

High-Flying Software Framework (HSF-LPBx30) 用户手册 V1.1x

版本 1.1x
2018 年 01 月

更新记录:

修改时间	作者	修改	删除
2017.08.10	HF	初版	
2017.10.20	HF	补充完善	
2018.01.02	HF		

目录

1. SDK 说明	4
1.1 型号说明	4
1.2 版本变更历史	4
2. 编译环境安装	5
2.1 编译方式说明	5
2.2 搭建 Keil+GCC 编译环境	5
2.3 SDK 目录结构	7
3. 开始编译	8
3.1 工程文件说明	8
3.2 开始编译	8
3.3 用户添加源代码文件	9
3.4 开发注意事项	10
3.5 特别注意事项	11
3.6 常见问题分析	11
4. 资源分配	13
4.1 1MB Flash 资源分配	13
4.2 4MB Flash 资源分配	13
4.3 Ram 资源	15
4.4 关于 1M Flash 的网页	15
5. 串口打印调式信息	16
6. 怎样升级程序	17
6.1 通过串口升级	17
6.2 通过 HF 生产工具批量升级	19
7. Example	21
7.1 创建 AT 命令	22
7.2 定时器控制 nReady 灯闪烁	22
7.3 串口回调机制控制 nLink 灯状态	22
7.4 无线 OTA	22

1. SDK 说明

1.1 型号说明

HF-LPX30（或称为 HF-LPx30）系列模块是上海汉枫电子科技有限公司开发的一款低功耗嵌入式 Wi-Fi 模块，目前系列中包含但不限于 HF-LPT230、HF-LPT130、HF-LPB130 等型号。

本系列中所有模块可以使用同一份 SDK 和 API 手册，以下文档中以“HF-LPX30”统称此平台上模块，部分情况下也会以 HF-LPB130 表示此系列模块。

1.2 版本变更历史

2017-0810:

初版；

2017-1020:

补充完善文档格式；

增加型号 HF-LPT130；

增加 Keil+GCC 编译方式。

2018-0102:

增加平台差异性说明。

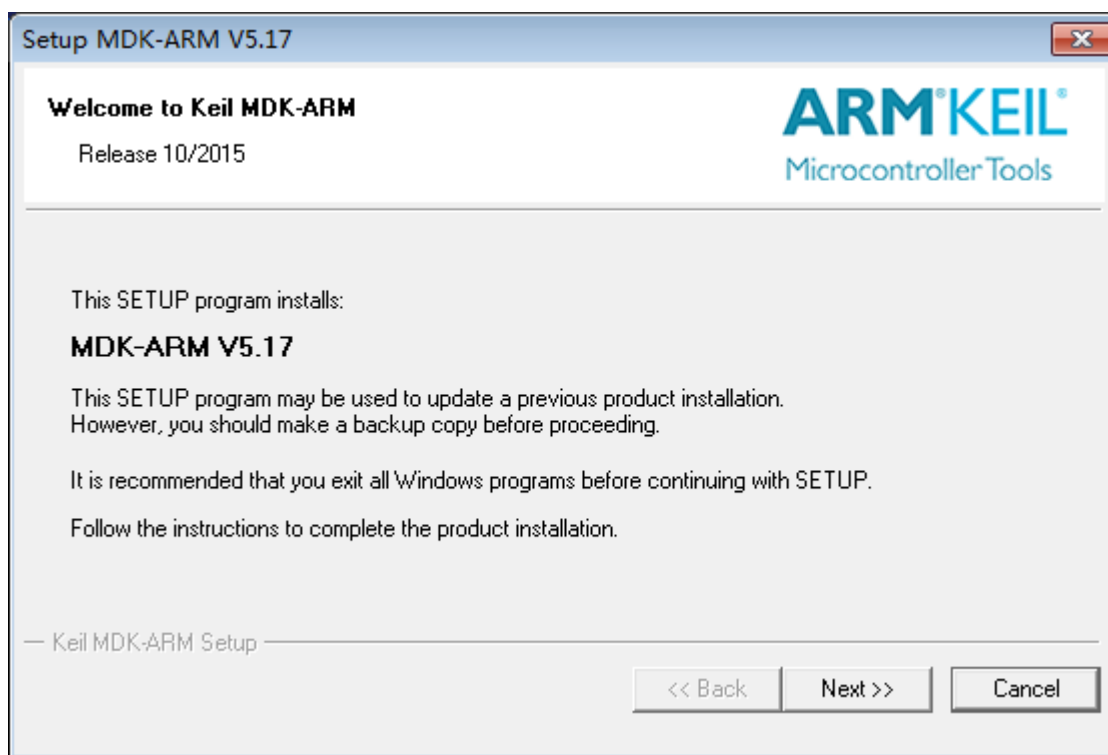
2. 编译环境安装

2.1 编译方式说明

HF-LPX30 SDK 采用 Keil+GCC 方式编译，SDK 中提供的 example、第三方库等都以此种方式编译提供。

2.2 搭建 Keil+GCC 编译环境

HF-LPX30 SDK 使用的是 MDK-ARM 5.17 编译，为避免兼容性问题，请安装如下参考版本 Keil_v5，下载链接地址：<http://pan.baidu.com/s/1nvHUQ1b>，下载完成后选择默认安装 Keil 即可。



GCC 请使用 5_4-2016q3 及以上版本，GCC 编译器请前往官网或如下地址下载：链接：<https://pan.baidu.com/s/1bpMwJ4F> 密码: 97gi，下载后根据视频安装。

安装后打开 windows 的命令行，输入 `arm-none-eabi-gcc --version` 查询是否已经安装成功，请确保是 5_4-2016q3 以上版本。



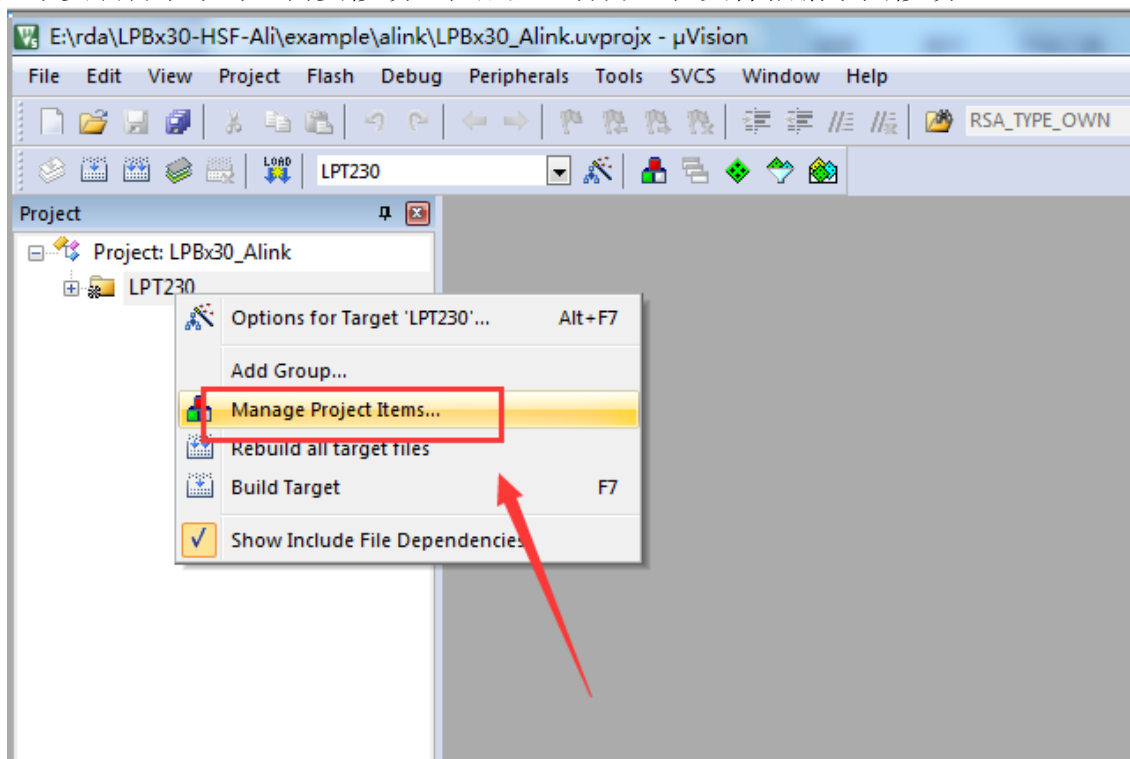
```
管理员: 命令提示符
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

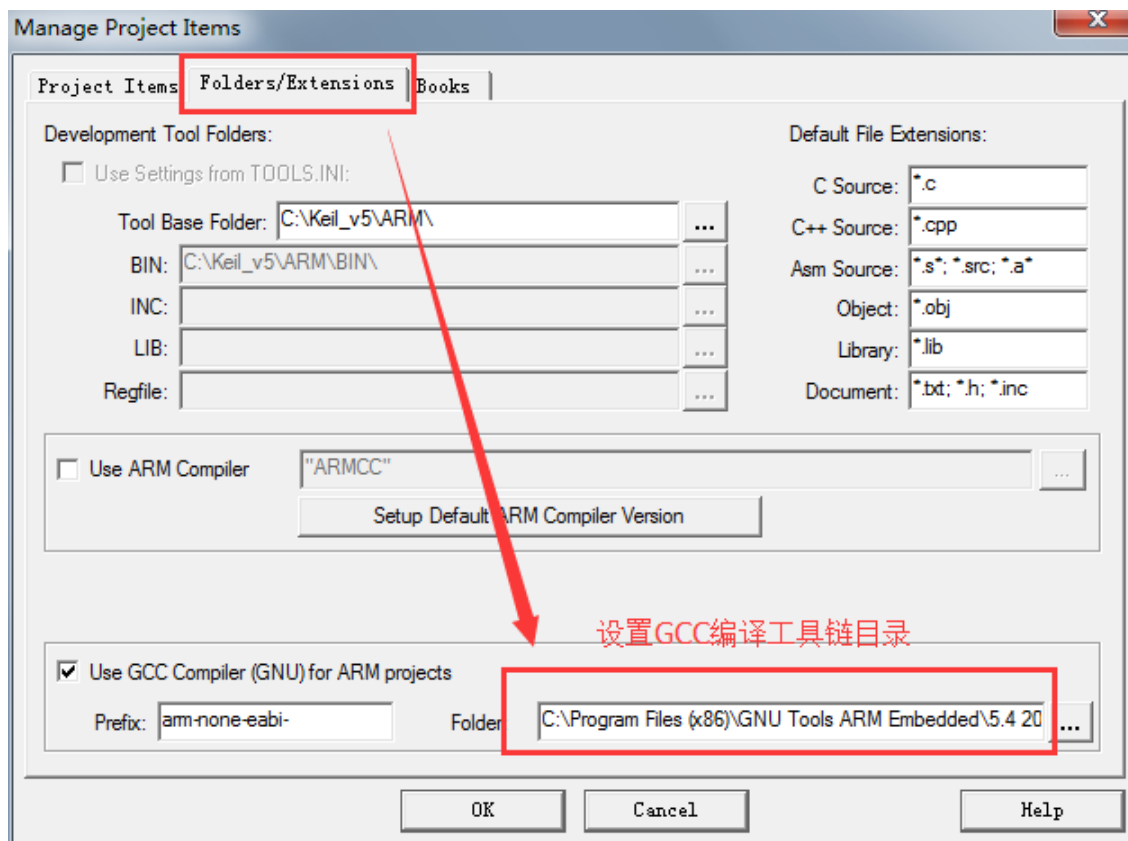
C:\Users\Administrator>arm-none-eabi-gcc --version
arm-none-eabi-gcc (GNU Tools for ARM Embedded Processors) 5.4.1 20160919 (release) [ARM/embedded-5-branch revision 240496]
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\Users\Administrator>
```









安装完 GCC 编译工具链后请确认您的安装目录是：C:\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3。

如果安装目录不对，需要修改工程配置，打开工程文件根据下图修改：





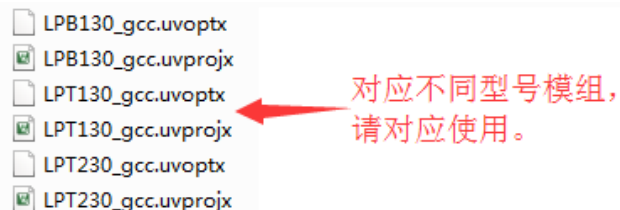
2.3 SDK 目录结构

 doc	使用文档、API手册	文件夹
 example	example代码	文件夹
 projects	编译工程文件	文件夹
 sdk	SDK lib库、头文件、链接脚本	文件夹
 src	程序入口，用户代码	文件夹
 thirdpartylib	第三方库	文件夹
 tools	工具	文件夹
 readMe.txt		文本文档

3. 开始编译

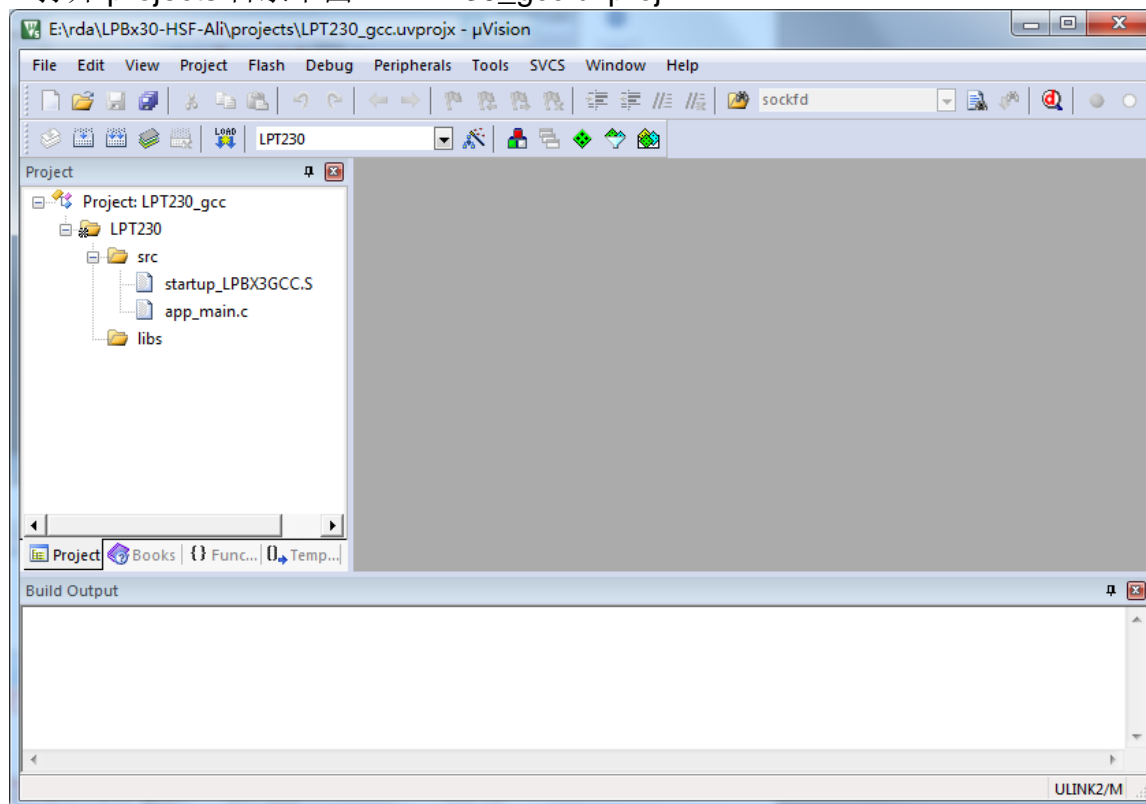
3.1 工程文件说明

请选择与模组型号对应的工程进行开发，型号选择错误可能导致 RF 性能变差、GPIO 引脚异常。

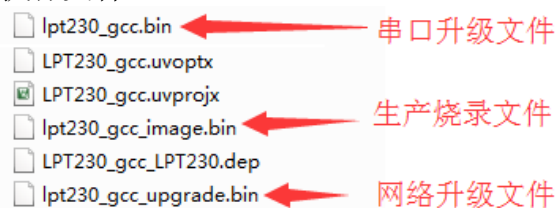


3.2 开始编译

1. 打开 projects 目录下面 “LPT230_gcc.uvprojx”



2 点击 “build”，生成执行文件



3.3 用户添加源代码文件

用户函数定义约定

返回类型 + **USER_FUNC** + 函数名称 + 参数

例如：

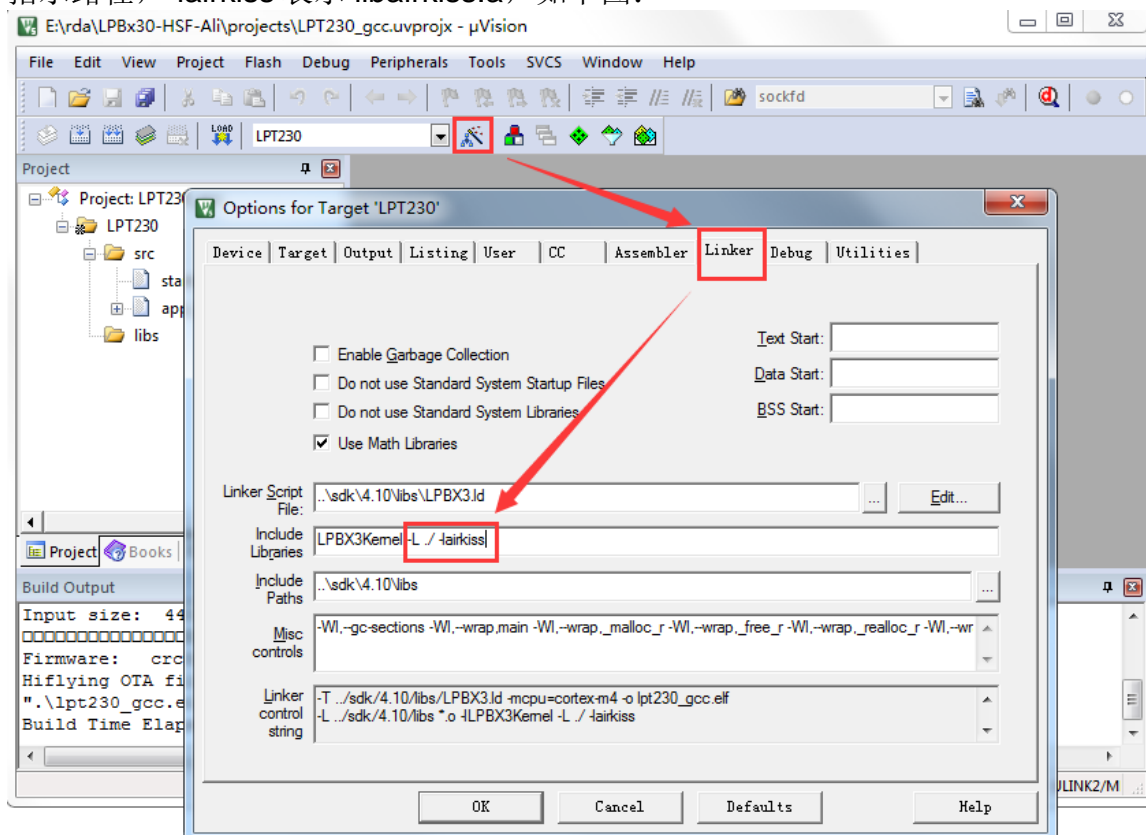
`void USER_FUNC test_func1(char *a);` `USER_FUNC` 为函数修饰符号，为了更好的兼容性请加上 `USER_FUNC` 这个标识，如果不加在有的平台编译出来的程序将无法运行

用户添加源代码文件

添加.c 文件，基于 HSF 的源文件都要包含 `<hsf.h>` 头文件，包含这个头文件后，源代码里面可以调用基于 HSF 的 API 函数；如果要使用 `libc` 接口函数，请 `#include` 相关的头文件，例如如果调用字符串操作函数 `#include <string.h>`，调用时间函数 `#include <time.h>` 等。

用户添加第三方库

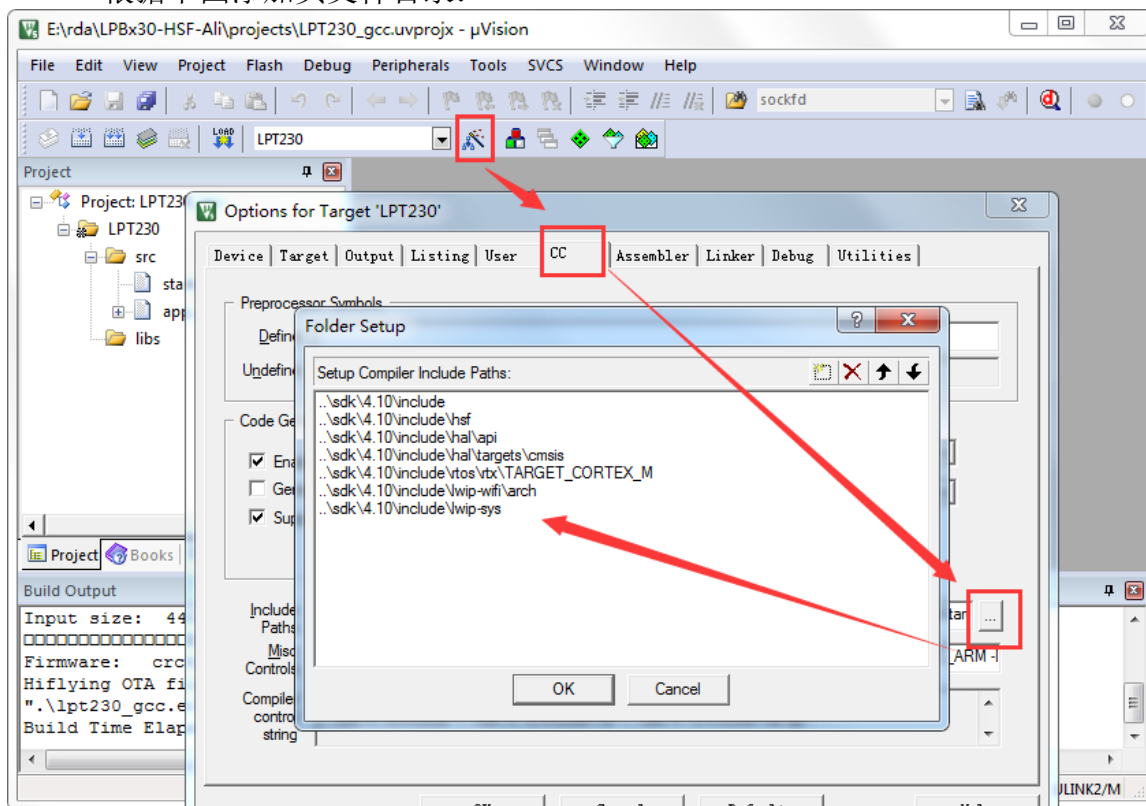
根据下图添加第三方库，例如：添加当前路径下的 `libairkiss.a` 库文件，`-L` 指示路径，`-lairkiss` 表示 `libairkiss.a`，如下图：



由于 GCC 编译器链接特性，可能需要反复链接库文件，例如：a 库调用 b 库函数，b 库调用 a 库函数，则链接需要写成：`-la -lb -la`，否则可能提示 b 库中调用的 a 库函数未定义。

用户添加头文件目录

根据下图添加头文件目录：



注意：

SDK 中已经集成 Cyassl、MQTT、websocket 等第三方库，可以直接在工程中添加使用，使用方法参考 example 下的例子。

为了利于 SDK 升级，请把不要在 app_main.c 文件里面添加太多代码，最好只需要添加自己的一个入口函数。其它的源文件都放在自己的目录下面，可以在 src 目录下面建一个目录放置自己的源代码，这样升级的时候，只需要把 app_main.c 文件修改一下，其它的 SDK 文件全部用新的替换就可以了。

3.4 开发注意事项

为避免影响到汉枫出厂测试而导致无法生产，在 SDK 开发过程中请注意以下几点：

1、不可影响启动自动连接路由器功能，例如：上电启动扫描路由器或上电进入 SmartLink 等都会影响汉枫的产测；

解决方案：在 Flash 中增加标志位或采用读取 GPIO 电平的方式来区分启动流程，如果采用 GPIO 方式请一定确保 GPIO 电平的稳定；

2、必须保留汉枫的 AT 命令功能，尽量不要修改汉枫的原始 AT 命令名；

3、尽量定义用户版本查询命令为 AT+APPVER，不可使用汉枫原始的 AT+VER 和 AT+LVER；

4、确保模块执行 AT+RELD 命令后波特率不会变化。

3.5 特别注意事项

1、由于编译环境特殊性，请不要在代码将指针和数组混合声明，如定义为数组 `char tmp[200]`，请不要采用指针形式的声明如：`char *tmp`，应使用 `char tmp[]`；

3.6 常见问题分析

1、Panic 问题

问题描述：程序运行中出现如下 Panic 打印，然后模块重启

```

R11 = 00000000
R12 = 1802a0bf
LR [R14] = 1800cc13
PC [R15] = 00000000
xPSR = 80000000
PRIMASK = 00000000, IRQ: Enable
FAULTMASK = 00000000
BASEPRI = 00000000
CONTROL = 00000004
SHCSR = 00070008
CFSR = 00020000
HFSR = 00000000
DFSR = 00000000
MMFAR = e000ed34
BFAR = e000ed38
AFSR = 00000000

[Detail report]
APSR Flags: NZCV
ProcStack: 00103518, MainStack: 0018ffb0

Memory Manage Faults
MM_FAULT_STAT:0x00
IACCVIOL :0 DACCVIOL :0
MUNSTKERR:0 MSTKERR :0
MLSPERR :0 MMARVALID:0

Bus Faults
BUS_FAULT_STAT:0x00
IBUSERR :0 PRECISERR:0
IMPRECISERR:0 UNSTKERR :0
STKERR :0 LSPERR :0
BFARVALID :0

Usage Faults
USG_FAULT_STAT:0x02
UNDEFINSTR:0 INVSTATE :1
INVPC :0 NOCP :0
UNALIGNED :0 DIVBYZERO:0

Stack Dump(sp: 00103518):
[00103500-0010351c]: 1800cba9 00000006 00182744 40119420 00000000 00000000 00000000
1 00000000
[00103520-0010353c]: 00002800 00105970 1802a0bf 1800cc13 00000000 80000000 00000000
0 00000000
[00103540-0010355c]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0 00000000
[00103560-0010357c]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0 00000000
[00103580-0010359c]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0 00000000
[001035a0-001035bc]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0 00000000
[001035c0-001035dc]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0 00000000
[001035e0-001035fc]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0 00000000
[Report done]
Panic...

```

问题分析：上述问题一般是由程序中内存越界引起的，可以查看打印中 LR 指针和 PC 指针的值来分析，PC 指针表示当前函数运行代码的地址，LR 指针为子函数的返回地址，也就是上一个函数地址；

图中 LR 指针的地址为 0x1800cc13，PC 指针地址为 0x00000000，函数地址应该是 0x18000000 开始的，说明 PC 指针已经被破坏不具备参考意义了；

对应查看编译出的 map 文件，uart_recv_callback 函数的地址范围是 0x1800cba8-0x1800cc20，可以分析出 LR 指针运行在 uart_recv_callback 函数内，内存越界应该出现在此函数中。

```
.text.airkiss_check
0x1800c960      0x1a0 ../BUILD/UNO_91H/GCC_ARM/./airkiss.o
0x1800c960      airkiss_check
.text.airkiss_fix_channel
0x1800cb00      0x3c ../BUILD/UNO_91H/GCC_ARM/./airkiss.o
0x1800cb00      airkiss_fix_channel
.text.socketb_recv_callback
0x1800cb3c      0x38 ../BUILD/UNO_91H/GCC_ARM/./app_main.o
.text.socketa_recv_callback
0x1800cb74      0x34 ../BUILD/UNO_91H/GCC_ARM/./app_main.o
.text.uart_recv_callback
0x1800cba8      0x78 ../BUILD/UNO_91H/GCC_ARM/./app_main.o
.text.sys_event_callback
0x1800cc20      0xa4 ../BUILD/UNO_91H/GCC_ARM/./app_main.o
.text.app_init
0x1800ccc4      0xc ../BUILD/UNO_91H/GCC_ARM/./app_main.o
0x1800ccc4      app_init
.text.app_main
0x1800ccd0      0xf8 ../BUILD/UNO_91H/GCC_ARM/./app_main.o
0x1800ccd0      app_main
.text._ZL9add_replyv
0x1800cdc8      0xc0 ../BUILD/UNO_91H/GCC_ARM/./assis_thread.o
.text._ZL15m2m_assis_writePKc
0x1800ce88      0x6c ../BUILD/UNO_91H/GCC_ARM/./assis_thread.o
0x1800ce88      m2m_assis_write(char const*)
.text._ZL13ASSIS_ReceivePvP7udp_pcbP4pbufPK8ip4_addrt
0x1800cef4      0x350 ../BUILD/UNO_91H/GCC_ARM/./assis_thread.o
```

2、RTX error 问题

问题描述：程序运行中出现如下 RTX error 打印，然后模块重启

RTX error code: 0x00000001, task ID: 0x00188EA0

问题分析：上述问题一般由线程异常引起，task ID 表示异常线程的 ID 号，error code 一般有如下几个值：

0x00000001 表示线程栈溢，可以通过 hftthread_show 查询所有由 hftthread_create 创建的线程的状态，包含 task ID、优先级、栈使用情况，也可以通过 hftthread_get_current_taskid 查询当前线程的 task ID；

0x00000004 表示定时器栈溢出，可以减少定时器回调函数中的处理代码，或者通过在 app_init 中调用 hftimer_set_driven_typ(HFTIMER_DRIVEN_BY_THREAD)函数将定时器驱动方式改为 thread 方式。

3、DRAM overflowed 问题

问题描述：编译提示 DRAM overflowed，编译失败

问题分析：上述问题是由于 DRAM（静态变量区）溢出引起，可以将代码中部分静态变量改为 malloc。

4. 资源分配

4.1 1MB Flash 资源分配

0x0000 0000	SYSTEM_SECTOR (4KB) 系统用
0x0000 1000	BOOT(12KB) Bootloader 区域
0x0000 4000	BOOT_CONFIG(12KB) 系统用
0x0000 7000	USERPAGE (4KB) 用户参数保存区域
0x0000 8000	USERPAGE_BACKUP (4KB) 用户参数保存备份区域
0x0000 9000	F_SETTING (4KB) 出厂参数保存区域
0x0000 A000	USER_BIN_FILE (4KB) hffile_userbin_write API 接口实际物理地址
0x0000 B000	USER_BIN_BACK_FILE (4KB) hffile_userbin_write API 接口实际物理地址备份区域
0x0000 C000	CODE (588KB) 运行代码区
0x0009 F000	OTA UPGRADE(372KB) OTA 升级备份区域
0x000F C000	WEB (8KB) 外部网页
0x000F E000	UFLASH (8KB) 用户 flash

4.2 2MB Flash 资源分配

0x0000 0000	SYSTEM_SECTOR (4KB) 系统用
0x0000 1000	BOOT(12KB) Bootloader 区域
0x0000 4000	BOOT_CONFIG(12KB) 系统用
0x0000 7000	USERPAGE (4KB) 用户参数保存区域
0x0000 8000	USERPAGE_BACKUP (4KB) 用户参数保存备份区域

0x0000 9000	F_SETTING (4KB) 出厂参数保存区域
0x0000 A000	USER_BIN_FILE (4KB) hffile_userbin_write API 接口实际物理地址
0x0000 B000	USER_BIN_BACK_FILE (4KB) hffile_userbin_write API 接口实际物理地址备份区域
0x0000 C000	CODE (800KB) 运行代码区
0x000D 4000	UFLASH (160KB) 用户 flash
0x0010 0000	OTA UPGRADE(512KB) OTA 升级备份区域
0x0018 0000	WEB (200KB) 外部网页
0x001B 2000	未使用区域(312KB)

4.3 4MB Flash 资源分配

0x0000 0000	SYSTEM_SECTOR (4KB) 系统用
0x0000 1000	BOOT(12KB) Bootloader 区域
0x0000 4000	未使用区域(32KB)
0x0000 C000	CODE (1040KB) 运行代码区
0x0011 0000	OTA UPGRADE(1008KB) OTA 升级备份区域
0x0020 C000	UFLASH (460KB) 用户 flash
0x0028 0000	WEB(512KB) Web 网页地址
0x0030 0000	WEB_SCAN(16KB)
0x0030 4000	BOOT_CONFIG(12KB) 系统用
0x0030 A000	USER_BIN_FILE (4KB) hffile_userbin_write API 接口实际物理地址
0x0030 B000	USER_BIN_BACK_FILE (4KB) hffile_userbin_write API 接口实际物理地址备份区域
0x0030 C000	未使用区域(16KB)

0x0031 0000	UFLASH1 (240KB) 用户 flash
-------------	-----------------------------

4.4 Ram 资源

HF-LPX30 模块剩余 80KB 左右 Ram 可供使用。

全局变量会占用静态 RAM 空间，总共剩余有 20KB 空间。

malloc，线程栈空间等都会占用动态 RAM 资源。总共剩余约 60KB 空间。

4.5 关于 1M Flash 的网页

由于 1M Flash 空间有限，仅提供 8K flash 空间存放网页文件，sdk/tools 目录下有简单的中英文网页可供升级使用，升级方法：打开模块所在路径的 iweb.html 页面，选择“Upgrade customized webpage”进行升级，如模块的 IP 为 192.168.11.100，则使用浏览器打开 <http://192.168.11.100/iweb.html> 进行升级。

5. 串口打印调式信息

如果程序想通过串口打印调式信息，HSF 中提供了 `u_printf` 和 `HF_Debug` 两个 API 函数，默认情况下程序中调用这两个函数是不会有打印信息出来的，因为默认调式是关闭的，要通过 `hfdbg_set_level(X)` 打开调式串口输出，X 代表输出的调试信息等级，或者使用 AT 命令来打开调试信息输出“AT+NDBG=2,0”打开，“AT+NDBG=0,0”关闭。调试信息就会从串口 1 输出(模块 UART1_TX 引脚)，如下图。

(注.程序最后发布的时候要把 debug 模式关闭)

```
boot_main->start
boot_main->end ver1.09

D4 EE 07 2D 14 1E
sta channel=11

*****OTA FLAG:00000000 0 a5010203*****
uart thread start 8

HF-LPT120 Start Nov 26 2015 15:14:30

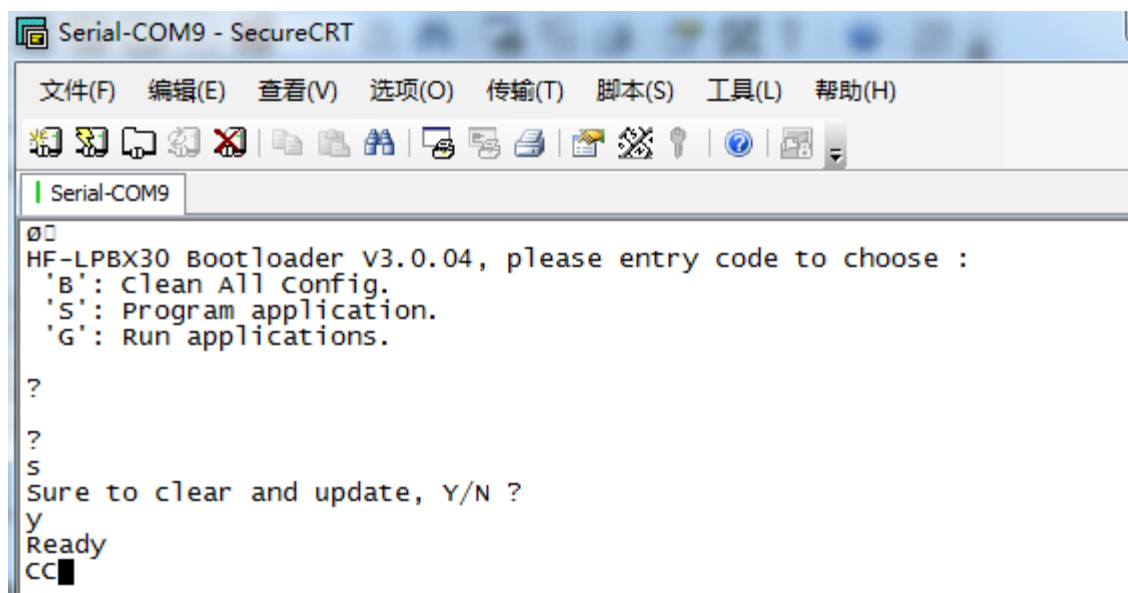
Listen Port 8899
wifi connecting.....
[handle_dhcp] +++
dhcp : init
tx_probe_req +++
[rx_probe_rsp] : probe_response->capability = 0x 11
[rx_probe_rsp] : gCabrioConf.wifi_security = 3
[rx_probe_rsp] : ---
[tx_authentication_req_seq1] : +++
[tx_association_req] : +++
[rx_process_eapol] : +++
[rx_process_wpa] : +++ : eapol_key->type = 2, eapol_key->key_info = 0x008a

AT... AT... AT... AT... AT+Z ND... TLS... +++ a NETP Defau
```

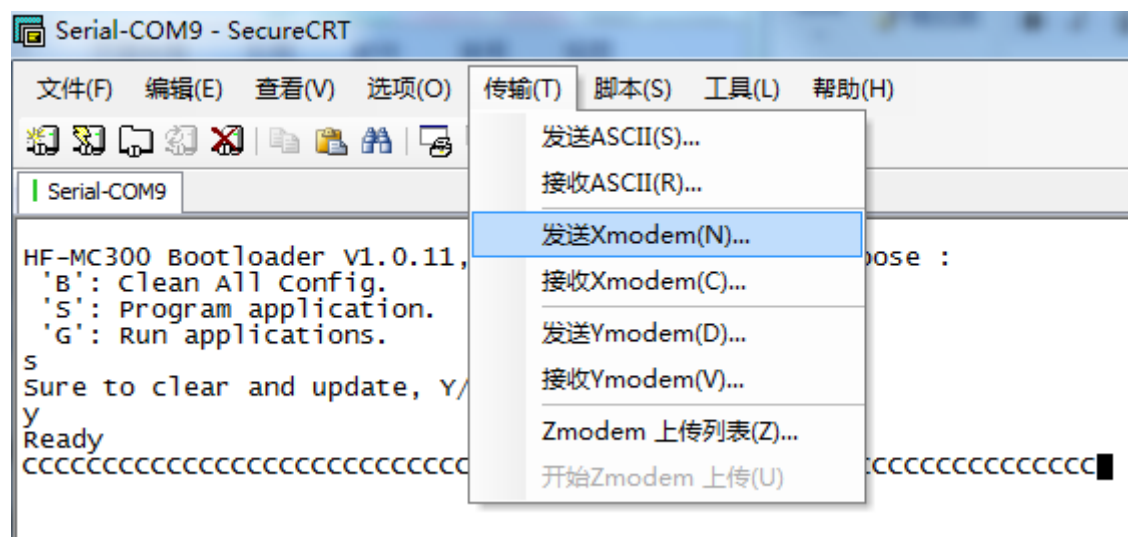

6. 怎样升级程序

6.1 通过串口升级

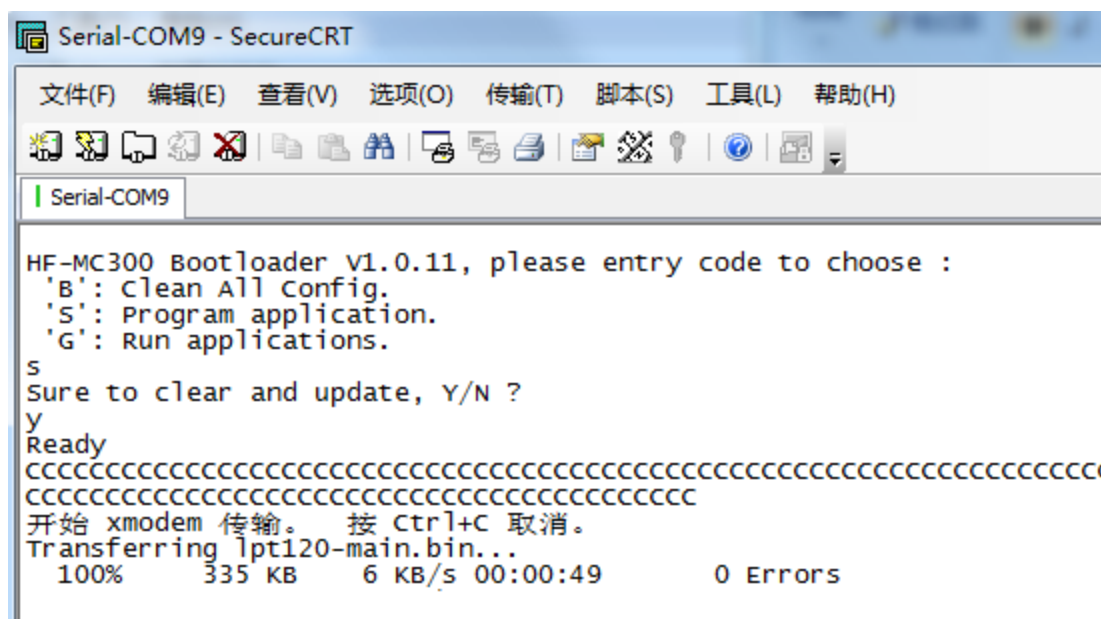
Step 1: 配合 SecureCRT 软件在 **230400** 波特率下，按住模块的 nReload 按键（拉低）重启模块，1 秒后输入一个空格字符，可以进入模块的 Bootloader 进行升级，成功进入后出现如下界面：



Step 2: 输入命令'S'，输入确认'Y'进行文件升级，打开 SecureCRT 的 Xmode 传输发送文件，如下图：

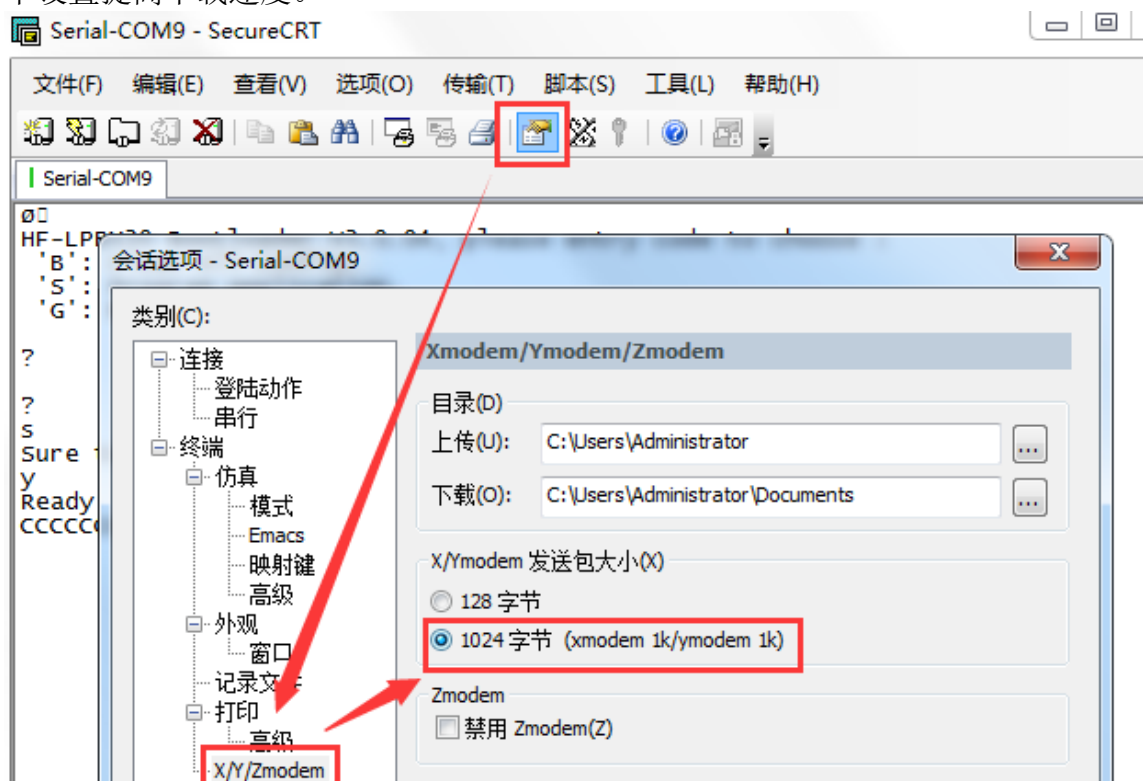


Step 3: 选择需要升级的文件，虚拟机编译出不带 upgrade 后缀的 bin 文件。



Step 4: 等待传输完成后重启模块即可，如果升级到一半卡住请重启模块和软件重新进入 Bootloader 开始升级。

注意：LPBx30 系列 Bootloader 支持 1024 字节传输的 X modem 协议，可以进行以下设置提高下载速度。



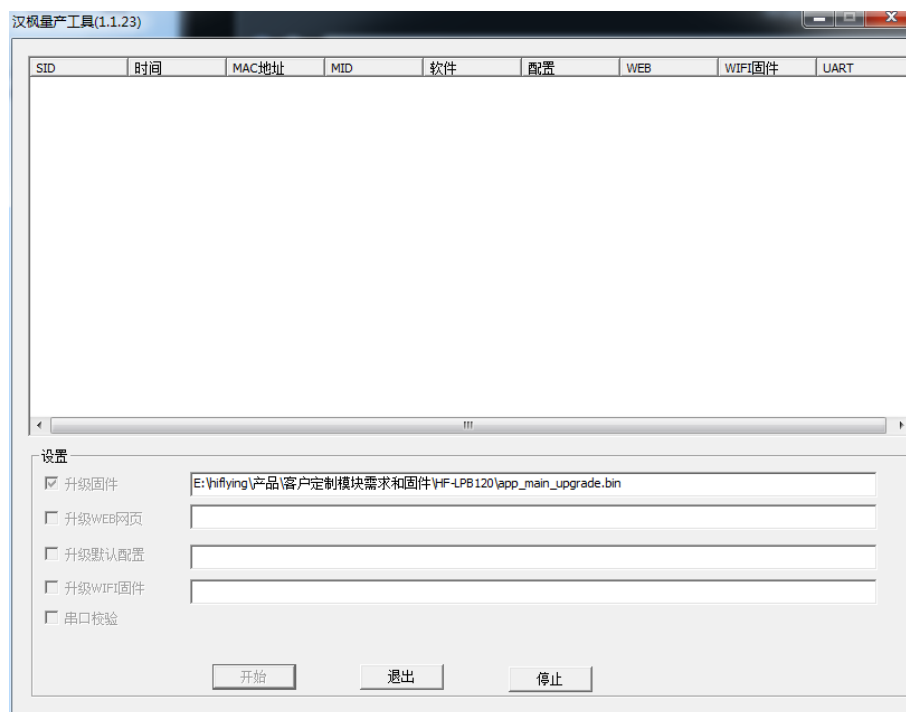
6.2 通过 HF 生产工具批量升级

Step 1: 从汉枫官网下载生产工具。

http://gb.hi-flying.com/download_detail_dc/&downloadslid=1822d146-343d-4332-af8b-137c0fb4d967.html



Step 2: PC 连接到路由器，打开工具中的 HFUpdate.exe 工具并加载升级文件 app_main_upgrade.bin，若不能打开，请安装 gtk2-runtime 运行环境，工具使用过程中需要关闭电脑防火墙，且最好把工具放到磁盘根目录下（不要带中文路径名）。



Step 3: 配置模块连接到 PC 所连接的同一路由器下。

```

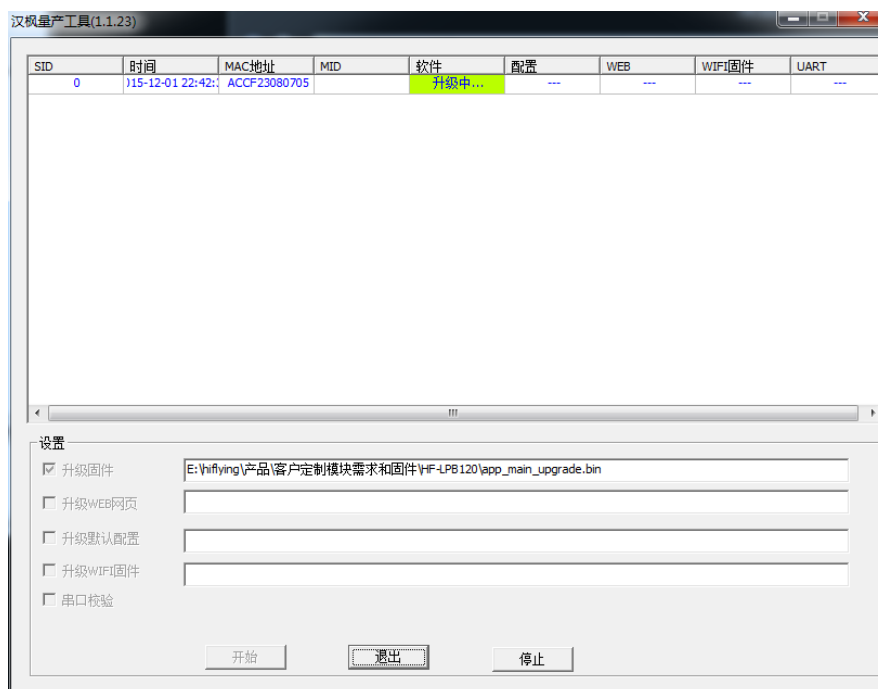
AT+WMODE
+ok=STA

AT+WSSSID
+ok=Sam401

AT+WSKEY
+ok=WPA2PSK,AES,gongyuhui

AT+WANN
+ok=DHCP,192.168.199.147,255.255.255.0,192.168.199.1
  
```

Step 4: 输入 AT+OTA 命令执行升级，上位机软件会显示此模块的信息，等待升级完成，打开调试信息输出可查看升级过程。



Step 5: 升级完成后，debug 输出如下信息表明升级完成，可运行测试新程序了，上位机界面可能会一直提示升级中，不予理会。

```

image file size=000534b4(341172) checksum=02120e54
*write boot image to flash!
*****D4 EE 07 2D 14 1E
sta channel=11

*****OTA FLAG:0505AA50 0 a5010202*****
*
HF-LPT120 Start Nov 28 2015 01:52:06
  
```

注意：

串口升级和量产工具升级所使用的升级文件是不同的，汉枫默认固件中带有 **upgrade** 的方式是可以通过量产工具进行升级的。

app_main_upgrade.bin 是在 **app_main** 的基础上增加了 **CRC** 校验的算法，避免了量产工具升级错误的文件导致无法启动。

7. Example

1. HSF-LPBx30 SDK 提供了相应功能的 **example**，可以在 **example** 文件夹下找到相应的工程文件和代码。可以通过修改 **example.h** 文件中的宏 **EXAMPLE_USE_DEMO** 选择要编译哪一个例子，修改后即可运行 **example** 工程编译，默认编译为 **HF-LPT230** 型号模块。运行例子需要通过 “**AT+NDBGL=2**” 来设置消息显示级别为 2，这样 **u_printf** 函数可以通过串口打印调试信息。

```
#define _H_EXAMPLE_H_H_H_

#define USER_AT_CMD_DEMO 1

#define USER_GPIO_DEMO 2

#define USER_TIMER_DEMO 3

#define USER_THREAD_DEMO 4

#define USER_CALLBACK_DEMO 5

#define USER_FILE_DEMO 6

#define USER_FLASH_DEMO 7

#define USER_SOCKET_DEMO 8

#define USER_IR_DEMO 9

#define USER_URL_DEMO 10

#define WIFI_TEST_DEMO 11

#define UPDATE_TEST_DEMO 12

#define SSL_TEST_DEMO 13

#define UART_OP_DEMO 14

#define USER_UPNP_DEMO 15

#define WEBSOCKET_TEST_DEMO 16

//通过下面可以选择不同的例子进行编译
#define EXAMPLE_USE_DEMO USER_AT_CMD_DEMO
```

7.1 创建 AT 命令

代码在 `example/at/attest.c`，通过这个例子可以了解怎么样添加用户自定义 AT 命令。

7.2 定时器控制 nReady 灯闪烁

代码在 `example/timer/timertest.c`，通过这个例子可以了解线程的创建，定时器的创建，以及相关的 API 函数的用法

运行结果 nReady 灯以 1HZ 的频率闪烁。

7.3 串口回调机制控制 nLink 灯状态

代码在 `example/netcallback/callbacktest.c`，通过这个例子可以熟悉串口发送 API,以及串口回调处理机制。

执行结果，当从串口主动发送“GPIO NLINK LOW”给模块，nLink 灯处于低电平，当从串口主动发送“GPIO NLINK HIGH”给模块，nLink 灯处于高电平，当从串口主动发送“GPIO NLINK FLASH”给模块，nLink 灯以 1HZ 的频率闪烁。

7.4 无线 OTA

代码在 `example/update/updatetest.c`，通过这个例子可以熟悉无线 OTA 相关的 API。

执行结果，串口使用命令“AT+UPGRADESW=http://192.168.1.1/update.bin”进行无线升级。

© Copyright High-Flying, Jan, 2016

The information disclosed herein is proprietary to High-Flying and is not to be used by or disclosed to unauthorized persons without the written consent of High-Flying. The recipient of this document shall respect the security status of the information.

The master of this document is stored on an electronic database and is “write-protected” and may be altered only by authorized persons at High-Flying. Viewing of the master document electronically on electronic database ensures access to the current issue. Any other copies must be regarded as uncontrolled copies.

<结束>