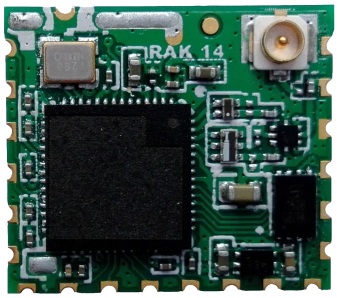
RAK439 API Library
用户手册 V1.1



深圳市瑞科慧联科技有限公司 www.rakwireless.com

邮箱：info@rakwireless.com

目 录

1. SDK 库概述

RAK439 模组是一款超低功耗、低成本、全面支持主流以及 WAPI 加密模式、支持 802.11b/g/n 的 SPI 接口的 Wi-Fi®模组。模组内部集成射频基站、巴仑、射频开关、晶体振荡器、电源变换电路，外围只需要很少的电路就能完成硬件设计。模块内置 IPEX 外置天线座，同时也引出天线脚，可以进行板载天线的设计。

RAK439 SDK 软件库，封装了对 WIFI 模块的接口驱动，提供了平台相关的驱动函数，客户将比较容易移植到自己的平台，软件库也提供了网络相关的 API，可以快速完成模块的网络配置和连接，socket 部分使用类 BSD socket 操作方式，操作方便。

软件库分为不跑系统和跑系统两种，用户可以根据自己平台的实际情况，灵活选择。

2. API Library 介绍

这一章节介绍 API 的数据结构和 API 的使用方法。

2.1 API 的数据结构

这一部分介绍 API 中重要的几个数据结构体，rw_lib.h 中包含驱动库的驱动函数声明，软件库配置数据结构，网络部分的数据结构，回调函数的类型声明。

2.1.1 HAL 驱动函数类型

硬件相关的驱动函数，是与主机平台相关的硬件实现，详见 **RAK439 驱动移植手册。**

2.1.2 LIB 配置数据结构

该配置结构可以对软件库占用资源大小和 WIFI 模块工作参数进行设置，及初始化驱动函数，回调函数等，用户可以根据需求，自行设置。

```
typedef struct
{
    bool    spi_int_enable;    // customer can choose enable or disenable spi int event driver
    uint8_t rx_queue_num;         // rx buffer queue num >= 1
    uint8_t socket_max_num;       // module support socket numbers max 8
    uint8_t scan_max_num;         // scan result buffer numbers normal : 10 if you need more
                                  // can raise it
    uint8_t  tcp_retry_num;      // tcp backoff retry numbers
    char*    host_name;           // module host name ,you can see it in router clients when
                                  // DHCP success
    char*    country_code;        // set module country code ,CN (1-13),JS(1-14),UP(1-11)
    struct   driver_cb_ driver_cb; // platform related driver used
    struct   app_cb_ app_cb;       // application related callback info
```

```
}rw_DriverParams_t;
```

参数说明：

结构体成员	成员类型	成员描述
spi_int_enable	bool	是否使用模块的 INT 中断脚功能，如果禁用则主机不需要开启检测外部中断功能，驱动使用轮询接收模块至主机的数据。 NOS 时，此选项为可选 。 OS 时 ，此选项必须设为 true。不支持轮询方式。
rx_queue_num	uint8_t	RX buffer 的个数>=1，一个 RX buffer 的大小为 1664 字节，RAK439 加载驱动（执行 rw_sysDriverInit）的时候从堆里分配，卸载驱动（执行 rw_sysDriverDeinit）的时候释放。适当增加接收 buffer 的个数可以提高模块双向数据时的吞吐率
socket_max_num	uint8_t	Socket 的个数，最大8个，减少个数可以减少驱动库的资源占用
scan_max_num	uint8_t	扫描到网络最大个数设置，驱动库会根据设置的个数申请空间。不宜设置太小，当不指定 SSID 扫描时，如果该 SSID 信号较差，扫描列表按信号值排序，可能会被丢掉
tcp_retry_num	uint8_t	该系统参数，用来设置 TCP 发送失败后的重试次数，不宜设置太小，会导致 TCP 连接在拥堵情况下容易断开。也不宜设置太长，会导致 TCP 断开事件长时间不通知。默认为5次。
host_name	char*	模块的主机名，dhcpclient 时设置，可以在路由中显示
country_code	char*	模块的无线工作国家代码，设置相应的国家，模块将在对应国家指定的信道内工作 CN (1-13)，JP(1-14)，US(1-11)
driver_cb	driver_cb_	驱动库的硬件相关驱动函数
app_cb	app_cb_	驱动库的应用相关的回调函数

2.1.3 网络信息结构体

扫描时返回网络的信息结构，包括扫描到的个数，及每个网络的名称，信道，加密方式等信息。

```
typedef struct {
    uint8_t           channel;
    uint8_t           ssid_len;
    uint8_t           rssi;
    uint8_t           bssid[RW_BSSID_LEN];
    uint8_t           ssid[RW_MAX_SSID_LEN + 1];
    uint8_t           sec_mode;
    uint8_t           auth_mode;
}rw_WlanNetworkInfo_t;

typedef struct {
    int32_t num;
    rw_WlanNetworkInfo_t*  WlanNetworkInfo;
}rw_WlanNetworkInfoList_t;
```

参数说明：

结构体成员	成员类型	成员描述
num	int32	扫描到的有效网络的个数，最大为16个
rw_WlanNetworkInfo_t.channel	uint8_t	当前网络所在的信道， 1-14
rw_WlanNetworkInfo_t.ssid_len	uint8_t	当前网络的名称，字符串的长度，最大32B
rw_WlanNetworkInfo_t.rssi	uint8_t	当前网络的信号强度，如值为50，表示信号强度为 -50dbm
rw_WlanNetworkInfo_t.bssid	uint8_t	当前网络的 bssid,一般为路由的 MAC 地址，长度为6B
rw_WlanNetworkInfo_t.ssid	uint8_t	当前网络的名称，字符串，最大32B
rw_WlanNetworkInfo_t.sec_mode	uint8_t	当前网络是否加密，0：开放 1：加密
rw_WlanNetworkInfo_t.auth_mode	uint8_t	当前网络的加密方式，详见下列 rw_AuthMode_t 类型

2.1.4 连接/创建网络结构体

连接网络时传入的结构体，包含需要连接的网络的名称，是否加密，加密方式密码等。还包含一些高级参数，如可以加入指定路由（设置 bssid,区分相同 SSID 存在时，漫游网络），如果不需要指定，请将该项设为 NULL。可以使用 PMK 密钥进行联网，可以加快连接网络的速度，如果不需要请将该项设为 NULL。如果不确定当前路由的加密方式，可以将

auth_mode 设为 auto。创建网络 AP 时，只需要指定 AP 的名称，信道等。

```
typedef struct {

    rw_WlanMode_t      role_mode;
    uint8_t            channel;   // used to point which channel to connect or creat
    rw_SecMode_t       sec_mode;
    rw_AuthMode_t      auth_mode;
    char*              psk;
    char*              pmk;     //32B hex，64B ascii ,used for auth_mode WPA,WPA2,
                               //NULL:  don't used
    char*              ssid;    // can not be NULL
    uint8_t*           bssid;   //NULL: don't used
}rw_WlanConnect_t;
```

参数说明：

结构体成员	成员类型	成员描述
role_mode	rw_WlanMode_t	网络模式，目前支持 ROLE_STA ，ROLE_AP
channel	uint8_t	连接或创建网络所在的信道， 1-14 连接时可以不指定信道 ，设为0
sec_mode	rw_SecMode_t	网络是否加密，必须设置项，RW_SEC_TYPE_OPEN，RW_SEC_TYPE_SEC
auth_mode	rw_AuthMode_t	网络的加密方式。连接时 如果不确定当前路由的加密方式，可以设为 auto，AP 模式下，只支持 RW_AUTH_TYPE_WPA2_PSK_AES
psk	char*	当前网络的密码，字符串，最大64B，使用 PMK 时，可以设为 NULL
pmk	char*	当前网络的 PMK 密码，固定32B，使用 psk 时，可以设为 NULL
ssid	char*	当前网络的名称，字符串，最大32B
bssid	uint8_t*	当前网络的 bssid，一般为路由的 MAC 地址，长度为6B，不需要指定加入时，需要设为 NULL

2.1.5 IP 参数结构体

设置 IP 参数，有几种方式，静态配置，DHCP 动态配置。AP 模式时，静态设置 IP，并开启 DHCPsever，为加入的无线客户端分配 IP。IP 参数包含 IP 地址，子网掩码，网关和 DSN 服务器地址等。

```
typedef enum
{
    IP_CONFIG_STATIC = 0,
    IP_CONFIG_QUERY,
    DHCP_CLIENT,
    DHCP_SERVER
}rw_IpConfigMode_t;
typedef struct {
    uint32_t          addr;
    uint32_t          mask;
    uint32_t          gw;
    uint32_t          svr1;
    uint32_t          svr2;
}rw_IpConfig_t;
```

参数说明：

结构体成员	成员类型	成员描述
	rw_IpConfigMode_t	IP 设置的方式：IP_CONFIG_STATIC = 0, IP_CONFIG_QUERY, DHCP_CLIENT,DHCP_SERVER
rw_IpConfig_t.addr	uint32_t	IPV4 参数 IP 地址
rw_IpConfig_t.mask	uint32_t	IPV4 参数 子网掩码
rw_IpConfig_t.gw	uint32_t	IPV4 参数 IP 网关地址
rw_IpConfig_t.svr1	uint32_t	IPV4 参数 DNS 服务器地址首选
rw_IpConfig_t.svr2	uint32_t	IPV4 参数 DNS 服务器地址备选

2.1.6 WPS EASY 配置返回结构体

模块支持 WPS2.0和 Easyconfig 一键配置联网，返回需要连接路由的相关信息，如必需的 ssid，密码等。用户可以将必要的参数设置到联网参数中，就可以连接上指定的路由。

```
typedef struct {
    uint8_t           bssid[RW_BSSID_LEN];
    uint8_t           ssid[RW_MAX_SSID_LEN];
    uint8_t           psk[RW_MAX_PASSPHRASE_SIZE];
    uint8_t           channel;
}rw_WlanEasyConfigWpsResponse_t;
```

参数说明：

结构体成员	成员类型	成员描述
bssid	uint8_t	连接当前网络的 bssid，一般为路由的 MAC 地址，长度为 6B
ssid	uint8_t	连接当前网络的 ssid，字符串，最大32B
psk	uint8_t	当前网络的密码，字符串，最大64B
channel	uint8_t	当前网络所在的信道 1-14

2.2 API 驱动函数

API 函数包含了驱动初始化，网络配置，网络连接，查询类，socket 操作函数等。网络部分在 rw_lib.h 中，socket 部分在 rw_socket.h 中。

2.2.1 驱动初始化

```
int rw_sysDriverInit(rw_DriverParams_t* params);
```

参数：

[in] rw_DriverParams_t* params --驱动参数

返回：

[out]

```
RW_ERR_INIT_DRIVER_FAILD      -- initialize driver failed
RW_ERR_PARAM_INVAILD          -- parameter  invaild please check
RW_ERR                        -- cmd execute failed
RW_OK                         -- cmd execute success
```

说明：

驱动库的基本设置，驱动函数，回调函数设置，及驱动初始化，如果返回驱动初始化失败，请检查 SPI 等连接。该 API 需要第一个被调用，成功初始化后才能执行其他的 API 函数。

2.2.2 扫描网络

```
int rw_wlanNetworkScan(char* pssid, int channel);
```

参数：

```
[in]    pssid          -- 是否指定 ssid 不指定时传入 NULL
        Channel        -- 扫描指定信道，如要扫描全信道 设为 0
```

返回：

```
[out]
RW_ERR_INVAILD_CHANNEL      -- channel parameter invaild
RW_ERR_INVAILD_SSID         -- ssid parameter invaild
RW_ERR                      -- cmd execute failed
RW_OK                       -- cmd execute success
```

说明：

该命令用于扫描网络 AP 信息，可以扫描指定名称的网络或者指定信道扫描，不使用带操作系统的软件库时，扫描结果将以回调函数 rw_WlanNetworkInfoList_t*返回出来，扫描时驱动库会根据设置的最大扫描个数，申请空间，将扫描到的 AP 信息，按信号强度排列，由强到弱。扫描信息使用完后 应当释放该部分空间 ，避免空间重复申请，浪费资源。使用 OS 的软件库时，扫描结果可以用 rw_wlanGetScanInfo 函数直接获取 ，同样需要释放资源。

2.2.3 获取扫描结果

```
int rw_wlanGetScanInfo(rw_WlanNetworkInfoList_t *pInfoList);
```

参数：

```
[in]  pInfoList          -- 指向存放扫描结果指针
```

返回：

```
[out]
```

RW_ERR -- cmd execute failed
RW_OK -- cmd execute success

说明：

该函数在带 OS 的软件库时使用，函数会堵塞至内部扫描结果的返回。超时时间：5s。获取的扫描结果存放在 rw_WlanNetworkInfoList_t 类型的指针 pInfoList 中，处理完结果后，注意释放内存。

2.2.4 AP 配置

```
int rw_wlanApConfig(uint8_t is_hidden);
```

参数：

[in] is_hidden -- 1： 隐藏 0： 不隐藏

返回：

[out]
RW_ERR -- cmd execute failed
RW_OK -- cmd execute success

说明：

该命令用于设置 AP 模式时，是否隐藏 AP 的名称，防止其他无线客户端的加入。该命令需要在 rw_wlanConnect 命令前，进行调用。

2.2.5 连接/创建网络

```
int rw_wlanConnect(const rw_WlanConnect_t *conn);
```

参数：

[in] conn -- 连接/创建网络时的网络参数，网络的名称，密码等

返回：

[out]
RW_ERR_INVAILD_ROLE_MODE -- role mode parameter invaild
RW_ERR_INVAILD_SSID -- ssid parameter invaild
RW_ERR_INVAILD_CHANNEL -- channel parameter invaild
RW_ERR_INVAILD_SEC_MODE -- sec_mode parameter invaild
RW_ERR_INVAILD_AUTH_MODE -- auth_mode parameter invaild
RW_ERR -- cmd execute failed
RW_OK -- cmd execute success

说明：

连接/创建网络，作为 STA 时连接路由器，需要设置路由器的相关参数，rw_WlanConnect_t 结构体中包含了连接是需要的参数。创建 AP 时，也需要设置基础的参数，

网络名称，信道，密码，是否加密，加密方式，AP 加密方式只支持 WPA2-AES。 结构体具体参数详见参数部分。

2.2.6 IP 参数设置

```
int rw_ipConfig(rw_IpConfig_t* ip_addr, rw_IpConfigMode_t mode);
```

参数：

[in] ip_addr -- IP 地址，网关等信息
[in] mode -- IP 设置/查询模式,ip 静态设置,dhcp client,dhcpserver

返回：

[out]
RW_ERR_IP_DHCP -- ipdhcp fail or timeout
RW_ERR -- cmd execute failed
RW_OK -- cmd execute success

说明：

IP 设置命令，连接/创建网络成功后，需要调用该命令来设置模块的 IP 信息，可以使用静态配置（需要注意和路由器的网段一致），也可以使用动态分配的方式从路由器获取 IP 信息 成功失败都会给出回调通知。AP 模式时需要先设置静态模块的 IP 信息，然后再开启模块的 DHCPserver 功能，给无线客户端分配 IP。命令示例：

```
rw_network_init(&conn, IP_CONFIG_STATIC, &ipconfig);
```

2.2.7 DNS 域名解析

```
int rw_dnsRequest(const char *host_name, uint8_t name_len, uint16_t
family);
```

参数：

[in] host_name -- 需要解析的域名
name_len -- 域名的长度
family --使用的协议族 目前支持 AF_INET IPv4

返回：

[out]
RW_ERR -- cmd execute failed
RW_OK -- cmd execute success

说明：

DNS 域名解析命令是用来解析给定域名到一个 IP 地址，命令调用需要在模块作为 STA，已经联网并获得了 IP 地址和 DNS 服务器地址（一般为网关地址），静态设置模块 IP 时需要设

置可用的 DNS 服务器地址。该命令是非阻塞的，DNS 结果将在回调函数中输出。详见 DNS 回调函数介绍。BSD socket 中有类似函数：int gethostbyname(const char *hostname ,uint8_t namelen ,uint32_t* out_ipaddr,uint16_t family); 该函数为堵塞调用，直接返回 IP 地址。

2.2.8 WPS EASY 配置网络

```
int rw_startEasyWps(rw_ConfigMode_t mode);
```

参数：

[in] mode -- rw_ConfigMode_t 类型，CONFIG_EASY，CONFIG_WPS 模式

返回：

[out]

RW_ERR_INVAILD_CONFIG_MODE	-- config mode parameter invaild
RW_ERR	-- cmd execute failed
RW_OK	-- cmd execute success

说明：

WPS，EASYconfig 是两种一键配置网络参数的命令，模块正常启动后 就可以执行该命令，easyconfig 配置手机 APP，获取到需要连接路由的名称和密码等信息，软件库通过回调函数就配置结果告知上层。WPS 操作 ，需要按下路由的 WPS 按键。获取到名称密码后，用户可以保存至 Flash，下次上电就可以自动连接。

2.2.9 设置功耗模式

```
int rw_setPwrMode(rw_PowerMode_t pwr_mode);
```

参数：

[in] pwr_mode -- rw_PowerMode_t 类型，POWER_MAX，POWER_SAVE 模式

返回：

[out]

RW_ERR_INVAILD_CONFIG_MODE	-- config mode parameter invaild
RW_ERR	-- cmd execute failed
RW_OK	-- cmd execute success

说明：

设置模块的功耗模式，仅 STA 模式下有效。power_max 模块会全速运行，性能最好。设置节省功耗模式时，模块成功连接路由后将与路由保持一个 DTIM 间隔，唤醒，接收发送数据。

2.2.10 获取 PMK

```
int rw_getPMK(char* pmk);
```

参数：

[in] pmk -- char*类型，指向 PMK 值需要填入的地址

返回：

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

说明：

获取网络 PMK 的值，仅 STA 模式下，WPA 或 WPA2加密时有效 ，PMK 的值为固定的32字节。模块在成功加入网络后，就可以获取到 PMK 的值（类似于密码），使用 PMK 代替密码加入网络，可以节省大约1.5S 的时间。在某些需要快速连接网络的场景 可以使用该密码来连接网络。

2.2.11 获取信号强度

```
int rw_getRSSI(void);
```

参数：

[in] NULL

返回：

[out]

RW_ERR -- cmd execute failed

>0 -- rssi value

说明：

获取与当前路由器连接的信息强度，STA 模式下有效，函数返回值为正值（最大为96），如返回50，表示 RSSI 的值 为-50， 值越小表示信号强度越强 。

2.2.12 获取信号强度

```
int rw_getMacAddr(char* mac_data);
```

参数：

[in] mac_data -- char*类型，指向 MAC 值需要填入的地址

返回：

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

说明：

模块的 MAC 在出厂前已经扫写了全球唯一的 MAC 地址，6 个字节，作为模块硬件表识。

2.2.13 获取软件版本

```
int rw_getLibVersion(char* version);
```

参数：

[in] version -- char*类型，指向版本值需要填入的地址

返回：

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

说明：

获取软件库及 WIFI 的版本 如1.0.4-2.1.39 .

2.2.14 反初始化

```
int rw_sysDriverDeinit(void);
```

参数：

[in] NULL

返回：

[out]

RW_ERR_DEINIT_DRIVER_FAILD -- deinit driver failed

RW_OK -- cmd execute success

说明：

禁用模块相关的硬件初始化，及软件驱动库 ，当用过想禁用 WIFI 功能时，可以调用该

函数，硬件相关的操作由用户在驱动函数中完成，例如关闭 MOS 管 拉低复位引脚 ，设 SPI 管脚为输入上拉等，节省功耗。OS 软件时，会同时删除 WIFI 驱动任务。

2.2.15 复位驱动

```
int rw_sysDriverReset(void);
```

参数：

[in] NULL

返回：

[out]

RW_ERR_DEINIT_DRIVER_FAILD	-- deinit driver failed
RW_ERR_INIT_DRIVER_FAILD	-- initialize driver failed
RW_ERR_PARAM_INVAILD	-- parameter invaild please check
RW_ERR	-- cmd execute failed
RW_OK	-- cmd execute success

说明：

复位软件驱动库及对模块进行硬件复位等，当模块出现异常，或者需要重新连接网络时，对模块驱动和硬件进行复位。OS 软件时，会同时删除 WIFI 驱动任务。

2.2.16 Driverloop

```
int rw_sysDriverLoop(void);
```

参数：

[in] NULL

返回：

[out]

RW_ERR	-- cmd execute failed
RW_OK	-- cmd execute success

说明：

Driver 循环函数，模块的驱动和网络连接的管理，超时的检测等 。该函数只在不跑系统的软件库中使用，维持模块驱动正常，返回回调或者异常事件。如果用户 WIFI 应用数据交互比较频繁 ，则需要经常调用该函数，才能将数据返回给上层。在 send，recv，delay 函数

中，软件库也会调用该函数。如果该函数返回失败 一般是参数设置问题 用于参数调试。

2.2.17 延时函数

```
void rw_sysSleep(int ms);
```

参数：

[in] ms --需要延时的毫秒数

返回：

[out]

NULL

说明：

延时函数用于软件库上的延时 ，主要在延时中加入了 rw_sysDriverLoop 函数，保证事件的回调，数据收发及时。

2.2.18 设置超时函数

```
void  rwSetFutureStamp(rw_stamp_t* stamp, uint32_t msec);
```

参数：

[in] stamp --返回设置过的时间

msec --需要延时的毫秒数

返回：

[out]

NULL

说明：

该函数一般用于不跑操作系统的 软件库中 ，设置一个超时时间，用于判断从命令发送 到超时时 是否收到命令的返回 ，方便客户使用。该类函数需要主控平台提供一个向上计数的计数值。

2.2.19 判断超时函数

```
bool  rwIsStampPassed(rw_stamp_t* stamp);
```

参数：

```
[in]  stamp              --设置过的时间值
```

返回：

```
[out]    Bool 值
    true                 --已超过设定值
    false                --未超过设定值
```

说明：

该函数判断是否超时，配合 rwSetFutureStamp 函数一起使用。

2.3 API 回调函数

软件库 设置了多个回调函数，如扫描结果的回调，网络连接断开事件的回调，IP 获取状态的回调，DNS 解析结果的回调，TCP 客户端连接结果的回调，WPS EASYconfig 配置网络的回调。回调基本是在不带 OS 的软件库中使用。rw_lib.h 中给出了回调函数的类型，具体函数定义 是由用户设置到系统参数中。注意回调函数中不要再执行软件中的任何 API 函数，占用时间不要太长，只需要完成数据的拷贝或者置标志位操作。

2.3.1 网络信息回调

```
typedef void(*rw_WlanScan_)(rw_WlanNetworkInfoList_t* scan_info);
```

参数：

```
[in]  scan_info      --rw_WlanNetworkInfoList_t* 类型
```

返回：

```
[out]    NULL
```

说明：

扫描结果的通知，内存空间有软件库申请 ，注意释放。

2.3.2 网络状态回调

```
typedef void(*rw_WlanConnEvent_)(uint8_t event, rw_WlanConnect_t*
wlan_info, uint8_t dis_reasoncode);
```

参数：

[in] event --表示哪种网络事件 有 STA 连接网络成功，断网等 如下图
wlan_info --返回当前成功连接的路由器信息
dis_reasoncode --断开网络原因 code 如下图

返回：

[out] NULL

说明：

网络事件的通知包含 STA 模式下 连接网络和断开网络的通知 。AP 模式下 建立网络，有客户端连接和断开的事件。回调中还包括当前成功连接的网络信息，名称密码，信道，加密方式等，AP 的 BSSID 等 ，需要时可以进行校验及保存这些网络参数 。断开 code 可以判断出网络设置参数是否有问题，或者未找到与设置相匹配的网络 （名称，密码，加密方式，BSSID）。一般0x01 ，未找到匹配网络，0x0a 密码错误。

```
#define CONN_STATUS_STA_CONNECTED             0
#define CONN_STATUS_STA_DISCONNECT            1
#define CONN_STATUS_AP_ESTABLISH              2 //ap establish
#define CONN_STATUS_AP_CLT_CONNECTED           3 //client connect
#define CONN_STATUS_AP_CLT_DISCONNECT          4 //client disconnect
/*
 * Disconnect Event
 */
typedef enum {
    RW_NO_NETWORK_AVAIL   = 0x01,     /*  not find the match AP,same ssid or authmode */
    RW_LOST_LINK          = 0x02,     /* bmiss */
    RW_DISCONNECT_CMD     = 0x03,
    RW_BSS_DISCONNECTED   = 0x04,
    RW_AUTH_FAILED        = 0x05,
    RW_ASSOC_FAILED       = 0x06,
    RW_NO_RESOURCES_AVAIL = 0x07,
    RW_CSERV_DISCONNECT   = 0x08,
    RW_INVALID_PROFILE    = 0x0a,
    RW_DOT11H_CHANNEL_SWITCH = 0x0b,
    RW_PROFILE_MISMATCH   = 0x0c,
    RW_CONNECTION_EVICTED = 0x0d,
    RW_IBSS_MERGE         = 0xe
```

```
} RW_DISCONNECT_REASON;
```

2.3.3 配置网络回调

```
typedef void(*rw_WlanEasyWps_)(rw_WlanEasyConfigWpsResponse_t*
pResponse, int status);
```

参数：

[in] pResponse --配置网络的参数返回
status --配置是否成功 失败时指超时

返回：

[out] NULL

说明：

一键配置网络和 WPS 配置网络时， 模块监听到手机的发出的路由信息，驱动就会给上层回调，告知调用的成功或失败，成功时，返回当前的网络信息 。

2.3.4 IP 配置回调

```
typedef void(*rw_IpDhcp_)(rw_IpConfig_t* addr, int status);
```

参数：

[in] addr --IP 配置返回的参数
status --IP 获取是否成功

返回：

[out] NULL

说明：

用户执行 DHCP client 时，函数是非阻塞的，DHCP 的结果通过回调通知上层，根据 status 值判断是否失败，成功时可以看到模块分配到 IP 地址，网关地址，DNS 服务器信息。

2.3.5 DNS 回调

```
typedef void(*rw_DnsResult_) (int dnsIp);
```

参数：

[in] dnsIp --DNS 解析出的 IP 地址，为 0 时 表示失败。

返回：

[out] NULL

说明：

DNS 解析的结果回调，如果 IP 地址为0 表示 DNS 失败超时，上层需要重新调用解析。

2.3.6 驱动库 assert 出错回调

```
static void customer_assert(const char* file, int line)
```

参数：

[in] file --运行异常时代码所在的文件

line --运行异常时代码所在的行号

返回：

[out] NULL

说明：

为了让用户更好的了解和使用模块驱动，现将模块的驱动运行的异常通知给用户，异常的原因需要看具体的情况 按照正常的操作流程，硬件连接比较稳定的情况下 一般不会出现该错误。出现该错误后 建议客户复位驱动 ，模块测试时 排查原因。

2.4 BSD socket 部分

BSD socket 采用类似 winsock 和 Linux socket 操作方式，可以对 socket 的数据进行查询接收。发送接收可以使用堵塞和非堵塞方式，提供类似标准的 API 函数。

2.4.1 创建 socket

```
int socket(int domain, int type, int protocol);
```

参数：

```
[in]  domain          --网络协议簇 目前支持 AF_INET
      type            --网络协议类型 目前支持 TCP：SOCK_STREAM
                                               UDP：SOCK_DGRAM
      Protocol        --设为 0
```

返回：

```
[out] socket_fd       --socket 句柄，套接字描述符
      RW_ERR          -- cmd execute failed
```

说明：

创建 socket，返回一个以后函数都需要用到的套接字描述符 范围为 0-socket_max_num （用户设置的驱动库配置）。

2.4.2 绑定本地端口

```
int bind(int sockfd,const void *myaddr, socklen_t addrlen);
```

参数：

```
[in]  sockfd          --socket 句柄，套接字描述符
      myaddr          --sockaddr_in 类型，本地 IP 端口信息
      addrlen         --sockaddr_in 结构体长度
```

返回：

```
[out] RW_ERR_SOCKET_INVAILD      --未找到对应 socket 套接字描述符
      RW_ERR_CMD_PENDING          --该 socket 有命令堵塞
      RW_OK                      --执行成功
      RW_ERR                     -- cmd execute failed
```

说明：

```
typedef  struct sockaddr_in {
                uint16_t sin_port ;             //Port number
                uint16_t sin_family ;        //ATH_AF_INET
                uint32_t sin_addr ;             //IPv4 Address
}SOCKADDR_IN;
```

绑定 socketfd 至本地，该函数用来指定一个端口号，一个 IP 地址，两者都指定，或者两者都不指定.可以不使用该函数调用。使用 socket()得到套接口后可以直接调用函数 connect（）或者 listen（）.不使用该函数，内部自动分配固定端口1024。TCP 连接时，建议客户进行端口绑定 ，并使用随机端口号 ，防止模块硬件复位时，服务器的 TCP 连接没有正常释放。

2.4.3 TCP 连接

```
int connect(int sockfd, void *serv_addr, int addrlen);
```

参数：

```
    [in]  sockfd            --socket 句柄，套接字描述符
          serv_addr         --sockaddr_in 类型，服务器 IP 端口信息
          addrlen           --sockaddr_in 结构体长度
```

返回：

```
    [out] RW_ERR_SOCKET_INVAILD          --未找到对应 socket 套接字描述符
          RW_ERR_CMD_PENDING              --该 socket 有命令堵塞
          RW_OK                          --执行成功
          RW_ERR                          -- cmd execute failed
```

说明：

连接一个指定 IP 和端口的远端服务器，NOS 下为非阻塞操作，连接结果（成功/失败）将在 TCPC 的回调函数中给出，OS 下为阻塞操作，函数返回判断是否连接成功。

2.4.4 TCP 监听

```
int listen(int sockfd, int backlog);
```

参数：

```
    [in]  sockfd            --socket 句柄，套接字描述符
          backlog           --未使用，填 1 即可
```

返回：

```
    [out] RW_ERR_SOCKET_INVAILD          --未找到对应 socket 套接字描述符
```

```
RW_ERR_CMD_PENDING              --该 socket 有命令堵塞
RW_OK                           --执行成功
RW_ERR                          -- cmd execute failed
```

说明：

Sockfd 是创建 TCP socket 时返回的描述符，TCPserver 监听 TCP 连接，通过 select 函数监听该描述符，发现新连接时，则调用 accept 函数来接收这个新连接。

2.4.5 接收 TCP 连接

```
int accept(int sockfd, void *addr, int *addrlen);
```

参数：

```
[in]  sockfd            --socket 句柄，套接字描述符
      addr              --sockaddr_in 类型，新连接 IP 端口信息
      addrlen           --sockaddr_in 结构体长度
```

返回：

```
[out] RW_ERR_SOCKET_INVAILD         --未找到对应 socket 套接字描述符
      RW_ERR_CMD_PENDING            --该 socket 有命令堵塞
      sockfd                        --socket 句柄，新 TCP 连接使用
      RW_ERR                        -- cmd execute failed
```

说明：

TCPServer 接收一个新的 TCP 连接，addr 中保存了新连接的 IP 和端口号信息，如果调用成功，将返回新的 sockfd ，新连接将使用该 sockfd 进行通信。

2.4.6 send 函数

```
int send(int sockfd, const void *msg, int len, int flags);
```

参数：

```
[in]  sockfd            --socket 句柄，套接字描述符
      msg               --需要发送的数据指针
      len               --需要发送的数据长度
      Flags             --发送标记 一般为 0
```

返回：

```
[out] RW_ERR_SOCKET_INVAILD         --未找到对应 socket 或已经被断开
      RW_ERR_NO_MEMORY              --申请内存空间失败
      RW_ERR_SEND_BUFFER_FULL       --内部发送 buff 满，需要重试
```

RW_ERR	-- cmd execute failed
Sendlen (>0)	--发送成功 返回发送的字节数

说明：

发送函数 一般用来发送指定 sockfd 的 TCP 数据，不需要指定对方的 IP 和端口信息。NOS 下 发送函数是非阻塞的 ，发送成功后会返回发送的字节数，该发送函数最大一包只能发送 MAX_SEND_PACKET_LEN，一般为1400字节。

2.4.7 sendto 函数

```
int sendto(int sockfd, const void *data, size_t size, int flags, const
void *to, socklen_t tolen);
```

参数：

[in]	sockfd	--socket 句柄，套接字描述符
	data	--需要发送的数据指针
	size	--需要发送的数据长度
	Flags	--发送标记 一般为 0
	to	--sockaddr_in 类型 接收端的 IP 和端口信息
	tolen	--sockaddr_in 结构体长度

返回：

[out]	RW_ERR_SOCKET_INVAILD	--未找到对应 socket 或连接已断开
	RW_ERR_NO_MEMORY	--申请内存空间失败
	RW_ERR_SEND_BUFFER_FULL	--内部发送 buff 满，需要重试
	RW_ERR	-- cmd execute failed
	Sendlen (>0)	--发送成功 返回发送的字节数

说明：

该函数主要用来发送指定 sockfd 的 UDP 数据，函数需要填写对方的 IP 地址和端口号。NOS 下 发送函数是非阻塞的 ，发送成功后会返回发送的字节数，该发送函数最大一包只能发送 MAX_SEND_PACKET_LEN，一般为1400字节。

2.4.8 recv 函数

```
int recv(int sockfd, void *buf, int len, unsigned int flags);
```

参数：

[in]	sockfd	--socket 句柄，套接字描述符
	buf	--接收数据需要存放的位置

```
        len                 --数据包最大的长度
        flags               --接收标记 一般为 0
```

返回：

```
    [out] RW_ERR_SOCKET_INVAILD          --未找到对应 socket 或连接已断开
          RW_ERR                          -- cmd execute failed
         recvlen (>0)                    --实际接收到的字节数
```

说明：

该函数主要用来接收指定 sockfd 的 TCP 的数据，接收时需要指出接收一包时最大的数据长度，MAX_RECV_PACKET_LEN，UDP: 1536 TCP:1452 。接收时需要上层定义该大小的 buffer,如果设置的值小于该值，收到一包大于设置值的数据，驱动只会拷贝设置长度的数据至指定 buffer，后面的数据将会被丢弃， 防止 buffer 溢出。函数返回实际接收到的数据长度。

2.4.9 recvfrom 函数

```
int recvfrom(int sockfd, void *mem, size_t len, int flags, void *from,
socklen_t *fromlen);
```

参数：

```
    [in]  sockfd              --socket 句柄，套接字描述符
          mem                 --接收数据需要存放的位置
          len                 --数据包最大的长度
          flags               --接收标记 一般为 0
          from                --sockaddr_in 类型 对端的 IP 和端口信息
          fromlen             --sockaddr_in 结构体长度
```

返回：

```
    [out] RW_ERR_SOCKET_INVAILD          --未找到对应 socket 或连接已断开
          RW_ERR                          -- cmd execute failed
         recvlen (>0)                    --实际接收到的字节数
```

说明：

该函数主要用来接收指定 sockfd 的 UDP 的数据，接收时需要指出接收一包时最大的数据长度，MAX_RECV_PACKET_LEN，UDP: 1536 TCP:1452 。接收时需要上层定义该大小的 buffer,如果设置的值小于该值，收到一包大于设置值的数据，驱动只会拷贝设置长度的数据至指定 buffer，后面的数据将会被丢弃， 防止 buffer 溢出。函数返回实际接收到的数据长度。数据的发送端信息将会保存到 from 和 fromlen 中。

2.4.10 shutdown，close 函数

```
int shutdown(int sockfd, int how);
int close(int sockfd);
```

参数：

```
[in]  sockfd              --socket 句柄，套接字描述符
      how                 --未使用 填 2 即可 等同于 close
```

返回：

```
[out] RW_ERR_SOCKET_INVAILD        --未找到对应 socket 套接字描述符
      RW_ERR                       -- cmd execute  failed
      RW_OK                        --  cmd execute success
```

说明：

程序进行网络传输完毕后，你需要关闭这个套接字描述符所表示的连接，可以调用 close 函数关闭。发送或接收过程中，连接异常断开后， 出现 RW_ERR_SOCKET_INVAILD 错误，也可调用该函数，清除内部连接。

2.4.11 gethostbyname 函数

```
int gethostbyname(const char *hostname ,uint8_t namelen ,uint32_t*
out_ipaddr,uint16_t family);
```

参数：

```
[in]  hostname              --需要解析的域名
      namelen               --域名字节长度
      out_ipaddr            --保存解析到的 IP 地址
      family                --未使用 填 AF_INET 即可
```

返回：

```
[out] RW_ERR                       -- cmd execute  failed
      RW_OK                        --  cmd execute success
```

说明：

该函数执行时堵塞的，NOS 和 OS 时 都是堵塞的 ，用户传入需要解析的域名和长度，函数返回成功后 IP 地址将保存在 out_ipaddr 中。

2.4.12 select 函数

```
int select(int sockfd, uint32_t timeout_ms);
```

参数：

```
[in]  sockfd                    --需要等待数据的 socket 描述符
      timeout_ms                --堵塞时间  0：never timeout
                                            >0: times (ms)
```

返回：

```
[out] RW_ERR_SOCKET_INVAILD     --未找到对应 socket 套接字描述符
      RW_ERR_CMD_PENDING        --该 socket 有命令堵塞
      RW_ERR                    -- cmd execute  failed
      RW_OK                     --  cmd execute success
```

说明：

该函数用来等待指定 socket 的事件，可以一直等待也可以指定等待的毫秒数。目前 select 函数只能用来监听一个 socket 描述符的数据。

3. ERROR CODE 表

Error code	code 解释
0	RW_OK 命令执行成功
-1	RW_ERR 命令内部执行发送失败
-2	RW_ERR_PARAM_INVAILD 参数出错
-3	RW_ERR_INIT_DRIVER_FAILD 初始化驱动失败
-4	RW_ERR_DEINIT_DRIVER_FAILD 反初始化驱动失败
-5	RW_ERR_NO_DRIVER 未初始化驱动
-6	RW_ERR_NO_MEMORY 内存不够
-7	RW_ERR_INVAILD_SSID 非法的 SSID 设置
-8	RW_ERR_INVAILD_PSK 非法的密码设置
-9	RW_ERR_INVAILD_CHANNEL 非法的信道设置
-10	RW_ERR_INVAILD_SEC_MODE 非法的加密模式
-11	RW_ERR_INVAILD_AUTH_MODE 非法的加密方式
-12	RW_ERR_INVAILD_ROLE_MODE 非法的网络模式
-13	RW_ERR_INVAILD_CONFIG_MODE 非法的配置模式
-14	RW_ERR_CMD_PENDING 内部命令正在执行
-15	RW_ERR_NO_CONNECT 网络无连接
-16	RW_ERR_NOT_FIND_SSID 没有找到指定的网络
-17	RW_ERR_IP_DHCP IP DHCP 分配失败
-18	RW_ERR_SOCKET_INVAILD socket 参数无效或已经被删除
-19	RW_ERR_SEND_BUFFER_FULL 发送 buff 满 需要重新发送
-20	RW_ERR_TIME_OUT 命令执行超时 ， 未等到命令的返回

4 销售与服务

北京

FAE 邮箱：allan.jin@rakwireless.com 金彦哲

电话：010-62716015

传真：010-62716015

地址：北京市海淀区德胜门外西三旗金燕龙大厦 1108 室

上海

FAE 邮箱：steven.tang@rakwireless.com 汤孝义

电话：021-54721182

传真：021-54721038

地址：上海市闵行区万源路 2161 弄 150 号冉东商务中心 1 幢 306 室

深圳

FAE 邮箱：vincent.wu@rakwireless.com 吴先顺

电话：0755- 26506594

传真：0755- 86152201

地址：深圳市南山区科技园北区清华信息港综合楼 406 室

5 历史版本

版本号	修改内容	修改日期
V1.0	建立文档	2015-04-20
V1.1	1. 添加部分系统参数 2. 删除 TCP 回调事件 3. 添加命令执行超时错误 4. 补充 socket 命令执行返回	2015-06-06