RT58x BLE Mesh SDK Application Guide V1.2

About this Document

This document supports at least "***Rafael RT58x SDK v1.2.3***".

Table of Contents

1. Introduction

The RT58x is built around the 32-bit ARM® Cortex™-M3 CPU with clock running up to 64 MHz. And it is capable of running multi-protocol concurrently. It can support protocols including Bluetooth LE, Bluetooth mesh, ZigBee, 802.15.4, and 2.4GHz proprietary.

This document is mainly about the instructions related to the BLE Mesh example demonstration. The content includes server roles (Lightness_TRSP, one element with lightness & vendor model).

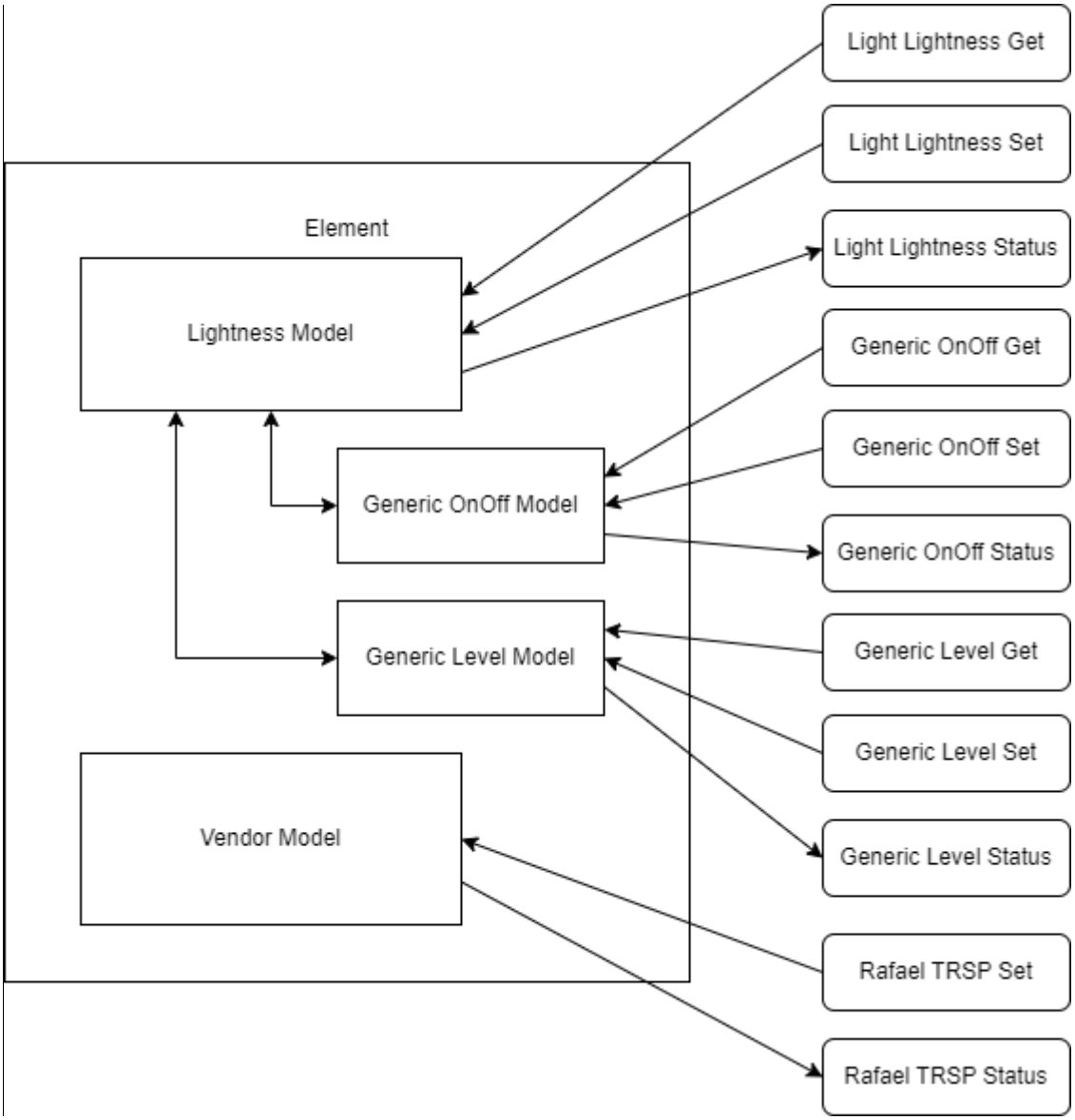
(intentionally blank)

2. BLE Mesh Server Demo Application

There are one BLE mesh server demo projects for reference, which are Lightness_TRSP.

2.1 Lightness_TRSP

The Lightness_TRSP demonstrates one element that contains the part of lightness mode and vendor model, which means device can be turn on/off via generic on off mode, level control by generic level model and using vendor model to transmit/receive the user data from/to UART.



2.1.1 Lightness Model

The lightness server model in our demo consists of two generic model, generic OnOff server and generic level server.

2.1.2 Vendor Model

Because our company ID is 0x0864, so that we define the model id of private model "**Rafael TRSP server**" as 0x08640831, "**Rafael TRSP client**" as 0x08640832, and we also defined three kind of messages:

- **Rafael TRSP Status message (opcode 0x0864C0)**
- **Rafael TRSP Set message (opcode 0x0864C1)**
  When Rafael TRSP Set message received, the message payload will loopback to sender by message "Rafael TRSP Status" automatically.
- **Rafael TRSP Set unacknowledged message (opcode 0x0864C2)**

The maximum size can be transmitting by vendor model in one time is 377 bytes and there was a simple formula to calculate the maximum transmission time:

*(200ms + 50ms * **segmented number** * **TTL)** * **retransmission count***.

Segmented number: Depends on vendor model message size. For maximum message size 377 bytes the segmented number is 32
TTL: The number of hops, we define 4 for our demonstration and user can change the value by different situation.
Retransmission count: means this segment packet will be retransmitted how many times. For our demonstration retransmission count is 5.

For example:

1 segment packet (4 hops and 5 retransmission) needs ((200+50*1*4)*5) 2000ms
32 segment packet (4 hops and 5 retransmission) needs ((200+50*32*4)*5) 33000ms

2.1.3 Handle Lightness Data Received Over BLE Mesh

Implement the callback function "app_process_element_lightness_model_state()" and we can get element index by first input parameter "**element_address**" and according to second input parameter "**state**" to adjust actual level of different element.

```
void app_process_element_lightness_model_state(uint16_t element_address, uint16_t state)
{
    uint8_t element_idx;

    element_idx = element_address - pib_primary_address_get();
    info_color(LOG_GREEN, "Set element[%d] act level %d\n", element_idx, state);

    if (state > 0)
    {
        set_duty_cycle(&pwm_para_config[0], state);
        set_duty_cycle(&pwm_para_config[1], state);
        set_duty_cycle(&pwm_para_config[2], state);
    }
    else
    {
        set_duty_cycle(&pwm_para_config[0], 0);
        set_duty_cycle(&pwm_para_config[1], 0);
        set_duty_cycle(&pwm_para_config[2], 0);
    }
}
```

This callback function is registered in file “ble_mesh_element_def.c”.

```
ble_mesh_model_param_t      el0_light_lightness_model =
{
    MMDL_LIGHT_LIGHTNESS_SR_MDL_ID,                     //model ID
    0,                                                  //rcv_tid
    FALSE,                                              //is extended
    &el0_light_lightness_state,                         //state
    &el0_light_lightness_model_publish_entry,           //publish ptr
    el0_light_lightness_subscribe_list,                 //subscript list ptr
    (void *)NULL,                                       //upper callback
    (void *)app_process_element_lightness_model_state       //call back
};


……

/* Declare the model using in element */
ble_mesh_model_param_t *el0_light_lightness_element_model_list[] =
{
    &el0_light_lightness_model,
    &el0_light_lightness_setup_model,
    &el0_gen_level_model,
    &el0_gen_on_off_model,
    &el0_scene_model,
    &el0_scene_setup_model,
    &el0_raf_trsp_sr_model,
    &el0_raf_trsp_cl_model,
};

/*****************************************************************************************************************/

ble_mesh_element_param_t g_element_info[] =
{
    // element 0, lightness element
    {
        el0_light_lightness_element_model_list,         /*model list*/
        5,                                              /*model count*/
        0xFFFF,                                         /*element address*/
        0,                                              /*tx transcation id*/
    },
};
```

2.1.4 Handle Vendor Data Received Over BLE Mesh

Implement the callback function “app_process_element_raf_trsp_sr_model_state ()” & “app_process_element_raf_trsp_cl_model_state ()” and we can get following input parameters

in the parameter “p_raf_trsp_cb_params”:

- **src_addr**: The source address of receiving TRSP data.
- **dst_addr**: The destination address of receiving TRSP data.
- **appkey_index**: The application key index of receiving TRSP data.
- **is_group**: Indicates this vendor data send over group or unicast.
- **data_len**: The data length of receiving vendor data.
- **data**: The data pointer of receiving vendor data.

```
void app_process_element_raf_trsp_sr_model_state (raf_trsp_cb_params_t *p_raf_trsp_cb_params)
{
    info_color(LOG_GREEN, "Get Rafael TRSP set from address 0x%04x\n", p_raf_trsp_cb_params->src_addr);

    for (uint16_t i = 0; i < p_raf_trsp_cb_params->data_len; i++)
    {
        info_color(LOG_GREEN, "%02x ", p_data[i]);
    }
    info_color(LOG_GREEN, "\n");

}

void app_process_element_raf_trsp_cl_model_state (raf_trsp_cb_params_t *p_raf_trsp_cb_params)
{
    info_color(LOG_GREEN, "Get Rafael TRSP status from address 0x%04x\n", p_raf_trsp_cb_params->src_addr);

    for (uint16_t i = 0; i < p_raf_trsp_cb_params->data_len; i++)
    {
        info_color(LOG_GREEN, "%02x ", p_data[i]);
    }
    info_color(LOG_GREEN, "\n");

}
```

This callback function is registered in file “ble_mesh_element_def.c”.

```
……

raf_trsp_state_t            el0_raf_trsp_state;
uint16_t                    el0_raf_trsp_subscribe_list[RAF_BLE_MESH_SUBSCRIPTION_LIST_SIZE];
ble_mesh_model_param_t      el0_raf_trsp_sr_model =
{
    MMDL_RAFAEL_TRSP_SR_MDL_ID,
    0,
    FALSE,
    &el0_raf_trsp_state,
    (void *)NULL,                                   //publish ptr
    el0_raf_trsp_subscribe_list,                    //subscript list ptr
    (void *)NULL,
    (void *)app_process_element_raf_trsp_sr_model_state
};

ble_mesh_model_param_t      el0_raf_trsp_cl_model =
{
    MMDL_RAFAEL_TRSP_CL_MDL_ID,
    0,
    FALSE,
    NULL,
    (void *)NULL,                                   //publish ptr
    NULL,                                           //subscript list ptr
    (void *)NULL,
    (void *)app_process_element_raf_trsp_cl_model_state
};

/* Declare the model using in element */
ble_mesh_model_param_t *el0_light_lightness_element_model_list[] =
{
    &el0_light_lightness_model,
    &el0_light_lightness_setup_model,
    &el0_gen_level_model,
    &el0_gen_on_off_model,
    &el0_scene_model,
    &el0_scene_setup_model,
    &el0_raf_trsp_sr_model,
    &el0_raf_trsp_cl_model,
};

/*****************************************************************************************************************/

ble_mesh_element_param_t g_element_info[] =
{
    // element 0, lightness element
    {
        el0_light_lightness_element_model_list,     /*model list*/
        5,                                          /*model count*/
        0xFFFF,                                     /*element address*/
        0,                                          /*tx transcation id*/
    },
};
```

2.1.5 Running “Lightness_TRSP” Demo

This demo code can be tested with two kind of situations:

1. Testing with the smart phone.
2. Testing with device.

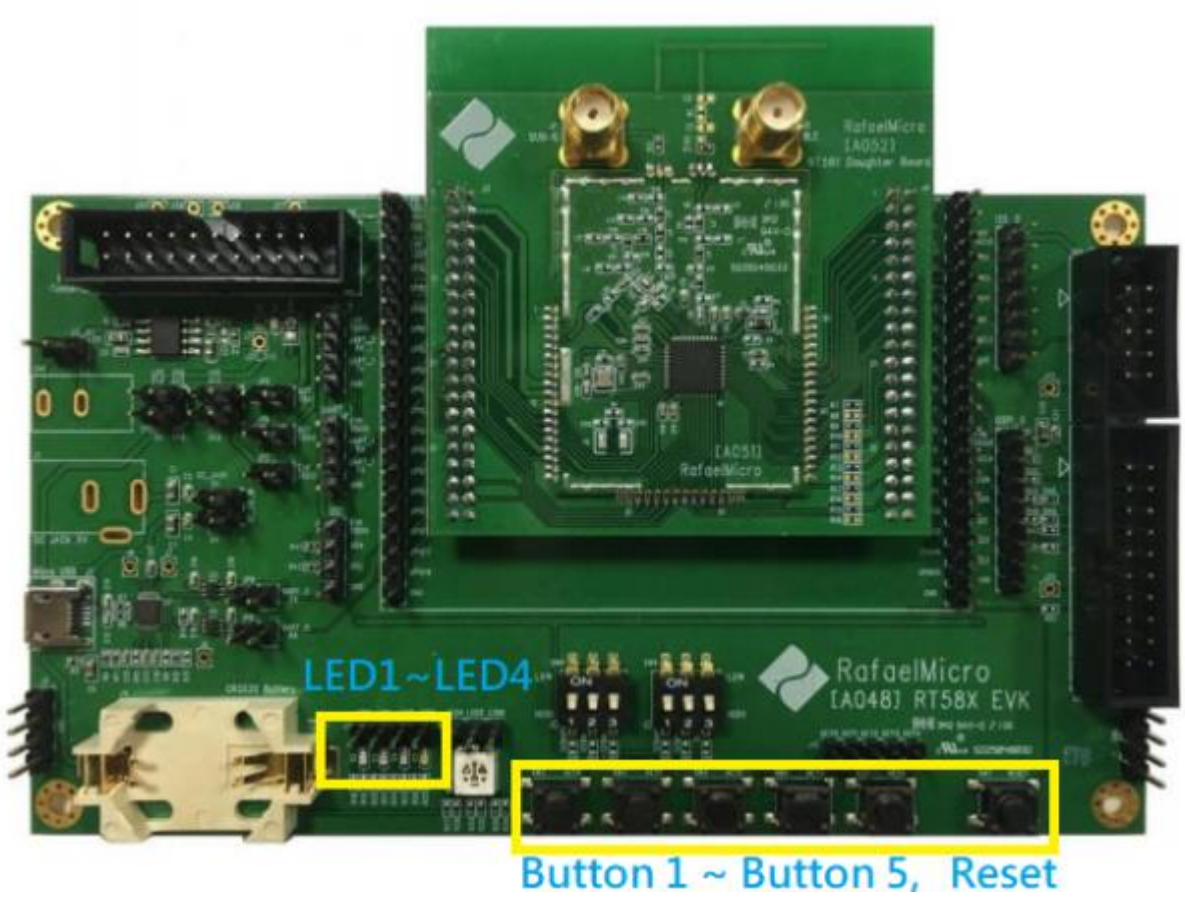
Before device start testing, complete the following steps to know device current status

- Connect EVK board to the USB port

- Start serial tool (Putty or Tera Term)
- Configure baud rate to 115200

[Device local reset to factory new]

RT58x EVK board:



Button behavior of lightness device:

- Button1: Reset device to factory new and start BLE mesh provision
- Button2: Reset device to factory new and start BLE GATT provision

[Testing with smart phone]

Reference document "[Tool_05]Android_APP_BLE_Mesh_user_guide_V1.3" section 5, this document shows how to provision a new device into mesh network. After device provisioned, we can control the LED of the device by lightness model and also using vendor model to send specific data to device.

[Testing with device]

After multiple devices provisioned by smart phone, the log printed device primary address which assigned by smart phone. We can use primary address to send UART data over vendor model

from single device to another device.

```
BLE Mesh Stack initial ...
BLE Mesh MMDL initial ...
timer_low_running_handler(0)
MAC address: 0x43 0x4e 0x3e 0x6f 0x30 0xc1
sysinfo: sys ver 0001
Device provisioned ... primary address 0x0004
```

Enter "*DstAddr0x0003*" to set destination address of UART data transfer as "0x0003"

```
COM113:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Help
sysinfo: sys ver 0001
Device provisioned ... primary address 0x0004

destination address invalid, uart data ignore
command example:
DstAddr0x001c
TxAckEnable0
DstAddr0x0003
destination address for uart data: 0x0003
```

After setting destination address, subsequent UART data will be sent to address 0x0003

```
TxAckEnable0
DstAddr0x0003
destination address for uart data: 0x0003
123456
Send Rafael TRSP set, dst addr 0x0003
31 32 33 34 35 36
... result 0
```

```
BLE Mesh Stack initial ...
BLE Mesh MMDL initial ...
timer_low_running_handler(0)
MAC address: 0xc1 0x66 0x77 0x88 0x99 0xc1
sysinfo: sys ver 0001
Device provisioned ... primary address 0x0003
Get Rafael TRSP set from address 0x0004
31 32 33 34 35 36
```

Enter "*TxAckEnable1*" to request destination device loopback the received data to sender

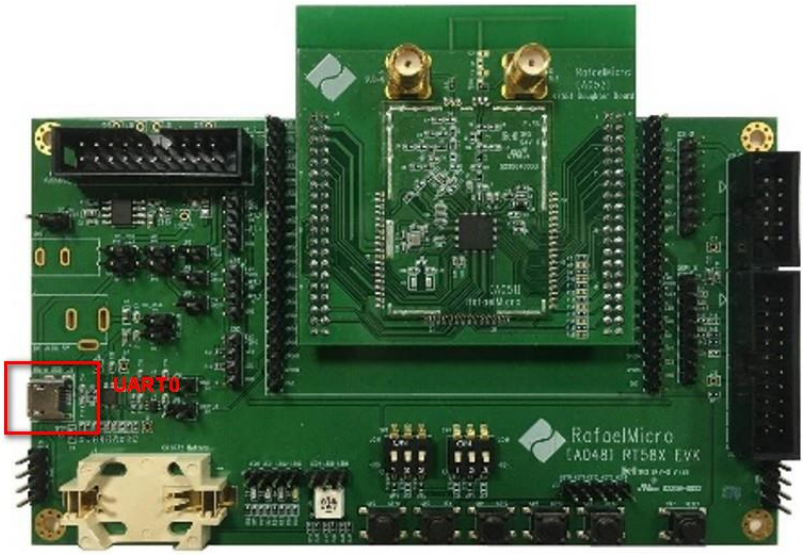
```
Send Rafael TRSP set, dst addr 0x0003
31 32 33 34 35 36
... result 0
TxAckEnable1
uart data tx with ack 1
12345
Send Rafael TRSP set, dst addr 0x0003
31 32 33 34 35
... result 0
Get Rafael TRSP status from address 0x0003
31 32 33 34 35
```

```
BLE Mesh Stack initial ...
BLE Mesh MMDL initial ...
timer_low_running_handler(0)
MAC address: 0xc1 0x66 0x77 0x88 0x99 0xc1
sysinfo: sys ver 0001
Device provisioned ... primary address 0x0003
Get Rafael TRSP set from address 0x0004
31 32 33 34 35 36
Get Rafael TRSP set from address 0x0004
31 32 33 34 35
```

2.2 Gateway

The Gateway demonstration project provide two kind of command, basic CLI commands from debug console (UART0) to taste the BLE mesh network. And the UART command set (Hex format) from UART1 for a host MCU to use this command set to manage BLE mesh network. Document [SW_21]RT58x_BLE_Mesh_Gateway_Command_Manual_V0.1.pdf list all the detail control & response command.

2.2.1 Running the demo with CLI Commands



complete the following steps to use CLI command:

- Connect the RT58x EVK boards to the USB ports.
- Start the serial tools (Putty, Tera Term)
- Config the baud rate to 115200, none, 1.

The following table shows all the supported CLI command:

Command	Usage	example	Description
help	help	help	List commands
provisionall	provisionall [start]	provisionall 1	Start/stop provision all device
compget	compget [address] [page num]	compget 0x0123 0	Get device's composition data
appkeyadd	appkeyadd [address]	appkeyadd 0x0123	add app key to device with index 0
modelappbind	modelappbind [address] [model id]	modelappbind 0x0123 0x1000	bind app key index 0 to specific model
onoff	onoff [address] [onoffstatus]	onoff 0x0123 0	set device onoff status
level	level [address] [levelstatus]	level 0x0123 0x8000	set device level status
nodereset	nodereset [address]	nodereset 0x0123	reset device to factory new

Note that when using CLI command to provision device, the maximum provision device number is 50. This limitation caused by the device key of provisioned device were store into “prov_device_list” and the size of “prov_device_list” is 50 (MAX_PROV_DEVICE_NUM). Moreover, the “prov_device_list” was not store into flash, so that after gateway reset, the “prov_device_list” will also reset to empty so the provisioned device would not configure again.

[Provision new device]

When Lightness_TRSP device is reset to factory new and start mesh provisioning, we can use the command "provisionall" to start scanning for unprovisioned devices and provision all devices in sequence.

```
~# RT58x SDK for BLE Mesh ...
BLE Mesh Client Stack initial ...
timer_low_running_handler(0)
MAC address: 0xce 0x00 0xfc 0xf3 0x44 0xa0
sysinfo: sys ver 0001
~# provisionall 1
auto provision device start
~# start provision device: 0x030903030308030006f030e040e0403
Send link open to 03 09 03 03 03 08 03 00 06 0f 03 0e 04 0e 04 03
Send link open to 03 09 03 03 03 08 03 00 06 0f 03 0e 04 0e 04 03
Send Provisioning Invite PDU.
Send Provisioning PDU
Get MESH_PRV_PDU_CAPABILITIES
Send Start PDU.
```

[Configure device]

Use the command "appkeyadd" & “modelappbind” to configure device’s App key and bind this App key to specific model.

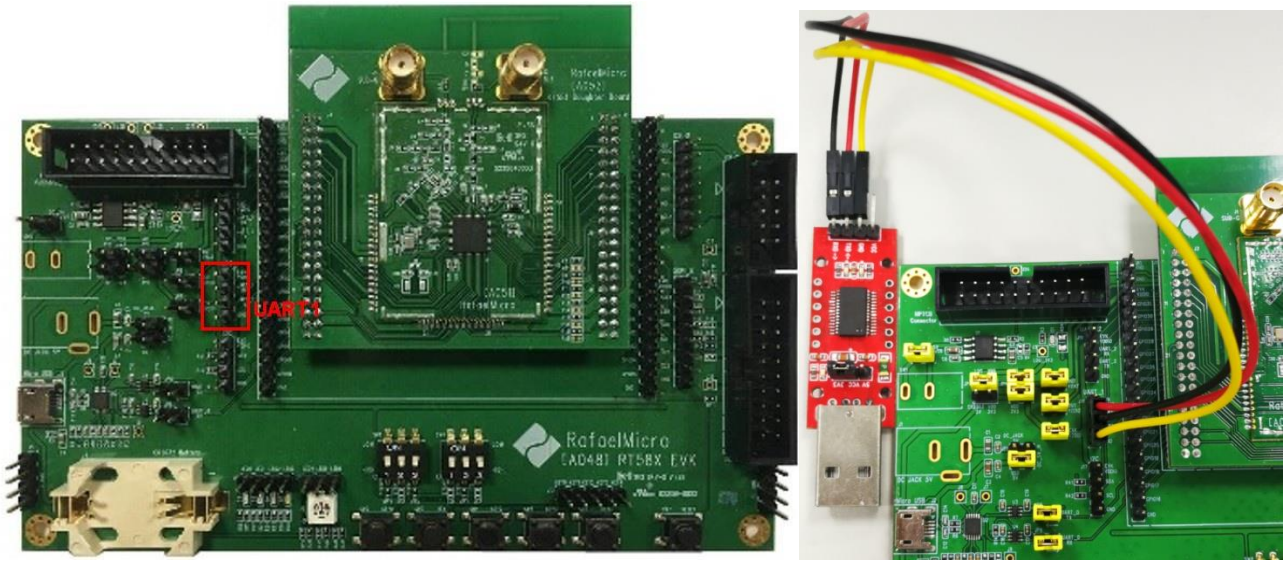
```
Get MESH_PRV_PDU_COMPLETE
Provision success
device uuid: 0x030903030308030006f030e040e0403
element address 0x0100
element count 1

~# appkeyadd 0x0100
add app key to node 0x0100
~# config appkey status [src address: 0x0100]
status 0x00, index: 0
modelappbind 0x0100 0x1000
bind device 0x0100's app key to model 0x1000
~# config model app status [src address: 0x0100]
status 0x00, element_address 0x0100, appkey index: 0 sig model ID 0x1000
```

[Control device]

After configuring the device, we can control the model that successfully binds the App key

```
~#
~# onoff 0x0100 0
set onoff status 0 to node 0x0100
~# generic onoff status [src address: 0x0100]
App key index: 0
present onoff: 0
onoff 0x0100 1
set onoff status 1 to node 0x0100
~# generic onoff status [src address: 0x0100]
App key index: 0
present onoff: 1
```

2.2.2 Running the demo with UART Command



The user can use the UART to USB dongle and connect to EVK board UART1 port to send the hex format command with an appropriate PC application tool. For the details, please refer to the document, [SW_21]RT58x_BLE_Mesh_Gateway_Command_Manual_V0.1.pdf.

(intentionally blank)

Revision History

Revision	Description	Owner	Date
1.0	Initial version.	Nat	2022/10/20
1.1	Modify vendor model	Nat	2022/12/06
1.2	Add gateway demo	Nat	2023/06/06