High-Flying Software Framework (HSF)

User Manual

V1.1x

Version 1.1x

2013.11

Update History：

Update Time	Editor	Content	Note
2013.08.28	Jim	HSF-V1.0	
2013.09.16	Jim	HSF-V1.03, Add example code	
2013.10.17	Jim	HSF-V1.1x, Add HF-LPB100	
2013.11.26	Jim	Add software watchdog Add PWM interface	

Catalog

1.   SDK History

1.1.   HSF-V1.00 Version

HSF SDK Version 1.00 is released, support HF-LPB.

1.2.   HSF-V1.03 Version

Add UART API.

Add Timer API.

Add SOCKETA/B Callback Function.

Add Example Code.

Add SOCKETA,SOCKETB API.

Add DEMO Project.

Add Callback Function of SOCKET TCP Connection and Disconnection time.

1.3.   HSF-V1.13 Version

Add support to HF-LPB100.

Add edge trigger interrupt.

Add file operation API (Save user parameters) API.

Fix some bug about GPIO interrupt.

Fix some bug of protocol stack.

Fix some bug of Wi-Fi driver.

1.4.   HSF-V1.17 Version

Add Watch Dog Thread.

Add SmartLink.

Add WPS.

Modify timer period interface.

Add NVM interface

Add Mass Production Upgrade function

Support HF-LPB100 2M Flash SDK, code size is up to 512KB

Add flash interface for 128KB size.

Support TCP keepalive.

Fix UNLINK2 download failure problem

Support upgrade from 1MB SDK (1.03a)
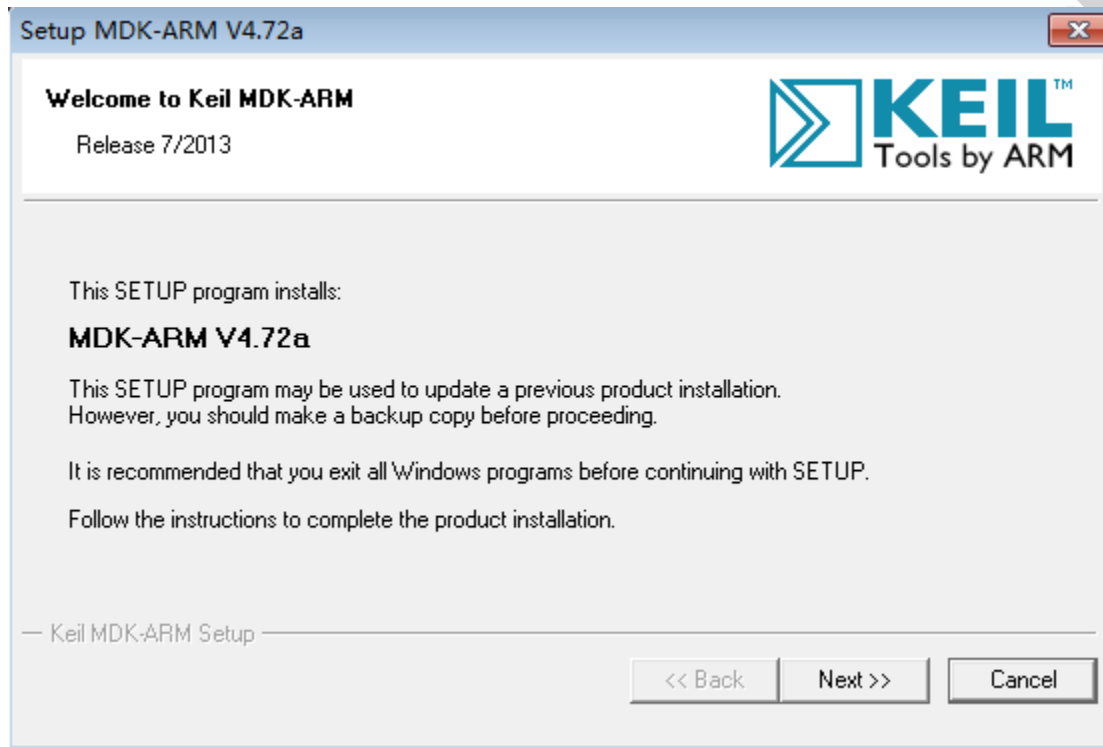
2.　Intall Compiler Environment

LPB100 is based on Keil MDK.

2.1 Keil MDK Environment(HF-LPB100)

LPB100 is based on Keil MDK.

1. Install Keil MDK, the SDK use Keil MDK-ARM version V4.72a to compile. Better to use this or higher version .

2. The SDK directory is as following.
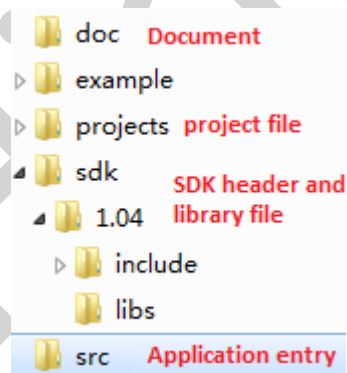
3. Copy "OTP+FLASH.HF-48MRC.O18B.FLM" in "doc" directory to "Keil MDK\installPah\ARM\Flash\". If this path doesn't exist, search the Flash directory and copy the file to this directory.

4.   Open "LPBS2W.uvproj" in the "projects" directory.
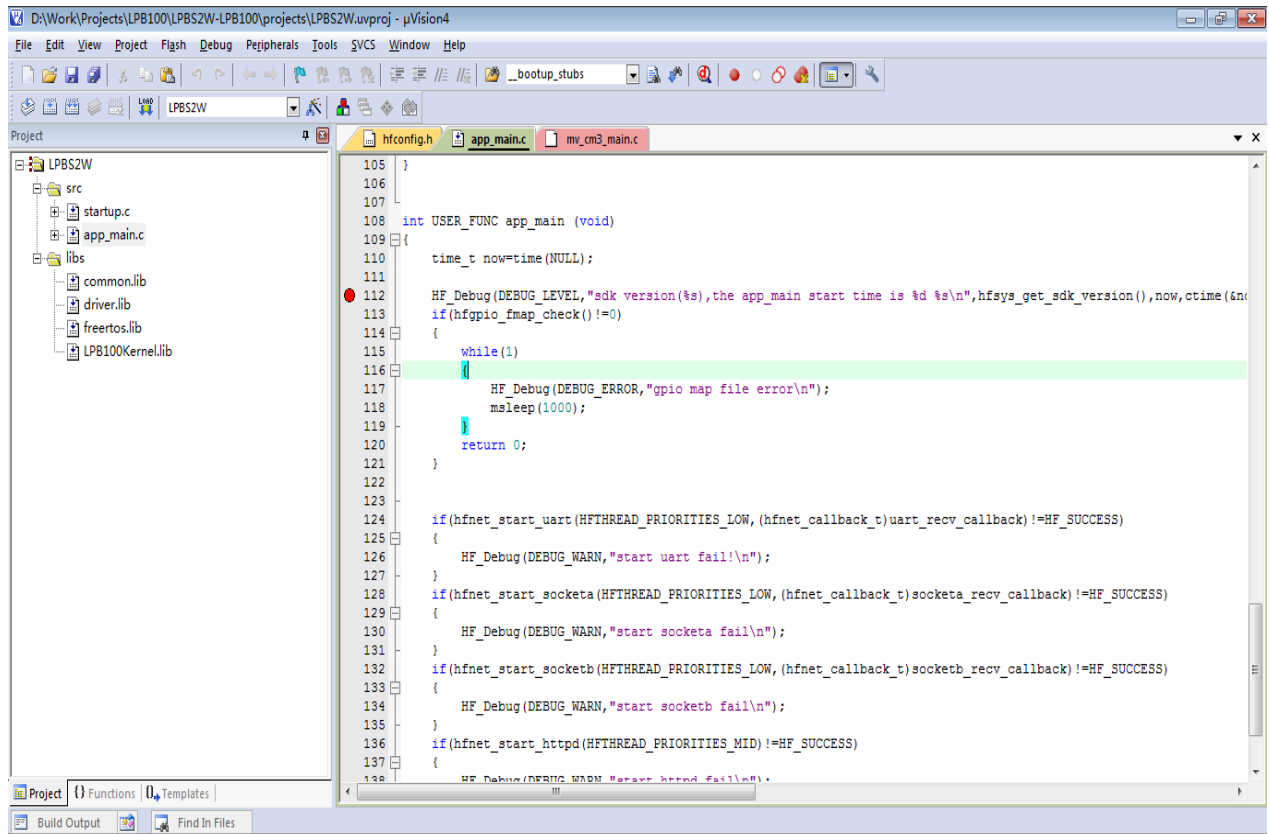
Utilities set option
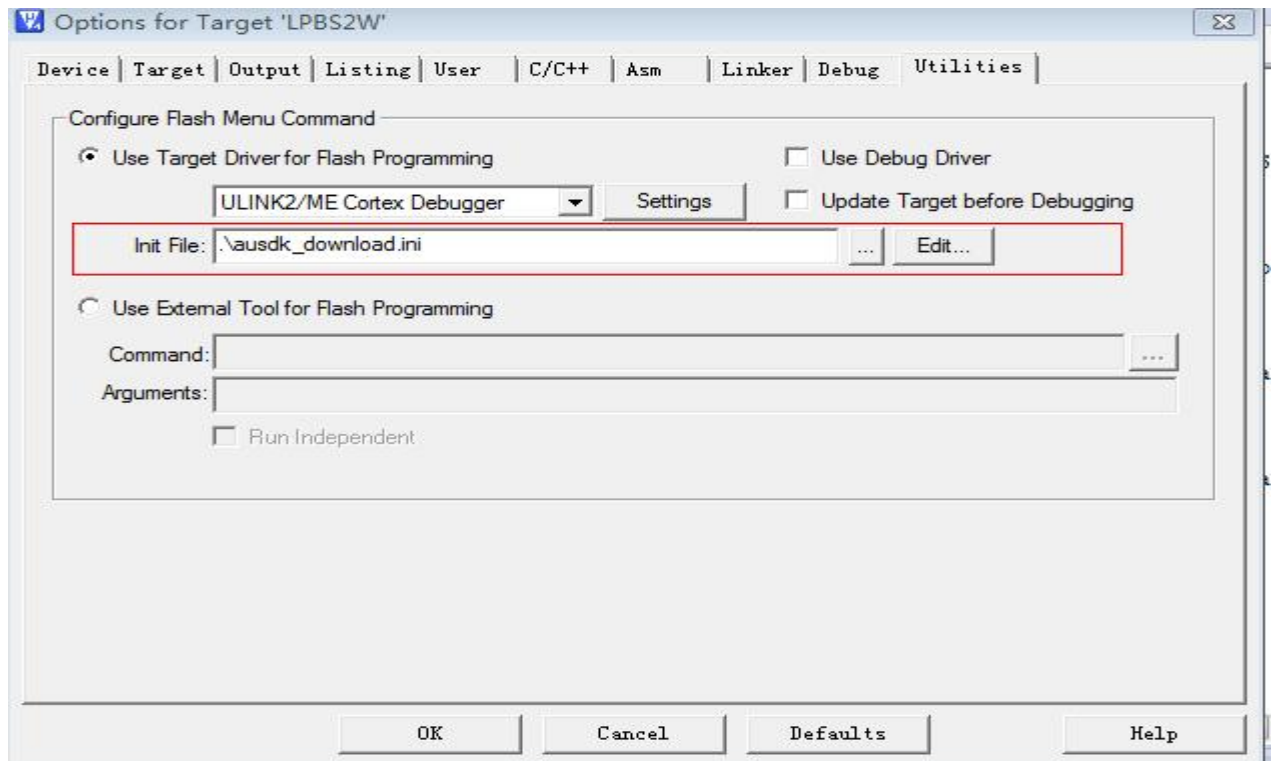
Debug setting
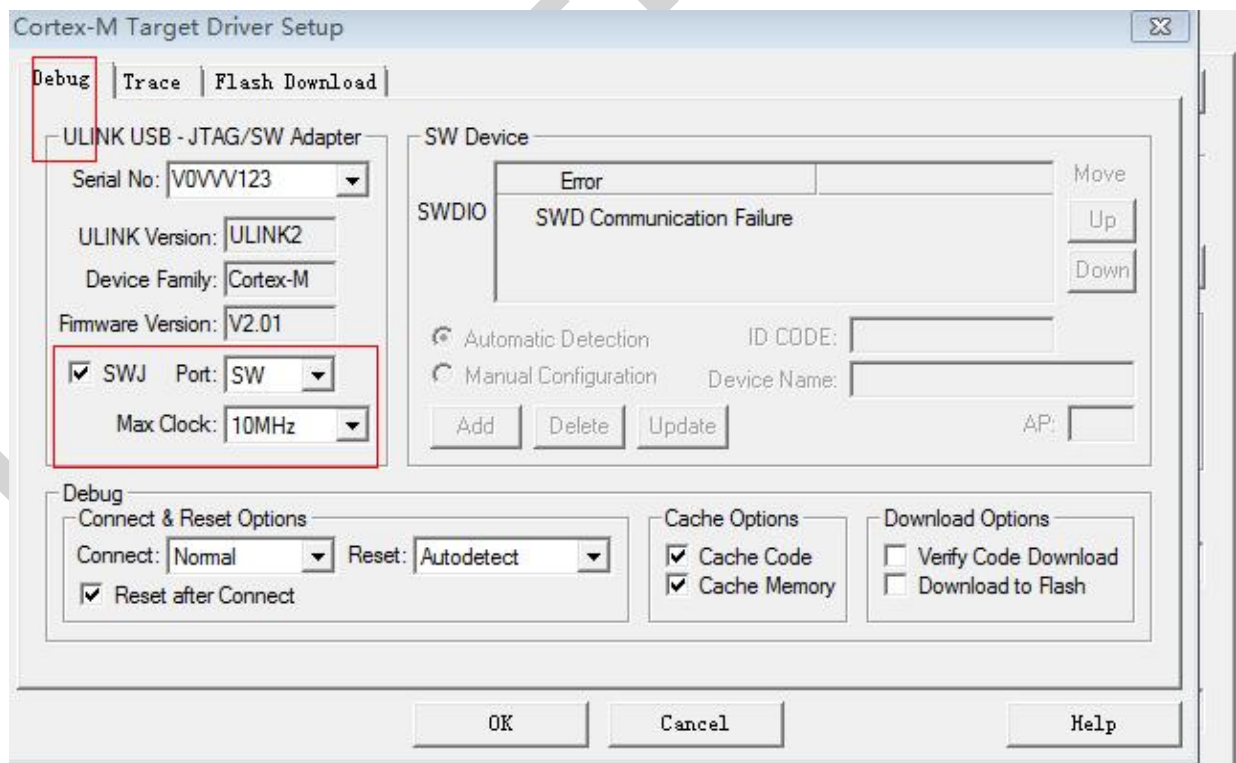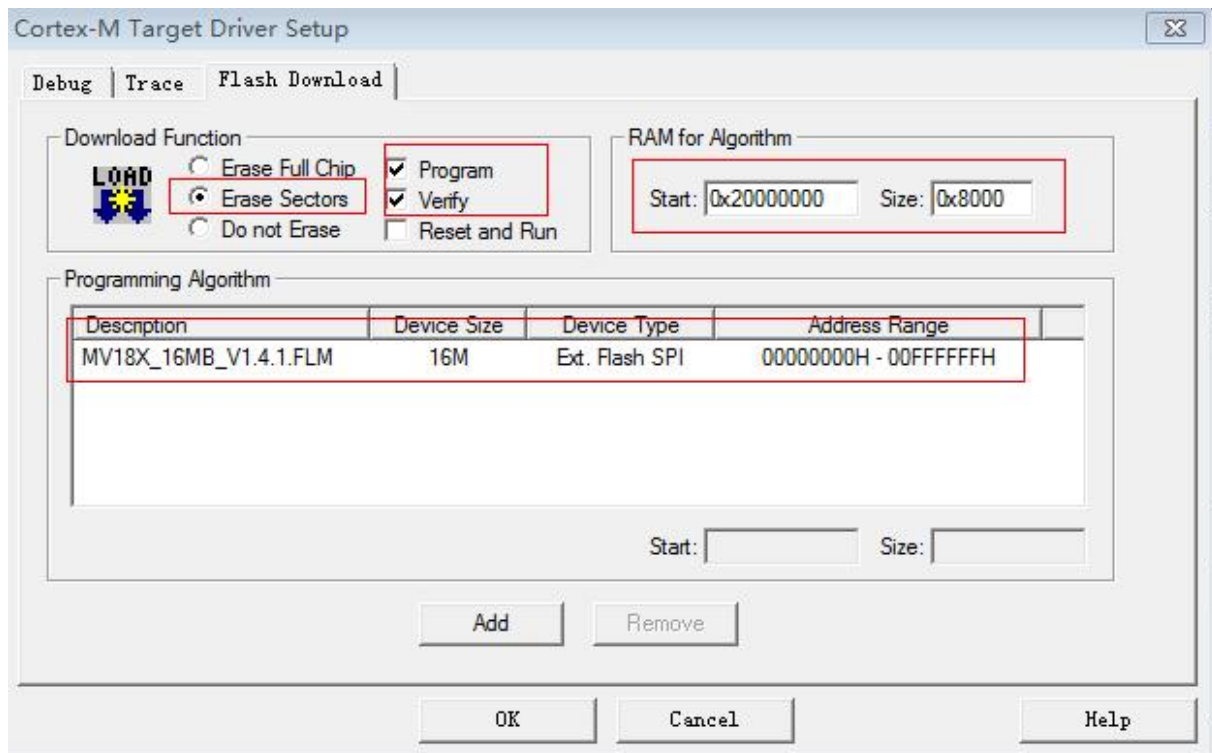
5. Click "build" to create executable binary file

3.   Resource Allocation

3.1.   LPB100 Resource Allocation

LPB100 1MB SDK code area is 400kb, LPB200 2MB SDK code area is 512 KB. If do you apply SDK own system, only 1wip and wifi. LPB100 has at most free 26kb RAM and 100-200 KB ROM.

LPB100,LPT100 flash is 2MB, can apply 2MB SDK, LPT200 is 1MB flash ,can apply 1MB SDK.

4. Add User Source Code

4.1. User Function definition

[Return Type] + USER_FUNC + [Function Name] + [Parameter]

Example:

void USER_FUNC test_func1(char *a);

"USER_FUNC" is function decoration symbol, it is for better compatibility. The program compiled in some platform may not work if don't add "USER_FUNC".

4.2. Add User Source Code File

Any C source file must include "hsf.h" header file to call standard Socket function or other APIs in HSF. Include other related headers file to use API function in "libc". For example, include "<stringh.h>" to use string operation function, include "<time.h>" to use timer function.

Don't add too much code in "app_main.c" file for SDK update. It's better to add only an entry function in "app_main.c" and create a user directory in "src" directory to put user source file in this directory. If the SDK is update, just modify the "app_main.c" and copy the user directory to finish update.

5.   User Defined GPIO

5.1.   PIN Definition

HSF use First Grade Mapping Table, mapping module's pin to Function Code. Users may define each pin themselves(Except some pin that can only be used for peripheral interface) . See "hfgpio.h" for more deails.

The current pin attribute is as follows.

F_GPI: GPIO Input
F_GPO: GPIO Output
F_GPIO: GPIO Input or Output
F_GND: GND
F_VCC: VCC
F_NC: No Connection.
F_RST: Hardware Reset
F_IT: Interrupt
F_PI: Perpheral Interface Pin, for   example, MOSI of SPI Interface
F_PWM: PWM
F_ADC: ADC

Users may call "HF_M_PINNO(_pinno)" to read each pin attribute. "_pinno" is the pin number. See user manual or "hfgpio.h" for more reference.

	HF-LPB100	HF-LPT100	HF-LPT200
1	GND	1	16
2	SWCLK		
3	N.C		
4	N.C		
5	SWD		
6	N.C		
7	GPIO7		14
8	GPIO8		
9	DVDD	2	15
10	N.C		
11	GPIO11	10	
12	GPIO12	9	
13	GPIO13		
14	N.C		
15	GPIO15 (WPS)		
16	N.C		
17	GND		
18	GPIO18	8	
19	N.C		
20	GPIO20		

21	N.C		
22	N.C		
23	GPIO23		9
24	N.C		
25	PWR_SW	7	
26	N.C		
27	SPI_MISO		3
28	SPI_CLK		2
29	SPI_CS		4
30	SPI_MOSI		1
31	DVDD		
32	GND		
33	N.C		
34	DVDD		
35	N.C		
36	N.C		
37	N.C		
38	N.C		
39	UART0_TX	6	5
40	UART0_RTS		8
41	UART0_RX	5	6
42	UART0_CTS		7
43	nLink		13
44	nReady		11
45	nReload	3	12
46	N.C		
47	EXT_RESETn	4	10
48	GND		

5.2 System Defined Function Code

The Function Code defined in HSF is as follows:

Fixed Function Code is as follows:
HFGPIO_F_JTAG_TCK
HFGPIO_F_JTAG_TDO,
HFGPIO_F_JTAG_TDI
HFGPIO_F_JTAG_TMS
HFGPIO_F_USBDP

```
HFGPIO_F_USBDM
HFGPIO_F_UART0_TX
HFGPIO_F_UART0_RTS
HFGPIO_F_UART0_RX
HFGPIO_F_UART0_CTS
HFGPIO_F_SPI_MISO
HFGPIO_F_SPI_CLK
HFGPIO_F_SPI_CS
HFGPIO_F_SPI_MOSI
HFGPIO_F_UART1_TX
HFGPIO_F_UART1_RTS
HFGPIO_F_UART1_RX
HFGPIO_F_UART1_CTS
```

The above Function Code can only be used for corresponding pin. Take HF-LPB00   for example, "HFGPIO_F_JTAG_TCK" can only be used for "HF_M_PIN(2)", pin number 2 . If the corresponding pin can be used for GPIO, users may set this Function Code as "HFM_NOPIN", then this pin can be configured as GPIO. Take "HFGPIO_F_UART0_TX" for example, if the UART is not used, users may set the corresponding pin as "HFM_NOPIN", then this pin can be   configured as user defined function.

The following Function Code can be configured to pins with "F_GPIO" pin attribute or "HFM_NOPIN". This function will not be used if it is configured as "HFM_NOPIN".

```
/////////////////////////////////
HFGPIO_F_NLINK
HFGPIO_F_NREADY
HFGPIO_F_NRELOAD
HFGPIO_F_SLEEP_RQ// the function is not available yet
HFGPIO_F_SLEEP_ON// the function is not available yet
```

5.3 Modify Mapping Table

The mapping Table is defined in "app_main.c".

```
const int hf_gpio_fid_to_pid_map_table[HFM_MAX_FUNC_CODE]=
{
    HF_M_PIN(2), //HFGPIO_F_JTAG_TCK
    HF_M_PIN(3), //HFGPIO_F_JTAG_TDO
    HF_M_PIN(4), //HFGPIO_F_JTAG_TDI
    HF_M_PIN(5), //HFGPIO_F_JTAG_TMS
    HFM_NOPIN,   //HFGPIO_F_USBDP
    HFM_NOPIN,   //HFGPIO_F_USBDM
    HF_M_PIN(39),     //HFGPIO_F_UART0_TX
    HF_M_PIN(40),     //HFGPIO_F_UART0_RTS
    HF_M_PIN(41),     //HFGPIO_F_UART0_RX
    HF_M_PIN(42),     //HFGPIO_F_UART0_CTS

    HF_M_PIN(27),     //HFGPIO_F_SPI_MISO
    HF_M_PIN(28),     //HFGPIO_F_SPI_CLK
    HF_M_PIN(29),     //HFGPIO_F_SPI_CS
    HF_M_PIN(30),     //HFGPIO_F_SPI_MOSI

    HFM_NOPIN,   //HFGPIO_F_UART1_TX,
    HFM_NOPIN,   //HFGPIO_F_UART1_RTS,
```

```
        HFM_NOPIN,   //HFGPIO_F_UART1_RX,
        HFM_NOPIN,   //HFGPIO_F_UART1_CTS,

        HF_M_PIN(43),      //HFGPIO_F_NLINK
        HF_M_PIN(44),      //HFGPIO_F_NREADY
        HF_M_PIN(45),      //HFGPIO_F_NRELOAD
        HF_M_PIN(7), //HFGPIO_F_SLEEP_RQ
        HF_M_PIN(8), //HFGPIO_F_SLEEP_ON

        HFM_NOPIN,         //HFGPIO_F_RESERVE0
        HFM_NOPIN,         //HFGPIO_F_RESERVE1
        HFM_NOPIN,         //HFGPIO_F_RESERVE2
        HFM_NOPIN,         //HFGPIO_F_RESERVE3
        HFM_NOPIN,         //HFGPIO_F_RESERVE4
        HFM_NOPIN,         //HFGPIO_F_RESERVE5

        HFM_NOPIN,   //
        HFM_NOPIN,   //
};
```

Function Code is the index number of "hf_gpio_fid_to_pid_map_table". The value of array is the pin number and it is mapping to the corresponding function. For example, the function of HF-LPB 43 pin is "HFGPIO_F_NLINK".

If   users define Function Code themselves, don't be identical to   the system defined Function Code   and start from "HFGPIO_F_USER_DEFINE".

Example 1：

Take HF-LPB for instance, we use one pin to enter "AT Command Mode". It is defined as：

```
        #define USERGPIO_F_ATCMD_MODE    (HFGPIO_F_USER_DEFINE+0)
```

The corresponding mapping table   is:

```
const int hf_gpio_fid_to_pid_map_table[HFM_MAX_FUNC_CODE]=
{
        HF_M_PIN(2),       //HFGPIO_F_JTAG_TCK
        HF_M_PIN(3),       //HFGPIO_F_JTAG_TDO
        HF_M_PIN(4),       //HFGPIO_F_JTAG_TDI
        HF_M_PIN(5),       //HFGPIO_F_JTAG_TMS
        HFM_NOPIN,         //HFGPIO_F_USBDP
        HFM_NOPIN,         //HFGPIO_F_USBDM
        HF_M_PIN(39),         //HFGPIO_F_UART0_TX
        HF_M_PIN(40),         //HFGPIO_F_UART0_RTS
        HF_M_PIN(41),         //HFGPIO_F_UART0_RX
        HF_M_PIN(42),         //HFGPIO_F_UART0_CTS

        HF_M_PIN(27),         //HFGPIO_F_SPI_MISO
        HF_M_PIN(28),         //HFGPIO_F_SPI_CLK
        HF_M_PIN(29),         //HFGPIO_F_SPI_CS
        HF_M_PIN(30),         //HFGPIO_F_SPI_MOSI

        HFM_NOPIN,      //HFGPIO_F_UART1_TX,
        HFM_NOPIN,      //HFGPIO_F_UART1_RTS,
        HFM_NOPIN,      //HFGPIO_F_UART1_RX,
        HFM_NOPIN,      //HFGPIO_F_UART1_CTS,
```

```
        HF_M_PIN(43),          //HFGPIO_F_NLINK
        HF_M_PIN(44),          //HFGPIO_F_NREADY
        HF_M_PIN(45),          //HFGPIO_F_NRELOAD
        HF_M_PIN(7),       //HFGPIO_F_SLEEP_RQ
        HF_M_PIN(8),       //HFGPIO_F_SLEEP_ON

        HFM_NOPIN,        //HFGPIO_F_RESERVE0
        HFM_NOPIN,        //HFGPIO_F_RESERVE1
        HFM_NOPIN,        //HFGPIO_F_RESERVE2
        HFM_NOPIN,        //HFGPIO_F_RESERVE3
        HFM_NOPIN,        //HFGPIO_F_RESERVE4
        HFM_NOPIN,        //HFGPIO_F_RESERVE5

        HF_M_PIN(16),      // USERGPIO_F_ATCMD_MODE,   HF-LPB00 16 pin can be used for interrupt
        HFM_NOPIN,   //
};

void hfgpio_interrupt_func(uint32_t,uint32_t)
 {
        if(hfgpio_fpin_is_high(USERGPIO_F_ATCMD_MODE))
        {
                hfsys_switch_run_mode(HFSYS_STATE_RUN_CMD);
}
else
{
    hfsys_switch_run_mode(HFSYS_STATE_RUN_THROUGH);
}
 }


int USER_FUNC app_main (void)
 {
      //Config "USERGPIO_F_ATCMD_MODE" pin as rising edge triggered interrupt.
      if(hfgpio_configure_fpin_interrupt(USERGPIO_F_ATCMD_MODE, HFPIO_IT_RISE_EDGE,
hfgpio_interrupt_func,1)!=HF_SUCCESS)
        {
             return -1;
}

 }
```

Example 2：

Switch "nLink" and "nReady" function of the EVK.

```
const int hf_gpio_fid_to_pid_map_table[HFM_MAX_FUNC_CODE]=
{
        HF_M_PIN(2),           //HFGPIO_F_JTAG_TCK
        HF_M_PIN(3),           //HFGPIO_F_JTAG_TDO
        HF_M_PIN(4),           //HFGPIO_F_JTAG_TDI
        HF_M_PIN(5),           //HFGPIO_F_JTAG_TMS
        HFM_NOPIN,             //HFGPIO_F_USBDP
        HFM_NOPIN,             //HFGPIO_F_USBDM
        HF_M_PIN(39),   //HFGPIO_F_UART0_TX
        HF_M_PIN(40),   //HFGPIO_F_UART0_RTS
        HF_M_PIN(41),   //HFGPIO_F_UART0_RX
        HF_M_PIN(42),   //HFGPIO_F_UART0_CTS
```

```
    HF_M_PIN(27),    //HFGPIO_F_SPI_MISO
    HF_M_PIN(28),    //HFGPIO_F_SPI_CLK
    HF_M_PIN(29),    //HFGPIO_F_SPI_CS
    HF_M_PIN(30),    //HFGPIO_F_SPI_MOSI

    HFM_NOPIN,           //HFGPIO_F_UART1_TX,
    HFM_NOPIN,           //HFGPIO_F_UART1_RTS,
    HFM_NOPIN,           //HFGPIO_F_UART1_RX,
    HFM_NOPIN,           //HFGPIO_F_UART1_CTS,

    HF_M_PIN(44),    //HFGPIO_F_NLINK
    HF_M_PIN(43),    //HFGPIO_F_NREADY
    HF_M_PIN(45),    //HFGPIO_F_NRELOAD
    HF_M_PIN(7),         //HFGPIO_F_SLEEP_RQ
    HF_M_PIN(8),         //HFGPIO_F_SLEEP_ON

    HFM_NOPIN,           //HFGPIO_F_RESERVE0
    HFM_NOPIN,           //HFGPIO_F_RESERVE1
    HFM_NOPIN,           //HFGPIO_F_RESERVE2
    HFM_NOPIN,           //HFGPIO_F_RESERVE3
    HFM_NOPIN,           //HFGPIO_F_RESERVE4
    HFM_NOPIN,           //HFGPIO_F_RESERVE5

    HF_M_PIN(16),    // USERGPIO_F_ATCMD_MODE, HF-LPB00 16 pin can be used for
interrupt

    HFM_NOPIN,       //
};
```

Example 3 :

Modify the "nLink" and "nReady" default function.
```
const int hf_gpio_fid_to_pid_map_table[HFM_MAX_FUNC_CODE]=
{
    HF_M_PIN(2),          //HFGPIO_F_JTAG_TCK
    HF_M_PIN(3),          //HFGPIO_F_JTAG_TDO
    HF_M_PIN(4),          //HFGPIO_F_JTAG_TDI
    HF_M_PIN(5),          //HFGPIO_F_JTAG_TMS
    HFM_NOPIN,           //HFGPIO_F_USBDP
    HFM_NOPIN,           //HFGPIO_F_USBDM
    HF_M_PIN(39),    //HFGPIO_F_UART0_TX
    HF_M_PIN(40),    //HFGPIO_F_UART0_RTS
    HF_M_PIN(41),    //HFGPIO_F_UART0_RX
    HF_M_PIN(42),    //HFGPIO_F_UART0_CTS

    HF_M_PIN(27),    //HFGPIO_F_SPI_MISO
    HF_M_PIN(28),    //HFGPIO_F_SPI_CLK
    HF_M_PIN(29),    //HFGPIO_F_SPI_CS
    HF_M_PIN(30),    //HFGPIO_F_SPI_MOSI

    HFM_NOPIN,           //HFGPIO_F_UART1_TX,
```

```
    HFM_NOPIN,          //HFGPIO_F_UART1_RTS,
    HFM_NOPIN,          //HFGPIO_F_UART1_RX,
    HFM_NOPIN,          //HFGPIO_F_UART1_CTS,

    HFM_NOPIN,          //HFGPIO_F_NLINK
    HFM_NOPIN,          //HFGPIO_F_NREADY
    HF_M_PIN(45),   //HFGPIO_F_NRELOAD
    HF_M_PIN(7),        //HFGPIO_F_SLEEP_RQ
    HF_M_PIN(8),        //HFGPIO_F_SLEEP_ON

    HFM_NOPIN,          //HFGPIO_F_RESERVE0
    HFM_NOPIN,          //HFGPIO_F_RESERVE1
    HFM_NOPIN,          //HFGPIO_F_RESERVE2
    HFM_NOPIN,          //HFGPIO_F_RESERVE3
    HFM_NOPIN,          //HFGPIO_F_RESERVE4
    HFM_NOPIN,          //HFGPIO_F_RESERVE5

    HF_M_PIN(16),   // USERGPIO_F_ATCMD_MODE, HF-LPB 16 pin can be used for interrupt
    HFM_NOPIN,      //
};
```

After modified, "nLink" LED and "nReady" LED are never on.。

5.4 Control GPIO PIN

HSF support Function Code to control GPIO rather than direct control. So users must define a Function Code before manipulate GPIO and map the corresponding PIN with the Function Code in "hf_gpio_fid_to_pid_map_table".

Refer to "HSF1.1x API Reference Manual" for more Function Code API.

6. User Defined AT Commands

HSF is able to analyze AT commands. Use need to modify the following table to add new AT commands.

```
const hfat_cmd_t user_define_at_cmds_table[]=
{
    {"UMYATCMD",hf_atcmd_myatcmd,"    AT+UMYATCMD=code\r\n",NULL},
    {NULL,NULL,NULL,NULL}        //The last item must be NULL for   HSF to judge the end of table.
};
```

Each item in "user_define_at_cmds_table" represent one AT command. Every AT command has four attribute: Command Name, Command Entry Function, Command Help Description(This is displayed for AT+H), the last attribute is reserved.

Steps to add user defined AT command.

1， Add AT command name

2， Add AT command entry process function.

3， Add help description of this command.

Reter to example in SDK directory "example/attest.c" for more details. Input "AT+H" to see AT command list as following picture.

```
AT+SOCKB: Set/Get Parameters of socket_b.
AT+TCPLKB: Get The state of TCP_B link.
AT+TCPTOB: Set/Get TCP_B time out.
AT+TCPDISB: Connect/Dis-connect the TCP_B Client link.
AT+RCVB: Recv data from socket_b
AT+SNDB: Send data to socket_b
AT+MDCH: Put on/off automatic switching WIFI mode.
AT+RELD: Reload the default setting and reboot.
AT+SLPEN: Put on/off the GPIO7.
AT+RLDEN: Put on/off the GPIO45.
AT+Z: Reset the Module.
AT+MID: Get The Module ID.
AT+VER: Get application version.
AT+UMYATCMD=code                              ────► User Defined AT Command
AT+NTIME:set/get system time
AT+NMEM:show system memory stat
AT+NDBGL:set/get debug level
AT+H:show help
+ok
```

7. Print Debug Information

HSF support "u_printf" and "HF_debug" API function to enable UART printing debug information. Debug information is not print by default. Call "hfdbg_set_level" API or AT command "AT+NDBGL=1" to enable output debug information. "At+NDBGL=0" to disable output.

(remark: debug mode should be closed when program issued ,because under debug mode, the hardware watch dog is disabled.)

8. ULINK online Debug under Keil-MDK environment

If under Keil MDK environment, LPB100-EVK must debug via ULINK2

1. Press "build" (or press F7) to create target file;
2. Make sure ulink connect to device and the current firmware is run under Debug mode, can configure via AT+NDBG; press "download" ,download the target file to module.
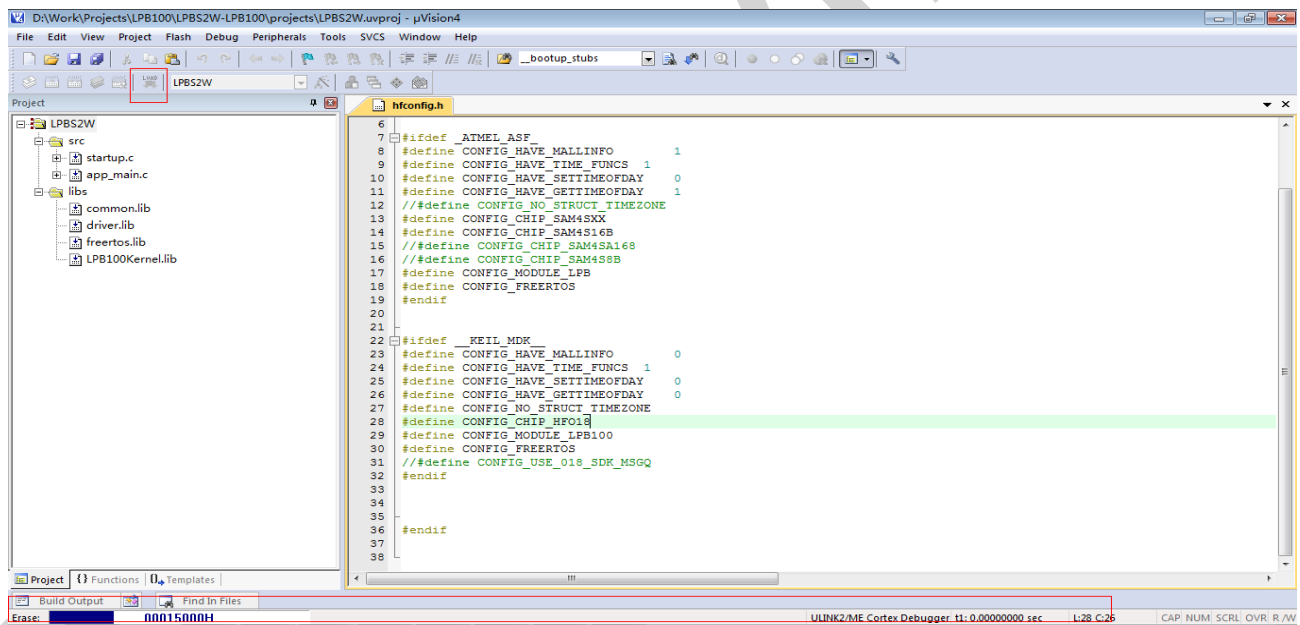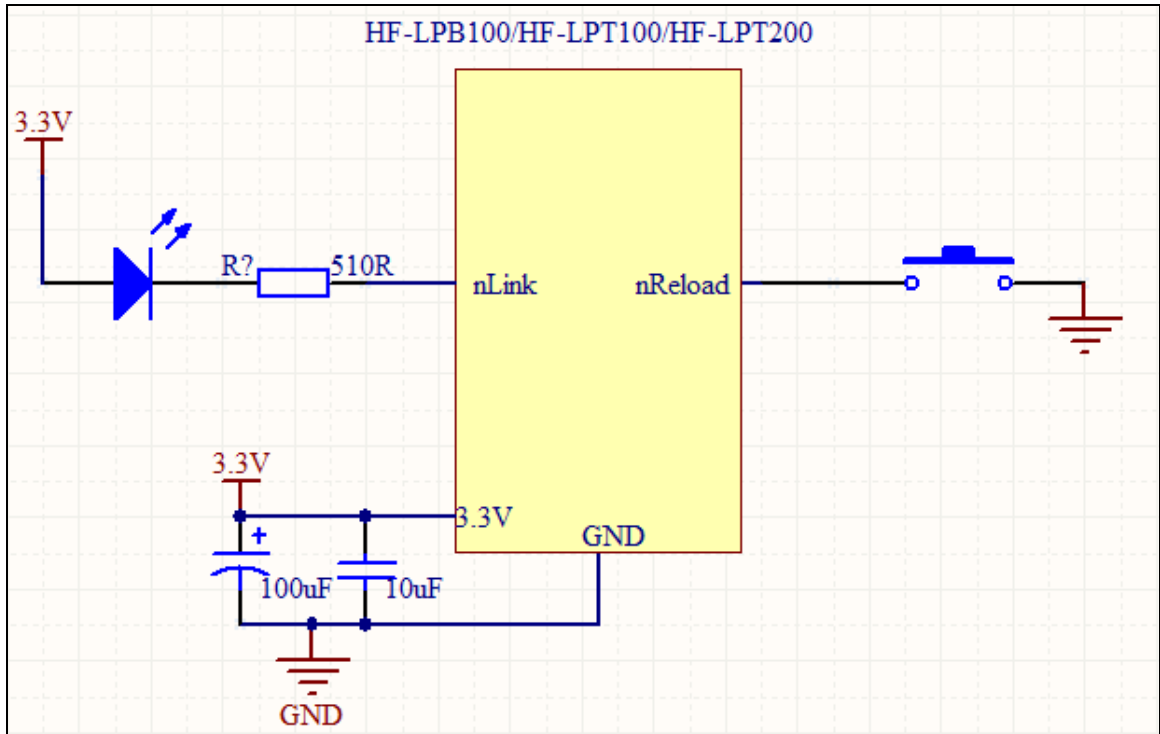

3. Slect menu "debug" – "Start/Stop Debug session" to start debug. Now user can start to debug own program.
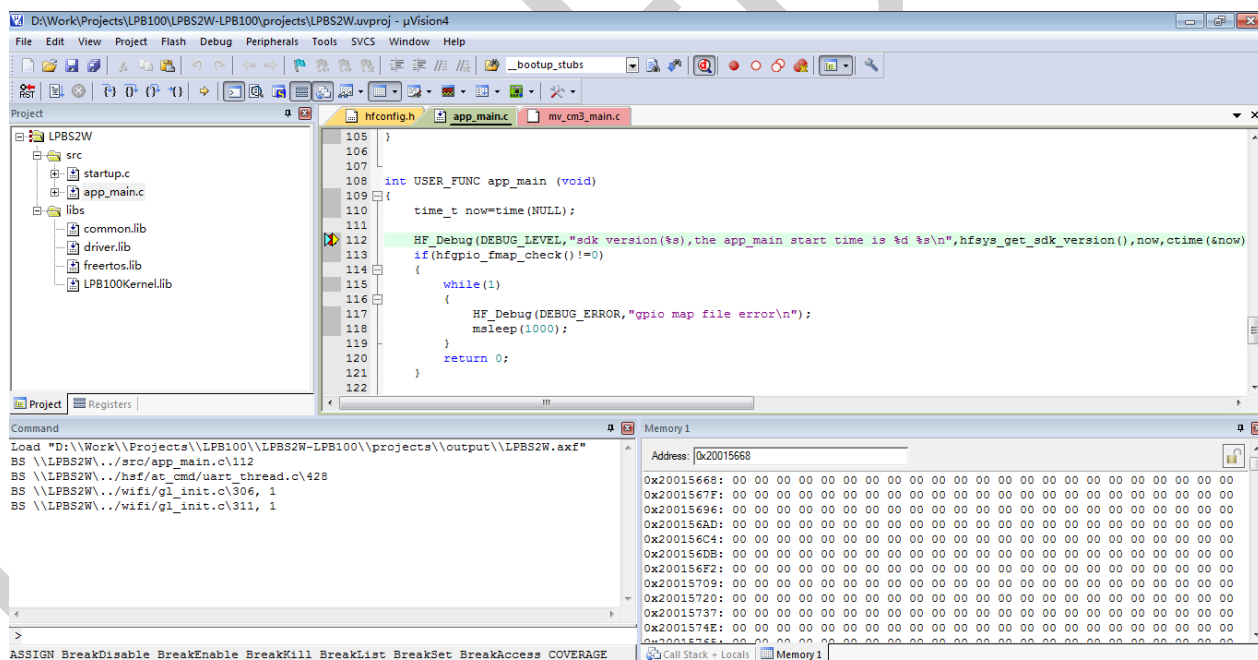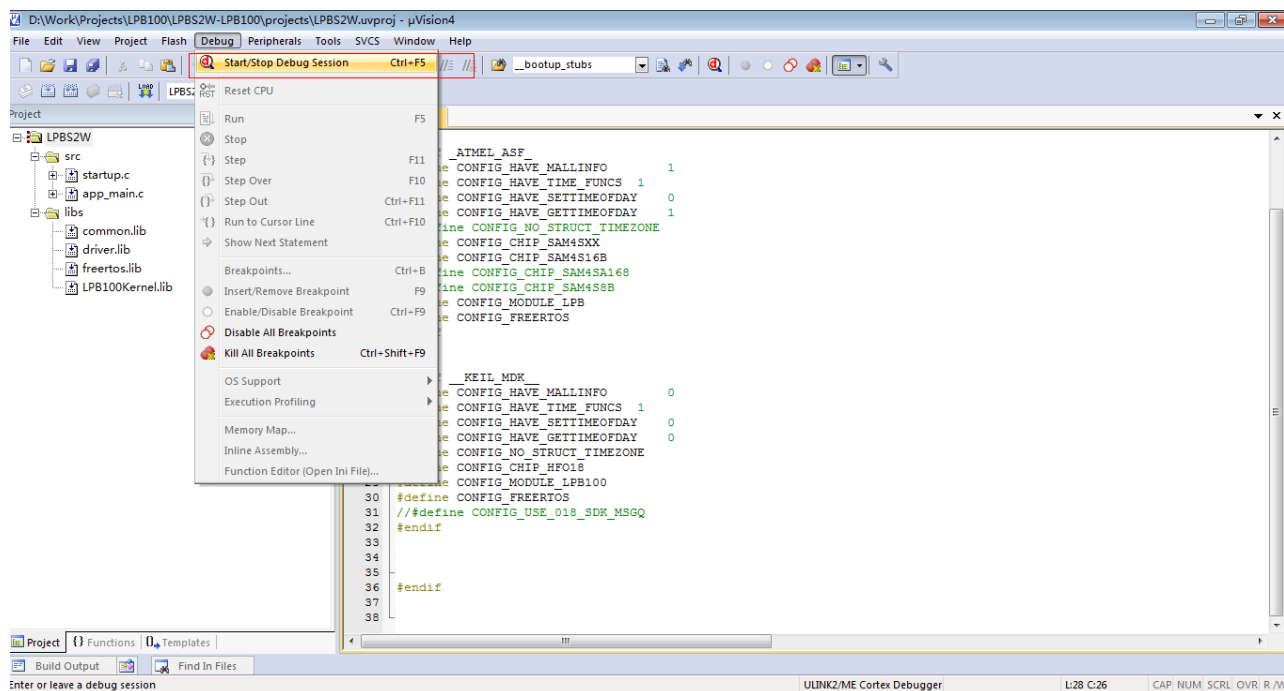
9.   Hardware recommend connection

9.1 hardware recommend connection

nReload button function:

1.   When module power on, if the pin is low (press the button), the module enter mass upgrade ,configure mode

(Please refer user manual affiliate D, download production tool from our web, support user to mass upgrade and configure.)

2.   After power on, press the button shortly(<3S), module enters into smart link mode, wait for APP to push password

   (Please refer user manual affiliate D, download smart link app from our web, support user to one-key configure module)

3. After power on, press the button over 3 seconds, the module restore to factory setting.

Remark: if user need to mass upgrade or configure software, we strongly recommend set this pin.

nLink indication function

1. When mass upgrade and configure wirelessly, it indicates the completion of upgrade and configuration.
2. Under smart link mode, flash slowly indicates APP is smart linking.
3. In normal mode, it act ass wifi connect status.

   Remark : : if user need to mass upgrade or configure software, we strongly recommend set this pin

10. How to upgrade

10.1 upgrade via uart

Press reload button, and press space key at the same time, user can enter into boot program command, to select firmware to upgrade.

```
HF-LPB100 Bootloader V1.0.5, please entry code to choose :
 'B': Clean All Config.
 'F': Program Firmware. 升级WIFI 固件
 'N': Program NVRAM data.
 'S': Program application.升级应用程序
 'G': Run applications.
```

10.2 upgrade via WEB

Via browsers to visit LPB100 server to upgrade.

10.3 upgrade via ULINK

Please refer Keil MDK user manual

10.4 upgrade via HF production tool to mass upgrade

Please refer (HF-LPB100 module upgrade procedure)

10.5 notice

Since LPB100 and LPT100 was edited via 1MB SDK (1.03a), there is two situation when upgrade from 1.03a to 2MB SDK
1. The program edited from 2MB SDK is less than 400KB, user can upgrade from above program directly.

2.   The program edited from 2MB SDK is bigger than 400KB, it is unable to upgrade from 1.03a. It requires a transition program (a program less than 400KB edited from 2MB SDK). In SDK, doc have two transition program.

  LPBS2W_1MB_TO_2MB.bin   if upgrade via uart can apply this.
  LPBS2W_1MB_TO_2MB_UPGRADE. bin   if upgrade via web or HF production tool apply this.

After upgrade the transition program, then upgrade the program which is bigger than 400KB. If upgrade directly without transition program, the module' MAC address ,configuration, wifi firmware will be erased and unable to launch. Only via boot-loader to upgrade wifi firmware, then the program can be launch.

Upgrade between 1MB SDK ; and between 2MB SDK, there is no above limitation and able to upgrade directly.
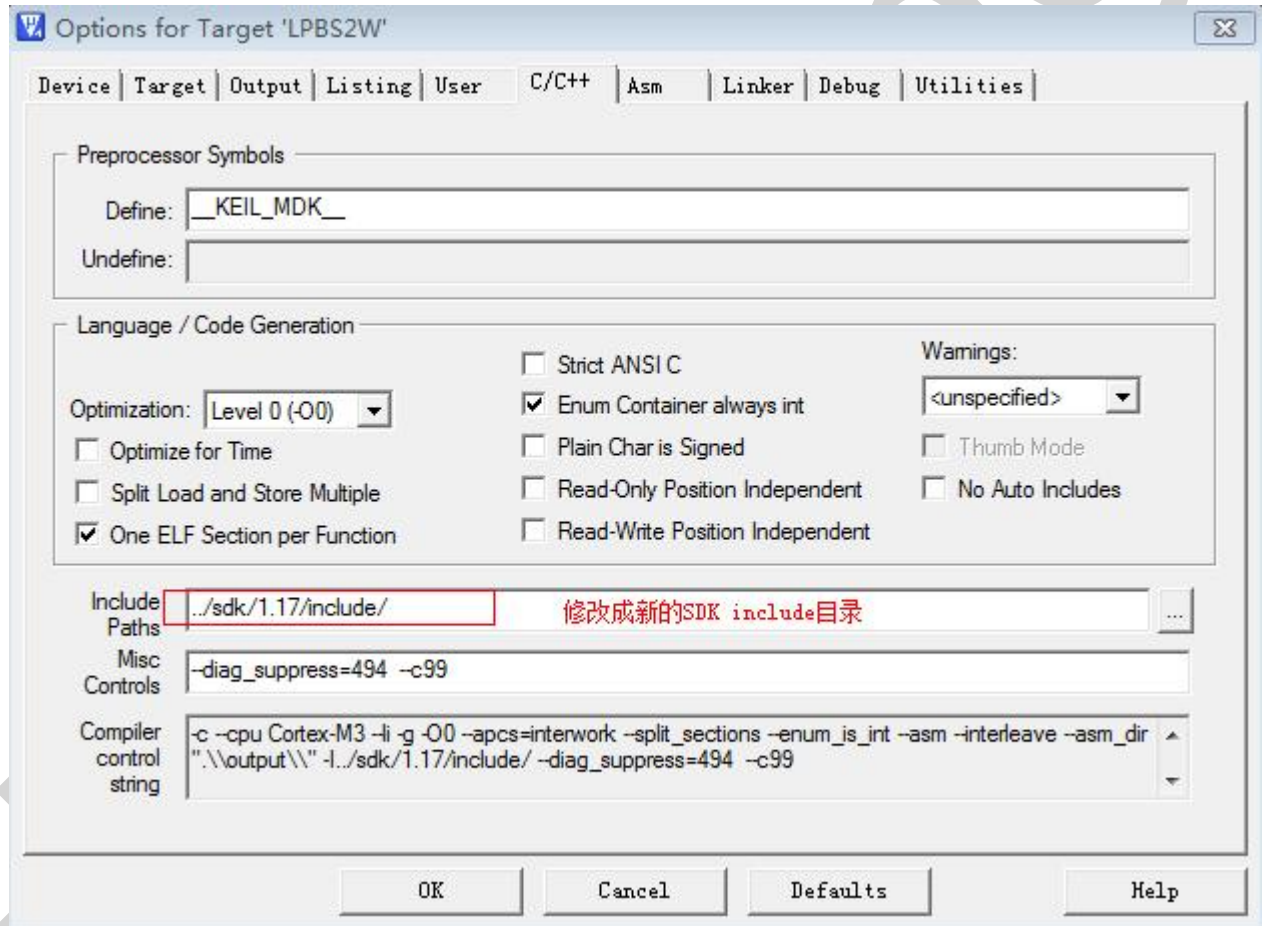
11. SDK Upgrade

The released new SDK is usually compatible with old SDK. It is better for user to apply new SDK file. User just need to copy his own code to new SDK, replace SDK file with his own app_main.c

Or revise his file according to new SDK file, replace old lib and include directory with new lib and include directory in new SDK.

Revise new include route.

Revise new lib route
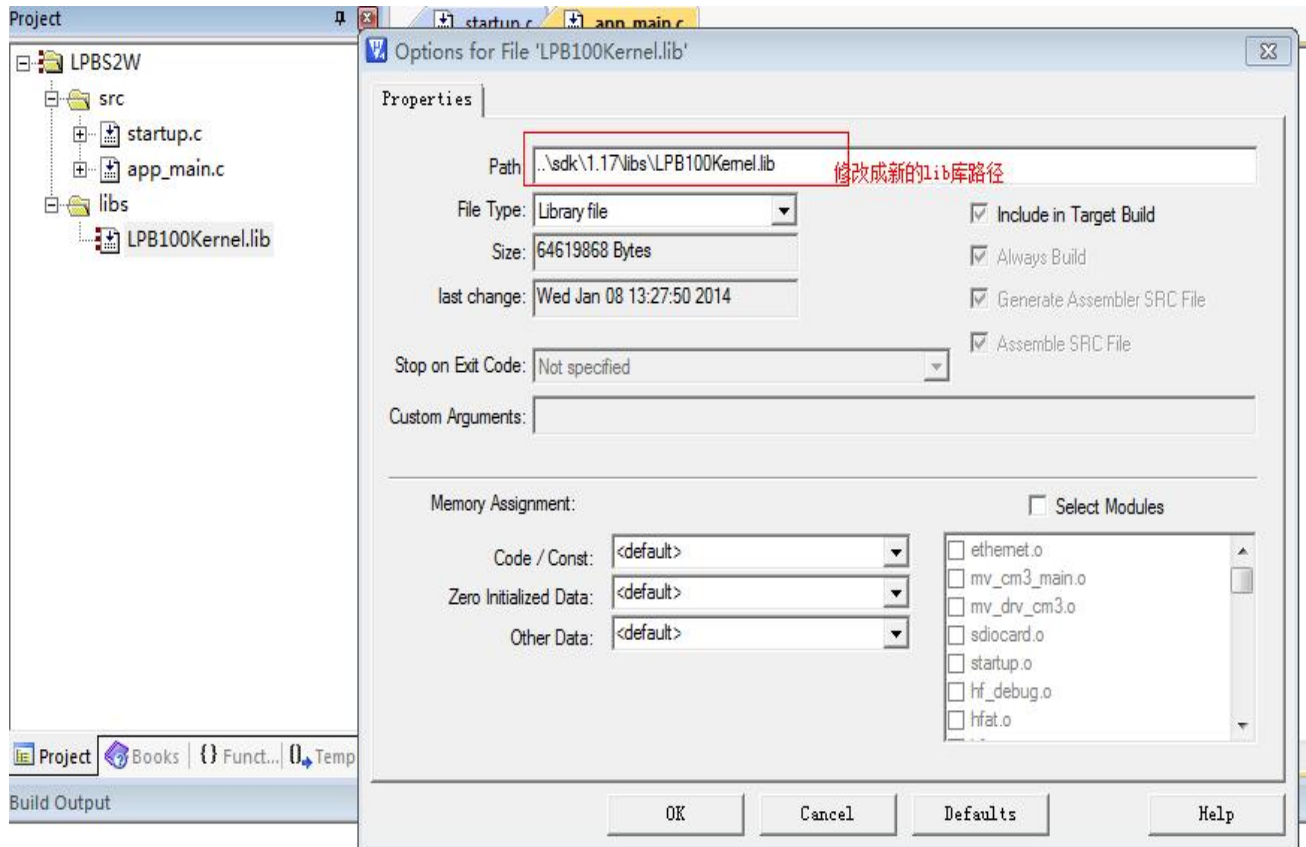
12. Example

Example project of HF-LPB100 is "LPBExample.cproj". There is several examples. Edit "#define" in "example.h" to select one example to compile. Select the "LPBExapmple.cproj" project as startup project. As for HF-LPB100, example project is in "example/projects", better to set debug level two(AT+NDBGL=2) to enable "u_printf" API printing debug information via serial port.

```
]#ifndef _H_EXAMPLE_H_H_H_
 #define _H_EXAMPLE_H_H_H_

 #define USER_AT_CMD_DEMO 1    AT 命令 Demo

 #define USER_GPIO_DEMO   2   GPIO 控制 Demo

 #define USER_TIMER_DEMO  3   定时器 Demo

 #define USER_THREAD_DEMO 4  多线程Demo

 #define USER_CALLBACK_DEMO 5   SOCKETA/SOCKETB 回调机制

 #define USER_FILE_DEMO      6 用户配置文件 Demo

 #define USER_FLASH_DEMO     7用户Flash操作

 #define USER_SOCKET_DEMO    8    标准socket Demo

 #define USER_IR_DEMO      9    红外遥控 Demo

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

12.1. Send HTTP Request via AT command

Example code is in "example/at/attest.c". The corresponding macro define is "USER_AT_CMD_DEMO". Be familiar with "httpc" interface function and add user AT command via this example.

Compile and run, input AT command to send HTTP request and print the server response via serial port.

Example:

Input "AT+UMYATCMD=1,www.baidu.com" and the response is as follows.

```
AT+UMYATCMD=1,www.baidu.com
send_at_cmd [207] name:UMYATCMD [3]
UMYATCMD
1
www.baidu.com
<!DOCTYPE html><!--STATUS OK--><html><head><meta http-equiv="content-type" conte
nt="text/html;charset=utf-8"><title>鐧惧害涓€涓嬶紝浣犲氨鐭ラ亾</title><style >h
tml,body{height:100%}html{overflow-y:auto}#wrapper{position:relative;_position:;
min-height:100%}#content{padding-bottom:100px;text-align:center}#ftCon{height:10
0px;position:absolute;bottom:44px;text-align:center;width:100%;margin:0 auto;z-i
ndex:0;overflow:hidden}#ftConw{width:720px;margin:0 auto}body{font:12720px;margi
px arial;text-align:;background:#fff}body,p,form,ul,li{margin:0;padding:0;list-s
tyle:none}body,form,#fm{position:relative}td{text-align:left}img{border:0}a{colo
r:#00c}a:active{color:#f60}.bg{background-image:url(http://s1.bdstatic.com/r/www
/cache/static/global/img/icons_0bf3824a.patic/glng);background-repeat:no-repeat;
_background-image:url(http://s1.bdstatic.com/r/www/cache/static/global/img/icons
_0bf2f8a8.gif)}.c-icon{display:inline-block;width:14px;height:14px;vertical-alig
n:text-bottom;font-style normal;overflow:hidden;background:url(http://s1.bdstati
c.com/r/www/cache/static/global/img/icons_0bf3824a.png) no-repeat 0 0;_backgroun
d-image:url(http://s1.bdstatic.com/r/www/cache/static/global/img/icons_0bf2f8a8.
gif)}.c-icon-triangle-down从www.baidu.com页面下载来的网页文件480px -24px}.c-icon-chevron
-unfold2{background-position:-504px -24px}#u{color:#999;padding:4px 10px 5px 0;t
ext-align:right}#u a{margin:0 5px}#u .reg{margin:0}#m{width:720px;margin:0 auto}
#nv a,#nv b,.btn,#lk{font-size:14px}#fm{padding-left:110px;text-align:left;z-ig-
left:110px;text-ndex:1}input{border:0;padding:0}#nv{height:19px;font-size:16px;m
argin:0 0 4px;text-align:left;text-indent:137px}.s_ipt_wr{width:418px;height:30p
x;display:inline-block;margin-right:5px;background-position:0 -288px;border:1px
solid #b6b6b6;border-color:#9a9a9a #cdcdcd #cdcdcd #9a9a9a;vertical-align:top}.s
_ipt{width:405px;height:22px;font:16px/22px arial;margin:5px 0 0 7px;background:
#fff;outline:0;-webkit-appearance:none}.s_btn{width:95px;height:32px;padding-top
:2px\9;font-size:14px;background-color:#ddd;background-position:0 -240px;cursor:
pointer}.s_btn_h{background-position:-240px -240px}.s_btn_wr{width:97px;height:3
4px;display:inline-block;background-position:-120px -240px;*position:relative;z-
index:0;vertical-align:top}#lg img{vertical-alal-align:top}#lg iign:top;margin-b
ottom:3px}#lk{margin:33px 0}#lk span{font:14px "渔骓绲"}#lm{height:60px}#lh{marg
in:16px 0 5px;word-spacing:3px}.tools{position:absolute;top:-4px;*top:10px;right
```

12.2. User Defined nReload and nLink

Example code is in "example/gpio/gpiotest.c". The corresponding macro define is "USER_GPIO_DEMO". Be familiar with GPIO API via modify the default PIN function.

Compile and run, press RELOAD is no longer restore factory setting, but became the nLink light switch. nLink light did not light up after STA connect successfully, but after TCP connect successfully. Press RQ key, under DEBUG mode, will print data. (DEBUG mode can set through AT-NDBGL=2)

12.3 Timer control the flash of nLink ,nReady.

Code is in example/gpio/gpiotest.c, the correspondent macro define is USER_GPIO_DEMO. Through this example can be familiar with the create of thread,

the creation of timer and related API function.

Compile and run, nLink and nReady flash at 1HZ frequency.

12.4 Net callback system control NLINK status

Example Code is in example/betcallback.c, the correspondent macro is USER_CALLBACK_DEMO, through this example, user can be familiar with socketa/b send API, serial send API, and network callback.

Compile and run, when remote and module connected through SockrtA, module send "CONNECT OK" to remote, remote send "GPIO NLINK LOW" ,nLink light is off; if "GPIO NLINK HIGH", then nLink light is on; if "GPIO NLINK FLASH", the nLink flash. Other character transparent transmitted to serial.

12.5 Via AT command operate user configure file.

Example code is in example/filetest.c, the correspondent macro is USER_FILE_DEMO, through this examples, we can be familiar with user file API, user define AT command

Compile and run, add an AT command: AT+FTEST. Via this command can operate user file. AT+FTEST=code,offset,len/value

Code: 0   length of file ; AT+FTEST=0,0,0

Code:1   the content of file and print via serial; AT+FTEST=1,0,100

Code:5   the content of file and print via serial as hex formal. AT+FTEST=1,100,100

Code: 2   write file; AT+FTEST=2,0,test123456789

Code: 3   test if file port is correct; AT+FTEST=3,0,0

Code :4   clear all user file ; AT+FTEST=4,0,0
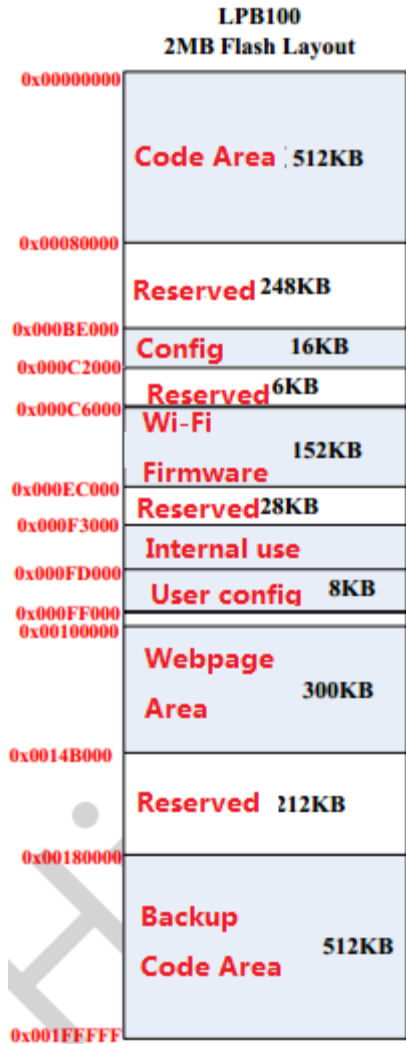
Offset: the offset of file

Len: the length of reading file

Value: the character string writing into file

13. Flash Layout

13.1 LPB100 Flash Layout

**LPB100
2MB Flash Layout**

Address	Region	Size
0x00000000	Code Area	512KB
0x00080000	Reserved	248KB
0x000BE000	Config	16KB
0x000C2000	Reserved	6KB
0x000C6000	Wi-Fi Firmware	152KB
0x000EC000	Reserved	28KB
0x000F3000	Internal use	
0x000FD000	User config	8KB
0x000FF000		
0x00100000	Webpage Area	300KB
0x0014B000	Reserved	212KB
0x00180000	Backup Code Area	512KB
0x001FFFFF		

Address: Room.511/510, Building 7, No.365, Chuanhong Road, Pudong New Area,
Shanghai, China, 201202
Web: www.hi-flying.com
Service Online: 400-189-3108
Sales Contact: sales@hi-flying.com

For more information about High-Flying modules, applications, and solutions, please visit our web site http://www.hi-flying.com/en/

END OF DOCUMENT