AN13499

U-Disk and SD Card Update in SFW

Rev. 0 — 25 December 2021 Application Note

Contents

1 Introduction

This application note introduces two local update methods implemented in SFW: U-Disk and SD card update.

2 SFW overview

SFW is an application project used with the SBL project launched by the i.MXRT series MCU's SE team. The main function is to demonstrate how to run the firmware update with the SD card, U disk, and ALI or AWS cloud platform.

SFW is created and developed based on FreeRTOS. It can work with SBL to perform a complete FOTA process. Two print tasks are created in SFW, and print "hello world" at a frequency of 1 second. The two tasks are established as simulated application tasks. Then create SD card update tasks, U disk update tasks, and AWS cloud or Alibaba Cloud update tasks according to the macro configured in the `menuconfig` [1] configuration.

3 Firmware update process

SFW is developed as a firmware application supporting the SBL. The basic function of SFW is to receive new firmware and write it to flash in various ways. SBL supports Swap and Remap firmware updates and these two modes have different requirements for writing addresses. SFW also distinguishes between Swap and Remap mode.

In the Swap mode, the flow of SFW firmware update task is as shown in Figure 1.

[1] A graphical configuration interface in SBL and SFW projects. For details, see SBL and SFW user guide.

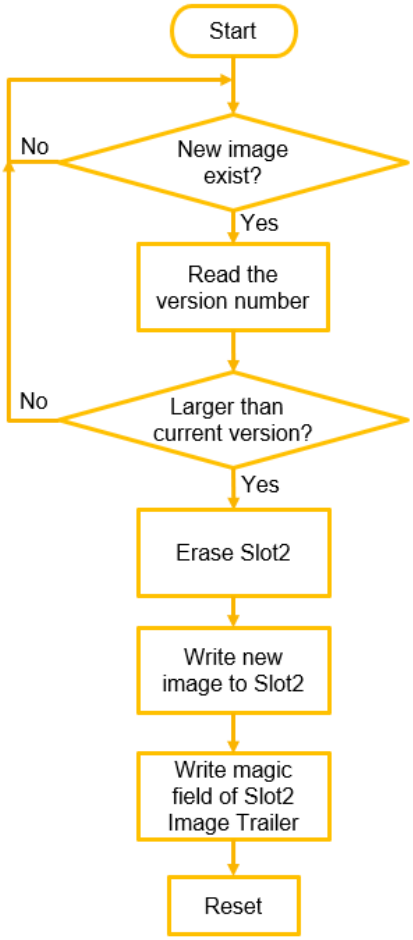

Figure 1. Update flow of Swap mode

As introduced in *FOTA Design for SBL and SFW* (document AN13460), SBL has three areas in flash, SBL area, Slot1, and Slot2. In the Swap mode, the new firmware image is always saved to Slot2. After receiving the new firmware image, SFW verifies the version number. If the new version number is greater than the old version number, SFW erases the Slot 2 part, reads the image data in batches, and writes to Slot2. After all the image data are written, SFW needs to set the flags, where SFW writes the magic field of the Image Trailer structure at the end of Slot2. After the writing completes, reset the chip.

In the Remap mode, the flow of the SFW firmware update task is as shown in Figure 2.

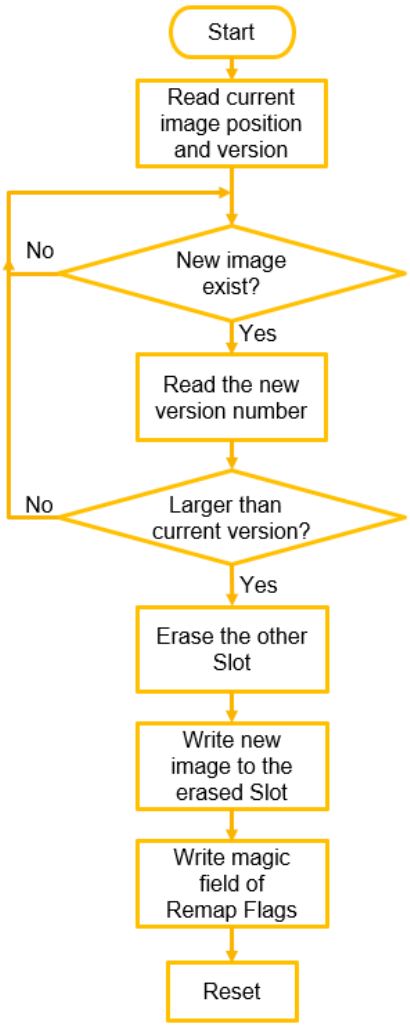

Figure 2. Update flow of Remap mode

In the Remap mode, there is no fixed storage location for the new firmware image. SBL sets a flag to indicate the location of the currently running firmware image. For the specific introduction of this flag, see *FOTA design for SBL and SFW* (document AN13460). After detecting the insertion of the SD card or U disk, SFW reads the location and the version number of the currently running image. Then, read the new version number from the Header of the new image. After SFW verifies that the new firmware version number is greater than the current version number, it erases the space of another Slot according to the location read before and reads the image data in batches according to the set buffer size and then write to the erased Slot. After all the image data are written, set the flag bytes. SFW writes the magic field of the Remap Flags structure at the end of the SBL area. After the writing is completed, reset the chip.

For both Swap mode and Remap mode, after writing the new firmware image to the corresponding flash space, the last step is to write the flag. SFW provides the `enable_image()` function to write the flag bytes. It uses a macro to distinguish the definition of this function in Swap mode and Remap mode.

3.1 U-Disk update task

The feature of U-disk update is based on the USB Host Controller of the i.MXRT series and LPC55 series chips. These chips have integrated USB PHY internally. Supported by corresponding SDK software packages, the chips can quickly enable commonly used applications.

In SFW, the update of the U-disk refers to the `usb_host_msd_fatfs` demo in the SDK. The MCU operates as a USB Host. After inserting the U-Disk, SFW detects whether there is an image file in the U-Disk. If the image exists, SFW runs the update process.

The U-Disk update process includes two tasks:

- `USB_HostTask` is used to manage the USB data transmission related to the controller. Notify the application layer USB device status through the callback function.
- `USB_HostApplicationTask` is used to manage the device status and the running status of the device class instance. After initializing the clock and USB Host stack, SFW creates these two tasks.

In the SDK demo, after the U-Disk is inserted and configured, it calls the function that runs the read-and-write test of the files in the U-Disk. In SFW, the update process described in the previous section replaces the read-and-write test function. The corresponding driver of `FatFs` is integrated in the demo. SFW calls the function of `FatFs` in the update task to mount the U-Disk and to read file. Set the file name of the new firmware as `newapp.bin`. When there is a file with this name in the U-Disk, the file content is read. The header of the file stores the version information of the firmware, so the SFW reads the header and verifies the version number. If the verification is passed, SFW reads the image in batches and writes them into the flash. After the writing completes, call the `enable_image()` function to enable the flag bytes and performs a software reset.

3.2 SD card update task

Similar to the U-Disk update, the SD card update depends on the chip's support for the SD interface. The SDK packages of the i.MXRT series and LPC55 series chips provide the `sdcard_fatfs_freertos` demo. SFW implements the SD card firmware update based on this demo.

The SD card update only creates one task called `sdcard_ota_app`, which is an SD card detection task. After the SD interface is initialized, a semaphore is created. Wait for the SD card to be inserted. In the main loop of the task, try to obtain the semaphore. When the SD card insertion is detected, the previously created semaphore is released in the callback function. After the semaphore is obtained, power on the SD card and enter the update process described above.

Also like the U-Disk, the SD card update uses the `FatFs` function to mount SD card and reads the files stored in it. The file name of the new firmware is also `newapp.bin`. When a file with the same name exists in the SD card, the version information of the header is read. After the version verification is passed, read the contents of the image in batches and write to the flash. After the writing completes, SFW calls the `enable_image()` function to enable the flag bytes and then performs a software reset.

To cooperate with SBL, there is another operation to write an extra flag byte in SFW to let the updated firmware in effect permanently. Write this flag byte when the new firmware is running well. If this flag is not set, SBL reverts the firmware image to the previous old image when the next time it encounters a reset event.

The function for writing this flag byte is defined as `write_image_ok()` in SFW. It is called in the initialization process of SD card and U disk.

- If the latest-updated source is a U-Disk, it is called once before the main loop of `USB_HostApplicationTask`.
- If the latest-updated source is an SD card, this function is called after the SD interface is initialized.

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

Revision number	Date	Substantive changes
0	25 December 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.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25 December 2021
Document identifier: AN13499

arm