

Power Save Profile 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 in deep sleep power save profile mode before association.

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	Application Overview	4
1.1.1	Overview	4
1.1.2	Sequence of Events	4
1.2	Application Setup	4
1.2.1	SPI based Setup Requirements	4
1.2.2	UART/USB-CDC based Setup Requirements	4
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	10

Table of Figures

Figure 1:	Setup Diagram	5
Figure 2:	Deep Sleep power save profile	12
Figure 3:	Deep sleep and wakeup power save profile	13

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

This is a sample application demonstrating how to enable power save deep sleep profile with WiSeConnectTM module. This application enable power mode 8 and then wait in a scheduler for some time. Once it will come out of delay, it will connect to configured AP and then open udp client socket. It then sends some packet to the udp server and then disconnect from AP and goes back to deep sleep.

1.1.2 Sequence of Events

This Application explains user how to:

- Create device as a station
- Enable power mode 8 and then wait in a scheduler for some time.
- Once it will come out of delay, connect to configured AP
- Open udp client socket.
- Sends some packet to the udp server and then disconnect from AP and goes back to deep sleep.

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
- WiFi Access point
- Windows PC2 with UDP server application (iperf)
- Agilent power analyzer

1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- WiFi Access point
- Windows PC2 with UDP server application (iperf)
- Agilent power analyzer

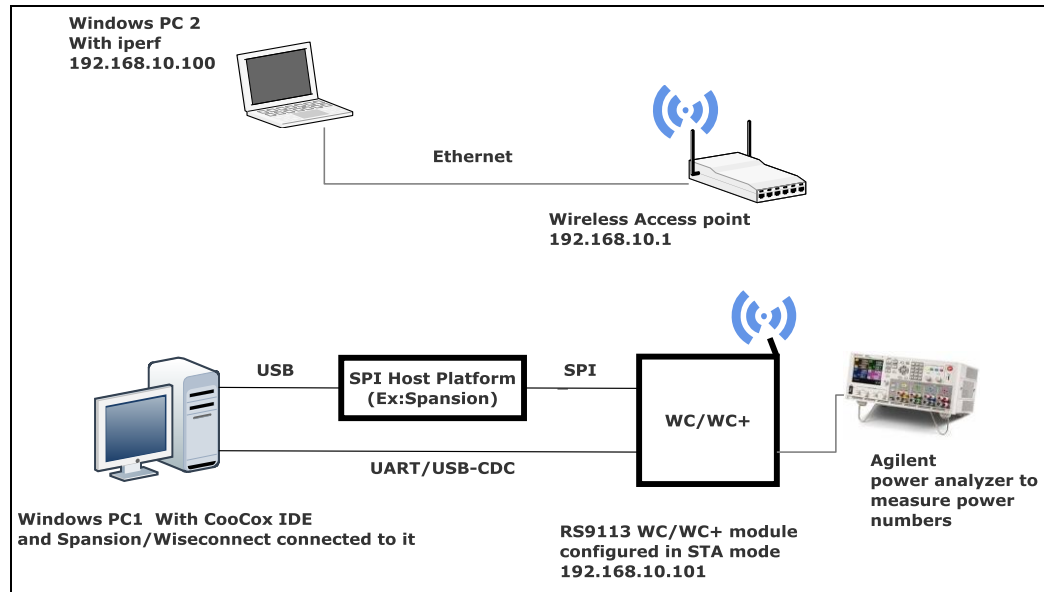


Figure 1: Setup Diagram

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 *power_save* example in CoCoX 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 *power_save* example in Dev-C++ IDE

2.2 Configuring the Application

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

SSID refers to the name of the Access point.

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

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>"
```

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

```
#define SERVER_PORT          5001
```

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 send from device to remote UDP server.

```
#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
```

In this application, default power save mode configuration is set to low power mode 8 (**RSI_SLEEP_MODE_8**) with maximum power save (**RSI_MAX_PSP**) with message based hand shake.

```
#define PSP_MODE    RSI_SLEEP_MODE_8
```

```
#define PSP_TYPE    RSI_MAX_PSP
```

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
```

Default configuration of low power save mode 2

```
#define RSI_HAND_SHAKE_TYPE MSG_BASED
#define RSI_SELECT_LP_OR_ULP_MODE RSI_LP_MODE
#define RSI_DTIM_ALIGNED_TYPE 0
#define RSI_MONITOR_INTERVAL 50
#define RSI_WMM_PS_ENABLE RSI_DISABLE
#define RSI_WMM_PS_TYPE 0
#define RSI_WMM_PS_WAKE_INTERVAL 20
#define RSI_WMM_PS_UAPSD_BITMAP 15
```

3. If user wants to select different power save mode profiles, please go through the step #4 and #5 other wise skip step #4 and #5
4. Open *sapis/examples/wlan/power_save/rsi_wlan_power_save_profile.c* file and update/modify following macros,

PSP_MODE refers power save profile mode. WiSeConnect device supports following power modes :

RSI_ACTIVE (0) : In this mode, module is active and power save is disabled.

RSI_SLEEP_MODE_1 (1) : In this power mode, module goes to power save after association with the Access Point. In this sleep mode, SoC will never turn off, therefore no handshake is required before sending data to the module.

RSI_SLEEP_MODE_2 (1) : In this power mode, module goes to power save after association with the Access Point. In this sleep mode, SoC will go to sleep based on GPIO hand shake or Message exchange, therefore handshake is required before sending data to the module.

RSI_SLEEP_MODE_8 (8) : In this power mode, module goes to power save when it is in unassociated state with the Access Point. In this sleep mode, SoC will go to sleep based on GPIO hand shake or Message exchange, therefore handshake is required before sending the command to the module.

```
#define PSP_MODE RSI_SLEEP_MODE_8
```

Note1: For **RSI_SLEEP_MODE_2** and **RSI_SLEEP_MODE_8** modes, GPIO or Message based hand shake can be selected using **RSI_HAND_SHAKE_TYPE** macro which is define in *sapis/include/rsi_wlan_config.h*

Note2: In this example user can verify **RSI_SLEEP_MODE_2** with Message based hand shake. If user wants to verify other power modes, user has to change the application as well as GPIO hand shake signals.

PSP_TYPE refers power save profile type. WiSeConnect device supports following power save profile types :

RSI_MAX_PSP (0) : In this mode, WiSeConnect device will be in Maximum power save mode. i.e Device will wake up for every DTIM beacon and do data Tx and Rx.

RSI_FAST_PSP (1) : In this mode, WiSeConnect device will disable power save for any Tx/Rx packet for monitor interval of time (monitor interval can be set through macro in **sapis/include/rsi_wlan_config.h** file, default value is 50 ms). If there is no data for monitor interval of time then module will again enable power save.

RSI_UAPSD (2) : This **PSP_TYPE** is used to enable WMM power save.

```
#define PSP_TYPE RSI_MAX_PSP
```

Note1:

PSP_TYPE is valid only when **PSP_MODE** set to **RSI_SLEEP_MODE_1** or **RSI_SLEEP_MODE_2** mode.

Note2:

RSI_UAPSD power profile type in **PSP_TYPE** is valid only when **RSI_WMM_PS_ENABLE** is enabled in **sapis/include/rsi_wlan_config.h** file.

- Open **sapis/include/rsi_wlan_config.h** file and update/modify following macros,
RSI_HAND_SHAKE_TYPE is used to select **GPIO** or Message based hand shake in **RSI_SLEEP_MODE_2** and **RSI_SLEEP_MODE_8** modes.

```
#define RSI_HAND_SHAKE_TYPE MSG_BASED
```

RSI_SELECT_LP_OR_ULP_MODE is used to select low power mode or ultra low power mode. Valid configurations are, **RSI_LP_MODE** or **RSI_ULP_WITH_RAM_RET** or **RSI_ULP_WITHOUT_RAM_RET**

RSI_LP_MODE: In this module will be in Low power mode.

RSI_ULP_WITH_RAM_RET: In this module will be in Ultra low power mode and it will remember the previous state after issuing power save mode command.

RSI_ULP_WITHOUT_RAM_RET: In this module will be in Ultra low power mode and it will not remember the previous state after issuing power save mode command. After wakeup, module will give CARD READY indication and user has to issue commands from wireless initialization.

```
#define RSI_SELECT_LP_OR_ULP_MODE RSI_LP_MODE
```

RSI_DTIM_ALIGNED_TYPE refers whether module has to wake up at normal beacon or DTIM beacon which is just before listen interval.

If **RSI_DTIM_ALIGNED_TYPE** is set to 0(Zero) i.e module will wake up at normal beacon which is just before listen interval

If **RSI_DTIM_ALIGNED_TYPE** is set to 1(Zero) i.e module will wake up at DTIM beacon which is just before listen interval

```
#define RSI_DTIM_ALIGNED_TYPE 0
```

RSI_MONITOR_INTERVAL refers amount of time (in ms) to wait for Tx or Rx before giving power save indication to connected Access Point.

```
#define RSI_MONITOR_INTERVAL 50
```

Note:

RSI_MONITOR_INTERVAL is applicable only when **PSP_TYPE** selected as **RSI_FAST_PSP**

RSI_WMM_PS_ENABLE is used to enable or disable WMM power save.

```
#define RSI_WMM_PS_ENABLE 0
```

RSI_WMM_PS_TYPE is used to set Tx based or Periodic based WMM power save.

Update **RSI_WMM_PS_TYPE** macro with 0 for Tx Based or 1 for periodic based WMM power save.

```
#define RSI_WMM_PS_TYPE 0
```

RSI_WMM_PS_WAKE_INTERVAL refers at periodic time (in ms) module has to wake up module when **RSI_WMM_PS_TYPE** selected as Periodic.

```
#define RSI_WMM_PS_WAKE_INTERVAL 20
```

RSI_WMM_PS_UAPSD_BITMAP refers UAPSD bitmap

```
#define RSI_WMM_PS_UAPSD_BITMAP 15
```

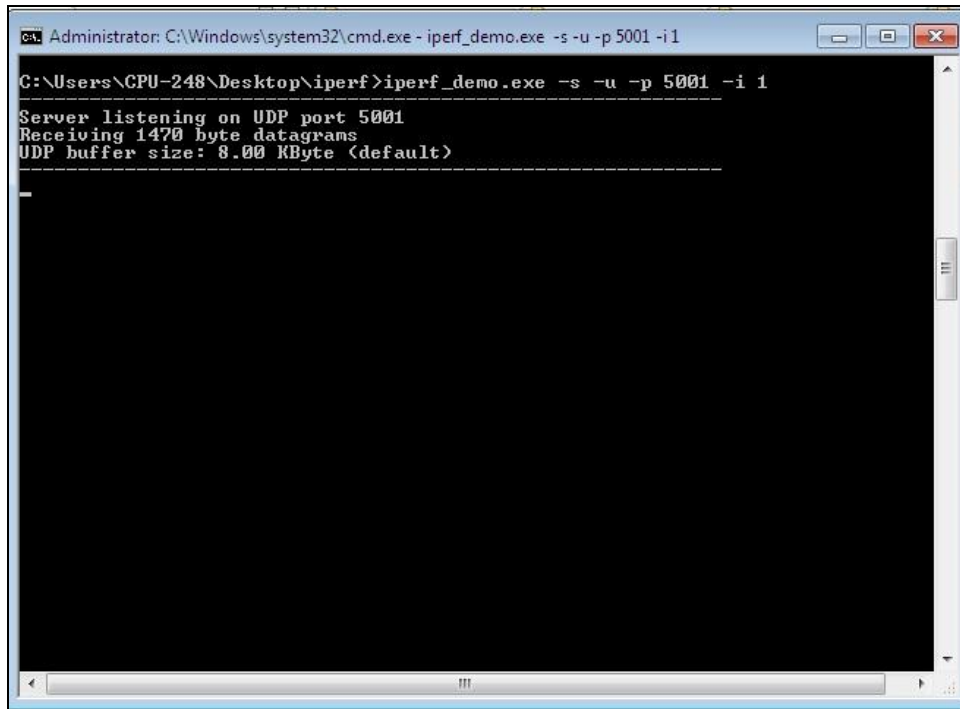
Note:

If **RSI_WMM_PS_ENABLE** is enabled then user has to set **PSP_TYPE** to **RSI_UAPSD** in order to work WMM power save.

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. Open UDP server application using iperf application in Windows PC2 which is connected to Access point through LAN.

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



```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -u -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -u -p 5001 -i 1
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
```

3. 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 **power_save** example in CooCox IDE.

4. 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 **power_save** example in Dev-C++ IDE

5. After program gets executed, WiSeConnect Device will go to sleep based on the selected power mode and wakes up after deep sleep timeout (Default deep sleep time is 3sec in RSI_SLEEP_MODE_8 with message based hand shake). Please refer the given below image for power save cycle for default deep sleep time.

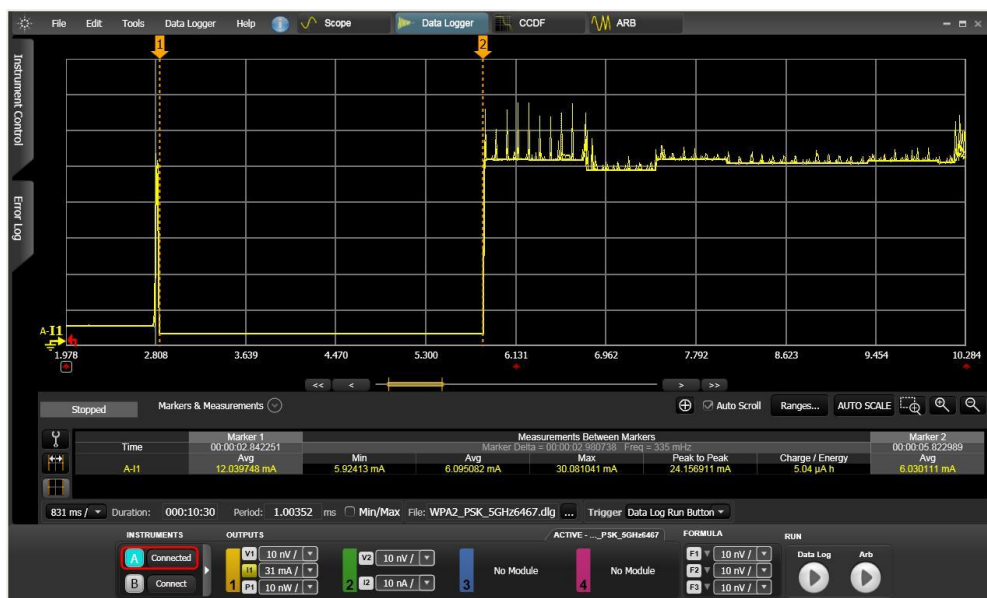
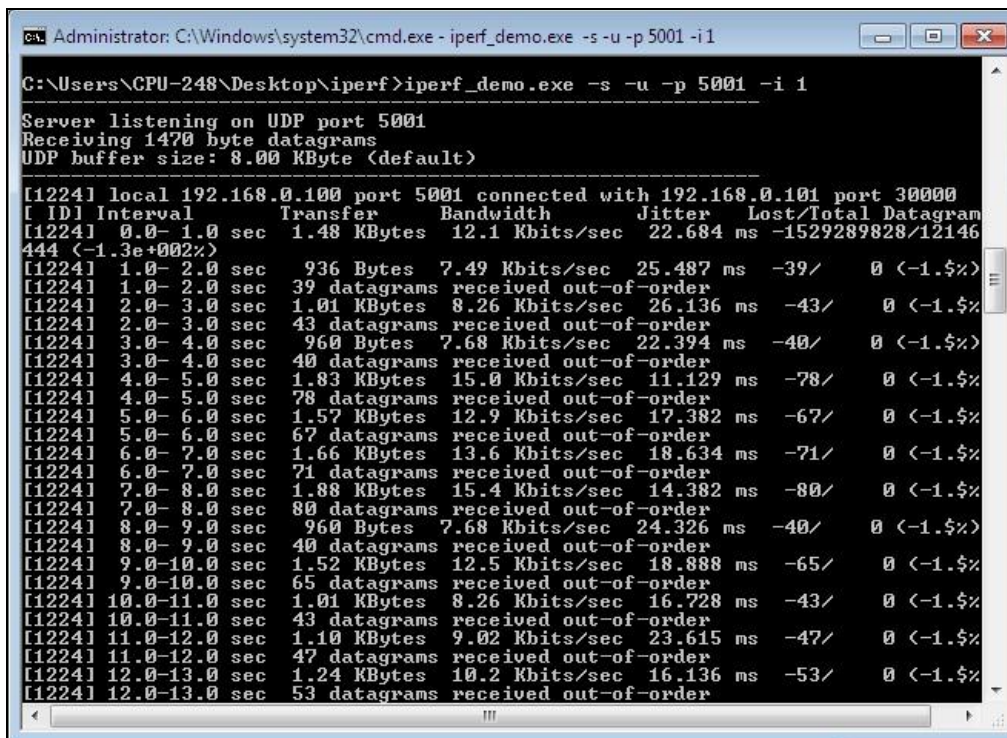


Figure 2: Deep Sleep power save profile

- After successful wake up from deep sleep, WiSeConenct device connects to AP and sends configured number of (**NUMBER_OF_PACKETS**) UDP packets to remote peer which is connected to Access point through LAN. Please refer the given below image for reception of UDP data on UDP server.



7. After sending configured number of packets, WiSeConnect device disconnects from connected AP and again repeat the steps from #10 to #11 (Again it will go to sleep and wakes up after time out and connects to AP and sends configured number of packets). Please find below image for power save profile cycle.



Figure 3: Deep sleep and wakeup power save profile