# RS9113 AP Start Application

## User Guide

## Version 0.2

## May 2016

**Redpine Signals, Inc.**
2107 N. First Street, #680
San Jose, CA95131.
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 an AP.

**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.

# Table of Contents



# Table of Figures



# 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 Application Overview

### 1.1.1 Overview

The AP start application demonstrates how to configure the WiSeConnect device as a soft Access point and allows stations to connect to it. The application also enables TCP data transmission from the connected WiFi station to WiSeConnect Access Point.

### 1.1.2 Sequence of Events

This Application explains user how to:

- Create WiSeConnect device as Soft Access point
- Open TCP server socket on configured port number on the device.
- Connect WiFi Station to WiSeConnect Access point
- Establish TCP connection from connected WiFi Station to TCP server opened on WiSeConnect Access Point.
- Send TCP data from Connected station to WiSeConnect Access point
- Read configured number of TCP data packets sent by connected WiFi station.

## 1.2 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:

### 1.2.1 SPI based Setup Requirements

- Windows PC 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-WiSeConnsect-SAPI-Porting-Guide-vx.x.pdf** for SAPIs porting to that particular platform.

- WiSeConnect device
- A Mobile device as a Wi-Fi station (This example uses a windows Laptop)
- A TCP client application running on the Wi-Fi station (This example uses iperf for windows )

### 1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- A Mobile device as a Wi-Fi station (This example sues a windows Laptop)
- A TCP client application running on the Wi-Fi station (This example uses iperf for windows )

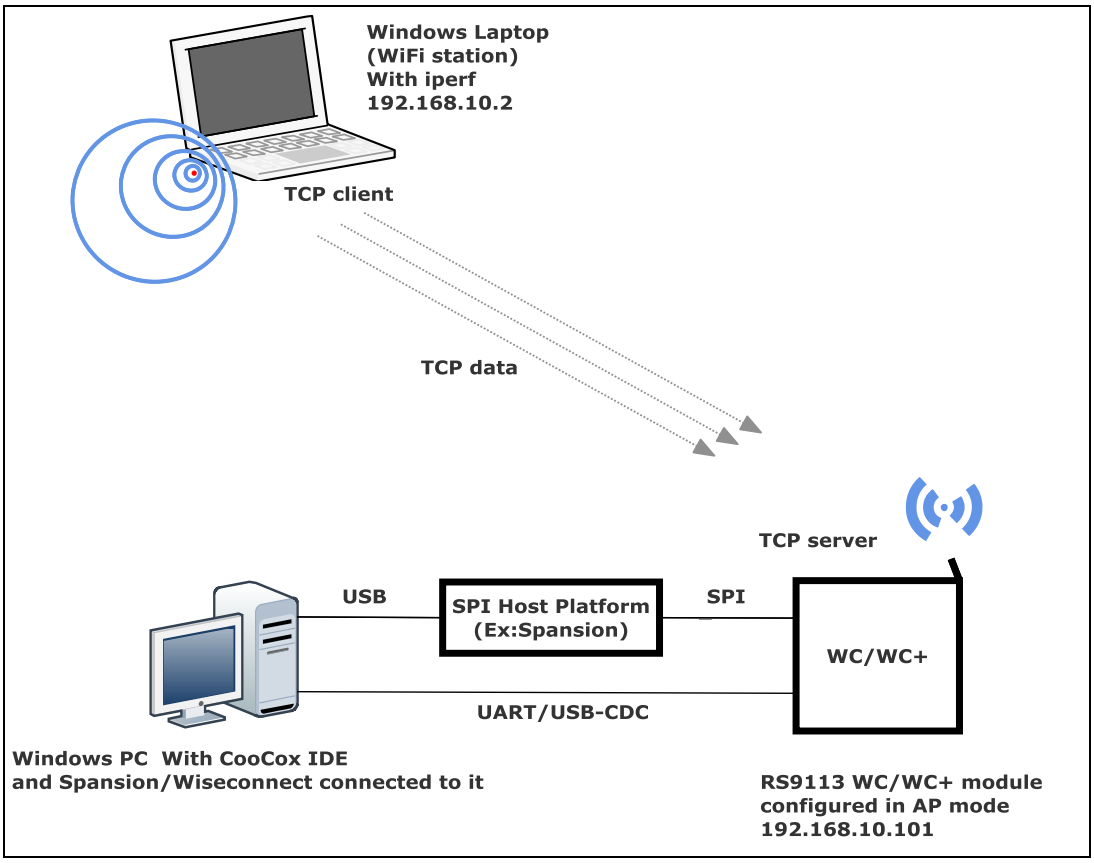



**Figure 1 Setup diagram**

# 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 ***access_point*** 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 ***access_point*** example in Dev-C++ IDE

## 2.2 Configuring the Application

1. Open ***sapis/examples/access_point/rsi_ap_start.c*** file and update/modify the following

macros.

**SSID** refers to the name of the Access point.

```
#define SSID                    "REDPINE_AP"
```

**CHANNEL_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO              11
```

Note: Valid values for CHANNEL_NO in 2.4GHz band are 1 to 11 and 5GHZ band are 36 to 48 and 149 to 165. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set RSI_BAND macro to 5GHz band in ***sapis/include/rsi_wlan_config.h*** file.

**SECURITY_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configurations are:

**RSI_OPEN** - For OPEN security mode

**RSI_WPA** - For WPA security mode

**RSI_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE           RSI_WPA2
```

**ENCRYPTION_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP encryption methods.

Valid configurations are:

**RSI_CCMP** - For CCMP encryption

**RSI_TKIP** - For TKIP encryption

**RSI_NONE** - For open encryption

```
#define ENCRYPTION_TYPE      RSI_CCMP
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK                  "12345678"
```

**BEACON_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL        100
```

**DTIM_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL          4
```

**DEVICE_PORT** port refers TCP server port number

```
#define DEVICE_PORT          5001
```

**NUMEBR_OF_PACKETS** refers how many packets to receive from remote TCP client.

```
#define NUMBER_OF_PACKETS      1000
```

**RECV_BUFFER_SIZE** refers receive data length

```
#define RECV_BUFFER_SIZE  1000
```

**To configure IP address**

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

Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE_IP** as **0x010AA8C0**.

```
#define  DEVICE_IP          0X010AA8C0
```

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
```

Note: In AP mode, configure same IP address for both `DEVICE_IP` and `GATEWAY` macros

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_PSK
#define  RSI_TCP_IP_BYPASS          RSI_DISABLE
```

```
#define  RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_DHCPV4_SERVER
#define  RSI_CUSTOM_FEATURE_BIT_MAP  0
#define  RSI_BAND                    RSI_BAND_2P4GHZ
```

## 2.3 Executing the Application

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 executing the ***access_point*** example in CooCox IDE.

2. **UART/USB-CDC Interface**

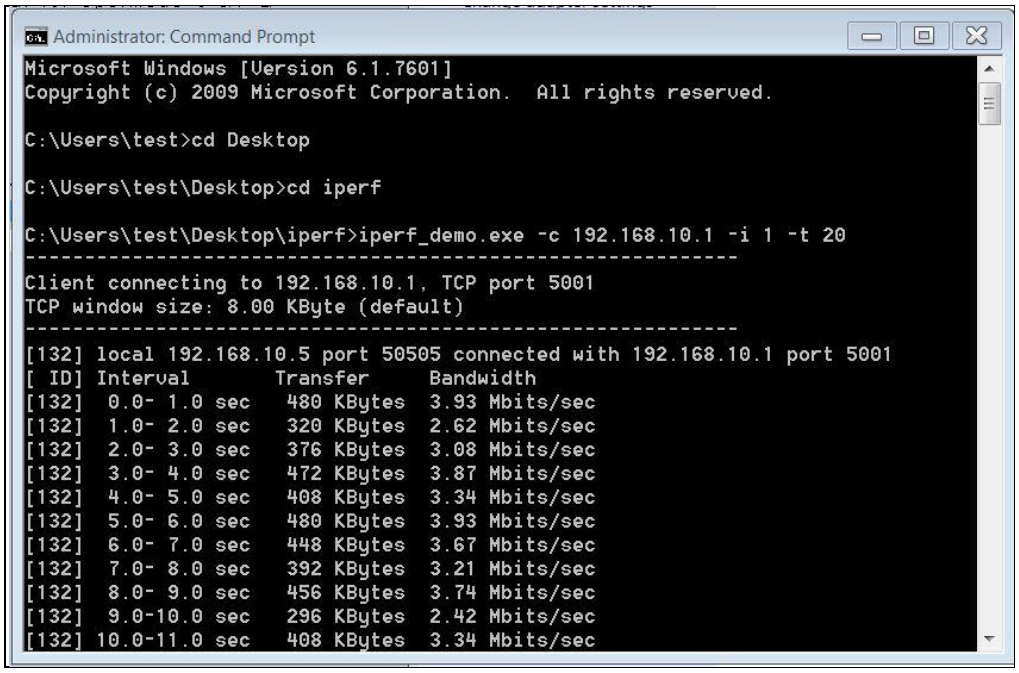
   If User using UART interface, Please refer the document ***sapis/platforms/windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf*** for executing the ***access_point*** example in Dev-C++ IDE

3. After the program gets executed, WiSeConnect Device will be created as Access point with configured **`SSID`** ( Ex: "**`REDPINE_AP`**")  and opens TCP server socket on **`DEVICE_PORT`**  and waits for TCP connection request from TCP client.

   Now scan and connect to WiSeConnect Access Point (Ex: "REDPINE_AP" is the AP name) from Laptop.


4. After successful connection, open iperf client from Laptop and connect to TCP server running on AP with port number **`DEVICE_PORT`**  using following command.

   **iperf.exe –c <DEVICE_IP> -p <DEVICE_PORT> -i 1 –t 100**




5. The device will accept connection request and receive data on the TCP server port and exit after receiving configured **NUMBER_OF_PACKETS.**