ASR582X 系列

开发入门指南

文档版本 1.0.1

发布日期 2023-09-26

关于本文档

本文档旨在指导用户快速入门，了解 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片的开发。

读者对象

本文档主要适用于以下工程师：

- 软件工程师
- 技术支持工程师

产品名称

本文档适用于 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片。

版权公告

商标声明

防静电警告

静电放电（ESD）可能会损坏本产品。使用本产品进行操作时，须小心进行静电防护，避免静电损坏产品。

免责声明

翱捷科技股份有限公司对本文档内容不做任何形式的保证，并会对本文档内容或本文中介绍的产品进行不定期更新。

本文档仅作为使用指导，本文的所有内容不构成任何形式的担保。本文档中的信息如有变更，恕不另行通知。

本文档不负任何责任，包括使用本文档中的信息所产生的侵犯任何专有权行为的责任。

翱捷科技股份有限公司

地址：上海市浦东新区科苑路399号张江创新园10号楼9楼　邮编：201203

官网：http://www.asrmicro.com/

文档修订历史

日期	版本号	发布说明
2022.12	1.0.0	首次发布。
2023.09	1.0.1	修改 SEL 引脚描述；更新了表 3-4。

目录

表格

插图

1. 概述

本文档旨在帮助开发者快速了解 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片，包括搭建开发环境、了解芯片相关资源、固件烧录、OTA 以及产测等功能。

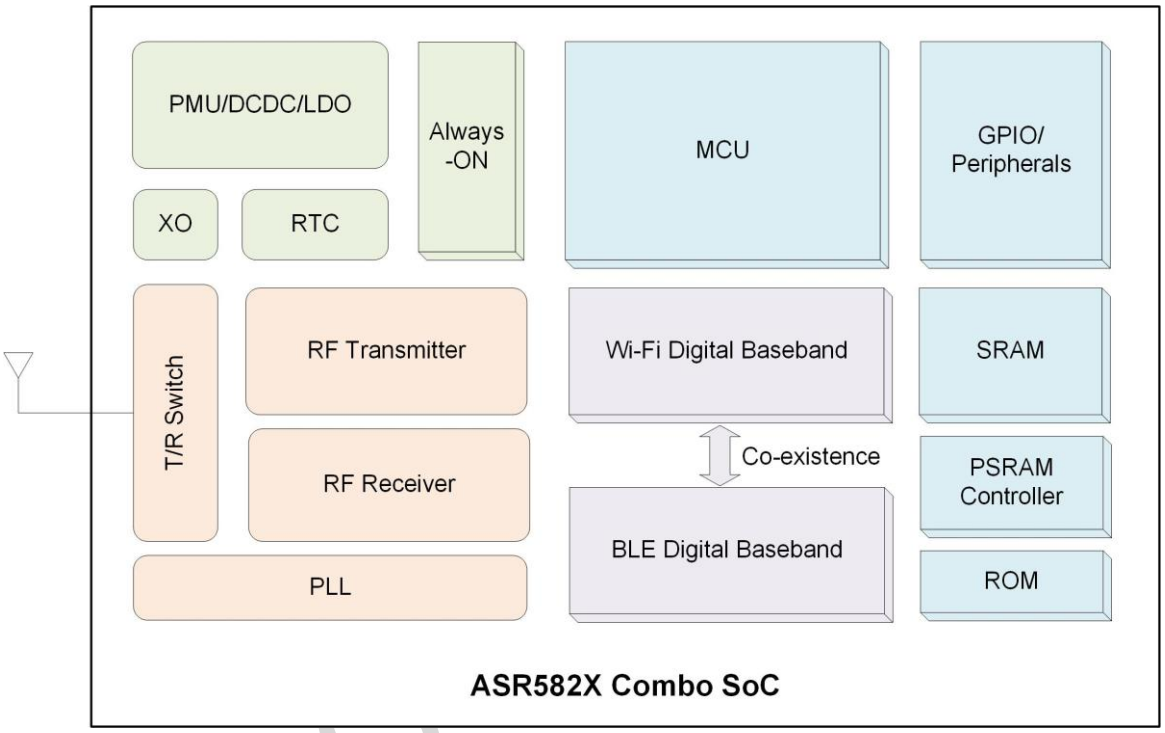

图 1-1 芯片框图

2. 平台相关

2.1 简介

ASR582X Software Development Kit（SDK）是 ASR 为开发者提供的 Wi-Fi/BLE Combo 物联网平台开发套件，目前提供 FreeRTOS、AliOS-Things 和 HarmonyOS 等主流 RTOS 开发平台的 SDK;

SDK 为开发者提供了一系列丰富的模块和示例，能够满足开发者各种物联网应用的开发，包括 Wi-Fi/BLE 功能应用/Mesh 功能应用以及各种外设接口驱动程序。

ASR 提供的 SDK 各部分 API 以不同的前缀统一命名，其中：

- 系统和 Wi-Fi 部分 API 统一以 **lega** 为前缀。
- 蓝牙部分 API 统一以 **sonata** 为前缀。
- 外设驱动部分统一以 **duet** 为前缀。

2.2 开发环境搭建

各平台 SDK 目录介绍以及开发环境的搭建，请参考文档《ASR582X 系列_开发环境搭建指南》。

2.3 开发板

ASR 综合客户需求、使用习惯专门设计的一款小巧便携式 Lite 开发板，极简的设计仍具备完整的功能，满足客户开发，如下图所示：

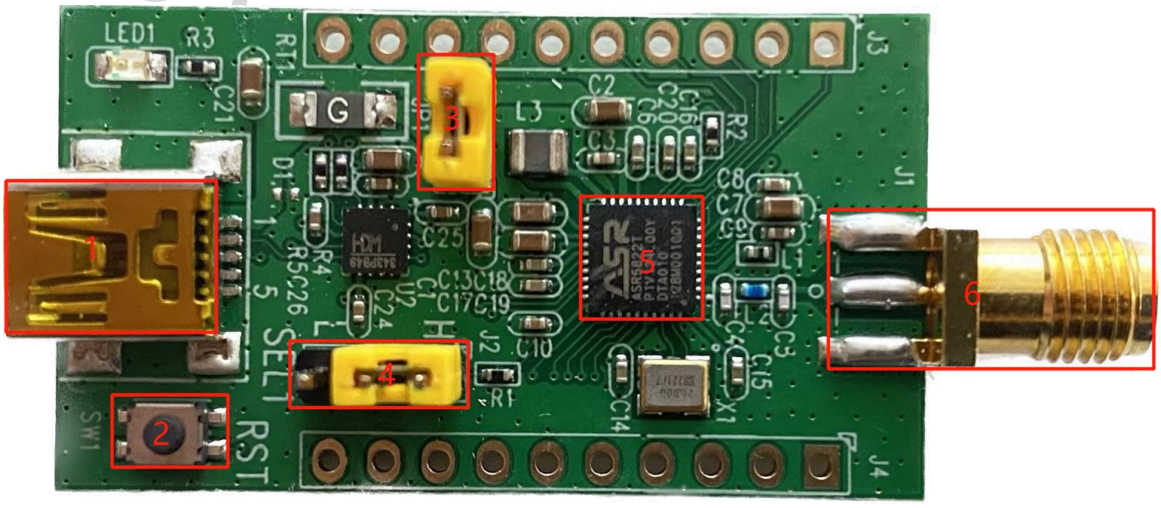

图 2-1 Lite 开发板

表 2-1 Lite 开发板功能描述

序号	功能
1	Mini B 接口（供电、烧录以及调试信息打印串口）
2	SW1 复位按键
3	JP1 跳线帽（电流测试，正常使用时必须跳线连接）
4	J2 跳线帽（SEL1 pin 接 H 端为烧录模式，接 L 端为运行模式）
5	ASR582X 芯片
6	天线，注意实际使用 Wi-Fi 功能要带上天线，否则射频功能会出现异常

用户使用开发板烧录固件的步骤如下:

1. 使用 mini USB 连接线通过开发板的 mini B 接口连接至 PC 端。
2. PC 端打开 Dogo 工具，选择对应的端口号，波特率为 115200 bps，芯片类型选择 582x。
3. 图 2-1 中红框 4 中的 J2 跳线帽接到 H 端。
4. 按下图 2-1 中红框 2 中 SW1 复位键，Dogo 工具界面打印“1F2E3D00”，表示已进入 UART 下载模式。
5. 下载完成后，将红框 4 中的 J2 跳线帽接到 L 端，按下复位键，正常会有 Dogo 工具运行 log 打印。
6. 开发板的详细使用请参考文档《ASR582X 系列_开发板使用指南》。

2.4 固件及烧录

2.4.1 固件

ASR582X 的固件以及功能如下:

- ***bootloader.bin***

引导程序，由 ASR 提供，log 输出口 UART1。

- ***app.bin***

由开发者在 SDK 上开发产生的应用程序 bin 文件，默认 log 输出口 UART1，软件内可配置为其他 UART 接口。

- ***ate.bin***

RF 产测校准时使用（需配合相关仪器使用），由 ASR 提供，默认 log 输出口 UART1，软件可配置。

➢ ***app.ota.bin***

和 app.bin 一样，由用户开发产生，用于 OTA 升级的 bin 文件，一般和 app.bin 放在同一目录，具体生成和使用请参考 3.10 OTA 升级章节。

bootloader/ate 由 ASR 提供。

- AliOS 平台固件目录为：`platform\mcu\asr5821\tools`
- FreeRTOS 平台固件目录为：`tools\factory_bin\ASR582X`
- HarmonyOS 平台固件目录为：`device\asr\asr582x\tools\factory_gen\factory_bin_files`

> ⚠ **注意：**
>
> *若在对应目下未找到 bin 档，请联系对应的代理商和 ASR，另外由于固件更新与 SDK 的 release 存在不同步的问题，需要获取最新固件，也可与代理商或 ASR 联系。*

2.4.2 固件烧录

1. ASR582X 系列芯片通过串口（固定 UART1，不可更改）将固件烧录到 Flash 中。
2. 固件烧录前需要将芯片 SEL1 脚拉高（3.3 V）后重启芯片进入烧录模式。
3. 只需要将 bootloader.bin & app.bin 烧入到 Flash 中系统即可正常运行。
4. ASR 提供 PC 端 Dogo 工具来实现固件烧录以及串口调试。
5. 一般情况下 UART1 口也做程序打印 log 的调试口。

- 在开发过程中使用开发板进行固件烧录，可以参考 2.3 开发板章节以及文档《ASR582X 系列_开发板使用指南》。
- 在量产环节进行固件批量烧录，可以参考 4.1 烧录章节以及文档《ASR IoT 芯片_MP_Pro 量产烧录工具使用说明》。
- Dogo 工具使用，请参考手册：《ASR IoT 芯片_DOGO 烧录调试工具》。

2.5 启动流程

2.5.1 工作模式选择

芯片上电或复位后，通过 SEL 引脚的电平组合进入的不同工作模式，包括 UART 烧录模式和 Flash 启动模式，具体电平状态见 2.5.2 SEL 引脚模式章节；芯片进入 Flash 启动模式后，首先会进入 Bootloader，通过对应的操作运行不同的固件程序，具体固件程序见 2.5.3 固件工作模式章节。

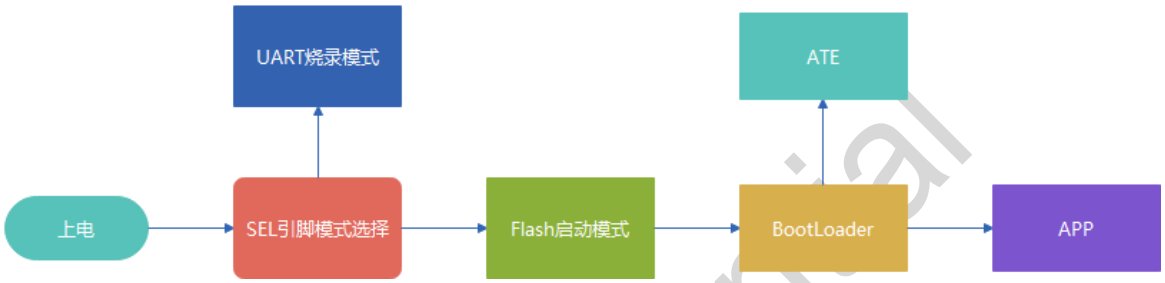

图 2-2 工作模式选择

2.5.2 SEL 引脚模式

SEL 引脚模式主要通过不同 SEL 引脚的电平组合来实现的，目前 ASR 提供以下几种工作模式：

- UART 烧录模式：从 UART1 下载固件到 Flash 中
- Flash 启动模式：引导运行 Flash 中烧录的代码。

表 2-2 SEL 引脚模式

硬件工作模式	SEL3 (PAD10)	DSEL2 (PAD15)	SEL1 (PAD14)
UART 烧录模式	0	0	1
Flash 启动模式	0	0	0

⚠ **注意：**

1. *SEL 所有引脚的高电平均为 3.3 V，不能输入 5 V 否则会损坏引脚。*
2. *上表中的数字表示上电时引脚的电平状态，0 为低电平，1 为高电平。*
3. *上述三个 SEL 引脚上电默认是低电平。*
4. *拉高 SEL1，其余引脚保持低电平，芯片上电复位后进入烧录模式。*

2.5.3 固件工作模式

固件工作模式由烧录的 Bootloader 固件来决定，目前 ASR 提供的 Bootloader 固件主要提供以下两种工作模式：

- APP 模式：正常运行用户的应用固件。
- ATE 模式：射频测试模式，主要用来对射频发射功率、接收灵敏度进行测试和校准射频参数；需要配合相关测试仪器。

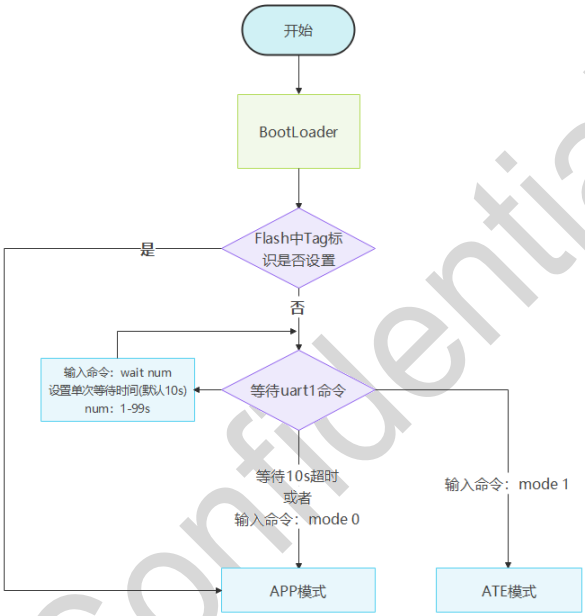

图 2-3 系统启动流程

如上图所示，系统上电或者 Reset 后进入 Bootloader 模式，Bootloader 会读取 Flash 中指定区域的 tag 标识是否被置起：

1. 如果被置起则直接跳到 APP 模式执行 app 应用；
2. 如果 tag 未被置起来则会等待串口（默认为 UART1）命令输入：
   - 无命令输入，等待 10 秒后超时，或者输入命令：mode 0，则也会跳到 app 模式。
   - 如果等待命令输入过程输入命令：mode 1，则跳转到 ATE 模式进行产测校准的工作。

注意：

1. *tag 标志被设置表示芯片已经进行产测，正常流程为产测完成并通过后，由产测工具触发将 tag 标志置起。*
2. *开发阶段，未进行产测又不想等待 10 秒或每次输入 mode 0 的命令，则可以在等待 UART1 命令时串口输入“boot_mode_set”命令，手动把 tag 置起，下次启动则直接跳到 app 模式（由于未进行产测 RF 校准，所以不能拿此版本来测量 RF 性能指标）。*

3. 软件资源

3.1 存储器简介

表 3-1 ASR582X 存储器

芯片型号	RAM	FLASH	OTP
ASR5822N	256 KB	2 MB	512 Bytes
ASR5822C	256 KB	2 MB	512 Bytes
ASR5822S	352 KB	2 MB	512 Bytes
ASR5822T	352 KB	4 MB	512 Bytes

下图为芯片内部 memory layout:

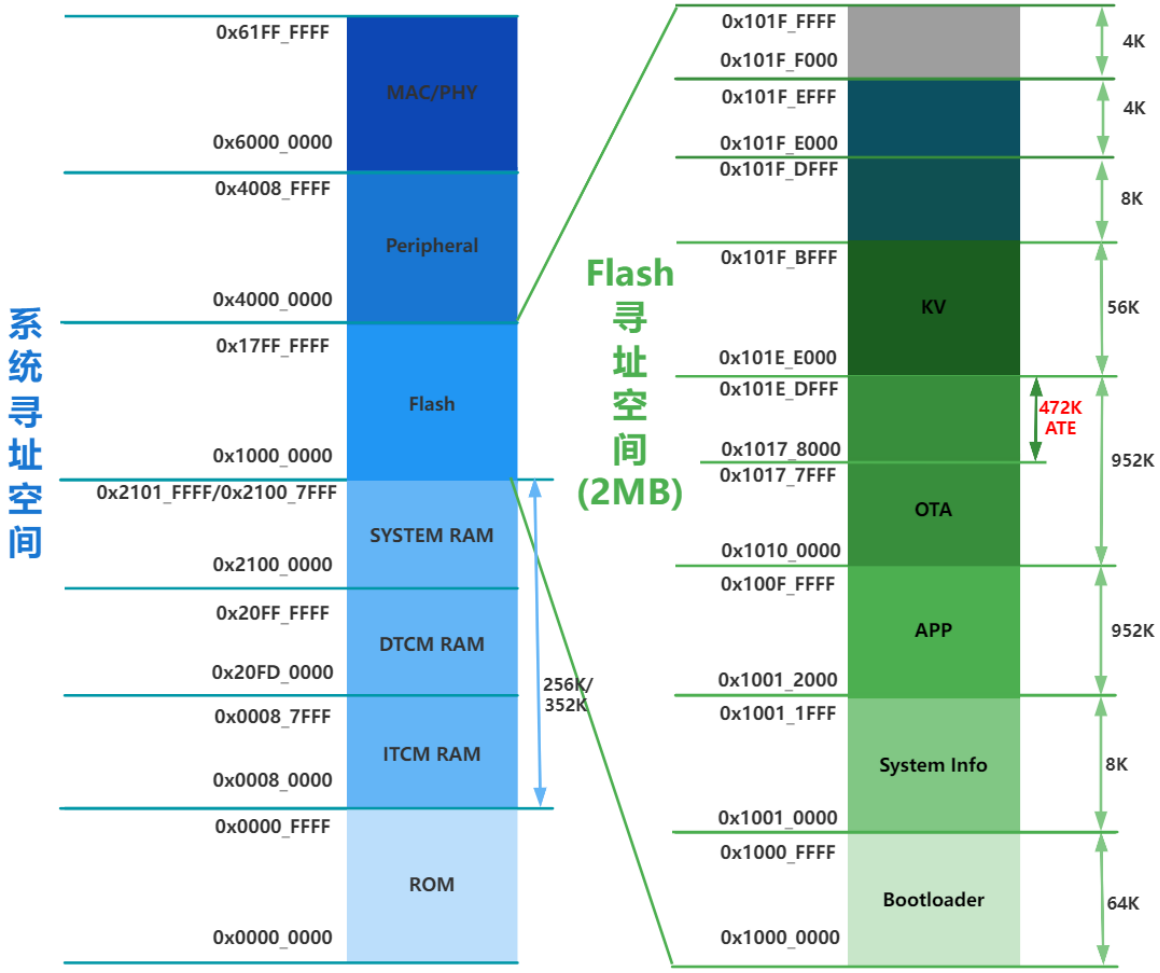

图 3-1 芯片寻址空间

3.2 Flash

ASR582X 系列芯片内置 2 M/4 M Flash，每个 block 大小为 4KB Bytes。ASR582X 系列芯片的 Flash 分区(逻辑地址)如下：

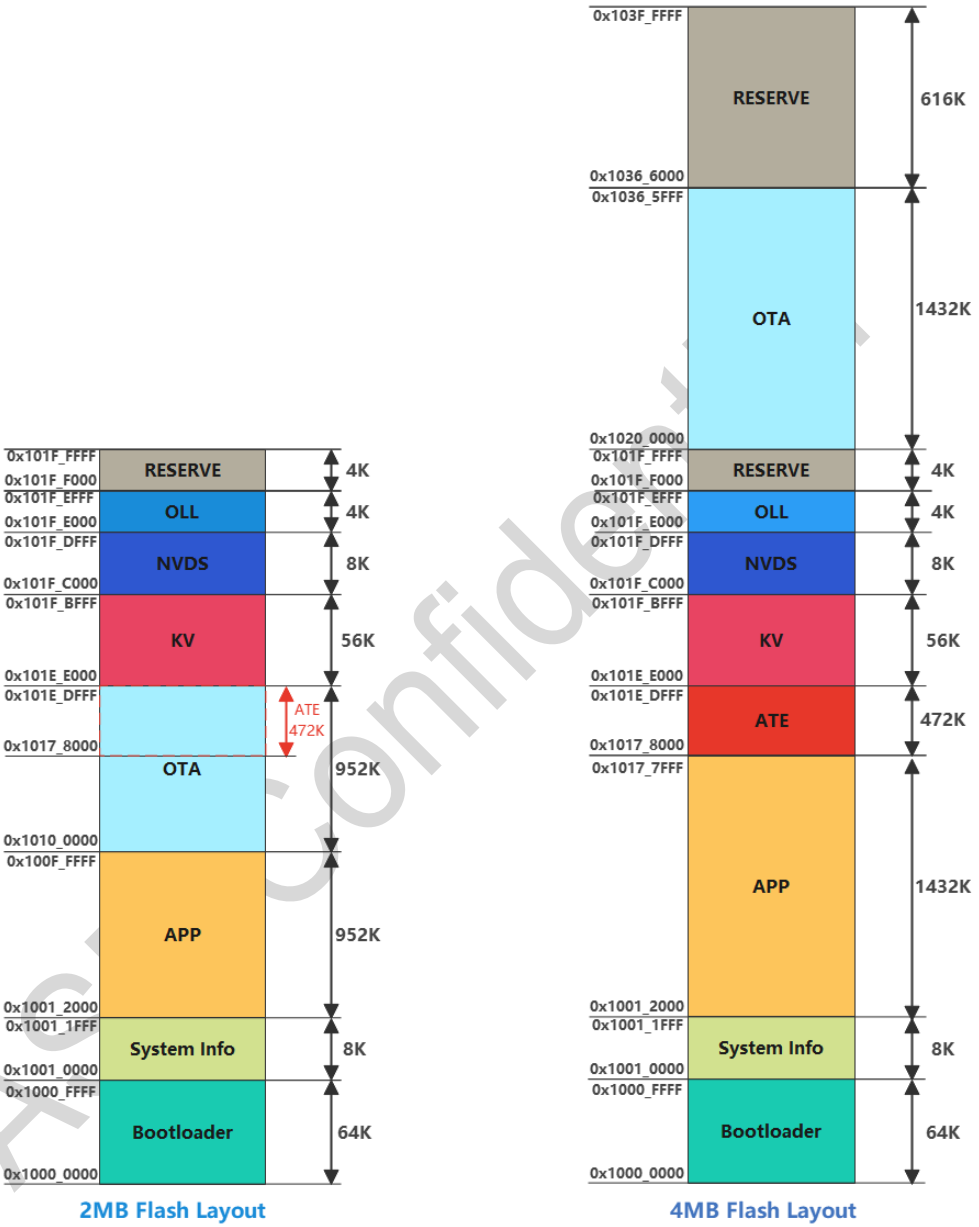

图 3-2 ASR582X Flash 分区

说明：

实际分区可能略有差异，具体可查看 SDK 中<duet_board.c>文件中的分区定义。

Flash 各分区的功能如下:

表 3-2 Flash 分区功能描述

分区名称	描述
Bootloader	程序引导区，分区大小 64 KB，bootloader.bin 烧录到此分区
System Info	系统信息区，分区大小 8 KB；前面 4 KB Flash 区域存放 OTA info 信息，后面 4 KB Flash 区域存放 TAG 信息。
APP	应用区，应用程序 app.bin 烧录到此分区。
OTA	升级区，OTA 在线升级时保存从云端获取到的 OTA 数据；OTA 升级的策略有两种，具体可到后面 3.10 OTA 升级章节查看
ATE	射频测试区，生产需要 RF 校准，此功能在设备端有独立的程序 ate.bin，ate.bin 文件会烧录到此区。
KV	用户存储区。
NVDS	BLE 系统信息存储区，用户不可使用。
OLL	系统相关信息存储，用户不可使用。

注意：

1. *在地址映射OTA 升级时，APP 分区和OTA 分区是不断交换的，具体可以查看3.10 OTA 升级章节。*
2. *2 M Flash 在第一次OTA 升级后，ate.bin 会被覆盖掉，之后无法再进入 ATE，如果 OTA 后需使用 ATE，则需要重新烧录 ate.bin。*
3. *分区 layout 定义建议不要轻易更改，否则有造成数据丢失的风险，如果开发者需要修改分区表大小或者增加新的分区，需要确保:*
   - *Bootloader/Info/APP/OTA/ATE 这几个分区的起始地址不能变。*
   - *APP 和 OTA 分区的大小应该保持一致。*
   - *改动或新增分区时，需要保证各分区之间没有重叠（特殊：2 M Flash 时，OTA 和 ATE 可能共用）*
   - *不能超过 Flash 总大小。*
   - *特殊：当使用 4 M Flash 时，ATE 开始位置 0x10178000，APP 起始位置为 0x10012000，也就意味着 APP 分区上限为 1432 KB。*

3.3 RAM

ASR582X 系列芯片的 RAM 主要由三部分组成，分别为 ITCM、DTCM 和 System RAM，它们在芯片内部有不同的起始地址；不仅用于用户变量的存储、系统堆栈，同时也是 Wi-Fi、蓝牙报文缓冲区的共享地址。

- ASR582X 系列芯片 RAM 大小：

  见表 3-1 ASR582X 存储器

- 系统可用的 RAM 还剩多少，与各 SDK 所运行的系统以及支持功能相关，用户可等平台运行起来后通过指令方式查看 SDK 剩余 RAM：

  - FreeRTOS：使用串口调试工具，输入如下指令查看：

    ```
    vtasklist
    ```

```
vtasklist
AT_task         R   20  0x000001ae  0x00001000  0x000874a0  0x000864a8  1
IDLE            R   0   0x000001bb  0x00000800  0x20fd5530  0x20fd4d38  6
timestamp_task  B   2   0x000001d8  0x00000800  0x20fd4d28  0x20fd4530  5
tcpip_thread    B   27  0x000001d0  0x00000800  0x20fd3d18  0x20fd3520  3
iperf-client    S   28  0x000001d5  0x00000800  0x20fd4520  0x20fd3d28  4
UWIFI_TASK      S   26  0x000006d3  0x00001c00  0x20fd3510  0x20fd1918  2
Tmr Svc         S   31  0x000003df  0x00001000  0x20fd6538  0x20fd5540  7

---------------------- Task Stack End ----------------------

left free heap 214696

OK
```

图 3-3 FreeRTOS 内存使用情况打印

  - AliOS：使用串口调试工具，输入如下指令查看：

    ```
    dumpsys mm_info
    ```

```
------------------------ memory allocation statistic -------------------------------
    free     |     used      |     maxused
     127608  |       50416   |       50856
```

图 3-4 AliOS 内存使用情况打印

  - HarmonyOS：使用串口调试工具，输入如下指令查看：

    ```
    heapshow
    ```

```
heapshow
pool0 :
pool addr         pool size    used size     free size    max free node size   used node num     free node num      UsageWaterLine
---------------   --------     -------       --------     ---------------      -------------     -------------      --------------
#x   0x20fd2ed8   0x2a128      0xa2dc        0x1fa94              0x1ea88             0x28              0x2
OK
```

图 3-5 HarmonyOS 内存使用情况打印

3.4 Efuse

ASR582X 系列芯片内置 4K bits (512 Bytes)的 efuse 存储，efuse 区域的值只能烧写一次，可多次读，efuse 的 layout 如下图:

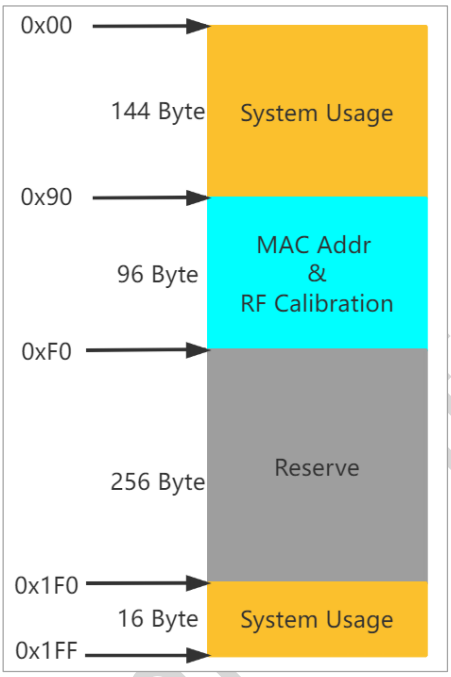

图 3-6 Efuse 分区表

0x90~0xEF 为 RF 校准参数和 MAC 地址参数，其中 RF 校准参数包括 Wi-Fi 和 BLE 的校准参数；每个参数区域目前都设置有 3 组，由于 efuse 只可烧写一次的特性，所以设置 3 组参数区域也就意味着芯片目前有且仅有三次重烧写的机会。

RF 校准参数和 MAC 地址只会读取最后一次烧写区域的数据，之前区域烧写的数据不会再被使用。

> ⚠ **注意：**
>
> ***Efuse 区域的特点为只能从“0”变成“1”***（这也是仅只能烧写一次的原因），系统提供操作 efuse 的最小单位为 Byte，如果强制重烧写 efuse 区域，则会得到与预期不一样的值。例如第一次烧写值 0x15 后，再次烧写值 0x43，最终 efuse 中存储的值为 0x57。

3.5 Lega RTOS

Lega RTOS 有封装主流 RTOS 的系统接口，建议用户应用层开发时也采用 Lega RTOS 提供的系统接口，这样后续能兼容不同 RTOS 平台。接口说明请参考文档：*ASR582X Series_RTOS Application Notes*。

3.6 Wi-Fi

Wi-Fi 接口 API 和应用实例，请参考文档：《ASR582X 系列_WLAN 开发指南》。

3.7 BLE

3.7.1 BLE 接口和参数说明

请参考文档：《ASR582X_BLE_API.chm》。

3.7.2 BLE 接口 API 使用示例

请参考文档：《ASR582X 系列_BLE 使用示例》。

3.8 AT 指令集

- FreeRTOS 平台 AT 指令说明及使用示例请参考文档《ASR582X 系列_FreeRTOS 平台_AT 命令使用指南》。
- AliOS 平台系统本身有调试命令，可在串口调试软件上输入“help”查看相关命令。调试命令的实现和使用请查看：`.\Living_SDK\tools\cli\cli.c`

```
help

====Build-in Commands====
====Support six c
mds once, separate by ; ====
help:
echo:
exit: CLI exit
devname: print device name
sysver:
reboot: reboot system
time: system time
ota: system ota
p: print memory
m: modify memory
ip: show lan ip
spi_read: read rf reg
spi_write: write rf reg

====User Commands====
lvl: Set loglevel. lvl [N/F/E/W/I/D]
tasklist: list all thread info
dumpsys: dump system info
tftp: tftp server/client control
udp: [ip] [port] [string data] send udp data
wifi_debug: wifi debug mode
mac: get/set mac
coredump: coredump [ram | flash] [all | pos [memory | wifi | info | rf | peri]]
kv: kv [set key value | get key | getb key | del key | clear]
version: show version
netmgr: netmgr [start|clear|connect ssid password]
reset: factory reset
active_awss: active_awss [start]
ble_awss: ble_awss [start]
awss: awss [start]
clear_awss: clear_awss [start]
linkkey: set/get linkkit keys. linkkey [<Product Key> <Device Name> <Device Secret> <Product Secret>]
dev_serv: device diagnosis service tests. like errcode, and log report
```

图 3-7 AliOS 调试命令

- HarmonyOS 平台系统本身有调试命令，可在串口调试软件上输入“help”查看相关命令。调试命令的实现和使用请查看：`.\device\asr\asr582x\drivers\at_cmd\atcmd_user.c`

```
help
AT command as below, use "help name" for more details

dbg

reset

version

echo

recovery

uart_config

uart_config_def

uart_config_def_del

coredump

vtasklist

heapshow

at_test1

at_harmony_xts

at_hilink_start

at_ota_init

at_kv_clear

testkv

AT+LOG

OK
```

图 3-8 HarmonyOS 调试命令

3.9 MAC 地址

实际产品中 MAC 地址由工厂在生产过程中通过测试工具写入到 efuse 存储器中。建议在进行 RF 校准测试后写入 MAC 地址。

通常用户的开发板未进行 RF 校准，但用户又需要进行基本的功能开发，SDK 有提供接口将 MAC 地址保存到 Flash 中。

3.9.1 Wi-Fi MAC 地址

ASR 的 SDK 内提供一组 API 对 Wi-Fi 的 MAC 地址进行操作(**<lega_wlan_api.h>**)，如下图所示：

```
263  /** @brief  used in station and softap mode, get mac address(in hex mode) of WIFI device
264   *
265   * @param mac_addr    : pointer to get the mac address
266   *
267   *  @return    0       : on success.
268   *  @return    other   : error occurred
269   */
270  int lega_wlan_get_mac_address(uint8_t *mac_addr);
271
272  /** @brief  used in station and softap mode, set mac address for WIFI device
273   *
274   *  @param mac_addr    : 6 bytes src mac array to set, DOES NOT support string or pointer
275   *
276   *  @return    0       : on success.
277   *  @return    -1      : NULL pointer
278   *  @return    -2      : mac address already exist in efuse, can't set
279   *  @return    -3      : param is invalid
280   */
281  int lega_wlan_set_mac_address(uint8_t *mac_addr);
```

图 3-9 Wi-Fi MAC 地址操作函数

- **lega_wlan_set_mac_address：**

功能：将 Wi-Fi MAC 地址保存到 Flash 中，若 efuse 中已写入 MAC 地址，则调用该接口返回错误。

- **lega_wlan_get_mac_addr：**

功能：首先查看 efuse 中是否有烧写 MAC 地址，有则返回此 MAC 地址，没有保存则查看 Flash 中是否保存过 MAC 地址，有保存过则返回此地址，没有保存则系统产生随机 MAC 地址，并且将产生的随机地址保存到 Flash 中，并返回此地址。

3.9.2 BLE MAC 地址

ASR 的 SDK 内提供一组 API 对蓝牙的 MAC 地址进行操作(<sonata_utils_api.h>)，如下图所示：

```
306  /*!
307   * @brief get BT address
308   * @return address
309   */
310  uint8_t * sonata_get_bt_address();
311
312
313  /**
314  ****************************************************************************************
315  * @brief sonata set bd addr
316  * @param[in] uint8_t* bd_addr : bd_addr value
317  *            uint8_t  length: addr length
318  * @return bool : true(success) or false(fail)
319  ****************************************************************************************
320  */
321  bool sonata_set_bt_address(uint8_t * bd_addr,uint8_t length);
```

图 3-10 蓝牙 MAC 地址操作函数

- **sonata_set_bt_address：**

功能：将 BLE MAC 地址保存到 Flash 中。

- **sonata_get_bt_address：**

功能：首先查看 Flash 中是否有保存过 BLE MAC 地址，有则返回此 MAC 地址，没有保存则会读取 Wi-Fi 的 MAC 地址，以 Wi-Fi MAC 地址+1 的规则生成 BLE 的 MAC 地址：

比如：Wi-Fi MAC 地址为 84:7c:9b:8f:2d:57，则 BLE MAC 地址为 84:7c:9b:8f:2d:58。

> **注意：**
> *如果有调用 sonata_set_bt_address 设置了蓝牙 MAC 地址后，会优先使用写入的 MAC 地址。*

3.10 OTA 升级

请参考文档：《ASR582X 系列_OTA 功能开发指导》。

3.11 PINMUX

表 3-3 Pin Mux-1

Num.	Pin Name	GPIO Func=0	GPIO Func=1	GPIO Func=2	GPIO Func=3
1	DIG_PAD0	GPIO0	UART0_TXD	SWC	SPI1_CSN
2	DIG_PAD1	GPIO1	UART0_RXD	SWD	SPI1_SCK
3	DIG_PAD2	GPIO2	UART1_TXD	UART1_TXD	SPI1_DI
4	DIG_PAD3	GPIO3	UART1_RXD	UART1_RXD	SPI1_DO
5	DIG_PAD4	SWC	GPIO4	SDIO_CMD	UART0_TXD
6	DIG_PAD5	SWD	GPIO5	SDIO_CLK	UART0_RXD
7	DIG_PAD6	GPIO6	SPI0_CSN	SDIO_DATA0	UART0_CTS
8	DIG_PAD7	GPIO7	SPI0_SCK	SDIO_DATA1	UART0_RTS
9	DIG_PAD8	GPIO8	SPI0_TX	SDIO_DATA2	I2C1_SCL
10	DIG_PAD9	GPIO9	SPI0_RX	SDIO_DATA3	I2C1_SDA
11	DIG_PAD10	MODE_SEL3	PWM1	GPIO10	UART2_CTS
12	DIG_PAD11	GPIO11	PWM3	SDIO_INT	UART2_RTS
13	DIG_PAD12	GPIO12	GPIO12	SPI2_CSN	UART2_TXD
14	DIG_PAD13	GPIO13	GPIO13	SPI2_DO	UART2_RXD
15	DIG_PAD14	STRAP/SEL1	PWM0	SPI2_SCK	UART1_CTS
16	DIG_PAD15	STRAP/SEL2	PWM2	SPI2_DI	UART1_RTS

表 3-4 Pin Mux-2

Num.	Pin Name	GPIO Func=4	GPIO Func=5	GPIO Func=6	GPIO Func=7	ADC Mux
1	DIG_PAD0	PWM5	N/A			
2	DIG_PAD1	PWM7	N/A			
3	DIG_PAD2	I2C0_SCL	N/A			
4	DIG_PAD3	I2C0_SDA	N/A			
5	DIG_PAD4	PWM0	N/A		PSRAM_SIO3	ADC0
6	DIG_PAD5	PWM2	N/A		PSRAM_SIO2	ADC1
7	DIG_PAD6	PWM4	N/A		PSRAM_SCK	ADC2
8	DIG_PAD7	PWM6	N/A	I2S_MCLK	PSRAM_SIO1	ADC3
9	DIG_PAD8	UART1_TXD	N/A	I2S_SCLK	PSRAM_SIO0	ADC4
10	DIG_PAD9	UART1_RXD	N/A	I2S_LRCLK	PSRAM_CSN	ADC5
11	DIG_PAD10	SPI2_SCK	N/A	I2S_DO		ADC6
12	DIG_PAD11	SPI2_DI	N/A	I2S_DI		ADC7
13	DIG_PAD12	GPIO12	N/A	I2S_DO		
14	DIG_PAD13	GPIO13	N/A			
15	DIG_PAD14	GPIO14	N/A			
16	DIG_PAD15	GPIO15	N/A			

- ASR582X 系列有 16 个通用的 GPIO 口，为了在有限的资源上丰富外设功能，GPIO 可配置成不同的功能。
- 引脚默认配置为 Func=0，如果 pinmux 配置成其他外设功能，需要使用 duet_pinmux_config 函数进行相应的配置。

例如：SDK 提供的 pwm driver 中对 pad14 用做 pwm0 的配置。如下图所示：

```
//pwm pinmux init
void duet_pwm_pinmux_init(duet_pwm_dev_t *pwm)
{
    switch(pwm->port)
    {
        case PWM_OUTPUT_CH0:
            //pin mux control
            //PWM0_PAD PAD14 1
            duet_pinmux_config(PAD14,PF_PWM0);
            break;
```

图 3-11 Pinmux 示例

3.12 外设

- 3 个 UART 接口
- 3 个 SPI 接口
- 2 个 I2C 接口
- 8 路 PWM 接口
- 8 个定时器
- 1 个 SDIO 接口
- 8 路 ADC 通道

外设 API 接口请参考文档：*ASR582X Series_Peripheral Application Notes*。

3.12.1 外设信息补充以及注意事项

3.12.1.1 GPIO

- 开机默认驱动模式
  1. 除 pad5/pad13 上拉输入外，其余都默认下拉输入。
  2. 驱动模式中的上拉/下拉都为芯片内部硬件驱动。
- 支持的驱动模式
  1. 输入上拉：内置上拉电阻大约 10 KΩ
  2. 输入下拉：内置下拉电阻大约 16 KΩ
  3. 高阻输入
  4. 推挽输出
  5. 中断模式：

  支持高电平、低电平、上升沿、下降沿四种触发方式
- 最大驱动电流：12 mA。

> ⚠ **注意：**
>
> 1. *PAD 悬空时上电默认：PAD5/PAD13 为高电平，其余 PAD 为低电平。*
> 2. *PAD13 有特殊作用，建议不使用，默认悬空处理。*
> 3. *PAD10、PAD14 和 PAD15 为硬件模式选择脚，为了不影响上电后的模式判断，这几个引脚不建议使用，如果确实要使用， 则需确保外部不能有长上拉电路*
> 4. *SEL 所有引脚的高电平均为 3.3 V，不能输入 5 V，否则会损坏引脚。*
> 5. *PAD0~PAD3 的电压跟随 VDD。*

3.12.1.2 Flash

- Flash 支持最高 10 W 次擦写；
- Flash 擦除操作的起始地址和擦除长度必须扇区对齐（4 KB）。

> 📖 **说明：**
>
> 1. *对 Flash 写数据不易过度频繁以及单次写数据量不要过大，因为在 Flash 进行擦写操作时，系统会先关闭系统中断，而 Wi-Fi/BLE 协议栈接受发送数据需要中断，长时间关闭系统中断会对 Wi-Fi 数据传输产生影响。*
> 2. *当存储的数据量比较小，或更新较为频繁的数据，建议使用 KV 存储，以增加 Flash 的寿命。*

3.12.1.3 KV

- KV 实际是用 key-value 方式的一种封装；系统将 Flash 中指定的一块区域作为 KV 存储空间，帮助用户管理键值对的映射关系，方便用户层使用，用户无需关心具体写入 Flash 的地址是多少，只需要对 key(键)的操作来完成 value（值）的存取。
- KV 存储的 API 接口如下图所示：

```
/**
 * Init the flash kv module.
 *
 * @retrun 0 on success, otherwise will be failed.
 */
int32_t duet_flash_kv_init(void);

/**
 * Deinit the kv module.
 *
 * @retrun none
 */
void duet_flash_kv_deinit(void);

/**
 * Add a new flash KV pair.
 *
 * @param[in]  key    the key of the KV pair.
 * @param[in]  value  the value of the KV pair.
 * @param[in]  len    the length of the value.
 * @param[in]  sync   save the KV pair to flash right now (should always be 1).
 *
 * @return  0 on success, negative error on failure.
 */
int32_t duet_flash_kv_set(const char *key, const void *value, int32_t len, int32_t sync);

/**
 * Get the flash KV pair's value stored in buffer by its key.
 *
 * @note: the buffer_len should be larger than the real length of the value,
 *        otherwise buffer would be NULL.
 *
 * @param[in]      key         the key of the KV pair to get.
 * @param[out]     buffer      the memory to store the value.
 * @param[in-out]  buffer_len  in: the length of the input buffer.
 *                             out: the real length of the value.
 *
 * @return  0 on success, negative error on failure.
 */
int32_t duet_flash_kv_get(const char *key, void *buffer, int32_t *buffer_len);
```

图 3-12 KV 存储 API 接口

KV 根据 key 的名称来保存和获取值。例如：

保存 user1 的名字：duet_flash_kv_set (“user1”， data_buf， sizeof (data_buf)， 1);

获取 user1 的名字：duet_flash_kv_get (“user1”， buffer， &readLen)。

说明：

1. *在使用 KV 存储之前需要调用 duet_flash_kv_init 初始化 KV 组件，只需要调用一次。*
2. *duet_flash_kv_get 获取 KV 数据，buffer_len 为需要读取的 value 长度，必须赋值，且值需要大于 0。*

4. 量产相关

4.1 烧录

MP 工具

如下图原厂提供量产快速烧录工具 MP tool 以及使用手册，MP 工具会将 bootloader/app/ate 三个 bin 文件烧录到 Flash 中的对应分区。

烧录工具：**MP_Pro_V3.8.2.exe**

使用说明参考文档《ASR IoT 芯片_MP_Pro 量产烧录工具使用说明》。

图 4-1 MP 烧录工具及使用说明

MP 工具的特点

- 支持 20 个设备同时烧录，建议客户拼板烧录。
- 单个设备烧录时间短，传输速率到达 1 Mbps。(Dogo 烧录工具的传输速率 115200 bps)。
- 从 V2.5 版本开始仅支持三合一版本 bin 档烧录，可有效避免工厂混用不同版本 bin。将 bootloader/app/ate 合成一个 bin 的工具为 MP_FG_Pro，由 ASR 提供。

EXTT 多合一工具

EXTT 工具目前集成了巡检、布局校准以及多合一固件生成功能

烧录工具：**EXTT_V1.0.4.exe**

使用说明参考文档：《多合一工具 EXTT 使用说明_V1.0.2》。

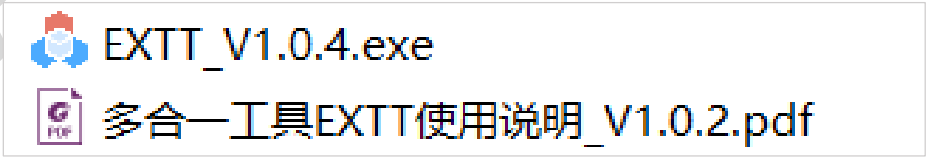

图 4-2 EXTT 工具及使用说明

4.2 产测

产测包括：RF 校准、RF 校准值和 MAC 地址的写入，可能还有其他信息的写入，例如 license/Tag。

RF 校准值和 MAC 地址都是写入到 efuse 存储器中，并且最多只能写 3 次。

> **说明：**
>
> *efuse 区域的特点就是仅只能写一次，不能重复烧写，为避免生产时非正常流程导致校准或者写入的值错误而产品报废的损失，系统实际是以牺牲 efuse 的空间来换取 2 次重烧写的次数。*

RF 校准所需的测试仪器，目前有 itest 极致汇仪和 APT 爱普特两家产商支持，如客户还需其他产商支持，需自己开发 PC 端工具，ASR 提供对应的软件接口。

- **itest 极致汇仪**

  工具名称：WLAN Facility.exe

- **APT 爱普特**

  工具名称：APTS_STARTER.exe

RF 校准测试请参考文档《ASR582X 系列_射频测试指南》。

5. 硬件资源

5.1 用户设计手册

请参考文档:《ASR582X 系列_硬件设计指南》。

5.2 参考设计

硬件设计请参考由 ASR 提供的如下文档:《ASR55822S_QFN40_DB.pdf》、《ASR55822S_QFN40_DB.pcb》。

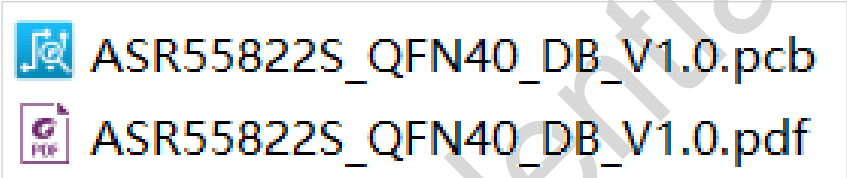

图 5-1 设计参考

A. 附录-文档目录

《ASR582X 系列_开发环境搭建》

《ASR582X 系列_开发板使用指南》

《ASR IoT 芯片_MP_Pro 量产烧录工具使用说明》

《ASR IoT 芯片_DOGO 烧录调试工具》

《ASR582X Series_RTOS Application Notes》

《ASR582X 系列_WLAN 开发指南》

《ASR582X_BLE_API.chm》

《ASR582X 系列_BLE 使用示例》

《ASR582X 系列_FreeRTOS 平台_AT 命令使用指南》

《ASR582X 系列_OTA 功能开发指导》

《ASR582X Series_Peripheral Application Notes》

《ASR IoT 芯片_MP_Pro 量产烧录工具使用说明》

《多合一工具 EXTT 使用说明》

《ASR582X 系列_硬件设计指南》

《ASR55822S_QFN40_DB.pdf》

《ASR55822S_QFN40_DB.pcb》