Airoha IoT SDK Firmware Update Developer's Guide

Version: 2.0

Release date: 14 March 2023

Document Revision History

Revision	Date	Description
1.0	17 October 2018	• Initial release
1.1	22 May 2019	• Add AM255x
1.2	25 June 2019	• Do not use Full binary. Add note for single device FOTA update and dual device FOTA update
1.3	19 August 2019	• Added AES key change and external flash support
1.4	29 November 2019	• Added the external flash description for TargetAddress
1.5	3 July 2020	• Added AB1565 and AB1568
1.6	20 August 2020	• Add constraints
1.7	27 October 2021	• Introduce how to use flash_download.cfg to configure the FOTA package • Synchronize the AB1565 memory layout with the latest SDK • Introduce FOTA update with the dongle involved
1.8	28 December 2021	• Added AB1565 and AB1568 8M flash layout
1.9	5 January 2022	• Added AB158x
2.0	14 March 2023	• Added AB157x

Table of Contents

Lists of Tables and Figures

1. Overview

Airoha IoT SDK enables firmware over the air (FOTA) update, a widely adopted cost and time efficient solution to update the firmware on the connected devices. The purpose of the developer's guide is to provide complete description on how to deploy FOTA on the NOR flash.

This document guides you through:

- FOTA architecture layout.
- FOTA update on Airoha IoT development platform.
- FOTA update workflow.

1.1. Feature support status on chips

Airoha IoT SDK development platform is an integrated solution with limited availability of FOTA features on each chip. The feature support summary is shown in Table 1. FOTA feature support status on chips and the chips of each product line are listed as below.

- Airoha IoT SDK for BT Audio: AB155x, AB1565, AB1568, AB158x, AB157x
- Airoha IoT SDK for Smart MCU: AM255x

Table 1. FOTA feature support status on chips

Product Line	Chip	Single device FOTA update	Dual device FOTA update	Update through Bluetooth
Airoha IoT SDK for BT Audio	AB155x	✓	✓	✓
	AB1565/AB1568	✓	✓	✓
	AB158x	✓	✓	✓
	AB157x	✓	✓	✓
Airoha IoT SDK for Smart MCU	AM255x	✓		✓

Note:

AM255x does not support MCSync (MultiCast Synchronization). Therefore, it does not support Dual device FOTA update.

Single device FOTA update: FOTA updates one device once.

Dual device FOTA update: FOTA updates two devices once.

1.2. Architecture layout of the SDK FOTA feature

The FOTA architecture for updating the firmware through Bluetooth using RACE CMD is shown in Figure 1.

The FOTA **Update Agent** is part of the **Bootloader** and uses the flash APIs to process the update on the target device.

The **RACE CMD protocol** on the Airoha IoT device and the mobile device is for exchanging data between the two devices through Bluetooth. The **FOTA application** on the mobile phone sends the FOTA package to the Airoha IoT device and triggers the Airoha IoT device to reboot and start the FOTA upgrade process.

The **FOTA packaging tool** is used on the PC to apply the Lempel–Ziv–Markov chain algorithm (LZMA) compression and sign the signature in the new firmware file and generate a FOTA update package.

A description of the numbered items in Figure 1 is given below:

1) **FOTA packaging tool** — Generate an updated package.
2) **HTTP server** — The storage location of the updated package.
3) **FOTA application** — The Bluetooth-enabled mobile device downloads the updated package from the server.
4) **RACE CMD protocol** — The mobile device pushes the updated package to the device using Bluetooth.
5) **FOTA RACE CMD** — Sets the triggered flag and reboots the device as **FOTA application** requested when the complete package is received.
6) **Update Agent** — Checks the flag status and proceeds with the update.

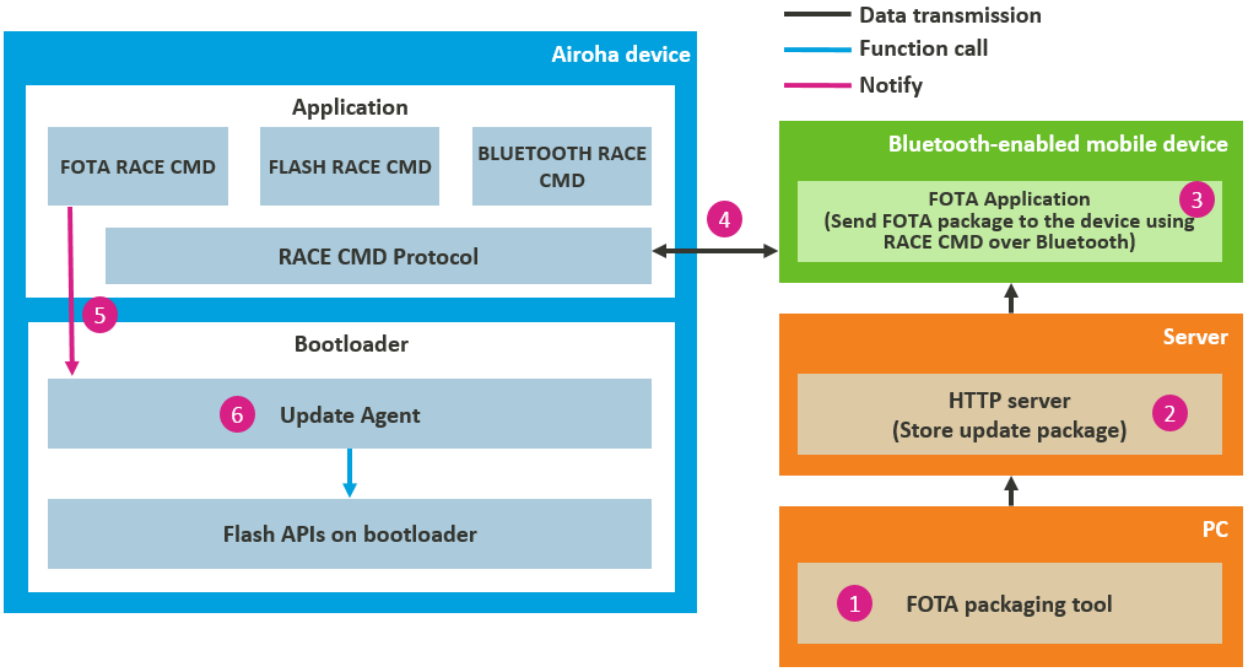

Figure 1. FOTA architecture layout for updating the firmware through Bluetooth using RACE CMD

2. Using the FOTA Update

2.1. FOTA packaging tool

Airoha provides a FOTA packaging tool that runs on Microsoft Windows OS to compress data, generate a checksum and prefix header for the new binary FOTA package file. During the update, the **Update Agent** parses the header, validates the package file using the checksum and identifies where the data should be written.
A FOTA package file is generated by using the FOTA package tool the UI of which is shown in Figure 2.

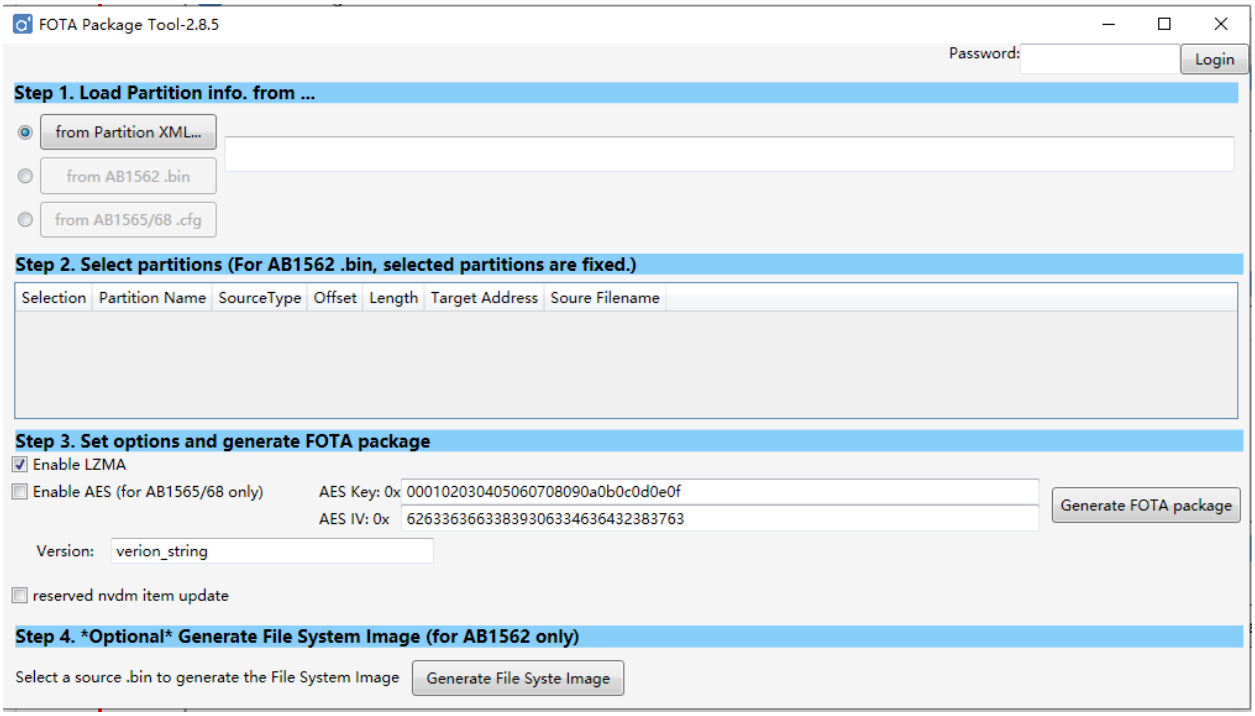

Figure 2. FOTA packaging tool UI

To use the FOTA packaging tool:

1) Double click Airoha.Tool.Kit.exe.
2) Click the "FOTA Tool" button.
3) Click the "from Partition XML…" button and select the specific partition xml.

   Refer to the Partition XML introduction below for more information.

   For AB1565/AB1568/AB158x/AB157x, instead of the Partition XML, flash_download.cfg can also be used to configure the FOTA package file. Click "from AB1565/68 .cfg" button to use flash_download.cfg file. Refer to the flash_download.cfg introduction below for more information.
4) Select the Flash Partition(s) that you want to upgrade.

   If you select more than one bin in the UI, the FOTA package file contains the information for those bins and the device upgrades multiple bins in multiple Flash sectors when executing the FOTA upgrade process.
5) Select whether "Enable LZMA" compression is applied.
6) Select whether "Enable AES" encryption is applied.

AES is dependent on LZMA. The AES checkbox is only available when the LZMA checkbox is selected. The AES key and AES IV set in the tool must be the same as the OTA_ENC_KEY and the OTA_ENC_IV in the BOOTLOADER_PROJECT_DIRECTORY/GCC/Makefile.

7) Click the "Generate FOTA package" button to create the final FOTA package.

A Partition XML file configures basic information for each of the Flash Partitions, such as the path of the bin file, the target address, and so on. The FOTA packaging tool creates the FOTA package based on the information the partition XML file provides. As shown in Figure 3, there are some XML elements in the Partition XML. Refer to the sample under <Tool_path>/setting folder with the name of partition_description_X.xml.

- Partition element – Each Flash partition has its own Partition element.
- Selected element – The Selected element shows whether the checkbox of the Flash partition is automatically selected in the UI. The checkbox is automatically selected only when the selected element is "True".
- Name element – The name of the Flash partition.
- SourceType element – The SourceType element shows the source type of the bin data for each Flash partition. WholeFile means the bin data for the current Flash partition is the complete file specified by SourceFilename.
- TargetAddress element – The TargetAddress element indicates the start address of the Flash partition in which the bin file is written. If the bin file is stored in the internal Flash, the TargetAddress element should be the physical address. And if it is stored in the external flash, the TargetAddress element should be the physical address with SPI_SERIAL_FLASH_ADDRESS.

  The physical address can be obtained with the address defined in the memory layout script such as ab155x_flash.ld in the project folder. For example, if the address is 0x08012000 in the memory layout script, its corresponding physical address is 0x00012000.

```
/* Memory Spaces Definitions */
MEMORY
{
   ...
   ROM_RTOS(rx) : ORIGIN = 0x08012000, LENGTH = 1128K
            ...
}
```

  SPI_SERIAL_FLASH_ADDRESS is defined in mcu/driver/CMSIS/Device/MTK/<Chip>/Include/<Chip>.h.

- SourceFilename element – The SourceFilename element indicates the path of the file which contains the bin data for the current Flash partition.

```
<Partition>
  <Selected>True</Selected>
  <Name>CM4</Name>
  <SourceType>WholeFile</SourceType>
  <TargetAddress>0x00012000</TargetAddress>
  <SourceFilename>D:\example\earbuds_ref_design\earbuds_ref_design.bin</SourceFilename>
</Partition>
```

Figure 3. Partition element

The flash_download.cfg file is generated automatically after the build flow. It can be found under the sub folder of <sdk_root>/out. Normally it contains the information of the flash partitions that need be updated. Because of this, it can be used to replace the partition XML file mentioned above to configure the FOTA package file by using the

FOTA packaging tool. Select the "from AB1565/68 .cfg" button of the FOTA packaging tool to use flash_download.cfg to configure the FOTA package file.

2.2. FOTA update methods

2.2.1. Single device FOTA update

This update method is easy to use and does not occupy much flash memory space if a compression algorithm is applied to reduce the update package size.

A new FOTA package file is generated using the FOTA package tool. There is a reserved space on the NOR Flash to store the new FOTA package file. Once the new FOTA package file is successfully transferred and stored in the specified address of the reserved space, the system reboots automatically. Normally, the system reboots to enter bootloader phase.

Once the bootloader detects there is a new FOTA package file on the flash, it moves the new bin file(s) included in the FOTA package file into the corresponding flash partitions. After it is done successfully, the system reboots with the new bin file(s).

The compile options for the single device FOTA update are configured in the main project and bootloader project makefiles (`feature.mk`). One of the makefiles is in the main application folder, and the other is in the bootloader project folder that builds the bootloader binary. Only when FOTA options in both of them are enabled, the FOTA feature is enabled.

Configure the following options in `PROJECT_DIRECTORY/GCC/feature.mk` of the main application.

```
AIR_FOTA_ENABLE = y
AIR_FOTA_VIA_RACE_CMD_ENABLE = y
```

The same options also need to be configured for the bootloader's makefile under `BOOTLOADER_PROJECT_DIRECTORY/GCC/feature.mk` in bootloader project folder, as shown below:

```
AIR_FOTA_ENABLE  = y
AIR_FOTA_VIA_RACE_CMD_ENABLE  = y
```

For AB156x (4MB flash) and AB157x (4MB flash), because of the limitation of their flash memory size, the ROFS flash partition can and should be updated independently. In other words, the ROFS flash partition should not be updated with other updatable flash partitions that are the Cortex-M4 firmware flash partition, the DSP0 flash partition and the LM flash partition. Refer to Section 2.2.4.2 for more information about the AB156x flash layout. Refer to Section 2.2.4.4 for the more information about the AB157x flash layout.

2.2.2. Dual device FOTA update

Dual device FOTA update is based on the single device FOTA update which is described in section 2.2.1. It can update two Bluetooth-enabled devices at almost the same time.
The two Bluetooth-enabled devices communicate with each other based on MCSync (MultiCast Synchronization). In MCSync terms, the device which connects to the mobile phone directly over Bluetooth is called the agent and the other device which does not connect to the phone is called the partner.
The FOTA application on the mobile phone first pushes the update package to the agent. It then pushes the update package to the partner. After that, it then makes the two devices set the upgrade flags and reboot to update the binary file.
The compile options for the dual device FOTA update are configured in the main project and the bootloader project makefiles (`feature.mk`). One of the makefiles is in the main application folder. The other makefile is in the bootloader project folder that builds the bootloader binary. The dual device FOTA update feature is only enabled when the FOTA options in both makefiles are enabled.
Configure the following options in `PROJECT_DIRECTORY/GCC/feature.mk` of the main application.

```
AIR_FOTA_ENABLE  = y
AIR_FOTA_VIA_RACE_CMD_ENABLE  = y
```

Configure the following options in BOOTLOADER_PROJECT_DIRECTORY/GCC/feature.mk of the bootloader project.

```
AIR_FOTA_ENABLE  = y
AIR_FOTA_VIA_RACE_CMD_ENABLE  = y
```

For AB156x (4MB flash) and AB157x (4MB flash), because of the limitation of their flash memory size, the ROFS flash partition can and should be updated independently. In other words, the ROFS flash partition should not be updated with other updatable flash partitions that are the Cortex-M4 firmware flash partition, the DSP0 flash partition and the LM flash partition. For the detail of the flash layout of AB156x, refer to 2.2.4.2 for the suggested flash layout setting for each chip with FOTA enabled. For the detail of the flash layout of AB157x, refer to 2.2.4.4 for the suggested flash layout setting for each chip with FOTA enabled

2.2.3. FOTA Update with the dongle involved

This section introduces the FOTA update method with the dongle involved by using the PC DFU tool. The architecture is shown in Figure 4. The FOTA update method in the device side is similar to the method mentioned above. Specifically, refer to Section 2.2.1 for details about the dongle and the headset and refer to Section 2.2.2 for the earbuds.

The dongle and the earbuds/headset are updated independently. The users can choose to update the dongle or the earbuds/headset from the PC DFU tool. The tool sends the FOTA package file to the dongle through USB HID.

- If the FOTA package file is for the dongle, the dongle keeps it into its flash.
- If the FOTA package file is for the earbuds/headset, the dongle forwards the file to the earbuds/headset by BLE/SPP connection. The earbuds/headset keeps the file received in the flash.

The FOTA update is triggered when the FOTA package file is downloaded completely and the device reboots to update the new image by the update agent in the bootloader.

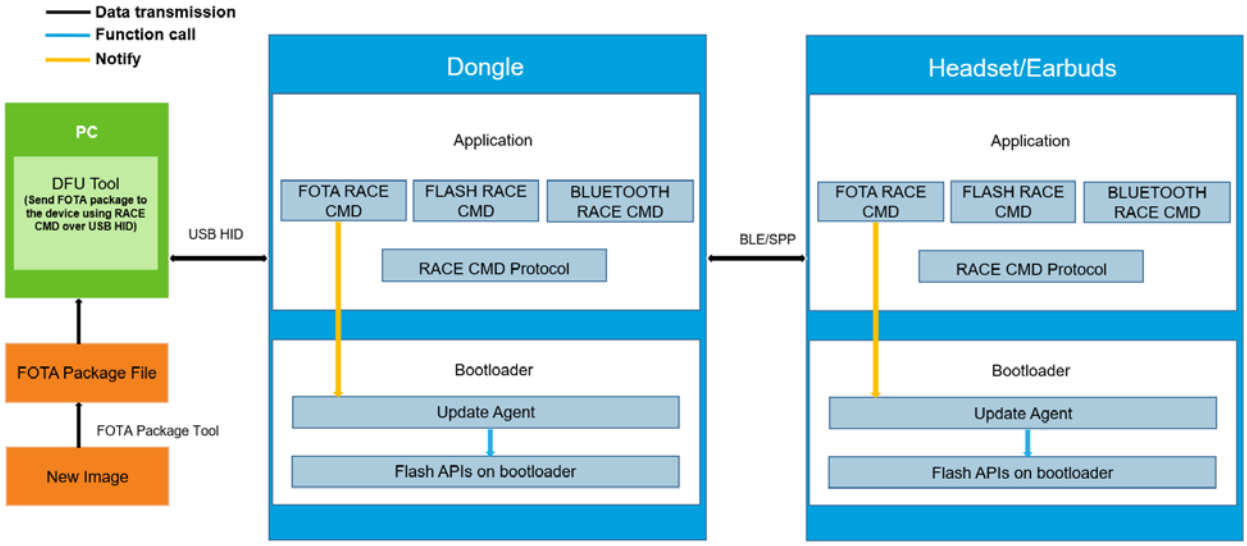

Figure 4. FOTA Update with dongle

2.2.4. Suggested flash layout setting for Airoha IoT SDK for BT Audio

The flash layout is described in the ld file under `<sdk_root>/mcu/project/<chip_name>/apps/<project_name>/GCC`. Refer to the memory layout developer's guide under `<sdk_root>/mcu/doc` for more information.

2.2.4.1. AB155x FOTA buffer partition configuration

The default flash layout with FOTA enabled on AB155x is shown in Figure 5.

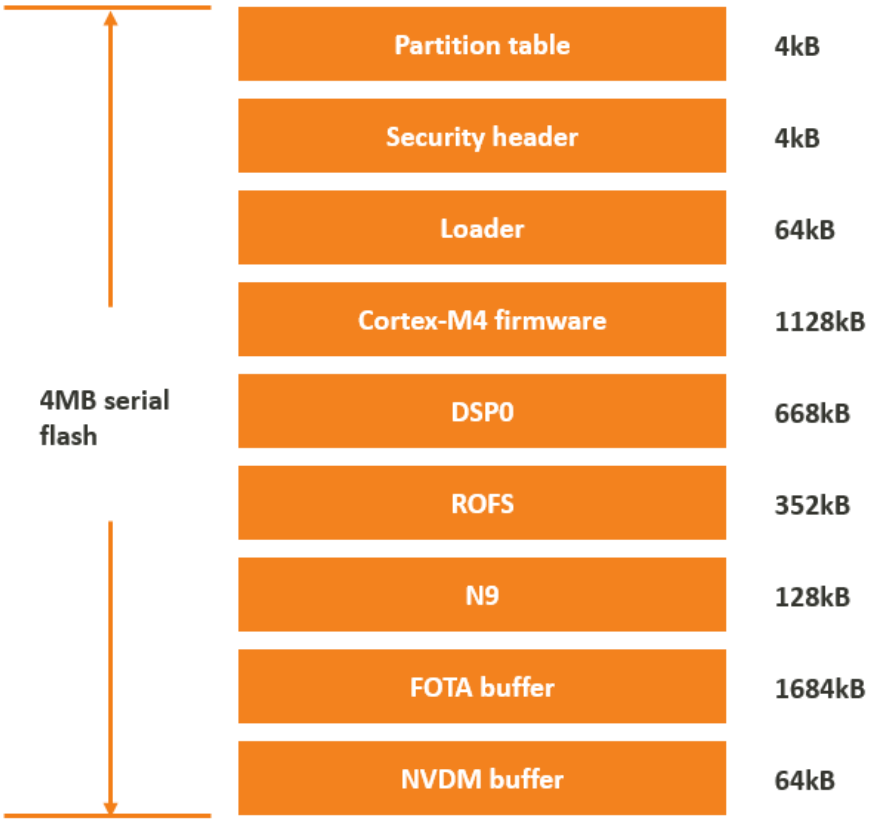

Figure 5. AB155x flash layout default settings

The FOTA buffer partition information is read from the partition table when the FOTA package is stored in the internal Flash. To change the FOTA buffer partition size, changes must be made to the partition table.

- Refer to the Memory Layout Developer's Guide under `<sdk_root>/mcu/doc` for more information.
- Make any necessary changes to the FOTA buffer's start address and length.

If the FOTA package is stored in the external Flash, you must make changes to fota_flash_config_init() to set the customized start address and length for external FOTA buffer.

2.2.4.2. AB1565/AB1568 FOTA buffer partition configuration

The default flash layout with FOTA enabled on AB1565/AB1568 are shown in Figure 6 and Figure 7.

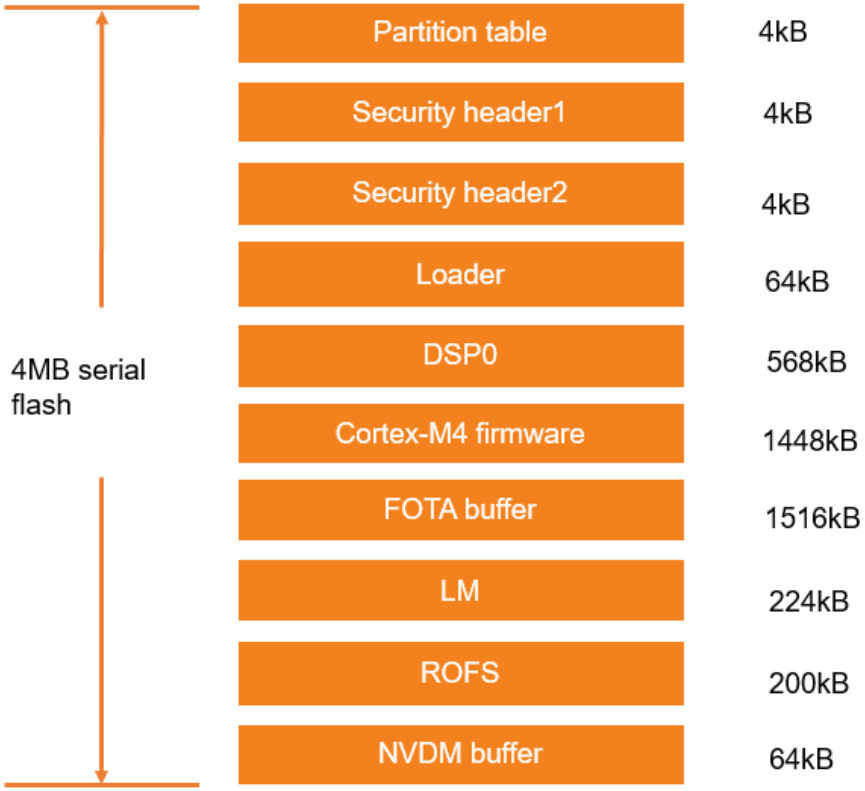

Figure 6. AB1565/AB1568 4M flash layout default settings

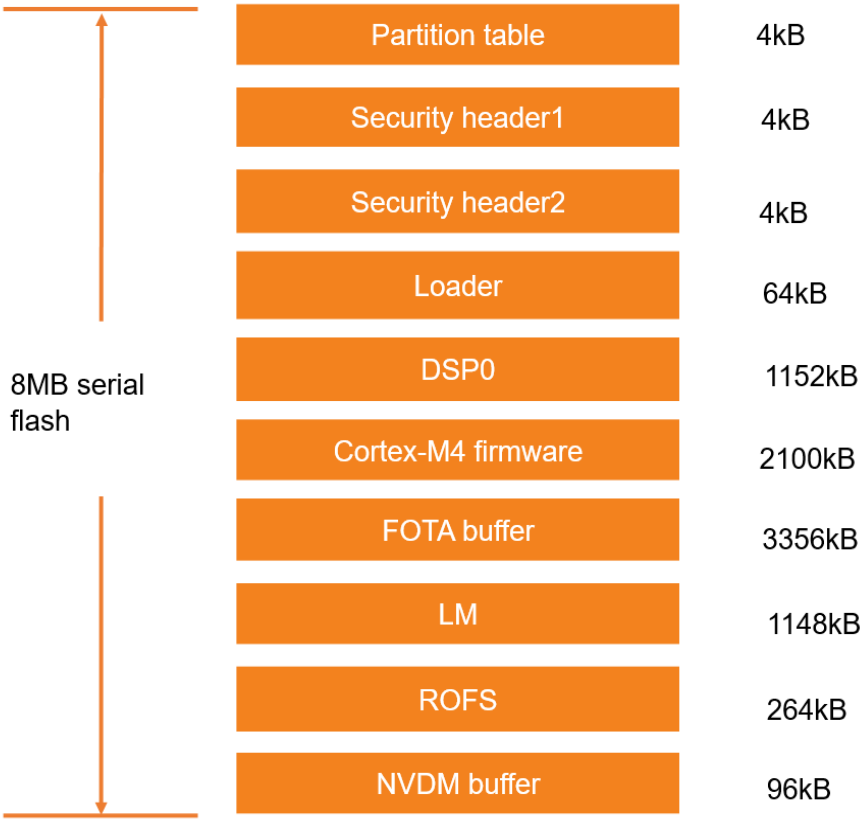

Figure 7. AB1565/AB1568 8M flash layout default settings

To change the FOTA reserved partition size, refer to Section 2.2.4.1.

2.2.4.3. AB158x FOTA buffer partition configuration

The default flash layout with FOTA enabled on AB158x is shown in Figure 8 and Figure 9.

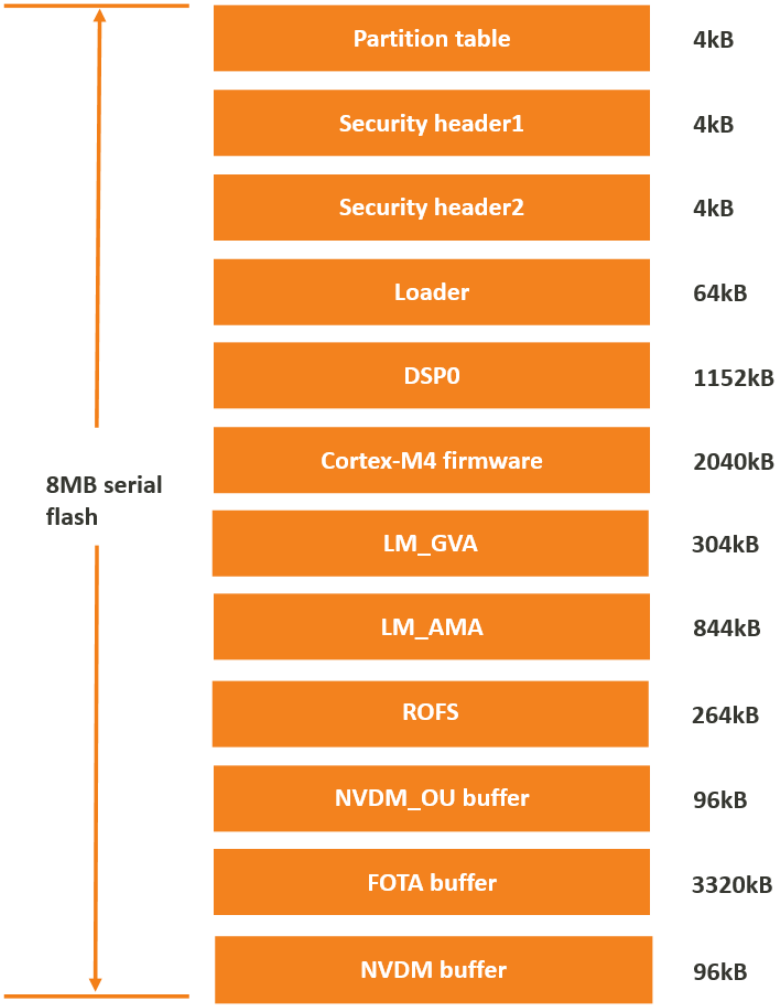

Figure 8. AB158x 8M flash layout default settings

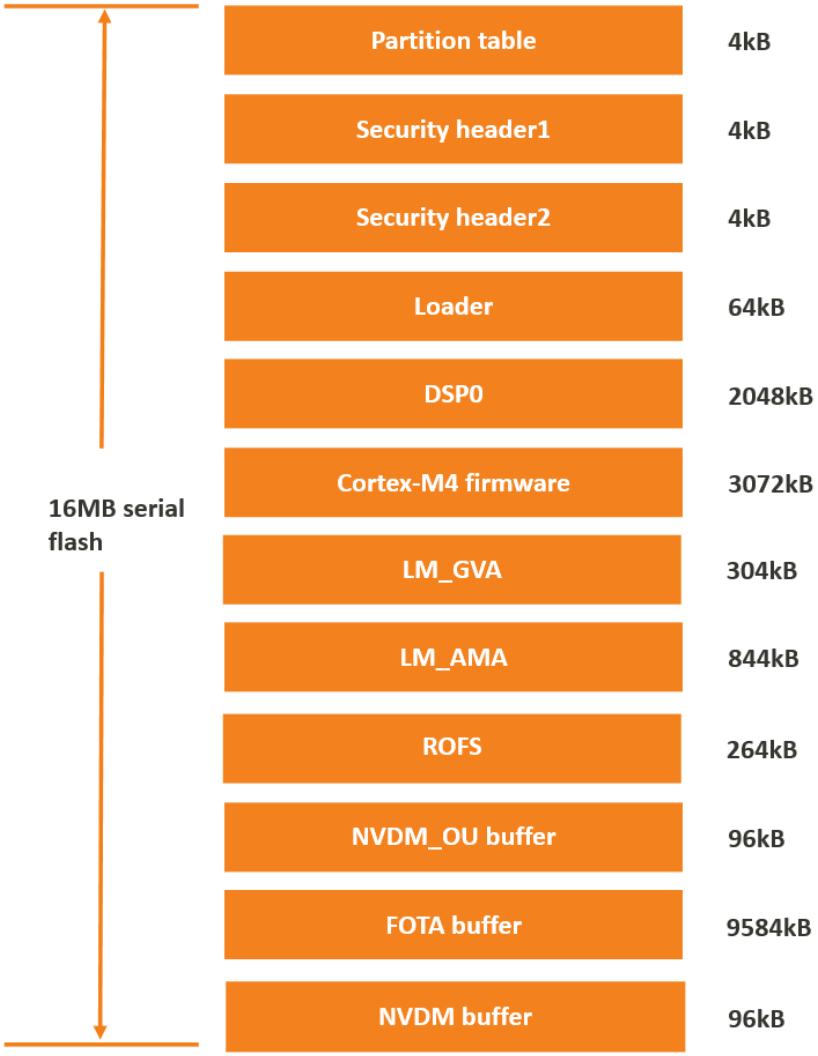

Figure 9. AB158x 16M flash layout default settings

To change the FOTA reserved partition size, refer to Section 2.2.4.1.

2.2.4.4. AB157x FOTA buffer partition configuration

The default flash layout with FOTA enabled on AB157x is shown in Figure 10 and Figure 11.

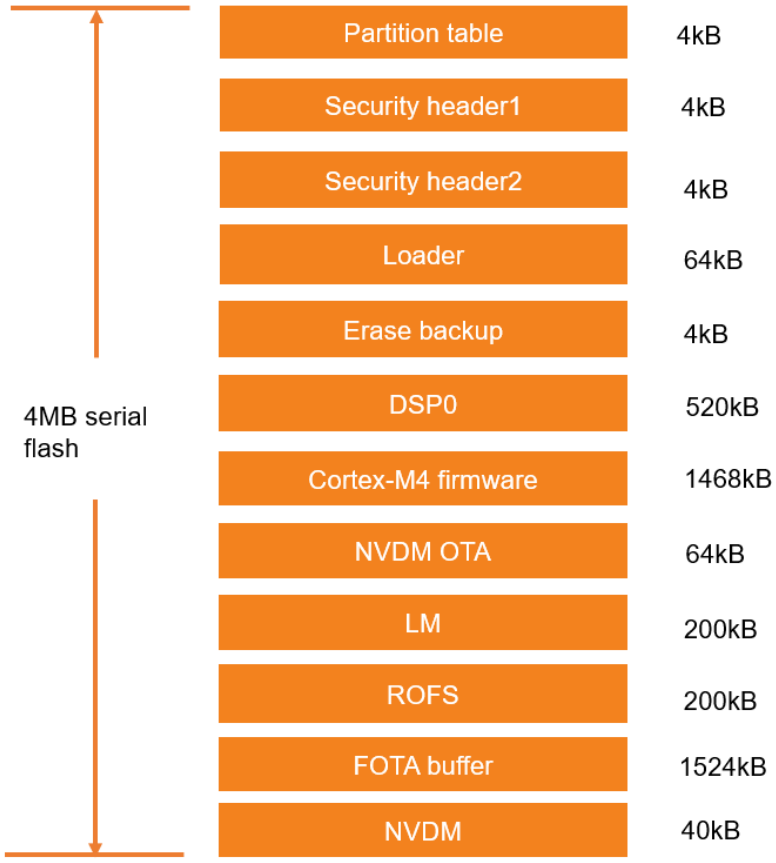

Figure 10. AB157x 4M flash layout default settings

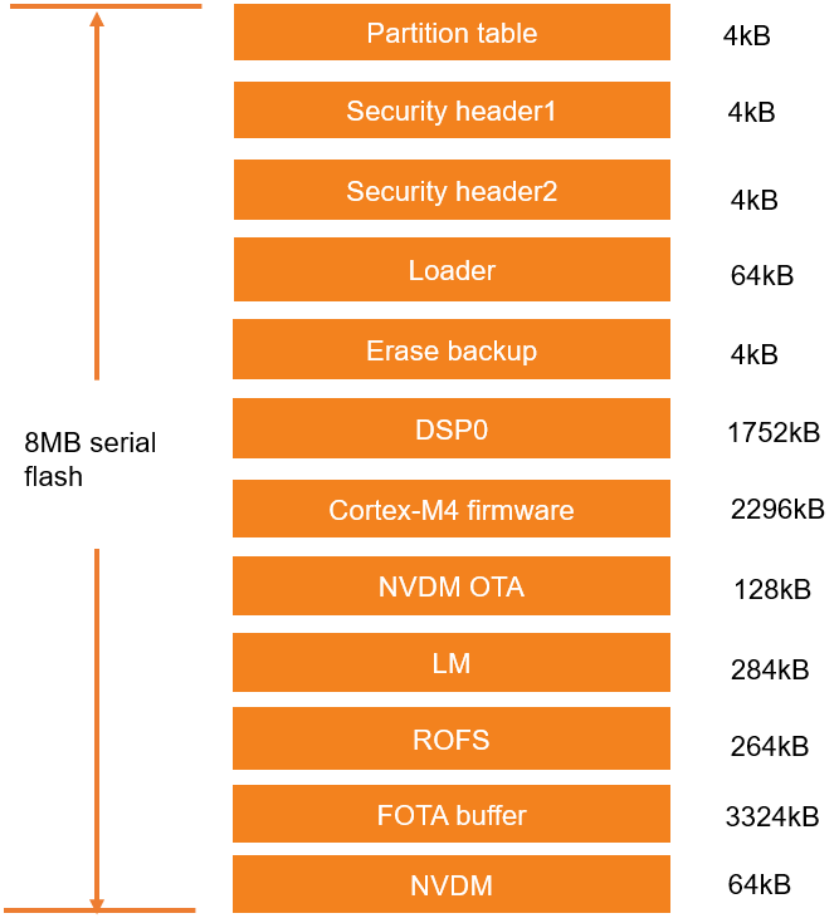

Figure 11. AB157x 8M flash layout default settings

To change the FOTA reserved partition size, refer to Section 2.2.4.1.

2.2.5. Suggested flash layout setting for Airoha IoT SDK for Smart MCU

The flash layout is described in the ld file under `<sdk_root>/mcu/project/<chip_name>/apps/<project_name>/GCC`. Refer to the memory layout developer's guide under `<sdk_root>/mcu/doc` for more information.

2.2.5.1. AM255x FOTA buffer partition configuration

The default flash layout with FOTA enabled on AM255x is shown in Figure 12.

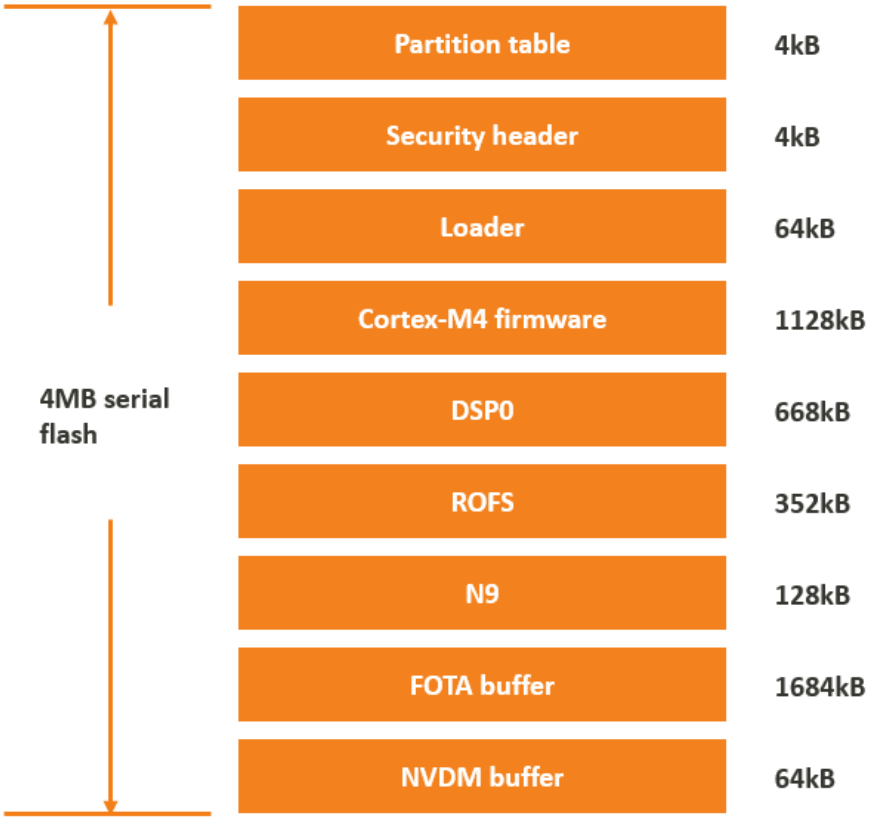

Figure 12. AM255x flash layout default settings

To change the FOTA reserved partition size, refer to the AB155x section.

2.3. External flash support

2.3.1. Upgrading the bin files stored in the external flash

There is the support for upgrading the bin files stored in the external flash. The compile option for this feature is configured in the bootloader project makefiles (`feature.mk`). When generating the FOTA package file using FOTA packaging tool, the TargetAddress element of the bin files stored in the external flash in the Partition XML should be set with SPI_SERIAL_FLASH_ADDRESS.
Configure the following option in BOOTLOADER_PROJECT_DIRECTORY/GCC/`feature.mk` of the bootloader project to support upgrade the bin files stored in the external flash.

```
BSP_EXTERNAL_SERIAL_FLASH_ENABLE = y
```

2.3.2. Store the FOTA package file in the external flash

The FOTA package file can be stored in either the internal flash or the external flash. By default, it is stored in the internal flash. To store it in the external flash, the related compile options should be configured both in the main project makefile (feature.mk) and the bootloader project makefile (`feature.mk`).

Configure the following options in PROJECT_DIRECTORY/GCC/`feature.mk` of the main application.

```
BSP_EXTERNAL_SERIAL_FLASH_ENABLE = y
MTK_FOTA_STORE_IN_EXTERNAL_FLASH = y
```

The same options also need to be configured for the bootloader's makefile under `BOOTLOADER_PROJECT_DIRECTORY/GCC/feature.mk` in bootloader project folder, as shown below:

```
BSP_EXTERNAL_SERIAL_FLASH_ENABLE = y
MTK_FOTA_STORE_IN_EXTERNAL_FLASH = y
```

To configure the FOTA partition in the external flash, call fota_flash_config_fota_partition_in_external_flash() after fota_flash_bootup() both in the main project and the bootloader project.

3. FOTA Update Workflow

This section provides details on how to download the FOTA package and update the firmware in the bootloader.

FOTA resuming is supported during the FOTA download. The SP FOTA application checks if there is continuous data stored from the start address of the FOTA partition. If there is, the application requests the device to send back the sha256 value of the data. This sha256 value is compared to the sha256 value generated from the corresponding part of the FOTA package file to be downloaded. When both sha256 values are the same, it continues sending the remaining FOTA package file to the device. If the sha256 values were different, it would usually require the device to erase the FOTA partition and start sending the complete FOTA package file to the device again. In such a way, if the FOTA download did not finish last time, it can continue downloading the remaining part instead of always downloading the file from the very beginning.

FOTA cancel is also supported during the FOTA download. FOTA download can be cancelled from the SP side or the device side. For dual-device FOTA, FOTA download can only be cancelled from the Agent side. When the FOTA download is cancelled, the SP FOTA application, the Agent, and the Partner terminate the FOTA download flow. To cancel FOTA download from the SP side, the user can press the cancel button in the SP FOTA application. To cancel FOTA download from the device side, the user can call the race_fota_cancel() API. Even if FOTA download is cancelled, the part of the FOTA package file that is already downloaded is stored so the FOTA download process can resume next time.

3.1. FOTA Package Download for the Single Device

When the user triggers the FOTA update process through the phone application, the phone sends the FOTA package to the device through Bluetooth using RACE CMD. Firstly, the phone application sends RACE CMD to query the start address and the length of the FOTA buffer. It then sends RACE CMD to write the FOTA package to the flash directly. When the complete FOTA package is sent, it sends RACE CMD to request an integrity check of the FOTA package that was just received. If the integrity check passes, it sends RACE CMD to trigger a reboot of the device to execute an FOTA upgrade by the bootloader. The FOTA package download process is shown in Figure 13.

The device remains passive during the download and can only receive the RACE CMD from the phone application and execute the operations indicated by the RACE CMD.

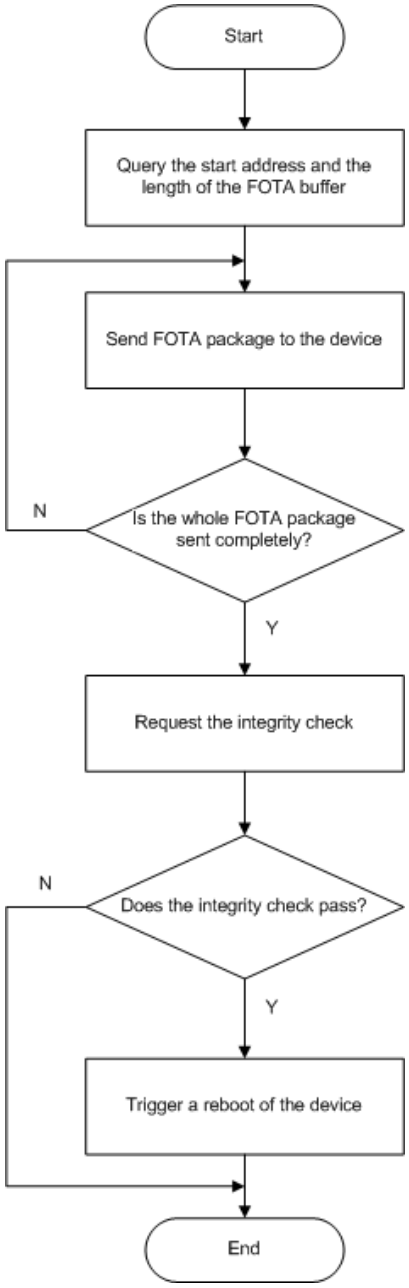

Figure 13. FOTA package download for single device workflow

3.2. FOTA Package Download for the Dual Device

FOTA download for dual device is based on the MCSync (MultiCast Synchronization) connection over which the agent and the partner can communicate with each other. In MCSync terms, the device which connects to the mobile phone directly over Bluetooth is called the agent and the other device which does not connect to the phone is called the partner.

The workflow of FOTA package download for the dual device is shown in Figure 14. The smartphone sends some requests to the agent to gather some necessary information from the agent and the partner over an SPP or BLE connection. The agent gets the corresponding information from the partner over the MCSync connection and returns the information of both the agent and the partner to the smartphone. After that, the smartphone requests the agent and the partner to erase their FOTA sectors. It then sends the FOTA package to the agent.

When the FOTA package is completely sent, the smartphone requests that the agent and the partner switch roles. When they change roles, the agent is the new partner and the partner is the new agent. An SPP or BLE connection is created between the smartphone and the new agent. The smartphone sends the FOTA package to the new agent. When the FOTA package is completely sent, both devices complete downloading the FOTA packages. The smartphone then requests the agent and the partner to reboot at the same time to execute the FOTA upgrade process.

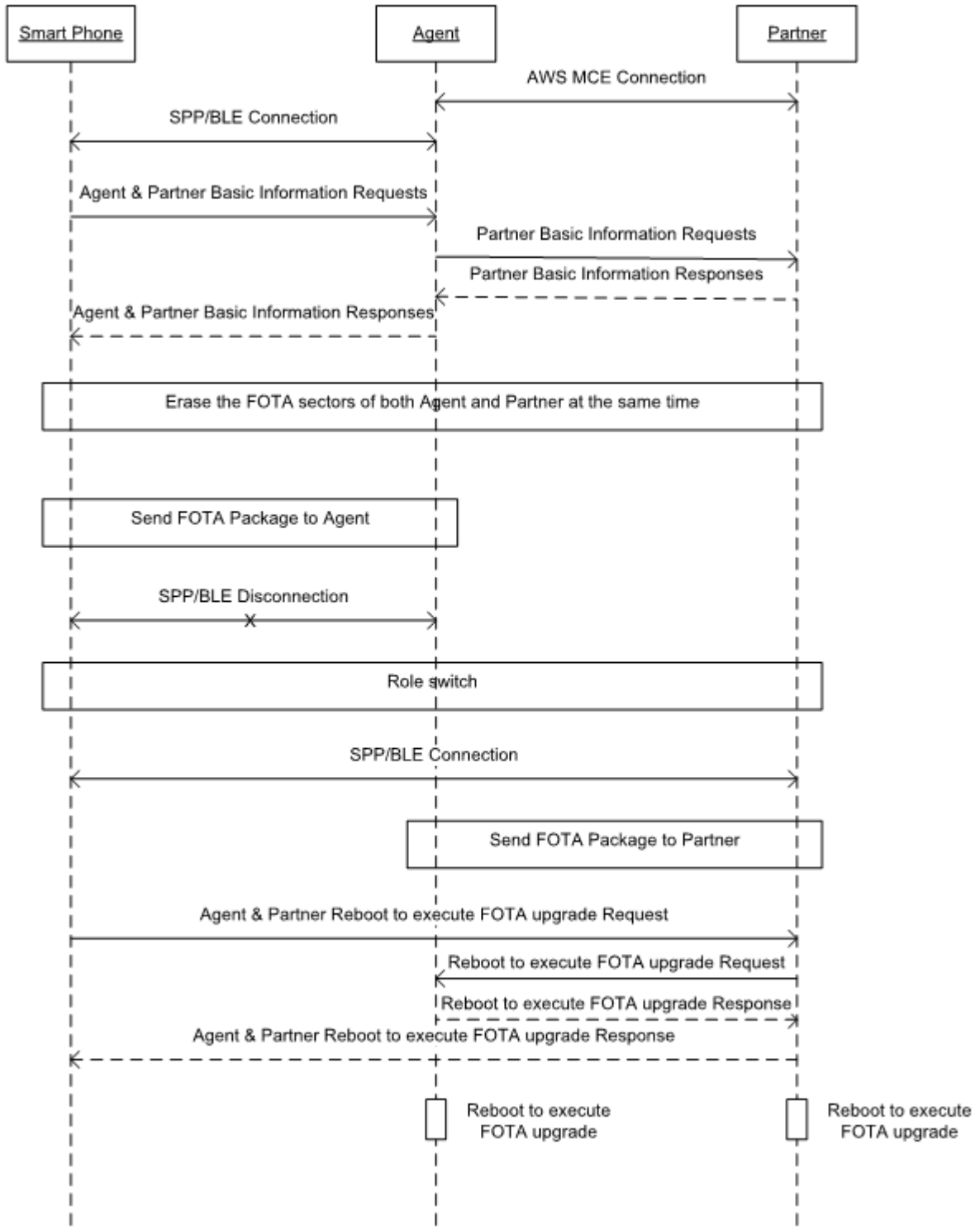

Figure 14. FOTA package download for dual device workflow

3.3. FOTA Upgrade

During the reboot process, the bootloader checks if the last history record is 0x0100, which means that the FOTA upgrade process has been executed one time, but for some reason it is not complete. If the last history record is 0x0100, bootloader starts moving the bin files included in the FOTA package from the FOTA buffer partition to the corresponding flash partitions. If the last history record is not 0x0100, it continues to check if the FOTA upgrade flag is set. If the flag is set, it also starts moving the bin files. If not, it goes into the normal boot up process. The FOTA upgrade workflow is shown in Figure 15.

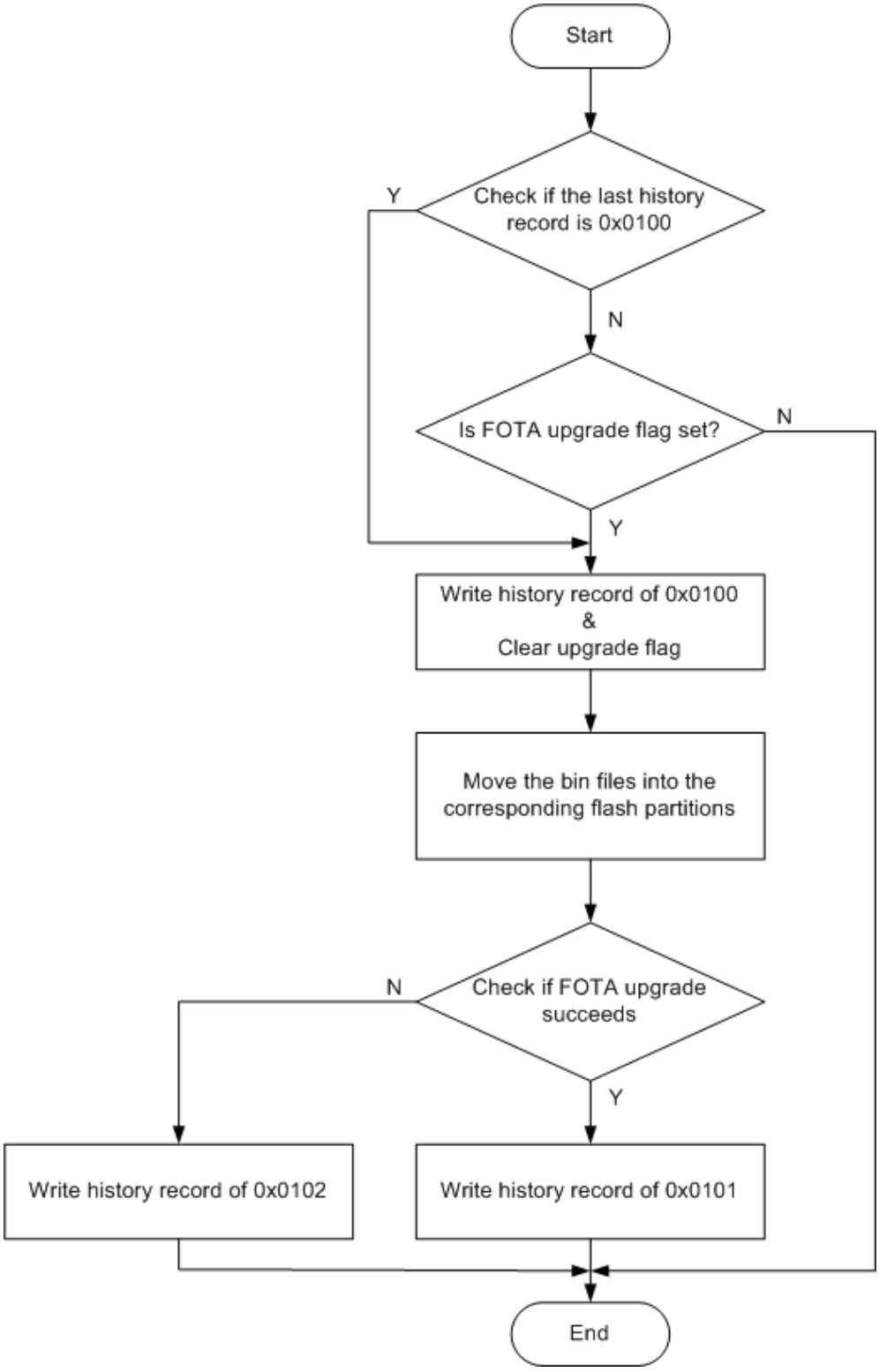

Figure 15. FOTA upgrade workflow

4. Appendix A: Acronyms and Abbreviations

The acronyms and abbreviations used in this developer's guide are listed in Table 2.

Table 2. Acronyms and abbreviations

Abbreviation/Term	Expansion/Definition
FOTA	Firmware over the air, a way to update firmware wireless
NOR flash	NOR flash is a type of non-volatile storage that does not require power to retain data.

5. Appendix B: Constraints

There are some constraints when using FOTA.

1. The agent and the partner must use the same CM4 binary. Otherwise, the RHO may not work.
2. The agent and the partner must be upgraded at the same time to prevent the inconsistent problems. Do not upgrade only one of the earbuds.
3. Do not put the earbuds into the charger case or trigger any unnecessary RHO during FOTA. Otherwise, the FOTA process fails.
4. As mentioned in section 2.2.1 and section 2.2.2, for AB156x(4MB flash) and AB157x(4MB flash), because of the limitation of its flash memory size, the ROFS flash partition can and should be updated independently.