

Store Configuration Profile Application

User guide

Version 0.1

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 storing the client/AP mode configuration profile in non-volatile memory of RS9113 based module.

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	Store Configuration Overview	4
1.1.1	In client mode	4
1.1.2	In Access Point mode.....	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	8

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 Store Configuration Overview

This feature is used to save the parameters into non-volatile memory which are used either to join to an Access point (auto-join mode) or to create an Access point (auto-create mode).

1.1.1 In client mode

The module can connect to a pre-configured access point after it boots up. This feature facilitates fast connection to a known network.

1.1.2 In Access Point mode

The module can be configured to come up as an Access Point every time it boots-up.

1.2 Application Overview

1.2.1 Overview

This application demonstrates how to store the given configuration in non-volatile memory and how to enable the stored configuration after bootup.

In this application, default configuration storing into the non-volatile memory is AP mode profile. After saving the configuration, application enables the configuration to get effective from next bootup/reset and then resets the device. After successful reset, WiSeConnect device starts as an Access point with the saved configuration. Now user can scan the Access point from station and connect to it and verify the connectivity by initiating Ping from connected station.

1.2.2 Sequence of Events

This Application explains user how to:

- Initialize the WiSeConnect device.
- Fill the configuration profile based on the selected mode (Ex: AP mode configuration)
- Add profile into non-volatile memory
- Get saved profile.
- Enable auto configuration
- Reset the device
- Initialize the device

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
- WiFi Access point

1.3.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- WiFi Access point

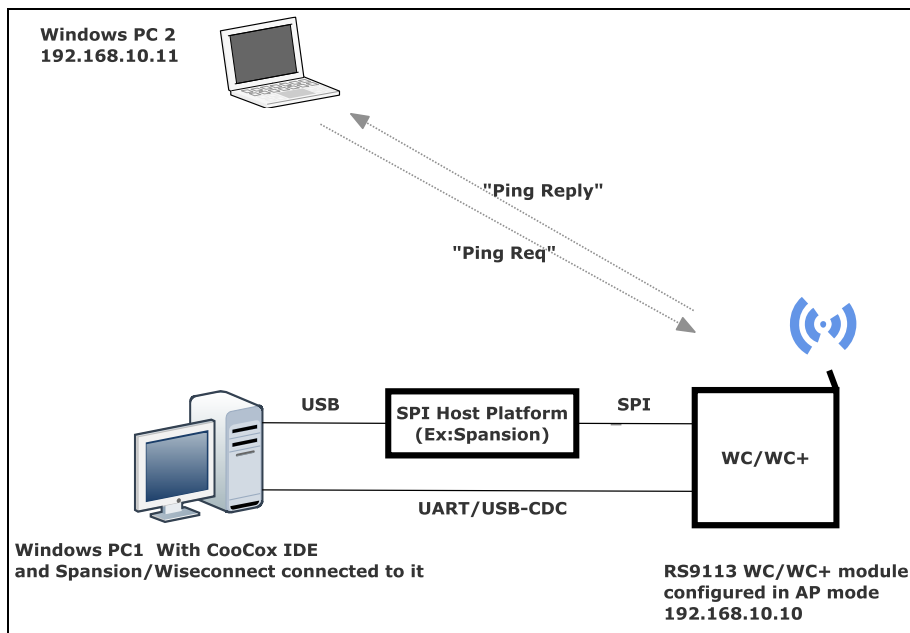


Figure 1: Setup Diagram

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

2.2 Configuring the Application

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

PROFILE_TYPE refers to the type of the profile storing into the non-volatile memory.

Valid configurations are,

RSI_WLAN_PROFILE_CLIENT	- 0
RSI_WLAN_PROFILE_P2P	- 1
RSI_WLAN_PROFILE_EAP	- 2
RSI_WLAN_PROFILE_AP	- 6

```
#define PROFILE_TYPE RSI_WLAN_PROFILE_AP
```

AUTO_CONFIG_ENABLE refers to enable or disable auto configuration.

Valid configurations are,

0 – to disable Auto configuration
2 – to enable Auto configuration

```
#define AUTO_CONFIG_ENABLE 2
```

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
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

3. Please find the **sapis/include/rsi_wlan_config.h** file for default configuration stored client/AP/EAP client/P2P mode profiles.

Following is the configuration parameters for AP mode profile.

```
//! Transmission data rate. Physical rate at which data has to be
transmitted.
#define RSI_CONFIG_AP_DATA_RATE RSI_DATA_RATE_AUTO
//! To set wlan feature select bit map
#define RSI_CONFIG_AP_WLAN_FEAT_BIT_MAP (FEAT_SECURITY_PSK)
//! TCP/IP feature select bitmap for selecting TCP/IP features
#define RSI_CONFIG_AP_TCP_IP_FEAT_BIT_MAP TCP_IP_FEAT_DHCPV4_SERVER
//! To set custom feature select bit map
#define RSI_CONFIG_AP_CUSTOM_FEAT_BIT_MAP 0
//! RSI_BAND_2P4GHZ(2.4GHz) or RSI_BAND_5GHZ(5GHz) or RSI_DUAL_BAND
#define RSI_CONFIG_AP_BAND RSI_BAND_2P4GHZ
//! Tx power level
#define RSI_CONFIG_AP_TX_POWER RSI_POWER_LEVEL_HIGH
//! AP SSID
#define RSI_CONFIG_AP_SSID "REDPINE_AP"
//! RSI_BAND_2P4GHZ(2.4GHz) or RSI_BAND_5GHZ(5GHz) or RSI_DUAL_BAND
#define RSI_CONFIG_AP_BAND RSI_BAND_2P4GHZ
//! To configure AP channel number
#define RSI_CONFIG_AP_CHANNEL 6
//! To configure security type
#define RSI_CONFIG_AP_SECURITY_TYPE RSI_WPA
//! To configure encryption type
#define RSI_CONFIG_AP_ENCRYPTION_TYPE 1
//! To configure PSK
#define RSI_CONFIG_AP_PSK "1234567890"
//! To configure beacon interval
#define RSI_CONFIG_AP_BEACON_INTERVAL 100
//! To configure DTIM period
#define RSI_CONFIG_AP_DTIM 2
//! This parameter is used to configure keep alive type
#define RSI_CONFIG_AP_KEEP_ALIVE_TYPE 0
//! RSI_NULL_BASED_KEEP_ALIVE
#define RSI_CONFIG_AP_KEEP_ALIVE_COUNTER 0 //!< 100
//! This parameter is used to configure keep alive period
#define RSI_CONFIG_AP_KEEP_ALIVE_PERIOD 100
//! This parameter is used to configure maximum stations supported
#define RSI_CONFIG_AP_MAX_STATIONS_COUNT 4
//! TCP_STACK_USED BIT(0) - IPv4, BIT(1) -IPv6, (BIT(0) | BIT(1)) -Both
IPv4 and IPv6
#define RSI_CONFIG_AP_TCP_STACK_USED BIT(0)
//! IP address of the module
//! E.g: 0x0AA0AA8C0 == 192.168.10.10
#define RSI_CONFIG_AP_IP_ADDRESS 0x0AA0AA8C0
//! IP address of netmask
```

```
//! E.g: 0x00FFFFFF == 255.255.255.0
#define RSI_CONFIG_AP_SN_MASK_ADDRESS          0x00FFFFFF
//! IP address of Gateway
//! E.g: 0x0A0AA8C0 == 192.168.10.10
#define RSI_CONFIG_AP_GATEWAY_ADDRESS          0x0A0AA8C0
```

4. If user wants to store different parameters in AP/client/eap client/p2p mode profile, please update the above configuration with required parameters.

2.3 Executing the Application

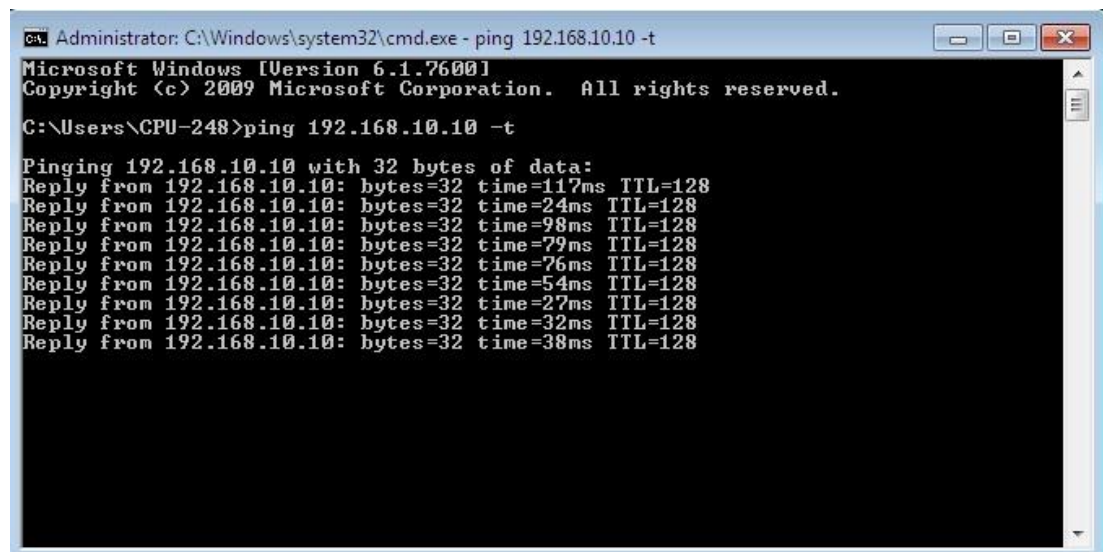
1. SPI Interface

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

3. After program gets executed, application fills the configuration structure with the given configuration and adds the configuration to non-volatile memory and enables the auto configuration.
4. After enabling auto configuration, application resets the WiSeConnect device. After successful reset, WiSeConnect device bootup into Access Point mode with the stored configuration (Default saved SSID is "REDPINE_AP").
5. Now scan from station and connect to AP and get IP through DHCP.
6. After successful connection, Initiate Ping to Access point IP address (default IP address stored is "192.168.10.10" from connected station).
7. WiSeConnect device will respond with ping response for the received Ping Request.



```
Administrator: C:\Windows\system32\cmd.exe - ping 192.168.10.10 -t
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\CPU-248>ping 192.168.10.10 -t

Pinging 192.168.10.10 with 32 bytes of data:
Reply from 192.168.10.10: bytes=32 time=117ms TTL=128
Reply from 192.168.10.10: bytes=32 time=24ms TTL=128
Reply from 192.168.10.10: bytes=32 time=98ms TTL=128
Reply from 192.168.10.10: bytes=32 time=79ms TTL=128
Reply from 192.168.10.10: bytes=32 time=76ms TTL=128
Reply from 192.168.10.10: bytes=32 time=54ms TTL=128
Reply from 192.168.10.10: bytes=32 time=27ms TTL=128
Reply from 192.168.10.10: bytes=32 time=32ms TTL=128
Reply from 192.168.10.10: bytes=32 time=38ms TTL=128
```