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.

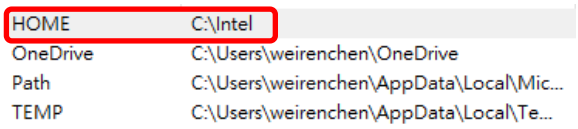
1 Building Environment

1.1 Setting up GCC Building Environment

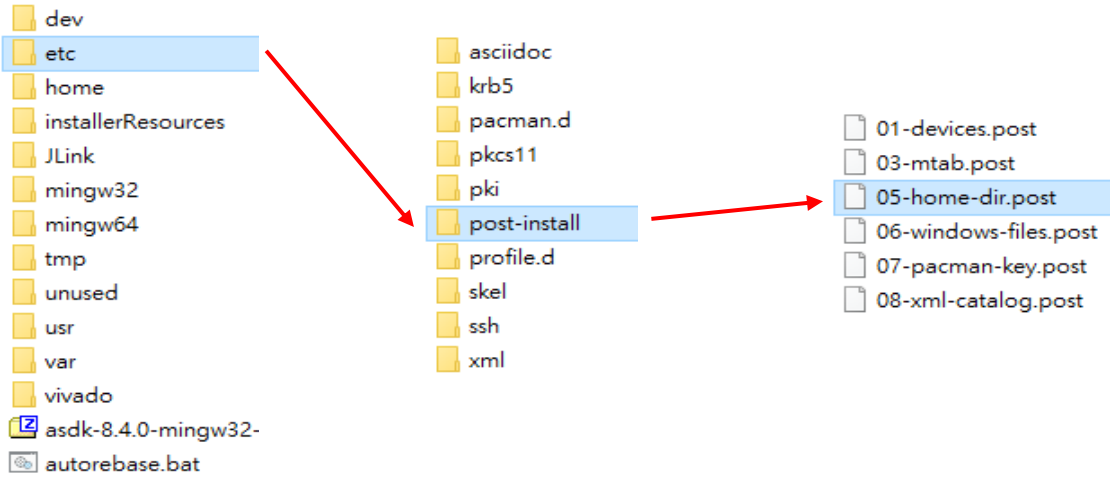
1.1.1 Building the project in GCC Building Environment (Windows)

1.1.1.1 Installing mingw with ASDK and setting up the CMake

- Download and extract msys64-1210.7z from tools folder
- Check the windows user Environment Variable



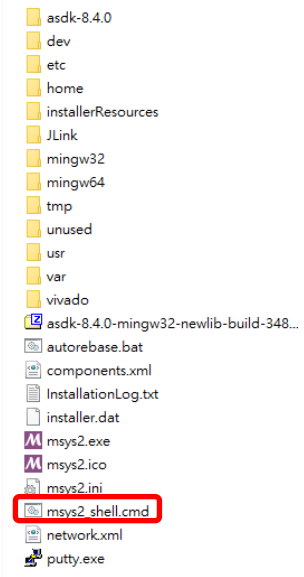
- If your windows has already have Environment Variable named HOME，open the file "/etc/post-install/05-home-dir.post



- ADD "HOME=<path to msys2_shell.cmd>/home/<folder name>". The suggested <folder name> is ${USER}. To prevent some error, recommend to use a name without blank for the <folder name>.

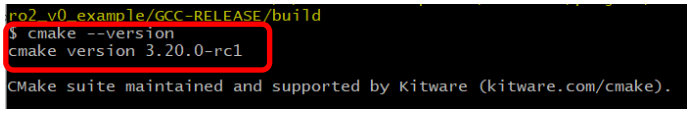
```
222  # If the home directory doesn't exist, create it.
223  HOME=C:/msys64-1210/msys64/home/${USER}
224  if [ ! -d "${HOME}" ]; then
225    if mkdir -p "${HOME}"; then
226      echo "Copying skeleton files."
227      echo "These files are for the users to personalise their msys2 experience."
228      echo
229      echo "They will never be overwritten nor automatically updated."
230      echo
231      local cur_pwd="$PWD"
232      cd /etc/skel || echo "WARNING: Failed attempt to cd into /etc/skel!"
233      local f=
234      /usr/bin/find . -type f | while read f; do
235        local fDest="${f#.}"
236        if [ ! -e "${HOME}${fDest}" -a ! -L "${HOME}${fDest}" ]; then
237          /usr/bin/install -D -p -v "${f}" "${HOME}${fDest}"
238        fi
239      done
240      cd "${cur_pwd}"
241    else
242      echo "${HOME} could not be created."
243      { [ -d "${TEMP}" ] && HOME="${TEMP}"; } ||
```

- Double click “msys2_shell.cmd” from mysys64 folder



- 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
- 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 (export PATH=<path to cmake.exe>:$PATH) in file msys64/home/<user name folder>/.bashrc

- Note: For the first time adding the CMake PATH, after adding the PATH, you need to re-open the msys2_shell and check whether the CMake is added.
- You can check the cmake installation by “cmake --version”



1.1.1.2 Adding and changing the toolchain

- Like adding PATH for cmake, user can add or change the toolchain in “msys64/home/<user name folder>/.bashrc”.

- 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:$PATH
elif [ -d "../../asdk-8.4.0" ]; then
  echo "asdk-8.4.0 exist"
  export PATH=/asdk-8.4.0/mingw32/newlib/bin:$PATH
elif [ -d "../../asdk-8.3.0" ]; then
  echo "asdk-8.3.0 exist"
  export PATH=/asdk-8.3.0/mingw32/newlib/bin:$PATH
else
  echo "unzip asdk-10.3.0"
  cd /
  #unzip asdk-8.4.0-mingw32-newlib-build-3485.zip
  unzip asdk-10.3.0-mingw32-newlib-build-3633-x86_64.zip
  export PATH=/asdk-10.3.0/mingw32/newlib/bin:$PATH
  cd ~
fi



export LANG='en_US.utf-8'
export PATH=/JLink:$PATH:/mingw64/bin
export PATH=/c/Program\ Files/cmake-3.20.0-rc1-windows/bin:$PATH
```

- Note: the recommended toolchain version is 10.3.0

1.1.1.3 Building the project

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

1.1.2 Building the project in GCC Building Environment (LINUX)

1.1.2.1 Add toolchain to the linux PATH

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

  tar -jxvf <path_to_toolcahin>/<toolchain_file(*.tar.bz2)> -C <path_to_file _extracted>

- Add the file to PATH:

  Method 1: type “export PATH=$PATH:<path_of_the_toolchain_execute_file>”
  By this method, the PATH will only effect on the terminal. It need to type the command, if you reopen a terminal.
  Method 2: add “export PATH=$PATH:<path_of_the_toolchain_execute_file>” in file /etc/profile and the PATH will add into linux system after reboot, which means that you do not need to type “export PATH=$PATH:<path_of_the_toolchain_execute_file>” when you open a new terminal.

- Note: make sure that all of the C compiler is under <path_of_the_toolchain_execute_file>

1.1.2.2 Installing cmake for linux

- Install cmake using terminal (like “sudo apt-get -y install cmake”), if the installation is successful, you can get the version after type “cmake --version”.

1.1.2.3 Building the project

- Open linux terminal and enter the project location: project/realtek_amebapro2_v0_example/GCC-RELEASE/.
- Create folder “build” and enter “build” folder.

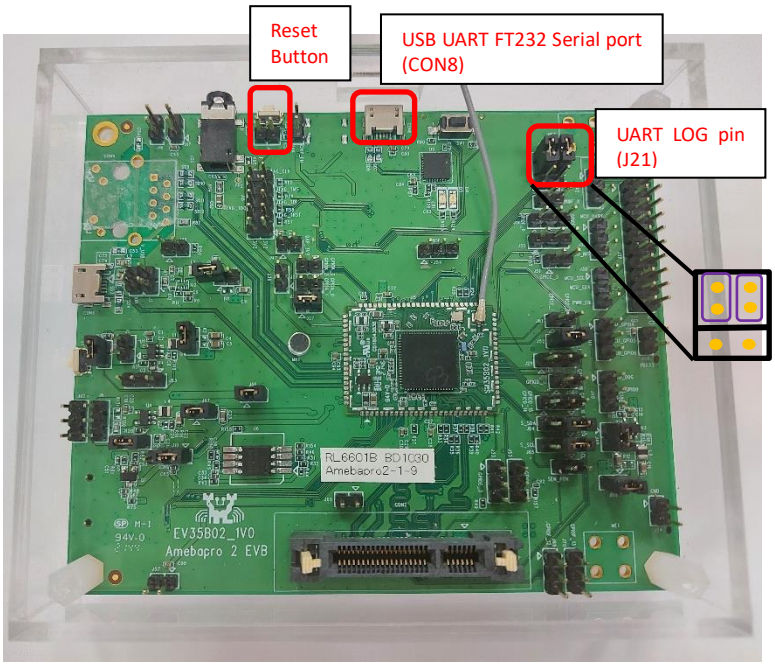
- Run “cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake” to create the makefile.
- Run “cmake --build . --target flash” to build and generate flash binary.
- Note: the folder name is not needed to be ‘build’, user can use other name.
- Note: if the folder ‘build’ has been used by other platform (like linux, mingw) or other toolchain version before, suggest to remove the CMakeCache.txt and rebuild the project.
- Note: If the folder of sdk do not have permissionn, you can give permission by “sudo chmod -R 777 <sdk folder name>”.

1.1.3 Choosing the fw image

- If building successfully, you can see flash_ntz.bin in the build folder. Download it to amebaPro2 device.

1.2 Log UART Settings

- To use AmebaPro2 log UART, the user needs to connect jumpers to **J21** for **FT232** (**CON8**). Remove the red line from pin J9 (close to audio jack one).



- 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 contain four folders. The folder “component” store the main component source and the folder “project” contains the project make file, 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 dri
	example	mmf audio examples (media_framework) utility examples: wlan_fast_connect/ssl_download/fatfs/uvc
	file_system	Fatfs DCT
	lwip	lwip API source code
	mbed	mbed API source code
	media	multi-media framework modules rtp codec for media
	network	cJSON coap dhcp http iperf mDNS mqtt ping rtsp sntp tftp websockect xml
	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
	wifi	wifi api and wifi config related source code and header files

2.2 Project

Folder	Sub-folder	Description
Project	**realtek_amebapro2_v0_example**	GCC project entry for Amebapro2
	$/GCC-RELEASE	GCC cmake projects
	$/~/application_ntz	libraries for (non trust zone) GCC project
	$/~/build	pre-build image files (boot.bin) and json files place for building cmake projects and generate flash image file (flah_ntz.bin)
	$/~/ROM	ROM code libraries
	$/inc	the header files for setting the project compile flag
	$/src	the main file source code for the project
	$/~/mmfv2_video_example	source code for mmf examples with video
	$/~/nn_verify(nn)	NN model sample code
	$/~/voe_lib	voe make file
	$/example_sources	examples for peripherals

2.3 Doc and tools

Folder	Sub-folder	Description
tools		AmebaZII_PGTool_v1.2.24 => for downloading image files to AmebaPro2 msys64-1210.7z => for building the environment of amebapro2 project (refer to 1.1)
docs		

3 GCC Makefile

Before building the amebapro2 project, you should install the mingw and CMake first. It can refer to 1.1.
After installing mingw and CMake, go to the project file located folder by command “cd <sdk-folder-location>/project/realtek_amebapro2_v0_example/GCC-RELEASE/build”.

```
asdk-8.4.0 exist

weirenchen@R074620100 MSYS ~
$ cd d:/amebapro2/sdk-ameba-v9.0a/project/realtek_amebapro2_v0_example/GCC-RELEA
SE/build
```

Use “cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake” to generate the make files according to the CMakeLists.txt
Note: If it show the error about the file is already existed, you can delete the file “CMakeCache.txt” and type the command again.

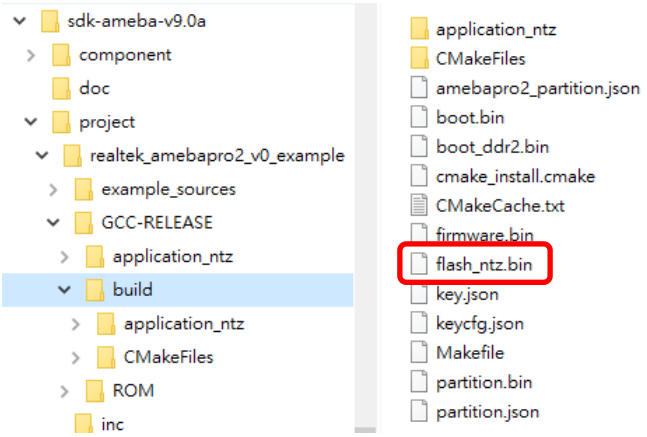
```
weirenchen@R074620100 MSYS /d/20210514_rtsp_h265/sdk_1206/project/realtek_amebap
ro2_v0_example/GCC-RELEASE/build_103
$ cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DCUTVER
=B -DEXAMPLE=media_framework
-- The C compiler identification is GNU 10.3.0
-- The CXX compiler identification is GNU 10.3.0
```

You can see “Build files have been written to:” if build successfully. After the makefile is build, you can use “cmake --build . --target flash” to build the image file through the makefiles.

```
-- Build files have been written to: D:/amebapro2/sdk-ameba-v9.0a/project/realte
k_amebapro2_v0_example/GCC-RELEASE/build

weirenchen@R074620100 MSYS /d/amebapro2/sdk-ameba-v9.0a/project/realtek_amebapro
2_v0_example/GCC-RELEASE/build
$ cmake --build . --target flash
```

If building successfully, you can see image file “flash_ntz.bin” in bui ld folder.

```
[INFO]
PTAB ==> partition.bin
BOOT ==> boot.bin
FW1 ==> firmware.bin
[100%] Built target flash
```



3.1 Adding a file in CMake project

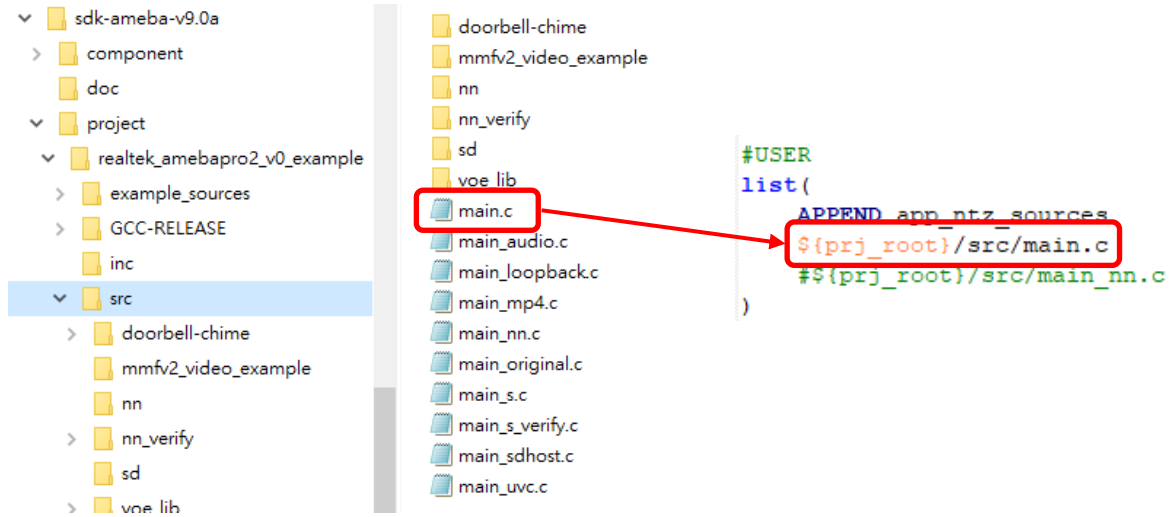
In the section, it will introduce how to add exact files to Amebapro2 project, including adding source, header files, creating and linking the library files.

3.1.1 Adding a source code file or header file

- Adding Source code file

  Open the CMakeLists.txt of ntz at “/project/realtek_amebapro2_v0_example/GCC-RELEASE/application_ntz/”.

Add the source code by append source code to app_ntz_sources (“list{ APPEND app_ntz_sources <path to source code> }”).



- Adding header file

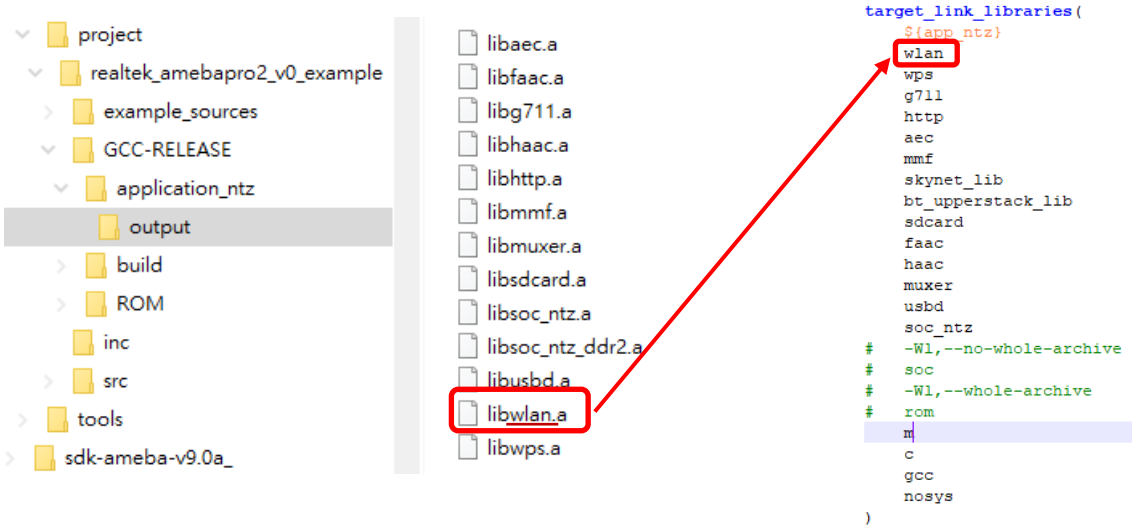
  Open the includepath.cmake at “/project/realtek_amebapro2_v0_example/GCC-RELEASE /”.
  Add the header files by append header files location to inc_path (“list{ APPEND inc_path “<path to header folder>” }”).
  Also add directory to be included by include_directories (<path to header folder>).

3.1.2 Adding a library file

- Method 1:

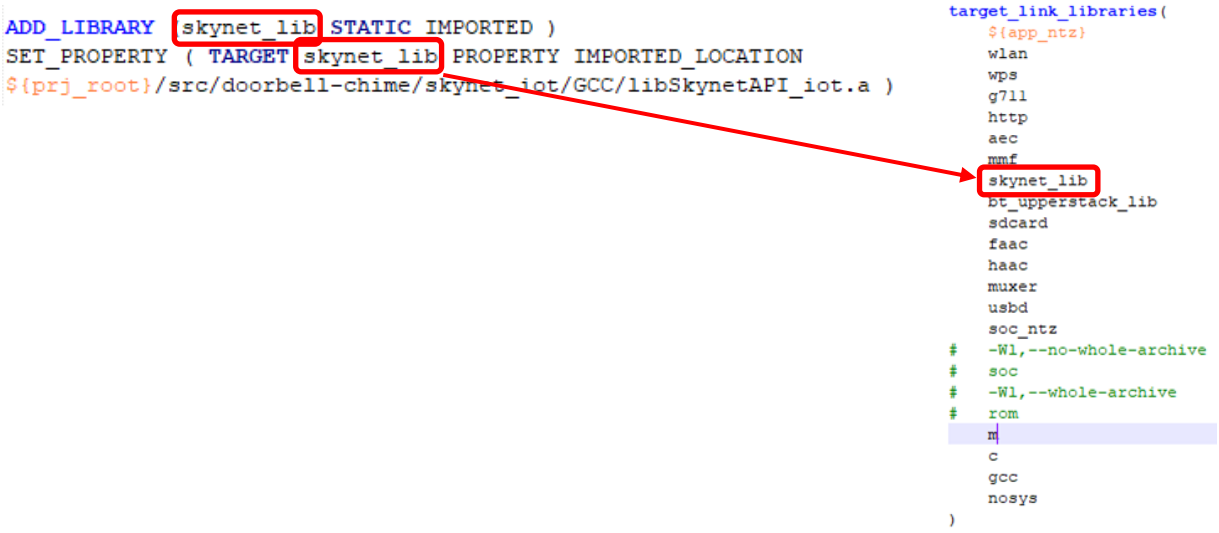
  You can place the library under “/project/realtek_amebapro2_v0_example/GCC-RELEASE/application_ntz/output” and the name need to be libxxx.

Then, you can add the library xxx by writing the name xxx inside the “target_link_libraries”.



- Method 2:

  Use “ADD_LIBRARY (<lib name> STATIC IMPORTED)” first to declare the lib name and type you will be added, where <lib_name> is the label for your library and do not need to be the same as your real library.
  Then, use “SET_PROPERTY (TARGET <lib name> PROPERTY IMPORTED_LOCATION <path-to-yourlibrary>)” or “set_target_properties (<lib name> PROPERTY IMPORTED_LOCATION <path-to-yourlibrary>)” to set up the location of the library.
  Finally, you can add the library xxx by writing the name xxx inside the target_link_libraries.



3.1.3 Building a library

- Create a cmake file for the library

  Set up required cmake version (the cmake version is just use to ensure the current 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)
```

Add the source code by append the list

```
list(
    APPEND test_sources
        ${prj_root}/src/test/test0.c
        ${prj_root}/src/test/test1.c
        ${prj_root}/src/test/test2.c
)
```

Select 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}
)
```

Add the compile flag for the library

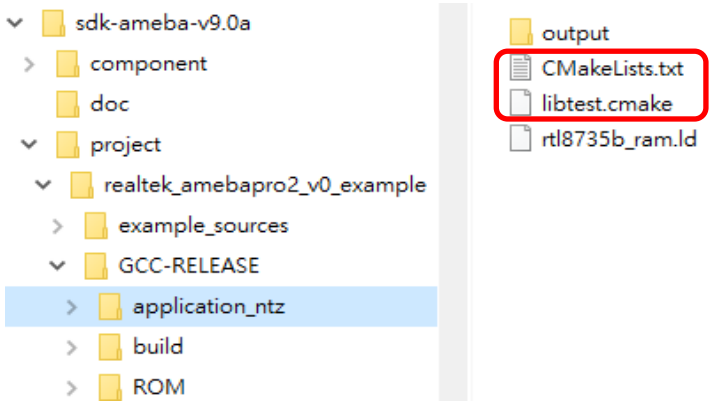
```
list(
    APPEND test_flags
    CONFIG_BUILD_ALL=1
    CONFIG_BUILD_LIB=1
    TEST_FLAGS
)

target_compile_definitions(${test} PRIVATE ${test_flags})
```

Add the header files need to be included in the library

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

    ${inc_path}
    ${prj_root}/test/include
)
```

- Add the cmake and link the library to the project

Place the cmake file (eg.libtest.cmake) and open the CMakeLists.txt of ntz at "/project/realtek_amebapro2_v0_example/GCC-RELEASE/application_ntz/".



Add the library cmake file by "include (./<cmake_filename>.cmake)" and also in target_link_libraries.

```
# root of realtek_amebapro2_v0_example
set (prj_root "${CMAKE_CURRENT_SOURCE_DIR}/../..")
# root of SDK
set (sdk_root "${CMAKE_CURRENT_SOURCE_DIR}/../../../..")
set(app_ntz application.ntz)

set(freertos "freertos_v202012.00")
set(lwip "lwip_v2.1.2")

include(../config.cmake)
link_directories(${prj_root}/GCC-RELEASE/application_ntz/output)
include(./libtest.cmake)
```

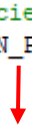
```
target_link_libraries(
    ${app_ntz}
    wlan
    wps
    g711
    test
    http
    aec
    mmf
    skynet_lib
    ht_upperstack_lib
```

After building you can get the lib file under "/project/realtek_amebapro2_v0_example/GCC-RELEASE/build/application_ntz/". You can move it to "/project/realtek_amebapro2_v0_example/GCC-RELEASE/application_ntz/output/" and remove the statement of "include (./<cmake_filename>.cmake)" in CMakeLists.txt.
Note if you need to rebuild the library, you need to remove the previous built library file.

3.1.3.1 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_ntz\CMakeFiles\<users lib name>.dir and turn on the CMAKE_DEPENDS_IN_PROJECT_ONLY.

```
# Consider dependencies only in project.
set(CMAKE_DEPENDS_IN_PROJECT_ONLY OFF)
```



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

Note: be sure the modification will have no influence when you turn on the CMAKE_DEPENDS_IN_PROJECT_ONLY.

3.2 Constructing a new application example

The application example folder of AmebaPro2 needs to extra add app_example.c and <the folder name>.cmake. The app_example.c is the entry of the example and the cmake file is for project include while building. Here are the steps for building up a new application example.

- Construct a folder under “sdk/component/example”, move the source code to the folder and extra add app_example.c and <the folder name>.cmake in the folder.
- Open app_example.c and add the entries of example under the function app_example

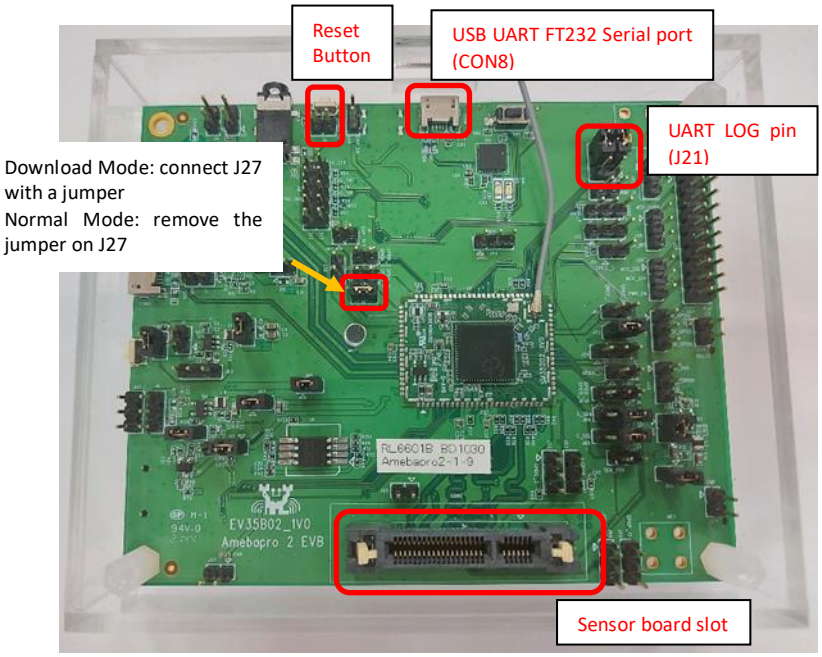
```
#include "example_audio_helix_aac.h"

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

- Append the source code, header file, compile flag and library needed under the lists in <the folder name>.cmake.
- After doing the previous steps, users can build up the new example project by “cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DEXAMPLE=<the example folder name>”and “cmake --build . --target flash”.

4 Demo Board

4.1 Demo Board overview

- Hardware: 8735 EVB * 1

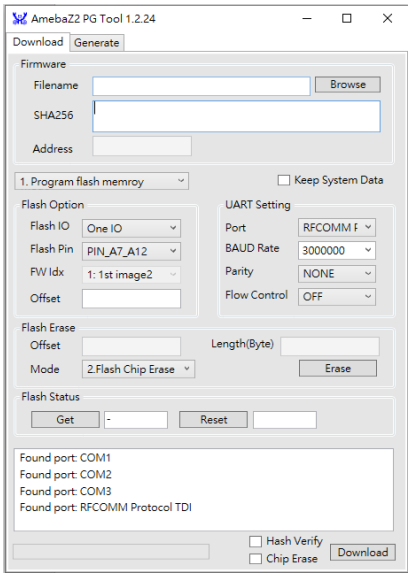


5 Image Tool

5.1 Introduction

This chapter will introduce how to use Image Tool to generate and download images. The image tool - AmebaZII_PGTool can be find in tools folder and it has two menu pages:

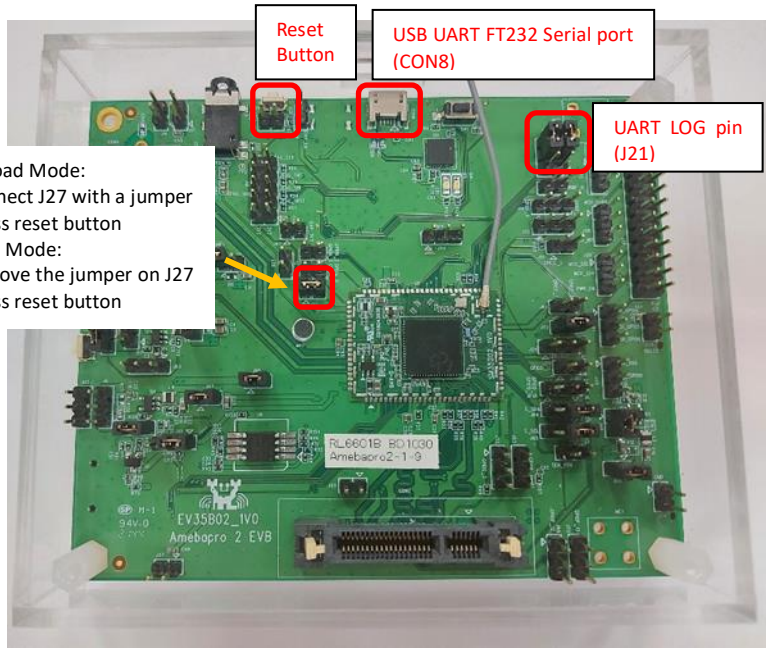
- Download: For downloading image to an amebapro2 device through UART.
- Generate: For contacting individual images to a composite image.



5.2 Download Environment Setup

5.2.1 Hardware Setup

To download image, the device must to be boot 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, connect the red line with J9 (close to phone jacket one) and press reset button to enter download mode.



5.2.2 Software Setup

- PC environment requirements: Windows 7 above with FT232 driver.
- AmebaZII_PGTool_vx.x.x.exe

5.3 Image Download

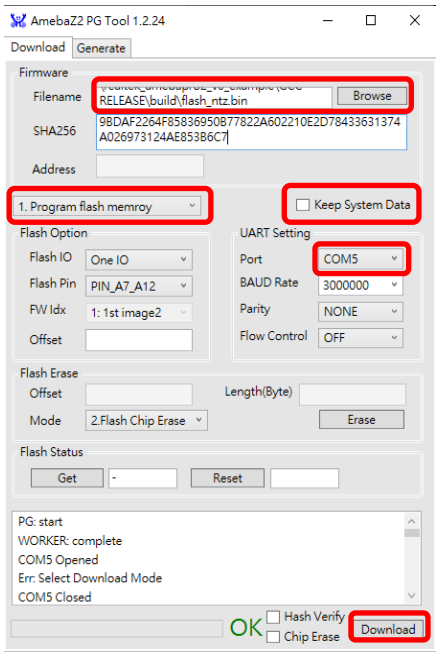
User can download the image to demo board by following steps:

- Trigger AmebaPro2 into **download mode**

  Note: after checking the device into download mode, remember to disconnect the log UART console before using Image Tool to download)

```
== Rtl8735b IoT Platform ==
Chip VID: 0, Ver: 0
ROM Version: v1.0
Test Mode: boot_cfg1=0x0

[test mode PG]
test_mode_img_download
Download Image over UART1[tx=5,rx=4] baud=115200
```

- Open the PG tool



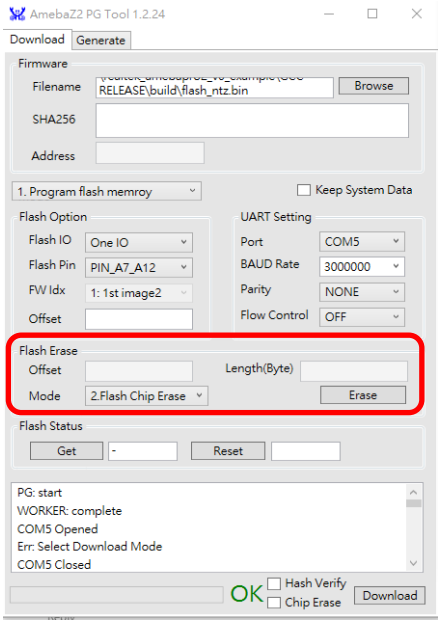
- "**Browse**" to choose the image will be downloaded (flash_ntz.bin)
- Choose "**1. Program flash memory**" and Select the correct CON port
- After the above steps are done, press the Download button and the image will start to be downloaded to Amebapro2 device. (Flash Pin will be set automatically, so we do not need to set.)
- Disable "Keep System Data" to prevent some error.

5.4 Erase Flash of device

For the image tool, it also provide user to erase the flash of the device and it provides two mode:

- Flash Sector Erase: erase the flash from the Offset with a Length of Byte that the user set.

- Flash Chip Erase: erase the whole flash



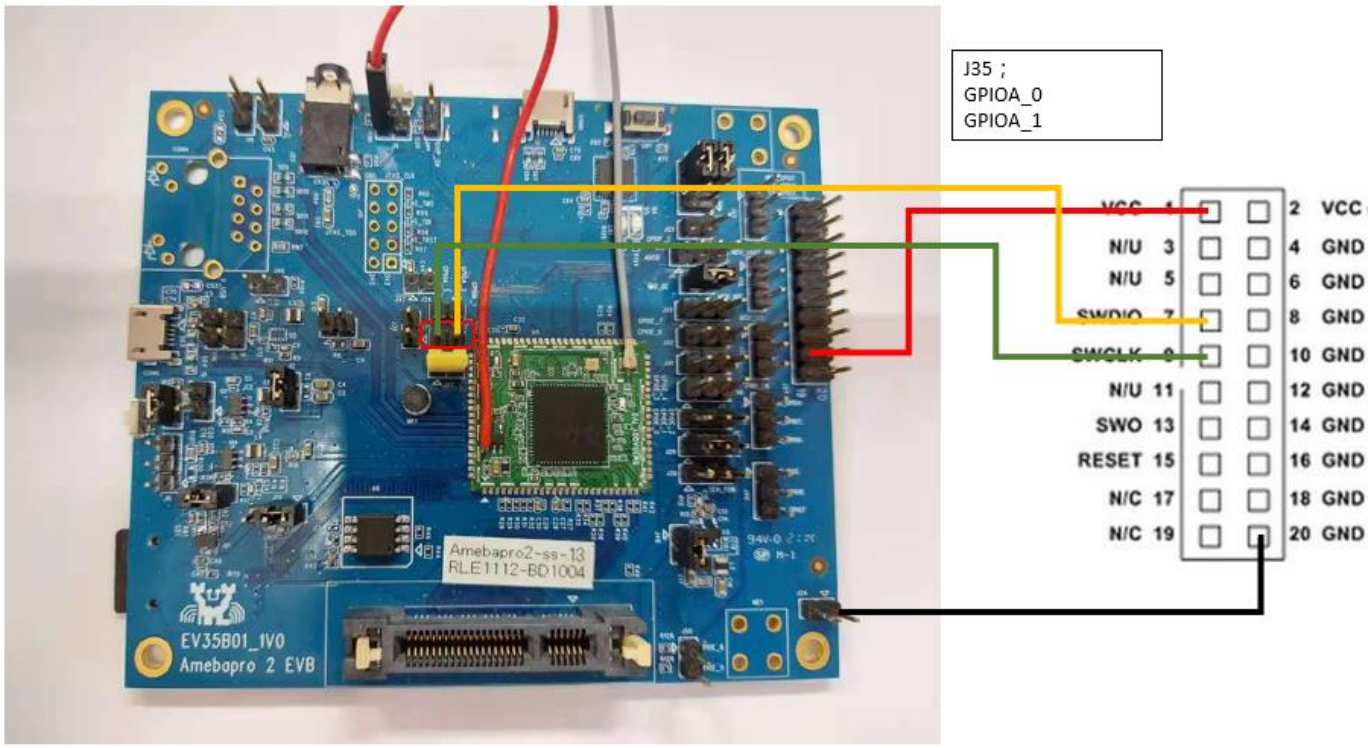
Note: After you press the erase button, it needs time for erasing the flash. Please wait until the PG tool shows the “erase successfully” message.

6 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.

6.1 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 configuration, 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.

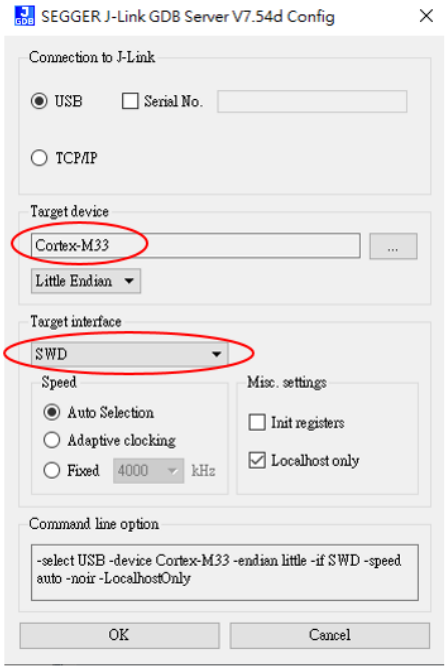


Note: EVB 1V0 module needs to HW rework, ask FAE for details.

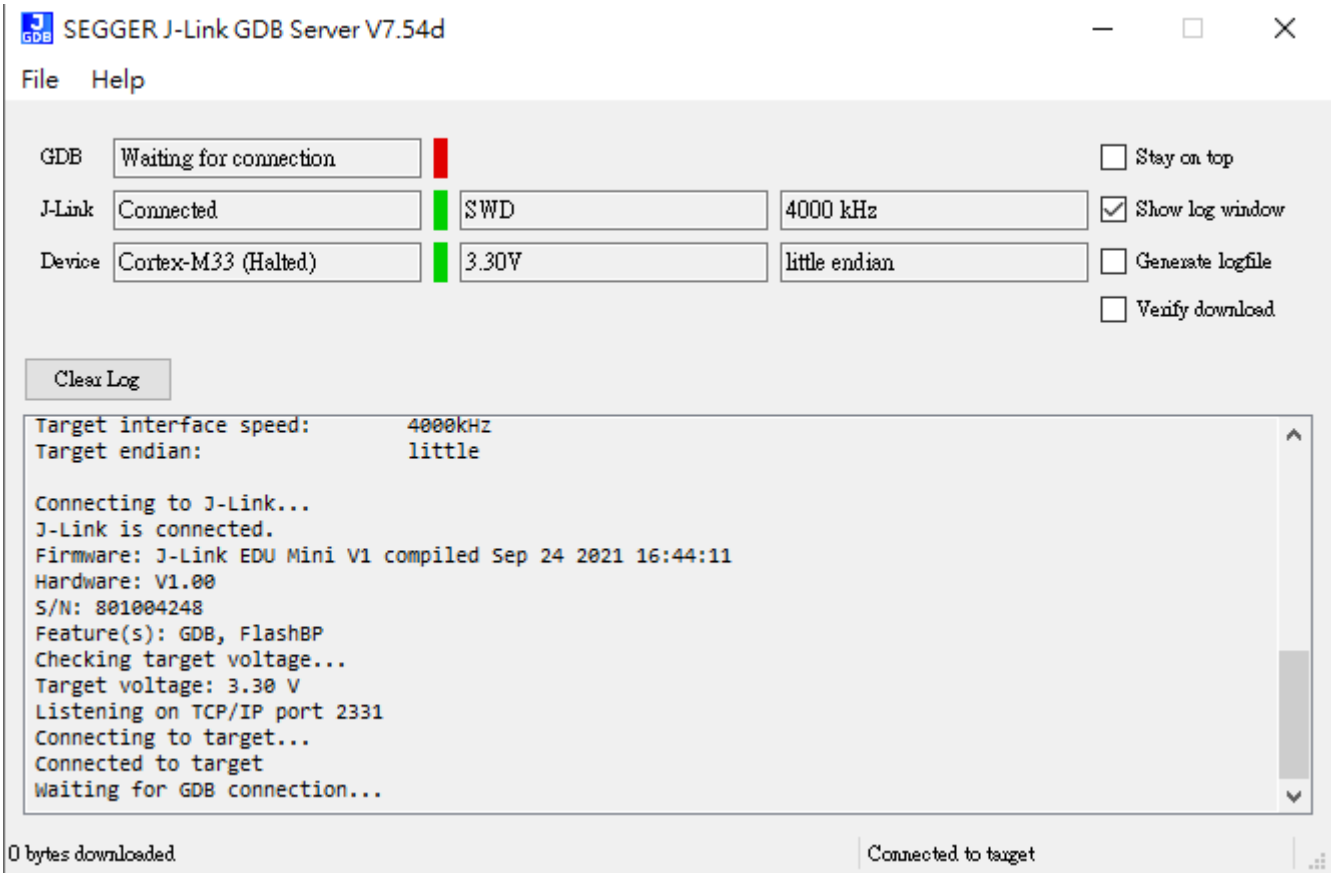
Note: To be able to debugger Ameba-Pro2 which is powered by Cortex-M33, user needs a J-Link debugger with the latest hardware version (Check https://wiki.segger.com/Software_and_Hardware_Features_Overview for details). J-Link EDU with hardware version V10 is used to prepare this document.

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).

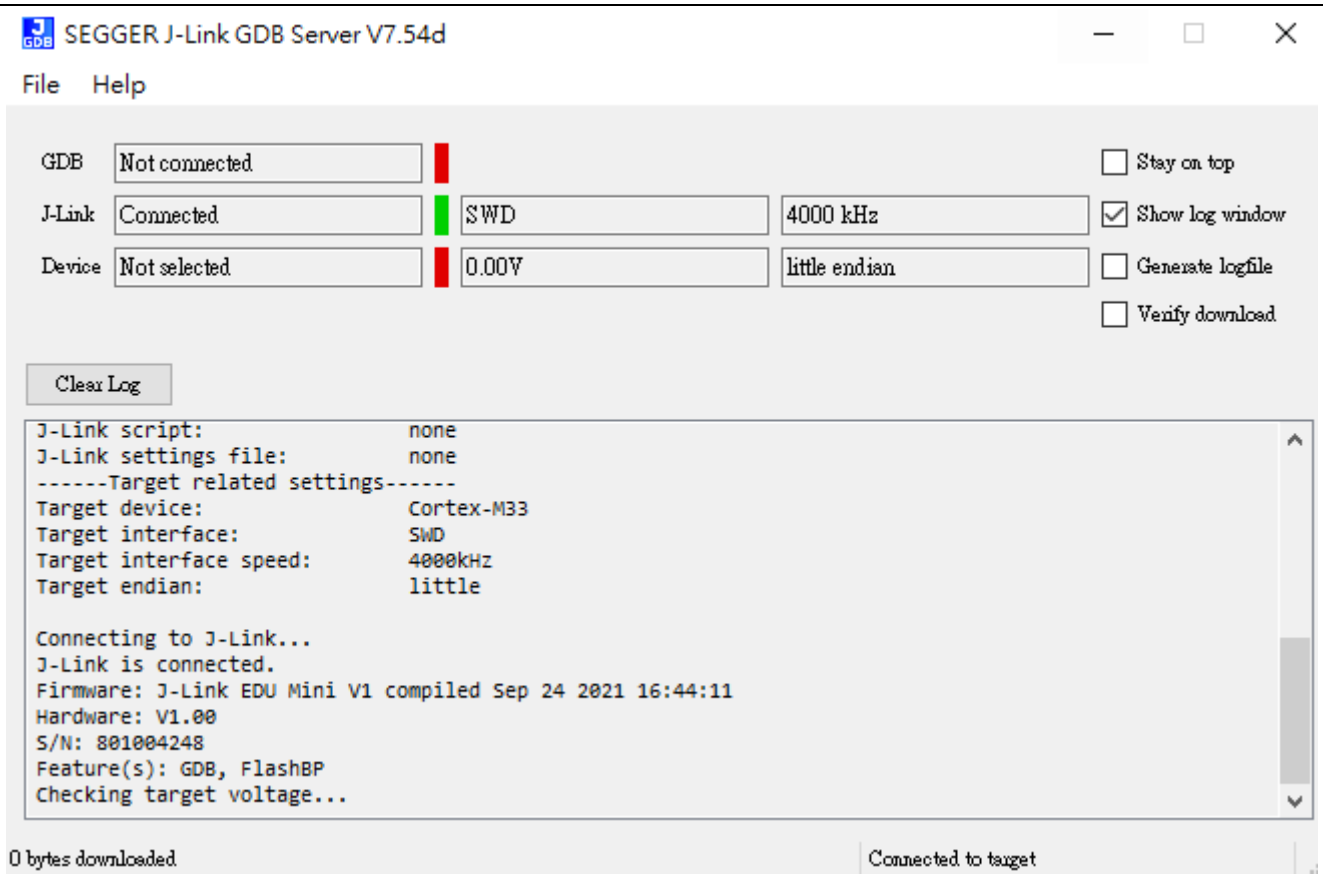
To check whether the connection works fine, user can go to the location of SEGGER J-Link tool and run “JLinkGDBServer.exe”. Choose target device Cortex-M33(for AmebaPro2), and target interface SWD. Click “OK”.



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



If connection fails, J-Link GDB will show:



7 How to use example source code

In this chapter, it will describe how to use the example source code for AmebaPro2.

7.1 Application example source

The AmebaPro2 application's example source codes can be found under folder sdk/component/example. The example entry function is defined as app_example and only one example is exist in the same project. Here are steps to build up the example:

- Choose the example and use "cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DEXAMPLE=<the example folder name> "to generate make file from cmake

- If the example folder exist, you can see "Found <example>" message

```
-- build libraries
-- Build libraries ON
-- Build FPGA OFF
-- Build PXP OFF
-- EXAMPLE = audio_helix_aac
-- Found audio_helix_aac include project
-- build MMF libraries
-- WITH DDR
-- Configuring done
-- Generating done
-- Build files have been written to: D:/20210514_rtsp_h265/sdk_1020/project/real
tek_amebapro2_v0_example/GCC-RELEASE/build
```

- If the example folder not exist, the "<example> Not Found" message will show. Please check the example folder name.

```
-- build libraries
-- Build libraries ON
-- Build FPGA OFF
-- Build PXP OFF
-- EXAMPLE = audio_helix_aac_fake
-- audio_helix_aac_fake Not Found
-- WITH DDR
-- Configuring done
-- Generating done
```

- If successfully building makefile from previous steps, you can use “cmake --build . --target flash” to generate the image of the example project.

Note: In Amebapro2 project, when using the -DEXAMPLE=<the example folder name>, -DVIDEO_EXAMPLE=on and –DDOORBELL_CHIME=on, the flags will be set to off after they import the needed source. It means that next time while you barely type “cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake” without any –D mark in the same folder, the project will not build with the flag (EXAMPLE, VIDEO_EXAMPLE, DOORBELL_CHIME) use in last time.

7.1.1 MMF example source

In sdk/component/example/media_framework, it provide audio-only MMF examples. The example are based on the Multimedia Framework Architecture and the detail can refer to 8 Multimedia Framework Architecture.

7.2 Peripheral example source

The peripheral example source is for helping user utilize peripheral function. The example source is located at the folder SDK/project/realtek_amebapro_v0_example/example_sources and basically provide mian.c and readme file. The mian.c file contains the usage of peripheral function and user should replace it with the original main.c (in SDK/project/realtek_amebapro_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.

7.3 Wi-Fi example source

7.3.1 Use AT command to connect WLAN

For user’s 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 be referenced in AN0025 Realtek at command.pdf.

8 Multimedia Framework Architecture

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

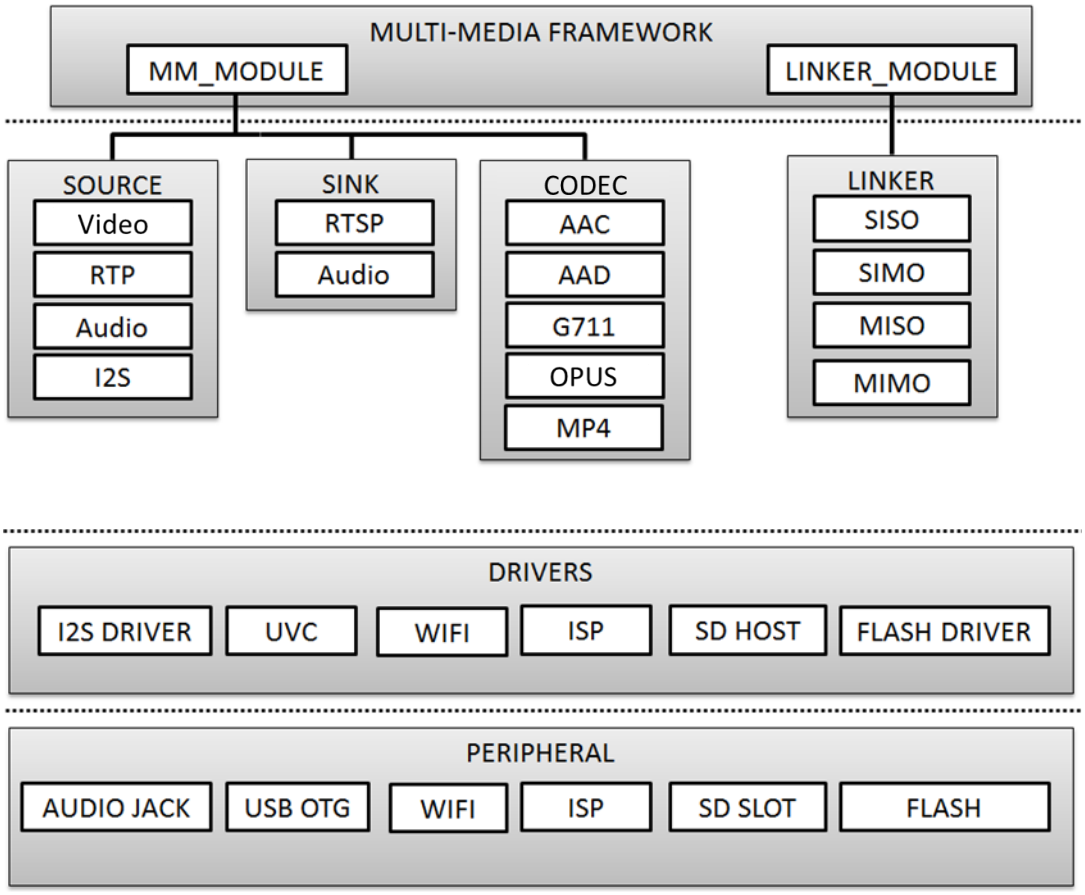
8.1 Architecture

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

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

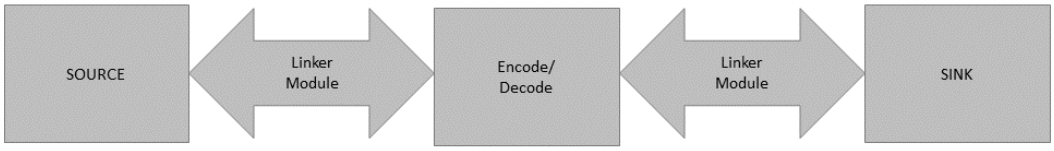


In order to use the MMFv2, here are some aspects must to be followed.

- 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**.



8.1.1 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 implement 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;
```

8.1.1.1 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 create queue item that will be send to input and output queue, only will be used when setting MM_CMD_INIT_QUEUE_ITEMS to MMQI_FLAG_STATIC.

- del_item

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

- rsz_item

  Pointer to the function decrease memory pool size, only will 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 option can be used MM_MASK_SRC, MM_MASK_DSP and MM_MASK_SINK.

- name

  Pointer to the module name.

8.1.1.2 mm_module_ctrl

Here lists some command defined 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.

8.1.2 Context

MMFv2 context supply 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 default support these modules (video, AAC_encoder, AAC_decoder, audio, g711, opus, mp4, rtp, rtsp). 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.

8.1.3 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.

8.1.3.1 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 stops SISO execution and free space of mm_siso_t entity.

- siso_ctrl

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

8.1.3.2 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 confer 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 declares 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 calls simo_stop() to stop SIMO execution and free space.

- simo_ctrl

Pointer to the function 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.

8.1.3.3 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 confer 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 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 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 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.

8.1.3.4 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 use 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 calls mimo_stop() to stop the mimo module and free space.

- mimo_ctrl

  Pointer to the function 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 that, 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.

8.2 MM_Module Type and Module Parameter

8.2.1 Video

The video module integrates the process from sensor use, isp to video encoder.
Here shows the context of the video module.

```
typedef struct video_ctx_s {
        void *parent;

        hal_video_adapter_t *v_adp;

        video_params_t params;
        int (*snapshot_cb)(uint32_t, uint32_t);

} 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.

8.2.1.1 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, included 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    direct_output;
        uint32_t    use_static_addr;
} 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). Note that the video resolution must be less than the sensor's maximum width or height.
- 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.

8.2.1.2 Video module rate control (RC) adjustment

CBR:

- Fixed bit rate
- QP range default value is [0, 0, 51].
- Can be adjusted [minQp, minIQp, maxQp] by control of encode_rc_parm_t in module_video.h, use API video_control (minIQp is I frame QP.)

VBR:

- When the screen is still, it will automatically adjust to 1/2 bit rate. When the screen changes, it will exceed bit rate setting of the video module parameters (bps). The excess amplitude is controlled by maxQp. The larger the maxQp, the smaller the bit rate is.
- QP range default value is [20, 20, 45].

```
typedef struct encode_rc_parm_s {
        unsigned int    rcMode;
        unsigned int    iQp;            // for fixed QP
        unsigned int    pQp;            // for fixed QP
        unsigned int    minQp;          // for CBR/VBR
        unsigned int    minIQp;           // for CBR/VBR
        unsigned int    maxQp;          // for CBR/VBR
} encode_rc_parm_t;
```

Use **CMD_VIDEO_SET_RCPARAM** to set up the rate control parameters.

- minQp: Minimum QP.
- minIQp: Minimum QP of the I frame, It is recommended that the value of this parameter be equal to minQp in normal scenarios.
- maxQp: Maximum QP.

Improving Image Quality:

- Set the maxQp: Setting the maximum QP helps effectively protect the image quality, but bit rate overshooting is prone to occur.
- Set the minQp: This parameter is used to control the highest image quality. The QP is not decreased any longer after being adjusted to this value, which may cause bit rate insufficiency. This parameter is intended to reduce the bit rate in simple still scenarios.

8.2.2 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}
};
```

8.2.3 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.

8.2.4 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 be the same source to decode correctly.
- channel: Need to be the same source to decode correctly.

8.2.5 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

        int             channel;        // 1: Mono
        int             enable_aec;             // 0: off  1: on
        int             enable_ns;              // 0: off, 1: out 2: in 3: in/out ns
        int             enable_agc;             // 0: off, 1: out 2: in 3: in/out agc
        int             enable_vad;             // 0: off  1: input vad
        int             mix_mode;               // 0
} 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: Microphone gain value. Support 0, 20, 30, 40 DB.
- channel: The number of channel is supported. Currently, support mono so set it to 1.
- enable_aec, enable_vad: switch of the AEC and VAD while audio RX process. Set 1 to enable them while 0 to disable the function.
- enable_ns, enable_agc: switch of ANS and AGC of audio TX and RX. 0 to turn off them, 1 only turn on the process in TX, while 2 to enable them on RX and finally if you want to enable algorithm of TX and RX, set it to 3.
- mix_mode: enable mix mode of all the logic input, the default setting is 0.

Note: See the more ASP setting detail in 14.2 Open ASP algorithm

8.2.6 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.

8.2.7 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).

8.2.8 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.

8.2.9 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 be the same source 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.

8.2.10 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 seem 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 (sound only).
- record_file_num: Set the number of file 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.

8.2.11 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.

8.2.12 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.

8.2.13 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 will be started to play and it will also be used to keep the play location while the array module process.

8.3 Using the MMF example

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

8.3.1 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.

8.3.1.1 Requisites and Setup

Pre-requisites:

- AmebaPro2 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 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.

8.3.1.2 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.

- **Video only examples**:

Example	Description	Result

mmf2_video_example_v1_init	CH1 Video -> H264/HEVC -> RTSP	Transfer AmebaPro2's H264/HEVC video stream over the network. Video default format: 720P 30FPS.
mmf2_video_example_v2_init	CH2 Video -> H264/HEVC -> 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/HEVC -> 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/HEVC) -> 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/HEVC) -> 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/HEVC -> 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/HEVC) -> MP4 (SD card)	AmebaPro2 will record H264/HEVC stream array to the SD card for 30 second. Video default format: 25FPS.

- **Video + Audio examples**:

Example	Description	Result
mmf2_video_example_av_init	1 Video (H264/HEVC) 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/HEVC) 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/HEVC) 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/HEVC) 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/HEVC) -> 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/HEVC) -> RTSP (V1+A) Video (H264/HEVC) -> 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/HEVC) -> MP4 (V1+A) Video (H264/HEVC) -> 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/HEVC) -> 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/HEVC) -> 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.

8.3.1.3 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

8.3.1.4 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

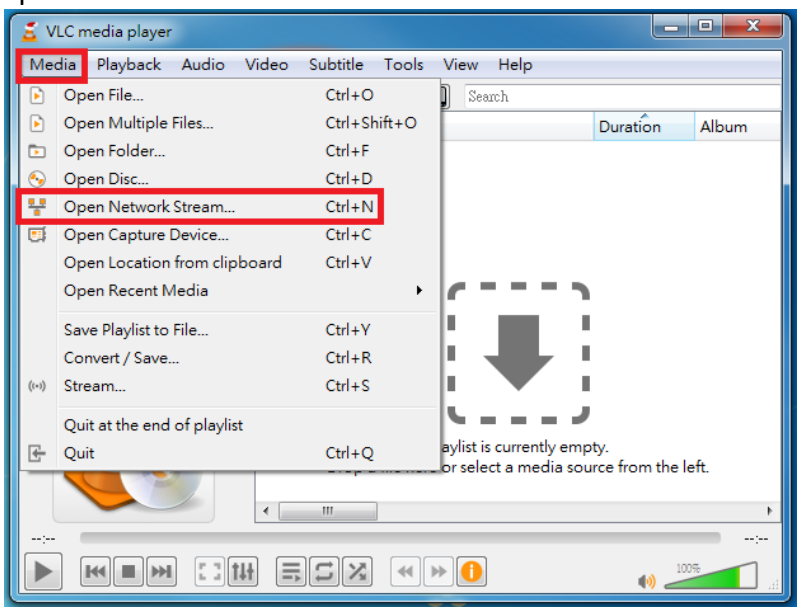
8.3.2 VLC media player settings

For RTSP examples, you can use VLC media player to receive or transmit the stream. Download VLC media

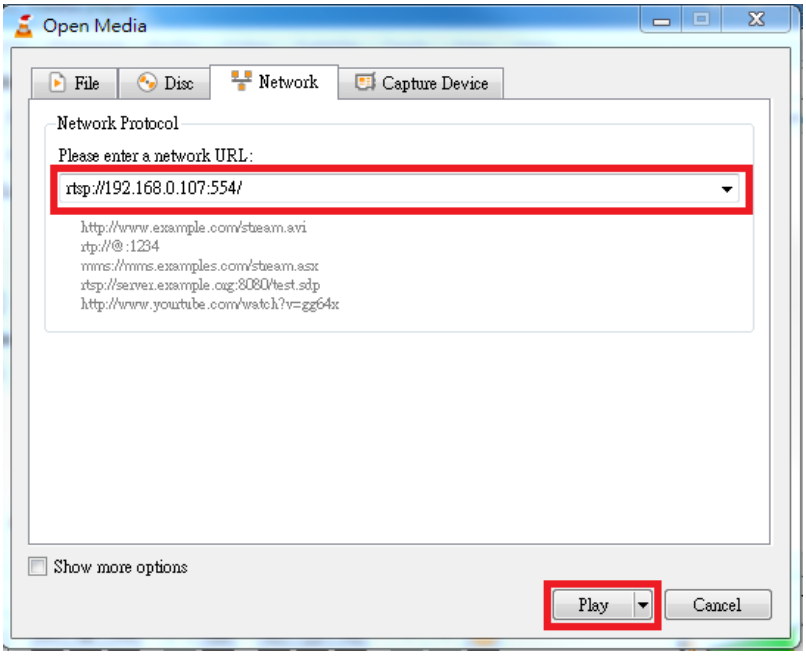
player from website https://www.videolan.org/.

8.3.2.1 Stream audio/video from AmebaPro2 to VLC player

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

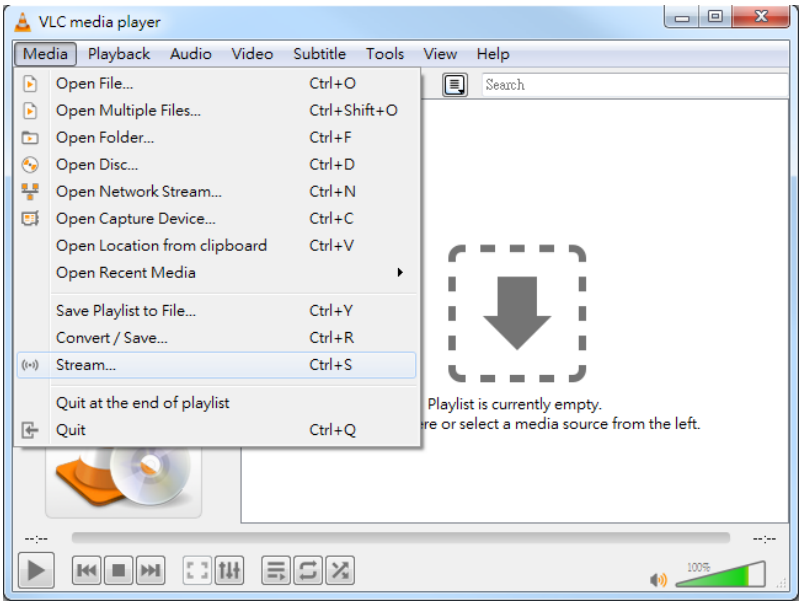


- Enter “rtsp://xxx.xxx.xxx.xxx:yyy/”, where xxx.xxx.xxx.xxxis the Ameba IP addressa and yyy is the RTSP server port (default is 554), and click “Play”.

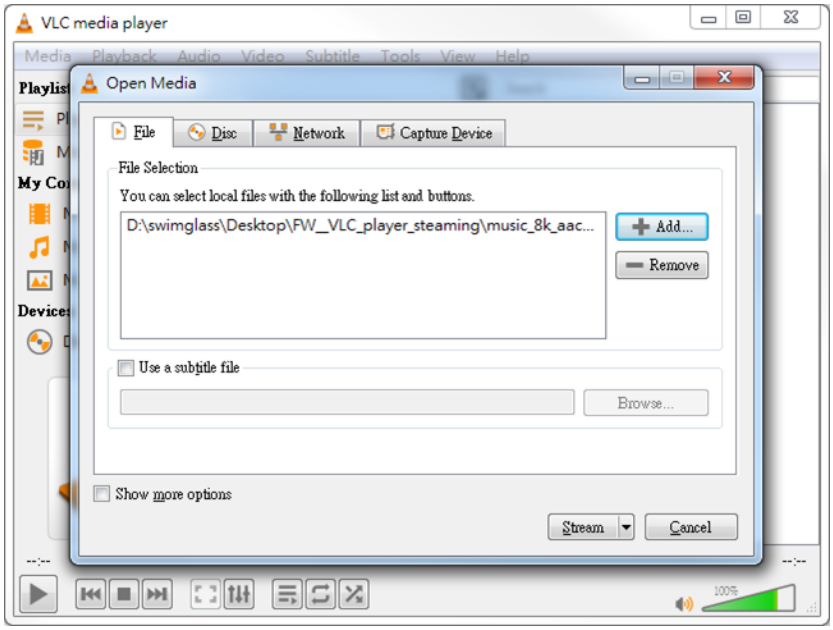


8.3.2.2 Stream audio from VLC player to AmebaPro2

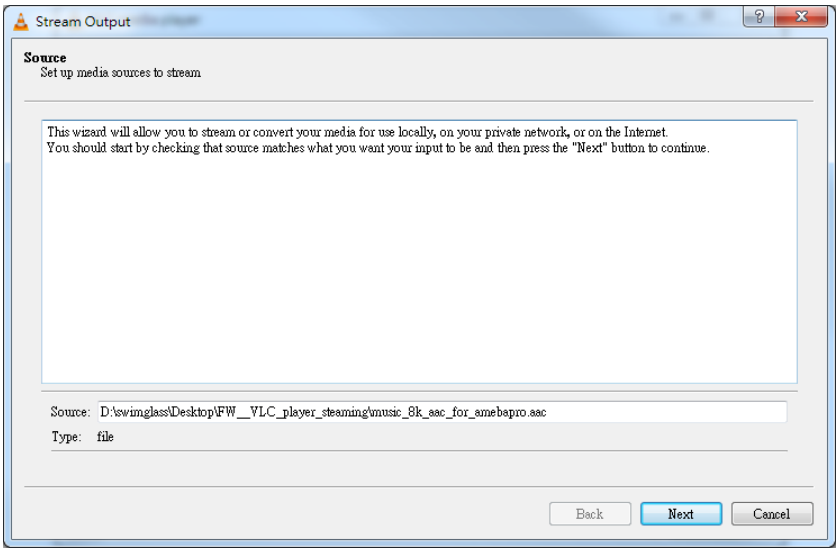
- Click “Media” -> “Stream”.



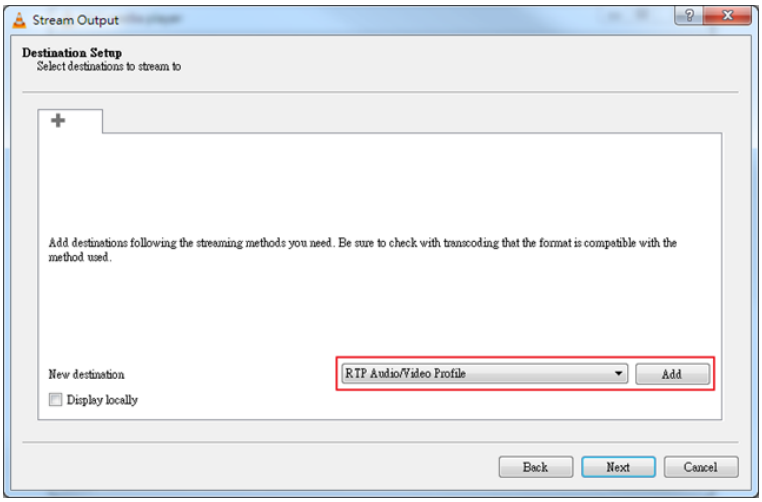
- Select “File”, choose the file by “Add” and finally chick the “Stream”. (If the startup example is RTP -> AAD -> AUDIO please select the audio file with the file name .aac (The file format must be the same as the AAC decoder setting, the default is mono, sampling rate = 8k Hz). If the startup example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



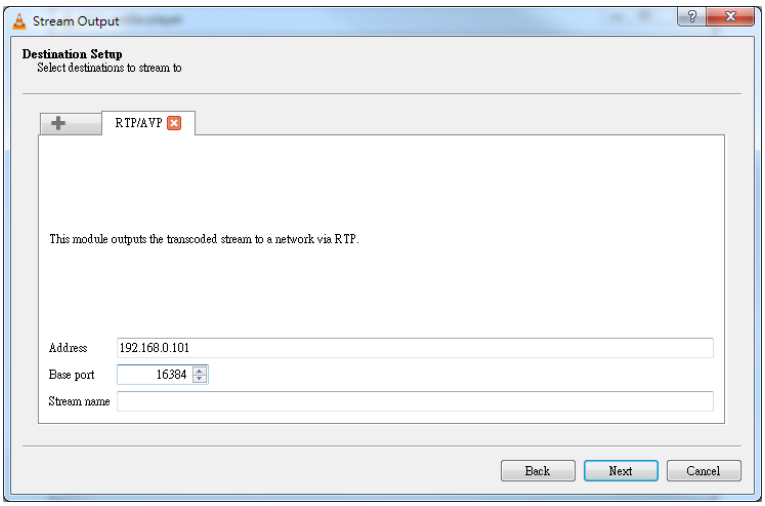
- You will see your select file after push “Stream”. Check it and click “Next”.



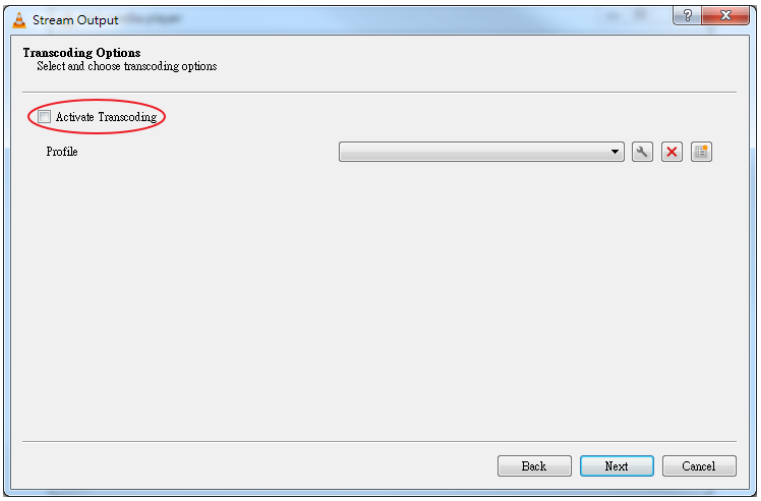
- Select “RTP Audio/Video Profile”, and click “Add”.

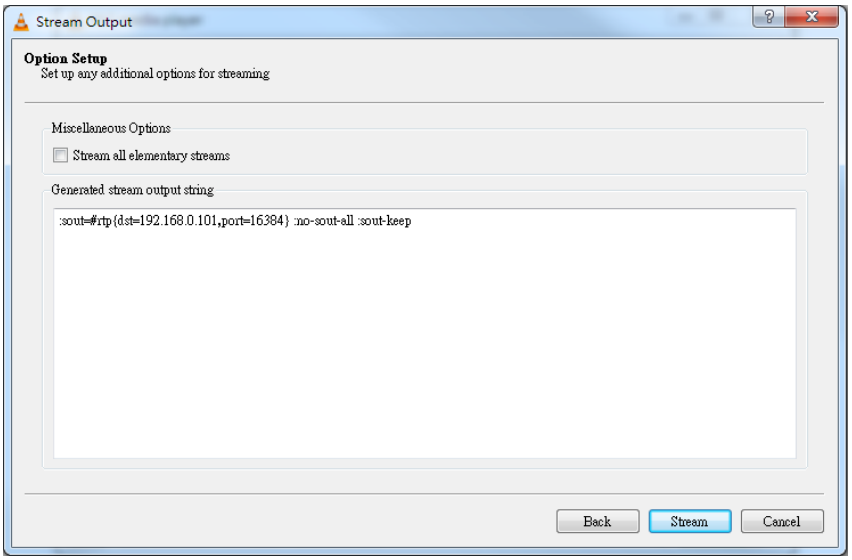


- Enter AmebaPro's IP Address in “Address” field, with “Base port” set to 16384, and click “Next”.



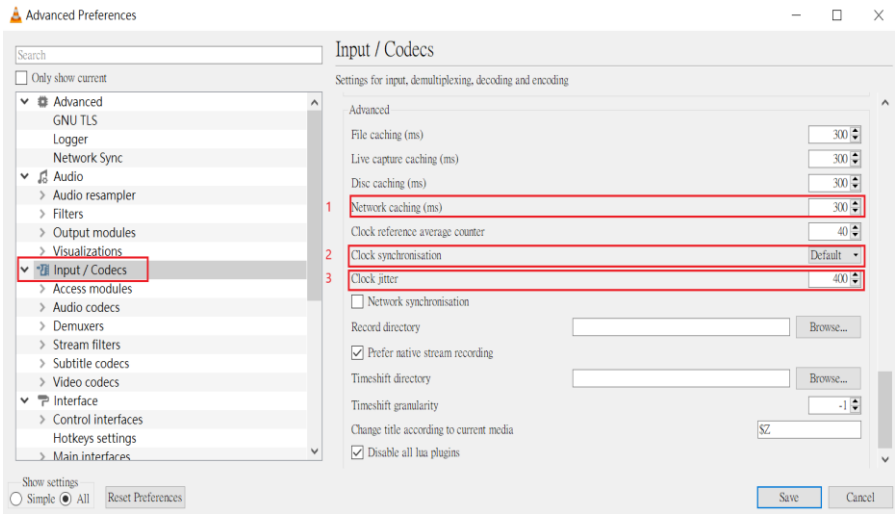
- Confirm “Activate Transcoding” is unchecked, and click “Next” -> “Stream”. Then the sound can be heard on AmebaPro2 3.5mm audio jack.



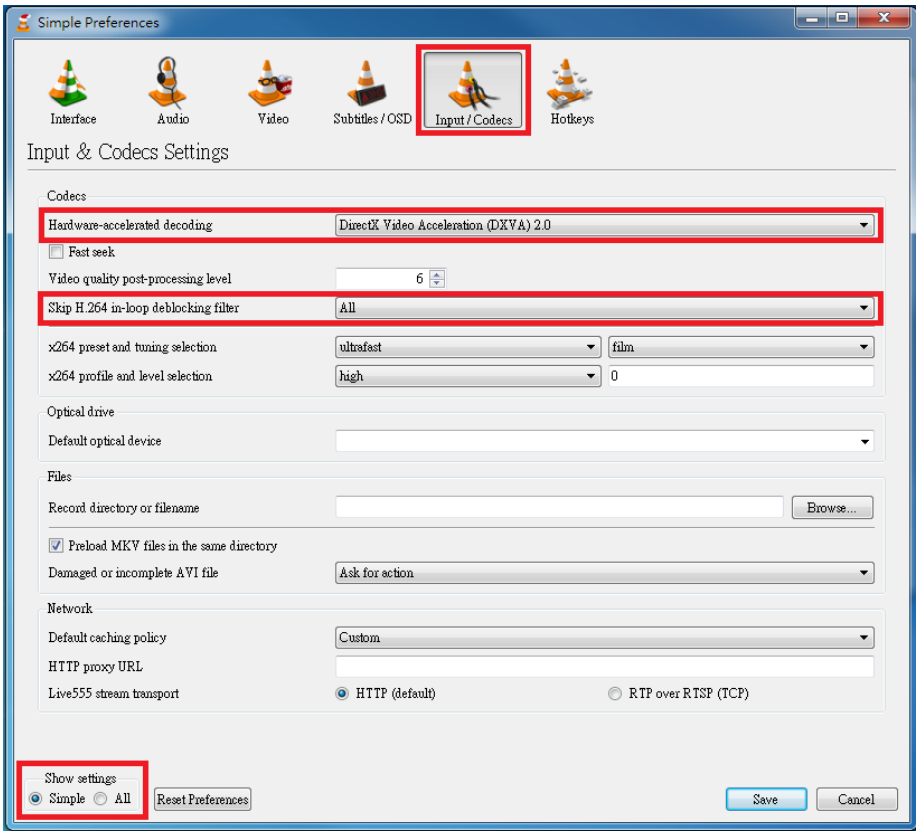


8.3.2.3 Adjust latency (buffer) related settings

- 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.

8.3.3 Echo Cancellation

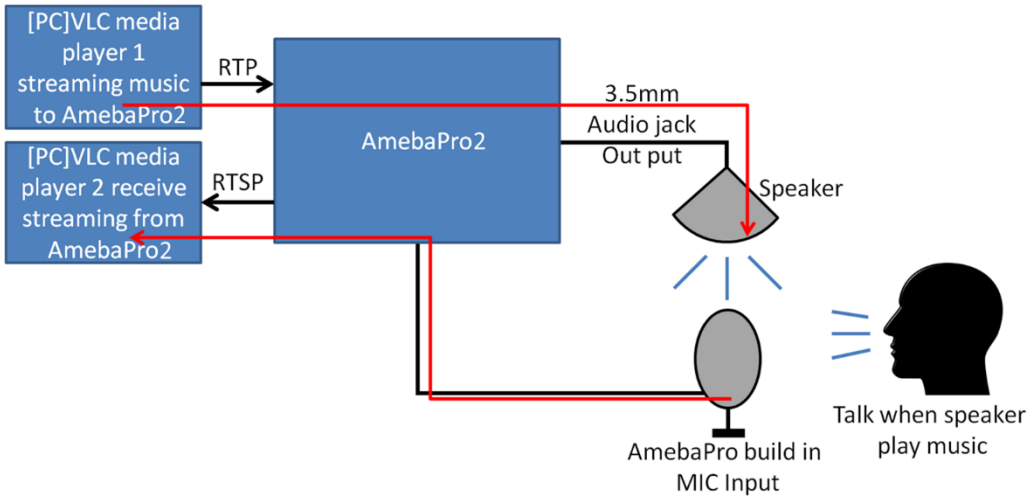
Echo cancellation is default provided in the audio part of MMFv2. To test whether the echo cancellation function is correct, use VLC media player to verify it on the computer.

Usage Note (Refer to Audio optimization):

- 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:

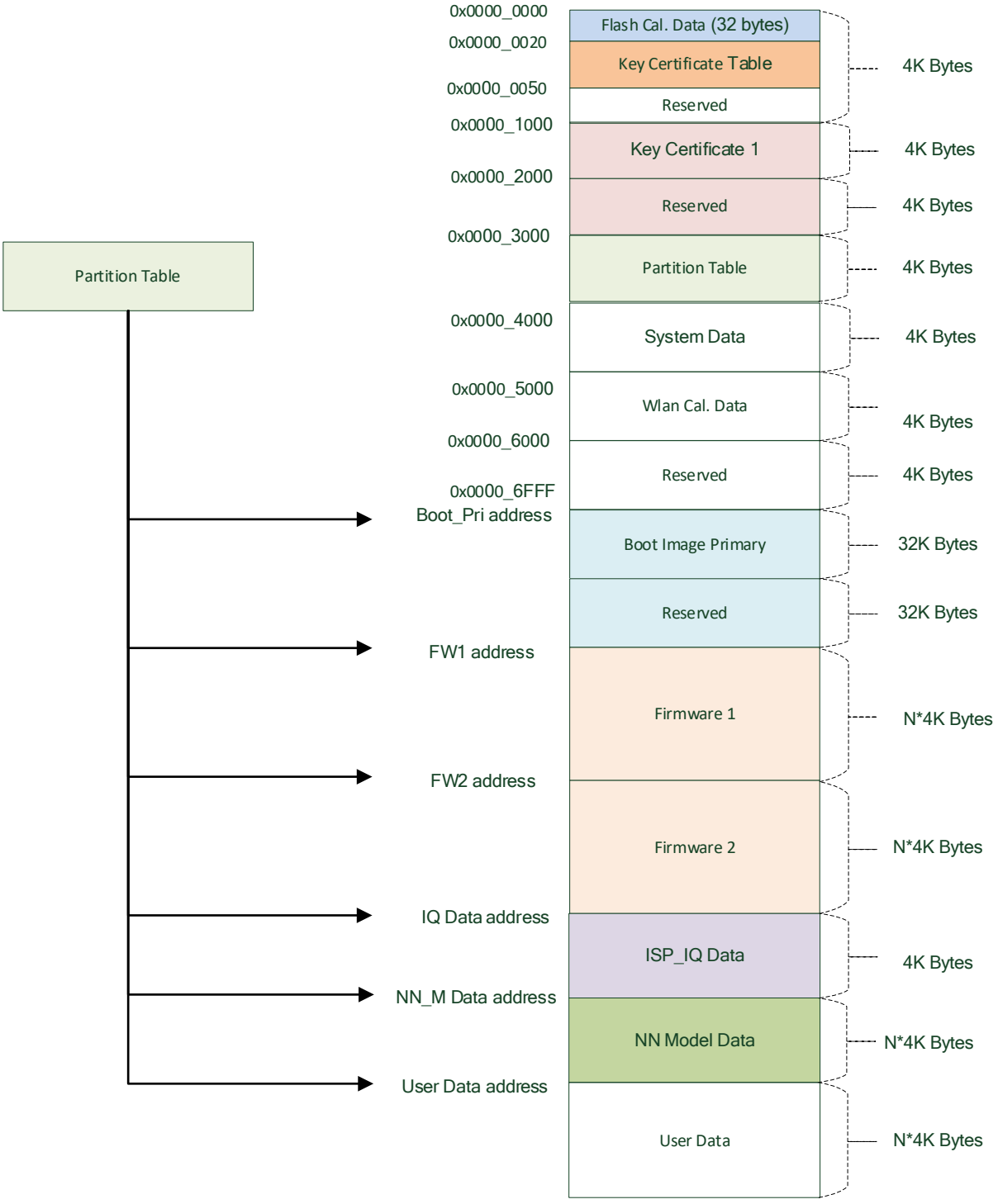
- Use VLC media player on the PC to stream voice signal to AmebaPro2.
- Put AmebaPro2 speaker next to AmebaPro2 built-in Mic and speak at the same time.
- Then pass the received sound to the VLC media player on the PC via AmebaPro2 to see if the sound in step 1 is small enough or even disappear.



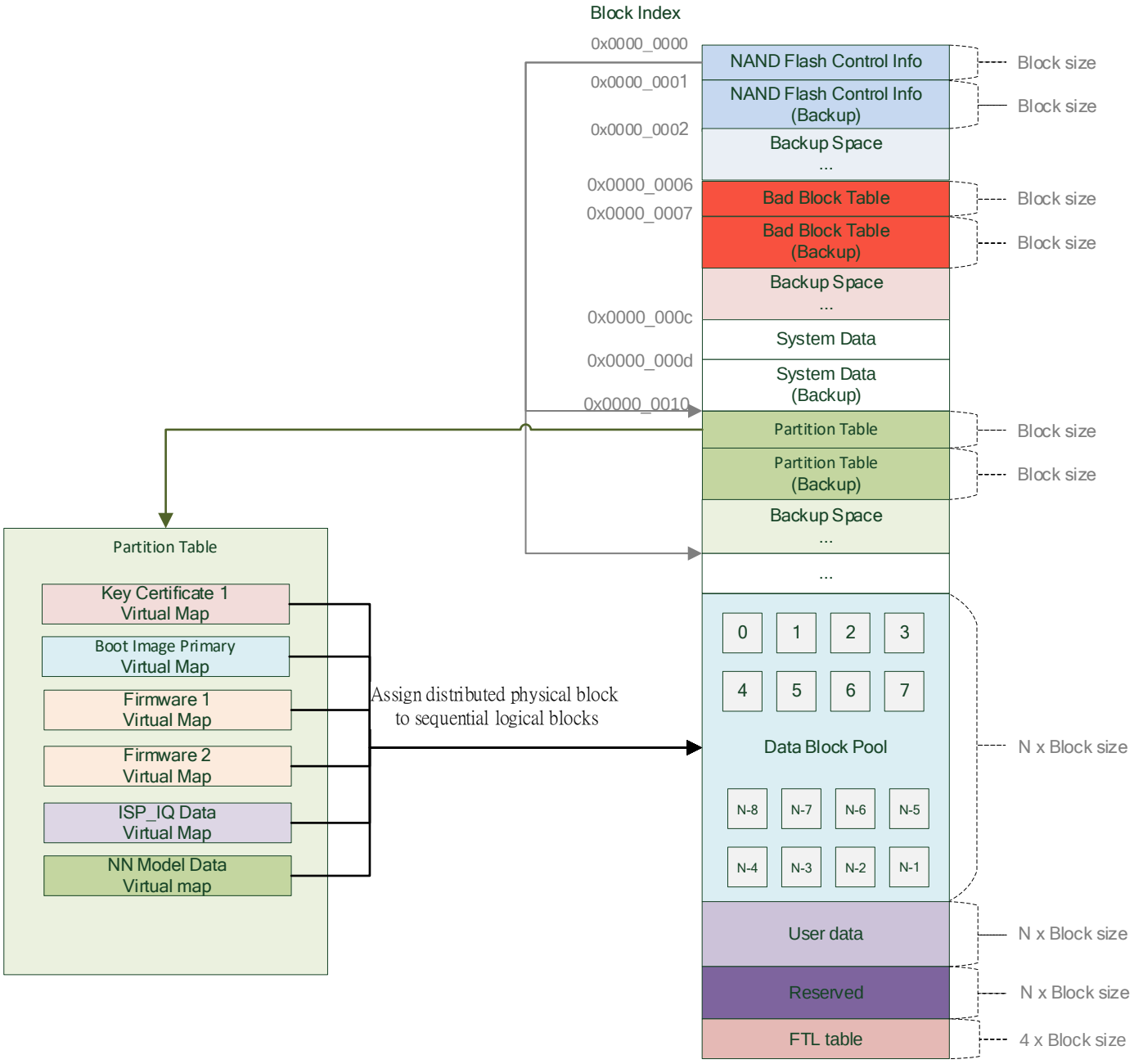
- There is an AT command to store MIC/AEC signal and transfer to PC to analyze AEC result. The built-in command is ATAD. User must execute “ATAD=start,10” before farend signal starting sending (by VLC) and the number 10 inside command is mean recording 10 seconds. After 10 seconds pass, Use “ATAD=dump” and “ATAD=dump,aec” to transfer MIC data and AEC data to PC via Log UART port by latest version PG Tools. The transfer data will be saved as two .wav files in PG Tools folder. Compare those two .wav files to check AEC result by visualized audio tool (like audacity).

9 Flash Layout

9.1 NOR Flash Layout overview



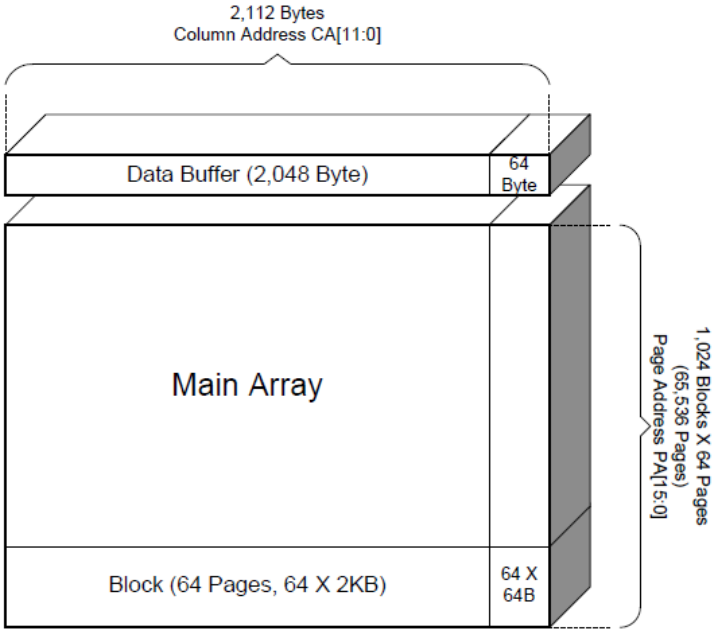
9.2 NAND Flash Layout overview

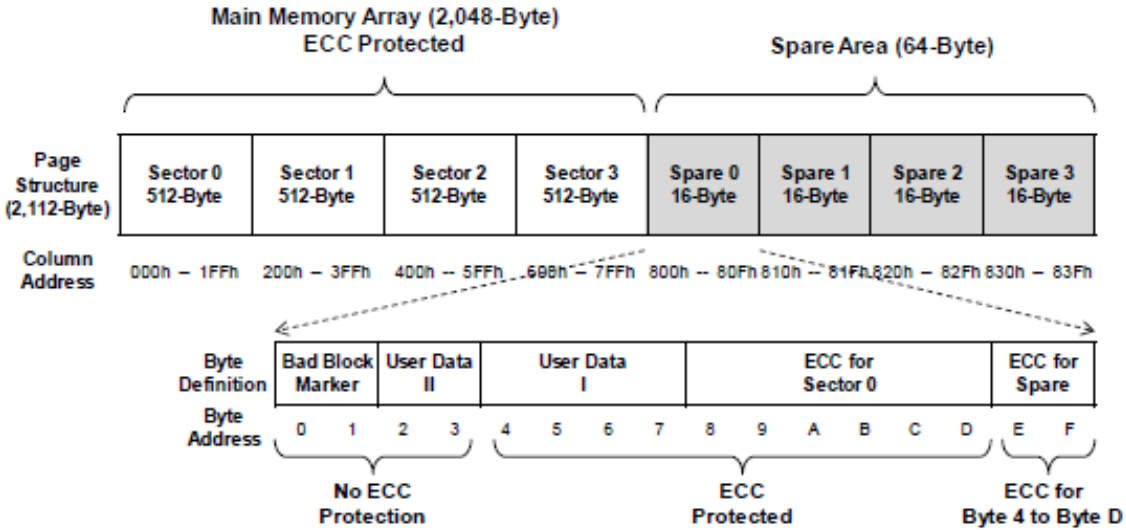


9.3 NAND Flash

The NAND flash(GD5F1GQ) 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(GD5F1GQ) has 1,024 erasable blocks.

- 1Gb SLC NAND Flash: 1G-bit / 128M-byte
- On chip ECC for memory array





9.3.1 NAND Flash mbed API

NAND Flash mbed API is used to access Flash physical location.
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);
```

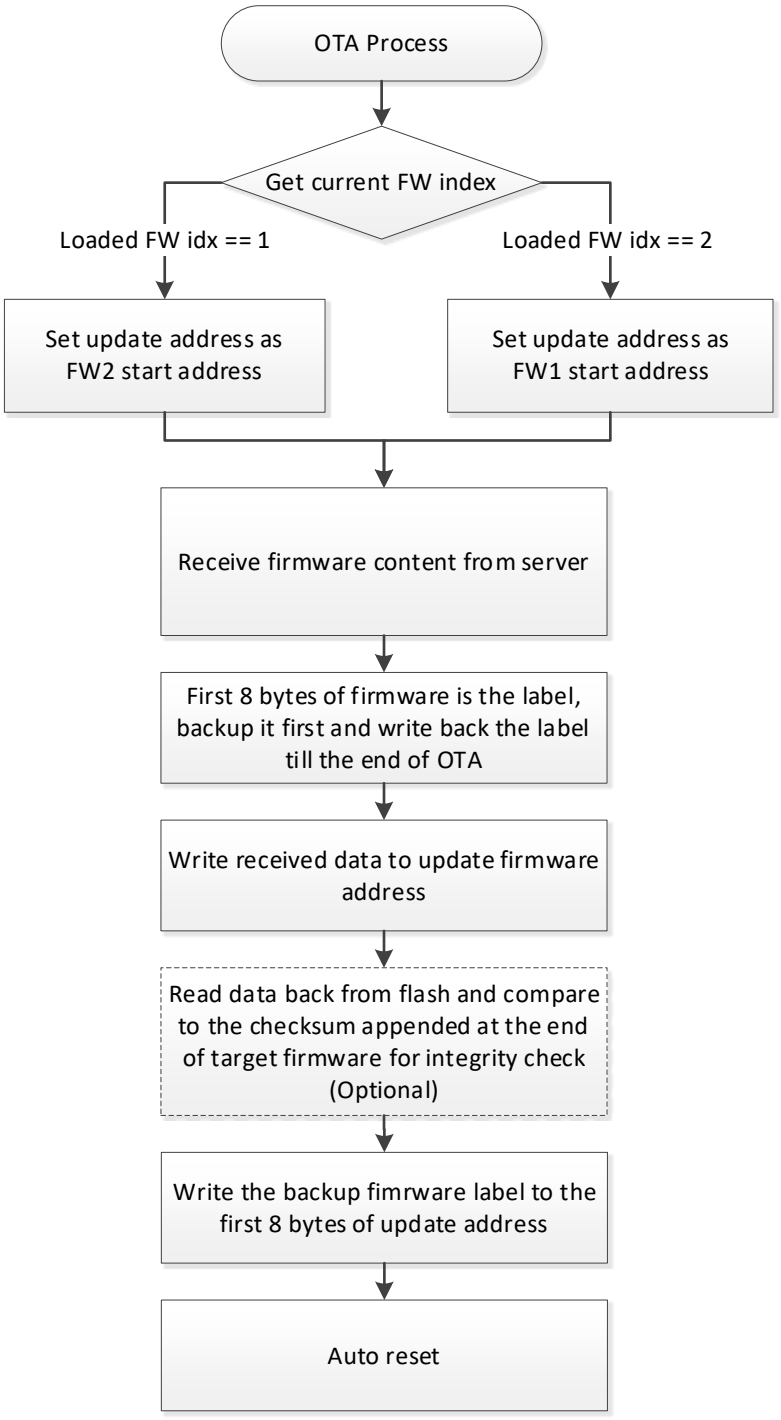
9.3.1.1 NAND flash mbed example

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.

10 OTA

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

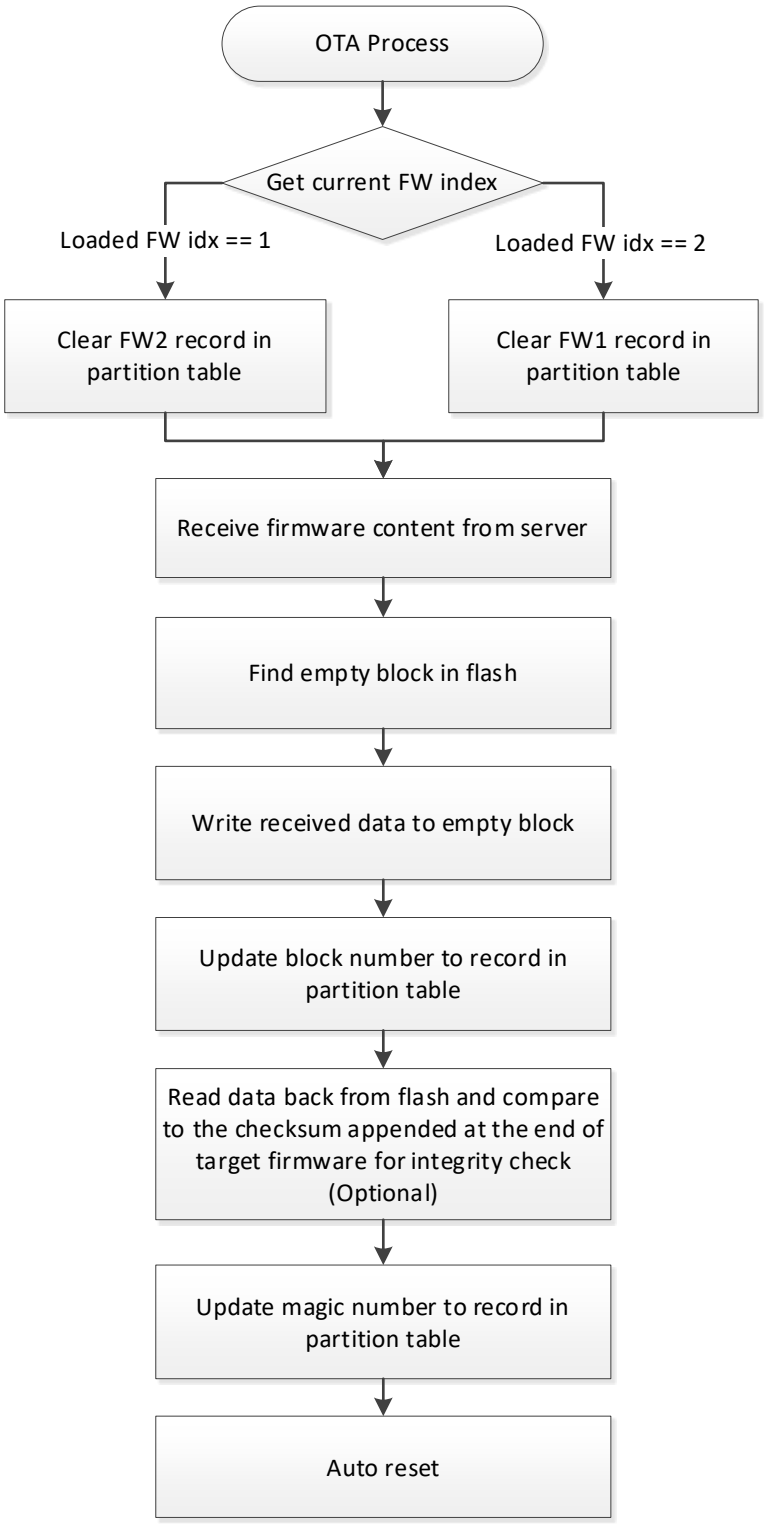
10.1 OTA Operation Flow for NOR Flash



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 0xff, which is invalid. The backup label needs to be written

back at the end of OTA process to prevent device booting from incomplete firmware.

10.2 OTA Operation 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.

10.3 OTA Checksum Mechanism

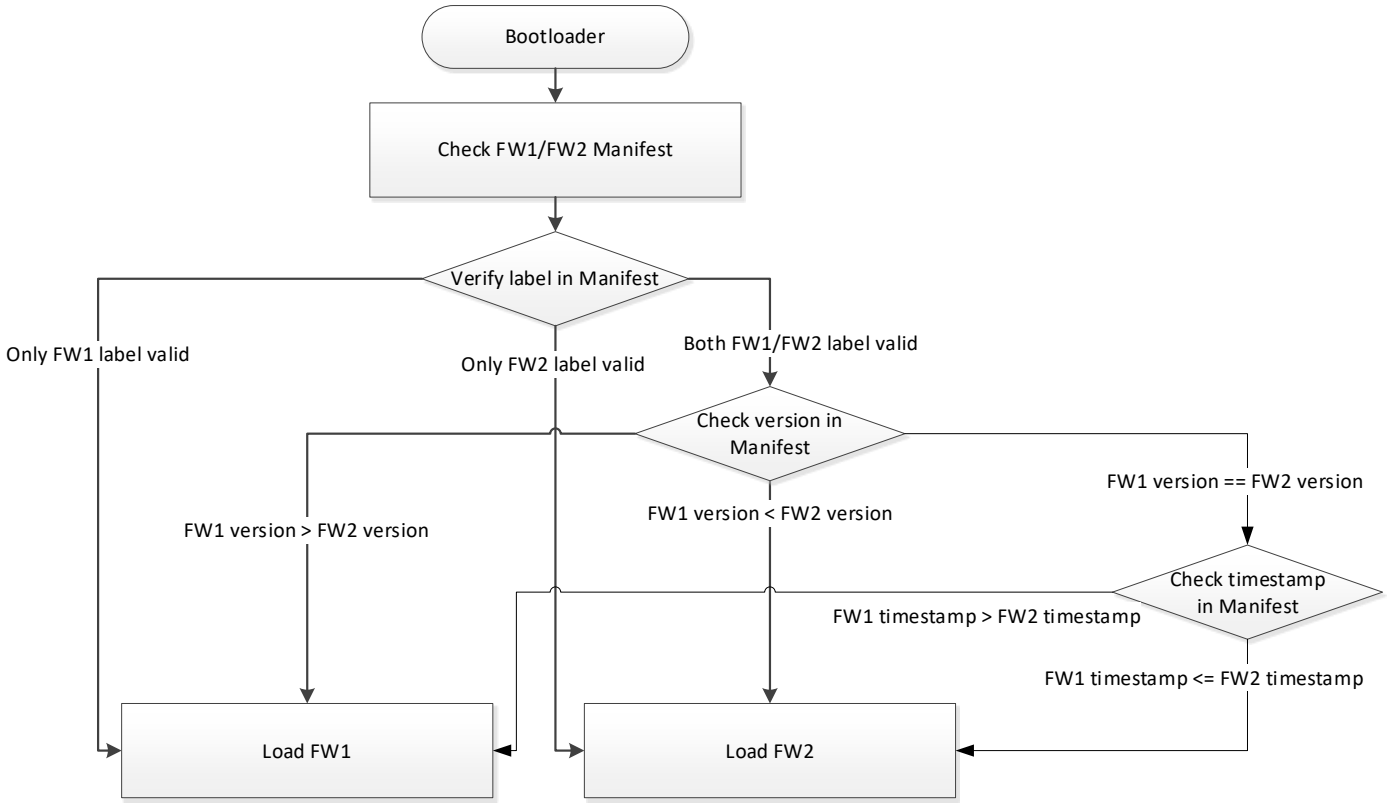
The first 8 bytes firmware label for NOR flash OTA process and the magic number of record in partition table for NAND flash OTA process are used to notice the bootloader the overall OTA process is done without any network disconnection or re-boot during the OTA. However, firmware label or magic number 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
```

10.4 Boot Process Flow

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



10.5 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
    },
```

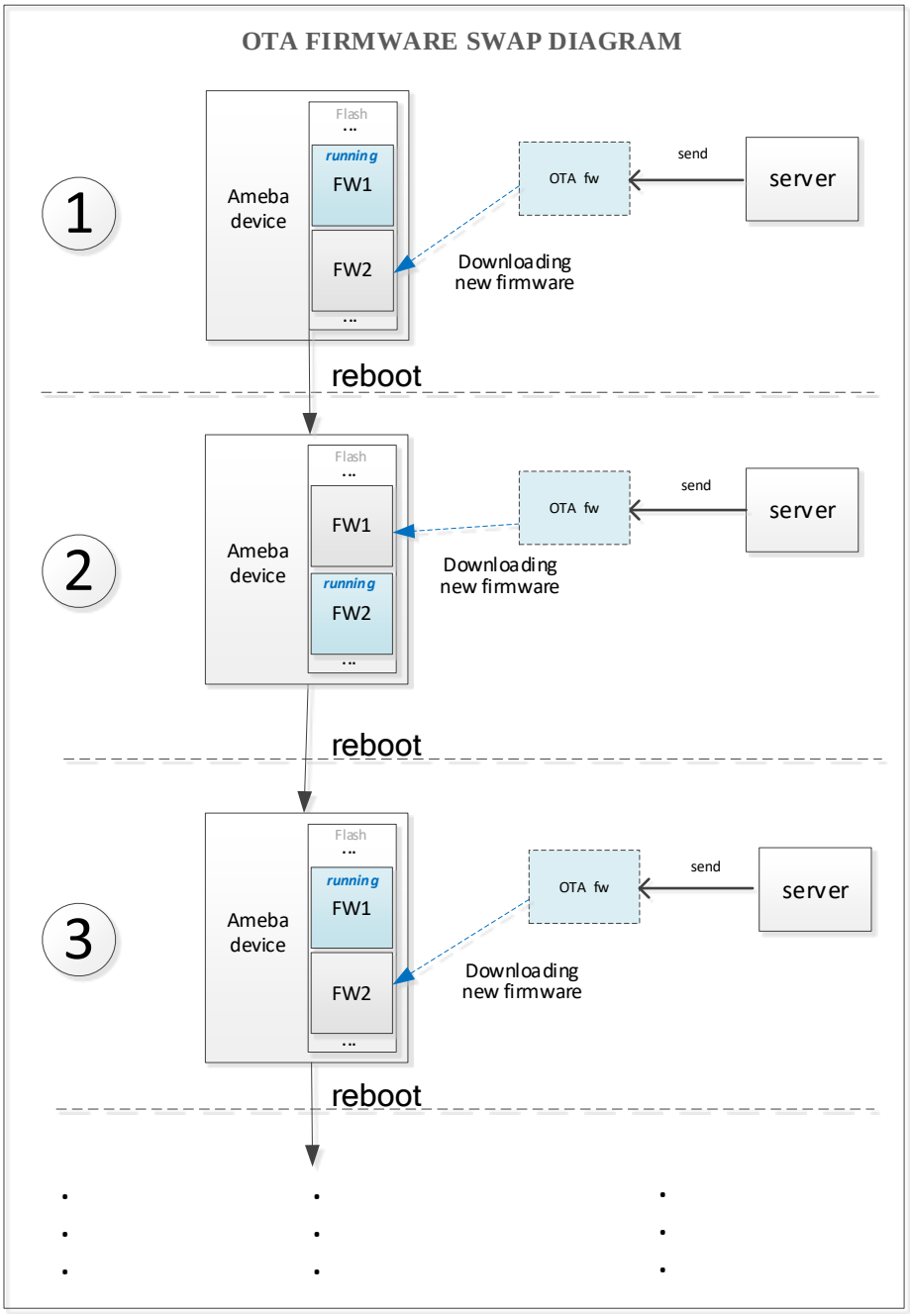
For NOR flash, OTA firmware is written to the partition start address in flash, and OTA firmware size is checked with the partition length. For NAND flash, OTA firmware is written to empty blocks distributed in flash, and OTA firmware size is checked with the partition length.

10.6 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.

10.6.1 OTA Firmware Swap Behavior

When device executes OTA procedure, it would update anther 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.



10.6.2 Version and Timestamp

AmebaPro2 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.

In 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":"FEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"},
```

```
        {"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 'EFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' is zero in bit 4, and version of 'FFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' 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 timestamp is an 8bytes value in little endian order. The timestamp which presents image build time will be automatically generated when image build.

10.7 Implement OTA over Wi-Fi

10.7.1 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.

10.7.1.1 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.

10.7.1.2 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'.

```
c():checksum 0x769b23a
Listening on port (8082) to send ota.bin (1306628 bytes)

Waiting for client ...
```

10.7.1.3 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.

```
ATWO=192.168.13.162[8082]
[ATWO]: _AT_WLAN_OTA_UPDATE_

[MEM] After do cmd, available heap 55369888



#

[update_ota_local_task] Update task start
[update_ota_local_task] Read info first
[update_ota_local_task] info 12 bytes
```

```
[update_ota_local_task] tx file size 0x13f004
[update_ota_local_task] Current firmware index is 1

[update_ota_local_task] Start to read data 1306628 bytes
..
[ota_flash_NOR] target_fw_addr=0x400000, target_fw_len=0x300000
.....
.....
.....
.....
flash checksum 0x 769b08d attached checksum 0x 769b08d
[ota_flash_NOR] Append FW label
[ota_flash_NOR] FW label:
 52 54 4C 38 37 33 35 42

Read data finished

[update_ota_local_task] Update task exit
[update_ota_local_task] Ready to reboot
```

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

10.7.2 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.

10.7.2.1 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.

10.7.2.2 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'.

```
<Local HTTP Download Server>
Listening on port (8082) to send ota.bin (1310724 bytes)

Waiting for client ...
```

10.7.2.3 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.

```
[http_update_ota] Download new firmware begin, total size : 1310724

[http_update_ota] Current firmware index is 1
..
[ota_flash_NOR] target_fw_addr=0x400000, target_fw_len=0x300000
.....
.....
.....
.....
flash checksum 0x 76fa723 attached checksum 0x 76fa723
[ota_flash_NOR] Append FW label
[ota_flash_NOR] FW label:
 52 54 4C 38 37 33 35 42

[http_update_ota] Download new firmware 1310724 bytes completed

[http_update_ota_task] Update task exit
[http_update_ota_task] Ready to reboot
```

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

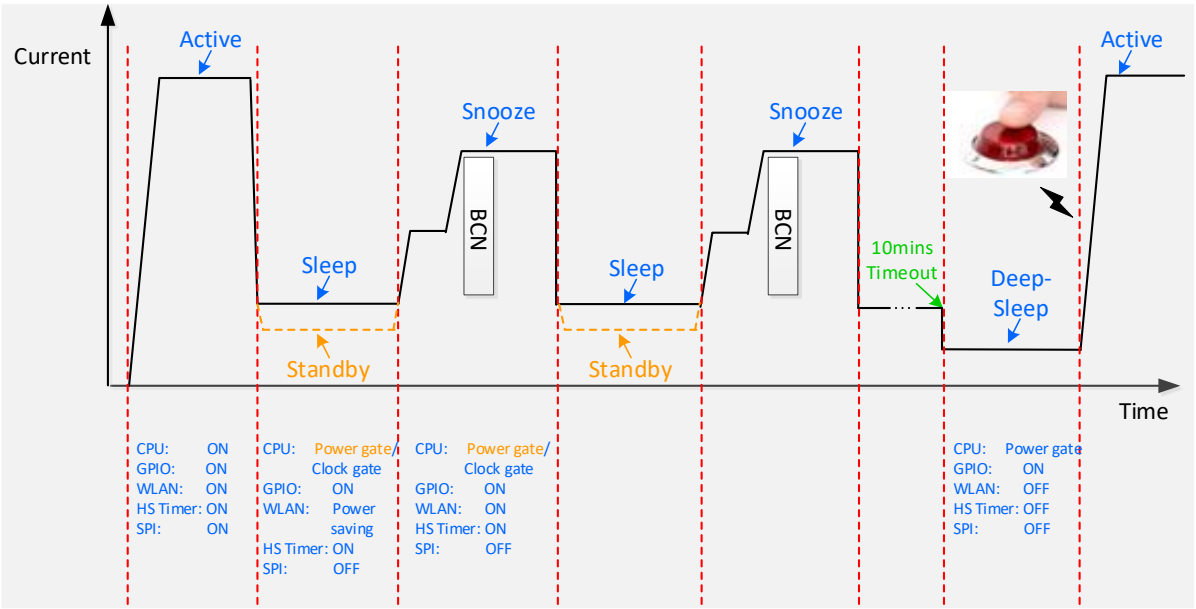
11 Power Save

11.1 Overview

11.1.1 Application Scenario

Ameba-Pro II achieves low power consumption with a combination of several proprietary technologies. The power-saving architecture features six reduced power modes of operation: active, snooze, sleep, standby, 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



11.1.2 Features

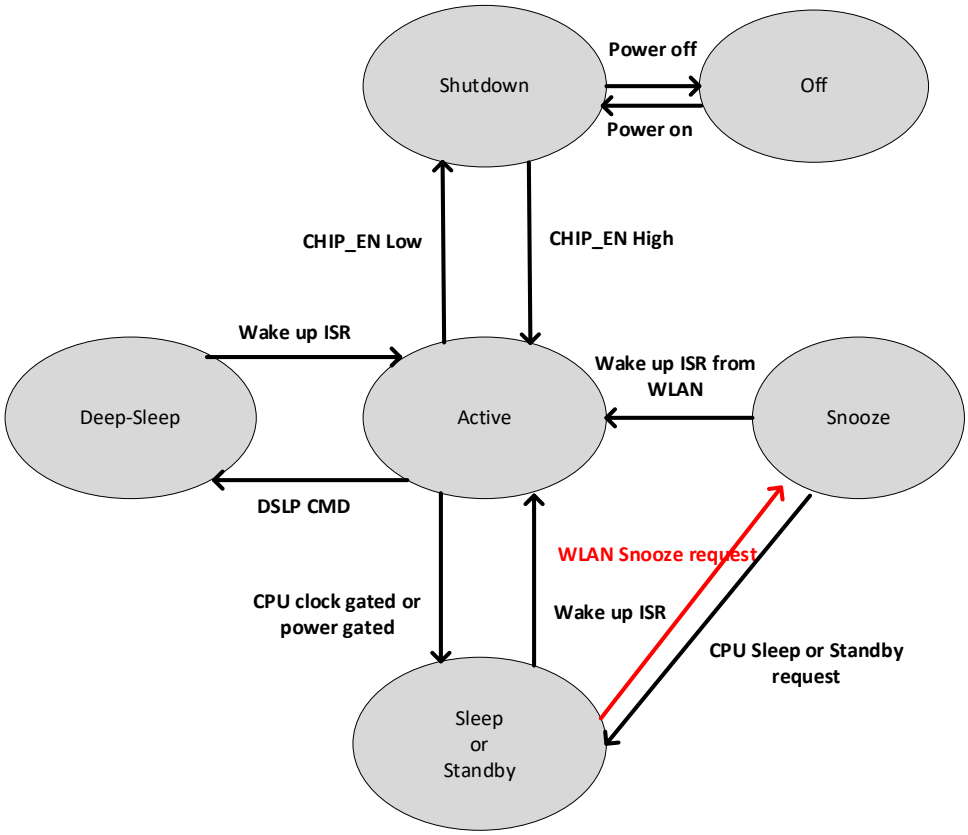
- **Active Mode**: The CPU in active mode and all peripherals are available.
- **Snooze Mode**: In this mode system can regularly wake up to receive WLAN beacon without software intervention. The significant difference between Snooze and Sleep/Standby mode is WLAN capability and could be receive and transmit beacon in this state.
- **Sleep Mode**: The CPU 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 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.

- **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.

11.1.3 Power Mode and Power Consumption

The mode changing 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.

11.2 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.

11.2.1 Wakeup Source

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

11.3 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.

11.3.1 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

11.4 Sleep Mode

- CHIP_EN keeps high. User can invoke 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 -> Execution of instructions continues.

11.4.1 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

11.5 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.

11.5.1 Wakeup Source

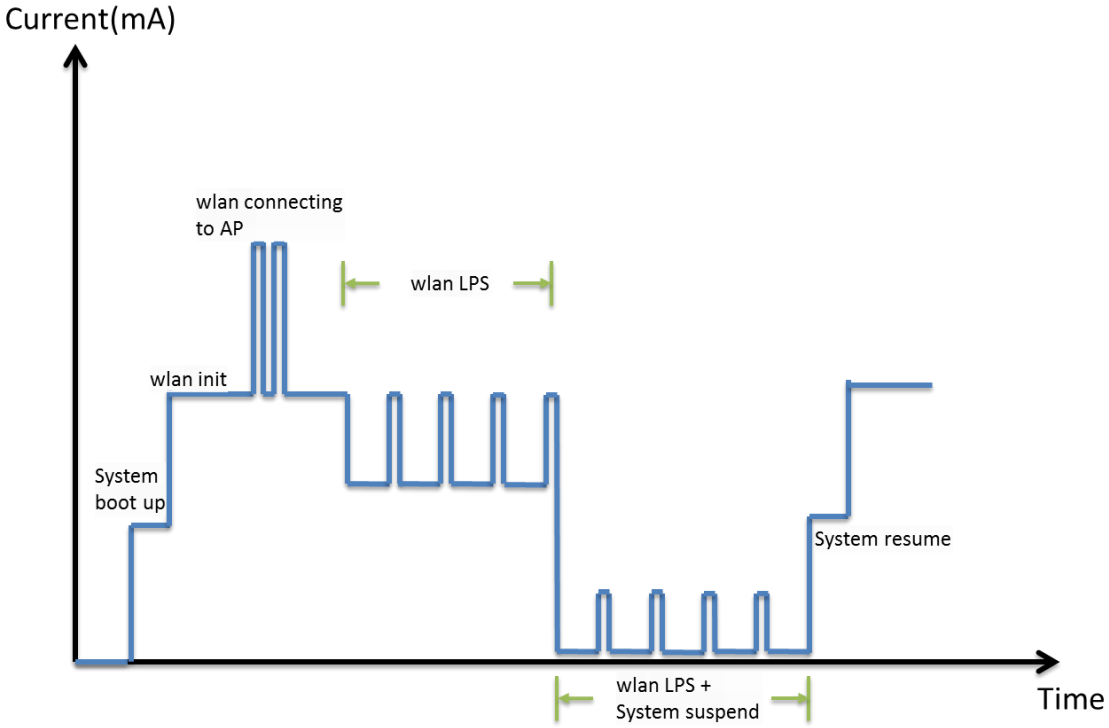
In snooze mode, the only wake up source is WLAN. The wakeup condition could be configured by WLAN driver according to system application. Once the event takes place, WLAN hardware would raise interrupt to PMC that could change hardware state.

11.6 Wowlan Mode

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:



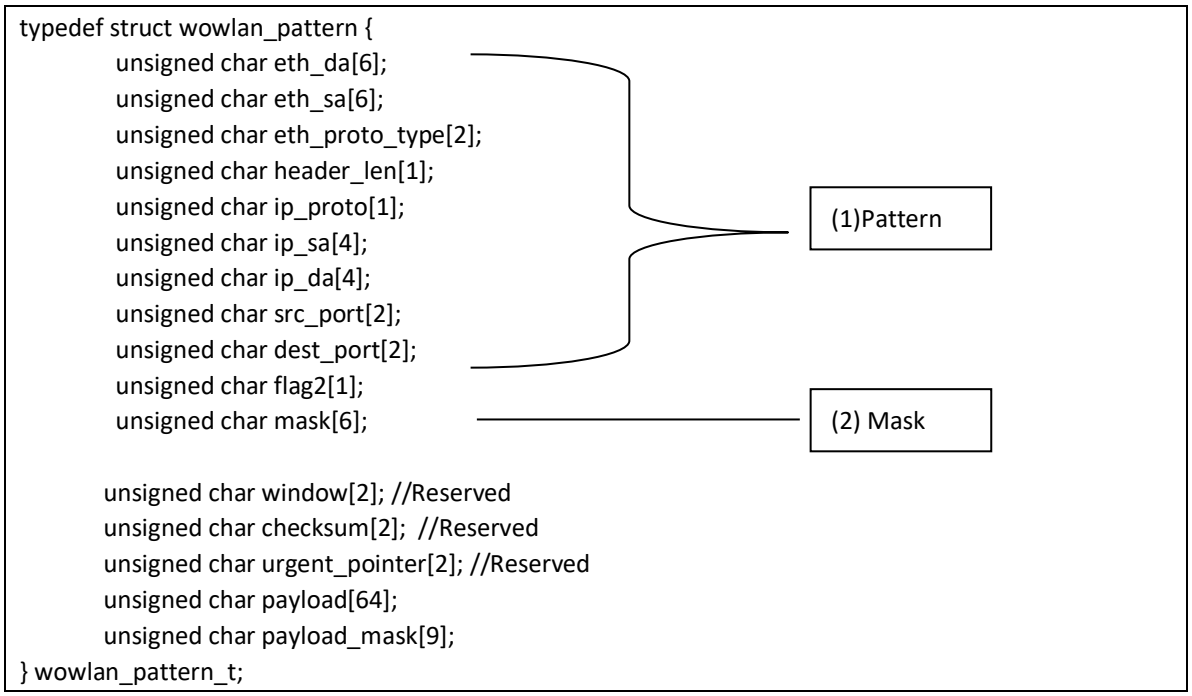
- At first system boot up, initialize wlan, connecting to AP.
- After connected to AP, wlan would enter LPS state if there is no heavy data traffic. In LPS state, wlan would listen beacon for every 100ms (if DTIM is 1). So you could see power consumption rise for every 100ms. Power consumption would 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.

11.6.1 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:



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. These fields can refer to the TCP packet format:

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

11.6.1.1 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.

11.6.2 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.

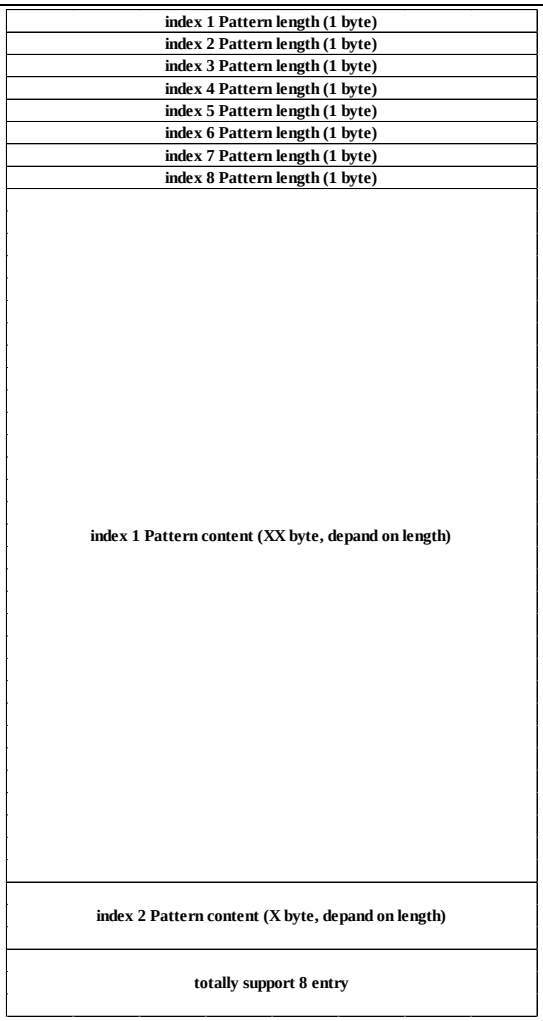
11.6.2.1 SSL Key offload

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)

11.6.2.2 SSL Wakeup pattern payload

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.



11.6.3 Keep-alive mechanism

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

11.6.3.1 TCP keep alive offload

In first, set 2 parameter 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.

11.6.3.2 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 .

12 System API

12.1 System reset

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

12.2 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)
```

12.3 JTAG/SWD disable

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

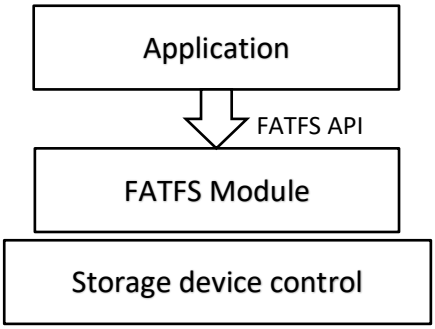
```
/**
  * @brief  Turn off the JTAG/SWD function.
  * @retval none
  */
void sys_jtag_off(void)
```

13 File System

13.1 FATFS file system introduction

Storage is a key feature of embedded system. AmebaPro2 provides flexible method of storage management. In this document, three kinds of application scenarios will be mentioned. It include the SD, RAM and flash.

13.1.1 FATFS Module



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

13.1.2 FATFS API

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

File Access API

- 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_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

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

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

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

13.1.3 FATFS EXAMPLE

The example for NOR flash and ram file system that it need to assign the start address to run the example.

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

```
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.

13.1.4 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 an 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 read back correctly.

At the end, we list all files and directories in the drive

13.1.5 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"*).

After all operations, we list all files and directories in each drive.

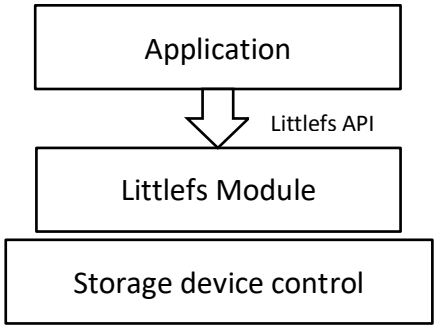
13.2 LITTLEFS File System

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

Microcontrollers and flash storage present three challenges for embedded storage: power loss, wear and limited RAM and ROM. This file system provides a solution to all three of these problems.

- **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.

13.2.1 LITTLEFS Module



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

13.2.2 LITTLEFS API

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.

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.

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.

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.

13.2.3 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.

14 Audio optimization

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

14.1 Audio setting

14.1.1 Gain setting

The audio input gain can be namely divided into digital gain and analog gain.
As for the analog gain configuration, it support 0 ~ 40 dB to support the gain optimization. The corresponding configuration method can refer to Mic Gain. Recommend that customers can first configure the analog gain. If the audio signal gain need to increase but the analog gain achieve the maximum range, then configure the digital gain.
For digital gain configuration, the suggestion gain configuration is 0dB (0x2F) ~ 4dB (0x3C), though the range is -17.625dB (0x00) ~ 30dB (0x7F). If the digital gain is too large (more than 12dB (0x4F)), digital gain will affect the sound effect and noise will be obvious. ADC Volume shows the corresponding detail setting method of the digital gain.

- **CMD_AUDIO_SET_ADC_GAIN** can be used to set the input digital gain – ADC Volume.

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.

- **CMD_AUDIO_SET_DAC_GAIN** can be used to set the output digital gain – DAC Volume.

14.1.2 Other setting

Here are some command about the audio setting:

- **CMD_AUDIO_SET_RESET** will be re-initialize the audio setting and also the ASP algorithms. If you do some change 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.

14.2 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	Causes
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

14.2.1 Open ASP algorithm

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) and VAD (Voice Activity Detection) algorithms to obtain better audio effects.

- The parameters, sample_rate and mic_gain, and the initialization of NS (enable_ns), AEC (enable_aec), AGC (enable_agc), VAD (enable_vad) and other algorithms will be setting at CMD_AUDIO_APPLY and CMD_AUDIO_SET_RESET.

```
================== Open ASP algorithm (module_audio.h) ================
#define ENABLE_ASP          1

typedef struct audio_param_s {
        audio_sr            sample_rate;   // ASR_8KHZ
        audio_wl            word_length;     // WL_16BIT
        audio_mic_gain      mic_gain;                  // MIC_40DB

        int                 channel;        // 1
        int                 enable_aec;       // 0: off  1: on
        int                 enable_ns;        // 0: off, 1: out 2: in 3: in/out
        int                 enable_agc;       // 0: off, 1: output agc
        int                 enable_vad;       // 0: off  1: input vad
        int                 mix_mode;         // 0
        //...
} audio_params_t;

typedef struct audio_ctx_s {
        void                *parent;
        audio_t             *audio;
        audio_params_t      params;
        uint8_t             inited_aec;
        uint8_t             inited_ns;
        uint8_t             inited_agc;
        uint8_t             inited_vad;
        uint8_t             run_aec;
```

```
        uint8_t             run_ns;
        uint8_t             run_agc;
        uint8_t             run_vad;

        uint32_t            sample_rate;
        uint8_t             word_length;        // Byte
        // for AEC
        TaskHandle_t        aec_rx_task;
        xSemaphoreHandle  aec_rx_done_sema;
} audio_ctx_t;

=================== ASP algorithm function (AEC.h) ================
void AEC_init(int16_t frame_size , int32_t sample_freq , AEC_CORE aec_core,
              int speex_filter_length, int16_t agc_mode, int16_t compression_gain_db, uint8_t
              limiter_enable,    int ns_mode, float snd_amplification);

int AEC_set_level(int level);
int AEC_process(const int16_t* farend, const int16_t* nearend, int16_t* out);
void AEC_destory();

void AGC_init(int32_t sample_freq, int16_t agc_mode, int16_t compression_gain_db, uint8_t
limiter_enable);
void AGC_destory(void);
void AGC_process(int16_t frame_size, int16_t* out);

void AGC2_init(int32_t sample_freq, int16_t agc_mode, int16_t compression_gain_db, uint8_t
limiter_enable);
void AGC2_destory(void);
void AGC2_process(int16_t frame_size, int16_t* out);

void NS_init(int32_t sample_freq, int16_t ns_mode);
void NS_destory(void);
void NS_process(int16_t frame_size, int16_t* out);

void NS2_init(int32_t sample_freq, int16_t ns_mode);
void NS2_destory(void);
void NS2_process(int16_t frame_size, int16_t* out);

void VAD_init(int32_t sample_freq, int16_t vad_mode);
void VAD_destory(void);
int VAD_process(int16_t frame_size, int16_t* out);
```

14.2.1.1 ASP algorithm usage

- 8K and 16K audio sample rate are supported in the ASP algorithms.
- When enable_ns is set, 0 is to disable NS, 1 is to enable NS_init() for Speaker, 2 means enable NS2_init() for MIC, 3 is to enable both directions.
- When enable_agc is set, 0 is to disable AGC, 1 is to enable AGC_init() for Speaker, 2 is to enable AGC2_init() for MIC, and 3 is to enable both directions.
- Set enable_aec 0/1 to disable/enable the AEC_init().
- In AEC_process, it will also run NS and AGC algorithms, so it will be unnecessary to apply additional NS and AGC for MIC (NS2_init() and AGC2_init()) if open enable_aec.
- Set enable_vad to 0/1 to disable/enable VAD_init().

14.2.1.1.1 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.
- Note that if using WEBRTC_AECM (default) as AEC_CORE to do initialization, the AEC strength level is 0 ~ 4 and the minimum strength is 0; while using WEBRTC_AEC as AEC_CORE to do initialization, the AEC strength level is 0 ~ 2 and the minimum strength is 0.
- Default value of WEBRTC_AECM is 3, and default value of WEBRTC_AEC is 1
- Please make sure that the level setting is after that the audio is initialed.
- The parameters agc_mode, compression_gain_db and limiter_enable of AEC_init are for setting the agc processing mode, compression gain (default 18dB) and limiter (default 0 => disable) of AGC algorithm in AEC_process.
- The parameters ns_mode of AEC_init are for setting the NS strength of NS algorithm in AEC_process.

14.2.1.1.2 AEC effect

Here is the estimation result of AEC algorithm on the device.

- For the audio setting, the MIC gain, ADC gain and DAC gain are set as 40dB, 12dB (0x4F) and 0dB (0xAF).
- Only the NS process with NS mode level 3 is applied before AEC algorithm to decrease the noise of the rx input.
- The ACOM is obtained by the average 1 minutes speech result.

For single talk test, the ACOM will be calculated after 1 second for waiting the AEC algorithm frozen.
Note: The echo loss is not large than 10 dB

The table of single talk ACOM of male and female speech

Input signal level	-10 dBm0	-15 dBm0	-20 dBm0	-25 dBm0	-30 dBm0
Male speech	71.27 dB	68.60 dB	70.60 dB	63.19 dB	57.89 dB
Female speech	66.23 dB	66.23 dB	61.40 dB	61.36 dB	51.92 dB

Note: According to the ITU G.167, for mobile radio systems, ACOM (single talk) shall be at least [45 dB] when no acoustic noise is added at the Sin interface.

For double talk test, the ACOM will be calculated right after the far end talk end (about 30 seconds) for waiting the AEC algorithm frozen.
The table of double talk ACOM of male and female speech

Input signal level	-10 dBm0	-15 dBm0	-20 dBm0	-25 dBm0	-30 dBm0

Male speech	75.30 dB	68.89 dB	67.52 dB	61.71 dB	50.96 dB
Female speech	60.74 dB	59.38 dB	62.36 dB	56.32 dB	50.65 dB

Note According to the ITU G.167, for mobile radio systems, ACOM (double talk) shall be at least [30 dB] when no acoustic noise is added at the Sin interface.

14.2.1.1.3 NS setting

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

- Use CMD_AUDIO_SET_NS_ENABLE to determine whether NSx_init() is enabled during audio reset.
- Setting parameter NS_MODE in NSx_init() to adjust ns aggressive level. The value is from 0 to 3 and the default value is 3.
- Use CMD_AUDIO_RUN_NS to dynamically switch the use of NSx_process().

14.2.1.1.4 AGC setting

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

- Use CMD_AUDIO_SET_AGC_ENABLE to determine whether AGCx_init() is enabled during audio reset.
- The parameter of agc_mode can choose the AGC mode to initial, compression_gain_db (default 18/24 for in/not in the AEC) can setting max gain of AGC and limiter_enable (default 0) to restrict the signal level.
- Use CMD_AUDIO_RUN_AGC can dynamically switch the use of AGCx_process().

14.2.1.1.5 VAD setting

The VAD algorithm is applied to do voice enhancement.

- Use CMD_AUDIO_SET_VAD_ENABLE to determine whether VAD_init() is enabled during audio reset.
- The parameter VAD MODE can set the aggressive level of VAD. The value can be configured from 0 to 3and default is 1.
- Use CMD_AUDIO_RUN_VAD to dynamically switch the use of VAD_process().

15 NN

AmebaPro2 has an NN H/W engine to accelerate the neural network inference process. NN models obtained from training with AI framework, such as Tensorflow, Caffe, MXNet and etc, are quantized to uint8 with Verisilicon's Acuity complier for AmebaPro2 NN IP.

15.1 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.

15.1.1 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;
        int input_count;
        int output_count;
        vipnn_preproc_t pre_process;
        vipnn_postproc_t post_process;
        disp_postprcess_t disp_postproc;
        bool module_out_en;
} 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
- input_count: The input tensor number of the NN network
- output_count: The output tensor number of the NN network
- disp_postproc: Set the callback function for display the NN result on video frame. It could be set by using CMD_VIPNN_SET_DISPPOST.

15.1.1.1 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 = FRAME_WIDTH,
                .height = FRAME _HEIGHT,
                .rgb = 1,
                .roi = {
                        .xmin = ROI_Xmin,
                        .ymin = ROI_Ymin,
                        .xmax = ROI_Xmax,
                        .ymax = ROI_Ymax,
                }
        }
};
```

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: input frame RGB order (1:RGB, 0:BGR).
- img.roi: ROI of input frame (xmin, ymin, xmax, ymax).

15.1.1.2 Set NN model to vipnn module

There are two model provided in SDK for evaluate – Yolov4-tiny (Yolov3-tiny) and Mobilenet-SSD. Those are both used to do object detection. 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)&yolov3_tiny);
        //mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_MODEL, (int)&mbnetssd);
        …
}
```

Note: If using Yolov4-tiny, just set model as yolov3_tiny here.

15.1.1.3 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);
        …
}
```

15.1.1.4 Set NN object detection threshold

There are two threshold value 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);
```

15.2 Object detection model

Currently, the SDK provide 2 object detection model for user to evaluate –Yolov4-tiny and Mobilenet_SSD.

15.2.1 Yolov4-tiny

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

Note: SDK also provide older Yolov3-tiny model. However, Yolov4-tiny is recommended for its better performance.

15.2.2 Mobilenet-SSD

Mobilenet is a light weight network model proposed by Google, and it can be integrated with SSD (single shot multi-box detector) to do efficient object detection.

For more information, see TensorFlow official Github:

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md

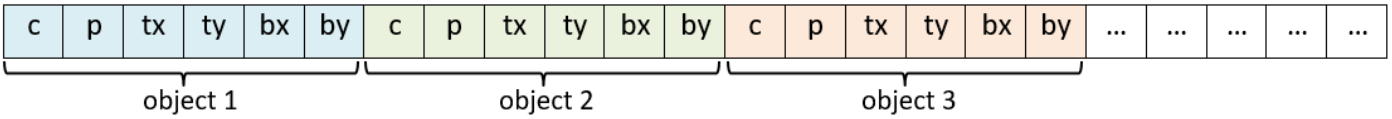
15.3 Object detection result

After NN do the inference, the detection result will be store in NN output tensor. 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 30
typedef struct objdetect_res_s {
        float result[MAX_DETECT_OBJ_NUM * 6];
        int obj_num;
```

```
} 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 following



・c: class_index

・p: probability

・tx, ty, bx, by: bounding box(top_x, top_y, bottom_x, bottom_y)

15.4 NN model prepare

NN model should be prepared before using the NN example

15.4.1 Using existing NN model in SDK

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

- yolov4_tiny.nb (yolov3_tiny.nb)
- mobilenet_ssd_uint8.nb

They are located in “project\realtek_amebapro2_v0_example\src\test_model”.

15.5 Using the NN MMF example with VIPNN module – Nand/Nor flash

The NN example is a part of mmf video joined example. Please uncomment the example want to execute.
Video joined: \project\realtek_amebapro2_v0_example\src\mmfv2_video_example\

video_example_media_framework.c

Uncomment the example want to execute:

```
mmf2_video_example_vipnn_rtsp_init();
```

Current supported VIP NN examples:

Example	Description	Result
mmf2_video_example_vipnn_rtsp_init	CH1 Video -> H264/HEVC -> RTSP CH4 Video -> RGB -> NN	RTSP video stream over the network. NN do object detection and draw the bounding box to RTSP channel.

15.5.1 Build NN example

Since it’s a part of video mmf example, user should use the following command to generate the makefile.

Build example with SDK 9.2b + patch_support_model_store_in_SNAND or later version:

Set the NN model in “project\realtek_amebapro2_v0_example\GCC-RELEASE\CMakeLists.txt”:

```
set(USED_NN_MODEL      ${prj_root}/src/test_model/yolov4_tiny.nb)
```

Check the model you set in example code is correct:

```
mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_MODEL, (int)&yolov3_tiny_fwfs);
```

Note: If using Yolov4-tiny, just set the model as “yolov3_tiny_fwfs” here.

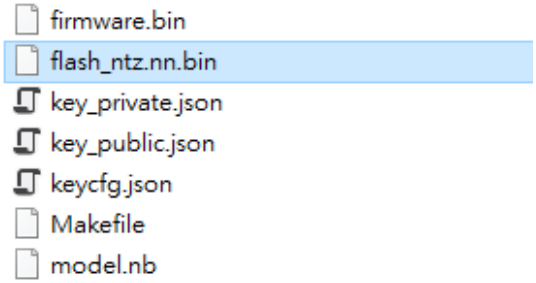
Generate the makefile for the NN project:

- 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 in “project\realtek_amebapro2_v0_example\GCC-RELEASE\build”



Then, use the image tool to download it to Amebapro2.

Build example with SDK 9.2b or older version (Nor flash only):

Check the model you set in example code is correct:

```
mm_module_ctrl(vipnn_ctx, CMD_VIPNN_SET_MODEL, (int)&yolov3_tiny);
```

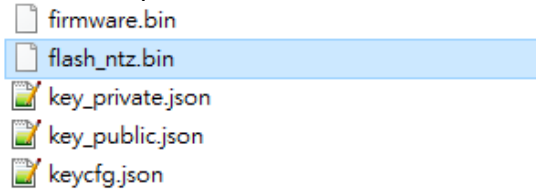
Note: If using Yolov4-tiny, just set the model as “yolov3_tiny” here.

Generate the makefile for the NN project:

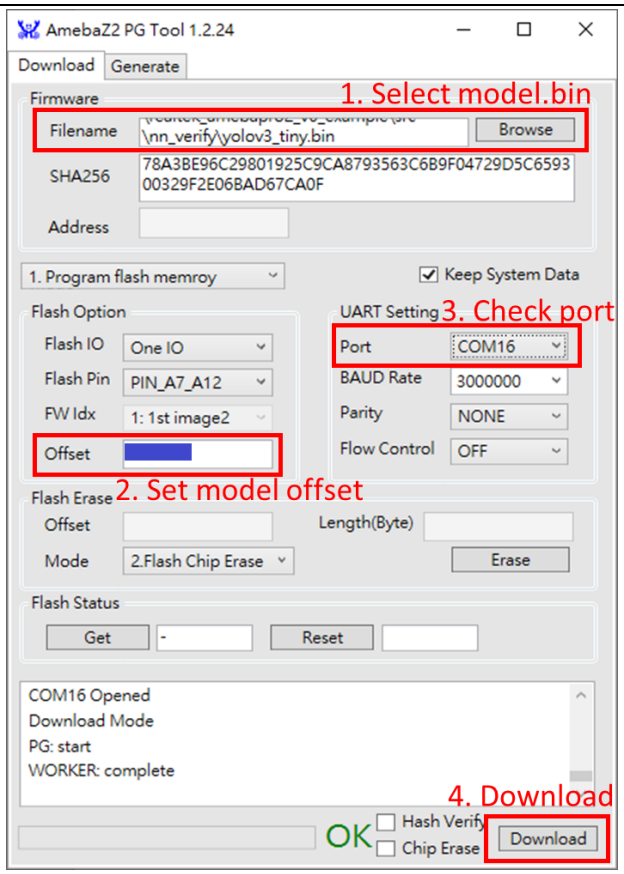
- cmake .. -G"Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=../toolchain.cmake -DVIDEO_EXAMPLE=ON
- cmake --build . --target flash

After running the command above, you will get the “flash_ntz.bin” in “project\realtek_amebapro2_v0_example\GCC-RELEASE\build”.

Use the image tool to download it to Amebapro2.



Then, also download the model binary file “yolov4_tiny.bin” to flash with offset **0x710000** (project\realtek_amebapro2_v0_example\src\test_model\yolov4_tiny.bin)



15.5.2 Validate NN example

Reboot your device and check the logs.

While runnung 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.

16 IQ

16.1 UVC Example

User can check image quality through AmebaPro2 UVC examples. User can preview image through UVC related tool, such as potplayer, Amcap or RTK realcam. This section illustrates how to build and run UVC examples in SDK

- In project\realtek_amebapro2_v0_example\inc\platform_opts.h

  Switch CONFIG_EXAMPLE_MEDIA_FRAMEWORK to 1. Then switch both

  CONFIG_EXAMPLE_MEDIA_UVCD and CONFIG_EXAMPLE_MEDIA_VIDEO to 1

```
 /* For MMFv2 example */
#define CONFIG_EXAMPLE_MEDIA_FRAMEWORK  1
#if (defined(CONFIG_EXAMPLE_MEDIA_FRAMEWORK) &&
CONFIG_EXAMPLE_MEDIA_FRAMEWORK)
#define CONFIG_EXAMPLE_MEDIA_UVCD   1
#define CONFIG_EXAMPLE_MEDIA_VIDEO  1
#define CONFIG_EXAMPLE_MEDIA_CLOUD  0
#define CONFIG_EXAMPLE_MEDIA_NN     0
#define FATFS_DISK_SD     1
#endif
```

- In project\realtek_amebapro2_v0_example\inc\platform_opts.h

```
#define SENSOR_GC2053 1
#define SENSOR_PS5258 2
#define SENSOR_GC4653 3

#define USE_SENSOR SENSOR_GC2053
```

  Select sensor through USE_SENSOR.

- Build images(For more detail, please refer to sector **1.1**)
- Use ImageTool to download images to your board.
- Use UVC related tool to preview image

17 Bluetooth

17.1 BT Example

AmebaPro2 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.

17.1.1 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(For more detail, please refer to sector **1.1**)

(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.

17.1.2 Examples List

17.1.2.1 ble_peripheral

This example shows how to create and run GATT service on GATT server.

17.1.2.1.1 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.

17.1.2.1.2 Test Procedure

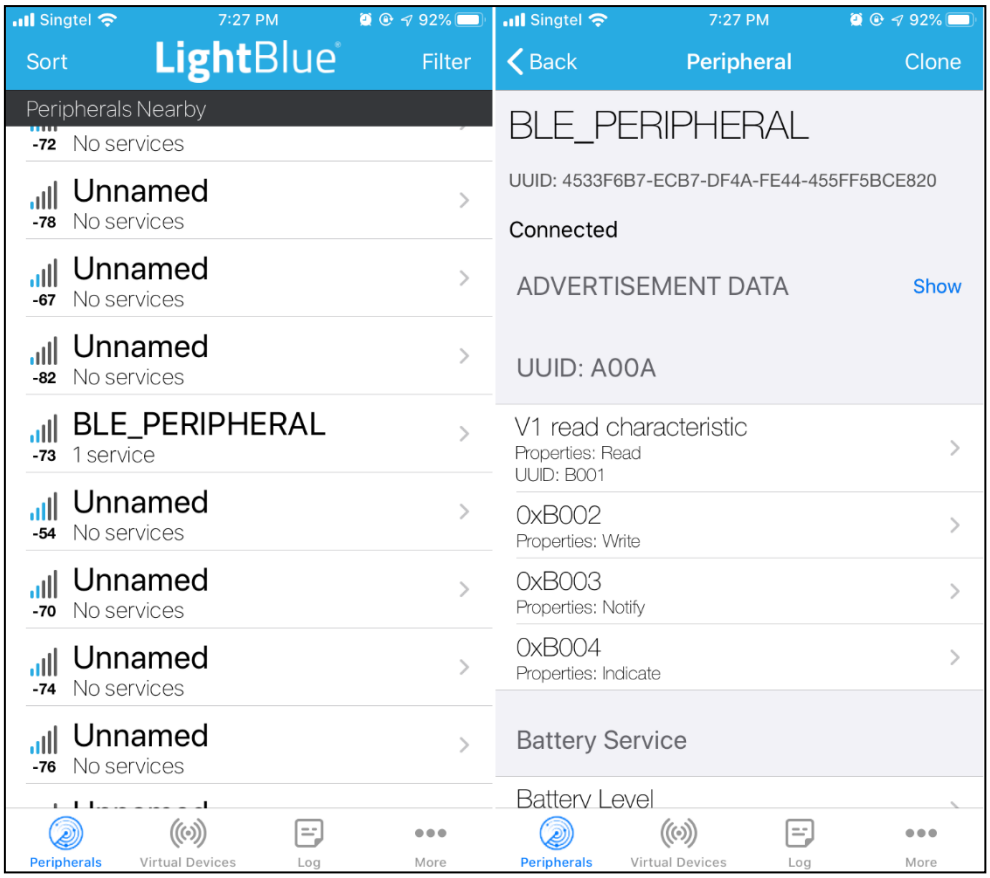
(1) After download image to your AmebaPro2 board, reset it. The default device name is BLE_PERIPHERAL.

(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.

```
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.



17.1.2.2 ble_central

This example shows how to discover service on GATT server.

17.1.2.2.1 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
```

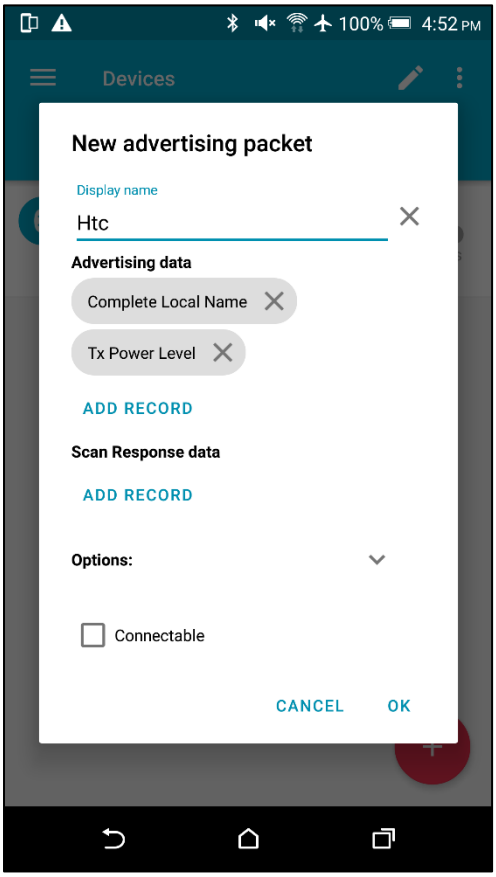
(2) Build image and download image to your board.

17.1.2.2.2 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*** '.

17.1.2.3 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.

17.1.2.3.1 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.

17.1.2.3.2 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*** '.

17.1.2.4 bt_beacon

This example shows how to send BLE Beacons. AmebaPro2 provides two types of Beacon: Apple iBeacon and Radius Networks AltBeacons.

17.1.2.4.1 Image Generation

(1) To run bt_beacon 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
```

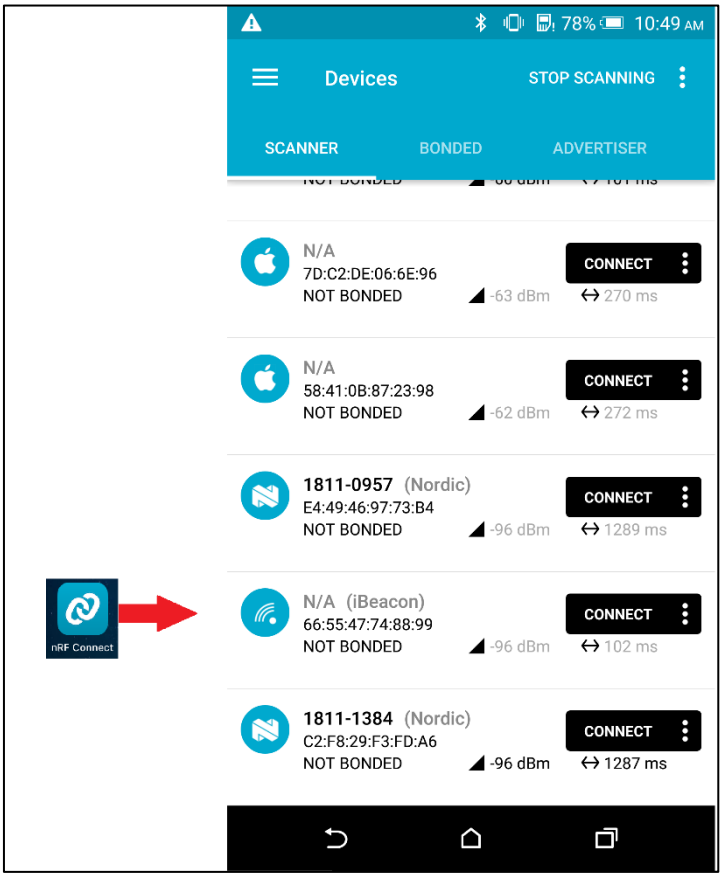
(2) Build image and download image to your board.

17.1.2.4.2 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 “Locate” on iOS, “LightBlue” or “nRF Connect” to observe beacons. “Locate” observe beacon by it adv UUID. Below screenshot is taken using Android “nRF Connect”.



17.1.2.5 bt_config

BT Config provides a simple way for Wi-Fi device to associate to AP easily.

17.1.2.5.1 Image Generation

(1) To run bt_config 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.

17.1.2.5.2 APP Installation

(1) Search “Easy WiFi Config” in the application store. You can install Android or iOS as your phone OS.

17.1.2.5.3 Test Procedure

(2) 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 AmebaPro2. Reset your AmebaPro2 board, and input command “ATBB=1”.

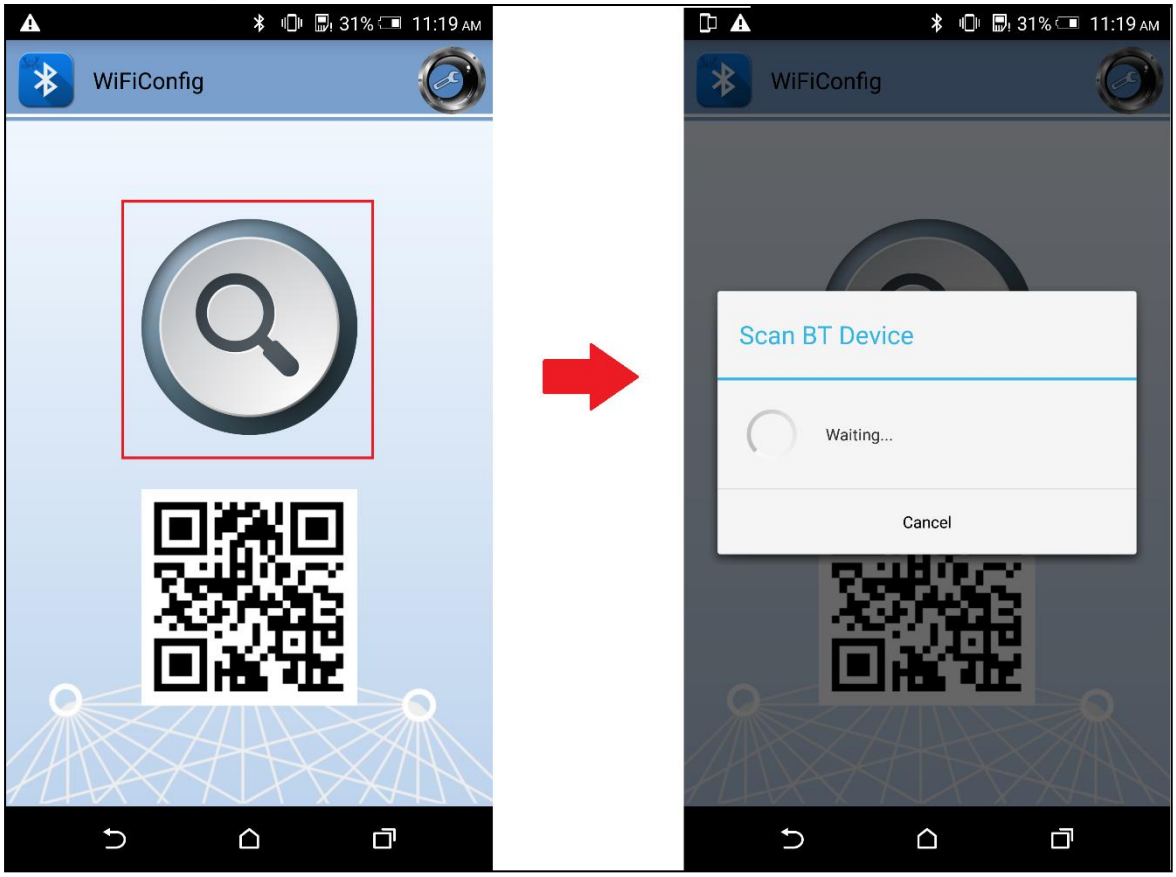
(3) Once see the following message, you can open BT Config APP to associate AP.

BT Initialize and start adv log:

```
[BT Config Wifi] BT Config Wifi ready

[BT Config Wifi] ADV started
```

(4) Click the BT config icon to launch it. Scan and connect with AmebaZII BT using BT Config app.

Display on BT config app:

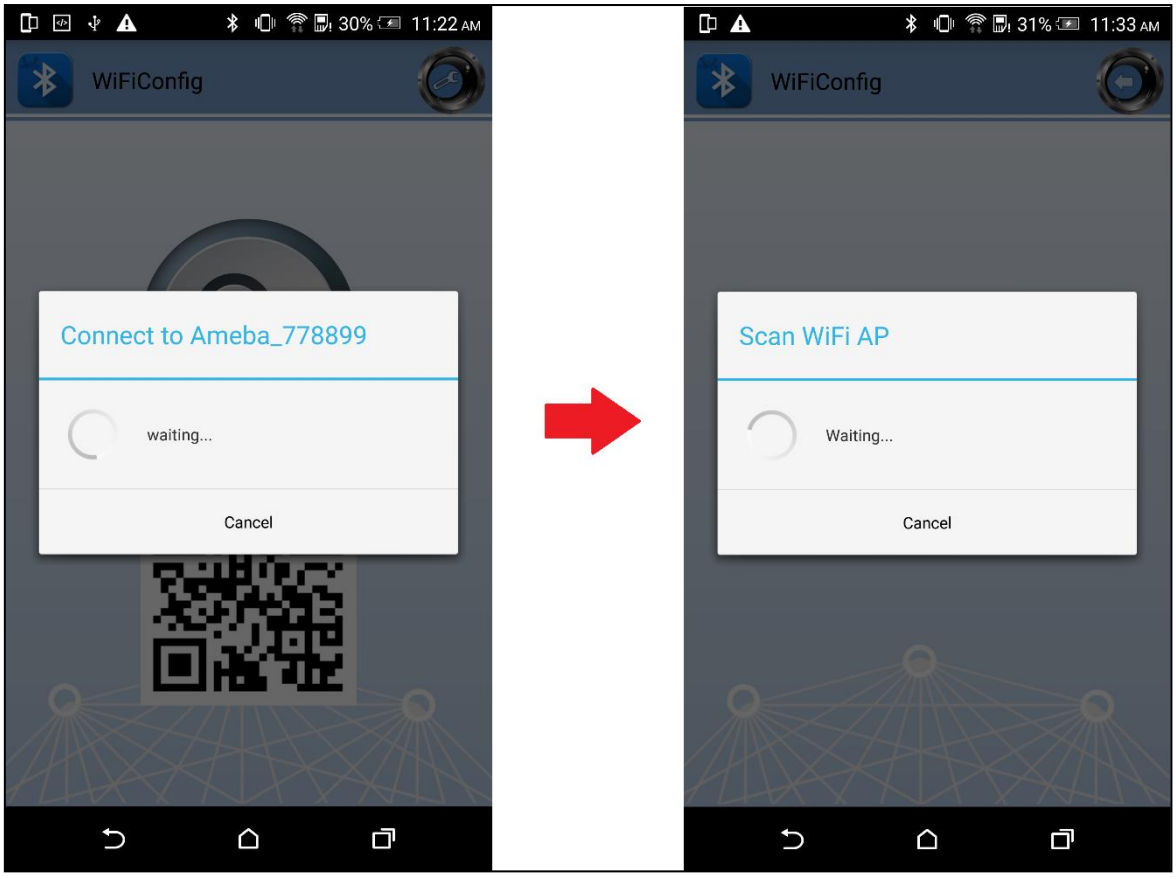


(5) Once BT Config APP connected to AmebaZII, below log will be show. When connection is established AmebaZII will start searching for AP.

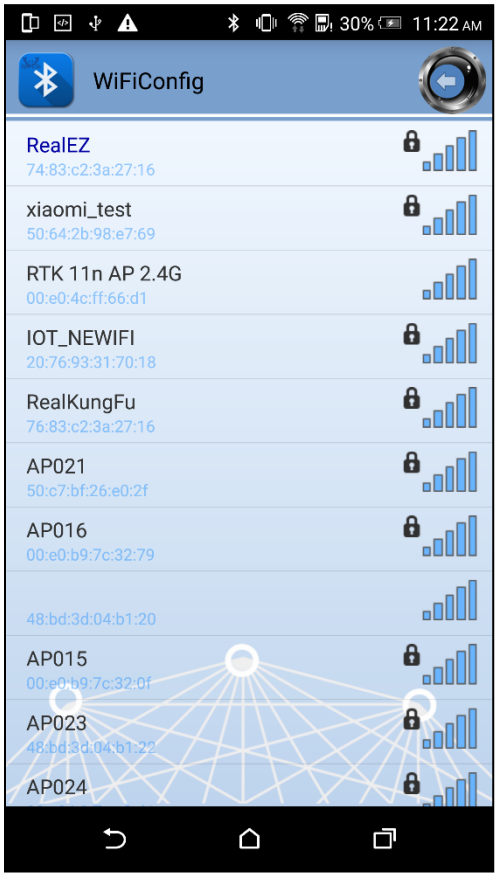
BT Connection log:

```
[BT Config Wifi] Bluetooth Connection Established

[BT Config Wifi] Band Request
[BT Config Wifi] Scan Request
[BT Config Wifi] Scan 2.4G AP
```

Display on BT config app:



Scanned and reachable APs will be show on BT config app:

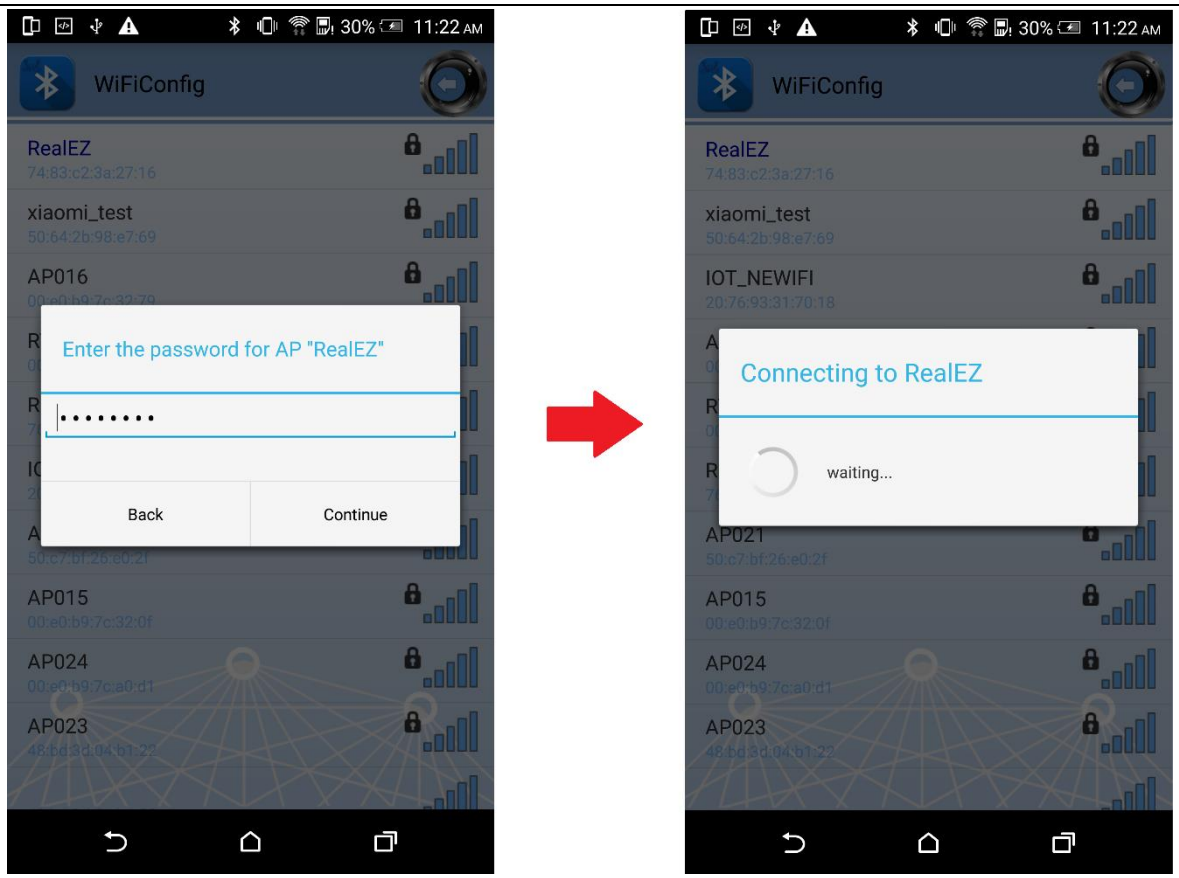


(6) Select an AP to connect to and input password (if any).

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 AES-
4)

[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.
```

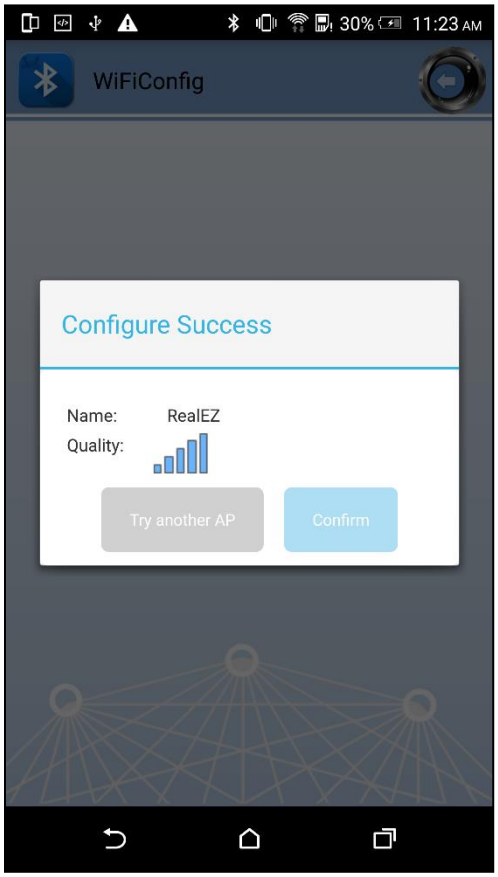
Display on BT config app:



(7) When AmebaZII 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, AmebaZ2 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:



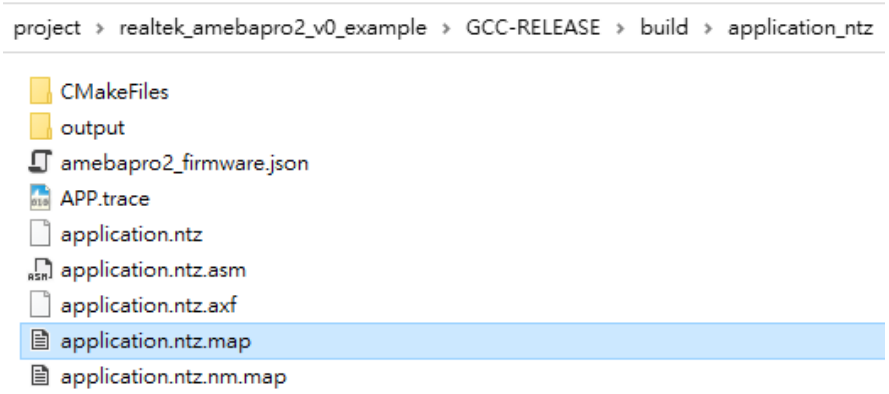
(8) You can use “ATBB=1” to restart BT Config mode again.

Command	Usage
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.

18 Memory Usage Evaluation

This section will explain how to calculate user used memory size in GCC project. User can refer “application.ntz.map” to observe them after project build. This file can be found in your project “project/realtek_amebapro2_v0_example/GCC-RELEASE/build/application_ntz/’ folder.



18.1 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

18.2 Memory Size

18.2.1 Memory Size in SRAM

There are several sections in SRAM, including “.ram.code_text”, “.ram.data”, “.ram.code_rodata”, “.ram.bss”, “.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_ntz \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.

18.2.2 Memory Size in DDR Memory (ERAM)

There are several sections in DDR Memory, ".ddr.bss", ".ddr.text", ".ddr.data", ".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 0x512af

```
.ddr.rodata     0x00000000702c45a4          0x512af
                0x00000000702c45a4          . = ALIGN (0x4)
                0x00000000702c45a4          __eram_rodata_start__ = .
```

"heap": we cannot calculate the heap usage now. However, we can get its value after running an application on ApebaPro2

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 54893248
```

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_ntz \rtl8735b_ram.ld:

```
DDR (rwx)    : ORIGIN = 0x70100000, LENGTH = 0x73900000 - 0x70100000 /* 56MB */
```

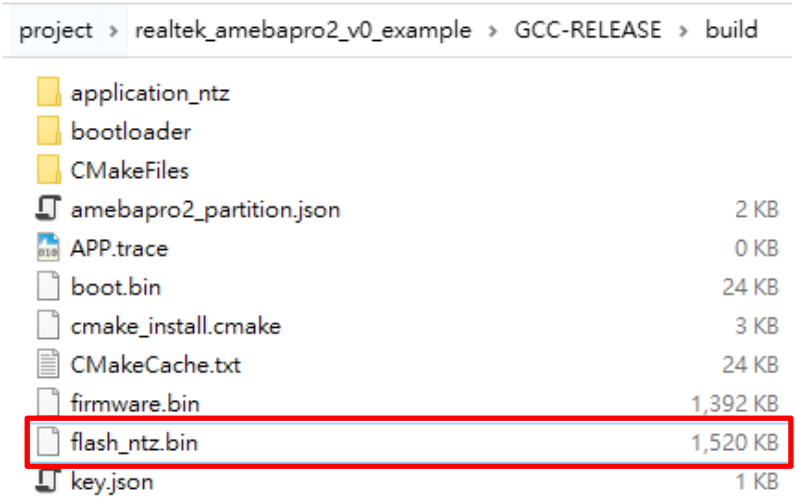
For this case, the DDR memory total size is 0x73900000 - 0x70100000 = 0x3800000 = 56MB.

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

18.3 Code Size

The size of flash_ntz.bin is the code size



19 CPU Utilization

CPU utilization can be evaluated by entering 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
[ATSS]: _AT_SYSTEM_CPU_STATS_
log_service     180             <1%
IDLE            40925           93%
siso1120        876             1%
TCP_IP          1               <1%
Tmr Svc         0               <1%
fileloader_     861             1%
mmf2_1          631             1%
vip 0           0               <1%
siso2043        317             <1%
cmd_thread      39              <1%
rtw_interru     5               <1%
rtw_recv_ta     50              <1%
rtw_xmit_ta     0               <1%
```

Revision History

Date	Version	Change
2021-09-30	V01	Initial draft
2021-11-15	V02	File System