

High-Flying Software Framework (HSF-MC300) User Manual V1.1x

Version 1.1
2016-02

Version List:

Time	Editor	Description	
2015.12.01	Sam	First Version	
2016.01.10	LiuBo	Add Example	
2016.02.15	LiuBo	Modify SDK File Structure	

Content:

1.	SDK Description	4
1.1	LPT120/LPB120/LPT220 Description.....	4
1.2	Version List	4
2.	Compile Environment	5
2.1	Based on Ubuntu 12.04 Linus OS.....	5
2.2	Structure of HSF-MC300 SDK catalogue.....	6
3.	Start compiling	7
3.1	Compile LPT120/LPB120/LPT220	7
3.2	Add source code file.....	7
3.3	Common question during compiling	8
4.	MC300 Resource Distribution.....	10
4.1	2MB Flash resource distribution	10
4.2	Ram resource.....	10
4.3	Ram function	11
5.	Serial port Debug information.....	12
6.	How to update program	13
5.1	By serial port	13
5.2	HF Production Upgrade Tool	15
7.	Examples	17
6.1	Create AT command	17
6.2	Custmized GPIO	17
6.3	Timer control nReady light flashing.....	18
6.4	Serial port callback mechanism control nLink light status	18
6.5	The creation and switch of tasks	18
6.6	The usage of uFlash	18
6.7	Wireless OTA	18
6.8	Create TCP Server.....	19
6.9	Create TCP Client	19
6.10	Create UDP	19

1. SDK Description

1.1 LPT120/LPB120/LPT220 Description

LPT120/LPB120/LPT220 is based on High-Flying SOC MC300 platform Wi-Fi to UART module, it use the same SDK and API, the following MC300 includes the LPB120/LPT120/LPT220 modules, the difference between modules is just the PIN mapping and defination.

1.2 Version List

2016-0215:

Modify SDK File Structure

2016-0110:

Add example;

2015-1126:

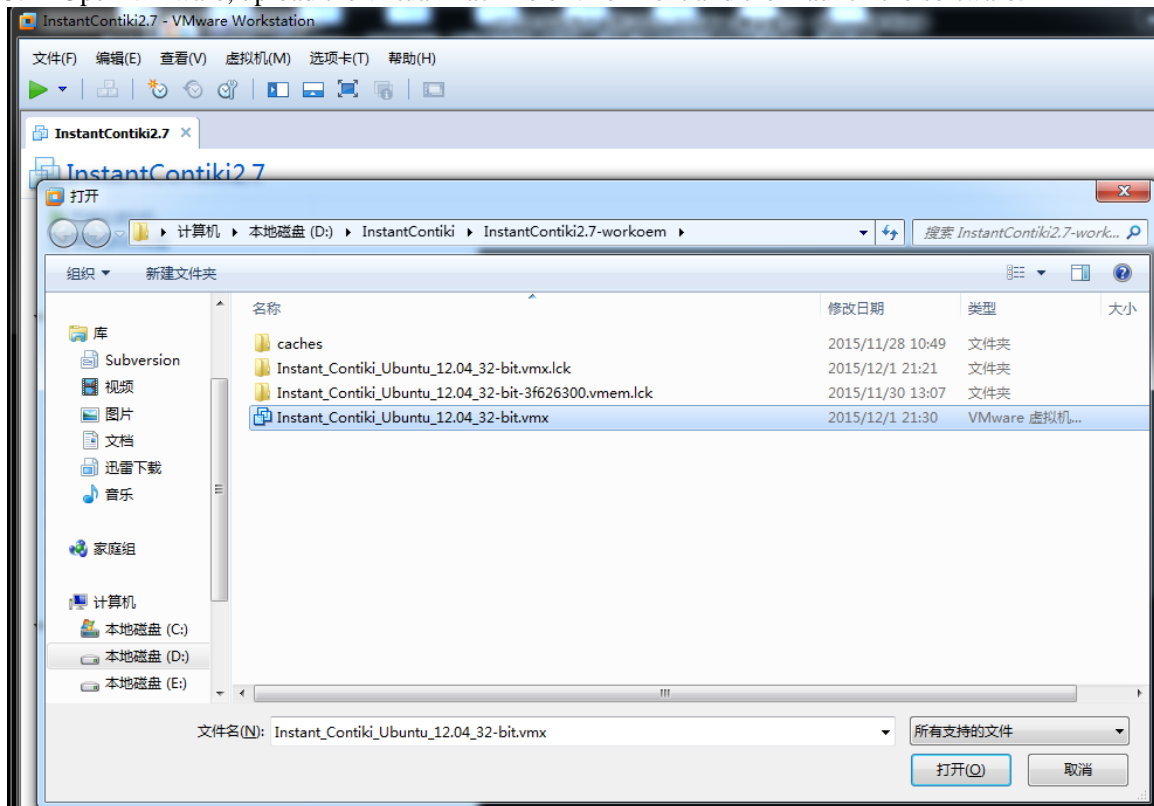
First version

2. Compile Environment

HF-MC300 uses Linux compile environment, so please use our environment provided.

2.1 Based on Ubuntu 12.04 Linus OS

1. Please install VMware 10.0 to avoid compatibility.
Link: <http://pan.baidu.com/s/1nukpk85>
Password: hvtf
2. Download the virtual machine environment according to the following address.
Link: <http://pan.baidu.com/s/1dDkEqVR>
Password: 0shn
3. Open VMware, upload the virtual machine environment and then launch the software.



4. Type the password(password: **user**) and log in(account: **user**).
5. Copy the LPT120 SDK into the system and enter into the following catalogue.
user@iot-work:~/Desktop/LPT120-HSF-20151128/examples/LPT120\$
6. Input **make** to compile.

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

7. Generate **app_main.bin** and **ap_main_upgrade.bin** for updating.

```
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. Type **make clean** to clear all bin files generated before.

```
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 Structure of HSF-MC300 SDK catalogue

	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. Start compiling

3.1 Compile LPT120/LPB120/LPT220

HSF MC300 SDK is suitable for these three kinds of module (LPT120/LPB120/LPT220). The default file compiled by the system is used for module LPT120. If you need compile the firmware of HF-LPB120, please modify the *Makefile* file under the source code folder changing the define `__HF_MODULE_ID__` to the related module. It is shown below:

```
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 Add source code file

The first step is to add .C file. The source file based HSF must include header file `<hsf.h>`. After including this header, the source code can call the API function based on HSF. If the libc interface function is needed, please `#include` the related header file.

To simplify the approach to updating SDK, please DO NOT add too many code into *app_main.c* file. It is better to add a interface function and put all the other source files into the directory itself. Modify Makefile and add source file, macro definition and the definition of header files:

```
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/

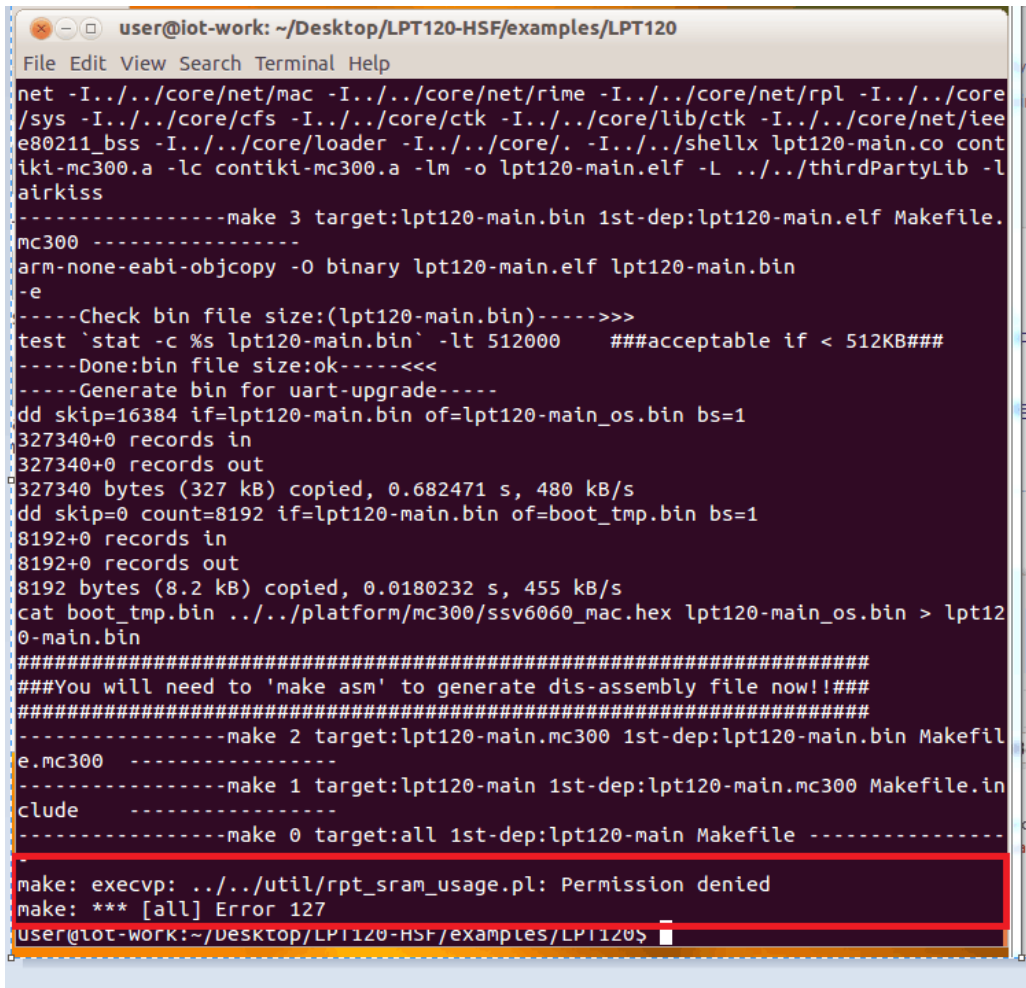
CONTIKI_OBJS=$(addprefix $(OBJECTDIR)/,$(CONTIKI_SRC:.c=.o) $(CONTIKI_SRC:.c=.o))

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

3.3 Common question during compiling

Question one: System hint the following error:

:

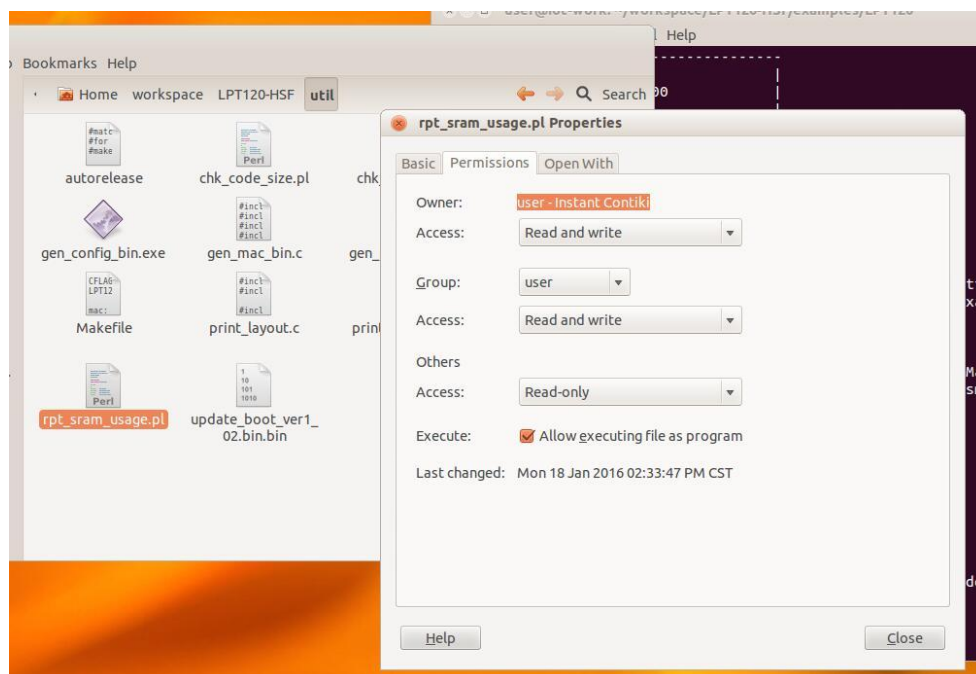


```

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$

```

Solution: modify the attribute of file rpt_sram_usage.pl



4. MC300 Resource Distribution

4.1 2MB Flash resource distribution

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: The maximum excuting code section is 512KB. As a result, the space of the generation file can not exceed this.

Not used: It is extra space for free operation.

HF System Config: Storing section for module's working parameters.

Maker:

File System:

Boot Config:

Upgrade Image: It is backup section for excuting code and it has maximum 512KB for OTA updating.

4.2 Ram resource

Totally 192 KB ram can be provided to use.

```

<-----check sram usage----->(read dump.log by ./util/rpt_sram_usage.pl)
Idx Name          Size      VMA      LMA      00050000  2**2
 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

```

Sram free stands for remaining RAM resource in compiling result. The main function code of global variables, malloc, and process will occupy RAM resource.

4.3 Ram function

To accelerate the executing speed of function, function can be defined and executed in RAM. The method of definition is to add macro `ATTRIBUTE_SECTION_KEEP_IN_SRAM`, it is shown as follow.

```

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 ?

```

The functions executing in the RAM will occupy RAM resource permanently, so please put the necessary functions in the RAM.

5. Serial port Debug information

If the program need print debug information through serial port, HSF provides with two API function, `u_printf` and `HF_Debug` respectively. In default situation, program cannot print any debug information by calling these two functions. That is because default configuration is closed. It needs `hfdbg_set_level(X)` to open serial output, where X represents the debug level. It can also open serial output by AT command, whose 'AT+NDBGL=2,0' and 'AT+NDBGL=0,0' means open and close respectively. The debug information will output from serial port one. It is shown in the following figure:

(Note that: Debug mode MUST be closed when the program is released)

```
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

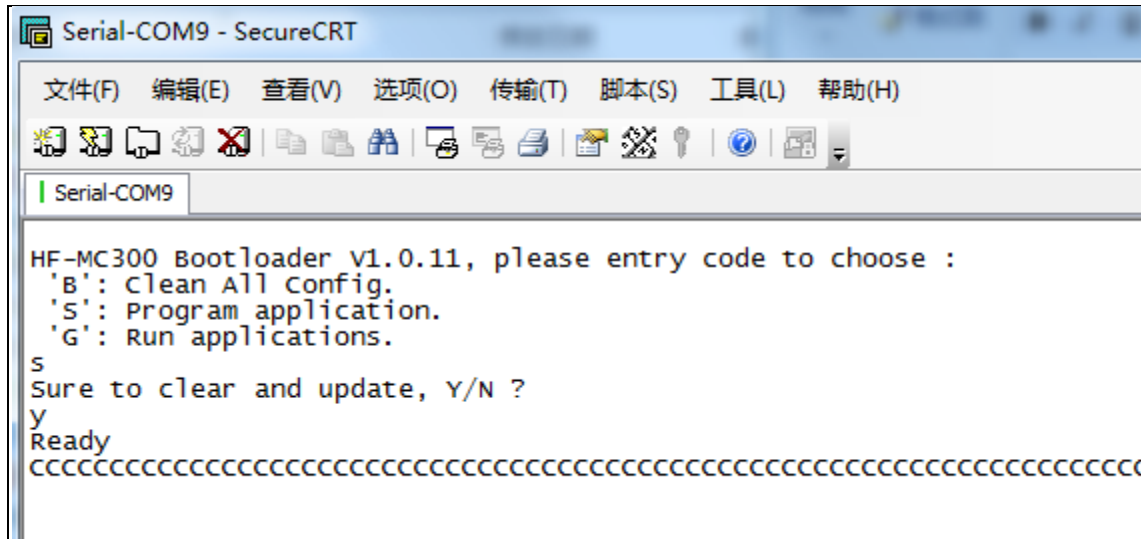
Listen Port 8899
wifi connecting.....
[handle_dhcp] +++
dhcp : init
tx_probe_req +++
[rx_probe_rsp] : probe_response->capability = 0x 11
[rx_probe_rsp] : gCabrioConf.wifi_security = 3
[rx_probe_rsp] : ---
[tx_authentication_req_seq1] : +++
[tx_association_req] : +++
[rx_process_eapol] : +++
[rx_process_wpa]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

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

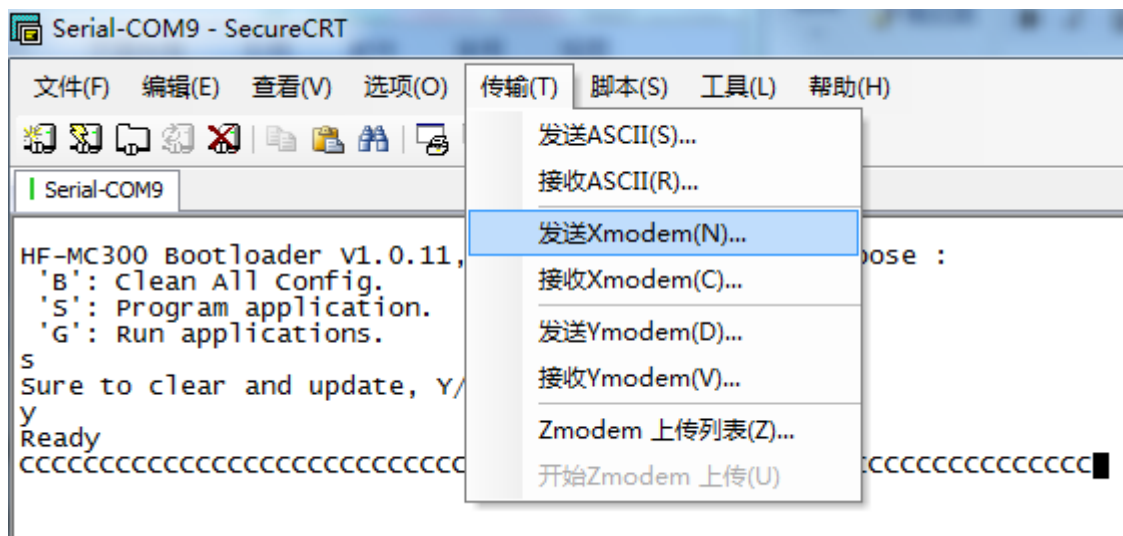
6. How to update program

5.1 By serial port

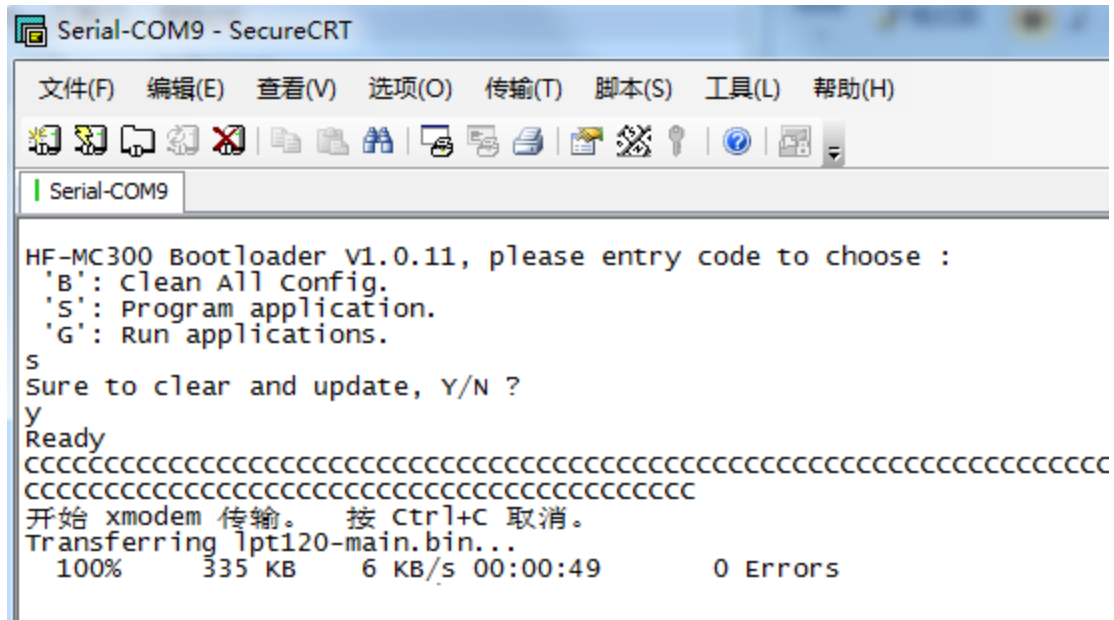
Step 1: Configure the baud rate to 230400 in serial tool SecureCRT. Press on the button nReload(driving down) to reset the module. And the type the *space* character in one second. Module will enter Bootloader to update. The successful screen shot is shown below:



Step 2: Input command 'S' and type 'Y' to update file. Open the Xmode of SecureCRT to transmit the file. It is shown below:



Step 3: Choose the updating file without **upgrades** suffix which is compiled by virtual machine.



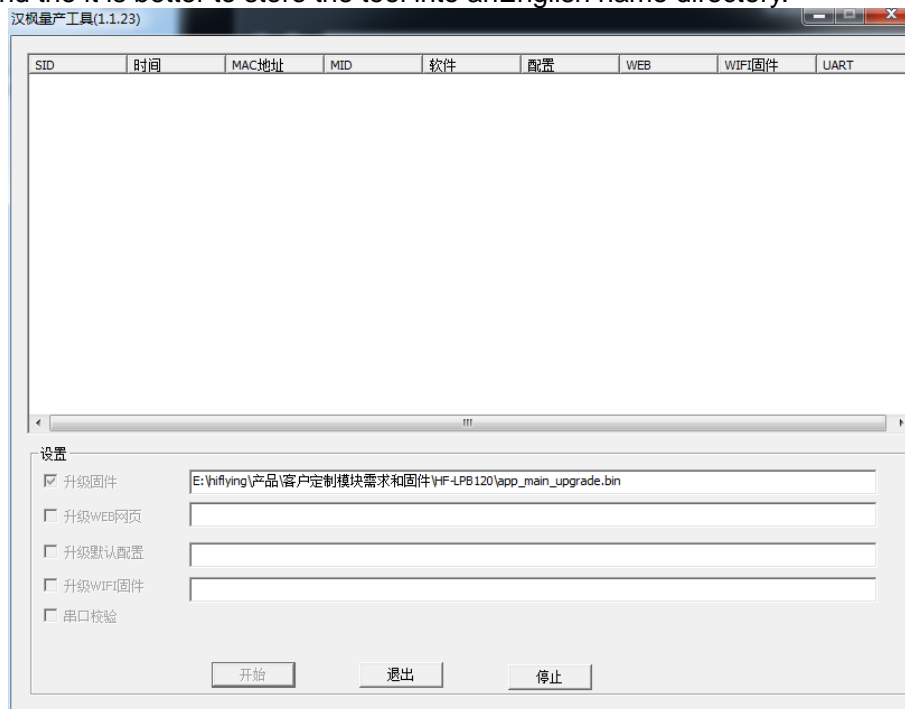
Step 4: Reset the module until the transmission is finished. If the software stuck during updating, please reset the module and enter into the Bootloader again.

5.2 HF Production Upgrade Tool

Step 1: Download the **HF Production Upgrade Tool** from the Hi-flying official website.
http://gb.hi-flying.com/download_detail_dc/&downloadslid=1822d146-343d-4332-af8b-137c0fb4d967.html



Step 2 : Connect the PC with router and open **HFUpdate.exe** to upload the updating files(app_main_upgrade.bin) in the directory. If it cannot be opened, please install executing environment **gtk2-runtime**. During the use of this tool, make sure the PC's firewall has been closed. And the it is better to store the tool into anEnglish name directory.



Step 3 : Configure the module and PC to connect to the same router.

```

AT+WMODE
+ok=STA

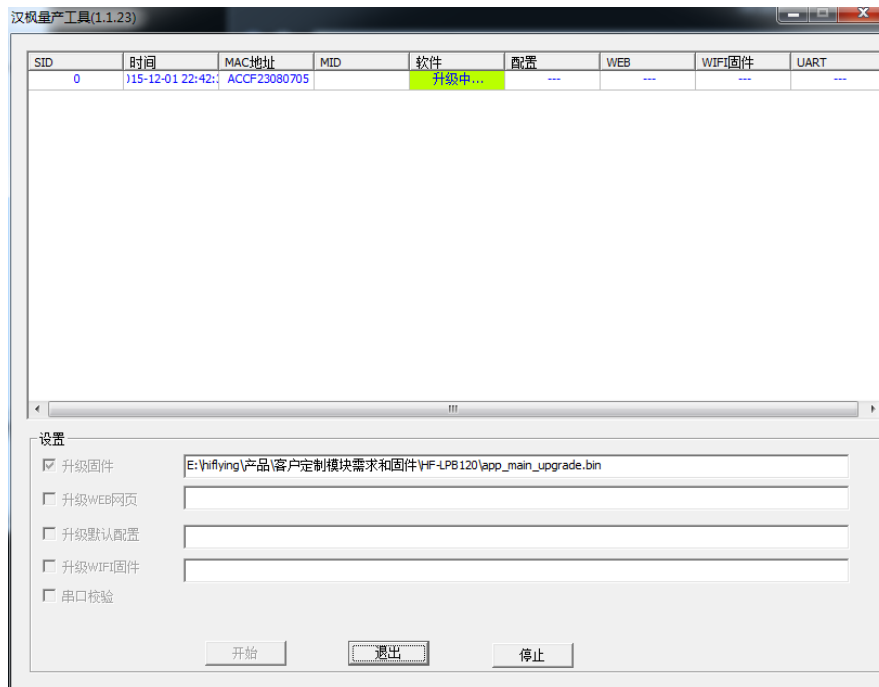
AT+WSSSID
+ok=Sam401

AT+WSKEY
+ok=WPA2PSK,AES,gongyuhui

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

```

Step 4: Input command **AT+OTA** to start updating. PC will show the module's information and wait until it is finished. Open the serial port debug information and check the procedure of updating,



Step 5: After the updating is finished, serial port will print the following information to confirm the finish. And then, you can execute and test the new firmware. PC might hint still updating, please ignore it.

```

image file=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

```

Note that:

The file used for serial update and HF Production Upgrade Tool is different. The default firmware which has *upgradesuffix* should be updated by HF Production Upgrade Tool. *app_main_upgrade.bin* has CRC algorithm based on *app_main*. It is used to avoid updating the wrong file.

7. Examples

HSF-MC300 SDK provides with related examples, you can find the related code in the **example** directory. You are able to choose the particular example through macro definition in **example.h**. After modifying the definition, you can directly compile by command **make**. When the executing example need check the debug information, you can use AT command 'AT+NDBGL=2' to set to level 2. As a result, **u_printf** function can print debug information through serial port.

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 Create AT command

The code in example/at/attest.c, related definition USER_AT_CMD_DEMO, can add customized AT commands. The data of AT commands are stored in FLASH.

For example, the customized AT command 'AT+TEST' has been programmed. After compiling successfully, you can check the result through serial port.

```
AT+TEST=ABCD1234
+ok

AT+TEST
+ok=ABCD1234
```

6.2 Custmized GPIO

The code in example/gpio/gpiotest.c, related definition USER_GPIO_DEMO can show you how to define GPIO ports, modify function of PIN in general version and be familiar with

GPIO API function. For example, redefine GPIO2 and GPIO15 as input and output respectively. The electrical level is opposite. As a result, when the GPIO2 is high level, the GPIO15 will output low level. If the GPIO2 is low, the GPIO15 will be high.

6.3 Timer control nReady light flashing

The code in example/timer/timertest.c, related definition USER_GPIO_DEMO can help you understand the creation of thread, the creation of timer and the usage of related API functions.

The result shows the nReady light is flashing by 1HZ frequency.

6.4 Serial port callback mechanism control nLink light status

The code in example/netcallback/callbacktest.c, related definition USER_CALLBACK_DEMO can help you to be familiar with how the serial port transmit and call back the API.

Results: when serial port transmit 'GPIO NLINK LOW' to module, the nLink light stay low. When serial port transmit 'GPIO NLINK HIGH' to module, the nLink light stay high. When serial port transmit 'GPIO NLINK FLASH' to module, the nReady light is flashing by 1HZ frequency.

6.5 The creation and switch of tasks

The code in example/process/processtest.c, related definition USER_PROCESS_DEMO can help you to be familiar with the creation, switch and communication between tasks of **process**.

Results: in every 30 seconds, the module scan AP information nearby.

6.6 The usage of uFlash

The code in example/uflash/uflashtest.c, related definition USER_FLASH_DEMO can help you to be familiar with the usage of uFlash.

Results: module can read and write the content of uFlash by serial port AT command.

6.7 Wireless OTA

The code in example/update/updatetest.c, related definition USER_UPDATE_DEMO can help you be familiar with the related API in wireless OTA.

Results: serial port use command 'AT+UPGRADESW=http://192.168.1.1/update.bin' to update wirelessly.

6.8 Create TCP Server

The code in `example/nettest/tcpservertest.c`, related definition `USER_NET_TCPSERVER_DEMO` can help you be familiar with how to create a TCP server.

Results: you can generate a TCP client in TCPUDP tool connected with 28899 port in the module

6.9 Create TCP Client

The code in `example/nettest/tcpclienttest.c`, related definition `USER_NET_TCPCLIENT_DEMO` can help you be familiar with how to create a TCP server and DNS.

Results: try to connect to Baidu server 'www.baidu.com:80'.

6.10 Create UDP

The code in `example/nettest/udpctest.c`, related definition `USER_NET_UDP_DEMO` can help you be familiar with how to create a UDP connection.

Results: you can transmit data by TCPUDP tool when a UDP connection to 38899 port of module has been created.

© 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>