API

HF-LPX70 SDK API Reference

CONTENT

1. LINUX STANDARD C FUNCTION

HSF-LPX30 is compatible with standard C library function, such as memory management, string, time and standard input and output etc. Some relative function introduction please refer to standard C library function.

2MB Flash version support TLS encryption, see detail interface in mbedtls_compatible_cyassl.h and example code in ssltest.c file.

2. SYSTEM ERROR CODE DEFINITION

The return value of API function（except special instructions）provision， success "HF_SUCCESS", or ">0", fail "<0". The error code is 4Bytes signed integer, The return value is negative of error code. "31-24" bit for module index, "23-8" reserved, "7-0" is error code.

```
#define MOD_ERROR_START(x) ((x << 16) | 0)
/* Create Module index */
#define MOD_GENERIC 0
/** HTTPD module index */
#define MOD_HTTPDE 1
/** HTTP-CLIENT module index */
#define MOD_HTTPC 2
/** WPS module index */
#define MOD_WPS 3
/** WLAN module index */
#define MOD_WLAN 4
/** USB module index */
#define MOD_USB 5
/*0x70~0x7f user define index*/
#define MOD_USER_DEFINE (0x70)
/* Globally unique success code */
#define HF_SUCCESS 0
enum hf_errno {
/* First Generic Error codes */
   HF_GEN_E_BASE = MOD_ERROR_START(MOD_GENERIC),
   HF_FAIL,
   HF_E_PERM, /* Operation not permitted */
   HF_E_NOENT, /* No such file or directory */
   HF_E_SRCH, /* No such process */
   HF_E_INTR, /* Interrupted system call */
   HF_E_IO, /* I/O error */
   HF_E_NXIO, /* No such device or address */
   HF_E_2BIG, /* Argument list too long */
   HF_E_NOEXEC, /* Exec format error */
   HF_E_BADF, /* Bad file number */
   HF_E_CHILD, /* No child processes */
   HF_E_AGAIN, /* Try again */
   HF_E_NOMEM, /* Out of memory */
```

```
    HF_E_ACCES, /* Permission denied */
    HF_E_FAULT, /* Bad address */
    HF_E_NOTBLK, /* Block device required */
    HF_E_BUSY, /* Device or resource busy */
    HF_E_EXIST, /* File exists */
    HF_E_XDEV, /* Cross-device link */
    HF_E_NODEV, /* No such device */
    HF_E_NOTDIR, /* Not a directory */
    HF_E_ISDIR, /* Is a directory */
    HF_E_INVAL, /* Invalid argument */
    HF_E_NFILE, /* File table overflow */
    HF_E_MFILE, /* Too many open files */
    HF_E_NOTTY, /* Not a typewriter */
    HF_E_TXTBSY, /* Text file busy */
    HF_E_FBIG, /* File too large */
    HF_E_NOSPC, /* No space left on device */
    HF_E_SPIPE, /* Illegal seek */
    HF_E_ROFS, /* Read-only file system */
    HF_E_MLINK, /* Too many links */
    HF_E_PIPE, /* Broken pipe */
    HF_E_DOM, /* Math argument out of domain of func */
    HF_E_RANGE, /* Math result not representable */
    HF_E_DEADLK, /*Resource deadlock would occur*/
};
```

Header File：

hferrno.h

3. AT COMMAND API

● hfat_get_words

Function prototype:

int hfat_get_words((char *str,char *words[],int size);

Description:

Get all the response parameters of AT command

Parameters:

str: Pointer to the AT command response string(Ex. "+ok=WPA2PSK,AES,12345678"), the str pointed address should be in RAM area.

words: Pointer to the string value of each AT command response parameters

size: number of words.

Return Value:

<=0: The string of str pointed is not a valid AT command response.

>0: The number of words parsed.

Remark:

AT command use the following character separator ',', '=', ' ', "\r\n"

Example:

Example/attest.c

Header file:

hfat.h

● hfat_send_cmd

Function prototype:

int hfat_send_cmd(char *cmd_line,int cmd_len,char *rsp,int len);

Description:

Send AT command. Response is saved in buffer.

Parameters:

cmd_line: AT command string,

Format is AT+CMD_NAME[=][arg,]...[argn], E.g "AT+WMODE\r\n"

cmd_len: Length of cmd_line including end character

rsp: The AT command response buffer

len: The response length

Return Value:

HF_SUCCESS:Set success,HF_FAIL:Set fail

Remark:

The function execute is same as through UART send AT command, It don’t support "AT+H" and "AT+WSCAN" at persent, Wi-Fi scan can refer to hfwifi_scan, AT command execute result saved in rsp. rsp is a string, To the specific format, Pls refer to UART AT command help manual; Through the function can get setting system configuration.

Attention: This function can’t send extended AT command through user_define_at_cmds_table. Cuz the extended AT command can be called directly , No need to be implemented by sending the AT command, If user through user_define_at_cmds_table to extend existed AT command, Such as ”AT+VER“,

If send hfat_send_cmd(“AT+VER\r\n”,sizeof(“AT+VER\r\n”),rsp,64);

The return will come with T+VER instead of extended.

Example:

example/attest.c

Header file:

hfat.h

● hfat_enable_uart_session

Function prototype:

int hfat_enable_uart_session(char enable);

Description:

Enable/close +++ transparent mode to command mmode

Parameters:

enable：1-enable，0-close；

Return Value:

HF_success: success，HF_FAIL: fail；

Notes:

None

Example:

None

Header:

hfat.h

4. DEBUG API

● HF_Debug

Function prototype:

void HF_Debug(int debug_level,const char *format , ...);

Description:

Output debug information to UART

Parameters:

debug_level: Debug level, it can be:.

#define DEBUG_LEVEL_LOW 1

#define DEBUG_LEVEL_MID 2

#define DEBUG_LEVEL_HI 3

Or other larger value, With hfdbg_set_level set debug level can only output the above setting level of logo information,Log information needs to be enabled first.

format: formated output, the same as printf.

Maximum 250 bytes. If exceed, pls call more times to print.

Return Value:

None

Notes:

AT+NDBGL=X,Y can enable debug information output,X represent debug level(0:close), Y represent UART number (0: UART 0,1:UART 1), Recommend debug info ouput to UART 1(To UART 1 pin, Pls refer to module manual), UART 0 used for normal interactive communication, Pls dynamic turn to debug after program released, Then can use AT+NDBGL command open, Don’ t need to debug when AT+NDBGL=0 off.

Examples:

None

Header file:

hf_debug.h

● hfdbg_get_level

Function prototype:

int hfdbg_get_level ();

Description:

Get current debug level;

Parameters:

None

Return Value:

Return the current debug level.

Remark:

None

Examples:

None

Header file:

hf_debug.h

- ## hfdbg_set_level

Function prototype:

void hfdbg_set_level (int debug_level);

Description:

Set debug info output level or close debug level

Parameters:

debug_level: debug level, it can be 0:Close debug info output

#define DEBUG_LEVEL_LOW 1

#define DEBUG_LEVEL_MID 2

#define DEBUG_LEVEL_HI 3

Return Value:

None

Remark:

Recommend to use UART AT+NDBGL command dynamic enable or close debug info output. Then user can check log anytime and don' t need modify program.

Examples:

None

Header file:

hf_debug.h

5. GPIO CONTROL API

● hfgpio_configure_fpin

Function prototype:

int hfgpio_configure_fpin(int fid,int flags);

Description:

Configure the PIN according to fid(function id);

Parameters:

fid(function id)

```
enum HF_GPIO_FUNC_E
{
    //////fix////////////////////
    HFGPIO_F_JTAG_TCK=0,
    HFGPIO_F_JTAG_TDO=1,
    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,
    ///////////////////////////////
    HFGPIO_F_NLINK,
    HFGPIO_F_NREADY,
    HFGPIO_F_NRELOAD,
    HFGPIO_F_SLEEP_RQ,
    HFGPIO_F_SLEEP_ON,
    HFGPIO_F_WPS,
    HFGPIO_F_IR,
    HFGPIO_F_RESERVE2,
    HFGPIO_F_RESERVE3,
    HFGPIO_F_RESERVE4,
    HFGPIO_F_RESERVE5,
    HFGPIO_F_USER_DEFINE
};
```

Fid can also be user defined function id. It should start from HFGPIO_F_USER_DEFINE.

flags: PIN property, it can be one or multiple of the following value(use '|' operation).

HFPIO_DEFAULT	Default
HFM_IO_TYPE_INPUT	Input Mode
HFM_IO_OUTPUT_0	Output low level
HFM_IO_OUTPUT_1	Output High level

Return Value:

HF_SUCCESS:Set success,HF_E_INVAL: fid is invalid or PIN is invalid.

HF_E_ACCES: The corresponding PIN does not have the setting property(flags), For example HFGPIO_F_JTAG_TCK corresponding pin is a peripheral PIN, not a GPIO, it can’ t configue other property except HFPIO_DEFAULT.

Remark:

Before setting, User should be clear about the property of function id corresponding to the PIN , Pls refer to related datasheet to check every pin property. If configure a PIN which does not have the property, Pls return to HF_E_ACCES.

Examples:

None

Header file:

hfgpio.h

- ## hfgpio_fconfigure_get

Function prototype:

int hfgpio_fconfigure_get(int fid);

Description:

Get the fid mapping PIN property value.

Parameters:

fid: Function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid.

Return Value:

Success return to corresponding property value of PIN, Property value can refer to hfgpio_configure_fpin，HF_E_INVAL: fid illgel or the corresponding PIN is illgel.

Remark:

None

Example:

gpiotest.c

Header file:

hfgpio.h

● hfgpio_fpin_add_feature

Function prototype:

int HSF_API hfgpio_fpin_add_feature(int fid,int flags);

Description:

Add property value for corresponding PIN of fid.

Parameters:

fid: Function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid.

flags: Refer to hfgpio_configure_fpin flags;

Return Value:

HF_SUCCESS:Set success, HF_E_INVAL: fid or PIN is illegal

Notes:

None

Examples:

gpiotest.c

Header file:

hfgpio.h

● hfgpio_fpin_clear_feature

Function prototype:

int HSF_API hfgpio_fpin_clear_feature (int fid,int flags);

Description:

Clear one or multi property of fid PIN

Parameters:

fid: Function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid.

flags: Refer hfgpio_configure_fpin flags;

Return Value:

HF_SUCCESS:Set success, HF_E_INVAL: fid or PIN is illegal.

Notes:

None

Examples:

gpiotest.c

Header file:

hfgpio.h

- ## hfgpio_fpin_is_high

Function prototype:

int hfgpio_fpin_is_high(int fid);

Description:

Judge the fid PIN is high level or not;

Parameters:

fid: Function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid.
,the corresponding PIN must have F_GPO or F_GPI attribute;

Return Value:

Return 0 if PIN is low level, return 1 if PIN is high level, reutrn <0 if PIN is illegal.

Notes:

None

Examples:

example/gpiotest.c

Header file:

hfgpio.h

- ## hfgpio_fset_out_high

Function prototype:

int hfgpio_fset_out_high(int fid);

Description:

Set the fid mapping PIN as output high level.

Parameters:

fid: Function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid。

Return Value:

HF_SUCCESS:Set success,HF_E_INVAL: fid or PIN is invalid,
HF_FAIL:Set fail； HF_E_ACCES:The pin property don’t support input

Notes:

This API equal to hfgpio_configure_fpin(fid, HFM_IO_OUTPUT_1| HFPIO_DEFAULT);

Example:

example/gpiotest.c

Header file:

hfgpio.h

- ## hfgpio_fset_out_low

Function prototype:

int hfgpio_fset_out_low(int fid);

Description:

Set fid mapping pin to output low level

Parameters:

fid: function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid.

Return Value:

HF_SUCCESS:Set success,HF_E_INVAL: fid or PIN is invalid.

Remark:

The API is equal to hfgpio_configure_fpin(fid, HFM_IO_OUTPUT_0| HFPIO_DEFAULT);

Example:

example/gpiotest.c

Header file:

hfgpio.h

- ## hfgpio_pwm_disable

Function prototype:

int hfgpio_pwm_disable(int fid);

Definition:

Close PWM output

Parameter:

fid: function id, refer to HF_GPIO_FUNC_E, it can also be user defined fid.

Return Value:

HF_SUCCESS:set success，HF_E_INVAL: fid illegal or the related PIN illegal.

HF_FAIL:set fail；HF_E_ACCES: related PIN don't have F_PWM property,can not confirgure to PWM mode;

Remark:

None

Example:

attest.c

Header File:

hfgpio.h

- ## hfgpio_pwm_enable

Function prototype:

int HSF_API hfgpio_pwm_enable(int fid, int freq, int hrate);

Definition:

Open PWM output

Parameter:

fid: function id, refer to HF_GPIO_FUNC_E,   it can also be user defined fid.

freq: output frequency，Range 200-1000000, GPIO 3 only support range

2000-1000000.
hrate: Output duty ratio, Range 1-99

Return Value:

HF_SUCCESS: set success，HF_E_INVAL: fid illegal or the related PIN illegal,

HF_FAIL:set fail；HF_E_ACCES:related PIN don't have F_PWM property,can not confirgure to PWM mode;

Remark:

None

Example:

attest.c

Header file:

hfgpio.h

● hfgpio_adc_enable

Function prototype:

int hfgpio_adc_enable(int fid);

Definition:

Open ADC function

Parameter:

fid: function id ,refer to HF_GPIO_FUNC_E,it can also be user defined fid;

Return Value:

HF_SUCCESS: set success，HF_E_INVAL: fid illegal or its related PIN illegal,
HF_FAIL: set fail ;HF_E_ACCES: related PIN don't have F_ADC property,can not configured as ADC mode;

Notes:

None

Example:

attest.c

Header:

hfgpio.h

● hfgpio_adc_get_value

Function prototype:

int HSF_API hfgpio_adc_get_value(int fid);

Definition:

Read ADC level value

Parameter:

fid:function id,refer to HF_GPIO_FUNC_E, it can also be user defined fid;

Return Value:

ADC Level，Range 0-1023

Remark:
ADC is 10-bit sample, Before call this function,User need call hfgpio_adc_enable to open ADC, Sample value is corresponding to 2V, 0V the return value is 0, 2V the return value is 1023
Example:

attest.c

Header File:

hfgpio.h

6. WIFI API

● hfsmtlk_start

Function prototype:

int HSF_API hfsmtlk_start(void);

Description:

Start smartlink.

Parameters:

None

Return Value:

Return HF_SUCCESS if success, return others if fail

Remark:

After calling this function, the program will be rebooted immediately

Examples:

None

Header file:

hfsmtlk.h

● hfsmtlk_stop

Function prototype:

int HSF_API hfsmtlk_stop(void);

Description:

Stop smartlink.

Parameters:

None

Return Value:

Return HF_SUCCESS if success, return others if fail

Remark:

None

Examples:

None

Header file:

hfsmtlk.h

● hfwifi_scan

Function prototype:

int HSF_API hfwifi_scan(hfwifi_scan_callback_t p_callback);

Description:

Scan the existing AP

Parameters:

hfwifi_scan_callback_t: When the device scan to existing AP, Through this callback and tell user the AP details.

```
typedef int (*hfwifi_scan_callback_t)( PWIFI_SCAN_RESULT_ITEM );
typedef struct _WIFI_SCAN_RESULT_ITEM
{
   uint8_t auth; //Authentication
   uint8_t encry;//Encryption
   uint8_t channel;//AP Channel
   uint8_t rssi;//RSSI in percentage
   char ssid[32+1];//AP SSID
   uint8_t mac[6];//AP MAC
   int rssi_dbm;//The RSSI in dBm value
   int sco;
}WIFI_SCAN_RESULT_ITEM,*PWIFI_SCAN_RESULT_ITEM;
#define WSCAN_AUTH_OPEN 0
#define WSCAN_AUTH_SHARED 1
#define WSCAN_AUTH_WPAPSK 2
#define WSCAN_AUTH_WPA2PSK 3
#define WSCAN_AUTH_WPAPSKWPA2PSK 4
#define WSCAN_ENC_NONE 0
#define WSCAN_ENC_WEP 1
#define WSCAN_ENC_TKIP 2
#define WSCAN_ENC_AES 3
#define WSCAN_ENC_TKIPAES 4
```

Return Value:

Return HF_SUCCESS if success, return others if fail

Remark:

The function will not exit during scan, If exit, It means finished.

Example:

example/wifitest.c

Header file:

hfwifi.h

- ## hfwifi_scan_ex

Function prototype:

int HSF_API hfwifi_scan(hfwifi_scan_callback_t p_callback);

Description:

Scan the existing AP

Parameters:

hfwifi_scan_callback_t: When the device scan to existing AP, Through this callback and tell user the AP details.

```
typedef int (*hfwifi_scan_callback_t)( PWIFI_SCAN_RESULT_ITEM );
typedef struct _WIFI_SCAN_RESULT_ITEM
{
   uint8_t auth; //Authentication
   uint8_t encry;//Encryption
   uint8_t channel;//AP Channel
   uint8_t rssi;//RSSI in percentage
```

```
    char ssid[32+1];//AP SSID
    uint8_t mac[6];//AP MAC
    int rssi_dbm;//The RSSI in dBm value
    int sco;
}WIFI_SCAN_RESULT_ITEM,*PWIFI_SCAN_RESULT_ITEM;
#define WSCAN_AUTH_OPEN 0
#define WSCAN_AUTH_SHARED 1
#define WSCAN_AUTH_WPAPSK 2
#define WSCAN_AUTH_WPA2PSK 3
#define WSCAN_AUTH_WPAPSKWPA2PSK 4
#define WSCAN_ENC_NONE 0
#define WSCAN_ENC_WEP 1
#define WSCAN_ENC_TKIP 2
#define WSCAN_ENC_AES 3
#define WSCAN_ENC_TKIPAES 4
```

ctx: Callback function parameters；

Return Value:

Return HF_SUCCESS if success, return others if fail

Remark:

The function will not exit during scan, If exit, It means finished.

Example:

example/wifitest.c

Header file:

hfwifi.h

- ## hfwifi_sta_is_connected

Function prototype:

Int hfwifi_sta_is_connected(void);

Description:

Judge the WiFi can successfully connect or not in STA mode.

Parameters:

None

Return Value:

If success, return 1. Otherwise return 0.

Notes:

None

Example:

wifitest.c

Header:

hfwifi.h

● hfwifi_transform_rssi

Function prototype:

int hfwifi_transform_rssi(int rssi_dbm);

Description:

dBm signal format transfer to percentage.

Parameters:

dbm: Signal strength, It’ s negative number.

Return Value:

The percentage strength of signal

Notes:

Transform equation equal to rssi=(dBm+95)*2

Example:

wifitest.c

Header:

hfwifi.h

● hfwifi_sta_get_current_rssi

Function prototype:

int hfwifi_sta_get_current_rssi(int *dBm);

Description:

Acquire signal strength of current connected router

Parameters:

dBm: Signal strength. If disconnect the value is -100.

Return Value:

The percentage of signal strength. Return -1, if disconnect.

Notes:

None

Example:

wifitest.c

Header:

hfwifi.h

● hfwifi_sta_get_current_bssid

Function prototype:

int hfwifi_sta_get_current_bssid(uint8_t *bssid);

Description:

Acquire BSSID of current connected router.

Parameters:

bssid：Store the connected router bssid.

Return Value:

Return HF_SUCCESS if success. Otherwise return -1.

Notes:

None

Example:

wifitest.c

Header:

hfwifi.h

- **hfwifi_read_sta_mac_address**

Function prototype:

int hfwifi_read_sta_mac_address(uint8_t *mac);

Description:

Acquire module MAC address.

Parameters:

mac: Store MAC address.

Return Value:

Return HF_SUCCESS if success. Otherwise is fail.

Notes:

The acquired MAC address is 6pcs 8bit number. If the value is 0xac,0xcf,0x88,0x88,0x88,0x88, it means the module has not been checked and it can not use WiFi function.

Example:

wifitest.c

Header:

hfwifi.h

7. UART API

● hfuart_send

Function prototype:

int HSF_API hfuart_send(hfuart_handle_t huart,char *data,uint32_t bytes, uint32_t timeouts);

Description:

Send data to UART

Parameters:

huart: UART device, Can choose HFUART0 or HFUART1(UART 0 or UART 1)

data: The buffer of send data.

bytes: The length of send date

timeouts: Timeout, invalid value,Default to 0.

Return Value:

If success, It will return actual send data, If fail it will return error code.

Notes:

None

Examples:

None

Header file:

hfuart.h

8. TIMER API

● hftimer_start

Function prototype:

int HSF_API hftimer_start(hftimer_handle_t htimer);

Description:

Boot up a timer

Parameters:

Htimer:create by hftimer_create；

Return Value:

Return HF_SUCCESS if success, return HF_FAIL if fail;

Notes:

It is not allowed to be used in hardware interrupts；

Example:

example/timertest.c

Header file:

hftimer.h

● hftimer_create

Function prototype:

hftimer_handle_t HSF_API hftimer_create(const char *name, int32_t period, bool auto_reload,uint32_t timer_id, hf_timer_callback p_callback,uint32_t flags);

Description:

Create a timer.

Parameters:

name: the name of timer.

period: The trigger cycle of timer , Use ms as unit;

auto_reload: appoint as automatically or manually. If is true, only need to call hftimer_start once, after timer triggerred, no need to call hftimer_start again. If false, then user need to call hftimer_start again after trigger.

flags: currently can be 0 or HFTIMER_FLAG_HARDWARE_TIMER, if the created timer is a hardware timer, please set the flag as HFTIMER_FLAG_HARDWARE_TIMER.

Return Value:

if succeed, will return a pointer which point to a timer object, otherwise return NULL;

Notes:

The timer won't start until call hftimer_start when create a timer object.

Example:

example/timertest.c

Header file:

hftimer.h

● hftimer_change_period

Function prototype:

void HSF_API hftimer_change_period(hftimer_handle_t htimer,int32_t new_period);

Description:

Modify timer period.

Parameters:

htimer: Object created by hftimer_create

new_period: new period unit is ms. If create hardware timer, the unit is in us.

Return Value:

None;

Notes:

Modify the new timer period, After call the function,The timer will running in new period; It is not allowed to be used in hardware interrupts；

Example:

example/timertest.c

Header file:

hftimer.h

- ## hftimer_delete

Function prototype:

void HSF_API hftimer_delete(hftimer_handle_t htimer);

Description:

Delete a timer

Parameters:

htimer: The deleted timer created by hftimer_create

Return Value:

None

Remark:

It is not allowed to be used in hardware interrupts；

Example:

example/timertest.c

Header file:

hftimer.h

- ## hftimer_get_timer_id

Function prototype:

uint32_t HSF_API hftimer_get_timer_id(hftimer_handle_t htimer);

Description:

Get timer ID.

Parameters:

htimer: created by hftimer_create;

Return Value:

Return timer ID if success, Apoint by ftimer_creat, Return HF_FAIL if fail;

Notes:

This API is used in timer callback, to distinguish multiple timer which use the same timer callback function.

Example:

example/timertest.c

Header file:

hftimer.h

- ## hftimer_stop

Function prototype:

void HSF_API hftimer_stop(hftimer_handle_t htimer);

Description:

Stop a timer

Parameters:

htimer: created by hftimer_create；

Return Value:

None;

Notes:

After call this API, the timer stop counting unless hftimer_start is recalled;

Example:

example/timertest.c

Header file:

hftimer.h

9. MUTIL THREAD API

● hfthread_create

Function prototype:

int hfthread_create(PHFTHREAD_START_ROUTINE routine,const char * const name, uint16_t stack_depth, void *parameters, uint32_t uxpriority,hfthread_hande_t *created_thread, uint32_t *stack_buffer);

Description:

create a thread

Parameter:

routine: input parameter: entrance function of thread,

typedef void (*PHFTHREAD_START_ROUTINE)(void *);

stack_depth:Input parameter thread stack depth, depth is 4Bytes/unit, stack_size = stack_depth*4;

parameters: input parameter, thread entrance function parameter;

uxpriority: input parameter, thread priority level, HSF priority level has:

HFTHREAD_PRIORITIES_LOW: priority level low

HFTHREAD_PRIORITIES_MID: priority level middle

HFTHREAD_PRIORITIES_NORMAL: priority level normal

HFTHREAD_PRIORITIES_HIGH: priority level high

User thread usually use HFTHREAD_PRIORITIES_MID, HFTHREAD_PRIORITIES_LOW;

created_thread: optional, if function succeed, returns a   pointer to the thread creation ; if none, no returns

stack_buffer: reserve for further use

Return value:

Return HF_SUCCESS, if success, otherwise failure, please check HSF error code

Remark:

For stable, User thread recommend to use HFTHREAD_PRIORITIES_LOW and HFTHREAD_PRIORITIES_MID, It’ s better to not use HFTHREAD_PRIORITIES_NORMAL.

Example:

Example/threadtest.c

Header File:

hfthread.h

● hfthread_delay

Function prototype:

void hf_thread_delay(uint32_t ms);

Description:

Suspend current thread by ms

Parameter:

ms : Suspend time(unit is ms).

Return value:

None

Remark:

The function actually thread dormant time may have some difference with actual time.

Example:

Example/threadtest.c

Header file:

hfthread.h

- ## hfthread_destroy

Function prototype:

void hfthread_destroy(hfthread_hande_t thread);

Description:

Delete the thread created by hfthread_create

Parameter:

thread: point to deleted thread, if null, delete current thread.

Return value:

None

Remark:

When the function used for delete thread, The resource won' t release immediately;

Example:

example/threadtest.c

Header file:

hfthread.h

- ## hfthread_enable_softwatchdog

Function prototype:

int HSF_API hfthread_enable_softwatchdog(hfthread_hande_t thread,uint32_t time);

Description:

the software watchdog of enable thread

Parameter:

thread: the pointer point to thread, will return hfthread_create, this parameter can be NULL, if is NULL, enable the software watch dog of current thread

time: software watchdog overtime, unit is S

Return value:

Return HF_SUCCESS if success,, otherwise failure, please check HSF error code.

Remark:

Thread watchdog can check thread working status. if check watchdog enable, thread does not call hfthread_reset_softwatchdog in set time, The module will do soft-reset. This function can be re-called many times, can change overtime dynamically. When calling, system will reset thread software watchdog.

The thread watchdog is disabled by default. It only works when calling this thread function.

Example:

Example/threadtest.c

Header file :

hfthread.h

- **hfthread_disable_softwatchdog**

Function prototype:

int HSF_API hfthread_disable_softwatchdog(hfthread_hande_t thread);

Description:

Disable thread software watchdog

Parameter:

thread: the pointer point to thread, will return hfthread_create, the parameter can be NULL,when the return is NULL,disable the software watchdog of current thread.

Return value:

Return HF_SUCCESS if success , otherwise failure, please check HSF error code

Remark:

in the thread operation process, if one operation tooks too long time (or waiting too long time for a signal) and the time is longer than overtime, user can disable the software watchdog first, To avoid the watchdog effect cuz the long time and cause module restart; enable the watchdog after the operation finished.

Example:

Example/threadtest.c

Header file:

hfthread.h

● hfthread_reset_softwatchdog

Function prototype:

int HSF_API hfthread_reset_softwatchdog(hfthread_handle_t thread);

Description:

Restore thread software watchdog(feed dog)

Parameter:

thread: the pointer point to thread,will return hfthread_create, the parameter can be NULL, if is NULL,restore the software watchdog of current thread;

Return value:

Return HF_SUCCESS if success , otherwise failure, please check HSF error code

Remark:

After enable watchdog, thread must call this function within set time to do feed dog operation; when overtime, module will do soft reset;

Example:

Example/threadtest.c

Header file:

hfthread.h

● hfthread_mutext_new

Function prototype:

int hfthread_mutext_new(hfthread_mutex_t *mutex);

Description:

create a thread mutex

Parameter:

mutex: after successful implement the function,the return will point to created mutex;

Return value

Return HF_SUCCESS if success , otherwise failure, please check HSF error code

Remark:

when do not use created mutex, please use hfthread_mutext_free to release resource;

Example:

Example/threadtest.c

Header file:

hfthread.h

● hfthread_mutext_free

Function prototype:

void hfthread_mutext_free(hfthread_mutex_t mutex);

Description:

Destroy the thread mutex created by hfthread_mutext_new

Parameter:

mutex: point to the deleting mutex;

Return value:

No return value

Remark:

None

Example:

Example/threadtest.c

Header file:

hfthread.h

- ## hfthread_mutext_unlock

Function prototype:

void hfthread_mutext_unlock(hfthread_mutex_t mutex);

Description:

release mutex

Parameter:

mutex: point to a mutex, created by hfthread_mutext_new

Return value:

The function don’t have return value ;

Remark:

None

Example:

Example/threadtest.c

Header file:

hfthread.h

- ## hfthread_mutext_lock

Function prototype:

int hfthread_mutext_lock (hfthread_mutex_t mutex);

Description:

acquire mutex.

Parameter:

mutex: point to a mutex,created by hfthread_mutext_new

Return value:

Return HF_SUCCESS if success；otherwise return HF_FAIL if failure.

, others please refer HSF error code

Remark:

hfthread_mutext_lock and hfthread_mutex_unlock appear in pair. If call hfthread_mutex_lock, and do not call hfthread_mutex_unlock at the same time, recall hfthread_mutex_lock may occurs deadlock

Example:

Example/threadtest.c

Requests:

hfthread.h

● hfthread_mutext_trylock

Function prototype:

int hfthread_mutext_trylock(hfthread_mutex_t mutex);

Description:

Check the thread mutex locked or not

Parameter:

mutex: point to a mutex,created by hfthread_mutext_new;

Return value:

return 0 if mutext lock, otherwise mutext don’t lock.

Remark:

None

Example:

Example/threadtest.c

Header File:

hfthread.h

10. NETWORK API

● hfnet_wifi_is_active

Function prototype:

 int HSF_API hfnet_wifi_is_active(void);

Description:

Judge if module has connceted the router when working under STA mode.

Parameter:

None

Return value:

Return HF_SUCCESS if success；otherwise failure HF_FAIL, others please refer HSF error code

Notes:

Module can be allowed to create socket or other network communicaiton when it is working as STA mode and it connected to router. If no connection, lwip has not been initialized, and module don’t allowed to create socket or relative function. It can also remove this judgment, but start network communication must wait to created until HFE_DHCP_OK occur.

There is no effect on AP mode, module will skip this step.

This function is totally different with LPB100.

Example:

Header file:

hfnet.h

● hfnet_start_uart

Function prototype:

int hfnet_start_uart(uint32_t uxpriority,hfnet_callback_t p_uart_callback);

Description:

start HSF own UART serial transceiver control service;

Parameter:

uxpriority:uart service thread priority level ;please refer to hfthread_create parameter uxpriority

p_uart_callback: serial callback function, optional; if don’t need, please set as NULL, user can call when serial receive data, the description of callback function definition and parameter please refer to hfnet_start_socketa;

Return value:

Return HF_SUCCESS if success；otherwise return HF_FAIL if failure.

Remark:

When serial receiving data, if p_uart_callback is not NULL, call p_uart_callback first; if work in transparent transmit mode, send the receiving data to socketa, sockatb servie (if the two services existed), if work in command mode, pass the receiving command to command resolve program.

Under transparent transmit mode, user can realize the encryption, decryption and secondary treatment of data by the callback function and socketa, socketb service callback; under command mode, user can define AT command name and form by callback

Example:

Example/callbacktest.c

Header file:

hfnet.h

- ## hfnet_start_socketa

Function prototype:

int hfnet_start_socketa(uint32_t uxpriority,hfnet_callback_t p_callback);

Description:

Start HSF own socketa service.

Parameter:

uxpriority:socketa service priority level, please refer to hfthread_create parameter uxpriority.

p_callback: callback funtion,optional, if don’t need callback, set as NULL,Pls touch off when socketa service receiving data or status have some change;

int socketa_recv_callback_t(uint32_t event,void *data,uint32_t len,uint32_t buf_len);

event:ID ;

data: ponint to the buffer of receive date, User can modifly the buffer value by callback fuction; When working on UDP mode, The data+len behind 6 bytes as send port 4Bytes ip address and 2 Bytes port number, If the socketa working on TCP server port mode ; data+len behind 4 个 Bytes as customer port cid, Through hfnet_socketa_get_client or detailed info.

len: The length of receive data;

buf_len:data point to buffer actual length, The value greater than or equal to len;

Callback funtion return value, It’s for the length of customer handled data, If user don’t do any modify to the data, Just read, The return value will equal to len;

Return value:

Return HF_SUCCESS if success；otherwise return HF_FAIL if failure.

, others please refer HSF error code

Example:

Please refer to hfnet_start_socketb

Requests:

hfnet.h

- **hfnet_start_socketb**

Function prototype:

int hfnet_start_socketb(uint32_t uxpriority,hfnet_callback_t p_callback);

Description:

Start HSF own socketb service。

Parameter:

uxpriority:socketb service priority level, please refer to hfthread_create parameter uxpriority.

p_callback:optional, if don’t need callback NullL,Pls refer to hfnet_start_socketa.

Return Value:

Return HF_SUCCESS if success；otherwise return HF_FAIL if failure.

, others please refer HSF error code

Remark:

None

Example:

callbacktest.c

Header file:

hfnet.h

- **hfnet_ping**

Function prototype:

int hfnet_ping(const char* ip_address);

Description:

send ping package to target address, check if the address arrived or not.

Parameter:

ip_address: check the character string of target IP address, address format is xxx.xxx.xxx.xxx, if need ping domain name, please call hfnet_gethostbyname to get IP address;

Return value:

Return HF_SUCCESS if success；otherwise return HF_FAIL if failure.

, others please refer HSF error code

Remark:

if network disconnect, DNS server set error or the check info both will cause failure.

Example:

Requests:

hfnet.h

- ## hfnet_gethostbyname

Function prototype:

int hfnet_gethostbyname(const char *name, ip_addr_t *addr);

Description:

Get the domain name IP address.

Parameter:

name: Domain name.

addr: IP address

Return value:

Return HF_SUCCESS if success； otherwise return HF_FAIL if failure.

, others please refer HSF error code.

Remark:

Example:

Requests:

hfnet.h

- ## hfnet_start_httpd

Function prototype:

int hfnet_start_httpd(uint32_t uxpriority);

Description:

start httpd, a small web server

Parameter:

uxpriority: httpd service priority level ,please refer to hfthread_create parameter uxpriority;

Return value:

Return HF_SUCCESS if success； otherwise return HF_FAIL if failure.

, others please refer HSF error code.

Remark:

If the application requires to support web interface. Please call this function when start.

Example:

Header file:

hfnet.h

● hfnet_httpd_set_get_nvram_callback

Function prototype:

```
 void HSF_API hfnet_httpd_set_get_nvram_callback(
         hfhttpd_nvset_callback_t p_set,
         hfhttpd_nvget_callback_t p_get);
```

Description:

set webserver to get set module parameter callback。

Parameter:

p_set：optional parameter， If don’t need extend WEB setting parameter interface, Please set as NULL, Otherwise point to entrance function of setting.

The type of setting call funtion：

int hfhttpd_nvset_callback(char * cfg_name,int name_len,char* value,int val_len);

cfg_name is the name of configure，name_len is the length of cfg_name，valueis the valye of configure，val_len is the length of value；

p_get: optional parameter，If don’t need extend WEB setting parameter interface，Please set as NULL, Otherwise point to entrance function of setting；

The call funtion type of get parameter：

int hfhttpd_nvget_callback(char *cfg_name,int name_len,char *value,int val_len);
cfg_name is the name of get parameter, Attention: cfg_name not always contain end of string,, name_len:cfg_name length ,value: save cfg_name configure value,val_len:value correpending length；

Return value:

None

Remark:

Example:

Header file:

hfnet.h

● hfnet_socketa_send

Function prototype:

```
int hfnet_socketa_send(char *data,uint32_t len,uint32_t timeouts);
```

Description:

send data to SOCKETA

Parameter:

data: buffer area for reserve sending data

len: the length of sending buffer

timeouts: send timeout, not available currently

Return value:

Will return the actual send data length If success, Otherwise it will return error code

Remark:

Example:

Header file:

hfnet.h

● hfnet_socketb_send

Function prototype:

```
int    HSF_API hfnet_socketb_send(
                char *data,
                uint32_t len,
                uint32_t timeouts)
```

Description:

send data to SOCKETB(AT+SOCKB)

Parameter:

data: buffer area for reserve sending data

len: the length of sending buffer;

timeouts: send timeout, not   available currently

Return value:

If succeed, feedback the length of actual sending data, otherwise feedback error code

Requests:

hfnet.h

● hfnet_socketa_fd

Function prototype:

```
int HSF_API hfnet_socketa_fd(void);
```

Description:

Get socketa(AT+NETP) socket fd.

Parameter:

None

Return value:

Return socketa fd, otherwise return < 0.

Remark:

If work at TCP Server mode, the return is the listen server socket fd.

Example:

netcallback

Requests:

hfnet.h

● hfnet_socketa_get_client

Function prototype:

int HSF_API hfnet_socketa_get_client(int cid,phfnet_socketa_client_t p_client);

Description:

Get TCP client information when socketa(AT+NETP) socket working at TCP Server.

Parameter:

cid: Client ID, 0~4.

p_client, can not be NULL, point to the client information.

Return value:

Return HF_SUCCESS if client information is set to p_client, otherwise return fail.

Remark:

Socketa must working at TCP Server mode.

Example:

netcallback

Requests:

hfnet.h

● hfnet_socketb_fd

Function prototype:

int HSF_API hfnet_socketb_fd(void);

Description:

Get socketb(AT+SOCKB) socket fd.

Parameter:

None

Return value:

Return socketb fd, otherwise return < 0.

Example:

netcallback

Requests:

hfnet.h

● hfnet_socketa_close_client_by_fd

Function prototype:

int HSF_API hfnet_socketa_close_client_by_fd(int sockfd)

Description:

Close client socket fd connected to socketa.

Return value:

if succeed, feedback HF_SUCCESS， HF_FAIL means failed

Requests:

hfnet.h

- ## Standard socket　API

HSF apply lwip protocol stack ,compatible with standard socket interface, such as socket, recv, select,sendto,ioct. If source code apply standard socket function, user just need to import head file hsf.hand hfnet.h. the use of standard socket please refer to relevant Manuel.

11. SYSTEM FUNCTION

● hfmem_free

Function prototype:

void HSF_API hfmem_free(void *pv);

Description:

Free the memory allocated by hfmem_malloc

Parameters:

pv: point to the address of releasing memory.

Return Value:

None

Notes:

the function is thread-safe, if multiple thread apply this function, don't use the "free" in libc, it is non-thread-safe function

Examples:

None

Header file:

hfsys.h

● hfmem_malloc

Function prototype:

void *hfmem_malloc(size_t size)

Parameters:

Allocate memory

Parameters:

size: memory size

Return Value:

If NULL,means system has no free memory, if succeed, feedback the memory address;

Notes:

the function is thread-safe, if multiple thread apply this function, do not use malloc in libc, it is non-thread-safe function,in LPB100 series, call the meomry management function in libc is not successful.Do not call libc malloc.

Header file:

hfsys.h

● hfmem_realloc

Function prototype:

void HSF_API *hfmem_realloc(void *pv,size_t size)；

Description:

Reallocate RAM resource

Parameters:

pv:point to the previous allocated address by hfmem_malloc;

size:the size of reallocated memory

Return Value:

None

Notes:

refer to libc realloc, user can not call realloc function directly, but only use the API.

Examples:

None

Header file:

hfsys.h

● hfsys_get_memory

Function prototype:

uint32_t HSF_API hfsys_get_memory(void)；

Description:

Query the unused IRAM size left in byte.

Notes:

This API query for the IRAM size left(about 57KB left), not include the DRAM part(10KB left, global variable is saved in DRAM). If too much global variable is defined, it will occupy more than 10KB. Please change to use hfmem_malloc to allocate these

Header file:

hfsys.h

● hfsys_get_reset_reason

Function prototype:

uint32_t HSF_API hfsys_get_reset_reason (void);

Description:

Get module reboot reason.

Parameters:

None

Return Value:

REturn reboot reason. It can be the following one or more..

HFSYS_RESET_REASON_NORMAL	Caused by power on/off

HFSYS_RESET_REASON_ERESET	Caused by hardware watchdog or external reset PIN
HFSYS_RESET_REASON_IRESET0	Caused by hfsys_softreset API（Software watchdog reset, RAM access error will all call this API）
HFSYS_RESET_REASON_IRESET1	Caused by hfsys_reset API
HFSYS_RESET_REASON_WPS	Caused by WPS start(Reserved)
HFSYS_RESET_REASON_SMARTLINK_START	Caused by Smartlink start
HFSYS_RESET_REASON_SMARTLINK_OK	Caused by Smartlink finished
HFSYS_RESET_REASON_WPS_OK	Caused by WPS finished.

Notes:

Usually call this to do special operation due to different reboot reason.

Examples:

None

Header file:

hfsys.h

- ## hfsys_get_run_mode

Function prototype:

int hfsys_get_run_mode()

Description:

Get system run mode(AT+TMODE)

Parameters:

None

Return Value:

It can be the following mode:

```
enum HFSYS_RUN_MODE_E
{
   HFSYS_STATE_RUN_THROUGH=0,
   HFSYS_STATE_RUN_CMD=1,
   HFSYS_STATE_MAX_VALUE
};
```

Header file:

hfsys.h

- **hfsys_get_time**

Function prototype:

```
uint32_t HSF_API hfsys_get_time (void);
```

Description:

Get system running time in ms

Parameters:

None

Return Value:

Return the OS running time in ms

Notes:

None

Examples:

None

Header file:

hfsys.h

- **hfsys_nvm_read**

Function prototype:

```
int HSF_API hfsys_nvm_read(uint32_t nvm_addr, char* buf, uint32_t length);
```

Description:

Read data from NVM

Parameters:

nvm_addr: NVM address, which can be (0-99);

buf:Save the read data from NVM into buffer;

length: Sum of length and nvm_addr is less than 100;

Return Value:

Success returns HF SUCCESS, otherwise the return value is less than zero

Notes:

When the module restart or soft reset, NVM data will not be cleared. It provides 100 bytes of NVM. If module power off, the data of NVM will be cleared.

Examples:

None

Header file:

hfsys.h

- ## hfsys_nvm_write

Function prototype:

int HSF_API hfsys_nvm_write(uint32_t nvm_addr, char* buf, uint32_t length);

Description:

Write data into NVM

Parameters:

nvm_addr: NVM address, which can be (0-99);

buf: Save the read data from NVM into buffer;

length: Sum of length and nvm_addr is less than 100;

Return Value:

Success returns HF SUCCESS, otherwise the return value is less than zero.

Notes:

When the module restart or soft reset, NVM data will not be cleared. It provides 100 bytes of NVM. If module power off, the data of NVM will be cleared.

Examples:

None

Header file:

hfsys.h

- ## hfsys_register_system_event

Function prototype:

int HSF_API hfsys_register_system_event(hfsys_event_callback_t p_callback);

Description:

Register system event callback

Parameters:

p_callback: Point to the callback function when event occures.;

Return Value:

Return HF_SUCCESS if success, otherwise return HF_FAIL.

Notes:

The time consuming operation is not allowed in the callback function, the callback function should immediate return after process.The support event is as following

HFE_WIFI_STA_CONNECTED	When STA connect to AP

HFE_WIFI_STA_DISCONNECTED	When STA disconnect to AP
HFE_CONFIG_RELOAD	When reload is execute.(nReload Pin or AT+RELD)
HFE_DHCP_OK	When STA connect to AP and get DHCP IP address from AP 当 STA
HFE_SMTLK_OK	When Smartlink get AP password, the default operation is reboot, if the callback return value is not HF_SUCCESS, the module won't do reboot operation, user need to reboot manually.

Examples:

None

Header file:

hfsys.h

- ## hfsys_reload

Function prototype:

void HSF_API hfsys_reload();

Description:

Restore the parameter to factory setting

Parameters:

None

Return Value:

None

Notes:

After call this API, suggest to call the hfsys_reset to reboot the system due to some parameter is valid only after reboot.

Examples:

None

Header file:

hfsys.h

hfsys_reset

Function prototype:

void HSF_API hfsys_reset(void);

Description:

Restart system, IO status not hold.

Parameters:

None

Return Value:

None

Notes:

None

Examples:

None

Header file:

hfsys.h

hfsys_softreset

Function prototype:

void HSF_API hfsys_softreset(void);

Description:

Software reset, keep the current IO status

Parameters:

None

Return Value:

None

Notes

None

Examples:

None

Header file:

hfsys.h

hfsys_switch_run_mode

Function prototype:

int hfsys_switch_run_mode(int mode);

Description:

Switch system running mode.

Parameters:

mode: The following mode is supported.

```
enum HFSYS_RUN_MODE_E

{

        HFSYS_STATE_RUN_THROUGH=0,

        HFSYS_STATE_RUN_CMD=1

};
```

HFSYS_STATE_RUN_THROUGH:Throughput mode
HFSYS_STATE_RUN_CMD:Command mode

Return Value:

HF_SUCCESS: success, otherwise HF_FAIL

Header file:

hfsys.h

- ## hfconfig_file_data_read

Function prototype:

int hfconfig_file_data_read(int offset, unsigned char *data, int len);

Description:

Read parameter from config area.

Parameters:

offset: Prameter offset
data: Store read parameter
len: Read size

Return Value:

HF_SUCCESS: success, otherwise HF_FAIL

Header:

hfconfig.h

- ## hfconfig_file_data_write

Function prototype:

int hfconfig_file_data_write(int offset, unsigned char *data, int len);

Description:

Set parameter in config area.

Parameters:

offset: Parameter offset in config area
data: Setting parameter
len: Setting size

Return Value:

HF_SUCCESS: success, otherwise HF_FAIL

Header :

hfconfig.h

12. USER FLASH API

● hfuflash_erase_page

Function prototype:

int HSF_API hfuflash_erase_page(uint32_t addr, int pages);

Description:

Erase user flash page.

Parameters:

addr: logical address of user flash(not the flash real address).

pages : The page number need to be erased.

Return Value:

Return HF_SUCCESS if success, otherwise return HF_FAIL;

Notes:

LPT230 not supported.

The use flash is a 128KB size of flash in the reserved flash real area.

Examples:

example/uflashtest.c

Header file:

hfflash.h

● hfuflash_read

Function prototype:

int HSF_API hfuflash_read(uint32_t addr, char *data, int len);

Description:

Read data from flash.

Parameters:

addr: The logical address of flash(0- HFUFLASH_SIZE-2);

data : The received data buffer.

len : The data buffer length;

Return Value:

Return the bytes number read if success, otherwise return <0

Notes:

Examples:

example/uflashtest.c

Header file:

hfflash.h

● hfuflash_write

Function prototype:

int HSF_API hfuflash_write(uint32_t addr, char *data, int len);

Description:

Write data to flash

Parameters:

addr: The logical address of flash(0- HFUFLASH_SIZE-2)；

data : Data buffer；

len : Data buffer length;

Return Value:

the bytes number if write success, otherwise return <0;

Notes:

Need to erase the flash page if the address to be written has previous data in it.

Examples:

example/uflashtest.c

Header file:

hfflash.h

13. USER FILE API

● hffile_userbin_read

Function prototype:

int HSF_API hffile_userbin_read(uint32_t offset,char *data,int len);

Description:

Read data from user files;

Parameters:

offset: File offset;

data : Save the data from read file to buffer;

len : Size of the buffer;

Return Value:

If return value is less than zero,then it fails.Otherwise,the function returns the number of actual Byte read from the file;

Examples:

None

Header file:

hffile.h

● hffile_userbin_size

Function prototype:

int HSF_API hffile_userbin_size(void);

Description:

Read size from user bin’s file;

Parameters:

None

Return Value:

Failure is less than zero, otherwise the file size;

Notes:

None

Examples:

None

Header file:

hffile.h

● hffile_userbin_write

Function prototype:

int HSF_API hffile_userbin_write(uint32_t offset,char *data,int len);

Description:

Write the data into user file.

Parameters:

offset: File offset;

data : Save the data from read file to buffer;

len : Size of the buffer;

Return Value:

If return value is less than zero,then it fails.Otherwise,the function returns the number of actual Byte written into the file;

Notes:

A user profile is a fixed-size file, the file is stored in flash, you can save user data.User profile has backup function, so users do not need to worry about power outages during programming.If it powers off, it will automatically revert to the content before.

Examples:

None

Header file:

hffile.h

- **hffile_userbin_zero**

Function prototype:

int HSF_API hffile_userbin_zero (void);

Description:

Quickly clear the content of the entire file.

Parameters:

None

Return Value:

Failure is less than zero, otherwise the file size;

Notes:

Calling this function can quickly clear up the entire contnet of file, faster than hffile_userbin_write

Examples:

None

Header file:

hffile.h

14. OTA-UPGRADE API

The module OTA firmware should be download into the OTA upgrade back area via the following area. if check the firmware is correct, it set flag to indicate do upgrade operation in bootloader. If next reboot see this flag, bootloader will copy the OTA upgrade backup file to the CODE running area and clear the flag.
Flash mapping see SDK manual

● hfupdate_start

Function prototype:

int hfupdate_start(HFUPDATE_TYPE_E type);

Description:

Start to upgrade, erase the OTA upgrade backup area.

Parameters:

type: Upgrade type

```
typedef enum HFUPDATE_TYPE
{
   HFUPDATE_SW=0,      //OTA upgrade backup area
}HFUPDATE_TYPE_E;
```

Return Value:

Return the bytes number if write success, otherwise return <0;

Notes:

Call this API when start do the OTA operation, it will erase the flash OTA backup area.

Examples:

example/updatetest.c

Header file:

hfupdate.h

● hfupdate_write_file

Function prototype:

int hfupdate_write_file(HFUPDATE_TYPE_E type ,uint32_t offset,char *data,int len);

Description:

Copy the upgrade file data to upgrade backup flash area.

Parameters:

type: type, should be HFUPDATE_SW

offset: The upgrade file flash offset address, start from 0, the actual address is the OTA upgrade area.

data: the upgrade file data

len: The upgrade file length, should be less than 4K and is not allowed to write over page when call this API once.(4K for one page)

Return Value:

>=0 for success, otherwise return HF_FAIL.

Notes:

Examples:

example/updatetest.c

Header file:

hfupdate.h

- **hfupdate_complete**

Function prototype:

int hfupdate_complete(HFUPDATE_TYPE_E type,uint32_t file_total_len);

Description:

Upgrade finished

Parameters:

type: upgrade type, should be HFUPDATE_SW

file_total_len: upgrade file length

Return Value:

HF_SUCCESS if success, HF_FAIL if fail

Notes:

When the upgrade file has been download into the module, call this function to check the OTA upgrade file, if file is valid, wirte upgrade flag for the bootloader to do upgrade operation when next reboot.

Examples:

example/updatetest.c

Header file:

hfupdate.h

15. ENCRYTION AND DECRYPTION API

● hfcrypto_aes_ecb_encrypt

Function prototype:

int hfcrypto_aes_ecb_encrypt(const unsigned char *key, unsigned char *data, int data_len);

Description:

AES ECB（128Bit）encryption；

Parameters:

key: secret key

data: Data need to be encrypted. Store data after encryption.

len: Encrypted data length(encrypted as integer multiple of 16)

Return Value:

Data length of encryption

Notes:

Examples:

Header file:

hfcrypto.h

● hfcrypto_aes_ecb_decrypt

Function prototype:

int hfcrypto_aes_ecb_decrypt(const unsigned char *key, unsigned char *data, int data_len);

Description:

AES ECB（128Bit）decryption；

Parameters:

key: secret key

data: Data need to be decrypted. Store data after decryption.

len: Decrypted data length(decrypted as integer multiple of 16)

Return Value:

Data length of decryption

Notes:

Examples:

Header file:

hfcrypto.h

● hfcrypto_aes_cbc_encrypt

Function prototype:

int hfcrypto_aes_cbc_encrypt(const unsigned char *key, unsigned char *data, int data_len);

Description:

AES CBC（128Bit）decryption；

Parameters:

key: secret key

data: Data need to be decrypted. Store data after decryption.

len: Decrypted data length(decrypted as integer multiple of 16)

Return Value:

Data length of decryption

Notes:

Examples:

Header file:

hfcrypto.h

- **hfcrypto_aes_cbc_decrypt**

Function prototype:

int hfcrypto_aes_cbc_decrypt(const unsigned char *key, unsigned char *data, int data_len);

Description:

AES CBC（128Bit）decryption；

Parameters:

key: secret key

data: Data need to be decrypted. Store data after decryption.

len: Decrypted data length(decrypted as integer multiple of 16)

Return Value:

Data length of decryption

Notes:

Examples:

Header file:

hfcrypto.h