

# M5311 OPENCPU 开发指导 手册

NB-IoT 模组

版 本：V1.2.0

日 期：2019-4-19

中移物联网有限公司

[iot.10086.cn](http://iot.10086.cn)



# 重要声明

## 版权声明

本文档中的任何内容受《中华人民共和国著作权法》的保护，版权所有 © 2018，中移物联网有限公司，保留所有权利，但注明引用其他方的内容除外。

## 商标声明

中移物联网有限公司和中移物联网有限公司的产品是中移物联网有限公司专有。在提及及其他公司及其产品时将使用各自公司所拥有的商标，这种使用的目的仅限于引用。

## 不作保证声明

中移物联网有限公司不在此文档中的任何内容作任何明示或暗示的陈述或保证，而且不对特定目的的适销性及适用性或者任何间接、特殊或连带的损失承担任何责任。

## 保密声明

本文档（包括任何附件）包含的信息是保密信息。接收人了解其获得的本文档是保密的，除用于规定的目的外不得用于任何目的，也不得将本文档泄露给任何第三方。

## 关于文档

### 修订记录

版本	日期	作者	描述
1.0.0	2019.1.16	余洋意	首次创建
1.1.0	2019.1.22	谢刚亮	补充整理内容
1.1.1	2019.2.26	谢刚亮	增加一些提示点
1.2.0	2019.4.19	谢刚亮	增加 7.8 节，log 相关



## 目录

关于文档 .....	2
1. 概述.....	4
2. M5311 OpenCPU SDK 开发包介绍 .....	4
3. gcc 编译环境安装 .....	5
4. 程序编译.....	5
5. 系统烧录.....	6
6. DEMO 例程演示 .....	9
7. SDK 开发指导 .....	10
7.1.OpenCPU 模式启动方法 .....	10
7.2.示例程序执行流程.....	10
7.3.SDK API 函数功能测试方法 .....	11
7.4.增加源文件方法.....	11
7.5.M5311 OPENCPU 低功耗模式下的编程 .....	11
7.6.RIL 层功能使用方法.....	12
7.7 SDK 硬件版本说明 .....	13
7.8 SDK 调试及疑难问题解决 .....	13
8. SDK API 分类说明 .....	13
8.1 模组操作系统.....	13
8.2 模组网络相关接口.....	14
8.3 模组基础功能接口.....	14
8.4 硬件外设接口.....	14
8.5FOTA 接口 .....	14
9. LOG 工具使用 .....	14
9.1 驱动安装.....	14
9.2 工具使用.....	15

## 1. 概述

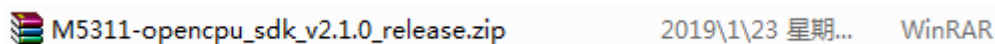
本文档介绍在 Windows7/Windows10 环境下开发基于 M5311 OpenCPU SDK 的应用程序。

M5311 OpenCPU SDK 提供交叉编译链、所需的库文件和头文件、API 和 API 调用示例程序，这些 API 可以实现客户相应的需求。所有的内容将以 SDK 安装包的形式提供给客户，客户只需要按照使用向导解压 SDK 包(注:SDK 的安装路径中不能包含中文和空格)，然后进行应用程序编写，编译完成后烧入模组开发板中即可调试使用。

M5311 OpenCPU SDK 的开发环境位于 sdk 目录下。

## 2. M5311 OpenCPU SDK 开发包介绍

1、解压开发包 ZIP 包文件（版本请以实际为准）：



2、解压后得到如下文件（不同版本会有细微差别）：

名称	修改日期	类型	大小
apb	2019\1\23 星期...	文件夹	
firmware	2019\1\24 星期...	文件夹	
fota	2019\1\25 星期...	文件夹	
inc	2019\1\22 星期...	文件夹	
lib	2019\1\24 星期...	文件夹	
lwip	2018\12\28 星期...	文件夹	
os	2018\12\28 星期...	文件夹	
out	2019\1\24 星期...	文件夹	
src	2019\1\24 星期...	文件夹	
utils	2019\1\24 星期...	文件夹	
build.bat	2019\1\24 星期...	Windows 批处理...	2 KB
clean.bat	2019\1\25 星期...	Windows 批处理...	1 KB
config.mk	2019\1\25 星期...	Makefile	1 KB
do_check.bat	2019\1\24 星期...	Windows 批处理...	1 KB
Makefile	2019\1\24 星期...	文件	2 KB
readme.txt	2019\1\25 星期...	文本文档	1 KB

进入 SDK 目录：

- “apb” 目录是基础功能相关头文件
- “firmware” 目录是编译后固件生成的目录
- “fota” 目录是 FOTA 升级相关目录
- “inc” 目录是 user 头文件目录

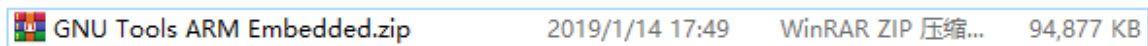
“lib”	目录是 SDK 部分库文件
“lwip”	目录是 lwip 协议栈头文件
“os”	目录是操作系统相关头文件
“out”	目录是部分库文件及编译结果目录
“src”	目录是 user 源文件目录
“utils”	目录是编译环境相关文件

其中 `buil.bat` 文件为编译脚本，Windows 环境安装 gcc 编译环境后可编译软件版本。

## 3.gcc 编译环境安装

客户首次进行 OpenCPU 开发，需准备好编译环境。

- ① 在工具附录包中找到 “GNU Tools ARM Embedded.zip” 文件并将 zip 包解压至 “C:\Program Files (x86)” 目录下。



- ② 将 “C:\Program Files (x86)\GNU Tools ARM Embedded\4.9 2015q3\bin” 路径添加至系统环境变量。
- ③ 添加完环境变量后，重启电脑完成安装。

到此，OpenCPU 编译环境安装完成，可以使用下一章节程序编译来检测编译工具是否安装成功。

## 4.程序编译

首先确认要使用的硬件版本，以 LV 为例，编译命令为：`build.bat LV`

在命令行中进入 SDK 主目录，输入 `build.bat LV` 命令后回车，即可执行编译：



```
D:\ME311-opencpu_sdk_v2.1.0_release>build.bat LV
Making LV Version
'compile src/opencpu_base_func_demo.c...'
'compile src/m5311_opencpu.c...'
'compile src/opencpu_other_dev_demo.c...'
'compile src/opencpu_iic_demo.c...'
'compile src/opencpu_network_demo.c...'
'compile src/opencpu_fota_demo.c...'
'compile src/onenet_test.c...'
'compile src/opencpu_gpio_demo.c...'
'compile src/opencpu_spi_demo.c...'
'compile src/opencpu_uart_demo.c...'
"linking LV libs"
D:\ME311-opencpu_sdk_v2.1.0_release>
```

编译成功会在 firmware/LV 目录下生成 nbiot\_m2m\_demo.bin 固件

名称	修改日期	类型	大小
flash_download.cfg	2018\12\28 星期...	CFG 文件	2 KB
mt2625_bootloader.bin	2018\12\28 星期...	BIN 文件	46 KB
nbio_m2m_demo.bin	2019\1\25 星期...	BIN 文件	2,149 KB
nbio_m2m_demo.elf	2019\1\25 星期...	ELF 文件	31,514 KB

在命令中输入 `.\build.bat CLEAN` 命令后回车，即可删除编译结果

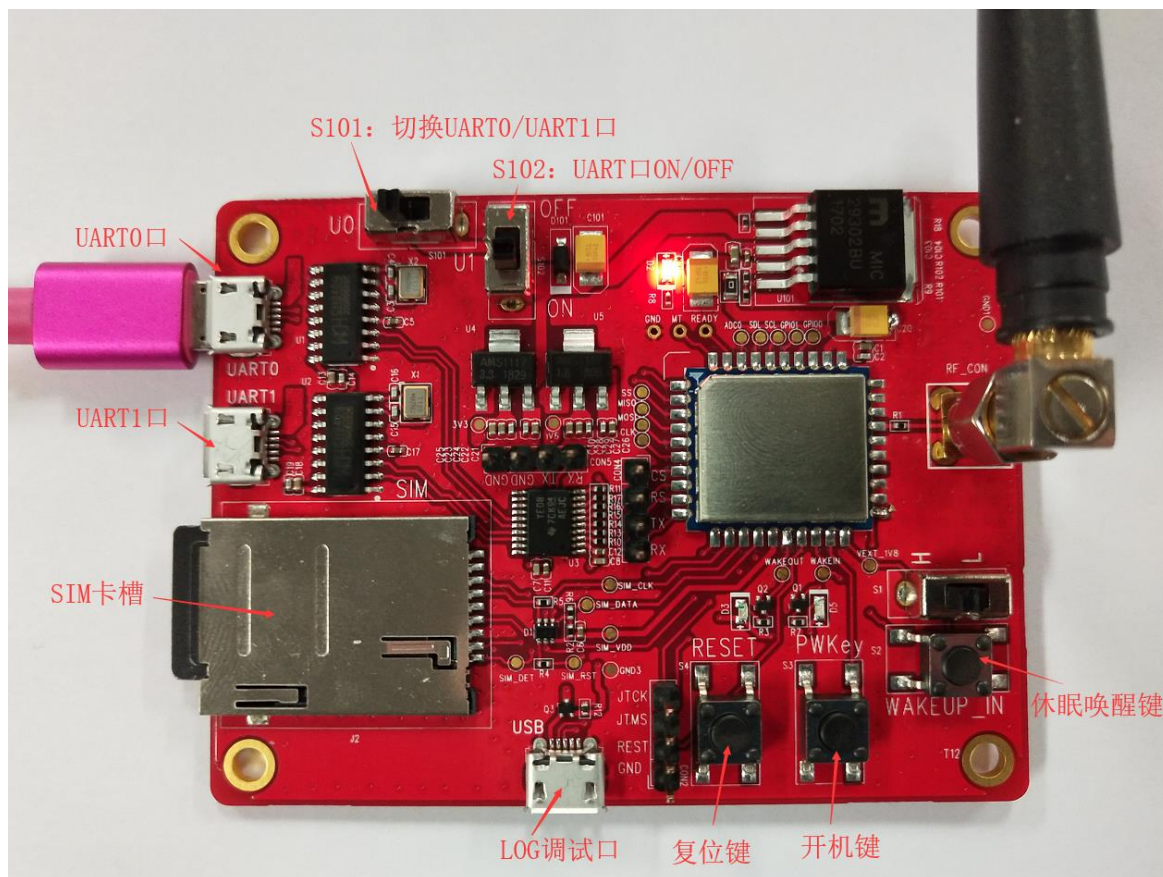
```
D:\opencpu1.7_fota>build.bat CLEAN
clean ends
```

目前编译脚本支持三个参数（区分大小写）：  
 build.bat LV:编译生成硬件版本为 LV 的固件  
 build.bat CM:编译生成硬件版本为 CM 的固件  
 build.bat CLEAN:清除编译结果

## 5. 系统烧录

编译完版本，生成 BIN 文件之后，将版本文件烧录至开发板：

- ① 如下图，将开发板 UART0 口通过 micro-USB 线连接至 PC，板子左上方两个切换开关分别切换至 U0 和 ON，板子通电指示灯亮起。

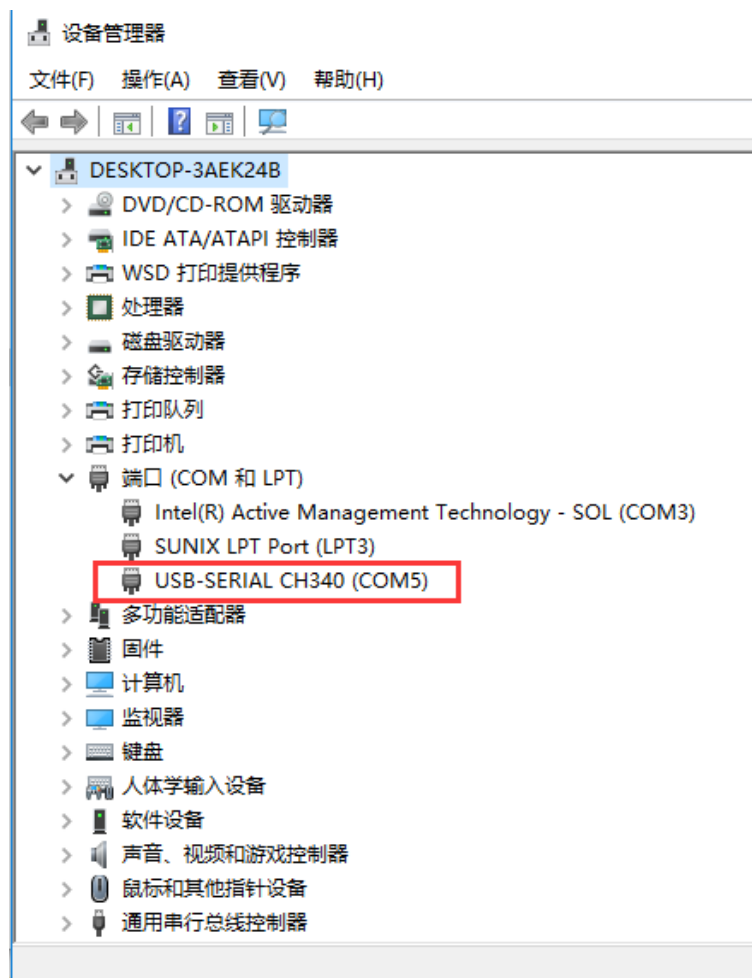


- ② 安装 CH340 串口驱动程序（如已安装则跳过此步骤）  
安装 CH340 驱动程序，安装成功后，设备管理器中会出现相关 COM 口。

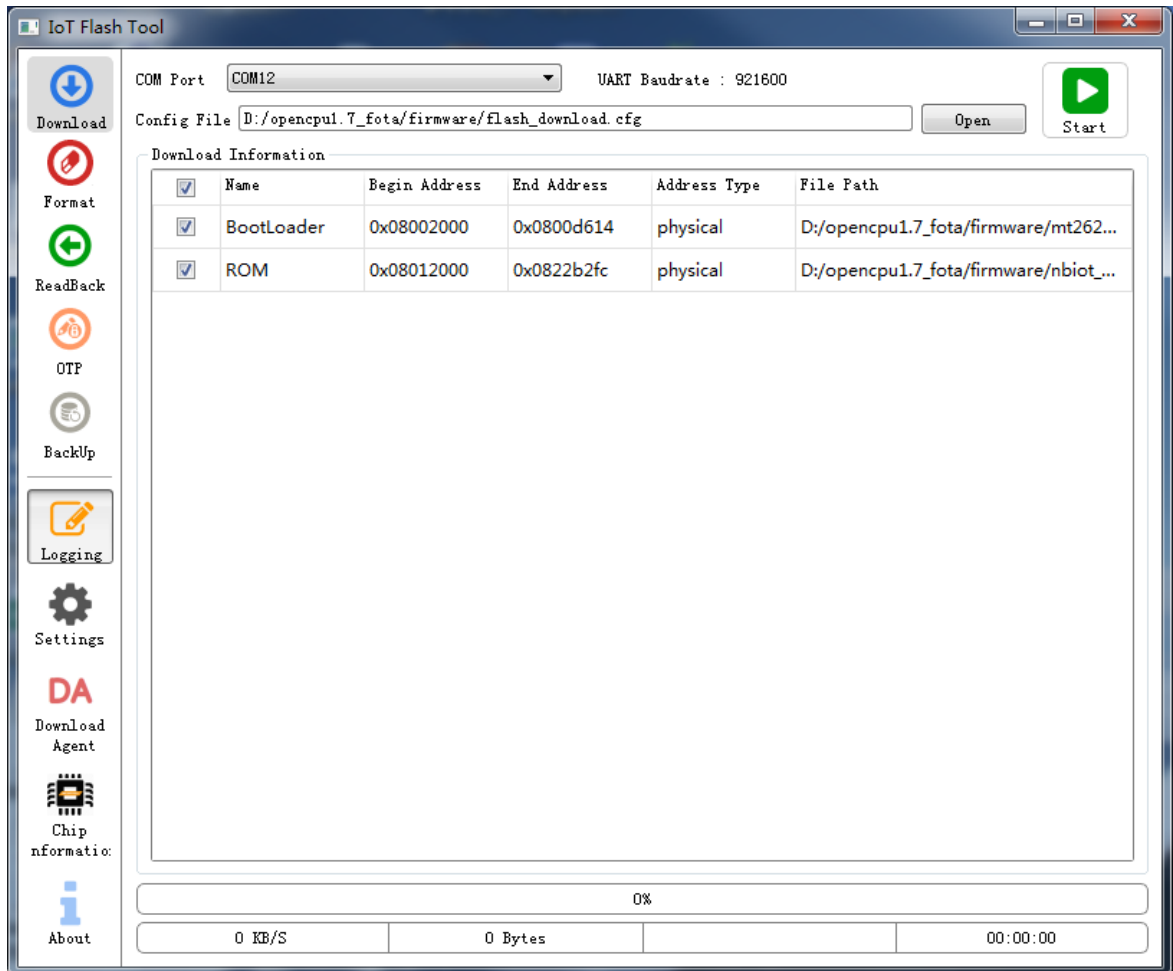
CH341SER.EXE 2018/12/25 9:37 应用程序 238 KB







- ③ 打开 FlashTool.exe 下载工具，选择好 COM 口之后，在 Config File 栏选择 firmware 目录下的 flash\_download.cfg 文件，工具会自动加载目录下 BootLoader 文件和 ROM 文件，点击 start 键，然后并长摁开发板右下角 PWKey 键 2S 开机（如果已经开机，则按一下 RESET 按键即可），直至出现烧录进度条。进度条完成之后即下载成功



## 6.DEMO 例程演示

开发板 SDK 中默认代码有简单测试例程，将版本烧录至开发板后，将 micro-USB 线连接至 UART1 口，打开串口工具，选择对应 COM 口，波特率设置为 115200。

打开串口后，长摁 PWKey 键 1-2 秒，系统启动后会通过串口打印初始化信息，串口助手接收界面会出现如下图所示打印信息。

图中例子：发送 M，返回单板 IMEI 信息。

```
[2019-01-16_18:00:41:251]BOOT CAUSE:POWER_ON OR RESET
[2019-01-16_18:00:41:254]M5311 opencpu ready!!
[2019-01-16_18:00:41:258]waiting for network...
[2019-01-16_18:00:43:499]ICCID:89860417221890133892
[2019-01-16_18:00:43:499]network registering...
[2019-01-16_18:00:47:958]network register success
[2019-01-16_18:00:47:958]network ready!!
[2019-01-16_18:00:47:962]please input cmd$Mresult:0
[2019-01-16_18:00:53:898]IMEI:869975030076338
```

## 7.SDK 开发指导

### 7.1.OpenCPU 模式启动方法

在代码中有 `get_factory_mode()` 函数，该函数在 SDK 内部调用，在启动的时候进行判断：

当该函数返回 1 时，第一次把 OpenCPU 下载到新模组上时，模组会默认以 AT 模式运行，不会启动 OpenCPU 模式，此时用户需要在 UART1 手动执行 `AT+ATCLOSE`，然后按复位键，模组即启动 OpenCPU 模式，并且以后都会保持 OpenCPU 模式。要从 OpenCPU 模式切换回 AT 模式，需要在 OpenCPU 程序里执行 `opencpu_at_open()` 函数，此时模组会自动重启，并以 AT 模式运行。

当该函数返回 0 时，任何情况下模组以 OpenCPU 方式运行。

### 7.2.示例程序执行流程

以下代码在模组主程序的 main 函数中被调用，不能阻塞。这个函数新建了一个任务，该任务是 OPENCPU 用户添加定制化代码的地方。

```
void test_opencpu_start()
{
    xTaskCreate(opencpu_task_main,"opencpu",1024,NULL,TASK_PRIORITY_NORMAL,NULL);
}
```

以下代码是用户任务的函数体，该任务在整个操作系统开始调度后作为一个普通任务与其他系统任务一起运行，用户的所有操作在该函数中进行，这个任务如果退出，则代表用户执行流也退出了。

```
void opencpu_task_main()
{
    test_all_in_one();
    vTaskDelete(NULL);
}
```

## 7.3.SDK API 函数功能测试方法

在示例代码中，`test_all_in_one()`函数是用户执行功能测试的函数，该函数 `opencpu_task_main()` 任务中调用，模组开机就开始执行。

在该函数中，首先初始化了串口 1，设置了串口 1 回调函数，然后打印了一些提示信息，  
然后锁定睡眠，使模组不能进入睡眠。

以阻塞方式获取 ICCID，如果没插卡，则程序不能继续往下执行，以阻塞方式等待程序驻网 (CGACT)，如果驻网不成功，则程序不能继续往下执行。

驻网成功后，循环等待用户命令。用户此时可在串口 1 输入单字符命令(A~Z,a~z,0~9)以执行相应的功能函数

## 7.4.增加源文件方法

如在 `src` 目录下添加源文件，可自动被加入编译。

如需在增加新目录，则需将该目录相对 SDK 根目录的相对路径填入 `Makefile` 中，添加方法如下：

比如要在 `src` 目录下新增 `src` 目录，则将该目录的相对路径添加到 `SRC_DIRS` 下，以空格分隔。

```
SRC_DIRS := src src/src
```

新增的源文件目录也会被自动添加到头文件搜索目录。

## 7.5.M5311 OPENCPU 低功耗模式下的编程

如果用户不使用低功耗模式 (PSM)，则在程序开头调用 `opencpu_lock_light_sleep()`，程序将不会进入低功耗模式。此时模组编程模式不受低功耗限制。

伪代码示例：

```
user_task
{
    opencpu_lock_light_sleep();
    user_app();
}
```

如果用户需要使用低功耗模式 (PSM)，则编程模式将会受限。模组在低功耗模式下，RAM 断电，除 RTC 外的所有外设也断电，程序停止运行。从低功耗模式唤醒，模组的程序会从头开始执行，模组从低功耗模式唤醒的方式有：

1. wakeup 引脚低电平
2. RTC 定时器到期
3. PSM 睡眠时间到期。

异步唤醒只能通过 wakeup 引脚低电平来唤醒，同步唤醒可以靠 2、3 两种方式。模组的启动原因可以通过接口 `opencpu_is_boot_from_sleep()` 来查询。

所以在需要使用低功耗模式的情况下，用户需要首先调用 `opencpu_lock_light_sleep()` 来防止模组进入睡眠，然后确认 PSM 参数已生效，未生效则需要设置。然后执行用户的应用逻辑程序，执行完成后，如果有关键数据和状态需要保存，可以保存到 flash 之中，flash 使用方法请参考 `m5311_opencpu.c` 中示例和文档。调用 `opencpu_unlock_light_sleep()` 来取消睡眠禁止，等待模组睡眠。调用 `opencpu_unlock_light_sleep()` 后，用户不应再主动进行任何程序操作，如有异步事件发生造成了模组唤醒，则应首先锁定睡眠，事件处理完成后再释放睡眠。

伪代码示例：

```
User_task
{
    opencpu_lock_light_sleep();
    opencpu_set_psm();
    user_app();
    data_and_state_save();
    opencpu_unlock_light_sleep();
    task_exit();
}
```

## 7.6.RIL 层功能使用方法

RIL 层是 SDK 开放给用户执行一些 AT 命令的接口，用户可参考 `inc/apb/ril.h`，完成相关 AT 命令的发送和结果获取。比如，获取模组 ICCID 的 RIL 层函数如下：

```
static char opencpu_iccid_buf[30];
//该函数为 AT 命令执行结果的回调函数
int32_t opencpu_iccid_callback(ril_cmd_response_t *response)
{
    ril_read_usim_iccid_rsp_t *param = (ril_read_usim_iccid_rsp_t*)response->cmd_param;
    //如果执行结果为成功且返回参数不为空
    if (response->res_code == RIL_RESULT_CODE_OK && param != NULL){
        memcpy(opencpu_iccid_buf, param->iccid, strlen(param->iccid));
    }
    else
    {
        return -1;
    }
    return RIL_STATUS_SUCCESS;
}
//用户调用此函数获取 ICCID
int opencpu_iccid(unsigned char *buf)
```

```
{  
int i = 0;  
memset(opencpu_iccid_buf,0,30);  
//执行获取 ICCID 的 AT 命令  
ril_request_read_usim_iccid(RIL_ACTIVE_MODE,opencpu_iccid_callback,NULL);  
//等待回调函数执行完成  
while(opencpu_iccid_buf[0] == 0)  
{  
i++;  
vTaskDelay(10);  
if(i>20)  
{  
return -1;  
}  
}  
strcpy(buf,opencpu_iccid_buf);  
return 0;  
}
```

其他 AT 命令，用户可首先在 ril.h 中搜索命令关键字，找到相关函数和结构体，然后按上述结构实现即可。

## 7.7 SDK 硬件版本说明

目前 M5311 支持 OpenCPU 开发的硬件版本有 CM、LV 版本，开发前请确认要使用的版本，然后再选择正确的编译参数。

## 7.8 SDK 调试及疑难问题解决

目前模组支持 GKI 和 HSL log 输出，输出的串口可根据示例程序中进行配置。客户可使用 opencpu\_log 函数进行 log 输出。如出现客户无法排除的 SDK 问题，可抓取 log，并保留出现问题的 elf 文件，交给我司进行分析。

# 8.SDK API 分类说明

## 8.1 模组操作系统

M5311 OPENCPU SDK 使用 FreeRTOS V8.2.0 嵌入式操作系统，所有的接口和标准 FREERTOS 一致，遇到相关问题，开发者可登录 <https://www.freertos.org/> 搜索相关资料解决，也可联系我们解



决。相关结构体、参数等可以参考 os 目录下头文件。

部分主要函数接口请参见文档：

《M5311 OPENCPU MANUAL.chm》，Modules/FUNCTIONS/RTOS\_INSTRUCTION

## 8.2 模组网络相关接口

M5311 OPENCPU SDK 使用 LWIP TCP/IP 协议栈，socket 接口符合 POSIX 标准，用户可参照相关例程进行应用程序编写。例程请参考文档：

《M5311 OPENCPU MANUAL.chm》，Modules/NETWORK\_FUNCTIONS

## 8.3 模组基础功能接口

模组基础功能接口包含基本的模组信息获取，模组控制，模组状态设置等接口。API 见

《M5311 OPENCPU MANUAL.chm》，Modules/BASE\_FUNCTIONS

## 8.4 硬件外设接口

硬件外设接口包括 GPIO、I2C、SPI、PWM、ADC、RTC、FLASH、UART 等，相关 API 请参见

《M5311 OPENCPU MANUAL.chm》，Modules/，

接口说明见文档：《M5311 OpenCPU 资源综述.pdf》

## 8.5 FOTA 接口


M5311 FOTA 方案基于中国移动 onenet 云平台，具体使用方法参见文档：

《M5311 OneNet\_FOTA 使用手册.pdf》

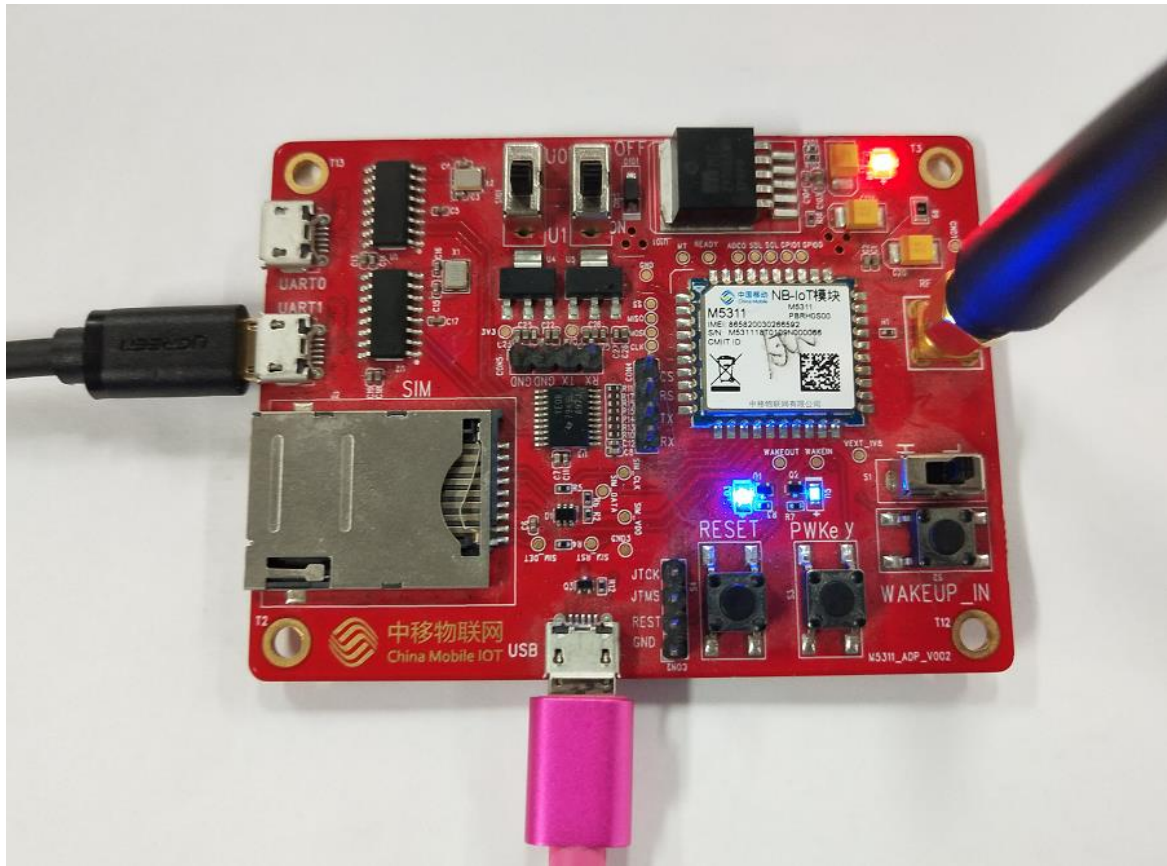
# 9.LOG 工具使用

## 9.1 驱动安装

驱动文件如下，将文件解压后得到安装文件，进行安装：

 MTK\_USB\_COM\_Driver\_SDK\_v3.16.46.1\_HCK.rar


驱动安装成功后，将开发板 USB 口连接至电脑，按电源键开机后，开发板开机键上方蓝色 LED 会点亮，设备管理器中会出现对应 COM 口。



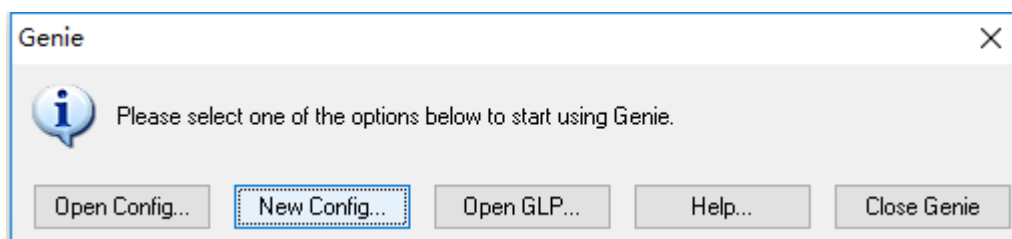
- 端口 (COM 和 LPT)
  - Intel(R) Active Management Technology - SOL (COM3)
  - SUNIX LPT Port (LPT3)
  - USB Debug Port (COM7)
  - USB Modem Port (COM8)
  - USB-SERIAL CH340 (COM6)

## 9.2 工具使用

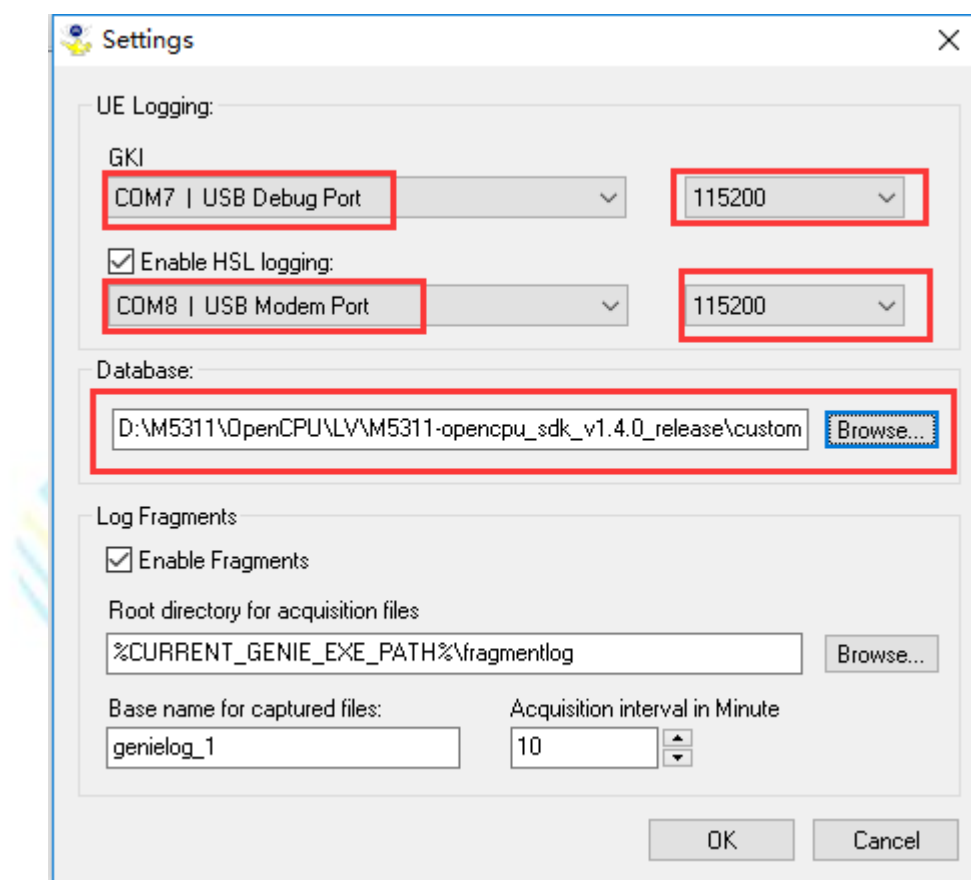
log 工具文件包位于 SDK 工具包下如下图文件，将文件解压后得到 genie log 抓取工具：

 MT2625\_nbiot\_tools\_20180929\_exe\_V1.1839.5.zip

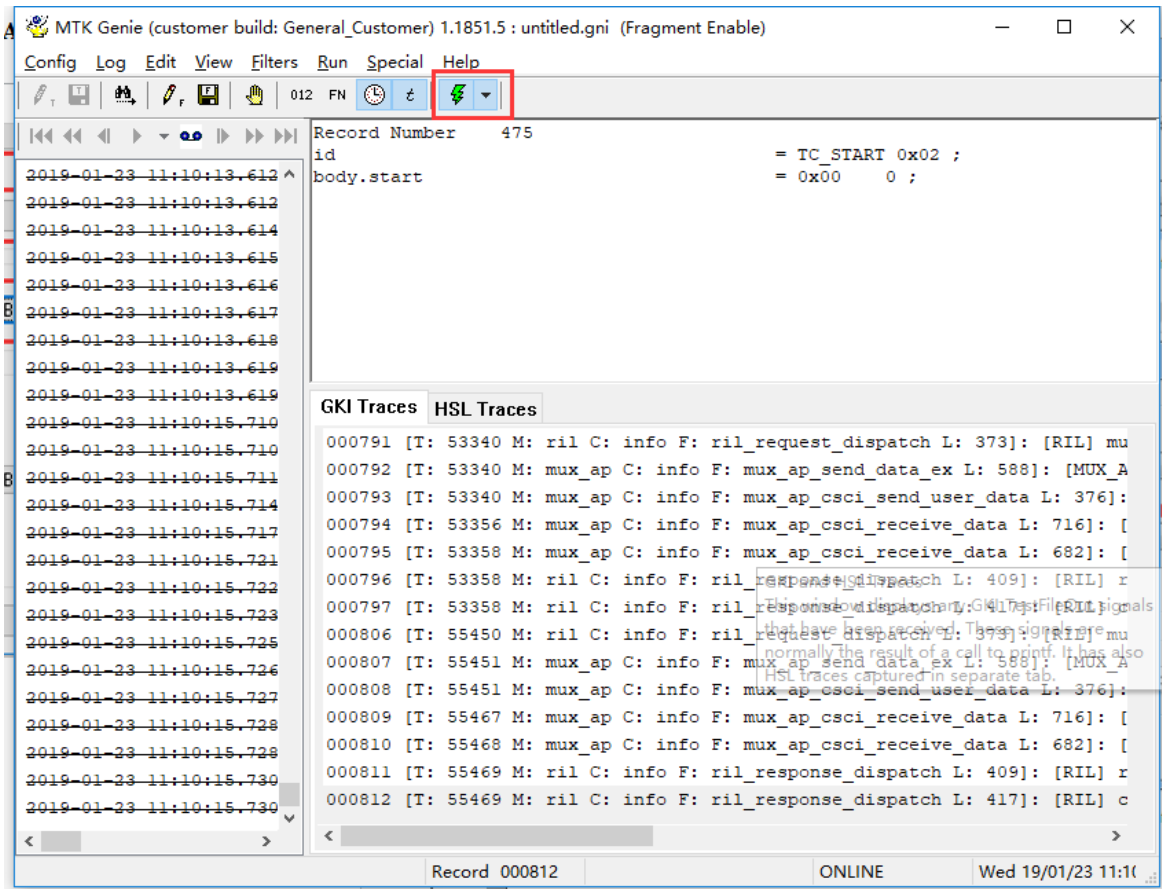
- ①打开 genie 工具软件，程序位于目录\*\nbiot\tools\core\genie 下；
- ②配置 Config 文件，第一次使用 LOG 工具，没有配置过 Config 文件，选择“New Config”；



③根据下图参数配置，选择对应配置项。GKI 端口选择 Debug 口，HSL 端口选择 Modem 口，波特率设置为 115200。Database 选择 OpenCPU SDK 根目录下 utils 目录中的 .dec 文件



④成功配置 Config 后，点击  开始抓取 log，抓 LOG 界面如下图：



⑤ 保存 log

