ASR582X 系列

BLE 使用示例

ASR Confidential

文档版本 1.0.0

发布日期 2022-12-19

版权所有 © 2022 翱捷科技

关于本文档

本文档旨在指导了解 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片的 BLE 部分基础功能的使用示例，帮助用户快速了解 API 调用和相关回调的使用。

读者对象

本文档主要适用于以下工程师：

- 软件工程师
- 技术支持工程师

产品名称

本文档适用于 ASR582X 系列 Wi-Fi+BLE Combo SoC 芯片。

版权公告

版权归 © 2022 翱捷科技股份有限公司所有。保留一切权利。未经翱捷科技股份有限公司的书面许可，不得以任何形式或手段复制、传播、转录、存储或翻译本文档的部分或所有内容。

商标声明

ASR、翱捷和其他翱捷商标均为翱捷科技股份有限公司的商标。

本文档提及的其他所有商标名称、商标和注册商标均属其各自所有人的财产，特此声明。

防静电警告

静电放电（ESD）可能会损坏本产品。使用本产品进行操作时，须小心进行静电防护，避免静电损坏产品。

ASR Confidential

免责声明

翱捷科技股份有限公司对本文档内容不做任何形式的保证，并会对本文档内容或本文中介绍的产品进行不定期更新。

本文档仅作为使用指导，本文的所有内容不构成任何形式的担保。本文档中的信息如有变更，恕不另行通知。

本文档不负任何责任，包括使用本文档中的信息所产生的侵犯任何专有权行为的责任。

翱捷科技股份有限公司

地址：上海市浦东新区科苑路399号张江创新园10号楼9楼　邮编：201203

官网：http://www.asrmicro.com/

文档修订历史

日期	版本号	发布说明
2022.12	1.0.0	首次发布。

ASR Confidential

目录

1. BLE 接口和参数 1

2. BLE 接口 API 使用示例 2

2.1.1 BLE 初始化 2

2.1.2 开启 BLE 3

2.1.3 Advertising 的配置 6

2.1.4 Scanning 的配置 9

2.1.5 连接到其他 BLE 设备 13

ASR Confidential

插图

图 2-1 BLE 任务初始化函数 2
图 2-2 搜索 BLE 设备 8
图 2-3 BLE 扫描结果 11
图 2-4 连接上 BLE 设备 15

ASR Confidential

1. BLE 接口和参数

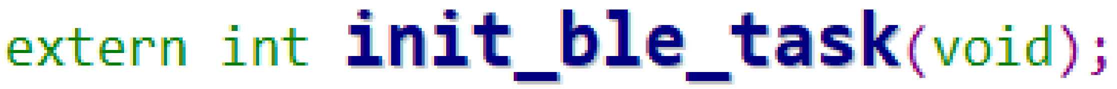
说明请参考文档：《ASR582X_BLE_API.chm》。

ASR Confidential

2. BLE 接口 API 使用示例

2.1 BLE 初始化

在使用 BLE 之前，需要初始化 BLE 任务，这个函数是封装在库里面，需要使用 extern 在外部显式声明，然后在合适的位置调用该函数：



```
extern int init_ble_task(void);
```

图 2-1 BLE 任务初始化函数

用户进行 BLE 开发时，需要实现 **sonata_ble_hook_t app_hook** 这个结构体中的函数，用户需要在合适的位置初始化结构体：其中 app_init 是应用层的 BLE 入口函数，用于 BLE 回调函数的注册以及启动 BLE 功能。

```
        sonata_ble_hook_t app_hook =
        {
            assert_err,
            assert_param,
            assert_warn,
            app_init,
            platform_reset,
            get_stack_usage,
            __wrap_printf,
            app_prf_api_init,
        #ifdef SONATA_RTOS_SUPPORT
            (void *)lega_rtos_init_semaphore,
            (void *)lega_rtos_get_semaphore,
            (void *)lega_rtos_set_semaphore,
        #endif
        };
```

ASR Confidential

2.2 开启 BLE

开启 BLE 功能前，需要先注册 GAP、GATT 和 Complete 执行结果等事件回调。后续 BLE 协议栈有事件需要上报时，都会通过对应的回调通知 APP。

GAP 和 GATT 对应的事件回调函数有很多，APP 可以根据应用场景选择进行注册，应用中不关心的事件可以不用注册。回调注册完成后，APP 再调用 BLE 开启接口和后续方法接口。

1. 用户实现 app_init 函数，参考代码如下：

```
void app_init(void)
{
    APP_TRC("APP: %s \r\n",_FUNCTION__);
    sonata_log_level_set(SONATA_LOG_VERBOSE);
    sonata_ble_register_gap_callback(&ble_gap_callbacks);//注册 GAP 回调
    sonata_ble_register_gatt_callback(&ble_gatt_callbacks);//注册 GATT 回调
    sonata_ble_register_complete_callback(&ble_complete_callbacks);//注册
    Complete 事件回调
    app_ble_on();//开启 BLE 模块
}
```

回调函数初始化如下：

```
static ble_gap_callback ble_gap_callbacks = {
    /*************** GAP Manager's callback ***************/
    //Must if use scan function, peer's information will show in this callback
    .gap_scan_result = app_gap_scan_result_callback,
    /*************** GAP Controller's callback ***************/
    //Optional,   used   for   get   peer   att   information   when   call
    sonata_ble_gap_get_peer_info()
    .gap_get_peer_info = app_gap_peer_info_callback,
    //Optional,   used   for   get   peer   att   information   when   call
    sonata_ble_gap_get_peer_info()
    .gap_get_peer_att_info = app_gap_peer_att_info_callback,
    //Optional, if peer device get local device's information, app can deal
with it in this callback
    .gap_peer_get_local_info = app_gap_peer_get_local_info_callback,
    //Optional
    .gap_disconnect_ind = app_gap_disconnect_ind_callback,
};
static ble_gatt_callback ble_gatt_callbacks = {
    //Must if use discovery all servcie function
    .gatt_disc_svc = app_gatt_disc_svc_callback,
    //Must if use discovery all characteristic function
    .gatt_disc_char = app_gatt_disc_char_callback,
    //Must if use discovery all description function
```

ASR Confidential

```
        .gatt_disc_char_desc = app_gatt_disc_desc_callback,
        //Must if use read attribute function
        .gatt_read = app_gatt_read_callback,
        //Optional, add this callback if app need to save changed mtu value
       .gatt_mtu_changed = app_gatt_mtu_changed_callback,
};
static ble_complete_callback ble_complete_callbacks = {
     //Must, app can do next operation in this callback
     .ble_complete_event = app_ble_complete_event_handler,
};
```

2. 用户实现app_ble_on()函数，参考代码如下：

```
static void app_ble_on()
{
    APP_TRC("APP: %s \r\n", __FUNCTION__);
    sonata_gap_set_dev_conf
    ig_cmd cmd = {0};
    cmd.role =
    SONATA_GAP_ROLE_ALL;
    cmd.gap_start_hdl = 0;
    cmd.gatt_start_hdl = 0;
    cmd.renew_dur = 0x0096;
    cmd.privacy_cfg = 0;
    cmd.pairing_mode = SONATA_GAP_PAIRING_SEC_CON | SONATA_GAP_PAIRING_LEGACY;
    cmd.att_cfg = 0x0080;
    cmd.max_mtu = 0x02A0;
    cmd.max_mps = 0x02A0;
    cmd.max_nb_lecb = 0x0A;
    cmd.hl_trans_dbg = false;
    uint16_t ret = sonata_ble_on(&cmd);//Next event:SONATA_GAP_CMP_BLE_ON
    if (ret != API_SUCCESS)
    {
        APP_TRC("APP: %s ERROR:%02X\r\n", __FUNCTION__, ret);
    }
}
```

该方法中参数含义见文档《ASR582X_BLE_API.chm》中**sonata_gap_set_dev_config_cmd**说明。

ASR Confidential

3. sonata_ble_on方法执行完成后，协议栈会调度执行complete事件回调，具体见下面代码中的 **SONATA_GAP_CMP_BLE_ON**事件：

```
static  uint16_t app_ble_complete_event_handler(sonata_ble_complete_type  opt_id,
                                                uint8_t status, uint16_t param, uint32_t dwparam)
{
     APP_TRC("APP_COMPLETE: %s opt_id=%04X,status=%02X,param=%04X,dwparam=%lu\r\n",
__FUNCTION__, opt_id, status, param, dwparam);
          switch (opt_id)
    {
    case SONATA_GAP_CMP_BLE_ON://0x0F01
        app_ble_config_legacy_advertising();
        break;
    default:
           break;
    }
     return CB_DONE;
}
```

更多的事件说明，见文档《ASR582X_BLE_API.chm》中**sonata_ble_complete_type**枚举。

ASR Confidential

2.3 Advertising 的配置

Advertising 的配置分为 3 步。需要先配置 advertising，然后设置 advertising 数据，再启动这个 advertising。

1. 配置 legacy advertising

```
static void app_ble_config_legacy_advertising()
{
    APP_TRC("APP: %s \r\n",_FUNCTION_);
    sonata_gap_directed_adv_create_param_t param = {0};
    param.disc_mode = SONATA_GAP_ADV_MODE_GEN_DISC;
    param.prop = SONATA_GAP_ADV_PROP_UNDIR_CONN_MASK;
    param.max_tx_pwr = 0xE2;
    param.filter_pol = ADV_ALLOW_SCAN_ANY_CON_ANY;
    // msg->adv_param.adv_param.peer_addr.addr.addr:00
    param.addr_type = 0;
    param.adv_intv_min = 64;
    param.adv_intv_max = 64;
    param.chnl_map = 0x07;
    param.phy = GAP_PHY_LE_1MBPS;
    uint16_t ret = sonata_ble_config_legacy_advertising(SONATA_GAP_STATIC_ADDR,
                   &param);
    //Next event:SONATA_GAP_CMP_ADVERTISING_CONFIG
    if (ret != API_SUCCESS)
    {
        APP_TRC("APP: %s ERROR:%02X\r\n",_FUNCTION_, ret);
    }
}
```

ASR Confidential

参数的说明请见文档《ASR582X_BLE_API.chm》中 **sonata_gap_directed_adv_create_param** 结构体。

2. 调用 **sonata_ble_config_legacy_advertising()**方法配置广播参数完成后，协议栈会调度执行 complete 事件回调，具体可见下面代码中的 **SONATA_GAP_CMP_ADVERTISING_CONFIG** 事件：

```
    case SONATA_GAP_CMP_ADVERTISING_CONFIG://0x0F02
        app_ble_set_adv_data();
        break;
```

进入该事件，表示 **sonata_ble_config_legacy_advertising()**方法执行完成，此时可以进行第二步，设置 advertising 数据：

```
static void app_ble_set_adv_data()

{

    APP_TRC("APP: %s \r\n", __FUNCTION__);

    uint8_t advData[] = {//Advertising data format

        8, GAP_AD_TYPE_COMPLETE_NAME, 'A', 'S', 'R', '-', 'B', 'L', 'E'

    };

    uint16_t ret = sonata_ble_set_advertising_data(sizeof(advData), advData);

    //Next

    event:SONATA_GAP_CMP_SET_ADV_DATA

    if (ret != API_SUCCESS){

        APP_TRC("APP: %s ERROR:%02X\r\n", __FUNCTION__, ret);

    }

}
```

3. 调用 **sonata_ble_set_advertising_data ()**方法设置广播数据完成后，协议栈会调度执行 complete 事件回调，具体见下面代码中的 **SONATA_GAP_CMP_SET_ADV_DATA** 事件：

```
      case SONATA_GAP_CMP_SET_ADV_DATA://0x01A9

          app_ble_start_advertising();

      break;
```

进入该事件，表示 **sonata_ble_set_advertising_data ()**方法执行完成，此时可以进行第三步，启动广播：

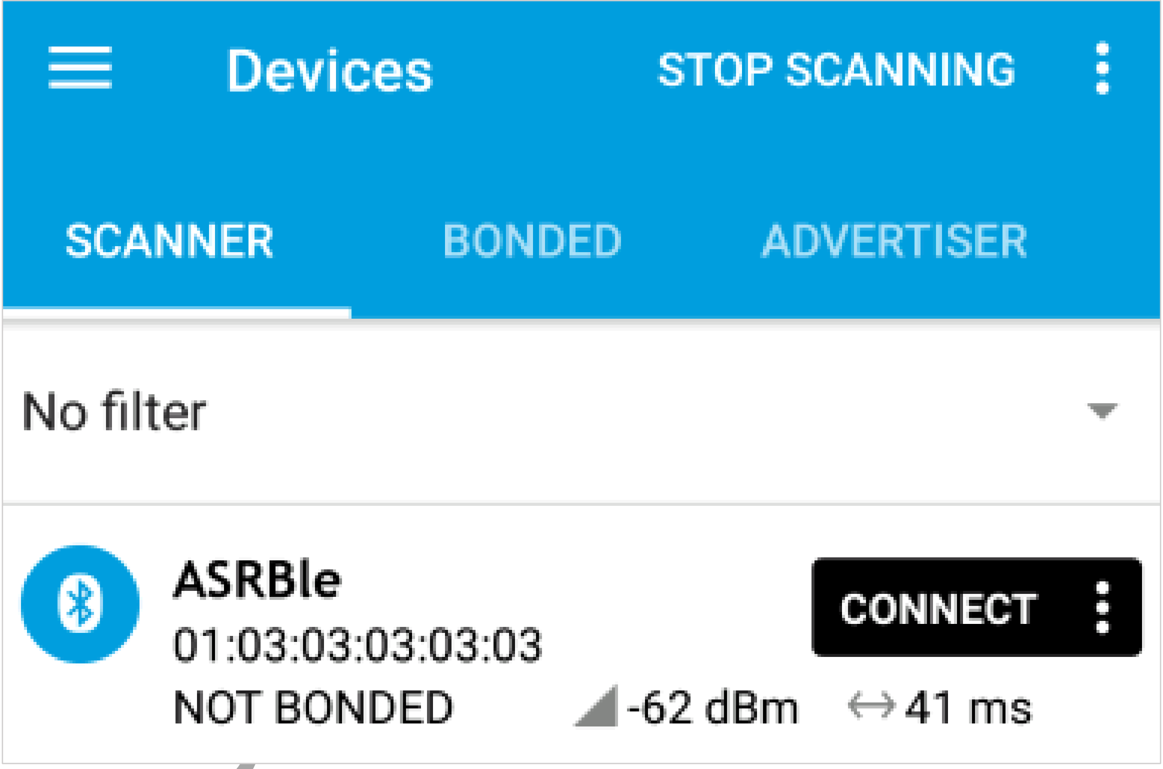
```
     static void app_ble_start_advertising()

     {

         APP_TRC("APP: %s \r\n", __FUNCTION__);

         uint16_t ret = sonata_ble_start_advertising(0, 0);

         //Next event:SONATA_GAP_CMP_ADVERTISING_START

         if (ret != API_SUCCESS) {

             APP_TRC("APP: %s ERROR:%02X\r\n", __FUNCTION__, ret);

         }

     }
```

ASR Confidential

4. **调用 sonata_ble_start_advertising()**方法开启广播完成后，协议栈会调度执行 complete 事件回调，具体见下面代码中的 **SONATA_GAP_CMP_ADVERTISING_START** 事件：

```
    case SONATA_GAP_CMP_ADVERTISING_START://0x0F06
     break;
```

5. 此时广播流程已执行完成，使用手机 BLE 应用就可以搜索到广播设备。



Devices　STOP SCANNING

SCANNER　BONDED　ADVERTISER

No filter

ASRBle
01:03:03:03:03:03
NOT BONDED　-62 dBm　↔41 ms

CONNECT

图 2-2 搜索 BLE 设备

2.4 Scanning 的配置

启动 Scanning 模式，需要在 ble on 的 complete 事件回调中配置并启动 Scanning 模式。步骤如下:

1. 调用 **app_ble_config_scanning()**

```
static uint16_t app_ble_complete_event_handler(sonata_ble_complete_type opt_id,
uint8_t
status, uint16_t param, uint32_t dwparam)

{

  APP_TRC("APP_COMPLETE: %sopt_id=%04X,status=%02X,param=%04X,dwparam=%lu\r\n",

       __FUNCTION__, opt_id, status, param, dwparam);

    switch (opt_id)

    {

    case SONATA_GAP_CMP_BLE_ON://0x0F01

        app_ble_config_scanning();

        break;

    default:

        break;

    }

    return CB_DONE;

}

static void app_ble_config_scanning()

{

    APP_TRC("APP: %s \r\n",_FUNCTION_);

    uint16_t ret = sonata_ble_config_scanning(SONATA_GAP_STATIC_ADDR);

    //Next event:SONATA_GAP_CMP_SCANNING_CONFIG

    if (ret != API_SUCCESS) {

        APP_TRC("APP: %s ERROR:%02X\r\n",_FUNCTION_, ret);

    }

}
```

ASR Confidential

2. **调用 sonata_ble_config_scanning** 方法配置扫描功能完成后，协议栈会调度执行 complete 事件回调，具体见下面代码中的 **SONATA_GAP_CMP_SCANNING_CONFIG** 事件。

   进入该事件表示 **sonata_ble_config_legacy_advertising()**方法执行完成，此时可以进行第二步，启动扫描:

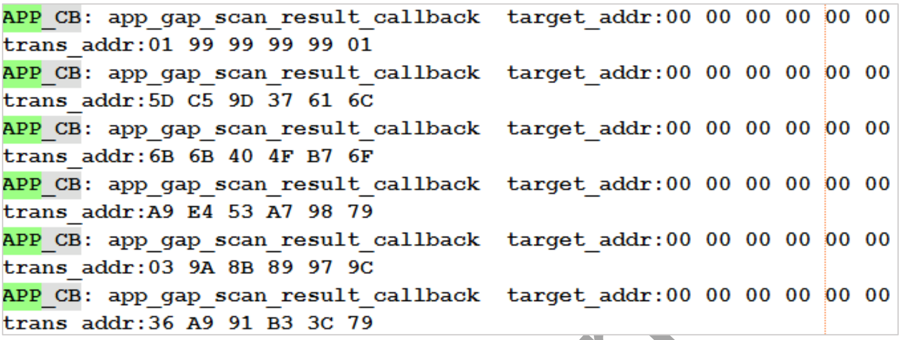
```
    case SONATA_GAP_CMP_SCANNING_CONFIG ://0x0F03
        app_ble_start_scanning();
        break;

    static void app_ble_start_scanning()
    {
          APP_TRC("APP: %s \r\n",_FUNCTION_);
           sonata_gap_scan_param_t param = {0};
           param.type = SONATA_GAP_SCAN_TYPE_GEN_DISC;
           // For continuous scan, use OBSERVER type, use duration to
           control scan timer.
           //   if   duration=0,   will   scan   for   ever   until
           sonata_ble_stop_scanning() called
           //param.type = SONATA_GAP_SCAN_TYPE_OBSERVER;
           param.prop = SONATA_GAP_SCAN_PROP_ACTIVE_1M_BIT
                     | SONATA_GAP_SCAN_PROP_PHY_1M_BIT;//0x05
           param.dup_filt_pol = SONATA_GAP_DUP_FILT_EN;
           param.scan_param_1m.scan_intv = 0x0140;
           param.scan_param_1m.scan_wd = 0x00A0;
           param.scan_param_coded.scan_intv = 0x0140;
           param.scan_param_coded.scan_wd = 0x00A0;
           param.duration = 0;
           param.period = 0;
           uint16_t ret = sonata_ble_start_scanning(&param);
           //Scan result will show in app_gap_scan_result_callback()
           if (ret != API_SUCCESS)
           {
               APP_TRC("APP: %s ERROR:%02X\r\n",_FUNCTION_, ret);
           }
     }
```

ASR Confidential

3. 调用 sonata_ble_start_scanning 方法配置扫描功能完成后，协议栈会调度执行 **complete** 事件回调，具体见下面代码中的 SONATA_GAP_CMP_SCANNING_START 事件。

```
    case SONATA_GAP_CMP_SCANNING_START://0x0F07
     break;
```

4. 此时 Scanning 已成功启动，扫描结果会通过 **gap_scan_result** 回调上报，应用层可通过回调函数的参数获取扫描结果。参考如下图：



```
APP_CB: app_gap_scan_result_callback  target_addr:00 00 00 00 00 00
trans_addr:01 99 99 99 99 01
APP_CB: app_gap_scan_result_callback  target_addr:00 00 00 00 00 00
trans_addr:5D C5 9D 37 61 6C
APP_CB: app_gap_scan_result_callback  target_addr:00 00 00 00 00 00
trans_addr:6B 6B 40 4F B7 6F
APP_CB: app_gap_scan_result_callback  target_addr:00 00 00 00 00 00
trans_addr:A9 E4 53 A7 98 79
APP_CB: app_gap_scan_result_callback  target_addr:00 00 00 00 00 00
trans_addr:03 9A 8B 89 97 9C
APP_CB: app_gap_scan_result_callback  target_addr:00 00 00 00 00 00
trans addr:36 A9 91 B3 3C 79
```

图 2-3 BLE 扫描结果

扫描结果回调示例代码如下：

```
static uint16_t app_gap_scan_result_callback(sonata_gap_ext_adv_report_ind_t
*result)
{
    APP_TRC("APP_CB: %s ",__FUNCTION__);
    APP_TRC("target_addr:");
    for (int i = 0; i < GAP_BD_ADDR_LEN; ++i)
    {
         APP_TRC("%02X ", result->target_addr.addr.addr[i]);
    }
     APP_TRC(" trans_addr:");
     for (int i = 0; i < GAP_BD_ADDR_LEN; ++i)
    {
         APP_TRC("%02X ", result->trans_addr.addr.addr[i]);
    }
     APP_TRC(" \r\n");
   #if DEMO_SCAN_AND_CONNECT
     //此处是判断是否搜到了对应设备，如果收到，则启动连接过程的事例代码
     if (memcmp(result->trans_addr.addr.addr, targetAddr1, GAP_BD_ADDR_LEN)
     == 0)
     {
         sonata_ble_stop_scanning();
           //If app adds gap_active_stopped() callback, SDK will callback
           when active stopped.App can restart or delete it.
```

ASR Confiden

```
         //If app not adds gap_active_stopped() callback,SDK will stop
         active and then delete it.
     }
  #endif
      return CB_DONE;
}
```

ASR Confidential

2.5 连接到其他 BLE 设备

在连接其他 BLE 设备之前，首先需要知道对端 BLE 设备的 MAC 地址和地址类型，这些信息可以在 2.4 章节中通过 scan 流程获得。发起连接步骤如下：

1. 在 ble on 的 Complete 事件（或者其他合适的事件中）调用 **app_ble_config_initiating()**：

```
static void app_ble_config_initiating()
{
    APP_TRC("APP: %s \r\n",_FUNCTION_);
    //Call api to config init
    uint16_t ret = sonata_ble_config_initiating(SONATA_GAP_STATIC_ADDR);
    //Next event:SONATA_GAP_CMP_INITIATING_CONFIG
    if (ret != API_SUCCESS)
    {
        APP_TRC("APP: %s ERROR:%02X\r\n",_FUNCTION_, ret);
    }
}
```

2. 调用 **sonata_ble_config_initiating** 方法配置发起连接功能完成后，协议栈会调度执行 complete 事件回调，具体见下面代码中的 **SONATA_GAP_CMP_INITIATING_CONFIG** 事件。配置连接功能完成后，调用 **sonata_ble_start_initiating** 方法发起建立连接。如下图：

```
case SONATA_GAP_CMP_INITIATING_CONFIG ://0x0F04
    APP_TRC("APP: %s start connect target1\r\n",_FUNCTION__);
    app_ble_start_initiating(targetAddr1);
    break;
static void app_ble_start_initiating(uint8_t *target)
{
    APP_TRC("APP: %s \r\n",_FUNCTION__);
    if (app_ble_check_address(target) == false) {
        APP_TRC("APP: %s, Target address is not right. Stop\r\n",_FUNCTION_);
        return;
    }
     sonata_gap_init_param_t param = {0};
     param.type = SONATA_GAP_INIT_TYPE_DIRECT_CONN_EST;
     param.prop = SONATA_GAP_INIT_PROP_1M_BIT | SONATA_GAP_INIT_PROP_2M_BIT
                 | SONATA_GAP_INIT_PROP_CODED_BIT;
     param.conn_to = 0;
     param.peer_addr.addr_type = SONATA_GAP_STATIC_ADDR;//Addr
     memcpy(param.peer_addr.addr.addr, target, GAP_BD_ADDR_LEN);
     if (param.prop & SONATA_GAP_INIT_PROP_1M_BIT)
```

ASR Confidential

```
      {
          APP_TRC("APP: %s (%02X)set SONATA_GAP_INIT_PROP_1M_BIT \r\n",_
FUNCTION__,
 param.prop);
          param.scan_param_1m.scan_intv = 0x0200;
          param.scan_param_1m.scan_wd = 0x0100;
          param.conn_param_1m.conn_intv_min=0x0028;
          param.conn_param_1m.conn_intv_max = 0x0028;
          param.conn_param_1m.conn_latency = 0;
          param.conn_param_1m.supervision_to = 0x02BC;
          param.conn_param_1m.ce_len_min = 0x0003;
          param.conn_param_1m.ce_len_max = 0x0003;
      }
      if (param.prop & SONATA_GAP_INIT_PROP_2M_BIT)
      {
           APP_TRC("APP: %s (%02X)set SONATA_GAP_INIT_PROP_2M_BIT \r\n",_
           FUNCTION__,
 param.prop);
          param.conn_param_2m.conn_intv_min = 0x0028;
          param.conn_param_2m.conn_intv_max = 0x0028;
          param.conn_param_2m.conn_latency = 0;
          param.conn_param_2m.supervision_to = 0x02BC;
          param.conn_param_2m.ce_len_min = 0x0003;
          param.conn_param_2m.ce_len_max = 0x0003;
      }
```

```
      if (param.prop & SONATA_GAP_INIT_PROP_CODED_BIT)
      {
          APP_TRC("APP: %s (%02X)set SONATA_GAP_INIT_PROP_CODED_BIT \r\n",_
          FUNCTION_,
 param.prop);
          param.scan_param_coded.scan_intv = 0x0200;
          param.scan_param_coded.scan_wd = 0x0100;
          param.conn_param_coded.conn_intv_min = 0x0028;
          param.conn_param_coded.conn_intv_max = 0x0028;
          param.conn_param_coded.conn_latency = 0;
          param.conn_param_coded.supervision_to = 0x02BC;
          param.conn_param_coded.ce_len_min = 0003;
          param.conn_param_coded.ce_len_max = 0003;
     }
     uint16_t ret = sonata_ble_start_initiating(&param);
     //Next event:If connected, SONATA_GAP_CMP_INITIATING_DELETE event will be
     received if (ret != API_SUCCESS)
     {
```

ASR Confidential

```
        APP_TRC("APP: %s ERROR:%02X\r\n", __FUNCTION__, ret);
    }
}
```

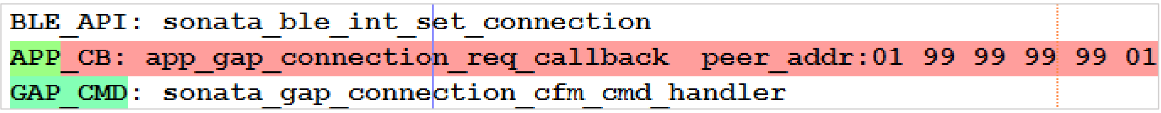
3. 调用 **sonata_ble_start_initiating** 方法配置扫描功能完成后，协议栈会调度执行 complete 事件回调，具体见下面代码中的 **SONATA_GAP_CMP_INITIATING_START** 事件。

```
    case SONATA_GAP_CMP_INITIATING_START://0x0F078
     break;
```

4. 此时表示协议栈已进入发起连接流程。连接建立成功后进入到**app_gap_connection_req_callback**回调。注意下面代码中黄色高亮代码。这个callback中，APP需要对Connection request返回Connect confirm消息（使用**sonata_gap_connection_cfm_cmd_handler** 方法）。如APP端不想对这个Request做进一步动作，则使用**return CB_REJECT**，SDK 收到这个返回值后，会自动发出默认的confirm消息，以确保建立简单连接。

```
static uint16_t app_gap_connection_req_callback(uint8_t conidx,
                                                 sonata_gap_connection_req_ind_t *req)
{
    APP_TRC("APP_CB: %s ", __FUNCTION__);
    APP_TRC("peer_addr:");
    for (int i = 0; i < GAP_BD_ADDR_LEN; ++i)
    {
        APP_TRC("%02X ", req->peer_addr.addr[i]);
    }
    APP_TRC(" \r\n");
    return CB_REJECT; //SDK will send connection confirm message
}
```

5. 连接上对方设备后，会自动停止当前的 Initiating 状态。应用层可以使用 **sonata_ble_gatt_disc_all_svc**、**sonata_ble_gatt_disc_all_characteristic** 等 API 来发现对端的 services/characteristic。连接成功 Log 如下图所示：



```
BLE_API: sonata_ble_int_set_connection
APP_CB: app_gap_connection_req_callback  peer_addr:01 99 99 99 99 01
GAP_CMD: sonata_gap_connection_cfm_cmd_handler
```

图 2-4 连接上 BLE 设备

ASR Confidential