



# AN0400

## Ameba-D Application Note

---



Realtek Semiconductor Corp.

No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan

Tel.: +886-3-578-0211. Fax: +886-3-577-6047

[www.realtek.com](http://www.realtek.com)

**COPYRIGHT**

©2021 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

**DISCLAIMER**

Please Read Carefully:

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

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

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

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

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

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

Realtek's Resources are provided "as is" and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding resources or use thereof, including but not limited to accuracy or completeness, title, any epidemic failure warranty and any implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third party intellectual property rights.

Realtek shall not be liable for and shall not defend or indemnify Customer against any claim, including but not limited to any infringement claim that related to or is based on any combination of products even if described in Realtek Resources or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek's Resources or use thereof, and regardless of whether Realtek has been advised of the possibility of such damages. Realtek is not responsible for any failure to meet such industry standard requirements.

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

Customers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Customers' own risk. Customers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

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

**TRADEMARKS**

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

**USING THIS DOCUMENT**

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

# Contents

**Contents**.....4

**Convention** .....12

**1 Building Environment** .....13

1.1 Introduction.....13

1.2 Preparing GCC Environment.....13

    1.2.1 Windows.....13

    1.2.2 Linux.....14

1.3 Building Code.....15

    1.3.1 Normal Image.....15

    1.3.2 MP Image.....17

1.4 Setting Debugger.....17

    1.4.1 Probe.....17

    1.4.2 J-Link.....22

1.5 Downloading Image to Flash.....27

1.6 Entering Debug Mode.....27

1.7 Command List.....28

1.8 GDB Debugger Basic Usage.....29

1.9 Q & A.....29

    1.9.1 “Error 127” for Building Code.....29

    1.9.2 “Permission denied” Error under Linux.....30

    1.9.3 How to Reset KM0/KM4 under Debug Mode?.....30

**2 SDK Architecture** .....32

2.1 component.....32

    2.1.1 common.....32

    2.1.2 os.....33

    2.1.3 soc.....33

2.2 doc.....33

2.3 tools.....33

2.4 GCC Project for KM4.....34

2.5 Critical Header Files.....34

**3 GCC Makefile** .....35

3.1 KM4 Makefile Architecture.....35

3.2 How to Build Code into Flash?.....35

3.3 How to Build Code into SRAM?.....36

3.4 How to Use Section Attribute?.....37

3.5 How to Build Library?.....38

3.6 How to Add Library?.....39

**4 C++ Standards Supported in GCC** .....40

4.1 Introduction.....40

4.2 To Compile C++ Codes.....40

**5 GCC Standard Library**.....42

- 5.1 Introduction..... 42
- 5.2 Default Use of Library Function ..... 42
- 5.3 To Use Configurable Function in GCC Standard Library ..... 43
- 5.4 Tips ..... 44
- 6 IAR Build Environment Setup..... 45**
  - 6.1 Requirement..... 45
    - 6.1.1 IAR Embedded Workbench ..... 45
    - 6.1.2 J-Link or RLX Probe..... 45
  - 6.2 Hardware Configuration ..... 45
    - 6.2.1 Connecting with J-Link ..... 46
    - 6.2.2 Connecting with RLX Probe..... 47
  - 6.3 How to Use IAR SDK?..... 48
    - 6.3.1 IAR Project Introduction ..... 48
    - 6.3.2 IAR Build ..... 49
    - 6.3.3 IAR Download ..... 53
    - 6.3.4 IAR Debug ..... 57
    - 6.3.5 IAR Memory Configuration ..... 62
  - 6.4 How to Build Sample Code? ..... 63
  - 6.5 Used Memory Size Calculation ..... 64
    - 6.5.1 Memory Section..... 64
    - 6.5.2 Memory Size ..... 65
- 7 Demo Board ..... 67**
  - 7.1 PCB Layout Overview..... 67
  - 7.2 Pin Out ..... 68
  - 7.3 DC Power Supply ..... 69
  - 7.4 USB Interface Configuration ..... 70
  - 7.5 LOGUART ..... 70
  - 7.6 SWD ..... 71
  - 7.7 VBAT ADC..... 71
- 8 ImageTool..... 72**
  - 8.1 Introduction..... 72
  - 8.2 Environment Setup ..... 72
    - 8.2.1 Hardware Setup..... 72
    - 8.2.2 Software Setup ..... 73
  - 8.3 Download ..... 73
    - 8.3.1 Image Download ..... 73
    - 8.3.2 Flash Erase..... 74
    - 8.3.3 Flash Status Operation ..... 75
  - 8.4 Generate..... 78
    - 8.4.1 Merge Images..... 78
    - 8.4.2 Generate Cloud OTA Image ..... 78
- 9 Memory Layout ..... 79**
  - 9.1 Flash Program Layout ..... 79
    - 9.1.1 Image Header ..... 80
    - 9.1.2 System Data (4K) ..... 81
    - 9.1.3 User Data..... 81
    - 9.1.4 4M Flash Usage Example..... 82
  - 9.2 SRAM Layout ..... 83
    - 9.2.1 KM4 SRAM Layout ..... 83

- 9.2.2 KMO SRAM Layout ..... 84
- 9.3 PSRAM Layout ..... 85
- 9.4 Retention SRAM ..... 86
  - 9.4.1 Retention SRAM Layout ..... 86
  - 9.4.2 User Area ..... 86
- 9.5 OTA Layout ..... 87
  - 9.5.1 OTA Program Layout ..... 87
  - 9.5.2 OTA Header Layout ..... 88
- 10 File System ..... 88**
  - 10.1 Introduction ..... 88
  - 10.2 FatFs on Flash ..... 88
    - 10.2.1 Software Setup ..... 88
    - 10.2.2 FatFs Bin File Generation ..... 89
  - 10.3 FatFs on SD Card ..... 89
    - 10.3.1 Hardware Setup ..... 89
    - 10.3.2 Software Setup ..... 90
- 11 Inter Processor Communication (IPC) ..... 91**
  - 11.1 Introduction ..... 91
  - 11.2 How to Use IPC? ..... 91
  - 11.3 IPC Program APIs ..... 92
    - 11.3.1 ipc\_table\_init ..... 92
    - 11.3.2 ipc\_send\_message ..... 92
    - 11.3.3 ipc\_get\_message ..... 92
  - 11.4 IPC Driver Code ..... 92
    - 11.4.1 IPC\_INTConfig ..... 92
    - 11.4.2 IPC\_IERSet ..... 92
    - 11.4.3 IPC\_IERGet ..... 93
    - 11.4.4 IPC\_INTRRequest ..... 93
    - 11.4.5 IPC\_INTClear ..... 93
    - 11.4.6 IPC\_INTGet ..... 93
    - 11.4.7 IPC\_CPUID ..... 93
    - 11.4.8 IPC\_SEMGet ..... 94
    - 11.4.9 IPC\_SEMFree ..... 94
    - 11.4.10 IPC\_INTHandler ..... 94
    - 11.4.11 IPC\_INTUserHandler ..... 94
- 12 OTA Firmware Update ..... 95**
  - 12.1 Introduction ..... 95
  - 12.2 RSIP-MMU ..... 95
  - 12.3 OTA Upgrade from Local Server ..... 96
    - 12.3.1 Firmware Format ..... 97
    - 12.3.2 OTA Flow ..... 97
    - 12.3.3 Boot Option ..... 98
    - 12.3.4 Address Remapping ..... 99
    - 12.3.5 How to Use OTA Demo? ..... 99
    - 12.3.6 OTA Firmware Swap ..... 101
  - 12.4 User Configuration ..... 102
- 13 eFuse ..... 104**
  - 13.1 Introduction ..... 104
  - 13.2 Power Requirement ..... 104

- 13.3 eFuse Auto-load ..... 105
- 13.4 Physical eFuse ..... 105
- 13.5 Logical eFuse ..... 105
  - 13.5.1 Logical eFuse Layout ..... 105
  - 13.5.2 Wi-Fi 2.4G Power Index ..... 106
  - 13.5.3 Wi-Fi 5G Power Index ..... 107
  - 13.5.4 Wi-Fi Channel Plan ..... 108
  - 13.5.5 Wi-Fi Crystal Calibration ..... 109
  - 13.5.6 Wi-Fi Thermal Meter ..... 109
  - 13.5.7 Wi-Fi MAC Address ..... 110
  - 13.5.8 Cap-Touch ..... 110
  - 13.5.9 BLE ..... 110
  - 13.5.10 HCI USB ..... 111
- 13.6 eFuse PG APIs ..... 111
  - 13.6.1 Low Level APIs ..... 111
  - 13.6.2 Mbed APIs ..... 112
- 13.7 eFuse PG Command ..... 113
- 14 Hardware Crypto Engine ..... 114**
  - 14.1 Introduction ..... 114
  - 14.2 Hardware Crypto Engine APIs ..... 114
    - 14.2.1 Crypto Engine Initialization API ..... 114
    - 14.2.2 Hash APIs ..... 114
    - 14.2.3 Cipher APIs ..... 119
  - 14.3 Hardware Crypto Engine APIs Usage ..... 122
    - 14.3.1 Starting Hardware Crypto Engine ..... 122
    - 14.3.2 Starting Crypto Engine Calculation ..... 122
  - 14.4 Demo Code Path ..... 123
- 15 User Configuration ..... 124**
  - 15.1 Configuration File List ..... 124
  - 15.2 flashcfg.c ..... 124
    - 15.2.1 Flash Classification ..... 125
    - 15.2.2 FlashClass1 ~ FlashClass6 ..... 129
  - 15.3 pinmapcfg.c ..... 132
  - 15.4 sleepcfg.c ..... 133
  - 15.5 bootcfg.c ..... 134
  - 15.6 ipccfg.c ..... 136
  - 15.7 lp\_intfcfg.c ..... 137
  - 15.8 hp\_intfcfg.c ..... 137
- 16 Flash Operation ..... 138**
  - 16.1 Functional Description ..... 138
  - 16.2 Protection Method ..... 138
  - 16.3 Flash Raw APIs ..... 139
    - 16.3.1 Read ..... 139
    - 16.3.2 Write ..... 140
    - 16.3.3 Erase ..... 140
    - 16.3.4 Receive/Transmit Command ..... 140
  - 16.4 Flash Mbed APIs ..... 141
  - 16.5 User Configuration ..... 141
- 17 Battery Measurement ..... 142**

17.1	Functional Description .....	142
17.2	Calibration .....	142
<b>18</b>	<b>Wi-Fi.....</b>	<b>144</b>
18.1	Wi-Fi Data Structures .....	144
18.2	Wi-Fi APIs .....	144
18.2.1	System APIs .....	144
18.2.2	Scan APIs .....	145
18.2.3	Connection APIs.....	146
18.2.4	Channel APIs.....	148
18.2.5	Power APIs.....	149
18.2.6	AP Mode APIs .....	150
18.2.7	Custom IE APIs.....	152
18.2.8	Wi-Fi Setting APIs .....	152
18.2.9	Wi-Fi Indication APIs.....	156
18.2.10	eFuse Writing APIs .....	157
18.3	Fast Connection .....	157
18.3.1	Implement .....	157
18.3.2	APIs.....	158
18.4	WPS APIs .....	158
18.4.1	wps_start.....	158
18.4.2	wps_stop .....	158
<b>19</b>	<b>Liquid Crystal Display Controller (LCDC) .....</b>	<b>159</b>
19.1	Interface .....	159
19.2	Resolution .....	159
19.3	Pinmux .....	160
19.4	LCDC APIs .....	160
19.4.1	MCU Function .....	160
19.4.2	RGB Function .....	161
19.4.3	LED Function.....	162
19.4.4	Common Function.....	162
19.5	How to Use LCDC .....	164
19.5.1	MCU Interface .....	164
19.5.2	RGB Interface.....	166
19.5.3	LED Interface .....	166
19.6	GUI .....	167
19.6.1	Authorization.....	167
19.6.2	emWin Software.....	167
19.6.3	How to Adapt to LCM .....	168
19.6.4	How to Adapt to Touch Panel.....	169
19.6.5	How to Use emWin in SDK.....	170
<b>20</b>	<b>PSRAM .....</b>	<b>171</b>
20.1	Throughput .....	171
20.2	Power Management .....	172
20.3	How to Use PSRAM? .....	172
20.3.1	Initializing PSRAM.....	172
20.3.2	Adding BSS/TEXT/DATA Section into PSRAM Region.....	172
20.3.3	Allocating Heap from PSRAM .....	172
20.4	PSRAM Cache “Write Back” Policy Change Note .....	173
20.4.1	Cache Policy Change.....	173
20.4.2	Scope .....	173



- 20.4.3 Notice ..... 173
- 21 MPU and Cache ..... 175**
  - 21.1 Functional description ..... 175
    - 21.1.1 MPU..... 175
    - 21.1.2 Cache..... 175
  - 21.2 MPU APIs..... 176
    - 21.2.1 mpu\_init ..... 176
    - 21.2.2 mpu\_set\_mem\_attr..... 176
    - 21.2.3 mpu\_region\_cfg ..... 176
    - 21.2.4 mpu\_entry\_free..... 176
    - 21.2.5 mpu\_entry\_alloc..... 177
  - 21.3 Cache APIs..... 177
    - 21.3.1 ICache\_Enable ..... 177
    - 21.3.2 ICache\_Disable ..... 177
    - 21.3.3 ICache\_Invalidate ..... 177
    - 21.3.4 DCache\_IsEnabled ..... 177
    - 21.3.5 DCache\_Enable..... 177
    - 21.3.6 DCache\_Disable ..... 178
    - 21.3.7 DCache\_Invalidate..... 178
    - 21.3.8 DCache\_Clean..... 178
    - 21.3.9 DCache\_CleanInvalidate..... 178
  - 21.4 How to Define a Non-cacheable Data Buffer ..... 178
- 22 Audio Codec Controller Guide ..... 179**
  - 22.1 Audio Codec ..... 179
    - 22.1.1 Diagram..... 179
    - 22.1.2 Features..... 180
    - 22.1.3 Application Mode ..... 180
  - 22.2 Audio Codec Controller ..... 183
    - 22.2.1 Features..... 183
    - 22.2.2 Control Interface..... 183
  - 22.3 Audio PLL..... 183
    - 22.3.1 Diagram..... 184
    - 22.3.2 Operation Mode ..... 185
  - 22.4 Audio Codec APIs ..... 185
    - 22.4.1 PLL APIs..... 185
    - 22.4.2 SPORT APIs ..... 186
    - 22.4.3 SI APIs ..... 189
    - 22.4.4 Codec APIs ..... 189
  - 22.5 How to Use AC APIs?..... 191
    - 22.5.1 Audio Play..... 191
    - 22.5.2 Audio Record ..... 192
    - 22.5.3 Example List..... 192
  - 22.6 Hardware Design Guide ..... 193
    - 22.6.1 Line-out..... 193
    - 22.6.2 AMIC-in..... 193
    - 22.6.3 Power..... 194
  - 22.7 Performance of Encoding & Decoding ..... 194
    - 22.7.1 AC3 Format..... 194
    - 22.7.2 OPUS Format ..... 194
    - 22.7.3 FLAC Format ..... 195
    - 22.7.4 AAC Format ..... 195

- 22.7.5 MP3 Format..... 195
- 22.8 Q & A..... 196
  - 22.8.1 How to Connect the Output of DAC to A Power Amplifier?..... 196
  - 22.8.2 What’s the Difference between Single-ended MIC Input and Differential MIC Input?..... 196
  - 22.8.3 How to Play Local Audio Files?..... 196
- 23 Cap-Touch ..... 197**
  - 23.1 Pinmux ..... 197
  - 23.2 APIs ..... 197
    - 23.2.1 CapTouch\_StructInit ..... 197
    - 23.2.2 CapTouch\_Init..... 197
    - 23.2.3 CapTouch\_Cmd..... 197
    - 23.2.4 CapTouch\_INTConfig ..... 198
    - 23.2.5 CapTouch\_GetISR ..... 198
    - 23.2.6 CapTouch\_INTClearPendingBit..... 198
  - 23.3 How to Use CTC..... 198
    - 23.3.1 CTC Initialization..... 198
    - 23.3.2 CTC Calibration ..... 199
  - 23.4 Cap-Touch Schematic Design and PCB Layout Guidelines ..... 199
    - 23.4.1 Cap-Touch Schematic Design..... 199
    - 23.4.2 PCB Layout..... 200
- 24 Infrared Radiation (IR)..... 203**
  - 24.1 Pinmux ..... 203
  - 24.2 Data Format ..... 203
    - 24.2.1 IR Tx..... 203
    - 24.2.2 IR Rx..... 204
  - 24.3 APIs ..... 204
    - 24.3.1 IR Setting APIs..... 204
    - 24.3.2 IR Tx APIs ..... 206
    - 24.3.3 IR Rx APIs..... 207
  - 24.4 IR Usage ..... 209
    - 24.4.1 Sending..... 209
    - 24.4.2 Receiving ..... 209
  - 24.5 Tx Compensation Mechanism..... 210
    - 24.5.1 Introduction..... 210
    - 24.5.2 Tx Compensation Mechanism Application ..... 210
  - 24.6 IR Schematic Design Guideline..... 211
    - 24.6.1 Leakage ..... 211
    - 24.6.2 Carrier Problem in Rx Learning..... 212
- 25 Brownout Detector (BOD) ..... 213**
  - 25.1 Recommended Threshold Parameter ..... 213
  - 25.2 BOD APIs ..... 214
    - 25.2.1 BOR\_ModeSet ..... 214
    - 25.2.2 BOR\_ThresholdSet..... 214
    - 25.2.3 BOR\_ClearINT ..... 214
    - 25.2.4 BOR\_DbncSet..... 214
- 26 Flash Translation Layer (FTL) ..... 216**
  - 26.1 Overview ..... 216
  - 26.2 Features ..... 217
  - 26.3 FTL APIs ..... 217

- 26.3.1 *ftl\_init* ..... 217
- 26.3.2 *ftl\_load\_from\_storage* ..... 217
- 26.3.3 *ftl\_save\_to\_storage*..... 218
- 26.4 How to Use FTL ..... 218
  - 26.4.1 *Precautions*..... 218
  - 26.4.2 *General Steps*..... 218
- 27 Audio Signal Generation and Analysis ..... 220**
  - 27.1 Introduction ..... 220
    - 27.1.1 *Compilation* ..... 220
    - 27.1.2 *Generating a Signal of a Certain Frequency* ..... 220
    - 27.1.3 *Analyzing Signals Collected by DMIC*..... 220
    - 27.1.4 *DAAD* ..... 220
    - 27.1.5 *Command* ..... 221
- 28 Key-Scan..... 222**
  - 28.1 Pinmux ..... 222
  - 28.2 APIs ..... 222
    - 28.2.1 *keyscan\_array\_pinmux*..... 222
    - 28.2.2 *keyscan\_getdatanum* ..... 222
    - 28.2.3 *keyscan\_read*..... 222
    - 28.2.4 *keyscan\_init*..... 223
    - 28.2.5 *keyscan\_set\_irq*..... 223
    - 28.2.6 *keyscan\_clear\_irq*..... 223
    - 28.2.7 *keyscan\_get\_irq\_status*..... 223
    - 28.2.8 *keyscan\_set\_irq\_handler*..... 223
    - 28.2.9 *keyscan\_enable* ..... 223
    - 28.2.10 *keyscan\_disable* ..... 224
    - 28.2.11 *keyscan\_clearifodata*..... 224
  - 28.3 Key-Scan Usage ..... 224
    - 28.3.1 *Using Default Configuration* ..... 224
    - 28.3.2 *Using APIs*..... 225
- 29 Bluetooth ..... 226**
  - 29.1 Overview ..... 226
  - 29.2 BT Examples ..... 226
    - 29.2.1 *GCC Project* ..... 226
    - 29.2.2 *IAR Project* ..... 227
  - 29.3 BT Debug ..... 227
    - 29.3.1 *Introduction*..... 227
    - 29.3.2 *Hardware & Software Preparation*..... 227
    - 29.3.3 *Prerequisite – Opening the Log Switch*..... 228
    - 29.3.4 *Capturing Log Files* ..... 229
    - 29.3.5 *FAQ*..... 232
- Revision History..... 234**

# Convention

Ameba-D consists of two MCUs:

- Real-M300: a high performance MCU (Armv8-M, Cortex-M33 instruction set compatible) called KM4 thereafter
- Real-M200: a low power MCU (Armv8-M, 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.
- During the installation of Cygwin package, include 'Devel -> make' and 'Math -> bc' utilities on the **Select Packages** step, as Fig 1-1 and Fig 1-2 shows.

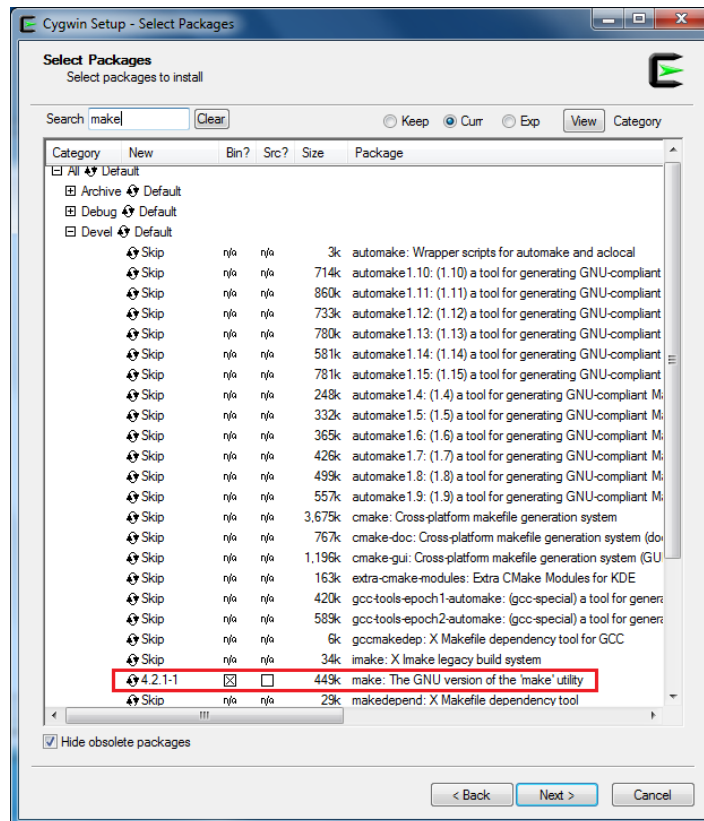


Fig 1-1 'Devel' setting during Cygwin installation

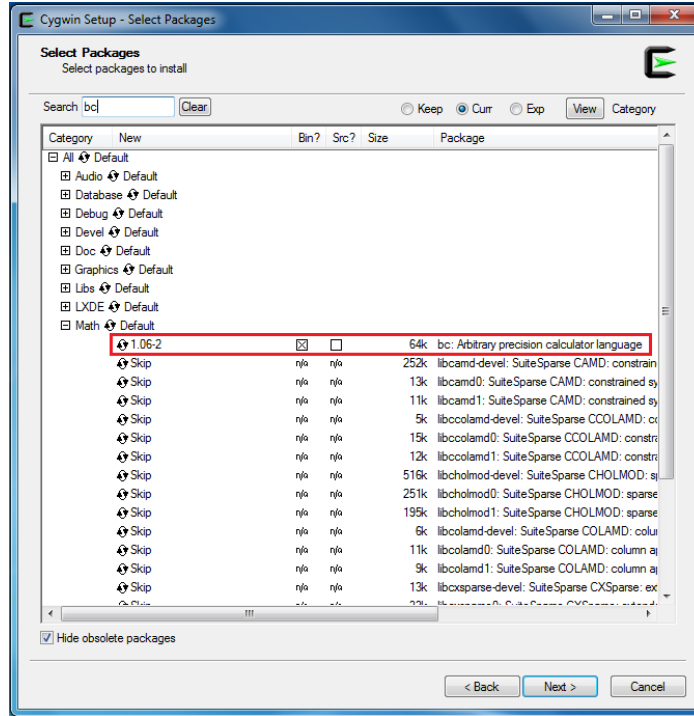


Fig 1-2 'Math' setting during Cygwin installation

## 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 pre-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** isn't installed, type **apt-get install make** to install **make**.

```

serik@realtek:~$ make -v
GNU Make 4.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
    
```

- **\$ bc -v**

If **bc** isn't installed, type **apt-get install bc** to install **bc**.

```

serik@realtek:~$ bc -v
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
Foundation, Inc.
    
```

- **\$ gawk --v**

If **gawk** isn't installed, type **apt-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 KMO or KM4 project directory of Ameba-D SDK.
  - Note:** You need to replace the *{path}* to your own SDK location, and add “cygdrive” prefix in front of the SDK location, so 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 KMO or KM4 project directory of Ameba-D 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 KMO Project

For KMO project, if the terminal contains “km0\_image2\_all.bin” and “Image manipulating end” output message, it means that the image has been built successfully, as Fig 1-3 shows.

```

/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp
./project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/xip_image2.bin __flash_text_start__ /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/target_img2.map /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/gnu_utility/prepend_header.sh /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/ram_retention.bin __retention_entry_func__ /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/target_img2.map cat /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/xip_image2_prepend.bin /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/ram_retention_prepend.bin > /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/km0_image2_all.bin /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/gnu_utility/pad.sh /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk/image/km0_image2_all.bin
===== Image manipulating end =====
make[1]: Leaving directory /cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk
/cygdrive/d/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp
$

```

Fig 1-3 KMO project make all

If somehow it is built failed, type **\$ make clean** to clean and then redo the make procedure. After successfully built, the image file is located in **project/realtek\_amebaD\_va0\_example/GCC-RELEASE/project\_lp/asdk/image**, as Fig 1-4 shows.

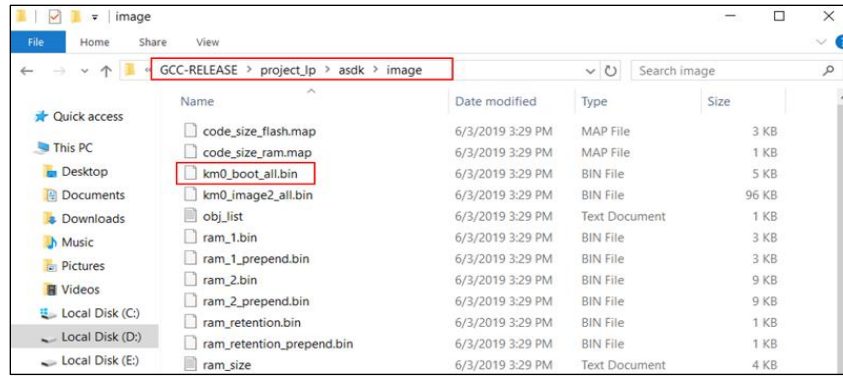


Fig 1-4 KM0 project bin generation

### 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 Fig 1-5 shows.

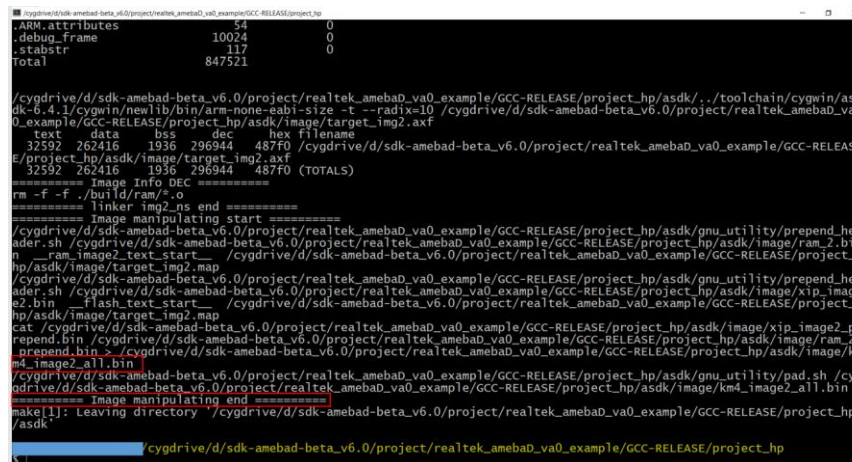


Fig 1-5 KM4 project make all

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 Fig 1-6 shows.

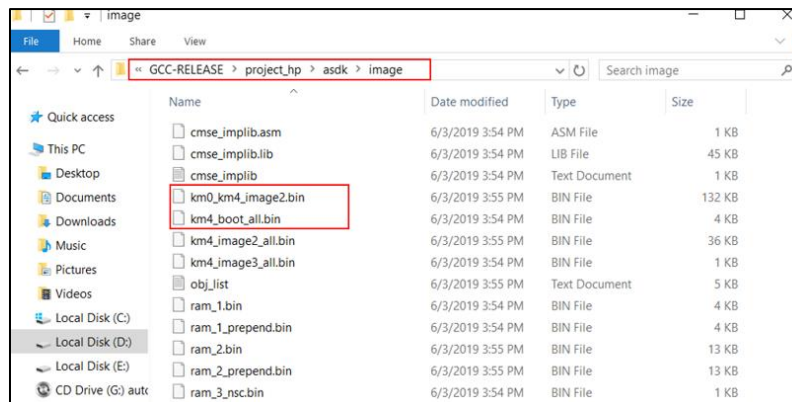


Fig 1-6 KM4 project bin generation



## 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 Fig 1-7.

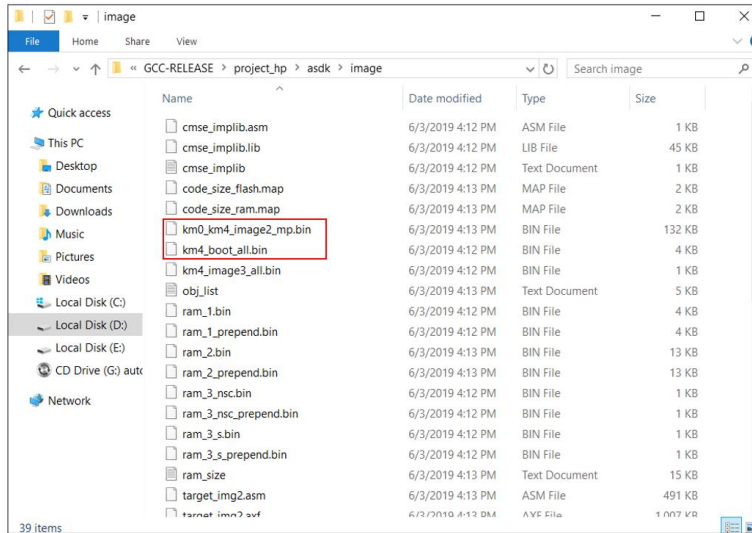


Fig 1-7 MP image generation

## 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 J-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

Ameba-D Device Board supports Probe debugger. We can use Probe to download the software and enter GDB debugger mode under GCC environment. Refer to Fig 1-8 to connect Probe debugger to the SWD of Ameba-D.

(1) Install Probe driver

Before using the Probe, its driver is required to be installed. Obtain the **RLX\_Probe\_Driver\_2.3.11p22\_Setup.exe** under `tools/probe`, then install it correctly. Refer to Fig 1-8 to connect Probe debugger to the SWD of Ameba-D, which is connecting TCK pin of Probe to SWD CLK pin of Ameba-D, and TMS pin of Probe to SWD DATA pin of Ameba-D. What's more, a common ground is needed between Probe Board and Ameba-D Device Board.

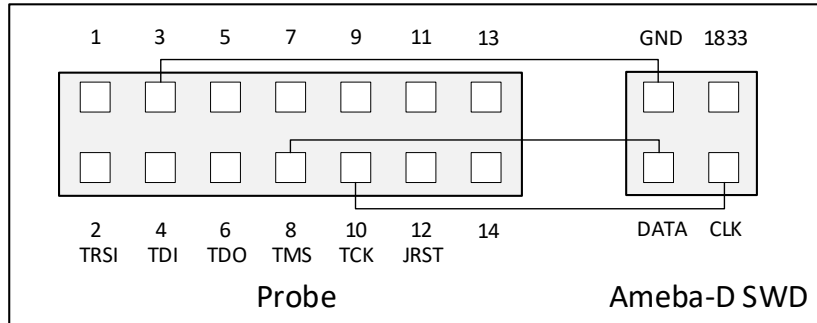


Fig 1-8 Wiring diagram of connecting Probe to SWD

(2) Execute the **cm0\_RTL\_Probe.bat**

After the installation of the software pack, 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 Fig 1-9. This window should NOT be closed if you want to download the image or enter debug mode.

```

C:\Windows\system32\cmd.exe
system reset-halted, DHCSR 0x02030003
This Core is implemented by Realtek Semiconductor Corporation.
This is a KM0 processor! KM0 found CPUID = 0x721cd200, Reversion is r1p0!
DCB_DHCSR 0x00030003;
NUIC_DFSR 0x00000009;
NUIC_AIRCR 0xfa052000; little-endian;
DCB_DCRSR 0x00000000;
DCB_DCRDR 0x00000000;
DWT_CYCCNT 0x00000000;
DWT_MASK0 0x00000000;
DWT_FUNCTION0 0x50000000;
This MCU has has a Flash Patch Breakpoint version 2!
This MCU has 2 of code comparators and 0 literal comparators in FPB!
Remapping not supported!
DWT_CTRL 0x1b000000;
DWT_CYCCNT 0x00000000;
DWT_COMP0 0x00000000;
This MCU has 1 comparators!
This MCU supports Trace sampling and exception tracing!
This MCU does not include external match signals, CMPMATCH[NI]!
This MCU supports a cycle counter!
This MCU supports the profiling counters!
Program Flow prediction disabled! Instruction caches disabled! Data and unified caches disabled!
Cache is implemented at level 1, and is Separate Instruction & Data Caches!
Level 1 I-Cache infomation:
I-Cache supports Write-Through!
I-Cache supports Write-Back!
I-Cache supports Read-Allocation!
I-Cache does not support Write-Allocation!
The processor's I-cache number of sets is 256!
The processor's I-cache number of ways is 2!
The processor's I-cache number of linesize is 8 Words
The processor's I-cache size is 16 KB!
Level 1 D-Cache infomation:
D-Cache supports Write-Through!
D-Cache supports Write-Back!
D-Cache supports Read-Allocation!
D-Cache supports Write-Allocation!
The processor's D-cache number of sets is 64!
The processor's D-cache number of ways is 2!
The processor's D-cache number of linesize is 8Words
The processor's D-cache size is 4KB!
*****/
IAP 1 Core 0 Initialize Over !
SUCCESS to Trigger this Core
*****/
    
```

Fig 1-9 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 Fig 1-10 shows.

```

/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 /cygdrive/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'

```

Fig 1-10 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 Fig 1-11. This window should NOT be closed if you want to enter debug mode.

```

C:\Windows\system32\cmd.exe
This Core supports Non-Secure Invasive Debug !
DHCSR = 0x00130003
This Core is in NON-Secure state, concurrent write to CDS would be ignored!
Now Access by Secure Request!
KM4 halted!
This Core is implemented by Realtek Semiconductor Corporation.
This is a KM4 processor!KM4 found CPUID = 0x721fd220, Revision is r1p0!
DCB_DHCSR 0x00130003;
NUIC_DFSR 0x00000001;
NUIC_AIBCR 0xfa052000; little-endian;
DCB_DCSR 0x00000000;
DCB_DCRDR 0x00000000;
DMT_CYCCNT 0x00000000;
DMT_MASK0 0x00000000;
DMT_FUNCTION0 0x58000000;
SAU information:
SAU ATTR: Memory is marked as Secure and is not Non-secure callable!
The MPU is Enabled!
MPU Region 0: 0x100268c0 ~ 0x1002c63f is marked as privileged access per
mitted only, Non-shareable, Read/write by any privilege level, Execution only pe
rmitted if read permitted, Normal memory, Outer Non-cacheable, Normal memory, In
ner Non-cacheable
MPU Region 1: 0x1010a000 ~ 0x1014001f is marked as privileged access per
mitted only, Non-shareable, Read/write by any privilege level, Execution only pe
rmitted if read permitted, Normal memory, Outer Non-cacheable, Normal memory, In
ner Non-cacheable
MPU Region 2: 0x101c0000 ~ 0x101d401f is marked as privileged access per
mitted only, Non-shareable, Read/write by any privilege level, Execution only pe
rmitted if read permitted, Normal memory, Outer Non-cacheable, Normal memory, In
ner Non-cacheable
MPU Region 3: 0x000c0000 ~ 0x000c041f is marked as privileged access per
mitted only, Non-shareable, Read/write by any privilege level, Execution only pe
rmitted if read permitted, Normal memory, Outer Non-cacheable, Normal memory, In
ner Non-cacheable
This MCU has a Flash Patch Breakpoint version 2!
This MCU has 2 of code comparators and 0 literal comparators in FPB!
Remapping not supported!
DMT_CTRL 0x1b000000;
DMT_CYCCNT 0x00000000;
DMT_COMP0 0x00000000;
This MCU has 1 comparators!
This MCU supports Trace sampling and exception tracing!
This MCU does not include external match signals, CMPMATCH [N]!
This MCU supports a cycle counter!
This MCU supports the profiling counters!
Program flow prediction disabled!Instruction caches enabled!Data and unified cac
hes enabled!
There is no D-cache in this processor!
/*****
TAP 1 Core 0 Initialize Over !
SUCCESS to Trigger this Core
*****/

```

Fig 1-11 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 **rlx\_probe0.cfg** under the folder **/project/realtek\_amebaD\_va0\_example/GCC-RELEASE/project\_hp/jlink\_script** and rename it as **rlx\_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 Fig 1-12 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/
project_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'
    
```

Fig 1-12 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\_v2.3.11p22** 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's more, a common ground is needed between Probe Board and Ameba-D Device Board.

After the installation of the software pack, use `ln -s` 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/
    
```

Fig 1-13 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 Fig 1-14.

```
wlan5@RS-wlan5: ~/V03/V03_bak/project/realtek_amebaD_cm0_verification/jlink_script
File Edit View Search Terminal Help
DAURHSTATUS = 0x0000000f!
This Core does not support Secure Non-Invasive Debug !
This Core does not support Secure Invasive Debug !
This Core supports Non-Secure Non-Invasive Debug !
This Core supports Non-Secure Invasive Debug !
DHCSR = 0x00030003
This Core is in NON-Secure state, concurrent write to CDS would be ignored!
Now Access by Non-Secure Request!
KMO halted!
This core is implemented by Realtek Semiconductor Corporation.
This is a KM0 processor!KMO found CPUID = 0x721cd200, Revision is rip0!
VCB_VCSCR 0x00000000;
NVIC_DFSR 0x00000001;
NVIC_AIRCR 0xfa052000; little-endian;
DCB_DCSR 0x00000000;
DCB_DCRDR 0x00000000;
DWT_CVCCNT 0x00000000;
DWT_MASK0 0x00000000;
DWT_FUNCTION0 0x50000000;
This MCU has a Flash Patch Breakpoint version 2!
This MCU has 2 of code comparators and 0 literal comparators in FPB!
Remapping not supported!
DWT_CTRL 0x1b000000;
DWT_CVCCNT 0x00000000;
DWT_CMP0 0x00000000;
This MCU has 1 comparators!
This MCU supports trace sampling and exception tracing!
This MCU does not include external match signals, CMPMATCH[N]!
This MCU supports a cycle counter!
This MCU supports the profiling counters!
Program flow prediction disabled!Instruction caches enabled!Data and unified caches enabled!
Cache is implemented at level 1, and is Separate Instruction & Data Caches!
Level 1 I-Cache information:
I-Cache supports Write-Through!
I-Cache supports Write-Back!
I-Cache supports Read-Allocation!
I-Cache does not support Write-Allocation!
The processor's I-cache number of sets is 256!
The processor's I-cache number of ways is 2!
The processor's I-cache number of linesize is 8 Words
The processor's I-cache size is 16 KB!
Level 1 D-Cache information:
D-Cache supports Write-Through!
D-Cache supports Write-Back!
D-Cache supports Read-Allocation!
D-Cache supports Write-Allocation!
The processor's D-cache number of sets is 64!
The processor's D-cache number of ways is 2!
The processor's D-cache number of linesize is 8Words
The processor's D-cache size is 4KB!
/*****
TAP 1 Core 0 Initialize Over !
SUCCESS to Trigger this Core
*****/
```

Fig 1-14 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 Fig 1-15.

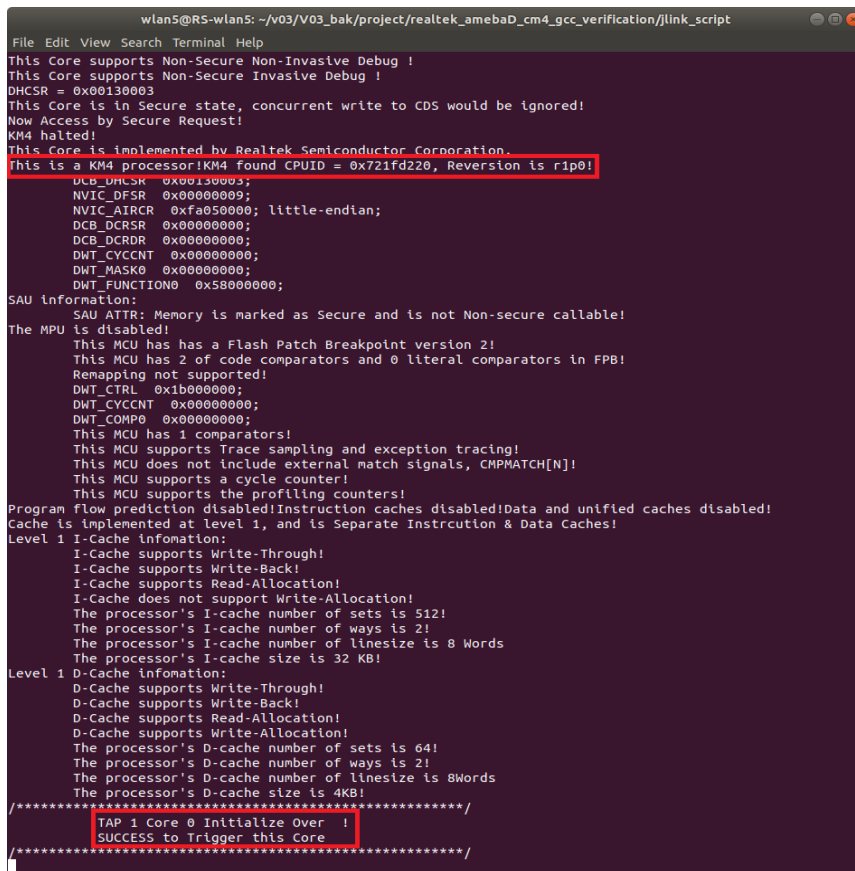
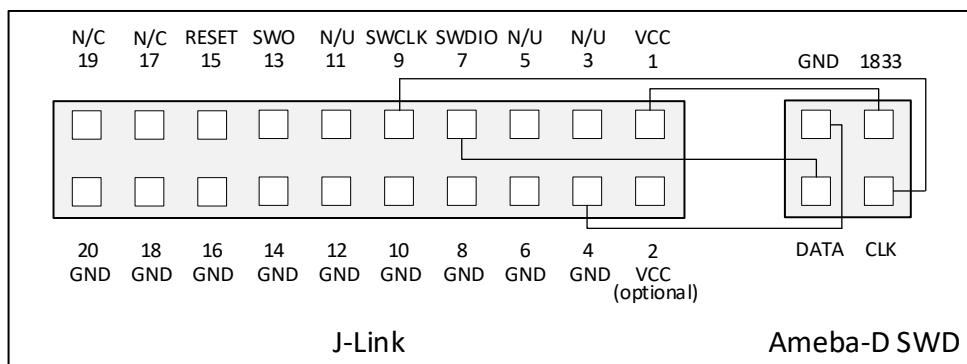


Fig 1-15 KM4 probe GDB server connection success under Linux

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

Ameba-D also supports J-Link debugger. You need to do some hardware configuration to use J-Link debugger. Refer to Fig 1-16 to connect J-Link to the SWD of Ameba-D, which is connecting SWCLK pin of J-Link to SWD CLK pin of Ameba-D, and SWDIO pin of J-Link to SWD DATA pin of Ameba-D. After finishing these configurations, connect it to PC.



DATA CLK |

Fig 1-16 Wiring diagram of connecting J-Link to SWD

**Note:** For Ameba-D 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/GCCRELEASE/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 the image or enter debug mode.

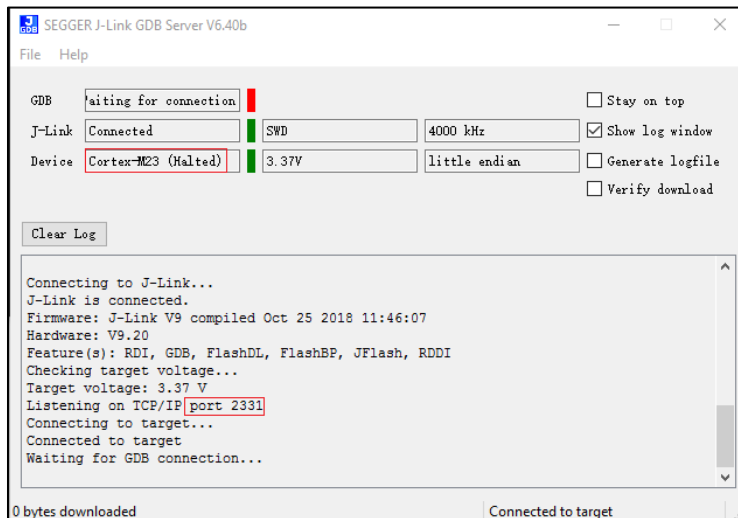


Fig 1-17 KMO J-Link GDB server connection under Windows

- (3) Setup J-Link for KMO  
 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-ameba-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-ameba-beta_v6.0/project/realtek_
amebaD_va0_example/GCC-RELEASE/project_lp/asdk'
-----
Setup jlink
-----
cp -p /cygdrive/d/sdk-ameba-beta_v6.0/project/realtek_amebaD_va0_example/GCC-
RELEASE/project_lp/asdk/gnu_utility/gnu_script/rtl_gdb_debug_jlink.txt /cygdriv
e/d/sdk-ameba-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-ameba-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-ameba-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-ameba-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-ameba-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-ameba-beta_v6.0/project/realtek_a
mebaD_va0_example/GCC-RELEASE/project_lp/asdk'

```

Fig 1-18 KM0 J-Link setup under Windows

- (4) Execute the **cm4\_jlink.bat**  
 Execute the **cm4\_jlink.bat** under **/project/realtek\_amebaD\_va0\_example/GCCRELEASE/project\_hp/jlink\_script** the same as executing the **cm0\_jlink.bat**. The started J-Link GDB server looks like Fig 1-19. This window should NOT be closed if you want to download the image or enter debug mode.

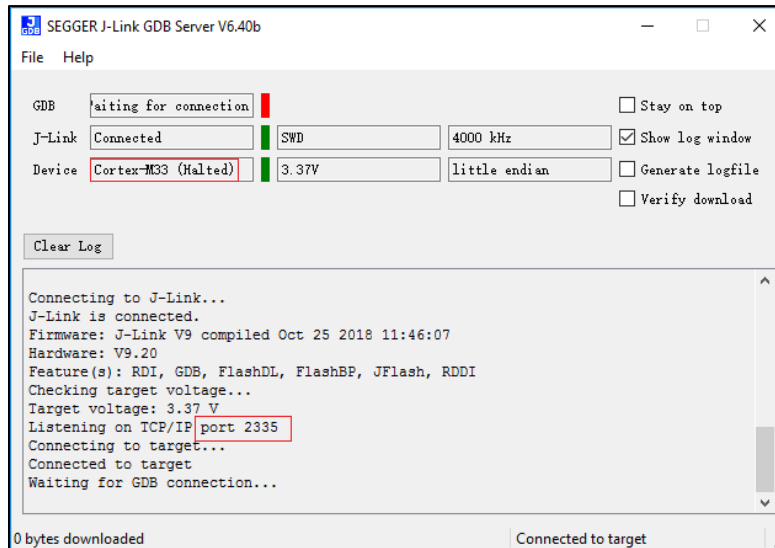


Fig 1-19 KM4 J-Link GDB server connection under Windows

- (5) Setup J-Link for KM4  
 On the Cygwin terminal, type **\$ make setup GDB\_SERVER=jlink** command to select J-Link debugger, as Fig 1-20 shows.

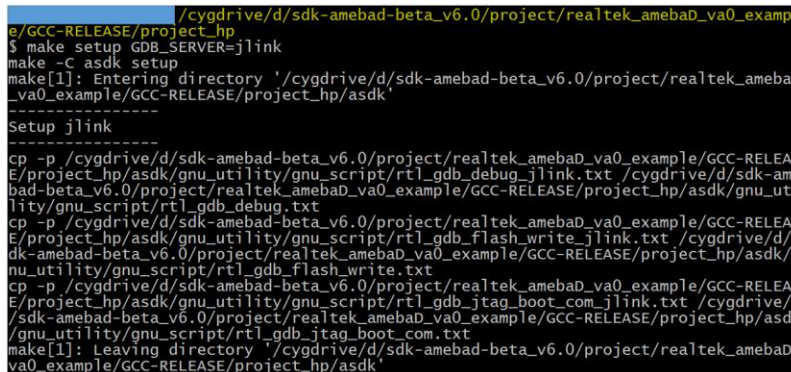


Fig 1-20 KM4 J-Link setup under Windows

### 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”. We suggest using Debian package manager to install the Debian version.

- **\$ dpkg -i jlink\_6.0.7\_x86\_64.deb**

After the installation of the software pack, there is a tool named “JLinkGDBServer” under JLink directory. Take Ubuntu 18.04 as an example, the 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 KMO Project

For KMO, 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 Fig 1-21 shows. If connection is successful, the log is shown as Fig 1-22.

```
wlan5@RS-wlan5:~/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-REL
EASE/project_lp/jlink_script$ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m2
3 -if SWD -scriptfile AP1_KM0.JLinkScript -port 2331
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-m23 -if SWD -scriptfile AP1_KM0.JLinkScript -port 2
331
-----GDB Server start settings-----
GDBInit file: none
GDB Server Listening port: 2331
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-----
```

Fig 1-21 KMO J-Link 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: AP1_KM0.JLinkScript
J-Link settings file: none
-----Target related settings-----
Target device: cortex-m23
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: 3.38 V
Listening on TCP/IP port 2331
Connecting to target... Connected to target
Waiting for GDB connection...
```

Fig 1-22 KMO J-Link 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 Fig 1-23 shows.

```
wlan5@RS-wlan5:~/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 '/home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/asdk'
Setup jlink
cp -p /home/wlan5/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_lp/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_lp/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_lp/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_lp/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_lp/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_lp/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_lp/asdk'
```

Fig 1-23 KMO J-Link 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 Fig 1-24 shows. If connection is successful, the log is shown as Fig 1-25.

```
wlan5@RS-wlan5:~/sdk-amebad-beta_v6.0/project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp/jlink_script$ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m33 -if SWD -scriptfile AP2_KM4.JLinkScript -port 2335
SEGGER J-Link GDB Server V6.46b Command Line Verston

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

Command line: -device cortex-m33 -if SWD -scriptfile AP2_KM4.JLinkScript -port 2335
-----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-----
```

Fig 1-24 KM4 J-Link 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...
```

Fig 1-25 KM4 J-Link 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 Fig 1-26 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 /home/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.txt /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'
```

Fig 1-26 KM4 J-Link terminal setup under Linux

## 1.5 Downloading Image to Flash

This section illustrates how to download image to Flash.

To download software into Ameba-D 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 codes by this command. This command downloads 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 codes success log

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

**Note:** For Probe downloading,

- Make chip enter download mode before downloading code to Flash
- Download KM4 project code also through **cm0\_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 resetting 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.

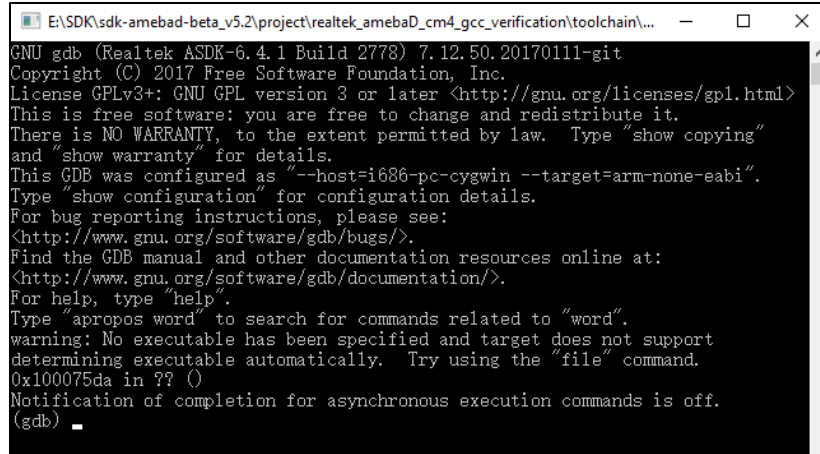


Fig 1-28 Debug window under Windows

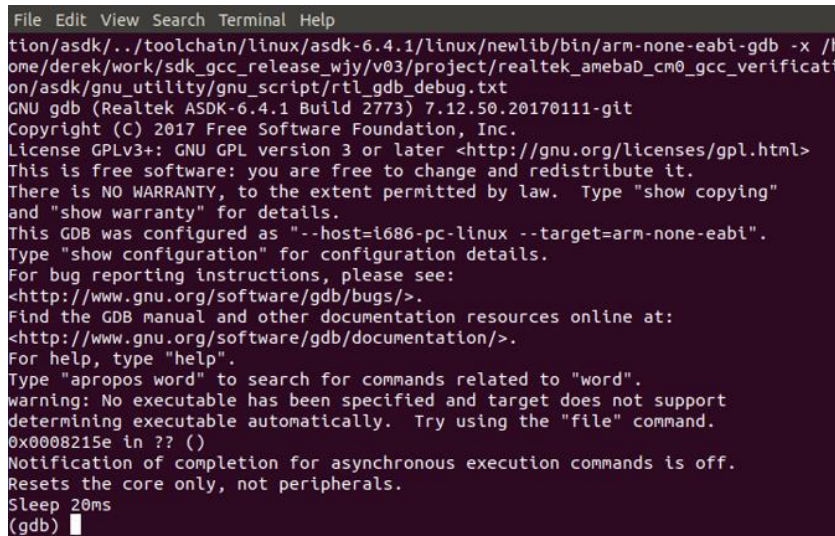


Fig 1-29 Debug window under Linux

## 1.7 Command List

The commands used above are listed in Table 1-1.

Table 1-1 Command list

Usage	Command	Description
all	\$ make all	Compile project to generate ram_all.bin
setup	\$ make setup GDB_SERVER= jlink	Select GDB_SERVER
flash	\$ make flash	Download ram_all.bin to Flash
clean	\$ make clean	Remove compile file (*.bin, *.o, ...)
clean_all	\$ make clean_all	Remove compile result and toolchains
debug	\$ make debug	Enter debug 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 <i>break</i> command (abbreviated <i>b</i> ). The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Breaks.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Breaks.html</a>
Watchpoint	\$ watch	You can use a watchpoint to stop execution whenever the value of an expression changes. The related commands include <i>watch</i> , <i>rwatch</i> , and <i>awatch</i> . The usage of these commands can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Watchpoints.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Watchpoints.html</a> <b>Note:</b> 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 <i>info</i> command. You can simply type <i>info</i> to know its usage.
Delete breakpoints	\$ delete	To eliminate the breakpoints, use the <i>delete</i> command (abbreviated <i>d</i> ). The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Delete-Breaks.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Delete-Breaks.html</a> .
Continue	\$ continue	To resume program execution, use the <i>continue</i> command (abbreviated <i>c</i> ). The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html</a> .
Step	\$ step	To step into a function call, use the <i>step</i> command (abbreviated <i>s</i> ). It will continue running your program until control reaches a different source line. The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html</a> .
Next	\$ next	To step through the program, use the <i>next</i> command (abbreviated <i>n</i> ). The execution will stop when control reaches a different line of code at the original stack level. The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html</a> .
Quit	\$ quit	To exit GDB debugger, use the <i>quit</i> command (abbreviated <i>q</i> ), or type an end-of-file character (usually <i>Ctrl-d</i> ). The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Quitting-GDB.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Quitting-GDB.html</a> .
Backtrace	\$ backtrace	A backtrace is a summary of how your program got where it is. You can use <i>backtrace</i> command (abbreviated <i>bt</i> ) to print a backtrace of the entire stack. The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Backtrace.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Backtrace.html</a> .
Print source lines	\$ list	To print lines from a source file, use the <i>list</i> command (abbreviated <i>l</i> ). The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/List.html">https://sourceware.org/gdb/current/onlinedocs/gdb/List.html</a> .
Examine data	\$ print	To examine data in your program, you can use <i>print</i> command (abbreviated <i>p</i> ). It evaluates and prints the value of an expression. The usage can be found at <a href="https://sourceware.org/gdb/current/onlinedocs/gdb/Data.html">https://sourceware.org/gdb/current/onlinedocs/gdb/Data.html</a> .

## 1.9 Q & A

### 1.9.1 “Error 127” for Building Code

If you use \$ **make all** command to build code but encounter “Error 127” error message like Fig 1-30, it’s caused by incorrect Cygwin package. Both 32-bit and 64-bit operating system need to install 32-bit installation Cygwin Package – setup-x86.exe.

```

../component/common/application/apple/homekit/crypto/poly1305 -I/cygdrive/f/v03
/project/realtek_amebaD_cm0_gcc_verification/asdk/../../../../component/common/appl
ication/apple/homekit/crypto/ed25519 -I/cygdrive/f/v03/project/realtek_amebaD_cm
0_gcc_verification/asdk/../../../../component/common/application/apple/homekit/cryp
to/ed25519/core -I/cygdrive/f/v03/project/realtek_amebaD_cm0_gcc_verification/as
dk/../../../../component/common/application/apple/homekit/crypto/sha512 -DCONFIG_PL
ATFORM_8721D /cygdrive/f/v03/project/realtek_amebaD_cm0_gcc_verification/asdk/f
lashloader/rtl_flash_download.c -o rtl_flash_download.o
make[2]: *** [I/cygdrive/f/v03/project/realtek_amebaD_cm0_gcc_verification/asdk/f
lashloader/rtl_flash_download.o] Error 127
make[2]: Leaving directory '/cygdrive/f/v03/project/realtek_amebaD_cm0_gcc_verif
ication/asdk/make/Flashloader'
make[1]: *** [Makefile:87: make_subdirs_flashloader] Error 2
make[1]: Leaving directory '/cygdrive/f/v03/project/realtek_amebaD_cm0_gcc_verif
ication/asdk'
make: *** [Makefile:15: all] Error 2
    
```

Fig 1-30 “Error 127” error message

### 1.9.2 “Permission denied” Error under Linux

If you use \$ make all command to compile project but encounter “Permission denied” error message like Fig 1-31, it’s 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.

```

===== Image manipulating start =====
/home/jesse/sdk-amebaD-beta_v4_20180914/project/realtek_amebaD_cm0_gcc
_verification/asdk/gnu_utility/prepend_header.sh /home/jesse/sdk-ameba
D-beta_v4_20180914/project/realtek_amebaD_cm0_gcc_verification/asdk/im
age/ram_1.bin __ram_start_table_start__ /home/jesse/sdk-amebaD-beta
_v4_20180914/project/realtek_amebaD_cm0_gcc_verification/asdk/image/tar
get_loader.map
make[1]: execvp: /home/jesse/sdk-amebaD-beta_v4_20180914/project/real
tek_amebaD_cm0_gcc_verification/asdk/gnu_utility/prepend_header.sh: Per
mission denied
Makefile:163: recipe for target 'linker_loader' failed
make[1]: *** [linker_loader] Error 127
make[1]: Leaving directory '/home/jesse/sdk-amebaD-beta_v4_20180914/pr
oject/realtek_amebaD_cm0_gcc_verification/asdk'
Makefile:15: recipe for target 'all' failed
make: *** [all] Error 2
    
```

Fig 1-31 “Permission denied” error message under Linux

### 1.9.3 How to Reset KM0/KM4 under Debug Mode?

Steps to reset KM0 and KM4 are different.

- For KM0, you need to use the “monitor reset” instruction in corresponding rtl\_gdb\_debug.txt to reset it under debug. The default path of this file is under \project\realtek\_amebaD\_va0\_example\GCC-RELEASE\project\_lp\asdk\gnu\_utility\gnu\_script\rtl\_gdb\_debug.txt. Please change the default setting which is “#monitor reset 1” to “monitor reset”.
- For KM4, you need to use the “monitor reset” instruction in corresponding rtl\_gdb\_debug.txt first (this process is similar to the process 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 this operation, the KM4 can't jump out of the boot function. Details of this operation are as follows: After executing “make debug” instruction, a debug window pops up. In this window. Set bit[25] to 1, then type “c”, the KM4 is reset. refer to Fig 1-32 for more information.

**Note:** Only resetting KM0 may cause KM4 to work in an abnormal way because KM0 reboot will change some settings of KM4. If you find KM4 doesn't work after KM0 reboot, you should reset KM4.

```

D:\Project\Ameba_D\Task\release\sdk-amebad-beta_v5.2\sdk-amebad-beta_v5.2\project\r...
GNU gdb (Realtek ASDK-6.4.1 Build 2778) 7.12.50.20170111-git
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later (http://gnu.org/licenses/gpl.html)
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
warning: No executable has been specified and target does not support
determining executable automatically. Try using the "file" command.
0x0e007564 in ?? ()
Notification of completion for asynchronous execution commands is off.
Resetting target.
(gdb) x/lwx 0x480003f8
0x480003f8: 0x00000201
(gdb) set (unsigned)0x480003f8=0x02000201
(gdb) b main
Breakpoint 1: file main.c, line 211.
(gdb) c
Continuing.

Breakpoint 1, main () at main.c:211
211  <
(gdb) n
212      if (wifi_config.wifi_ultra_low_power &&
(gdb) n
216      InterruptRegister(IPC_INTHandler, IPC_IRQ, IPCM0_DEU, 10);
(gdb) n
217      InterruptEn(IPC_IRQ, 10);
(gdb) c
Continuing.

```

Fig 1-32 KM4 “monitor reset” setting

## 2 SDK Architecture

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

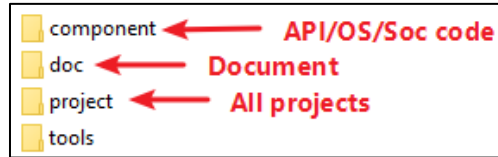


Fig 2-1 SDK architecture

### 2.1 component

#### 2.1.1 common

Items	Description
api	<ul style="list-style-type: none"> <li>● AT command</li> <li>● Platform_stdlib.h: standard library header</li> <li>● Wi-Fi driver interface</li> <li>● Wi-Fi promisc mode interface</li> <li>● Wi-Fi simple configuration</li> </ul>
application	<ul style="list-style-type: none"> <li>● DuerOS</li> <li>● MQTT</li> </ul>
audio	Audio related
bluetooth	Internal BT driver
drivers	<ul style="list-style-type: none"> <li>● alc5616/ alc5640/ alc5651/ alc5660/ alc5679/ alc5680/ sgtl5000 drivers</li> <li>● IR NEC driver</li> <li>● SDIO host driver</li> <li>● Ameba-D internal codec rl6548 driver</li> <li>● USB host and device drivers</li> <li>● WLAN driver</li> </ul>
example	Utility examples: wlan_fast_connect/ssl_download/ fatfs/mdns/media_geo_mp4 ...
file_system	File system
grahpic	JPEG decoder
mbed	mbed API source code
media	Multi-media framework
network	<ul style="list-style-type: none"> <li>● coap</li> <li>● dhcp</li> <li>● http</li> <li>● lwip</li> <li>● mDNS</li> <li>● rtsp</li> <li>● sntp</li> <li>● ssl (MBEDTLS)</li> <li>● tftp</li> <li>● websocket</li> </ul>
test	<ul style="list-style-type: none"> <li>● WLAN test file</li> </ul>
ui	<ul style="list-style-type: none"> <li>● emWin</li> <li>● littlevGL</li> </ul>
utilities	<ul style="list-style-type: none"> <li>● cJSON</li> <li>● http_client</li> <li>● ssl_client</li> </ul>



	<ul style="list-style-type: none"> <li>● tcpecho</li> <li>● udpecho</li> <li>● webserver/xml</li> </ul>
video	Video related

### 2.1.2 os

Items	Description
freertos	FreeRTOS source code
os_dep	<ul style="list-style-type: none"> <li>● osdep_service.c: Realtek encapsulating interface for FreeRTOS</li> <li>● osdep_service.h: Realtek encapsulating interface header</li> </ul>

### 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 file like ota/pmu...
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 Ameba-D
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_3.4b	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.

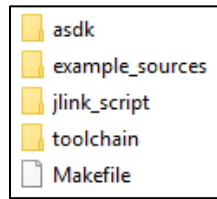


Fig 2-2 Architecture of KM4

Items	Description
asdk	<ul style="list-style-type: none"> <li>● Menuconfig</li> <li>● Link script</li> <li>● Library: lib_wlan.a/lib_wps.a/rom_symbol ...</li> <li>● Makefile</li> </ul>
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/BIT(x) ...
platform_stdlib.h	Standard library API: memcmp/strcpy/atoll/strpbrk/sscanf/printf/sprintf/snprintf/vsnprintf ...
section_config.h	Section definition used in link script: IMAGE2_RAM_TEXT_SECTION ...
mbed api headers	component/common/mbed/ ... <ul style="list-style-type: none"> <li>● Peripheral header files for mbed APIs</li> <li>● If you want to use mbed APIs, related headers must be included.</li> </ul>
ameba_soc.h	<ul style="list-style-type: none"> <li>● Peripheral header files for raw APIs</li> <li>● Raw APIs have more features than mbed APIs, mbed APIs just have basic features.</li> <li>● If you want to use raw APIs, this header must be included.</li> </ul>

## 3 GCC Makefile

### 3.1 KM4 Makefile Architecture

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



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 in project/SRAM is 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 attribute "IMAGE2\_RAM\_TEXT\_SECTION" to locate some of your functions into SRAM.

```
#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 should include "section\_config.h"
- const/read-only data will build into Flash
- Non read-only data will build into SRAM (If you need read only data build into SRAM to run faster, you should not use "const")
- 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 Ameba-D 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).

<pre> .xip_image2.text : {     __flash_text_start__ = .;      *(.img2_custom_signature*)      *(.text*)     *(.rodata*)      /* Add This for C++ support */     . = ALIGN(4);     __preinit_array_start = .;     KEEP(*(SORT(.preinit_array)))     __preinit_array_end = .;     . = ALIGN(4);     __init_array_start = .;     KEEP(*(SORT(.init_array.*)))     KEEP(*(SORT(.init_array)))     __init_array_end = .;     . = ALIGN(4);     __fini_array_start = .;     KEEP(*(SORT(.fini_array.*)))     KEEP(*(SORT(.fini_array)))     __fini_array_end = .;     /*-----*/      . = ALIGN(4);     __cmd_table_start__ = .;     KEEP(*(SORT(.cmd.table.data*)))     __cmd_table_end__ = .;      __flash_text_end__ = .;      . = ALIGN(16); } &gt; KM4_IMG2 </pre>	<pre> /* Add This for C++ support */ .ARM.extab : {     *(.ARM.extab* .gnu.linkonce.armextab.*) } &gt; KM4_IMG2  .ARM.exidx : {     __exidx_start = .;     *(.ARM.exidx* .gnu.linkonce.armexidx.*)     __exidx_end = .;     end = .; } &gt; 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 = puPortMalloc; __wrap_realloc = puPortReAlloc; __wrap_free = vPortFree; /*-----*/ </pre>
--	--

- (2) Modify startup code: `rtl8721dhp_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(SUGCall_IRQn, (VOID*)uPortSUCHandler);
    __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_OBJ_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

<pre> #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-xxa-atexit                 </pre>	<pre> LD_ARG += -nostartfiles LD_ARG += -specs nosys.specs LD_ARG += -Wl,-gc-sections LD_ARG += -Wl,-warn-section-align LD_ARG += -Wl,-Map=xxx.map LD_ARG += -Wl,-cref LD_ARG += -Wl,-build-id=none LD_ARG += -save-temps  LD_ARG += -Wl,-wrap,strcmp -Wl,-wrap,strchr -Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp -Wl,-wrap,strcpy #Wl,-wrap,strncmp LD_ARG += -Wl,-wrap,strcpy -Wl,-wrap,strlen #Wl,-wrap,strcpy LD_ARG += -Wl,-wrap,strlen -Wl,-wrap,strcmp -Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp LD_ARG += -Wl,-wrap,strcmp #Wl,-wrap,strcmp                 </pre>
--	--

```

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

- (4) Add C++ files into protect.
  - 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 Ameba-D.

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

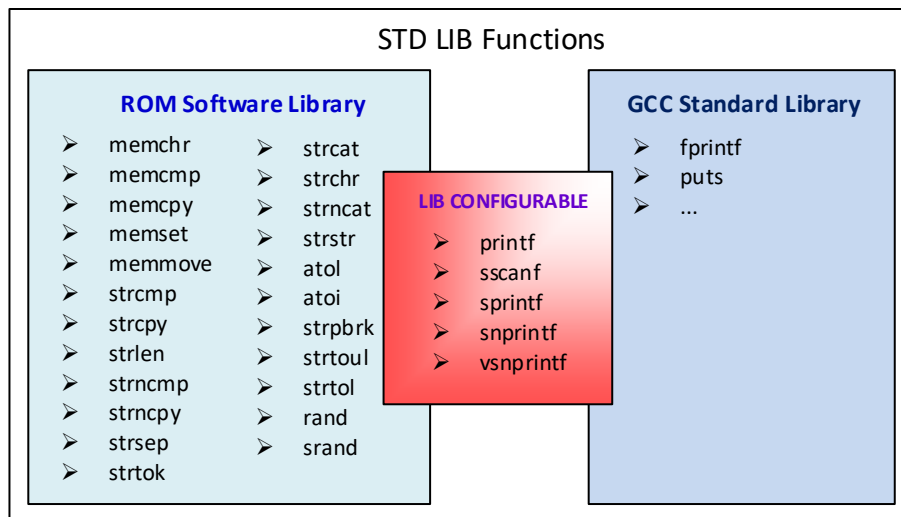


Fig 5-1 Library function guide

Table 5-1 Default use of library function

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

Because `printf/sprintf/snprintf/vsprintf/sscanf` in ROM software library are a 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
<code>printf</code>	<code>_rtl_printf</code>

sprintf	_rtl_sprintf
snprintf	_rtl_snprintf
vsprintf	_rtl_vsprintf
sscanf	_rtl_sscanf

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

### 5.3 To Use Configurable Function in GCC Standard Library

Compared to printf/sprintf/snprintf/vsprintf/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, %f, %L, %l, %u, %U, %e, %E, ...

For printf/sprintf/snprintf/vsprintf/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/sprintf/snprintf/vsprintf/sscanf in GCC standard library, follow these steps:

- (1) Add **#define STD\_PRINTF** in the front.

There are two cases:

Item	Operation	Description
case 1	Add <b>#define STD_PRINTF</b> in the front of c files, and include “ <b>platform_stdlib.h</b> ”	Change printf/sprintf/snprintf/vsprintf/sscanf of some files’ link to GCC standard library. Example: <pre>// main.c #define STD_PRINTF #include "main.h" #include "platform_stdlib.h"  int main(void) {     printf("%f\n", 1.2);     printf("%u\n", 20); }</pre>
case 2	Add <b>#define STD_PRINTF</b> in the front of “ <b>platform_stdlib_rtl8721d.h</b> ”	Change printf/sprintf/snprintf/vsprintf/sscanf of all SDK’s 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/vsprintf/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/vsprintf/sscanf maintain the default state, you need to modify “**platform\_stdlib\_rtl8721d.h**” file.

```

00026: #ifndef STD_PRINTF
00027: #undef printf
00028: #undef vsnprintf
00029: #undef sprintf
00030: #undef snprintf
00031: #undef sscanf
00032: #endif
00033: #undef memchr
00034: #undef memcmp
00035: #undef memcpy
00036: #undef memset
00037: #undef memmove
00038: #undef strcmp
00039: #undef strcpy
00040: #undef strlen
00041: #undef strncmp
00042: #undef strncpy
00043: #undef strsep
00044: #undef strtok
00045: #undef strcat
00046: #undef strchr
00047: #undef strncat
00048: #undef strstr
00049: #undef atol
00050: #undef atoi
00051: #undef strpbk
00052: #undef strtoul
00053: #undef strtol
00054: #undef rand
00055: #ifndef STD_PRINTF
00056: #define printf          _rtl_printf
00057: #define sprintf        _rtl_sprintf
00058: #define snprintf      _rtl_snprintf
00059: #define vsnprintf     _rtl_vsnprintf
00060: #define sscanf        _rtl_sscanf
00061: #endif

```

modify →

```

#ifndef STD_PRINTF
#define printf          _rtl_printf
#endif
#define sprintf        _rtl_sprintf
#define snprintf      _rtl_snprintf
#define vsnprintf     _rtl_vsnprintf
#define sscanf        _rtl_sscanf

```

↑ modify

```

#ifndef STD_PRINTF
#define printf          _rtl_printf
#endif
#define sprintf        _rtl_sprintf
#define snprintf      _rtl_snprintf
#define vsnprintf     _rtl_vsnprintf
#define sscanf        _rtl_sscanf

```

// NULL function  
// if use sscanf in std libc.a, please delete \_strtol

## 5.4 Tips

- If you want to use GCC standard library, we recommend only link some user specified c files to GCC standard library instead of link all SDK to GCC standard library, because printf/sprintf/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, KMO does not support floating-point temporarily, so printf cannot dump %f in KMO 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-D SDK, including building projects, downloading images and debugging.

### 6.1 Requirement

#### 6.1.1 IAR Embedded Workbench

IAR provides an IDE environment for code building, downloading, and debugging. Check “IAR Embedded Workbench” on <http://www.iar.com/>, and a trial version is available for 30 days.

**Note:** To support ARMv8-M with Security Extension (Ameba-D 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 Ameba-D with IAR, then a J-Link adapter or a RLX Probe is necessary.

**Note:** For Ameba-D CPU, the J-Link version must be v9 or higher.

### 6.2 Hardware Configuration

The hardware block diagram of Ameba-D 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 Ameba-D to run firmware after IAR completes downloading.

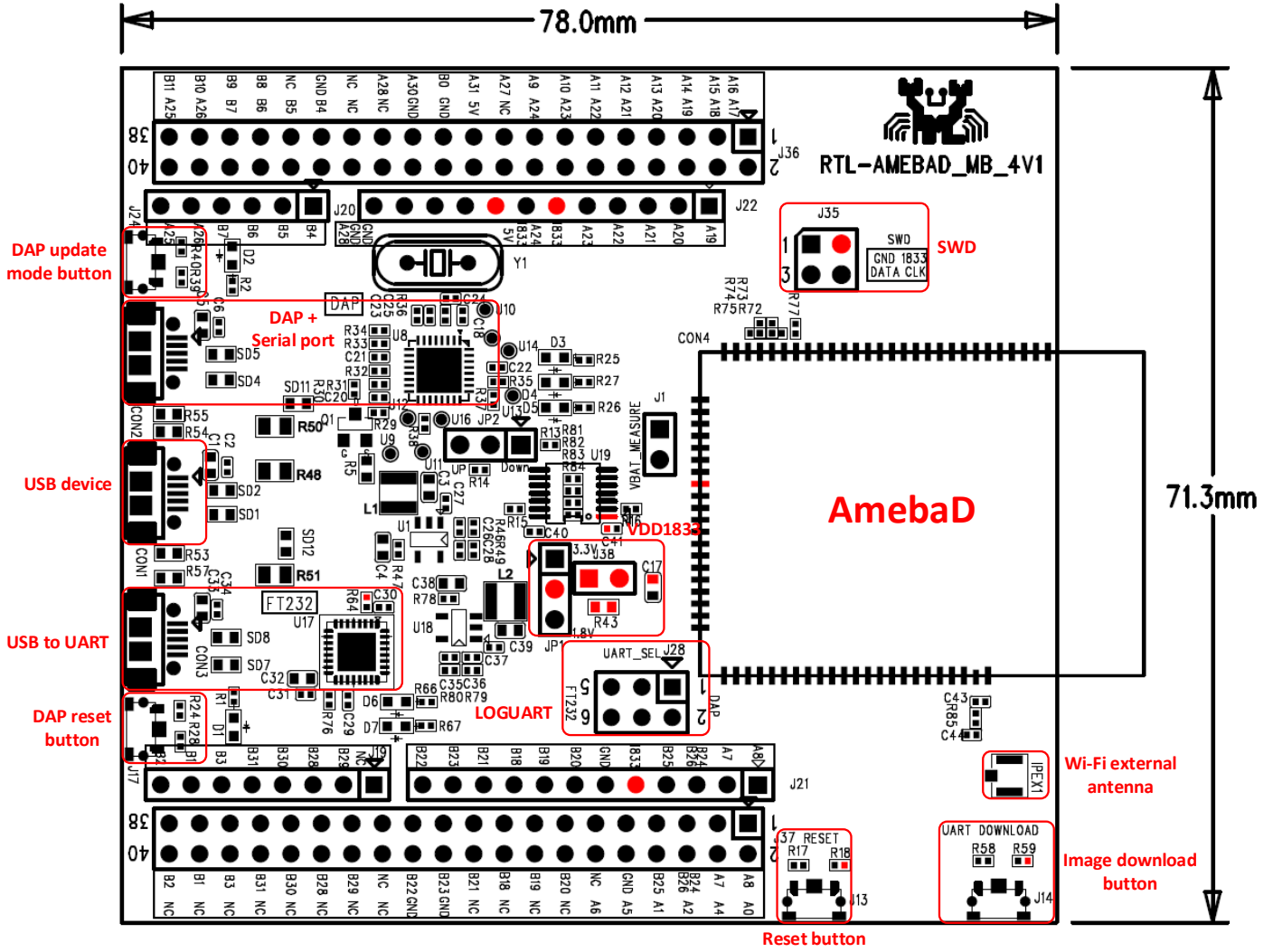


Fig 6-1 Ameba-D demo board

### 6.2.1 Connecting with J-Link

Refer to Fig 6-2 and Fig 6-3 to connect Ameba-D SWD interface with J-Link.

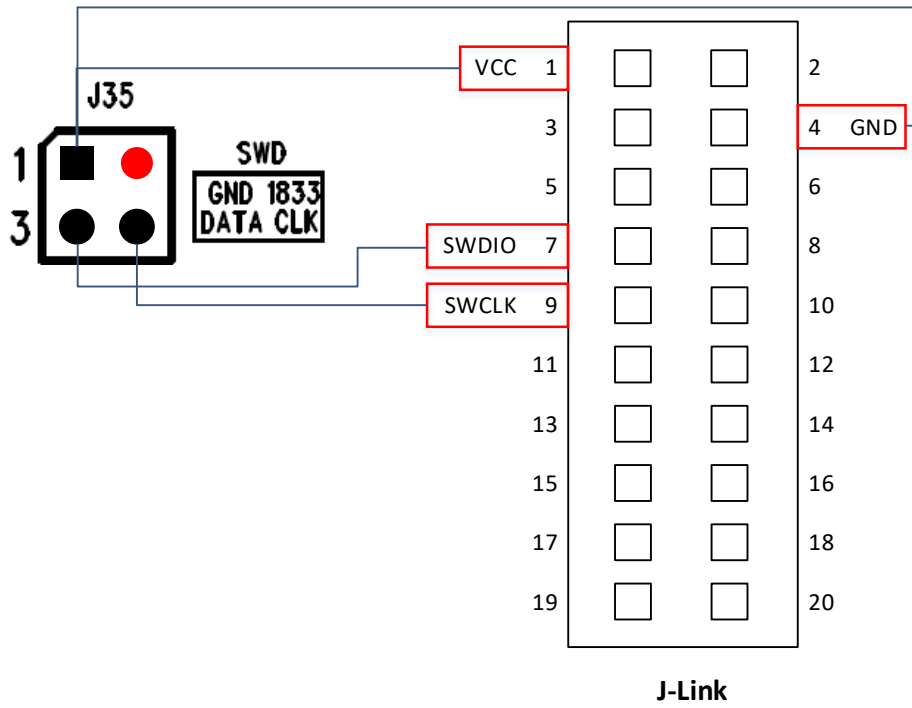


Fig 6-2 J-Link and SWD connection diagram

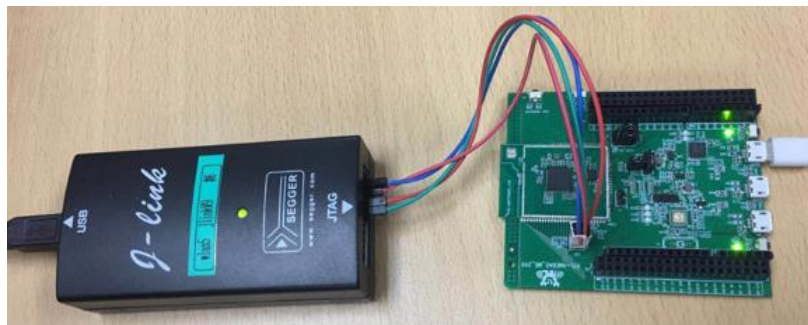


Fig 6-3 J-Link and Ameba-D SWD connection

## 6.2.2 Connecting with RLX Probe

Refer to Fig 6-4 and Fig 6-5 to connect Ameba-D SWD interface with RLX Probe.

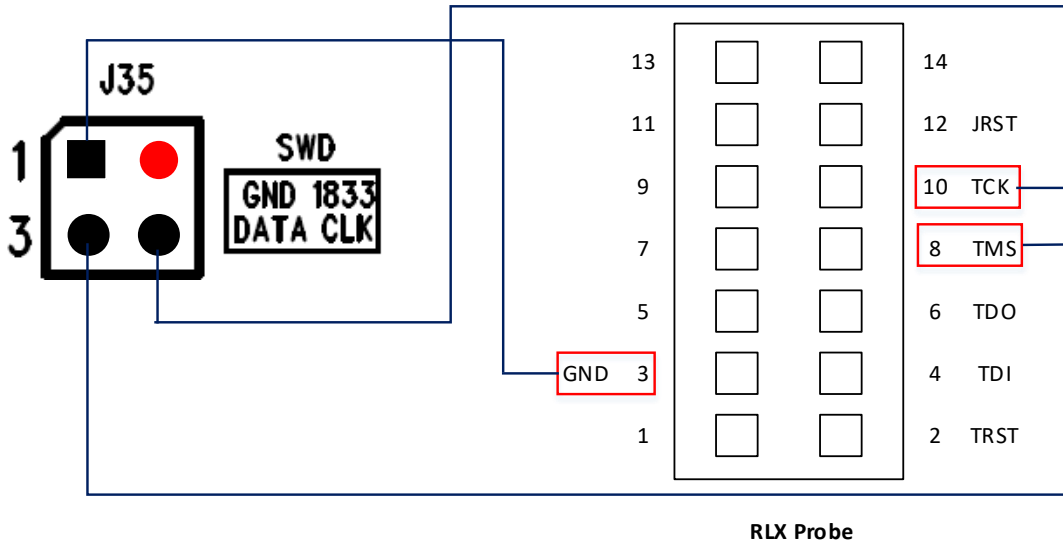


Fig 6-4 RLX Probe and SWD connection diagram

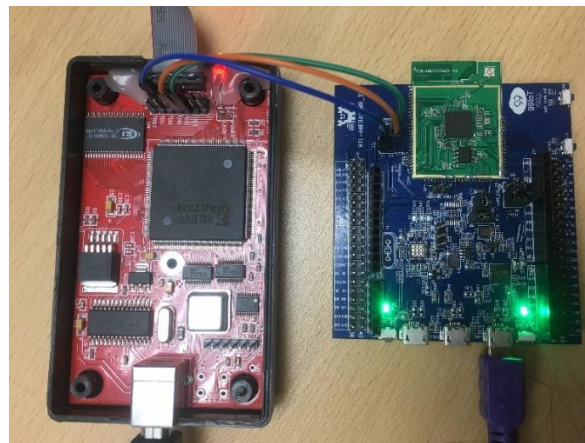


Fig 6-5 RLX Probe and Ameba-D SWD connection

## 6.3 How to Use IAR SDK?

### 6.3.1 IAR Project Introduction

Because Ameba-D 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 <sup>1</sup>	N	N
	km4_bootloader - tz <sup>2</sup>	Y	N
km4_application	km4_application - is	N	N
	km4_application - tz	Y	N
	km4_application - is (mp <sup>3</sup> )	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 items with “-is” are ignore secure configuration, which are designed for applications that do not use TrustZone.
  2. The configuration items with “-tz” are TrustZone configuration, which are designed for applications that use TrustZone.
  3. The configuration items with “mp” are mass production configuration, which are designed for generating MP image.
- 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.

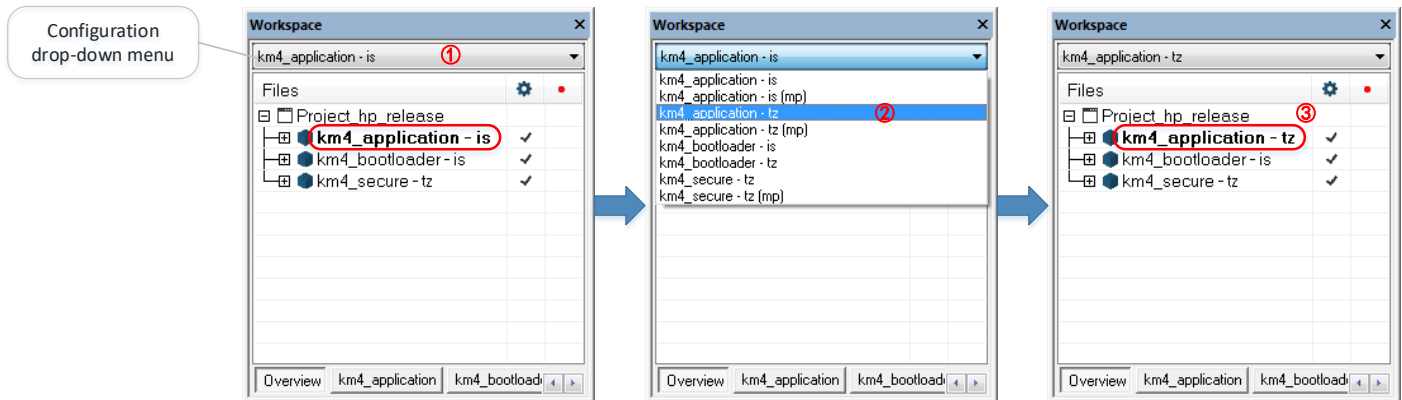


Fig 6-6 How to choose a build configuration

### 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.
- (3) Right click the project and choose "Rebuild All", as Fig 6-8 shows. The km0\_bootloader and km0\_application should compile in order.

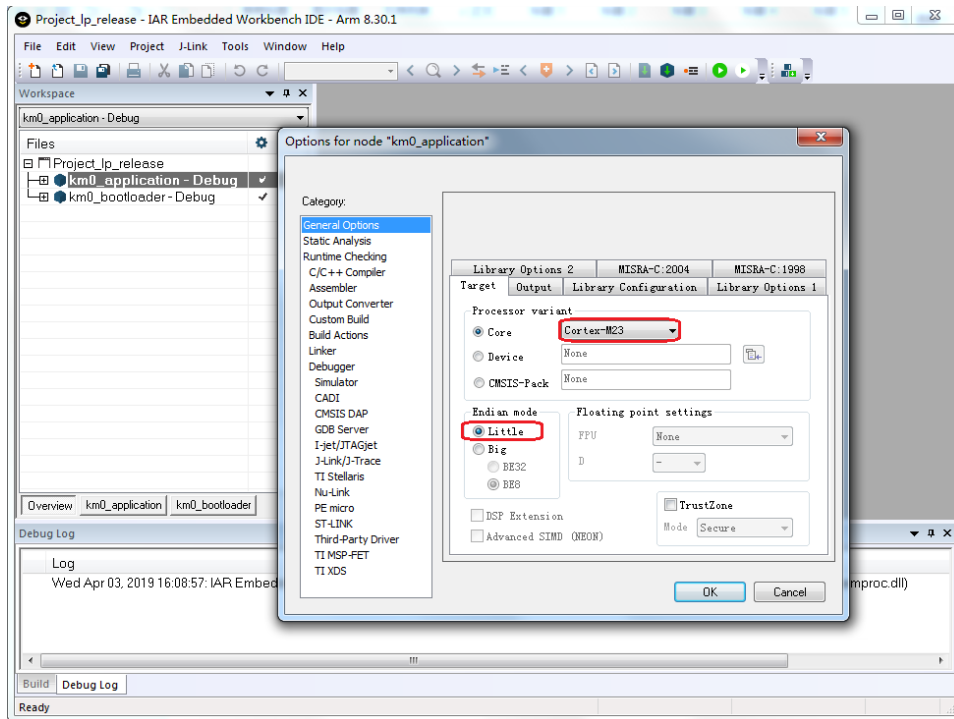


Fig 6-7 KM0 processor options

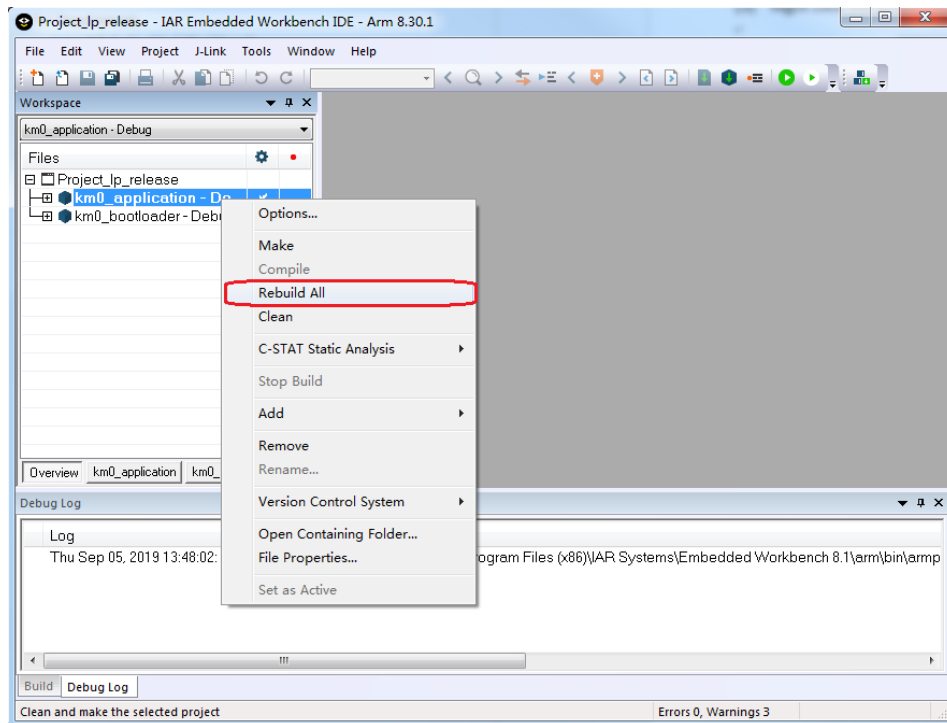
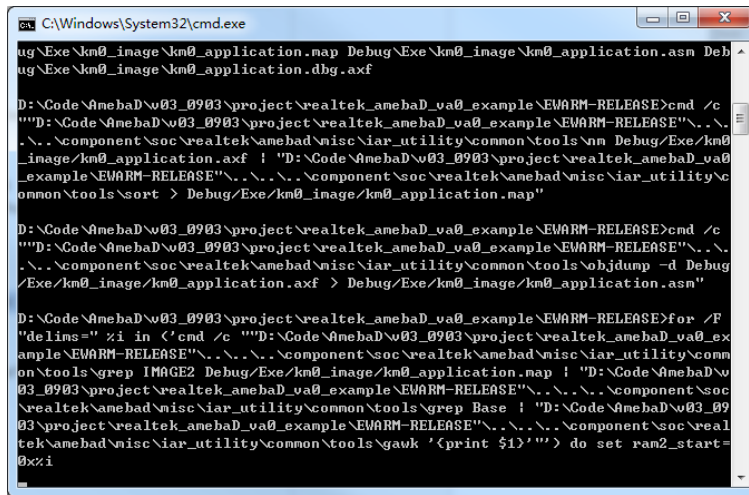


Fig 6-8 Building KM0 project

**Note:** After building each project, IAR will pop up a command prompt window to execute post-build action to generate images from executable files. This may takes several seconds. Don't stop it while it is in progress. After post-build action is completed, the window would 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\_hp\_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.

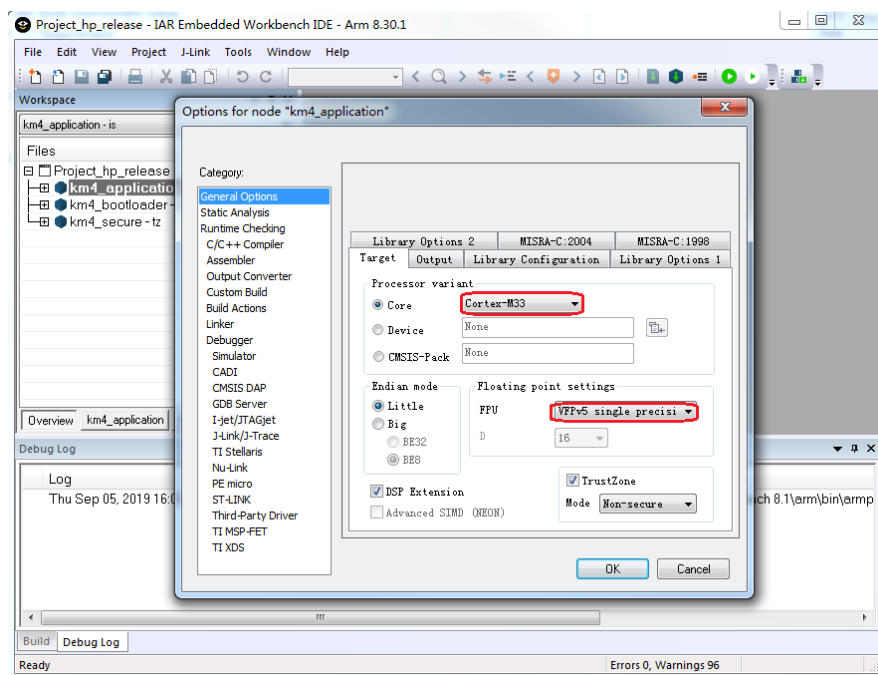


Fig 6-9 KM4 processor options

- (4) Right click the project and choose “Rebuild All”, as Fig 6-10 shows. The km4\_bootloader, km4\_secure and km4\_application should compile in order.

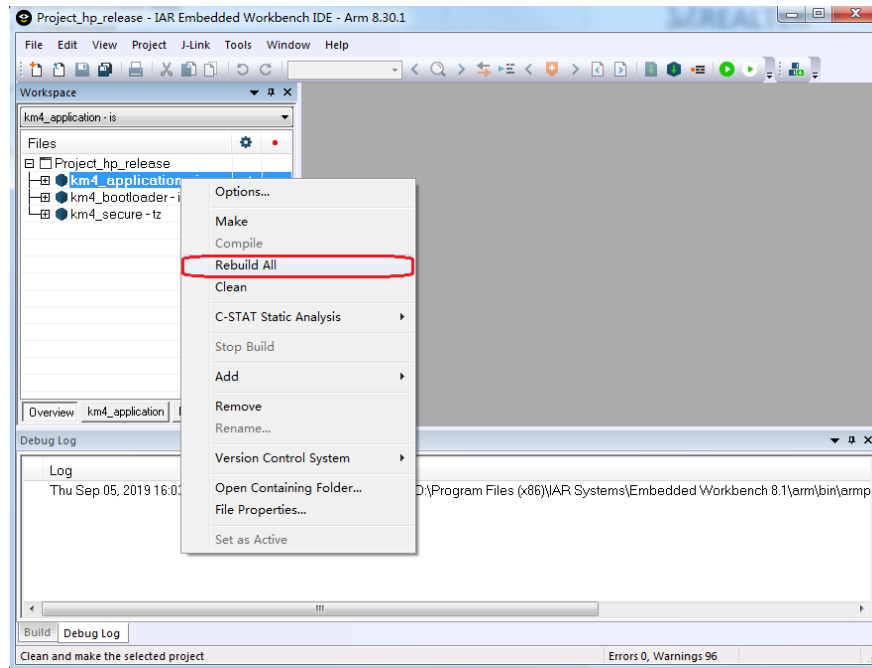
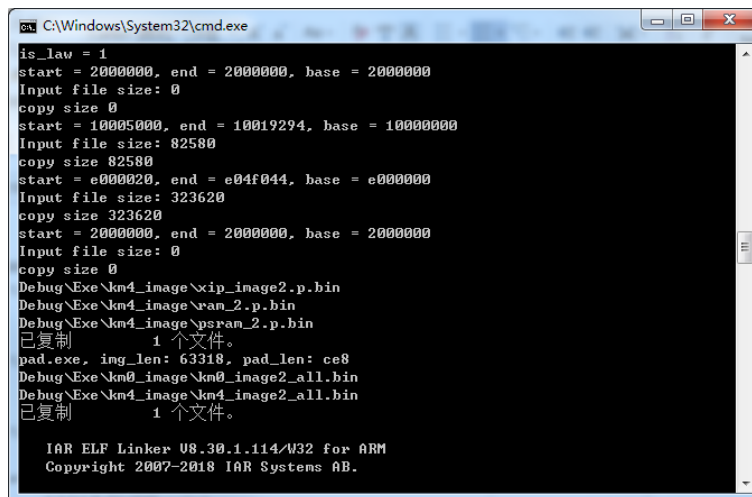


Fig 6-10 Building KM4 project

**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. This may takes several seconds. Don't stop it while it is in 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)
- Ameba-D ImageTool, refer to ImageTool for more information.

Ameba-D 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 can't connect to KM4 and show the error message as Fig 6-11 shows. For RLX probe, RLX Probe driver can't be opened under KM4.

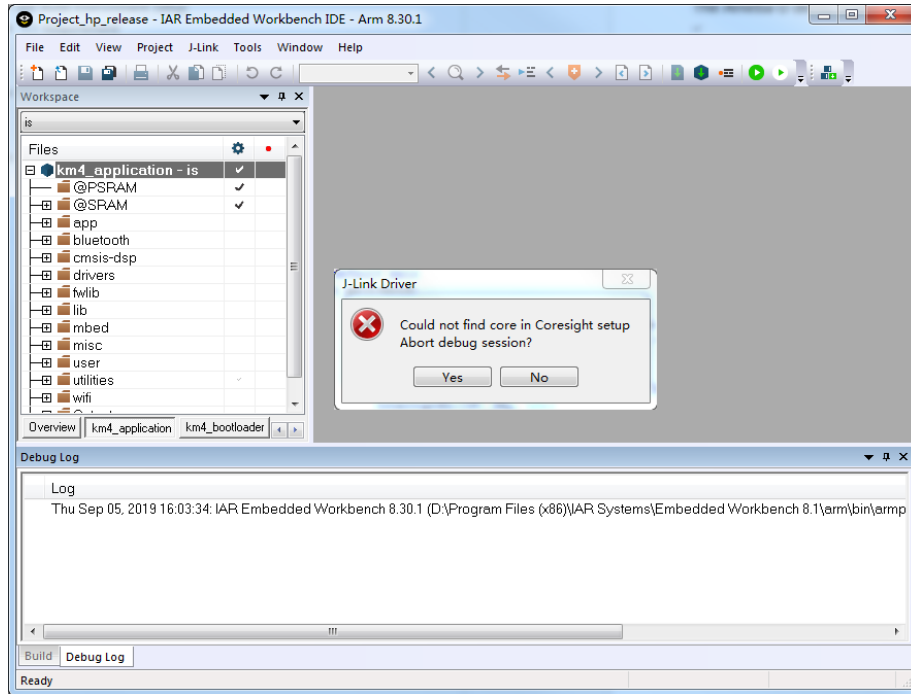


Fig 6-11 J-Link cannot find KM4

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 Fig 6-24 displayed in IAR window, refer to 6.3.4.3.

- (1) Choose the target project display in Workspace window, for example, km4\_bootloader as Fig 6-12 shows.
- (2) If using J-link debugger, check whether the J-link debugger setting is correct.
  - a) Click **Project > Options > Debugger > Setup > Driver**, and choose "J-Link/J-Trace", as Fig 6-13 shows.
  - b) Click **Debugger > J-Link/J-Trace > Connection > Interface**, and choose "SWD", as Fig 6-14 shows.

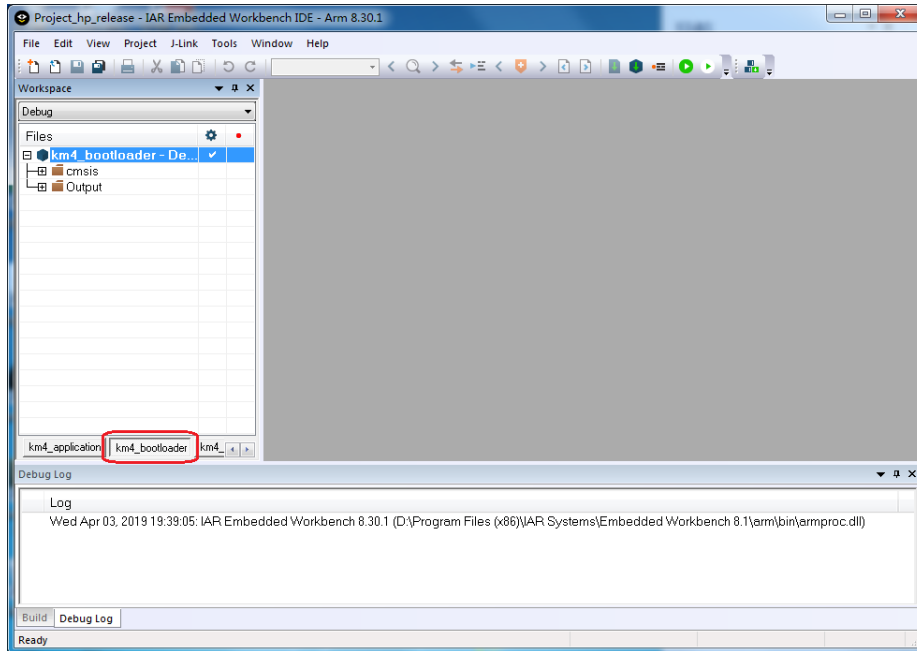


Fig 6-12 Switching to the target project view

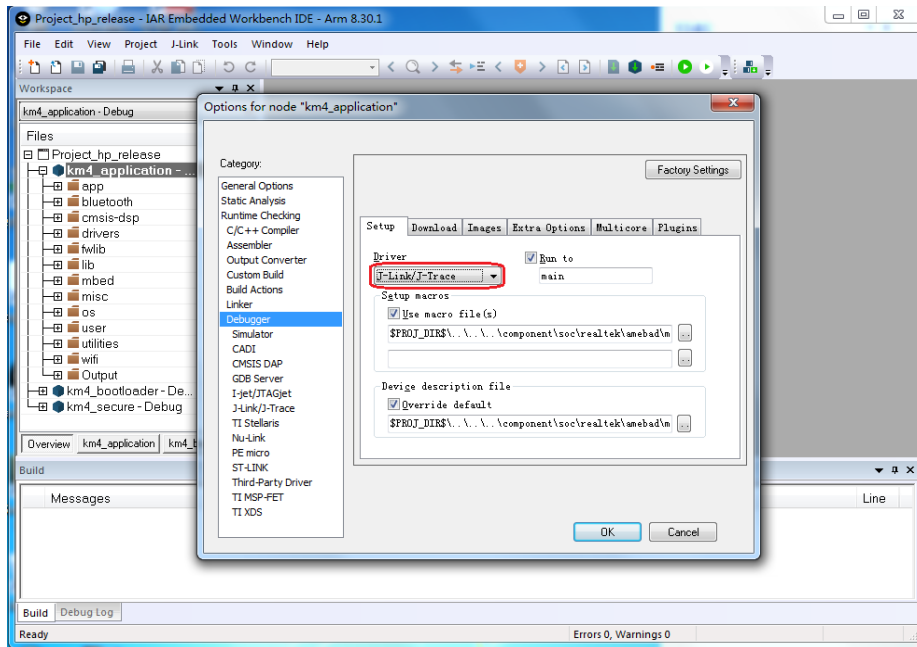


Fig 6-13 J-Link debugger setup

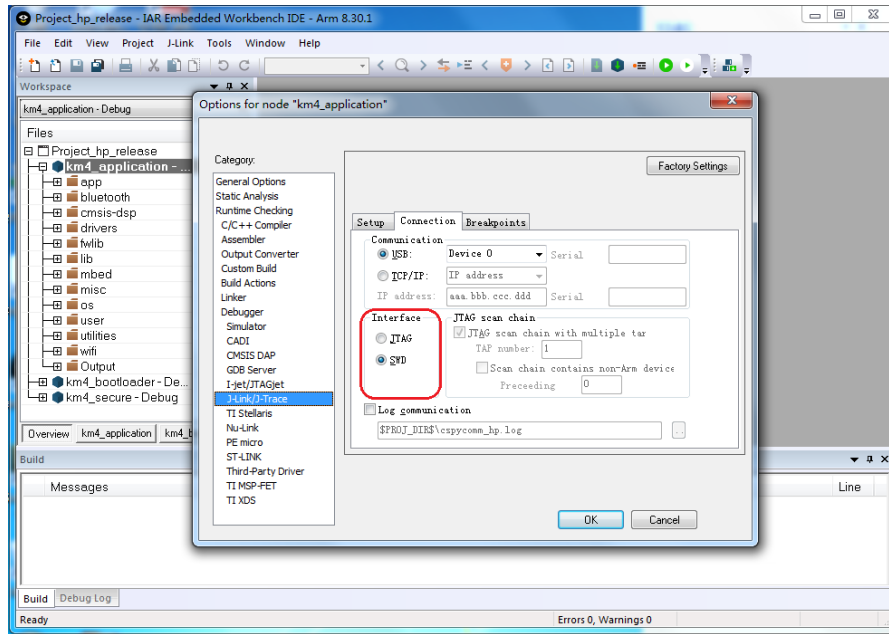


Fig 6-14 J-Link interface setup

- (3) If using RLX Probe, check whether the RLX Probe setting is correct.
  - a) Click **Project > Options > Debugger > Setup > Driver**, choose "GDB Server" and don not choose "Run to", as Fig 6-15 shows.
  - b) Click **GDB Server > GDB Server**, and put correct value in "TCP/IP address or hostname", as Fig 6-16 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**.
    - ◆ KMO: 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 Fig 6-17 shows.
 

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

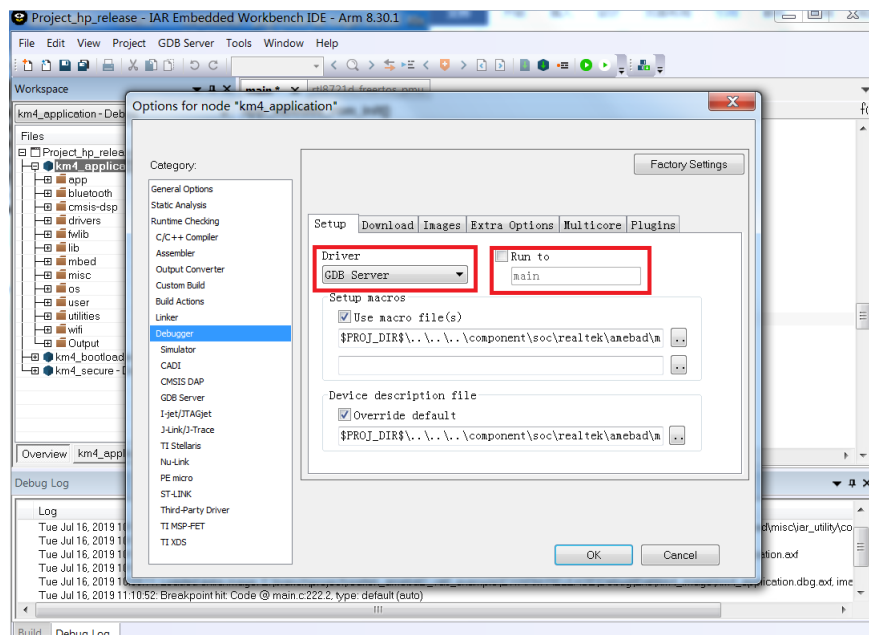


Fig 6-15 RLX Probe setup

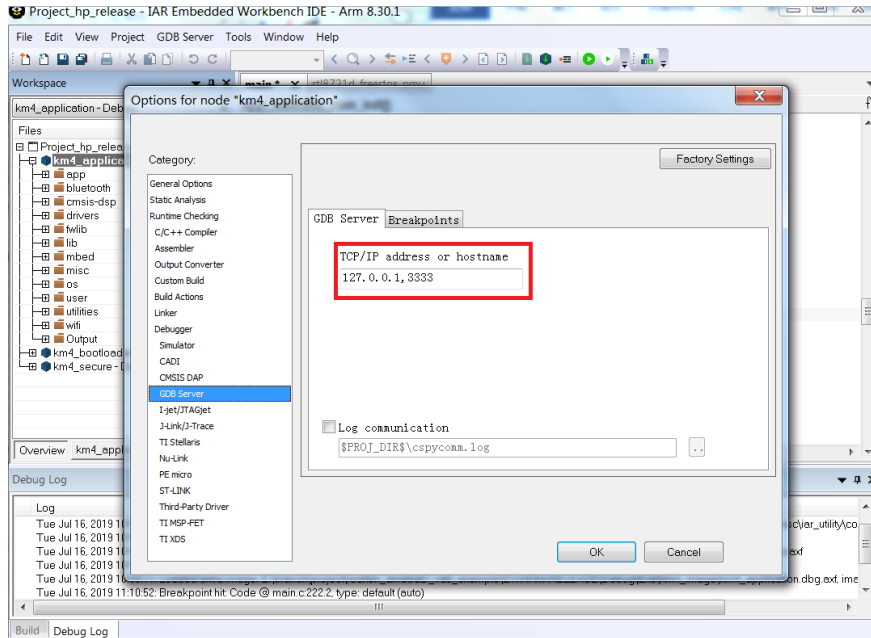


Fig 6-16 RLX Probe interface setup

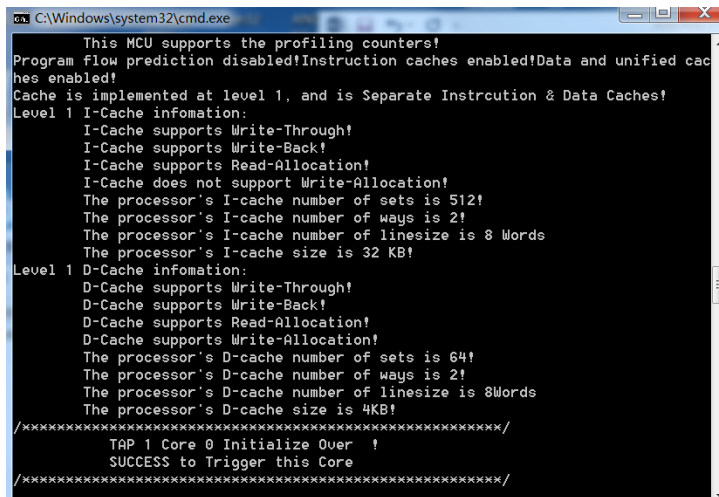


Fig 6-17 RLX Probe window

- (4) Click **Project > Download > Download active application**, image downloading starts. When downloading, Ameba-D prints the log, as Fig 6-18 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
    
```

Fig 6-18 Downloading log

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

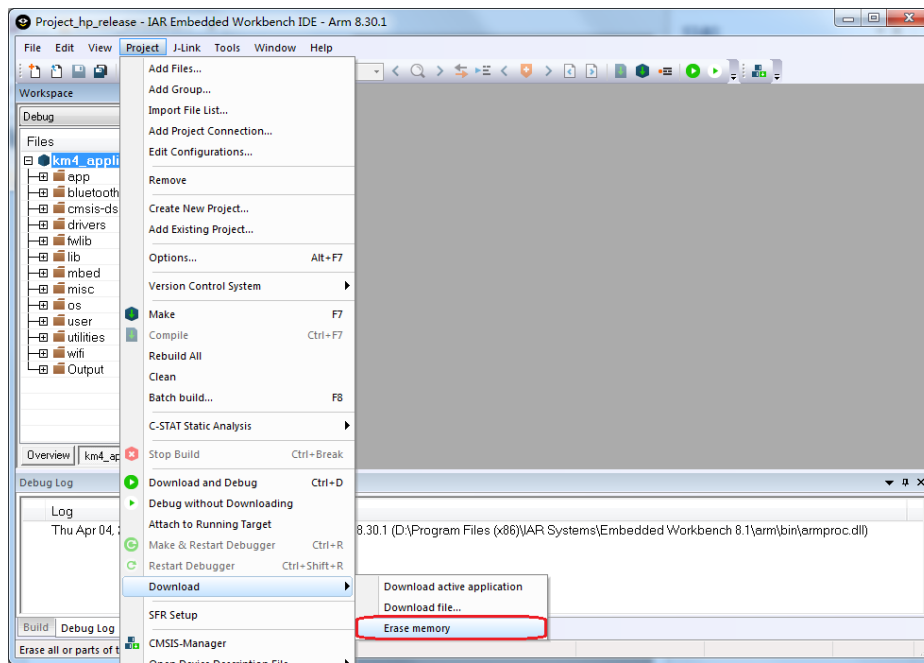


Fig 6-19 Erasing Flash memory

### 6.3.4 IAR Debug

You can debug or trace KM0 and KM4 system individually with J-Link 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 Fig 6-20 shows.
- (4) Toggles a breakpoint at the statement or instruction that contains or is located near the cursor in the source window. The “Toggle Breakpoint” button is on the debug toolbar, as Fig 6-21 shows.

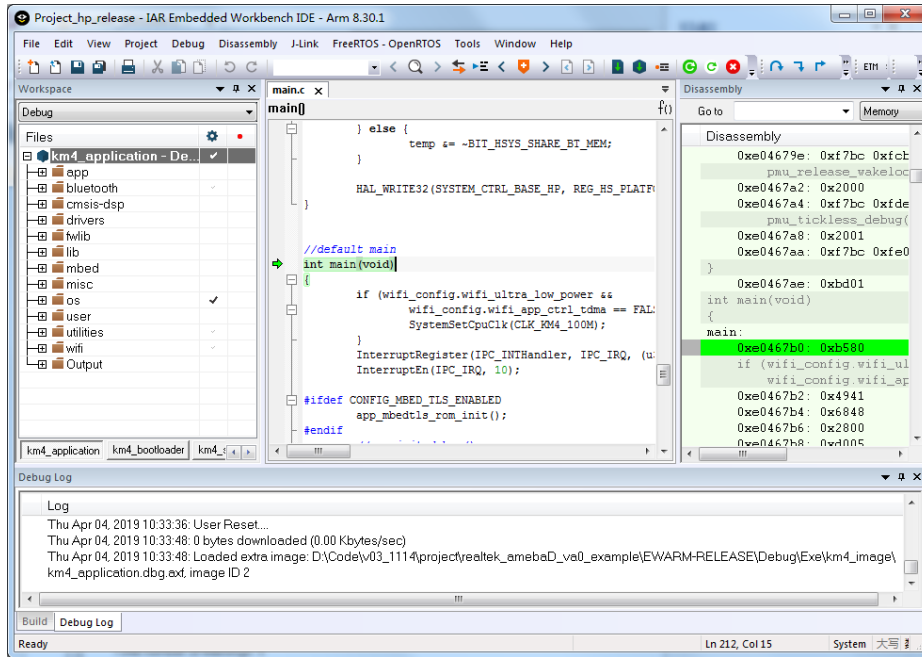


Fig 6-20 Running to main() when debug

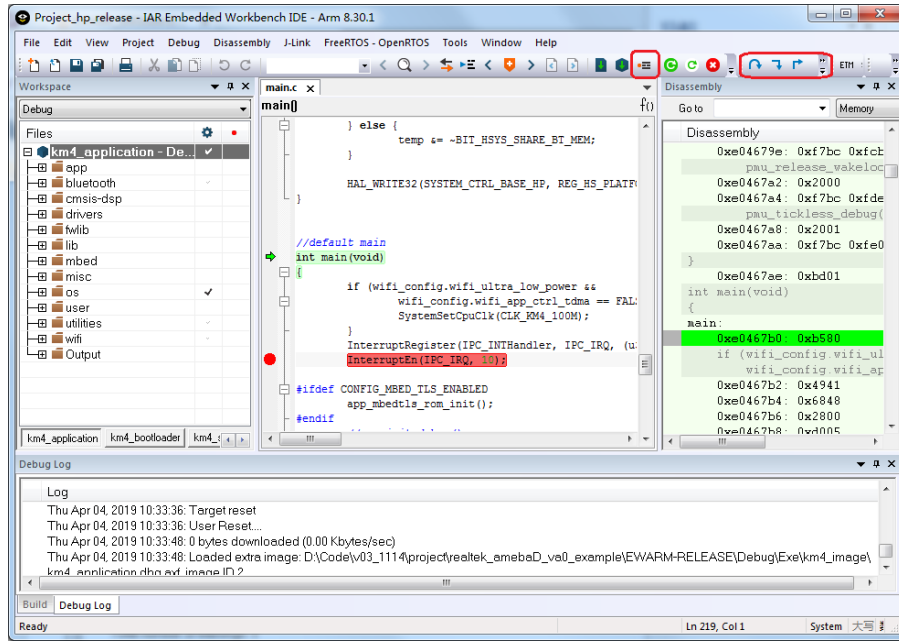


Fig 6-21 Toggle breakpoint

(5) You can trace code step by step with “Step Into” or “Go” until triggering a breakpoint. These function buttons are available on toolbar.

### 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 “`ew .pc = 0xxxxxxx`” and change the address to the appointed value, as Fig 6-22 and Fig 6-23 shows.

```

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

Fig 6-22 Setting the target address

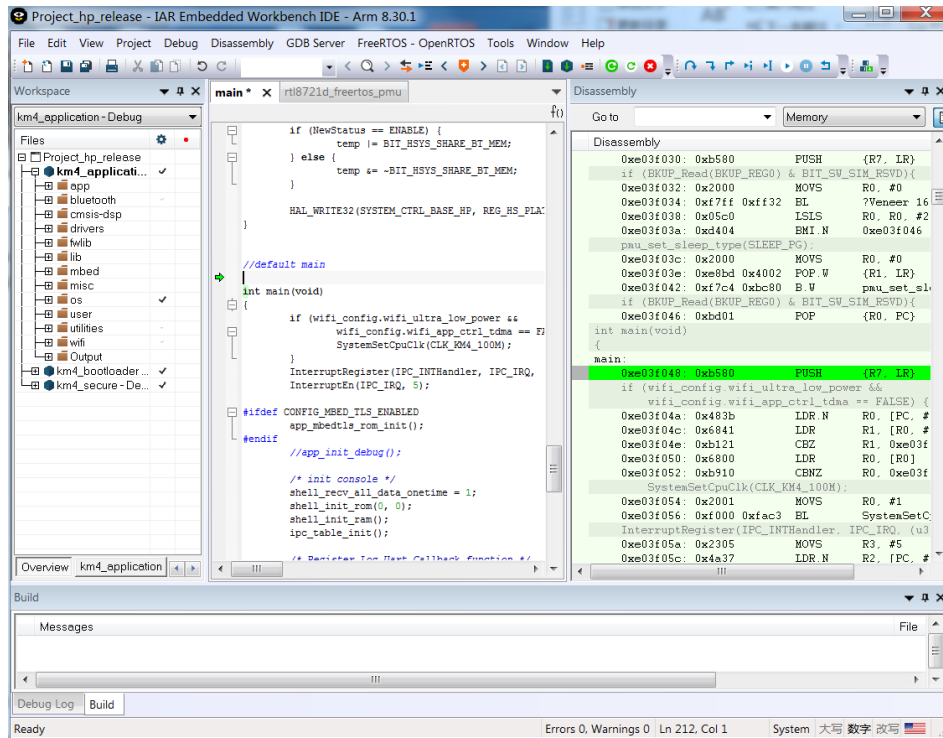


Fig 6-23 Running to the target address when debug

- (4) Toggles a breakpoint at the statement or instruction that contains or is located near the cursor in the source window. The “Toggle Breakpoint” button is on the debug toolbar, as Fig 6-21 shows.
- (5) You can trace code step by step with “Step Into”, or “Go” until triggering a breakpoint. These function buttons are available on toolbar.

### 6.3.4.3 IAR Debug or Download Error

Because Ameba-D 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 Fig 6-24 happens.

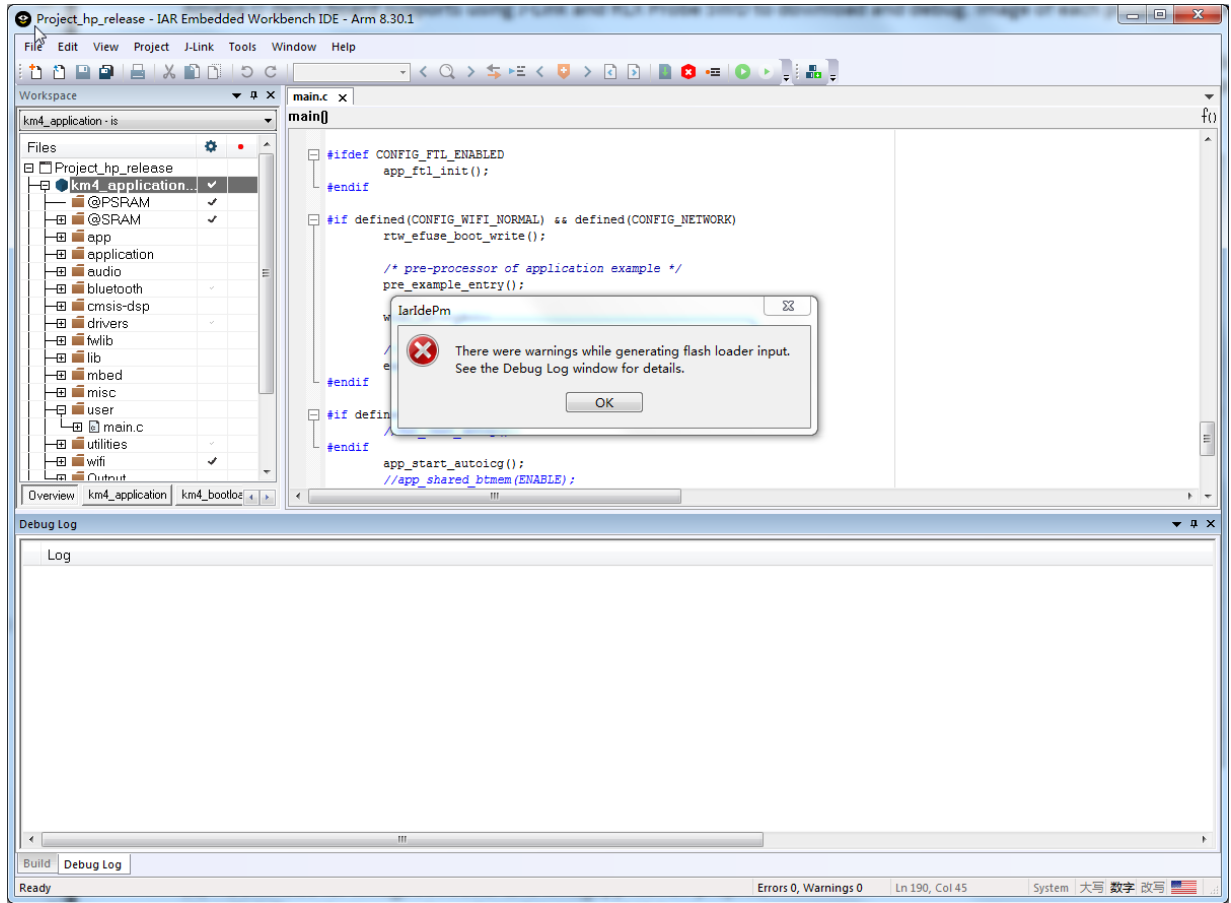


Fig 6-24 IAR debug or download error

To avoid this error, you should build manually before debug or download, and disable auto-build from **Tools > Options > Project > Make before debugging**, as Fig 6-25 shows.

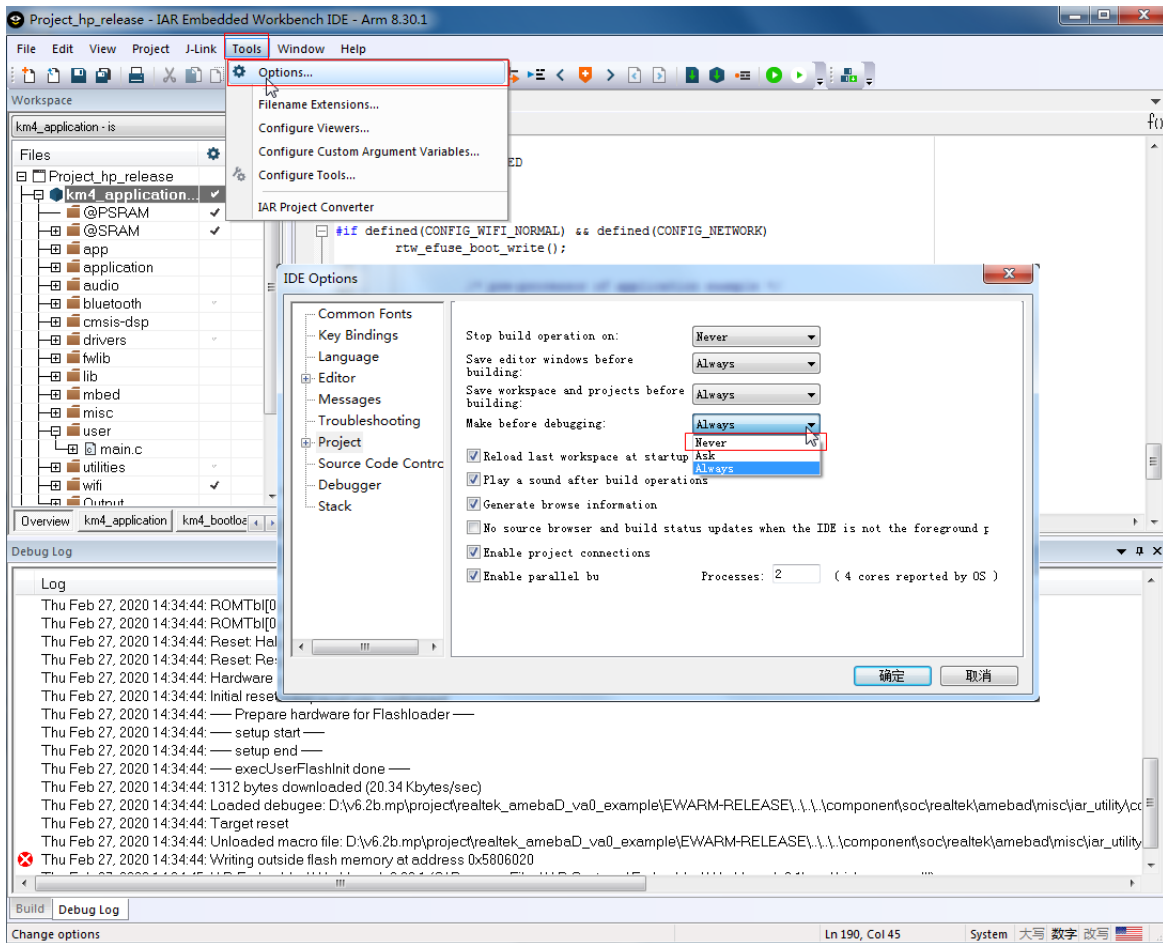


Fig 6-25 IDE options

## 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. In IAR workspace, there are “@PSRAM” and “@SRAM” group. The code in “@PSRAM” group would be linked and loaded into PSRAM, and the code in “@SRAM” group would be linked and loaded into SRAM. The rest of code will be placed on Flash and execute in place.

**Note:** Considering only SRAM and PSRAM contains secure regions, the code in km4\_secure project should be placed either in “@PSRAM” or in “@SRAM”. It can’t be placed outside these two groups.

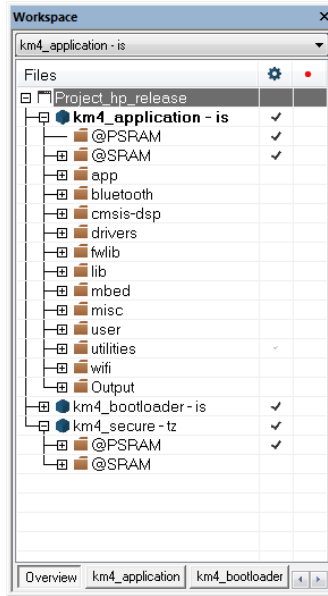


Fig 6-26 Memory location configuration

### 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 Ameba-D location:

- "project\realtek\_amebaD\_va0\_example\EWARM-RELEASE\rtl8721dhp\_image2\_is.icf" for ignore secure project
- "project\realtek\_amebaD\_va0\_example\EWARM-RELEASE\rtl8721dhp\_image2\_tz.icf" for TrustZone project

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 "main.c" file in the target example to `project\realtek_amebaD_va0_example\src\src_hp` and replace the original one.

For example, you can copy "main.c" from `project\realtek_amebaD_va0_example\example_sources\I2C\mbed\i2c_int_mode\src` to use i2c\_int\_mode example code, as Fig 6-27 shows.

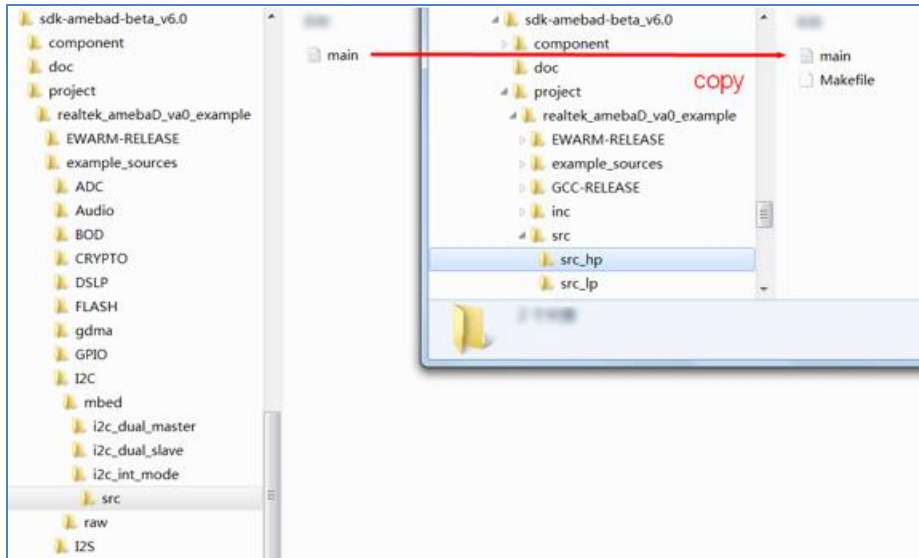
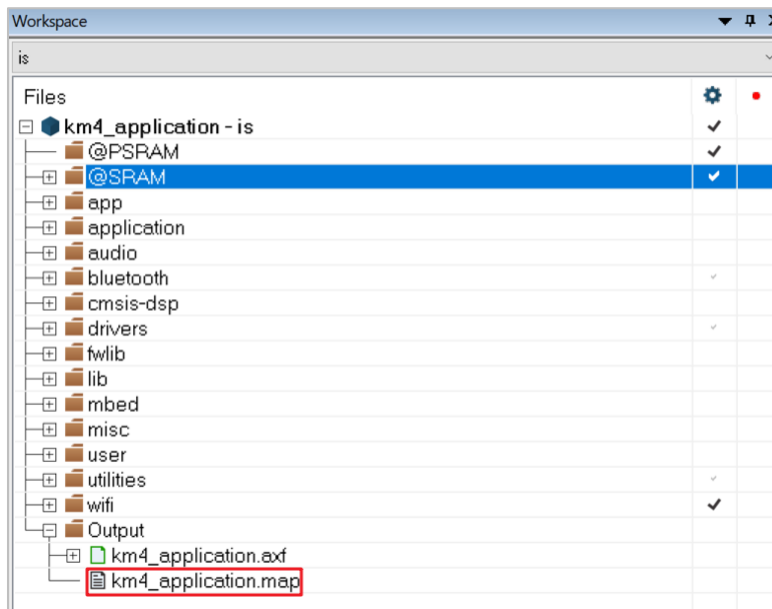


Fig 6-27 Building sample code

## 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” to observe them after project build. This file can be found in the following folders:

- Project folder: project\realtek\_amebaD\_va0\_example\EWARM-RELEASE\Debug\List\km4\_application
- IAR GUI’s folder: Output



### 6.5.1 Memory Section

- A7 (PSRAM): This section is read-write data in PSRAM (0x2000000 to 0x23FFFFFF).
- BTTRACE: This section is reserved for BTTRACE.
- 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	rt18721dhp 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 = 470K bytes memory in SRAM is used.

The total SRAM space is defined in project\realtek\_amebaD\_va0\_example\EWARM-RELEASE\rt18721d\_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.

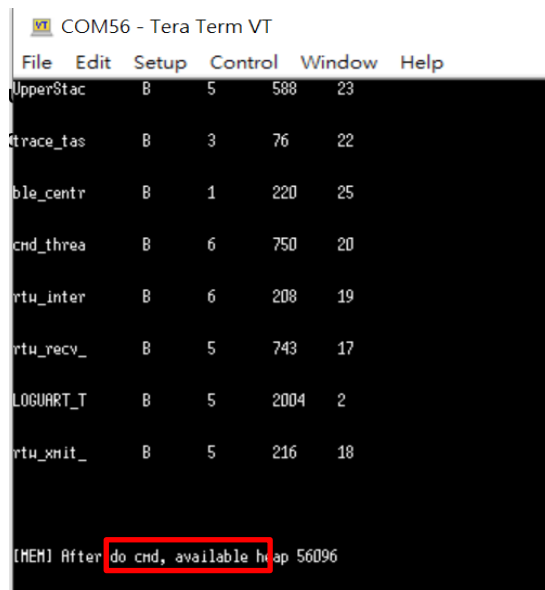
A5 has size 0x8afe8.

```
"A5":
  .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 Fig 7-1 and Fig 7-2.

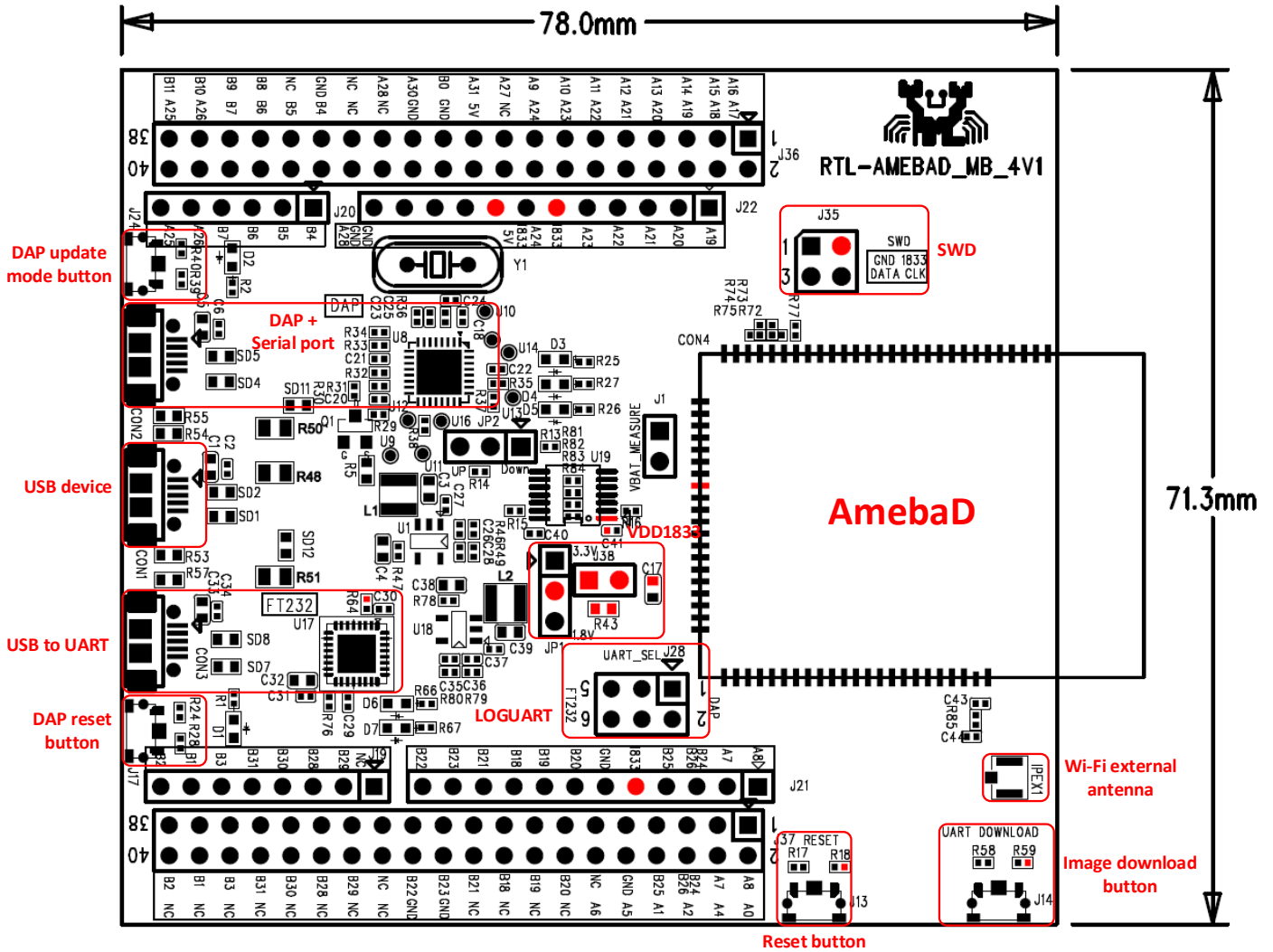


Fig 7-1 Demo board – PCB layout (2D)

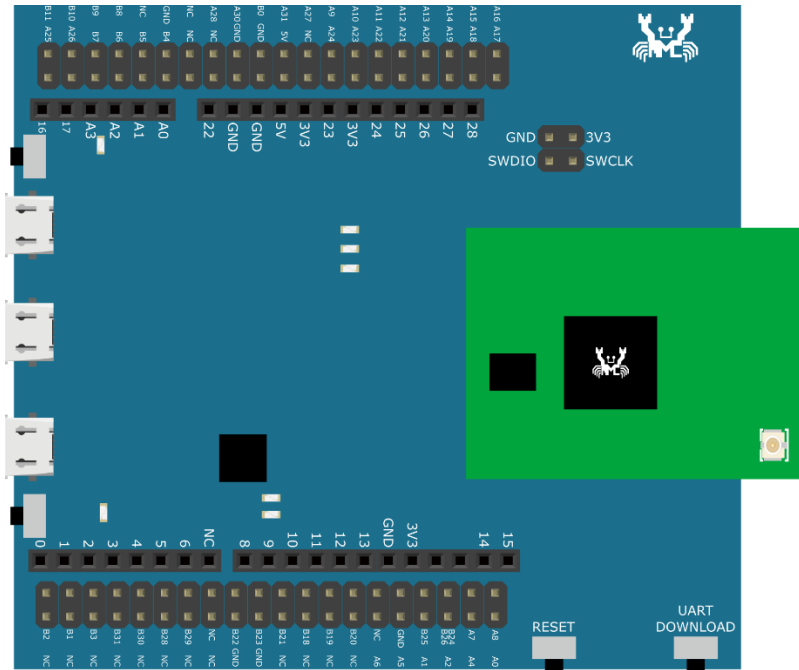


Fig 7-2 Demo board – PCB layout (3D)

## 7.2 Pin Out

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

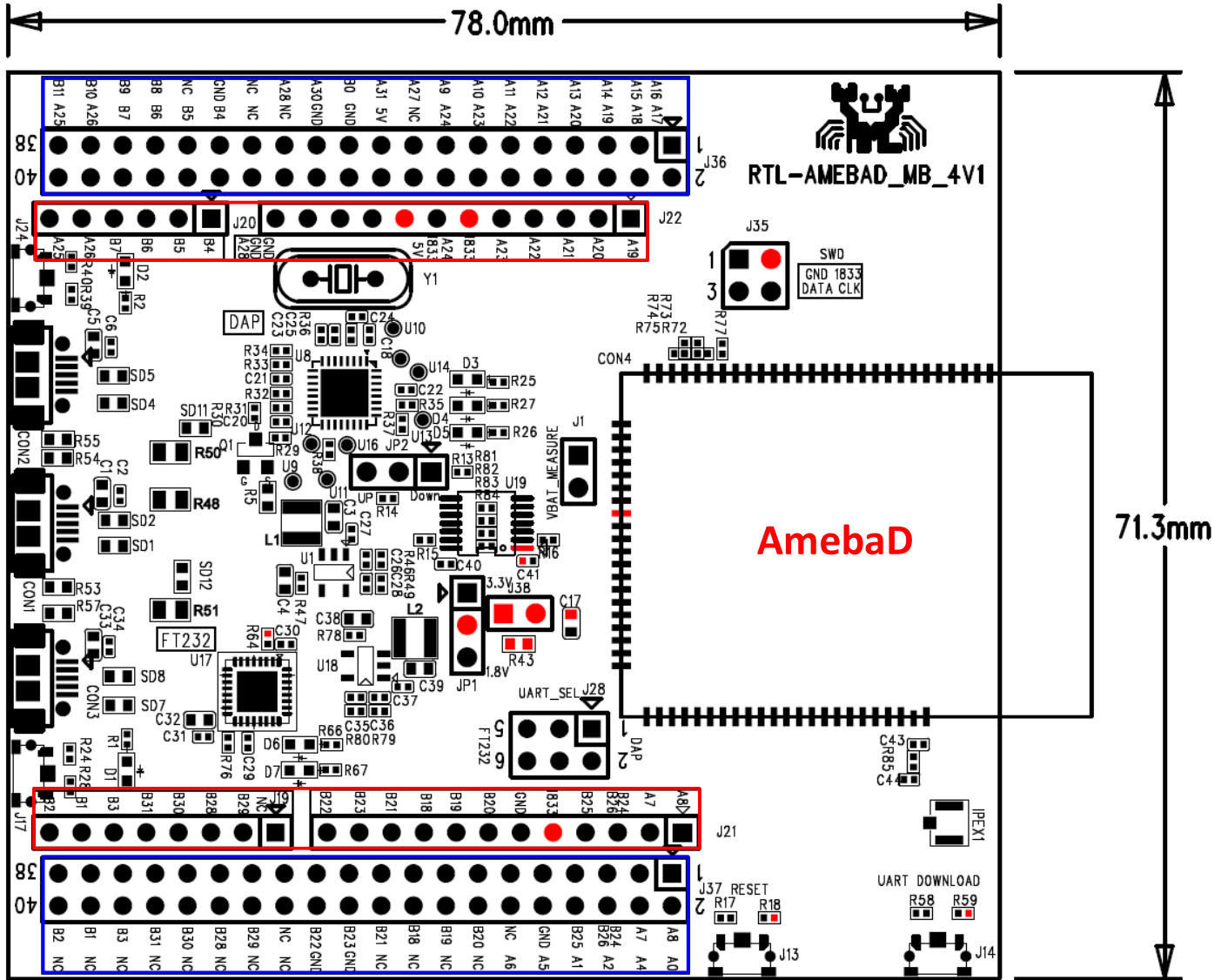


Fig 7-3 Demo board – pin out

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 Fig 7-4.

- 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.

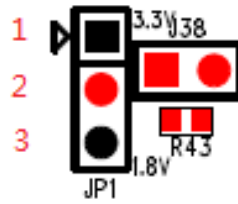


Fig 7-4 Demo board – 3.3V/1.8V power supply

When you select power supply, refer to Table 7-1.

Table 7-1 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 Fig 7-5 and Fig 7-6.



Fig 7-5 Mother board – USB interface configuration

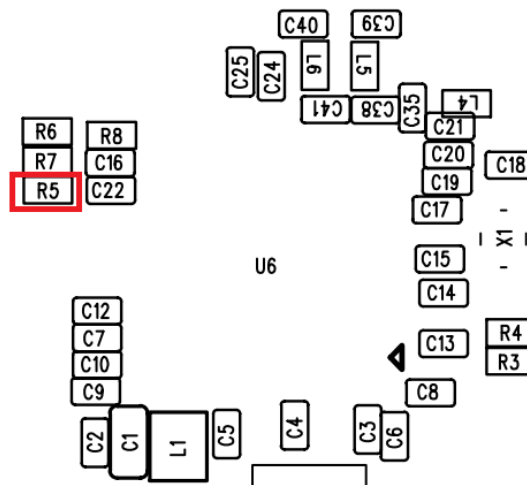


Fig 7-6 Module board – USB interface configuration

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 Fig 7-7. When you select LOGUART, please refer to Table 7-2.

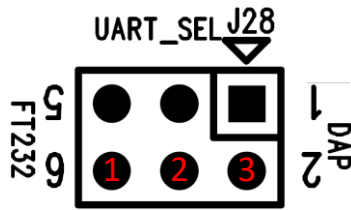


Fig 7-7 Demo board – LOGUART

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.

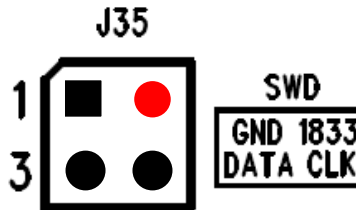


Fig 7-8 Demo board – SWD

**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 Fig 8-1, ImageTool has four tabpages.

- Download: used as image download server to transmit images to Ameba through UART.
- Generate: contact 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.

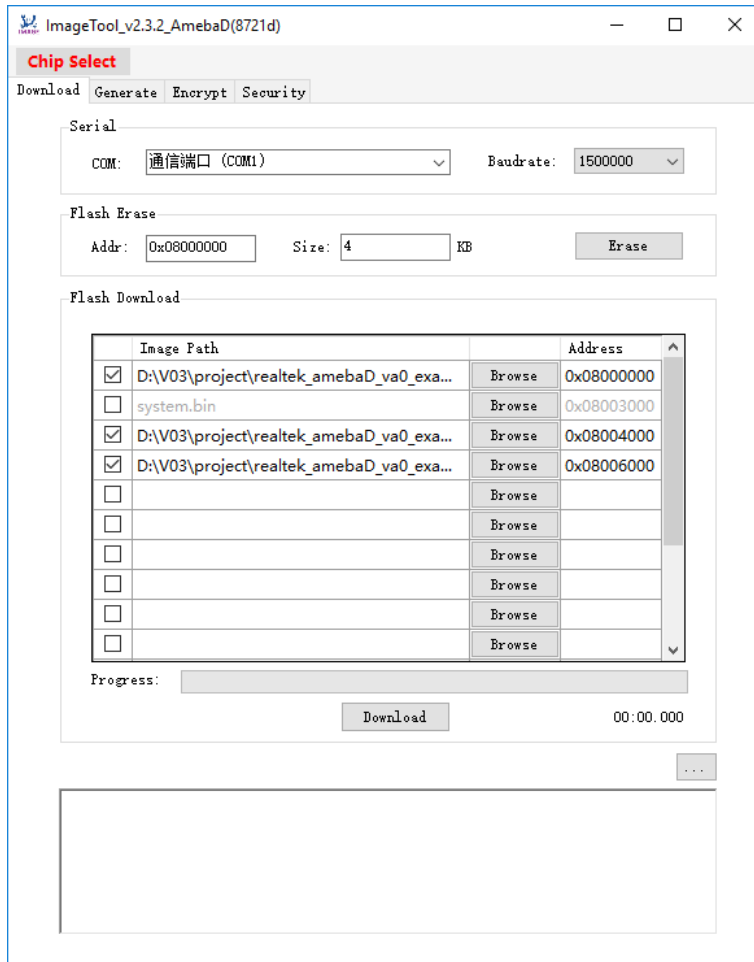


Fig 8-1 ImageTool UI

## 8.2 Environment Setup

### 8.2.1 Hardware Setup

The hardware setup is shown in Fig 8-2.

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



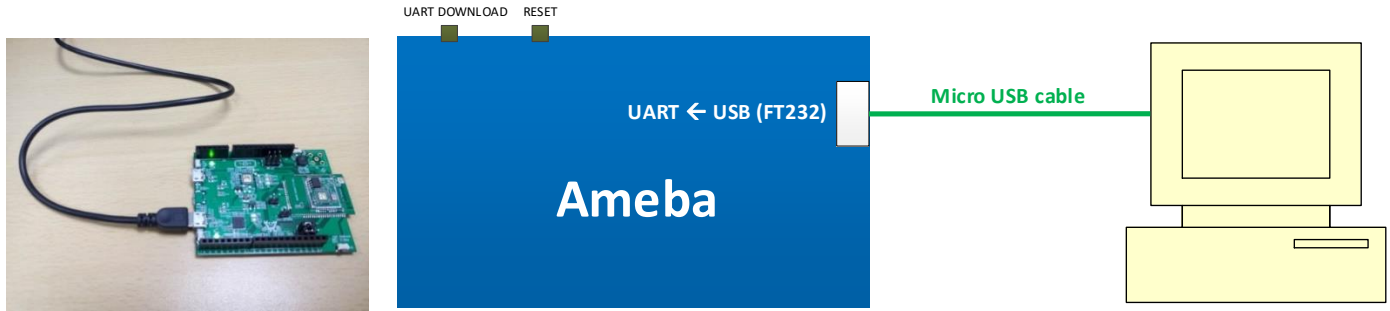


Fig 8-2 Hardware setup

## 8.2.2 Software Setup

- Environment Requirements: EX. WinXP, Win 7 Above, 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
<b>ImageTool.exe</b>	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 Ameba (client) 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.

- Enter into UART\_DOWNLOAD mode.
  - Push the **UART DOWNLOAD** button and keep it pressed.
  - Re-power on the board or press the **Reset** button.
  - Release the **UART DOWNLOAD** button.

Now, Ameba board gets into UART\_DOWNLOAD mode and is ready to receive data.
- Click **Chip Select** (in red) on UI and select chip (AmebaD or AmebaZ).
- Select the corresponding serial port and transmission baud rate. The default baud rate is 1.5Mbps (recommended).
- 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 in `{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.

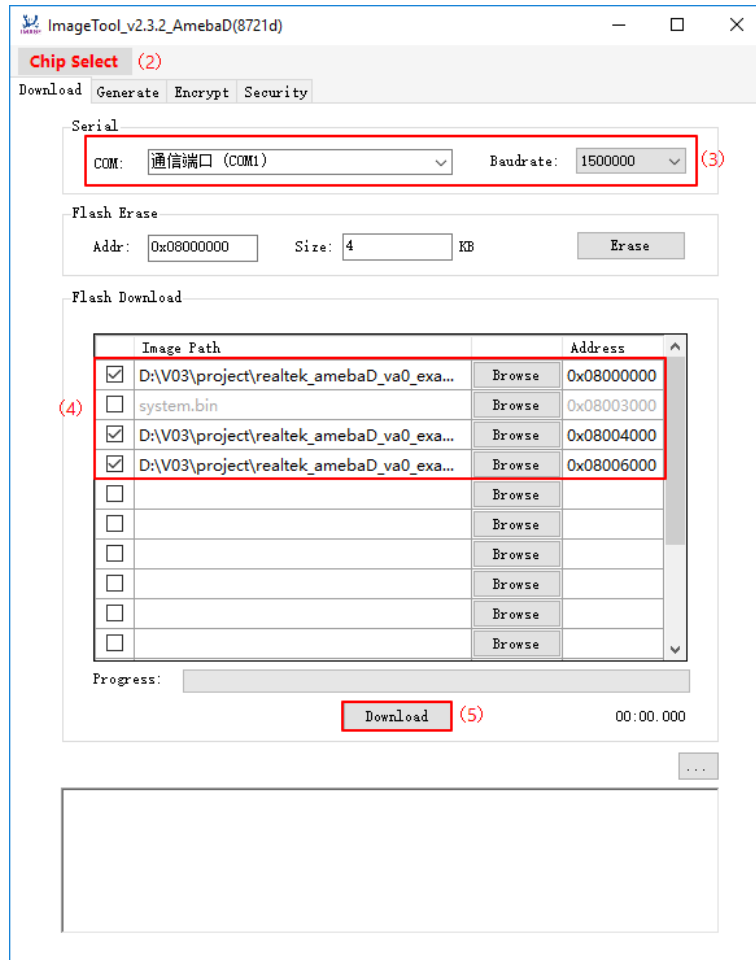


Fig 8-3 ImageTool 'Download' tabpage setting

### 8.3.1.2 Based on Command

The way based on command is the auto-download and alternate method to download images. We suggest you adopt the method “Based on Hardware Reset” because the auto-download method is dependent on the command in SDK.

**Steps:** Execute step (2) ~ (5) mentioned in section “Based on Hardware Reset”.

After that, Ameba board will enter into UART\_DOWNLOAD mode, and automatically response to the command “reboot uartburn” if this command is not removed from SDK and KM4 is running normally.

**Note:** The command “reboot uartburn” in SDK must not be removed anyhow if you want to download images in this way. Otherwise, download images according to the steps listed in section “Based on Hardware Reset”.

### 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 4-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.

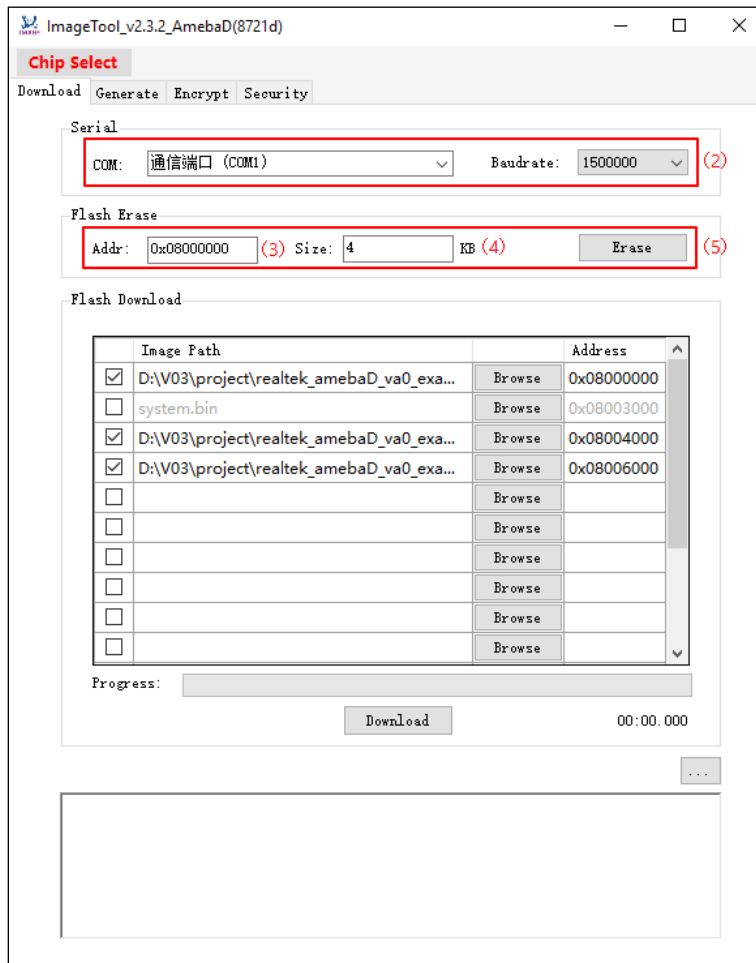


Fig 8-4 Flash erase operation

**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 of setting and getting 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 Fig 8-5, and the **Advanced Settings** window shown in Fig 8-6 popups.

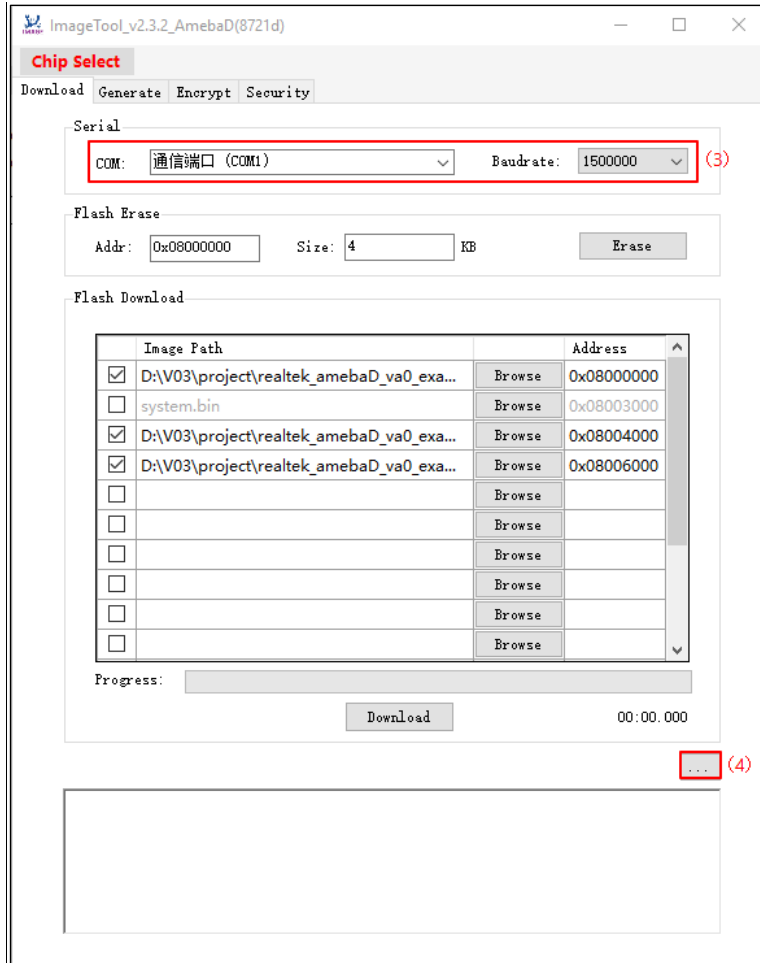


Fig 8-5 'Advanced Settings' window

- (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, or **Write** button to write status value to Flash.

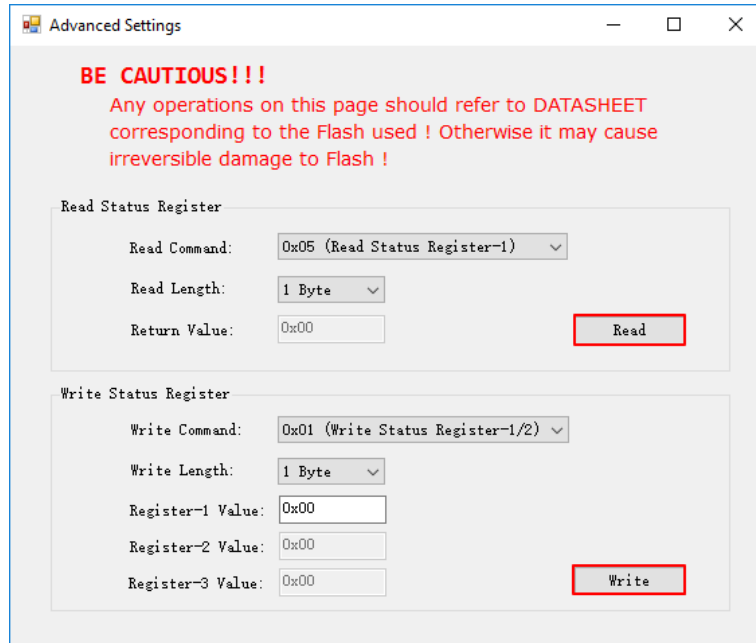


Fig 8-6 Flash status operation

In case of MXIC Flash chip, the status register is shown in Table 8-1.

- 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

Table 8-1 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	BPO	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

The 'Generate' tabpage has two functions:

- 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.

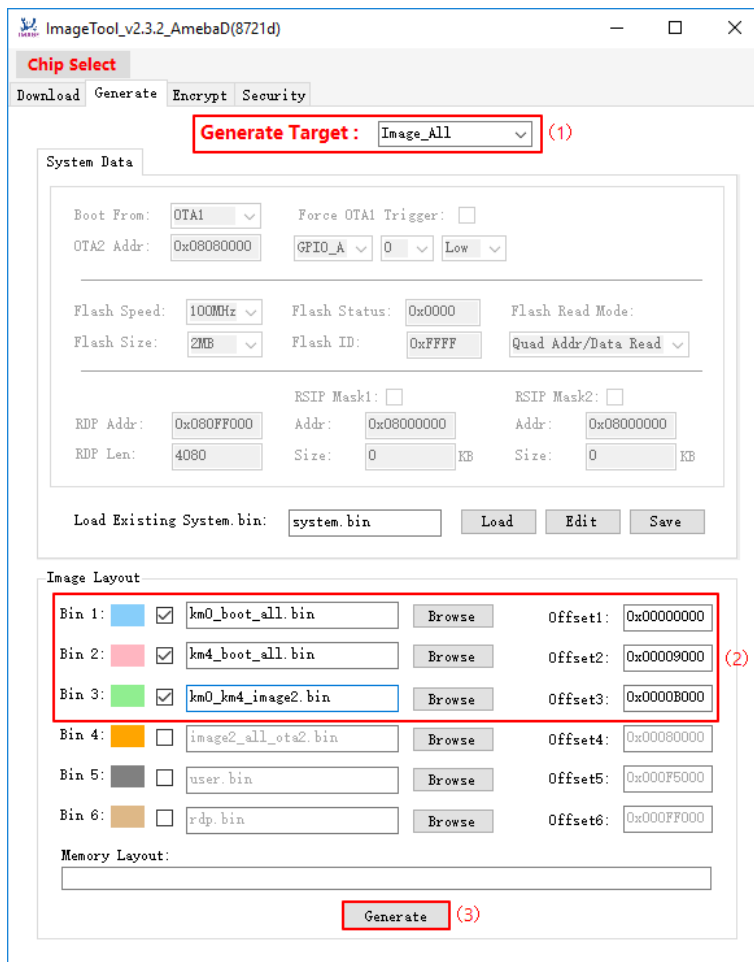


Fig 8-7 Image\_All.bin generation

### 8.4.2 Generate Cloud OTA Image

Steps to generate a Cloud OTA image are as follows:

- (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.

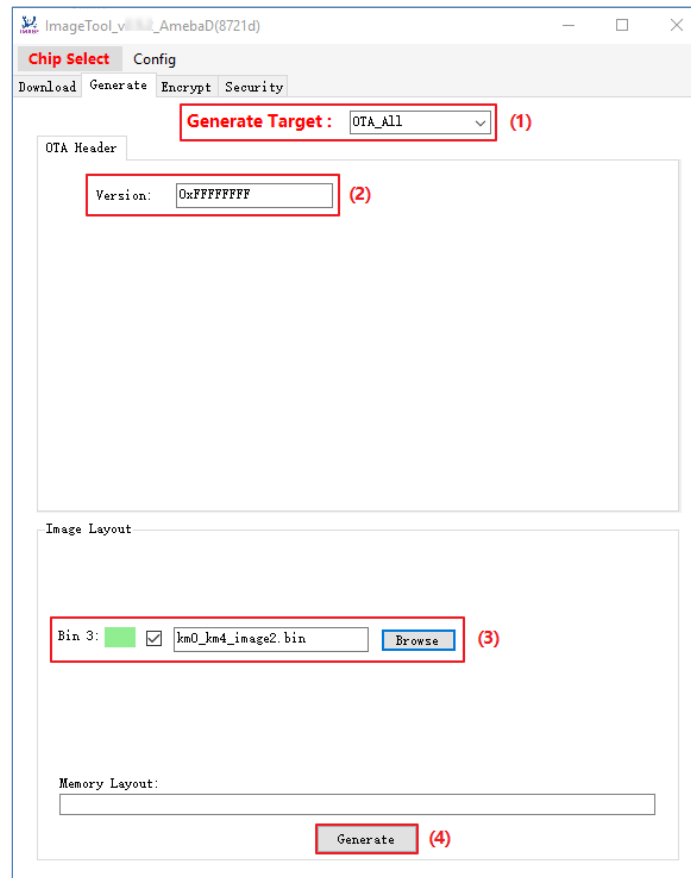


Fig 8-8 OTA\_All generation

## 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 users, but before changing these addresses, users must make sure that they are familiar with Ameba-D system and SDK.

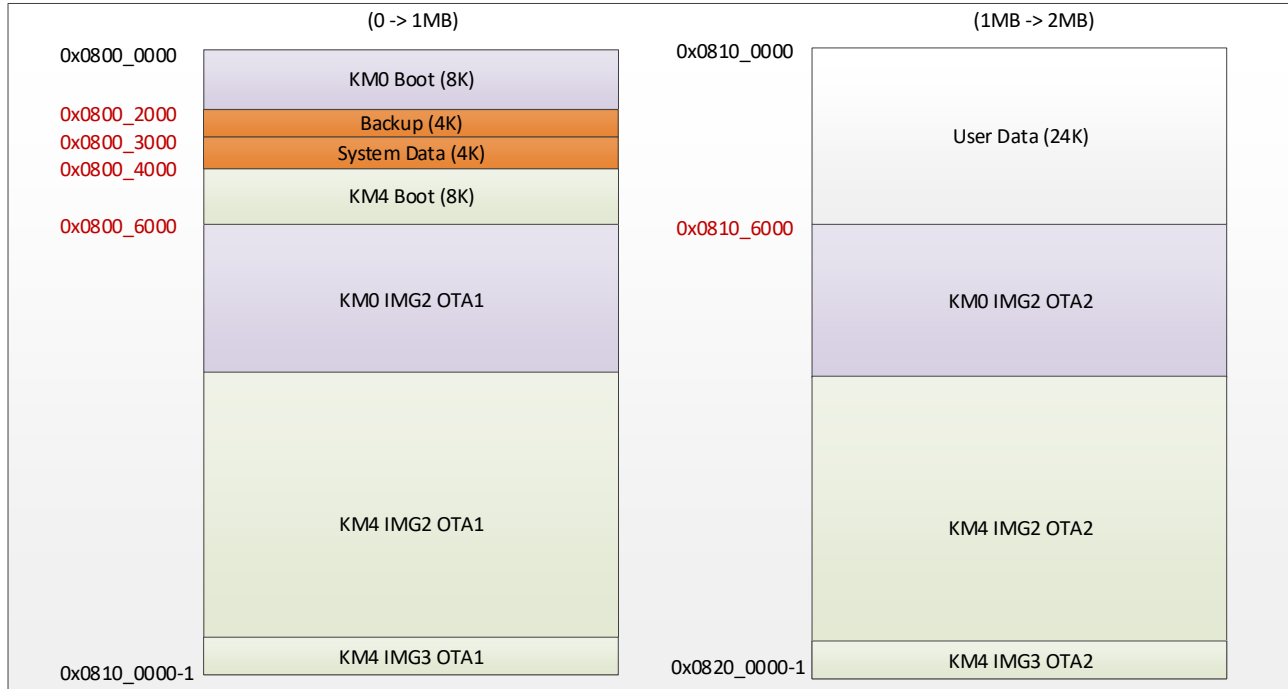


Fig 9-1 1M/2M Flash program layout

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 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	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 it's used to realize ECC secure boot.

### 9.1.1 Image Header

Normal bootloader doesn't include secure boot signature. It's just used for loading code or data to the correct destination, as shown in see Fig 9-2 and Table 9-2.



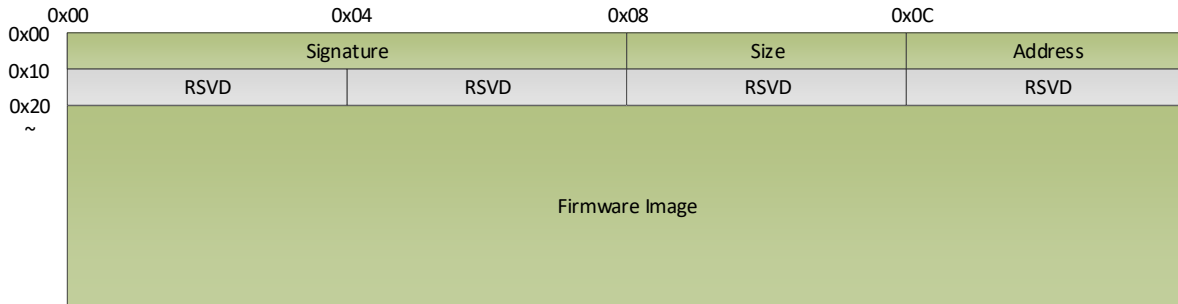


Fig 9-2 Normal image header

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.

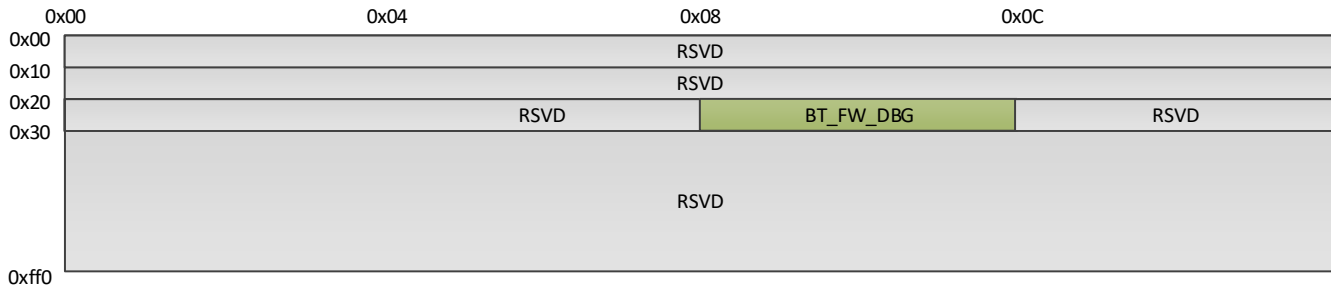


Fig 9-3 System data layout

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.

Table 9-4 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 <code>ftl_phy_page_start_addr</code> variable ( <code>rtl8721dhp_intfcfg.c</code> ). 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 <b>FAST_RECONNECT_DATA</b> macro ( <b>platform_opts.h</b> ). 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 illustrated in Fig 9-4. In this example, 2024K Flash size can be used for KM0 & KM2 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 Ameba-D system and SDK.

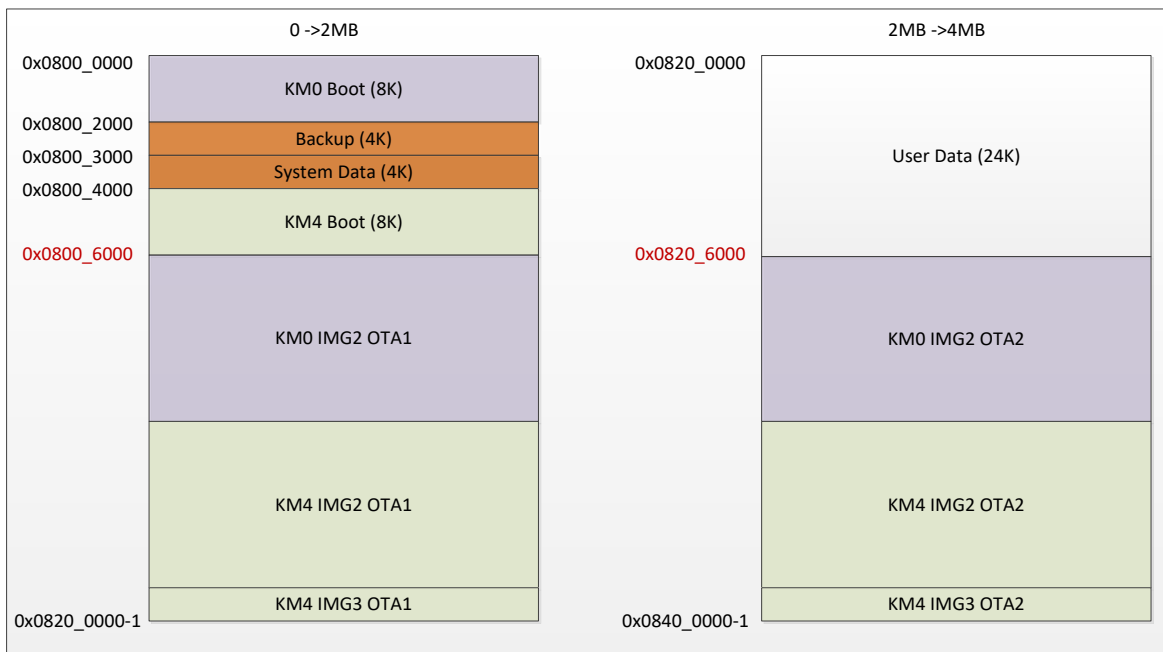


Fig 9-4 4M Flash program layout example

Table 9-5 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 needs to 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 Fig 9-4, 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 Fig 9-4, 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 your 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, re-define **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 Fig 9-4, 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 Fig 9-4, 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_COMMON	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 [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.

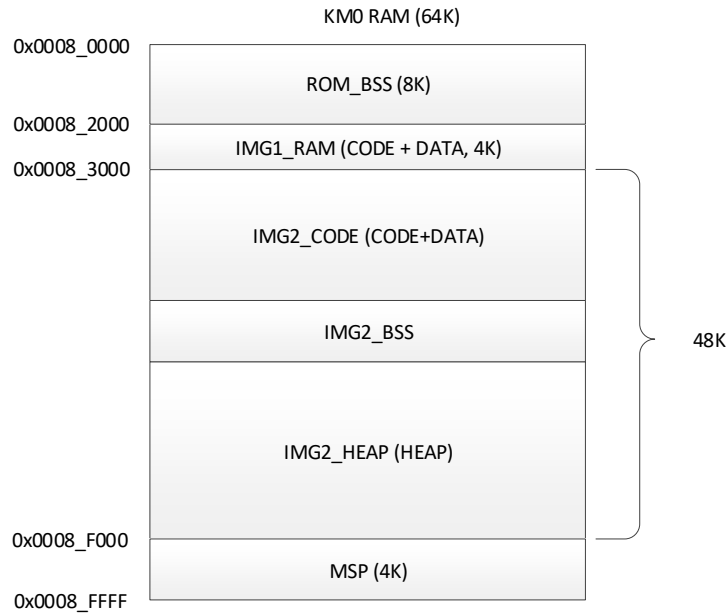


Fig 9-6 KM0 RAM program layout

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 HEAP	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.

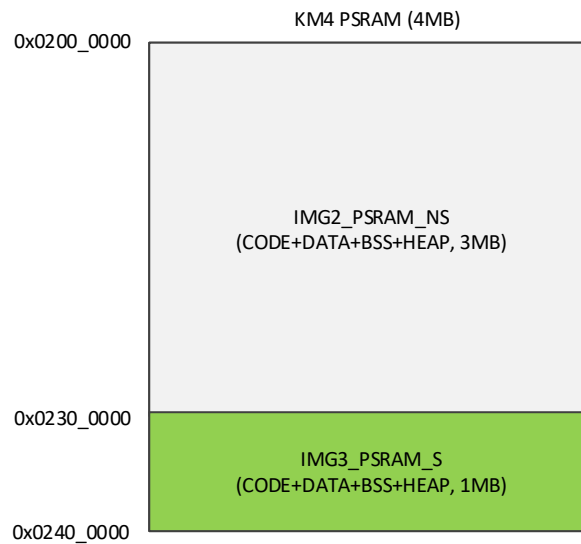


Fig 9-7 KM4 PSRAM program layout

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 Trust-Zone 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 into deepsleep mode. As a result, it is designed to retain data or code which would be lost in SRAM for some cases, such as deepsleep.

### 9.4.1 Retention SRAM Layout

The start address of Retention SRAM is 0x000C\_0000. The size is 1KB. The memory layout is illustrated 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 dword (0x000C_0004).	Yes
User Area	0x000C_0080	0x000C_0130-1	176B	Refer to RRAM_TypeDef, and RRAM_USER_RSVD [user_size] is reserved for customer use.	No
Wi-Fi Area	0x000C_0130	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 FLASH_quad_valid_cmd;
uint32_t FLASH_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_cycle0;
uint8_t FLASH_rd_dummy_cycle1;
uint8_t FLASH_rd_dummy_cycle2;

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.

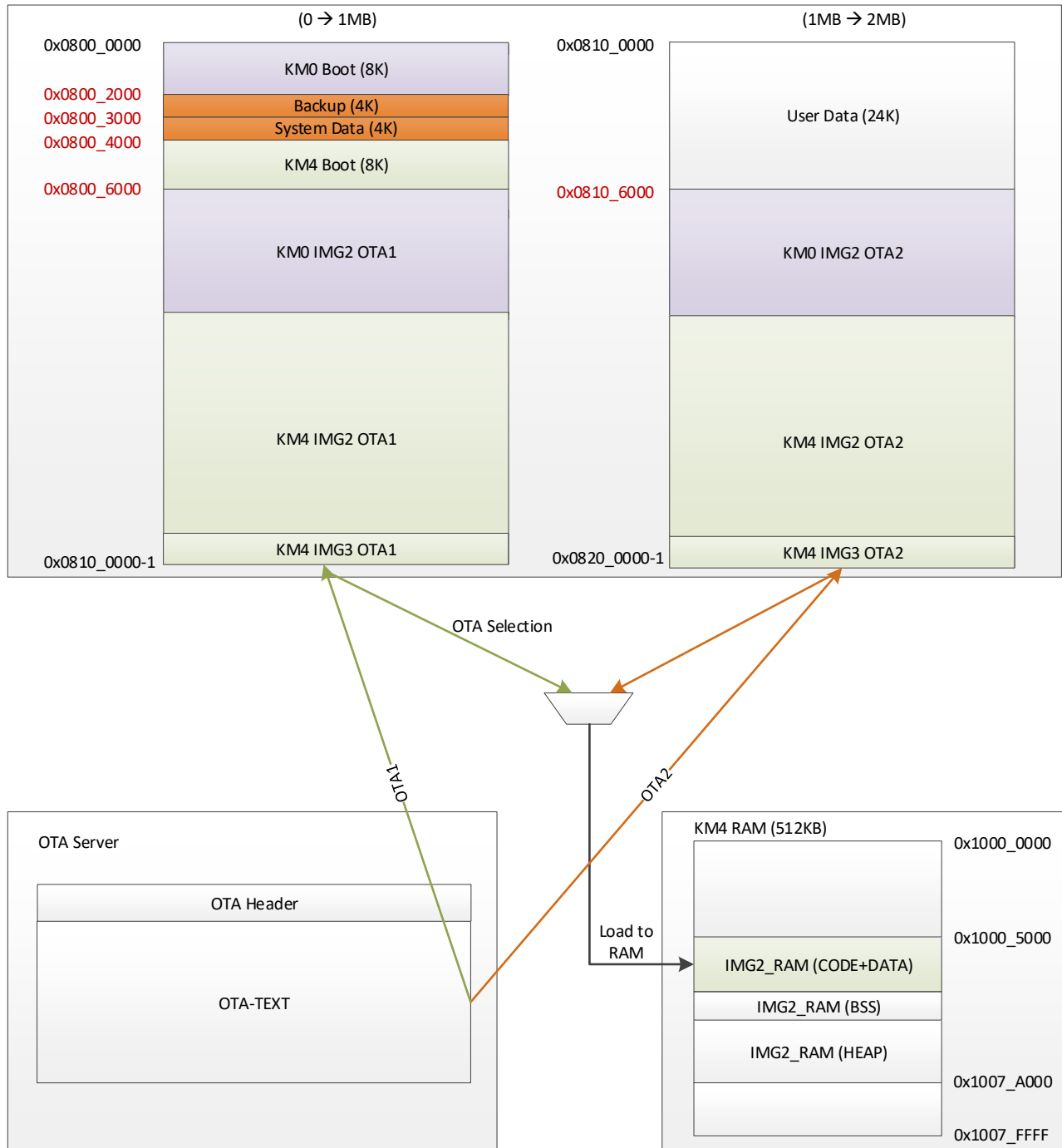


Fig 9-8 OTA program layout

## 9.5.2 OTA Header Layout

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

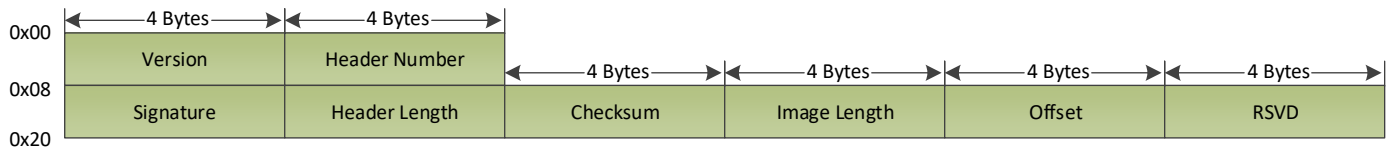


Fig 9-9 OTA header layout

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 Ameba-D, this value is 0x01
Signature	0x08	4 bytes	OTA Signature is string. For Ameba-D, this value is "OTA"
Header Length	0x0C	4 bytes	The length of OTA header. For Ameba-D, 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

# 10 File System

## 10.1 Introduction

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

## 10.2 FatFs on Flash

### 10.2.1 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` both 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` according to user’s requirements.

```
#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 Ameba-D.

### 10.2.2 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 table.
- 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.

## 10.3 FatFs on SD Card

### 10.3.1 Hardware Setup

Connect your Ameba-D 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 J29-J33 is shown as Fig 10-1.



Fig 10-1 Hardware setup

### 10.3.2 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` both to 0.

```

/* For FATFS example*/
#define CONFIG_EXAMPLE_FATFS 1
#if CONFIG_EXAMPLE_FATFS
#define CONFIG_FATFS_EN 1
#endif 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 Ameba-D. FAT file of SD card can be generated and copied directly by PC.

# 11 Inter Processor Communication (IPC)

## 11.1 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 11-1.

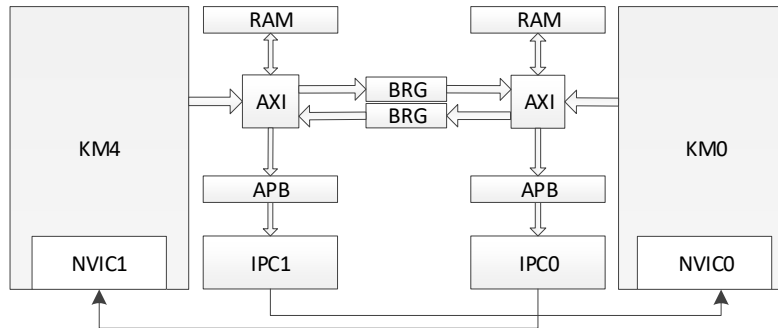


Fig 11-1 IPC system architecture

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

## 11.2 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 type: 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 matches the type on 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, call `ipc_get_message()` to get message if need.

**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.

## 11.3 IPC Program APIs

### 11.3.1 ipc\_table\_init

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

### 11.3.2 ipc\_send\_message

Items	Description
Introduction	Request interrupt to target CPU by specified channel and exchange messages between KM0 and KM4
Parameters	<ul style="list-style-type: none"> <li>● IPC_ChNum: the IPC channel number</li> <li>● Message: the message to be exchanged, and should not be stored in stack</li> </ul>
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.

### 11.3.3 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.

## 11.4 IPC Driver Code

### 11.4.1 IPC\_INTConfig

Items	Description
Introduction	Enable or disable the specified IPC channel interrupts.
Parameters	<ul style="list-style-type: none"> <li>● IPCx:                             <ul style="list-style-type: none"> <li>■ IPCM0_DEV for KM0</li> <li>■ IPCM4_DEV for KM4</li> </ul> </li> <li>● IPC_ChNum: the channel that want to be set. This parameter must be set to a value of 0 ~ 31.</li> <li>● NewState:                             <ul style="list-style-type: none"> <li>■ ENABLE</li> <li>■ DISABLE</li> </ul> </li> </ul>
Return	N/A

### 11.4.2 IPC\_IERSet

Items	Description
Introduction	Set IER of specified IPC channel interrupts

Parameters	<ul style="list-style-type: none"> <li>● IPCx:             <ul style="list-style-type: none"> <li>■ IPCM0_DEV for KM0</li> <li>■ IPCM4_DEV for KM4</li> </ul> </li> <li>● IPC_Ch: the channel that want to be enable</li> </ul>
Return	N/A

### 11.4.3 IPC\_IERGet

Items	Description
Introduction	Get IER of specified IPCx
Parameters	IPCx: <ul style="list-style-type: none"> <li>● IPCM0_DEV for KM0</li> <li>● IPCM4_DEV for KM4</li> </ul>
Return	The IER of specified IPCx

### 11.4.4 IPC\_INTRequest

Items	Description
Introduction	Request a core-to-core interrupt
Parameters	<ul style="list-style-type: none"> <li>● IPCx:             <ul style="list-style-type: none"> <li>■ IPCM0_DEV for KM0</li> <li>■ IPCM4_DEV for KM4</li> </ul> </li> <li>● IPC_ChNum: the channel that want to request interrupt. This parameter must be set to a value of 0 ~ 31.</li> </ul>
Return	N/A

### 11.4.5 IPC\_INTClear

Items	Description
Introduction	Clear a core-to-core interrupt
Parameters	<ul style="list-style-type: none"> <li>● IPCx:             <ul style="list-style-type: none"> <li>■ IPCM0_DEV for KM0</li> <li>■ IPCM4_DEV for KM4</li> </ul> </li> <li>● IPC_ChNum: the channel that want to clear interrupt. This parameter must be set to a value of 0 ~ 31.</li> </ul>
Return	N/A

### 11.4.6 IPC\_INTGet

Items	Description
Introduction	Get a core-to-core interrupt status
Parameters	IPCx: <ul style="list-style-type: none"> <li>● IPCM0_DEV for KM0</li> <li>● IPCM4_DEV for KM4</li> </ul>
Return	The ISR of specified IPCx

### 11.4.7 IPC\_CPUID

Items	Description
Introduction	Get the current CPU ID
Parameters	N/A
Return	CPU ID: <ul style="list-style-type: none"> <li>● 0: KM0</li> </ul>

	<ul style="list-style-type: none"> <li>● 1: KM4</li> </ul>
--	--

### 11.4.8 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. <ul style="list-style-type: none"> <li>● 0 ~ 7: Reserved for Realtek use</li> <li>● 8 ~ 15: Reserved for Customer use</li> </ul>
Return	Result: <ul style="list-style-type: none"> <li>● TRUE</li> <li>● FALSE</li> </ul>

### 11.4.9 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. <ul style="list-style-type: none"> <li>● 0 ~ 7: Reserved for Realtek use</li> <li>● 8 ~ 15: Reserved for Customer use</li> </ul>
Return	Result: <ul style="list-style-type: none"> <li>● TRUE</li> <li>● FALSE</li> </ul>

### 11.4.10 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.

### 11.4.11 IPC\_INTUserHandler

Items	Description
Introduction	Register a user interrupt handler for a specified IPC channel
Parameters	<ul style="list-style-type: none"> <li>● IPC_ChNum: the channel that want to register IRQ handler. This parameter must be set to a value of 0 ~ 31.</li> <li>● IrqHandler: the IRQ handler to be assigned to the specified IPC channel</li> <li>● IrqData: the pointer will be passed to the IRQ handler</li> </ul>
Return	N/A

# 12 OTA Firmware Update

## 12.1 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 principles and usage of OTA from local server. It has well-transportability to porting to OTA applications from cloud.

## 12.2 RSIP-MMU

The Flash MMU diagram is shown in Fig 12-1.

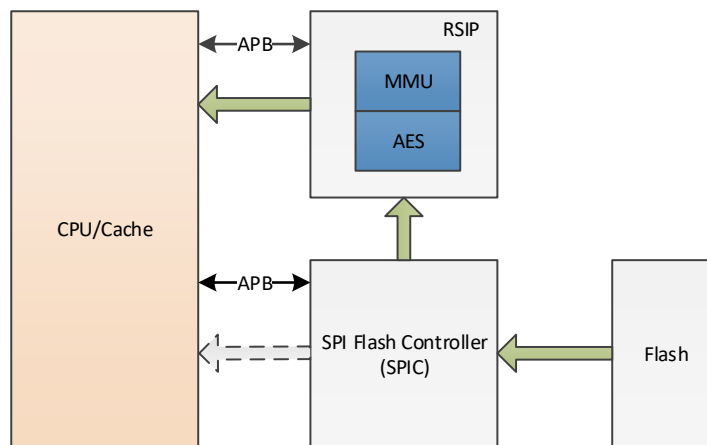


Fig 12-1 Flash MMU

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 12-2.

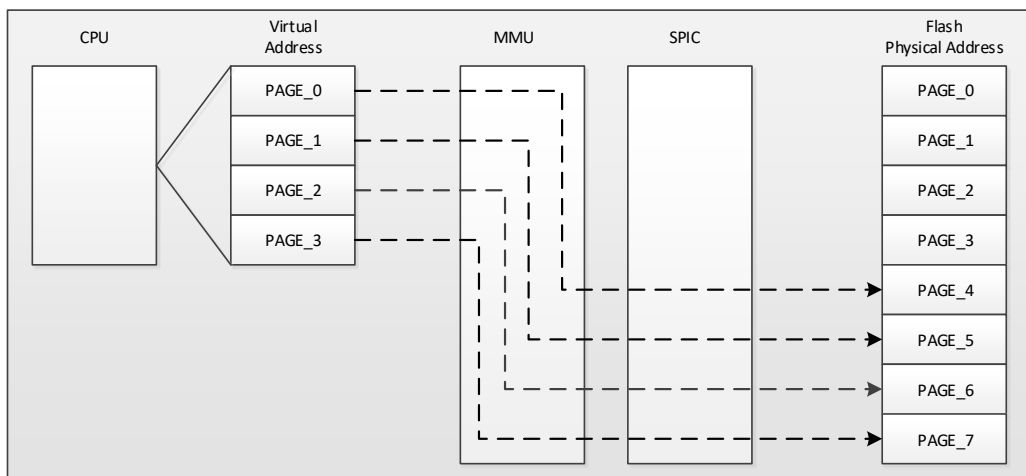


Fig 12-2 MMU virtual address to physical address

Ameba-D 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 Fig 12-3 shows. Fig 12-3 is an example for 2M Flash OTA, we need 2 MMU entries in this example.

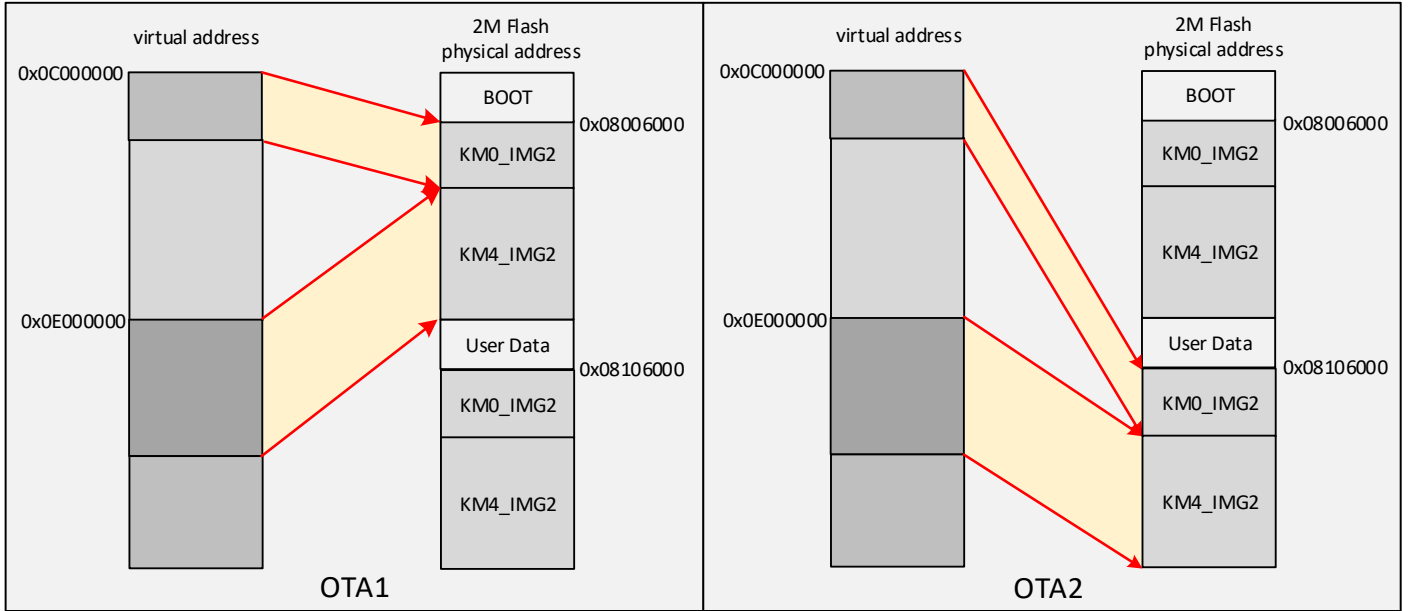


Fig 12-3 2M Flash OTA MMU

MMU entry should be set as Table 12-1.

Table 12-1 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.

### 12.3 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 Fig 12-4 shows.

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



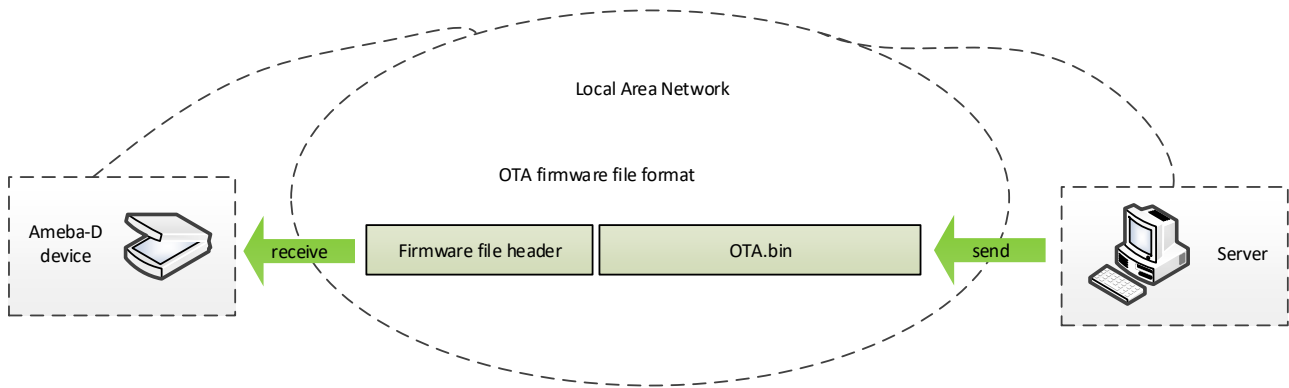


Fig 12-4 OTA update diagram

### 12.3.1 Firmware Format

The firmware format is illustrated in Fig 12-5.

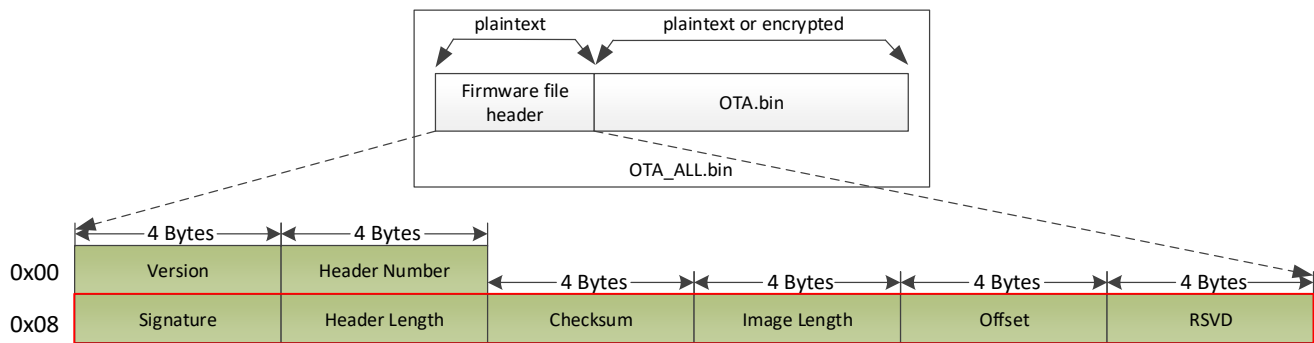


Fig 12-5 Firmware format

Table 12-2 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 Ameba-D, this value is 0x01
Signature	0x08	4 bytes	OTA Signature is string. For Ameba-D, this value is "OTA"
Header Length	0x0C	4 bytes	The length of OTA header. For Ameba-D, 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

### 12.3.2 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 boot from new firmware next time.
- (8) OTA finish and reset the device. Then it would boot from new firmware.

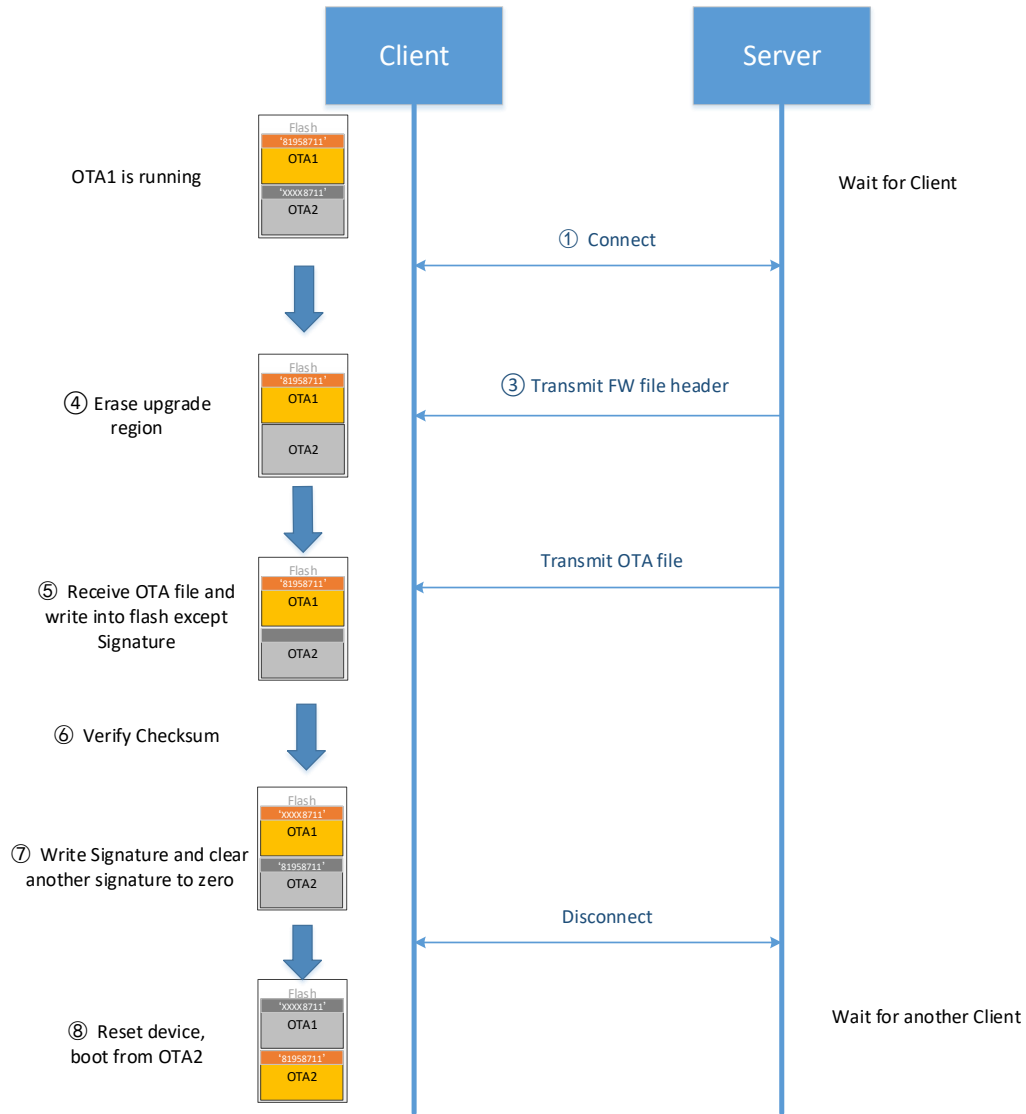


Fig 12-6 OTA operation flow

### 12.3.3 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 rti8721d\_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

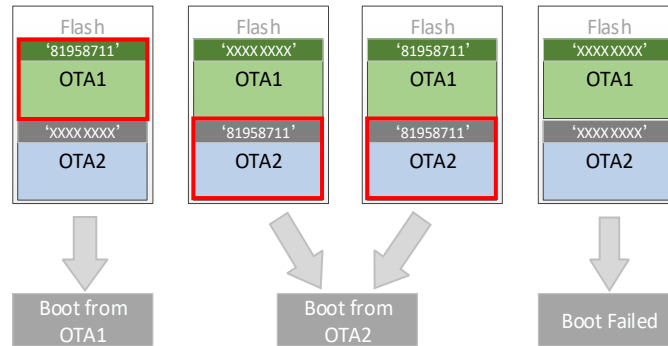


Fig 12-7 OTA select diagram

### 12.3.4 Address Remapping

In bootloader, MMU entry0 and entry1 would be set as Table 12-1 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.

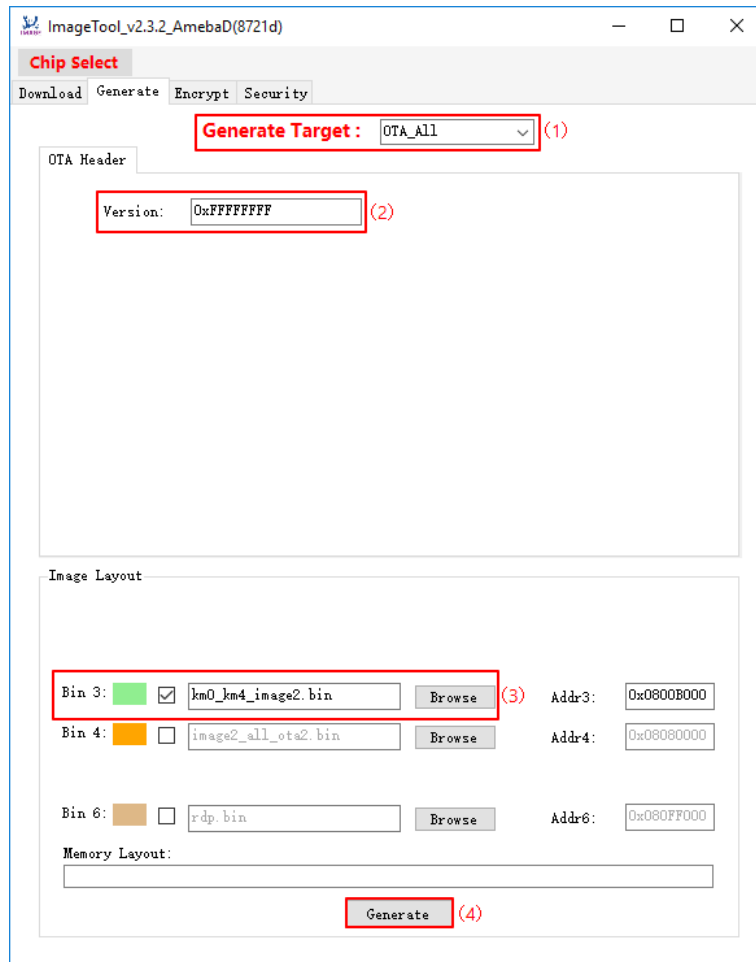
As Fig 12-3 illustrates, when boot from OTA, CPU may access different physical address.

- 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 0x0C00\_0000, it actually accesses physical address 0x0800\_6000.
- 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 0x0C00\_0000, it actually accesses physical address 0x0810\_6000.

### 12.3.5 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 depends on 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

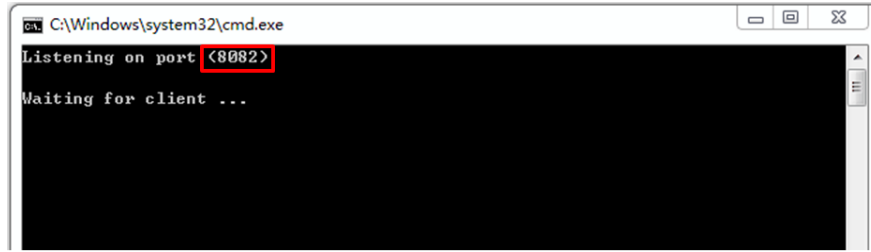
DownloadServer.exe	2016/11/7 16:12	34 KB
<b>OTA_All.bin</b>	2016/11/7 15:21	657 KB
start.bat	2016/11/6 11:43	1 KB

(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 ...
```

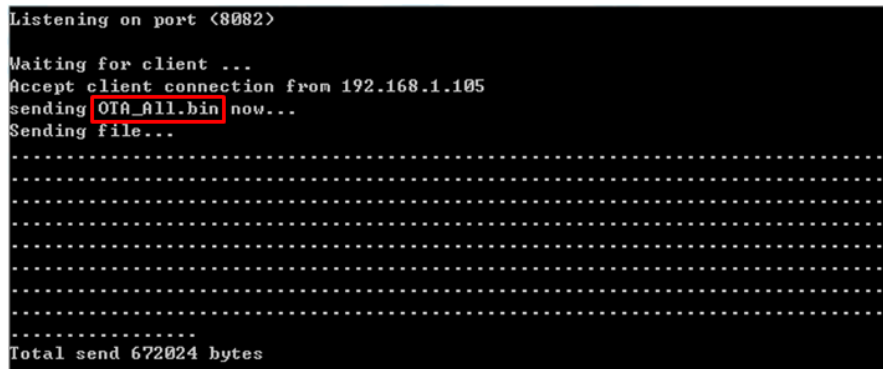
(6) Click the **start.bat**, start to download server program.



- (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

```
# ATWO=192.168.0.20[8082]  
[ATWO]: _AT_WLAN_OTA_UPDATE_  
  
[MEM] After do cmd, available heap 28400  
  
#  
[ota_update_local_task] Update task start
```

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.

### 12.3.6 OTA Firmware Swap

Fig 12-8 shows the firmware swap procedure after OTA upgrade.

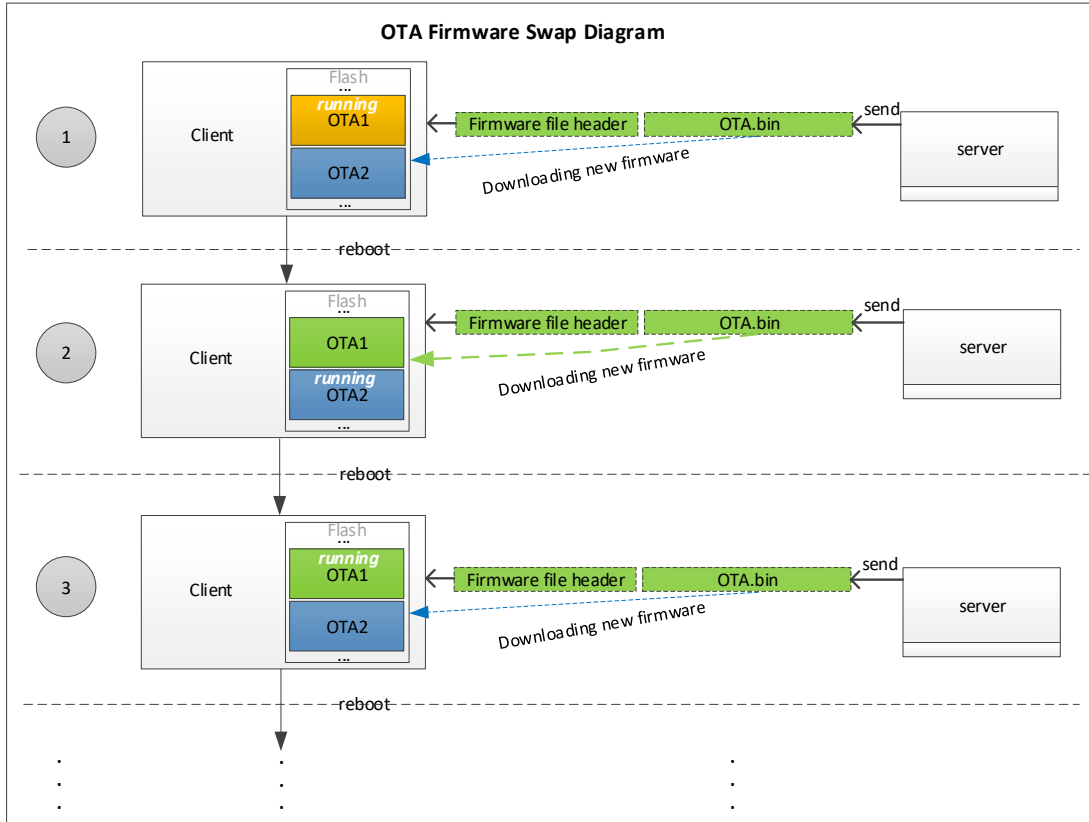


Fig 12-8 OTA firmware swap procedure

## 12.4 User Configuration

Users can find the following configure items in `rtl8721d_bootcfg.c`.

- OTA start address

```

/*
 * @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 */
};
    
```

- User-defined MMU configure

```

/**
 * @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[] = {
    /*UAddrStart,  UAddrEnd,          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

```

/**
 * @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.
 */
BOOT_RAM_DATA_SECTION
FwCheckFunc FwCheckCallback = NULL;

```

# 13 eFuse

## 13.1 Introduction

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

The total size of physical eFuse is 512 bytes, and divided into two parts by software. As Fig 13-1 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 multi-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 in any mode.

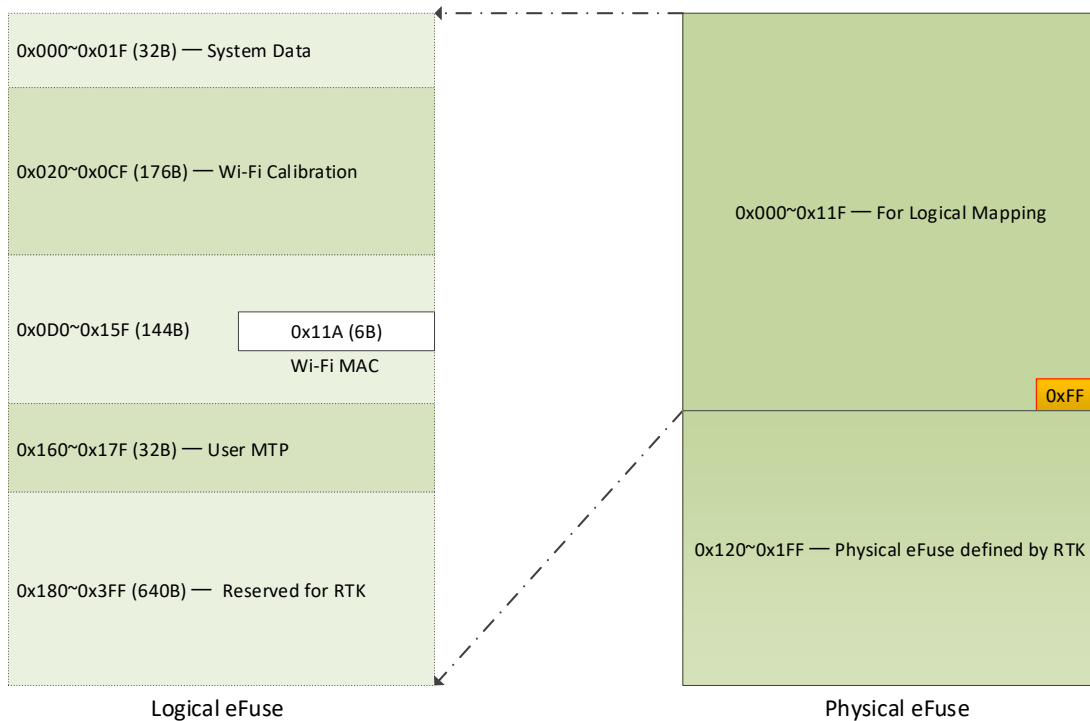


Fig 13-1 eFuse diagram

## 13.2 Power Requirement

The power requirement of eFuse operation is listed in Table 13-1.

Table 13-1 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.



### 13.3 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 mistakes.
- Reboot the chip, and the System Configure Registers will load the new values.

### 13.4 Physical eFuse

For detailed information about physical eFuse of Ameba-D, refer to AN0411.

### 13.5 Logical eFuse

#### 13.5.1 Logical eFuse Layout

There are 1024 bytes logical eFuse in Ameba-D, as Table 13-2 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 13-2 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

20~25	2.4G CCK Index	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
26~2A	2.4G BW40 Index	000	21	87	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
2B	2.4G Difference	010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
32~3F	5G BW40 Index	020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
40	5G Difference	030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
C8	channel plan	040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
C9	Crystal Calibration	050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
CA	Thermal meter	060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
11A~11F	MAC Address	070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		0A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		0B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		0C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		0D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		0E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		0F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		100	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		110	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		120	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		130	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		140	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		150	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		160	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		170	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		180	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		190	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		1A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		1B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		1C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		1D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		1E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
		1F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

For Wi-Fi:  
Users only need to fill on these eFuse locations for MP (Mass Production) calibration.

Fig 13-2 Wi-Fi eFuse data

### 13.5.2 Wi-Fi 2.4G Power Index

The address and specification of 2.4G power index is shown in Fig 13-3 and Fig 13-4.

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

Fig 13-3 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.
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.

Fig 13-4 2.4G Wi-Fi power index specification

### 13.5.3 Wi-Fi 5G Power Index

The address and specification of 5G power index is shown in Fig 13-5 and Fig 13-6.

	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

Fig 13-5 5G Wi-Fi power index address

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.
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.

Fig 13-6 5G Wi-Fi power index specification

### 13.5.4 Wi-Fi Channel Plan

The address of Wi-Fi channel plan is shown in Fig 13-7.

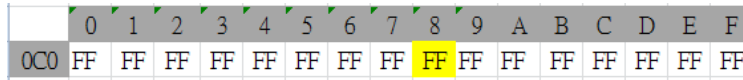


Fig 13-7 Channel plan address

eFuse	Name	Bit	Default	Comment
C8	Channel Plan	[7]	0h	Software configure mode <ul style="list-style-type: none"> <li>● 0h: Enable software configure (refer to Channel Plane Domain Code)</li> <li>● 1h: Disable software configure (can't change Channel Plan Setting)</li> </ul>
		[6:0]	7Fh	Channel Plan <ul style="list-style-type: none"> <li>● 0x20: 2G Worldwide 13, Active scan Ch1 ~ Ch11, Passive scan Ch12, Ch13</li> <li>● 0x7F: <ul style="list-style-type: none"> <li>■ 2G Worldwide 13, Active scan Ch1 ~ Ch11, Passive scan Ch12, Ch13</li> <li>■ 5G Worldwide, all bands are Passive scan.</li> </ul> </li> </ul>

Frequently-used country codes and the corresponding channel plans are listed in Table 13-3, Table 13-4 and Table 13-5.

Table 13-3 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 13-4 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 13-5 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, 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	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, 124, 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, 124, 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, 124, 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
0x4D	5G_01	36, 40, 44, 48 52, 56, 60, 64 100, 104, 108, 112, 116, 132, 136, 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, 124, 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, 124, 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.

### 13.5.5 Wi-Fi Crystal Calibration

The address of Wi-Fi crystal calibration is shown in Fig 13-8.

	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

Fig 13-8 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

### 13.5.6 Wi-Fi Thermal Meter

The address of Wi-Fi thermal meter is shown in Fig 13-9.

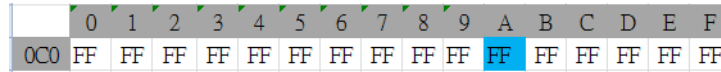


Fig 13-9 Thermal Meter address

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

### 13.5.7 Wi-Fi MAC Address

The Wi-Fi MAC address is shown in Fig 13-10.

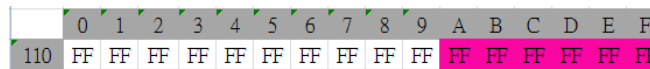


Fig 13-10 MAC Address

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

### 13.5.8 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]

### 13.5.9 BLE

Address Offset	Bit	Default	Description
190 ~ 195h	[47:0]	0xFFFFFFFFFFFF	BT address
196h	[7:0]	0x08	<ul style="list-style-type: none"> <li>● Bit[0]: Fixed to 0</li> <li>● Bit[1]: Tx gain K valid bit</li> <li>● Bit[2]: Flatness K valid bit</li> <li>● Bit[3]: Fixed to 1</li> <li>● Bit[7:4]: Fixed to 0</li> </ul> <b>Note:</b> 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 <b>Note:</b> 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 <ul style="list-style-type: none"> <li>● 0xFF: Disable logic map IQK/KOK values</li> <li>● 0xFE: Enable logic map IQK/KOK values</li> </ul>
1A6h ~ 1A9h	[31:0]	0xFFFFFFFF	Logic map IQK values
1AAh ~ 1ABh	[15:0]	0xFFFF	Logic map KOK values

### 13.5.10 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

## 13.6 eFuse PG APIs

Items	API	Comment
Low Level API	EFUSE_PMAP_WRITE8	Physical map writes one byte
	EFUSE_PMAP_READ8	Physical map reads 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)

### 13.6.1 Low Level APIs

#### 13.6.1.1 EFUSE\_PMAP\_WRITE8

Items	Description
Introduction	Physical map writes one byte.
Parameters	<ul style="list-style-type: none"> <li>● CtrlSetting: eFuse control setting read from REG_LP_EFUSE_CTRL1 (suggest 0)</li> <li>● Addr: eFuse physical address</li> <li>● Data: 1 byte data to write</li> <li>● L25OutVoltage: L25EOUTVOLTAGE.</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● _TRUE: Write is ok.</li> <li>● _FALSE: Write is failed.</li> </ul>

#### 13.6.1.2 EFUSE\_PMAP\_READ8

Items	Description
Introduction	Physical map reads one byte.
Parameters	<ul style="list-style-type: none"> <li>● CtrlSetting: eFuse control setting read from REG_LP_EFUSE_CTRL1 (suggest 0)</li> <li>● Addr: eFuse physical address</li> <li>● Data: 1 byte data buffer for eFuse data read</li> </ul>

	<ul style="list-style-type: none"> <li>● L25OutVoltage: L25EOUTVOLTAGE.</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● _TRUE: Read is ok.</li> <li>● _FALSE: Read is failed.</li> </ul>

### 13.6.1.3 EFUSE\_LMAP\_WRITE

Items	Description
Introduction	Write eFuse logical address by length
Parameters	<ul style="list-style-type: none"> <li>● addr: logical address, should be word-aligned</li> <li>● cnts: byte number, should be even</li> <li>● data: data buffer to be write</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● _SUCCESS: Write is ok.</li> <li>● _FAIL: Write is failed.</li> </ul>

### 13.6.1.4 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: <ul style="list-style-type: none"> <li>● _SUCCESS: Read is ok.</li> <li>● _FALSE: Read is failed.</li> </ul>

## 13.6.2 Mbed APIs

### 13.6.2.1 efuse\_otp\_write

Items	Description
Introduction	Write content to OTP eFuse by length (mbed API)
Parameters	<ul style="list-style-type: none"> <li>● address: Specifies the offset of the programmed OTP.</li> <li>● len: Specifies the data length of programmed data.</li> <li>● buf: Specified the data to be programmed.</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● 0: Success</li> <li>● -1: Failure</li> </ul>

### 13.6.2.2 efuse\_otp\_read

Items	Description
Introduction	Read eFuse OTP content by length (mbed API)
Parameters	<ul style="list-style-type: none"> <li>● address: Specifies the offset of the OTP.</li> <li>● len: Specifies the length of readback data.</li> <li>● buf: Specified the address to save the readback data.</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● 0: Success</li> <li>● -1: Failure</li> </ul>

### 13.6.2.3 efuse\_mtp\_write

Items	Description
-------	-------------



Introduction	Write user's content to USER MTP Space (0x160~0x17F)
Parameters	<ul style="list-style-type: none"> <li>● data: Specified the data to write</li> <li>● len: Specified the data length to write</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● 0 ~ 32: Success</li> <li>● 0 or -1: Failure</li> </ul>

### 13.6.2.4 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

## 13.7 eFuse PG Command

Items	Offset	Command
Tx Power Index	0x20~0x2B	iwpriv config_set wmap, 020, 20202020202020202020202020202002
Channel plan, XTAL & Thermal & RTK reserved	0xC8~0xCF	iwpriv config_set wmap, 0C8, 20201A05000000FF <sup>1</sup>
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.

# 14 Hardware Crypto Engine

## 14.1 Introduction

Hardware Crypto Engine is used to authenticate, encrypt and decrypt packets. It can 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, 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.

## 14.2 Hardware Crypto Engine APIs

### 14.2.1 Crypto Engine Initialization API

#### 14.2.1.1 rtl\_cryptoEngine\_init

Items	Description
Introduction	Hardware crypto engine initialization
Parameters	N/A
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: initialization is ok.</li> </ul>

### 14.2.2 Hash APIs

#### 14.2.2.1 rtl\_crypto\_md5

Items	Description
Introduction	MD5 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of MD5 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: MD5 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.2 rtl\_crypto\_md5\_init

Items	Description
Introduction	MD5 initialization (used for sequential hash)
Parameters	N/A
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: MD5 initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.3 rtl\_crypto\_md5\_update**

Items	Description
Introduction	MD5 block calculation (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: MD5 update is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.4 rtl\_crypto\_md5\_final**

Items	Description
Introduction	MD5 last block calculation (used for sequential hash)
Parameters	pDigest: Result of MD5 function
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: MD5 final is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.5 rtl\_crypto\_sha1**

Items	Description
Introduction	SHA1 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of SHA1 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA1 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.6 rtl\_crypto\_sha1\_init**

Items	Description
Introduction	SHA1 initialization (used for sequential hash)
Parameters	N/A
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA1 initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.7 rtl\_crypto\_sha1\_update**

Items	Description
Introduction	SHA1 block calculation (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA1 update is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.8 rtl\_crypto\_sha1\_final**

Items	Description
-------	-------------

Introduction	SHA1 last block calculation (used for sequential hash)
Parameters	pDigest: Result of SHA1 function
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA1 final is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.9 rtl\_crypto\_sha2

Items	Description
Introduction	SHA2 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● sha2type: SHA2 type, SHA2_224 or SHA2_256</li> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of SHA2 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA2 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.10 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: <ul style="list-style-type: none"> <li>● SUCCESS: SHA2 initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.11 rtl\_crypto\_sha2\_update

Items	Description
Introduction	SHA2 block calculation (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA2 update is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.12 rtl\_crypto\_sha2\_final

Items	Description
Introduction	SHA2 last block calculation (used for sequential hash)
Parameters	pDigest: Result of SHA2 function
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: SHA2 final is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.13 rtl\_crypto\_hmac\_md5

Items	Description
Introduction	HMAC-MD5 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● key: HMAC-MD5 key, need to be 4-byte aligned</li> <li>● keylen: Key length</li> <li>● message: Plaintext</li> </ul>

	<ul style="list-style-type: none"> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of HMAC-MD5 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-MD5 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.14 rtl\_crypto\_hmac\_md5\_init**

Items	Description
Introduction	HMAC-MD5 initialization (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● key: HMAC-MD5 key, need to be 4-byte aligned</li> <li>● keylen: Key length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-MD5 initialization ok</li> <li>● Others: fail, refer to ERRNO</li> </ul>

**14.2.2.15 rtl\_crypto\_hmac\_md5\_update**

Items	Description
Introduction	HMAC-MD5 block calculation (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> </ul>
Return	retval status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-MD5 update is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.16 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: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-MD5 final is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.17 rtl\_crypto\_hmac\_sha1**

Items	Description
Introduction	HMAC-SHA1 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● key: HMAC-SHA1 key, need to be 4-byte aligned</li> <li>● keylen: key length</li> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of HMAC-SHA1 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA1 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.18 rtl\_crypto\_hmac\_sha1\_init**

Items	Description
Introduction	HMAC-SHA1 initialization (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● key: HMAC-SHA1 key, need to be 4-byte aligned</li> </ul>

	<ul style="list-style-type: none"> <li>● keylen: key length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA1 initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.19 rtl\_crypto\_hmac\_sha1\_update**

Items	Description
Introduction	HMAC-SHA1 block calculation (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA1 update is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.20 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: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA1 final is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.21 rtl\_crypto\_hmac\_sha2**

Items	Description
Introduction	HMAC-SHA2 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● sha2type: SHA2 type, SHA2_224 or SHA2_256</li> <li>● key: HMAC-SHA2 key, need to be 4-byte aligned</li> <li>● keylen: Key length</li> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of HMAC-SHA2 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA2 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.22 rtl\_crypto\_hmac\_sha2\_init**

Items	Description
Introduction	HMAC-SHA2 initialization (used for sequential hash)
Parameters	<ul style="list-style-type: none"> <li>● sha2type: SHA2 type, SHA2_224 or SHA2_256</li> <li>● key: HMAC-SHA2 key, need to be 4-byte aligned</li> <li>● keylen: Key length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA2 initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.2.23 rtl\_crypto\_hmac\_sha2\_update**

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

Parameters	<ul style="list-style-type: none"> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA2 update is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.24 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: <ul style="list-style-type: none"> <li>● SUCCESS: HMAC-SHA2 final is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.2.25 rtl\_crypto\_poly1305

Items	Description
Introduction	poly1305 calculation digest
Parameters	<ul style="list-style-type: none"> <li>● key: poly1305 key, need to be 4-byte aligned</li> <li>● keylen: Key length</li> <li>● message: Plaintext</li> <li>● msglen: Plaintext length</li> <li>● pDigest: Result of poly1305 function</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: poly1305 is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

### 14.2.3 Cipher APIs

#### 14.2.3.1 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	<ul style="list-style-type: none"> <li>● key: Cipher key.</li> <li>● keylen: Key length</li> </ul>
Return	status value: <ul style="list-style-type: none"> <li>● SUCCESS: initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

#### 14.2.3.2 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	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● iv: Initial vector</li> <li>● ivlen: iv length</li> <li>● pResult: Point to cipher result</li> </ul>

Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: encryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>
--------	---

### 14.2.3.3 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	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● iv: Initial vector</li> <li>● ivlen: IV (initialization vector) length</li> <li>● pResult: Point to cipher result</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: decryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

### 14.2.3.4 rtl\_crypto\_aes\_gcm\_init

Items	Description
Introduction	AES-GCM initialization
Parameters	<ul style="list-style-type: none"> <li>● key: Cipher key</li> <li>● keylen: Key length</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

### 14.2.3.5 rtl\_crypto\_aes\_gcm\_encrypt

Items	Description
Introduction	AES-GCM encryption
Parameters	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● iv: Initial vector</li> <li>● aad: Point to AAD (additional authentication data)</li> <li>● aadlen: AAD length</li> <li>● pResult: Point to cipher result</li> <li>● pTag: Point to MAC (Message Authentication Code)</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: encryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

### 14.2.3.6 rtl\_crypto\_aes\_gcm\_decrypt

Items	Description
Introduction	AES-GCM decryption
Parameters	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● iv: Initial vector</li> <li>● aad: Point to AAD (additional authentication data)</li> <li>● aadlen: AAD length</li> <li>● pResult: Point to cipher result</li> </ul>



	<ul style="list-style-type: none"> <li>● pTag: Point to MAC (Message Authentication Code)</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: decryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.3.7 rtl\_crypto\_chacha\_init**

Items	Description
Introduction	ChaCha20 initialization
Parameters	key: Cipher key.
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: initialization OK</li> <li>● Others: fail, refer to ERRNO</li> </ul>

**14.2.3.8 rtl\_crypto\_chacha\_encrypt**

Items	Description
Introduction	ChaCha20 encryption
Parameters	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● iv: Point to IV (initialization vector)</li> <li>● count: Counter value.</li> <li>● pResult: Point to cipher result</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: encryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.3.9 rtl\_crypto\_chacha\_decrypt**

Items	Description
Introduction	ChaCha20 decryption
Parameters	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● iv: Point to IV (initialization vector)</li> <li>● count: Counter value</li> <li>● pResult: Point to cipher result</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: decryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.3.10 rtl\_crypto\_chacha\_poly1305\_init**

Items	Description
Introduction	ChaCha Poly1305 initialization
Parameters	key: Cipher key
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: initialization is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

**14.2.3.11 rtl\_crypto\_chacha\_poly1305\_encrypt**

Items	Description
Introduction	ChaCha Poly1305 encryption

Parameters	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● nonce: Random value</li> <li>● aad: Point to AAD (additional authentication data)</li> <li>● aadlen: AAD length</li> <li>● pResult: Point to cipher result</li> <li>● pTag: Point to MAC (Message Authentication Code)</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: encryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

### 14.2.3.12 rtl\_crypto\_chacha\_poly1305\_decrypt

Items	Description
Introduction	ChaCha Poly1305 decryption
Parameters	<ul style="list-style-type: none"> <li>● message: Point to source message</li> <li>● msglen: Message length</li> <li>● nonce: Random value</li> <li>● aad: Point to AAD (additional authentication data)</li> <li>● aadlen: AAD length</li> <li>● pResult: Point to cipher result</li> <li>● pTag: Point to MAC (Message Authentication Code)</li> </ul>
Return	Status value: <ul style="list-style-type: none"> <li>● SUCCESS: decryption is ok.</li> <li>● Others: fail, refer to ERRNO.</li> </ul>

## 14.3 Hardware Crypto Engine APIs Usage

### 14.3.1 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.

### 14.3.2 Starting Crypto Engine Calculation

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

#### 14.3.2.1 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-byte aligned if it's a local variable.
- Or call the following line to restore data from memory before reading digest when calculation is finished if it's a local variable.

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

### 14.3.2.2 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.

## 14.4 Demo Code Path

Hardware Crypto Engine demo code locates in folder: **project/realtek\_amebaD\_va0\_example/example\_sources/CRYPTO**

# 15 User Configuration

Some functions have complex parameters, it is hard to give standard APIs 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.

## 15.1 Configuration File List

The user configuration files are listed in Table 15-1.

Table 15-1 User configuration file

Configuration file	Configuration item	Image
rtl8721dlp_flashcfg.c	Used to configure Flash settings: <ul style="list-style-type: none"> <li>● Flash_Speed</li> <li>● Flash_ReadMode</li> <li>● Flash_AVL</li> </ul> You can also configure your Flash use flash_init_userdef, if your Flash is not in Flash_AVL.	KM0 image2
Rtl8721dlp_pinmapcfg.c	Used to reduce I/O power leakage: <ul style="list-style-type: none"> <li>● Per pin function configuration</li> <li>● Per pin function pull up &amp; pull down</li> <li>● Per pin sleep pull up &amp; pull down</li> </ul>	KM0 image2
Rtl8721dlp_sleepcfg.c	<ul style="list-style-type: none"> <li>● Sleep/deepsleep power management</li> <li>● Sleep/deepsleep wakeup event</li> <li>● Sleep/deepsleep wake pin</li> </ul>	KM0 image2
rtl8721d_bootcfg.c	Used to configure bootloader: <ul style="list-style-type: none"> <li>● Flash_MMU_Config</li> <li>● RSIP_Mask_Config</li> <li>● Force_OTA1_GPIO</li> <li>● Boot_Log_En</li> </ul>	KM0 & KM4 bootloader
rtl8721d_ipccfg.c	Used to define your IPC channels	KM0 & KM4 image2
rtl8721dlp_intfcfg.c	Enable/disable low power Rx	KM0 image2
rtl8721dhp_intfcfg.c	<ul style="list-style-type: none"> <li>● PSRAM configuration:                             <ul style="list-style-type: none"> <li>■ Enable PSRAM</li> <li>■ Enable PSRAM calibration function</li> <li>■ Enable PSRAM retention</li> </ul> </li> <li>● SDIO host configuration:                             <ul style="list-style-type: none"> <li>■ SDIO Host bus speed</li> <li>■ SDIO Host bus width</li> <li>■ Card Detect pin</li> <li>■ Write Protection pin</li> </ul> </li> <li>● FTL parameter configuration:                             <ul style="list-style-type: none"> <li>■ The number of physical map pages</li> <li>■ The offset of Flash sectors which is allocated to FTL physical map</li> </ul> </li> </ul>	KM4 image2

## 15.2 flashcfg.c

The Flash configuration items for users are listed in Table 15-2.

Table 15-2 User Flash configuration

Item	Comment	Default
Flash_Speed	Indicates the Flash baud rate. It can be one of the following value:	0xFFFF

	<ul style="list-style-type: none"> <li>● 0xFFFF: 80MHz</li> <li>● 0x7FFF: 100MHz</li> <li>● 0x3FFF: 67MHz</li> <li>● 0x1FFF: 57MHz</li> <li>● 0x0FFF: 50MHz</li> </ul>	
Flash_ReadMode	<p>Indicates the Flash read I/O mode. It can be one of the following value:</p> <ul style="list-style-type: none"> <li>● 0xFFFF: Read quad I/O, Address &amp; Data 4 bits mode</li> <li>● 0x7FFF: Read quad O, just data 4 bits mode</li> <li>● 0x3FFF: Read dual I/O, Address &amp; Data 2 bits mode</li> <li>● 0x1FFF: Read dual O, just data 2 bits mode</li> <li>● 0x0FFF: 1 bit mode</li> </ul> <p>If the configured read mode is not supported, other modes would be searched until find the appropriate mode.</p>	0xFFFF
Flash_AVL	Maintains the Flash IC supported by SDK, refer to Fig 15-1 for detailed information.	
flash_init_userdef	Initializes the parameters in this function in order to adopt user-defined Flash on Ameba-D.	

### 15.2.1 Flash Classification

Ameba-D support Flash chips of multi-vendor, such Winbond, MXIC, Gigadevice, ESMT etc. The Flash chips which have been verified on Ameba platform can be found in *UM0400 Ameba Flash AVL.pdf*. We divide the supported Flash into 6 species as Table 15-3 shows according to their characteristics, and they can be used on Ameba-D directly.

Table 15-3 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 Ameba-D.

- (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 Fig 15-1 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,      0x0000043FC,      NULL}, /* Winbond: MANUFACTURER_ID_WINBOND */
    {0xA1,          0x000000FF,      FlashClass1,      0x00000FFF,      NULL}, /* Fudan Micro: MANUFACTURER_ID_FM */
    {0x0B,          0x000000FF,      FlashClass1,      0x0000043FC,      NULL}, /* XTX */
    {0x0E,          0x000000FF,      FlashClass1,      0x0000043FC,      NULL}, /* XTX (FT) */
    {0xC8,          0x000000FF,      FlashClass2,      0x0000043FC,      NULL}, /* GD normal: MANUFACTURER_ID_GD */
    {0x28C2,       0x00000FFF,      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},
};
    
```

Fig 15-1 Flash\_AVL

### 15.2.1.1 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 Fig 15-2 shows.
- There are 4 dummy cycles in Read Dual I/O mode, as Fig 15-3 shows.

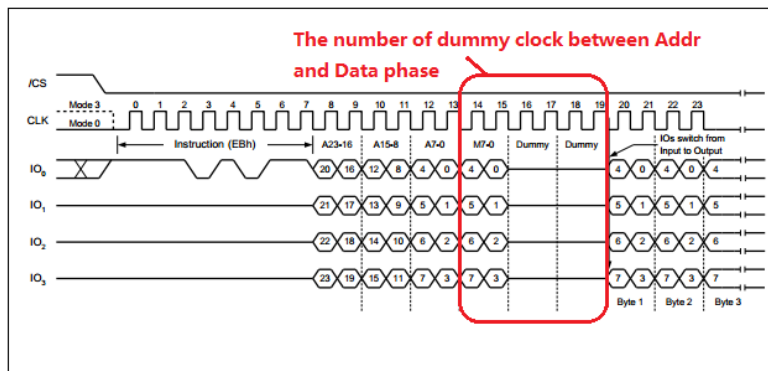


Fig 15-2 Fast read quad I/O instruction sequence

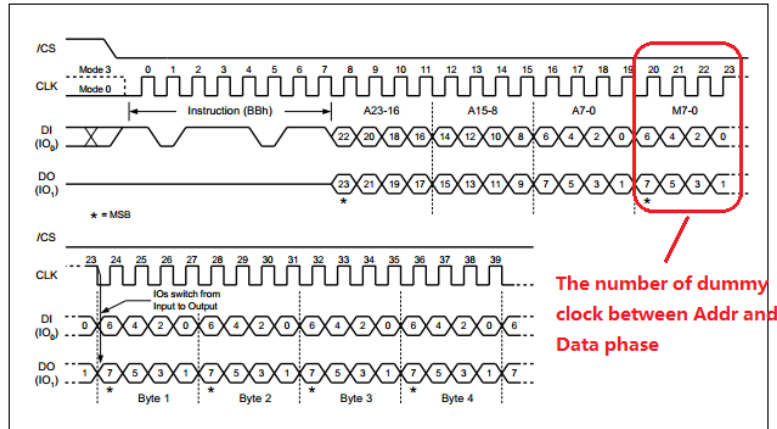


Fig 15-3 Fast read dual I/O instruction sequence

Users can find the dummy cycle of Read Quad Output and Read Dual Output mode in the same way.

### 15.2.1.2 Other Parameters

The parameters comparison of these species are listed in Table 15-4.

Table 15-4 Parameters comparison of different Flash species

Item	Parameter	FlashClass 1	FlashClass 2	FlashClass 3	FlashClass 4	FlashClass 5	FlashClass 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 <sup>1</sup>	-	-/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 <sup>2</sup>	-	-	-	-	-	0x15
	Write Configuration Register <sup>3</sup>	-	-	-	-	-	-

**Note 1:**

- For FlashClass1, the Write Status Register 2 is written together with the Write Status Register 1 using 0x01, as Fig 15-4 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 Fig 15-4 shows. When GD density is larger than 2MB, the Write Status Register 2 is written individually using 0x31, as Fig 15-5 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 Fig 15-6 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 Fig 15-7 shows.

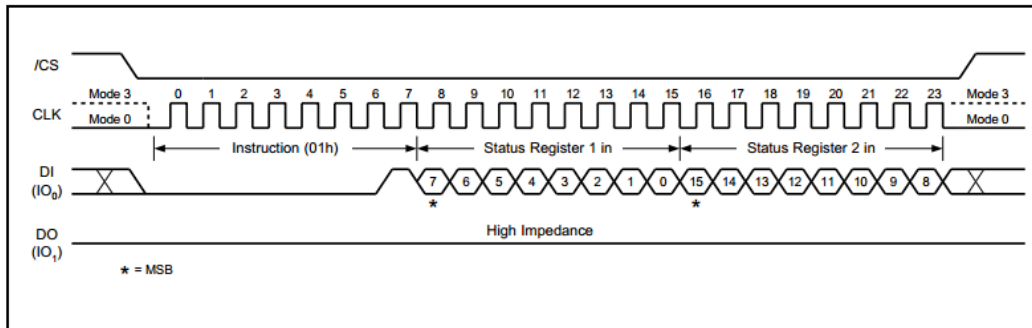


Fig 15-4 Write Status Register 1 & 2 sequence (FlashClass1, FlashClass2 when density < 2M)

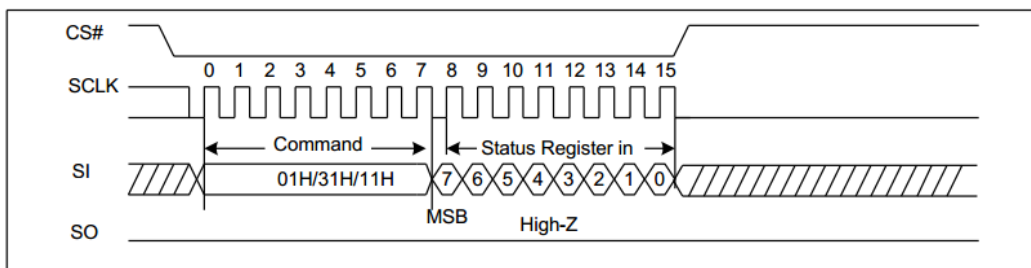


Fig 15-5 Write Status Register 1/2 sequence (FlashClass2 when density > 2MB)

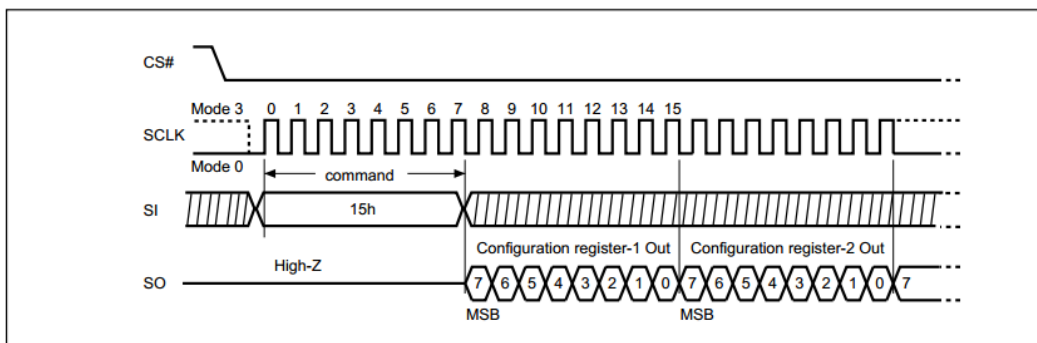


Fig 15-6 Read Configuration Register sequence (FlashClass6)



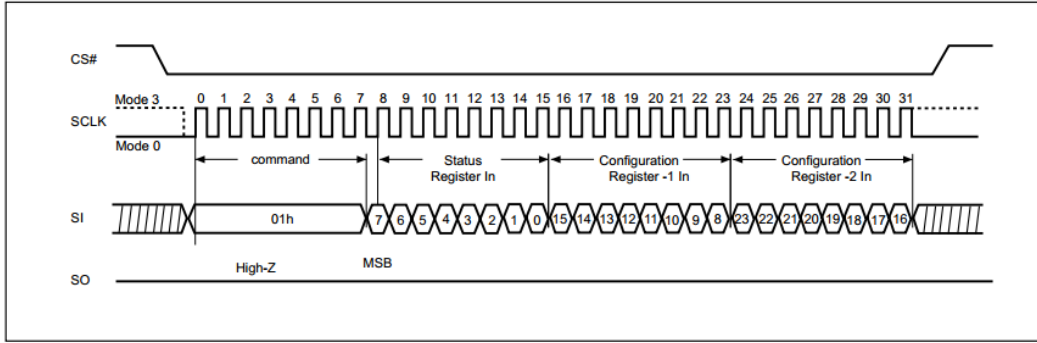


Fig 15-7 Write Status Register and Write Configuration Register sequence (FlashClass6)

## 15.2.2 FlashClass1 ~ FlashClass6

Table 15-5 lists the parameters for FlashClass1 to FlashClass6 which cannot be modified by users.

Table 15-5 Parameters of different Flash classification (FlashClass1 ~ FlashClass6)

Flash classification	Command list	Note
FlashClass1	<pre>FLASH_InitStruct-&gt;FLASH_Id = FLASH_ID_WINBOND;  /*1.1 status bit define */ FLASH_InitStruct-&gt;FLASH_QuadEn_bit = BIT(9); FLASH_InitStruct-&gt;FLASH_Busy_bit = BIT(0); FLASH_InitStruct-&gt;FLASH_WLE_bit = BIT(1); FLASH_InitStruct-&gt;FLASH_Status2_exist = 1;  /*1.2 dummy cycle */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[0] = 0; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[1] = 0x04; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[2] = 0x06;  /*1.3 set 2 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_dual_io = 0xBB; FLASH_InitStruct-&gt;FLASH_rd_dual_o = 0x3B;  /*1.4 set 4 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_quad_io = 0xEB; FLASH_InitStruct-&gt;FLASH_rd_quad_o = 0x6B;  /*1.5 other Flash command set */ FLASH_InitStruct-&gt;FLASH_cmd_wr_en = 0x06; FLASH_InitStruct-&gt;FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct-&gt;FLASH_cmd_rd_status = 0x05; FLASH_InitStruct-&gt;FLASH_cmd_rd_status2 = 0x35; //FLASH_CMD_RD_SR2 FLASH_InitStruct-&gt;FLASH_cmd_wr_status = 0x01; FLASH_InitStruct-&gt;FLASH_cmd_wr_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_chip_e = 0x60; FLASH_InitStruct-&gt;FLASH_cmd_block_e = 0xD8; FLASH_InitStruct-&gt;FLASH_cmd_sector_e = 0x20; FLASH_InitStruct-&gt;FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct-&gt;FLASH_cmd_pwdn = 0xB9;</pre>	
FlashClass2	<pre>FLASH_InitStruct-&gt;FLASH_Id = FLASH_ID_GD;  /*1.1 status bit define */</pre>	FLASH_cmd_wr_status2 would be set when density is larger than 2MB in SDK.

	<pre> FLASH_InitStruct-&gt;FLASH_QuadEn_bit = BIT(9); FLASH_InitStruct-&gt;FLASH_Busy_bit = BIT(0);    //WIP FLASH_InitStruct-&gt;FLASH_WLE_bit = BIT(1);    //WLE FLASH_InitStruct-&gt;FLASH_Status2_exist = 1;  /*1.2 dummy cycle */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[0] = 0; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[1] = 0x04;    //(M7-4)2 + 2 dummy FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[2] = 0x06;    //(M7-0)2 + 4 dummy  /*1.3 set 2 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_dual_io = 0xBB; FLASH_InitStruct-&gt;FLASH_rd_dual_o = 0x3B;  /*1.4 set 4 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_quad_io = 0xEB; FLASH_InitStruct-&gt;FLASH_rd_quad_o = 0x6B;  /*1.5 other Flash command set */ FLASH_InitStruct-&gt;FLASH_cmd_wr_en = 0x06; FLASH_InitStruct-&gt;FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct-&gt;FLASH_cmd_rd_status = 0x05; FLASH_InitStruct-&gt;FLASH_cmd_rd_status2 = 0x35; FLASH_InitStruct-&gt;FLASH_cmd_wr_status = 0x01; FLASH_InitStruct-&gt;FLASH_cmd_wr_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_chip_e = 0x60; FLASH_InitStruct-&gt;FLASH_cmd_block_e = 0xD8; FLASH_InitStruct-&gt;FLASH_cmd_sector_e = 0x20; FLASH_InitStruct-&gt;FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct-&gt;FLASH_cmd_pwdn = 0xB9; </pre>	
FlashClass3	<pre> FLASH_InitStruct-&gt;FLASH_Id = FLASH_ID_MXIC;  /*1.1 status bit define */ FLASH_InitStruct-&gt;FLASH_QuadEn_bit = BIT(6); FLASH_InitStruct-&gt;FLASH_Busy_bit = BIT(0);    //WIP FLASH_InitStruct-&gt;FLASH_WLE_bit = BIT(1);    //WLE FLASH_InitStruct-&gt;FLASH_Status2_exist = 0;  /*1.2 dummy cycle */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[0] = 0; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[1] = 0x04; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[2] = 0x06;  /*1.3 set 2 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_dual_io = 0xBB; FLASH_InitStruct-&gt;FLASH_rd_dual_o = 0x3B;  /*1.4 set 4 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_quad_io = 0xEB; FLASH_InitStruct-&gt;FLASH_rd_quad_o = 0x6B;  /*1.5 other Flash command set */ FLASH_InitStruct-&gt;FLASH_cmd_wr_en = 0x06; FLASH_InitStruct-&gt;FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct-&gt;FLASH_cmd_rd_status = 0x05; FLASH_InitStruct-&gt;FLASH_cmd_rd_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_wr_status = 0x01; FLASH_InitStruct-&gt;FLASH_cmd_wr_status2 = 0; </pre>	

	<pre>FLASH_InitStruct-&gt;FLASH_cmd_chip_e = 0x60; FLASH_InitStruct-&gt;FLASH_cmd_block_e = 0xD8; FLASH_InitStruct-&gt;FLASH_cmd_sector_e = 0x20; FLASH_InitStruct-&gt;FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct-&gt;FLASH_cmd_pwdn = 0xB9;</pre>	
FlashClass4	<pre>FLASH_InitStruct-&gt;FLASH_Id = FLASH_ID_MXIC;  /*1.1 status bit define */ FLASH_InitStruct-&gt;FLASH_QuadEn_bit = 0; FLASH_InitStruct-&gt;FLASH_Busy_bit = BIT(0);           //WIP FLASH_InitStruct-&gt;FLASH_WLE_bit = BIT(1);           //WLE FLASH_InitStruct-&gt;FLASH_Status2_exist = 0;  /*1.2 dummy cycle */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[0] = 0; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[1] = 0x04; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[2] = 0x06;  /*1.3 set 2 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_dual_io = 0xBB; FLASH_InitStruct-&gt;FLASH_rd_dual_o = 0x3B;  /*1.4 set 4 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_quad_io = 0xEB; FLASH_InitStruct-&gt;FLASH_rd_quad_o = 0x6B;  /*1.5 other Flash command set */ FLASH_InitStruct-&gt;FLASH_cmd_wr_en = 0x06; FLASH_InitStruct-&gt;FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct-&gt;FLASH_cmd_rd_status = 0x05; FLASH_InitStruct-&gt;FLASH_cmd_rd_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_wr_status = 0x01; FLASH_InitStruct-&gt;FLASH_cmd_wr_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_chip_e = 0x60; FLASH_InitStruct-&gt;FLASH_cmd_block_e = 0xD8; FLASH_InitStruct-&gt;FLASH_cmd_sector_e = 0x20; FLASH_InitStruct-&gt;FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct-&gt;FLASH_cmd_pwdn = 0xB9;</pre>	
FlashClass5	<pre>FLASH_InitStruct-&gt;FLASH_Id = FLASH_ID_MICRON;  /*1.1 status bit define */ FLASH_InitStruct-&gt;FLASH_QuadEn_bit = 0; FLASH_InitStruct-&gt;FLASH_Busy_bit = BIT(0); FLASH_InitStruct-&gt;FLASH_WLE_bit = BIT(1); FLASH_InitStruct-&gt;FLASH_Status2_exist = 0;  /*1.2 dummy cycle */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[0] = 0; /* set Micron rd_dummy_cyle based on baud rate, default 100MHz */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[1] = 0x05; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[2] = 0x09;  /*1.3 set 2 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_dual_io = 0xBB; FLASH_InitStruct-&gt;FLASH_rd_dual_o = 0x3B;  /*1.4 set 4 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_quad_io = 0xEB;</pre>	

	<pre>FLASH_InitStruct-&gt;FLASH_rd_quad_o = 0x6B;  /*1.5 other Flash command set */ FLASH_InitStruct-&gt;FLASH_cmd_wr_en = 0x06; FLASH_InitStruct-&gt;FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct-&gt;FLASH_cmd_rd_status = 0x05; FLASH_InitStruct-&gt;FLASH_cmd_rd_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_wr_status = 0x01; FLASH_InitStruct-&gt;FLASH_cmd_wr_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_chip_e = 0xC7; FLASH_InitStruct-&gt;FLASH_cmd_block_e = 0xD8; FLASH_InitStruct-&gt;FLASH_cmd_sector_e = 0x20; FLASH_InitStruct-&gt;FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct-&gt;FLASH_cmd_pwdn = 0xB9;</pre>	
FlashClass6	<pre>FLASH_InitStruct-&gt;FLASH_Id = FLASH_ID_MXIC;  /*1.1 status bit define */ FLASH_InitStruct-&gt;FLASH_QuadEn_bit = BIT(6); FLASH_InitStruct-&gt;FLASH_Busy_bit = BIT(0); //WIP FLASH_InitStruct-&gt;FLASH_WLE_bit = BIT(1); //WLE FLASH_InitStruct-&gt;FLASH_Status2_exist = 0;  /*1.2 dummy cycle */ FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[0] = 0; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[1] = 0x04; FLASH_InitStruct-&gt;FLASH_rd_dummy_cyle[2] = 0x06;  /*1.3 set 2 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_dual_io = 0xBB; FLASH_InitStruct-&gt;FLASH_rd_dual_o = 0x3B;  /*1.4 set 4 bits mode command */ FLASH_InitStruct-&gt;FLASH_rd_quad_io = 0xEB; FLASH_InitStruct-&gt;FLASH_rd_quad_o = 0x6B;  /*1.5 other Flash command set */ FLASH_InitStruct-&gt;FLASH_cmd_wr_en = 0x06; FLASH_InitStruct-&gt;FLASH_cmd_rd_id = 0x9F; FLASH_InitStruct-&gt;FLASH_cmd_rd_status = 0x05; FLASH_InitStruct-&gt;FLASH_cmd_rd_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_wr_status = 0x01; FLASH_InitStruct-&gt;FLASH_cmd_wr_status2 = 0; FLASH_InitStruct-&gt;FLASH_cmd_chip_e = 0x60; FLASH_InitStruct-&gt;FLASH_cmd_block_e = 0xD8; FLASH_InitStruct-&gt;FLASH_cmd_sector_e = 0x20; FLASH_InitStruct-&gt;FLASH_cmd_pwdn_release = 0xAB; FLASH_InitStruct-&gt;FLASH_cmd_pwdn = 0xB9;</pre>	

### 15.3 pinmapcfg.c

When entering sleep and deepsleep mode, I/O pull control is needed. Otherwise it results in power leakage. For example, UART voltage level is high. If we pull down UART pin or not pull, then power leakage happens. So we must make sure that each pin has proper pull control.

In SDK, you should set GPIO function pull control and sleep pull control in pmap\_func based on your PCB board, and SDK will set pull control based on your setting between suspend and resume.

```

const PMAP_TypeDef pmap_func[] =
{
// Pin Name      Func PU/PD      Slp PU/PD      DSlp PU/PD      LowPowerPin
{ _PA_0,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_1,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_2,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_3,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_4,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_5,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_6,        GPIO_PuPd_DOWN, GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, //
{ _PA_7,        GPIO_PuPd_KEE,  GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, // UART_LOG_TXD
{ _PA_8,        GPIO_PuPd_KEE,  GPIO_PuPd_KEE, GPIO_PuPd_KEE,  FALSE}, // UART_LOG_RXD
{ _PA_9,        GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_10,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_11,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_12,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_13,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_14,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_15,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_16,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_17,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_18,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_19,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_20,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_21,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_22,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_23,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_24,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_25,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_26,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  TRUE}, // keyscan
{ _PA_27,       GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // SWD_DATA or normal_mode_sel
{ _PA_28,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_29,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_30,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PA_31,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_0,        GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_1,        GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_2,        GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_3,        GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // SWD_CLK
{ _PB_4,        GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // Touch0
{ _PB_5,        GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // Touch1
{ _PB_6,        GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // Touch2
// { _PB_6,      GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_7,        GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // Touch3
{ _PB_8,        GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_9,        GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_10,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_11,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_12,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // SPI_DATA3
{ _PB_13,       GPIO_PuPd_UP,    GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  FALSE}, // SPI_CLK
{ _PB_14,       GPIO_PuPd_UP,    GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  FALSE}, // SPI_DATA0
{ _PB_15,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // SPI_DATA2
{ _PB_16,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // SPI_CS
{ _PB_17,       GPIO_PuPd_UP,    GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  FALSE}, // SPI_DATA1
{ _PB_18,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_19,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_20,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_21,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_22,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // IR pin should no-pull when you use IR
// { _PB_22,    GPIO_PuPd_NOPULL, GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_23,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_24,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_25,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_26,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // fix antenna diversity can not use issue
{ _PB_27,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_28,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_29,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_30,       GPIO_PuPd_UP,    GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PB_31,       GPIO_PuPd_DOWN,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, //
{ _PNC,        GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  GPIO_PuPd_KEE,  FALSE}, // table end
};
    
```

## 15.4 sleepcfg.c

This file should be kept the default settings, you can only change it with Realtek RD's help.

## 15.5 bootcfg.c

Bootloader can be configured through this file, as Table 15-6 and Fig 15-8 show.

Table 15-6 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

```

/*
 * @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_ConfDefFlash_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},
};

/*
 * @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[z] = {
    0x08006000, /* OTA1 region start address */
    0x08106000, /* OTA2 region start address */
};

/*
 * @brif 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_MaskDefRSIP_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},
};

/**
 * @brif 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;

/**
 * @brif boot log enable or disable.
 * FALSE: disable
 * TRUE: enable
 */
BOOT_RAM_DATA_SECTION
u8 Boot_Log_En = FALSE;

/**
 * @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.
 * @param None
 * @retval 1: Firmware verify successfully, 0: verify failed
 */
BOOT_RAM_DATA_SECTION
FuncPtr FwCheckCallback = NULL;

/**
 * @brif 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;

```

Fig 15-8 User bootcfg

## 15.6 ipccfg.c

You can add IPC entry referring to Fig 15-9, 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

```

Fig 15-9 User ipccfg

**i** 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.



## 15.7 lp\_intfcfg.c

UART low power Rx enable/disable can be configured through this file.

```

UARTCFG_TypeDef uart_config[2]=
{
    /* UART0 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE, /* Enable low power RX*/
    },
    /* UART1 */
    {
        .LOW_POWER_RX_ENABLE = DISABLE,
    },
};

```

## 15.8 hp\_intfcfg.c

PSRAM parameters, SDIO host parameters and FTL parameters can be configured through this 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
};

#ifdef CONFIG_FTL_ENABLED
#define FTL_MEM_CUSTEM 0
#else
#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."
#endif
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

```

# 16 Flash Operation

## 16.1 Functional Description

Ameba-D 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 SPI Flash through setting up SPIC registers.
- Auto mode: Compared to user mode, auto mode is a hardware control flow to execute. After initialize setting, user don't need to configure the related control register for each transfer operation. Auto mode is a convenient way to access SPI 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 16-1.

Table 16-1 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 can't be used at the same time.

## 16.2 Protection Method

Considering Ameba-D supports Flash executed 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 16-1 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 16.3 and 16.4 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.

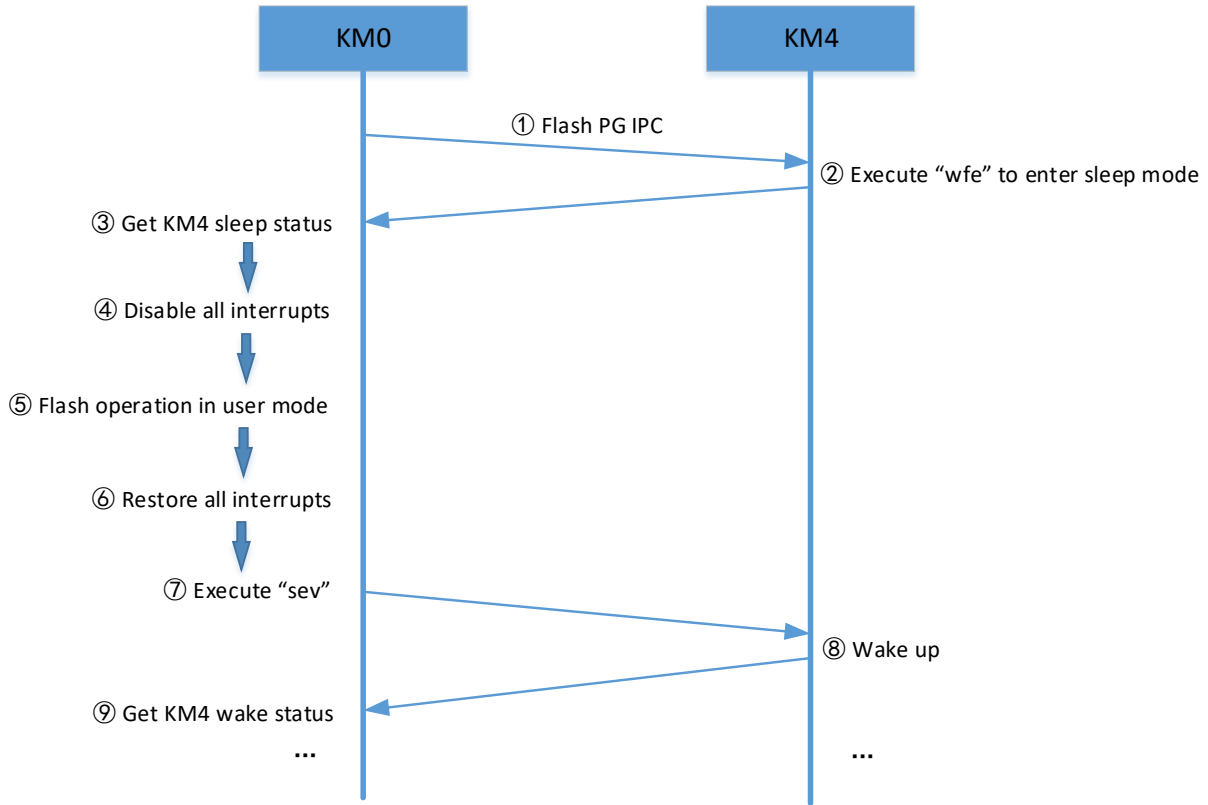


Fig 16-1 User mode operation flow

## 16.3 Flash Raw APIs

### 16.3.1 Read

#### 16.3.1.1 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	<ul style="list-style-type: none"> <li>● base: the base address of Flash memory. It should be SPI_FLASH_BASE which is 0x08000000.</li> <li>● 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.</li> </ul>
Return	The read data

#### 16.3.1.2 FLASH\_ReadStream

Items	Description
Introduction	Read a stream of data from specified offset address of Flash in auto mode.
Parameters	<ul style="list-style-type: none"> <li>● address: specify the starting offset address to read from. It has no alignment restrictions.</li> <li>● len: specify the length of the data to read.</li> <li>● data: specify the buffer to save the read-back data.</li> </ul>
Return	Status <ul style="list-style-type: none"> <li>● 1: success</li> <li>● Others: fail</li> </ul>

## 16.3.2 Write

### 16.3.2.1 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	<ul style="list-style-type: none"> <li>● StartAddr: start offset address in Flash from which SPIC writes.</li> <li>● DataPhaseLen: the number of bytes that SPIC sends in Data Phase. It should not be more than 12 bytes.</li> <li>● pData: pointer to a byte array that is to be sent.</li> </ul>
Return	N/A

### 16.3.2.2 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	<ul style="list-style-type: none"> <li>● address: specifies the starting offset to write to.</li> <li>● len: specifies the length of the data to write. It has no restrictions of less than 12 bytes.</li> <li>● data: pointer to a byte array that is to be written.</li> </ul>
Return	Status <ul style="list-style-type: none"> <li>● 1: success</li> <li>● Others: fail</li> </ul>

## 16.3.3 Erase

### 16.3.3.1 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	<ul style="list-style-type: none"> <li>● EraseType: can be one of the following value.                             <ul style="list-style-type: none"> <li>■ EraseChip: erase the whole chip.</li> <li>■ EraseBlock: erase the specified block (64KB).</li> <li>■ EraseSector: erase the specified sector (4KB).</li> </ul> </li> <li>● Address: address should be 4-byte aligned. The block/sector which the address in will be erased.</li> </ul>
Return	N/A

## 16.3.4 Receive/Transmit Command

### 16.3.4.1 FLASH\_RxCmdXIP

Items	Description
Introduction	Send Rx command to Flash to get status register or Flash ID in user mode. This function is already protected and can be called by user directly.
Parameters	<ul style="list-style-type: none"> <li>● cmd: command that need to be sent.</li> <li>● read_len: the number of bytes that will be read by SPIC after sending command.</li> <li>● read_data: pointer to a byte array which is used to save data received.</li> </ul>
Return	N/A

### 16.3.4.2 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	<ul style="list-style-type: none"> <li>● cmd: command to be sent.</li> <li>● len: the number of bytes to be sent after sending command.</li> <li>● status: pointer to byte array to be sent.</li> </ul>
Return	N/A

## 16.4 Flash Mbed APIs

The descriptions of Flash Mbed APIs can be found in *AN403 Peripheral Driver Mbed API.chm*. All these functions have already been protected and can be called directly.

## 16.5 User Configuration

Ameba-D provides `rtl8721dlp_flashcfg.c` to configure Flash speed and read mode.

```
/**
 * @brief 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 16-2 Configuring Flash speed

```
/**
 * @brief 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 0, Just data 4 bits mode
 *      0x3FFF: Read dual IO, Address & Data 2 bits mode
 *      0x1FFF: Read dual 0, 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 16-3 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.

# 17 Battery Measurement

## 17.1 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 17-1 shows. The VBAT channel measurable range is 0~5V.

Table 17-1 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.

## 17.2 Calibration

The relationship between input voltage and sample data is almost linear. Fig 17-1 shows the conversion of sample data and input voltage of VBAT\_MEAS.

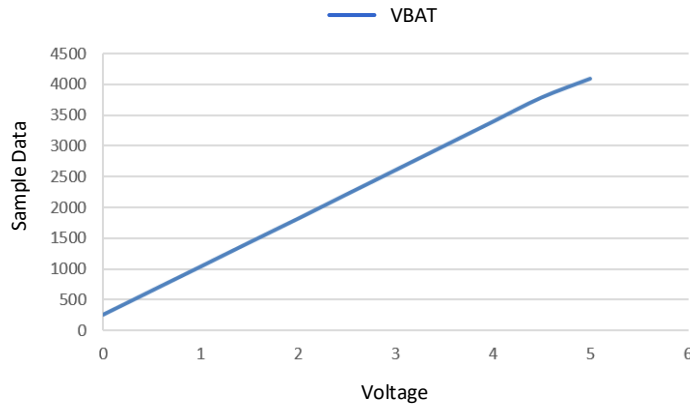


Fig 17-1 Relationship between sample data and input voltage

To obtain voltage from sample data, the following formula can be used:

$$\text{Voltage} = \frac{10 * \text{Data} - \text{OFFSET}}{\text{GAIN}} \text{ (V)}$$

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 precise 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 Linear 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,

$$\text{Voltage} = \frac{\text{Data} - \frac{\text{OFFSET}}{10}}{\frac{\text{GAIN}}{10}} = \frac{10 * \text{Data} - \text{OFFSET}}{\text{GAIN}} \text{ (V)}$$

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 17-2 shows.

Table 17-2 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.

# 18 Wi-Fi

## 18.1 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, used to start AP mode.
<rtw_network_info>	The structure is used to describe the station mode setting about SSID, security type and password, used 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, RSSI etc.

## 18.2 Wi-Fi APIs

### 18.2.1 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.

#### 18.2.1.1 wifi\_on

Enable Wi-Fi: bring the wireless interface “Up”.

Parameter	Type	Introduction
<mode>	rtw_mode_t	Decide to enable Wi-Fi in which mode. The optional modes are enumerated in rtw_mode_t.

#### 18.2.1.2 wifi\_off

Disable Wi-Fi.

Parameter: None.

#### 18.2.1.3 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_interface_t).

#### 18.2.1.4 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 <ul style="list-style-type: none"> <li>● RTW_STA_INTERFACE</li> <li>● RTW_AP_INTERFACE</li> </ul>

### 18.2.2 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

#### 18.2.2.1 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), Ad-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 specific 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 1 ~ 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.

#### 18.2.2.2 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. The callback, user\_data variables will be referenced after the function returns. Those variables must remain valid until the scan is completed. The usage of this API can refer to ATWS in atcmd\_wifi.c.

#### 18.2.2.3 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_bufllen>	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.

### 18.2.3 Connection APIs

API	Introduction
<wifi_connect>	Join a Wi-Fi network with specified SSID.
<wifi_connect_bssid>	Join a Wi-Fi network with specified BSSID.
<wifi_disconnect>	Disassociates from current Wi-Fi network.
<wifi_is_connected_to_ap>	Check if Wi-Fi has connected to AP before dhcp.
<wifi_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.

#### 18.2.3.1 wifi\_connect

Join a Wi-Fi network with specified SSID. Scan for, associate and authenticate with a Wi-Fi 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: <ul style="list-style-type: none"> <li>● RTW_SECURITY_OPEN: open security</li> <li>● RTW_SECURITY_WEP_PSK: WEP Security with open authentication</li> <li>● RTW_SECURITY_WEP_SHARED: WEP Security with shared authentication</li> <li>● RTW_SECURITY_WPA_TKIP_PSK: WPA Security</li> <li>● RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher</li> <li>● RTW_SECURITY_WPA2_TKIP_PSK: WPA2 Security using TKIP cipher</li> <li>● RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers</li> </ul>
<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 value.

**Note:** Make sure the Wi-Fi is enabled before invoking this function. (wifi\_on())

#### 18.2.3.2 wifi\_connect\_bssid

Join a Wi-Fi network with specified BSSID. Scan for, associate and authenticate with a Wi-Fi network. On successful return, the system is ready to send data packets.

Parameter	Type	Introduction
<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: <ul style="list-style-type: none"> <li>● RTW_SECURITY_OPEN: open security</li> <li>● RTW_SECURITY_WEP_PSK: WEP Security with open authentication</li> <li>● RTW_SECURITY_WEP_SHARED: WEP Security with shared authentication</li> <li>● RTW_SECURITY_WPA_TKIP_PSK: WPA Security</li> <li>● RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher</li> <li>● RTW_SECURITY_WPA2_TKIP_PSK: WPA2 Security using TKIP cipher</li> <li>● RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers</li> </ul>
<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.
<ley_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 value.

**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().

**18.2.3.3 wifi\_disconnect**

Disassociates from current Wi-Fi network.

Parameter: None.

**18.2.3.4 wifi\_is\_connected\_to\_ap**

Check if Wi-Fi has connected to AP before dhcp.

Parameter: None.

**18.2.3.5 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.

**18.2.3.6 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 enable/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.

### 18.2.3.7 wifi\_get\_autoreconnect

Get the result of setting reconnection mode.

Parameter	Type	Introduction
<mode>	_u8	Pointer to the result of setting reconnection mode.

### 18.2.3.8 wifi\_get\_last\_error

Present the device disconnect reason while connecting.

Parameter: None.

## 18.2.4 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.

### 18.2.4.1 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 Wi-Fi driver, since it will be determined by the channel from received beacon.

### 18.2.4.2 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.

### 18.2.4.3 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 <b>wifi_constants.h</b>

### 18.2.4.4 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 <b>wifi_constants.h</b>

## 18.2.5 Power APIs

API	Introduction
<wifi_enable_powersave>	Enable Wi-Fi power save mode.
<wifi_disable_powersave>	Disable Wi-Fi power save mode.
<wifi_get_txpower>	Get the Tx 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.

### 18.2.5.1 wifi\_enable\_powersave

Enable Wi-Fi power save mode. When 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.

### 18.2.5.2 wifi\_disable\_powersave

Disable Wi-Fi power save mode.

Parameter: None.

### 18.2.5.3 wifi\_get\_txpower

Get the Tx power in index units.

Parameter	Type	Introduction
<poweridx>	int *	The variable to receive the Tx power in index.

### 18.2.5.4 wifi\_set\_txpower

Set the Tx power in index units.

Parameter	Type	Introduction
<poweridx>	int	The desired Tx power in index.

### 18.2.5.5 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. Wi-Fi 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. Wi-Fi 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.

### 18.2.5.6 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 packages 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.

### 18.2.5.7 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 packages buffered at AP side.

## 18.2.6 AP Mode APIs

API	Introduction
<wifi_start_ap>	Trigger Wi-Fi driver to start an infrastructure Wi-Fi network.
<wifi_start_ap_with_hidden_ssid>	Start an infrastructure Wi-Fi 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.

### 18.2.6.1 wifi\_start\_ap

Trigger Wi-Fi driver to start an infrastructure Wi-Fi network.

Parameter	Type	Introduction
<ssid>	char *	A null terminated string containing the SSID name of the network.
<security_type>	security_type	<ul style="list-style-type: none"> <li>● RTW_SECURITY_OPEN: Open Security</li> <li>● RTW_SECURITY_WPA_TKIP_PSK: WPA Security</li> <li>● RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher</li> <li>● RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers</li> </ul> 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. It will not use the channel provided.

### 18.2.6.2 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	<ul style="list-style-type: none"> <li>● RTW_SECURITY_OPEN: Open Security</li> <li>● RTW_SECURITY_WPA_TKIP_PSK: WPA Security</li> <li>● RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher</li> <li>● RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers</li> </ul> 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. It will not use the channel provided.

### 18.2.6.3 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	<ul style="list-style-type: none"> <li>● RTW_SECURITY_OPEN: Open Security</li> <li>● RTW_SECURITY_WPA_TKIP_PSK: WPA Security</li> <li>● RTW_SECURITY_WPA2_AES_PSK: WPA2 Security using AES cipher</li> <li>● RTW_SECURITY_WPA2_MIXED_PSK: WPA2 Security using AES and/or TKIP ciphers</li> </ul> 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. It will not use the channel provided.

### 18.2.6.4 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.

### 18.2.6.5 wifi\_enable\_forwarding

Enable packets forwarding in AP mode.

Parameter: None.

### 18.2.6.6 wifi\_disable\_forwarding

Disable packets forwarding in AP mode.

Parameter: None.

### 18.2.7 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.

#### 18.2.7.1 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:** This API can't be executed twice before deleting the previous custom IE list.

#### 18.2.7.2 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.

#### 18.2.7.3 wifi\_del\_custom\_ie

Delete Wi-Fi custom IE list.

Parameter: None.

### 18.2.8 Wi-Fi Setting APIs

API	Introduction
<wifi_get_mac_address>	Retrieves the current Media Access Control (MAC) address (or Ethernet hardware address) of the 802.11 device.
<wifi_get_ap_bssid>	Get connected AP's BSSID.
<wifi_get_ap_info>	Get the SoftAP information.
<wifi_set_code_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.

### 18.2.8.1 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.

### 18.2.8.2 wifi\_get\_ap\_bssid

Get connected AP's BSSID.

Parameter	Type	Introduction
<bssid>	unsigned char *	The location where the AP BSSID will be stored.

### 18.2.8.3 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.

### 18.2.8.4 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.

### 18.2.8.5 wifi\_get\_sta\_max\_data\_rate

Retrieved STA mode max. data rate.

Parameter	Type	Introduction
<inidata_rate>	_u8 *	Max data rate.

### 18.2.8.6 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.

### 18.2.8.7 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.

### 18.2.8.8 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.

### 18.2.8.9 wifi\_set\_mib

Setup the adaptive mode. You can replace this weak function by the same name function to setup adaptive mode you can.

Parameter: None.

### 18.2.8.10 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.

### 18.2.8.11 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 should be larger or equal to 50ms. It means 2 slot period is slot_period, 100-slot_period If we separate TBTT into 3 slots, then slot_period should be less or equal to 33ms. It means 3 slot period is 100 - 2 * slot_period, slot_period, slot_period.
<rfof_period_len_1>	unsigned char	RF on period of slot 1.
<rfof_period_len_2>	unsigned char	RF on period of slot 2.
<rfof_period_len_3>	unsigned char	RF on period of slot 3.

### 18.2.8.12 wifi\_get\_setting

Get current Wi-Fi setting from driver.

Parameter	Type	Introduction
<ifname>	const char *	The WLAN interface name

		<ul style="list-style-type: none"> <li>● WLAN0_NAME</li> <li>● WLAN1_NAME</li> </ul>
<pSetting>	rtw_wifi_setting_t *	Points to the rtw_wifi_setting_t structure to store the Wi-Fi setting gotten from driver.

### 18.2.8.13 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().

### 18.2.8.14 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. <ul style="list-style-type: none"> <li>● RTW_NETWORK_B</li> <li>● RTW_NETWORK_BG</li> <li>● RTW_NETWORK_BGN</li> </ul>

### 18.2.8.15 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.

### 18.2.8.16 wifi\_get\_antenna\_info

Get antenna information.

Parameter	Type	Introduction
<antenna>	unsigned char *	Points to store the antenna value gotten from driver. <ul style="list-style-type: none"> <li>● 0: main</li> <li>● 1: aux</li> </ul>

### 18.2.8.17 wifi\_set\_ch\_deauth

Set flag for concurrent mode wlan1 issue\_deauth when channel switched by wlan0.

Parameter	Type	Introduction
<enable>	_u8	<ul style="list-style-type: none"> <li>● 0: Disable</li> <li>● 1: Enable</li> </ul>

Usage: wifi\_set\_ch\_deauth(0) -> wlan0 wifi\_connect -> wifi\_set\_ch\_deauth(1)

## 18.2.9 Wi-Fi Indication APIs

API	Introduction
<wifi_manager_init>	Initialize Realtek Wi-Fi 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.

### 18.2.9.1 wifi\_manager\_init

Initialize Realtek Wi-Fi 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.

### 18.2.9.2 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 current stack size usage (by using uxTaskGetStackHighWaterMark()), and tries not to share the same stack with WLAN driver if remaining stack space is not available for the following operations.
- Using semaphore to notice another thread instead of handing event directly in **wifi\_indication()**.

### 18.2.9.3 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.

### 18.2.9.4 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.

### 18.2.9.5 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.

## 18.2.10 eFuse Writing APIs

API	Introduction
<wifi_set_mac_address>	This function sets the current Media Access Control (MAC) address of the 802.11 device.

**Note:** eFuse writing related API can only be called when the voltage is 3.3V. eFuse writing with the voltage 1.8V may fail. What's more, eFuse writing can only be operated one time, so be careful to do eFuse writing.

### 18.2.10.1 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.

## 18.3 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.

### 18.3.1 Implement

#### 18.3.1.1 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 pre-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**.

#### 18.3.1.2 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 Wi-Fi initializing. After Wi-Fi initialized, if **p\_wlan\_init\_done\_callback** points to a valid address, this function will be called.

#### 18.3.1.3 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.

### 18.3.2 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 in Flash and execute connection.

#### 18.3.2.1 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: <ul style="list-style-type: none"> <li>● 0: write is ok</li> <li>● -1: write is failed</li> </ul>

**Note:** This is only a demo API, user should define his own API.

## 18.4 WPS APIs

API	Introduction
<wps_start>	Start WPS enrollee process
<wps_stop>	Stop WPS enrollee process

### 18.4.1 wps\_start

Start WPS enrollee process.

Parameter	Type	Introduction
<wps_config>	u16	WPS configure method: <ul style="list-style-type: none"> <li>● WPS_CONFIG_DISPLAY</li> <li>● WPS_CONFIG_KEYPAD</li> <li>● WPS_CONFIG_PUSHBUTTON</li> </ul>
<pin>	char *	PIN number. Can be set to NULL if using WPS_CONFIG_PUSHBUTTON.
<channel>	u8	Channel. Currently un-used, can be set to 0.
<ssid>	char *	Target network SSID. Can be set to NULL if no target network specified.

**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.

### 18.4.2 wps\_stop

Stop WPS enrollee process.

Parameter: None.

**Note:** Make sure CONFIG\_ENABLE\_WPS is enabled in **platform\_opts.h**.

# 19 Liquid Crystal Display Controller (LCDC)

Ameba-D LCDC supports Thin Film Transistor (TFT) color display. It provides I8080 MCU interface (8-bit bus width) and RGB interface (6-bit bus width), and supports RGB565 data format

This chapter introduces how to use LCDC I/F to control LCD module.

## 19.1 Interface

The available LCDC interfaces for different IC packages are shown in Table 19-1.

Table 19-1 LCDC I/F supported for different packages

IC Package	MCU 8-bit	RGB 6-bit
RTL8722D/RTL8722CS	✓	✓
RTL8721D/RTL8721CS/RTL8720D/RTL8720CS	✗	✗

The data format supported by each interface is shown in Table 19-2. In order to display normally, the data format of LCM and LCDC interface should match.

Table 19-2 LCDC I/F data format

Interface	Data Format							Comment
MCU 8-bit	Count	0	1	2	3	4	...	2 transfers/pixel, RGB565
	RS	0	1	1	1	1	...	
	D7	C7	P1R4	P1G2	P2R4	P2G2	...	
	D6	C6	P1R3	P1G1	P2R3	P2G1	...	
	D5	C5	P1R2	P1G0	P2R2	P2G0	...	
	D4	C4	P1R1	P1B4	P2R1	P2B4	...	
	D3	C3	P1R0	P1B3	P2R0	P2B3	...	
	D2	C2	P1G5	P1B2	P2G5	P2B2	...	
	D1	C1	P1G4	P1B1	P2G4	P2B1	...	
	D0	C0	P1G3	P1B0	P2G3	P2B0	...	
P1R4 : Pixel1/Red_bit4								
RGB 6-bit	Count	0	1	2	3	4	...	3 transfers/pixel, RGB565
	D5	P1R4	P1G5	P1B4	P2R4	P2G5	...	
	D4	P1R3	P1G4	P1B3	P2R3	P2G4	...	
	D3	P1R2	P1G3	P1B2	P2R2	P2G3	...	
	D2	P1R1	P1G2	P1B1	P2R1	P2G2	...	
	D1	P1R0	P1G1	P1B0	P2R0	P2G1	...	
	D0		P1G0			P2G0	...	
P1R4 : Pixel1/Red_bit4								

## 19.2 Resolution

The maximum resolution supported by each interface is listed in Table 19-3.

Table 19-3 Maximum supported resolution

Interface	Display Type	Max. Support Resolution
MCU 8-bit	Static Display	1024*1024
	Dynamic Display	645*645 (30fps)
RGB 6-bit	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 19-1, 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 6-bit I/F mode:

$$image\_size = MAX\_DOT\_CLK / (3 * F) - (width + HBP + HFP) * (VBP + VFP) - height * (HBP + HFP);$$

$$\rightarrow F = 50M / (width * height + (width + HBP + HFP) * (VBP + VFP) + height * (HBP + HFP)) / 3;$$

When frame rate is lower than 30fps, the screen flickering may happen. Users should evaluate the visual artifacts when setting the frame rate lower.

## 19.3 Pinmux

The pin assignments of LCDC are listed in Table 19-4.

Table 19-4 LCDC pin assignments

Port Name	MCU 8-bit	RGB 6-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

## 19.4 LCDC APIs

### 19.4.1 MCU Function

#### 19.4.1.1 LCDC\_MCUStructInit

Items	Description
Introduction	Initializes the parameters in the LCDC_MCUInitStruct with default values.
Parameters	LCDC_MCUInitStruct: pointer to an 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 8-bit, IO mode or DMA mode, the plane size and some timing control. The parameters in this structure should be set according to LCD module datasheet.

#### 19.4.1.2 LCDC\_MCUInit

Items	Description
Introduction	Initializes the LCDC peripheral according to the specified parameters in LCDC_MCUInitStruct.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● LCDC_MCUInitStruct: pointer to a LCDC_MCUInitTypeDef structure that contains the configuration information.</li> </ul>
Return	N/A

#### 19.4.1.3 LCDC\_MCUWriteCmd

Items	Description
Introduction	Writes command to LCD module via MCU I/F.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● Cmd: the command to transmit.</li> </ul>
Return	N/A

#### 19.4.1.4 LCDC\_MCUWriteData

Items	Description
Introduction	Writes data to LCD module via MCU I/F.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● Data: the data to transmit.</li> </ul>
Return	N/A

#### 19.4.1.5 LCDC\_MCUReadData

Items	Description
Introduction	Reads data from LCD module via MCU I/F.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	The read value

#### 19.4.1.6 LCDC\_MCU DMA Trigger

Items	Description
Introduction	Triggers to transfer data of one frame from DMA buffer to LCDC Transmit FIFO.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	N/A

### 19.4.2 RGB Function

#### 19.4.2.1 LCDC\_RGBStructInit

Items	Description
Introduction	Initializes the parameters in the LCDC_RGBInitStruct with default values.
Parameters	LCDC_RGBInitStruct: pointer to an LCDC_RGBInitStructTypeDef structure which is initialized.
Return	N/A

**Note:** LCDC\_RGBInitStruct contains the RGB configurable parameters for LCDC, which determine that the RGB I/F 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.

### 19.4.2.2 LCDC\_RGBInit

Items	Description
Introduction	Initializes the LCDC peripheral according to the specified parameters in the LCDC_RGBInitStruct.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● LCDC_RGBInitStruct: pointer to a LCDC_RGBInitTypeDef structure that contains the configuration information.</li> </ul>
Return	N/A

## 19.4.3 LED Function

### 19.4.3.1 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

### 19.4.3.2 LCDC\_LEDInit

Items	Description
Introduction	Initializes the LCDC peripheral according to the specified parameters in the LCDC_LEDInitStruct.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● LCDC_LEDInitStruct: pointer to an LCDC_LEDInitTypeDef structure which will be initialized.</li> </ul>
Return	N/A

## 19.4.4 Common Function

### 19.4.4.1 LCDC\_DMAModeConfig

Items	Description
Introduction	Configures LCDC DMA burst size.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● BurstSize: DMA burst size; unit is 64 bytes.</li> </ul>
Return	N/A

**Note:**

- If BurstSize = 1, the actual burstsize = 1x64 bytes; if BurstSize = 2, the actual burstsize = 2x64 = 128 bytes; ...
- The parameter BurstSize is not more than 8. The recommended value is 2.

### 19.4.4.2 LCDC\_DMAImageBaseAddrConfig

Items	Description
Introduction	Configures image base address.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● ImgBaseAddr: the buffer address.</li> </ul>
Return	N/A

### 19.4.4.3 LCDC\_INTConfig

Items	Description
Introduction	Enables or disables the specified LCDC interrupts.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● LCDC_IT: specifies the LCDC interrupt sources to be enabled or disabled. This parameter can be any combination of the following values:                             <ul style="list-style-type: none"> <li>■ LCDC_IT_DMAUNDFW: DMA FIFO underflow interrupt</li> <li>■ LCDC_IT_FRDN: LCD refresh done interrupt</li> <li>■ LCDC_IT_LINE: line interrupt</li> <li>■ LCDC_IT_IO_TIMEOUT: I/O write/read timeout interrupt</li> <li>■ LCDC_IT_FRM_START: Frame Start interrupt</li> </ul> </li> <li>● NewState: new state of the specified LCDC interrupts. This parameter can be ENABLE or DISABLE.</li> </ul>
Return	N/A

### 19.4.4.4 LCDC\_GetINTStatus

Items	Description
Introduction	Gets LCDC interrupt status.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	Interrupt status

### 19.4.4.5 LCDC\_ClearINT

Items	Description
Introduction	Clears the LCDC's interrupt pending bits.
Parameters	<ul style="list-style-type: none"> <li>● LCDC_IT: specifies the interrupt to be cleared. This parameter can be any combination of the following values:                             <ul style="list-style-type: none"> <li>■ LCDC_IT_LINE: line interrupt</li> <li>■ LCDC_IT_FRDN: refresh frame done interrupt</li> <li>■ LCDC_IT_DMAUNDFW: DMA FIFO under flow interrupt</li> <li>■ LCDC_IT_IO_TIMEOUT: IO write/read timeout interrupt</li> <li>■ LCDC_IT_FRM_START: Frame Start interrupt</li> </ul> </li> </ul>
Return	N/A

### 19.4.4.6 LCDC\_Cmd

Items	Description
Introduction	Enables or disables the LCDC.
Parameters	<ul style="list-style-type: none"> <li>● LCDCx: where LCDCx can be LCDC.</li> <li>● NewState: new state of the LCDC. This parameter can be: ENABLE or DISABLE.</li> </ul>
Return	N/A

**Note:** When NewState is DISABLE, during the period of valid line (VTIMING =valid data), the disabled action 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 disabled action is performed instantly.

### 19.4.4.7 LCDC\_InsDisable

Items	Description
Introduction	Disables the LCDC instantly.
Parameters	LCDCx: where LCDCx can be LCDC.
Return	N/A

### 19.4.4.8 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.

## 19.5 How to Use LCDC

Table 19-5 lists the typical application scenarios of LCDC.

Table 19-5 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

### 19.5.1 MCU Interface

#### 19.5.1.1 I/O Mode

To use the LCDC MCU interface I/O mode, the following steps are mandatory.

(1) Configure the LCDC pinmux according to Table 19-4.

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_MCUInitStructInit (LCDC_MCUInitStructTypeDef * LCDC_MCUInitStruct);
```

b) Change other parameters according to LCM datasheet, such as 8-bit 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_MCUInitStructTypeDef * LCDC_MCUInitStruct);
```

(4) Enable the LCDC using the function LCDC\_Cmd().

(5) Send commands and parameters to LCM using the function LCDC\_MCUWriteCmd()/LCDC\_MCUWriteData() to initialize the LCD module.

(6) After LCM is initialized, call LCDC\_MCUWriteCmd()/LCDC\_MCUWriteData()/LCDC\_MCUReadData() to send commands/write data to LCM or read data from LCM to drive LCD displaying.

#### 19.5.1.2 Trigger DMA Mode

To use the LCDC MCU interface trigger DMA mode, the following steps are mandatory.

(1) Configure the LCDC pinmux according to Table 19-4.

(2) Configure LCDC to work in MCU I/O mode, then send commands and parameters to LCM to initialize LCD. After that, users need to send a command to inform LCM to start receiving data before transferring frame data.

(3) Initialize the LCDC\_MCUInitStruct variable.

a) Configure the LCDC\_MCUInitStruct parameter to Trigger DMA mode.

```
LCDC_MCUInitStructInit (LCDC_MCUInitStructTypeDef * LCDC_MCUInitStruct);
```

```
LCDC_MCUInitStruct->LCDC_MCU_Mode = LCDC_MCU_DMA_MODE;
```

```
LCDC_MCUInitStruct->LCDC_MCU_DMA_Mode = LCDC_TRIGGER_DMA_MODE;
```

b) Change other parameters according to LCM datasheet, such as LCD width/height, 8-bit I/F 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.

### 19.5.1.3 VSYNC Mode

To use the LCDC MCU I/F VSYNC mode, the following steps are mandatory.

- (1) Configure the LCDC pinmux according to Table 19-4.
- (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.

### 19.5.1.4 TE Mode

To use the LCDC MCU Iinterface TE mode, the following steps are mandatory.

- (1) Configure the LCDC pinmux according to Table 19-4.
- (2) Configure LCDC to work in MCU I/O mode, then send commands and parameters to LCM to let LCD work in 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 the TE signal from LCM, and you can update the frame buffer to change the display.

## 19.5.2 RGB Interface

### 19.5.2.1 DE Mode

To use the LCDC RGB interface DE mode, the following steps are mandatory.

- (1) Configure the LCDC pinmux according to Table 19-4.
- (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 to change the display.

### 19.5.2.2 HV Mode

To use the LCDC RGB I/F HV mode, the following steps are mandatory.

- (1) Configure the LCDC pinmux according to Table 19-4.
- (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, HSW, refresh frequency, LCD width/height, 6-bit 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 to change the display.

## 19.5.3 LED Interface

To use the LCDC LED interface mode, the following steps are mandatory.

- (1) Configure the LCDC pinmux according to Table 19-4.
- (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 active 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 frame buffer to change the display.

## 19.6 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 emWin LICENSED SOFTWARE, which is located in component\common\ui\emwin, is available in object code form for Realtek customers who have been authorized.

### 19.6.1 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 accept and agree this Agreement, then you can unzip “emwin.zip” with the password, which can be found in *emWin\_Software\_License\_Agreement.pdf*.

### 19.6.2 emWin Software

After the “emwin.zip” is extracted, you can see the following directories of emWin software.

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 <b>GUIConf.c</b> and <b>LCDCConf.c</b> according to their devices. The details of these two files can be found in <i>Chapter 38 Configuration, UM03001_emWin5.pdf</i> . For Ameba-D, we provide porting samples of EVB LCM with MCU or RGB I/F. The samples are located in <b>component\common\ui\emwin\Sample\rtl8721d\Config</b> . It’s much easier for users to modify these files to adapt to new LCM. The files in this sub-directory are used to adapter for different LCD module and different scenarios.
	Doc	<i>UM03001_emWin5.pdf</i> is the User Guide and Reference Manual of emWin.
	GUI_X	<b>GUI_X_Freertos.c</b> 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 Ameba-D driving EVB LCM. <ul style="list-style-type: none"> <li>● MCU I/F LCM on EVB: ILI9488 TFT-LCD module (320*480, MCU 8-bit I/F)</li> <li>● RBG I/F LCM on EVB: TCX043DTLN-04 TFT-LCD module (480*272, RGB 6-bit I/F)</li> </ul>
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.

### 19.6.3 How to Adapt to LCM

As mentioned above, we provide porting samples of Ameba-D driving EVB LCM in `component\common\ui\emwin\Sample\rtl8721d`. Table 19-6 demonstrate how to customize configuration files to adapt a new LCM.

Table 19-6 Customizing configuration files

Configuration file	Step	Description	Note
GUIConf.c	<ol style="list-style-type: none"> <li>(1) Modify the value of macro <b>GUI_NUMBYTES</b> to define the size of memory block.</li> <li>(2) Define macro <b>PSRAM_BUF_USED</b> in LCDConf.h to place the memory block into PSRAM. Otherwise, undefine it to place the memory block into SRAM.</li> </ol>	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	<ol style="list-style-type: none"> <li>(1) Set <b>XSIZE_PHYS</b> and <b>YSIZE_PHYS</b> to the horizontal and vertical resolution of LCM.</li> <li>(2) Implement the functions illustrated in Table 19-7. <ul style="list-style-type: none"> <li>■ void lcd_mcu_write (int x0, int y0, int x1, int y1, void *buf, U32 color)</li> <li>■ void lcd_mcu_read (int x0, int y0, int x1, int y1, void *buf)</li> <li>■ void lcd_init (void)</li> </ul> </li> </ol>	Used for MCU I/F LCM. As an example, these functions are implemented in <code>Sample\rtl8721d\hal\hal_mcu_lcd.c</code> . 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	<ol style="list-style-type: none"> <li>(1) Set <b>XSIZE_PHYS</b> and <b>YSIZE_PHYS</b> to the horizontal and vertical resolution of LCM.</li> <li>(2) Define macro <b>PSRAM_BUF_USED</b> in LCDConf.h to place the VRAM buffer into PSRAM. Otherwise, undefine it to place the VRAM buffer into SRAM.</li> <li>(3) Implement the functions illustrated in Table 19-8. <ul style="list-style-type: none"> <li>■ void lcd_init (unsigned int width, unsigned int height, unsigned int bufAddr)</li> <li>■ void lcd_set_dma_addr (unsigned int bufAddr)</li> </ul> </li> </ol>	Used for RGB I/F LCM. As an example, these functions are implemented in <code>Sample\rtl8721d\hal\hal_rgb_lcd.c</code> . 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	<ol style="list-style-type: none"> <li>(1) Set <b>XSIZE_PHYS</b> and <b>YSIZE_PHYS</b> to the horizontal and vertical resolution of LCM.</li> <li>(2) Implement the functions illustrated in Table 19-9. <ul style="list-style-type: none"> <li>■ void lcd_spi_write (int x0, int y0, int x1, int y1, void *buf, U32 color)</li> <li>■ void lcd_spi_read (int x0, int y0, int x1, int y1, void *buf)</li> <li>■ void lcd_init (void)</li> </ul> </li> </ol>	Used for SPI I/F LCM. As an example, these functions are implemented in <code>Sample\rtl8721d\hal\hal_spi_lcd.c</code> . You should modify this file according to the flow of Initialize, Read/Write pixel specified by datasheet of LCM.	<ul style="list-style-type: none"> <li>● lcd_spi_read function is not necessary.</li> <li>● 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.</li> </ul>

Table 19-7 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	<ul style="list-style-type: none"> <li>● x0: the start column of area to be updated</li> <li>● y0: the start row of area to be updated</li> <li>● x1: the end column of area to be updated</li> <li>● y1: the end row of area to be updated</li> <li>● buf: pointer to a buffer which is used to store pixel data of the area defined by (x0,y0,x1,y1)</li> </ul>	<ul style="list-style-type: none"> <li>● x0: the start column of area to be read</li> <li>● y0: the start row of area to be read</li> <li>● x1: the end column of area to be read</li> <li>● y1: the end row of area to be read</li> <li>● buf: pointer to a buffer which is used to store pixel data read from the area defined by (x0,y0,x1,y1)</li> </ul>	N/A



	<ul style="list-style-type: none"> <li>color: If buf is NULL, fill the area with the same color</li> </ul>		
Return	N/A	N/A	N/A

Table 19-8 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 Buffering or Virtual Screen function is used.
Parameters	<ul style="list-style-type: none"> <li>width: horizontal resolution</li> <li>height: vertical resolution</li> <li>bufAddr: the start address of VRAM</li> </ul>	bufAddr: the start address of VRAM
Return	N/A	N/A

Table 19-9 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	<ul style="list-style-type: none"> <li>x0: the start column of area to be updated</li> <li>y0: the start row of area to be updated</li> <li>x1: the end column of area to be updated</li> <li>y1: the end row of area to be updated</li> <li>buf: pointer to a buffer which is used to store pixel data of the area defined by (x0,y0,x1,y1)</li> <li>color: If buf is NULL, fill the area with the same color</li> </ul>	<ul style="list-style-type: none"> <li>x0: the start column of area to be read</li> <li>y0: the start row of area to be read</li> <li>x1: the end column of area to be read</li> <li>y1: the end row of area to be read</li> <li>buf: pointer to a buffer which is used to store pixel data read from the area defined by (x0,y0,x1,y1)</li> </ul>	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

### 19.6.4 How to Adapt to Touch Panel

We provide porting samples of Ameba-D 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.

- 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.
- Modify `hal_touch.h/hal_touch.c`
  - Define the macro "TP\_ID" which means the touch panel you want to use in `hal_touch.h`
  - 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 point
Parameters	N/A	N/A	N/A
Return	N/A	X coordinate of touch point	Y coordinate of touch point

## 19.6.5 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.

# 20 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

## 20.1 Throughput

PSRAM supports direct access and DMA access.

Table 20-1 PSRAM throughput

Access mode		Write		Read	
		Theory	Test	Theory	Test
Direct Access	Cache off	200Mbps	< 160Mbps	177.78Mbps	$< (32)/(180ns+360ns) = 59.26 \text{ Mbps}$
	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 accesses 32 bits once with header and delay.
  - Instruction execution time also needs 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 giving 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 giving 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 20-2 PSRAM throughput theoretical calculation

Item	Writing 32 bits	Reading 32 bits
Header + delay	$[3 + (3-1)] * 20ns = 100ns$	$[3 + (3-1)] * 20ns = 100ns$
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	$32/160ns = 200Mbps$	$(32*8)/460ns = 556.52Mbps$

## 20.2 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 20-3 PSRAM power management

Retention	Options	Power Consumption (uA)	Comment
No	<ul style="list-style-type: none"> <li>● KM4 PG</li> <li>● PSRAM_LDO OFF</li> </ul>	<30	PSRAM cannot keep, so memory retention applications are not supported.
Yes	<ul style="list-style-type: none"> <li>● KM4 CG</li> <li>● PSRAM_LDO ON</li> <li>● PSRAM Half Sleep mode</li> </ul>	About 325	

## 20.3 How to Use PSRAM?

### 20.3.1 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
};
```

### 20.3.2 Adding BSS/TEXT/DATA Section into PSRAM Region

Table 20-4 lists how to add a BSS/TEXT/DATA section into PSRAM region.

Table 20-4 Add a BSS/TEXT/DATA section into PSRAM region

Section	Description	Operation
BSS	To put a BSS section in PSRAM, add <code>PSRAM_BSS_SECTION</code> before the buffer definition.	<code>PSRAM_BSS_SECTION u32 PSRAM_Testbuf[1024];</code>
TEXT	To put a TEXT section in PSRAM, add <code>PSRAM_TEXT_SECTION</code> before the function definition.	<pre>PSRAM_TEXT_SECTION VOID Test_Function(VOID) {     u32 i = 0;     for (i = 0; i &lt; 10; i++) {         DBG_8195A("test\r\n");     } }</pre>
DATA	To put a DATA section in PSRAM, add <code>PSRAM_DATA_SECTION</code> or <code>PSRAM_RODATA_SECTION</code> before the data definition.	<code>PSRAM_DATA_SECTION u8 Test_Data = FALSE;</code>

After the operation, rebuild KM4 project.

### 20.3.3 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`.

## 20.4 PSRAM Cache “Write Back” Policy Change Note

### 20.4.1 Cache Policy Change

To prevent PSRAM cell charge leak from frequent access, the cache policy is changed to “Write Back” instead. “Write back” policy also can enhance memory access performance and efficiency.

### 20.4.2 Scope

All of chips stacking PSRAM inside on Ameba-D series: RTL8721CSM, RTL8722CSM, RTL8721DM, and RTL8722DM.

### 20.4.3 Notice

On “Write Through” policy, the written data into cache are updated simultaneously to the corresponding PSRAM. However, on the “Write Back” policy, the synchronization operations need to be taken between cache and PSRAM to keep content consistency, especially for multiple 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 Ameba-D 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.

#### 20.4.3.1 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, keyword “`__attribute__((aligned(32)))`” can be added to realize the starting address 32 bytes alignment.

```
char rx_buf[RX_BUF_SIZE] __attribute__((aligned(32)));
```

#### 20.4.3.2 DMA Operation

The following steps should be added when executing DMA Rx/Tx.

Operation	Step
DMA Rx	(1) Prepare Rx Buffer (2) Do <code>DCache_CleanInvalidate()</code> to avoid cache data write back during DMA Rx (3) Do DMA Rx config (4) Trigger DMA Rx interrupt

	<p>(5) Do DCache_Invalidate() in Rx Done Handler to clean old data (take <b>example_sources/USI_UART/raw/usi_uart_stream_dma/src/main.c</b> for example, <code>uart_recv_string_done()</code> is DMA Rx Done Interrupt Handler)</p> <pre> void uart_recv_string_done(void) {     &gt;&gt; DCache_Invalidate((u32)rx_buf, -SRX_BUF_SZ); /*!!!To solve the cache consistency problem, DMA mode need it!!!*/     &gt;&gt; dma_free();     &gt;&gt; rx_done += 1; }                 </pre> <p>(6) CPU reads Rx Buffer</p>
DMA Tx	<p>(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</p>

In SDK, only the example of one time xxx\_GDMA\_Init one time transmission is illustrated. The step (2) 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.

<pre> BOOL USI_UARTTXGDMA_Init(     &gt;&gt; u8 USIIndex,     &gt;&gt; GDMA_InitTypeDef *GDMA_InitStruct,     &gt;&gt; void *CallbackData,     &gt;&gt; IRQ_FUN CallbackFunc,     &gt;&gt; u8 *pTxBuf,     &gt;&gt; int TxCount     &gt;&gt; ) {     &gt;&gt; u8 GdmaChnl;     &gt;&gt;     &gt;&gt; assert_param(GDMA_InitStruct != NULL);     &gt;&gt; DCache_CleanInvalidate((u32)pTxBuf, TxCount); }                 </pre>	<pre> BOOL USI_UARTRXGDMA_Init(     &gt;&gt; u8 USIIndex,     &gt;&gt; GDMA_InitTypeDef *GDMA_InitStruct,     &gt;&gt; void *CallbackData,     &gt;&gt; IRQ_FUN CallbackFunc,     &gt;&gt; u8 *pRxBuf,     &gt;&gt; int RxCount     &gt;&gt; ) {     &gt;&gt; u8 GdmaChnl;     &gt;&gt; USI_TypeDef *pUSIx;     &gt;&gt;     &gt;&gt; assert_param(GDMA_InitStruct != NULL);     &gt;&gt; DCache_CleanInvalidate((u32)pRxBuf, RxCount); }                 </pre>
---	--

### 20.4.3.3 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.

# 21 MPU and Cache

## 21.1 Functional description

### 21.1.1 MPU

Memory Protection Unit (MPU) is used to provide Hardware Protection by Software definition. Our code provides `mpu_region_config` structure to include the region memory attribute. Default attribute of all KM0 & KM4 SRAM are write-through.

Table 21-1 shows member variables of the `mpu_region_config` structure.

Table 21-1 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 <ul style="list-style-type: none"> <li>● MPU_EXEC_ALLOW: Allows program execution in this region</li> <li>● MPU_EXEC_NEVER: Does not allow program execution in this region</li> </ul>
ap	uint8_t	Access permissions <ul style="list-style-type: none"> <li>● MPU_PRIV_RW: Read/write by privileged code only</li> <li>● MPU_UN_PRIV_RW: Read/write by any privilege level</li> <li>● MPU_PRIV_R: Read only by privileged code only</li> <li>● MPU_PRIV_W: Read only by any privilege level</li> </ul>
sh	uint8_t	Share ability for Normal memory <ul style="list-style-type: none"> <li>● MPU_NON_SHAREABLE: Non-shareable</li> <li>● MPU_OUT_SHAREABLE: Outer shareable</li> <li>● MPU_INR_SHAREABLE: Inner shareable</li> </ul>
attr_idx	uint8_t	Memory attribute indirect index This parameter can be a value of 0 ~ 7, the detailed attribute is defined in <code>mpu_init()</code> and is customized. The typical definition is as following: <ul style="list-style-type: none"> <li>● 0: MPU_MEM_ATTR_IDX_NC, defines memory attribute of Normal memory with non-cacheable</li> <li>● 1: MPU_MEM_ATTR_IDX_WT_T_RA, defines memory attribute of Normal memory with write-through transient, read allocation</li> <li>● 2: MPU_MEM_ATTR_IDX_WB_T_RWA, defines memory attribute of Normal memory with write-back transient, read and write allocation</li> <li>● 3 ~ 7: MPU_MEM_ATTR_IDX_DEVICE, defines memory attribute of Device memory with non-gathering, non-recording, non-early Write Acknowledge</li> </ul>

Table 21-2 shows how to set a MPU region.

Table 21-2 How to set a MPU region

Steps	Description
New variable and structure	<ul style="list-style-type: none"> <li>● Variable to store MPU entry index</li> <li>● Structure <code>mpu_region_config</code> to store the region memory attribute</li> </ul>
Allocate a free MPU entry	Call <code>mpu_entry_alloc()</code>
Set region memory attribute	Set structure of region memory attribute
Configure MPU region memory attribute	Call <code>mpu_region_cfg()</code>

### 21.1.2 Cache

Cache is used to improve the CPU performance of data access. Ameba-D Cache supports Enable/Disable, Flush and Clean Operation, as Table 21-3 lists.

Table 21-3 Enable/Disable, Flush and Clean operation supported by Cache

Operation	Description	I-Cache	D-Cache
Enable/Disable	Enable or Disable Cache function	✓	✓
Flush (Invalidate)	<ul style="list-style-type: none"> <li>● Flush Cache</li> <li>● D-Cache can be flushed by address</li> <li>● Can be used after DMA Rx, and CPU reads DMA data from DMA buffer for D-Cache</li> </ul>	✓	✓
Clean	<ul style="list-style-type: none"> <li>● Clean D-Cache</li> <li>● D-Cache will be write back to memory</li> <li>● D-Cache can be cleaned by address</li> <li>● Can be used before DMA Tx, after CPU writes data to DMA buffer for D-Cache</li> </ul>	✗	✓

## 21.2 MPU APIs

### 21.2.1 mpu\_init

Items	Description
Introduction	Initialize MPU region memory attribute to typical value
Parameters	N/A
Return	N/A

### 21.2.2 mpu\_set\_mem\_attr

Items	Description
Introduction	Change MPU region memory attribute
Parameters	<ul style="list-style-type: none"> <li>● attr_idx: region memory attribute index, which can be 0 ~ 7.</li> <li>● mem_attr: region memory attributes.</li> </ul>
Return	N/A

### 21.2.3 mpu\_region\_cfg

Items	Description
Introduction	Configure MPU region memory attribute.
Parameters	<ul style="list-style-type: none"> <li>● region_num: <ul style="list-style-type: none"> <li>■ KM0: 0 ~ 3</li> <li>■ KM4_NS: 0 ~ 7</li> <li>■ KM4_S: 0 ~ 3</li> </ul> </li> <li>● pmpu_cfg: pointer to an <b>mpu_region_config</b> structure which has been configured</li> </ul>
Return	N/A

### 21.2.4 mpu\_entry\_free

Items	Description
Introduction	Free MPU entry
Parameters	entry_index: <ul style="list-style-type: none"> <li>● KM0: 0 ~ 3</li> <li>● KM4_NS: 0 ~ 7</li> <li>● KM4_S: 0 ~ 3</li> </ul>
Return	N/A



### 21.2.5 mpu\_entry\_alloc

Items	Description
Introduction	Allocate a free MPU entry
Parameters	N/A
Return	MPU entry index: <ul style="list-style-type: none"> <li>● KM0: 0 ~ 3</li> <li>● KM4_NS: 0 ~ 7</li> <li>● KM4_S: 0 ~ 3</li> <li>● Fail: -1</li> </ul>

## 21.3 Cache APIs

### 21.3.1 ICache\_Enable

Items	Description
Introduction	Enable I-Cache
Parameters	N/A
Return	N/A

### 21.3.2 ICache\_Disable

Items	Description
Introduction	Disable I-Cache
Parameters	N/A
Return	N/A

### 21.3.3 ICache\_Invalidate

Items	Description
Introduction	Invalidate I-Cache
Parameters	N/A
Return	N/A

### 21.3.4 DCache\_IsEnabled

Items	Description
Introduction	Check D-Cache enabled or not
Parameters	N/A
Return	D-Cache enable status: <ul style="list-style-type: none"> <li>● 1: Enable</li> <li>● 0: Disable</li> </ul>

### 21.3.5 DCache\_Enable

Items	Description
Introduction	Enable D-Cache
Parameters	N/A
Return	N/A

### 21.3.6 DCache\_Disable

Items	Description
Introduction	Disable D-Cache
Parameters	N/A
Return	N/A

### 21.3.7 DCache\_Invalidate

Items	Description
Introduction	Invalidate D-Cache by address
Parameters	<ul style="list-style-type: none"> <li>● Address: Invalidated address (aligned to 32-byte boundary)</li> <li>● Bytes: Size of memory block (in number of bytes)</li> </ul>
Return	N/A

### 21.3.8 DCache\_Clean

Items	Description
Introduction	Clean D-Cache by address
Parameters	<ul style="list-style-type: none"> <li>● Address: Clean address (aligned to 32-byte boundary).</li> <li>● Bytes: size of memory block (in number of bytes).</li> </ul> <p><b>Note:</b> Address set 0xFFFFFFFF is used to clean all D-Cache.</p>
Return	N/A

### 21.3.9 DCache\_CleanInvalidate

Items	Description
Introduction	Clean and invalidate D-Cache by address
Parameters	<ul style="list-style-type: none"> <li>● Address: Clean and invalidated address (aligned to 32-byte boundary)</li> <li>● Bytes: size of memory block (in number of bytes)</li> </ul> <p><b>Note:</b> Address set 0xFFFFFFFF is used to clean and flush all D-Cache.</p>
Return	N/A

## 21.4 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];
```

# 22 Audio Codec Controller Guide

## 22.1 Audio Codec

Ameba-D audio codec (AC) 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.

Ameba-D audio codec integrates anti-pop circuit for audible pop noise cancellation, MIC bias circuit for MIC power supply and programmable MIC boost ability.

Ameba-D audio codec transmits record data to or receives playback data from platform through SPORT interface. By means of specific serial interface (SI), Ameba-D platform configures and drives audio codec.

Ameba-D also provides external I<sup>2</sup>S interface for audio application extension, which supports the highest 384kHz sampling frequency.

### 22.1.1 Diagram

The diagram of AC is shown in Fig 22-1.

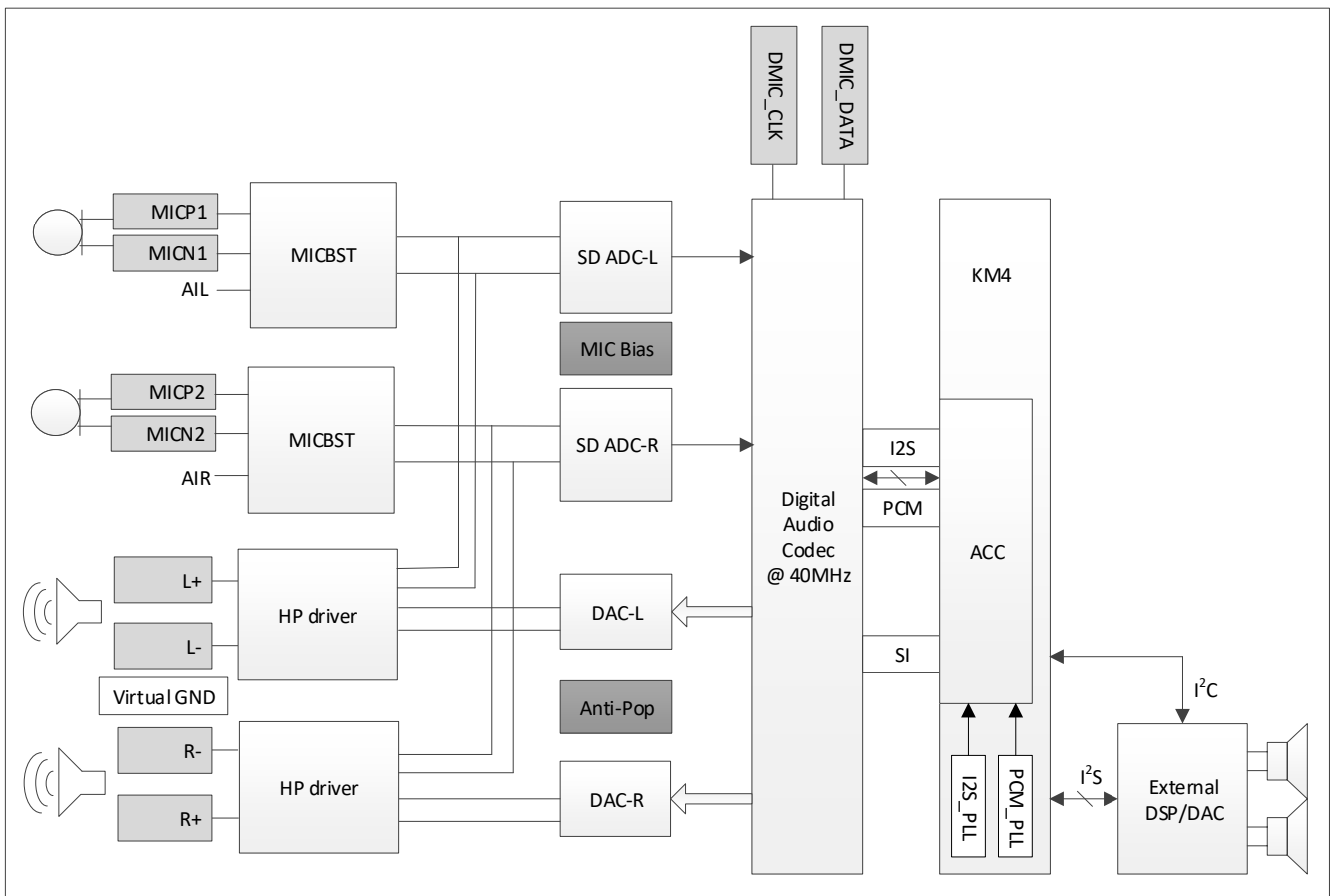


Fig 22-1 Audio codec diagram

## 22.1.2 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-N and PCM mode B-N 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

## 22.1.3 Application Mode

Audio codec supports three input ways: line-in, AMIC-in and DMIC-in, but only supports one output way: line-out.

### 22.1.3.1 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.

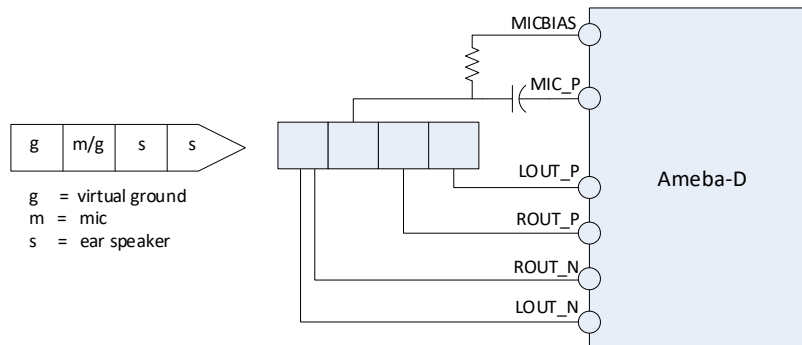


Fig 22-2 Cap-less mode connection with headphone jack

- 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.

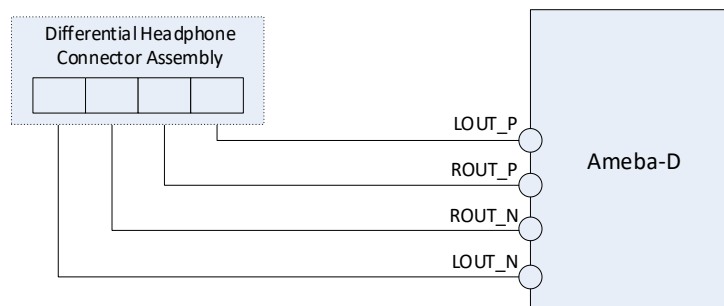


Fig 22-3 Differential mode connection with headphone jack

● Line-out single-ended mode

In this mode, board circuit designer needs to place a capacitor to the P-end output path for analog audio signal pick-up. No N-end output is required.

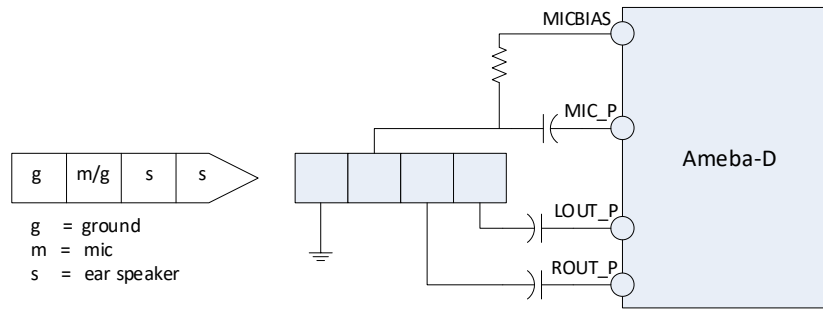


Fig 22-4 Single-ended mode connection with headphone jack

22.1.3.2 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.

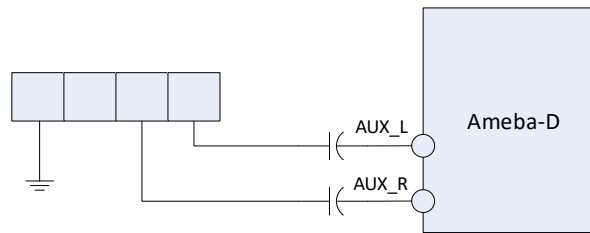


Fig 22-5 Line-in mode connection

22.1.3.3 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.

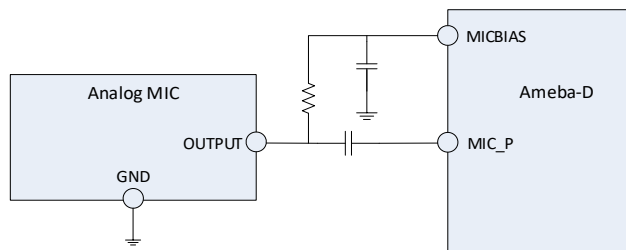


Fig 22-6 AMIC-in single-ended mode connection

● AMIC-in differential mode

Connect MIC\_P/MIC\_N with differential analog microphone, while MICBIAS provides the microphone bias voltage.

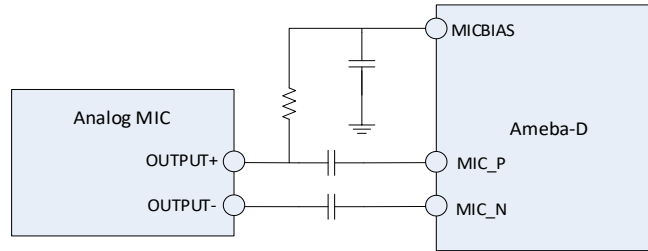


Fig 22-7 AMIC-in differential mode connection

### 22.1.3.4 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.

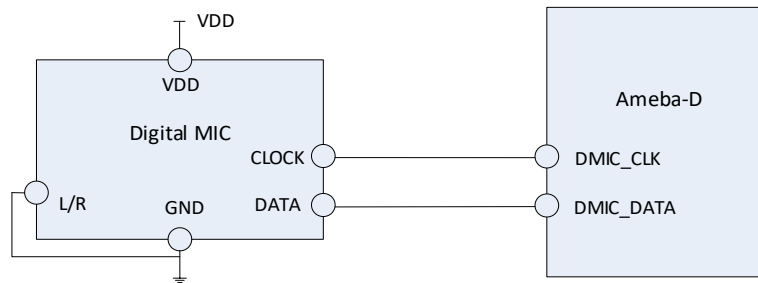


Fig 22-8 DMIC-in mono mode connection

- 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.

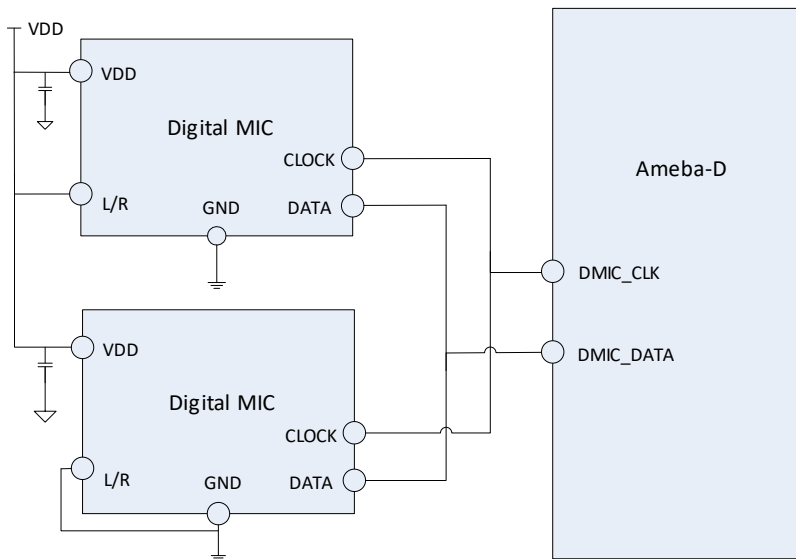


Fig 22-9 DMIC-in stereo mode connection

## 22.2 Audio Codec Controller

Ameba-D audio codec controller (ACC) is the bridge between host audio buffers and audio codec module. It is used for audio codec input/output control.

Ameba-D 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 22-10.

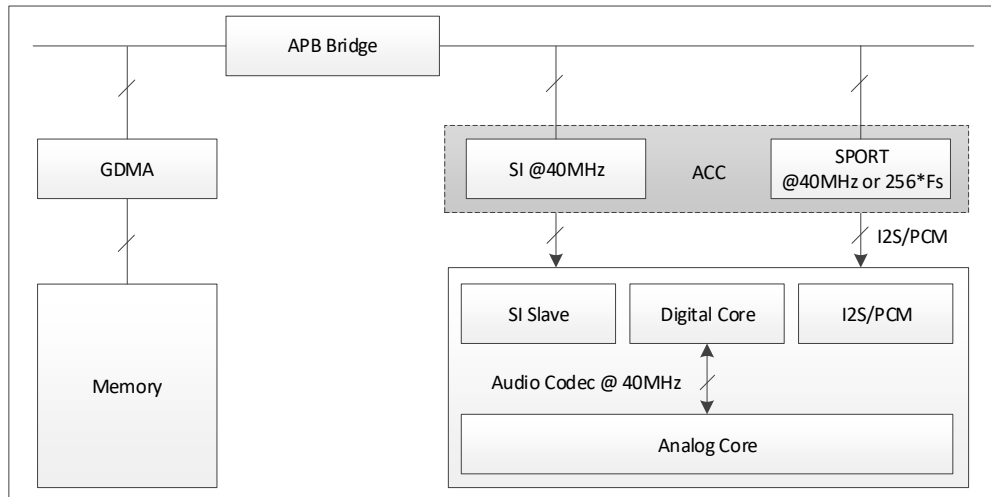


Fig 22-10 Audio codec controller diagram

### 22.2.1 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-N and PCM mode B-N data format
- Mandatory or optional sample rate which audio codec module declares for support
- GDMA for data moving
- Data loopback between SDI and SDO

### 22.2.2 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 user's setting. It supports multiple data format, such as I2S, left justify, PCM mode A, PCM mode B, PCM mode A-N and PCM mode B-N.

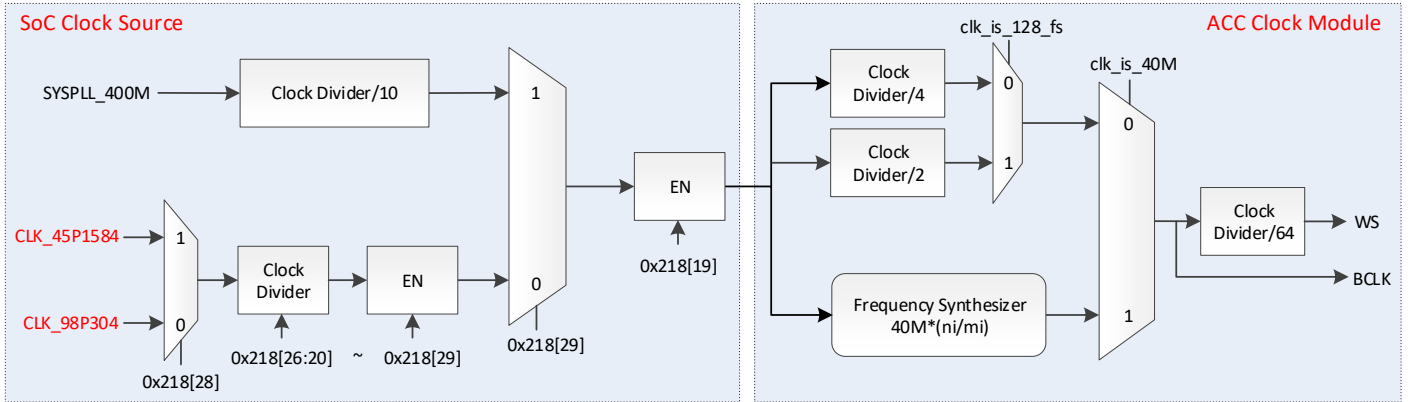
- SI interface

Audio codec needs to configure the related analog and digital parameters before transferring audio data. SI interface is used to configure codec parameters. It can read or write codec register.

For more details about AC and AAC, refer to UM0400 Ameba-D User Manual.

## 22.3 Audio PLL

The ACC clock architecture is shown in Fig 22-11.



Note: CLK\_45P1584 is clock derived from PCM PLL internal, while CLK\_98P304 from I<sup>2</sup>S PLL.

Fig 22-11 ACC clock architecture

### 22.3.1 Diagram

The ACC clock diagram is illustrated in Fig 22-12.

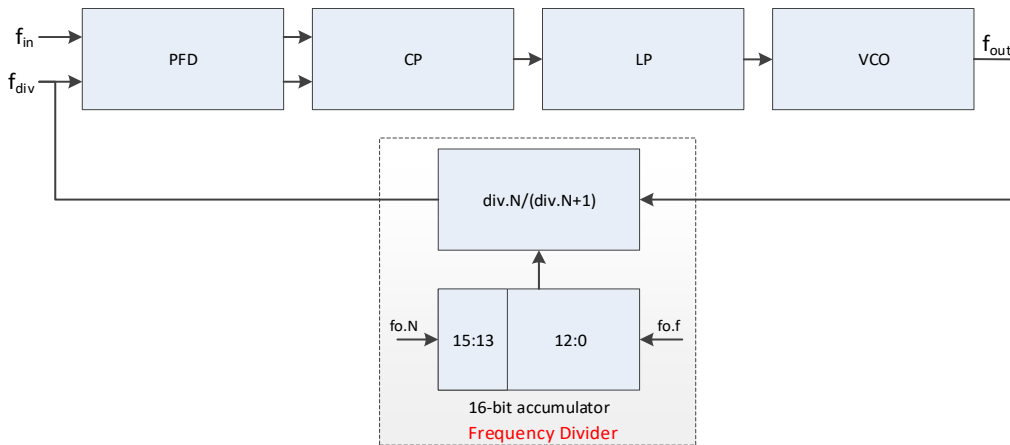


Fig 22-12 ACC clock diagram

$$\frac{f_{out}}{f_{in}} = \frac{div.N \times (2^{16} - (fo.N \times 2^{13} + fo.f)) + (div.N + 1) \times (fo.N \times 2^{13} + fo.f)}{2^{16}} = div.N + \frac{fo.N}{2^3} + \frac{fo.f}{2^{16}}$$

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 Table 22-1.

Table 22-1 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

## 22.3.2 Operation Mode

### 22.3.2.1 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, 196.608MHz (=98.304MHz x 2) is offered by I<sup>2</sup>S PLL while 180.6336MHz (=45.1584MHz x 4) by PCM PLL.

### 22.3.2.2 Manual Mode

If fine tuning PLL clock is required, users could switch the I<sup>2</sup>S or PCM PLL to manual mode. In this mode, users could control the output frequency of I<sup>2</sup>S 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<sub>out</sub> equals to f<sub>in</sub>/2<sup>16</sup>, f<sub>in</sub> decides the exact ppm per step ((f<sub>in</sub>/2<sup>16</sup>)/196.608 or (f<sub>in</sub>/2<sup>16</sup>)/180.6336).

The relationship between f<sub>in</sub> and ppm per step is listed in Table 22-2.

Table 22-2 Relationship between f<sub>in</sub> and ppm per step

f <sub>in</sub> (MHz)	I <sup>2</sup> S 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

## 22.4 Audio Codec APIs

### 22.4.1 PLL APIs

API	Introduction
<PLL_Div>	Divider to generate 256*fs or 128*fs clock
<PLL_Sel>	Selects I <sup>2</sup> S PLL or PCM PLL
<PLL_I2S_Set>	Enables or disables I <sup>2</sup> S PLL
<PLL_PCM_Set>	Enables or disables PCM PLL
<PLL_I2S_ClkTune>	Tunes I <sup>2</sup> S 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
-------------------	--

### 22.4.1.1 PLL\_Div

Parameter	Type	Introduction
<div>	u32	Divider to generate 256*fs or 128*fs clock.

### 22.4.1.2 PLL\_Sel

Parameter	Type	Introduction
<sel>	u32	<ul style="list-style-type: none"> <li>● Selects I<sup>2</sup>S PLL if fs=8/16/32/48/96kHz</li> <li>● Selects PCM PLL if fs=44.1/88.2kHz</li> </ul>

### 22.4.1.3 PLL\_I2S\_Set

Parameter	Type	Introduction
<new_state>	u32	Enables or disables I <sup>2</sup> S PLL

### 22.4.1.4 PLL\_PCM\_Set

Parameter	Type	Introduction
< new_state >	u32	Enables or disables PCM PLL

### 22.4.1.5 PLL\_I2S\_ClkTune

Parameter	Type	Introduction
<ppm>	u32	Required fine tuning ppm value
<action>	U32	Faster or slower, or reset to auto mode

### 22.4.1.6 PLL\_PCM\_ClkTune

Parameter	Type	Introduction
<ppm>	u32	Required fine tuning ppm value
<action>	U32	Faster or slower, or reset to auto mode

## 22.4.2 SPORT APIs

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

### 22.4.2.1 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

### 22.4.2.2 AUDIO\_SP\_Init

Initializes the audio SPORT registers according to the specified parameters in SP\_InitStruct.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	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

### 22.4.2.3 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_TypeDef*	The base address of audio SPORT peripheral
<NewState>	u32	State (enable or disable) of the SPORT Tx

### 22.4.2.4 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

### 22.4.2.5 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

### 22.4.2.6 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

### 22.4.2.7 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

### 22.4.2.8 AUDIO\_SP\_GetWordLen

Gets the audio SPORT word length.

Parameter	Type	Introduction
<SPORTx>	AUDIO_SPORT_TypeDef*	The base address of audio SPORT peripheral

### 22.4.2.9 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

### 22.4.2.10 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

### 22.4.2.11 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
----------	-----	----------------

### 22.4.3 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

#### 22.4.3.1 AUDIO\_SI\_Cmd

Enables or disables the specified audio SI peripheral.

Parameter	Type	Introduction
<new_state>	u8	New state of the SI peripheral

#### 22.4.3.2 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

#### 22.4.3.3 AUDIO\_SI\_ReadReg

Uses SI interface to read codec register.

Parameter	Type	Introduction
<address>	u32	Codec register address

#### 22.4.3.4 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

### 22.4.4 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 AD channel
<CODEC_MutePlay>	Mutes or unmutes per DA channel.
<CODEC_Delnit>	De-initializes codec peripheral

#### 22.4.4.1 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

#### 22.4.4.2 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) <ul style="list-style-type: none"> <li>● 8'hAF: 0dB</li> <li>● 8'h00: -65.625dB</li> </ul>
<vol_rch>	u8	Codec mon DAC right channel DVOL gain control (0.375dB/step) <ul style="list-style-type: none"> <li>● 8'hAF: 0dB</li> <li>● 8'h00: -65.625dB</li> </ul>

#### 22.4.4.3 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)

#### 22.4.4.4 CODEC\_SetSr

Sets codec ADC and DAC sample rate.

Parameter	Type	Introduction
<sample_rate>	u32	Codec ADC and DAC sample rate

#### 22.4.4.5 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

#### 22.4.4.6 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

#### 22.4.4.7 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

#### 22.4.4.8 CODEC\_SetMicBias

Sets MIC\_BIAS output voltage.

Parameter	Type	Introduction
<mic_bias>	u8	Microphone bias voltage setting

#### 22.4.4.9 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

#### 22.4.4.10 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

#### 22.4.4.11 CODEC\_DeInit

De-initializes codec peripheral.

Parameter	Type	Introduction
<application>	u32	Codec application mode

## 22.5 How to Use AC APIs?

### 22.5.1 Audio Play

To play the audio data through audio codec, follow the steps below:

- (1) Open audio codec clock and function
  - PLLx\_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);

### 22.5.2 Audio Record

To record the audio data through audio codec, follow the steps below:

- (1) Open audio codec clock and function  
PLLx\_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 can select APP\_AMIC\_IN for analog microphone, select APP\_DMIC\_IN for digital microphone, or select APP\_LINE\_IN.
- (4) If codec needs to change volume, use  
CODEC\_SetVolume (vol\_lch, vol\_rch);
- (5) If codec needs 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);

### 22.5.3 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/adac
	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 self-parse	/component/common/example/audio_sport/audio_m4a_selfparse
	MP3	/component/common/example/audio_sport/audio_mp3

**Note:** In the example code, we don't disable the GDMA even if no valid data are available. If you want to disable the GDMA, it should be done in a GDMA related interrupt routine. Disabling the GDMA outside of an interrupt routine may cause an exception if the GDMA is still transferring data.

## 22.6 Hardware Design Guide

### 22.6.1 Line-out

The line-out connection of audio codec is illustrated in Fig 22-13. 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.

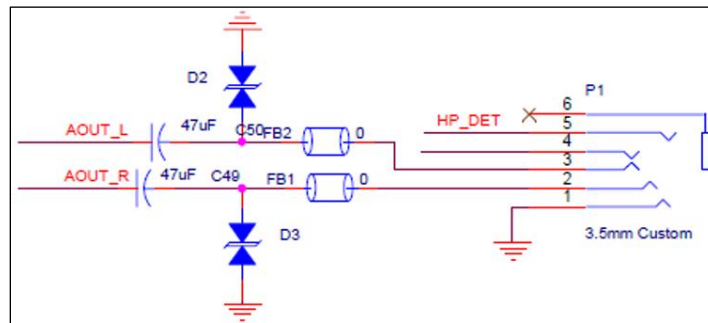


Fig 22-13 Line-out connection

### 22.6.2 AMIC-in

The AMIC-in connection of audio codec is illustrated in Fig 22-14. The capacitor between analog microphone and IC should be 1uF. Larger capacitance value makes longer period needed for capacitor charging.

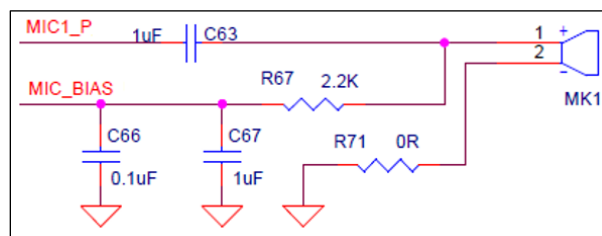


Fig 22-14 AMIC-in connection

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.

### 22.6.3 Power

The power connection of audio codec is illustrated in Fig 22-15.

- The capacitor between AUDIO\_VREF and ground should be 4.7nF. Larger capacitance value makes longer period needed for AVCC's stabilization.
- The capacitor between AVCC or AVCC\_DRIV and ground should be 1uF or a little larger to keep voltage stable.

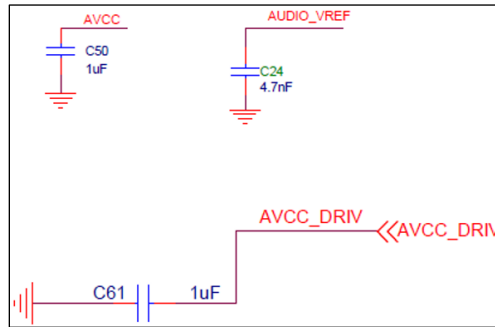


Fig 22-15 Power connection

## 22.7 Performance of Encoding & Decoding

The performance of decoding or encoding audio files of different formats are listed in Table 22-3 ~ Table 22-9. 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.

### 22.7.1 AC3 Format

The performance of decoding AC3 format audio data is listed in Table 22-3.

Table 22-3 Performance of decoding AC3 format audio data

AC Channel	Bit Rate (kHz)	Sample Rate (kHz)	Output Channel	Average MIPS Measured <sup>1</sup>	MIPS Simulated <sup>2</sup>	
					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.
2. The values are evaluated with the Keil simulator.

### 22.7.2 OPUS Format

The simulated CPU load of encoding and decoding of OPUS audio data is listed in Table 22-4. 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 22-4 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 22-5. The version used here is libopus 1.1.4.

Table 22-5 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 22-6, CPU load on using Silk encoder is listed.

Table 22-6 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	

### 22.7.3 FLAC Format

The simulated (with Keil simulator) and measured CPU load for decoding FLAC audio data is listed in Table 22-7. 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 22-7 CPU load of decoding FLAC audio data

Rate (kHz)	Channel	Word Length	MIPS Simulated	MIPS Measured
44.1	2	16	19	9.66

### 22.7.4 AAC Format

The measured CPU load for decoding AAC audio data is listed in Table 22-8. 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 22-8 CPU load of decoding AAC audio data

Rate (kHz)	Channel	Bit Rate (kbps)	MIPS
48	2	320	28.93

### 22.7.5 MP3 Format

The measured CPU load for decoding MP3 audio data is listed in Table 22-9. 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 22-9 CPU load of decoding MP3 audio data

Rate (kHz)	Channel	Bit Rate (kbps)	MIPS
32	2	320	30.43

## 22.8 Q & A

### 22.8.1 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 22-16 (power amplifier is LM4991).
- If the power amplifier is a D type amplifier, refer to the design shown in Fig 22-17 (power amplifier is ALC1003).

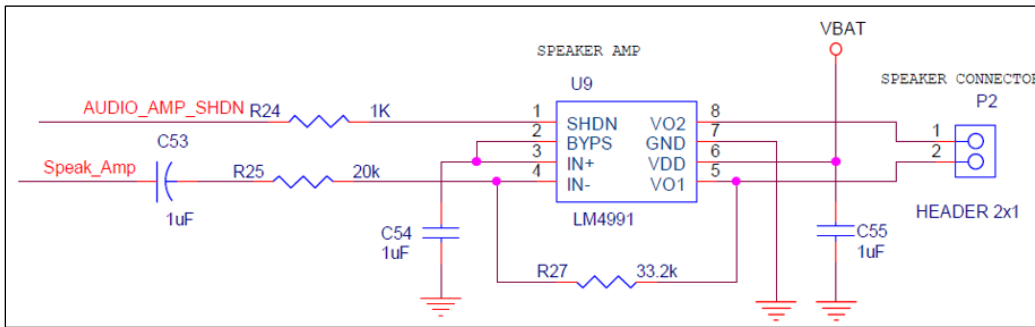


Fig 22-16 Reference design of using AB type power amplifier

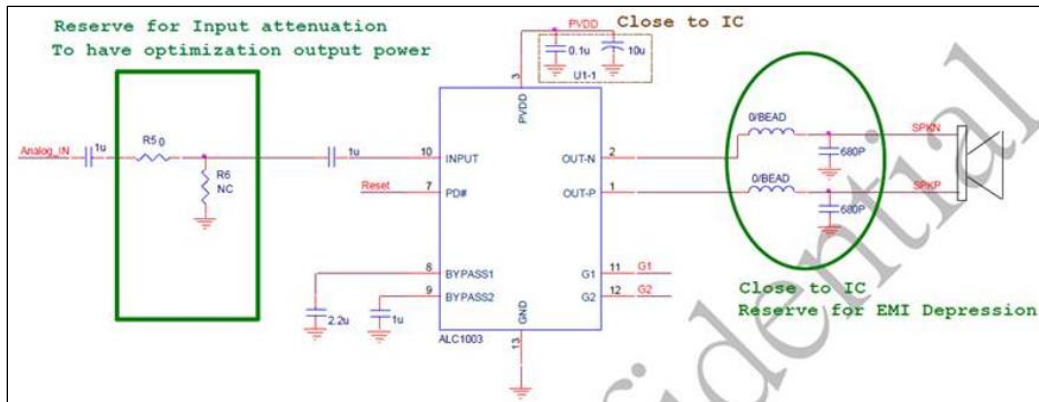


Fig 22-17 Reference design of using D type power amplifier

### 22.8.2 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-D supports 2 mics. One supports both single-ended input and differential inputs, but another only supports single-ended input.

### 22.8.3 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 them. Passing address directly to AUDIO\_SP\_TXGDMA\_Init() doesn't work. The reason is that flash doesn't support GDMA transfer, but SRAM or PSRAM supports.

# 23 Cap-Touch

Ameba-D Cap-Touch 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 Cap-Touch and design touch key.

## 23.1 Pinmux

The pin assignments of Cap-Touch channels are listed in Table 19-4.

Table 23-1 Cap-Touch 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	✗	✓	✓

## 23.2 APIs

### 23.2.1 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 Cap-Touch and each channel, which includes the debounce control, ETC parameters, mbias current and threshold for channels.

### 23.2.2 CapTouch\_Init

Items	Description
Introduction	Initializes the Cap-Touch peripheral according to the specified parameters in the CapTouch_InitStruct.
Parameters	<ul style="list-style-type: none"> <li>CapTouch: which should be CAPTOUCH_DEV.</li> <li>CapTouch_InitStruct: pointer to a CapTouch_InitTypeDef structure that contains the configuration information for the specified Cap-Touch peripheral.</li> </ul>
Return	N/A

### 23.2.3 CapTouch\_Cmd

Items	Description
Introduction	Enables or disables the specified Cap-Touch peripheral.
Parameters	<ul style="list-style-type: none"> <li>CapTouch: which should be CAPTOUCH_DEV.</li> <li>NewState: new state of the Cap-Touch peripheral. This parameter can be ENABLE or DISABLE.</li> </ul>
Return	N/A

### 23.2.4 CapTouch\_INTConfig

Items	Description
Introduction	Enables or disables the specified Cap-Touch interrupts.
Parameters	<ul style="list-style-type: none"> <li>● CapTouch: which should be CAPTOUCH_DEV.</li> <li>● CapTouch_IT: specifies the Cap-Touch interrupt to be enabled or masked. This parameter can be one or combinations of the following values:                             <ul style="list-style-type: none"> <li>■ BIT_CT_OVER_N_NOISE_THRESHOLD_INT: Cap-Touch negative noise overflow interrupt</li> <li>■ BIT_CT_FIFO_OVERFLOW_INT: Cap-Touch FIFO overflow interrupt</li> <li>■ BIT_CT_OVER_P_NOISE_THRESHOLD_INT: Cap-Touch positive noise overflow interrupt</li> <li>■ CT_CHX_PRESS_INT(x): Cap-Touch channel(x) press interrupt, where x can be 0~3</li> <li>■ CT_CHX_RELEASE_INT(x): Cap-Touch channel(x) release interrupt, where x can be 0~3</li> </ul> </li> <li>● NewState: new state of the specified Cap-Touch interrupts mask. This parameter can be ENABLE or DISABLE.</li> </ul>
Return	N/A

### 23.2.5 CapTouch\_GetISR

Items	Description
Introduction	Gets Cap-Touch interrupt status.
Parameters	CapTouch: which should be CAPTOUCH_DEV.
Return	Interrupt status

### 23.2.6 CapTouch\_INTClearPendingBit

Items	Description
Introduction	Clears the specified Cap-Touch interrupt pending bit.
Parameters	<ul style="list-style-type: none"> <li>● CapTouch: which should be CAPTOUCH_DEV.</li> <li>● CapTouch_IT: specifies the Cap-Touch interrupt to be cleared. This parameter can be one or combinations of the following values:                             <ul style="list-style-type: none"> <li>■ BIT_CT_N_NOISE_OVERFLOW_INT_CLR: Cap-Touch negative noise overflow interrupt</li> <li>■ BIT_CT_FIFO_OVERFLOW_INT_CLR: Cap-Touch FIFO overflow interrupt</li> <li>■ BIT_CT_P_NOISE_OVERFLOW_INT_CLR: Cap-Touch positive noise overflow interrupt</li> <li>■ CT_CHX_PRESS_INT(x): Cap-Touch channel(x) press interrupt, where x can be 0~3</li> <li>■ CT_CHX_RELEASE_INT(x): Cap-Touch channel(x) release interrupt, where x can be 0~3</li> </ul> </li> </ul>
Return	N/A

## 23.3 How to Use CTC

### 23.3.1 CTC Initialization

The Cap-Touch 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 (Rtl8721dip_sleepcfg.c)` to `BIT_CAPTOUCH_ENABLE` to enable Cap-Touch when KM0 boots.

```

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 (**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 23.3.2.

```
static CapTouch_CHInitTypeDef CTCChan[4] =
{
    /*DiffThreshold, MbiasCurrent, ETCNNoiseThr, ETCNNoiseThr, 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 re-burn images, the Cap-Touch would work after boot up. When finger touch or proximity is detected, the Cap-Touch would send a “Key Press” interrupt to system.

### 23.3.2 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 23-2. 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\(\)](#) to enable channel 0.
 

```
CapTouch_StructInit(&CapTouch_InitStruct);
CapTouch_InitStruct.CT_Channel[0].CT_CHEnable = ENABLE;
```
- (2) Initialize the Cap-Touch peripheral according to CapTouch\_InitStruct in step (1).
 

```
CapTouch_Init(CAPTOUCH_DEV, &CapTouch_InitStruct);
```
- (3) Enable Cap-Touch.
 

```
CapTouch_Cmd(CAPTOUCH_DEV, ENABLE);
```
- (4) Call CapTouch\_GetChAveData() to read the sample data from channel 0 periodically.
 

```
CapTouch_GetChAveData(CAPTOUCH_DEV, 0);
```

Table 23-2 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

## 23.4 Cap-Touch Schematic Design and PCB Layout Guidelines

The CTC of Ameba-D has 4 Cap-Touch channels, and all the four channels support buttons and proximity sensors. Both the schematic design and PCB layout are important for Cap-Touch application. This chapter provides guidelines of Cap-Touch schematic design and PCB layout.

### 23.4.1 Cap-Touch Schematic Design

The CTC supports 4 channels, each channel supports button and proximity sensor, while only 2 channels are used. When designing the Cap-Touch schematic, you shouldn't 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 Cap-Touch channels must have a 47Ω series resistance (placed close to the chip) to improve noise immunity both for button and proximity sensor. If any Cap-Touch 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 23-1 is a schematic in which only 4 channels are enabled.

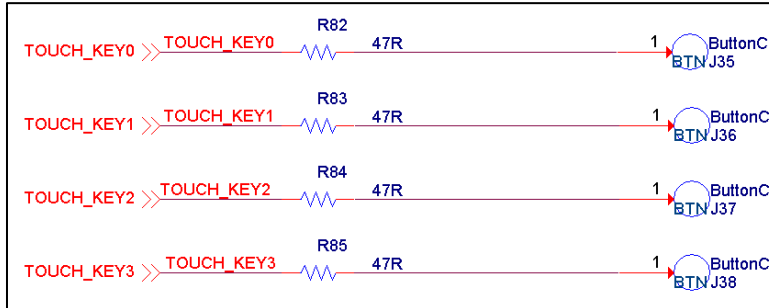


Fig 23-1 4 Channels enabled schematic

### 23.4.2 PCB Layout

In a typical Cap-Touch application, sensors are constructed with traces on a FR4/FR2 or flexible printed circuit (FPC). Cap-Touch layout design is an important step in the design phase, following the PCB layout design guidelines 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 like an antenna and reduces the noise immunity of the sensor.
- Power consumption: The power consumption of the sensor depends on 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 23-3 summarizes the sensor layout guidelines while designing a Cap-Touch application, both of button application and proximity application.

Table 23-3 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	<ul style="list-style-type: none"> <li>● Circular or rectangular loop (with round corners) on PCB for large area.</li> <li>● Circular or rectangular solid fill (with round corners) for small area.</li> <li>● Line with recommended sensor trace width.</li> <li>● A GND loop surrounding the sensor results in better noise immunity but worse sensitivity.</li> </ul>
	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 23-4 summarizes the layout guidelines for placement of components, routing of sensor, sensor trace and GND layer while designing a Cap-Touch application, both button application and proximity application.

Table 23-4 Other layout recommendations

Category	Item	Min.	Max.	Description
Placement	2 layer PCB	N/A	N/A	<ul style="list-style-type: none"> <li>● Top: Sensors, devices, and components; VDD and GND traces, other signal traces</li> <li>● Bottom: Sensor traces, devices and components; VDD and GND traces, other signal traces</li> </ul> <b>Note:</b> <ul style="list-style-type: none"> <li>● Avoid VDD traces, clock traces, and driven traces over sensor traces.</li> <li>● Avoid sensor trace run parallel to a clock signal or driven signal.</li> </ul>
	4 layer PCB	N/A	N/A	<ul style="list-style-type: none"> <li>● Top: Sensors, devices, and components</li> <li>● Second layer: Hatched ground</li> <li>● Third layer: Sensor traces, VDD layer</li> <li>● Bottom: Devices, and components</li> </ul> <b>Note:</b> 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. <b>Note:</b> 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 or no 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 23-2 is a PCB design of button and proximity application.

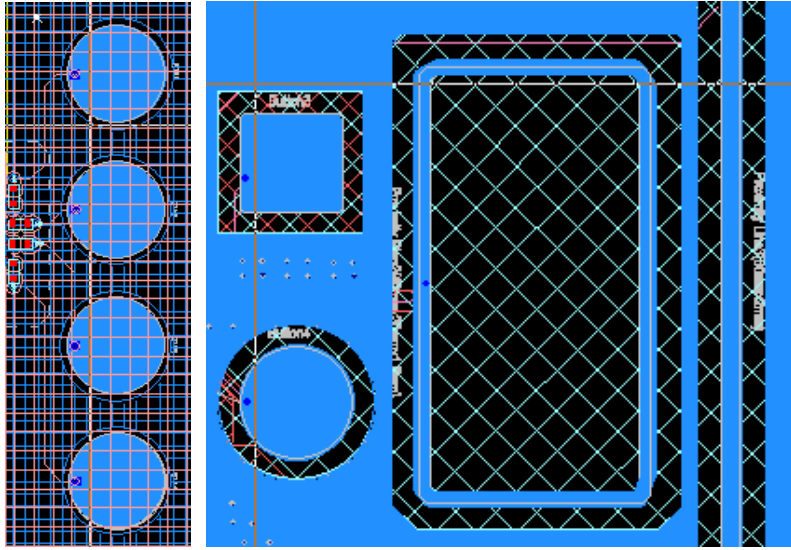


Fig 23-2 PCB design of button and proximity application

# 24 Infrared Radiation (IR)

Ameba-D Infrared Radiation (IR) provides hardware modulation for Infrared Radiation sending and hardware auto capture for receiving.

This chapter introduces how to use IR.

## 24.1 Pinmux

The pin assignments of IR are listed in Table 24-1.

Table 24-1 IR pin assignments

Port Name	Pin Name	QFN48	QFN68	QFN88
PA[25]	IR_TX	√	√	√
PA[26]	IR_RX	√	√	√
PB[23]	IR_TX	X	√	√
PB[22]	IR_RX	X	√	√
PB[31]	IR_TX	X	√	√
PB[29]	IR_RX	X	√	√

## 24.2 Data Format

This section introduces the data format of Tx FIFO and Rx FIFO.

### 24.2.1 IR Tx

To send IR data, you should write the data into IR Tx FIFO register, so it's important to understand the data format to be sent to Tx FIFO 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. Let  $f_{carrier}$  represents the carrier frequency (the unit of  $f_{carrier}$  is kHz),  $T_{duration}$  represents the duration of carrier or no carrier symbol (the unit of  $T_{duration}$  is us). Then,

$$\text{bit}[27:0] = f_{carrier} * T_{duration} / 1000$$

Take NEC protocol for example, the carrier frequency is 38kHz. The format of NEC is shown in Fig 24-1 and the modulation of NEC is shown in Fig 24-2. The NEC format consists of 2 start symbols, 64 data symbols and 1 stop symbol.



Fig 24-1 NEC format

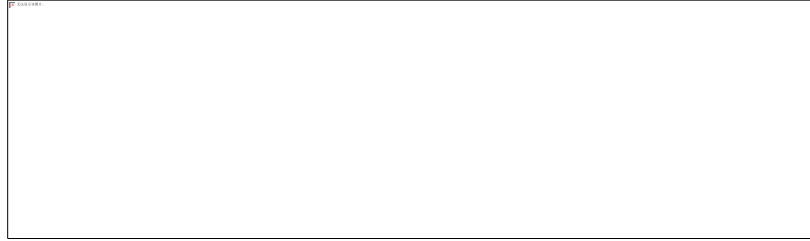


Fig 24-2 NEC modulation

- To send logical “1”, you should write two data into Tx FIFO register.
  - 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.
- To send logical “0”, you should write two data into Tx FIFO register.
  - 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.

### 24.2.2 IR Rx

To receive IR data, you should read the data in Rx FIFO register, so it’s important to understand the data format to be received in Rx FIFO 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  $f_{sampling}$  represents the sampling frequency (the unit of  $f_{sampling}$  is kHz),  $T_{level}$  represents the duration of each high/low level (the unit of  $T_{level}$  is us). Then,

$$T_{level} = \frac{bit[30:0]}{f_{sampling}} * 1000$$

With the use of  $T_{level}$ , you can do further processing to get the information you need.

## 24.3 APIs

### 24.3.1 IR Setting APIs

#### 24.3.1.1 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

#### 24.3.1.2 IR\_Init

Items	Description
Introduction	Initializes the IR peripheral according to the specified parameters in the IR_InitStruct.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● IR_InitStruct: pointer to a IR_InitTypeDef structure that contains the configuration information for the specified IR peripheral.</li> </ul>
Return	N/A

24.3.1.3 IR\_Cmd

Items	Description
Introduction	Enables or disables the selected IR mode.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● mode: selected IR operation mode. This parameter can be the following values:                             <ul style="list-style-type: none"> <li>■ IR_MODE_TX: Transmission mode.</li> <li>■ IR_MODE_RX: Receiving mode.</li> </ul> </li> <li>● NewState: new state of the operation mode.                             <ul style="list-style-type: none"> <li>■ ENABLE</li> <li>■ DISABLE</li> </ul> </li> </ul>
Return	N/A

24.3.1.4 IR\_INTConfig

Items	Description
Introduction	Enables or disables the specified IR interrupts.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● IR_INT: specifies the IR interrupt sources to be enabled or disabled. This parameter can be one or combinations of the following values:                             <ul style="list-style-type: none"> <li>■ IR_TX_FIFO_EMPTY_INT_EN: Tx FIFO empty interrupt.</li> <li>■ IR_TX_FIFO_LEVEL_INT_EN: Tx FIFO threshold interrupt.</li> <li>■ IR_TX_FIFO_OVER_INT_EN: Tx FIFO overflow interrupt.</li> <li>■ IR_RX_FIFO_FULL_INT_EN: Rx FIFO full interrupt.</li> <li>■ IR_RX_FIFO_LEVEL_INT_EN: Rx FIFO threshold interrupt.</li> <li>■ IR_RX_CNT_OF_INT_EN: Rx counter overflow interrupt.</li> <li>■ IR_RX_FIFO_OF_INT_EN: Rx FIFO overflow interrupt.</li> <li>■ IR_RX_CNT_THR_INT_EN: Rx counter threshold interrupt.</li> <li>■ IR_RX_FIFO_ERROR_INT_EN: Rx FIFO error read interrupt. Trigger when Rx FIFO empty and read Rx FIFO.</li> </ul> </li> <li>● NewState: new state of the specified IR interrupts.                             <ul style="list-style-type: none"> <li>■ ENABLE</li> <li>■ DISABLE</li> </ul> </li> </ul>
Return	N/A

24.3.1.5 IR\_MaskINTConfig

Items	Description
Introduction	Mask or unmask the specified IR interrupt.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● IR_INT: specifies the IR interrupt sources to be mask or unmask. This parameter can be one or combinations of the following values:                             <ul style="list-style-type: none"> <li>■ IR_TX_FIFO_EMPTY_INT_MASK: Tx FIFO empty interrupt mask.</li> <li>■ IR_TX_FIFO_LEVEL_INT_MASK: Tx FIFO threshold interrupt mask.</li> <li>■ IR_TX_FIFO_OVER_INT_MASK: Tx FIFO overflow interrupt mask.</li> <li>■ IR_RX_FIFO_FULL_INT_Msk: Rx FIFO full interrupt mask.</li> <li>■ IR_RX_FIFO_LEVEL_INT_Msk: Rx FIFO threshold interrupt mask.</li> <li>■ IR_RX_CNT_OF_INT_Msk: Rx counter overflow interrupt mask.</li> <li>■ IR_RX_FIFO_OF_INT_Msk: Rx FIFO overflow interrupt mask.</li> <li>■ IR_RX_CNT_THR_INT_Msk: Rx counter threshold interrupt mask.</li> <li>■ IR_RX_FIFO_ERROR_INT_Msk: Rx FIFO error read interrupt mask. Trigger when Rx FIFO empty and read Rx FIFO.</li> </ul> </li> <li>● NewState: new state of the specified IR interrupt mask.                             <ul style="list-style-type: none"> <li>■ ENABLE</li> <li>■ DISABLE</li> </ul> </li> </ul>
Return	N/A

### 24.3.1.6 IR\_GetINTStatus

Items	Description
Introduction	Get the specified IR interrupt status.
Parameters	IRx: selected IR peripheral.
Return	The new state of IR interrupt <ul style="list-style-type: none"> <li>● SET</li> <li>● RESET</li> </ul>

### 24.3.1.7 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 <ul style="list-style-type: none"> <li>● SET</li> <li>● RESET</li> </ul>

### 24.3.1.8 IR\_FSRunning

Items	Description
Introduction	Get the specified IR FSM status.
Parameters	IRx: selected IR peripheral.
Return	The new state of FSM: <ul style="list-style-type: none"> <li>● TRUE: RUN</li> <li>● FALSE: IDLE</li> </ul>

### 24.3.1.9 IR\_ClearINTPendingBit

Items	Description
Introduction	Clears the IR interrupt pending bits.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● IR_CLEAR_INT: specifies the interrupt pending bit to clear. This parameter can be any combination of the following values:                             <ul style="list-style-type: none"> <li>■ IR_TX_FIFO_EMPTY_INT_CLR: Clear Tx FIFO empty interrupt.</li> <li>■ IR_TX_FIFO_LEVEL_INT_CLR: Clear Tx FIFO threshold interrupt.</li> <li>■ IR_TX_FIFO_OVER_INT_CLR: Clear Tx FIFO overflow interrupt.</li> <li>■ IR_RX_FIFO_FULL_INT_CLR: Clear Rx FIFO full interrupt.</li> <li>■ IR_RX_FIFO_LEVEL_INT_CLR: Clear Rx FIFO threshold interrupt.</li> <li>■ IR_RX_CNT_OF_INT_CLR: Clear Rx counter overflow interrupt.</li> <li>■ IR_RX_FIFO_OF_INT_CLR: Clear Rx FIFO overflow interrupt.</li> <li>■ IR_RX_CNT_THR_INT_CLR: Clear Rx counter threshold interrupt.</li> <li>■ IR_RX_FIFO_ERROR_INT_CLR: Clear Rx FIFO error read interrupt. Trigger when Rx FIFO empty and read Rx FIFO.</li> </ul> </li> </ul>
Return	N/A

## 24.3.2 IR Tx APIs

### 24.3.2.1 IR\_SendBuf

Items	Description
Introduction	Send data buffer.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> </ul>

	<ul style="list-style-type: none"> <li>● pBuf: data buffer to send.</li> <li>● len: buffer length.</li> <li>● IsLastPacket: this parameter can be the following values:                             <ul style="list-style-type: none"> <li>■ ENABLE: The last data in IR packet and there is no continuous data. In other words, an infrared data transmission is completed.</li> <li>■ DISABLE: There is data to be transmitted continuously.</li> </ul> </li> </ul>
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 sending data is the length of data buffer, while IR\_SendData() only send one data once.

### 24.3.2.2 IR\_SendData

Items	Description
Introduction	Send one data.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● data: data to send.</li> </ul>
Return	N/A

### 24.3.2.3 IR\_SetTxThreshold

Items	Description
Introduction	Set Tx threshold. When Tx FIFO depth <= threshold value, trigger interrupt.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● thd: Tx threshold.</li> </ul>
Return	N/A

### 24.3.2.4 IR\_GetTxFIFOFreeLen

Items	Description
Introduction	Get free size of Tx FIFO.
Parameters	IRx: selected IR peripheral.
Return	The free size of Tx FIFO.

### 24.3.2.5 IR\_ClearTxFIFO

Items	Description
Introduction	Clear IR Tx FIFO.
Parameters	IRx: selected IR peripheral.
Return	N/A

## 24.3.3 IR Rx APIs

### 24.3.3.1 IR\_ReceiveBuf

Items	Description
Introduction	Read data from Rx FIFO.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● pBuf: buffer address to receive data.</li> <li>● len: read data length.</li> </ul>
Return	N/A

**Note:** the difference between IR\_ReceiveBuf() and IR\_ReceiveData() is that IR\_ReceiveBuf() read a data buffer once and the number of receiving data can be defined by users but shouldn't 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().

### 24.3.3.2 IR\_ReceiveData

Items	Description
Introduction	Read one data.
Parameters	IRx: selected IR peripheral.
Return	Data which read from Rx FIFO.

### 24.3.3.3 IR\_SetRxThreshold

Items	Description
Introduction	Set Rx threshold. When Rx FIFO depth > threshold value, trigger interrupt
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● thd: Rx threshold.</li> </ul>
Return	N/A

### 24.3.3.4 IR\_GetRxDataLen

Items	Description
Introduction	Get data size in Rx FIFO.
Parameters	IRx: selected IR peripheral.
Return	Current data size in Rx FIFO.

### 24.3.3.5 IR\_ClearRxFIFO

Items	Description
Introduction	Clear IR Rx FIFO.
Parameters	IRx: selected IR peripheral.
Return	N/A

### 24.3.3.6 IR\_SetRxCounterThreshold

Items	Description
Introduction	Configure counter threshold value in receiving mode. You can use it to stop receiving IR data.
Parameters	<ul style="list-style-type: none"> <li>● IRx: selected IR peripheral.</li> <li>● IR_RxCntThrType: This parameter can be the following values:                             <ul style="list-style-type: none"> <li>■ IR_RX_Count_Low_Level: Low level counter value &gt;= IR_RxCntThr, trigger IR_INT_RX_CNT_THR interrupt.</li> <li>■ IR_RX_Count_High_Level: High level counter value &gt;= IR_RxCntThr, trigger IR_INT_RX_CNT_THR interrupt.</li> </ul> </li> <li>● IR_RxCntThr: Configure IR Rx counter threshold value which can be 0 to 0x7ffffff.</li> </ul>
Return	N/A

### 24.3.3.7 IR\_StartManualRxTrigger

Items	Description
Introduction	Start trigger only in manual receive mode. <b>Note:</b> manual receive mode is seldom used.
Parameters	IRx: selected IR peripheral.
Return	N/A



## 24.4 IR Usage

### 24.4.1 Sending

#### 24.4.1.1 Tx Polling Mode

To use IR sending function, the following steps are mandatory.

(1) Configure the IR pinmux according to Table 24-1.

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 FIFO. You can refer to section 24.2.1.

#### 24.4.1.2 Special Notes

##### 24.4.1.2.1 Tx FIFO Offset Issue

If you want to judge whether Tx data in FIFO has been sent completely or not, you'd better check Tx FIFO empty flag rather than TX\_FIFO\_OFFSET.

##### 24.4.1.2.2 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, writing some data packets to Tx FIFO. You can refer to step (2) and step (6) in section 24.4.1.1.

## 24.4.2 Receiving

### 24.4.2.1 Rx Interrupt Mode

To use IR receiving function, the following steps are mandatory.

(1) Configure the IR pinmux according to Table 24-1.

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 counter threshold type, Rx counter 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 make 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 24.2.2.
- Waveform inverse issue: in Rx ending, if the waveform is inverse, you should #define INVERSE\_DATA in Ir\_nec\_protocol.h and set IR\_InitStruct.IR\_RxCntThrType = IR\_RX\_COUNT\_HIGH\_LEVEL.

**24.4.2.2 Rx Learning**

The process of Rx learning is similar to common Rx introduced in section 24.4.1.2.1. As shown in Fig 24-3, the difference is that in interrupt handle function, Rx learning should store each pulse of the Rx waveform, while common Rx only needs to store the carrier or un-carrier duration.

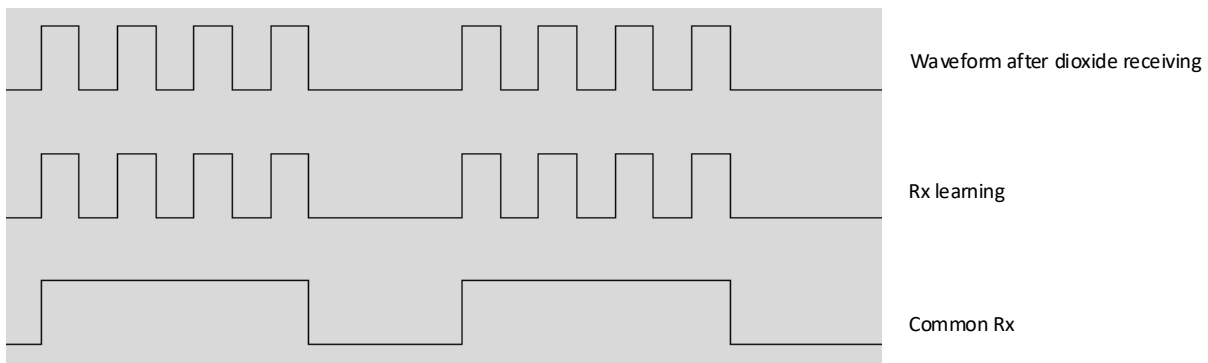


Fig 24-3 Difference of waveform between Rx learning and common Rx

**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 400kHz, hardware may cannot respond to interrupt in time, which will result in decoding carrier frequency failed.

**24.5 Tx Compensation Mechanism**

**24.5.1 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 effective compensation mechanism, the error will increase. So Tx compensation is proposed and used in IR Tx to decrease the error.

In most scenarios, it is not necessary to adopt compensation mechanism. Therefore, if you’re not interested in compensation mechanism, just skip this section. Next we would introduce the application of compensation mechanism.

**24.5.2 Tx Compensation Mechanism Application**

Take NEC application for example, the Tx NEC waveform shown in Fig 24-4 is with  $f_{carrier} = 38kHz$ , and duty = 1/3. The system clock is 100MHz.

For 38kHz carrier frequency:

$$T_{carrier} = \frac{1}{f_{carrier}} = \frac{1}{38K} = 26.31us$$

$$T_{carrier\_duty} = T_{carrier} * duty = 8.77us$$

Compensation frequency  $f_{comp}$  is a dependent clock, you can set  $f_{comp}$  to any value you want. In this example, we set  $f_{comp} = 1\text{MHz}$ , so  $T_{comp} = \frac{1}{f_{comp}} = 1\mu\text{s}$ .

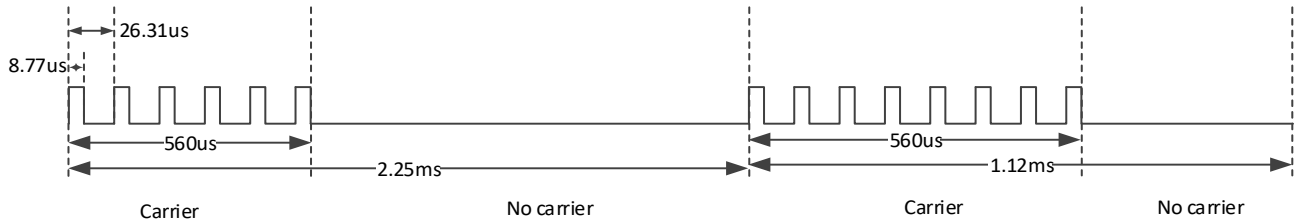


Fig 24-4 Tx NEC waveform

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 24.2.1, cycle = 21, 560us  $\approx 21 * 26.31 + 8.77 = 561.28\mu\text{s}$ .
  - The second no carrier symbol: cycle = (2250 - 560) \* 38/1000 = 64, (2250-560)us = 1690us  $\approx 64 * 26.31 + (26.31 - 8.77) = 1701.38\mu\text{s}$ .
  - The third carrier symbol is the same as the first one.
  - The fourth no carrier symbol: cycle = 21, 560us  $\approx 21 * 26.31 + (26.31 - 8.77) = 570.05\mu\text{s}$ .
- If compensation mechanism is adopted (IR\_TX\_COMPENSATION = 3), the real wave can be calculated as follows.
  - The first carrier symbol: Referring to section 24.2.1, cycle = 21, 560us  $\approx 21 * 26.31 + 8.77 = 561.28\mu\text{s}$ .
  - The second no carrier symbol: cycle = (2250 - 560) \* 1000/1000 = 1690, (2250-560)us = 1690us = 1690 \* 1us.
  - The third carrier symbol is the same as the first one.
  - The fourth no carrier symbol: cycle = 560, 560us = 560 \* 1us.

Compare the above two calculating methods, we can find that by 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).

## 24.6 IR Schematic Design Guideline

### 24.6.1 Leakage

To avoid the leakage problem, we suggest using the IR circuit which is shown in Fig 24-5.

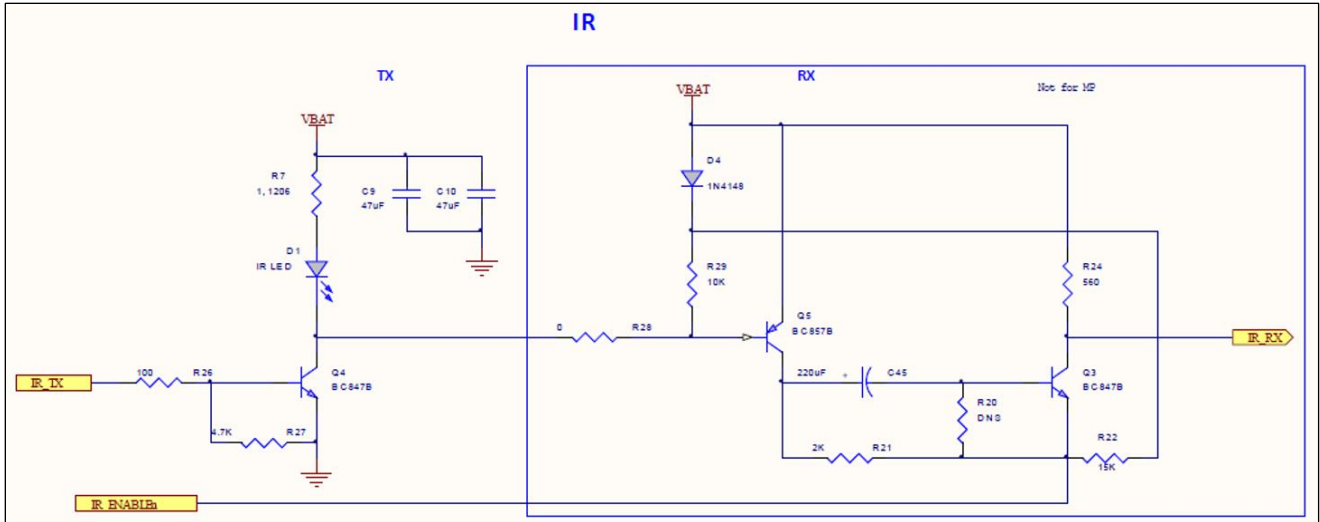


Fig 24-5 Circuit of IR

### 24.6.2 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. In 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 24.6.1 can support the maximum carrier frequency 70kHz, but you can choose the better transistor to acquire higher supported carrier frequency.

# 25 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 25-1 and **Note:** The voltage value may have  $\pm 5\%$  error. Table 25-2.

Table 25-1 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  $\pm 5\%$  error.

Table 25-2 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  $\pm 5\%$  error.

## 25.1 Recommended Threshold Parameter

The recommended threshold parameters are shown in Table 25-3.

Table 25-3 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.

## 25.2 BOD APIs

### 25.2.1 BOR\_ModeSet

Items	Description
Introduction	Choose BOD mode and enable BOD function
Parameters	<ul style="list-style-type: none"> <li>● Option:                             <ul style="list-style-type: none"> <li>■ BOR_RESET: BOD reset mode</li> <li>■ BOR_INTR: BOD interrupt mode</li> </ul> </li> <li>● NewStatus:                             <ul style="list-style-type: none"> <li>■ ENABLE</li> <li>■ DISABLE</li> </ul> </li> </ul>
Return	N/A

### 25.2.2 BOR\_ThresholdSet

Items	Description
Introduction	Set BOD high and low thresholds
Parameters	<ul style="list-style-type: none"> <li>● Thres_Low: BOD low threshold                             <ul style="list-style-type: none"> <li>■ BOR_TH_LOW1</li> <li>■ BOR_TH_LOW2</li> <li>■ BOR_TH_LOW3</li> <li>■ BOR_TH_LOW4</li> <li>■ BOR_TH_LOW5</li> <li>■ BOR_TH_LOW6</li> <li>■ BOR_TH_LOW7</li> </ul> </li> <li>● Thres_High: BOD high threshold                             <ul style="list-style-type: none"> <li>■ BOR_TH_HIGH1</li> <li>■ BOR_TH_HIGH2</li> <li>■ BOR_TH_HIGH3</li> <li>■ BOR_TH_HIGH4</li> <li>■ BOR_TH_HIGH5</li> <li>■ BOR_TH_HIGH6</li> <li>■ BOR_TH_HIGH7</li> </ul> </li> </ul>
Return	N/A

### 25.2.3 BOR\_ClearINT

Items	Description
Introduction	Clear BOD interrupt.
Parameters	N/A
Return	N/A

### 25.2.4 BOR\_DbncSet

Items	Description
Introduction	Set BOD interrupt mode debounce cycle
Parameters	<ul style="list-style-type: none"> <li>● Option:                             <ul style="list-style-type: none"> <li>■ BOR_INTR: BOD interrupt mode</li> </ul> </li> </ul>

	● Dbnc_Value: debounce cycle, in unit of ANA4M clock cycles.
Return	N/A

**Note:** Only BOD interrupt mode can set debounce cycle.

## 26 Flash Translation Layer (FTL)

### 26.1 Overview

NOR-Flash is comprised of blocks, which contains 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 26-1.

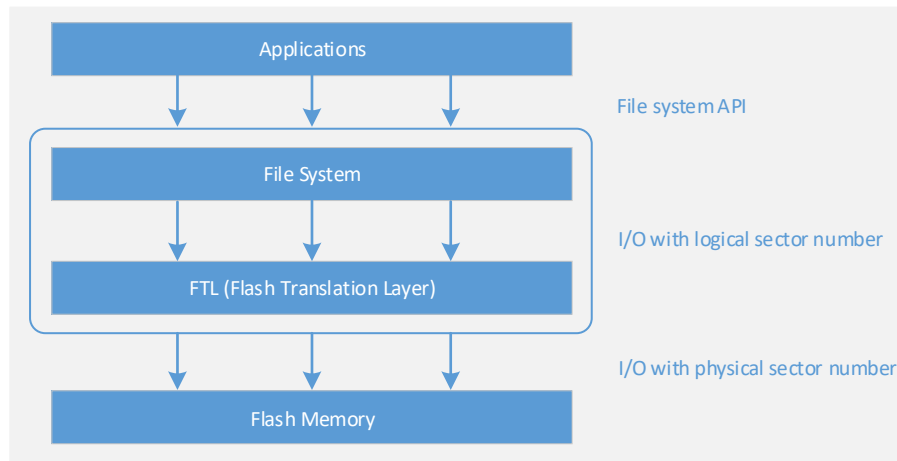


Fig 26-1 Architecture of Flash memory system

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 7-1 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 FTL, users don't need to care about physical map, which is maintained automatically by FTL.



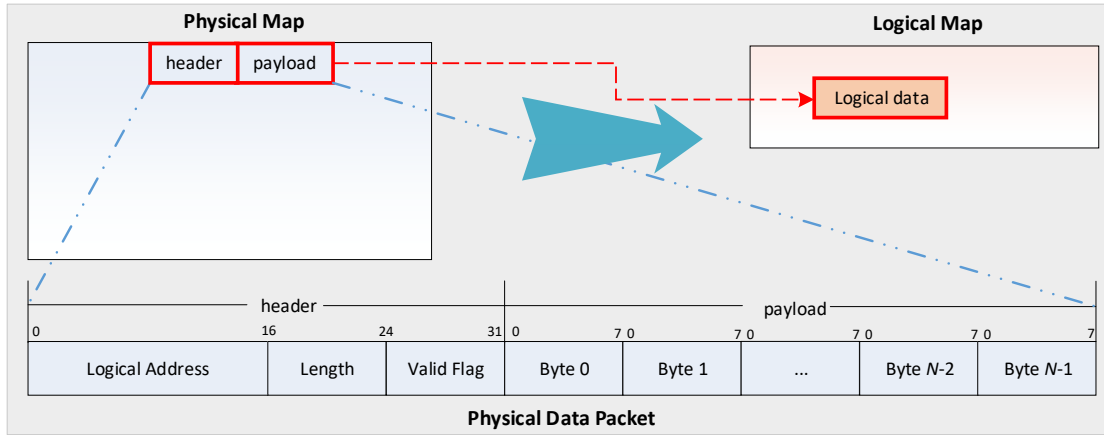


Fig 26-2 FTL overview

## 26.2 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  $((511 * (\text{physical page number} - 1)) - 1) * 4$ .
- Auto Garbage Collection
- Abnormal Power-off Protection

## 26.3 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.

### 26.3.1 ftl\_init

Items	Description
Introduction	Initializes FTL
Parameters	<ul style="list-style-type: none"> <li>● u32PageStartAddr: The start address of physical map</li> <li>● pagenum: The page number of physical map</li> </ul>
Return	N/A

### 26.3.2 ftl\_load\_from\_storage

Items	Description
Introduction	Gets specified length of data from logical map.
Parameters	<ul style="list-style-type: none"> <li>● pdata_tmp: Pointer to a buffer to save logical data</li> <li>● offset: From which address read the logical map</li> <li>● size: The number of bytes to read</li> </ul>
Return	<ul style="list-style-type: none"> <li>● 0: Getting logical map values successfully</li> <li>● Others: Failed to get logical map value</li> </ul>

### 26.3.3 ftl\_save\_to\_storage

Items	Description
Introduction	Writes specified length of data to logical map.
Parameters	<ul style="list-style-type: none"> <li>● pdata_tmp: Pointer to byte array of new logical data</li> <li>● offset: From which address to update the logical map</li> <li>● size: The number of bytes to update</li> </ul>
Return	<ul style="list-style-type: none"> <li>● 0: Writing logical map values successfully</li> <li>● Others: Failed to write logical map value</li> </ul>

## 26.4 How to Use FTL

### 26.4.1 Precautions

When using FTL, you must know the precautions below.

- Theoretically, the logical map area can be extended to 64KB; but the maximum usable logical map area limited by the actual physical space is  $((511 * (\text{physical page number} - 1)) - 1) * 4$ .
- 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 connections' 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 26-1 lists there typical scenarios of BLE, refer to it to set the appropriate offset of application according to your actual situation.

Table 26-1 Examples of recommended setting

Scenario	Calculation	Start address of application		Remark
		Configurable	Recommended	
<ul style="list-style-type: none"> <li>● 3 connections</li> <li>● Less than 50 mesh nodes</li> <li>● 1KB FTL space for application</li> </ul>	$((3 * 256 + 20 * 50) + 1024)$ bytes (< 4084 bytes) are needed totally.	1768 ~ 3060	2500	<ul style="list-style-type: none"> <li>● It is recommended to reserve some memory for system extension. The recommended offset has some reservation for both system and application.</li> <li>● For these two scenarios, 3 physical pages are enough.</li> </ul>
<ul style="list-style-type: none"> <li>● 5 connections</li> <li>● 2KB FTL space for application</li> </ul>	$(5 * 256 + 2048)$ bytes (< 4084 bytes) are needed totally.	1280 ~ 2036	1600	
<ul style="list-style-type: none"> <li>● 5 connections</li> <li>● Less than 60 mesh nodes</li> <li>● 6KB FTL space for application</li> </ul>	$((5 * 256 + 20 * 60) + 6144)$ bytes are needed totally.	-	-	It is not recommended to store so much information in FTL. Try to find another solution, such as DCT.

### 26.4.2 General Steps

To use FTL in Ameba-D, 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 overwrites FTL area 0x0810\_2000~0x0810\_4FFF, you also need to modify the physical page start address. Otherwise, an error would be thrown out to remind you.

```
-o rt18721dhp_intfcfg.o
/cygdrive/e/AmebaD/AmebaD_svn/project/realtek_amebaD_cm4_gcc_verification/asdk/../../../../component/soc/realtek/amebaD/fwlib/usrcfg/rt18721dhp_intfcfg.c:64:2: error: #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. "
#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. "
***
make[4]: *** [/cygdrive/e/AmebaD/AmebaD_svn/project/realtek_amebaD_cm4_gcc_verification/asdk/Makefile.include.gen:449: rt18721dhp_intfcfg.o] Error 1
make[4]: Leaving directory '/cygdrive/e/AmebaD/AmebaD_svn/project/realtek_amebaD_cm4_gcc_verification/asdk/make/target/fwlib'
make[3]: *** [Makefile:19: all] Error 2
```

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.

# 27 Audio Signal Generation and Analysis

## 27.1 Introduction

When developing digital mic (DMIC) related applications, you may need to generate a signal of a certain frequency to test whether the 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) doesn't work well when problems occur.

### 27.1.1 Compilation

You need to add the `example_audio_signal_generate.c` 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) Set the macro named "AUDIO\_SIGNAL\_GENERATE" to 1 in `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).

### 27.1.2 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 command "Audio\_generate tone spkr  $f$ " to generate a signal whose frequency is equal to  $f$  ( $20 < f < 20000$ ).

### 27.1.3 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 command "Audio\_generate dmic fft 1" 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 closed. You can use this command to analyze signals whose frequencies range from 20~20000Hz.

### 27.1.4 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 27-1 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 command "Audio\_generate daad fft  $f$ " 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 closed to the one you generate, it means that the internal codec works well. The problem may be caused by DMIC.

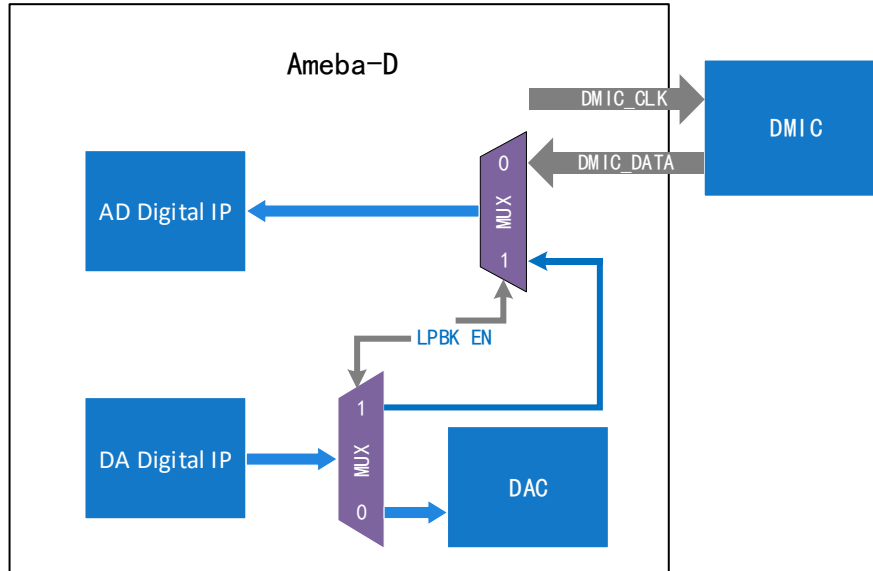


Fig 27-1 DAAD loopback diagram

### 27.1.5 Command

All the commands that can be used are listed in Table 27-1. Each command is composed of several parameters, and the parameters which are surrounded by () should be set by users.

Table 27-1 Available commands in audio signal generation and analysis

Command Format	Command Function	Example
Audio_generate tone spkr ( <i>f</i> )	Generates a tone whose frequency is <i>f</i>	Audio_generate tone spkr 1000
Audio_generate dmic fft 1	Analyzes the signal collected by DMIC	Audio_generate dmic fft 1
Audio_generate daad fft ( <i>f</i> )	DAAD loopback analysis	Audio_generate daad fft 1000
Audio_generate stop	Stops audio test	Audio_generate stop

# 28 Key-Scan

Ameba-D Key-Scan provides up to 6\*6 (36) keypad array with 12 GPIOs, which can be configured for different applications, for example: 5\*7 or 3\*4 keypad arrays. Also multi-key detection and low power mode are supported.

This chapter introduces the APIs of Key-Scan, and how to use Key-Scan.

## 28.1 Pinmux

The pin assignments of Key-Scan are listed in Table 19-4.

Table 28-1 Key-Scan pin assignment

Port Name	Pin Name	QFN48	QFN68	QFN88
PA[12]	KEY_ROW0	Y	Y	Y
PA[13]	KEY_ROW1	Y	Y	Y
PA[14]	KEY_ROW2	Y	Y	Y
PA[15]	KEY_ROW3/KEY_COL6	Y	Y	Y
PA[16]	KEY_ROW4/KEY_COL5	N	Y	Y
PA[17]	KEY_ROW6/KEY_COL3	N	Y	Y
PA[18]	KEY_ROW5/KEY_COL4	N	Y	Y
PA[19]	KEY_COL2	N	Y	Y
PA[20]	KEY_COL7	N	N	Y
PA[21]	KEY_ROW7	N	N	Y
PA[25]	KEY_COL1	Y	Y	Y
PA[26]	KEY_COL0	Y	Y	Y

## 28.2 APIs

### 28.2.1 keyscan\_array\_pinmux

Items	Description
Introduction	Initialization of pinmux settings and pad settings
Parameters	<ul style="list-style-type: none"> <li>col: selected column number depending on KeyColumn (for example: col0 &amp; col2, col is 5)</li> <li>row: selected column number depending on KeyRow</li> </ul>
Return	N/A

### 28.2.2 keyscan\_getdatanum

Items	Description
Introduction	Gets data number of Key-Scan FIFO
Parameters	obj: Key-Scan object defined in application software
Return	Data number of Key-Scan FIFO

### 28.2.3 keyscan\_read

Items	Description
Introduction	Reads data from Key-Scan FIFO
Parameters	<ul style="list-style-type: none"> <li>obj: Key-Scan object defined in application software.</li> <li>pBuf: buffer to save data read from Key-Scan FIFO</li> </ul>

	<ul style="list-style-type: none"> <li>● num: number of data to be read</li> </ul>
Return	N/A

### 28.2.4 keyscan\_init

Items	Description
Introduction	Initializes the Key-Scan device.
Parameters	obj: Key-Scan object defined in application software
Return	N/A

### 28.2.5 keyscan\_set\_irq

Items	Description
Introduction	Enables or disables the specified Key-Scan interrupts mask.
Parameters	<ul style="list-style-type: none"> <li>● obj: Key-Scan object defined in application software.</li> <li>● keyscan_IT: specifies the Key-Scan interrupt sources to be enabled or masked.</li> <li>● newstate: new state of the specified Key-Scan interrupts mask.</li> </ul>
Return	N/A

### 28.2.6 keyscan\_clear\_irq

Items	Description
Introduction	Clears the specified Key-Scan interrupt pending bit.
Parameters	<ul style="list-style-type: none"> <li>● obj: Key-Scan object defined in application software.</li> <li>● keyscan_IT: specifies the Key-Scan interrupt to be cleared.</li> </ul>
Return	N/A

### 28.2.7 keyscan\_get\_irq\_status

Items	Description
Introduction	Gets Key-Scan interrupt status.
Parameters	obj: Key-Scan object defined in application software.
Return	Interrupt status

### 28.2.8 keyscan\_set\_irq\_handler

Items	Description
Introduction	Sets Key-Scan interrupt handler.
Parameters	<ul style="list-style-type: none"> <li>● obj: Key-Scan object defined in application software.</li> <li>● func: the interrupt handler function.</li> </ul>
Return	N/A

### 28.2.9 keyscan\_enable

Items	Description
Introduction	Enables the specified Key-Scan peripheral
Parameters	obj: Key-Scan object defined in application software.
Return	N/A

### 28.2.10 keyscan\_disable

Items	Description
Introduction	Disables the specified Key-Scan peripheral
Parameters	obj: Key-Scan object defined in application software.
Return	N/A

### 28.2.11 keyscan\_clearfifodata

Items	Description
Introduction	Clears the FIFO data
Parameters	obj: Key-Scan object defined in application software.
Return	N/A

## 28.3 Key-Scan Usage

There are two ways to use the Key-Scan: using default settings or using APIs listed in 28.2.

### 28.3.1 Using Default Configuration

When using default settings, only the keypad can be configured.

The Key-Scan initialization is implemented by the function `app_keyscan_init()` in `main.c`. User can follow these steps to use it.

- (1) Set the `km0_enable_key_touch` in `ps_config` (`Rtl8721dlp_sleepcfg.c`) to `BIT_KEY_ENABLE` to enable Key-Scan when KMO boots.

```
PSCFG_TypeDef ps_config = {
    .km0_config_wifi_enable = TRUE,
    .km0_enable_key_touch = FALSE, //BIT_KEY_ENABLE | BIT_CAPTOUCH_ENABLE,
    .km0_tickers_debug = TRUE, // * 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) Configure the GPIOs in the following structure (`touch_key.c`), and set valid to 1 for the pins which will be used in keypad.

```
static KeyMatrix_TypeDef KeyRow[8] =
{
    /* pinmux, valid */
    {_PA_12, 1}, /* row 0 */
    {_PA_13, 1}, /* row 1 */
    {_PA_14, 1}, /* row 2 */
    {_PA_15, 1}, /* row 3 */
    {_PA_16, 1}, /* row 4 */
    {_PA_18, 0}, /* row 5 */
    {_PA_17, 0}, /* row 6 */
    {_PA_21, 0}, /* row 7 */
};

static KeyMatrix_TypeDef KeyColumn[8] =
{
    /* pinmux, valid */
    {_PA_26, 1}, /* Col 0 */
    {_PA_25, 1}, /* Col 1 */
    {_PA_19, 1}, /* Col 2 */
    {_PA_17, 1}, /* Col 3 */
    {_PA_18, 1}, /* Col 4 */
    {_PA_16, 0}, /* Col 5 */
    {_PA_15, 0}, /* Col 6 */
    {_PA_20, 1}, /* Col 7 */
};
```

- (3) Rebuild SDK and re-burn images, the Key-Scan would work after boot up.



## 28.3.2 Using APIs

When using APIs, more parameters can be configured.

- (1) A Key-Scan object must be defined in application software first, then the necessary parameters must be configured. For the parameter row, if row 0, row 2 and row 5 will be used, row must be set to 0x25 (100101), the same as parameter col. For the parameter clk, scan clock=bus clock/(clk+1).

```
struct keyscan_s {  
    u32 row;  
    u32 col;  
    u32 clk;  
    u32 workmode; //0 for regular scan mode, 1 for event trigger mode  
    u32 keylimit;  
    u32 overctrl; //0 for discard new, 1 for discard oldest  
};
```

- (2) Use `keyscan_init()` in `keyscan_api.c` to initialize Key-Scan.
- (3) Use `keyscan_set_irq()` in `keyscan_api.c` to enable the interrupt wanted.
- (4) Use `keyscan_set_irq_handler()` in `keyscan_api.c` to handle the triggered interrupt.
- (5) Use `keyscan_enable()` in `keyscan_api.c` to enable the Key-Scan. Then the Key-Scan would work.

# 29 Bluetooth

## 29.1 Overview

Bluetooth (BT) is one of the most popular short-range wireless communication technologies. The lower power feature of Bluetooth is redesigned and greatly enhanced since Bluetooth 4.0. The latest version of Bluetooth Low Energy (BLE) core specification is 5.2. BLE enables short-distance data exchange between fixed devices and mobile devices, and aims to build personal area networks (PAN) for its outstanding low power feature.

In Ameba-D, BLE shares WLAN's Radio Frequency (RF) circuit and antenna to transmit and receive data because BLE does not have a separate radio system. Users must ensure that WLAN is initialized before BLE's initialization, and BLE is de-initialized before WLAN's de-initialization.

When BLE is enabled for the first time, there is a compile error to remind users to configure the start address of Flash Translation layer (FTL). Users need to ensure that this area does not overlap with other areas.

Two ways of debug log are offered for BLE: the up level debug log and the low level debug log. The up level log is steaming out from PA[26], and the low level log is steaming out from PA[16]. Fortunately, these two GPIOs only work as UART pins when BLE log is enabled, so they can be used as normal GPIOs, only if a 0 ohm resistance is placed between the GPIOs and the peripheral circuit.

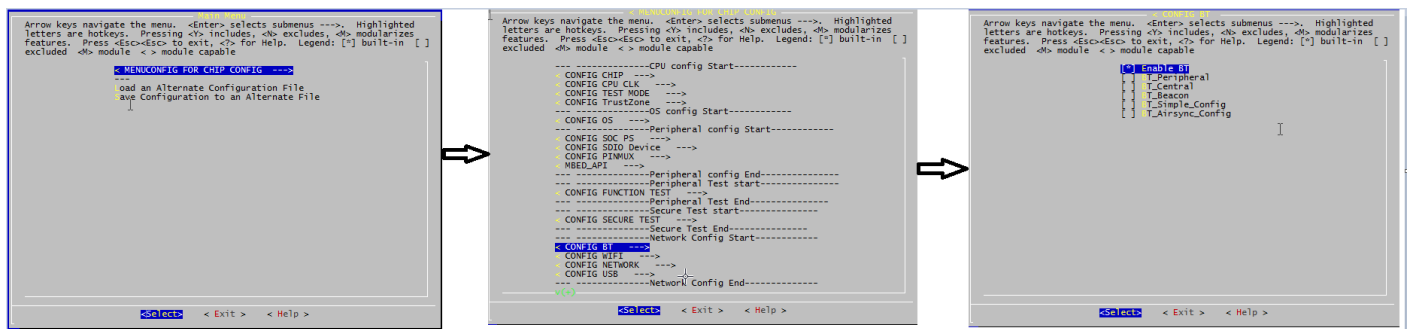
## 29.2 BT Examples

Ameba-D BT examples provide a set of BT functionality, such as BT Config, BT Peripheral, BT Central and BT Beacon. For more information, refer to *UM0201 Ameba Common BT Application User Manual EN.pdf*.

This section illustrates how to build and run BT examples in our SDK, including GCC and IAR environment.

### 29.2.1 GCC Project

- Enter SDK path: `project/realtek_amebaD_va0_example/GCC-RELEASE/project_hp`, input command `make menuconfig` to enable BT.



- Allocate FTL physical map by setting `FTL_MEM_CUSTEM` to 1 and setting FTL start address in `rtl8721dhp_intfcfg.c`.

```
#if defined(CONFIG_FTL_ENABLED)
#define FTL_MEM_CUSTEM 1
#else
#error "You should allocate flash sectors to for FTL physical map as following, then set FTL_MEM_CUSTEM to 1. For more information, refer to the user manual."
#endif

const u8 ftl_phy_page_num = 3; /* The number of physical map pages,
const u32 ftl_phy_page_start_addr = 0x00102000; /* The start offset of flash pages which
Users should modify it according to the user manual.*/

#endif
#endif
```

- Build images and use ImageTool to download images to your board.

**Note:** You can select one BLE example or all the examples at once. If all the examples are selected at once, the integrated image can support all BLE test commands, and the scatternet configuration will display only when both peripheral and central are selected.

## 29.2.2 IAR Project

- Enter SDK path: `project/realtek_amebaD_va0_example/inc/inc_hp`, edit `platform_autoconf.h` to enable BT by defining the corresponding macro to 1.

```

/*
 * < CONFIG BT
 */
#define CONFIG_BT_EN 1
#define CONFIG_BT 1
#define CONFIG_BT 1
#define CONFIG_BT_PERIPHERAL 1
#define CONFIG_BT_CENTRAL 1
#define CONFIG_BT_SCATTERNET 1
#define CONFIG_BT_BEACON 1
#define CONFIG_BT_CONFIG 1
#define CONFIG_BT_AIRSYNC_CONFIG 1
    
```

- Allocate FTL physical map by setting `FTL_MEM_CUSTEM` to 1 and setting FTL start address in `rtl8721dhp_intfcfg.c`.

```

#if defined(CONFIG_FTL_ENABLED)
#define FTL_MEM_CUSTEM 1
#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 the user manual."
#else
const u8 ftl_phy_page_num = 3; /* The number of physical map pages,
const u32 ftl_phy_page_start_addr = 0x00102000; /* The start offset of flash pages which
Users should modify it according to the user manual."
#endif
#endif
    
```

- Build image and use ImageTool to download images to your board.

**Note:** You can select one BLE example or all the examples at once. If all the examples are selected at once, the integrated image can support all BLE test commands, and the scatternet configuration will display only when both peripheral and central are selected.

## 29.3 BT Debug

### 29.3.1 Introduction

This section focuses on how to use log tools to capture log files. When user wants to debug an issue of BT SDK, four log files are needed, as shown in Table 29-1.

Table 29-1 Log files

File	File type	Description
FW LOG	RawData_XXXX.log	BT controller log
BT TRACE LOG	.cfa & .bin	BT host log
LOGUART LOG	.txt	System log
APP.trace	.trace	The escaped index of BT host log

### 29.3.2 Hardware & Software Preparation

If user wants to capture log files to debug, the needed hardware and software are listed below.

#### 29.3.2.1 Hardware

When using BT function, along with using the following GPIOs at the same time, it is best to series a 0 ohm resistor between these GPIOs and peripherals, so that you can easily pull out a test point when debugging BT. For more information, refer to the design of HDK.

Package	Log	Hardware pin	Default baud rate
---------	-----	--------------	-------------------

QFN48 (RTL8720xx)	FW LOG	PA[15]	115200
	BT TARCE LOG	PA[26]	150000
QFN68 (RTL8721xx)/QFN88 (RTL8722xx)	FW LOG	PA[16]	115200
	BT TARCE LOG	PA[26]	150000

### 29.3.2.2 Software

Name	Path	Description
BTDebugger.exe	\tools\bluetooth\BTFWDebugger	Used to capture the FW LOG
DebugAnalyzer	\tools\bluetooth\DebugAnalyzer	Used to capture the BT TRACE LOG
SecureCRT	-	Terminal emulator supporting SSH (SSH1 and SSH2), used to capture the LOGUART LOG
UartAssist	-	Serial port debugging tool, used to check that whether the log can be captured

### 29.3.3 Prerequisite – Opening the Log Switch

The log switch is closed by default. You need to open it when debugging the BT issue.

The debug bit is used to control the log switch. The default value of all bits are 1.

- 1: The log switch is closed.
- 0: The log switch is open.

Bit	Description	Comment
Bit[1]	The switch of FW LOG	For Ameba-D, the debug bit is located in the address of 0x8003028 in Flash. The value of the word is 0x9d, that is to say, bit[1], bit[5] and bit[6] are all 0. You can use the tool shown in Fig 29-1 to change the word value to 0x9d.
Bit[5]	The switch of BT TRACE LOG	
Bit[6]	The switch of DRIVER debug	

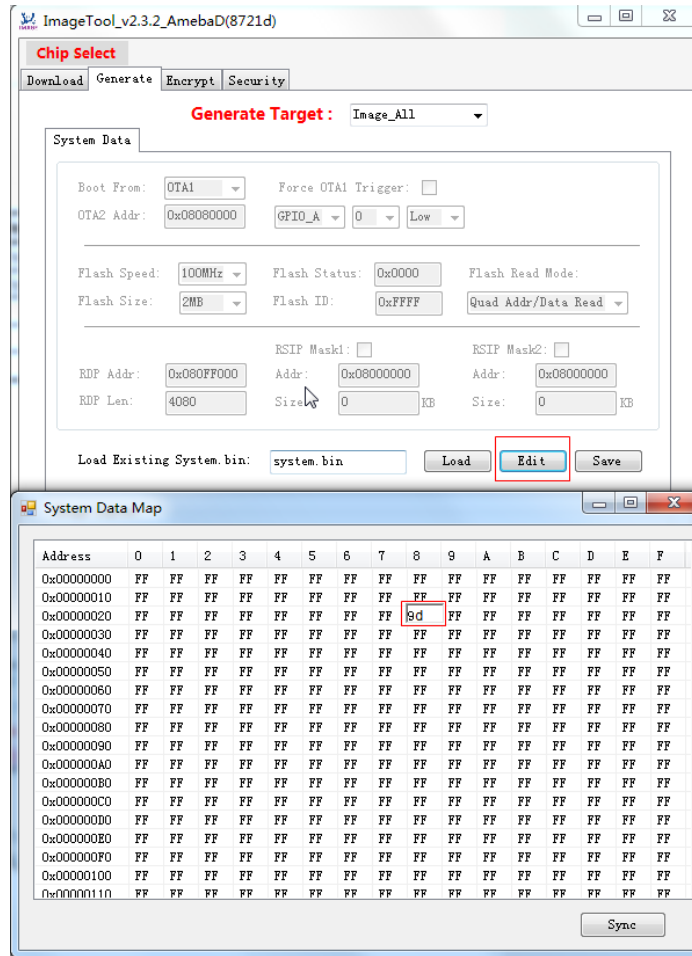


Fig 29-1 Changing the world value in Flash

**Note:**

- The debug bit can be written only once. You must erase the sector when writing the debug bit again, otherwise hardfault occurs.
- The debug bit is used to debug only. If you want to exit the debug mode, you need to erase the Flash.

### 29.3.4 Capturing Log Files

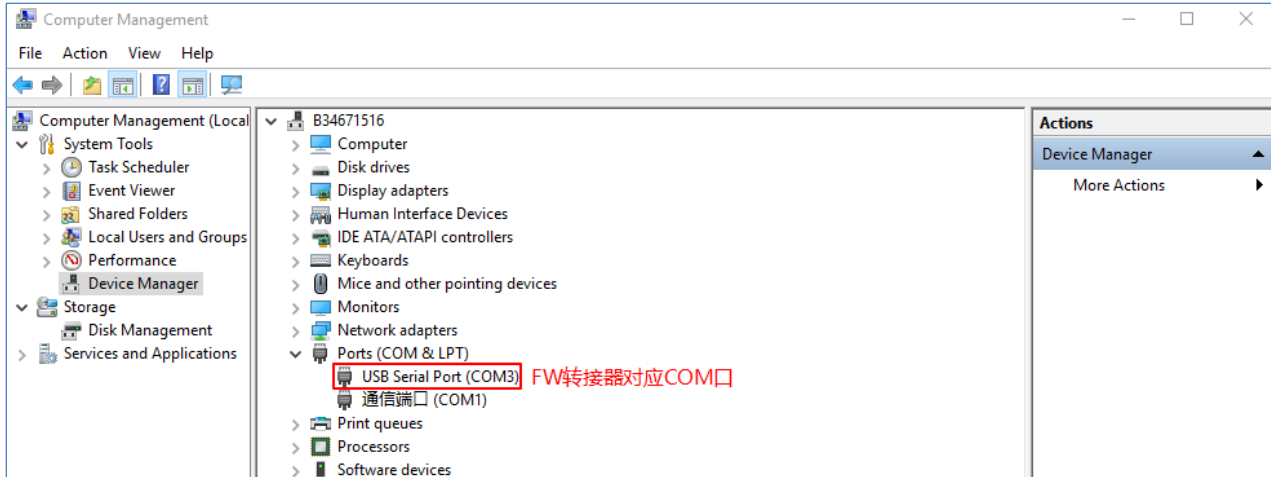
As mentioned above, four log files should be captured when debugging, the operations are described below.

#### 29.3.4.1 FW LOG

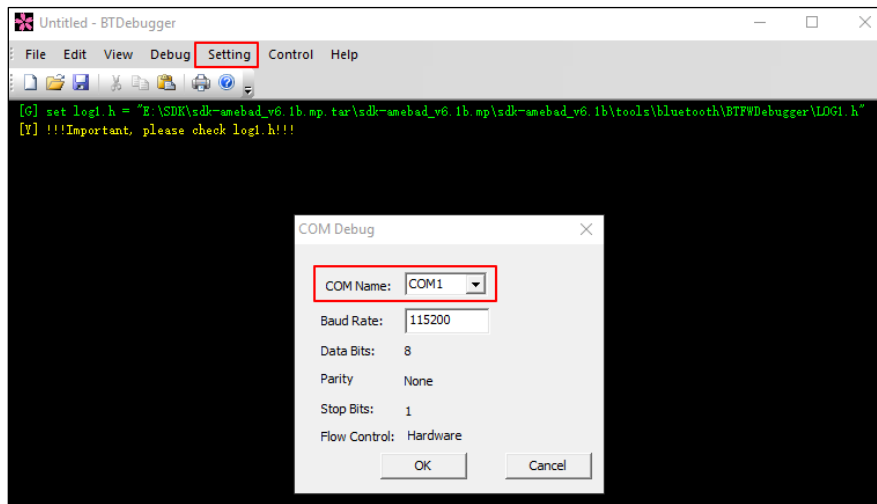
The FW LOG pin for Ameba-D is PA[15] or PA[16]. This is a hardware pin, you can't change it. So remember to reserve the pin when debugging the BT issue.

Follow the steps below to capture the FW LOG:

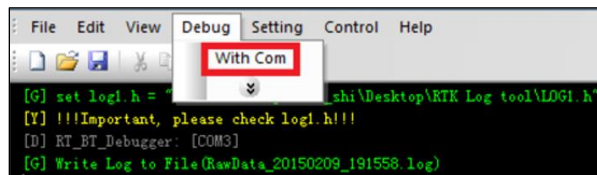
- (1) Open the Device Manager of your PC to check the number of the UART COM (e.g. COM3).



(2) Launch the software – BTDebugger.exe, then click **Setting** > **Com Setting** to choose the corresponding COM of UART.



(3) Click **Debug** > **With Com** to start the capture of FW LOG.



If the log capture is successful, you can see that the FW LOG (RawData\*.log) under the tool directory is constantly increasing during the Bluetooth operation. Once the file size is found to be constant even though you refresh or operate the Bluetooth, re-check the above steps.

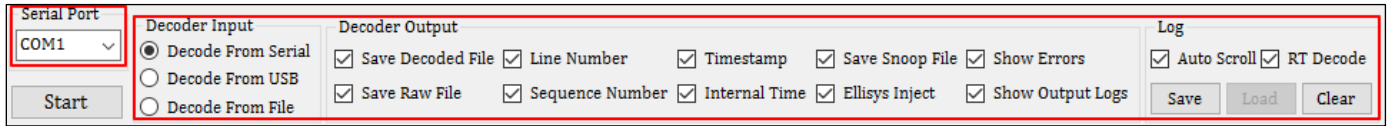
BTDebugger.exe	2016/7/22 16:15	应用程序	3,836 KB
RawData_20190416_111955.log	2019/4/16 11:21	文本文档	3 KB

### 29.3.4.2 BT TRACE LOG

The BT TRACE LOG pin for Ameba-D is PA[26]. This pin shares the software pin of UART, it can be changed to another pin according to the Datasheet.

Follow the steps below to capture the BT TRACE LOG:

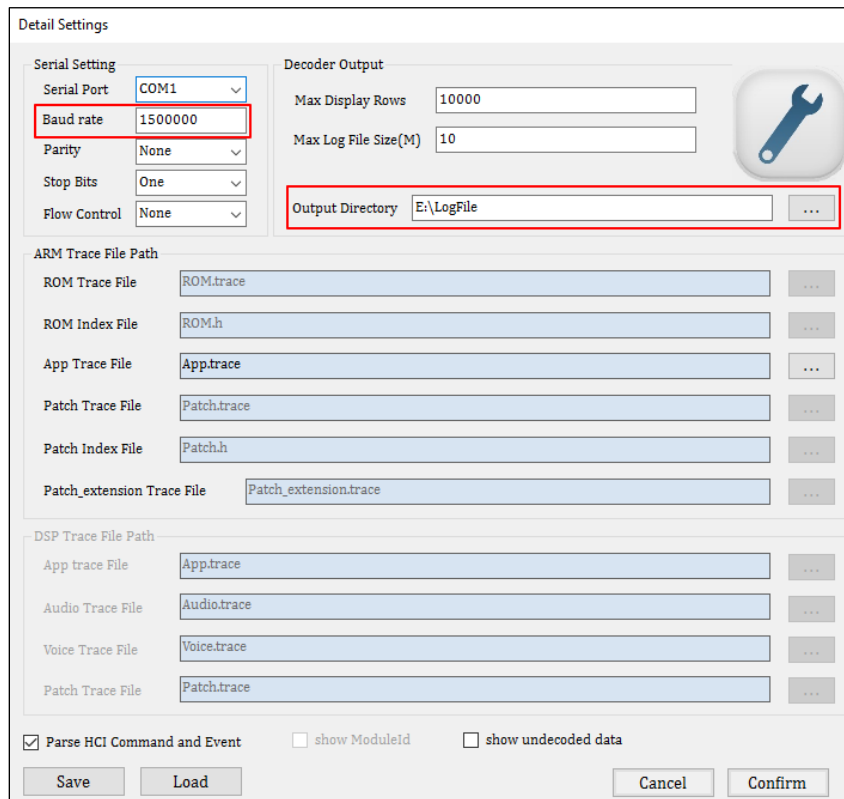
- (1) Open the Device Manager of your PC to check the number of the UART COM.
- (2) Launch the software – DebugAnalyzer.exe, then choose the corresponding COM of Serial Port, and configure the items.



- (3) Click **Settings** to change the details of settings.

**Note:**

- For Ameba-D, the baud rate is 1500000.
- The output directory of BT TRACE LOG can be defined by yourself.



- (4) Click **Start** to start the capture of BT TRACE LOG.
- (5) Reset the device, and look over the log.

```

00009 04-16#14:23:38.916 008 01765 [APP] !**gaps_add_client: client ID = 0
00010 04-16#14:23:38.916 009 01765 [APP] !**tp_client_add_client: client ID = 1
00011 04-16#14:23:38.917 010 01765 [APP] !**vendor_pxpect_client_add: client ID = 2
00012 04-16#14:23:39.414 151 02247 [APP] !**profile callback PROFILE_EVT_SRV_REG_COMPLETE result
00013 04-16#14:23:39.416 154 02247 [APP] !**app_handle_dev_state_evt: init state 1 scan state 0 adv state 0 conn state 0
00014 04-16#14:23:39.416 155 02247 [APP] !**GAP stack ready
00001 2019-04-16#14:29:32.913 Realtek Semiconductor Corporation <-> File Version [2.1.8.2]
00001 2019-04-16#14:30:21.613 Realtek Semiconductor Corporation <-> File Version [2.1.8.2]
    
```

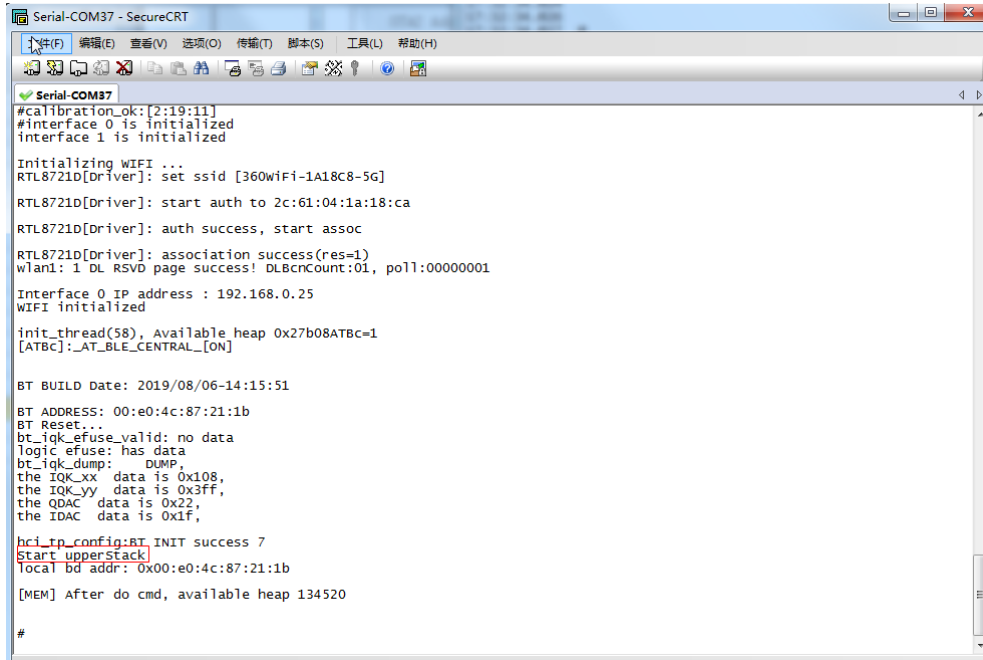
The log files located in the LogFile folder are COM\*.cfa and COM\*.bin, select them according to the operation time. If the COM\*.bin file is constantly increasing, the log may be right.

	COM14_btsnoop_2019-04-16_14-33-14.cfa	2019/4/16 14:36	Protocol Analyze...	1 KB
	COM14_2019-04-16_14-33-14.bin	2019/4/16 14:36	UltraEdit Docum...	5 KB
	COM14_2019-04-16_14-33-24.log	2019/4/16 14:36	文本文档	1 KB

### 29.3.4.3 LOGUART LOG

The LOGUART LOG can be seen in UART log tools such as SecureCRT. If the hardware level initialization has been done successfully, the log of “Start upperStack” will be printed.

The LOGUART LOG can be stored in a \*.txt file.



### 29.3.4.4 APP.trace

APP.trace is for BT TRACE LOG. For BT debug, App.trace must be offered.

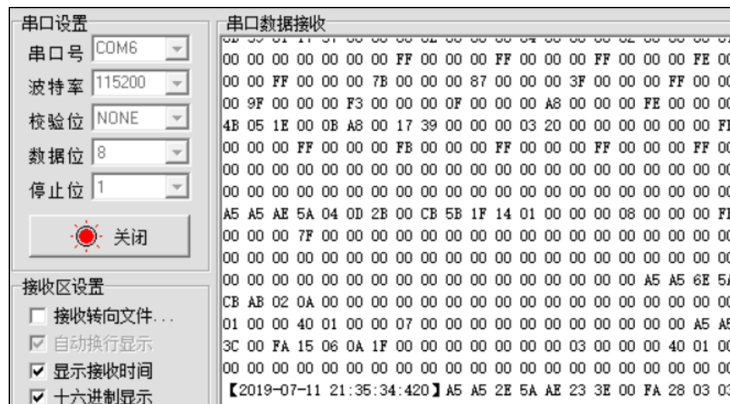
This file is located in project\\*\asdk\image or tools\bluetooth\DebugAnalyzer after building the project.

### 29.3.5 FAQ

Q: How to know quickly that whether the log can be captured?

A: UartAssist.exe tool can be used to check. Launch the UartAssist.exe, then select the corresponding serial port and baud rate.

- FW LOG: If you can receive a data packet in UartAssist every 10 seconds, the FW LOG can be captured.





- BT TRACE LOG: If you can see a data packet starting with "7E" in UartAssist when resetting the device every time, the BT TRACE LOG can be captured.



# Revision History

Date	Version	Change
2021/8/3	v2	Updated the chapter Liquid Crystal Display Controller (LCDC): 16-bit mode is not supported.
2021/5/25	v1	Released version for open-source