AN0700
AmebaPro2 Application Note

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211. Fax: +886-3-577-6047

www.realtek.com

DISCLAIMER
Please Read Carefully:

Realtek Semiconductor Corp., (Realtek) reserves the right to make corrections, enhancements, improvements and other changes to its products and services. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

Reproduction of significant portions in Realtek data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Realtek is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions.

Buyers and others who are developing systems that incorporate Realtek products (collectively, "Customers") understand and agree that Customers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Customers have full and exclusive responsibility to assure the safety of Customers' applications and compliance of their applications (and of all Realtek products used in or for Customers' applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Customer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Customer agrees that prior to using or distributing any applications that include Realtek products, Customer will thoroughly test such applications and the functionality of such Realtek products as used in such applications.

Realtek's provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation kits, (collectively, "Resources") are intended to assist designers who are developing applications that incorporate Realtek products; by downloading, accessing or using Realtek's Resources in any way, Customer (individually or, if Customer is acting on behalf of a company, Customer's company) agrees to use any particular Realtek Resources solely for this purpose and subject to the terms of this Notice.

Realtek's provision of Realtek Resources does not expand or otherwise alter Realtek's applicable published warranties or warranty disclaimers for Realtek's products, and no additional obligations or liabilities arise from Realtek providing such Realtek Resources. Realtek reserves the right to make corrections, enhancements, improvements and other changes to its Realtek Resources. Realtek has not conducted any testing other than that specifically described in the published documentation for a particular Realtek Resource.

Customer is authorized to use, copy and modify any individual Realtek Resource only in connection with the development of applications that include the Realtek product(s) identified in such Realtek Resource. No other license, express or implied, by estoppel or otherwise to any other Realtek intellectual property right, and no license to any technology or intellectual property right of Realtek or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Realtek products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of Realtek Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from Realtek under the patents or other Realtek's intellectual property.

Realtek's Resources are provided "as is" and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding resources or use thereof, including but not limited to accuracy or completeness, title, any epidemic failure warranty and any implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third-party intellectual property rights.
Realtek shall not be liable for and shall not defend or indemnify Customer against any claim, including but not limited to any infringement claim that related to or is based on any combination of products even if described in Realtek Resources or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek's Resources or use thereof, and regardless of whether Realtek has been advised of the possibility of such damages. Realtek is not responsible for any failure to meet such industry standard requirements.

Where Realtek specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Customers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any Realtek products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death. Such equipment includes, without limitation, all medical devices identified by the U.S.FDA as Class III devices and equivalent classifications outside the U.S. Customers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Customers' own risk. Customers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Customer will fully indemnify Realtek and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

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

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.

Contents

Convention

AmebaPro2 is a high-integrated IC. Its features include 802.11 Wi-Fi, H.264/H.265 video codec, Audio Codec.

This manual introduce users how to develop AmebaPro2, including SDK compiling and downloading image to AmebaPro2.

1 Building Environment

1.1 Setting up GCC Building Environment

1.1.1 Building the project in GCC Building Environment (Windows)

1.1.1.1 Installing mingw with ASDK and setting up the CMake

(1) Download and extract msys64_v10_3.7z from tools folder
(2) Check the windows Environment Variable HOME by Command Prompt

```
echo %HOME%
```

NOTE

If %HOME% is not exist, the command will simply print %HOME%

(3) If your windows already have Environment Variable named HOME, open the file "msys64/etc/post-install/05-home-dir.post" Add "HOME=<PATH_TO_YOUR_MSYS64>/home/<USER_FOLDER>".

```
# If the home directory doesn't exist, create it.
HOME=C:/msys64_v10_3/msys64/home/${USER}
if [ ! -d "${HOME}" ]; then
  if mkdir -p "${HOME}"; then
    echo "Copying skeleton files."
```

NOTE

By default <USER_FOLDER> is ${USER}. To prevent some errors, do not include space characters in <USER_FOLDER>

(4) Double click "msys2_shell.cmd" from msys64 folder
(5) After setting up mingw, you need to install cmake. Download cmake in https://github.com/Kitware/CMake/releases/download/v3.20.0-rc1/cmake-3.20.0-rc1-windows-x86_64.msi and install it
(6) Add location of cmake.exe to PATH of msys2_shell by using vim ~/.bashrc and appending path of cmake.exe to environment variable PATH or using editor to directly append the path to file "msys64/home/<USER_FOLDER>/.bashrc"

```
export PATH=/c/Program\ Files/CMake/bin:$PATH
```

CAUTION

If your PATH contains space characters, remember to use "\" to escape

NOTE

For the first time adding the CMake PATH, after adding the PATH, you need to re-open the msys2_shell and check by:

```
$ cmake --version
cmake version 3.20.0-rc1

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

1.1.1.2 Adding toolchain to msys2

(1) Like adding PATH for cmake, user can add or change the toolchain in "msys64/home/<USER_FOLDER>/.bashrc".
(2) Add toolchain PATH by "export PATH=<path to toolchain>:$PATH".

```
if [ -d "../../asdk-10.3.0" ]; then
  echo "asdk-10.3.0 exist"
  export PATH=/asdk-10.3.0/mingw32/newlib/bin:$PATH
```

NOTE

The recommended toolchain version is 10.3.0

1.1.1.3 Building the project

(1) Open mingw by double clicking "msys2_shell.cmd".
(2) Enter the project location: project/realtek_amebapro2_v0_example/GCC-RELEASE.
(3) Create folder "build" and enter "build" folder.
(4) Run "cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake" to create the makefile.
(5) Run "cmake --build . --target flash" to build and generate flash binary.

NOTE

If building successfully, you can see flash_ntz.bin in the build folder

1.1.2 Building the project in GCC Building Environment (LINUX)

1.1.2.1 Add toolchain to the linux PATH

(1) Extract the toolchain file (the toolchain file may provide in tools folder):

```
tar -jxvf <PATH_TO_YOUR_TOOLCHAIN.tar.bz2> -C <DIR_TO_EXTRACT>
```

(2) Add toolchain to PATH:

```
export PATH=<PATH_TO_YOUR_TOOLCHAIN>/asdk-10.3.0/linux/newlib/bin:$PATH
```

NOTE

You can add PATH to ~/.bash_profile

1.1.2.2 Installing cmake for linux

(1) Install cmake using terminal (like "sudo apt-get -y install cmake"), if the installation is successful, you can get the version by "cmake --version".

1.1.2.3 Building the project

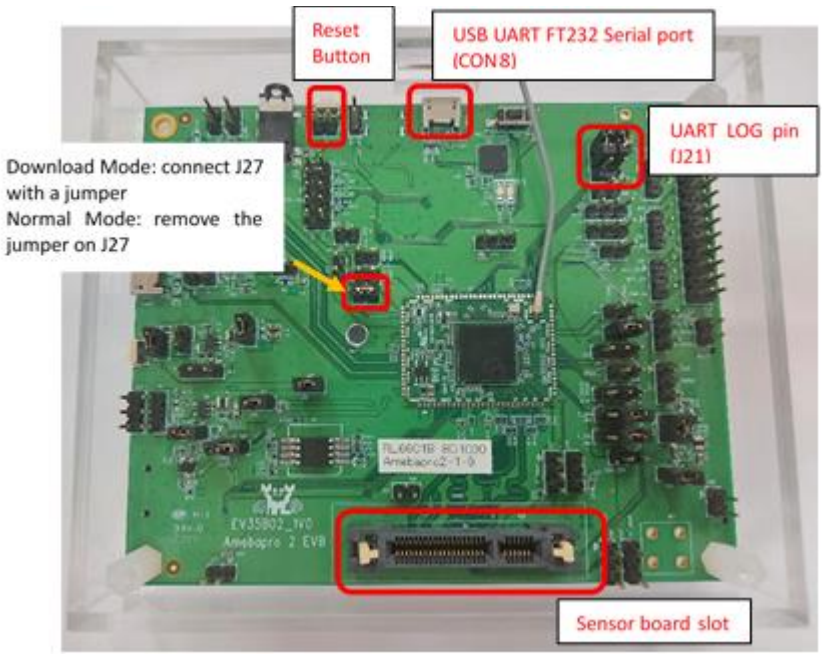
(1) Open linux terminal and enter the project location: project/realtek_amebapro2_v0_example/GCC-RELEASE/.
(2) Create folder "build" and enter "build" folder.
(3) Run "cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake" to create the makefile.
(4) Run "cmake --build . --target flash" to build and generate flash binary.

NOTE

- *If building successfully, you can see flash_ntz.bin in the build folder*
- *If the 'build' folder has been used by others, you can remove 'build' folder first to have clean build*
- *If there's some permission issues, you can do "chmod -R 777 <PATH_TO_YOUR_SDK>"*

1.2 Log UART Settings

(1) To use AmebaPro2 log UART, the user needs to connect jumpers to **J21** for **FT232 (CON8)**.
(2) After using CON8 to connect to PC, you can use console tools (like tera term, MoBaxterm) to get log from EVB by setting baud rate as **115200**.



2 SDK Architecture

In AmebaPro2 sdk, it mainly contains four folders. The folder “component” store the main component source and the folder “project” contains the project makefile, compile flag and some examples. The folder “doc” and “tool” provide the document and tools for assisting you to set up the project.

2.1 Component

Folder	Sub-folder	Description
component	at-cmd	AT-command
	audio	ASP algorithm api audio codec
	bluetooth	bluetooth driver
	example	amazon related examples audio related examples fatfs example mmf examples socket related examples …
	file_system	Fatfs and Littlefs
	lwip	lwip API source code
	mbed	mbed API source code
	media	multi-media framework modules muxer and demuer rtp codec for media
	network	cJSON coap dhcp httpc and httpd iperf mDNS mqtt ping rtsp sntp tftp websockect
	os	freertos: freertos source code os_dep: Realtek encapsulating interface for FreeRTOS, ram usage…
	soc	app: monitor and shell cmsis: cmsis style header file and startup file fwlib: hal drivers and nn api mbed-drivers: mbed API source code misc: driver and utilities
	ssl	ssl stub function and ram map source code
	stdlib	stdlib header files
	usb	usb and uvc header files
	video	ISP and video related api
	wifi	wifi api and wifi config related source code and header files

2.2 Project

Folder	Sub-folder	Description
Project/*	example_sources	examples for peripherals
	GCC-RELEASE	GCC cmake projects
	GCC-RELEASE/application	libraries for (non-trust zone) GCC project
	GCC-RELEASE/bootloader	bootloader project
	GCC-RELEASE/build	pre-build image files (boot.bin) and json files place for building cmake projects and generate flash image file (flash_ntz.bin)
	GCC-RELEASE/mp	for mp
	GCC-RELEASE/ROM	ROM code libraries
	inc	the header files for setting the project compile flag
	src	the main file source code for the project

2.3 Doc and tools

Folder	Sub-folder	Description
tools		PGTool: for downloading image files to AmebaPro2 msys64: for building the environment of AmebaPro2 project
doc		document for AmebaPro2

2.4 Binary files

File name	Location in flash layout (*1)	OTA after MP	Purpose	Comment
certable.bin	Key Certificate Table	N	Used to set the size and position of the certificate.bin	
partition.bin	Partition Table	N	Used to set the size and position of each bin	
certificate.bin	Key Certificate 1	N	Certificate bin file is used by secure boot. There is only one copy	certificate_ota.bin (for ota in MP)
boot.bin	Boot Image Primary	N	bootloader	boot_ota.bin (for ota in MP)
firmware_isp_iq.bin	ISP_IQ Data	N	Consists of fcs_data.bin & sensor.bin & iq.bin fcs_data.bin : load fcs data in rom code and initialize the sensor sensor.bin : initialize the sensor in normal boot iq.bin : IQ parameter adjustment	isp_iq_ota.bin (for ota in MP)

firmware.bin	Firmware 1 or 2	Y	Firmware image	ota.bin (for ota)
nn_model.bin	NN Model Data	Y	NN Model	nn_model_ota.bin (for ota)
boot_fcs.bin	FCS Data	Y	fcs video parameters	Update parameters through FWFS (*2)

*1: please refer to figure in chapter. FLASHLAYOUT
*2: please refer to chapter. FCS_MULTI

2.5 SDK and lib version

For more easily management prebuilt libraries version in SDK, each library has its own "get version" API and the output version string follows a specific format.
API in each library

```
char * lib<lib_name>_get_version(void);
```

Output version string format

```
lib<lib_name>:YEAR.MON.DAY.HOUR.MIN.SEC_b<branch_name>_<branch SHA1>
```

YEAR, MON, DAY, HOUR, MIN and SEC are the building date and time of this prebuilt lib.

For example
If user want to obtain the version information for a specific library, such as the "wlan" lib, user could implement the code like following code piece.

```
extern char* libwlan_get_version(void);
printf("%s\n\r", libwlan_get_version());
```

Device should output string like

```
libwlan:2023.05.01.12.20.00_b9.5_0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f
```

For the version of "video" lib,

```
extern char* libvideo get version(void);
printf("%s\n\r", libvideo_get_version());
```

Device should output string like

```
libvideo:2023.05.01.12.22.00_b9.5_0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e
```

Additionally, user could directly search for the version string by opening the prebuilt library binary file using a hex editor and searching for the string, for example "**libvidoe:**" or "**libwlan:**".

This version information is also useful for user for solving issues with factory support.

The API is only accessible when the libraries linked into caller application. Cannot get libraries' version linked into non-secure application from secure application, or vice versa.

2.5.1 Video version

If you need to get the version information of the video, please enable the streaming first and use **video_get_version()** to get the information.
The version information about the video has the following items

- VOE
- SENSOR DRIVER
- SENSOR TIMESTAMP
- FCS
- IQ TIMESTAMP
- IQ VERSION

The above information are all displayed in

```
void video_get_version()
```

Taking sensor gc4653 with FCS under voe 1.4.2.1 as example, after calling video_get_version(), you can see the content as follows:

```
voe ver: 1.4.2.1
sensor_voe_ver: 1.4.2.1
sensor timestamp: 2023/04/20
fcs_version: 0x5306
iq timestamp: 2023/05/12 16:04:30
iq_cus_ver: 0x01
```

Below is more description for above information,

voe_ver is an abbreviation for VOE version, where VOE is another core for ISP.

sensor_voe_ver means sensor VOE version, which is the sensor version on VOE.

sensor_timestamp is the release information of sensor driver.

fcs_version is the version of FCS driver.

iq_timestamp and **iq_cus_ver** are the release information of iq table.

- iq_timestamp record the date/time information of the IQ release day.
- iq_cus_ver record the version for specific IQ of an ongoing project.

At the same time, you can also use AT Command to display the version information of the video, the command is as follows:

```
ATII=version
```

If the user needs to show the respective version information on the application side, please refer to the content of video_get_version() for coding.

Note: Difference of libvideo_get_versonion() and video_get_version()

libvideo_get_versonion() is the version information in video driver running on main core v8m.

video_get_version() retrieve all the other version information from video offload engine, including VOE(bin file for Video offload engine), Sensor Driver, FCS, IQ Timestamp and IQ version.

2.5.2 Bootloader version

At offset 0x2B0 of bootloader image, the version is a 32bytes value in little endian order. The definition of version is explained in the "Version and Timestamp" section of OTA chapter. The version can be configured in 'amebapro2_bootloader.json' under 'project\realtek_amebapro2_v0_example\GCC-RELEASE\mp'.

```
"MANIFEST":{
"label":"RTL8735B",
"vrf alg": "NA VRF CHECK",
"tlv":[
{"type":"PK", "length":384, "value":"auto"},
{"type":"TYPE ID", "length":2, "value":"IMG BL"},
{"type":"VERSION",
 "length":32,
 "value":"FEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"},
{"type":"TIMST", "length":8, "value":"auto"},
```

For a bootloader image file in file system, file API, such as fopen() and fread() can be used to open the file and read bootloader version at the offset 0x2B0 of bootloader image file. For bootloader partition in flash, FWFS API, such as pfw_open() and pfw_read() can be used to open the partition and read bootloader version at the offset 0x2B0 of bootloader partition.

3 GCC Makefile

3.1 Adding files in CMake project

3.1.1 Adding sources and headers

In the section, we will introduce how to add files to AmebaPro2 project, including adding source, header files, creating and linking the library files.

(1) Adding Source files
Open the application.cmake at "project/realtek_amebapro2_v0_example/GCC-RELEASE/application/".
Add the source code by append to app_sources:

```
list(
  APPEND app_sources
  ...
  ${PATH_TO_YOUR_SOURCE_FILES}
  ...
)
```

(2) Adding header files
Open the includepath.cmake at "project/realtek_amebapro2_v0_example/GCC-RELEASE/".
Add the header files by append to inc_path_re:

```
list(
  APPEND inc_path_re
  ...
  ${PATH_TO_YOUR_HEADER_FILES}
  ...
)
```

Also add directory to be included by include_directories(<path to header folder>).

```
include_directories (${PATH_TO_YOUR_INCLUDE_DIR})
```

3.1.2 Adding library files

3.1.2.1 Method 1

You can place the library under "project/realtek_amebapro2_v0_example/GCC-RELEASE/application/output".
Assume your library file is libABC.a, you can modify application.cmake like:

```
target_link_libraries(
  ${app}
  ...
  ABC
  ...
)
```

3.1.2.2 Method 2

(1) Declare your library

```
add_library (<LIBRARY_NAME> STATIC IMPORTED)
```

(2) Setup location of your library by:

```
set_property (TARGET <LIBRARY_NAME> PROPERTY IMPORTED_LOCATION <PATH_TO_YOUR_LIBRARY>)
```

or

```
set_target_properties (<LIBRARY_NAME> PROPERTY IMPORTED_LOCATION <PATH_TO_YOUR_LIBRARY>)
```

(3) Link to your library

```
target_link_libraries(
  ${app}
  ...
  <LIBRARY_NAME>
  ...
)
```

3.1.3 Building a library

3.1.3.1 Create a cmake file for the library

(1) Set up minimum required cmake version and the project name. Here the output library file name will be libtest.a or libtest.so.

```
cmake_minimum_required(VERSION 3.6)
project(test)
set(test test)
```

(2) Append source files to project source list

```
list(APPEND test_sources
  $(PROJ_ROOT)/example/test01.c
  $(PROJ_ROOT)/example/test02.c
)
```

(3) Assign the library type, STATIC means that the library will be built as static-link library (*.a), while SHARED means that the library will be built as dynamic-link library (*.so).

```
add_library(
  ${test} STATIC ${test_sources}
)
```

(4) Add the compile flag for the library

```
list(APPEND test_flags
  CONFIG_BUILD_ALL=1
  CONFIG_BUILD_LIB=1
  ${YOUR COMPILE FLAGS}
)
```

(5) Add the header files need to be included in the library

```
include(../includepath.cmake)
target_include_directories(${test} PUBLIC
  ${YOUR INCLUDE DIRS}
)
```

3.1.3.2 Add the cmake and link the library to the project

You can include and link to your library in application.cmake files by:

```
include(./libtest.cmake)
...
target_link_libraries(
  ${app}
  ...
  test
  ...
)
...
```

3.1.3.3 Turn off the dependency of the library

If users do not want to rebuild their own library each time when modification is not related to their library, users can open the DependInfo.cmake under GCC-RELEASE/build/application/CMakeFiles/<YOUR_LIBRARY.dir> and turn on the CMAKE_DEPENDS_IN_PROJECT_ONLY.

```
# Consider dependencies only in project.
set(CMAKE_DEPENDS_IN_PRJECT_ONLY ON)
```

3.2 Creating a new application example

The application example folder of AmebaPro2 needs to have app_example.c and <EXAMPLE_FOLDER.cmake>. The app_example.c is the entry of the example and the cmake file is for project build. Here are the steps for building up a new application example.

(1) Create a folder under “sdk/component/example”, move the source code to the folder and add app_example.c and <EXAMPLE_FOLDER.cmake> in the folder.

(2) Open app_example.c and call to the entries of example under the function app_example

```
void app_example(void)
{
   example_audio_helix_aac();
}
```

(3) Append the source code, header file, compile flag and library needed under the lists in <EXAMPLE_FOLDER.cmake>.

(4) After done the previous steps, users can build up the new example project by:

```
cmake .. -G"Unix Makefiles" -DCMAKE TOOLCHAIN FILE=../toolchain.cmake -
DEXAMPLE=<EXAMPLE FOLDER>”
cmake --build . --target flash
```

3.3 How to use example source code

In this section, we will describe how to use the example source code for AmebaPro2

3.3.1 Application example source

AmebaPro2 application's example source codes which can be separate into function examples and integrated examples. For function examples, it is typically shows how to use the normal function provided in AmebaPro2 like SD, file, audio, file system, etc. For integrated examples, they provide some integrated application for AmebaPro2 functions, like video and audio streaming and simple doorbell-chime application.

Each example subfolder contains only one example, the entry function is app_example(void).

The example entry function is defined as app_example and only one example is exist in the same project.

3.3.1.1 Function examples

The function examples can be found under folder "sdk/component/example".

Here are steps to build up the example:

(1) Create example build folder in "project/realtek_amebapro2_v0_example/GCC-RELEASE" and enter it

```
cd project/realtek amebapro2 v0 example/GCC-RELEASE
mkdir build_example && cd build_example
```

(2) Use cmake to create makefile for example. The <EXAMPLE_FOLDER_NAME> can refer to the folders under sdk/component/example.

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -
DEXAMPLE=<EXAMPLE_FOLDER_NAME>
```

NOTE

If the example folder not exist, the "<EXAMPLE_FOLDER_NAME> Not Found" message will show, please check the example folder name

(3) If example configured successfully, run build command to generate flash image

```
cmake --build . --target flash
```

NOTE

In AmebaPro2 project, when -DEXAMPLE=<EXAMPLE_FOLDER_NAME> is used, the integrated function -DVIDEO_EXAMPLE=on and -DDOORBELL_CHIME=on will be set to off after the needed source imported.

3.3.1.2 Integrated examples

The function examples can be found under folder "sdk/project/realtek_amebapro2_v0_example/src".

- Video examples

  These examples could be found under "sdk/project/realtek_amebapro2_v0_example/src/mmfv2_video_example" and opened by using compiling flag –DVIDEO_EXAMPLE=on. The detail of these examples can refer to Multimedia Framework Architecture.

- Doorbell and chime example

  The example could be found under "sdk/project/realtek_amebapro2_v0_example/src/doorbell-chime", which provide users to construct a simple doorbell system which could push video and audio and get audio streaming though the Skynet. This example can be opened through -DDOORBELL_CHIME=on.

NOTE

In AmebaPro2 project, the flag of integrated could not be used in the same time.

3.3.2 MMF example source

In sdk/component/example/media_framework, it provides audio-only MMF examples. The examples are based on the Multimedia Framework Architecture and the detail can refer to **Multimedia Framework Architecture**.

3.3.3 Peripheral example source

The peripheral example sources are located at the folder sdk/project/realtek_amebapro2_v0_example/example_sources and basically provide main.c and readme file. The main.c file contains the usage of peripheral function and user should replace it with the original main.c (in SDK/project/realtek_amebapro2_v0_example/src). On the other hand, like application example source, the method to compile example and adjust the important parameters is described in the readme file. After the setting, user can rebuild the project with peripheral example.

3.3.4 WiFi example source

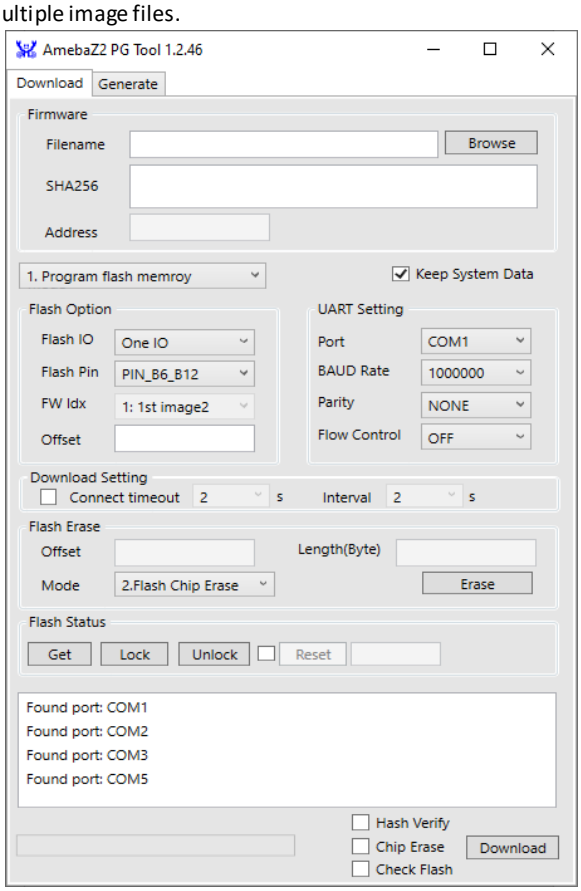
For user to test and development, we provide AT command in AmebaPro2. Users can key in AT command to connect WLAN by the console in PC. AT command can refer to "AN0025 Realtek at command.pdf".

4 Image Tool

4.1 Introduction

This chapter introduces how to use Image Tool to generate and download images. The image tool - PGTool can be found in tools folder and it has two functions:

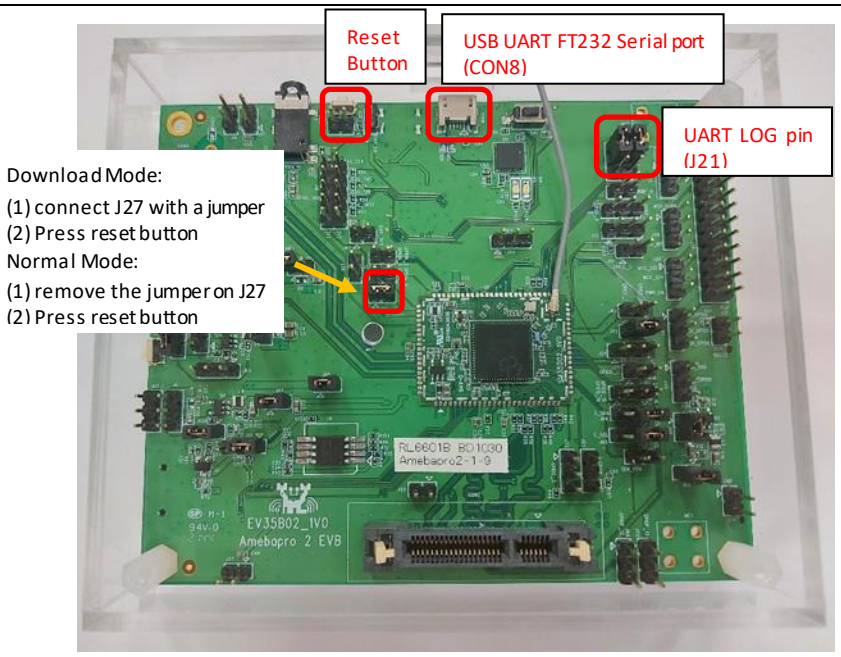
(1) Download image to an AmebaPro2 device through UART.
(2) Generate composited image from multiple image files.



4.2 Download Environment Setup

4.2.1 Hardware Setup

To download image, the device must to be booted as download mode. Users need to first set up the UART with PC by connecting J21 with jumpers and CON8 with PC. Then, connect J27 with a jumper and press reset button to enter download mode.



4.2.2 Software Setup

- PC environment requirements: Windows 7 above with FT232 driver.
- PGTool

4.3 Image Download

User can download the image to demo board by following steps:

(1) Boot AmebaPro2 into download mode

NOTE

You can check whether your board is in download mode by UART message:

```
== Rtl8735b IoT Platform ==
Chip VID: 0, Ver: 0
ROM Version: v3.0
Test Mode: boot_cfg1=0x0

[test mode PG]
test_mode_img_download
Download Image over UART1[tx=4,rx=3] baud=115200
```

CAUTION

Remember to disconnect log UART console before downloading image

(2) Run image tool

(3) Browse your image: flash_ntz.bin (other /*.bin file is used in other application or just to confirm the version is correct or not)

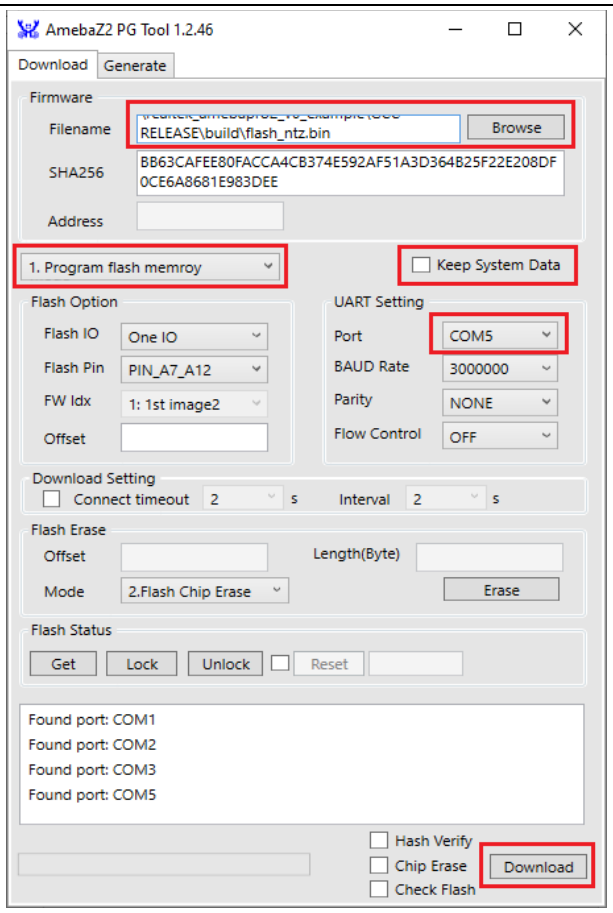
(4) Choose "1. Program flash memory" and select the correct COM port

(5) Disable "Keep System Data" to prevent some errors

(6) Press the Download button and the image will start to be downloaded to AmebaPro2 device

NOTE

Flash Pin will be set automatically, we do not need to set



NOTE

Once the flash could not boot up successfully, user could build an image without any examples, erase the flash (refer to **Erase Flash of device***) and download the building image.*

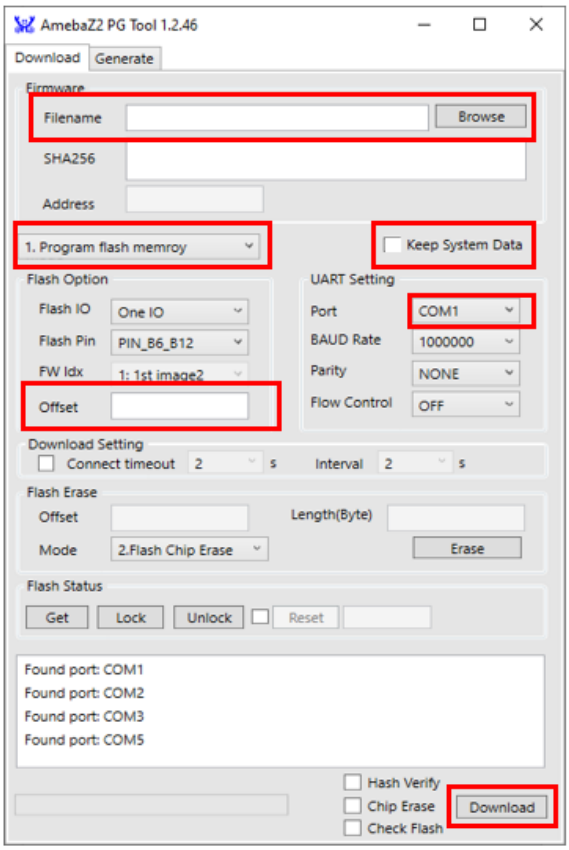
4.4 Partial image Download

User can download the image to demo board by following steps:

(1) Boot AmebaPro2 into download mode. Remember to disconnect log UART console before downloading image.
(2) Run image tool
(3) Browse your image: firmware.bin, which is FW1 or FW2
(4) Choose "1. Program flash memory" and select the correct COM port
(5) Disable "Keep System Data" to prevent some errors
(1) Input "Offset" which has to be 4-byte aligned. For the starting address of FW, please refer to NOR Flash Layout in amebapro2_partitiontable.json
(2) Press the Download button and the image will start to be downloaded to AmebaPro2 device

NOTE

Flash Pin will be set automatically, we do not need to set



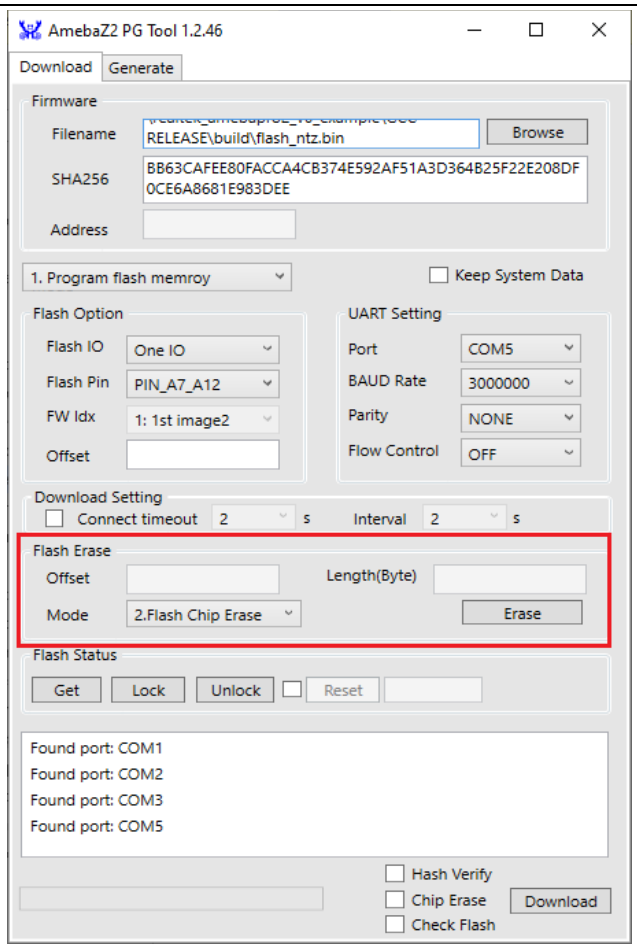
4.5 Erase Flash of device

Image tool also provides user to erase the flash of the device, it has two mode:

(1) Flash Sector Erase: erase the flash from the Offset with a Length of Byte that the user set.
(2) Flash Chip Erase: erase the whole flash

NOTE

After you press the erase button, it needs time for erasing the flash. Please wait until the PG tool shows the "erase successfully" message



4.6 Command line image tool for NOR flash

Command line image tool supports Windows, Linux and MacOS platforms:

- Windows: uartfwburn.exe
- Linux: uartfwburn.linux
- MacOS: uartfwburn.darwin

NOTE

boot_recover.bin & flash_loader_nor.bin & flash_control_info.bin needs to be placed in the same folder as uartfwburn tool

4.6.1 Normal PG mode

Write whole image to NOR flash, run command:

```
uartfwburn -p <COM_PORT> -f flash_ntz.bin -b 3000000 -U
```

NOTE

MacOS: <COM_PORT> use /dev/cu.xxxxx

If success, command will print: nor download success

4.6.2 Partial image PG mode

To set the offset and write part of the image to NOR flash, run the command:

```
uartfwburn -p <COM_PORT> -f XXX.bin -b 3000000 -s 0x100000 -U
```

-s [offset]: image download offset(hex, eg, 0x100000)

NOTE

Offset address needs to be 64K byte aligned.

4.7 Command line image tool for NAND flash

Command line image tool supports Windows, Linux and MacOS platforms:

- Windows: uartfwburn.exe
- Linux: uartfwburn.linux
- MacOS: uartfwburn.darwin

NOTE

boot_recover.bin & flash_control_info.bin needs to be placed in the same folder as uartfwburn tool

4.7.1 Normal PG mode

Write whole image to NAND flash, run command:

```
uartfwburn -p <COM_PORT> -f flash_ntz.bin -b 3000000 -n pro2
```

NOTE

MacOS: <COM_PORT> use /dev/cu.xxxxx

*I*f success, command will print: nand download success

4.7.2 Partial image PG mode

Write partial image to NAND flash, run command:

```
uartfwburn -p <COM_PORT> -f flash_ntz.bin -b 3000000 -n pro2 -t 0x81cf
```

-t [type_id] : pro2 nand flash partial image download, refer to the table below.

Short name	Size (Bytes)	Type ID
PT_KEY_CER1	2	0xe9c2
PT_BL_BRI	2	0xd1c5
PT_FW1	2	0xc1c7
PT_FW2	2	0xb9c8
PT_ISP_IQ	2	0x89c1
PT_NN_MDL	2	0x81cf

NOTE

*I*f success, command will print: nand download success

4.7.3 FTL user data PG mode

Write FTL user data to NAND flash, run command:

```
uartfwburn -p <COM_PORT> -f user.bin -b 3000000 -n pro2 -w 800 950
```

-w [block_s] [block_bs] : pro2 nand FTL write, block_s is block start, block_bs is block backup start.

NOTE

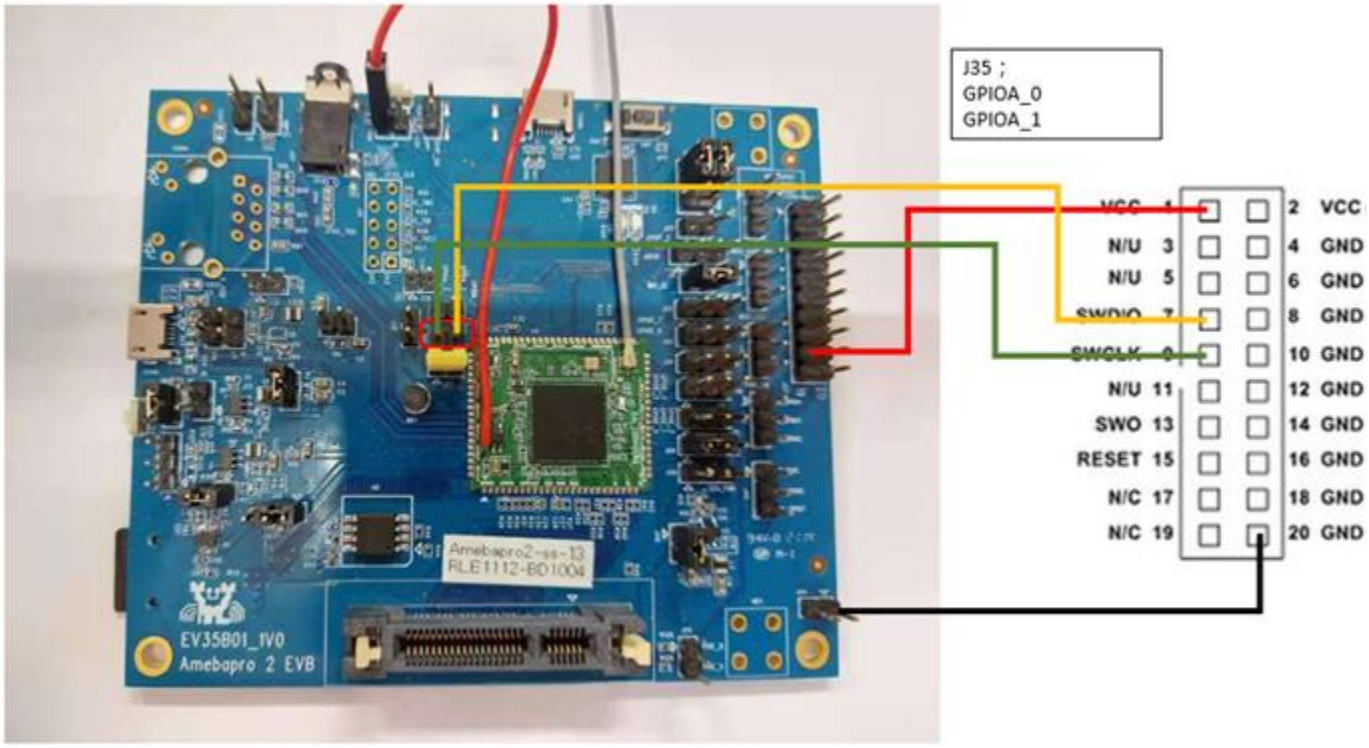
*I*f success, command will print: nand download success

5 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.

5.1 SWD connection

Ameba Pro2 supports J-Link debugger. We need to connect the SWD connector to J-Link debugger. The SWD connection is shown as below. After finished these configurations, please connect it to PC side. Note that if you are using Virtual Machine as your platform, please make sure the USB connection setting between VM host and client is correct so that the VM client can detect the device.



NOTE

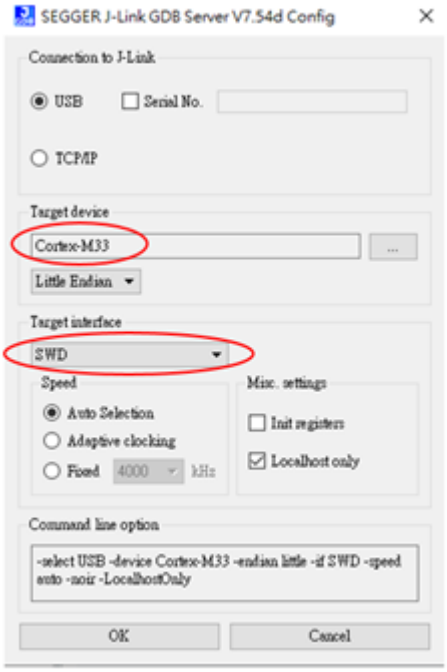
- *EVB 1V0 module needs to HW rework, ask FAE for details*
- *To be able to debugger AmebaPro2 which is powered by Cortex-M33, user needs a J-Link debugger with the latest hardware version (Check https://wiki.segger.com/Software_and_Hardware_Features_Overview for details). J-Link EDU with hardware version V10 is used to prepare this document*

5.2 Software installation

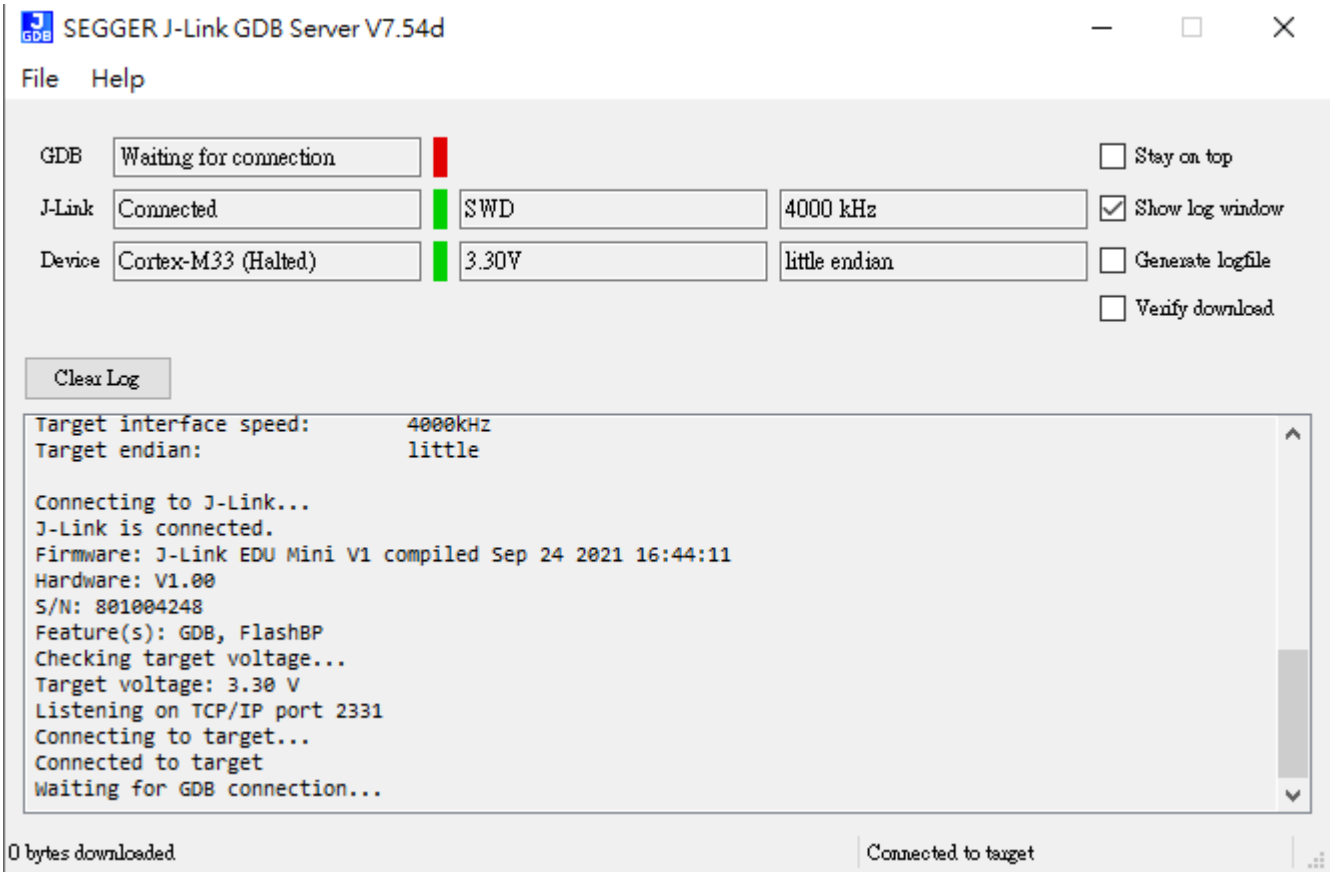
To be able to use J-Link debugger, user needs install J-Link GDB server first. For Windows, please check http://www.segger.com and download "J-Link Software and Documentation Pack" (https://www.segger.com/downloads/jlink).

5.3 Setup environment

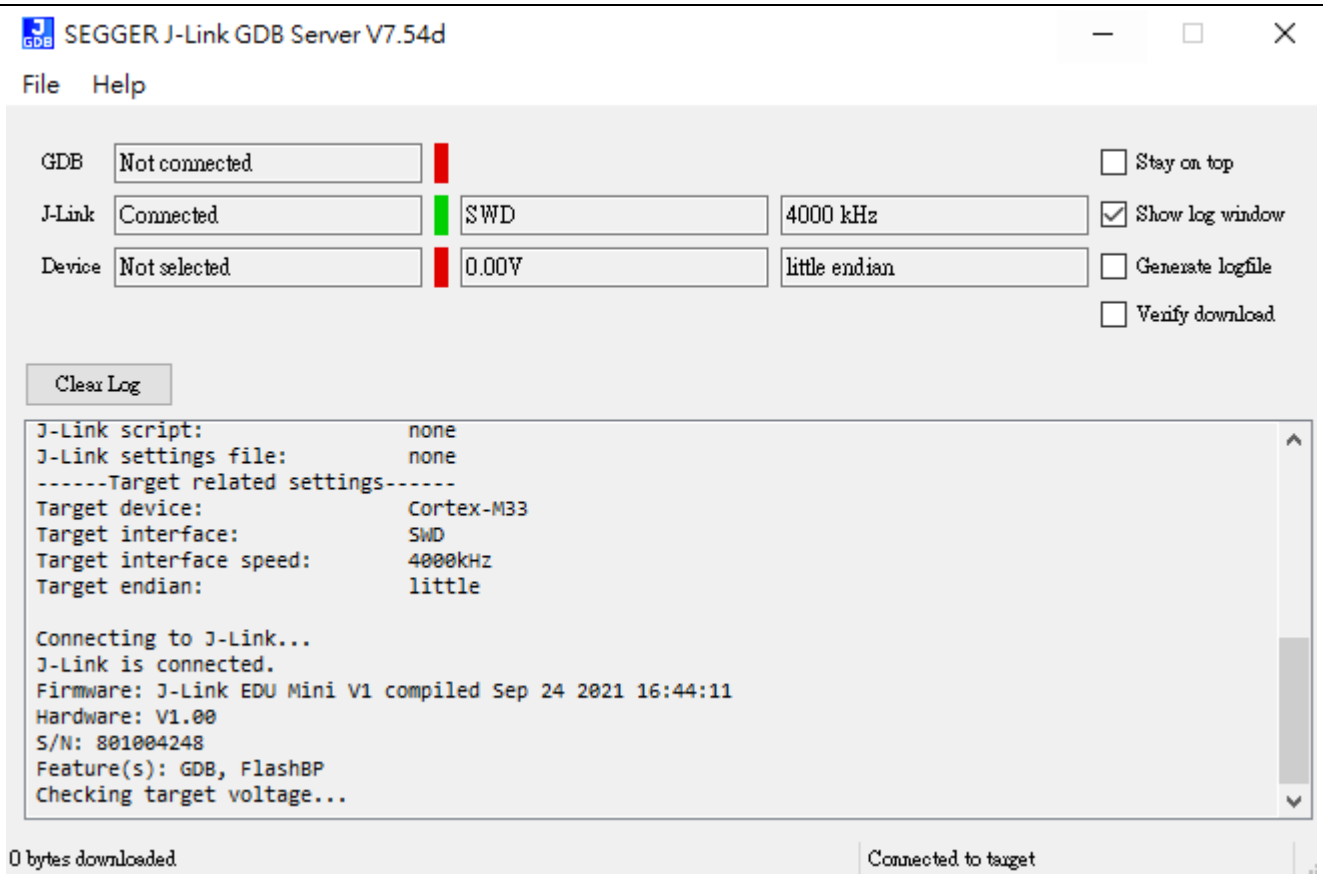
To check whether the connection works fine, user can go to the location of SEGGER J-Link tool and run "JLinkGDBServer.exe". Choose target device Cortex-M33 (for AmebaPro2), and target interface SWD. Click "OK"



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



If connection fails, J-Link GDB will show:



6 Multimedia Framework Architecture

The Multimedia Framework Architecture version 2(MMFv2) is responsible for handling the connection and management of different media resources on AmebaPro2.

6.1 Architecture

The structure of MMFv2 is as shown in the following chart and there are two important entities in the MMFv2, **MM_MODULE** and **LINKER_MODULE**:

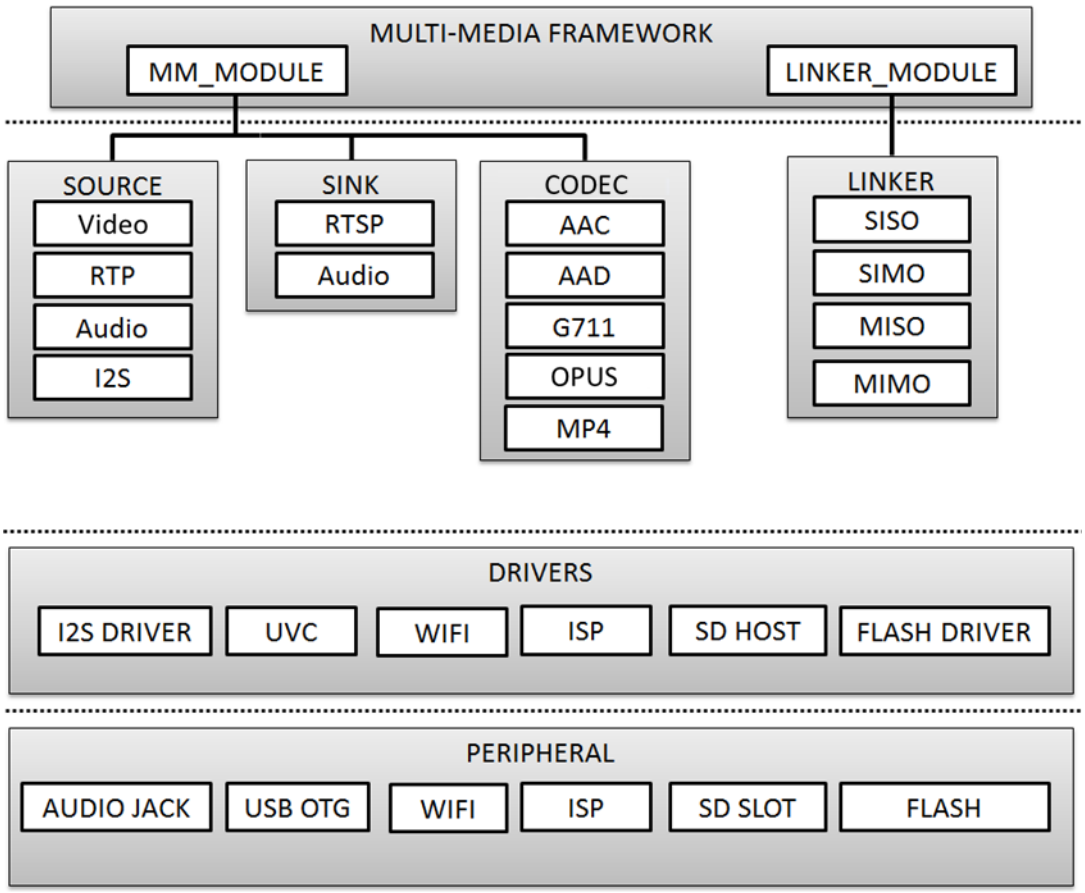
MM_MODULE includes the media source, sink and the codec modules.

- Source module: produce resource, it can be the file input, microphone, camera, or storage.
- Codec module: mainly provide the audio codec, AAC, G711 or opus for customers to do audio encode or decode before sending streaming to sink module. In the mp4 module, it will automatically send the result into storage, SD card or ram disk.
- Sink module: consume resource from the source modules or after encoded/decoded by codec modules, like RTSP or other steaming.

NOTE

The video modules uses VOE to contain the process of sensor catching, ISP and video encoding algorithms (jpeg, H264, HEVC (H265)…).

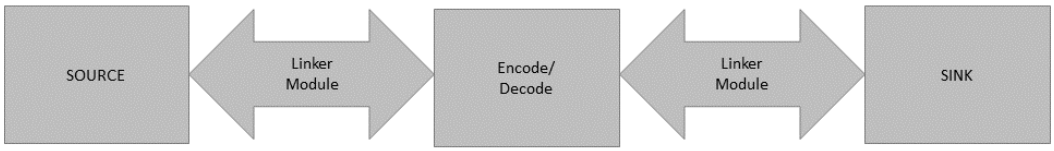
LINKER_MODULE connect different type of module and deal with inter module communication, included siso, simo, miso and mimo.



In order to use the MMFv2, here are some aspects must to be followed.

- Define valid source
- Define valid sink
- Define valid codec (encode/decode) if needed.
- Define valid linker modules to link the above media modules.

The following picture shows the main usage flow to initialize different **MM_MODULE**, and connect different **MM_MODULE** through **LINKER_MODULE**.



6.1.1 MM_Module Prototype

MMFv2 allows users to define customized source, sink and encoder/decoder modules depending on the application. Although implementation details may be different, basic rules of the MMF structure are similar.

The MMFv2 requires users to predefine both source and sink modules through implementing create, destroy, control, handle, new_item, del_item and rsz_item function callbacks. The structure mmf_module_t provides the interface for communication between mmf modules. In order to maintain the flexibility and convenience between modules, modules only retain the interface of each type to provide module to access. Function's constant of each module is defined by module itself.

```
typedef struct mm module s {
        void*               (*create)(void *);
        void*               (*destroy)(void *);
        int                 (*control)(void *, int, int);
        int                 (*handle)(void *, void *, void *);
        void*               (*new_item)(void *);
        void*               (*del_item)(void *, void *);
        void*               (*rsz_item)(void *, void *, int);
        void*               (*vrelease item)(void *, void *, int);

        uint32_t             output_type;
        uint32_t             module_type;
        char *               name;
} mm_module_t;
```

6.1.1.1 Function description

- create

Pointer to the function that loads and initializes the module that you wish to add. For example, for Audio source, it points to the function in which the Audio driver is initialized and the corresponding context is returned.

- destroy

Pointer to the function that de-initializes module instance and releases resource. For example, for Audio source, it points to function in which Audio driver is initialized and the corresponding context is released.

- control

Pointer to function that sends the control command to the MMF module layer (see **mm_module_ctrl**) or a specific module. For example, for Audio source, it points to function that controls Audio parameters ("sample rate", "word length", "mic gain", etc.) and MMFv2 service task on or off.

- handle

Pointer to the function that manipulates media data (how to produce data in source or how to consume data in sink). Data is transferred from source to sink and vice versa by means of OS message queue. Please note that MMF service task reacts differently based on message exchange buffer status.

- new_item

Pointer to the function that creates queue item that will be send to input and output queue, will only be used when setting MM_CMD_INIT_QUEUE_ITEMS to MMQI_FLAG_STATIC.

- del_item

Pointer to the function that destroys queue item, will only be used when setting MM_CMD_INIT_QUEUE_ITEMS to MMQI_FLAG_STATIC.

- rsz_item

Pointer to the function decreases memory pool size, will only be used when video (H264, HEVC (H265)…) and AAC module is created.

- output_type and module_type

Output_type indicates output mode. There are MM_TYPE_NONE, MM_TYPE_VSRC, MM_TYPE_ASRC, MM_TYPE_VDSP, MM_TYPE_ADSP, MM_TYPE_VSINK, MM_TYPE_ASINK, and MM_TYPE_AVSINK can be used, corresponding to different module usage scenarios, let application know which mode the output is. module_type represents the identity of the module, and there are three options can be used MM_MASK_SRC, MM_MASK_DSP and MM_MASK_SINK.

- name

Pointer to the module name.

6.1.1.2 mm_module_ctrl

Here lists some commands defined in MMF module layer. Call by mm_module_ctrl (mm_context_t *ctx, int cmd, int arg) to use them.

- MM_CMD_INIT_QUEUE_ITEMS: initialize static queue item.
- MM_CMD_SET_QUEUE_LEN: Set one queue's length.
- MM_CMD_SET_QUEUE_NUM: Set number of queue, not more than 3.
- MM_CMD_SELECT_QUEUE: select queue from multi queues.
- MM_CMD_CLEAR_QUEUE_ITEMS: clear queue item.

6.1.2 Context

MMFv2 context supplies message transfer between different modules. It contains mm_module_t, and queue that used to pass data. There are 6 types of status that mm_context support (MM_STAT_INIT, MM_STAT_READY, MM_STAT_ERROR, MM_STAT_ERR_MALLOC, MM_STAT_ERR_QUEUE, MM_STAT_ERR_NEWITEM), these status are responsible for maintaining the module state to ensure the program runs smoothly.

```
typedef struct mm contex s {
        union {
                struct {
                        xQueueHandle    output ready;
                        xQueueHandle    output_recycle;
                        int32_t                 item_num;
                };
                mm_conveyor_t         port[4];
        };

        mm_module_t*   module;

        void*                       priv;   // private data structure for created instance

 // module state
        uint32_t                state;
        int32 t                 queue num;      // number of queue
        int32_t                 curr_queue;
} mm_context_t;
```

The mm_context is responsible for maintaining each module entity. MMFv2 support these modules (video, AAC_encoder, AAC_decoder, audio, g711, opus, mp4, rtp, rtsp) by default. Each module is independent and corresponding to the individual input/output queue, state and in the mm_context of the module to update parameters and delivery entities.

6.1.3 Module Inter Connection

This section introduces mm_siso_t, mm_simo_t, mm_miso_t, mm_mimo_t and its corresponding create, delete, ctrl, start, stop, pause, resume function, which is responsible for connection and control between modules in mmfv2.

6.1.3.1 SISO module (Single Input Single Output)

The SISO module is a unidirectional interface between modules. Input and output are independent. The status of the SISO module is responsible for determining the correct process. The stack_size is used to determine the size of the handler, while xTaskHandle task, task_priority and taskname are reserved to control the use of the task, task priority and task name.

```
typedef struct mm_siso_s {
        mm_context_t *input;
        mm_context_t *output;
        int             input_port_idx;
// default is 0, can be set to 1 or 2 or 3 if source module support 2 or more output queue

        uint32_t     status;
        uint32_t     stack_size;
        uint32_t     task priority;
        char         taskname[16];
        xTaskHandle  task;
} mm_siso_t;
```

There are some functions in the SISO module responsible for the module inter-connection. By these functions, it will be simple to update the status of the task and are handed over to the task handler for the main processing:

- siso_create

Pointer to the function that siso_create declares the space of mm_siso_t and returns mm_siso_t entity after initialization.

- siso_delete

Pointer to the function that stops SISO execution and free space of mm_siso_t entity.

- siso_ctrl

Pointer to the function that sends the control command to siso module.
MMIC_CMD_ADD_INPUT link the input module to the input of the siso module.
MMIC_CMD_ADD_OUTPUT link the output module to the output of the siso module.
MMIC_CMD_SET_TASKPRIORITY set the task priority for the linker task. If setting as 0, it will be configured to tskIDLE_PRIORITY + 1 automatically.
MMIC_CMD_SET_TASKNANE set the task names for the linker task.
MMIC_CMD_SET_STACKSIZE add size to the stack_size of siso.

NOTE

For consistency, the setting task size will be divided by 4. Make sure setting an enough and valid stack_size for the task.

- siso_start

Pointer to the function that checks whether there is anything in the input and output module before siso start. If the answer is yes, siso task will create a task handler to send data from input module to the output module.

- siso_stop

Pointer to the function that updates status to MMIC_STAT_SET_EXIT and wait for task handler to switch status to MMIC_STAT_EXIT.

- siso_pause

Pointer to the function that updates status to MMIC_STAT_SET_PAUSE and wait for task handler to switch status to MMIC_STAT_PAUSE.

- siso_resume

Pointer to the function that updates status to MMIC_STAT_SET_RUN and wait for the task handler to switch status to MMIC_STAT_RUN.

6.1.3.2 SIMO module (Single Input Multiple Output)

The SIMO module is a unidirectional interface between modules. Input and output are independent, and output_cnt represents the number of simultaneous output modules. The array – status[4] maintains the state of the SIMO module to check the process is correct in the middle of the transfer, stack_size is used to determine the size of the handler task for intermediate transfers. Similarly, it also provides xTaskHandle task, task_priority, taskname for xTaskCreate. Note that each output will be served by one unique task and pause mask will control which output will be blocked.

```
typedef struct mm_simo_s {
        mm_context_t *input;
        int                             output cnt;
        mm_context_t *output[4];
 // internal queue to handle reference count and usage log
        mm_simo_queue_t queue;

        uint32_t      pause_mask;
        uint32_t      status[4];;
        uint32_t      stack_size;
        uint32_t      task_priority;
        char          taskname[4][16];
        xTaskHandle   task[4];
} mm_simo_t;
```

There are some functions in the SIMO module responsible for the module inter-connection. By these functions, it will be simple to update the status of the task and are handed over to the task handler for the main processing:

- simo_create

Pointer to the function that simo_create dedares the space of mm_simo_t entity and returns mm_siso_t after initialization, and simo_create crate a queue head and a queue lock to protect the results of multiple outputs.

- simo_delete

Pointer to the function that calls simo_stop() to stop SIMO execution and free space.

- simo_ctrl

Pointer to the function that sends the control command to simo module.
MMIC_CMD_ADD_INPUT link the input module to the input of the simo module.
MMIC_CMD_ADD_OUTPUT0, MMIC_CMD_ADD_OUTPUT1, MMIC_CMD_ADD_OUTPUT2, MMIC_CMD_ADD_OUTPUT3 link output module to the corresponding output and increase the output_cnt to record number of output modules.
MMIC_CMD_SET_TASKPRIORITY set the task priority for the linker task. If setting as 0, it will be configured to tskIDLE_PRIORITY + 1 automatically.
MMIC_CMD_SET_TASKNANEx set the task names for the linker task corresponding to MMIC_CMD_ADD_OUTPUTx (x = 0~3).
MMIC_CMD_SET_STACKSIZE add size to simo stack_size.

NOTE

For consistency, the setting task size will be divided by 4 and it means each task will only have task_size/4 for task stack size. Make sure setting an enough and valid stack_size for the task.

- simo_start

Pointer to the function that simo_start will create corresponding number of task handlers based on simo -> output_cnt, and each task handler

will be used to send the received data.

- simo_stop

Pointer to the function that simo_stop sets each simo status to MMIC_STAT_SET_EXIT,and waits for the task handler to switch each status to MMIC_STAT_EXIT.

- simo_pause

Pointer to the function that simo_pause will set each simo -> status to MMIC_STAT_SET_PAUSE according to pause_mask, and wait for the task handler to switch each status to MMIC_STAT_PAUSE.

- simo_resume

Pointer to the function that simo_resume will set each simo -> status to MMIC_STAT_SET_RUN, and wait for the task handler to switch each status to MMIC_STAT_RUN.

6.1.3.3 MISO module (Multiple Input Single Output)

The MISO module is a unidirectional interface between modules. Input and output are independent, and input_cnt represents the number of simultaneous input modules. The status maintains the state of the MISO module to check the process is correct in the middle of the transfer, stack_size is used to determine the size of the handler task for intermediate transfers, and finally the xTaskHandle task, task_priority and taskname are reserved for xTaskCreate to control the use of the task. The pause_mask can be controlled to block the inputs or the single output.

```
typedef struct mm_miso_s {
        int                         input_cnt;
        mm_context_t *input[4];  // max 4 input
        int                         input_port_idx[4];

        mm_context_t *output;

        uint32_t     pause_mask;
        uint32_t     status;
        uint32_t     stack_size;
        uint32_t     task_priority;
        char         taskname[16];
        xTaskHandle  task;
} mm_miso_t;
```

There are some functions in the MISO module responsible for the module inter-connection. By these functions, it will be simple to update the status of the task and are handed over to the task handler for the main processing:

- miso_create

Pointer to the function that space of mm_miso_t is declared in miso_create and initialized to return mm_miso_t entity.

- miso_delete

Pointer to the function that calls miso_stop() to stop MISO and free space.

- miso_ctrl

Pointer to the function that sends the control command to miso module.
MMIC_CMD_ADD_INPUT0, MMIC_CMD_ADD_INPUT1, MMIC_CMD_ADD_INPUT2, MMIC_CMD_ADD_INPUT3 couple input modules to the corresponding miso input and increase the value of input_cnt for number of input module.
MMIC_CMD_ADD_OUTPUT links the output module to the output of the miso module.
MMIC_CMD_SET_TASKPRIORITY set the task priority for the linker task. If setting as 0, it will be configured to tskIDLE_PRIORITY + 1 automatically.
MMIC_CMD_SET_TASKNANE set the task names for the linker task.
MMIC_CMD_SET_STACKSIZE add size to miso stack_size.

NOTE

For consistency, the setting task size will be divided by 4. Make sure setting an enough and valid stack_size for the task.

- miso_start

Pointer to the function that checks whether there is anything in the input and output module before starting. If the answer is yes, a task handler will be created, and the data of the input module will be sent to the output module.

- miso_stop

Pointer to the function that sets the miso status to MMIC_STAT_SET_EXIT and wait for the task handler to switch the status to MMIC_STAT_EXIT.

- miso_pause

Pointer to the function that miso_pause will set miso -> status to MMIC_STAT_SET_PAUSE according to pause_mask, waiting for the task handler to switch status to MMIC_STAT_PAUSE.

- miso_resume

Pointer to the function that miso_resume will set miso -> status to MMIC_STAT_SET_RUN, waiting for the task handler to switch each status to MMIC_STAT_RUN.

6.1.3.4 MIMO module (Multiple Input Multiple Output)

The MIMO module is a unidirectional interface between modules, Input[4] and output[4] represent input and output modules respectively, and input_cnt represents the number of simultaneous input modules. Input and output support up to 4 outputs at the same time, MIMO module also needs mm_mimo_queue_t queue[4] to maintain the synchronization problem of each input queue. Each mm_mimo_queue_t has a lock and head to record the beginning of each queue and whether a program is already in use. The array, status[4], maintains the state of the MIMO module to determine the correct process in the middle of the transfer, stack_size is used to determine the size of the handler task for the intermediate transfer, and the xTaskHandle task of xTaskCreate is reserved to control the use of the task. The array, pause_mask[4], is used to control the input or output streaming for each task.

```
typedef struct mm_mimo_s {
        int                 input_cnt;

 // depend on intput count
        mm_context_t*       input[4];
        mm_mimo_queue_t     queue[4];

        int                 output cnt;

 // depend on output count
        uint32_t            pause_mask[4];
        mm_context_t*       output[4];     // output module context
        uint32 t            output dep[4]; // output depend on which input, bit mask
        uint32_t            input_mask[4]; // convert from output_dep, input referenced by
which output, bit mask
        uint32 t            status[4];
        uint32_t            stack_size;
        uint32_t            task_priority;
        char                taskname[4][16];
        xTaskHandle         task[4];
} mm_mimo_t;
```

There are some functions in the MIMO module responsible for the module inter-connection. By these functions, it will be simple to update the status of the task and are handed over to the task handler for the main processing:

- mimo_create

Pointer to the function mimo_create declares the space of mm_mimo_t entity and returns mm_mimo_t after initialization.

- mimo_delete

Pointer to the function that calls mimo_stop() to stop the mimo module and free space.

- mimo_ctrl

Pointer to the function that sends the control command to miso module.
MMIC_CMD_ADD_INPUT0, MMIC_CMD_ADD_INPUT1, MMIC_CMD_ADD_INPUT2, and MMIC_CMD_ADD_INPUT3 link input module to the input corresponding to the mimo module and increase the value of input_cnt to record the number of input modules.
MMIC_CMD_ADD_OUTPUT0, MMIC_CMD_ADD_OUTPUT1, MMIC_CMD_ADD_OUTPUT2, and MMIC_CMD_ADD_OUTPUT3 couple the output module to the output of the mimo module and increase the value of output_cnt to record the number of output modules. The inputs corresponding to outputs modules can be set by arg2 of mimo_ctrl using the union of MMIC_CMD_ADD_INPUTx.
MMIC_CMD_SET_TASKPRIORITY set the task priority for the linker task. If setting as 0, it will be configured to tskIDLE_PRIORITY + 1 automatically.
MMIC_CMD_SET_TASKNANE set the task names for the linker task.

NOTE

For consistency, the setting task size will be divided by 4 and it means each task will only have task_size/4 for task stack size. Make sure setting an enough and valid stack_size for the task.

- mimo_start

Pointer to the function that mimo_start will generate corresponding task handler according to output_cnt to transfer the received data.

- mimo_stop

Pointer to the function that mimo_stop will set the mimo status to MMIC_STAT_SET_EXIT according to output_cnt, and waiting for the task handler switch the status to MMIC_STAT_EXIT.

- mimo_pause

Pointer to the function that miso_pause will set each mimo -> status to MMIC_STAT_SET_PAUSE according to pause_mask, and waiting for the task handler to switch status to MMIC_STAT_PAUSE.

- mimo_resume

Pointer to the function that mimo_resume will set mimo -> status in the task of MMIC_STAT_PAUSE for each status to MMIC_STAT_SET_RUN, and waiting for the task handler to switch each status to MMIC_STAT_RUN.

6.2 MM_Module Type and Module Parameter

6.2.1 Video

The video module processes the data from sensor and outputs the video streaming data for user.
Here shows the context of the video module.

```
typedef struct video_ctx_s {
        void *parent;

        hal_video_adapter_t *v_adp;
        void *mem_pool;

        video_params_t params;
        int (*snapshot_cb)(uint32_t, uint32_t);
        void (*change_parm_cb)(void *);
        video_state_t state;
} video_ctx_t;
```

- v_adp: Point to the video adapter which will use in the video process.
- params: Basic parameters for the video module.
- snapshot_cb: Set the callback function for snapshot, which will be called while doing snapshot. It could be set by using CMD_VIDEO_SNAPSHOT_CB.

6.2.1.1 Basic video module parameters setting

Presetting the voe_heap_size:
Use **CMD_VIDEO_SET_VOE_HEAP** to set up the heap size that will be used in the voe process, including the output buffer for ISP (, snapshot) and Encoder, before setting the video parameters.
Here are some video module parameters provided to set.

```
typedef struct video_param_s {
        uint32_t stream_id;
        uint32_t type;
        uint32_t resolution;
        uint32_t width;
        uint32_t height;
        uint32_t bps;
        uint32_t fps;
        uint32_t gop;
        uint32_t rc_mode;
        uint32_t jpeg_qlevel;
        uint32_t rotation;
        uint32_t out_buf_size;
        uint32_t out_rsvd_size;
        uint32_t direct_output;
        uint32_t use_static_addr;
        uint32_t fcs;
        uint32_t use_roi;
        struct video_roi_s {
                uint32_t xmin;
                uint32_t ymin;
                uint32_t xmax;
                uint32_t ymax;
        } roi;
} video_params_t;
```

Use **CMD_VIDEO_SET_PARAMS** to set up the VIDEO parameters.

- stream_id: Select the ISP channel, it can be set from 0~4.
- type: Select the video encode type. Currently support HEVC (VIDEO_HEVC), H264 (VIDEO_H264), JPEG (VIDEO_JPEG), NV12 (VIDEO_NV12), RGB (VIDEO_RGB), NV16 (VIDEO_NV16), HEVC+JPEG (VIDEO_HEVC_JPEG) and H264+JPEG (VIDEO_H264_JPEG).
- resolution: Set the video frame resolution. Currently support VIDEO_QCIF (144*176), VIDEO_CIF (288*352), VIDEO_WVGA (360*640), VIDEO_VGA (480*640), VIDEO_D1 (480*720), VIDEO_HD (720*1280), VIDEO_FHD (1080*1920), VIDEO_3M (1536*2048), VIDEO_5M (1944*2592).
- width: Set the video frame resolution's width.
- height: Set the video frame resolution's height.

- bps: Configure the video encoder's bit rate (bits per second).
- fps: Configure the video module output frame rate (frames per second).
- gop: Set the group of the picture which can be seem as the cycle that I frame will update.
- rc_mode: Determine use CBR (1) or VBR (2).
- direct_output: If set 1, the video module output will not be sent to the video module output ready queue.
- use_static_addr: If setting use_static_addr to 1, the output_item data address will directly point to the isp_addr; while setting to 0, it will allocate a new space for the output item address.
- use_roi: If set 1, the video will be cropped according to roi parameter settings.
- roi: If use_roi be set to 1, the original video will be cropped using the area defined by the four parameters (xmin, ymin, xmax, ymax), and then the cropped image will be resized according to the video frame resolution setting (width, height). For example, If set (xmin, ymin, xmax, ymax) to (0, 0, 800, 800), it will crop the top left corner of the image to a width of 800px and a height of 800p x from the origin of video frame, and the cropped image will be scaled down to width x height according to resolution of video.

NOTE

In VOE, OSD is applied after the cropped and resized image, so OSD size and offset are not affected by video cropping and resizing.

Video resolution alignment

Encoder input width require 16 alignment, so video module will do width alignment automatically. For example, if user set (width, height) to (1080, 1080), ISP will give 1088x1080 video frame to encoder. Then, encoder will encode the data and crop to 1080x1080 as video module output. ROI region parameters xmin, ymin, xmax, ymax should be 2 aligned and within the maximum resolution of the sensor. In addition, the roi_w(a.k.a. xmax-xmin) should no less than 16-aligned video_w (a.k.a. 16-aligned width) and roi_h(a.k.a. ymax-ymin) should no less than video_h(a.k.a. height), because the ROI only support scale down. In other words, if user want a 1080x1080 output of video module and require the usage of ROI, the ROI region width should >=1088 and ROI region height should >=1080.

NOTE

Sensor model and ISP will restrict the video resolution and fps. For the sensor's max resolution and fps, please check the sensor list in ISP chapter. The ISP supported max resolutions for each channel are as followed:

- *Ch0: 2704 x 1960*
- *Ch1: 1920 x 1080*
- *Ch2: 2592 x 1688*
- *Ch4: 1280 x 720*

Please confirm the selected width, height and fps of each video channel are within the limits.
For VOE 1.4.3.0 and its later version, the ISP supported max resolution for all channel is 2704 x 1960.

6.2.1.2 Resolution adjustment

Ameba pro2 ISP support scaling down function with non-aspect ratio window (it should be less than sensor output size). User can set 「use_roi」 to enable this function. Take example for 1080P sensor below:

```
typedef struct video_param_s {
        …
        uint32_t width;             // 640
        uint32_t height;            // 480
        …
        uint32_t use_roi;
        struct video_roi_s {
               uint32_t xmin;       // 240
               uint32_t ymin;       // 0
               uint32_t xmax;       // 1679
               uint32_t ymax;       // 1079
        } roi;
} video_params_t;
```

Table 6-1 Image aspect ratio example image (full view)

Original Image (Full Size with Crop Window)	Output Image (After scaling down)

Table 6-2 Image aspect ratio example image (partial view)

Original Image (Full Size with Crop Window)	Output Image (After Crop with scaling down)
	

6.2.1.3 Video module rate control (RC) adjustment

Ameba pro2 support two bit rate control mode, Variable Bitrate (VBR) and Constant Bitrate (CBR), all based on frame level rate control.

- Variable bitrate mode (VBR):

Taking the set 1/2 bps as the target bitrate, the actual picture quality is optimize through the set minQp and maxQp. When the scene can be effectively compressed to reduce the bitrate, the compressed Qp will go to minQp until minQP is the best picture set. When the scene cannot effectively compress the bitrate, the compression Qp goes to maxQp until maxQP is the maximum compression rate. At this time, if the maxQP setting is larger, the compression efficiency will be better. Exceeds the set Max bitrate. QP range default value is [20, 45], If there is an adjustment requirement in the [minQp, maxQp] control of VIDEO_SET_RCPARAM.

- Constant bitrate mode (CBR):

Fixed bit rate, bit rate is control by bps setting, QP range default value is [0, 51], If there is an adjustment requirement in the [minQp, maxQp] control of VIDEO_SET_RCPARAM.

Ameba pro2 provide four parameters for image quality adjustment, Adjust the deviation of the direct QP of I frame and P frame, and control the size ratio between I frame and P frame. The smaller the QP of I frame, the larger I frame, and the clearer the image, which improves the overall image quality to a certain extent.

However, I frame cannot be adjusted too large. I frame eats up all the bandwidth, and P frame can only be edited to be more blurred, which aggravates the breathing effect.

- intraQpDelta: QP adjustment for intra frames.
- picQpDeltaRange: QP range of the single frame.
- smoothPsnrInGOP: Smooth the PSNR for frames in one GOP.
- chromaQPOffset: Chroma QP index offset.

6.2.1.4 Framerate Adjustment

There are two limitations of setting framerate for multi-channel video stream.

- The framerate of the first open video channel must be the maximum framerate.
- When video streaming on, if the maximum framerate is adjusted to new value, the framerate of other channel will be scaled at the same time. For example, if the maximum framerate of channel 0 is 30, and the framerate of channel 2 is 15. Then the framerate of

channel 1 is adjusted to 20, the framerate of channel is adjusted to 10. So the framerate adjustment is used to adjust the maximum framerate and scale other framerate.

6.2.1.5 Video Auto Rate Control Mechanism

- Ameba pro2 provide video auto rate control mechanism for video stream by setting for parameters which are as bellow.

```
typedef struct rate_control {
  uint32_t     sampling_time;
  uint32_t     maximun_bitrate;
  uint32_t     minimum_bitrate;
  uint32_t     target_bitrate;
} rate_ctrl_t;
```

- sampling_time: It is the unit of the video rate control, and It's based on the setting of the GOP of the video channel which we want to do the video auto rate control because I frame is the largest frame.
- maximun_bitrate: It is the threshold of the high rate control. If the bitrates is higher than maximun_bitrate, the system will do the video rate control automatically by dropping a half of frames and maintain the current bitrate until the low rate control is triggered.
- minimum_bitrate: It is the threshold of the low rate control. If the bitrates is lower than minimum_bitrate, the system will do the video rate control automatically by restoring the original framerate and maintain the current bitrate until the high rate control is triggered.
- target_birate: It is the expected bitrates of the user. If minimum_bitrate is not be set or higher than maximum_bitrate, the system will use target_bitrate for the threshold of the low rate control.
- How to enable the mechanism: Please reference the example, mmf2_video_example_v1_rate_control_init.c.

6.2.2 RTSP

```
typedef struct rtsp2_params_s {
 uint32_t      type;
 union {
        struct rtsp_video_param_s {
               uint32_t      codec_id;
               uint32_t      fps;
               uint32_t      bps;
               uint32_t      ts_flag;
               char*           sps;
               char*           pps;
               char*           lv;
        } v;
        struct rtsp_audio_param_s {
               uint32_t     codec_id;
               uint32_t     channel;
               uint32_t     samplerate;
        } a;
        struct rtsp_audio_opus_param_s {
               uint32_t     codec_id;
               uint32_t     channel;
               uint32_t     samplerate;
               uint32_t     max_average_bitrate;
               uint32_t     frame_size;
        } a_opus;
 } u;
} rtsp2_params_t;
```

Use **CMD_RTSP2_SELECT_STREAM** to select the RTSP stream index, currently support 0 and 1.
Use **CMD_RTSP2_SET_PARAMS** to set up the RTSP parameters.

- type: Media type, available Video (AVMEDIA_TYPE_VIDEO), Audio (AVMEDIA_TYPE_AUDIO).
- codec_id: RTSP supported codec ID, available AV_CODEC_ID_MJPEG, AV_CODEC_ID_H264, AV_CODEC_ID_PCMU, AV_CODEC_ID_PCMA, AV_CODEC_ID_MP4A_LATM, AV_CODEC_ID_MP4V_ES, AV_CODEC_ID_H265, AV_CODEC_ID_OPUS, AV_CODEC_ID_RGB888.
- fps: Video frame rate.
- bps: Bit per second
- ts_flag: H264 rtsp time sync enable switch.

- sps,pps,lv: Set sps, pps and profile level of H264.
- channel: Audio channel.
- samplerate: Audio samplerate.
- max_average_bitrate: Set the max_average_bitrate for OPUS rtsp.
- frame_size: Set the using OPUS encode frame size (the unit is msec) which will be related to the timestamp increase of opus rtp packet.

Current codec table:

```
static const struct codec_info av_codec_tables[] = {
 {AV_CODEC_ID_MJPEG, "MJPEG", RTP_PT_JPEG, 90000, 0, 0},
 {AV_CODEC_ID_H264, "H264", RTP_PT_DYN_BASE, 90000, 0, 0},
 {AV_CODEC_ID_PCMU, "PCMU", RTP_PT_PCMU, 8000, 1, 0},
 {AV_CODEC_ID_PCMA, "PCMA", RTP_PT_PCMA, 8000, 1, 0},
 {AV_CODEC_ID_MP4A_LATM, "MP4A", RTP_PT_DYN_BASE, 8000, 2, 0},
 {AV_CODEC_ID_MP4V_ES, "MP4V", RTP_PT_DYN_BASE, 90000, 0, 0},
 {AV_CODEC_ID_H265, "H265", RTP_PT_DYN_BASE, 90000, 0, 0},
 {AV_CODEC_ID_OPUS, "opus", RTP_PT_DYN_BASE, 48000, 2, 0}
};
```

6.2.3 AAC Encoder (AAC)

```
typedef struct aac_param_s {
        uint32_t     sample_rate;       // 8000
        uint32_t     channel;                    // 1
        uint32_t     bit_length;        // FAAC_INPUT_16BIT
        uint32_t     output_format;           // 0: Raw  1: ADTS
        uint32_t     mpeg_version;      // 0: MPEG4  1: MPEG2

        uint32_t     mem_total_size;
        uint32_t     mem_block_size;
        uint32_t     mem_frame_size;

        int          samples_input;
        int          max_bytes_output;
 //...
} aac_params_t;
```

Use **CMD_AAC_SET_PARAMS** to set up the AAC parameters.

- sample_rate: Sample rate for AAC encoder must be the same as the Audio codec setting. For instance, if using ASR_8KHZ as the Audio codec sample rate, the sample rate of AAC must be configured to 8000 or the codec result will be unexpected.
- channel: Set the audio channel number. The mono is set as 1, while the stereo is set as 2. This setting is related to the Audio codec.
- bit_length: The bit length use in AAC encoder. The bit length configuration must be identical to the Audio codec, like if audio codec word length is equal to WL_16BIT, which must be set to FAAC_INPUT_16BIT.
- output_format: The AAC output format, the default setting is 1 (ADTS).
- mpeg_version: Setting MPEG version, the default setting is 0 (MPEG4).
- mem_total_size: Memory pool size of AAC encoder output.
- mem_block_size: Block size used by Memory pool.
- mem_frame_size: Set maximum FRAME SIZE capacity.
- samples_input: It will be automatically configured when AAC initialization, no need to do setting.
- max_bytes_output: It will be automatically configured when AAC initialization, no need to do setting.

6.2.4 AAC Decoder (AAD)

```
typedef struct aad_param_s {
        uint32_t      type;             // TYPE_RTP_RAW or TYPE_ADTS
        uint32_t      sample_rate;      // 8000
        uint32_t      channel;          // 1
} aad_params_t;
```

Use **CMD_AAD_SET_PARAMS** to set up the AAD parameters.

- type: TYPE_ADTS is used when the source is AAC encoder, TYPE_RTP_RAW is used when source is RTP, and TYPE_TS is not currently supported.
- sample_rate: Need to match source sample rate to decode correctly.

- channel: Need to match source channel to decode correctly.

6.2.5 Audio Codec

The ASP algorithms, AGC (Automatic gain control), ANS (Adaptive noise suppression), AEC (Acoustic echo cancellation) and VAD (Voice Activity Detection), are included in this module.

```
typedef struct audio_param_s {
        audio_sr            sample_rate;  // ASR_8KHZ
        audio_wl            word_length;  // WL_16BIT
        audio_mic_gain      mic_gain;           // MIC_40DB
        audio_dmic_gain     dmic_l_gain;        // DMIC_BOOST_24DB
        audio_dmic_gain     dmic_r_gain;        // DMIC_BOOST_24DB
        int                 channel;            // 1
        int                 mix_mode;           // 0
        uint8_t             use_mic_type;       // 0: AMIC 1: DMIC
        int                 mic bias;           // 0:0.9 1:0.86 2:0.75
        int                 hpf_set;            // 0~7
        eq_cof_t            mic_l_eq[5];
        eq cof t            mic r eq[5];
        eq_cof_t            spk_l_eq[5];
        int                 ADC gain;
        int                 DAC_gain;
        int                 enable_record;
        uint8 t             fcs avsync en;
        uint32_t            fcs_avsync_vtime;
STEREO_DMIC
} audio_params_t;
```

Use **CMD_AUDIO_SET_PARAMS** to set up the audio parameters.

- sample_rate: Currently support 8K (ASR_8KHZ), 16K, 32K, 44.1K (ASR_44p1KHZ), 48K, 88.2K, 96K HZ.
- word_length: Currently support 16 bits (WL_16BIT), 24 bits (WL_24BIT).
- mic_gain: Analog microphone gain value. Support 0, 20, 30, 40 DB.
- dmic_l_gain: Left digital gain value. Support 0, 12, 24, 36 DB.
- dmic_r_gain: Right digital gain value. Support 0, 12, 24, 36 DB.
- channel: The number of channel is supported. Currently, support mono so set it to 1.
- use_mic_type: set the mic type, 0 is the analog microphone, 1 is the left digital mic, 2 is the right digital mic and 3 is the stereo digital mic.
- mic_bias: set the amic bias, the default value is 0.
- hpf_set: set the hpf level in mic path.
- mic_l_eq[5]: five band eq filters for setting in left mic path (amic path).
- mic_r_eq[5]: five band eq filters for setting in right mic path.
- spk_l_eq[5]: five band eq filters for setting in speaker path.
- ADC_gain: set the dgain for mic path. Support -17.625dB (0x00) ~ 30dB (0x7F).
- DAC_gain: set the dgain for speaker path. -65.625dB (0x00) ~ 0dB (0xAF).
- enable_record: enable the audio recording or not. If enabling, it will execute the function set by CMD_AUDIO_SET_MIC_RECORD_FUN.
- fcs_avsync_en: enable the function of inserting the audio dummy packets, if enabled, module audio will insert the dummy packet regarding to fcs_avsync_vtime and the first audio frame time into the output queue.
- fcs_avsync_vtime: setting the timestamp for the reference of the audio dummy packets.

NOTE

For the fcs_avsync_en *and fcs_avsync_vtime, please make sure the audio have enough length of the queue to keep the dummy packets. For the audio signal processing setting, please see the more ASP setting detail in 15.2 Open ASP algorithm*

6.2.6 RTP Input

```
typedef struct rtp_param_s {
        uint32_t      valid_pt;
        uint32_t      port;
        uint32_t      frame_size;
        uint32_t      cache_depth;
} rtp_params_t;
```

Use **CMD_AUDIO_SET_PARAMS** to set up the audio parameters.

- valid_pt: Processable RTP payload types. Set 0xFFFFFFFF to handle RTP_PT_PCMU (0), RTP_PT_PCMA (8) and RTP_PT_DYN_BASE (dynamic, default setting 96).
- port: The port to receive the RTP packet.
- frame_size: Maximum RTP packet size.
- cache_depth: The number of caches for RTP packets. The cache handler will send the RTP packet in the cache to the output of the module when the number of packets in the cache >= 50% cache depth.

6.2.7 G711 Codec

G711 Encode and G711 Decode use the same parameter structure.

```
typedef struct g711_param_s {
        uint32_t      codec_id;           // AV_CODEC_ID_PCMA or AV_CODEC_ID_PCMU
        uint32_t      buf_len;           // output buffer length
        uint32_t      mode;       // decode or encode
} g711_params_t;
```

Use **CMD_G711_SET_PARAMS** to set up the G711 parameters.

- codec_id: Set the codec type for G711 encoder/decoder. G711 currently supports PCMU (AV_CODEC_ID_PCMA) and PCMA (AV_CODEC_ID_PCMU) codec modes.
- buf_len: Determine the length (byte) of the encode buffer.
- mode: Determine whether the G711 codec module is an encoder (G711_ENCODE) or decoder (G711_DECODE).

6.2.8 OPUS Encoder (OPUSC)

```
typedef struct opusc_param_s {
        uint32_t      sample_rate;             // 8000
        uint32_t      channel;                                 // 1
        uint32_t      bit_length;              // 16
        uint32_t      complexity;
        uint32_t      use_framesize;

 //VBR CBR setting
        uint32_t      bitrate;                                 //default 25000
        uint32_t      enable_vbr;
        uint32_t      vbr_constraint;
        uint32_t      packetLossPercentage;

        uint32_t      opus_application;

        int               samples_input;
        int               max_bytes_output;

} opusc_params_t;
```

Use **CMD_OPUSC_SET_PARAMS** to set up the OPUSC parameters.

- sample_rate: Sample rate for OPUS encoder must be the same as the Audio codec setting. For instance, if using ASR_8KHZ as the Audio codec sample rate, the sample rate of OPUS must be configured to 8000 or the codec result will be unexpected.
- channel: Set the audio channel number. The mono is set as 1, while the stereo is set as 2. This setting is related to the Audio codec.
- bit_length: The bit length use in OPUS encoder. The bit length configuration must be identical to the Audio codec, like if audio codec word length is equal to WL_16BIT, which must be set to 16.
- complexity: Set the opus encoder's complexity, and the value is from 0 (low complexity) to 10 (high complexity). The higher complexity is configured the better quality encoding at a given bitrate but it also means more CPU consumption.
- use_framesize: The frame size contains in one OPUS packet. Since it will be related to the opus rtsp timestamp, if using RTSP, this must be the same as frame_size in rtsp module. Recommend to be the same or larger than AUDIO_DMA_PAGE_SIZE/(sample_rate / 1000)/2 but less than 60.
- bitrate: Set the bit rate for the opus encoder, the default value is 25000.
- enable_vbr: Enable VBR (variable bit rate) of the opus encoder.
- vbr_constraint: Makes constrained VBR if setting as 1.
- packetLossPercentage: Set the percentage of packet loss, the default value is 0.

- opus_application: Set the opus application type, broadcast/high-fidelity application (OPUS_APPLICATION_AUDIO), VoIP/videoconference applications (OPUS_APPLICATION_VOIP) and lowest-achievable latency (OPUS_APPLICATION_RESTRICTED_LOWDELAY). The default setting is OPUS_APPLICATION_AUDIO.
- samples_input: Not need to be set, it will be automatically set in the process of opus encoder.
- max_bytes_output: Not need to be set, it will be automatically set in the process of opus encoder.

6.2.9 OPUS Decoder (OPUSD)

```
typedef struct opusd_param_s {
        uint32_t      sample_rate;              // 8000
        uint32_t      channel;                    // 1
        uint32_t      bit_length;                // 16
        uint32_t      frame_size_in_msec;
        uint32_t      opus_application;
        uint8_t        with_opus_enc;

        int                samples_input;
        int                max_bytes_output;
} opusd_params_t;
```

Use **CMD_OPUSD_SET_PARAMS** to set up the OPUSD parameters.

- sample_rate: The sample of the opus packet will be decoded, must be the same as the audio codec.
- channel: Need to match source channel to decode correctly.
- bit_length: The audio bit length will be decoded, suggest to set as 16.
- frame_size_in_msec: No need to be set, it will be automatically set when using it.
- opus_application: Set the opus application type, broadcast/high-fidelity application (OPUS_APPLICATION_AUDIO), VoIP/videoconference applications (OPUS_APPLICATION_VOIP) and lowest-achievable latency (OPUS_APPLICATION_RESTRICTED_LOWDELAY). The default setting is OPUS_APPLICATION_AUDIO.
- with_opus_enc: Set to 1, if the application with opus encoder.
- samples_input: Not need to be set, it will be automatically set in the process of opus decoder.
- max_bytes_output: Not need to be set, it will be automatically set in the process of opus decoder.

6.2.10 MP4

```
typedef struct mp4_param_s {
        uint32_t      width;
        uint32_t      height;
        uint32_t      fps;
        uint32_t      gop;

        uint32_t      sample_rate;
        uint32_t      channel;

        uint32_t      record_length;
        uint32_t      record_type;
        uint32_t      record_file_num;
        char          record_file_name[32];
        uint32_t      fatfs_buf_size;
        uint32_t      mp4_user_callback;
} mp4_params_t
```

Use **CMD_MP4_SET_PARAMS** to set up the MP4 parameters.

- width: Set the max video frame width.
- height: Set the max video frame height.
- fps: Set the frame number per second.
- gop: Set the group of the picture which can be seemed as the cycle that I frame will update.
- sample_rate: The audio sample rate.
- channel: The audio channel number.
- record_length: Set the record file length in second.
- record_type: Set the record media type, STORAGE_ALL (with bot audio and video), STORAGE_VIDEO (video only), STORAGE_AUDIO (audio only).
- record_file_num: Set the number of file that will be recorded.

- record_file_name: Set the record file name.
- fatfs_buf_size: FATFS cache buffer size.
- mp4_user_callback: Configure the user callback function. If enable this, be sure that callback function for open (CMD_MP4_SET_OPEN_CB), write (CMD_MP4_SET_WRITE_CB), seek (CMD_MP4_SET_SEEK_CB) and close (CMD_MP4_SET_CLOSE_CB) have been set.

6.2.11 I2S

```
typedef struct i2s_param_s {
        int                 sample_rate;              // SR_32KHZ
        int                 out_sample_rate;                // SR_8KHZ
        int                 word_length;              // WL_24b
        int                 out_word_length;           // WL_16b
        audio_mic_gain       mic_gain;               // MIC_40DB
        int                 channel;                          // 1
        int                 out channel;
        int                 enable_aec;                    // 0
        int                 mix_mode;                     // 0
} i2s_params_t;
```

Use **CMD_I2S_SET_PARAMS** to set up the I2S parameters.

- sample_rate: Currently support 8K, 16K, 32K, 44.1K, 48K, 88.2K, 96K (, 12K, 24K, 64K 192K, 384K, 7.35K, 11.025K, 14.7K, 22.05K, 58.8K, 176.4K) HZ
- out_sample_rate: Currently supported sampling rate is the same as the sample rate, but less than or equal to sample_rate.
- word_length: 16 (WL_16b), 24 (WL_24b), 32 (WL_32b) bits.
- out_word_length: Currently supported bit depth is the same as the word_length, but less than or equal to word_length.
- mic_gain: Microphone gain value. Support 0, 20, 30, 40 DB.
- channel: Currently supports stereo or mono, please set to 2 or 1, and also supports 5.1 channels (but only support tx).
- out_channel: Currently supported channel is the same as the channel, but less than or equal to channel.
- enable_aec: The switch of enabling AEC.
- mix_mode: The switch of enabling mix mode.

6.2.12 Httpfs

The httpfs module to construct a HTTP File Server and send the media file on it.

```
typedef struct httpfs_param_s {
        char             fileext[4];
        char             filedir[32];
        char             request_string[128];
        uint32_t     fatfs_buf_size;
} httpfs_params_t;
```

Use **CMD_HTTPFS_SET_PARAMS** to set up the HTTPFS parameters.

- fileext: Set the file extension, for example “mp4”.
- filedir: Directory where the file is located, for example “VIDEO”.
- request_string: The string of http page, for example "/video_get.mp4".
- fatfs_buf_size: Buffer size of read file.

6.2.13 Array

The array module is use to play the small size and predefinition media streaming (like doorbell ring). It can be seemed as a source module.

```
typedef struct array_param_s {
        uint32_t     type;
        uint32_t     codec_id;
        uint8_t       mode;
        union {
                struct array_video_param_s {
                        uint32_t     fps;
                        uint8_t       h264 nal size;
                } v;
                struct array audio param s {
                        uint32_t     channel;
                        uint32_t     samplerate;
                        uint32_t     sample bit length;
                        uint32_t     frame_size;
```

```
                } a;
        } u;
} array_params_t;

typedef struct array_s {
        uint32_t     data_addr;
        uint32_t     data_len;
        uint32_t     data_offset;
} array_t;
```

Use the command **CMD_ARRAY_SET_PARAMS** to set up the parameters for the array module.

- type: Media type, available Video (AVMEDIA_TYPE_VIDEO), Audio (AVMEDIA_TYPE_AUDIO).
- codec_id: Set the codec ID of the array, like AV_CODEC_ID_MJPEG, AV_CODEC_ID_H264, AV_CODEC_ID_PCMU, AV_CODEC_ID_PCMA, AV_CODEC_ID_MP4A_LATM, AV_CODEC_ID_MP4V_ES, AV_CODEC_ID_H265, AV_CODEC_ID_OPUS, AV_CODEC_ID_RGB888.
- mode: set the array play mode, once (ARRAY_MODE_ONCE) or repeat (ARRAY_MODE_LOOP).
- h264_nal_size: Set the NALU length of h264 or h265 media array.
- channel: Set the audio channel.
- samplerate: Set the audio sample rate.
- sample_bit_length: bit length for one audio sample.
- frame_size: Set the using audio frame size (the unit is samples).

Use the command **CMD_ARRAY_SET_ARRAY** to set up the array input.

- data_addr: Set the media array store address.
- data_len: Set the media array total size.
- data_offset: Set the offset that will be started to play and it will also be used to keep the play location while the array module process.

6.3 Using the MMF example

Describe how to use the sample program to construct the applicational data stream.
In this section, there will be an introduction to correctly select the mmfv2 sample program and adjust the parameters.

6.3.1 Selecting and setting up sample program

For audio only samples, they are in function example_mmf2_audio_only while video joined samples are listed in example_mmf2_video_surport. Pick the example want to open before using it, remove the comment, and recompile. Opening more than two examples at the same time will result in unpredictable program execution results.

6.3.1.1 Requisites and Setup

Pre-requisites:

- Ameba Pro2 board
- Camera sensor board
- Micro USB cable
- WIFI (for transferring rtsp stream)
- MicroSD card (for saving the mp4 data)

Hardware setup:

- Connect the camera sensor board to the AmebaPro2's camera sensor board slot (CON1).
- Connect the PC with the AmebaPro2 CON8 port by the Micro USB cable.
- Insert the MicroSD card to the AmebaPro2's SD card slot.

Software setup:

- In project\realtek_amebapro2_v0_example\inc\platform_opts.h select the usage sensor.
- For audio only example, use "cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DEXAMPLE=media_framework" to build up the project.
- For video joined example, use "cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DVIDEO_EXAMPLE=on" to build up the project.
- Uncomment the example you want to execute.

The sample program is located at:
Audio only: \component\example\media_framework\ example_media_framework.c
Video joined: \project\realtek_amebapro2_v0_example\src\mmfv2_video_example\ video_example_media_framework.c

For example: open mmf2_video_example_joint_test_rtsp_mp4_init

```
// Joint test RTSP MP4
// H264 -> RTSP (V1)
// H264 -> MP4 (V2)
// AUDIO -> AAC  -> RTSP and mp4
// RTP   -> AAD  -> AUDIO
//mmf2_video_example_joint_test_rtsp_mp4_init();
```

Uncomment the example want to execute

```
// Joint test RTSP MP4
// H264 -> RTSP (V1)
// H264 -> MP4 (V2)
// AUDIO -> AAC  -> RTSP and mp4
// RTP   -> AAD  -> AUDIO
mmf2_video_example_joint_test_rtsp_mp4_init();
```

NOTE

Uncomment two media examples in the same time may cause unexpected result.

- Compile and execute firmware. The compilation and execution can refer to the previous chapter.

6.3.1.2 Currently supported example

- Audio only examples:

Example	Description	Result
mmf2_example_a_init	audio -> AAC -> RTSP(A)	AmebaPro2's AAC sound stream over the network. The sound received by AmebaPro2 is encoded by AAC and then streamed through the network (rtsp).
mmf2_example_audioloop_init	PCM audio -> PCM audio, audio loopback	The sound received by AmebaPro2 can be broadcast from the 3.5 audio channel of AmebaPro2, and the PCM transmission is directly used in the procedure.
mmf2_example_g711loop_init	audio -> G711E -> G711D -> audio	The sound received by AmebaPro2 can be broadcast from the 3.5 audio channel of AmebaPro2. PCM is encoded by G711 and transmit, then decoded by G711 and playback.
mmf2_example_aacloop_init	audio -> AAC -> AAD -> audio	The sound received by AmebaPro2 can be broadcast from the 3.5 audio channel of AmebaPro2. PCM is encoded by AAC and transmit, then decoded by AAD and playback.
mmf2_example_rtp_aad_init	RTP -> AAD -> audio	Stream AAC sound over the network to AmebaPro2 for playback. Streaming audio is decoded by AAD and played through 3.5 audio jack.
mmf2_example_2way_audio_init	audio -> AAC -> RTSP RTP -> AAD -> audio	Stream AAC sound to AmebaPro2's audio jack via the network and transmit the sound received by AmebaPro2 over the network simultaneously.
mmf2_example_pcmu_array_rtsp_init	ARRAY (PCMU) -> RTSP (A)	Transmitting PCMU sound arrays within AmebaPro2 over the network.
mmf2_example_aac_array_rtsp_init	ARRAY (AAC) -> RTSP (A)	Transfer AAC sound arrays in AmebaPro2 over the network.
mmf2_example_opusloop_init	audio -> OPUSC -> OPUSD -> audio	The sound received by AmebaPro2 can be broadcast from the 3.5 audio channel of AmebaPro2. PCM is encoded by OPUS and transmit, then decoded by OPUS and playback.
mmf2_example_a_opus_init	Audio -> OPUSC -> RTSP(A)	AmebaPro2's OPUS sound stream over the network. The sound received by AmebaPro2 is encoded by OPUSC and then streamed through the network (rtsp).
mmf2_example_rtp_opusd_init	RTP -> OPUSD -> audio	Stream OPUSC sound over the network to AmebaPro2 for playback. Streaming audio is

		decoded by OPUSD and played through 3.5 audio jack.
mmf2_example_2way_audio_opus_init	audio -> OPUSC -> RTSP RTP -> OPUSD -> audio	Stream OPUS sound to AmebaPro2's audio jack via the network and transmit the sound received by AmebaPro2 over the network simultaneously.
mmf2_example_pcm_array_audio_init	Array (pcm) -> audio	Play the array pcm data through AmebaPro2

- Video only examples:

Example	Description	Result
mmf2_video_example_v1_init	CH1 Video -> H264/H265 -> RTSP	Transfer AmebaPro2's H264/HEVC video stream over the network. Video default format: 720P 30FPS.
mmf2_video_example_v2_init	CH2 Video -> H264/H265-> RTSP	Transfer AmebaPro2's H264/HEVC video stream over the network. Video default format: 1080P 30FPS.
mmf2_video_example_v3_init	CH3 Video -> JPEG -> RTSP	Transfer AmebaPro2's JPEG video stream over the network. Video default format: 1080P 30FPS.
mmf2_video_example_v1_shapshot_init	CH1 Video -> H264/H265-> RTSP + SNAPSHOT	Transfer AmebaPro2's H264/HEVC video stream over the network and snapshot (JPEG) while streaming.
mmf2_video_example_simo_init	1 Video (H264/H265) -> 2 RTSP (V1, V2)	Transmitting two H264/HEVC video streams from AmebaPro2 over the network, the source of the video is the same video stream. Video default format: 1080P 30FPS.
mmf2_video_example_array_rtsp_init	ARRAY (H264/H265) -> RTSP (V)	Transfer H264/HEVC stream array in AmebaPro2 over the network. Video default format: 25FPS.
mmf2_video_example_v1_param_change_init	CH1 Video -> H264/H265-> RTSP (parameter change)	Transfer AmebaPro2's H264/HEVC video over the network and support dynamic adjustment of video parameters. The parameters of dynamic adjustment are Resolution, Rate Control Mode, Bit Rate in order.
mmf2_video_example_h264_array_mp4_init	ARRAY (H264/H265) -> MP4 (SD card)	AmebaPro2 will record H264/HEVC stream array to the SD card for 30 second. Video default format: 25FPS.
mmf2_video_example_md_rtsp_init	CH1 Video -> H264/H265-> RTSP CH4 Video -> RGB -> MD	RTSP video stream over the network. MD detect motion and draw the motion region to RTSP channel.
mmf2_video_example_v12_adjust_framerate_init	CH1 Video -> H264/H265->RTSP-> adjust framerate CH2 Video -> H264/H265-> RTSP-> adjust framerate	Transfer AmebaPro2's H264/HEVC video stream over the network. Video default format: 1080P 30FPS, and then adjust framerate when streaming on. Transfer AmebaPro2's H264/HEVC video stream over the network. Video default format: 720P 15FPS, and then adjust framerate when streaming on.
mmf2_video_example_jpeg_external_init	EXTERNAL DATA -> JPEG	Use video HW encode any data (NV12, NV16...) to jpeg. The results will be saved to SD card as test_0001.jpg, test_0002.jpg...

- Video + Audio examples:

Example	Description	Result
mmf2_video_example_av_init	1 Video (H264/H265) and 1 Audio -> AAC -> RTSP	Transfer AmebaPro2's H264/HEVC video and AAC sound stream over the network. Video default format: 1080P 30FPS.
mmf2_video_example_av2_init	2 Video (H264/H265) and 1 Audio -> AAC -> 2 RTSP (V1+A, V2+A)	Transmitting two H264/HEVC videos and AAC audio streams from AmebaPro2 over the network. The source of the videos is different ISP channel. The videos formats are set to 1080P 30FPS (V1) and 720P 30FPS (V2) respectively.

mmf2_video_example_av21_init	1 Video (H264/H265) and 1 Audio -> 2 RTSP (V+A)	Transfer two copies of AmebaPro2's H264/HEVC video (1080P 30FPS) and AAC sound stream through the network, the video source is the same ISP channel.
mmf2_video_example_av_mp4_init	1 Video (H264/H265) and 1 Audio -> MP4 (SD card)	AmebaPro2 will record three videos (1080P 30FPS) to the SD card for 30 seconds each The default storage name is : AmebaPro2_recording_0.mp4 AmebaPro2_recording_1.mp4 AmebaPro2_recording_2.mp4
mmf2_video_example_av_rtsp_mp4_init	Video (H264/H265) -> RTSP and mp4 AUDIO -> AAC -> RTSP and MP4	(1) Transfer AmebaPro2's H264/HEVC video and AAC sound stream over the network. Video default format: 1080P 30FPS. (2) AmebaPro2 will record three videos (1080P 30FPS+AAC) to the SD card for 30 seconds each. The default storage name is : AmebaPro2_recording_0.mp4 AmebaPro2_recording_1.mp4 AmebaPro2_recording_2.mp4 (3) Streaming AAC sounds to AmebaPro2 via the network. Note: (1) video source of (2) is from the same ISP channel.
mmf2_video_example_joint_test_init	Video (H264/H265) -> RTSP (V1+A) Video (H264/H265) -> RTSP (V2+A) AUDIO -> AAC -> RTSP RTP -> AAD -> AUDIO	(1) Transmitting two H264/HEVC video streams from AmebaPro2 over the network, the source of the video is the different video stream. Video default format: 1080P 30FPS (V1) and 720P 30FPS (V2). (2) Streaming two copies of AAC sounds to AmebaPro2 via the network.
mmf2_video_example_joint_test_rtsp_mp4_init	Video (H264/H265) -> MP4 (V1+A) Video (H264/H265) -> RTSP (V2+A) AUDIO -> AAC -> RTSP and MP4 RTP -> AAD -> AUDIO	(1) Transfer AmebaPro2's H264/HEVC video and AAC sound stream over the network. Video default format: 1080P 30FPS. (2) AmebaPro2 will record three videos (720P 30FPS+AAC) to the SD card for 30 seconds each. The default storage name is : AmebaPro2_recording_0.mp4 AmebaPro2_recording_1.mp4 AmebaPro2_recording_2.mp4 (3) Streaming AAC sounds to AmebaPro2 via the network. (4) RTP send the audio stream from network to AmebaPro2 and the stream is decoded by AAD and played through 3.5 audio jack. Note: (1) video source of (2) is from different ISP channels.
mmf2_video_example_2way_audio_pcmu_doorbell_init	Video (H264/H265) -> RTSP (V1) AUDIO -> G711E -> RTSP RTP -> G711D -> AUDIO ARRAY (PCMU) -> G711D -> AUDIO (doorbell)	(1) Transmitting AmebaPro2's H264/HEVC stream and PCMU sound stream over the network. Video default format: 1080P 30FPS. (2) PCMU sound can be streamed to AmebaPro2 via the Internet and playback. (3) Play PCMU sound array in AmebaPro2 (default is the doorbell).
mmf2_video_example_2way_audio_pcmu_init	Video (H264/H265) -> RTSP (V1) AUDIO -> G711E -> RTSP RTP -> G711D -> AUDIO	(1) Transmitting AmebaPro2's H264/HEVC stream and PCMU sound stream over the network. Video default format: 1080P 30FPS. (2) PCMU sound can be streamed to AmebaPro2 via the Internet and playback.
mmf2_video_example_av_mp4_httpfs_init	1 Video (H264) 1 Audio -> MP4 (SD card) Http File Server	AmebaPro2 will record a video every 30 seconds and save it to the SD card (1080P 30FPS+AAC).

		The default is to record 60 files, and repeat the recording after the end. The default storage name is: mp4_record_0.mp4~mp4_record_29.mp4 Also open Http File Server for client to do playback.
mmf2_video_example_h264_pcmu_array_mp4_init	1 Video array and 1 Audio array(pcmu) -> MP4 (SD card)	Save 1 video stream and 1 pcmu audio stream to mp4 file (the record file may not play on some player)
mmf2_video_example_demuxer_rtsp_init	Demux a mp4 file in SD card (based on record file name) to 1 Video and 1 Audio -> RTSP	Demux a mp4 file (suggest to use a file created by Ameba Pro2) and send the video and audio data through rtsp

- Video + NN examples:

Example	Description	Result
mmf2_video_example_vipnn_rtsp_init	Video (H264/H265)-> RTSP (V1) Video (RGB) -> NN (V4)	(1) RTSP video stream over the network. (2) NN do object detection and draw the bounding box to RTSP channel. (Please see NN chapter for more details)
mmf2_video_example_md_nn_rtsp_init	Video (H264/H265) -> RTSP (V1) Video (RGB) -> MD (V4)-> NN	(1) RTSP video stream over the network. (2) MD module detect motion. If there is motion detected, it will trigger NN module to detect object and draw the bounding box to RTSP channel.
mmf2_video_example_vipnn_facedet_init	Video (H264/H265) -> RTSP (V1) Video (RGB) -> NN face detect (V4)	(1) RTSP video stream over the network. (2) NN do face detection then draw the bounding box and face landmark to RTSP channel. (Please see NN chapter for more details about how to load face detection NN model)
mmf2_video_example_face_rtsp_init	Video (H264/H265) -> RTSP (V1) Video (RGB) -> NN face detect (V4) -> NN face recognition	(1) RTSP video stream over the network. (2) NN do face detection and face recognition, and then draw the bounding box and face recognition result to RTSP channel. ((Please see NN chapter for more details about how to load face detection/recognition NN model)
mmf2_video_example_joint_test_all_nn_rtsp_init	Video (H264/H265) -> RTSP (V1) RGB -> NN object detect (V4) RGB -> NN face detect (V4) -> NN face recognition AUDIO -> NN audio classification	(1) RTSP video stream over the network. (2) NN do object detection, face detection and face recognition, and then draw the bounding box and face recognition result to RTSP channel. NN do audio classification. (Please see NN chapter for more details about how to load face detection/recognition NN model)
mmf2_video_example_joint_test_vipnn_rtsp_mp4_init	H264 -> MP4 (V1) Video (H264/H265) -> RTSP (V2) RGB -> NN object detect (V4) RGB -> NN face detect (V4) -> NN face recognition (optional) AUDIO -> AAC -> RTSP and mp4 RTP -> AAD -> AUDIO AUDIO -> NN audio classification	(1) RTSP video stream over the network. (2) Ameba Pro2 will record three videos (720P 30FPS+AAC) to the SD card for 30 seconds each. The default storage name is : Ameba Pro2_recording_0.mp4 Ameba Pro2_recording_1.mp4 Ameba Pro2_recording_2.mp4 (3) Streaming AAC sounds to Ameba Pro2 via the network. (4) RTP send the audio stream from network to Ameba Pro2 and the stream is decoded by AAD and played through 3.5 audio jack. (5) NN do object detection, face detection and face recognition, and then draw the bounding box and face recognition result to RTSP channel. NN do audio classification. (Please see NN chapter for more details about how to load face detection/recognition NN model)

mmf2_video_example_vipnn_facedet_sync_init	Video (H264/H265) -> RTSP (V1) Video (H264/H265) -> RTSP (V2, Sync mode) Video (RGB) -> NN face detect (V4)	(1) RTSP video stream over the network. (2) NN do face detection then draw the bounding box and face landmark to RTSP channel. (Please see NN chapter for more details about how to load face detection NN model)
mmf2_video_example_vipnn_facedet_sync_snapshot_init	CH1 Video -> JPEG (SNAPSHOT, Sync mode) Video (RGB) -> NN face detect (V4)	NN do face detection then draw the bounding box and face landmark to JPEG. The results will be saved to SD as test_0001.jpg, test_0002.jpg...

- Audio + NN examples:

Example	Description	Result
mmf2_video_example_audio_vipnn_init.c	AUDIO -> NN	The sound received by AmebaPro2 can be transmitted to NN engine to do sound classification. Please see NN chapter for more details

- Video + Audio + FCS:

Example	Description	Result
mmf2_video_example_joint_test_rtsp_mp4_init_fcs	The same as mmf2_video_example_joint_test_rtsp_mp4_init	Please see FCS chapter for more details

6.3.1.3 Execution and testing

Before executing example, it is necessary to set up console tool first (Tera Term, MobaXterm or PuTTY……) and configure serial port baud to 115200. Once the setting is completed, AmebaPro2 is also connected with the PC and booted to get the Log message output of AmebaPro2.

- For examples with rtsp stream, we must first set up AmebaPro2 to connect with the network. Use AT command below to do the connect with an AP device:

```
ATW0=<Name of WIFI SSID> => Set the WiFi AP SSID to be connected
ATW1=<Password>          => Set the WiFi AP password, if needed
ATWC                     => Initiate the connection
```

6.3.1.4 When the "RTSP stream enabled" message shown on console, it indicates that the RSTP server is already running. You can use VLC player to check the rtsp stream. For rtsp usage can refer to 6.3.3.

6.3.2 MMF AT command

MMF video examples provide commands for user to refer the audio reset and de-initialize the MMF and linker modules

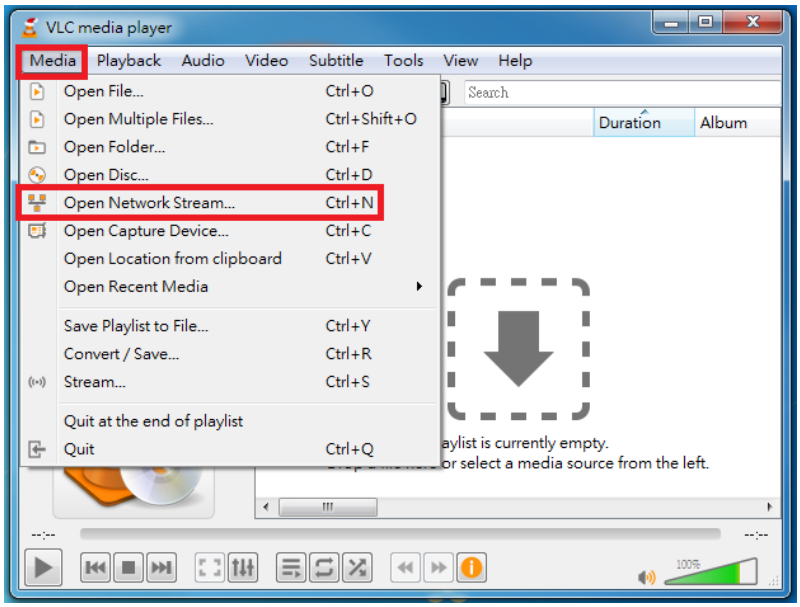
- UC=TD: use for de-initialize the whole flow of corresponding examples
- UC=TSR: reset the whole system

6.3.3 VLC media player settings

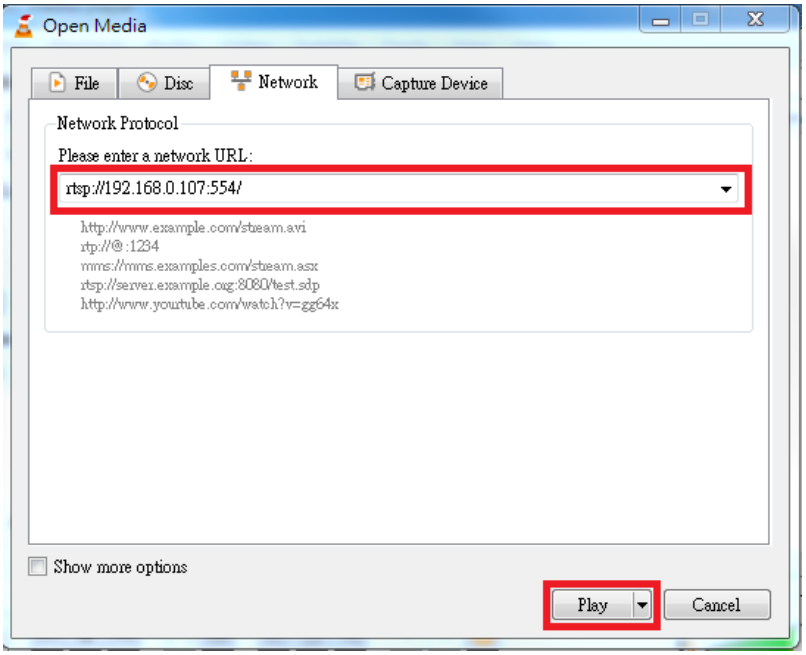
For RTSP examples, you can use VLC media player to receive or transmit the stream. Download VLC media player from website https://www.videolan.org/.

6.3.3.1 Stream audio/video from AmebaPro2 to VLC player

- Click “Media” -> “Open Network Stream”.

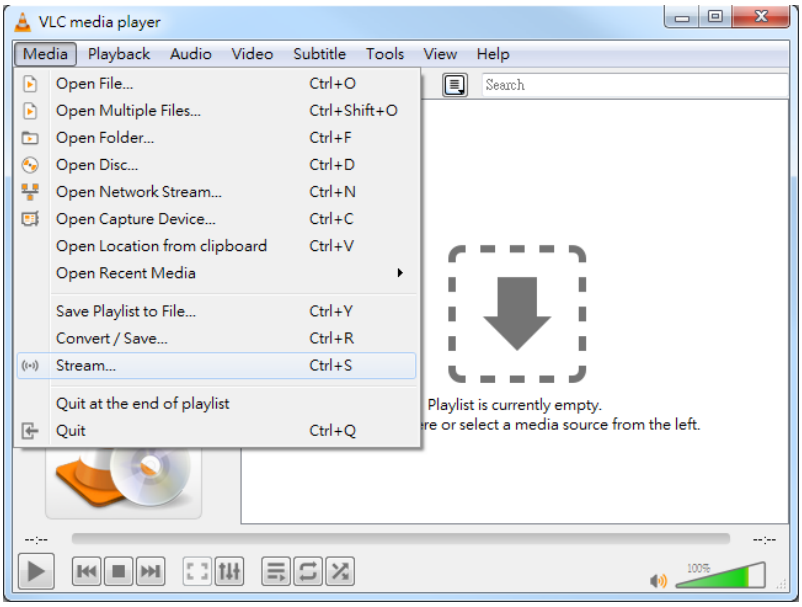


- Enter “rtsp://xxx.xxx.xxx.xxx:yyy/”, where xxx.xxx.xxx.xxx is the Ameba IP address and yyy is the RTSP server port (default is 554), and click “Play”.

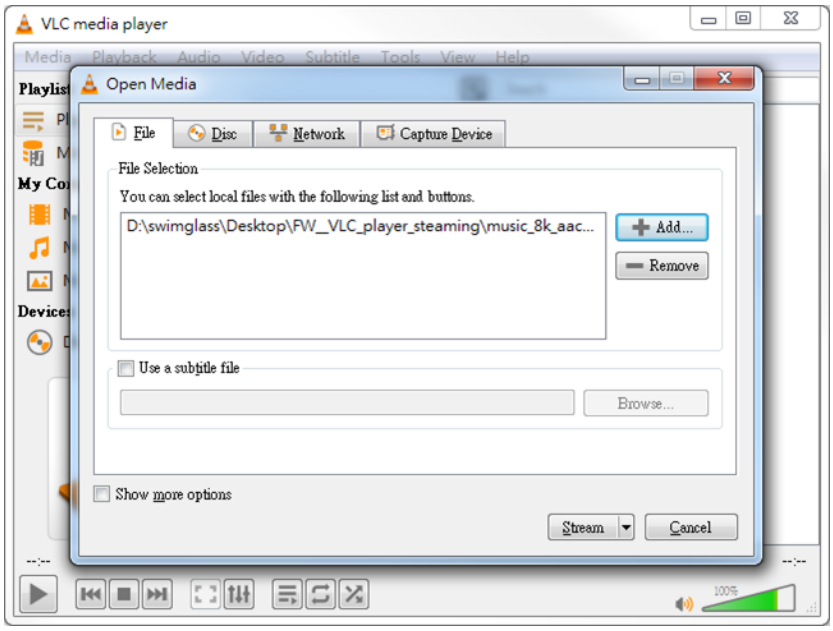


6.3.3.2 Stream audio from VLC player to AmebaPro2

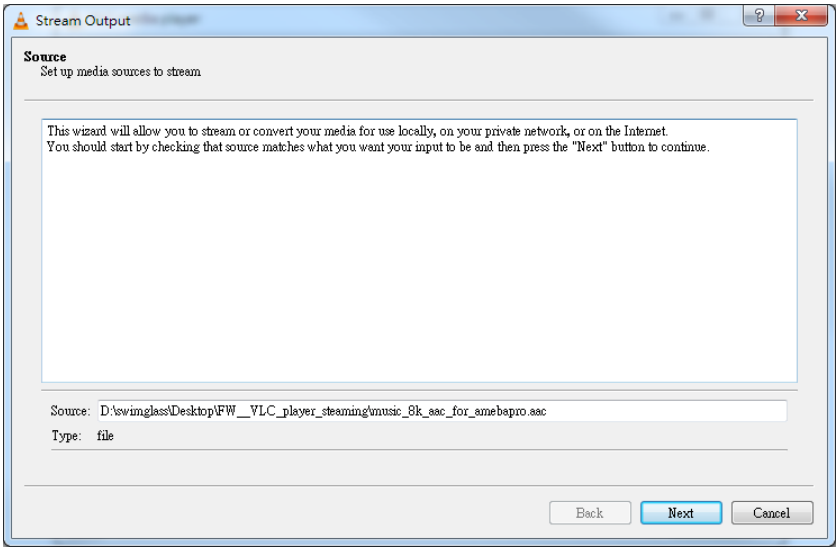
- Click “Media” -> “Stream”.



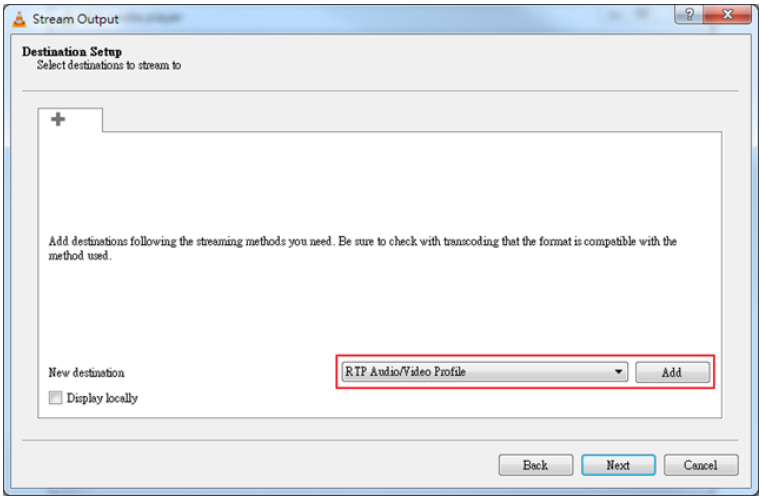
- Select “File”, choose the file by “Add” and finally click the “Stream”. (If the startup example is RTP -> AAD -> AUDIO please select the audio file with the file name .aac (The file format must be the same as the AAC decoder setting, the default is mono, sampling rate = 8k Hz). If the startup example is RTP -> G711D -> AUDIO, please select the audio file with the file extension .wav). If the startup example is RTP -> OPUSD -> AUDIO, please select the audio file with the file name .opus)



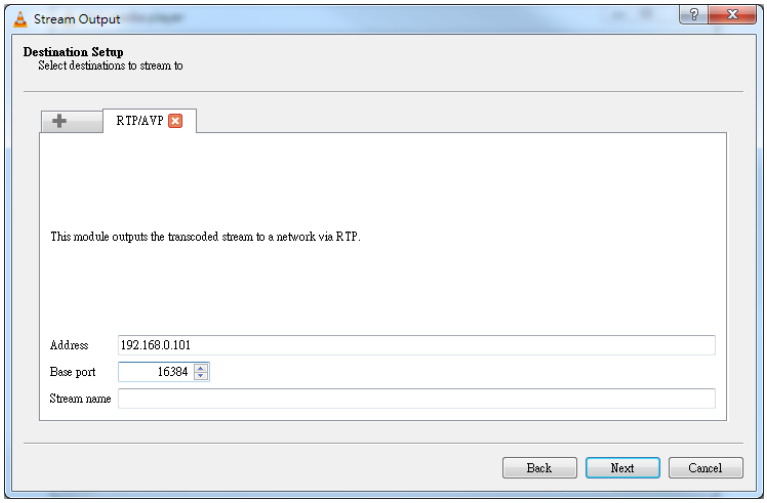
- You will see your select file after push “Stream”. Check it and click “Next”.



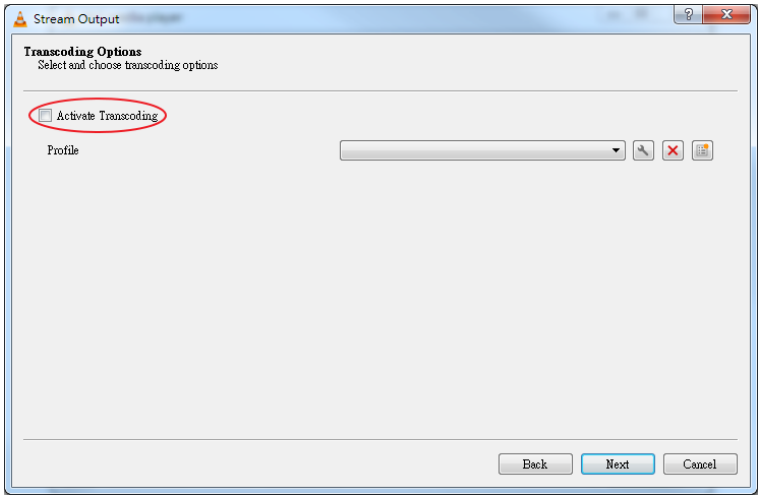
- Select “RTP Audio/Video Profile”, and click “Add”.

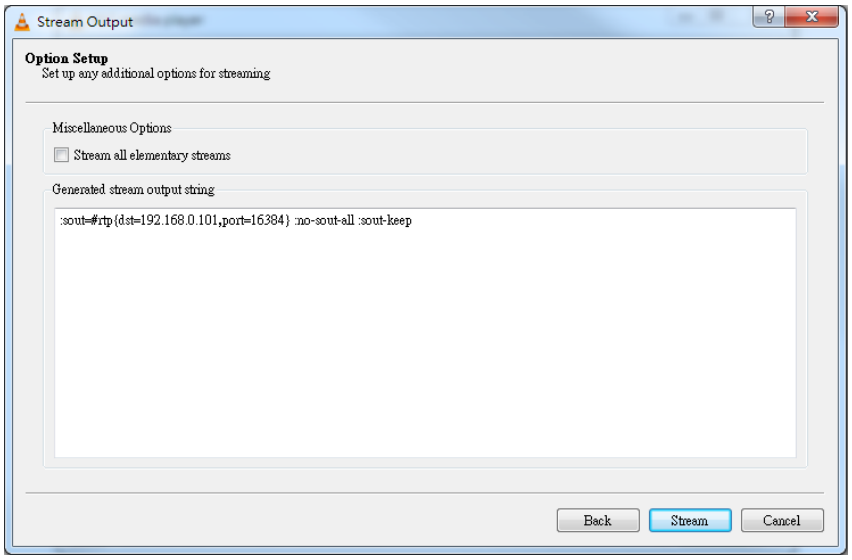


- Enter AmebaPro's IP Address in “Address” field, with “Base port” set to 16384, and click “Next”.



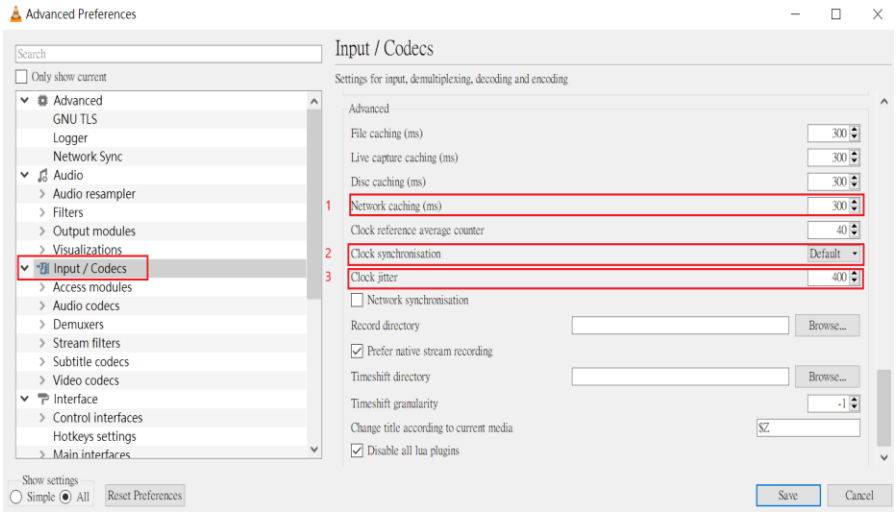
- Confirm “Activate Transcoding” is unchecked, and click “Next” -> “Stream”. Then the sound can be heard on AmebaPro2 3.5mm audio jack.



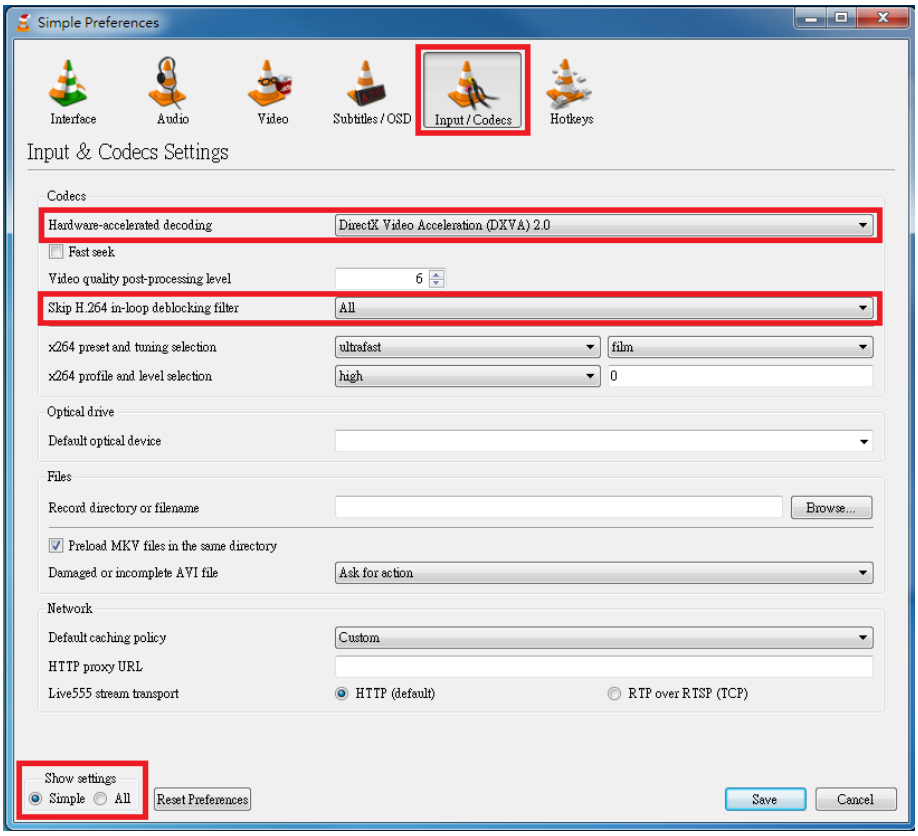


6.3.3.2.1 Adjust latency (buffer) related settings

- Click “Tools” -> “Preferences” -> “Show settings: All” (lower left corner) -> “Input/ Codecs”, (1) set “Network caching” to 300ms (recommended), (2)set “Clock synchronisation” to Default, (3) set “Clock jitter” to 400ms (recommended).



- Click “Tools” -> “Preferences” -> “Show settings: Simple” (lower left corner) -> “Input/Codecs”. Enable “Hardware-accelerated decoding” if available, and set “Skip H.264 in-loop deblocking filter” to “All”.



- VLC have a pts_delay buffer by "network buffer" and "clock jitter". The maximum value of this buffer is equal to "network buffer" plus "clock jitter". The video display on the VLC side will delay due to the increase of pts_delay buffer. By reducing the "network cache" and "clock jitter" can achieve the effect of shortening the delay.

6.3.4 Echo Cancellation

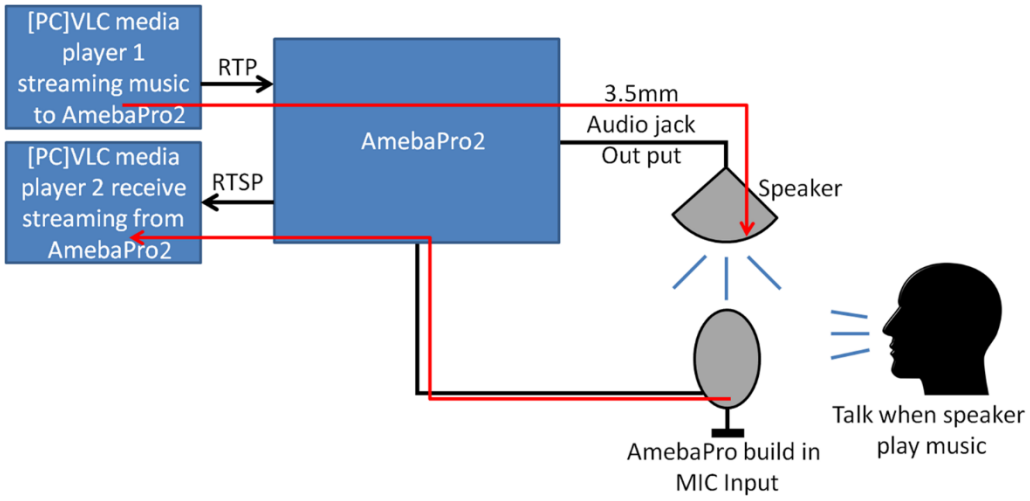
Echo cancellation is default provided in the audio part of MMFv2. To test whether the echo cancellation function is correct, use VLC media player to verify it on the computer.

Usage Note (Refer to Audio optimization chapter):

- Sample rate must be 8K Hz/16K Hz
- Frame size must be the multiplies of 10ms (suggest to be 10ms or 20ms about 160 samples and 320 samples)
- Two input signals must keep unchanged during AEC_process.
- Time for executing AEC_process must be under 10ms or 20ms (up on the frame size).
- Please check microphone and speaker signal and make sure there is no clipping signal.

The verification method is as follows:

- Use VLC media player on the PC to stream voice signal to AmebaPro2.
- Put Ameba Pro2 speaker next to AmebaPro2 built-in Mic and speak at the same time.
- Then pass the received sound to the VLC media player on the PC via AmebaPro2 to see if the sound in step 1 is small enough or even disappear.



6.4 AV SYNC

6.4.1 Timestamp

In all of the mmf modules in SDK, the timestamps is obtained by functions mm_read_mediatime_ms_fromisr() (using in isr) and mm_read_mediatime_ms (using in non-isr). If user want to sync the time with the mmf modules, please use these two functions to obtain the timestamps.

6.4.2 Sync audio and video first frames by dummy packets

In some situation, user may need to sync the first frames of the audio and video. In ameba Pro2 SDK, we provide a method by inserting audio dummy packets to sync the video first frame.

To sync the first frame, user need to open the video streaming first to obtain the video first frame information, then setting the audio parameters fcs_avsync_en to 1 and fcs_avsync_vtime to the video first timestamp. The audio module will insert the enough dummy packets to output queue till the audio output queue is full. In mmf2_video_example_joint_test_rtsp_mp4_init_fcs example, it will show the sample for how to add the audio dummy frames to sync the first video frame's timestamp of the fcs process.

The following code shows how to set up the audio dummy frames parameters.

```
//waiting until get the fcs first frame information
void fcs_avsync(bool enable)
{
 if (enable) {
        //get the fcs time need to what video first frame
        int fcs_video_starttime = 0;
        int fcs_video_endtime = 0;
        int audio_framesize_ms = 0;
        while (!fcs_video_starttime) {
                vTaskDelay(1);
                video_get_fcs_queue_info(&fcs_video_starttime, &fcs_video_endtime);
        }

        audio_params.fcs_avsync_en = 1;
        audio_params.fcs_avsync_vtime = fcs_video_starttime;
```

```
 }
}
```

Besides the parameters' setting, user also needs to modify the audio output queue length. The following code shows the sample about how to approximate the audio output queue length.

```
uint32_t audio_expected_queue = 800; //set 800 length as the maximum value
        uint32_t audio_apply_time = mm_read_mediatime_ms();
        uint32_t audio_frame_ms;
        mm_module_ctrl(audio_ctx, CMD_AUDIO_GET_FRAMESIZE_MS, (int)&audio_frame_ms);
        if (audio_frame_ms) {
               if (audio_expected_queue > (audio_apply_time -
audio_params.fcs_avsync_vtime) / audio_frame_ms + 30) {
                     audio_expected_queue = (audio_apply_time -
audio_params.fcs_avsync_vtime) / audio_frame_ms + 30;
               }
        }
        printf("audio length = %d\r\n", audio expected queue);
        mm_module_ctrl(audio_ctx, MM_CMD_SET_QUEUE_LEN, audio_expected_queue); //Add the
queue buffer to avoid to lost data.
```

7 Memory Layout

7.1 Programming Space

Start Address	Size	Cache Support	IP Function
0x0000_0000	32 KB	-	ITCM ROM
0x0001_0000	128 KB	-	ITCM SRAM

7.2 Data Space

Start Address	Size	Cache Support	IP Function
0x2000_0000	16 KB	-	DTCM ROM
0x2001_0000	80 KB	-	ITCM SRAM

7.3 Extension Memory Space

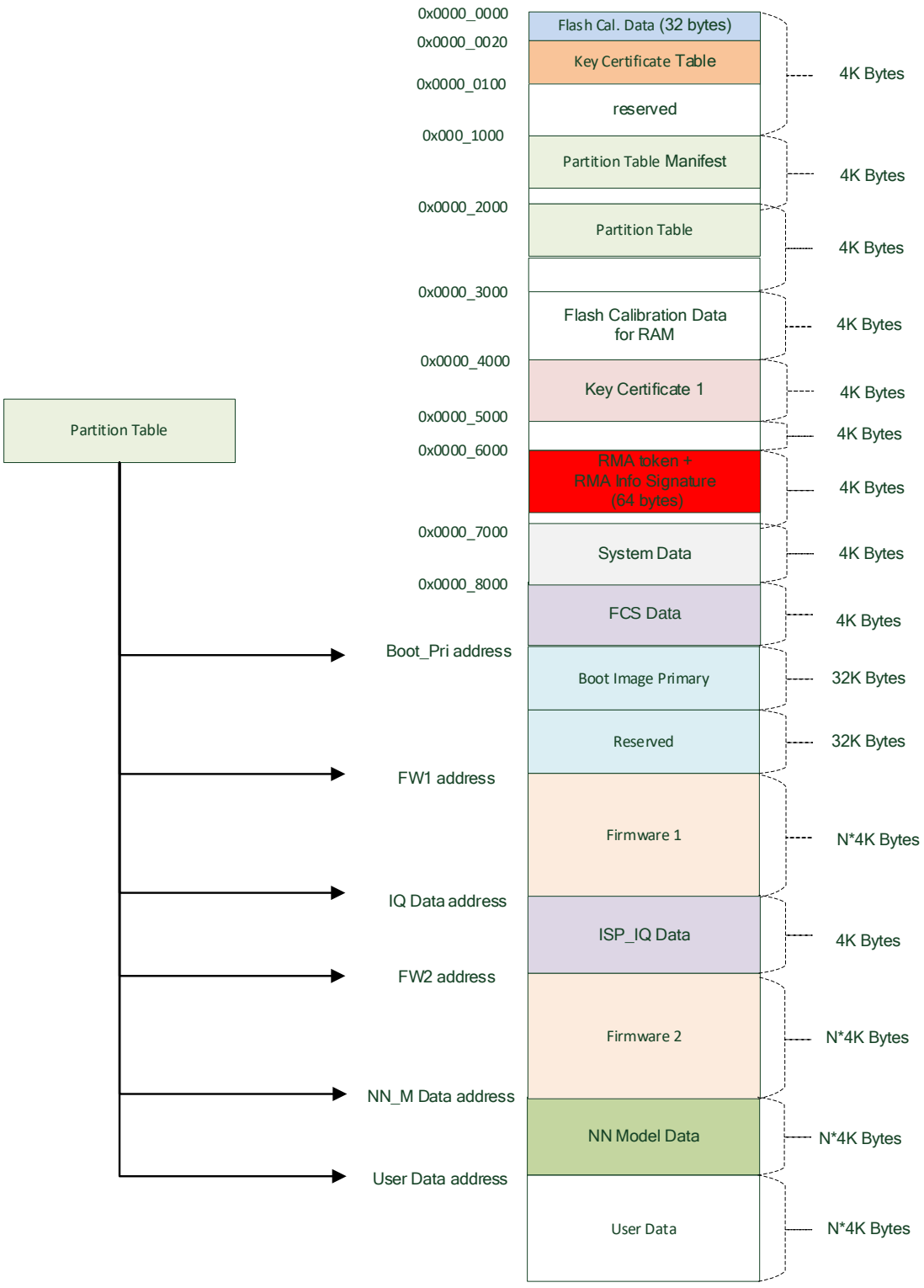
Name	Physical Address	Size	IP Function
Flash	0x0800_0000~0x0FFF_FFFF	128 MB	External flash memory
DRAM	0x0700_0000~0x7FFF_FFFF	128 MB	Extended DRAM memory

NOTE

- *The external flash memory address base is from 0x0800_0000 to 0x0FFF_FFFF*
- *The extended DRAM address base is from 0x7000_0000 to 0x7FFF_FFFF*
- *Both of flash and DRAM access are cacheable design*

8 Flash Layout

8.1 NOR Flash Layout overview



8.1.1 Blocks used in SDK

- File System

Place in the NOR flash:0xE00000
Content: The littlefs or the fatfs file system.

- System Data for BT and Wi-Fi SSID

Content: The following picture shows the flash layout of system data, signature and version are used to record the status of system data. The BT parameter size is 4bytes. The actual wifi-ssid(wifi fast connection data) is placed in 0xf00020-0x1000000.
Place in the NOR flash:

	0x00	**0x04**	**0x08**	**0x0C**
0xf00000	Signature(SYSd)	Version	RSVD	RSVD
0xf00010	BT Para Data	RSVD	RSVD	RSVD
0xf00020 ~ 0x1000000	Wifi fast connection data			

- Secure Storage

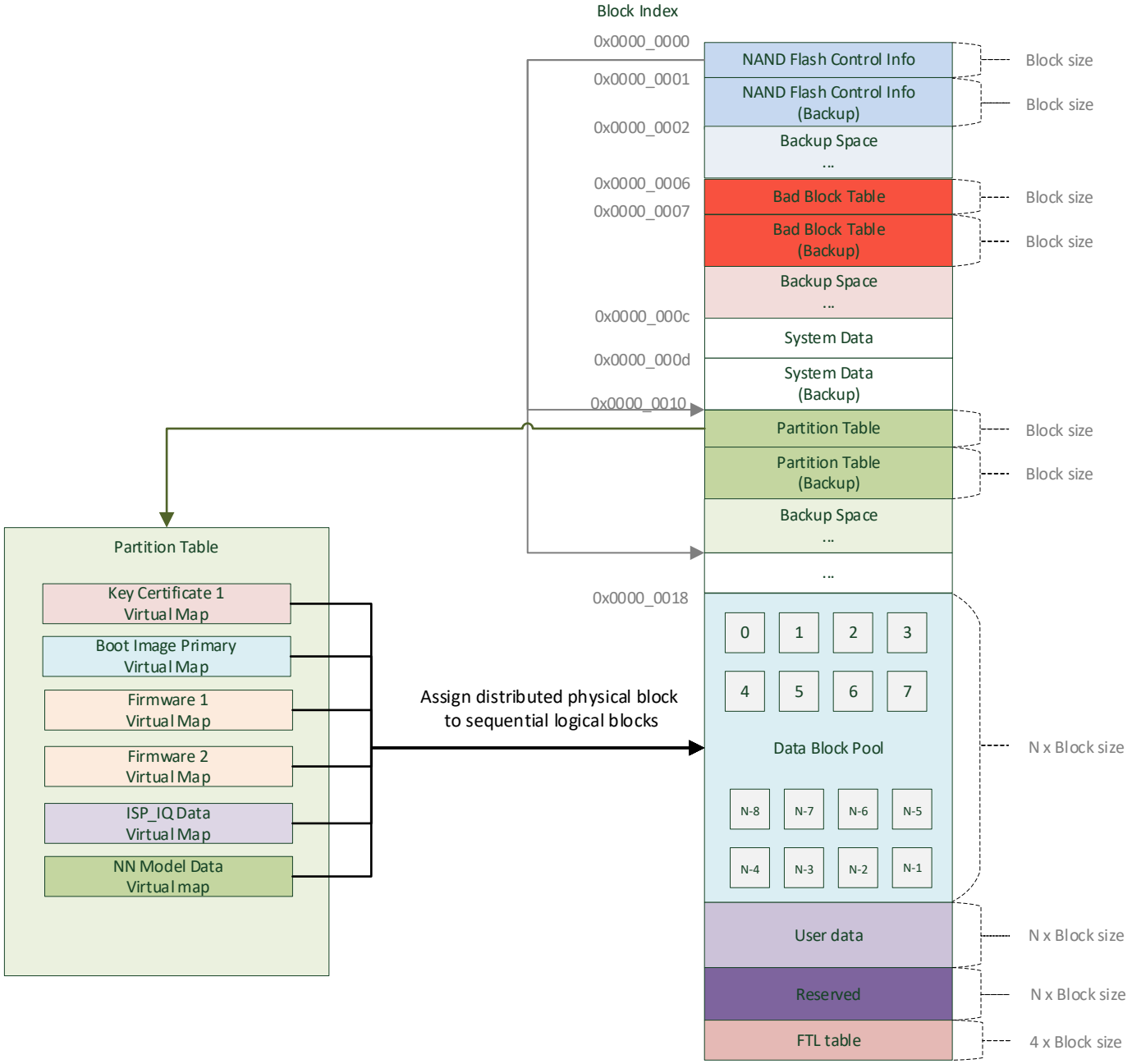
Content: The secure_storage example stores/loads an AES GCM encrypted user data in flash. The plaintext of user data is like the following structure. The structure can be defined based on the requirement of user data.

```
typedef struct user_data_s {
 char client_key[512];
} user_data_t;
```

Place in the NOR flash:
The flash address used by secure storage is defined in platform_opts.h and can be modified based on requirement.

```
#define SECURE_STORAGE_BASS        (0xF00000 + 0x4000) // 4KB
```

8.2 NAND Flash Layout overview



8.2.1 Blocks used in SDK

- File System

Content: The littlefs file system.
Place in the NAND flash: 0x4000000

- System Data for BT and Wi-Fi SSID

Content: The following picture shows the flash layout of system data, signature and version are used to record the status of system data. The BT parameter size is 4bytes. The actual wifi-ssid(wifi fast connection data) is placed in 0xf00020 – 0x1000000.
Place in the NAND flash:

	0x00	**0x04**	**0x08**	**0x0C**
0xf00000	Signature(SYSd)	Version	RSVD	RSVD

0xf00010	BT Para Data	RSVD	RSVD	RSVD
0xf00020 ~ 0x1000000	Wifi fast connection data			

- Secure Storage

Content: The secure_storage example stores/loads an AES GCM encrypted user data in flash. The plaintext of user data is like the following structure. The structure can be defined based on the requirement of user data.

```
typedef struct user_data_s {
 char client_key[512];
} user_data_t;
```

Place in the NAND flash:

The flash address used by secure storage is defined in platform_opts.h and can be modified based on requirement.

```
#define SECURE_STORAGE_BASS        (0x7A00000 + 0x4000) // 4KB
```

8.3 NAND Flash

The NAND flash 1G-bit memory array is organized into 65,536 programmable pages of 2,048-bytes each. The entire page can be programmed at one time using the data from the 2,048-Byte internal buffer. Pages can be erased in groups of 64 (128KB block erase). The NAND flash has 1,024 erasable blocks.

1Gb SLC NAND Flash: 1G-bit / 128M-byte

On chip ECC for memory array

Page Data Buffer (2048 Byte)	Spare (64 Byte)
Block (64 Pages, 64*2048 Byte)	64*64 Byte
Total flash 1024 Blocks (65536 Pages, 1024 Blocks * 64 Pages)	1024*64*64 Byte

Page Structure (2112 Byte)	**Page Data Buffer (2048 Byte) ECC Protected**				**Spare(64 Byte)**			
	Sector0	Sector1	Sector2	Sector3	Spare0	Spare1	Spare2	Spare3

The first byte of spare0 is bad block marker.

8.3.1 NAND Flash mbed API

NAND Flash mbed API is used to access Flash physical location and can be used in the bootloader.

Please refer to snand_api.h & snand_api.c

```
/**
  * @brief  Init Flash
  * @param  obj: address of the flash object
  * @retval   none
  */
void snand_init(snand_t *obj);

/**
  * @brief  Erase flash block, usually 1 block = 64K bytes
    Please refer to flash data sheet to confirm the actual block size.
    The actual address which being erased always aligned with block size.
  * @param  address: Specifies the starting address to be erased.
  * @retval SUCCESS, FAIL
  */
int snand_erase_block(snand_t *obj, uint32_t address);
```

```
/**
  * @brief  Read a stream of data from specified address vai user mode
  * @param  obj: Specifies the parameter of flash object.
  * @param  address: Specifies the address to be read.
  * @param  len: Specifies the length of the data to read.
  * @param  data: Specified the address to save the readback data.
  * @retval SUCCESS, FAIL
  */
int  snand_page_read(snand_t *obj, uint32_t address, uint32_t Length, uint8_t *data);

/**
* @brief  Write a stream of data to specified address
* @param  obj: Specifies the parameter of flash object.
* @param  address: Specifies the address to be programmed.
* @param  Length: Specifies the length of the data to write.
* @param  data: Specified the pointer of the data to be written.
          If the address is in the flash, full address is required, i.e. SPI_SNAND_BASE +
Offset
* @retval SUCCESS, FAIL

*/
int snand_page_write(snand_t *obj, uint32_t address, uint32_t Length, uint8_t *data);
```

8.3.1.1 NAND flash mbed example

This example demonstrates how use mbed API to scan bad block and read/write a NAND flash.

The example is located in:

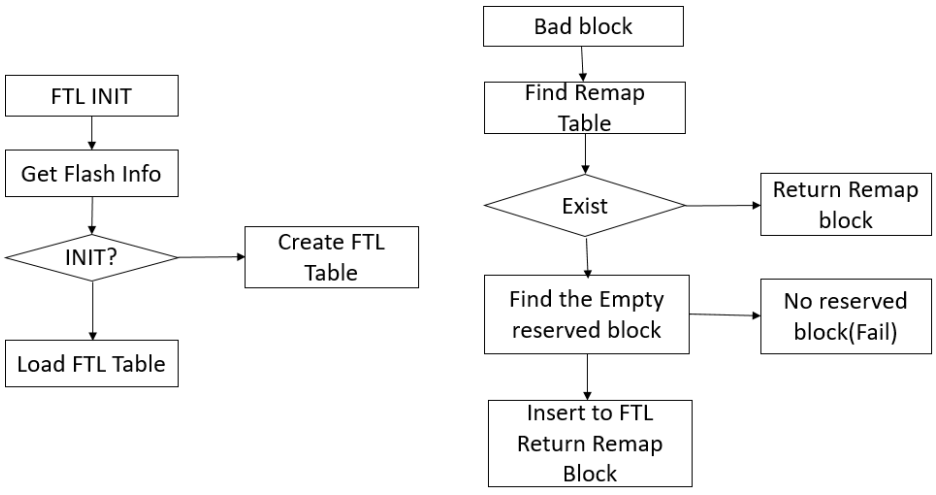
"\project\realtek_amebapro2_v0_example\example_sources\nand_flash\"

Copy main.c to src folder, compile project, and the download the binary.

8.3.2 NAND FTL

The flash translation layer (FTL) performs logical-to-physical address. It is block-mapping method; you do not need to deal with the bad block operation. It reserved 5% reserved blocks to replace the bad block. The detail procedure is as below.

FTL must be used after the OS init in main(), because it will be related to the boot speed.



FTL table

Start Tag		Bad block Number		Remap block Number		End Tag	
BB		Number		Number		bb	
0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF
0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF

Bad block is 800 and the remap block is 950 for the example

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
BB		800		950		bb	
0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF
0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF	0XFF

User data
Reserved block
FTL Bad block Remap Table

8.3.2.1 FTL API

- ftl_common_read – Read the data from flash.
- ftl_common_write – Write the data into flash
- ftl_common_erase – Erase the sector or block from flash.
- ftl_erase_sector – Erase the 4k sector from flash.
- ftl_common_info – Get the flash type, page size, block size and block count from flash.

8.3.2.2 FTL Example

We can use the ATCMD and Littlefs as the example.

About the atcmd, please modify the platform_opts.h to enable the ATCMD example. We provide the below command to r/w the flash, it is located at the atcmd_ftl.c.

```
#define CONFIG_FTL                                      0   //support FTL AT command
```

- AFWD – Write the data into flash.
- AFRD – Read the data from flash.
- AFTR – Select the block and page to read the Nand flash data.
- AFTR – Select the sector to read the Nor flash data.
- AFTW – Select the block and page to write the data into nand flash.
- AFTW – Select the sector to write the data into nor flash.
- AFTE – Select the block to erase the Nand flash.
- AFTE – Select the sector to erase the nor flash.

About the file system, please select the littlefs to execute the example.

```
$ cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DEXAMPLE=std_file
```

8.3.2.3 Read FTL flash data in bootloader

The normal FTL API must be used after OS initialization in main(). To read FTL flash data in bootloader, please refer to video_user_boot.c.

```
int boot_read_flash_data(unsigned int address,unsigned char *buf,int length)
```

8.4 NOR Flash

8.4.1 NOR Flash mbed API

NOR Flash mbed API is used to access Flash physical location and can be used in the bootloader.
Please refer to flash_api.h & flash_api.c

```
/**
  * @brief  Erase flash sector
  * @param  obj: Flash object define in application software.
  * @param  address: Specifies the starting address to be erased.
  * @retval none
  */
void flash erase sector(flash t *obj, uint32 t address);

/**
  * @brief  Erase flash block(64KB)
  * @param  obj: Flash object define in application software.
  * @param  address: Specifies the starting address to be erased.LSB 16bits will be
masked.
  * @retval none
  */
void flash_erase_block(flash_t *obj, uint32_t address);

/**
  * @brief  Read a word from specified address
  * @param  obj: Flash object define in application software.
  * @param  address: Specifies the address to read from.
  * @param  data: Specified the address to save the readback data.
  * @retval 1: Success
  * @note auto mode read is ok, because we have flash cache
  */
int flash_read_word(flash_t *obj, uint32_t address, uint32_t *data);

/**
  * @brief  Write a word to specified address
  * @param  obj: Flash object define in application software.
  * @param  address: Specifies the address to be programmed to.
  * @param  data: Specified the data to be programmed.
  * @retval 1: Success
  * @note user mode write used
  */
int flash_write_word(flash_t *obj, uint32_t address, uint32_t data);

/**
  * @brief  Read a stream of data from specified address
  * @param obj: Flash object define in application software.
  * @param  address: Specifies the starting address to read from.
  * @param  len: Specifies the length of the data to read.
  * @param  data: Specified the address to save the readback data.
  * @retval 1: Success
  * @note auto mode is ok, because we have flash cache
  */
int flash stream read(flash t *obj, uint32 t address, uint32 t len, uint8 t *data);

/**
  * @brief  Write a stream of data to specified address
  * @param  obj: Flash object define in application software.
```

```
  * @param  address: Specifies the starting address to write to.
  * @param  len: Specifies the length of the data to write.
  * @param  data: Pointer to a byte array that is to be written.
  * @retval 1: Success
  */
int flash_stream_write(flash_t *obj, uint32_t address, uint32_t len, uint8_t *data);
```

8.4.1.1 NOR flash mbed example

This example demonstrates how use mbed API to read/write a NOR flash.

The example is located in:

"\project\realtek_amebapro2_v0_example\example_sources\flash\"

Copy main.c to src folder, compile project, and the download the binary.

9 OTA

Over-the-air programming (OTA) provides a methodology to update device firmware remotely via TCP/IP network connection. OTA only supports firmware1 and firmware2, OTA does not support updating Key Certificate, Bootloader and ISP_IQ data.

9.1 OTA Operation Flow for NOR Flash

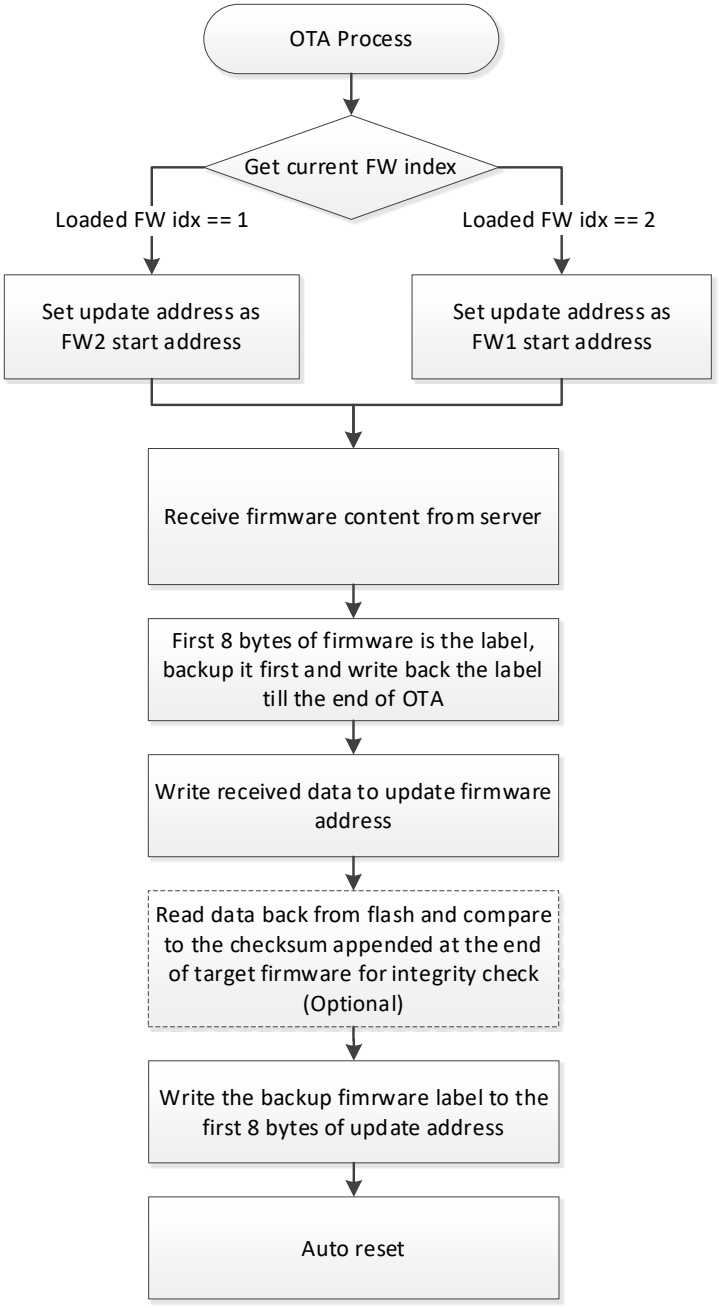


Figure 9-1 OTA Process Flow for NOR Flash

The first 8 bytes of firmware image would be a label. During the step of “Write received data to update firmware address”, the 8 bytes label need set to 0xffffffffffffffff. That means the label is invalid. The backup label needs to be written back at the end of OTA process to prevent device booting from incomplete firmware.

9.2 OTA Operation Flow for NAND Flash

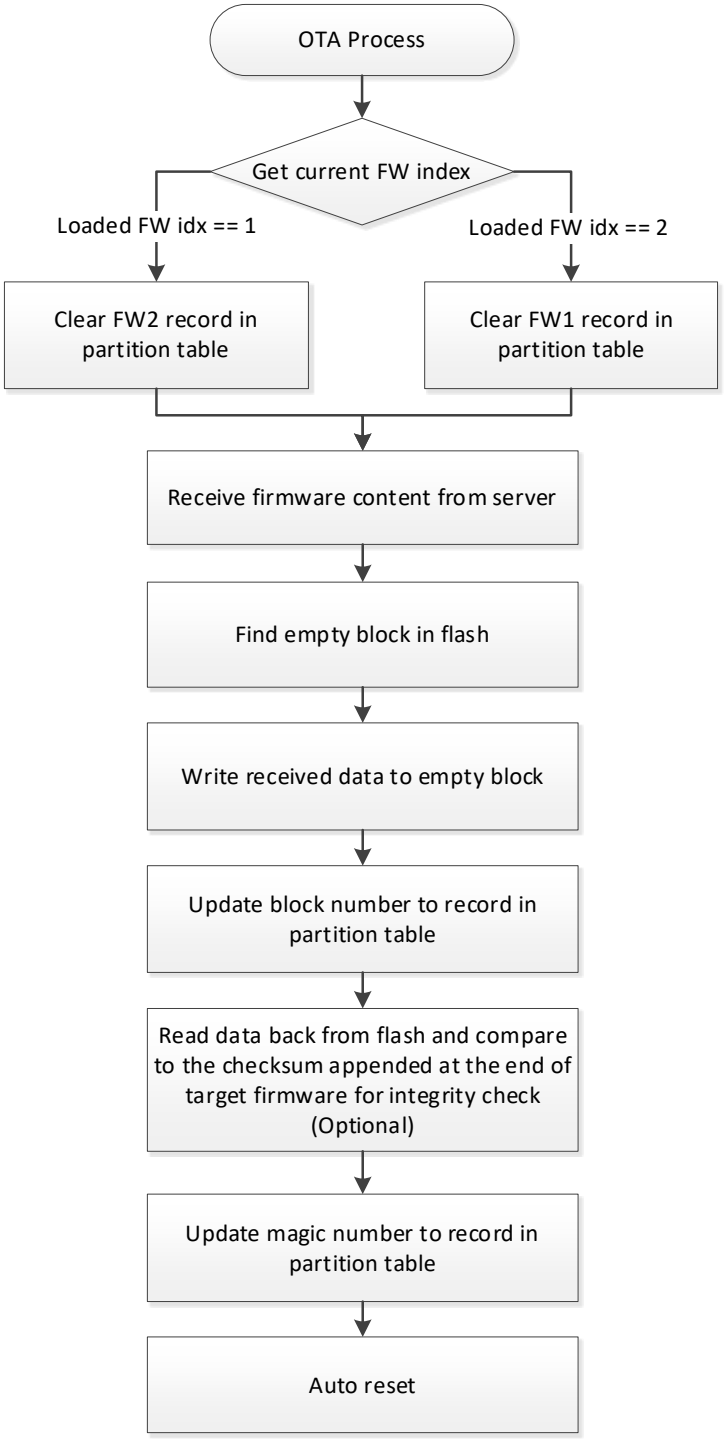


Figure 9-2 OTA Process Flow for NAND Flash

During the step of "Write received data to empty block", the magic number of this new record in partition table is invalid. The magic number is updated at the end of OTA process to make this record valid.

9.3 OTA Checksum Mechanism

The first 8 bytes firmware label for NOR flash OTA process and the magic number of record in partition table for NAND flash OTA process are used to notice the bootloader the overall OTA process is done without any network disconnection or re-boot during the OTA. However, firmware label or magic number cannot guarantee the content of firmware image is correct.

User can design a mechanism to calculate the hash of target OTA firmware for integrity check during the OTA update process. For the default OTA example in SDK, there is USE_CHECKSUM option for this integrity check purpose. During image build, SDK would append 4 bytes checksum at the end of firmware image to become OTA image (ota.bin). When performing OTA routine, right after the firmware is

downloaded and programmed into flash, it would read back all the programed data from flash and compare with the checksum value from target firmware if USE_CHECKSUM enabled. In such way, it can ensure the downloaded firmware is transferred completely and correct. For the detail implementation, please refer to OTA example ota_8735b.c in SDK:

```
#define USE_CHECKSUM 1
```

9.4 Boot Process Flow

Boot loader will select latest firmware based on firmware version and timestamp, and load it.

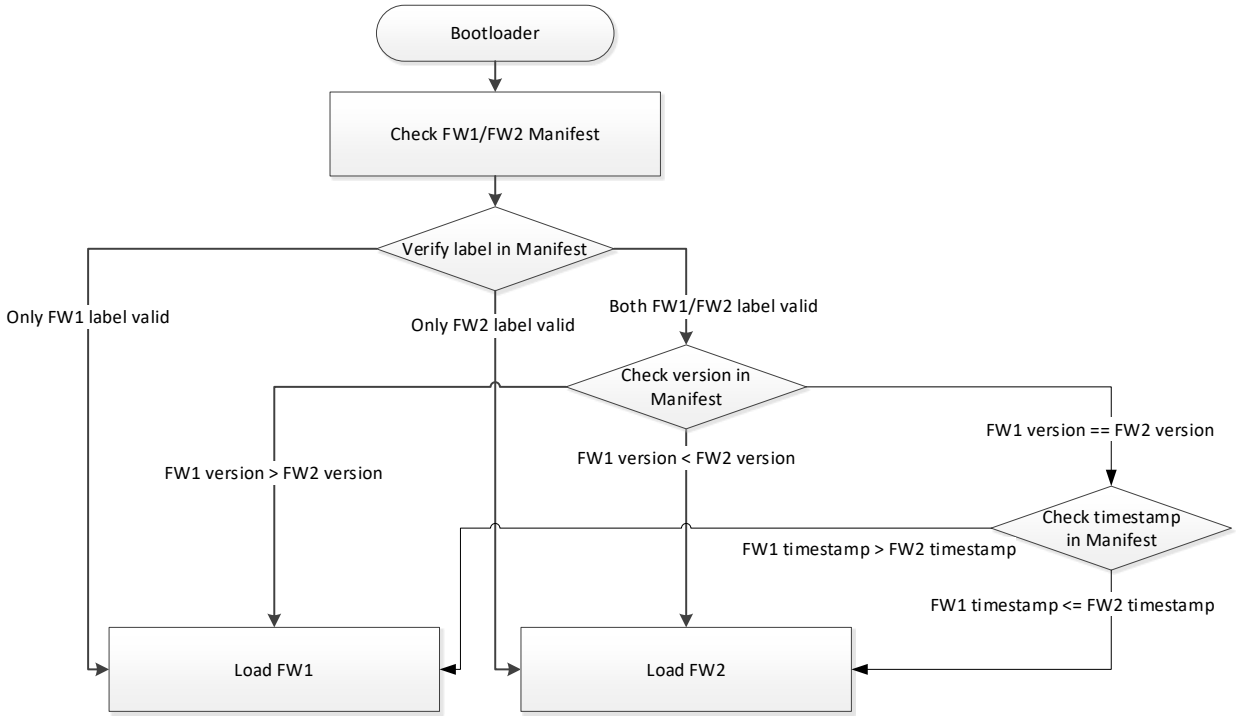


Figure 9-3 Boot Process Flow

Moreover, the default boot process flow can be overwritten by implementing the user_boot_fw_selection() in component\soc\8735b\misc\platform\user_boot.c. The user_boot_fw_selection() should return the index of firmware that will be loaded by bootloader. If the return value is USER_LOAD_FW_FOLLOW_DEFAULT(0), booloader will select FW1 or FW2 based on default boot process flow. If the return value is 1, bootloader will load FW1. If the return value is 2, bootloader will load FW2. To apply user_boot_fw_selection(), it is necessary to re-build bootloader.

```
#include "fw_img_export.h"

uint8_t user_boot_fw_selection(
fw_img_user_export_info_type_t *pfw_img_user_export_info)
{
 uint8_t fw_ld_idx = USER_LOAD_FW_FOLLOW_DEFAULT;
 return fw_ld_idx;
}
```

9.5 Upgraded Partition

In AmebaPro2 OTA update procedure, Firmware 1 and Firmware 2 are swapped to each other.

The Firmware 1/Firmware 2 partition addresses and length are stored in partition records, defined in 'amebapro2_partitiontable.json' under 'project\realtek_amebapro2_v0_example\GCC-RELEASE\mp'. Please adjust it according to your firmware size.

```
        "fw1":{
            "start addr" : "0x100000",
            "length" : "0x300000",
            "type": "PT_FW1",
            "valid": true
        },
        "fw2":{
            "start addr" : "0x400000",
            "length" : "0x300000",
            "type": "PT_FW2",
            "valid": true
```

```
        },
```

For NOR flash, OTA firmware is written to the partition start address in flash, and OTA firmware size is checked with the partition length. For NAND flash, OTA firmware is written to empty blocks distributed in flash, and OTA firmware size is checked with the partition length.

9.6 Firmware Image Output

After building project source files in SDK, it would generate firmware as 'firmware.bin', and OTA firmware as 'ota.bin' which is firmware.bin with 4 bytes checksum appended at the end.

9.6.1 OTA Firmware Swap Behavior

When device executes OTA procedure, it would update another firmware partition, rather than the current running firmware partition. The OTA firmware swap behavior should be looked like as below figure if the updated firmware keeps using newer firmware version and timestamp.

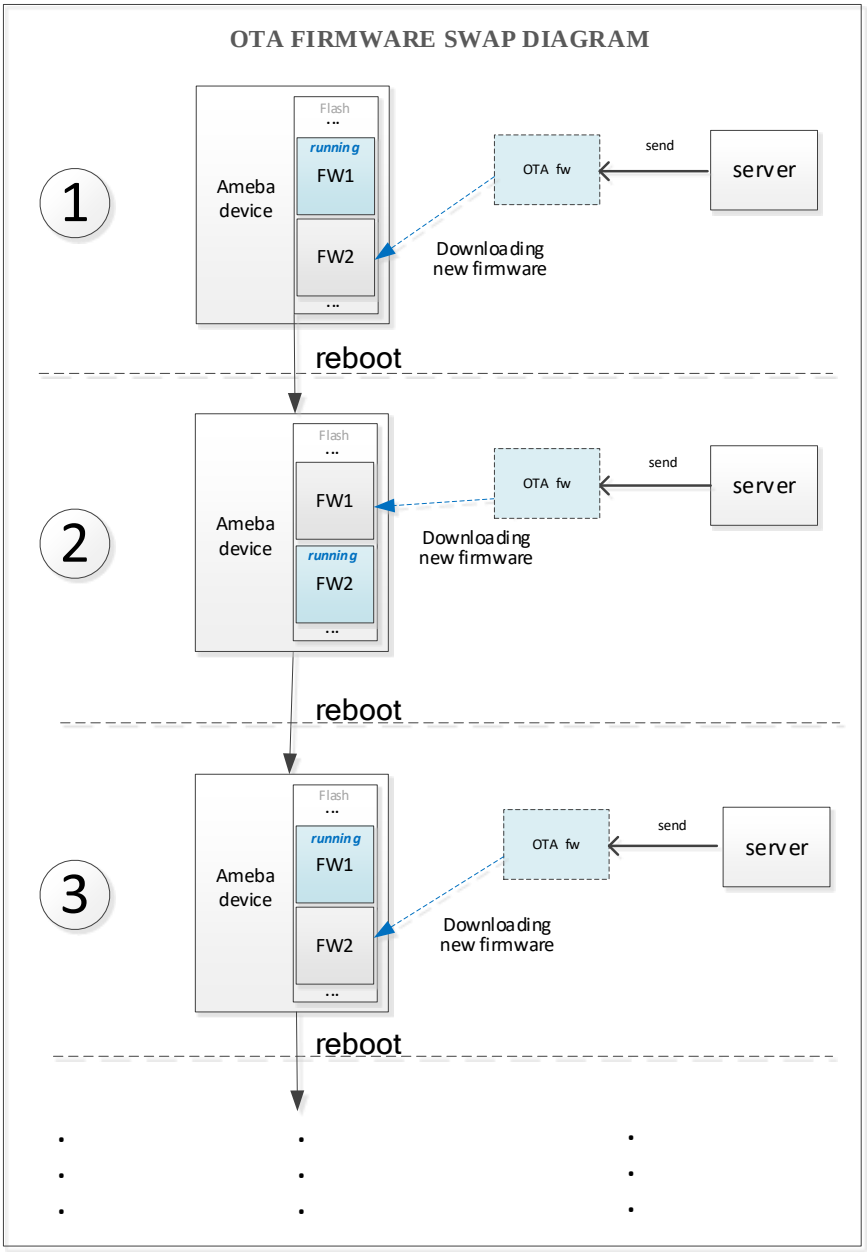


Figure 9-4 OTA Firmware SWAP Procedure

9.6.2 Version and Timestamp

Ameba Pro2 bootloader boots to Firmware 1 or Firmware 2 based on firmware version and timestamp. Please check the version and timestamp of generated OTA firmware are expected.

At offset 0x2B0 of firmware image, the version is a 32bytes value in little endian order. The version can be configured in 'amebapro2_firmware_ntz.json' under 'project\realtek_amebapro2_v0_example\GCC-RELEASE\mp'.

```
"MANIFEST":{
"label":"RTL8735B",
"vrf_alg": "NA_VRF_CHECK",
"tlv":[
…
{"type":"TYPE_ID", "length":2, "value":"IMG_FWHS_S"},
{"type":"VERSION",
 "length":32,
 "value":"FEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"},
{"type":"TIMST", "length":8, "value":"auto"},
```

The version which higher bit is zero presents higher version. For example, the version of 'FEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' is zero in bit 0, version of 'EFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' is zero in bit 4, and version of 'FFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' is zero in bit 8. Then, it will be version with bit 8 zero > version with bit 4 zero > version with bit 0 zero. The version definition in "value" is customizable, user could apply its own method by implementing user_boot_fw_selection(). Moreover, the advantage of using "value" field in MANIFEST instead of maintaining a separate record in NOR/NAND flash is that, user does not need to handle bad block management for NAND flash that stores firmware version control information.

At offset 0x2D4 of firmware image, the timestamp is an 8bytes value in little endian order and is the number of seconds since January 1st, 1970 00:00:00 UTC. The timestamp which presents image build time will be automatically generated when image build.

In user_boot_fw_selection() of bootloader, developer could get firmware version and timestamp from the inputted parameter pfw_img_user_export_info of user_boot_fw_selection(). The pfw_img_user_export_info-> fw1_ld_sel_info.version and pfw_img_user_export_info-> fw2_ld_sel_info.version are the raw data of FW1 and FW2 version. The pfw_img_user_export_info-> fw1_ld_sel_info.timestamp and pfw_img_user_export_info-> fw2_ld_sel_info.timestamp are 4-bytes of FW1 and FW2 timestamp. In application firmware, developer could get firmware version and timestamp from hal_sys_get_fw_version_raw() and hal_sys_get_fw_timest().

```
// hal_sys_ctrl.h
void hal_sys_get_fw_version_raw(const uint8_t ld_img_idx, uint8_t *pver_raw_buf);
uint32_t hal_sys_get_fw_timest(const uint8_t ld_img_idx);
```

9.7 Implement OTA over Wi-Fi

9.7.1 OTA Using Local Download Server Base on Socket

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

NOTE

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

9.7.1.1 Build OTA Application Image

Enable CONFIG_OTA_UPDATE flag in 'project\realtek_amebapro2_v0_example\inc\platform_opts.h' to support ATWO AT command for OTA with local download server.

```
#define CONFIG_OTA_UPDATE       1
```

Download the firmware to AmebaPro2 board to execute OTA.

9.7.1.2 Setup Local Download Server

Step 1: Build new ota.bin and place it to 'tools\DownloadServer' folder.

Step 2: Edit 'tools\DownloadServer\start.bat' file for server port and OTA file name

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

Step 3: Execute 'tools\DownloadServer\start.bat'.

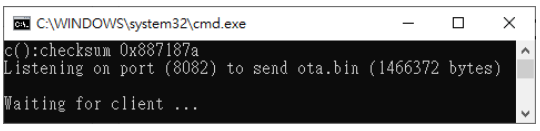


Figure 9-5 Download Server

9.7.1.3 Execute OTA Procedure

After device connects to AP, enter command: ATWO=IP[PORT]. Please note that the device and your PC need under the same AP. The IP in ATWO command is the IP of your PC.

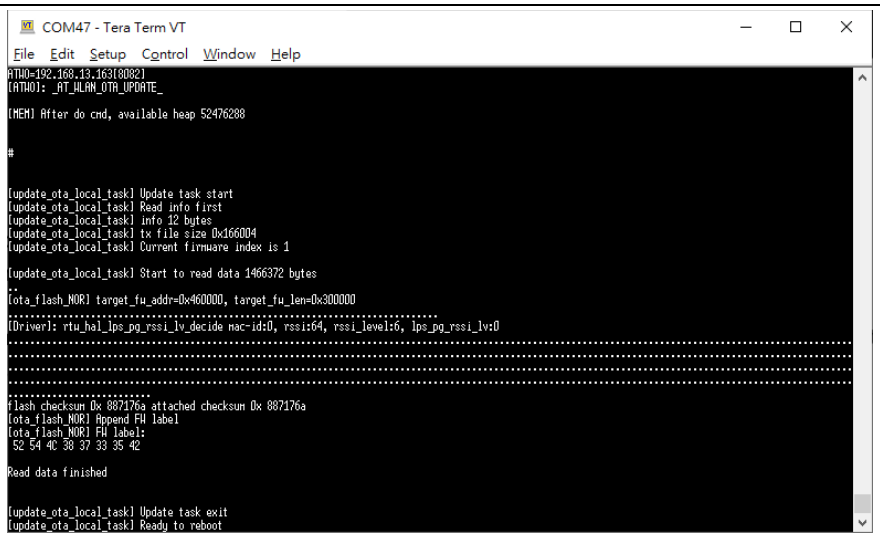


Figure 9-6 ATWO command

After finishing OTA download, device will reboot automatically, and the bootloader will boot to new firmware according to firmware version and timestamp.

9.7.2 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.bin' after receiving the http request.

NOTE

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

9.7.2.1 Build OTA Application Image

Set server IP, port, and resource in ota_http example (component\example\ota_http\example_ota_http.c).

```
#define PORT   8082
#define HOST   "192.168.1.100"
#define RESOURCE "ota.bin"
```

Build firmware with ota_http example.

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake
-DEXAMPLE=ota_http
```

Download the firmware to AmebaPro2 board to execute OTA.

9.7.2.2 Setup Local HTTP Download Server

Step 1: Build new ota.bin and place it to 'tools\DownloadServer(HTTP)' folder.

Step 2: Edit 'tools\DownloadServer\start.bat' file for server port and OTA file name

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

Step 3: Execute 'tools\DownloadServer(HTTP)\start.bat'.

Figure 9-7 HTTP Download Server

9.7.2.3 Execute OTA Procedure

Reboot the device and connect to AP, it should execute ota_http example automatically to start the OTA update through HTTP protocol.

After finishing OTA download, device will reboot automatically, and the bootloader will boot to new firmware according to firmware version and timestamp.

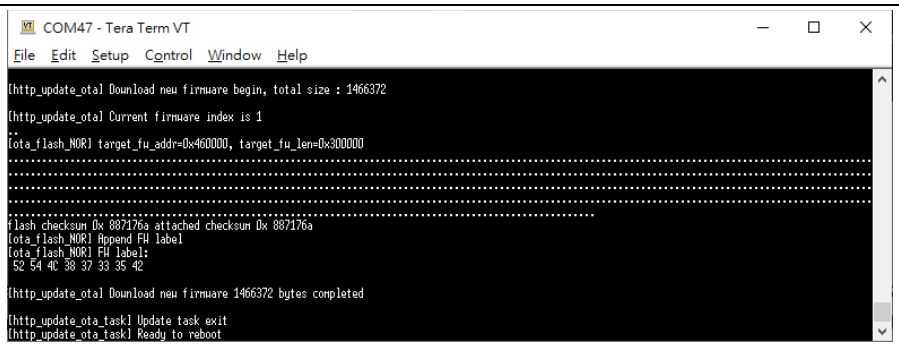


Figure 9-8 OTA HTTP Example

9.8 Verify Signature after OTA

For secure boot enabled devices, ota_verify_flash_signature() and ota_verify_signature() are provided in ota_8735b.c to verify the public key and signature in manifest of signed bootloader or firmware after OTA.

```
// ota_8735b.c
int ota_verify_flash_signature(char *partition_name);
int ota_verify_signature(char *partition_name, uint8_t *manifest);
```

ota_verify_signature() can be used to verify the public key and signature in a manifest buffer. The manifest for partitions of "BL_PRI", "FW1" or "FW2" can be inputted. In ota_verify_signature(), the public key in the manifest is verified with the public key hash in Key Certificate in flash, and the signature in the manifest is verified with this verified public key. ota_verify_flash_signature() can be used to verify the public key and signature in the manifest of partitions in flash. ota_verifiy_flash_signature() reads manifest from the partitions in flash by using FWFS API, such pfw_open() and pfw_read(), and uses ota_verify_signature() to verify this manifest. After OTA, ota_verifiy_flash_signature() can be used to verify the updated firmware partition before reboot. Should not switch firmware to the updated firmware if the verification of public key or signature of the updated firmware is failed.

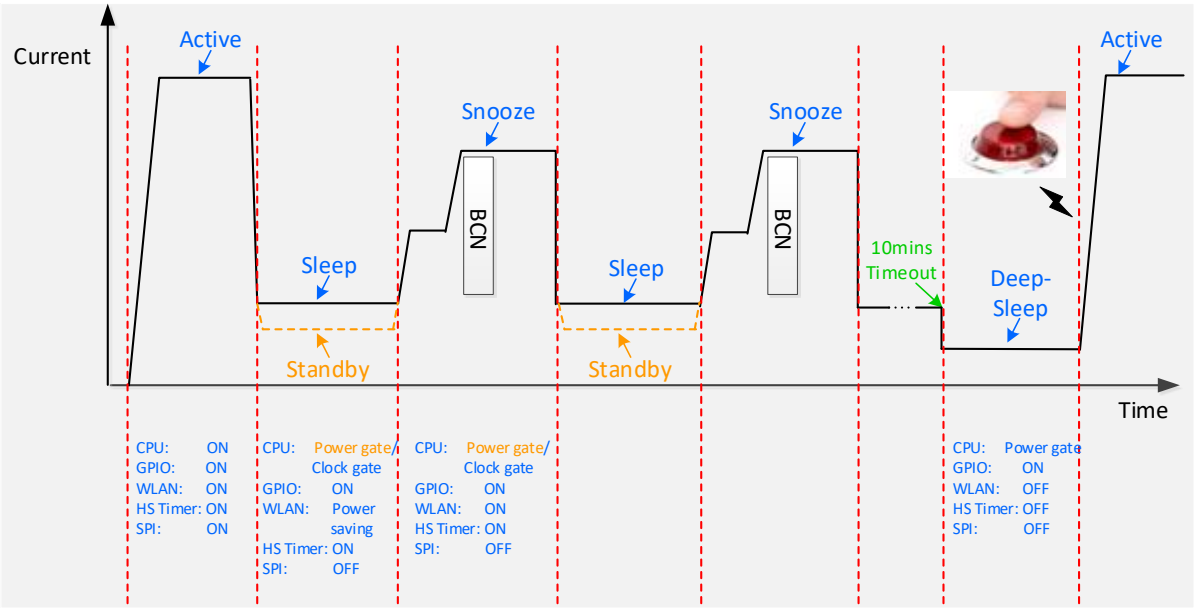
10 Power Save

10.1 Overview

10.1.1 Application Scenario

Ameba Pro2 achieves low power consumption with a combination of several proprietary technologies. The power-saving architecture features six reduced power modes of operation: active, sleep, standby, snooze, deepsleep, shutdown mode. With the elaborate architecture, the battery life of whole IOT system could be extended.

For reading pen application, it can divide into three-scenario. First, press the power button to power on reading pen to active mode and then connect to the cloud to download data. Second, once the reading pen without any activity for 2 minutes, the system will go to sleep mode (for system fast resume and keep WIFI connect) or standby mode (for lower power consumption and keep WIFI connect) and regularly wake up to receive WLAN beacon while into snooze mode. At last, without using the reading pen exceeds 10 minutes, the system will into deep sleep mode and waiting for any button signals to wake up system into active mode. The application scenario flow was shown in

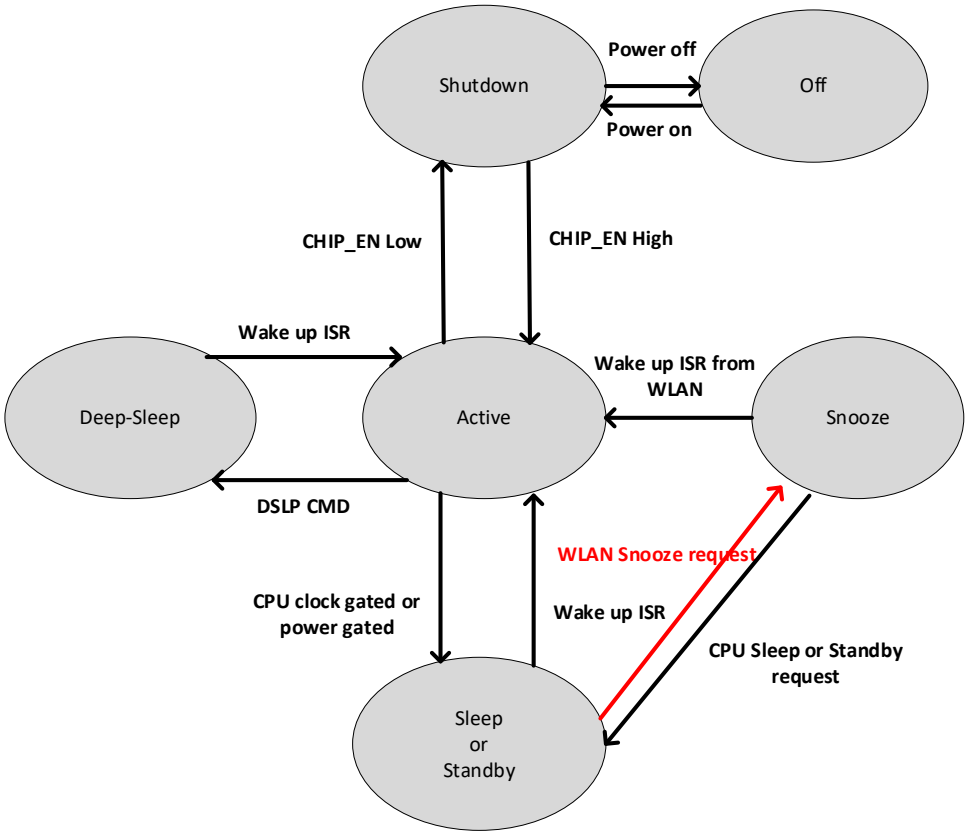


10.1.2 Features

- **Active Mode**: The CPU is in active mode and all peripherals are available.
- **Sleep Mode**: The CPU is in clock-gated and can be woken up by most of peripherals. The system resume time could be much faster than the Standby mode and the WLAN could be ON or power saving mode in this state.
- **Standby Mode**: The CPU is in power-gated and can be woken up by most of peripherals. The power consumption could be lower than the Sleep mode and the WLAN could be ON or power saving mode in this state.
- **Snooze Mode**: In this mode system can regularly wake up to receive WLAN beacon without software intervention. The significant difference between Snooze and Sleep/Standby mode is WLAN capability and could be receive and transmit beacon in this state.
- **DeepSleep Mode**: The lowest power consumption than the other power mode except for shutdown mode, it can be only woken up by LP Timer or GPIO.
- **Shutdown Mode:** The CPU will be shutdown while CHIP_EN was Low.

10.1.3 Power Mode and Power Consumption

The mode transition diagram is given in.



In Figure, the power mode can be divided into 6 states except for “off” state and the each power In Figure, the power mode can be divided into 6 states except for “off” state and the each power consumption was shown in Figure. The introduction of each power mode, clock-gated and power-gated state will be in the following sections. Clock/power gated state could be regarded as a status of any hardware.

10.2 Deep Sleep Mode

- CHIP_EN keeps high. User can invoke Deep Sleep API to force into deep sleep mode. By using specified interrupts to wake up system.
- The following wake flow: Wake up ISR is high -> PMC -> enable CPU -> Reboot flow.

10.2.1 Wakeup Source

Aon GPIO, RTC, comparator, Aon Timer
Aon GPIO: GPIOA0~GPIOA3
Comparator: GPIOA0~GPIOA3

10.3 Standby Mode

- CHIP_EN keeps high. User can invoke Standby API to force into deep sleep mode. By using specified interrupts to wake up system.
- The following wake flow: Wake up ISR is high -> PMC -> enable CPU -> Fast reboot flow.

10.3.1 Wakeup Source

Aon GPIO, RTC, comparator, Aon Timer, Gtimer0, PWM, Pon GPIO, Uart0, Wlan
Aon GPIO: GPIOA0~GPIOA3
Pon GPIO: GPIOF0~GPIOF17
Comparator: GPIOA0~GPIOA3
*The clock source of gtimer0 and Uart0 only support 4MHz clk.

10.4 Sleep Mode

- CHIP_EN keeps high. User can invoke Sleep API to force system into deep sleep mode. By using specified interrupts to wake up system.
- The following wake flow: Wake up ISR is high -> PMC -> enable CPU -> Execution of instructions continues.

10.4.1 Wakeup Source

Aon GPIO, RTC, comparator, Aon Timer, Gtimer0, PWM, Pon GPIO, Uart0, Wlan
Aon GPIO: GPIOA0~GPIOA3
Pon GPIO: GPIOF0~GPIOF17
Comparator: GPIOA0~GPIOA3

10.5 Snooze Mode

- CHIP_EN keeps high. By using specified interrupts to wake up system.
- The following wake flow: WLAN power on request-> Receive particular beacon-> Wake up ISR is high -> PMC -> enable CPU -> Execution of instructions continues or fast reboot flow.

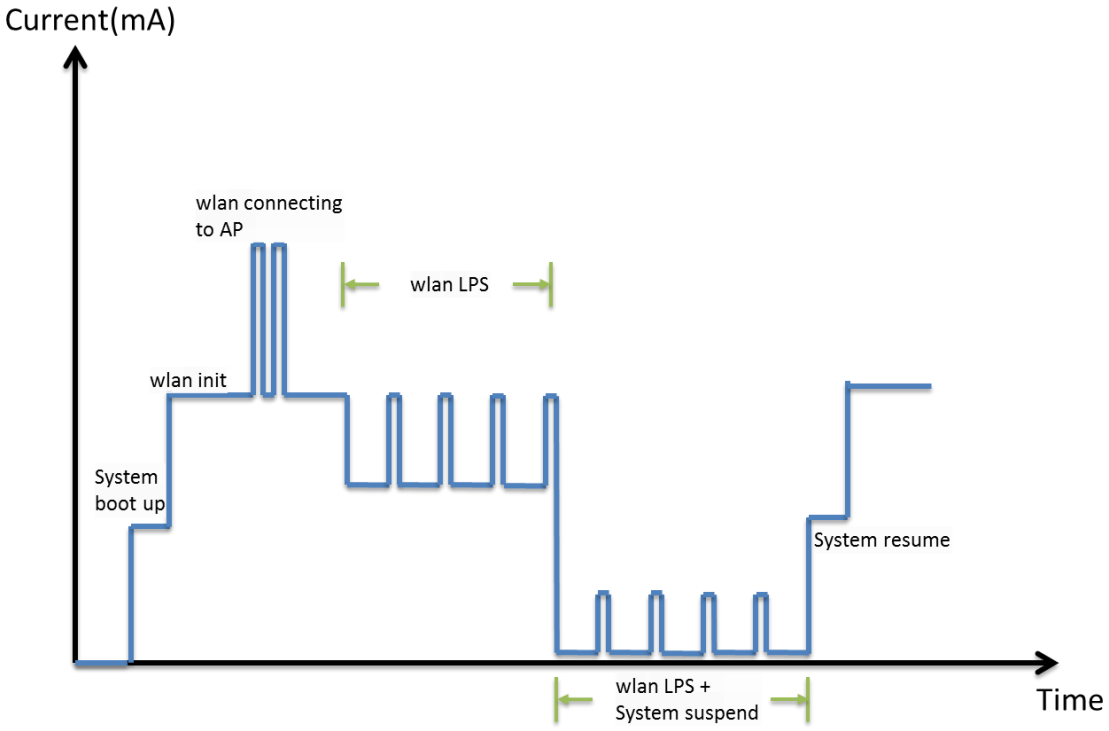
10.5.1 Wakeup Source

In snooze mode, the only wake up source is WLAN. The wakeup condition could be configured by WLAN driver according to system application. Once the event triggered, WLAN hardware would raise interrupt to PMC that could change hardware state.

10.5.2 Wakeup from WLAN

If user choose wakeup from WLAN in Standby mode, he needs to configure wlan before invoke Standby.
After user configure wlan and invoke Standby, system would enter suspend state. In this state, MCU would periodically to check wlan state and decide if it needs to wakeup. After wlan receive the wakeup packet that matches the wakeup pattern, then wakeup.

The whole progress is like below diagram:



- At first system boot up, initialize wlan, connecting to AP.
- 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.
- If user try to make system save more power with wlan associate idle, he could invoke Standby. 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 and current meter could handle the voltage drop and rise between hundreds of micro amp and dozens of milliamp.

10.5.3 Keep-alive mechanism

Wowlan mode can perform TCP keep alive and MQTT ssl ping request keep alive by setting a fixed pattern.

10.5.3.1 TCP keep alive

In first, set 2 parameters: IP & port of TCP server. The setting would become effective after system enter wake on wlan mode. By default it sends a TCP packet every 30s, and resend after 10s if STA does not receive TCP ACK. You can modify the interval in the setting.
In the offload setting, it also adds a wake on wlan pattern that matches TCP packet with same source port in tcp keep alive, and match TCP flag with PSH+ACK. These setting would allow TCP server wakeup STA by sending a packet to STA.

10.5.3.2 MQTT SSL keep alive

Just like tcp keep alive, we only need to set the IP & port of MQTT server and SSL Key offload after TLS connection, and fill in {0xc0, 0x00} in the tcp payload to send ping request packets at a fixed time for MQTT keep alive .

10.5.4 Wakeup from pattern

When the user needs to use the remote wake-up function, the system can leave Standby mode when wlan receives a matching data packet. The SDK has been configured with the ICMP pattern to wake up, and supports user-set custom patterns.
The part of the Wakeup pattern comparison includes the data of Destination, BSSID, Source in the MAC Header, and Destination IP Address from the Protocol Type of the LLC Header to the Destination IP Address of the IP Header. To set a custom pattern, users need to set (1) Pattern content and (2) Mask: Pattern content to compare Byte, the above two items must be set in the wowlan_pattern_t structure:

```
typedef struct wowlan_pattern {
 unsigned char eth_da[6];
 unsigned char eth_sa[6];
 unsigned char eth_proto_type[2];
 unsigned char header_len[1];
 unsigned char ip_proto[1];
 unsigned char ip_sa[4];
 unsigned char ip_da[4];
 unsigned char src_port[2];
 unsigned char dest_port[2];
 unsigned char flag2[1];
 unsigned char mask[6];

unsigned char window[2]; //Reserved
unsigned char checksum[2];  //Reserved
unsigned char urgent_pointer[2]; //Reserved
unsigned char payload[64];
unsigned char payload_mask[9];
} wowlan_pattern_t;
```

After setting the pattern and mask, you need to set to wlan using the wifi_wowlan_set_pattern API.
Take the TCP data packet as an example, assuming the Ameba MAC address is 00: E0: 4C: 87: 00: 00, the following description sets the TCP Unicast data packet whose receiver is Ameba as the wake-up packet. First, the MAC Destination of the comparison packet needs to be Ameba, so set the eth_da field of wowlan_pattern_t to 00: E0: 4C: 87: 00: 00. Next, set the Protocol type of the LLC Header to IP Protocol: {0x08, 0x00}, Version + Length: {0x45}, the Protocol Type of the IP Header to TCP: {0x06}, and set the Destination IP to Ameba's IP.

After the above fields are set, they will be converted into HW comparison format, as follows:

eth_da (6)	eth_sa (6)	eth_proto_type (2)	header_len (1)	Rsvd (8)	ip_proto (1)	Rsvd (2)	ip_sa (4)	ip_da (4)	src_port (2)	Dest_port (2)	Flag2 (1)

Therefore, the HW Pattern after the TCP Pattern conversion in this example is:

00 e0 4c 87 00 00	00 00 00 00 00 00	08 00	45	00 00 00 00 00 00 00	06	00 00	00 00 00 00	c0 a8 00 c4	00 64	e1 6c	18

Next, the user needs to set the Mask. 1 bit in the Mask corresponds to 1 byte of the HW pattern, and the byte corresponding to the 1 bit in the Mask will be added for comparison. The following explains how Mask is composed:
First use the bit sequence mask to identify the bytes to be compared:

111111	000000	11	1	00000000	1	00	0000	1111	11	11	1

This bit sequence is composed of 1 byte every 8 bits:

11111100	00001110	00000001	00000011	111111	1

HW is compared from bit0 of each byte, so the bit order of each byte must be reversed.

00111111	01110000	10000000	11000000	00111111	10000000

Convert from step 3 to Hex: {0x3f, 0x70, 0x80, 0xc0, 0x3f, 0x80} to get the final Mask.

10.5.4.1 Wakeup pattern payload

Set the payload_mask, 1 bit in the Mask corresponds to 1 byte of the HW pattern, and the byte corresponding to the bit of Mask 1 will be added to the comparison. Assuming that a payload of 10 bytes is set, the following explains how the Mask is composed:
Every 8 bits of the bit sequence form 1 byte, and the first 6 bits are reserved:

00000011	11111111	00000000	00000000	00000000	00000000	00000000	00000000	00000000

HW is compared from the bit0 of each byte, so the bit order of each byte must be reversed

11000000	11111111	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Convert to Hex from step 2: {0xc0, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, get the final Mask.

10.5.5 Wakeup from SSL pattern

Wake up from ssl mode support tls cipher suite aes256, sha384.
After establishing a TLS connection, you can wake up by setting ssl key offload and ssl wakeup mode.
When the user needs to use the remote wake-up function, when wlan receives a matching data packet, the system can exit the Standby mode.

10.5.5.1 SSL key offload

The part of the SSL Key offload comparison includes the ssl_ctr, ssl_iv, ssl_enc_key, ssl_dec_key and ssl hmac key. To set a SSL Key offload, users need to set wifi_set_ssl_offload() after TLS connection, the above items must be set in the SSL Key offload structure:

SSL_CTR (8 bytes)
SSL_IV (16 bytes)
SSL_ENC_KEY (32 bytes)
SSL_DEC_KEY (32 bytes)
SSL_HMAC_KEY (48 bytes)
SSL_IS_ETM (1 bytes)

10.5.5.2 SSL wakeup pattern

SSL Wakeup pattern support 8 groups, each pattern supports up to 64 Bytes. The user can fill in the patterns used for waking up in sequence through wifi_wowlan_set_ssl_pattern(). After waking up, the wakeup reason can know which pattern was awakened.

10.5.5.3 SSL wakeup pattern

SSL Wakeup pattern support 8 groups, each pattern supports up to 64 Bytes. The user can fill in the patterns used for waking up in sequence through wifi_wowlan_set_ssl_pattern(). After waking up, the wakeup reason can know which pattern was awakened.

10.5.6 Example of Wakeup from WLAN

The sample program is located at:
project\realtek_amebapro2_v0_example\src\wowlan\main_wowlan.c

Example	Description	Result
WiFi Keep alive	Use command: PS=wowlan	Enter L2 keep alive mode

TCP Keep alive	Send keep alive packet interval setting:
	static uint32_t interval_ms = 60000;
	static uint32_t resend_ms = 10000;
	
	Use command:
	PS=tcp_keep_alive,[ip],[port]
	PS=wowlan
		A tcp keep alive packet is sent every 60 seconds, if no tcp ack is received, the tcp keep alive packet will be resend 10 seconds later.
		
		Connect to tcp server and enter tcp keep alive mode.

MQTT SSL Keep alive	Send keep alive packet interval setting:
	static uint32_t interval_ms = 60000;
	static uint32_t resend_ms = 10000;
		A mqtt ping request packet is sent every 60 seconds, if no tcp ack is received, the mqtt ping request packet will be resend 10 seconds later.

	Use command: #define MQTTSSL_KEEPALIVE 1 #define AWS_IOT_MQTT 1 PS=mqtt_keep_alive PS=wowlan	Connect to mqtt server and enter tcp keep alive mode

10.5.7 Suspend fail handle

When entering wlan suspend, there may be a chance of sleep failure, such as receiving a wakeup event or wifi disconnection during the suspend process. At this time, the system needs to be rebooted to resume normal wlan operation.

```
ret = rtl8735b_suspend(0);
if (ret != 0)
{
	/*****************************************
	ret = -1;  //rtw_hal_wow_enable fail
	ret = -2;  //disable wlan interrupt fail
	ret = -3;  //wlan enter LPS mode fail
	ret = -4; //wifi not connected when suspend
	ret = -5; //reserve
	ret = -6; //wifi off status check fail
	*****************************************/
	sys reset();
}
```

10.6 Set AON GPIO pull control before power save mode

Check the AON GPIO external circuit status and set the AON GPIO states matching the external circuit to prevent leakage.

- External pull high → internal pull high
- External pull low or floating → internal pull low

For the detailed definition of AON GPIO control register, please refer to rtl8735b_aon_type.h.

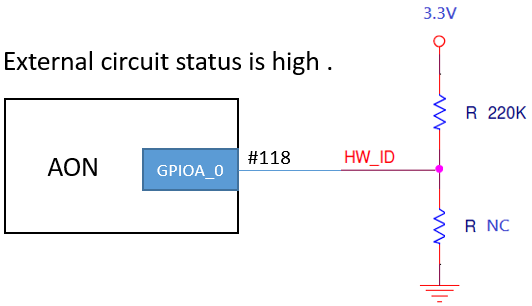
```
#define AON_SHIFT_AON_GPIO1_SLEW_RATE                28
#define AON_BIT_AON_GPIO1_SLEW_RATE                  ((u32)0x00000001 << 28)
/*!<R/W 0  0:Disable,1:Enable */
#define AON_SHIFT_AON_GPIO1_DRIVING                  26
#define AON_MASK_AON_GPIO1_DRIVING                   ((u32)0x00000003 << 26)
/*!<R/W 0  0:4mA, 1:8mA */
#define AON_SHIFT_AON_GPIO1_SMT_EN                   25
#define AON_BIT_AON_GPIO1_SMT_EN                     ((u32)0x00000001 << 25)
/*!<R/W 0  Enable AON GPIO1 Schmitt trigger; 1: enable */
#define AON_SHIFT_AON_GPIO1_PULL_CTRL                22
#define AON_MASK_AON_GPIO1_PULL_CTRL                 ((u32)0x00000003 << 22)
/*!<R/W 0  2b'00: high impedence; 2b'01: pull low; 2b'10: pull high; 2b'11: reserved */
#define AON_SHIFT_AON_GPIO1_PINMUX_SEL               16
#define AON_MASK_AON_GPIO1_PINMUX_SEL                ((u32)0x0000000F << 16)
/*!<R/W 1111  0000: Comparator_ADC 0001: I2C0_SDA 0010: 0011:JTAG_CLK/SWD_CK 0100: 0101:
0110: 0111: 1000: 1001: 1010: 1011: 1100: 1101: 1110: 1111: GPIO */
#define AON_SHIFT_AON_GPIO0_SLEW_RATE                12
#define AON_BIT_AON_GPIO0_SLEW_RATE                  ((u32)0x00000001 << 12)
/*!<R/W 0  0:Disable,1:Enable */
#define AON_SHIFT_AON_GPIO0_DRIVING                  10
#define AON_MASK_AON_GPIO0_DRIVING                   ((u32)0x00000003 << 10)
/*!<R/W 0  0:4mA, 1:8mA */
#define AON_SHIFT_AON_GPIO0_SMT_EN                   9
#define AON_BIT_AON_GPIO0_SMT_EN                     ((u32)0x00000001 << 9)
/*!<R/W 0  Enable AON GPIO0 Schmitt trigger; 1: enable */
#define AON_SHIFT_AON_GPIO0_PULL_CTRL                6
#define AON_MASK_AON_GPIO0_PULL_CTRL                 ((u32)0x00000003 << 6)
/*!<R/W 0  2b'00: high impedence; 2b'01: pull low; 2b'10: pull high; 2b'11: reserved */
#define AON_SHIFT_AON_GPIO0_PINMUX_SEL               0
#define AON_MASK_AON_GPIO0_PINMUX_SEL                ((u32)0x0000000F << 0)
/*!<R/W 1111  0000: Comparator_ADC 0001: I2C0_SCL 0010: 0011: JTAG_TMS/SWD_IO 0100: 0101:
0110: 0111: 1000: 1001: 1010: 1011: 1100: 1101: 1110: 1111: GPIO */
```

10.7 Set PON GPIO pull control before standby

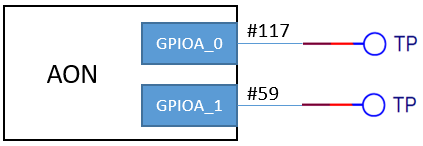
If PON GPIO is set as standby wake up source, please check the PON GPIO external circuit status first, and set the PON GPIO status matching the external circuit before standby to prevent leakage.

- External pull high → internal pull high
- External pull low or floating → internal pull low

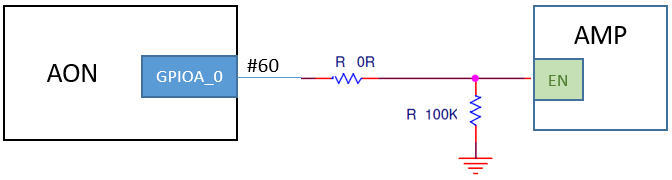
Ex. External circuit status is high .



Ex. External circuit status is NC or TP .



Ex. External circuit status is low.



For the detailed definition of PON GPIO control register, please refer to rtl8735b_pon_type.h.

```
#define PON_SHIFT_PON_GPIO1_SLEW_RATE                28
#define PON_BIT_PON_GPIO1_SLEW_RATE                  ((u32)0x00000001 << 28)
/*!<R/W 0  0:Disable,1:Enable */
#define PON_SHIFT_PON_GPIO1_DRIVING                  26
#define PON_MASK_PON_GPIO1_DRIVING                   ((u32)0x00000003 << 26)
/*!<R/W 0  0:4mA, 1:8mA */
#define PON_SHIFT_PON_GPIO1_SMT_EN                   25
#define PON_BIT_PON_GPIO1_SMT_EN                     ((u32)0x00000001 << 25)
/*!<R/W 0  Enable PON GPIO1 Schmitt trigger; 1: enable */
#define PON_SHIFT_PON_GPIO1_PULL_CTRL                22
#define PON_MASK_PON_GPIO1_PULL_CTRL                 ((u32)0x00000003 << 22)
/*!<R/W 0  2b'00: high impedence; 2b'01: pull low; 2b'10: pull high; 2b'11: reserved */
#define PON_SHIFT_PON_GPIO1_PINMUX_SEL               16
#define PON_MASK_PON_GPIO1_PINMUX_SEL                ((u32)0x0000000F << 16)
/*!<R/W 1111  0000: ADC1 0001: I2C1_SCL 0010: 0011: RFE_CTRL_0 0100: 0101: 0110:UART1_CTS
0111: 1000: 1001: 1010: 1011: 1100: 1101: 1110: 1111: GPIO */
#define PON_SHIFT_PON_GPIO0_SLEW_RATE                12
#define PON_BIT_PON_GPIO0_SLEW_RATE                  ((u32)0x00000001 << 12)
/*!<R/W 0  0:Disable,1:Enable */
#define PON_SHIFT_PON_GPIO0_DRIVING                  10
#define PON_MASK_PON_GPIO0_DRIVING                   ((u32)0x00000003 << 10)
/*!<R/W 0  0:4mA, 1:8mA */
#define PON_SHIFT_PON_GPIO0_SMT_EN                   9
#define PON_BIT_PON_GPIO0_SMT_EN                     ((u32)0x00000001 << 9)
/*!<R/W 0  Enable PON GPIO0 Schmitt trigger; 1: enable */
#define PON_SHIFT_PON_GPIO0_PULL_CTRL                6
#define PON_MASK_PON_GPIO0_PULL_CTRL                 ((u32)0x00000003 << 6)
/*!<R/W 0  2b'00: high impedence; 2b'01: pull low; 2b'10: pull high; 2b'11: reserved */
#define PON_SHIFT_PON_GPIO0_PINMUX_SEL               0
#define PON_MASK_PON_GPIO0_PINMUX_SEL                ((u32)0x0000000F << 0)
/*!<R/W 1111  0000: ADC0 0001: 0010: 0011: 0100: 0101: 0110: 0111: 1000: 1001: 1010: 1011:
1100: 1101: 1110: 1111: GPIO */
```

Default settings for EVB.

```
//AON_GPIO
 HAL_WRITE32(0x40009094, 0x0, 0x4f004f); //GPIOA_1/GPIOA_0
 HAL_WRITE32(0x40009098, 0x0, 0x4f004f); //GPIOA_3/GPIOA_2
 HAL_WRITE32(0x4000909c, 0x0, 0x4f004f); //GPIOA_5/GPIOA_4


//PON_GPIO
 HAL_WRITE32(0x40009850, 0x0, 0x4f004f); //GPIOF_1/GPIOF_0
 HAL_WRITE32(0x40009854, 0x0, 0x8f004f); //GPIOF_3/GPIOF_2
 HAL_WRITE32(0x40009858, 0x0, 0x4f008f); //GPIOF_5/GPIOF_4
 HAL_WRITE32(0x4000985c, 0x0, 0x4f004f); //GPIOF_7/GPIOF_6
 HAL_WRITE32(0x40009860, 0x0, 0x4f004f); //GPIOF_9/GPIOF_8
 HAL_WRITE32(0x40009864, 0x0, 0x4f004f); //GPIOF_11/GPIOF_10
 HAL_WRITE32(0x40009868, 0x0, 0x4f004f); //GPIOF 13/GPIOF 12
 HAL_WRITE32(0x4000986C, 0x0, 0x4f004f); //GPIOF_15/GPIOF_14
 HAL_WRITE32(0x40009870, 0x0, 0x4f004f); //GPIOF_17/GPIOF_16
//HAL_WRITE32(0x4000Ae04, 0x0, 0x20000); //GPIOF_17(VDD_DDR_EN) INPUT MODE
```

11 System API

11.1 System reset

Reset the system

```
/**
  * @brief  system software reset.
  * @retval none
  */
void sys_reset(void)
```

11.2 Get boot select

Identify the system is boot from NAND or NOR flash

```
/**
  * @brief  Get boot select function.
  * @retval boot select device
  * @note
  *  BootFromNORFlash            = 0,
  *  BootFromNANDFlash           = 1,
  *  BootFromUART                = 2
  */
uint8_t sys_get_boot_sel(void)
```

11.3 JTAG/SWD disable

JTAG/SWD is enabled by default. Turn off JTAG/SWD to release more pins to the application

```
/**
  * @brief  Turn off the JTAG/SWD function.
  * @retval none
  */
void sys_jtag_off(void)
```

11.4 MPU read only section protect

Using MPU read only section protection may help to find the issue that may break the read only section like .text or .rodata. The data in those sections should be unchangeable, and protecting those area may help to find the issue that may damage read only sections. To enable this function, please uncomment the following code in main.c.

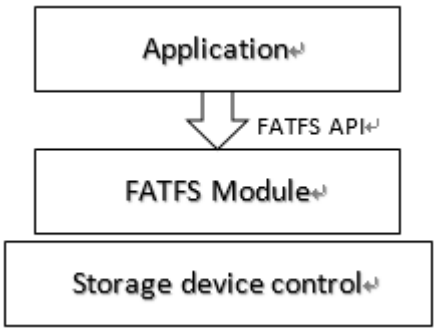
```
/* for debug, protect rodata*/
mpu_rodata_protect_init();
```

12 File System

12.1 FATFS File System

Storage is a key feature of embedded system. AmebaPro2 provides flexible method of storage management. In this chapter, three kinds of application scenarios (SD/RAM/flash) will be mentioned.

12.1.1 FATFS Module



AmebaPro2 utilizes FAT File system Module to provide access to low level storage devices. Applications can manage and operate the file system through FATFS API.

12.1.2 FATFS API

AmebaPro2 SDK uses open source FATFS module. The application interface provides various functions for applications to manipulate the file system.

(1) File Access

- f_open - Open/Create a file
- f_close - Close an open file
- f_read - Read data from the file
- f_write - Write data to the file
- f_lseek - Move read/write pointer, Expand size
- f_truncate - Truncate file size
- f_sync - Flush cached data
- f_forward - Forward data to the stream
- f_expand - Allocate a contiguous block to the file
- f_gets - Read a string
- f_putc - Write a character
- f_puts - Write a string
- f_printf - Write a formatted string
- f_tell - Get current read/write pointer
- f_eof - Test for end-of-file
- f_size - Get size
- f_error - Test for an error

(2) Directory Access

- f_opendir - Open a directory
- f_closedir - Close an open directory
- f_readdir - Read an directory item
- f_findfirst - Open a directory and read the first item matched
- f_findnext - Read a next item matched

(3) File and Directory Management

- f_stat - Check existance of a file or sub-directory
- f_unlink - Remove a file or sub-directory
- f_rename - Rename/Move a file or sub-directory

- f_chmod - Change attribute of a file or sub-directory
- f_utime - Change timestamp of a file or sub-directory
- f_mkdir - Create a sub-directory
- f_chdir - Change current directory
- f_chdrive - Change current drive
- f_getcwd - Retrieve the current directory and drive

(4) Volume Management and System Configuration

- f_mount - Register/Unregister the work area of the volume
- f_mkfs - Create an FAT volume on the logical drive
- f_fdisk - Create logical drives on the physical drive
- f_getfree - Get total size and free size on the volume
- f_getlabel - Get volume label
- f_setlabel - Set volume label
- f_setcp - Set active code page

More details about the usage of FATFS API, please visit http://elm-chan.org/fsw/ff/00index_e.html

12.1.3 FATFS Example

The example for NOR flash and RAM filesystem need to be assigned the start address to run:

In FLASH_FATFS.c

```
define FLASH_APP_BASE                  0x180000
#define FLASH_BLOCK_SIZE               512
#define FLASH_SECTOR_COUNT             256
#define SECTOR_SIZE_FLASH              4096
```

In FATFS_RAMDISK_API.c

```
#define RAM_DISK_SZIE                  1024*1024*10
#define SECTOR_SIZE_RAM                 512
#define SECTOR_COUNT_RAM               (RAM_DISK_SZIE/512)
```

Please execute the example_fatfs.c to run the example.

12.1.4 FATFS Behavior Description

In this example, we demonstrate how to use FATFS on AmebaPro2 flash memory and manage files and directories in the file system.

First, we use FATFS API to register flash disk driver and get a drive number for the flash drive. We use this drive number as its path and mount to a FATFS object.

Next, the example list files currently exist in the flash memory, clear all files and directories, and list files again to check if the drive is all clean and empty.

Next, the example uses f_mkdir API to create a directory named “ameba_dir” in the root of the filesystem and use f_open to create a file named “ameba_dir_file” in ameba_dir. Then list files to show the created directory and file.

Next, we create a file named “ameba_root_file” at the root of the drive, and use f_write API to try to write some content to the file. Then use f_read API to read from the file to check if the content written to the file can be read back correctly.

Finally we list all files and directories in the drive.

12.1.5 Dual Fat File system (File system on both SD Card and Flash)

Please modify the example_fatfs.h to enable the SD and FLASH function.

```
#define CONFIG_FATFS_IF_SD          1
#define CONFIG_FATFS_IF_FLASH       1
```

In this example, we demonstrate how to use FATFS on both AmebaPro2 flash memory and SD card, and manage files and directories in the two filesystems.

First, we use FATFS API to register flash disk driver and SD disk driver, and each drive gets a drive number. We use the drive number as drive path and mount flash drive and SD drive, each with a FATFS object.

Next, the example clears files currently exist in both drives, and list files again to check if the drives are all clean and empty.

Next, the example tests operations on the SD drive. We create a new file("sd_file") and perform read/write to the file, then create a new directory("sd_dir") and open a new file in the directory("sd_file2").

Next, the example tests similar operations on the flash drive. Create a new file("flash_file") and perform read/write to the file. Then we create a new directory("flash_dir") and open a new file in the directory("flash_file2").

Finally we list all files and directories in each drive.

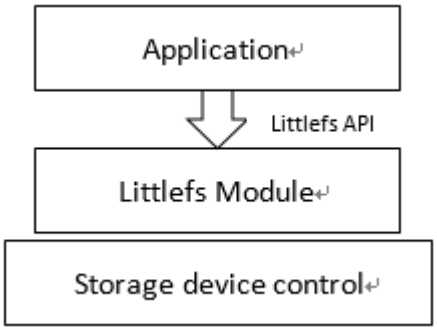
12.2 LITTLEFS File System

The little file system (LittleFS) is a fail-safe file system designed for embedded systems, specifically for microcontrollers that use external flash storage.

There are three challenges for embedded storage on microcontrollers and flash storage: power loss, wear and limited RAM and ROM. This file system provides the solution to these challenges.

- Bounded RAM/ROM - This file system works with a limited amount of memory. It avoids recursion and limits dynamic memory to configurable buffers that can be provided statically.
- Power-loss resilient - We have designed this for operating systems that may have random power failures. It has strong copy-on-write guarantees and keeps storage on disk in a valid state.
- Wear leveling - Because the most common form of embedded storage is erodible flash memories, the file system provides a form of dynamic wear leveling for systems that cannot fit a full flash translation layer.

12.2.1 LITTLEFS Module



Ameba Pro2 utilizes Littlefs File system Module to provide access to low level storage devices. Applications can manage and operate the file system through Littlefs API.

12.2.2 LITTLEFS API

(1) File Access

- lfs_file_open - Open a file.
- lfs_file_opencfg - Open a file with extra configuration.
- lfs_file_close - Close a file.
- lfs_file_sync - Synchronize a file on storage.
- lfs_file_read - Read data from file.
- lfs_file_write - Write data to file
- lfs_file_seek - Change the position of the file.
- lfs_file_truncate - Truncates the size of the file to the specified size
- lfs_file_tell - Return the position of the file.
- lfs_file_rewind - Change the position of the file to the beginning of the file.
- lfs_file_size - Return the size of the file.

(1) Directory Access

- lfs_mkdir - Create a directory.
- lfs_dir_open - Open a directory.

- lfs_dir_close - Close a directory
- lfs_dir_read - Read an entry in the directory.
- lfs_dir_seek - Change the position of the directory.
- lfs_dir_tell - Return the position of the directory.
- lfs_dir_rewind - Change the position of the directory to the beginning of the directory.

(2) File and Directory Management

- lfs_remove - Removes a file or directory.
- lfs_rename - Rename or move a file or directory.
- lfs_stat - Find info about a file or directory.
- lfs_getattr - Get a custom attribute.
- lfs_setattr - Set custom attributes.
- lfs_removeattr - Removes a custom attribute
- lfs_fs_size - Finds the current size of the file system.
- lfs_fs_traverse - Traverse through all blocks in use by the file system.

(3) Volume Management and System Configuration

- lfs_format - Format a block device with the littlefs.
- lfs_mount - Mounts a littlefs.
- lfs_unmount - Unmounts a littlefs.
- lfs_migrate - Attempts to migrate a previous version of littlefs.

12.2.3 LITTLEFS Example

It can support the NOR and NAND flash. It depend on different file operation.
Please run the example_littlefs to run the example. The behavior is the same as fatfs example.
Both of them need to assign the start address and block size. Please reference the lfs_nor_api.c and lfs_nand_api.c to do the setup.

13 Audio optimization

The following chapters describe the software and hardware optimization solutions of AmebaPro2 audio.

13.1 Audio setting

13.1.1 Gain setting

13.1.1.1 Analog microphone gain setting

The audio analog input gain can be namely divided into analog gain and digital gain.

- Analog mic gain

It supports 0, 20, 30, 40 dB for the gain optimization.
User can use audio_mic_analog_gain or set the parameter mic_gain for audio module to set it.

- ADC gain can be used to set the input (analog to) digital gain – ADC Volume.

The range is -17.625dB (0x00) ~ 30dB (0x7F)

User can use the function audio_adc_digital_vol or use CMD_AUDIO_SET_ADC_GAIN to control audio module to use the function.
A digital gain configuration is offered to control the audio output gain. Customers can set a reasonable gain value via DAC Volume to obtain the appropriate audio output volume according to their needs. Basically setting the gain to 0dB (0xAF), the output amplitude will meet the board audio output volume requirements. Note that a sound breakage will happen when the output gain is setting too large.

If the analog gain is too large, analog gain will affect the sound effect and noise will be obvious.
Recommend that customers can first configure the digital gain. If the audio signal gain need to increase but the digital gain achieves the maximum range, then configure the analog gain.

13.1.1.2 Digital microphone gain setting

- Left mic gain

Left dmic gain and it supports 0, 12, 24, 36 dB for the gain optimization.
User can use audio_l_dmic_gain or set the parameter dmic_l_gain for audio module to set it.

- Right mic gain

Right dmic gain and it supports 0, 12, 24, 36 dB for the gain optimization.
User can use audio_r_dmic_gain or set the parameter dmic_r_gain for audio module to set it.

13.1.1.3 Speaker gain setting

- DAC gain can be used to set the output digital (to analog) gain – DAC Volume.

The range is -65.625dB (0x00) ~ 0dB (0xAF)
User can use the function audio_dac_digital_vol or use CMD_AUDIO_SET_DAC_GAIN to control audio module to use the function.

13.1.2 HPF setting

In AmebaPro2, it provides a high pass filter for user to filter the low frequency noise.
Here is the function:

```
void audio_adc_l_hpf(audio_t *obj, BOOL en, audio_hpf_fc hpf_fc);
```

The parameters mean:

- obj: Audio object define in application software.
- en: enable the high pass frequency or not.
- hpf_fc: set the cutoff frequency, the value is from 0~7; fc ~= 5e-3 / (hpf_fc + 1) * fs.

13.1.3 EQ setting

In AmebaPro2, it also provides five sets of biquad filters in three sides for left digital mic (analog mic), right digital mic and audio output.
One biquad filter can switch to high-pass, low-pass, band-pass, notch, peak, low shelf, and high shelf filter by register settings.
Here are some tips for user to use the EQ:

- Select the biquad filter

User can use the following websites to configure the preferred filter type, sample rate, cutoff frequency, Q value and Gain first:
https://www.earlevel.com/main/2021/09/02/biquad-calculator-v3/

- Get the registers' value of selecting filter

User can use AmebaPro2_EQ_tool.exe to generate register settings. For example, if we choose a high pass filter with cutoff frequency 200Hz and Q value 0.707, user can type the setting and get the registers' value (0x1e45618, 0x1c000000, 0x2000000, 0x3c72d61, 0x1e35d500) for this setting.

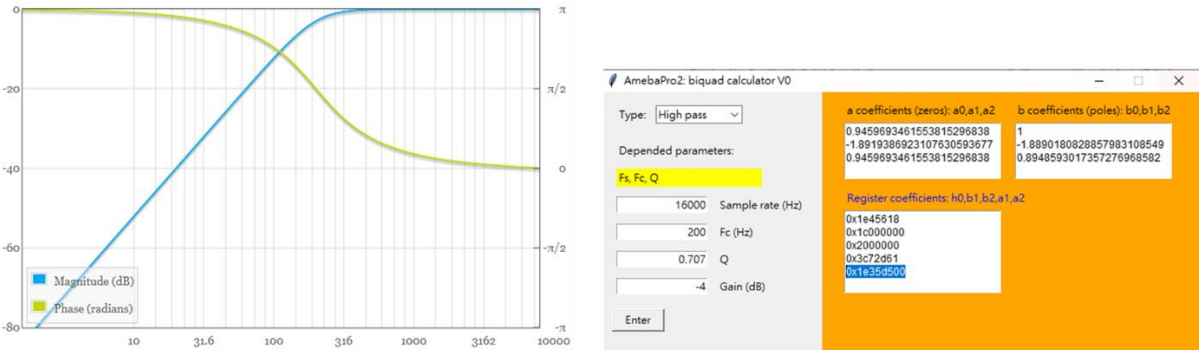


Figure 13-1 EQ setting

- Set the register value got from AmebaPro2_EQ_tool.exe

After getting the registers' value, user can use the following function to apply the filter setting on AmebaPro2 on left digital mic (analog mic), right digital mic and audio output. Note that there are 5 filters could be set in each side.

```
void audio input l eq(audio t *obj, audio eq eq, BOOL en, u32 h0, u32 b0, u32 b1, u32 a0,
u32 a1);
void audio input r eq(audio t *obj, audio eq eq, BOOL en, u32 h0, u32 b0, u32 b1, u32 a0,
u32 a1);
void audio_output_l_eq(audio_t *obj, audio_eq eq, BOOL en, u32 h0, u32 b0, u32 b1, u32 a0,
u32 a1);
```

Here are the parameters:

- obj: Audio object define in application software.
- eq: Select the EQ number, can be 0~4.
- en: enable the eq filter or not
- h0, b0, b1, a0, a1: the registers' value gotten from AmebaPro2_EQ_tool.exe.

13.1.4 Other setting

Here are some commands about the module audio setting:

- CMD_AUDIO_SET_RESET

will be re-initialize the audio setting and also the ASP algorithms. If you do some changes need to reset the audio configuration, like change the sample rate, reset the audio to switch the configuration.

- CMD_AUDIO_SET_SAMPLERATE

can set the sample rate. After using this command, a reset is needed to apply the sample rate configuration on audio and ASP algorithms.

NOTE

If using audio codec, be sure the sample rate is fitting the sample rate used in audio codec.

- CMD_AUDIO_SET_TRX

provide a way to stop and re-start the audio without re-initialize the audio system and ASP algorithms. Set 0 to stop the tx and rx progresses or 1 to start them.

13.2 Audio ASP algorithm

The following table shows some common audio problem with their causes and also the adjustment using ASP algorithm.

Situation	Algorithm	Influence End	Cases
Distortion	AGC	transmitting end	● The ambient sound is too high ● Headphone preGain ● Compression_gain_db of AGC is too large
Low audio volume	AGC	transmitting end	● The original input volume is too low ● Compression_gain_db of AGC is too small ● AGC is not working properly
Echo or howling	AEC	transmitting end	● Too close between transmitting and receiving end device ● Volume too large or mic too sensitive

			● AEC is not turn on ● AEC parameters is not setting correctly (frame_size, sample rate, set_sndcard_delay_ms)
Intermittent voice	AEC、NS	transmitting end	● NS or AEC suppression
Noise floor	NS	transmitting end	● NS mode setting too low ● Caused by the environment, NS can't do well
Mechanical sound	Network、Device	Receiving end	● Poor network environment ● Device sampling is unstable or device hardware problem

13.2.1 Open ASP algorithm

For using ASP algorithm, user need to turn off the new library by –DBUILD_CT=off. For example, cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake –DAUDIO_TEST_TOOL=on.
The codes and functions related to the ASP algorithm are shows in the table.
Enable ENABLE_ASP in module_audio.h and use the 3A (AGC: Automatic gain control; ANS: Adaptive noise suppression; AEC: Acoustic echo cancellation) algorithms to obtain better audio effects.

- The parameters, sample_rate and mic_gain, and the initialization of NS, AEC, AGC and other algorithms will be setting at CMD_AUDIO_APPLY and CMD_AUDIO_SET_RESET.

To enable ASP function user can use the following parameters in ASP.h:

```
=================== Open ASP algorithm (ASP.h) ================
typedef struct CTNS_cfg_s {
 int16_t NS_EN;
 int NSLevel;

 int16_t Reserve1;
 int16_t Reserve2;
 int16_t Reserve3;
} CTNS_cfg_t;

typedef struct CTAGC_cfg_s {
 int16_t AGC_EN;
 CT_AGC_MODE AGCMode;
 int16_t ReferenceLvl;
 int16_t RefThreshold;
 int16_t AttackTime;
 int16_t ReleaseTime;
 int16_t Ratio[3];
 int16_t Threshold[3]; // Threshold1, Threshold2, NoiseGateLvl
 int16_t KneeWidth;

 int16_t Reserve1;
 int16_t Reserve2;
 int16_t Reserve3;
} CTAGC_cfg_t;

typedef struct CTAEC_cfg_s {
 int16_t AEC_EN;
 int16_t EchoTailLen;
 int16_t CNGEnable;
 int16_t PPLevel;

 int16_t Reserve1;
 int16_t Reserve2;
 int16_t Reserve3;
} CTAEC_cfg_t;
```

The following is detail of the parameters in each configuration

configuration	Parameters
CTAEC_cfg_t	● AEC_EN: enable the AEC module in AEC process ● EchoTailLen: setting the echo tail length for the AEC ● CNGEnable: enable the comfortable noise generation in AEC

	● PPLevel: adjust the AEC aggressive level, support 1~18
CTAGC_cfg_t	● AGC_EN: enable the AGC module in the AGC process ● AGCMode: set the AGC mode for AGC module in the AGC process, the value is 0 (CT_ALC), 1 (CT_DRC), 2 (CT_LIMITER) ● ReferenceLvl: the output target reference level (dBFS) support 0,1,...,30 (0,-1,...,-30dBFs) ● RefThreshold: the threshold limits the output to this level, only support in **CT_DRC** and **CT_LIMITER**, support 0,1...,30 (0,-1...,-30dBFs) ● Ratio[3]: support three ratios for adjusting the AGC gain curve ● Threshold[3]: support three thresholds for adjusting the AGC gain curve ● KneeWidth: adjust the soft knee of the AGC gain curve
CTNS_cfg_t	● NS_EN: enable the NS module in the NS process ● NSLevel: set the aggressive level (the larger the more aggressive) for NS module in the NS process, the value is from 3~15

In "ASP.h" it defined some function for the ASP setting. The following tables hows the functions for setting the ASP algorithm:

Function	Related module	Parameters	Note
AEC_init	NS, AEC, AGC	● frame_size: setting the frame size for the AEC module, the unit is "sample" ● sample_freq: audio sample rate (support 8k and 16k) ● RX_AEC: the pointer for the new library AEC setting in mic path ● RX_AGC: the pointer for the new library AGC setting in mic path ● RX_NS: the pointer for the new library NS setting in mic path ● snd_amplification: set the amplification for the output result	● For mic side ASP ● The AEC process include NS, AEC and AGC. ● If use AEC process, additional NS and AGC process is no need in mic side
AEC_set_level	AEC	● level: the aggressive level (the larger the more aggressive) for AEC module, the level is from 0~4 ● RX_AEC: the pointer for the new library AEC setting	● For mic side ASP
NS_set_level_for_AEC	NS	● Level: the NS level for RX path ● RX_NS: the pointer for the new library NS setting	● For mic side ASP
AEC_process	NS, AEC, AGC	● farend: the array input for the far-end data ● nearend: the array input for the far-end data ● out: the space to save the ASP processed data	● For mic side ASP
AEC_destory	NS, AEC, AGC		● For mic side ASP ● For destroy the modules initialed in AEC_init
AGC_init	AGC	● sample_freq: audio sample rate (support 8k and 16k) ● TX_AGC: the pointer for the webrtc AGC setting in speaker path	● For output (speaker) side ASP
AGC_process	AGC	● frame_size: setting the frame size for the AGC module, the unit is "sample" ● out: the data will be used to do AGC process, the data will directly be modified	● For output (speaker) side ASP
AGC_destory	AGC		● For output (speaker) side ASP ● For destroy the modules creates in AGC_init
NS_init	NS	● sample_freq: audio sample rate (support 8k and 16k) ● TX_NS: the pointer for the webrtc NS setting in speaker path	● For output (speaker) side ASP
NS_process	NS	● frame_size: setting the frame size for the AGC module, the unit is "sample"	● For output (speaker) side ASP

		• out: the data will be used to do AGC process, the data will directly be modified	
NS_destory	NS		• For output (speaker) side ASP • For destroy the modules creates in NS_init

13.2.1.1 ASP algorithm usage

Here are the configurations for ASP algorithm:

- 8K and 16K audio sample rate are supported in the ASP algorithms.
- The default ASP settings - default_rx_asp_params and default_tx_asp_params are defined in module_audio.c.
- Users can use CMD_AUDIO_GET_RXASP_PARAM and CMD_AUDIO_GET_TXASP_PARAM to get the ASP parameters for RX and TX ASP parameters in the audio module.
- Users can use CMD_AUDIO_SET_RXASP_PARAM and CMD_AUDIO_SET_TXASP_PARAM to set the ASP parameters for RX and TX ASP parameters.
- When default_rx_asp_params.agc_cfg.AGC_EN and default_tx_asp_params.agc_cfg.AGC_EN set 0 which means disable the AGC process in RX and TX direction, while 1 means enabling the AGC process. When default_rx_asp_params.ns_cfg.NS_EN and default_tx_asp_params.ns_cfg.NS_EN set 0 which means disable the NS process in RX and TX direction, while 1 means enabling the NS process.
- When default_rx_asp_params.aec_cfg.AEC_EN set 0 which means disable the AEC process in RX direction, while 1 means enabling the AEC process.

13.2.1.2 AEC setting

The AEC algorithm includes three parts: delay adjustment strategy, linear echo estimation, and nonlinear echo suppression.

- Use CMD_AUDIO_RUN_AEC to dynamically switch the use of AEC_process().
- Use CMD_AUDIO_SET_AEC_ENABLE to determine whether AEC_init() is enabled during audio reset.
- CMD_AUDIO_SET_AEC_LEVEL can set the strength of cancellation.

13.2.1.3 NS setting

The NS algorithm is aimed at decrease the noise or environment sound, so it is recommended to use before other ASP algorithms .

- Use CMD_AUDIO_SET_NS_ENABLE to determine whether NSx_init() is enabled during audio reset.
- Use CMD_AUDIO_RUN_NS to dynamically switch the use of NSx_process().

13.2.1.4 AGC setting

The AGC algorithm is used to balance the audio volume of signal streaming.

- Use CMD_AUDIO_SET_AGC_ENABLE to determine whether AGC_init() is enabled during audio reset.
- Use CMD_AUDIO_RUN_AGC can dynamically switch the use of AGC_process().

13.3 Audio test tool

Ameba Pro2 provide an example for audio testing.
User can use the following steps to build up the audio test tool image

- Step1: cd project\realtek_amebapro2_v0_example\GCC-RELEASE
- Step2: mkdir build
- Step3: cd build
- Step4: cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake –DAUDIO_TEST_TOOL=on
- Step5: cmake --build . --target flash

The following shows the command of the test tool

- Common command

command	parameters	description

AUMMODE	● [mic_mode]= amic/l_dmic/r_dmic/stereo_dmic	Set up the microphone type
AUMG	● [mic_gain]=0~3	Set up the analog mic gain (0: 0dB, 1: 20dB, 2: 30dB, 3:40dB)
AUMB	● [mic_bias]=0~2	Set up the mic bias (0: 0.9, 1: 0.86, 2: 0.75)
AUMLG	● [left_dmic_gain]=0~3	Set up the left digital mic (0: 0dB, 1: 12dB, 2: 24dB, 3:36dB)
AUMRG	● [right_dmic_gain]=0~3	Set up the right digital mic (0: 0dB, 1: 12dB, 2: 24dB, 3:36dB)
AUADC	● [ADC_gain]=0x00~0x7F	Set up audio input digital gain level, the gain level is up 0.375dB/step. The max and min gains are 30dB and -17.625dB.
AUHPF	● [cutoff num]=0~8	Set up the mic first HPF cutoff frequency. The cutoff frequency is equal to 5e-3 / (cutoff num + 1) * fs (sample rate frequency).
AUMLEQ	● [eq num]=0~4 ● [register h0],[register a1],[register a2],[register b1],[register b2]	Set up the EQ for analog or left digital microphone (PDM rising trigger). There are 5 EQs ([eq num]) can be used (the EQ0 is used for a HPF default). The register setting can generate by AmebaPro2_EQ_tool.exe.
AUMREQ	● [eq num]=0~4 ● [register h0],[register a1],[register a2],[register b1],[register b2]	Set up the EQ for analog or right digital microphone (PDM falling trigger). There are 5 EQs ([eq num]) can be used (the EQ0 is used for a HPF default). The register setting can generate by AmebaPro2_EQ_tool.exe.
AUMICM	● [enable_mute]=0 or 1	Mute MIC or not
AUSPEQ	● [eq num]=0~4 ● [register h0],[register a1],[register a2],[register b1],[register b2]	Set up the EQ for audio output. There are 5 EQs ([eq num]) can be used (the EQ0 is used for a HPF default). The register setting can generate by AmebaPro2_EQ_tool.exe.
AUDAC	● [DAC_gain]=0x00~0xAF	Set up the audio output digital gain, the gain level is up 0.375dB/step. The max and min gains are 0dB and -65.625dB.
AUSPM	● [enable_mute]=0 or 1	Mute the audio output ([enable_mute]=1) or unmute the audio output ([enable_mute]=0).
TONEDBSW	● [sweep_DB_interval(ms)]	Enable the dB sweep when play tone
AUTXMODE	● [tx_mode]= noplay/playback/playtone/playmusic ● [audio_tone(Hz)]	Set up the audio output mode, there are four modes supported now. The noplay mode will stop sending data to audio output .The playmusic mode will start to play the music (support 8k or 16k). The playback mode will audio input will directly send to audio output. The audio tone mode will start playing the audio tone setting by [audio_tone(Hz)].
AUAMPIN	● [pin_name]=pin name ● [on/off]=1/0	Set up or down the audio amplifier pin.
AUSR	● [sample_rate]=8000, 16000, 32000, 44100, 48000, 88200, 96000	Set up the audio input and output sample rate.
AUTRX	● [enable]=0 or 1	Set down or up the audio input and output.
AURES	● [reset_enable]	Reset the audio module to enable the previous audio setting.
AUFFTS	● [FFT_EN]=0 or 1	Enable print audio input FFT result, but the play back mode is not supported.
AUFILE	● [filename]	Set up the audio save file name. The legth
AUCOPY	● [mode]=NOCOPY, SD, TFTP[enable_sd_download]=0 or 1 ● [tftp_ip],[tftp_port]: set TFTP server IP and port	Set the audio file will be copied to other storage place after each record. NOCOPY: just save in RAM TFTP: copy through tftp server SD: copy to the SD cardEnable the save file download to SD card or not. If enabling the SD download, device will copy the save data to SD card after each record.
AUREC	● [record_time]	Start record the audio data for record time. RECORD_TYPE can select to save RX (audio input), TX (audio output before ASP) and ASP (audio input after ASP), TXASP(audio output after ASP).

	• [RECORD_TYPE1], [RECORD_TYPE2], [RECORD_TYPE3], [RECORD_TYPE4]=RX,TX,ASP,TXASP	
AUMSGS	• [MSGLevel]=0,1,2,3	set the audio message show level • 0: no message • 1: inf, warn and err • 2: warn, err • 3: err

- ASP command using new library

command	parameters	description
AUAEC	• [enable]=0 or 1 • [level]=1~18 • [Tailength]=32, 64, 128 • [CNG_enasble]=0 or 1	Open the AEC or not. [level]: larger level for more aggressive echo cancelation, but may erase more local speech.
AUNS	• [NS_enable]= 0 or 1 • [NS_level]=3~15	Set up the noise suppression level for audio input. The larger level is, the more aggressive NS process. Close the NS module by setting [NS_ enable] to 0.
AUAGC	• [AGC_enable]=0 or 1 • [AGC_mode]=0~2 • [AGC_referenceLvl]=0,1,..,30 • [AGC_ref_threshold]=0,1,..,30 • [AGC_AttackTime]=20~1000 • [AGC_ReleaseTime]=20~1000 • [AGC_Ratio1]=1,2,3….,50 • [AGC_Ratio2]=1,2,3….,50 • [AGC_Ratio3]=1,2,3….,50 • [AGC_Threshold1]=0,1,…,40 • [AGC_Threshold2]=0,1,…,40 • [AGC_NoiseGateLvl]=40,41,…,100 • [AGC_KneeWidth]=0,1,2,…,10	Set up the auto gain control, • [AGC_enable] enable the rx path (mic path) AGC • [AGC_mode] can select the AGC mode. Support three modes, CT_ALC, CT_DRC and CT_LIMITER • [AGC_referenceLvl] is output target reference level (dBFS), support 0,1,..,30 (0,-1,…,-30dBFs) • [AGC_ref_threshold] limits the output to this level, only support in CT_DRC and CT_LIMITER, support 0,1,…,30 (0,-1,…,-30dBFs) • [AGC_AttackTime] is the transition time of changes to signal amplitude compression, 20~1000 • [AGC_ReleaseTime] is the transition time of changes to signal amplitude boost, 20~1000 • [AGC_Ratio1] is the compression ratio for ReferenceLvl/RefThreshold, 1,2,3….,50 (slope 1,1/2,1/3….,1/50) • [AGC_Ratio2] is the compression ratio for Threshold1, 1,2,3….,50 (slope 1,1/2,1/3….,1/50) • [AGC_Ratio3] is the compression ratio for Threshold2, 1,2,3….,50 (slope 1,1/2,1/3….,1/50) • [AGC_Threshold1] is the parameter determines the second knee of the curve, 0,1,…,40 (0,-1,…,-40dBFs) • [AGC_Threshold2] is the parameter determines the third knee of the curve, 0,1…,40 (0,-1,…,-40dBFs) • [AGC_NoiseGateLvl] is the noise floor level, 40,41,…,100 (-40,-41,…,-100dBFs) • [AGC_KneeWidth] is the knee width, 0,1,2,…,10 (0,1,2,…,10dBFs)
AUSPNS	• [NS_enable]= 0 or 1 • [NS_level]=3~15	Set up the noise suppression level for audio output. The larger level is, the more aggressive NS process. Close the NS module by setting [NS_ enable] to 0.
AUSPAGC	• [AGC_enable]=0 or 1 • [AGC_mode]=0~2	Set up the auto gain control, • [AGC_enable] enable the rx path (mic path) AGC

	● [AGC_referenceLvl]=0,1,..,30 ● [AGC_ref_threshold]=0,1,..,30 ● [AGC_AttackTime]=20~1000 ● [AGC_ReleaseTime]=20~1000 ● [AGC_Ratio1]=1,2,3….,50 ● [AGC_Ratio2]=1,2,3….,50 ● [AGC_Ratio3]=1,2,3….,50 ● [AGC_Threshold1]=0,1,…,40 ● [AGC_Threshold2]=0,1,…,40 ● [AGC_NoiseGateLvl]=40,41,…,100 ● [AGC_KneeWidth]=0,1,2,…,10	● [AGC_mode] can select the AGC mode. Support three modes, CT_ALC, CT_DRC and CT_LIMITER ● [AGC_referenceLvl] is output target reference level (dBFS), support 0,1,..,30 (0,-1,…,-30dBFs) ● [AGC_ref_threshold] limits the output to this level, only support in CT_DRC and CT_LIMITER, support 0,1,…,30 (0,-1,…,-30dBFs) ● [AGC_AttackTime] is the transition time of changes to signal amplitude compression, 20~1000 ● [AGC_ReleaseTime] is the transition time of changes to signal amplitude boost, 20~1000 ● [AGC_Ratio1] is the compression ratio for ReferenceLvl/RefThreshold, 1,2,3….,50 (slope 1,1/2,1/3….,1/50) ● [AGC_Ratio2] is the compression ratio for Threshold1, 1,2,3….,50 (slope 1,1/2,1/3….,1/50) ● [AGC_Ratio3] is the compression ratio for Threshold2, 1,2,3….,50 (slope 1,1/2,1/3….,1/50) ● [AGC_Threshold1] is the parameter determines the second knee of the curve, 0,1,…,40 (0,-1,…,-40dBFs) ● [AGC_Threshold2] is the parameter determines the third knee of the curve, 0,1…,40 (0,-1,…,-40dBFs) ● [AGC_NoiseGateLvl] is the noise floor level, 40,41,…,100 (-40,-41,…,-100dBFs) [AGC_KneeWidth] is the knee width, 0,1,2,…,10 (0,1,2,…,10dBFs)

14 NN

Ameba Pro2 has an NN H/W engine to accelerate the neural network inference process. NN models obtained from different AI framework, such as Keras, Tensorflow, Tensorflow Lite, PyTorch, Caffe, ONNX, Darknet and etc, can be converted to network binary graph file and be deployed on Ameba Pro2 easily.

14.1 NN module

The NN mmf module – vipnn is provided to process the input RGB frame from video module, and do NN inference. Then, NN inference result will be stored in network output tensor. Since the output tensor format of each model are different, vipnn module will also do the post-process work for the NN output tensor to extract the information and convert to understandable message. The pre-process and post-process function can also be registered with their customized model.

14.1.1 VIPNN module

The context of the video module shows as following:

```
typedef struct vipnn_ctx_s {
    void *parent;
    vip_network network;
    vip_buffer_create_params_t vip_param_in[MAX_IO_NUM];
    vip_buffer_create_params_t vip_param_out[MAX_IO_NUM];
    vip_buffer input_buffers[MAX_IO_NUM];
    vip_buffer output_buffers[MAX_IO_NUM];
    vipnn_params_t params;
    vipnn_status_t status;
    char network_name[64];
    int input_count;
    int output_count;
    vipnn_preproc_t pre_process;
    vipnn_postproc_t post_process;
    disp_postprcess_t disp_postproc;
    vipnn_cascaded_mode_t cas_mode;
    bool module_out_en;
    vipnn_measure_t measure;
} vipnn_ctx_t;
```

Description of parameter in vipnn_ctx_t:

- network: an opaque handle to the new network object if the request is executed successfully.
- vip_param_in: parameter of network input tensor.
- vip_param_out: parameter of network output tensor.
- input_buffers: buffer for model input tensor.
- output_buffers: buffer for model output tensor.
- params: basic parameters for the vipnn module.
- status: record status of vipnn module.
- network_name: nn network name.
- input_count: the number of input tensor in the NN network.
- output_count: the number of output tensor in the NN network.
- pre_process: pre-process function for processing the data before passing to NN inference.
- post_process: post-process function for decoding the data from NN inference.
- disp_postproc: Set the callback function for display the NN result on video frame. It could be set by using CMD_VIPNN_SET_DISPPOST.
- module_out_en: enable module output.
- measure: time measurement

14.1.1.1 Basic vipnn module parameters setting

Here are some vipnn module parameters provided to set.

```
typedef struct vipnn_param_s {
    int model_type;
    char model_file[64];
    uint8_t *model_mem;
    uint32_t model_size;
    int fps;
    int in_width, in_height;
```

```
    rect_t roi;
    int m_width, m_height;        // should read from model, not user setting
    nn_data_param_t *in_param;
    nnmodel_t *model;
} vipnn_params_t;
...
nn_data_param_t in_param = {
    .img = {
        .width = NN_WIDTH,
        .height = NN_HEIGHT,
        .rgb = 0,
        .roi = {
            .xmin = 0,
            .ymin = 0,
            .xmax = NN_WIDTH,
            .ymax = NN_HEIGHT,
        }
    },
    .codec_type = AV_CODEC_ID_RGB888
};
```

Use CMD_VIPNN_SET_IN_PARAMS to set up the NN input parameters.

- img.width: input frame width.
- img.height: input frame height.
- img.rgb: reserved, do not care.
- img.roi: ROI of input frame. Usually, we set it to (0,0,NN_WIDTH, NN_HEIGHT).
- codec_type: model input type, could be AV_CODEC_ID_RGB888, AV_CODEC_ID_NV12, AV_CODEC_ID_NN_RAW.

NOTE

Model input codec type should be matched with media source output codec type.

14.1.1.2 Set NN model to vipnn module

Use CMD_VIPNN_SET_MODEL to set up the NN model:

```
vipnn_ctx = mm_module_open(&vipnn_module);
if (vipnn_ctx) {
    …
    mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_MODEL, (int)&yolov4_tiny);
    …
}
```

14.1.1.3 Set NN result display callback function

User can register a call back function to so display the NN result or do their own customized additional post-processing. Use CMD_VIPNN_SET_DISPPOST to set up callback function for display the NN result:

```
static void nn_result_display (void *p, void *img_param)
{
    objdetect_res_t *res = (objdetect_res_t *)p;
    nn_data_param_t *im = (nn_data_param_t *)img_param;

    /* Process or display the result here */
}
…
…
vipnn_ctx = mm_module_open(&vipnn_module);
if (vipnn_ctx) {
    …
    mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_DISPPOST, (int)nn_result_display);
    …
}
```

14.1.1.4 Set NN object/face detection threshold

There are two threshold values related to NN post-processing result – confidence & NMS threshold.
Confidence is the score of the bounding box. Use CMD_VIPNN_SET_SCORE_THRES to set up confidence score threshold:

```
static float nn_confidence_thresh = 0.5;
mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_CONFIDENCE_THRES, (int)&nn_confidence_thresh);
```

For the same class, if the IOU (Intersection over union) of two bounding box larger then NMS threshold, these two objects will be considered the same object. Use CMD_VIPNN_SET_NMS_THRES to set up NMS threshold:

```
static float nn_nms_thresh = 0.3;
mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_NMS_THRES, (int)&nn_nms_thresh);
```

14.2 Model Zoo

Currently, the SDK provides several deployed models. They are listed in following table:

Table 14-1 Pro2 model list

Category	Model	Description
Object detection	Yolov3-tiny Yolov4-tiny Yolov7-tiny	https://github.com/AlexeyAB/darknet
Face detection	SCRFD	https://github.com/deepinsight/insightface/tree/master/detection/scrfd
Face Recognition	MobileFaceNet	https://github.com/deepinsight/insightface/tree/master/recognition
Sound classification	YAMNet	https://github.com/tensorflow/models/tree/master/research/audioset/yamnet

14.2.1 Object detection model

SDK provides object detection model for user to evaluate – Yolov3-tiny, Yolov4-tiny and Yolov7-tiny.

14.2.1.1 Yolo series model

YOLO (you only look once) is a neural network algorithm for object detection, implemented with darknet architecture. Yolo is well-known for its lightweight, less dependent and efficient in algorithms.
For more information, see Yolo's Github maintain by its authors: https://github.com/AlexeyAB/darknet

14.2.2 Face detection model

Currently, the SDK provide a face detection model for user to evaluate – SCRFD.

14.2.2.1 SCRFD

SCRFD(Sample and Computation Redistribution for Efficient Face Detection) is an efficient high accuracy face detection approach.
For more information, see InsightFace official Github: https://github.com/deepinsight/insightface/tree/master/detection/scrfd

14.2.3 Face Recognition model

SDK provide a face recognition model for user to evaluate – MobileFaceNet(ArcFace).

14.2.3.1 MobileFaceNet with ArcFace

MobileFaceNet is a face recognition model trained with ArcFace (Additive Angular Margin Loss).
For more information, see InsightFace official Github: https://github.com/deepinsight/insightface/tree/master/recognition

14.2.4 Sound classification model

A pre-trained sound classification model is provided in SDK – YAMNet.

14.2.4.1 YAMNet

YAMNet is a model that can predicts 521 audio event classes based on the AudioSet.
For more information, see TensorFlow official Github:
https://github.com/tensorflow/models/tree/master/research/audioset/yamnet

- yamnet_fp16: official model used to predict 521 sounds

- yamnet_s: RTK self-trained model to predict 2 alarm sounds — CO, Smoke

14.3 NN result format

After NN model inference, the inference result will be store in NN output tensor. These output tensors should be decoded in post-processing. vipnn_res_t structure is used to store the post-processing result:

```
typedef struct vipnn_res_s {
    union {
        objdetect_res_t od_res;          // for object detection
        facedetect_res_t fd_res;        // for face detection
        face_feature_res_t frec_res;       // for face recognition
    };
    int type;
} vipnn_res_t;
```

Object detection

For the object detection result, the post-processing will parse the object position and probability from the output tensor, and fill the results to an objdetect_res_t structure:

```
#define MAX_DETECT_OBJ_NUM 128
typedef struct objdetect res s {
    int obj_num;
    union {
        float result[MAX_DETECT_OBJ_NUM * 6];
        detobj_t res[MAX_DETECT_OBJ_NUM];
    };
} objdetect_res_t;
```

Description of parameter in objdetect_res_t:

- obj_num: indicate the number of object detected in current frame.
- result: record the class_index, probability and bounding box position for each object as format in following figure.
  - c: class_index
  - p: probability
  - tx, ty, bx, by: bounding box(top_x, top_y, bottom_x, bottom_y)

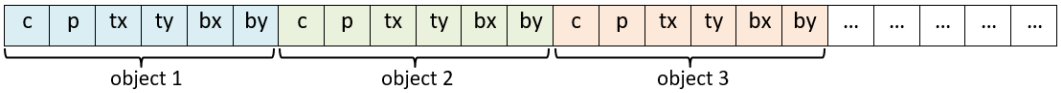


Figure 14-1 object detection format

Face detection

For the face detection result, the post-processing will parse the face position, score and landmarks from the output tensor, and fill the results to facedetect_res_t structure:

```
typedef struct facedetect_res_s {
    int obj_num;
    union {
        float result[MAX_DETECT_OBJ_NUM * 6];
        detobj_t res[MAX_DETECT_OBJ_NUM];
    };
    landmark_t landmark[MAX_DETECT_OBJ_NUM];
} facedetect_res_t;
```

Face recognition

For the face recognition result, the post-processing will decode the face feature, and fill the results to face_feature_res_t structure:

```
#define MAX FACE FEATURE DIM 128
typedef struct face_feature_res_s {
    float result[MAX_FACE_FEATURE_DIM];
} face_feature_res_t;
```

14.4 NN model preparation

NN model should be prepared before using the NN example

14.4.1 Using existing NN model in SDK

There are several existing NN model binary files provided in SDK:

- yolov3_tiny.nb (416x416), yolov4_tiny.nb (416x416), yolov7_tiny.nb (416x416), yolov4_tiny_576x320.nb
- yamnet_fp16.nb, yamnet_s.nb
- scrfd_500m_bnkps_576x320_u8.nb, scrfd_500m_bnkps_640x640_u8.nb
- mobilefacenet_int8.nb (112x112), mobilefacenet_int16.nb (112x112)

They are located in "project/realtek_amebapro2_v0_example/src/test_model".

NOTE

In face recognition application, it would be better to remain the image aspect ratio to get a better accuracy. If user want to run object detection and face detection/recognition at the same time, we can choose scrfd_500m_bnkps_576x320_u8.nb and yolov4_tiny_576x320.nb.

14.5 NN memory evaluation

This section shows how to evaluate NN model size and DDR usage. The following table shows the memory information of existing model provided in SDK:

Table 14-2 Model memory and size

Category	Model	Input size	Quantized	DDR memory	File size
Object detection	Yolov3-tiny	416x416	uint8	6.9 MB (6,946,128 bytes)	5.6 MB (5,568,384 bytes)
	Yolov4-tiny	416x416	uint8	7.7 MB (7,712,412 bytes)	4.1 MB (4,131,712 bytes)
	Yolov4-tiny	576x320	uint8	7.48 MB (7,840,836 bytes)	3.85 MB (4,043,136 bytes)
	Yolov7-tiny	416x416	uint8	8.2 MB (8,597,072 bytes)	4.44 MB (4,664,512 bytes)
Face detection	SCRFD	640x640	uint8	4.1 MB (4,291,200 bytes)	0.68 MB (715,584 bytes)
	SCRFD	576x320	uint8	2.6 MB (2,753,864 bytes)	0.56 MB (583,232 bytes)
Face Recognition	MobileFaceNet	112x112	int8	1.72 MB (1,799,716 bytes)	0.86 MB (904,576 bytes)
	MobileFaceNet	112x112	int16	5.1 MB (5,343,948 bytes)	3.42MB (3,590,656 bytes)
Sound classification	YAMNet	15600x1	fp16	9.2 MB (9,172,348 bytes)	8.7 MB (8,669,888 bytes)
	YAMNet_s	96x64	hybrid	0.73 MB (729,608 bytes)	0.67 MB (678,336 bytes)

14.5.1 Evaluate memory usage of model

Please refer the above table to evaluate the ddr memory usage of the model. Take yolov4-tiny for example, it requires at least 8MB ddr memory. Therefore, we have to make sure the NN ddr region in link script is enough for this model.

Check and modify in "project\realtek_amebapro2_v0_example\GCC-RELEASE\application\rtl8735b_ram.ld

```
/* DDR memory */

  VOE     (rwx)     : ORIGIN = 0x70000000, LENGTH = 0x70100000 - 0x70000000  /*  1MB */
  DDR     (rwx)     : ORIGIN = 0x70100000, LENGTH = 0x73000000 - 0x70100000  /* 49MB */
  NN      (rwx)     : ORIGIN = 0x73000000, LENGTH = 0x74000000 - 0x73000000  /* 16MB */
```

NOTE

Please also modify project/realtek_amebapro2_v0_example/GCC-RELEASE/bootloader/rtl8735b_boot_mp.ld to make the NN ddr region be consistent with rtl8735b_ram.ld. In addition, if building a TrustZone project, rtl8735b_ram_ns.ld should be modified instead of rtl8735b_ram.ld.

14.5.2 Evaluate model size

Please make sure the NN region in partition table is larger than your model size, so that the model can be downloaded to flash correctly.

Take yolov4-tiny for example, the model size is about 4MB

Figure 14-2 model network binary

The nn region length in “project\realtek_amebapro2_v0_example\GCC-RELEASE\mp\amebapro2_partitiontable.json” should not less than 4MB

```
"nn":{
            "start_addr" : "0x770000",
            "length" : "0x700000",
            "type": "PT_NN_MDL",
            "valid": true
      },
```

14.6 Using the NN MMF example with VIPNN module

The NN example is a part of mmf video joined example. Please uncomment the example want to execute.
(project/realtek_amebapro2_v0_example/src/mmfv2_video_example/video_example_media_framework.c)

```
mmf2_video_example_vipnn_rtsp_init();
//mmf2_video_example_vipnn_facedet_init();
//mmf2_video_example_face_rtsp_init();
//mmf2_video_example_joint_test_all_nn_rtsp_init();
//mmf2_video_example_joint_test_vipnn_rtsp_mp4_init;
//mmf2_video_example_audio_vipnn_init();
```

Current supported VIP NN examples

Table 14-3 NN examples

Example	Description	Result
mmf2_video_example_vipnn_rtsp_init	Video (H264/H265)-> RTSP (V1) Video (RGB) -> NN (V4)	(1) RTSP video stream over the network. (2) NN do object detection and draw the bounding box to RTSP channel.
mmf2_video_example_md_nn_rtsp_init	Video (H264/H265) -> RTSP (V1) Video (RGB) -> MD (V4)-> NN	(1) RTSP video stream over the network. (2) MD module detect motion. If there is motion detected, it will trigger NN module to detect object and draw the bounding box to RTSP channel.
mmf2_video_example_vipnn_facedet_init	Video (H264/H265) -> RTSP (V1) Video (RGB) -> NN face detect (V4)	(1) RTSP video stream over the network. (2) NN do face detection then draw the bounding box and face landmark to RTSP channel.
mmf2_video_example_face_rtsp_init	Video (H264/H265) -> RTSP (V1) Video (RGB) -> NN face detect (V4) -> NN face recognition	(1) RTSP video stream over the network. (2) NN do face detection and face recognition, and then draw the bounding box and face recognition result to RTSP channel.
mmf2_video_example_joint_test_all_nn_rtsp_init	Video (H264/H265) -> RTSP (V1) RGB -> NN object detect (V4) RGB -> NN face detect (V4) -> NN face recognition AUDIO -> NN audio classification	(1) RTSP video stream over the network. (2) NN do object detection, face detection and face recognition, and then draw the bounding box and face recognition result to RTSP channel. NN do audio classification.
mmf2_video_example_joint_test_vipnn_rtsp_mp4_init	H264 -> MP4 (V1) Video (H264/H265) -> RTSP (V2) RGB -> NN object detect (V4) RGB -> NN face detect (V4) -> NN face recognition (optional) AUDIO -> AAC -> RTSP and mp4 RTP -> AAD -> AUDIO AUDIO -> NN audio classification	(1) RTSP video stream over the network. (2) AmebaPro2 will record three videos (720P 30FPS+AAC) to the SD card for 30 seconds each. The default storage name is : AmebaPro2_recording_0.mp4 AmebaPro2_recording_1.mp4 AmebaPro2_recording_2.mp4 (3) Streaming AAC sounds to AmebaPro2 via the network.

		(4) RTP send the audio stream from network to Ameba Pro2 and the stream is decoded by AAD and played through 3.5 audio jack. (5) NN do object detection, face detection and face recognition, and then draw the bounding box and face recognition result to RTSP channel. NN do audio classification.
mmf2_video_example_audio_vipnn_init.c	AUDIO -> NN	The sound received by Ameba Pro2 can be transmitted to NN engine to do sound classification.

14.6.1 Set RGB video resolution as model input size

If setting the RGB resolution according to NN model input tensor shape, it can avoid image resizing and save pre-processing time.

For example, if you are using yolov4-tiny with input size 416x416, you should set NN_WIDTH and NN_HEIGHT to 416 in video_v4_params.

```
#define YOLO_MODEL              1
#define USE_NN_MODEL            YOLO_MODEL
…
#if (USE_NN_MODEL==YOLO_MODEL)
#define NN_WIDTH     416
#define NN_HEIGHT    416
static float nn_confidence_thresh = 0.4;
static float nn_nms_thresh = 0.3;
#else
#error Please set model correctly. (YOLO_MODEL)
#endif
…
static video_params_t video_v4_params = {
    .stream_id       = NN_CHANNEL,
    .type            = NN_TYPE,
    .resolution      = NN_RESOLUTION,
    .width           = NN_WIDTH,
    .height          = NN_HEIGHT,
    .bps             = NN_BPS,
    .fps             = NN_FPS,
    .gop             = NN_GOP,
    .direct_output   = 0,
    .use_static_addr = 1
};
```

NOTE

Please always check the NN model input size and set RGB video output size according to it. Otherwise, software image resizing will be used in pre-process, and it will cost a lot of CPU usage.

14.6.2 Set RGB video output as resized or cropped

The RGB video parameter can be adjust and make RGB output a resized image or cropped image.

Output cropped RGB:

```
static video_params_t video_v4_params = {
 .stream_id          = NN_CHANNEL,
 .type               = NN_TYPE,
 .resolution         = NN_RESOLUTION,
 .width              = NN_WIDTH,
 .height             = NN_HEIGHT,
 .bps                = NN_BPS,
 .fps                = NN_FPS,
 .gop                = NN_GOP,
 .direct_output      = 0,
 .use_static_addr = 1
};
```

Output resized RGB (1920 x 1080 → NN_WIDTH x NN_HEIGHT):

```
static video_params_t video_v4_params = {
 .stream_id          = NN_CHANNEL,
 .type               = NN_TYPE,
 .resolution         = NN_RESOLUTION,
 .width              = NN_WIDTH,
 .height             = NN_HEIGHT,
 .bps                = NN_BPS,
 .fps                = NN_FPS,
 .gop                = NN_GOP,
 .direct_output      = 0,
 .use_static_addr = 1,
 .use_roi = 1,
 .roi = {
        .xmin = 0,
        .ymin = 0,
        .xmax = 1920, //SENSOR MAX WIDTH
        .ymax = 1080, //SENSOR MAX HEIGHT
 }
};
```

14.6.3 Choose NN model

Please check the desired models are selected in amebapro2_fwfs_nn_models.json, so that the model will be packed into the final firmware image. For example, if we want to use yolov4_tiny and YAMNet_s, go to "project/realtek_amebapro2_v0_example/GCC-RELEASE/mp/amebapro2_fwfs_nn_models.json" and set model yolov4_tiny - "MODEL0" and YAMNet_s - "MODEL2" be used:

```
{
    "msg_level":3,

    "PROFILE":["FWFS"],
    "FWFS":{
         "files":[
               "MODEL0",
               "MODEL2"
         ]
    },
    "MODEL0":{
        "name" : "yolov4_tiny.nb",
        "source":"binary",
        "file":"yolov4_tiny.nb"

    },
    "MODEL1":{
        "name" : "yamnet_fp16.nb",
        "source":"binary",
        "file":"yamnet_fp16.nb"

    },
    "MODEL2":{
        "name" : "yamnet_s.nb",
        "source":"binary",
        "file":"yamnet_s.nb"

    },
    "MODEL3":{
        "name" : "mobilefacenet_int16.nb",
        "source":"binary",
        "file":"mobilefacenet_int16.nb"

    }
}
```

NOTE

After choosing the model, user have to check the ddr memory and flash size usage of models. Please refer 14.5.1 and 14.5.2 to do evaluation.

14.6.4 Build NN example

Since it's a part of video mmf example, user should use the following command to generate the makefile.
Generate the makefile for the NN project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DVIDEO_EXAMPLE=ON
```

Then, use the following command to generate an image with NN model inside:

```
cmake --build . --target flash_nn
```

After running the command above, you will get the flash_ntz.nn.bin (including the model) in "project\realtek_amebapro2_v0_example\GCC-RELEASE\build"

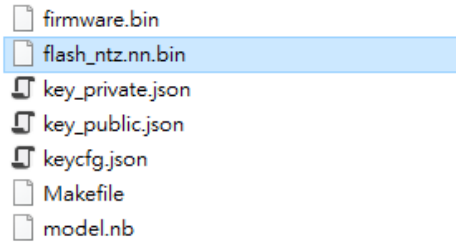


Figure 14-3 image with NN model

Then, use the image tool to download it to Ameba Pro2.

14.6.5 Update NN model on flash

If user just want to update the NN model instead of updating whole firmware, the following command can be used to update NN section on flash partially:

Nand flash

```
$ .\uartfwburn.exe -p COM? -f .\flash_ntz.nn.bin -b 3000000 -n pro2 -t 0x81cf
```

14.6.6 Validate NN example

Refer the following section to validate nn examples.

14.6.6.1 Object detection example

While running the example, you may need to configure WiFi connection by using these commands in uart terminal.

```
ATW0=<WiFi_SSID> : Set the WiFi AP to be connected
ATW1=<WiFi_Password> : Set the WiFi AP password
ATWC : Initiate the connection
```

If everything works fine, you should see the following logs

```
…
[VOE]RGB3 640x480 1/5
[VOE]Start Mem Used ISP/ENC:     0 KB/    0 KB Free=  701
hal_rtl_sys_get_clk 2
GCChipRev data = 8020
GCChipDate data = 20190925
queue 20121bd8 queue mutex 71691380
npu gck vip_drv_init, video memory heap base: 0x71B00000, size: 0x01300000
yuv in 0x714cee00
[VOE][process_rgb_yonly_irq][371]Errrgb ddr frame count overflow : int_status 0x00000008
buf status 0x00000010 time 15573511 cnt 0
input 0 dim 416 416 3 1, data format=2, quant_format=2, scale=0.003660, zero_point=0
ouput 0 dim 13 13 255 1, data format=2, scale=0.092055, zero_point=216
ouput 1 dim 26 26 255 1, data format=2, scale=0.093103, zero point=216
--------------------------------
input count 1, output count 2
input param 0
        data_format  2
        memory type  0
        num_of_dims  4
        quant_format 2
        quant data   , scale=0.003660, zero point=0
        sizes        1a0 1a0 3 1 0 0
```

```
output param 0
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant_data  , scale=0.092055, zero_point=216
        sizes        d d ff 1 0 0
output param 1
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant_data  , scale=0.093103, zero_point=216
        sizes        1a 1a ff 1 0 0
--------------------------------
in 0, size 416 416
VIPNN opened
siso_array_vipnn started
nn tick[0] = 47
object num = 0
nn tick[0] = 46
object num = 0
…
```

Then, open VLC and create a network stream with URL: rtsp://192.168.x.xx:554
If everything works fine, you should see the object detection result on VLC player.

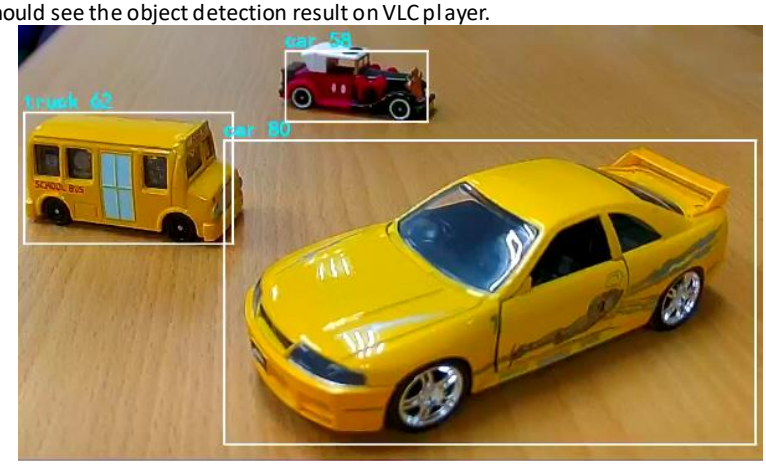


Figure 14-4 VLC validation

14.6.6.2 Face detection example

While running the example, you may need to configure WiFi connection by using these commands in uart terminal.

```
ATW0=<WiFi_SSID> : Set the WiFi AP to be connected
ATW1=<WiFi_Password> : Set the WiFi AP password
ATWC : Initiate the connection
```

If everything works fine, you should see the following logs

```
…
[VOE]RGB3 640x480 1/10
[VOE]zoom default setting
[VOE]status == 1718
[VOE]release s4 isp buffer 0
[VOE]release s4 isp buffer 1
hal_rtl_sys_get_clk 0
GCChipRev data = 8020
GCChipDate data = 20190925
queue 20129598 queue mutex 71c17500
npu gck vip_drv_init, video memory heap base: 0x72800000, size: 0x01800000
NN IRQ default priority : 0, set to 9
--------------------------------
input count 1, output count 4
```

```
input param 0
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant_data  , scale=1.000000, zero_point=0
        sizes        280 1e0 3 1 0 0
output param 0
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant_data  , scale=0.003550, zero_point=0
        sizes        a0 78 1 1 0 0
output param 1
        data format  2
        memory_type  0
        num_of_dims  4
        quant format 2
        quant_data  , scale=0.019616, zero_point=0
        sizes        a0 78 2 1 0 0
output param 2
        data_format  2
        memory_type  0
        num of dims  4
        quant_format 2
        quant data  , scale=0.004289, zero point=130
        sizes        a0 78 2 1 0 0
output param 3
        data format  2
        memory_type  0
        num_of_dims  4
        quant format 2
        quant_data  , scale=0.004716, zero_point=1
        sizes        a0 78 a 1 0 0
--------------------------------
VIPNN opened
siso_video_vipnn started
NN OSD Draw start
nn_rect_ch:0, nn_rect_txt_w:16, nn_rect_txt_h:32.
font resize new size: 4768.
font resize new size: 3688.
font resize from 32 64 to 16 32.
font resize from 64 64 to 32 32.
font resize:22.
object num = 1
0,c0:928 333 1235 700
object num = 1
0,c0:942 325 1237 691
object num = 1
0,c0:944 332 1234 684
object num = 1
0,c0:933 332 1229 684
object num = 1
0,c0:930 332 1232 684
…
```

Then, open VLC and create a network stream with URL: rtsp://192.168.x.xx:554
If everything works fine, you should see the face detection result on VLC player.

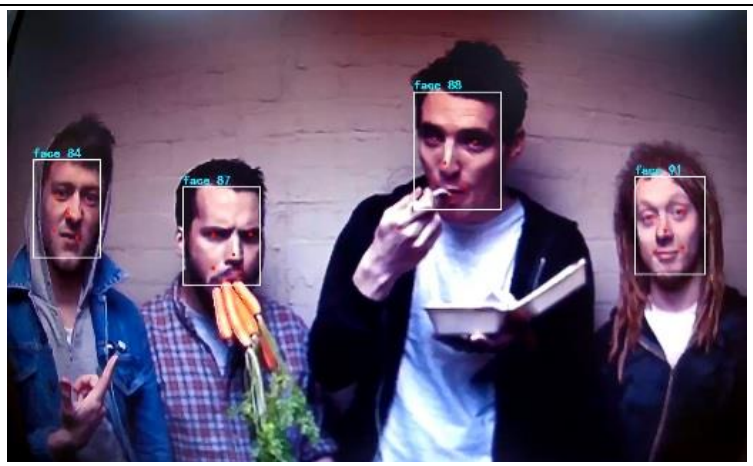


Figure 14-5 face detection VLC validation (COCO 2017)

14.6.6.3 Face recognition example

While running the example, you may need to configure WiFi connection by using these commands in uart terminal.

```
ATW0=<WiFi_SSID> : Set the WiFi AP to be connected
ATW1=<WiFi_Password> : Set the WiFi AP password
ATWC : Initiate the connection
```

If everything works fine, you should see the following logs

```
…
[VOE]RGB3 640x480 1/10
[VOE]zoom default setting
[VOE]status == 1718
[VOE]release s4 isp buffer 0
[VOE]release s4 isp buffer 1
hal_rtl_sys_get_clk 0
GCChipRev data = 8020
GCChipDate data = 20190925
queue 20129d78 queue mutex 71c19aa0
npu gck vip_drv_init, video memory heap base: 0x72800000, size: 0x01800000
NN IRQ default priority : 0, set to 9
Init 1 Queue elements
--------------------------------
input count 1, output count 4
input param 0
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant_data  , scale=1.000000, zero_point=0
        sizes        280 1e0 3 1 0 0
output param 0
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant data  , scale=0.003550, zero point=0
        sizes        a0 78 1 1 0 0
output param 1
        data_format  2
        memory_type  0
        num of dims  4
        quant format 2
        quant_data  , scale=0.019616, zero_point=0
        sizes        a0 78 2 1 0 0
output param 2
        data format  2
        memory_type  0
```

```
        num_of_dims  4
        quant_format 2
        quant_data  , scale=0.004289, zero_point=130
        sizes        a0 78 2 1 0 0
output param 3
        data_format  2
        memory_type  0
        num_of_dims  4
        quant_format 2
        quant_data  , scale=0.004716, zero_point=1
        sizes        a0 78 a 1 0 0
--------------------------------
VIPNN opened
Init 1 Queue elements
--------------------------------
input count 1, output count 1
input param 0
        data_format  2
        memory type  0
        num_of_dims  4
        quant_format 0
        quant data  , none-quant
        sizes        70 70 3 1 0 0
output param 0
        data format  5
        memory_type  0
        num of dims  2
        quant_format 1
        quant_data  , dfp=13
        sizes        80 1 0 0 0 0
--------------------------------
VIPNN2 opened
FACERECOG opened
siso_facenet_facerecog started
siso_facedet_facenet started
siso_array_vipnn started
NN OSD Draw start
nn_rect_ch:0, nn_rect_txt_w:16, nn_rect_txt_h:32.
font resize new size: 4768.
font resize new size: 3688.
font resize from 32 64 to 16 32.
font resize from 64 64 to 32 32.
font resize:22.
update ROI 408 0 531 92
center 437,20 rotate -5.042451
--------> no data
object num = 0
update ROI 406 0 533 95
center 433,23 rotate -5.194429
--------> no data
object num = 0
update ROI 410 0 529 94
center 435,25 rotate -6.115504
--------> no data
…
```

User can use console command to register their face with person ID.
Let camera shooting your face and run the following command to register your face:

```
FREG=My_Name
```

Then, your ID will be registered and please open VLC to check the result

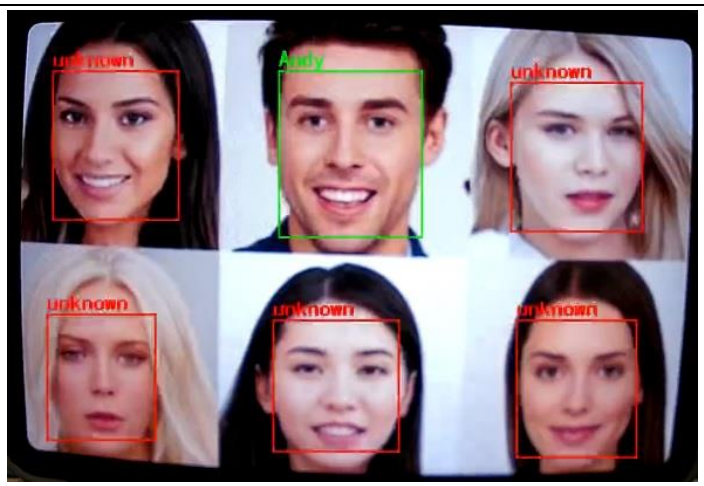


Figure 14-6 face recognition VLC validation (AI generated faces)

You can also register other person with their ID

```
FREG=Person_ID_1
…
FREG=Person_ID_2
…
FREG=Person_ID_3
…
```

If you want to clear and reset all registered person ID, run following command to reset:

```
FRFR
```

If you want to save the registered face feature to flash, please run

```
FRFS
```

After saving the features to flash, you can load the registered face feature from flash after system reboot

```
FRFL
```

14.6.6.4 Audio classification example

If everything works fine, you should see the following logs

```
…
Deploy YAMNET_S
fci part tbl start   10
fci part tbl dup cnt 8
update page size 2048  page per block 64
type_name NN_MDL, file_name yamnet_s.nb
open: part_rec 7043d6a0, part_recs_cnt 1, type_id 81cf
file yamnet_s.nb, len 678336
network 70431540
input 0 dim 1 64 96 1, data format=1, quant_format=0, none-quant
ouput 0 dim 3 1 0 0, data format=1, none-quant
---------------------------------
input count 1, output count 1
input param 0
        data_format  1
        memory_type  0
        num of dims  4
        quant_format 0
        quant_data  , none-quant
        sizes        1 40 60 1 0 0
output param 0
        data format  1
        memory_type  0
        num_of_dims  2
        quant format 0
        quant_data  , none-quant
```

```
        sizes        3 1 0 0 0 0
--------------------------------
in 0, size 1 64
VIPNN opened
siso_audio_vipnn started
YAMNET_S tick[0] = 2
class 1, prob 1.00
YAMNET_S tick[0] = 2
class 1, prob 1.00
YAMNET_S tick[0] = 1
class 1, prob 1.00
YAMNET_S tick[0] = 1
class 1, prob 1.00
YAMNET_S tick[0] = 1
class 1, prob 1.00
…
```

User can use audio sample to validate the result. Use CO & smoke audio smaple in https://research.google.com/audioset/dataset/smoke_detector_smoke_alarm.html to verify the result.
YAMNet_s can recognize 3 audio classes:

(1) class 0: CO
(2) class 1: Others
(3) class 2: Smoke

15 ISP

AmebaPro2 has a video engine integrating following function block: sensor driver, image signal processor, OSD and motion detection. The following chapters describe the software flow to setup AmebaPro2 ISP function.

15.1 Sensor Driver

AmebePro2 supports single sensor input with ISP engine processing and output up to 4 channel with independent configuration. Based on AmebaPro2 Spec, ISP can support up to 5m resolution (but fps need to be calculated by bandwidth limitation). In this chapter, we will introduce sensor driver information such as sensor list and sensor configuration.

15.1.1 Sensor list

RTK will update support sensor driver regularly. User can contact RTK-FAE to get newest sensor list. The following table shows support list with sensor information.

Table 15-1 Support sensor list on AmebaPro2

Vendor	Sensor	Max Resolution and FPS	Description
Galaxycore	GC2053	1920x1080 * 30	
Galaxycore	GC4653	2560x1440 * 15	
PrimeSensor	PS5258	1920x1080 * 30	
PrimeSensor	PS5270	1536x1536 * 30	
SmartSens	SC2336	1920x1080 * 30	
SmartSens	SC301IOT	2048x1536 * 20	
SOI	JXF37P	1920x1080 * 30	
SOI	JXF51	1536x1536 * 30	
Sony	IMX307	1920x1080 * 30	With HDR
Sony	IMX327	1920x1080 * 30	With HDR
ImageDesign	MIS2008	1920x1080 * 30	

15.1.2 Sensor configuration

User can check supported sensor list and sensor settings is in "project\realtek_amebapro2_v0_example\inc\sensor.h":

15.1.2.1 Check sensor list

First, check whether your sensor is in the list or not. If not, add your sensor by below steps.
Take "**SENSOR_SC301**" as example,

1. Add sensor parameters to the list. (refers to the words in red)
2. Rename your FCS/Sensor/IQ files as **fcs_data_sc301.bin**, **sensor_sc301.bin** and **iq_sc301.bin** (fcs_data_xxx.bin is optional.)
3. Copy FCS/Sensor/IQ files to this path: **component\soc\8735b\fwlib\rtl8735b\lib\source\ram\video\voe_bin**

```
#define SENSOR_DUMMY        0x00 //For dummy sensor, no support fast camera start
#define SENSOR_SC2336             0x01
#define SENSOR_GC2053             0x02
…
#define SENSOR_PS5268             0x0F
#define SENSOR_SC2310             0x10
#define SENSOR_SC301        0x11

static const struct sensor_params_t sensor_params[] = {
 {1920, 1080, 30}, //DUMMY
 {1920, 1080, 30}, //SC2336
 {1920, 1080, 30}, //GC2053
…
 {1920, 1080, 30}, //PS5268
 {1920, 1080, 30}, //SC2310
 {2048, 1536, 20}, //SC301
};
```

15.1.2.2 Setup max sensor number.

```
#define SENSOR_MAX          5
```

[Description]

- **SENSOR_MAX**: If 4 sensors are used in the project, please set the parameter to 5, because the first column needs to place the dummy data required by FCS.

15.1.2.3 Fill in sensor pool

```
static const unsigned char sen_id[SENSOR_MAX] = {
 SENSOR_DUMMY,
 SENSOR_GC2053,
 SENSOR_GC4653,
 SENSOR_GC4023,
 SENSOR_SC2333
};
#define USE_SENSOR          SENSOR_GC2053
```

[Description]

- **sen_id[ ]**: Select the sensor to be used from the list and fill in this array
- **USE_SENSOR**: Select the sensor to be used from the sen_id[ ].

15.1.2.4 Setup manual IQ

Select IQ manually.

```
static const      char manual_iq[SENSOR_MAX][64] = {
 "iq",
 "iq_gc2053",
 "iq_gc4653",
 "iq_gc4023",
 "iq_sc2333",
};
#define MANUAL_SENSOR_IQ    0xFF
```

[Description]

- **manual_iq[ ]**: Fill in the IQ file name corresponding to sen_id[ ].
- **MANUAL_SENSOR_IQ**: Manually select the sensor ID in firmware to bring in.

NOTE

*Enable manual IQ when MANUAL_SENSOR_IQ value is set between **0 ~ SENSOR_MAX-1**, disable when value is set to **0xFF.***

15.1.2.5 Cleanup and Rebuild

After finishing the configuration, please cleanup below path, and rebuild the project to make sure sensor setting take effect.

```
project\realtek_amebapro2_v0_example\GCC-RELEASE\build
```

15.1.3 How to apply “sensor.h” file to fit to customized usage

Supposed that we have following complexity set for one project, dual mode for sensor (HDR / Linear), dual lens(1st source / 2nd source) with dual mode (production / manufacture). Here is an example of configuring the **sensor.h** file.

Table 15-2 Combination cases for SENSOR/IQ/FCS usage.

Index	Fcs Sensor Bin	Normal Sensor Bin	IQ Bin
0	fcs_data_sensor_linear.bin	sensor_linear.bin	iq_sensor_project_1stlens_linear_production
1	fcs_data_sensor_linear.bin	sensor_linear.bin	iq_sensor_project_1stlens_linear_manufacture
2	fcs_data_sensor_linear.bin	sensor_linear.bin	iq_sensor_project_2ndlens_linear_production
3	fcs_data_sensor_linear.bin	sensor_linear.bin	iq_sensor_project_2ndens_linear_manufacture
4	fcs_data_sensor_hdr.bin	sensor_hdr.bin	iq_sensor_project_1stlens_hdr_production
5	fcs_data_sensor_hdr.bin	sensor_hdr.bin	iq_sensor_project_1stlens_hdr_manufacture
6	fcs_data_sensor_hdr.bin	sensor_hdr.bin	iq_sensor_project_2ndlens_hdr_production
7	fcs_data_sensor_hdr.bin	sensor_hdr.bin	iq_sensor_project_2ndens_hdr_manufacture

According to the table, first make sure that the sensor/IQ/FCS files has been placed in the path:
component\soc\8735b\fwlib\rtl8735b\lib\source\ram\video\voe_bin
As follows, then we can implement this example through the configuration in **sensor.h**

```
#define SENSOR_MAX          9
static const unsigned char sen_id[SENSOR_MAX] = {
 SENSOR_DUMMY,
```

```
 SENSOR_LINEAR,
 SENSOR_LINEAR,
 SENSOR_LINEAR,
 SENSOR_LINEAR,
 SENSOR_HDR,
 SENSOR_HDR,
 SENSOR_HDR,
 SENSOR_HDR
};
#define USE_SENSOR          SENSOR_LINEAR
static const      char manual_iq[SENSOR_MAX][64] = {
 "iq_dummy",
 "iq_sensor_project_1stlens_linear_production",
 "iq_sensor_project_1stlens_linear_manufacture",
 "iq_sensor_project_2ndlens_linear_production",
 "iq sensor project 2ndens linear manufacture",
 "iq_sensor_project_1stlens_hdr_production",
 "iq_sensor_project_1stlens_hdr_manufacture",
 "iq sensor project 2ndlens hdr production",
 "iq_sensor_project_2ndens_hdr_manufacture",
};
#define MANUAL_SENSOR_IQ    0x1
```

15.2 Image Quality

User can bring up sensor with basic image quality on RTK-EVT or user's DUT. Image Quality will vary based on selected lens and optical structure. For different application, end customer will also have different image quality criteria. Based on each project, user can check subjective and objective image quality criteria or compare with target DUT. For advanced image quality tuning support, user can contact RTK-FAE.

15.2.1 Use UVC Example

Ameba Pro2 ISP can support compressed (H264 / H265 / JPG) and uncompressed (NV16 / NV12) image through UVC (wired transmission), and user can check video on pc with Potplayer, Amcap or RTK-realcam. For uncompressed format, user need to install RTK decoder to get video on computer. User can use following flow to build UVC example. Generate the makefile for the UVC project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain_ci.cmake -
DEXAMPLE=media_uvcd
```

Then, use the following command to generate an image:

```
cmake --build . --target flash
```

15.2.2 Mode switch through IQ configuration

Ameba Pro2 ISP use iq bin defined at json file to set image quality parameter. For each iq bin file, it would contain up to 3 i q tables. User can use ISP API to switch. (isp_set_day_night() for iq table switch, isp_set_gray_mode() for color/gray mode switch) For general usage, suggestion configuration will be defined as follow

Table 15-3 Example for iq mode switch

Index	Mode	IQ Table command	Color mode command
0	RGB mode (RGB parameter with color)	isp_set_day_night(0)	isp_set_gray_mode(0)
1	IR mode (IR parameter w/o color)	isp_set_day_night(1)	isp_set_gray_mode(1)
2	Other mode(like spotlight mode with color or IR mode without IR LED w/o color)	isp_set_day_night(2)	User define

15.2.3 Image Quality Criteria

For first draft image quality version, RTK will provide image quality patch following RTK criteria. User can check table for detailed.

Table 15-4 Objective image quality on AmebaPro2

Category	Condition	Criteria
Lens Shading	D65 & CWF & A	Relative illumination > 80%

	D65 & CWF & A	R/G [0.9~1.1]
	D65 & CWF & A	B/G [0.9~1.1]
	D65 & CWF & A	B/G [0.9~1.1]
Color Checker	D65 & CWF & A	Saturation [100%~120%]
	D65 & CWF & A	Mean △C ≦10
	D65 & CWF & A	Max △C ≦ 30
	D65 & CWF & A	Mean △E ≦20
	D65 & CWF & A	Max △E ≦ 30
Auto White Balance	D65 & CWF & A	#20~#23 Max △S ≦0.1
Resolution (1080P)	D65	Center Horizontal: ≧1000
		Center Vertical: ≧1000
		Corner Horizontal ≧600
		Corner Vertical ≧600
Dynamic Range	D65	Max Y ≧ 200
		Step ≧ 14
Defect Pixel	Dark & White	None

For advanced image quality such as customized objective image criteria or quality benchmark with target DUT, user can contact with RTK-FAE for tuning support.

15.3 OSD2

15.3.1 OSD2 introduction

The text image display consists of hardware maps, drivers and provided Libs. Users use the provided API to create instances, set alphanumeric and image properties, and place alphanumerics or images on streaming images. Users can replace different font files, and then provide enough memory for the OSD to convert alphanumerics into color images according to the font size and text length. The input and output image buffers here need to be physically continuous memory.

15.3.2 Configuration

- Display image
- Display alphanumeric
- Display date and time
- Alphanumeric rotation, stroke, transparency
- Font library capability;
  - Supports up to 3 sets of different fonts
  - Each font group supports up to 1 single-character glyph file and 1 double-character glyph file
- Each stream can display up to 6 sets of OSD Block

NOTE

The starting address of the image Array must be 16Byte align with the Hardware DMA limit

15.3.3 OSD example

OSD example is included in RTSP (**-DVIDEO_EXAMPLE=ON**) and UVCD (**-DEXAMPLE=media_uvcd**) examples, and it is located at the path "component\video\osd2\isp_osd_example.c"
Take UVCD for example, before building the firmware, run below command to create the makefile.

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DCUTVER=B -
DEXAMPLE=media_uvcd
```

- Execution and testing
  - Open the ISP AT command in platform_otps.h:

```
#define CONFIG_ISP   1
```

  - Build code and load image.
  - Connect the USB cable to the AmebaPro2 CON port and the other end to the PC.
  - Open potplayer, enter atcmd "ATIO=task,0,0,28,56" will show results.
    - Command parameters:"ATIO=task,0,0(channel),28(char_width),56(char_height)"

15.3.4 OSD Enumerations and Data Structures

In this chapter, we will list OSD API with function parameter introduction.

15.3.4.1 OSD Data Structures

Table 15-5 OSD data structure

Data Structures	Introduction
<osd_text_info_st>	Text type OSD parameters.
<rt_font_st>	Fonts parameters
<osd_pict_st>	Picture type OSD parameters.
<rt_osd2_info_st>	OSD parameters.

Table 15-6 OSD data structure: osd_text_info_st

Parameter	Type	Introduction
<chn_id>	int	Channel ID: 0~2
<blk_idx>	int	Block index: 0~5
<font>	rt_font_st	Please refers to table of rt_font_st
<start_x>	uint32_t	x-coordinate of start point
<start_y>	uint32_t	y-coordinate of start point
<rotate>	uint32_t	Please refers to enumeration of rt_rotate_t
<str>	char *	String content

Table 15-7 OSD data structure: rt_font_st

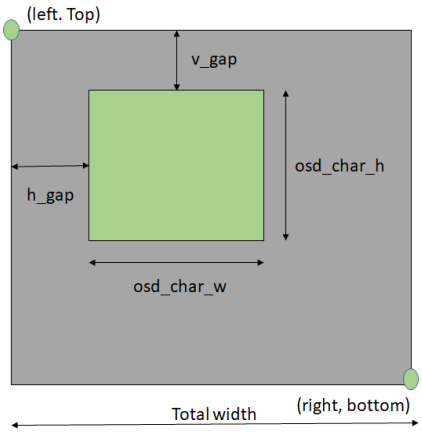
Parameter	Type	Introduction
<block_alpha>	uint8_t	Transparent value: 0~15.
<ch_color>	uint32_t	Character color in RGB.
<bg_enable>	uint8_t	Enable background: 0~1.
<bg_color>	uint32_t	Background color in RGB.
<h_gap>	uint8_t :4	 The meaning of the field in the osd structure is shown in the figure. The horizontal interval is h_gap, and the vertical interval is v_gap.
<v_gap>	uint8_t :4	
<time_fmt>	rts_osd_time_fmt	Time format, please refer to introduction of rts_osd_time_fmt
<date_fmt>	rts_osd_date_fmt	Date format, please refer to introduction of rts_osd_date_fmt

Table 15-8 OSD data structure: osd_pict_st

Parameter	Type	Introduction
<chn_id>	int	Channel ID: 0~2
<osd2>	rt_osd2_info_st	OSD parameters.

Table 15-9 OSD data structure: rt_osd2_info_st

Parameter	Type	Introduction
<blk_idx>	int	Block index: 0~5
<blk_fmt>	rts_osd2_blk_fmt	Block format: Please refers to enumeration rts_osd2_blk_fmt
<start_x>	uint32_t	x-coordinate of start point
<start_y>	uint32_t	y-coordinate of start point
<end_x>	uint32_t	x-coordinate of end point
<end_y>	uint32_t	y-coordinate of end point
<color_1bpp>	uint32_t	Set the RGB color when format is RTS_OSD2_BLK_FMT_1BPP
<buf>	uint8_t *	Image buffer
<len>	uint32_t	Image buffer length

15.3.4.2 OSD Enumerations

Table 15-10 OSD Enumerations

Enumerations	Introduction
<rt_rotate_t>	Rotation angle, include 0, 90, 180, 270..
<rts_osd_time_fmt>	Time format
<rts_osd_date_fmt>	Date format
<rts_osd2_blk_fmt>	Block format

Table 15-11 OSD data structure: rt_rotate_t

Definition	Introduction
<RT_ROTATE_0>	None rotation
<RT_ROTATE_90R>	Rotate 90 degree to the right
<RT_ROTATE_180R>	Rotate 180 degree to the right
<RT_ROTATE_270R>	Rotate 270 degree to the right
<RT_ROTATE_90L>	Rotate 90 degree to the left
<RT_ROTATE_180L>	Rotate 180 degree to the left
<RT_ROTATE_270L>	Rotate 270 degree to the left

Table 15-12 OSD data structure: rts_osd_time_fmt

Definition	Type	Introduction
<osd_time_fmt_no>	Not display time	Not display time
<osd_time_fmt_24>	hh:mm:ss	14:32:58
<osd_time_fmt_12>	hh:mm:ss	02:32:58
<osd_time_fmt_12_1>	Phh:mm:ss	P02:32:58
<osd_time_fmt_12_2>	PMhh:mm:ss	PM02:32:58
<osd_time_fmt_12_3>	PM~hh:mm:ss	PM~02:32:58
<osd_time_fmt_12_4>	hh:mm:ssPM	02:32:58PM
<osd_time_fmt_12_5>	hh:mm:ss~PM	02:32:58~PM
<osd_time_fmt_12_6>	hh:mm:ss~~PM	02:32:58~~PM
<osd_time_fmt_12_7>	hh:mm:ss~~~PM	02:32:58~~~PM

Table 15-13 OSD data structure: rts_osd_date_fmt

Definition	Type	Example
<osd_date_fmt_no>	Not display date	Not display date
<osd_date_fmt_0>	dd/MM/yyyy	26/05/2015
<osd_date_fmt_1>	dd/MM/yy	26/05/15
<osd_date_fmt_2>	d/M/yy	26/5/15
<osd_date_fmt_3>	M/d/yyyy	5/26/2015
<osd_date_fmt_4>	M/d/yy	5/26/15
<osd_date_fmt_5>	MM/dd/yy	05/26/15
<osd_date_fmt_6>	MM/dd/yyyy	05/26/2015
<osd_date_fmt_7>	yyyy/M/d	2015/5/26
<osd_date_fmt_8>	yyyy-M-d	2015-5-26

<osd_date_fmt_9>	yyyy-MM-dd	2015-05-26
<osd_date_fmt_10>	yyyy/MM/dd	2015/05/26
<osd_date_fmt_11>	yy-MM-dd	15-05-26
<osd_date_fmt_12>	yy/M/d	15/5/26
<osd_date_fmt_13>	yy-M-d	15-5-26
<osd_date_fmt_14>	yy/MM/dd	15/05/26
<osd_date_fmt_15>	yyyy.mm.dd	2015.05.26
<osd_date_fmt_16>	dd.mm.yyyy	26.05.2015
<osd_date_fmt_17>	mm.dd.yyyy	05.26.2015
<osd_date_fmt_18>	mm-dd-yyyy	05-26-2015
<osd_date_fmt_19>	dd-mm-yyyy	26-05-2015
<osd_date_fmt_20>	dd-mm-yyyy www	26-05-2015 Tue
<osd_date_fmt_21>	dd/mm/yyyy www	26/05/2015 Tue
<osd_date_fmt_22>	dd.mm.yyyy www	26.05.2015 Tue

Table 15-14 OSD data structure: rts_osd2_blk_fmt

Definition	Introduction
<RTS_OSD2_BLK_FMT_1BPP>	Format in 1BPP, pixel size: 1 bit.
<RTS_OSD2_BLK_FMT_RGBA1111>	Format in RGBA1111, pixel size: 4 bit.
<RTS_OSD2_BLK_FMT_RGBA2222>	Format in RGBA2222, pixel size: 1 byte.
<RTS_OSD2_BLK_FMT_RGBA5551>	Format in RGBA5551, pixel size: 2 bytes.
<RTS_OSD2_BLK_FMT_RGBA4444>	Format in RGBA4444, pixel size: 2 bytes.
<RTS_OSD2_BLK_FMT_RGBA8888>	Format in RGBA8888, pixel size: 4bytes.

If the block type is rts_osd2_type_date, rts_osd2_type_time or rts_osd2_type_text, block format is always RGBA1111.If the block type is rts_osd2_type_pict, below all block format are supported.

15.3.5 OSD API

15.3.5.1 rts_osd_init

Initial function is used to create OSD data, font lib and set the time-zone for the indicated stream.

Table 15-15 OSD API: rts_osd_init

Parameter	Type	Introduction
<chn_id >	int	Stream channel ID.
<char_resize_w> <char_resize_h>	int	Character size in width.
	int	Character size in height.
<timezone_s>	int	Time-zone, unit in seconds.
<chn_id >	int	Stream channel ID.

15.3.5.2 rts_osd_deinit

De-initialize the OSD data of indicated stream.

Table 15-16 OSD API: rts_osd_deinit

Parameter	Type	Introduction
<chn_id>	int	Stream channel ID.

15.3.5.3 rts_osd_set_info

It sets OSD data of indicated stream and block. Each video stream has a separate OSD module. Each OSD module supports up to 6 blocks, a block is an area in the image for displaying characters or images, which represented by the structure “osd_text_info_st” or “osd_pict_st”. English and digital width of a word are inconsistent with Chinese in display. English and array use a single, the width and font files are saved in the single font file. The Chinese display takes up double width, and the font file is saved in the double wide font file. For the detail of “osd_text_info_st” and

"osd_pict_st", refer to previous instructions

Table 15-17 OSD API: rts_osd_set_info

Parameter	Type	Introduction
<osd_type>	int	Types include ● rts_osd2_type_date, ● rts_osd2_type_time, ● rts_osd2_type_pict, ● rts_osd2_type_text.
<osd_info>	void *	Block detail description, which includes "osd_text_info_st*" and "osd_pict_st*". ● "osd_text_info_st*" includes rts_osd2_type_date, rts_osd2_type_time, rts_osd2_type_text ● "osd_pict_st*" includes rts_osd2_type_pict

15.3.5.4 rts_osd_get_timezone

Get the time-zone.
Parameter: None.

15.3.5.5 rts_osd_set_timezone

Set the time-zone.

Table 15-18 OSD API: rts_osd_set_timezone

Parameter	Type	Introduction
<timezone_s>	int	The value of time-zone in seconds.

15.3.5.6 rts_osd_isp_refresh_datetime

Refresh date-time. All stream use the same date-time information.
Parameter: None.

15.3.5.7 rts_osd_block_hide

The function used to hide the indicated block.

Table 15-19 OSD API: rts_osd_block_hide

Parameter	Type	Introduction
<chn_id>	int	Channel ID: 0~2
<idx>	int	Block index: 0~5

15.3.5.8 rts_osd_block_show

The function used to show the indicated block.

Table 15-20 OSD API: rts_osd_block_show

Parameter	Type	Introduction
<chn_id>	int	Channel ID: 0~2
<idx>	int	Block index: 0~5

15.3.5.9 rts_set_char_size

This function used to change character size dynamically.

Table 15-21 OSD API: rts_set_char_size

Parameter	Type	Introduction
<chn_id>	Int	Stream channel ID.
<char_resize_w>	Int	Character size in width.

<char_resize_h>	int	Character size in height.

15.3.5.10 rts_set_font_char_size

This function used to change font lib and character size dynamically.

Table 15-20 OSD API: rts_set_font_char_size

Parameter	Type	Introduction
<chn_id>	Int	Stream channel ID.
<char_resize_w>	Int	Character size in width.
<char_resize_h>	int	Character size in height.
<font_eng>	void *	English font lib.
<font_chi>	void *	Chinese font lib.

15.3.5.11 rts_osd_task

OSD task function.
Parameter: None.

NOTE

Please use xTaskCreate to create the task.

15.3.6 OSD Render Task

For MD and NN examples, we provided an osd_render_task to dynamically render OSD object in real time.
For the usage of osd_render_task, please first initial OSD objects and font library, and then start the osd_render_task.

```
//osd render init
int ch_enable[3] = {1, 0, 0};
int char_resize_w[3] = {16, 0, 0}, char_resize_h[3] = {32, 0, 0};
int ch_width[3] = {RTSP_WIDTH, 0, 0}, ch_height[3] = {RTSP_HEIGHT, 0, 0};
osd_render_dev_init(ch_enable, char_resize_w, char_resize_h);
osd_render_task_start(ch_enable, ch_width, ch_height);
```

When the osd_render_task start properly, user should create a bitmap. After bitmap created, user can use canvas function, such as canvas_set_point, canvas_set_line, canvas_set_rect, canvas_set_text to draw desired object. Then, update the canvas content to the video.

```
//update osd object
canvas_create_bitmap(ch, idx, RTS_OSD2_BLK_FMT_1BPP);
canvas_set_point(ch, idx, xmin, ymin, point_width, color);
canvas_set_line(ch, idx, xmin, ymin, xmax, ymax, line_width, color);
canvas_set_rect(ch, idx, xmin, ymin, xmax, ymax, line_width, color);
canvas_set_text(ch, idx, xmin, ymin, text_string, color);
canvas_update(ch, idx, 1);
```

Use the following API to stop the osd_render_task and de-initialize OSD objects and font lib.

```
//osd render deinit
osd_render_task_stop();
osd_render_dev_deinit_all();
```

We provided several video examples using OSD render task to show the detection result.

Table 15-22 Video Example with OSD Render Task

Example	Description	Result
mmf2_video_example_md_rtsp_init	CH1 Video -> H264/HEVC -> RTSP CH4 Video -> RGB -> MD	(1) RTSP video stream over the network. (2) MD detect motion and draw the motion region to RTSP channel.
mmf2_video_example_vipnn_rtsp_init	CH1 Video -> H264/HEVC -> RTSP CH4 Video -> RGB -> NN	(1) RTSP video stream over the network. (2) NN do object detection and draw the bounding box to RTSP channel.

15.3.6.1 osd_render_dev_init

Initial function to create OSD data and font lib.

Parameter	Type	Introduction
<ch_enable>	Int*	Select the video channel that desired to draw OSD object. Channel 0~2 are available to draw. For example, if only want to draw on video channel 0, we will set ch_enable[3] = {1, 0, 0}
<char_resize_w>	Int*	Font width settings for each video channel. The unit is pixel.
<char_resize_h>	Int*	Font height settings for each video channel. The unit is pixel.

15.3.6.2 osd_render_dev_deinit

De-initialize OSD object and font lib.

Parameter	Type	Introduction
<ch>	Int	Select the video channel that desired to de-initialize OSD object

15.3.6.3 osd_render_dev_deinit_all

De-initialize OSD object and font lib for all video channel.

15.3.6.4 osd_render_task_start

Start OSD render task.

Parameter	Type	Introduction
<ch_visible>	Int*	Select the video channel that desired to draw OSD object. Channel 0~2 are available to draw. For example, if only want to draw on video channel 0, we will set ch_visible[3] = {1, 0, 0}
<ch_width>	Int*	The resolution width settings for each video channel. The unit is pixel.
<ch_height>	Int*	The resolution height settings for each video channel. The unit is pixel.

15.3.6.5 osd_render_task_stop

Stop OSD render task.

15.3.6.6 canvas_create_bitmap

Create bitmap for OSD render object.

Parameter	Type	Introduction
<ch>	Int	Channel index: 0~2
<idx>	Int	Block index: 0~23
<bmp_format>	enum rts_osd2_blk_fmt	OSD render task only support RTS_OSD2_BLK_FMT_1BPP and RTS_OSD2_BLK_FMT_RGBA2222. RTS_OSD2_BLK_FMT_1BPP use less storage but can only draw one color. RTS_OSD2_BLK_FMT_RGBA2222 can draw 27 colors with 3 transparency settings with more storage usage.

15.3.6.7 canvas_update

Update OSD render object. Note that the latest update of OSD object with the same channel id and block id will be shown, so make sure not using the same channel and id for different object.

Parameter	Type	Introduction
<ch>	Int	Channel index: 0~2
<idx>	Int	Block index: 0~23
<ready2update>	Int	When ready2update is set to 1, all the OSD block at the same channel will be shown.

15.3.6.8 canvas_set_point

Draw point on bitmap.

Parameter	Type	Introduction
<ch>	Int	Channel index: 0~2
<idx>	Int	Block index: 0~23

<x>	Int	Point x coordinate value. The unit is pixel.
<y>	Int	Point y coordinate value. The unit is pixel.
<point_width>	Int	Point width. The unit is pixel.
<color>	uint32_t	Point color

15.3.6.9 canvas_set_line

Draw line on bitmap.

Parameter	Type	Introduction
<ch>	Int	Channel index: 0~2
<idx>	Int	Block index: 0~23
<xstart>	Int	The start point x-coordinate value of the line. The unit is pixel.
<ystart>	Int	The start point y-coordinate value of the line. The unit is pixel.
<xend>	Int	The end point x-coordinate value of the line. The unit is pixel.
<yend>	Int	The end point y-coordinate value of the line. The unit is pixel.
<line_width>	Int	Line width. The unit is pixel.
<color>	uint32_t	Line color

15.3.6.10 canvas_set_rect

Draw rect on bitmap.

Parameter	Type	Introduction
<ch>	Int	Channel index: 0~2
<idx>	Int	Block index: 0~23
<xmin>	Int	The upper left x-coordinate value of the rectangle. The unit is pixel.
<ymin>	Int	The upper left y-coordinate value of the rectangle. The unit is pixel.
<xmax>	Int	The bottom right x-coordinate value of the rectangle. The unit is pixel.
<ymax>	Int	The bottom right y-coordinate value of the rectangle. The unit is pixel.
<line_width>	Int	Line width. The unit is pixel. When set to -1, the rectangle will be filled.
<color>	uint32_t	Rectangle color

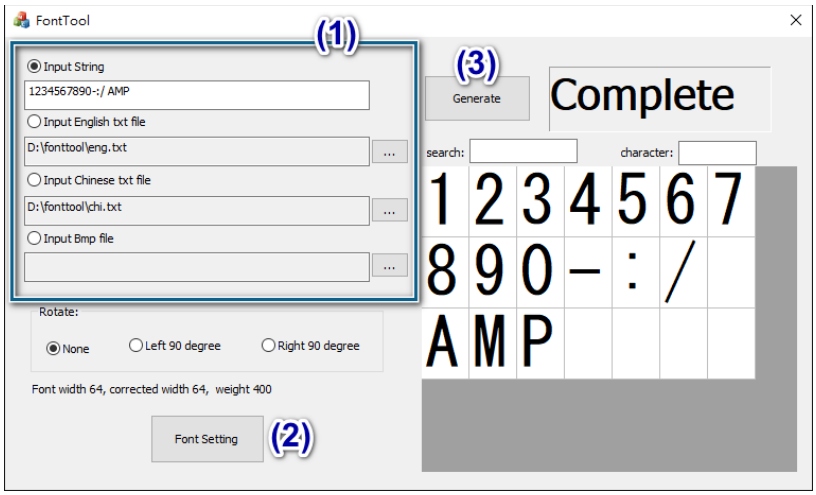
15.3.6.11 canvas_set_text

Draw text on bitmap.

Parameter	Type	Introduction
<ch>	Int	Channel index: 0~2
<idx>	Int	Block index: 0~23
<xmin>	Int	The upper left x-coordinate value of the text. The unit is pixel.
<ymin>	Int	The upper left y-coordinate value of the text. The unit is pixel.
<text_string>	char *	Text string
<color>	uint32_t	Text color

15.3.7 OSD Tools

15.3.7.1 Font Tool



(1) 3 options to select the input method: input by strings, English text files, Chinese text files
- Please note the option "Input Bmp file" is not used for generating font lib.

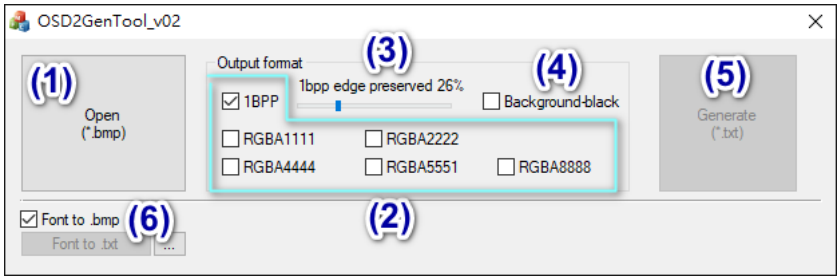
(2) Select font: Please avoid the situation when incomplete characters in the preview screen

(3) Generate font-lib: test.bin

[Usage]

Apply test.bin by OSD function: rts_set_font_char_size()

15.3.7.2 Bitmap Generate Tool



(1) Open the .bmp file.

(2) Select the conversion format.
- When you want to output 1BPP(1 bit per pixel) format, the input image should be less gray level, and using black/white as much as possible.

(3) When 1BPP is checked, select the percentage of you want to preserve gray level.

(4) Select the background as black or white
- Check: white color maps to foreground; black color maps to background.
- Uncheck: white color maps to background; black color maps to foreground.

(5) Generate a text file
- Text file is placed in the same path as the input image file.
- File content includes information about length, width/height, format and array.

[Usage]

i. Please reference to the OSD example in the path to update data:

```
component\video\osd2\isp_osd_example.c
```

ii. Paste the generated OSD data to the example code, and rename the variables depend on your requirement. For example:

```
int logo_w = 142;
int logo_h = 66;
enum rts_osd2_blk_fmt logo_fmt = RTS_OSD2_BLK_FMT_1BPP;
int logo_size = 1584;
unsigned char logo_custom[] __attribute__((aligned(32))) = {…};
```

iii. Replace the variables in below static functions (emphasized in bold):

```
osd_pict_st posd2_pic_0;
init_osd_bitmap_pos(&posd2_pic_0, ch, 150, 200, logo_w, logo_h);
init_osd_bitmap_blk(&posd2_pic_0, blk_idx, logo_fmt, 0);
```

```
init_osd_bitmap_buf(&posd2_pic_0, logo_custom, logo_size);
```

(6) Please note it is an isolated function to transform font-lib to .txt file, and save those characters in the font-lib to .bmp files if "Font to .bmp" is checked. Please follow below steps.

- Press "..." to open fon-lib (generated by FontTool).
- Check "Font to .bmp" if you also want to save those characters in the font-lib to .bmp.
- Press "Font to .txt" to generate .txt and .bmp files from the font-lib.

15.4 ISP Control API

In this chapter, we list all ISP control API at application layer. User can use these API to do customized image tuning. For all API, we divide them into 5 category: AE, AWB, image tuning, mode, WDR and dehaze. And we will also show an example to evaluate lens through ISP API.

15.4.1 ISP Control API (AE)

15.4.1.1 isp_set_exposure_mode

Table 15-23 ISP API: isp_set_exposure_mode

Parameter	Type	Introduction
<val>	int	The mode of exposure, value is 0 or 1. (0: manual, 1: Auto).

15.4.1.2 isp_get_exposure_mode

Table 15-24 ISP API: isp_get_exposure_mode

Parameter	Type	Introduction
<pval>	Int *	Retrieve the mode of exposure, value is 0 or 1. (0: manual, 1: auto)

15.4.1.3 isp_set_power_line_freq

Table 15-25 ISP API: isp_set_power_line_freq

Parameter	Type	Introduction
<val>	int	Anti-flicker mode. Range: 0 ~ 3 0: Disable, 1: 50Hz, 2: 60Hz, 3: Auto Remark: 1.Auto mode: (A)Auto mode include flicker detection method, and use 50hz as default configuration to check whether there is flicker. If no-flicker happens, it would keep default configuration. Other, it would use 60 hz. This function will always run when ae enable. 2.50 Hz (A) The lowest exposure time to stop flicker is 10ms. If lower, flicker might happen. (B) IQ parameters can hold the flicker off, but side effect is over exposure under high brightness environment. (C) If IQ parameters cannot stop flicker, some FPS settings, such as 25, 20, or 10 can stop the moving. (Banding still exist.) 3.60 Hz (A) The lowest exposure time to stop flicker is 8.33ms. If lower, flicker might happen. (B) IQ parameters can hold the flicker off, but side effect is over exposure under high brightness environment. (C) If IQ parameters cannot stop flicker, some FPS settings, such as 30, 24, 20, 15, or 12 can stop the moving. (Banding still exist.)

15.4.1.4 isp_get_power_line_freq

Table 15-26 ISP API: isp_get_power_line_freq

Parameter	Type	Introduction
<val>	int	Anti-flicker mode. Range: 0 ~ 3 0: Disable, 1: 50Hz, 2: 60Hz, 3: Auto Remark: 1. Auto mode: (A) Auto mode include the algorithm of flicker detection, the detection fail rate might result in flicker problem. 2. 50 Hz (A) The lowest exposure time to stop flicker is 10ms. If lower, flicker might happen. (B) IQ parameters can hold the flicker off, but side effect is over exposure under high brightness environment. (C) If IQ parameters cannot stop flicker, some FPS settings, such as 25, 20, or 10 can stop the moving. (Banding still exist.) 3. 60 Hz (A) The lowest exposure time to stop flicker is 8.33ms. If lower, flicker might happen. (B) IQ parameters can hold the flicker off, but side effect is over exposure under high brightness environment. (C) If IQ parameters cannot stop flicker, some FPS settings, such as 30, 24, 20, 15, or 12 can stop the moving. (Banding still exist.)

15.4.1.5 isp_set_exposure_time

Table 15-27 ISP API: isp_set_exposure_time

Parameter	Type	Introduction
<pval>	Int *	The exposure time, unit is us. Range is 1~100,000. (Depend on sensor driver) Adjustable precision is +-1.

15.4.1.6 isp_get_exposure_time

Table 15-28 ISP API: isp_get_exposure_time

Parameter	Type	Introduction
<val>	int	Retrieve the exposure time, unit is us. Range is 1~100,000. (Depend on sensor driver)

15.4.1.7 isp_set_ae_gain

Table 15-29 ISP API: isp_set_ae_gain

Parameter	Type	Introduction
<val>	int	Gain value. Range: 256~32768 Adjustable precision: +-1

15.4.1.8 isp_get_ae_gain

Table 15-30 ISP API: isp_get_ae_gain

Parameter	Type	Introduction
<pval>	Int *	Retrieve gain value Range: 256~32768

15.4.2 ISP Control API (AWB)

15.4.2.1 isp_set_awb_ctrl

Table 15-31 ISP API: isp_set_awb_ctrl

Parameter	Type	Introduction
<val>	int	Mode of white balance. 0: Manual temperature, 1: Auto.

NOTE

The API of manual temperature is not supported.

15.4.2.2 isp_get_awb_ctrl

Table 15-32 ISP API: isp_get_awb_ctrl

Parameter	Type	Introduction
<pval>	Int *	Retrieve the mode of white balance. 0: Manual, 1: Auto.

15.4.2.3 isp_set_wb_temperature

Table 15-33 ISP API: isp_set_wb_temperature

Parameter	Type	Introduction
<val>	int	white balance temperature Range: 1000~10000. Adjustable precision: +-1

15.4.2.4 isp_get_wb_temperature

Table 15-34 ISP API: isp_get_wb_temperature

Parameter	Type	Introduction
<pval>	Int *	Retrieve the current white balance temperature

15.4.2.5 isp_set_red_balance

Table 15-35 ISP API: isp_set_red_balance

Parameter	Type	Introduction
<val>	int	Red balance value based on 256. Range: 256~2047. Adjustable precision: +-1.

15.4.2.6 isp_get_red_balance

Table 15-36 ISP API: isp_get_red_balance

Parameter	Type	Introduction
<pval>	Int *	Retrieve the red balance value.

15.4.2.7 isp_set_green_balance

Table 15-37 ISP API: isp_set_green_balance

Parameter	Type	Introduction
<val>	int	Green balance value based on 256. Range: 256~2047.

		Adjustable precision: +-1

NOTE

Usually this value is set 256 as default.

15.4.2.8 isp_get_green_balance

Table 15-38 ISP API: isp_get_green_balance

Parameter	Type	Introduction
<pval>	Int *	Retrieve the green balance value

15.4.2.9 isp_set_blue_balance

Table 15-39 ISP API: isp_set_blue_balance

Parameter	Type	Introduction
<val>	int	Blue balance value based on 256. Range: 256~2047. Adjustable precision: +-1.

15.4.2.10 isp_get_blue_balance

Table 15-40 ISP API: isp_get_blue_balance

Parameter	Type	Introduction
<pval>	Int *	Retrieve the blue balance value.

15.4.3 ISP Control API (Image Tuning)

15.4.3.1 isp_set_brightness

Table 15-41 ISP API: isp_set_brightness

Parameter	Type	Introduction
<val>	int	The brightness value of the image. Range: -64 to 64. Adjustable precision: +-1.

15.4.3.2 isp_get_brightness

Table 15-42 ISP API: isp_get_brightness

Parameter	Type	Introduction
<pval >	int *	Retrieves the current brightness value. Range: -64 to 64.

15.4.3.3 isp_set_contrast

Table 15-43 ISP API: isp_set_contrast

Parameter	Type	Introduction
<val>	int	image contrast value. Range: 0~100. Adjustable precision is +-1.

15.4.3.4 isp_get_contrast

Table 15-44 ISP API: isp_get_contrast

Parameter	Type	Introduction
<pval>	Int *	Get the current contrast value. Range: 0~100.

15.4.3.5 isp_set_saturation

Table 15-45 ISP API: isp_set_saturation

Parameter	Type	Introduction
<val>	int	ISP saturation. Range: 0 to 100. Adjustable accuracy: +-1.

15.4.3.6 isp_get_saturation

Table 15-46 ISP API: isp_get_saturation

Parameter	Type	Introduction
<pval>	Int *	Get the current saturation. Range: 0 to 100.

15.4.3.7 isp_set_gamma

Table 15-47 ISP API: isp_set_gamma

Parameter	Type	Introduction
<val>	int	Gamma coefficient. Range: 100~500. Adjustable precision: +-1.

15.4.3.8 isp_get_gamma

Table 15-48 ISP API: isp_get_gamma

Parameter	Type	Introduction
<pval>	Int *	Retrieve the current Gamma coefficient from 100~500.

15.4.3.9 isp_set_sharpness

Table 15-49 ISP API: isp_set_sharpness

Parameter	Type	Introduction
<val>	int	Sharpness of isp Range: 0~100. Adjustable precision: +-1

15.4.3.10 isp_get_sharpness

Table 15-50 ISP API: isp_get_sharpness

Parameter	Type	Introduction
<pval>	Int *	Retrieve the current sharp value from 0 to 100

15.4.3.11 isp_set_denoise_level

Table 15-51 ISP API: isp_set_denoise_level

Parameter	Type	Introduction

<val>	int	The level of noise reduction. Range: 0~8 Adjustable precision: +-1

15.4.3.12 isp_get_denoise_level

Table 15-52 ISP API: isp_get_denoise_level

Parameter	Type	Introduction
<pval>	Int *	Retrieve the level of noise reduction. Range: 0~8

15.4.4 ISP Control API (Mode)

15.4.4.1 isp_set_day_night

Table 15-53 ISP API: isp_set_day_night

Parameter	Type	Introduction
<val>	int	The value of day/night/other mode. 0: day mode, 1: night mode, 2: other

15.4.4.2 isp_get_day_night

Table 15-54 ISP API: isp_get_day_night

Parameter	Type	Introduction
<pval>	Int *	Retrieve the value of day/night/other mode. 0: day mode, 1: night mode, 2: other

15.4.4.3 isp_set_gray_mode

Table 15-55 ISP API: isp_set_gray_mode

Parameter	Type	Introduction
<val>	int	The value of gray/color mode. 0: color mode, 1: gray mode

15.4.4.4 isp_get_gray_mode

Table 15-56 ISP API: isp_get_gray_mode

Parameter	Type	Introduction
<pval>	Int *	Retrieve the value of gray/color mode. 0: color mode , 1: gray mode

15.4.5 ISP Control API (WDR)

15.4.5.1 isp_set_wdr_mode

Table 15-57 ISP API: isp_set_wdr_mode

Parameter	Type	Introduction
<val>	int	WDR mode. Range: 0 ~ 2 0: Disable, 1: Manual, 2: Auto

15.4.5.2 isp_get_wdr_mode

Table 15-58 ISP API: isp_get_wdr_mode

Parameter	Type	Introduction
<pval>	Int *	Retrieve the value of WDR mode. Range: 0 ~ 2

15.4.5.3 isp_set_wdr_level

Table 15-59 ISP API: isp_set_wdr_level

Parameter	Type	Introduction
<val>	int	WDR level. Range: 0~100. Adjustable precision: +-1

15.4.5.4 isp_get_wdr_level

Table 15-60 ISP API: isp_get_wdr_level

Parameter	Type	Introduction
<pval>	Int *	Retrieve the value of WDR level. Range: 0~100.

15.4.6 ISP Control API (Dehaze)

15.4.6.1 isp_set_dehaze

Table 15-61 ISP API: isp_set_dehaze

Parameter	Type	Introduction
<val>	int	The value of enable/disable mode. 0: disable, 1: enable

15.4.6.2 isp_get_dehaze

Table 15-62 ISP API: isp_get_dehaze

Parameter	Type	Introduction
<pval>	Int *	Retrieve the value of enable/disable mode. 0: disable , 1: enable

15.4.6.3 isp_set_dehaze_level

Table 15-63 ISP API: isp_set_dehaze_level

Parameter	Type	Introduction
<val>	int	The level of dehaze. Range: 0~255 Adjustable precision: +-1

15.4.6.4 isp_get_dehaze_level

Table 15-64 ISP API: isp_get_dehaze_level

Parameter	Type	Introduction
<pval>	Int *	Retrieve the level of dehaze.

		Range: 0~255

15.4.7 Lens evaluation

For lens performance evaluation, user may need to configuration isp. And we have prepared quick start guide.

Table 15-65 ISP API: Lens evaluation flow

ISP API	Description	Flow
isp_set_exposure_mode	0: Manual, 1:Auto	For golden lens, use auto mode to get AE & AWB information.
isp_set_awb_ctrl	0: Manual, 1:Auto	
isp_get_exposure_time	Exposure (unit: us)	
isp_get_ae_gain	Gain (unit: 256=1x)	
isp_get_red_balance	Gain (unit: 256=1x)	
isp_get_blue_balance	Gain (unit: 256=1x)	
isp_set_exposure_time	Exposure (unit: us)	For competitor lens, use manual mode and set AE & AWB information.
isp_set_ae_gain	Gain (unit: 256=1x)	
isp_set_red_balance	Gain (unit: 256=1x)	
isp_set_blue_balance	Gain (unit: 256=1x)	

15.5 Motion Detection

Motion detection is achieved by computing YRGB information.

15.5.1 Motion Detection Architecture

Motion detection architecture is shown in Figure 15-7. First, it will confirm whether the value of the automatic exposure (AE) is stable. After stabilization, get 32x32 YRGB value to initialize the background model. The 32x32 YRGB value is calculated by averaging the RGB image generated by from video channel 4. After initialization, calculate the difference between the YRGB values and the background model and the average difference of the entire image. Use the difference information to determine whether to trigger motion detection and update the background model immediately.

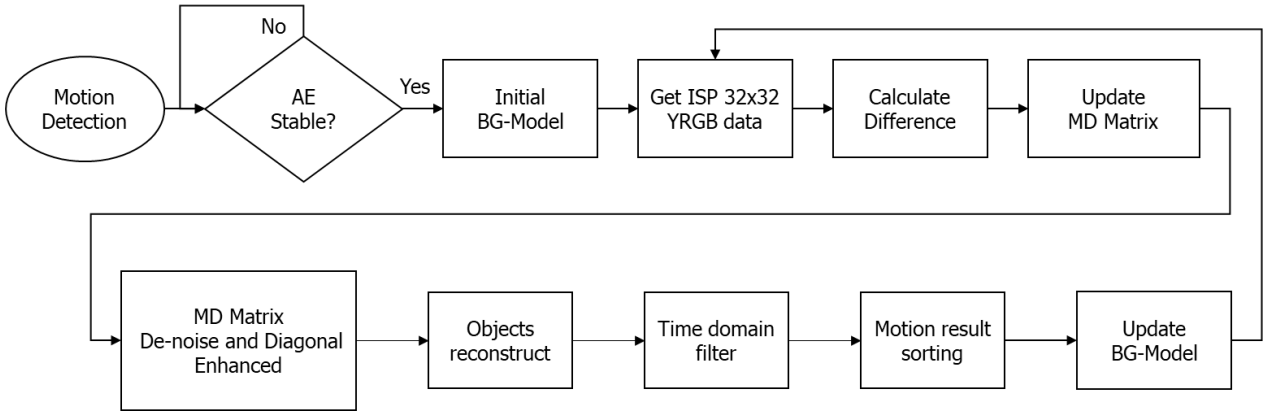


Figure 15-7 Motion detection architecture

15.5.1.1 YRGB 32x32 Data

The 32x32 YRGB value is calculated by averaging the RGB image generated by from video channel 4. Each value corresponds to the average YRGB value for dividing the image into a 32x32 frame. As shown in Figure 15-8. Calculating motion with 32x32 YRGB data has some advantages, such as (1) saving computation time, (2) taking into account color information, (3) filtering out noise

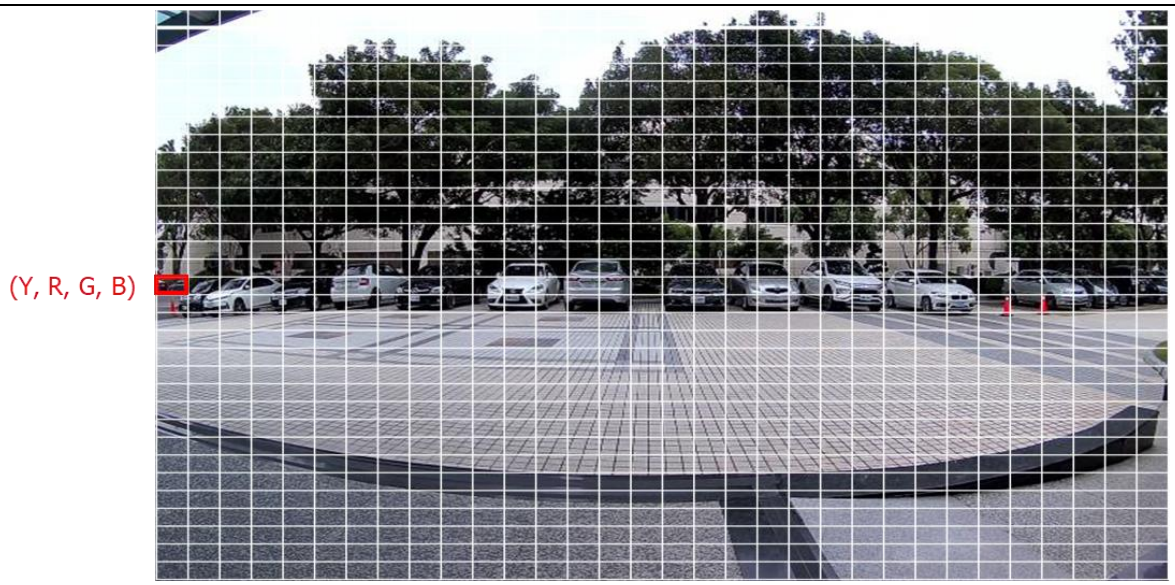


Figure 15-8 Average YRGB value for each block are calculated

15.5.1.2 Background Model

The calculation method of the background model is to calculate the average value of the current YRGB value and the recorded background model. This method can preserves background features, improve the sensitivity of motion detection, and update the background model in real time to avoid the problem of false alarm caused by the background change.

15.5.1.3 MD Matrix Calculation

The motion information is obtained by calculating the difference and average difference between the YRGB value of each frame and the background model. When the YRGB value difference is greater than the threshold, motion detection is triggered. The threshold is not a fixed value, but dynamically set with reference to the average difference value of each frame.

Usually the same difference through whole image is caused by noise or light change. By calculating the average difference value of a frame, the area with the difference smaller than the average difference can be filtered out. Take Figure 15-9 as an example. Initially, all the background value are 1. Motion occurred in the black region, and light change simultaneously. By calculating the average difference change of the whole image, we get average difference of 1.02. After filtering out differences less than 1.02, we get a motion detection matrix that shows where the actual motion occurred, as shown in the black areas in in Figure 15-9.

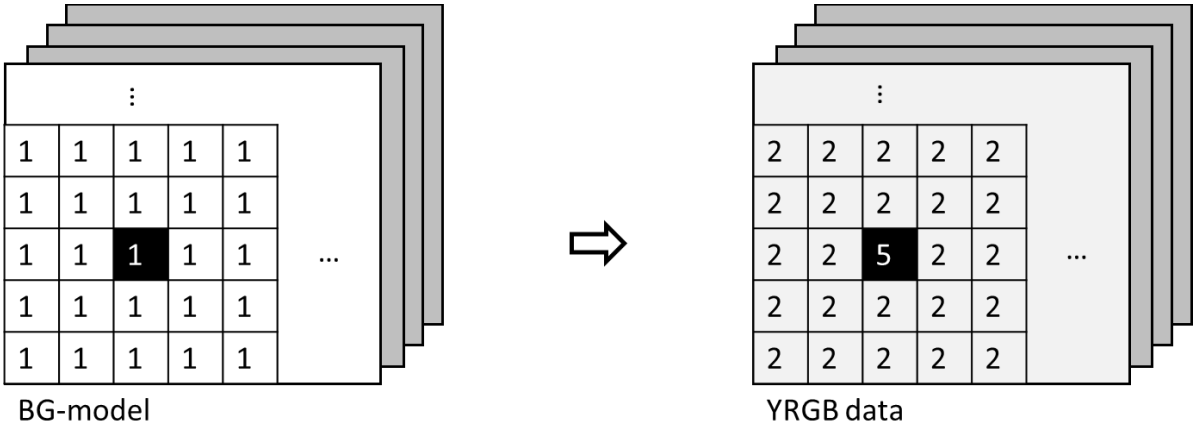


Figure 15-9 YRGB difference calculation

15.5.1.4 MD Matrix Post-Processing

After obtaining the motion detection matrix, the matrix will be de-noised and diagonally enhanced. Then, we will reconstruct motion detection matrix into several motion objects. We will filter out motion objects that do not overlap with any motion objects in the last motion result. This removes motion objects that move too fast, such as dust flying. Finally, we will sort the motion detection result according to the area size of the objects.

15.5.2 MD Module

The context of the md module shows as following:

```
typedef struct md_context_s {
 int count;
 int md_trigger_block;
 int AE_stable;
 char md_mask[MD_MAX_COL * MD_MAX_ROW];

 float md_adapt_thr[MD_MAX_COL * MD_MAX_ROW];
 int md_adapt_thr_raise_window[MD_FPS][MD_MAX_COL * MD_MAX_ROW];

 uint8_t md_his_idx[MD_MAX_COL * MD_MAX_ROW];
 uint8_t *md_his_background[MD_MAX_COL * MD_MAX_ROW];

 short *md_sensitivity_map;
 md result t md result;
 group_obj_t _objs[MD_MAX_TIME_FILTER_INTERVAL][MD_MAX_GOBJ];
 int  result idx;
 int _event_windows[MD_FPS];
 int _event_idx;

 motion_detect_bgmodel_t md_bgmodel;
 eip_YRBG_data_t YRGB_data;
 motion detect threshold t *md threshold;
} md_context_t;
…
typedef struct md_ctx_s {
 void *parent;
 md param t params;
 md_config_t md_config;
 md context t *motion detect ctx;
 md_disp_postprcess disp_postproc;
 unsigned long md_time0;
 bool md_out_en;
 int md_status;
} md_ctx_t;
```

Description of parameter in md_context_t:

- md_result: motion detection result structure.
- md_mask: motion detection mask array. set 0: disable MD, 1: enable MD
- en_AE_stable: check if auto explosion is stable, before initialize background model. set 0: disable , 1: enable.
- md_trigger_block_threshold: md triggered when at least n motion block triggered
- md_obj_sensitivity : motion detection threshold. The larger the value is set, the smaller the motion can be detected.
- md_bgmodel: current background model
- YRGB_data: current YRGB value
- disp_postproc: Set the callback function for display the MD result on video frame.

15.5.2.1 Set MD Resolution Parameter

User can set MD parameters by using CMD_MD_SET_PARAMS:

```
typedef struct md_param_s {
 int image_width;
 int image_height;
 int md row;
 int md_col;
} md_param_t;

md_ctx  = mm_module_open(&md_module);
if (md_ctx) {
    …
    mm module ctrl(md ctx, CMD MD SET PARAMS, (int)&md param);
}
```

- image_width: input frame width resolution. Please set to the same width resolution of video channel 4.
- image_height: input frame height resolution. Please set to the same height resolution of video channel 4.
- md_col: motion detection width resolution. Supports setting the value to 32.

- md_row: motion detection height resolution. Supports setting the value to 32.

Accelerated computing is supported for input images with specific resolutions and md resolution 32x32. The input image resolution that supports acceleration are 640x480, 640x360, 576x320, 416x416, 320x180. The CPU utilization of motion detection calculation for different resolution in pure sdk are shows as following. However, the performance of motion detection CPU utilization will be affected if the system is busy.

Table 15-66 MD CPU Utillization

Input Image Resolution	MD Resolution	Video Channel 4 FPS	CPU Utilization
640x480	32x32	10	9%
640x360	32x32	10	7%
576x320	32x32	10	8%
416x416	32x32	10	8%
320x180	32x32	10	5%

15.5.2.2 Set MD Configuration

User can set MD parameters by using CMD_MD_SET_MD_CONFIG:

```
typedef struct md_config_s {
 //adaptive parameter
 int adapt_mode; //0: off, 1: normal mode, 2: raising mode
 float adapt_level;
 int adapt_step; //adapt step frames

 //his parameter
 int his_threshold; //40 ~ 70
 int his_resolution; //4 or 5 //cannot set in run time

 int en_AE_stable; //0: off, 1: on
 int bg_mode; //0: normal, 1: de-noise

 int detect_interval; //detect every n frames
 int md_time_filter_interval; // 3 ~ 5 //filter fast motion that cannot be tract in n
frames
 int md_trigger_block_threshold; // motion blocks to trigger nn
 int md_obj_sensitivity;    // 0~100
} md_config_t;

md_ctx  = mm_module_open(&md_module);
if (md_ctx) {
    …
    mm_module_ctrl(md_ctx, CMD_MD_SET_MD_CONFIG, (int)&md_config);
}
```

- adapt_mode: adaptive mode. Adaptive mode will be replaced with histogram setting, so it is off by default. Set 0: disable adaptive mode, 1: normal adaptive mode, 2: raising adaptive mode. When adaptive mode is enable, the block difference threshold will gradually increase until adapt_level times the average difference.
- adapt_level: adapt level.
- adapt_step: adapt step frames.
- his_threshold: background histogram filter threshold. It should be set between 40 to 70. The background color will be stored in a histogram. When the frequency of the color of the motion trigger block exceeds the background histogram threshold, it will be filtered a s a dynamic background. Normally used for filtering fan and leaves shaving.
- his_resolution: background histogram resolution settings. Only support to set 4 or 5. The value can only be set before running. With the higher resolution, motion detection will be more sensitivity to color change.
- en_AE_stable: wait for automatic exposure (AE) is stabilize. Set 0: disable, 1: enable
- bg_mode: background update mode. Set 0: normal update mode, 1: de-noise update mode.
- detect_interval: detect every n frames.
- md_time_filter_interval: motion detection time filter interval. The value should be set to 3~5. Fast motion that cannot be tracked within n frames will be filtered out with this setting.
- md_trigger_block_threshold: motion blocks threshold settings to trigger cascaded NN module.
- md_obj_sensitivity: motion detection sensitivity settings.

15.5.2.3 Set MD Sensitivity

User can set MD sensitivity value by using CMD_MD_SET_MD_SENSITIVITY:

```
int md_sensitivity_value = 80;
md_ctx  = mm_module_open(&md_module);
if (md_ctx) {
    …
    mm_module_ctrl(md_ctx, CMD_MD_SET_MD_SENSITIVITY, md_sensitivity_value);
}
```

The sensitivity of motion detection can be set in the range of 0~100. Each sensitivity value corresponding to md trigger block count, as shown in Table 15-67.

Table 15-67 MD Sensitivity Mapping

RTK sensitivity	Trigger Block Count									
100-91	2	2	2	2	2	2	2	2	3	4
90-81	5	5	6	6	7	8	9	10	11	12
80-71	13	14	16	17	18	20	21	23	24	26
70-61	27	29	31	32	34	36	38	40	42	44
60-51	46	48	50	53	55	57	60	62	65	67
50-41	70	72	75	78	81	83	86	89	92	95
40-31	98	101	104	108	111	114	118	121	124	128
30-21	131	135	139	142	146	150	154	158	162	166
20-11	170	174	178	182	186	191	195	199	204	208
10-1	213	217	222	227	231	236	241	246	251	256

It can detect small movements when the sensitivity is set to a larger value. Under normal conditions, we recommend setting sensitivity to 83 to detect person movement within 9m. Set sensitivity to 78 to detect person movement within 6m. Set sensitivity to 56 to detect person movement within 3m. However, the performance of motion detection will be affected by the camera placement angle and image distortion, it should be tested on a case-by-case basis.

15.5.2.4 Get MD Sensitivity

User can get MD sensitivity value by using CMD_MD_GET_MD_SENSITIVITY:

```
int md_sensitivity_value;
if (md_ctx) {
    mm_module_ctrl(md_ctx, CMD_MD_GET_MD_SENSITIVITY, (int)&md_sensitivity_value);
}
```

15.5.2.5 Get MD Result

The context of the motion detection result structure shows as following:

```
typedef struct md_pos_s {
 float xmin, ymin, xmax, ymax;
} md_pos_t;

typedef struct md_result_s {
 char matrix[MD_MAX_COL * MD_MAX_ROW];
 int motion_cnt;
 md_pos_t md_pos[MD_MAX_GOBJ];
 int event_cnt;
} md_result_t;
```

- motion_cnt: number of motion detection objects.
- md_pos: motion detected object position.
- event_cnt: number of motion detection events in one second.

User can get MD result value by using CMD_MD_GET_MD_RESULT. It will return a motion detection result structure.

```
md_result_t res;
if (md_ctx) {
 mm_module_ctrl(md_ctx, CMD_MD_GET_MD_RESULT, (int)&res);
 for (int i = 0; i < res.count; i++) {
        int xmin = (int)(res.md_pos[i].xmin * RTSP_WIDTH);
        int ymin = (int)(res.md_pos[i].ymin * RTSP_HEIGHT);
        int xmax = (int)(res.md_pos[i].xmax * RTSP_WIDTH);
        int ymax = (int)(res.md_pos[i].ymax * RTSP_HEIGHT);
        printf("%d: x(%d,%d), y(%d,%d)\r\n", i, xmin, xmax, ymin, ymax);
```

```
 }
}
```

15.5.2.6 Set MD Wait AE Stable

To avoid false motion detection triggers, user can enable wait for automatic exposure (AE) is stabilize by using CMD_MD_EN_AE_STABLE:

```
if (md_ctx) {
    mm_module_ctrl(md_ctx, CMD_MD_EN_AE_STABLE, 1);
}
```

15.5.2.7 Set MD Adaptive Threshold Mode

Enable to adaptive threshold mode to reduce false alarm caused by repetitive motion, such as fan motion and rain motion. Note that enabling adaptive threshold mode makes motion detection less sensitive. Set mode = 0, disable adaptive threshold mode. Set mode = 1, enable normal adaptive threshold mode. Set mode = 2, enable strong adaptive threshold mode, which filter repetitive motion faster than normal mode.

```
if (md_ctx) {
    mm_module_ctrl(md_ctx, CMD_MD_SET_ADAPT_THR_MODE, 1);
}
```

15.5.2.8 Set MD BG-model Update Mode

User can change BG-model update mode by using CMD_MD_SET_BGMODE. Set mode = 0, enable normal update mode. Set mode = 1, enable smart update mode, which can detect if there is motion detected and choose a better way to update background model. In the case of mode 1, it can better filter out the dust of night mode.

```
if (md_ctx) {
    mm_module_ctrl(md_ctx, CMD_MD_SET_BGMODE, 1);
}
```

15.5.2.9 Set MD Mask

User can disable motion detect in desired region by setting motion detection mask array. Setting 0 to disable MD, 1 to enable MD.
Take care lane image as an example. If we want to filter out the car motion when far from camera, we can disable MD in the upper part of the image, as shown in Figure 15-10.

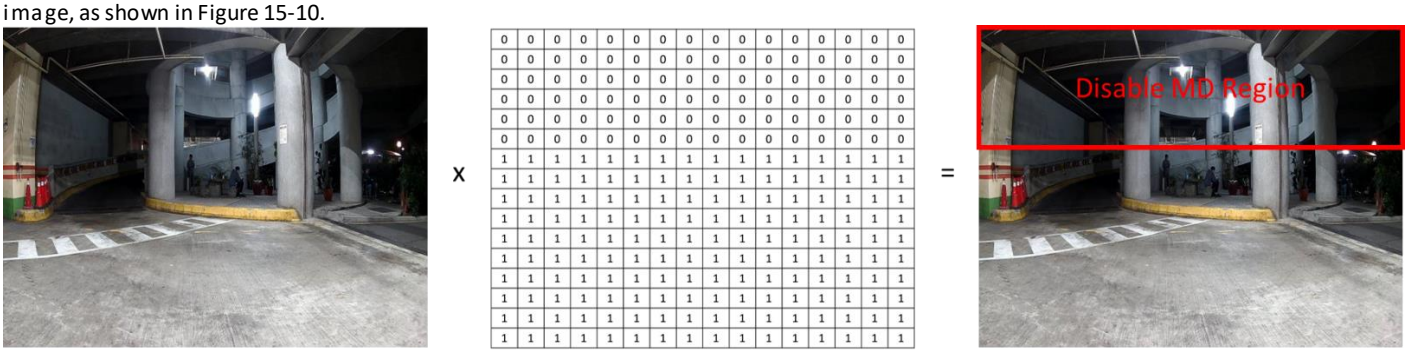


Figure 15-10 MD Mask

User can set MD mask value by using CMD_MD_SET_MD_MASK.

```
char md_mask [MD_COL * MD_ROW] = {0};
for (int i = 0; i < MD_COL * MD_ROW; i++) {
 md_mask[i] = 1;
}
if (md_ctx) {
    mm_module_ctrl(md_ctx, CMD_MD_SET_MD_MASK, (int)&md_mask);
}
```

15.5.2.10 Get MD Mask

User can get MD mask value by using CMD_MD_GET_MD_MASK:

```
char md_mask [MD_COL * MD_ROW];
if (md_ctx) {
    mm_module_ctrl(md_ctx, CMD_MD_GET_MD_MASK, (int)&md_mask);
}
```

15.5.2.11 Set MD Trigger Block Threshold

User can set motion detect trigger block threshold by using CMD_MD_SET_TRIG_BLK.
This command only used in motion trigger NN example. Trigger NN detection when at least n motion block are triggered

```
if (md_ctx) {
    …
    mm_module_ctrl(md_ctx, CMD_MD_SET_TRIG_BLK, 3);
}
```

15.5.2.12 Set MD result display callback function

User can register a call back function to so display the MD result. Use CMD_MD_SET_DISPPOST to set up callback function for display the MD result:

```
static void md_process(void *md_result)
{
    md_result_t *md_res = (md_result_t *) md_result;
    /* Process or display the result here */
}
if (md ctx) {
    …
    mm_module_ctrl(md_ctx, CMD_MD_SET_DISPPOST, (int)md_process);
}
```

15.5.3 MD Example

The MD example is a part of mmf video joined example. Please uncomment the example want to execute.
(project/realtek_amebapro2_v0_example/src/mmfv2_video_example/ video_example_media_framework.c)

```
mmf2_video_example_md_rtsp_init();
//mmf2_video_example_md_nn_rtsp_init();
```

Table 15-68 MD example

Example	Description	Result
mmf2_video_example_md_rtsp_init	CH1 Video -> H264/HEVC -> RTSP CH4 Video -> RGB -> MD	RTSP video stream over the network. MD detect motion and draw the motion region to RTSP channel.
mmf2_video_example_md_nn_rtsp_init	CH1 Video -> H264/HEVC -> RTSP CH4 Video -> RGB -> MD -> NN	RTSP video stream over the network. MD module detect motion. If there is motion detected, it will trigger NN module to detect object and draw the bounding box to RTSP channel.

15.5.3.1 Build MD Example

Since it's a part of video mmf example, user should use the following command to generate the makefile.
Generate the makefile for the MD project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DVIDEO_EXAMPLE=ON
```

Then, use the following command to generate an image:

```
cmake --build . --target flash
```

After running the command above, you will get the flash_ntz.bin in "project\realtek_amebapro2_v0_example\GCC-RELEASE\build". Then, use the image tool to download it to AmebaPro2.

15.5.3.2 Build MD & NN Example

Since it's a part of video mmf example, user should use the following command to generate the makefile.
Generate the makefile for the MD project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DVIDEO_EXAMPLE=ON
```

If running mmf2_video_example_md_nn_rtsp_init example, use the following command to generate an image with NN model inside:

```
cmake --build . --target flash_nn
```

After running the command above, you will get the flash_ntz.nn.bin in "project\realtek_amebapro2_v0_example\GCC-RELEASE\build". Then,

use the image tool to download it to AmebaPro2.

15.5.3.3 Validate MD example

While running the example, you may need to configure WiFi connection by using these commands in uart terminal.

```
ATW0=<WiFi_SSID> : Set the WiFi AP to be connected
ATW1=<WiFi_Password> : Set the WiFi AP password
ATWC : Initiate the connection
```

If everything works fine, you should see the following logs. Motion detection result will show in logs.

```
[MD] MD_v12.1
[MD] his params 50, 5
[MD] time filter interval 3
Set MD Mask:
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
…
[VOE]RGB3 320x180 1/10
[VOE]status == 1718
[VOE]release s4 isp buffer 0
[VOE][WARN]useless release s4 slot0 status 0x00000000
[VOE]release s4 isp buffer 1
[VOE][WARN]useless release s4 slot1 status 0x00000000
font resize new size: 4840 byte-w:2 byte-h:32.
font resize new size: 3688 byte-w:4 byte-h:32.
font resize from 32 64 to 16 32.
font resize from 64 64 to 32 32.
font resize:21.
osd_update_custom_init Jun 14 2023
osd ch 0 e1 num 24 (0, 1, 2)
osd_render_task start
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
```



```
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[MD] FPS = 47.53
md initial
[MD] FPS = 10.00
[MD] FPS = 10.00
[MD] FPS = 10.00
```

If desire to see the motion detected region, set MD_DRAW to 1.

```
#define MD_DRAW 1
```

Then, open VLC (or PotPlayer) and create a network stream with URL: rtsp://192.168.x.xx:554
When motion detected, it will draw the motion detected region.

NOTE

Motion detection frame rate is fix to 10 FPS. Since the motion detection performance will be effected by frame rate, the actual motion detect frame rate will be shown in every 10 second. Please check if the motion detection frame rate is fix to 10 FPS.

Figure 15-11 Motion Detection Example

15.5.3.4 Validate MD & NN example

While running the example, you may need to configure WiFi connection by using these commands in uart terminal.

```
ATW0=<WiFi_SSID> : Set the WiFi AP to be connected
ATW1=<WiFi_Password> : Set the WiFi AP password
ATWC : Initiate the connection
```

If everything works fine, you should see the following logs. Motion detection result will show in logs.

```
[VOE]RGB3 416x416 1/10
[VOE]status == 1718
[VOE]release s4 isp buffer 0
[VOE][WARN]useless release s4 slot0 status 0x00000000
[VOE]release s4 isp buffer 1
[VOE][WARN]useless release s4 slot1 status 0x00000000
siso_md_nn started
font resize new size: 4840 byte-w:2 byte-h:32.
font resize new size: 3688 byte-w:4 byte-h:32.
font resize from 32 64 to 16 32.
font resize from 64 64 to 32 32.
font resize:21.
osd_update_custom_init Mar 29 2023
osd ch 0 e1 num 24 (0, 1, 2)
osd_render_task start
```

```
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
AE not sable
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
AE not sable
[VOE]isp ctrl 0x00980911 id 17
[VOE]isp_ctrl 0x00980913 id 19
[VOE]isp_ctrl 0x00980911 id 17
[VOE]isp ctrl 0x00980913 id 19
md initial
[MD] MD v10
Motion Detected
YOLOv4t tick[0] = 70
object num = 1
0,c0:1136 299 1374 1060
Motion Detected
YOLOv4t tick[0] = 75
object num = 2
0,c0:1132 299 1369 1060
1,c0:1594 317 1709 1048
```

Then, open VLC (or PotPlayer) and create a network stream with URL: rtsp://192.168.x.xx:554
When motion detected, it will trigger object detection, and draw the detection result

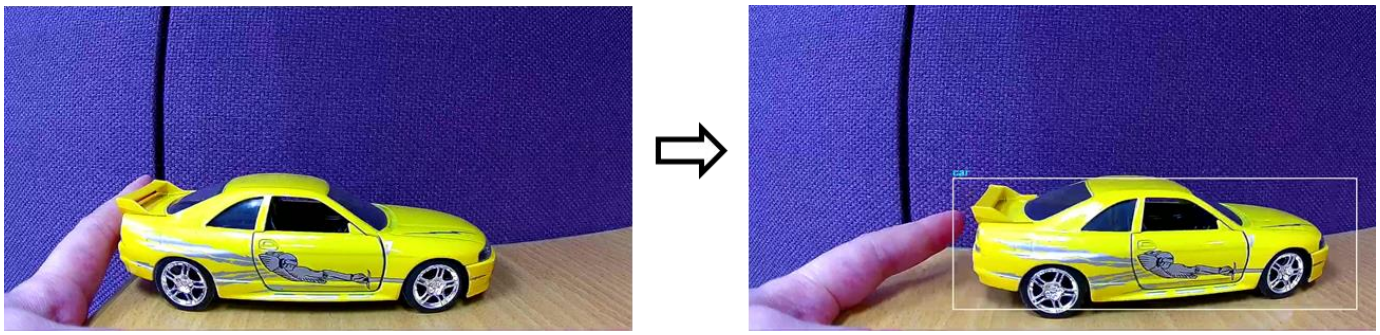


Figure 15-12 MD & NN Example

16 Bluetooth

16.1 BT Example

Ameba Pro2 BT examples provide a set of BT functionality, such as BT Config, BT Peripheral, BT Central and BT Beacon.
BT Config example demonstrates how to transfer SSID profile from Mobile Phone to device.
For more information, please refer to **UM0201 Ameba Common BT Application User Manual EN.pdf**.

This section illustrates how to build and run BT examples in our SDK, including GCC and IAR environment.

16.1.1 GCC Project

(1) Enter SDK path: project\realtek_amebapro2_v0_example\inc and modify file platform_opts_bt.h to enable BT.

```
#define CONFIG_BT 1 //This define must be enabled.
//Enable corresponding define for which example want to be used.
//Here using example bt peripheral for instance.
#define CONFIG_BT_CONFIG 0
#define CONFIG_BT_AIRSYNC_CONFIG 0
#define CONFIG_BT_PERIPHERAL 1
#define CONFIG_BT_CENTRAL 0
#define CONFIG_BT_SCATTERNET 0
#define CONFIG_BT_BEACON 0
```

(2) Build images

(3) Use ImageTool to download images to your board.

NOTE

You can select one BLE example or all the examples at once. If all the examples are selected at once, the integrated image can support all BLE test commands, and the scatternet configuration will display only when both peripheral and central are selected.

16.1.2 Examples List

16.1.2.1 ble_peripheral

This example shows how to create and run GATT service on GATT server.

16.1.2.1.1 Image Generation

(1) To run ble_peripheral example, turn on the following flags defined in \project\realtek_amebapro2_v0_example\inc\platform_opts_bt.h

```
#define CONFIG_BT 1
#if CONFIG_BT
#define CONFIG_FTL_ENABLED
#define CONFIG_BT_CONFIG 0
#define CONFIG_BT_PERIPHERAL 1
#define CONFIG_BT_CENTRAL 0
#define CONFIG_BT_SCATTERNET 0
#define CONFIG_BT_BEACON 0
#define CONFIG_BT_MESH_PROVISIONER 0
#define CONFIG_BT_MESH_DEVICE 0
```

(2) Build image and download image to your board.

16.1.2.1.2 Test Procedure

(1) After download image to your Ameba Pro2 board, reset it. The default device name is BLE_PERIPHERAL.

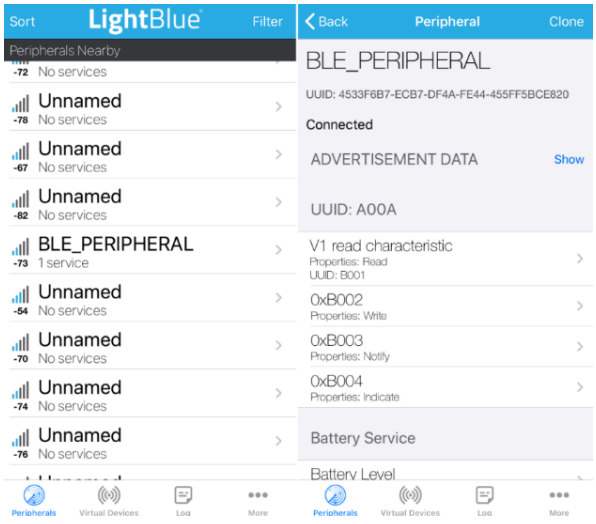
(2) Download apps such as “LightBlue” or “nRF Connect” and use as GATT Client to connect it.

(3) ATBp is an AT command for BT Peripheral. Using “ATBp=1” to initialize BT Peripheral stack, which can send advertising package out and scannable by other devices. Below is the BT peripheral example initialization success log.

```
hci_borad_controller_reset:346(info) BT Reset OK!
amebapro2_uart_set_bdrate:72(info)
Set baudrate to 921600 success!
[BLE peripheral] GAP stack ready local bd addr: 0x
[MEM] After do cmd, available heap 46760992
#
89:51:12:36:28:11
```

```
GAP adv start
```

(4) Search for BLE_PERIPHERAL device and connect to it.



16.1.2.2 ble_central

This example shows how to discover service on GATT server.

16.1.2.2.1 Image Generation

(1) To run ble_central example, turn on the following flags defined in \project\realtek_amebapro2_v0_example \inc\platform_opts_bt.h

```
#define CONFIG_BT 1
#if CONFIG_BT
#define CONFIG_FTL_ENABLED
#define CONFIG_BT_CONFIG 0
#define CONFIG_BT_PERIPHERAL 0
#define CONFIG_BT_CENTRAL 1
#define CONFIG_BT_SCATTERNET 0
#define CONFIG_BT_BEACON 0
#define CONFIG_BT_MESH_PROVISIONER 0
#define CONFIG_BT_MESH_DEVICE 0
```

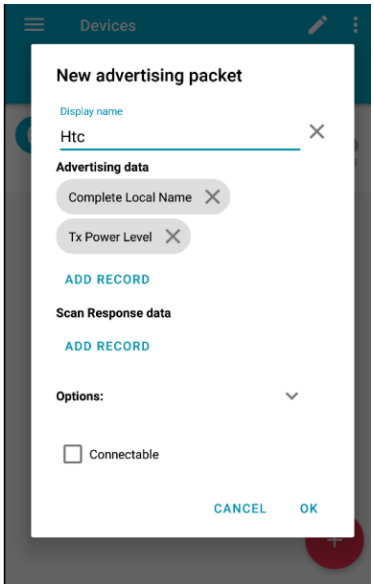
(2) Build image and download image to your board.

16.1.2.2.2 Test Procedure

(1) After download image to your AmebaPro2 board, reset it.

(2) Download app "nRF Connect" and use as GATT Server to be connected.

(3) Add new advertising packet and set its additional data.



(4) ATBc is an AT command for BT Central. Using "ATBc=1" to turn BT Central stack ON.

(5) Using "ATBS=1" to scan available BT devices nearby.

(6) Using "ATBC=P/R, BLE_BD_ADDR" to connect to the device.

BT Central scan and connect log:

```
#ATBS=1
Start scan, scan_filter_policy = 0, scan_filter_duplicate = 1 [MEM] After do cmd,
available heap 46756320
#
GAP scan start
ADVType | AddrType |BT_Addr |rssi
CON_UNDIRECT random 4f:6e:3e:75:56:2e -80
GAP_ADTYPE_FLAGS: 0x1a
GAP_ADTYPE_MANUFACTURER_SPECIFIC: company_id 0x4c, len 24
ADVType | AddrType |BT_Addr |rssi
CON_UNDIRECT random 70:20:ca:98:7a:88 -74
GAP_ADTYPE_FLAGS: 0x1a
GAP_ADTYPE_POWER_LEVEL: 0x18
GAP_ADTYPE_MANUFACTURER_SPECIFIC: company_id 0x4c, len 7
#ATBS=0 Stop scan
[MEM] After do cmd, available heap 46756320
# GAP scan stop
# ATBC=R, 665544778899 [MEM] After do cmd, available heap 46756320
# cmd_con, DestAddr: 0x66:0x55:0x44:0x77:0x88:0x99
```

For more AT commands used for BT Central, please refer to user manual **UM0201 Ameba Common BT Application User Manual EN.pdf**.

16.1.2.3 ble_scatternet

BLE Scatternet is the coexistence of BLE Central mode and BLE Peripheral mode. Once BLE Scatternet stack initialized, AT command of BLE Central and BLE Peripheral are available. This example shows how to turn BLE Scatternet on.

16.1.2.3.1 Image Generation

(1) To run ble_central example, turn on the following flags defined in \project\realtek_amebapro2_v0_example \inc\platform_opts_bt.h

```
#define CONFIG_BT 1
#if CONFIG_BT
#define CONFIG_FTL_ENABLED
#define CONFIG_BT_CONFIG 0
#define CONFIG_BT_PERIPHERAL 0
#define CONFIG_BT_CENTRAL 0
#define CONFIG_BT_SCATTERNET 1
#define CONFIG_BT_BEACON 0
#define CONFIG_BT_MESH_PROVISIONER 0
#define CONFIG_BT_MESH_DEVICE 0
```

(2) Build image and download image to your board.

16.1.2.3.2 Test Procedure

(1) After download image to your AmebaPro2 board, reset it.

(2) Using "ATBf=1" to turn BT Scatternet stack ON.

(3) Once see the following message, you can continue input other AT command of BT Scatternet mode as well as BT Central mode and BT Peripheral mode.

```
hci_borad_controller_reset:346(info) BT Reset OK!
amebapro2_uart_set_bdrate:72(info) Set baudrate to 921600 success!
local bd addr: 0x89:51:12:36:28:11
[MEM] After do cmd, available heap 46754528
#

GAP adv start
```

For other AT commands used for BT Scatternet, please refer to **UM0201 Ameba Common BT Application User Manual EN.pdf**.

16.1.2.4 bt_beacon

This example shows how to send BLE Beacons. AmebaPro2 provides two types of Beacon: Apple iBeacon and Radius Networks AltBeacons.

16.1.2.4.1 Image Generation

(1) To run ble_central example, turn on the following flags defined in \project\realtek_amebapro2_v0_example \inc\platform_opts_bt.h

```
#define CONFIG_BT 1
#if CONFIG_BT
#define CONFIG_FTL_ENABLED
#define CONFIG_BT_CONFIG 0
#define CONFIG_BT_PERIPHERAL 0
#define CONFIG_BT_CENTRAL 0
#define CONFIG_BT_SCATTERNET 0
#define CONFIG_BT_BEACON 1
#define CONFIG_BT_MESH_PROVISIONER 0
#define CONFIG_BT_MESH_DEVICE 0
```

(2) Build image and download image to your board.

16.1.2.4.2 Test Procedure

(1) Choose beacon type by using “ATBJ=1,1” or “ATBJ=1,2” command.

```
# ATBJ
[ATBJ] Start BT I_Beacon: ATBJ=1,1
[ATBJ] Start BT Alt_Beacon: ATBJ=1,2
[ATBJ] Stop BT Beacon: ATBJ=0
```

(2) You can use apps such as “LightBlue” or “nRF Connect” to observe beacons. “Locate” observe beacon by it adv UUID. Below screenshot is taken using Android “nRF Connect”.

16.1.2.5 bt_config

BT Config provides a simple way for Wi-Fi device to associate to AP easily.

16.1.2.5.1 Image Generation

(1) To run ble_central example, turn on the following flags defined in \project\realtek_amebapro2_v0_example \inc\platform_opts_bt.h

```
#define CONFIG_BT 1
#if CONFIG_BT
#define CONFIG_FTL_ENABLED
#define CONFIG_BT_CONFIG 1
#define CONFIG_BT_PERIPHERAL 0
#define CONFIG_BT_CENTRAL 0
#define CONFIG_BT_SCATTERNET 0
#define CONFIG_BT_BEACON 0
#define CONFIG_BT_MESH_PROVISIONER 0
#define CONFIG_BT_MESH_DEVICE 0
```

(2) Build image and download image to your board.

16.1.2.5.2 APP Installation

Search “Easy WiFi Config” in the application store. You can install Android or iOS as your phone OS.

16.1.2.5.3 Test Procedure

(1) ATBB is an AT command for BT Config. Using “ATBB=1” to enter BT Config mode, which allows BT Config APP to discover and connect to Ameba Pro2. Reset your AmebaPro2 board, and input command “ATBB=1”.

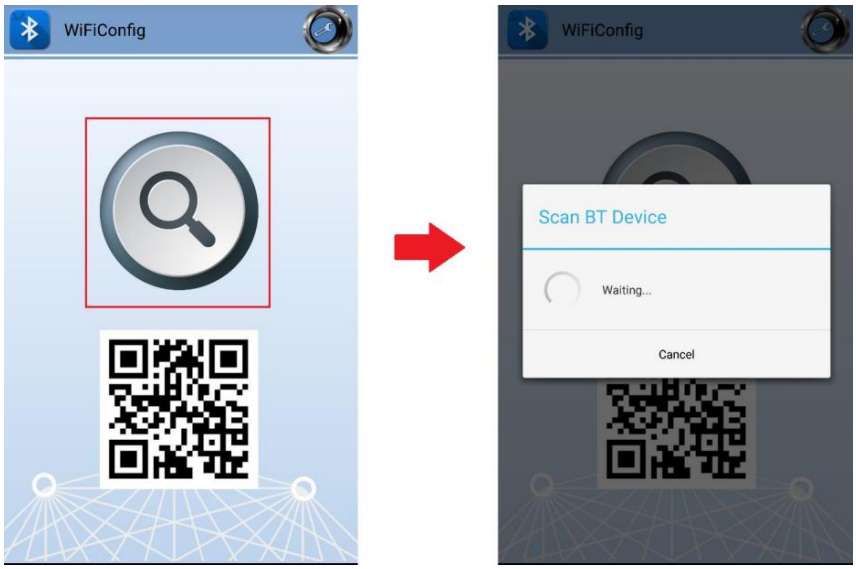
(2) Once see the following message, you can open BT Config APP to associate AP.

BT Initialize and start adv log:

```
[BT Config Wifi] BT Config Wifi ready
[BT Config Wifi] ADV started
```

(3) Click the BT config icon to launch it. Scan and connect with Ameba Pro2 BT using BT Config app.

Display on BT config app:

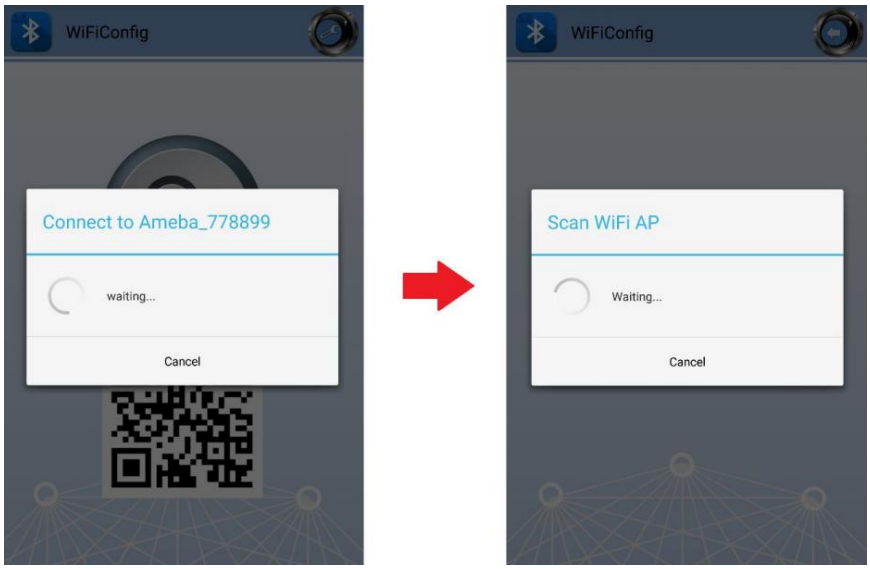


(4) Once BT Config APP connected to AmebaPro2, below log will be show. When connection is established AmebaPro2 will start searching for AP.

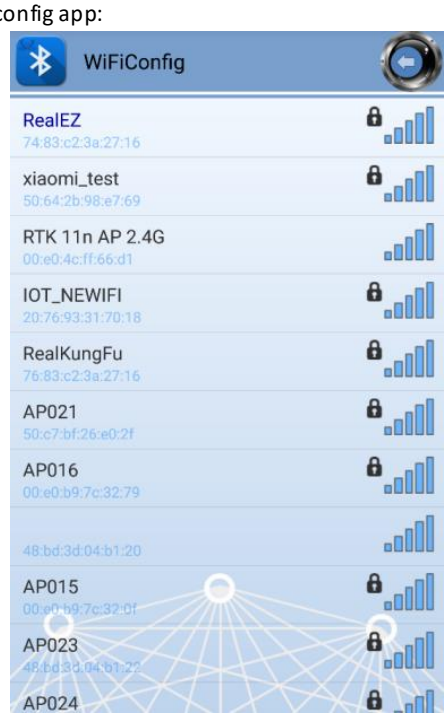
BT Connection log:

```
[BT Config Wifi] Bluetooth Connection Established
[BT Config Wifi] Band Request
[BT Config Wifi] Scan Request
[BT Config Wifi] Scan 2.4G AP
[BT Config Wifi] Scan 5G AP
```

Display on BT config app:



Scanned and reachable APs will be show on BT config app:

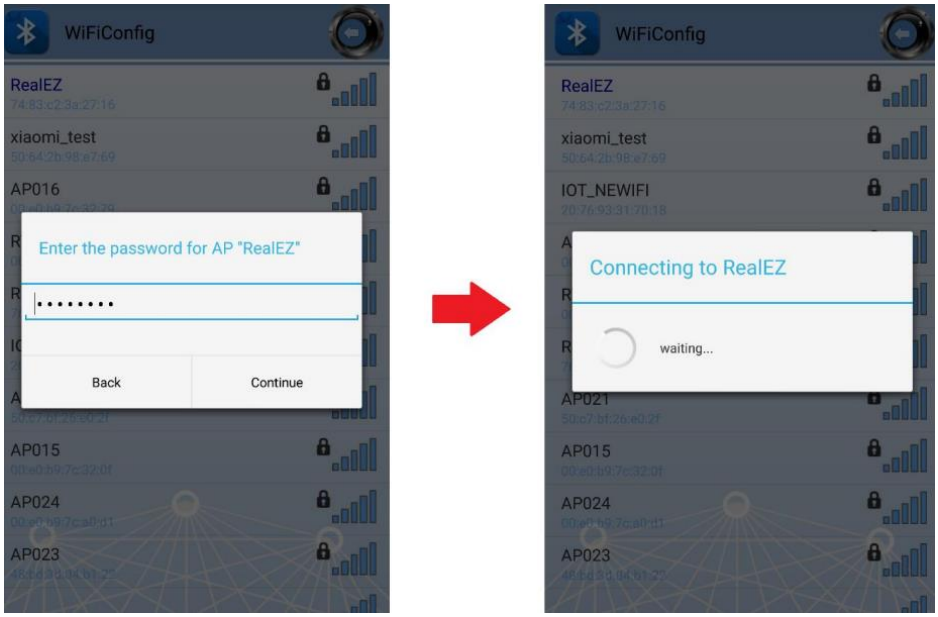


(5) Select an AP to connect to and input password (if any).

AP Connection log:

```
[BT Config Wifi] Connect Request
[Driver]: set BSSID: 90:94:e4:c5:d3:f0
[Driver]: set ssid [Test_ap]
[Driver]: start auth to 90:94:e4:c5:d3:f0
[Driver]: auth success, start assoc
[Driver]: association success(res=7)
[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES4)
[Driver]: set group key to hw: alg:2(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1
[BT Config Wifi] Connected after 3458ms.
Interface 0 IP address : 192.168.0.102 [BT Config Wifi]
Got IP after 3500ms.
```

Display on BT config app:

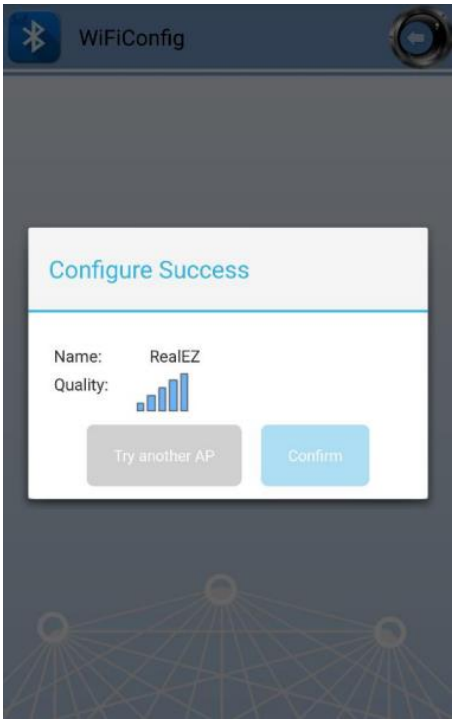


(6) When AmebaPro2 is connected to an AP, user can confirm connection or select another AP. Click “Confirm” to confirm AP connection. Click “Try another AP” to go back to Wi-Fi scan list page and choose another AP to connect to. After confirming BT config result, Bluetooth connection is disconnected, AmebaPro2 becomes undiscoverable to BT Config APP.

BT Disconnect log:

```
[BT Config Wifi] Bluetooth Connection Disconnected
[BT Config Wifi] ADV started
[BT Config Wifi] [BC_status_monitor] wifi connected, delete BC_cmd_task and
BC_status_monitor
[BT Config Wifi] ADV stopped
```

Display on BT config app:



(7) You can use "ATBB=1" to restart BT Config mode again.

Command	Introduction
ATBB=1	Start BT Config
ATBB=0	Stop BT Config

NOTE

Enter BT Config mode will disconnect existing Wi-Fi connection.
Please refer to BT Config APP User Guide for more details

16.2 BLE Security

If password input is planned to be integrated into the pairing process, BLE provide secure pairing procedure.

16.2.1 Security features

The security architecture of BLE has five distinct features: pairing, bonding, device authentication, encryption and message integrity.

- **Pairing** is the process for creating shared secret keys
- **Bonding** is to store the keys generated during pairing for use in subsequent connections in order to form a trusted device pair.
- **Device authentication** is to verify the two devices have the same keys
- **Encryption** is the process that provides message confidentiality
- **Message integrity** protects against message forgeries.

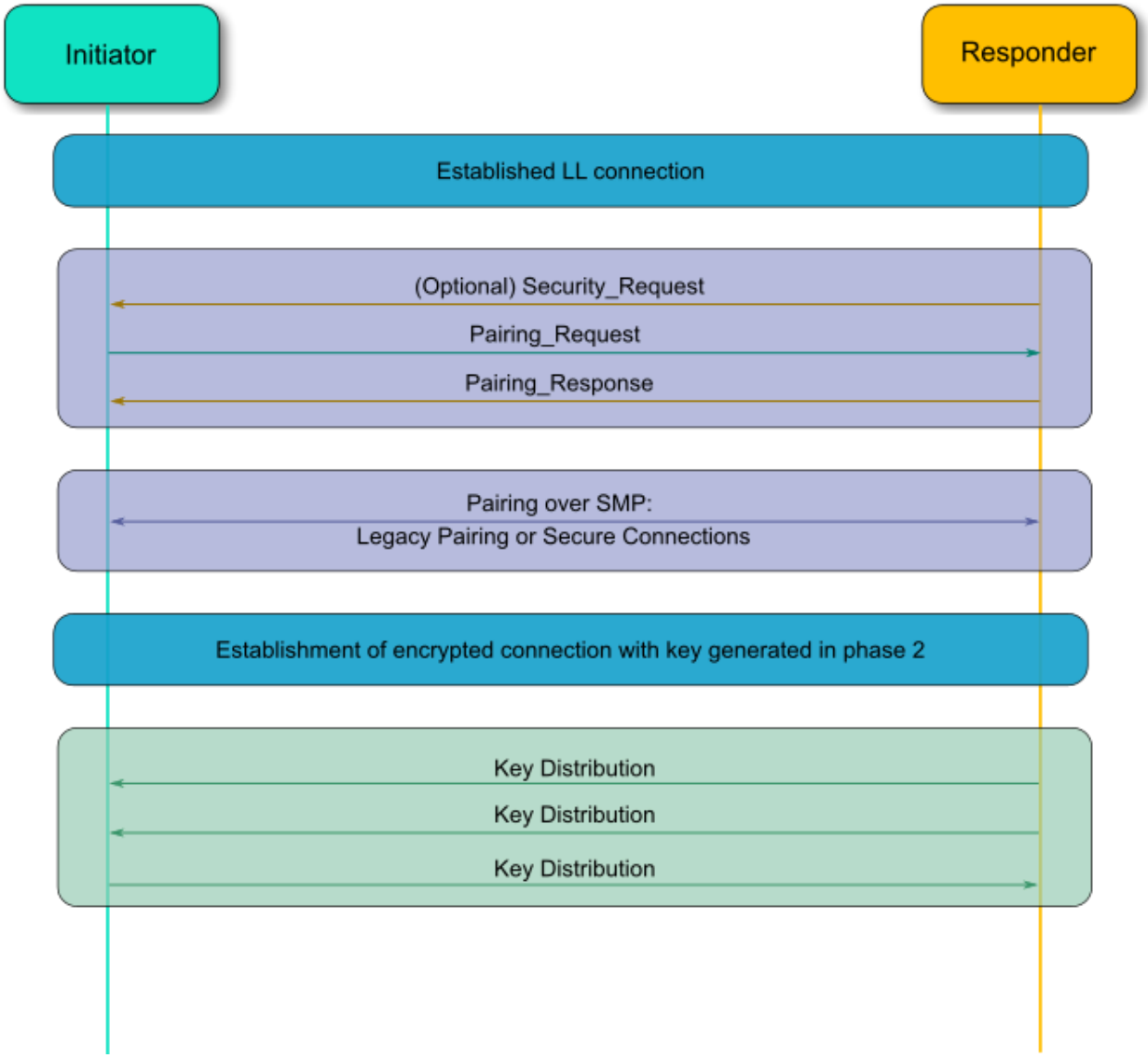
Pairing feature will be introduced in the following chapters.

16.2.2 Pairing

In BLE, pairing is used to generate keys and encrypt the connection is called pairing, and it consists with three phases:

- Phase 1: Pairing Feature exchange
- Phase 2 (LE Legacy Pairing): Short Term Key (STK) Generation
- Phase 2 (LE Secure Connections): Long Term Key (LTK) Generation
- Phase 3: Transport Specific Key Distribution

Pairing begins after the devices have connected with each other, after that they exchange information about their I/O capabilities. In the next phase the user will assist, if possible, for identification of the device, and based on whether we use LE Legacy Pairing or Secure Connections, the appropriate scheme will be used to generate the link key. All subsequent communications will be encrypted with this key. When the link is encrypted, specific keys will be exchanged that aid in resolving the private address of the device, or signing and authenticating the data.



16.2.3 Example

In the example, there are several variables related to security features.

- GAP_PARAM_BOND_PAIRING_MODE set Ameba Device is pairable or not

```
#define GAP_PAIRING_MODE_NO_PAIRING          0x00  //!< Pairing is not allowed.
#define GAP_PAIRING_MODE_PAIRABLE            0x01  //!< Pairable, Wait for a pairing request
from master or security request from slave.
```

In the example code, the default setting is:

```
uint8_t  auth_pair_mode = GAP_PAIRING_MODE_PAIRABLE;
gap_set_param(GAP_PARAM_BOND_PAIRING_MODE, sizeof(auth_pair_mode), &auth_pair_mode);
```

- GAP_PARAM_BOND_AUTHEN_REQUIREMENTS_FLAGS sets up pair mode and ability, default is **GAP_AUTHEN_BIT_BONDING_FLAG** which enable legacy pairing mode and enable bounding feature

```
#define GAP_AUTHEN_BIT_NONE                  0      //!<  No authentication required.
#define GAP_AUTHEN_BIT_BONDING_FLAG          0x0001 //!<  Bonding is required
#define GAP_AUTHEN_BIT_MITM_FLAG             0x0004 //!<  Mitm is preferred
#if F_BT_LE_4_2_SC_SUPPORT
#define GAP_AUTHEN_BIT_SC_FLAG               0x0008 //!<  Secure connection is preferred
```

```
#define GAP_AUTHEN_BIT_SC_ONLY_FLAG          0x0200 //!<  Secure connection only mode for BLE is
required
#endif
#define GAP_AUTHEN_BIT_FORCE_BONDING_FLAG    0x0100 //!<  Force bonding is required
```

In the example code, the default setting is:

```
uint16_t auth_flags = GAP_AUTHEN_BIT_BONDING_FLAG;
gap_set_param(GAP_PARAM_BOND_AUTHEN_REQUIREMENTS_FLAGS, sizeof(auth_flags), &auth_flags);
```

If secure connection is prefer, the user can add up auth_flags with **GAP_AUTHEN_BIT_SC_FLAG**:

```
uint16_t auth_flags = GAP_AUTHEN_BIT_BONDING_FLAG | GAP_AUTHEN_BIT_SC_FLAG;
gap_set_param(GAP_PARAM_BOND_AUTHEN_REQUIREMENTS_FLAGS, sizeof(auth_flags), &auth_flags);
```

- GAP_PARAM_BOND_IO_CAPABILITIES sets up I/O capabilities.

```
GAP_IO_CAP_DISPLAY_ONLY,        //!<  Only a Display present, no Keyboard or Yes/No Keys.
GAP_IO_CAP_DISPLAY_YES_NO,      //!<  Display and Yes/No Keys present.
GAP_IO_CAP_KEYBOARD_ONLY,       //!<  Only a Keyboard present, no Display.
GAP_IO_CAP_NO_INPUT_NO_OUTPUT,  //!<  No input/output capabilities.
GAP_IO_CAP_KEYBOARD_DISPLAY,    //!<  Keyboard and Display present.
```

In the example code, the default setting is:

```
uint8_t  auth_io_cap = GAP_IO_CAP_NO_INPUT_NO_OUTPUT;
gap_set_param(GAP_PARAM_BOND_IO_CAPABILITIES, sizeof(auth_io_cap), &auth_io_cap);
```

- GAP_PARAM_BOND_OOB_ENABLED sets up whether using OOB while pairing.

```
GAP_PARAM_BOND_OOB_ENABLED              = 0x205,//!< OOB data available for pairing algorithm.
Read/Write. Size is uint8_t. Default is 0(disabled).
```

In the example code, the default setting is:

```
#if F_BT_LE_SMP_OOB_SUPPORT
      uint8_t  auth_oob = false;
#endif
#if F_BT_LE_SMP_OOB_SUPPORT
      gap_set_param(GAP_PARAM_BOND_OOB_ENABLED, sizeof(auth_oob), &auth_oob);
#endif
```

- GAP_PARAM_BOND_FIXED_PASSKEY and GAP_PARAM_BOND_FIXED_PASSKEY_ENABLE is used to fix pass key.

```
GAP_PARAM_BOND_FIXED_PASSKEY        = 0x211,//!< The fix passcode for MITM protection.
Read/Write. size is uint32_t. Range is 0 - 999,999. Default is 0.

GAP_PARAM_BOND_FIXED_PASSKEY_ENABLE = 0x212,//!< The fix passcode available for pairing.
Read/Write. size is uint8_t. Default is 0(disabled).
```

In the example code, the default setting is:

```
uint8_t  auth_use_fix_passkey = false;
uint32_t auth_fix_passkey = 0;
le_bond_set_param(GAP_PARAM_BOND_FIXED_PASSKEY, sizeof(auth_fix_passkey), &auth_fix_passkey);
le_bond_set_param(GAP_PARAM_BOND_FIXED_PASSKEY_ENABLE, sizeof(auth_use_fix_passkey),
                  &auth_use_fix_passkey);
```

- GAP_PARAM_BOND_SEC_REQ_ENABLE and GAP_PARAM_BOND_SEC_REQ_REQUIREMENT automatically send Security_Request or not.

```
GAP_PARAM_BOND_SEC_REQ_ENABLE       = 0x213,//!< Send smp security request when connected.
Read/Write. size is uint8_t. Default is 0(disabled).

GAP_PARAM_BOND_SEC_REQ_REQUIREMENT  = 0x214,//!< Security request requirements. Read/Write. size
is uint8_t. Default is GAP_AUTHEN_BIT_BONDING_FLAG (@ref BOND_MITM_DEFINES)
```

In the example code, the default setting is:

```
uint8_t  auth_sec_req_enable = false;
uint16_t auth_sec_req_flags = GAP_AUTHEN_BIT_BONDING_FLAG;

le_bond_set_param(GAP_PARAM_BOND_SEC_REQ_ENABLE, sizeof(auth_sec_req_enable),
                  &auth_sec_req_enable);
le_bond_set_param(GAP_PARAM_BOND_SEC_REQ_REQUIREMENT, sizeof(auth_sec_req_flags),
                  &auth_sec_req_flags);
```

If secure connection is prefer, the user can add up auth_flags with **GAP_AUTHEN_BIT_SC_FLAG**:

```
uint8_t  auth_sec_req_enable = true;
uint16_t auth_sec_req_flags = GAP_AUTHEN_BIT_BONDING_FLAG | GAP_AUTHEN_BIT_SC_FLAG;
le_bond_set_param(GAP_PARAM_BOND_SEC_REQ_ENABLE, sizeof(auth_sec_req_enable),
                  &auth_sec_req_enable);
le_bond_set_param(GAP_PARAM_BOND_SEC_REQ_REQUIREMENT, sizeof(auth_sec_req_flags),
                  &auth_sec_req_flags);
```

17 System Resource Evaluation

This section will explain how to calculate used system resource in GCC project.

17.1 Memory Section

We can find the memory configuration in "rtl8735b_ram.ld":

- .ram.code_rodata : This section is read-only data in SRAM
- .ram.data : This section is read-write data in SRAM
- .ram.bss : This section is data has no initial values in SRAM
- .ddr.rodata : This section is read-only data in DDR Memory
- .ddr.data : This section is read-write data in DDR Memory
- .ddr.bss : This section is data has no initial values in DDR Memory

17.2 Memory Size

User can refer "application.ntz.map" to observe them after project build. This file can be found in "project/realtek_amebapro2_v0_example/GCC-RELEASE/build/application" folder.

17.2.1 Memory Size in SRAM

There are several sections in SRAM, including ".ram.code_text", ".ram.data", ".ram.code_rodata", ".ram.bss" and ".non_secure.bss". We can sum up these sections:

".ram.code_text" has size 0xdef0:

```
.ram.code_text  0x0000000020100b00                0xdef0
                0x0000000020100b00                 . = ALIGN (0x4)
                0x0000000020100b00                   etext2 = .
                0x0000000020100b00                 . = ALIGN (0x20)
                0x0000000020100b00                 __ram_entry_text_start__ = .
```

".ram.data" has size 0x9b4:

```
.ram.data       0x000000002010e9f0                 0x9b4
                0x000000002010e9f0                 __fw_img_start__ = .
                0x000000002010e9f0                 __etext = .
                0x000000002010e9f0                 __data_start__ = .
```

".ram.code_rodata" has size 0x5d8:

```
.ram.code_rodata
                0x000000002010f3a8                 0x5d8
                0x000000002010f3a8                 . = ALIGN (0x4)
                0x000000002010f3a8                 __ram_code_rodata_start__ = .
```

".ram.bss" has size 0x139fc:

```
.ram.bss        0x000000002010f980                 0x139fc
                0x000000002010f980                 . = ALIGN (0x4)
                0x000000002010f980                 __bss_start__ = .
```

".non_secure.bss" has size 0x4:

```
.non_secure.bss
                0x000000002012337c                 0x4
                0x0000000020123380                 . = ALIGN (0x10)
```

So user totally use 0xdef0 + 0x9b4 + 0x5d8 + 0x139fc + 0x4 = 0x2287c = 138KB memory in SRAM.
And the total SRAM space is defined at project\realtek_amebapro2_v0_example\GCC-RELEASE\application\rtl8735b_ram.ld:

```
RAM (rwx)    : ORIGIN = 0x20100B00, LENGTH = 0x20177B00 - 0x20100B00  /* 476KB */
```

For this case, the SRAM total size is 476KB. So user still have free SRAM space about 476KB - 138KB = 338KB.

17.2.2 Memory Size in DDR Memory (ERAM)

There are several sections in DDR Memory, ".ddr.bss", ".ddr.text", ".ddr.data" and ".ddr.rodata". We can sum up these sections:

".ddr.bss" has size 0x1424d8:

```
.ddr.bss        0x0000000070100000              0x1424d8
                0x0000000070100000              . = ALIGN (0x4)
                0x0000000070100000              __eram_bss_start__ = .
```

".ddr.text" has size 0x7d278:

```
.ddr.text       0x00000000702424e0              0x7d278
                0x00000000702424e0              . = ALIGN (0x4)
                0x00000000702424e0              __eram_text_start__ = .
```

".ddr.data" has size 0x4e08:

```
.ddr.data       0x00000000702bf79c              0x4e08
                0x00000000702bf79c              . = ALIGN (0x4)
                0x00000000702bf79c              __eram_data_start__ = .
```

".ddr.rodata" has size 0x512a:

```
.ddr.rodata     0x00000000702c45a4              0x512af
                0x00000000702c45a4              . = ALIGN (0x4)
                0x00000000702c45a4              __eram_rodata_start__ = .
```

we cannot calculate the heap usage now. However, we can get its value after running an application on AmebaPro2.
DDR memory total size = .ddr.bss + .ddr.text + .ddr.data + .ddr.rodata + heap usage + heap available

We can get the heap available size by entering AT command – "ATW?" in uart console:

```
[MEM] After do cmd, available heap 55318304
```

For this example, the heap available size is 54893248 = 53606 KB = 52 MB
And the total ERAM space is defined at project\realtek_amebapro2_v0_example\GCC-RELEASE\application\rtl8735b_ram.ld:

```
DDR (rwx)    : ORIGIN = 0x70100000, LENGTH = 0x73900000 - 0x70100000 /* 56MB */
```

For this case, the DDR memory total size is 0x73900000 - 0x70100000 = 0x3800000 = 56MB.
So we can calculate the heap usage now:
heap usage = DDR memory total size - .ddr.bss - .ddr.text - .ddr.data - .ddr.rodata - heap available = 0x3800000(56MB) - 0x1424d8 - 0x7d278 - 0x4e08 - 0x512af - 54893248 = 1,641,785 B = 1.6 MB

17.3 Code Size

The size of flash_ntz.bin:

```
$ ls -al GCC-RELEASE/build/flash_ntz.bin
-rw-r--r-- 1 user Domain Users 4599808 May 27 16:11 flash_ntz.bin
```

For this case, the code size is about 4.4MB

17.4 CPU Utilization

CPU utilization can be evaluated by AT command - "ATSS" in uart console.
It will show the amount of time each task has spent in the Running state (how much CPU time each task has consumed).

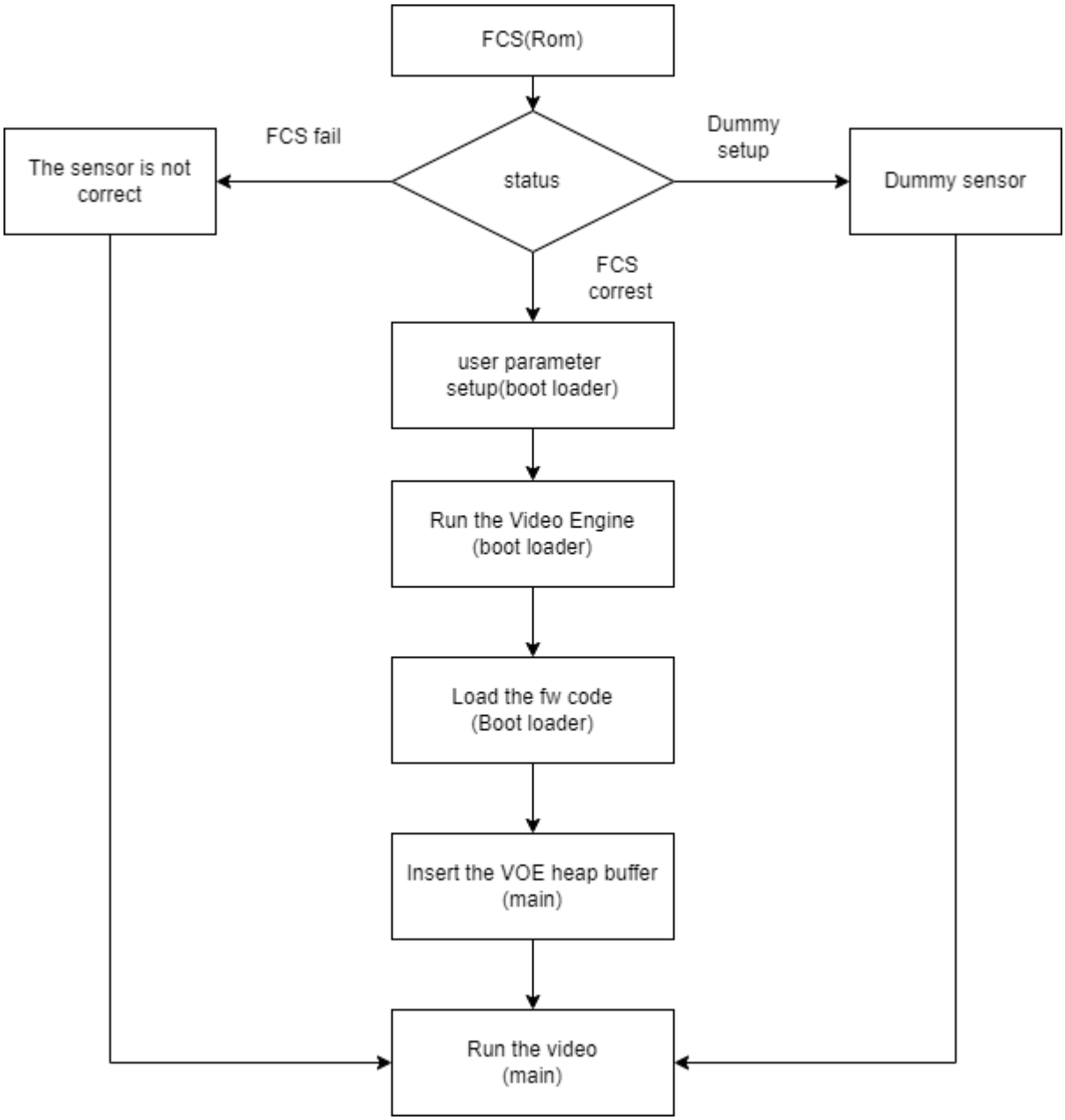
```
[ATSS]: _AT_SYSTEM_CPU_STATS_
log_service     161             <1%
IDLE            4983981         99%
Tmr Svc         0               <1%
TCP_IP          0               <1%
cmd_thread      1246            <1%
rtw_interru     0               <1%
rtw_recv_ta     0               <1%
rtw_xmit_ta     0               <1%
```

18 FCS and multi sensor

The fast camera start (FCS) can speed up the sensor bring up time. The sensor starts from rom code and initial the Video engine from bootloader. It can speed up the time to get the first frame. The multi sensor can support auto sensor recognition. The sensor firmware is located at the flash; it will try the sensor list to get the current sensor driver. The first time will take time to scan the sensor, and then the result will be stored at the flash.

18.1 FCS Procedure

According to the following flowchart, Rom code will check whether the FCS process needs to be executed, load user data after the condition is established, and execute the VOE startup process in the bootloader, if the condition is not established, then start in Normal mode. It must be confirmed that the sesnor power needs to be given at the beginning, otherwise the fcs will fail

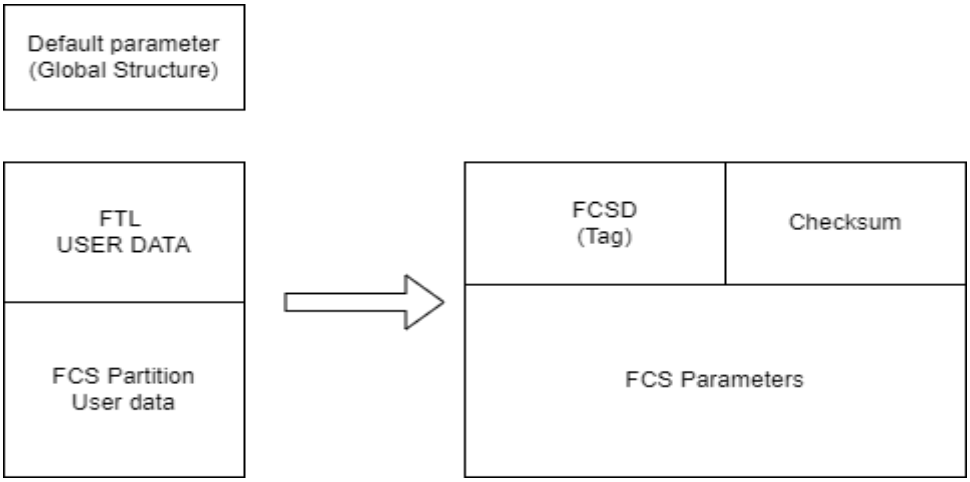


18.1.1 The FCS parameter

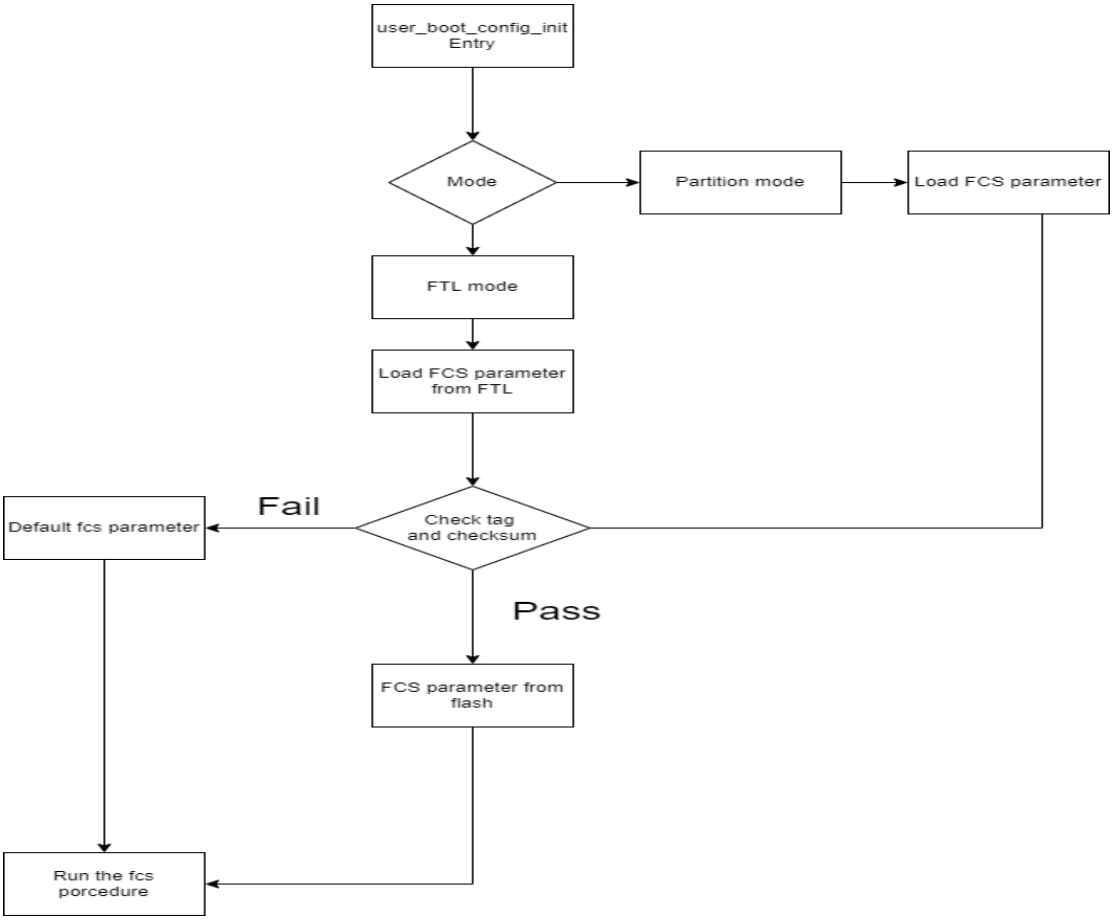
The main purpose of the FCS parameter is to set the FCS image parameters, which will include two types, one is the default parameter, which is currently defined as GLOBAL STRUCTURE, and the other is to save the data by storing it in FLASH.

FLASH storage parameters, the content will have a TAG plus Checksum to ensure that the data is correct, as shown in the figure below, you can choose one of FTL and FCS Partition, and if you use FTL, it must be the first PAGE of NAND BLOCK.

- FTL stored parameters: Update parameters through FTL API
- Parameters stored in FCS Partition: Update parameters through FWFS



In the FCS parameter process, enter the user_boot_config_init weak function at the beginning, determine whether it is Partition or FTL mode, load the FCS User Data to check whether the data is correct, load the data if it succeeds, and load the default global variables if it fails, please refer to the following flowchart



The following structure is the FCS parameter content.

```
typedef struct video_boot_stream_cfg {
 video_params_t video_params[4];//For different channel parameter
 isp_info_t isp_info;//isp_info
 uint32_t voe_heap_addr;//Heap address for VOE
 uint32_t voe_heap_size;//Heap size for VOE
 uint8_t video_enable[4];//Enable channel
 uint8_t video_snapshot[4];//Enable the snapshot, it only support channel 0 now
 isp_multi_fcs_ld_info_t p_fcs_ld_info;//The info to load iq and sensor firmware
 uint32_t fcs_channel;//channel -> //How many channel to use fcs, the default is 1
 uint32_t fcs_status;// status  -> 1:successful 0:fail
 uint32_t fcs_setting_done;// status  -> 1:successful 0:fail
 uint32_t fcs_voe_fw_addr;//
 uint32_t fcs_isp_ae_enable;//Enable the AE init function
 uint32_t fcs_isp_ae_init_exposure;//Setup the exposure parameters.
 uint32_t fcs_isp_ae_init_gain;//Setup the gain parameters.
 uint32_t fcs_isp_awb_enable;//Enable the AWB init function.
 uint32_t fcs_isp_awb_init_rgain;//Setup the AWB rgain parameters.
 uint32_t fcs_isp_awb_init_bgain;//Setup the AWB bgain parameters.
    uint32_t fcs_isp_init_daynight_mode;//0 day mode ; 1 night mode
 uint32_t fcs_lookup_count;//Lookup table count for AE&AWB
 uint32_t fcs_als_thr[11];//Threashold
 uint32_t fcs_isp_ae_table_exposure[11];//AE Exposure time
 uint32_t fcs_isp_ae_table_gain[11];//AE gain
 uint32_t fcs_isp_awb_table_rgain[11];//AWB r-gain
 uint32_t fcs_isp_awb_table_bgain[11];//AWB b-gain
 uint32_t fcs_isp_mode_table[11];//day night mode table
 uint32_t fcs_isp_iq_id;//0:default parameter, other number : IQ table index
 uint8_t  fcs_isp_reserved_buf[FCS_SYSTEM_REV_SIZE];//Reserved for fcs system data
 uint8_t  fcs_user_buffer[FCS_USER_REV_SIZE];//User can use the buffer to transfer to
application
 uint32_t fcs_start_time;//bootloader to fcs user boot function
 uint32_t fcs_voe_time;//bootloader to voe init function
 video_boot_private_mask_t private_mask;
 uint32_t meta_enable;//
 uint32_t meta_size;//enalbe the meta size for
 video_boot_isp_initial_items_t init_isp_items;
} video_boot_stream_t;
typedef struct  {
 uint32_t enable;
 uint32_t init_flicker;
 uint32_t init_saturation;
 int32_t init_brightness;
 uint32_t init_contrast;
 uint32_t init_hue;
 uint32_t init_wdr_mode;
 uint32_t init_wdr_level;
 uint32_t init_hdr_mode;
 uint32_t init_mirrorflip;
} video_boot_isp_initial_items_t;
typedef struct video_param_s {
 uint32_t stream_id; //Channel ID
 uint32_t type; //Codec type
 uint32_t resolution;//Resolution
 uint32_t width;
 uint32_t height;
 uint32_t bps;
 uint32_t fps;
 uint32_t gop;
 uint32_t rc_mode;
 uint32_t jpeg_qlevel;
 uint32_t rotation;//
 uint32_t out_buf_size;//Reserve buf to encode queue
 uint32_t out_rsvd_size;//Don't care
 uint32_t direct_output;//Don’t care
 uint32_t use_static_addr;//Don’t care
 uint32_t fcs;//Enable the FCS mode
```

```
} video_params_t;
```

For detailed parameter settings, please refer to the following file.

```
component\video\driver\RTL8735B\video_user_boot.c
```

Please be careful not to add or remove structure parameters, this may cause internal data errors.

18.1.2 Configure default parameter for FCS

User can configure FCS default parameter, which is a global struct variable, in video_user_boot.c before running the FCS example. For instance, if user want to enable V1 streaming with FCS, the "video_params[STREAM_V1].fcs" should be set to 1. Then, V1 FCS mode will be enabled with the specified parameters such as width, height, bps, fps and Qp. In addition, if a jpeg snapshot is required in FCS, the "video_snapshot[STREAM_V1]" should also be set to 1. After that, user can get the jpeg snapshot from the registered callback function.

```
video_boot_stream_t video_boot_stream = {
 .video_params[STREAM_V1].stream_id = STREAM_V1,
 .video_params[STREAM_V1].type = CODEC_H264,
 .video_params[STREAM_V1].resolution = 0,
 .video_params[STREAM_V1].width = 1920,
 .video_params[STREAM_V1].height = 1080,
 .video_params[STREAM_V1].bps = 2 * 1024 * 1024,
 .video_params[STREAM_V1].fps = 15,
 .video_params[STREAM_V1].gop = 15,
 .video_params[STREAM_V1].rc_mode = 2,
 .video_params[STREAM_V1].minQp = 25,
 .video_params[STREAM_V1].maxQp = 48,
 .video_params[STREAM_V1].jpeg_qlevel = 0,
 .video_params[STREAM_V1].rotation = 0,
 .video_params[STREAM_V1].out_buf_size = V1_ENC_BUF_SIZE,
 .video_params[STREAM_V1].out_rsvd_size = 0,
 .video_params[STREAM_V1].direct_output = 0,
 .video_params[STREAM_V1].use_static_addr = 0,
 .video_snapshot[STREAM_V1] = 1,
 .video_drop_frame[STREAM_V1] = 0,
 .video_params[STREAM_V1].fcs = 1,//Enable the fcs for channel 1
 …
 …
}
```

18.1.2.1 ISP initial parameter setting in FCS

There are some ISP parameters can be set before video opening in FCS:

```
typedef struct  {
 uint32_t enable;
 uint32_t init_flicker;   //DISABLE = 0, 50HZ = 1, 60HZ = 2, AUTO = 3
 uint32_t init_saturation;
 int32_t init_brightness;
 uint32_t init_contrast;
 uint32_t init_hue;
 uint32_t init_wdr_mode;   // 0: DISABLE, 1: MANUAL, 2: AUTO
 uint32_t init_wdr_level;
 uint32_t init_hdr_mode;
 uint32_t init_mirrorflip;   // bit 0: MIRROR, bit 1: Flip
} video_boot_isp_initial_items_t;
```

NOTE

For the mirror/flip feature, there is a control timing difference for each sensor. Even though mirror/flip enabled before streaming open, for some sensor model, the mirror/flip will be effective on 2nd or 3rd frame. Therefore, if user want to get the FCS snapshot with mirror/flip, "video_drop_frame" may need to be set to 1 or 2 to achieve this.

18.1.3 Secondary FCS parameter

Default FCS parameters set is loaded in bootloader. It is able load secondary FCS parameter from user defined data section, and this secondary FCS parameter could be updated through OTA.

Modify the component\video\driver\RTL8735B\video_user_boot.c

```
#define NAND_FLASH_FCS 0x7080000
#define NOR_FLASH_FCS  (0xF00000 + 0xD000)
```

```
int boot_read_flash_data(unsigned int address, unsigned char *buf, int length);
```

18.1.4 FCS example

Modify the project\realtek_amebapro2_v0_example\GCC-RELEASE\mp\amebapro2_isp_iq.json

```
"VARIABLE":{
        "*type*":"0x00 (INVALID) is reserved for invalid type, 0x01 (SENSOR_ID) is
reserved for senesor id. default offset is 2048 from manifest start",
        "tlv":[
            {"type":"SENSOR_ID", "length":1, "value":"3"} //value : sensor id
        ]
    },
```

It need to change the value to your sensor id.

Modify the file from component\video\driver\RTL8735B\video_user_boot.c

```
void user_boot_config_init(void *parm)
{
 //Insert your code into here
 //dbg_printf("user_boot_config_init\r\n");
}
```

If you need to execute the operation at boot loader step, you can add your code at the API.

Modify the file from sdk\project\realtek_amebapro2_v0_example\inc\sensor.h

```
#define USE_SENSOR          SENSOR_GC4653
```

Modify the media_framework.c and choose the below example. The default parameter is channel 0.

```
mmf2_video_example_joint_test_rtsp_mp4_init_fcs();
```

Currently, the default update FCS parameter is FTL mode. If you want to use FCS Partition mode, please open MARCO below. There are two places that need to be updated, user_boot_confic.c and mmf2_video_example_joint_test_rtsp_mp4_init_fcs

```
//#define FCS_PARTITION
```

Build your code and upgrade your FW.

Use ATCMD to switch the FCS parameter example, currently you can switch Resolution, if you need to switch other parameters, you can modify it according to this example. If you need to modify the IQ parameters, you need to maintain two different parameters before you can switch, 0 is the default value.

```
FCST=ch,width,height,iq_id
```

The example support to streaming, recording and snapshot. If you need to snapshot for the first frame, it need to enable the video_snapshot parameter for your video channel. There will have the callback function to get the picture.

About the fast forward problem, it need to initialize the video first to reduce the cached images. If it have the cached images that it need to calculate the length to skip the timestamp, it can get the information form the priv_data attribute of the MMF module.

18.1.5 Measure the time to first frame?

Before measure the time to first frame, user must disable the log of rom code, and the method to turn off the log of rom code is as follows.

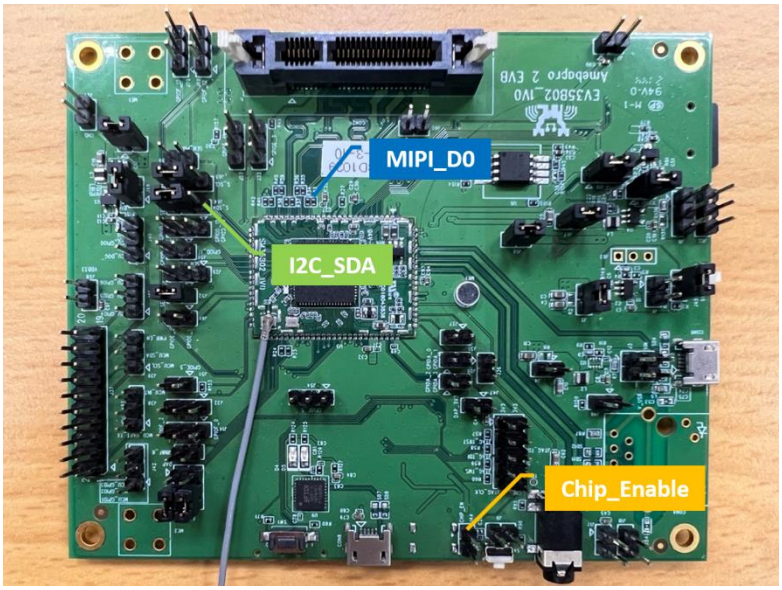
The first step is replaced libwlan.a to libwlan_mp.a (please delete libwlan.a and modify the file name of libwan_mp.a to libwlan.a). The file path of libwlan.a and libwlan_mp.a is as bellow.

```
project\realtek_amebapro2_v0_example\GCC-RELEASE\application\output\libwlan.a
project\realtek_amebapro2_v0_example\GCC-RELEASE\application\output\libwlan_mp.a
```

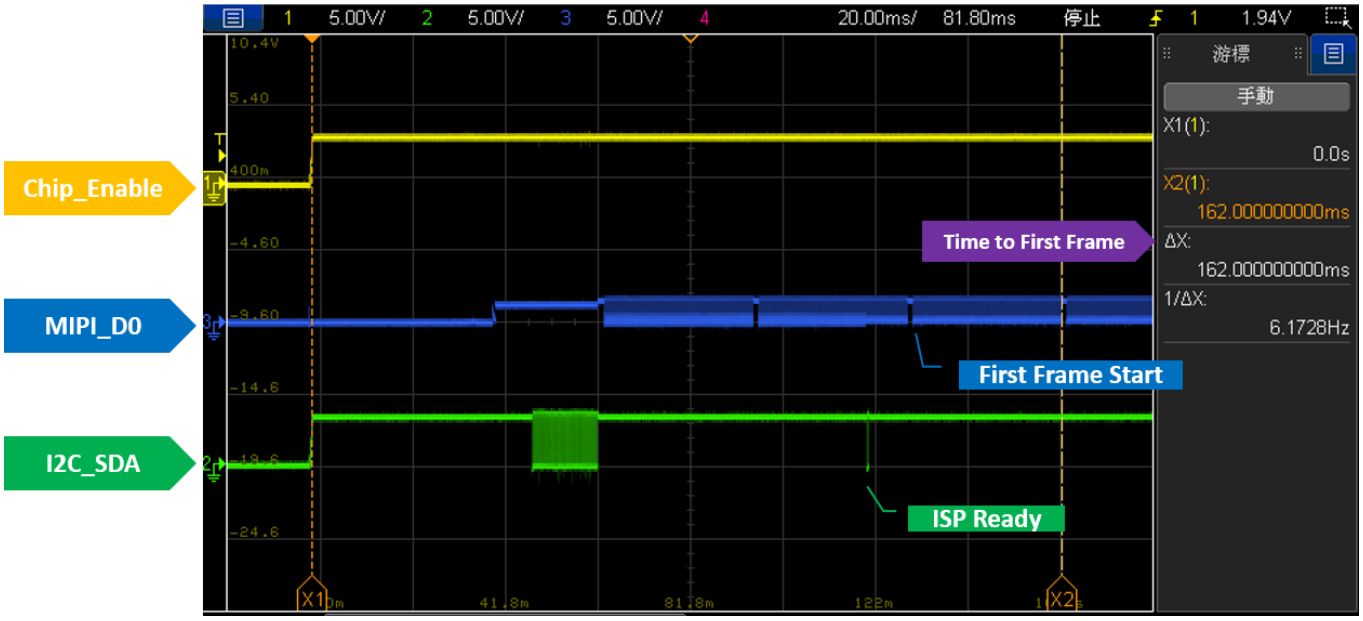
The second step is enable CONFIG_MP_INCLUDE in autoconf.h, and the file path of autoconf.h is as bellow.

```
component\wifi\driver\include\autoconf.h
```

Then, build the image and download. The third step, start the system and key the command, "iwpriv efuse_set wmap,74,0008". At last, please restart the system, and the log of rom will be disable. After turn off the log of rom code, please use the scope to get the signal of Chip_Enable, MIPI_D0, and I2C_SDA, and the measurement points of these are as follows.



The first frame after the ISP ready is the first output frame of amebapro2, so user can measure the time between the Chip_Enable and the first frame to get the time to first frame. The result of measurement is as bellow.



Here it can measure the time from the bootloader to the first frame done by software. You can use the following API to get the time. Note that this does not include the time from power-on to bootloader.

```
int video_get_fcs_cost_time(void);//The time unit is ms
```

18.1.6 How to get good image quality under FCS mode

Ameba Pro2 has designed quick convergence mode for AE and AWB with information supported by ambient light sensor (ALS). When use the reference value from ALS, Amebapro2 can set suitable initial value for exposure (with look up value with white balance gain). Even with the reference to ALS, there are two more reason that we need isp to do auto exposure and white balance. Fist item is the detection range is not same due to the FOV difference between ALS and sensor. Second item is ALS can only the luminance value but can't get color temperature information. So when we set initial value for isp , we also need to use quick auto exposure and auto white balance convergence method to achieve target luminance and color temperature.

At bootloader stage, amebapro2 can communicate through UART with MCU to get ALS value without convergence time, and then get initial value through look up table.

Luminance (lux)	ALS value (example)	Mode	Scene	Exposure Time (us)	Gain	Color temperature	R Gain	B Gain
> 60000	> 150000	RGB	Outdoor scene with strong sun light	45	256	7310	662	446
> 40000	> 100000			112	256	7310	662	446
> 13000	> 62905			289	274	7310	662	446
> 5000	> 55000			1000	256	7310	662	446
> 2000	> 49661		With strong light	2000	256	6400	585	478
> 1000	> 30000			3112	256	6400	585	478
> 600	> 15298			3534	256	6400	585	478
> 350	> 7524		With suitable light	4734	256	4980	537	542
> 200	> 3822			10000	328	4980	537	542
> 100	> 2529			10000	512	4980	537	542
> 70	> 1222		With low light	20000	512	4020	512	600
> 50	> 800			20000	768	4020	512	600
> 20	> 581		With shimmer	30000	768	3400	377	685
> 10	> 260			30000	1024	3400	377	685
< 5	< 100	IR	Extreme low lux and switch to IR Mode	30000	2048	-		

18.1.7 FCS parameter setting

The FCS data structure is defined in component\video\driver\RTL8735B\video_boot.h, and cannot be changed or re-ordered.

The definition of video_boot_stream_cfg

Name	Mim	Max	Modify	Description
video_params			Yes	See Bellowing
auto_rate_control			Yes	See Bellowing
isp_info			Yes	See Brllowing
voe_heap_addr			No	Video heap address
voe_heap_size			No	Video heap size
video_enable			Yes	Enable the video channel.(0:Disable 1:Enable)
video_snapshot			Yes	Support snapshot with streaming.(0:Disable 1:Enable)
video_drop_frame			Yes	Drop video frame count.
p_fcs_ld_info			No	Fcs driver index alignment with index defined at 「amebapro2_isp_iq.json」
fcs_channel			Yes	Enable the video to fcs channel.(0:Disable1:Enable)
fcs_status			No	FCS status(0:Disable 1:Enable)
fcs_setting_done			No	FCS setting status.(0:Setting done 1:Setup procedure)
fcs_voe_fw_addr			No	VOE firmware address
fcs_isp_ae_enable			Yes	Enable Auto exposure initial value configuration
fcs_isp_ae_init_exposure			Yes	Initial value for exposure time (active when fcs_isp_ae_enable=1)
fcs_isp_ae_init_gain			Yes	Initial value for exposure gain (active when fcs_isp_ae_enable=1)
fcs_isp_awb_enable			Yes	Enable Auto white balance initial value configuration
fcs_isp_awb_init_rgain			Yes	Initial value for white balance R-gain (active when fcs_isp_awb_enable=1)
fcs_isp_awb_init_bgain			Yes	Initial value for white balance B-gain (active when fcs_isp_awb_enable=1)
fcs_isp_init_daynight_mode			Yes	Initial value for IQ table index (0=RGB, 1=IR, 2=other)
fcs_isp_gray_mode			Yes	Initial value for color mode (0=RGB, 1=Gray)
fcs_lookup_count			Yes	Count for look up table, can refer to 「USE_FCS_LOOKUPTABLE_SAMPLE」
fcs_als_thr			Yes	Threshold for ALS (lookup table)
fcs_isp_ae_table_exposure			Yes	Initial value for exposure time (lookup table)
fcs_isp_ae_table_gain			Yes	Initial value for exposure gain (lookup table)
fcs_isp_awb_table_rgain			Yes	Initial value for R-Gain (lookup table)
fcs_isp_awb_table_bgain			Yes	Initial value for B-Gain (lookup table)
fcs_isp_mode_table			Yes	Initial mode (lookup table)
fcs_isp_iq_id			Yes	IQ index (defined by user, could be non-sync with p_fcs_ld_infor)
fcs_isp_reserved_buf			No	System reserved buffer.
fcs_user_buffer			Yes	User buffer.
fcs_start_time			No	Boot loader to video initial time
fcs_voe_time			No	Boot loader to video start time
private_mask				See Bellowing
meta_enable			Yes	Enable the meta data feature (0:Disable 1: Enable)
meta_size			Yes	User meta size
nit_isp_items				See bellowing

Sub-structure: The definition of video_boot_stream_cfg.video_params

Name	Mim	Max	Modify	Description
stream_id	0	3	Yes	Video channel number.
type			Yes	Video type (CODEC_H264 or CODEC_HEVC)
width	0		Yes	Video width
Height	0		Yes	Vidoe height

bps			Yes	Target bit rate
fps			Yes	Frame rate
gop			Yes	Group of Picture
out_buf_size			No	Video output buffer size
out_rsvd_size			No	Video output reserved size.
jpeg_level	0	9	Yes	Jpeg level
Direct_output			No	Not used
use_static_addr			No	Not used
fcs	0	1	Yes	0:Enable the fcs mode 1:Disable the fcs mode.
Level			Yes	Encoder level.
Cavlc	0	1	Yes	1:cavlc,0:cabac
MinQP	0	51	Yes	Minimum QP value
MaxQP	0	51	Yes	Maximum QP value

Sub-structure: The definition of video_boot_stream_cfg.auto_rate_control

Name	Mim	Max	Modify	Description
sampling_time			Yes	Sample interval time
maximum_bitrate			Yes	Maximum bitrate
minimum_bitrate			Yes	Minimun bitrate
target_bitrate			Yes	Target bitrate

Sub-structure: The definition of video_boot_stream_cfg.isp_info

Name	Mim	Max	Modify	Description
sensor_width	0	Based on sensor spec	Yes	Sensor width
sensor_height	0	Based on sensor spec	Yes	Sensor height
sensor_fps	5	30	Yes	Sensor fps
osd_enable	0	1	No	Enable bit to control voe heap contains osd usage or not
md_enable	0	1	No	Enable bit to control voe heap contains md usage or not
hdr_enable	0	1	No	Enable bit to control voe heap contains hdr usage or not
osd_buf_size	0		No	Buffer size for osd (no use in fcs)
md_buf_size	0		No	Buffer size for md (no use in fcs)

Sub-structure: The definition of video_boot_stream_cfg.private_mask

Name	Mim	Max	Modify	Description
enable	0	1	Yes	Enable Privacy Mask initial configuration
color	0		Yes	Color for privacy mask, format: 0xRRGGBB (RR/GG/BB = 0~255)
en	0	4	Yes	Enable control for rectangle (1~4) and grid (0)
start_x	0	Image width	Yes	Start position of x axis (should be 2 alignment)
start_y	0	Image height	Yes	Start position of y axis (should be 2 alignment)
w	0	Can't be over sensor image width / height	Yes	Width for ROI window (Rect) or width for each grid
h	0		Yes	Width for ROI window (Rect) or height for each grid
cols	0		Yes	Number of grid columns (should be 2 alignment)
rows	0		Yes	Number of grid rows (should be 2 alignment)
bitmap			Yes	Enable bit for each grid (start from left top grid)

Sub-structure: The definition of video_boot_stream_cfg. init_isp_items

Name	Mim	Max	Default	Modify	Description

enable	0	1	0	Yes	Enable isp initial configuration
init_flicker	0	3	1	Yes	Refer to isp_set_power_line_freq()
init_saturation	0	100	50	Yes	Refer to isp_set_saturation()
init_brightness	-64	64	0	Yes	Refer to isp_set_brightness()
init_contrast	0	100	50	Yes	Refer to isp_set_contrast()
init_wdr_mode	0	2	0	Yes	Refer to isp_set_wdr_mode()
init_wdr_level	0	100	50	Yes	Refer to isp_set_wdr_level()
init_hdr_mode	0	1	0	Yes	Enable HDR mode (should support by sensor)
init_mirrorflip	0xf0	0xf3	0xf0	Yes	Modify sensor output orientation

18.1.8 Disable the boot loader log

At present, the default rom and bootloader will print the message, you can close the log message by modifying efuse, and speed up the time to enter the main, please reference the follow command.

```
iwpriv efuse_set wmap,74,0008 //disable bootloader log
iwpriv efuse_set wmap,74,0000 //enable bootloader log
iwpriv efuse get rmap,74,2    //check the return value
ATWZ=dbg 0x7fdd0000 5          //check efuse available raw size
```

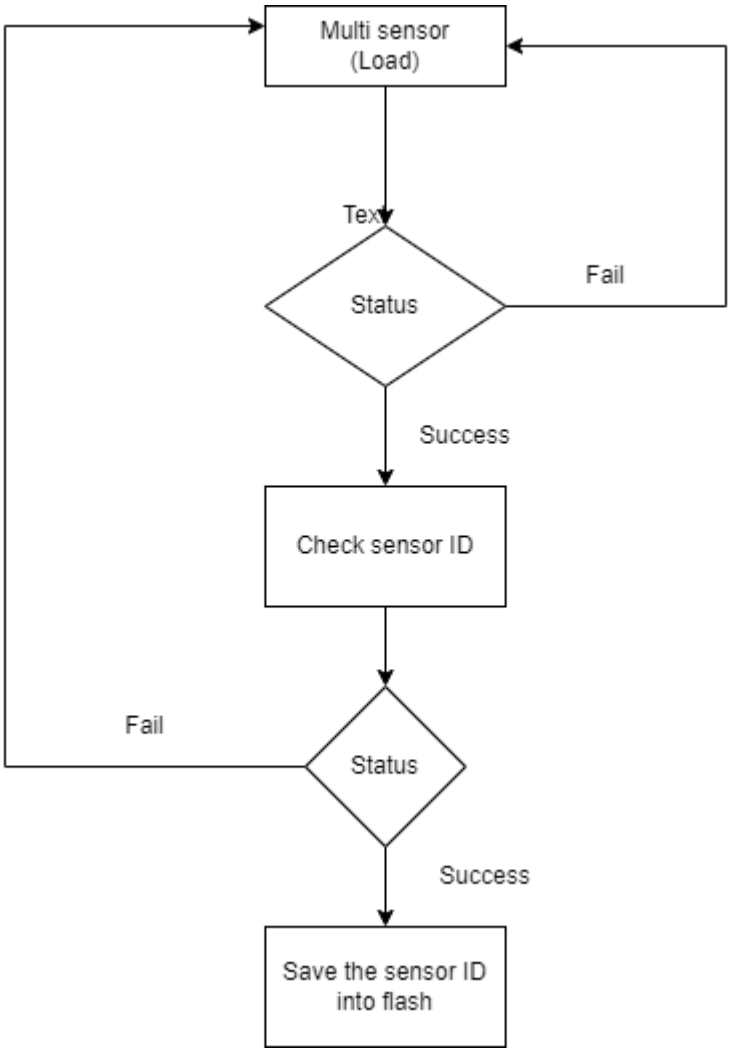
Note that printf cannot be used to print messages here, please change to dbg_printf

18.1.9 Disable the fcs at weak function

If you have started fcs, but want to close it under certain circumstances, you can close it through the following weak function

```
//video_user_boot.c
int user disable fcs(void)
{
 return 0;//1:disable fcs, 0:Don't care
}
```

18.2 Multi sensor



18.2.1 Configure multi sensor

We use the GC2053 for the example, it need to assign the sensor, FCS and IQ data.

Modify the sdk\project\realtek_amebapro2_v0_example\GCC-RELEASE\mp\amebapro2_isp_iq.json

```
"VARIABLE":{
        "*type*":"0x00 (INVALID) is reserved for invalid type, 0x01 (SENSOR_ID) is
reserved for senesor id. default offset is 2048 from manifest start",
        "tlv":[
            {"type":"SENSOR_ID", "length":1, "value":"0"}
        ]
    },
```

The value need to setup as zero. The non-zero value is for FCS mode.

Modify the sdk\project\realtek_amebapro2_v0_example\GCC-RELEASE\mp\ amebapro2_sensor_set.json

We support five sensors. The default 0 is not used.

```
 "ISP_SENSOR_SETS":{
        "multi_fcs_hdr": "MULTI_FCS_HDR",
        "multi_fcs_info": "MULTI_FCS_INFO",
        "sensor_sets":[
              "SENSOR_SET0",//Dummy setup
              "SENSOR_SET1",//GC2336
              "SENSOR_SET2",//GC2053
              "SENSOR_SET3",//GC4653
              "SENSOR_SET4",//MIS2008
              "SENSOR_SET5"//PS5258
        ]
 },
```

Add your sensor into to the below structure. The ID is two for gc2053. The maximum support sensor size is nine.

```
"SENSOR_SET2":{
        "merge_en": true,
        "fcs_data": {
            "source":"binary",
            "file":"fcs_data_gc2053.bin"
        },
        "iq_data": {
            "source":"binary",
            "file":"iq.bin"
        },
        "sensor_data": {
            "source":"binary",
            "file":"sensor_gc2053.bin"
        }
    },
```

Modify the file from sdk\project\realtek_amebapro2_v0_example\inc\sensor.h

```
#define SENSOR_DUMMY        0x00 //For dummy sensor, no support fast camera start
#define SENSOR_SC2336       0x01
#define SENSOR_GC2053       0x02
#define SENSOR_GC4653        0x03
#define SENSOR_MIS2008      0x04
#define SENSOR_PS5258       0x05 //It don't support the multi sensor for PS5258 now.If you
want to use the sensor,please remove it.

#define MULTI_DISABLE       0x00
#define MULTI_ENABLE        0x01

#define MULTI_SENSOR        MULTI_ENABLE
#define USE_SENSOR          SENSOR_GC2053
```

19 USB

Ameba Pro2 provides USB stack features as below:

- Compatible with USB 2.0 specification
- Device-specified API for class and application development
- UVC, CDC and DFU support

19.1 UVC

19.1.1 UVC device example

Ameba Pro2 ISP can support compressed (H264 / H265 / JPG) and uncompressed (NV16 / NV12) image through UVC (wired transmission), and user can check video on pc with Potplayer, Amcap or RTK-realcam. For uncompressed format, user need to install RTK decoder to get video on computer. User can use following flow to build UVC example. Generate the makefile for the UVC project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DCUTVER=B -
DEXAMPLE=media_uvcd
```

Then, use the following command to generate an image:

```
cmake --build . --target flash
```

19.2 USB CDC

19.2.1 CDC device example

Ameba Pro2 supports USB cdc device mode that can be used to emulate a serial port providing a virtual COM port UART interface. A CDC loopback example to provided, user can use following flow to build CDC example. Generate the makefile for the CDC project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DCUTVER=B -
DEXAMPLE=usb_cdc
```

Then, use the following command to generate an image:

```
cmake --build . --target flash
```

19.2.2 CDC Class API

19.2.2.1 API for Application

API	Description
usbd_cdc_acm_init	Initialize the class with parameters: ● RX buffer length (rx_buf_len): BULK OUT buffer length ● TX buffer length (tx_buf_len): BULK IN buffer length ● Application callback (cb)
usbd_cdc_acm_deinit	De-initialize the class
usbd_cdc_acm_transmit	Transmit BULK IN data to host, the data length shall not be larger than the TX buffer length
usbd_cdc_acm_receive	Prepare to receive BULK OUT data, the data length will be limited to RX buffer length

19.2.2.2 Application Callback

CDC ACM class provides callbacks for user application, the callbacks are defined:

```
typedef struct {
    u8(* init)(void);
    u8(* deinit)(void);
    u8(* setup)(u8 cmd, u8 *buf, u16 len, u16 value);
    u8(* receive)(u8 *buf, u32 len);
} usbd_cdc_acm_cb_t;
```

Description of the callbacks:

API	Description
init	Called at the end of class initialization flow, for application-specific initialization
deinit	Called at the beginning of class de-initialization flow, for application-specific de-initialization
setup	Called at setup phase or data out phase of class-specific control requests, for application-specific setup

receive	Called at data out phase of BULK OUT transfer, for application to handle the received data

19.3 USB DFU

AmebaPro2 supports DFU (Device Firmware Upgrade), with DFU user can download firmware to AmebaPro2 over USB, user can use following flow to build DFU example. Generate the makefile for the DFU project:

```
cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DCUTVER=B -
DEXAMPLE=usb_dfu_ota
```

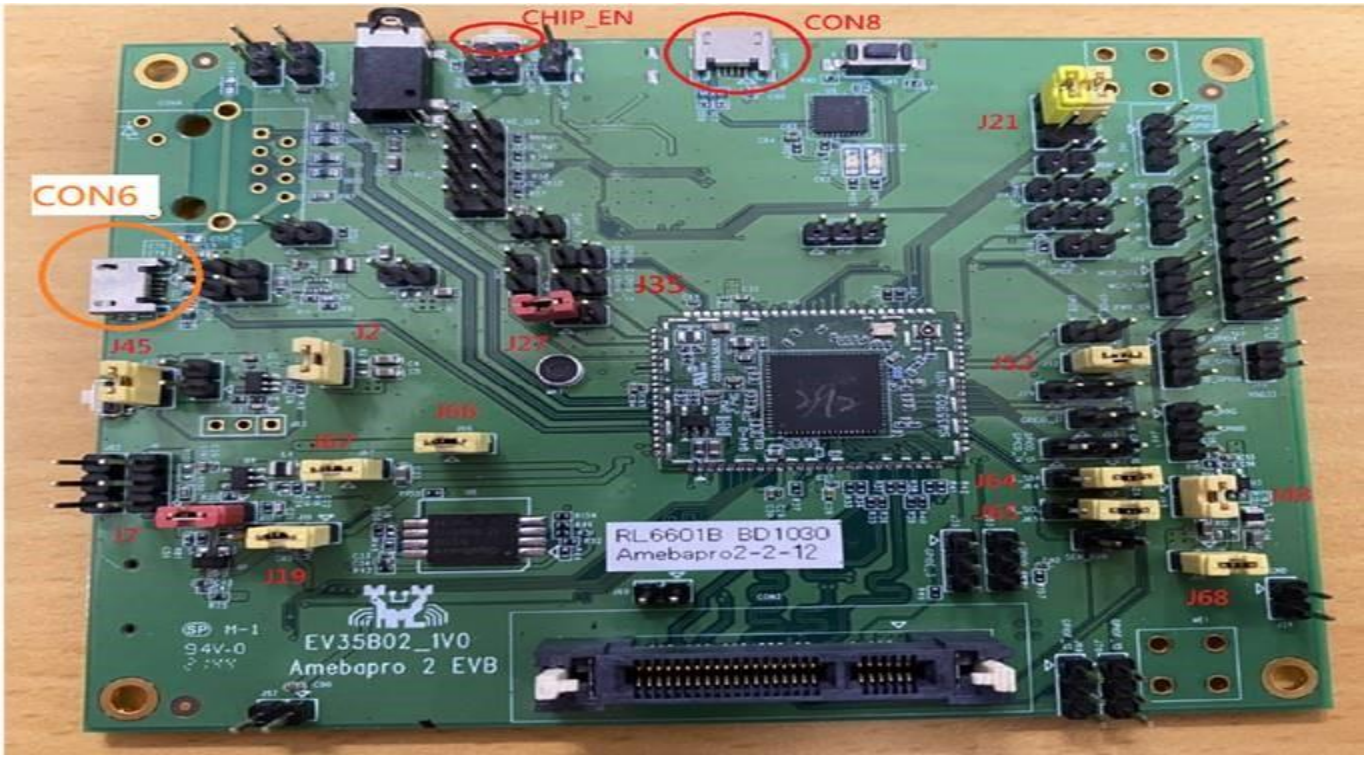
Then, use the following command to generate an image:

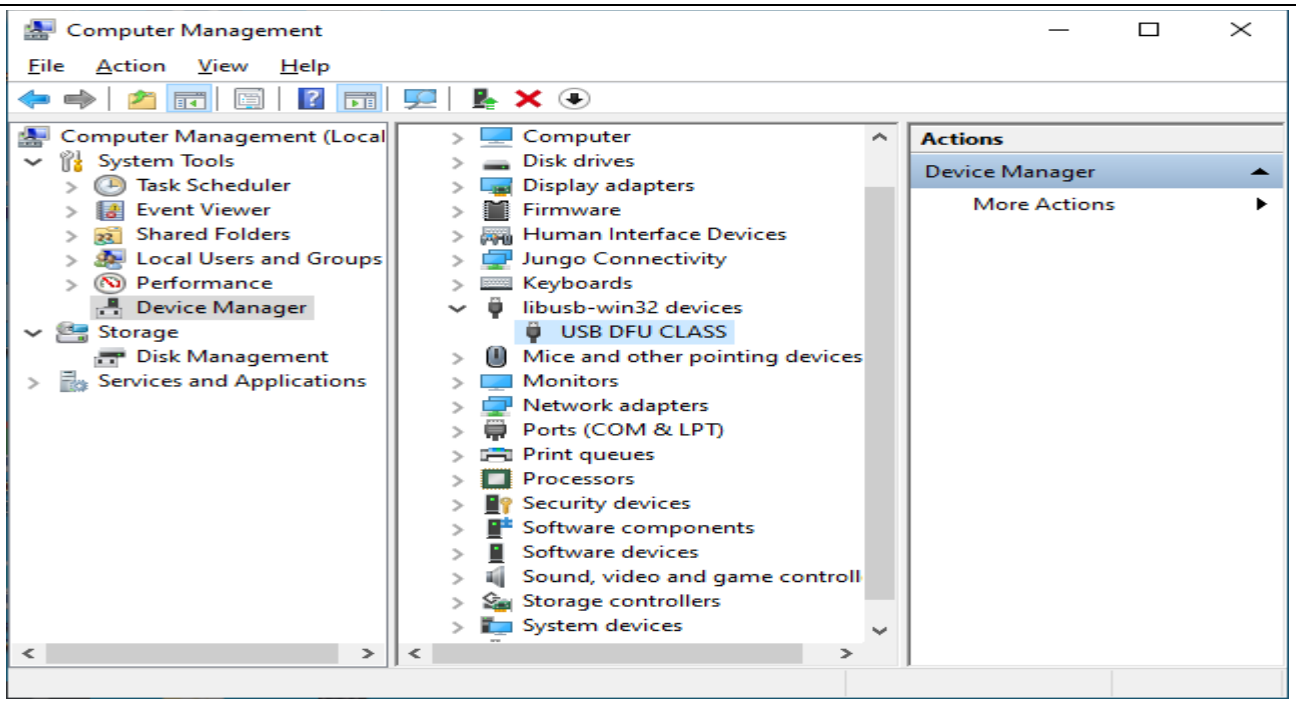
```
cmake --build . --target flash
```

19.3.1 How to download firmware over USB DFU

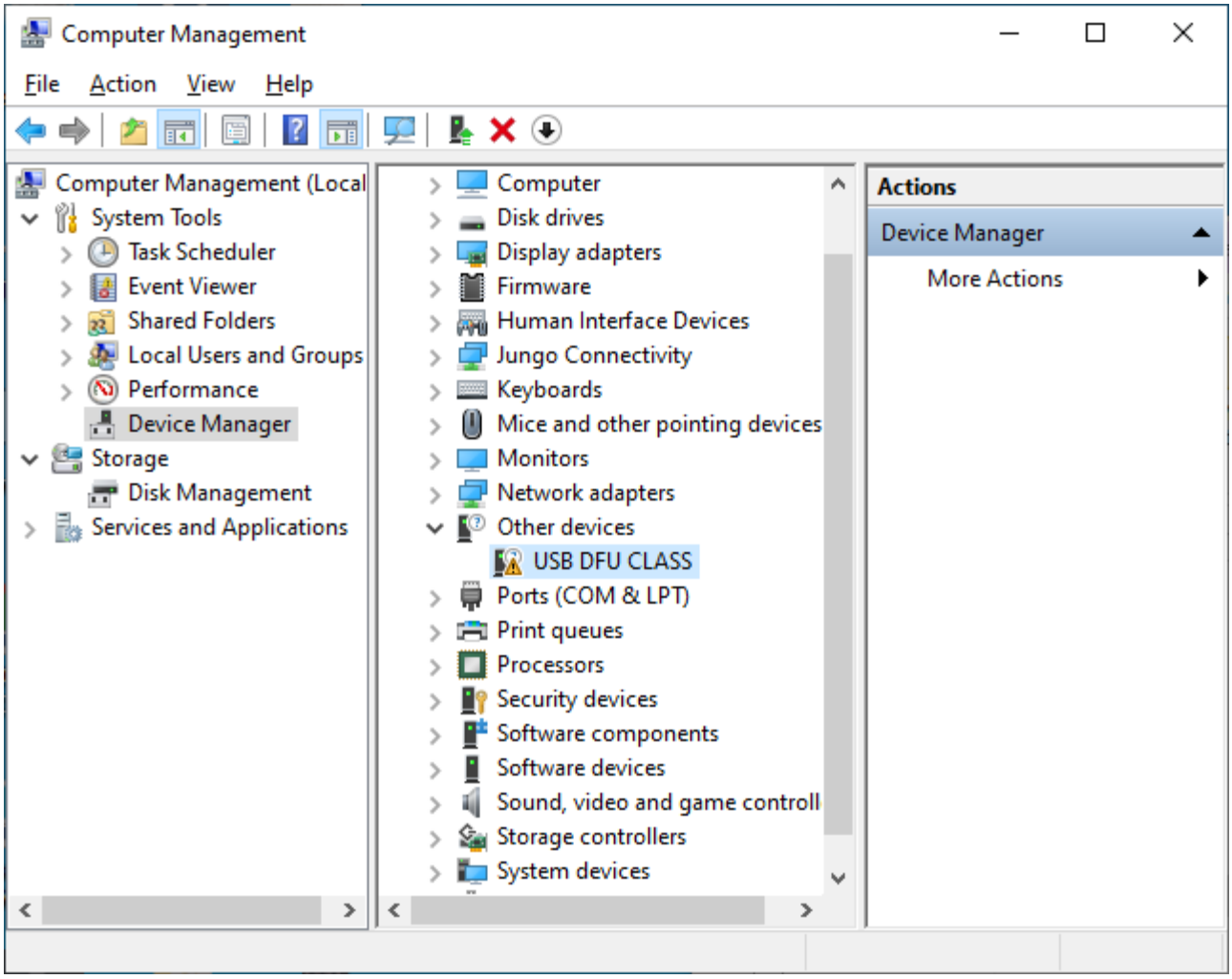
19.3.1.1 Install driver for USB DFU

After the DFU example code is started, and you have USB (CON6) connected to your windows computer, you can see DFU device in Device Manager as below:





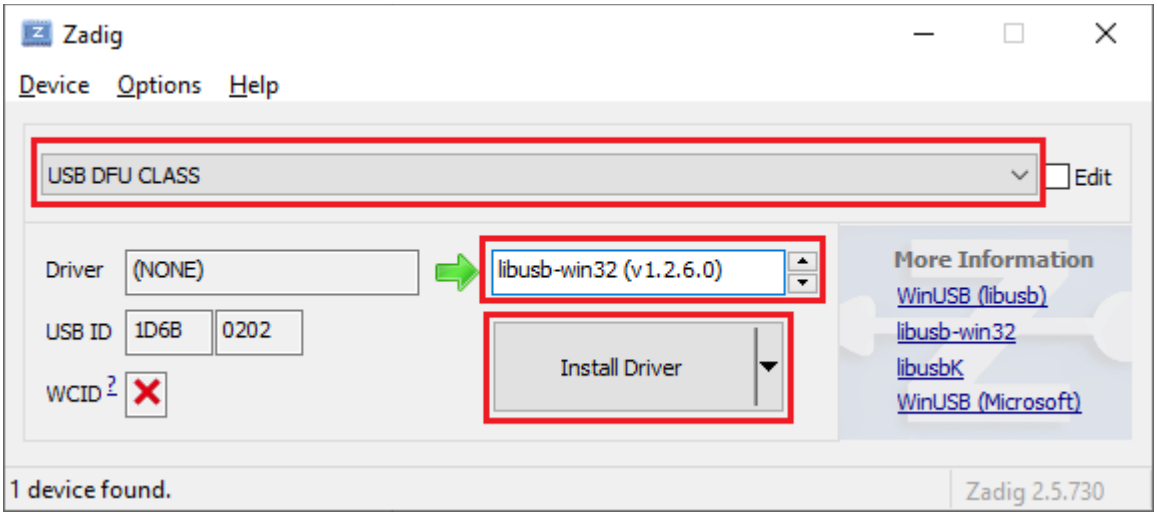
If your DFU device is unable to work (driver is not installed), you will see:



In this case, please follow these steps to install the driver:

(1) Open "Zadig.exe"
(2) Select "USB DFU CLASS"

(3) Select "libusb-win32"
(4) Click "Install Driver"

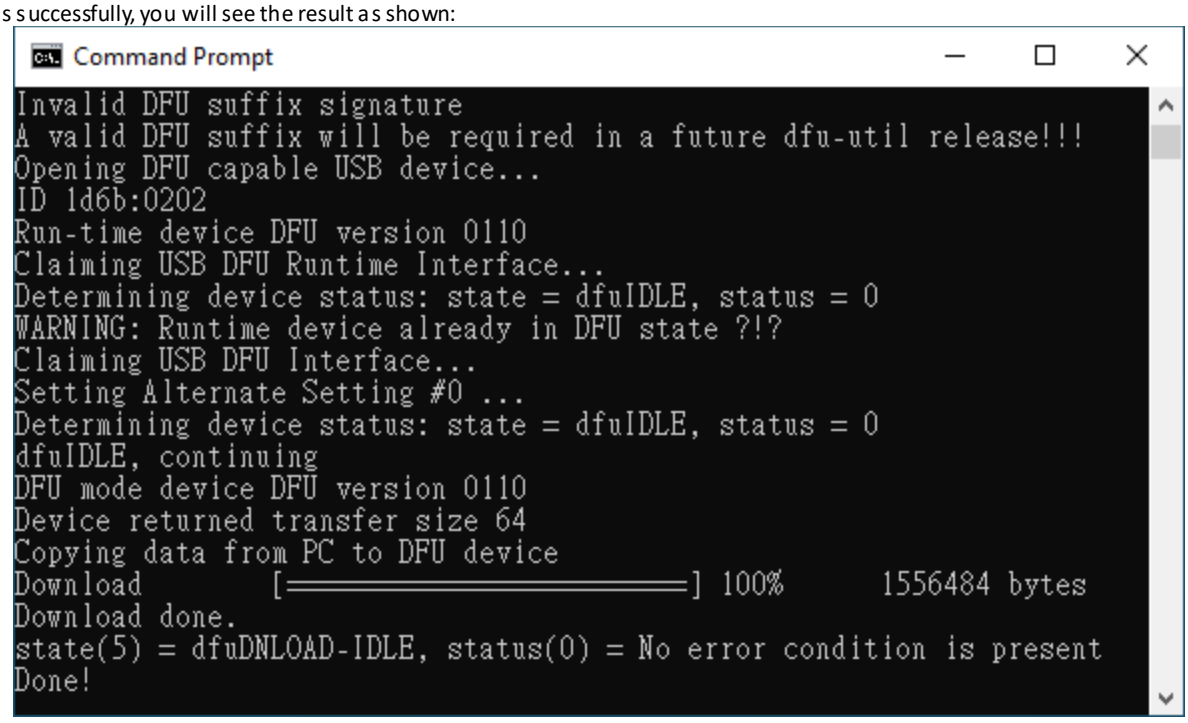


19.3.1.2 Burn DFU firmware to AmebaPro2

(1) Copy "ota.bin" from image build folder
(2) Run dfu-util tool:

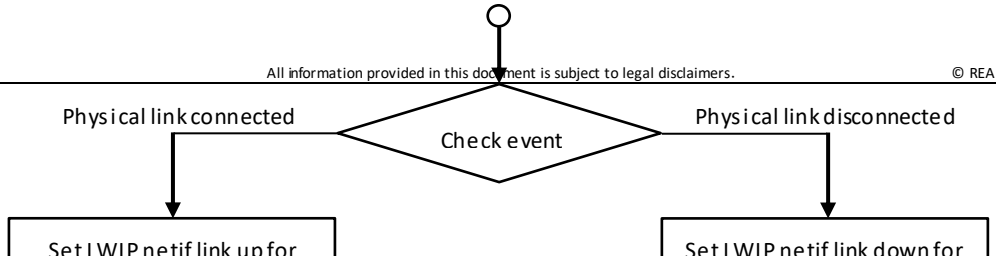
```
dfu-util -d 1d6b:0202 -a 0 -D ota.bin
```

(1) If DFU is successfully, you will see the result as shown:



19.4 USB PoE and Wi-Fi Network Connection Detection

USB PoE or Ethernet should be able to handle the events for physical link connected and disconnected. For the events about physical link, Ethernet may be related to Ethernet cable plugin/unplug, and USB PoE may be related to USB device attach/detach. The following is the flow to switch network between Wi-Fi and USB PoE/Ethernet. The event callbacks set LWIP netif link status and handle IP address for USB Poe/Ethernet interfaces. Finally, the LWIP default gateway is also set to specify the interface used to transmit the packets to external network.



20 Wi-Fi

20.1 Wi-Fi Data Structures

Data Structures	Introduction
<_cus_ie>	The structure is used to set Wi-Fi custom IE list, and type match CUSTOM_IE_TYPE. The IE will be transmitted according to the type.
<rtw_ssid_t>	The structure is used to describe the SSID.
<rtw_mac_t>	The structure is used to describe the unique 6-byte MAC address.
<rtw_softap_info_t>	The structure is used to describe the setting about SSID, security type, password and default channel, used to start AP mode.
<rtw_network_info_t>	The structure is used to describe the station mode setting about SSID, security type and password, etc., used when connecting to an AP.
<rtw_scan_param_t>	The structure is used to describe the scan parameters used for scan, including SSID, channel, user callback, etc.
<rtw_scan_result_t>	The structure is used to describe the scan result of the AP.
<rtw_wifi_setting_t>	The structure is used to store the Wi-Fi setting gotten from Wi-Fi driver.
<rtw_wifi_config_t>	The structure is used to describe the setting when configure the network.
<rtw_maclist_t>	The structure is used to describe the maclist.
<rtw_bss_info_t>	The structure is used to describe the bss info of the network. It include the version, BSSID, beacon_period, capability, SSID, channel, atm_window, dtim_period, RSSI e.g.
<rtw_packet_filter_pattern_t>	The structure is used to set WIFI packet filter pattern.
<ieee80211_frame_info_t>	The structure is used to describe the 802.11 frame info.
<rtw_packet_filter_info_t>	The structure is used to describe the packet filter info.
<rtw_mac_filter_list_t>	The structure is used to describe the mac filter list.
<wowlan_pattern_t>	The structure is used to describe the wowlan pattern.
<psk_info>	The structure is used to describe the psk info.
<rtw_sw_statistics_t>	The structure is used to describe the sw statistics.
<rtw_phy_statistics_t>	The structure is used to describe the phy statistics.
<raw_data_desc_t>	The structure is used to describe the data description.
<wifi_user_conf>	The structure is used to describe the wifi user configuration.

20.2 Wi-Fi APIs

20.2.1 System APIs

API	Introduction
<wifi_on>	Enable Wi-Fi.
<wifi_off>	Disable Wi-Fi.
<wifi_is_running>	Check if the specified wlan_idx is running.
<wifi_set_mode>	Switch Wi-Fi Mode.

20.2.1.1 wifi_on

Enable Wi-Fi: Bring the Wireless interface "Up".

Parameter	Type	Introduction
<mode>	rtw_mode_t	Decide to enable WiFi in which mode. The optional modes are RTW_MODE_STA, RTW_MODE_AP, RTW_MODE_STA_AP, RTW_MODE_PROMISC and RTW_MODE_P2P.

Return:

- RTW_SUCCESS: If the WiFi chip initialized successfully.

- RTW_ERROR: If the WiFi chip initialization failed.

20.2.1.2 wifi_off

Disable Wi-Fi.

Parameter: None.

Return:

- RTW_SUCCESS: Deinit success, wifi mode is changed to RTW_MODE_NONE.
- RTW_ERROR: Failed

20.2.1.3 wifi_is_running

Check if the specified wlan_idx is running.

Parameter	Type	Introduction
<wlan_idx>	unsigned char	Can be set as WLAN0_IDX or WLAN1_IDX.

Return:

- 1: Success
- 0: Failed

20.2.1.4 wifi_set_mode

Switch Wifi Mode

Parameter	Type	Introduction
<mode>	rtw_mode_t	Decide to switch WiFi to which mode. The optional modes are RTW_MODE_STA, RTW_MODE_AP, RTW_MODE_STA_AP, RTW_MODE_PROMISC.

Return:

- RTW_SUCCESS: WiFi switch mode success.
- RTW_ERROR: WiFi switch mode failed.

20.2.2 Scan APIs

API	Introduction
<wifi_scan_networks>	Initiate a scan to search for 802.11 networks.
<wifi_get_scan_records>	Get scan results.
<wifi_scan_abort>	Abort ongoing wifi scan.

20.2.2.1 wifi_scan_networks

Initiate a scan to search for 802.11 networks.

Parameter	Type	Introduction
<scan_param>	rtw_scan_param_t *	Specifies the scan parameters, including scan type, specific SSID, specific channel list, channel scan time, and scan callback. There are two types of scan callback: ● scan_user_callback ● scan_report_each_mode_user_callback If registered, scan_user_callback will be executed when the scan is finished and report the total number of scanned APs, and the detailed scanned AP info can be got by calling wifi_get_scan_records. This callback is suitable for a normal asynchronous scan. If registered, scan_report_each_mode_user_callback is used when configuring RTW_SCAN_REPORT_EACH in options of rtw_scan_param, and it will be executed every time a AP is scanned, and the AP info will be directly reported by this callback.
<block>	unsigned char	If set to 1, it's synchronized scan and this API will return after scan is done. If set to 0, it's asynchronized scan and this API will return immediately.

Return:

- RTW_SUCCESS: Success for asynchronized scan.
- RTW_ERROR: Failed.
- Otherwise: Scanned AP number for synchronized scan.

NOTE

- *If this API is called, the scanned APs are stored in Wi-Fi driver dynamic allocated memory, for a synchronous scan or asynchronous scan which does not use RTW_SCAN_REPORT_EACH, these memories will be freed when wifi_get_scan_records is called.*
- *When configuring TW_SCAN_REPORT_EACH, scan_report_each_mode_user_callback will report NULL to indicate that the scan is done.*
- *Only one callback function can be chosen to register. Both scan_user_callback and scan_report_each_mode_user_callback are not supported to be registered in one scan.*
- *The scan callback function will be executed in the context of the RTW thread.*
- *When scanning specific channels, devices with strong signal strength on nearby channels may be detected.*

20.2.2.2 wifi_get_scan_records

Get scan results.

Parameter	Type	Introduction
<AP_num>	unsigned int *	Input the pointer to the number of scanned ap info which want to get, output the number of scanned ap info which can actually get.
<scan_buf>	char *	Pointer to the buf where scan result will be stored, the scanned AP info will be stored one by one in form of struct rtw_scan_result_t.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed

NOTE

For an asynchronous scan configuring RTW_SCAN_REPORT_EACH, every time an AP is scanned, the AP info will be directly reported through scan_report_each_mode_user_callback and freed after user callback is executed, thus there is no need to use this function to get the scan result.

20.2.2.3 wifi_scan_abort

Abort ongoing scan.

Parameter: None.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed

NOTE

This is an asynchronous function and will return immediately. Return value only indicates whether the scan abort command is successfully notified to the driver or not. When the scan is actually aborted, the user callback registered in wifi_scan_networks will be executed. If there is no Wi-Fi scan in progress, this function will just return RTW_SUCCESS and user callback won't be executed.

20.2.3 Connection APIs

API	Introduction
<wifi_connect>	Join a Wi-Fi network with a specified SSID or BSSID.
<wifi_disconnect>	Disassociates from current Wi-Fi network.
<wifi_is_connected_to_ap>	Check if Wi-Fi has connected to AP before DHCP.
<wifi_get_join_status>	Get latest Wi-Fi join status.
<wifi_get_disconn_reason_code>	Get reason code of latest disassociation or de-authentication.
<wifi_config_autoreconnect>	Set reconnection mode with configuration.
<wifi_get_autoreconnect>	Get the result of setting reconnection mode.

20.2.3.1 wifi_connect

Join a Wi-Fi network with a specified SSID or BSSID. Scan for, associate, and authenticate with a Wi-Fi network. On successful return, the system is ready to send data packets.

Parameter	Type	Introduction
<connect_param>	rtw_network_info_t *	The pointer of a struct which store the connection info, including ssid, bssid, password, etc, for details, please refer to struct rtw_network_info_t in wifi_structures.h.
<block>	unsigned char	If block is set to 1, it means synchronized wifi connect, and this API will return until connect is finished; if block is set to 0, it means asynchronized wifi connect, and this API will return immediately.

Return:

- RTW_SUCCESS: When the system is joined for synchronized wifi connect, when connect cmd is set successfully for asynchronized wifi connect.
- RTW_ERROR: If an error occurred.

NOTE

- *Make sure the Wi-Fi is enabled before invoking this function (wifi_on()).*
- *The parameter channel and pscan_option in connect_param can be used to perform fast survey on the specified channel during Wi-Fi connection.*
- *When the channel is set to a specified channel and pscan_option is set to PSCAN_FAST_SURVEY, during Wi-Fi connection, an active scan will be only performed on the specified channel, the active scan will retry at most 8 times with each round interval 25ms.*
- *joinstatus_user_callback in connect_param can be registered to get the real-time join status changes since this callback will be executed every time join status is changed.*

20.2.3.2 wifi_disconnect

Disassociates from current Wi-Fi network.

Parameter: None.

Return:

- RTW_SUCCESS: On successful disassociation from the AP.
- RTW_ERROR: If an error occurred.

20.2.3.3 wifi_is_connected_to_ap

Check if Wi-Fi has connected to AP before DHCP.

Parameter: None.

Return:

- RTW_SUCCESS: If connected.
- RTW_ERROR: If not connected.

20.2.3.4 wifi_get_join_status

Get the latest Wi-Fi join status.

Parameter: None.

Return:

- RTW_JOINSTATUS_UNKNOWN: Unknown join status.
- RTW_JOINSTATUS_STARTING: Join is starting.
- RTW_JOINSTATUS_SCANNING: Scan is in progress.
- RTW_JOINSTATUS_AUTHENTICATING: Authentication is in progress.
- RTW_JOINSTATUS_AUTHENTICATED: Already authenticated.
- RTW_JOINSTATUS_ASSOCIATING: Association is in progress.
- RTW_JOINSTATUS_ASSOCIATED: Already associated.
- RTW_JOINSTATUS_4WAY_HANDSHAKING: 4-way handshake is in progress.
- RTW_JOINSTATUS_4WAY_HANDSHAKE_DONE: 4-way handshake is done.
- RTW_JOINSTATUS_SUCCESS: Join is success.
- RTW_JOINSTATUS_FAIL: Join is failed.
- RTW_JOINSTATUS_DISCONNECT: Disconnected.

NOTE

Wi-Fi join status will be set during Wi-Fi connection and Wi-Fi disconnection.

20.2.3.5 wifi_get_disconn_reason_code

Present the reason code of the latest disassociation or de-authentication.

Parameter	Type	Introduction
<reason_code>	unsigned short *	A pointer to the variable where the reason code will be written.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.2.3.6 wifi_config_autoreconnect

Set reconnection mode with configuration.

Parameter	Type	Introduction
<mode>	__u8	Set 1/0 to enalbe/disable the reconnection mode.
<retry_times>	__u8	The number of retry limit.
<timeout>	__u16	The timeout value (in seconds).

Return:

- 0: Success.
- -1: Failed.

20.2.3.7 wifi_get_autoreconnect

Get the result of setting reconnection mode.

Parameter	Type	Introduction
<mode>	__u8 *	Point to the result of setting reconnection mode.

Return:

- 0: Success.
- -1: Failed.

20.2.4 Channel APIs

API	Introduction
<wifi_set_channel>	Set the listening channel for promiscuous mode. Promiscuous mode will receive all the packets in this channel.
<wifi_get_channel>	Get the current channel on STA interface(WLAN0_NAME).

20.2.4.1 wifi_set_channel

Set the listening channel for promiscuous mode. Promiscuous mode will receive all the packets in this channel.

Parameter	Type	Introduction
<channel>	int	The desired channel.

Return:

- RTW_SUCCESS: If the channel is successfully set.
- RTW_ERROR: If the channel is not successfully set.

NOTE

Do not need to call this function for STA mode Wi-Fi driver, since it will be determined by the channel from the received beacon.

20.2.4.2 wifi_get_channel

Get the current channel on STA interface(WLAN0_NAME).

Parameter	Type	Introduction
<channel>	int *	A pointer to the variable where the channel value will be written.

Return:

- RTW_SUCCESS: If the channel is successfully read.
- RTW_ERROR: If the channel is not successfully read.

20.2.5 Power Save API

API	Introduction
<wifi_set_powersave_mode>	Set IPS/LPS mode.

20.2.5.1 wifi_set_powersave_mode

Set IPS/LPS mode.

- IPS is the abbreviation of Inactive Power Save mode. Wi-Fi automatically turns RF off if it is not associated with AP.
- LPS is the abbreviation of Leisure Power Save mode. Wi-Fi automatically turns RF off during the association with AP if traffic is not busy, while it also automatically turns RF on to listen to the beacon of the associated AP.

Parameter	Type	Introduction
<ips_mode>	u8	The desired ips mode, which can be: ● IPS_MODE_NONE: leave IPS ● IPS_MODE_NORMAL: enable to enter IPS ● IPS_MODE_RESUME: resume to the last IPS mode which recorded in Wi-Fi driver
<lps_mode>	u8	The desired LPS mode, which can be: ● LPS_MODE_NONE: leave LPS ● LPS_MODE_NORMAL: enable to enter LPS ● LPS_MODE_RESUME: resume to the last LPS mode which recorded in Wi-Fi driver

Return:

- RTW_SUCCESS: If setting the corresponding mode successful.
- RTW_ERROR: Failed.

20.2.6 AP Mode APIs

API	Introduction
<wifi_start_ap>	Trigger Wi-Fi driver to start an infrastructure Wi-Fi network.
<wifi_get_associated_client_list>	Get the associated clients with SoftAP.
<wifi_del_station>	Delete a STA.

20.2.6.1 wifi_start_ap

Trigger Wi-Fi driver to start an infrastructure Wi-Fi network.

Parameter	Type	Introduction
<softAP_config>	rtw_softap_info_t *	The pointer of a struct which store the softAP configuration, please refer to struct rtw_softap_info_t in wifi_structures.h.

Return:

- RTW_SUCCESS: If successfully creates an AP.
- RTW_ERROR: If an error occurred.

20.2.6.2 wifi_get_associated_client_list

Get the associated clients with SoftAP.

Parameter	Type	Introduction
<client_list_buffer>	void *	The location where the client list will be stored.
<buffer_length>	unsigned short	The buffer length is reserved for future use. Currently, buffer length is set to a fixed value: 25.

Return:

- RTW_SUCCESS: The result is successfully got.
- RTW_ERROR: The result is not successfully got.

20.2.6.3 wifi_del_station

Delete a STA.

Parameter	Type	Introduction
<wlan_idx>	unsigned char	The wlan interface index, can be WLAN0_IDX or WLAN1_IDX.
<hwaddr>	unsigned char *	The pointer to the MAC address of the STA which will be deleted.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.2.7 Raw frame Tx API

API	Introduction
<wifi_send_raw_frame>	Send raw frame.

20.2.7.1 wifi_send_raw_frame

Send raw frame.

Parameter	Type	Introduction
<raw_data_desc>	raw_data_desc_t *	The pointer of a descriptor about the raw frame, including the buffer address where the frame is stored, frame length, the initial Tx rate of this frame (the default initial Tx rate will be 1Mbps).

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.2.8 Custom IE APIs

API	Introduction
<wifi_add_custom_ie>	Setup custom IE list.
<wifi_update_custom_ie>	Update the item in custom IE list.
<wifi_del_custom_ie>	Delete custom IE list.

NOTE

These three APIs are only effective on beacon, probe request, and probe response frames.

20.2.8.1 wifi_add_custom_ie

Setup custom IE list.

Parameter	Type	Introduction
<cus_ie>	void *	Pointer to custom IE list.
<ie_num>	int	The number of custom IE list.

Return:

- 0: Success.
- -1: Failed.

NOTE

This API cannot be executed twice before deleting the previous custom IE list.

20.2.8.2 wifi_update_custom_ie

Update the item in custom IE list.

Parameter	Type	Introduction
<cus_ie>	void *	Pointer to custom IE list.
<ie_index>	int	Index of custom IE list.

Return:

- 0: Success.
- -1: Failed.

20.2.8.3 wifi_del_custom_ie

Delete custom IE list.

Parameter: None.

Return:

- 0: Success.
- -1: Failed.

20.2.9 Wi-Fi Setting APIs

API	Introduction
<wifi_get_mac_address>	Retrieves the current Media Access Control (MAC) address (or Ethernet hardware address) of the 802.11 device.
<wifi_get_setting>	Get current Wi-Fi setting from driver.
<wifi_set_network_mode>	Set the network mode according to the data rate it supported.
<wifi_set_mfp_support>	Set Management Frame Protection Support.
<wifi_set_group_id>	Set group id of SAE.
<wifi_set_pmk_cache_enable>	Enable or disable pmk cache.
<wifi_psk_info_set>	Set psk related info, including ssid, passphrase, psk.
<wifi_psk_info_get>	Get psk related info, including ssid, passphrase, psk.
<wifi_get_ccmp_key>	Get enctryption ccmp key used by wifi (sta mode only).
<wifi_get_sw_statistic>	Show the TX and RX statistic information which counted by software (wifi driver, not phy layer).
<wifi_fetch_phy_statistic>	Fetch statistic info about wifi.
<wifi_set_indicate_mgnt>	Configure mode of HW indicating packets(mgnt and data) and SW reporting packets to wifi_indication().
<wifi_get_antenna_info>	Get antenna infomation.
<wifi_get_auto_chl>	Get an auto channel.
<wifi_get_band_type>	Get band type.
<wifi_get_tsf_low>	Get wifi TSF register[31:0].

20.2.9.1 wifi_get_mac_address

Retrieves the current Media Access Control (MAC) address (or Ethernet hardware address) of the 802.11 device.

Parameter	Type	Introduction
<mac>	rtw_mac_t *	Pointer to the struct rtw_mac_t which contain obtained mac address.

Return:

- 0: Success.
- -1: Failed.

20.2.9.2 wifi_get_setting

Get current Wi-Fi setting from driver.

Parameter	Type	Introduction
<wlan_idx>	unsigned char	WLAN0_IDX or WLAN1_IDX.
<psetting>	rtw_wifi_setting_t *	Points to the rtw_wifi_setting_t structure which information is gotten.

Return:

- RTW_SUCCESS: The result is successfully got.
- RTW_ERROR: The result is not successfully got.

20.2.9.3 wifi_set_network_mode

Set the network mode according to the data rate it supported. The driver works in BGN mode in default after driver initialization.

This function is used to change wireless network mode for station mode before connecting to AP.

Parameter	Type	Introduction

<mode>	rtw_network_mode_t	Network mode to set. The value can be: RTW_NETWORK_B, RTW_NETWORK_BG and RTW_NETWORK_BGN.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.2.9.4 wifi_set_mfp_support

Set Management Frame Protection Support.

Parameter	Type	Introduction
<value>	unsigned char	The value can be: ● NO_MGMT_FRAME_PROTECTION: not support ● MGMT_FRAME_PROTECTION_OPTIONAL: capable ● MGMT_FRAME_PROTECTION_REQUIRED: required

Return:

- RTW_SUCCESS: If setting Management Frame Protection Support successful.
- RTW_ERROR: Failed.

20.2.9.5 wifi_set_group_id

Set group id of SAE.

Parameter	Type	Introduction
<value>	unsigned char	Group id which want to be set

Return:

- RTW_SUCCESS: If setting is successful.
- RTW_ERROR: Failed.

20.2.9.6 wifi_set_pmk_cache_enable

Enable or disable pmk cache.

Parameter	Type	Introduction
<value>	unsigned char	The value can be: ● 1: enable ● 0: disable

Return:

- RTW_SUCCESS: If setting is successful.
- RTW_ERROR: Failed.

20.2.9.7 wifi_psk_info_set

Set psk related info, including ssid, passphrase, psk.

Parameter	Type	Introduction
<psk_data>	struct psk_info *	Pointer to the structure that can set psk related info in driver.

Return: None.

20.2.9.8 wifi_psk_info_get

Get psk related info, including ssid, passphrase, psk.

Parameter	Type	Introduction
<psk_data>	struct psk_info *	Pointer to the structure that will restore psk related info.

Return: None.

20.2.9.9 wifi_get_ccmp_key

Get enctryption ccmp key used by wifi (sta mode only).

Parameter	Type	Introduction
<uncst_key>	unsigned char *	The location where the CCMP TK (temporary key) will be stored.

<group_key>	unsigned char *	The location where the CCMP GTK (group key) will be stored.

Return:

- RTW_SUCCESS: The result is successfully got.
- RTW_ERROR: The result is not successfully got.

20.2.9.10 wifi_get_sw_statistic

Show the TX and RX statistic information which counted by software (wifi driver, not phy layer).

Parameter	Type	Introduction
<idx>	unsigned char	The wlan interface index, can be WLAN0_IDX or WLAN1_IDX.
<sw_statistics>	rtw_sw_statistics_t *	The pointer to the structure where store the software statistics.

Return: None.

20.2.9.11 wifi_fetch_phy_statistic

Fetch statistic info about wifi.

Parameter	Type	Introduction
<phy_statistic>	rtw_phy_statistics_t *	The pointer to the structure that stores the PHY statistics.

Return:

- RTW_SUCCESS: If the statistic info is successfully got.
- RTW_ERROR: If the statistic info is not successfully got.

20.2.9.12 wifi_set_indicate_mgnt

Configure mode of HW indicating packets(mgnt and data) and SW reporting packets to wifi_indication().

Parameter	Type	Introduction
<enable>	int	Value of enable could be: ● WIFI_INDICATE_DISABLE: disable mode (default), HW only indicates BSSID-matched packets and SW doesn't report. ● WIFI_INDICATE_NORMAL: HW only indicates BSSID-matched packets and SW reports. ● WIFI_INDICATE_WILD: HW indicates all packets and SW reports.

Return: None.

20.2.9.13 wifi_get_antenna_info

Get antenna infomation.

Parameter	Type	Introduction
<antenna>	unsigned char *	Point to the antenna value obtained from driver, 0 means main antenna, 1 means aux antenna.

Return:

- 0: Success.
- -1: Failed.

20.2.9.14 wifi_get_auto_chl

Get an auto channel.

Parameter	Type	Introduction
<wlan_idx>	unsigned char	The wlan interface index, can be WLAN0_IDX or WLAN1_IDX.
<channel_set>	unsigned char *	The pointer to the channel set which auto channel will be selected from.
<channel_num>	unsigned char	The number of channel in channel set.

Return:

- The selected auto channel.

20.2.9.15 wifi_get_band_type

Get Wi-Fi band type.

Parameter: None.

Return:

- WL_BAND_2_4G: only 2.4G supported
- WL_BAND_5G: only 5G supported
- WL_BAND_2_4G_5G_BOTH: both 2.4G and 5G supported

20.2.9.16 wifi_get_tsf_low

Get wifi TSF register[31:0].

Parameter	Type	Introduction
<port_id>	unsigned char	Specify the Wi-Fi port which could be 0 or 1. ● For STA mode and SoftAP mode, use port 0. ● For concurrent mode, STA uses port 0 and AP uses port 1.

Return:

- TSF[31:0] or 0

20.2.10 Wi-Fi Indication APIs

API	Introduction
<init_event_callback_list>	Initialize the event callback list.
<wifi_reg_event_handler>	Register the event listener.
<wifi_unreg_event_handler>	Un-register the event listener.

20.2.10.1 init_event_callback_list

Initialize the event callback list.

Parameter: None.

Return: None.

NOTE

Make sure this function has been invoked before using the event handler-related mechanism.

20.2.10.2 wifi_reg_event_handler

Register the event listener.

Parameter	Type	Introduction
<event_cmds>	unsigned int	The event command number indicated.
<handler_func>	rtw_event_handler_t	The callback function which will receive and process the event.
<handler_user_data>	void *	User specific data that will be passed directly to the callback function.

Return:

- RTW_SUCCESS: If successfully registers the event.
- RTW_ERROR: If an error occurred.

NOTE

Setting the same even_cmds with empty handler_func will unregister the event_cmds.

20.2.10.3 wifi_unreg_event_handler

Un-register the event listener.

Parameter	Type	Introduction
<event_cmds>	unsigned int	The event command number indicated.
<handler_func>	rtw_event_handler_t	The callback function which will receive and process the event.

Return:

- RTW_SUCCESS: If successfully un-registers the event.
- RTW_ERROR: If an error occurred.

20.2.11 Promisc APIs

API	Introduction
<wifi_enter_promisc_mode>	Let Wi-Fi enter promiscuous mode.
<wifi_set_promisc>	Set the chip to start or stop the promiscuous mode.
<wifi_init_packet_filter>	Initialize packet filter related data.
<wifi_add_packet_filter>	Add packet filter.
<wifi_remove_packet_filter>	Remove the packet filter.
<wifi_enable_packet_filter>	Enable the packet filter.
<wifi_disable_packet_filter>	Disable the packet filter.

20.2.11.1 wifi_enter_promisc_mode

Let Wi-Fi enter promiscuous mode.

Parameter: None.

Return: None.

NOTE

If Wi-Fi is originally in concurrent mode or SoftAP mode, a mode switch will be performed; if Wi-Fi is originally in STA mode and already connected to AP, the connection to AP will disconnect.

20.2.11.2 wifi_set_promisc

Set the chip to start or stop the promiscuous mode.

Parameter	Type	Introduction
<enabled>	rtw_rcr_level_t	The value can be: ● 0: disable the promisc. ● 1: enable the promisc special for all ethernet frames. ● 2: enable the promisc special for Broadcast/Multicast ethernet frames. ● 3: enable the promisc special for all 802.11 frames. ● 4: enable the promisc special for Broadcast/Multicast 802.11 frames.
<callback>	void (*)(unsigned char *, unsigned int, void *)	The callback function which will receive and process the network data.
<len_used>	unsigned char	Specify if the the promisc data length is used. If len_used set to 1, packet (frame data) length will be saved and transferred to callback function.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

NOTE

Use wifi_enter_promisc_mode() to make sure Wi-Fi in correct mode, and use wifi_set_promisc() to enable promisc receiving and promisc callback.

20.2.11.3 wifi_init_packet_filter

Initialize packet filter related data.

Parameter: None.

Return: None.

20.2.11.4 wifi_add_packet_filter

Add packet filter.

Parameter	Type	Introduction
<filter_id>	unsigned char	The filter id.
<patt>	rtw_packet_filter_pattern_t*	Point to the filter pattern.
<rule>	rtw_packet_filter_rule_t	Point to the filter rule, the value could be:

		• RTW_POSITIVE_MATCHING: accept the frame if matching with this pattern, otherwise discard the frame. • RTW_NEGATIVE_MATCHING: discard the frame if matching with this pattern, otherwise accept the frame.

Return:

- 0: Success.
- -1: Failed.

NOTE

This is software filter in promisc mode.

20.2.11.5 wifi_remove_packet_filter

Remove the packet filter.

Parameter	Type	Introduction
<filter_id>	unsigned char	The filter id, the value can be 0 to 4.

Return:

- 0: Success.
- -1: Failed.

20.2.11.6 wifi_enable_packet_filter

Enable the packet filter.

Parameter	Type	Introduction
<filter_id>	unsigned char	The filter id, the value can be 0 to 4.

Return:

- 0: Success.
- -1: Failed.

20.2.11.7 wifi_disable_packet_filter

Disable the packet filter.

Parameter	Type	Introduction
<filter_id>	unsigned char	The filter id, the value can be 0 to 4.

Return:

- 0: Success.
- -1: Failed.

20.2.12 Mac filter APIs

API	Introduction
<wifi_init_mac_filter>	Initialize mac address filter list.
<wifi_add_mac_filter>	Add mac address to mac filter list then this address will be rejected during authentication.
<wifi_del_mac_filter>	Delete mac address from mac filter list.

NOTE

These APIs should be used only when operating as softAP.

20.2.12.1 wifi_init_mac_filter

Initialize mac address filter list.

Parameter: None.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.2.12.2 wifi_add_mac_filter

Add mac address to mac filter list then this address will be rejected during authentication.

Parameter	Type	Introduction
<hwaddr>	unsigned char *	The mac address which will be added to mac filter list.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.2.12.3 wifi_del_mac_filter

Delete mac address from mac filter list.

Parameter	Type	Introduction
<hwaddr>	unsigned char *	The mac address which will be deleted from mac filter list.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

20.3 Fast Connection

This section illustrates the principle of fast connection and how to implement user's own fast connection code.

Fast connection is used to reconnect with AP automatically after Wi-Fi is initialized, the principle is to store the AP information in Flash and reconnect to AP after Wi-Fi is initialized.

20.3.1 Implementation

20.3.1.1 AP Information Storage

User should implement a function to write AP information to Flash, just like demo function write_fast_connect_data_to_flash() in the source file wifi_fast_connect.c. In this function, you should reserve some space for AP information, and write the AP information to the reserved space in a pre-defined data format. The address of the function must be assigned to the global variable p_store_fast_connect_info. After Wi-Fi connection is successful, if p_store_fast_connect_info points to a valid address, write_fast_connect_data_to_flash() will be called.

NOTE

The path of source codes is SDK/component/wifi/wifi_fast_connect/wifi_fast_connect.c.

20.3.1.2 Reconnection

User can implement his own function to read AP information from Flash and connect to AP, just like demo function wifi_do_fast_connect() in the example code. The address of this function must be assigned to the global variable p_wifi_do_fast_connect, which should be defined before Wi-Fi initialization. After Wi-Fi is initialized, if p_wifi_do_fast_connect points to a valid address, this function will be called.

20.3.1.3 Fast Connection Data Erase

User should implement his own function to erase fast connection data, just like demo function Erase_Fastconnect_data() in the source code.

20.3.2 APIs

API	Introduction
<write_fast_connect_data_to_flash>	Wi-Fi connection indication trigger this function to save current Wi-Fi profile in flash.
<wifi_do_fast_connect>	Read previous saved Wi-Fi profile in flash and execute connection.

20.3.2.1 write_fast_connect_data_to_flash

Wi-Fi connection indication trigger this function to save current Wi-Fi profile in flash.

Parameter	Type	Introduction
<offer_ip>	unsigned int	DHCP offered IP.
<server_ip>	unsigned int	DHCP server IP.

Return:

- RTW_SUCCESS: Success.
- RTW_ERROR: Failed.

NOTE

offer_ip and server_ip will be used only when CONFIG_FAST_DHCP is set to 1, they will be stored to Flash for fast DHCP. If CONFIG_FAST_DHCP is not configured to 1, offer_ip and server_ip are unaffected. Other Wi-Fi profiles will be directly got by API wifi_get_setting() and wifi_psk_info_get(), not depending on the input parameter of this function.

20.3.2.2 wifi_do_fast_connect

Read previous saved Wi-Fi profile in flash and execute connection.
Parameter: None.
Return:

- 0: Success.
- -1: Failed.

20.4 WPS APIs

API	Introduction
<wps_start>	Start WPS enrollee process.
<wps_stop>	Stop WPS enrollee process.

20.4.1 wps_start

Start WPS enrollee process.

Parameter	Type	Introduction
<wps_config>	u16	WPS configure method. Options are: ● WPS_CONFIG_DISPLAY ● WPS_CONFIG_KEYPAD ● WPS_CONFIG_PUSHBUTTON
<pin>	char *	PIN number. Can be set to NULL if using WPS_CONFIG_PUSHBUTTON.
<channel>	u8	Channel. Currently un-used, can be set to 0.
<ssid>	char *	Target network SSID. Can be set to NULL if no target network specified.

Return:

- 0: Success.
- Otherwise: Failed.

NOTE

- *Before invoking this function, the Wi-Fi should be enabled by calling wifi_on().*
- *Make sure CONFIG_ENABLE_WPS is enabled in platform_opts.h. After calling wps_start(), the longest time of WPS is 120s. You can call wps_stop() to quit WPS.*

20.4.2 wps_stop

Stop WPS enrollee process.
Parameter: None.
Return: None.

NOTE

Make sure CONFIG_ENABLE_WPS is enabled in platform_opts.h.

20.5 Wi-Fi event callback function

20.5.1 Register Wi-Fi event callback function

The callback function is defined by user and need to be registered.

API	Introduction
<wifi_reg_event_handler>	Register callback function to the event listener.

Parameter	Type	Introduction
<event_cmds>	unsigned int	The event number indicated.
<handler_func>	rtw_event_handler_t	The callback function which will receive and process the event.
<handler_user_data>	void *	User specific data that will be passed directly to the callback function.

20.5.2 Wi-Fi event introduction

```
enum _WIFI_EVENT_INDICATE {
 /* common event */
 WIFI_EVENT_STA_ASSOC = 0,                //used in p2p, simple config, 11s, customer
 WIFI_EVENT_STA_DISASSOC,                 //used in p2p, customer
 WIFI_EVENT_RX_MGNT,                      //used in p2p, customer
 WIFI_EVENT_CONNECT,                      //used in ipv6 example, p2p, wifi manager example
 WIFI_EVENT_DISCONNECT,                   //used in wifi_manager example, p2p, eap, atcmd……

 WIFI_EVENT_GROUP_KEY_CHANGED,
 WIFI_EVENT_RECONNECTION_FAIL,
 WIFI_EVENT_ICV_ERROR,
 WIFI_EVENT_CHALLENGE_FAIL,

 WIFI_EVENT_JOIN_STATUS,                  //internally use for joinstatus indicate

 /* p2p event */
 WIFI_EVENT_P2P_SEND_ACTION_DONE = 15,

 /* wpa event */
 WIFI_EVENT_WPA_STA_WPS_START = 20,
 WIFI_EVENT_WPA_WPS_FINISH,
 WIFI_EVENT_WPA_EAPOL_START,
 WIFI_EVENT_WPA_EAPOL_RECVD,

 /* 11s event */
 WIFI_EVENT_11S_PATHSEL_GEN_RREQ = 59,
 WIFI_EVENT_11S_PATHSEL_GEN_RERR,
 WIFI_EVENT_11S_PATHSEL_RECV_RREQ,
 WIFI_EVENT_11S_PATHSEL_RECV_RREP,
 WIFI_EVENT_11S_PATHSEL_RECV_RERR,
 WIFI_EVENT_11S_PATHSEL_RECV_PANN,
 WIFI_EVENT_11S_PATHSEL_RECV_RANN,

 WIFI_EVENT_11S_PATHSEL_GEN_PREQ = 150,
 WIFI_EVENT_11S_PATHSEL_GEN_PERR,
 WIFI_EVENT_11S_PATHSEL_RECV_PREQ,
 WIFI_EVENT_11S_PATHSEL_RECV_PREP,
 WIFI_EVENT_11S_PATHSEL_RECV_PERR,
 WIFI_EVENT_11S_PATHSEL_RECV_GANN,

 /* csi rx done event */
 WIFI_EVENT_CSI_DONE = 160,

 WIFI_EVENT_MAX,
};
```

20.5.2.1 WIFI_EVENT_STA_ASSOC

Indicate client associated in AP mode

Parameter	Type	Introduction
<buf>	char *	Assoc Request Frame sent by client
<buf_len>	int	Size of Assoc Request Frame
<flags>	int	Unused

20.5.2.2 WIFI_EVENT_STA_DISASSOC

Indicate client disassociated in AP mode

Parameter	Type	Introduction
<buf>	char *	client Mac : 6 Bytes reason code : 2 Bytes
<buf_len>	int	Size of Assoc Request Frame
<flags>	int	Unused

Reason code :

```
#define WLAN_REASON_UNSPECIFIED 1
#define WLAN_REASON_PREV_AUTH_NOT_VALID 2
#define WLAN_REASON_DEAUTH_LEAVING 3
#define WLAN_REASON_DISASSOC_DUE_TO_INACTIVITY 4
#define WLAN_REASON_DISASSOC_AP_BUSY 5
#define WLAN_REASON_CLASS2_FRAME_FROM_NONAUTH_STA 6
#define WLAN_REASON_CLASS3_FRAME_FROM_NONASSOC_STA 7
#define WLAN_REASON_DISASSOC_STA_HAS_LEFT 8
#define WLAN_REASON_STA_REQ_ASSOC_WITHOUT_AUTH 9
#define WLAN_REASON_MIC_FAILURE 14
#define WLAN_REASON_4WAY_HANDSHAKE_TIMEOUT 15
#define WLAN_REASON_ACTIVE_ROAM 65533
#define WLAN_REASON_JOIN_WRONG_CHANNEL       65534
#define WLAN_REASON_EXPIRATION_CHK 65535
```

20.5.2.3 WIFI_EVENT_RX_MGNT

Indicate management frame received. Need to call API wifi_set_indicate_mgnt(1) to enable this event.

Parameter	Type	Introduction
<buf>	char *	Management Frame
<buf_len>	int	Size of Management Frame
<flags>	int	channel : 1 Byte

20.5.2.4 WIFI_EVENT_CONNECT

Indicate station connect to AP.

Parameter	Type	Introduction
<buf>	char *	AP BSSID : 6 Bytes
<buf_len>	int	6
<flags>	int	Unused

20.5.2.5 WIFI_EVENT_DISCONNECT

Indicate station disconnect with AP.

Parameter	Type	Introduction
<buf>	char *	Null Mac : 6 Bytes reason code : 2 Bytes
<buf_len>	Int	8
<flags>	Int	Unused

20.5.2.6 WIFI_EVENT_RECONNECTION_FAIL

Indicate wifi reconnection failed

Parameter	Type	Introduction
<buf>	char *	"RECONNECTION FAILURE"
<buf_len>	Int	strlen of "RECONNECTION FAILURE"
<flags>	Int	Unused

20.5.2.7 WIFI_EVENT_ICV_ERROR

Indicate that the receiving packets has ICV error

Parameter	Type	Introduction
<buf>	char *	"ICV Eror"
<buf_len>	Int	strlen of "ICV Eror"
<flags>	Int	Unused

20.5.2.8 WIFI_EVENT_CHALLENGE_FAIL

Indicate authentication failed because of challenge failure

Parameter	Type	Introduction
<buf>	char *	"Auth Challenge Fail"
<buf_len>	Int	strlen of "Auth Challenge Fail"
<flags>	Int	Unused

20.5.2.9 WIFI_EVENT_WPA_STA_WPS_START

Indicate WPS process starting.

Parameter	Type	Introduction
<buf>	char *	Source Mac : 6 Bytes
<buf_len>	Int	6
<flags>	Int	Unused

20.5.2.10 WIFI_EVENT_WPA_WPS_FINISH

Indicate WPS process finish

Parameter	Type	Introduction
<buf>	char *	NULL
<buf_len>	Int	0
<flags>	Int	Unused

20.5.2.11 WIFI_EVENT_WPA_EAPOL_START

Indicate to send EAPOL_START packets in eap process

Parameter	Type	Introduction
<buf>	char *	Source Mac : 6 Bytes
<buf_len>	Int	6
<flags>	Int	Unused

20.5.2.12 WIFI_EVENT_WPA_EAPOL_RECVD

Indicate EAPOL frame received in wps/eap process

Parameter	Type	Introduction
<buf>	char *	Eapol frame
<buf_len>	Int	Size of Eapol frame
<flags>	Int	Unused

21 Peripheral

21.1 Flash spic QPI or QSPI setting

SDK bootloader sets the flash to QPI mode by default. Boot loader will use SPIC_BIT_MODE_SETTING in hal_spic.h as the parameter of flash init. Moreover, the default boot process flow can be overwritten by implementing the spic_user_select () in component\soc\8735b\misc\platform\user_boot.c. The spic_user_select () should set the *pspic_bit_mode that will be loaded by bootloader. If the SPIC_BIT_MODE_SETTING is SpicQpiMode, bootloader will init flash as QPI mode. If the SPIC_BIT_MODE_SETTING is SpicQuadIOMode, bootloader will init flash as QSPI mode. To apply spic_user_select (), it is necessary to re-build bootloader.

```
//hal_spic.h
#define SPIC_BIT_MODE_SETTING SpicQpiMode  // Default for ram code using is qpi mode
```

```
//user_boot.c
void spic user select(uint8 t *pspic bit mode)
{
 *pspic_bit_mode = SPIC_BIT_MODE_SETTING;
}
```

21.2 Aon watchdog

21.2.1 Check reboot reason was AON WDT or not

```
//wdt api.h
/**
*  @brief   Check reboot reason was AON WDT or not
*
*  @param   None
*  @retval  1:Reboot reason was AON WDT
*           0:Reboot reason wasn't AON WDT
*/
uint8_t watchdog_aon_reboot_check(void);
```

21.2.2 Clear AON WDT boot reason

User needs to call watchdog_aon_clear() to clear AON WDT boot reason.

```
//wdt_api.h
/**
*  @brief   Clear AON WDT boot reason
*
*  @param   None
*  @retval  None
*/
void watchdog_aon_clear(void);
```

22 Log System

SDK provide several mechanisms to build up log system. Default log comes out from UART, and by integrating other features, the log could be deliver to only one port, ex UART, or two ports concurrently, ex: UART + CDC.

To check how to hook lout out to call back, check sample code here:

```
extern void console_stdio_init(void *read_cb, void *write_cb);//default uart or cdc
extern void remote_stdio_init(void *read_cb, void *write_cb);//default telnet or ssh

unsigned user_read_buffer(unsigned fd, void *buf, unsigned len)
{
        //read the cdc or uart data to buf
}

unsigned user_write_buffer(unsigned fd, const void *buf, unsigned len)
{
        //write the buf to cdc or uart
}

void user_log_init(void)
{
        console_stdio_init(user_read_buffer, user_write_buffer);//Register the log system
}
```

The application side needs to complete the read, write callbacks, and register to the log system through the above sample code. Currently, there are related examples of implementation for UART LOG, USB CDC, and TELNET. For detailed methods, please refer to each example. If you need to support the output of two log services at the same time, you need to register the remote and console separately at the same time.

Revision History

Date	Version	Change
2021-09-30	V01	Initial draft
2021-11-15	V02	File System
2022-03-23	V03	NN,BT