

High-Flying Software Framework (HSF-MC300) 用户手册 V1.1x

版本 1.1x
2016 年 02 月

更新记录:

修改时间	作者	修改	删除
2015.12.01	Sam	初版	
2016.01.10	LiuBo	添加 example 说明	
2016.02.15	LiuBo	修改 SDK 文件架构	

目录

1. SDK 说明.....	4
1.1 LPT120/LPB120/LPT220 说明.....	4
1.2 版本变更历史.....	4
2. 编译环境安装.....	5
2.1 基于 Ubuntu 12.04 的编译环境安装.....	5
2.2 HSF-MC300 SDK 目录结构.....	6
3. 开始编译.....	7
3.1 编译 LPT120/LPB120/LPT220.....	7
3.2 用户添加源代码文件.....	7
3.3 编译常见问题.....	8
4. MC300 资源分配.....	10
4.1 2MB Flash 资源分配.....	10
4.2 Ram 资源.....	10
4.3 Ram 函数.....	11
5. 串口打印调试信息.....	12
6. 怎样升级程序.....	13
5.1 通过串口升级.....	13
5.2 通过 HF 生产工具批量升级.....	15
7. Example.....	17
6.1 创建 AT 命令.....	17
6.2 自定义 GPIO.....	17
6.3 定时器控制 nReady 灯闪烁.....	18
6.4 串口回调机制控制 nLink 灯状态.....	18
6.5 任务创建切换.....	18
6.6 uFlash 的使用.....	18
6.7 无线 OTA.....	18
6.8 创建 TCP Server.....	18
6.9 创建 TCP Client.....	19
6.10 创建 UDP.....	19

1.SDK 说明

1.1 LPT120/LPB120/LPT220 说明

LPT120/LPB120/LPT220 同为上海汉枫科技基于 MC300 平台开发的串口转 WiFi 模块，可以使用同一份 SDK 和 API 手册，以下文档中以“MC300”统称此平台上模块。从软件角度来讲，三款模块主要是 PCB Layout 不同，导致 GPIO 的对应和 RF 值不一样，如果没有匹配使用会导致 GPIO 无法对应和无线信号较差。

1.2 版本变更历史

2016-0215:

修改 SDK 文件架构；

2016-0110:

支持 example；

2015-1126:

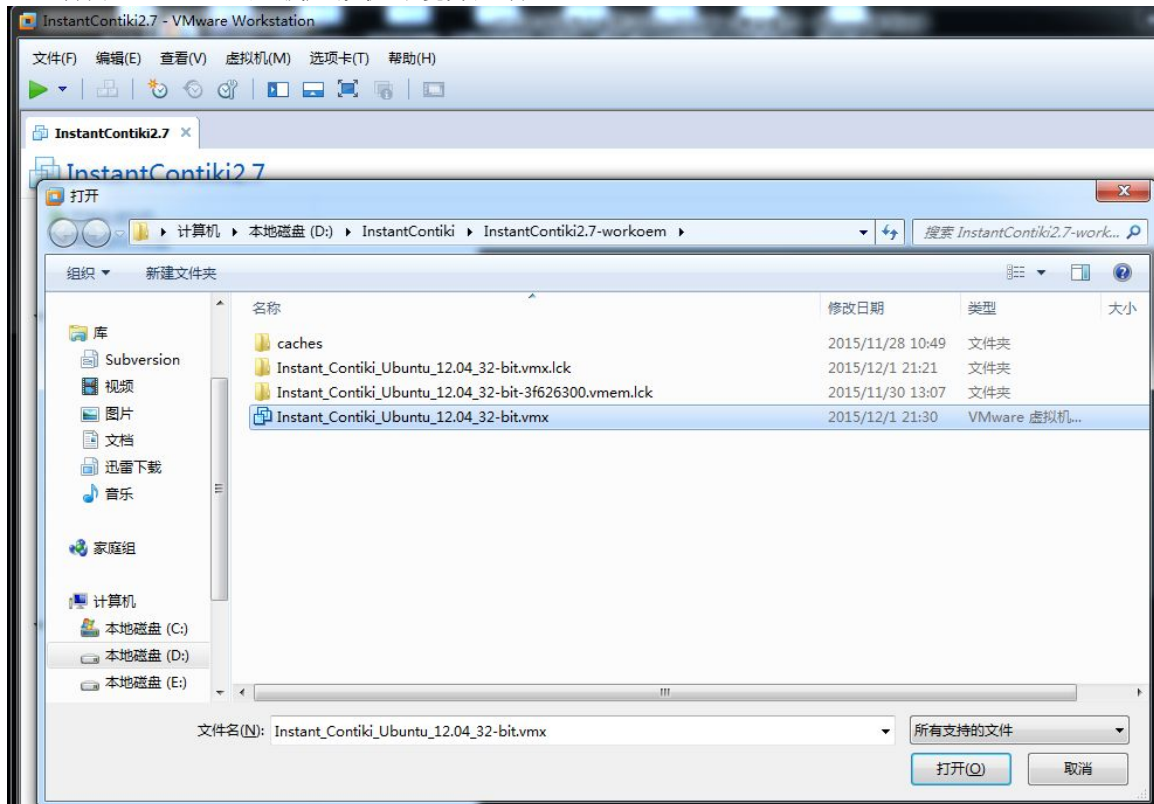
初版；

2. 编译环境安装

MC300 采用的是 Linux 的编译环境，请使用我们提供的虚拟机环境。

2.1 基于 Ubuntu 12.04 的编译环境安装

1. 为避免兼容性请安装 VMware 10.0 版本。
链接: <http://pan.baidu.com/s/1nukpk85>
密码: hvtf
2. 从以下链接下载编译所用虚拟机环境。
链接: <http://pan.baidu.com/s/1dDkEqVR>
密码: 0shn
3. 打开 VMware，加载虚拟机环境并启动。



4. 输入密码 user 登录账户（账户名: user）
5. 复制 LPT120 SDK 到系统中，进入以下目录。
`user@iot-work:~/Desktop/LPT120-HSF-20151128/examples/LPT120$`
6. 输入 make 进行编译。

```
user@iot-work:~/Desktop/LPT120-HSF-20151128/examples/LPT120$ make
```

7. 生成 app_main.bin 和 ap_main_upgrade.bin 升级文件。

```
user@iot-work:~/Desktop/LPT120-HSF-20151128/examples/LPT120$ ls
app_main.asm  app_main.elf  dump.log  ssv6060_config.hex
app_main.bin  app_main_upgrade.bin  Makefile
app_main.c    contiki-mc300.a  Makefile.target
app_main.co   contiki-mc300.map  smartTye.h
```

8. 输入 make clean 清除之前生成的 bin 文件。

```
user@iot-work:~/Desktop/LPT120-HSF-20151128/examples/LPT120$ make clean
using saved target 'mc300'
rm -f *~ *core core *.srec \
        *.lst *.map \
        *.cprg *.bin *.data *.firmware core-labels.S *.ihex *.ini \
        *.ce *.co
rm -rf %.elf app_main.elf
rm -rf obj_mc300
```

2.2 HSF-MC300 SDK 目录结构

	doc	使用文档，API手册	2016/2/16 星期...	文件夹	
	example	example代码	2016/2/16 星期...	文件夹	
	sdk	SDK lib和头文件	2016/2/16 星期...	文件夹	
	src	程序入口，用户代码	2016/2/16 星期...	文件夹	
	thirdpartylib	第三方库	2016/2/16 星期...	文件夹	
	tools	工具	2016/1/27 星期...	文件夹	
	util	编译链文件	2016/2/16 星期...	文件夹	
	boot.s		2016/1/27 星期...	S 文件	13 KB
	hfrelease		2016/1/27 星期...	文件	4 KB
	LICENSE		2016/1/27 星期...	文件	2 KB
	main.c		2016/1/27 星期...	C 文件	1 KB
	Makefile		2016/2/15 星期...	文件	5 KB
	Makefile.mk		2016/1/27 星期...	MK 文件	1 KB
	mc300.lds		2016/1/27 星期...	LDS 文件	10 KB
	mc300_mac.hex		2016/1/27 星期...	HEX 文件	8 KB

3. 开始编译

3.1 编译 LPT120/LPB120/LPT220

HSF MC300 SDK 适用于 LPT120/LPB120/LPT220 这三种模块，默认编译出的固件适用于 HF-LPT120 模块，如果需要编译 HF-LPB120 的固件请修改根目录下的 Makefile 文件，将宏 `__HF_MODULE_ID__` 改为相应的模块即可，如下图：

```
TARGET=mc300
CRT0 = boot.s
FLASH_LAYOUT=layout_lpt120
ADD_LIB=-L ./thirdpartylib -lairkiss

SDK_DIR = $(shell pwd)/sdk/2.03

INCLUDE = -I$(SDK_DIR)/include -I$(SDK_DIR)/include/net -I$(SDK_DIR)/include/bsp -I$(SDK_DIR)
INCLUDE += -I$(SDK_DIR)/include/hsf/include -I$(SDK_DIR)/include/matrixssl
TOOLS_CFLAGS = -I$(SDK_DIR)/include

__HF_MODULE_ID__ = $(HFM_LPB120)
```

3.2 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。

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。

修改 `src` 文件夹下的 Makefile 增加源文件，宏定义，头文件定义：

```
include ../Makefile.mk

CLEAN += %.elf $(CONTIKI_PROJECT).elf

### Compilation rules

# Don't treat %.elf %.bin as an intermediate file!
.PRECIOUS: %.elf %.bin

OBJECTDIR=objs

CONTIKI_SRC = \
app_main.c \
Application/custom.c 增加源文件 custom.c

APPCFLAGS = -DSUPPORT_UART_THROUGH \
            -DUSER_APP_MACRO \ 增加宏定义：USER_APP_MACRO
            -I ./Application/ 增加头文件：./Application/

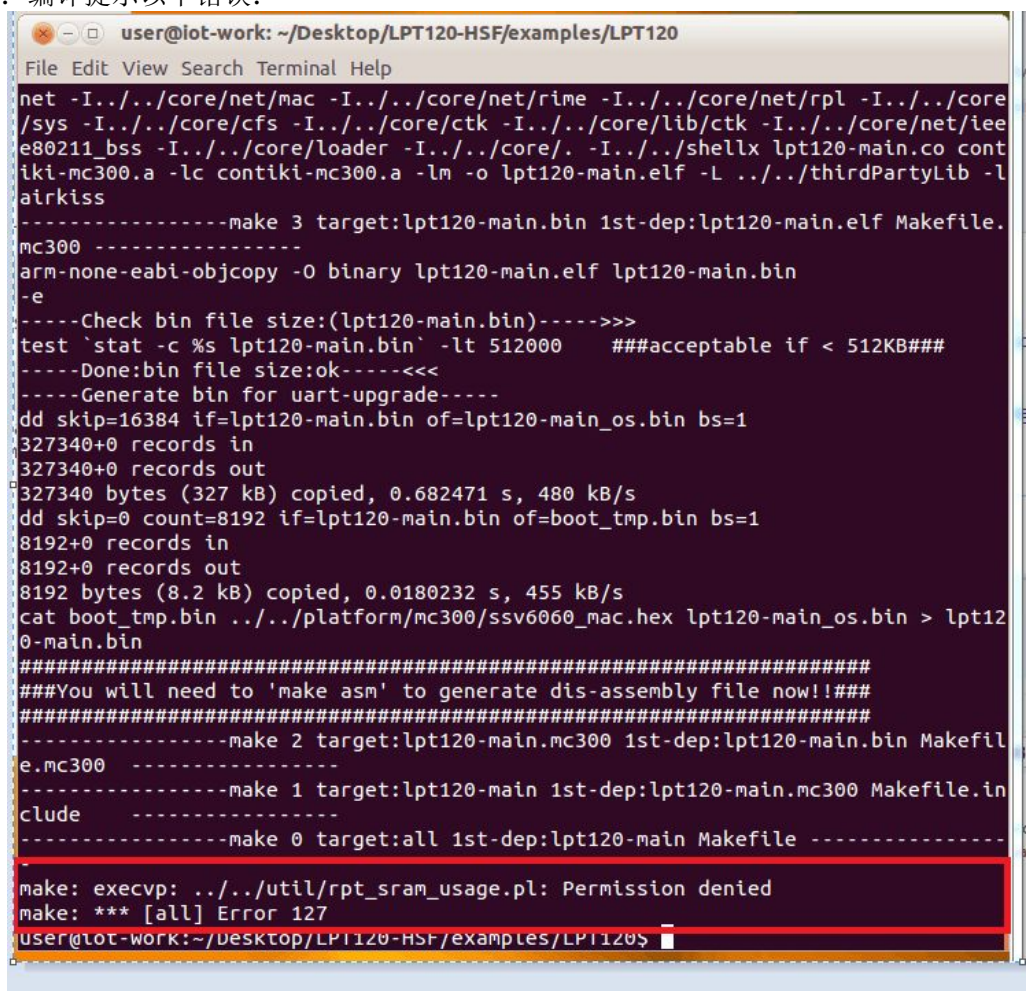
CONTIKI_OBJS=$(addprefix $(OBJECTDIR)/,$(CONTIKI_SRC:.c=.o) $(CONTIKI_SRC:.c=.o))

all:
@mkdir -p objs
@mkdir -p objs/Application 增加创建 objs/Application 文件夹
make userapps.a
```


3.3 编译常见问题

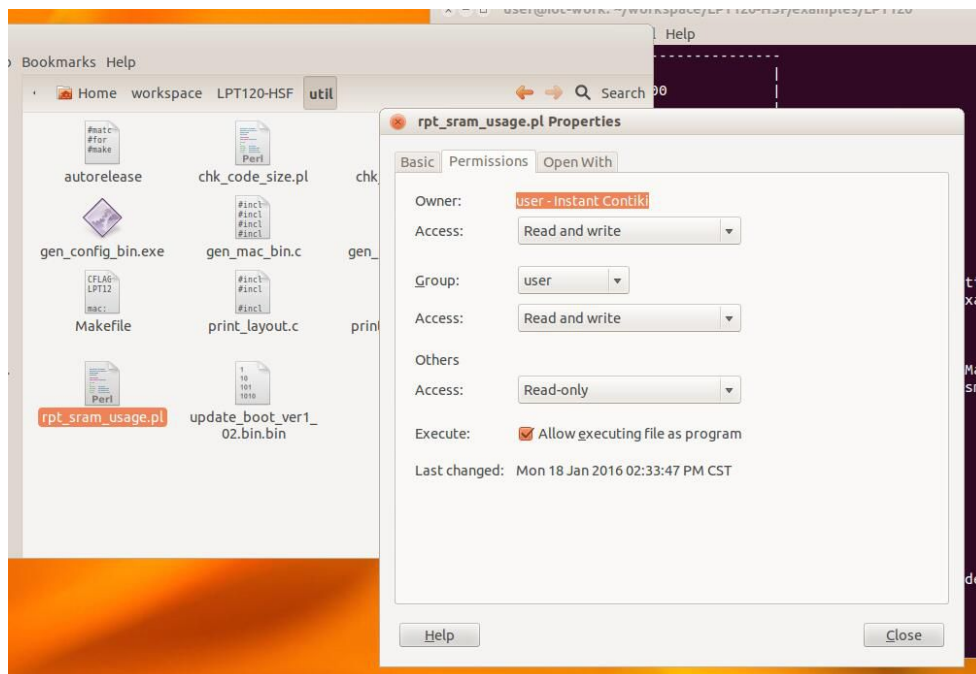
常见问题一：

问题：编译提示以下错误：



```
user@iot-work: ~/Desktop/LPT120-HSF/examples/LPT120
File Edit View Search Terminal Help
net -I../core/net/mac -I../core/net/rime -I../core/net/rpl -I../core
/sys -I../core/cfs -I../core/ctk -I../core/lib/ctk -I../core/net/iee
e80211_bss -I../core/loader -I../core/. -I../shellx lpt120-main.co cont
iki-mc300.a -lc contiki-mc300.a -lm -o lpt120-main.elf -L ../thirdPartyLib -l
airkiss
-----make 3 target:lpt120-main.bin 1st-dep:lpt120-main.elf Makefile.
mc300 -----
arm-none-eabi-objcopy -O binary lpt120-main.elf lpt120-main.bin
-e
-----Check bin file size:(lpt120-main.bin)----->>>
test `stat -c %s lpt120-main.bin` -lt 512000    ###acceptable if < 512KB###
-----Done:bin file size:ok-----<<<
-----Generate bin for uart-upgrade-----
dd skip=16384 if=lpt120-main.bin of=lpt120-main_os.bin bs=1
327340+0 records in
327340+0 records out
327340 bytes (327 kB) copied, 0.682471 s, 480 kB/s
dd skip=0 count=8192 if=lpt120-main.bin of=boot_tmp.bin bs=1
8192+0 records in
8192+0 records out
8192 bytes (8.2 kB) copied, 0.0180232 s, 455 kB/s
cat boot_tmp.bin ../../platform/mc300/ssv6060_mac.hex lpt120-main_os.bin > lpt12
0-main.bin
#####
###You will need to 'make asm' to generate dis-assembly file now!!###
#####
-----make 2 target:lpt120-main.mc300 1st-dep:lpt120-main.bin Makefil
e.mc300 -----
-----make 1 target:lpt120-main 1st-dep:lpt120-main.mc300 Makefile.in
clude -----
-----make 0 target:all 1st-dep:lpt120-main Makefile -----
make: execvp: ../../util/rpt_sram_usage.pl: Permission denied
make: *** [all] Error 127
user@iot-work:~/Desktop/LPT120-HSF/examples/LPT120$
```

解决方式：修改 rpt_sram_usage.pl 文件的属性。



4. MC300 资源分配

4.1 2MB Flash 资源分配

0x0000 0000	Boot Sector (8KB)
0x0000 2000	MP (8KB)
0x0000 4000	Code (512KB)
0x0000 8400	Not Used(224KB)
HF System Config(16KB)	
Maker (4KB)	
0x000C 0000	File System (124KB)
Not Used (124KB)	
0x000F F000	Boot Config (4KB)
Not Used (512KB)	
0x0018 0000	Upgrade Image (512KB)

Boot Sector:

MP:

Code: 运行代码区, 最大 512KB, 生成的 app_main.bin 不能超过此大小。

Not used: 未使用, 可供自由操作。

HF System Config: 模块工作参数保存区域

Maker:

File System:

Boot Config:

Upgrade Image: 运行代码备份区, 最大 512KB 用于 OTA 升级。

4.2 Ram 资源

总共 192KB Ram 可供使用。

```

<-----check sram usage----->(read dump.log by ./util/rpt_sram_usage.pl)
Idx Name      Size      VMA      LMA
2 prog_in_sram 0000fd34 02000000 0303b2bc 00050000 2**2
3 .ARM         00000008 0200fd34 0304aff0 0005fd34 2**2
4 .data        00000e94 0200fd40 0304b000 0005fd40 2**3
5 .bss         000076c0 02010be0 0304bea0 00060bd4 2**4
6 boot_in_sram 00001734 02021000 03000190 00009000 2**2
7 .boot_data   00000010 02024000 030018c4 0000c000 2**2
8 .boot_bss    00001540 02024010 030018d4 0000c010 2**2

-----SRAM report-----
sram total -----> 160 KB
sram in use -----> 107 KB
sram free -----> 48 KB

```

编译结果中的 sram free 表示剩余 RAM 资源，全局变量，malloc，Process 的主函数代码都会占用 RAM 资源。

4.3 Ram 函数

为加快函数的运行速度可以将函数定义在 RAM 中运行，定义方式在函数前面增加宏 ATTRIBUTE_SECTION_KEEP_IN_SRAM，如下图：

```

ATTRIBUTE_SECTION_KEEP_IN_SRAM int8_t verify_8bit(uint8_t *ptr,int len)
{
    uint8_t crc;
    uint8_t i,data;
    crc = 0;
    while(len--)
    {
        data = (*ptr)&0xFF;
        crc ^= data;
        for(i = 0;i < 8;i++)
        {
            if(crc & 0x01)
            {
                crc = (crc >> 1) ^ 0x8C;
            }
            else
                crc >>= 1;
        }
        ptr++;
    }
    return (int8_t)crc;
} ? end verify_8bit ?

```

在 RAM 中运行的函数会永久占用 RAM 资源，请只把必要的函数放在 RAM 中。

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+NDBGL=2,0”打开，“AT+NDBGL=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

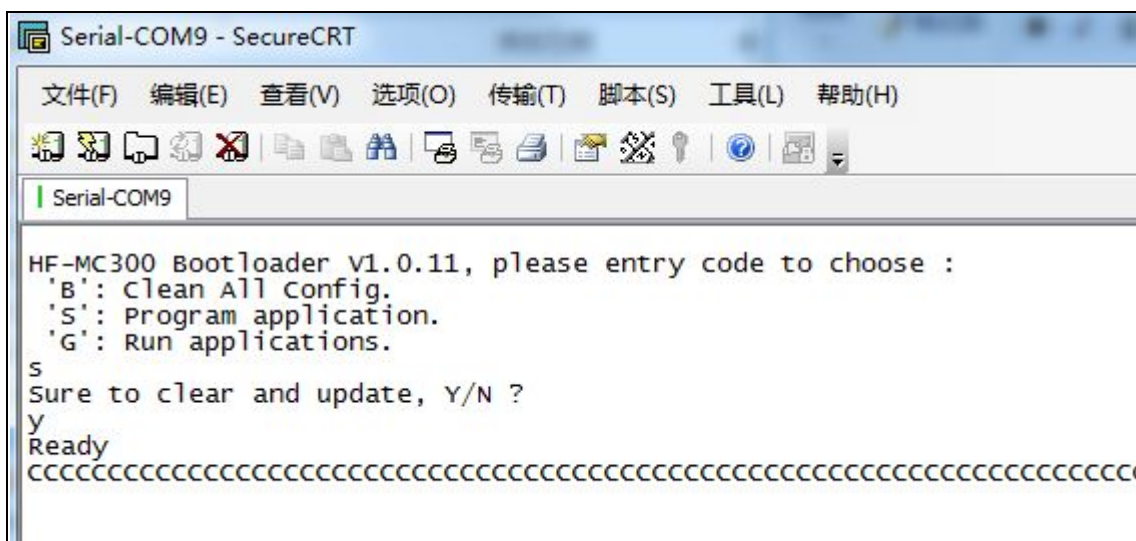
Listen Port 8899
wifi connecting.....
[handle_dhcp] +++
dhcp : init
tx_probe_req +++
[rx_probe_rsp] : probe_response->capability = 0x 11
[rx_probe_rsp] : gCabrioConf.wifi_security = 3
[rx_probe_rsp] : ---
[tx_authentication_req_seq1] : +++
[tx_association_req] : +++
[rx_process_eapol] : +++
[rx_process_wpa]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

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

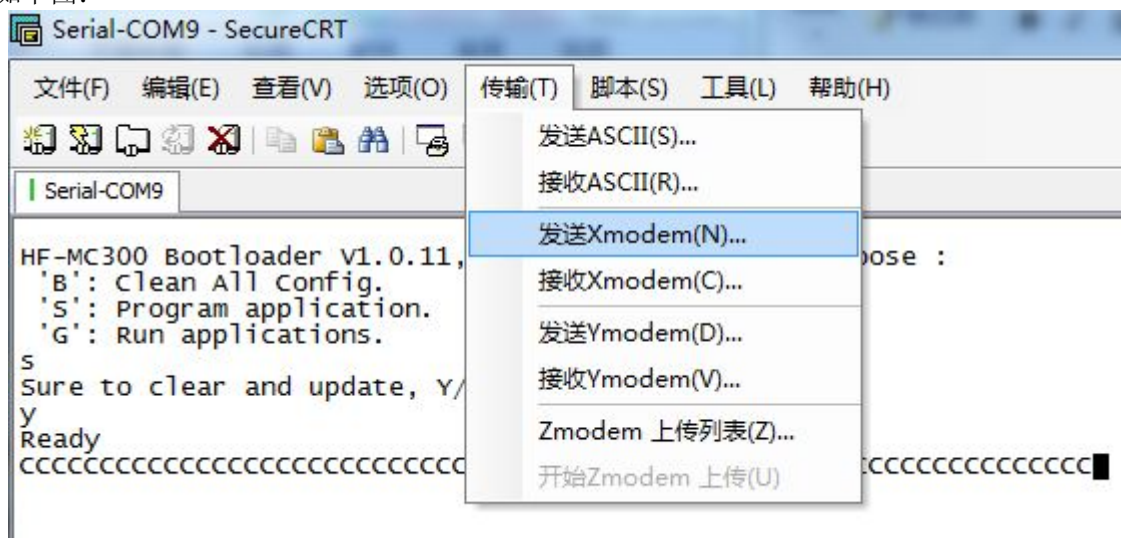
6. 怎样升级程序

5.1 通过串口升级

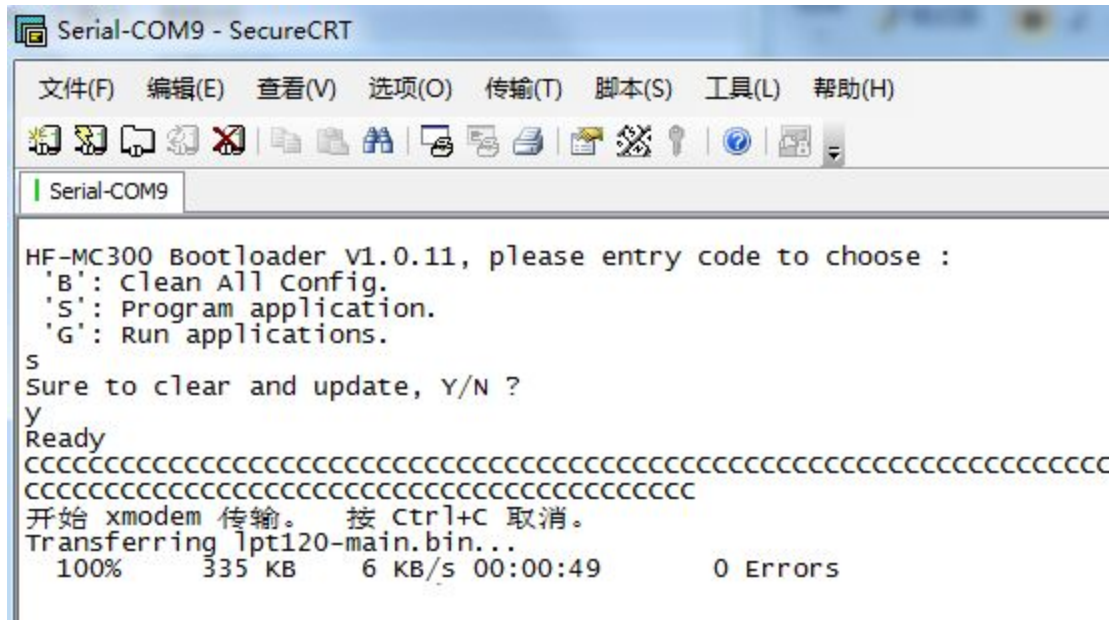
Step 1: 配合 SecureCRT 软件在 230400 波特率下，按住模块的 nReload 按键（拉低）重启模块，1 秒内输入一个空格字符，可以进入模块的 Bootloader 进行升级，成功进入后出现如下界面：



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。

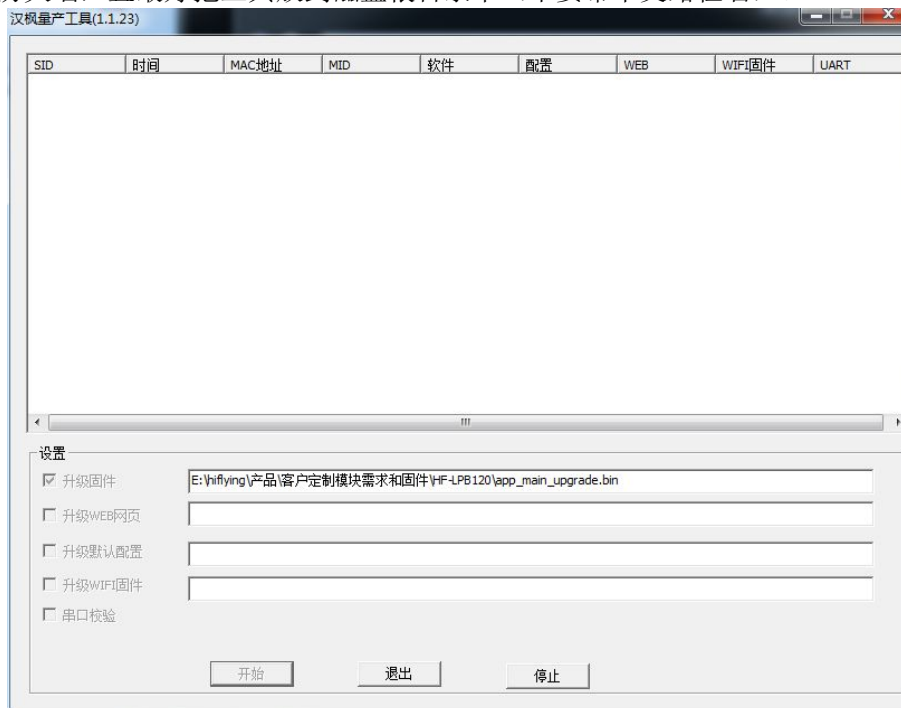
5.2 通过 HF 生产工具批量升级

Step 1: 从汉枫官网下载生产工具。

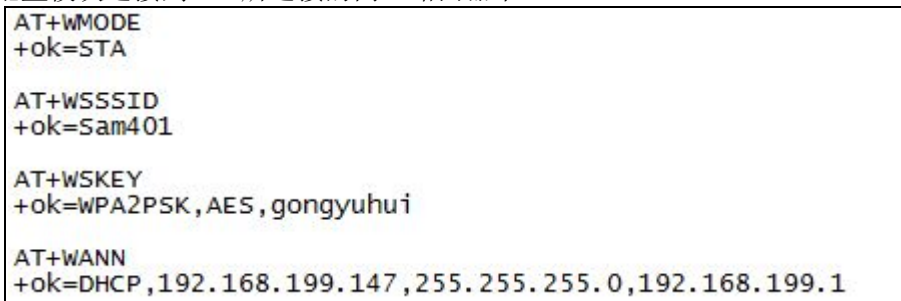
http://gb.hi-flying.com/download_detail_dc/&downloadId=1822d146-343d-4332-af8b-137c0fb4d967.html



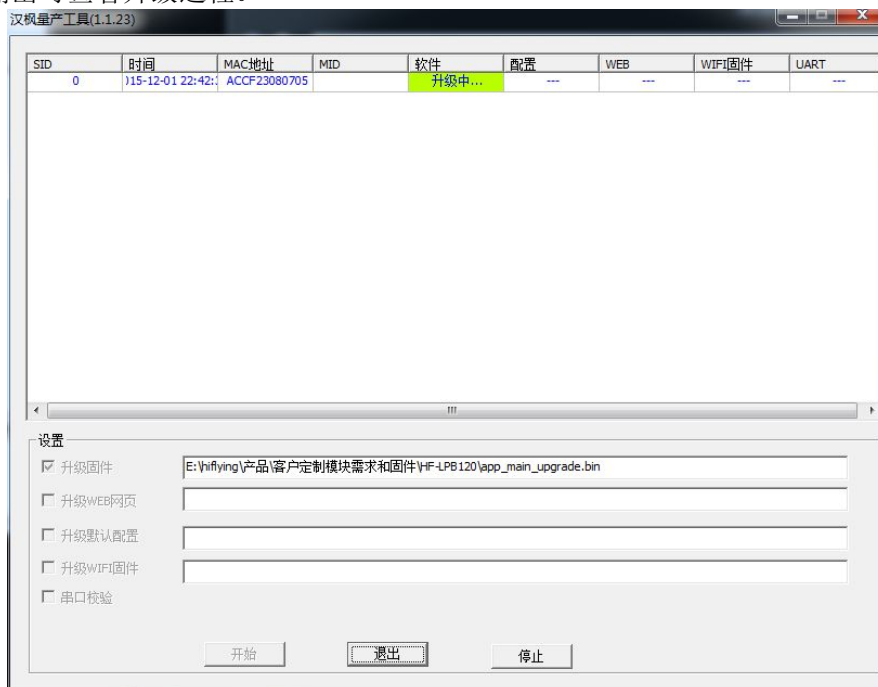
Step 2: PC 连接到路由器，打开工具中的 HFUpdate.exe 工具并加载升级文件 app_main_upgrade.bin，若不能打开，请安装 gtk2-runtime 运行环境，工具使用过程中需要关闭电脑防火墙，且最好把工具放到磁盘根目录下（不要带中文路径名）。



Step 3: 配置模块连接到 PC 所连接的同一路由器下。



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

HSF-MC300 SDK 提供了相应功能的 **example**，可以在 **example** 文件夹下找到相应的代码。可以通过修改根目录下的 **Makefile** 文件选择要编译哪一个例子，修改后直接 **make** 即可编译。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。

修改前：

```
#EXAMPLE_NAME=https
EXAMPLE_NAME ?= null

ifeq ($(EXAMPLE_NAME),null)
APPDIR = src
else
APPDIR = example/$(EXAMPLE_NAME)
endif
```

修改后：

```
EXAMPLE_NAME=at 打开注释，定义需要编译的example文件夹名
EXAMPLE_NAME ?= null

ifeq ($(EXAMPLE_NAME),null)
APPDIR = src
else
APPDIR = example/$(EXAMPLE_NAME)
endif
```

6.1 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，将 AT 命令的数据保存在 Flash 中。

运行结果，可以在串口执行 **AT+TEST** 命令。

编译升级后通过串口工具执行 **AT+TEST** 结果：

```
AT+TEST=ABCD1234
+ok

AT+TEST
+ok=ABCD1234
```

6.2 自定义 GPIO

代码在 **example/gpio/gpiotest.c**，通过这个例子可以了解怎么样自定义 GPIO 脚，修改通用版自带的 PIN 脚功能，熟悉 GPIO API 函数的用法。重定义 GPIO2 和 GPIO15，分别作为输入输出，取反电平。

运行结果，如果 GPIO2 为输入高电平，GPIO15 就输出低电平，如果 GPIO2 为输入低电平，GPIO15 就输出高电平。

6.3 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。

6.4 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。

6.5 任务创建切换

代码在 `example/process/processtest.c`，通过这个例子可以熟悉 process 的创建、切换、任务间的通讯。

执行结果，每 30 秒模块进行一次扫描附近的 AP 信息。

6.6 uFlash 的使用

代码在 `example/uflash/uflashtest.c`，通过这个例子可以熟悉 uFlash 的使用。

执行结果，可以通过串口 AT 命令来读写 uFlash 中的内容。

6.7 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。

6.8 创建 TCP Server

代码在 `example/nettest/tcpservertest.c`，通过这个例子可以熟悉如何创建一个 TCP server。

执行结果，可以通过 TCPUDP 工具创建一个 TCP Client 连接上模块的 28899 端口。

6.9 创建 TCP Client

代码在 `example/nettest/tcpclienttest.`，通过这个例子可以熟悉如何创建一个 TCP client 和 DNS 域名解析。

执行结果，尝试连接百度服务器 “`www.baidu.com:80`”。

6.10 创建 UDP

代码在 `example/nettest/udptest.c`，通过这个例子可以熟悉如何创建一个 UDP 连接。

执行结果，以通过 TCPUDP 工具创建一个 UDP 连接与模块的 38899 端口传输数据。

© 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>