

REALTEK

AN0300

AmebaPro application note

Abstract

AmebaPro is a high-integrated IC. Its features include 802.11 Wi-Fi, H.264 video codec, Audio Codec. This manual introduce users how to develop AmebaPro , including SDK compiling and downloading image to AmebaPro.

Table of Contents

1	<i>Compiling and downloading.....</i>	<i>5</i>
1.1	SDK Project introduction	5
1.2	Compile program	6
1.2.1	Compile big CPU	6
1.2.2	Compile little CPU	6
1.2.3	Generating image (Bin)	6
1.3	Using image tool to download	7
1.4	Using JTAG/SWD to debug	9
1.4.1	JTAG/SWD connection	9
1.5	Use uart to observe the log of the little cpu	12
2	<i>Development Board</i>	<i>13</i>
2.1	Choose Image Sensor	13
2.1.1	OV2735	13
2.1.2	SC2232	13
3	<i>Flash Layout.....</i>	<i>14</i>
3.1	Flash Layout overview.....	14
4	<i>How to use example source code.....</i>	<i>15</i>
4.1	Application example source	15
4.2	Peripheral example source	15
4.3	Wi-Fi example source.....	15
4.3.1	Use AT command to connect WLAN	15
4.3.2	WLAN scenario example	16
4.4	Video example source.....	16
5	<i>Memory configuration and usage</i>	<i>17</i>
5.1	Memory type.....	17
5.1.1	The size and configuration in AmebaPro(big CPU)	17
5.1.2	The size and configuration in AmebaPro(little CPU)	19
5.2	Memory Configuration.....	20
5.2.1	Configure memory in IAR	20
5.2.2	Configure memory in ICF file.....	21
5.2.3	Memory overflow	21

5.3	Video in DRAM	21
5.4	Unmovable program	22
5.4.1	XIP-related Flash API	22
5.5	Counting system of memory.....	22
6	OTA	23
6.1	OTA operation flow.....	24
6.2	Boot process flow	25
6.3	Upgraded partition	26
6.4	Firmware image output.....	27
6.4.1	OTA file format.....	27
6.4.2	OTA firmware swap behavior	28
6.4.3	Configuration for building OTA firmware	29
6.5	Implement OTA over Wi-Fi	31
6.5.1	OTA using local download server base on socket	31
6.5.2	OTA using local download server based on HTTP	34
6.6	OTA signature	37
7	Power Management	38
7.1	AmebaPro Power Save Modes.....	38
7.1.1	DeepSleep	38
7.1.2	Standby	39
7.1.3	SleepPG	39
7.1.4	Wakeup from WLAN	40
7.2	Examples	41
7.2.1	Example: power_save	41
7.2.2	Example: system suspend with wake on wlan after streaming stop	43
8	System.....	47
8.1	ICC (Inter CPUs Communication).....	47
8.1.1	Example: ICC	47
8.2	System reset	47
8.2.1	ls_sys_reset	47
8.2.2	sys_reset	47
9	Hardware Peripherals	48
9.1	GTimer	48
9.1.1	gtimer_rtc	48
9.1.2	SNTP	48

9.2	RTC.....	48
9.3	PWM	48
9.4	Efuse	49
9.4.1	User OTP	49
10	<i>File system</i>	50
10.1	FAT Filesystem on Flash	50
10.1.1	Software Setup	50
10.2	Dual FAT Filesystem - File system on both SD Card and Flash.....	51
10.2.1	Hardware Setup	51
10.2.2	Software Setup	51

1 **Compiling and downloading**

This chapter introduces users how to develop AmebaPro. AmebaPro is composed of one main board, one sensor board, and one daughter board with LED, light sensor, and IR-LED. AmebaPro SDK provides all the example source code for the function mentioned above.

To get start, users will need to set up the software to program the board.

IAR IDE provides the toolchain for AmebaPro. It allows users to write programs, compile and upload them to your board. Also, it supports step-by-step debug. Realtek also provides Image Tool for users to do downloading code process.



Note : Please use IAR version 8.30.

1.1 **SDK Project introduction**

Currently users can use ignore secure mode. Project_is(ignore secure) is the project without Arm TrustZone technology. This project is easier to develop and suit for first-time developer.

1.2 Compile program

AmebaPro use the newest Big-Little architecture. Big CPU is 300MHz, supporting high speed function like WiFi, ISP, Encoder and Codec. Little CPU is 4MHz, supporting low power peripheral function. Big CPU supports power-save mode while little CPU is operating. Big CPU power-save mode can be awaked by event trigger. Since the big CPU will depend on the setting of small CPU, it is necessary to compile the small CPU before the big CPU.



Communication between big CPU and little CPU is based on share memory. There are some examples in SDK explaining how this works. Please refer to Inter Channel Communication.

1.2.1 Compile big CPU

Step1. Open SDK/project/realtek_amebapro_v0_example/EWARM-RELEASE/Project_is.eww.

Step2. Confirm application_is in WorkSpace, right click application_is and choose "Rebuild All" to compile.

Step3. Make sure there is no error after compile.

1.2.2 Compile little CPU

Step1. Open SDK/project/realtek_amebapro_v0_example/EWARM-RELEASE/Project_lp.eww.

Step2. Confirm application_lp in WorkSpace, right click application_lp and choose "Rebuild All" to compile.

Step3. Make sure there is no error after compile.

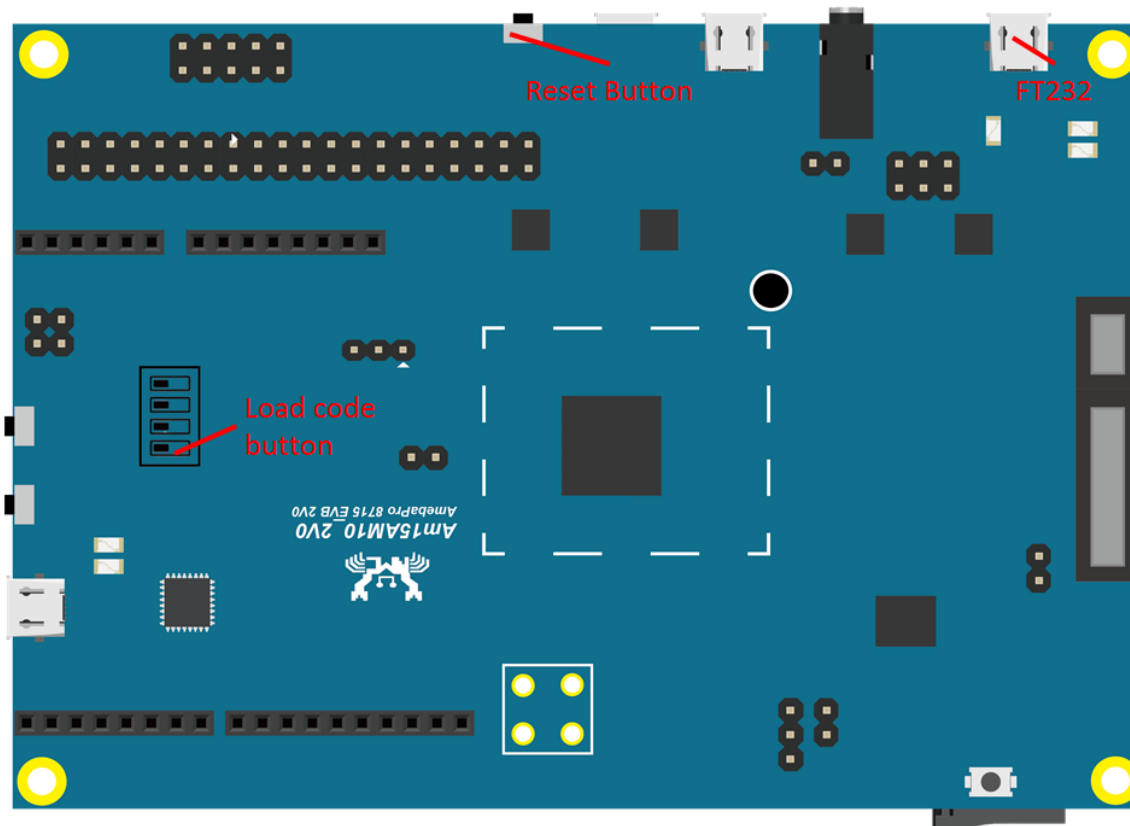
1.2.3 Generating image (Bin)

After compile, the images partition.bin, boot.bin, firmware_is.bin and flash_is_ota1.bin can be seen in the EWARM-RELEASE\Debug\Exe.

Partition.bin stores partition table, recording the address of Boot image and firmware image. Boot.bin is bootloader image; firmware_is.bin is application image, flash_is_ota1.bin links partition.bin, boot.bin and firmware_is.bin. Users need to choose flash_is_ota1.bin when downloading the image to board by PG Tool.

1.3 Using image tool to download

- a. Before using PG tool, user should connect AmebaPro to PC and switch to download mode:



Step 1: connect FT232 to PC

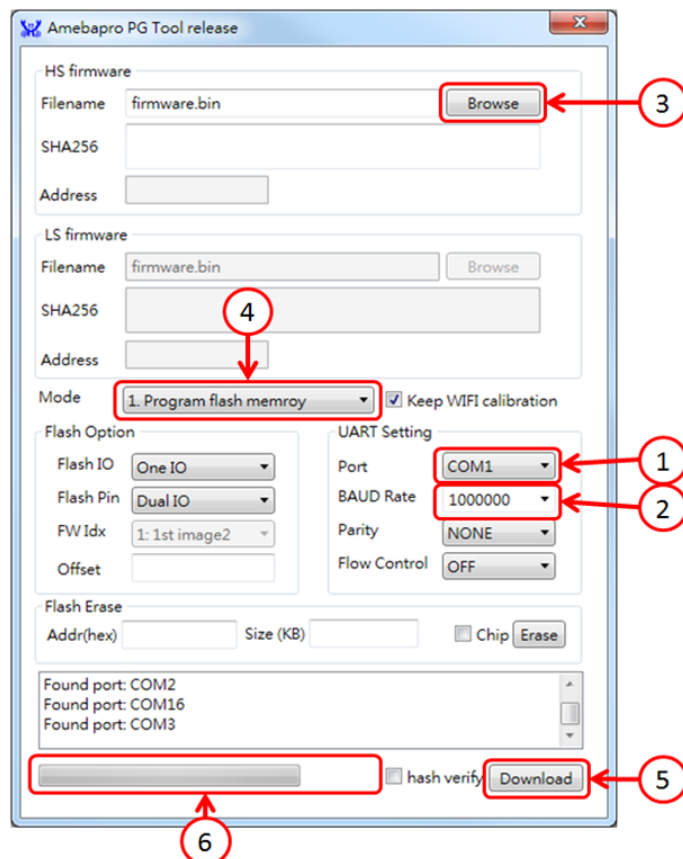
Step 2: enter test mode: put load code button on the graph to ON and reboot by power button

Step 3: Use AmebaPro PG Tool to download image to AmebaPro(reference part b.)

Step 4: after downloading, load code button switch OFF and reboot AmebaPro. The log can be seen in console.

(p.s. To observe the log of the little cpu, please refer to 1.5 Using uart to observe the log of the little cpu.)

b. AmebaPro PG Tool usage:



Step 1: AmebaPro PG Tool will automatically detect UART ports , choose UART ports of AmebaPro

Step 2: choose baud rate

p.s. default using parity check and flow control

Step 3: choose the image “flash_xx.bin”

Step 4: choose the mode “ Program flash memory”

Step 5: click Download to download the image to board

Step 6: the progress will show in the bar and the success/fail result will show here

NOTE: other setting please keeps default , Flash IO : “One IO” , Flash Pin : “Dual IO” ,

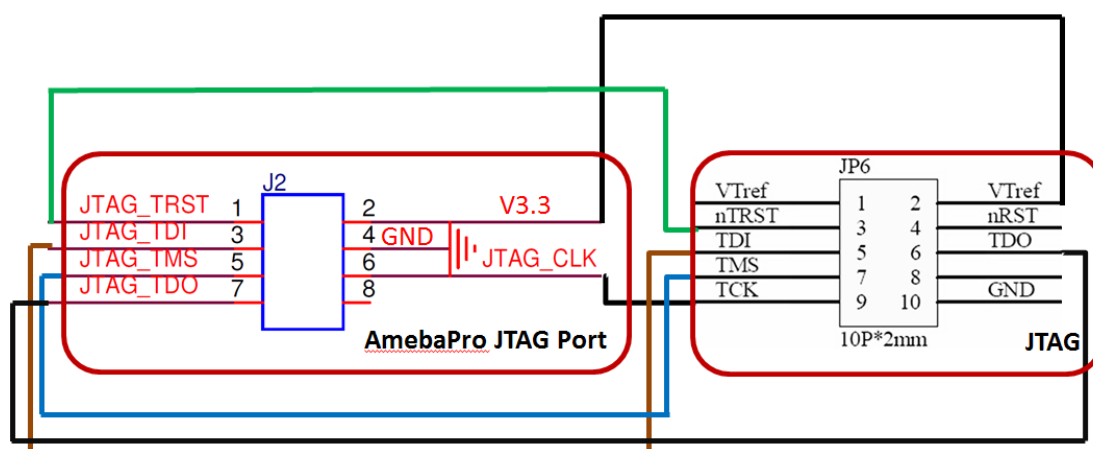
Parity: “NONE”, Flow Control : “OFF”

1.4 Using JTAG/SWD to debug

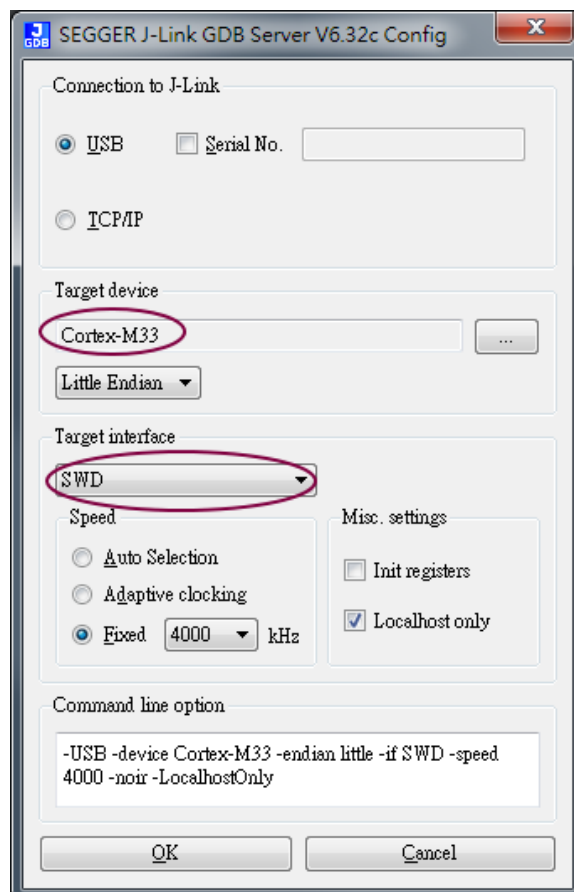
JTAG/SWD is a universal standard for chip internal test. The external JTAG interface has four mandatory pins, TCK, TMS, TDI, and TDO, and an optional reset, nTRST. JTAG-DP and SW-DP also require a separate power-on reset, nPOTRST. The external SWD interface requires two pins: bidirectional SWDIO signal and a clock, SWCLK, which can be input or output from the device.

1.4.1 JTAG/SWD connection

Make sure the six pin (VTref、TCK、TMS、TDI、TDO and nTRST) connect well done as the graph below, if using SWD, please check four pin (VTref[VDD]、TMS[SWDIO]、TCLK[SWCLK] and TDO[SWO]).



After connection, open J-Link GDB server. Choose target device Cortex-M33(for AmebaPro), and target interface JTAG / SWD. Click OK.



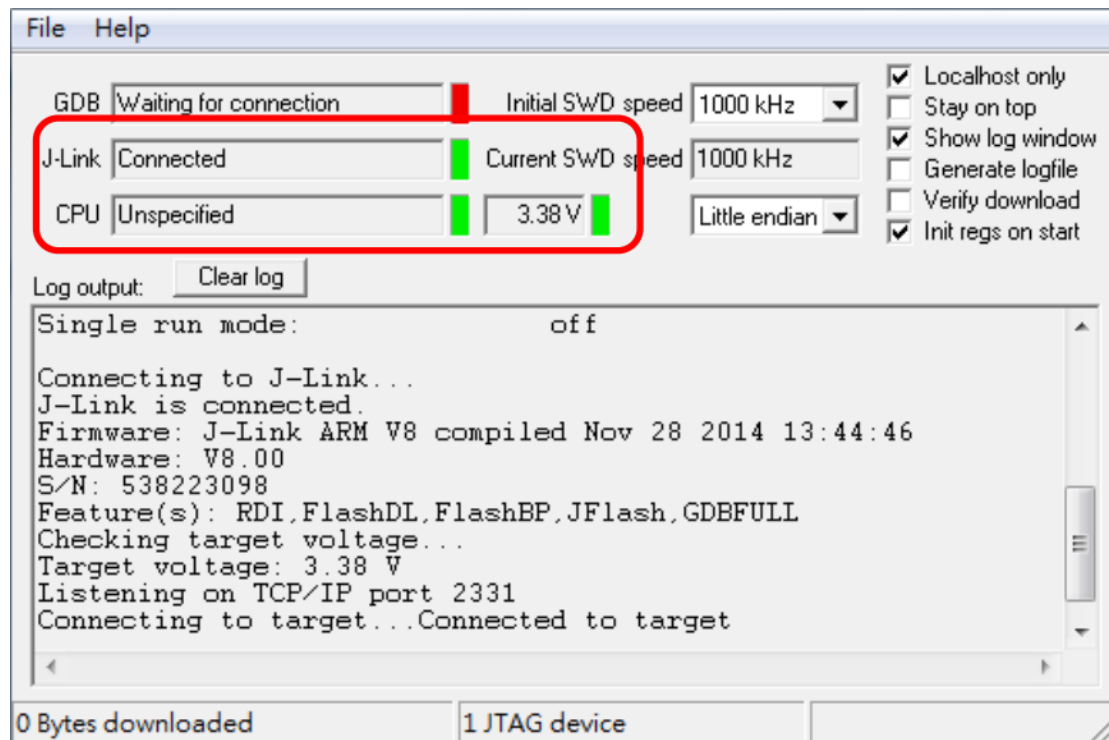
Note that CMSIS-DAP have two limitations in this situation:

- (1) CMSIS-DAP transmission speed is too slow.
- (2) System reset cannot be completely reset system can only reset HS core.

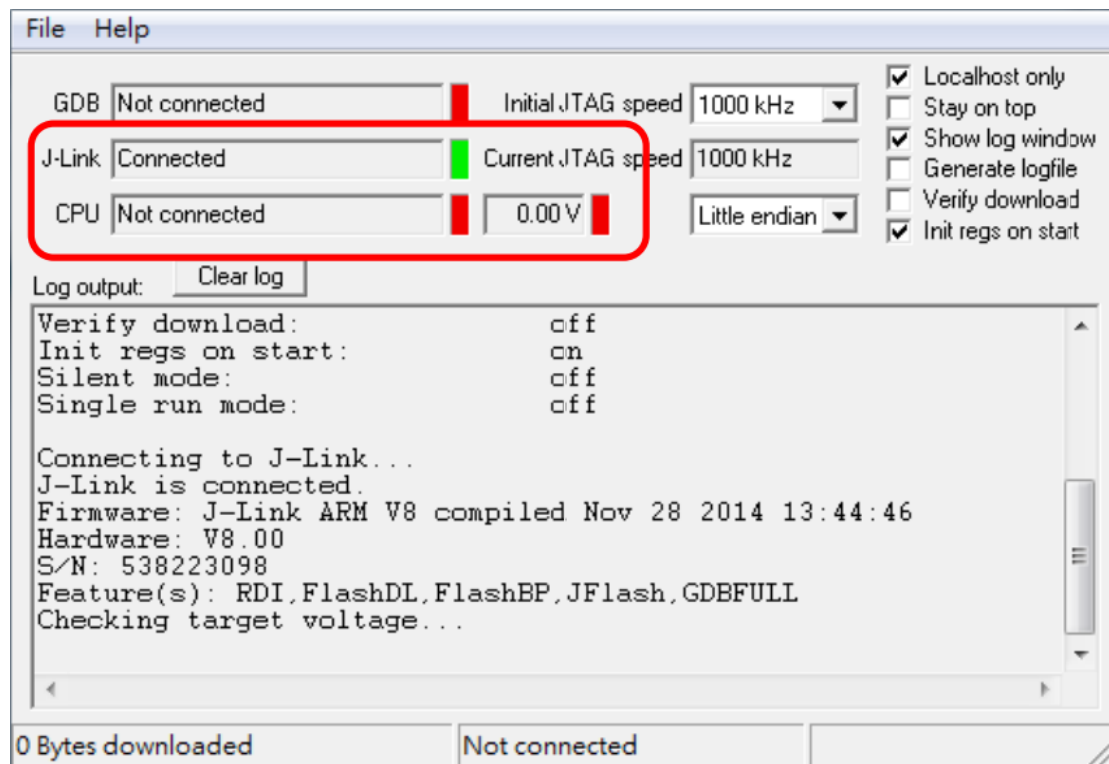
To use WATCHDOG RESET + SYSTEM RESET, only JLINK can be used. Can't be used in CMSIS-DAP.

It is recommended to use JLINK V9 or higher version. Currently, step in/out/over function is not supported. If want to set the interrupt position, set breakpoint directly at the point you want to see.

If connection succeeds, J-Link GDB server must show as below.

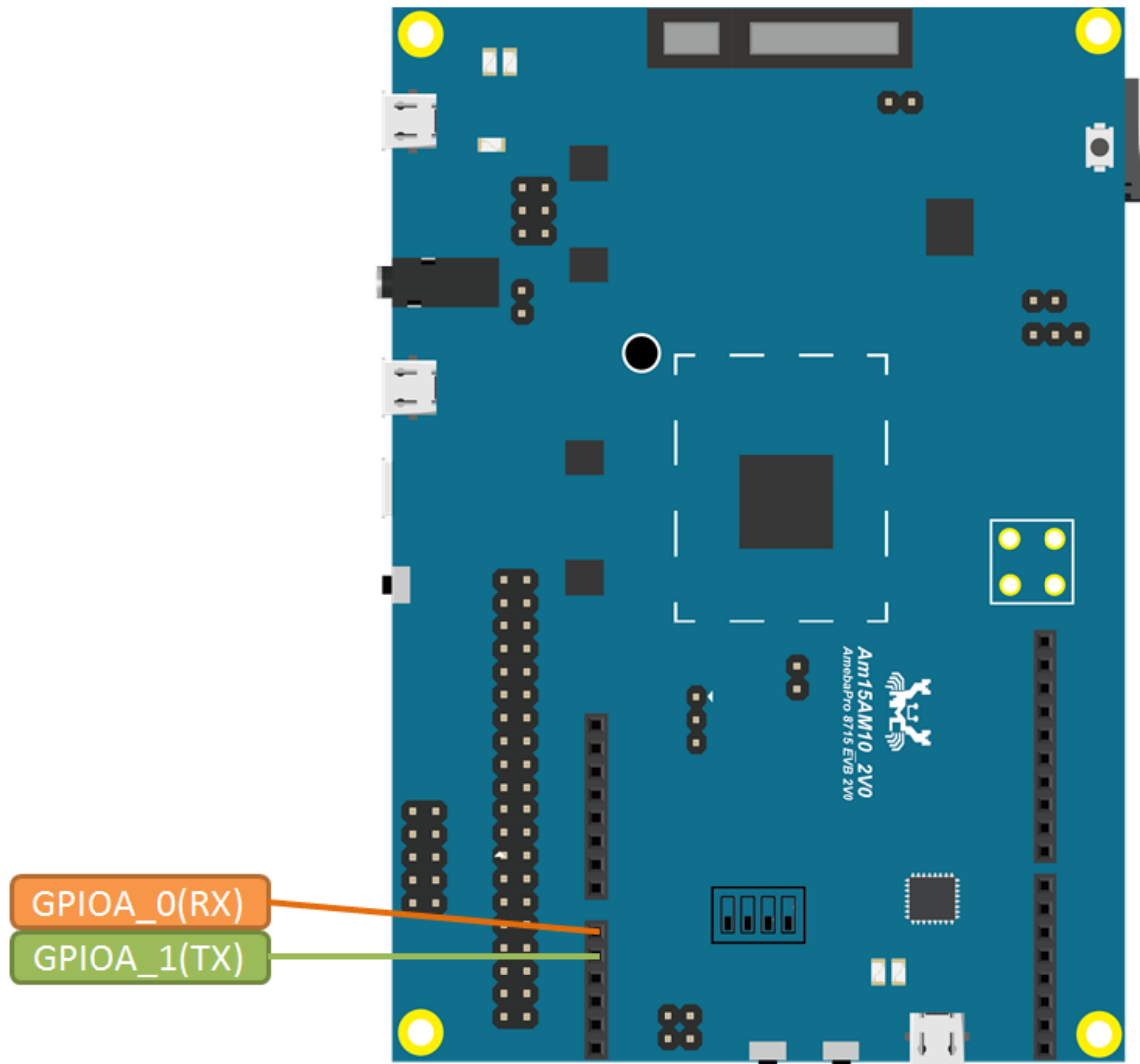


If connection fails, J-Link GDB will show:



1.5 Use uart to observe the log of the little cpu

In AmebaPro, the log of the big cpu and the little cpu is separate. The big cpu log can be observed through the FT232 or DAP, and the little cpu needs to be connected to the uart console through the A0 and A1 pins to observe the log of the small core.



2 Development Board

This chapter introduces users how to use AmebaPro Image sensor board. With correct hardware setting and software setting, image sensor board can work properly. Also, our SDK provide API to check whether sensor hardware matches software setting to prevent wrong manipulation.

2.1 Choose Image Sensor

This is sensor list that AmebaPro support:

AmebaPro EVAL	Sensor	Note
AmebaPro	OV2735	
AmebaPro	SC2232	

2.1.1 OV2735

Hardware setting

- Plug correct OV2735 sensor board

Software setting

- In project\realtek_amebapro_v0_example\inc\sensor.h
Set the definition: `#define SENSOR_USE Sensor_OV2735`
- In component\soc\realtek\8195b\misc\bsp\image
Delete original isp.bin, and rename isp_ov2735.bin to isp.bin

2.1.2 SC2232

Hardware setting

- Plug correct SC2232 sensor board

Software setting

- In project\realtek_amebapro_v0_example\inc\sensor.h
Set the definition: `#define SENSOR_USE Sensor_SC2232`
- In component\soc\realtek\8195b\misc\bsp\image
Delete original isp.bin, and rename isp_sc2232.bin to isp.bin

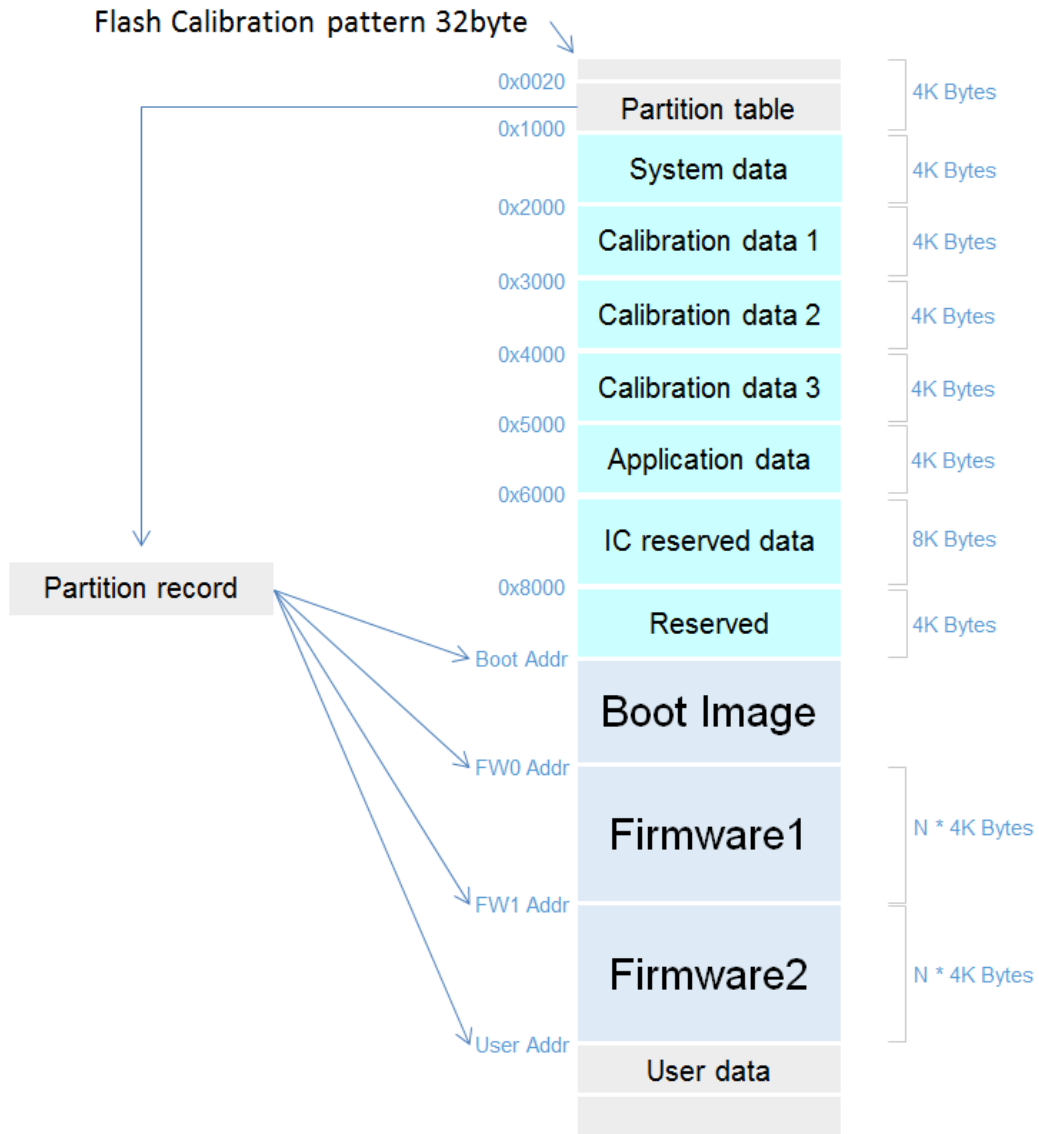


If users use the wrong hardware or software setting does not match hardware setting , API `video_sensor_check(SENSOR_USE)` will return -1

3 Flash Layout

AmebaPro use Big-Little architecture and sub-mcu system design, which provide high extensibility for developers.

3.1 Flash Layout overview



4 How to use example source code

4.1 Application example source

The examples for AmebaPro application is the SDK/common/example file. All the example provide related files including .c,.h, and readme. The readme file explains how to compile and important parameter.

After opening IAR, the first step is adding example source code(.c) into application_is -> utilities -> example(right click example and choose Add -> Add Files or drag-and-drop the file into it).

After adding example code, user should use platform_opts.h to switch on the example. For example, if users are going to use DCT function, compile flag CONFIG_EXAMPLE_DCT should be set to 1, which means

```
#define CONFIG_EXAMPLE_DCT 1
```

In platform_opts.h so that the example function in example_entry will execute . After this procedure, rebuild application_is project to execute the example.

4.2 Peripheral example source

Peripheral example source can help us utilize peripheral function. Peripheral example source code locates in SDK/project/realtek_amebapro_v0_example/example_sources. There are main.c and readme.txt in each example file. The main.c in the example should be used to replace original main.c(in SDK/project/realtek_amebapro_v0_example/src). The readme file explains how to compile and important parameter. After that, rebuild application_is project to execute the Peripheral example.

4.3 Wi-Fi example source

4.3.1 Use AT command to connect WLAN

AmebaPro provide AT command for user to test and develop. Users can key in AT command to connect WLAN by the console in PC. AT command can be referenced in AN0025 Realtek at command.pdf.

Wi-Fi direct GO and Concurrent mode are still under development, we will release in future version.

4.3.2 WLAN scenario example

AmebaPro provide WLAN scenario example for users to develop a variety of WiFi function, the path if example is

`\component\common\example\wlan_scenario\example_wlan_scenario.c`. This example provides many features including station mode, AP mode, Concurrent mode, WPS and P2P GO. The detail and the usage of the example can be read in `readme.txt`. Wi-Fi direct GO and Concurrent mode are still under development, we will release in future version.

4.4 Video example source

AmebaPro provide Multimedia Framework v2. Video example code is based on this architecture. Any detail about Multimedia Framework v2 architecture can be referenced in document in UM0301.

5 Memory configuration and usage

This chapter introduces the memory in AmebaPro, including OM, RAM, SRAM, TCM, DRAM, Flash. Also, this chapter provides the guide that users can place their program to the specific memory to fit user's requirement. However, some of program is fixed in specific memory and cannot be moved. This program is also discussed in this chapter. Reference the chapter Flash Layout if the memory is related to Flash.

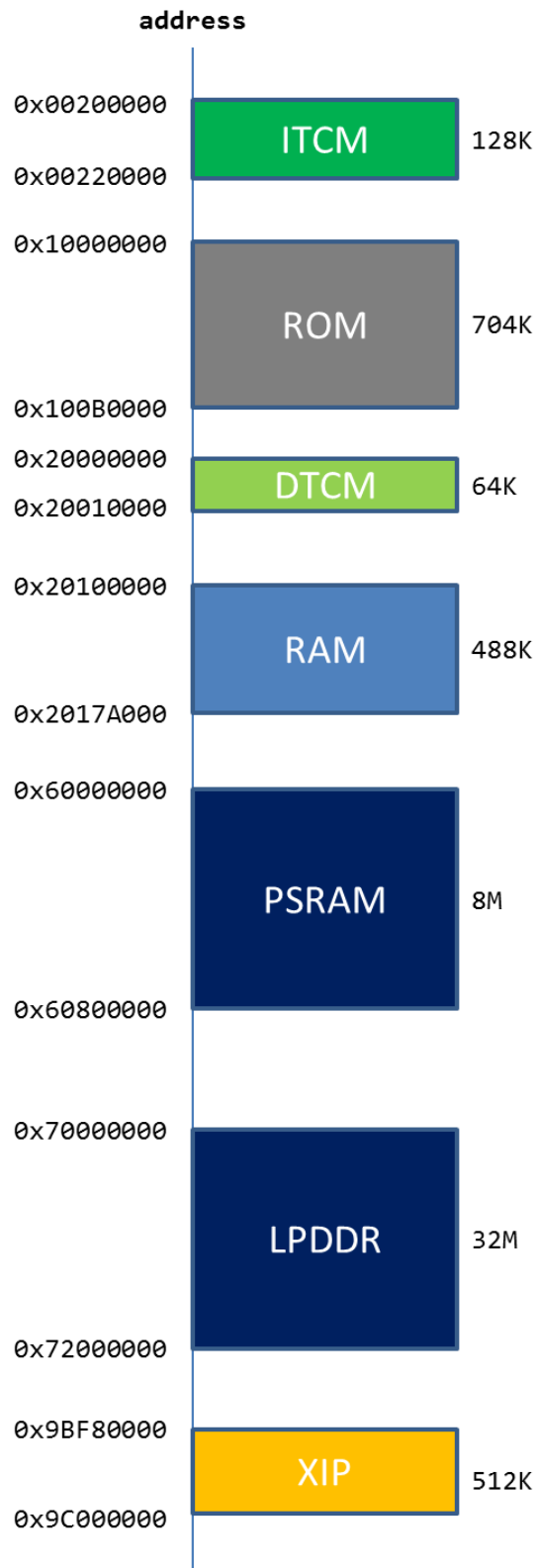
5.1 Memory type

5.1.1 The size and configuration in AmebaPro(big CPU)

The size and configuration in AmebaPro(big CPU) is as shown below

	Size(bytes)	Description
ITCM	128K	can place instruction
ROM	704K	
DTCM	64K	can place Read/write data
RAM	488K	SRAM which size is 128K
PSRAM	8M	Choose either PSRAM or LPDDR. Evaluation Reference Board is attached with LPDDR °
LPDDR	32M	Choose either PSRAM or LPDDR. Evaluation Reference Board is attached with LPDDR ° ° LPDDR is also called as DRAM
XIP	512K	Execute In Place, text section in Flash can be placed in XIP

The graph of configuration in AmebaPro big CPU is as shown below:

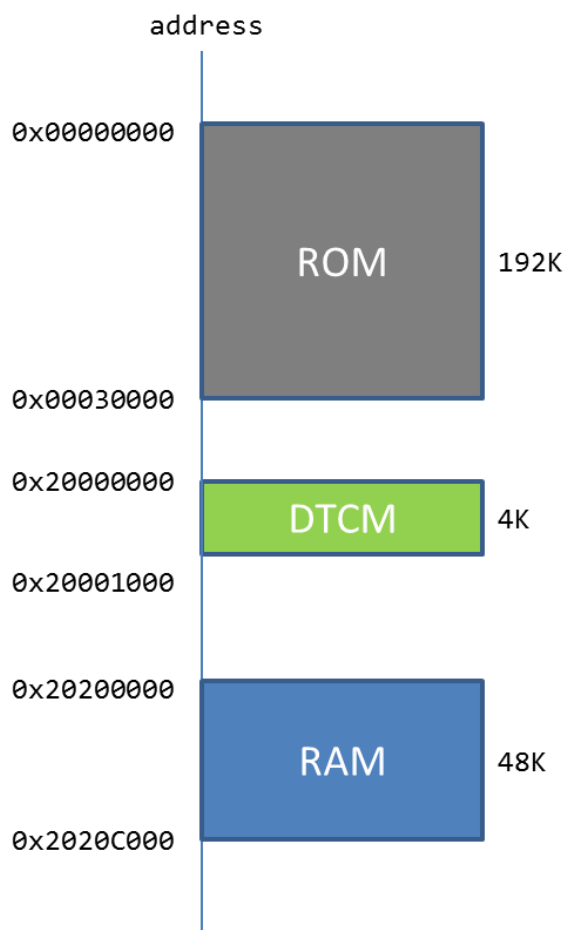


5.1.2 The size and configuration in AmebaPro(little CPU)

The size and configuration in AmebaPro(little CPU) is as shown below

	Size(bytes)	Description
ROM	192K	
DTCM	4K	can place Read/write data
RAM	48K	Is also known as SRAM

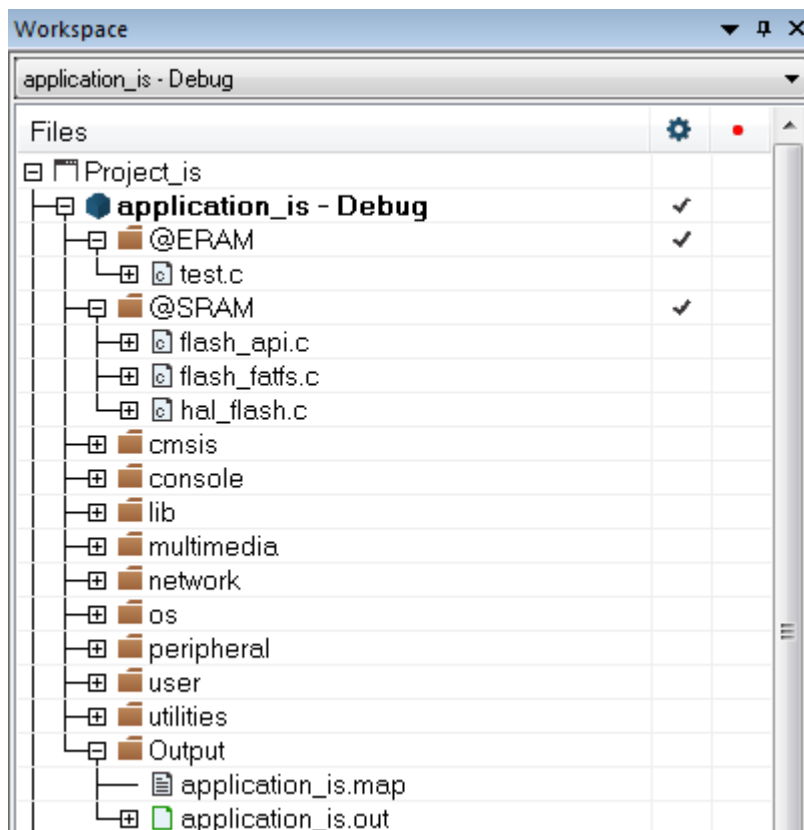
The graph of configuration in AmebaPro little CPU is as shown below:



5.2 Memory Configuration

5.2.1 Configure memory in IAR

In IAR Workspace, there are “@ERAM” and “@SRAM” group. “@ERAM” represents external RAM, which can be known as PSRAM or LPDDR. “@SRAM” represents SRAM. Except “@ERAM” and “@SRAM” group, the rest of text section will be placed in XIP.



If users want to place specific text section of source file into PSRAM/LPDDR, users can drag-and-drop the files into “@ERAM”. For example, text section of “test.c” will be placed in PSRAM/LPDDR as the graph above. If users want to place specific source file into SRAM, users can drag-and-drop the files into “@SRAM”. For example, “flash_api.c”, “flash_fatfs.c”, “hal_flash.c” is placed in SRAM as the graph above.

5.2.2 Configure memory in ICF file

IAR uses ICF (IAR Configuration File) to configure memory allocation so users can configure memory allocation by ICF file.

ICF file of Big CPU in AmebaPro locates:

“SDK/project/realtek_amebapro_v0_example/EWARM-RELEASE/ application_is.icf”

Open “application_is.icf” with editor. There are some memory regions in it, which is:

- ITCM_RAM_region
- DTCM_RAM_region
- RAM_region
- RAM_NC_region
- ERAM_region
- XIP_FLASH_region

Users can reference IAR document if users don't know the format of ICF.

5.2.3 Memory overflow

In default, AmebaPro place text section in XIP area. If XIP does not have enough space, it will show the errors as below when linking.

Error[Lp011]: section placement failed
unable to allocate space for sections/blocks with a total estimated minimum size of 0xa0051 bytes (max align 0x8) in <[0x9bf80140-0x9bffffff]> (total uncommitted space 0x7fec0).

The solution is to move the program in XIP to other region.

5.3 Video in DRAM

Video source example costs a lot of memory:

- Stream 1: H264 1080p, 15fps, bitrate 2M + 8K AAC
- Stream 2: 720p, 30fps, bitrate 1M + 8K AAC
- Snapshot mode: 720p, JPEG Level 5

EX: Video costs **30.58MB** in DRAM

- 1080p H264: Encoder: 6.54MB, ISP buffer: $2.97 * 3 = 8.91\text{MB}$; Encoder buffer pool: 2.97MB , **total 18.42MB**
- 720p H264: Encoder: 2.92MB, ISP buffer: $1.32 * 4 = 5.28\text{MB}$; Encoder buffer pool: 1.32MB , **total 9.52MB**
- 720p JPEG: Encoder: 3.21KB, ISP buffer: 1.32 MB; Snapshot buffer: 1.32MB , **total 2.64MB**

5.4 Unmovable program

5.4.1 XIP-related Flash API

Because XIP use flash interface so that flash api are placed in SRAM. Users cannot move these file to other place, including:

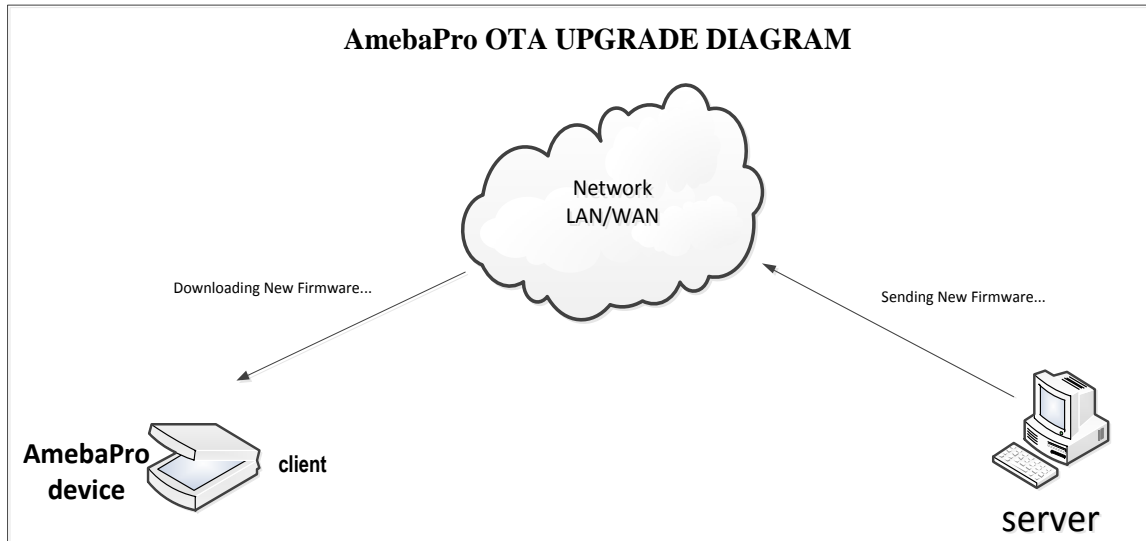
- flash_api.c
- flash_fatfs.c
- hal_flash.c.

5.5 Counting system of memory

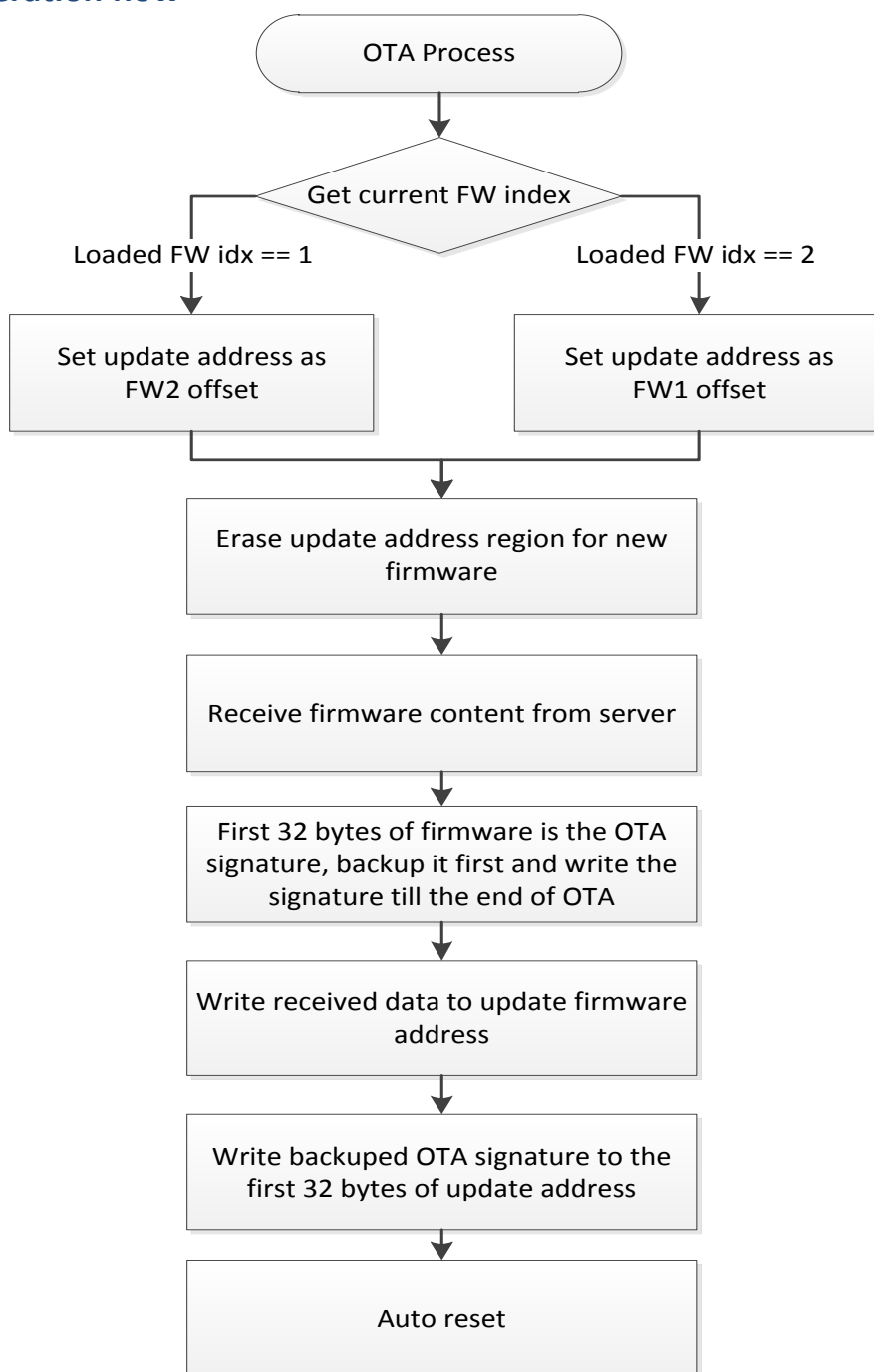
Please refer to UM0070.


6 **OTA**

Over-the-air programming (OTA) provides a methodology to update device firmware remotely via TCP/IP network connection.

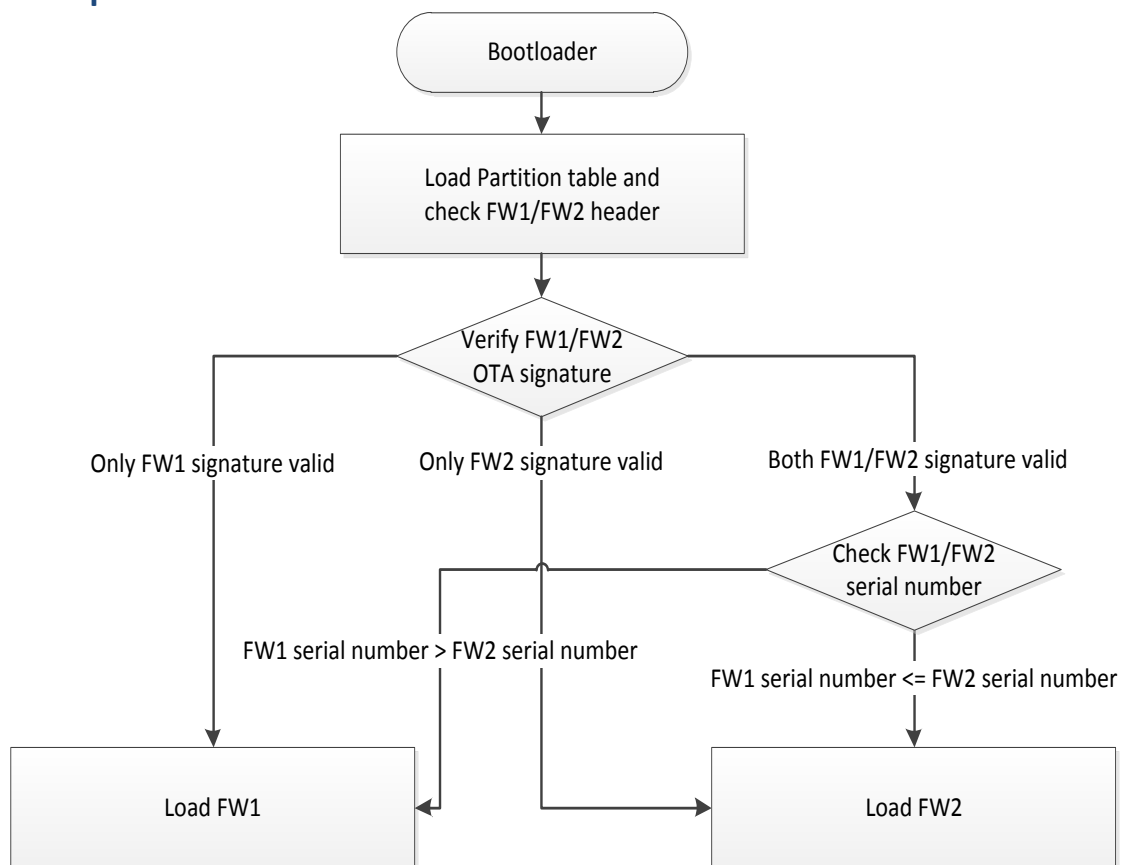


6.1 OTA operation flow



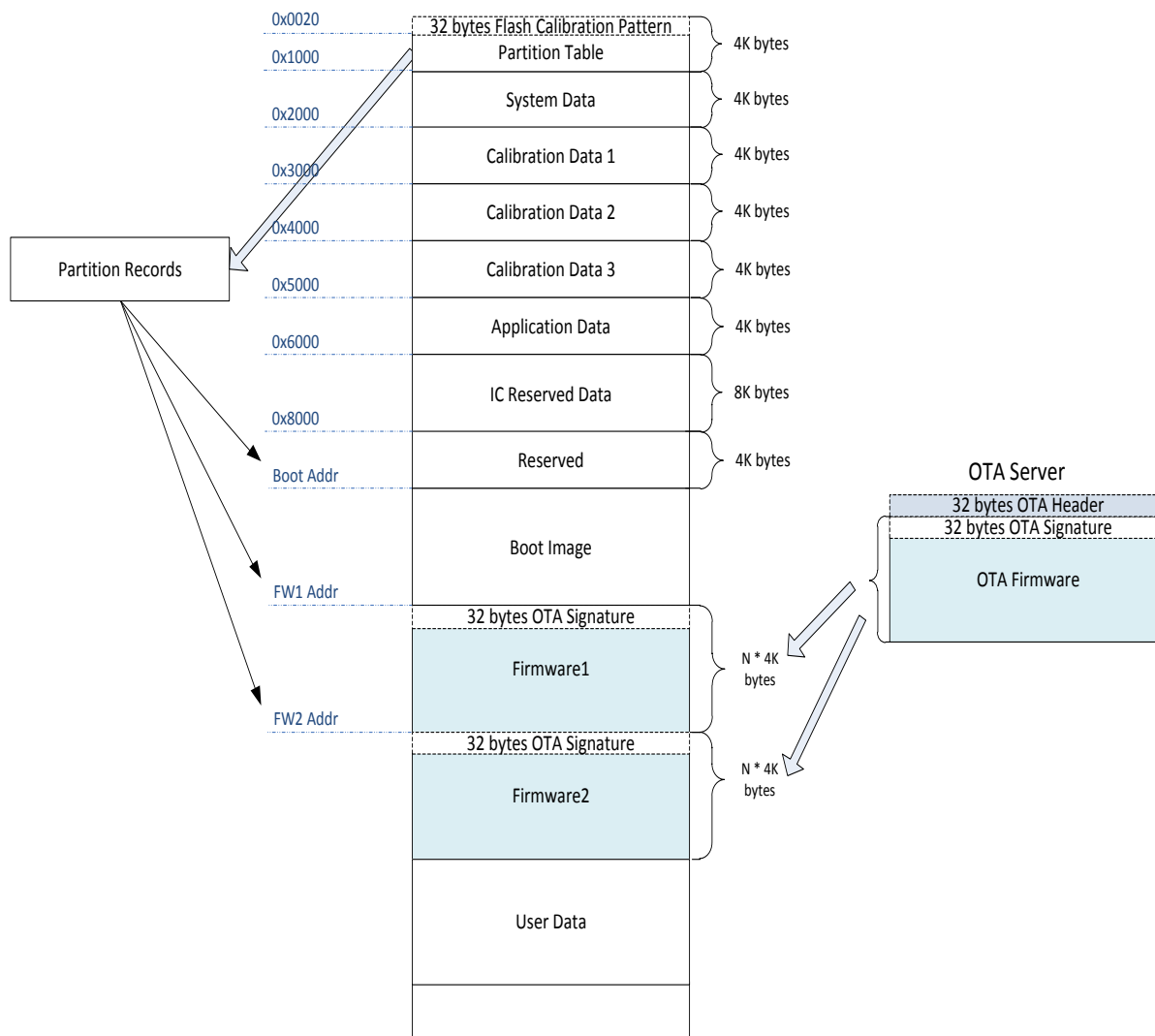
 During the step of “Write received data to update firmware address”, the 32 bytes OTA signature need set to 0xff, which is invalid signature. The correct OTA signature needs to be appended at the end of OTA process to prevent device booting from incomplete firmware.

6.2 Boot process flow



Boot loader will select latest (based on serial number) updated firmware and load it.

6.3 Upgraded partition



In AmebaPro OTA update procedure, Firmware1 and Firmware2 are swapped each other. The Firmware1/Firmware2 addresses are stored in partition records, defined in *partition.json* under *project\realtek_amebapro_v0_example\EWARM-RELEASE*. Please adjust it according to your firmware size. Also, please note that the Firmware1/Firmware2 address is required set to a 256KB aligned address, e.g. 0x40000 or 0x240000, due to the restriction of XIP remapping architecture.

```
"fw1":{
  "start_addr" : "0x40000",
  "length" : "0x200000",
  "type": "FW1",
  "dbg_skip": false,
  "hash_key": "000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E5F"
```

```

},
"fw2":{
    "start_addr" : "0x240000",
    "length" : "0x200000",
    "type": "FW2",
    "dbg_skip": false,
    "hash_key": "000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E5F"
}

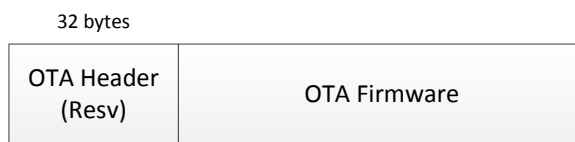
```

6.4 Firmware image output

After building project source files in SDK, it would generate ota_is.bin, which is the OTA Firmware as mentioned earlier. This ota_is.bin can be used for remote OTA server. When device executes the OTA procedure, it would determine target OTA firmware and only would program its corresponding content into the flash.

6.4.1 OTA file format

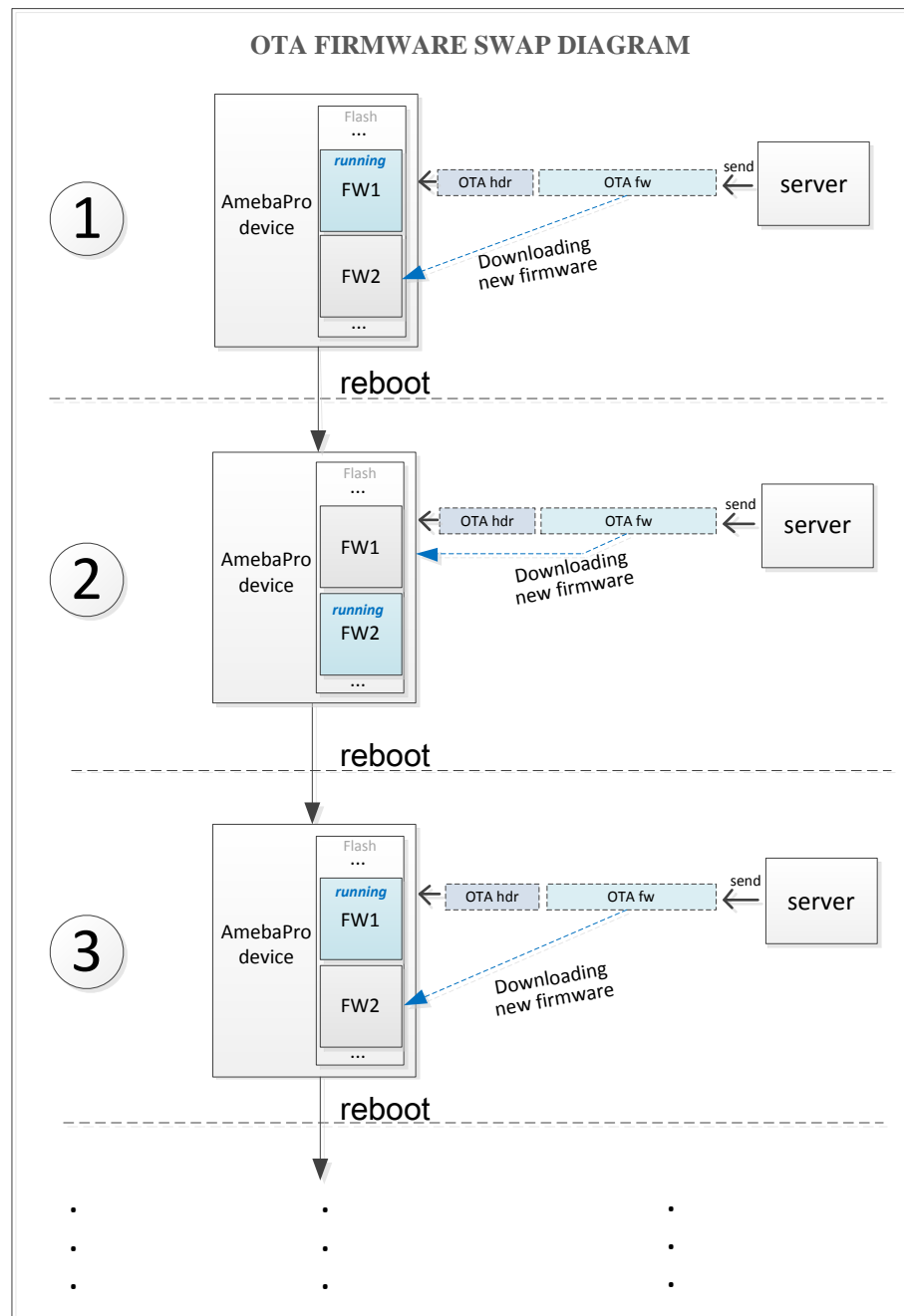
Below is the format of ota_is.bin used by OTA server:



ota_is.bin includes OTA firmware and a 32 bytes OTA Header. Currently, the OTA Header is reserved for future extension.

6.4.2 OTA firmware swap behavior

When device executes OTA procedure, it would update the other OTA block, rather than the current running OTA block. The OTA firmware swap behavior should be looked like as below figure if the updated firmware keeps using newer serial number value.



6.4.3 Configuration for building OTA firmware

Since the bootloader would check the serial number of OTA firmware to determine the boot sequence, the serial number of the OTA firmware need to be configured correctly before project build.

6.4.3.1 Serial number

AmebaPro OTA use serial number to decide the boot sequence if the signature of both firmware are valid. Hence before building the project, please make sure the serial number is correctly configured. To set the serial number of a firmware, please follow below steps:

Step 1: The serial number setting of a firmware is as same as the serial number of its first image. You can check the images sequence in *project\realtek_amebapro_v0_example\EWARM-RELEASE\amebapro_firmware_is.json*.

```
"FIRMWARE":{
  "images":[
    {"img": "XIP", "offset":"0x00"},
    {"img": "ISP", "offset":"0x00"},
    {"img": "CINIT", "offset":"0x00"},
    {"img": "FWLS", "offset":"0x00"},
    {"img": "FWHS", "offset":"0x00"},
    {"img": "WLAN", "offset":"0x00"},
    {"img": "WOWLAN", "offset":"0x00"}
  ]
}
```

For this example, the XIP is located at the top sequence. Hence it is the first image of this firmware.

Step 2: Modify the serial number setting of the first image. Take above figure for example, we need to modify the serial number of **"XIP"**:

```
"XIP": {
  "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,MO,CPFW",
    "type": "XIP",
    "enc": false,
    "__comment_pkey_idx": "assign by program, no need to configure",
    "serial": 100
  },
}
```

The Serial number is stored as 4 byte digital number and is valid from 0. Please modify it according to your firmware version.

Step 3: After building project source files in SDK, it should automatically generate *SDK_folder/project/project_name/EWARM-RELEASE/Debug/Exe/ota_is.bin*, which is the application only firmware as mentioned as OTA Firmware. The serial information would also be included in this firmware.

6.5 Implement OTA over Wi-Fi

6.5.1 OTA using local download server base on socket

The example shows how device updates image from a local download server. The local download server send image to device based on network socket.

Make sure both device and PC are connecting to the same local network.

6.5.1.1 Build OTA application image

Turn on OTA command

The flag defined in `\project\realtek_amebapro_v0_example\inc\platform_opts.h`.

```
#define CONFIG_OTA_UPDATE 1
```

Check current loaded firmware index and acquire the upgraded firmware address

```
/* ota_8195b.c */
uint32_t update_ota_prepare_addr(void){
    uint32_t NewFWAddr;
    fw_img_export_info_type_t *pfw_image_info;

    pfw_image_info = get_fw_img_info_tbl();

    printf("WnWr[%s] Get loaded_fw_idx %dWnWr", __FUNCTION__, pfw_image_info->loaded_fw_idx);
    if(pfw_image_info->loaded_fw_idx == 1)
        NewFWAddr = pfw_image_info->fw2_start_offset;
    else if(pfw_image_info->loaded_fw_idx == 2)
        NewFWAddr = pfw_image_info->fw1_start_offset;
    else {
        printf("WnWr[%s] Unexpected index %d", __FUNCTION__, pfw_image_info->loaded_fw_idx);
        return -1;
    }

    printf("WnWr[%s] NewFWAddr %08XWnWr", __FUNCTION__, NewFWAddr);
    return NewFWAddr;
}
```

6.5.1.2 Setup local download server

Step 1: Build new firmware ota_is.bin and place to tools\DownloadServer folder.

Step 2: Edit start.bat file: Port = 8082, file = ota_is.bin

```
@echo off
DownloadServer 8082 ota_is.bin
set /p DUMMY=Press Enter to Continue ...
```

Step 3: Execute start.bat.

```
c():checksum 0x84dd63b
Listening on port (8082) to send ota_is.bin (1372256 bytes)

waiting for client ...
```

6.5.1.3 Execute OTA procedure

After device connect to AP, enter command: ATWO=IP[PORT]

```
#ATWO=192.168.0.107[8082]
[ATWO]: _AT_WLAN_OTA_UPDATE_

[MEM] After do cmd, available heap 33446656

#
[update_ota_local_task] Update task start
[update_ota_prepare_addr] Get loaded_fw_idx 1
[update_ota_prepare_addr] NewFWAddr 00240000
[update_ota_local_task] Read info first
[update_ota_local_task] info 12 bytes
[update_ota_local_task] tx file size 0x14f060
[update_ota_erase_upg_region] NewFWLen 1372224
[update_ota_erase_upg_region] NewFWBlkSize 336 0x150
[update_ota_local_task] Start to read data 1372224 bytes
.
[update_ota_local_task] sig_backup for 32 bytes from index 0
.....
.....
.....
.....
Read data finished

[update_ota_signature] Append OTA signature
[update_ota_signature] signature:
 3C 9D D9 9F E9 30 C1 17 D5 97 E1 9B 35 46 09 77
 5B D4 FD B0 96 7A 81 31 1E 69 B3 F4 BD FE D2 A5
[update_ota_local_task] Update task exit
[update_ota_local_task] Ready to reboot
== Rt18195bh IoT Platform ==
```


Local download server success message:

[illegible]

After finishing downloading image, device will be auto-rebooted, and the bootloader will load new firmware if it exists.

6.5.2 OTA using local download server based on HTTP

This example shows how device updates image from a local http download server. The local http download server will send the http response which data part is *ota_is.bin* after receiving the http request.

Make sure both device and PC are connecting to the same local network.

6.5.2.1 Build OTA application image

Turn on OTA command

The flags defined in `\project\realtek_amebapro_v0_example\inc\platform_opts.h` and `\component\soc\realtek\8195b\misc\platform\ota_8195b.h`.

```
/* platform_opts.h */
#define CONFIG_OTA_UPDATE      1
#define CONFIG_EXAMPLE_OTA_HTTP 1
```

```
/* ota_8195b.h */
#define HTTP_OTA_UPDATE
```

Define Server IP and PORT in example_ota.c file

(In `\component\common\example\ota_http\example_ota_http.c`)

```
#define PORT      8082
#define IP        "192.168.0.107"
#define RESOURCE  "ota_is.bin"
```

Example:

SERVER: <http://m-apps.oss-cn-shenzhen.aliyuncs.com/051103061600.bin>

Setting: #define PORT 80
 #define HOST "m-apps.oss-cn-shenzhen.aliyuncs.com"
 #define RESOURCE "051103061600.bin"

Check current loaded firmware index and acquire the upgraded firmware address

```
/* ota_8195b.c */
uint32_t update_ota_prepare_addr(void){
    uint32_t NewFWAddr;
    fw_img_export_info_type_t *pfw_image_info;

    pfw_image_info = get_fw_img_info_tbl();

    printf("WnWr[%s] Get loaded_fw_idx %dWnWr", __FUNCTION__, pfw_image_info->loaded_fw_idx);
    if(pfw_image_info->loaded_fw_idx == 1)
        NewFWAddr = pfw_image_info->fw2_start_offset;
    else if(pfw_image_info->loaded_fw_idx == 2)
        NewFWAddr = pfw_image_info->fw1_start_offset;
    else {
        printf("WnWr[%s] Unexpected index %d", __FUNCTION__, pfw_image_info->loaded_fw_idx);
        return -1;
    }

    printf("WnWr[%s] NewFWAddr %08XWnWr", __FUNCTION__, NewFWAddr);
    return NewFWAddr;
}
```

Communication with Local HTTP download server

1. In *http_update_ota_task()*, after connecting with server, Ameba will send a HTTP request to server : "GET /RESOURCE HTTP/1.1\r\nHost: host\r\n\r\n".
2. The local HTTP download server will send the HTTP response after receiving the request. The response header contains the "Content-Length" which is the length of the *firmware_is.bin*. The response data part is just *firmware_is.bin*.
3. After Ameba receiving the HTTP response, it will parse the http response header to get the content length to judge if the receiving *firmware_is.bin* is completed.

6.5.2.2 Setup local HTTP download server

Step 1: Build new firmware ota_is.bin and place to tools\DownloadServer(HTTP) folder.

Step 2: Edit start.bat file: Port = 8082, file = ota_is.bin

```
@echo off
DownloadServer 8082 ota_is.bin
set /p DUMMY=Press Enter to Continue ...
```

Step 3: Execute start.bat.

```
<Local HTTP Download Server>
Listening on port (8082) to send ota_is.bin (1365984 bytes)

waiting for client ...
```

6.5.2.3 Execute OTA procedure

Reboot the device and connect to AP, it should start the OTA update through HTTP protocol after 1 minute.

```
#
[update_ota_prepare_addr] Get loaded_fw_idx 1
[update_ota_prepare_addr] NewFWAddr 00240000

[http_update_ota] Download new firmware begin, total size : 1365984

[update_ota_erase_upg_region] NewFWLen 1365952
[update_ota_erase_upg_region] NewFWBlkSize 334 0x14e.
[http_update_ota] sig_backup for 32 bytes from 0 index
.....
[http_update_ota] Download new firmware 1365952 bytes completed

[update_ota_signature] Append OTA signature
[update_ota_signature] signature:
 47 8C 7E E0 31 01 F1 FA 5E 81 58 88 B9 70 20 65
 D5 91 B1 CA 3A 2E F1 9B D5 BD 13 54 09 F1 11 5A
[http_update_ota_task] Update task exit
[http_update_ota_task] Ready to reboot
== Rt18195bh IoT Platform ==
```

Local download server success message:

[illegible]

After finishing downloading image, device will be auto-rebooted, and the bootloader will load new firmware if it exists.

6.6 OTA signature

To Clear or Recover OTA signature for verification via UART at command, please refer to AN0025.

7 Power Management

7.1 AmebaPro Power Save Modes

AmebaPro provide several power modes. Table 1.1 describes DeepSleep, Standby, and SleepPG:

Mode	DeepSleep	Standby	SleepPG
Initiator/domain	LS	LS	HS
WLAN wakeup	No	Yes	Yes
STimer wakeup	Yes	Yes	No
GTimer wakeup	No	Yes	Yes
HSTimer wakeup	No	Yes	No
GPIO wakeup	Yes	Yes	Yes
ADP wakeup	Yes	No	No
RTC wakeup	Yes	No	No
PWM wakeup	No	Yes	Yes
UART wakeup	No	Yes	Yes
MII wakeup	No	No	No
I2C wakeup	No	Yes	Yes
ADC wakeup	No	Yes	Yes
COMP wakeup	No	Yes	No
USB wakeup	No	No	Yes
SDIO wakeup	No	No	Yes
SGPIO wakeup	No	Yes	Yes

Table 7.1 AmebaPro Power mode and wakeup source

There are some features needs to be noticed:

- DeepSleep saves the most power among all power modes
- In most cases, Standby saves slightly more power than SleepPG. But Standby and SleepPG are compatible power modes.
- After resume, DeepSleep, Standby, and SleepPG are running code from initial, not from the position where program suspend.

7.1.1 DeepSleep

DeepSleep save the most power among all power modes. HS is turned off. LS only keep resources that are required by the wakeup source.

DeepSleep needs to be invoked from LS. If user wants to invoke DeepSleep from HS, he needs to design his own ICC command/message and implement related code.

7.1.2 Standby

In Standby mode, HS is turned off. LS only keep resources that are required by the wakeup source.

In most of wakeup source except WLAN wakeup, the power consumption is stable and can be measured from current meter with low sample rate.

If user choose wakeup from WLAN, user need configures WLAN part before call Standby. Please reference [“Wakeup from WLAN”](#) for more information.

Standby needs to be invoked from LS. If user wants to invoke Standby from LS, he needs to design his own ICC command/message and implement related code.

7.1.3 SleepPG

In SleepPG mode, HS and LS are keep resources that are required by the wake source. Compare to Standby mode, SleepPG reserve some HS function for wakeup source. The power consumption between Standby and SleepPG are compatible. User can choose which one suites his needs especially that both Standby and SleepPG support wakeup from WLAN.

If user choose wakeup from WLAN, user need configures WLAN part before call SleepPG. Please reference [“Wakeup from WLAN”](#) for more information.

SleepPG needs to be invoked from HS.

7.1.4 Wakeup from WLAN

If user choose wakeup from WLAN in either Standby mode or SleepPG mode, he needs to configure wlan before invoke Standby or SleepPG.

After user configure wlan and invoke Standby or SleepPG, system would enter suspend state. In this state, LS MCU would wakeup/suspend periodically to check wlan state and decide if it needs to wakeup HS MCU. After wlan receive the wakeup packet that matches the wakeup pattern, then LS MCU would wakeup HS and

The whole progress is like below diagram:

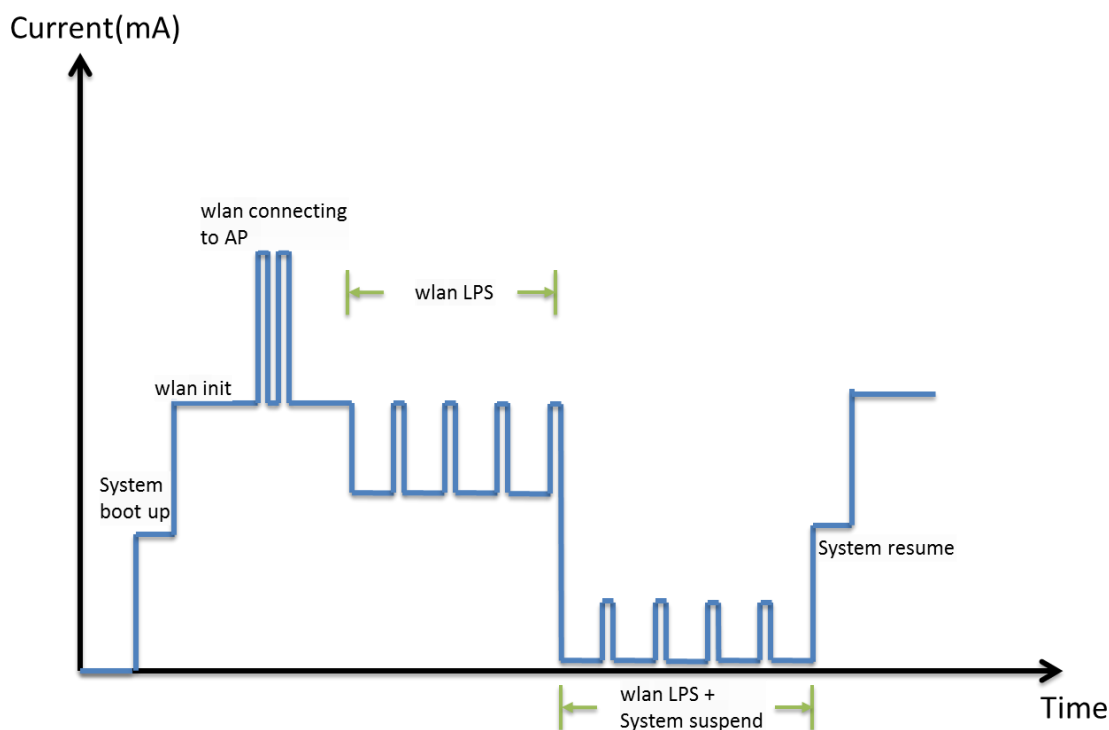


Diagram 1.1 power consumption of wake on wlan

1. At first system boot up, initialize wlan, connecting to AP.
2. After connected to AP, wlan would enter LPS state if there is no heavy data traffic. In LPS state, wlan would listen beacon for every 100ms (if DTIM is 1). So you could see power consumption rise for every 100ms. Power consumption would drop after wlan receive the beacon and the TIM field has no packet for this device, then wlan would turn off RF and try to keep in low power state.

3. If user try to make system save more power with wlan associate idle, he could invoke Standby or SleepPG. You could see the power consumption drops more in wlan LSP with system suspend.

Please note that if you want to measure the power consumption when system suspend with wlan LPS, you have to make sure the voltage regulator of power supply or current meter could handle the voltage drop and rise between hundreds of micro amp and dozens of milliamp.

7.2 Examples

7.2.1 Example: power_save

7.2.1.1 Abstract

This example add a AT command “PS” that user can test power saving related function included “Deepsleep”, “Standby”, “SleepPG”, “wake on wlan”, “tcp keep alive” and wake up reason.

7.2.1.2 Setup example

The example is located in:

“\project\realtek_amebapro_v0_example\example_sources\power_save\”

Copy HS & LS folder to src folder, compile LS & HS project, and the download the binary.

7.2.1.3 Deepsleep

Type below command in HS console:

PS=deepsleep

It would show below logs in LS console:

```
deepsleep wake event:0x0001
```

You can see current changes if you have connected current meter. System would resume after 60s. HS & LS would run from the start just like reboot. In the log in LS, “wake event” is the wake up source. You can reference “power_mode_api.h” for correspond wake up source.

Type below command in HS console:

PS=deepsleep,stimer=10

The would set stimer wake up in deepsleep for 10s.

Type below command in LS console:

PS=deepslee,gpio

This would set GPIOA_13 wake up in deepsleep.

7.2.1.4 Standby

Type below command in HS:

PS=standby

It would show below logs in LS console:

```
Standby wake event:0x0001
```

You can see current changes if you have connected current meter. System would resume after 60s. HS & LS would run from the start just like reboot. In the log in LS, “wake event” is the wake up source. You can reference “power_mode_api.h” for correspond wake up source.

After system resume, it would show wake reason in LS log:

```
wake from AON_TIMER
```

Type below command in HS:

PS=standby,stimer=10

This would set stimer wake up in standby for 10s.

Type below command in HS:

PS=standby,gpio=2

This would set GPIOA_2 wake up in standby.

7.2.1.5 LS wake reason

As system resume from standby, HS may need to know the wake up source in LS. You can type below command in HS console:

PS=ls_wake_reason

It would show below log in HS log:

```
wake from AON_TIMER
```

7.2.1.6 SleepPG

Type below command in HS console:

PS=sleeppg

You can see current changes if you have connected current meter. System would resume after 60s. HS & LS would run from the start just like reboot.

7.2.1.7 Wake on wlan

Type below command in HS console:

```
PS=wowlan,your_ssid,your_pw
```

System would try to connect specified AP with your_ssid & your_pw. It would enter wake on wlan mode after connected for 16s. The wake on wlan mode by default use standby with WLAN wake up.

In wake on wlan mode, LS would keep restart correspond the AP beacon interval.

Wlan would wake system after it receive wake up packet. (PING, by default)

7.2.1.8 TCP keep alive

Type below command in HS console to setup TCP keep alive:

```
PS=tcp_keep_alive,192.168.1.100,5566
```

The last 2 parameter is the IP & port of TCP server. The setting would become effective after system enter wake on wlan mode. So you still need type

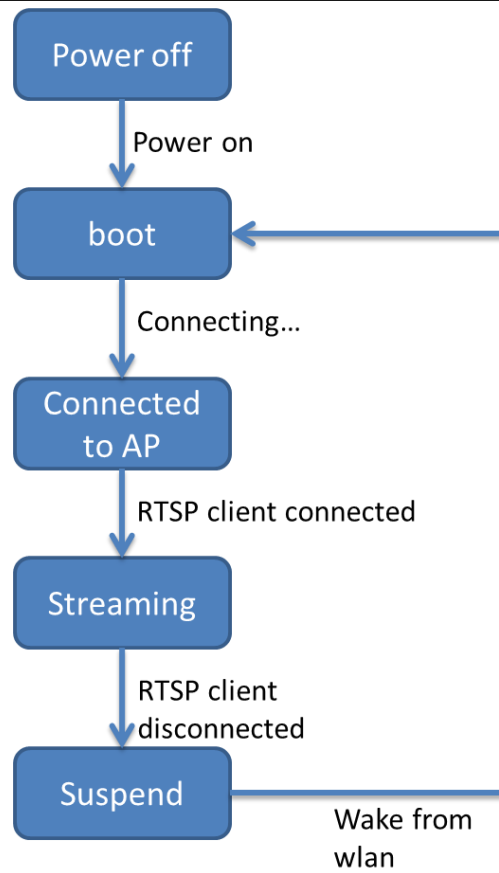
“PS=wowlan,your_ssid,your_pw” to make it works.

7.2.2 Example: system suspend with wake on wlan after streaming stop

7.2.2.1 Abstract

This example demonstrate (1) how to get first video frame after boot/resume, and (2) how to make system suspend and able to be waked from wlan.

The life cycle is shown as below diagram:



- At first, system is boot from power on or wake from wlan(describe later)
- Then AmebaPro try connecting to AP either by fast connect feature or AT command.
- After AmebaPro connected to AP, AmebaPro would start RTSP server and wait for RTSP client.
- User may use VLC or other RTSP client connects to AmebaPro. User should see streaming content **which is buffered after boot up**. So user can evaluate the time from boot up to get the first frame.
- After user stop VLC client, AmebaPro would enter system suspend with wlan association idle.
- User can wake Ameba by sending wlan unicast to AmebaPro. (Ex. Ping can do this job)

7.2.2.2 Setup example

You need modify file **"platform_opts.h"** (placed in "project\realtek_amebapro_v0_example\inc\platform_opts.h")
And enable these compile flags:

You can reference example “**media_framework**” (placed in “component\common\example\media_framework”)

```
#define CONFIG_EXAMPLE_MEDIA_FRAMEWORK 1
#if CONFIG_EXAMPLE_MEDIA_FRAMEWORK
#define FATFS_DISK_SD 1
#define ISP_BOOT_MODE_ENABLE 1
#if ISP_BOOT_MODE_ENABLE
#define ISP_BOOT_MP4 0
#define ISP_BOOT_RTSP 1
#if (ISP_BOOT_MP4 == 1 && ISP_BOOT_RTSP == 1) || (ISP_BOOT_MP4 == 0 && ISP_BOOT_RTSP == 0)
#error "It only can select the mp4 or rtsp"
#endif
#endif
#endif
#endif
```

This would enable media framework example. You can reference “example_media_framework.c” (placed in “\component\common\example\media_framework\example_media_framework.c”).

After compile and flash image, boot up AmebaPro and should see logs as below:

```
WIFI initialized

init_thread(53), Available heap 0x9b0f60
use ATW0, ATW1, ATWC to make wifi connection
wait for wifi connection...
```

Then you can use AT command to connect to AP.

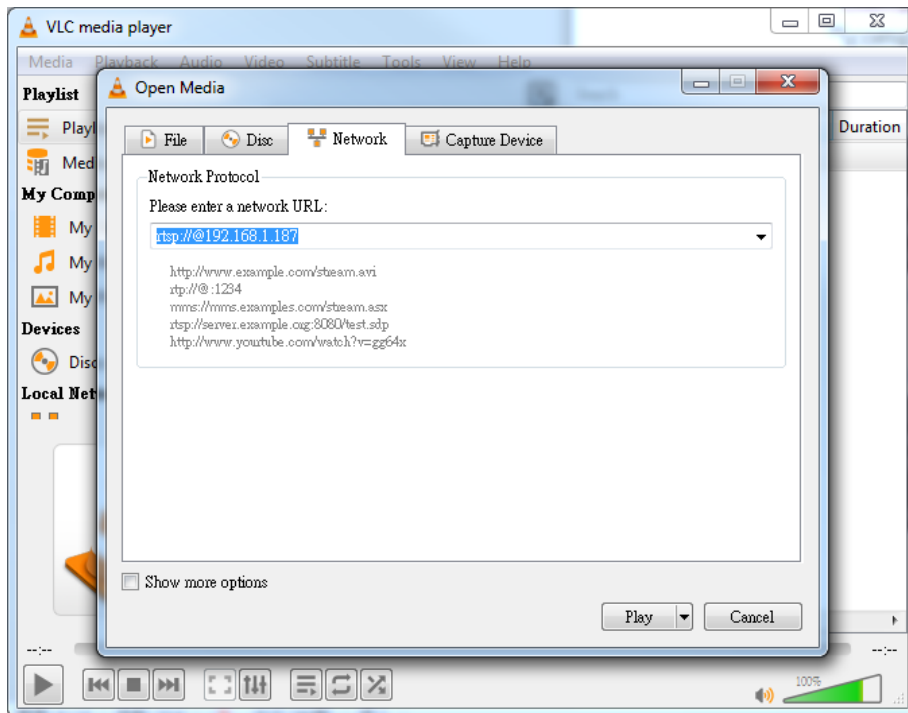
After connected to AP, AmebaPro would start RTSP server.

```
RTSP[0] port: 554
time_sync_enable
connect successful sta mode

rtsp stream enabled
```

You can use ping command to (1) test AmebaPro’s connection, and (2) update PC or Smart Phone’s ARP table.

Then open VLC as RTSP client and connected to AmebaPro:



You should see streaming after RTSP client connected.

After you stop streaming, system would enter power save state.

By default AmebaPro is wake by wlan unicast. So you can ping AmebaPro again to resume it.

After resume, AmebaPro is just like boot from power on, so you can exam the example again.

8 System

8.1 ICC (Inter CPUs Communication)

There are 2 CPUs, HP (High-Power) CPU and LP (Low-Power) CPU, on the RTL8195B platform. The ICC (Inter CPUs Communication) hardware is designed for the purpose of making 2 CPUs can communicate each other.

8.1.1 Example: ICC

8.1.1.1 Abstract

This example shows:

1. How to send ICC commands from HS/LS platform to the LS/HS platform.
2. A ICC message loop back. Icc message data is sent from HS to LS and then LS loop back ICC message to the HS.

8.1.1.2 Setup example

The example is located in:

“\project\realtek_amebapro_v0_example\example_sources\icc\”

You should copy both HS and LS folder to project folder. Compile both LS and HS code and download the program.

8.2 System reset

8.2.1 ls_sys_reset

```
/* Reset LS platform via ICC command*/  
void ls_sys_reset( void )
```

8.2.2 sys_reset

```
/* Reset some register status and do HS watchdog reset */  
void sys_reset( void )
```

9 **Hardware Peripherals**

9.1 **GTimer**

- 16 Gtimer supported at HS domain and 8 Gtimer supported at LP domain.
- Source clock is selected from 20MHz and moderate accuracy 32KHz.

9.1.1 **gtimer_rtc**

This example shows how to use general timer API to implement a software RTC.

gtimer_rtc example is located in

project\realtek_amebapro_v0_example\example_sources\gtimer_rtc

9.1.2 **SNTP**

This example shows system time maintained by time from NTP server.

sntp_showtime example is located in component\common\example\sntp_showtime

9.1.2.1 *Software Setup*

configure the SNTP example settings in *(project)\inc\platform_opts.h*:

```
/* For sntp show time example */  
#define CONFIG_EXAMPLE_SNTP_SHOWTIME      1
```

9.2 **RTC**

RTC is invoked from LS MCU. This example shows how to init and read/write RTC.

RTC example is located in project\realtek_amebapro_v0_example\example_sources\rtc

9.3 **PWM**

- Support maximum 8 PWM functions at HS domain and LP domain
- 0~100% duty can be configurable
- Use selected Gtimer interrupt as counter source

9.4 Efuse

- LS efuse
 - Logical area : 0x0 ~ 0x1CF , size: 464 bytes
 - Reserved area : 0x1D0 ~ 0x1FF , size: 48 bytesTotal size : 512 bytes

- HS efuse
 - Logical area : 0x0 ~ 0x10F , size: 272 bytes
 - OTP : 0x110~0x12F = 32 bytes
 - security area 0x130~0x1AF = 128 bytes
 - ◆ security key: 0x130~0x14F
 - ◆ writable security zone : 0x150~0x16F
 - ◆ super security key: 0x170~0x18F
 - ◆ reserved: 0x190~0x1AF
 - wlan hidden area : 0x1B0 ~ 0x1FF = 80 bytesTotal size : 512 bytes

9.4.1 User OTP

User OTP : 0x110~0x12F = 32 bytes

efuse_api

```
/**
 * @brief Read efuse OTP content
 * @param address: Specifies the offset of the OTP.
 * @param len: Specifies the length of readback data.
 * @param buf: Specified the address to save the readback data.
 * @retval 0: Success
 * @retval -1: Failure
 */
int efuse_otp_read(u8 address, u8 len, u8 *buf);

/**
 * @brief Write user's content to OTP efuse
 * @param address: Specifies the offset of the programmed OTP.
 * @param len: Specifies the data length of programmed data.
 * @param buf: Specified the data to be programmed.
 * @retval 0: Success
 * @retval -1: Failure
 */
```

```
int efuse_otp_write(u8 address, u8 len, u8 *buf);
```

10 File system

AmebaPro support file system based on flash and SD card and AmebaPro can support these two storages simultaneously.

10.1 **FAT Filesystem on Flash**

Flash file system example is located in *component\common\example\fatfs*

10.1.1 **Software Setup**

First, configure the FATFS example settings in *(project)\inc\platform_opts.h*:

```
/* For FATFS example*/
#define CONFIG_EXAMPLE_FATFS          1
#if CONFIG_EXAMPLE_FATFS
#define CONFIG_FATFS_EN              1
#if CONFIG_FATFS_EN
// fatfs version
#define FATFS_R_10C
// fatfs disk interface
#define FATFS_DISK_USB              0 // Not supported on AmebaPro
#define FATFS_DISK_SD               0
#define FATFS_DISK_FLASH           1
#endif
#endif
```

Next, modify parameters in

component\common\file_system\fatfs\r0.10c\include\ffconf.h:

```
...
#define _USE_MKFS              1//0   /* 0:Disable or 1:Enable */
...
...
#define _MAX_SS                4096//512
...
```

Please note that the flash memory base for the flash filesystem used in the Flash FATFS example is defined in the file

component\common\file_system\fatfs\disk_if\src\flash_fatfs.c:

```
...
#define FLASH_APP_BASE 0x180000
...
```

Finally, rebuild the project and download active application to DEV board.

10.2 Dual FAT Filesystem - File system on both SD Card and Flash

Dual file system example is located in *component\common\example\fatfs*

10.2.1 Hardware Setup

Please connect your Ameba DEV board with SD/MMC card connector as described in UM0073.

10.2.2 Software Setup

First, configure the FATFS example settings in *(project)\inc\platform_opts.h*:

```
/* For FATFS example*/
#define CONFIG_EXAMPLE_FATFS          1
#if CONFIG_EXAMPLE_FATFS
#define CONFIG_FATFS_EN              1
#if CONFIG_FATFS_EN
// fatfs version
#define FATFS_R_10C
// fatfs disk interface
#define FATFS_DISK_USB              0 // Not supported on AmebaPro
#define FATFS_DISK_SD              1
#define FATFS_DISK_FLASH           1
#endif
#endif
```

Next, modify parameters in *ffconf.h*:

```
...
#define _USE_MKFS              1
...
#define _VOLUMES              2
...
#define _MAX_SS                4096
...
```

Please note that the flash memory base for the flash filesystem used in the Flash FATFS example is defined in the file

component\common\file_system\fatfs\disk_if\src\flash_fatfs.c:

```
...
#define FLASH_APP_BASE 0x180000
...
```

Finally, rebuild the project and download active application to DEV board.

THIS SOFTWARE AND DOCUMENT ARE PROVIDED “AS IS” WITHOUT ANY WARRANTIES OF ANY KIND. REALTEK MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THIS THE SOFTWARE AND DOCUMENT AT ANY TIME AND AT ITS SOLE DISCRETION. WITH RESPECT TO THE SOFTWARE; DOCUMENT; INFORMATION; MATERIALS; SERVICES; AND ANY IMPROVEMENTS AND/OR CHANGES THERETO PROVIDED BY REALTEK, REALTEK DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.