Application Report
SPRACR5–February 2020

AWR2243 Bootloader Flow

ABSTRACT

This application report describes the AWR2243 bootloader flow.

Contents

List of Figures

Trademarks

1 Introduction

AWR2243 Device could be broadly split as two sub-systems:

- Master Subsystem:
  - Bootloader – Responsible for the device initialization, boot time tests, APLL open loop calibration, loading of application images, downloading of images to SFLASH (device management mode, SOP5).
  - Functional firmware – Is responsible for the external host API communication, BSS API handshake, data path (LVDS/CSI2) control, safety and monitoring of the entire device.
- Radar/Millimetre Wave Subsystem:
  - Is responsible for configuring RF/analog and digital front-end in real-time, as well as to periodically schedule calibration and functional safety monitoring. This enables the mm-Wave front-end to be self-contained and capable of adapting itself to handle temperature and ageing effects, and to enable significant ease-of-use from an external host perspective.

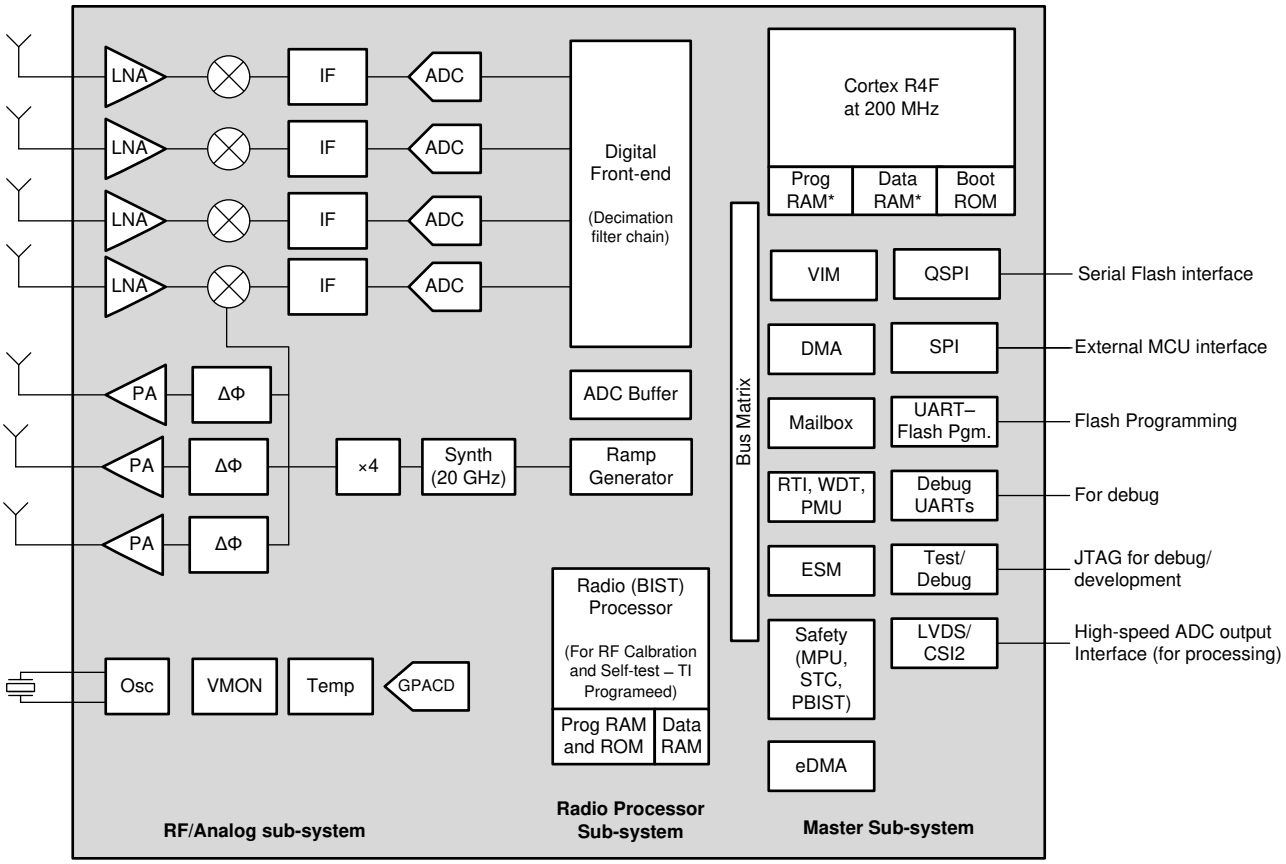

Figure 1. Simplified Representation of AWR2243 Device

- Master subsystem is the first programmable block to get activated after AWR2243 device reset is de-asserted. AWR2243 device's bootloader is hosted in master subsystem's read only memory (ROM) and takes control immediately.
- From this point onwards, the AWR2243 bootloader can operate in two modes: Flashing and Execution modes.
- Bootloader looks for the state of Sense On Power (SOP) I/Os (SOP lines – driven externally for choosing the specific mode).

Table 1. Sense On Power (SOP) Lines and Boot Modes

SOP2	SOP1	SOP0	Bootloader mode & operation
0	0	1	Functional Mode Primary deployment mode. After the patches are loaded (over SFLASH or SPI), the functional firmware executes and the device is controlled by commands over SPI. The ADC data is available on the high speed interface of choice (LVDS/CSI2).
1	0	1	Device Management Mode Flash programming mode. The images (patches) are downloaded onto the SFLASH using a flashing utility that transfers the images over the UART.

- Device Management (Flashing) Mode of the bootloader allows an external entity to load the customer application image to SerialDataFlash (SDF).

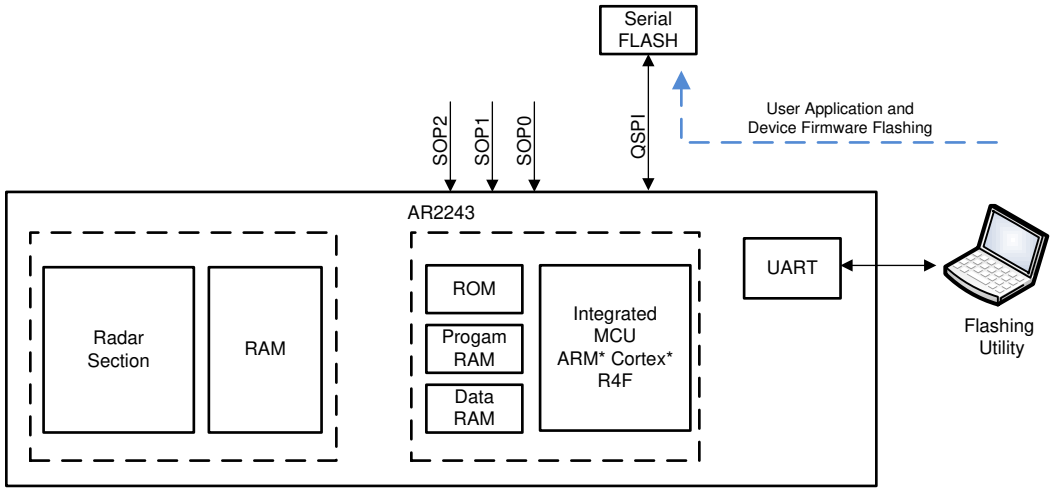

Figure 2. Flashing Mode of Bootloader

- Execution (or Functional) Mode of the bootloader has two boot modes:
  - Boot Mode – SFLASH (Development Phase)

    If the presence of a Serial Flash is detected with a valid image, the bootloader relocates the image stored in SDF to R4F and Radar section memory subsystems. Towards the end of this process, bootloader would pass the control MSS Functional firmware.

    The SFLASH is present only in development versions of the silicon where the functional firmware (MSS and Radar section) does not execute from ROM, hence, it is a large image size.

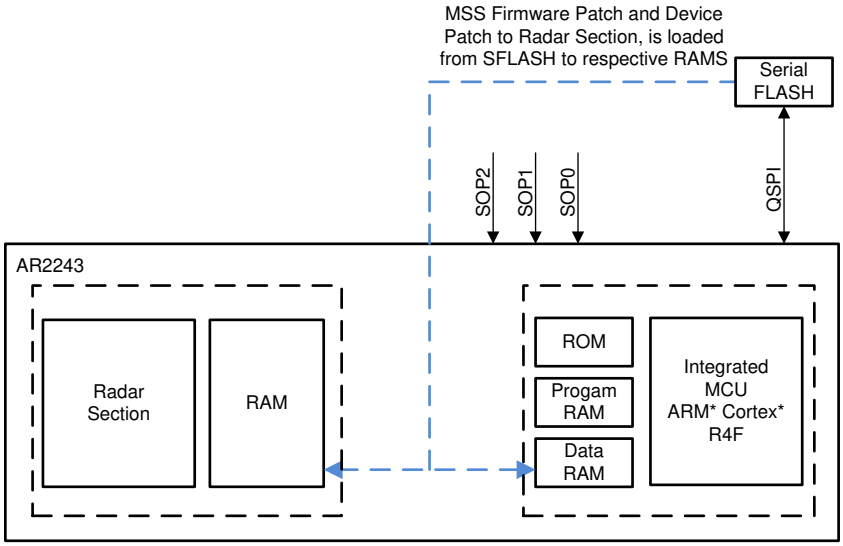

Figure 3. Execution Mode of Bootloader (image load from SFLASH)

  - Boot Mode – SPI (Deployment Phase)

    If the serial flash is not detected or a valid image is not detected in the serial flash, the bootloader loads the images (patches) to the respective memories of the MSS R4F and Radar section subsystems by receiving the data from an external host over SPI. Towards the end of this process, the bootloader would pass the control MSS Functional firmware.

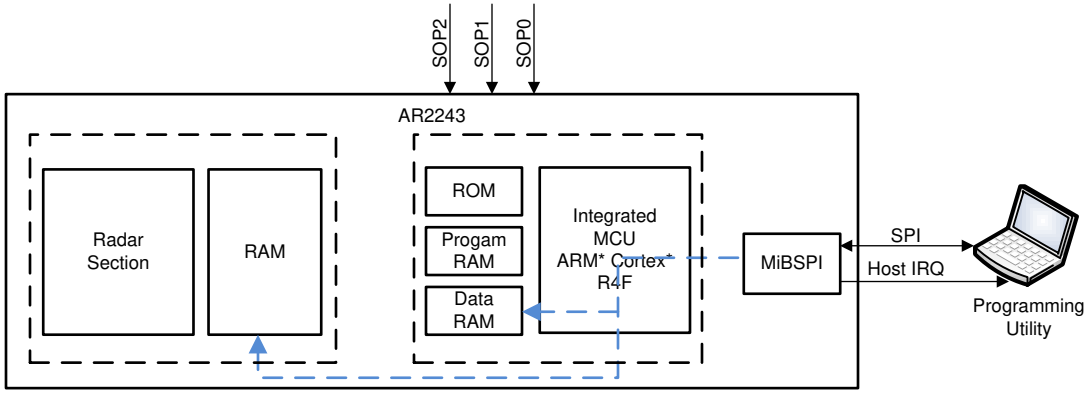

Figure 4. Execution Mode of Bootloader (image load over SPI)

2 Basic Bootloader Flow

A high level bootloader operation could be split into three phases:

- Device Initialization: bootloader uses “Built In Self Test” (BIST) Engines for hardware diagnostics (for example, RAM tests)
- Sets up the root clock by starting the APLL
- Checks SOP lines to proceed with either the Flashing or Execution Modes

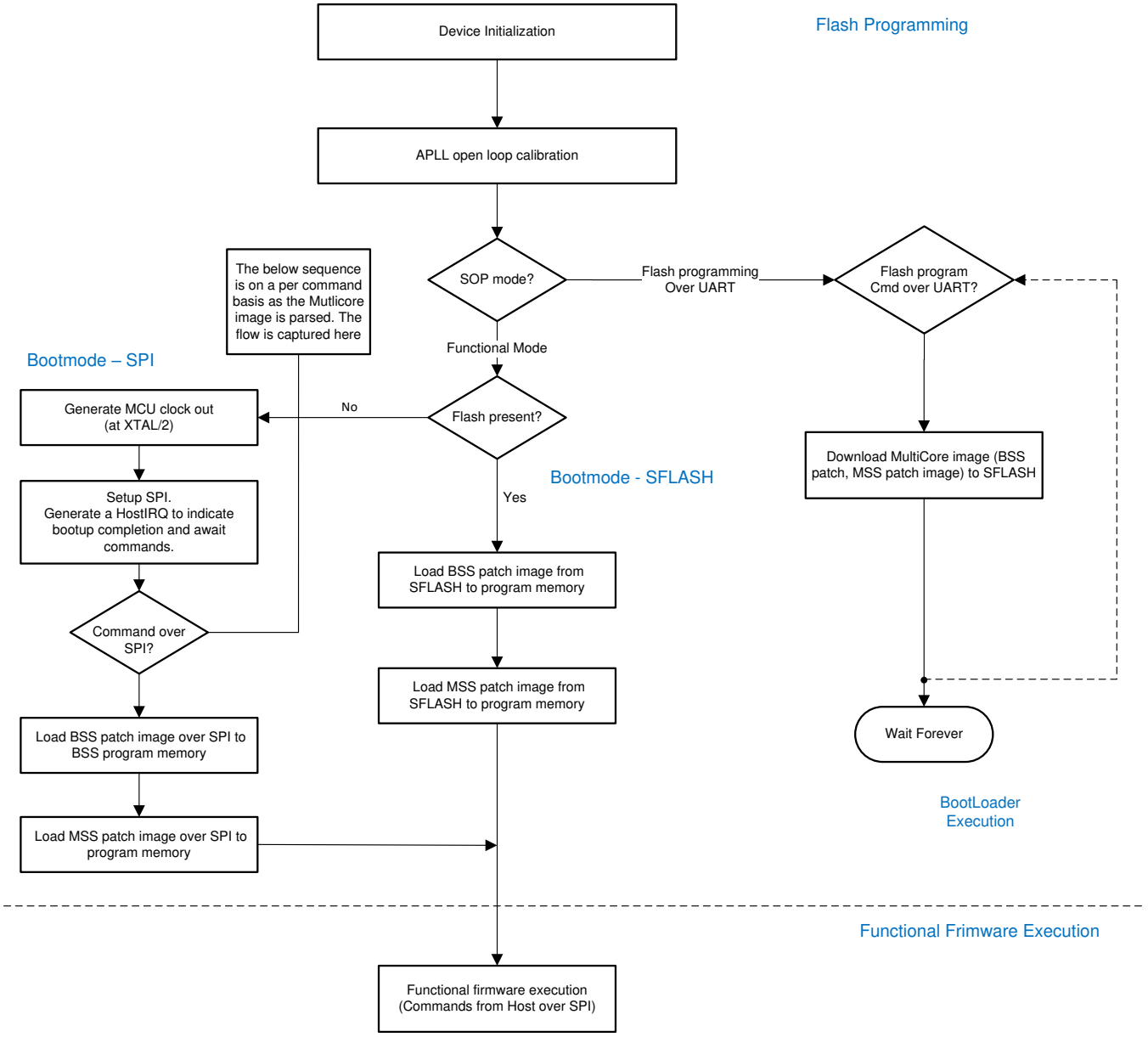

Figure 5. Basic Bootloader Flow Chart

2.1 Bootmode – SFLASH

2.1.1 Image Load Sequence

In functional mode, the bootloading of an image from the SDF is the first bootmode attempted by the bootloader. This involves the following steps:

- Pinmux the QSPI pins of AWR2243 device
  - QSPI[0]: Ball R11
  - QSPI[1]: Ball P9
  - QSPI[2]: Ball R12
  - QSPI[3]: Ball P10
  - QSPI_CLK: Ball R10
  - QSPI_CS_N: Ball P8
- QSPI is setup to operate at (System clock/5) = (200/5) = 40 MHz
- The SFLASH Discoverable Parameters (SFDP) command is issued to retrieve the JEDEC compliant response that includes information regarding the SFLASH capabilities and command set. On successful reception of the SFDP response, the information is used to communicate with the SDF and further interpret the contents and load the images.

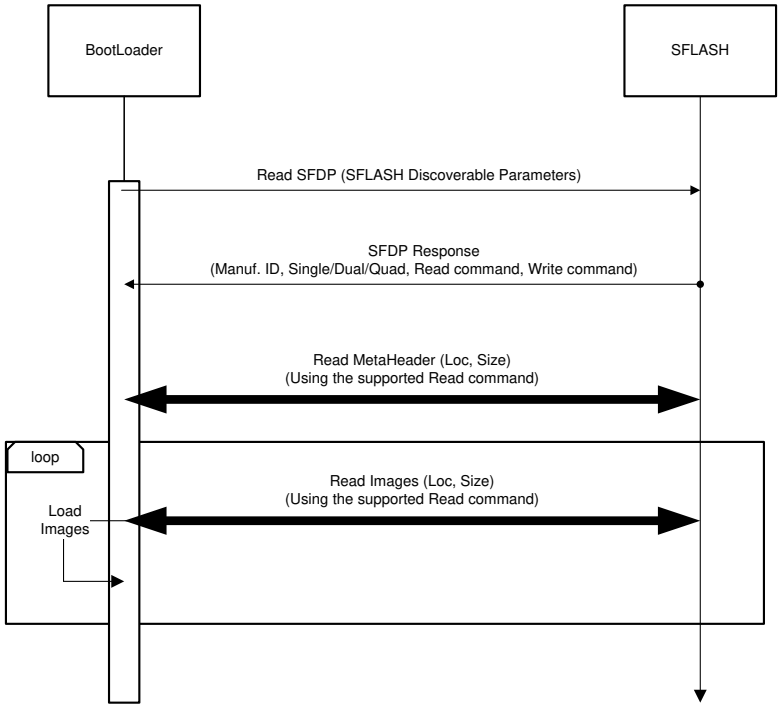

Figure 6. Image Load Sequence

Key Points:

- The ROM bootloader performs the read from the SDF based on the highest capability mode (Quad/Dual/Single) as published by the SDF in response to the SFDP command.
- For SDF variants that support Quad mode, the Quad mode commands will be issued and in case the Quad Enable (QE) bit is not set, the communication will fail. In such cases, the load flow expects that the “Quad Enable (QE)” bit in the SDF is already set.

2.1.2 ROM Assisted Image Download Sequence

The ROM assisted image download sequence is entered by placing the device in flashing mode. Refer to section 3, for further details on the handshake with an external host to receive the image. The communication with the SDF is depicted in Figure 7.

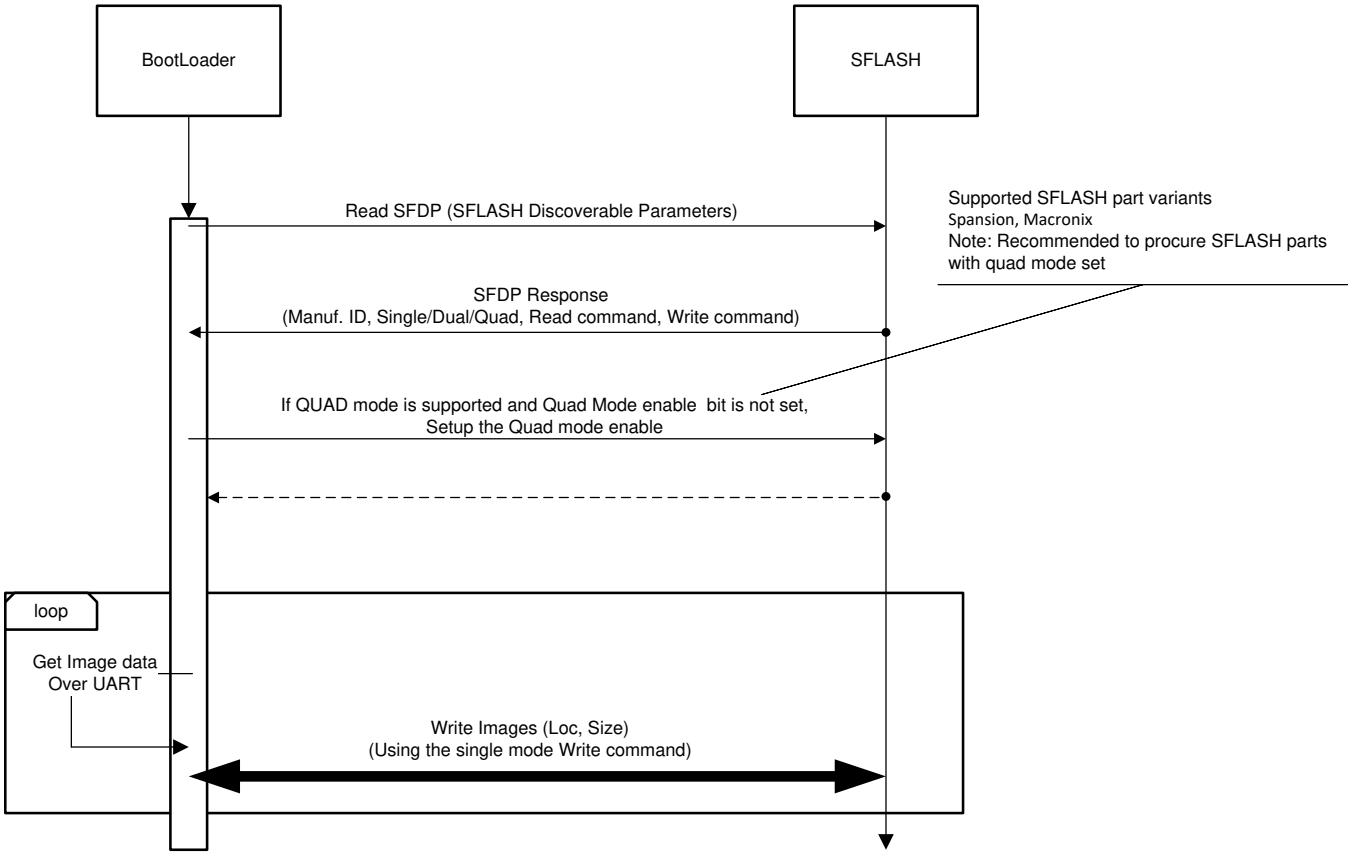

Figure 7. ROM Assisted Image Download Sequence

Key Points:

- The ROM assisted download should work with all flash variants that allow for “Memory mapped mode” and “Page program command (0x2)” with 1 dummy byte and 24bit addressing.
- The setting of the “Quad Enable” bit varies from one SDF vendor to another. The ROM bootloader supports setting the “Quad Enable” bit for Spansion and Macronix variants (certain specific part variants only) in this flow.

In addition to a checksum based integrity check for every packet received over the UART, a CRC32-based integrity check is performed over the complete image. The CRC32 is computed incrementally as the packets are received and written to the SDF.

2.2 Bootmode – SPI

In functional mode, if no valid image header is found in the predefined SDF location, the bootloader will enter the SPI based bootloading mode.

This involves the following steps:

- Pinmux the SPI pins - [SPI_MOSI: Ball R8, SPI_MISO: Ball P5, SPI_CLK: Ball R9, SPI_CS_N: Ball R7 and SPI_HOST_INTR: Ball P6 of AWR2243 device]
- Follows the “Communication Protocol” as defined in the Radar Interface Control document to communicate with an external host to receive the images to be loaded as message packets over SPI.
- Once the load of all images is complete, the patches to the ROM get applied and then the functional firmware begins execution that could then receive control API commands over SPI and send out the data over the high speed interfaces (LVDS/CSI2).

The handshake with the external host is as depicted in Figure 8.

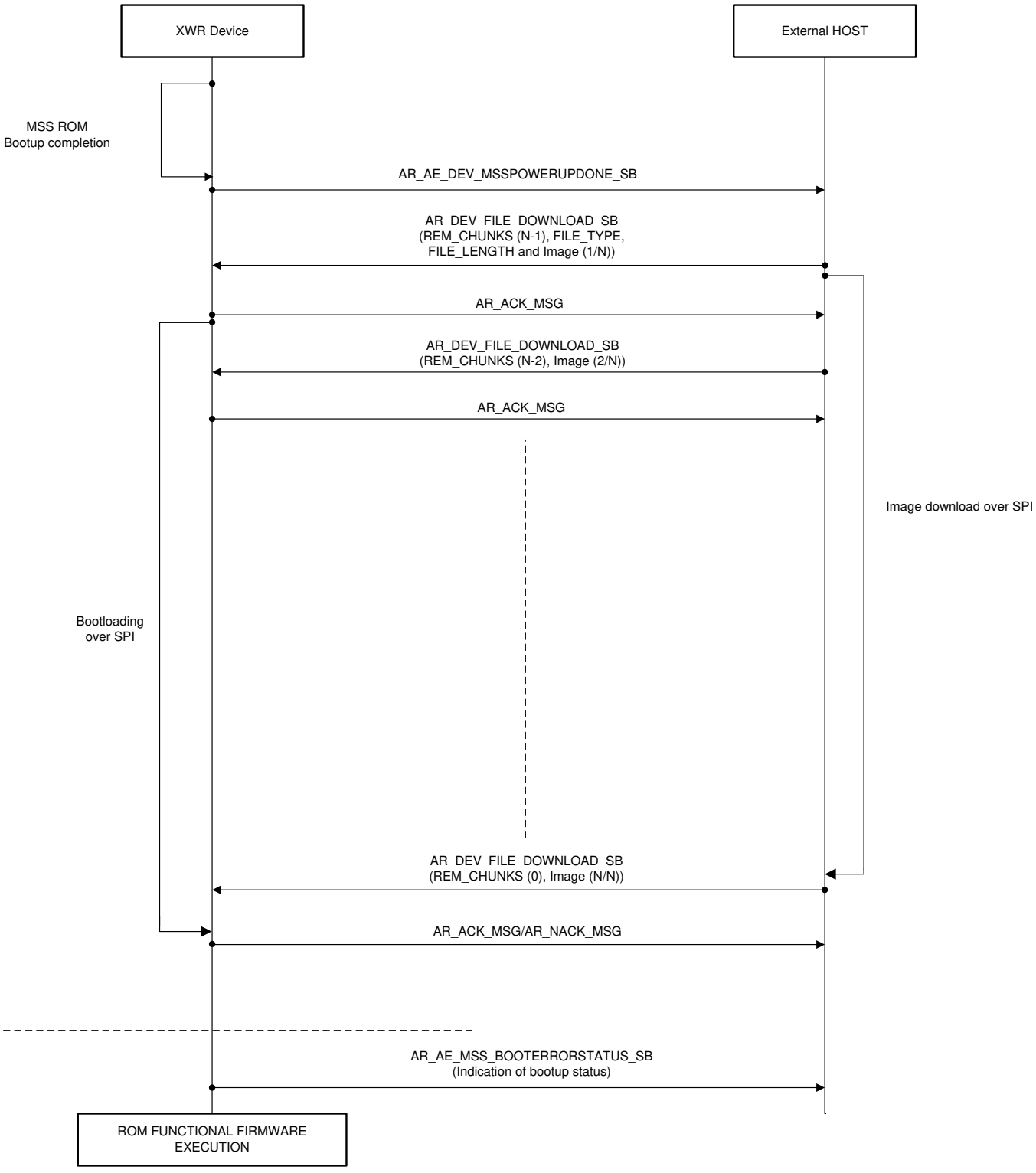

Figure 8. Bootmode - SPI

3 Programming Serial Data Flash Over UART (Bootloader Service)

TI's AWR2243 device supports interfacing with an SDF to obtain the images to be loaded as a part of the bootloading process. During the pre-production phase, the ROM is eclipsed and the majority of the execution of the MasterSS and RadarSS firmware happens from the RAM. Hence, the image sizes being considerably large, are loaded faster when present in the SDF with a QSPI interface.

The flash programmer connects to the device over UART. Specifics are as follows:

- MSS_UARTA [RX: Ball N5 and TX: Ball N6 of AWR2243 device]
- Baud Rate: 115200
- Maximum Data Chunk Size: 240 bytes

NOTE: Commands 'Write File to SFLASH' and 'Write File to SRAM' support a maximum data chunk size of 240 bytes only.

The file is split into N commands where:

N = (file size/240) + ((file size%240) ? 1 : 0)

3.1 File to Download

One or more of the following binaries comprises of the files to be downloaded:

- RadarSS Image (Complete firmware (pre-production) Or Patch if any (production devices))
- MasterSS Image (Functional firmware (pre-production) Or Patch if any (production devices))

In AWR2243 ES3.0 (production variant), these two images are combined into a single MetaImage file which must be flashed to the device.

TI's DeviceFirmwarePackage (DFP) for XWR12xx would include the necessary files to be downloaded to both the MasterSS as well as RadarSS.

TI's mmWaveSDK package for XWR14xx would include the "Image Creator" utility that would help constructing the complete image with the above listed components for use with XWR14xx devices.

3.2 Flash Programming Sequence

1. Boot the device in SOP 5 mode (see Table 1)
2. Open the "UniFlash" tool. [As listed in mmWaveSDK]
3. Connect to the device over the UARTA com port. (the device expects a UART break signal – this is generated by the UniFlash tool)
4. Flash the desired images <META_IMAGE1/ META_IMAGE2/ META_IMAGE3/ META_IMAGE4>

NOTE: With AWR2243 ES3.0, the user must select a single MetaImage file (META_IMAGE1).

3.3 Supported Commands and Format

Table 2. Supported Commands and Format

Command	Command Id	Description	Fields
PING	0x20	The device responds with ACK	
OPEN FILE	0x21	Command that gives details about the type of file being downloaded	File Size: Total File size being downloaded. File Type: META IMG1(4), META IMG2(5),META IMG3(6), and META IMG4(7) Storage type: 2- SFLASH and 4- SRAM
WRITE FILE to SFLASH	0x24	Command that gives the content of the file to write to SFLASH	
WRITE FILE to RAM	0x26	Command that gives the content of the file and the file is directly written to RAM	
CLOSE FILE	0x22	Command that indicates the end of file download	Storage type: 2- SFLASH and 4- SRAM
GET STATUS	0x23	Command that requests the status of the previous command. The device responds with the status of the previous command issued	
ERASE DEVICE	0x28	Command to erase the contents of the SFALSH	
GET VERSION	0x2F	Command that requests the version of the ROM. Device responds with the version information	
ACK response	0xCC	Response from the device.	

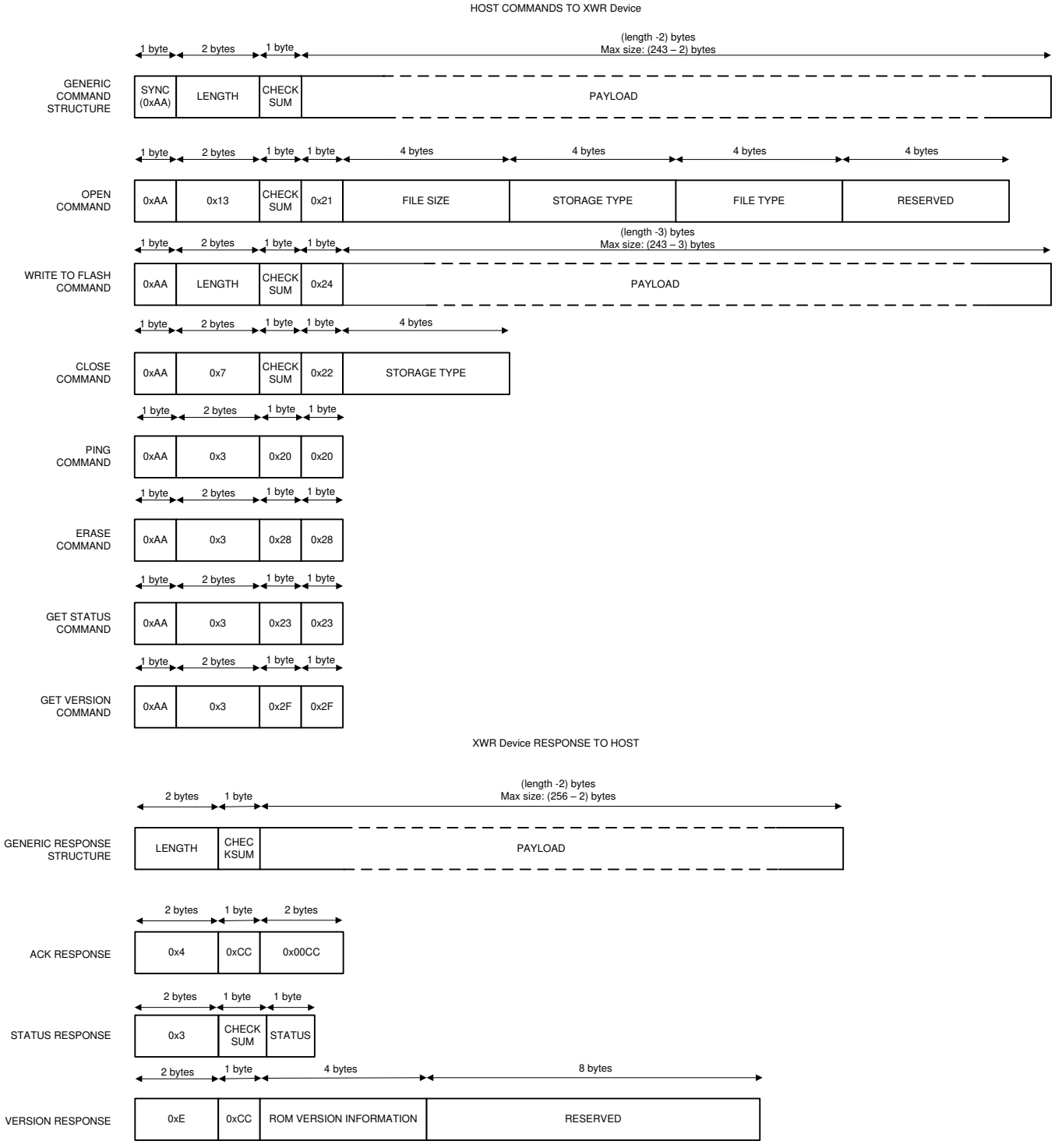

Figure 9. HOST ↔ XWR Device UART Communication

The 8-bit checksum in each UART command is a simple unsigned sum of the unsigned values of all the bytes of the payload of the command, where only the least-significant 8 bits of the sum are kept. For example, the pseudo-code to calculate the checksum would be:

```
checksum = 0
```

for each byte in the payload, checksum = (checksum + (unsigned) byte) AND (0xFF).

The STATUS RESPONSE returned from the device is the bootloader error status based on the last actionable command executed. Actionable commands include OPEN, WRITE TO FLASH, CLOSE, and ERASE. Status commands like PING, GET STATUS, and GET VERSION do not affect the error status reported in the STATUS RESPONSE. The possible returned STATUS values are as follows:

- 0x00 = INITIAL_STATUS (before any actionable commands are issued)
- 0x40 = SUCCESS
- 0x4B = ACCESS_IN_PROGRESS
- Any RESERVED fields in commands sent from the host should be set to 0x0

3.4 Flashing Sequence

Figure 10 shows the flash programming sequence. The initial handshake starts with a UART break issued by the external host. This break is followed by the command sequence in Figure 10. The bootloader uses a command-response protocol. So the host should wait after sending each command until an ACK response is received from the device. Note that the GET STATUS command is unique in that it returns only the STATUS RESPONSE message without sending an ACK response.

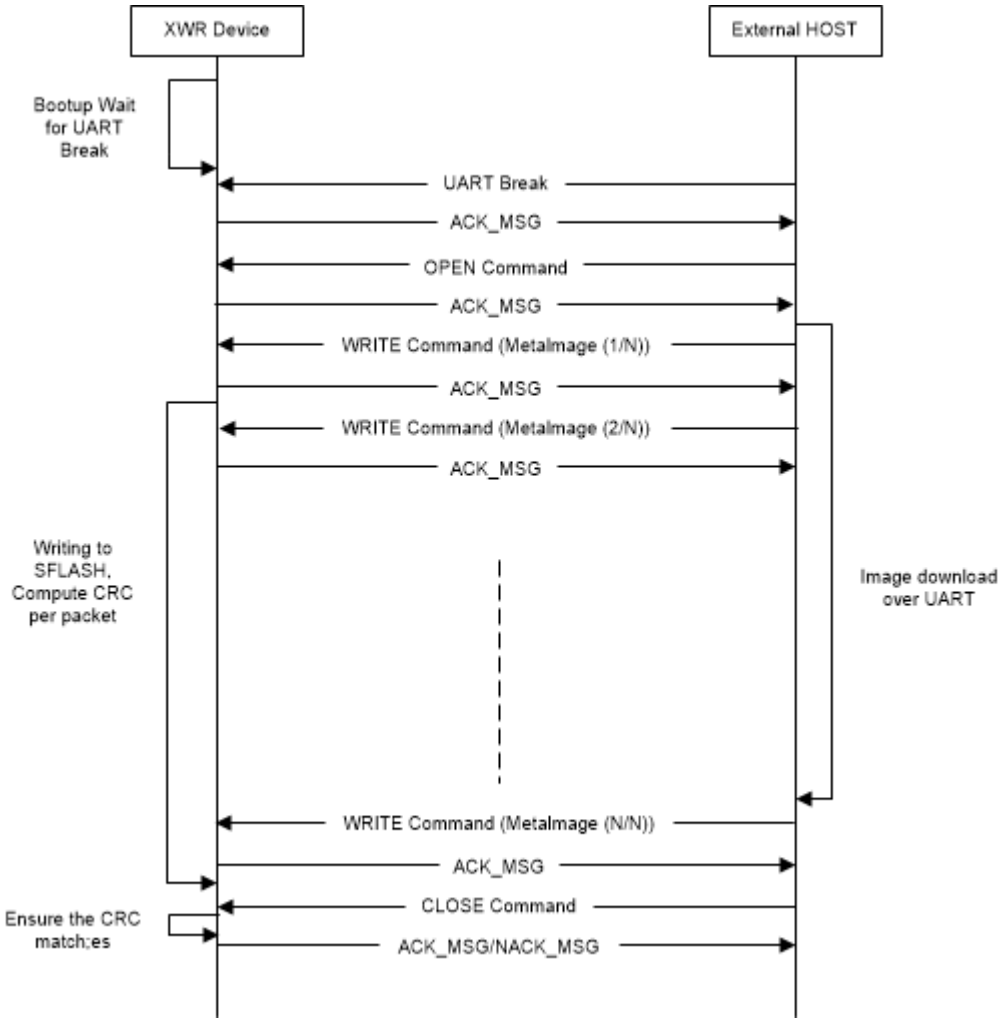

Figure 10. Flashing Sequence

Key Points:

- The host processor needs to split the each file/image into smaller chunks and send each chunk in a WRITE TO FLASH command. The LENGTH field of the command should be set to the total payload size (which includes the image data chunk and 1 byte for the 0x24 opcode) + 2. The max chunk size is243 - 3 = 240 bytes.
- The byte order for the words in the UART commands is big-endian (i.e. transmit most-significant byte first). The BSS/MSS/Config images should be transmitted as bytes in the order they are in the binary files.
- The ERASE command is not required but can be used to make sure the entire SDFLASH is cleared before the new images are written.
- The GET STATUS command is not required but can be used to check status of device.
- The GET VERSION command not required but can be used by the host processor to allow it to operate differently depending on the device silicon version.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265