A76xx Series Open SDK_编译下载及调试方法

LTE 模组

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
技术支持邮箱：support@simcom.com
官网：www.simcom.com

名称：	A76xx Series Open SDK_编译下载调试方法及应用指导
版本：	V1.00
类别：	应用文档
状态：	已发布

版权声明

芯讯通无线科技(上海)有限公司
上海市长宁区临虹路289号3号楼芯讯通总部大楼
电话：86-21-31575100
邮箱：simcom@simcom.com
官网：www.simcom.com

了解更多资料，请点击以下链接：
http://cn.simcom.com/download/list-230-cn.html

技术支持，请点击以下链接：
http://cn.simcom.com/ask/index-cn.html 或发送邮件至 support@simcom.com

Version History

Version	Date	Owner	What is new
V1.00	2022-11-21		第一版

About this Document

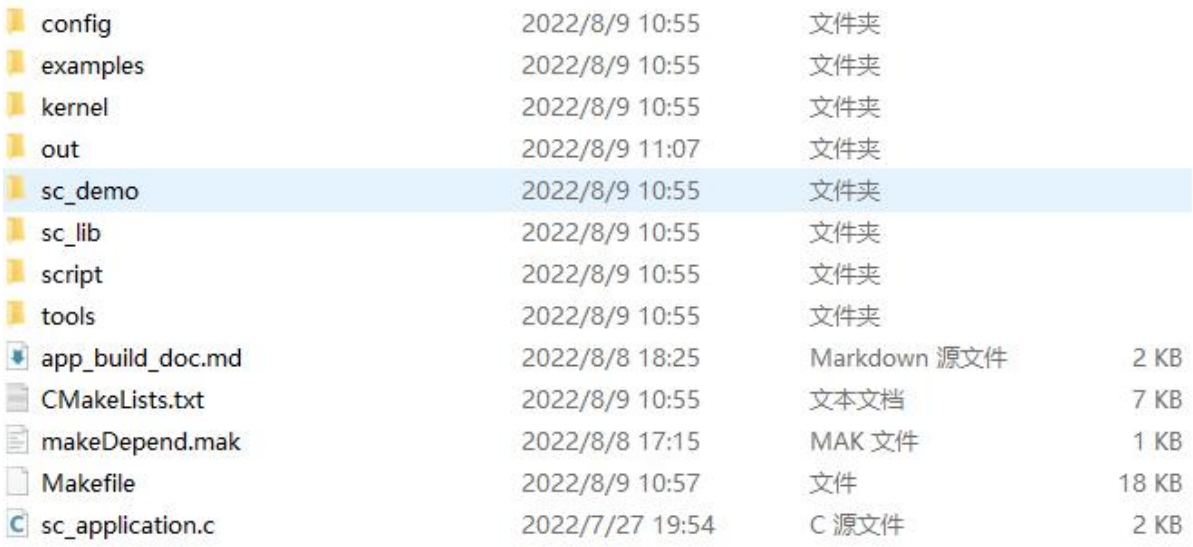
本文档适用于 A1803S open 系列、A1603 open 系列、A1606 open 系列。

目录

1 编译

1.1 16XX 编译

1.1.1 目录结构



config:每个模块关于 APP 编译的一些编译选项，app 链接脚本，cmake 编译 tool 配置；
kernel:每个模块 kernel 编译产生的文件(打包时需要依赖)；
out:编译生成的目标文件所在位置；
sc_demo:api 的使用案例；
sc_lib:api 的静态库和头文件；
script:暂时只用来放置打包脚本；
tools:编译和打包工具；
app_build_doc.md:app 编译简单介绍；
cmakelists.txt:cmake 入口；
makeDepend.mak:Makefile 的依赖关系文件，详细信息查看 app_build_doc.md 说明；
makefile:make 入口；

1.1.2 编译工具

工具	作用

gcc-arm-none-eabi	交叉编译工具链，用于编译 app
crc_set	app 校验工具
7z	压缩工具
aboot	ASR 烧录镜像打包工具
cmake	window 的 cmake 组装工具
Ninja（可选）	windows 的 Make 编译系统
GNUmake（默认）	Windows 下的 make 工具

1.1.3 编译环境配置

配置项	方法
kernel 编译环境配置	直接释放 bin 文件，不需要编译
app 编译环境配置（win32）	编译工具在 tools/win32 目录下,cmd 执行“make+回车”即可看到目标
app 编译环境配置（linux）	linux 只需安装 cmake,且版本号大于 3.10，使用终端编译即可

1.1.4 编译指令

执行动作	指令
编译生成目标文件	根目录下执行 make A7670C_LANS
清除某个模块的编译	根目录下执行 make clean_ A7670C_LANS
清除所有模块的编译	根目录下执行 make clean

用户可以直接在根目录下输入 make 或者 gnumake 可以查看支持型号，编译参数。输入 make all 或 gnumake all 一次性列出所有模块及其对应的选项。

1.1.5 编译生成

用户在 SDK 的根目录启动 cmd 终端，根据模组的型号，输入对应的型号进行编译。本次使用 A7670C_LANS 作示例。

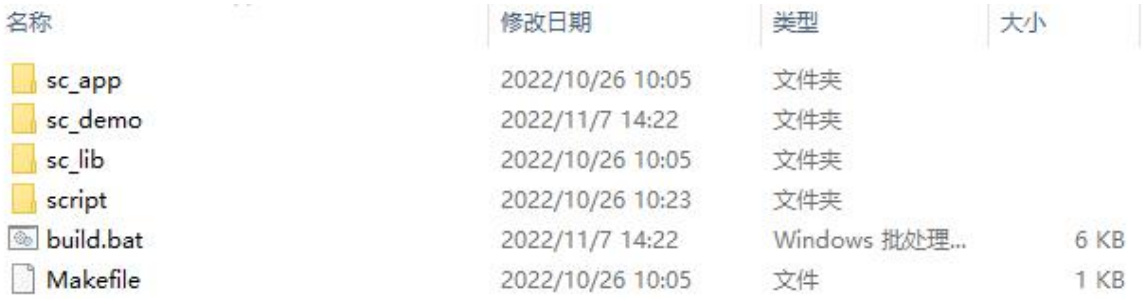
执行完成后，会在 S:\out 目录下生成 A7630C_LANS

NOTE

路径请不要包含中文和空格。

1.2 18xx 编译

1.2.1 目录结构



1.2.2 编译工具

工具	作用
gcc-arm-none-eabi	交叉编译工具链，用于编译 app
gnumake	makefile 管理工具
crc_set	目标文件校验工具
SWD	ASR 烧录镜像打包工具
Python3.8.5	编译环境

1.2.3 编译环境配置

配置项	方法
kernel 编译环境配置	直接释放 bin 文件，不需要编译
app 编译环境配置（win32）	直接释放 bin 文件，不需要编译
app 编译环境配置（linux）	linux 只需安装 cmake,且版本号大于 3.10，使用终端编译即可

1.2.4 编译指令

执行动作	指令
编译生成目标文件	根目录下执行 make A7600C_LABE_WIFI
清除某个模块的编译	根目录下执行 make clean_A7600C_LABE_WIFI

清除所有模块的编译	根目录下执行 make clean

用户可以直接在根目录下输入 make 或 gnumake 可以查看支持型号，编译参数。输入 make all 或 gnumake all 一次性列出所有模块及其对应的选项。

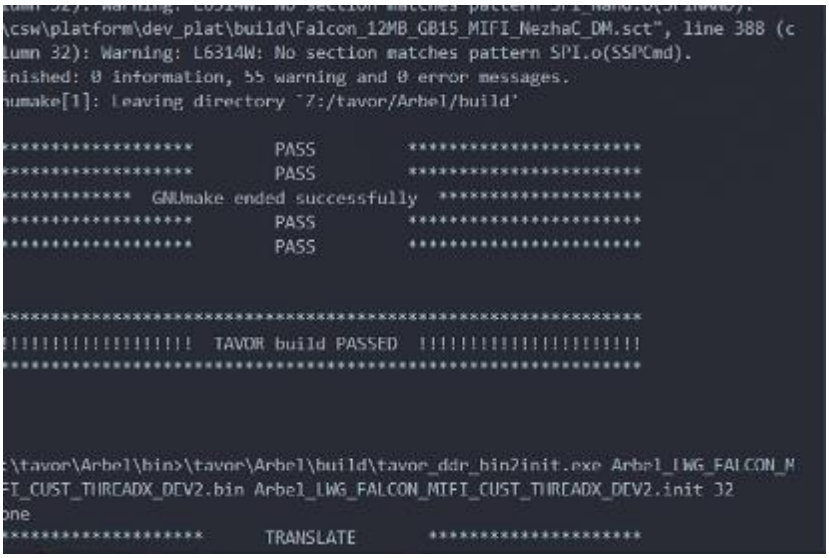
```
X:\>make
-------------------------------------------
-
-  build method: gnumake [target] (GIT_ID=[work id])
-  target:[module](_[custom])(_app/_kernel/_force/_remake),[clean option],[install option]
-
-  GIT_ID is used to dowload userspace code and tools automatically.
-  show customs of the modules: gnumake [module]-list
-  show all modules and customs at onece: gnumake all/info-all
-
-  module list:
-      A5360G
-      A7100CE_GW
-      A7100CE_GW_B
-      A7600C_M2
-      A7600C_LABD
-      A7600C_LABE
-      A7600C_MABE
-      A7600C_TABE
-      A7602E_H_LABD
-      A7602E_H_LBBE
-      A7602E_H_TABE
-      A7602E_H_TBBE
-      A7605C_LCBE_R2
-      A7605C_TABE
-      A7605C_TCBE_R2
-      A7605E_H_TABE
-      A7605E_H_TCBE_R3
-      A7605SA_H_TABE
-      A7608E_H_LABE
-      A7608E_H_LBBE
-      A7608E_H_TABE
-      A7608E_H_TBBE
-      A7608SA_H_LABD
-      A7608SA_H_TABE
-      E9730C_LCBE
-      E9730C_MCBE
-      E9730C_TCBE
-      SIM7100CE_GW
-      SIM7100CE_GW_B
-      SIM7100CE_GW_B_SJ
-
-  clean option list:
-      clean                [clean all modules.]
-      clean_app            [clean all app, target and object files.]
-      clean_kernel         [clean all kernel, just target files.]
-      clean_[module]       [clean app and kernel, just kernel target files.]
-      clean_[module]_app   [target and object files.]
-      clean_[module]_kernel [just target files.]
-
-  install option list:
-      install_[module]     [release one module to a SDK package. it will compile automatically when it not been compiled].
-      install_set          [release all modules in one SDK package, which have been compiled.]
-      uninstall_[module]   [delete the SDK package for one module.]
-      uninstall_set        [delete the SDK package, which is mix some different modules.]
```

1.2.5 编译生成

用户在 SDK 的根目录启动 cmd 终端，根据模组的型号，输入对应的型号进行编译。本次使用 A7600C_LABE_WIFI 作示例。

```
X:\>make A7600C_LABE_WIFI
```

执行完成后，



会生成一个名为 A7630C_LABE_WIFI_SWD 的文件夹。

NOTE

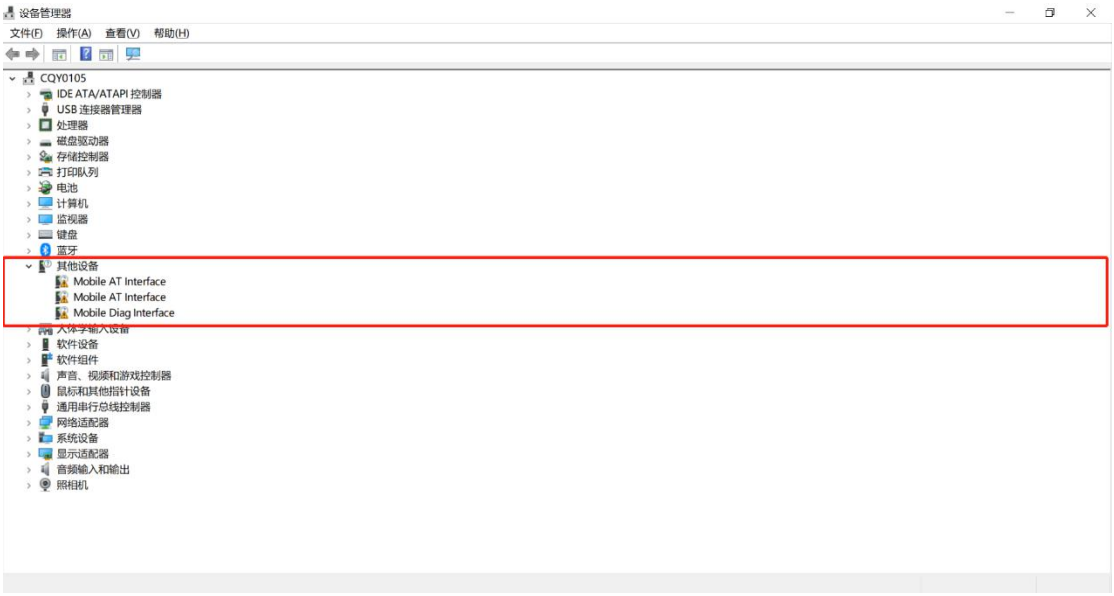
路径请不要包含中文和空格。

2 烧录

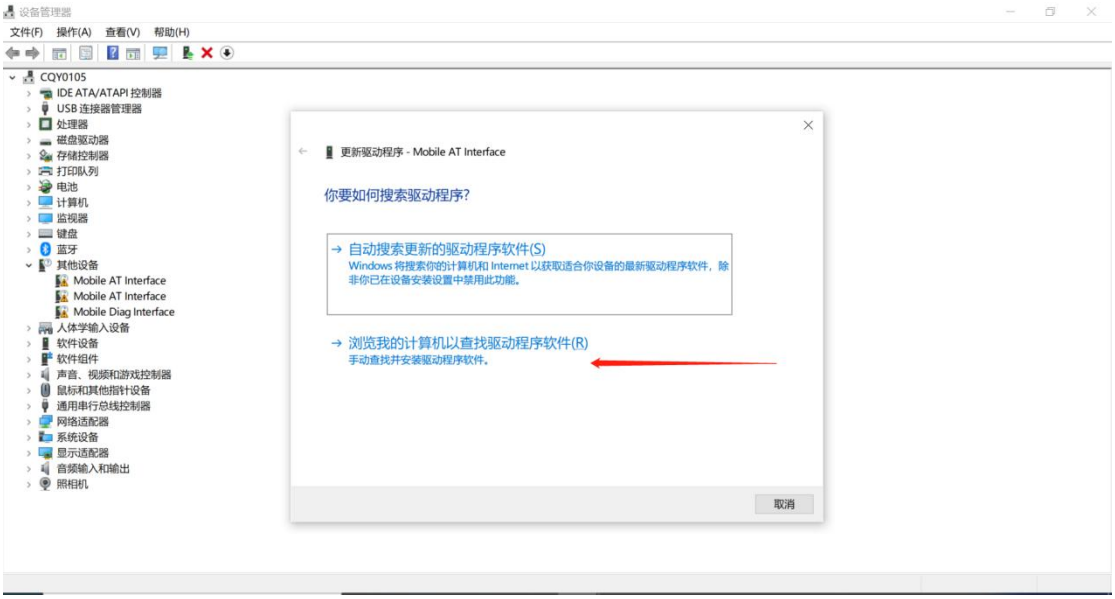
2.1 安装相关驱动

SIMCom 提供了 USB 的相关驱动程序如下：

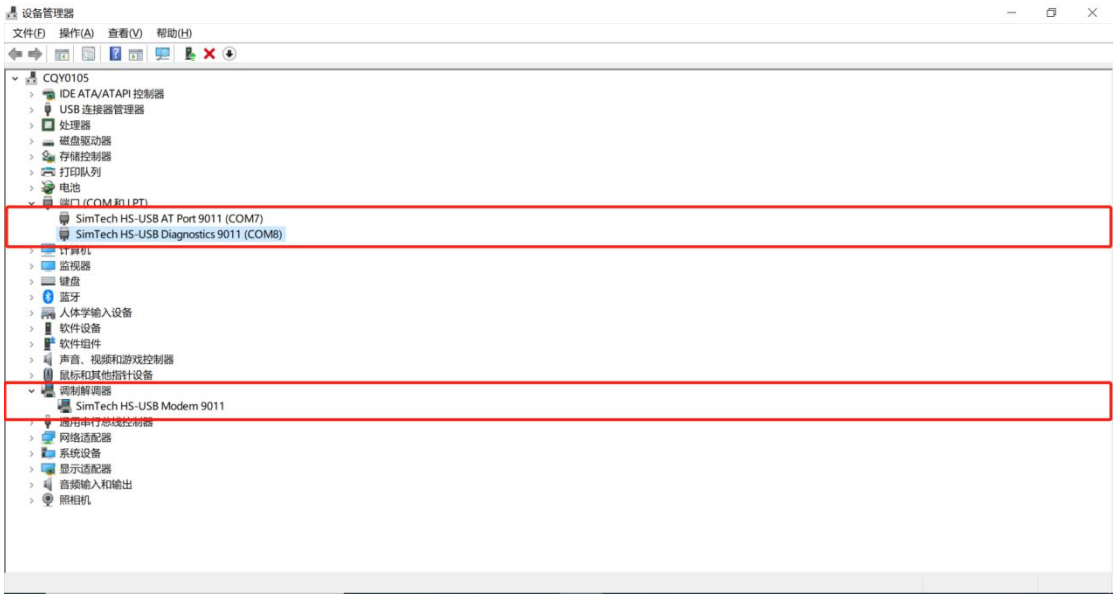
在 PC 未安装驱动程序的情况下，插入 USB 后会提示以下设备安装驱动程序



第一步： 用我们支持的驱动程序扫描电脑更新驱动程序。

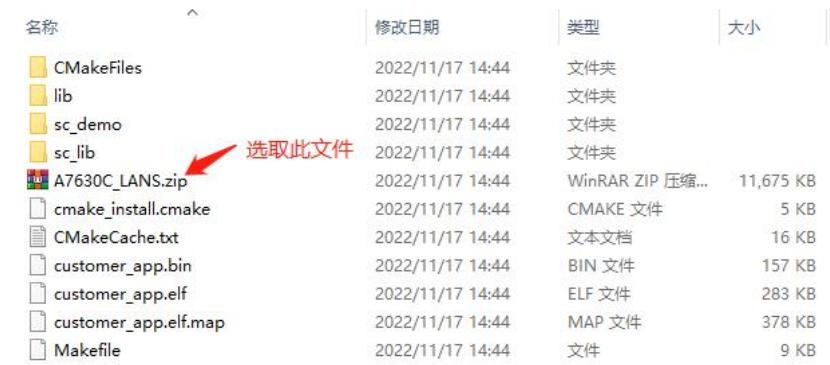


第二步： 按如下步骤依次安装驱动程序，直到全部安装成功。

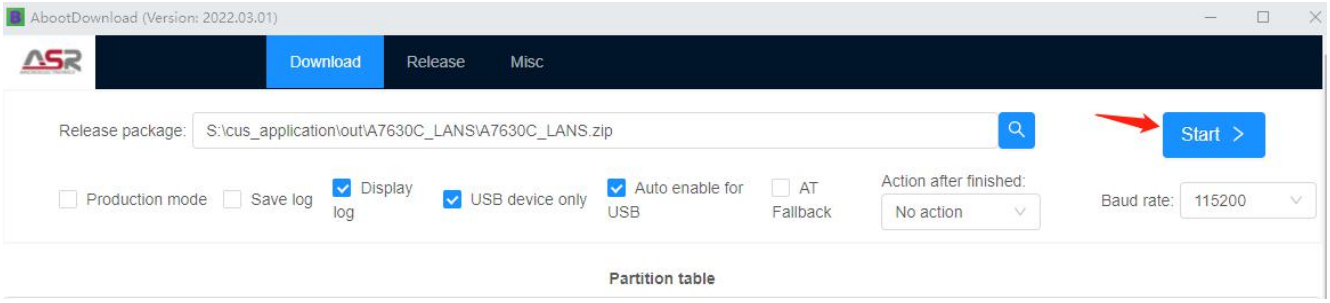


2.2 16XX 烧录

第一步：打开 **aboot.exe**
第二步：选取烧录文件



第三步：点击 Start 开始烧录

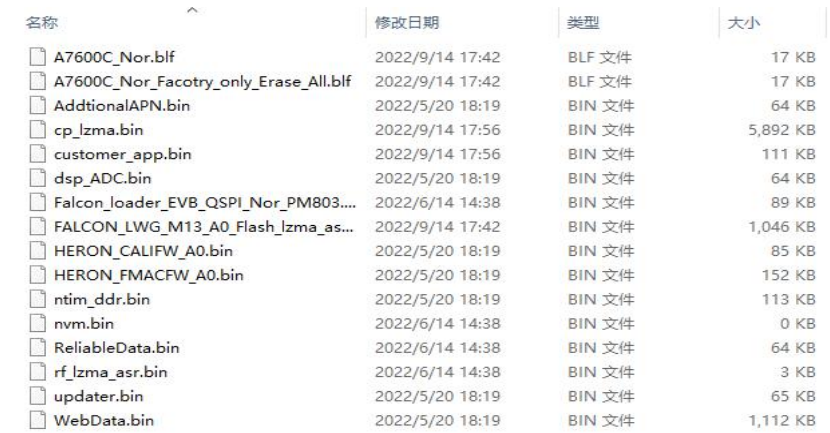


2.3 18XX 烧录

在烧录之前，请确保已经编译了固件包。目前，提供了独立的烧录 APP 和完整的烧录固件包。请确保每一步都顺利完成，否则可能会出现错误。详细步骤如下所示。

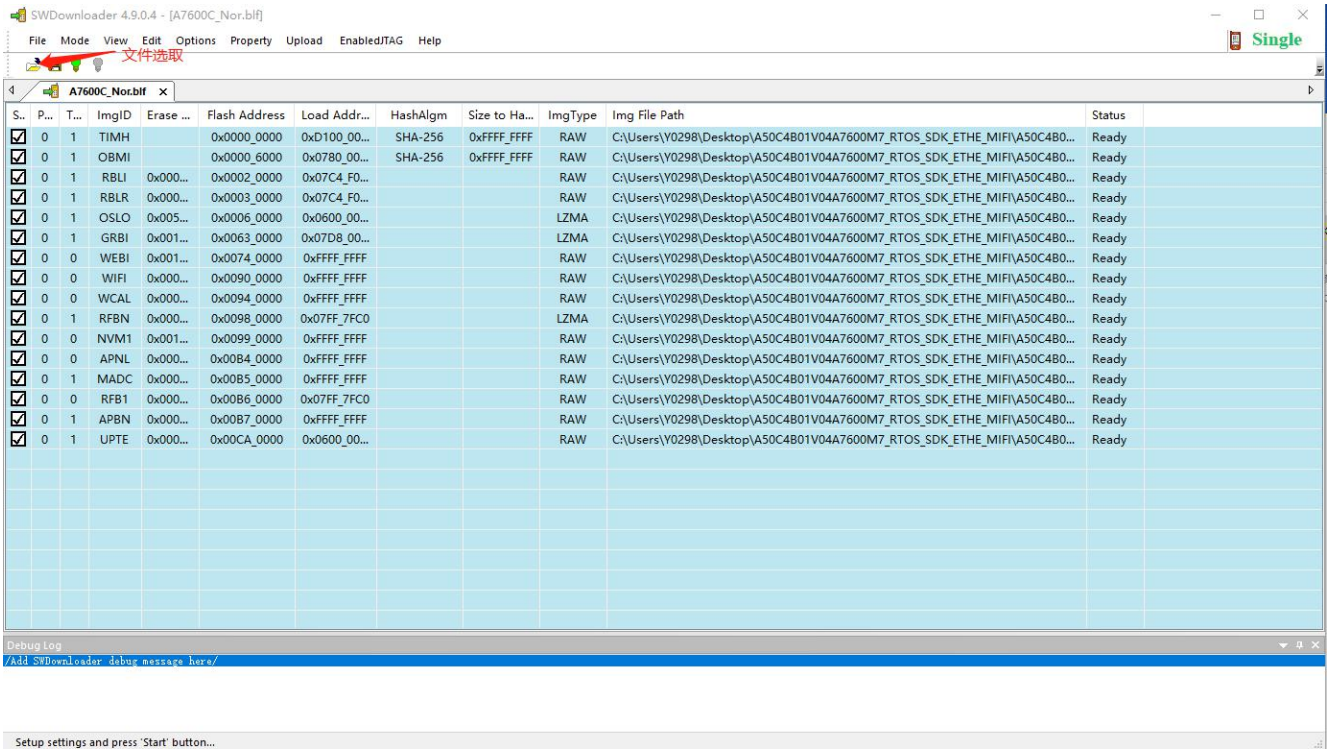
第一步：检查固件包。

在编译生成的 A7630C_LABE_WIFI_SWD 文件中包含有所有烧录需要用到固件包。



第二步：打开 SWDownloader.exe。

第三部：选择固件包。

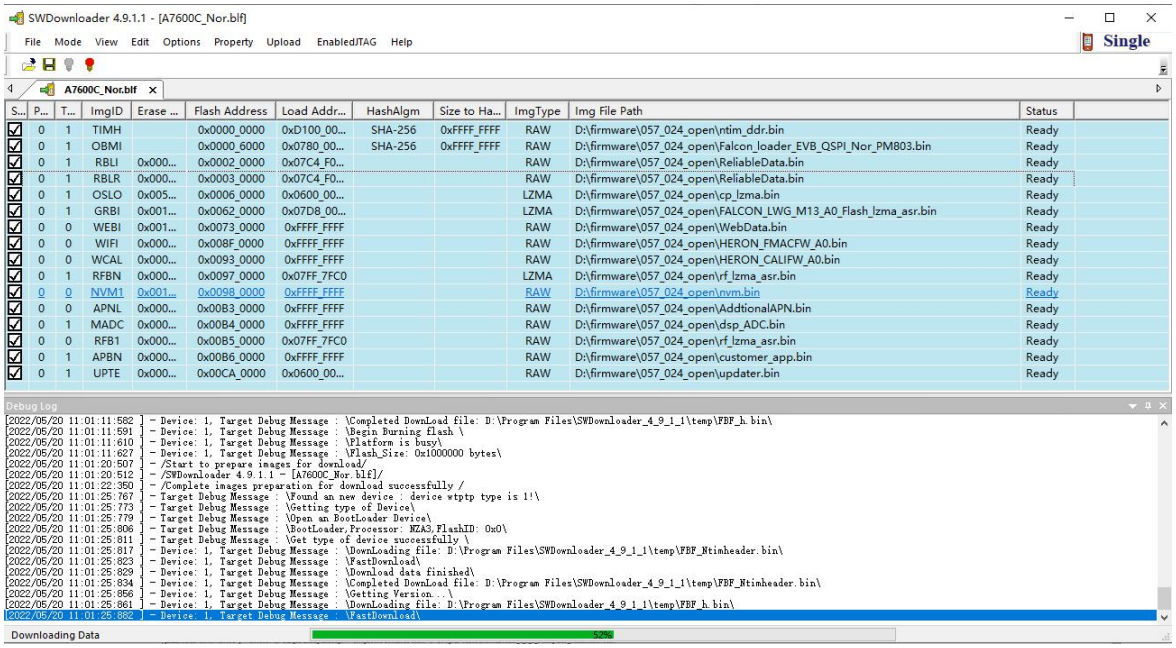


NOTE

A76xxx_Nor.blf ——只删除所选分区并升级
A76xxx_Nor_Facotry_only_Erase_All.blf ——删除所有 flash 并升级

A76xxx_Nor_Facotry_only_Erase_All.blf 将擦除所有 flash，包括 RD 数据和工厂校准数据，如 iMEI。请勿在生产环境中执行此操作！软件升级时推荐使用 A76xxx_nor.blf。

第四步：点击“开始”按钮开始下载，然后按下 POWER_KEY 下载。

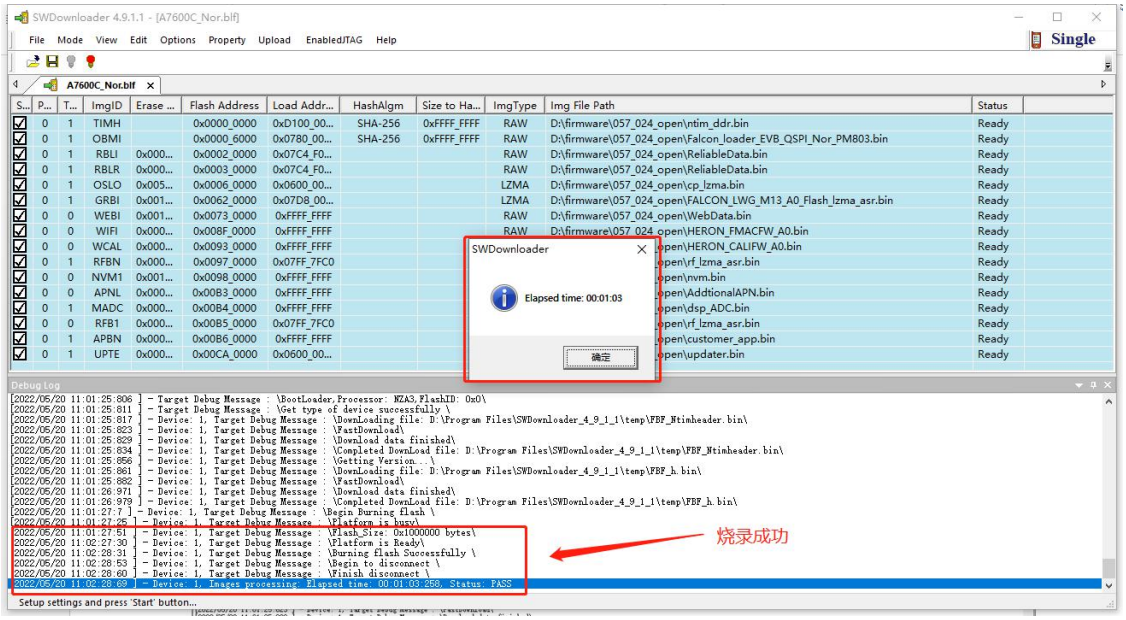


NOTE

如果没有出现进度条，请尝试重复第四步。

第五步:如果下载成功，将显示以下内容。

请记得点击“返回”按钮，否则模块重新上电后可能会再次触发下载模式。



3 应用指导

3.1 添加 Task

第一步：在子目录 demo_helloworld.c 文件 sAPP_HelloWorldDemo()函数中添加 task。添加 task 需要使用 sAPI_TaskCreate()，另外，入口函数注册的结构体可以设置入口函数的优先级、堆栈等信息，有关 task 更详细操作的使用方法请参考文档《A76xx Series Open SDK_RTOS 开发_应用指导》。

参考 demo_helloworld.c 建立 task。

```
#include "simcom_os.h"
#include "simcom_debug.h"
#include "stdio.h"


sTaskRef helloWorldProcesser;
static UINT8 helloWorldProcesserStack[1024];
#if 0
/**
 * @brief simcom printf
 * @param pointer *format
 * @note  The message will be output by sAPI_Debug and captured by CATSTUDIO tool.
 * @retval void
 */
void simcom_printf(const char *format,...){
    char tmpstr[200];

    va_list args;

    memset(tmpstr,0,sizeof(tmpstr));

    va_start(args,format);
    sAPI_Vsnprintf(tmpstr,sizeof(tmpstr),format,args);
    va_end(args);

    sAPI_Debug("simcom_printf [%s]",tmpstr);

}
#endif
```

```c
void sTask_HelloWorldProcesser(void* argv)
{
    sAPI_Debug("Task runs successfully");

    //simcom_printf("%s %d %f","simcom",2020,10.23);

}

/**
 * @brief helloworld task initial
 * @param void
 * @note
 * @retval void
 */
void sAPP_HelloWorldDemo(void)
{
    SC_STATUS status = SC_SUCCESS;

    status = sAPI_TaskCreate(&helloWorldProcesser, helloWorldProcesserStack, 1024, 150, "helloWorldProcesser",sTask_HelloWorldProcesser,(void *)0);
    if(SC_SUCCESS != status)
    {
        sAPI_Debug("Task create fail,status = [%d]",status);
    }
}
```

利用sAPI_TaskCreate创建task

task优先级

task主函数

第二步：将 **task** 创建函数添加到根目录 **sc_application.c** 文件 **userSpace_Main()**函数中。

参考 sc_application.c

```c
void userSpace_Main(void *arg)
{
    /* simcom api init. Do not modify! */
    ApiMapInit(arg);
    //sAPI_enableDUMP();
    /* UI demo task for customer with CLI method for all demo running,
       customer need to define SIMCOM_UI_DEMO_TO_USB_AT_PORT or
       SIMCOM_UI_DEMO_TO_UART1_PORT to select hardware interface.
    */
    sleep(5);//Wait for USB initialization to complete and print catstudio log.

    sAPP_SimcomUIDemo();

    sAPP_UartTask();
    sAPP_UrcTask();
    sAPP_UsbVcomTask();
    sAPP_HelloWorldDemo();
    //sAPI_FotaCallback(fotacb);
    printf("user app is running...");

}
```

第三步：根目录下的 **CMakeLists.txt** 里面添加目录名并且添加链接库。

参考根目录下 CMakeLists.txt。

```
38  # 编译子目录
39  if(NOT DEFINED SIMCOM_SDK)
40      if(DEFINED QL)
41          add_subdirectory(./ql_lib ql_lib)
42      else()
43          add_subdirectory(./sc_lib sc_lib)
44      endif()
45  endif()
46
47  if(DEFINED QL)
48      add_subdirectory(./ql_demo ql_demo)
49  else()
50      add_subdirectory(./sc_demo sc_demo)
51  endif()
```

```
117 if(DEFINED QL)
118 if(NOT DEFINED SIMCOM_SDK)
119 target_link_libraries(userspace PRIVATE ql_lib)
120 endif()
121 target_link_libraries(userspace PRIVATE ql_demo)
122 else()
123 if(NOT DEFINED SIMCOM_SDK)
124 target_link_libraries(userspace PRIVATE sc_lib)
125 endif()
126 target_link_libraries(userspace PRIVATE sc_demo)
127 endif()
```

第四步：子目录下新建相应的 **CMakeLists.txt**，并且添加相应编译的源文件和依赖的头文件路径。

参考子目录下 CMakeLists.txt。

```
1  cmake_minimum_required(VERSION 3.10)
2
3  AUX_SOURCE_DIRECTORY(./src sc_demo_src)
4  AUX_SOURCE_DIRECTORY(./src/token sc_demo_src)
5  AUX_SOURCE_DIRECTORY(./src/utils sc_demo_src)
6
7
8  # Add the static library
9  add_library(sc_demo STATIC ${sc_demo_src})
10 target_include_directories(sc_demo PUBLIC
11     ./inc
12     ${CMAKE_SOURCE_DIR}/sc_lib/inc
13     ${CMAKE_SOURCE_DIR}/sc_lib/${SCMODULE}/inc
14     ${CMAKE_SOURCE_DIR}/sc_lib/inc/GPIO
15     ${CMAKE_SOURCE_DIR}/sc_lib/inc/token
16     ${CMAKE_SOURCE_DIR}/sc_lib/inc/utils
17     ${CMAKE_SOURCE_DIR}/sc_demo/inc
18 )
19
20 SET(LIBRARY_OUTPUT_PATH ${PROJECT_BINARY_DIR}/lib)
```

4 调试

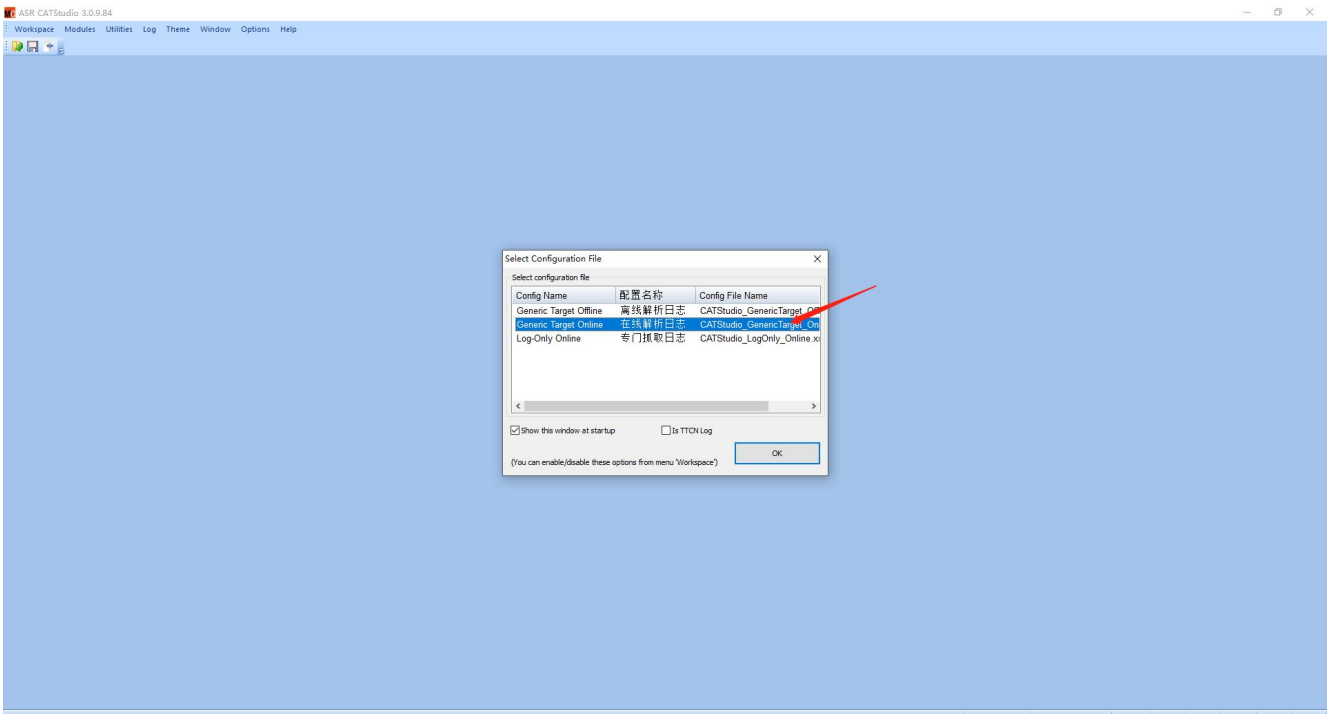
我们支持使用 CATStudio 进行调试。

NOTE

ACAT 日志分为在线日志和离线日志。

4.1 普通调试

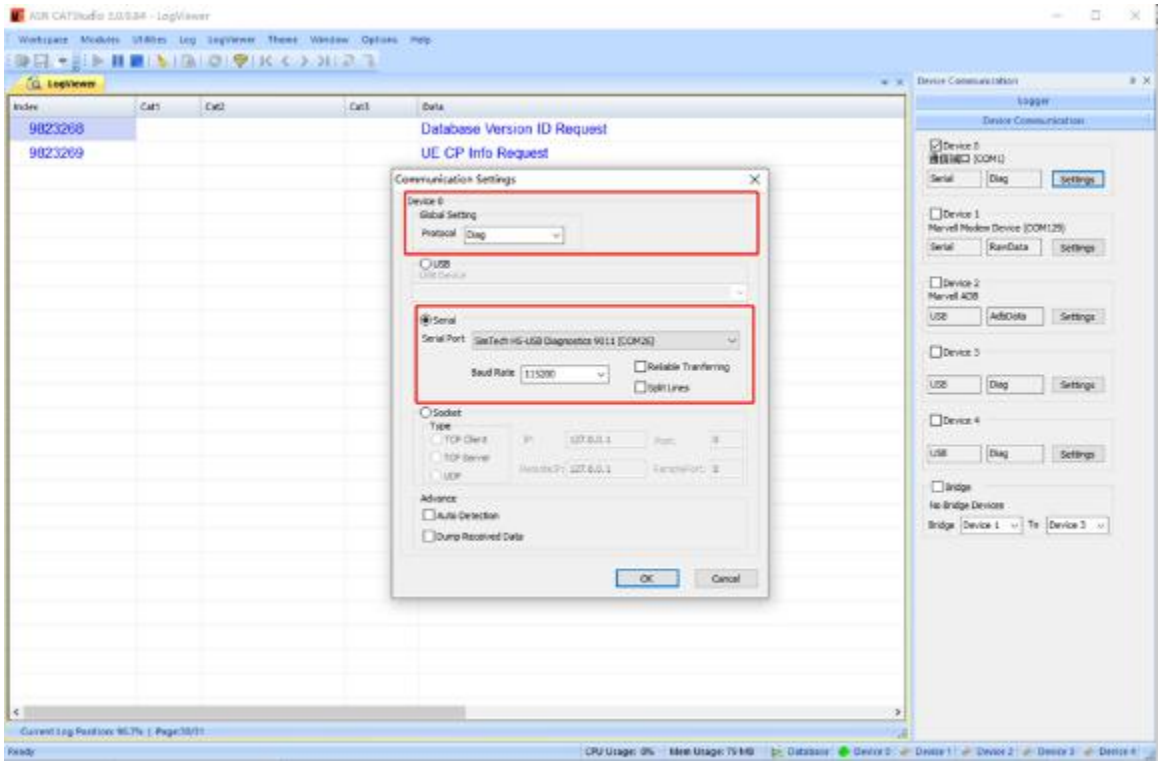
第一步：运行 CATStudio。如果 CATStudio 运行成功，则需要配置文件，选择所需的文件，单击“OK”。



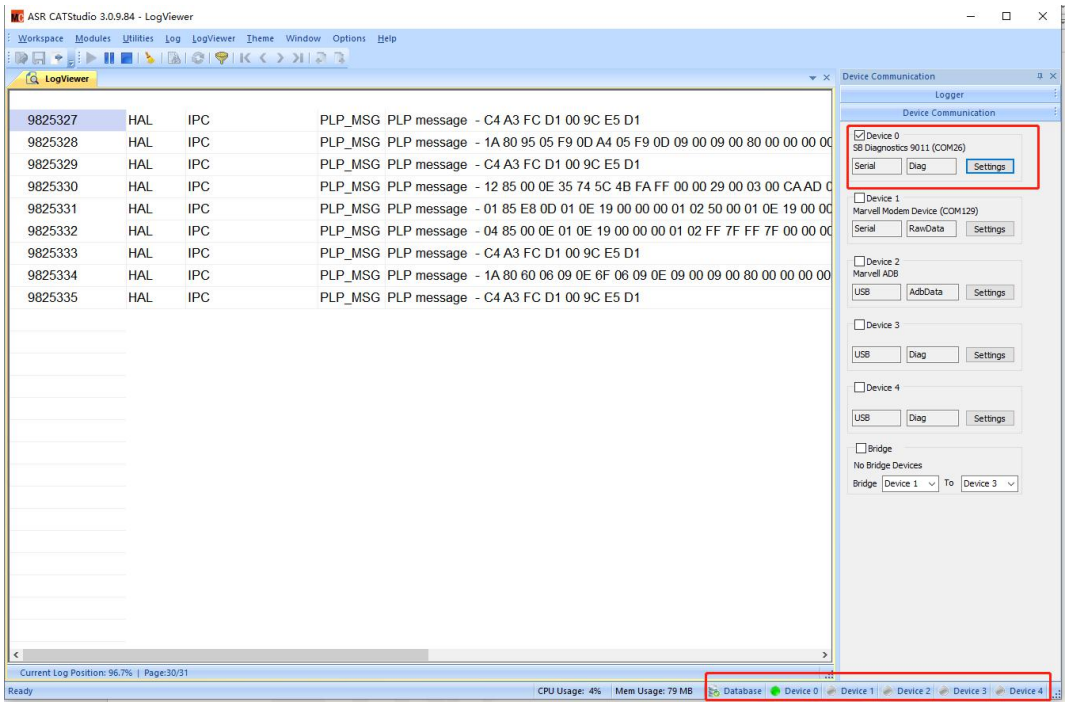
NOTE

在线解析日志，请选择 CATStudio_LWG_Online.xml。
离线解析日志，请选择 CATStudio_LWG_Offline.xm。

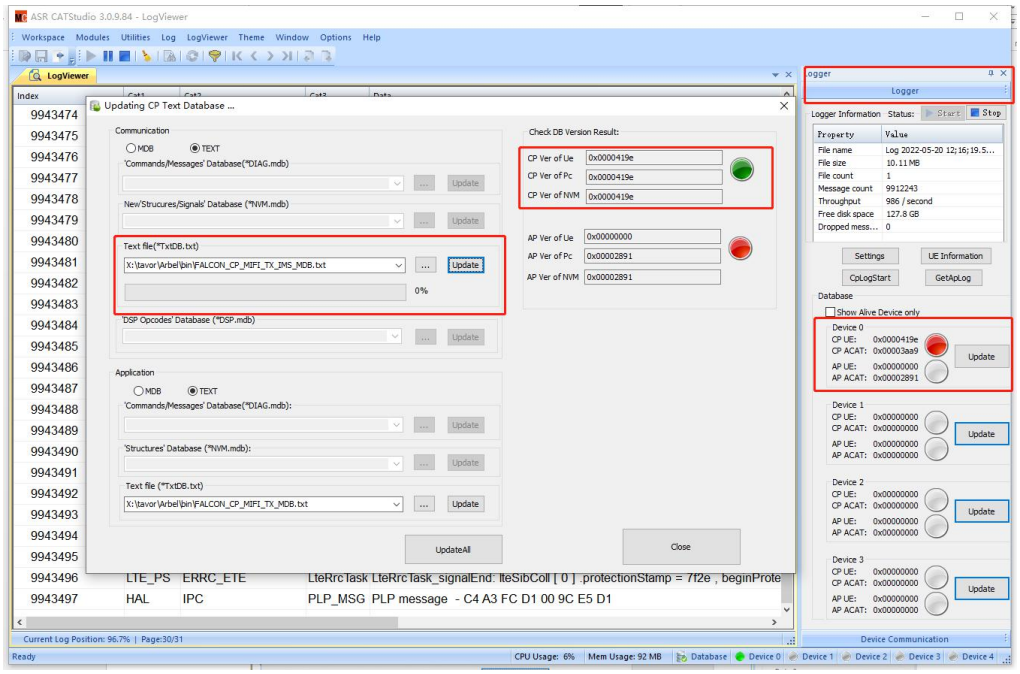
第二步：如图所示，单击“Device Communication”中某个设备号的设置，选择连接设备，如可以配置为 Device0 输出日志。



第三步：检查上一步选择的设备，连接成功时显示绿色，连接失败时显示红色。



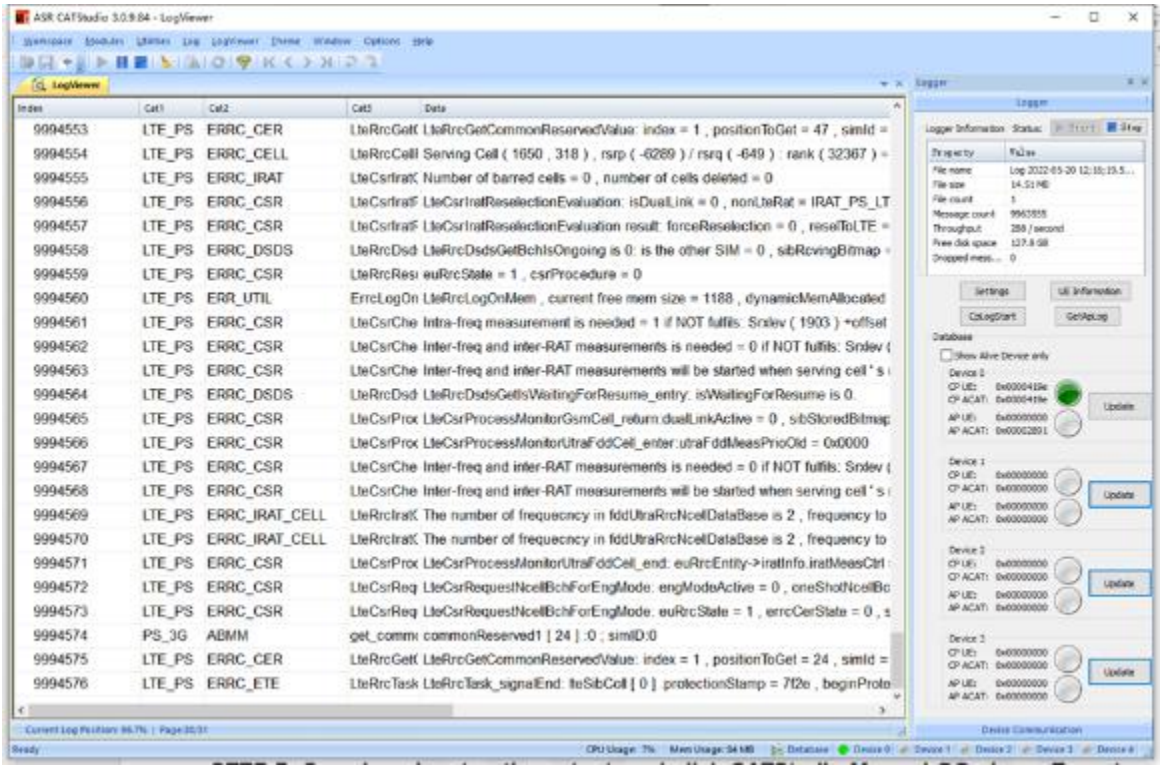
第四步：打开 Logger 面板，更新 DB(cp_MDB.txt 文件存储在\tavor\Arbel\bin 目录下)，如果.txt 与模块中下载的 cp 完全匹配，则 DB 版本结果将显示为绿色。



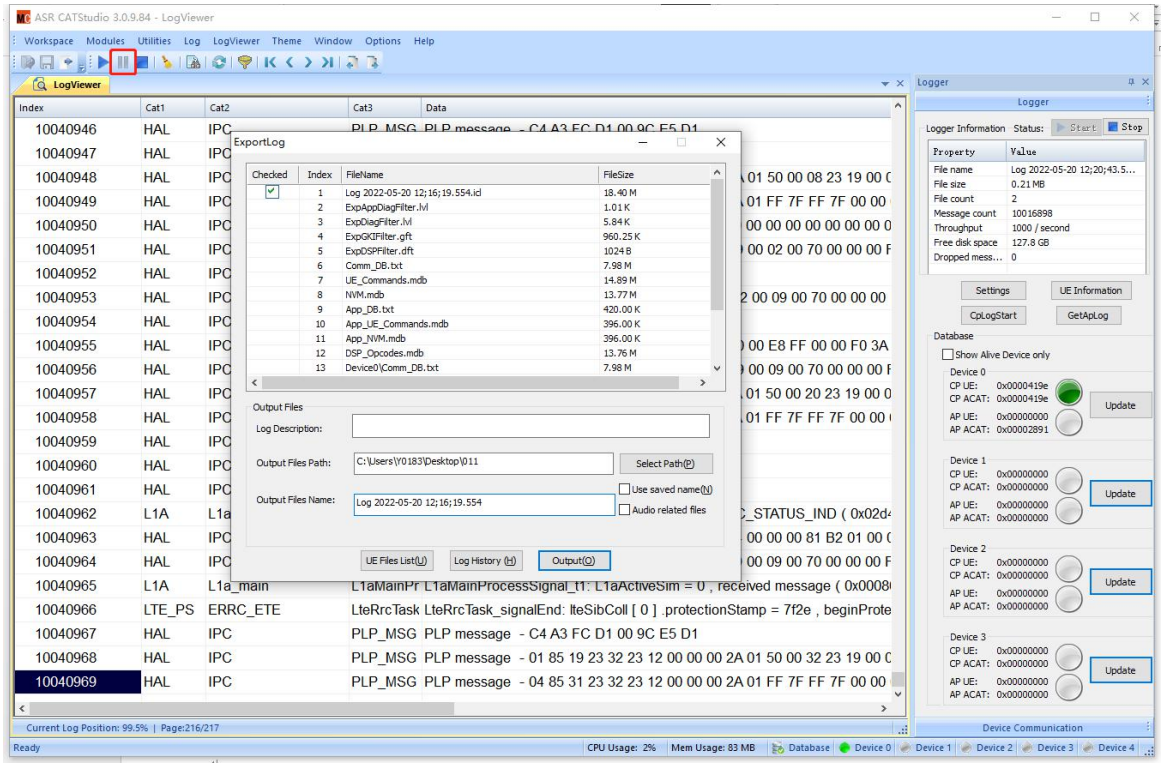
NOTE

请确保 DB 和 CP.bin 完全匹配，否则一些日志无法分析。

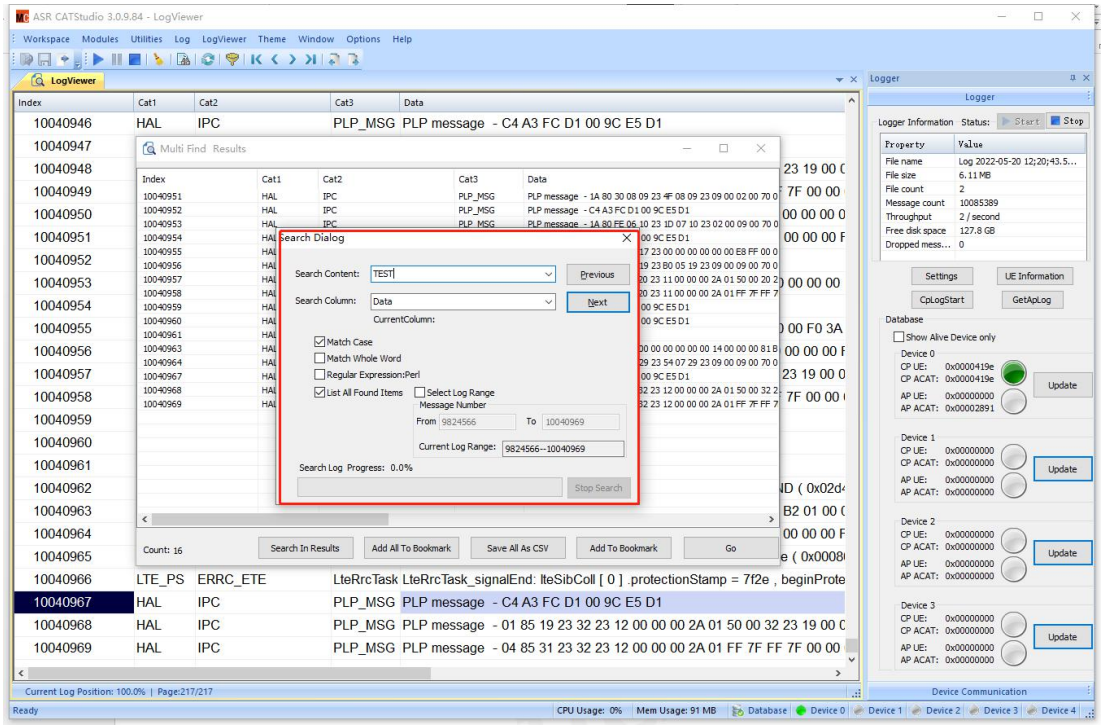
第五步：点击 **CATStudio Menu: Modules->LogViewer** 打开 **LogViewer** 面板，日志显示在这里。



第六步：通过停止输出保存日志，点击 **CATStudio** 的 **Menu:LOG->Log-> Export LOG- file**。点击“**output**”按钮将 **zip** 格式的日志保存到指定路径。



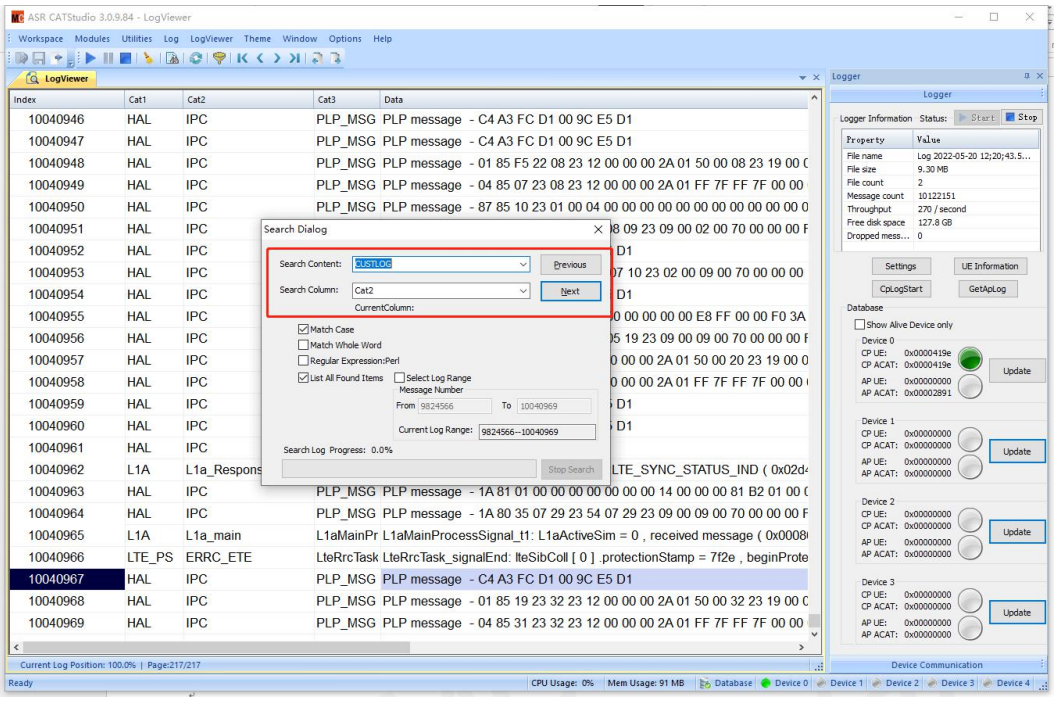
第七步：当输出停止时，可以通过 CTRL+F 搜索指定的信息。通过选择“Previous”或“Next”可以向上或向下搜索。



NOTE

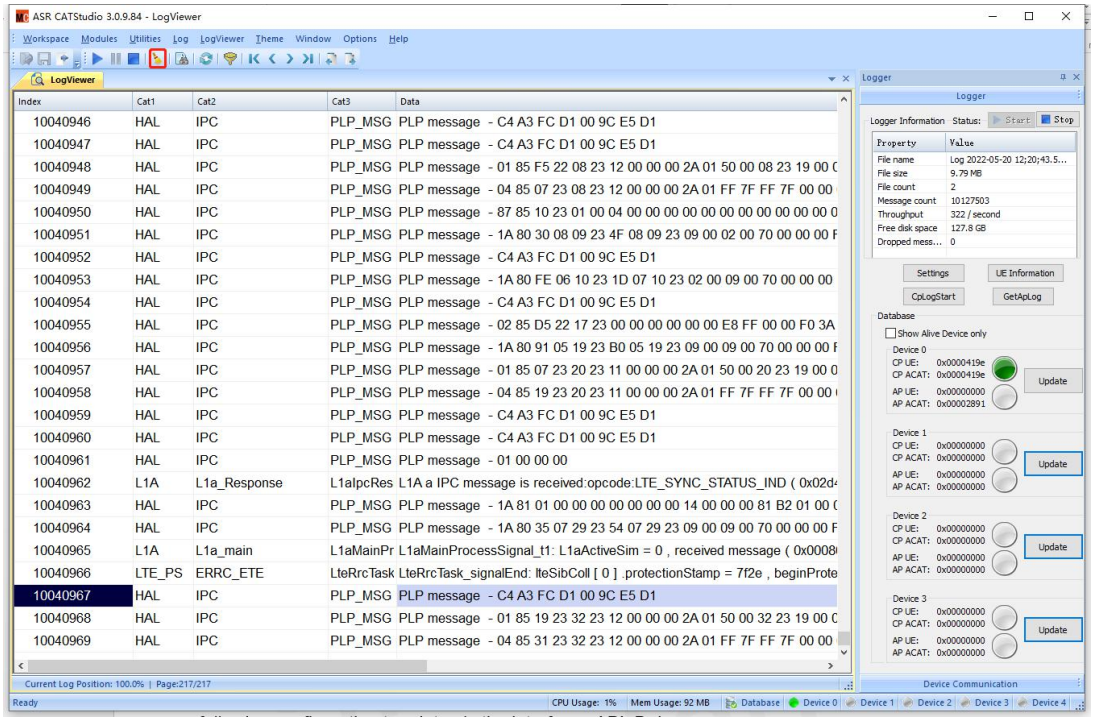
CATStudio 支持过滤搜索结果，支持在“查找结果”面板中的结果中搜索。

第八步：使用 CTRL+F 获取 api sAPI_Debug 输出日志，并设置如下所示的条件。



第九步：如果一段时间内测试结果没有问题，说明到目前为止的日志可能是无用的，请单击“清除”按钮清除日志。

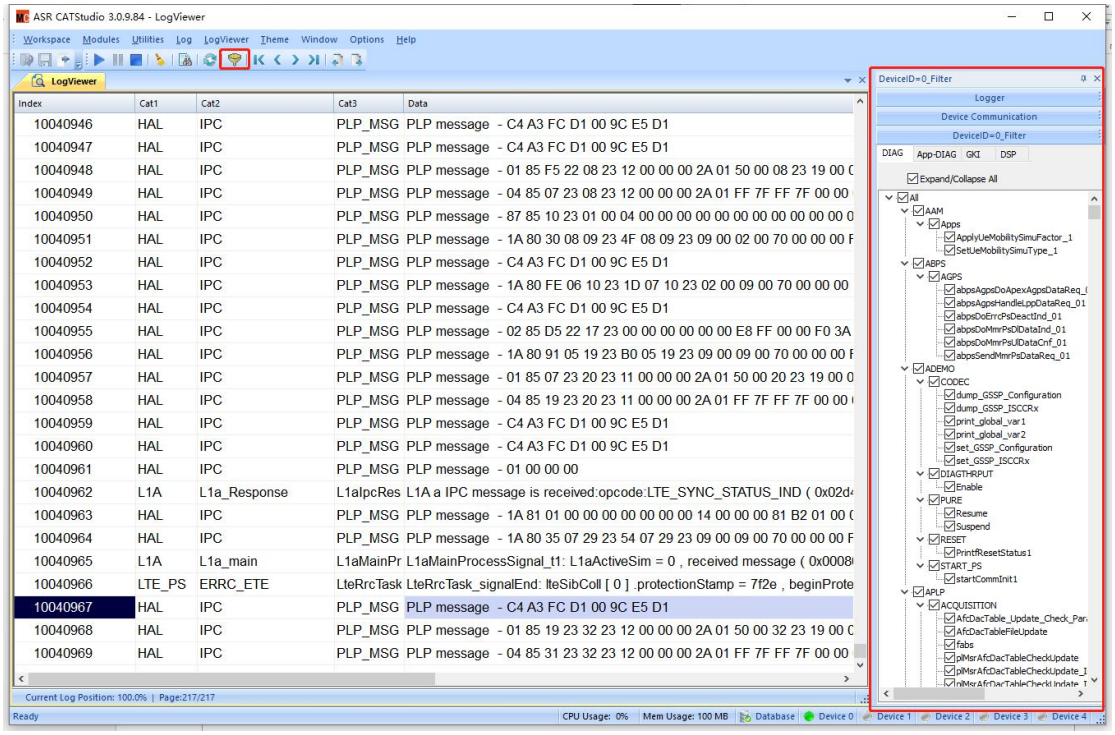
这个操作有利于分析，因为日志文件不会太大，无用的信息不会显示在日志中。



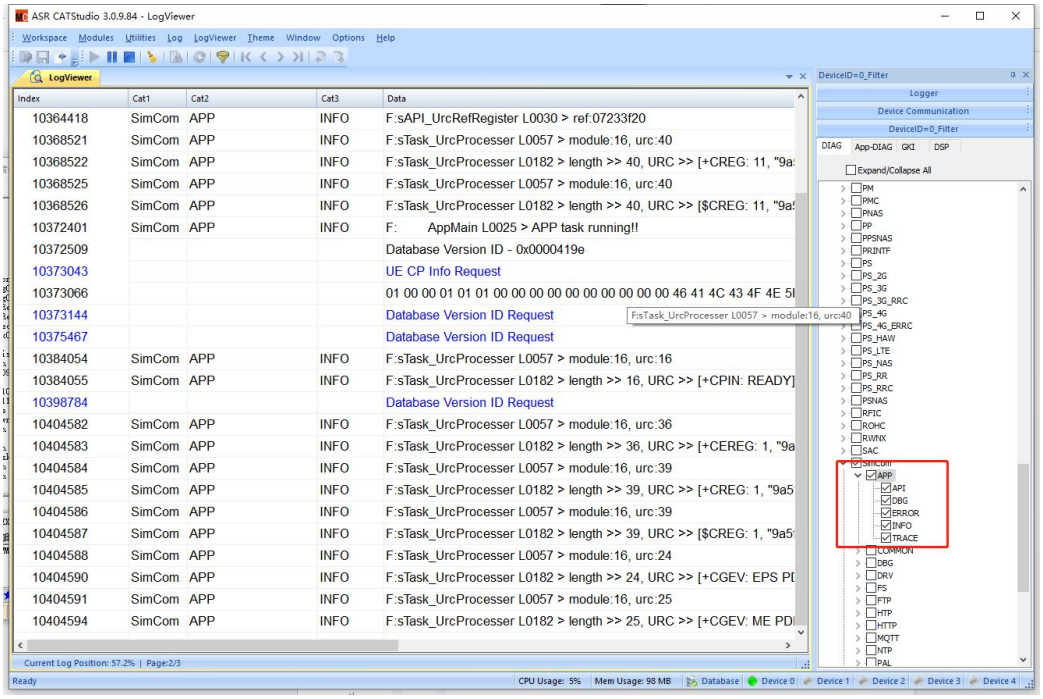
4.2 日志过滤

CATStudio 支持日志过滤 ，CATStudio 默认打印所有日志，日志过多会影响调试效率，请参考以下配置，只打印 sAPI_Debug 接口。操作步骤如下。

第一步：单击如下所示的 Show Filter 按钮。



第二步：点击 DIAG 按钮，只检查 CUST，删除 App-DIAG GKT DSP 的选项，这时，你只能看到 sAPI_Debug 接口打印



NOTE

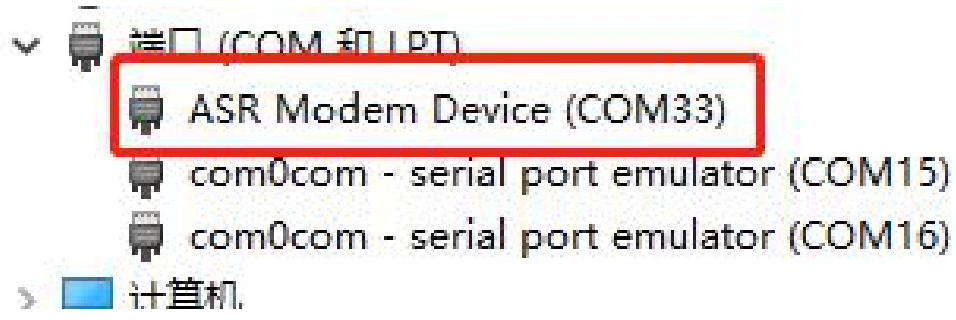
1.日志将自动保存在安装路径的 Exec\Bin Logs 目录下。
2.请清理无用的日志，以免磁盘空间耗尽。而且，路径下的.icl 可以由 CATStudio 在脱机模式下打开，并且有完美匹配的 DB 文件。

5DUMP

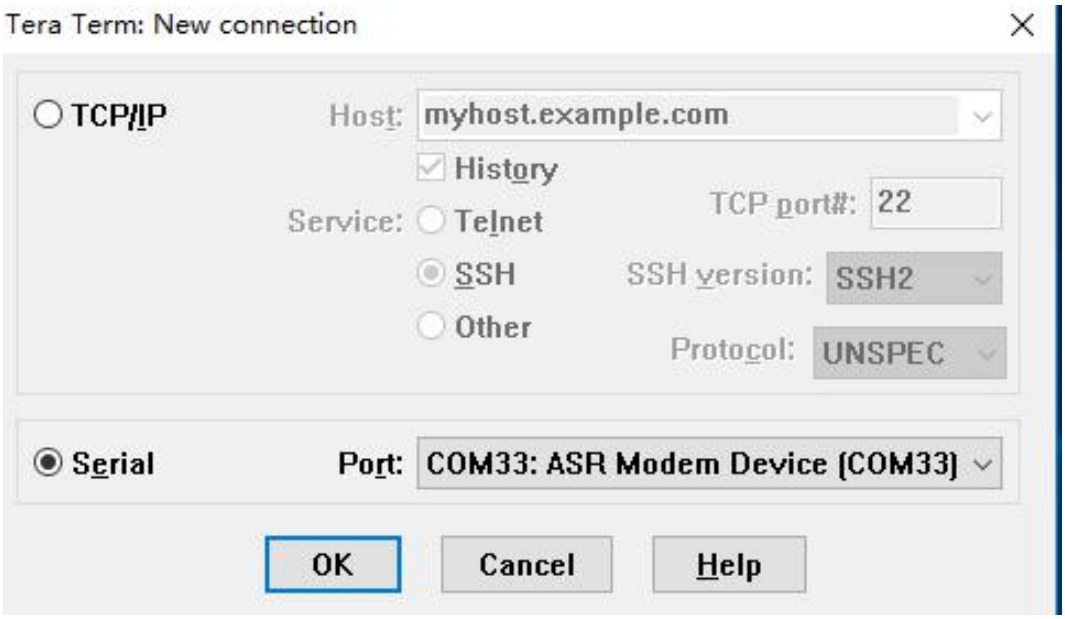
当模块运行时，由于某些原因(内存应用失败、非法指针访问、堆栈溢出等)，它会进入 dump 模式。此时需要导出 dump 信息进行分析。

第一步：打开 ttermpro.exe。

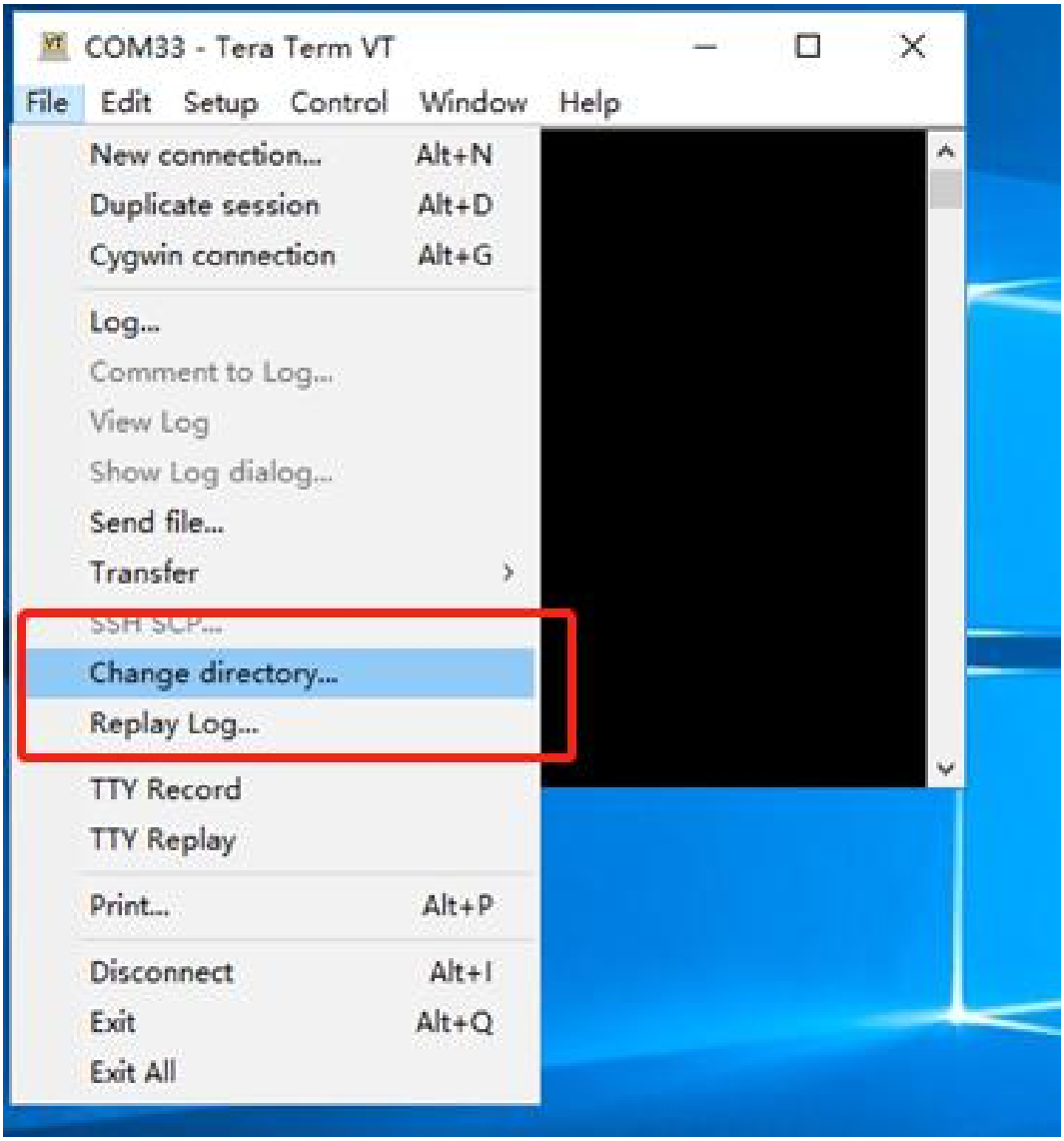
第二步：在设备管理器中确认 ASR Modem 设备端口号的 com 号。



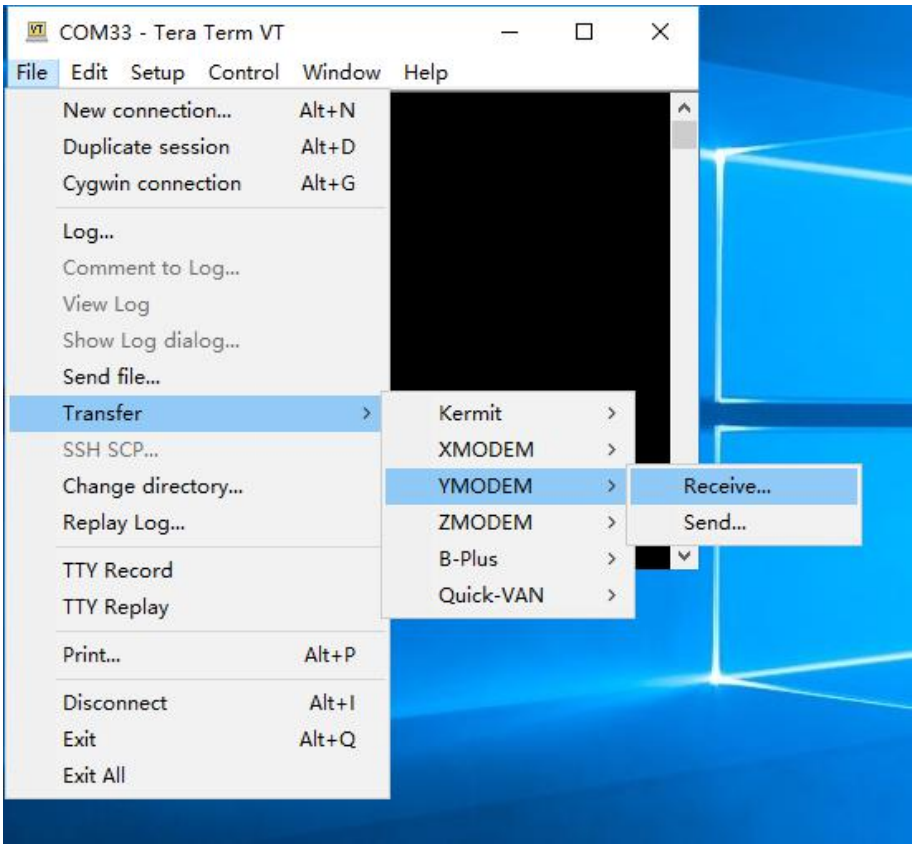
第三步：打开 ttermpro 并选择 com 端口来抓取 dump。

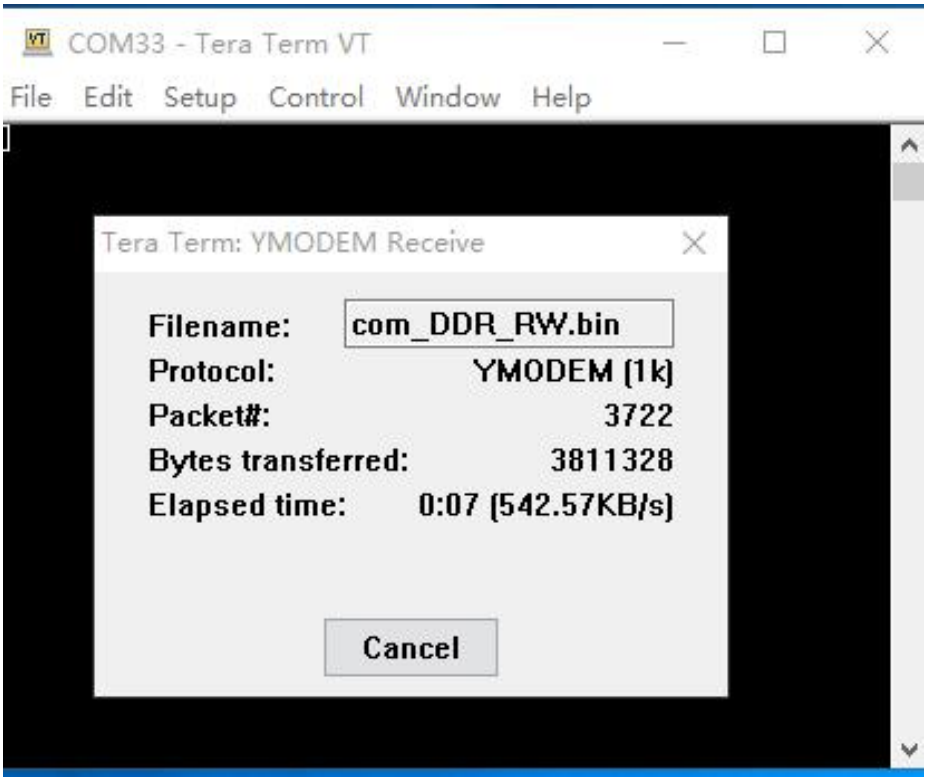


第四步：“更改目录”用于选择 dump 文件的保存位置。



第五步：选择接收 **dump** 文件的 **ymodem** 协议。





第六步：检查收到的 **dump** 文件。

com_compmde.bin	2020/7/28 16:44	BIN 文件	1 KB
com_CustVer.bin	2020/7/28 16:44	BIN 文件	1 KB
com_DDR_RW.bin	2020/7/28 16:44	BIN 文件	14,272 KB
com_DSP_DDR.bin	2020/7/28 16:44	BIN 文件	2,112 KB
com_DSPdtcm.bin	2020/7/28 16:44	BIN 文件	256 KB
com_DTCM.bin	2020/7/28 16:44	BIN 文件	64 KB
com_dtcSpy.bin	2020/7/28 16:44	BIN 文件	2 KB
com_dvfmSpy.bin	2020/7/28 16:44	BIN 文件	1 KB
com_EE_Hbuf.bin	2020/7/28 16:44	BIN 文件	1 KB
com_GpioSpy.bin	2020/7/28 16:44	BIN 文件	26 KB
com_ITCM.bin	2020/7/28 16:44	BIN 文件	64 KB
com_L1xSpy.bin	2020/7/28 16:44	BIN 文件	6 KB
com_L2_sram.bin	2020/7/28 16:44	BIN 文件	128 KB
com_PS_DAT.bin	2020/7/28 16:44	BIN 文件	448 KB
com_ri_tek.bin	2020/7/28 16:44	BIN 文件	1 KB
com_SQU.bin	2020/7/28 16:44	BIN 文件	64 KB
com_usbIntD.bin	2020/7/28 16:44	BIN 文件	1 KB
com_usbLog.bin	2020/7/28 16:44	BIN 文件	32 KB
com_usbReset.bin	2020/7/28 16:44	BIN 文件	1 KB
com_wdtKICK.bin	2020/7/28 16:44	BIN 文件	1 KB
com_wifiSpy.bin	2020/7/28 16:44	BIN 文件	2 KB