Co-Ex Power Save Profile Application

User guide
Version 0.1

September 2015

Redpine Signals, Inc.

2107 N. First Street, #680

San Jose, CA 95131.

Tel: (408) 748-3385

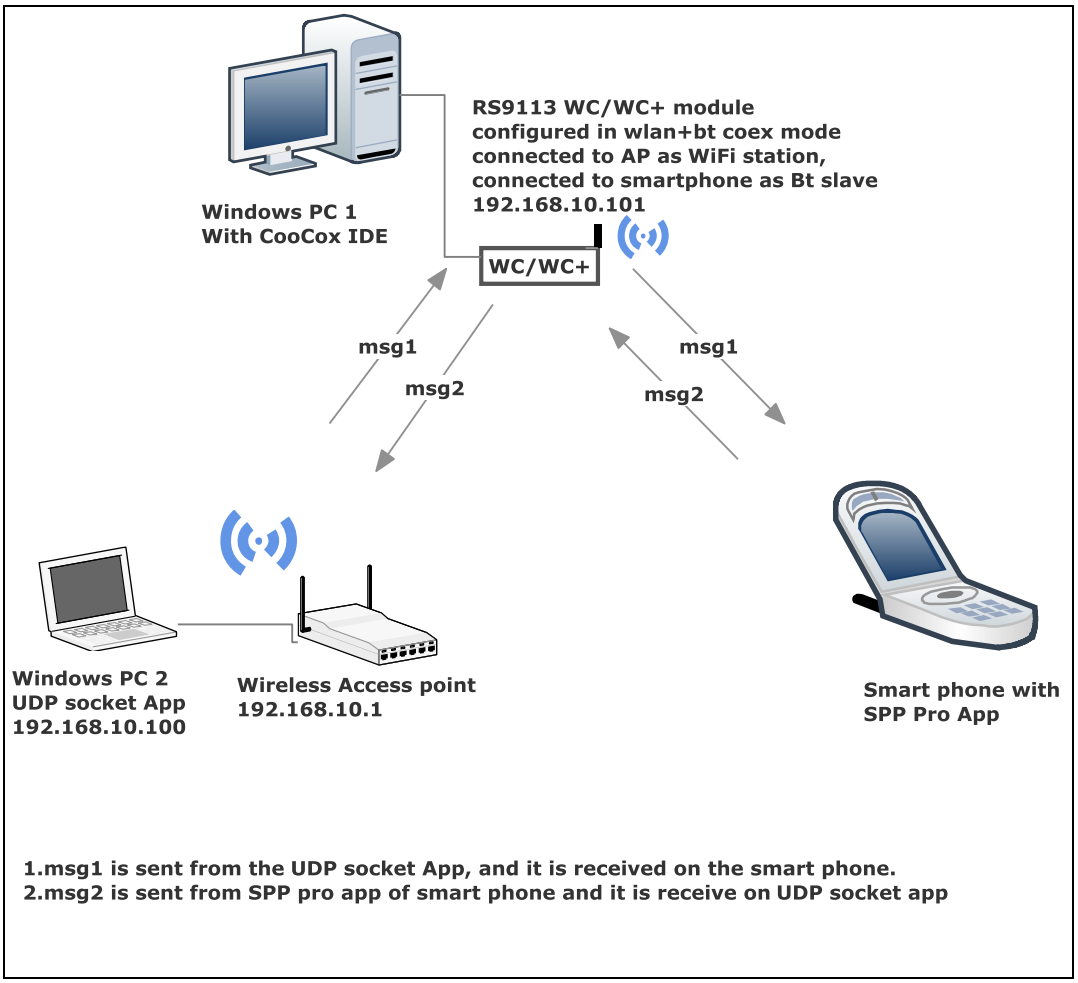
This project is applicable to all the WiSeConnect variants like WiSeConnect Plus, WiSeMCU and WYZBEE. The term WiSeConnect refers to its appropriate variant.

Application Overview:

The coex application demonstrates how enable power save connected sleep profile power mode 2 for BT and WLAN protocols, and the information can be exchanged seamlessly using two wireless protocols (WLAN and BT) running in the same device.

Setup required:

1. Windows PC with Coocox IDE
2. WiSeConnect device
3. Smart phone/tablet with BT Application (Ex: Bluetooth SPP Pro)
4. WLAN Access Point and a Windows PC with UDP socket application



Description:

The coex application has WLAN and BT tasks and acts as an interface between Smartphone and PC.

Smartphone interacts with BT task, while PC[1] interacts with WLAN task

When Smartphone connects and sends message to WiSeConnect, BT task accepts and sends to WLAN task, which in turn sends to Access Point connected PC.

Similarly, when PC sends message to WiSeConnect device, the message will be sent to Smartphone via BT task.

Thus messages can be seamlessly transferred between Windows PC and Smartphone.

Details of the Application:

WiSeConnect WLAN acts as a Station and connects to an Access Point

WiSeConnect BT acts as a Slave device with Bluetooth SPP Pro running in it, while Smart phone acts as a Master device with Bluetooth SPP Pro running in it.

Initially, proprietary Simple chat service is created with SPP profile (WiSeConnect device) to facilitate message exchanges.

- The WLAN task (running in WiSeConnect device) mainly includes following steps.
    1. Connects to a Access Point
    2. Exchanges data over UDP socket with the peer(windows PC)
    3. Initiating power save after connection establishment.
- The BT task (running in WiSeConnect device) mainly includes following steps.
    1. Creates chat service
    2. Configures the device to Discoverable and Connectable modes.
    3. Initiating power save after connection establishment.

WLAN and BT tasks forever run in the application to serve the asynchronous events

Configuring the WLAN task:

[1] Both PC and WiSeConnect WLAN would be connected to a Wireless Access Point, thus both are connected together wirelessly

Edit the **rsi_wlan_app.c** file in the following path to establish connection with the Access-point.

sapis/examples/wlan_bt/power_save/

1. From given configuration,

   SSID refers to the Access point to which user wants to connect.

   SECURITY_TYPE is the security type of the Access point.

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

```
#define SSID                "<ap_name>"
#define SECURITY_TYPE        <security-type>
#define PSK                   ""
```

2. To configure static IP address

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

   Example: To configure "192.168.10.101" as IP address, update the macro **DEVICE_IP** as **0x650AA8C0**.

```
#define  DEVICE_IP          0x650AA8C0
```

   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
```

3. To establish UDP connection and transfer data to the remote socket configure the below macros. Internal socket port number.

```
#define DEVICE_PORT 5001
```

   Port number of the remote server

```
#define SERVER_PORT 5001
```

   IP address of the remote server

```
#define SERVER_IP_ADDRESS 0x020AA8C0
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

4. Open UDP socket on remote peer (PC).

   User can use Test UDP application from below link to create UDP socket on remote peer

   http://sourceforge.net/projects/sockettest/files/latest/download

5. Include rsi_wlan_app.c , rsi_bt_app.c and main.c files in the project, build and launch the application

6. To enable power save following macro need to change Power Save Procedure mode (psp_mode), Follwing psp_mode is defined.

   - RSI_ACTIVE (0): In this mode module is active and power save is disabled
   - RSI_SLEEP_MODE_1 (1): This is connected sleep mode. In this sleep mode, SoC will never turn off, therefore no handshake is required before sending data to the module.
   - RSI_SLEEP_MODE_2 (2): This is connected sleep mode. In this sleep mode, SoC will go to sleep based on GPIO or Message, therefore handshake is required before sending data to the module.
   - RSI_SLEEP_MODE_8 (8): This is disconnected sleep mode. In this sleep mode, module will turn off the SoC with other component. Since SoC is turn off, therefore handshake is required before sending data to the module.

   ```
   #define PSP_MODE                RSI_SLEEP_MODE_2
   ```

7. Power Save Procedure type (psp_type), Follwing psp_type is defined.
   - RSI_MAX_PSP (0) : This psp_type will be used for max power saving.
   - RSI_FAST_PSP (1) : This psp_type allows module to disable power save for any Tx/Rx packet for monitor interval of time (monitor interval can be set through configuration 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
   ```

NOTE:
1. `PSP_TYPE` is only valid in `RSI_SLEEP_MODE_1` and `RSI_SLEEP_MODE_2`.
2. For `psp_type` RSI_UAPSD is valid only if WMM_PS is enabling in `rsi_wlan_config.h` file.

Update the Wlan configuration file:

sapis/include/rsi_wlan_config.h

CONCURRENT_MODE	DISABLE
RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
RSI_TCP_IP_BYPASS	ENABLE
RSI_TCP_IP_FEATURE_BIT_MAP	TCP_IP_FEAT_BYPASS
RSI_CUSTOM_FEATURE_BIT_MAP	0
RSI_BAND	RSI_BAND_2P4GHZ

Configuring the BT Application:

Edit the **rsi_bt_app.c** file in the following path of the Application

sapis/examples/wlan_bt/power_save/

Configure the below macros in the Application file.

1. RSI_BT_LOCAL_NAME **-** Name of the Wyzbee (Master) device
2. PIN_CODE - Four byte string required for pairing process.
3. PSP_TYPE – Power save profile type.

NOTE:
1. PSP_TYPE is only valid RSI_SLEEP_MODE_2.
2. RSI_MAX_PSP is only valid in case of BT.

4. SNIFF_MAX_INTERVAL - Sniff Maximum interval value
5. SNIFF_MIN_INTERVAL – Sniff Minimum interval value
6. SNIFF_ATTEMPT - Sniff Attempt Value
7. SNIFF_TIME_OUT – Sniff Timeout Value

Following are the **non-configurable** Macros in the Application file.

8. BT_GLOBAL_BUFF_LEN – Number of bytes required for the Application and the Driver.
9. RSI_APP_EVENT_CONNECTED - Event number to be set on connection establishment.
10. RSI_APP_EVENT_DISCONNECTED - Event number to be set on disconnection.

11. RSI_APP_EVENT_PINCODE_REQ - Event number to be set on Pincode request for pairing.
12. RSI_APP_EVENT_LINKKEY_SAVE - Event number to be set on link key save.
13. RSI_APP_EVENT_AUTH_COMPLT - Event number to be set on authentication complete.
14. RSI_APP_EVENT_LINKKEY_REQ - Event number to be set on link key request for connection.
15. RSI_APP_EVENT_SPP_CONN - Event number to be set on SPP connection.
16. RSI_APP_EVENT_SPP_DISCONN - Event number to be set on SPP disconnection.
17. RSI_APP_EVENT_SPP_RX - Event number to be set on SPP data received from Master

Executing the coex Application:

1. Connect WiSeConnect device to the Windows PC running Cocoox IDE.
2. Configure the macros in the files located at

   sapis/examples/wlan_bt/power_save/rsi_bt_app.c

   sapis/examples/wlan_bt/power_save/rsi_wlan_app.c

3. Build and launch the application.
4. After the program gets executed, WiSeConnect BT is in Discoverable state and WLAN has established connection and opened a UDP socket
5. After connection establishment in WLAN it provides a power save command.
6. Open a BT SPP Pro App in the Smartphone and do scan.
7. In the App, WiSeConnect BT would appear with the name configured in the macro RSI **RSI_BT_LOCAL_NAME**.
8. Now initiate connection from the SPP App running in the Smartphone.
9. After connection establishment in BT it provides a power save command.
10. After BT connection is established, send a message from the App to WiSeConnect BT. Observe this message in the PC connected via UDP socket with WiSeConnect WLAN.
11. Now, send a message from PC to WiSeConnect WLAN via UDP socket and observe the same in the Smartphone.

12. rsi_bt_app_send_to_wlan() function defined in rsi_wlan_app.c to send message from BT task to WLAN task.
13. With the help of wlan task, message is transferred to PC.
14. Message from PC to WLAN application via socket and rsi_wlan_app_send_to_bt() function defined in rsi_bt_app.c called asynchronously to send message from WLAN task to BT task. From BT task message transferred to client with the event.