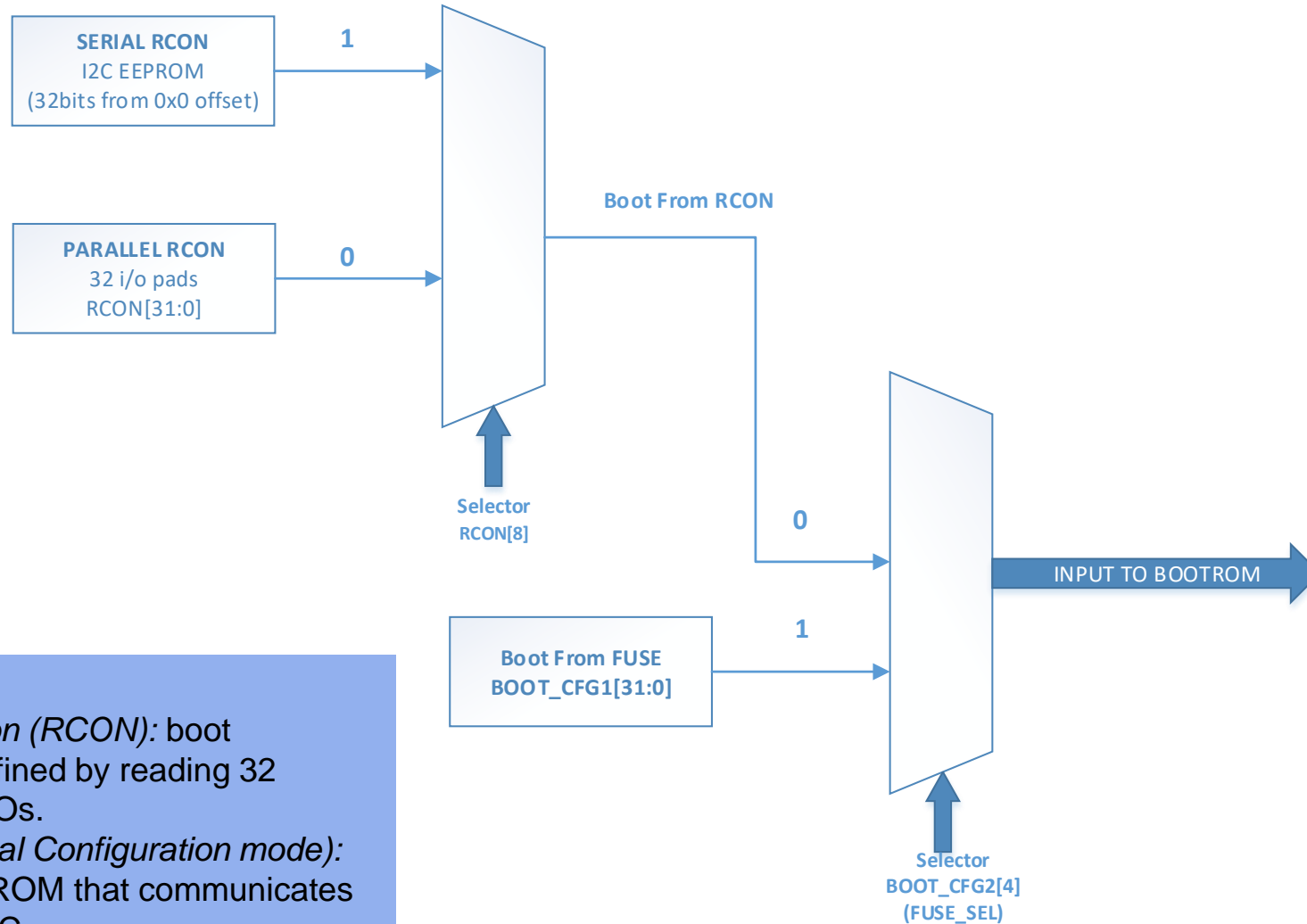
### Boot Target

- M7\_0
- A53\_0

## Notes:

- The only CPU that comes out of reset based on hardware is **HSE M7 aka Boot CPU**
- BootROM code runs on Boot CPU

## PARALLEL RCON/SERIAL RCON:



### Boot from RCON

- *Reset Configuration (RCON)*: boot configuration is defined by reading 32 general purpose I/Os.
- *Serial RCON (Serial Configuration mode)*: uses a serial EEPROM that communicates with the chip via I2C.

(Atmel AT24C01C /  
OnSemiconductor CAT24C01)

# EXTERNAL MEMORY CONTENTS

External  
Memory

IVT

DCD (primary)

HSE  
Firmware  
(primary)

Customer  
Bootloader  
(primary)

Customer  
Application  
(Version A)

Address	Size (Bytes)	Name	Comments
0h	4	IVT header	Header showing the start of the IVT.
4h	4	Reserved 1	
8h	4	Self-Test DCD Pointer	Pointer to the start of the configuration data used for BIST.
Ch	4	Self-Test DCD Pointer (backup)	Pointer to the start of the backup configuration data used for BIST.
10h	4	DCD Pointer	Pointer to the start of DCD configuration data.
14h	4	DCD Pointer (backup)	Pointer to the start of backup DCD configuration data.
18h	4	HSE Firmware Flash Start Pointer	Pointer to the start of the HSE firmware in flash memory.
1Ch	4	HSE Firmware Flash Start Pointer (backup)	Pointer to the start of the backup HSE firmware in flash memory.
20h	4	Application Boot Code Flash Start Pointer	Pointer to the start of the application boot code in flash memory.
24h	4	Application Boot Code Flash Start Pointer (backup)	Pointer to the start of the backup application boot code in flash memory.
28h	4	Boot Configuration Word	Configuration data used to select the boot configuration. This includes bit(s) to indicate if HSE firmware is encrypted.
2Ch	4	Life-Cycle Configuration Word	Configuration data used for advancing Life-Cycle
Reserved			
34h	32	Reserved for HSE FW	To be defined by HSE f/w specification.
54h	156	Reserved2	
F0h	16	GMAC	Cryptographic Hash of data from IVT Header till end of Reserved2 section.

## IVT:

Vector table containing start address in flash for the code and data required during boot.

The IVT is stored at a fixed location in flash.

GMAC is generated using “derived key” – this is device specific and is autogenerated based on parameters of device.

“Application Boot Code Flash” == customer bootloader

# EXTERNAL MEMORY CONTENTS



## DCD:

Device Configuration Data. List of commands used to by BootROM to configure memory interfaces, DRAM etc.

## HSE Firmware:

NXP provided firmware for the security engine

## Customer Bootloader:

First customer code to be executed on the device.

Any further application loading is controlled by the user.

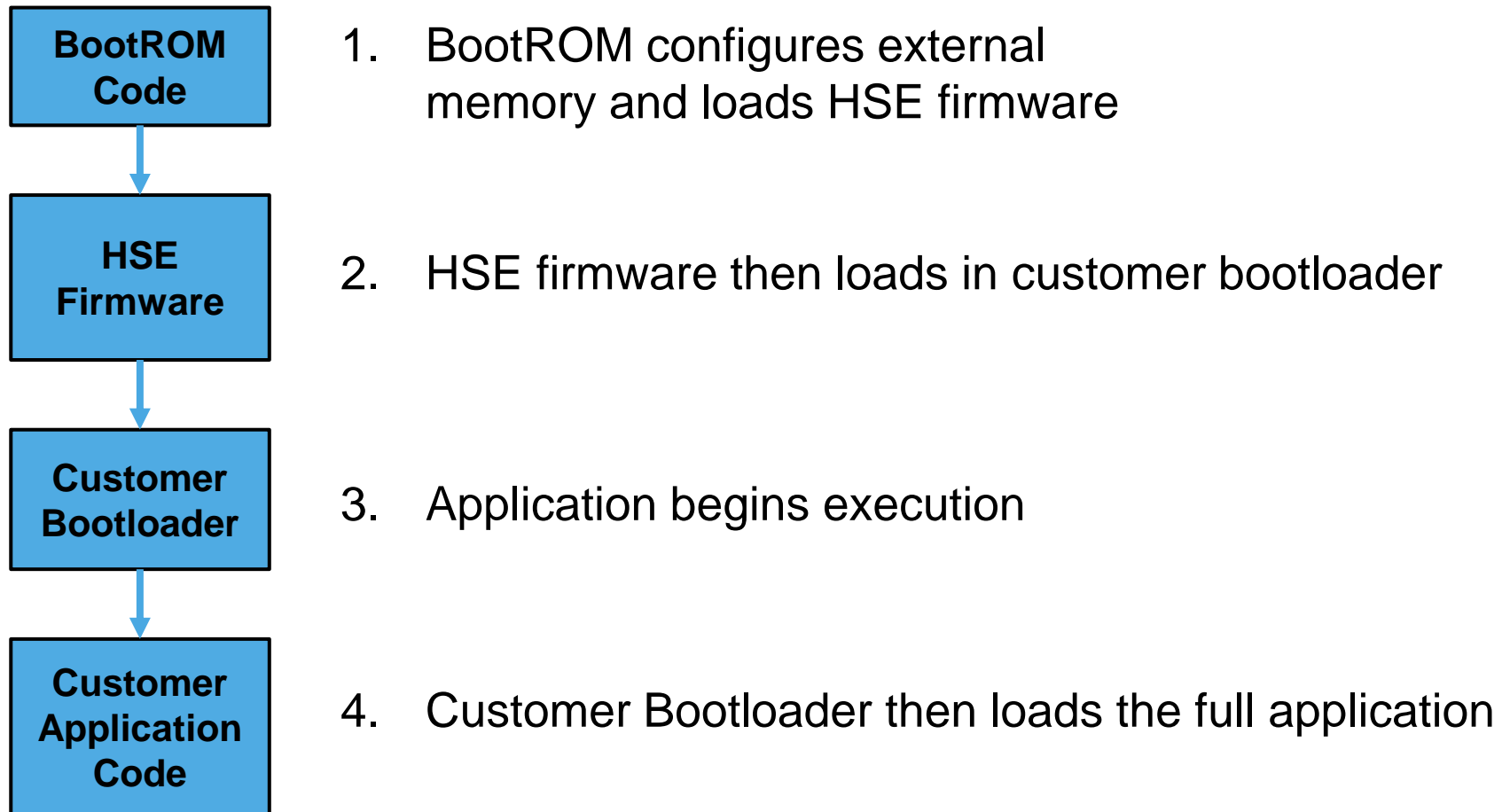
Loads over the main application – possibly in segments to allow faster start of execution

- E.g., load over CAN stack first

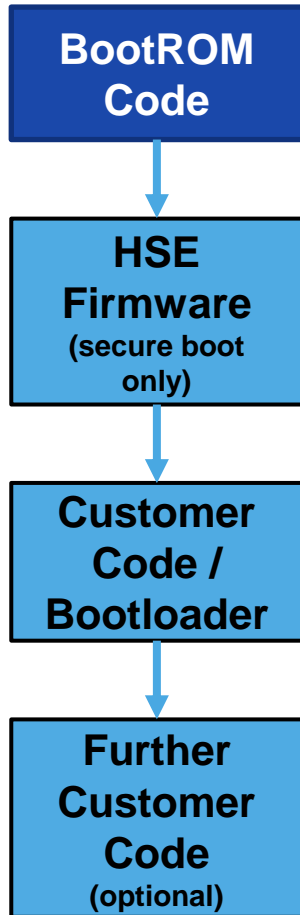


## BOOT FLOW

- The full boot flow multiple software bootloaders – this gives a great deal of scope for customisation to ensure that boot process is optimal for specific use cases.
- From a high level, the flow is as follows:



## BOOTFLOW (1/3)



### BootROM Code:

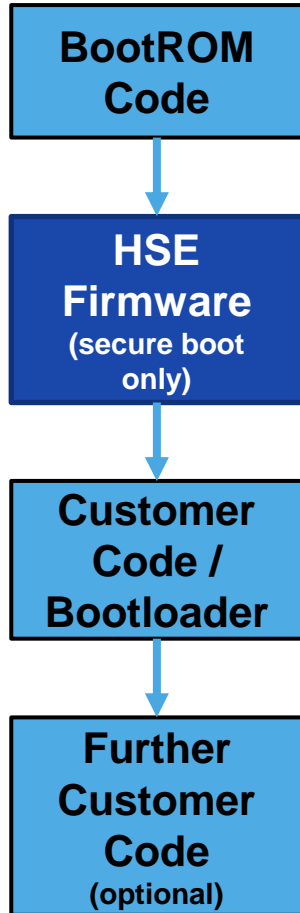
**Executes on:** HSE M7 Core

**Code Location:** Internal ROM on device

### **Task:**

- Setup external memory interface (QuadSPI or uSDHC (SD or MMC))
- Copy IVT and DCD from memory to internal RAM
- Authenticate IVT and DCD (secure boot only)
- Execute DCD instructions
- Proceed based on IVT secure boot parameter:
  - Non-secure boot:
    - Load the customer code pointed to by IVT
    - Return everything to default state
    - Hand control to customer code
  - Secure boot:
    - Load the HSE firmware pointed to by IVT
    - Decrypt and Authenticate the HSE firmware
    - Hand control to HSE firmware

## BOOTFLOW (2/3)



### HSE Firmware:

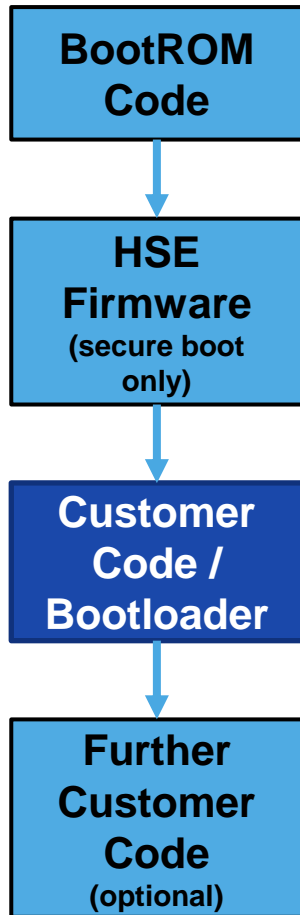
**Executes on:** HSE M7 Core

**Code Location:** Stored in external flash, executed from internal HSE RAM

### **Task:**

- Load the Customer Code/Bootloader pointed to by IVT to system SRAM
- Decrypt and authenticate the Customer Code (optional)
- Setup the M7 or A53 based on IVT
- Hand control to Customer Code executing on selected core

## BOOTFLOW (3/3)



### Customer Code/Bootloader:

**Executes on:** M7\_0 Core or A53 core (R52 on S32S)

**Code Location:** Stored in external flash, executed from internal system RAM

#### **Task:**

In basic use case, this can be the full customer application and can begin executing immediately.

More commonly, this will be the bootloader. Typically, it will:

- Enable crystal, switch to PLL
- Start CAN communications
- Run logic/safety tests
- Enable caches, etc.
- Copy/decrypt/authenticate (Using HSE) further application code from external flash.
- Enable any additional required cores and begin executing main application

## BOOT CONFIGURATION WORD – FOR CUSTOMER BOOTLOADER

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BOOT_SEQ	SWT	BOOT_TARGET	

### BOOT\_TARGET[1:0]:

- 00: A53\_0
- 01: M7\_0

### SWT = Boot Target Watchdog

- 0: Boot Target CPU watchdog is disabled
- 1: Boot Target CPU watchdog is enabled

### BOOT\_SEQ = Secure Boot Mode

- 0: Non secure Boot
- 1: Secure Boot

## CUSTOMER BOOT CODE IMAGE FORMAT

Address offset	Size (bytes)	Name	Comments
0h	4	Image header	Marks start of application image.
4h	4	RAM start pointer	Pointer to the first RAM address to which BootROM must load application boot code.
8h	4	RAM entry pointer	Pointer to out of RESET start of boot target core. For Cortex-M7, it corresponds to VTOR. For A53, it corresponds to start of code execution. This pointer should be within the section of SRAM where the application image is downloaded.
Ch	4	Code length	Length of code section of the image.
10h	48	Reserved	Reserved
40h	Code_len	Code	Code can be any size up to the maximum size of system SRAM.

- Non-secure Application Boot Code image structure (BOOT\_SEQ == 0)

# OTA Updates

---



SECURE CONNECTIONS  
FOR A SMARTER WORLD

PUBLIC

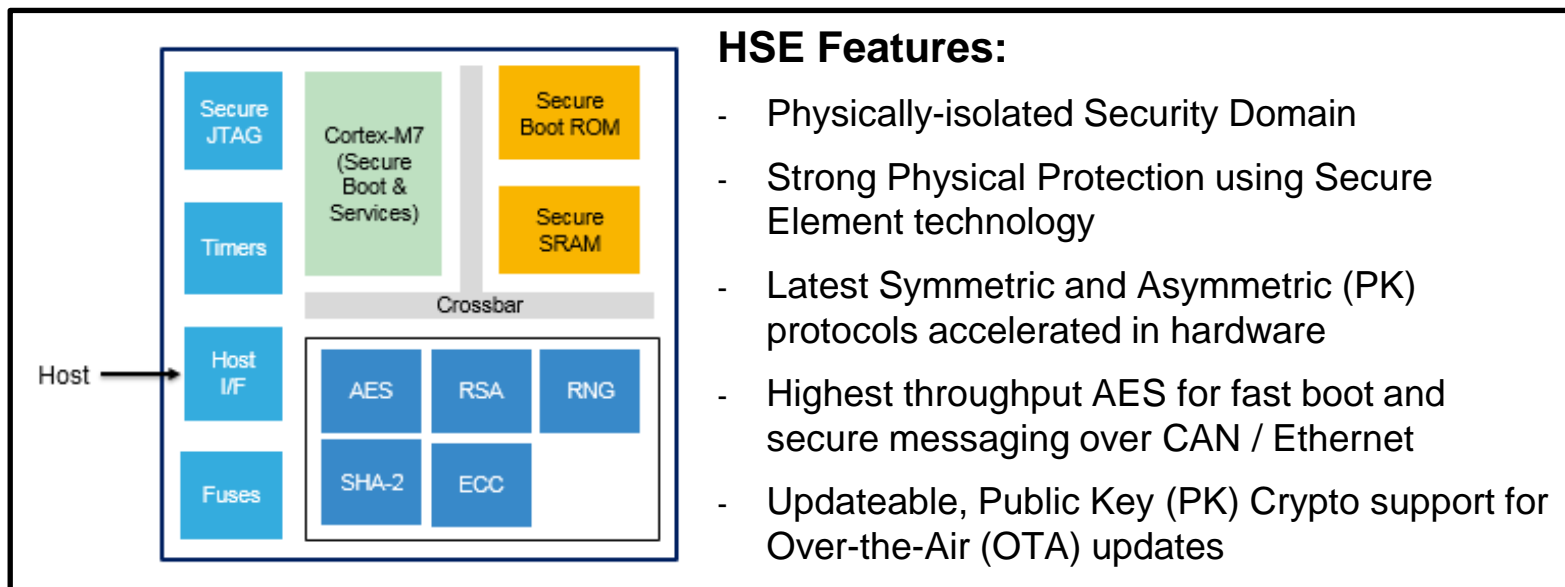
NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.  
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2021 NXP B.V.



## OVER THE AIR UPDATES (OTA)

### OTA Requirement

- Higher level of security needed to address lifetime of vehicles beyond today's specifications (SHE, HSM, EVITA)
- Higher data bandwidth drives higher security performance
- Updatable security required to address future vulnerabilities



## Value Proposition

- Highest performance and most secure embedded security module in automotive processors



## OTA UPDATES

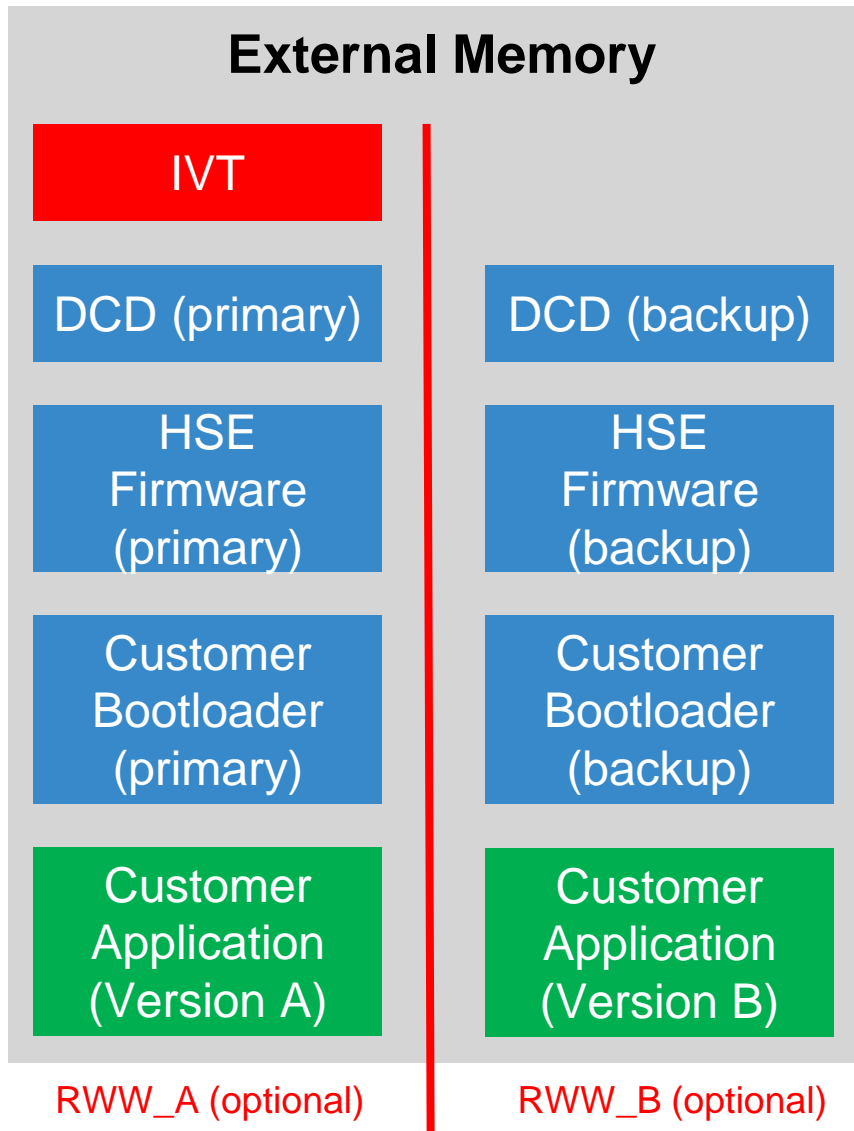
The NXP hardware and firmware controls the booting of the following:




- DCD Records
- HSE Firmware
- Customer Bootloader

The Customer bootloader then loads the **customer application** using a software defined method

Therefore, there is a separate update method depending on what needs updating

## EXTERNAL MEMORY CONTENTS



-  Locked – not updatable
-  Version selected by IVT
-  Version selected by Customer Bootloader

The IVT contains primary and backup addresses for each boot element. If no header is found at primary location, then backup address will be searched.

Backup location can be:

- Exact copy of primary code
- New or older version of primary code
- Small recovery application
- Erased/Invalid

### A/B Method

- Create a backup of the current working version in the location of the IVT's backup pointer.
- Erase the primary image including the boot header. This means that a booting of that image will fail, and the device will auto boot from the backup image.
- Program the content of the primary pointer, except the boot header.
- Verify the new code and data.
- Finally, program the boot header at the primary pointer.
- Upon next boot, the BootROM will boot the new firmware.

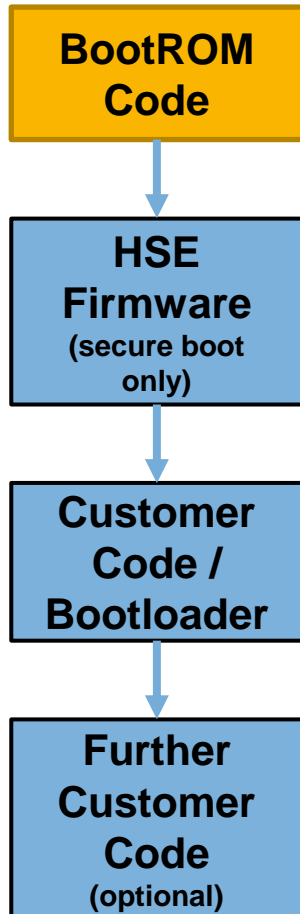
## UPDATING CUSTOMER APPLICATION

The loading of the application will be covered by the customer's bootloader – the process is therefore customer defined.

One method is to have a table of boot vectors and version numbers in OTP flash. With the latest entry pointing to the latest image.

- The update flow:
  - Erase old firmware block
  - Program the new code image
  - Verify the new code image
  - Program new version number and location to OTP flash
  - Upon next boot, the bootloader loads the new version

## ROLL BACK IN CASE OF FAILED UPDATE – WHY?



### Overview:

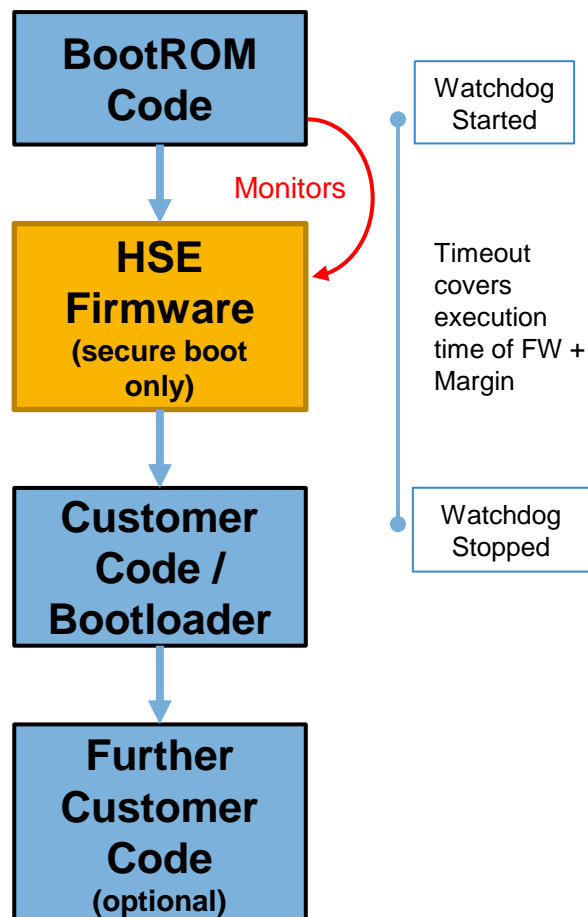
The OTA Client shall check the **integrity** and **authenticity** of the updated code in flash prior to activating it.

What if the code is installed correctly, but the code itself contains an error which prevents the device from advancing to the next stage?

### Protecting against errors in BootROM Code update:

This is a ROM so will not be updated.

## ROLL BACK IN CASE OF FAILED UPDATE – HSE FIRMWARE



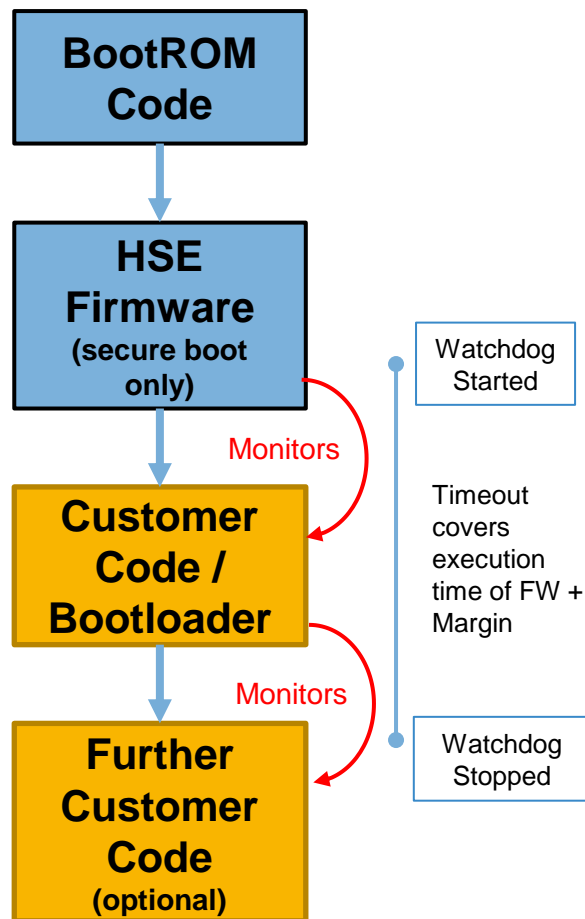
### Protecting against errors in HSE FW Code update:

The BootROM can be responsible for ensuring the HSE FW executes correctly, and hands control over to the customer bootloader.

- BootROM starts a watchdog before handing over control to HSE FW.
  - Timeout set such that it covers execution time of HSE FW + margin
- Customer Bootloader is responsible for disabling the watchdog once it begins execution. So, a watchdog timeout and reset shall only occur if the HSE Firmware fails to load the bootloader correctly.
- At each reset/boot event, BootROM check the reset source. Implements count – if HSE FW watchdog triggers reset >8 times, then it rolls back to “**Backup Image.**”
- Upon first successful boot after an update, OTA Client is responsible for **updating the backup image** to the latest version. This closes a potential attack window whereby the hacker could potentially exploit the rollback feature to re-open a patched security bug.

Note that if the HSE Firmware header is corrupted and the BootROM cannot even begin to execute it, then the BootROM can switch to serial boot mode.

## ROLL BACK IN CASE OF FAILED UPDATE – CUSTOMER CODE



### Protecting against errors in remaining Code:

At this point everything is handled by software. So, any protection can be implemented. Recommend to follow a similar strategy to the BootROM whereby the previous software sets a check that the next software must answer to show that it has begun execution correctly.

### Other Points:

In the event that multiple firmware are being updated at once, it may be necessary to ensure that if one firmware rolls back, then all others also roll back.

- E.g., if a new version of the Bootloader relies on a new feature in a new version of the HSE Firmware. You could not roll back the HSE Firmware but keep new version of bootloader.



SECURE CONNECTIONS  
FOR A SMARTER WORLD