App_att.c 中的 My_Attributes 列表

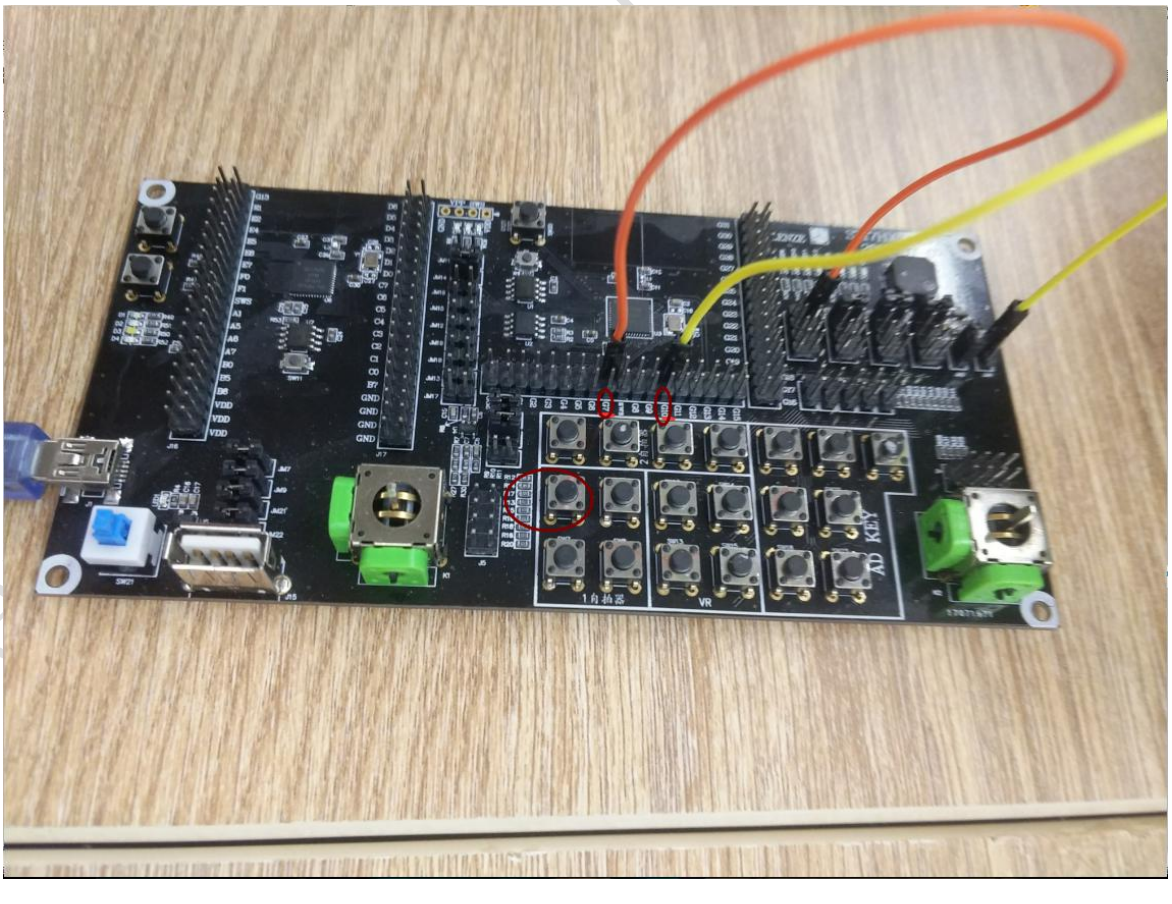
----设备接收来自主端指令 ，以及发送对应的按键信息

下面将介绍从端向主端的数据传输部分（即长按，短按，连按按键的命令发送 ）功能，以及主端向从端的数据传输部分（ 接收来自 alert UUID 的数据指令）的基础:服务（GATT: general attributes）。

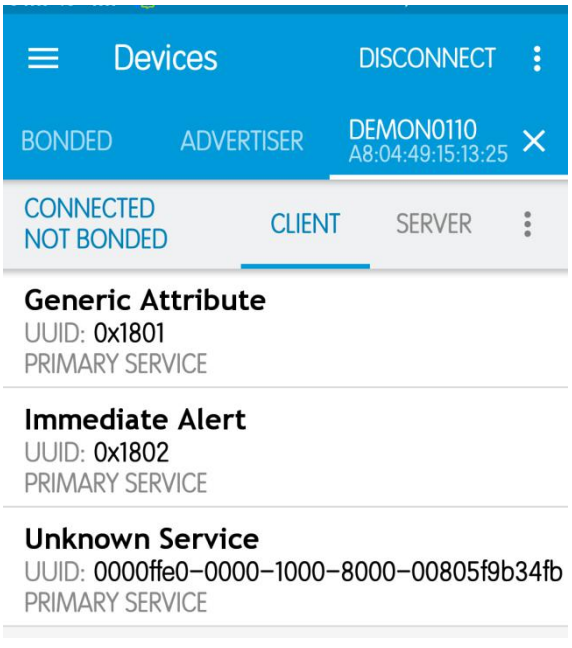
本文是基于“v0110”版本的 SDK -H26SDKCommon_GATT_GPIO_TEST 的使用介绍，读者在看本文档时，可以直接打开”NordicBLE 测试工具”的 apk，加以实验。

主端 主机发送 alert 写命令到从端 从端

主端 从端在 att_write_cb 写回调函数中，针对对应服务（handle=12）收到的数据进行处理，报警或停止报警 从端



一、生成一个简单的蓝牙服务列表



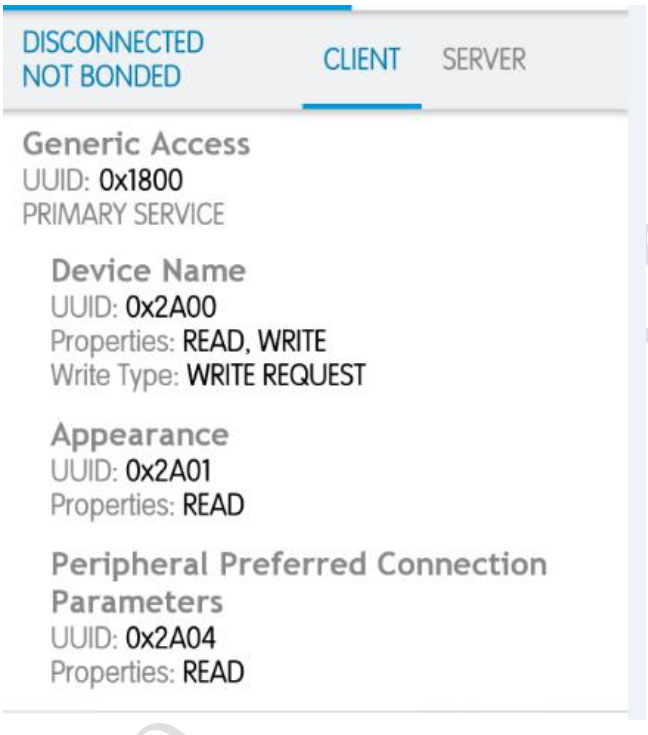
我们如果想在手机端获取上面这样的一个服务列表，首先应在函数中写入下面的一个常量服务列表

```
151
152 const attribute_t my_Attributes[] =
153 {
154 //DFSADF
155     {12,0,0,0,0}, //
156
157
158     {4,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
159     {0,2,1,(u8*)(&my_characteristicUUID),          (u8*)(&PROP_INDICATE)},
160     {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
161     {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
162 /*********************************************************************
163     5                   5  Immediate Alert  (Services)
164 *********************************************************************/
165     {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
166     {0,2,1,(u8*)(&my_characteristicUUID),        (u8*)(&immediateAlertLevel_prop)},
167     {0,2,1,(u8*)(&my_2a06_UIID),                    (u8*)(&immediateAlertLevel_value)},//7
168 /*********************************************************************
169      8                  28  Private  (Services)
170 *********************************************************************/
171     {4,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&FFE0_UUID)},
172     {0,2,1,(u8*)(&my_characteristicUUID),        (u8*)(&FFE1_prop)},
173     {0,2,1,(u8*)(&FFE1_charUUID),           (u8*)(&FFE1_value)},  //10
174     {0,2,sizeof(FFE1_value),(u8*)(&clientCharacterCfgUUID),    (u8*)(FFE1_value)},  //value
```

其中有三个主服务，对应关系如下图所示：

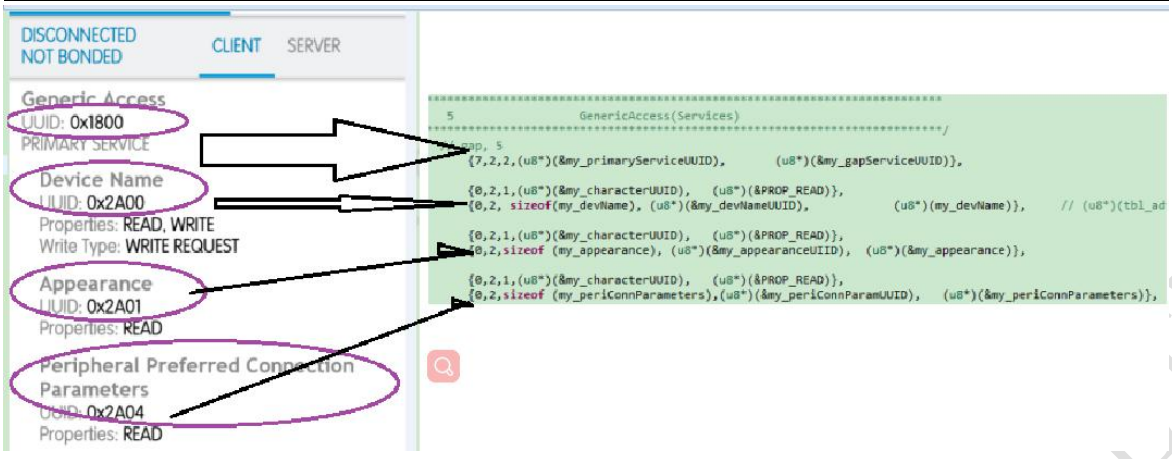

```
151
152 const attribute_t my_Attributes[] =
153 {
154 //DFSADF
155     {12,0,0,0,0}, //
156
157
158     {4,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
159     {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
160     {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
161     {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
162 /*******************************************************************
163     5                    5  Immediate Alert  (Services)
164 *******************************************************************/
165     {3,2,2,(u8*)(&my_primaryServiceUUID),    (u8*)(&immediateAlert_serviceUUID)},
166     {0,2,1,(u8*)(&my_characterUUID),         (u8*)(&immediateAlertLevel_prop)},
167     {0,2,1,(u8*)(&my_2a06_UIID),               (u8*)(&immediateAlertLevel_value)},//7
168 /*******************************************************************
169      8                  28  Private  (Services)
170 *******************************************************************/
171     {4,2,2,(u8*)(&my_primaryServiceUUID),    (u8*)(&FFE0_UUID)},
172     {0,2,1,(u8*)(&my_characterUUID),         (u8*)(&FFE1_prop)},
173     {0,2,1,(u8*)(&FFE1_charUUID),            (u8*)(&FFE1_value)},  //10
174     {0,2,sizeof(FFE1_value),(u8*)(&clientCharacterCfgUUID),    (u8*)(FFE1_value)},  //value
```

如果想加入下图的服务，我们可以加入后一张图片中所示的程序。

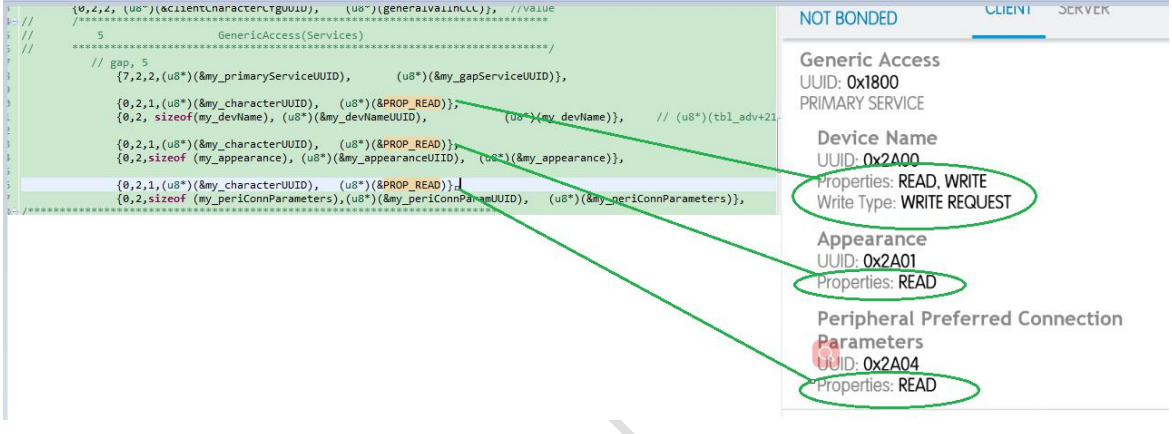



```
*******************************************************************
  5                  GenericAccess(Services)
*******************************************************************/
 // gap, 5
     {7,2,2,(u8*)(&my_primaryServiceUUID),       (u8*)(&my_gapServiceUUID)},

     {0,2,1,(u8*)(&my_characterUUID),   (u8*)(&PROP_READ)},
     {0,2, sizeof(my_devName), (u8*)(&my_devNameUUID),           (u8*)(my_devName)},     // (u8*)(tbl_ad

     {0,2,1,(u8*)(&my_characterUUID),   (u8*)(&PROP_READ)},
     {0,2,sizeof (my_appearance), (u8*)(&my_appearanceUIID),  (u8*)(&my_appearance)},

     {0,2,1,(u8*)(&my_characterUUID),   (u8*)(&PROP_READ)},
     {0,2,sizeof (my_periConnParameters),(u8*)(&my_periConnParamUUID),   (u8*)(&my_periConnParameters)},
```

下图中可以清晰地看出父子服务间的排列规律，下图中首个最大的箭头对应的主服务为父服务，后面的小箭头对应的是子服务。



同时，我们可以注意到每个子服务都有一个对应的读写属性，其中的排列关系如下所示：



为了充分地存储这些信息，我们需要自己定义一个如下所示的变量组，仔细观察，也是和 UUID 一一对应的变量。

```
78  //UUID:1800
79  const u16 my_gapServiceUUID = SERVICE_UUID_GENERIC_ACCESS;
80  u8 my_devName[8]={};
81  const u16 my_appearance = GAP_APPEARE_ROLE;//global
82  const gap_periConnectParams_t my_periConnParameters = {30, 60, 4, 1000};
83
```

由此，我们可以把想要让上位机（也可以是手机）知道的信息放在一个只读的服务中。把一个可接受上位机修改的信息，放在可读可写的服务中。把需要接受外界控制的信息，放在只可以写的服务中。

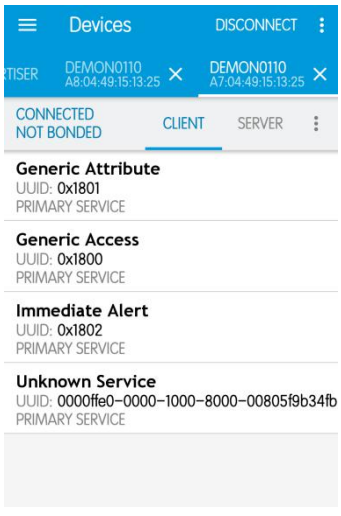
类似于上述的服务列表，我们以下图所示的新的服务列表为例，加入一个可以写的服务。

```
const attribute_t my_Attributes[] =
{
//DFSADF
    {17,0,0,0,0}, //

/*****************************************************************************
                        GenericAttribute(Services)
*****************************************************************************/
    {4,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
    {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
//  {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
/*****************************************************************************
                        GenericAccess(Services)
*****************************************************************************/
    // gap, 5
    {5,2,2,(u8*)(&my_primaryServiceUUID),      (u8*)(&my_gapServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_appearance), (u8*)(&my_appearanceUIID),  (u8*)(&my_appearance)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_periConnParameters),(u8*)(&my_periConnParamUUID),   (u8*)(&my_periConnParameters)},
    /*****************************************************************************
            10  Immediate Alert  (Services)
    *****************************************************************************/
        {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&immediateAlertLevel_prop)},
        {0,2,1,(u8*)(&alertLevel_charUUID),         (u8*)(&immediateAlertLevel_value)},  // handle=7
    //  {0,2,2, (u8*)(&clientCharacterCfgUUID),    (u8*)(immediateAlertLevel_valueInCCC)},  //value
    /*****************************************************************************
            13   Private  (Services)
    *****************************************************************************/
        {4,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&FFE0_UUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&FFE1_prop)},
        {0,2,sizeof(FFE1_value),(u8*)(&FFE1_charUUID),      (u8*)(&FFE1_value)},  // handle= 11 value
        {0,2,sizeof(FFE2_value),    (u8*)(&clientCharacterCfgUUID),    (u8*)(FFE2_value)},
/*****************************************************************************
```

由于第一组服务的 att_Num 申明 handle 为 4 ，而实际的 handle 值是 3，手机端连接后判断设备第一组服务不全，没有显示。我们将 att_Num 变为 3，程序变为下面

```
const attribute_t my_Attributes[] =
{
//DFSADF
    {17,0,0,0,0}, //

/*****************************************************************************
                        GenericAttribute(Services)
*****************************************************************************/
    {3,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
    {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
//  {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
/*****************************************************************************
```

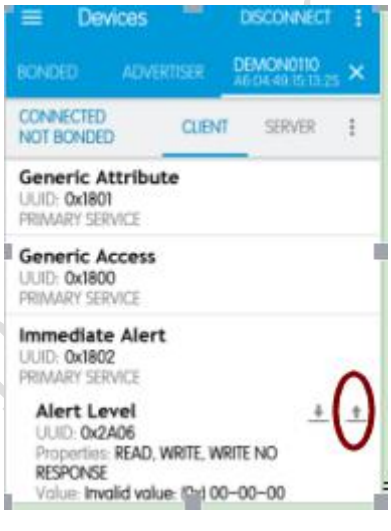
即在手机端获得了与我们在表中所定义的相似的 4 组服务，如下所示：



二、完成一次真正的蓝牙控制

```
/*********************************************************************
        10  Immediate Alert  (Services)
*********************************************************************/
    {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&immediateAlertLevel_prop)},
    {0,2,3,(u8*)(&alertLevel_charUUID),         (u8*)(&immediateAlertLevel_value)},  // handle=7
//  {0,2,2,(u8*)(&clientCharacterCfgUUID),    (u8*)(immediateAlertLevel_valueInCCC)},  //value
/*********************************************************************
        13  Private  (Services)
```

连接设备之后，点击下图所示的 Alert Level 右侧圈住的箭头，在弹出的数据框中输入‘1’或者‘2’，然后 send 到 DEMON0110 设备中。



我们可以看到最下面一行对 value 的描述变为 value：0x01,或 0x02,设备读取后，改变 LED 的状态。

三、服务列表定义和含义

A)、attribute 的格式

```
typedef struct attribute
{
#if(UUID_SUPPORT_TYPE==ONLY_UUID16)
      u8  attNum;
    //  u8  uuidLen;
      u8  attrLen;
    //  u8  attrMaxLen;
      u16 uuid;
      u8* pAttrValue;
#else
      u8  attNum;
      u8  uuidLen;
      u8  attrLen;
    //  u8  attrMaxLen;
      u8* uuid;
      u8* pAttrValue;
#endif
} attribute_t;
```

这是在 my_attributes[]列表中加入具有 read 和 notify 属性的条目：如图所示，在橙色箭头所指向的区域的 FFE0_UUID 服务下，加入 FFE1_prop（具有 read 和 notify 属性），FFE1_value 等属性类型以及数据。首先请注意，Attribute Table 的定义前面加了 const：

const attribute_t my_Attributes[] = { ... };

const 关键字会让编译器将这个数组的数据最终都存储到 flash，这个 table 里所有内容是只读的，不能改写。

1)attNum

attNum 有两个作用。

a) attNum 第一个作用是表示当前 Attribute Table 中有效 Attribute 数目，即 Attribute Handle 的最大值，该数目只在 Attribute Table 数组的第 0 项无效 Attribute 中使用：

{14,0,0,0},

attNum = 14 表示当前 Attribute Table 中共有 14 个 Attribute。

在 BLE 里，Attribute Handle 值从 0x0001 开始，往后加一递增，而数组的下标从 0 开始，在 Attribute Table 里加上上面这个虚拟的 Attribute，正好使得后面每个 Attribute 在数据里的下标号等于其 Attribute Handle 的值。当定义好了 Attribute Table 后，在 Attribute Table 中数 Attribute 在当前数组中的下标号，就能知道该 Attribute 当前的 Attribute Handle 值。如基于下面的 table，如果用户需要 notify FFE1 相关的数据，那么其所使用的 handle 就是 15。

```
const attribute_t my_Attributes[] =
{
//DFSADF
    {17,0,0,0,0}, //

/*****************************************************************************
                        GenericAttribute(Services)
*****************************************************************************/
    {4,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
    {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
    {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
/*****************************************************************************
                        GenericAccess(Services)
*****************************************************************************/
    // gap, 5
    {5,2,2,(u8*)(&my_primaryServiceUUID),      (u8*)(&my_gapServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_appearance), (u8*)(&my_appearanceUIID),  (u8*)(&my_appearance)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_periConnParameters),(u8*)(&my_periConnParamUUID),   (u8*)(&my_periConnParameters)},
    /*****************************************************************************
            10  Immediate Alert  (Services)
    *****************************************************************************/
        {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&immediateAlertLevel_prop)},
        {0,2,1,(u8*)(&alertLevel_charUUID),         (u8*)(&immediateAlertLevel_value)},  // handle=7
    //  {0,2,2, (u8*)(&clientCharacterCfgUUID),    (u8*)(immediateAlertLevel_valueInCCC)},  //value
    /*****************************************************************************
            13   Private  (Services)
    *****************************************************************************/
        {4,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&FFE0_UUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&FFE1_prop)},
        {0,2,sizeof(FFE1_value),(u8*)(&FFE1_charUUID),      (u8*)(&FFE1_value)},
        {0,2,sizeof(FFE2_value),    (u8*)(&clientCharacterCfgUUID),    (u8*)(FFE2_value)},
/*****************************************************************************
                            OTA (Services)
*****************************************************************************/
```

handle = 15

b) attNum 第二个作用是用于表示当前 service 类型。

如上面截图所示，Attribute Handle 5 - Attribute Handle 9 这 5 个 Attribute 是属于 gap service 的描述，它们属于一大类，所以这一组 Attribute 中首个 Attribute 的 UUID 为 0x2800(GATT_UUID_PRIMARY_SERVICE) ， 且 attribute value 为 0x1800(SERVICE_UUID_GENERIC_ACCESS)用于指明当前的 service 类型，那么它的 attNum 写为 5 后，BLE SDK 底层就知道从 Attribute Handle 1 开始的连续 5 个 Attribute 为 gap service。

同样，上图中的 immediateAlert service 的首个 Attribute 的 attNum 设为 3 后，底层会知道从这个 Attribute Handle 开始的连续 3 个 Attribute 都属于 immediateAlert service。

{3,2,2,(u8*)(&my_primaryServiceUUID), (u8*)(&immediateAlert_serviceUUID)} ,

除了第 0 项 Attribute 和每一类 service 首个用于说明当前 service 类型的 Attribute 外，其他所有的 Attribute 的 attNum 的值都必须设为 0。

B)、attribute 的必要要素

在 my_attributes[]列表中加入具有 read 和 notify 属性的条目，app 端的蓝牙连接设备并获取到该服务后，就能够对之相应的服务 FFE1_UUID 进行读和接收来自设备的通知。如下介绍 my_attributes 的第一张图所示，在选中的行所对应的区域的 immediateAlert_serviceUUID 服务下，加入 immediateAlert_prop（具有 read,write 和 write_without_rsp 属性 ），immediateAlertLevel_value 等属性类型以及数据。

```
static const u16 FFE0_UUID = 0xffe0;
static const u16 FFE1_charUUID = 0xffe1;
static const u8  FFE1_prop = CHAR_PROP_READ | CHAR_PROP_NOTIFY;
//
```

在 my_attributes[]列表中加入具有 read，write 属性的条目，app 端的蓝牙连接设备并获取到该服务后，就能够对与之相应的 immediateAlert_serviceUUID 服务进行读和写设备的操作。如下图所示，在选中的行所对应的区域的 immediateAlert_serviceUUID 服务下，加入 immediateAlert_prop（具有 read,write 和 write_without_rsp 属性，对应定义的格式如下图！），immediateAlertLevel_value 等属性类型以及数据。

```
static const u8 immediateAlertLevel_prop = CHAR_PROP_READ | CHAR_PROP_WRITE | CHAR_PROP_WRITE_WITHOUT_RSP;//
u8 immediateAlertLevel_value = 0;
u8 immediateAlertLevel_valueInCCC[2];
```

C)、attribute 的附加要素

app_att.c | blueLight.h | gatt_server.h | gatt_uuid.h

```
#pragma once

/** @addtogroup  GATT_Common_Module GATT UUID

/** @addtogroup GATT_UUID GATT 16 bit UUID definition
 * @{
 */
#define GATT_UUID_PRIMARY_SERVICE        0x2800   //!< Primary Service
#define GATT_UUID_SECONDARY_SERVICE      0x2801   //!< Secondary Service
#define GATT_UUID_INCLUDE                0x2802   //!< Include
#define GATT_UUID_CHARACTER              0x2803   //!< Characteristic
#define GATT_UUID_CHAR_EXT_PROPS         0x2900   //!< Characteristic Extended Properties
#define GATT_UUID_CHAR_USER_DESC         0x2901   //!< Characteristic User Description
#define GATT_UUID_CLIENT_CHAR_CFG        0x2902   //!< Client Characteristic Configuration
#define GATT_UUID_SERVER_CHAR_CFG        0x2903   //!< Server Characteristic Configuration
#define GATT_UUID_CHAR_PRESENT_FORMAT    0x2904   //!< Characteristic Present Format
#define GATT_UUID_CHAR_AGG_FORMAT        0x2905   //!< Characteristic Aggregate Format
#define GATT_UUID_VALID_RANGE            0x2906   //!< Valid Range
#define GATT_UUID_EXT_REPORT_REF         0x2907   //!< External Report Reference
#define GATT_UUID_REPORT_REF             0x2908   //!< Report Reference

#define GATT_UUID_DEVICE_NAME            0x2a00   //!< Report Reference
#define GATT_UUID_BATTERY_LEVEL          0x2A19
#define GATT_UUID_FIRMWARE_VER           0x2a26   //!<Firmware Version
#define GATT_UUID_SERVICE_CHANGE         0x2a05
/** @} end of group GATT UUID */


/** @addtogroup  GATT_UUID_Variables GATT UUID variables
 *  @{
 */
```

在上图中常用的是第一个 Primary Service, 后面的所有 uuid 都是作为可选的服务，根据实际项目需要添加，每一条服务的含义都在箭头起始位置一一列举。

四、my_attributes 列表的规定以及一些理解的技巧

```
const attribute_t my_Attributes[] =
{
//DFSADF
    {17,0,0,0,0}, //

/**********************************************************************
                    GenericAttribute(Services)
**********************************************************************/
    {4,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
    {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
    {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
/**********************************************************************
                    GenericAccess(Services)
**********************************************************************/
    // gap, 5
    {5,2,2,(u8*)(&my_primaryServiceUUID),      (u8*)(&my_gapServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_appearance), (u8*)(&my_appearanceUUID),  (u8*)(&my_appearance)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_periConnParameters),(u8*)(&my_periConnParamUUID),   (u8*)(&my_periConnParameters)},
    /**********************************************************************
            10  Immediate Alert  (Services)
    **********************************************************************/
        {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&immediateAlertLevel_prop)},
        {0,2,1,(u8*)(&alertLevel_charUUID),         (u8*)(&immediateAlertLevel_value)},  // handle=7
    //  {0,2,2, (u8*)(&clientCharacterCfgUUID),    (u8*)(immediateAlertLevel_valueInCCC)},  //value
    /**********************************************************************
            13   Private  (Services)
    **********************************************************************/
        {4,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&FFE0_UUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&FFE1_prop)},
        {0,2,sizeof(FFE1_value),(u8*)(&FFE1_charUUID),      (u8*)(&FFE1_value)},
        {0,2,sizeof(FFE2_value),    (u8*)(&clientCharacterCfgUUID),    (u8*)(FFE2_value)},
/**********************************************************************
                        OTA (Services)
**********************************************************************/
```

handle = 15

Attribute Protocol 定义了两种角色：server 和 client。17H26 BLE SDK 中，slave 主要定义的是 server 端.对应的 Android、iOS 设备是 client。Server 端与 client 端的关系是，server 的一组的 Attribute 可被 client 访问。Client 向 server 的某个具有 write 属性的 uuid_value 写入数据，会在 att_write_cb（）中的 rf_packet_att_write_t 结构体的 value 值中得以体现,只要判定对应 handle 值和其中的 value 就能实现主到从数据的通信了。

```
typedef struct{
    u32 dma_len;            //won't be a fixed number as previous, should adjust
    u8  type;               //RFU(3)_MD(1)_SN(1)_NESN(1)-LLID(2)
    u8  rf_len;             //LEN(5)_RFU(3)
    u16 l2capLen;
    u16 chanId;
    u8  opcode;
    u8 handle;
    u8 handle1;
    u8 value;
}rf_packet_att_write_t;
```

也可以是value[23],只要不超过MTU的范围

Server 向 client 发送数据则需要设置一个具有 read 以及 notify 属性的服务，并在主函数中调用 blt_push_notify()函数将 att_value 发送到对应该 handle 的主端 即可。

五、 Attribute 表不规范导致的问题，以及修改 my_Attributes 列表之后要注意的事项

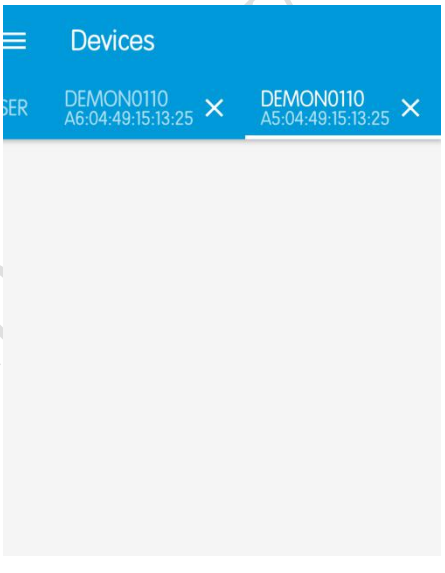
第一种问题：handle = 0 ，且 att_Num 小于实际 handle 数目

我们把第 0 条 handle 所对应的数据改为下面图示的，并修改 tbl_mac 中的 MAC 地址值，不难发现在手机端搜不到对应的设备。

```
const attribute_t my_Attributes[] =
{
//DFSADF
    {15,0,0,0,0}, //

/****************************************************************************
                        GenericAttribute(Services)
****************************************************************************/
    {3,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
    {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
//  {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
/****************************************************************************
```

第二种问题： handle!=0,且对应 service_value 的那行描述中 att_length 与实际不符
B)att_length 小于实际的长度

```
    {0,2,sizeof (my_periConnParameters),(u8*)(&my_periConnParamUUID),   (u8*)(&my_periConnParameters)},
    /****************************************************************************
           10  Immediate Alert  (Services)
    ****************************************************************************/
        {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
        {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&immediateAlertLevel_prop)},
        {0,2,0,(u8*)(&alertLevel_charUUID),       (u8*)(&immediateAlertLevel_value)},  // handle=7
    //  {0,2,2, (u8*)(&clientCharacterCfgUUID),    (u8*)(immediateAlertLevel_valueInCCC)},  //value
    /****************************************************************************
```



此时会引起连接后的获取服务异常：

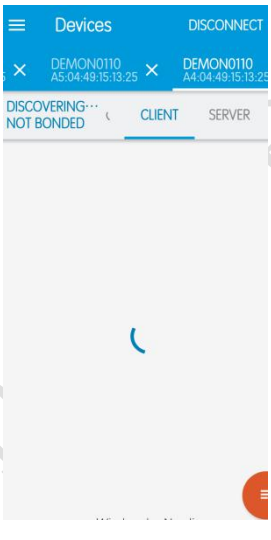
第三种问题： handle！=0，且 my_attributes[]列表中的结构与系统定义的 attribute_t 结构体不相符。

在这种情况下，虽然也能编译出 xx.bin 文件,搜到设备广播，但是手机端却会一直卡在获取服务阶段，如下所示。

```
/*********************************************************************
                        GenericAttribute(Services)
*********************************************************************/
    {3,2,2,(u8*)(&my_primaryServiceUUID),          (u8*)(&my_gattServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),               (u8*)(&PROP_INDICATE)},
    {0,2,4,(u8*)(&my_serviceChangeUUID),           (u8*)(serviceChangeVal)},
//  {0,2,2,(u8*)(&clientCharacterCfgUUID),         (u8*)(serviceChangeCCC)},
/*********************************************************************
                        GenericAccess(Services)
*********************************************************************/
    // gap, 5
    {5,2,2,(u8*)(&my_primaryServiceUUID),      (u8*)(&my_gapServiceUUID)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_appearance), (u8*)(&my_appearanceUIID),  (u8*)(&my_appearance)},
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ)},
    {0,2,sizeof (my_periConnParameters),(u8*)(&my_periConnParamUUID),   (u8*)(&my_periConnParameter
    /*********************************************************************
            10  Immediate Alert  (Services)
    *********************************************************************/
        {3,2,0x2800,    (u8*)(&immediateAlert_serviceUUID)},
        {0,1,0x2803,        (u8*)(&immediateAlertLevel_prop)},
        {0,1,0x2A06,    (u8*)(&immediateAlertLevel_value)},  // handle=7
//      {3,2,2,(u8*)(&my_primaryServiceUUID),   (u8*)(&immediateAlert_serviceUUID)},
//      {0,2,1,(u8*)(&my_characterUUID),        (u8*)(&immediateAlertLevel_prop)},
//      {0,2,0,(u8*)(&alertLevel_charUUID),         (u8*)(&immediateAlertLevel_value)},  // handle=
```

Problems | Console | Call Hierarchy | Tasks | Properties | Progress

```
CDT Build Console [H26SDKCommon_GATT_GPIO_TEST]
tc32-elf-objcopy -O binary 17H26_VR_OTP.elf  "17H26_VR_OTP.bin"
Invoking: Print Size
tc32-elf-size -t 17H26_VR_OTP.elf
   text    data     bss     dec     hex filename
  11028     348     575   11951    2eaf 17H26_VR_OTP.elf
  11028     348     575   11951    2eaf (TOTALS)
Finished building: sizedummy

Finished building: 17H26_VR_OTP.bin

Finished building: 17H26_VR_OTP.lst


17:50:23 Build Finished (took 27s.80ms)
```



综上所述的几个问题是比较典型的，那么在实际操作中可能还会有很多意想不到的问题和现象，但基本可以确定是这几个问题之中的一个或多个所造成的异常。希望在做这部分工作的时候，用户要多多注意下细节。

其他注意事项

每次更改 my_Attributes 表之后,都**需要修改设备对应的 mac 地址**（即 Ui.c 中的 tbl_mac[]对应的数），修改的目的是方便发现更改后 att 表的问题。（手机的 APP 中或蓝牙会根据设备 mac

地址或名称来缓存设备的服务列表，重新连接过程可能不会重新获取设备服务，因此，需要我们更新设备 mac 告诉手机端，必须重新获取服务）。

解决 att 故障的方法很简单，只要保证数据范围在规定 MTU 之内，且格式，长度与我们上图中对 att 结构体的定义一一对应，att 表的结构体定义一般是不予修改的，其中的 u8* uuid; u8* pAttrValue;作为指针可以指向一个任意长度的位置进而生成任意 16 位，24 位，128 位长度不限的 UUID 服务编号，已经为我们自定义的设计提供了很大方便和灵活性。

```
const u8 a000_UUID[16]={0x11,0x11,0x22,0x22,0x33,0x33,0x44,0x44,0x55,0x55,0x66,0x66,0x00,0xa0,0x00,0x00};
u8 a001_value[9]={0};
static const u8  a001_prop = CHAR_PROP_READ | CHAR_PROP_WRITE ;
const u8 a001_charUUID[16]={0x11,0x11,0x22,0x22,0x33,0x33,0x44,0x44,0x55,0x55,0x66,0x66,0x01,0xa0,0x00,0x00};
//const u16 a001_charUUID=0xa001;

//const u16 a002_charUUID=0xa002;
const u8 a002_charUUID[16]={0x11,0x11,0x22,0x22,0x33,0x33,0x44,0x44,0x55,0x55,0x66,0x66,0x02,0xa0,0x00,0x00};
u8 a002_value[9];
static const u8  a002_prop = CHAR_PROP_READ | CHAR_PROP_NOTIFY;

u8 a003_value[9]={0};
static const u8  a003_prop = CHAR_PROP_READ |CHAR_PROP_NOTIFY;
const u8 a003_charUUID[16]={0x11,0x11,0x22,0x22,0x33,0x33,0x44,0x44,0x55,0x55,0x66,0x66,0x03,0xa0,0x00,0x00};
```