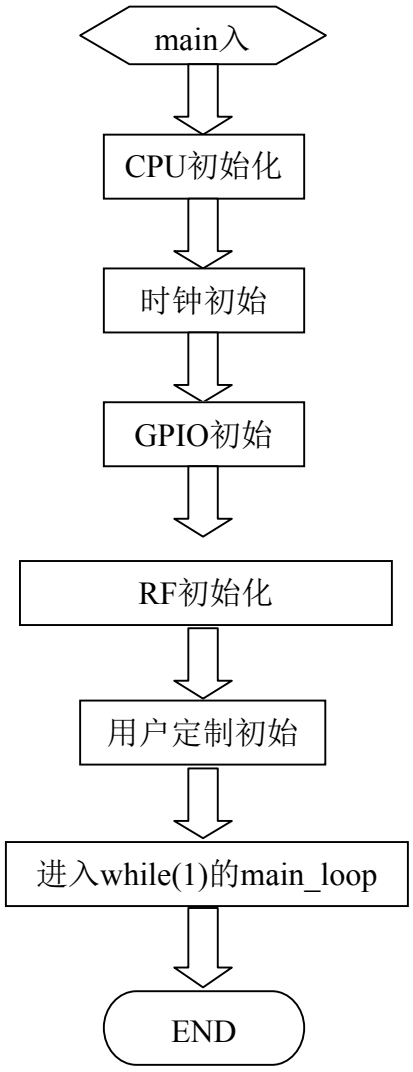
ST17H26 ble_sdk开发说明 V1

SDK 开发说明

1. 软件框架。



2. 主要参数

2.1 必设参数

(1) u8 tbl_mac [];

说明:用于 ble slave 广播数据 mac 地址(6 byte)。

示例: u8 tbl_mac [] = {0x11, 0x22, 0x33, 0x33, 0x22, 0x11};

(2) u8 tbl_adv [];

说明: 用于 ble slave 广播数据。(其中 mac address 为占位,没有意义)

示例:

```
u8 tbl_adv [] =

{0x0,      //     adv type
 0x1d,     //   length = sizeof(tbl_adv - 5)
 0xee, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5,       //mac address
```

```
 0x05, 0x09, 'a', 'n', 't', 'i',          //mac name
 0x02, 0x01, 0x05,                        //Mark
 0x03, 0x19, 0xC1, 0x03,               //Appearance
 0x07, 0x02, 0x03, 0x18, 0x03, 0x18, 0x0f, 0x18 //antiloss device
 0,0,0                //CRC预留字节3 bytes
};
```

(3) u8 tbl_rsp [];

说明：用于ble slave端scan response

示例：

```
u8 tbl_rsp [] =
{0x0,             //reserved
 0x0f,            //length = sizeof(tbl_rsp-5)
 0xef, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5,      //mac address
 0x08, 0x09, 't', 'l','_', 'a', 'n', 't', 'i' //scan name "l_anti"
 0,0,0                //CRC预留字节3 bytes
};
```

(4) const attribute_t my_Attributes[];

说明：此变量为对ble协议的初始化。其中包括各个sevice,以及character的细节部分。具体见《LENZE ble attribute table说明》文档。

3. 主要接口（BLE）

3.1 初始化

(1) void my_att_init ();

说明：用于对ble协议栈的初始化。

3.1 广播部分

(1) **blt_init** (u8 *p_mac, u8 *p_adv, u8 *p_rsp);

说明：用于ble slave端mac地址，广播数据以及scan response数据初始化。

参数1： p_mac :: mac address。

参数2： p_adv :: 广播数据

参数3： p_rsp :: scan response数据

示例：blt_init (tbl_mac, tbl_adv, tbl_rsp);

(2) void **blt_set_adv_interval** (u32 t_us);

说明：设置广播数据间隔。

参数1：t_us :: 广播数据间隔 单位us

示例：blt_set_adv_interval (30000); //广播间隔为30ms

(3) void rf_set_power_level_index (int level);

说明：设置ble slave端发数据的功率。

参数1：level :: 发包功率。Level 值选择如下：

```
enum {
        RF_POWER_8dBm = 0,
        RF_POWER_4dBm = 1,
        RF_POWER_0dBm = 2,
        RF_POWER_m4dBm    = 3,
```

```
    RF_POWER_m10dBm   = 4,
    RF_POWER_m14dBm   = 5,
    RF_POWER_m20dBm   = 6,
  RF_POWER_m24dBm   = 8,
  RF_POWER_m28dBm   = 9,
  RF_POWER_m30dBm   = 10,
  RF_POWER_m37dBm   = 11,
  RF_POWER_OFF   = 16,
  };
```

示例：rf_set_power_level_index (RF_POWER_8dBm);
设置ble slave端发包功率为8dbm.

(4) **blt_send_adv** (int mask);

说明：根据参数mask(广播数据channel),发送广播数据。正常情况下，广播数据channel 为37,38,39。 而在调试时，可以使用1个channel,便于抓包分析。

参数1：mask :: 广播数据channel值。Mask值选择如下：

```
#define          BLT_ENABLE_ADV_37        BIT(0)
#define          BLT_ENABLE_ADV_38        BIT(1)
#define          BLT_ENABLE_ADV_39        BIT(2)
#define          BLT_ENABLE_ADV_ALL
                 (BLT_ENABLE_ADV_37 | BLT_ENABLE_ADV_38 |
                 BLT_ENABLE_ADV_39)
```

示例：blt_send_adv (BLT_ENABLE_ADV_38);//只在38 channel上广播
blt_send_adv (BLT_ENABLE_ADV_39);//只在39 channel上广播
blt_send_adv (BLT_ENABLE_ADV_ALL);//在37/38/39 channel上广播

3.2 连接部分

(1) blt_brx ();

说明：此函数用来处理ble slave与ble master每一次通信时收发包。一般情况下不用改变。

(2)更新连接参数接口

说明：一般情况下，ble slave要更新ble的连接参数，需要先向master端发送update parameter request, ble master会根据slave端请求的参数选择合适的连接的参数，并且给ble slave端发送连接参数更新命令并且确认回response 包,或者在master认为slave的参数不合适，会回拒绝参数更新。

在LENZE ble sdk中更新参数一版步骤为(1)调用以下函数a) (2)调用以下函数b)

a) **void blt_conn_para_request** (u16 min_interval, u16 max_interval, u16 latency, u16 timeout

说明:此函数用在ble slave端需要更新参数当前连接参数之前。但是此函数不包含发送更新参数命令。Slave 与master更新的连接间隔介于Min interval 与max interval 之间。

参数1：min interval :: 表示ble slave 需要更新参数（连接通信间隔）的最小阈值。

参数2:max interval :: 表示ble slave 需要更新参数的最大阈值。

参数3： latency :: 表示ble slave需要更新参数的latency

参数4： timeout :: 表示slave 与master端多久未连接表示端口断开

连接。

b) **void blt_update_parameter_request** ();

说明：此函数用于ble slave想ble master 发送更新参数命令。

3.3 其他

(1) u8 **blt_push_notify** (u16 handle, u32 val, **int** len);

说明：ble slave端向ble master端notify数据。一般情况下数据size小于等于4时，会用到此函数。

参数1：handle :: ble slave端通过某个handle向master发数据。其handle 参照变量my_Attributes的设置。

参数2：val ::发送数据值

参数3：len ::val值的大小

示例：blt_push_notify (15,3,1);//通过handle:15 notify 字节数为1的数据3。

(2) u8 **blt_push_notify_data** (u16 handle, u8 *p, **int** len);

说明：ble slave端向ble master端notify数据。一般情况下，当数据的小大于4时，会用此函数发送数据。

参数1：handle :: ble slave端通过某个handle向master发数据。其handle 参照变量my_Attributes的设置。

参数2：p ::要发送数据的指针。

参数3：len ::要发送数据的size

示例：

```
u8 buffer[8];
blt_push_notify_data(15,buffer,8);//通过handle:15 notify 字节数为8的数据,此数据存在buffer开始的指针开始的地方。
```

(3) u8 **blt_enable_suspend** (u8 en);

说明：设置ble 连接过程中的suspend使能状态。

参数1：en :: 此参数广播或者连接过程中的使能状态。en值可共选择值如下：

```
#define          SUSPEND_ADV             BIT(0)
#define          SUSPEND_CONN            BIT(1)
```

示例：

```
blt_enable_suspend(SUSPEND_ADV);//此系统仅在广播状态下会进入suspend状态。
blt_enable_suspend(SUSPEND_CONN);//此系统仅在连接状态下会进入suspend状态。
blt_enable_suspend(SUSPEND_CONN | SUSPEND_ADV);//此系统在连接或者广播状态下状态下均会进入suspend状态。
blt_enable_suspend(0);//此系统在任何状态下都不会进入suspend状态。
```

3.4 加密部分

说明：一般情况ble slave端包含HID(Human Interface Device)service时，会需要加密。另外，某些手机在原生蓝牙中，也会要求加密（即ble master会主动发出smp paring request）。

(1) blt_smp_func_init ();

说明：在数据需要加密的情况下，需要调用此函数。

(2) blt_smp_store_in_ram_enable ();

说明：数据加密信息存在 ram 中，掉电丢失。

3.5 回调函数

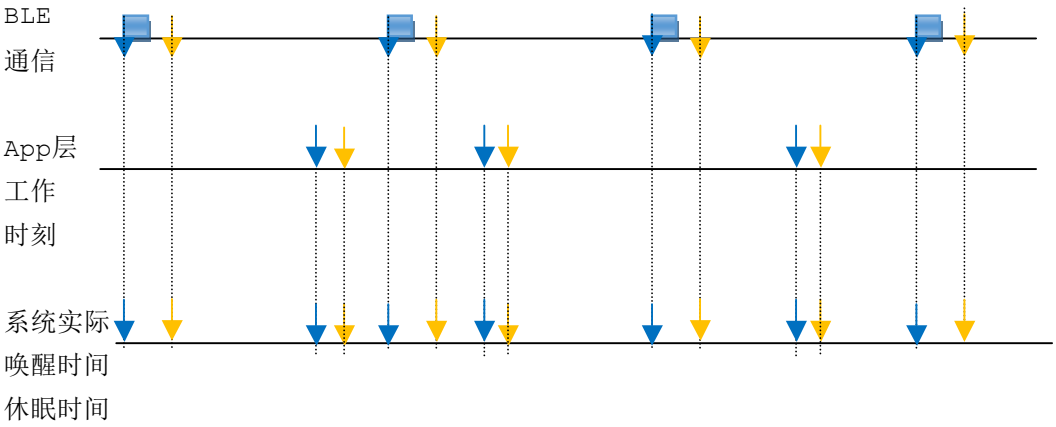
说明：LENZE ble sdk系统函数一系列的回调函数，用于ble连接过程中比较重要的时间点给出的标志。详情见《ble 回调函数讲解》

3.6 休眠函数

(1) **void** blt_brx_sleep (u32 app_wakeup_tick);

说明：此函数用于处理ble连接过程的idle状态。默认状态下，退出此函数函数的时间为下一次ble通信的时间。

参数1：app_wakeup_tick :: 用户自定义的退出此函数的时间。此参数为是时刻值，而不是时间段值。



：唤醒时刻

：休眠时刻

进入低功耗函数：

(1)**int cpu_sleep_wakeup** (**int** deepsleep, **int** wakeup_src, u32 wakeup_tick);

说明：此函数用于进入系统低功耗状态以及设置唤醒源。一旦调用此函数，则系统进入低功耗状态

参数：deepsleep :: 0 → 表示进入suspend mode. 1→表示deepsleep mode.

Wakeup_src :: 表示唤醒源。值选择如下：

PM_WAKEUP_CORE , // 表示数字部分唤醒。（比如gpio）用于suspend

PM_WAKEUP_TIMER ,// 表示定时唤醒,用于suspend mode and deepsleep

PM_WAKEUP_PAD , //用于deepsleep mode下gpio 唤醒。

Wakeup_tick ::用于定时唤醒时，时间设置。该时间为一个时间点。

示例：

a. **cpu_sleep_wakeup** (0, *PM_WAKEUP_TIMER*, next_wakeup_tick)

以上调用之后，系统将会进入一个 suspend mode 状态。可以用 timer 唤醒，唤醒的时刻为 next_wakeup_tick。注意：next_wakeup_tick 必须为是未来的时间(当前系统时间之后的一个时间点)。

b. **cpu_sleep_wakeup** (0, *PM_WAKEUP_CORE*, next_wakeup_tick)

以上调用之后，系统会进入 suspend mode 状态。此状态只可以通过外部的 io 状态改变唤醒。注意：即使此时 next_wakeup_tick 值不为 0，也不会通过 timer 唤醒。如果使用 IO 唤醒，另有 IO 口的其他配置

c. **cpu_sleep_wakeup** (1, *PM_WAKEUP_TIMER*, next_wakeup_tick)

以上调用之后，系统会进入 deepsleep mode 状态。此状态可以通过 timer 唤醒，同 suspend timer 唤醒机制。

d. **cpu_sleep_wakeup** (1, *PM_WAKEUP_PAD*, next_wakeup_tick)

以上调用之后，系统会进入 deepsleep mode 状态。此状态可以通过 io 唤醒。如果使用 IO 唤醒，另有 IO 口的其他配置

e. **cpu_sleep_wakeup** (1, *PM_WAKEUP_PAD* | *PM_WAKEUP_TIMER*, next_wakeup_tick)

以上调用之后，系统会进入一个 deepsleep mode 状态。可以通过 io 唤醒，也可以通过 timer 唤醒。

(2) **void blt_set_adv_type**(u8 type);

说明：设置广播type.设置之后，如在广播模式下，则立即生效。

参数：type :: 广播类型。

广播类型为一下类型之一：

```
EVENT_PKT_ADV = 0,    //connected undirected advertising type
EVENT_PKT_CDA = 1,    //connected directed advertising type
EVENT_PKT_NUA = 2,    //non-connected undirected advertising
EVENT_PKT_DISCV = 6,//connected discovery advertising type
```

示例：**blt_set_adv_type**(EVENT_PKT_ADV);

调用此函数后，则广播类型为connected undirected advertising