APP Data Tool Quick Start Guide

V1.0

2023/11/01

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Modify | Author | Reviewer |
| **2023/11/01** | V1.0 | First Release | Arthur |  |

Contents

[Revision History 2](#_Toc149741854)

[Contents 3](#_Toc149741855)

[1 Overview 4](#_Toc149741856)

[2 Operation steps 5](#_Toc149741857)

[2.1 APP Data1 5](#_Toc149741858)

[2.2 APP Data2 6](#_Toc149741859)

[2.3 APP Data3~6 6](#_Toc149741860)

[3 Version management of APP Data 7](#_Toc149741861)

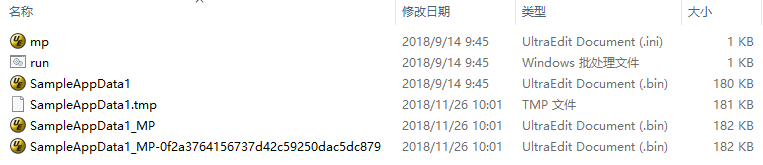
# Overview

APP Data Tool converts the original APP Data bin file into APP Data type image that can be burned into External Flash by RTL87x2G MP Tool. After APP Data Tool processes, it will add a 1280 bytes image header and 512 bytes MP header in front of the original APP data, which are used for Bootloader check and MP Tool burning respectively. Assuming that the size of the bin file of the original APP Data is 120KB, the final burn to the flash will be 1280 bytes image header plus 120KB APP data payload.

# Operation steps

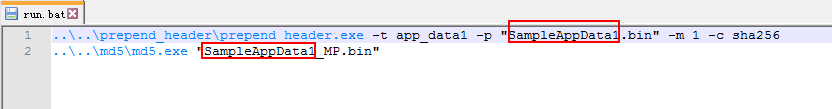
## APP Data1

1. The directory of sdk\tools\AppData\AppData1 is as follows:



SampleAppData1.bin is the bin file of the original APP Data.

SampleAppData1\_MP-0f2a3764156737d42c59250dac5dc879.bin is the APPData1 bin file generated by tool for burning.

1. Open “run.bat”, replace the following two parts with the bin file name of the APPdata to be processed. Note that the second line must be "xxx\_MP.bin", and the suffix "\_MP" cannot be changed. 
2. Run “run.bat”, three additional files will be generated:
3. SampleAppData1.tmp is a temporary file with 1280 bytes image header added to the bin file of the original APP Data;
4. SampleAppData1\_MP.bin is the bin file with 512 bytes of MP Header added to the temporary file;
5. SampleAppData1\_MP-0f2a3764156737d42c59250dac5dc879: The MD5 check value is calculated on the basis of "xxx\_MP.bin". Note that only bin files with both \_MP and MD5 check can be used for burning and OTA.

## APP Data2

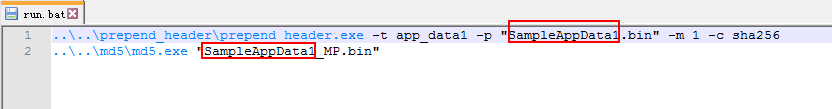
1. The directory of sdk\tools\AppData\AppData2 is as follows:



SampleAppData2.bin is the bin file of the original APP Data.

SampleAppData2\_MP-d873c53e6ba57dfc476defeba1794f97.bin is the APPData2 bin file generated by tool for burning.

1. Open “run.bat”, replace the following two parts with the bin file name of the APPdata to be processed. Note that the second line must be "xxx\_MP.bin", and the suffix "\_MP" cannot be changed.



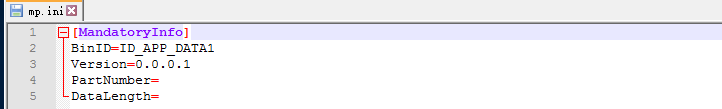
1. Run “run.bat”, three additional files will be generated:
2. SampleAppData2.tmp is a temporary file with 1280 bytes image header added to the bin file of the original APP Data;
3. SampleAppData2\_MP.bin is the bin file with 512 bytes of MP Header added to the temporary file;
4. SampleAppData2\_MP-d873c53e6ba57dfc476defeba1794f97: The MD5 check value is calculated on the basis of "xxx\_MP.bin". Note that only bin files with both \_MP and MD5 check can be used for burning and OTA.

## APP Data3~6

The operation is similar to the processing of APP data1 and APP data2 above, so it won't repeat them here.

# Version management of APP Data

There is a "mp.ini” configuration file in the sdk\tool\AppData\AppDataX directory as shown in the figure below. By modifying the configuration file, you can generate bin files of different versions of APP data.



Among them, "Version=0.0.0.1" means the version of the generated APP data. The four-segment version number at this point indicates the version information "Major. Minor. Revision. Reserve", which occupies a total of 32 bits, corresponding to the following structure.typedef struct

typedef struct

{

union

{

uint32\_t version;

struct

{

uint32\_t \_version\_major: 4; //!< major version

uint32\_t \_version\_minor: 8; //!< minor version

uint32\_t \_version\_revision: 15; //!< revision version

uint32\_t \_version\_reserve: 5; //!< reserved

} img\_sub\_version; //!< other image sub version including patch, app, app data1-6

} ver\_info;

} T\_IMAGE\_VERSION;

The version information will be stored in the git\_ver field in the image header of the first 1280 bytes, offset 0x1fe.

typedef union \_IMG\_HEADER\_FORMAT

{

uint8\_t bytes[DEFAULT\_HEADER\_SIZE];

struct

{

T\_AUTH\_HEADER\_FORMAT auth;

T\_IMG\_CTRL\_HEADER\_FORMAT ctrl\_header;

uint8\_t uuid[16];

uint32\_t exe\_base;

uint32\_t load\_src;

uint32\_t load\_len;

uint32\_t image\_base;

uint16\_t dev\_id; // for AMZN

uint16\_t flash\_layout\_size\_4k; // ex. 1MB flash used size => flash\_layout\_size\_4k = 1MB/4KB = 256

uint32\_t magic\_pattern;

uint8\_t dec\_key[16];

uint32\_t load\_dst;

T\_EXTRA\_INFO\_FORMAT ex\_info;

T\_VERSION\_FORMAT git\_ver;

PUBLIC\_KEY PubKey;

T\_FLASH\_SEC\_FORMAT flash\_sec\_cfg;

union

{

uint8\_t reserved\_data[344];

struct

{

uint32\_t ver\_val;

uint32\_t image\_info[(IMG\_MAX - IMG\_OTA - 1) \* 2];

uint32\_t reserved0[45];

};

struct \_\_attribute\_\_((packed))

{

uint32\_t tool\_version;

uint32\_t timestamp;

uint16\_t reserved1[168];

};

};

};

} T\_IMG\_HEADER\_FORMAT;