Steps to Execute Demo_54

Silicon Labs Product Documentation

Exported on 07/14/2023

Table of Contents

1 Introduction:

The purpose of this demo is to execute different protocols individually under opermode 0x109.

2 Configurations of Application

1. Select demo from 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define UNIFIED_PROTOCOL        1 //Set this to 1
```

Note:
Make sure to set remaining all demo Macros to 0

3 BLE alone requirments

3.1 BLE multiple connections test

3.1.1 Steps to test 'Gatt Notifications' in multi connections:

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST    0 //Set this to 0

#define    RSI_ENABLE_BT_TEST    0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST    0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST     0 //Set this to 0
```

```
#define RSI_BLE_MULTICONN_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_POWER_NUM_TEST        0 //Set this to 0

#define RSI_BLE_ADVERTISING_TEST        0 //Set this to 0

#define RSI_BLE_SANITY_TEST        0 //Set this to 0
```

```
#define CHECK_NOTIFICATIONS      1 //Set this to
```

> **Note:**
> Make sure to set below macros to 0
>
> ```
> #define CHECK_WRITE_WITHOUT_RESP        0 //Set this to 0
>
> #define CHECK_WRITE_PROPERTY        0 //Set this to 0
>
> #define CHECK_INDICATIONS        0 //Set this to 0
> ```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h'.

   [**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

   To identify remote device with BD Address/device name

   ```
   #define CONNECT_OPTION CONN_BY_NAME //CONN_BY_NAME or CONN_BY_ADDR
   ```

   If CONNECT_OPTION is set to CONN_BY_NAME, configure the below macros.

   ```
   Add the remote BLE device name to connect

   #define RSI_REMOTE_DEVICE_NAME1                      "slave1"
   #define RSI_REMOTE_DEVICE_NAME2                      "slave2"
   #define RSI_REMOTE_DEVICE_NAME3                      "slave3"
   ```

   If CONNECT_OPTION is set to CONN_BY_ADDR, configure the below macros.

   ```
   Configure the address type of remote device as either Public Address or Random
   Address

   #define RSI_BLE_DEV_ADDR_TYPE LE_PUBLIC_ADDRESS //!LE_PUBLIC_ADDRESS or
   LE_RANDOM_ADDRESS

   Add the BD Address of remote BLE device to connect

   #define RSI_BLE_DEV_1_ADDR "88:DA:1A:FE:2A:2C"
   #define RSI_BLE_DEV_2_ADDR "7E:E6:5E:30:77:6F"
   #define RSI_BLE_DEV_3_ADDR "70:1A:69:32:7C:8E"
   ```

3. Compile the project and flash the binary onto FRDM-K28.
4. Module starts advertising and scanning.
5. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it
6. After connection establishment, application initiates MTU exchange request.
7. Mobile starts service discovery and enables notification for RSI_BLE_ATTRIBUTE_1_UUID(0x1AA1).
8. Module sends continuous notifications to mobile..
9. Start advertising the slave devices.
10. Once the advertised slave device information (BD Address or name) is matched, the module tries to connect to slave.

11. After successful connection, Modules starts service discovery and enables notification for Heart Rate(0x180D) profile and receives continuous notifications from that profile.
12. Slave and Mobile devices may send read request for reading data on different service UUIDs of module.
13. Above procedure repeats for all connections (2 Masters + 3 slaves).

3.1.2 Steps to test 'Gatt Indications' in multi connections:

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST    0 //Set this to 0

#define    RSI_ENABLE_BT_TEST    0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST    0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST    0 //Set this to 0
```

```
#define CHECK_INDICATIONS              1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define CHECK_WRITE_WITHOUT_RESP    0 //Set this to 0

#define CHECK_WRITE_PROPERTY        0 //Set this to 0

#define CHECK_NOTIFICATIONS        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
```

```
#define     RSI_ADV_DATA_NAME                             "RSI_BLE_UNIFIED_DEMO"

To send the indication acknowledgement from host, configure the below macro
#define    RSI_BLE_INDICATE_CONFIRMATION_FROM_HOST    0x01    //! 0x01 -
enable, 0x00 - disable
```

To identify remote device with BD Address/device name

```
#define CONNECT_OPTION CONN_BY_NAME //CONN_BY_NAME or CONN_BY_ADDR
```

If CONNECT_OPTION is set to CONN_BY_NAME, configure the below macros.

```
Add the remote BLE device name to connect

#define RSI_REMOTE_DEVICE_NAME1     "slave1"
#define RSI_REMOTE_DEVICE_NAME2     "slave2"
#define RSI_REMOTE_DEVICE_NAME3     "slave3"
```

If CONNECT_OPTION is set to CONN_BY_ADDR, configure the below macros.

```
Configure the address type of remote device as either Public Address or Random
Address

#define RSI_BLE_DEV_ADDR_TYPE LE_PUBLIC_ADDRESS //!LE_PUBLIC_ADDRESS or
LE_RANDOM_ADDRESS

Add the BD Address of remote BLE device to connect

#define RSI_BLE_DEV_1_ADDR "88:DA:1A:FE:2A:2C"
#define RSI_BLE_DEV_2_ADDR "7E:E6:5E:30:77:6F"
#define RSI_BLE_DEV_3_ADDR "70:1A:69:32:7C:8E"
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection establishment, application initiates MTU exchange request.
6. Start advertising the slave devices.
7. Once the advertised slave device information (BD Address or name) is matched, the module tries to connect to slave.
8. After connection establishment, application initiates MTU exchange requests.
9. Module starts service discovery and enables the indication for Health Thermometer(0x1809) and Temperature Measurement(0x2A1C) attribute for client characteristic configuration descriptor.
10. At mobile side, write/update the value of Temperature measurement(0x2A1C) attribute.
11. Module receives the indication and application acknowledge it.
12. Slave and Mobile devices may send read request for reading data on different service UUIDs of module.
13. Repeat steps 9 and 10 for remaining slave and master devices.

3.1.3 Steps to test 'write without response' in multi connections:

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST    0 //Set this to 0

#define    RSI_ENABLE_BT_TEST    0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST    0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST    0 //Set this to 0
```

```
#define CHECK_WRITE_WITHOUT_RESP           1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define CHECK_NOTIFICATIONS        0 //Set this to 0

#define CHECK_WRITE_PROPERTY        0 //Set this to 0

#define CHECK_INDICATIONS        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h'.

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                        "RSI_BLE_UNIFIED_DEMO"
```

To identify remote device with BD Address/device name

```
#define CONNECT_OPTION CONN_BY_NAME //CONN_BY_NAME or CONN_BY_ADDR
```

If CONNECT_OPTION is set to CONN_BY_NAME, configure the below macros.

```
Add the remote BLE device name to connect
```

```
#define RSI_REMOTE_DEVICE_NAME1      "slave1"
#define RSI_REMOTE_DEVICE_NAME2      "slave2"
#define RSI_REMOTE_DEVICE_NAME3      "slave3"
```

If CONNECT_OPTION is set to CONN_BY_ADDR, configure the below macros.

```
Configure the address type of remote device as either Public Address or Random
Address

#define RSI_BLE_DEV_ADDR_TYPE LE_PUBLIC_ADDRESS //!LE_PUBLIC_ADDRESS or
LE_RANDOM_ADDRESS

Add the BD Address of remote BLE device to connect

#define RSI_BLE_DEV_1_ADDR "88:DA:1A:FE:2A:2C"
#define RSI_BLE_DEV_2_ADDR "7E:E6:5E:30:77:6F"
#define RSI_BLE_DEV_3_ADDR "70:1A:69:32:7C:8E"

RSI_BLE_NEW_SERVICE_UUID
RSI_BLE_ATTRIBUTE_1_UUID
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection establishment, application initiates MTU exchange request.
6. Start advertising the slave devices.
7. Once the advertised slave device information (BD Address or name) is matched, the module tries to connect to slave.
8. After connection establishment, application initiates MTU exchange requests.
9. Module starts service discovery and writes to Immediate Alert profile (0x1802).
10. Slave and Mobile devices may send read request for reading data on different service UUIDs of module.
11. Step8 repeats for all the connections (2 masters + 3 slaves)]

3.1.4 Steps to test 'write with response' in multi connections:

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARBLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST         0 //Set this to 0
```

```
#define    RSI_ENABLE_BT_TEST        0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define CHECK_WRITE_PROPERTY            1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define CHECK_WRITE_WITHOUT_RESP        0 //Set this to 0

#define CHECK_NOTIFICATIONS        0 //Set this to 0

#define CHECK_INDICATIONS        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"
```

To identify remote device with BD Address/device name

```
#define CONNECT_OPTION CONN_BY_NAME //CONN_BY_NAME or CONN_BY_ADDR
```

If CONNECT_OPTION is set to CONN_BY_NAME, configure the below macros.

```
Add the remote BLE device name to connect

#define RSI_REMOTE_DEVICE_NAME1     "slave1"
#define RSI_REMOTE_DEVICE_NAME2     "slave2"
#define RSI_REMOTE_DEVICE_NAME3     "slave3"
```

If CONNECT_OPTION is set to CONN_BY_ADDR, configure the below macros.

```
Configure the address type of remote device as either Public Address or Random
Address

#define RSI_BLE_DEV_ADDR_TYPE LE_PUBLIC_ADDRESS //!LE_PUBLIC_ADDRESS or
LE_RANDOM_ADDRESS
```

```
Add the BD Address of remote BLE device to connect

#define RSI_BLE_DEV_1_ADDR "88:DA:1A:FE:2A:2C"
#define RSI_BLE_DEV_2_ADDR "7E:E6:5E:30:77:6F"
#define RSI_BLE_DEV_3_ADDR "70:1A:69:32:7C:8E"

RSI_BLE_NEW_SERVICE_UUID
RSI_BLE_ATTRIBUTE_1_UUID
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection establishment, application initiates MTU exchange request.
6. Start advertising the slave devices.
7. Once the advertised slave device information (BD Address or name) is matched, the module tries to connect to slave.
8. After connection establishment, application initiates MTU exchange requests.
9. Module starts service discovery and writes to Heart Rate profile (0x1802) continuously based on write response, see the console logs to observe 'write responses for each connection'.
10. Slave and Mobile devices may send read request for reading data on different service UUIDs of module.
11. Step8 repeats for all the connections (2 masters + 3 slaves).

3.1.5 steps to check 'Whitelist procedure' in multi connection:

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define RSI_ENABLE_WHITELIST            1 //Set this to 1
```

> **Note:**
> Make sure to configure any of below macros to test data transfer
>
> ```
> #define CHECK_WRITE_WITHOUT_RESP        0 //Set this to 0
>
> #define CHECK_NOTIFICATIONS        0 //Set this to 0
>
> #define CHECK_INDICATIONS        0 //Set this to 0
>
> #define CHECK_WRITE_PROPERTY        0 //Set this to 0
> ```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

   [**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

   ```
   Configure the name of the device to advertise
   #define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"
   ```

   ```
   #define SMP_ENABLE                  1 //set this to 1
   By default this macro is set to 0
   ```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection, device adds to 'Whitelist' and corresponding logs can be seen in console.
6. After adding, application initiates MTU exchange request.
7. Module starts service discovery and pairing to remote device happens

> **Note:**
> 'Whitelist' procedure is verified only in single connection when DUT is slave.

3.2 BLE Power save numbers test

3.2.1 steps to test:

1. Configure following parameters in 'rsi_common_app.h'.

   ```
   [File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
   ```

   ```
   #define RSI_ENABLE_BLE_TEST     1 //Set this to 1
   ```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST         0 //Set this to 0

#define    RSI_ENABLE_BT_TEST         0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST         0 //Set this to 0
```

```
#define RSI_BLE_POWER_NUM_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_SANITY_TEST            0 //Set this to 0

#define RSI_BLE_ADVERTISING_TEST         0 //Set this to 0

#define RSI_BLE_MULTICONN_TEST         0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h'

   [**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

   ```
   Configure the name of the device to advertise
   #define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"
   ```

3. Module starts advertising and enters power save mode. Calculate power number.
4. Scan and connect the module to remote device and calculate power number.

3.3 BLE Advertising test

3.3.1 steps to test:

1. Configure following parameters in 'rsi_common_app.h'.

   ```
   [File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
   ```

```
#define RSI_ENABLE_BLE_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define RSI_BLE_ADVERTISING_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_SANITY_TEST            0 //Set this to 0

#define RSI_BLE_POWER_NUM_TEST        0 //Set this to 0

#define RSI_BLE_MULTICONN_TEST        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h'

[**File path:-** RSI_SDK_WEARABLEs_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"
```

3. Module starts advertising with an interval of 20ms+[0-10]ms random offset, verify the interval in sniffer logs.

3.4 BLE SANITY TEST

3.4.1 steps to test notifications in sanity test:

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define RSI_BLE_SANITY_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_POWER_NUM_TEST        0 //Set this to 0

#define RSI_BLE_ADVERTISING_TEST        0 //Set this to 0

#define RSI_BLE_MULTICONN_TEST        0 //Set this to 0
```

```
#define CHECK_NOTIFICATIONS             1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define CHECK_INDICATIONS        0 //Set this to 0

#define WRITE_TO_READONLY_CHAR_TEST        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"
```

Configure below paramater to enable pairing

```
#define SMP_ENABLE                 1 //Set this to 1
//By default this is set to '0'
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection establishment, application initiates MTU exchange request.
6. Mobile starts service discovery and enables notification for RSI_BLE_ATTRIBUTE_1_UUID(0x1AA1).
7. Module sends continuous notifications to the mobile.

3.4.2 steps to execute Gatt indications sanity test :

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST         0 //Set this to 0

#define    RSI_ENABLE_BT_TEST         0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST         0 //Set this to 0
```

```
#define RSI_BLE_SANITY_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_POWER_NUM_TEST        0 //Set this to 0

#define RSI_BLE_ADVERTISING_TEST        0 //Set this to 0

#define RSI_BLE_MULTICONN_TEST        0 //Set this to 0
```

```
#define CHECK_INDICATIONS               1 //Set this to
```

Note:
Make sure to set below macros to 0

```
#define CHECK_NOTIFICATIONS        0 //Set this to 0

#define WRITE_TO_READONLY_CHAR_TEST        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"

To send the indication acknowledgement from host, configure the below macro
#define    RSI_BLE_INDICATE_CONFIRMATION_FROM_HOST    0x01    //! 0x01 -
enable, 0x00 - disable
```

Configure below parameter to enable pairing

```
#define SMP_ENABLE              1 //Set this to 1
//By default this is set to '0'
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection establishment, application initiates MTU exchange request.
6. Module starts service discovery and enables the indication for Health Thermometer(0x1809) and Temperature Measurement(0x2A1C) attribute for client characteristic configuration descriptor.
7. At mobile side, write/update the value of Temperature measurement(0x2A1C) attribute.
8. Module receives the indication and application acknowledge it.

3.4.3 steps to execute 'BLE disconnect on authentication failure' :

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST         0 //Set this to 0

#define    RSI_ENABLE_BT_TEST         0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST         0 //Set this to 0
```

```
#define RSI_BLE_SANITY_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_POWER_NUM_TEST         0 //Set this to 0

#define RSI_BLE_ADVERTISING_TEST         0 //Set this to 0

#define RSI_BLE_MULTICONN_TEST         0 //Set this to 0
```

Configure below parameter to enable pairing

```
#define SMP_ENABLE              1 //Set this to 1
//By default this is set to '0'
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

   [**File path:-** RSI_SDK_WEARABLES/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                         "RSI_BLE_UNIFIED_DEMO"
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Run nRF connect on mobile, scan for RSI_ADV_DATA_NAME and connect to it.
5. After connection establishment, application initiates MTU exchange request.
6. Module starts service discovery and initiates SMP procedure.
7. Enter the smp passkey other than displayed in serial console.
8. Verify that disconnect occurs on authentication failure and that error condition is reported in callbacks

3.4.4 steps to execute 'write to read only characteristic' in sanity test :

1. Configure following parameters in 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BLE_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define    RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0

#define    RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define RSI_BLE_SANITY_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_BLE_POWER_NUM_TEST        0 //Set this to 0

#define RSI_BLE_ADVERTISING_TEST        0 //Set this to 0

#define RSI_BLE_MULTICONN_TEST        0 //Set this to 0
```

```
#define WRITE_TO_READONLY_CHAR_TEST             1 //Set this to
```

Note:
Make sure to set below macros to 0

```
#define CHECK_NOTIFICATIONS        0 //Set this to 0

#define CHECK_INDICATIONS        0 //Set this to 0
```

2. Configure following parameters in 'rsi_ble_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
Configure the name of the device to advertise
#define     RSI_ADV_DATA_NAME                            "RSI_BLE_UNIFIED_DEMO"
```

To identify remote device with BD Address/device name

```
#define CONNECT_OPTION CONN_BY_NAME //CONN_BY_NAME or CONN_BY_ADDR
```

If CONNECT_OPTION is set to CONN_BY_NAME, configure the below macros.

```
Add the remote BLE device name to connect

#define RSI_REMOTE_DEVICE_NAME1                          "slave1"
```

If CONNECT_OPTION is set to CONN_BY_ADDR, configure the below macros.

```
Configure the address type of remote device as either Public Address or Random
Address

#define RSI_BLE_DEV_ADDR_TYPE LE_PUBLIC_ADDRESS //!LE_PUBLIC_ADDRESS or
LE_RANDOM_ADDRESS

Add the BD Address of remote BLE device to connect

#define RSI_BLE_DEV_1_ADDR "88:DA:1A:FE:2A:2C"
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Advertise the slave device.
5. Once the advertised slave device information (BD Address or name) is matched, the module tries to connect to slave.
6. After connection establishment, application initiates MTU exchange request.
7. Module starts service discovery and writes to 'Body sensor location (0x2A38)' characteristic in Heart Rate profile (0x180D)
8. As Write cannot happen to read only characteristic, module reports error in callback and respective prints can be seen in serial console.

4 BT alone requirments:

4.1 steps to execute BT functionalities :

1. configure following parameters in 'rsi_common_app.h',

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BT_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

2. Configure following parameters in 'rsi_bt_config_DEMO_54.h'

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

Enter the remote BT device address as the value to RSI_BT_REMOTE_BD_ADDR

```
#define     RSI_BT_REMOTE_BD_ADDR                        (void
*)"B8:D5:0B:9B:D6:B2"
```

Set below macro to start bt inquiry

```
#define     INQUIRY_ENABLE      1 //By default this macro is set to '0'
```

Configure below macro to call remote name request explicitly followed by connection

```
#define     INQ_REMOTE_NAME_REQUEST     1 //By default this macro is set to '1'
```

Configure below macro, to check inquiry+connection

```
#define     INQUIRY_CONNECTION_SIMULTANEOUS     1 //By default this macro is
set to '0'
```

3. This Demo supports PCM audio inputs.

4. This frame work has MP3 Decoder (Evaluation library) and SBC Encoder (GPL library) in host MCU, which host will send SBC Audio to RS9116.
5. Copy 'pcm.wav' to SD card and plug into FRDM SD slot.
6. Compile the project and flash the binary onto FRDM-K28.
7. Connect any serial terminal (Teraterm/cutecom) to FRDM-K28 serial interface (would be displayed as 'mbed port') for debug prints. \[Baudrate = 115200\].
8. Configure the remote BT device in pairing mode.
9. Start FRDM-K28, demo application starts executing and respective prints are displayed on serial console.
10. Remote BT device will get paired up with the Silabs Module and songs stored in SD card will be played in sequence.
11. To test the BT power numbers, configure below paramater.

```
[File path:-
RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54/
```

Note:
By default this macro is set to 0

```
#define MEASURE_AUDIO_POWER_NUMBER       1 //Set this to 1
```

4.2 BT Features: Extended Inquiry Response

4.2.1 Requirement:

Inquiry can be performed multiple times in a crowded RF environment with a pair of recommended headphones (with a friendly name). The friendly name shall be resolved within 30 seconds.

4.2.2 Steps to execute:

1. configure following parameters in 'rsi_common_app.h',

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BT_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0
```

```
#define RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST         0 //Set this to 0
```

configure below macro

```
#define BT_EIR_FRIENDLY_NAME_TEST       1 //Set this to 1
```

Note:
By default this macro is set to '0'

2. Add remote device friendly names in 'rsi_bt_config_DEMO_54.h'

```
[File path:-
RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54/
rsi_bt_config_DEMO_54.h]
```

```
#define RSI_BT_REMOTE_DEV1_NAME                          "Name1"
#define RSI_BT_REMOTE_DEV2_NAME                          "Name2"
```

```
#define RSI_APP_AVDTP_ROLE                                    INITIATOR_ROLE
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Connect any serial terminal (Teraterm/cutecom) to FRDM-K28 serial interface (would be displayed as 'mbed port') for debug prints. \[Baudrate = 115200\].
5. Start FRDM-K28, demo application starts executing and respective prints are displayed on serial console.
6. Configure the above devices in pairing mode along with some other BT headsets/remote devices(to create crowded environment).
7. Application starts inquiry multiple times untill above configured friendly names are retrieved in inquiry response.
8. Inquiry stops after finding the desired results and calculates the inquiry time.
9. The inquiry time should be <30sec.

4.3 BT Power consumption

4.3.1 Steps to execute:

1. configure following parameters in 'rsi_common_app.h',

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_BT_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

configure below macro

```
#define CHECK_BT_POWER_STATE       1 //Set this to 1
```

Note:
Makesure to set remaining macros to '0'
#define BT_EIR_FRIENDLY_NAME_TEST   0

2. Configure below macros in 'rsi_bt_config_DEMO_54.h'

```
[File path:-
RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54/
rsi_bt_config_DEMO_54.h]
```

Enter the remote BT device address as the value to RSI_BT_REMOTE_BD_ADDR

```
#define     RSI_BT_REMOTE_BD_ADDR                        (void
*)"B8:D5:0B:9B:D6:B2"
```

Set below macro to start bt inquiry

```
#define     INQUIRY_ENABLE      1 //By default this macro is set to '0'
```

Makesure to set below macros to '0'

```
#define INQUIRY_CONNECTION_SIMULTANEOUS      '0'
#define INQ_REMOTE_NAME_REQUEST              '0'
```

```
#define RSI_APP_AVDTP_ROLE                         INITIATOR_ROLE
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Connect any serial terminal (Teraterm/cutecom) to FRDM-K28 serial interface (would be displayed as 'mbed port') for debug prints. \[Baudrate = 115200\].
5. Start FRDM-K28, demo application starts executing and respective prints are displayed on serial console.
6. When application starts, initially device will be in deepsleep.
7. After 30sec , device starts remote device inquiry untill above configured device is retrieved in inquiry response.
8. Inquiry stops after finding the desired results and starts sending audio data.
9. Device goes into deepsleep powersave when headset is disconnected(disconnect at random time).

5 WLAN alone requirments

5.1 WLAN SSL receive max throughput

5.1.1 steps to execute SSL recieve max throughput :

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST        1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define SSL_RX_MAX_THROUGHPUT_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST    0 //Set this to 0

#define WIFI_DEEPSLEEP_STANDBY_TEST            0 //Set this to 0

#define STA_SCAN_CONN_DISCNCT_TEST            0 //Set this to 0

#define MULTITHREADED_HTTP_DOWNLOAD_TEST    0 //Set this to 0

#define MULTITHREADED_TCP_TX_TEST            0 //Set this to 0
```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
- SSID ---------------------Name of the Access Point SSID to connect
- SECURITY_TYPE ------------Security type of Access Point to join
- PSK ----------------------Access Point passphrase
- SERVER_IP_ADDRESS---------SSL Server IP address
- SERVER_PORT --------------SSL Server port
```

3. Compile the project and flash the binary onto FRDM-K28
4. Copy 'SSL_tx_throughput.py' to any remote PC.

   [**File path:-** RSI_SDK_WEARABLES_vx/utilities/scripts/SSL_tx_throughput.py]

5. Connect remote PC to the same Access Point to which Silabs Module would be connected.
6. Make sure to copy 'server-cert.pem' and 'server-key.pem' files in the same directory, where the server started.

   [**File path:-** RSI_SDK_WEARABLES_vx/utilities/certificates/]

7. Start SSL server by running 'SSL_tx_throughput.py' using below command.

```
#python SSL_tx_throughput.py
```

> **Note:**
> By default SSL server runs on port 5001. Same is configured in 'SERVER_PORT' in 'rsi_wlan_config_DEMO_54.h'.

8. Start FRDM-K28, UNIFIED_PROTOCOL_DEMO_54 demo application starts executing and respective prints are displayed on serial console.
9. After receiving 10000 packets, throughput would be printed on serial terminal.

5.2 WLAN secure and TLS socket connections and disconnections

5.2.1 steps to execute WLAN secure connection and disconnection :

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST        1 //Set this to 1
```

> **Note:**
> Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST  1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define SSL_RX_MAX_THROUGHPUT_TEST            0 //Set this to 0

#define WIFI_DEEPSLEEP_STANDBY_TEST            0 //Set this to 0

#define STA_SCAN_CONN_DISCNCT_TEST            0 //Set this to 0

#define MULTITHREADED_HTTP_DOWNLOAD_TEST    0 //Set this to 0

#define MULTITHREADED_TCP_TX_TEST            0 //Set this to 0
```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h'

   [**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
- SECURE_CON_DISCON --------1
- TCP_TLS_CON_DISCON -------0
- SSID ------------Name of the Access Point SSID to connect
- SECURITY_TYPE ------------Security type of Access Point to join
- PSK ------------Access Point passphrase
```

3. Compile the project and flash the binary onto FRDM-K28.
4. Start FRDM-K28, application starts executing and respective prints are displayed on serial console.

5.2.2 steps to execute TLS SOCKET connection and disconnection :

1. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

   [**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
- SECURE_CON_DISCON --------0
- TCP_TLS_CON_DISCON -------1
- SSID ------------Name of the Access Point SSID to connect
- SECURITY_TYPE ------------Security type of Access Point to join
- PSK ------------Access Point passphrase
- SERVER_PORT ------------SSL Server port
- SERVER_IP_ADDRESS---------SSL Server IP address
```

2. Compile the project and flash the binary onto FRDM-K28.
3. Connect remote PC to the same Access Point to which Silabs Module would be connected.
4. Copy server-cert.pem and server-key.pem to any remote PC preferably Linux

```
[File  path:- RSI_SDK_WEARABLES_vx/utilities/certificates]
```

5. Run SSL server using below command. Make sure 'server-cert.pem' and 'server-key.pem' files are in the same directory.

```
#openssl s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
```

> **Note:**
> By default SSL server runs on port 5001. Same is configure in 'SERVER_PORT' in 'rsi_wlan_config_DEMO_54.h'.

6. Connect remote PC to the same Access Point to which Silabs Module would be connected.
7. Start FRDM-K28, application starts executing and respective prints are displayed on serial console.

5.3 WLAN STA Scan,connection and disconnection

5.3.1 steps to test scan, connection and disconnection:

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST       1 //Set this to 1
```

> **Note:**
> Make sure to set below macros to 0
>
> `#define RSI_ENABLE_BT_TEST           0 //Set this to 0`
>
> `#define RSI_ENABLE_BLE_TEST            0 //Set this to 0`

```
#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define STA_SCAN_CONN_DISCNCT_TEST          1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST    0 //Set this to 0

#define WIFI_DEEPSLEEP_STANDBY_TEST            0 //Set this to 0

#define SSL_RX_MAX_THROUGHPUT_TEST            0 //Set this to 0

#define MULTITHREADED_HTTP_DOWNLOAD_TEST    0 //Set this to 0

#define MULTITHREADED_TCP_TX_TEST            0 //Set this to 0
```

2. Configure following parameters in rsi_wlan_http_s_DEMO_54.c and rsi_wlan_config_DEMO_54.h

```
struct rsi_wlan_con_discon_s rsi_wlan_con_discon[RSI_MAX_NO_OF_AP_TO_CONNECT]=
{
        { "power", "12345678",RSI_WPA2},
        { "kill", "12345678",RSI_WPA2},
        { "rps", "12345678",RSI_WPA2},
        { "Hotspot", "12345678",RSI_WPA2},
        { "Name", "12345678",RSI_WPA2},
        { "scan", "12345678",RSI_WPA2},
        { "vivo", "12345678",RSI_WPA2},
        { "run", "12345678",RSI_WPA2},
        { "mark", "12345678",RSI_WPA2},
        { "bssid", "12345678",RSI_OPEN}
};

rsi_wlan_config_DEMO_54.h
#define RSI_MAX_NO_OF_AP_TO_CONNECT                    10
#define RSI_MAX_NO_OF_ITERATIONS                       10
#define RSI_STATUS_RETURN_ON_FAILURE                    0
#define RSI_NO_OF_RETRIES                              2
```

3. Compile the project and flash the binary onto FRDM-K28
4. Start FRDM-K28, UNIFIED_PROTOCOL_DEMO_54 demo application starts executing and should scan/connect/disconnect to the configured AP and should run the total no of iterations configured.

5.4 WLAN Multithreaded HTTP download test

5.4.1 steps to execute Multithreaded HTTP download test :

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

2. Select demo from 'rsi_common_app.h'.

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define MULTITHREADED_HTTP_DOWNLOAD_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST    0 //Set this to 0
#define WIFI_DEEPSLEEP_STANDBY_TEST            0 //Set this to 0

#define STA_SCAN_CONN_DISCNCT_TEST            0 //Set this to 0

#define SSL_RX_MAX_THROUGHPUT_TEST            0 //Set this to 0

#define MULTITHREADED_TCP_TX_TEST            0 //Set this to 0
```

3. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

```
- SSID ------------Name the of Access Point SSID to connect
- SECURITY_TYPE ------------Security type of Access Point to join
- PSK ------------Access Point passphrase
- SERVER_IP_ADDRESS ---------server IP address
```

4. Configure following parameters in 'rsi_sock_test_DEMO_54.h',

```
- SOCKTEST_INSTANCES_MAX ------------Number of threads (1 to 4)
- NO_OF_ITERATIONS ------------ Number of times the threads to download
- BYTES_TO_RECEIVE ----------- File size to receive.
```

5. Compile the project and flash the binary onto FRDM-K28
6. Copy 'simple_http_server.py'

   [**File path:-** RSI_SDK_WEARABLES_vx/utilities/scripts]

7. Connect remote PC to the same Access Point to which Silabs Module would be connected.
8. Start HTTP server on the remote (x86, preferred) using 'simple_http_server.py'.

   [**File path:-** RSI_SDK_WEARABLES_vx/utilities/scripts/simple_http_server.py]

9. Run 'simple_http_server.py' using below command. Make sure 'dltestdata32.txt' (based on the http_request_str_first[] http header requested file in rsi_sock_test_DEMO_54.c)file is present in the same folder as 'simple_http_server.py'.

```
#python simple_http_server.py 80
```

10. Start FRDM-K28,demo application starts executing and respective prints are displayed on serial console.
11. To covert the incoming data endianess, configure the following parameter in rsi_sock_test_DEMO_54.c

```
BIG_ENDIAN_CONVERSION_REQUIRED ----- 1 //set this to 1
```

5.5 WLAN Multithreaded TCP TX test

5.5.1 steps to test Multithreaded TCP TX test :

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST        1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define MULTITHREADED_TCP_TX_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST    0 //Set this to 0

#define WIFI_DEEPSLEEP_STANDBY_TEST           0 //Set this to 0

#define STA_SCAN_CONN_DISCNCT_TEST            0 //Set this to 0

#define MULTITHREADED_HTTP_DOWNLOAD_TEST    0 //Set this to 0

#define SSL_RX_MAX_THROUGHPUT_TEST            0 //Set this to 0
```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

```
- SSID ------------Name the of Access Point SSID to connect
- SECURITY_TYPE ------------Security type of Access Point to join
- PSK ------------Access Point passphrase
- SERVER_IP_ADDRESS---------TCP server IP address
- SERVER_PORT ------------- server port number
```

3. Configure following parameters in 'rsi_sock_test_DEMO_54.h',

```
- SOCKTEST_INSTANCES_MAX ------------Number of threads (1 to 4)
- NO_OF_ITERATIONS ------------ Number of times the threads to send the data
- BYTES_TO_TRANSMIT ----------- Number of bytes to send
```

4. Compile the project and flash the binary onto FRDM-K28
5. Connect remote PC to the same Access Point to which Silabs Module would be connected.

6. Run 'iPerf' as TCP Server on remote PC using below command,

```
#iperf -s -i 1
```

7. Start FRDM-K28, demo application starts executing and respective prints can be seen in console.

5.6 WLAN Powersave

5.6.1 steps to execute standby state :

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST        1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define WIFI_DEEPSLEEP_STANDBY_TEST     1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST    0 //Set this to 0

#define SSL_RX_MAX_THROUGHPUT_TEST            0 //Set this to 0

#define STA_SCAN_CONN_DISCNCT_TEST            0 //Set this to 0

#define MULTITHREADED_HTTP_DOWNLOAD_TEST    0 //Set this to 0
```

```
#define MULTITHREADED_TCP_TX_TEST            0 //Set this to 0
```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h'

```
- SSID -----------Name of the Access Point SSID to connect
- SECURITY_TYPE -----------Security type of Access Point to join
- PSK -----------Access Point passphrase
```

3. Set following parameters in 'rsi_wlan_config_DEMO_54.h',

```
-> Set 'WLAN_STANDBY' to 1 --Standby Enable
-> Set 'WLAN_DEEPSLEEP' to 0
-> Set 'DP_SLEEP_WITH_RAM_RET' to 0
```

4. Compile the project and flash the binary into FRDM-K28
5. Start FRDM-K28 , RSI module connects to the Access Point and goes into STANDBY mode ,respective demo prints are displayed on serial console.
6. Measure current consumption.

5.6.2 Steps to execute DEEPSLEEP (With RAM Retention):

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54

```
-> Set 'WLAN_DEEPSLEEP' to 1 ------------Enable Deepsleep
-> Set 'DP_SLEEP_WITH_RAM_RET' to 1 --------Enable Sleep with RAM retention
-> Set 'WLAN_STANDBY' to 0------------Disable standby mode
```

3. Compile the project and flash the binary into FRDM-K28
4. Start FRDM-K28 , RSI module WLAN radio is initialized and enter DEEPSLEEP mode ,respective demo prints are displayed on serial console.
5. Measure current consumption.

5.6.3 Steps to execute DEEPSLEEP (Without RAM Retention):

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST       1 //Set this to 1
```

> **Note:**
> Make sure to set below macros to 0
>
> ```
> #define RSI_ENABLE_BT_TEST           0 //Set this to 0
>
> #define RSI_ENABLE_BLE_TEST            0 //Set this to 0
>
> #define RSI_ENABLE_PROP_PROTOCOL_TEST           0 //Set this to 0
>
> #define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
> ```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
-> Set 'WLAN_DEEPSLEEP' to 1 ------------Enable Deepsleep
-> Set 'DP_SLEEP_WITH_RAM_RET' to 0 --------Enable Sleep with RAM retention
-> Set 'WLAN_STANDBY' to 0------------Disable standby mode
```

3. Compile the project and flash the binary into FRDM-K28
4. Start FRDM-K28 , RSI module WLAN radio is initialized and enter DEEPSLEEP mode ,respective demo prints are displayed on serial console.
5. Measure current consumption.

5.7 WLAN Power Consumption:

5.7.1 steps to execute :

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_ENABLE_WIFI_TEST       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_BT_TEST           0 //Set this to 0

#define RSI_ENABLE_BLE_TEST            0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST            0 //Set this to 0

#define    RSI_GENERAL_POWER_TEST        0 //Set this to 0
```

```
#define CHECK_WIFI_POWER_STATE      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define SSL_RX_MAX_THROUGHPUT_TEST          0 //Set this to 0

#define WIFI_STA_AND_SSL_CONN_DISCONN_TEST    0 //Set this to 0

#define WIFI_DEEPSLEEP_STANDBY_TEST            0 //Set this to 0

#define STA_SCAN_CONN_DISCNCT_TEST            0 //Set this to 0

#define MULTITHREADED_HTTP_DOWNLOAD_TEST    0 //Set this to 0

#define MULTITHREADED_TCP_TX_TEST            0 //Set this to 0
```

2. Configure following parameters in 'rsi_wlan_config_DEMO_54.h',

[**File path:-** RSI_SDK_WEARABLES_vx/examples/UNIFIED_PROTOCOL_DEMO_54]

```
- SSID ---------------------Name of the Access Point SSID to connect
- SECURITY_TYPE ------------Security type of Access Point to join
- PSK ----------------------Access Point passphrase
```

3. Compile the project and flash the binary onto FRDM-K28
4. Initially device will be in deepsleep power save for 30sec.
5. After 30sec device will start scanning the configured accesspoint.
6. After successfull scan, device goes to deepsleep for 30sec.
7. After 30sec device initiates connection request to configured accesspoint and after connection goes to connected sleep for 10sec.
8. Device disconnects from associated AP after 10sec and moves in to deepsleep for 30sec.
9. After completion of 30sec, steps 5 to 8 repeats continuously.

6 General power test

6.1 Deep sleep immediately after reset

6.1.1 Steps to test Power consumption in deep sleep immediately after reset:

1. configure following parameters in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_GENERAL_POWER_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0
```

```
#define RSI_DEEP_SLEEP_WITH_RAM_RET_AFTER_RESET            1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_STANDBY        0 //Set this to 0
#define RSI_POWER_AFTER_CHIP_INIT        0 //Set this to 0

#define RSI_DEEP_SLEEP_WITH_N_WO_RAM_RET_AFTER_RESET    0 //Set this to
0

#define RSI_DEEP_SLEEP_WITHOUT_RAM_RET   0 //Set this to 0
```

2. Compile the project and flash the binary into FRDM-K28
3. Start FRDM-K28 , RSI module WLAN radio is initialized after reset and enters DEEPSLEEP mode ,respective demo prints are displayed on serial console.
4. Measure current consumption.

6.2 Power consumption immediately after chip initialization

6.2.1 Steps to test Power consumption immediately after chip initialization:

1. Configure below macro in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_GENERAL_POWER_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0
```

```
#define RSI_POWER_AFTER_CHIP_INIT                       1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_STANDBY        0 //Set this to 0
#define RSI_DEEP_SLEEP_WITH_RAM_RET_AFTER_RESET        0 //Set this to 0

#define RSI_DEEP_SLEEP_WITH_N_WO_RAM_RET_AFTER_RESET    0 //Set this to
0

#define RSI_DEEP_SLEEP_WITHOUT_RAM_RET   0 //Set this to 0
```

2. Compile the project and flash the binary into FRDM-K28
3. Start FRDM-K28 , RSI module WLAN radio is initialized and enters DEEPSLEEP mode ,respective demo prints are displayed on serial console.
4. Measure current consumption.

6.3 Deep sleep without RamRetention

6.3.1 Steps to test Power consumption in deep sleep without RamRetention:

1. configure following parameters in 'rsi_common_app.h'

   ```
   [File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
   ```

   ```
   #define RSI_GENERAL_POWER_TEST      1 //Set this to 1
   ```

   Note:
   Make sure to set below macros to 0

   ```
   #define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

   #define RSI_ENABLE_BLE_TEST        0 //Set this to 0

   #define RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

   #define    RSI_ENABLE_BT_TEST        0 //Set this to 0
   ```

   ```
   #define RSI_DEEP_SLEEP_WITHOUT_RAM_RET             1 //Set this to 1
   ```

   Note:
   Make sure to set below macros to 0

   ```
   #define RSI_STANDBY        0 //Set this to 0
   #define RSI_POWER_AFTER_CHIP_INIT        0 //Set this to 0

   #define RSI_DEEP_SLEEP_WITH_N_WO_RAM_RET_AFTER_RESET     0 //Set this to
   0

   #define RSI_DEEP_SLEEP_WITH_RAM_RET_AFTER_RESET              1 //Set
   this to 1
   ```

2. Compile the project and flash the binary into FRDM-K28
3. Start FRDM-K28 , RSI module WLAN radio is initialized after reset and enters in to DEEPSLEEP without ram retention mode,respective demo prints are displayed on serial console.
4. Measure current consumption.

6.4 Power consumption immediately after chip initialization

6.4.1 Steps to test Power consumption immediately after chip initialization:

1. Configure below macro in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_GENERAL_POWER_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST         0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0
```

```
#define RSI_POWER_AFTER_CHIP_INIT                          1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_STANDBY        0 //Set this to 0
#define RSI_DEEP_SLEEP_WITH_RAM_RET_AFTER_RESET        0 //Set this to 0

#define RSI_DEEP_SLEEP_WITH_N_WO_RAM_RET_AFTER_RESET    0 //Set this to
0

#define RSI_DEEP_SLEEP_WITHOUT_RAM_RET   0 //Set this to 0
```

2. Compile the project and flash the binary into FRDM-K28
3. Start FRDM-K28 , RSI module WLAN radio is initialized and enters DEEPSLEEP mode ,respective demo prints are displayed on serial console.
4. Measure current consumption.

6.5 Power consumption in standby state

6.5.1 Steps to test Power consumption in standby state:

1. Configure below macro in 'rsi_common_app.h'

```
[File path:- RSI_SDK_WEARABLES_vx/examples/inc/rsi_common_app.h]
```

```
#define RSI_GENERAL_POWER_TEST      1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_ENABLE_WIFI_TEST        0 //Set this to 0

#define RSI_ENABLE_BLE_TEST        0 //Set this to 0

#define RSI_ENABLE_PROP_PROTOCOL_TEST        0 //Set this to 0

#define    RSI_ENABLE_BT_TEST        0 //Set this to 0
```

```
#define RSI_STANDBY                                        1 //Set this to 1
```

Note:
Make sure to set below macros to 0

```
#define RSI_DEEP_SLEEP_WITH_RAM_RET_AFTER_RESET        0 //Set this to 0

#define RSI_POWER_AFTER_CHIP_INIT        0 //Set this to 0

#define RSI_DEEP_SLEEP_WITH_N_WO_RAM_RET_AFTER_RESET    0 //Set this to
0

#define RSI_DEEP_SLEEP_WITHOUT_RAM_RET   0 //Set this to 0
```

2. Compile the project and flash the binary into FRDM-K28
3. Start FRDM-K28 , RSI module connects to the Access Point and goes into STANDBY mode ,respective demo prints are displayed on serial console.
4. Measure current consumption.