HEKR ESP8266 SDK 接口说明

v1.1.0 by zengxuefeng@hekr.me 2015/10/10 11:49:36

1-1 HekrConfig (Wi-Fi一键配置)

```
void start_hekr_config(hekr_config_event_cb_t event_cb, size_t timeout)
```

参数

- `event_cb` HekrConfig事件回调函数
- `timeout` 配置超时时间 (单位：`毫秒`)

返回值

- `无`

头文件

- `#include <module_wifi.h>`

参数类型定义

```
typedef enum
{
    EVENT_CONNECT_WIFI_FAIL = 0,    //配置失败，未能连接上Wi-Fi
    EVENT_HEKR_CONFIG_TIMEOUT,      //配置超时
    EVENT_HEKR_CONFIG_FINISH        //配置成功，将会自动去连接服务器
}config_event_t;

typedef void( *hekr_config_event_cb_t)(config_event_t event);
```

说明

- 上电的时候，如果FLASH中没有Wi-Fi配置信息会自动进入配置模式。

1-2 取消Wi-Fi 一键配置

```
void stop_hekr_config(void)
```

参数

- `无`

返回值

- `无`

1-3 获取设备状态

```
uint8 device_status_get(device_status_type_t item)
```

参数

- `item` 设备状态项

返回值

- `1` 设备状态项值为1
- `0` 设备状态项值为0

头文件

- `include<device_status.h>`

参数类型定义

```
typedef enum
{
    DEVICE_WLAN_CONNECTED = 0,      //设备连接上wifi
    DEVICE_WLAN_CONNECTING =2,      //正在连接wifi
    GOT_SERVER_IP =6,               //通过DNS服务成功获取的服务器IP
    LOGGED_IN_SERVER =7,            //成功登入服务器
    HEKR_CONFIG_RUNNING =11,        //一件配置模式正在运行
    SOFTAP_CONFIG_RUNNING =13,      //热点配置模式正在运行
    LOG_PRINT_ENABLE =16            //LOG输出开启
}device_status_type_t;
```

1-4 设置设备状态指示灯

```
void device_status_led_task_install(uint32_t pin, uint32_t reverse)
```

参数

- `pin` 设备状态指示引脚led1
- `reverse` 设置需要取反的引脚。通常不需要设置，即值为0

返回值

- `无`

头文件

- `include<device_status.h>`

说明

- 该函数启用led引脚用来指示设备当前状态，不同状态时led闪烁频率不一致。

状态灯：

- 一件配置模式时 亮1.5s灭1.5s
- 服务器连接正常 5s间隔闪烁
- 设置登录服务器失败 1s间隔闪烁
- DNS解析失败 0.5s间隔闪烁
- 未连接Wi-Fi 指示灯长亮

2-1 连接服务器

```
void connect_server(cloud_conn_event_cb_t cb)
```

参数

- `cb` 与服务器连接事件回调函数

返回值

- `无`

头文件

- `#include <module_wifi.h>`

参数类型定义

```
typedef enum {
    CLOUD_EVEN_WIFI_DISCONNED = 0,      //Wi-Fi未连接
    CLOUD_EVEN_NO_ACCESSKEY,            //设备没有accesskey
    CLOUD_EVEN_DNS_ERROR,               //无法获取服务器ip
    CLOUD_EVEN_DEVICE_LOGIN_ERROR,      //设备登入服务器失败
    CLOUD_EVEN_DEVICE_LOGIN_SUCCESS,    //设备登入服务器成功
    CLOUD_EVEN_DISCONNED_FROM_CLOUD,    //设备与服务器连接断开
    CLOUD_EVEN_CONNECT_CLOUD_DISABLE    //设备连接服务器已被禁用
}cloud_conn_event_t;

typedef void(*cloud_conn_event_cb_t)(cloud_conn_event_t event);
```

2-2 断开与服务器的连接

```
void disconnect_from_server(void)
```

参数

- `无`

返回值

- `无`

2-3 向远程终端发送消息

```
uint8_t send_message_to_remote(char *tid, void *data, size_t size)
```

说明

- 给远端终端单播或者组播消息，当`tid==NULL`时为组播。
- 目前只支持组播

参数

- `tid` 终端ID号
- `data` 待发送的数据
- `size` 数据的大小 （单位：`字节`）

返回值

- `1` 成功
- `0` 失败

3-1 设备升级

```
bool start_update(char *URL, char *MD5, char firmware_type)
```

参数

- `URL` 待升级固件的http地址
- `MD5` 固件的MD5校验码
- `firmware_type` 固件类型 ('A'和'B')

返回值

- `无`

头文件

- `<module_upgrade.h>`

说明

- 如需使用APP进行OTA请到 Hekr数据管理中心 上传固件

示例

```
start_update("http://192.168.1.22/firmware1.bin","d41d8cd98f00b204e9800998ecf8427e",'A');
```

4-1 注册按键中断

```
uint8_t register_key_intrrupt_handle
(
    size_t pin,
    GPIO_INT_TYPE intr_state,
    size_t long_press_time,
    callbcak_t *short_press_handle,
    callbcak_t *long_press_handle)
```

参数

- `pin` 按键引脚
- `intr_state` 中断类型
- `long_press_time` 长按所需时间，单位为ms
- `short_press_handle` 短按回调函数
- `long_press_handle` 长按回调函数

返回值

- `1` 注册成功
- `0` 注册失败

头文件

- `#include <module_key.h>`

示例

```
register_key_intrrupt_handle
    (
        13,
        GPIO_PIN_INTR_NEGEDGE,
        3000,
```

```
        (callbcak_handle_t *)&plug_power_change,
        (callbcak_handle_t *)&wifi_config_reset
    );
```

- 说明：注册`GPIO13`引脚拉低`3000ms`执行`wifi_config_reset()`,短于`3000ms`执行`plug_power_change()`

参数类型定义

```
typedef enum {
    GPIO_PIN_INTR_DISABLE = 0,  //中断禁止
    GPIO_PIN_INTR_POSEDGE = 1,  //上升沿触发
    GPIO_PIN_INTR_NEGEDGE = 2,  //下降沿触发
    GPIO_PIN_INTR_ANYEGDE = 3,  //上升沿或下降沿触发
    GPIO_PIN_INTR_LOLEVEL = 4,  //低电平触发
    GPIO_PIN_INTR_HILEVEL = 5   //高电平触发
} GPIO_INT_TYPE;

typedef void (callbcak_handle_t)(vcboid *arg);
```

5-1获取SDK版本号

```
char *get_hekr_sdk_version(void)
```

参数

- `无`

返回值

- 类型 `字符串`
- 值 `版本号`

头文件

- `#include <device_info.h>`

5-2 系统日志

```
void system_log_set(log_port_t port)
```

参数

- `port` 设置输出端口，可选参数`PORT_UART0` 、`PORT_UART1` 、`PORT_NULL`

返回值

- `无`

参数类型定义

```
typedef enum
{
    PORT_UART0 = 0, //通过uart0输出
    PORT_UART1,     //通过uart1输出
    PORT_NULL       //不输出
}log_port_t;
```

头文件

- `#include <log.h>`

说明

- 日志输出会占用资源，生成固件建议关闭

6-1 用户函数执行入口

```
void hekr_main(void)
```

说明

- 该函数在设备启动过程执行，初始化的函数放在此处执行。
- 网络、Wi-Fi相关的函数请放在系统初始化完成后执行

示例

```
void hekr_main(void)
{
    /*通过串口GPIO2输出信息*/
    /*波特率9600 8位数据位 1位停止位*/
    uart_init(BIT_RATE_9600, BIT_RATE_9600);
    os_printf("\n\nsystem run !! \n\n");
    os_printf("hekr sdk ver:%s\n", get_hekr_sdk_version());
}
```

参数

- `无`

返回值

- `无`

头文件

- `无` 该函数用户自己定义

6-2 注册系统初始化完成后的回调函数

```
void register_hekr_system_init_done_callback(hekr_system_init_done_cb_t handle)
```

说明

- 此函数代替`system_init_done_cb` 用户请不要再使用`system_init_done_cb`

参数

- `handle` 回调函数

返回值

- `无`

头文件

- `#inlcude <sys.h>`

备注

备注

基于ESP8266 SDK 1.4.0开发， ESP8266 通用API请参考 ESP官方文档