Raw Data 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 an AP and used for raw data packets transmission.

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 raw data application demonstrates how WiSeConnect device receives the raw data packets (packets of other IP network) and sends them to host, and also how it receives raw data packets from host and sends on air.

In this Application WiSeConnect device will be created as Access point, allow WiFi stations to connect to it. It processes the ARP request packet (raw data) and sends ARP response (raw data). It also process ping request (raw data) of other IP network, and sends ping response (raw data) to it.

1.1.2 Sequence of Events

This Application explains user how to:

- WiSeConnect Device starts as an Access point
- Allow stations to connect
- Reply for ping request and ARP request of other networks also

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

- WiSeConnect device
- Windows Laptop for WiFi Station

1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Windows Laptop for WiFi Station

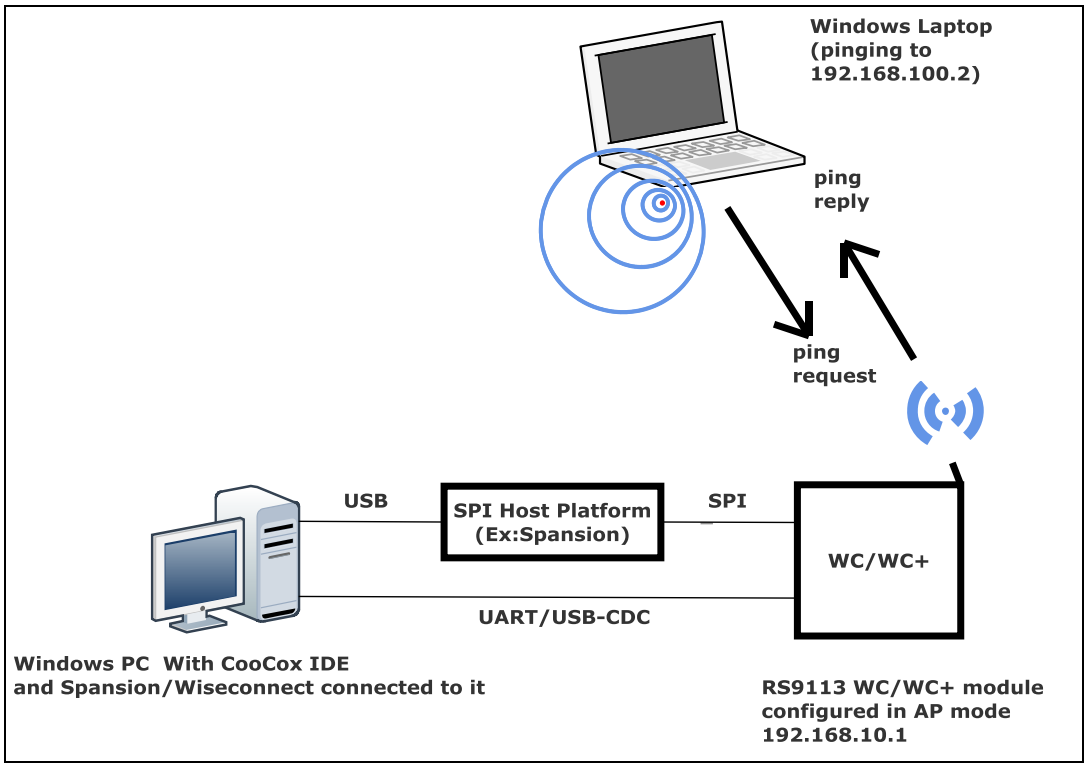


Figure 1:Setup Diagram

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 ***raw_data*** 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 ***raw_data*** example in Dev-C++ IDE

2.2 Configuring the Application

1. Open ***sapis/examples/raw_data/rsi_raw_data_app.c*** file and update/modify following macros,

SSID refers to the name of the Access point.

```
#define SSID                    "<ap name>"
```

CHANNEL_NO refers to the channel in which AP would be started

```
#define CHANNEL_NO              11
```

Note:

Valid values for **CHANNEL_NO** are 1 to 11 in 2.4GHz and 36 to 48 & 149 to 165 in 2.4GHz. 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 configuration is:

RSI_OPEN - For OPEN security mode

RSI_WPA - For WPA security mode

RSI_WPA2 - For WPA2 security mode

```
#define SECURITY_TYPE           RSI_OPEN
```

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

Valid configuration is:

RSI_CCMP - For CCMP encryption

RSI_TKIP - For TKIP encryption

RSI_NONE - For open encryption

```
#define ENCRYPTION_TYPE      RSI_NONE
```

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

```
#define PSK                  "<psk>"
```

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
```

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             DISABLE
#define  RSI_FEATURE_BIT_MAP         FEAT_SECURITY_PSK
#define  RSI_TCP_IP_BYPASS           DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP  (TCP_IP_FEAT_DHCPV4_SERVER
                                    | TCP_IP_FEAT_RAW_DATA)
#define  RSI_CUSTOM_FEATURE_BIT_MAP  0
#define  RSI_BAND                    RSI_BAND_2P4GHZ
```

2.3 Executing the Application

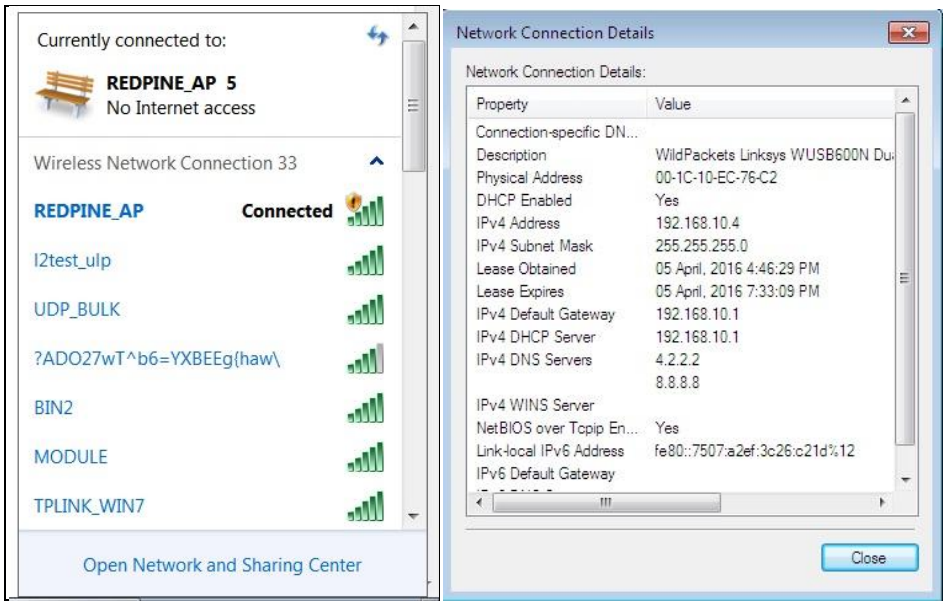
1. **SPI Interface**

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

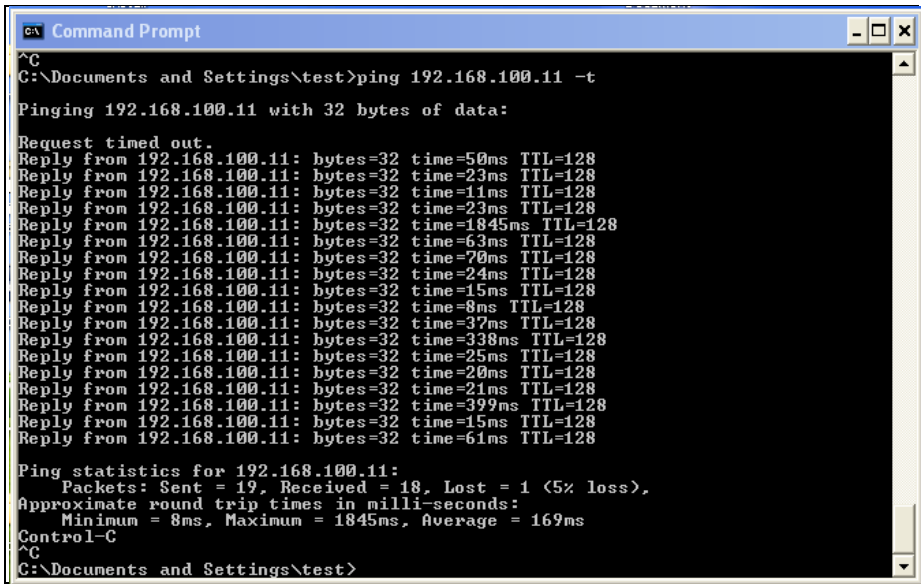
3. After the program gets executed, WiSeConnect Device will be created as Access point and starts beaconing.
4. Now connect WiFi STA (Laptop) to WiSeconnect AP (Ex: AP SSID is "REDPINE_AP"). After successful connection, WiFi STA gets IP in the configured IP network (Ex: 192.168.10.4)



5. Initiate ping to an IP of other network (Ex: 192.168.100.11) from WiFi STA (laptop).

Ping 192.168.100.11 –t

6. Module will reply with ARP response, if connected stations try to ping other IP (which is not in a connected network) and also responds with ping reply for the prior resolved ARP.


Command Prompt
^C
C:\Documents and Settings\test>ping 192.168.100.11 -t
Pinging 192.168.100.11 with 32 bytes of data:
Request timed out.
Reply from 192.168.100.11: bytes=32 time=50ms TTL=128
Reply from 192.168.100.11: bytes=32 time=23ms TTL=128
Reply from 192.168.100.11: bytes=32 time=11ms TTL=128
Reply from 192.168.100.11: bytes=32 time=23ms TTL=128
Reply from 192.168.100.11: bytes=32 time=1845ms TTL=128
Reply from 192.168.100.11: bytes=32 time=63ms TTL=128
Reply from 192.168.100.11: bytes=32 time=70ms TTL=128
Reply from 192.168.100.11: bytes=32 time=24ms TTL=128
Reply from 192.168.100.11: bytes=32 time=15ms TTL=128
Reply from 192.168.100.11: bytes=32 time=8ms TTL=128
Reply from 192.168.100.11: bytes=32 time=37ms TTL=128
Reply from 192.168.100.11: bytes=32 time=338ms TTL=128
Reply from 192.168.100.11: bytes=32 time=25ms TTL=128
Reply from 192.168.100.11: bytes=32 time=20ms TTL=128
Reply from 192.168.100.11: bytes=32 time=21ms TTL=128
Reply from 192.168.100.11: bytes=32 time=399ms TTL=128
Reply from 192.168.100.11: bytes=32 time=15ms TTL=128
Reply from 192.168.100.11: bytes=32 time=61ms TTL=128
Ping statistics for 192.168.100.11:
Packets: Sent = 19, Received = 18, Lost = 1 (5% loss),
Approximate round trip times in milli-seconds:
Minimum = 8ms, Maximum = 1845ms, Average = 169ms
Control-C
^C
C:\Documents and Settings\test>