Initialization and Configuration of Wi-Fi Interface

This document provides a guideline for Initialization and configuration of Wi-Fi interface. Three methodologies are provided: (1) Pre-definition during code builds (2) Input SSID during System Initialization (3) Command line shell for wlan interface control.

Table of Contents

1 Introduction

This document provides a guideline for Initialization and configuration of Wi-Fi interface. Three methodologies are provided: (1) Pre-definition during code builds [chapter 2] (2) Input SSID during System Initialization [chapter 3] (3) command line shell for wlan interface control [chapter 4].

2 Pre-definition During Code Builds

Some pre-definitions in codes can be used to determine the initialization and configuration of Wi-Fi interface in compile time.

2.1 Configuration of Station Mode with WPA2

The following is the configuration for station mode. CONFIG_INIT_NET in main.c is used to enable network stack and Wi-Fi driver. Set CONFIG_START_STA to 1 enables station to automatically connect to a pre-defined SSID during system initialization. Set CONFIG_WPA2 to 1 for WPA2-AES PSK connection. The network SSID and passphrase can be pre-defined by STA_MODE_SSID and WPA_PASSPHRASE in main.h. Set CONFIG_DHCP_CLIENT to 1 to get IP from AP dynamically, otherwise use the default setting IP.

```
//Config in main.c
#define CONFIG_INIT_NET             1
#define CONFIG_WEP                  0
#define CONFIG_WPA                  0
#define CONFIG_WPA2                 1
#define CONFIG_PING_TEST            0
#define CONFIG_INTERACTIVE_MODE     0
#define CONFIG_POST_INIT            0
#define CONFIG_START_MP             0
#define CONFIG_START_STA            1
#define CONFIG_START_AP             0
#define CONFIG_START_STA_AP         0
#define CONFIG_DHCP_SERVER          0
#define CONFIG_DHCP_CLIENT          0


//Config in main.h
#define STA_MODE_SSID                           "wlab_ap_ssid"
#define WPA_PASSPHRASE                          "12345678"
```

The following represents an example of automatically starting in station mode and connecting to an existed SSID with WPA2-AES PSK.

```
RTL871X: set ssid [galex_mini]

Handshake not done yet

Handshake not done yet

Handshake not done yet

Handshake not done yet

Handshake not done yet

Handshake not done yet

Handshake not done yet

Handshake not done yet

Handshake not done yet

RTL871X: start auth

RTL871X: auth success, start assoc

RTL871X: association success

RTL871X: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1

RTL871X: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

Get essid galex_mini


WIFI Setting:
==========================
      MODE => STATION
      SSID => galex_mini
   CHANNEL => 6
  SECURITY => WPA2
  PASSWORD => 1234567890
```

2.2 Configuration of AP Mode with WPA2

The following is the configuration about AP mode. CONFIG_INIT_NET in main.c is used to enable network stack and Wi-Fi driver. CONFIG_START_AP makes Wi-Fi driver automatically change to AP mode and create a network with a pre-defined SSID when system initialization. Set CONFIG_WPA2 to 1 for WPA2-AES PSK connection. Set CONFIG_DHCP_SERVER to 1 to start DHCP server. The created network SSID, default channel and passphrase can be pre-defined by AP_MODE_SSID, AP_DEFAULT_CH and WPA_PASSPHRASE in main.h.

```
//Config in main.c
#define CONFIG_INIT_NET             1
#define CONFIG_WEP                  0
#define CONFIG_WPA                  0
#define CONFIG_WPA2                 1
#define CONFIG_PING_TEST            0
#define CONFIG_INTERACTIVE_MODE     0
#define CONFIG_POST_INIT            0
#define CONFIG_START_MP             0
#define CONFIG_START_STA            0
#define CONFIG_START_AP             1
#define CONFIG_START_STA_AP         0
#define CONFIG_DHCP_SERVER          0
#define CONFIG_DHCP_CLIENT          0
//Config in main.h
#define AP_MODE_SSID                        "wlan_ap_ssid"
#define AP_DEFAULT_CH                       6
#define WPA_PASSPHRASE                      "12345678"
```

The following represents an example of automatically starting in AP mode and creating a network based on indicated SSID, channel and passphrase with WPA2-AES PSK.

2.3 Configuration of Station and AP Con-current Mode with WPA2

The following is the configuration about Station Mode and AP Con-current mode. CONFIG_INIT_NET in main.c is used to enable network stack and Wi-Fi driver. Set CONFIG_START_STA_AP to 1 to enables station to automatically connect to a pre-defined STA_MODE_SSID and start AP with a pre-defined AP_MODE_SSID during system initialization. Set CONFIG_WPA2 to 1 for WPA2-AES PSK connection. Set CONFIG_DHCP_CLIENT to 1 to get IP from AP dynamically, otherwise use the default setting IP. Set CONFIG_DHCP_SERVER to 1 to start DHCP server for the AP. The AP's channel follows the station's channel if the station connected with some AP.

```
//Config in main.c
```

```
#define CONFIG_INIT_NET             1
#define CONFIG_WEP                  0
#define CONFIG_WPA                  0
#define CONFIG_WPA2                 1
#define CONFIG_PING_TEST            0
#define CONFIG_INTERACTIVE_MODE     0
#define CONFIG_POST_INIT            0
#define CONFIG_START_MP             0
#define CONFIG_START_STA            0
#define CONFIG_START_AP             0
#define CONFIG_START_STA_AP         1
#define CONFIG_DHCP_SERVER          1
#define CONFIG_DHCP_CLIENT          1


//Config in main.h
#define STA_MODE_SSID                           "wlan_ap_ssid"
#define WPA_PASSPHRASE                          "12345678"


#define AP_MODE_SSID                        "wlan_ap_ssid"
#define AP_DEFAULT_CH                           6
#define WPA_PASSPHRASE                          "12345678"
```

The following represents an example of automatically starting in AP mode and creating a network based on indicated SSID, channel and passphrase with WPA2-AES PSK.

```
WIFI  wlan0 Setting:
==============================
      MODE => STATION
      SSID => tplink841_ben
   CHANNEL => 6
  SECURITY => OPEN
  PASSWORD =>

WIFI Status wlan0 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=17, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=3225, rx_dropped=7002, rx_bytes=394358
[rltk_wlan_statistic] min_free_heap_size=21200, current heap free size=22328, wlan_heap_used=27720
[rltk_wlan_statistic] max_skbbuf_used_num=4, skbbuf_used_num=0
[rltk_wlan_statistic] max_skbdata_used_num=3, skbdata_used_num=0
[rltk_wlan_statistic] max_timer_used_num=16

Interface (wlan0)
==============================
  MAC => 00:e0:4c:87:00:03
  IP  => 192.168.1.194
  GW  => 192.168.1.1
```

```
WIFI  wlan1 Setting:
==============================
      MODE => AP
      SSID => dddd
   CHANNEL => 6
  SECURITY => OPEN
  PASSWORD =>

WIFI Status wlan1 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=22, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=15, rx_dropped=8204, rx_bytes=52130

Interface (wlan1)
==============================
  MAC => 02:e0:4c:87:00:03
  IP  => 192.168.43.1
  GW  => 192.168.43.1

 default netif:==============================
  IP  => 192.168.43.1
```

3 Programming Guide

Here lists the different APIs about concurrent mode from v02.4 to v02.5 as follows:

Previous API (v2.4)	Current API (v2.5)
void dhcps_init(void)	void dhcps_init(struct netif * pnetif)
bool rltk_wlan_init(void)	int rltk_wlan_init(int idx_wlan, unsigned int mode)
void rltk_wlan_start(void)	void rltk_wlan_start(int idx_wlan)
void rltk_wlan_statistic(void)	void rltk_wlan_statistic(unsigned char idx)
unsigned char rltk_wlan_running(void)	unsigned char rltk_wlan_running(unsigned char idx)

Except dhcp_init function, other functions are included in wificonf application programming interface (see UM0006 Realtek wificonf application programming interface.pdf). You can refer to the way to use these APIs in application programming interface implementation.

The main changes of these API are that they need to decide which net interface you want to use. Because we enable concurrent mode in v02.5, there's two net interfaces if you start AP and STA mode at the same time. And at this mode, net interface 0 means STA and net interface 1 means AP.

Here shows the example to use "dhcp_init" as follows:

```
//Example in main.c
void init_thread(void *param)
{
        …
        #if CONFIG_START_STA_AP
```

```
                /* Start DHCP Server */
                dhcps_init(&xnetif[1]);
        #else
                /* Start DHCP Server */
                dhcps_init(&xnetif[0]);
        #endif //CONFIG_START_STA_AP
        …
}
```

The parameter xnetif[index] in the concurrent mode, it's STA when index equals to 0 and is AP mode when index equals to 1.

If it's STA or AP standalone mode, index's always equals to 0.

4 Command Line Shell for WLAN Interface Control

Interactive mode provides an input method similar to command line shell. The following is the related configuration about interactive mode. CONFIG_INIT_NET in main.c is used to enable network stack and Wi-Fi driver. CONFIG_INTERACTIVE_MODE can be used to start the interactive mode when system initialization. Before system enters interactive mode, user can also pre-configured CONFIG_START_STA or CONFIG_START_AP or CONFIG_START_STA_AP to initially set Wi-Fi in STA ,AP or concurrent mode during initialization. Set CONFIG_DHCP_CLIENT to 1 to get IP from AP dynamically, otherwise use the default setting IP. Set CONFIG_DHCP_SERVER to 1 to start DHCP server for the AP. The SERIAL_DEBUG_RX in main.h must be enabled to support UART input.

```
//Config in main.c
#define CONFIG_INIT_NET             1
#define CONFIG_WEP                  0
#define CONFIG_WPA                  0
#define CONFIG_WPA2                 0
#define CONFIG_PING_TEST            0
#define CONFIG_INTERACTIVE_MODE     1
#define CONFIG_POST_INIT            0
#define CONFIG_START_MP             0
#define CONFIG_START_STA            0
#define CONFIG_START_AP             0
#define CONFIG_START_STA_AP         0


//Config in main.h
#define SERIAL_DEBUG_RX
```

The following is the example to start interactive mode when system initialization. Since both of CONFIG_START_STA and CONFIG_START_AP are disabled, it will not connect to or create network. After starting interactive mode, the command prompt is shown. The setup of interactive mode and supported commands are presented in chapter 5 and 6 in detail.

4.1 Step by Step

Before using interactive mode, some configuration must be enabled, and then rebuild the image.

Step 1: Enable UART RX. The SERIAL_DEBUG_RX in main.h must to be defined to support UART RX interrupt.

Step 2: Disable debug message if required. Please note that too many debug messages may affect UART input.

Step 3: Enable interactive mode in INIT task. The CONFIG_INTERACTIVE_MODE in main.c must to be set to 1 to make INIT task start the interactive mode when system initialization.

Step 4: UART Terminal Setup. Set Baud rate to 115200, Data 8 bits, Parity none, Stop 1 bit, Flow control none.

4.2 Command Usage

UART interactive mode provides some commands to control Wi-Fi. Users can also implement their commands and add them into command table. The following is the description of built-in commands.

4.2.1 Help

The help command can be used to get supported commands.

```
# help
COMMAND LIST:
==============================
    wifi_connect
    wifi_disconnect
    wifi_info
    wifi_on
    wifi_off
    wifi_ap
    wifi_scan
    wifi_get_rssi
    iwpriv
    wifi_promisc
    wifi_simple_config
    wifi_sta_ap
    ttcp
    ping
    exit
    help
```

4.2.2 Disable/Enable WI-FI

The wifi_on and wifi_off commands are used to initialize and de-initialize Wi-Fi driver correspondingly. Before using the functionality of Wi-Fi driver, it needs to be initialized. After Wi-Fi driver is initialized, it will be in station mode. The following are the output when executing wifi_on and wifi_off commands.

```
# wifi_on
Initializing WIFI ...
WIFI initialized
```

```
# wifi_off
Deinitializing WIFI ...
[mod_timer] netif is DOWN
WIFI deinitialized
```

4.2.3 Network Connection

The wifi_connect command can be used to connect to an access point. By typing this command without parameters will show the command usage. If password is not given, this command will try to connect to the network in open mode. Otherwise, it will try to connect to the network with WPA2-AES PSK. If connecting to an access pointer, the wifi_disconnect command will make disconnected to this network.

To disconnect AP, type wifi_disconnect.

WPA2 and WPA STA mode

Command: wifi_connect SSID Passphrase

```
# wifi_connect galex_mini 1234567890
Joining BSS ...
RTL871X: set ssid [galex_mini]

RTL871X: start auth

RTL871X: auth success, start assoc

RTL871X: association success

RTL871X: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:1

RTL871X: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)

galex_mini connected
```

```
# wifi_disconnect
Deassociating AP ...
ioctl[SIOCGIWESSID] ssid = NULL, not connected
WIFI disconnected
```

WEP mode

Command: wifi_connect wep_key key_id

The WEP key can be 5 ASCII characters for WEP 40 or 13 ASCII characters for WEP 104. The key ID should be 0, 1, 2 or 3. The following is an example to connect network by using WEP 40 with key ID 0.

```
# wifi_connect galex_mini 12345 0
Joining BSS ...
RTL871X: set group key to hw: alg:1(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:0

RTL871X: set ssid [galex_mini]

ioctl[SIOCGIWESSID] ssid = NULL, not connected
RTL871X: start auth

RTL871X: auth success, start assoc

RTL871X: association success

galex_mini connected
```

4.2.4 Wi-Fi Information

The wifi_info command can be used to get the information of Wi-Fi driver, including some Wi-Fi statistic, setting, status and memory usage. The following is an example of the output of wifi_info command when Wi-Fi is disabled. The Wi-Fi status information shows the Wi-Fi driver is stopped. When Wi-Fi is stopped, tx/rx status is not shown.

```
# wifi_info
WIFI Status wlan0 (Stopped)
================================
[rltk_wlan_statistic] min_free_heap_size=22344, current heap free size=72696, wlan_heap_used=0
[rltk_wlan_statistic] max_skbbuf_used_num=4, skbbuf_used_num=0
[rltk_wlan_statistic] max_skbdata_used_num=3, skbdata_used_num=0
[rltk_wlan_statistic] max_timer_used_num=15

Interface (wlan0)
================================
  MAC => 00:e0:4c:87:00:03
  IP  => 192.168.1.194
  GW  => 192.168.1.1

WIFI Status wlan1 (Stopped)
================================

Interface (wlan1)
================================
  MAC => 02:e0:4c:87:00:03
  IP  => 192.168.43.1
  GW  => 192.168.43.1

 default netif:================================
  IP  => 192.168.43.1
[MEM] After do cmd, available heap 72696
```

The following is the output of wifi_info command when Wi-Fi driver is enabled and disconnected. The Wi-Fi status shows the WI-Fi driver is running without SSID connected. The wlan statistic includes the memory usage that wlan heap used.

```
WIFI  wlan0 Setting:
==============================
      MODE => STATION
      SSID =>
   CHANNEL => 1
  SECURITY => OPEN
  PASSWORD =>

WIFI Status wlan0 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=0, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=0, rx_dropped=0, rx_bytes=0
[rltk_wlan_statistic] min_free_heap_size=32016, current heap free size=32272, wlan_heap_used=23400
[rltk_wlan_statistic] max_skbbuf_used_num=1, skbbuf_used_num=0
[rltk_wlan_statistic] max_skbdata_used_num=1, skbdata_used_num=0
[rltk_wlan_statistic] max_timer_used_num=8

Interface (wlan0)
==============================
  MAC => 00:e0:4c:87:00:03
  IP  => 192.168.1.4
  GW  => 192.168.1.1

WIFI Status wlan1 (Stopped)
==============================

Interface (wlan1)
==============================
  MAC => 02:e0:4c:87:00:03
  IP  => 192.168.43.1
  GW  => 192.168.43.1

 default netif:==============================
  IP  => 192.168.1.4
```

The following is the output of wifi_info command when Wi-Fi is connected. Wi-Fi setting shows the Wi-Fi driver is in station mode and connecting to a SSID. The connection information in Wi-Fi setting also includes current channel and security.

```
# wifi_info

WIFI  wlan0 Setting:
==============================
      MODE => STATION
      SSID => tplink841_ben
   CHANNEL => 6
  SECURITY => OPEN
  PASSWORD =>

WIFI Status wlan0 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=6, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=21, rx_dropped=190, rx_bytes=2589
[rltk_wlan_statistic] min_free_heap_size=22344, current heap free size=23328, wlan_heap_used=27704
[rltk_wlan_statistic] max_skbbuf_used_num=4, skbbuf_used_num=0
[rltk_wlan_statistic] max_skbdata_used_num=3, skbdata_used_num=0
[rltk_wlan_statistic] max_timer_used_num=15

Interface (wlan0)
==============================
  MAC => 00:e0:4c:87:00:03
  IP  => 192.168.1.194
  GW  => 192.168.1.1


WIFI  wlan1 Setting:
==============================
      MODE => AP
      SSID => dddd
   CHANNEL => 6
  SECURITY => OPEN
  PASSWORD =>

WIFI Status wlan1 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=0, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=0, rx_dropped=188, rx_bytes=0

Interface (wlan1)
==============================
  MAC => 02:e0:4c:87:00:03
  IP  => 192.168.43.1
  GW  => 192.168.43.1

 default netif:==============================
  IP  => 192.168.43.1
```

4.2.5 Start AP

The Wi-Fi driver can be switched from station mode to AP mode. The wifi_ap command can be used to start a Wi-Fi AP with indicated SSID, channel and password. Only typing the command without parameters shows the command usage. If password is not given, this command starts AP in open mode. Otherwise, it starts AP with WPA2 security.

```
# wifi_ap
Usage: wifi_ap SSID CHANNEL [PASSWORD]

# wifi_ap galex_ap 6 1234567890
Starting AP ...
galex_ap started
```

The following is the output of wifi_info command when AP mode. The Wi-Fi setting shows the Wi-Fi driver is operating in AP mode with SSID, channel, security.

```
# wifi_info

WIFI Setting:
==============================
      MODE => AP
      SSID => galex_ap
   CHANNEL => 6
  SECURITY => WPA2
  PASSWORD => 1234567890

WIFI Status (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=0, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=0, rx_dropped=30, rx_bytes=0
[rltk_wlan_statistic] min_free_heap_size=73672, current heap free size=74672, wlan_heap_used=48744
[rltk_wlan_statistic] max_skbbuf_used_num=1, skbbuf_used_num=0
[rltk_wlan_statistic] max_skbdata_used_num=1, skbdata_used_num=0
[rltk_wlan_statistic] max_timer_used_num=8
```

To switch back from AP to STA mode, set wifi_connect command.

4.2.6 Start STA+AP

The Wi-Fi driver can start station mode and AP mode concurrently. The “wifi_sta_ap” command can be used to start a Wi-Fi AP with indicated SSID, channel and password and start a station mode together. Only typing the command without parameters shows the command usage. If password is not given, this command starts AP in open mode. Otherwise, it starts AP with WPA2 security. And the “wifi_connect” command is used to connect with a AP.

```
# wifi_sta_ap aptest 6 12344321
```

```
# wifi_connect netgare 12345678
```

```
WIFI  wlan0 Setting:
==============================
      MODE => STATION
      SSID => netgaretim
   CHANNEL => 10
  SECURITY => WPA2
  PASSWORD => 12345678

WIFI Status wlan0 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=7, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=13, rx_dropped=156, rx_bytes=2054
[rltk_wlan_statistic] min_free_heap_size=7368, current heap free size=8352, wlan_heap_used=42602
[rltk_wlan_statistic] max_skbbuf_used_num=4, skbbuf_used_num=0
[rltk_wlan_statistic] max_skbdata_used_num=3, skbdata_used_num=0
[rltk_wlan_statistic] max_timer_used_num=15

Interface (wlan0)
==============================
  MAC => 00:e0:4c:87:00:03
  IP  => 192.168.1.4
  GW  => 192.168.1.1


WIFI  wlan1 Setting:
==============================
      MODE => AP
      SSID => aptest
   CHANNEL => 6
  SECURITY => WPA2
  PASSWORD => 12344321

WIFI Status wlan1 (Running)
==============================
[rltk_wlan_statistic] tx stat: tx_packets=0, tx_dropped=0, tx_bytes=0
[rltk_wlan_statistic] rx stat: rx_packets=0, rx_dropped=152, rx_bytes=0

Interface (wlan1)
==============================
  MAC => 02:e0:4c:87:00:03
  IP  => 192.168.43.1
  GW  => 192.168.43.1

 default netif:==============================
  IP  => 192.168.43.1
```

4.2.7 Ping

The ping command continues sending 5 ping packets, each in one second, to an indicated IP address. Please note that if DHCP client is not enabled, it is required to pre-configured default IP in main.h. It is useful when testing the network connection.

```
# ping 192.168.1.1
[ping_test] PING 192.168.1.1 120(148) bytes of data
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=1 time=49 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=2 time=5 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=3 time=14 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=4 time=10 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=5 time=23 ms
```

To ping y packets, type ping xxx.xxx.xxx.xxx y

```
# ping 192.168.1.1 6
[ping_test] PING 192.168.1.1 120(148) bytes of data
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=1 time=28 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=2 time=154 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=3 time=52 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=4 time=40 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=5 time=48 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=6 time=78 ms
```

To ping continuously, type *ping* ***xxx.xxx.xxx*** ***loop***. Please note that currently, exiting infinite ping loop by UART command is not supported yet.

```
# ping 192.168.1.1 loop
[ping_test] PING 192.168.1.1 120(148) bytes of data
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=1 time=61 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=2 time=15 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=3 time=34 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=4 time=23 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=5 time=25 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=6 time=31 ms
[ping_test] 128 bytes from 192.168.1.1: icmp_seq=7 time=55 ms
```

4.2.8 TCP RX/TX Throughput Test

4.2.8.1 TCP Test

TCP transmit and receive throughput can be measured by iperf.exe tool which you can get from $sdk/tools/iperf.exe.

4.2.8.1.1 Receive Throughput Test

Receive test measures receive throughput of the development board. Start TCP server in the development board, listen to port 5001 and wait for connection from iperf client. Iperf on the Windows platforms connects to the TCP server via AP and transmits data to it. Iperf client running on the Windows platforms computes bytes of data transmitted, and print it out every 1 second. A sample session is illustrated as bellow:

Type the following command to start TCP server on the console of development board:

tcp -s

The “-s” command-line option starts a TCP server.

```
# tcp -s
[MEM] After do cmd, available heap 36152


#
TCP: Start tcp Server!
TCP: Create server socket 0

TCP: Bind successfully.
TCP: Listen port 5001
TCP: Accept socket 1 successfully.
TCP: Recieved 3637 packets successfully.
```

Type the following command to start Iperf client on Windows platforms:

~:> iperf .exe –c 192.168.0.102 –i 1 –t 60

The “-c” command-line option means starting a TCP client and connecting to “192.168.0.102”, “-i” is seconds between periodic bandwidth reports, “-t” is time in seconds to transmit for (default 10 seconds).

```
G:\yangjue\iot_sdk\v03_1\tools>iperf.exe -c 192.168.0.102 -i 1 -t 10
------------------------------------------------------------
Client connecting to 192.168.0.102, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[128] local 192.168.0.100 port 53222 connected with 192.168.0.102 port 5001
[ ID] Interval       Transfer     Bandwidth
[128]  0.0- 1.0 sec   528 KBytes  4.33 Mbits/sec
[128]  1.0- 2.0 sec   352 KBytes  2.88 Mbits/sec
[128]  2.0- 3.0 sec   408 KBytes  3.34 Mbits/sec
[128]  3.0- 4.0 sec   632 KBytes  5.18 Mbits/sec
[128]  4.0- 5.0 sec   584 KBytes  4.78 Mbits/sec
[128]  5.0- 6.0 sec   464 KBytes  3.80 Mbits/sec
[128]  6.0- 7.0 sec   672 KBytes  5.51 Mbits/sec
[128]  7.0- 8.0 sec   576 KBytes  4.72 Mbits/sec
[128]  8.0- 9.0 sec   520 KBytes  4.26 Mbits/sec
[128]  9.0-10.0 sec   440 KBytes  3.60 Mbits/sec
[128]  0.0-10.0 sec  5.06 MBytes  4.23 Mbits/sec
```

4.2.8.1.2 Transmit Throughput Test

Transmit test measures the transmission throughput of the development board. Start TCP Client in the development board and connect to Iperf server on the Windows platforms via AP. Iperf server works on the default port 5001 and should not be changed since TCP client is fixed to connect with this port. TCP client send 10000 packets with length 1460 one time as default. Iperf server running on the Windows platforms computes bytes of data received, and print it out every 1 second. A sample session is illustrated as below:

Type the following command to start Iperf server on Windows platforms:

~:> iperf.exe –s –i 1

The “-s” command-line option starts a TCP server, “-i” is seconds between periodic bandwidth reports.

```
G:\yangjue\iot_sdk\v03_1\tools>
G:\yangjue\iot_sdk\v03_1\tools>iperf.exe -s -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[212] local 192.168.0.100 port 5001 connected with 192.168.0.102 port 4101
[ ID] Interval       Transfer     Bandwidth
[212]  0.0- 1.0 sec   167 KBytes  1.37 Mbits/sec
[212]  1.0- 2.0 sec   482 KBytes  3.95 Mbits/sec
[212]  2.0- 3.0 sec   684 KBytes  5.61 Mbits/sec
[212]  3.0- 4.0 sec   540 KBytes  4.43 Mbits/sec
[212]  4.0- 5.0 sec   416 KBytes  3.41 Mbits/sec
[212]  5.0- 6.0 sec   431 KBytes  3.53 Mbits/sec
[212]  6.0- 7.0 sec   630 KBytes  5.16 Mbits/sec
[212]  7.0- 8.0 sec   669 KBytes  5.48 Mbits/sec
[212]  0.0- 8.4 sec  4.18 MBytes  4.17 Mbits/sec
```

Type the following command to start TCP client on the development board:

tcp –c 192.168.0.100 1500 3000

The “-c” command-line option starts a TCP client, “192.168.0.100” is IP address of the Windows platforms, “1500” is the length of packet to be transmitted, “3000” is the number of packets transmitted to Iperf Server. Please note that packet length is no more than 4300 .

```
# tcp -c 192.168.0.100 1500 3000
[MEM] After do cmd, available heap 41128


#
TCP: Start tcp client!
TCP: ServerIP=192.168.0.100 port=5001.
TCP: Create socket 0.
TCP: Connect server successfully.
TCP: Sent 3000 packets successfully.
TCP: Tcp client stopped!
```

4.2.8.1.3 Transmit and Receive Throughput Test

The concurrent throughput test measures receive and transmit throughput concurrently. The development board run “tcp -s” to start a TCP server and communicate with iperf client on Windows platform, run “tcp –c 192.168.0.100 1500 100000” to start a TCP client and communicate with iperf server on Windows platform. A sample session is illustrated as bellow:

Step 1: Start Iperf server on Windows platforms:

~:> iperf.exe -s -i 1

Step 2: Start TCP server on the development board:

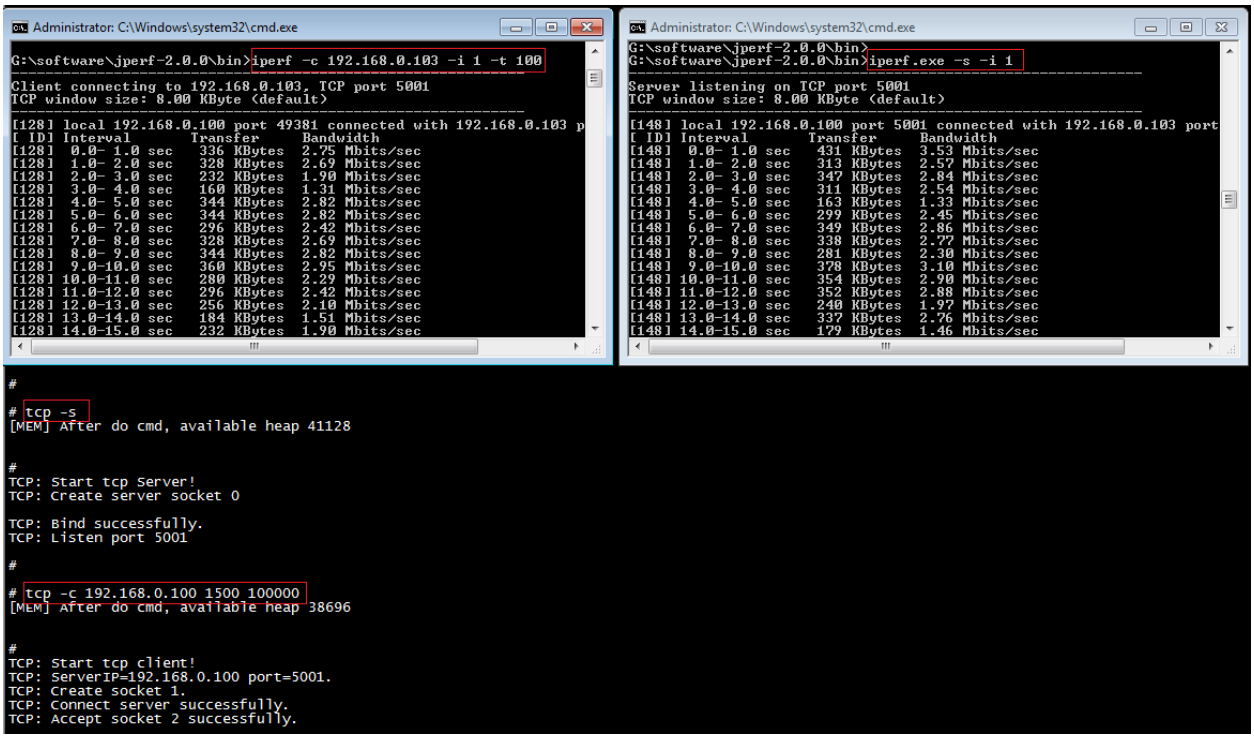
tcp -s

Step 3: Start Iperf client on Windows platforms:

~:> iperf.exe –c 192.168.0.103 –i 1 –t 100

Step 4: Start TCP client on the development board:

tcp –c 192.168.0.100 1500 100000



4.2.9 Start Web Server

The wifi_start_webserver command can be used to start webserver. Web server works only after Wi-Fi driver switched to AP mode or concurrent AP mode. After client associated with the AP and get right IP address, the client PC can open web browser and enter http://192.168.1.1 (http://192.168.1.1 in AP mode or http://192.168.43.1 in concurrent AP mode) to get or set AP settings. For details, please refer to the document UM0014 Realtek web server user guide.pdf.

4.2.10 Wi-Fi Simple Config

This wifi_simple_config command provides a simple way for device to associate to AP. For details, please refer to the document AN0011 Realtek wlan simple configuration.pdf.

4.2.11 Wi-Fi Protected Setup

The wifi_wps command provides another simple way for device to associate to AP. After pressing WPS button on the AP, execute “wifi_wps pbc” in the command line, then the device will automatically associate with the AP. PIN method also supported. Please refer to the document AN0011 Realtek wlan simple configuration.pdf for more detail.

4.2.12 Exit

The exit command makes leaving from UART interactive mode. The stack used by interactive task is released to get more memory.

```
# exit
Leave INTERACTIVE MODE
```