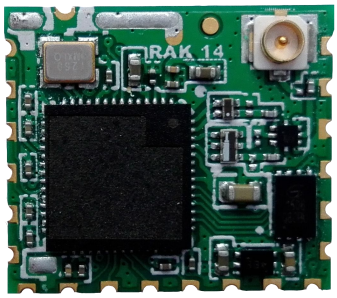
RAK439 API Library User Guide V1.1



Shenzhen Rakwireless Technology Co.,Ltd

www.rakwireless.com

Email: info@rakwireless.com

Content

1. SDK Library Overview

RAK439 is an ultra-low-power, low-cost module that fully supports of major encryption modes as well as WAPI encryption mode, a Wi-Fi ® module with SPI interface that supports 802.11b/g/n protocol. The module internally integrates RF station, balun, RF switch, crystal oscillator and power switch circuit, enabling fast hardware design with very little circuits for peripherals. Plus, the module has built-in IPEX external antenna base, also leads antenna foot that allows on-board antenna design。

RAK439 SDK software library has encapsulation for WIFI module interface driver, providing a platform-specific driver function, so that customers can easily integrate the module into their main platform, greatly shorten the evaluation cycle. Software library also provides a network-related API; user can quickly complete the module's network configuration and connection. Socket section uses BSD socket type operation, easy to operate.

Driver Library supports both NOS operation and OS operation; user can choose either according to actual operation flexibility.

2. API Library Introduction

This section describes data structures of API and how to use API.

2.1 API Data Structure

This section describes several types of important data structures of API. The rw_lib.h includes driver function declarations in driver library, data structure of software library configuration, data structure of network part and type declaration of callback function.

2.1.1 HAL Driver Function Type

The hardware-related driver functions are associated with host platform hardware, refer to RAK439 Driver Migration Manual.

2.1.2 LIB Configuration Data Structures

The configuration structure can set the resources size of software library and WIFI module parameters, also can set initialization of drive function, the callback function, user can set up their own configuration.

```
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;
```

Parameter Description:

Structure Member	Member Type	Member Description
spi_int_enable	bool	Whether use module interrupt pin， if disable,host did not need to detect this interrupt,driver lib will loop read if there is some packet to send to the host If NOS, this is an option for customer. If OS , SPI int must be set ture .
rx_queue_num	uint8_t	The number of RX buffer > = 1, one RX buffer size is 1664 bytes, RAK439 allocates buffer when load drivers (execute rw_sysDriverInit), releases buffer when uninstall drivers (execute rw_sysDriverDeinit). Appropriately increasing the number of the receiving buffers can increase throughput of module bi-directional data
socket_max_num	uint8_t	Number of Sockets, the maximum is eight, reducing the number can reduce resource consumption of Driver Library
scan_max_num	uint8_t	Maximum number of network scanned. Library will apply the space according to the number set before. Do not set too small space. When scan with no SSID specified, if the SSID has poor signal, scan list is sorted by signal value, the SSID with poor signal may be lost
tcp_retry_num	uint8_t	This parameter is used to set the tcp retry times when send failure.The value should not set too low,it will lead the tcp disconnect event easily when Network congestion. It alse should not set too high ,it will delay the application to know the tcp disconnect event . Default is 5 times.
host_name	char*	Host name of module, can be set at dhcpclient and display the name in route
country_code	char*	wireless work country code of module, set the appropriate country, the module work within the corresponding channel designated by the State CN (1-13), JP (1-14), US (1-11)
driver_cb	driver_cb_	Hardware-related driver functions of Driver Library
app_cb	app_cb_	Application-specific callback functions of Driver Library

2.1.3 Network Information structure

The information structure of the network returned when scan, including the number of the scanned, and name each network, channel, encryption and other information.

```
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;
```

Parameter Description:

Structure Member	Member Type	Member Description
num	int32	The number of valid network scanned, maximum is 16
rw_WlanNetworkInfo_t.channel	uint8_t	The current channel where network is located, 1-14
rw_WlanNetworkInfo_t.ssid_len	uint8_t	Name of the current network, the length of the string, max is 32B
rw_WlanNetworkInfo_t.rssi	uint8_t	The signal strength of the current network, such as 50, indicating the signal strength is -50dbm
rw_WlanNetworkInfo_t.bssid	uint8_t	bssid of current network, typically is the MAC address of the route, length is 6B
rw_WlanNetworkInfo_t.ssid	uint8_t	The name of current network, string, max is 32B
rw_WlanNetworkInfo_t.sec_mode	uint8_t	The encrypted status of current network, 0: Open 1: Encrypted
rw_WlanNetworkInfo_t.auth_mode	uint8_t	encryption mode of current network, as described in the following rw_AuthMode_t type

2.1.4 Connect / Create network structure

The structure passed when connected to network contains the name of network to be connected, whether set encryption, password, and encryption method. Also it contains some advanced parameters, such as, specified route can be added (set bssid, distinguish SSID when there are same SSIDs, roaming network), if no SSID to be specified, set this to NULL. PMK keys can be used for networking, speeding up connection to network, if do not use this set, set it to NULL. If user is unsure about encryption of route, sets auth_mode to auto. When create network AP, simply specify the name and channel, etc. for the AP.

```
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;
```

Parameter Description:

Structure Member	Member Type	Member Description
role_mode	rw_WlanMode_t	Network mode, currently support ROLE_STA, ROLE_AP
channel	uint8_t	Connect or create a channel where network stays, 1-14. Setting it to 0, meaning not specify a channel
sec_mode	rw_SecMode_t	Whether or not the network is secured, required item, RW_SEC_TYPE_OPEN, RW_SEC_TYPE_SEC
auth_mode	rw_AuthMode_t	Encryption of network. If user is unsure about the routing connection encryption mode, it can be set to auto. In AP mode, support RW_AUTH_TYPE_WPA2_PSK_AES only
psk	char*	Password of current network, string, max is 64B. when using PMK, can be set to NULL
pmk	char*	PMK password of current network, fixed-32B. when using psk, can be set to NULL
ssid	char*	The name of current network, string, max is 32B
bssid	uint8_t*	bssid of Ccrrent network, typically is the length of route MAC address with length of 6B. when it is not required to join, set to NULL

2.1.5 IP parameter structure

Setting IP parameters, there are several ways, static configuration, DHCP dynamic configuration. In AP mode, set static IP, and enable DHCPsever, assign IP joined wireless client. IP parameter contains IP address, subnet mask, gateway, and DSN server address.

```
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;
```

Parameter Description:

Structure Member	Member Type	Member Description
	rw_IpConfigMode_t	IP Configuration: IP_CONFIG_STATIC = 0, IP_CONFIG_QUERY, DHCP_CLIENT,DHCP_SERVER
rw_IpConfig_t.addr	uint32_t	IPV4 parameter: IP address
rw_IpConfig_t.mask	uint32_t	Subnet Mask
rw_IpConfig_t.gw	uint32_t	IP gateway address
rw_IpConfig_t.svr1	uint32_t	Preferred DNS server address
rw_IpConfig_t.svr2	uint32_t	Alternative DNS server address

2.1.6 Return Structure of WPS EASY Configuration

The module supports WPS2.0 and Easyconfig networking, returns information of the route to be connected, such as ssid, password. Users can set the necessary parameters to network parameters to connect the route simply.

```
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;
```

Parameter Description:

Structure Member	Member Type	Member Description
bssid	uint8_t	bssid of Ccrrent network, typically is the length of route MAC address with length of 6B
ssid	uint8_t	Ssid connected with current network, string, the maximum is 32B
psk	uint8_t	password of current network, string, the maximum is 64B
channel	uint8_t	The current channel where network stays, 1-14

2.2 API Driver Function

API function contains driver initialization, network configuration, network connection, query, socket and other operational functions. The network part is in Rw_lib.h, and socket part in rw_socket.h.

2.2.1 Driver Initialization

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

Parameter

[in] rw_DriverParams_t* params -- Driver parameters

Return Value:

[out]

RW_ERR_INIT_DRIVER_FAILD -- initialize driver failed

RW_ERR_PARAM_INVAILD -- parameter invalid, please check

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

Basic settings of driver library, driver functions, the settings of callback function, and driver initialization. If the return of drive initialization fails, check the connection, such as SPI. The API needs to be called firstly, then execute other API functions after successful initialization.

2.2.2 Scan Network

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

Parameter:

[in] pssid -- whether specifies Ssid, if ssid is not specified, passing NULL

Channel -- Scan the specified channel, if scan all channels, set to 0

Return Value:

[out]

RW_ERR_INVAILD_CHANNEL -- channel parameter invalid

RW_ERR_INVAILD_SSID -- ssid parameter invalid

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

This command is used to scan network AP information, user can scan the network with specified name or scan the specified channel. When use the software libraries without operating system, the scan results will be returned with rw_WlanNetworkInfoList_t * callback. When scan, the library will apply space based on the maximum number set, and sort the AP information, according to signal strength, from strong to weak. After scanning, release the portion of space to avoid repeat applications of space, wasting of resources. When use the software library with OS, scan results can be directly acquired by rw_wlanGetScanInfo function, and also need to release space resources.

2.2.3 Get Scan Results

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

Parameter:

[in] pInfoList -- Pointing to the pointer stored scan results

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

This function is used in software library with OS. The function can block the return of internal scan results. Timeout: 5s. Scan results acquired will be stored in pInfoList with rw_WlanNetworkInfoList_t type of pointer. Release the memory after processing the results.

2.2.4 AP Configuration

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

Parameter:

[in] is_hidden --1： Hidden 0：Unhidden

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

The command is used to set AP name hidden status to prevent other wireless clients to join. This command needs to be called before rw_wlanConnect.

2.2.5 Connect / Create network

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

Parameter:

[in] conn -- Network parameters for connecting or creating network, the name of network, passwords, etc.

Return Value:

[out]

RW_ERR_INVAILD_ROLE_MODE -- role mode parameter invalid

RW_ERR_INVAILD_SSID -- ssid parameter invalid

RW_ERR_INVAILD_CHANNEL -- channel parameter invalid

RW_ERR_INVAILD_SEC_MODE -- sec_mode parameter invalid

RW_ERR_INVAILD_AUTH_MODE -- auth_mode parameter invalid

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

As the STA connecting to a router, user needs to set parameters of router, rw_WlanConnect_t structure contains the required parameters of the connection. When create AP, also need to set basic parameters, network name, channel, password, encrypted status, encryption method. AP encryption only supports WPA2-AES. For specific parameters of Structure, refer to Parameter section.

2.2.6 IP Parameter Setting

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

Parameter:

[in] ip_addr -- IP address, gateway and other information

[in] mode -- IP Set / query mode, IP static settings, dhcp client, dhcpserver

Return Value:

[out]

RW_ERR_IP_DHCP -- ipdhcp fail or timeout

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

IP set command. After a network is connected / created successfully, user needs to call this command to set IP information of module. It can be a static configuration (same as network segment of the router), or a dynamic assigned IP from the router. The callback notification will be given no matter the set is succeeded or failed. In AP mode, firsly set IP information for static module, then enable DHCPserver function of the module to assign IP for wireless client. Command Example:

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

2.2.7 DNS

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

Parameter:

[in] host_name -- domain name

name_len -- length of domain name

family -- Protocol suite currently used, support AF_INET IPv4

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

DNS DNS command is used to resolve a given domain name to an IP address, the command invokes the need in the module as STA, has networked and access to the IP address and DNS server address (usually the gateway address), when you need to set a static IP setting module

available the DNS server addresses. The command is non-blocking, DNS results will be output in the callback function. See DNS callback introduction. BSD socket has a similar function: int gethostbyname (const char * hostname, uint8_t namelen, uint32_t * out_ipaddr, uint16_t family); the function is blocked calls, direct return IP addresses.

2.2.8 WPS EASY Newwork Configuration

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

Parameter:

[in] mode -- rw_ConfigMode_t Type，CONFIG_EASY，CONFIG_WPS Mode

Return Value:

[out]

RW_ERR_INVAILD_CONFIG_MODE -- config mode parameter invaild

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

WPS and EASYconfig are two methods of easy configuration for network parameters, the module can execute these commands after a normal start. easyconfig configures phone APP, getting the required name and password information of the route to be connected, software libraries informs the upper layer with configuration results via a callback function. For WPS, you press the WPS button on route. Get the name and password, user can save them to Flash, next enabling automatically connection when power on.

2.2.9 Set Power Mode

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

Parameter:

[in] pwr_mode -- rw_PowerMode_t type，POWER_MAX，POWER_SAVE mode

Return Value:

[out]

RW_ERR_INVAILD_CONFIG_MODE -- config mode parameter invaild

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

Set power mode for module. It is only effective in STA mode. power_max module will run in full speed with the best performance. When switch to power saving mode, the module keeps a DTIM interval with route after a successful connection, to wake, send and receive data.

2.2.10 Get PMK

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

Parameter:

[in] pmk -- char * type, the value needs to be filled in when pointing to PMK

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

Get the PMK value of network, effective only in STA mode, WPA or WPA2 encryption. The PMK value is fixed 32 bytes. After module successfully joined the network, PMK value is acquired (similar to a password). This way of using the PMK instead of password to join the network, can save about 1.5S time. User can use this password to connect to network in some scenes where requires a fast network connection.

2.2.11 Get signal strength

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

Parameter:

[in] NULL

Return Value:

[out]

RW_ERR -- cmd execute failed

>0 -- rssi value

Description：

Get the strength of the current router connected, active in STA mode. The return value is positive (up to 96), such as return = 50, meaning the value of RSSI is -50, the smaller the value,

the stronger the signal strength.

2.2.12 Get signal strength

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

Parameter:

[in] mac_data -- char * type, the address needs to be filled in when pointing to MAC values

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

MAC of module, globally unique MAC address is scanned after factory, 6 bytes, as module hardware identification.

2.2.13 Get Software Version

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

Parameter:

[in] version -- char * type, the address needs to be filled in when pointing to version value

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

Get software library and WIFI version, such as 1.0.4-2.1.39.

2.2.14 Deinitialization

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

Parameter:

[in] NULL

Return Value:

[out]

RW_ERR_DEINIT_DRIVER_FAILD -- deinit driver failed

RW_OK -- cmd execute success

Description:

Disable the module-associated hardware initialization, and software driver library. After WIFI function is disabled, user can call this function; hardware-related operations are carried out by user in driver functions, such as turning off MOS pin, pull down reset pin, setting SPI pin as input pull up, etc., to save power. When conduct software OS, module deletes WIFI driver tasks.

2.2.15 Reset Driver

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

Parameter:

[in] NULL

Return Value:

[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

Description:

Reset Software driver library and reset module hardware. When the module is abnormal, or need to re-connect to network, reset module drivers and hardware. When conduct software OS, it also deletes WIFI driver tasks.

2.2.16 Driverloop

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

Parameter:

[in] NULL

Return Value:

[out]

RW_ERR -- cmd execute failed

RW_OK -- cmd execute success

Description:

It is driver loop function for driver and network connection management and timeout detection. This function is only run in software library without OS, to maintain drivers working normally and return callbacks or abnormal events. If user's WIFI application data exchange frequently, user needs to frequently call this function so that to return the data to upper layer. In the functions of send, recv, and delay, the software library will also call this function. If this function returns failed, it is usually caused by parameter configuration.

2.2.17 Delay function

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

Parameter:

[in] ms -- The number of milliseconds required of delay

Return Value:

[out]

NULL

Description:

Delay function is used for delaying software library, mainly added rw_sysDriverLoop function in the delay, ensuring callback events, sending and receiving data in a timely manner.

2.2.18 Set timeout function

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

Parameter:

[in] stamp --Return the time set

msec -- The number of milliseconds required of delay

Return Value:

[out]

NULL

Description:

This function is generally used for the software library that does not run operating system, set a timeout to judge whether it has received returns of commands within the time period from commands sent to command timeout, enabling customers to use. The function requires a a count-up counter value provided by master platform.

2.2.19 Timeout Function

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

Parameter:

[in] stamp --Time value that has been set

Return Value:

[out] Bool value

true -- It exceeds the set value

false -- Does not exceed the set value

Description:

This function determines whether timeout, should use together with rwSetFutureStamp.

2.3 API Callback Function

Software Library sets multiple callbacks, such as callback of scan results, events callback of network connection and disconnection, callback of IP state, callback of DNS resolution result, callback of TCP client connection result, callback of WPS EASYconfig network configuration. Callback is basically used in the software library without OS. In rw_lib.h, it introduces types of callback function; the specific function definition is set by user into system parameters. In callback, do not execute any API function in the software, do not take too long time, only need to complete the operations of data copy and set flag.

2.3.1 Network Information Callback

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

Parameter:

[in] scan_info --rw_WlanNetworkInfoList_t* type

Return Value:

[out] NULL

Description:

This is notification of scan results. Software library should apply memory space, pay attention to the release a certain space.

2.3.2 Network Status Callback

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

Parameter:

[in] event -- It indicates which network events have successful and unsuccessful STA connection to network, as shown below

wlan_info -- Returns information of the current successful connected router

dis_reasoncode -- The reason of disconnection from network, code as shown below

Return Value:

[out] NULL

Description:

Network event notification includes the notifications of network connection and disconnection in STA mode. Creating network in AP mode, there are events of client connection and disconnection. Callback also includes the network information of the current successful network connection, such as the name, password, channel, encryption methods, AP's BSSID, etc. When necessary, verify and save these network parameters. Disconnecting code can determine whether there is a problem for network setup parameters, or the network which is matched with network parameters (name, password, encryption, BSSID) was not found. Basically 0x01, means no matching network, and 0x0a means password error.

```
#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 Configuring Network Callback

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

Parameter:

[in] pResponse -- Configure the parameter that will be returned

status -- When configuration is successful, a timeout means failure

Return Value:

[out] NULL

Description:

When conduct Easy Config and WPS configuration, once module listens to the routing information sent by mobile phone, drivers will give callback to upper layer, informing success or failure, if succeed, return the current network information.

2.3.4 IP Configuration Callback

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

Parameter:

[in] addr -- Parameters returned from IP configuration

status -- whether it is successful to get IP

Return Value:

[out] NULL

Description:

When user performs a DHCP client, the function is non-blocking, the results of DHCP will be notified to upper layer via callback, according to status value to determine whether it is failed or not. When it is successful, IP address, gateway address, DNS server information can be checked.

2.3.5 DNS Callback

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

Parameter:

[in] dnsIp -- the IP address parsed out via DNS, 0 means failure

Return Value:

[out] NULL

Description:

Results callback of DNS resolution, if IP address is 0, it indicates DNS failure timeout, the upper layer has to call again to resolve.

2.3.6 Error Callback of Driver Library Assert

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

Parameter:

[in] file -- the code file when there is exceptional runtime

line -- the line of code when there is exceptional runtime

Return Value:

[out] NULL

Description:

To allow user to better understand and use the module drivers, here to inform the running abnormal of module driver to user, the reason of abnormal case is rather case by case in specific circumstances. Usually tt does not occur error if operate in accordance with normal operating procedures and hardware connection is stable. However, recommend that customers reset drive, do module test to investigate possible reasons if any error occurs.

2.4 BSD Socket Part

With similar operating mode as winsock and Linux socket, BSD socket can query and receive data of socket. Transmission and reception can be in blocking and non-blocking ways to provide similar standards of API functions.

2.4.1 Create Socket

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

Parameter:

[in] domain -- network protocol stack, AF_INET is currently supported

type -- Network protocol, currently supports TCP: SOCK_STREAM
UDP: SOCK_DGRAM

Protocol --set to 0

Return Value:

[out] socket_fd -- socket handle, socket descriptor

RW_ERR -- cmd execute failed

Description:

Create socket; return a socket descriptor that will be commonly used by later functions with range of 0- socket_max_num (driver library configuration set by user).

2.4.2 Binding a Local Port

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
myaddr -- sockaddr_in type, local IP port information
addrlen -- length of sockaddr_in structure

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- The corresponding socket descriptor was not found
RW_ERR_CMD_PENDING --socket is blocked, wait last cmd response
RW_OK -- Successfully executed
RW_ERR -- cmd execute failed

Description:

```
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;
```

Bind socketfd to local, this function is used to specify a port number, an IP address, specify both, or specify neither. User may not use this function to call. After using the socket () to obtain a socket user can directly call function connect () or listen (). Without this function, a fixed port 1024 is assigned automatically. When in TCP connection, it is recommended for customers to bind a port, and use random port number to ensure TCP connection of server is normally release when module hardware reset.

2.4.3 TCP Connection

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

Parameter:

[in] sockfd --socket handle, socket descriptor
serv_addr -- sockaddr_in type, server IP port information
addrlen --length of sockaddr_in structure

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- The corresponding socket descriptor was not found
RW_ERR_CMD_PENDING --socket is blocked, wait last cmd response
RW_OK -- Successfully executed
RW_ERR -- cmd execute failed

Description:

Connect to a remote server with specified IP and port, non-blocking operation with NOS, the join result (success / failure) will be given in TCPC callback function; blocking operation with OS, the function return determines whether the connection is successful.

2.4.4 TCP Listen

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
backlog -- if not used, fill 1

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- The corresponding socket descriptor was not found
RW_ERR_CMD_PENDING --socket is blocked, wait last cmd response
RW_OK -- Successfully executed
RW_ERR -- cmd execute failed

Description:

Sockfd is the descriptor returned when create a TCP socket, TCP server listens TCP connections, monitoring the descriptor with select function, when a new connection is found, call accept function to receive this new connection.

2.4.5 Receiving TCP Connection

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
addr -- sockaddr_in type, IP port information of new connection
addrlen --length of sockaddr_in structure

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- The corresponding socket descriptor was not found
RW_ERR_CMD_PENDING --socket is blocked, wait last cmd response
sockfd -- socket handle, use in new TCP connection
RW_ERR -- cmd execute failed

Description:

TCPServer receives a new TCP connection, the IP and port number information of the new

connection are saved in addr, if the call is successful, it will return a new sockfd, new connections will communicate with the sockfd.

2.4.6 Send Function

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
msg -- data pointer to be sent
len -- data length to be sent
Flags -- Send flag, usually is 0

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- the corresponding socket is not found or has been disconnected
RW_ERR_NO_MEMORY -- memory space application is failed
RW_ERR_SEND_BUFFER_FULL -- Internal transmit buff is full, needs to retry
RW_ERR -- cmd execute failed
Sendlen (>0) -- sent successfully, return the number of bytes sent

Description:

Send function that is generally used to send TCP data of specified sockfd, no need to specify each other's IP and port information. In NOS, send function is non-blocking, the number of bytes will be returned when sent successfully, and transmission function can only send a packet with MAX_SEND_PACKET_LEN, typically 1400 bytes.

2.4.7 Sendto Function

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
data -- Data pointer to be sent
size -- data length to be sent
Flags -- Send flag, usually is 0
to -- sockaddr_in type, IP and port information in receiving end
tolen --length of sockaddr_in structure

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- the corresponding socket is not found or has been disconnected

RW_ERR_NO_MEMORY -- memory space application is failed

RW_ERR_SEND_BUFFER_FULL -- Internal transmit buff is full, needs to retry

RW_ERR -- cmd execute failed

Sendlen (>0) -- sent successfully, return the number of bytes sent

Description:

This function is mainly used to send the UDP data of specified sockfd, need to fill each other's IP address and port number. In NOS, send function is non-blocking, the number of bytes will be returned when sent successfully, transmission function can only send a packet with MAX_SEND_PACKET_LEN, typically 1400 bytes.

2.4.8 Recv Function

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

Parameter:

[in] sockfd -- socket handle, socket descriptor

buf -- location required to restore received data

len -- maximum length of packet

flags -- receive flag, usually is 0

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- the corresponding socket is not found or has been disconnected

RW_ERR -- cmd execute failed

recvlen (>0) -- Actual number of bytes received

Description:

This function is mainly used to receive TCP data of specified sockfd, the maximum data length of a packet to be received should be specified, MAX_RECV_PACKET_LEN, UDP: 1536 TCP: 1452. Also, same size of the buffer in upper layer should be defined, if the value is less than the set value, and receive a package of data that is greater than the set value, the drive will only copy the length of the data set to specified buffer, subsequent data will be discarded to prevent buffer overflowed. Function returns the actual length of the received data.

2.4.9 Recvfrom Function

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
mem -- location required to restore received data
len -- maximum length of packet
flags -- receive flag, usually is 0
from -- sockaddr_in type, IP and port information of peer
fromlen --length sockaddr_in structure

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- the corresponding socket is not found or has been disconnected
RW_ERR -- cmd execute failed
recvlen (>0) -- Actual number of bytes received

Description:

This function is mainly used to receive UDP data of specified sockfd, the maximum data length of a packet to be received should be specified, MAX_RECV_PACKET_LEN, UDP: 1536 TCP: 1452. Also, same size of the buffer in upper layer should be defined, if the value is less than the set value, and receive a package of data that is greater than the set value, the drive will only copy the length of the data set to specified buffer, subsequent data will be discarded to prevent buffer overflowed. Function returns the actual length of the received data. The sender information of data will be saved and in from and fromlen.

2.4.10 Shutdown, Close Function

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

Parameter:

[in] sockfd -- socket handle, socket descriptor
how -- Not available, just fill 2, equivalent to close

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- The corresponding socket descriptor was not found
RW_ERR -- cmd execute failed
RW_OK -- cmd execute success

Description:

When the procedure of network transmission is completed, the connection that represented by socket descriptor needs to be closed, via close function. During the period of sending or receiving, when the connection abnormal is disconnected, RW_ERR_SOCKET_INVAILD error occurs, user can also call this function, remove internal connection.

2.4.11 Gethostbyname Function

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

Parameter:

[in] hostname -- domain name
namelen -- Domain length in bytes
out_ipaddr -- Save IP address resolved
family -- Not available, just fill AF_INET

Return Value:

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

Description:

When the function executes it is blocked, when NOS and OS, both are blocked, users pass domain name to be resolved and length, after function returns successfully, IP address will be saved in out_ipaddr.

2.4.12 Select Function

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

Parameter:

[in] sockfd -- need to wait for data socket descriptor
timeout_ms -- Blocked time 0：never timeout
>0: times (ms)

Return Value:

[out] RW_ERR_SOCKET_INVAILD -- The corresponding socket descriptor was not found
RW_ERR_CMD_PENDING --socket is blocked, wait last cmd response
RW_ERR -- cmd execute failed
RW_OK -- cmd execute succeed

Description:

This function is used to wait for a specified socket event, it can be set to always wait or

specify a number of milliseconds to wait. Currently function can only be used to monitor one socket descriptor data.

3. ERROR CODE Table

Error code	Code Explanation
0	RW_OK Command executed successfully
-1	RW_ERR command failed internally
-2	RW_ERR_PARAM_INVAILD Parameter error
-3	RW_ERR_INIT_DRIVER_FAILD Driver initialization failure
-4	RW_ERR_DEINIT_DRIVER_FAILD Deinitialization driver failure
-5	RW_ERR_NO_DRIVER uninitialized Driver
-6	RW_ERR_NO_MEMORY Out of memory
-7	RW_ERR_INVAILD_SSID Invalid SSID Settings
-8	RW_ERR_INVAILD_PSK Invalid password Settings
-9	RW_ERR_INVAILD_CHANNEL Invalid channel Settings
-10	RW_ERR_INVAILD_SEC_MODE Invalid encryption mode
-11	RW_ERR_INVAILD_AUTH_MODE Invalid authentication mode
-12	RW_ERR_INVAILD_ROLE_MODE Invalid network mode
-13	RW_ERR_INVAILD_CONFIG_MODE Invalid configuration mode
-14	RW_ERR_CMD_PENDING Internal command is pending execution
-15	RW_ERR_NO_CONNECT Network is disconnected
-16	RW_ERR_NOT_FIND_SSID Didn't find the specified network
-17	RW_ERR_IP_DHCP IP DHCP allocation failure
-18	RW_ERR_SOCKET_INVAILD socket parameter is invalid or has been deleted
-19	RW_ERR_SEND_BUFFER_FULL Sending buff is full, needs to send again
-20	RW_ERR_TIME_OUT send cmd timeout , can not receive the response

4 Sales and Service

Beijing Office

FAE mailbox: allan.jin@rakwireless.com 金彦哲

Tel: 010-62716015

Fax: 010-62716015

Address: Room 1108, Jinyan Long Building, Deshengmenwai Xisanqi, Haidian District, Beijing

Shanghai Office

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

Tel: 021-54721182

Fax: 021-54721038

Address: Room 306, 1# Building, Ran Dong Business Center, No.2161, Lane 150 Wanyuan Road, Minhang District, Shanghai

Shenzhen Office

FAE mailbox:vincent.wu@rakwireless.com 吴先顺

Tel：0755- 26506594

Fax：0755-86152201

Address: Room 406, Tsinghua Building, North Science Technology Park, Nanshan District, Shenzhen

5 Revision History

Version No.	Modification	Date
V1.0	Initial draft	2015-04-20
V1.0	1. Add two system parameters 2. Delete the tcpcallback function 3. Add the cmd execute timeout 4. Add socket cmd return code	2015-04-20