



第 1 章 开发环境搭建

本章介绍如何在 windows 系统下搭建 SDK 开发环境，包括 IDE 的安装，驱动安装和调试工具软件的安装。

1.1 IAR8.32 的安装和注册

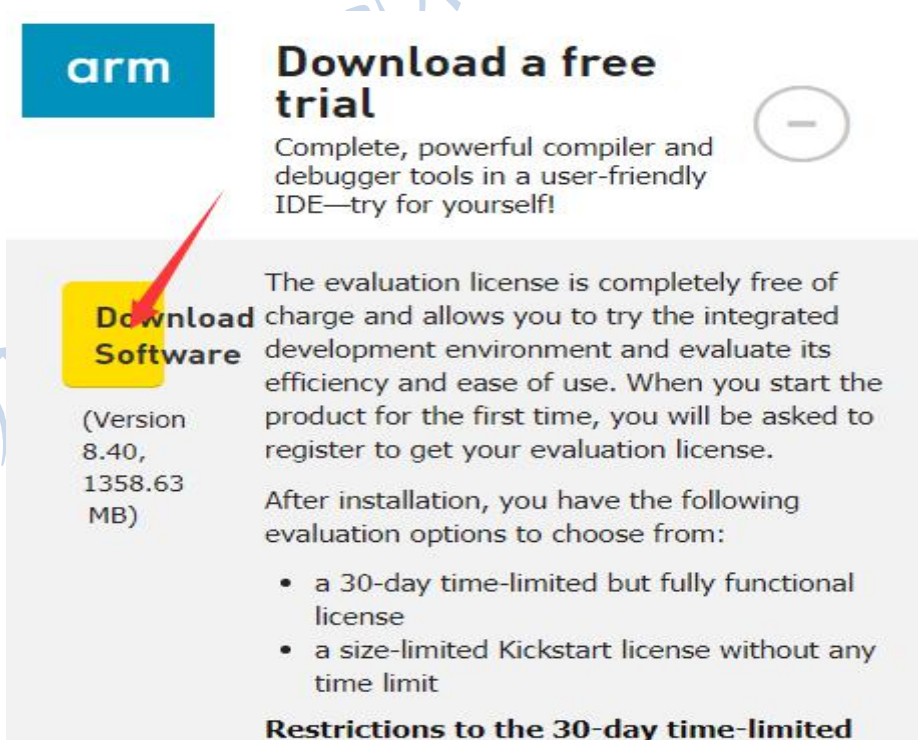
我们提供的 SDK 是基于 IAR 8.32.1 版本的工程，需要安装 8.32.1 或更高的软件版本才能正常打开和编译，低版本的 IAR 打开工程编译会出错。

1.1.1 下载 IAR8

登录 IAR 官网下载，官网地址

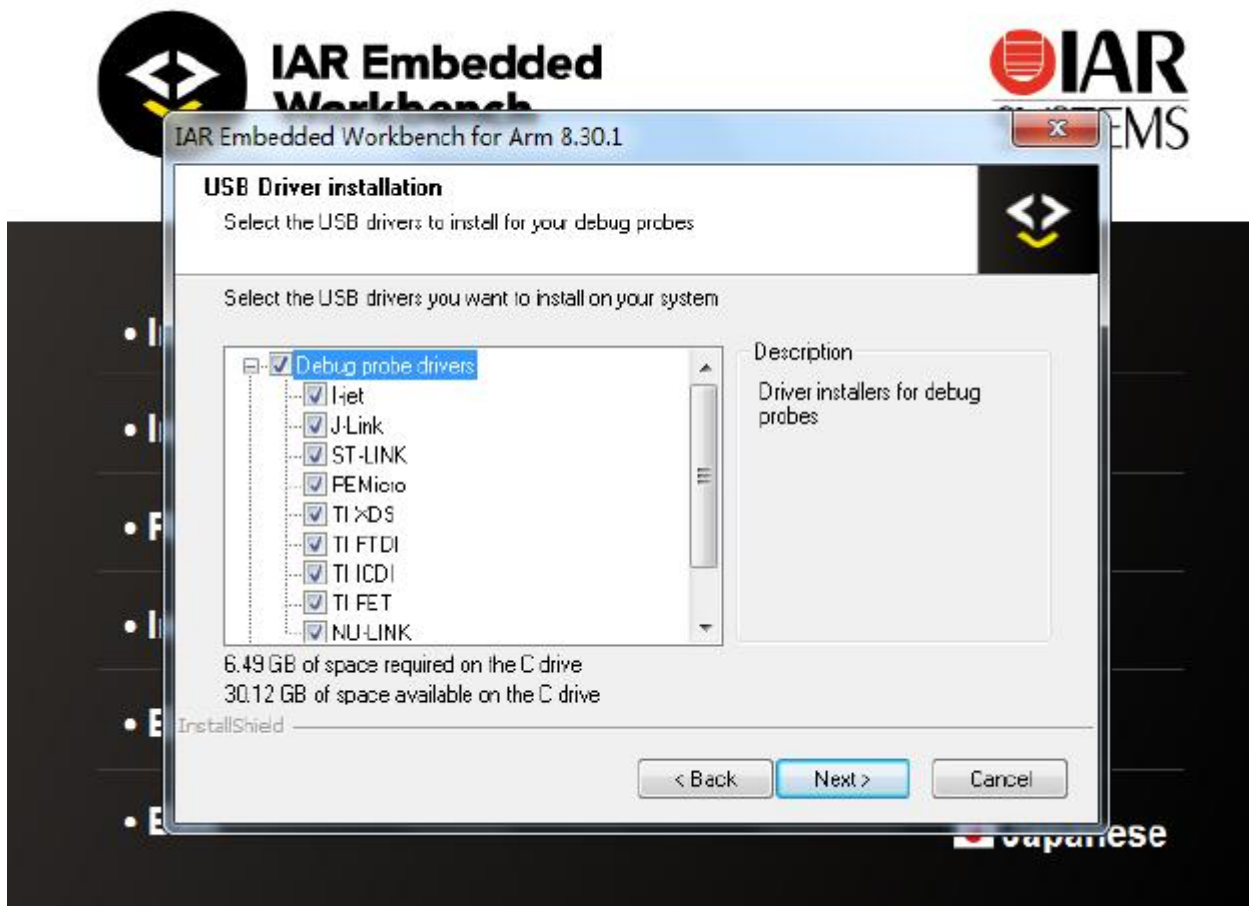
<https://www.iar.com/iar-embedded-workbench/#!?architecture=Arm>

点击箭头处下载按钮进行下载（当前最新版本是 8.40）



1.1.2 安装和注册 IAR8

软件安装过程比较简单，打开下载的安装包，按照提示进行安装即可。安装过程中需要注意，选择安装路径的时候不要有中文。



按照自己的需求选择要安装的驱动包，安装驱动时不要将下载器插入电脑。

安装完成后先不用注册，进入软件后进行注册。



到此软件安装步骤完成，打开安装完成的软件，下面需要完成的是注册 License，软件不注册只有 30 天的试用时间，工具栏中的 Help->License Manager 进行注册。支持正版软件，如仅用于学习用途，可自行百度一下注册方法。

1.2 JLINK 驱动安装

AmebaZ II 的 SDK 支持 JLINK 的 SWD 接口在线调试代码和下载程序，方便对 SDK 进行二次开发。

使用 JLINK 进行调试先要安装 JLINK 驱动，JLINK 驱动官方下载地址：

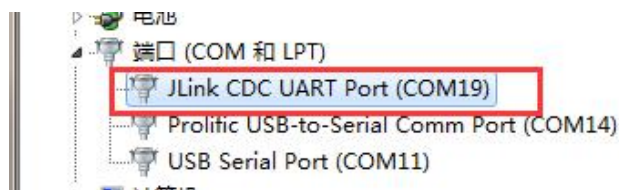
<https://www.segger.com/downloads/jlink#J-LinkSoftwareAndDocumentationPack>

本文档是在 v6.40 版本上进行调试的。

注意：由于 AmebaZ II 是 Cortex-M33 的内核，JLINK 的硬件版本要在 V9 及以上才支持，低版本的 JLINK 不支持 Cortex-M33。

安装好 JLINK 驱动后打开电脑的设备管理器，可以看到 JLINK

设备，如下图：



软件安装完成后打开 IAR 工程配置好相关选项即可用 JLINK 在线调试代码和下载程序到 FLASH。

1.3 辅助开发调试工具介绍

1.3.1 串口调试助手

SDK 中系统的 debug 信息通过 log 串口输出，有完善的 log 输出接口给用户使用，方便在开发过程中输出用户自己的调试 log，遇到 bug 时快速定位。把模块的 debug 串口通过 TTL 电平转换工具接到 PC 端，在 PC 端打开串口调试工具即可看到模块的 log 信息。在开发阶段也可以通过串口助手从 PC 端给模块发送 AT 指令进行指令集的开发。

比较好用的串口工具是 [SecureCRT](#)，支持日志自动保存到文件，时间戳等功能。SecureCRT 工具可以自行在网上下载安装，使用方法也比较简单，设置好串口参数，连接上目标串口即可。

1.3.2 TCP/UDP 网络助手

在 wifi 模块的 SDK 开发过程中离不开网络的测试，频繁的测试 TCP/UDP 连接和数据的收发，验证 WIFI 的性能，需要 PC 端的软件来配合调试。常用的 PC 端工具软件有 [NetAssist](#)。NetAssist 是一款简易的网络测试工具，不用安装，下



载后直接打开使用。支持 TCP/UDP Server, TCP/UDP Client, 跟 WIFI 模块建立 TCP/UDP 连接后进行数据收发。

工欲善其事必先利其器，到此我们的 SDK 的开发和调试环境基本建立起来了，下一章开始我们的 SDK 开发之旅。

九九物联（深圳）有限公司

第 2 章 SDK 介绍

2.1 SDK 工程文件夹介绍

component: 包含 FreeRTOS, Lwip 协议栈, 蓝牙, mqtt 协议, AT 指令集和上层的 API, 所有的源码都放在这个文件夹, 根据子文件的命名可以清晰的找到各个部分的代码。

doc: doc 文件夹里包含了 API 的说明文档, 原厂的 AT 指令集和各功能模块的说明文档, 用户可以查阅里面的相关文档来熟悉 SDK 的各个功能模块的实现和配置。

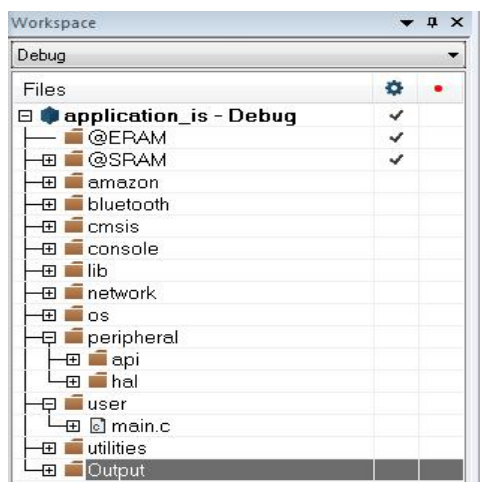
project: 该文件夹包含 IAR 工程的配置文件, 编译输出文件和模块外设的使用参考例程代码。

2.1 SDK IAR 工程结构介绍

打开 IAR 软件 (8.32 以上的版本), 加载 IAR 的工程文件 Project_is.eww, 工程文件在 SDK 中的路径为

project\realtek_amebaz2_v0_example\EWARM-RELEASE

打开工程后目录结构如下图:





@ERAM 和 **@SRAM** 目录根据用户的需求，将关键代码放置到这两个目录，具体的说明参照文档 AN0500 的 2.4.3.1 章节。

amazon: 连接亚马逊云服务的 demo 源码。

bluetooth: 蓝牙配网，GATT 服务的例程和蓝牙协议栈的相关代码。

cmsis: 进入 main 函数前的相关初始代码。

console: 指令集 API 函数集合，包括 WIFI 指令，蓝牙指令，log 串口指令，系统设置相关指令。

lib: 原厂封装的 wlan 等应用库，包含 wlan，蓝牙 gap，http 等，不提供源代码，以封装库的形式给用户调用。

network: 网络部分相关的源代码，包含 lwip，http，google demo，ssl 加密等。

os: FreeRTOS 系统源码。

peripheral: 外围接口操作 API，包含 i2c，sdio，gpio，timer，pwm，efuse 和串口等。使用 MCU 的外围接口需要调用对应文件的 API。

user: main 函数入口文件，包含控制台初始化入口，网络初始化入口和应用入口函数 example_entry 的调用。

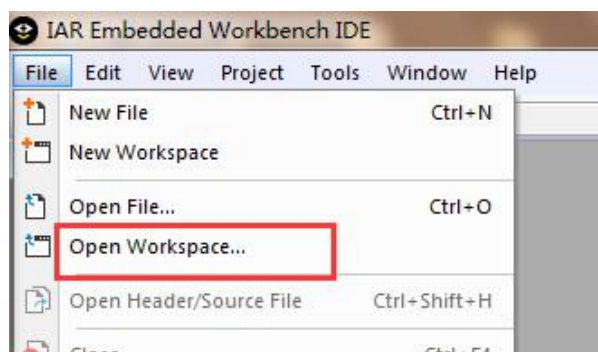
utilities: 功能模块的入口代码，每个源文件对应一个功能模块，通过配置头文件 platform_opt.h 中的宏定义来开启对应的功能。

output: 编译产生的.o 文件，.map 文件。

2.1 编译 SDK

2.1.1 打开 SDK 工程

打开 IAR 软件，加载工程文件 file→open workspace



加载 project\realtek_amebaz2_v0_example\EWARM-RELEASE 路径下的 Project_is.eww 文件即可打开 SDK 工程。

2.1.2 编译 SDK

首次打开工程先 clean 以前编译的信息 project→clean，清除旧的编译信息之后 project→make 开始 SDK 的编译工作，编译完成后，在以下目录生成可执行的 bin 文件 project\realtek_amebaz2_v0_example\EWARM-RELEASE\Debug\Exe。生成的 bin 文件有 bootloader.bin，firmware_is.bin，flash_is.bin 和 partition.bin。

partition.bin：存放分区列表信息，boot image 和 firmware image 的地址信息。

bootloader.bin：模块的启动引导文件，模块启动时执行 bootloader.bin 对硬件底层进行初始化和分区跳转，boot 文件由原厂提供，不需要修改。

firmware_is.bin：是应用固件，执行完 boot 程序之后跳转到应用固件执行。

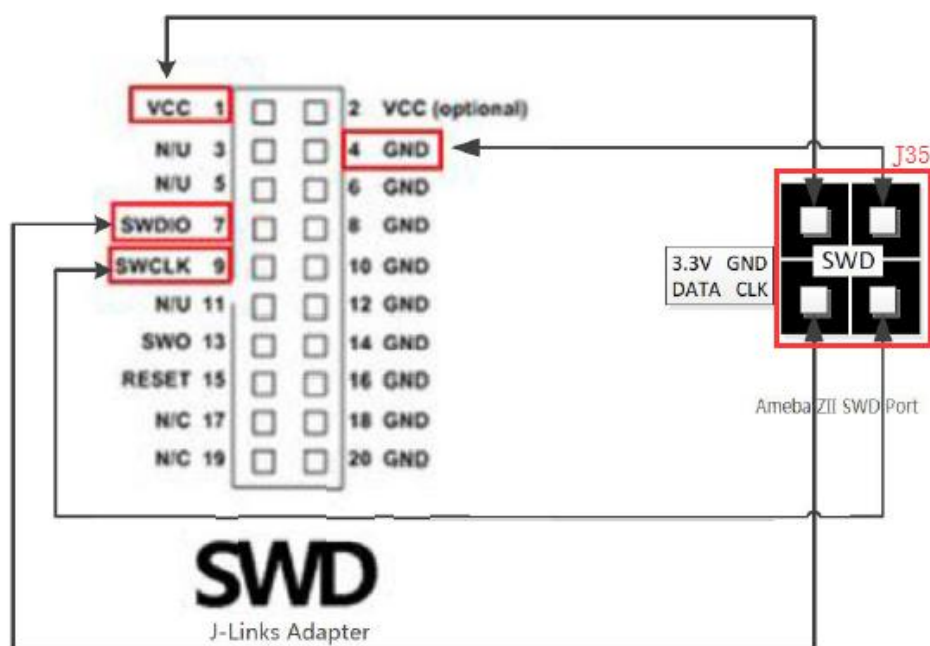
flash_is.bin：合成固件，将 bootloader.bin，firmware_is.bin 和 partition.bin

合成一个bin文件，可以用PG Tool下载到flash中。

2.2 固件下载

2.2.1 用 JLINK 下载固件

SDK 支持 JLINK 的 SWD 方式下载固件，首先将 JLINK 连接到电脑上，用杜邦线将开发板的 SWD 下载调试接口 J35 与 JLINK 的 SWD 接口按照下图进行连接：



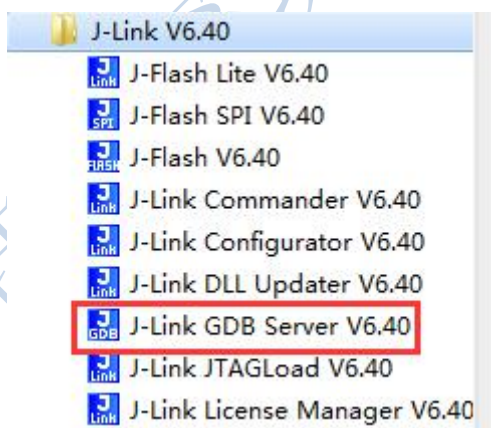
JLINK 引脚	开发板 J35 引脚
VCC	3.3V
GND	GND
SWDIO	DATA
SWCLK	CLK

实物连接图如下：

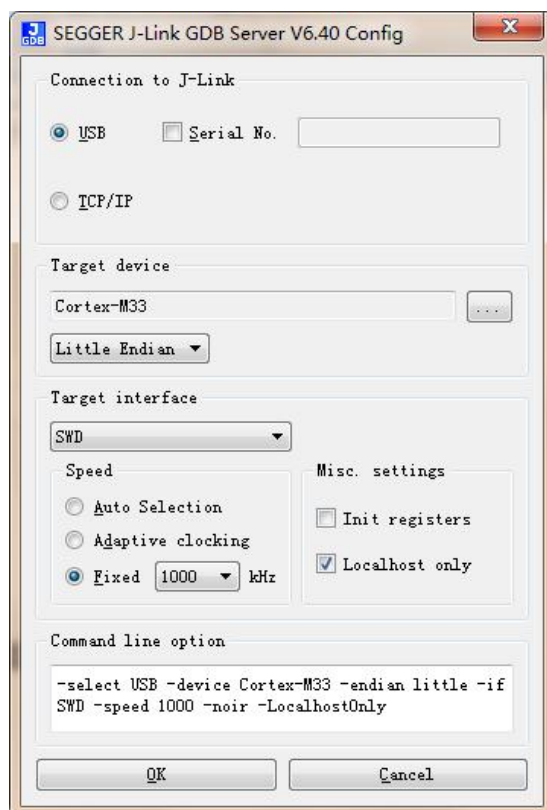


连接上 JLINK 之后还要用 micro usb 线将开发板的 CON3 接口连接到电脑，电脑自动识别安装串口驱动，在电脑端打开串口软件 SecureCRT 并连接开发板的 log 串口看板子输出的 log 信息。

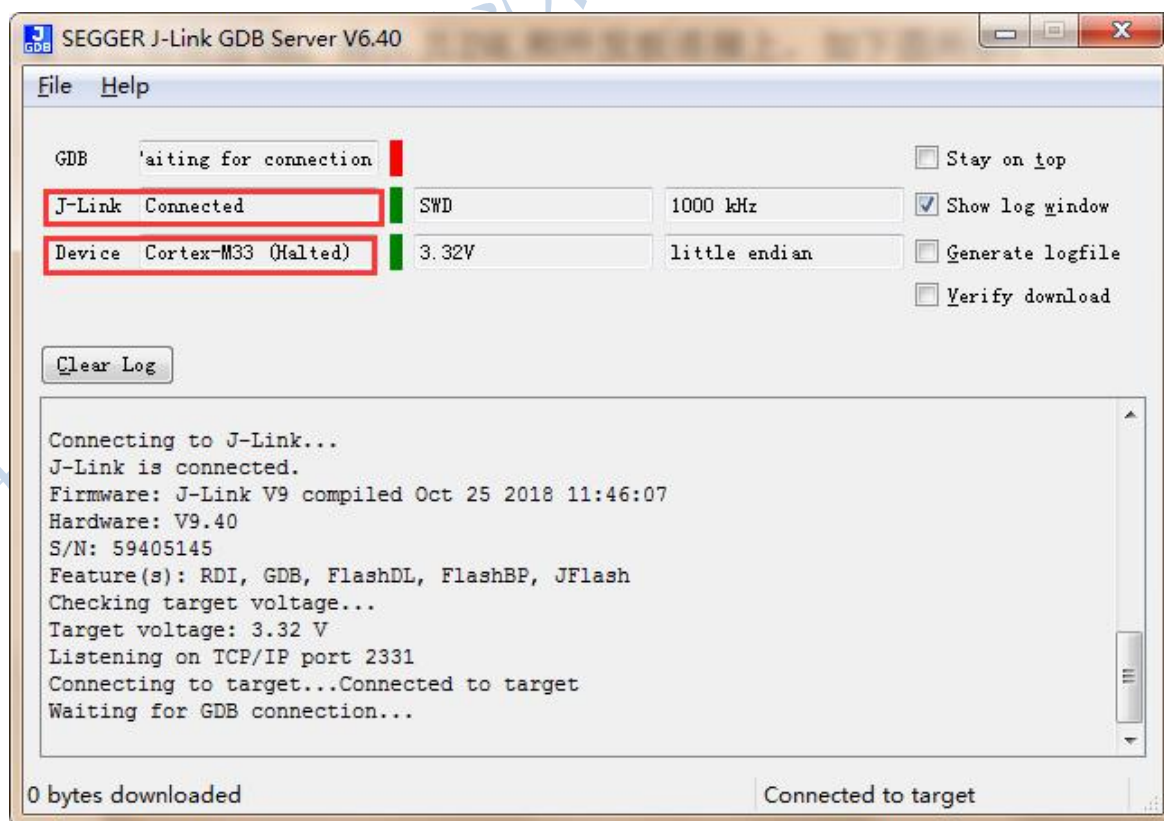
硬件连接完成之后打开安装好的 JLINK 软件（上一章介绍了怎么安装），打开 GDB Server，如下图所示：



打开后按照下图配置软件：



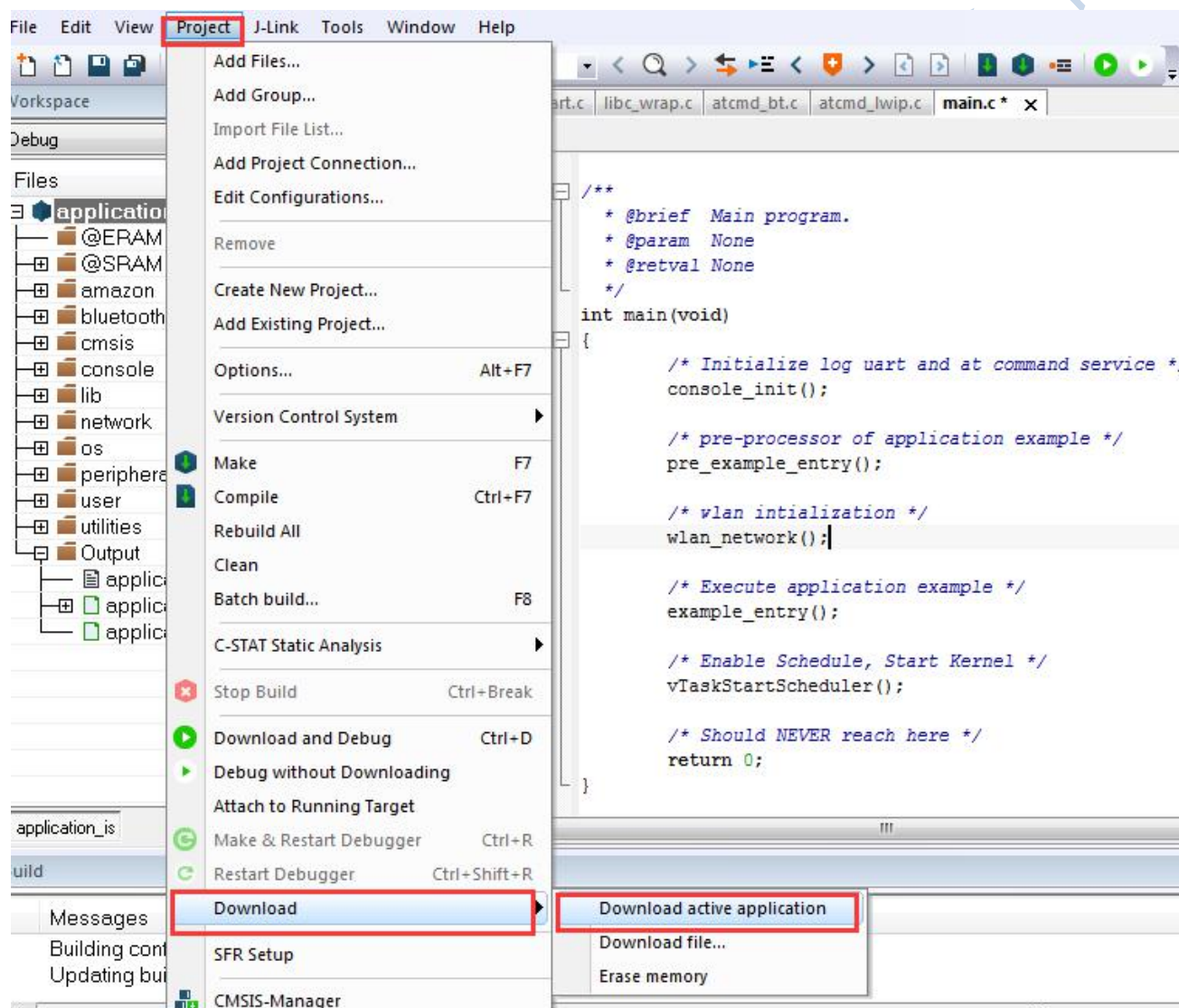
配置好之后点击 OK，确认 JLINK 和开发板连接上，如下图所示：



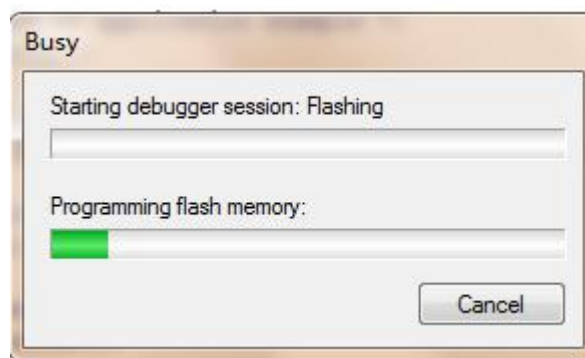
注意：上图红色框框 J-Link: Connected 表示 PC 端已经连接上了 JLINK

Device: Cortex-M33 (Halted) 表示 JLINK 已经连接上开发板，这两项如果不是显示以上内容则表示连接不成功，检查硬件连接和软件设置。

连接好 PC→JLINK→开发板之后就可以在 IAR 中进行固件下载了，下载流程如下：Project→Download→ Download active application, 如下图：

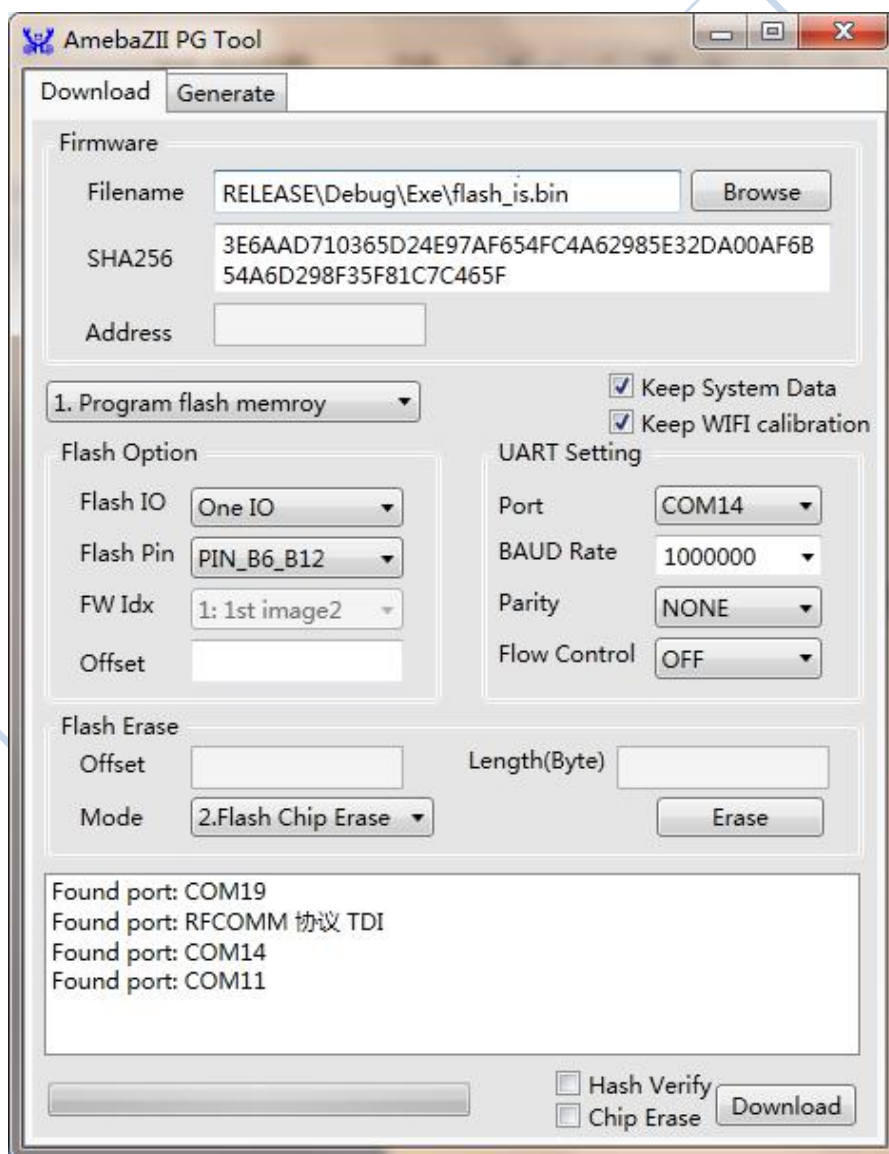


点击 Download active application 之后如果代码有修改，会先编译代码生成新的 bin 文件，等待编译完成之后开始下载，下载过程如下图：



2.2.2 用 PG tool 下载固件

Ameba-ZII 也可使用原厂提供的下载工具 PG tool 进行固件一键下载，方便快捷，软件界面如下：



软件设置:

Firmware 区: 加载需要烧写的 bin 文件。

Program flash memory: 选择烧写模式为 flash。

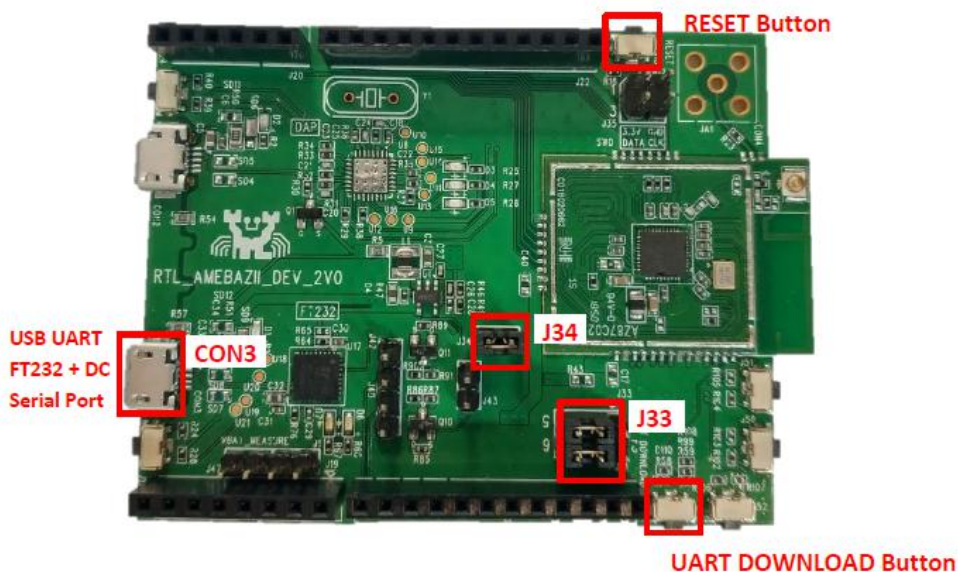
UART Setting: 选择串口端口号并设置串口参数，端口号为开发板的 log 串口连接上 PC 的端口号，在 PC 的设备管理器中可以看到。波特率为 1000000，无流控和校验。

Flash Option: 设置 flash 参数，Flash IO 为 One IO，Flash Pin 不同的芯片设置不一样，芯片型号对应 IO 设置如下图：

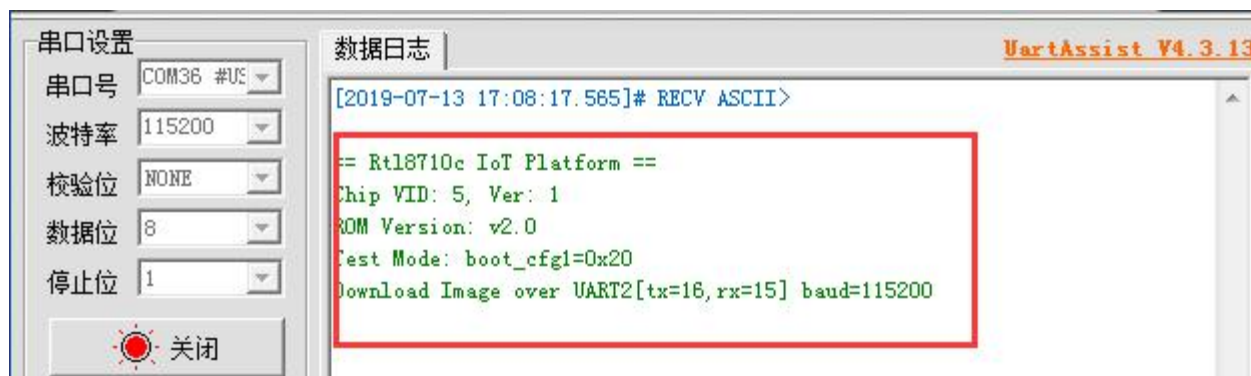
Flash Pin	IC part number
PIN_A7_A12	RTL8710CX/RTL8720CM
PIN_B6_B12	RTL8720CF

软件设置好后按照以下流程进行烧写:

第一步：开发板用 micro usb 线将开发板的 CON3 与 PC 端的 USB 口连接，连接好后开发板上的 D1 和 D2 亮，在电脑的设备管理器中能看到连接上的端口。J34 和 J33 用跳线帽按照下图跳线：

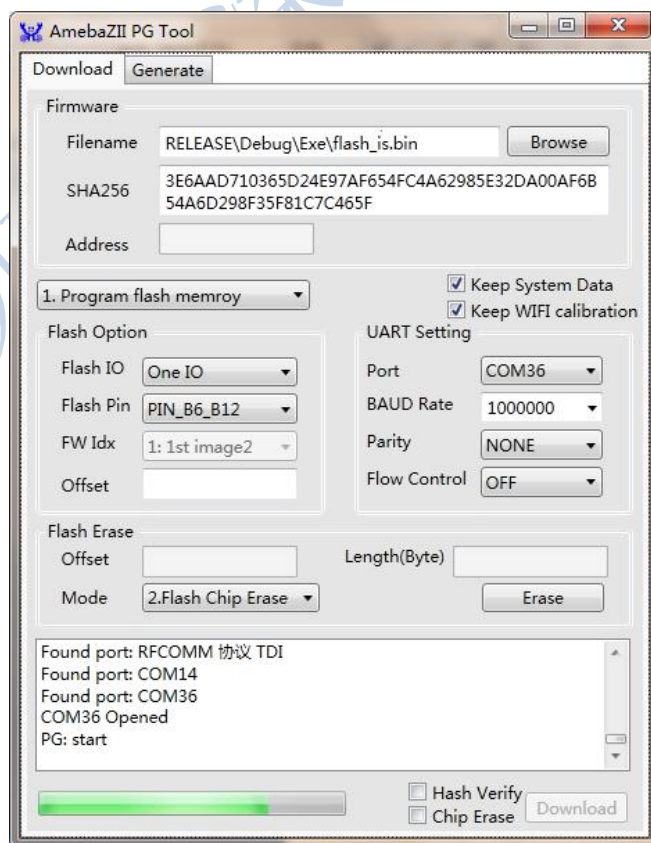


第二步：按下 UART DOWNLOAD Button 和 REST Button 后先松开 REST Button 再松开 UART DOWNLOAD Button，让开发板进入下载模式，在 PC 端用串口软件查看开发板的 log 信息，log 信息为下图所示表示已经进入下载模式：

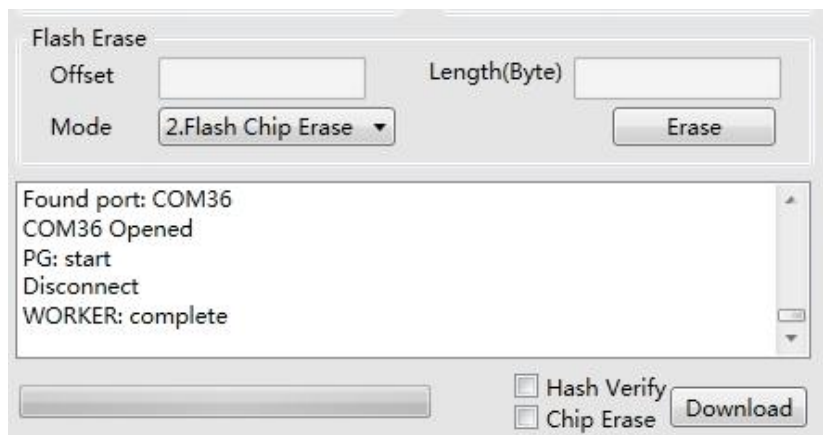


注意：看到以上打印信息后，要把串口助手关闭，不关闭的话串口被占用，PG tool 无法打开串口进行下载。

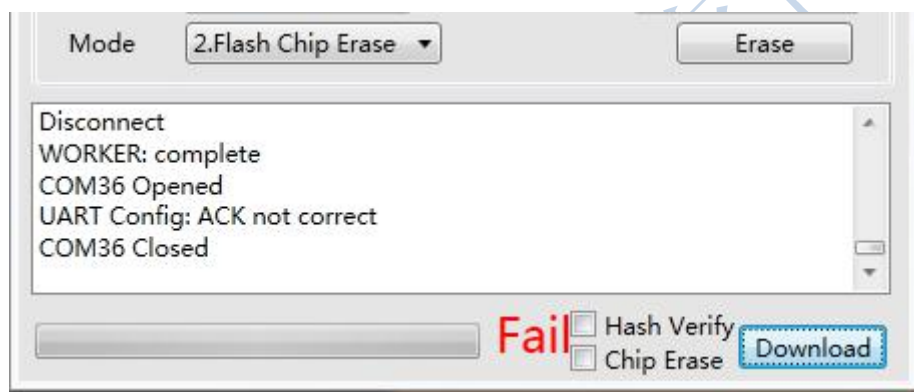
第三步：在 PG tool 中加载编译生成的 flash_is.bin 文件，选择 Program flash memory，点击 Download 按钮开始下载固件，如下图：



下载成功后信息提示框中显示如下：



下载失败显示如下：

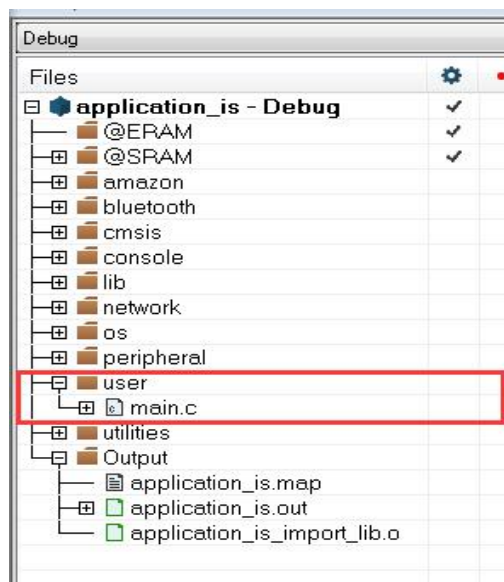


到此 SDK 的编译和固件的下载工作已经完成，下一章将介绍 SDK 的代码框架，如何开发 AT 指令集和蓝牙相关功能。

第 3 章 SDK 代码配置

3.1 SDK 入口

SDK 的入口函数在 main.c 文件中，在 IAR 工程中的位置如下图所示：



该文件的 main 函数是整个 SDK 的入口函数，在 main 函数中主要调用各个应用模块的入口 API，初始化和初始化 log 串口的命令任务，开启任务调度。

console_init: log 串口的初始化工作，创建 log 串口的 AT 命令服务线程，初始化完成后可以通过 log 串口发送指令实现相关功能。log 指令集的格式和说明参考 SDK 包的 doc 文件夹下的 AN0025 文档。

wlan_network: 初始化 Lwip 协议栈。

example_entry: SDK 功能模块入口函数，通过配置相关的宏定义来开启或关闭相关的功能，例如 AT 指令集，蓝牙 GATT 服务，各种云服务 demo，HTTP 等。

3.2 AT 指令集开发流程

3.2.1 v2.0 AT 指令集介绍

SDK 中有两套 AT 指令集，一套是基于 log 串口的，指令交互是通过 log 串口进行，参考文档是 doc 目录下的 AN0025。另一套是 v2.0 的指令集，基于独立的命令串口，指令集参考文档是 doc 目录下的 AN0075，两套指令集指令格式不一样，文档中有每条指令的使用方法。

3.2.2 v2.0 AT 指令集代码配置

第 1 步：打开 AT 指令集宏定义

v2.0 的 AT 指令集入口在 `example_entry.c` 文件中的 `example_uart_atcmd` 函数。SDK 默认是不开启指令集模块的，需要在 `platform_opts.h` 头文件中将 AT 指令集的配置宏打开，配置如下：

```
#define CONFIG_EXAMPLE_UART_ATCMD 1
```

第 2 步：配置命令串口参数

AT 指令是通过串口交互的，所以首先要配置一下串口参数，包括引脚，流控，波特率等，配置代码在 `example_uart_atcmd.c` 文件的 `uart_atcmd_main` 函数中：模块上电后将从 FLASH 读保存的串口参数进行配置，如果没有通过指令设置过串口参数，将使用默认的参数，默认参数如下：

波特率：38400

数据位：8

校验位：无



停止位：1

硬件流控：关闭

通过指令配置过参数后将保存到 flash 中，后面每次上电将读取 flash 的参数设置串口。

注意：由于串口有多组复用引脚，所以配置串口的引脚不要与其他功能引脚有冲突，引脚的配置在头文件 example_uart_atcmd.h 中，配置代码如下：

```
#define AMEBAZ2      4

#define CONFIG_AMEBA AMEBAZ2

#if (CONFIG_AMEBA == AMEBA1 )

#define UART_TX      PA_4

#define UART_RX      PA_0

#define UART_RTS     PA_2

#define UART_CTS     PA_1

#elif (CONFIG_AMEBA == AMEBAZ )

#define UART_TX      PA_23

#define UART_RX      PA_18

#define UART_RTS     PA_22

#define UART_CTS     PA_19

#elif (CONFIG_AMEBA == AMEBAD )

#define UART_TX      PA_18

#define UART_RX      PA_19

#define UART_RTS     PA_16
```



```
#define UART_CTS          PA_17

#elif (CONFIG_AMEBA == AMEBAZ2)

#define UART_TX           PA_14

#define UART_RX           PA_13

#define UART_RTS          NC

#define UART_CTS          NC

#endif
```

第 3 步：按前两步配置好代码后重新编译 SDK，将新的 bin 文件下载到开发板。

第 4 步：测试 AT 指令集

将命令串口的 Tx（PA_14）和 Rx（PA_13）连接到 TTL 电平转换工具的 Rx 和 Tx，在电脑端打开串口助手连接到开发板，按照 AN0075 文档中的指令格式给开发板发送 AT 指令。每条指令是以回车换行为结束标志，下面以连接路由器指令为例：

串口助手向开发板发送字符串：ATPN=99iot, 12345678

开发板收到字符串之后会连接名称为 99iot，密码为 12345678 的路由器，字符串后面要加上回车换行符<CR><LF>，不然无法识别指令。更多的指令测试参照九九物联提供的 AT 指令集文档和上手指南。

3.2.3 v2.0 AT 指令开发

SDK 集成了常用的连网，数据传输，参数设置等常用指令，用户也可以根据实际需求增加自己的指令。AT 指令的处理流程包括指令的接收，指令的解析和指令的执行，下面介绍分步介绍指令开发的流程。

1. 指令的定义

指令的添加需要先定义指令格式和指令的执行函数，按照约定的格式每条指令的前两个字母为“AT”，在 `atcmd_wifi.c` 文件的 `at_wifi_items` 结构体数组中进行定义，如下图：

```
/* NULL */
log_item_t at_wifi_items[ ] = {
#ifdef ATCMD_VER == ATVER_1
#ifdef CONFIG_LWIP_LAYER
    {"ATWL", fatwl, {NULL, NULL}},
    {"ATWI", fatwi, {NULL, NULL}},
    {"AIWT", fatwt, {NULL, NULL}},
    {"ATWU", fatwu, {NULL, NULL}},
#endif
#ifdef WIFI_LOGO_CERTIFICATION_CONFIG
    {"ATPE", fatpe, }, // set static IP for STA
#endif
#ifdef CONFIG_WLAN
    {"ATW0", fatw0, {NULL, NULL}},
    {"ATW1", fatw1, {NULL, NULL}},
    {"ATW2", fatw2, {NULL, NULL}},
    {"ATW3", fatw3, {NULL, NULL}},
#endif
}
```

“ATxx”为指令识别字符，“fATxx”为指令执行函数，用户自定义指令需要自行添加指令识别字符和实现指令执行函数。

2. 指令的接收、解析和执行

AT 指令的接收和数据打包在串口中断函数 `uart_irq` 中执行，中断函数在 `example_uart_atcmd.c` 文件中。中断函数中主要是将串口接收到的数据打包，以回车换行符为结束标志。接收完一条指令后通过发送信号 `rtw_up_sema_from_isr(&log_rx_interrupt_sema)` 通知指令处理线程进行指令的解析和处理。

指令的解析和执行函数在 `log_service` 函数中进行，`service` 函数等待串口接收中断函数发送 `log_rx_interrupt_sema` 信号，接收到信号后调用 `log_handler`



从指令列表中匹配对应的指令，匹配上指令之后调用对应的指令处理函数实现相关的功能。

3.3 蓝牙配网功能

3.3.1 蓝牙配网功能介绍

Ameba-ZII 硬件和软件上集成和 WIFI 通信和蓝牙通信功能，SDK 默认上电是开启 WIFI 关闭蓝牙功能，用户可以通过配置 SDK 的代码来开启蓝牙功能。给 WIFI 芯片配网是 WIFI 通信中重要的一步，通过多种方式将路由器的 SSID 和密码传给 WIFI 模块，WIFI 模块再去连接路由器，从而实现上网功能。

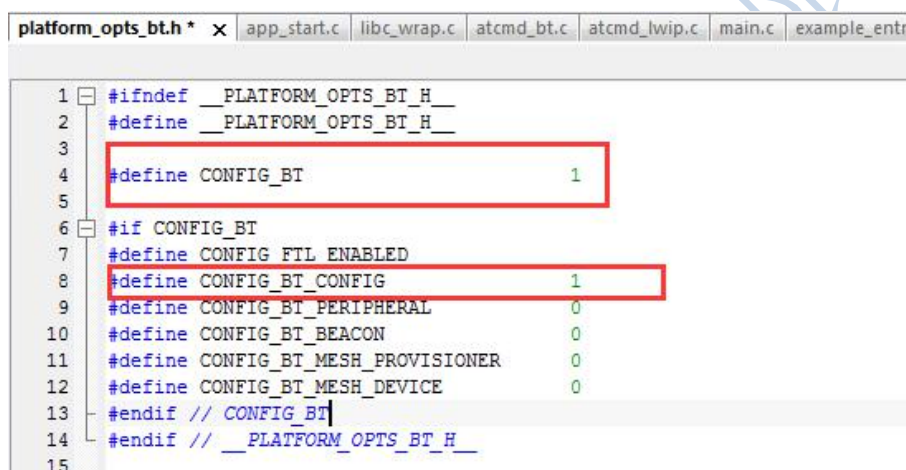
传统的单 WIFI 芯片的配网方式有以下几种方式：

1. Simpleconfig 方式：WIFI 模块端进入混杂模式监听周围的广播包，手机 APP 端通过将路由器信息加入到广播包的指定字段，向周围设备发送广播。模块监听到广播之后按照指定的协议解析数据包提取需要的路由器信息进行连接路由器。
2. AP 方式：WIFI 模块端自身起 AP 热点，手机连接上模块的热点跟模块建立 UDP 通信，将路由器信息按指定的格式传送给 WIFI 模块，模块收到后提取路由信息进行连接。
3. AT 指令配网：通过命令串口发送配网指令 ATPN 将路由 SSID 和密码传给模块，模块收到路由信息后执行连接路由器的操作。

4. 蓝牙配网：WIFI 模块开启蓝牙配网功能，手机开启蓝牙并连接上模块的 GATT Server，APP 将路由信息写入指定的 service，模块端读取到路由信息后执行连接路由器的操作，从而实现蓝牙配网功能。

3.3.2 开启蓝牙配网功能

SDK 中的蓝牙功能默认是关闭的，要开启蓝牙配网首先要在 platform_opts_bt.h 文件中打开相关的宏定义，宏配置如下：



```
1 #ifndef __PLATFORM_OPTS_BT_H__
2 #define __PLATFORM_OPTS_BT_H__
3
4 #define CONFIG_BT 1
5
6 #if CONFIG_BT
7 #define CONFIG_BT_ENABLED 1
8 #define CONFIG_BT_CONFIG 1
9 #define CONFIG_BT_PERIPHERAL 0
10 #define CONFIG_BT_BEACON 0
11 #define CONFIG_BT_MESH_PROVISIONER 0
12 #define CONFIG_BT_MESH_DEVICE 0
13 #endif // CONFIG_BT
14 #endif // __PLATFORM_OPTS_BT_H__
15
```

按照以上配置好代码重新编译 SDK 并下载固件到 flash 中，即可进行蓝牙配网，蓝牙配网的步骤如下：

第 1 步：通过命令串口向模块发送 AT 指令 ATBB=1 开启蓝牙配网功能。

第 2 步：手机开启蓝牙，打开蓝牙配网 APP “WiFiConfig” 软件界面如下图：





第 3 步：点击软件界面中的搜索图标 搜索周围的蓝牙设备，搜索到设备后 APP 自动连接上设备。

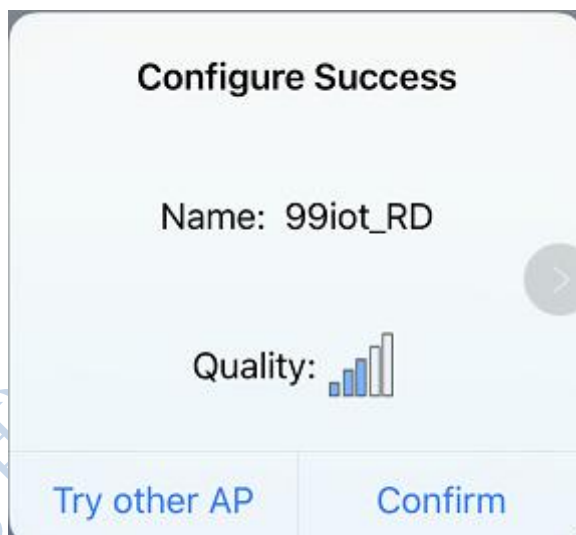
第 4 步：APP 端通过蓝牙向模块发送扫描周围 wifi 热点的请求，模块扫描完后把当前周围存在的 wifi 设备列表上传给 APP 并显示出来，如下图：



第 5 步：在 APP 端选取设备要连接的热点并输入 wifi 密码，按 Next 后 APP 端下发 SSID 和密码信息给模块。



第 6 步：模块收到信息后连接路由并返回连接结果给 APP，连接上路由后在 APP 端确认，配网流程结束。



配网成功后模块退出蓝牙模式，此时模块已经成功连接上指定的路由器，用户可以使用 AT 指令进行 TCP/UDP 建连，数据传输等操作。

3.4 蓝牙 BLE Peripheral 应用例程

3.4.1 蓝牙 BLE Peripheral 概述

Ameba-ZII 可以作为 BLE 外围设备向外界发送广播和提供 GATT 服务，用户可以自定义自己需要的 GATT 服务。模块作为服务端，通过提供 GATT 服务跟连接进来的客户端进行数据交互，消息订阅、推送、读、写等操作。

3.4.2 蓝牙 BLE Peripheral 代码分析

BLE Peripheral 应用的代码在 SDK 中的路径为：

component\common\bluetooth\realtek\sdk\example\ble_peripheral。

开启应用例程需要打开 platform_opts_bt.h 头文件下的相关宏定义，代码如下：

```
1  #ifndef __PLATFORM_OPTS_BT_H_
2  #define __PLATFORM_OPTS_BT_H_
3
4  #define CONFIG_BT 1
5
6  #if CONFIG_BT
7  #define CONFIG_FTL_ENABLED
8  #define CONFIG_BT_CONFIG 0
9  #define CONFIG_BT_PERIPHERAL 1
10 #define CONFIG_BT_BEACON 0
11 #define CONFIG_BT_MESH_PROVISIONER 0
12 #define CONFIG_BT_MESH_DEVICE 0
13 #endif // CONFIG_BT
14
15 #endif // __PLATFORM_OPTS_BT_H_
```

配置好以上宏之后重新编译 SDK 并将新的固件下载到开发板，重新上电后程序会调用应用的入口函数 ble_app_main 执行相关的初始化，ble_app_main 函数位于 ble_app_main.c 文件中。


```
int ble_app_main(void)
{
    bt_trace_init();
    bt_stack_config_init();
    bte_init();
    board_init();
    le_gap_init(APP_MAX_LINKS);
    app_le_gap_init();
    app_le_profile_init();
    pwr_mgr_init();
    task_init();

    return 0;
}
```

GAP 和 GATT Profiles 初始化:

le_gap_init: 初始化 GAP 并设置 link 数为 1.

app_le_gap_init: GAP 参数的初始化, 用户可以通过修改以下参数来自定义应用:

(1) Device Name 和 Device Appearance 设置设备的名称和类型。

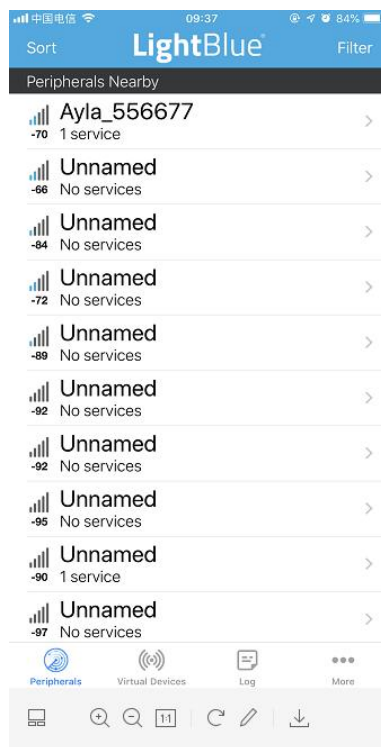
(2) Advertising 设置广播参数

(3) Bond Manager 参数

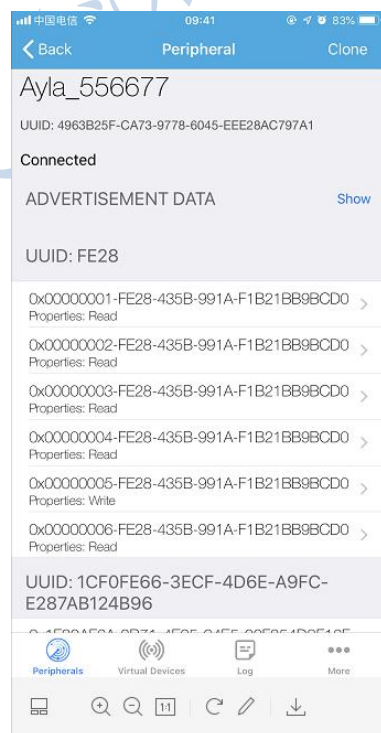
app_le_profile_init: 初始化基于 GATT 的 Profile, 添加用户需要的服务列表和注册回调函数 app_profile_callback, 在回调函数中处理消息的读、写、推送等操作。

3.4.3 蓝牙 BLE Peripheral 测试

开启 BLE Peripheral 应用开发板上电后会创建一个 BLE 外围设备, 并向周围发出广播, 手机应用可以扫描到该设备并创建连接。用户可以使用蓝牙测试 APP “LightBlue” 进行测试, 在 AppStore 和安卓市场上都能下载 LightBlue。打开软件界面如下图所示:



APP 能扫描出周围的蓝牙设备，设备名称为“Ayla_556677”为我测试用开发板设备的名称，点击后可连接该设备，连接上之后可读取到该蓝牙设备提供的所有 GATT 服务列表，如下图所示：



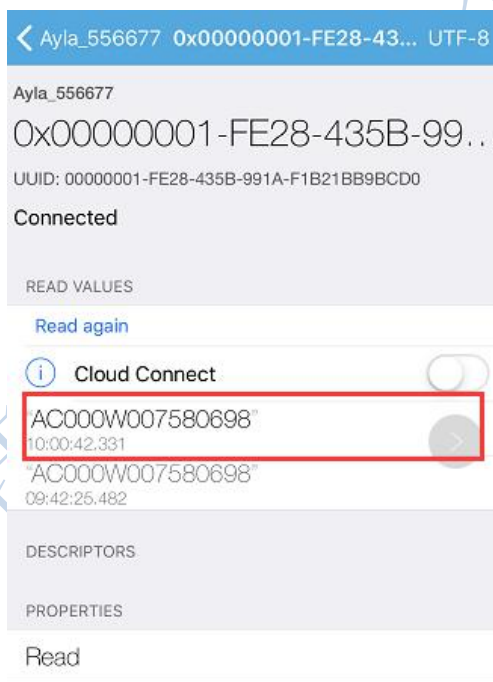
每个 Service 的结构包括 Service 的 UUID，Service 下的 Characteristic UUID，

和 Characteristic 属性等。

Characteristic 属性有读、写、通知及三种属性的组合，例如 UUID 为 0X0C000001-FE28-435B-991A-F1B21BB9BCD0 的 Characteristic 为可读属性

0x00000001-FE28-435B-991A-F1B21BB9BCD0 >
Properties: Read

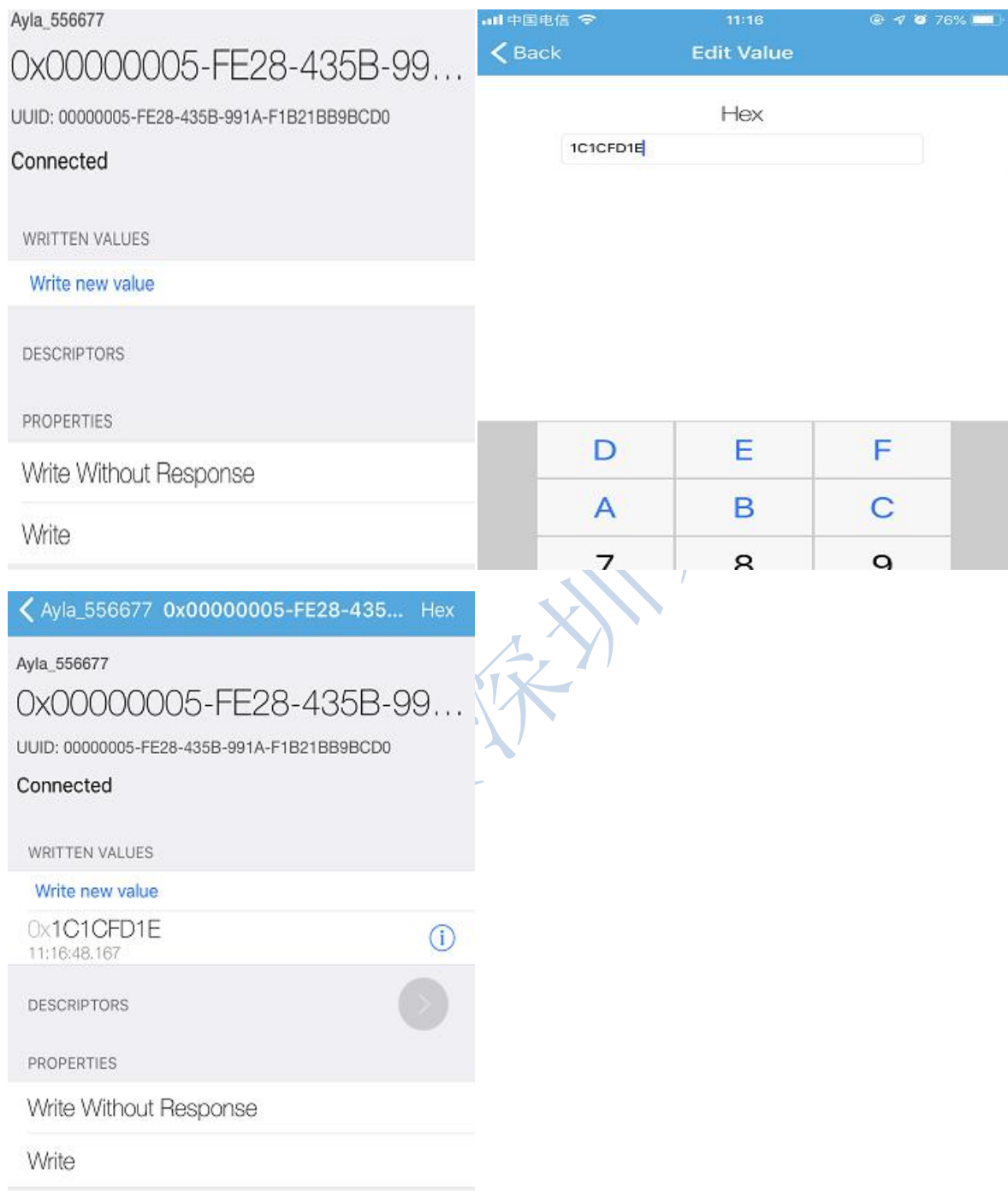
点击该 Characteristic 进入到该 Characteristic 的属性界面，点击 Read again 读取该 Characteristic 的值，Characteristic 值在注册的回调函数 app_profile_callback 中调用 simp_ble_service_set_parameter 将要发送给 APP 的数据写入即可，如下图“AC000W007580698”为 APP 收到模块端写入的数据：



如果 Characteristic 的属性为 Write，APP 端可以通过在 Characteristic 中写入数据下发给模块端，如下图：

0x00000004-FE28-435B-991A-F1B21BB9BCD0 >
Properties: Read
0x00000005-FE28-435B-991A-F1B21BB9BCD0 >
Properties: Write
0x00000006-FE28-435B-991A-F1B21BB9BCD0 >

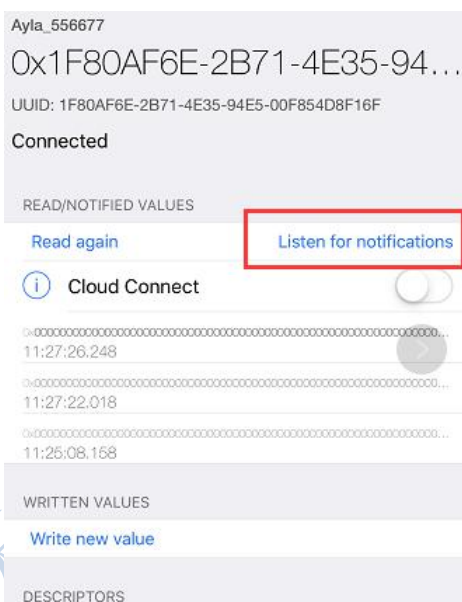
进入该 Characteristic 有“Write new value”按钮，点击之后可编辑要下发的数据，如下图：



Characteristic 的属性有 Notify 时,手机端开启对该 Characteristic 的监听后,模块端可以主动推送数据给手机端,不需要主动读取。



APP 端进入该 Characteristic 界面,点击“Listen for notifications”按钮开启对 Characteristic 的监听,有消息推送 APP 端就能收到推送的数据如下图:



到此,AT 指令集,蓝牙配网,蓝牙数据传输应用的代码配置和测试流程已介绍完。Ameba-ZII 的 SDK 还有许多的功能默认是关闭的,如亚马逊,谷歌的云应用,HTTTPC,HTTTPD,OTA 升级,SPI 通信等应用 demo。这些 demo 的入口都在 example_entry.c 文件中调用,通过配置相关宏定义来开启。芯片外设接口的使用例程位于目录 project\realtek_amebaz2_v0_example\example_sources 中,用户在开发 SDK 的过程中如果要使用到不同外设,可以参照 example_sources 中的 demo 例程。



九九物联专业开发 WIFI 底层固件，APP 和对接各大云平台，我们的 Ameba-ZII 已经应用到实际的智能家居产品中，用户在开发 SDK 的过程中遇到什么问题可以联系我们，加入到我们的技术讨论群，众多工程师和开发者在线为您解答。

九九物联（深圳）有限公司