RTL8762C Keyboard Application Design Spec

V1.0

2018/5/9

修订历史

日期	版本	修改	作者
2018/5/9	Draft		Parker_xue

目录

图表目录

1 概述

1.1 器件清单

1. Bee2 Evaluation Board
2. 4*4 矩阵键盘

1.2 系统需求

PC 端需要下载和安装的工具：

1. Keil MDK-ARM
2. SEGGER’s J-Link tools
3. RTL8762C SDK
4. RTL8762C Flash programming algorithm

1.3 术语定义

1. DLPS：Deep Low Power State。

2 软件结构

键盘应用中主要与 IO driver 和 Upper task 交互，完成特定的应用功能，其中 IO 部分包括了 keyscan 模块以及 GPIO 按键模块，单独的 GPIO 模块主要是用于配对按键。其架构如图所示。

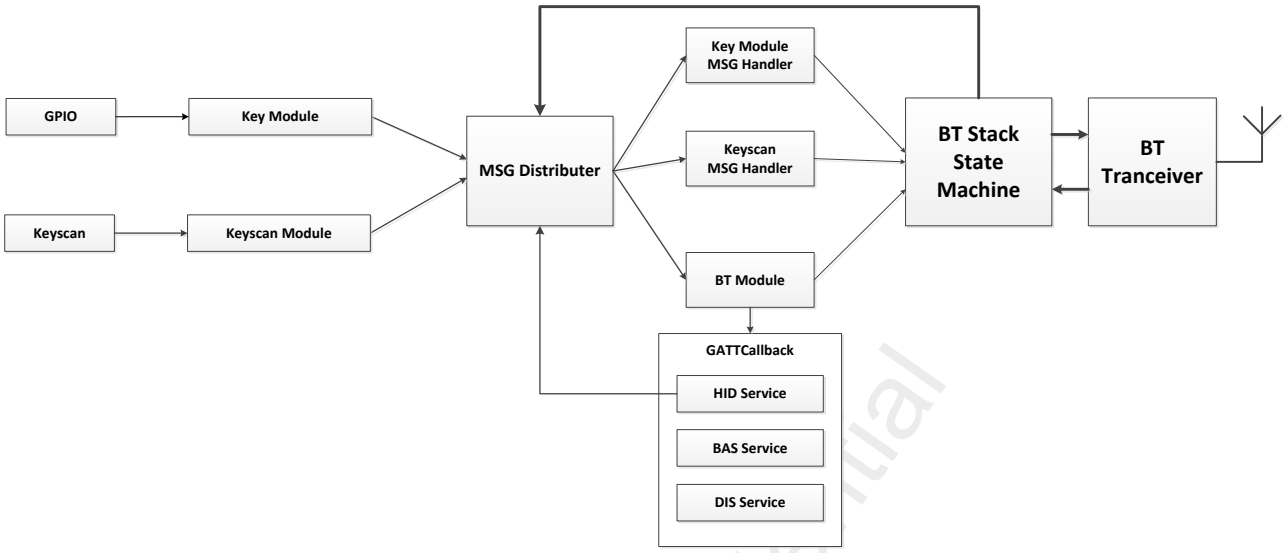

图 2.1 软件架构

系统启动初始化后，App Task 开始运行，等待接收 GPIO 按键、Keyscan 以及 upper stack 发送的 message 命令。为方便对不同消息队列的处理，App Task 中设计了 Event Queue，外部对 App Task 发送消息时，需首先向 Event Queue 中发送该消息所属的类型。而 App Task 则根据消息类型分别读取相应消息队列中的信息，并对消息进一步处理。

```
while (true)
{
    if (os_msg_recv(evt_queue_handle, &event, 0xFFFFFFFF) == true)
    {
        if (event == EVENT_IO_TO_APP)
        {
            T_IO_MSG io_msg;
            if (os_msg_recv(io_queue_handle, &io_msg, 0) == true)
            {
                app_handle_io_msg(io_msg);
            }
        }
        else
        {
            gap_handle_msg(event);
        }
    }
}
```

2.1 按键模块

键盘应用中配对按键使用的是单独的 GPIO 实现。在未配对时，长按配对键可发送配对广播，已配对的情况下，长按配对键会清除配对信息，并重新发出配对广播。

软件中，将按键所连接的 GPIO 口设置为电平触发中断的模式，并结合硬件 Timer 实现对按键的去抖处理。在配置 GPIO 中断时，首先配置为低电平触发，当按键按下触发 GPIO 中断时，在中断处理子函数中，系统将读取按键状态，并为按下和释放两种状态分别设置去抖时间，随后打开定时器。

在定时器中断处理子函数中，系统将再次读取按键状态，若与之前不同，则直接结束本次操作，表明此时有按键抖动情况。否则，系统将设置要发送的按键消息，并翻转触发 GPIO 中断的电平，为下一步释放/按下做准备。最后打开 GPIO 中断使能，发送按键消息到 App Task。

8762C GPIO 支持硬件去抖功能，结合 edge 触发使用，可以简化按键部分代码，省去 Timer 部分。但该功能只能在没有进入 DLPS 时使用。

长按键可结合 timer 来实现。Press 事件发生后，发送消息给 app task，在 task 中可以开启 timer。若在 timer 到期时，还未收到 release 事件，则在 timeout 的 callback 中处理长按事件。否则，release 时，timer 还未到期，则直接关闭 timer。

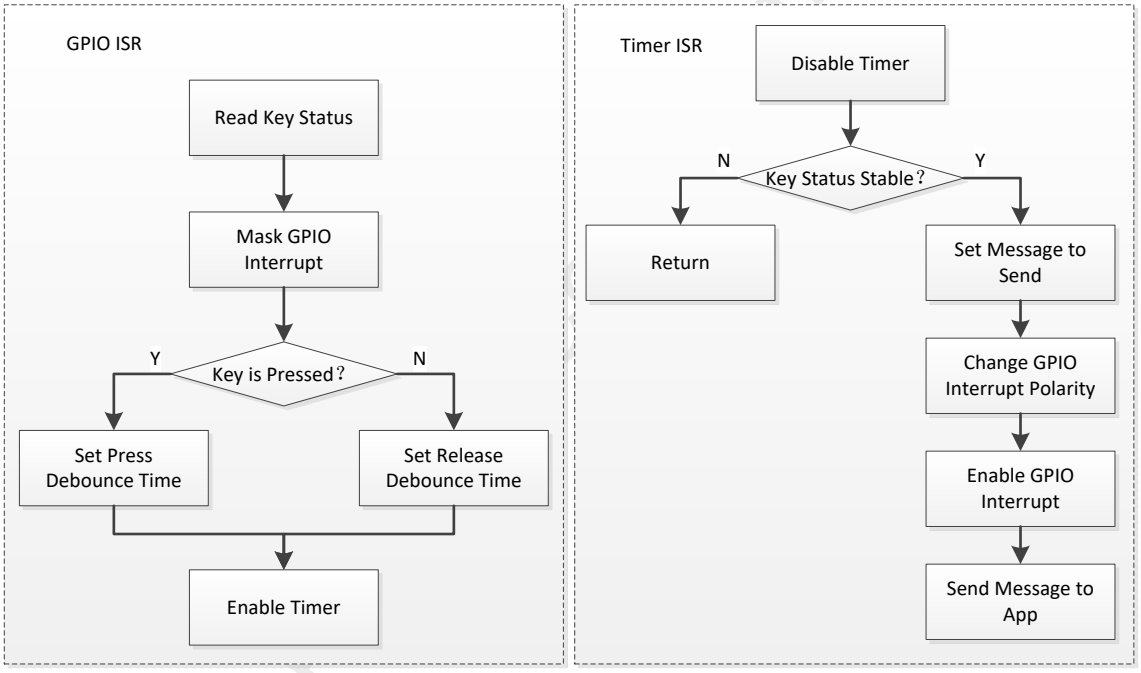

图 2.2 按键处理流程

2.2 Keyscan 模块

Keyboard 使用矩阵键盘，并通过 keyscan 模块进行扫描，在扫描到按键按下时，会通过可 keyscan 中断反馈给 APP。

```
1.  void keyscan_interrupt_handler(void)
2.  {
3.      APP_PRINT_INFO0("[keyscan_interrupt_handler] interrupt handler");
4.
```

```
    T_IO_MSG bee_io_msg;

    if (KeyScan_GetFlagState(KEYSCAN, KEYSCAN_INT_FLAG_SCAN_END) == SET)
    {
        keyscan_global_data.is_allowed_to_enter_dlps = true;
        KeyScan_INTMask(KEYSCAN, KEYSCAN_INT_SCAN_END, ENABLE);  /* Mask keyscan interrupt */

        keyscan_global_data.cur_fifo_data.len = KeyScan_GetFifoDataNum(KEYSCAN);
        if (keyscan_global_data.cur_fifo_data.len != 0)
        {
            /* read keyscan fifo data */
            KeyScan_Read(KEYSCAN, (uint16_t *) &(keyscan_global_data.cur_fifo_data.key[0]),
                         keyscan_global_data.cur_fifo_data.len);
            keyscan_global_data.is_key_pressed = true;
            keyscan_global_data.is_all_key_released = false;

            /* start sw timer to check press status */
            if (!os_timer_restart(&keyscan_timer, KEYSCAN_SW_INTERVAL))
            {
                APP_PRINT_ERROR0("[keyscan_interrupt_handler] restart xTimersKeyScan failed!");
                /* set flag to default status and reinit keyscan module with debounce enabled */
                keyscan_init_data();
                keyscan_init_driver(KeyScan_Debounce_Enable);
                return;
            }

            if (false == keyscan_global_data.is_allowed_to_repeat_report)
            {
                if (!memcmp(&keyscan_global_data.cur_fifo_data, &keyscan_global_data.pre_fifo_data,
                            sizeof(T_KEYSCAN_FIFIO_DATA)))
                {
                    /* some keyscan FIFO data, just return */
                    return;
                }
                else
                {
                    /* updata previous keyscan FIFO data */
                    memcpy(&keyscan_global_data.pre_fifo_data, &keyscan_global_data.cur_fifo_data,
```

```
                        sizeof(T_KEYSCAN_FIFIO_DATA));
                }
            }

            bee_io_msg.type = IO_MSG_TYPE_KEYSCAN;
            bee_io_msg.subtype = IO_MSG_KEYSCAN_RX_PKT;
            bee_io_msg.u.buf   = (void *)(&keyscan_global_data.pre_fifo_data);
            if (false == app_send_msg_to_apptask(&bee_io_msg))
            {
                APP_PRINT_ERROR0("[keyscan_interrupt_handler] send IO_MSG_KEYSCAN_RX_PKT message failed!");
                /* set flag to default status and reinit keyscan module with debounce enabled */
                keyscan_init_data();
                os_timer_stop(&keyscan_timer);
                keyscan_init_driver(KeyScan_Debounce_Enable);
                return;
            }
        }
        else
        {
            if (false == keyscan_global_data.is_all_key_released)
            {
                /* keyscan release event detected */
                APP_PRINT_INFO0("[keyscan_interrupt_handler] keyscan release event detected");
                T_IO_MSG bee_io_msg;
                bee_io_msg.type = IO_MSG_TYPE_KEYSCAN;
                bee_io_msg.subtype = IO_MSG_KEYSCAN_ALLKEYRELEASE;

                if (false == app_send_msg_to_apptask(&bee_io_msg))
                {
                    APP_PRINT_ERROR0("[keyscan_interrupt_handler] Send IO_MSG_TYPE_KEYSCAN message failed!");
                }

                keyscan_init_data();
                keyscan_init_driver(KeyScan_Debounce_Enable);
            }
            else
            {
                /*if system active, keyscan no debounce can arrive here*/
                APP_PRINT_INFO0("[keyscan_interrupt_handler] if system active, keyscan no debounce can arrive here");
```

```
82.                 keyscan_init_data();
83.                 keyscan_init_driver(KeyScan_Debounce_Enable);
84.                 return;
85.             }
86.         }
87.     }
88.     else
89.     {
90.         /* if not KEYSCAN_INT_FLAG_SCAN_END interrupt */
91.         APP_PRINT_INFO0("[keyscan_interrupt_handler] not KEYSCAN_INT_FLAG_SCAN_END
    interrupt");
92.         keyscan_init_data();
93.         keyscan_init_driver(KeyScan_Debounce_Enable);
94.         return;
95.     }
96. }
97.
```

2.3 App task 对各模块消息的处理

GPIO 按键、Keyscan 模块在产生数据后，最终会通过 send message 的方式发送到 App task 中。App task 的任务就是要将不同的 Message 分配给相应的处理函数，最终按照 HID Report map 中定义的数据格式（参见 3.2 节），将其发送到对端。

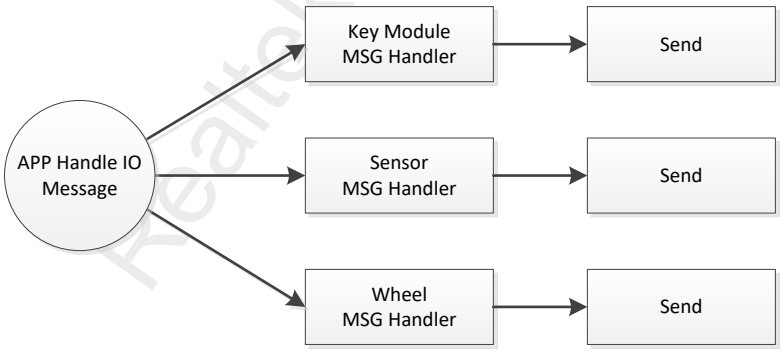

图 2.3 Apptask 对各 IO 消息的分配处理

2.4 配对及回连处理

首次使用未配对时，通过长按配对按键触发配对模式，系统会发送 Undirected Advertising，Host 端搜到设备后，可进行配对操作。配对完成后，会将该设备添加到 resolving list 及 white list 中，resolving list 是底层用于解析 Resolvable Private Address 而使用的。

断线回连时，APP 会根据对端的地址类型选择不同的回连广播，若是 public address 或 static random

address，则直接发送 Direct Advertising。若对端采用 Resolvable Private Address，则发送 Undirected Advertising，并开启 white list 进行过滤。因为 RTL8762C 支持 LL Privacy，所以在正确配置 resolving list，并开启解析时，
可对 Resolvable Private Address 实现过滤。

另外，对于已配对过的设备，断电重新上电时，APP 会检查是否有配对信息存在，若存在则将对应的设备加入 resolving list 及 white list 中，否则作为未配对设备。

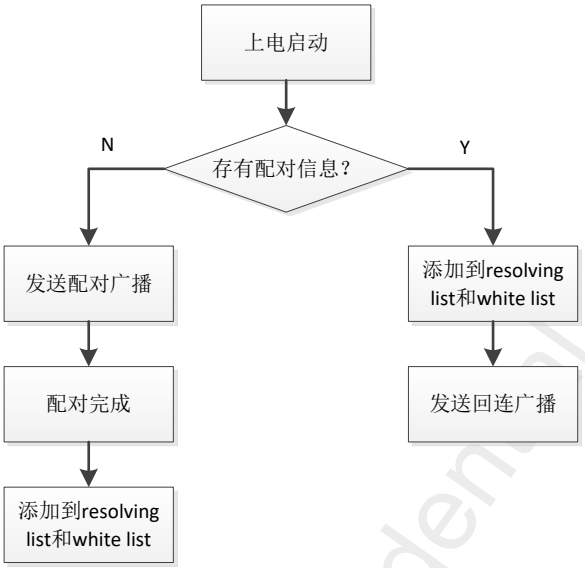

图 2.4 配对回连流程

3 蓝牙相关操作

3.1 Service 与 Characteristic

键盘应用中包含如下几个 Service:

1. HID Service：人机接口设备协议；
2. Batter Service：回报设备的电池电量，提醒更换电池，电量过低时不可以进行 OTA；
3. Device Information Service：显示设备基本信息；

各服务名称及 UUID 如表 3.1 所示。

表 3.1 包含的 Service 及 UUID 列表

Service Name	Service UUID
HID Service	0x1812
Battery Service	0x180F
Device Information Service	0x180A

3.1.1 HID Service

表 3.2 HID Service Characteristic 列表

Characteristic Name	Requirement	Characteristic UUID	Properties	Description
Protocol Mode	M	0x2A4E	Read/WriteWithoutResponse	See Protocol Mode
Report	O			
Report:Input	M	0x2A4D	Read/Write/Notify	See Report Characteristict
Report:Output	M	0x2A4D	Read/Write/ WriteWithoutResponse	See Report Characteristic
Report:Feature	M	0x2A4D	Read/Write	See Report Characteristic
Report Map	M	0x2A4B	Read	See Report Map
Boot Key Input Report	M	0x2A22	Read/Write/Notify	See Boot Key Input Report
Boot Key Output Report	M	0x2A32	Read/Write/ WriteWithoutResponse	See Boot Key Output Report
HID Information	M	0x2A4A	Read	See HID Information
HID Control Point	M	0x2A4C	WriteWithoutResponse	See HID Control Point
Report: Input(For	M	0x2A4D	Read/Write/Notify	See Report

3.1.2 Protocol Mode Characteristic

Protocol Mode Characteristic 用于暴露当前的 HID 服务的 Protocol Mode，或者设定期望的 HID 服务的 Protocol Mode。

表 3.3 Protocol Mode Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Protocol Mode	Mandatory	uint8	N/A	N/A	Enumerations

Key	Value
0	Boot Protocol Mode
1	Report Protocol Mode
2-255	Reserved for future use

3.1.3 Report Characteristic

Report Characteristic 包括了 HID 设备端和主机端传输的 Input Report，Output Report，或者 Feature Report(双向)的数据,不同的 Report 通过 Report ID 和 Report Type 进行区分。

表 3.4 Report Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Report	Mandatory	uint8	N/A	N/A	This field may be repeated

3.1.4 Report Map Characteristic

Report Map Characteristic 用于定义 HID 设备端和主机端传输 Input Report，Output Report，或者 Feature Report 数据时的格式。

表 3.5 Report Map Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Report Map	Mandatory	uint8	N/A	N/A	This field may be repeated

3.1.5 Boot Key Input Report Characteristic

Boot Key Input Report Characteristic 用于在启动协议模式下操作的 HID 主机和对应于启动键盘的 HID 服务之间传输固定格式和长度的输入报告数据。

表 3.6 Boot Mouse Input Report Characteristic Value Format

Names	Field	Format	Minimum	Maximum	Additional Information

	Requirement		Value	Value	
Boot Mouse Input Report	Mandatory	uint8	N/A	N/A	This field may be repeated

3.1.6 Boot Key Output Report Characteristic

Boot Key Output Report Characteristic 用于在启动协议模式下操作的 HID 主机和对应于启动键盘的 HID 服务之间传输固定格式和长度的输出报告数据。

表 3.7 Boot Mouse Input Report Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Boot Mouse Input Report	Mandatory	uint8	N/A	N/A	This field may be repeated

3.1.7 HID Information Characteristic

HID Information Characteristic 包含了 HID 的属性，该 Characteristic 的值是静态的，并且可以为 HID 设备和主机的绑定永久保存。

表 3.8 HID Information Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
bcdHID	Mandatory	uint16	N/A	N/A	None
bCountryCode	Mandatory	8bit	N/A	N/A	None
Flags	Mandatory	8bit	N/A	N/A	See Bit Field

表 3.9 Bit Field

Bit	Size	Name	Definition	
			Key	Value
0	1	Remote Wake	0	The device is not designed to be capable of providing wake-up signal to a HID host
			1	The device is designed to be capable of providing wake-up signal to a HID host
1	1	Normally Connectable	0	The device is not normally connectable
			1	The device is normally connectable
2	6	Reserved for future use		

3.1.8 HID Control Point Characteristic

HID Control Point Characteristic 是一个控制点属性，定义了如下 HID 命令：

1） Suspend

2） Exit Suspend

表 3.10 HID Control Point Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
HID Control Point Command	Mandatory	uint8	N/A	N/A	Enumerations Key: Value 0: Suspend 1: Exit Suspend 2-255: Reserved for future use

3.1.9 Battery Service

Battery Service 包含一个 Battery Level 的 Characteristic，如表 3.11 所示。

表 3.11 Battery Service Characteristic 列表

Characteristic Name	Requirement	Characteristic UUID	Properties	Description
Battery Level	M	0x2A19	Read/Notify	See Battery Level

3.1.10 Battery Service Characteristic

Battery Level 表示当前电量水平，范围从 0%-100%，数据格式为无符号 8 位整型，如表 3.12 所示。

表 3.12 Battery Level Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Battery Level	Mandatory	uint8	0	100	Enumerations Key: Value 101-255: Reserved

3.1.11 Device Information Service

Device Information Service 包含 9 个 Characteristic，如表 3.13 所示。

表 3.13 Device Information Service Characteristic 列表

Characteristic Name	Requirement	Characteristic UUID	Properties
Manufacturer Name String	O	0x2A29	Read
Model Number String	O	0x2A24	Read
Serial Number String	O	0x2A25	Read

Hardware Revision String	O	0x2A27	Read
Firmware Revision String	O	0x2A26	Read
Software Revision String	O	0x2A28	Read
System ID	O	0x2A23	Read
Regulatory Certification Data List	O	0x2A2A	Read
PnP ID	O	0x2A50	Read

3.1.12 Device Information Service Characteristic

Device Information Service 中包含一部分显示设备名称和固件版本等基本信息的 Characteristic，如表 3.14 所示。

表 3.14 Device Information Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Manufacturer Name	Mandatory	utf8s	N/A	N/A	None
Model Number	Mandatory	utf8s	N/A	N/A	None
Serial Number	Mandatory	utf8s	N/A	N/A	None
Hardware Revision	Mandatory	utf8s	N/A	N/A	None
Firmware Revision	Mandatory	utf8s	N/A	N/A	None
Software Revision	Mandatory	utf8s	N/A	N/A	None

1) System ID Characteristic

System ID 由两个字段组成，分别为 40bit 制造商定义的 ID 和 24bit 组织唯一标识符（OUI），如表 3.15 所示。

表 3.15 System ID Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Manufacturer Identifier	Mandatory	uint40	0	1099511627775	None
Organization Unique Identifier	Mandatory	uint24	0	16777215	None

2) IEEE 11073-20601 Regulatory Certification Data List Characteristic

IEEE 11073-20601 Regulatory Certification Data List 列举了设备依附的各种各样的管理或服从认证的项目，如

表 3.16 所示。

表 3.16 IEEE 11073-20601 Regulatory Certification Data List Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Data	Mandatory	reg-cert-data-	N/A	N/A	None

list[1]

3) PnP ID Characteristic

PnP ID 是一组用于创建唯一设备 ID 的数值，包括了 Vendor ID Source、Vendor ID、Product ID、Product Version，这些数值被用来辨别具有给定的类型/模型/版本的所有设备，如表 3.16 所示。

表 3.17 PnP ID Characteristic Value Format

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Vendor ID Source	Mandatory	uint8	1	2	See Enumerations
Vendor ID	Mandatory	uint16	N/A	N/A	None
Product ID	Mandatory	uint16	N/A	N/A	None
Product Version	Mandatory	uint16	N/A	N/A	None

表 3.18 Enumerations

Key	1	2	3-255	0
Value	Bluetooth SIG assigned Company Identifier value from the Assigned Numbers document	USB Implementer's Forum assigned Vendor ID value	Reserved for future use	Reserved for future use

3.2 Report Map 及数据发送格式

Report Map 用于定义 HID 设备端和主机端传输 Input Report，Output Report，或者 Feature Report 数据时的格式。

键盘按键分为修饰键和普通按键，8 个修饰键的按下与抬起状态分别通过 1 个 bit 来表示，普通按键则发送对应的 usage id，最大支持同时发送 6 个普通按键值。

键盘按键数据格式如下：

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Modifier Keys	Reserved	Keycode1	Keycode2	Keycode3	Keycode4	Keycode5	Keycode6

多媒体按键使用不同的 report id，有单独的发送格式。SDK 中 Hid Keyboard profile 采用了枚举的方式，列了 24 个多媒体按键。发送数据长度为 3 bytes，每一个 bit 对应一个多媒体按键。

多媒体按键数据格式如下：

Byte	Bit							
	7	6	5	4	3	2	1	0
1	Scan Next Track	Scan Previous Track	Stop	Play/Pause	Mute	Bass Boost	Loudness	Volume Increment
2	Volume Decrement	Bass Increment	Bass Decrement	Treble Increment	Treble Decrement	AL Consumer Control	AL Email Reader	AL Calculator

						Configuration		
3	AL Local Machine Browser	AC Search	AC Home	AC Back	AC Forward	AC Stop	AC Refresh	AC Bookmarks

示例 00000001 00000000 00100000 表示音量+和 Home 键按下。（对照 report map 中的枚举顺序）

```
const uint8_t hids_report_descriptor[] =
{
    0x05, 0x01,     /* USAGE_PAGE       (Generic Desktop) */
    0x09, 0x06,     /* USAGE            (Keyboard) */
    0xa1, 0x01,     /* COLLECTION       (Application) */
    0x85, HOGP_KB_REPORT_ID,    /* REPORT_ID        (3) */
    0x05, 0x07,     /* USAGE_PAGE       (Keyboard) */
    0x19, 0xe0,     /* USAGE_MINIMUM    (Keyboard Left Control) */
    0x29, 0xe7,     /* USAGE_MAXIMUM    (Keyboard Right GUI) */
    0x15, 0x00,     /* LOGICAL_MINIMUM  (0) */
    0x25, 0x01,     /* LOGICAL_MAXIMUM  (1) */
    0x75, 0x01,     /* REPORT_SIZE      (1) */
    0x95, 0x08,     /* REPORT_COUNT     (8) */
    0x81, 0x02,     /* INPUT            (Data,Var,Abs) */
    0x95, 0x01,     /* REPORT_COUNT     (1) */
    0x75, 0x08,     /* REPORT_SIZE      (8) */
    0x81, 0x01,     /* INPUT            (Cnst,Var,Abs) */
    0x95, 0x05,     /* REPORT_COUNT     (5) */
    0x75, 0x01,     /* REPORT_SIZE      (1) */
    0x05, 0x08,     /* USAGE_PAGE       (LEDs) */
    0x19, 0x01,     /* USAGE_MINIMUM    (Num Lock) */
    0x29, 0x05,     /* USAGE_MAXIMUM    (Kana) */
    0x91, 0x02,     /* OUTPUT           (Data,Var,Abs) */
    0x95, 0x01,     /* REPORT_COUNT     (1) */
    0x75, 0x03,     /* REPORT_SIZE      (3) */
    0x91, 0x01,     /* OUTPUT           (Cnst,Var,Abs) */
    0x95, 0x06,     /* REPORT_COUNT     (6) */
    0x75, 0x08,     /* REPORT_SIZE      (8) */
    0x15, 0x00,     /* LOGICAL_MINIMUM  (0) */
    0x25, 0xa4,     /* LOGICAL_MAXIMUM  (164) */    /* Can be 255 */
    0x05, 0x07,     /* USAGE_PAGE       (Keyboard) */
    0x19, 0x00,     /* USAGE_MINIMUM    (Reserved-no event indicated) */
    0x29, 0xa4,     /* USAGE_MAXIMUM    (Keyboard Application) */      /* Can be 255
*/
    0x81, 0x00,     /* INPUT            (Data,Ary,Abs) */
    0xc0,           /* END_COLLECTION */
#ifdef MULTIMEDIA_KEYBOARD
    0x05, 0x0c,     /* USAGE_PAGE       (Consumer) */
```

```
    0x09, 0x01,     /* USAGE           (Consumer Control) */
    0xa1, 0x01,     /* COLLECTION      (Application) */
    0x85, 0x04,     /* REPORT_ID       (4) */
    0x15, 0x00,     /* LOGICAL_MINIMUM  (0) */
    0x25, 0x01,     /* LOGICAL_MAXIMUM  (1) */
    0x75, 0x01,     /* REPORT_SIZE     (1) */
    0x95, 0x18,     /* REPORT_COUNT    (24) */
    0x09, 0xb5,     /* USAGE           (Scan Next Track) */
    0x09, 0xb6,     /* USAGE           (Scan Previous Track) */
    0x09, 0xb7,     /* USAGE           (Stop) */
    0x09, 0xcd,     /* USAGE           (Play/Pause) */
    0x09, 0xe2,     /* USAGE           (Mute) */
    0x09, 0xe5,     /* USAGE           (Bass Boost) */
    0x09, 0xe7,     /* USAGE           (Loudness) */
    0x09, 0xe9,     /* USAGE           (Volume Increment) */
    0x09, 0xea,     /* USAGE           (Volume Decrement) */
    0x0a, 0x52, 0x01,     /* USAGE     (Bass Increment) */
    0x0a, 0x53, 0x01,     /* USAGE     (Bass Decrement) */
    0x0a, 0x54, 0x01,     /* USAGE     (Treble Increment) */
    0x0a, 0x55, 0x01,     /* USAGE     (Treble Decrement) */
    0x0a, 0x83, 0x01,     /* USAGE     (AL Consumer Control Configuration) */
    0x0a, 0x8a, 0x01,     /* USAGE     (AL Email Reader) */
    0x0a, 0x92, 0x01,     /* USAGE     (AL Calculator) */
    0x0a, 0x94, 0x01,     /* USAGE     (AL Local Machine Browser) */
    0x0a, 0x21, 0x02,     /* USAGE     (AC Search) */
    0x0a, 0x23, 0x02,     /* USAGE     (AC Home) */
    0x0a, 0x24, 0x02,     /* USAGE     (AC Back) */
    0x0a, 0x25, 0x02,     /* USAGE     (AC Forward) */
    0x0a, 0x26, 0x02,     /* USAGE     (AC Stop) */
    0x0a, 0x27, 0x02,     /* USAGE     (AC Refresh) */
    0x0a, 0x2a, 0x02,     /* USAGE     (AC Bookmarks) */
    0x81, 0x02,     /* INPUT           (Data,Var,Abs) */
    0xc0           /* END_COLLECTION */
#endif
};

```

3.3 Advertising Data And Scan Response

对测端在与键盘建立连线前需要执行搜索操作，并从 advertising data 与 scan response data 中获取模块名称、service UUID 等相关信息。Advertising data 与 scan response data 的数据格式定义如下：

```
/** @brief  GAP - scan response data (max size = 31 bytes) */
```

```
static const uint8_t scan_rsp_data[] =
{
    0x03,                           /* length */
    GAP_ADTYPE_APPEARANCE,           /* type="Appearance" */
    LO_WORD(GAP_GATT_APPEARANCE_KEYBOARD),
    HI_WORD(GAP_GATT_APPEARANCE_KEYBOARD),
};

/** @brief  GAP - Advertisement data (max size = 31 bytes, best kept short to
conserve power) */
static const uint8_t adv_data[] =
{
    /* Flags */
    0x02,            /* length */
    GAP_ADTYPE_FLAGS, /* type="Flags" */
    GAP_ADTYPE_FLAGS_LIMITED | GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED,
    /* Service */
    0x03,            /* length */
    GAP_ADTYPE_16BIT_COMPLETE,
    0x12,
    0x18,
    /* Local name */
    0x0D,            /* length */
    GAP_ADTYPE_LOCAL_NAME_COMPLETE,
    'B', 'L', 'E', '_', 'K', 'E', 'Y', 'B', 'O', 'A', 'R', 'D'
};
```

4 参考文献

[1] IEEE Std 11073-20601 ™- 2008 Health Informatics - Personal Health Device Communication - Application Profile - Optimized Exchange Protocol - version 1.0 or later.