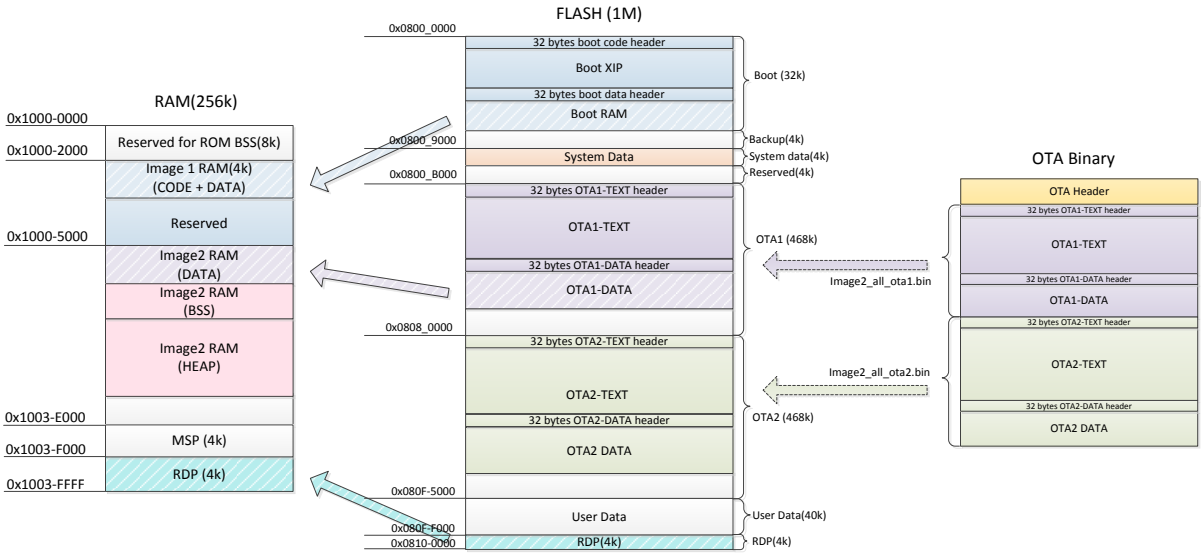
# Ameba-Z Memory Layout

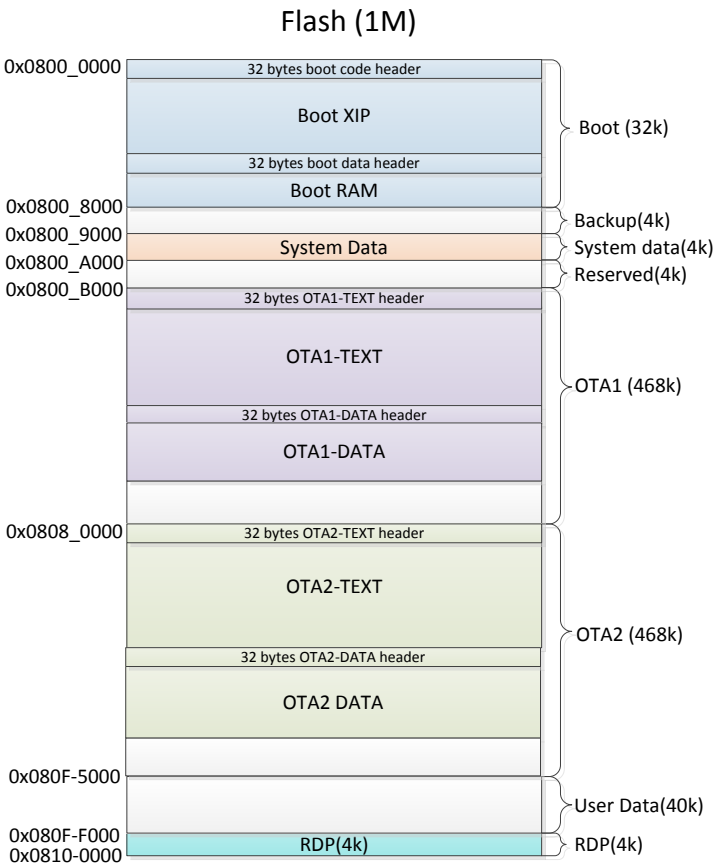
This document introduces usage of SRAM, Flash, and OTA.

Table of Contents

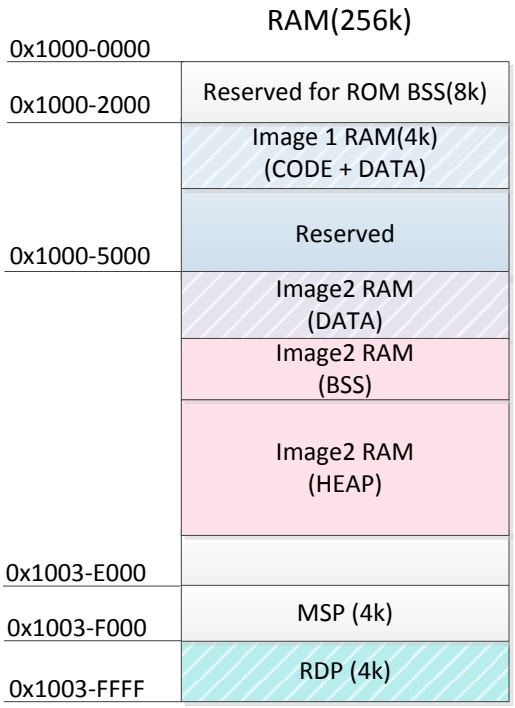
# 1. Memory layout



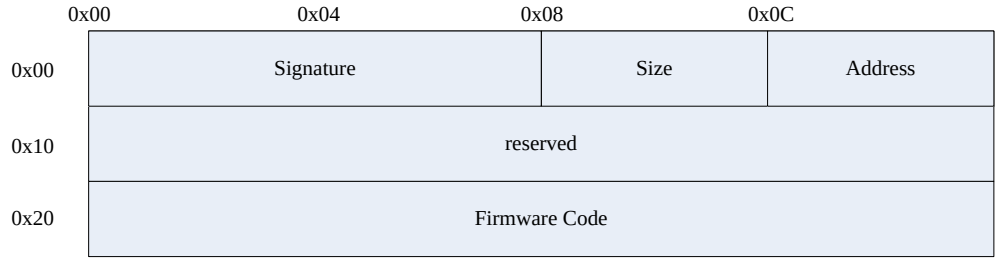

## 1.1. Flash Layout



## 1.2. SRAM layout

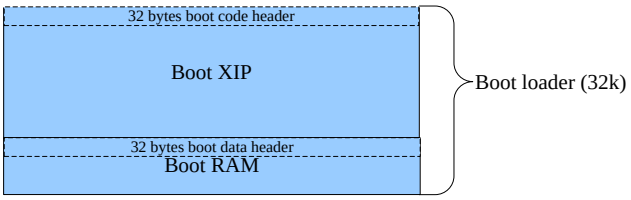



# 2. Image Header




- Signature:
  - Image1 Signature for Flash calibration: 8 bytes
  - Image2 Signature is string: “81958711”
- Size:
  - The size of image: 4 bytes
- Address:
  - Code executes address after boot.
  - ‘BOOT RAM’, ‘OTA1 DATA’ ‘OTA2 DATA’ is target RAM address
  - ‘BOOT XIP’. ‘OTA1 TEXT’, ‘OTA2 TEXT’ is Flash XIP address

# 3. Boot Loader (Image 1)




- Boot_all.bin
- Hardware initialization
- Call Image 2 Firmware based on system data
- Bootloader RAM means this part will be load to RAM.
- Bootloader XIP means this part will be run in flash.
- Should run independently, should not depend on image2.

# 4. System Data (4k)

- system.bin is generated by image tool.

| | 0x00 | 0x04 | 0x08 | 0x0C |
|---|---|---|---|---|
| 0x00 | OTA2 Flash Address | Valid IMG2 | Forth OTA1 GPIO | OTA Rsvd |
| 0x10 | RDP Flash Address | RDP Len (no checksum 4B) | RDP Rsvd | RDP Rsvd |
| 0x20 | WORD1: SPI Speed WORD0: SPI Mode | WORD1: Flash Size WORD0: Flash ID | Flash Rsvd | Flash Rsvd |
| 0x30 | ULOG BaudRate | ULOG Rsvd | ULOG Rsvd | ULOG Rsvd |
| 0x40 | Reserved for Realtek | | | |
| | reserved | | | |
| 0x100 ~ 0x147 | USB Parameter | | | |
| | reserved | | | |
| 0x200 | ADC Prameter | | | |

## 4.1. OTA system data (0x00)

### 4.1.1. OTA2 Flash Address

- Flash address for OTA2.

- If 0xFFFFFFFF, select OTA1

### 4.1.2. Valid IMG2

- Used for OTA1 and OTA2 switch
  - LSB 0 bits number is odd: like 0xFFFFFFFE/0xFFFFFFF8... means select OTA2
  - LSB 0 bits number is even: like 0xFFFFFFFF/0xFFFFFFFC... means select OTA1

### 4.1.3. Force OTA1 GPIO

- GPIO force OTA1 as image2.
  - BIT[4:0]: pin num 0~31
  - BIT[5]: port: 0/1
  - BIT[7]: active_state 0/1

## 4.2. RDP system data (0x10)

- RDP Flash address:
  - Flash address for RDP
- RDP Len: RDP image length
  - 16 bytes alignment
  - Not include checksum 4bytes

## 4.3. Flash system data (0x20)

### 4.3.1. SPI Speed

- 0xFFFF: 100MHz
- 0x7FFF: 83MHz
- 0x3FFF: 71MHz
- 0x1FFF: 62MHz
- 0x0FFF: 55MHz
- 0x07FF: 50MHz
- 0x03FF: 45MHz

### 4.3.2. SPI Mode

- 0xFFFF: Read quad IO, Address & Data 4 bits mode
- 0x7FFF: Read quad O, Just data 4 bits mode
- 0x3FFF: Read dual IO, Address & Data 2 bits mode
- 0x1FFF: Read dual O, Just data 2 bits mode
- 0x0FFF: 1 bit mode

### 4.3.3. Flash size

- 0xFFFF: 2MB
- 0x7FFF: 32M
- 0x3FFF: 16M
- 0x1FFF: 8MB
- 0x0FFF: 4MB
- 0x07FF: 2MB
- 0x03FF: 1MB

### 4.3.4. Flash ID

- Reserved, use it when flash ID cannot get from flash ID cmd.

## 4.4. LOG UART system data (0x30)

### 4.4.1. ULOG BaudRate

- 0xFFFFFFFF: 115200
- 115200~3000000

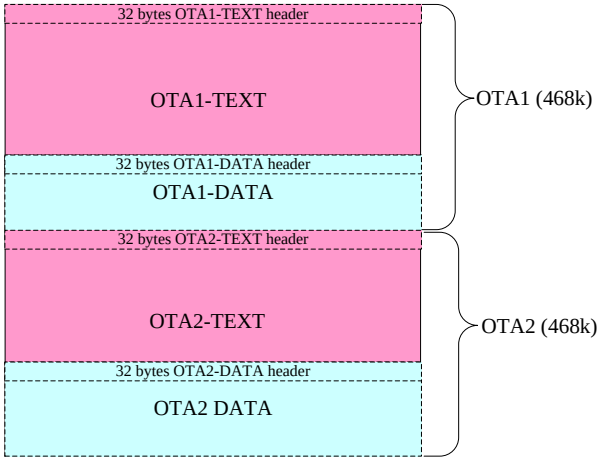
## 4.5. USB Parameter (0x100~0x147)

- Just used for USB dongle mode (AN0117)

## 4.6. ADC Parameter

- ADC calibration data

# 5. OTA-X (Image 2)




- Image2_all_ota1.bin / image2_all_ota2.bin
- OTA-X-TEXT: OTA-X code and RO data, this will not be loaded to RAM
- OTA-X-DATA: OTA-X DATA, this will be loaded to RAM.

## 5.1. Image2.icf

Image2.icf defines different content in different places.

```
/*****************************************
 * Memory Regions                        *
 *****************************************/
define symbol __ICFEDIT_region_ROM_start__              = 0x00000000;
define symbol __ICFEDIT_region_ROM_end__                = 0x0007FFFF;
define symbol __ICFEDIT_region_ROMBSS_RAM_start__       = 0x10000000;
define symbol __ICFEDIT_region_ROMBSS_RAM_end__         = 0x10001FFF;
define symbol __ICFEDIT_region_BOOTLOADER_RAM_start__   = 0x10002000;
define symbol __ICFEDIT_region_BOOTLOADER_RAM_end__     = 0x10004FFF;
define symbol __ICFEDIT_region_BD_RAM_start__           = 0x10005000;
define symbol __ICFEDIT_region_BD_RAM_end__             = 0x1003DFFF;
define symbol __ICFEDIT_region_MSP_RAM_start__          = 0x1003E000;
define symbol __ICFEDIT_region_MSP_RAM_end__            = 0x1003EFFF;
define symbol __ICFEDIT_region_RDP_RAM_start__          = 0x1003F000;
define symbol __ICFEDIT_region_RDP_RAM_end__            = 0x1003FFEF;
define symbol __ICFEDIT_region_IMG2_TEMP_start__        = 0x10006000;
define symbol __ICFEDIT_region_IMG2_TEMP_end__          = 0x1000BFFF;
define symbol __ICFEDIT_region_XIP_BOOT_start__         = 0x08000000+0x20;
define symbol __ICFEDIT_region_XIP_BOOT_end__           = 0x08003FFF;
define symbol __ICFEDIT_region_XIP_OTA1_start__         = 0x0800B000+0x20;
define symbol __ICFEDIT_region_XIP_OTA1_end__           = 0x080FFFFF;
```

## 5.2. Data default in SRAM

As shown below, all data defaults to the SRAM area.

```
/*************************************
 * BD RAM Section config             *
 *************************************/
keep { section .image2.entry.data* };
keep { section .image2.validate.rodata* };
define block .ram_image2.entry with fixed order{ section .image2.entry.data*,
                                                 section .image2.validate.rodata*,
                                                 };
define block SHT$$PREINIT_ARRAY { preinit_array };
define block SHT$$INIT_ARRAY { init_array };
define block CPP_INIT with fixed order { block SHT$$PREINIT_ARRAY,
                                         block SHT$$INIT_ARRAY };
define block .ram.data with fixed order{ section .data*,
                                         section DATA,
                                         section .iar.init_table,
                                         section __DLIB_PERTHREAD,
                                         block CPP_INIT,
                                         section .mdns.data,
                                         };
```

## 5.3. Text default in XIP

As shown below, all text defaults to the Flash area.

```
/*************************************
 * XIP OTA1 Section config           *
 *************************************/
define block .xip_image2.text with fixed order{ section .img2_custom_signature*,
                                                section .text*,
                                                section .mdns.text,
                                                section .rodata*,
                                                section .debug_trace,
                                                section CODE,
                                                section Veneer, // object startup.o,
                                                };
```

## 5.4. Put object in SRAM

If it is an object, you can use the following method to place it in the SRAM area. Use lxbus_ops.o and lib_rtlstd.a as example:

```
/**************************************
 * BD RAM Section config              *
 **************************************/
keep { section .image2.entry.data* };
keep { section .image2.validate.rodata* };
define block .ram_image2.entry with fixed order{ section .image2.entry.data*,
                                                 section .image2.validate.rodata*,
                                                 };
define block SHT$$PREINIT_ARRAY { preinit_array };
define block SHT$$INIT_ARRAY { init_array };
define block CPP_INIT with fixed order { block SHT$$PREINIT_ARRAY,
                                         block SHT$$INIT_ARRAY };
define block .ram.data with fixed order{ section .data*,
                                         section DATA,
                                         section .iar.init_table,
                                         section __DLIB_PERTHREAD,
                                         block CPP_INIT,
                                         section .mdns.data,
                                         };
define block .ram.text with fixed order{section .image2.ram.text*,
                                         section .text* object lxbus_ops.o,
                                         section .text* object lib_rtlstd.a,
                                         };
define block IMAGE2 with fixed order {  block .ram_image2.entry,
                                        block .ram.data,
                                        block .ram.text,
                                        };
```

## 5.5. Put code in SRAM

You can also define section to specify the location of the code or data. Use mdns as example:

```
/**************************************
 * BD RAM Section config              *
 **************************************/
keep { section .image2.entry.data* };
keep { section .image2.validate.rodata* };
define block .ram_image2.entry with fixed order{ section .image2.entry.data*,
                                                 section .image2.validate.rodata*,
                                                 };
define block SHT$$PREINIT_ARRAY { preinit_array };
define block SHT$$INIT_ARRAY { init_array };
define block CPP_INIT with fixed order { block SHT$$PREINIT_ARRAY,
                                         block SHT$$INIT_ARRAY };
define block .ram.data with fixed order{ section .data*,
                                         section DATA,
                                         section .iar.init_table,
                                         section __DLIB_PERTHREAD,
                                         block CPP_INIT,
                                         section .mdns.data,
                                         };
define block .ram.text with fixed order{section .image2.ram.text*,
                                         section .mdns.text,
                                         };
define block IMAGE2 with fixed order {  block .ram_image2.entry,
                                        block .ram.data,
                                        block .ram.text,
                                        };
```

# 6. User Data

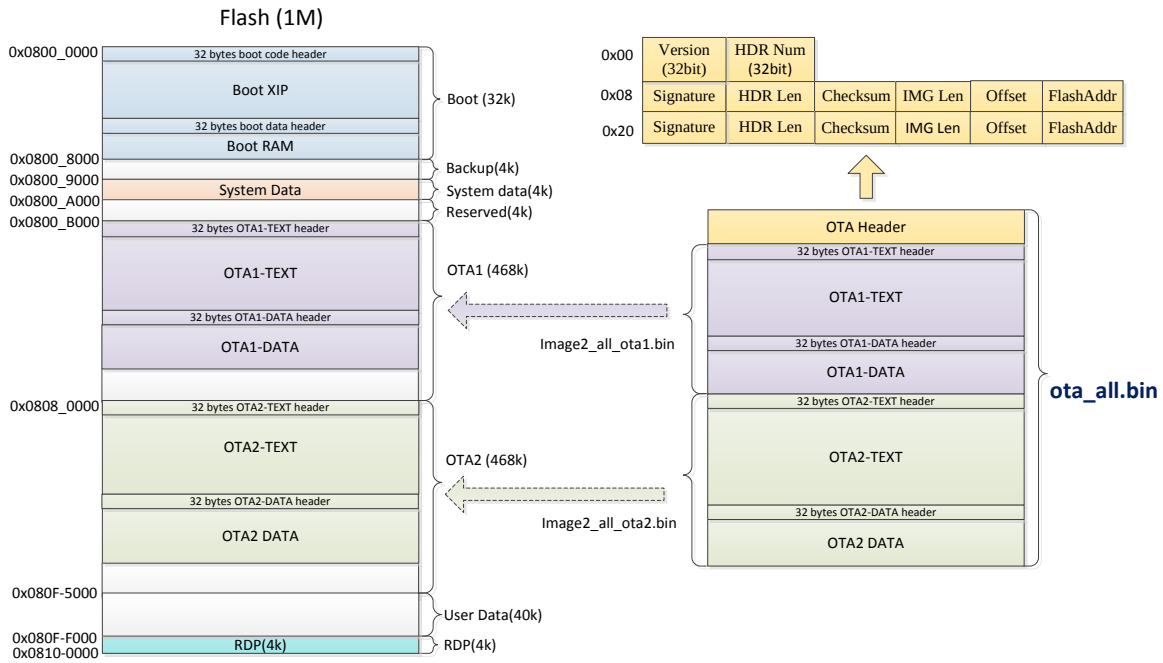
- This section of flash memory is used by application like WIFI profile.

# 7. RDP Encrypt Area (4k)

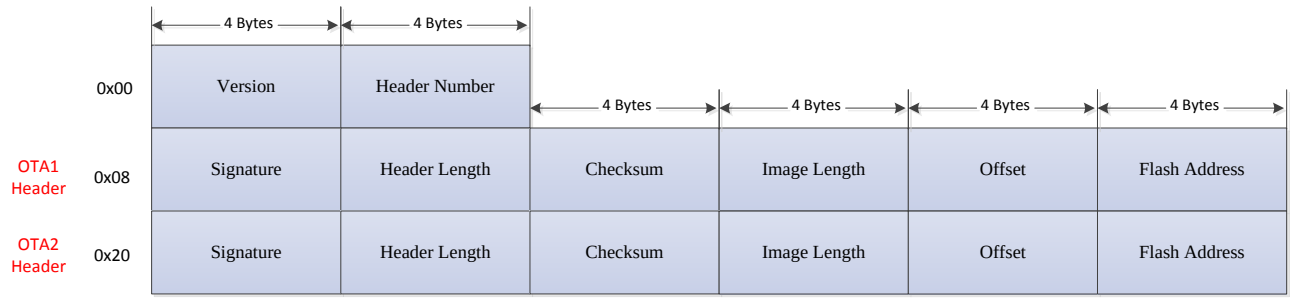
- This section of flash memory is used for FW protection.
- Flash memory address:
  - Always bottom 4k of flash
- RAM address
  - Always bottom 4k of CM4 RAM

# 8. OTA update

Please reference AN0110_Realtek_Ameba-Z_over_the_air_firmware_update for more detail.



## 8.1. OTA Header




- Version:
  - The version of OTA image: 4 bytes
- Header Number:
  - The number of OTAx Header : 4 bytes
- Signature:
  - OTAx Signature is string: “OTAx”, 4 bytes.
  - For example, “OTA1” for OTA1, and “OTA2 ” for OTA2.
- Header Length
  - The length of OTAx header, 4bytes
- Checksum
  - The checksum of OTAx image, 4 bytes
- Image Length:
  - The size of OTAx image: 4 bytes
- Offset:
  - The start position of OTAx in current image: 4 bytes
- Flash Address:
  - Address in flash where OTAx will be programmed, 4 bytes

# 9. Memory Usage Evaluation

8710Bx-Ax series have internal 256K SRAM and external Flash selected by customers. 8710Bx-Lx series only have internal 200K SRAM. The following example takes 8710BN-A0 and sdk-ameba-4.0a as an example to calculate the free SRAM and the used Flash.

User can refer “application.map” to observe them after project build. This file can be found in your project “project\realtek_amebaz_va0_example\EWARM-RELEASE\Debug\Exe’ folder.

## 9.1. Free SRAM

SRAM for OTA-X(Image 2) is from 0x10005000 to 0x1003dfff, shown as below:

```
"A4":  place at start of [0x10005000-0x1003dfff] {
          rw, block IMAGE2, block .ram_image2.bss, block .ram_image2.skb.bss,
```

After linking the actual size of the “A4” is 0x27228.

```
"A4":                                          0x27228
  IMAGE2                          0x10005000    0x10c8  <Block>
```

***Free SRAM (not linked) = 0x1003dfff – 0x10005000 – 0x27228 = 0x11dd7***

In the already used SRAM space, we assign a large area for use as a Heap, the size of the Heap can be adjusted by modifying the value of the configTOTAL_HEAP_SIZE. So after the Wi-Fi completes initialization, the remaining Heap is also part of the Free SRAM.

```
#
WIFI initialized

init_thread(53), Available heap 0xe2d0
```

***Free SRAM = Free SRAM (not linked) + Free Heap = 0x11dd7 + 0xe2d0 = 131239***

## 9.2. Flash Used

Flash for OTA-X(Image 2) is from 0x0800b020 to 0x080fffff, shown as below:

```
"A6":  place at start of [0x0800b020-0x080fffff] { rw, block .xip_image2.text };
```

After linking the actual size of the "A6" is 0x45580.

```
"A6":                                          0x45580
  .xip_image2.text                0x0800b020    0x45580  <Block>
```

Boot loader plus Realtek reserved for a total of 44K.

***Flash Used = 44K + 0x45580 = 321K***