AN13460

FOTA Design for SBL and SFW

Rev. 0 — 20 November 2021 Application Note

Contents

1 Introduction

This application note introduces the design of secure OTA projects, Secure BootLoader (SBL) and Secure FirmWare (SFW), launched by the system application team of RT series MCUs. It also introduces the realization of the FOTA function on it.

SBL and SFW are an OTA project for MCU launched by the System Application team of NXP Edge Processing BL. Now, the project supports most chips of i.MXRT series and LPC55S69 chips.

This application note introduces the design and implementation of FOTA based on i.MXRT1010-EVK board and i.MXRT1020-EVK board.

2 SBL and SFW overview

SBL and SFW are secure firmware update projects for MCU launched by NXP. SBL is a second bootloader used with the FOTA-capable firmware. It manages the update period by verifying and writing the new firmware image to the designated area of internal or external storage devices.

SFW is based on FreeRTOS and is designed to implement a complete FOTA process together with SBL. SFW supports to obtain new firmware images through U disk and SD card locally and through AWS cloud or Alibaba Cloud remotely. After getting a new firmware image, SFW itself writes the image to the storage device, sets the corresponding flag, and reboots the device. After entering SBL, SBL checks the new firmware image and completes the update. Figure 1 shows the structural block diagram of SBL and SFW.

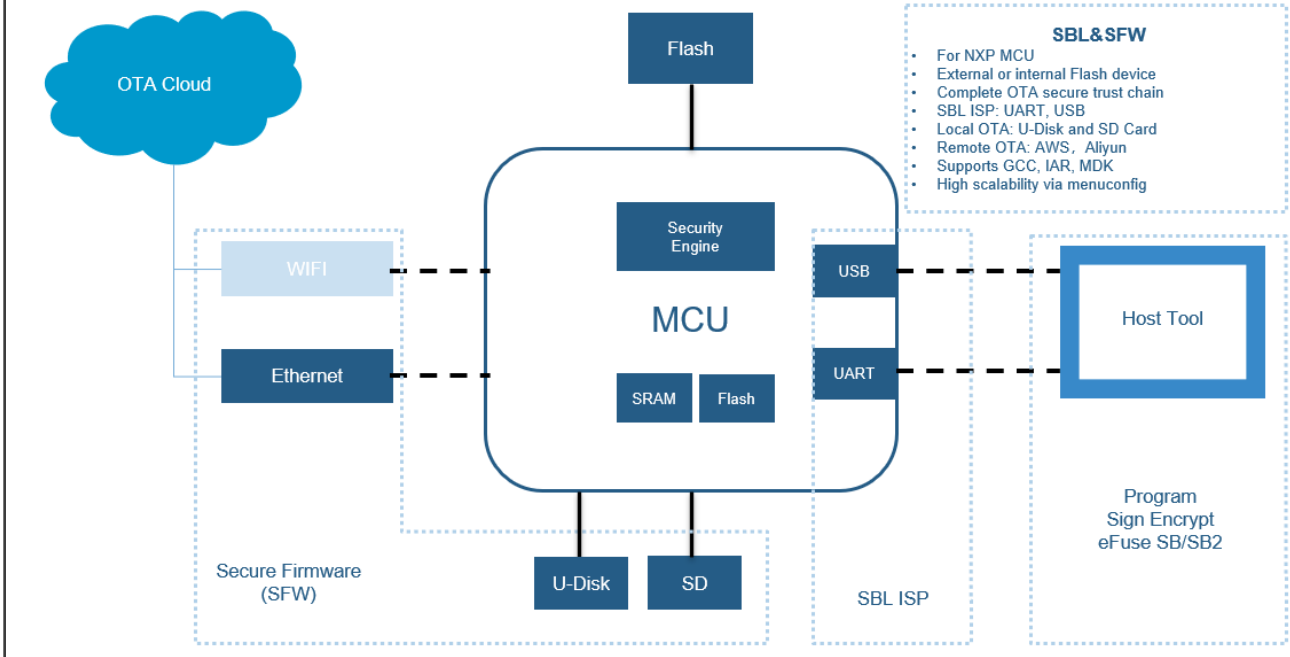

Figure 1. SBL and SFW diagram

3 FOTA structure design

In this application note, FOTA generally refers to firmware updates. The following FOTAs all represent firmware update.

In this project, SFW is the start of a FOTA period. SFW is based on FreeRTOS, so in addition to the task group used to implement application functions, it also creates multiple tasks for FOTA. These tasks are responsible for receiving new firmware and writing it into flash. In the SFW project, SD card update task, U disk update task, and AWS or ALI cloud update task are set respectively. These update tasks are optional and can be enabled in the `menuconfig` of SFW.

The update task of U disk and SD card detects whether there is a new firmware image in the inserted U disk or SD card. If there is a new firmware image, it reads the image and stores the image in the designated location of the flash. The update task of AWS or ALI cloud receives the OTA instruction of the cloud platform. When the cloud platform initiates an OTA request, the update task of AWS or ALI cloud receives the new firmware image from the cloud platform and stores it in the designated location of the flash. Figure 2 shows the basic process.

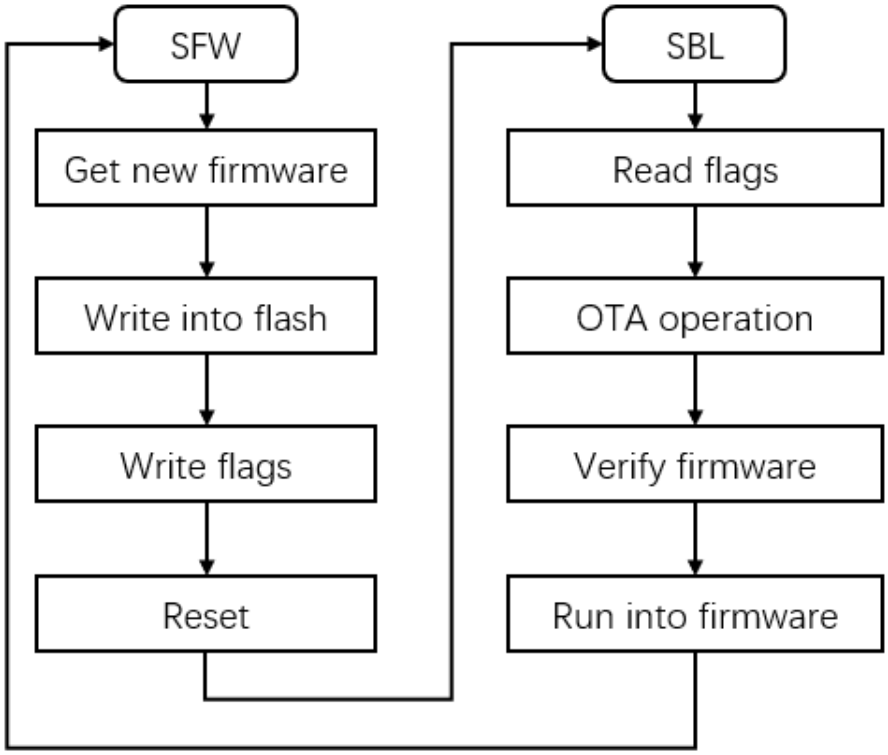

Figure 2. FOTA basic workflow

Many i.MXRT series MCUs chips have the remap function, so SBL and SFW define two different modes on FOTA: Swap mode and Remap mode. The operations corresponding to the write flags, read flags, and OTA operation in the flowchart are different, as described in Swap mode and Remap mode.

To support the revert function of the firmware image, the flash must be divided into at least three pieces to store bootloader, firmware1, and firmware2 respectively. The three pieces are defined as SBL area, Slot1, and Slot2.

3.1 Swap mode

For MCU applications, different from MPU, there is no clear separation between the system and the application. To update the application of the MCU, update the whole firmware. For MCU, the execution address of the program depends on the designation of the link file. The starting address of the program is fixed when the code is linked. To ensure the simplicity of the bootloader, keep the link address of the new firmware consistent with the old one. To achieve this requirement, the bootloader must move the new firmware to the address specified by the link file and then jump to run.

To achieve the revert function, bootloader must save the old firmware. The best way to achieve the two requirements is to exchange the location of the old image and new image in the storage. This mode is defined as the Swap mode.

The Swap mode in SBL is based on the open source `mcuboot` project, using the v1.7.0 of the `mcuboot` source code. In the Swap mode, the firmware image must have a fixed structure. SBL and SFW can control the FOTA process through operating the flags at the specified position of the flash. Figure 3 shows the flash organization of the Swap mode.

Figure 3. Flash layout of Swap mode

SBL is at the start of the flash. The ROM of the i.MXRT series chips jumps to this position in XIP mode. The firmware images stored in Slot1 and Slot2 are composed of four parts. Image Header is at the top of the image. It tores the length, version number, and other information of the firmware. Following the Image Header is the firmware itself. The firmware signature is behind the Image. At the end of the slots is the Image Trailer, which stores the flag of the FOTA process.

The SFW generates a pure firmware image without Image Header, signature and Image Trailer after compiling. To be used as the new firmware image in the FOTA, use a script to add a header and signature to the pure image. SFW and SBL programs edit the Image Trailer part. Table 1 describes the structure of the Image Header.

Table 1. Image header structure

Offset	Width (bytes)	Field	Description
0x00	4	magic	Image header tag Fixed value
0x04	4	load_addr	Point to the load address of the application
0x08	2	header_size	Size of the image header
0x0a	2	reserved	Reserved for future use
0x0c	4	image_size	The size of the image (not including the Image Header Size)
0x10	4	flags	Not used now
0x14	8	image_version	Image version
0x1c	4	reserved	Reserved for future use

Table 2 describes the structure of Image Trailer.

Table 2. Image trailer structure

Offset	Width (bytes)	Field	Description
0x00	1	copy_done	Flag that the Swap done
0x01	7	Pad	Reserved
0x08	1	image_ok	Flag that control the OTA state
0x09	7	pad	Reserved
0x10	16	magic	Image trailer tag Fixed value

In the Swap mode, the jump address of SBL is set as the firmware start address of Slot1 fixedly. The new firmware image obtained from SFW is always stored in Slot2. When SFW gets the new image through the local or remote ways, it writes the image into the flash space of Slot2. After the writing completes, the magic field in the Image Trailer of Slot2 is written to let the SBL know that there exists a new firmware image.

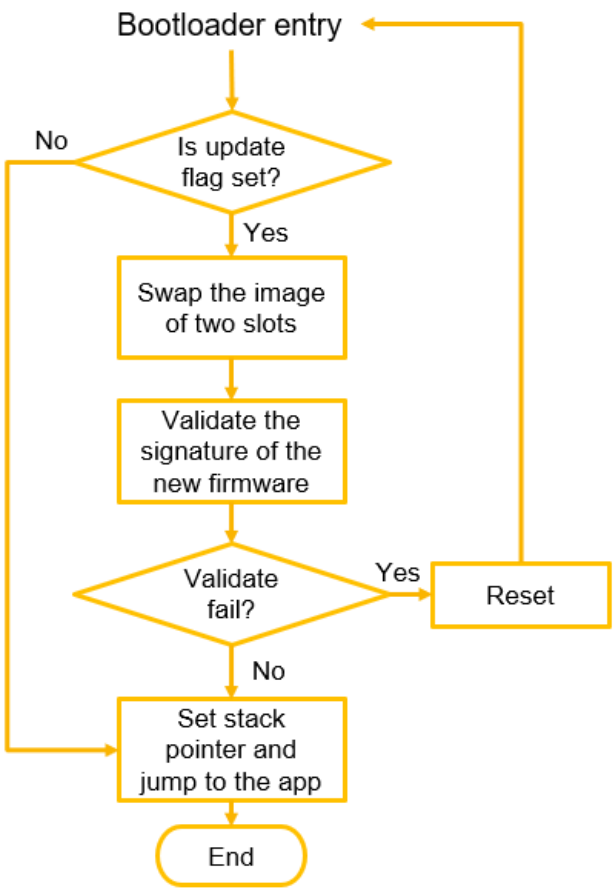

Figure 4. Swap diagram of SBL

Figure 4 shows the basic logic of the Swap mode in SBL. SFW resets the MCU, the code runs into SBL, and the process of the SBL is slightly different from the process in `mcuboot`. In the `mcuboot` project, first verify the signature of the new image stored in

Slot2. If the new image passes the verification, SBL exchanges the images in Slot1 and Slot2. However, in i.MXRT series chips, ROM has the hardware verification function which is related to the link address. When verifying the signature, the address of the image must be the same as the link address. To support the ROM verification function, exchange the image first. Verify that the signature after the new image is stored in Slot1.

Before swapping images, SBL must judge the current FOTA status, and then decide whether to swap the images of the two Slots. SBL defines four states for FOTA in the Swap mode, they are Test, Perm, Revert, and None.

- **Test** means temporary exchange. The exchange is reverted if the `image_ok` flag is not set.
- **Perm** means permanent exchange. This exchange is permanent and cannot be reverted.
- **Revert** means rollback exchange. The new firmware meets problems and reverts to old firmware.
- **None** means no exchange.

SBL gets the current FOTA state by reading the value of the Image Trailer of the two Slots. For Test, Perm, and Revert states, Figure 5 shows the configurations of the Image Trailer.

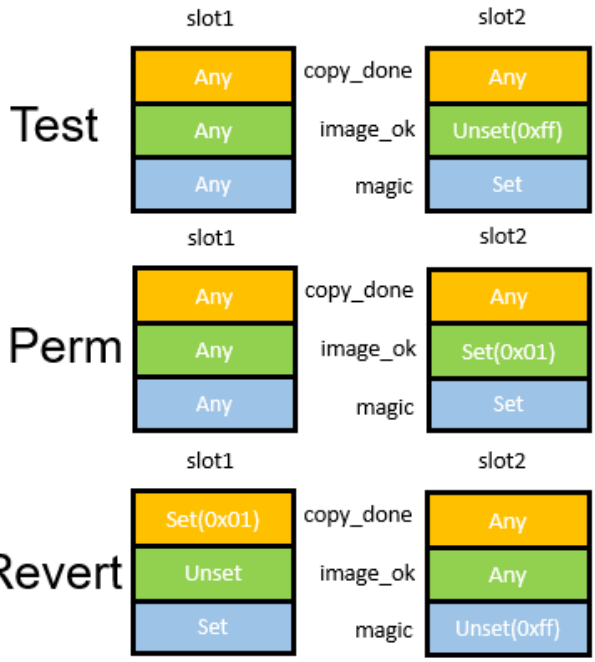

Figure 5. FOTA states for Swap mode

All Image Trailer values that do not belong to the above three states are None state, and no image exchange occur. In the actual SBL project, the image revert function is supported by default, so the Perm state is not used.

The implementation of the revert function also based on the state of the Image Trailer. When SFW receives a new image and writes it to Slot 2, it writes the magic field of the Image Trailer of Slot 2. After restarting and go back to SBL, SBL gets the Test state, and perform the exchange of the two Slots. The Image Trailer of Slot 1 is not copied to Slot 2, and the original Image Trailer of Slot 2 becomes the Image Trailer of Slot 1 after the exchange. Now the Image Trailer of slot2 is all `0xFF`. When the exchange completes, SBL writes the `copy_done` flag of Image Trailer of Slot1 as `0x01`. After doing the above operations, SBL verifies the signature of the image in Slot1, and the jump is performed after the verification is successful. After jumping to the new firmware, SFW writes the `image_ok` field of Slot1 as `0x01` if the application runs well. If the SFW occurs a problem before the writing operation and the `image_ok` field is not set, after the program resetting, SBL gets the Revert state. Then, it reswaps the images of Slot1 and Slot2, which called image revert.

Figure 6 shows the SBL code flow of the entire the Swap mode.

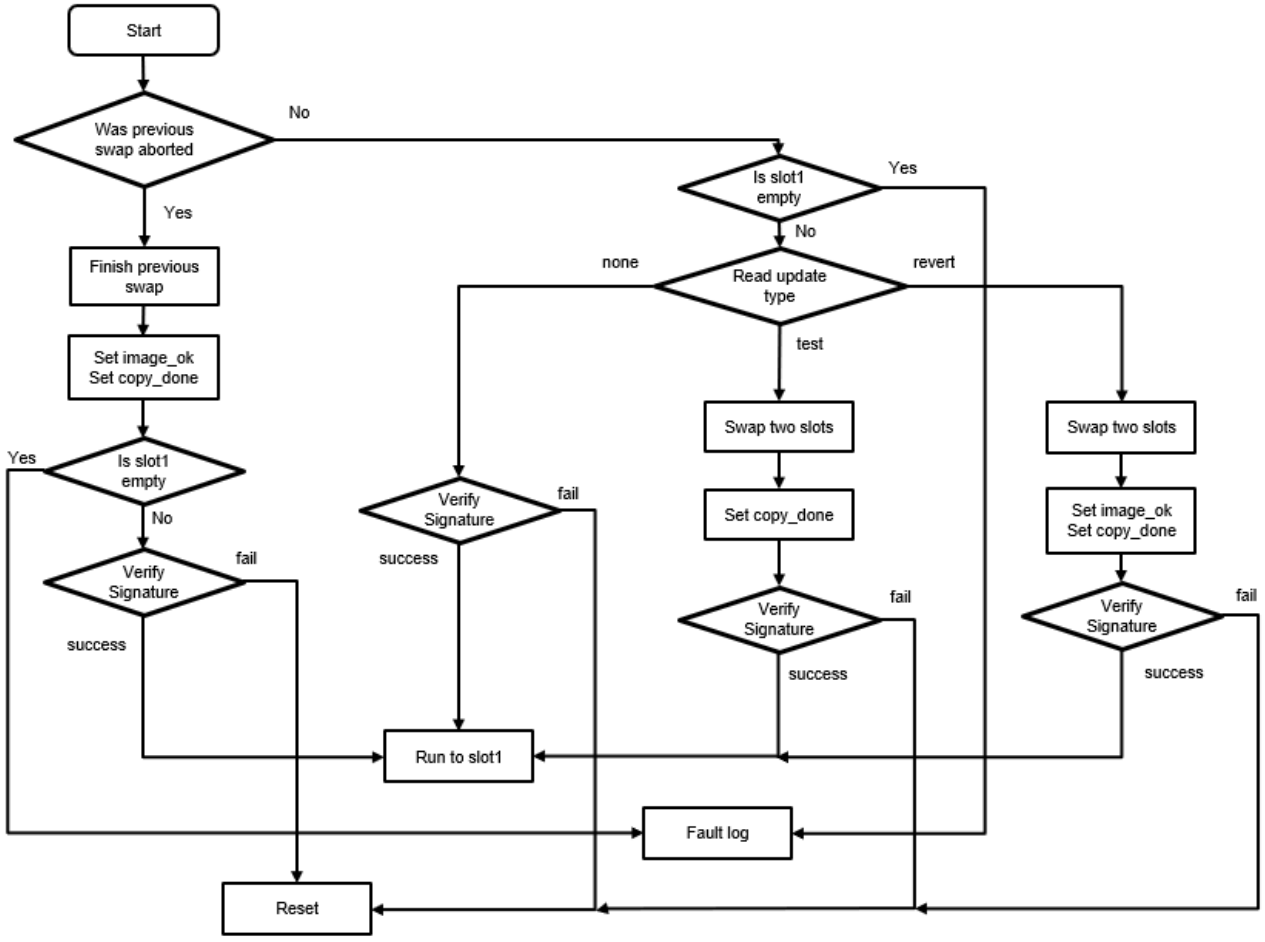

Figure 6. Code flow of Swap mode

The exchange of two Slots in the Swap mode does not use the third flash piece. The size of the firmware image cannot exceed the size of the Slot minus the size of the two Sectors (one is the Image Trailer part). SBL uses the method below to finish the exchange. For image in Slot1, shift the entire firmware to the lower address by one sector size, copy the first sector of Slot2 to the first Sector of Slot1, copy the second sector of Slot1 to the first sector of Slot1, and so on.

3.2 Remap mode

Some chips of i.MXRT series have the feature of remap. The addition of Remap mode is based on this feature. To enable this feature, set up three registers, the start address, end address, and offset address. After enabling this function, if the MCU accesses the flash through the AHB bus and the access address is between the previous start address and end address, the actual physical address accessed becomes the value of the access address plus the offset address.

This feature is naturally suitable for FOTA application. After enabling this feature, bootloader must not exchange images and the SFW can directly use the same linker file to run on different physical addresses. Compared with the Swap mode, this function greatly reduces the number of flash erasing and writing, prolongs the service life, and shortens the time for FOTA greatly.

Figure 7 shows the organization of the flash in the Remap mode.

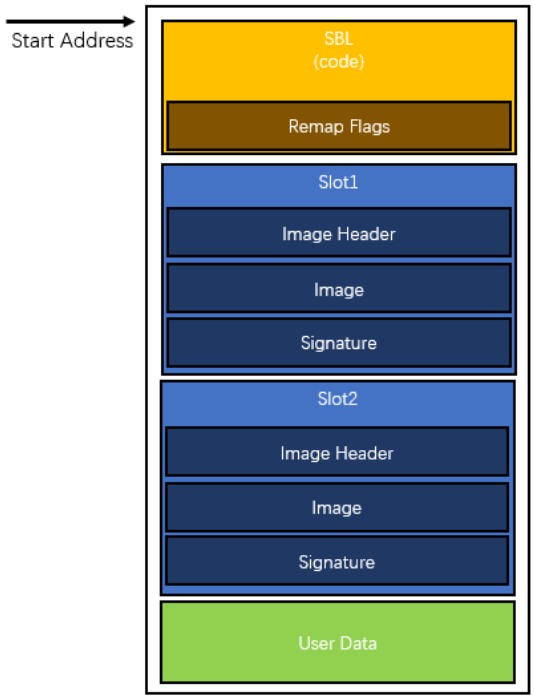

Figure 7. Flash layout of Remap mode

Similar to the division of Swap mode, it is divided into SBL, Slot1, Slot2, and User Data. The difference is that the Image Trailer part of the flag that stores the FOTA process in Slot1 and Slot2 has been canceled. Instead, at the end of the SBL region, set a structure called Remap Flags to control the FOTA process.

The function and structure of the Image Header part are the same in the Swap mode, as shown in Table 1.

Table 3 describes the structure of Remap Flags.

Table 3. Remapping flag structure

Offset	Width (bytes)	Field	Description
0x00	1	Image_position	The current firmware position `0x01` | `0x02`
0x01	7	Pad	Reserved
0x08	1	image_ok	Flag that control the OTA state
0x09	7	Pad	Reserved
0x10	16	magic	Image trailer magic Fixed value

In the Remap mode, the logical jump address of SBL is still fixed to the start address of the firmware image in Slot1. Unlike Swap, SBL sets the Remap area to the size of Slot1 and the offset value is also set to the size of Slot1. From the AHB bus, the logical address accessed by the core that in the Slot1 area accesses the same position in Slot2 physically. In other words, by enabling and disabling the remap function, SBL can jump to a different physical address to perform firmware using the same jump address.

Therefore, the new firmware image obtained from SFW does not have a fixed storage location. According to the Slot in which the previous firmware is running, SFW stores the received new firmware image in another Slot. Therefore, there is 1 byte in the Remap Flags in Table 3 to store the location information of the current firmware. When SFW receives a new firmware image and writes it into another Slot, it sets the magic value in Remap Flags to let the SBL know that the new firmware must be updated.

After resetting back to SBL, the code logic of SBL follows the process shown in Figure 8. To support the verification function of ROM in the i.MXRT series chips, SBL first reads the Remap Flags to get the position of the currently running firmware and then judges whether there is a new firmware to be updated according to the magic field.

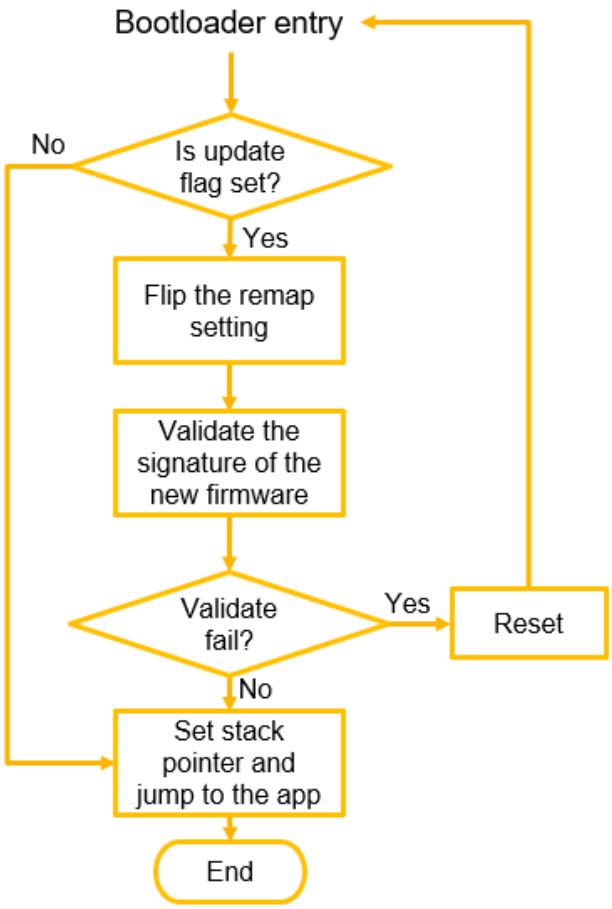

Figure 8. Remap diagram of SBL

Since the entire Slot has been remapped and the content of the flag has changed, the Remap Flags cannot be set in Slot1 and Slot2. The SBL and SFW projects use the last 32 bytes of the SBL area in the flash as Remap Flags. The revert of firmware images is also supported in Remap mode. The state of Remap Flags controls the FOTA process. In remap mode in SBL, FOTA defines four states, Test, Perm, Revert, and None.

- **Test** means temporary update.
- **Perm** means permanent update.
- **Revert** means rollback.
- **None** means no update.

SBL gets to the current FOTA state by reading the value of Remap Flags. For Test, Perm, and Revert states, Figure 9 shows the settings of Remap Flags.

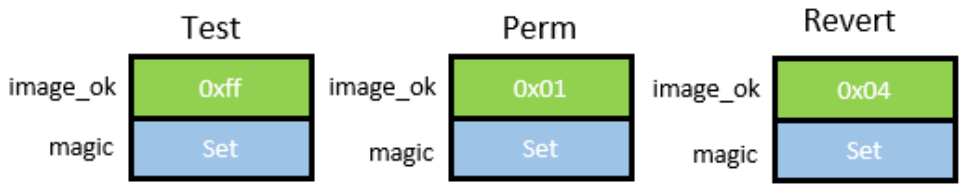

Figure 9. FOTA states of Remap mode

The Revert function of Remap mode is implemented based on the value of the `image_ok` field. When SFW receives a new image, it reads the `image_position` field in Remap Flags, gets the slot where the current firmware is located, stores the new firmware image in another slot, and writes the magic value to the corresponding field of Remap Flags. After rebooting back to SBL, SBL reads the location of the current firmware image and gets to the Test state.

- If the current firmware image is in Slot1, SBL enables the remap function and maps the address of Slot1 to the physical address of Slot2.
- If the current firmware image is in Slot2, SBL disables the remap function and the logical address of Slot1 remains the same as the physical address.

After verifying that the signature of the new firmware image is correct, SBL writes the `image_position` field as another Slot (`0x01` to `0x02` or `0x02` to `0x01`) and writes the `image_ok` field as `0x04` which represents the Revert state. Then, SBL jumps to the logical address of Slot1. After jumping to the new image, if the basic tasks run without any problems, SFW writes the `image_ok` field of Remap Flags as `0x01`. If there is a problem during the operation, this field is not written. After the program resetting, SBL gets to the Revert state, re-enables or disables the remap function, and finally reverts the image.

Figure 10 shows the SBL code flow of the entire Remap mode.

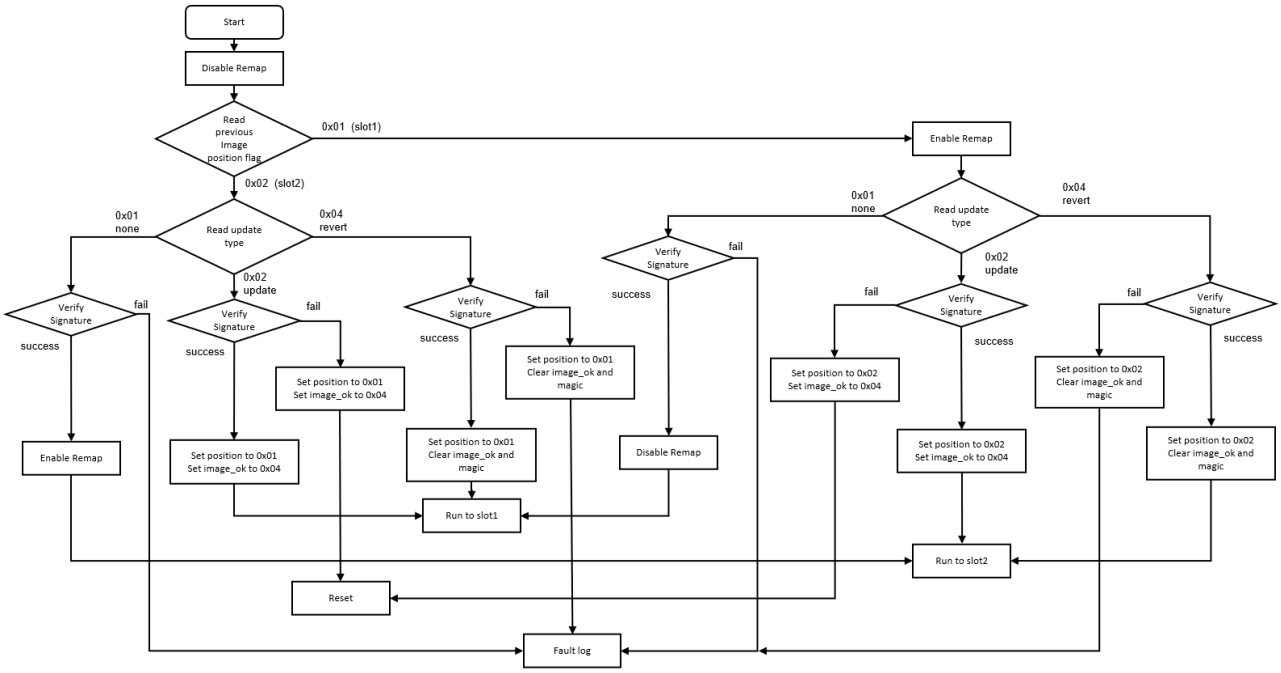

Figure 10. Code flow of Remap mode

During the whole judgment process, when the signature of the updated firmware fails to pass the verification, SBL actively restarts the board, re-enters the judgment process of SBL, gets the revert state, and completes the program rollback.

Figure 11 describes the entire process of rolling back after an unsuccessful update and then performing a successful update.

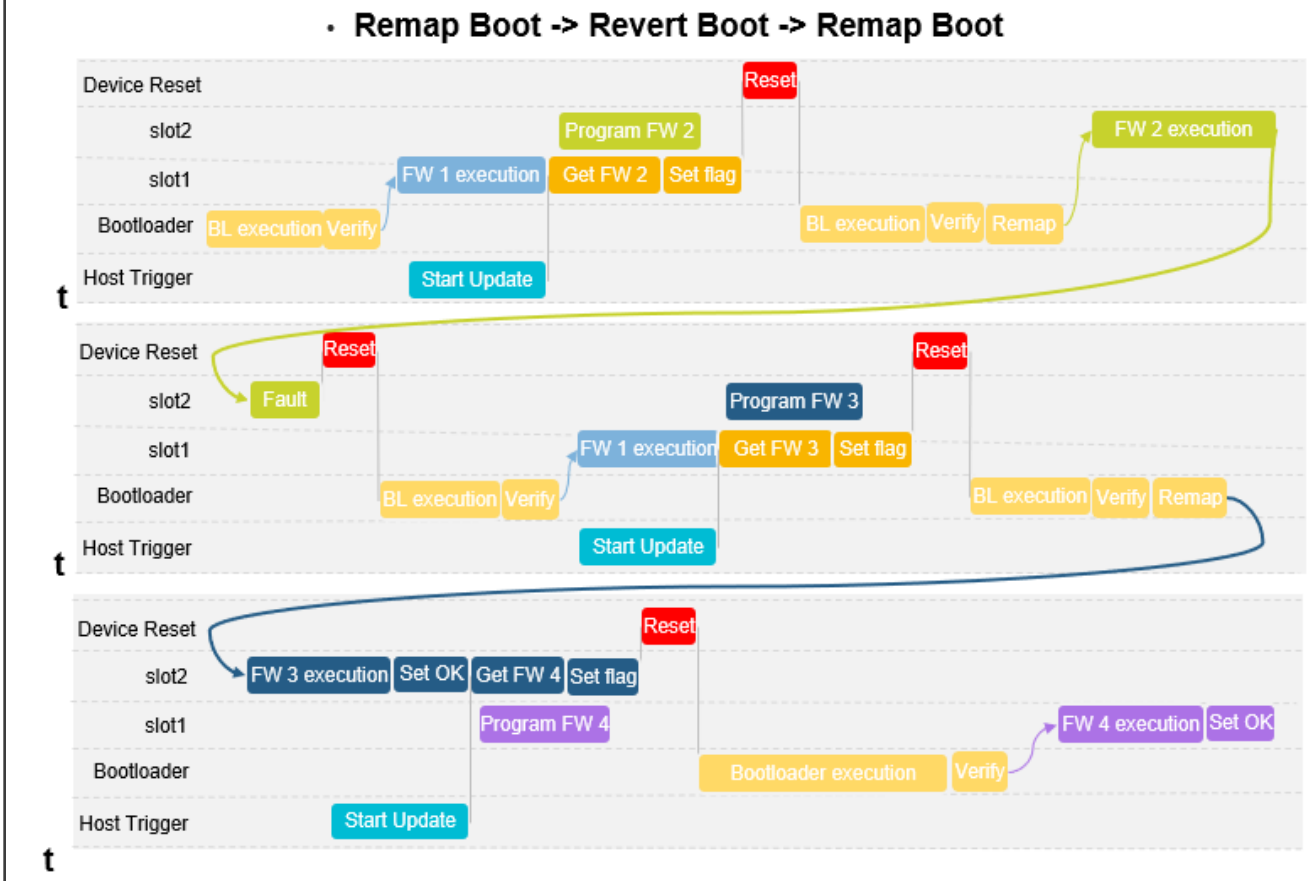

Figure 11. Remap mode update process

4 References

1. SBL Repository https://github.com/NXPmicro/sbl
2. SFW Repository https://github.com/NXPmicro/sfw
3. MCU-OTA SBL and SFW User Guide (document MCUOTASBLSFWUG)

5 Revision history

Rev.	Date	Description
0	20 November 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. M, M Mobileye and other Mobileye trademarks or logos appearing herein are trademarks of Mobileye Vision Technologies Ltd. in the United States, the EU and/or other jurisdictions.

© NXP B.V. 2021. **All rights reserved.**

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 20 November 2021

Document identifier: AN13460

arm