AN0400
Ameba-D Application Note

COPYRIGHT

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 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 [illegible] Customers [illegible] 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 service information, including, but not limited to, reference designs and materials relating to evaluation [illegible] designers who are developing applications that incorporate Realtek products; by downloading, accessing or using Realtek 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 Resource 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 Customer'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

AmebaD consists of two MCUs:

- Real-M300: a high performance MCU (Armv8, Cortex-M33 instruction set compatible) called KM4 thereafter
- Real-M200: a low power MCU (Armv8, Cortex-M23 instruction set compatible) called KM0 thereafter

1 Building Environment

1.1 Introduction

This chapter illustrates how to build Realtek Wi-Fi SDK under GCC environment. It focuses on both Windows platform and Linux distribution. The build and download procedure are quite similar between Windows and Linux operating system.

- For Windows, Windows 7 64-bit is used as platform.
- For Linux distribution, Ubuntu 18.04 64-bit is used as platform.

1.2 Preparing GCC Environment

1.2.1 Windows

On Windows, you can use Cygwin as the GCC environment. Cygwin is a large collection of GNU and open source tools which provide functionality similar to a Linux distribution on Windows.

Click http://cygwin.com and download the Cygwin package setup-x86.exe for your Windows platform.

Note:

- Only 32-bit Cygwin is supported both for 32-bit Windows and 64-bit Windows
- [illegible] Select Packages [illegible] Fig 1-1 and Fig 1-2 shows

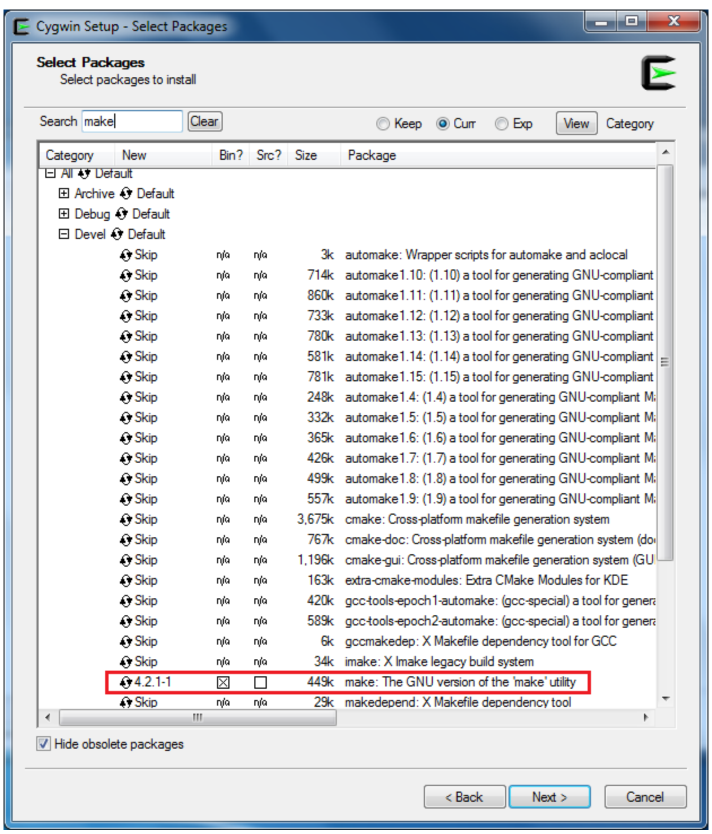


Fig1-1 "Devel" Setting during Cygwin installation

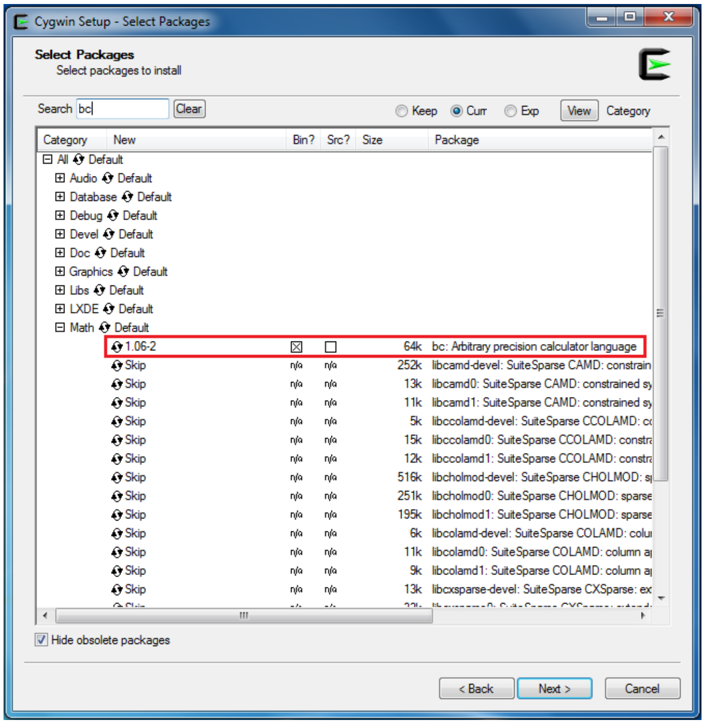


Fig1-2 ΨMath Ωsetting during Cygwin installation

1.2.2 Linux

On Linux, some packages must be installed for the GCC environment

- libc6-i386 (GNU C library for 64-bit platform. If you are using 32-bit platform, install libc6 instead)
- lib32ncurses5 (32-bit terminal handling for 64-bit platform. If you are using 32-bit platform, install libncurses5 instead)
- make
- bc
- gawk
- ncurses

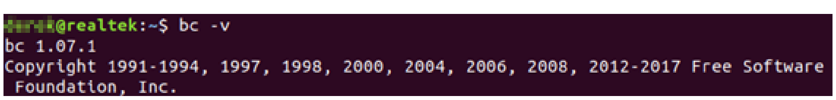
Some of these packages may have been installed in your operating system. Use package manager to check and decide whether to install. For the last three packages, you can also type the corresponding version command on terminal like the following figures to check whether it exists. If not, make these packages installed.

$ make -v

If make λ ă ý Ω ű λ ýăpt-get install make to install make.

$ bc -v

If bc λ ă ý Ω ű λ ýăpt-get install bc to install bc.



$ gawk --v

If gawk λ ă ý Ω ű λ ýăpt-get install gawk to install gawk.

```
@realtek:~$ gawk --v
GNU Awk 4.1.4, API: 1.1 (GNU MPFR 4.0.1, GNU MP 6.1.2)
Copyright (C) 1989, 1991-2016 Free Software Foundation.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see http://www.gnu.org/licenses/.
```

ncurses

ncurses is needed if you want to use make menuconfig command. Type apt-get install ncurses-dev to install ncurses

1.3 Building Code

This section illustrates how to build SDK. First, you need to switch to GCC project directory.

For Windows, open Cygwin terminal and use cd command to change directory to KM0 or KM4 project directory of AmebaD SDK.

Note: You need to replace the {path} to your own SDK location, and ensure that Cygwin can access your file system.

```
$ cd /cygdrive/{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp
$ cd /cygdrive/{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp
```

For Linux, open its own terminal and use cd command to change directory to KM0 or KM4 project directory of AmebaD SDK.

```
$ cd /{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp
$ cd /{path}/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp
```

1.3.1 Normal Image

To build SDK for normal image, simply use make all command under the corresponding project directories on Cygwin (Windows) or terminal (Linux).

1.3.1.1 KM0 Project

For KM0 project, if the terminal contains km0_image2_all.bin and [illegible] image has been built successfully, as Fig 1-3 shows.

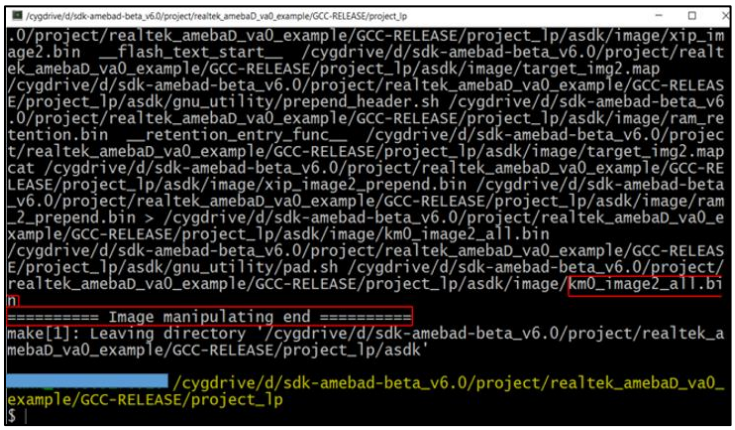


Fig 1-3 KM0 project make all

If somehow it is built failed, type $ make clean to clean and then redo the make procedure. After successfully build, image file is located in project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image, as Fig 1-4 shows.

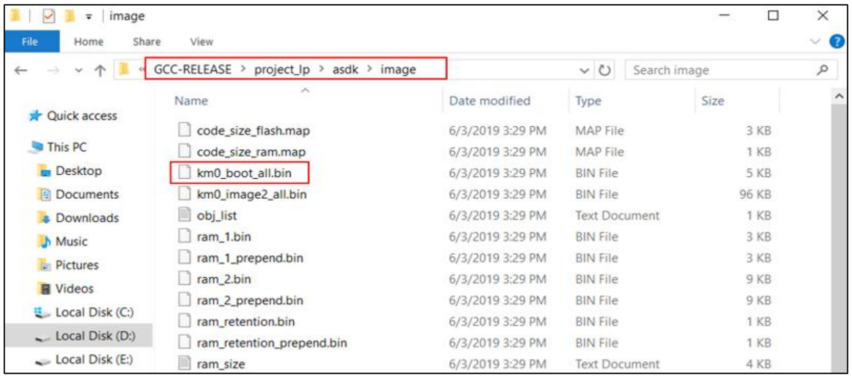


Fig1-4 KM0 project bin generation

1.3.1.2 KM4 Project

For KM4 project, if the terminal contains "km4_image2_all.bin" and "Image manipulating end" output message, it means that the image has been built successfully, as Fig1-5 shows.

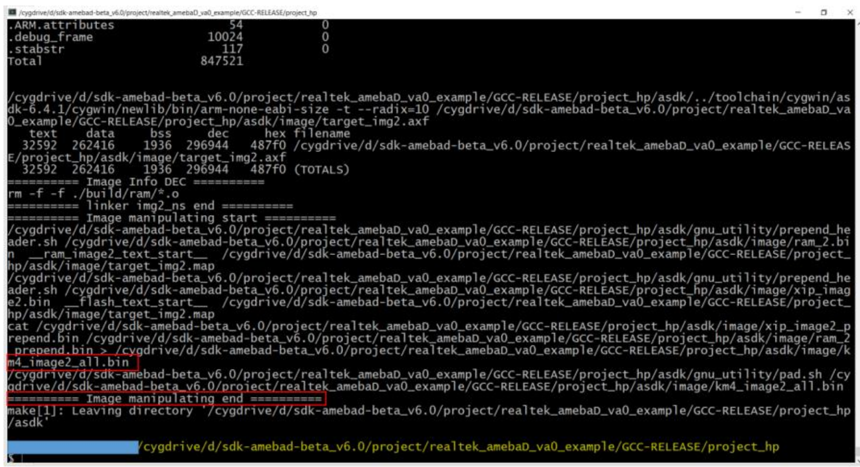


Fig1-5 KM4 project make all

If somehow it built failed, type $ make clean to clean and then redo the make procedure. After built successfully, the image file is located in project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/image, as Fig1-6 shows.

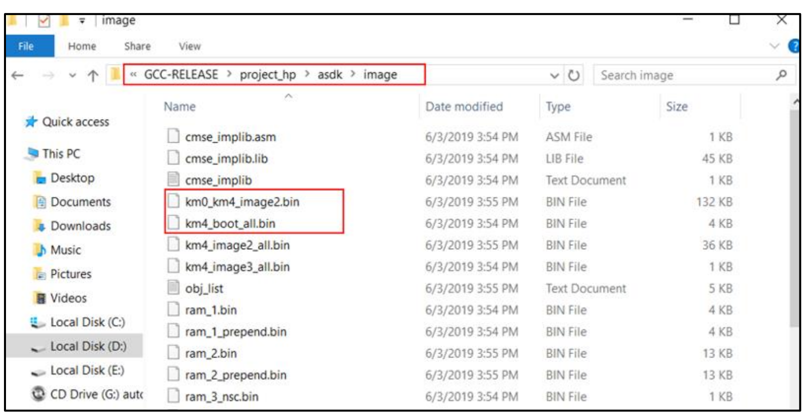


Fig1-6 KM4 project bin generation

1.3.2 MP Image

Use make mp command under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp to generate MP image. After successful compilation, you will find the generated images under project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/image, as shown in Fig1-7.

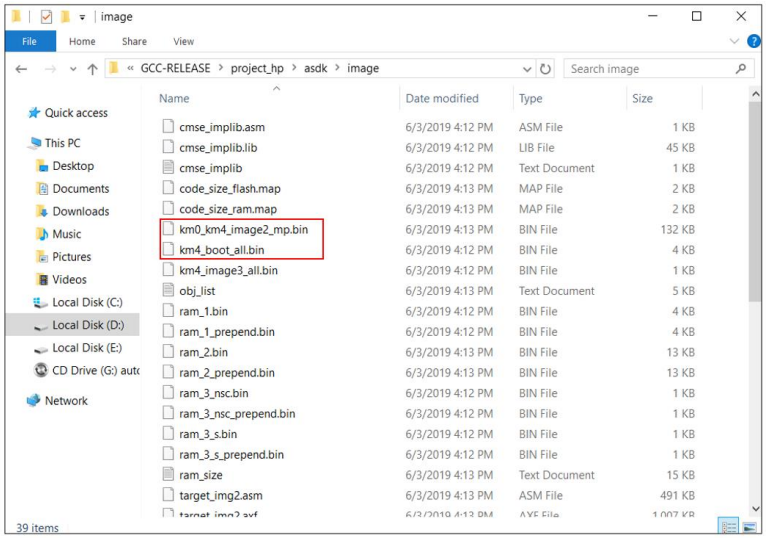


Fig1-7 MP image generation

1.4 Setting Debugger

This section illustrates how to enter GDB debugger mode.

Note: Considering KM4 is powered on by KM0, make sure that KM0 has boot up already before accessing KM4 by Link or Probe. Clicking Reset button on demo board is a recommended way to boot up KM0.

1.4.1 Probe

1.4.1.1 Windows

AmebaD Device Board supports Probe debugger. We can use Probe to download the software and enter GBD debugger mode under GCC environment. Refer to Fig1-8 to connect Probe debugger to the SWD of AmebaD.

(1) Install Probe driver

Before using the Probe, its driver is required to be installed. Obtain the RLX_Probe_Driver_2.3.1220_Setup.exe under tools/probe, then install it correctly. Refer to Fig1-8 to connect Probe debugger to the SWD of AmebaD, which is connecting TCK pin of Probe to SWD CLK pin of AmebaD, and TMS pin of Probe to SWD DATA pin of AmebaD. What's more, a common ground is needed between Probe Board and AmebaD Device Board.

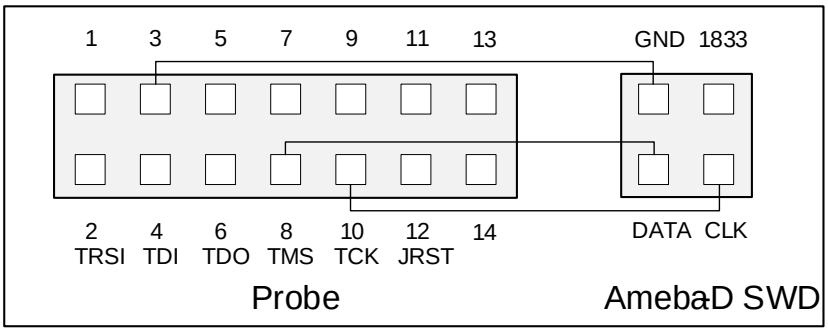


Fig1-8 Wiring diagram of connecting Probe to SWD

(2) Execute the cm0_RTL_Probe.bat

After the installation of the software package, execute the cm0_RTL_Probe.bat under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script.

Note: The default path of Probe driver in cm0_RTL_Probe.bat file is C:\RLX\PROBE\rlx_probe_driver.exe, you may have to change the path according to your own settings.

The started Probe server looks like Fig1-9. This window should NOT be closed if you want to download the image or enter debug mode.

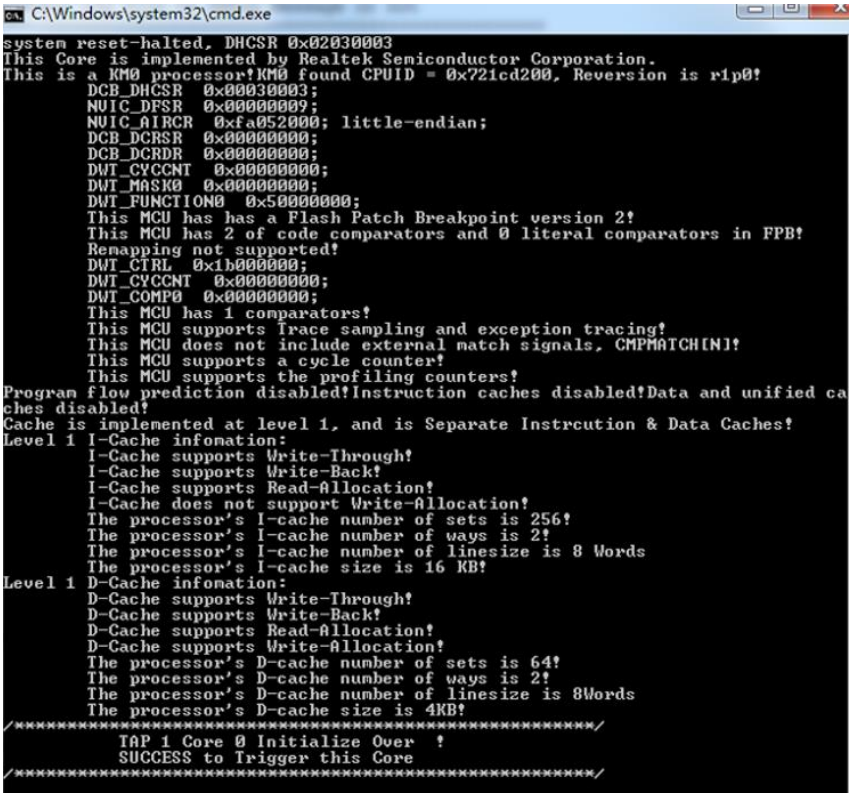


Fig1-9 KM0 Probe server connection under Windows

(3) Setup Probe for KM0

On the Cygwin terminal, type $ make setup GDB_SERVER=Probe command to select Probe debugger, as Fig1-10 shows.

```
[redacted] /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_
example/GCC-RELEASE/project_lp
$ make setup GDB_SERVER=probe
make -C asdk setup
make[1]: Entering directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_
amebaD_va0_example/GCC-RELEASE/project_lp/asdk'
----------------
Setup probe
----------------
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_debug_probe.txt /cygdri
ve/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/proje
ct_lp/asdk/gnu_utility/gnu_script/rtl_gdb_debug.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write_probe.txt /
cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE
/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com_probe.txt
 /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEA
SE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com.txt
make[1]: Leaving directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_a
mebaD_va0_example/GCC-RELEASE/project_lp/asdk'
```

Fig1-10 KM0 Probe setup under Windows

(4) Execute the cm4_RTL_Probe.bat

Execute the cm4_RTL_Probe.bat under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script the same as executing the cm0_RTL_Probe.bat. The started Probe server looks like Fig1-11. This window should NOT be closed if you want to enter debug mode.

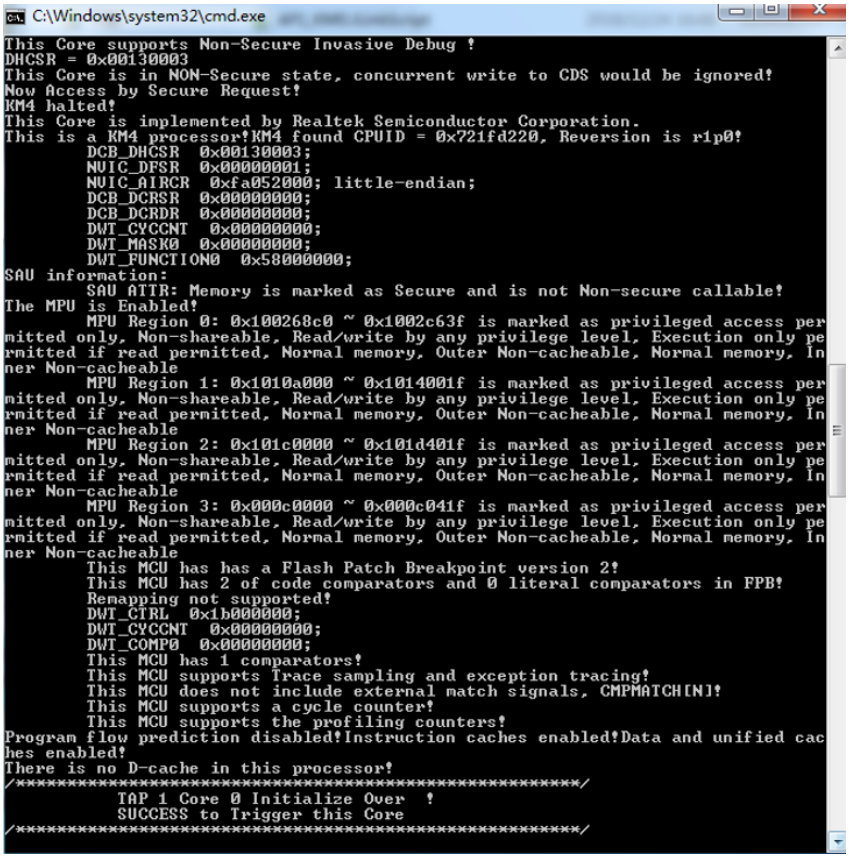


Fig1-11 KM4 Probe server connection under Windows

Note: When cm4_RTL_Probe.bat is running, the connection built by running cm0_RTL_Probe.bat will be closed. If you want to connect the Probe to KM0 and KM4 simultaneously, follow the steps below:

a) Make a copy of flx_probe0.cfg under the folder /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script and rename it as flx_probe1.cfg

b) Cut rlx_probe1.cfg and move it to /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script.
c) Connect the Probe to both KM0 and KM4 by executing cm0_RTL_Probe.bat under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script.

(5) Setup Probe for KM4
On the Cygwin terminal, type $ make setup GDB_SERVER=Probe command to select Probe debugger, as Fig1-12 shows.

```
/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_
$ make setup GDB_SERVER=probe
make -C asdk setup
make[1]: Entering directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_
p/asdk'
----------------
Setup probe
----------------
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
script/rtl_gdb_debug_probe.txt /cygdrive/d/sdk-amebad-beta_v6.0/project/realte
_hp/asdk/gnu_utility/gnu_script/rtl_gdb_debug.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
script/rtl_gdb_flash_write_probe.txt /cygdrive/d/sdk-amebad-beta_v6.0/project/
roject_hp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
script/rtl_gdb_jtag_boot_com_probe.txt /cygdrive/d/sdk-amebad-beta_v6.0/projec
/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com.txt
make[1]: Leaving directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_a
/asdk'
```

Fig1-12 KM4 Probe setup under Windows

1.4.1.2 Linux

Before using Probe, its driver is required to be installed. Obtain the RLX_Probe_Linux_Driver_v2.3.11.zip under tools/probe and install it correctly. Then connect TCK pin of Probe to SWD CLK pin and TMS pin of Probe to SWD DATA pin. What more, a common ground is needed between Probe Board and AmebaD Device Board.

After the installation of the software pack, use cs command in both folder /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script and /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script to create a soft link to link the rlx_probe_driver.elf.

```
wlan5@RS-wlan5:~/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script
$ ln -s /home/wlan5/RLX_Probe_driver/RLX_Probe_Linux_Driver/rlx_probe_driver.elf /home/wlan5/sdk-amebad-beta
_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script/
wlan5@RS-wlan5:~/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script
$ ln -s /home/wlan5/RLX_Probe_driver/RLX_Probe_Linux_Driver/rlx_probe_driver.elf /home/wlan5/sdk-amebad-beta
_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script/
```

Fig1-13 Creating soft link to elf

Open a new terminal under the corresponding project folder and type the following command to start probe server. This terminal should NOT be closed if you want to download the image or enter debug mode.

KM0 project
For KM0, open a new terminal under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script, and type $ sudo ./rlx_probe_driver.elf. If connection is successful, the log is shown as Fig1-14.

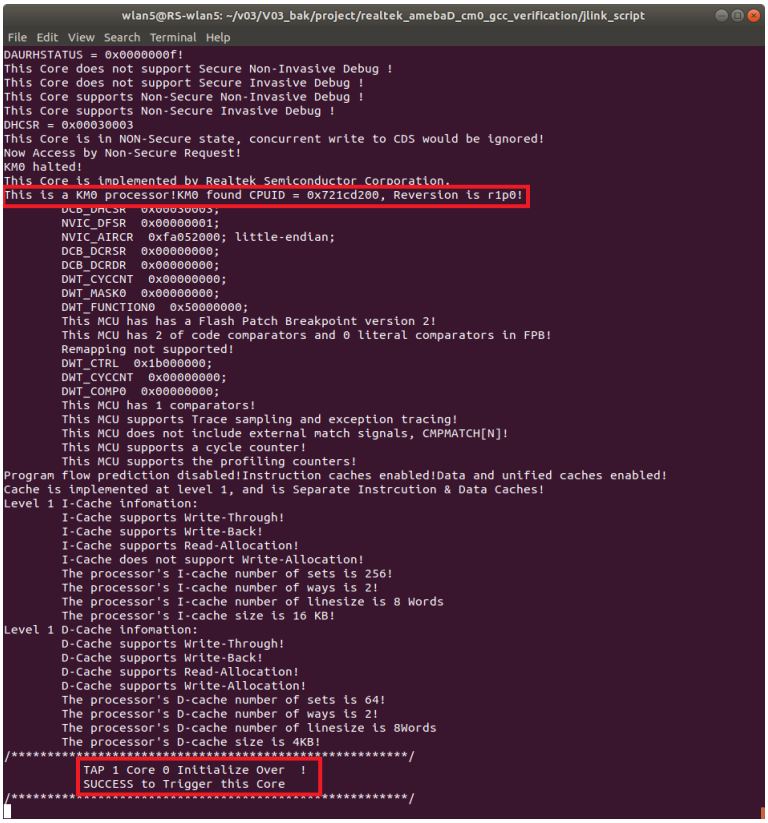


Fig1-14 KM0 probe GDB server connection success under Linux

Open a new terminal under the same project folder (/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp), type $ make setup GDB_SERVER=Probe command to select Probe debugger

KM4 project

For KM4, open a new terminal under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script, and type $ sudo./rlx_probe_driver.elf. If connection is successful, the log is shown as Fig1-15.

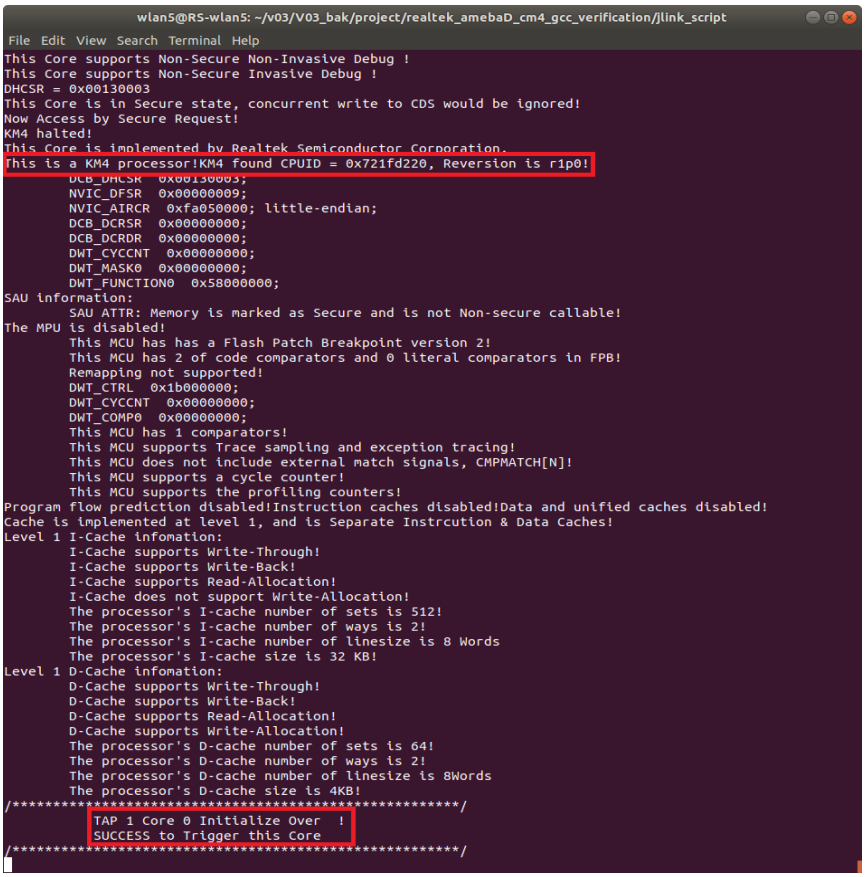


Fig1-15 KM4 probe GDB server connection success under Linux

Open a new terminal under the same project folder (/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp), type $ make setup GDB_SERVER_Probe command to select Probe debugger

1.4.2 J-Link

AmebaD also supports J-Link debugger. You need to do some hardware configuration to use J-Link debugger. Refer to Fig1-16 to connect J-Link to the SWD of AmebaD, which is connecting SWCLK pin of J-Link to SWD CLK pin of AmebaD, and SWDIO pin of J-Link to SWD DATA pin of AmebaD. After finishing these configurations, connect it to PC.

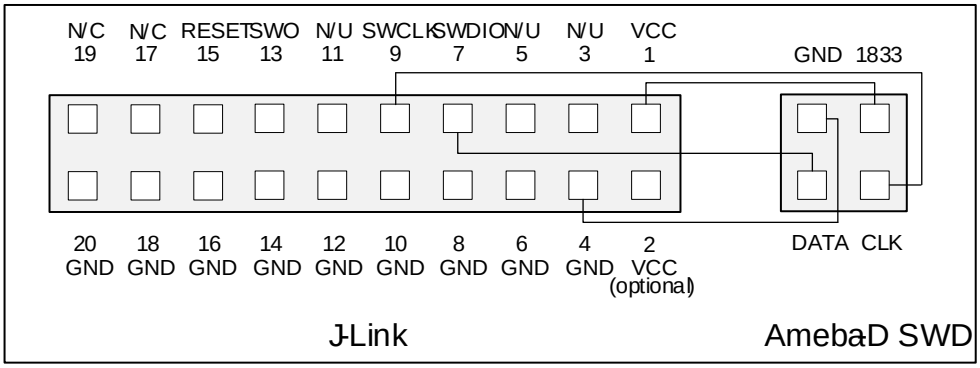


Fig1-16 Wiring diagram of connecting J-Link to SWD

Note: For AmebaD CPU, the J-Link version must be v9 or higher. If Virtual Machine (VM) is used as your platform, make sure the USB connection setting between VM host and client is correct, so that the VM host can detect the device.

1.4.2.1 Windows

(1) Install J-Link GDB server
Besides the hardware configuration, J-Link GDB server is also required to install. For Windows, click https://www.segger.com/downloads/jlink and download the software in J-Link Software and Documentation Pack, then install it correctly.
Note: The version of J-Link GDB server displayed in the pictures below is just an example, you can select the latest version to download.

(2) Execute the cm0_jlink.bat
After the installation of the software pack, execute the cm0_jlink.bat under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script.
Note: The default path of J-Link driver in cm0_jlink.bat file is C:\Program Files (x86)\SEGGER\JLink_V634b\JLinkGDBServer.exe, you may have to change the path according to your own settings.

The started J-Link GDB server looks like Fig 1-17. This window should NOT be closed if you want to download image or enter debug mode.

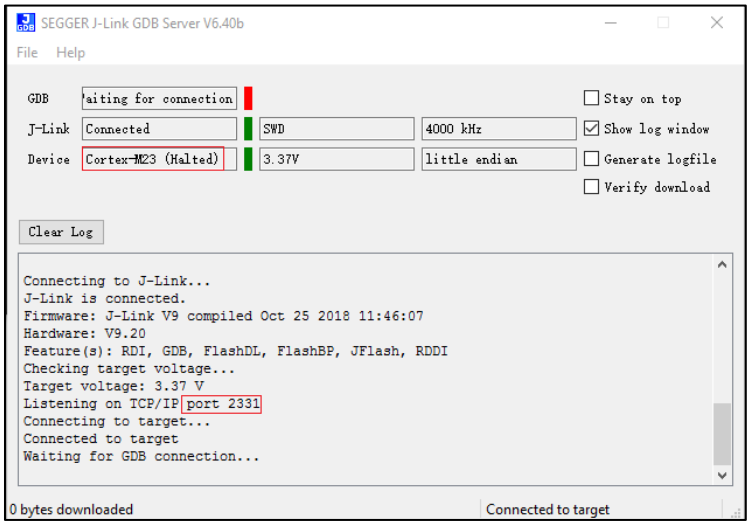


Fig 1-17 KM0 J-Link GDB server connection under Windows

(3) Setup J-Link for KM0
On the Cygwin terminal, type $ make setup GDB_SERVER=jlink command before to select J-Link debugger, as Fig 1-18 shows.

```
/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_
example/GCC-RELEASE/project_lp
$ make setup GDB_SERVER=jlink
make -C asdk setup
make[1]: Entering directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_
amebaD_va0_example/GCC-RELEASE/project_lp/asdk'
---------------
Setup jlink
---------------
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_debug_jlink.txt /cygdri
ve/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/proje
ct_lp/asdk/gnu_utility/gnu_script/rtl_gdb_debug.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write_jlink.txt /
cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE
/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com_jlink.txt
 /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEA
SE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com.txt
make[1]: Leaving directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_a
mebaD_va0_example/GCC-RELEASE/project_lp/asdk'
```

Fig1-18 KM0 J-Link setup under Windows

(4) Execute the cm4_jlink.bat

Execute the cm4_jlink.bat under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script the same as executing the cm0_jlink.bat. The started J-Link GDB server looks like Fig1-19. This window should NOT be closed if you want to download the image or enter debug mode.

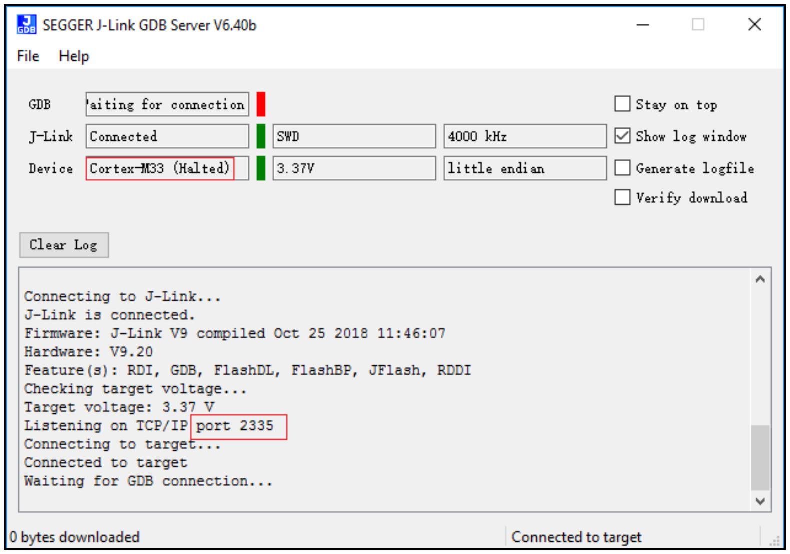


Fig1-19 KM4 J-Link GDB server connection under Windows

(5) Setup J-Link for KM4

On the Cygwin terminal, type $ make setup GDB_SERVER=jlink command to select J-Link debugger, as Fig1-20 shows.

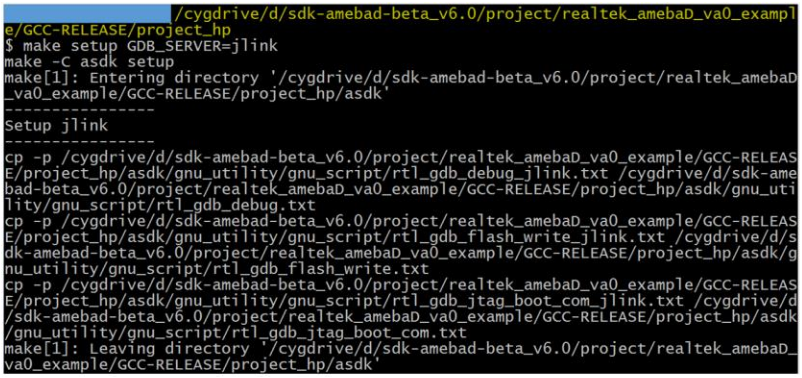

```
/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_exampl
e/GCC-RELEASE/project_hp
$ make setup GDB_SERVER=jlink
make -C asdk setup
make[1]: Entering directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD
_va0_example/GCC-RELEASE/project_hp/asdk'
----------------
Setup jlink
----------------
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEAS
E/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_debug_jlink.txt /cygdrive/d/sdk-ame
bad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_uti
lity/gnu_script/rtl_gdb_debug.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEAS
E/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write_jlink.txt /cygdrive/d/s
dk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/g
nu_utility/gnu_script/rtl_gdb_flash_write.txt
cp -p /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEAS
E/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com_jlink.txt /cygdrive/d
/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk
/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com.txt
make[1]: Leaving directory '/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_
va0_example/GCC-RELEASE/project_hp/asdk'
```

Fig1-20 KM4 J-Link setup under Windows

1.4.2.2 Linux

For J-Link GDB server, click https://www.segger.com/downloads/jlink and download the software in J-Link Software and Documentation Pack. ɛ Φ ² Ŝ ădzƎƎŜăű dzăλƴƎ 5Ŝōλŀƴ LJŀŎ.ʅ ŀƎŜ Ƴŀƴŀ ƎŜNJ ű ƻ λƴăűŀ ƭ ƭ űKŜ 5Ŝō

$ dpkg -i jlink_6.0.7_x86_64.deb

After the installation of the software pack, there is ŀ ű ƻƻ ƭ ƴŀƳŜŘ ά W [λ ƴ ʅ D 5 . { ŜNJǾ ŜNJ ɛ04 dzƴ ŘŜNJ example, the [λ ƴ ʅ JLinkGDBServer can be found at /opt/SEGGER/JLink. Open a new terminal under the corresponding project folder and type the following command to start GDB server. This terminal should NOT be closed if you want to download the image or enter debug mode.

1.4.2.2.1 KM0 Project

For KM0, open a new terminal under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/jlink_script, and type $ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m23 -if SWD -scriptfile AP1_KM0.JLinkScript -port 2331, as Fig1-21 shows. If connection is successful, the log is shown as Fig1-22.

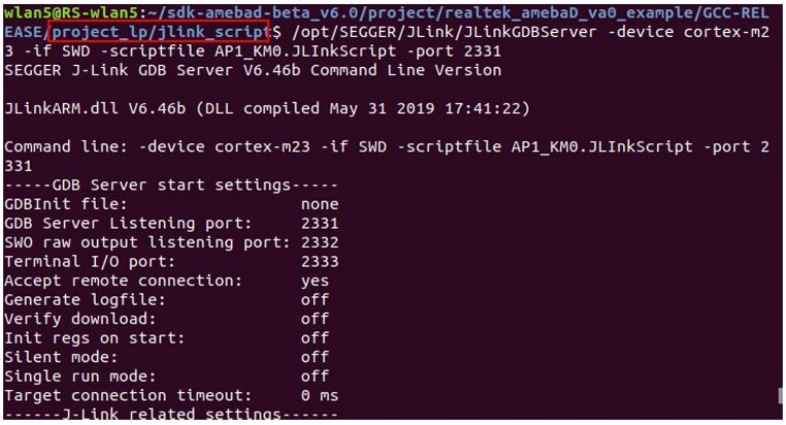


Fig1-21 KM0 JLink GDB server connection setting under Linux

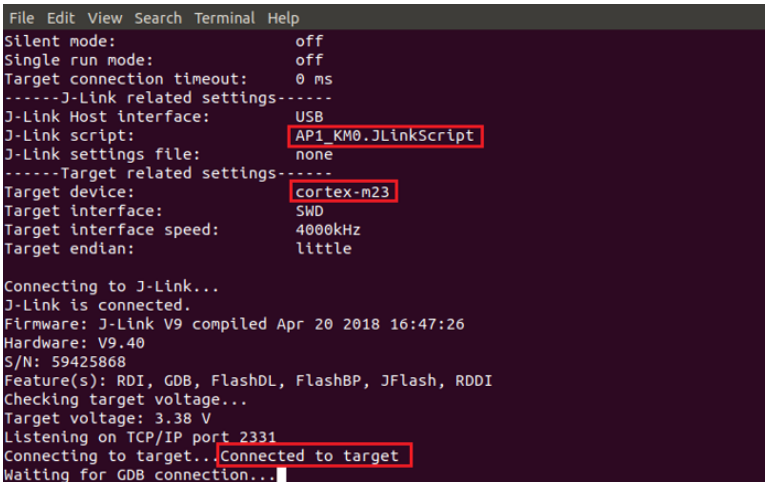


Fig1-22 KM0 JLink GDB server connection success under Linux

Open a new terminal under the same project folder, type $ make setup GDB_SERVER=jlink command to select J-Link debugger, as Fig1-23 shows

Fig1-23 KM0 JLink terminal setup under Linux

1.4.2.2.2 KM4 Project

For KM4, open a new terminal under /project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script, and type $ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m33 -if SWD -scriptfile AP2_KM4.JLinkScript -port 2335, as Fig1-24 shows. If connection is successful, the log is shown as Fig1-25.

```
wlan5@RS-wlan5:~/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-REL
EASE/project_hp/jlink_script$ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m3
3 -if SWD -scriptfile AP2_KM4.JLinkScript -port 2335
SEGGER J-Link GDB Server V6.46b Command Line Version

JLinkARM.dll V6.46b (DLL compiled May 31 2019 17:41:22)

Command line: -device cortex-m33 -if SWD -scriptfile AP2_KM4.JLinkScript -port 2
335
-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2335
SWO raw output listening port: 2332
Terminal I/O port:             2333
Accept remote connection:      yes
Generate logfile:              off
Verify download:               off
Init regs on start:            off
Silent mode:                   off
Single run mode:               off
Target connection timeout:     0 ms
------J-Link related settings------
```

Fig1-24 KM4 JLink GDB server connection setting under Linux

```
File  Edit  View  Search  Terminal  Help
Silent mode:                   off
Single run mode:               off
Target connection timeout:     0 ms
------J-Link related settings------
J-Link Host interface:         USB
J-Link script:                 AP2_KM4.JLinkScript
J-Link settings file:          none
------Target related settings------
Target device:                 cortex-m33
Target interface:              SWD
Target interface speed:        4000kHz
Target endian:                 little

Connecting to J-Link...
J-Link is connected.
Firmware: J-Link V9 compiled Apr 20 2018 16:47:26
Hardware: V9.40
S/N: 59425868
Feature(s): RDI, GDB, FlashDL, FlashBP, JFlash, RDDI
Checking target voltage...
Target voltage: 3.37 V
Listening on TCP/IP port 2335
Connecting to target...Connected to target
Waiting for GDB connection...
```

Fig1-25 KM4 JLink GDB server connection success under Linux

Open a new terminal under the same project folder, type $ make setup GDB_SERVER=jlink command to select J-Link debugger, as Fig1-26 shows.

```
wlan5@RS-wlan5:~/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp$ make setup GDB_SERVER=jlink
make -C asdk setup
make[1]: Entering directory '/home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk'
---------------
Setup jlink
---------------
cp -p /home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_debug_jlink.txt /ho
e/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_debug.txt
cp -p /home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write_jlink.t
t /home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_flash_write.txt
cp -p /home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com_jlink
txt /home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/gnu_utility/gnu_script/rtl_gdb_jtag_boot_com.txt
make[1]: Leaving directory '/home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk'
```

Fig1-26 KM4 JLink terminal setup under Linux

1.5 Downloading Image to Flash

This section illustrates how to download image to Flash.

To download software into AmebaD Device Board, make sure steps mentioned in section 1.3 to 1.4 are done and then type $ make flash command on Cygwin (Windows) or terminal (Linux).

Both KM0 and KM4 download code by this command. This command download the software into Flash and it will take several seconds to finish, as shown in Fig 1-27.

```
Breakpoint 1, RtlFlashProgram () at /cygdrive/e/v03newest/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/flashloader/rtl_flash_d
88                      asm("nop");
flash_write FileName:2
flash_write FileSize:0
flash_write FlashAddrForWrite:6000
FileSize: 0
Loopnumber = 0
TailSize = 0
global variables
FlashDatSrc:822ac
FlashBlockWriteSize:800
FlashAddrForWrite:6000Flash write start...
dump for check

Breakpoint 2, RtlFlashProgram () at /cygdrive/e/v03newest/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/flashloader/rtl_flash_d
120                     if  (FlashWriteComplete == 1) {
make[1]: Leaving directory '/cygdrive/e/v03newest/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk'

fiona_shen@Y38471517 /cygdrive/e/v03newest/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp
```

```
17:05:21.123  #ROM:[V1.2]
17:05:21.183  FLASHRATE:1
17:05:21.194  ============================================================
17:05:21.259  Flash Downloader Build Time: 2019/07/05-16:54:02
17:05:21.276  ============================================================
17:05:21.291  Flash init start
17:05:21.309  Flash init done
17:05:21.322  Flash download start
17:05:21.712  Flash download done
```

log in trace tool

Fig 1-27 Download code's success log

After downloaded successfully, press the Reset button and you can see the device is booted with new image.

NOTE

For new device board or device board with erased all images, program both KM0 and KM4 by commands with the sequence of $ make flash in project_lp, then change the path and type $ make flash in project_lp. After both images are downloaded successfully, press the Reset ō dz ű ű 2 y' Φ h ű K Ŝ NJ ǿ ƛ ă Ŝ Σ ά C f l· ă K b 2 ű LJNJ 2 ∃ NJ l· Y Y Ŝ Ř ε ǿ ƛ f f
For Probe download,

- *Make chip enter download mode before download code to Flash*
- *Download KM4 project code also through km0_RTL_Probe.bat*
- *Probe uses USB 1.0 interface, so its download rate is limited by the USB 1.0 protocol*

1.6 Entering Debug Mode

To enter GDB debugger mode, make sure steps mentioned in section 1.3 to 1.5 are finished and then reset the device first. After reset the chip, type $ make debug command on Cygwin (Windows) or terminal (Linux).

For Windows, a popup window will display, as shown in Fig 1-28.
For Linux, the log is shown in Fig 1-29.

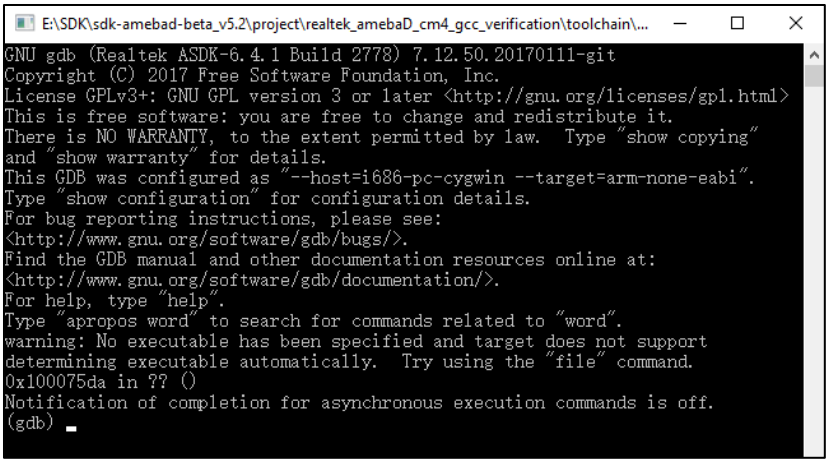


Fig1-28 Debugwindow under Windows

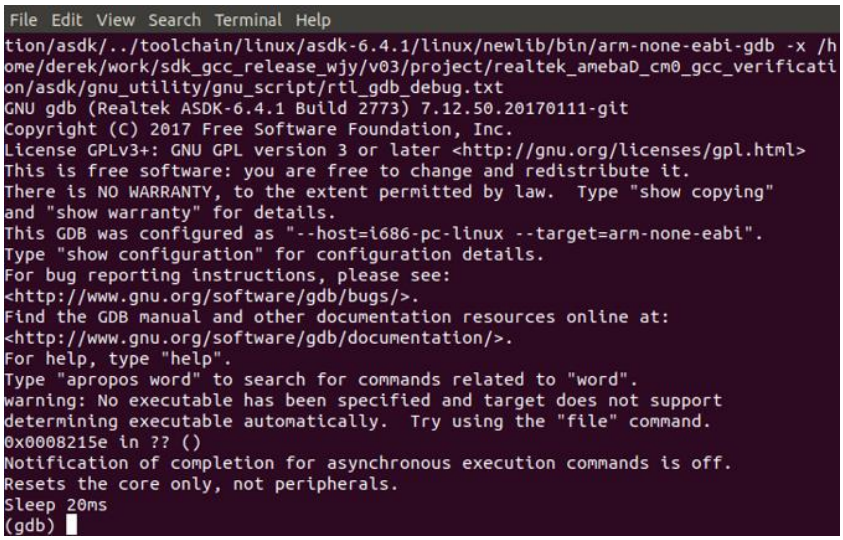


Fig1-29 Debugwindow under Linux

1.7 Command List

Thecommands used above are listedTiable1-1.

Table1-1 Commandist

Usage	Command	Description
all	$ make all	Compile project to generate ram_all.bin
setup	$ make setup GDB_SERVER= jlink	SelectGDB_SERVER
flash	$ make flash	Download ram_all.bin toFlash
clean	$ make clean	Remove compilefile ó ╒ Φ ō λ γ Σ ╒ Φ ≥ Σ X ú
clean_all	$ make clean_all	Remove compile result and toolchains
debug	$ make debug	Enterdebug mode

1.8 GDB Debugger Basic Usage

GDB, the GNU project debugger, allows you to examine the program while it executes, and it is helpful for catching bugs. Section 1.6 has described how to enter GDB debugger mode, this section illustrates some basic usage of GDB commands. For further information about GDB debugger and its commands, click https://www.gnu.org/software/gdb/ and https://sourceware.org/gdb/current/onlinedocs/gdb/

Table 1-2 GDB debugger command list

Usage	Command	Description
Breakpoint	$ break	Breakpoints are set with the *break* command (abbreviated *b*). The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Breaks.html
Watchpoint	$ watch	You can use a watchpoint to stop execution whenever the value of an expression changes. The re commands include *watch*, *rwatch*, and *awatch*. The usage of these commands can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Set Watchpoints.html Note: Keep the range of watchpoints less than 20 bytes
Print breakpoints & watchpoints	$ info	To print a table of all breakpoints, watchpoints set and not deleted, use the *info* command. You can simply type *info* to know its usage.
Delete breakpoints	$ delete	To eliminate the breakpoints, use the *delete* command (abbreviated *d*). The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Delete-Breaks.html
Continue	$ continue	To resume program execution, use the *continue* command (abbreviated *c*). The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html
Step	$ step	To step into a function call, use the *step* command (abbreviated *s*). It will continue running your program until control reaches a different source line. The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html
Next	$ next	To step through the program, use the *next* command (abbreviated *n*). The execution will stop when control reaches a different line of code at the original stack level. The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html
Quit	$ quit	To exit GDB debugger, use the *quit* command (abbreviated *q*), or type an end-of-file character (usually *Ctrl-d*). The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Quitting-GDB.html
Backtrace	$ backtrace	A backtrace is a summary of how your program got where it is. You can use *backtrace* command (abbreviated *bt*) to print a backtrace of the entire stack. The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Backtrace.html
Print source lines	$ list	To print lines from a source file, use the *list* command (abbreviated *l*). The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/List.html
Examine data	$ print	To examine data in your program, you can use *print* command (abbreviated *p*). It evaluates and prints the value of an expression. The usage can be found at https://sourceware.org/gdb/current/onlinedocs/gdb/Data.html

1.9 Q & A

1.9.1 ά 9NJNJ2NJ ᴍʜᴛέ Ŧ2NJ .dzƛ f Řƛ ƴ∃ /2ŘŜ

If you use $ make all command to build code but encounter ά 9NJNJ2NJ ᴍʜᴛέ Figure 1-30 ...
Both 32-bit and 64-bit operating system need to install 32-bit installation Cygwin Package Setup-x86.exe

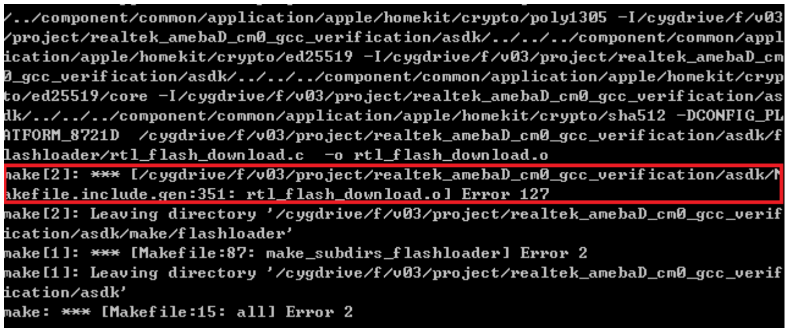


Fig1-30 ά rror 127 ἐ Ŝ NJNJ ≥ NJ Y Ŝ ă ă Ⅰ ∃ Ŝ

1.9.2 ά t Ŝ NJ Y ʎ ă ʎ ≥ ƴ Ř Ŝ ƴ ʎ Ŝ Ř ἐ 9 NJNJ ≥ NJ dz ƴ Ř Ŝ NJ [ʎ ƴ dz È

If you use $ makeall command to compile project but encounter Permission denied Ŝ NJNJ ≥ NJ like Fig1-31, it is caused by file access permission. You can enter chmod -R 777 $SDK_FILENAME under the path of SDK first to change the permission of files in SDK, and then continue compile operation.

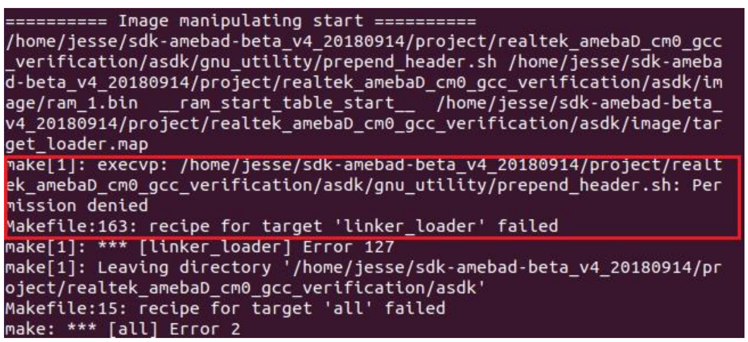


Fig1-31 ά t Ŝ NJ Y ʎ ă ă ʎ ≥ ƴ Ř Ŝ ƴ under Linux Ŝ NJNJ ≥ NJ Y Ŝ ă ă Ⅰ ∃ Ŝ

1.9.3 How to Reset KM0/KM4 under Debug Mode?

Steps to reset KM0 and KM4 are different.

C ≥ NJ Y a л Σ è ≥ dz ƴ Ŝ Ŝ Ř ű ≥ dz ă Ŝ ű K Ŝ ά Y ≥ ƴ ʎ ű ≥ NJ NJ Ŝ ă Ŝ ű ἐ ʎ ƴ ă ű NJ dz Ǒ ű ʎ ≥ ƴ the path of ʎ ƴ Ǒ this file is under \project\realtek_amebaD_va0_example\GCC-RELEASE\project_lp\asdk\gnu_utility\gnu_script\rtl_gdb_debug.txt. Please Ǒ K Ⅰ ƴ ∃ Ŝ ű K Ŝ Ř Ŝ Ŧ Ⅰ dz ƒ ű ă Ŝ ű ű ʎ ƴ ∃ ǿ K ʎ. Ǒ K ʎ ă ά Ⅰ Y ≥ ƴ ʎ ű ≥ NJ NJ Ŝ ă Ŝ ű м ἐ ű ≥ ά Y ≥

C ≥ NJ Y a п Σ è ≥ dz ƴ Ŝ Ŝ Ř ű ≥ dz ă Ŝ ű K Ŝ ά Y ≥ ƴ ʎ ű ≥ NJ NJ Ŝ ă Ŝ ű ἐ ʎ ƴ ă ű NJ dz the process ʎ ƴ Ǒ of KM0). Then, set the bit[25] of memory address 0x4800_03f8 to 1 because the boot process of KM4 is controlled by the KM0. Without ű K ʎ ă ≥ LJ Ŝ NJ Ⅰ ű ʎ ≥ ƴ Σ ű K Ŝ Y a п Ǒ Ⅰ ƴ Ч ű ƨ dz Y LJ ≥ dz ű ≥ Ŧ ű K Ŝ ō ≥ ≥ ű Ř Ŝ dz ∃ ű ʎ ≥ ƴ Ф instruction, a debug window pops up. In this window. Set bit[6 ű ≥ м Σ ű K Ŝ ƴ ű è LJ Ŝ Fig1-32. For more Ŝ Y a п ʎ ă information.

Note: Only resetting KM0 may cause KM4 to work in an abnormal way because KM0 boot will change some settings of KM4. If you find KM4 Ř ≥ Ŝ ă ƴ Ω ű ǿ ≥ reboot, you should reset KM4.

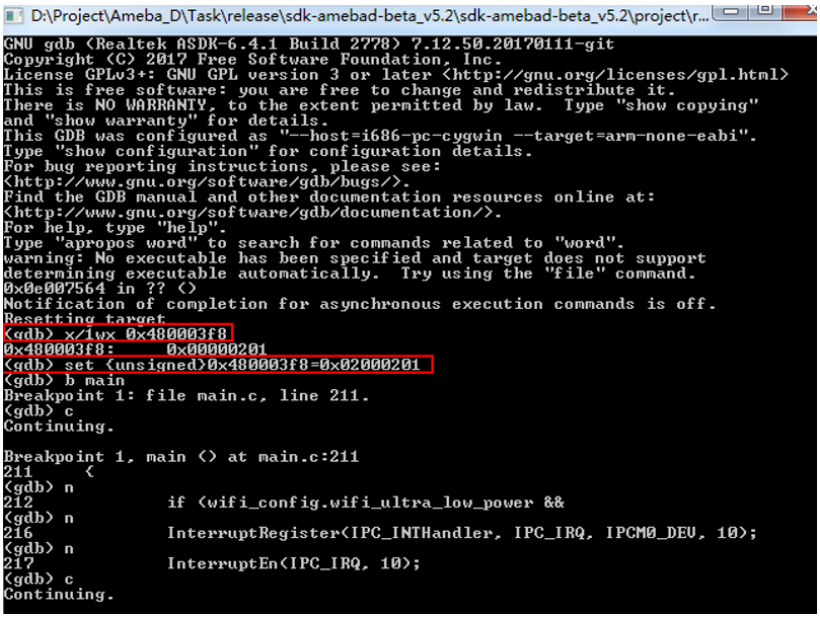


Fig1-32 KM4 通monitor reset設setting

2 SDK Architecture

The architecture of SDK is shown in Fig 1-1.

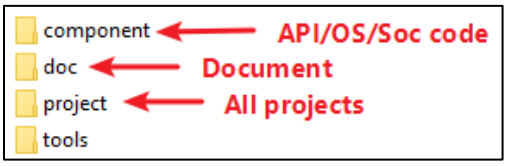


Fig2-1 SDK architecture

2.1 component

2.1.1 common

Items	Description
api	AT command Platform_stdlib.h standard library header Wi-Fi driver interface Wi-Fi promisc mode interface Wi-Fi simple configuration
application	DuerOS MQTT
audio	Audio related
bluetooth	Internal BT driver
drivers	alc5616/ alc5640/ alc5651/ alc5660/ alc5679/ alc5680/ sgtl5000 drivers IR NEC driver SDIO host driver AmebaD internal codec rl6548 driver USB host and device drivers WLAN driver
example	Utility examples: wlan_fast_connect/ ssl_download/ fatfs/ mdns/ media_geo_mp4X
file_system	Filesystem
grahpic	JPEG decoder
mbed	mbed API source code
media	Multi-media framework
network	coap dhcp http lwip mDNS rtsp sntp ssl(MBEDTLS) tftp websocket
test	WLAN test file
ui	emWin littlevGL
utilities	cJSON http_client ssl_client

	tcpecho udpecho webserver/xml
video	Video related

2.1.2 os

Items	Description
freertos	FreeRTOS source code
os_dep	osdep_service.c: Realtek encapsulating interface for FreeRTOS osdep_service.h: Realtek encapsulating interface header

2.1.3 soc

Items	Description
app	monitor and shell
bootloader	Bootloader
cmsis	ARM headers, include ARM CPU registers and operations
cmsis-dsp	ARM CMSIS-DSP source code
fwlib	Low level drivers like: UART/I2C/SPI/Timer/PWM
fwlib/usrcfg	User configuration files, maintained by user: bootcfg/trustzonecfg/sleepcfg/flashcfg/pinmapcfg
img3	Files for image3
imgtool_floader	Flash loader for image tool
misc	misc Ŧ ƛ f Ŝ f ƛ ļ Ŝ 2 ű ŀ ĸ LJ Y dz X
swlib	Standard software library supported by ROM code, like: _memcpy/_memcmp

2.2 doc

Items	Description
AN0004	Realtek low power Wi-Fi MP user guide
AN0011	Realtek WLAN simple configuration
AN0012	Realtek secure socket layer (SSL)
AN0025	Realtek AT command
AN0075	Realtek Ameba all at command v2.0
AN0096	Realtek Ameba all xmodem uart update firmware
UM0150	Realtek Ameba CoAP User Guide.pdf

2.3 tools

Items	Description
ImageTool	tools\AmebaD\Image_Tool image tool for AmebaD
autopatch	Automatically patched script
bluetooth	Tools for BT
DownloadServer	Used to send image to device based on socket by OTA function.
DownloadServer(HTTP)	Used to send image to device based on socket by OTA function.
file_check_sum	File used to check sum for Flash
probe	Tools for RLX Probe debug
serial_to_usb	Driver for serial to usb
simple_config_wizard	Tools for Wi-Fi simple configuration
simple_config_wizard_3.4	Tools for Wi-Fi simple configuration
iperf.exe	iperf for Wi-Fi performance test

2.4 GCC Project for KM4

The architecture of KM4 project is shown in Fig 2-2.

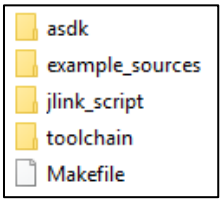


Fig 2-2 Architecture of KM4

Items	Description
asdk	Menuconfig Link script Library: lib_wlan.a/lib_wps.a/rom_symbol.x Makefile
example_sources	Peripherals driver demo code: ADC/Audio/LCD/UART/I2C
jlink_script	J-Link script for KM4
toolchain	GCC & Cygwin toolchain
Makefile	Top Makefile

2.5 Critical Header Files

Items	Description
basic_types.h	SUCCESS/FAIL/TRUE/FALSE/NULL/u8/u16/u32/BOOL
platform_stdlib.h	Standard library API: memcmp/strcpy/atoll/strpbrk/sscanf/printf/sprint/snprintf/vsnprintf
section_config.h	Section definition used in link script: IMAGE2_RAM_TEXT_SECTION
mbed api headers	component/common/mbed/ Peripheral header files for mbed APIs If you want to use mbed APIs, related headers must be included.
ameba_soc.h	Peripheral header files for raw APIs Raw APIs have more features that mbed APIs, mbed APIs just have basic features. If you want to use raw APIs, this header must be included.

3 GCC Makefile

3.1 KM4 Makefile Architecture

The architecture of KM4 makefile is shown in Fig 3-1.

```
project_hp\Makefile->
        project_hp\asdk\Makefile->
                project_hp\asdk\make\Makefile->
                        project_hp\asdk\make\api\Makefile
                        project_hp\asdk\make\app\Makefile
                        project_hp\asdk\make\application\Makefile
                        project_hp\asdk\make\audio\Makefile
                        project_hp\asdk\make\bootloader\Makefile
                        project_hp\asdk\make\cmsis\Makefile
                        project_hp\asdk\make\preject\Makefile
                                project_hp\asdk\make\project\sram\Makefile: how to build code into ram
                                project_hp\asdk\make\project\xip\Makefile: how to build code into flash
                                project_hp\asdk\make\project\library\Makefile: how to build library
                        project_hp\asdk\make\drivers\Makefile
                        project_hp\asdk\make\example\Makefile
                        project_hp\asdk\make\file_system\Makefile
                        project_hp\asdk\make\mbed\Makefile
                        project_hp\asdk\make\mbedtls\Makefile
                        project_hp\asdk\make\network\Makefile
                        project_hp\asdk\make\os\Makefile
                        project_hp\asdk\make\rtl_bluetooth\Makefile
                        project_hp\asdk\make\ssl\Makefile
                        project_hp\asdk\make\tartget\Makefile
                        project_hp\asdk\make\utilities\Makefile
                        project_hp\asdk\make\utilities_example\Makefile
                        project_hp\asdk\make\wlan\Makefile
                        project_hp\asdk\make\wps\Makefile
                        project_hp\asdk\make\utilities_example\Makefile
```

Fig 3-1 KM4 Makefile architecture

3.2 How to Build Code into Flash

Makefile in project/xip is an example for how to build code into Flash.

```
include $(MAKE_INCLUDE_GEN)

.PHONY: all clean

#*******************************************************************************#
#                                 VARIABLES                                     #
#*******************************************************************************#
#add your include path here
IFLAGS += #-I$(BASEDIR)/component/common/network/coap/include

#set your source code path here
CUSTOMER_DIR = $(ROOTDIR)/make/project/xip

#*******************************************************************************#
#                                 Source FILE LIST                              #
#*******************************************************************************#
#add your source code here
OBJS += \
                $(CUSTOMER_DIR)/xip_test.o\

#*******************************************************************************#
#                                 Include Dependency                            #
#*******************************************************************************#
-include $(OBJS:.o=.d)


#*******************************************************************************#
#                            RULES TO GENERATE TARGETS                          #
#*******************************************************************************#
all: CORE_TARGETS COPY_RAM_OBJS
#*******************************************************************************#
#               GENERATE OBJECT FILE
#*******************************************************************************#
CORE_TARGETS: $(OBJS)

#*******************************************************************************#
#               CLEAN GENERATED FILES                                           #
#*******************************************************************************#
clean: CLEAN_OBJS
        $(REMOVE) *.o
        $(REMOVE) *.i
        $(REMOVE) *.s
        $(REMOVE) *.d

-include $(DEPS)
```

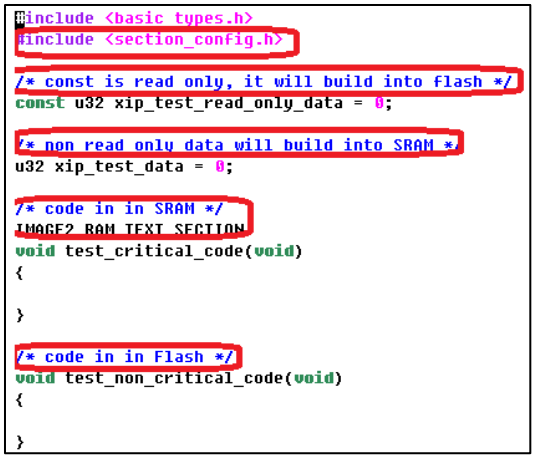
3.3 How to Build Code into SRAM

Makefile inproject/SRAMs an example for how to build code into SRAM

```
include $(MAKE_INCLUDE_GEN)


.PHONY: all clean

#*******************************************************************************#
#                                  VARIABLES                                    #
#*******************************************************************************#
#add your include path here
IFLAGS += #-I$(BASEDIR)/component/common/network/coap/include

#set your source code path here
CUSTOMER_DIR = $(ROOTDIR)/make/project/sram

#*******************************************************************************#
#                                  Source FILE LIST                             #
#*******************************************************************************#
#add your source code here
OBJS += \
                $(CUSTOMER_DIR)/ram_test.o\

#*******************************************************************************#
#                                  Include Dependency                           #
#*******************************************************************************#
-include $(OBJS:.o=.d)


#*******************************************************************************#
#                            RULES TO GENERATE TARGETS                          #
#*******************************************************************************#
all: CORE_TARGETS RENAME_CODE2SRAM COPY_RAM_OBJS
#*******************************************************************************#
#               GENERATE OBJECT FILE
#*******************************************************************************#
CORE_TARGETS: $(OBJS)

%.o:%.c
        $(CC) $(GLOBAL_CFLAGS) $(MODULE_IFLAGS) $<  -o $@

#*******************************************************************************#
#               CLEAN GENERATED FILES                                           #
#*******************************************************************************#
clean: CLEAN_OBJS
        $(REMOVE) *.o
        $(REMOVE) *.i
        $(REMOVE) *.s
        $(REMOVE) *.d

-include $(DEPS)
```

3.4 How to Use Section Attribute?

Because SRAM space is limited, we suggest you build critical code/data into SRAM and let other code/data left in Flash. You should use section



```
#include <basic_types.h>
#include <section_config.h>

/* const is read only, it will build into flash */
const u32 xip_test_read_only_data = 0;

/* non read only data will build into SRAM */
u32 xip_test_data = 0;

/* code in in SRAM */
IMAGE2_RAM_TEXT_SECTION
void test_critical_code(void)
{

}

/* code in in Flash */
void test_non_critical_code(void)
{

}
```

Note:

You ă K ƨ dz f Ř λ y Ŏ f dz Ř Ŝ ά ă Ŝ Ŏ ű λ ƨ y ψ Ŏ ƨ y ∓ λ ∃ Φ K έ
const/read-only data will build into Flash
Non read-only data will build into S ! a ó L ∓ è ƨ dz y Ŝ Ŝ Ř NJ Ŝ ŀ Ř ƨ y f è Ř ŀ ű ŀ ō dz λ f Ř λ y ű ƨ { w ! a
Functions limited by IMAGE2_RAM_TEXT_SECTION will build into SRAM
Functions not limited by IMAGE2_RAM_TEXT_SECTION will build into Flash

3.5 How to Build Library?

Makefile in project/library is an example for build user library

```
include $(MAKE_INCLUDE_GEN)

.PHONY: all clean

#*******************************************************************************#
#                                 VARIABLES                                     #
#*******************************************************************************#
#add your include path here
IFLAGS += #-I$(BASEDIR)/component/common/network/coap/include

#set your source code path here
CUSTOMER_DIR = $(ROOTDIR)/make/project/library

#*******************************************************************************#
#                                 Source FILE LIST                              #
#*******************************************************************************#
#add your source code here
OBJS += \
                $(CUSTOMER_DIR)/lib_user_test.o\

#*******************************************************************************#
#                                 Dependency                                    #
#*******************************************************************************#
-include $(OBJS:.o=.d)

#*******************************************************************************#
#                           RULES TO GENERATE TARGETS                           #
#*******************************************************************************#

# Define the Rules to build the core targets
all: CORE_TARGETS COPY_RAM_OBJS
        $(AR) rvs lib_user.a $(OBJS) $(OBJS_SRAM)
        $(MOVE) -f lib_user.a $(ROOTDIR)/lib/application

#*******************************************************************************#
#               GENERATE OBJECT FILE                                            #
#*******************************************************************************#
CORE_TARGETS:   $(OBJS)

#*******************************************************************************#
#                           RULES TO CLEAN TARGETS                              #
#*******************************************************************************#
clean: CLEAN_OBJS
        $(REMOVE) *.o
        $(REMOVE) *.i
        $(REMOVE) *.s
        $(REMOVE) *.d
```

3.6 How to Add Library?

Open project_hp\asdk\Makefile, and add lib_user.a into LINK_APP_LIB

```
LINK_APP_LIB+=$(ROOTDIR)/lib/application/lib_user.a
```

4 C++ Standards Supported in GCC

4.1 Introduction

This chapter mainly introduces how to build C++ codes in AmebaD GCC project.

Note: "iostream" is not available currently.

4.2 To Compile C++ Codes

The following steps are necessary to support C++ in current GCC project.

(1) Modify the link script rlx8721d_img2_is.ld(non-security) or rlx8721d_img2_tz.ld(security)

```
.xip_image2.text :
{
        __flash_text_start__ = .;

        *(.img2_custom_signature*)

        *(.text*)
        *(.rodata*)

        /* Add This for C++ support */
        . = ALIGN(4);
        __preinit_array_start = .;
        KEEP(*(.preinit_array))
        __preinit_array_end = .;
        . = ALIGN(4);
        __init_array_start = .;
        KEEP(*(SORT(.init_array.*)))
        KEEP(*(.init_array))
        __init_array_end = .;
        . = ALIGN(4);
        __fini_array_start = .;
        KEEP(*(SORT(.fini_array.*)))
        KEEP(*(.fini_array))
        __fini_array_end = .;
        /*------------------*/

        . = ALIGN (4);
        __cmd_table_start__ = .;
        KEEP(*(.cmd.table.data*))
        __cmd_table_end__ = .;

        __flash_text_end__ = .;

        . = ALIGN (16);
} > KM4_IMG2
```

```
/* Add This for C++ support */
.ARM.extab   :
{
        *(.ARM.extab* .gnu.linkonce.armextab.*)

} > KM4_IMG2

.ARM.exidx :
{
        __exidx_start = .;
        *(.ARM.exidx* .gnu.linkonce.armexidx.*)
        __exidx_end = .;
        end = .;
} > KM4_IMG2

__wrap_printf = 0x1010a3f5;
__wrap_sprintf = 0x1010a471;
__wrap_strcat = 0x10111635;
__wrap_strchr = 0x10111675;
__wrap_strcmp = 0x10111745;
__wrap_strncmp = 0x101118f9;
__wrap_strlen = 0x10111839;
__wrap_strnlen = 0x10111a05;
__wrap_strncat = 0x1011189d;
__wrap_strpbrk = 0x10111a39;
__wrap_strstr = 0x10111d25;
__wrap_strtok = 0x1011201d;
__wrap_strsep = 0x10111a65;
__wrap_strtoll = 0x10111ffd;
__wrap_strtoul = 0x101122e9;
__wrap_strtoull = 0x10111f3d;
__wrap_atoi = 0x101115e1;
__wrap_strcpy = 0x101117b9;
__wrap_strncpy = 0x1011199d;
__wrap_memset = 0x10110ea1;
__wrap_memcpy = 0x10110d2d;
__wrap_memcmp = 0x10110cc9;
__wrap_memmove = 0x10110dd9;
__wrap_snprintf = 0x1010a49d;
__wrap_malloc = pvPortMalloc;
__wrap_realloc = pvPortReAlloc;
__wrap_free = vPortFree;
/*-----------------*/
```

(2) Modify startup codertl8721dhp_app_start.c

```
#if defined (__GNUC__)
/* Add This for C++ support to avoid compile error */
void _init(void){}
#endif

// The Main App entry point
void app_start(void)
{
        simulation_stage_set(SIMULATION_KM4_CPUID, BIT_KM4_APP_ENTER);

        SOCPS_InitSYSIRQ_HP();

        __NVIC_SetVector(SVCall_IRQn, (VOID*)vPortSVCHandler);
        __NVIC_SetVector(PendSV_IRQn, (VOID*)xPortPendSVHandler);
        __NVIC_SetVector(SysTick_IRQn, (VOID*)xPortSysTickHandler);

#if defined ( __ICCARM__ )
        __iar_cstart_call_ctors(NULL);
#endif

#if defined (__GNUC__)
        /* Add This for C++ support */
        __libc_init_array();
#endif

        // force SP align to 8 byte not 4 byte (initial SP is 4 byte align)
        __asm(
                "mov r0, sp\n"
                "bic r0, r0, #7\n"
                "mov sp, r0\n"
        );

        main();
}
```

(3) Modify makefile

project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\Makefile

```
linker_image2_ns:
        @echo "========= linker img2_ns start ========="
        $(LD) $(LD_ARG) target_img2.axf $(RAM_OBJS_LIST) $(LINK_ROM_LIB_NS) $(LINK_APP_LIB) $(IMAGE_TARGET_FOLDER)/cmse_implib.lib
-lm -lc -lnosys -lgcc -lstdc++
```

project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\Makefile.include.gen

```
#GLOBAL_CFLAGS += -specs nosys.specs
#GLOBAL_CFLAGS += -fno-short-enums
GLOBAL_CFLAGS += -Wextra

GLOBAL_CFLAGS += $(IFLAGS)
GLOBAL_CFLAGS += -DCONFIG_PLATFORM_8721D
GLOBAL_CFLAGS += -DCONFIG_USE_MBEDTLS_ROM_ALG
GLOBAL_CFLAGS += -DCONFIG_FUNCION_00_OPTIMIZE
GLOBAL_CFLAGS += -DDM_ODM_SUPPORT_TYPE=32

CFLAGS = $(GLOBAL_CFLAGS)
CFLAGS += -Wstrict-prototypes

CPPFLAGS = $(GLOBAL_CFLAGS)
CPPFLAGS += -std=c++11 -fno-use-cxa-atexit
```

```
LD_ARG += -nostartfiles
LD_ARG += -specs nosys.specs
LD_ARG += -Wl,--gc-sections
LD_ARG += -Wl,--warn-section-align
LD_ARG += -Wl,-Map=text.map
LD_ARG += -Wl,--cref
LD_ARG += -Wl,--build-id=none
LD_ARG += -save-temps

LD_ARG += -Wl,-wrap,strcat  -Wl,-wrap,strchr   -Wl,-wrap,strcmp
LD_ARG += -Wl,-wrap,strncmp -Wl,-wrap,strcpy   #-Wl,-wrap,strnicmp
LD_ARG += -Wl,-wrap,strncpy -Wl,-wrap,strlen   #-Wl,-wrap,strlcpy
LD_ARG += -Wl,-wrap,strnlen -Wl,-wrap,strncat  -Wl,-wrap,strpbrk
LD_ARG += -Wl,-wrap,strstr   -Wl,-wrap,strtok  #-Wl,-wrap,strspn
LD_ARG += -Wl,-wrap,strsep  #-Wl,-wrap,strxfrm  -Wl,-wrap,strtod
LD_ARG += -Wl,-wrap,strtoll #-Wl,-wrap,strtof  -Wl,-wrap,strtold
LD_ARG += -Wl,-wrap,strtoul -Wl,-wrap,strtoull -Wl,-wrap,atoi
LD_ARG += #-Wl,-wrap,atoui   -Wl,-wrap,atol     -Wl,-wrap,atoul
LD_ARG += #-Wl,-wrap,atoull  -Wl,-wrap,atof
LD_ARG += -Wl,-wrap,malloc   -Wl,-wrap,free     -Wl,-wrap,realloc
LD_ARG += -Wl,-wrap,memcmp  -Wl,-wrap,memcpy
LD_ARG += -Wl,-wrap,memmove -Wl,-wrap,memset
LD_ARG += -Wl,-wrap,printf  -Wl,-wrap,sprintf
LD_ARG += -Wl,-wrap,snprintf  #-Wl,-wrap,vsnprintf
```

```
#*******************************************************************************#
#             RULES TO GENERATE OBJECT FILE FROM .CPP FILE                      #
#*******************************************************************************#
%.o:%.cpp
        $(CC) $(CPPFLAGS) $< -o $@
```

(4) Add C++ files into project

We have tested a small program and provide it as an example, which can be found in main.cpp.

(5) Use command make in console to compile project with C++ files.

5 GCC Standard Library

5.1 Introduction

This chapter mainly introduces how to use GCC standard library in AmebaD.

To save flash memory and improve efficiency, ROM code has implemented a standard software library like _memcpy and _memcmp, and some functions are simplified version in ROM software library, such as printf and sprintf.

There are two methods when using standard library functions such as printf and sprintf in GCC:
- Using ROM software library.
- Using GCC standard library.

5.2 Default Use of Library Function

In current SDK, the default use of library function is illustrated in Fig 5-1 and Table 5-1

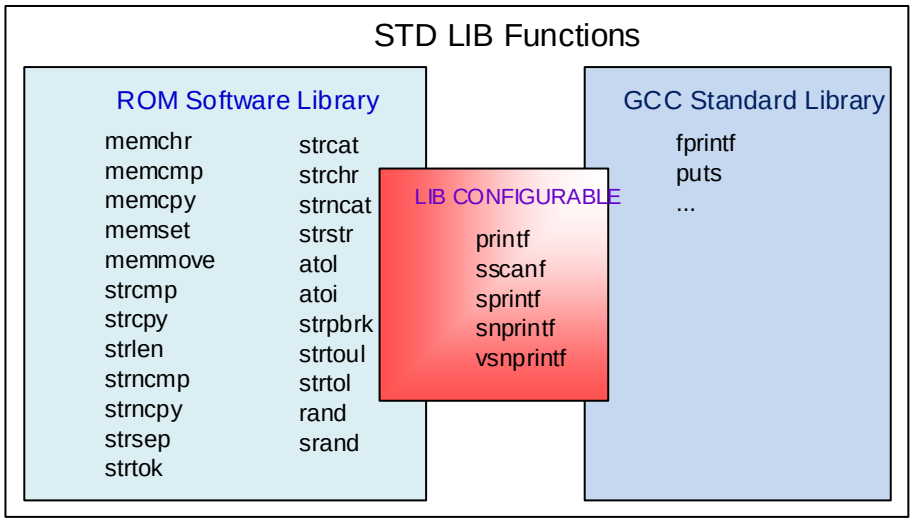


Fig 5-1 Library function guide

Table 5-1 Default use of library function

Item	Function
Using ROM software library	memchr/ memcmp/ memcpy/ memset/ memmove/ strcmp/ strcpy/ strlen/ strncmp/ strncpy/ strsep/ strtok/ strcat /strchr/ strncat/ strstr/ atoll/ atoi/ strpbrk/ strtoul/ strtol/ rand/ srand
Using GCC standard library	fprintf/ puts/ X
Library is configurable	printf/ sprintf/ snprintf/ vsnprintf/ sscanf Note: The default library is ROM software library, you can switch to GCC standard library by defining macro STD_PRINTF

Because printf/ sprintf/ snprintf/ vsnprintf/ sscanf in ROM software library are simplified version compared with these in GCC standard library, they only support a few formats. In order to remind users that unsupported format occurs when using these functions in ROM library, SDK wraps up these functions.

Table 5-2 Wrapper functions

Item	Wrapper Function
printf	_rtl_printf

sprintf	_rtl_sprintf
snprintf	_rtl_snprintf
vsnprintf	_rtl_vsnprintf
sscanf	_rtl_sscanf

When using printf/sprintf/snprintf/vsnprintf/sscanf in GCC standard library, the memory size will increase 40KB compared to using these functions in ROM software library.

5.3 To Use Configurable Function in GCC Standard Library

Compared to printf/sprintf/snprintf/vsnprintf/sscanf functions in GCC library, they only support a few formats in ROM library. The following content takes printf as an example.

Table 5-3 printf supported format

Library	printf Supported Format
ROM software library	%s, %x, %X, %p, %P, %d, %c
GCC standard library	%s, %x, %X, %p, %P, %d, %c, %L, %l, %u, %U, %e, %E,

For printf/sprint/snprintf/vsnprintf/sscanf, ROM library is linked by default. If format not support! log dumps out in trace tool. It means that unsupported format occurs, you should link these functions to GCC standard library.

To use printf/sprint/snprintf/vsnprintf/sscanf in GCC standard library, follows the steps:

(1) Add #define STD_PRINTF in the front.

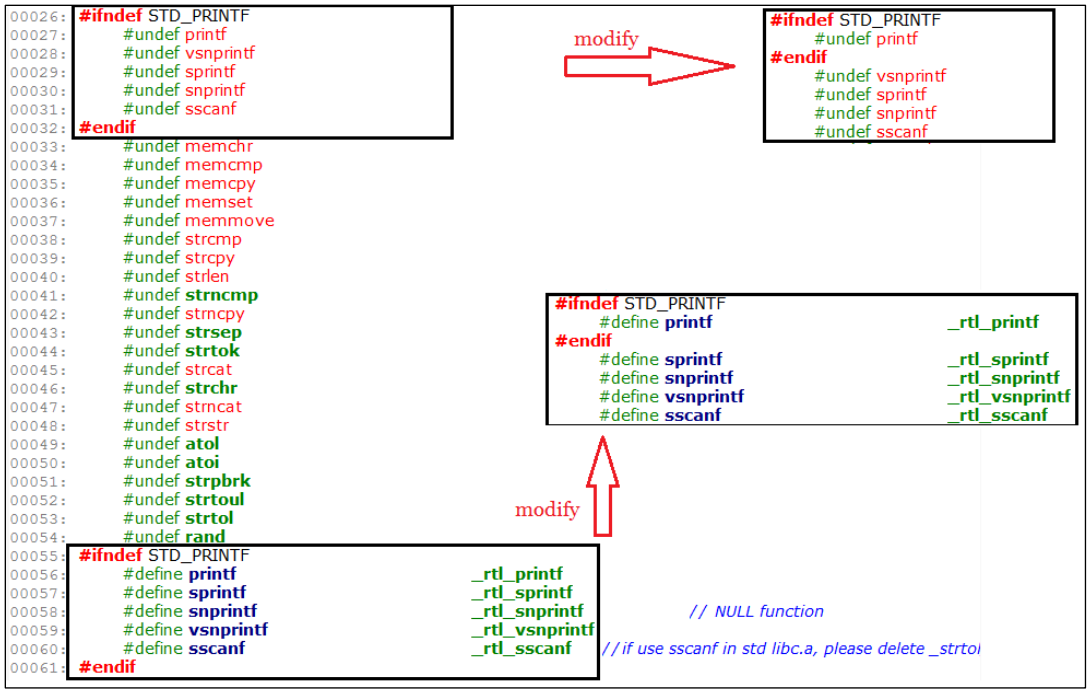
There are two cases:

Item	Operation	Description

case 1	Add #define STD_PRINTF in the front of c files, and included "platform_stdlib.h"	Change printf/sprint/snprintf/vsnprintf/sscanf of some files to link to GCC standard library
		Example:
		// main.c
		#define STD_PRINTF
		#include "main.h"
		#include "platform_stdlib.h"
		int main(void)
		{
		printf("%f\n", 1.2);
		printf("%u\n", 20);
		}

case 2	Add #define STD_PRINTF in the front of "platform_stdlib_rtl8721d.h"	Change printf/sprint/snprintf/vsnprintf/sscanf of all SDK to link to GCC standard library

(2) Modify "platform_stdlib_rtl8721d.h" file if you don't want to switch to use all the five functions in GCC standard library
In SDK, macro STD_PRINTF controls printf/sprintf/snprintf/vsnprintf/sscanf at the same time. If you only want to configure some but not all, for example, you only want to use printf in GCC standard library, and other four functions sprintf/snprintf/vsnprintf/sscanf maintain the default state, you need to modify "platform_stdlib_rtl8721d.h" file.



5.4 Tips

If you wants to use GCC standard library, we recommend only link some user specified c files to GCC standard library and link all SDK to GCC standard library, because printf/sprint/snprintf/vsnprintf/sscanf in our SDK should link to ROM standard library.
If using printf in GCC standard library libc.a, and there is no "\n" in the end, use fflush(stdout) after printf to dump log in cache. For example:

```
printf("hello");
fflush(stdout);
```

In current GCC project, KM0 does not support floating-point temporarily, so printf cannot dump %f in KM0 in default setting.
If using sscanf in GCC standard library libc.a, delete _strtol_r and _strtoul_r symbols in rlx8721d_rom_symbol_acut.ld

6 IAR Build Environment Setup

This chapter illustrates how to setup IAR development environment for Realtek Ameba SDK, including building projects, downloading images and debugging.

6.1 Requirement

6.1.1 IAR Embedded Workbench

L ! w LJNJ2 Ø ƛ Ř Ŝ ă l· y˝ L 5 9 Ŝ y˝ Ø ƛ NJ 2 y˝ Y Ŝ y˝ ú ₸ 2 NJ Ŏ 2 Ř Ŝ ō dz ƛ f Ř ƛ y˝ Ǝ http://www.iar.com/, Ř ƛ y˝ Ǝ Σ and a trail version is available for 30 days.

Note: To support ARMv8M with Security Extension (AmebaD HS CPU, also called KM4), IAR version must be 8.30 or higher.

6.1.2 J-Link or RLX Probe

If you need to download images or debug code for AmebaD with IAR, then a J-Link adapter or a RLX Probe is necessary.

Note: For AmebaD CPU, the J-Link version must be 9.0 or higher.

6.2 Hardware Configuration

The hardware block diagram of AmebaD demo board is shown in Fig 6-1.

USB TO UART: supply power and print logs, baud rate is 115200 bps.
SWD: SWD interface, used for image downloading and debugging with IAR.
Reset button: reset AmebaD to run firmware after IAR completes downloading.

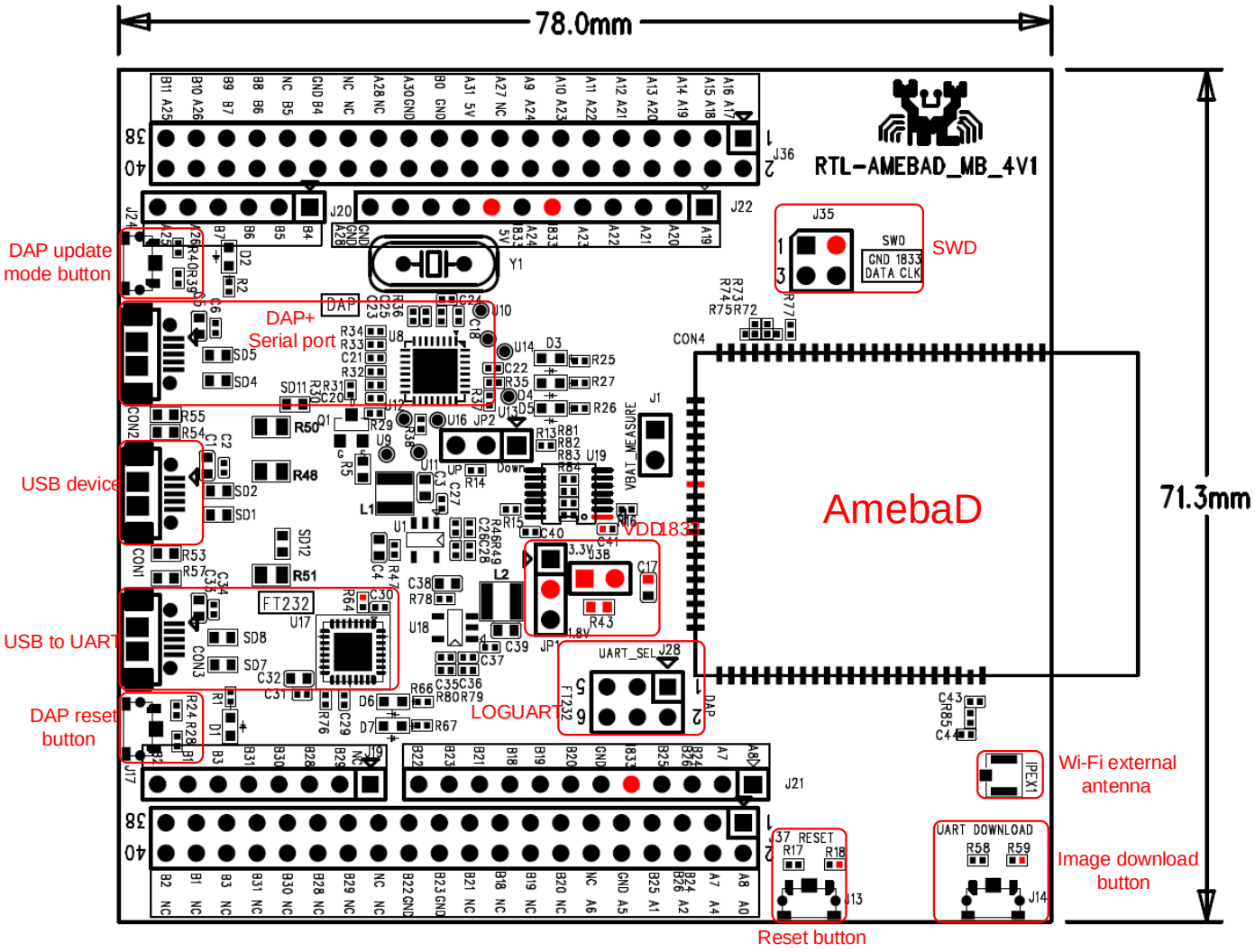


Fig6-1 AmebaD demo board

6.2.1 Connecting with J-Link

Refer to Fig6-2 and Fig6-3 to connect AmebaD SWD interface with J-Link.

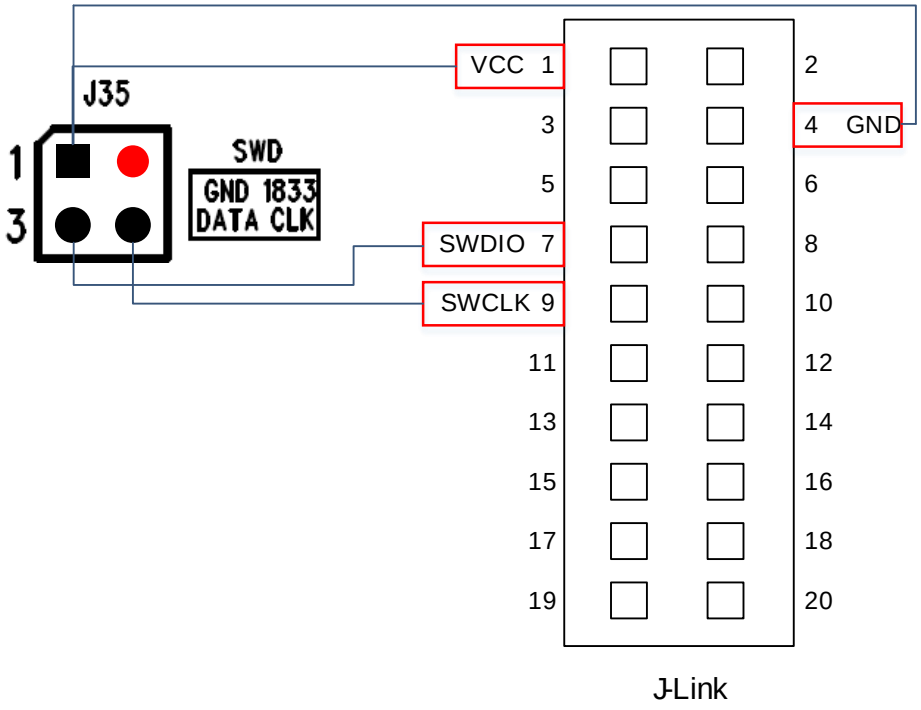


Fig6-2 J-Link and SWD connection diagram

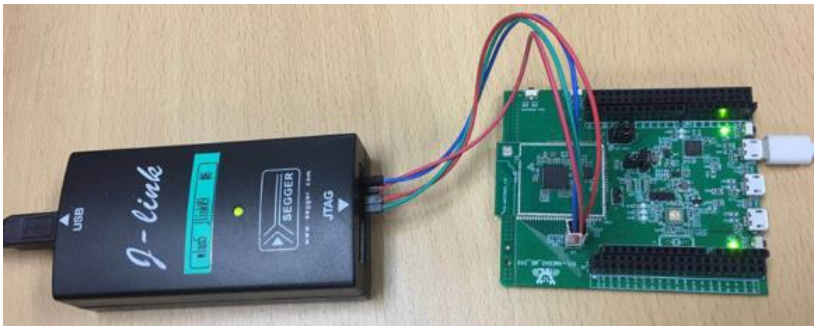


Fig6-3 J-Link and AmebaD SWD connection

6.2.2 Connecting with RLX Probe

Refer to Fig6-4 and Fig6-5 to connect AmebaD SWD interface with RLX probe.

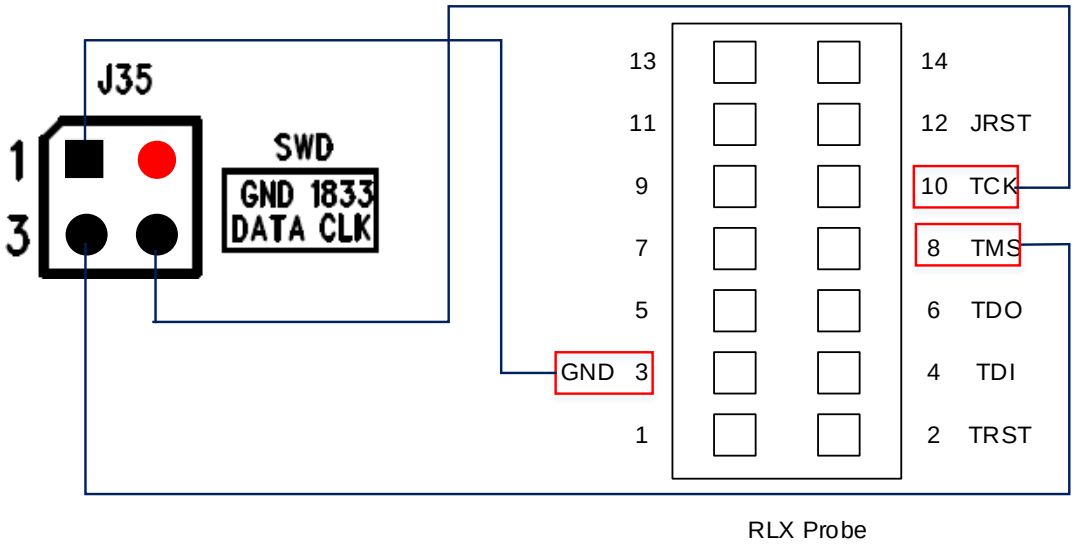


Fig6-4 RLX Probe and SWD connection diagram

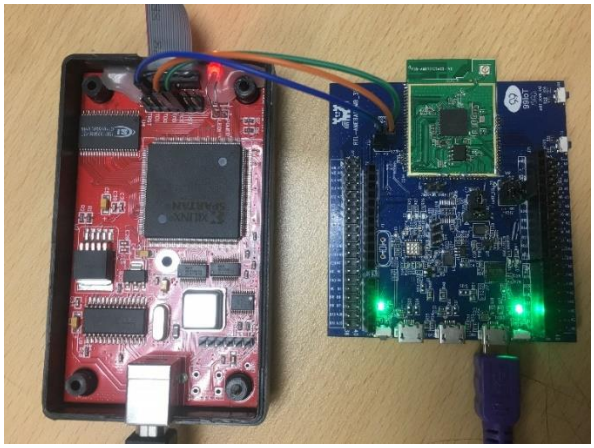


Fig6-5 RLX Probe and AmebaD SWD connection

6.3 How to Use IAR SDK?

6.3.1 IAR Project Introduction

Because AmebaD is a dual-core CPU platform, two workspaces are provided to build for each core in project\realtek_amebaD_va0_example\EWARM-RELEASE

Project_lp_release.eww (KM0 workspace) contains the following projects:
- km0_bootloader
- km0_application

Project_hp_release.eww (KM4 workspace) contains the following projects:
- km4_bootloader
- km4_application
- km4_secure

Each project in KM4 workspace has different build configurations, as Table 6-1 shows.

Table 6-1 Build configurations for KM4 project

Project	Build Configuration	Configure TrustZone	Enable MP
km4_bootloader	km4_bootloader - is[1]	N	N
	km4_bootloader - tz[2]	Y	N
km4_application	km4_application - is	N	N
	km4_application - tz	Y	N
	km4_application - is (mp[3])	N	Y
	km4_application - tz (mp)	Y	Y
km4_secure	km4_secure - tz	Y	N
	km4_secure - tz (mp)	Y	Y

Note:
1. The configuration ƛ ű Ŝ Ỵ ă - ƛ ǽ ƛ ű Ķ ŀ NJ Ŝ ƛ Ǝ ỿ ƻ NJ Ŝ ă Ŝ Ŏ dz NJ Ŝ Ŏ ƻ ỿ Ŧ ƛ Ǝ dz NJ ŀ ű ƛ ƻ ỿ Ʃ ǿ Ķ ƛ Ŏ Ķ ŀ NJ Ŝ Ř
2. The Ŏ ƻ ỿ Ŧ ƛ Ǝ dz NJ ŀ ű ƛ - ǽ ỿ ἐ ƛ ű Ŝ Ŝ ă ¢ Ň dz ǽ Ķ ½ ǽ ỿ Ŝ Ŏ ƻ ỿ Ŧ ƛ Ǝ dz NJ ŀ ű ƛ ƻ ỿ Ʃ ǿ Ķ ƛ Ŏ Ķ ŀ NJ Ŝ Ř Ŝ ă ƛ Ǝ
3. The configuration items ǿ ƛ ű Ķ ά Ỵ LJ ἐ ŀ NJ Ŝ Ỵ ŀ ă ă LJNJ ƻ Ř dz Ŏ ű ƛ ƻ ỿ Ŏ ƻ ỿ Ŧ ƛ Ǝ dz NJ ŀ ű ƛ ƻ ỿ Ʃ ǿ Ķ ƛ Ŏ Ķ

For applications that do not use TrustZone, users should apply ignore secure configurations as Table 6-2 shows. The km4_secure project which contains Trustzone-protected code, is not used.
For applications that use TrustZone, users should apply TrustZone configurations as Table 6-2 shows.

Table 6-2 Configurations for project with/without TrustZone

Project	TrustZone	Normal Image	MP Image
km4_bootloader	N	km4_bootloader - is	km4_bootloader - is
	Y	km4_bootloader - tz	km4_bootloader - tz
km4_application	N	km4_application - is	km4_application - is (mp)
	Y	km4_application - tz	km4_application - tz (mp)
km4_secure	Y	km4_secure - tz	km4_secure - tz (mp)

At the top of the Workspace window, there is a drop-down list where you can choose a build configuration for a specific project.

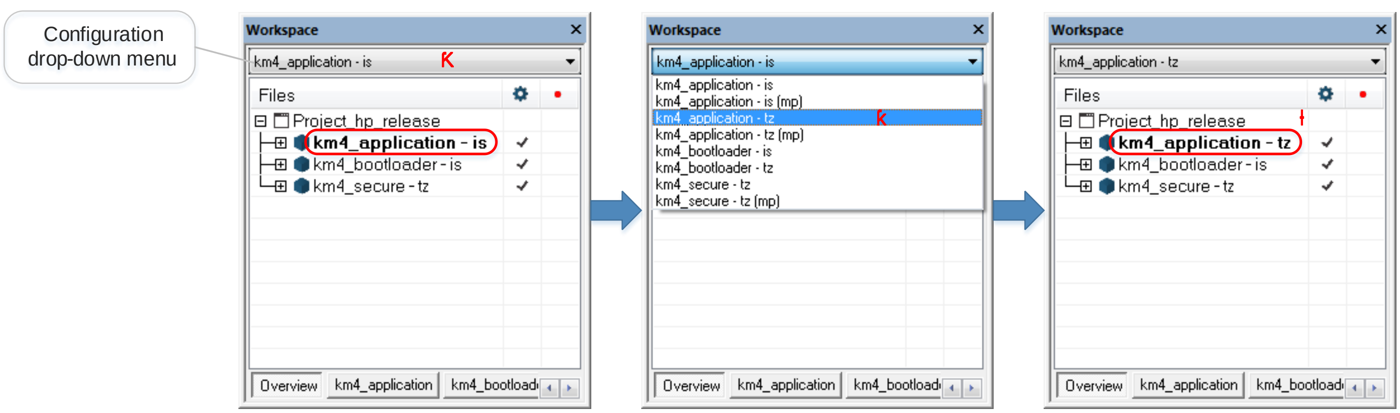


Fig 6-6 How to choose a build configuration

6.3.2 IAR Build

When building SDK for the first time, you should build both KM0 project and KM4 project. Other times, you only need to rebuild the modified project.

6.3.2.1 Building KM0 Project

The following steps show how to build KM0 project:
(1) Open project\realtek_amebaD_va0_example\EWARM-RELEASE\Project_lp_release.eww

(2) Make sure km0_bootloader and km0_application are in Workspace. Click Project > Options > General Options > Target > Processor Variant > Core, verify the CPU configurations according to Fig 6-7.

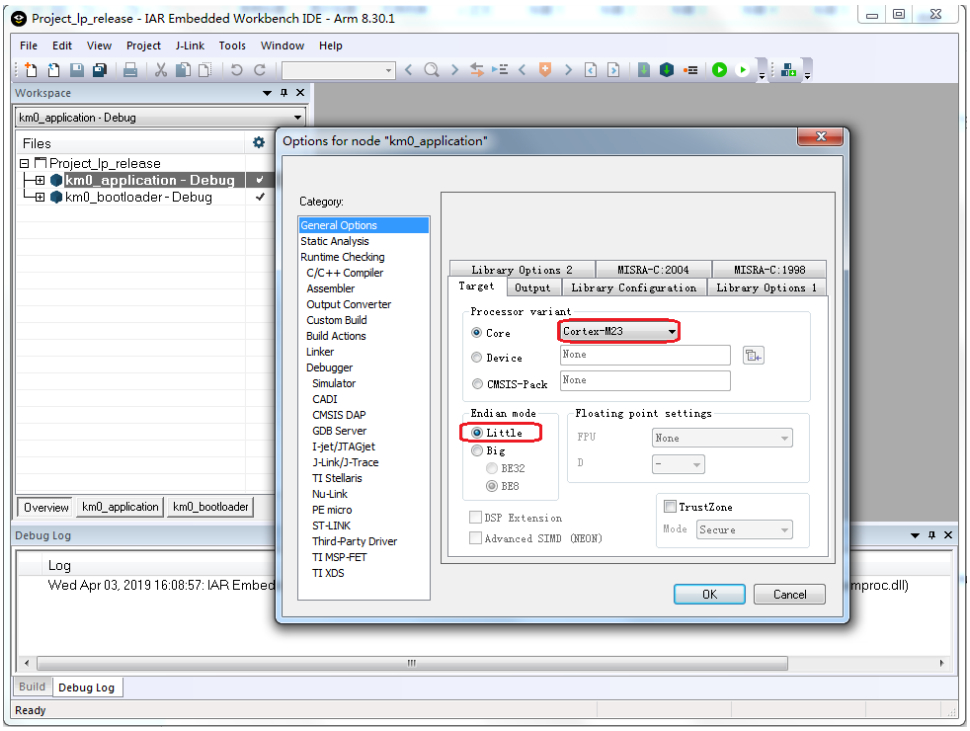
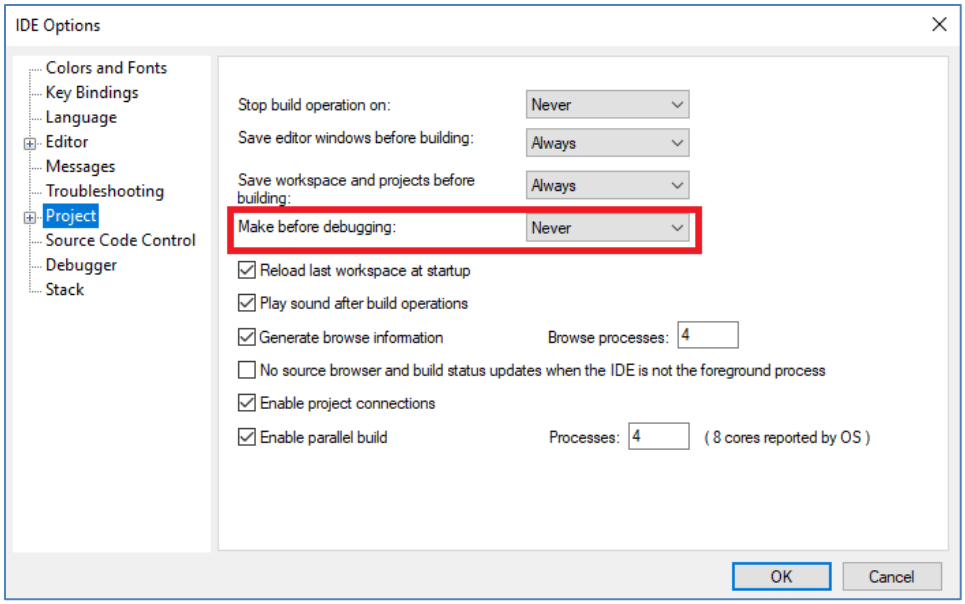


Fig6-7 KM0 processor options

Note: If you have installed IAR version 9.xx or above from official website, click Tools > Option > Project, and make ensure the setting á a l· ļ Ŝ ō Ŝ Ŧ 2 NJ Ŝ Ř Ŝ ō dz Ǝ Ǝ λ ƴ Ǝ é λ ă (3). Ŝ ú ú 2 á b Ŝ Ø Ŝ NJ é ō Ŝ Ŧ 2 NJ Ŝ Ŝ È Ŝ Ŏ dz ű λ ƴ Ǝ ă ú



(3) w λ Ǝ K ú Ŏ f λ Ŏ ļ ú K Ŝ LJNJ 2 2 Ŝ Fig 6-8 shows. Then KM0_bootloader and KM0_application should compile in order.

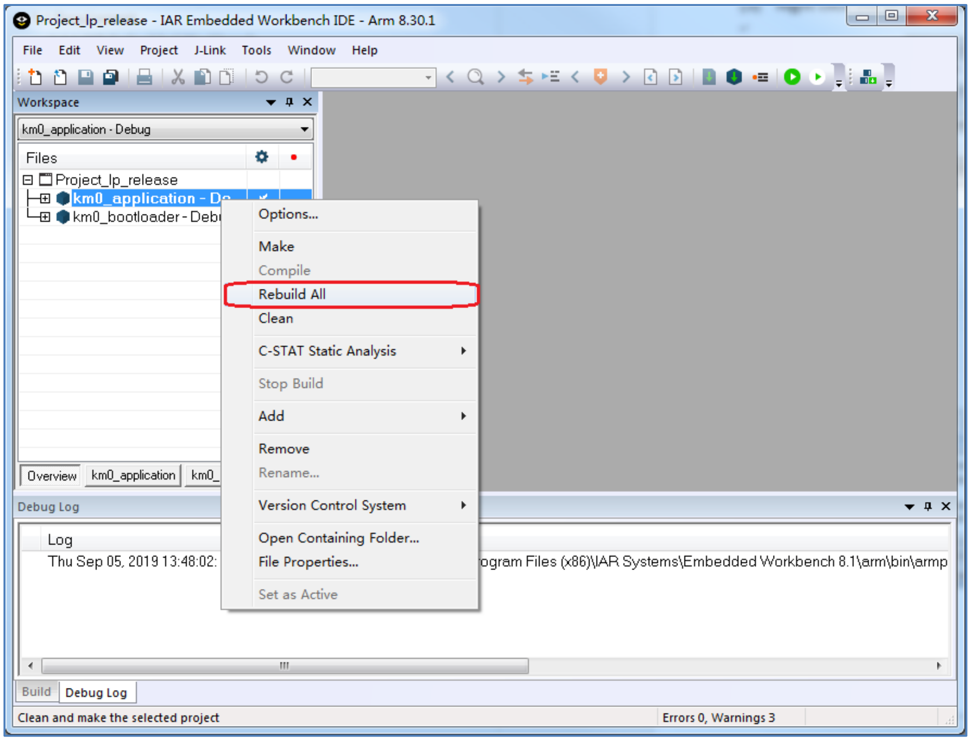
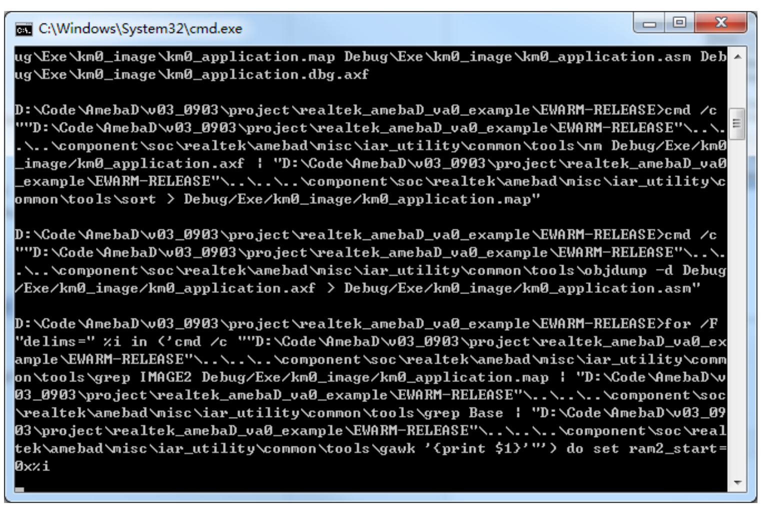


Fig6-8 Building KM0 project

Note: After building each project, IAR will pop up a command prompt window to execute post-build action to generate images from executable [illegible] build action is completed, the window will disappear automatically.



(4) After compile, the images km0_boot_all.bin and km0_image2_all.bin can be seen in project\realtek_amebaD_va0_example\EWARM-RELEASE\Debug\Exe\km0_image

6.3.2.2 Building KM4 Project

The following steps show how to build KM4 project:

(1) Open project\realtek_amebaD_va0_example\EWARM-RELEASE\Project_lp_release.eww
(2) Refer to 6.3.1 and choose the build configurations for each project according to your application.

(3) Click Project > Options, General Options > Target > Processor Variant > Core, verify the CPU configurations according to Fig 6-9.

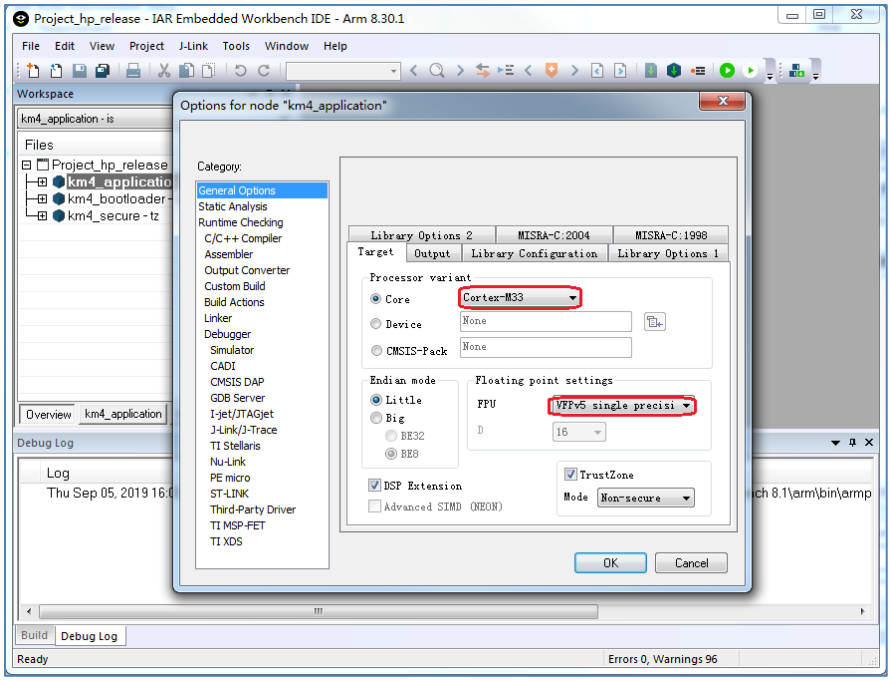
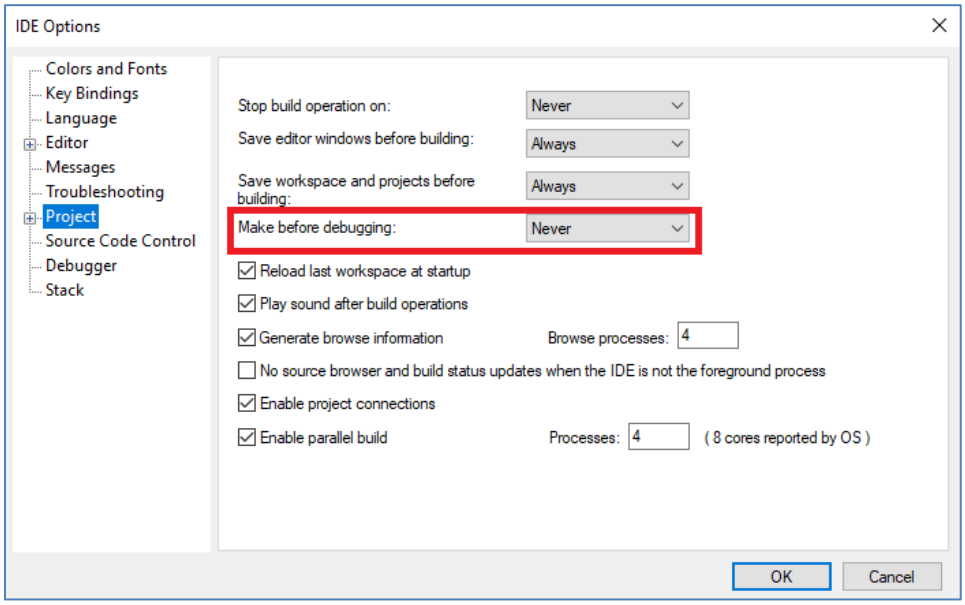


Fig6-9 KM4 processor options

Note: If you have installed IAR version 9.xx or above from official website, click Tools > Option > Project, and make sure the setting á a l· ļ Ŝ ō Ŝ Ŧ ƨ NJ Ŝ Ř Ŝ ō dz Ǝ Ǝ ƛ ƴ Ǝ ɛ ƛ ă (4).Ŝ ú ú ƨ á b Ŝ Ø Ŝ NJ ɛ ō Ŝ Ŧ ƨ NJ Ŝ Ŝ È Ŝ Ŏ dz ú ƛ ƴ Ǝ ă ú



(4) w ƛ Ǝ K ú Ŏ f ƛ Ŏ ļ ú K Ŝ LJ NJ ƨ ƨ Ŝ F ō 6-10 shows. The km4_bootloader, km4_secure and km4_application should compile in order.

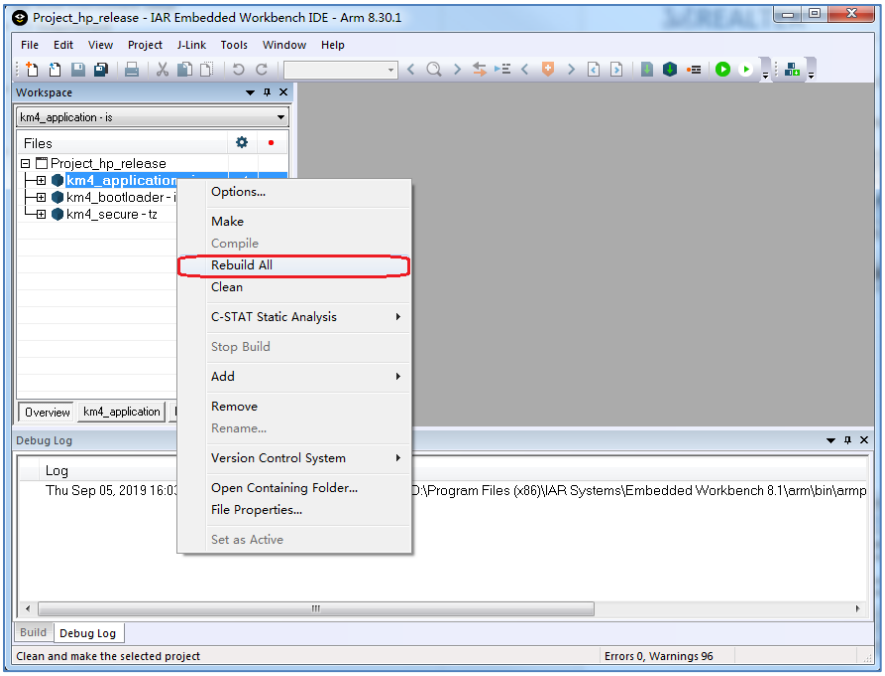
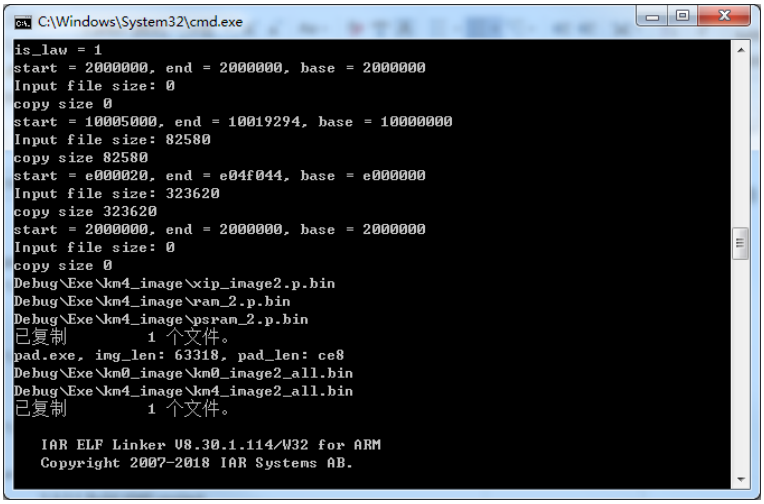


Fig6-10 BuildingKM4 project

Note:

When TrustZone is enable, the km4_secure project must be built before the km4_application project. When TrustZone is not used, there is no need to compile the km4_secure project.

After building each project, IAR will pop up a command prompt window to execute post-build action to generate images from executable files. Please wait while the building progress. After post-build action is completed, the window would disappear automatically.



(5) After compile, the images km4_boot_all.bin and km0_km4_image2.bin can be seen in project\realtek_amebaD_va0_example\EWARM-RELEASE\Debug\Exe\km4_image. For MP configurations, the km0_km4_image2_mp.bin would be generated instead.

6.3.3 IAR Download

The generated images can be downloaded in two ways:

IAR J-Link or RLX Probe SWD (introduced in the next section)

AmebaD ImageTool, refer to ImageTool for more information.

AmebaD demo board supports using J-Link and RLX Probe SWD to download and debug. Image of each project can be download individually.

Note: Considering KM4 is powered on by KM0, you should make sure that KM0 has boot up already before downloading images to KM4. Otherwise, for J-Link, J-Link [illegible] Fig6-11 shows [illegible] opened under KM4.

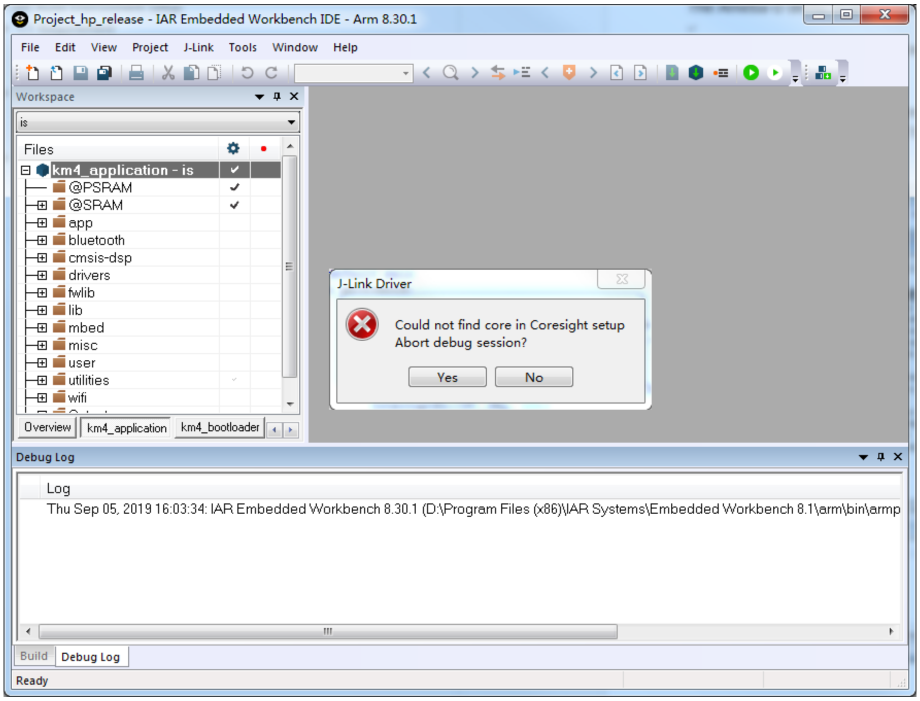


Fig6-11 J-Link cannot find KM4

As a result, if the Flash memory is empty, the sequence to download images is:
(1) Download for km0_bootloader and km0_application projects
(2) Click Reset button on demo board to make KM0 boots up
(3) Download for km4_bootloader and km4_application projects

During development, if Flash memory is not empty and KM0 can boot up successfully, then you can download updated images to KM4 directly, and there is no need to re-download for KM0.

The following steps show how to download image for the target project with IAR. If there is an error like Fig6-24 displayed in IAR window, refer to 6.3.4.3
(1) Choose the target project display in Workspace window, for example, km4_bootloader as Fig6-12 shows.
(2) If using J-link debugger, check whether the J-link debugger setting is correct
   a) Click Project > Options > Debugger > Setup > Driver [illegible] J-Link [illegible] Fig6-13 shows.
   b) Click Debugger > J-Link/J-Trace > Connection > Interface [illegible] Fig6-14 shows. [illegible]

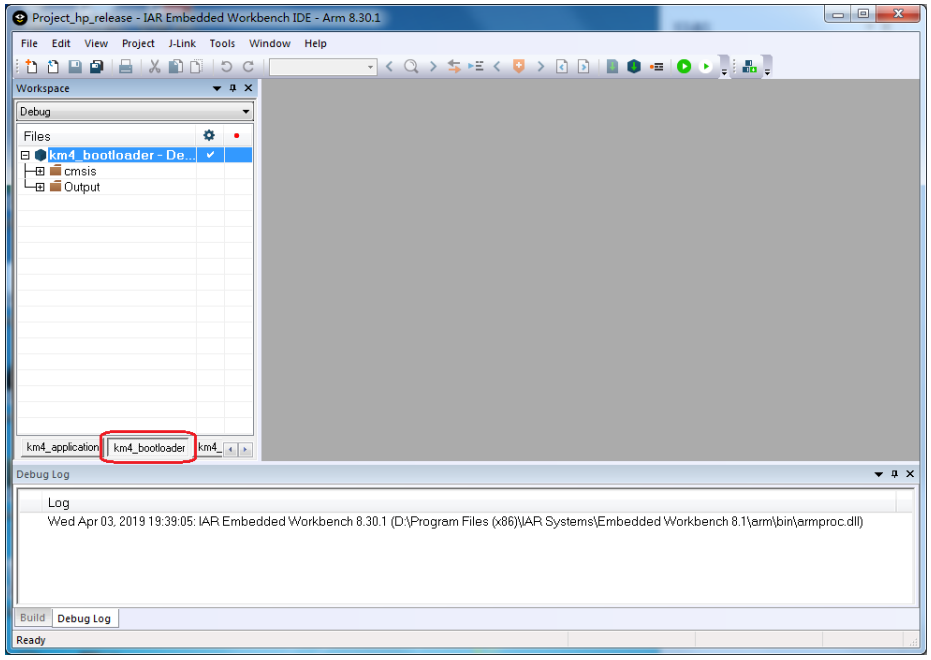


Fig6-12 Switching to the target project view

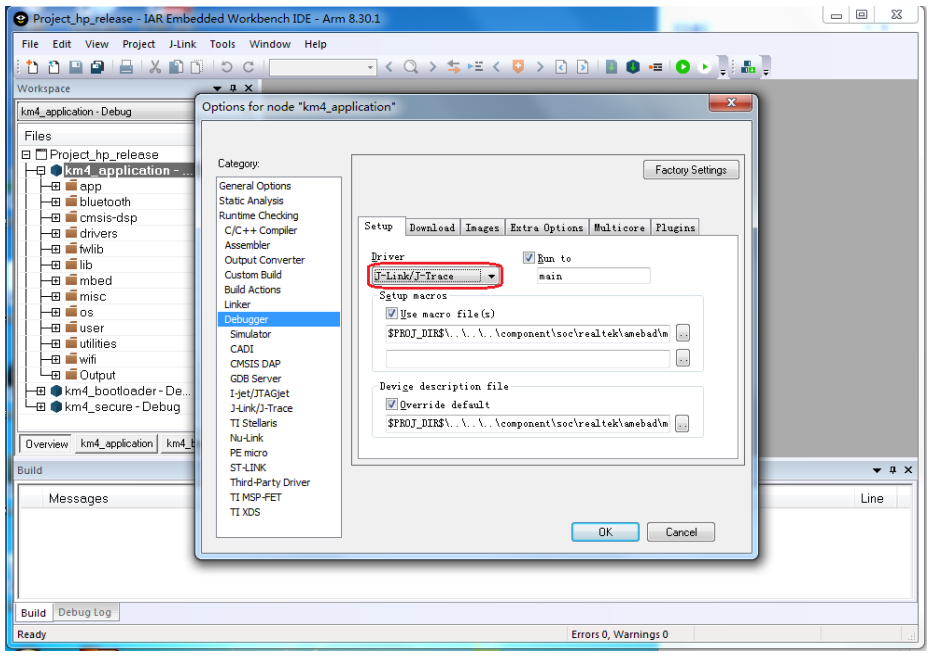


Fig6-13 J-Link debugger setup

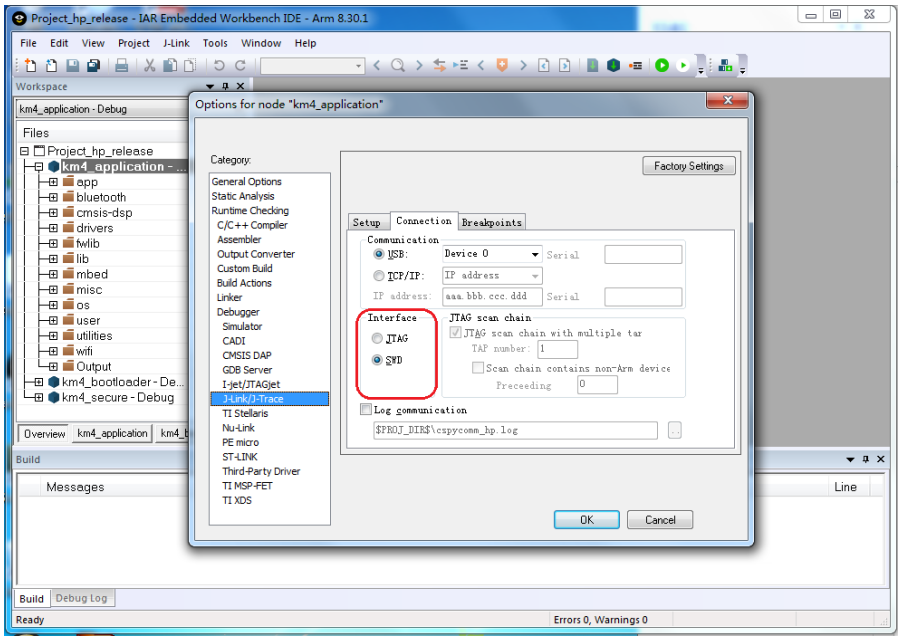


Fig6-14 J-Link interface setup

(3) If using RLX Probe, check whether the RLX Probe setting is correct

a) Click Project > Options > Debugger > Setup > Driver. Do not [illegible] Fig6-15 shows.

b) Click GDB Server > GDB Server, and [illegible] Fig6-16 shows. The value should be *localhost, port number*.

KM4: The default port numbers is 3333, which is set in project\realtek_amebaD_va0_example\EWARM-RELEASE\Probe\cm4\rlx_probe0.cfg

KM0: The default port numbers is 2331, which is set in project\realtek_amebaD_va0_example\EWARM-RELEASE\Probe\cm0\rlx_probe0.cfg

c) Open RLX Probe in project\realtek_amebaD_va0_example\EWARM-RELEASE\Probe\cm4\cm4_RTL_Probe, as Fig6-17 shows.

Note: The board must be reset before opening the RLX Probe, otherwise the connection may fail.

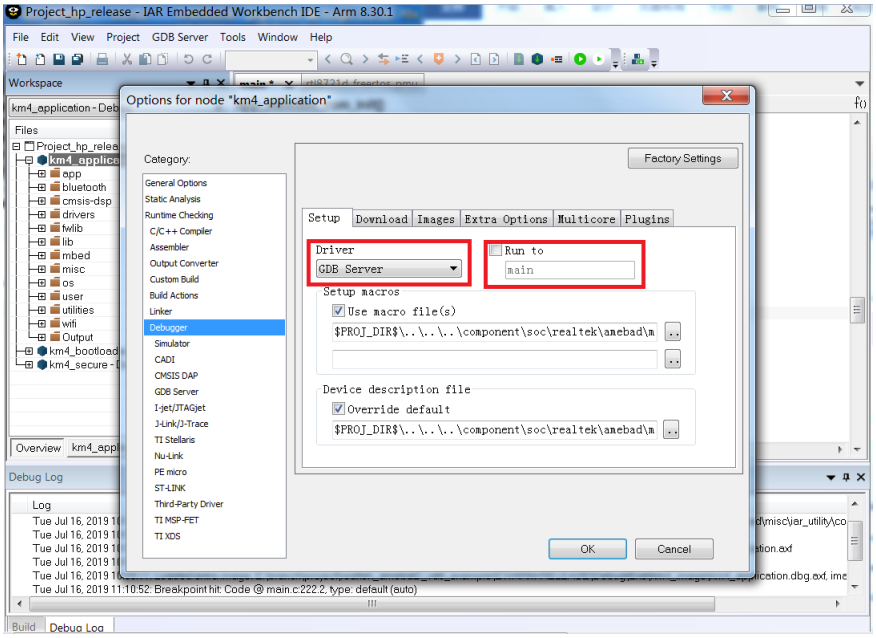


Fig6-15 RLX Probe setup

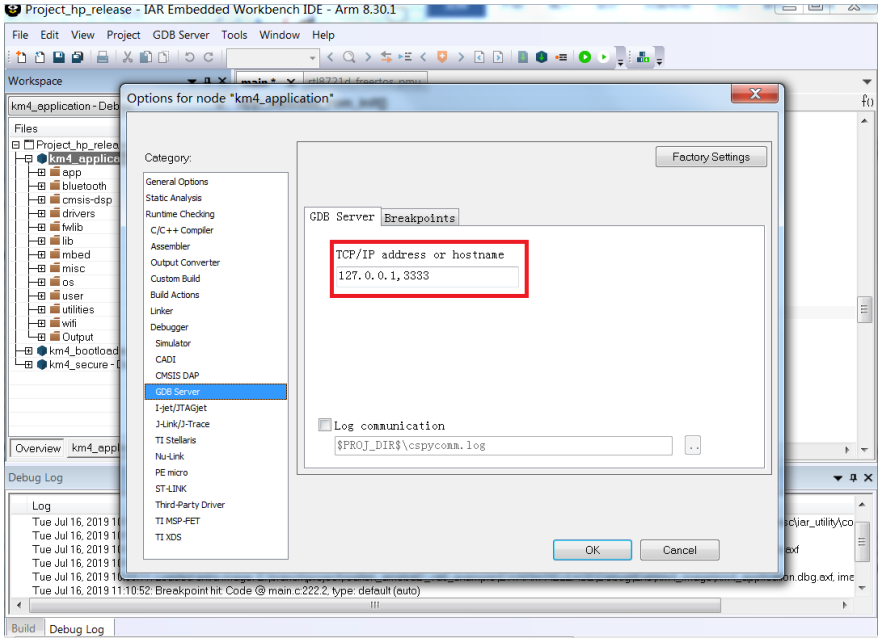


Fig6-16 RLX Probe interface setup

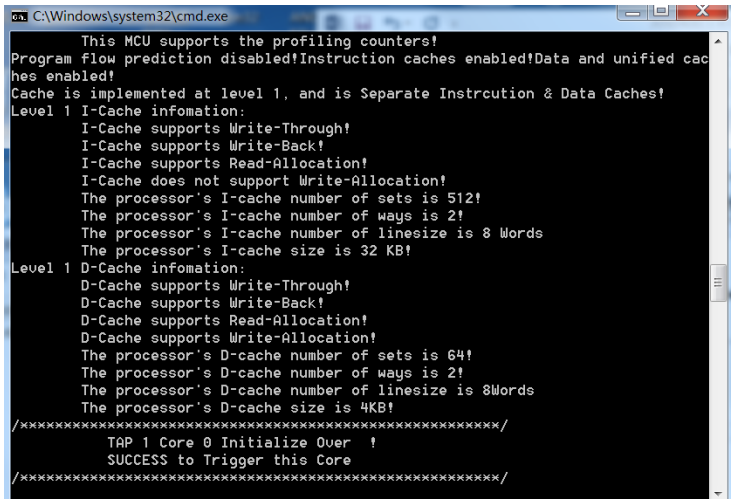


Fig6-17 RLX Probe window

(4) Click Project > Download > Download active application, image downloading starts.
When downloading, AmebaD prints the log, as Fig6-18 shows. You can check the log to see if download is successful.

```
[FlashInit] image_size:f48, link_address:8000000, flags:0
[FlashErase] block_start:0, block_size:1000
[FlashWrite] block_start:0, offset_into_block:0, count:f48
[FlashInit] image_size:82000, link_address:8006000, flags:0
[FlashErase] block_start:6000, block_size:1000
[FlashErase] block_start:7000, block_size:1000
[FlashErase] block_start:8000, block_size:1000
[FlashErase] block_start:9000, block_size:1000
[FlashErase] block_start:a000, block_size:1000
[FlashErase] block_start:b000, block_size:1000
[FlashErase] block_start:c000, block_size:1000
[FlashErase] block_start:d000, block_size:1000
[FlashErase] block_start:e000, block_size:1000
[FlashErase] block_start:f000, block_size:1000
[FlashErase] block_start:10000, block_size:1000
[FlashErase] block_start:11000, block_size:1000
[FlashWrite] block_start:6000, offset_into_block:0, count:c000
[FlashErase] block_start:12000, block_size:1000
[FlashErase] block_start:13000, block_size:1000
[FlashErase] block_start:14000, block_size:1000
[FlashErase] block_start:15000, block_size:1000
[FlashErase] block_start:16000, block_size:1000
[FlashErase] block_start:17000, block_size:1000
[FlashErase] block_start:18000, block_size:1000
[FlashErase] block_start:19000, block_size:1000
[FlashErase] block_start:1a000, block_size:1000
[FlashErase] block_start:1b000, block_size:1000
[FlashErase] block_start:1c000, block_size:1000
[FlashErase] block_start:1d000, block_size:1000
[FlashWrite] block_start:12000, offset_into_block:0, count:c000
[FlashErase] block_start:1e000, block_size:1000
[FlashErase] block_start:1f000, block_size:1000
[FlashErase] block_start:20000, block_size:1000
[FlashErase] block_start:21000, block_size:1000
[FlashErase] block_start:22000, block_size:1000
[FlashErase] block_start:23000, block_size:1000
[FlashErase] block_start:24000, block_size:1000
```

Fig6-18 Downloading log

(5) You can also erase all parts of the Flash memory if necessary.

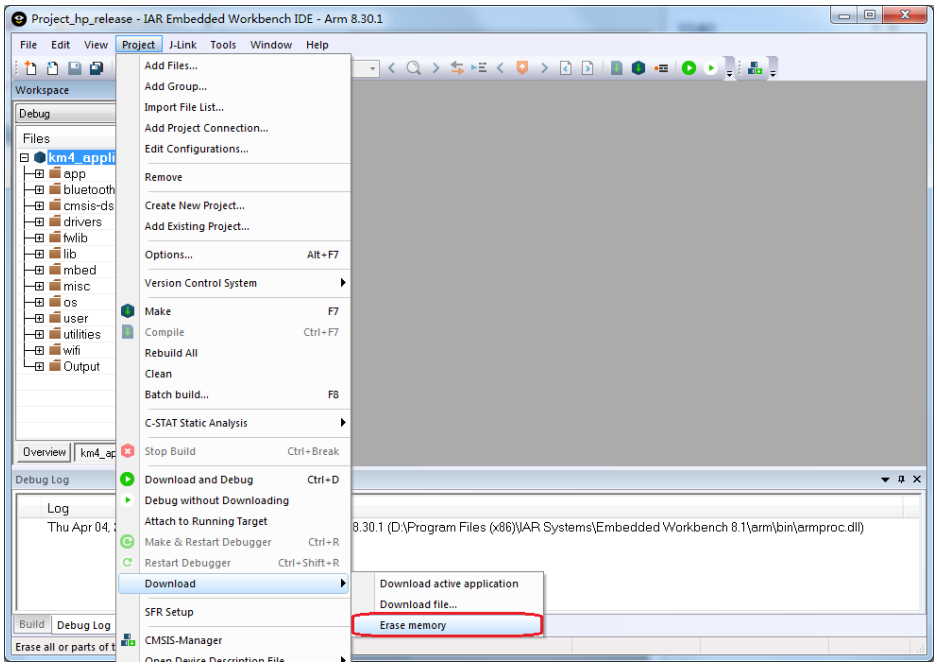


Fig6-19 Erasing Flash memory

6.3.4 IAR Debug

You can debug or trace KM0 and KM4 system individually with JLink or RLX Probe SWD.

Note: Considering KM4 is power on by KM0, you should make sure that KM0 has already boot up before debug KM4. For KM0, there is no such requirement because KM0 is power on immediately after reset.

6.3.4.1 J-Link Debug

Follow the steps to debug and trace code of target project with IAR by J-Link:

(1) Set the target project as active project and verify the debugger configurations as step (1) and (2).

(2) Click Project > Download and Debug or Project > Debug without Downloading

Download and Debug downloads the application and debug the project object file. If necessary, a make will be performed before download to ensure the project is up to date.

Debug without Downloading debug the project object file

(3) When starting IAR C-SPY to debug, it will firstly reset the target CPU and run to the main function as Fig6-20 shows

(4) Toggles a breakpoint at the statement or instruction that contains or is located near the cursor in the source window. K Ŝ á ¢ 2 Ǝ Ǝ f Ŝ . NJ Ŝ I·] button is found in the debug toolbar, as Fig6-21 shows.

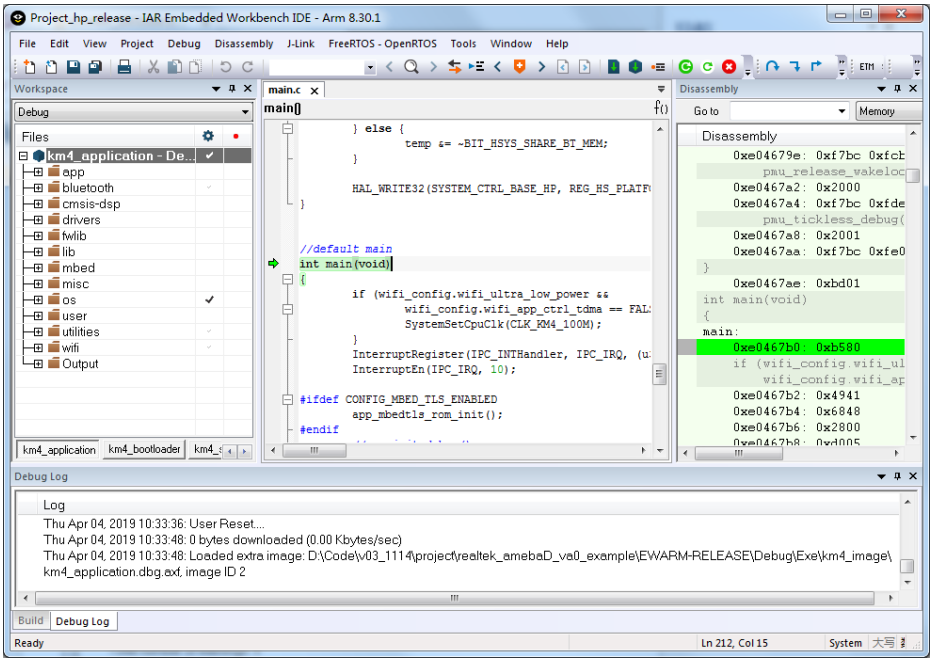


Fig6-20 Running to main() when debug

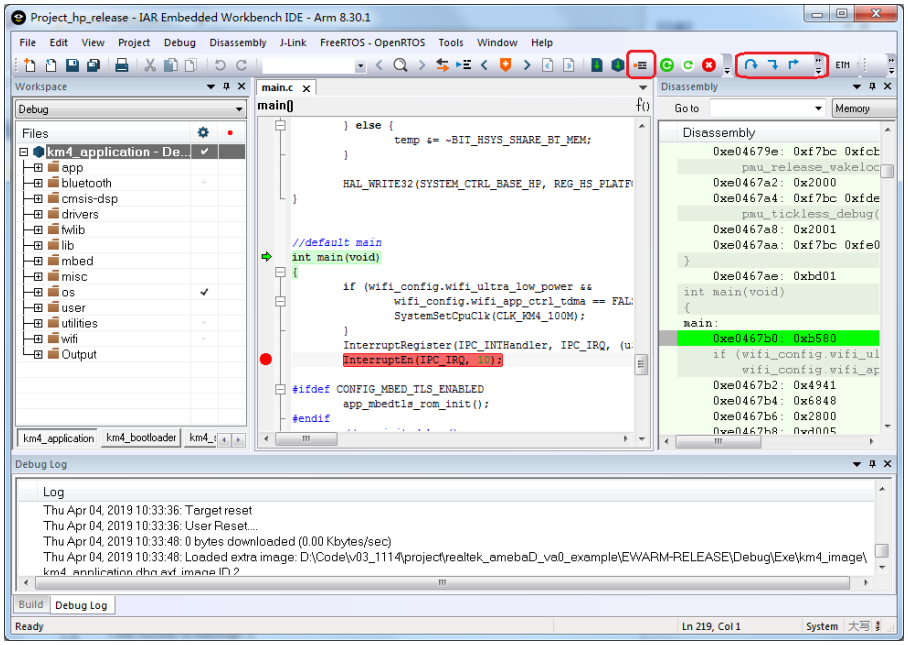
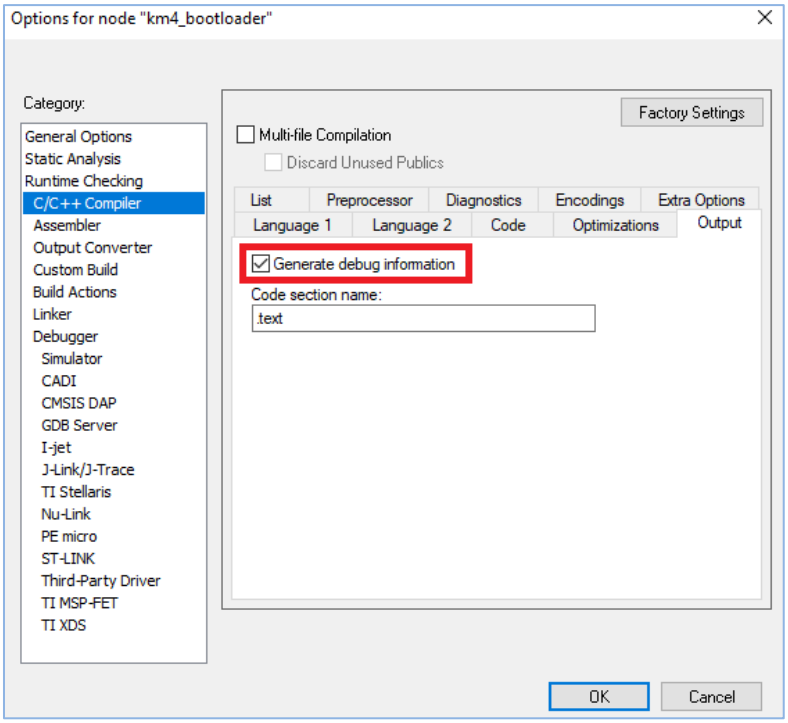


Fig6-21 Toggle breakpoint

(5) You can trace Ď ƨ Ř Ŝ ă ű ŜLJ ō è ă ű ŜLJ ǿ ƛ ű K ά { ű ŜLJ L ƴ ű ƨ έ ƨNJ ά D ƨ έ dz ƴ ű ƛ f ű NJ ƛ Ǝ Ǝ Ŝ

Note: If you want to trace the code step by step in km4_bootloader, click Options > C/C++ Compiler > Output > Generate under
ļ Ƴ π ψ ō ƨ ƨ ű f ƨ ŀ Ř ŜNJ Σ ŀ ƴ Ř Ď K Ŝ Ď ļ ű K Ŝ ă Ŝ ű ű ƛ ƴ Ǝ ά D Ŝ ƴ ŜNJ ŀ ű Ŝ Ř Ŝ ō dz Ǝ ƛ ƴ Ŧ ƨNJ Ƴ ŀ ű ƛ



6.3.4.2 RLX Probe Debug

Follow the steps to debug and trace code of target project with IAR by RLX Probe:

(1) Set the target project as active project and verify the debugger configurations as step (1) and step (3).
(2) Click Project > Attach to Running Target
(3) Set the target address.
When starting debug with RLX Probe, it will firstly reset the target CPU. If you want to run to a target address at first, you can set the PC address in project\realtek_amebaD_va0_example\EWARM-RELEASE\probe\cm4\rlx_probe0.cfg. The method is that uncomment
ά Ŝ ǿ ΦLJŎ ґ л ÈÈÈÈÈÈÈÈÈÈ έ Ͱ ỵ Ř Ŏ K Ͱ ỵ Fig6-22 and Fig6-23 shows ă ă ű ƨ ű K Ŝ Ͱ LJLJ ƨ ƛ ỵ ű Ŝ Ř

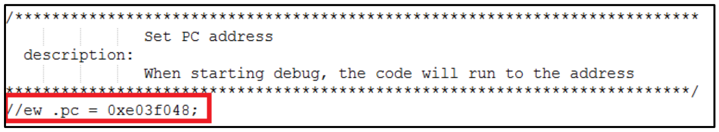

```
/*************************************************************************
                Set PC address
  description:
                When starting debug, the code will run to the address
*************************************************************************/
//ew .pc = 0xe03f048;
```

Fig6-22 Setting the target address

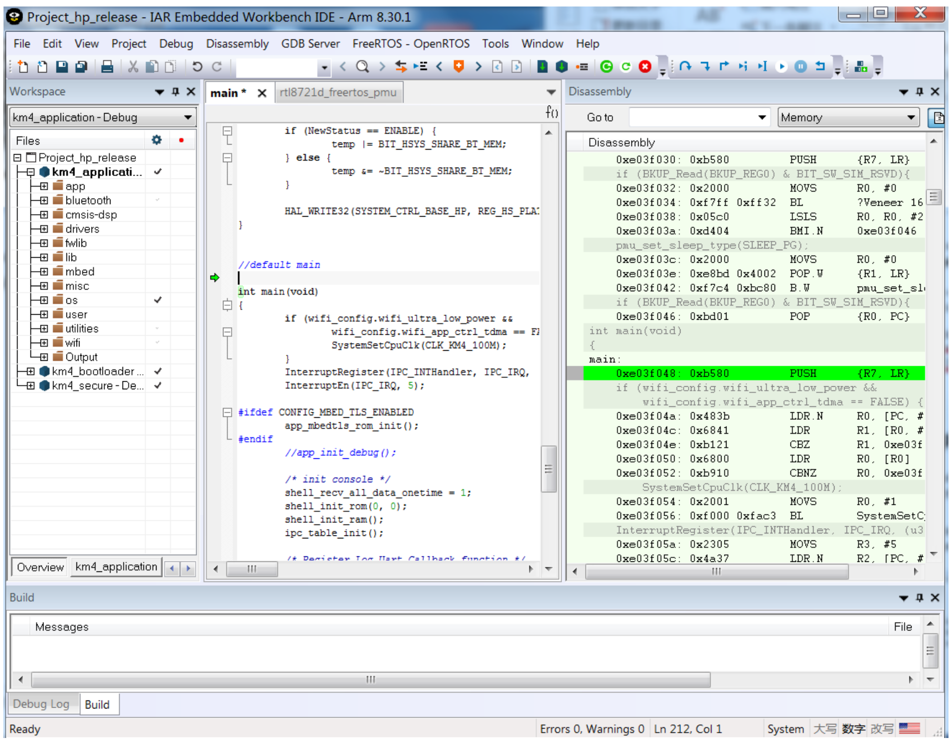


Fig6-23 Running to the target address when debug

(4) Toggles a breakpoint at the statement or instruction that contains or is located near the cursor in the source window. K Ŝ ά ¢ ƨ ∃ ∃ ƒ Ŝ . NJ Ŝ Ͱ ʅ button ʂ on the debug toolbar as Fig6-21 shows.
(5) You can trace ƨ Ř Ŝ ă ű Ŝ LJ ō è ă ű Ŝ NJ ǿ Ɖ ǿ K ǿ ỵ ű Ŝ LJ ű NJ Ŕ ∃ ∃ Ŝ NJ ƛ ỵ ∃ Ͱ ō NJ Ŝ Ͱ ʅ LJ ƨ ƛ ỵ ű Φ ¢ K

6.3.4.3 IAR Debug or Download Error

Because AmebaD has two CPU cores, and a post build script will be run after make, sometimes the debug or download thread cannot get the correct AXF file for debug or download, the error like Fig6-24 happens

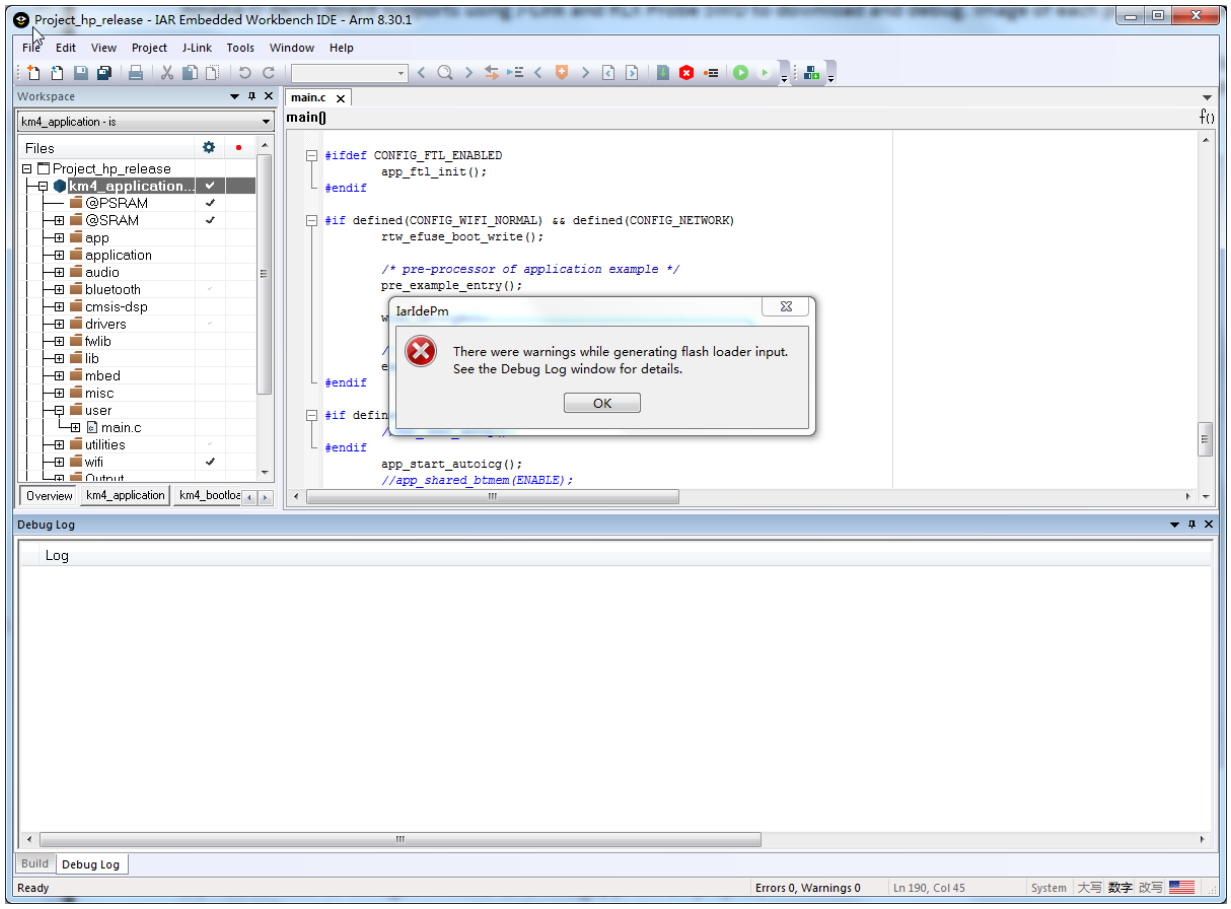


Fig6-24 IAR debug or download error

To avoid this error, you should build manually before debug or download, and disable auto build from Tools> Options > Project > Make before debugging as Fig6-25 shows

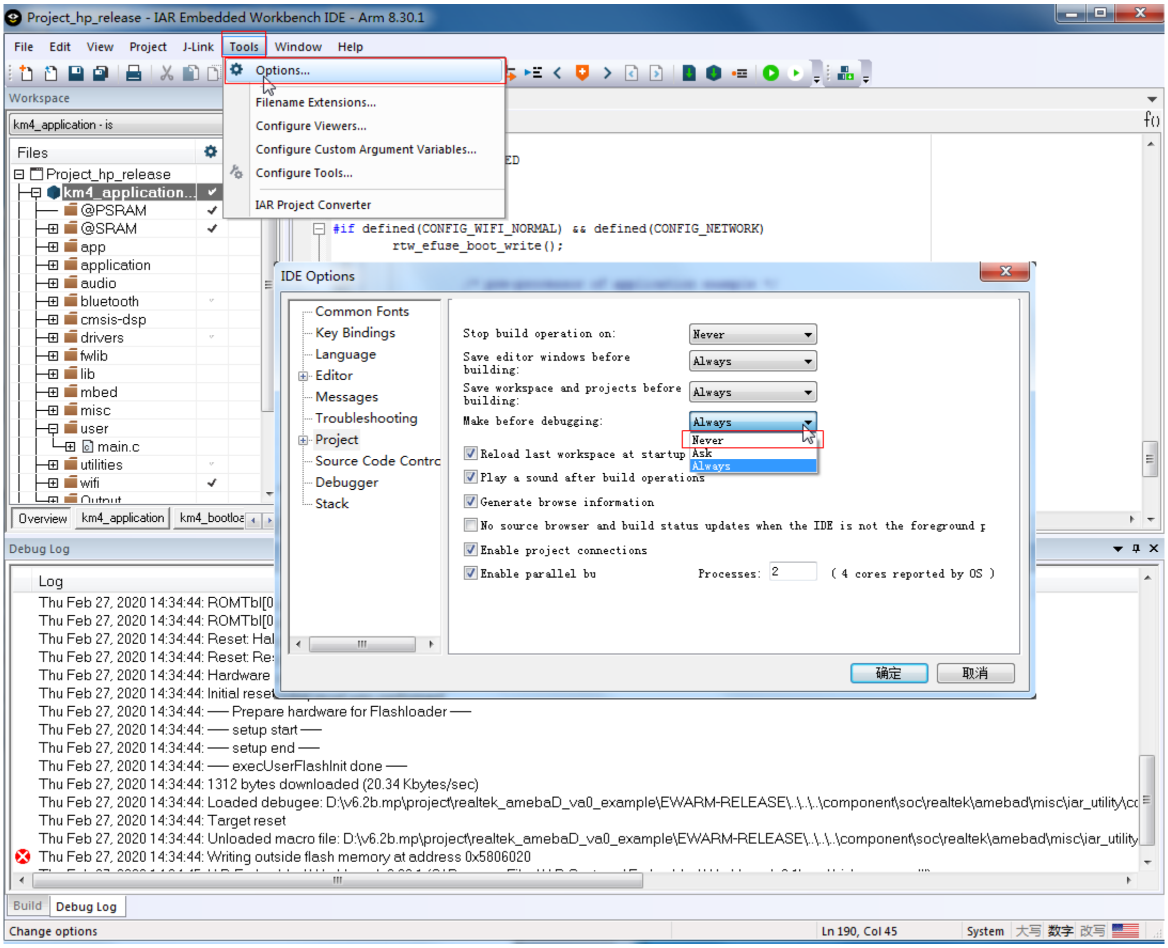


Fig6-25 IDE options

6.3.5 IAR Memory Configuration

6.3.5.1 Configuring Memory from IAR IDE

In order to allow users to manage memory flexibly, there are some configurations to put some code into certain memory region. Lɏ L!w ǿ2NJ ǃăLJŀŎŜ and ứƘ$ŴŜ a ẾNJŜNJ ȩzLJФw ȼƘŜ Ŏ2ŘŜ ƛɏ ά Ϋ t {w!aέ ∃NJ2dzLJ ǿ2dz Ŏ2ŘŜ ƛɏ group would be linked and loaded into SRAM. The rest of code will be placed on Flash and execute in place.

Note: / 2ɏăƛŘŜNJƛɏ∃ 2ɏtè {w!a ŀɏŘ t{w!a Ŏ2ɏűŀƛɏă ăŜŎdzNJŜ NJŜ∃ƛ2ɏrăn∑ ứƘŜ ά Ϋ {w!aέ to be placed outside these two groups.

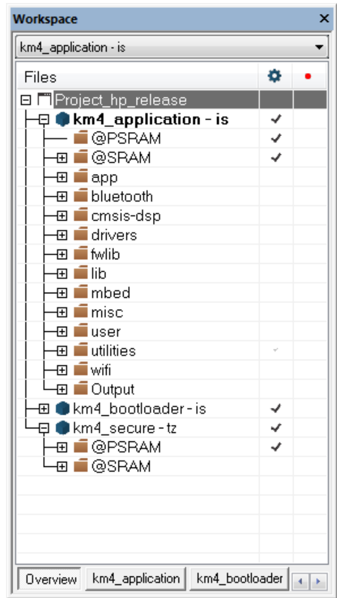


Fig6-26 Memory location configuration

6.3.5.2 Configuring Memory from ICF File

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

ICF file of AmebaD location:

ά LJNJ 2 realtek_amebaD_va0_example\EWARM-RELEASE\ ... ű ƒ y т н м Ř Ќ LJ ψ ƛ Ƴ ŀ Ǝ Ŝ н ψ ƛ ă Φ ƛ Ŏ Ŧ ές Ŧ 2 NJ ƛ Ǝ ƴ 2 NJ Ŝ

ά LJNJ 2 realtek_amebaD_va0_example\EWARM-RELEASE\ ... ű ƒ y т н м Ř Ќ LJ ψ for TrustZone project Φ ƛ Ŏ Ŧ ές

If having a good understand of the format of ICF, users can modify the section location in ICF file.

6.4 How to Build Sample Code?

The example source code is located in project\realtek_amebaD_va0_example\example_sources. To build sample code, you should copy the ά Ƴ ŀ ƛ ƴ Φ Ŏ ές Ŧ ƛ ƒ Ŝ ƛ project\realtek_amebaD_va0_example\src_hp and replace the original one.

C 2 NJ Ŝ È ŀ Ƴ LJ ƒ Ŝ Σ ȅ 2 dz project\realtek_amebaD_va0_example\example_sources\I2C\mbed\i2c_int_mode\src to use i2c_int_mode example code, as Fig6-27 shows.

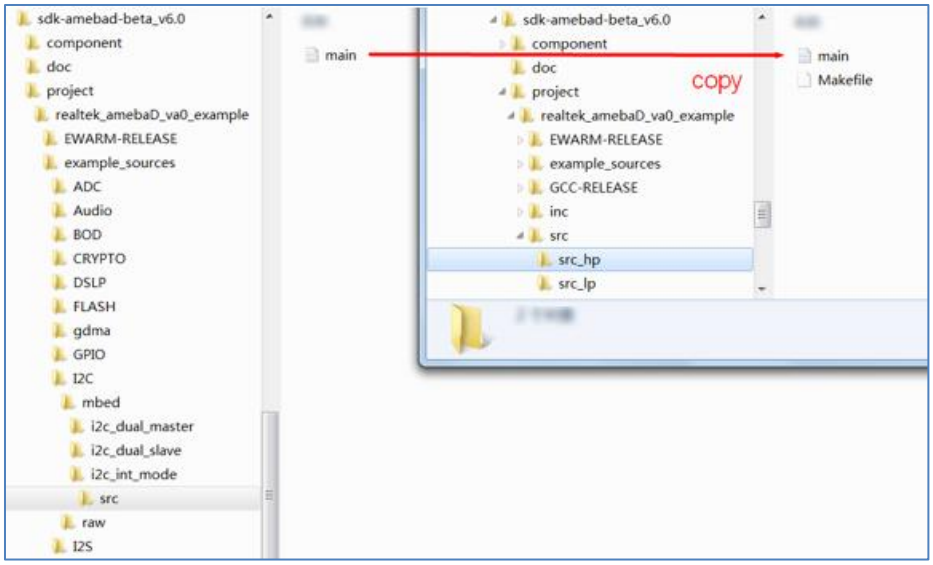
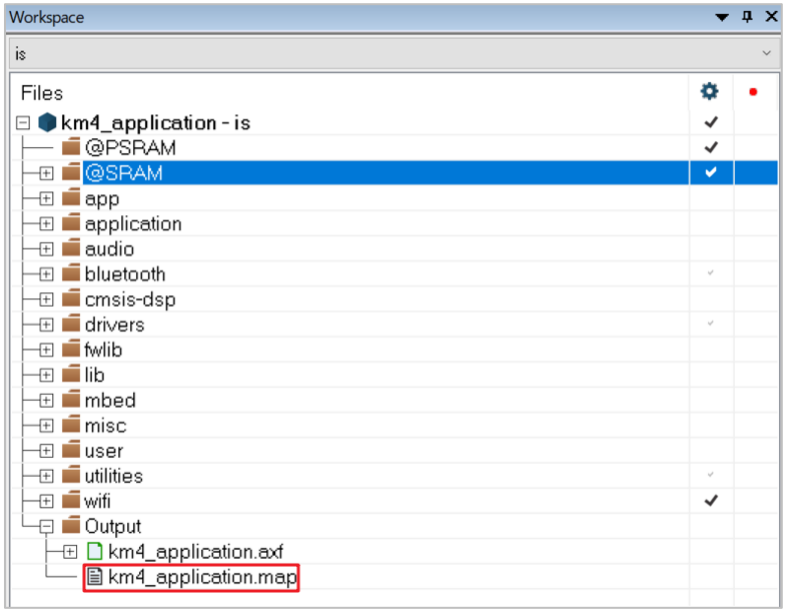


Fig6-27 Building sample code

6.5 Used Memory Size Calculation

This section explains how to calculate used memory size by users in IAR project, whose version is IAR 8.30.1 or higher. You can refer to km4_application.map file in the following folder: Project folder: project\realtek_amebaD_va0_example\EWARM-RELEASE\Debug\List\km4_application



6.5.1 Memory Section

A7 (PSRAM): This section is read/write data in PSRAM (0x2000000 to 0x23FFFFFF).
BTTRACE: This section is reserved for BT TRACE
A5 (XIP): This section is read only data in XIP

A4 (SRAM): This section is read/write data in SRAM.
P1 (SRAM): This is BSS section in SRAM.

6.5.2 Memory Size

6.5.2.1 Memory Size in SRAM

There are two sections resident in SRAM, which are A4 and P1. As we can see from the map file, for standard SDK, these two sections use almost all of the memory space from SRAM (476KB).

A4 has size 0x17ad7.

```
"A4":                                      0x17ad7
  IMAGE2                          0x1000'5000  0x17ad7  <Block>
    .ram image2.entry             0x1000'5000    0x20  <Block>
      .image2.entry.data  rw data 0x1000'5000     0xc  rtl8721dhp app start.o [1]
```

P1 has size 0x5dfc8.

```
"P1":                                      0x5dfc8
  .ram heap.data                  0x1001'cae0  0x50000  <Block>
    .ram image2.bfsram.data
                                  0x1001'cae0  0x50000  <Block>
      .bfsram.data        uninit  0x1001'cae0  0x50000  freertos heap5 config.o [1]
```

So totally 0x17ad7 + 0x5dfc8 = 0x75ab0 = 470K bytes memory in SRAM is used.

The total SRAM space is defined in project\realtek_amebaD_va0_example\EWARM-RELEASE\RTL8721d_memory_layout_is.icf

```
define symbol __ICFEDIT_region_HS_BD_RAM_NS_start__          = 0x10005000;
define symbol __ICFEDIT_region_HS_BD_RAM_NS_end__            = 0x1007C000-1;
```

For this case, the total size of SRAM is 0x1007C000 - 0x10005000 = 0x77000 = 476K bytes, there is still 6K (= 476K - 470K) bytes free SRAM space.

6.5.2.2 Memory Size in PSRAM

There is one section in PSRAM called A7, the memory space from PSRAM (4MB).

A7 has size 0x54800

```
"A7":                                      0x54800
  IMG2 PSRAM                    0x200'0000      0x0  <Block>
  .psram ns.bss                 0x200'0000  0x54800  <Block>
    .psram.bss                  0x200'0000  0x54800  <Block>
      .psram.bss          uninit  0x200'0000  0x54800  rtw opt skbuf.o [1]
```

So totally 0x54800 = 338K bytes memory in PSRAM is used.

6.5.2.3 Memory Size in XIP

XIP can only place text section, so there is only one section called A5.

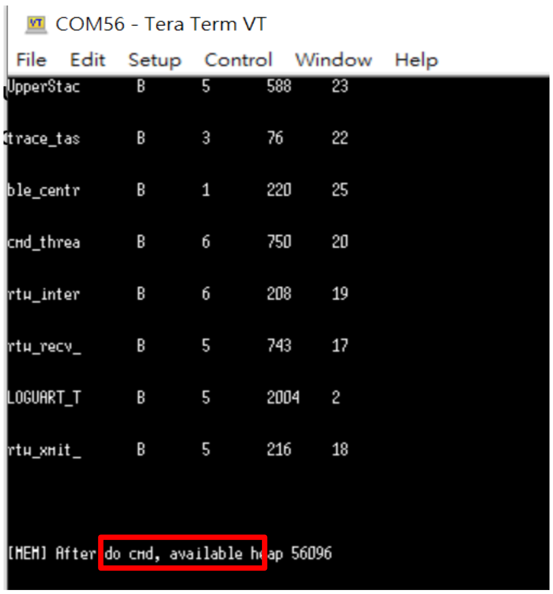
A5 has size 0x8fe8.

```
"A5":                                          0x8afe8
  .xip image2.text               0xe00'0020   0x8afe8   <Block>
    .text                ro code  0xe00'0020     0xfc   app task.o [1]
    .text.os msg queue create intern
                         ro code  0xe00'011c     0x40   os msg.o [2]
```

So totally 0x8afe8 = 555K bytes memory in XIP is used

6.5.2.4 Available Heap Size

When calculating total used memory size, available heap size after WLAN association and BT connection needs to be considered. In the above case, it is 56096 bytes.



Finally, you can use total SRAM 476K bytes to subtract these sections of memory to obtain totally free global memory and free heap in SRAM

Formula is as below:

SRAM free global memory: 476K – (A4) – (P1) = 476*1024 – 96983 – 384968 = 5473 (bytes)
Available heap: 56096 bytes.
Totally free SRAM memory and heap: 5473 + 56096 = 61569 (bytes)
Totally free PSRAM memory: 4*1024*1024 – 338 * 1024 (A7) = 3848192 (bytes)

7 Demo Board

7.1 PCB Layout Overview

The PCB layout of 2D and 3D are shown in Fig7-1 and Fig7-2.

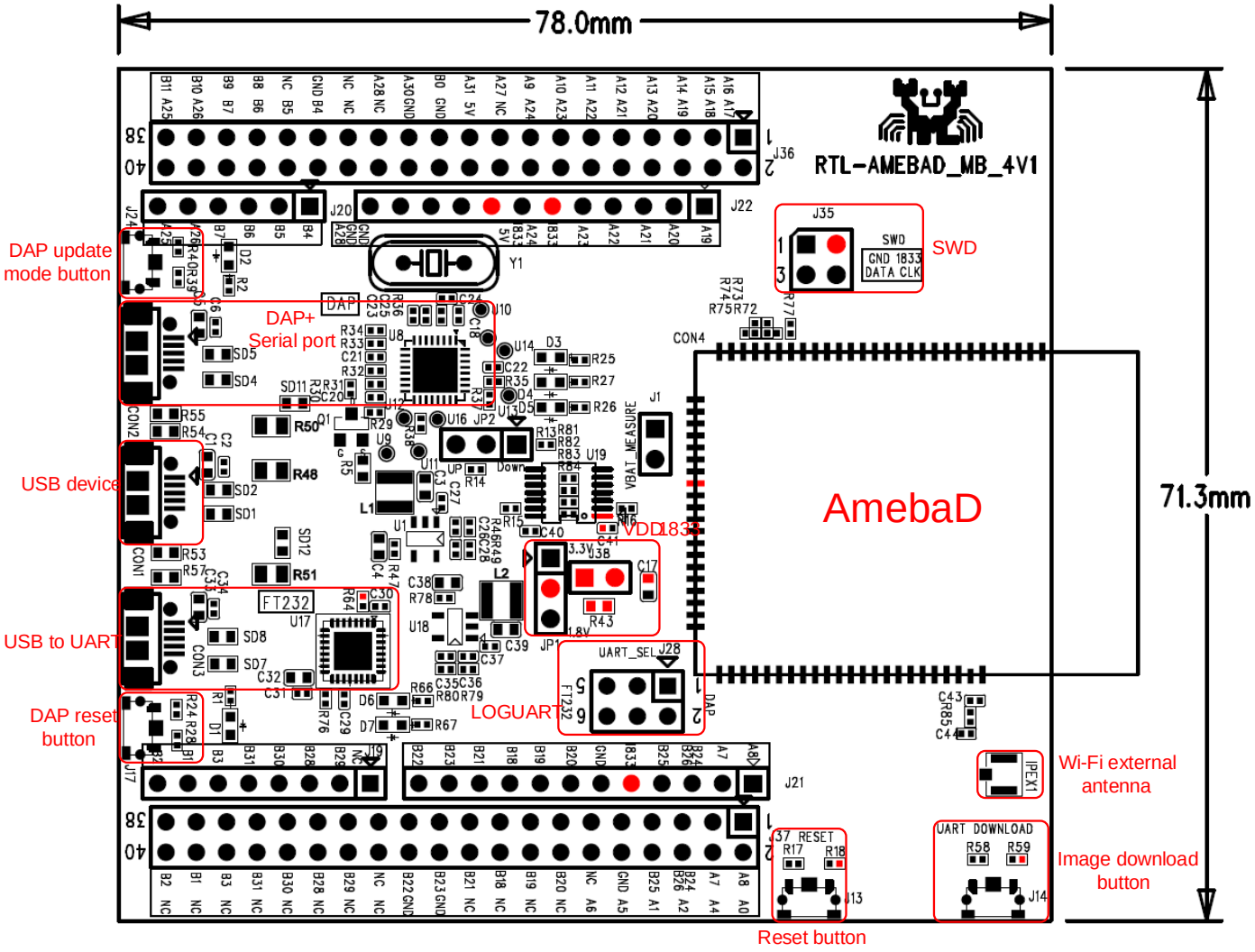


Fig7-1 Demo board PCB layout (2D)

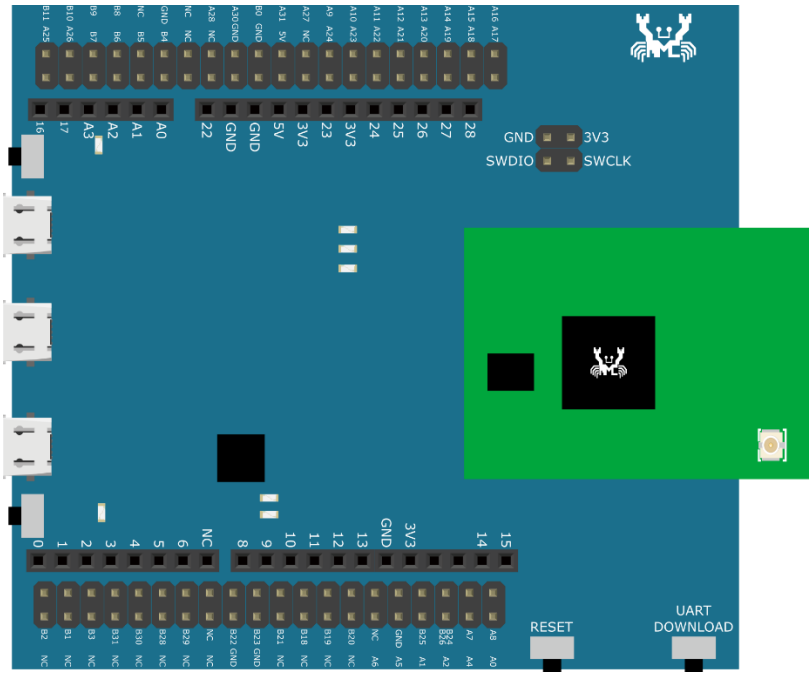


Fig 7-2 Demo board PCB layout (3D)

7.2 Pin Out

The pin out board is shown in Fig 7-3.

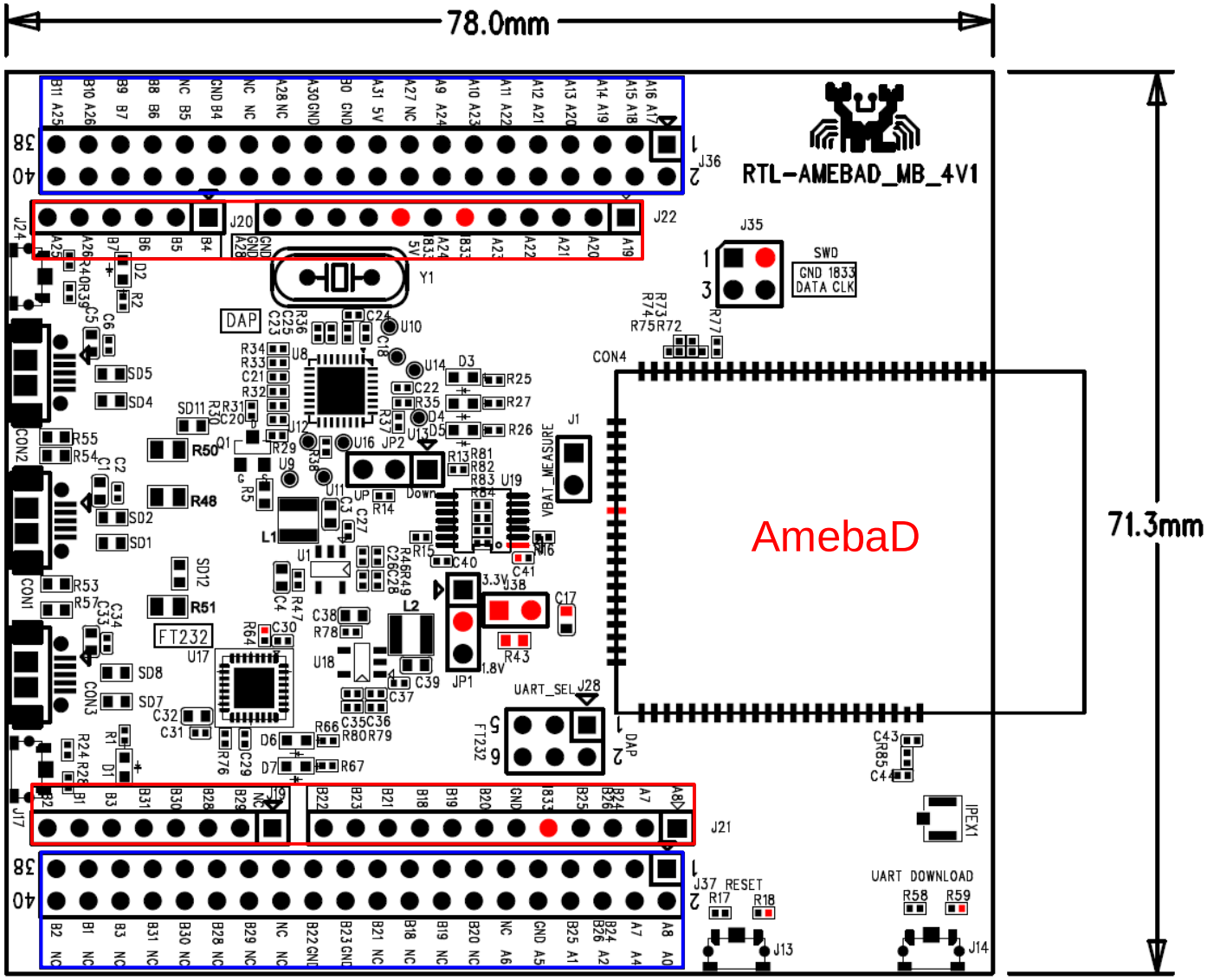


Fig7-3 Demo board I/O pin out

There are four rows of pins on the board.

The pins in the red box are used for Arduino REF

The pins in the blue box are all the GPIO pins

7.3 DC Power Supply

The 3.3V/1.8V power supply board is shown in Fig7-4.

Jump JP1 is used to select 3.3V or 1.8V power supply

Jump J38 is for current test. You can test the current power after taking off the R43.

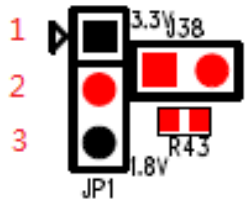


Fig7-4 Demo board – 3.3V/1.8V power supply

When you select power supply, refer to Table7-1.

Table7-1 3.3V/1.8V power supply selection

Power Supply Select	JP1
3.3V	1-2 connected
1.8V	2-3 connected

7.4 USB Interface Configuration

The USB interface configuration board is shown in Fig7-5 and Fig7-6.

Fig7-5 Mother board – USB interface configuration

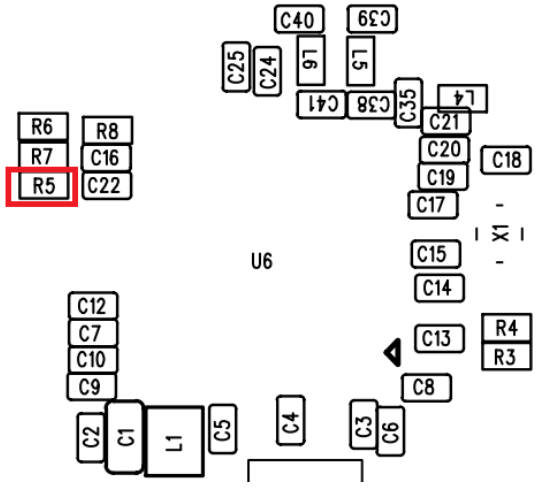


Fig7-6 Module board – USB interface configuration

For normal GPIO usage by default, R72/R75/R77 on mother board will part on with 0 Ohm resistors, R5 on module board needs to take off. For USB usage, you need to take off R77 part on R73 & R74 with 0 Ohm resistors on mother board and part on R5 on module board with a 12K Ohm 1% precision resistor.

7.5 LOGUART

The LOGUART board is shown in Fig7-7. When you select LOGUART, please refer to Table7-2.

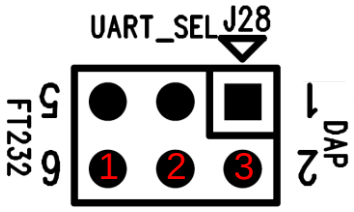


Fig 7-7 Demo board – LOGUART

Table 7-2 LOGUART Selection

LOGUART Select	JP1
FT232	1-2 connected
DAP	2-3 connected

7.6 SWD

The SWD board is shown in Fig 7-8.

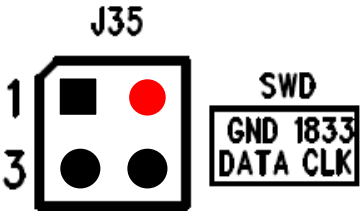


Fig 7-8 Demo board – SWD

Note: For 1V0 board, there is a bug, you should use CLK as DATA, and use DATA as CLK.

7.7 VBAT ADC

The VBAT ADC board is shown in Fig 7-9. J1 is used to test VBAT ADC.

Fig 7-9 Demo board – VBAT ADC

8 ImageTool

8.1 Introduction

This chapter introduces how to use ImageTool to encrypt, generate and download images. As show in Fig8-1, ImageTool has four tab pages.

Download: used as image download server to transmit images to Ameba through UART
Generate: combine individual images and generate a composite image
Encrypt: encrypt images which are used for firmware protection
Security: generate key pair and signature for secure bootloader and save as new image

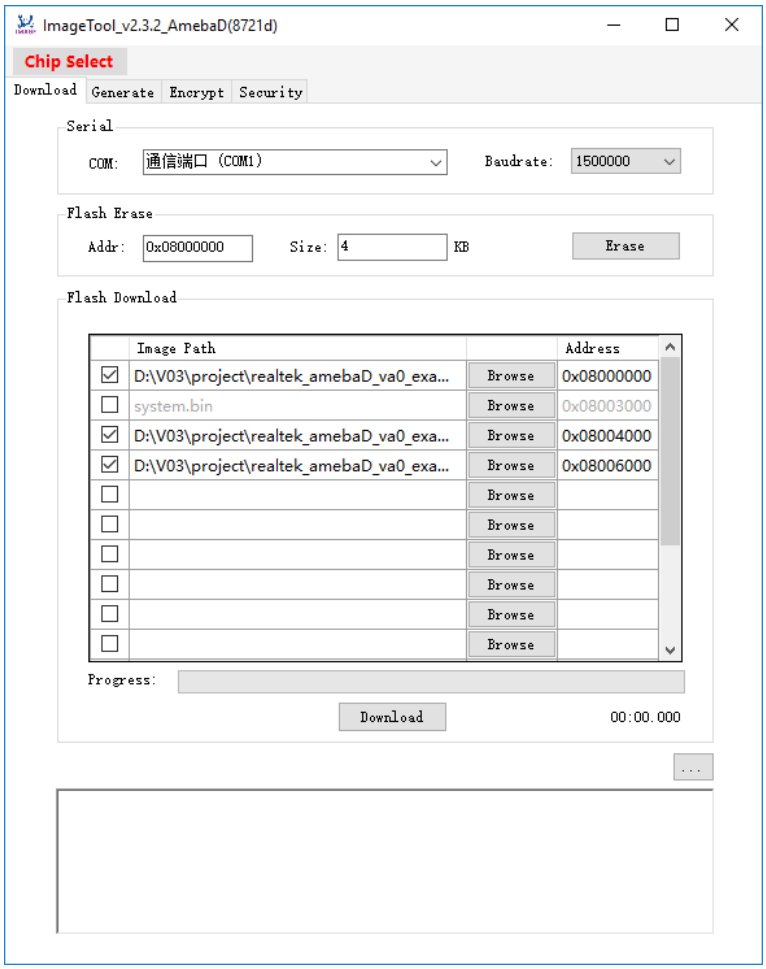


Fig8-1 ImageTool UI

8.2 Environment Setup

8.2.1 Hardware Setup

The hardware setup is shown in Fig8-2.

Note: If using external UART to download images, FT232 USB UART dongle must be used.

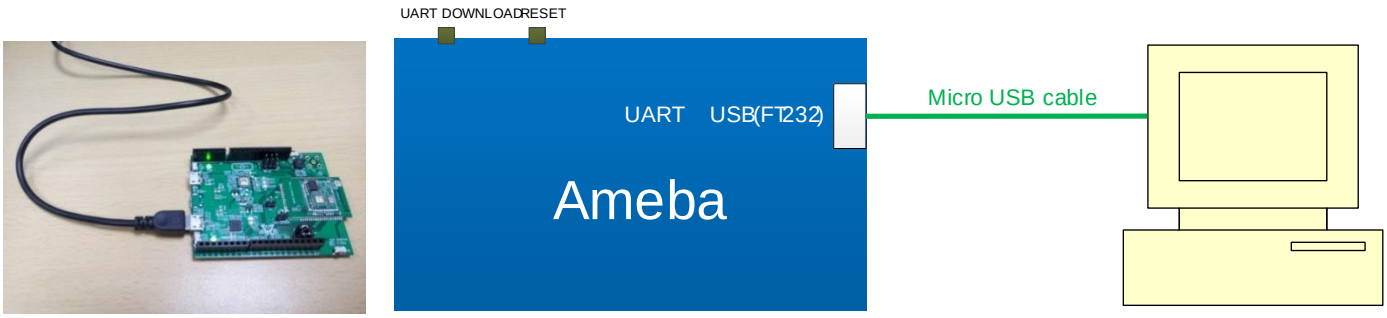
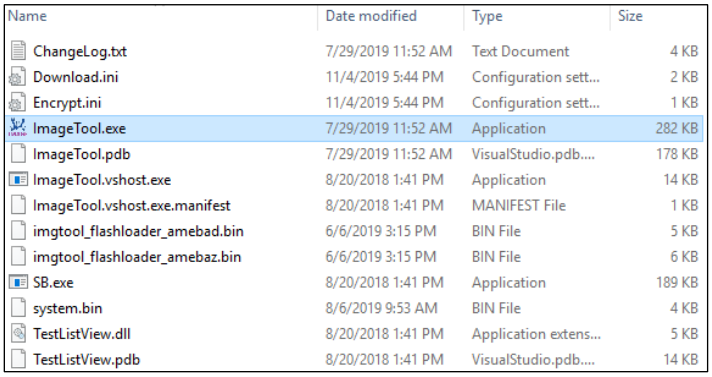


Fig8-2 Hardware setup

8.2.2 Software Setup

Environment Requirements: EX. WinXP, Win 7 Administrator, Microsoft .NET Framework 3.5
ImageTool.exe Location: tools\AmebaD\Image_Tool\ImageTool.exe



Name	Date modified	Type	Size
ChangeLog.txt	7/29/2019 11:52 AM	Text Document	4 KB
Download.ini	11/4/2019 5:44 PM	Configuration sett...	2 KB
Encrypt.ini	11/4/2019 5:44 PM	Configuration sett...	1 KB
ImageTool.exe	7/29/2019 11:52 AM	Application	282 KB
ImageTool.pdb	7/29/2019 11:52 AM	VisualStudio.pdb....	178 KB
ImageTool.vshost.exe	8/20/2018 1:41 PM	Application	14 KB
ImageTool.vshost.exe.manifest	8/20/2018 1:41 PM	MANIFEST File	1 KB
imgtool_flashloader_amebad.bin	6/6/2019 3:15 PM	BIN File	5 KB
imgtool_flashloader_amebaz.bin	6/6/2019 3:15 PM	BIN File	6 KB
SB.exe	8/20/2018 1:41 PM	Application	189 KB
system.bin	8/6/2019 9:53 AM	BIN File	4 KB
TestListView.dll	8/20/2018 1:41 PM	Application extens...	5 KB
TestListView.pdb	8/20/2018 1:41 PM	VisualStudio.pdb....	14 KB

8.3 Download

8.3.1 Image Download

Assuming that the ImageTool on PC is a server, it sends images files to client (Ameba) through UART. There are two ways to download images to board.

8.3.1.1 Based on Hardware Reset

The way based on hardware reset is a manual method to download images, and it is the primary and recommended method.

(1) Enter into UART_DOWNLOAD mode
   a) Push the UART DOWNLOAD button and keep it pressed.
   b) Re-power on the board or press the Reset button.
   c) Release the UART DOWNLOAD button.
   Now, Ameba board get into UART_DOWNLOAD mode and is ready to receive data.
(2) Click Chip Select (in red) on UI and select chip (AmebaD or AmebaZ)
(3) Select the corresponding serial port and transmission baud rate. The default baud rate is 1.5Mbps (recommended)
(4) Click the Browse button to select the images (km0_boot_all.bin/km4_boot_all.bin/km0_km4_image2.bin) to be programmed and input addresses.
   The image path is located {*path*}\project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\image and {*path*}\project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\image, where {*path*} is the location of the project on your own computer.

The default target address is the SDK default image address, you can use it directly.

(5) Click Download button to start. The progress bar will show the transmit progress of each image. You can also get the message of operation successfully or errors from the log window.

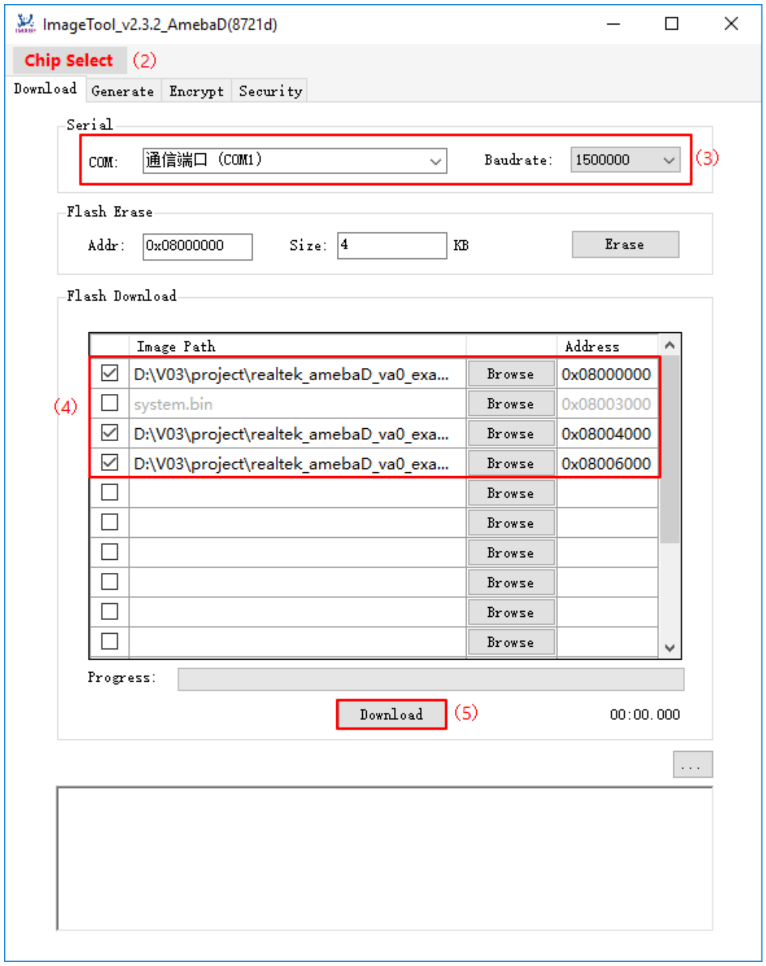


Fig8-3 ImageTool Download tabpage setting

8.3.1.2 Based on Command

The way based on command is the auto-download and alternate method to download images. Φ ² Ŝ ădzƎ Ǝ Ŝ ă ű è ƻ Based on ƻ LJ ű ű Ƙ Hardware Reset ǝ ō Ŝ The auto-download method is dependent on the command in SDK.

Steps Execute step (2) ~ (5) mentioned in section Based on Hardware Reset ǝ

After that, Ameba board will enter into UART_DOWNLOAD mode, and automatically response to the command reboot uartburn ǝ ƛ Ŧ ű Ƙ ƛ ă command is not removed from SDK and KM4 is running normally.

Note: The command reboot uartburn ǝ ƛ ƴ { 5 Y Ƴdz ă ű ƴ ƻ ű ō Ŝ NJŜ Ƴ ƻ Ǿ Ŝ Ř ŀ ƴ è Ƙ ƻ ǿ ƛ Ŧ è ƻ dz ǿ ŀ ƴ ű ƛ Ƴ ŀ Ǝ Ŝ ă ŀ Ŏ Ŏ ƻ NJŘ ƛ ƴ Ǝ ű ƻ Based on Hardware Reset ǝ ƛ ă ű Ŝ Ř ƛ ƴ ă Ŝ Ŏ ű ƛ ƻ ƴ ά

8.3.2 Flash Erase

Steps to erase Flash are as follows:

(1) Ameba enters into UART_DOWNLOAD mode as introduced in section 8.3.1.

(2) Select the corresponding serial port and baud rate.
(3) Input erase start address which has to be 4K byte aligned.
(4) Input erase size which would be cast to a multiple of 4KB
(5) Click Erase button, and erase operation begins. You would get the operation result from the log window.

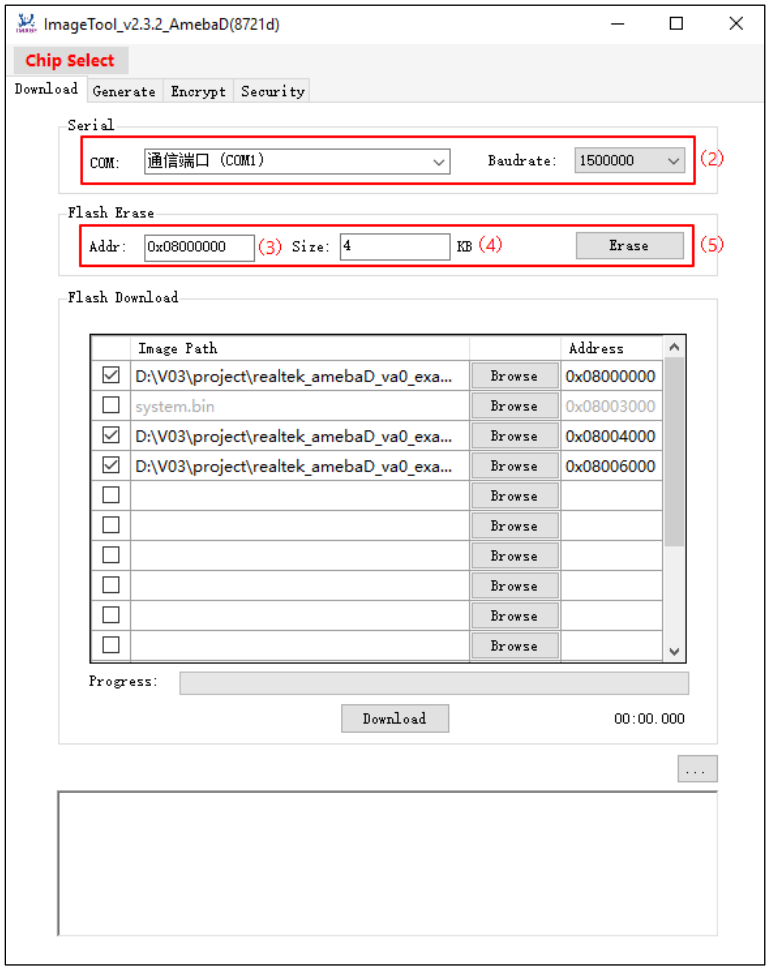


Fig8-4 Flash erase operation

Note: Users don't need to erase Flash manually before download images into Flash. Image Download operation would erase Flash automatically according to the image and address to be programmed.

8.3.3 Flash Status Operation

After some abnormal operations such as power lost when Flash is programming, the Flash would enter into Write Protection (WP) to protect some area against program and erase.

To release Flash from WP state, the BPs in status register should be cleared. ImageTool provides the function to get and set Flash status register value. To use it, the following steps are necessary

(1) Check the Read/Write Status Register Command and Sequence according to the Flash datasheet. Notice that different Flash IC would have different sequence.
(2) Get Ameba into UART_DOWNLOAD mode as introduced in section 8.3.1.
(3) Select the corresponding serial port and baud rate.
(4) Click Advanced Settings button in Fig8-5, and the Advanced Settings window shown in Fig8-6 popups.

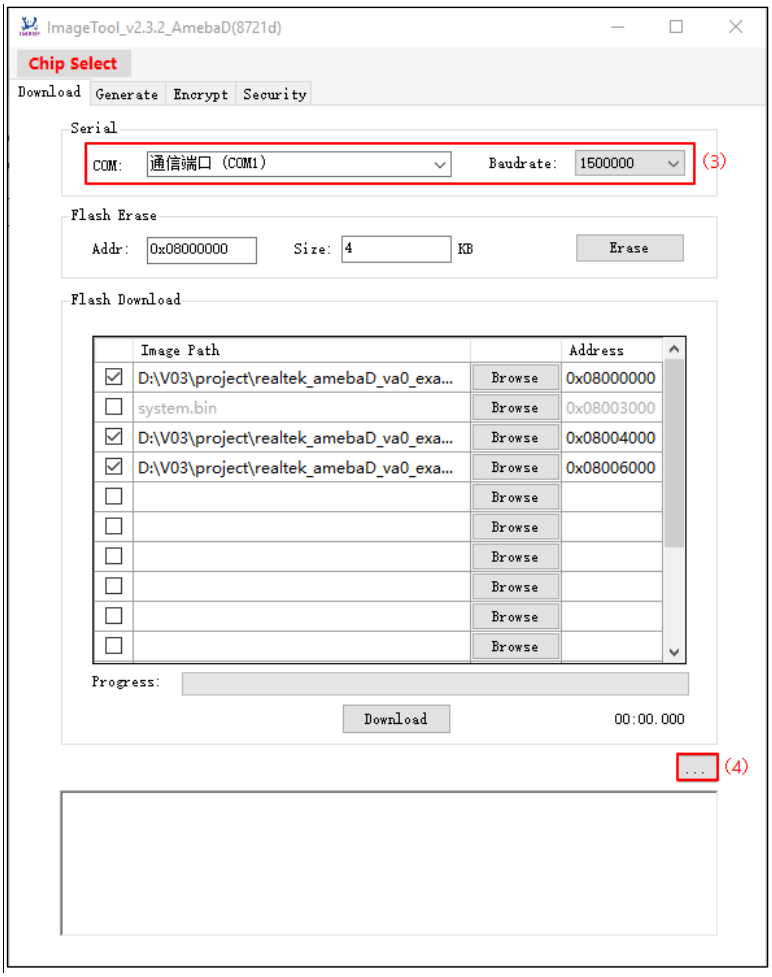


Fig8-5 Ψdvanced Setting ă Ω ø λ y˝ Ř ƨ ø

(5) Select Read/Write Command and Length according to Flash datasheet. If writing status to Flash, remember to input value.
(6) Click Read button to read status register value, Write button to write status value to Flash.

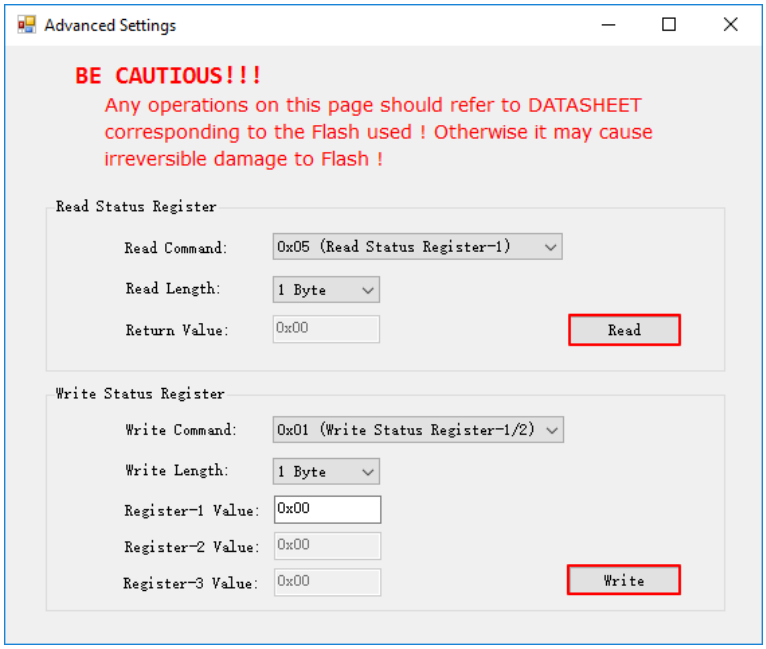


Fig8-6 Flash status operation

In case of MXIC Flash chip, the status register is shown in Table8-1.

To clear BP bits, read status register configuration is as below:

Read Command: 0x05

Read Length: 1 Byte

Write status register configuration is as below:

Write Command: 0x01

Write Length: 1 Byte

Register-1 Value: 0x40

Table8-1 MXIC Flash status register

Bit	Name	Description	Volatile Bit
7	SRWD	Status register write protect 1: Status register write disable 0: Status register write enable	No
6	QE	Quad enable. 1: Quad enable 0: Not Quad enable	No
5	BP3	Level of protected block.	No
4	BP2	Level of protected block.	No
3	BP1	Level of protected block.	No
2	BP0	Level of protected block.	No
1	WEL	Write enable latch. 1: Write enable 0: Not write enable	Yes
0	WIP	Write in progress bit. 1: In write operation 0: Not in write operation	Yes

Note: For other Flash IC, please refer to corresponding datasheet. Otherwise, wrong operation would cause irreversible damage to Flash.

8.4 Generate

¢ K Ŝ Ψ D Ŝ y ŜNJ Ⅰ ű Ŝ Ω ű Ⅰ ō LJ Ⅰ Ǝ Ŝ K Ⅰ ă ű ø ƨ Ŧ dz y Ǒ ű ƛ ƨ y ă Y
Merge individual images and generate a composite image named Image_All.bin
Generate Cloud OTA image named OTA_All.bin

8.4.1 Merge Images

Steps to generate a composite image are as follows:
(1) Select Image_All as Generate Target type (in red).
(2) Click Browse button to select images to be contacted and input corresponding relative address. The Memory Layout bar will show the relative positions of the selected images. If the contiguous images overlap, the overlapped area is in red color for warning.
(3) Click Generate button. Specify the output file name and path yourself in the popup "Save As" window, the final image (Image_All.bin) is generated

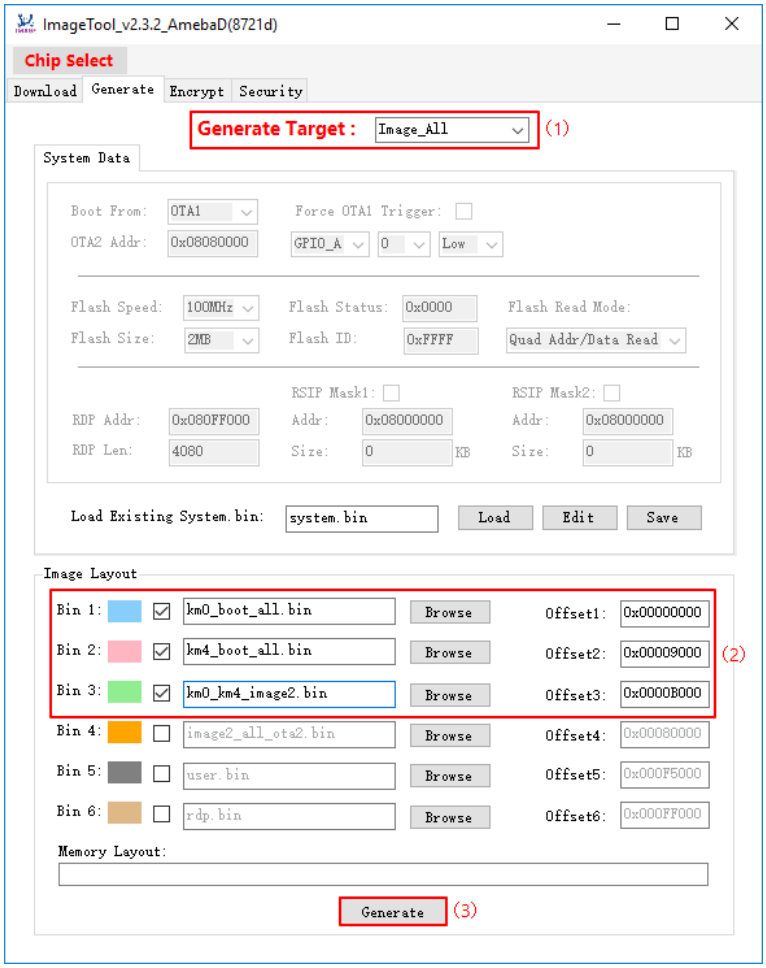


Fig 8-7 Image_All.bin generation

8.4.2 Generate Cloud OTA Image

Steps to generate a Cloud OTA image are as follow
(1) Select OTA_All as Generate Target type (in red).
(2) Input Image Version, the default value is 0xFFFFFFFF

(3) Click Browse button to select images. The address can be ignored. The Memory Layout bar will show the relative positions of the two images. If they overlap, the overlapped area is in red color for warning.
(4) Click Generate button to specify the name and path of the output file. After the operation is done, the cloud image (OTA_All.bin) is generated

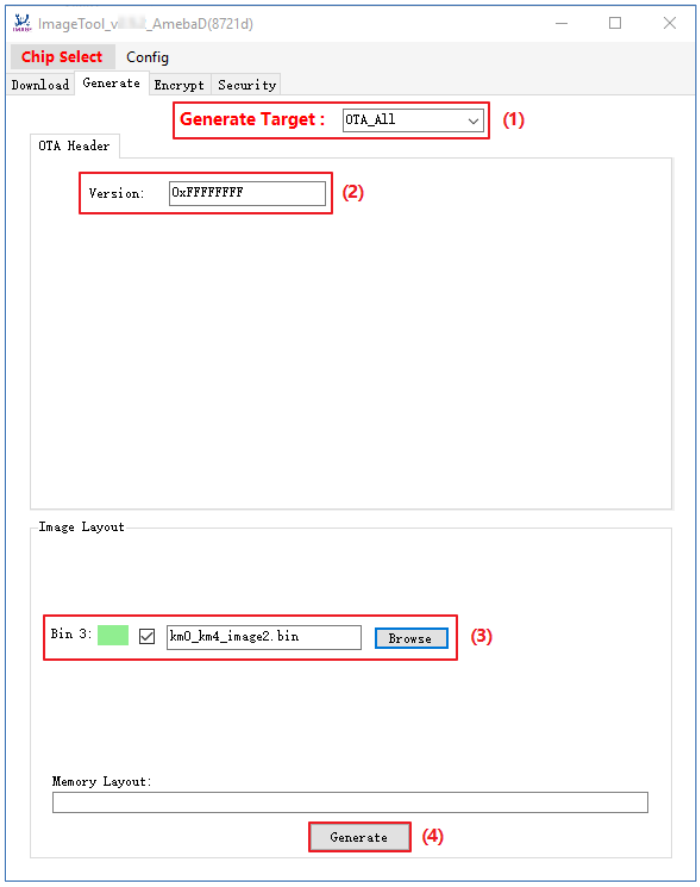


Fig8-8 OTA_All generation

8.5 Encrypt

The tool provides encrypting images which is used in firmware protection. Steps to encrypt images are as follows:
(1) Select Encryption Type
(2) Input Key (16 bytes).
(3) Select Images that need to be encrypted and input corresponding address
(4) Click Encrypt button. You will get the encrypted images with suffix "_enc" in the same directory of original file, which located in the same directory.

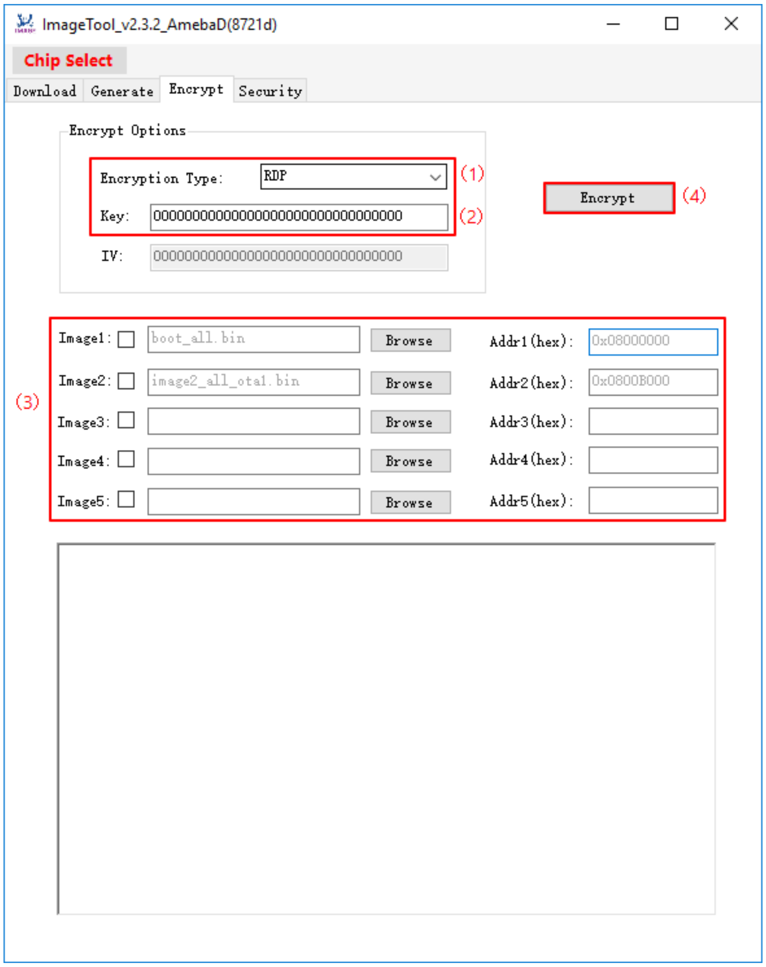


Fig8-9 ImageTool Ψ 9 ƴ ǑNJ è LJ ś setting ű ŀ ō LJ ŀ ∃ Ŝ

8.6 Security

¢ K Ŝ Security Ħ tab page is used to generate secure boot image. Steps to generate Secure Boot image are as follows:

(1) Input Seed (32 bytes), which is used to generate key pair.
(2) Click Gen Keypair button to generate Public Key and Private Key.
(3) Click Browse button to select the bootloader image and input the Flash address of this image.
(4) Click Gen Signature button to calculate signature for the image.
(5) Click Save Image button to generate the new image which contains Secure Boot information and signature.

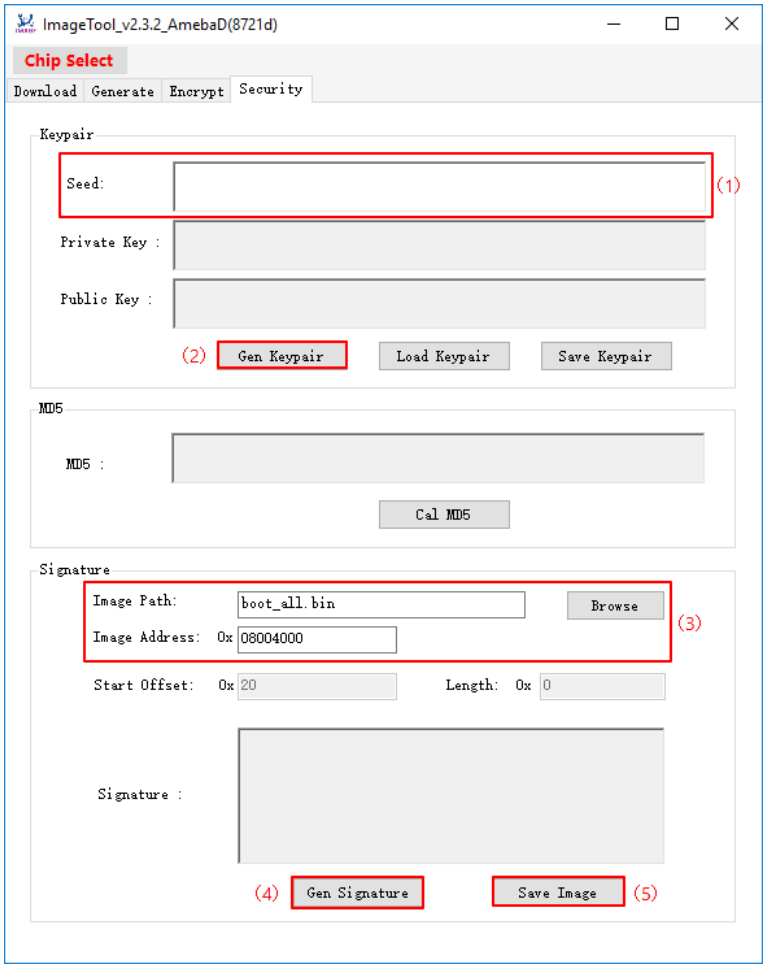


Fig8-10 ImageTool_v2 Security tabpage setting

9 Memory Layout

9.1 Flash Program Layout

The default Flash layout used in SDK is illustrated in Fig 9-1 and Table 9-1. Because bootloader will be called by ROM code, the address of bootloader cannot be changed. Other addresses except bootloader can all be changed by user, but before changing these addresses, users must make sure that they are familiar with AmebaD system and SDK.

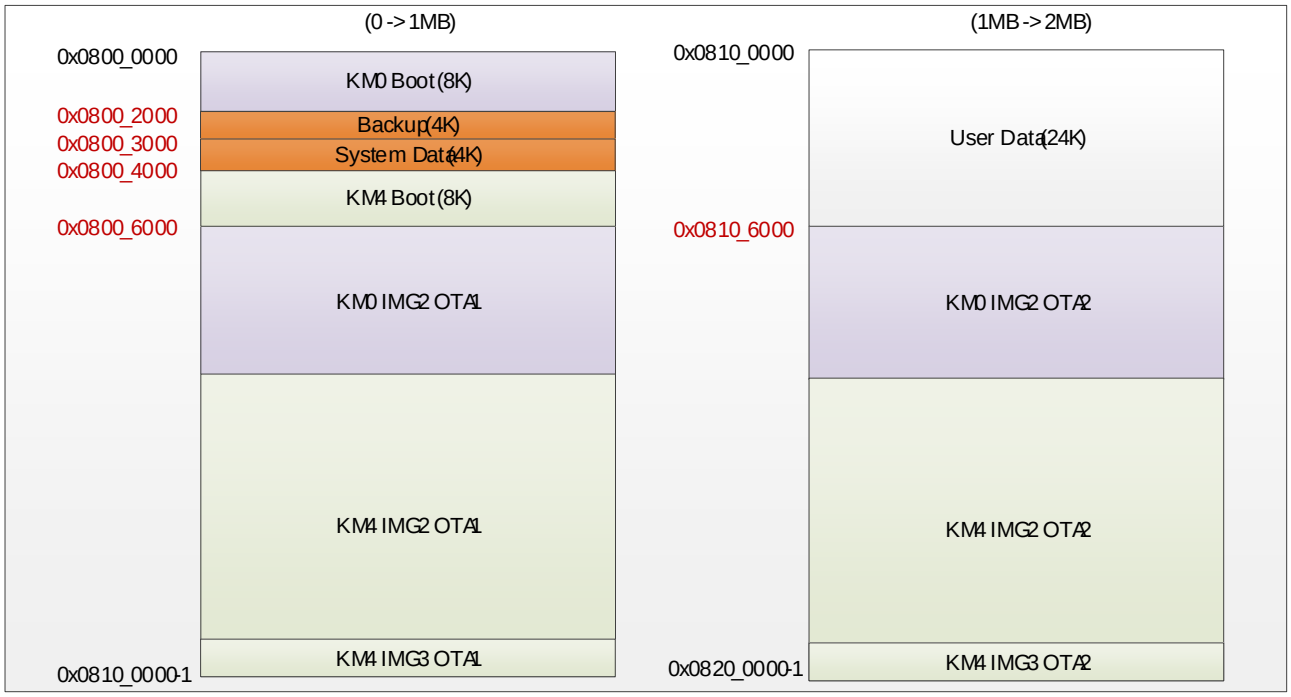


Fig 9-1 1M/2M Flash program layout

Table 9-1 1M/2M Flash program layout

Items	Start Address	Limited Address	Size	Description	Mandatory
KM0 Boot	0x0800_0000	0x0800_2000-1	8K	KM0 bootloader code/data	Yes
Backup	0x0800_2000	0x0800_3000-1	4K	Backup Flash area, reserved for system us When program system data, this area will be used as backup area.	Yes
System Data	0x0800_3000	0x0800_4000-1	4K	System Data, user should program this are carefully.	Yes
KM4 Boot	0x0800_4000	0x0800_6000-1	8K	KM4 bootloader code/data	Yes
KM0 IMG2 OTA1	0x0800_6000	0x0810_0000-1	1000K	KM0 image2 OTA1 code/data	No
KM4 IMG2 OTA1				KM4 image2 OTA1 code/data	No
KM4 IMG3 OTA1				RDP image OTA1 code/data	No
User Data	0x0810_0000	0x0810_2000-1	8K	Reserved for user data	No
	0x0810_2000	0x0810_5000-1	12K	BT FTL physical map	No
	0x0810_5000	0x0810_6000-1	4K	WIFI Fast Connect profile	No
KM0 IMG2 OTA2	0x0810_6000	0x0820_0000-1	1000K	KM0 image2 OTA2 code/data	No
KM4 IMG2 OTA2				KM4 image2 OTA2 code/data	No
KM4 IMG3 OTA2				RDP image OTA2 code/data	No

There is an image header for every image; it helps the bootloader to load code or data to the correct destination address, or is used to realize ECC secure boot.

9.1.1 Image Header

Normal bootloader's image header does not include secure boot signature. It is just used for loading code or data to the correct destination, as shown in Fig 9-2 and Table 9-2.

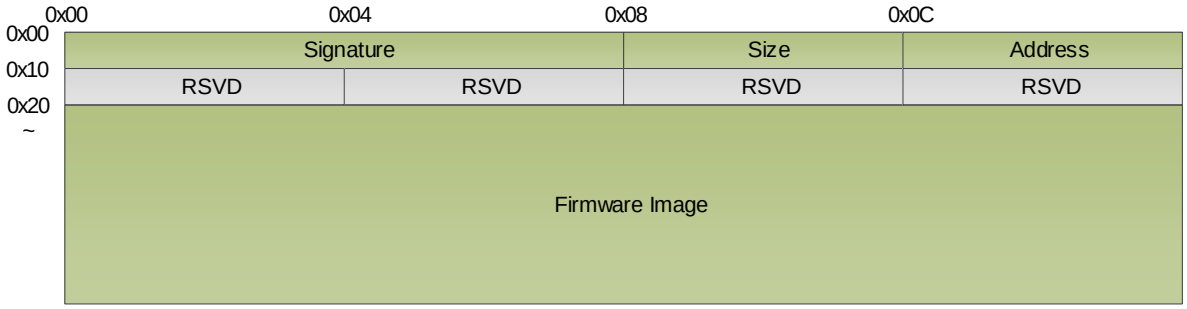


Fig 9-2 Normal image header

Table 9-2 Image header without secure boot

Items	Comment
Signature (8 bytes)	Bootloader Signature for Flash calibration
Size (4 bytes)	The size of Firmware Image
Address (4 bytes)	Code executes start address after boot

Image header with secure includes secure boot signature. It is used to realize ECC secure boot. For more information, refer to AN0410.

9.1.2 System Data (4K)

The System data layout is illustrated in Fig 9-3 and Table 9-3.

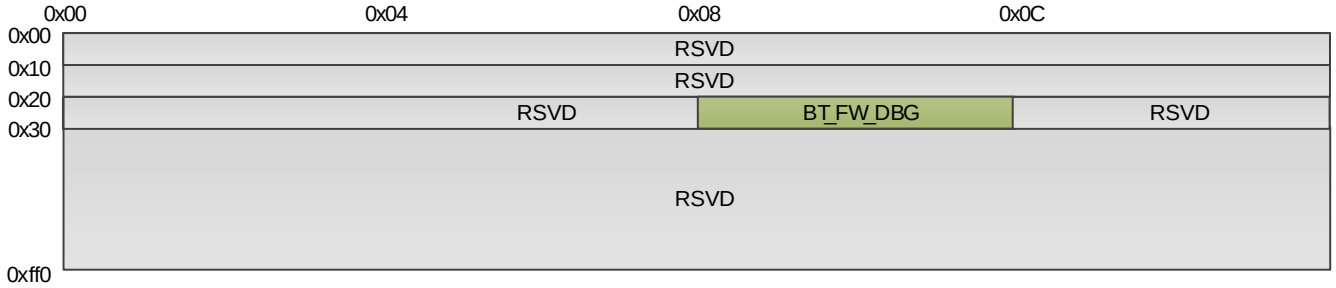


Fig 9-3 System data layout

Table 9-3 System Data Layout

Items	Default	Description
BT_FW_DBG	0xFFFFFFFF	Reserved for FW debug.

9.1.3 User Data

There are totally 24KB for user data. Parts of this region are defined by default SDK.

Table9-4 User data layout

Items	Range	Description
Reserved	0x0810_0000~0x0810_2000-1	Reserved for user data
BT FTL	0x0810_2000~0x0810_5000-1	Bluetooth FTL physical map. The start address is defined by phy_page_start_addr variable (rtl8721dhp_intfcfg.c). If Bluetooth is not enabled in your system, this can be used by users.
Wi-Fi Fast Connect Profile	0x0810_5000~0x0810_6000-1	Used to store Wi-Fi fast connect profile. Defined by FAST_RECONNECT_DATA_ADDR (platform_opts.h). If Wi-Fi fast connect function is not enabled in your system, this area can be used by users.

Warning: If Flash memory layout is modified and the 0x0800_2000~0x0800_5FFF is covered, it is important to modify the FTL physical map or Wi-Fi fast connect area address.

9.1.4 4M Flash Usage Example

For 4M Flash, an example of Flash layout is listed in Fig9-4. In this example, 2024K Flash size can be used for KM0 & KM4 image, and you can modify the layout according to your actual conditions. Because bootloader will be called by ROM code, the address of bootloader cannot be changed. Other addresses except bootloader can all be changed by user, but before you change these addresses, make sure that you are familiar with AmebaD system and SDK.

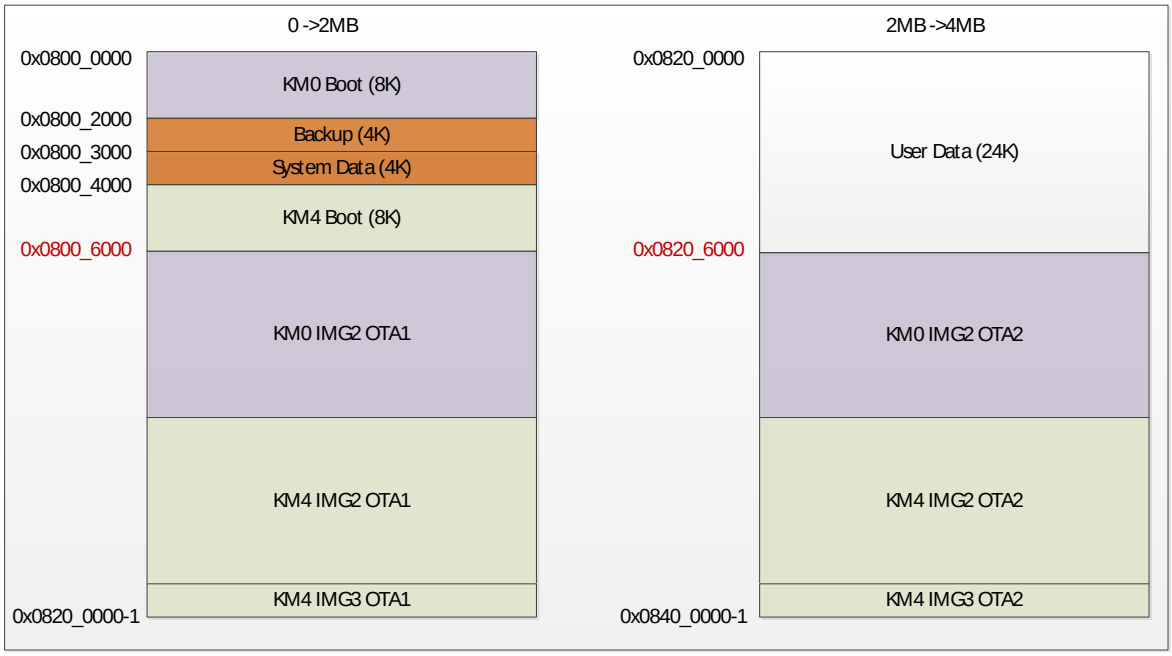


Fig9-4 4M Flash program layout example

Table9-5 4M Flash program layout example

Items	Start Address	Limited Address	Size	Description	Mandatory
KM0 Boot	0x0800_0000	0x0800_2000-1	8K	KM0 bootloader code/data	Yes
Backup	0x0800_2000	0x0800_3000-1	4K	Backup Flash area, reserved for system use. When program system data, this area will be used as backup area.	Yes
System Data	0x0800_3000	0x0800_4000-1	4K	System Data, user should program this area carefully.	Yes
KM4 Boot	0x0800_4000	0x0800_6000-1	8K	KM4 bootloader code/data	Yes

KM0 IMG2 OTA1	0x0800_6000	0x0820_0000-1	2024K	KM0 image2 OTA1 code/data	No
KM4 IMG2 OTA1				KM4 image2 OTA1 code/data	No
KM4 IMG3 OTA1				RDP image OTA1 code/data	No
User Data	0x0820_0000	0x0820_2000-1	8K	Reserved for user data	No
	0x0820_2000	0x0820_5000-1	12K	BT FTL physical map	No
	0x0820_5000	0x0820_6000-1	4K	Wi-Fi Fast Connect profile	No
KM0 IMG2 OTA2	0x0820_6000	0x0840_0000-1	2024K	KM0 image2 OTA2 code/data	No
KM4 IMG2 OTA2				KM4 image2 OTA2 code/data	No
KM4 IMG3 OTA2				RDP image OTA2 code/data	No

For 4M Flash, some configurations need to be modified in default SDK

OTA start address

The default OTA start address in SDK is set for 2M Flash, thus OTA address in rtl8721d_bootcfg.h and rtl8721d_ota.h should be modified when using 4M Flash.

(1) Modify OTA_Region in rtl8721d_bootcfg.h, change OTA2 region start address according to your actual Flash layout. When using 4M Flash program layout example as shown in Fig9-4, OTA region should be modified as below:

```
/*
* @brief   OTA start address. Because KM0 & KM4 IMG2 are combined, users only need to set the start address
*    of KM0 IMG2.
*/
BOOT_RAM_DATA_SECTION
u32 OTA_Region[2] = {
    0x08006000,      /* OTA1 region start address */
    0x08206000,      /* OTA2 region start address */
};
```

(2) Modify OTA address definition in rtl8721d_ota.h, redefine LS_IMG2_OTA2_ADDR Macro according to your actual Flash layout. When using 4M Flash program layout example as shown in Fig9-4, OTA address should be modified as below:

```
#define LS_IMG2_OTA1_ADDR  0x08006000                /* KM0 OTA1 start address*/
#define LS_IMG2_OTA2_ADDR  0x08206000                /* KM0 OTA2 start address*/
```

User data Flash address

In default SDK, 24KB user data is defined in Flash address 0x0800_2000~0x0800_5FFF, which is now used for BT FTL and Wi-Fi fast connect profile. For 4M Flash, if Bluetooth and Wi-Fi fast connect function are enabled in user system, and Flash address 0x0800_2000~0x0800_5FFF is covered, you need to modify BT FTL address and Wi-Fi fast connect profile address according to your actual Flash layout.

(1) For BT FTL start address, redefine ftl_phy_page_start_addr variable in rtl8721dhp_intfcfg.c, note that SPI FLASH address base 0x0800_0000 is subtracted when calculating ftl_phy_page_start_addr variable, and ftl_phy_page will consume 3*4=12KB Flash starting from ftl_phy_page_start_addr. When using 4M Flash program layout example as shown in Fig9-4, BT FTL start address should be modified as below:

```
const u32 ftl_phy_page_start_addr = 0x00202000;
```

(2) For Wi-Fi fast connect profile address, redefine FAST_RECONNECT_DATA Macro in platform_opts.h, and SPI FLASH address base 0x0800_0000 is subtracted when calculating FAST_RECONNECT_DATA Macro. Note that Wi-Fi fast connect profile address should not recover the 12KB Flash starting from ftl_phy_page_start_addr, as this region is reserved for BT FTL. When using 4M Flash program layout example as shown in Fig9-4, Wi-Fi fast connect profile address should be set to 0x202000+0x3000=0x205000

```
#define FAST_RECONNECT_DATA    0x205000
```

After the above modification, rebuild KM0 project and KM4 project, then download km0_boot_all.bin, km4_boot_all.bin and km0_km4_image2.bin.

9.2 SRAM Layout

9.2.1 KM4 SRAM Layout

The start address of KM4 SRAM is 0x1000_0000. The size is 512KB. The memory layout is illustrated in Fig 9-5 and Table 9-6.

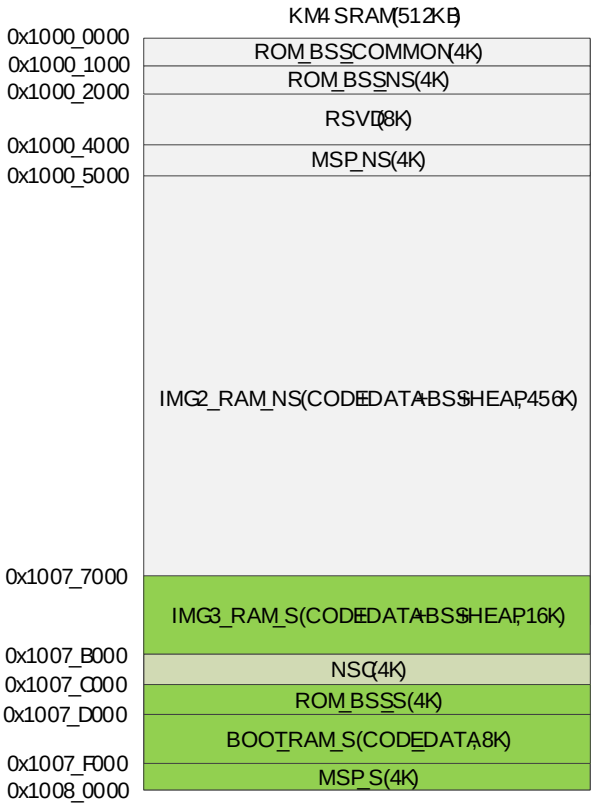


Fig 9-5 KM4 RAM program layout

Table 9-6 KM4 RAM program layout

Items	Start Address	End Address	Size	Description	Mandatory
ROM_BSS_COMMO	0x1000_0000	0x1000_1000-1	4K	ROM Secure and Non-secure common .bss	Yes
ROM_BSS_NS	0x1000_1000	0x1000_2000-1	4K	ROM Non-secure common .bss	Yes
RSVD	0x1000_2000	0x1000_4000-1	8K	Reserved for Non-secure MSP & Non-secure ROM .bss	No
MSP_NS	0x1000_4000	0x1000_5000-1	4K	Non-secure Main Stack Pointer	Yes
IMG2_RAM_NS	0x1000_5000	0x1007_7000-1	456K	Image2 Non-secure RAM includes CODE, DATA, BSS and HEAP	No
IMG3_RAM_S	0x1007_7000	0x1007_B000-1	16K	Image3 Secure RAM includes CODE, DATA, BSS and HEAP	No
NSC	0x1007_B000	0x1007_C000-1	4K	Images Secure function call entries	No
ROM_BSS_S	0x1007_C000	0x1007_D000-1	4K	ROM Secure only .bss	Yes
BOOT_RAM_S	0x1007_D000	0x1007_F000-1	8K	KM4 Secure bootloader, includes CODE and DATA	Yes
MSP_S	0x1007_F000	0x1008_0000-1	4K	Secure Main Stack Pointer	Yes

Note: If RDP is not enabled, the IMG3_RAM_S and NSC are non-secure and would be merged into IMG2_RAM_NS. So the IMG2_RAM_NS region would be [0x1000_5000, 0x1007_C000-1].

9.2.2 KM0 SRAM Layout

The start address of KM0 SRAM is 0x0008_0000. The size is 64KB. The memory layout is illustrated in Fig 9-6 and Table 9-7.

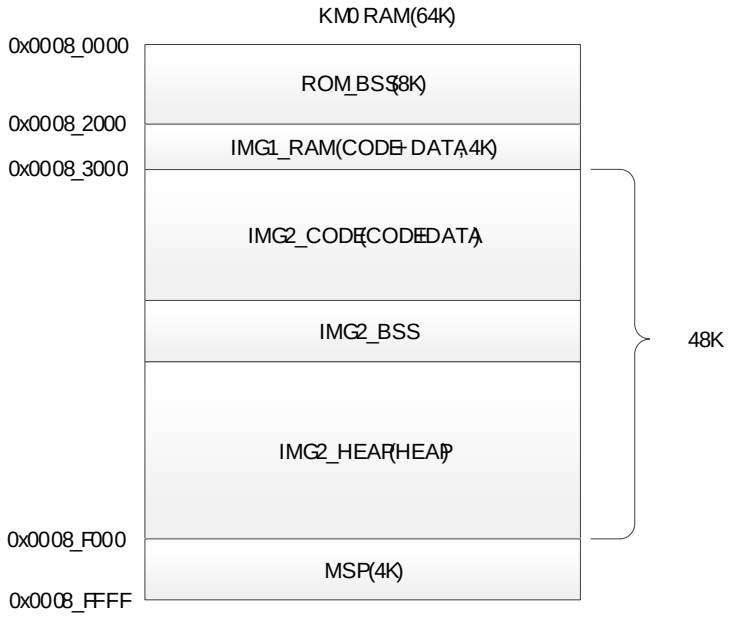


Fig 9-6 KM0 RAM program layout

Table 9-7 KM0 RAM program layout

Items	Start Address	End Address	Size	Description	Mandatory
ROM_BSS	0x0008_0000	0x0008_2000-1	8K	ROM .bss	Yes
IMG1_RAM	0x0008_2000	0x0008_3000-1	4K	KM0 bootloader SRAM, including CODE and DATA	Yes
IMG2_RAM	0x0008_3000	0x0008_F000-1	48K	KM0 image2 SRAM, including CODE, DATA, BSS and H	No
MSP	0x0008_F000	0x0009_0000-1	4K	Main Stack Pointer	Yes

9.3 PSRAM Layout

The start address of KM4 PSRAM is 0x0200_0000. The total size is 4MB. The memory layout is illustrated in Fig 9-7 and Table 9-8.

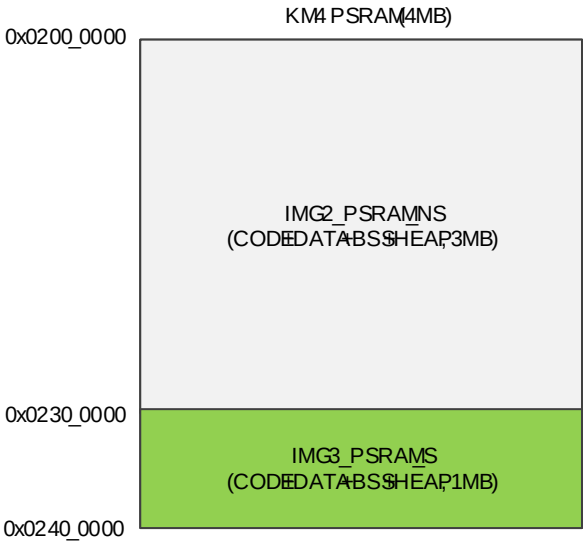


Fig 9-7 KM4 PSRAM program layout

Table 9-8 KM4 PSRAM program layout

Items	Start Address	End Address	Size	Description	Mandatory
IMG2_PSRAM_NS	0x0200_0000	0x0230_0000-1	3MB	Image2 Non-secure PSRAM includes CODE, DATA, BSS and HEAP	No
IMG3_PSRAM_S	0x0230_0000	0x0240_0000-1	1MB	Image3 Secure PSRAM includes CODE, DATA, BSS and HEAP	No

Note: If TrustZone is not enabled, the whole PSRAM would be non-secure. So the IMG2_PSRAM_NS region would be [0x0200_0000, 0x0240_0000-1].

9.4 Retention SRAM

Retention SRAM would not be power off even when system enters deep sleep mode. As a result, it is designed to retain data or code which would be lost in SRAM for some cases, such as deep sleep.

9.4.1 Retention SRAM Layout

The start address of Retention SRAM is 0x000C_0000. The size is 1KB. The memory layout is listed in Table 9-9.

Table 9-9 Retention SRAM layout

Items	Start Address	End Address	Size	Description	Mandatory
System Area	0x000C_0000	0x000C_0080-1	128B	Used by Realtek. The size is indicated by the second dw (0x000C_0004).	Yes
User Area	0x000C_0080	0x000C_0300-1	176B	Refer to RRAM_TypeDef and RRAM_USER_RSVD [user_size] is reserved for customer use.	No
Wi-Fi Area	0x000C_0300	0x000C_0400-1	720B	Realtek Wi-Fi FW use	Yes

9.4.2 User Area

RRAM_USER_RSVD is reserved for customer use. The size for user area is 124B now, it may be changed latter.

```
/*****************************************************************************//**
 * @defgroup AMEBAD_RRAM
 * @{
 * @brief AMEBAD_RRAM Declaration
 * @ the Max space for RRAM_TypeDef is 0xB0, user can alloc space from RRAM_USER_RSVD
 *****************************************************************************/
typedef struct {
    uint8_t RRAM_WIFI_STATUS; /* RETENTION_RAM_SYS_OFFSET 0x80 */
    uint8_t RRAM_WIFI_P_SECURITY;
    uint8_t RRAM_WIFI_G_SECURITY;
    uint8_t RRAM_RSVD1;

    uint32_t RRAM_NET_IP;
    uint32_t RRAM_NET_GW;
    uint32_t RRAM_NET_GW_MASK;

    uint32_t FLASH_ID2;             /*offset 0x90*/
    uint32_t FLASH_cur_cmd;
    uint32_t FALSH_quad_valid_cmd;
    uint32_t FALSH_dual_valid_cmd;
    uint32_t FLASH_QuadEn_bit;      /*offset 0xA0*/
    uint32_t FLASH_StructInit;

    uint8_t FLASH_phase_shift_idx;
    uint8_t FLASH_rd_sample_phase_cal;
    uint8_t FLASH_class;
    uint8_t FLASH_cur_bitmode;

    uint8_t FLASH_ClockDiv;
    uint8_t FLASH_ReadMode;
    uint8_t FLASH_pseudo_prm_en;
    uint8_t FLASH_addr_phase_len;

    uint8_t FLASH_cmd_wr_status2;   /*offset 0xB0*/
    uint8_t FLASH_rd_dummy_cyle0;
    uint8_t FLASH_rd_dummy_cyle1;
    uint8_t FLASH_rd_dummy_cyle2;

    uint8_t RRAM_USER_RSVD[124];          /*usr can alloc from this RSVD space*/

} RRAM_TypeDef;
/** @} */
```

9.5 OTA Layout

9.5.1 OTA Program Layout

The OTA program layout is shown in Fig 9-8.

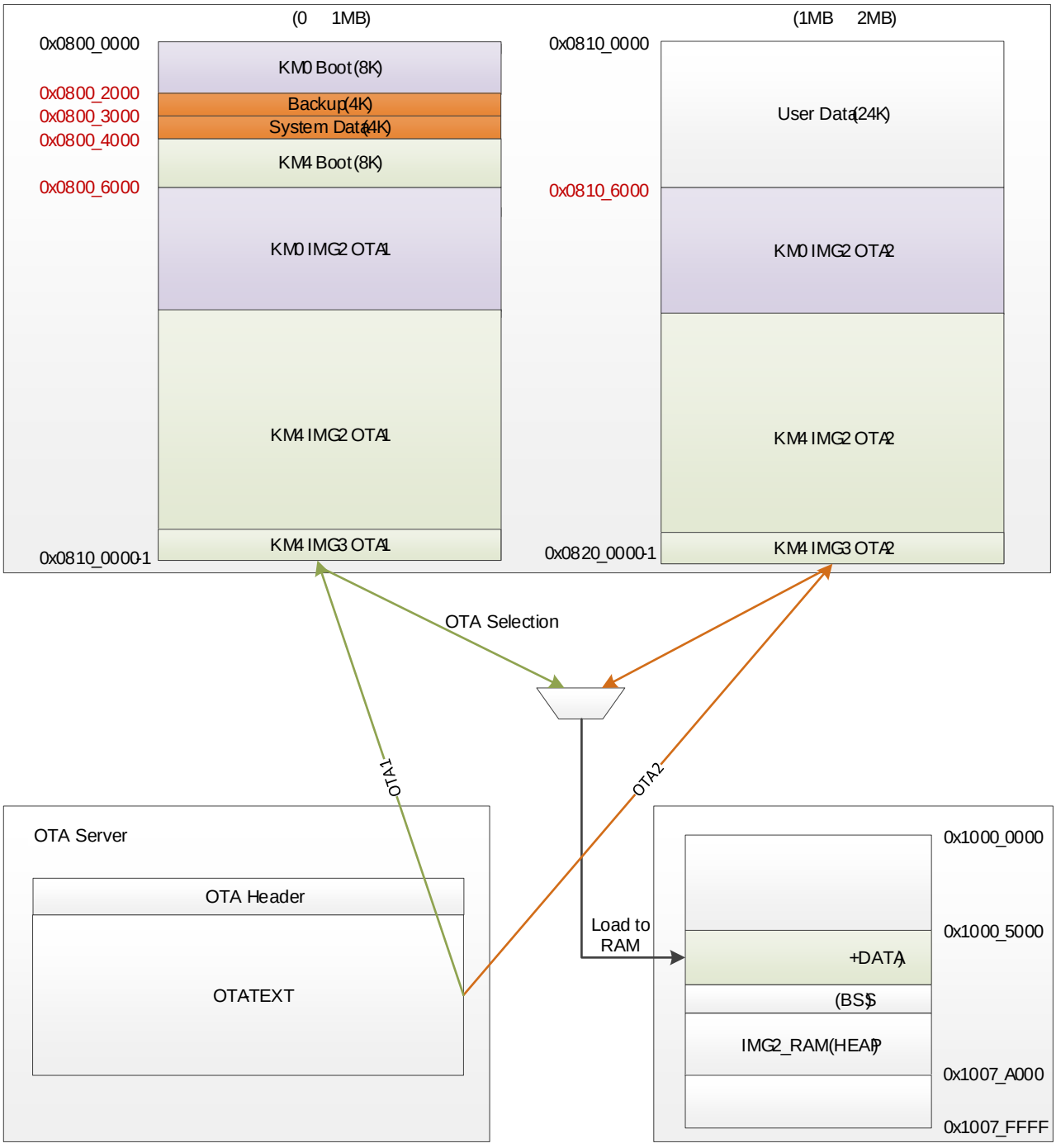


Fig 9-8 OTA program layout

9.5.2 OTA Header Layout

The OTA header layout is given in Fig 9-9 and Table 9-10.

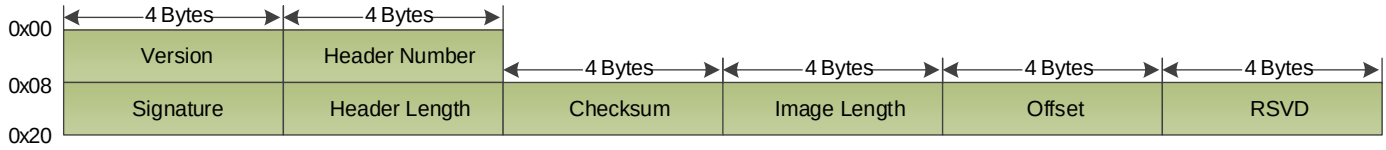


Fig 9-9 OTA header layout

Table 9-10 OTA header layout

Items	Address Offset	Size	Description
Version	0x00	4 bytes	The version of OTA Header, default 0xFFFFFFFF
Header Number	0x04	4 bytes	The number of OTA Header. For AmebaD, this value is 0x01
Signature	0x08	4 bytes	OTA Signature is string. For AmebaD, this value is 0x18
Header Length	0x0C	4 bytes	The length of OTA header. For AmebaD, this value is 0x18
Checksum	0x10	4 bytes	The checksum of OTA image
ImageLength	0x14	4 bytes	The size of OTA image
Offset	0x18	4 bytes	The start position of OTA image in current image
RSVD	0x1C	4 bytes	Reserved

9.6 TrustZone Memory Layout

All addresses are Secure when boot. Some areas are set to Non-secure using SAU & IDAU in bootloader.

When TrustZone is enabled, the default security setting in AmebaD SDK is shown in Fig 9-10.

When TrustZone is disabled, SDK would set IMG3_RAM_S, NSC and IMG3_PSRAM_S to Non-Secure. Other secure areas used by ROM code are mandatory.

Fig9-10 TrustZone layout

10 Boot Process

10.1 Features

- On-chip boot ROM
- Containing the bootloader with In-System Programming (ISP) facility
- Secure boot process using Elliptic Curve Cryptography (ECC)
- Suspend resume process

10.2 Boot Address

After reset, CPU will boot from vector table start address, and address is fixed by hardware. KM0 and KM4 boot from different addresses as Table10-1 lists.

Table10-1 Boot ROM address

CPU	Address	Type
KM4	0x1010_0000	KM4 ITCM ROM
KM0	0x0000_0000	KM0 ITCM ROM

10.3 Pin Description

The parts support ISP via LOGUART (PA[7] & PA[8]). The ISP mode, given in Table10-2, is determined by the state of the PA7 pin at boot time.

Table10-2 ISP mode

Boot Mode	PA[7] (UART_DOWNLOAD)	Description
No ISP	HIGH	ISP bypassed. Part attempts to boot from flash
ISP	LOW	Part enters ISP via LOGUART

10.4 Boot Flow

The boot flow of AmebaD is shown in Fig10-1. After a power-up or hardware reset, hardware will boot KM0 at clock 26MHz or 20MHz based on XTAL clock set in Fuse. The boot process is handled by the on-chip boot ROM (0x0000_0000) and is always executed by the KM0 core. After the KM0 bootloader code, the KM0 will set up the environment for the KM4.

- Power on PLL
- Flash calibration @100MHz
- Power on KM4
- After that the KM4 can be taken out of reset by the KM0.
- KM4 will boot from ROM code 0x1010_0000.

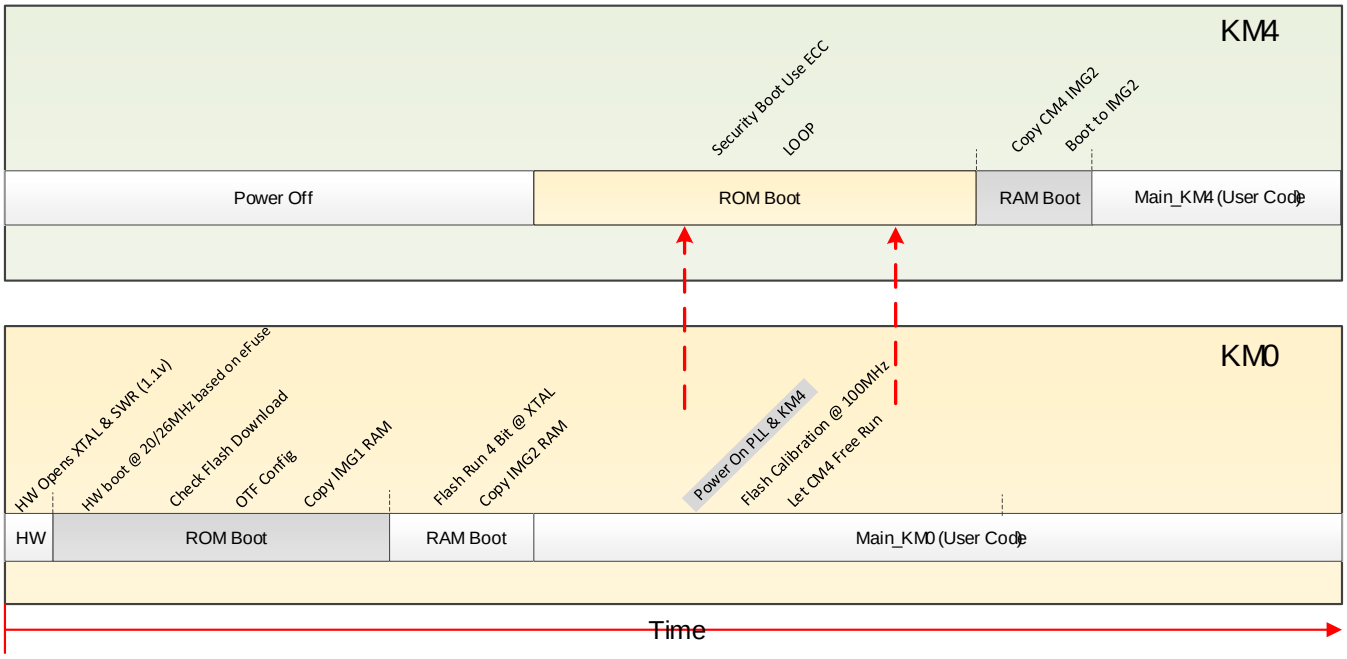


Fig10-1 Boot flow

10.5 Boot Time

Boot time was tested when this document was written, the result may be changed as the changing of SDK.

Table 10-3 Boot Time

CPU	Normal BOOT (ms)	Deepsleep WAKEUP (ms)	Comment
KM0 BOOT	30	18	
KM4 BOOT	6+x	3+x	x=image2 size/flash rate If image2 size = 100KB, Flash rate = 80Mbps & 2bit mode: x=100*1024*8/160=5.1ms (100KB is SDK current image2 size)
WIFI ON	40	40	

10.6 General Description

The bootloader controls initial operation after reset or powered on and also provides the means to program the flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device.

Assuming that power supply pins are at their nominal levels when the rising edge of RESET pin is generated, then boot pins are sampled and the decision whether to continue with user code or ISP handler is made. If the pins are sampled LOW, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution, a search is made for a valid user program. If a valid user program is found, then the execution control is transferred to it. If a valid user program is not found, the dead loop is invoked.

10.6.1 KM0 ROM Boot

The KM0 ROM boot process is shown in Fig10-2.

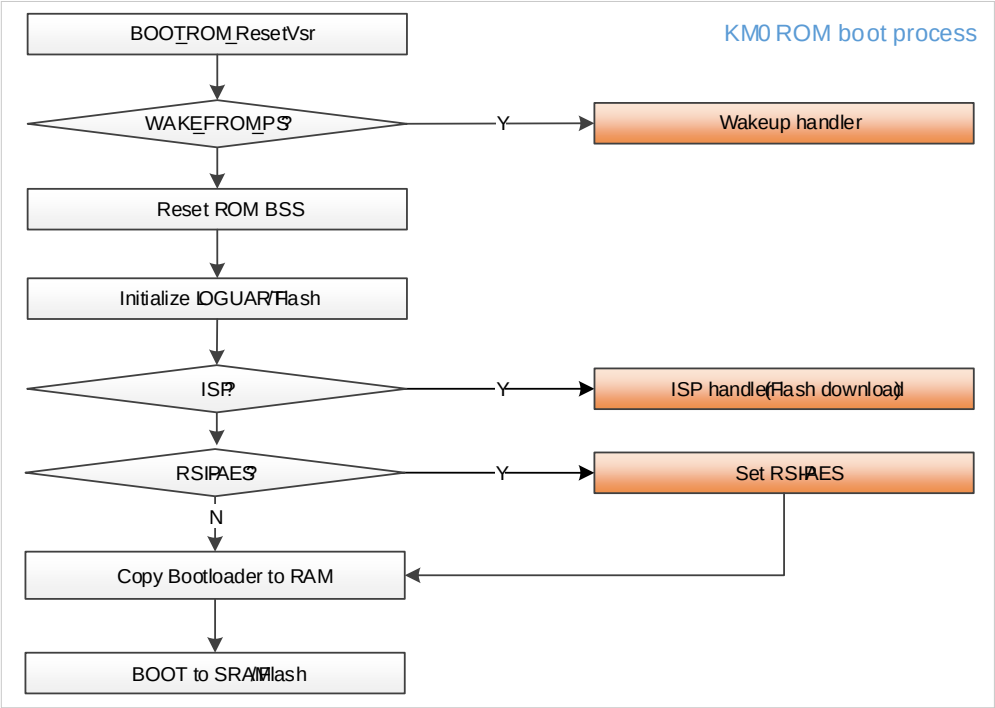


Fig10-2 KM0 ROM boot process

10.6.2 KM4 ROM Boot

The KM4 ROM boot process is shown in Fig10-3.

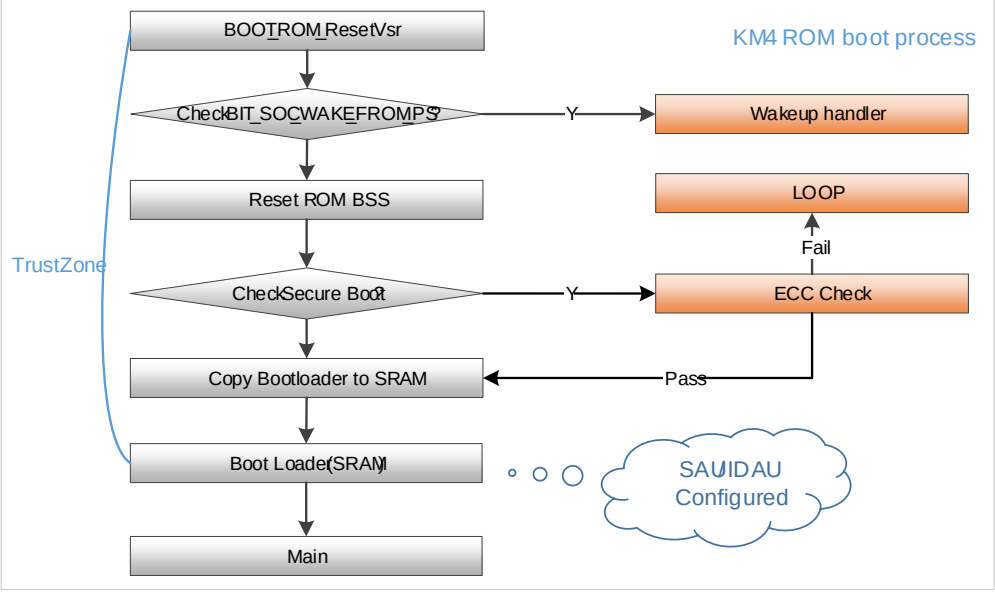


Fig10-3 KM4 ROM boot process

10.7 Boot Reason APIs

Source code: rtl8721d_backup_reg.c

API	Introduction
< BOOT_Reason >	Get boot reason

Table 10-4 Boot reason definition

Bit	Comment
BIT_BOOT_BOD_RESET_HAPPEN	BOD Reset
BIT_BOOT_DSLP_RESET_HAPPEN	Wakeup form deep sleep
BIT_BOOT_KM4WDG_RESET_HAPPEN	KM4 watchdog reset
BIT_BOOT_KM4SYS_RESET_HAPPEN	KM4 system reset
BIT_BOOT_WDG_RESET_HAPPEN	KM0 watchdog reset
BIT_BOOT_SYS_RESET_HAPPEN	KM0 system reset

11 File System

11.1 Introduction

This chapter introduces how to run FAT file system (FatFs) example on AmebaD, including FatFs examples of Flash and SD card. The example source code of FatFs is located in component\common\example\fatfs.

11.2 FatFs on Flash

11.2.1 Software Setup

(1) Change some configurations in project\realtek_amebaD_cm4_gcc_verification\inc\platform_opts.h as follows:
   a) Set the parameter CONFIG_EXAMPLE_FATFS to 1
   b) Set FATFS_DISK_FLASH to 1, while FATFS_DISK_USB and FATFS_DISK_SD to 0.

```
/* For FATFS example*/
#define CONFIG_EXAMPLE_FATFS            1
#if CONFIG_EXAMPLE_FATFS
#define CONFIG_FATFS_EN      1
#if CONFIG_FATFS_EN
// fatfs version
#define FATFS_R_10C
// fatfs disk interface
#define FATFS_DISK_USB  0
#define FATFS_DISK_SD   0
#define FATFS_DISK_FLASH    1
#endif
#endif
```

(2) Change some parameters in component\common\file_system\fatfs\r0.10c\include\ffconf.h as follows:
   a) Set _MAX_SS to 4096 to define the maximum sector size supported
   b) Set _USE_MKFS to 1 to enable f_mkfs() function which creates FatFs volume on Flash

```
#define   _MIN_SS     512
#define   _MAX_SS     4096//512
```

```
#define   _USE_MKFS     1    /* 0:Disable or 1:Enable */
```

(3) Change Flash memory base in component\common\file_system\fatfs\disk_if\src\flash_fatfs.c l· Ŏ Ŏ 2 NJ Ř λ y˜ Ǝ ú 2 dz ă Ŝ NJ Ω ă NJ Ŝ

```
#define FLASH_APP_BASE  0x100000 // 0x440000
```

(4) Check the stack size that needs to be at least 4096 in component\common\example\fatfs\example_fatfs.c

```
#define STACK_SIZE          4096
```

(5) Rebuild the project and download image files to AmebaD.

11.2.2 FatFs Bin File Generation

The commands to generate FatFs bin file are as follows:
(1) root@ubuntu # dd if=/dev/zero of=test.bin count=64 bs=1KB
(2) root@ubuntu # fdisk test.bin
(3) root@ubuntu # mkfs.msdos -S 4096 -F 12 test.bin
(4) root@ubuntu # mcopy -i test.bin ./fs/hello.txt ::hello.txt
(5) root@ubuntu # sudo mount test.bin ./fs
(6) sudo umount ./fs

In step (1), test.bin is created which has 64 blocks and each block is 1KB.

In step (2), test.bin is partitioned. Some additional options have to be chosen in this step in order to create a new partition.
In step (3), a MS-DOS file system is built.
In step (4), files that users want to store are copied into test.bin. In this step, hello.txt is stored in test.bin.
In step (5), test.bin is mounted to file folder fs.
In step (6), after unmounting test.bin, the FatFs file is generated.
Users should find other related information from the internet, and copy test.bin into user data area of Flash finally.

11.3 FatFs on SD Card

11.3.1 Hardware Setup

Connect your AmebaD EVB board to PC with micro-USB cable, and then plug in a compatible SD card to the card connector. Note that the terminal of the middle of J24 need to be connected to ground. Connection of J329 is shown as Fig 11-1.

Fig 11-1 Hardware setup

11.3.2 Software Setup

(1) Change some configurations in project\realtek_amebaD_va0_example\inc\inc_hp\platform_opts.h as follows:
   a) Set the parameter CONFIG_EXAMPLE_FATFS to 1
   b) Set FATFS_DISK_SD to 1 while FATFS_DISK_USB and FATFS_DISK_FLASH to 0.

```
/* For FATFS example*/
#define CONFIG_EXAMPLE_FATFS            1
#if CONFIG_EXAMPLE_FATFS
#define CONFIG_FATFS_EN     1
#if CONFIG_FATFS_EN
// fatfs version
#define FATFS_R_10C
// fatfs disk interface
#define FATFS_DISK_USB  0
#define FATFS_DISK_SD   1
#define FATFS_DISK_FLASH    0
#endif
#endif
```

(2) Change some parameters in component\common\file_system\fatfs\r0.10c\include\ffconf.h as follows:
   a) Set _MAX_SS to 512 to define the maximum sector size supported
   b) Set _USE_MKFS to 1.

```
#define   _MIN_SS     512
#define   _MAX_SS     512
```

```
#define   _USE_MKFS     1   /* 0:Disable or 1:Enable */
```

(3) Change some parameters in component\soc\realtek\amebad\fwlib\usrcfg\rtl8721dhp_intfcfg.c as follows:

```
SDIOHCFG_TypeDef sdioh_config = {
    .sdioh_bus_speed = SD_SPEED_HS,
    .sdioh_bus_width = SDIOH_BUS_WIDTH_4BIT,
    .sdioh_cd_pin = _PA_6,
    .sdioh_wp_pin = _PNC,
};
```

(4) Rebuild the project and download image files to AmebaD. FAT file of SD card can be generated and copied directly by PC.

12 Inter Processor Communication (IPC)

12.1 Introduction

Inter-Processor Communication (IPC) hardware is designed for the purpose of making two CPUs communicate with each other. The system architecture is shown in Fig 12-1.

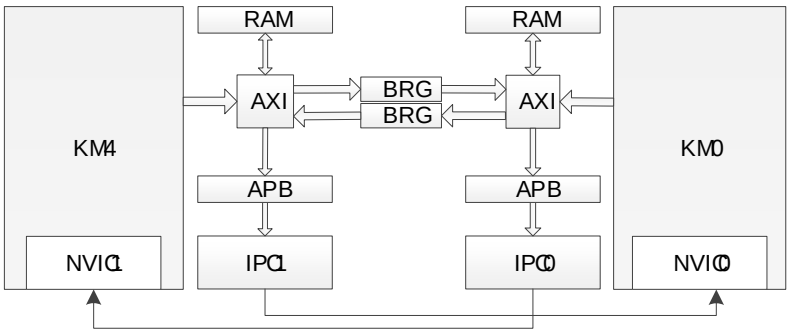


Fig 12-1 IPC system architecture

If you want KM0 to execute its IPC handler, IPC interrupt of KM4 should be enable and the status should be set, vice versa.

12.2 How to Use IPC

It is complex because both KM0 and KM4 should be set to realize IPC. SDK has provided API and configuration file to simplify the procedure. Take KM4 requests an interrupt into direction KM0 for example.

(1) Add new item in KM0 core section of ipc_init_config[] which is in rtl8721d_ipccfg.c for selected channel. IRQ handler and data should be set and defined. If message exchange is needed, you should specify the message data or point.
Note that the message type for a specific channel should be the same on both ARM_CORE_CM4 side and ARM_CORE_CM0 side. In the following figure, although IRQFUNC of channel 3 on ARM_CORE_CM4 side is not used, its message type should be set the same as ARM_CORE_CM0 side.

```
const IPC_INIT_TABLE ipc_init_config[] =
{
#if defined (ARM_CORE_CM4)
    //USER_MSG_TYPE     IRQFUNC                         IRQDATA
    {IPC_USER_DATA,     (VOID*) shell_switch_ipc_int,   (VOID*) NULL},//channel 0: IPC_INT_CHAN_SHELL_SWITCH
    {IPC_USER_DATA,     (VOID*) NULL,                   (VOID*) NULL},//channel 1: IPC_INT_CHAN_WIFI_FW
    {IPC_USER_DATA,     (VOID*) FLASH_Write_IPC_Int,    (VOID*) NULL},//channel 2: IPC_INT_CHAN_FLASHPG_REQ
    {IPC_USER_POINT,    (VOID*) NULL,                   (VOID*) NULL},//channel 3: IPC_INT_KM4_TICKLESS_INDICATION
#else
    //USER_MSG_TYPE     IRQFUNC                         IRQDATA
    {IPC_USER_DATA,     (VOID*) shell_switch_ipc_int,   (VOID*) NULL},//channel 0: IPC_INT_CHAN_SHELL_SWITCH
    {IPC_USER_DATA,     (VOID*) driver_fw_flow_ipc_int, (VOID*) IPCM4_DEV},//channel 1: IPC_INT_CHAN_WIFI_FW
    {IPC_USER_DATA,     (VOID*) FLASH_Write_IPC_Int,    (VOID*) NULL},//channel 2: IPC_INT_CHAN_FLASHPG_REQ
    {IPC_USER_POINT,    (VOID*) km4_tickless_ipc_int,   (VOID*) NULL},//channel 3: IPC_INT_KM4_TICKLESS_INDICATION
#endif
    {0xFFFFFFFF,        OFF,                            OFF},    /* Table end */
};
```

(2) SDK will enable the IPC interrupt of KM4 for the channel according to ipc_init_config[], and register the corresponding KM0 IRQ handler and data for the channel.
(3) When KM4 wants to request an interrupt into direction KM0, it should call ipc_send_message() and specify the channel number and message.
(4) Corresponding KM0 IRQ handler will be executed, in which call ipc_get_message() to get message if need.

Note: User IPC channel 16~31 and IPC semaphore index 8~15, Channel 0~15 and semaphore index 0~7 are reserved for Realtek use.

12.3 IPC Program API

12.3.1 ipc_table_init

Items	Description
Introduction	Initialize IPC channels according to ipc_init_config[]
Parameters	N/A
Return	N/A

12.3.2 ipc_send_message

Items	Description
Introduction	Request interrupt to target CPU by specified channel and exchange messages between KM0 and KM4
Parameters	IPC_ChNum: the IPC channel number Message: the message to be exchanged, and should not be stored in stack
Return	N/A

Note:

The message supports two types: data or point. The point is used to point to complex message structure

The message is shared between two CPUs, so the point can't be stored in stack.

12.3.3 ipc_get_message

Items	Description
Introduction	Get IPC message from specified channel.
Parameters	IPC_ChNum: the IPC channel number
Return	Message: the message to be exchanged

Note: Cache in the system is write-through type, so the corresponding data cache must be invalidated before data fetch.

12.4 IPC Driver Code

12.4.1 IPC_INTConfig

Items	Description
Introduction	Enable or disable the specified IPC channel interrupts
Parameters	IPCx: IPCM0_DEV for KM0 IPCM4_DEV for KM4 IPC_ChNum: the channel that want to be set. This parameter must be set to a value of 0 ~ 31. NewState: ENABLE DISABLE
Return	N/A

12.4.2 IPC_IERSet

Items	Description
Introduction	Set IER of specified IPC channel interrupts

Parameters	IPCx: IPCM0_DEV for KM0 IPCM4_DEV for KM4 IPC_Ch: the channel that want to be enable
Return	N/A

12.4.3 IPC_IERGet

Items	Description
Introduction	Get IER of specified IPCx
Parameters	IPCx: IPCM0_DEV for KM0 IPCM4_DEV for KM4
Return	The IER of specified IPCx

12.4.4 IPC_INTRequest

Items	Description
Introduction	Request a core-to-core interrupt
Parameters	IPCx: IPCM0_DEV for KM0 IPCM4_DEV for KM4 IPC_ChNum: the channel that want to request interrupt. This parameter must be set to a value of 0 ~ 31.
Return	N/A

12.4.5 IPC_INTClear

Items	Description
Introduction	Clear a core-to-core interrupt
Parameters	IPCx: IPCM0_DEV for KM0 IPCM4_DEV for KM4 IPC_ChNum: the channel that want to clear interrupt. This parameter must be set to a value of 0 ~ 31.
Return	N/A

12.4.6 IPC_INTGet

Items	Description
Introduction	Get a core-to-core interrupt status
Parameters	IPCx: IPCM0_DEV for KM0 IPCM4_DEV for KM4
Return	The ISR of specified IPCx

12.4.7 IPC_CPUID

Items	Description
Introduction	Get the current CPU ID
Parameters	N/A
Return	CPU ID 0: KM0

	1: KM4

12.4.8 IPC_SEMGet

Items	Description
Introduction	Get a core-to-core hardware semaphore
Parameters	SEM_Idx: the semaphore index that want to get. This parameter must be set to a value of 0 ~ 15. 0 ~ 7: Reserved for Realtek use 8 ~ 15: Reserved for Customer use
Return	Result TRUE FALSE

12.4.9 IPC_SEMFree

Items	Description
Introduction	Free a core-to-core hardware semaphore
Parameters	SEM_Idx: the semaphore index that want to free. This parameter must be set to a value of 0 ~ 15. 0 ~ 7: Reserved for Realtek use 8 ~ 15: Reserved for Customer use
Return	Result TRUE FALSE

12.4.10 IPC_INTHandler

Items	Description
Introduction	The common IPC interrupt handler
Parameters	Data: the data passed to the IRQ Handler
Return	0

Note: Before entering the specified IRQ handler, the common IPC interrupt handler clears the pending interrupt first. There is no need to clear the pending interrupt in user defined IRQ handler

12.4.11 IPC_INTUserHandler

Items	Description
Introduction	Register a user interrupt handler for a specified IPC channel
Parameters	IPC_ChNum: the channel that want to register IRQ handler. This parameter must be set to a value of 0 ~ 31. IrqHandler: the IRQ handler to be assigned to the specified IPC channel IrqData: the pointer will be passed to the IRQ handler
Return	N/A

13 OTA Firmware Update

13.1 Introduction

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

RTL8721d provides solutions to implement OTA firmware upgrade from local server or cloud. Next we will introduce the design and usage of OTA from local server. It has well transportability to porting to OTA applications from cloud.

13.2 RSIP MMU

The Flash MMU diagram is shown in Fig 13-1.

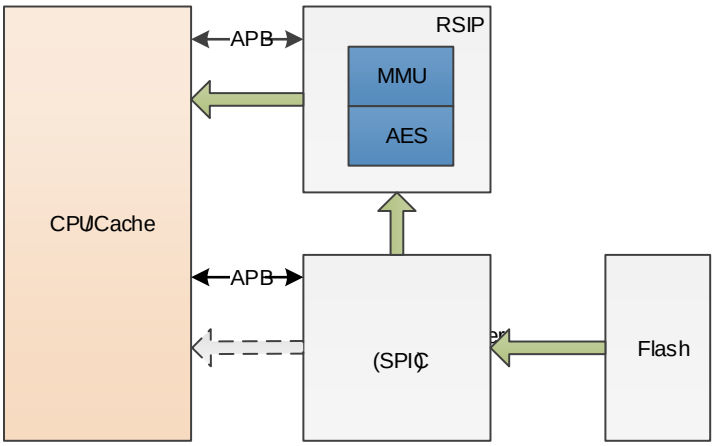


Fig 13-1 Flash MMU

The RSIP MMU can perform virtual-to-physical memory address translation. This can be used to map an external Flash memory to a certain virtual memory area. For example, If you want to access physical page 4 ~ 7 through virtual page 0 ~ 3, you can use a MMU entry to map virtual page 0 ~ 3 to physical page 4 ~ 7 like Fig 13-2.

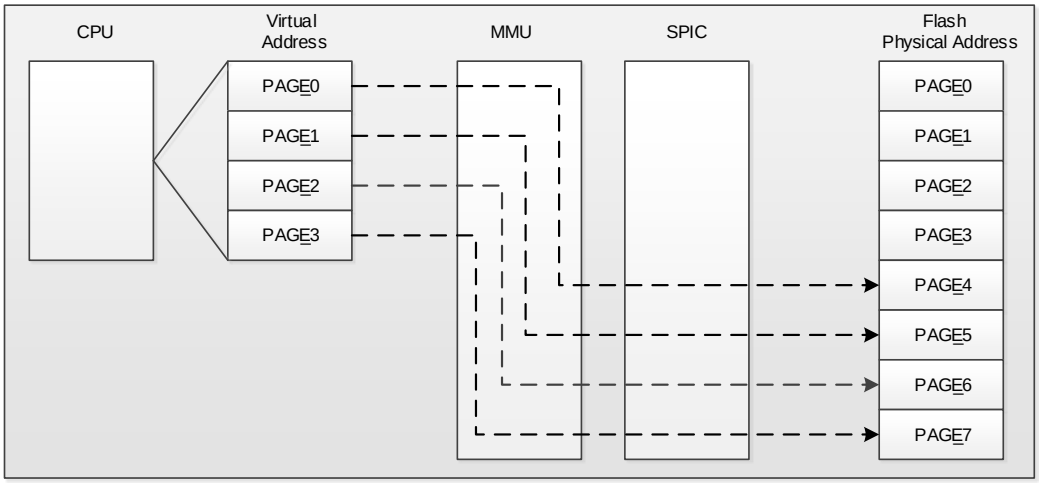


Fig 13-2 MMU virtual address to physical address

AmebaD provides 8 MMU Entries. If virtual address is not included in the MMU entry, use virtual address as physical address. If virtual address is included in the MMU entry, physical address should be VAddress + MMU_ENTRYx_OFFSET.

MMU is implemented to facilitate OTA update procedure as Fig13-3 shows. Fig13-3 is an example for 2M Flash OTA, we need 2 MMU entries in this example.

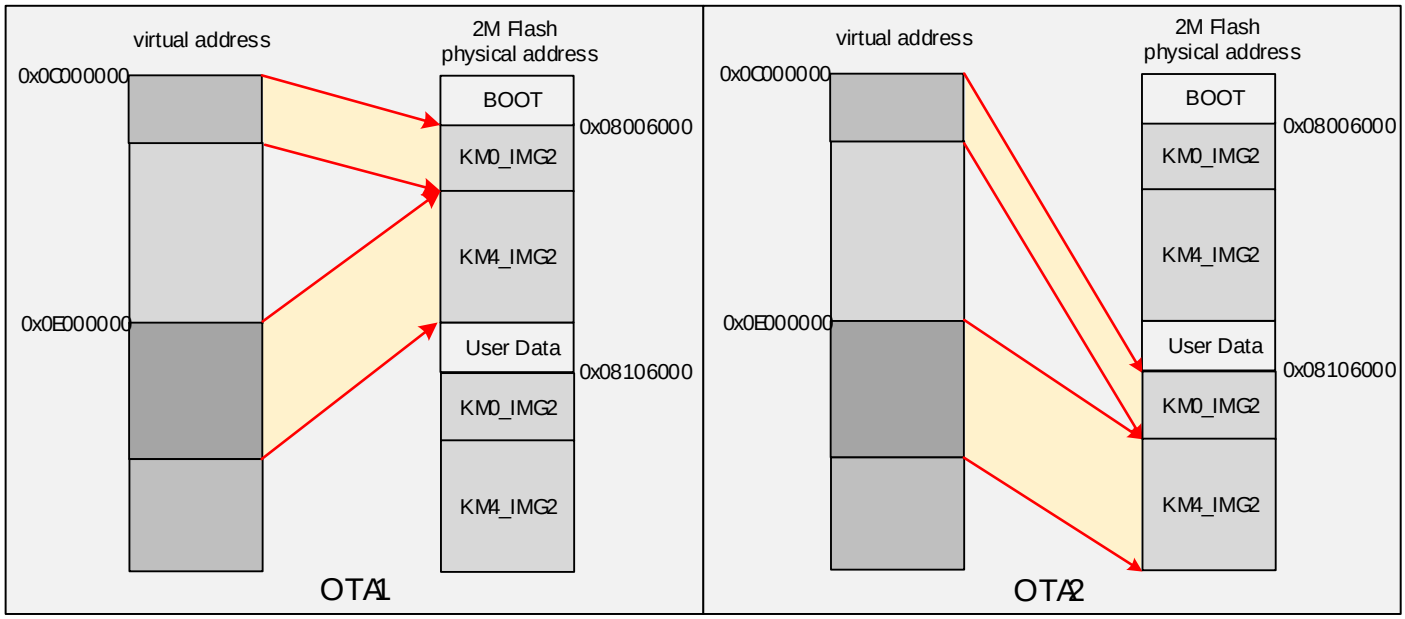


Fig13-3 2M Flash OTA MMU

MMU entry should be set as Table13-1.

Table13-1 OTA under MMU

Register Type	Register	OTA1	OTA2
MMU Entry0	MMU_ENTRY0_STRADDR	0x0C00_0000	0x0C00_0000
	MMU_ENTRY0_ENDADDR	0x0C00_0000 + KM0 IMG2 size	0x0C00_0000 + KM0 IMG2 size
	MMU_ENTRY0_OFFSET	0x0C00_0000 - 0x0800_6000	0x0C00_0000 - 0x0810_6000
	+/-	-	-
MMU Entry1	MMU_ENTRY1_STRADDR	0x0E00_0000	0x0E00_0000
	MMU_ENTRY1_ENDADDR	0x0E00_0000 + KM4 IMG2 size	0x0E00_0000 + KM4 IMG2 size
	MMU_ENTRY1_OFFSET	0x0E00_0000 - KM4 start physical address	0x0E00_0000 - KM4 start physical address
	+/-	-	-

Note: Considering KM4 IMG2 is appended to the tail of KM0 IMG2, the KM4 start physical address is determined by KM0 start physical address and KM0 IMG2 size.

13.3 OTA Upgrade from Local Server

The OTA from local server shows how device updates image from a local download server. The local download server sends image to device based on network socket as Fig13-4 shows.

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

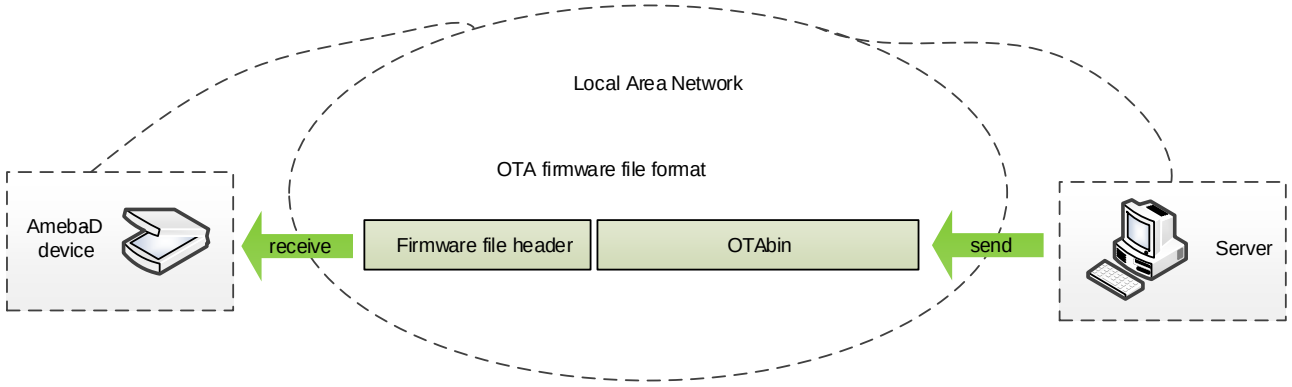


Fig13-4 OTA update diagram

13.3.1 Firmware Format

The firmware format is illustrated in Fig13-5.

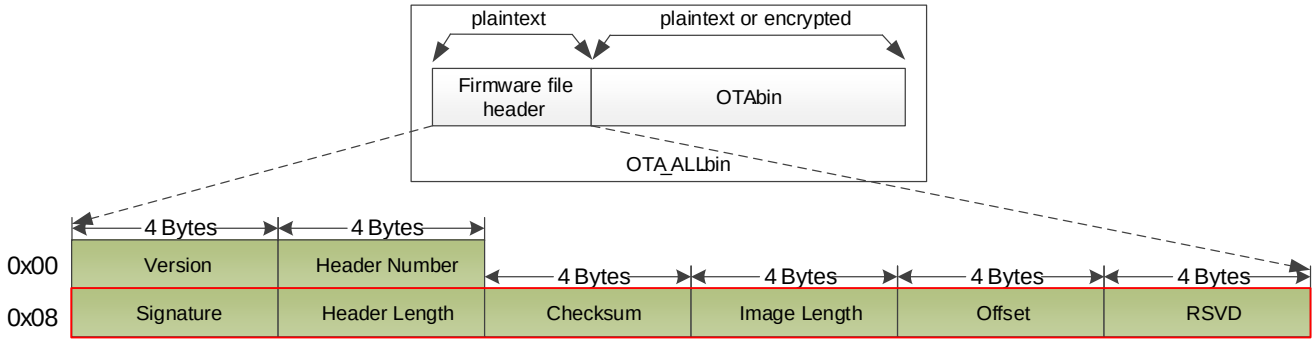


Fig13-5 Firmware format

Table13-2 Firmware header

Items	Address Offset	Size	Description
Version	0x00	4 bytes	The version of OTA Header, default 0xFFFFFFFF
Header Number	0x04	4 bytes	The number of OTA Header. For AmebaD, this value is 0x01
Signature	0x08	4 bytes	OTA Signature is string. For AmebaD, this value is 0x35 Σ ú K λ ă Ø l · f dz Ŝ λ ă á h ¢ ! é
Header Length	0x0C	4 bytes	The length of OTA header. For AmebaD, this value is 0x18
Checksum	0x10	4 bytes	The checksum of OTA image
Image Length	0x14	4 bytes	The size of OTA image
Offset	0x18	4 bytes	The start position of OTA image in current image
RSVD	0x1C	4 bytes	Reserved

13.3.2 OTA Flow

The OTA demo is provided in rtl8721d_ota.c The image upgrading is implemented in the following steps:

(1) Connect to local server with socket. The IP address and port are needed.
(2) Acquire the older firmware address to be upgraded according to MMU setting. If address is re-mapping to OTA1 space by MMU, the OTA2 address would be selected to upgrade. Otherwise OTA1 address would be selected.
(3) Receive firmware file header to get the target OTA image information, such as image length and destination address.
(4) Erase Flash space for new firmware

(5) Download new firmware from server and write it to flash
(6) Verify checksum. If checksum error, OTA fail.
(7) If checksum is ok, write signature to the upgraded firmware region and change another signature to all zero to indicate to boot from new firmware next time.
(8) OTA finish and reset the device. Then it would boot from new firmware.

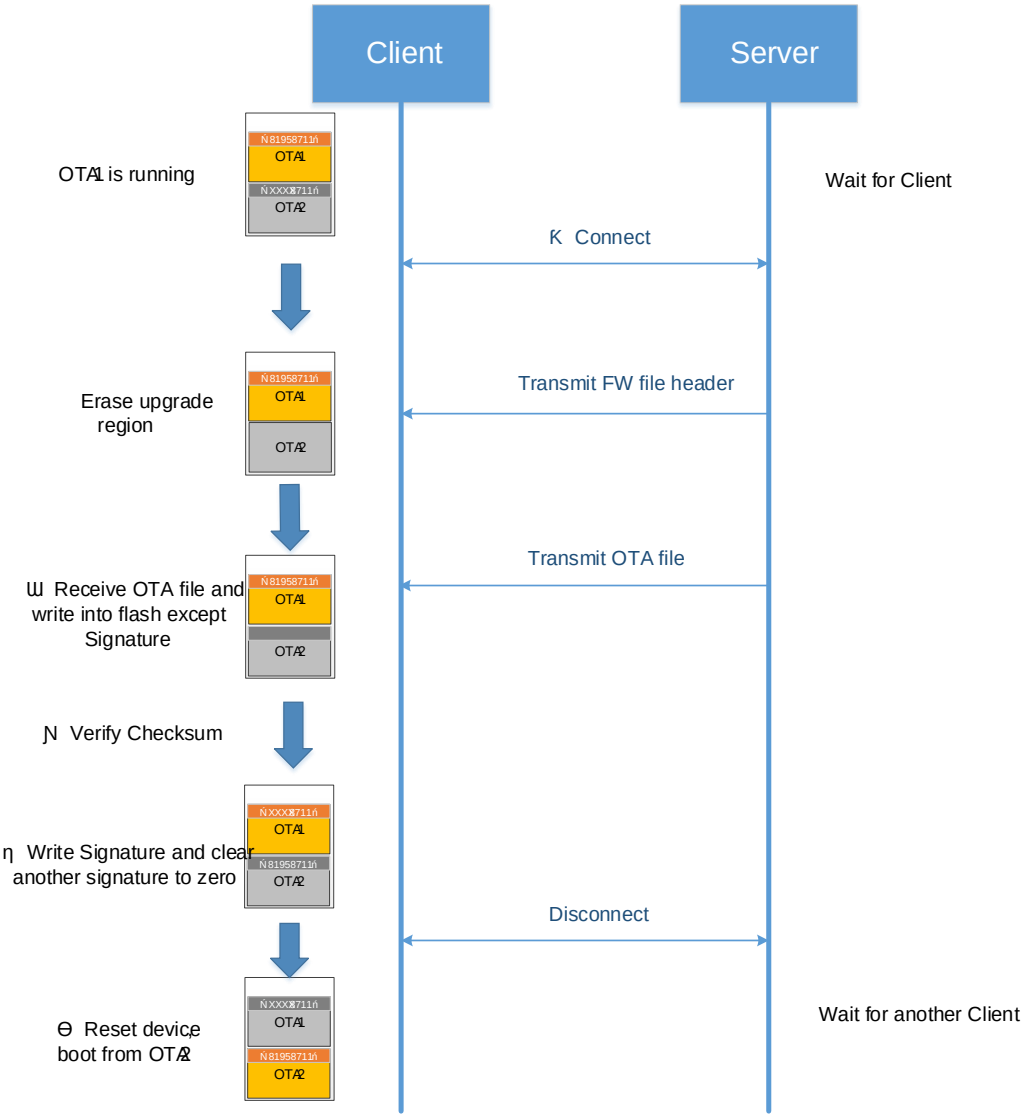


Fig13-6 OTA operation flow

13.3.3 Boot Option

When reboot after OTA finished, device would check firmware to determine to boot from OTA1 or OTA2. The following items must be checked for each firmware:

Signature. If it is 81958711, the signature is valid. Otherwise, the signature is invalid.
Hash/checksum if needed. Users can define FwCheckCallback in rtl8721d_bootcfg.

The boot flow is as follows:
(1) Check Signature and hash (if need) of OTA1 and OTA2 firmware.
(2) If both images are invalid, boot fail
(3) If only one image is valid, boot from this image

(4) If both images are valid, boot from OTA2

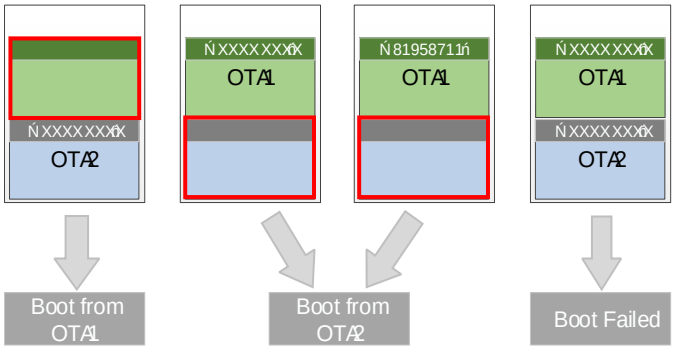


Fig13-7 OTA select diagram

13.3.4 Address Remapping

In bootloader, MMU entry0 and entry1 would be set as Table13-1 to remap memory. Here, the OTA space includes these images: KM0 IMG2, KM4 IMG2, KM4 IMG3 if need. These images all boot from OTA1 or OTA2.

As Fig13-3 illustrates, when boot from OTA, CPU may access different physical address

When boot from OTA1, MMU entry0 and entry1 would be set to remap virtual memory to OTA1 address space. That is, if CPU wants to read code or data from address 0x00_0000, it actually accesses physical address 0x0806_0000

When boot from OTA2, MMU entry0 and entry1 would be set to remap virtual memory to OTA2 address space. That is, if CPU wants to read code or data from address 0x00_0000, it actually accesses physical address 0x0810_6000.

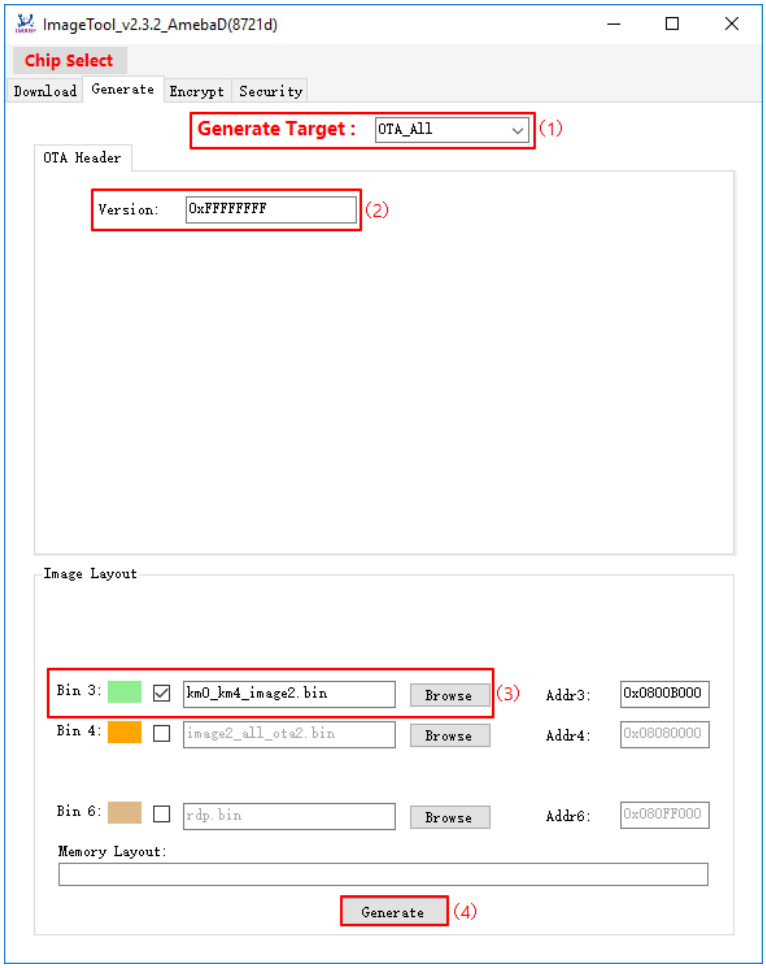
13.3.5 How to Use OTA Demo

These steps can be followed to run the OTA demo:

(1) Define CONFIG_OTA_UPDATE Macro to 1 in platform_opts.h to enable OTA function. Rebuild the project.

(2) Download images to device

(3) Generate OTA upgrade image file, named OTA_All.bin, with Image Tool and new firmware. This operation perpends firmware header information to new firmware, which is necessary for OTA.



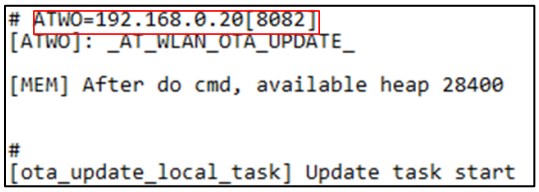
(4) Copy OTA_All.bin into the DownloadServer folder.
Tool path in SDK tools\DownloadServer

(5) Edit tools\DownloadServer\start.bat.
Port = 8082
File name=OTA_All.bin

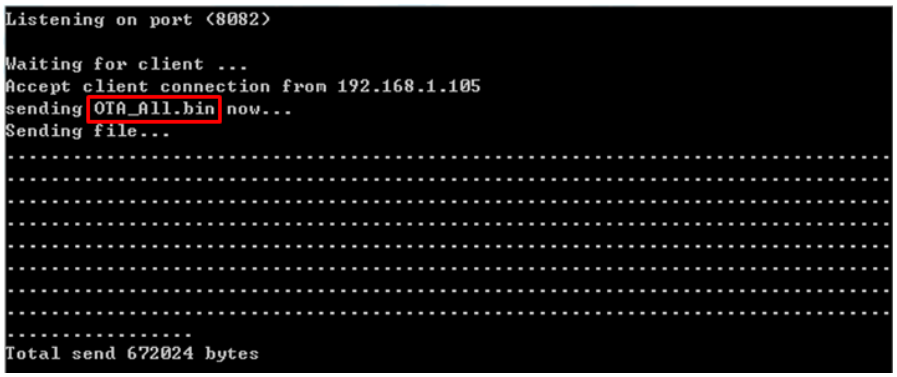
```
@echo off
DownloadServer 8082 OTA_All.bin
set /p DUMMY=Press Enter to Continue ...
```

(6) Click the start.bat, start to download server program.

(7) Reboot the DUT and connect the device to the AP which the OTA Server in.
(8) Enter command: ATWO=*IP*[*PORT*].
IP: IP of the OTA Server
Port: 8082, the same with start.bat



OTA upgrade procedure is started between DUT and server. Here is the local download server success message



(9) Reboot DUT to execute the new firmware after finishing image download

13.3.6 OTA Firmware Swap

Fig 13-8 shows the firmware swap procedure after OTA upgrade.

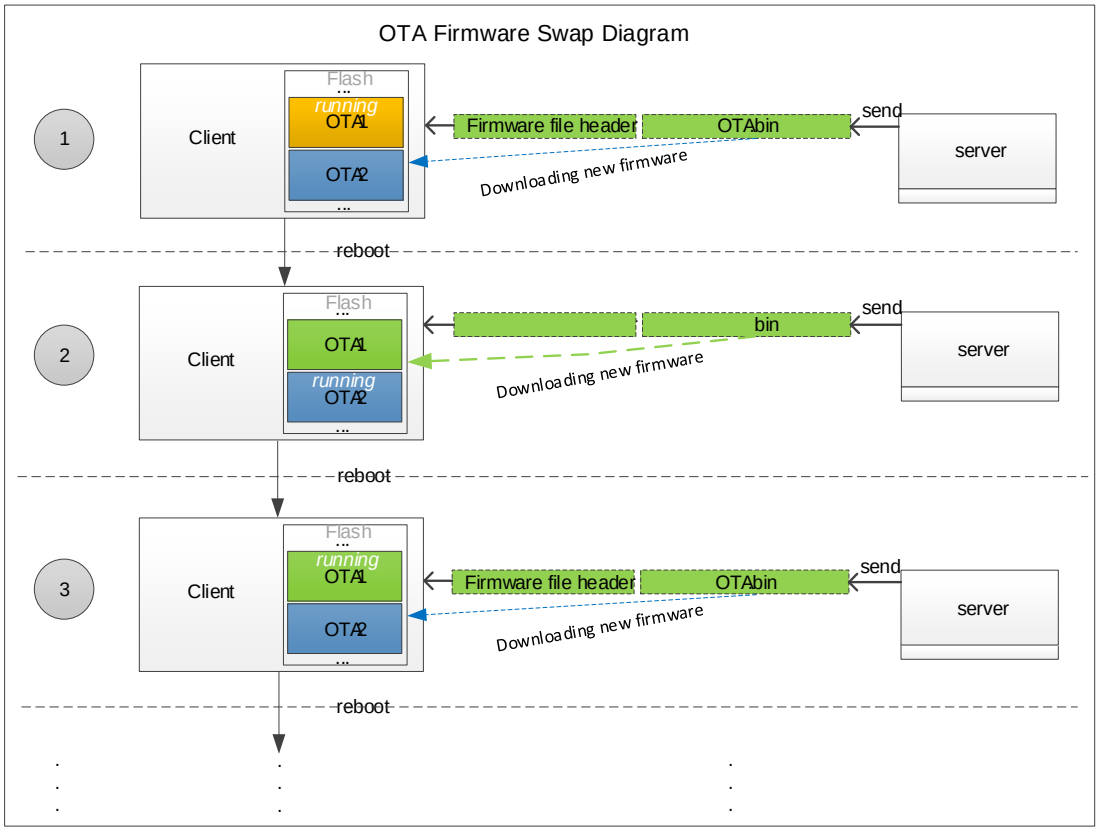


Fig13-8 OTA firmware swap procedure

13.4 User Configuration

Users can find the following configure items in rtl8721d_bootcfg.c

OTA start address

```
/*
* @brif OTA start address. Because KM0 & KM4 IMG2 are combined, users only need to set the start address
*       of KM0 IMG2.
*/
BOOT_RAM_DATA_SECTION
u32 OTA_Region[2] = {
        0x08006000,             /* OTA1 region start address */
        0x08106000,             /* OTA2 region start address */
};
```

User-defined MMU configure

```
/*
* @brif MMU Configuration.
*       There are 8 MMU entries totally. Entry 0 & Entry 1 are already used by OTA, Entry 2~7 can be used by Users.
*/
BOOT_RAM_DATA_SECTION
MMU_ConfDef Flash_MMU_Config[] = {
        /*VAddrStart,   VAddrEnd,                   PAddrStart,           PAddrEnd*/
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},    //Entry 2
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},    //Entry 3
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},    //Entry 4
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},    //Entry 5
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},    //Entry 6
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},    //Entry 7
        {0xFFFFFFFF,    0xFFFFFFFF,                 0xFFFFFFFF,           0xFFFFFFFF},
};
```

Firmware check handler configure

```
/**
* @brif  Firmware verify callback handler.
*       If users need to verify checksum/hash for image2, implement the function and assign the address
*       to this function pointer.
*/
BOOT_RAM_DATA_SECTION
FwCheckFunc FwCheckCallback = NULL;
```

14 eFuse

14.1 Introduction

eFuse belongs to One Time Programmable (OTP) technology, its default value is 0. And can only be changed from 0 to 1. eFuse can be used to hold the individual and stable data such as key, calibration data, MAC address, specific setting

The total size of physical eFuse is 512 bytes, and divided into two parts by software. Fig 14-1 shows, the first 288 bytes of physical eFuse are used for logical mapping, which can be mapped to logical eFuse by some algorithm and can be programmed multiple times. The left 224 bytes of physical eFuse are defined by Realtek.

Logical eFuse program will program header and package, and the package is in word, so it takes at least three bytes a time. Software reserves one byte for isolation, so when the space of eFuse for logical mapping is less than four bytes, it cannot be programmed anymore.

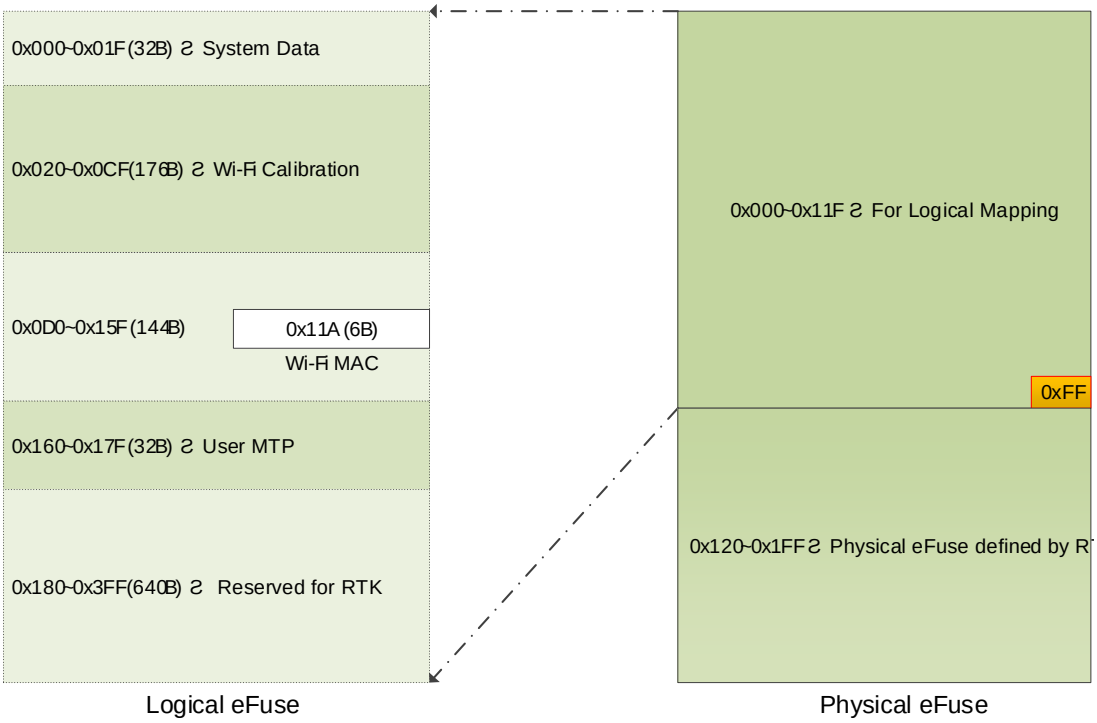


Fig14-1 eFuse diagram

14.2 Power Requirement

The power requirement of eFuse operation is listed in Table 14-1.

Table14-1 Operation under different voltage

Supply Voltage (V)	Operation
1.8	Only read operation is allowed
3.3	Read & Write

Note: If you want to program eFuse, you must switch the power supply to 3.3V.

14.3 eFuse Auto-load

eFuse can auto-load part of setting to control the circuit. eFuse auto-load is changed in word (two bytes). Under default condition, value of all the System Data bytes is 0xFF, and the System Configure Registers have its default values.

The way to change the value of System Configure Registers is as follows:

Make sure that the first two bytes of System Data area are written to 0x21, 0x87 correctly
Program the corresponding bytes in word. If just programming in byte, another byte will load the eFuse default value 0xFF, which may make mistake.
Reboot the chip, and the System Configure Registers will load the new values

14.4 Physical eFuse

For detailed information about physical eFuse of AmebaD, refer to AN0411.

14.5 Logical eFuse

14.5.1 Logical eFuse Layout

There are 1024 bytes logical eFuse in AmebaD, as Table 14-2 shows. Most of the logical eFuse space are reserved for extension, so the 1024 bytes of logical eFuse can be easily mapped to 287 bytes of physical eFuse.

Table 14-2 Logical eFuse Layout

Start Address	End Address	Description
0x000	0x01F	System Data to be auto-loaded
0x020	0x15F	Wi-Fi calibration data
0x160	0x17F	User MTP
0x180	0x183	Cap-Touch
0x184	0x18F	Reserved
0x190	0x1A4	BT parameters
0x1A5	0x1BF	Reserved
0x1C0	0x1CF	HCI USB
0x1D0	0x2FF	Reserved

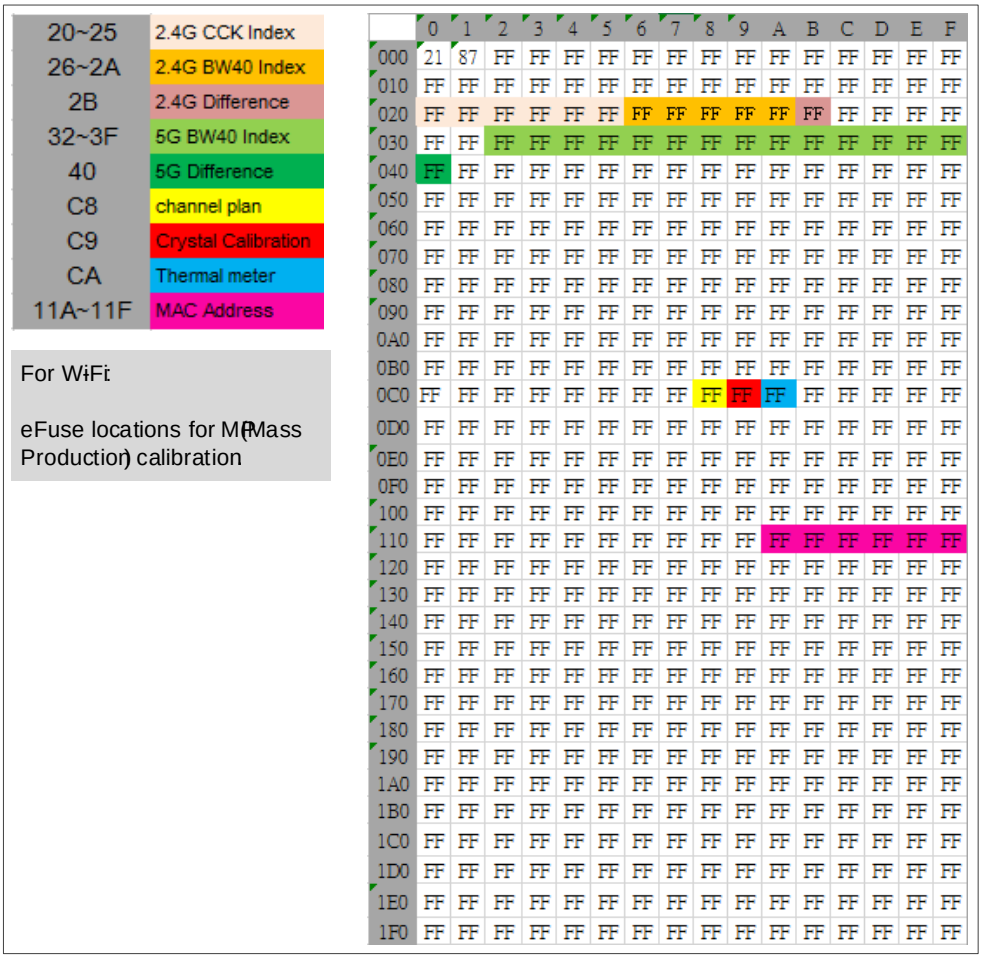


Fig14-2 Wi-Fi eFuse data

14.5.2 SPIC Address-4Byte Enable

Address Offset	Name	Bit	Default	Description
0E	SPIC address-4byte enable	[6]	0	SPIC address-4byte enable 1: Enable 0: Disable

If you want to access the address length of >32 for the memory area of higher density (larger than 128MB), you should enable 4-byte mode. In short, you should read the eFuse 0x0E bit[6] first and then program it to 1.

14.5.3 Wi-Fi 2.4G Power Index

The address and specification of 2.4G power index is shown in Fig14-3 and Fig14-4.

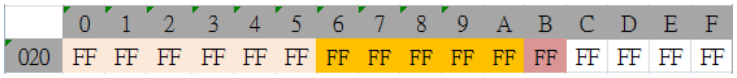


Fig14-3 2.4G Wi-Fi power index address

offset	name	comment
20~25	2.4G CCK Index	Path A CCK Power Index for Ch 1,2, Range 0~127,0.25dbm/step.
		Path A CCK Power Index for Ch 3, 4, 5, Range 0~127,0.25dbm/step.
		Path A CCK Power Index for Ch 6, 7 ,8, Range 0~127,0.25dbm/step.
		Path A CCK Power Index for Ch 9, 10, 11, Range 0~127,0.25dbm/step.
		Path A CCK Power Index for Ch 12, 13, Range 0~127,0.25dbm/step.
		Path A CCK Power Index for Ch 14, Range 0~127,0.25dbm/step.
26~2A	2.4G BW40 Index	Path A 2G BW40-1S Power Index for Ch 1, 2, Range 0~127,0.25dbm/step.
		Path A 2G BW40-1S Power Index for Ch 3, 4, 5, Range 0~127,0.25dbm/step.
		Path A 2G BW40-1S Power Index for Ch 6, 7 ,8, Range 0~127,0.25dbm/step.
		Path A 2G BW40-1S Power Index for Ch 9, 10, 11, Range 0~127,0.25dbm/step.
		Path A 2G BW40-1S Power Index for Ch 12, 13, 14 Range 0~127,0.25dbm/step.
2B	2.4G Difference	Power Index Difference between BW20-1S and BW40-1S.
		Bit[7:4]: Path A 2G Offset, Range –8~7,0.5dbm/step.
		Power Index Difference between OFDM-1Tx and BW40-1S.
		Bit[3:0]: Path A 2G Offset, Range –8~7,0.5dbm/step.

Fig14-4 2.4GWi-Fipower indexspecification

14.5.4 Wi-Fi5GPower Index

The addressandspecificationof 5G power index is shown inFig14-5 andFig14-6.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

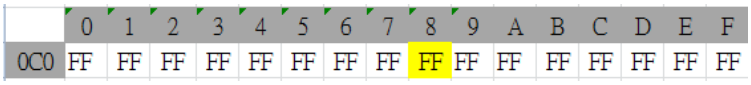
Fig14-5 5GWi-Fipower indexaddress

32~3F	5G BW40 Index	Path A 5G BW40-1S Power Index for Ch 36,38,40, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 44,46,48, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 52,54,56, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 60,62,64, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 100,102,104, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 108,110,112, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 116,118,120, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 124,126,128, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 132,134,136, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 140,142,144, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 149,151,153, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 157,159,161, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 165,167,169, Range 0~127,0.25dbm/step.
		Path A 5G BW40-1S Power Index for Ch 173,175,177, Range 0~127,0.25dbm/step.
40	5G Difference	Power Index Difference between BW20-1S and BW40-1S.
		Bit[7:4]: Path A 5G Offset, Range –8~7,0.5dbm/step.
		Power Index Difference between OFDM-1Tx and BW40-1S.
		Bit[3:0]: Path A 5G Offset, Range –8~7,0.5dbm/step.

Fig14-6 5GWi-Fipower indexspecification

14.5.5 Wi-Fi Channel Plan

The address of Wi-Fi channel plan is shown in Fig 14-7.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Fig 14-7 Channel plan address

eFuse	Name	Bit	Default	Comment
C8	Channel Plan	[7]	0h	Software configure mode 0h: Enable software configure (refer to Channel Plane Domain Code) 1h: Disable software configure (cannot change Channel Plan Setting)
		[6:0]	7Fh	Channel Plan 0x20: 2G Worldwide 13, Active scan Ch1 ~ Ch11, Passive scan Ch12, Ch13 0x7F: 2G Worldwide 13, Active scan Ch1 ~ Ch11, Passive scan Ch12, Ch13 5G Worldwide, all bands are Passive scan

Frequently used country codes and the corresponding channel plans are listed in Table 14-3, Table 14-4 and Table 14-5.

Table 14-3 Relationship between country code and channel plan

Country	Abbreviation	Enumeration Variable Name	Channel Plan
America	US	RTW_COUNTRY_US	0x76
Australia	AS	RTW_COUNTRY_AS	0x2A
Chile	CL	RTW_COUNTRY_CL	0x2D
China	CN	RTW_COUNTRY_CN	0x48
Europe	EU	RTW_COUNTRY_UA	0x26
Japan	JP	RTW_COUNTRY_JP	0x27
Mexico	MX	RTW_COUNTRY_MX	0x4D
Ukraine	UA	RTW_COUNTRY_UA	0x35
Default	-	RT_CHANNEL_DOMAIN_REALTEK_DEFINE	0x7F

Table 14-4 2.4G channel map of the above channel plan

Channel Plan	2.4G Channel Definition	Channels	Passive Scan
0x76	2G_02	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	-
0x2A	2G_02	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	-
0x2D	2G_01	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	12,13
0x48	2G_01	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	12,13
0x26	2G_01	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	12,13
0x27	2G_04	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	-
0x4D	2G_02	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	-
0x35	2G_01	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	12,13
0x7F	2G_01	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13	12,13

Note: Active scan channels are channels listed in table except the passive scan channels.

Table 14-5 5G channel map of the above channel plan

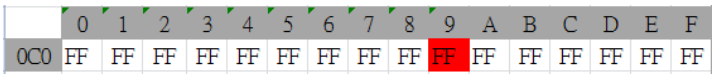
Channel Plan	5G Channel Definition	Channels	Passive Scan	DFS Channels
0x76	5G_22	36, 40, 44, 48 52, 56,60, 64 100, 104, 108, 112, 116, 120, 12 128, 132, 136, 140, 144 149, 153, 157, 161, 165	52, 56, 60, 64 100, 104, 108, 112, 116, 120 124, 128, 132, 136, 140, 144	52, 56, 60, 64 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144
0x2A	5G_00	-	-	-
0x2D	5G_22	36, 40, 44, 48	52, 56, 60, 64	52, 56, 60, 64

		52, 56, 60, 64 100, 104, 108, 112, 116, 120, 12 128, 132, 136, 140, 144 149, 153, 157, 161, 165	100, 104, 108, 112, 116, 120 124, 128, 132, 136, 140, 144	100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144
0x48	5G_07	36, 40, 44, 48 52, 56, 60, 64 149, 153, 157, 161, 165	52, 56, 60, 64	52, 56, 60, 64
0x26	5G_02	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 120, 12 128, 132, 136, 140	52, 56, 60, 64 100, 104, 108, 112, 116, 120 124, 128, 132, 136, 140	52, 56, 60, 64 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140
0x27	5G_02	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 120, 12 128, 132, 136, 140	52, 56, 60, 64 100, 104, 108, 112, 116, 120 124, 128, 132, 136, 104	52, 56, 60, 64 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140
0x4D	5G_01	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 132, 13 140 149, 153, 157, 161, 165	52, 56, 60, 64 100, 104, 108, 112, 116, 132 136, 140	52, 56, 60, 64 100, 104,108, 112, 116, 132, 136, 140
0x35	5G_03	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 120, 12 128, 132, 136, 140 149, 153, 157, 161, 165	52, 56, 60, 64 100, 104, 108, 112, 116, 120 124, 128, 132, 136, 140	52, 56, 60, 64 100, 104, 108, 112,116, 120, 124, 128, 132, 136, 140
0x7F	5G_36	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 120, 12 128, 132, 136, 140, 144 149, 153, 157, 161, 165	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 120 124, 128, 132, 136, 140, 144 149,153, 157, 161, 165	52, 56, 60, 64 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144

Note: Active scan channels are channels listed in table except the passive scan channels.

14.5.6 Wi-Fi Crystal Calibration

The address of Wi-Fi crystal calibration is shown in Fig14-8.



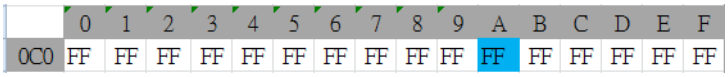
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Fig14-8 Crystal Calibration address

eFuse	Name	Bit	Default	Comment
C9	Crystal Calibration	[7]	0h	Reserved
		[6:0]	40h	XTAL_K Value Xi=Xo, range 0~7Fh FFh = 00h

14.5.7 Wi-Fi Thermal Meter

The address of Wi-Fi thermal meter is shown in Fig14-9.



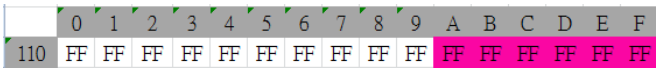
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Fig14-9 Thermal Meter address

eFuse	Name	Bit	Default	Comment
CA	Thermal Meter	[7:0]	16h	Thermal Meter Value System maker will calibrate a value and save it in EEPROM. 0xFF: Disable Tx power tracking function

14.5.8 Wi-Fi MAC Address

The Wi-Fi MAC address is shown in Fig14-10.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
110	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Fig14-10 MAC Address

Address Offset	Bit	Default	Comment
11A-11F	[47:0]	00E04C872101	After the auto-load command or hardware reset, AmebaD loads MAC addresses to MACID of the I/O registers.

14.5.9 Cap-Touch

Address Offset	Bit	Default	Description
180h	[7:0]	0xFF	Touch key channel 0 threshold[7:0]
181h	[6:0]	0xFF	Touch key channel 1 threshold diff[6:0]
	[7]	0xFF	Touch key channel 0 threshold[8]
182h	[6:0]	0xFF	Touch key channel 2 threshold diff[6:0]
	[7]	0xFF	Touch key channel 0 threshold[9]
183h	[6:0]	0xFF	Touch key channel 3 threshold diff[6:0]
	[7]	0xFF	Touch key channel 0 threshold[10]

14.5.10 BLE

Address Offset	Bit	Default	Description
190~195h	[47:0]	0xFFFFFFFFFFFF	BT address
196h	[7:0]	0x08	Bit[0]: Fixed to 0 Bit[1]: Tx gain K valid bit Bit[2]: Flatness K valid bit Bit[3]: Fixed to 1 Bit[7:4]: Fixed to 0 Note: After Tx gain K flow, remember to enable Tx gain K valid bit, then 0x196 will be 0x0A.
197h	[7:0]	0xFF	Tx gain K Note: Users need to write Tx gain K value if Tx gain K valid bit is enabled.
198~199h	[15:0]	0xFFFF	Flatness K
19A~19Bh	[15:0]	0xFFFF	Reserved for Realtek
19Ch	[7:0]	0x23	Max. Tx gain LE1M iqm_max_txgain_LE1M
19Dh	[7:0]	0x23	Max. Tx gain LE2M iqm_max_txgain_LE2M
19E~19Fh	[15:0]	0xFFFF	Reserved for Realtek

1A0h	[7:0]	0xFF	BT thermal meter tmeterx4_txgaink_module
1A1h	[7:0]	0xFF	Logic map IQK/KOK valid bit 0xFF: Disable logic map IQK/KOK values 0xFE: Enable logic map IQK/KOK values
1A6h ~ 1A9h	[31:0]	0xFFFFFFFF	Logic map IQK values
1AAh ~ 1ABh	[15:0]	0xFFFF	Logic map KOK values

14.5.11 HCI USB

Address Offset	Bit	Name
1C0h	[7:0]	VID[7:0]
1C1h	[7:0]	VID[15:8]
1C2h	[7:0]	PID[7:0]
1C3h	[7:0]	PID[15:8]
1C4h	[7:0]	USB device type
1C5~1CFh		RSVD

14.6 eFuse PG APIs

Items	API	Comment
Low Level API	EFUSE_PMAP_WRITE8	Physical map write one byte
	EFUSE_PMAP_READ8	Physical map read one byte
	EFUSE_LMAP_WRITE	Write eFuse logical address by length
	EFUSE_LMAP_READ	Read total eFuse logical map
mbed API	efuse_otp_write	Write content to OTP eFuse by length
	efuse_otp_read	Read eFuse OTP content by length
	efuse_mtp_write	Write user's content to USER MTP Space (0x160~0x17F)
	efuse_mtp_read	Read eFuse content from address (0x160 ~ 0x17F)

14.6.1 Low Level APIs

14.6.1.1 EFUSE_PMAP_WRITE8

Items	Description
Introduction	Physical map write one byte
Parameters	CtrlSetting: eFuse control setting read from REG_LP_EFUSE_CTRL (Trigger 0) Addr: eFuse physical address Data: 1 byte data to write L25OutVoltage: L25EOUTVOLTAGE.
Return	Status value: _TRUE: Write is ok. _FALSE: Write is failed.

14.6.1.2 EFUSE_PMAP_READ8

Items	Description
Introduction	Physical map read one byte
Parameters	CtrlSetting: eFuse control setting read from REG_LP_EFUSE_CTRL (Trigger 0) Addr: eFuse physical address Data: 1 byte data buffer for eFuse data read

	L25OutVoltage: L25EOUTVOLTAGE.
Return	Status value: _TRUE: Read is ok. _FALSE: Read is failed.

14.6.1.3 EFUSE_LMAP_WRITE

Items	Description
Introduction	Write eFuse logical address by length
Parameters	addr: logical address, should be word aligned cnts: byte number, should be even data: data buffer to be write
Return	Status value: _SUCCESS: Write is ok. _FAIL: Write is failed.

14.6.1.4 EFUSE_LMAP_READ

Items	Description
Introduction	Read total eFuse logical map
Parameters	pbuf: 1024 bytes length buffer used for eFuse logical map
Return	Status value: _ SUCCESS: Read is ok. _FALSE: Read is failed.

14.6.2 Mbed APIs

14.6.2.1 efuse_otp_write

Items	Description
Introduction	Write content to OTP eFuse by length (mbed API)
Parameters	address: Specifies the offset of the programmed OTP. len: Specifies the data length of programmed data. buf: Specified the data to be programmed.
Return	Status value: 0: Success -1: Failure

14.6.2.2 efuse_otp_read

Items	Description
Introduction	Read eFuse OTP content by length (mbed API)
Parameters	address: Specifies the offset of the OTP. len: Specifies the length of readback data. buf: Specified the address to save the readback data.
Return	Status value: 0: Success -1: Failure

14.6.2.3 efuse_mtp_write

Items	Description

Introduction	Write user's content to USER MTP Space (0x160~0x17F)
Parameters	data: Specified the data to write len: Specified the data length to write
Return	Status value: 0 ~ 32: Success 0 or -1: Failure

14.6.2.4 efuse_mtp_read

Items	Description
Introduction	Read eFuse content from address (0x160 ~ 0x17F)
Parameters	Data: Specified the address to save the read back data
Return	N/A

14.7 eFuse PG Command

Items	Offset	Command
Tx Power Index	0x20~0x2B	iwpriv config_set wmap, 020, 202020202020202020202002
Channel plan, XTAL & Thermal & RTK reserved	0xC8~0xCF	iwpriv config_set wmap, 0C8, 20201A0500000[1]0FF
MAC Address	0x11A~0x11F	iwpriv config_set wmap, 11a, 00e04c870102
Realtek RSVD	0x130~0x139	iwpriv config_set wmap, 130, FF01001000FF00FF1000
Get all eFuse map		iwpriv config_get realmap

1. Channel plan does not affect the results of RF verification, so you can choose whether to write the correct channel plan here.

15 Power Save

15.1 Power Save Mode

15.1.1 Summary

AmebaD supports two low power modes which are deepsleep mode and sleep mode. Deepsleep mode turns off more power domain than sleep mode so it has lower power consumption. Tickless is a FreeRTOS low power feature, which just halt CPU (no clock or power) for a while when it has nothing to do. Table 15-1 explains power save related terms.

Table 15-1 Power save mode

Name	Domain	Description
Tickless	Software	FreeRTOS low power feature
Sleep mode	Chip-level	A power save mode on chip level
Deepsleep mode	Chip-level	A more power save mode on chip level

NOTE

configUSE_TICKLESS_IDLE must be enabled in sleep mode because sleep mode is based on tickless.

There are various of sources, and all wake-up source is to wake KM0. Then KM0 wakes KM4 as appropriate. Refer to Fig 15-1 of document UM0400 which show power of peripherals is on or off when they are in sleep mode or deepsleep mode by different colors.

Fig 15-1 shows all the wake-up sources. HS and AON are special because they are mater switch that manages some wake-up sources. HS UART is peripheral belongs to KM4. CapTouch, RTC, etc. belong to AON domain.

Peripherals in AON domain can wake system both in sleep mode and deepsleep mode.

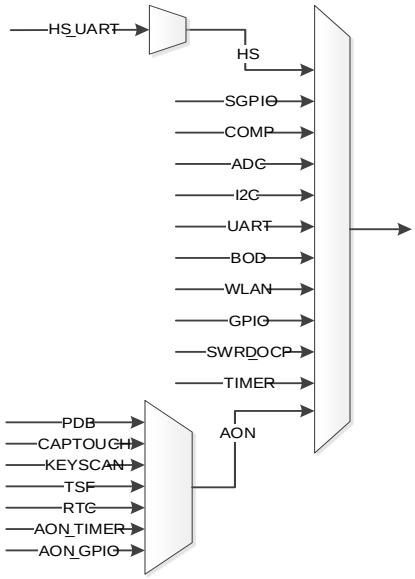


Fig 15-1 Wake sources

15.1.2 FreeRTOS Tickless

FreeRTOS supports a low power feature called tickless. It is implemented in an idle task which has the lowest priority. It is invoked when there is no other task under running.

Fig15-2 shows idle task code flow. In idle task, it will check wakelock to determine if CPU need to enter sleep mode or software tickless. L Ŧ y ˜ 2 ű Σ / t ¦ ø ƛ f f Ŝ È Ŝ Ŏ dz ű Ŝ l · ypt) which makes system enter KM0 sleep or suspend CPU clock is stopped) until Ŧ 2 NJ ƛ y interrupt happens. Normally systick interrupt resume it. This is software tickless.
If yes, it will execute function freertos_pre_sleep_processing() to enter sleep mode.

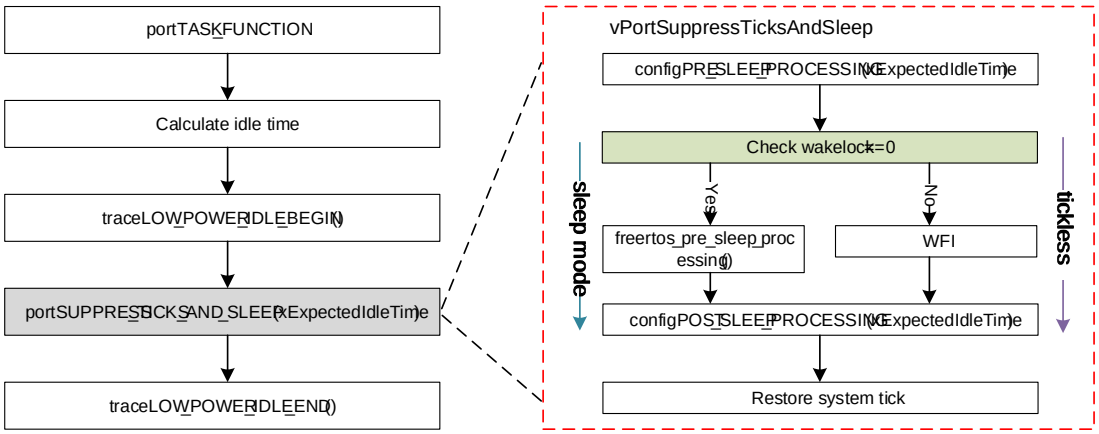


Fig15-2 FreeRTOS tickless in an idle task

15.1.3 SleepMode

15.1.3.1 Sleep Mode Flow

Dual-core design of KM0 and KM4 is largely for power save. KM4 is used for the application while KM0 is used for power save, Wi-Fi firmware, and power/clock control. Sleep mode is realized with the tickless and wakelock mechanism.

There are two sleep modes: clock-gating (CG) and power-gating (PG). CG means to turn off particular clock and PG means to turn off some power domains. So PG has lower power consumption. Recommended sleep mode for dual cores is KM0 CG + KM4 PG/CG. Fig15-3 shows sleep and wake flow of KM0 CG+ KM4 PG.

When KM4 is in an idle task, it will judge whether it is necessary to enter sleep mode. If yes, KM4 executes the suspend flow to back up the registers of CPU, NVIC, MPU, etc. KM4 sends IPC message to KM0 and then executes WFE (waiting for event) to enter halt. Then KM0 receives IPC message and closes KM4 power in IPC interrupt. KM0 will judge whether KM0 needs to enter CG in the idle task. CG itself.

If wakeup event occurs, KM0 will continue to execute code from where it halts. It will turn on the power of KM4 and exit CG if need to wake KM4. After KM4 is powered on, it will do the boot on process, and check whether it wakes up from PG. If yes, it will back up register value into the CPU, NVIC, MPU, etc. KM4 will jump to the code where it backs up CPU register.

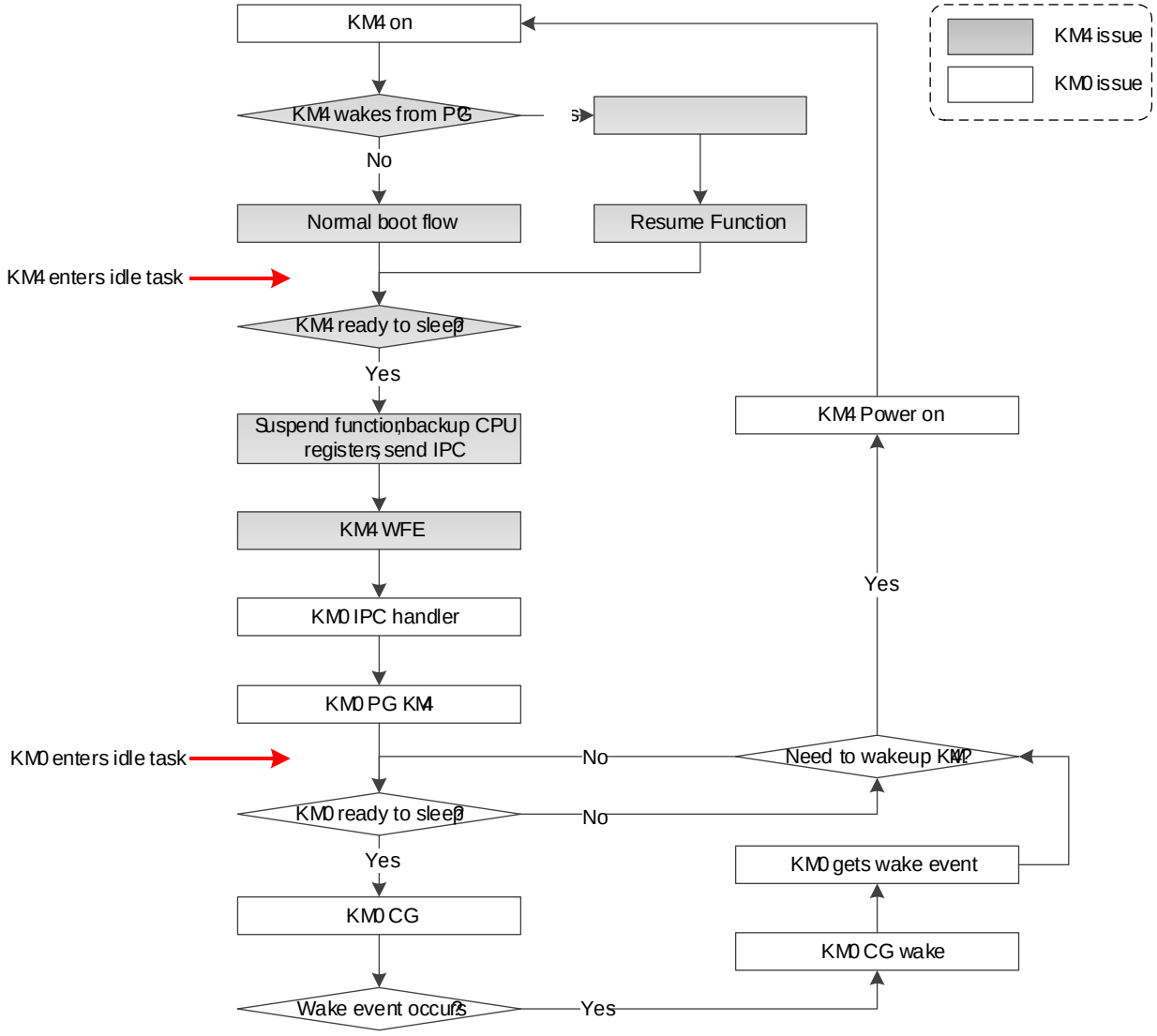


Fig15-3 KM0 CG + KM4 PG

KM0 CG + KM4 CG is relatively simple. KM4 sends IPC and executes ó ø Ŧ to Ŝ halt itself. KM0 will close KM4 clock ỷ L t / Ƙ l · y ˇ Ř f Ŝ NJ Φ Y a is the same with KM0 CG + KM4 PG. After KM0 wakes up, if KM4 needs to wake up, it will enable KM4's clock and execute ś ε v ε λ y ˇ ă ű NJ dz Ŏ ú λ ž wake KM4. KM4 will execute code from where it halts.

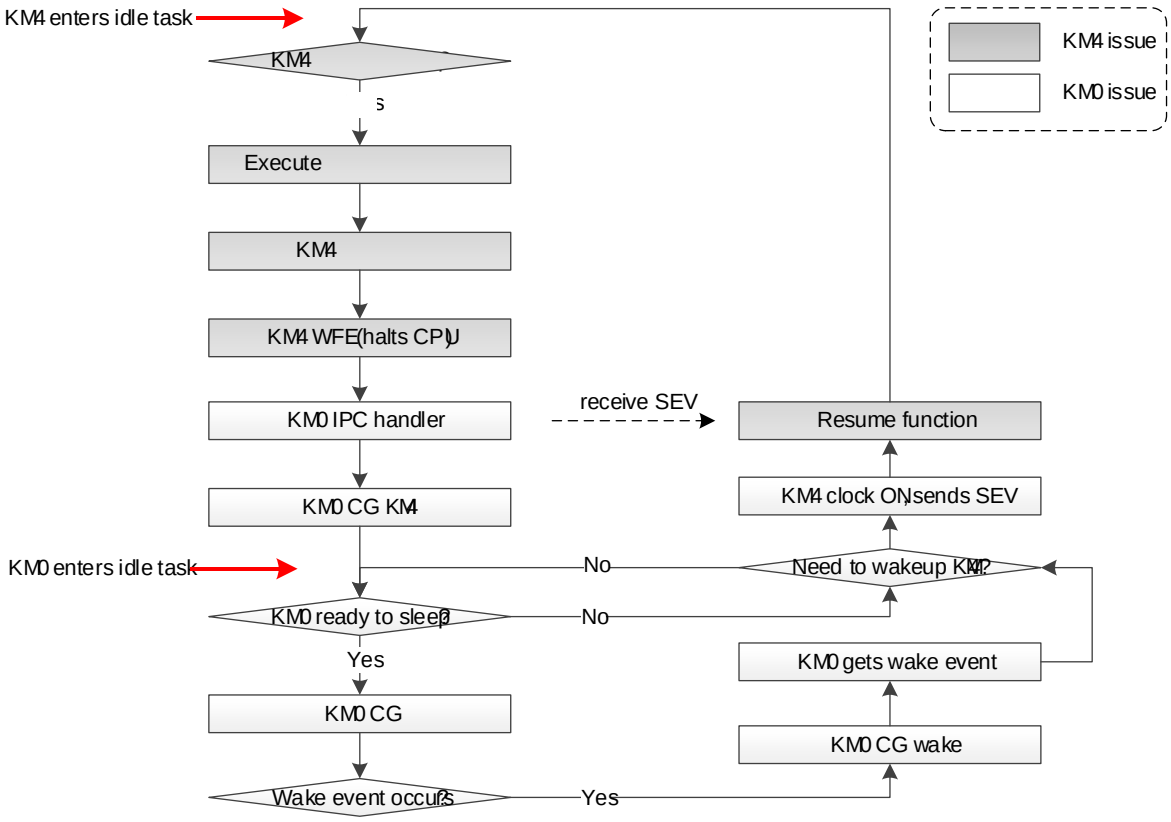


Fig15-4 KM0 CG+KM4 CG

It can be seen from the flow that Y a n ƨ dz ă ú Ŝ È Ŝ Ŏ dz ú Ŝ ă ά ø Ŧ Ŝ ε ű ƨ Ķ l ŧ ú ƛ ű ă Ŝ ŧ Ŧ ƛ y ō ƨ ű K / D Application should be designed like that idle task has the opportunity to be executed and tickless should be enabled, or the sleep flow will not be executed.

NOTE

It is not recommended to develop application on KM0 platform because of limited resources

Fig15-5 shows the code flow of PG and CG in function freertos_pre_sleep_processing() on KM4 platform. Arrow in the left of function means to enter function while it means exit function if in the right

Refer to Fig15-2 to achieve how an idle task executes to this function.

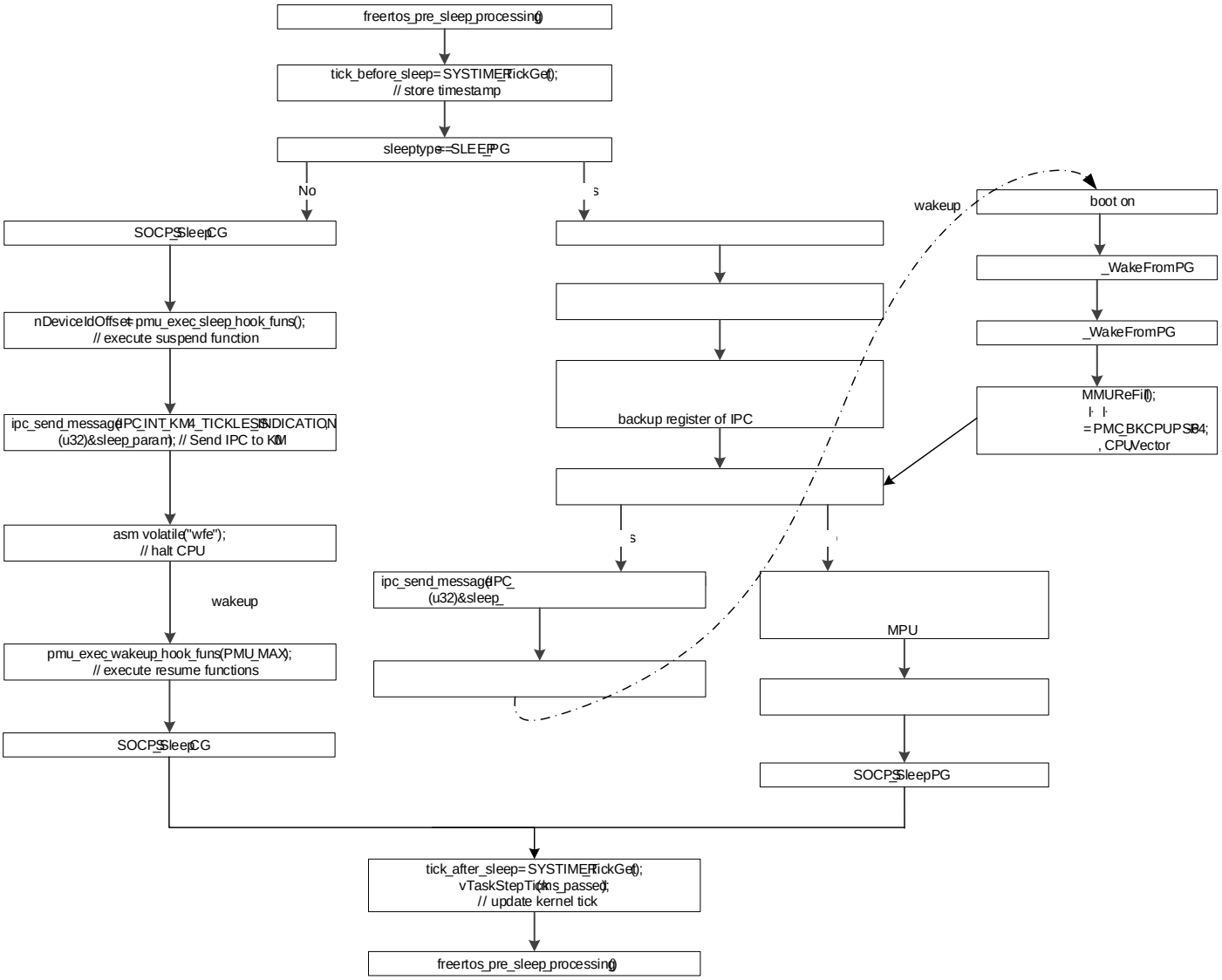


Fig15-5 Code flow of KM4 PG and CG

Fig15-6 shows the code flow of sleep mode related IPC handler on KM0 platform.

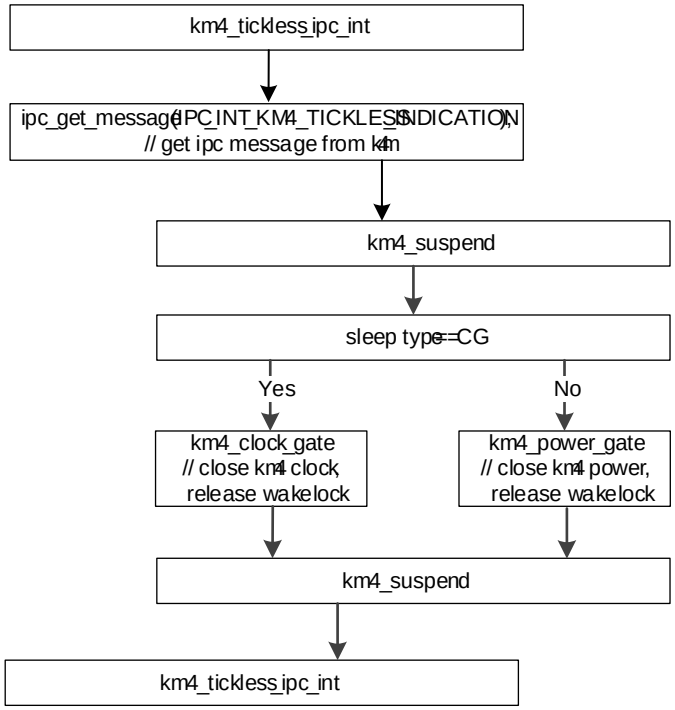


Fig15-6 Code flow of IPC handler in sleep mode on KM0

Fig15-7 shows the code flow CG in function freertos_pre_sleep_processing() in KM0 platform. Refer to Fig15-2 to achieve how idle task executes to this function.

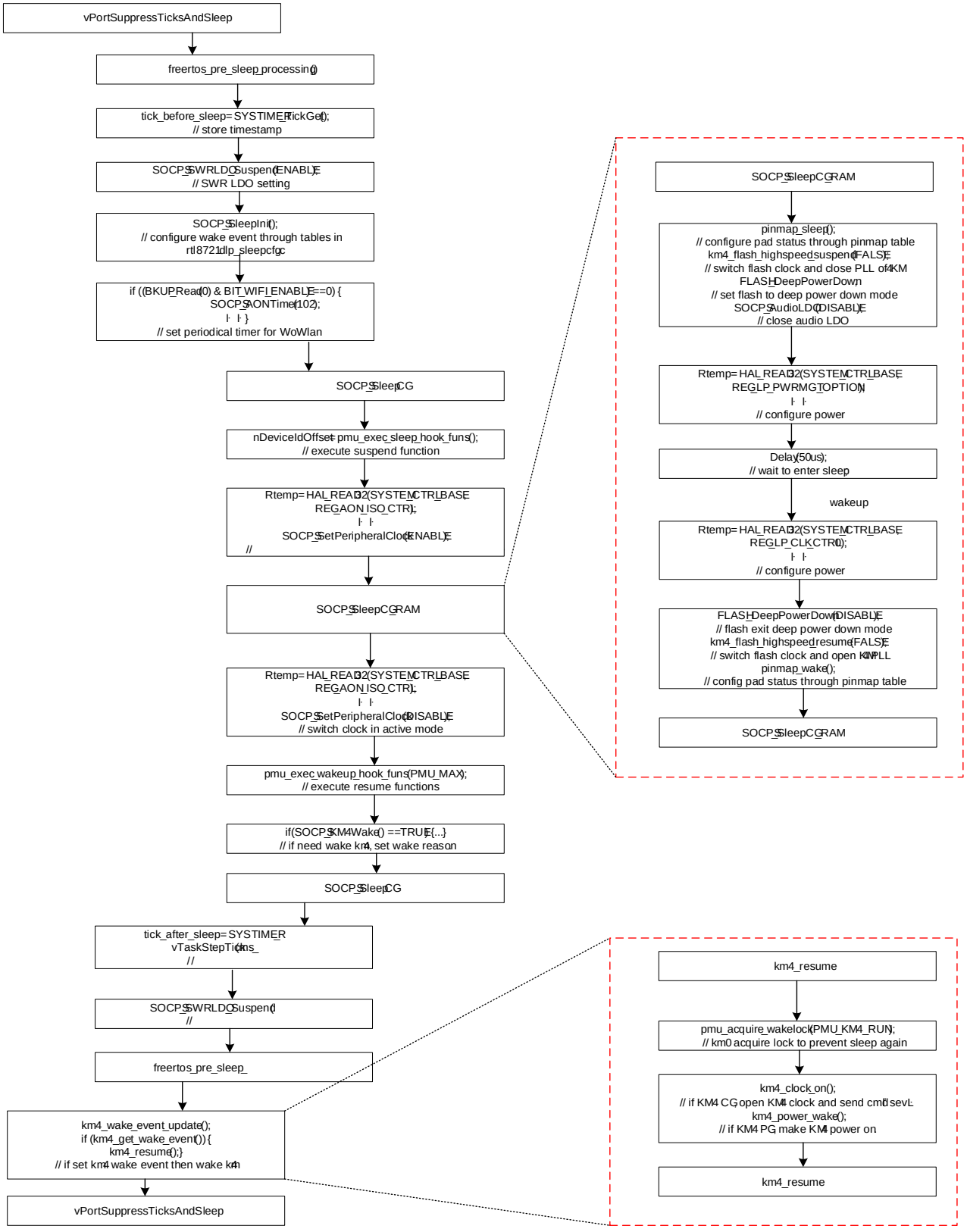


Fig15-7 Code flow of KM0 CG

15.1.3.2 Wakelock

In some situations, system needs to keep awake to receive certain events. Otherwise, event may be missed when the system is under sleep. An idea of wakelock is introduced that system cannot sleep if some module is holding wakelock.

A wakelock bit map is used to store the wakelock status. Each module has its own bit in wakelock bit map (see enum. PMU_DEVICE). Users can also add wakelock in the enum. PMU_DEVICE. If the wakelock bit map equals zero, it means that there is no module holding wakelock. If the wakelock bit map is larger than zero, it means that there is some module holding wakelock.

Wakelock is a judging condition in function freertos_ready_to_sleep(). When system boot, KM4 holds wakelock PMU_OS and KM0 holds PMU_KM4_RUN and PMU_OS wakelock. Only if wakelocks are released, KM0 or KM4 are permitted to enter sleep mode. Fig15-8 shows function freertos_ready_to_sleep() and it will judge the value of wakelock.

```
int freertos_ready_to_sleep(void)
{
    u32 current_tick = xTaskGetTickCount();

    /* timeout */
    if (current_tick < sleepwakelock_timeout) {
        return FALSE;
    }

    if (wakelock == 0) {                //return TRUE only if all wakelock release
#if defined (ARM_CORE_CM0)
        /* timeout */
        if (current_tick >= km4_sleep_timeout) {
            return FALSE;
        } else {
            if (km4_sleep_timeout != 0xffffffff) {
                SOCPS_AONTimer(km4_sleep_timeout - current_tick);
                SOCPS_SetWakeEventAON(BIT_AON_WAKE_TIM0_MSK, ENABLE);
                SOCPS_AONTimerCmd(ENABLE);
            }
        }
#endif
        return TRUE;
    }
    else
        return FALSE;
} ? end freertos_ready_to_sleep ?
```

Fig15-8 Function freertos_ready_to_sleep()

It is recommended to enter sleep mode by release wakelock. After PMU_OS wakelock of KM4 release, KM4 will enter sleep mode in an idle task and send IPC to KM0. KM0 will power gate or clock gate KM4 when release PMU_OS and PMU_KM4_RUN. The judgment of wakelock in function freertos_ready_to_sleep() will be the value TRUE, so KM0 can clock gate itself in its idle task.

When the system wakeup, it will enter sleep mode again quickly unless it acquire wakelock.

The definition of related functions is shown in section 15.2.

15.1.3.3 Suspend and Resume Function

It is a good way to use suspend and resume function If there is something to do before chip sleep or after chip wake up. The suspend and resume functions are executed respectively before CPU enters sleep mode and after CPU wakes up in an idle task.

A typical application of resume function is to acquire the wakelock to prevent chip sleep again. Also, If KM4 chooses PG, some peripheral will lose power so they need to re-initialize It can be implemented in the resume function.

A typical application of suspend function is WoWLAN. Refer to section Wi-Fi Power Save to get more information.

The definition of related functions is shown in section 15.2.

15.1.3.4 Wi-Fi Power Save

In IEEE 802.11 power save management, it allows the station to enter its own sleep state. It defines that station needs to keep awake in a certain timestamp and enters sleep state otherwise.

WLAN driver acquires wakelock to avoid the system enter sleep mode when WLAN needs to keep awake. And it releases wakelock when it is permitted to enter sleep state.

IEEE 802.11 power management allows station enter power save mode. Station cannot receive any frame during power saving. Thus AP needs to buffer these frames and requires station periodically wakeup to check beacon which has the information of buffered frames.

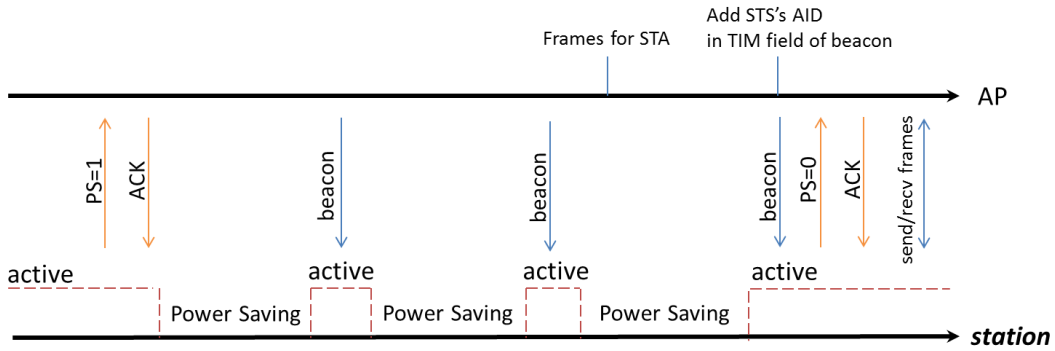


Fig15-9 Timeline of power save

In SDK, IEEE 802.11 power management is called LPS, and if KM4 enters sleep mode when Wi-Fi is in LPS mode, we call it WoWLAN mode.

In WoWLAN mode, a timer with a period of about 102ms will be set in the suspend function, and KM0 will wake up every 102ms to receive the beacon to maintain the connection.

Except LPS and WoWLAN, we also have IPS, which can be used when Wi-Fi is not connected. Table15-2 lists all three power save mode for Wi-Fi.

Table15-2 Wi-Fi power save mode

Items	Wi-Fi Status	Comment
IPS	Not associated	Driver automatically turns off Wi-Fi to save power.
LPS	Associated	LPS is used to implement IEEE 802.11 power management. KM0 will control RF ON/OFF based on TSF and TIM IE in beacon.
WoWLAN	Associated	LPS + Tickless KM0 will wakeup KM4 when receiving data packet.

15.1.3.5 Wakeup Time

CPU can execute IRQ handler considering that system is wakeup, it takes about 2.8ms to wake KM0 only and 3.5ms to wake both KM0 and KM4 in KM0CG + KM4 PG mode. For KM0, 1.6ms to initialize hardware (mainly XTAL), 1.1ms to execute critical code, and 300us to enter IRQ handler.

15.1.4 Sleep Mode Configuration

KM0 sleep mode is CG by default and KM4 sleep mode can be selected to CG or PG by setting parameter in Table15-8.

15.1.4.1 Wakeup Source Setup

Table15-3 Sleep wakeup source

Wakeup source	Description	Default status
BIT_LP_WEVT_HS_MSK	1: enable HS wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_AON_MSK	1: enable AON wakeup event 0: disable the event to wake up the system NOTE *It is a master switch for all AON wakeup sources. And there has 6 AON wakeup sources, listed in* Table15-4.	ON
BIT_LP_WEVT_SGPIO_MSK	1: enable SGPIO wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_COMP_MSK	1: enable comparator wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_ADC_MSK	1: enable ADC wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_I2C_MSK	1: enable I2C wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_UART_MSK	1: enable UART wakeup event 0: disable the event to wake up the system	ON
BIT_LP_WEVT_BOD_MSK	1: enable BOD wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_WLAN_MSK	1: enable WLAN wakeup event 0: disable the event to wake up the system	ON
BIT_LP_WEVT_GPIO_MSK	1: enable GPIO wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_SWRD_OCP_M	1: enable DCORE SWR OCP wakeup event 0: disable the event to wake up the system	OFF
BIT_LP_WEVT_TIMER_MSK	1: enable timer wakeup LP event 0: disable the event to wake up the system	OFF

Table15-4 Sleep AON wakeup sources for BIT_LP_WEVT_AON_MSK

Wakeup source	Description	Default status
BIT_CHIP_PDB_MSK	1: Enable chip powered down wake	ON
BIT_CAPTOUCH_WAKE_MSK	1: Enable CapTouch wake	ON
BIT_KEYSCAN_WAKE_MSK	1: enable KeyScan wake	ON
BIT_DLPS_TSF_WAKE_MSK	1: enable TSF wake under dlps mode	OFF
BIT_RTC_WAKE_MSK	1: enable RTC wake	OFF
BIT_AON_WAKE_TIM0_MSK	1: enable AON timer wake	ON
BIT_GPIO_WAKE_MSK	1: enable GPIO wake	ON

To enable a specific wakeup source, the corresponding status in array sleep_wevent_config[] in rtl8721dlp_sleepcfg.c should be set firstly, as shown in Fig15-10.

```
/* if X can wakeup dsleep, it can wakeup dstandby & sleep */
/* if X can wakeup dstandby, it can wakeup sleep */
PWRCFG_TypeDef sleep_wevent_config[]=
{
//      Module                                  Status
    {BIT_LP_WEVT_HS_MSK,                    OFF},
    {BIT_LP_WEVT_AON_MSK,                   ON},
    {BIT_LP_WEVT_SGPIO_MSK,                 OFF},
    {BIT_LP_WEVT_COMP_MSK,                  OFF},
    {BIT_LP_WEVT_ADC_MSK,                   OFF},
    {BIT_LP_WEVT_I2C_MSK,                   OFF},
    {BIT_LP_WEVT_UART_MSK,                  ON},
    {BIT_LP_WEVT_BOD_MSK,                   OFF},
    {BIT_LP_WEVT_WLAN_MSK,                  ON},
    {BIT_LP_WEVT_GPIO_MSK,                  OFF},
    {BIT_LP_WEVT_SWRD_OCP_MSK,              OFF},
    {BIT_LP_WEVT_TIMER_MSK,                 OFF},

    {0xFFFFFFFF,                            OFF},   /* Table end */
};
```

Fig15-10 Sleep wake source setup

If the AON wakeup event is chosen as the wakeup source, a specific AON wakeup source should be selected in sleep_aon_wevent_config[] in rtl8721dlp_sleepcfg.c, as shown in Fig15-11.

```
PWRCFG_TypeDef sleep_aon_wevent_config[]=
{
//      Module                                  Status
    {BIT_CHIP_PDB_MSK,          ON},    /* [7]  R/W 0       1: Indicate chip power-down */
    {BIT_CAPTOUCH_WAKE_MSK,     ON},    /* [6]  R/W 0       1: Indicate captouch wake event */
    {BIT_KEYSCAN_WAKE_MSK,      ON},    /* [4]  R/W 0       1: Indicate keyscan wake */
    {BIT_DLPS_TSF_WAKE_MSK,     OFF},   /* [3]  R/W 0       1: Indicate tsf wake under deep-lps mode */
    {BIT_RTC_WAKE_MSK,          OFF},   /* [2]  R/W 0       1: Indicate RTC wake */
    {BIT_AON_WAKE_TIM0_MSK,     ON},    /* [1]  R/W 0       1: Indicate AON timer wake */
    {BIT_GPIO_WAKE_MSK,         ON},    /* [0]  R/W 0       1: Indicate GPIO wake, see aon_wakepin & dsleep_wakepin_config */

    {0xFFFFFFFF,                        OFF},   /* Table end */
};
```

Fig15-11 Sleep AON wake source setup

The wakeup sources mentioned above are for KM0. You can choose if waking KM4 system up at the same time according to Table15-5.

Table15-5 Wakeup sources for KM4

Wakeup source	Description	Default status
BIT_LP_WEVT_HS_STS	1: Indicates HS wakeup event	ON
BIT_LP_WEVT_AON_STS	1: Indicates AON wakeup event	OFF
BIT_LP_WEVT_SGPIO_STS	1: Indicates SGPIO wakeup event	OFF
BIT_LP_WEVT_COMP_STS	1: Indicates Comparator wakeup event	OFF
BIT_LP_WEVT_ADC_STS	1: Indicates ADC wakeup event	OFF
BIT_LP_WEVT_I2C_STS	1: Indicates I2C wakeup event	OFF
BIT_LP_WEVT_UART_STS	1: Indicates UART wakeup event	OFF
BIT_LP_WEVT_BOD_STS	1: Indicates BOD wakeup event	OFF
BIT_LP_WEVT_WLAN_STS	1: Indicates WLAN wakeup event	OFF
BIT_LP_WEVT_GPIO_STS	1: Indicates GPIO wakeup event	OFF
BIT_LP_WEVT_SWRD_OCP_S	1: Indicates DCORE SWR OCP wakeup event	OFF
BIT_LP_WEVT_TIMER_STS	1: Indicates timer wakeup LP event	OFF
BIT_CHIP_PDB_STS	1: Indicates chip powered down	OFF
BIT_CAPTOUCH_WAKE_STS	1: Indicates CapTouch wake event	ON
BIT_KEYSCAN_WAKE_STS	1: Indicates KeyScan wake	ON
BIT_DLPS_TSF_WAKE_STS	1: Indicates TSF wake for WiFi FW	OFF
BIT_RTC_WAKE_STS	1: Indicates RTC wake	ON
BIT_AON_WAKE_TIM0_STS	1: Indicates AON Timer wake	OFF
BIT_GPIO_WAKE_STS	1: Indicates AON wakepin wake	ON

To enable a specific wakeup source for KM4, corresponding status in array hs_wakeevent_config [] in rtl8721dlp_sleepcfg.c should be set firstly, as shown in Fig15-12.

```
HSWAKEEVENT_TypeDef hs_wakeevent_config[]=
{
//      Module                                  Event                       Status
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_HS_STS,          ON},    /* [30] 1: Indicate HS Wakeup event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_AON_STS,         OFF},   /* [29] 1: Indicate AON Wakeup event (0x128) */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_SGPIO_STS,       OFF},   /* [28] 1: Indicate SGPIO Wakeup event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_COMP_STS,        OFF},   /* [27] 1: Indicate Comparetor Wakeup  event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_ADC_STS,         OFF},   /* [26] 1: Indicate  ADC Wakeup  event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_I2C_STS,         OFF},   /* [24] 1: Indicate I2C  Wakeup  event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_UART_STS,        OFF},   /* [20] 1: Indicate UART  Wakeup  event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_BOD_STS,         OFF},   /* [6]  1: Indicate BOD Wakeup  event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_WLAN_STS,        OFF},   /* [5]  1: Indicate WLAN Wakeup  event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_GPIO_STS,        OFF},   /* [4]  1: Indicate GPIO Wakeup event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_SWRD_OCP_STS,    OFF},   /* [2]  1: Indicate DCORE SWR OCP event */
    {REG_LP_SLP_WAKE_EVENT_STATUS0, BIT_LP_WEVT_TIMER_STS,       OFF},   /* [1]  1: Indicate GTimer Wakeup system event; */

    {REG_AON_WAKE_OPT_STS,          BIT_CHIP_PDB_STS,            OFF},   /* [7]  1: Indicate chip power-down */
    {REG_AON_WAKE_OPT_STS,          BIT_CAPTOUCH_WAKE_STS,       ON},    /* [6]  1: Indicate captouch wake event */
    {REG_AON_WAKE_OPT_STS,          BIT_KEYSCAN_WAKE_STS,        ON},    /* [4]  1: Indicate keyscan wake */
    {REG_AON_WAKE_OPT_STS,          BIT_DLPS_TSF_WAKE_STS,       OFF},   /* [3]  1: Indicate tsf wake under deep-lps mode */
    {REG_AON_WAKE_OPT_STS,          BIT_RTC_WAKE_STS,            ON},    /* [2]  1: Indicate RTC wake */
    {REG_AON_WAKE_OPT_STS,          BIT_AON_WAKE_TIM0_STS,       OFF},   /* [1]  1: Indicate AON timer wake */
    {REG_AON_WAKE_OPT_STS,          BIT_GPIO_WAKE_STS,           ON},    /* [0]  1: Indicate GPIO wake, see aon_wakepin & dsleep_wakepin_config */

    {0xFFFFFFFF,                    OFF,                         0},     /* Table end */
};
```

Fig15-12 Sleep wake source setup for KM4

NOTE

The power of KM4 is controlled by KM0. KM0 will decide whether to wakeup KM4 according to the wakeup source and the status of the wakeup source in array hs_wakeevent_config[].

Some peripherals like HS UART need ANA4M, and it can be enabled in km0_pwrmgt_config[] in rtl8721dlp_sleepcfg.c, as shown in Fig15-13.

```
PWRCFG_TypeDef km0_pwrmgt_config[]=
{
//      Module                                  Status
    {BIT_LSYS_PST_SLEP_EACK,            OFF},   //BIT1: 1 Enable ANA4M CLK when PMC enter into sleep mode
    {BIT_LSYS_PST_SLEP_EBUS,            OFF},   //BIT2: 1 Enable platform clock when PMC entro into sleep mode
    {BIT_LSYS_PST_SLEP_EMPM,            OFF},   //BIT3: 1 means RAM can not enter low power mode, 0 can enter,
    {BIT_LSYS_PST_SLEP_LDLM,            OFF},   //BIT4 0 LPSDO enter sleep mode
    {BIT_LSYS_PST_SLEP_ERCK,            OFF},   //BIT5 0 gate ls system clock
    {BIT_LSYS_PST_SLEP_DPSW,            ON},    //BIT6: 1: disable power switch and use SWR mode, 0: enable po
    {BIT_LSYS_PST_SLEP_EPWM,            OFF},   //BIT7: 1 LDO mode: 1.1V or SWR PWM mode, 0 LDO mode 0.9v or S
    {BIT_LSYS_PST_SLEP_ESWR,            ON},    //BIT8: 0 is disable SWR when enter sleep
    {BIT_LSYS_PST_SLEP_EXTL,            OFF},   //BIT9 0 disable XTAL when sleep, should conflict with  BIT_LS
    {BIT_LSYS_PST_SLEP_XACT,            ON},    //BIT10 based on BIT_LSYS_PST_SLEP_EXTL

    {0xFFFFFFFF,                        OFF},   /* Table end */
};
```

Fig15-13 Power management configuration

Some peripherals also need clock OSC2M in sleep mode. Whether to close clock OSC2M when the system is in sleep mode is determined by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig15-14. Value TRUE means it will close clock OSC2M in sleep mode.

```
PSCFG_TypeDef ps_config = {
    .km0_config_wifi_enable = TRUE,
    .km0_enable_key_touch = FALSE,//BIT_KEY_ENABLE | BIT_CAPTOUCH_ENABLE,
    .km0_tickles_debug = FALSE,/* if open WIFI FW, should close it, or beacon will lost in WOWLAN */
    .km0_osc2m_close = TRUE,
    .km0_pg_enable = FALSE,
    .km0_rtc_calibration = FALSE,
    .km0_audio_pad_enable = TRUE,
};
```

Fig15-14 Control of clock OSC2M in sleep mode

15.1.4.1.1 HS Wakeup Event (HS UART)

HS wakeup event includes wakeup sources of HS UART (UART0). When using HS wakeup event as the wakeup source, clock ANA4M should not be closed. So BIT_LSYS_PST_SLEP_EACK should set to ON, as shown in Fig15-13.

When using UART0 as wakeup source, clock OSC2M: km0_osc2m_close in Fig 15-14 should equal FALSE.

The following steps show how to configure HS UART in sleep mode.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig 15-14.
(2) Set BIT_LSYS_PST_SLEP_EACK in structure km0_pwrmgt_config as shown in Fig 15-13.
(3) Initialize UART and enable its interrupt.
(4) Switch UART to low power Rx mode and switch clock source to OSC2M. Using UART mbed API (serial_api.c): Set corresponding LOW_POWER_RX_ENABLE item to ENABLE in structure uart_config[] as shown in Fig 15-15 and then it will switch automatically in function serial_baud(). If using low level API, please do it manually as shown in Fig 15-16.

```
UARTCFG_TypeDef uart_config[4]=
{
    /* HS UART--> UART0 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE, /*to enable low power RX*/
    },
    /* BT UART--> UART1 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE,
    },
    /* Log UART--> UART2 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE,
    },
    /* LPUART-->UART3 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE,
    },
};
```

Fig 15-15 Configure low power RX mode for UART mbed API

```
UART_MonitorParaConfig(UART0_DEV, 100, ENABLE);
UART_RxMonitorCmd(UART0_DEV, ENABLE);            //low power rx monitor
RCC_PeriphClockSource_UART(UART0_DEV, UART_RX_CLK_OSC_LP); //switch clock
UART_LPRxBaudSet(UART0_DEV, baudrate, 2000000);      //low power rx mode
UART_RxCmd(UART0_DEV, ENABLE);
```

Fig 15-16 Configure low power RX mode manually for UART low level API

(5) Set BIT_LP_WEVT_HS_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10.
(6) Set BIT_LP_WEVT_HS_STS status ON in array hs_wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.
(7) Set HS UART wake event in KM4 platform before sleep by using "SOCPS_SetWakeEvent_HP(BIT_HS_WEVT_UART_MSK, 1);"

15.1.4.1.2 AON

AON wakeup event includes PDB, CapTouch, Key-Scan, RTC, AON timer and GPIO.

PDB

(1) Change PMC power down direction to SW by register AON_PM_OPT.
(2) Enable the interrupt of PDB by register AON_CHIP_PWR_DOWN_MSK.
(3) Set BIT_LP_WEVT_AON_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10 and set BIT_CAPTOUCH_WAKE_MSK status ON in array sleep_aon_wevent_config[] as shown in Fig 15-11.
(4) Set BIT_LP_WEVT_AON_STS and BIT_CHIP_PDB_STS status ON in array hs_wakeevent_config [] as shown in Fig 15-12 if you want to wakeup KM4 at the same time.

Cap-Touch

(1) Initialize CapTouch and enable its interrupts.
(2) Set BIT_LP_WEVT_AON_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10 and set BIT_CHIP_PDB_MSK status ON in array sleep_aon_wevent_config[] as shown in Fig 15-11.
(3) Set BIT_LP_WEVT_AON_STS and BIT_CAPTOUCH_WAKE_STS status ON in array hs_wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

NOTE

w Ŝ Ŧ ŜNJ ű 2 Ŏ 2 Ř Ŝ CapTouch initialization Ŝ è Φ Ŏ Ω Ŧ 2 NJ

Key-Scan

(1) Initialize KeyScan and enable its interrupts.
(2) Set BIT_LP_WEVT_AON_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10 and set BIT_KEYSCAN_WAKE_MSK status ON in array sleep_aon_wevent_config[] as shown in Fig 15-11.
(3) Set BIT_LP_WEVT_AON_STS and BIT_KEYSCAN_WAKE_STS status ON in array wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

NOTE

w Ŝ Ŧ Ŝ NJ ű 2 Ŏ 2 Ř Ŝ Scan initialization. Ŝ è Φ Ŏ Ω Ŧ 2 NJ Y Ŝ è

RTC

(1) Initialize RTC and enable its interrupts.
(2) Set BIT_LP_WEVT_AON_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10 and set BIT_RTC_WAKE_MSK status ON in array sleep_aon_wevent_config[] as shown in Fig 15-11.
(3) Set BIT_LP_WEVT_AON_STS and BIT_RTC_WAKE_STS status ON in array wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

AON Timer

(1) Enable AON Timer and set the time to wake up system.
(2) Set BIT_LP_WEVT_AON_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10 and set BIT_AON_WAKE_TIM0_MSK status ON in array sleep_aon_wevent_config[] as shown in Fig 15-11.
(3) Set BIT_LP_WEVT_AON_STS and BIT_AON_WAKE_TIM0_STS status ON in array wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

NOTE

The AON Timer is imprecise milliseconds Its maximum value is 32760000ms (546mins).

GPIO (Wakepin)

(1) The setup of GPIO is shown in 15.1.6.1.5
(2) Set BIT_LP_WEVT_AON_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10 and set BIT_GPIO_WAKE_MSK status ON in array sleep_aon_wevent_config[] as shown in Fig 15-11.
(3) Set BIT_LP_WEVT_AON_STS and BIT_GPIO_WAKE_STS status ON in array wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

15.1.4.1.3 SGPIO

When dz ă λ y˝ ∃ { D t L h l· ă ø l· ł Ŝ dzLJ ă 2 dzNJ Ŏ Ŝ Σ Ŏ f 2 Ŏ ł h { / н a ă K 2 dz f Ř y˝ Ω ű ō Ŝ Ŏ f 2 ă Ŝ Ř system is in sleep mode is decided by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig 15-14.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig 15-14.
(2) Initialize SGPIO and enable its interrupt.
(3) Set BIT_LP_WEVT_SGPIO_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10.
(4) Set BIT_LP_WEVT_SGPIO_STS status ON in array wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.
(5) Switch SGPIO clock to OSC2M by register REG_SYS_CLK_CTRL1 before entering sleep mode.

15.1.4.1.4 Comparator

When dz ă λ y˝ ∃ / 2 Y LJ l· NJ l· ű 2 NJ l· ă ø l· ł Ŝ dzLJ ă 2 dzNJ Ŏ Ŝ Σ Ŏ f 2 Ŏ ł h { / н a ă K 2 dz f Ř y˝ Ω ű ō Ŝ Ŏ when system is in sleep mode is decided by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig 15-14.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig 15-14.
(2) Initialize ADC and comparator, and enable corresponding interrupt.
(3) Set BIT_LP_WEVT_COMP_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10.
(4) Set BIT_LP_WEVT_COMP_STS status ON in array wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

(5) Switch ADC clock to OSC2M by register REG_SYS_CLK_CTRL1 before entering sleep mode.

15.1.4.1.5 ADC

When dz ă ƛ ƴ Ǝ ! 5 / ǀ ă ø ǀ ț Ŝ dzLJ ă ƨ dzNJ Ŏ Ŝ Σ Ŏ f ƨ Ŏ ț h { / н a ă Ƙ ƨ dz f Ř ƴ Ω ű ō Ŝ Ŏ f ƨ ă Ŝ Ř ø system is in sleep mode is decided by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig15-14.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig15-14.
(2) Initialize ADC and enable its interrupt.
(3) Set BIT_LP_WEVT_ADC_MSK status ON in array sleep_wevent_config[] as shown in Fig15-10.
(4) Set BIT_LP_WEVT_ADC_STS status ON in array hs_wakeevent_config [] as shown in Fig15-12 if you want to wake up KM4 at the same time.
(5) Switch ADC clock to OSC2M by register REG_SYS_CLK_CTRL1 before entering sleep mode.

15.1.4.1.6 I2C

When dz ă ƛ ƴ Ǝ L н / ǀ ă ø ǀ ț Ŝ dzLJ ă ƨ dzNJ Ŏ Ŝ Σ Ŏ f ƨ Ŏ ț h { / н a ǀ ƴ Ř ! b ! п a ă Ƙ ƨ dz C2M Ř ƴ Ω ű ō when system is in sleep mode is decided by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig15-14, while clock ANA4M can be configured in km0_pwrmgt_config[] in rtl8721dlp_sleepcfg.c, as shown in Fig15-13.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig15-14, and set BIT_LSYS_PST_SLEP_EACK status ON in array km0_pwrmgt_config[], as shown in Fig15-13.
(2) Initialize I2C and enable its interrupt.
(3) Set BIT_LP_WEVT_I2C_MSK status ON in array sleep_wevent_config[] as shown in Fig15-10.
(4) Set BIT_LP_WEVT_I2C_STS status ON in array hs_wakeevent_config [] as shown in Fig15-12 if you want to wake up KM4 at the same time.
(5) Switch I2C clock to OSC2M by calling RCC_PeriphClockSource_I2C() before entering sleep mode.

15.1.4.1.7 LP UART

When using LP UART (UART3) as wakeup source, clock OSC2M should be set Ř ƴ Ω ű ō Ŝ Ŏ f ƨ ă Ŝ Ř ø Ƙ Ŝ ƴ ă è ă ű Ŝ Y ƛ ă ƛ ƴ ă OSC2M when system is in sleep mode is decided by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig15-14.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig15-14.
(2) Initialize UART and enable its interrupt.
(3) Switch UART to low power Rx mode and switch clock source to OSC2M. Using UART mbed API (serial_api.c), Set corresponding LOW_POWER_RX_ENABLE to ENABLE in structure uart_config[] as shown in Fig15-15 and then it will switch automatically in function serial_baud() . If using low level API, please do it manually as shown in Fig15-16.
(4) Set BIT_LP_WEVT_UART_MSK status ON in array sleep_wevent_config[] as shown in Fig15-10.
(5) Set BIT_LP_WEVT_UART_STS status ON in array hs_wakeevent_config [] as shown in Fig15-12 if you want to wake up KM4 at the same time.

15.1.4.1.8 Log UART

When using Log UART (UART2) ǀ ă ø ǀ ț Ŝ dzLJ ă ƨ dzNJ Ŏ Ŝ Σ Ŏ f ƨ Ŏ ț h { / н a ă Ƙ ƨ dz f Ř ƴ Ω ű ō Ŝ Ŏ f ƨ ă Ŝ Ř ø OSC2M when system is in sleep mode is decided by km0_osc2m_close in structure ps_config in rtl8721dlp_sleepcfg.c, as shown in Fig15-14.

(1) Set km0_osc2m_close = FALSE in structure ps_config as shown in Fig15-14.
(2) Initialize UART and enable its interrupt.
(3) Set BIT_LP_WEVT_UART_MSK status ON in array sleep_wevent_config[] as shown in Fig15-10.
(4) Set BIT_LP_WEVT_UART_STS status ON in array hs_wakeevent_config [] as shown in Fig15-12 if you want to wake up KM4 at the same time.

15.1.4.1.9 BOD

(1) Set BOD threshold and enable its interrupt.
(2) Set BIT_LP_WEVT_BOD_MSK status ON in array sleep_wevent_config[] as shown in Fig15-10.

(3) Set BIT_LP_WEVT_BOD_STS status ON in array ls_wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

15.1.4.1.10 WLAN

(1) Set BIT_LP_WEVT_WLAN_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10.
(2) Set BIT_LP_WEVT_WLAN_STS status ON in array ls_wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

15.1.4.1.11 GPIO

(1) Initialize a GPIO and enable its interrupt.
(2) Set BIT_LP_WEVT_GPIO_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10.
(3) Set BIT_LP_WEVT_GPIO_STS status ON in array ls_wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

15.1.4.1.12 LPTimer

When dz ă ƛ y˝ Ǝ ű ƛ Y Ŝ NJ ƛ y˝ Y a л ŀ ă ø ŀ ļ Ŝ dzLJ ă ƨ dzNJ Ŏ Ŝ Σ Ŏ f ƨ Ŏ ļ ! b ! п a ă K ƨ dz ƛ Ř y˝ Ω ű ō Ŝ can be configured in km0_pwrmgt_config[] in rtl8721dlp_sleepcfg.c, as shown in Fig 15-13.

(1) Set BIT_LSYS_PST_SLEP_EACK status ON in array km0_pwrmgt_config[], as shown in Fig 15-13.
(2) Initialize the timer and enable its interrupt in KM0 platform.
(3) Set BIT_LP_WEVT_TIMER_MSK status ON in array sleep_wevent_config[] as shown in Fig 15-10.
(4) Set BIT_LP_WEVT_TIMER_STS status ON in array ls_wakeevent_config [] as shown in Fig 15-12 if you want to wake up KM4 at the same time.

NOTE

LP Timer (TIMM00-05) belongs to KM0, and only TIM00-03 can be wakeup source. Please distinguish them with HS timer (TIM10) which can not be wakeup source.

15.1.4.2 How to Enter Sleep Mode

Function pmu_release_wakelock(PMU_OS) can be called in KM4 if the system wants to enter into sleep mode. The global parameter wakelock will be checked in idle task, once wakelock equals zero and there is nothing to do, then system will enter into sleep mode.

15.1.4.3 How to Wake Up from Sleep Mode

15.1.4.3.1 HS Wakeup Event

The chip will wake up from sleep mode when the specific interrupt that enabled is trigged.

NOTE

Switch the clock back after the system wakes up from sleep mode if the clock of the source has been changed before entering sleep mode.

15.1.4.3.2 AON

PDB

The chip will wake up from sleep mode when the interrupt of PDB that enabled is trigged.

Cap-Touch

The chip will wake up from sleep mode when the interrupt of Cap-Touch that enabled is trigged.

Key-Scan

The chip will wake up from sleep mode when the interrupt of f-Scan that enabled is trigged.

RTC

The chip will wake up from sleep mode when the interrupt of RTC that enabled is trigged.

AON Timer

¢ K Ŝ Ŏ K ƛ LJ ø ƛ f f ø l· ļ Ŝ dzLJ ₮ NJ 2 Y ă f Ŝ Ŝ LJ Y 2 Ř Ŝ ø K Ŝ y ű K Ŝ ă è ă ű Ŝ Y K l· ă l· f NJ Ŝ l·

GPIO (Wakepin)

The chip will wake up from sleep mode when there is a corresponding edge of AON pin(s) that setup in section 15.1.6.1.5

15.1.4.3.3 SGPIO

The chip will wake up from sleep mode when the interrupt of SGPIO that enabled is trigged.

NOTE

Switch SGPIO clock back to LS APB Clock by register REG_SYS_CLK_CTRL1 after the system wakes up from sleep mode.

15.1.4.3.4 Comparator

The chip will wake up from sleep mode when the interrupt of comparator that enabled is trigged.

NOTE

Switch comparator clock back to LS APB Clock by register REG_SYS_CLK_CTRL1 after the system wakes up from sleep mode.

15.1.4.3.5 ADC

The chip will wake up from sleep mode when the interrupt of ADC that enabled is trigged.

NOTE

Switch ADC clock back to LS APB Clock by register REG_SYS_CLK_CTRL1 after the system wakes up from sleep mode.

15.1.4.3.6 I2C

The chip will wake up from sleep mode when the interrupt of I2C that enabled is trigged.

NOTE

Switch I2C clock back to LS APB Clock by calling RCC_PeriphClockSource_I2C() after the system wakes up from sleep mode .

15.1.4.3.7 LP UART

The chip will wake up from sleep mode when the interrupt of UART that enabled is trigged.

NOTE

Switch UART clock back to XTAL by calling RCC_PeriphClockSource_UART() after the system wakes up from sleep mode if need high baud rate.

15.1.4.3.8 Log UART

The chip will wake up from sleep mode when the interrupt of UART that enabled is trigged.

15.1.4.3.9 BOD

The chip will wake up from sleep mode when the interrupt of BOD that enabled is trigged.

15.1.4.3.10 WLAN

The chip will wakeup from sleep mode every beacon interval, and the default interval is 102.4ms.

15.1.4.3.11 GPIO

The chip will wake from sleep mode when the interrupt of GPIO that enabled is trigged.

NOTE

Switch GPIO clock back to LS APB by register REG_SYS_CLK_CTRL1 after system wakes up from sleep mode.

15.1.4.3.12 Timer

The chip will wakeup ŦNJ2Y ă f ŜŜLJ Y2ŘŜ ó KŜy ú KŜ ă è ă ú ŜY K l· ă l· f NJŜ l· Ř è ă f ŜLJ ú Ŧ2NJ

15.1.4.4 How to Get Wakeup Information

Table15-6 Sleep wakeup information

API	Introduction	Parameters
WakeEvent	Gets sleep wake reason It can only be accessed in KM0	Bit[1]: GTimer Bit[2]: DCORE SWR OCP Bit[4]: GPIO Bit[5]: WLAN Bit[6]: BOD Bit[20]: UART Bit[24]: I2C Bit[26]: ADC Bit[27]: Comparator Bit[28]: SGPIO Bit[29]: AON Bit[30]: HS
int SOCPS_AONWakeReason(vo	Gets AON wake reason It can be accessed in KM0 and KM4	Parameter: None Retval status value: Bit[0]: AON wakepin Bit[1]: AON Timer Bit[2]: RTC Bit[3]: TSF Timer Bit[4]: KeyScan Bit[6]: CapTouch
int SOCPS_WakePinCheck(void)	Gets AON Wakepin index It can be accessed in KM0 and KM4	Parameter: None Retval status value: Bit[0]: wakepin0 Bit[1]: wakepin1 Bit[2]: wakepin2 Bit[3]: wakepin3

15.1.5 Deepsleep Mode

Deepsleep mode is has a lower power consumption only AON domain is power on. So only peripheral in AON domain can wakeup chip. There are 2 ways to enter deepsleep mode. One is to release deepwake lock, the other is KM4 sends related IPC to KM0 directly. So If using way2, tickless is not necessary. Once KM0 receive IPC message, KM0 makes KM0 and KM4 enter deep sleep mode in IPC interrupt.

The wake process of deep sleep is chip boot on process. Fig15-17 shows deepsleep mode flow.

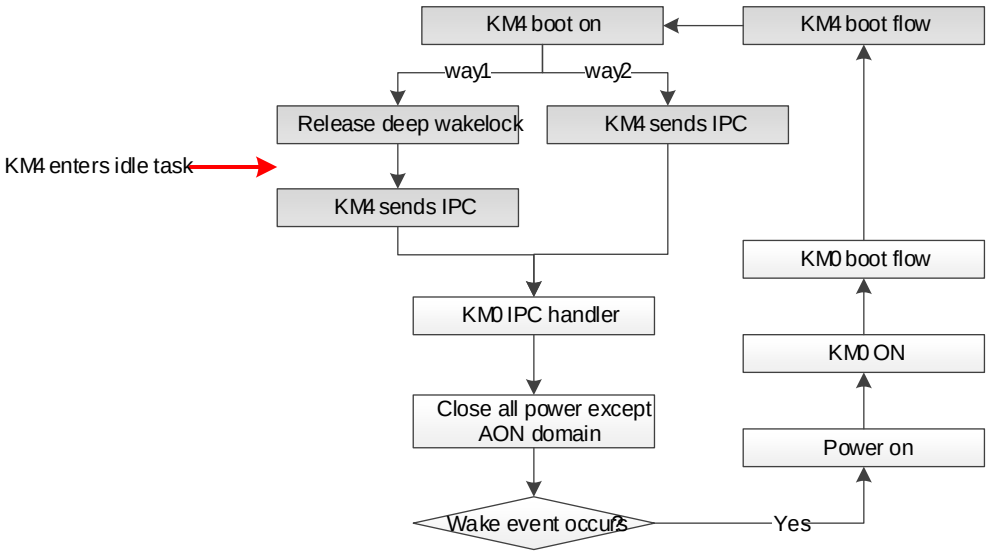


Fig15-17 Deepsleep mode flow

Fig15-18 showsthe code flow of IPC handler in deepsleep mode on KM0 platform.

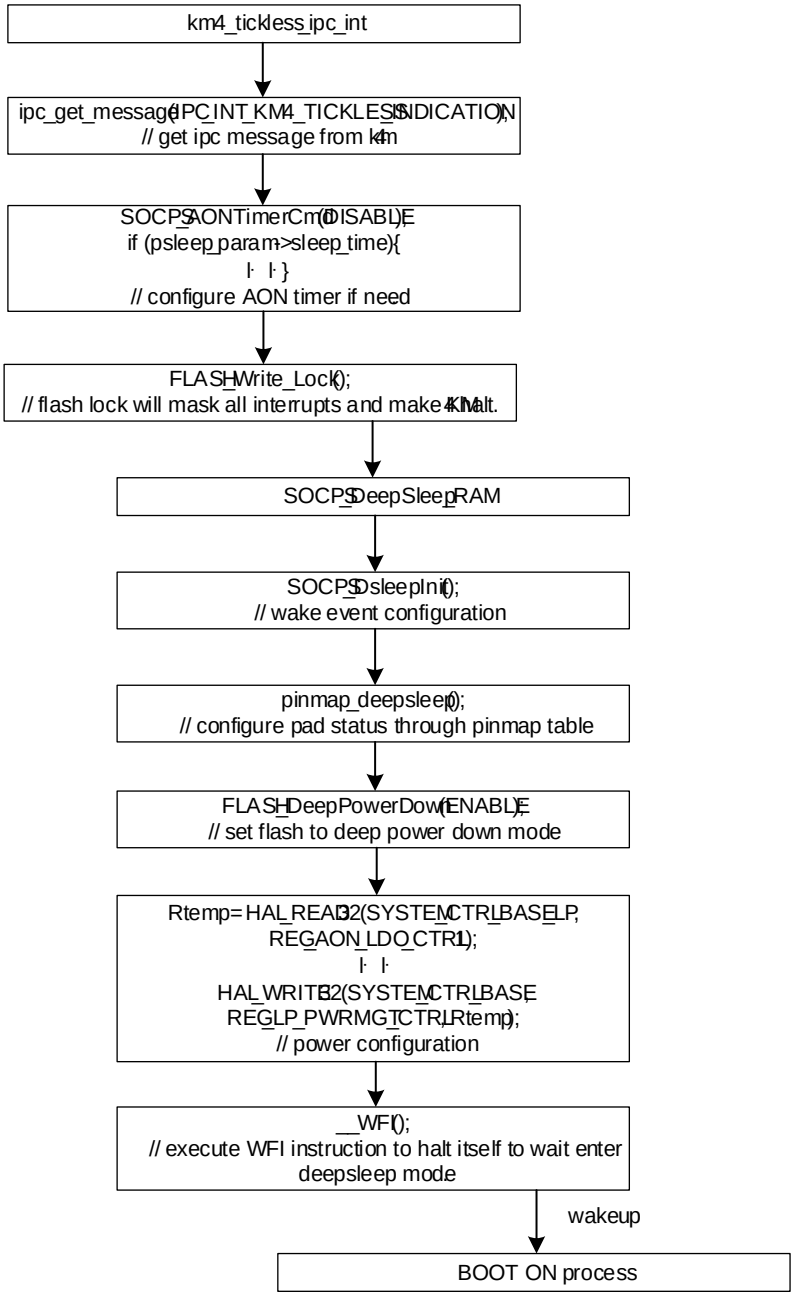


Fig15-18 Code flow of IPC handler in deepsleep mode

15.1.6 Deepsleep Mode Configuration

15.1.6.1 Wakeup Source Setup

Table15-7 Deepsleep wakeup source

Wakeupsource	Description	Default status
BIT_CAPTOUCH_WAKE_STS	IndicatesCapTouch wake event	ON
BIT_KEYSCAN_WAKE_STS	Indicates KeyScan wake	ON
BIT_DLPS_TSF_WAKE_STS	Indicates TSF wake for WiFi FW	ON
BIT_RTC_WAKE_STS	Indicates RTC wake	OFF

BIT_AON_WAKE_TIM0_STS	Indicates AON Timer wake	ON
BIT_GPIO_WAKE_STS	Indicates AON wakepin wake	ON

To set corresponding deepsleep source, firstly you need to set corresponding wakeup source Status ON in array dsleep_aon_wevent_config[] in rtl8721dlp_sleepcfg.c, as shown in Fig 15-19.

```
PWRCFG_TypeDef dsleep_aon_wevent_config[]=
{
//      Module                          Status
    {BIT_CAPTOUCH_WAKE_STS,             ON},    /* [6] R/W 0      1: Indicate captouch wake event */
    {BIT_KEYSCAN_WAKE_STS,              ON},    /* [4] R/W 0      1: Indicate keyscan wake */
    {BIT_DLPS_TSF_WAKE_STS,             ON},    /* [3] R/W 0      1: Indicate tsf wake under deep-lps mode */
    {BIT_RTC_WAKE_STS,                  OFF},   /* [2] R/W 0      1: Indicate RTC wake */
    {BIT_AON_WAKE_TIM0_STS,             ON},    /* [1] R/W 0      1: Indicate AON timer wake */
    {BIT_GPIO_WAKE_STS,                 ON},    /* [0] R/W 0      1: Indicate GPIO wake, see aon_wakepin & dsleep_wakepin_config */

    {0xFFFFFFFF,                        OFF},   /* Table end */
};
```

Fig 15-19 Deepsleep wake source setup

When KM4 wants to enter deep sleep mode, it will construct a global variable in structure KM4SLEEP_ParamDef as shown in Fig 15-20. Then transmit it to KM0 to inform KM0 execute corresponding operation. The variable can be local variable, or if KM4 releases the variable, KM0 may get wrong information.

NOTE

Asrtl8721dlp_sleepcfg.c is the code under KM0, if you modify it, you need to rebuild KM0 project, and if you are using IAR, KM4 project also needs to be rebuilt to sync the KM0 image.

```
typedef struct
{
    u8  dlps_enable;
    u8  sleep_type;
    u32 sleep_time;

} KM4SLEEP_ParamDef;
```

Fig 15-20 KM4 sleep parameter structure

Table 15-8 Structure KM4SLEEP_ParamDef member description

Member	Description
dlps_enable	Enable deep sleep mode: TRUE: enable deep sleep FALSEdisable deep sleep
sleep_type	Just for sleep mode usage SLEEP_PG: KM4 enter power gate mode SLEEP_CG: KM4 enter clock gate mode
sleep_time	If using AON Timer to wakeup chip, you should set the variable to control the time to sleep.

15.1.6.1.1 CapTouch

(1) Initialize CapTouch and enable its interrupts that you want to wakeup chip in deepsleep mode. As RAM and CPU are shut down in deepsleep mode while the CapTouch register keep alive, the interrupt handler and interrupt enable of CPU should be reinitialized, then CapTouch interrupts can be handled.

(2) SetBIT_CAPTOUCH_WAKE_STS Status ON in array dsleep_aon_wevent_config[] as shown in Fig 15-19.

NOTE

w Ŝ Ŧ ŜNJ ű ² Ŏ ² Ř Ŝ CapTouch initialization Ŝ è Ф Ŏ Ω Ŧ ² NJ

15.1.6.1.2 Key-Scan

(1) Initialize Key-Scan and enable its interrupts that you want to wakeup chip in deepsleep mode. As RAM and CPU are shut down in deepsleep mode while the Key-Scan register keep alive, the interrupt handler and interrupt enable of CPU should be reinitialized, then Key-Scan interrupts can be handled.
(2) Set BIT_KEYSCAN_WAKE_STS Status ON in array dsleep_aon_wevent_config[] as shown in Fig 15-19.

NOTE

w Ŝ Ŧ Ŝ NJ ű 2 Ŏ 2 Ř Ŝ Scan initialization.Ŝ è Φ Ŏ Ω Ŧ 2 NJ Y Ŝ è

15.1.6.1.3 RTC

(1) Initialize RTC and enable its interrupts that you want to wakeup chip in deepsleep mode. As RAM and CPU are shut down in deepsleep mode while the RTC register keep alive, the interrupt handler and interrupt enable of CPU should be reinitialized, then RTC interrupts can be handled.
(2) Set BIT_RTC_WAKE_STS Status ON in array dsleep_aon_wevent_config[] as shown in Fig 15-19.

15.1.6.1.4 AON Timer

(1) Set AON Timer and transmit it to KM0 by ipc_send_message(), the detailed steps is shown in Section 15.1.6.2
(2) Set BIT_AON_WAKE_TIM0_STS Status ON in array dsleep_aon_wevent_config[] as shown in Fig 15-19.

NOTE

The AON Timer is imprecise milliseconds. Its maximum value is 32760000ms (546mins).

15.1.6.1.5 GPIO

The GPIOs that can wake up chip in deepsleep is not all the common GPIOs, only 12 GPIOs can wake the chip, as shown in Fig 15-21, we call ű K Ŝ ă Ŝ ă LJ Ŝ Ŏ ƛ l· f D t L h ă Ψ ! h only four AON wakepins can be set at the same time. As shown in Fig 15-21, there are four AON wakepins (0, 1, 2, 3), each AON wakepin has three PINMUX (S0, S1, S2).

For each AON wakepin, you can only select one PINMUX at a time by the Module column with the Status column ON in array dsleep_aon_wevent_config[] in rtl8721dlp_sleepcfg.c, as shown in Fig 15-22. The Polarity column is used to choose wakeup edge, AmebaD only supports AON wakepin wake up the chip by edges. The value 0 means Falling Edge wakeup and value 1 means Rising Edge wakeup.

For AON wakepins, if setting Falling Edge wakeup, SDK will automatically set the pin internal pull up. And if setting Rising Edge wakeup, SDK will automatically set the pin internal pull down.

```
u8 aon_wakepin[4][3] = {
    {   /* wakepin 0 */
        _PA_12, /* PINMUX_S0 */
        _PA_16, /* PINMUX_S1 */
        _PA_20, /* PINMUX_S2 */
    },
    {   /* wakepin 1 */
        _PA_13, /* PINMUX_S0 */
        _PA_17, /* PINMUX_S1 */
        _PA_21, /* PINMUX_S2 */
    },
    {   /* wakepin 2 */
        _PA_14, /* PINMUX_S0 */
        _PA_18, /* PINMUX_S1 */
        _PA_25, /* PINMUX_S2 */
    },
    {   /* wakepin 3 */
        _PA_15, /* PINMUX_S0 */
        _PA_19, /* PINMUX_S1 */
        _PA_26  /* PINMUX_S2 */
    },
};
```

Fig 15-21 AON wakepins

```
WAKEPIN_TypeDef sleep_wakepin_config[]=
{
//   Module              Status        Polarity
     {PINMUX_S0,         OFF,            1},/* wakeup_0 config */
     {PINMUX_S0,         OFF,            1},/* wakeup_1 config */
     {PINMUX_S0,         OFF,            1},/* wakeup_2 config */
     {PINMUX_S0,         OFF,            1},/* wakeup_3 config */

     {0xFFFFFFFF,        OFF,            0},/* Table end */
};
```

Fig15-22 AON wakepins setup

AON wakepin setup flow:
(1) Set BIT_GPIO_WAKE_STS status ON in array dsleep_aon_wevent_config[] as shown Fig15-19.
(2) Set AON Wakepin and expected edge to wakeup chip in array sleep_wakepin_config[] as shown Fig15-22.

Example:
If want a failing edge of _PA_18 to wakeup chip, the AON wakepin setting is as shown Fig15-23. _PA_18 is PINMUX_S1 of wakepin 2.

```
WAKEPIN_TypeDef sleep_wakepin_config[]=
{
//   Module              Status        Polarity
     {PINMUX_S0,         OFF,            1},/* wakeup_0 config */
     {PINMUX_S0,         OFF,            1},/* wakeup_1 config */
     {PINMUX_S1,         ON,             0},/* wakeup_2 config */
     {PINMUX_S0,         OFF,            1},/* wakeup_3 config */

     {0xFFFFFFFF,        OFF,            0},/* Table end */
};
```

Fig15-23 AON Wakepins setup example

15.1.6.2 How to Enter Deepsleep Mode

To enter deepsleep mode, you should execute the following steps:
(1) Construct a global variable with the structure KM4SLEEP_ParamDef
(2) { Ŝ ú Y Ŝ Y ō Ŝ NJ Ψ Ř f LJ ă ψ g b l v a r i a b l e t o e n a b l e d e e p s l e e p m o d e ú K Ŝ
(3) { Ŝ ú Y Ŝ Y ō Ŝ NJ if you want to wakeup chip at specified time by AON Timer wakeup source
(4) / l f f Ψ ƛ LJ Ŏ ψ ă Ŝ y Ř ψ Y Ŝ ă ă l ∃ Ŝ ó L t / ψ L g l o b a l v a r i a b l e ψ Q L ú Y [ó Ŝ { ψ L Y ō ō L / Ŝ È Ŝ Ŏ d z Ŝ ó Ŝ z ó ú

NOTE

Refer to example project \realtek_amebaD_va0_example\example_sources\SPMC\raw\LJ Y ψ Ř ă f LJ Ω Φ

Release deepwakelock is also a way to enter deepsleep mode but not recommend.

15.1.6.3 How to WakeUp from Deepsleep Mode

When the chip wakes up from deep sleep, it will do the boot process.

15.1.6.3.1 Cap-Touch

The interrupt of Cap-Touch that enabled happens will wake up the chip.

15.1.6.3.2 Key-Scan

The interrupt of Key-Scan that enabled happens will wake up the chip.

15.1.6.3.3 RTC

The interrupt of RTC that enabled happens will wake up the chip.

15.1.6.3.4 AON Timer

[illegible] reach [illegible] will wake up the chip.

15.1.6.3.5 GPIO

The corresponding edge of AON wakepin(s) that setup in section 15.1.6.1.5 happens will wake up the chip.

15.1.6.4 How to Get Wakeup Information

Table 15-9 Deepsleep wakeup information API

API	Introduction	Parameters
u32 OCPS_DsleepWakeStatusGet(void	Gets deepsleep wake status	Parameter: None Retval status value: TRUE: boot from deepsleep FALSE: boot from power on
int SOCPS_AONWakeReason(void)	Gets deepsleep wake reason	Parameter: None Retval status value: Bit[0]: AON wakepin Bit[1]: AON Timer Bit[2]: RTC Bit[3]: TSF Timer Bit[4]: Key-Scan Bit[6]: Cap-Touch
int SOCPS_WakePinCheck(void)	Gets AON wakepin index	Parameter: None Retval status value: Bit[0]: wakepin0 Bit[1]: wakepin1 Bit[2]: wakepin2 Bit[3]: wakepin3

15.1.6.5 How to Maintain Information in Deepsleep Mode

The memory of Retention RAM can be kept when the chip enters deepsleep mode, so we can use it to keep some necessary information. RRAM_USER_RSVD is reserved for customer usage as shown in Figure 15-24.


```
/******************************************************************************//**
 * @defgroup AMEBAD_RRAM
 * @{
 * @brief AMEBAD_RRAM Declaration
 * @the Max space for RRAM_TypeDef is 0xB0, user can alloc space from RRAM_USER_RSVD
 *****************************************************************************/
typedef struct {
    uint8_t RRAM_WIFI_STATUS; /* RETENTION_RAM_SYS_OFFSET 0x80 */
    uint8_t RRAM_WIFI_P_SECURTY;
    uint8_t RRAM_WIFI_G_SECURTY;
    uint8_t RRAM_RSVD1;

    uint32_t RRAM_NET_IP;
    uint32_t RRAM_NET_GW;
    uint32_t RRAM_NET_GW_MASK;

    uint32_t FLASH_ID2;        /*offset 0x90*/
    uint32_t FLASH_cur_cmd;
    uint32_t FALSH_quad_valid_cmd;
    uint32_t FALSH_dual_valid_cmd;
    uint32_t FLASH_QuadEn_bit;  /*offset 0xA0*/
    uint32_t FLASH_StructInit;

    uint8_t FLASH_phase_shift_idx;
    uint8_t FLASH_rd_sample_phase_cal;
    uint8_t FLASH_class;
    uint8_t FLASH_cur_bitmode;

    uint8_t FLASH_ClockDiv;
    uint8_t FLASH_ReadMode;
    uint8_t FLASH_pseudo_prm_en;
    uint8_t FLASH_addr_phase_len;

    uint8_t FLASH_cmd_wr_status2;  /*offset 0xB0*/
    uint8_t FLASH_rd_dummy_cyle0;
    uint8_t FLASH_rd_dummy_cyle1;
    uint8_t FLASH_rd_dummy_cyle2;

    uint8_t RRAM_USER_RSVD[124];      /*usr can alloc from this RSVD space*/

} end {anonRRAM_TypeDef}  RRAM_TypeDef;
/** @} */
```

Fig15-24 Retention RAM usage

15.1.7 GPIO Pull Control

When entering sleep and deepsleep mode, I/O pull control is needed. Otherwise, it will results in power leakage. For more information, refer to AN0400 (section: User Configuration pinmapcfg).

15.2 Power Save Related APIs

API	Introduction
<pmu_register_sleep_callback>	Register suspend/resume call back function for one module
<pmu_unregister_sleep_callback>	Unregister suspend/resume call back function
<pmu_acquire_wakelock>	Acquire wakelock for one module
<pmu_release_wakelock>	Release wakelock
<pmu_set_sysactive_time>	Acquire wakelock, and release wakelock automatically after timeout
<pmu_set_max_sleep_time>	Set max sleep time

15.2.1 pmu_register_sleep_callback

Register suspend/resume call back function for <nDeviceId>. Suspend callback function will be called by PMU before system sleep in deep mode, and resume callback function will be called after system resume.

NOTE

Yield OS is not permitted in suspend/resume callback function like taskYIELD(), vTaskDelay, mutex, sema and so on.
pmu_set_sysactive_time is not permitted in suspend callback function, while permitted in resume call back function

Parameter	Type	Introduction
<nDeviceId>	uint32_t	Device ID need suspend/resume callback typedef enum { PMU_OS = 0, X PMU_MAX = 31 } PMU_DEVICE;
<sleep_hook_fun>	PSM_HOOK_FUN	Suspend call back function
<sleep_param_ptr>	void*	Suspend call back function parameter
<wakeup_hook_fun>	PSM_HOOK_FUN	Resume call back function
<wakeup_param_ptr>	void*	Resume call back function parameter

15.2.2 pmu_unregister_sleep_callback

Unregister suspend/resume call back function for <nDeviceId>.

Parameter	Type	Introduction
<nDeviceId>	uint32_t	The same as pmu_register_sleep_callback

15.2.3 pmu_acquire_wakelock

Wakelock is a 32bit map. Each module owns 1 bit in this bit map. FreeRTOS tickless reference wakelock and decides that whether it can enter sleep state.

If any module acquires and holds a bit in wakelock, then the whole system won't enter sleep state.

Parameter	Type	Introduction
<nDeviceId>	uint32_t	The same as pmu_register_sleep_callback

15.2.4 pmu_release_wakelock

Release bit[nDeviceId] of wakelock bit map. If wakelock equals to 0, then the system may enter sleep state after system idle.

Parameter	Type	Introduction
<nDeviceId>	uint32_t	The same as pmu_register_sleep_callback

15.2.5 pmu_set_sysactive_time

Set system active time. System cannot sleep before timeout.

NOTE

pmu_set_sysactive_time is not permitted in suspend callback function, it is ineffective, while permitted in resume call back function.

Parameter	Type	Introduction
<timeout>	uint32_t	System cannot sleep before timeout. Unit is ms.

15.2.6 pmu_set_max_sleep_time

Set system max sleep time.

NOTE

System may be waked by other wake event before this timer is timeout.
Before entering deepsleep mode, disable the AON timer by SOCPS_AONTimerCmd(DISABLE)
This setting only works once, the timer will be cleared after system wakeup

Parameter	Type	Introduction
<timeout>	uint32_t	System max sleep timeout. Unit is ms.

15.3 Power Consumption Measurement

15.3.1 Power Consumption Summary

The power consumption values below are tested with the AmebaD QFN68 board, and the image version is 6.2b.

Table 15-10 Power consumption of different power mode

Operation Mode		Condition	Current		Unit
Power Mode	Scenario		3.3V	1.8V	
Deepsleep	Deepsleep	RTC timer 1KB retention RAM	7~8	7-8	uA
	Deepsleep with Key-Scan	RTC timer 1KB retention RAM Key-Scan	12~13	12~13	uA
	Deepsleep with Cap-Touch (average current)	RTC timer 1KB retention RAM Cap-Touch	20	16	uA
Sleep	WoWLAN sleep power	KM4 power gating KM0 clock gating All RAM retained Wi-Fi retained	30~50	30~50	uA
LPS	Wi-Fi connected (idle)	KM0, KM4 in active mode Wi-Fi set LPS mode	20	30	mA
Active	Wi-Fi Rx Idle	HT20 MCS0~7 normal mode KM4 in active mode Rx idle	52	81	mA
WoWLAN	WoWLAN Rx Beacon	Rx beacon mode @ normal mode KM4 in sleep mode	28	45	mA
	WoWLAN DTIM=1 (Average)	KM4 in sleep mode All SRAM retained Wi-Fi retained Shielding room	700~800	1100~1200	uA
		KM4 in sleep mode All SRAM retained Wi-Fi retained Open space	1~2	1.1~2	mA

Table 15-11 WoWLAN power with different DTIM

Condition	DTIM	Current		Unit
		3.3V	1.8V	
KM4 in sleep mode All SRAM retained Wi-Fi retained	1	942	1360	uA
	2	473	671	
	3	316	486	

Shielding room	5	184	299	
	10	92	152	
Test condition: AP: WS 5100 in 802.11g mode				

Table 5-12 Wi-Fi Tx power consumption of 2G/5G

Channel	Power Supply	Operation Mode	Current		Unit
			2.4G	5G	
2442MHz@2.4G 5500MHz@5G 20M 5510MHz@5G 40M	1.8V	1T-MCS7/BW40M (9dBm@2.4G, 8dBm@5G)	181	216	mA
		1T-MCS7/BW40M (12dBm@2.4G, 11dBm@5G)	195	237	
		1T-MCS7/BW20M (9dBm@2.4G, 8dBm@5G)	180	214	
		1T-MCS7/BW20M (12dBm@2.4G, 11dBm@5G)	193	234	
		1T-Legacy_OFDM54M (10dBm@2.4G, 9dBm@5G)	184	219	
		1T-Legacy_OFDM54M (13dBm@2.4G, 12dBm@5G)	214	245	
		1T_CCK11M (21dBm)	203	-	
		1T_CCK11M (16dBm)	224	-	
		1R-Idle/BW40	89	90	
		1R-MCS7/BW40M (Pin=-60dBm)	98	103	
		1R-MCS7/BW20M (Pin=-60dBm)	102	106	
		1R-Legacy_OFDM54M (Pin=-60dBm)	98	103	
		1R-CCK11M (Pin=-60dBm)	81	-	
		RF Standby	58	55	
		RF Disable	43	42	
	3.3V	1T-MCS7/BW40M (15dBm)	206	286	
		1T-MCS7/BW40M (18dBm@2.4G, 17dBm@5G)	247	310	
		1T-MCS7/BW20M (15dBm)	204	286	
		1T-MCS7/BW20M (18dBm)	248	308	
		1T-Legacy_OFDM54M (16dBm)	214	296	
		1T-Legacy_OFDM54M (19dBm@2.4 18dBm@5G)	262	323	
		1T_CCK11M (21dBm)	257	-	
		1T_CCK11M (21dBm)	312	-	
		1R-Idle/BW40	52	53	
		1R-MCS7/BW40M (Pin=-60dBm)	61	64	
		1R-MCS7/BW20M (Pin=-60dBm)	62	63	
		1R-Legacy_OFDM54M (Pin=-60dBm)	61	62	
		1R-CCK11M (Pin=-60dBm)	52	-	
		RF Standby	24	23	
		RF Disable	24	23	

NOTE

The values above are tested with the Amb QFN68 board.
Tx: Continue TX OFDM packets
Rx: Set Rx packets idle interval as short as possible
Load W11N_11M0 Idle Waveform (8us idle interval) for 1R-CCK11M test.

Table 5-13 BT power consumption of different BT mode

Item	Condition	Power Supply	BT Mode	Current	Unit
BT Tx	KM0 ON KM4 ON	3.3V	BT MP	100 (4.45dBm)	mA
				111 (8.15dBm)	
				129 (12dBm)	
BT Rx			BT Central Mode	56.1	
BT ADV			BT Peripheral Mode	56.2	

BT Connection			BT Central Mode	56.3	

15.3.2 Test Command

We provide command in KM4 for tickless testing. Below is the description.

Items	Comment
tickps r debug	Release OS wakelock to open tickles function, KM4 will enter sleep mode when idle. In program you can use pmu_release_wakelock(PMU_OS) to open tickless function.
tickpsa	Get OS wakelock to close tickless function.
tickpsget	Get current wakelock Φ L ű LJNJ ƛ ƴ ű ă ŎdzNJNJ Ŝ ƴ ű ø l· ļ Ŝ f ƨ Ŏ ļ ō ƛ ű Ƴ l· LJ Φ ¸ ƨ enter sleep

15.3.3 Hardware Preparation

In AmebaD reference board, there are other components that consume power. For example, DAP, LEDs, FT232, and capacitances. To measure power consumptions only for AmebaD, you need to remove resistance at R43 (Fig. 15-25).

You can use micro-USB to supply power for the board, and link current meter use J38 (Fig. 15-26).

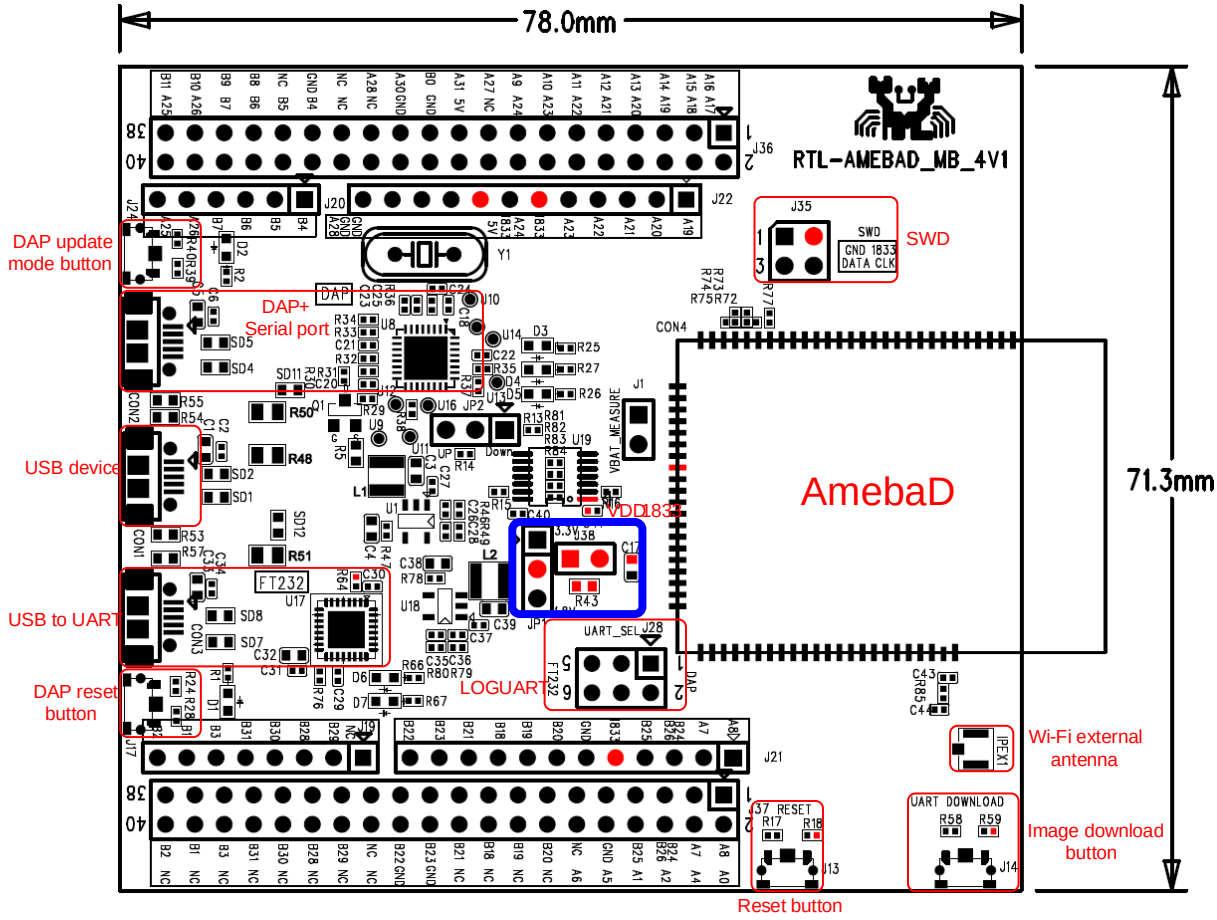


Fig. 15-25 Power consumption measurement

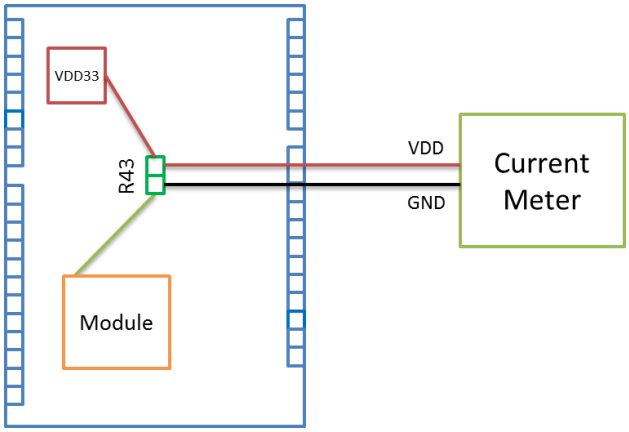


Fig15-26 Measuring power consumption from micro USB

16 Hardware Crypto Engine

16.1 Introduction

Hardware Crypto Engine is used to authenticate, encrypt and decrypt packets. It support the following Hash and Cipher functions:

Hash (include HMAC): MD5/SHA1/SHA2 (224/256), Poly1305

Cipher: AES (ECB/CBC/CFB/OFB/CTR/GMAC/GHASH/GCM), 3DES (ECB/CBC/CFB/OFB/CTR), DES(ECB/CBC/CFB/OFB/CTR), ChaCha20_poly1305

Hardware Crypto Engine also support sequential hash for long plaintext length.

The following sections illustrate the Hardware Crypto Engine APIs and how to use these APIs.

16.2 Hardware Crypto Engine APIs

16.2.1 Crypto Engine Initialization API

16.2.1.1 rtl_cryptoEngine_init

Items	Description
Introduction	Hardware crypto engine initialization
Parameters	N/A
Return	Status value: SUCCESS: Initialization is ok.

16.2.2 Hash APIs

16.2.2.1 rtl_crypto_md5

Items	Description
Introduction	MD5 calculation digest
Parameters	message: Plaintext msglen: Plaintext length pDigest: Result of MD5 function
Return	Status value: SUCCESS: MD5 is ok. Others: fail, refer to ERRNO

16.2.2.2 rtl_crypto_md5_init

Items	Description
Introduction	MD5 initialization (used for sequential hash)
Parameters	N/A
Return	Status value: SUCCESS: MD5 initialization is ok. Others: fail, refer to ERRNO

16.2.2.3 rtl_crypto_md5_update

Items	Description
Introduction	MD5 block calculation (used for sequential hash)
Parameters	message: Plaintext msglen: Plaintext length
Return	Status value: SUCCESS: MD5 update is ok. Others: fail, refer to ERRNO

16.2.2.4 rtl_crypto_md5_final

Items	Description
Introduction	MD5 last block calculation (used for sequential hash)
Parameters	pDigest: Result of MD5 function
Return	Status value: SUCCESS: MD5 final is ok. Others: fail, refer to ERRNO

16.2.2.5 rtl_crypto_sha1

Items	Description
Introduction	SHA1 calculation digest
Parameters	message: Plaintext msglen: Plaintext length pDigest: Result of SHA1 function
Return	Status value: SUCCESS: SHA1 is ok. Others: fail, refer to ERRNO

16.2.2.6 rtl_crypto_sha1_init

Items	Description
Introduction	SHA1 initialization (used for sequential hash)
Parameters	N/A
Return	Status value: SUCCESS: SHA1 initialization is ok. Others: fail, refer to ERRNO

16.2.2.7 rtl_crypto_sha1_update

Items	Description
Introduction	SHA1 block calculation (used for sequential hash)
Parameters	message: Plaintext msglen: Plaintext length
Return	Status value: SUCCESS: SHA1 update is ok. Others: fail, refer to ERRNO

16.2.2.8 rtl_crypto_sha1_final

Items	Description

Introduction	SHA1 last block calculation (used for sequential hash)
Parameters	pDigest: Result of SHA1 function
Return	Status value: SUCCESS: SHA1 final is ok. Others: fail, refer to ERRNO

16.2.2.9 rtl_crypto_sha2

Items	Description
Introduction	SHA2 calculation digest
Parameters	sha2type: SHA2 type, SHA2_224 or SHA2_256 message: Plaintext msglen: Plaintext length pDigest: Result of SHA2 function
Return	Status value: SUCCESS: SHA2 is ok Others: fail, refer to ERRNO

16.2.2.10 rtl_crypto_sha2_init

Items	Description
Introduction	SHA2 initialization (used for sequential hash)
Parameters	sha2type: SHA2 type, SHA2_224 or SHA2_256
Return	Status value: SUCCESS: SHA2 initialization is ok. Others: fail, refer to ERRNO

16.2.2.11 rtl_crypto_sha2_update

Items	Description
Introduction	SHA2 block calculation (used for sequential hash)
Parameters	message: Plaintext msglen: Plaintext length
Return	Status value: SUCCESS: SHA2 update is ok. Others: fail, refer to ERRNO

16.2.2.12 rtl_crypto_sha2_final

Items	Description
Introduction	SHA2 last block calculation (used for sequential hash)
Parameters	pDigest: Result of SHA2 function
Return	Status value: SUCCESS: SHA2 final is ok. Others: fail, refer to ERRNO

16.2.2.13 rtl_crypto_hmac_md5

Items	Description
Introduction	HMAC MD5 calculation digest
Parameters	key: HMAC MD5 key, need to be 4 byte aligned keylen: Key length message: Plaintext

	msglen: Plaintext length pDigest: Result of HMAC-MD5 function
Return	Status value: SUCCESS: HMAC-MD5 is ok. Others: fail, refer to ERRNO

16.2.2.14 rtl_crypto_hamc_md5_init

Items	Description
Introduction	HMAC-MD5 initialization (used for sequential hash)
Parameters	key: HMAC-MD5 key, need to be 4-byte aligned keylen: Key length
Return	Status value: SUCCESS: HMAC-MD5 initialization ok Others: fail, refer to ERRNO

16.2.2.15 rtl_crypto_hmac_md5_update

Items	Description
Introduction	HMAC-MD5 block calculation (used for sequential hash)
Parameters	message: Plaintext msglen: Plaintext length
Return	retval status value: SUCCESS: HMAC-MD5 update is ok. Others: fail, refer to ERRNO

16.2.2.16 rtl_crypto_hmac_md5_final

Items	Description
Introduction	HMAC-MD5 last block calculation (used for sequential hash)
Parameters	pDigest: Result of HMAC-MD5 function
Return	Status value: SUCCESS: HMAC-MD5 final is ok. Others: fail, refer to ERRNO

16.2.2.17 rtl_crypto_hmac_sha1

Items	Description
Introduction	HMAC-SHA1 calculation digest
Parameters	key: HMAC-SHA1 key, need to be 4-byte aligned keylen: key length message: Plaintext msglen: Plaintext length pDigest: Result of HMAC-SHA1 function
Return	Status value: SUCCESS: HMAC-SHA1 is ok. Others: fail, refer to ERRNO

16.2.2.18 rtl_crypto_hmac_sha1_init

Items	Description
Introduction	HMAC-SHA1 initialization (used for sequential hash)
Parameters	key: HMAC-SHA1 key, need to be 4-byte aligned

	keylen: key length
Return	Status value: SUCCESS: HMAC-SHA1 initialization is ok. Others: fail, refer to ERRNO

16.2.2.19 rtl_crypto_hmac_sha1_update

Items	Description
Introduction	HMAC-SHA1 block calculation (used for sequential hash)
Parameters	message: Plaintext msglen: Plaintext length
Return	Status value: SUCCESS: HMAC-SHA1 update is ok. Others: fail, refer to ERRNO

16.2.2.20 rtl_crypto_hmac_sha1_final

Items	Description
Introduction	HMAC-SHA1 last block calculation (used for sequential hash)
Parameters	pDigest: Result of HMAC-SHA1 function
Return	Status value: SUCCESS: HMAC-SHA1 final is ok. Others: fail, refer to ERRNO

16.2.2.21 rtl_crypto_hmac_sha2

Items	Description
Introduction	HMAC-SHA2 calculation digest
Parameters	sha2type: SHA2 type, SHA2_224 or SHA2_256 key: HMAC-SHA2 key, need to be 4-byte aligned keylen: Key length message: Plaintext msglen: Plaintext length pDigest: Result of HMAC-SHA2 function
Return	Status value: SUCCESS: HMAC-SHA2 is ok. Others: fail, refer to ERRNO

16.2.2.22 rtl_crypto_hmac_sha2_init

Items	Description
Introduction	HMAC-SHA2 initialization (used for sequential hash)
Parameters	sha2type: SHA2 type, SHA2_224 or SHA2_256 key: HMAC-SHA2 key, need to be 4-byte aligned keylen: Key length
Return	Status value: SUCCESS: HMAC-SHA2 initialization is ok. Others: fail, refer to ERRNO

16.2.2.23 rtl_crypto_hmac_sha2_update

Items	Description
Introduction	HMAC-SHA2 block calculation (used for sequential hash)

Parameters	message: Plaintext msglen: Plaintext length
Return	Status value: SUCCESS: HMAC-SHA2 update is ok. Others: fail, refer to ERRNO

16.2.2.24 rtl_crypto_hmac_sha2_final

Items	Description
Introduction	HMAC-SHA2 last block calculation (used for sequential hash)
Parameters	pDigest: Result of HMAC-SHA2 function
Return	Status value: SUCCESS: HMAC-SHA2 final is ok. Others: fail, refer to ERRNO

16.2.2.25 rtl_crypto_poly1305

Items	Description
Introduction	poly1305 calculation digest
Parameters	key: poly1305 key need to be 4-byte aligned keylen: Key length message: Plaintext msglen: Plaintext length pDigest: Result of poly1305 function
Return	Status value: SUCCESS: poly1305 is ok. Others: fail, refer to ERRNO

16.2.3 Cipher APIs

16.2.3.1 rtl_crypto_xxx_xxx_init

Items	Description
Introduction	xxx initialization xxx can be aes_cbc/aes_ecb/aes_ctr/aes_cfb/aes_ofb/aes_gcm/3des_cbc/3des_ecb/3des_ctr/3des_cfb/3des_ofb/des_cbc/des_ecb/des_ctr/des_cfb/des_ofb
Parameters	key: Cipher key keylen: Key length
Return	status value: SUCCESS: Initialization is ok. Others: fail, refer to ERRNO

16.2.3.2 rtl_crypto_xxx_xxx_encrypt

Items	Description
Introduction	xxx encryption xxx can be aes_cbc/aes_ecb/aes_ctr/aes_cfb/aes_ofb/aes_gcm/3des_cbc/3des_ecb/3des_ctr/3des_cfb/3des_ofb/des_cbc/des_ecb/des_ctr/des_cfb/des_ofb
Parameters	message: Point to source message msglen: Message length iv: Initial vector ivlen: iv length pResult: Point to cipher result

Return	Status value: SUCCESS: Encryption is ok. Others: fail, refer to ERRNO

16.2.3.3 rtl_crypto_xxx_xxx_decrypt

Items	Description
Introduction	xxx decryption xxx can be aes_cbc/aes_ecb/aes_ctr/aes_cfb/aes_ofb/aes_gcm/3des_cbc/3des_ecb/3des_ctr/3des_cfb/3des_ofb/des_cbc/des_ecb/des_ctr/des_cfb/des_ofb
Parameters	message: Point to source message msglen: Message length iv: Initial vector ivlen: IV (initialization vector) length pResult: Point to cipher result
Return	Status value: SUCCESS: Decryption is ok. Others: fail, refer to ERRNO

16.2.3.4 rtl_crypto_aes_gcm_init

Items	Description
Introduction	AES GCM initialization
Parameters	key: Cipher key keylen: Key length
Return	Status value: SUCCESS: Initialization is ok. Others: fail, refer to ERRNO

16.2.3.5 rtl_crypto_aes_gcm_encrypt

Items	Description
Introduction	AES GCM encryption
Parameters	message: Point to source message msglen: Message length iv: Initial vector aad: Point to AAD (additional authentication data) aadlen: AAD length pResult: Point to cipher result pTag: Point to MAC (Message Authentication Code)
Return	Status value: SUCCESS: Encryption is ok. Others: fail, refer to ERRNO

16.2.3.6 rtl_crypto_aes_gcm_decrypt

Items	Description
Introduction	AES GCM decryption
Parameters	message: Point to source message msglen: Message length iv: Initial vector aad: Point to AAD (additional authentication data) aadlen: AAD length pResult: Point to cipher result

	pTag: Point to MAC (Message Authentication Code)
Return	Status value: SUCCESS: Decryption is ok. Others: fail, refer to ERRNO

16.2.3.7 rtl_crypto_chacha_init

Items	Description
Introduction	ChaCha20 initialization
Parameters	key: Cipher key
Return	Status value: SUCCESS: Initialization OK Others: fail, refer to ERRNO

16.2.3.8 rtl_crypto_chacha_encrypt

Items	Description
Introduction	ChaCha20 encryption
Parameters	message: Point to source message msglen: Message length iv: Point to IV (initialization vector) count: Counter value pResult: Point to cipher result
Return	Status value: SUCCESS: Encryption is ok. Others: fail, refer to ERRNO

16.2.3.9 rtl_crypto_chacha_decrypt

Items	Description
Introduction	ChaCha20 decryption
Parameters	message: Point to source message msglen: Message length iv: Point to IV (initialization vector) count: Counter value pResult: Point to cipher result
Return	Status value: SUCCESS: Decryption is ok. Others: fail, refer to ERRNO

16.2.3.10 rtl_crypto_chacha_poly1305_init

Items	Description
Introduction	ChaCha Poly1305 initialization
Parameters	key: Cipher key
Return	Status value: SUCCESS: Initialization is ok. Others: fail, refer to ERRNO

16.2.3.11 rtl_crypto_chacha_poly1305_encrypt

Items	Description
Introduction	ChaCha Poly1305 encryption

Parameters	message: Point to source message msglen: Message length nonce: Random value aad: Point to AAD (additional authentication data) aadlen: AAD length pResult: Point to cipher result pTag: Point to MAC (Message Authentication Code)
Return	Status value: SUCCESS: encryption is ok. Others: fail, refer to ERRNO

16.2.3.12 rtl_crypto_chacha_poly1305_decrypt

Items	Description
Introduction	ChaCha Poly1305 decryption
Parameters	message: Point to source message msglen: Message length nonce: Random value aad: Point to AAD (additional authentication data) aadlen: AAD length pResult: Point to cipher result pTag: Point to MAC (Message Authentication Code)
Return	Status value: SUCCESS: decryption is ok. Others: fail, refer to ERRNO

16.3 Hardware Crypto Engine API Usage

16.3.1 Starting Hardware Crypto Engine

Before using hardware crypto engine, you need to initialize it first. You can use Crypto Engine Initialization API to do this.

16.3.2 Starting Crypto Engine Calculation

Choose a hash or cipher algorithm, and call the following APIs to calculate hash digest or ciphertext.

16.3.2.1 Hash Algorithm

If a hash algorithm is selected, then Hash APIs (*rtl_crypto_xxx)* can be used to calculate digest.

Sequential hash is needed when source message length beyond hardware support. Sequential hash breaks a whole long message into several piece of message payload, then calculate the payload in sequence, until the last message payload.

When use sequential hash, you can follow the steps below:

(1) Initialize sequential hash: use Hash APIs (*rtl_crypto_xxx_init*) to initialize.
(2) Handle each piece of message payload: call Hash APIs (*rtl_crypto_xxx_update*) once when one piece message payload.
(3) Handle the last piece of message payload: call Hash APIs (*rtl_crypto_xxx_final*) once when one piece message payload.

Note: Considering that the Crypto Engine moves data through DMA and bypasses the D-Cache, if the destination array is placed in stack and the start address is not 32-byte aligned, the cache line would be dirty during function call in some cases. To avoid reading wrong digest back, users should choose one of the following methods:

- The digest array is a global variable.
- Or the start address of digest array should be 32 ȅ ű Ŝ Ŀ ƒ ƛ Ǝ ƴ Ŝ Ř ƛ Ŧ ƛ ű Ω ă Ŀ ƒ ƻ Ŏ Ŀ ƒ Ǿ Ŀ NJ ƛ Ŀ ō ƒ Ŝ Φ
- Or call ƒ ű K Ŝ Ŧ ƻ ƒ ƒ ƻ ǿ ƛ ƴ Ǝ ƒ ƛ ƴ Ŝ ű ƻ NJ Ŝ ă ű ƻ NJ Ŝ Ř Ŀ ű Ŀ Ŧ NJ ƻ Ƴ Ƴ Ŝ Ƴ ƻ NJ ȅ ō Ŝ Ŧ ƻ NJ Ŝ NJ Ŝ


```
DCache_Invalidate(((u32)digest & CACHE_LINE_ADDR_MSK), (sizeof(digest) + CACHE_LINE_SIZE));
```

16.3.2.2 Cipher Algorithm

Steps to encrypt or decrypt message are as following:

(1) Initialize
Use Cipher APIs (*rtl_crypto_xxx_xxx_init*) to initialize.

(2) Encrypt or decrypt message
Call Cipher APIs (*rtl_crypto_xxx_xxx_encrypt*) to encrypt source message.
Call Cipher APIs (*rtl_crypto_xxx_xxx_decrypt*) to decrypt source message.

16.4 Demo Code Path

Hardware Crypto Engine demo code locate folder: project/realtek_amebaD_va0_example/example_sources/CRYPTO

17 User Configuration

Some functions have complex parameters, it is hard to give standard API in SDK. For example, different users have different parameters for BOOT. So we provide some user configuration files which will be called by SDK, and SDK will execute different activities based on different user configurations.

These configuration files should be maintained by users.

17.1 Configuration File List

The user configuration files are listed in Table 17-1.

Table 17-1 User configuration file

Configuration file	Configuration item	Image
rtl8721dhp_boot_trustzonecfg.c	Used to configure KM4 TrustZone non-secure areas.	KM4 bootloader
rtl8721dlp_flashcfg.c	Used to configure Flash settings: Flash_Speed Flash_ReadMode Flash_AVL You can also configure your Flash using flash_init_userdef if your Flash is not in Flash_AVL	KM0 image2
rtl8721dlp_pinmapcfg.c	Used to reduce I/O power leakage Per pin function configuration Per pin function pull up & pull down Per pin sleep pull up & pull down	KM0 image2
rtl8721dlp_sleepcfg.c	Sleep/deepsleep power management Sleep/deepsleep wakeup event Sleep/deepsleep wake pin	KM0 image2
rtl8721d_bootcfg.c	Used to configure bootloader: Flash_MMU_Config RSIP_Mask_Config Force_OTA1_GPIO Boot_Log_En	KM0 & KM4 bootloader
rtl8721d_ipccfg.c	Used to define your IPC channels	KM0 & KM4 image2
rtl8721dlp_intfcfgc	Enable/disable low power Rx	KM0 image2
rtl8721dhp_intfcfg.c	PSRAM configuration: Enable PSRAM Enable PSRAM calibration function Enable PSRAM retention SDIO host configuration SDIO Host bus speed SDIO Host bus width Card Detect pin Write Protection pin FTL parameter configuration: The number of physical map pages The offset of Flash sectors which is allocated to FTL physical map	KM4 image2

17.2 Boot TrustZone Configuration

The whole address space will be secure when boot up. SAU is used to configure secure or non-secure area for some special address space. SAU should be configured in secure bootloader, and it cannot be changed again after setting, it is protected by hardware.

There are 8 SAU entries in AmebaD, entry0 to entry4 has been used by system, and entry5 to entry7 are left for user. You can configure your own TrustZone non-secure area.

```
BOOT_RAM_DATA_SECTION
const TZ_CFG_TypeDef tz_config[]=
{
//  Start                   End                     NSC
    {0x40000000,            0x50000000-1,           0}, /* entry0: Peripherals NS */
    {0x1010A000,            0x101D4000-1,           0}, /* entry1: IROM & DROM NS */
    {0x00080000,            __ram_image3_start__ -1,0}, /* entry2: KM0 SRAM, Retention SRAM, PSRAM & FLASH & SRAM NS */
    {__ram_image3_end__,    0x1007C000-1,           1}, /* entry3: NSC 4K */
    {0x100E0000,            0x10100000-1,           0}, /* entry4: BT/WIFI Extention SRAM */
    {0xFFFFFFFF,            0xFFFFFFFF,             0}, /* entry5: TODO */
    {0xFFFFFFFF,            0xFFFFFFFF,             0}, /* entry6: TODO */
    {0xFFFFFFFF,            0xFFFFFFFF,             0}, /* entry7: TODO */
    {0xFFFFFFFF,            0xFFFFFFFF,             0},
};
```

17.3 Flash Configuration

The Flash configuration items for users are listed in Table 17-2.

Table 17-2 User Flash configuration

Item	Comment	Default
Flash_Speed	Indicates the Flash baud rate. It can be one of the following value: 0xFFFF: 80MHz 0x7FFF: 100MHz 0x3FFF: 67MHz 0x1FFF: 57MHz 0x0FFF: 50MHz	0xFFFF
Flash_ReadMode	Indicates the Flash read I/O mode. It can be one of the following value: 0xFFFF: Read quad I/O, Address & Data 4 bits mode 0x7FFF: Read quad O, just data 4 bits mode 0x3FFF: Read dual I/O, Address & Data 2 bits mode 0x1FFF: Read dual O, just data 2 bits mode 0x0FFF: 1 bit mode If the configured read mode is not supported, other modes would be searched until find the appropriate mode.	0xFFFF
Flash_AVL	Maintains the Flash IC supported by SDK, refer to Fig 17-1 for detailed information.	
flash_init_userdef	Initializes the parameters in this function in order to adopt user-defined Flash on AmebaD.	

17.3.1 Flash Classification

AmebaD support Flash chips of multi-vendor, such Winbond, MXIC, Gigadevice, ESMT etc. The Flash chips which have verified on Ameba platform can be found in *UM0400 Ameba Flash Appdf*. We divide the supported Flash into 6 species as Table 17-3 shows according to their characteristics, and they can be used on AmebaD directly.

Table 17-3 Flash species

Flash classification	Manufacture	Flash ID
FlashClass1	Winbond	0xEF
	FM	0xA1
	XTX	0x0B
	XTX(FT)	0x0E
FlashClass2	GD(GD normal)	0xC8
FlashClass3	MXIC(MXIC normal)	0xC2
	Hua Hong	0x68
	GD(GD MD serial)	0x51
FlashClass4	ESMT	0x1C
FlashClass5	Micron	0x20
FlashClass6	MXIC(MXIC wide-range VCC)	0x28C2

For a new Flash chip that is not available in AVL, users should follow these steps to check if it can be supported by AmebaD.

(1) Review Flash datasheet to collect essential information
   Flash ID
   Number of Flash status register and the definition (QE, BUSY, WEL, BP etc)
   Flash Commands (Write/Read status register, Read, Read Dual/Quad Output/IO, Page Program, Sector/Block/Chip Erase, Enter/Release from Power-down etc)
   Dummy cycle number of Read Dual/Quad Output/IO mode

(2) Check if these parameters match with those of the existing species
   a) If the Flash belongs to one of the Realtek defined classes, it is already supported by SDK and you can use it directly
   b) If all the parameters except Flash ID match with those of one species, user should add the Flash ID into Flash_AVL table in SDK before using it, as Fig17-1 shows.
   c) If both the Flash parameters and Flash ID mismatch with the existing species, user should define the parameters in function flash_init_userdef.

```
const FlashInfo_TypeDef Flash_AVL[] = {
        /*flash_id,     flash_id_mask,  flash_class,        status_mask,    FlashInitHandler */

        /* case1: Realtek defined class, any modification is not allowed */
        {0xEF,          0x000000FF,     FlashClass1,        0x000043FC,     NULL},  /* Winbond: MANUFACTURER_ID_WINBOND */
        {0xA1,          0x000000FF,     FlashClass1,        0x0000FFFC,     NULL},  /* Fudan Micro: MANUFACTURER_ID_FM */
        {0x0B,          0x000000FF,     FlashClass1,        0x000043FC,     NULL},  /* XTX */
        {0x0E,          0x000000FF,     FlashClass1,        0x000043FC,     NULL},  /* XTX(FT) */
        {0xC8,          0x000000FF,     FlashClass2,        0x000043FC,     NULL},  /* GD normal: MANUFACTURER_ID_GD */
        {0x28C2,        0x0000FFFF,     FlashClass6,        0x000200FC,     NULL},  /* MXIC wide-range VCC: MANUFACTURER_ID_MXIC */
        {0xC2,          0x000000FF,     FlashClass3,        0x000000FC,     NULL},  /* MXIC normal: MANUFACTURER_ID_BOHONG */
        {0x68,          0x000000FF,     FlashClass3,        0x000000FC,     NULL},  /* Hua Hong */
        {0x51,          0x000000FF,     FlashClass3,        0x000000FC,     NULL},  /* GD MD serial */
        {0x1C,          0x000000FF,     FlashClass4,        0x000000FC,     NULL},  /* ESMT: MANUFACTURER_ID_EON */
        {0x20,          0x000000FF,     FlashClass5,        0x000000FC,     NULL},  /* Microm: MANUFACTURER_ID_MICRON */

        /* case2: new flash, ID is not included in case1 list, but specification is compatible with FlashClass1~FlashClass6 */
        //{0xXX,            0x0000XXXX,     FlashClassX,        0x0000XXXX,     NULL},

        /* case3: new flash, ID is not included in case1 list, and specification is not compatible with FlashClass1~FlashClass6 */
        {0x00,          0x000000FF,     FlashClassUser,     0xFFFFFFFF,     &flash_init_userdef},

        /* End */
        {0xFF,          0xFFFFFFFF,     FlashClassNone,     0xFFFFFFFF,     NULL},
};
```

Fig17-1 Flash_AVL

17.3.1.1 Dummy Cycles

When reading Flash, host sends command and address to Flash. After that host needs to send dummy cycles to Flash before it sends data back. The number of dummy cycles is various for different read modes.

The following section shows you how to find dummy cycles from Flash datasheet.

Taking Winbond W25Q16JV for an example
   There are 6 dummy cycles in Read Quad I/O mode, as Fig17-2 shows
   There are 4 dummy cycles in Read Dual I/O mode, as Fig17-3 shows.

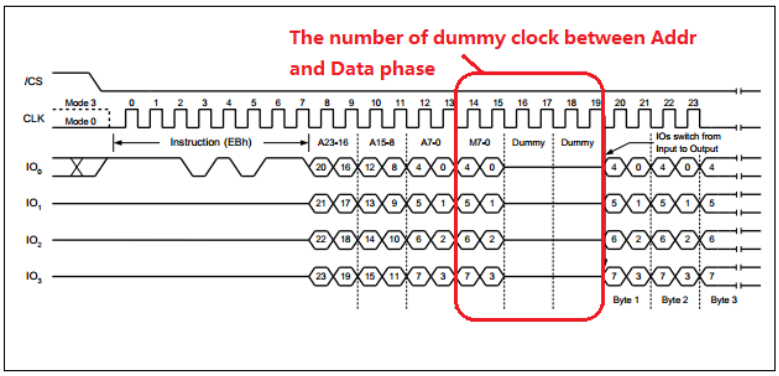


Fig17-2 Fast read quad I/O instruction sequence

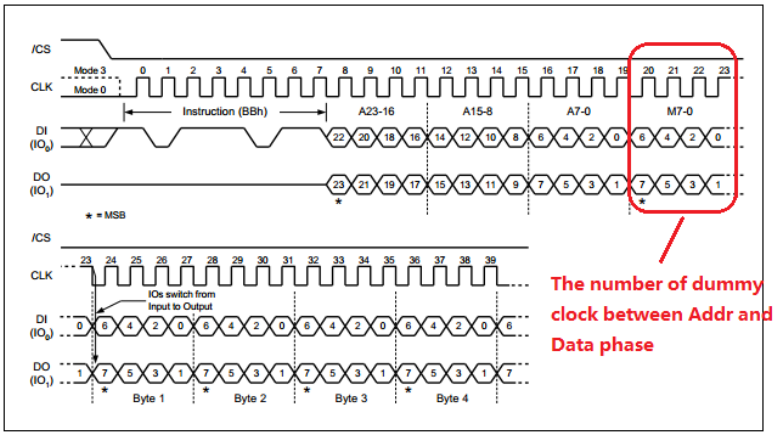


Fig17-3 Fast read dual I/O instruction sequence

Users can find the dummy cycle of Read Quad Output and Read Dual Output mode in the same way.

17.3.1.2 Other Parameters

The parameters comparison of these species are listed in Table 17-4.

Table 17-4 Parameters comparison of different Flash species

Item	Parameter	Flash Class 1	Flash Class 2	Flash Class 3	Flash Class 4	Flash Class 5	Flash Class 6
Status Register	Status Register Number	2	2 or 3	1	1	1	1
	QE bit	Bit[9]	Bit[9]	Bit[6]	No exist	No exist	Bit[6]
	BUSY bit	Bit[0]					
	WEL bit	Bit[1]					
Dummy Cycle	READ	0	0	0	0	0	0
	DREAD (1I/2O Read)	8	8	8	8	3	8
	2READ (2*I/O Read)	4	4	4	4	5	4
	QREAD (1I/4O Read)	8	8	8	8	5	8
	4READ (4*I/O Read)	6	6	6	6	9	6
Command	Write Enable	0x06	0x06	0x06	0x06	0x06	0x06
	Read JEDEC ID	0x9F	0x9F	0x9F	0x9F	0x9F	0x9F
	Read Status Register-1	0x05	0x05	0x05	0x05	0x05	0x05
	Read Status Register-2	0x35	0x35	-	-	-	-
	Write Status Register-1	0x01	0x01	0x01	0x01	0x01	0x01
	Write Status Register-2[1]	-	-/0x31	-	-	-	-
	Chip Erase	0x60	0x60	0x60	0x60	0xC7	0x60
	Block Erase	0xD8	0xD8	0xD8	0xD8	0xD8	0xD8
	Sector Erase	0x20	0x20	0x20	0x20	0x20	0x20
	Power-down	0xB9	0xB9	0xB9	0xB9	0xB9	0xB9
	Release from Power-down	0xAB	0xAB	0xAB	0xAB	0xAB	0xAB
	READ	0x03	0x03	0x03	0x03	0x03	0x03
	DREAD (1I/2O Read)	0x3B	0x3B	0x3B	0x3B	0x3B	0x3B
	2READ (2*I/O Read)	0xBB	0xBB	0xBB	0xBB	0xBB	0xBB
	QREAD (1I/4O Read)	0x6B	0x6B	0x6B	0x6B	0x6B	0x6B
	4READ (4*I/O Read)	0xEB	0xEB	0xEB	0xEB	0xEB	0xEB
	Read Configuration Register[2]	-	-	-	-	-	0x15
	Write Configuration Register[2]	-	-	-	-	-	-

Note 1:

For Flash Class-1, the Write Status Register-2 is written together with the Write Status Register-1 using 0x01, as Fig17-4 shows

For FlashClass2, when GD density is less than 2MB, the Write Status Register 2 is written together with the Write Status Register 1 using 0x01, as Fig17-4 shows. When GD density is larger than 2MB, Write Status Register 2 is written individually using 0x31, as Fig17-5 shows.

Note 2: For MXIC wide-range VCC, the Read Configuration Register would be checked whether the chip is in high performance mode by default once power-on, as Fig17-6 shows. If not, we would switch it to high performance mode.

Note 3: For MXIC wide-range VCC, the Write Configuration Register is written together with Write Status Register using 0x01, as Fig17-7 shows.

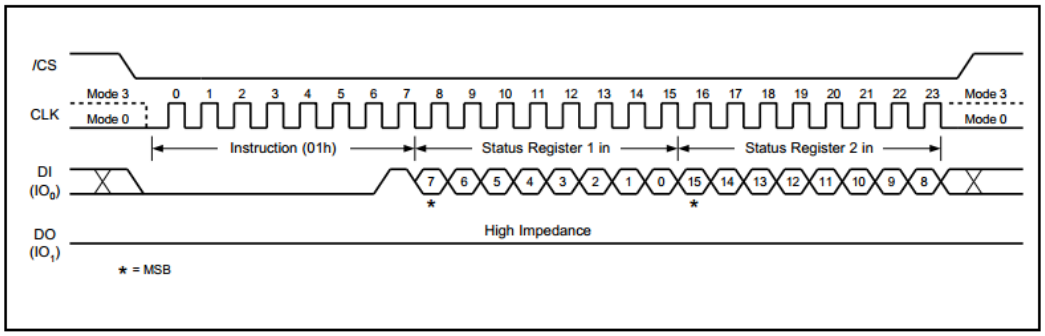


Fig17-4 Write Status Register 1 & 2 sequence (FlashClass1, FlashClass2 when density < 2M)

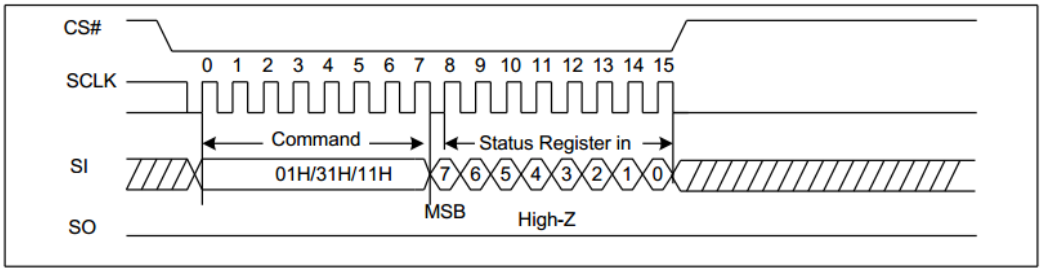


Fig17-5 Write Status Register 1/2 sequence (FlashClass2 when density > 2MB)

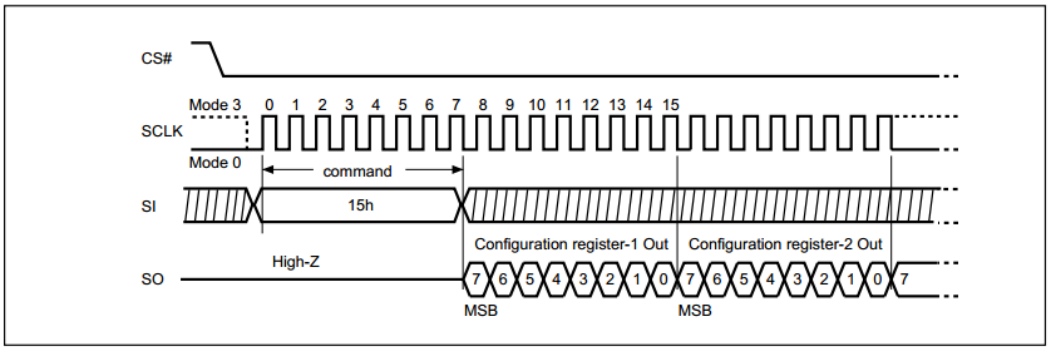


Fig17-6 Read Configuration Register sequence (FlashClass6)

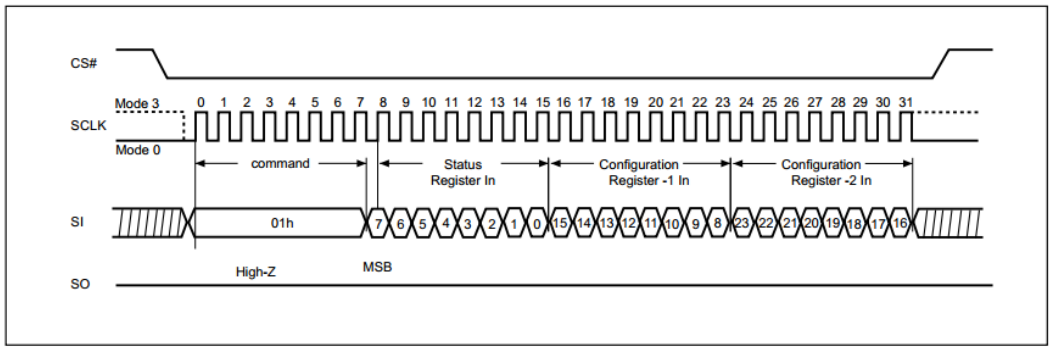


Fig17-7 Write Status Register and Write Configuration Register sequence (FlashClass6)

17.3.2 FlashClass1~FlashClass6

Table17-5 lists the parameters for FlashClass1 to FlashClass6 which cannot be modified by users.

Table17-5 Parameters of different Flash classification (FlashClass1~ FlashClass6)

Flash classification	Command list	Note

FlashClass1	FLASH_InitStruct.FLASH_Id = FLASH_ID_WINBOND;
	
	/*1.1 status bit define */
	FLASH_InitStruct.FLASH_QuadEn_bit = BIT(9);
	FLASH_InitStruct.FLASH_Busy_bit = BIT(0);
	FLASH_InitStruct.FLASH_WLE_bit = BIT(1);
	FLASH_InitStruct.FLASH_Status2_exist = 1;
	
	/*1.2 dummy cycle */
	FLASH_InitStruct.FLASH_rd_dummy_cyle[0] = 0;
	FLASH_InitStruct.FLASH_rd_dummy_cyle[1] = 0x04;
	FLASH_InitStruct.FLASH_rd_dummy_cyle[2] = 0x06;
	
	/*1.3 set 2 bits mode command */
	FLASH_InitStruct.FLASH_rd_dual_io = 0xBB;
	FLASH_InitStruct.FLASH_rd_dual_o = 0x3B;
	
	/*1.4 set 4 bits mode command */
	FLASH_InitStruct.FLASH_rd_quad_io = 0xEB;
	FLASH_InitStruct.FLASH_rd_quad_o = 0x6B;
	
	/*1.5 other Flash command set */
	FLASH_InitStruct.FLASH_cmd_wr_en = 0x06;
	FLASH_InitStruct.FLASH_cmd_rd_id = 0x9F;
	FLASH_InitStruct.FLASH_cmd_rd_status = 0x05;
	FLASH_InitStruct.FLASH_cmd_rd_status2 = 0x35; //FLASH_CMD_RDSR2
	FLASH_InitStruct.FLASH_cmd_wr_status = 0x01;
	FLASH_InitStruct.FLASH_cmd_wr_status2 = 0;
	FLASH_InitStruct.FLASH_cmd_chip_e = 0x60;
	FLASH_InitStruct.FLASH_cmd_block_e = 0xD8;
	FLASH_InitStruct.FLASH_cmd_sector_e = 0x20;
	FLASH_InitStruct.FLASH_cmd_pwdn_release = 0xAB;
	FLASH_InitStruct.FLASH_cmd_pwdn = 0xB9;	

FlashClass2	FLASH_InitStruct.FLASH_Id = FLASH_ID_GD; /*1.1 status bit define */	FLASH_cmd_wr_status2 would be set when density is larger than 2MB in SDK.

	FLASH_InitStruct.FLASH_QuadEn_bit = BIT(9);
	FLASH_InitStruct.FLASH_Busy_bit = BIT(0); //WIP
	FLASH_InitStruct.FLASH_WLE_bit = BIT(1); //WLE
	FLASH_InitStruct.FLASH_Status2_exist = 1;
	
	/*1.2 dummy cycle */
	FLASH_InitStruct.FLASH_rd_dummy_cyle[0] = 0;
	FLASH_InitStruct.FLASH_rd_dummy_cyle[1] = 0x04; //(M7:4)2 + 2 dummy
	FLASH_InitStruct.FLASH_rd_dummy_cyle[2] = 0x06; //(M7:0)2 + 4 dummy
	
	/*1.3 set 2 bits mode command */
	FLASH_InitStruct.FLASH_rd_dual_io = 0xBB;
	FLASH_InitStruct.FLASH_rd_dual_o = 0x3B;
	
	/*1.4 set 4 bits mode command */
	FLASH_InitStruct.FLASH_rd_quad_io = 0xEB;
	FLASH_InitStruct.FLASH_rd_quad_o = 0x6B;
	
	/*1.5 other Flash command set */
	FLASH_InitStruct.FLASH_cmd_wr_en = 0x06;
	FLASH_InitStruct.FLASH_cmd_rd_id = 0x9F;
	FLASH_InitStruct.FLASH_cmd_rd_status = 0x05;
	FLASH_InitStruct.FLASH_cmd_rd_status2 = 0x35;
	FLASH_InitStruct.FLASH_cmd_wr_status = 0x01;
	FLASH_InitStruct.FLASH_cmd_wr_status2 = 0;
	FLASH_InitStruct.FLASH_cmd_chip_e = 0x60;
	FLASH_InitStruct.FLASH_cmd_block_e = 0xD8;
	FLASH_InitStruct.FLASH_cmd_sector_e = 0x20;
	FLASH_InitStruct.FLASH_cmd_pwdn_release = 0xAB;
	FLASH_InitStruct.FLASH_cmd_pwdn = 0xB9;	

FlashClass8	FLASH_InitStruct.FLASH_Id = FLASH_ID_MXIC;
	
	/*1.1 status bit define */
	FLASH_InitStruct.FLASH_QuadEn_bit = BIT(6);
	FLASH_InitStruct.FLASH_Busy_bit = BIT(0); //WIP
	FLASH_InitStruct.FLASH_WLE_bit = BIT(1); //WLE
	FLASH_InitStruct.FLASH_Status2_exist = 0;
	
	/*1.2 dummy cycle */
	FLASH_InitStruct.FLASH_rd_dummy_cyle[0] = 0;
	FLASH_InitStruct.FLASH_rd_dummy_cyle[1] = 0x04;
	FLASH_InitStruct.FLASH_rd_dummy_cyle[2] = 0x06;
	
	/*1.3 set 2 bits mode command */
	FLASH_InitStruct.FLASH_rd_dual_io = 0xBB;
	FLASH_InitStruct.FLASH_rd_dual_o = 0x3B;
	
	/*1.4 set 4 bits mode command */
	FLASH_InitStruct.FLASH_rd_quad_io = 0xEB;
	FLASH_InitStruct.FLASH_rd_quad_o = 0x6B;
	
	/*1.5 other Flash command set */
	FLASH_InitStruct.FLASH_cmd_wr_en = 0x06;
	FLASH_InitStruct.FLASH_cmd_rd_id = 0x9F;
	FLASH_InitStruct.FLASH_cmd_rd_status = 0x05;
	FLASH_InitStruct.FLASH_cmd_rd_status2 = 0;
	FLASH_InitStruct.FLASH_cmd_wr_status = 0x01;
	FLASH_InitStruct.FLASH_cmd_wr_status2 = 0;	

	FLASH_InitStruct->FLASH_cmd_chip_e = 0x60; FLASH_InitStruct->FLASH_cmd_block_e = 0xD8; FLASH_InitStruct->FLASH_cmd_sector_e = 0x20; FLASH_InitStruct->FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct->FLASH_cmd_pwdn = 0xB9;	

FlashClass4	FLASH_InitStruct->FLASH_Id = FLASH_ID_MXIC;
	
	/*1.1 status bit define */
	FLASH_InitStruct->FLASH_QuadEn_bit = 0;
	FLASH_InitStruct->FLASH_Busy_bit = BIT(0); //WIP
	FLASH_InitStruct->FLASH_WLE_bit = BIT(1); //WLE
	FLASH_InitStruct->FLASH_Status2_exist = 0;
	
	/*1.2 dummy cycle */
	FLASH_InitStruct->FLASH_rd_dummy_cyle[0] = 0;
	FLASH_InitStruct->FLASH_rd_dummy_cyle[1] = 0x04;
	FLASH_InitStruct->FLASH_rd_dummy_cyle[2] = 0x06;
	
	/*1.3 set 2 bits mode command */
	FLASH_InitStruct->FLASH_rd_dual_io = 0xBB;
	FLASH_InitStruct->FLASH_rd_dual_o = 0x3B;
	
	/*1.4 set 4 bits mode command */
	FLASH_InitStruct->FLASH_rd_quad_io = 0xEB;
	FLASH_InitStruct->FLASH_rd_quad_o = 0x6B;
	
	/*1.5 other Flash command set */
	FLASH_InitStruct->FLASH_cmd_wr_en = 0x06;
	FLASH_InitStruct->FLASH_cmd_rd_id = 0x9F;
	FLASH_InitStruct->FLASH_cmd_rd_status = 0x05;
	FLASH_InitStruct->FLASH_cmd_rd_status2 = 0;
	FLASH_InitStruct->FLASH_cmd_wr_status = 0x01;
	FLASH_InitStruct->FLASH_cmd_wr_status2 = 0;
	FLASH_InitStruct->FLASH_cmd_chip_e = 0x60;
	FLASH_InitStruct->FLASH_cmd_block_e = 0xD8;
	FLASH_InitStruct->FLASH_cmd_sector_e = 0x20;
	FLASH_InitStruct->FLASH_cmd_pwdn_release = 0xAB;
	FLASH_InitStruct->FLASH_cmd_pwdn = 0xB9;	

FlashClass5	FLASH_InitStruct->FLASH_Id = FLASH_ID_MICRON; /*1.1 status bit define */ FLASH_InitStruct->FLASH_QuadEn_bit = 0; FLASH_InitStruct->FLASH_Busy_bit = BIT(0); FLASH_InitStruct->FLASH_WLE_bit = BIT(1); FLASH_InitStruct->FLASH_Status2_exist = 0; /*1.2 dummy cycle */ FLASH_InitStruct->FLASH_rd_dummy_cyle[0] = 0; /* set Micron rd_dummy_cyle based on baud rate, default 100MHz */ FLASH_InitStruct->FLASH_rd_dummy_cyle[1] = 0x05; FLASH_InitStruct->FLASH_rd_dummy_cyle[2] = 0x09; /*1.3 set 2 bits mode command */ FLASH_InitStruct->FLASH_rd_dual_io = 0xBB; FLASH_InitStruct->FLASH_rd_dual_o = 0x3B; /*1.4 set 4 bits mode command */ FLASH_InitStruct->FLASH_rd_quad_io = 0xEB;	

	FLASH_InitStruct->FLASH_rd_quad_o = 0x6B; /*1.5 other Flash command set */ FLASH_InitStruct->FLASH_cmd_wr_en = 0x06; FLASH_InitStruct->FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct->FLASH_cmd_rd_status = 0x05; FLASH_InitStruct->FLASH_cmd_rd_status2 = 0; FLASH_InitStruct->FLASH_cmd_wr_status = 0x01; FLASH_InitStruct->FLASH_cmd_wr_status2 = 0; FLASH_InitStruct->FLASH_cmd_chip_e = 0xC7; FLASH_InitStruct->FLASH_cmd_block_e = 0xD8; FLASH_InitStruct->FLASH_cmd_sector_e = 0x20; FLASH_InitStruct->FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct->FLASH_cmd_pwdn = 0xB9;	

FlashClass6	FLASH_InitStruct->FLASH_Id = FLASH_ID_MXIC;
	
	/*1.1 status bit define */
	FLASH_InitStruct->FLASH_QuadEn_bit = BIT(6);
	FLASH_InitStruct->FLASH_Busy_bit = BIT(0); //WIP
	FLASH_InitStruct->FLASH_WLE_bit = BIT(1); //WLE
	FLASH_InitStruct->FLASH_Status2_exist = 0;
	
	/*1.2 dummy cycle */
	FLASH_InitStruct->FLASH_rd_dummy_cyle[0] = 0;
	FLASH_InitStruct->FLASH_rd_dummy_cyle[1] = 0x04;
	FLASH_InitStruct->FLASH_rd_dummy_cyle[2] = 0x06;
	
	/*1.3 set 2 bits mode command */
	FLASH_InitStruct->FLASH_rd_dual_io = 0xBB;
	FLASH_InitStruct->FLASH_rd_dual_o = 0x3B;
	
	/*1.4 set 4 bits mode command */
	FLASH_InitStruct->FLASH_rd_quad_io = 0xEB;
	FLASH_InitStruct->FLASH_rd_quad_o = 0x6B;
	
	/*1.5 other Flash command set */
	FLASH_InitStruct->FLASH_cmd_wr_en = 0x06;
	FLASH_InitStruct->FLASH_cmd_rd_id = 0x9F;
	FLASH_InitStruct->FLASH_cmd_rd_status = 0x05;
	FLASH_InitStruct->FLASH_cmd_rd_status2 = 0;
	FLASH_InitStruct->FLASH_cmd_wr_status = 0x01;
	FLASH_InitStruct->FLASH_cmd_wr_status2 = 0;
	FLASH_InitStruct->FLASH_cmd_chip_e = 0x60;
	FLASH_InitStruct->FLASH_cmd_block_e = 0xD8;
	FLASH_InitStruct->FLASH_cmd_sector_e = 0x20;
	FLASH_InitStruct->FLASH_cmd_pwdn_release = 0xAB;
	FLASH_InitStruct->FLASH_cmd_pwdn = 0xB9;	

17.4 Pinmap Configuration

During the process of boot, sleep and deep sleep, I/O pull control is needed, and the chip will load the PAD status of each GPIO from the AON register. The correct configuration of pinmap can achieve low power consumption, otherwise, the unsuitable configuration may lead to leakage. For example, UART voltage level is high. If the UART pin is pulled down or not pulled, leakage power happens. So it is necessary to make sure that each pin has proper pull control.

In SDK, you should set GPIO function pull control and sleep/deep sleep pull control based on your PCB board, and SDK will set pull control based on your setting between suspend and resume.

In the ńNJ ú f y т н м Ř f LJ ψ LJ ƛ ƴ Y ŀ LJ Ŏ Ŧ Ǝ Φ Ŏ έ Ŧ ƛ f Ŝ Σ t a ! t ψ ¢ è LJ Ŝ 5 Ŝ Ŧ LJ Y ŀ LJ ψ Ŧ dz ƴ Ŏ ώ ϐ ă K ₴ dz f

Pin Name: indicates the GPIO.

Func PU/PD: is used to configure the PAD status when power-on.

Slp PU/PD: is used to configure the PAD status at sleep mode.

Dslp PU/PD: is used to configure the PAD status at deep sleep mode.

LowPowerPin: indicates that whether this GPIO is a low power pin. It is no need to be modified.

```
const PMAP_TypeDef pmap_func[]=
{
// Pin Name    Func PU/PD              Slp PU/PD              DSlp PU/PD             LowPowerPin
    {_PA_0,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_1,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_2,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_3,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_4,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_5,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_6,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_7,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //UART_LOG_TXD
    {_PA_8,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //UART_LOG_RXD
    {_PA_9,    GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_10,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_11,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_12,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_13,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_14,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_15,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_16,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_17,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_18,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_19,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_20,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_21,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_22,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_23,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_24,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_25,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_26,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    TRUE},   //Key-Scan
    {_PA_27,   GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //SWD_DATA or normal_mode_sel
    {_PA_28,   GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_29,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_30,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PA_31,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_0,    GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_1,    GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_2,    GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_3,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //SWD_CLK
    {_PB_4,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //Touch0
    {_PB_5,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //Touch1
    {_PB_6,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //Touch2
    {_PB_7,    GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //Touch3
    {_PB_8,    GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_9,    GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_10,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_11,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},
    {_PB_12,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //SPI_DATA3
    {_PB_13,   GPIO_PuPd_UP,      GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    FALSE},  //SPI_CLK
    {_PB_14,   GPIO_PuPd_UP,      GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    FALSE},  //SPI_DATA0
    {_PB_15,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //SPI_DATA2
    {_PB_16,   GPIO_PuPd_UP,      GPIO_PuPd_KEEP,    GPIO_PuPd_KEEP,    FALSE},  //SPI_CS
    {_PB_17,   GPIO_PuPd_UP,      GPIO_PuPd_DOWN,    GPIO_PuPd_KEEP,    FALSE},  //SPI_DATA1
```

```
    {_PB_18,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_19,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_20,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_21,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_22,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},  //IR pin should be no pull when using IR
    {_PB_23,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_24,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_25,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_26,    GPIO_PuPd_SHUTDOW   GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},  //This is a PCB board error
    {_PB_27,    GPIO_PuPd_UP,       GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_28,    GPIO_PuPd_UP,       GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_29,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_30,    GPIO_PuPd_UP,       GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PB_31,    GPIO_PuPd_DOWN,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
    {_PNC       GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,     GPIO_PuPd_KEEP,   FALSE},
};
```

There are five states of the PAD.

GPIO_PuPd_UP: indicates that the PAD is pulled up to VDDIO.
GPIO_PuPd_DOWN: indicates that the PAD is pulled down to GND.
GPIO_PuPd_NOPULL: indicates that the PAD is in Z High
GPIO_PuPd_KEEP: is used for sleep and deep sleep mode, and indicates that the PAD will maintain the last status before the chip enters sleep mode or deep sleep mode.
GPIO_PuPd_SHUTDOWN: indicates that the PAD is shut down.

The following section illustrates the recommendation of pad status configurations according to the function of different pins.

17.4.1 Normal GPIO

When the chip pin is used as a normal GPIO connecting with external circuit, the pad status depends on the state of the external circuit.

Table 17-6 Normal GPIO pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
Connected to VDD	UP/NO PULL	UP/NO PULL/KEEP	UP/NO PULL/KEEP
Connected to GND	DOWN/NO PULL	DOWN/NO PULL/KEEP	DOWN/NO PULL/KEEP
Floating	DOWN	DOWN	DOWN

17.4.2 Key-Scan

When the pin is used as Key-Scan function, the pad status depends on the state of the Key-Scan ROW or COL.

Table 17-7 Key-Scan pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
ROW	UP	KEEP	KEEP
COL	DOWN	KEEP	KEEP

17.4.3 LOGUART

The pins PA_7 and PA_8 are fixed to LOGUART function. The pad status is listed in Table 17-8.

Table 17-8 LOGUART pad status

Pin Function	Pin Name	Func PU/PD	Slp PU/PD	DSlp PU/PD
UART_LOG_TXD	PA_7	UP	KEEP	KEEP
UART_LOG_RXD	PA_8	UP	KEEP	KEEP

17.4.4 SWD

When the pin is configured as SWD function, the pad status is listed in Table 17-9.

Table 17-9 SWD pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
SWD_DATA	UP	KEEP	KEEP
SWD_CLK	UP	KEEP	KEEP

17.4.5 Flash Pin

Generally, the Flash data pin is pulled to DOWN. If you still need to access Flash (such as single bit access) during sleep, this pin needs to be configured to UP before entering sleep mode. The pad status are listed in Table 17-9

Table 17-10 Flash pin pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
SPI_DATA_X	DOWN	KEEP	KEEP
SPI_CS	UP	KEEP	KEEP

17.4.6 Cap-Touch

When the pin is configured as Cap-Touch function, it is not necessary to set the pad status, which should be NO PULL.

Table 17-11 Cap-Touch pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
Cap-Touch	NO PULL	KEEP	KEEP

17.4.7 ADC

When the pin is configured as ADC function, it is not necessary to set the pad status, which should be NO PULL.

Table 17-12 ADC pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
ADC	NO PULL	KEEP	KEEP

17.4.8 Power

The GPIO pin can also output 3.3V by being set to output High. However, the driving capacity is limited, usually about 4mA. At this time, the pad status should be NO PULL.

Table 17-13 Power pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
Power source	NO PULL	KEEP	KEEP

17.4.9 DMIC

When the pin is used as the DMIC function for audio, the pad status should be pulled down.

Table 17-14 DMIC pad status

Pin Function	Func PU/PD	Slp PU/PD	DSlp PU/PD
DMIC_DATA	DOWN	KEEP	KEEP

DMIC_CLK	DOWN	KEEP	KEEP

17.4.10 AMIC N/P

When the pin is used as the AMIC N/P function for audio, the pad status should be NO PULL.

Table 17-15 AMIC N/P pad status

Pin Function	Pin Name	Func PU/PD	Slp PU/PD	DSlp PU/PD
MIC_N	X	NO PULL	KEEP	KEEP
MIC_P	X	NO PULL	KEEP	KEEP
MICBIAS	X	NO PULL	KEEP	KEEP

17.4.11 AOUT_L/R N/P

When the pin used as the ! h ¦ ¢ ψ [κ w b κ t Ŧ ƨ NJ ŀ dz Ř ƛ ƨ Σ ű Ƙ Ŝ LJ ŀ Ř ă ű ŀ ű dz ă ă Ƙ ƨ dz ƭ Ř ō Ŝ Ψ b h

Table 17-16 AOUT_L/R N/P PAD status

Pin Function	Pin Name	Func PU/PD	Slp PU/PD	DSlp PU/PD
AOUT_N	X	NO PULL	KEEP	KEEP
AOUT_P	X	NO PULL	KEEP	KEEP

17.5 Sleep Configuration

The rtl8721dlp_sleepcfg.c should keep ű Ƙ Ŝ Ř Ŝ Ŧ ŀ dz ƭ ű ă Ŝ ű ű ƛ ƴ Ǝ ă Σ è ƨ dz Ǒ ŀ ƴ ƨ ƴ ƭ è Ǒ Ƙ ŀ ƴ Ǝ Ŝ ƛ ű ø ƛ

17.6 Boot Configuration

Bootloader can be configured through the rtl8721d_bootcfg.c file, as Table 17-17 and Fig 17-8 show.

Table 17-17 User bootcfg

Configuration items	Comment
Flash_MMU_Config	Used for MMU configuration. OTA2 address mapping should be settled here.
OTA_Region	OTA start address
RSIP_Mask_Config	RSIP mask configure
Force_OTA1_GPIO	Force OTA1 GPIO configuration
Boot_Log_En	Used to disable Realtek boot log
FwCheckCallback	Firmware verify callback handler
OTASelectHook	Firmware select hook

```
/*
* @brief     MMU Configuration.
*     There are 8 MMU entries totally. Entry 0 & Entry 1 are already used by OTA, Entry 2~7 can be used by Users.
*/
BOOT_RAM_DATA_SECTION
MMU_ConfDef Flash_MMU_Config[] = {
    /*VAddrStart,  VAddrEnd,          PAddrStart,          PAddrEnd*/
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},    //Entry 2
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},    //Entry 3
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},    //Entry 4
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},    //Entry 5
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},    //Entry 6
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},    //Entry 7
    {0xFFFFFFFF,    0xFFFFFFFF,       0xFFFFFFFF,       0xFFFFFFFF},
};

/*
* @brief     OTA start address. Because KM0 & KM4 IMG2 are combined, users only need to set the start address
*     of KM0 IMG2.
*/
BOOT_RAM_DATA_SECTION
u32 OTA_Region[2] = {
    0x08006000,     /* OTA1 region start address */
    0x08106000,     /* OTA2 region start address */
};

/*
* @brief     RSIP Mask Configuration.
*     There are 4 RSIP mask entries totally. Entry 0 is already used by System Data, Entry 3 is reserved by Realtek.
*     Only Entry 1 & Entry 2 can be used by Users.
*     MaskAddr: start address for RSIP Mask, should be 4KB aligned
*     MaskSize: size of the mask area, unit is 4KB
*/
BOOT_RAM_DATA_SECTION
RSIP_MaskDef RSIP_Mask_Config[] = {
    /*MaskAddr,          MaskSize*/
    {0x08001000,    3},         //Entry 0: 4K system data & 4K backup & DPK

    /* customer can set here */
    {0xFFFFFFFF,    0xFFFF},    //Entry 1: can be used by users
    {0xFFFFFFFF,    0xFFFF},    //Entry 2: can be used by users
    {0xFFFFFFFF,    0xFFFF},    //Entry 3: Reserved by Realtek. If RDP is not used, this entry can be used by users.

    /* End */
    {0xFFFFFFFF,    0xFFFF},
};

/**
* @brief  GPIO force OTA1 as image2. 0xFF means force OTA1 trigger is disabled.
*     BIT[7]: active level, 0 is low level active, 1 is high level active,
*     BIT[6:5]: port, 0 is PORT_A, 1 is PORT_B
*     BIT[4:0]: pin num 0~31
*/
BOOT_RAM_DATA_SECTION
u8 Force_OTA1_GPIO = 0xFF;

/**
* @brief  boot log enable or disable.
*     FALSE: disable
*     TRUE: enable
*/
BOOT_RAM_DATA_SECTION
u8 Boot_Log_En = FALSE;


/**
* @brief  Firmware verify callback handler.
*     If users need to verify checksum/ hash for image2, implement the function and assign the address
*     to this function pointer.
* @param  None
* @retval 1: Firmware verify successfully, 0: verify failed
*/
BOOT_RAM_DATA_SECTION
FuncPtr FwCheckCallback = NULL;

/**
* @brief  Firmware select hook.
*     If users need to implement own OTA select method, implement the function and assign the address
*     to this function pointer.
* @param  None
* @retval 0: both firmwares are invalid, select none, 1: boot from OTA1, 2: boot from OTA2
*/
BOOT_RAM_DATA_SECTION
FuncPtr OTASelectHook = NULL;
```

Fig17-8 User bootcfg

17.7 IPC Configuration

You can add IPC entry referring to Fig17-9, so that KM4 & KM0 can send message to each other by the method you defined.

For USER_MSG_TYPE

IPC_USER_POINT: Message in IRQFUNC is a pointer (address).

IPC_USER_DATA: Message in IRQFUNC is just a value

```
#if defined (ARM_CORE_CM4)
const IPC_INIT_TABLE ipc_init_config[] =
{
    // USER_MSG_TYPE        IRQFUNC                         IRQDATA
    {IPC_USER_DATA,         shell_switch_ipc_int,       (VOID*) NULL},//channel 0: IPC_INT_CHAN_SHELL_SWITCH
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 1: IPC_INT_CHAN_WIFI_FW
    {IPC_USER_DATA,         FLASH_Write_IPC_Int,        (VOID*) NULL},//channel 2: IPC_INT_CHAN_FLASHPG_REQ
    {IPC_USER_POINT, NULL,                              (VOID*) NULL},//channel 3: IPC_INT_KM4_TICKLESS_INDICATION
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 4: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 5: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 6: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 7: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 8: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 9: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 10: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 11: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 12: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 13: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 14: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 15: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 16: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 17: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 18: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 19: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 20: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 21: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 22: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 23: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 24: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 25: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 26: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 27: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 28: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 29: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 30: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 31: Reserved for Customer use

    {0xFFFFFFFF,            OFF,                            OFF},     /* Table end */
};
#else
#if CONFIG_WIFI_FW_EN
extern void driver_fw_flow_ipc_int(VOID *Data, u32 IrqStatus, u32 ChanNum);
#define fw_flow_ipc_int driver_fw_flow_ipc_int
#else
#define fw_flow_ipc_int NULL
#endif
const IPC_INIT_TABLE ipc_init_config[] =
{
    // USER_MSG_TYPE        IRQFUNC                         IRQDATA
    {IPC_USER_DATA,         shell_switch_ipc_int,       (VOID*) NULL},//channel 0: IPC_INT_CHAN_SHELL_SWITCH
    {IPC_USER_DATA,         fw_flow_ipc_int,                (VOID*) IPCM4_DEV},//channel 1: IPC_INT_CHAN_WIFI_FW
    {IPC_USER_DATA,         FLASH_Write_IPC_Int,        (VOID*) NULL},//channel 2: IPC_INT_CHAN_FLASHPG_REQ
    {IPC_USER_POINT, km4_tickless_ipc_int,      (VOID*) NULL},//channel 3: IPC_INT_KM4_TICKLESS_INDICATION
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 4: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 5: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 6: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 7: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 8: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 9: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 10: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 11: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 12: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 13: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 14: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 15: Reserved for Realtek use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 16: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 17: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 18: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 19: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 20: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 21: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 22: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 23: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 24: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 25: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 26: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 27: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 28: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 29: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 30: Reserved for Customer use
    {IPC_USER_DATA,         NULL,                           (VOID*) NULL},//channel 31: Reserved for Customer use

    {0xFFFFFFFF,            OFF,                            OFF},     /* Table end */
};
#endif
```

Fig17-9 User ipccfg

NOTE

Use IPC channel 16~31 and IPC semaphore index 8~15, Channel 0~15 and semaphore index 0~7 are reserved for Realtek use

17.8 LP_intf Configuration

UART low power RX enable/disable can be configured through the rtl8721dlp_intfcfg.c file.

```
UARTCFG_TypeDef uart_config[2]=
{
    /* UART0 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE, /*Enable low power RX*/
    },
    /* UART1 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE,
    },
};
```

17.9 HP_intf Configuration

PSRAM parameters, SDIO host parameters and FTL parameters can be configured through the rtl8721dhp_intfcfg.c file.

```
#include "ameba_soc.h"
#include "autoconf.h"


PSRAMCFG_TypeDef psram_dev_config = {
    .psram_dev_enable = FALSE,                  //enable psram
    .psram_dev_cal_enable = FALSE,          //enable psram calibration function
    .psram_dev_retention = FALSE,           //enable psram retention
};

SDIOHCFG_TypeDef sdioh_config = {
    .sdioh_bus_speed = SD_SPEED_HS,             //SD_SPEED_DS or SD_SPEED_HS
    .sdioh_bus_width = SDIOH_BUS_WIDTH_4BIT,   //SDIOH_BUS_WIDTH_1BIT or SDIOH_BUS_WIDTH_4BIT
    .sdioh_cd_pin = _PB_25,                     //_PB_25/_PA_6/_PNC
    .sdioh_wp_pin = _PNC,                       //_PB_24/_PA_5/_PNC
};

#if defined(CONFIG_FTL_ENABLED)
#define FTL_MEM_CUSTEM      0
#if FTL_MEM_CUSTEM == 0
#error "You should allocate flash sectors to for FTL physical map as following, then set FTL_MEM_CUSTEM to 1. For more information, Please refer to Application Note, FTL chapter. "
#else
const u8 ftl_phy_page_num = 3;                          /* The number of physical map pages, default is 3*/
const u32 ftl_phy_page_start_addr = 0x00102000;         /* The start offset of flash pages which is allocated to FTL physical map.
                                                           Users should modify it according to their own memory layout! ! */
#endif
#endif
```

18 Flash Operation

18.1 Functional Description

AmebaD uses SPI Controller (SPIC) to communicate with SPI NOR Flash.

There are two operation modes for SPIC: auto mode and user mode.

User mode: A typical software flow to implement all serial transfer. User can transmit or receive data from Flash through setting up SPIC registers.

Auto mode: Compared to user mode, auto mode is a hardware control flow to execute. After initial setting, user don't need to configure the related control register for each transfer operation. Auto mode is a convenient way to access Flash just as access memory (SRAM or DRAM).

The SPIC is initialized automatically in boot sequence, and user must not initialize SPIC manually. The supported mode for different operations can be found in Table 18-1.

Table 18-1 SPIC operation mode

Flash Operation	SPIC Operation Mode
Read data	Auto mode / User mode
Program data	User mode
Erase	User mode
Receive/transmit command	User mode

Note: Auto mode and user mode cannot be used at the same time.

18.2 Protection Method

Considering AmebaD supports Flash execute in place (XIP), which enables code execute directly in Flash memory without copying to RAM. Both KM0 and KM4 CPU cores can issue request to SPIC in auto mode to read instructions from Flash to execute.

To prevent being interrupted by auto mode reading, the user mode operation should be protected until it is finished. Fig. 18-1 shows the flow of KM0 manipulating Flash safely in user mode, which has been protected. The method of KM4 manipulating Flash is in the same way.

The protection methods are implemented in FLASH_Write_Lock() and FLASH_Write_Unlock() functions. The APIs provided by Realtek in 18.3 and 18.4 are already protected and can be called by users directly.

Note: If users need to implement user mode function by themselves, the code of manipulating SPIC should locate in RAM instead of Flash. They should also call FLASH_Write_Lock() before operation and call FLASH_Write_Unlock() to release from protection after operation.

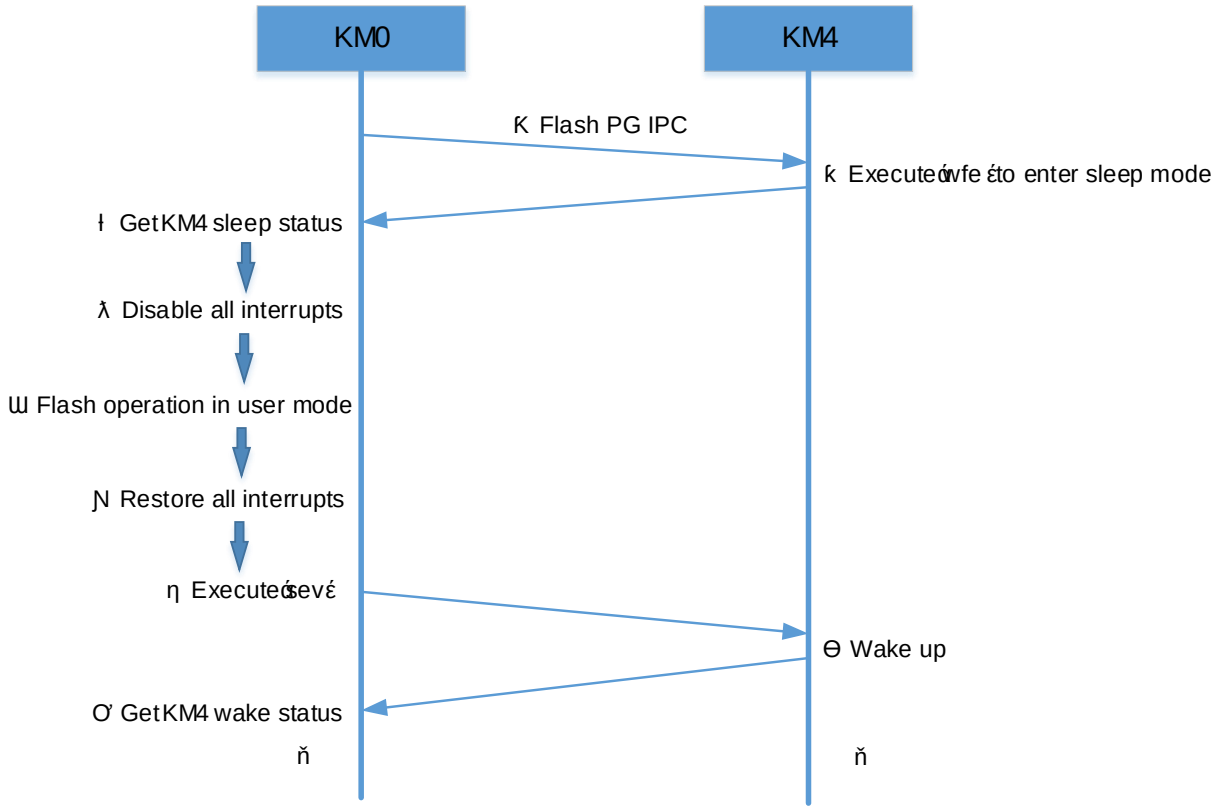


Fig18-1 User mode operation flow

18.3 Flash Raw APIs

18.3.1 Read

18.3.1.1 HAL_READ8/HAL_READ16/HAL_READ32

Items	Description
Introduction	Read 1-byte/1-word/1-dword data from Flash in auto mode, just as access SRAM
Parameters	base: the base address of Flash memory, it should be SPI_FLASH_BASE which is 0x08000000. addr: the offset address in Flash memory, it should be byte-aligned for HAL_READ8, word-aligned for HAL_READ16 and dword-aligned for HAL_READ32
Return	The read data

18.3.1.2 FLASH_ReadStream

Items	Description
Introduction	Read a stream of data from specified offset address of Flash in auto mode.
Parameters	address: specify the starting offset address to read from. It has no alignment restrictions. len: specify the length of the data to read. data: specify the buffer to save the read back data.
Return	Status 1: success Others: fail

18.3.2 Write

18.3.2.1 FLASH_TxData12BXIP

Items	Description
Introduction	Write data to flash in user mode. This function is already protected and can be called by user directly.
Parameters	StartAddr: start offset address in Flash from which SPIC writes. DataPhaseLen: the number of bytes that SPIC sends in Data Phase. It should not be more than 12 bytes. pData: pointer to a byte array that is to be sent.
Return	N/A

18.3.2.2 FLASH_WriteStream

Items	Description
Introduction	Write a stream of data to specified address in user mode. This function is already protected and can be called by user directly.
Parameters	address: specifies the starting offset to write to. len: specifies the length of the data to write. It has no restrictions of less than 12 bytes data: pointer to a byte array that is to be written.
Return	Status 1: success Others: fail

18.3.3 Erase

18.3.3.1 FLASH_EraseXIP

Items	Description
Introduction	Erase sector/block/chip for Flash in user mode. The function is already protected and can be called by users directly.
Parameters	EraseType: can be one of the following value. EraseChip: erase the whole chip. EraseBlock: erase the specified block (64KB). EraseSector: erase the specified sector (4KB). Address: address should be 4-byte aligned. The block/sector which the address in will be erased.
Return	N/A

18.3.4 Receive/Transmit Command

18.3.4.1 FLASH_RxCmdXIP

Items	Description
Introduction	Send Rx command to Flash to get status register or Flash in user mode. This function is already protected and can be called by user directly.
Parameters	cmd: command that need to be sent. read_len: the number of bytes that will be read by SPIC after sending command. read_data: pointer to a byte array which is used to save data received.
Return	N/A

18.3.4.2 FLASH_SetStatusXIP

Items	Description

Introduction	Set Flash status register in user mode. The function is already protected and can be called by users directly.
Parameters	cmd: command to be sent. len: the number of bytes to be sent after sending command. status: pointer to byte array to be sent.
Return	N/A

18.4 Flash Mbed APIs

The descriptions of Flash Mbed APIs can be found in *AN0403 Peripheral Driver Mbed API.chm*. All these functions have already been protected and can be called directly.

18.5 User Configuration

AmebaD provides rtl8721dlp_flashcfg.c to configure Flash speed and read mode.

```
/**
* @brif Indicate the flash baudrate. It can be one of the following value:
*       0xFFFF: 80MHz
*       0x7FFF: 100MHz
*       0x3FFF: 67MHz
*       0x1FFF: 57MHz
*       0x0FFF: 50MHz
*/
const u16 Flash_Speed = 0xFFFF;
```

Fig 18-2 Configuring Flash speed

```
/**
* @brif Indicate the flash read I/O mode. It can be one of the following value:
*       0xFFFF: Read quad IO, Address & Data 4 bits mode
*       0x7FFF: Read quad O, Just data 4 bits mode
*       0x3FFF: Read dual IO, Address & Data 2 bits mode
*       0x1FFF: Read dual O, Just data 2 bits mode
*       0x0FFF: 1 bit mode
* @note If the configured read mode is not supported, other modes would be searched until find the appropriate mode.
*/
const u16 Flash_ReadMode = 0xFFFF;
```

Fig 18-3 Configuring Flash read mode

Note: For some Flash chip which is not available in Flash AVL, the principle to determine whether it can be supported by SDK and the method to add a new Flash can be found in User Configuration

19 Battery Measurement

19.1 Functional Description

LP-ADC has a total of 11 channels, of which 8 are external channels and 3 are internal channels. 8 external battery measurement channels include 7 normal channels and 1 VBAT channel, as Table 19-1 shows. The VBAT channel measurable range is 0~5V.

Table 19-1 Channel description

Function	ADC Channel	Pin Name	Voltage Range
Normal channel	CH0~CH6	PB[4]~PB[7] (CH0~CH3)	0~3.3V (when power is 3.3V)
		PB[1]~PB[3] (CH4~CH6)	0~1.8V (when power is 1.8V)
VBAT	CH7	VBAT_MEAS	0~5V

When one channel is added to channel switch list, LP-ADC would convert the voltage of corresponding pin to digital data. The resolution of digital sample data is 12-bit.

To obtain sample data, auto mode, timer-trigger mode and software-trigger mode can be adopted according to different usages. LP-ADC can also be configured to send wakeup and interrupt signal to system when the sample data matches criteria.

19.2 Calibration

The relationship between input voltage and sample data is almost linear. Fig. 19-1 shows the conversion of sample data and input voltage of VBAT_MEAS.

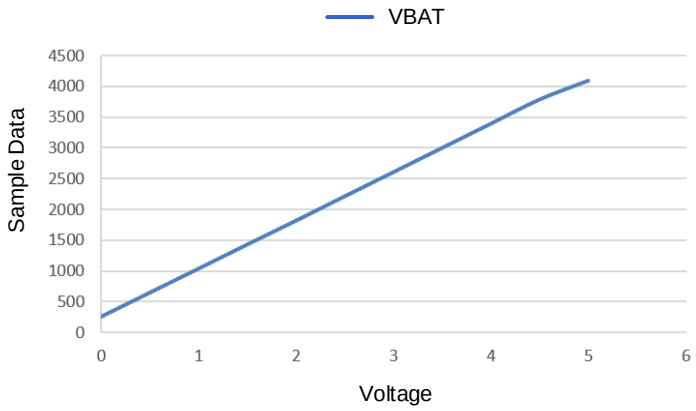


Fig. 19-1 Relationship between sample data and input voltage

To obtain voltage from sample data, the following formula can be used:

$$\text{Voltage} = \frac{\text{Data} - \text{OFFSET}}{\text{GAIN}}$$

Where

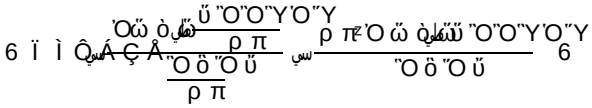
Data: 12-bit sample data

OFFSET: 10 times of sample data of 0.000V

GAIN: 10 times of sample data increment when voltage rises 1V

To enhance conversion accuracy, ADC calibration is necessary to obtain OFFSET and GAIN parameters for channels.

The calibration operations can be done in FT test. The FT test would sample for different voltages and use Fitting Method to calculate OFFSET and GAIN. Pay attention that the OFFSET and GAIN values gotten from FT test are 10 times of the actual values. When calculating voltage using the above two values, they must be divided by 10 first; that is,



As normal external channels use the same voltage divider circuit, they can use one set of calibration parameters. But VBAT channel need to be calibrated independently. These parameters would be programmed into eFuse after calibration, as Table 19-2 shows

Table 19-2 Calibration parameters location

Channel	Input Mode	Parameter	Address
Normal (CH0~CH6)	Single ended	OFFSET	Physical map 0x1D0~0x1D1
		GAIN	Physical map 0x1D2~0x1D3
	Differential	OFFSET	Physical map 0x1D8~0x1D9
		GAIN	Physical map 0x1DA~0x1DB
VBAT (CH7)	Single ended	OFFSET	Physical map 0x1D4~0x1D5
		GAIN	Physical map 0x1D6~0x1D7

To calculate voltage from sample data, user only need to obtain calibration parameters from eFuse and call the voltage computational formula

20 Wi-Fi

20.1 Wi-Fi Data Structures

Data Structures	Introduction
<_cus_ie>	The structure is used to set Wi-Fi custom IE list, and type match CUSTOM_IE_TYPE. The IE will be transmitted according to the type.
<rtw_ssid>	The structure is used to describe the SSID.
<rtw_mac>	The structure is used to describe the unique 6-byte MAC address.
<rtw_ap_info>	The structure is used to describe the setting about SSID, security type, password and default channel, start AP mode.
<rtw_network_info>	The structure is used to describe the station mode setting about SSID, security type and password, use when connecting to an AP.
<rtw_scan_result>	The structure is used to describe the scan result of the AP.
<rtw_scan_handler_result>	The structure is used to describe the data needed by scan result handler function.
<rtw_wifi_setting>	The structure is used to store the Wi-Fi setting gotten from Wi-Fi driver.
<rtw_wifi_config>	The structure is used to describe the setting when configure the network.
<rtw_maclist_t>	The structure is used to describe the maclist.
<rtw_bss_info_t>	The structure is used to describe the bss info of the network. It includes the version, BSSID, beacon_period, capability, SSID, channel, atm_window, dtim_period, SSI, R

20.2 Wi-Fi APIs

20.2.1 System APIs

API	Introduction
<wifi_on>	Enable Wi-Fi.
<wifi_off>	Disable Wi-Fi.
<wifi_is_up>	Check if the specified interface is up.
<wifi_is_ready_to_transceive>	Determine if a particular interface is ready to transceiver Ethernet packets.
<wifi_rf_on>	Enable Wi-Fi RF.
<wifi_rf_off>	Disable Wi-Fi RF.

20.2.1.1 wifi_on

Enable Wi-Fi: ō NJ ƛ ƴ Ǝ ű K Ŝ ø ƛ NJ Ŝ f Ŝ ă ă ƛ ƴ ű Ŝ NJ Ŧ I· Ŏ Ŝ ά ¦ LJ ε Φ

Parameter	Type	Introduction
<mode>	rtw_mode_t	Decide to enable Wi-Fi in which mode. The optional modes are enumerated in rtw_mode_t.

20.2.1.2 wifi_off

Disable Wi-Fi.

Parameter: None.

20.2.1.3 wifi_is_up

Check if the specified interface is up.

Parameter	Type	Introduction
<interface>	rtw_interface_t	The interface can be set as RTW_STA_INTERFACE or RTW_AP_INTERFACE (rtw_i

20.2.1.4 wifi_is_ready_to_transceive

Determine if a particular interface is ready to transceiver Ethernet packets.

Parameter	Type	Introduction
<interface>	rtw_interface_t	Interface to check RTW_STA_INTERFACE RTW_AP_INTERFACE

20.2.2 Scan APIs

API	Introduction
<wifi_scan>	Initiate a scan to search for 802.11 networks
<wifi_scan_networks>	Simplify the scan operation based on wifi_scan to search for 802.11 networks
<wifi_scan_networks_with_ssid>	Initiate a scan to search for 802.11 networks with specified SSID

20.2.2.1 wifi_scan

Initiate a scan to search for 802.11 networks.

Parameter	Type	Introduction
<scan_type>	rtw_scan_type_t	Specifies whether the scan should be active, passive or scan prohibited channels.
<bss_type>	rtw_bss_type_t	Specifies whether the scan should search for infrastructure networks (those using an AP), hoc networks, or both types.
<result_ptr>	void *	Scan specific ssid. The first 4 bytes is ssid length, and ssid name append after it. If no s ssid need to scan, please clean result_ptr before pass it into parameter.
<result_ptr>	void *	[out] a pointer to a pointer to a result storage structure.

Note:

The scan progressively accumulates results over time, and may take 6 seconds to complete. The results of the scan will be individually provided to the callback function.

The callback function will be executed in the context of the RTW thread.

When scanning specific channels, devices with a strong signal strength on nearby channels may be detected

20.2.2.2 wifi_scan_networks

Initiate a scan to search for 802.11 networks, a higher level API based on wifi_scan to simplify the scan operation.

Parameter	Type	Introduction
<results_handler>	rtw_scan_result_handler_t	The callback function which will receive and process the result data.
<user_data>	void *	User specified data that will be passed directly to the callback function

Note: Callback must not use blocking functions, since it is called from the context of the RTW thread. Callback, user_data variables will be referenced after the function returns. Those variables must remain valid until the scan is complete. The usage of this API can refer to ATWS in atcmd_wifi.c.

20.2.2.3 wifi_scan_networks_with_ssid

Initiate a scan to search for 802.11 networks with specified SSID.

Parameter	Type	Introduction
<results_handler>	int	The callback function which will receive and process the result data.
<user_data>	void *	User specified data that will be passed directly to the callback function
<scan_buflen>	int	The length of the result storage structure.
<ssid>	char *	The SSID of target network.
<ssid_len>	int	The length of the target network SSID.

Note: Callback must not use blocking functions, since it is called from the context of the RTW thread. The callback, user_data variables will be referenced after the function returns. Those variables must remain valid until the scan is completed.

20.2.3 Connection APIs

API	Introduction
<wifi_connect>	Join a WiFi network with specified SSID.
<wifi_connect_bssid>	Join a WiFi network with specified BSSID.
<wifi_disconnect>	Disassociates from current WiFi network.
<wifi_is_connected_to_ap>	Check if WiFi has connected to AP before dhcp.
<wifi_config_autoreconnect>	Set reconnection mode with configuration.
<wifi_set_autoreconnect>	Set reconnection mode with 3 retry limit and 5 seconds timeout as default.
<wifi_get_autoreconnect>	Get the result of setting reconnection mode.
<wifi_get_last_error>	Present the device disconnect reason while connecting.

20.2.3.1 wifi_connect

Join a WiFi network with specified SSID. Scan for, associate and authenticate with a WiFi network. On successful return, the system is ready to send data packets

Parameter	Type	Introduction
<ssid>	char *	A null terminated string containing the SSID name of the network to join.
<security_type>	rtw_security_t	Authentication type: RTW_SECURITY_OPEN: open security RTW_SECURITY_WEP_PSK: WEP Security with open authentication RTW_SECURITY_WEP_SHARED: WEP Security with shared authentication RTW_SECURITY_WPA_TKIP_PSK: WPA Security RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher RTW_SECURITY_WPA2_TKIP_PSK: WPA2 Security using TKIP cipher RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers
<password>	char *	A byte array containing either the cleartext security key for WPA/WPA2 secured networks, or a pointer to an array of rtw_wep_key_t structures for WEP secured networks.
<ssid_len>	int	The length of the SSID in bytes.
<password_len>	int	The length of the security_key in bytes.
<key_id>	int	The index of the wep key (0, 1, 2, or 3). If not using it, leave it with value -1
<semaphore>	void *	A user provided semaphore is flagged when the join is complete. If not using, leave it with NULL

Note: Make sure the WiFi is enabled before invoking this function. (wifi_on())

20.2.3.2 wifi_connect_bssid

Join a WiFi network with specified BSSID. Scan for, associate and authenticate with a WiFi network. On successful return, the system is ready to send data packets.

Parameter	Type	Introduction
<bssid>	unsigned char	The specified BSSID to connect.

<ssid>	char *	A null terminated string containing the SSID name of the network to join.
<security_type>	rtw_security_t	Authentication type: RTW_SECURITY_OPEN: open security RTW_SECURITY_WEP_PSK: WEP Security with open authentication RTW_SECURITY_WEP_SHARED: WEP Security with shared authentication RTW_SECURITY_WPA_TKIP_PSK: WPA Security RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher RTW_SECURITY_WPA2_TKIP_PSK: WPA2 Security using TKIP cipher RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers
<password>	char *	A byte array containing either the cleartext security key for WPA/WPA2 secured networks, or a p to an array of rtw_wep_key_t structures for WEP secured networks.
<ssid_len>	int	The length of the SSID in bytes.
<password_len>	int	The length of the security_key in bytes.
<key_id>	int	The index of the wep key (0, 1, 2, or 3). If not using it, leave with value-1.
<semaphore>	void *	A user provided semaphore is flagged when the join is complete. If not using, leave it with NULL

Note:

Make sure the Wi-Fi is enabled before invoking this function. (wifi_on())
The difference between wifi_connect_bssid() and wifi_connect() is that BSSID has higher priority as the basis of connection in wifi_connect_bssid().

20.2.3.3 wifi_disconnect

Disassociates from current Wi-Fi network.

Parameter None.

20.2.3.4 wifi_is_connected_to_ap

Check if Wi-Fi has connected to AP before dhcp.

Parameter None.

20.2.3.5 wifi_config_autoreconnect

Set reconnection mode with configuration.

Parameter	Type	Introduction
<mode>	_u8	Set 1/0 to enable/disable the reconnection mode.
<callback>	_u8	The number of retry limit.
<len_used>	_u16	The timeout value (in seconds).

Note: The difference between wifi_config_autoreconnect() and wifi_set_autoreconnect() is that user can specify the retry times and timeout value in wifi_config_autoreconnect(). But in wifi_set_autoreconnect(), these values are set with 8 retry limit and 5 seconds timeout as default.

20.2.3.6 wifi_set_autoreconnect

Set reconnection mode with 8 retry limit and 5 seconds timeout as default.

Parameter	Type	Introduction
<mode>	_u8	Set 1/0 to enalbe/disable the reconnection mode.

Note: The difference between wifi_config_autoreconnect() and wifi_set_autoreconnect() is that user can specify the retry times and timeout value in wifi_config_autoreconnect(). But in wifi_set_autoreconnect(), these values are set with 8 retry limit and 5 seconds timeout as default.

20.2.3.7 wifi_get_autoreconnect

Get the result of setting reconnection mode.

Parameter	Type	Introduction
<mode>	_u8	Pointer to the result of setting reconnection mode.

20.2.3.8 wifi_get_last_error

Present the device disconnect reason while connecting.

Parameter None.

20.2.4 Channel APIs

API	Introduction
<wifi_set_channel>	Set the listening channel on STA interface.
<wifi_get_channel>	Get the current channel on STA interface.
<wifi_set_channel_plan>	Set channel plan into eFuse, must reboot after setting channel plan.
<wifi_get_channel_plan>	Get channel plan from eFuse.

20.2.4.1 wifi_set_channel

Set the listening channel on STA interface.

Parameter	Type	Introduction
<channel>	int	The desired channel.

Note: Do not need to call this function for STA mode FW driver, since it will be determined by the channel from received beacon.

20.2.4.2 wifi_get_channel

Get the current channel on STA interface.

Parameter	Type	Introduction
<channel>	int *	A pointer to the variable where the channel value will be written.

20.2.4.3 wifi_set_channel_plan

Set channel plan into eFuse, must reboot after setting channel plan.

Parameter	Type	Introduction
<channel_plan>	uint8_t	The value of channel plan, define in wifi_constants.h

20.2.4.4 wifi_get_channel_plan

Get channel plan from eFuse.

Parameter	Type	Introduction
<channel_plan>	uint8_t	Point to the value of channel plan, define in wifi_constants.h

20.2.4.5 wifi_change_channel_plan

Switch the current channelPlan via thesoftware way

Parameter	Type	Introduction
<channel_plan>	uint8_t	The value of channel plan, definewifi_constants.h

20.2.4.6 wifi_ set_country

Setcountry code and setthe channel plan accordig to the country code via thsoftware way

Parameter	Type	Introduction
<country_code>	rtw_country_code_t	Set country code andset the channel plan according tthe country code

20.2.5 Power APIs

API	Introduction
<wifi_enable_powersave>	Enable Wi-Fi powersave mode.
<wifi_disable_powersave>	DisableWi-Fi powersave mode.
<wifi_get_txpower>	Get theTx power in index units.
<wifi_set_txpower>	Set the Tx power in index units.
<wifi_set_power_mode>	Set IPS/LPS mode.
<wifi_set_lps_dtim>	Set LPS DTIM.
<wifi_get_lps_dtim>	Get LPS DTIM.

20.2.5.1 wifi_enable_powersave

Enable Wi-Fi powersave modeWhen power save mode is enabled, RF will wake up only when there is data to be sent or beacon to be received. Otherwise, RF will be awake all the time.

Parameter None.

20.2.5.2 wifi_disable_powersave

DisableWi-Fi powersave mode.

Parameter None.

20.2.5.3 wifi_get_txpower

Get the Tx power in index units.

Parameter	Type	Introduction
<poweridx>	int *	The variable to receive theTxpower in index.

20.2.5.4 wifi_set_txpower

Set the Txpower in index units.

Parameter	Type	Introduction
<poweridx>	int	The desiredTxpower in index.

20.2.5.5 wifi_set_power_mode

Set IPS/LPS mode.

Parameter	Type	Introduction
<ips_mode>	unsigned char	The desired IPS mode. It becomes effective when WLAN enters IPS. ips_mode is inactive power save mode. FW automatically turns RF off if the device is not associated to AP. Set 1 to enable inactive power save mode.
<lps_mode>	unsigned char	The desired LPS mode. It becomes effective when WLAN enters LPS. lps_mode is leisure power save mode. FW turns RF on to listen to beacon automatically and periodically during the association to AP. If traffic is not busy, it also automatically turns RF off. Set 1 to enable leisure power save mode.

20.2.5.6 wifi_set_lps_dtim

Set LPS DTIM.

Parameter	Type	Introduction
<dtim>	unsigned char	In LPS, the package can be buffered at AP side. STA leaves LPS until DTIM count of packet buffered at AP side.

Note: DTIM is the duration to listen beacon. The default DTIM is 1, which is 100ms. If DTIM is set bigger than 1, the advantage is that power can be saved and the disadvantage is that STA has the risk of losing packets.

20.2.5.7 wifi_get_lps_dtim

Get LPS DTIM.

Parameter	Type	Introduction
<dtim>	unsigned char *	In LPS, the package can be buffered at AP side. STA leaves LPS until DTIM count of packet buffered at AP side.

20.2.6 APMode APIs

API	Introduction
<wifi_start_ap>	Trigger Wi-Fi driver to start an infrastructure WiFi network.
<wifi_start_ap_with_hidden_ssid>	Start an infrastructure WiFi network with hidden SSID.
<wifi_restart_ap>	Trigger Wi-Fi driver to restart an infrastructure Wi-Fi network.
<wifi_get_associated_client_list>	Get the associated clients with SoftAP.
<wifi_enable_forwarding>	Enable packets forwarding in AP mode.
<wifi_disable_forwarding>	Disable packets forwarding in AP mode.

20.2.6.1 wifi_start_ap

Trigger Wi-Fi driver to start an infrastructure WiFi network.

Parameter	Type	Introduction
<ssid>	char *	A null terminated string containing the SSID name of the network.
<security_type>	security_type	RTW_SECURITY_OPEN: Open Security RTW_SECURITY_WPA_TKIP_PSK: WPA Security RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher

		RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers WEP security is NOT IMPLEMENTED. It is NOT SECURE!
<password>	char *	A byte array containing the cleartext security key for the network.
<ssid_len>	int	The length of the SSID in bytes.
<password_len>	int	The length of the security_key in bytes.
<channel>	int	802.11 channel number.

Note:

Make sure the Wi-Fi is enabled before invoking this function. (wifi_on())

If a STA interface is active when this function is called, the softAP will start on the same channel as the STA, using this channel provided.

20.2.6.2 wifi_start_ap_with_hidden_ssid

Start an infrastructure Wi-Fi network with hidden SSID.

Parameter	Type	Introduction
<ssid>	char *	A null terminated string containing the SSID name of the network.
<security_type>	security_type	RTW_SECURITY_OPEN: Open Security RTW_SECURITY_WPA_TKIP_PSK: WPA Security RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers WEP security is NOT IMPLEMENTED. It is NOT SECURE!
<password>	char *	A byte array containing the cleartext security key for the network.
<ssid_len>	int	The length of the SSID in bytes.
<password_len>	int	The length of the security_key in bytes.
<channel>	int	802.11 channel number.

Note: if a STA interface is active when this function is called, the softAP will start on the same channel as the STA, using this channel provided.

20.2.6.3 wifi_restart_ap

Trigger Wi-Fi driver to restart an infrastructure Wi-Fi network.

Parameter	Type	Introduction
<ssid>	unsigned char *	A null terminated string containing the SSID name of the network.
<security_type>	rtw_security_t	RTW_SECURITY_OPEN: Open Security RTW_SECURITY_WPA_TKIP_PSK: WPA Security RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers WEP security is NOT IMPLEMENTED. It is NOT SECURE!
<password>	unsigned char *	A byte array containing the cleartext security key for the network.
<ssid_len>	int	The length of the SSID in bytes.
<password_len>	int	The length of the security_key in bytes.
<channel>	int	802.11 channel number.

Note:

Make sure the Wi-Fi is enabled before invoking this function. (wifi_on())

If a STA interface is active when this function is called, the softAP will start on the same channel as the STA, using this channel provided.

20.2.6.4 wifi_get_associated_client_list

Get the associated clients with SoftAP.

Parameter	Type	Introduction
<client_list_buffer>	int *	The location where the client list will be stored.
<buffer_length>	unsigned short	The buffer length.

20.2.6.5 wifi_enable_forwarding

Enable packets forwarding in AP mode.

Parameter None.

20.2.6.6 wifi_disable_forwarding

Disable packets forwarding in AP mode.

Parameter None.

20.2.7 Custom IE APIs

API	Introduction
<wifi_add_custom_ie>	Setup custom IE list.
<wifi_update_custom_ie>	Update the item in Wi-Fi custom IE list.
<wifi_del_custom_ie>	Delete Wi-Fi custom IE list.

Note: These three APIs are only effective on beacon, probe request and probe response frames.

20.2.7.1 wifi_add_custom_ie

Setup custom IE list.

Parameter	Type	Introduction
<cus_ie>	void *	Pointer to Wi-Fi CUSTOM IE list.
<ie_num>	int	The number of Wi-Fi CUSTOM IE list.

Note: ¢ K ƛ ă ! t L Ŏ l· y΄ Ω ú ō Ŝ Ŝ È Ŝ Ŏ dz ú Ŝ Ř ú ø ƛ Ŏ Ŝ ō Ŝ Ŧ 2 NJ Ŝ Ř Ŝ f Ŝ ú ƛ y΄ ∃ ú K Ŝ LJNJ Ŝ Ø ƛ 2 dz ă

20.2.7.2 wifi_update_custom_ie

Update the item in Wi-Fi custom IE list.

Parameter	Type	Introduction
<cus_ie>	void *	Pointer to Wi-Fi CUSTOM IE list.
<ie_index>	int	The number of Wi-Fi CUSTOM IE list.

20.2.7.3 wifi_del_custom_ie

Delete Wi-Fi custom IE list.

Parameter None.

20.2.8 Wi-Fi Setting APIs

API	Introduction
<wifi_get_mac_address>	Retrieves the current Media Access Control (MAC) address (or Ethernet hardware address) of t 802.11 device.
<wifi_get_ap_bssid>	Get Ŏ ž ý˝ ý Ŝ Ŏ ű Ŝ Ř ! t Ω ă . { { L 5 Φ
<wifi_get_ap_info>	Get the SoftAP information.
<wifi_set_country>	Set the country code to driver to determine the channel set.
<wifi_get_sta_max_data_rate>	Retrieved STA mode max. data rate.
<wifi_get_rssi>	Retrieved the latest RSSI value.
<wifi_register_multicast_address>	Register interest in a multicast address. Once a multicast address has been registered, all packets detected on the medium destined for that address are forwarded to the host. Otherwise they are ignored.
<wifi_unregister_multicast_address>	Unregister interest in a multicast address. Once a multicast address has been unregistered, all packets detected on the medium destined for that address are ignored.
<wifi_set_mib>	Setup the adaptive mode. You can replace this weak function by the same name function to setup adaptive mode you can.
<wifi_set_country_code>	Setup country code. You can replace this weak function by the same name function to setup country code you want.
<wifi_set_tdma_param>	Set TDMA parameters.
<wifi_get_setting>	Get current Wi-Fi setting from driver.
<wifi_show_setting>	Show the network information stored in the rtw_wifi_setting_t structure.
<wifi_set_network_mode>	Set the network mode according to the data rate it supported.
<wifi_get_network_mode>	Get the network mode.
<wifi_get_antenna_info>	Get antenna information.
<wifi_set_ch_deauth>	Set flag for concurrent mode wlan1 issue deauth when channel switched by wlan0.

20.2.8.1 wifi_get_mac_address

Retrieve the current Media Access Control (MAC) address (or Ethernet hardware address) of the 802.11 device.

Parameter	Type	Introduction
<mac>	char *	Point to the result of the mac address will be get.

20.2.8.2 wifi_get_ap_bssid

Get Ŏ ž ý˝ ý Ŝ Ŏ ű Ŝ Ř ! t Ω ă . { { L 5 Φ

Parameter	Type	Introduction
<bssid>	unsigned char*	The location where the AP BSSID will be stored.

20.2.8.3 wifi_get_ap_info

Get the SoftAP information.

Parameter	Type	Introduction
<ap_info>	rtw_bss_info_t*	The location where the AP info will be stored.
<security>	rtw_security_t *	The security type.

20.2.8.4 wifi_set_country

Set the country code to driver to determine the channel set.

Parameter	Type	Introduction
<country_code>	rtw_country_code_t	Specify the country code.

20.2.8.5 wifi_get_sta_max_data_rate

Retrieved STA mode max. data rate.

Parameter	Type	Introduction
<inidata_rate>	_u8*	Max data rate.

20.2.8.6 wifi_get_rssi

Retrieved the latest RSSI value.

Parameter	Type	Introduction
<pRSSI>	int *	Points to the integer to store the RSSI value gotten from driver.

20.2.8.7 wifi_register_multicast_address

Register interest in a multicast address.

Once a multicast address has been registered, all packets detected on the medium destined for that address are forwarded to the host, otherwise they are ignored.

Parameter	Type	Introduction
<mac>	rtw_mac_t *	Ethernet MAC address.

20.2.8.8 wifi_unregister_multicast_address

Unregister interest in a multicast address.

Once a multicast address has been unregistered, all packets detected on the medium destined for that address are ignored.

Parameter	Type	Introduction
<mac>	rtw_mac_t *	Ethernet MAC address.

20.2.8.9 wifi_set_mib

Setup the adaptive mode. You can replace this weak function by the same name function to setup adaptivity mode you want.

Parameter None.

20.2.8.10 wifi_set_country_code

Setup country code. You can replace this weak function by the same name function to setup country code you want.

Parameter None.

20.2.8.11 wifi_set_tdma_param

Set TDMA parameters.

Parameter	Type	Introduction
<slot_period>	unsigned char	We separate TBTT into 2 or 3 slots. If we separate TBTT into 2 slots, then slot_period s be larger or equal to 50ms. It means 2 slot period is slot_period, slot_period If we separate TBTT into 3 slots, then slot_period should be less or equal to 33ms. It means 3 slo period is 100 - 2 * slot_period, slot_period, slot_period.
<rfon_period_len_1>	unsigned char	RF on period of slot 1.
<rfon_period_len_2>	unsigned char	RF on period of slot 2.
<rfon_period_len_3>	unsigned char	RF on period of slot 3.

20.2.8.12 wifi_get_setting

Get current Wi-Fi setting from driver.

Parameter	Type	Introduction
<ifname>	const char *	The WLAN interface name WLAN0_NAME WLAN1_NAME
<pSetting>	rtw_wifi_setting_t *	Points to the rtw_wifi_setting_t structure to store the Wi-Fi setting gotten from driver.

20.2.8.13 wifi_show_setting

Show the network information stored in the rtw_wifi_setting_t structure.

Parameter	Type	Introduction
<ifname>	const char *	The WLAN interface name, can be WLAN0_NAME or WLAN1_NAME.
<pSetting>	rtw_wifi_setting_t *	Points to the rtw_wifi_setting_t structure which information is gotten by wifi_get_setting

20.2.8.14 wifi_set_network_mode

Set the network mode according to the data rate it supported. Driver works in BGN mode in default after driver initialization. This function is used to change wireless network mode for station mode before connecting to AP.

Parameter	Type	Introduction
<mode>	rtw_network_mode_t	Network mode to set. RTW_NETWORK_B RTW_NETWORK_BG RTW_NETWORK_BGN

20.2.8.15 wifi_get_network_mode

Get the network mode. Driver works in BGN mode in default after driver initialization. This function is used to get the current wireless network mode for station mode.

Parameter	Type	Introduction
<pmode>	rtw_network_mode_t *	Network mode to get.

20.2.8.16 wifi_get_antenna_info

Get antenna information.

Parameter	Type	Introduction
<antenna>	unsigned char *	Points to store the antenna value gotten from driver. 0: main

		1: aux

20.2.8.17 wifi_set_ch_deauth

Set flag for concurrent mode wlan1 issue_deauth when channel switched by wlan0.

Parameter	Type	Introduction
<enable>	_u8	0: Disable 1: Enable

Usage: wifi_set_ch_deayth(0) -> wlan0 wifi_connect -> wifi_set_ch_deauth(1)

20.2.9 Wi-Fi Indication APIs

API	Introduction
<wifi_manager_init>	Initialize Realtek WiFi API System.
<wifi_indication>	WLAN driver indicate event to upper layer through wifi_indication.
<init_event_callback_list>	Initialize the event callback list.
<wifi_reg_event_handler>	Register the event listener.
<wifi_unreg_event_handler>	Un-register the event listener.

20.2.9.1 wifi_manager_init

Initialize Realtek WiFi API System

Initialize the required parts of the software platform, such as worker, event registering and semaphore

Initialize the RTW API thread which handles the asynchronous event

Parameter: None.

20.2.9.2 wifi_indication

WLAN driver indicates event to upper layer through wifi_indication().

Parameter	Type	Introduction
<event>	rtw_event_indicate_t	An event reported from driver to upper layer application. Refer to rtw_event_indicate_t enum.
<buf>	char *	If it is not NUL, buf is a pointer to the buffer for message string.
<buf_len>	int	The length of the buffer.
<flags>	int	Indicate some extra information, sometimes it is 0.

Note:

If upper layer application triggers additional operations on receiving of wext_wlan_indicate, please strictly check task stack size usage (by using uxTaskGetStackHighWaterMark()), and tries not to share the same stack with WLAN driver if remaining stack is not available for the following operations.

Using semaphore to notice another thread instead of handing event directly in wifi_indication().

20.2.9.3 init_event_callback_list

Initialize the event callback list.

Parameter: None.

Note: Make sure this function has been invoked before using the event handler related mechanism.

20.2.9.4 wifi_reg_event_handler

Register the event listener.

Parameter	Type	Introduction
<event_cmds>	unsigned int	The event command number indicated.
<handler_func>	rtw_event_handler_t	The callback function which will receive and process the event.
<handler_user_data>	void *	User specific data that will be passed directly to the callback function

Note: Set the same even_cmds with empty handler_func will unregister the event_cmds.

20.2.9.5 wifi_unreg_event_handler

Un-register the event listener.

Parameter	Type	Introduction
<event_cmds>	unsigned int	The event command number indicated.
<handler_func>	rtw_event_handler_t	The callback function which will receive and process the event.

20.2.10 eFuse Writing APIs

API	Introduction
<wifi_set_mac_address>	This function sets the current Media Access Control (MAC) address of the 802.11 de

Note: Ŝ C dz ă Ŝ ø NJ ƛ ű ƛ y˝ Ǝ NJ Ŝ f ł· ű Ŝ Ř ! t L Ŏ ł· y˝ ž y˝ f è ō Ŝ Ŏ ł· f f Ŝ Ř ø K Ŝ y˝ ű nk, eFuse ž f ű ł· Ǝ Ŝ writing can only be operated one time, so be careful to do eFuse writing.

20.2.10.1 wifi_set_mac_address

Sets the current Media Access Control (MAC) address of the 802.11 device.

Parameter	Type	Introduction
<mac>	char *	Wi-Fi MAC address.

20.3 Fast Connection

This section illustrates the principle of fast connection and how to implement user's own fast connection code.

Fast connection is used to reconnect with AP automatically after Wi-Fi initialized, the principle is to store the AP information in Flash and reconnect to AP after Wi-Fi initialized.

20.3.1 Implement

20.3.1.1 AP Information Storage

User should implement a function to write AP information to Flash, just like demo function wlan_wrtie_reconnect_data_to_flash() in example code. In this function, you should reserve some space for AP information, and write the AP information to the reserved space in a defined data format. The address of the function must be assigned to the global variable p_write_reconnect_ptr. After Wi-Fi connection success, if p_write_reconnect_ptr points to a valid address, wlan_wrtie_reconnect_data_to_flash() will be called.

Note: The path of example source code is SDK/component/common/example/example_wlan_fast_connect/example_wlan_fast_connect.c

20.3.1.2 Reconnection

User should implement his own function to read AP information from Flash and connect to AP, just like demo function wlan_init_done_callback() in example code. The address of the function must be assigned to the global variable p_wlan_init_done_callback. This global variable should be defined before WiFi initializing. After Wi-Fi initialized, if p_wlan_init_done_callback points to a valid address, this function will be called.

20.3.1.3 Fast Connection Data Erase

User should implement his own function to erase fast connection data, just like demo function Erase_Fastconnect_data() in example code.

20.3.2 APIs

API	Introduction
<wlan_wrtie_reconnect_data_to_flash>	Wi-Fi connection indication trigger this function to save current Wi-Fi profile in Flash.
<wlan_init_done_callback>	After Wi-Fi initialization done, WLAN driver calls this function to check whether auto connection is required. This function reads previous saved WLAN profile from Flash and execute connection.

20.3.2.1 wlan_wrtie_reconnect_data_to_flash

Wi-Fi connection indications trigger this function to save current Wi-Fi profile in Flash. Write AP information to Flash.

Parameter	Type	Introduction
<data>	u8 *	Data will be written to Flash
<len>	uint32_t	Length of data
return	int	Status value: 0: write is ok -1: write is failed

Note: This is only a demo API, user should define his own API.

20.4 WPS APIs

API	Introduction
<wps_start>	Start WPS enrollee process
<wps_stop>	Stop WPS enrollee process

20.4.1 wps_start

Start WPS enrollee process.

Parameter	Type	Introduction
<wps_config>	u16	WPS configure method WPS_CONFIG_DISPLAY WPS_CONFIG_KEYPAD WPS_CONFIG_PUSHBUTTON
<pin>	char *	PIN number. Can be set to NULL if using WPS_CONFIG_PUSHBUTTON.
<channel>	u8	Channel. Currently unused, can be set to 0.
<ssid>	char *	Target network SSID. Can be set to NULL if no target network specified.

Note:

Before invoking this function, the Wi-Fi should be enabled by calling wifi_on().
Make sure CONFIG_ENABLE_WPS is enabled in platform_opts.h. After calling wps_start(), the longest time of WPS is 120s. You can call wps_stop() to quit WPS.

20.4.2 wps_stop

Stop WPS enrollee process.

Parameter None.

Note: Make sure CONFIG_ENABLE_WPS is enabled in platform_opts.h.

21 Liquid Crystal Display Controller (LCDC)

AmebaD LCDC supports Thin Film Transistor (TFT) color display. It provides I8080 MCU interface (8-bit bus width) and RGB interface (6-bit bus width), andsupports RGB565 data format

This chapter introduces how to use LCDC I/F to control LCD module.

21.1 Interface

The available LCDC interfaces for different IC packages are shown in Table 21-1.

Table21-1 LCDC I/F supported for different packages

IC Package	MCU 8bit	RGB 6bit
RTL8722D/RTL8722CS		
RTL8721D/RTL8721CS/RTL8720D/RTL8720CS		

The data format supported by each interface is shown in Table 21-2. In order to display normally, the data format of LCM and LCDC interface should match.

Table21-2 LCDC I/F data format

Interface	Data Format	Comment
MCU 8bit	Count 0 1 2 3 4 X RS 0 1 1 1 1 X D7 C7 P1R4 P1G2 P2R4 P2G2 X D6 C6 P1R3 P1G1 P2R3 P2G1 X D5 C5 P1R2 P1G0 P2R2 P2G0 X D4 C4 P1R1 P1B4 P2R1 P2B4 X D3 C3 P1R0 P1B3 P2R0 P2B3 X D2 C2 P1G5 P1B2 P2G5 P2B2 X D1 C1 P1G4 P1B1 P2G4 P2B1 X D0 C0 P1G3 P1B0 P2G3 P2B0 X P1R4 Pixel1/Red_bit4	2 transfers/pixel, RGB565
RGB 6bit	Count 0 1 2 3 4 X D5 P1R4 P1G5 P1B4 P2R4 P2G5 X D4 P1R3 P1G4 P1B3 P2R3 P2G4 X D3 P1R2 P1G3 P1B2 P2R2 P2G3 X D2 P1R1 P1G2 P1B1 P2R1 P2G2 X D1 P1R0 P1G1 P1B0 P2R0 P2G1 X D0 P1G0 P2G0 X P1R4 Pixel1/Red_bit4	3 transfers/pixel, RGB565

21.2 Resolution

The maximum resolution supported by each interface is listed in Table 21-3.

Table21-3 Maximum supported resolution

Interface	Display Type	Max. Support Resolution
MCU 8bit	Static Display	1024*1024
	Dynamic Display	645*645(30fps)
	Dynamic Display	912*912(30fps)
RGB 6bit	Dynamic Display	600*400(60fps)

For RGB LCD, the typical frame rate is 60fps.

In order to support RGB LCD with higher resolution than parameters in Table 21-1, users have to set the frame rate lower. The maximum frame rate for a RGB LCD is calculated as follows:

Max. dot clock: MAX_DOT_CLK = system_clock / 2 = 100M / 2 = 50MHz

Width, height, HBP, HFP, VBP, VFP are specified by LCD datasheet.

For 16-bit I/F mode:

```
image_size = MAX_DOT_CLK / (3 * (width + HBP + HFP) * (VBP + VFP + height * (HBP + HFP);
F = 50M / (width * height + (width + HBP + HFP) * (VBP + VFP + height * (HBP + HFP)) / 3;
```

When frame rate is lower than 30fps, the screen flickering may happen. Users should evaluate the visual artifacts when setting frame rate lower.

21.3 Pinmux

The pin assignments of LCDC are listed in Table 21-4.

Table 21-4 LCDC pin assignments

Port Name	MCU 8-bit	RGB 16-bit	LED I/F
PB[19]	-	-	-
PB[18]	-	-	-
PB[11]	-	-	-
PB[10]	-	-	-
PB[9]	-	-	-
PB[8]	-	-	-
PA[25]	-	-	-
PA[26]	-	-	-
PA[28]	D[7]	-	-
PA[30]	D[6]	-	-
PB[0]	D[5]	D[5]	D[5]
PA[31]	D[4]	D[4]	D[4]
PA[24]	D[3]	D[3]	D[3]
PA[23]	D[2]	D[2]	D[2]
PA[20]	D[1]	D[1]	D[1]
PA[19]	D[0]	D[0]	D[0]
PB[20]	TE/VSYNC	VSYNC	-
PB[21]	RS	-	-
PB[22]	RD	HSYNC	LAT
PB[23]	WR	DCLK	DCLK
PB[28]	CS	ENABLE	OE

21.4 LCDC APIs

21.4.1 MCU Function

21.4.1.1 LCDC_MCUStructInit

Items	Description
Introduction	Initializes the parameters in the LCDC_MCUInitStruct with default values.
Parameters	LCDC_MCUInitStruct: pointer to a LCDC_MCUInitTypeDef structure which is initialized.
Return	N/A

Note: LCDC_MCUInitStruct contains the MCU configurable parameters for LCDC, which determine that the MCU I/F is in CPU mode or DMA mode, the plane size and some timing control. The parameters in this structure should be set according to LCD module datasheet.

21.4.1.2 LCDC_MCUInit

Items	Description
Introduction	Initializes the LCDC peripheral according to the specified parameters in LCDC_MCUInitStruct.
Parameters	LCDCx: where LCDCx can be LCDC. LCDC_MCUInitStruct: pointer to a LCDC_MCUInitTypeDef structure that contains the configuration infor
Return	N/A

21.4.1.3 LCDC_MCUIOWriteCmd

Items	Description
Introduction	Writes command to LCD module via MCU I/F.
Parameters	LCDCx: where LCDCx can be LCDC. Cmd: the command to transmit.
Return	N/A

21.4.1.4 LCDC_MCUIOWriteData

Items	Description
Introduction	Writes data to LCD module via MCU I/F.
Parameters	LCDCx: where LCDCx can be LCDC. Data: the data to transmit.
Return	N/A

21.4.1.5 LCDC_MCUIOReadData

Items	Description
Introduction	Reads data from LCD module via MCU I/F.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	The read value

21.4.1.6 LCDC_MCUDMATrigger

Items	Description
Introduction	Triggers to transfer data of one frame from DMA buffer to LCDC TransFIFO.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	N/A

21.4.2 RGB Function

21.4.2.1 LCDC_RGBStructInit

Items	Description
Introduction	Initializes the parameters in the LCDC_RGBInitStruct with default values.
Parameters	LCDC_RGBInitStruct: pointer to a LCDC_RGBInitTypeDef structure which is initialized.
Return	N/A

Note: LCDC_RGBInitStruct contains the RGB configurable parameters for LCDC, which determine that the RGB IF is 6-bit, DE or HV mode, the plane size, the refresh frequency and VSYNC & HSYNC control. The parameters in this structure should be set according to LCD module datasheet.

21.4.2.2 LCDC_RGBInit

Items	Description
Introduction	Initializes the LCDC peripheral according to the specified parameters in the LCDC_RGBInitStruct.
Parameters	LCDCx: where LCDCx can be LCDC. LCDC_RGBInitStruct: pointer to a LCDC_RGBInitTypeDef structure that contains the configuration inform
Return	N/A

21.4.3 LED Function

21.4.3.1 LCDC_LEDStructInit

Items	Description
Introduction	Initializes the parameters in the LCDC_LEDInitStruct with default values.
Parameters	LCDC_LEDInitStruct: pointer to an LCDC_LEDInitTypeDef structure which will be initialized.
Return	N/A

21.4.3.2 LCDC_LEDInit

Items	Description
Introduction	Initializes the LCDC peripheral according to the specified parameters in the LCDC_LEDInitStruct.
Parameters	LCDCx: where LCDCx can be LCDC. LCDC_LEDInitStruct: pointer to an LCDC_LEDInitTypeDef structure which will be initialized.
Return	N/A

21.4.4 Common Function

21.4.4.1 LCDC_DMAModeConfig

Items	Description
Introduction	Configures LCDC DMA burst size.
Parameters	LCDCx: where LCDCx can be LCDC. BurstSize: DMA burst size; unit is 64 bytes.
Return	N/A

Note:

If BurstSize = 1, the actual burstsize = 1x64 bytes; if BurstSize = 2, the actual burstsize = 2x64 bytes; ...

The parameter BurstSize is not more than 8. The recommended value is 2.

21.4.4.2 LCDC_DMAImageBaseAddrConfig

Items	Description
Introduction	Configures image base address.
Parameters	LCDCx: where LCDCx can be LCDC. ImgBaseAddr: the buffer address.
Return	N/A

21.4.4.3 LCDC_INTConfig

Items	Description
Introduction	Enables or disables the specified LCDC interrupts.
Parameters	LCDCx: where LCDCx can be LCDC. LCDC_IT: specifies the LCDC interrupt sources to be enabled or disabled. This parameter can be any combination of the following values: LCDC_IT_DMAUNDFW: DMA FIFO underflow interrupt LCDC_IT_FRDN: LCD refresh done interrupt LCDC_IT_LINE: line interrupt LCDC_IT_IO_TIMEOUT: IO write/read timeout interrupt LCDC_IT_FRM_START: Frame Start interrupt NewState: new state of the specified LCDC interrupts. This parameter can be ENABLE or DISABLE.
Return	N/A

21.4.4.4 LCDC_GetINTStatus

Items	Description
Introduction	Gets LCDC interrupt status.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	Interrupt status

21.4.4.5 LCDC_ClearINT

Items	Description
Introduction	Clears the LCDC's interrupt pending bits.
Parameters	LCDC_IT: specifies the interrupt to be cleared. This parameter can be any combination of the following v LCDC_IT_LINE: line interrupt LCDC_IT_FRDN: refresh frame done interrupt LCDC_IT_DMAUNDFW: DMA FIFO under flow interrupt LCDC_IT_IO_TIMEOUT: IO write/read timeout interrupt LCDC_IT_FRM_START: Frame Start interrupt
Return	N/A

21.4.4.6 LCDC_Cmd

Items	Description
Introduction	Enables or disables the LCDC.
Parameters	LCDCx: where LCDCx can be LCDC. NewState: new state of the LCDC. This parameter can be: ENABLE or DISABLE.
Return	N/A

Note: When NewState is DISABLE, during the period of valid line (VTIMING =valid data), the disable is performed after the last valid line has transferred. If you want to disable the LCDC instantly, use the API LCDC_InsDisable(). During the other periods, the disable is performed instantly.

21.4.4.7 LCDC_InsDisable

Items	Description
Introduction	Disables the LCDC instantly.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	N/A

21.4.4.8 LCDC_DeInit

Items	Description
Introduction	De-initializes the LCDC.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	N/A

Note: when disabling LCDC instantly, all interrupts are cleared and disabled.

21.5 How to Use LCDC

Table21-5 lists the typical application scenarios of LCDC.

Table21-5 Typical application scenario of LCDC

Interface	Data Mode	LCD GRAM	Ameba-D Frame Buffer	Application
MCU	I/O mode			Static Display
	DMA Trigger-mode			Static Display
	DMA Auto-mode (VSYNC/TE mode)			dynamic Display
RGB	DMA Auto-mode (DE/HV mode)			dynamic Display

21.5.1 MCU Interface

21.5.1.1 I/O Mode

To use the LCDC MCU interface I/O mode, the following steps are mandatory.

(1) Configure the LCDC pinmux according Table21-4.

For example, in order to use PA[19] as LCDC pin, call the following function. It is the same for other LCDC pins.

```
Pinmux_Config (_PA_19, PINMUX_FUNCTION_LCD);
```

(2) Initialize the LCDC_MCUInitStruct variable.

a) Use the following function to initialize LCDC_MCUInitStruct variable with default parameters.

```
LCDC_MCUStructInit (LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
```

b) Change other parameters according to LCM datasheet, such as bus interface mode, Data/WR/RD/CS/RS pulse polarity, WR/RD pulse width.

(3) Initialize the LCDC using the initialized structure in step (2).

```
LCDC_MCUInit (LCDC_TypeDef * LCDCx, LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
```

(4) Enable the LCDC using the function LCDC_Cmd().

(5) Send commands and parameters to LCM using the function LCDC_MCUIOWriteCmd()/LCDC_MCUIOWriteData() to initialize the LC module.

(6) After LCM is initialized, call LCDC_MCUIOWriteCmd()/LCDC_MCUIOWriteData()/LCDC_MCUIOReadData() to send commands, write data to LCM or read data from LCM to drive LCD displaying.

21.5.1.2 Trigger DMA Mode

To use the LCDC MCU interface trigger DMA mode, the following steps are mandatory.

(1) Configure the LCDC pinmux according Table21-4.

(2) Configure LCDC to work in MCU I/O mode, then send commands and parameters to LCM to initialize LCD. After that, user need to send command to inform LCM to start receiving data before starting frame data.

(3) Initialize the LCDC_MCUInitStruct variable.

a) Configure the LCDC_MCUInitStruct parameter to Trigger DMA mode.

```
LCDC_MCUStructInit (LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
LCDC_MCUInitStruct->LCDC_MCUMode = LCDC_MCU_DMA_MODE;
LCDC_MCUInitStruct->LCDC_MCUDMAMode = LCDC_TRIGGER_DMA_MODE;
```

b) Change other parameters according to LCM datasheet, such as LCD width/height, 8080 mode, Data/WR/RD/CS/RS pulse polarity, WR/RD pulse width.

(4) Initialize the LCDC using the initialized structure in step (3).
    LCDC_MCUInit (LCDC_TypeDef * LCDCx, LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
(5) Configure the LCDC DMA parameters.
    a) Configure the LCDC DMA burst size using LCDC_DMAModeConfig().
    b) Allocate DMA buffer and assign the address to LCDC using LCDC_DMAImageBaseAddrConfig().
(6) Enable LCDC interrupt using the function LCDC_INTConfig(), if needed.
(7) Enable the LCDC using the function LCDC_Cmd().
(8) Trigger one frame transfer using the function LCDC_MCUDMATrigger(), and update the frame buffer to change the display effect.

21.5.1.3 VSYNC Mode

To use the LCDC MCU I/F VSYNC mode, the following steps are mandatory.
(1) Configure the LCDC pinmux according Table 21-4.
(2) Configure LCDC to work in MCU I/O mode, then send commands and parameters to LCM to make LCD work in VSYNC mode.
(3) Initialize the LCDC_MCUInitStruct variable.
    a) Configure the LCDC_MCUInitStruct parameter corresponding to VSYNC mode.
       LCDC_MCUStructInit (LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
       LCDC_MCUInitStruct->LCDC_MCUMode = LCDC_MCU_DMA_MODE;
       LCDC_MCUInitStruct->LCDC_MCUDMAMode = LCDC_AUTO_DMA_MODE;
       LCDC_MCUInitStruct->LCDC_MCUSyncMode = LCDC_MCU_SYNC_WITH_VSYNC;
    b) Change other parameters according to LCM datasheet, such as VSYNC pulse polarity/pulse width/idle period, LCD width/height.
(4) Initialize the LCDC using the initialized structure in step (3).
    LCDC_MCUInit (LCDC_TypeDef * LCDCx, LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
(5) Configure the LCDC DMA parameters.
    a) Configure the LCDC DMA burst size using the function LCDC_DMAModeConfig().
    b) Allocate DMA buffer and assign the address to LCDC using the function LCDC_DMAImageBaseAddrConfig().
(6) Enable the specified LCDC interrupt using the function LCDC_INTConfig(), if needed.
(7) Enable the LCDC using the function LCDC_Cmd().
(8) The LCDC can transfer frame data to LCM automatically synchronized with the VSYNC signal to LCM, and you can update the frame buffer to change the display.

21.5.1.4 TE Mode

To use the LCDC MCU IinterfaceF TE mode, the following steps are mandatory.
(1) Configure the LCDC pinmux according Table 21-4.
(2) Configure LCDC to work in MCU I/O mode, then send commands and parameters to LCM to work in MCU TE mode.
(3) Initialize the LCDC_MCUInitStruct variable.
    a) Configure the LCDC_MCUInitStruct parameter corresponding to VSYNC mode.
       LCDC_MCUStructInit (LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
       LCDC_MCUInitStruct->LCDC_MCUMode = LCDC_MCU_DMA_MODE;
       LCDC_MCUInitStruct->LCDC_MCUDMAMode = LCDC_AUTO_DMA_MODE;
       LCDC_MCUInitStruct->LCDC_MCUSyncMode = LCDC_MCU_SYNC_WITH_TE;
    b) Change other parameters if needed, such as TE pulse polarity, TE Delay, LCD width/height.
(4) Initialize the LCDC using the initialized structure in step (3).
    LCDC_MCUInit (LCDC_TypeDef * LCDCx, LCDC_MCUInitTypeDef * LCDC_MCUInitStruct);
(5) Configure the LCDC DMA parameters.
    a) Configure the LCDC DMA burst size using the function LCDC_DMAModeConfig().
    b) Allocate DMA buffer and assign the address to LCDC using the function LCDC_DMAImageBaseAddrConfig().
(6) Enable the specified LCDC interrupt using the function LCDC_INTConfig(), if needed.
(7) Enable the LCDC using the function LCDC_Cmd().
(8) The LCDC can transfer frame data to LCM automatically synchronized with TE signal from LCM, and you can update the frame buffer to change the display.

21.5.2 RGB Interface

21.5.2.1 DE Mode

To use the LCDC RGB interface DE mode, the following steps are mandatory.
(1) Configure the LCDC pinmux according Table 21-4.
(2) Configure LCM parameters through SPI or other interfaces if needed.
(3) Initialize the LCDC_RGBInitStruct variable.
   a) Configure the LCDC_RGBInitStruct parameter corresponding to DE mode.
      LCDC_RGBStructInit (LCDC_RGBInitTypeDef * LCDC_RGBInitStruct);
      LCDC_RGBInitStruct->LCDC_RGBSyncMode = LCDC_RGB_DE_MODE;
   b) Change other parameters if needed, such as Date/ENABLE/VSYNC/HSYNC pulse polarity, DCLK active edge, VFP, VBP, VSW, HBP, HFP, HSW, refresh frequency, LCD width/height.
(4) Initialize the LCDC using the initialized structure in step (3).
      LCDC_RGBInit (LCDC_TypeDef* LCDCx, LCDC_RGBInitTypeDef* LCDC_RGBInitStruct);
(5) Configure the LCDC DMA parameters.
   a) Set burst size using the function LCDC_DMAModeConfig()
   b) Set DMA FIFO under flow mode and error data using the functions LCDC_DMAUnderFlowModeConfig() and LCDC_DMAUnderFlowModeConfig().
   c) Allocate DMA buffer and assign the address to LCDC using the function LCDC_DMAImageBaseAddrConfig().
(6) Enable the specified LCDC interrupt using the function LCDC_INTConfig(), if needed.
(7) Enable the LCDC using the function LCDC_Cmd().
(8) The LCDC can transfer frame data to LCM automatically according to the refresh frequency, and you can update the frame buffer address to the display.

21.5.2.2 HV Mode

To use the LCDC RGB I/F HV mode, the following steps are mandatory.
(1) Configure the LCDC pinmux according Table 21-4.
(2) Configure LCM parameters through SPI or other interfaces if needed.
(3) Initialize the LCDC_RGBInitStruct variable.
   a) Configure the LCDC_RGBInitStruct parameter corresponding to HV mode.
      LCDC_RGBStructInit (LCDC_RGBInitTypeDef * LCDC_RGBInitStruct);
   b) Change other parameters if needed, Date/ENABLE/VSYNC/HSYNC pulse polarity, DCLK active edge, VFP, VBP, VSW, HBP, HFP, refresh frequency, LCD width/height, 6bit parallel I/F mode, refresh frequency.
(4) Initialize the LCDC using the initialized structure in step (3).
      LCDC_RGBInit (LCDC_TypeDef* LCDCx, LCDC_RGBInitTypeDef* LCDC_RGBInitStruct)
(5) Configure the LCDC DMA parameters
   a) Set burst size using the function LCDC_DMAModeConfig().
   b) Set DMA FIFO under flow mode and error data using the functions LCDC_DMAUnderFlowModeConfig() and LCDC_DMAUnderFlowModeConfig().
   c) Allocate DMA buffer and assign the address to LCDC using the function LCDC_DMAImageBaseAddrConfig().
(6) Enable the specified LCDC interrupt using the function LCDC_INTConfig(), if needed.
(7) Enable the LCDC using the function LCDC_Cmd().
(8) The LCDC can transfer frame data to LCM automatically according to the refresh frequency, and you can update the frame buffer address to the display.

21.5.3 LED Interface

To use the LCDC LED interface mode, the following steps are mandatory.
(1) Configure the LCDC pinmux according Table 21-4.
(2) Initialize the LCDC_RGBInitStruct variable.
   a) Configure the LCDC_LEDInitStruct parameter corresponding to LED interface mode
      LCDC_LEDStructInit (LCDC_LEDInitTypeDef * LCDC_LEDInitStruct)
   b) Change other parameters if needed, such as color channel, color numbers, timing (latch start time, latch pulse width, OE width), refresh frequency, LED width/height.

(3) Initialize the LCDC using the initialized structure in step (2).
LCDC_LEDInit (LCDC_TypeDef* LCDCx, LCDC_LEDInitTypeDef * LCDC_LEDInitStruct)
(4) Configure the LCDC DMA parameters
a) Set burst size using the function LCDC_DMAModeConfig().
b) Allocate DMA buffer and assign the address to LCDC using the function LCDC_DMAImageBaseAddrConfig().
(5) Enable the specified LCDC interrupt using the function LCDC_INTConfig(), if needed.
(6) Enable the LCDC using the function LCDC_Cmd().
(7) The LCDC can transfer frame data to LED array board automatically according to the refresh frequency, and you can update the buffer to change the display.

21.6 GUI

emWin is the embedded GUI solution. It can be adapted to any size, either physical or virtual display, not dependent of the display controller. Making it a professional GUI for the embedded market, usable for multiple different scenarios.

Realtek has reached an emWin Pro Buyout Agreement with SEGGER. The LICENSED SOFTWARE, which is located in component\common\ui\emwin, is available in object code form for Realtek customers who have been authorized.

21.6.1 Authorization

In order to be authorized to use emWin software, you need to follow these steps:
(1) Read *emWin_Software_License_Agreement.pdf* carefully under the path component\common\ui. The document is a binding, legal agreement between Realtek and you (either an individual or a legal entity). It explains the terms and conditions that you should accept and agree.
(2) If you do not accept and agree to this Agreement, do not unzip emwin.zip and do not use any of the LICENSED SOFTWARE. If you do
[illegible] *emWin_Software_License_Agreement.pdf*

21.6.2 emWin Software

After [illegible]

Directory	Sub-directory	Description
emWinLibrary	/	Contains the basic contents in emWin PRO package
	Config	The files in this sub-directory are template configuration files for reference to adapt for different SoC and LCD module. You should customize GUIConf.c and LCDConf.c according to their devices. The details of these two files can be found in *Chapter 38 Configuration* of *UM03001_emWin5.pdf*. For AmebaD, we provide porting samples of EVB LCM with MCU or RGB I/F. The samples are located in component\common\ui\emwin\Sample\rtl8721d\Config. [illegible] files to adapt to new LCM. The files in this sub-directory are used to adapt to for different LCD module and different scenarios.
	Doc	*UM03001_emWin5.pdf* is the User Guide and Reference Manual of emWin.
	GUI_X	GUI_X_FreeRTOS.c is the configuration of the timing routines, the debugging routines and the kernel interface routines
	Include	The header files of emWin
	Lib	The object code of emWin
	Tool	The available tools which are useful during UI development, such as Font Converter, GUIBuilder
Sample	/	These are porting samples of AmebaD driving EVB LCM. MCU I/F LCM on EVB: ILI9488 TFT-LCD module (320*480, MCU-8bit I/F) RBG I/F LCM on EVB: CX043DTL-04 TFT-LCD module (480*272, RGB-16bit I/F)
Third_Party	/	It contains the TrueType Font for emWin, which should be used if fonts need to be scalable at run-time. This is the adapted version of FreeType font library from David Turner, Robert Wilhelm and Werner Lemberg

21.6.3 How to Adaptto LCM

As mentioned above, we provide porting samples of AmebaD driving EVB LCM in component\common\ui\emwin\Sample\rtl8721d. Table 21-6 demonstrate how to customize configuration files to adapt new LCM.

Table21-6 Customizing configuration files

Configuration file	Step	Description	Note
GUIConf.c	(1) Modify the value of macro GUI_NUMBYTES to define the size of memory block. (2) Define macro PSRAM_BUF_USED in LCDConf.h to place the memory block into PSRAM. Otherwise, undefine it to place the memory block into SRAM.	Provide emWin with the function GUI_X_Config() which is responsible for assigning a memory block to the memory management system.	
LCDConf_MCU_8bit_eval.c	(1) Set XSIZE_PHYS and YSIZE_PHYS to the horizontal and vertical resolution of LCM. (2) Implement the functions illustrated in Table 21-7. void lcd_mcu_write(int x0, int y0, int x1, int y1, void *buf, U32 color) void lcd_mcu_read (int x0, int y0, int x1, int y1, void *buf) void lcd_init(void)	Used for MCU I/F LCM. As an example, these functions are implemented in Sample\rtl8721d\hal\hal_mcu_lcd.c. You should modify this file according to the flow of Initialize, Read/Write pixel specified by datasheet of LCM.	The LCDConf_MCU_8bit_eval.c adopts GUI MCU driver porting by Realtek, and it applies to display controllers with indirect interface, such as MCU I/F LCM.
LCDConf_RGB_6bit_eval.c	(1) Set XSIZE_PHYS and YSIZE_PHYS to the horizontal and vertical resolution of LCM. (2) Define macro PSRAM_BUF_USED in LCDConf.h to place the VRAM buffer into PSRAM. Otherwise, undefine it to place the VRAM buffer into SRAM. (3) Implement the functions illustrated in Table 21-8. void lcd_init(unsigned int width, unsigned int height, unsigned int bufAddr) void lcd_set_dma_addr(unsigned int bufAddr)	Used for RGB I/F LCM. As an example, these functions are implemented in Sample\rtl8721d\hal\hal_rgb_lcd.c. You should modify this file according to the timing parameters specified by datasheet of LCM.	The LCDConf_RGB_6bit_eval.c adopts GUI LIN driver provided by emWin package and it applies to display controllers with linear video memory accessible via direct interface, such as RGB I/F LCM.
LCDConf_SPI_eval.c	(1) Set XSIZE_PHYS and YSIZE_PHYS to the horizontal and vertical resolution of LCM. (2) Implement the functions illustrated in Table 21-9. void lcd_spi_write (int x0, int y0, int x1, int y1, void *buf, U32 color) void lcd_spi_read (int x0, int y0, int x1, int y1, void *buf) void lcd_init(void)	Used for SPI I/F LCM. As an example, these functions are implemented in Sample\rtl8721d\hal\hal_spi_lcd.c. You should modify this file according to the flow of Initialize, Read/Write pixel specified by datasheet of LCM.	lcd_spi_read function is not necessary. The LCDConf_SPI_eval.c adopts GUI SPI driver porting by Realtek, and it applies to display controllers with indirect interface, such as SPI I/F LCM.

Table21-7 Functions for MCU I/F LCM

Function	lcd_mcu_write	lcd_mcu_read	lcd_init
Introduction	Writes pixels to LCD module.	Reads pixels from LCD module.	Initialize LCD controller and LCM.
Parameters	x0: the start column of area to be updated y0: the start row of area to be updated x1: the end column of area to be updated y1: the end row of area to be updated buf: pointer to a buffer which is used to store pixel data of the area defined by (x0,y0,x1,y1)	x0: the start column of area to be read y0: the start row of area to be read x1: the end column of area to be read y1: the end row of area to be read buf: pointer to a buffer which is used to store pixel data read from the area defined by (x0,y0,x1,y1)	N/A

	color: If buf is NULL, fill the area with the same color		
Return	N/A	N/A	N/A

Table 21-8 Functions for RGB I/F LCM

Function	lcd_init	lcd_set_dma_addr
Introduction	Initialize LCD controller	Set the address of VRAM. It is necessary when Multiple Bufferin Virtual Screen function is used
Parameters	width: horizontal resolution height: vertical resolution bufAddr: the start address of VRAM	bufAddr: the start address of VRAM
Return	N/A	N/A

Table 21-9 Functions for SPI I/F LCM

Function	lcd_spi_write	lcd_spi_read	lcd_init
Introduction	Writes pixels to LCD module.	Reads pixels from LCD module.	Initialize LCD controller and LCM.
Parameters	x0: the start column of area to be updatec y0: the start row of area to be updated x1: the end column of area to be updated y1: the end row of area to be updated buf: pointer to a buffer which is used to store pixel data of the area defined by (x0,y0,x1,y1) color: If buf is NULL, fill the area with the same color	x0: the start column of area to be reac y0: the start row of area to be read x1: the end column of area to be read y1: the end row of area to be read buf: pointer to a buffer which is used tc store pixel data read from the area defined by (x0,y0,x1,y1)	N/A
Return	N/A	N/A	N/A

Note:
To adapt the driver to a display controller not supported currently, user should

- Refer to Chapter 33.7.26 GUIDRV_Template in *UM03001_emWin5.pdf*
- Adapt the routines _SetPixelIndex() and _GetPixelIndex() in component\common\ui\emwin\emWinLibrary\Config\GUIDRV_Template.c
- Optimize some operation in GUIDRV_Template.c to improve drawing speed

21.6.4 How to Adapt to Touch Panel

We provide porting samples of AmebaD driving EVB touch panel in component\common\ui\emwin\Sample\rtl8721d. The following steps demonstrate how to customize configuration files to adapt a new touch panel.

(1) Copy your touch driver to the folder component\common\ui\emwin\Sample\rtl8721d\hal
The touch driver needs to provide an initialization function and functions to get the horizontal and vertical coordinates of the touch point.

(2) Modify hal_touch.h/hal_touch.c
[illegible] hal_touch.h [illegible]
Implement the following functions in hal_touch.c
void hal_touch_init(void)
int hal_touch_measureX(void)
int hal_touch_measureY (void)

Function	hal_touch_init	hal_touch_measureX	hal_touch_measureY
Introduction	Initialize the touch panel.	Measure the x coordinate of touch point	Measure the y coordinate of touch poin
Parameters	N/A	N/A	N/A
Return	N/A	X coordinate of touch point	Y coordinate of touch point

21.6.5 How to Use emWin in SDK

To build and run emWin demo in SDK, you need to follow these steps:

(1) Enable emWin compile
KM4 make menuconfig -> MENUCONFIG FOR CHIP CONFIG -> GUI Config -> Enable GUI and select emwin.

(2) Enable PSRAM if needed
By default, the VRAM buffer, which is necessary for RGB I/F LCM, is located in PSRAM. So you should enable PSRAM by setting psram_dev_config.psram_dev_enable to TRUE in rtl8721dhp_intfcfg.c.
If you want to put the VRAM buffer in RAM instead of PSRAM, you should modify LCDConf.h as follows:

```
#undef PSRAM_BUF_USED
```

(3) By default, the configuration files of RGB I/F LCM (hal_rgb_lcd.c, LCDConf_RGB_6bit_eval.c) is compiled.
To use MCU I/F LCM, you should modify project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\make\ui\emwin\Sample\Makefile to compile hal_mcu_lcd.c and LCDConf_MCU_8bit_eval.c instead.

(4) Add your own code or run emWin demo code.
The emWin demo code is located in project\realtek_amebaD_va0_example\example_sources\LCDC\GUI_demo\emWin. The ReadMe.txt in each demo folder demonstrates how to use it.

22 PSRAM

Pseudo Static Random Access Memory (PSRAM) is used for high speed transmission of data stream, and is suitable for audio codec. RTL8721D uses PSRAM controller to communicate with PSRAM. PSRAM is in KM4 platform, so only KM4 can access it.

The features of PSRAM are:

- Density: 32Mbit
- Address Mapping: 0x0200_0000 ~ 0x0240_0000
- Clock rate: 50MHz
- Double Data Rate (DDR)
- 16/32/64/128 bytes burst access
- Half sleep mode and deep power-down mode

22.1 Throughput

PSRAM supports direct access and DMA access.

Table 22-1 PSRAM Throughput

Access mode		Write		Read	
		Theory	Test	Theory	Test
Direct Access	Cache off	200Mbps	< 160Mbps	177.78Mbps	< (32)/(180ns+360ns) = 59.26Mbps
	Cache on	200Mbps	160Mbps (CS is high for 40ns between two transmits)	556.52Mbps	< (32*8)/(460ns+360ns) = 312Mbps
DMA		731.43Mbps	711.11Mbps	721.13Mbps	589.18Mbps

Note:

- When Cache off:
  - Read or write operation only access 32 bits once with header and delay
  - Instruction execution time also need to take into consideration
  - The 360ns in test is caused by hardware characteristic. Command from CPU to PSRAM controller needs 152ns to sync, PSRAM controller controlling PHY circuit to work needs 82us, PHY input read data to CPU needs 126ns
- When Cache on:
  - Read operation reads 32 bytes (cache line) once
  - Write operation writes 32 bits once, but the interval between two write operations is greatly reduced
  - The 360ns in test is caused by hardware characteristic. Command from CPU to PSRAM controller needs 152ns to sync, PSRAM controller controlling PHY circuit to work needs 82us, PHY input read data to CPU needs 126ns
- DMA sets burst length 128 bytes and disables cache.
  - For DMA write, DAM moving data to PSRAM FIFO and PSRAM controller writing FIFO data to PSRAM slave can work simultaneously so the interval between two burst write operations is small.
  - For DMA read, similar operation as DMA write is not supported, so the interval between two burst read operations is large.
  - The test data above takes variable initial latency and 3 clocks initial latency for example.

Table 22-2 PSRAM Throughput theoretical calculation

Item	Writing 32 bits	Reading 32 bits
Header + delay	[3 + (31)] * 20ns = 100ns	[3 + (31)] * 20ns = 100ns
Data transmit period	2 * 20ns = 40ns	16 * 20ns = 320ns
Hardware hold	1 * 20ns = 20ns	2 * 20ns = 40ns
Total without considering instruction execution time	100ns + 40ns + 20ns = 160ns	100ns + 320ns + 40ns = 460ns
Throughput theoretical value	32/160ns = 200Mbps	(32*8)/460ns = 556.52Mbps

22.2 Power Management

When entering KM4 powergate mode, PSRAM will be reset and its memory will be lost. If you want to retain the PSRAM, KM4 powergate mode is not supported to use. If you want to keep the PSRAM and save power, KM4 clockgate mode is supported.

Table 22-3 PSRAM power management

Retention	Options	Power Consumption (uA)	Comment
No	KM4 PG PSRAM_LDOFF	<30	PSRAM cannot keep no memory retention applications are not supported.
Yes	KM4 CG PSRAM_LDON PSRAM Half Sleep mode	About 325	

22.3 How to Use PSRAM

22.3.1 Initializing PSRAM

Before accessing PSRAM, you should enable PSRAM power, initialize PSRAM controller and PSRAM slave to synchronize the related parameters.

In SDK, you should set psram_dev_enable in rtl8721dhp_intfcfg.c. If the chip works in the environment with large fluctuations in temperature, you should set psram_dev_cal_enable to enable calibration function. If you want to keep the PSRAM and save power when KM4 enters sleep mode, you should also set psram_dev_retention to enable PSRAM retention.

```
PSRAMCFG_TypeDef psram_dev_config = {
    .psram_dev_enable = TRUE,                   //enable psram
    .psram_dev_cal_enable = TRUE,               //enable psram calibration function
    .psram_dev_retention = TRUE,                //enable psram retention
};
```

22.3.2 Adding BSS/TEXT/DATA Section into PSRAM Region

Table 22-4 lists how to add a BSS/TEXT/DATA section into PSRAM region.

Table 22-4 Add a BSS/TEXT/DATA section into PSRAM region

Section	Description	Operation
BSS	To put a BSS section in PSRAM, add PSRAM_BSS_SECTION before the buffer definition.	`PSRAM_BSS_SECTION u32 PSRAM_Testbuf[1024];`

TEXT	To put a TEXT section in PSRAM, add PSRAM_TEXT_SECTION before the function definition.	`PSRAM_TEXT_SECTION`
		`VOID Test_Function(VOID)`
		`{`
		`  u32 i = 0;`
		`  for (i = 0; i < 10; i++) {`
		`  DBG_8195A("test\r\n");`
		`  }`
		`}`

DATA	To put a DATA section in PSRAM, add PSRAM_DATA_SECTION or PSRAM_RODATA_SECTION before the data definition.	`PSRAM_DATA_SECTION` `u8 Test_Data = FALSE;`

After the operation, rebuild KM4 project.

22.3.3 Allocating Heap from PSRAM

Before allocating heap, initializing PSRAM must be completed, then follow the steps below.

(1) Confirm the heap size you want in Psram_reserve.c.

```
#define configTOTAL_PSRAM_HEAP_SIZE      (0x200000)
```

(2) Use void *Psram_reserve_malloc(int size) to allocate a heap or use void *Psram_reserve_calloc(int num, int size) to allocate several consecutive spaces. Both functions return a pointer to the start address of the allocation. Use void Psram_reserve_free(void *mem) to free the allocation.

Note: The PSRAM_BSS_SECTION will be cleared only in the function void app_init_psram(void) in rtl8721dhp_app_start.c

22.4 PSRAM [illegible] Write Back Policy Change Note

22.4.1 Cache Policy Change

To prevent PSRAM cell charge leak from frequent access, the cache policy is chang[illegible] enhance memory access performance and efficiency.

22.4.2 Scope

All of chips stacking PSRAM inside on AmebaD series: RTL8721CSM, RTL8722CSM, RTL8721DM, and RTL8722DM

22.4.3 Notice

On [illegible] the [illegible] for multiple [illegible] access by different sources, e.g. CPU, serial ports and peripherals.

Follow the described instructions on the following paragraphs while manipulating PSRAM for DMA application and heap usage

As cache line of AmebaD cache is 32 bytes, and cache operations are all based on cache line. So the buffer size and buffer starting address should be 32 bytes alignment to avoid synchronization issues.

22.4.3.1 DMA Buffer Definition

When using Psram_reserve_malloc() to allocate a space for DMA buffer, our API has realized 32 bytes of the header address of the memory allocated (This approach is recommended to definite DMA Buffer)

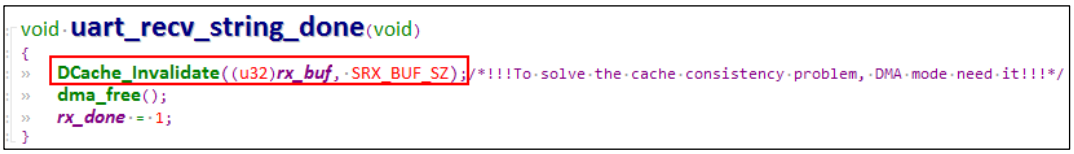
When allocating a space for DMA buffer from PSRAM by array, [illegible] ing address 32 bytes alignment.

```
char rx_buf[RX_BUF_SIZE] __attribute__((aligned(32)));
```

22.4.3.2 DMA Operation

The following steps should be added when executing DMA Rx/Tx.

Operation	Step
DMA Rx	(1) Prepare Rx Buffer (2) Do DCache_CleanInvalidate() to avoid cache data write back during DMA Rx (3) Do DMA Rx config (4) Trigger DMA Rx interrupt

	(5) Do DCache_Invalidate() in Rx Done Handler to clean old data (take example_sources/USI_UART/raw/usi_uart_stream_dma/src/main.c for example, uart_recv_string_done() is DMA Rx Done Interrupt Handler)  (6) CPU reads Rx Buffer
DMA Tx	(1) CPU prepares Tx buffer data (2) Do DCache_CleanInvalidate() for Tx buffer to synchronize the data (3) Do DMA tx config (4) Trigger DMA tx interrupt

```
void uart_recv_string_done(void)
{
    DCache_Invalidate((u32)rx_buf, SRX_BUF_SZ);/*!!!To solve the cache consistency problem, DMA mode need it!!!*/
    dma_free();
    rx_done = 1;
}
```

In SDK, only the example of one time xxx_GDMA_Init one time transmission is illustrated. The (2) step is included in xxx_GDMA_Init by default.

If you need multitime DMA TRx with only one time xxx_GDMA_Init, DCache_CleanInvalidate() should be called every time before DMA transmission starts.

```
BOOL USI_UARTTXGDMA_Init(
    u8 USIIndex,
    GDMA_InitTypeDef *GDMA_InitStruct,
    void *CallbackData,
    IRQ_FUN CallbackFunc,
    u8 *pTxBuf,
    int TxCount
    )
{
    u8 GdmaChnl;

    assert_param(GDMA_InitStruct != NULL);

    DCache_CleanInvalidate((u32)pTxBuf, TxCount);
```

```
BOOL USI_UARTRXGDMA_Init(
    u8 USIIndex,
    GDMA_InitTypeDef *GDMA_InitStruct,
    void *CallbackData,
    IRQ_FUN CallbackFunc,
    u8 *pRxBuf,
    int RxCount
    )
{
    u8 GdmaChnl;
    USI_TypeDef *pUSIx;

    assert_param(GDMA_InitStruct != NULL);

    DCache_CleanInvalidate((u32)pRxBuf, RxCount);
```

22.4.3.3 Heap Usage

If you want to allocate heap in PSRAM, void *Psram_reserve_malloc(int size) should be used to allocate a heap for the starting address 32 bytes alignment. It returns a pointer to the starting address of the allocation. void Psram_reserve_free(void *mem) is used to free the allocation

23 MPU and Cache

23.1 Functional description

23.1.1 MPU

Memory Protection Unit (MPU) is used to provide Hardware Protection by Software definition. Ameba provides mpu_region_config structure to include the region memory attribute. Default attribute of all KM0 & KM4 SRAM are write-through.

Table 23-1 shows member variables of the mpu_region_config structure.

Table 23-1 mpu_region_config structure

Member Variable Name	Type	Description
region_base	uint32_t	MPU region base, 32 bytes aligned
region_size	uint32_t	MPU region size, 32 bytes aligned
xn	uint8_t	Execute Never attribute MPU_EXEC_ALLOW: Allows program execution in this region MPU_EXEC_NEVER: Does not allow program execution in this region
ap	uint8_t	Access permissions MPU_PRIV_RW: Read/write by privileged code only MPU_UN_PRIV_RW: Read/write by any privilege level MPU_PRIV_R: Read only by privileged code only MPU_PRIV_W: Read only by any privilege level
sh	uint8_t	Shareability for Normal memory MPU_NON_SHAREABLE: Non-shareable MPU_OUT_SHAREABLE: Outer shareable MPU_INR_SHAREABLE: Inner shareable
attr_idx	uint8_t	Memory attribute indirect index This parameter can be a value of 0~7, the detailed attribute is defined in mpu_init() and is customized. The typical definition is as following: 0: MPU_MEM_ATTR_IDX_NC: Defines memory attribute of Normal memory with non-cacheable 1: MPU_MEM_ATTR_IDX_WT_T_RA: Defines memory attribute of Normal memory with write-through transient, read allocation 2: MPU_MEM_ATTR_IDX_WB_T_RWA: Defines memory attribute of Normal memory with write-back transient, read and write allocation 3~7: MPU_MEM_ATTR_IDX_DEV: Defines memory attribute of Device memory with non-gathering, non-recording, non-early Write Acknowledge

Table 23-2 shows how to set a MPU region.

Table 23-2 How to set a MPU region

Steps	Description
New variable and structure	Variable to store MPU entry index Structure mpu_region_config to store the region memory attribute
Allocate a free MPU entry	Call mpu_entry_alloc()
Set region memory attribute	Set structure of region memory attribute
Configure MPU region memory attribute	Call mpu_region_cfg()

23.1.2 Cache

Cache is used to improve the CPU performance of data access. AmebaD Cache supports Enable/Disable, Flush and Clean Operations. Table 23-3 lists.

Table 23-3 Enable, Disable, Flush and Clean operation supported by Cache

Operation	Description	I-Cache	D-Cache
Enable/Disable	Enable or Disable Cache function		
Flush(Invalidate)	Flush Cache D-Cache can be flushed by address Can be used after DMA Rx, and CPU read DMA data from DMA buffer for D-Cache		
Clean	Clean D-Cache D-Cache will be write back to memory D-Cache can be cleaned by address Can be used before DMA Tx, after CPU write data to DMA buffer for D-Cache		

23.2 MPU APIs

23.2.1 mpu_init

Items	Description
Introduction	Initialize MPU region memory attribute to typical value
Parameters	N/A
Return	N/A

23.2.2 mpu_set_mem_attr

Items	Description
Introduction	Change MPU region memory attribute
Parameters	attr_idx: region memory attribute index, which can be 0~7. mem_attr: region memory attributes.
Return	N/A

23.2.3 mpu_region_cfg

Items	Description
Introduction	Configure MPU region memory attribute
Parameters	region_num: KM0: 0~3 KM4_NS: 0~7 KM4_S: 0~3 pmpu_cfg: pointer to an mpu_region_config structure which has been configured
Return	N/A

23.2.4 mpu_entry_free

Items	Description
Introduction	Free MPU entry
Parameters	entry_index KM0: 0~3 KM4_NS: 0~7 KM4_S: 0~3
Return	N/A

23.2.5 mpu_entry_alloc

Items	Description
Introduction	Allocate a free MPU entry
Parameters	N/A
Return	MPU entry index KM0: 0~3 KM4_NS: 0~7 KM4_S: 0~3 Fail -1

23.3 Cache APIs

23.3.1 ICache_Enable

Items	Description
Introduction	Enable I-Cache
Parameters	N/A
Return	N/A

23.3.2 ICache_Disable

Items	Description
Introduction	Disable I-Cache
Parameters	N/A
Return	N/A

23.3.3 ICache_Invalidate

Items	Description
Introduction	Invalidate I-Cache
Parameters	N/A
Return	N/A

23.3.4 DCache_IsEnabled

Items	Description
Introduction	Check D-Cache enabled or not
Parameters	N/A
Return	D-Cache enable status: 1: Enable 0: Disable

23.3.5 DCache_Enable

Items	Description
Introduction	Enable D-Cache
Parameters	N/A
Return	N/A

23.3.6 DCache_Disable

Items	Description
Introduction	Disable D-Cache
Parameters	N/A
Return	N/A

23.3.7 DCache_Invalidate

Items	Description
Introduction	Invalidate D-Cache by address
Parameters	Address Invalidated address (aligned to 32 byte boundary) Bytes Size of memory block (in number of bytes)
Return	N/A

23.3.8 DCache_Clean

Items	Description
Introduction	Clean D-Cache by address
Parameters	Address Clean address (aligned to 32 byte boundary) Bytes size of memory block (in number of bytes) Note: Address set 0xFFFFFFFF is used to clean all D-Cache
Return	N/A

23.3.9 DCache_CleanInvalidate

Items	Description
Introduction	Clean and invalidate D-Cache by address
Parameters	Address Clean and invalidated address (aligned to 32 byte boundary) Bytes size of memory block (in number of bytes) Note: Address set 0xFFFFFFFF is used to clean and flush all D-Cache
Return	N/A

23.4 How to Define a Non-cacheable Data Buffer

To define a data buffer with non-cacheable attribute, you should add SRAM_NOCACHE_DATA_SECTION before the buffer definition.

```
SRAM_NOCACHE_DATA_SECTION u8 noncache_buffer[DATA_BUFFER_SIZE];
```

24 Audio Codec Controller Guide

24.1 Audio Codec

AmebaD audio codec (ACC) is often used to play and record audio data. It is a stereo audio codec with stereo headphone amplifiers, as well as 2-way inputs and 1-way stereo/mono output that are programmable to single-ended or differential.

AmebaD audio codec integrates anti-pop circuit for audible pop noise cancellation, MIC bias circuit for MIC power supply and programmable MIC boost ability.

AmebaD audio codec transmits record data to or receives playback data from platform through SPORT interface by means of specific serial interface (SI), AmebaD platform configures and drives audio codec.

AmebaD also provides external I²S interface for audio application extension, which supports the highest 384kHz sampling frequency.

24.1.1 Diagram

The diagram of ACC is shown in Fig 24-1.

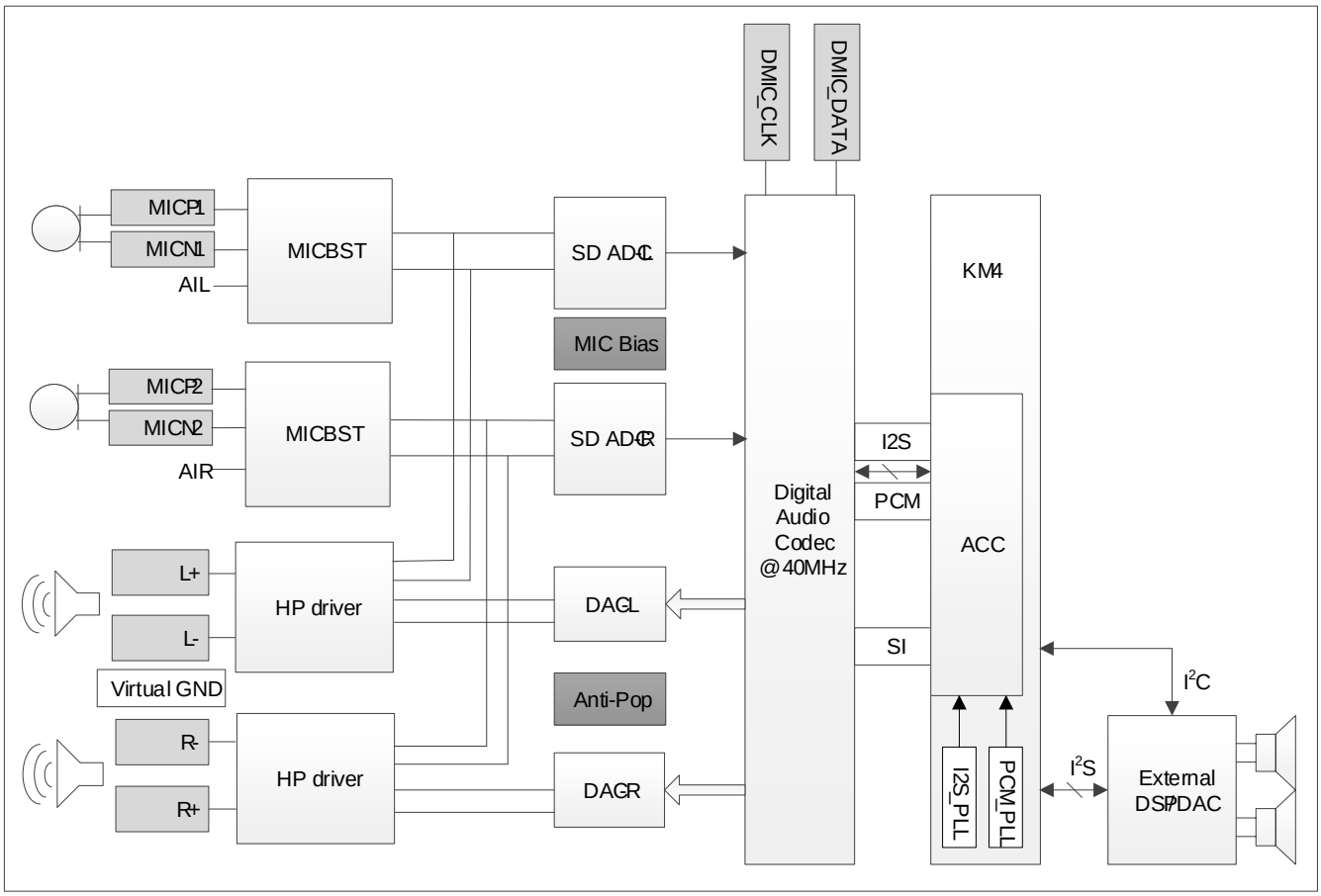


Fig 24-1 Audio codec diagram

24.1.2 Features

Mono and stereo channel
8-bit, 16-bit and 24-bit sample bits
8k, 16k, 32k, 48k, 96k, 44.1k, 48k and 88.2k sample rate
I2S, left justify, PCM mode A, PCM mode B, PCM mode A and PCM mode B data format
Anti-pop function to reduce audible pop
Programmable MIC boost gain
Programmable gain in ADC and DAC path
Three line-out output modes: cap-less, differential and single-end
Three input ways: line-in, AMIC-in and DMIC-in

24.1.3 Application Mode

Audio codec supports three input ways: line-in, AMIC-in and DMIC-in, but only supports one output way: line-out.

24.1.3.1 Line-out

Line-out has no amplifier to amplify output voice. Line-out supports three output modes: cap-less, differential and single-ended. User can select the wanted mode by setting the related registers.

Line-out cap-less mode
In this mode, the N-end of L/R channel outputs common-level voltage, while P-end drives the available analog audio signal. When earphone inserts into jack, the ground must short with N-end output for audio signal de-couple. That is why the ground is called virtual ground.

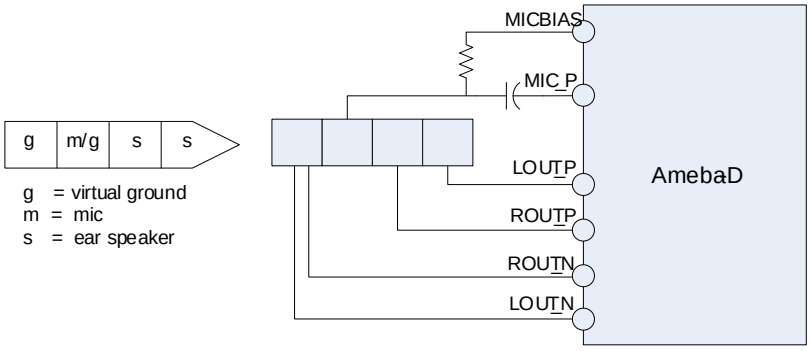


Fig 24-2 Cap-less mode connection with headphone jack

Line-out differential mode
In this mode, both N-end and P-end drive the available analog audio signal. User should select the differential jack and earphone accordingly.

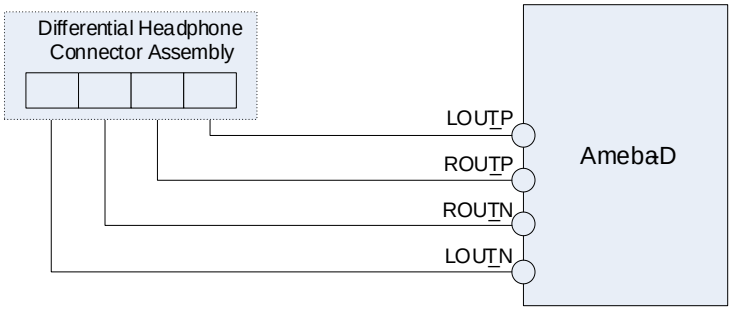


Fig 24-3 Differential mode connection with headphone jack

Line-out single-ended mode

In this mode, board circuit designer need to place a capacitor to the P-end output path for analog audio signal pick-up. No N-end output is required.

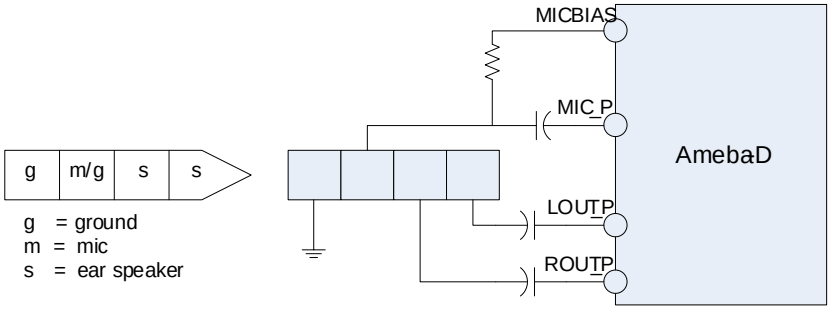


Fig24-4 Single-ended mode connection with headphone jack

24.1.3.2 Line-in

Line-in has no preamplifier, its input signal often has a large output power. It often connects to the audio output of equipment such as electric guitar, electronic organ and synthesizer

Connect the left channel of line-in signal to AUX_L, and the right channel to AUX_R accordingly.

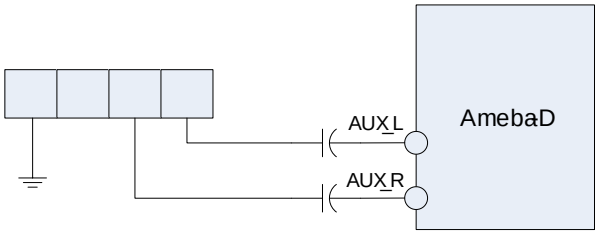


Fig24-5 Line-in mode connection

24.1.3.3 AMIC-in

Analog microphone (AMIC) records audio data, it has preamplifier, its input signal often has a low output power. AMIC-in supports differential mode and single-ended mode.

AMIC-in single-ended mode

Connect MIC_P with single-ended analog microphone, while MICBIAS provides the microphone bias voltage.

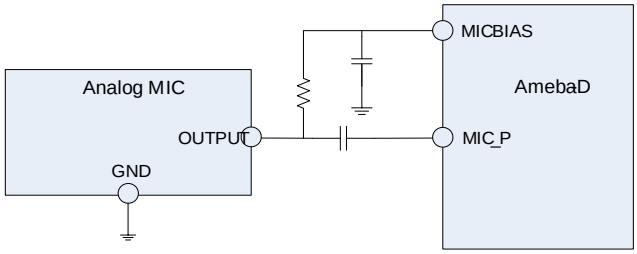


Fig24-6 AMIC-in single-ended mode connection

AMIC-in differential mode

Connect MIC_P/MIC_N with differential analog microphone, while MICBIAS provides the microphone bias voltage.

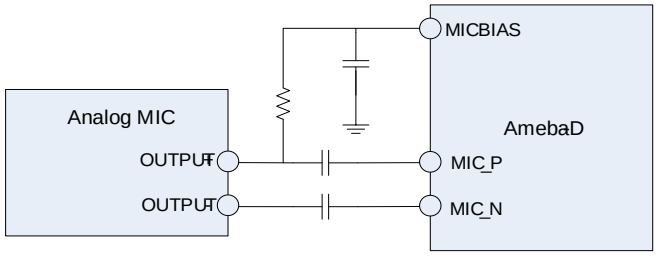


Fig24-7 AMIC-in differential mode connection

24.1.3.4 DMIC-in

Digital microphone (DMIC) records audio data, it is integrated with ADC internal, and can directly output digital signal. DMIC-in supports mono mode and stereo mode.

DMIC-in mono mode

Tie the L/R of digital microphone to ground or VDD if only one digital microphone is placed.

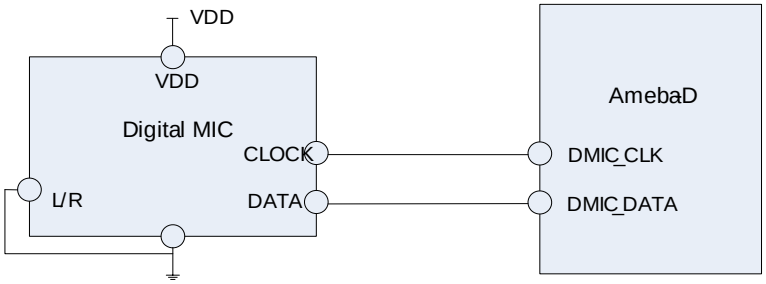


Fig24-8 DMIC-in mono mode connection

DMIC-in stereo mode

Tie the L/R of two digital microphones to ground and VDD respectively if stereo microphone is needed. The two microphones share the DMIC_DATA according to the rising/falling edge.

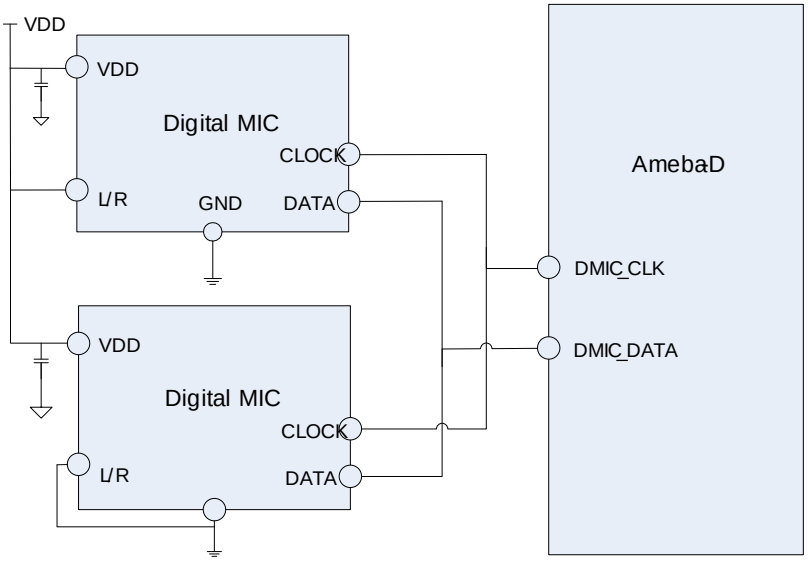


Fig24-9 DMIC-in stereo mode connection

24.2 Audio Codec Controller

AmebaD audio codec controller (ACC) is the bridge between host audio buffers and audio codec module. It is used for audio codec input/output control.

AmebaD audio codec controller uses GDMA to move data, and transfers audio data to or from audio codec module via SPORT, and configures audio codec module via SI.

The diagram of ACC is shown in Fig 24-10.

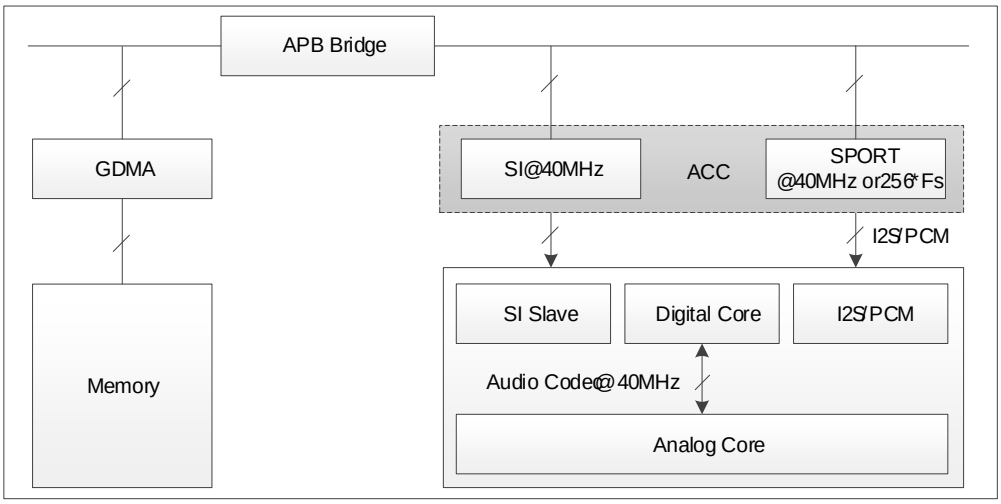


Fig 24-10 Audio codec controller diagram

24.2.1 Features

- Mono and stereo channel
- 8-bit, 16-bit and 24-bit sample bits
- I2S, left justify, PCM mode A, PCM mode B, PCM mode A and PCM mode B data format
- Mandatory or optional sample rate which audio codec module decides to support
- GDMA for data moving
- Data loopback between SDI and SDO

24.2.2 Control Interface

There are two control interfaces: SPORT interface and SI interface

SPORT interface

Audio codec transfers audio data via SPORT interface sequentially according to setting. It supports multiple data format, such as I2S, left justify, PCM mode A, PCM mode B, PCM mode A and PCM mode B.

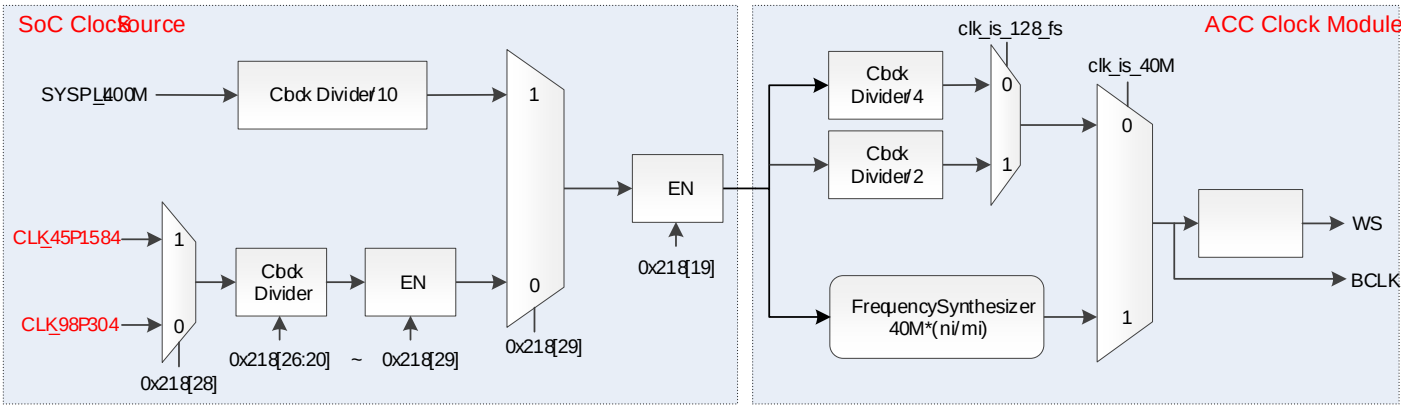
SI interface

Audio codec needs to configure the related analog and digital parameters before transfering audio data. SI interface is used to configure codec parameters. It can read or write codec register.

For more details about AC and ACC, refer to UM0400 AmebaD User Manual.

24.3 Audio PLL

The ACC clock architecture is shown in Fig 24-11.



Note: CLK_45P1584 is clock derived from PCM PLL internal, while CLK_98P304 from I²S PLL.

Fig24-11 ACC clock architecture

24.3.1 Diagram

The ACC clock diagram is illustrated in Fig24-12.

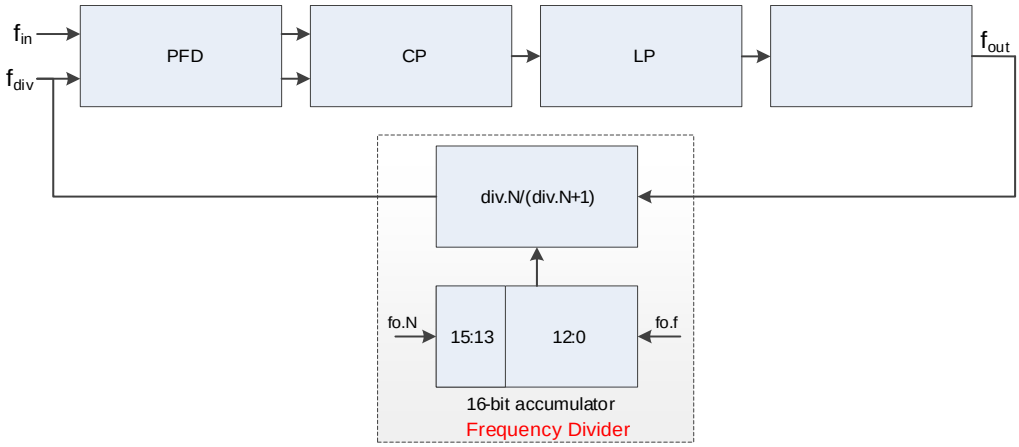


Fig24-12 ACC clock diagram

$$f_{out} = \frac{f_{in}}{div.N + \frac{fo.f}{2^{16}} + \frac{fo.N}{2^{13}}}$$

Where f_{in} is derived from the clock of crystal, while div.N, fo.N and fo.f from the register settings. Therefore, the resolution of f_{out} equals to $f_{in}/2^{16}$.

The relationship between crystal clock and f_{in} is listed in Table24-1.

Table24-1 Relationship between crystal clock and f_{in}

Crystal Clock (MHz)	f_{in} (MHz)	Divider
40	10	4
25	12.5	2
13	13	1
19.2	9.6	2
20	10	2
26	13	2
38.4	9.6	4

17.664	8.832	2
16	8	2
14.318	14.318	1
12	12	1
52	13	4
48	12	4
27	13.5	2
24	12	2

24.3.2 Operation Mode

24.3.2.1 Auto Mode

In auto mode, PLL circuit automatically sets the parameters (div.N, fo.N, fo.f, etc.) to output clock with the exact frequency, i.e. 196.608MHz (=98.304MHz x 2) is offered by I²S PLL while 180.6336MHz (=45.1584MHz x 4) by PCM PLL.

24.3.2.2 Manual Mode

If fine tuning PLL clock is required, users could switch the I²S or PCM PLL to manual mode. In this mode, users could control the output frequency of I²S PLL max. ±100ppm around 196.608MHz or PCM PLL max. ±100ppm around 180.6336MHz by calling the corresponding APIs. As the adjusted step of f_{out} equals to $f_{in}/2^{16}$, f_{in} decides the exact ppm per step, i.e. $(f_{in}/2^{16})/196.608$ or $(f_{in}/2^{16})/180.6336$).

The relationship between f_{in} and ppm per step is listed in Table 24-2.

Table 24-2 Relationship between f_{in} and ppm per step

f_{in} (MHz)	I²S ppm per step	PCM ppm per step
10	0.78	0.84
12.5	0.97	1.06
13	1.01	1.10
9.6	0.75	0.81
10	0.78	0.84
13	1.01	1.10
9.6	0.75	0.81
8.832	0.69	0.75
8	0.62	0.68
14.318	1.11	1.21
12	0.93	1.01
13	1.01	1.10
12	0.93	1.01
13.5	1.05	1.14
12	0.93	1.01

24.4 Audio Codec APIs

24.4.1 PLL APIs

API	Introduction
<PLL_Div>	Divider to generate 256*fs or 128*fs clock
<PLL_Sel>	Selects I²S PLL or PCM PLL
<PLL_I2S_Set>	Enables or disables I²S PLL
<PLL_PCM_Set>	Enables or disables PCM PLL
<PLL_I2S_ClkTune>	Tunes I²S PLL output faster or slower, or resets it to auto mode

<PLL_PCM_ClkTune>	Tunes PCM PLL output faster or slower, or resets it to auto mode

24.4.1.1 PLL_Div

Parameter	Type	Introduction
	u32	Divider to generate 256*fs or 128*fs clock.

24.4.1.2 PLL_Sel

Parameter	Type	Introduction
<sel>	u32	Selects I2S PLL if fs=8/16/32/48/96kHz Selects PCM PLL if fs=44.1/88.2kHz

24.4.1.3 PLL_I2S_Set

Parameter	Type	Introduction
<new_state>	u32	Enables or disables I2S PLL

24.4.1.4 PLL_PCM_Set

Parameter	Type	Introduction
< new_state >	u32	Enables or disables PCM PLL

24.4.1.5 PLL_I2S_ClkTune

Parameter	Type	Introduction
<ppm>	u32	Required fine tuning ppm value
<action>	U32	Faster or slower, or reset to auto mode

24.4.1.6 PLL_PCM_ClkTune

Parameter	Type	Introduction
<ppm>	u32	Required fine tuning ppm value
<action>	U32	Faster or slower, or reset to auto mode

24.4.2 SPORT API

API	Introduction
<AUDIO_SP_StructInit>	Fills each SP_StructInit member with its default value
<AUDIO_SP_Init>	Initializes the audio SPORT registers according to the specified parameters in SP_InitStruct
<AUDIO_SP_TxStart>	Starts or stops SPORT Tx path
<AUDIO_SP_RxStart>	Starts or stops SPORT Rx path
<AUDIO_SP_TdmaCmd>	Enables or disables SPORT TX DMA request
<AUDIO_SP_RdmaCmd>	Enables or disables SPORT Rx DMA request
<AUDIO_SP_SetWordLen>	Sets the AUDIO SPORT word length
<AUDIO_SP_GetWordLen>	Gets the AUDIO SPORT word length
<AUDIO_SP_SetMonoStereo>	Sets the AUDIO SPORT channel number
<AUDIO_SP_TXGDMA_Init>	Initializes GDMA peripheral for Tx data
<AUDIO_SP_RXGDMA_Init>	Initializes GDMA peripheral for Rx data

24.4.2.1 AUDIO_SP_StructInit

Fills each SP_StructInit member with its default value

Parameter	Type	Introduction
<SP_InitStruct>	SP_InitTypeDef*	SP_InitTypeDef structure that contains the configuration information for the specified audio SPORT peripheral

24.4.2.2 AUDIO_SP_Init

Initializes the audio SPORT registers according to the specified parameters in SP_InitStruct.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDe	The base address of audio SPORT peripheral
<SP_InitStruct>	SP_InitTypeDef*	SP_InitTypeDef structure that contains the configuration information for the specified AUDIO SPORT peripheral

24.4.2.3 AUDIO_SP_TxStart

Starts or stops SPORT Tx path.

If playing with audio codec, it starts SPORT Tx path.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDe	The base address of audio SPORT peripheral
<NewState>	u32	State (enable or disable) of the SPORT Tx

24.4.2.4 AUDIO_SP_RxStart

Starts or stops SPORT Rx path.

If recording with audio codec, it starts SPORT Rx path.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral
<NewState>	u32	State (enable or disable) of the SPORT Rx

24.4.2.5 AUDIO_SP_TdmaCmd

Enables or disables SPORT Tx DMA request.

If Tx DMA request is not enabled, you should start Tx when GDMA completes every time

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral
<NewState>	u32	State (enable or disable) of the SPORT Tx DMA request

24.4.2.6 AUDIO_SP_RdmaCmd

Enables or disables SPORT Rx DMA request.

If Rx DMA request is not enabled, you should start Rx when GDMA completes every time

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral
<NewState>	u32	State (enable or disable) of the SPORT RX DMA request

24.4.2.7 AUDIO_SP_SetWordLen

Sets the audio SPORT word length.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral
<SP_WordLen>	u32	The value of word length

24.4.2.8 AUDIO_SP_GetWordLen

Gets the audio SPORT word length

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral

24.4.2.9 AUDIO_SP_SetMonoStereo

Sets the audio SPORT channel number

SPORT only supports stereo channel and mono channel.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral
<SP_MonoStereo>	u32	Mono or stereo channel

24.4.2.10 AUDIO_SP_TXGDMA_Init

Initializes GDMA peripheral for sending data.

Parameter	Type	Introduction
<Index>	u32	GDMA index
<GDMA_InitStruct>	GDMA_InitTypeDef *	GDMA_InitTypeDef structure that contains the configuration information for the GDMA peripheral
<CallbackData>	void *	GDMA callback data
<CallbackFunc>	IRQ_FUN	GDMA callback function
<pTxData>	u8 *	Tx buffer that stores Tx data
<Length>	u32	Tx data length

24.4.2.11 AUDIO_SP_RXGDMA_Init

Initializes GDMA peripheral for receiving data

Parameter	Type	Introduction
<Index>	u32	GDMA index
<GDMA_InitStruct>	GDMA_InitTypeDef *	GDMA_InitTypeDef structure that contains the configuration information for the GDMA peripheral
<CallbackData>	void *	GDMA callback data
<CallbackFunc>	IRQ_FUN	GDMA callback function
<pRxData>	u8 *	Rx buffer that stores the received data

<Length>	u32	Rx data length

24.4.3 SI APIs

SI APIs are used to read and write codec register.

API	Introduction
<AUDIO_SI_Cmd>	Enables or disables the specified audio SI peripheral
<AUDIO_SI_WriteReg>	SI writes codec register
<AUDIO_SI_ReadReg>	SI reads codec register
<AUDIO_SI_ClkCmd>	Turns on or turns off the clock of register bank of audio codec

24.4.3.1 AUDIO_SI_Cmd

Enables or disables the specified audio SI peripheral.

Parameter	Type	Introduction
<new_state>	u8	New state of the SI peripheral

24.4.3.2 AUDIO_SI_WriteReg

Uses SI interface to write codec register

Parameter	Type	Introduction
<address>	u32	Codec register address
<data>	u32	Data value written to the register

24.4.3.3 AUDIO_SI_ReadReg

Uses SI interface to read codec register

Parameter	Type	Introduction
<address>	u32	Codec register address

24.4.3.4 AUDIO_SI_ClkCmd

Turns on or turns off the clock of register bank of audio codec.

Parameter	Type	Introduction
<new_state>	u8	New state of the clock of register bank of audio codec

24.4.4 Codec APIs

API	Introduction
<CODEC_Init>	Initializes codec peripheral according to the application mode
<CODEC_SetVolume>	Sets codec volume by controlling mon DAC channel DVOL gain
<CODEC_GetVolume>	Gets codec mon DAC channel gain control
<CODEC_SetSr>	Sets codec ADC and DAC sample rate
<CODEC_SetAdcGain>	Sets codec ADC gain
<CODEC_SetAmicBst>	Set codec AMIC boost
<CODEC_SetDmicBst>	Set codec DMIC boost

<CODEC_SetMicBias>	Sets MIC_BIAS output voltage
<CODEC_MuteRecord>	Mutes or unmutes per ADC channel
<CODEC_MutePlay>	Mutes or unmutes per DAC channel.
<CODEC_DeInit>	De-initializes codec peripheral

24.4.4.1 CODEC_Init

Initializes codec peripheral according to the application mode.

Parameter	Type	Introduction
<sample_rate>	u32	Codec ADC and DAC sample rate
<word_len>	u32	Codec data sample bit
<mono_stereo>	u32	Codec mono channel or stereo channel
<application>	u32	Codec application mode, such as APP_AMIC_IN, APP_LINE_OUT

24.4.4.2 CODEC_SetVolume

Sets codec volume by controlling mon DAC channel DVOL gain.

Parameter	Type	Introduction
<vol_lch>	u8	Codec mon DAC left channel DVOL gain control (0.375dB/step) 8'hAF: 0dB 8'h00: -65.625dB
<vol_rch>	u8	Codec mon DAC right channel DVOL gain control (0.375dB/step) 8'hAF: 0dB 8'h00: -65.625dB

24.4.4.3 CODEC_GetVolume

Gets codec mon DAC channel gain control.

Parameter	Type	Introduction
<vol>	u16 *	Mon DAC channel DVOL gain (high 8 bits is RCH gain, low 8 bits are LCH gain)

24.4.4.4 CODEC_SetSr

Sets codec ADC and DAC sample rate.

Parameter	Type	Introduction
<sample_rate>	u32	Codec ADC and DAC sample rate

24.4.4.5 CODEC_SetAdcGain

Sets codec ADC gain.

Parameter	Type	Introduction
<ad_gain_left>	u32	ADC left channel digital volume gain
<ad_gain_right>	u32	ADC right channel digital volume gain

24.4.4.6 CODEC_SetAmicBst

Sets codec AMIC boost.

Parameter	Type	Introduction
<amic_bst_left>	u32	AMIC left channel boost gain
<amic_bst_right>	u32	AMIC right channel boost gain

24.4.4.7 CODEC_SetDmicBst

Sets codec DMIC boost.

Parameter	Type	Introduction
<dmic_bst_left>	u32	DMIC left channel boost gain
<dmic_bst_right>	u32	DMIC right channel boost gain

24.4.4.8 CODEC_SetMicBias

Sets MIC_BIAS output voltage

Parameter	Type	Introduction
<mic_bias>	u8	Microphone bias voltage setting

24.4.4.9 CODEC_MuteRecord

Mutes or unmutes per AD channel.

Parameter	Type	Introduction
<mute_lch>	u32	Mutes for left AD channel
<mute_rch>	u32	Mutes for right AD channel

24.4.4.10 CODEC_MutePlay

Mutes or unmutes per DA channel.

Parameter	Type	Introduction
<mute_lch>	u32	Mutes for left DA channel
<mute_rch>	u32	Mutes for right DA channel

24.4.4.11 CODEC_DeInit

De-initializes codec peripheral.

Parameter	Type	Introduction
<application>	u32	Codec application mode

24.5 How to Use AC APIs

24.5.1 Audio Play

To play the audio data through audio codec, follow the steps below:

(1) Open audio codec clock and function

```
PLL_x_Set (0, ENABLE); // x is 0 or 1)
RCC_PeriphClockCmd (APBPeriph_AUDIOC, APBPeriph_AUDIOC_CLOCK, ENABLE);
```

```
    RCC_PeriphClockCmd (APBPeriph_SPORT, APBPeriph_SPORT_CLOCK, ENABLE);
```

(2) Enable pin for audio codec function

```
    PAD_CMD (PinName, DISABLE);
```

(3) Initialize codec with desired parameters

```
    CODEC_Init(SampleRate, WordLen, MonoStereo, Application);  //Application is APP_LINE_OUT
```

(4) If you need to change codec volume, use

```
    CODEC_SetVolume(vol_lch, vol_rch);
```

(5) If you want to adjust microphone bias output voltage, use

```
    CODEC_SetMicBias(mic_bias);
```

(6) Fill the SPORT desired parameters

```
    AUDIO_SP_InitStruct(&SP_InitStruct);
```

(7) Configure audio SPORT with the corresponding configuration

```
    AUDIO_SP_Init (AUDIO_SP_DEV, &SP_InitStruct)
```

(8) Start Tx path

```
    AUDIO_SP_TdmaCmd (AUDIO_SPORT_DEV, ENABLE);
    AUDIO_SP_TxStart (AUDIO_SPORT_DEV, ENABLE);
```

(9) Activate GDMA to Tx data

```
    AUDIO_SP_TXGDMA_Init(Index,&GDMA_InitStruct, CallbackData, CallbackFunc, pTxData, Length);
```

24.5.2 Audio Record

To record the audio data through audio codec, follow the steps below:

(1) Open audio codec clock and function

```
    PLL_Set (0, ENABLE);  //0 is 0 or 1)
    RCC_PeriphClockCmd (APBPeriph_AUDIOC, APBPeriph_AUDIOC_CLOCK, ENABLE);
    RCC_PeriphClockCmd (APBPeriph_SPORT, APBPeriph_SPORT_CLOCK, ENABLE);
```

(2) Enable pin for audio codec function

```
    PAD_CMD (PinName, DISABLE);
```

(3) Initialize codec with desired parameters

```
    CODEC_Init(SampleRate, WordLen, MonoStereo, Application);
```

Application can select APP_AMIC_IN for analog microphone, select APP_DMIC_IN for digital microphone, or select APP_LINE_IN

(4) If codec need to change volume, use

```
    CODEC_SetVolume(vol_lch, vol_rch);
```

(5) If codec need to change ADC gain, use

```
    CODEC_SetAdcGain(ad_gain_left, ad_gain_right);
```

(6) Fill the desired parameters

```
    AUDIO_SP_InitStruct(&SP_InitStruct);
```

(7) Configure audio SPORT with the corresponding configuration

```
    AUDIO_SP_Init (AUDIO_SP_DEV, &SP_InitStruct)
```

(8) Start Rx path

```
    AUDIO_SP_RdmaCmd (AUDIO_SPORT_DEV, ENABLE);
    AUDIO_SP_RxStart (AUDIO_SPORT_DEV, ENABLE);
```

(9) Activate GDMA to Rx data

```
    AUDIO_SP_RXGDMA_Init(Index,&GDMA_InitStruct, CallbackData, CallbackFunc, pRxData, Length);
```

24.5.3 Example List

Example		Location
Playback		/project/realtek_amebaD_va0_example/example_sources/Audio/dac
Record and playback	AMIC	/project/realtek_amebaD_va0_example/example_sources/Audio/adc
	DMIC	/project/realtek_amebaD_va0_example/example_sources/Audio/dmic
Record and store	AMIC	/component/common/example/audio_sport/audio_recorder
Record and upload	DMIC	/component/common/example/audio_sport/audio_pcm_upload
Decode algorithm	AC3	/component/common/example/audio_sport/audio_ac3
	AMR	/component/common/example/audio_sport/audio_amr
	FLAC	/component/common/example/audio_sport/audio_flac

	Helix AAC	/component/common/example/audio_sport/audio_helix_aac
	Helix MP3	/component/common/example/audio_sport/audio_helix_mp3
	HLS	/component/common/example/audio_sport/audio_hls
	M4A	/component/common/example/audio_sport/audio_m4a
	M4A selfparse	/component/common/example/audio_sport/audio_m4a_selfparse
	MP3	/component/common/example/audio_sport/audio_mp3

Note: [illegible] are available. If you want to disable the GDMA, it should be done [illegible] in a GDMA related interrupt routine. Disabling the GDMA outside of an interrupt routine may cause an exception if the GDMA is transferring data.

24.6 Hardware Design Guide

24.6.1 Line-out

The line-out connection of audio codec is illustrated in Fig24-13. The capacitors between 3.5mm jack and IC should be 47uF tantalum capacitors rather than ceramic capacitors. The reason is that capacitance value of ceramic capacitors may decrease when bias voltage is applied to them, which causes audio performance bad.

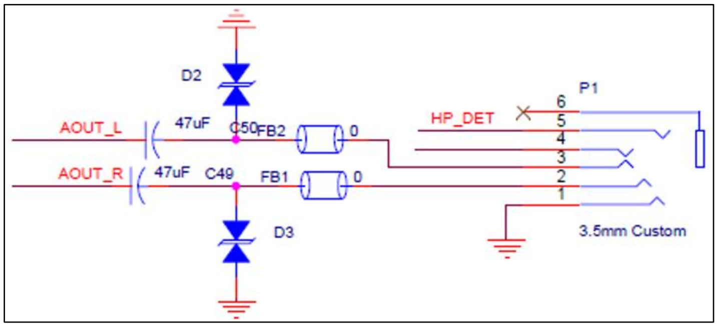


Fig24-13 Line-out connection

24.6.2 AMIC-in

The AMIC-in connection of audio codec is illustrated in Fig24-14. The capacitor between analog microphone and IC should be 1uF. Larger capacitance value makes longer period needed for cap. charging.

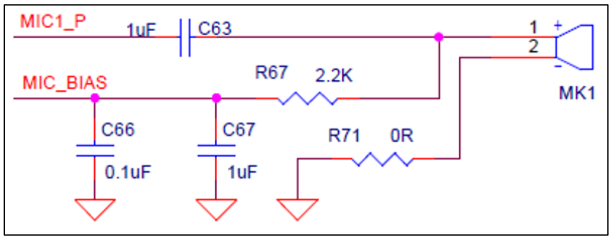


Fig24-14 AMIC-in connection

MIC_BIAS connects to the positive side of microphone through a 2.2kohm resistor to offer bias voltage.

Short the negative side of microphone to ground if working at single-ended mode, or connect to ground through a 2.2kohm resistor at differential mode.

Connect the negative side of microphone to MIC_N through a 1uF capacitor at differential mode.

24.6.3 Power

The power connection of audio codec is illustrated in Fig 24-15.

¢ K Ŝ Ŏ l· LJ l· Ŏ ƛ ű ƨ NJ ō Ŝ ű ø Ŝ Ŝ y˝ ! ¦ 5 L h ψ ± w 9 C l· y˝ Ř ∃ NJ ƨ dz y˝ Ř ă K ƨ dz ť Ř ō Ŝ π Φ т y˝ C Φ stabilization.

The capacitor between AVCC or AVCC_DRIV and ground should be 1uF or a little larger to keep voltage stable.

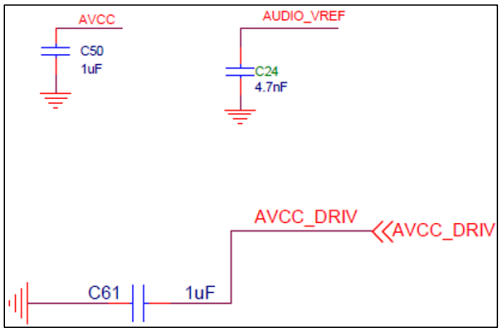


Fig 24-15 Power connection

24.7 Performance of Encoding & Decoding

The performance of decoding or encoding audio files of different formats are listed in Table 24-3 ~ Table 24-9. Data listed in these tables are obtained when the dynamic memory is allocated in SRAM. Data may be different if memory is allocated in PSRAM.

24.7.1 AC3 Format

The performance of decoding AC3 format audio data is listed in Table 24-3.

Table 24-3 Performance of decoding AC3 format audio data

AC 3 Channel	Bit Rate (kHz)	Sample Rate (kHz)	Output Channel	Average MIPS Measured[1]	MIPS Simulated[2]	
					Average	Maximum
5.1	640	48	1	50.5	53.7	54.5
5.1	448	48	1	72.2	70.8	76.8
5.1	448	48	1	69.9	68.2	90.8
5.1	448	48	1	68.7	65.9	92.3
2.1	192	48	1	54.8	50.6	52.2
2.0	192	48	1	TBD	50.7	49.6

1. The values are evaluated on the AmebaD_QFN88_EVB_V1.0
2. The values are evaluated with the Keil simulator.

24.7.2 OPUS Format

The simulated CPU load of encoding and decoding of OPUS audio data is listed in Table 24-4. Keil simulator is used during the whole process. The decoding files used in the measurement process are created with opus-tools-0.2-opus-1.3.

Table 24-4 Simulated CPU load of encoding and decoding of OPUS audio data

Rate (kHz)	Channel	Bit Rate (kbps)	Complexity	MIPS		Measured	
				Decoding	Encoding	Decoding	Encoding
48	2	256	10	63	153.5	82.7	261
48	2	256	3	58	122.5	71.4	142.2
48	2	256	0	53	102	67.6	129.8

48	1	256	3	38	74.5	53.4	81.8
16	1	48	3	8.5	46.5	TBD	108.6
16	1	20	3	7	44.5	TBD	101.6

If users decide to use libopus library to deal with their data, the code size requirement is listed in Table 24-5. The version used here is libopus 1.1.4.

Table 24-5 Code size requirement using libopus

Type	Program Size	Data Size
SILK encoder	77.1k	8.5k
Original Source Code (Encode + Decode)	133k	23.6k

In Table 24-6, CPU load on using Silk encoder is listed.

Table 24-6 CPU load on using Silk encoder

Test Condition	Complexity	Average CPU Load (MHz)	Remarks
SILK encoder, 16k, 1ch, 20kbps	3	42	SILK VBR
	10	182	

24.7.3 FLAC Format

The simulated (with Keil simulator) and measured CPU load for decoding FLAC audio data is listed in Table 24-7. The tested file lasts for 279.64s, and the decoding time is 13.51s, which means 2.89s decoding time for 1-minute audio data.

Table 24-7 CPU load of decoding FLAC audio data

Rate (kHz)	Channel	Word Length	MIPS Simulated	MIPS Measured
44.1	2	16	19	9.66

24.7.4 AAC Format

The measured CPU load for decoding AAC audio data is listed in Table 24-8. The tested file lasts for 24.576s, and the decoding time is 3.55s, which means 8.68s decoding time for 1-minute audio data.

Table 24-8 CPU load of decoding AAC audio data

Rate (kHz)	Channel	Bit Rate (kbps)	MIPS
48	2	320	28.93

24.7.5 MP3 Format

The measured CPU load for decoding MP3 audio data is listed in Table 24-9. The tested file lasts for 5.58s, and the decoding time is 849ms, which means 9.13s decoding time for 1-minute audio data.

Table 24-9 CPU load for decoding MP3 audio data

Rate (kHz)	Channel	Bit Rate (kbps)	MIPS
32	2	320	30.43

24.8 Q & A

24.8.1 How to Connect the Output of DAC to A Power Amplifier?

In our SDK, the output of DAC is configured as single-ended by default. The N-end should be left alone. The P-end of L/R should be connected to the amplifier respectively. Choose either the P-end of L or R if only one channel is needed.

If the power amplifier is an AB type amplifier, refer to the design shown in Fig 24-16 (power amplifier is LM4991).

If the power amplifier is a D type amplifier, refer to the design shown in Fig 24-17 (power amplifier is ALC1003).

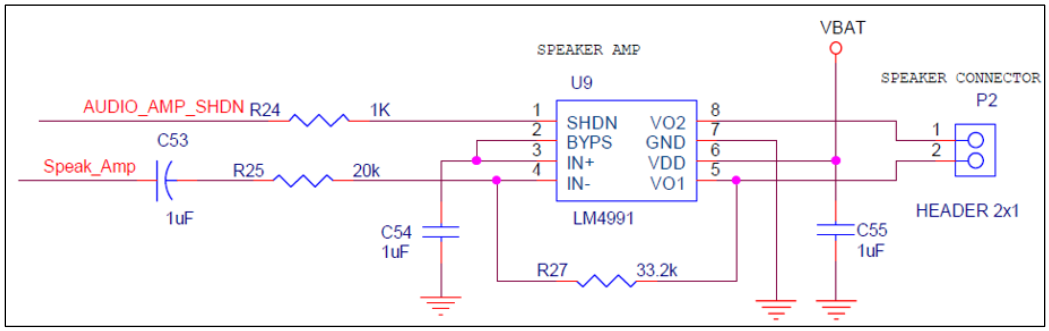


Fig 24-16 Reference design of using AB type power amplifier

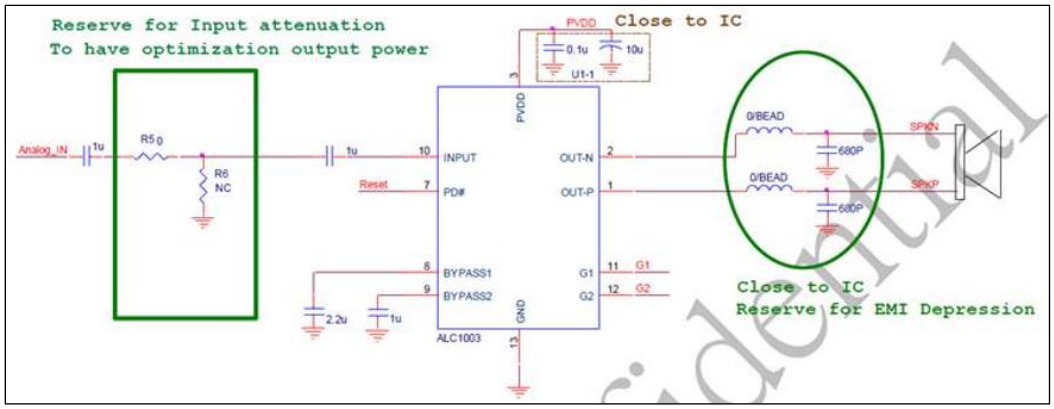


Fig 24-17 Reference design of using D type power amplifier

24.8.2 What's the Difference between Single-ended MIC Input and Differential MIC Input?

The SNR of differential mic input is better than single-ended input, but one more pin is needed. Ameba supports 2 mics. One supports both single-ended input and differential inputs, but another only supports single-ended input.

24.8.3 How to Play Local Audio Files?

If audio files are stored in SD card, you can use APIs related to SD card and file system to read and play audio files. If audio files are stored in flash, you need to copy the audio files to SRAM or PSRAM first, then call AUDIO_SP_TXGDMA_Init() to play, and pass the address directly to AUDIO_SP_TXGDMA_Init().

25 CapTouch

AmebaD CapTouch provides 4 channels for capacitive sensing. The sensitivity and threshold for each channel are configurable. For different applications and surroundings, users should tune parameters to achieve the best performance.

This chapter introduces how to use CapTouch and design touch key.

25.1 Pinmux

The pin assignments of CapTouch channels are listed in Table 21-4.

Table 25-1 CapTouch pin assignments

Channel	Port Name	Pin Name	QFN48	QFN68	QFN88
0	PB[4]	TOUCH_KEY0			
1	PB[5]	TOUCH_KEY1			
2	PB[6]	TOUCH_KEY2			
3	PB[7]	TOUCH_KEY3			

25.2 APIs

25.2.1 CapTouch_StructInit

Items	Description
Introduction	Initializes the parameters in the CapTouch_InitStruct with default values.
Parameters	CapTouch_InitStruct: pointer to a CapTouch_InitTypeDef structure which is initialized.
Return	N/A

Note: CapTouch_InitStruct contains the configurable parameters for CapTouch and each channel, which includes the debounce control, ETC parameters, mbias current and threshold for channels

25.2.2 CapTouch_Init

Items	Description
Introduction	Initializes the CapTouch peripheral according to the specified parameters in the CapTouch_InitStruct.
Parameters	CapTouch: which should be CAPTOUCH_DEV. CapTouch_InitStruct: pointer to a CapTouch_InitTypeDef structure that contains the configuration information for the specified CapTouch peripheral.
Return	N/A

25.2.3 CapTouch_Cmd

Items	Description
Introduction	Enables or disables the specified CapTouch peripheral.
Parameters	CapTouch: which should be CAPTOUCH_DEV. NewState: new state of the CapTouch peripheral. This parameter can be ENABLE or DISABLE.
Return	N/A

25.2.4 CapTouch_INTConfig

Items	Description
Introduction	Enables or disables the specified CapTouch interrupts.
Parameters	CapTouch: which should be CAPTOUCH_DEV. CapTouch_IT: specifies the CapTouch interrupt to be enabled or masked. This parameter can be one or combinations of the following values: BIT_CT_OVER_N_NOISE_THRESHOLD_INT: CapTouch negative noise overflow interrupt BIT_CT_FIFO_OVERFLOW_INT: CapTouch FIFO overflow interrupt BIT_CT_OVER_P_NOISE_THRESHOLD_INT: CapTouch positive noise overflow interrupt CT_CHX_PRESS_INT(x): CapTouch channel(x) press interrupt, where x can be 0~3 CT_CHX_RELEASE_INT(x): CapTouch channel(x) release interrupt, where x can be 0~3 NewState: new state of the specified CapTouch interrupts mask. This parameter can be ENABLE or DISABLE
Return	N/A

25.2.5 CapTouch_GetISR

Items	Description
Introduction	Gets CapTouch interrupt status.
Parameters	CapTouch: which should be CAPTOUCH_DEV.
Return	Interrupt status

25.2.6 CapTouch_INTClearPendingBit

Items	Description
Introduction	Clears the specified CapTouch interrupt pending bit.
Parameters	CapTouch: which should be CAPTOUCH_DEV. CapTouch_IT: specifies the CapTouch interrupt to be cleared. This parameter can be one or combinations of the following values: BIT_CT_N_NOISE_OVERFLOW_INT_CLR: CapTouch negative noise overflow interrupt BIT_CT_FIFO_OVERFLOW_INT_CLR: CapTouch FIFO overflow interrupt BIT_CT_P_NOISE_OVERFLOW_INT_CLR: CapTouch positive noise overflow interrupt CT_CHX_PRESS_INT(x): CapTouch channel(x) press interrupt, where x can be 0~3 CT_CHX_RELEASE_INT(x): CapTouch channel(x) release interrupt, where x can be 0~3
Return	N/A

25.3 How to Use CTC

25.3.1 CTC Initialization

The CapTouch initialization is implemented by the app_captouch_init() function in main.c. You can follow these steps to use it
(1) Set the km0_enable_key_touch in ps_config of RTL8721dlp_sleepcfg.c to BIT_CAPTOUCH_ENABLE to enable CapTouch when KM0 boot.

```
PSCFG_TypeDef ps_config = {
    .km0_config_wifi_enable = TRUE,
    .km0_enable_key_touch = FALSE, //BIT_KEY_ENABLE | BIT_CAPTOUCH_ENABLE,
    .km0_tickles_debug = FALSE,/* if open WIFI FW, should close it, or beacon will lost in WOWLAN */
    .km0_osc2m_close = TRUE,
    .km0_pg_enable = FALSE,
    .km0_rtc_calibration = FALSE,
    .km0_audio_pad_enable = TRUE,
};
```

(2) Set configuration parameters in the following structure in touch_key.c, which includes touch threshold, noise threshold, mbias current, enable control for each channel. The method to tune parameters for each channel can be found in 25.3.2.

```
static CapTouch_CHInitTypeDef CTCChan[4] =
{
    /*DiffThrehold, MbiasCurrent, ETCNNoiseThr, ETCPNoiseThr, CHEnable*/
    {35,        0x16,       3,              18,             ENABLE},  /* Channel 0 */
    {500,       0x08,       250,            250,            DISABLE}, /* Channel 1 */
    {500,       0x08,       250,            250,            DISABLE}, /* Channel 2 */
    {500,       0x0b,       250,            250,            DISABLE}, /* Channel 3 */
};
```

(3) Rebuild SDK and then burn images, the CapTouch would work after boot up. When finger touch or proximity is detected, the CapTouch would send a Key Press interrupt to system.

25.3.2 CTC Calibration

To achieve the best performance (sensitivity, reliability or response time), users need to tune touch threshold, noise threshold and mbias current during development. The reference value of parameters is shown in Table 25-2. Of course, users can tune other parameters manually to fit special requirements.

The parameters for each channel should be tuned individually. To calibrate for channel 0, for example, users can follow these steps:

(1) Initialize the parameter of CapTouch_InitStruct with default values using CapTouch_StructInit() and to enable channel 0.
```
CapTouch_StructInit(&CapTouch_InitStruct);
CapTouch_InitStruct.CT_Channel[0].CT_CHEnable = ENABLE;
```
(2) Initialize the CapTouch peripheral according to CapTouch_InitStruct in step(1).
```
CapTouch_Init(CAPTOUCH_DEV, &CapTouch_InitStruct);
```
(3) Enable CapTouch
```
CapTouch_Cmd(CAPTOUCH_DEV, ENABLE);
```
(4) Call CapTouch_GetChAveData() to read the sample data from channel 0 periodically.
```
CapTouch_GetChAveData(CAPTOUCH_DEV,
```

Table 25-2 Reference value of parameters

Parameters	Description	Reference Range	
		Finger Touch Detection	Proximity Detection
mbias	Set to a value which makes the baseline within the reference range	Baseline: 2500~3500	Baseline: 3500~3800
Difference Threshold	Set to 80% of the touched difference value	500~600	30~80
Noise Threshold	Set to no more than 50% of the difference threshold, and it can be adjusted according to the actual situation Maximum value: 255	200~255	0~40

25.4 CapTouch Schematic Design and PCB Layout Guidelines

The CTC of AmebaD has 4 CapTouch channels, and all the four channels support buttons and proximity sensors. Both the schematic design and PCB layout are important for CapTouch application. This chapter provides guidelines of CapTouch schematic design and PCB layout.

25.4.1 CapTouch Schematic Design

The CTC supports 4 channels, each channel supports button and proximity sensor. While only 2 channels are used. When designing the CapTouch schematic, you should try to select the sensor pins that are next to each other for better stability. Also, do not make the sensor trace run parallel to a clock signal or driven signal, which may result in cross-talk, and may cause the sensor next to those signals to make trigger false. Refer to PCB Layout in getting started with PCB layout design guidelines to avoid cross-talk.

All CapTouch channels must have a 47Ω (placed close to the chip) to improve noise immunity both for button and proximity sensor. If any CapTouch channel is not used, it is recommended to disable this channel and connect it to ground. Refer to How to Use CTC for details of disabling a channel.

For example, Fig 25-1 is a schematic in which only 4 channels are enabled.

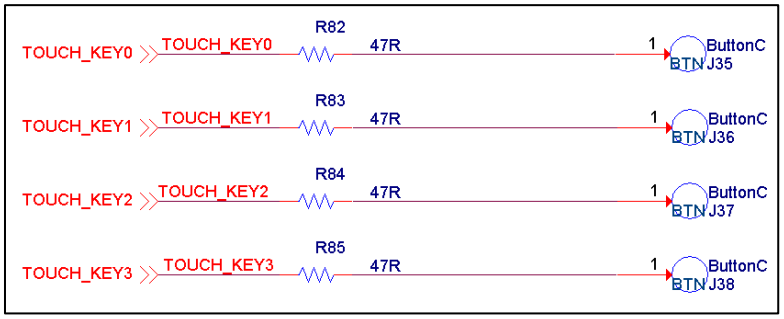


Fig 25-1 4 Channels enabled schematic

25.4.2 PCB Layout

In a typical CapTouch application, sensors are constructed with traces on a FR4/FR2 or flexible printed circuit (FPC). CapTouch layout design is an important step in the design phase, following the PCB layout design guide lines can help your design achieve higher noise immunity, lower parasitic capacitance (CP), and higher signal-to-noise ratio (SNR). The following factors must be considered during layout.

Sensor parasitic capacitance: Sensor CP depends on sensor dimensions, PCB overlay and trace length. A high sensor CP makes it more difficult to sense small changes in sensor capacitance and thereby reduces sensitivity.

Sensor dimensions: The sensor dimensions depend on the overlay thickness and suitable dimension for human touch. For touch convenience, a larger dimension is needed. A thicker overlay requires a larger sensor dimensions. But a larger dimension causes a higher parasitic capacitance.

Sensor trace length: Longer trace lengths add the sensor CP and reduce the sensor sensitivity. Also, a long trace acts as antenna and reduces the noise immunity of the sensor.

Power consumption: The power consumption of the sensor depends scan period and sensitivity (mbias), while the sensitivity depends on the CP of the sensor. Higher CP results in higher sensitivity, higher sensitivity and smaller scan period results in higher power consumption. To achieve a lower power consumption, reduce the sensor CP. To achieve a lower CP, higher SNR, and lower power consumption, follow the layout design guidelines.

For better performance, the follow guidelines provide recommendations of the sensor shape, minimum and maximum sensor dimensions, placement of the sensor, and routing traces on the PCB or FPC. Table 25-3 summarizes the sensor layout guidelines while designing a CapTouch application, both of button application and proximity application.

Table 25-3 Sensor layout recommendations

Category	Item	Min.	Max.	Description
Button	Button parasitic capacitance	4pF	50pF	Lower CP, lower sensitivity, lower power consumption
	Button shape	N/A	N/A	Round or rectangle with round corners
	Button size	5mm	15mm	Larger size causes more power consumption, the recommended size is 8mm~12mm for round shape
	Button-Button spacing	Button-Ground clearance	N/A	Depending on button application, for better stability, the recommended spacing is larger than 8mm.
	Button-GND clearance	0.5mm	2.54mm	The recommended clearance is equal to the overlay thickness
Proximity	Proximity parasitic capacitance	8pF	50pF	Proximity needs lower CP and higher mbias for better sensitivity, which causes more power consumption
	Proximity sensor shape	N/A	N/A	Circular or rectangular loop (with round corners) on PCB for large area Circular or rectangular solid fill (with round corners) for small area. Line with recommended sensor trace width A GND loop surrounding the sensor results in better noise immunity but worse sensitivity.
	Proximity sensor trace width	2mm		The recommended width is 2mm~3mm
	Proximity sensor-GND loop clearance	1.5mm	2.5mm	The recommended width is 2mm

	GND loop trace width	1.5mm		1.5mm
	Proximity sensor loop diameter			The recommended loop diameter needs to be larger than the proximity distance

Table 25-4 summarizes the layout guidelines for placement of components, routing of sensor, sensor trace and GND layer while designing a CapTouch application, both button application and proximity application.

Table 25-4 Other layout recommendations

Category	Item	Min.	Max.	Description
Placement	2 layer PCB	N/A	N/A	Top: Sensors, devices, and components; VDD and GND traces, other signal traces Bottom: Sensor traces, devices and components, VDD and GND traces, other signal traces Note: Avoid VDD traces, clock traces, and driven traces over sensor traces. Avoid sensor trace run parallel to a clock signal or driven signal
	4 layer PCB	N/A	N/A	Top: Sensors, devices, and components Second layer: Hatched ground Third layer: Sensor traces, VDD layer Bottom: Devices, and components Note: Avoid VDD traces, clock traces, and driven traces run parallel under or over sensor traces.
Routing	Sensor trace length			Not limited, make sure the trace capacitance and sensor capacitance is less than 50pF.
	Sensor trace width	6mil	9mil	The recommended value is 7mil (FR4/FR2/FPC)
	Sensor trace routing	N/A	N/A	The best choice for sensor is routed on the non-sensor side. Otherwise, make sure the sensor trace area is a touch forbidden area. Note: Avoid sensor trace run parallel to a clock signal or driven signal. If any non-sensor trace crosses the sensor trace ,ensure that the intersection is orthogonal
	Via position	N/A	N/A	Via should be placed near the edge.
	Via number	0	3	1 via is needed, more via results other parasitic capacitance
	Via hole size	12mil/6mil	24mil/12mil	The recommended via hole size is 16mil/10mil
	Trace series resistor	47	560	Series resistors close to the chip for better suppression and ESD
	Trace-GND layer distance	10mil	20mil	20mil
GND	GND-Top layer			Hatched pattern 7mil trace and 35~45 mil grid
	GND-Bottom layer			Hatched pattern 7mil trace and 65~75 mil grid for ground
Overlay	Overlay thickness	N/A	3mm	Depends on your product, 3mm for glass or plastic, higher thickness may decrease the sensitivity.
	Overlay material	N/A	N/A	Non-conductive material

For example, Fig 25-2 is a PCB design of button and proximity application.

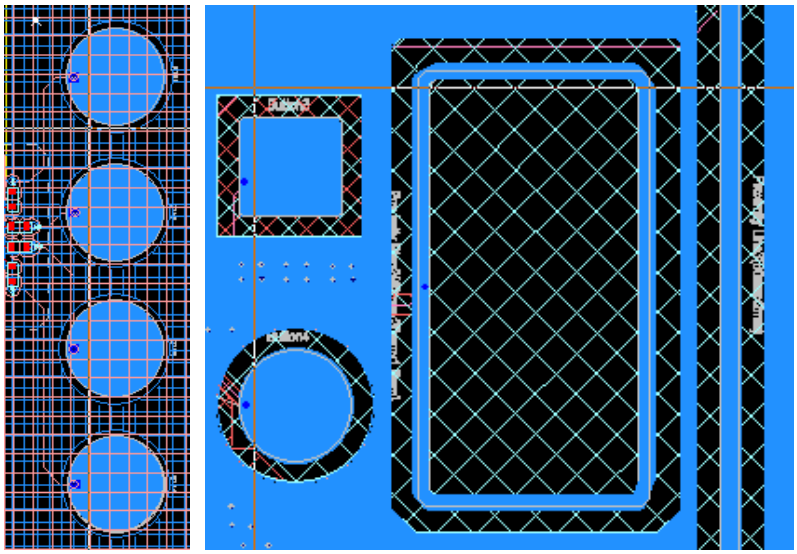


Fig25-2 PCB design of button and proximity application

26 DuerOS

26.1 DuerOS Platform

DuerOS is an important application of Baidu Artificial Intelligence (AI) technology. It has huge amounts of data, and can control or communication with the hardware through natural language. DuerOS is widely used in intelligent toys, bluetooth speakers, intelligent household appliances and other equipment.

DuerOS development platform aims to create intelligent terminal era of AI. Users can use it to build their own exclusive artificial intelligence products.

This chapter takes story machine as an example.

26.1.1 Apply product

DuerOS development platform home page url: http://open.duer.baidu.com/didp/main/index.

To apply your own product, follow the steps:
(1) Enter DuerOS home page and register as a developer.

Fig26-1 Register guide

(2) Configure new device.
a) After successful registered, click Configure New Device and enter the next step.

Fig26-2 Device control platform

b) Configure the device as your need, story machine is chosen as an example.

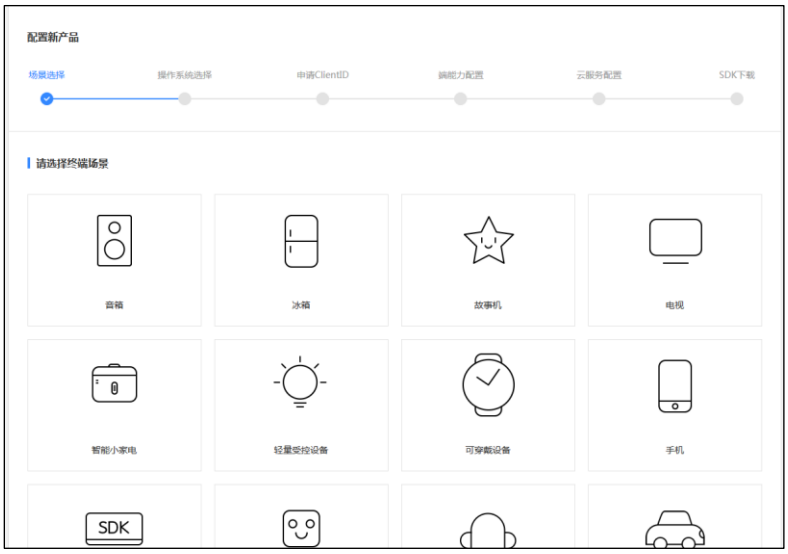


Fig26-3 Configuring new product

c) Choose FreeRTOS and enter the next step

Fig26-4 Choosing FreeRTOS

d) Give the product name and click the Apply ClientID button.

Fig26-5 Apply ClientID

e) Click the Light Device Configuration button.

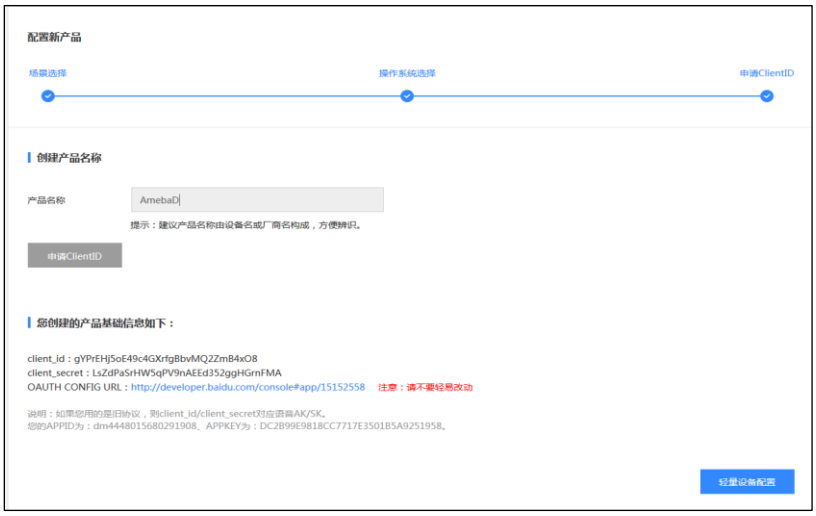


Fig26-6 Configuringdevice

After the abovesteps, story machine product has been created. You can get story machine information thProduct Information menu, and can ustomizeDuerOSservice

Fig26-7 Product center

26.1.2 Apply Profile

After product has been applied, get a set of profiles throughDevice Development menu.

Fig26-8 Getting profile

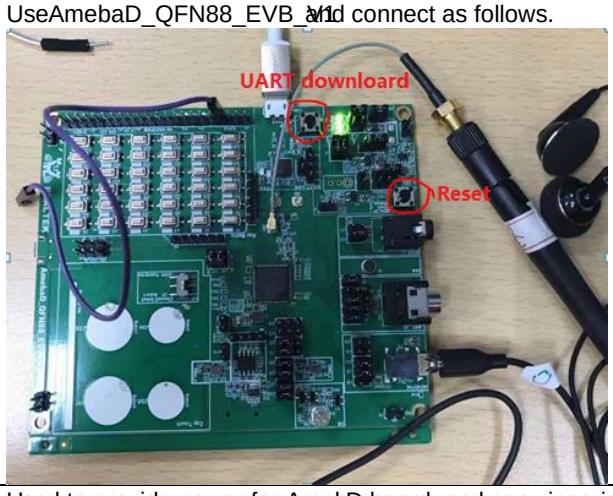
Duer profile, which contains UUID, Token, server address, port, and certificate, is used to establish a TLS connection with the cloud and register to the cloud

Note:

Every UUID can only be used by one device.

The profile cannot be used directly, because we only support mp3 format now. Product manager should negotiate with Baidu to obtain the other format for these profiles.

26.2 Hardware Requirement

Hardware	Description
AmebaD development board	Use AmebaD_QFN88_EVB_v1.1 and connect as follows. 
Micro-USB cable	Used to provide power for AmebaD board, and can view serial port log
Play equipment	Headset or speaker can be used to play audio data.

26.3 Software Component

DuerOS software component is shown in Fig1-1.

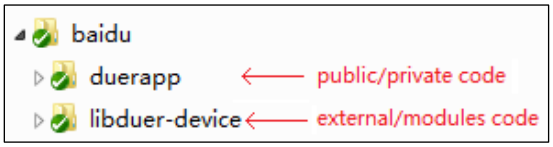


Fig26-9 Software component

26.3.1 duerapp

Items	Description
include	duerapp header files
src	public: AmebaD initializes peripheral: Audio codec, Wi-Fi, OTA etc private: parse play file, record from file, load profile.

26.3.2 libduer-device

Items	Description
external	DuerOS external library, include mbedtls, speex, zlib etc
framework	Utilities and core.
modules	Modules: OTA, HTTP, dcs, coap etc
platform	Connect and communicate with cloud

DuerOS libduer-device consists of three parts: Lightduer, Baidu CA, Speex.

Lightduer is the interface between DuerOS and external, it communicates with FreeRtos, Lwip, Mbedtls

Baidu CA is used to connect and communicate with Baidu Cloud

Speex is responsible for voice compression

26.4 How to Use DuerOS

26.4.1 Build and Download

(1) Modify the profile

As every device has a unique profile, we should change the profile for each device. Zip the profile and choose one file to open. Copy the profile to the component\common\application\baidu\duerapp\src\public\duerapp.conf and modify to the correct format.

```
#define DUER_PROFILE_UUID   1  //Range: 5,6,7,8,9
#if (PROFILE_FROM_SDCARD == 0)
#if (DUER_PROFILE_UUID == 1)
SDRAM_DATA_SECTION static const char duer_profile[] =
    "{\"configures\":\"{}\",\"bindToken\":\"9071f6a0954cfe17969699c4b83c2437\",\"coapPort\":443,\
\"token\":\"hpUun7nsMgXgk3Vvig8jc7Gg89CBjeAq\",\"serverAddr\":\"device.iot.baidu.com\",\"1wm2mPort\":443,\
\"uuid\":\"19c80000000001\",\"rsaCaCrt\":\"-----BEGIN CERTIFICATE-----\\n\
MIIDUDCCAjgCCQCmVPUErMYmCjANBgkqhkiG9w0BAQUFADBqMQswCQYDVQQGEwJD\\n\
TjETMBEGA1UECAwKU29tZS1TdGF0ZTEOMAwGA1UECgwFYmFpZHUxGDAWBgNVBAMM\\n\
DyouaW90LmJhaWR1LmNvbTEcMBoGCSqGSIb3DQEJARYNaW90QGJhaWR1LmNvbTAe\\n\
Fw0xNjAzMTEwMzMwNDlaFw0yNjAzMDkwMzMwNDlaMGoxCzAJBgNVBAYTAkNOMRMw\\n\
EQYDVQQIDApTb211LVN0YXR1MQ4wDAYDVQQKDAViYW1kdTEYMBYGA1UEAwwPKi5p\\n\
b3QuYmFpZHUuY29tMRwwGgYJKoZIhvcNAQkBFg1pb3RAYmFpZHUuY29tMIIBIjAN\\n\
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtbhIeiN7pznzuMwsLKQj2xB02+51\\n\
OvCJ5d116ZFLjecp9qtllqOfN7bm+AJa5N2aAHJtsetcTHMitY4dtGmOpw4d1Gqx\\n\
luoz50kWJWQjVR+z6DLPnGE4uELOS8vbKHUoYPPQTT80eNVn19S9h/17DcjEAJYC\\n\
IYJbf6+K9x+Ti9VRChvWcvgZQHMRym9j1g/7CKGMCIwkC+6ihkGD/XG40r7KRCyH\\n\
bD53KnBjBO9FH4IL3rG1ZWKWzMw3zC6RTS2ekfEsgAtYDvROKd4rNs+uDU9xaBLO\\n\
dXTl5uxgudH2VnVzWtj09OUbBtXcQFD2IhmOl20BrckYul+HEIMR0oDibwIDAQAB\\n\
MA0GCSqGSIb3DQEBBQUAA4IBAQCzTTH91jNh/uYBEFekSVNg1h1kPSujlwEDDf/W\\n\
pjqPJPqrZvW0w0cmYsYibNDy985JB87MJMfJVESG/v0Y/YbvcnRoi5gAenWXQNL4\\n\
h2hf08A5wEQfLO/EaD1GTH3OIierKYZ6GItGrz4uFKHV5fTMiflABCdu37ALGjrA\\n\
rIjwjxQG6WwLr9468hkKrWNG3dMBHKvmqO8x42sZOFRJMkqBbKzaBd1uW4xY5XwM\\n\
S1QX56tVrgO0A3S+4dEg5uiLVN4YVP/Vqh4SMtYkL7ZZiZAxD9GtNnhRyFsWlC2r\\n\
OVSdXs1ttZxEaEBGUl7tgsBte556BIvufZX+BXGyycVJdBu3\\n\
-----END CERTIFICATE-----\\n\",\"macId\":\"\",\"version\":12237}";

#elif (DUER_PROFILE_UUID == 5)
```

Fig26-10 Profile in duerapp.c

(2) Select AMIC or DMIC to record.
(3) Modify macro CONFIG_DMIC_SEL in project\realtek_amebaD_va0_example\inc\inc_hp\platform_opts.h.

Items	Value	Description
#define CONFIG_DMIC_SEL	0	Use AMIC to record
	1	Use DMIC to record

(4) Build the project to generate DuerOS image.
(5) Use ImageTool to download DuerOS image into AmebaD platform.

26.4.2 Configure the Network

After downloading the image successfully, reset the AmebaD board.
If it is the first up, it enters the simple configuration mode. You can use the simple configuration tools to configure the network.
If it is not the first reset, it connects to the network automatically, because the SSID and password are saved in the Flash.

```
18:09:01.397  Firmware Enable
18:09:01.400  ===>Retention Ram Init
18:09:01.413  Firmware Disable
18:09:01.417  [rltk_wlan_deinit] Wait for RxStop
18:09:01.446  WIFI deinitialized
18:09:01.449  Firmware Enable
18:09:01.451  ===>Retention Ram Init
18:09:01.464  Initializing WIFI ...
18:09:01.479  WIFI initialized
18:09:02.245
18:09:02.249  Switch to channel(2)
18:09:02.432
18:09:02.432  Switch to channel(3)
18:09:02.550
18:09:02.555  Switch to channel(4)
18:09:02.719
18:09:02.719  Switch to channel(5)
18:09:02.854
18:09:02.856  Switch to channel(6)
18:09:03.014
18:09:03.020  Switch to channel(7)
18:09:03.157
18:09:03.160  Switch to channel(8)
18:09:03.317
18:09:03.319  Switch to channel(9)
18:09:03.487
18:09:03.487  Switch to channel(10)
18:09:03.642
18:09:03.642  Switch to channel(11)
18:09:03.749  Input frame da: 01 00 5E 00 00 FB, data length: 39
18:09:03.781
```

Fig26-11 Simple configration log

26.4.3 DuerOS Connection Success

After connecting to the network successfully, the device connects to the Baidu Clouds automatically. When printing the log like Fig26-12, it means that DuerOS connection is successful.

```
18:38:02.748  duerapp: duer_events_call_internal, handler not initialied...
18:38:02.749  duerapp: duer_engine_start, g_handler:0x10053e00, length:1469, profile:0x10042690
18:38:02.750  duerapp:     duer_conf_get_string: uuid = 19c80000000001
18:38:02.751  duerapp:     duer_conf_get_string: serverAddr = device.iot.baidu.com
18:38:02.807  duerapp: DNS lookup succeeded. IP=180.97.33.165
18:38:08.612  duerapp: soc:0x100435a8, destroy:1, fd:-1, ref_count:0, timer:0x10043580
18:38:09.136  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:09.142  duerapp: duer_engine_start, g_handler:0x10053e00, length:0, profile:0x0
18:38:09.520  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:09.921  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:10.048  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:10.143  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:10.150  duerapp: duer_engine_start, g_handler:0x10053e00, length:0, profile:0x0
18:38:10.367  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:12.130  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:12.164  duerapp: duer_engine_start, g_handler:0x10053e00, length:0, profile:0x0
18:38:16.158  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:16.178  duerapp: duer_engine_start, g_handler:0x10053e00, length:0, profile:0x0
18:38:23.929  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:23.968  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:26.670  duerapp: will start latter(DUER_ERR_TRANS_WOULD_BLOCK)
18:38:26.676  duerapp: connect started!
18:38:26.680  duerapp: Mutex initializing
18:38:26.682  duerapp: event: 0
18:38:26.685  duerapp: current vol 10, mute 0
18:38:26.687
18:38:26.689  duerapp: add resource successfully!!
18:38:26.692  duerapp: add resource successfully!!
```

Fig26-12 DuerOS connection success log

26.4.4 Triger Record and Speak

(1) PA[22] pull high to trigger audio codec record.
In driver GPIO_IRQ_PIN set IRQ_RISE to trigger audio codec record, and GPIO_IRQ_PIN is PA[22], so use PA[22] to trigger.
(2) Make a voice command
(3) After PA[22] pull high, DuerOS would response with "Hello". Then after it says "Hello', you can make a voice command such as "What's the weather", "Sing a song", "Play a joke", "What's your name" and so on to chat with DuerOS.

```
void init_voice_trigger_irq(void (*callback) (uint32_t id, gpio_irq_event event))
{
    gpio_irq_t voice_irq;

    //init voice trigger pin
    gpio_irq_init(&voice_irq, GPIO_IRQ_PIN, callback, NULL);
    gpio_irq_set(&voice_irq, IRQ_RISE, 1);
    gpio_irq_enable(&voice_irq);
}
```

Fig26-13 Trigger code

26.5 OTA Upgrade

26.5.1 Generating OTA Image

(1) Change FIRMWARE_VERSION in component\common\application\baidu\duerapp\include\duerapp_ota.h

```
#define FIRMWARE_VERSION "1.0.1.5"  //new version,update it when make new version
#define CHIP_VERSION     "RTL8721d"
#define SDK_VERSION      "1.0"
```

Fig26-14 Firmware version

(2) Change OTA2 address mapping
As the size of DuerOS image is more than 1M, there should at least 4M Flash if OTA is needed. If the Flash size is 4M, modify the component\soc\realtek\amebad\fwlib\usrcfg\rtl8721d_bootcfg.c as follows

```
00038: /*
00039: * @brif     OTA start address. Because KM0 & KM4 IMG2 are combined, users only need to set the start address
00040: *      of KM0 IMG2.
00041: */
00042: BOOT_RAM_DATA_SECTION
00043: u32 OTA_Region[2] = {
00044:      0x08006000,          /* OTA1 region start address */
00045:      0x08206000,          /* OTA2 region start address */
00046: };
```

Fig26-15 OTA Region

(3) Generate OTA image
For AmebaD, km0_km4_image2.bin in folder project\realtek_amebaD_va0_example\GCC-RELEASE\project_hp\asdk\image is the OTA image

26.5.2 OTA with DuerOS Platform

(1) Add OTA firmware.
(2) Click Add button, then fill in the OTA firmware information, such as version, file, file name, and file type.

Fig26-16 Adding OTA firmware

(3) Verify OTA image.
a) Click Check button to enter the OTA image verification page.

Fig26-17 Check button

b) Verify the OTA firmware.

Fig26-18 Verification button

c) Fill in the UUID to show OTA status.

Fig26-19 Inputting profile UUID

(3) Check whether OTA update is succeed or not.
If OTA update is succeed, the state upgrade is shown as Fig26-20. Otherwise, OTA upgrade is failed.

Fig26-20 OTA update result

Note:
OTA upgrade need 4M byte Flash.

OTA firmware file name must consistent with the name in duer_ota_register_install function in duerapp_ota.c
OTA firmware version should be later than the old image.

26.6 How to Debug

26.6.1 Build Error

If the following error occurs, you must reduce the heap size in project\realtek_amebaD_va0_example\inc\inc_hp\FreeRTOSConfig.h

```
/cygdrive/d/iot/ameba-D/DuerOS/AmebaZ/v03_2018_back/project/realtek_amebaD_cm4_gcc_verification/toolchain/cygwin/asdk-6.4.1/cygwin/newlib/bin/../lib/gcc/arm-none-eabi/6.4.1/../../../../arm-
none-eabi/bin/ld: region `BD_RAM_NS' overflowed by 3008 bytes
collect2: error: ld returned 1 exit status
Makefile:223: recipe for target 'linker_image2_ns' failed
make[1]: *** [linker_image2_ns] Error 1
make[1]: Leaving directory '/cygdrive/d/iot/ameba-D/DuerOS/AmebaZ/v03_2018_back/project/realtek_amebaD_cm4_gcc_verification/asdk'
Makefile:29: recipe for target 'xip' failed
make: *** [xip] Error 2
```

26.6.2 Wi-Fi Signal

Use ATWS command to check the Wi-Fi signal. If the signal is bad, remove the electric resistance shown in Fig 26-21 to check.

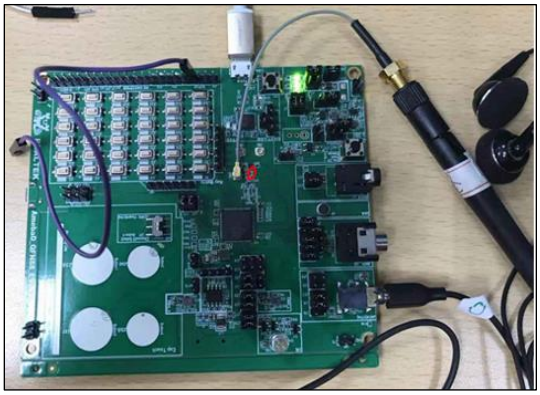


Fig 26-21 Electric resistance

26.6.3 Cannot Recognize the Recorder

(1) Check if the DMIC or AMIC is set correctly.
   a) Check if the jumpers are connected correctly, and the DMIC or AMIC is set correctly.
   b) Open project\realtek_amebaD_va0_example\inc\inc_hp\platform_opts.h and check the macro CONFIG_DMIC_SEL

(2) Check if the network is ok

```
18:57:25.976  #duerapp:     duer_conf_get_string: serverAddr = device.iot.baidu.com
18:57:30.875  duerapp: DNS failed!!!
18:57:30.879  duerapp: got ip failed!
18:57:30.882  duerapp: duer_coap_connect: transport connect failed by -52
18:57:30.885  duerapp: cached tracecode: 0x12010302
18:57:30.888  duerapp: start fail, status:0, rs:-52
18:57:30.891  duerapp: =====> event: 1[DUER_EVT_START], status: -1
18:57:30.893  duerapp: start fail! status:-1
18:57:30.896  duerapp: Action failed: event: 1, status: -1
18:57:30.900  duerapp: stop timer:0x0
18:57:30.904  duerapp: start last timer:0x0, soc:0x10043580
18:57:30.908  duerapp: no action
18:57:30.912  duerapp: =====> event: 5[DUER_EVT_STOP], status: -1
18:57:30.916  duerapp: connect stopped, status:-1!!
18:57:30.917  duerapp: event: 1
18:57:30.919  duerapp: duer_events_call_internal, handler not initialied...
18:57:30.920  duerapp: duer_events_call_internal, handler not initialied...
18:57:30.921  duerapp: duer_engine_start, g_handler:0x10053e00, length:1469, profile:0x10042690
18:57:30.923  duerapp:     duer_conf_get_string: uuid = 19c80000000001
```

Fig 26-22 Network fail log

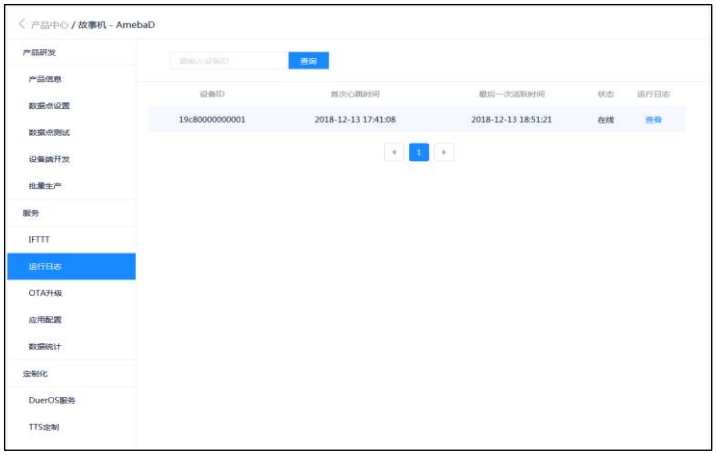
26.6.4 Checking the Memory

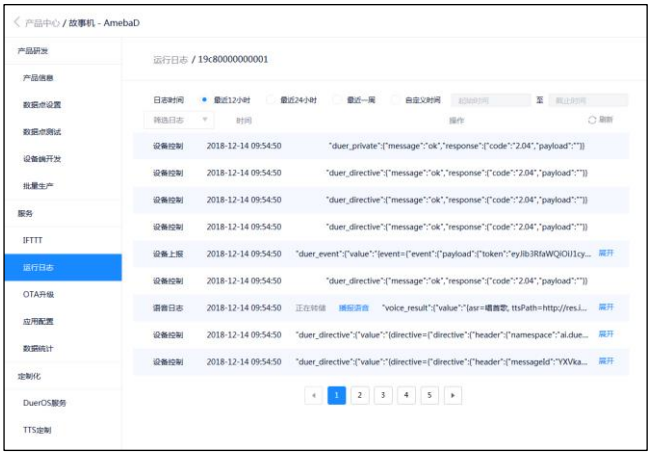
Use any command to check the heap size.

```
18:57:36.170
18:57:36.170  [MEM] After do cmd, available heap 55232
18:57:36.170
18:57:36.170
```

26.6.5 Checking the Device Status

You can see the running log on Baidu Cloud.





27 Infrared Radiation (IR)

AmebaD Infrared Radiation (IR) provides hardware modulation for Infrared Radiation sending and hardware auto capture for receiving.

This chapter introduces how to use IR.

27.1 Pinmux

The pin assignments of IR are listed in Table 27-1.

Table 27-1 IR pin assignment

Port Name	Pin Name	QFN48	QFN68	QFN88
PA[25]	IR_TX	V	V	V
PA[26]	IR_RX	V	V	V
PB[23]	IR_TX	X	V	V
PB[22]	IR_RX	X	V	V
PB[31]	IR_TX	X	V	V
PB[29]	IR_RX	X	V	V

27.2 Data Format

This section introduces the data format of Tx FIFO and Rx FIFO.

27.2.1 IR Tx

To send IR data, you ă K ž dz f Ř ø NJ ƛ ű Ŝ ű K Ŝ Ř I· ű I· ƛ y ű ž L w ¢ È C L C h NJ Ŝ Ǝ ƛ ă ű Ŝ NJ Σ ă ž ƛ ű Ω ă register. Before sending data, you should convert the data into the appropriate format that Tx FIFO register can recognized. The size of Tx FIFO register is 32 bits, where:

Bit[31] indicates data type.
- 0: inactive carrier
- 1: active carrier

Bit[30] is data end flag.
- 0: normal packet
- 1: last packet

Bit[27:0], represented by cycle, stores the data to be sent and the data format is the number of carrier cycles. [illegible] represents the carrier frequency (the unit of [illegible] is kHz); [illegible] represents the duration of carrier or no carrier symbol (the unit of [illegible] is us). Then,

[illegible]

Take NEC protocol for example, the carrier frequency is 38kHz. The format of NEC is shown in Fig27-1 and the modulation of NEC is shown in Fig27-2. The NEC format consists of 2 start symbols, 64 data symbols and 1 stop symbol.

Fig27-1 NEC format

Fig27-2 NEC modulation

¢ ƻ ăŜƴŘ ƒ ƻ∃ ƛ Ŏ ŀ ƒ ά м έ Σ ề ƻ dz ă Ķ ƻ dz ƒ Ř ǿ NJ ƛ ű Ŝ ű ǿ ƻ Ř ŀ ű ŀ ƛ ƴ ű ƻ ¢ È C L C h NJ
For the first data, bit[31] = 1, bit[30] = 0, bit[27:0] = 38*560/1000 = 21.
For the second data, bit[31] = 0, bit[30] = 0, bit[27:0] = 38*(2250-560)/1000 = 64.
¢ ƻ ăŜƴŘ ƒ ƻ∃ ƛ Ŏ ŀ ƒ ά л έ Σ ề ƻ dz ă Ķ ƻ dz ƒ Ř ǿ NJ ƛ ű Ŝ ű ǿ ƻ Ř ŀ ű ŀ ƛ ƴ ű ƻ ¢ È C L C h NJ
For the first data, bit[31] = 1, bit[30] = 0, bit[27:0] = 38*560/1000 = 21.
For the second data, bit[31] = 0, bit[30] = 0, bit[27:0] = 38*(1120-560)/1000 =21.

At last, you need to send the stop symbol. In this symbol, set bit[30] = 1 because the stop symbol indicates the data end of the current transmission.

27.2.2 IR Rx

To receive IR data, you should read the data in Rx FIFO register. ƛ ű Ω ă ƛ ŸLJ ƻ NJ ű ŀ ƴ ű ű ƻ dz ƴ Ř Ŝ NJ ă ű ŀ ƴ Ř ű Ķ Ŝ Ř ŀ register. The size of Rx FIFO register is 32 bits, where:

- Bit[31] indicates data level
  - 0: high level
  - 1: low level
- Bit[30:0] stores the data to be received and the data format is cycle duration. Let Ǫ represents the sampling frequency (the unit of Ǫ is kHz), Ÿ represents the duration of each high/low level (the unit of Ÿ is us). Then,

$$Ÿ = \frac{ὢ ''Ωὸ ŋπ}{Ǫ} \; ᶻ ρ π π π$$

With the use of Ÿ, you can do further processing to get the information you need.

27.3 APIs

27.3.1 IR Setting APIs

27.3.1.1 IR_StructInit

Items	Description
Introduction	Fills each IR_InitStruct member with its default value.
Parameters	IR_InitStruct: pointer to an IR_InitTypeDef structure which will be initialized.
Return	N/A

27.3.1.2 IR_Init

Items	Description
Introduction	Initializes the IR peripheral according to the specified parameters in the IR_InitStruct.
Parameters	IRx: selected IR peripheral. IR_InitStruct: pointer to a IR_InitTypeDef structure that contains the configuration information for the specified peripheral.
Return	N/A

27.3.1.3 IR_Cmd

Items	Description
Introduction	Enables or disables the selected IR mode.
Parameters	IRx: selected IR peripheral. mode: selected IR operation mode. This parameter can be the following values: IR_MODE_TX: Transmission mode. IR_MODE_RX: Receiving mode. NewState: new state of the operation mode. ENABLE DISABLE
Return	N/A

27.3.1.4 IR_INTConfig

Items	Description
Introduction	Enables or disables the specified IR interrupts.
Parameters	IRx: selected IR peripheral. IR_INT: specifies the IR interrupt sources to be enabled or disabled. This parameter can be one or combination of the following values: IR_TX_FIFO_EMPTY_INT_EN: Tx FIFO empty interrupt. IR_TX_FIFO_LEVEL_INT_EN: Tx FIFO threshold interrupt. IR_TX_FIFO_OVER_INT_EN: Tx FIFO overflow interrupt. IR_RX_FIFO_FULL_INT_EN: Rx FIFO full interrupt. IR_RX_FIFO_LEVEL_INT_EN: Rx FIFO threshold interrupt. IR_RX_CNT_OF_INT_EN: Rx counter overflow interrupt. IR_RX_FIFO_OF_INT_EN: Rx FIFO overflow interrupt. IR_RX_CNT_THR_INT_EN: Rx counter threshold interrupt. IR_RX_FIFO_ERROR_INT_EN: Rx FIFO error read interrupt. Trigger when Rx FIFO empty and read Rx NewState: new state of the specified IR interrupts. ENABLE DISABLE
Return	N/A

27.3.1.5 IR_MaskINTConfig

Items	Description
Introduction	Mask or unmask the specified IR interrupt.
Parameters	IRx: selected IR peripheral. IR_INT: specifies the IR interrupt sources to be mask or unmask. This parameter can be one or combina the following values: IR_TX_FIFO_EMPTY_INT_MASK: Tx FIFO empty interrupt mask. IR_TX_FIFO_LEVEL_INT_MASK: Tx FIFO threshold interrupt mask. IR_TX_FIFO_OVER_INT_MASK: Tx FIFO overflow interrupt mask. IR_RX_FIFO_FULL_INT_Msk: Rx FIFO full interrupt mask. IR_RX_FIFO_LEVEL_INT_Msk: Rx FIFO threshold interrupt mask. IR_RX_CNT_OF_INT_Msk: Rx counter overflow interrupt mask. IR_RX_FIFO_OF_INT_Msk: Rx FIFO overflow interrupt mask. IR_RX_CNT_THR_INT_Msk: Rx counter threshold interrupt mask. IR_RX_FIFO_ERROR_INT_Msk: Rx FIFO error read interrupt mask. Trigger when Rx FIFO empty Rx FIFO. NewState: new state of the specified IR interrupt mask. ENABLE DISABLE
Return	N/A

27.3.1.6 IR_GetINTStatus

Items	Description
Introduction	Get the specified IR interrupt status.
Parameters	IRx: selected IR peripheral.
Return	The new state of IR interrupt SET RESET

27.3.1.7 IR_GetIMR

Items	Description
Introduction	Get the specified IR interrupt mask status.
Parameters	IRx: selected IR peripheral.
Return	The new mask state of IR interrupt SET RESET

27.3.1.8 IR_FSMRunning

Items	Description
Introduction	Get the specified IR FSM status.
Parameters	IRx: selected IR peripheral.
Return	The new state of FSM: TRUE: RUN FALSE: IDLE

27.3.1.9 IR_ClearINTPendingBit

Items	Description
Introduction	Clears the IR interrupt pending bits.
Parameters	IRx: selected IR peripheral. IR_CLEAR_INT: specifies the interrupt pending bit to clear. This parameter can be any combination of the following values: IR_TX_FIFO_EMPTY_INT_CLR: Clear Tx FIFO empty interrupt. IR_TX_FIFO_LEVEL_INT_CLR: Clear Tx FIFO threshold interrupt. IR_TX_FIFO_OVER_INT_CLR: Clear Tx FIFO overflow interrupt. IR_RX_FIFO_FULL_INT_CLR: Clear Rx FIFO full interrupt. IR_RX_FIFO_LEVEL_INT_CLR: Clear Rx FIFO threshold interrupt. IR_RX_CNT_OF_INT_CLR: Clear Rx counter overflow interrupt. IR_RX_FIFO_OF_INT_CLR: Clear Rx FIFO overflow interrupt. IR_RX_CNT_THR_INT_CLR: Clear Rx counter threshold interrupt. IR_RX_FIFO_ERROR_INT_CLR: Clear Rx FIFO error read interrupt. Trigger when Rx FIFO empty FIFO.
Return	N/A

27.3.2 IR Tx APIs

27.3.2.1 IR_SendBuf

Items	Description
Introduction	Send data buffer.
Parameters	IRx: selected IR peripheral.

	pBuf: data buffer to send. len: buffer length. IsLastPacket: this parameter can be the following values: ENABLE: The last data in IR packet and there is no continuous data. In other words, an infrared da transmission is completed. DISABLE: There is data to be transmitted continuously.
Return	N/A

Note: the difference between IR_SendBuf() and IR_SendData() is that IR_SendBuf() send a data buffer once and the number of data is in the length of data buffer, while IR_SendData() only send one data once.

27.3.2.2 IR_SendData

Items	Description
Introduction	Send one data.
Parameters	IRx: selected IR peripheral. data: data to send.
Return	N/A

27.3.2.3 IR_SetTxThreshold

Items	Description
Introduction	Set Tx threshold. When Tx FIFO depth < threshold value, trigger interrupt.
Parameters	IRx: selected IR peripheral. thd: Tx threshold.
Return	N/A

27.3.2.4 IR_GetTxFIFOFreeLen

Items	Description
Introduction	Get free size of Tx FIFO.
Parameters	IRx: selected IR peripheral.
Return	The free size of Tx FIFO.

27.3.2.5 IR_ClearTxFIFO

Items	Description
Introduction	Clear IR Tx FIFO.
Parameters	IRx: selected IR peripheral.
Return	N/A

27.3.3 IR Rx APIs

27.3.3.1 IR_ReceiveBuf

Items	Description
Introduction	Read data from Rx FIFO.
Parameters	IRx: selected IR peripheral. pBuf: buffer address to receive data. len: read data length.
Return	N/A

Note: the difference between IR_ReceiveBuf() and IR_ReceiveData() is that IR_ReceiveBuf() read a data buffer once and the number [illegible] be larger than the data size in Rx FIFO, while IR_ReceiveData() only read one data once. The data size in Rx FIFO can be required by calling IR_GetRxDataLen().

27.3.3.2 IR_ReceiveData

Items	Description
Introduction	Read one data.
Parameters	IRx: selected IR peripheral.
Return	Data which read from Rx FIFO.

27.3.3.3 IR_SetRxThreshold

Items	Description
Introduction	Set Rx threshold. When Rx FIFO depth > threshold value, trigger interrupt
Parameters	IRx: selected IR peripheral. thd: Rx threshold.
Return	N/A

27.3.3.4 IR_GetRxDataLen

Items	Description
Introduction	Get data size in Rx FIFO.
Parameters	IRx: selected IR peripheral.
Return	Current data size in Rx FIFO.

27.3.3.5 IR_ClearRxFIFO

Items	Description
Introduction	Clear IR Rx FIFO.
Parameters	IRx: selected IR peripheral.
Return	N/A

27.3.3.6 IR_SetRxCounterThreshold

Items	Description
Introduction	Configure counter threshold value in receiving mode. You can use it to stop receiving IR data.
Parameters	IRx: selected IR peripheral. IR_RxCntThrType: This parameter can be the following values: IR_RX_Count_Low_Level: Low level counter value >= IR_RxCntThr, trigger IR_INT_RX_CNT_THR IR_RX_Count_High_Level: High level counter value >= IR_RxCntThr, trigger IR_INT_RX_CNT_TH IR_RxCntThr: Configure IR Rx counter threshold value which can be 0 to 0x7fffffff.
Return	N/A

27.3.3.7 IR_StartManualRxTrigger

Items	Description
Introduction	Start trigger only in manual receive mode. Note: manual receive mode is seldom used.
Parameters	IRx: selected IR peripheral.
Return	N/A

27.4 IR Usage

27.4.1 Sending

27.4.1.1 Tx Polling Mode

To use IR sending function, the following steps are mandatory.
(1) Configure the IR pinmux according to Table 27-1.
For example, in order to use PB[23] as IR Tx pin, call the following function. And it is the same for other IR pins.
```
Pinmux_Config(_PB_23, PINMUX_FUNCTION_IR);
```
(2) Call IR_Cmd() to disable IR.
(3) Set parameters, change some parameter if needed
```
IR_StructInit(IR_InitTypeDef *IR_InitStruct);
```
(4) Initialize hardware using the parameters in step (3).
```
IR_Init(IR_InitTypeDef *IR_InitStruct);
```
(5) Write Tx data to FIFO using IR_SendBuf() or IR_SendData().
(6) Call IR_Cmd() to enable IR to start transmission.
(7) Write more data to FIFO if needed.

Note:
In step (2) and step (6), It is suggested that disabling IR at first, and then enabling IR after writing data to FIFO.
In step (5), pay attention to convert the data into the appropriate format that Tx FIFO register can recognized before writing data to F
You can refer to section 27.2.1.

27.4.1.2 Special Notes

27.4.1.2.1 Tx FIFO Offset Issue

L Ŧ è ƨ dz ǿ ł ƴ ű ű ƨ ƨ dz Ř Ǝ Ŝ ǿ Ǩ Ŝ ű Ǩ Ŝ NJ ¢ È Ř ł ű ł ƛ ƴ C L C h Ǩ ł ă ō Ŝ ŝ ƴ ă Ŝ ƴ ű Ŏ ƨ Y LJ
TX_FIFO_OFFSET.

27.4.1.2.2 Tx Last Packet Cannot Let FSM Enter Idle Issue

if the last packet written to Tx FIFO cannot let Tx state machine enter idle, it is suggested that before enabling IR Tx, write data packets to Tx FIFO. You can refer step (2) and step (6) in section 27.4.1.1.

27.4.2 Receiving

27.4.2.1 Rx Interrupt Mode

To use IR receiving function, the following steps are mandatory.
(1) Configure the IR pinmux according to Table 27-1.
For example, in order to use PB[22] as IR Rx pin, call the following function. And it is the same for other IR pins.
```
Pinmux_Config(_PB_22, PINMUX_FUNCTION_IR);
```
(2) Set parameters, such as sampling frequency, Rx FIFO threshold level, Rx threshold type, Rx count threshold level, Rx trigger mode if needed.
```
IR_StructInit(IR_InitTypeDef *IR_InitStruct);
```
(3) Initialize hardware using the parameters in step (2).
```
IR_Init(IR_InitTypeDef *IR_InitStruct);
```
(4) Configure interrupt if needed and register interrupt callback function.
```
IR_INTConfig(IR_DEV, IR_RX_INT_ALL_EN, ENABLE);
InterruptRegister((IRQ_FUN) IR_irq_handler, IR_IRQ, (u32)NULL, 10);
InterruptEn(IR_IRQ, 10);
```
(5) Call IR_Cmd() to enable IR.

(6) Clear Rx FIFO by calling IR_ClearRxFIFO().
(7) When Rx FIFO threshold interrupt triggers, read data from Rx FIFO with the use of IR_ReceiveBuf() and IR_ReceiveData(), and m further processing in interrupt handle function.

Note:

In step(7), to decode the receiving data correctly, you should understand the data format in Rx FIFO register. You can refer to section 27.2.2

Waveform inverse issue: in Rx ending, if the waveform is inverse, you should #define INVERSE_DATA in Ir_nec_protocol.h and se IR_InitStruct.IR_RxCntThrType = IR_RX_COUNT_HIGH_LEVEL.

27.4.2.2 Rx Learning

The process of Rx learning is similar to common Rx introduced in section 27.4.1.2.1. As shown in Fig27-3, the difference is that in interrupt handle function, Rx learning should store each pulse of the Rx waveform, while common Rx only stores the carrier or un-carrier duration.

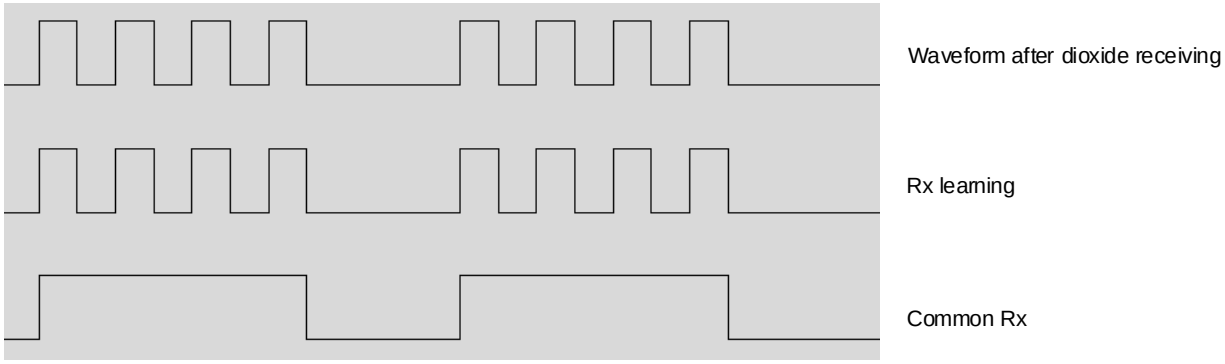


Fig27-3 Difference of waveform between Rx learning and common Rx

Note:

It is advised that putting the interrupt handle function code in RAM, and close other peripheral interrupt to avoid interfere
If the carrier frequency of learning waveform is larger than 40KHz, hardware may cannot respond to interrupt in time, which will result in decoding carrier frequency failed.

27.5 Tx Compensation Mechanism

27.5.1 Introduction

Tx waveforms are composed of some carrier symbols and no-carrier symbols. Software calculates the duration of certain symbol by application specification (such as NEC). But no carrier symbols cannot be divisible by carrier frequency accurately. If there is no compensation mechanism, the error will increase. So Tx compensation is proposed and used in IR Tx to decrease the error.

L y Y 2 ă ú ă Ŏ Ŝ y l NJ ƛ 2 ă Σ ƛ ú ƛ ă y 2 ú y Ŝ Ŏ Ŝ ă ă l NJ è ú 2 l Ř 2 LJ ú Ŏ 2 Y Jêh ă m ú 2 y Y Ŝ skip this section. Next we would introduce the application of compensation mechanism.

27.5.2 Tx Compensation Mechanism Application

Take NEC application for example, the Tx NEC waveform showing in Fig27-4 is with ¨Ω = 38kHz, and duty = 1/3. The system clock is 100MHz.

For 38kHz carrier frequency

Compensation frequency is a dependent clock, you can set to any value you want. In this example, we set 1MHz, so

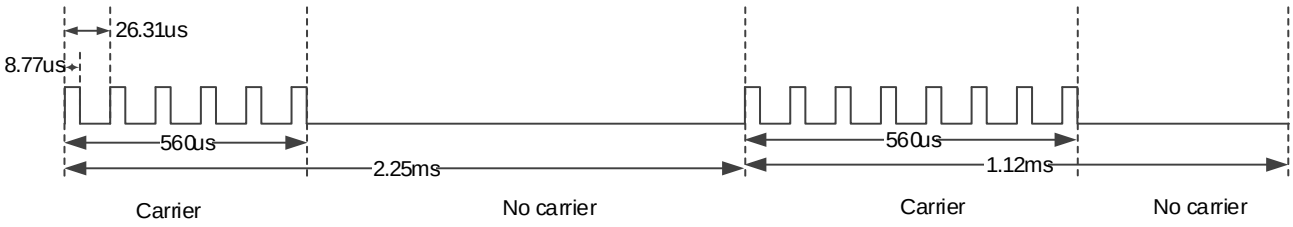


Fig27-4 Tx NEC waveform

We divided this waveform into four symbols: 560us carrier symbol, 2250us ~ 560us no carrier symbol, 560us carrier symbol, and 1120us ~ 560us no carrier symbol.

If no compensation mechanism is adopted (IR_TX_COMPENSATION = 0), the real wave can be calculated as follows.

The first carrier symbol: Referring to section 27.2.1, 560us = 21*26.31+8.77=561.28us.

The second no carrier symbol: 2250us = 64*26.31+(26.31-8.77)=1701.38us, (2250-560)us = 1690us.

The third carrier symbol is the same as the first one.

The forth no carrier symbol: 560us = 21*26.31+(26.31-8.77)=570.05us.

If compensation mechanism is adopted (IR_TX_COMPENSATION = 3), the real wave can be calculated as follows.

The first carrier symbol: Referring to section 27.2.1, 560us = 21*26.31+8.77=561.28us.

The second no carrier symbol: 2250us = 1690, (2250-560)us = 1690us = 1690 * 1us.

The third carrier symbol is the same as the first one.

The forth no carrier symbol: 560us = 560 * 1us.

Compare the above two calculating methods, we can find that using compensation mechanism, the accuracy can be improved.

Note:

Compensation mechanism can only be used for no carrier symbol.

IR_TX_COMPENSATION is bit[28:29] of IR_TX_FIFO register. Compensation method 1 (IR_TX_COMPENSATION = 1) and method 2 (IR_TX_COMPENSATION = 2) are not recommended. If you want to use compensation mechanism, refer to method 3 (IR_TX_COMPENSATION = 3).

27.6 IR Schematic Design Guideline

27.6.1 Leakage

To avoid the leakage problem, we suggest using the IR circuit which is shown in Fig27-5.

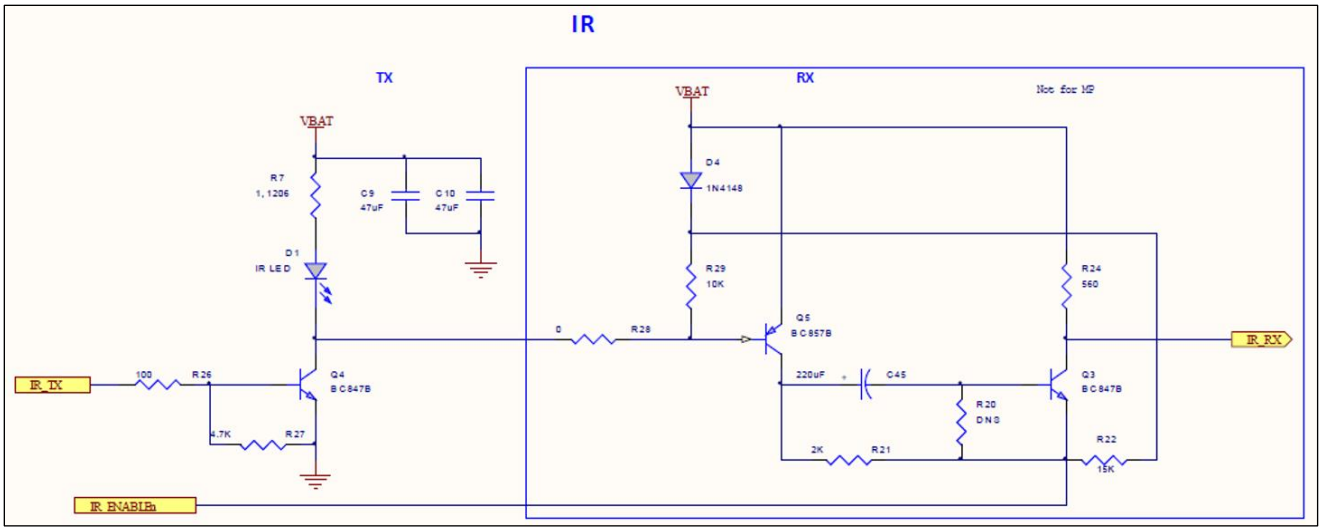


Fig27-5 Circuit of IR

27.6.2 Carrier Problem in Rx Learning

Due to the characteristics of transistor, the response speed will slow down when the transistor works in deep saturation area. Therefore, when carrier frequency is set too large, the hardware has the risk of receiving carrier waveform failed in receiving end during IR Rx learning. The maximum carrier frequency that can achieve is related to Rx circuit and the choice of transistor. The circuit we suggested in section 27.6.1 can support the maximum carrier frequency 70kHz, but you can choose the better transistor to acquire higher supported carrier frequency.

28 Brownout Detector (BOD)

BOD is mainly used to notify a user that the voltage level is low for the applications which use batteries. Ameba-D provides many thresholds to choose, the alternative levels and corresponding voltage value is shown in Table 28-1 and Table 28-2.

Note: The voltage value may have ±5% error.

Table 28-1 High threshold of interrupt mode and reset mode (3.3V/1.8V)

Voltage	Value	Symbol	Reset	Interrupt
3.3V	001	BOR_TH_HIGH1	1.887	2.297
	010	BOR_TH_HIGH2	1.970	2.397
	011	BOR_TH_HIGH3	2.061	2.507
	100	BOR_TH_HIGH4	2.139	2.602
	101	BOR_TH_HIGH5	2.224	2.704
	110	BOR_TH_HIGH6	2.315	2.815
	111	BOR_TH_HIGH7	2.388	2.904
1.8V	001	BOR_TH_HIGH1	1.422	1.498
	010	BOR_TH_HIGH2	1.480	1.560
	011	BOR_TH_HIGH3	1.543	1.627
	100	BOR_TH_HIGH4	1.598	1.684
	101	BOR_TH_HIGH5	1.656	1.746
	110	BOR_TH_HIGH6	1.719	1.812
	111	BOR_TH_HIGH7	1.770	1.865

Note: The voltage value may have ±5% error.

Table 28-2 Low threshold of interrupt mode and reset mode (3.3V/1.8V)

Voltage	Value	Symbol	Reset	Interrupt
3.3V	001	BOR_TH_LOW1	1.784	2.194
	010	BOR_TH_LOW2	1.863	2.290
	011	BOR_TH_LOW3	1.949	2.395
	100	BOR_TH_LOW4	2.023	2.486
	101	BOR_TH_LOW5	2.103	2.584
	110	BOR_TH_LOW6	2.190	2.690
	111	BOR_TH_LOW7	2.260	2.775
1.8V	001	BOR_TH_LOW1	1.345	1.422
	010	BOR_TH_LOW2	1.400	1.480
	011	BOR_TH_LOW3	1.460	1.543
	100	BOR_TH_LOW4	1.512	1.598
	101	BOR_TH_LOW5	1.567	1.656
	110	BOR_TH_LOW6	1.627	1.719
	111	BOR_TH_LOW7	1.674	1.770

Note: The voltage value may have ±5% error.

28.1 Recommended Threshold Parameter

The recommended threshold parameters are shown in Table 28-3.

Table 28-3 Recommended Threshold Parameter

Work Voltage	Reset		Interrupt	
	High Threshold	Low Threshold	High Threshold	Low Threshold
3.3V	BOR_TH_HIGH7	BOR_TH_LOW6	BOR_TH_HIGH7	BOR_TH_LOW6
1.8V	BOR_TH_HIGH5	BOR_TH_LOW3	BOR_TH_HIGH5	BOR_TH_LOW3

Note: To avoid voltage fluctuation during work trigger BOD interrupt or reset by mistake, the difference between high and low threshold can be appropriately increased

28.2 BOD APIs

28.2.1 BOR_ModeSet

Items	Description
Introduction	Choose BOD mode and enable BOD function
Parameters	Option: BOR_RESET: BOD reset mode BOR_INTR: BOD interrupt mode NewStatus ENABLE DISABLE
Return	N/A

28.2.2 BOR_ThresholdSet

Items	Description
Introduction	Set BOD high and low threshold
Parameters	Thres_Low: BOD low threshold BOR_TH_LOW1 BOR_TH_LOW2 BOR_TH_LOW3 BOR_TH_LOW4 BOR_TH_LOW5 BOR_TH_LOW6 BOR_TH_LOW7 Thres_High: BOD high threshold BOR_TH_HIGH1 BOR_TH_HIGH2 BOR_TH_HIGH3 BOR_TH_HIGH4 BOR_TH_HIGH5 BOR_TH_HIGH6 BOR_TH_HIGH7
Return	N/A

28.2.3 BOR_ClearINT

Items	Description
Introduction	Clear BOD interrupt.
Parameters	N/A
Return	N/A

28.2.4 BOR_DbncSet

Items	Description
Introduction	Set BOD interrupt mode debounce cycle
Parameters	Option: BOR_INTR: BOD interrupt mode

	Dbnc_Value: debounce cycle, in unit of ANA4M clock cycles.
Return	N/A

Note: Only BOD interrupt mode can set debounce cycle.

29 Flash Translation Layer (FTL)

29.1 Overview

NOR Flash is composed of blocks, which contain pages, and they contain individual cells of data. Flash read/write operations take place at page level. But erase operations take place at the block level. The flash needs to be erased before write. The memory portion for erasing differs in size from that for reading or writing, resulting in the major performance degradation of the overall flash memory system.

Therefore, a type of system software termed FTL has been introduced, which is provided to make the flash a friendly medium to store data. The architecture of Flash memory system is shown in Fig. 29-1.

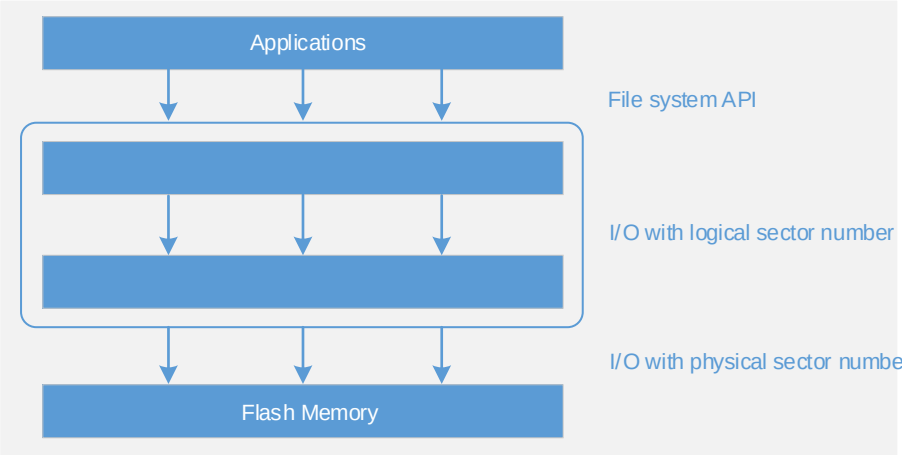


Fig. 29-1 Architecture of Flash memory system

The FTL algorithm provides the following functionalities:

- Logical-to-physical address mapping: Convert logical addresses from the file system to physical addresses in flash memory.
- Power-off recovery: Even when a sudden power-off event occurs during FTL operations, FTL data structures should be preserved and data consistency should be guaranteed.
- Wear-leveling: Wear down memory blocks as evenly as possible

As Fig. 29-2 shows, to write data to logical map, FTL would generate a data packet in specified format and store it in flash. To modify data in logical map, a new packet would be generated and appended to the end of physical map. When the physical pages are nearly full, the garbage collection is triggered to recycle the old pages which would be erased.

To read data from logical map, FTL would search the physical map to find the newest packet, which contains the data of specified address.

When using C ¢ [Σ dz ă Ŝ NJ ă Ř 2 y˝ Ω ú y˝ Ŝ Ŝ Ř ú 2 Ŏ I· NJ Ŝ I· ō 2 dz ú LJ Ƙ è ă ƛ Ŏ I· f Y˝ I· LJ Σ ǿ Ƙ ƛ Ŏ Ƙ ƛ ă

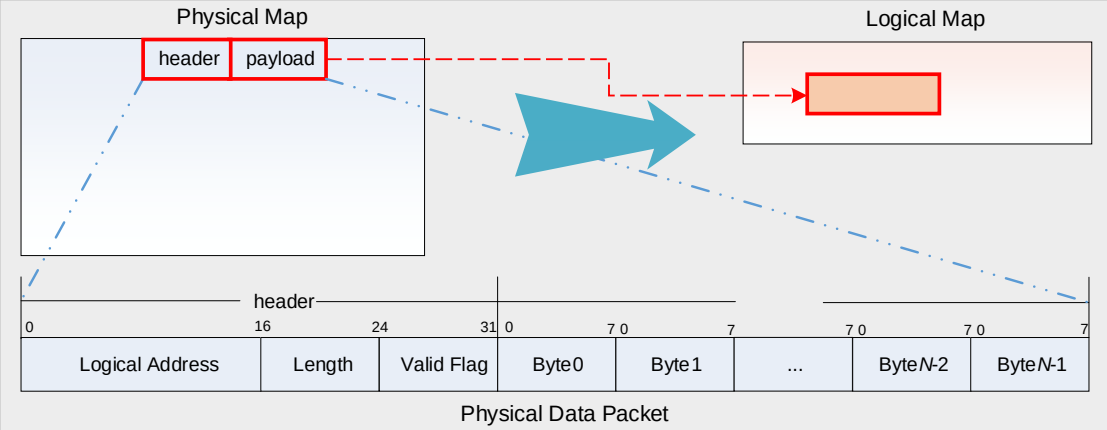


Fig29-2 FTLoverview

29.2 Features

Physical Map
- Physical page size: 4096 bytes
- Configurable physical page number
- Physical map size = 4096 * physical page number

Logical Map: The maximum logical map size determined by physical map size is (512 * (physical page number-1))-1) * 4.
Auto Garbage Collection
Abnormal Power-off Protection

29.3 FTL APIs

API	Introduction
<ftl_init>	Initializes FTL
<ftl_load_from_storage>	Gets specified length of data from logical map.
<ftl_save_to_storage>	Writes specified length of data to logical map.

29.3.1 ftl_init

Items	Description
Introduction	Initializes FTL
Parameters	u32PageStartAddr: The start address of physical map pagenum: The page number of physical map
Return	N/A

29.3.2 ftl_load_from_storage

Items	Description
Introduction	Gets specified length of data from logical map.
Parameters	pdata_tmp: Pointer to a buffer to save logical data offset: From which address read the logical map size: The number of bytes to read
Return	0: Getting logical map values successfully Others: Failed to get logical map value

29.3.3 ftl_save_to_storage

Items	Description
Introduction	Writes specified length of data to logical map.
Parameters	pdata_tmp: Pointer to byte array of new logical data offset: From which address to update the logical map size: The number of bytes to update
Return	0: Writing logical map values successfully Others: Failed to write logical map value

29.4 How to Use FTL

29.4.1 Precautions

When using FTL, you must know the precautions below.

The theoretically, the logical map area can be extended to 64KB; but the maximum usable logical map area is limited by the actual physical space is ((511 * (physical page number - 1)) - 1) * 4.

The default physical page number is 3, and it is not recommended to increase, because the default internal implement of FTL will malloc a buffer to cache the content of FTL.

For typical BLE application, each connection needs about 256 bytes FTL memory, and SDK will maintain 3 connection's information by default.

For BLE mesh, each node needs about 20 bytes FTL memory. The maximum node number can be set by dev_key_num.

Table 29-1 lists the typical scenarios of BLE, refer to it to set the appropriate offset of application according to your actual situation.

Table 29-1 Examples of recommended setting

Scenario	Calculation	Start address of application		Remark
		Configurable	Recommended	
3 connections Less than 50 mesh nodes 1KB FTL space for application	((3*256 + 20*50) + 1024) bytes (< 4084 bytes) are needed totally.	1768 ~ 3060	2500	It is recommended to reserve some memory for system extension. The recommended offset has some reservation for both system and application. For these two scenarios, 3 physical pages are enough.
5 connections 2KB FTL space for application	(5*256 + 2048) bytes (< 4084 bytes) are needed totally.	1280 ~ 2036	1600	
5 connections Less than 60 mesh nodes 6KB FTL space for application	((5*256 + 20*60) + 6144) bytes are needed totally.	-	-	It is not recommended to store so much information in FTL. Try to find another solution, such as DCT.

29.4.2 General Steps

To use FTL in AmebaD, the following steps are necessary.

(1) Define the macro CONFIG_FTL_ENABLED by yourself and set it to 1, the FTL initialization will be contained in main.

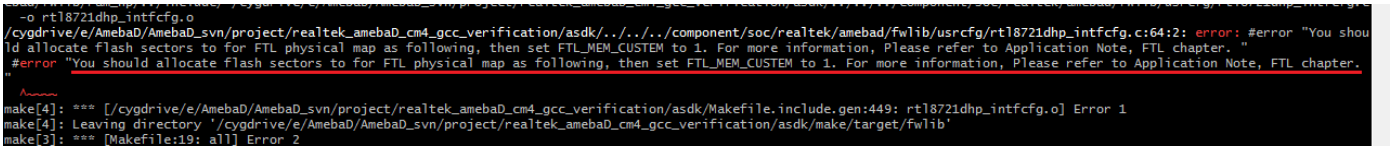
```
#define CONFIG_FTL_ENABLED    1
```

If Bluetooth is enabled in your system, CONFIG_FTL_ENABLED is defined automatically after defining CONFIG_BT_EN

(2) Define FTL_MEM_CUSTEM to 1 in rtl8721dhp_intfcfg.c

```
62 #if defined(CONFIG_FTL_ENABLED)
63: #define FTL_MEM_CUSTEM      1
64 #if FTL_MEM_CUSTEM == 0
65: #error "You should allocate flash sectors to for FTL physical map as following, then set FTL_MEM_CUSTEM to 1. For more information, Please refer to Application Note, FTL chapter. "
66 #else
67: const u8 ftl_phy_page_num = 3;                          /* The number of physical map pages, default is 3*/
68: const u32 ftl_phy_page_start_addr = 0x00102000;        /* The start offset of flash pages which is allocated to FTL physical map.
69:                                                            Users should modify it according to their own memory layout! ! */
70: #endif
71: #endif
```

Note: If the memory layout is modified and overwrite FTL area 0x0810_2000~0x0810_4FFF, you also need to modify the physical storage address. Otherwise, an error would be thrown out to remind you.



The default configurations for logical & physical map is as follows:

3 pages allocated for physical map: 0x0810_2000~0x0810_4FFF
Logical map size: 4084 bytes

(3) Call ftl_load_from_storage/ftl_save_to_storage functions to read from/write to logical map.

30 Audio Signal Generation and Analysis

30.1 Introduction

When developing digital mic (DMIC) related applications, you may need to generate a signal of a certain frequency to test whether DMIC works well or not. This chapter introduces how to generate a signal and how to analyze the frequency of a signal collected by the DMIC first. Then, digital analog-analog digital (DAAD) loopback is explained. DAAD loopback is helpful in determining which part (codec or DMIC [illegible] work well when problems occur.

30.1.1 Compilation

You need to add the example_audio_signal_generate to your project before you can use all the commands mentioned in this chapter. Related File used are placed under the path: /component/common/example/audio_sport/audio_signal_generate.

Follow two steps to use this function.

(1) [illegible] platform_opts.h under the path: /project/ realtek_amebaD_va0_example/ inc/ inc_hp.

(2) In Cygwin terminal, change to the directory /project/ realtek_amebaD_va0_example/ GCC-RELEASE/ project_hp, type make menuconfig, and enable audio related configurations (MENUCONFIG FOR CHIP CONFIG -> Audio Config -> Enable Audio).

30.1.2 Generating a Signal of a Certain Frequency

Frequencies of signals supported by the program range from 20Hz ~ 20000Hz, which is the frequency range that people can hear.

Type [illegible] (20 < f < 20000) [illegible]

30.1.3 Analyzing Signals Collected by DMIC

One way of testing a DMIC is to analyze the signals collected by the DMIC in frequency domain. If the detected frequency is the same as the frequency of the signal, the DMIC works well.

Type [illegible] to analyze the signal collected by DMIC in frequency domain. The main frequency and its relative strength are printed out. Due to frequency leakage, the frequency detected may not be the same as the one you play, but the number should be close. You can use this command to analyze signals whose frequencies range from 20~20000Hz.

30.1.4 DAAD

DMIC related problems can be caused by internal codec or DMIC itself. This section explains how to use DAAD loopback to determine whether the internal codec works in a normal way or not.

Fig 30-1 depicts how DAAD loopback works. The data transmitted from DA Digital IP is received directly by AD Digital IP without the interference of DMIC.

Type [illegible] to generate a signal whose frequency is f. The main frequency detected and relative strength calculated will be printed out. If the frequency detected is close to the one you generate, it means that the internal codec works well. The problem may be caused by DMIC.

Fig 30-1 DAAD loopback diagram

30.1.5 Command

All the commands that can be used are listed in Table 30-1. Each command is composed of several parameters and the parameters which are surrounded by ( ) should be set by users.

Table 30-1 Available commands in audio signal generation and analysis

Command Format	Command Function	Example
Audio_generate tone spkr(f)(	Generater	