ST17H26 ble_0109_sdk 开发说明 V1.02

App_att.c 用户文件模板

由于在<Lenze 17H26 BLE SDK User Guide_v1.01>中已经明确地说明了一个标准 SDK 的软件架构和基本文件，本节中主要介绍用户怎样在现有的 SDK 之上，合理安排程序，生成一个符合蓝牙 4.0 协议标准的服务属性。

```
#include "../../proj/tl_common.h"
#include "../../proj_lib/blt_ll/blt_ll.h"
#include "ui.h"


#define FW_VERSION_ID1 0x01
#define FW_VERSION_ID2 0x09


typedef struct
{
  /** Minimum value for the connection event (interval. 0x0006 - 0x0C80 * 1.25 ms)
*/
  u16 intervalMin;
  /** Maximum value for the connection event (interval. 0x0006 - 0x0C80 * 1.25 ms)
*/
  u16 intervalMax;
  /** Number of LL latency connection events (0x0000 - 0x03e8) */
  u16 latency;
  /** Connection Timeout (0x000A - 0x0C80 * 10 ms) */
  u16 timeout;
} gap_periConnectParams_t;

const u16 clientCharacterCfgUUID = GATT_UUID_CLIENT_CHAR_CFG;

//const u16 extReportRefUUID = GATT_UUID_EXT_REPORT_REF;

const u16 reportRefUUID = GATT_UUID_REPORT_REF;

//const u16 characterPresentFormatUUID = GATT_UUID_CHAR_PRESENT_FORMAT;

const u16 my_primaryServiceUUID = GATT_UUID_PRIMARY_SERVICE;

const u16 my_characterUUID = GATT_UUID_CHARACTER;

const u16 my_devServiceUUID = SERVICE_UUID_DEVICE_INFORMATION;

const u16 my_PnPUUID = CHARACTERISTIC_UUID_PNP_ID;
```

```
const u16 my_devNameUUID = GATT_UUID_DEVICE_NAME;
const u16 my_serviceChangeUUID = GATT_UUID_SERVICE_CHANGE;
const u16 my_appearanceUIID = 0x2a01;
const u16 my_periConnParamUUID = 0x2a04;
const u16 my_gattServiceUUID = SERVICE_UUID_GENERIC_ATTRIBUTE; //0x1801

extern u8 tbl_adv[];
////////////////////////////定义设备的服务属性//////////////////////////
const u8 PROP_READ = CHAR_PROP_READ;

const u8 PROP_WRITE = CHAR_PROP_WRITE;

const u8 PROP_INDICATE = CHAR_PROP_INDICATE;

const u8 PROP_WRITE_NORSP = CHAR_PROP_WRITE_WITHOUT_RSP;

const u8 PROP_READ_NOTIFY = CHAR_PROP_READ | CHAR_PROP_NOTIFY;

const u8 PROP_READ_WRITE_NORSP = CHAR_PROP_READ | CHAR_PROP_WRITE_WITHOUT_RSP;

const u8 PROP_READ_WRITE_WRITENORSP = CHAR_PROP_READ | CHAR_PROP_WRITE |
CHAR_PROP_WRITE_WITHOUT_RSP;

const u8 PROP_READ_WRITE = CHAR_PROP_READ|CHAR_PROP_WRITE;

const u8 PROP_READ_WRITE_NORSP_NOTIFY = CHAR_PROP_READ |
CHAR_PROP_WRITE_WITHOUT_RSP|CHAR_PROP_NOTIFY;
////////////////////////////定义设备的服务属性 结束//////////////////////////

const u8    my_PnPtrs [] = {0x02, 0x12, 0x34, 0x56, 0x78, FW_VERSION_ID2,
FW_VERSION_ID1};

u16 serviceChangeVal[4] = {0};
static u8 serviceChangeCCC[2]={0,0};
//////////////// //定义设备的服务属性 UUID,对应读写特性,以及特征值//////////////////

const u16 my_gapServiceUUID = SERVICE_UUID_GENERIC_ACCESS; //服务属性 UUID
const u16 my_appearance = GAP_APPEARE_ROLE;//global //
const gap_periConnectParams_t my_periConnParameters = {30, 60, 4, 1000};
//服务属性 UUID 对应的值
//////////////////////////////////////////////////////////////////////////
            HID 服务属性 UUID，对应读写特性，以及特征值
    //////////////////////////////////////////////////////////////////////////

const u16 my_hidServiceUUID                 = SERVICE_UUID_HUMAN_INTERFACE_DEVICE;
```

```
const u16 my_SppDataServer2ClientUUID      = SPP_DATA_SERVER2CLIENT;
const u16 my_SppDataClient2ServiceUUID     = AUDIO_UUID_SERVICE ;

const u16 hidServiceUUID          = SERVICE_UUID_HUMAN_INTERFACE_DEVICE;
const u16 hidProtocolModeUUID     = CHARACTERISTIC_UUID_HID_PROTOCOL_MODE;
const u16 hidReportUUID           = CHARACTERISTIC_UUID_HID_REPORT;
const u16 hidReportMapUUID        = CHARACTERISTIC_UUID_HID_REPORT_MAP;
const u16 hidbootKeyInReportUUID  = CHARACTERISTIC_UUID_HID_BOOT_KEY_INPUT;
const u16 hidbootKeyOutReportUUID = CHARACTERISTIC_UUID_HID_BOOT_KEY_OUTPUT;
const u16 hidbootMouseInReportUUID = CHARACTERISTIC_UUID_HID_BOOT_MOUSE_INPUT;
const u16 hidinformationUUID      = CHARACTERISTIC_UUID_HID_INFORMATION;
const u16 hidCtrlPointUUID        = CHARACTERISTIC_UUID_HID_CONTROL_POINT;
const u16 hidIncludeUUID          = GATT_UUID_INCLUDE;

static u8 protocolMode = DFLT_HID_PROTOCOL_MODE;

const u16 my_batServiceUUID               = SERVICE_UUID_BATTERY;
const u16 my_batCharUUID                  = CHARACTERISTIC_UUID_BATTERY_LEVEL;
u8 my_batVal                        = {100};
static const u16 FFE0_UUID = 0xffe0;
static const u16 FFE1_charUUID = 0xffe1;
static const u8  FFE1_prop = CHAR_PROP_READ | CHAR_PROP_NOTIFY;

static const u16 FFE2_charUUID = 0xffe2;
static const u8  FFE2_prop = CHAR_PROP_READ | CHAR_PROP_WRITE;

static const u16 FFE3_charUUID = 0xffe3;
static const u8  FFE3_prop = CHAR_PROP_READ;

static const u16 FFE4_charUUID = 0xffe4;
static const u8  FFE4_prop = CHAR_PROP_READ |CHAR_PROP_WRITE;

static const u16 FFE5_charUUID = 0xffe5;
static const u8  FFE5_prop = CHAR_PROP_READ |CHAR_PROP_WRITE;

static const u16 FFE6_charUUID = 0xffe6;
static const u8  FFE6_prop = CHAR_PROP_READ;

static const u16 FFE7_charUUID = 0xffe7;
static const u8  FFE7_prop = CHAR_PROP_READ |CHAR_PROP_WRITE;

u8 FFE1_value[1] = {0x00};
u8 FFE2_value[1] = {0x01};
u8 FFE3_value[6] = {0x00};
u8 FFE4_value[6] = {0x00};
u8 FFE5_value[1] = {0x00};
```

```
u8 FFE6_value[8] = {0x00}; ///读取当前状态
u8 FFE7_value[6] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};



static const u16 TxPower_serviceUUID  = SERVICE_UUID_TX_POWER;
static const u16 TxPower_charUUID = CHARACTERISTIC_UUID_TX_POWER_LEVEL;
static const u8 TxPower_prop =CHAR_PROP_READ;
u8 Txpower_value = 7;//TX_POWER_MAX;

//////////////////////      linkLoss Service
////////////////////////////////
static const u16 linkLoss_serviceUUID  = SERVICE_UUID_LINK_LOSS;
//static const u16 alertLevel_charUUID = CHARACTERISTIC_UUID_ALERT_LEVEL;
static const u8 linkLoss_prop = CHAR_PROP_READ | CHAR_PROP_WRITE;//CHAR_PROP_WRITE
| CHAR_PROP_NOTIFY;
u8 linkLoss_value = 60;
u8 linkLoss_valueInCCC[2];
u8 batValInCCC[2];



static const u16 immediateAlert_serviceUUID  = SERVICE_UUID_IMMEDIATE_ALERT;
static const u16 alertLevel_charUUID = CHARACTERISTIC_UUID_ALERT_LEVEL;
static const u8 immediateAlertLevel_prop = CHAR_PROP_WRITE |
CHAR_PROP_WRITE_WITHOUT_RSP;//CHAR_PROP_WRITE | CHAR_PROP_NOTIFY;
u8 immediateAlertLevel_value = 0;
u8 immediateAlertLevel_valueInCCC[2];

u8  generalValInCCC[2];

#if(KEYBOARD_REPORT_SUPPORT)
u8 reportKeyIn[8]={0,0,0,0,0,0,0,0};//globle
const static u8 reportRefKeyIn[2]
={HID_REPORT_ID_KEYBOARD_INPUT,HID_REPORT_TYPE_INPUT };

u8 reportKeyOut;
const static u8 reportRefKeyOut[2]={HID_REPORT_ID_KEYBOARD_INPUT,
HID_REPORT_TYPE_OUTPUT };

#endif

#if(JOYSTIC_REPORT_SUPPORT)
u8 reportJoyStickIn[9];//globle
//u8 generalValInCCC[2];
const static u8 reportRefJoyStickIn[2] ={HID_REPORT_ID_JOYSTIC_INPUT,
HID_REPORT_TYPE_INPUT };
```

```
#endif
/////////////// //定义设备的服务属性 UUID 为某个常数 end//////////////////////

///////////////// //定义设备的安全属性 值 ////////////////////

#if(CONSUME_REPORT_SUPPORT)
u8 reportConsumerControlIn[2];
//u8 generalValInCCC[2];
const static u8 reportRefConsumerControlIn[2] =
{ HID_REPORT_ID_CONSUME_CONTROL_INPUT, HID_REPORT_TYPE_INPUT };
#endif

#if(MOUSE_REPORT_SUPPORT)
u8 reportMouseIn[4];
// u8 generalValInCCC[2];
const static u8 reportRefMouseIn[2] = { HID_REPORT_ID_MOUSE_INPUT,
HID_REPORT_TYPE_INPUT };
#endif

// HID Information characteristic
const u8 hidInformation[] =
{
  U16_LO(0x0111), U16_HI(0x0111),           // bcdHID (USB HID version)
  0x00,                                     // bCountryCode
  0x01                                      // Flags
};

static u8 controlPoint;

u8  ph_devName [25] = {' ',' ',' ',' ',' ',' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' '};
```

HID 协议使得设备的实现变得简单，设备会定义数据包为 HID 描述符发送给主机。HID 描述符是描述设备数据包的固定代码字节数组，包括设备支持多少个包，包有多大，以及包中每个字节和比特的含义。比如，带有计算程序按键的键盘告诉主机按键是按下还是松开状态，该信息放在数据包 4 的第 6 个字节的第 2 个比特，注意这个位置是设备指定说明的。设备通常将 HID 描述符存放在 ROM 里，不必深入理解或分析 HID 描述符。今天市场上的一些鼠标和键盘硬件实现仅仅使用一个 8 比特的 CPU。
详情可查阅：https://zhuanlan.zhihu.com/p/27568561

```
//USB HID 报告描述符 为 IOS 设备使用的
static const u8 reportMapIos[] =
{

/*********************************keyboard*************************/
#if(KEYBOARD_REPORT_SUPPORT)
```

```
0x05, 0x01,  // Usage Pg (Generic Desktop)
0x09, 0x06,  // Usage (Keyboard)
0xA1, 0x01,  // Collection: (Application)
0x85, HID_REPORT_ID_KEYBOARD_INPUT,  // Report Id (2)
//
0x05, 0x07,  // Usage Pg (Key Codes)
0x19, 0xE0,  // Usage Min (224)
0x29, 0xE7,  // Usage Max (231)
0x15, 0x00,  // Log Min (0)
0x25, 0x01,  // Log Max (1)
//
//  Modifier byte
0x75, 0x01,  // Report Size (1)
0x95, 0x08,  // Report Count (8)
0x81, 0x02,  // Input: (Data, Variable, Absolute)
//
//  Reserved byte
0x95, 0x01,  // Report Count (1)
0x75, 0x08,  // Report Size (8)
0x81, 0x01,  // Input: (Constant)
//
//  LED report
0x95, 0x05,  // Report Count (5)
0x75, 0x01,  // Report Size (1)
0x05, 0x08,  // Usage Pg (LEDs)
0x19, 0x01,  // Usage Min (1)
0x29, 0x05,  // Usage Max (5)
0x91, 0x02,  // Output: (Data, Variable, Absolute)
//
//  LED report padding
0x95, 0x01,  // Report Count (1)
0x75, 0x03,  // Report Size (3)
0x91, 0x01,  // Output: (Constant)
//
//  Key arrays (6 bytes)
0x95, 0x06,  // Report Count (6)
0x75, 0x08,  // Report Size (8)
0x15, 0x00,  // Log Min (0)
0x25, 0x65,  // Log Max (101)
0x05, 0x07,  // Usage Pg (Key Codes)
0x19, 0x00,  // Usage Min (0)
0x29, 0x65,  // Usage Max (101)
0x81, 0x00,  // Input: (Data, Array)
//
0xC0,     // End Collection
```

```
#endif
/*********************consumer control************************/
#if(CONSUME_REPORT_SUPPORT)

0x05, 0x0C,     // USAGE_PAGE (Consumer Devices)
0x09, 0x01,     // USAGE (Consumer Control)
0xA1, 0x01,     // COLLECTION (Application)
0x85, HID_REPORT_ID_CONSUME_CONTROL_INPUT,     // Report ID (2)
0x75, 0x10,     // REPORT_SIZE (10)
0x95, 0x01,     // REPORT_COUNT (2)
0x15, 0x01,     // LOGICAL_MINIMUM (1)
0x26, 0x8c, 0x02, // LOGICAL_MAXIMUM (28c)
0x19, 0x01,     // USAGE_MINIMUM (Button 1)
0x2a, 0x8c, 0x02, // USAGE_MAXIMUM (Button 28c)
0x81, 0x60,     // INPUT (data, array, abs)
0xc0,   // END COLLECTION


#endif
};
//USB HID 报告描述符 为 ANDROID 设备使用
static const u8 reportMapAndroid[] =
{
#if 1
/*****************************keyboard(65)************************/
#if(KEYBOARD_REPORT_SUPPORT)

0x05, 0x01,  // Usage Pg (Generic Desktop)
0x09, 0x06,  // Usage (Keyboard)
0xA1, 0x01,  // Collection: (Application)
0x85, HID_REPORT_ID_KEYBOARD_INPUT,  // Report Id (2)
//
0x05, 0x07,  // Usage Pg (Key Codes)
0x19, 0xE0,  // Usage Min (224)
0x29, 0xE7,  // Usage Max (231)
0x15, 0x00,  // Log Min (0)
0x25, 0x01,  // Log Max (1)
//
//  Modifier byte
0x75, 0x01,  // Report Size (1)
0x95, 0x08,  // Report Count (8)
0x81, 0x02,  // Input: (Data, Variable, Absolute)
//
//  Reserved byte
```

```
0x95, 0x01,  // Report Count (1)
0x75, 0x08,  // Report Size (8)
0x81, 0x01,  // Input: (Constant)
//
//  LED report
0x95, 0x05,  // Report Count (5)
0x75, 0x01,  // Report Size (1)
0x05, 0x08,  // Usage Pg (LEDs)
0x19, 0x01,  // Usage Min (1)
0x29, 0x05,  // Usage Max (5)
0x91, 0x02,  // Output: (Data, Variable, Absolute)
//
//  LED report padding
0x95, 0x01,  // Report Count (1)
0x75, 0x03,  // Report Size (3)
0x91, 0x01,  // Output: (Constant)
//
//  Key arrays (6 bytes)
0x95, 0x06,  // Report Count (6)
0x75, 0x08,  // Report Size (8)
0x15, 0x00,  // Log Min (0)
0x25, 0x65,  // Log Max (101)
0x05, 0x07,  // Usage Pg (Key Codes)
0x19, 0x00,  // Usage Min (0)
0x29, 0x65,  // Usage Max (101)
0x81, 0x00,  // Input: (Data, Array)
//
0xC0,     // End Collection

#endif
/**********************consumer control(25)************************/
#if(CONSUME_REPORT_SUPPORT)


0x05, 0x0C,       // USAGE_PAGE (Consumer Devices)
0x09, 0x01,       // USAGE (Consumer Control)
0xA1, 0x01,       // COLLECTION (Application)
0x85, HID_REPORT_ID_CONSUME_CONTROL_INPUT,       // Report ID (2)
0x75, 0x10,       // REPORT_SIZE (10)
0x95, 0x01,       // REPORT_COUNT (2)
0x15, 0x01,       // LOGICAL_MINIMUM (1)
0x26, 0x8c, 0x02, // LOGICAL_MAXIMUM (28c)
0x19, 0x01,       // USAGE_MINIMUM (Button 1)
0x2a, 0x8c, 0x02, // USAGE_MAXIMUM (Button 28c)
0x81, 0x60,       // INPUT (data, array, abs)
0xc0,             // END COLLECTION
```

```
#endif

/*****************************mouse ************************/
#if(MOUSE_REPORT_SUPPORT)

  0x05, 0x01,  // Usage Page (Generic Desktop)
  0x09, 0x02,  // Usage (Mouse)
  0xA1, 0x01,  // Collection (Application)
  0x85, HID_REPORT_ID_MOUSE_INPUT,  // Report Id (1)
  0x09, 0x01,  //   Usage (Pointer)
  0xA1, 0x00,  //   Collection (Physical)
  0x05, 0x09,  //     Usage Page (Buttons)
  0x19, 0x01,  //     Usage Minimum (01) - Button 1
  0x29, 0x03,  //     Usage Maximum (03) - Button 3
  0x15, 0x00,  //     Logical Minimum (0)
  0x25, 0x01,  //     Logical Maximum (1)
  0x75, 0x01,  //     Report Size (1)
  0x95, 0x03,  //     Report Count (3)
  0x81, 0x02,  //     Input (Data, Variable, Absolute) - Button states
  0x75, 0x05,  //     Report Size (5)
  0x95, 0x01,  //     Report Count (1)
  0x81, 0x01,  //     Input (Constant) - Padding or Reserved bits
  0x05, 0x01,  //     Usage Page (Generic Desktop)
  0x09, 0x30,  //     Usage (X)
  0x09, 0x31,  //     Usage (Y)
  0x09, 0x38,  //     Usage (Wheel)
  0x15, 0x81,  //     Logical Minimum (-127)
  0x25, 0x7F,  //     Logical Maximum (127)
  0x75, 0x08,  //     Report Size (8)
  0x95, 0x03,  //     Report Count (3)
  0x81, 0x06,  //     Input (Data, Variable, Relative) - X & Y coordinate
  0xC0,        //   End Collection
  0xC0,        // End Collection

#endif

/*********************Game Pad************************/
#if(JOYSTIC_REPORT_SUPPORT)

0x05, 0x01,                    // USAGE_PAGE (Generic Desktop)
0x09, 0x05,                    // USAGE (Game Pad)
0xA1, 0x01,                  // Collection (Application)
0x85, HID_REPORT_ID_JOYSTIC_INPUT,                //     Report Id (4)
0x05, 0x01,                     // USAGE_PAGE (Generic Desktop)
0x09, 0x01,                    //   USAGE (Pointer)
```

```
0xA1, 0x00,                    //   COLLECTION (Physical)


0x09, 0x30,                    //     USAGE (X)
0x09, 0x31,                    //     USAGE (Y)
0x09, 0x32,                    //     USAGE (Z)
0x09, 0x35,                    //     USAGE (RZ)
//0x09, 0x33,                  //     USAGE (RX)
//0x09, 0x34,                  //     USAGE (RY)
0x15, 0x00,  //   Logical Minimum (-127)
0x26, 0xff,0x00 , //     Logical Maximum (127)
0x75, 0x08,  //   Report Size (8)
0x95, 0x04,  //   Report Count (3)
0x81, 0x02,  //     INPUT (Data,Var,Abs)
0XC0,

0x09, 0x39, //   USAGE (Hat switch)
0x15, 0x00,  //   Logical Minimum (-127)
0x25, 0x07,  //   Logical Maximum (127)
0x35, 0x00,
0x46, 0x3b,0x01,
0x65, 0x14,
0x75, 0x04,  //   Report Size (8)
0x95, 0x01,  //   Report Count (1)
0x81, 0x42,  //     INPUT (Data,Var,Abs)
0x75, 0x04,  //   Report Size (8)
0x95, 0x01,  //   Report Count (1)
0x81, 0x01,


0x05, 0x09,                    //   USAGE_PAGE (Button)
0x15, 0x00,                    //   LOGICAL_MINIMUM (0)
0x25, 0x01,                    //   LOGICAL_MAXIMUM (1)
0x19, 0x01,                    //   USAGE_MINIMUM (Button 1)
0x29, 0x10,                    //   USAGE_MAXIMUM (Button 12)
0x75, 0x01,                    //   REPORT_SIZE (1)
0x95, 0x10,                    //   REPORT_COUNT (12)
0x81, 0x02,                    //   INPUT (Data,Var,Abs)

0x05, 0x02,                    // USAGE_PAGE
0x15, 0x00,
0x26, 0xff,0x00,
0x09, 0xc4,
0x09, 0xc5,
0x75, 0x08,
0x95, 0x02,
```

```
0x81, 0x02,
0xc0,

#endif

};

//att 服务获取报告内容
extern u8 os_check;
u8* att_get_reportMap(){

    if(os_check == 2){
        return (u8*)(reportMapAndroid);
    }else{
        return (u8*)(reportMapIos);
    }
}
//att 服务获取报告内容的长度
int att_get_reportMapSize(){

    if(os_check == 2){
        return sizeof(reportMapAndroid);
    }else{
        return sizeof(reportMapIos);
    }

}
//定义 att_ota 服务的 UUID 为一个数组常数
const u8 ota_service_uuid[16] =
{0x11,0x19,0x0d,0x0c,0x0b,0x0a,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x0
0};
//定义 ota_write_服务的 UUID 为一个数组常数
const u8 ota_write_char_uuid[16]
={0x12,0x2B,0x0d,0x0c,0x0b,0x0a,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x
00} ;

u8 ota_data[20];
//定义 设备当前所有的服务为一个 attribute_t 结构体常数，且每个服务由本身的服务 UUID，名
下的属性 UUID 和属性
const attribute_t my_Attributes[] =
{
{14+HID_CONTROL_POINT_DP_H-HID_PS_H+1,0,0,0}, //定义一个长度为
14+HID_CONTROL_POINT_DP_H-HID_PS_H+1 的数组，用来存储当前设备的 handle 总条目数量
 /*****************************************************************************
                Handle 从 1 开始的 GenericAttribute(Services)  服务
*****************************************************************************/
```

```
    // gatt information
    {5,2,GATT_UUID_PRIMARY_SERVICE,    (u8*)(&my_gapServiceUUID)},
    {0,1,GATT_UUID_CHARACTER,      (u8*)(&my_devNameCharacter)},
    {0,sizeof (my_devName), GATT_UUID_DEVICE_NAME,
    (u8*)(my_devName)},
    {0,1,GATT_UUID_CHARACTER,      (u8*)(&my_appearanceCharacter)},
    {0,sizeof (my_appearance), 0x2a01,     (u8*)(&my_appearance)},

 ////////////////////////////////////////////////////
          Handle 从 6 开始的 Battery Service  服务
//////////////////////////////////////////////////////////////////////////
    {3,2,GATT_UUID_PRIMARY_SERVICE,    (u8*)(&my_batServiceUUID)},
    {0,1,GATT_UUID_CHARACTER,      (u8*)(&my_batProp)},              //prop
    {0,1,CHARACTERISTIC_UUID_BATTERY_LEVEL,    (u8*)(my_batVal)}, //value

    ////////////////////////////////////////////////////////////////

          Handle 从 9 开始的 Immediate Alert  Service  服务

   ////////////////////////////////////////////////
    {3,2,GATT_UUID_PRIMARY_SERVICE,    (u8*)(&immediateAlert_serviceUUID)},
    {0,1,GATT_UUID_CHARACTER,      (u8*)(&immediateAlertLevel_prop)},
    {0,1,CHARACTERISTIC_UUID_ALERT_LEVEL, (u8*)(&immediateAlertLevel_value)},

////////////////////////////////////////////////////
          Handle 从 12 开始的. Private Service 私有服务
////////////////////////////////////////////////////
    {3,2,GATT_UUID_PRIMARY_SERVICE,    (u8*)(&privateSeviceUUID)},
    {0,1,GATT_UUID_CHARACTER,      (u8*)(&privatekeyNoti_prop)},
    {0,1,0xffe1,    (u8*)(&privateKeyNoti_value)},

/*************************************************************************
       Handle 从 15 开始的 Human Interface Device   HID (Services)服务
*************************************************************************/
{HID_CONTROL_POINT_DP_H-HID_PS_H+1,2,2,(u8*)(&my_primaryServiceUUID),
(u8*)(&my_hidServiceUUID)},
//include battery service property
{0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ_WRITE_NORSP)},
{0,2,sizeof(protocolMode),(u8*)(&hidProtocolModeUUID),   (u8*)(&protocolMode)},
//当支持键盘 HID 报告时，则添加对应的 UUID 和可读可写属性
 #if(KEYBOARD_REPORT_SUPPORT)
    // report in : 4 (char-val-client-ref), handle start from 18
    //property
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ_NOTIFY)},
     //value
    {0,2,sizeof(reportKeyIn),(u8*)(&hidReportUUID),   (u8*)(reportKeyIn)},
```

```
{0,2,sizeof(generalValInCCC),(u8*)(&clientCharacterCfgUUID),
(u8*)(generalValInCCC)},
    {0,2,sizeof(reportRefKeyIn),(u8*)(&reportRefUUID),
(u8*)(reportRefKeyIn)},
#endif

#if(CONSUME_REPORT_SUPPORT)
//当支持 CONSUME HID 报告时，则添加对应的 UUID 和可读可写属性
    //  consumer report in: 4 (char-val-client-ref)
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ_NOTIFY)},
//prop
    {0,2,sizeof(reportConsumerControlIn),(u8*)(&hidReportUUID),
(u8*)(reportConsumerControlIn)}, //value
    {0,2,sizeof(generalValInCCC),(u8*)(&clientCharacterCfgUUID),
(u8*)(generalValInCCC)},  //value
    {0,2,sizeof(reportRefConsumerControlIn),(u8*)(&reportRefUUID),
(u8*)(reportRefConsumerControlIn)},  //value
#endif


#if(MOUSE_REPORT_SUPPORT)
//当支持鼠标 HID 报告时，则添加对应的 UUID 和可读可写属性
    {0,2,1,(u8*)(&my_characterUUID),      (u8*)(&PROP_READ_NOTIFY)},
    {0,2,sizeof(reportMouseIn),(u8*)(&hidReportUUID),
    (u8*)(&reportMouseIn)},
    {0,2,sizeof(generalValInCCC),(u8*)(&clientCharacterCfgUUID),
    (u8*)(generalValInCCC)},
    {0,2,sizeof(reportRefMouseIn),(u8*)(&reportRefUUID),
    (u8*)(reportRefMouseIn)},
#endif


#if(JOYSTIC_REPORT_SUPPORT)
    //当支持游戏手柄 HID 报告时，则添加对应的 UUID 和可读可写属性
     // report in : 4 (char-val-client-ref), handle start from 0x19
    //property
 {0,2,1,(u8*)(&my_characterUUID),    (u8*)(&PROP_READ_NOTIFY)},
 //value
{0,2,sizeof(reportJoyStickIn),(u8*)(&hidReportUUID),
(u8*)(reportJoyStickIn)},
{0,2,sizeof(generalValInCCC),(u8*)(&clientCharacterCfgUUID),
    (u8*)(generalValInCCC)},
 //value
        {0,2,sizeof(reportRefJoyStickIn),(u8*)(&reportRefUUID),
    (u8*)(reportRefJoyStickIn)},
#endif
```

```
 };


//初始化当前设备的服务属性，以及有用到 HID 的 SMP 加密属性时，完成对应的初始化
void    shutter_att_init ()
{
    extern attribute_t* gAttributes;
    gAttributes = (attribute_t *)my_Attributes;

     blt_smp_func_init ();// 完成 HID 的 SMP 加密属性的初始化

}
// HID 的 SMP 加密属性初始化
void hid_setting_flag(u16 en)
{

     generalValInCCC[0] = en;
}
// 查询和返回 HID 的 SMP 加密属性
u8  get_hid_ccc_flag()
{
    return generalValInCCC[0];
}
```