Station Ping Application

User guide

Version 0.2

May 2016

Redpine Signals, Inc.
2107 N. First Street, #680
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 WiFi station and sends PING requests to the desired IP.

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 PING Overview

Ping is used diagnostically to ensure that a host computer the user is trying to reach is actually operating. Ping works by sending an Internet Control Message Protocol (ICMP) Echo Request to a specified interface on the network and waiting for a reply. Ping can be used for troubleshooting to test connectivity and determine response time.

1.2 Application Overview

1.2.1 Overview

The application demonstrates how to configure WiSeConnect device in client mode to send ping request to target IP address.

1.2.2 Sequence of Events

This Application explains user how to:

- Connect to Access Point in station mode
- Send Ping requests to configured target IP address

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 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
- Wireless Access Point

1.3.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Wireless Access Point

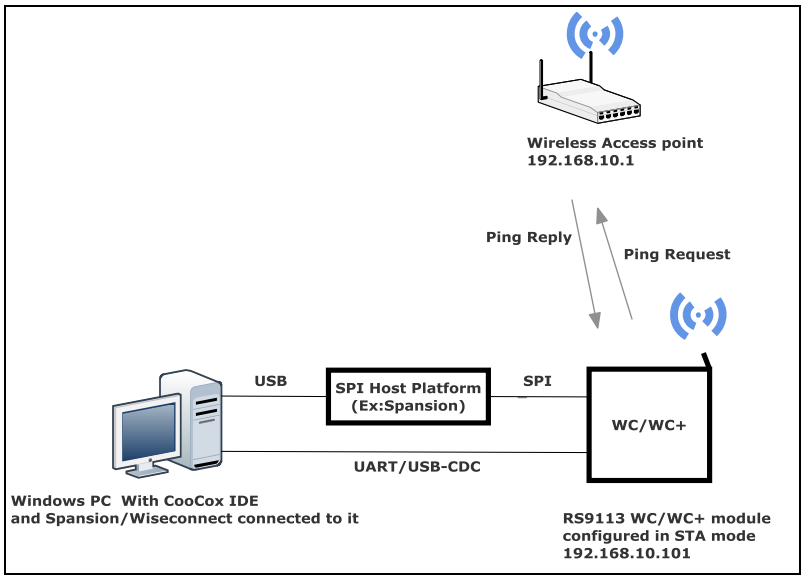

Figure 1: Setup Diagram

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

2.2 Configuring the Application

1. Open sapis/examples/wlan/station_ping/***rsi_station_ping.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 device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO          0
```

SECURITY_TYPE refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK 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
```

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

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

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
```

Configure following macors to ping initiate ping with the remote peer

IP address of the remote peer (AP IP address).

Example: To configure "192.168.10.1" as **REMOTE_IP**, update the macro **REMOTE_IP** as **0x0A0AA8C0**.

```
#define REMOTE_IP           0x010AA8C0
```

PING_SIZE refers the size of ping packet.

```
#define  PING_SIZE          100
```

NUMBER_OF_PACKETS refers how many number of pings to send from device.

```
#define NUMBER_OF_PACKETS   1000
```

Application memory length which is required by the driver

```
#define  GLOBAL_BUFF_LEN  8000
```

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
                                    | TCP_IP_FEAT_ICMP)
#define RSI_CUSTOM_FEATURE_BIT_MAP  0
```

```
#define RSI_BAND                              RSI_BAND_2P4GHZ
```

2.3 Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect WiSeConnect device in STA mode.
2. 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 ***station_ping*** example in CooCox IDE.
3. 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 ***station_ping*** example in Dev-C++ IDE
4. After the program gets executed, WiSeConnect module configured as client and connects to AP and gets IP.
5. After successful connection with the Access Point, Device starts sending ping requests to the given **`REMOTE_IP`** with configured **`PING_SIZE`** to check availability of target Device.
6. Device sends the number of ping packets configured in **`NUMBER_OF_PACKETS`**.
7. In rsi_station_ping.c file, **`rsi_wlan_ping_async`** API returns success status, which means that the ping request packet is successfully sent in to the medium. When actual ping response comes from the remote node, it is known from the status parameter of the callback function (**rsi_ping_response_handler**) registered in the Ping API.