

WPS station Application

User guide

Version 0.2

May 2016

Redpine Signals, Inc.

2107 N. First Street, #540

San Jose, CA 95131.

Tel: (408) 748-3385

Fax: (408) 705-2019

Email: info@redpinesignals.com

Website: www.redpinesignals.com

About this Document

This document describes the process of bringing up the RS9113 based module as a WPS enabled station.

Disclaimer:

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only. Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

Copyright © 2015 Redpine Signals, Inc. All rights reserved.

Table of Contents

1	Introduction	4
1.1	WPS Overview	4
1.2	Application Overview	4
1.2.1	Overview	4
1.2.2	Sequence of Events	4
1.3	Application Setup	4
1.3.1	SPI based Setup Requirements	4
1.3.2	UART/USB-CDC based Setup Requirements	5
2	Configuration and Execution of the Application	6
2.1	Initializing the Application	6
2.1.1	SPI Interface	6
2.1.2	UART/USB-CDC Interface	6
2.2	Configuring the Application	6
2.3	Executing the Application	7

Table of Figures

Figure 1: Setup Diagram	5
-------------------------------	---

Table of Tables

No table of figures entries found.

1 Introduction

This project is applicable to all the WiSeConnect variants like WiSeConnect Plus, WiSeMCU and WyzBee. The term WiSeConnect refers to its appropriate variant.

1.1 WPS Overview

WPS (WiFi Protected setup) is a network standard to create a secure wireless home network using PUSH button and PIN button method.

In Push button method, in which the user has to push a button, either an actual or virtual one, on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

In PIN method, in which the user has to enter secret PIN on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

1.2 Application Overview

1.2.1 Overview

WiSeConnect device supports both Push button method and wps pin method to connect to an Access point

The WPS station application demonstrates how WiSeConnect device would be connected to secured (WPA2) Access point using WPS PUSH method.

In this application, after successful connection with the access point using WPS PUSH button method, WiSeConnect device opens TCP socket and sends TCP data to configured remote peer.

1.2.2 Sequence of Events

This Application explains user how to:

- Configure device as WPS enable station
- Enable WPS PUSH button method in Access Point
- Connect to WPS enabled Access Point using PUSH button method
- Open TCP client socket after successful connection
- Establish TCP connection with the TCP server opened in remote peer.
- Send TCP data from WiSeConnect device to remote peer.

1.3 Application Setup

The WiSeConnect in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

1.3.1 SPI based Setup Requirements

- Windows PC1 with Coocox IDE
- Spansion (MB9BF568NBGL) micro controller

Note: If user does not have Spansion (MB9BF568NBGL) host platform, please go through the SPI-Porting guide [\sapis\docs\RS9113-WiSeConnect-SAPI-Porting-Guide-vx.x.pdf](#) for SAPIs porting to that particular platform.

- WiSeConnect device
- WPS supported Access Point
- Windows PC2
- A TCP server application running on the Windows PC2 (This example uses iperf for windows)

1.3.2 UART/USB-CDC based Setup Requirements

- Windows PC1 with Dev-C++ IDE
- WiSeConnect device
- WPS supported Access Point
- Windows PC2
- A TCP server application running on the Windows PC2 (This example uses iperf for windows)

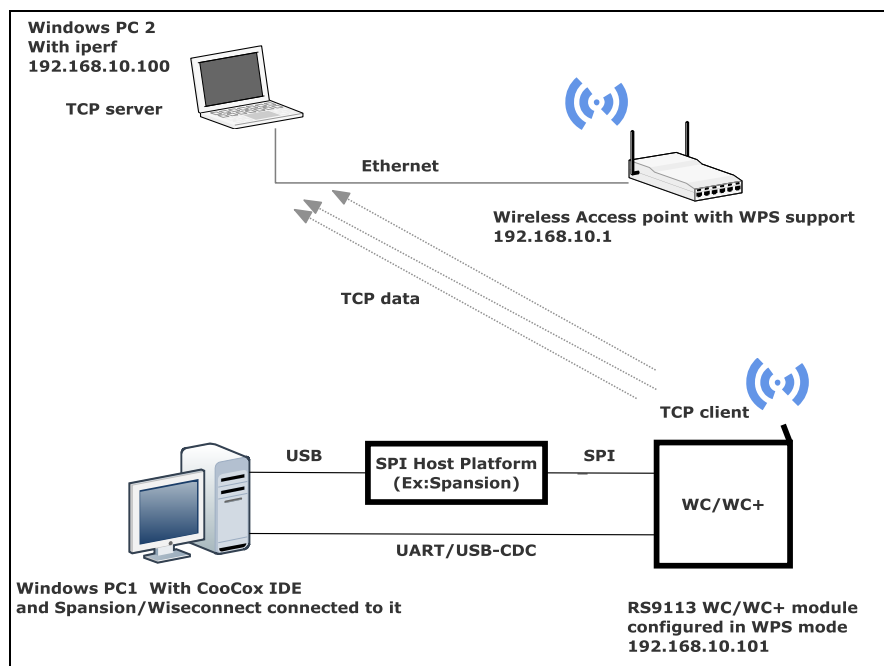


Figure 1: Setup Diagram

2 Configuration and Execution of the Application

The example application is available in the Release at `{Release $}/host/sapis/examples`. These examples will have to be initialized, configured and executed to test the application. The initialization varies based on the interface but configuration and execution are the common.

2.1 Initializing the Application

2.1.1 SPI Interface

If User using SPI interface, Please refer the document *sapis/platforms/spansion_MB9BF568NBGL/RS9113-WiSeConnect_SAPIS_Spansion_Project_User_guide.pdf* for opening the *wps_station* example in Coocox IDE.

2.1.2 UART/USB-CDC Interface

If User using UART interface, Please refer the document *sapis/platforms/windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf* for opening the *wps_station* example in Dev-C++ IDE

2.2 Configuring the Application

1. Open *sapis/examples/wlan/wps_station/rsi_wps_station.c* file and update/modify following macros :

SSID refers to the name of the Access point. In WPS method it should NULL.

```
#define SSID NULL
```

SECURITY_TYPE refers to the type of security. In WPS method WiSeConnect module supports WPS push button method and WPS PIN method.

In this examples, WiSeConnect module connecting to AP using WPS push button method. So, set **SECURITY_TYPE** macro to **RSI_WPS_PUSH_BUTTON**.

```
#define SECURITY_TYPE RSI_WPS_PUSH_BUTTON
```

PSK refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes. In WPS method PSK is not needed.

```
#define PSK NULL
```

DEVICE_PORT port refers TCP client port number

```
#define DEVICE_PORT <local port>
```

SERVER_PORT port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT <remote port>
```

SERVER_IP_ADDRESS refers remote peer IP address to connect with TCP server socket.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.100" as IP address, update the macro **DEVICE_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

NUMEBR_OF_PACKETS refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS    <no of packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN      8000
```

To configure IP address

DHCP_MODE refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE    1
```

Note: If user wants to configure STA IP address through DHCP then set **DHCP_MODE** to 1 and skip configuring the following **DEVICE_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP_MODE** macro to "0" and configure following **DEVICE_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP    0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY    0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK    0x00FFFFFF
```

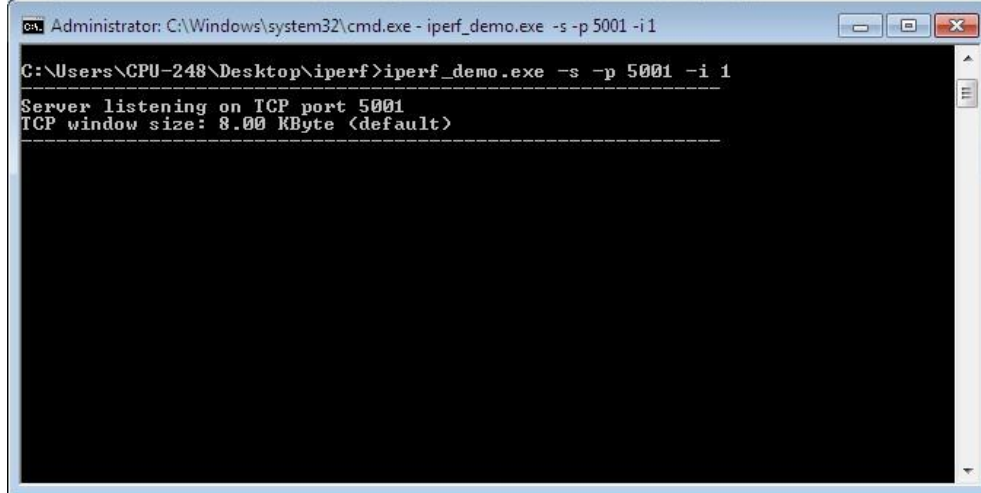
2. Open *sapis/include/rsi_wlan_config.h* file and update/modify following macros :

```
#define CONCURRENT_MODE    RSI_DISABLE
#define RSI_FEATURE_BIT_MAP    FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS    RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP    TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP    0
#define RSI_BAND    RSI_BAND_2P4GHZ
```

2.3 Executing the Application

1. Configure the Access point in WPA2-PSK mode to connect WiSeConnect device in STA mode.
2. Open TCP server application using iperf application in Windows PC1 which is connected to Access point through LAN.

```
iperf_demo.exe -s -p <SERVER_PORT> -i 1
```



3. Press the WPS Push button on Access point and it lasts for 2 minutes.

4. SPI Interface

If User using SPI interface, Please refer the document *sapis/platforms/spansion_MB9BF568NBGL/RS9113-WiSeConnect_SAPIS_Spansion_Project_User_guide.pdf* for executing the *wps_station* example in CooCox IDE.

5. UART/USB-CDC Interface

If User using UART interface, Please refer the document *sapis/platforms/windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf* for executing the *wps_station* example in Dev-C++ IDE

6.

7. After program gets executed, WiSeConnect Device would scan and connect to Access point using WPS push button method and get IP.

8. After successful connection, WiSeConenct STA connects to TCP server socket opened on Windows PC1 using TCP client socket and sends configured **NUMBER_OF_PACKETS** to remote TCP server. Please refer the given below image for reception of TCP data on TCP server,


```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1

C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
-----
[[1244]] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval      Transfer    Bandwidth
[[1244]] 0.0- 1.0 sec  2.04 KBytes 16.7 Kbits/sec
[[1244]] 1.0- 2.0 sec  504 Bytes  4.03 Kbits/sec
[[1244]] 2.0- 3.0 sec  1.99 KBytes 16.3 Kbits/sec
[[1244]] 3.0- 4.0 sec  2.11 KBytes 17.3 Kbits/sec
[[1244]] 4.0- 5.0 sec  3.47 KBytes 28.4 Kbits/sec
[[1244]] 5.0- 6.0 sec  0.00 Bytes  0.00 bits/sec
[[1244]] 6.0- 7.0 sec  96.0 Bytes  768 bits/sec
[[1244]] 7.0- 8.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 8.0- 9.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 9.0-10.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 10.0-11.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 11.0-12.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 12.0-13.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 13.0-14.0 sec  24.0 Bytes  192 bits/sec
[[1244]] 14.0-15.0 sec  0.00 Bytes  0.00 bits/sec
[[1244]] 15.0-16.0 sec  1.20 KBytes  9.79 Kbits/sec
```