Application Note

mmWave Image Creator User Guide

This document outlines the tools used for image creation to work with mmWave devices. mmWave SDK may contain high level scripts that execute all the steps mentioned in this document. The scope of this document is to provide an understanding of the flash image needed for mmWave devices.

Contents

Contents **1**

List of Figures **1**

1 Overview **2**

2 Application Image **3**

2.1 RPRC Image format 3

2.2 Multi core Image generation from the application RPRC images 4

List of Figures

Figure 1 - RPRC Format 3

Figure 2 - Meta Image format 4

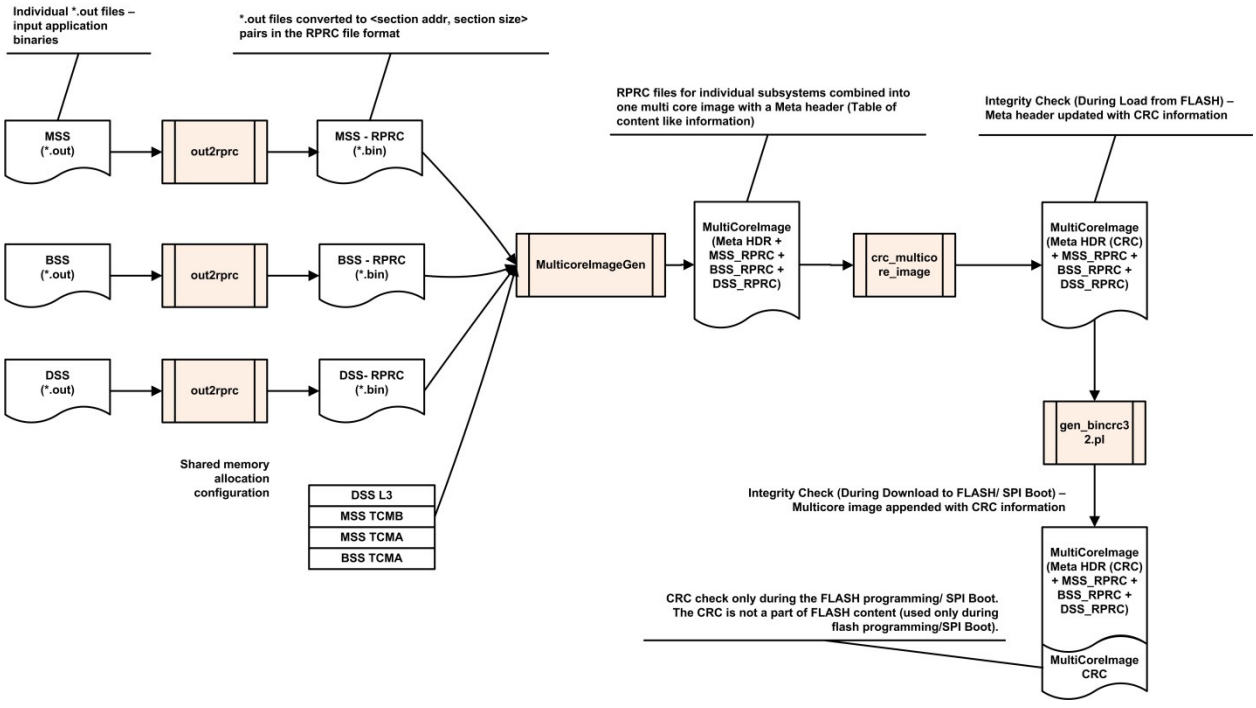
1 Overview

Application Image generation is two-step process:

- **RPRC format conversion**
  - Firstly, application executable has to be converted from ELF/COFF format to custom TI RPRC image format. For more information on RPRC format, refer to the section 2.1 of this user guide.
- **Multicore Image file generation**
  - The Application Image interpreted by the bootloader is a consolidated Multicore image file that includes the RPRC image file of individual subsystems along with a Meta header. The Meta Header is a Table of Contents like information that contains the offsets to the individual subsystem RPRC images along with an integrity check information using CRC. For more information on Multicore Image format, refer to the section 2.2 of this user guide.
  - In addition, the allocation of the shared memory to the various memories of the subsystems also has to be specified. The bootloader performs the allocation accordingly. It is recommended that the allocation of shared memory is predetermined and not changed dynamically.

An overview of the image creation process is depicted below. The key inputs are:

- the executables of the individual subsystems
- the choice of shared memory allocation



2 Application Image

2.1 RPRC Image format

Each of the core Images have to be converted into the RPRC format. The below Figure 1 shows the RPRC format:

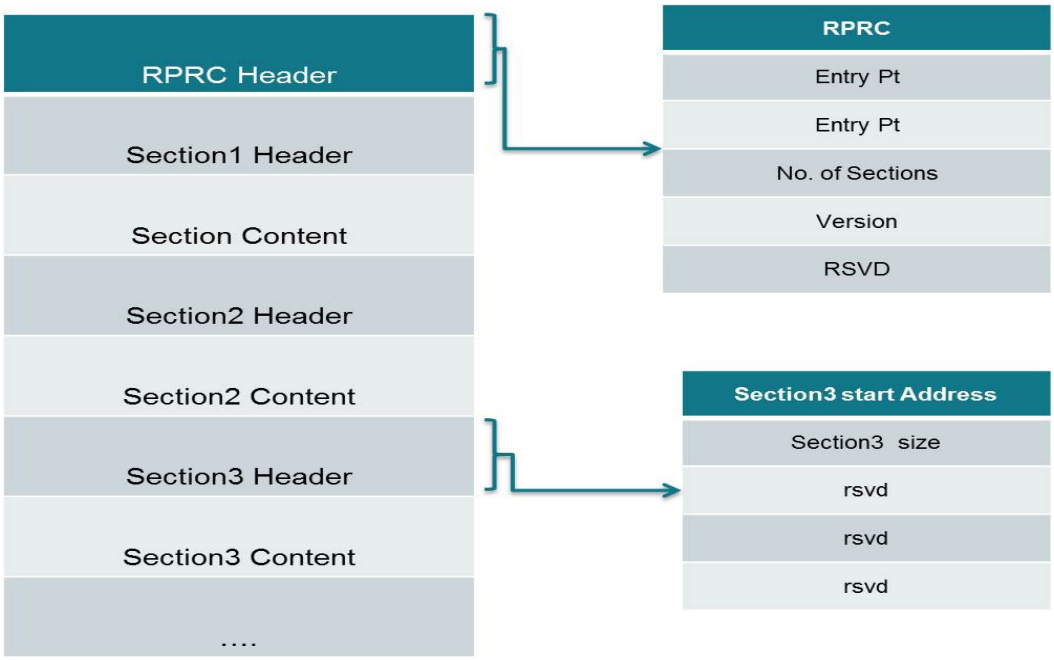


Figure 1 - RPRC Format

As shown in above diagram, RPRC file has one file header and multiple sections.

- File Header contains Magic string as start marker, Entry Point, Number of Sections and One reserved word.
- File header is followed by multiple Sections. Each Section has section header and section data.
    - The Section Header has five words: Load address, size and three reserved words. Address specifies the destination address of that section and size specifies section size in bytes of the Section Data that follows. The whole section has to be copied to the load address. RPRC sections are generated for each elf section that is a part of the elf executable.

Convert the different application image files (ELF/COFF) for various cores to RPRC image using the following command:

$out2rprc.exe <App_In_name(elf or coff)> <App_out_name>

2.2 Multi core Image generation from the application RPRC images.

Multi core image format is as shown below:

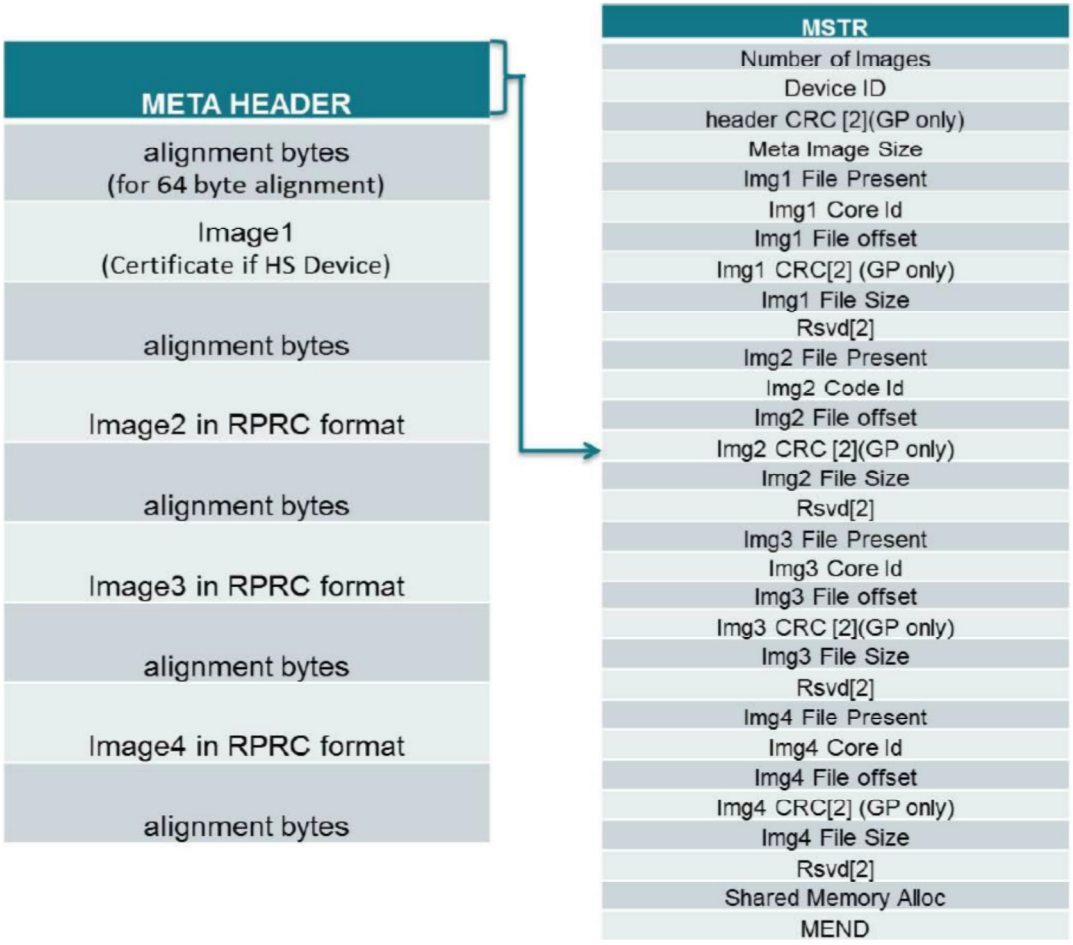


Figure 2 - Meta Image format

Meta Header is of variable length and depends on the number of files included.

- First word (MSTR in ASCII) is a magic string as start marker.
- Second word is the number of RPRC image files included in the multi core image file.
- Third word is the Device Id
- Followed by two words for 64-bit CRC.
- The next word is the total image size.
- After this, follows the eight word structure of image present details, magic word, the offset for RPRC Image i.e. the starting byte of RPRC image, CRC and the image size.
    - For a given structure, RPRC Image starting at the given offset will be loaded on the Core as interpreted from the Core ID.

There computed CRC should be compared with the Meta header CRC to validate header before parsing the RPRC images. String MEND in ASCII is used as last word at the end of header string.

The MSS boot loader supports download of a special file called the CONFIG file, which enables shared memory allocation. This file is in the RPRC format with definitions for shared memory allocation.

STEP1: Create the config file with the required shared memory configurations.

$ create_ConfigRPRC.exe -s <Shared memory alloc>

Shared memory alloc: This is to indicate the share memory allocation to different cores. Each byte represents a core in the order. This value should match the value that will be provided in the next step to multicore generator.

BSS|MSS TCMB|MSS TCMA|DSS

Note: *1. This file will be required only for 14xx ES3.0 devices. For all other devices, this step should be skipped and the corresponding core Id should not be passed to the multicore generator in the next step.*

2. ar1xxx_conf.bin is the hardcoded file name generated by the utility which should be used as an input to multicore generator in the next step along with its core Id.

STEP2: Create the Multicore Image from all the RPRC images using the following command.

$ MulticoreImageGen.exe <ENDIAN> <Dev Id> <Shared memory alloc> <App out file> <Core Id 1> <RPRC inp file for Core Id 1> [<Core Id n> <RPRC inp file for Core Id n> ...]

ENDIAN (LE): specifies whether the out file is in Big Endian/Little Endian format. Set this to LE always.

Dev Id: This is the device identifier. This field is not interpreted by the bootloader currently.

Shared memory alloc: This is to indicate the share memory allocation to different cores. Each byte represents a core in the order

BSS|MSS TCMB|MSS TCMA|DSS

For example, if one bank has to be allocated to BSS and 5 banks to DSS/L3,example allocation would be 0x01000005. (BSS – 1 bank, DSS L3 – 5 banks)

App out file: This is the name of the multicore image file to be created.

Core Id: This is the Id of the core as given in the table below to be followed by the RPRC image file name to be loaded in that core.

Core Name	Core Id
MSS Image	0x35510000

BSS Image	0xB5510000
DSS Image	0xD5510000
Config file	0xCF910000

Table 1

RPRC inp file for Core Id: The path for the RPRC file for this core Id.

STEP3: The CRC generator (crc_multicore_image.exe) is used for updating the CRC of the images and the meta header. The crc can be updated in the generated multicore Image using the below command. metaImage.tmp is name of a temp file that the utility needs (doesn't need to exist before the utility is called and can be deleted after returning from this utility).

$ crc_multicore_image.exe metaImage.bin metaImage.tmp

STEP4: Also, the gen_bincrc32 is used to calculate and append the CRC of the entire multicore image at the end of the file. This step ensures an integrity check while the image is being downloaded to the SFLASH assisted by the bootloader.

$ gen_bincrc32.exe metaImage.bin