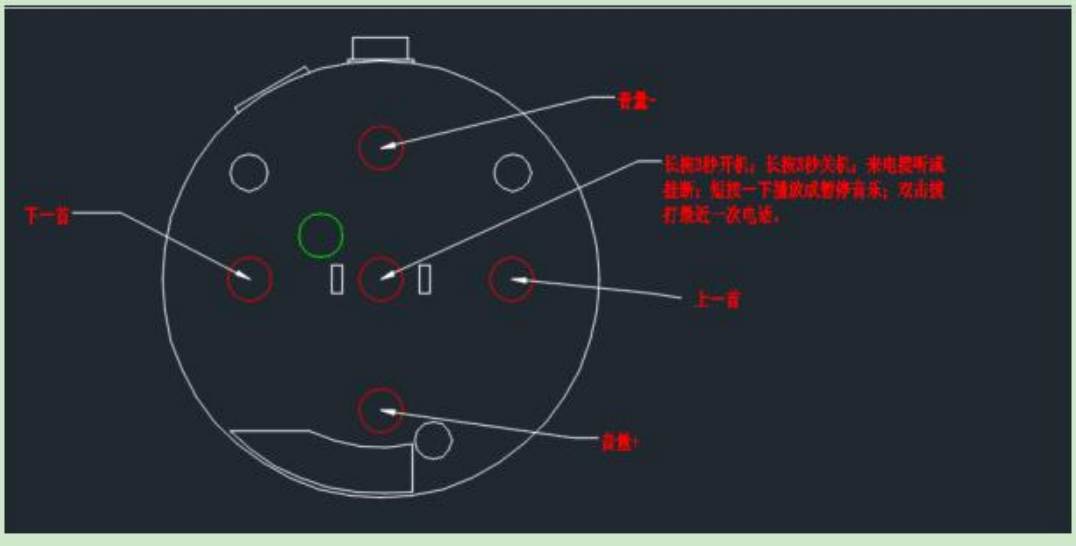
ST17H26 HID SDK introduction

ST17H26 HID SDK introduction....1
1 Customer Requirements....2
1.1 Analyse the protocol....2
1.2 add the needed HID protocol into app_att.c....3
1.2.1 Firstly ,try to understand the report description protocol....3
1.2.2 open the HID sdk protocol....4
1.2.3 add the HID UUID and attribute into this attribute_t list....9
1.2.4 Initialize the bond-related functions....10
1.2.4.1 call the function blt_smp_func_init(); to enable the SMP....11
1.2.4.2 change the task_bond_finished()function to enable the pair mode....11
1.2.4.3 add the IOS/Android Distinguish function among public_loop()....12
1.2.4.5 test whether the paring mode is successfully configured or not....12
1.3 Test Function....15

1 Customer Requirements



遥控器的显示名称为：X1-遥控器

5 个按键：功能按图片。 播放、暂停。上一歌曲，下一歌曲，声音加，声音减 。（不做电话功能）

控制手机。 手机会同时连接 2 个设备名称；一个遥控器，另外一个是蓝牙接收器。遥控器控制手机的功能。接收器出声音，2 个互相不干涉。

Figure 1customer requirements

This is a ble-based music controller , each button on this PCB reps a different function (eg:up-button: volume up; down-button : volume down ; middle button : long- press :on-off; short-press:play/pause).

When connect to the mobile phone ,it will connect this ble-based music controller ,which won't be necessary to install any special application,simply open the default music player on the mobile phone would be enough.

Actually , you can also develop an special Application and protocol with a GATT device ,without look into this HID sdk,however this is just a mention in passing!

1.1 Analyse the protocol

USB HID to PS/2 Scan Code Translation Table

Key Name	HID Usage Page	HID Usage ID	PS/2 Set 1 Make*	PS/2 Set 1 Break*	PS/2 Set 2 Make	PS/2 Set 2 Break

Scan Next Track	0C	00B5	E0 19	E0 99	E0 4D	E0 F0 4D
Scan Previous Track	0C	00B6	E0 10	E0 90	E0 15	E0 F0 15
Stop	0C	00B7	E0 24	E0 A4	E0 3B	E0 F0 3B
Play/ Pause	0C	00CD	E0 22	E0 A2	E0 34	E0 F0 34
Mute	0C	00E2	E0 20	E0 A0	E0 23	E0 F0 23
Bass Boost	0C	00E5	UNASSIGNED	UNASSIGNED	UNASSIGNED	UNASSIGNED
Loudness	0C	00E7	UNASSIGNED	UNASSIGNED	UNASSIGNED	UNASSIGNED
Volume Up	0C	00E9	E0 30	E0 B0	E0 32	E0 F0 32
Volume Down	0C	00EA	E0 2E	E0 AE	E0 21	E0 F0 21

Figure 2 HID protocol

Upwards are some report descriptor which come from the USB HID ,since functions that we want can be accomplished in this table ,you can see each one of them ,with a red arrow mark.For now, we can just follow the basic steps of BLE sending process to finish this.

Here is a pseudo code ,that we can call ,when detect the corresponding key on the PCB is pressed.

```
case BSP_EVENT_KEY_0:
    if ( m_conn_handle != BLE_CONN_HANDLE_INVALID )
    {
        media_player_control( 0x01);     //开始/暂停
    }
    break;

case BSP_EVENT_KEY_1:
    if ( m_conn_handle != BLE_CONN_HANDLE_INVALID )
    {
        media_player_control(0x04 );//下一首
    }
    break;

case BSP_EVENT_KEY_2:
    if ( m_conn_handle != BLE_CONN_HANDLE_INVALID )
    {
        media_player_control(0x08 );//上一首
    }
    break;

case BSP_EVENT_KEY_3:
    if ( m_conn_handle != BLE_CONN_HANDLE_INVALID )
    {
        media_player_control( 0x10);//音量-
    }
    break;

default:
    break;
```

Figure 3 implementing pseudo code

1.2 add the needed HID protocol into app_att.c

1.2.1 Firstly ,try to understand the report description protocol

Here is a reference doc in case you might want to have a look at the basic introduction of HID protocol https://zhuanlan.zhihu.com/p/27568561 . And down below is how the HID data format is organized in Lenze HID sdk.

```
    // Report ID 3: Advanced buttons
    0x05, 0x0C,         // Usage Page (Consumer)
    0x09, 0x01,         // Usage (Consumer Control)
    0xA1, 0x01,         // Collection (Application)
    0x85, 0x03,         //     Report Id (3)
    0x15, 0x00,         //     Logical minimum (0)
    0x25, 0x01,         //     Logical maximum (1)
    0x75, 0x01,         //     Report Size (1)
    0x95, 0x01,         //     Report Count (1)

    0x09, 0xCD,         //     Usage (Play/Pause)                          //开始/暂停
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0x0A, 0x83, 0x01,   //     Usage (AL Consumer Control Configuration)   //一键启动应用.
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0x09, 0xB5,         //     Usage (Scan Next Track)                     //下一首
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0x09, 0xB6,         //     Usage (Scan Previous Track)                 //上一首
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)

    0x09, 0xEA,         //     Usage (Volume Down)                         //音量-
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0x09, 0xE9,         //     Usage (Volume Up)                           //音量+
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0x0A, 0x25, 0x02,   //     Usage (AC Forward)                          //应用控制
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0x0A, 0x24, 0x02,   //     Usage (AC Back)                             //返回键
    0x81, 0x06,         //     Input (Data,Value,Relative,Bit Field)
    0xC0                // End Collection
};
```

Figure 4 HID data structure

1.2.2 open the HID sdk protocol

Take a look at it, you will find out this is similar to the GATT sdk, and since the HID sdk contains the standard HID architecture and basic files ,this chapter will introduce how to use this SDK to accomplish the customer's requirements.

Since this part is quiet similar to the <GATT SDK Introduction.pdf>,we recommend you go back to that doc and have a look .

```
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[];
```

```
//////////////////////////the properties of the UUID service ////////////////////////////
const u8 PROP_READ = CHAR_PROP_READ;
```

```
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};

/////////////// //Generic UUID，the property and the parameters.////////////////////////////

const u16 my_gapServiceUUID = SERVICE_UUID_GENERIC_ACCESS;

const u16 my_appearance = GAP_APPEARE_ROLE;//global //

const gap_periConnectParams_t my_periConnParameters = {30, 60, 4, 1000};

//////////////////////////////////////////////////////////////////////////////
 Here are the HID :Human interface Devices and the respected UUID ,property and parameters.
//////////////////////////////////////////////////////////////////////////////

const u16 my_hidServiceUUID = SERVICE_UUID_HUMAN_INTERFACE_DEVICE;

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

#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] = {'','','','','','','','','','','','','','','','','','','','','','','','',''};
//here is how the device distinguish the IOS and Android phone .
extern u8 os_check;
u8* att_get_reportMap(){
if(os_check == 2){
return (u8*)(reportMapAndroid);
}else{
return (u8*)(reportMapIos);
}
}

//here is how the device get the length of the Report Content.
int att_get_reportMapSize(){
if(os_check == 2){
return sizeof(reportMapAndroid);
}else{
return sizeof(reportMapIos);
}
}
```

1.2.3 add the HID UUID and attribute into this attribute_t list

```
const attribute_t my_Attributes[] =
{
  {50,0,0,0,0},    //the total attributes listed here.
/**********************************************************************
      generic UUID which start with handle =1
**********************************************************************/
// 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)},
/**********************************************************************
      battery service UUID which start with handle =6
**********************************************************************/

{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

/**********************************************************************
immediateAlert service UUID which start with handle=9
**********************************************************************/
{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)},
/**********************************************************************
    private Sevice UUID which start with handle =12
**********************************************************************/
{3,2,GATT_UUID_PRIMARY_SERVICE, (u8*)(&privateSeviceUUID)},
{0,1,GATT_UUID_CHARACTER, (u8*)(&privatekeyNoti_prop)},
{0,1,0xffe1, (u8*)(&privateKeyNoti_value)},
/**********************************************************************
    HID Sevice UUID which start with handle =15
**********************************************************************/

{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)},
// when enable this KEYBOARD REPORT SUPPORT,we can send the keyboard command to the
BLE // devices
```

```
#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)},
{0,2,sizeof(reportKeyIn),(u8*)(&hidReportUUID), (u8*)(reportKeyIn)},//value
{0,2,sizeof(generalValInCCC),(u8*)(&clientCharacterCfgUUID),(u8*)(generalValInCCC)},
{0,2,sizeof(reportRefKeyIn),(u8*)(&reportRefUUID),(u8*)(reportRefKeyIn)},
#endif
// when enable this CONSUME REPORT SUPPORT,we can send the media-related command (e.g.: volume up; volume down ;  previous song;  next song; play/pause )to the BLE devices
#if(CONSUME_REPORT_SUPPORT)
{0,2,1,(u8*)(&my_characterUUID), (u8*)(&PROP_READ_NOTIFY)},
//prop // consumer report data exist in  array :reportConsumerControlIn[2]
{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
// when enable this MOUSE REPORT SUPPORT,we can send the mouse-related command to the BLE devices
#if(MOUSE_REPORT_SUPPORT)
  {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
// when enable this JOYSTIC REPORT SUPPORT,we can send the joystic-related command to the BLE devices
#if(JOYSTIC_REPORT_SUPPORT)
  // report in : 4 (char-val-client-ref), handle start from 0x19
//report data exist in  array :reportJoyStickIn[]
{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
};
```

1.2.4 Initialize the bond-related functions

When HID relates to the SMP(ie: -security message protocol), you must follow the following programs without change them .

1.2.4.1 call the function blt_smp_func_init(); to enable the SMP

```
void shutter_att_init (){
extern attribute_t* gAttributes;
gAttributes = (attribute_t *)my_Attributes;
blt_smp_func_init ();//  initialize the HID SMP function.

}
```

```
678
679 void    shutter_att_init ()
680 {
681     /*blt_set_att_table ((u8 *)my_Attributes);*/
682     extern attribute_t* gAttributes;
683     gAttributes = (attribute_t *)my_Attributes;
684
685     //blt_smp_set_pairing_type(1,0);
686
687      blt_smp_func_init ();
688 }
689
690 void hid_setting_flag(u16 en)
691 {
```

This is related to pairing procedure.

Figure 5 pairing program

1.2.4.2 change the task_bond_finished()function to enable the pair mode

Ui_empo.c | yj_ui.c | yj_ui.h | app_att.c

```
709         //clear and init flag
710         selfie_adv_mode_start_tick =  clock_time ();
711         blt_is_reconnection = 0;
712         return;
713 }
714
715 /*task_bond_finished
716  * This event is returned once encryption process is finished
717  * ex: HID key can be used once connection is encrypted*/
718
719 void task_bond_finished(rf_packet_connect_t* p){
720     if(!blt_smp_paring_req_recvd){// reconnection
721         if(os_check == 2){
722             hid_setting_flag(1);
723         }
724     }
725 }
726 #endif
727
```

Figure 6 bond program

1.2.4.3 add the IOS/Android Distinguish function among public_loop()

```
}

static inline void public_loop()
{

    tick_app_wakeup = buzzer_led_ui() ;
    if(tick_app_wakeup)
        blt_brx_sleep (tick_app_wakeup);
    else{
        blt_sleep_wakeup(0,PM_WAKEUP_TIMER, 200 * CLOCK_SYS_CLOCK_1MS);
    }
   if(blt_state!=BLT_LINK_STATE_ADV)
   {
       blt_brx ();

       if(blt_conn_inst > 20 && os_check < 2)
       {
           os_check = 2;
           vr_autoSetMode();
           hid_setting_flag(1);    //  android set ccc at default
       }
   }
   else
   {
        // Must be on the final
        blt_send_adv (BLT_ENABLE_ADV_ALL);
        //blt_send_adv (BLT_ENABLE_ADV_38);
   }
}


void connection_para(){
    if (blt_state == BLT_LINK_STATE_ADV)
```

Figure 7 implementing HID protocol

1.2.4.5 test whether the paring mode is successfully configured or not

Go back to the Ui.c file ,build the program and load the BIN file into the black board . Connect to the mobile phone ,finish the pairing .

```
933
934   void main_loop()
935   {
936       test_suspend_time_tmp();
937       if(clock_time_exceed(tick_hardware_scan_tmp,20*1000))
938       {
939           tick_hardware_scan_tmp=clock_time();
940           extern u8 start_ota_flag;
941           if(start_ota_flag==0)
942           {
943   //          user_ui_process();
944           }
945
946       }
947
948       {
949       if (blt_state == BLT_LINK_STATE_ADV)
950       {
951           blt_wakeup_src = 0;
952           flag_has_new_event_tmp=0;
953           blt_adv_interval= 20 * CLOCK_SYS_CLOCK_1MS;
954           #if(TEST_OTA_1)
955               if(clock_time_exceed(tick_led_timer_tmp,1000*1000))
956           #else
957               if(clock_time_exceed(tick_led_timer_tmp,200*1000))
958           #endif
959           {
960               tick_led_timer_tmp=clock_time();
961               cur_led_state_tmp = !cur_led_state_tmp;
962               gpio_write(ADV_LED_PORT,cur_led_state_tmp);
963           }
964       }
965       else
966       {
967           //user_ui_process();
968           hw_scan();
969
970           if(connected_idle_time_count_tmp==0)
971           {
972   //          blt_retry=1;
973           }
974       }
975
976
977   }
978   /*****************************public area******************/
```

广播模式下的广播间隔参数设置以及led闪灯

连接之后的键值检测和音量加减控制函数。

Figure 8 implementing key board checking program

And detect the I/O state in the board ,send volume up or down command to mobile phone ,you can finish this function of changing the volume of the mobile phone.

The button-detect function is realized in hw_scan() shown below,and here we only list an example of sending the ‘volume up’ and ‘volume down’ command .After mobile phone is paired to the device ,it will be able to receive media command each time when the button is pressed.

```
static inline u16  button_get_value()
{
    u16 status,value;

    value=0;

    status=button_get_status(GPIO_GP10);//GPIO_GP10
    value|=(status==1)?0x01:((status==2)?0x02:0x00);
    return value;
}

static inline void hw_scan(){
    button_value = button_get_value();
    if(button_value == button_value_bcup){
        return;
    }
    button_value_bcup = button_value;
    if(button_value == 0){
        flag_has_new_event_tmp|=SEND_C_DATA;
        reportConsumerControlIn[0]=0x0;
        blt_push_notify_data(23,reportConsumerControlIn,2);   //HID_CONSUME_KB_REPORT_INPUT_DP_H

    }
    else{
        if(button_value&0x01){
            flag_has_new_event_tmp|=SEND_C_DATA;
            reportConsumerControlIn[0]=0xea;
            blt_push_notify_data(23,reportConsumerControlIn,2);   //HID_CONSUME_KB_REPORT_INPUT_DP_H

        }
        else if(button_value&0x02){
            flag_has_new_event_tmp|=SEND_C_DATA;
            reportConsumerControlIn[0]=0xe9;
            blt_push_notify_data(23,reportConsumerControlIn,2);   //HID_CONSUME_KB_REPORT_INPUT_DP_H

        }
    }


}
```

volume up +

volume down --

Figure 9 implementing HID command sending procedure

And you may ask why the command should sent through the handle=23 blt_push_notify_data(23,reportConsumerControlIn,2) ,why the handle=23 reps HID_CONSUME_KB_REPORT_INPUT_DP_H? Now,here is the answer.

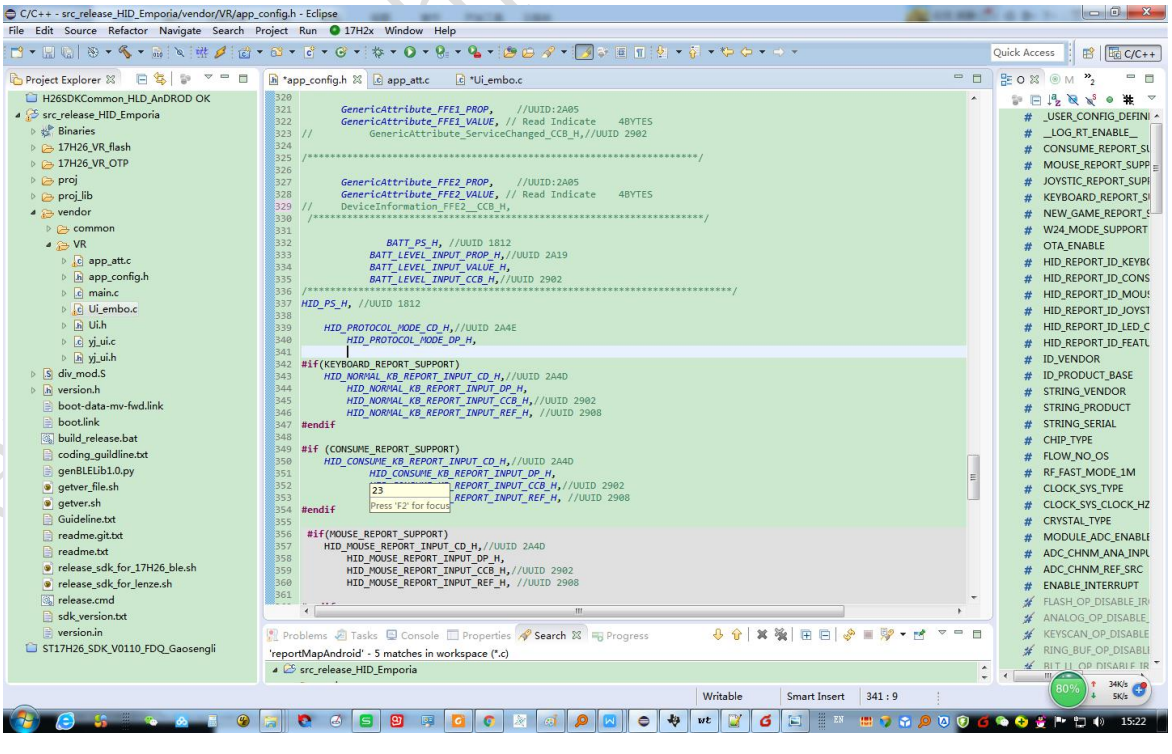


Figure 10 HID handle list in app_config.h

If you still didn't succeed in this step ,then you have to check the 'app_config.h ' file ,which can be understand by below .

When using the media-control function ,you must enable the CONSUME_REPORT_SUPPORT and KEYBOARD_REPORT_SUPPORT ,which is shown in the following picture with red mark .And when you are using the other peripheral function (keyboard ,mouse),just enable the corresponding KEYBOARD_REPORT_SUPPORT or MOUSE_REPORT_SUPPORT will be fine.

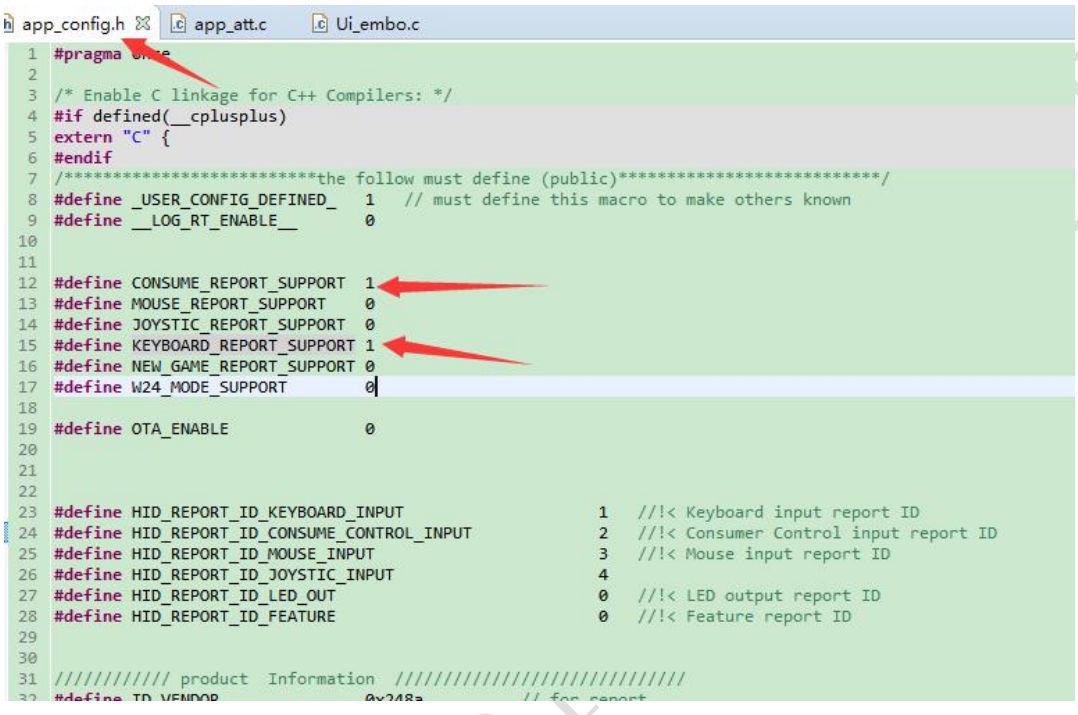


Figure 11 Enable HID Report Support

1.3 Test Function

The basic procedure is as same as the first and second step in the former test,if you accomplish the basic requirements of the product ,then you are succeed.