# Ameba-ZII Image PG Tool User Guide

This document introduces how to use Ameba-ZII Image PG Tool to download Images.

**DISCLAIMER**

Realtek provides this document "as is", without warranty of any kind. Realtek may make improvements and/or changes in this document or in the product described in this document at any time. This document could include technical inaccuracies or typographical errors.

**TRADEMARKS**

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

**USING THIS DOCUMENT**

This document is intended for the software engineer's reference and provides detailed programming information.

Though every effort has been made to ensure that this document is current and accurate, **more information may have become available subsequent to the production of this guide.**

**REVISION HISTORY**

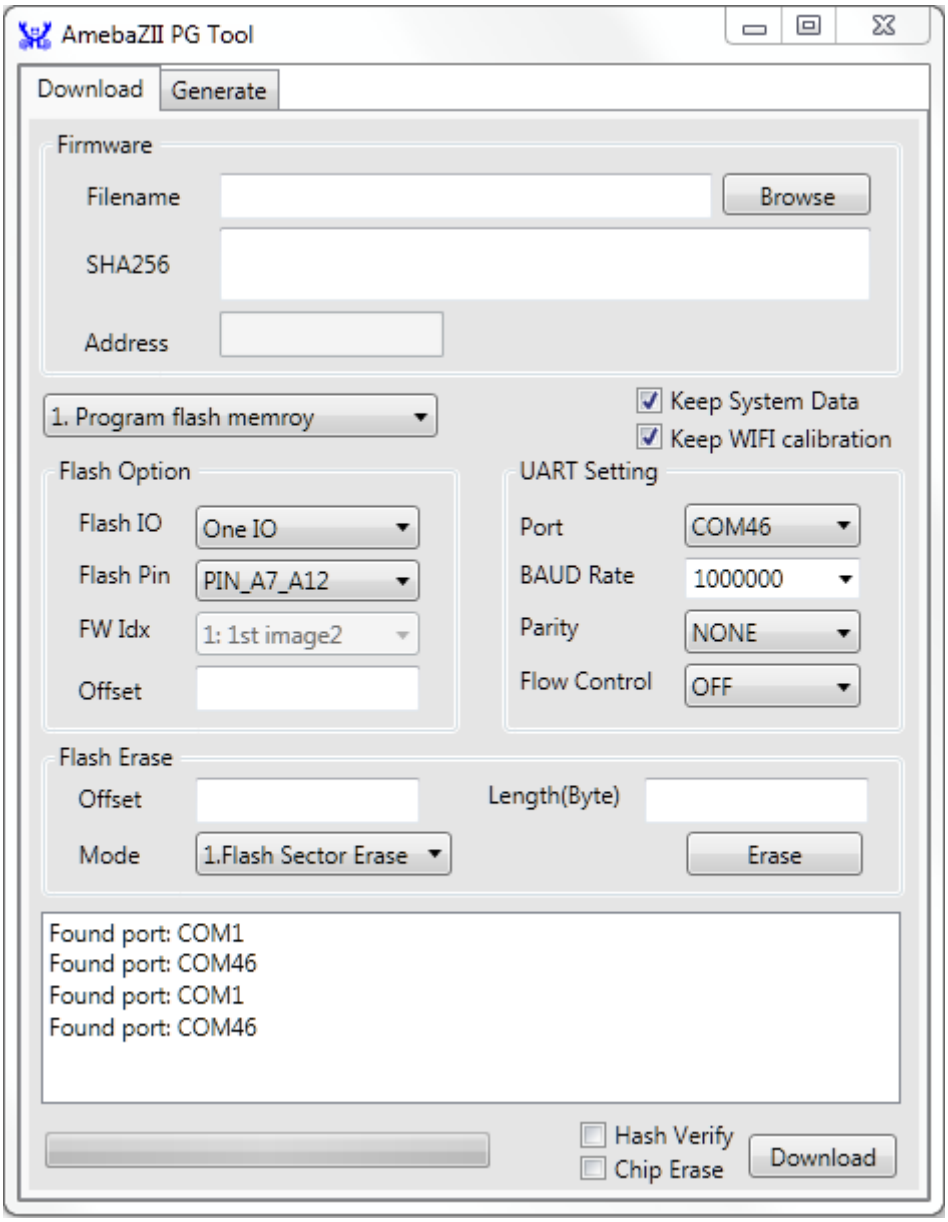
| Revision | Release Date | Summary |
|---|---|---|
| 0.1 | 2019/05/24 | Initial draft |
| 0.2 | 2019/06/13 | Update CMD mode functions, flash chip erase |
| 0.3 | 2019/06/28 | Update CMD mode for flash sector erase;<br>Update CMD mode for flash offset;<br>Add Auto Download Mode;<br>The Auto Download Mode enables download process at any states. AT command “ATXX” must be supported when running at normal state for activing the Auto Download Mode<br>The Auto Download Mode supports by both UI and CMD mode.<br>The update refer to AmebaZ2_PGTool version 1.0.7 |
| 0.4 | 2019/08/13 | Add Image generation, which is available after version PG Tool 1.2.0 |
| 0.5 | 2019/08/21 | Update CMD mode for baud rate |
| 0.6 | 2019/10/14 | Add Security Tool Implementation. |
| 0.7 | 2020/05/04 | Update CMD mode for multiple download, hash_verify, chip_erase and keep_sys_data. |
| 0.8 | 2020/06/15 | Update CMD mode for multiple download and normal download; Add flash status chapter |
| 0.9 | 2020/06/23 | Update description of “Flash Status” |
| 1.0 | 2020/07/28 | Add “boot option” feature in system data generation |
| 1.1 | 2020/09/17 | Add PGTool Linux version user guides |
| 1.2 | 2020/11/02 | Add encrypt trust zone project |
| 1.3 | 2021/06/04 | Add hash verification after erase |
| 1.4 | 2021/09/22 | Add download mode |
| 1.5 | 2021/11/09 | Add download with multiple setting options for multiple download |

# Table of Contents

# 1. Introduction

This document introduces how to use Image PG Tool to generate and download images. As show in following figure, Image PG Tool has two menu pages:

- Download: used as image download server to transmit images to Ameba through UART.
- Generate: contact individual images and generate a composite image.



# 2. Environment Setup

## 2.1. Hardware Setup

To download image, the following equipment is necessary:

- AmebaZII DEV
- micro USB cable
- USB FTDI cable (if need to download code via external UART)

  Please note that the USB FTDI cable **must use FT232 USB To UART dongle**.

The tool on PC sends images to AmebaZII through Log UART. The micro USB cable will both power the DEV and transmit data. If need to download code via external uart, micro USB cable only supply the power and USB FTDI cable transmits data. USB FTDI cable should connect to PA_15, PA_16 and GND.

## 2.2. Software Setup

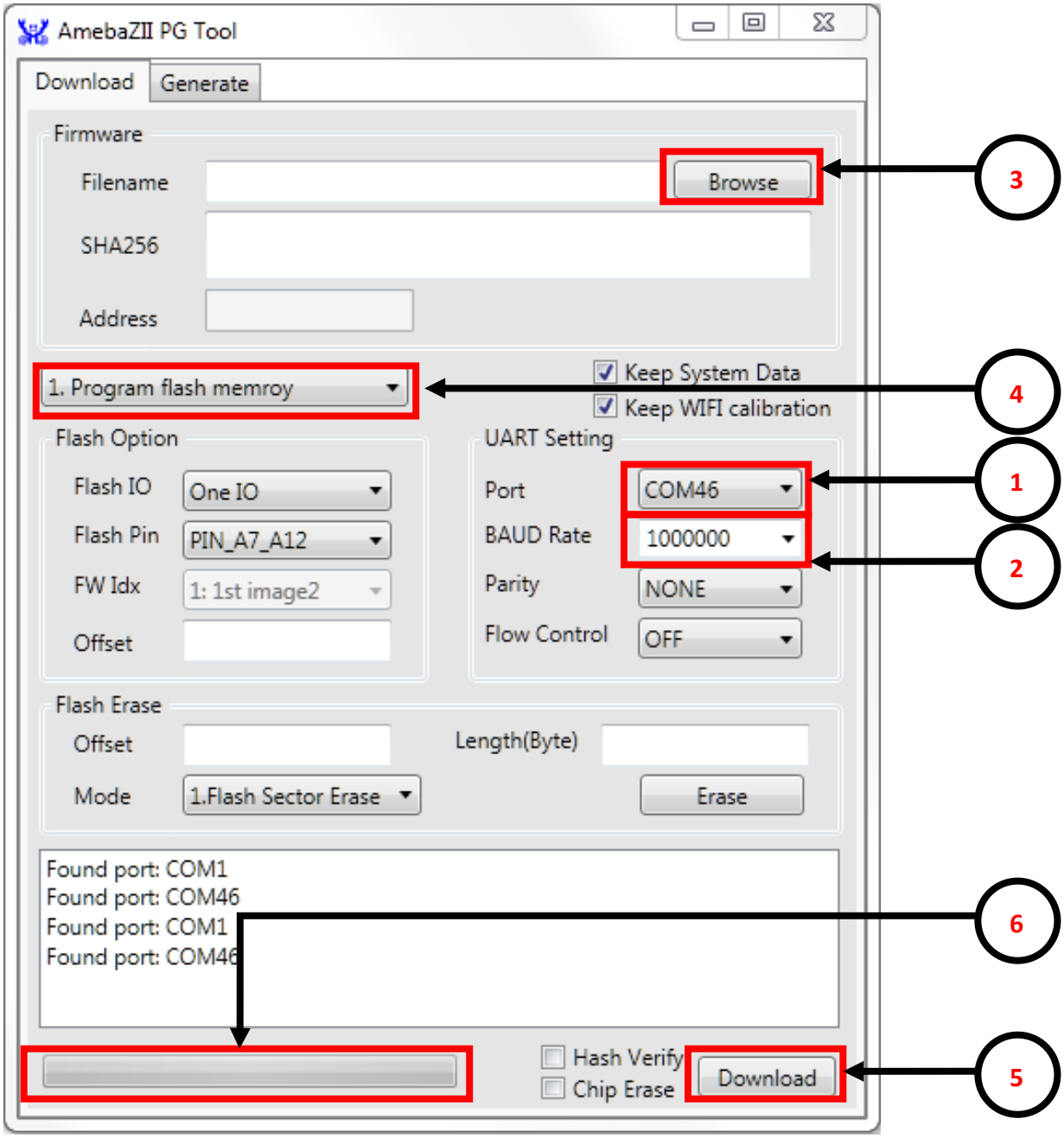
Environment Requirements:

- EX. WinXP, Win 7 or above
- Microsoft .NET Framework 3.5 or above

# 3. Image Download

## 3.1. Windows Form Mode

### 3.1.1. Quick start for full flash image burning




1) Application will scan available UART ports. Please choose correct **UART port**.

2) Choose desired **baud rate** between computer and AmebaZ2.

a) Use default value for parity and flow control.

3) Choose target flash binary image file "**flash_xx.bin**"

4) Check Mode is **1. Program flash memory**

5) Click **Download**, the tool will activated the Auto Download Mode for downloading process.

6) Progress will be shown on progress bar and result **OK/FAIL** will be shown after download finish.

NOTE:

- Other settings using default values,
- Flash IO : One IO / Flash Pin : PIN_A7_A12 / Parity : NONE / Flow Control : OFF

## 3.1.2. Function description

### 3.1.2.1 File selection, Hash verify, Chip erase

Click **Browse** button to select target flash image file, and it will calculate SHA256 hash verification word automatically.

To enable hash verification after downloading or after erasing, Check **hash verify** at the bottom of application.

To enable chip erase before downloading, Check **chip erase** at the bottom of application.

### 3.1.2.2 Mode

There are 6 modes for Image PG Tool.

1) Program flash memory

   Programing full flash image, image file is selected by user from previous step.

   Target file name is **flash_xx.bin** that will be created after SDK project built.

2) Program flash image 1

   Programing bootloader image, target file name is **boot.bin**.

3) Update image 2

   Programing firmware image, target file name is **firmware_xx.bin**.

User must select firmware target index by **FW Idx.**

4) Update image 2 auto

This is similar with option 3, but user does not need to select index. AmebaZ2 will choose partition that content older version to update firmware automatically.

5) Program RAM

This is a RESERVED mode for internal use.

6) UART boot with a FW image

This is a RESERVED mode for internal use.

### 3.1.2.3 Flash Option

1) Flash IO

Please use **One IO** if not sure.

2) Flash Pin

Choose correct "Flash Pin" according to the IC part number.

| Flash Pin | IC part number |
|---|---|
| PIN_A7_A12 | RTL8710CX/RTL8720CM |
| PIN_B6_B12 | RTL8720CF |

3) FW Idx

Select firmware partition when using **Mode 3.**

4) Offset

Offset address on the Flash.

### 3.1.2.4 UART setting

1) Port

Application will scan all available port on computer. Please check port number is match

corresponding hardware.

2) BAUD Rate/Parity/Flow control

Normal UART configuration, please check UART spec.

Please use **Parity: NONE** and **Flow Control: OFF** if not sure

Initial UART property is **115200 8-N-1** for setup stage**,** after download start UART will be reconfigured to user defined baud rate.

#### 3.1.2.5 Setting file

When logging out PG Tool, current settings will be save in setting.xml file under the same path.

When logging in PG Tool, it will load the setting.xml file first to recover the settings of last time.

The default value of UART timeout is 2 seconds, this value can be customize in setting.xml file.

The download speed can be increased by changing the “download_mode” to 4(PG test mode with RAM flash loader) in setting file. If the value of “download_mode” is not equal to 4, the download mode will be PG mode (default).

Please change “download_mode” when application is closed, and saved before open the application. The changes will take effective once reopen the application.

After change “download_mode” to 4, make sure the “Hash Verify” is ticked when you try to download the image.

## 3.2. Command Line Mode

The tool can also work in command line mode. Now the command line mode support normal download and multiple download. Start cmd.exe in windows and execute Ameba-ZII PG Tool AmebaZ2_PGTool.exe with defined parameters.

The supported parameters can be achieved by type “**-help**” as follows,

```
$ AmebaZ2_PGTool.exe -help
```

```
usage:

For normal download:
    -show [device|setting]
        E.g. -show device
    -set image <path>
        E.g. -set image D:\test\flash_is.bin
    -set mode <mode number>
            <mode number> 0 : Program flash memory
        E.g. -set mode 0
    -set flash_io <flash_io options>
            <flash_io options> 0 : One IO
        E.g. -set flash_io 0
    -set flash_offset <offset address>
        E.g. -set flash_offset 0x00000000
    -set baudrate <baudrate>
        E.g. -set baudrate 1000000
    -set hash_verify <hash_verify options>
        E.g. -set hash_verify 1
    -set chip_erase <chip_erase options>
        E.g. -set chip_erase 1
    -set keep_sys_data <keep_sys_data options>
        E.g. -set keep_sys_data 1
    -scan device
    -add device <device>
        E.g. -add device COM1
    -erase [sector <offect> <length(byte)> | chip]
        E.g. -erase sector 0x00000000 0x00000000
        E.g. -erase chip
    -download
```

```
For multiple download:
    -show COM1 <setting>
        E.g. -show COM1 setting
    -set COM1 image <path>
        E.g. -set COM1 image D:\test\flash_is.bin
    -set COM1 mode <mode number>
            <mode number> 0 : Program flash memory
        E.g. -set COM1 mode 0
    -set COM1 flash_io <flash_io options>
            <flash_io options> 0 : One IO
        E.g. -set COM1 flash_io 0
    -set COM1 flash_offset <offset address>
        E.g. -set COM1 flash_offset 0x00000000
    -set COM1 baudrate <baudrate>
        E.g. -set COM1 baudrate 1000000
    -set COM1 hash_verify <hash_verify options>
        E.g. -set COM1 hash_verify 1
    -set COM1 chip_erase <chip_erase options>
        E.g. -set COM1 chip_erase 1
    -set COM1 keep_sys_data <keep_sys_data options>
        E.g. -set COM1 keep_sys_data 1
    -scan device
    -add device <device>
        E.g. -add device COM1
    -erase COM1 [sector <offect> <length(byte)> | chip]
        E.g. -erase COM1 sector 0x00000000 0x00000000
        E.g. -erase COM1 chip
    -download COM1
```

### 3.2.1. Devices Connections

You can use "**-scan device"** parameter to check serial ports connected to PC as follows,

$ AmebaZ2_PGTool.exe -scan device

```
Found port: RFCOMM Protocol TDI
Found port: COM39
Found port: COM38
Device scan done
Available device port:    RFCOMM Protocol TDI
Available device port:    COM39
Available device port:    COM38
```

You can use "**-add device**" to connect device port for download, after adding device, there will be two setting files created, the one is named CMD_download_setting.txt which is for normal download, the other on is named COM38_CMD_download_setting.txt which is for multiple download.

$ AmebaZ2_PGTool.exe -add device COM38

```
Device added: COM38
```

## 3.2.2. Configuration

### 3.2.2.1 Configuration of Normal Download

- To check the tool configuration, you can use "**-show setting**" as parameters.

$ AmebaZ2_PGTool.exe -show setting

```
CMD_download_setting
 Image file path:  None
 Mode:             Program flash memory
 Flash IO:         One IO
 Flash offset:     0x00000000
 Baudrate:         1000000
 Hash Verify:      0
 Chip Erase:       0
 Keep System Data: 0
```

- To choose file to download, you can use "**-set image <path>**" as parameters.

$ AmebaZ2_PGTool.exe -set image D:\test\flash_is.bin

```
File path: d:\test\flash_is.bin
File length: 458816
```

***Currently, the tool only supports Program flash memory mode and One IO, and can select flash_pin automatically, so there is no need to set them again. The default value has been set in COMxx_CMD_download_setting.txt already, so setting mode, flash_io and flash_pin <u>can be skipped</u>.***

- To choose mode to download, you can use "**-set mode <mode number>**" as parameters.

$ AmebaZ2_PGTool.exe -set mode 0

```
Mode selection: Program flash memory
```

- To choose flash_io to download, you can use "**-set flash_io <flash_io options>**" as parameters.

$ AmebaZ2_PGTool.exe -set flash_io 0

```
Flash IO selection: One IO
```

- To set the downloading start address of the flash, you can use "**-set flash_offset <offset address>**".

$ AmebaZ2_PGTool.exe -set flash_offset 0x00000000

```
Mode: Program flash memory
Flash offset: 0x00000000
```

- To set the downloading baudrate, you can use "**-set baudrate <baudrate>**".

$ AmebaZ2_PGTool.exe -set baudrate 1000000

```
Baudrate: 1000000
```

- To enable/disable hash verify function, you can use **"-set hash_verify <hash_verify options>"**.

$ AmebaZ2_PGTool.exe -set hash_verify 1

```
Hash Verify: 1
```

- To enable/disable chip erase function, you can use **"-set chip_erase <chip_erase options>"**.

$ AmebaZ2_PGTool.exe -set chip_erase 1

```
Chip Erase: 1
```

- To enable/disable keep system data function, you can use **"-set keep_sys_data <keep_sys_data options>"**.

$ AmebaZ2_PGTool.exe -set keep_sys_data 0

```
Keep System Data: 0
```

- To increase download speed, you can use **"-set download_mode <download_mode options>"**. If set to 4, must make sure hash_verify is set to 1.

$ AmebaZ2_PGTool.exe -set download_mode 1

```
Download Mode: 1
```

- After adding device, there is a setting file name CMD_download_setting.txt. You can manually configure "**CMD_download_device.txt**" for download configuration.

CMD_download_setting - Notepad

File Edit Format View Help

```
#Image File Path#
d:\test\flash_is.bin
#Mode selection#
0
#Flash IO selection#
0
#Flash offset#
0x00000000
#Image File Length#
1746368
#Flash Erase Mode#
None
#Flash sector erase offset#
None
#Flash sector erase offset Length#
None
#Flash Pin#
0
#Baudrate#
1000000
#Hash Verify#
0
#Chip Erase#
0
#Keep System Data#
0
#Flash Status#
0x00000000
#Connect Timeout#
2
#Interval#
2
#Download Mode#
1
#Port Name#
COM38
```

Ln 28, Col 11 | 100% | Windows (CRLF) | UTF-8

#### 3.2.2.2 Configuration of Multiple Download

- To check the tool configuration, you can use "**-show COMxx setting**" as parameters.

$ AmebaZ2_PGTool.exe -show COM38 setting

```
CMD_download_setting
 Image file path:  None
 Mode:             Program flash memory
 Flash IO:         One IO
 Flash offset:     0x00000000
 Baudrate:         1000000
 Hash Verify:      0
 Chip Erase:       0
 Keep System Data: 0
```

- To choose file to download, you can use "**-set COMxx image <path>**" as parameters.

$ AmebaZ2_PGTool.exe -set COM38 image D:\test\flash_is.bin

```
File path: d:\test\flash_is.bin
File length: 458816
```

***Currently, the tool only supports Program flash memory mode and One IO, and can select flash_pin automatically, so there is no need to set them again. The default value has been set in COMxx_CMD_download_setting.txt already, so setting mode, flash_io and flash_pin <u>can be skipped</u>.***

- To choose mode to download, you can use "**-set COMxx mode <mode number>**" as parameters.

$ AmebaZ2_PGTool.exe -set COM38 mode 0

```
Mode selection: Program flash memory
```

- To choose flash_io to download, you can use "**-set COMxx flash_io <flash_io options>**" as parameters.

$ AmebaZ2_PGTool.exe -set COM38 flash_io 0

```
Flash IO selection: One IO
```

- To set the downloading start address of the flash, you can use "**-set COMxx flash_offset <offset address>**".

$ AmebaZ2_PGTool.exe -set COM38 flash_offset 0x00000000

```
Mode: Program flash memory
Flash offset: 0x00000000
```

- To set the downloading baudrate, you can use "**-set COMxx baudrate <baudrate>**".

$ AmebaZ2_PGTool.exe -set COM38 baudrate 1000000

```
Baudrate: 1000000
```

- To enable/disable hash verify function, you can use "-set COMxx hash_verify <hash_verify options>".

$ AmebaZ2_PGTool.exe -set COM38 hash_verify 1

```
Hash Verify: 1
```

- To enable/disable chip erase function, you can use "-set COMxx chip_erase <chip_erase options>".

$ AmebaZ2_PGTool.exe -set COM38 chip_erase 1

```
Chip Erase: 1
```

- To enable/disable keep system data function, you can use "-set COMxx keep_sys_data <keep_sys_data options>".

$ AmebaZ2_PGTool.exe -set COM38 keep_sys_data 0

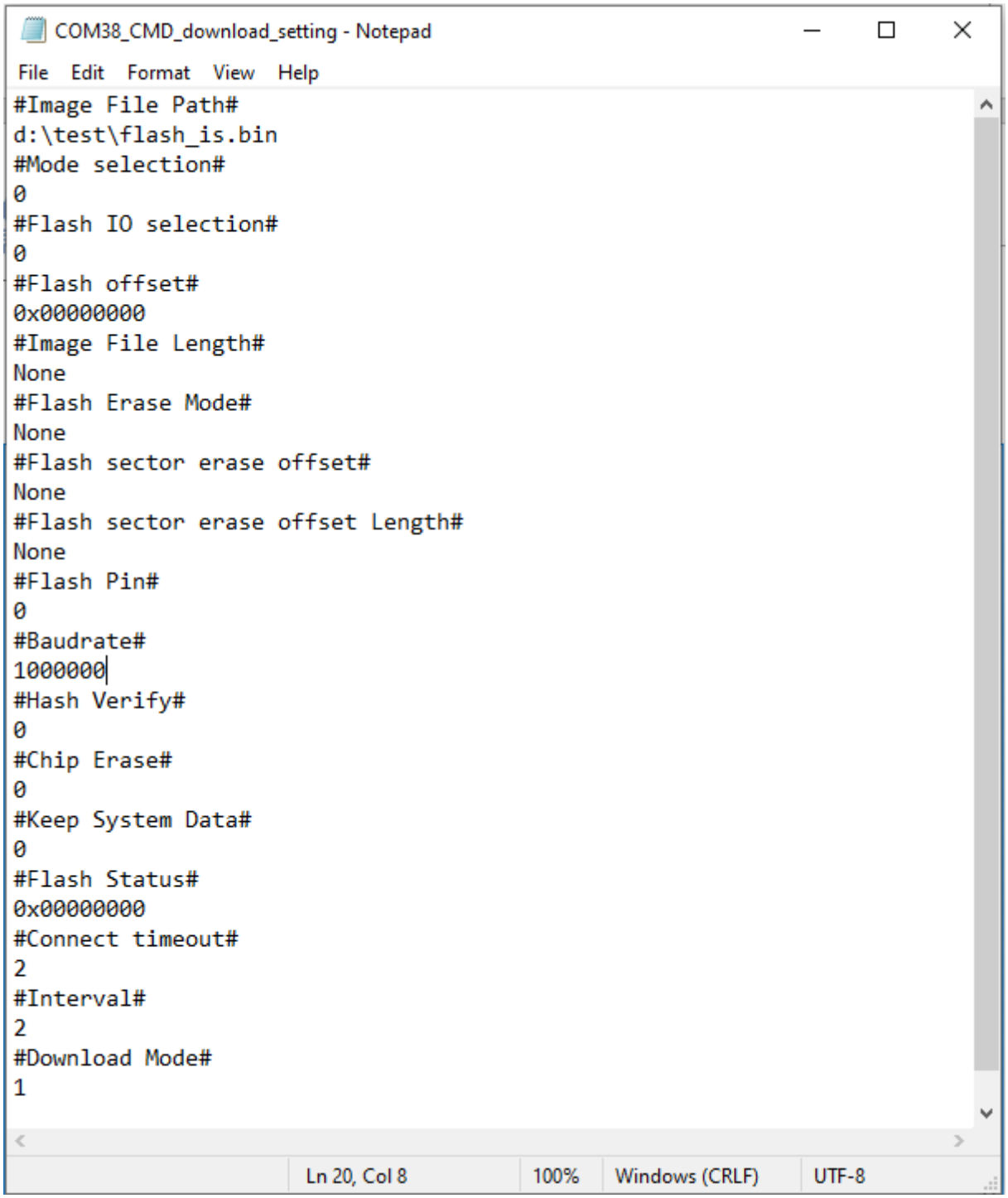
```
Keep System Data: 0
```

- To increase download speed, you can use "-set COMxx download_mode <download_mode options>". If set to 4, must make sure hash_verify is set to 1.

$ AmebaZ2_PGTool.exe -set COM38 download_mode 1

```
Download Mode: 1
```

- After adding device, there will be COMxx_download_setting.txt created. You can manually configure "**COMxx_CMD_download_setting.txt**" for download configuration.



### 3.2.3. Flash Erase

In order to successful apply the flash erase function, please enter the PG mode before apply any command.

- Please hold the UART download button and presses reset button to let the board enter the PG mode. (Please refer to ***AN0500 Realtek Ameba-ZII application note.en***)

#### 3.2.3.1 Flash Sector Erase

You can use "**-erase sector <offset> <length (byte)>**" to erase sector of the device that you added in **CMD_download_setting.txt**, also you can use "**-erase COMxx sector <offset> <length (byte)>**" to erase sectors of flash for target device COMxx. **<offset>** is to set the start address of the flash. **<length (byte)>** is to set the length of the erase area. Note that a sector is 4K bytes, the length should be Integer multiple of the 4K bytes.

$ AmebaZ2_PGTool.exe -erase sector 0x00001000 0x1000 ***or***

$ AmebaZ2_PGTool.exe -erase COM38 sector 0x00001000 0x1000

```
Flash Sector Erase offset: 0x00001000
Flash Sector Erase length: 0x1000
Start   Flash Sector Erase
Found port: RFCOMM Protocol TDI
Found port: COM39
Found port: COM38
Download Mode
Disconnect
ERASE: Success
End   Flash Sector Erase
```

#### 3.2.3.2 Flash Chip Erase

You can use "**-erase chip**" to erase chip of the device added in **CMD_download_setting.txt.**

You can also use "**-erase COMxx chip**" to erase flash chip for target device COMxx.

$ AmebaZ2_PGTool.exe -erase chip ***or***

$ AmebaZ2_PGTool.exe -erase COM38 chip

```
Start   Flash Chip Erase
Found port: RFCOMM Protocol TDI
Found port: COM39
Found port: COM38
Download Mode
Disconnect
ERASE: Success
End   Flash Chip Erase
```

#### 3.2.3.3 Hash verify after erase

If you want to make sure the erase has been taken place. You can enable the hash verification. The hash verification will be taken place after erase process is done. You will be able to see the Hash verify result to check whether the target place is erased.

```
Start   Flash Chip Erase
Found port: COM5
Found port: RFCOMM Protocol TDI
Download Mode
Hash checking
Hash verification: Pass
Disconnect
ERASE: Success
End   Flash Chip Erase
```

### 3.2.4. Download image

After all configuration is finished, you can use "**-download**" to download image to device added in **CMD_download_setting.txt**, or you can use "**-download COMxx**" to download image to target device COMxx. Or you can use one-line download command with multiple setting options "**-download COMxx -set [option1] -set[option2] ……**". The options will be overwritten the previous setting/default setting in **COMxx_CMD_download_setting.txt.** "-download", "-download COMxx", and "-download COMxx -set [option1] -set[option2] ……" enables the Auto Download Mode for downloading process.

- Please hold the UART download button and presses reset button to let the board enter the PG mode. (Please refer to ***AN0500 Realtek Ameba-ZII application note.en***)
- Auto Download Mode is supported, user may not necessary to enter the PG mode.

$ AmebaZ2_PGTool.exe -download ***or***

$ AmebaZ2_PGTool.exe -download COM38 ***or***

$ AmebaZ2_PGTool.exe -download COM38 -set image test.bin -set hash_verify 1-set chip_erase 1

```
Start   Download
Found port: RFCOMM Protocol TDI
Found port: COM39
Found port: COM38
Download Mode
Hash Verify enabled
Flash Erase enabled
PG: start
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %1
Downloading --- %5
Downloading --- %15
Downloading --- %21
Downloading --- %29
Downloading --- %40
Downloading --- %48
Downloading --- %54
Downloading --- %65
Downloading --- %73
Downloading --- %79
Downloading --- %87
Downloading --- %98
Downloading --- %100
Hash checking
Hash verification: Pass
WORKER: complete
End   Download
```

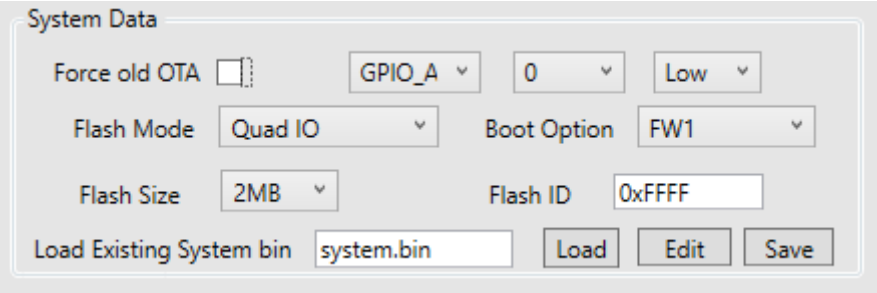
# 4. Image Generation

The Generate tab page has two functions:

- Generate binary for system data section
- Concat separate images and generate a final image named flash.bin

## 4.1. System data generation

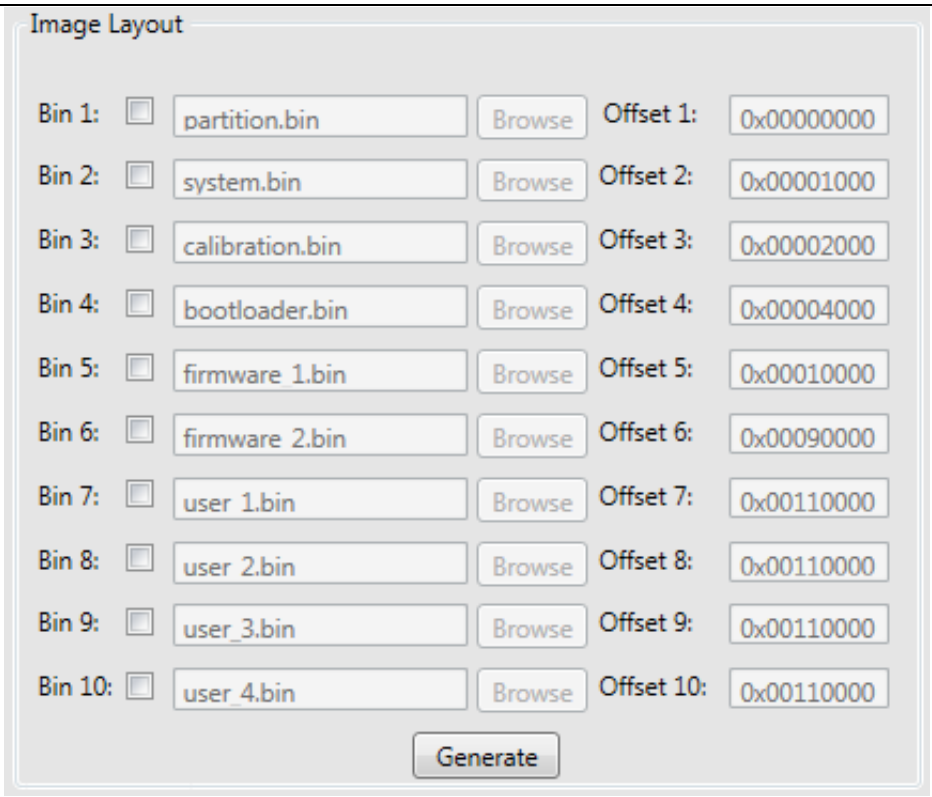
User can “Load” an existing binary file and “Edit” accordingly. After edit, system data can be saved to system.bin in tool directory after click “Save” button. You will see the introduction of each configuration in system data in AN0500 Realtek Ameba-ZII application note.en.




For PGTool which has “Boot Option” feature, please make sure the image supports SWAP boot first. Then you can decide the firmware you want to boot from by selecting FW1 or FW2.
Note: Don’t modify the system data unless it’s necessary.

## 4.2. Flash image generation

User can select images to be concated and input corresponding address. The memory layout can refer to AN0500 Realtek Ameba-ZII application note.en and the address can be found in partition.json which locates in project\realtek_amebaz2_v0_example\EWARM-RELEASE and project\realtek_amebaz2_v0_example\GCC-RELEASE.




### 4.2.1. How to concat a normal image and a MP image

Below are the simple steps of how to use PG Tool to concat a normal image and a MP image and switch between them during MP process.

- Check “Bin 5”, “Browse” for the normal image “flash_is.bin” generated by IAR/GCC project. (Note that flash_is.bin includes already partition.bin, bootloader.bin & firmware_1.bin)
- Set “Offset 5” to “0x0”
- Check “Bin 6”, “Browse” for the MP image “firmware_is.bin” generated by IAR/GCC project. (Note that firmware_is.bin includes only firmware_1.bin or firmware_2.bin)
- Set “Offset 6” to “0x90000” which is the default address for “firmware_2” in SDK.
- “Generate” to save the output image “flash.bin”
- Go back to the “Download” sub tab to download flash.bin
- Reboot the chip and use ATSR/ATSC to switch between normal image and mp image

# 5. Security

This feature is only available in 1.3.x or newer version of PG Tool. And it's only released on demand. Please approach our FAE if needed.

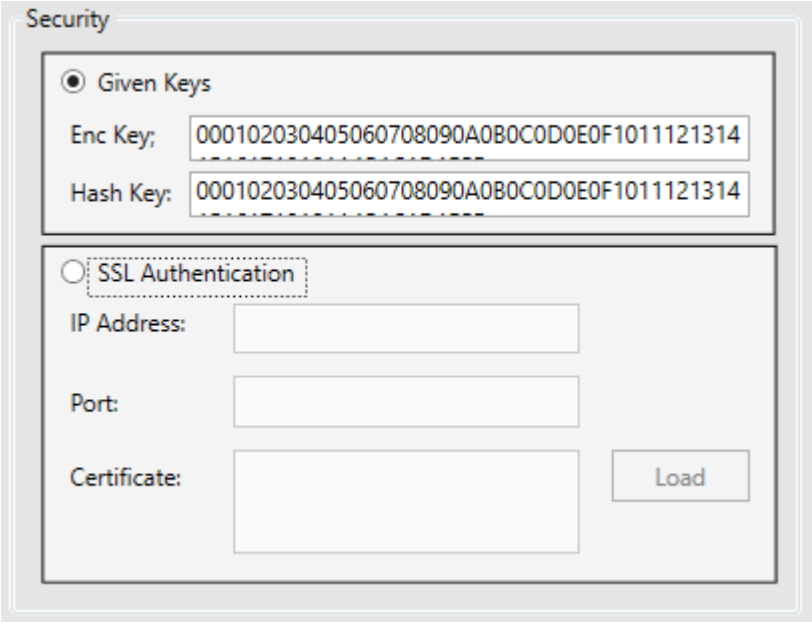
The security tab page mainly has two functions:

- Encrypt Image
- Program efuse and enable secure boot

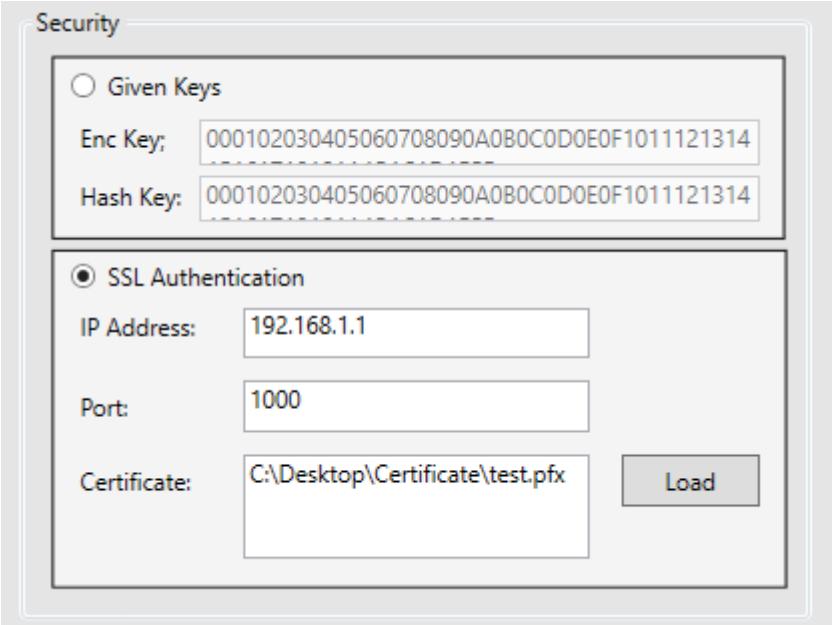
## 5.1. Encrypt Image

### 5.1.1. Get Keys

Encrypt Image need two keys, the one is encrypted key named Enc Key, the other is hash key named Hash Key. There are two ways to get the pair of keys. The keys can be given directly, also can be gotten from server by SSL authentication.



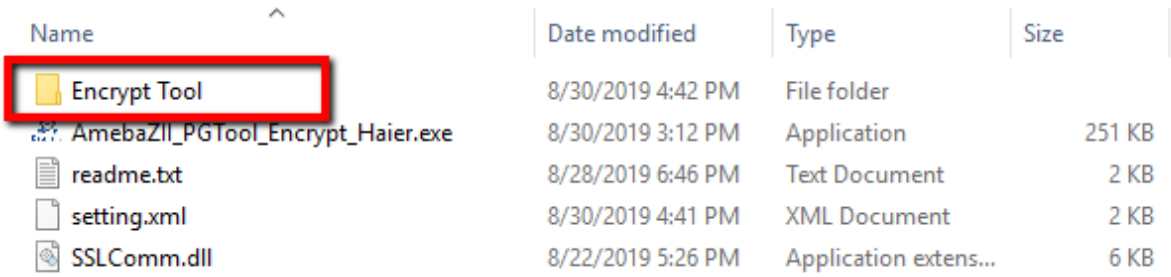
When keys are gotten by SSL authentication, the IP address and the port of the server need to be given in SSL Authentication box. The certificate also needs to be loaded to achieve SSL two-way authentication. The certificate should be in ".pfx" format.




## 5.1.2. Configuration

### 5.1.2.1 "is" project

What accompanies with the ".exe" is a folder named Encrypt Tool.



| Name | Date modified | Type | Size |
|---|---|---|---|
| Encrypt Tool | 8/30/2019 4:42 PM | File folder | |
| AmebaZII_PGTool_Encrypt_Haier.exe | 8/30/2019 3:12 PM | Application | 251 KB |
| readme.txt | 8/28/2019 6:46 PM | Text Document | 2 KB |
| setting.xml | 8/30/2019 4:41 PM | XML Document | 2 KB |
| SSLComm.dll | 8/22/2019 5:26 PM | Application extens... | 6 KB |

Open the folder, there are some files located here. The Encrypt Tool will use *amebaz2_bootloader.json*, *amebaz2_firmware_is.json* and *partition.json* to encrypt the image.

| Name | Date modified | Type | Size |
|---|---|---|---|
| Debug | 8/30/2019 4:42 PM | File folder | |
| amebaz2_bootloader.json | 7/24/2019 7:16 PM | JSON File | 3 KB |
| amebaz2_firmware_is.json | 7/24/2019 7:16 PM | JSON File | 6 KB |
| elf2bin.exe | 8/22/2019 11:12 AM | Application | 743 KB |
| key.json | 8/30/2019 4:33 PM | JSON File | 1 KB |
| partition.json | 8/30/2019 3:27 PM | JSON File | 2 KB |

The *partition.json* should keep same with the *partition.json* that building the plain image. The *amebaz2_bootloader.json* and *amebaz2_firmware_is.json* also need to keep the same with what used to build the plain image except the *"enc"* flag. These two files stored in Encrypt Tool folder have been configured as encrypt mode based on default files in Realtek standard SDK. If you need to change some configurations (e.g. serial number), please make sure these changes also are applied to these two files under Encrypt Tool folder.

In *amebaz2_bootloader.json,*

```
"PARTAB": {
        "source":null,
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP,M0,CPFW",
            "type":"PARTAB",
            "enc":true,
            "serial": 0
        },
        "list" : ["partab"],
        "partab": {
            "__comment_ptable":"move to partition.json",

            "__comment_file":"TODO: use binary file directly",
            "file": null
        }
    },
"BOOT": {
        "source":"Debug/Exe/bootloader.out",
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP,M0,CPFW",
```

```
                "type":"BOOT",
                "enc":true,

"user_key1":"AA0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1
D1E1F",
                "serial": 0
            },
            "list" : ["sram"],
            "sram": {
                "__comment_option":"TODO: not ready",
                "option": null,

                "__comment_entry":"startup function table symbol",
                "entry":"gRamStartFun",

                "start": "RAM_FUNTAB$$Base",
                "end": "RAM_RODATA$$Limit",

                "__comment_file":"TODO: use binary file directly",
                "file": null
            }
        }
```

In *amebaz2_firmware_is.json*, *"enc"* also is set to *true* to encrypt the image. But what needs to be noted is, the *"enc"* in "XIP_FLASH_P" could not be set to true because this section is reserved for plain data.

```
"FWHS": {
            "source":"Debug/Exe/application_is.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"FWHS_S",
                "enc":true,

        "user_key2":"BB0102030405060708090A0B0C0D0E0F101112131415161718191A1B
1C1D1E1F",
                "__comment_pkey_idx":"assign by program, no need to configurate",
```

```
                "serial": 107
            },
             "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",

                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":true,
                "cipherkey":null,
                "cipheriv":null
             },
           }

"XIP_FLASH_C": {
            "source":"Debug/Exe/application_is.dbg.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"XIP",
                "enc":true,
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 0
            },
             "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",
```

```
                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":true,
                "cipherkey":null,
                "cipheriv":null
            },
}

"XIP_FLASH_P": {
            "source":"Debug/Exe/application_is.dbg.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"XIP",
                "enc":false,
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 0
            },
             "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",

                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":false,
                "cipherkey":null,
                "cipheriv":null
            },
}
```

There is no need to do any change of elf2bin.exe. The ".exe" file is just called to build the

encrypted image.
*key.json* file is generated by the encrypted image building process. Every time the image is encrypted, the *key.json* file will be updated.

In Debug folder, there is only a folder named *Exe*. In *Exe* folder, there are two files, the one is *application_is.dbg.out*, the other is *bootloader.out*. The *application_is.dbg.out* is found under \project\realtek_amebaz2_v0_example\EWARM-RELEASE\Debug\Exe where you build the plain image. Copy the file from \project\realtek_amebaz2_v0_example\EWARM-RELEASE\Debug\Exe to Exe folder, the tool will use this file to generate encrypted image. The *bootloader.out* should also be replaced by the version you build the plain image under \component\soc\realtek\8710c\misc\bsp\image.

| Name | Date modified | Type | Size |
|---|---|---|---|
| application_is.dbg.out | 8/29/2019 10:22 AM | OUT File | 17,145 KB |
| bootloader.bin | 8/30/2019 3:27 PM | BIN File | 17 KB |
| bootloader.out | 7/30/2019 5:32 PM | OUT File | 464 KB |
| firmware_is.bin | 8/30/2019 3:27 PM | BIN File | 678 KB |
| flash_is.bin | 8/30/2019 3:27 PM | BIN File | 742 KB |
| partition.bin | 8/30/2019 3:27 PM | BIN File | 1 KB |

### 5.1.2.2 "tz" project

What accompanies with the ".exe" is a folder named Encrypt Tool.

| Name | Date modified | Type | Size |
|---|---|---|---|
| Encrypt Tool | 8/30/2019 4:42 PM | File folder | |
| AmebaZII_PGTool_Encrypt_Haier.exe | 8/30/2019 3:12 PM | Application | 251 KB |
| readme.txt | 8/28/2019 6:46 PM | Text Document | 2 KB |
| setting.xml | 8/30/2019 4:41 PM | XML Document | 2 KB |
| SSLComm.dll | 8/22/2019 5:26 PM | Application extens... | 6 KB |

Open the folder, there are some files located here. The Encrypt Tool will use *amebaz2_bootloader.json*, *amebaz2_firmware_tz.json* and *partition.json* to encrypt the image.

| | | | |
|---|---|---|---|
| Debug | 7/3/2020 3:14 PM | File folder | |
| amebaz2_bootloader.json | 7/24/2019 7:16 PM | JSON File | 3 KB |
| amebaz2_firmware_is.json | 9/25/2019 10:32 AM | JSON File | 6 KB |
| amebaz2_firmware_tz.json | 6/22/2020 9:29 PM | JSON File | 11 KB |
| elf2bin.exe | 6/22/2020 9:05 PM | Application | 753 KB |
| partition.json | 10/8/2019 3:03 PM | JSON File | 2 KB |

The *partition.json* should keep same with the *partition.json* that building the plain image.

The *amebaz2_bootloader.json* and *amebaz2_firmware_tz.json* also need to keep the same with what used to build the plain image except the *"enc"* flag. These two files stored in Encrypt Tool folder have been configured as encrypt mode based on default files in Realtek standard SDK. If you need to change some configurations (e.g. serial number), please make sure these changes also are applied to these two files under Encrypt Tool folder.

In *amebaz2_bootloader.json,*

```
"PARTAB": {
        "source":null,
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP,M0,CPFW",
            "type":"PARTAB",
            "enc":true,
            "serial": 0
        },
        "list" : ["partab"],
        "partab": {
            "__comment_ptable":"move to partition.json",

            "__comment_file":"TODO: use binary file directly",
            "file": null
        }
    },
"BOOT": {
        "source":"Debug/Exe/bootloader.out",
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP,M0,CPFW",
            "type":"BOOT",
            "enc":true,

"user_key1":"AA0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1
D1E1F",
            "serial": 0
        },
        "list" : ["sram"],
        "sram": {
```

```
            "__comment_option":"TODO: not ready",
            "option": null,

            "__comment_entry":"startup function table symbol",
            "entry":"gRamStartFun",

            "start": "RAM_FUNTAB$$Base",
            "end": "RAM_RODATA$$Limit",

            "__comment_file":"TODO: use binary file directly",
            "file": null
        }
    }
```

In *amebaz2_firmware_tz.json*, *"enc"* also is set to *true* to encrypt the image. But what needs to be noted is, the *"enc"* in "XIP_S_P" and ""XIP_NS_P" could not be set to true because this section is reserved for plain data.

```
{
     "msg_level": 3,

    "__comment":"example key",
    "000priv":"A0D6DAE7E062CA94CBB294BF896B9F68CF8438774256AC7403CA4FD9A
1C9564F",
    "000pub":"68513EF83E396B12BA059A900F36B6D31D11FE1C5D25EB8AA7C550307F
9C2405",
    "001priv":"882AA16C8C44A7760AA8C9AB22E3568C6FA16C2AFA4F0CEA29A10ABCD
F60E44F",
    "001pub":"48AD23DDBDAC9E65719DB7D394D44D62820D19E50D68376774237E98
D2305E6A",
    "002priv":"58A3D915706835212260C22D628B336D13190B539714E3DB249D823CA
5774453",
    "002pub":"FD8D3F3E516D96186E10F07A64B24C7DE736826A24FAFE367E79F1FBB2
F1C832",
    "003priv":"000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C
1D1E5F",
    "003pub":"8F40C5ADB68F25624AE5B214EA767A6EC94D829D3D7B5E1AD1BA6F3E2
138285F",

    "PROFILE":["FIRMWARE"],
```

```
    "FIRMWARE":{
        "rand_pad": false,
        "__comment_xip_pg_size":"XIP remapping page size/alignment setting: 0/1/2:
16K/32K/64K",
        "xip_pg_size": 0,

        "__comment_mode":"mode 0: bootloader and partition table, mode 1:
firmware",
        "mode": 1,
        "file":"Debug/Exe/firmware_tz.bin",
        "__comment_too_privkey":"if user want to fix key, can set priviate key here, if
not, will use random key",

    "privkey_enc":"A0D6DAE7E062CA94CBB294BF896B9F68CF8438774256AC7403CA4F
D9A1C9564F",

        "__comment_hash_key_src":"hash key from partition table FW1/FW2 (must
match type in partition item)",
        "hash_key_src" : "FW1",

        "__comment_images":"offset = null => cascade ( align to 64 ), should be zero if
valid",
        "images":[
            {"img": "FWHS_S", "offset":"0x00"},
            {"img": "FWHS_NSC", "offset":"0x00"},
            {"img": "FWHS_NS", "offset":"0x00"},
            {"img": "XIP_S_C", "offset":"0x00"},
            {"img": "XIP_S_P", "offset":"0x00"},
            {"img": "XIP_NS_C", "offset":"0x00"},
            {"img": "XIP_NS_P", "offset":"0x00"}
        ]
    },
    "FWHS_S": {
        "source":"Debug/Exe/application_s.dbg.out",
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
```

```
                "type":"FWHS_S",
                "enc":true,

"user_key2":"BB0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1
D1E1F",
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 100
            },
             "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",

                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":true,
                "cipherkey":null,
                "cipheriv":null
             },
            "list" : ["sram","psram"],
            "sram": {
                "secthdr":{
                     "type": "SRAM"
                },
                "__comment_option":"TODO: not ready",
                "option": null,

                "__comment_entry":"startup function table symbol",
                "entry":"gRamStartFun",

                "sections":   ["FIRMWARE_FUNTAB*", "FIRMWARE_SIGN*",
"FIRMWARE_SRAM_RO*", "FIRMWARE_SRAM_RW*"],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            },
```

```
            "psram": {
                "secthdr":{
                    "type": "PSRAM"
                },
                "__comment_option":"TODO: not ready",
                "option": null,

                "sections":   ["FIRMWARE_ERAM_RO*", "FIRMWARE_ERAM_RW*"],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            }
        },
        "FWHS_NS": {
            "source":"Debug/Exe/application_ns.dbg.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"FWHS_NS",
                "enc":true,
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 0
            },
             "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",

                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":true,
                "cipherkey":null,
                "cipheriv":null
             },
```

```
            "list" : ["sram","vector","psram"],
            "sram": {
                "secthdr":{
                    "type": "SRAM"
                },
                "__comment_option":"TODO: not ready",
                "option": null,

                "__comment_entry":"startup function table symbol",

                "sections":   ["FIRMWARE_FUNTAB*", "FIRMWARE_SIGN*",
"FIRMWARE_SRAM_RO*", "FIRMWARE_SRAM_RW*"],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            },
            "vector": {
                "secthdr":{
                    "type": "SRAM"
                },
                "__comment_option":"TODO: not ready",
                "option": null,

                "sections":   ["FIRMWARE_VECTOR*"],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            },
            "psram": {
                "secthdr":{
                    "type": "PSRAM"
                },
                "__comment_option":"TODO: not ready",
                "option": null,

                "sections":   ["FIRMWARE_ERAM_RO*", "FIRMWARE_ERAM_RW*"],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            }
        },
        "FWHS_NSC": {
```

```
        "source":"Debug/Exe/application_s.dbg.out",
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
            "type":"XIP",
            "enc":true,
            "__comment_pkey_idx":"assign by program, no need to configurate",
            "serial": 0
        },
         "FST":{
            "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
            "__comment_FST1": "validpat is used for section header validation",
            "__comment_FST2": "hash_en/enc_en?",
              "enc_algorithm":"cbc",
            "hash_algorithm":"sha256",
            "part_size":"4096",

            "__comment_validpat": "use auto or dedicated value",
            "validpat":"0001020304050607",
            "hash_en":true,
            "enc_en":true,
            "cipherkey":null,
            "cipheriv":null
         },
        "list" : ["nsc"],
         "nsc": {
            "secthdr":{
                "type": "XIP",
                "xip_key": "A0D6DAE7E062CA94CBB294BF896B9F68",
                "xip_iv": "948794879487948794879487948794879487"
            },
            "__comment_option":"TODO: not ready",
            "option": null,

            "__comment_entry":"XIP text, RO_data",

            "sections":   ["FIRMWARE_NSC*"],
```

```
                "__comment_file":"TODO: use binary file directly",
                "file": null
            }
        },
        "XIP_S_C": {
            "source":"Debug/Exe/application_s.dbg.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"XIP",
                "enc":true,
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 0
            },
              "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",

                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":true,
                "cipherkey":null,
                "cipheriv":null
              },
            "list" : ["xip"],
            "xip": {
                "secthdr":{
                    "type": "XIP",
                    "xip_key": "A0D6DAE7E062CA94CBB294BF896B9F68",
                    "xip_iv": "9487948794879487948794879487948794879487"
                },
                "__comment_option":"TODO: not ready",
```

```
                "option": null,

                "__comment_entry":"XIP text, RO_data",

                "sections":   ["FIRMWARE_XIP_S_C*"],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            }
        },
        "XIP_S_P": {
            "source":"Debug/Exe/application_s.dbg.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"XIP",
                "enc":false,
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 0
            },
             "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",

                "__comment_validpat": "use auto or dedicated value",
                "validpat":"0001020304050607",
                "hash_en":true,
                "enc_en":false,
                "cipherkey":null,
                "cipheriv":null
             },
            "list" : ["xip"],
            "xip": {
                "secthdr":{
```

```
                "type": "XIP",
                "xip_key": "A0D6DAE7E062CA94CBB294BF896B9F68",
                "xip_iv": "9487948794879487948794879487948794879487"
            },
            "__comment_option":"TODO: not ready",
            "option": null,

            "__comment_entry":"XIP text, RO_data",

            "sections":   ["FIRMWARE_XIP_S_P*"],
            "__comment_file":"TODO: use binary file directly",
            "file": null
        }
    },
    "XIP_NS_C": {
        "source":"Debug/Exe/application_ns.dbg.out",
        "header":{
            "next":null,
            "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
            "type":"XIP",
            "enc":true,
            "__comment_pkey_idx":"assign by program, no need to configurate",
            "serial": 0
        },
         "FST":{
            "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
            "__comment_FST1": "validpat is used for section header validation",
            "__comment_FST2": "hash_en/enc_en?",
              "enc_algorithm":"cbc",
            "hash_algorithm":"sha256",
            "part_size":"4096",

            "__comment_validpat": "use auto or dedicated value",
            "validpat":"0001020304050607",
            "hash_en":true,
            "enc_en":true,
            "cipherkey":null,
```

```
                "cipheriv":null
            },
            "list" : ["xip"],
            "xip": {
                "secthdr":{
                    "type": "XIP",
                    "xip_key": "A0D6DAE7E062CA94CBB294BF896B9F68",
                    "xip_iv": "948794879487948794879487948794879487"
                },
                "__comment_option":"TODO: not ready",
                "option": null,

                "__comment_entry":"XIP text, RO_data",

                "sections":   ["FIRMWARE_XIP_C*"
                ],
                "__comment_file":"TODO: use binary file directly",
                "file": null
            }
        },
        "XIP_NS_P": {
            "source":"Debug/Exe/application_ns.dbg.out",
            "header":{
                "next":null,
                "__comment_type":"Support
Type:PARTAB,BOOT,FWHS_S,FWHS_NS,FWLS,ISP,VOE,WLN,DTCM,ITCM,SRAM,ERAM,XIP
,M0,CPFW",
                "type":"XIP",
                "enc":false,
                "__comment_pkey_idx":"assign by program, no need to configurate",
                "serial": 0
            },
              "FST":{
                "__comment_FST0": "enc_algorithm: cbc/ecb with cipher key",
                "__comment_FST1": "validpat is used for section header validation",
                "__comment_FST2": "hash_en/enc_en?",
                  "enc_algorithm":"cbc",
                "hash_algorithm":"sha256",
                "part_size":"4096",
```

```
            "__comment_validpat": "use auto or dedicated value",
            "validpat":"0001020304050607",
            "hash_en":true,
            "enc_en":false,
            "cipherkey":null,
            "cipheriv":null
        },
        "list" : ["xip"],
        "xip": {
            "secthdr":{
                "type": "XIP",
                "xip_key": "A0D6DAE7E062CA94CBB294BF896B9F68",
                "xip_iv": "948794879487948794879487948794879487"
            },
            "__comment_option":"TODO: not ready",
            "option": null,

            "__comment_entry":"XIP text, RO_data",

            "sections":   ["FIRMWARE_XIP_P*"
            ],
            "__comment_file":"TODO: use binary file directly",
            "file": null
        }
    }
}
```

There is no need to do any change of elf2bin.exe. The ".exe" file is just called to build the encrypted image.

*key.json* file is generated by the encrypted image building process. Every time the image is encrypted, the *key.json* file will be updated.

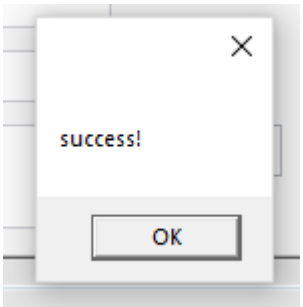
In Debug folder, there is only a folder named *Exe*. In *Exe* folder, there are three files, *application_s.dbg.out, application_ns.dbg.out, bootloader.out.* The *application_s.dbg.out* and *application_s.dbg.out* are found under \project\realtek_amebaz2_v0_example\EWARM-RELEASE\Debug\Exe where you build the plain image. Copy the files from \project\realtek_amebaz2_v0_example\EWARM-RELEASE\Debug\Exe to Exe folder, the tool will use this file to generate encrypted image. The *bootloader.out* should also be replaced by the version you build the plain image under

\component\soc\realtek\8710c\misc\bsp\image.

| | | | |
|---|---|---|---|
| application_is.dbg.out | 10/29/2020 5:37 PM | Wireshark capture... | 8,071 KB |
| application_ns.dbg.out | 10/8/2020 6:07 PM | Wireshark capture... | 7,696 KB |
| application_s.out | 10/8/2020 5:51 PM | Wireshark capture... | 1,230 KB |
| bootloader.out | 10/29/2020 2:34 PM | Wireshark capture... | 514 KB |

### 5.1.3. Encrypt

After choosing the way gotten keys and finishing configuration of encryption, click Encrypt button to encrypt the image.
If the image is encrypted successfully, a message box will turn up to show “success” as followed.




The encrypted *partition.bin*, *bootloader.bin*, *firmware_is.bin* and *flash_is.bin/flash_tz.bin* will be generated under Encrypt Tool/Debug/Exe. The *flash_is.bin/flash_tz.bin* is encrypted image that generated by the encrypt tool.

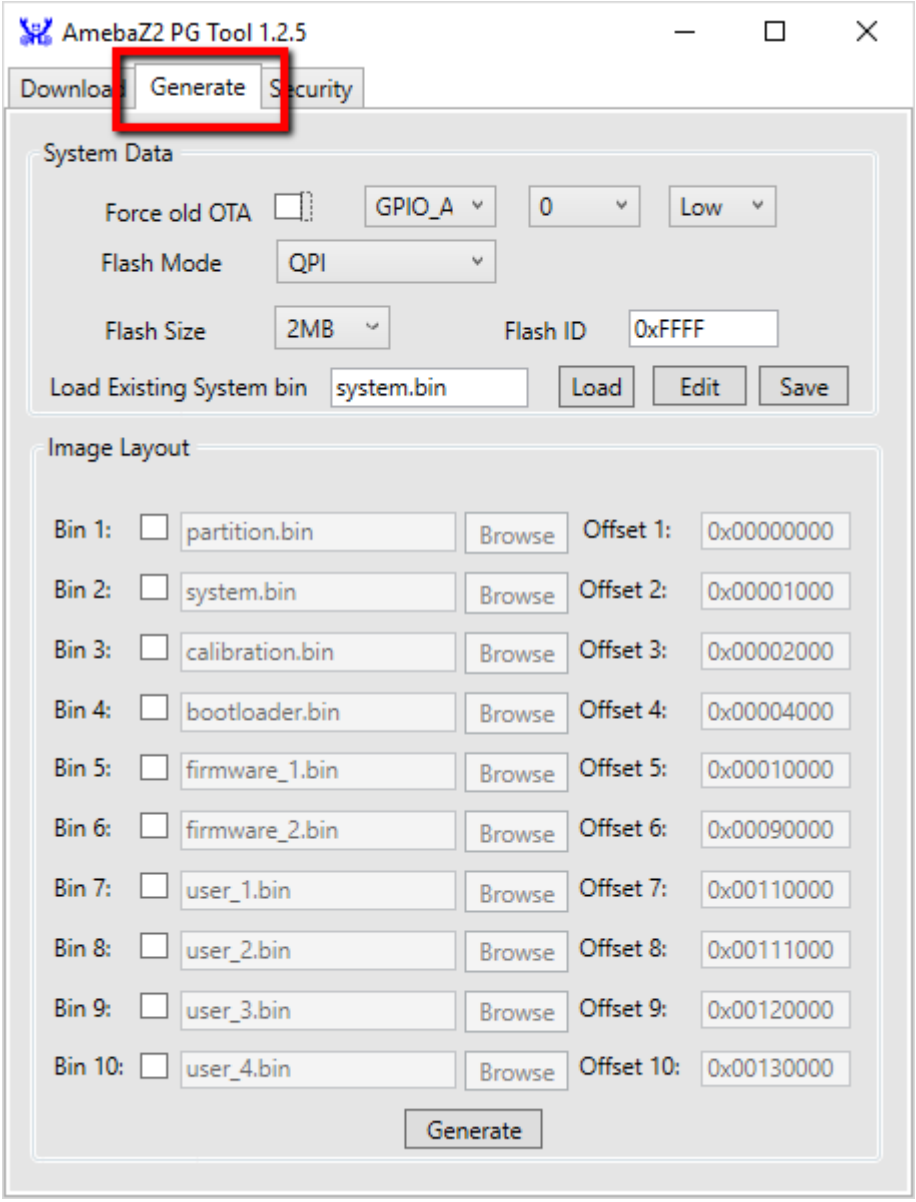
| | | | |
|---|---|---|---|
| application_is.dbg.out | 10/29/2020 5:37 PM | Wireshark capture... | 8,071 KB |
| application_ns.dbg.out | 10/8/2020 6:07 PM | Wireshark capture... | 7,696 KB |
| application_s.out | 10/8/2020 5:51 PM | Wireshark capture... | 1,230 KB |
| bootloader.bin | 11/2/2020 5:36 PM | BIN File | 17 KB |
| bootloader.out | 10/29/2020 2:34 PM | Wireshark capture... | 514 KB |
| firmware_is.bin | 11/2/2020 5:35 PM | BIN File | 437 KB |
| firmware_tz.bin | 11/2/2020 5:36 PM | BIN File | 516 KB |
| flash_is.bin | 11/2/2020 5:35 PM | BIN File | 501 KB |
| flash_tz.bin | 11/2/2020 5:36 PM | BIN File | 580 KB |
| partition.bin | 11/2/2020 5:36 PM | BIN File | 1 KB |

## 5.2. Program

Program is a function that writes Enc Key and Hash Key to efuse and enables secure boot and boots with encrypted image.

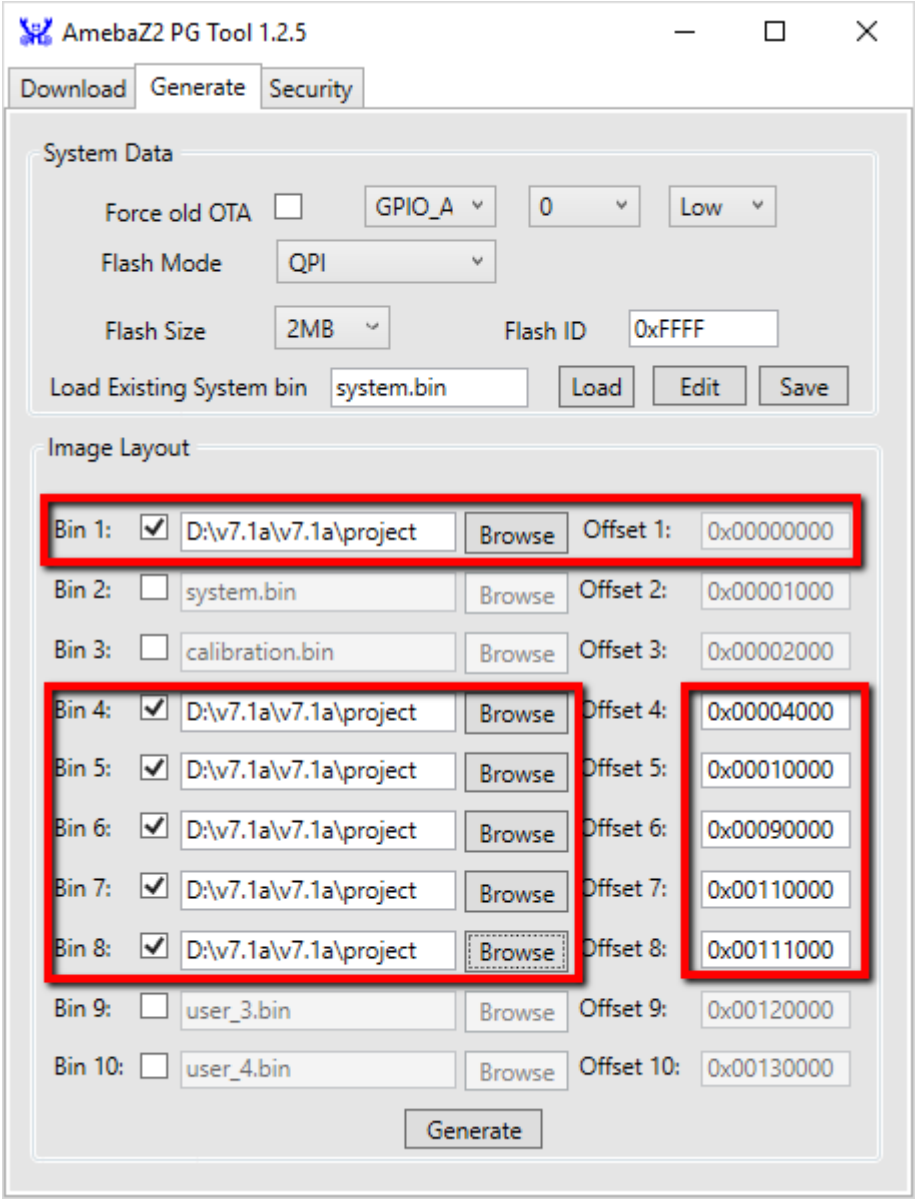
### 5.2.1. Generate image

Move to “Generate” tab

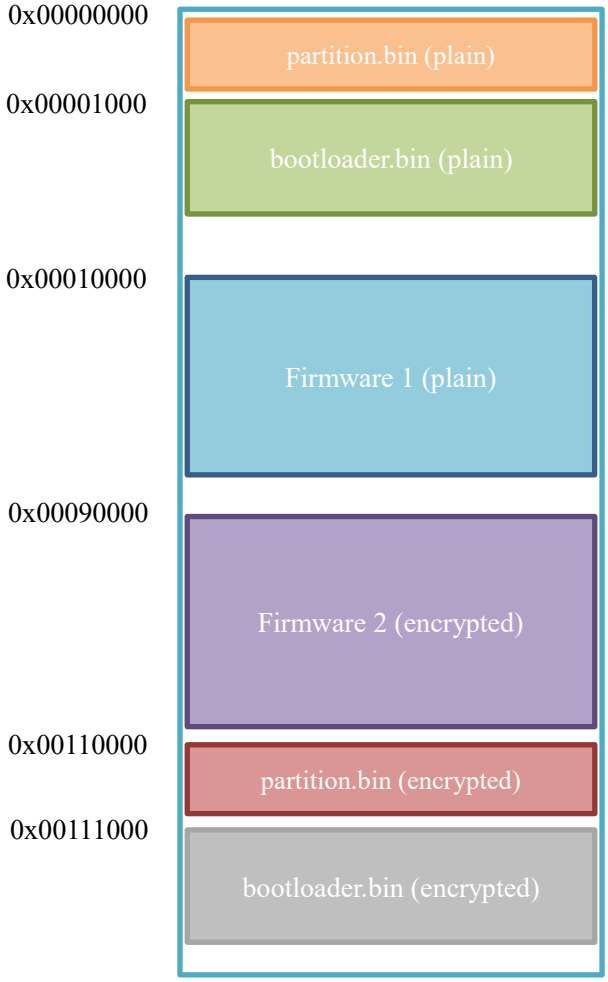



Generate the image with format “partition.bin (plain)” + “bootloader.bin (plain)” + “firmware_is.bin (plain)” + “firmware_is.bin (encrypted)” + “partition.bin (encrypted)” + “bootloader.bin (encrypted)”. The address of every section can be defined by user. During the process of program, it will write keys to efuse first, then erase the plain partition.bin and bootloader.bin, rewrite them with encrypted partition.bin and bootloader.bin which user

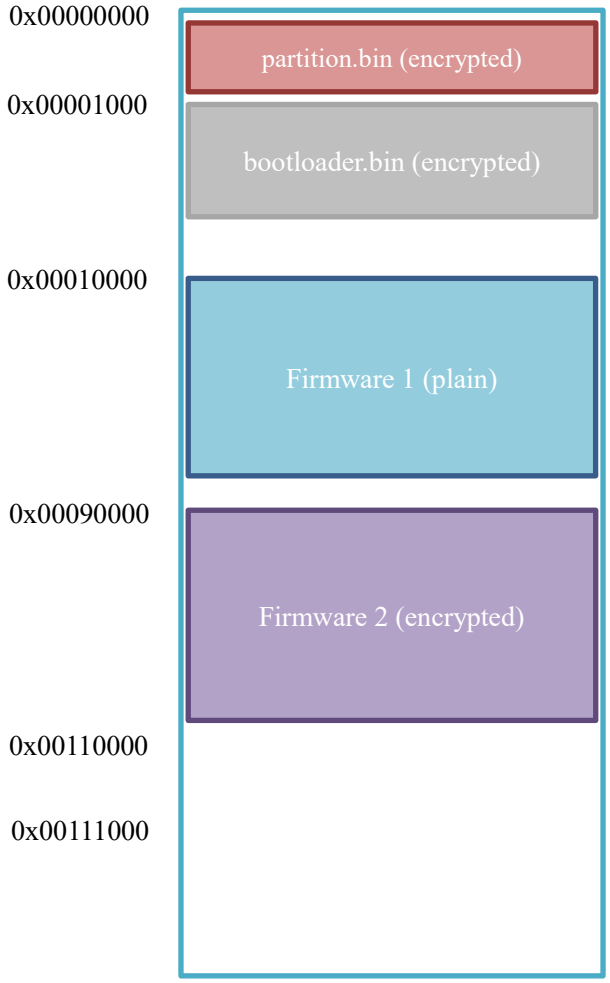
concat in the image, then enable secure boot, finally reboot with encrypted image.



After combination, the image layout is shown as below.



After Program, the image layout will be shown as below.

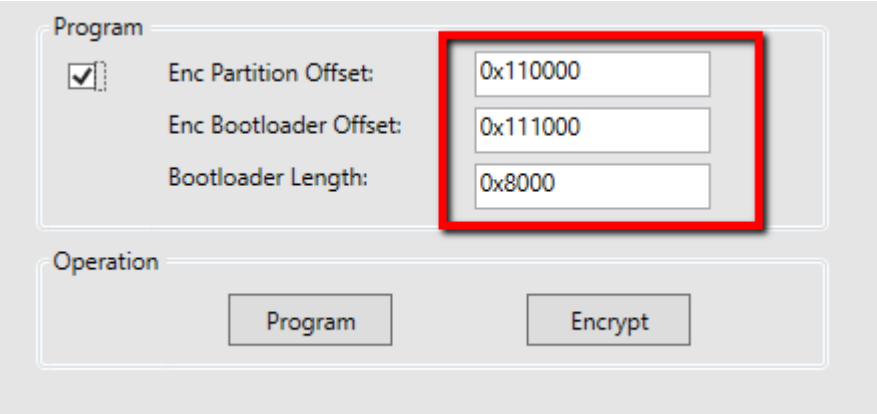


### 5.2.2. Get Keys

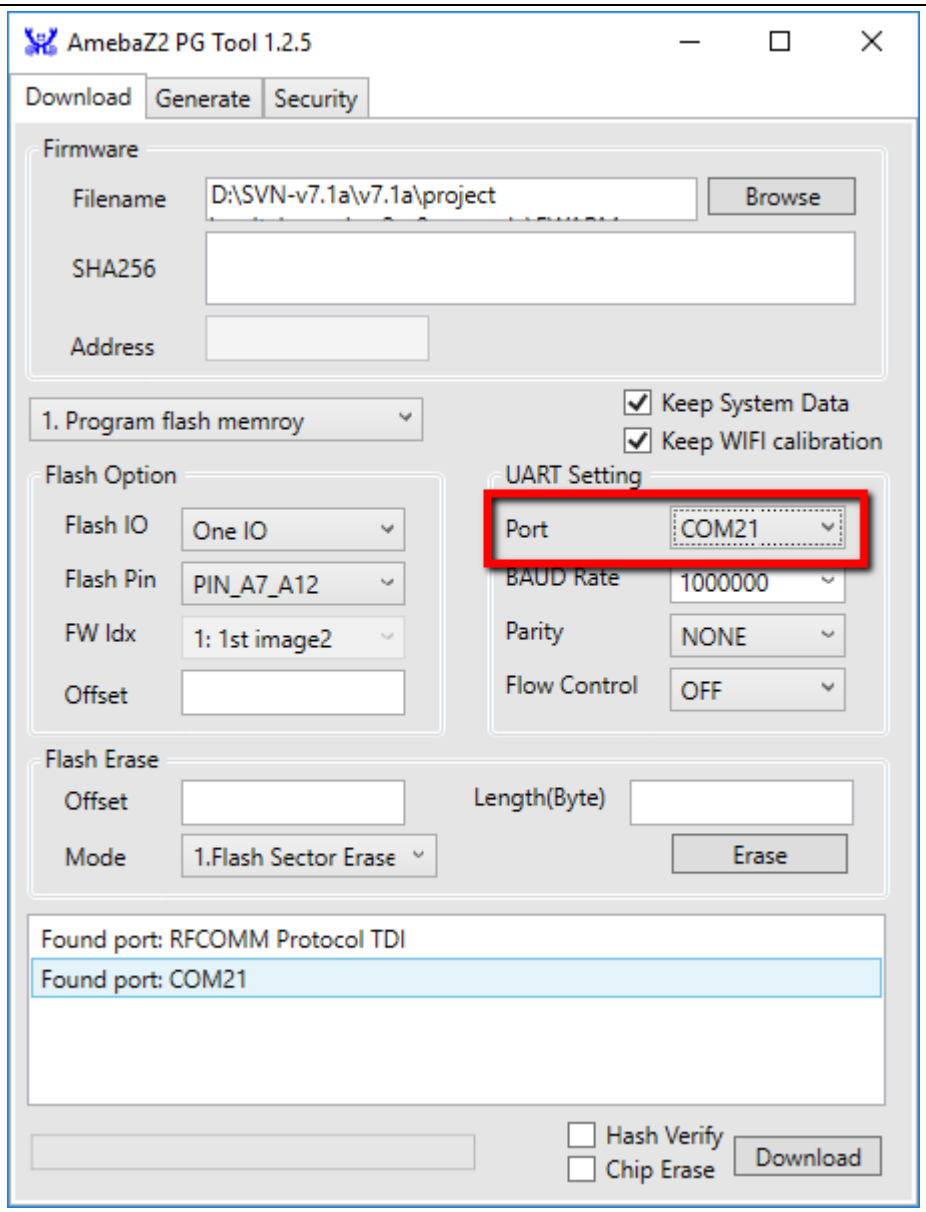
The ways that get the keys are the same with Encrypt. Please refer to chapter 5.1.1.

### 5.2.3. Program

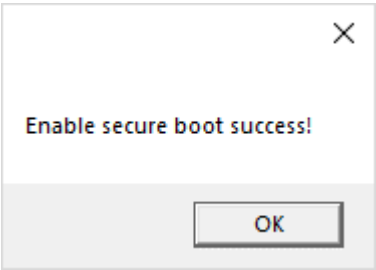
Click the check box to enable Program function. There are three parameters that you need to fill here. The “Enc Partition Offset” is the offset of encrypted partition.bin. The “Enc Bootloader Offset” is the offset of encrypted bootloader.bin. The “Bootloader Length” is the length of encrypted bootloader.bin.




Before click Program button, please make sure that the Port is selected to the right one in case Program need send command to chip. The other thing you need to confirm is, the chip has already been reset after downloading image.



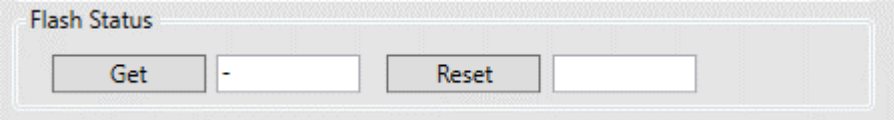

When everything is ready, please click Program to write keys to efuse and enable secure boot and boot with encrypted image.



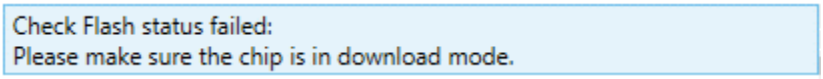

If the Program process is successful, the window which contains “Enable secure boot success!” will turn up. If the message window shows some error, please check the details in the log file which named by date and time under the working directory.

# 6. Flash Status

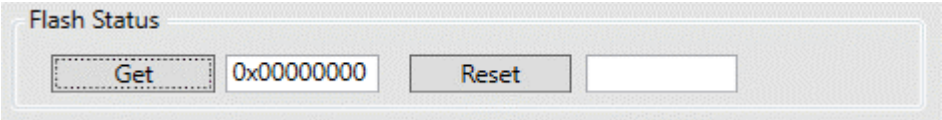
The tool supports get and set flash status.



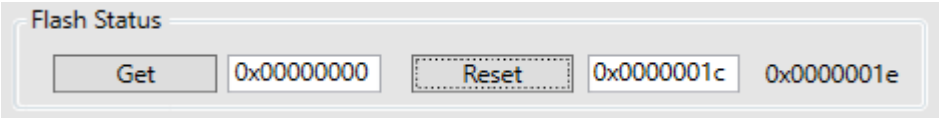

You can get flash status by clicking “Get” button and make sure the chip is in download mode, otherwise you will get the error as follows:




If you get the flash status successfully, the result will show in text box besides Get button.




Flash Status is a 4-byte value, of which Byte-0 is the value of status register, byte-1 is the value of status register 2, byte-2 is the value of status register 3 if applicable, byte-3 is reserved.
You can set the flash status by clicking Reset button with given value in the text box besides the Reset button. Please check the datasheet of flash chip for details of status register.




Note: this feature is only for debugging when flash can’t be erased or programmed. Please make sure you understand the value before you set the value to flash status register.
And this feature is NOT available if secure boot is already enabled (Check AN0500 for more information about secure boot) because of security reason.

# 7. Linux version PGTool

## 7.1. Command Line Mode

Now Linux version PGTool only supports command line mode. And the command line mode only supports normal download compared with Windows version. Open terminal in Linux and execute AmebaZII_PGTool_Linux.exe with defined parameters.

The supported parameters can be achieved by type "**-help**" as follows,

$ ./AmebaZ2_PGTool_Linux.exe -help

```
usage:
    -show [device|setting|generate]
        E.g. -show device
    -set image <path>
        E.g. -set image D:\test\flash_is.bin
    -set baudrate <baudrate>
        E.g. -set baudrate 1000000
    -set hash_verify <hash_verify_option>
        E.g. -set hash_verify 1
    -set keep_sys_data <sys_data_option>
        E.g. -set keep_sys_data 1
    -set keep_wifi_data <wifi_data_option>
        E.g. -set keep_wifi_data 1
    -set flash_offset <flash_offset_value>
        E.g. -set flash_offset 0x00000000
    -set chip_erase <chip_erase_value>
        E.g. -set chip_erase 1
    -scan device
    -add device <device>
        E.g. -add device /dev/ttyUSB0
    -erase chip
        E.g. -erase chip
    -erase sector <offset> <length>
        E.g. -erase sector 0x00004000 1000
    -download
    -set generate_bin <number of bin> <path> <offset>
        E.g. -set generate_bin 1 D:\test\partition.bin 0x0000
    -generate
```

### 7.1.1. Devices Connections

You can use "**-scan device"** parameter to check serial ports connected to PC as follows,

$ ./AmebaZ2_PGTool_Linux.exe -scan device

```
Path scanned...
Available device port: /dev/ttyUSB0
```

You can use "**-add device**" to add device port for download

$ ./AmebaZ2_PGTool_Linux.exe -add device /dev/ttyUSB0

### 7.1.2. Configuration

- To check the tool configuration, you can use "**-show setting**" as parameters.

$ ./AmebaZ2_PGTool_Linux.exe -show setting

```
CMD_download_setting
 Image file path:        NaN
 Baudrate:               1000000
 Chip Erase:             0
 Hash Verify:            1
 Keep sys data:          0
 Keep wifi data:         0
 Flash Offset:           0x0000
```

- To choose file to download, you can use "**-set image <path>**" as parameters.

$ ./AmebaZ2_PGTool_Linux.exe -set image /home/user/flash.bin

- To set the downloading start address of the flash, you can use "**-set flash_offset <flash_offset_value>".**

$ ./AmebaZ2_PGTool_Linux.exe -set flash_offset 0x00000000

- To set the downloading baudrate, you can use "**-set baudrate <baudrate>**".

$ ./AmebaZ2_PGTool_Linux.exe -set baudrate 1000000

- To enable/disable hash verify function, you can use **"-set hash_verify <hash_verify option>"**.

$ ./AmebaZ2_PGTool_Linux.exe -set hash_verify 1

- To enable/disable chip erase function, you can use **"-set chip_erase <chip_erase**

**value>”**.

$ ./AmebaZ2_PGTool_Linux.exe -set chip_erase 1

- To enable/disable keep system data function, you can use **“-set keep_sys_data <sys_data_option>”**.

$ ./AmebaZ2_PGTool_Linux.exe -set keep_sys_data 0

### 7.1.3. Flash Erase

In order to successful apply the flash erase function, please enter the PG mode before apply any command.

- Please hold the UART download button and presses reset button to let the board enter the PG mode. (Please refer to ***AN0500 Realtek Ameba-ZII application note.en***)

#### 7.1.3.1 Flash Sector Erase

You can use “**-erase sector <offset> <length>**” to erase sector of the device. **<offset>** is to set the start address of the flash. **<length>** is to set the length of the erase area. Note that a sector is 4K bytes, the length should be Integer multiple of the 4K bytes.

$ ./AmebaZ2_PGTool_Linux.exe -erase sector 0x00001000 0x1000

```
Ping received, beginning erase
device Configs set successfully
Erase sec cmd:seras 00000000 0x1000 0 0

Erase successful
```

#### 7.1.3.2 Flash Chip Erase

You can use “**-erase chip**” to erase chip of the device added.

$ ./AmebaZ2_PGTool_Linux.exe -erase chip

```
Ping received, beginning erase
device Configs set successfully
Received:OK length:2
Erase successful
```

### 7.1.4. Download image

After all configuration is finished, you can use "**-download**" to download image to device added.

$ ./AmebaZ2_PGTool_Linux.exe -download

```
Beginning XModem Download...
Ping received
device Configs set successfully
Erasing chip...
Received:OK length:2
Erase successful

Start download
Xmodem Downloading...
Downloading ---- 100%
Recv OK successful
Xmodem successfully transmitted 450560 bytes
HashVerify success
End Download
```

## 7.2. Image Generation

You can use **"-set generate_bin <number of bin> <path> <offset>"** to set the number of bin, the path of the bin and the offset of the bin file. And use **"-generate"** to do the image generation.

```
$ ./AmebaZII_PGTool_Linux_v1.0.9 -set generate_bin 1 /home/[illegible]/partition.bin 0x0000
$ ./AmebaZII_PGTool_Linux_v1.0.9 -set generate_bin 2 /home/[illegible]/bootloader.bin 0x4000
$ ./AmebaZII_PGTool_Linux_v1.0.9 -set generate_bin 3 /home/[illegible]/firmware_is.bin 0x9000
$ ./AmebaZII_PGTool_Linux_v1.0.9 -generate
```

There will be a file named flash.bin generated in working directory. Please note that, the tool supports maximum 10 bin files' stitching, and you can use **"-show generate"** to check the setting of image generation.

```
CMD_generate_map
 1  /home/[illegible]/partition.bin  0x0000
 2  /home/[illegible]/bootloader.bin  0x4000
 3  /home/[illegible]/firmware_is.bin  0x9000
 4  NaN  NaN
 5  NaN  NaN
 6  NaN  NaN
 7  NaN  NaN
 8  NaN  NaN
 9  NaN  NaN
 10  NaN  NaN
```