FAQ

在学习 BLUENRG-LP 的过程中，及时地把碰到的问题记录下来，并作解答，形成本 FAQ (frequently asked questions）文档

BLUENRG-LP 的特性

芯片 UUID

可以从 0x100007F4 和 0x100007F8 位置读出 8 bytes数据，其中两个字节为：0xAA, 0x55，其余的 6 个字节可保证唯一。

若想将此 UUID 作为蓝牙 MAC，由于

芯片内部有哪些存储介质，各的功能是什么?

- FLASH 存放固件和用户数据。支持读、写保护
- OTP (1KB) 存放用户数据，支持读、写保护
- ROM (7KB) 不对用户开放。存放 UART BOOTLOADER 固件和一些芯片的配置信息，如ADC校准值

BLE 5.0 的新特性及其作用：

2Mbps：更高数据吞吐率

LE 2M PHY，比特率为 2Mb/s，不支持 ECC；减少冗余度以提高传输吞吐率。

> 关于 1M PHY:
>
> 蓝牙 4.2 采用 LE 1M PHY，即比特率为 1Mb/s 的 PHY；必选。
>
> 支持 ECC (error correcting coding，可选)
>
> 根据编码方式的不同，支持 3 种比特率：1Mb/s，500Mb/s (LE Coded)，125Kb/s (LE Coded)，其实就是冗余度的增加导致了比特率的下降，带来的好处就是能对空中的数据进行纠错从而可以传的更远

Long range：更远的通讯距离

Channel selection algorithm #2：，通道选择算法 2，可避免干扰和多路径衰落效应

参考文章：https://blog.51cto.com/11134889/2317010

GATT caching：

以下是网友总结：

新增功能	协议层	描述
Slot Availability Mask (SAM)	物理层	标记设备的收发时间块。其他蓝牙设备获取该信息即可避免在该设备忙碌时争抢信道资源，这有利于在信道资源有限的情况下维持多设备同时工作，这个特性仅适用于经典蓝牙。
2 Mbit/s PHY for LE	物理层	2Mbit/s比特率的物理层。以前的物理层都是1Mbit/s比特率，这个特性从物理层提升了传输速率。
LE Long Range	物理层	该特性由多个方面共同实现。最高20dBm的发射功率（以前是10dBm），编码型物理层最低-82dBm接收灵敏度（以前是-70dBm），8位前向纠错编码FEC（以前没有），这些新特性共同实现了更远的通信距离，官方宣称能比过去提高4倍传输距离，实际测试高达750米。
High Duty Cycle Non-Connectable Advertising	链路层	在协议文档中未找到针对该特性的描述，从网络上也未获取有用信息。根据一个已有的类似概念“High Duty Cycle Connectable Advertising”，可以推测这个广播事件类型就是广播间隔更短，并且限制广播总时间，超时后将停止广播。
LE Advertising Extensions	链路层	扩展广播。以前广播仅仅使用37/38/39三个广播信道，现在可以使用扩展广播包，在0-36数据信道上传输，以前广播数据最大为31字节，现在扩展广播的数据长度最大可达255字节，所以官方宣称广播数据容量扩展8倍。
LE Channel Selection Algorithm #2	物理层	一种新的跳频算法。以前的算法仅用于连接数据的跳频，现在数据信道支持传输广播数据，以前的跳频算会产生性能问题，新的跳频算法可以用于连接数据通信的跳频和周期广播数据的跳频。

BLE 128kbps 的 long range 模式，有效数据的最大数据吞吐率是?

long range 模式仍然使用的是 1Msymbol/s 的 PHY。8 个 symbols 代表一个 bit，因此是 128Kbps 的比特率。有效数据的吞吐率大概是比特率的 70% ~ 80%

BLE 协议栈的版本能否更新?

BLUENRG-LP SDK 包中包含了协议栈固件库，并支持对该协议栈库进行函数级别的定制（尽可能让出更多的用户固件空间）。

即，协议栈可定制、升级

协议栈存储在芯片的什么位置?

存放在 FLASH。

协议栈以库的形式提供给用户工程，随同用户固件一起编译、链接。

有一些芯片厂商的协议栈代码是固化在芯片内部的，不可修改；相对于此，BLUENRG-LP 的形式更加灵活且可随蓝牙版本的升级而更新。

硬件设计

IO 口的选择注意事项

RF 设计注意事项

晶振选型、Layout 注意事项，对信号的影响，系统稳定性（高频干扰）等

竞争精度要求：< ±50ppm；射频偏移量 < ±50KHz

电感选型、Layout 注意事项，能源转换效率？哪一个供电电压效率最高？

电容选型、Layout 注意事项

低功耗控制：通过通、断电控制？通过使能脚控制？有哪些影响？

开发环境

调试目标板

快速上手 Windows Keil 固件开发环境

有哪些开发工具？

选型（电流评估、资源评估、成本评估）

开发（编译工具、调试工具）

生产（烧录工具、测试工具）

第一次使用 Keil 开发环境的配置？

SDK

定时执行任务？

可通过系统提供的虚拟定时器来实现定时执行某个任务：HAL_VTIMER_StartTimerMs

中断服务函数在哪?

bluenrg_lp_it.c 文件集中放置了各个中断服务函数，包括串口中断、唤醒 IO 中断，BLE 射频中断等

使能 Notify

客户端使能服务器的 notify 后，协议栈回调：aci_gatt_srv_attribute_modified_event 接口通知应用层

接收数据则是：aci_gatt_srv_write_event

获取 Long Term Key

当在 SDK 中启用了 bonding 功能后，协议栈会把配对过程产生的一些信息保存在 FLASH 的 NVM 区域。以下代码演示了将这些信息读出来的过程（其中就包含 LTK）：

```
#define NVM_BOND_INFO_ADDR   (0x1007F000u) // NVM 的起始地址，根据实际情况进行修改

#pragma pack(1) // 按字节对齐

typedef struct bond_info_s
{
    uint8_t nvm_record_header[4];
    uint8_t reserved_0[4];
    uint8_t ediv[2];
    uint8_t rand[8];
    uint8_t ltk[16];
    uint8_t irk[16];
    uint8_t csrk[16];
    uint8_t peer_addr[6];
    uint8_t reserved_1[12];
}bond_info_t;

#pragma pack() // 恢复默认的对齐方式

void print_bond_info(void)
{
    bond_info_t *p_bond_info = (bond_info_t *)NVM_BOND_INFO_ADDR;

    PRINTF("bond information: \r\n");

    PRINTF("ediv:     : ");
    for (uint8_t i = 0; i < 2; i++)
    {
        PRINTF("%02x ", p_bond_info->ediv[i]);
    }
    PRINTF("\r\n");

    PRINTF("rand      : ");
    for (uint8_t i = 0; i < 8; i++)
    {
        PRINTF("%02x ", p_bond_info->rand[i]);
    }
    PRINTF("\r\n");

    PRINTF("ltk       : ");
```

```
    for (uint8_t i = 0; i < 16; i++)
    {
        PRINTF("%02x ", p_bond_info->ltk[i]);
    }
    PRINTF("\r\n");

    PRINTF("irk       : ");
    for (uint8_t i = 0; i < 16; i++)
    {
        PRINTF("%02x ", p_bond_info->irk[i]);
    }
    PRINTF("\r\n");

    PRINTF("csrk      : ");
    for (uint8_t i = 0; i < 16; i++)
    {
        PRINTF("%02x ", p_bond_info->csrk[i]);
    }
    PRINTF("\r\n");

    PRINTF("peer addr : ");
    for (uint8_t i = 0; i < 6; i++)
    {
        PRINTF("%02x ", p_bond_info->peer_addr[i]);
    }
    PRINTF("\r\n");
}
```

以上 print_bond_info() 函数，可在配对成功后调用，如：

```
/* BLE Security v4.2 is supported: BLE stack FW version >= 2.x */
void aci_gap_pairing_complete_event(uint16_t Connection_Handle,
                                    uint8_t Status,
                                    uint8_t Reason)
{
    if (Status == BLE_STATUS_SUCCESS)
    {
        print_bond_info();
    }
}
```

> 注意。获取 LTK 的方式，上文已经提供了示例代码，用户只需要将示例代码引用到自己的工程里即可。对于 BLUENRG-1/2 的 SDK 或 BLUENRG-LP 的一些工程，需要依据以下内容进行设置。
>
> 1. 在 SDK 中使能绑定功能，以使协议栈将配对信息保存在 FLASH 里（否则不会保存）：
>
>    ```
>    #define SLAVE_BONDING_USAGE      BONDING
>    ```
>
> 2. 编译 SDK 的 security 工程，并打开 Keil 的 MAP 文件，查看 NVM 的起始地址：
>
>    ```
>    Execution Region REGION_FLASH_NVM (Exec base: 0x1007f000, Load base:
>    0x10060c1c, Size: 0x00001000, Max: 0x00001000, ABSOLUTE, UNINIT)
>    ```
>
>   可见：0x1007F000。

> 对于不同的 SDK 工程，此 NVM 的起始地址可能会不同，用户需要根据自己的工程、对上文提供的 获取 LTK 的示例代码进行修改，即修改以下宏
>
> ```
> #define NVM_BOND_INFO_ADDR (0x1007F000u) // NVM 的起始地址，根据实际情况进行修改
> ```

timer module 是什么？有什么作用?

timer module 是一个软件模块。

该软件模块管理了芯片内部链路控制器硬件的各个定时器，关于这些定时器的详细介绍，可在射频控制器参考手册里找到。

timer module 根据抽象程度的不同分为了两个层次 （HAL, LL）。可为设备的唤醒、用户超时触发和预配置的射频事务触发关联一个事件；举个例子，用户可编程一个事件，实现：

- 定时唤醒休眠的系统
- 或产生一个超时事件
- 或为蓝牙事件提供时间依据

timer module 的源码实现包含以下文件：

> bluenrg_lp_hal_vtimer.h
>
> bluenrg_lp_hal_vtimer.c
>
> bluenrg_lp_ll_timer.h
>
> bluenrg_lp_ll_timer.c

ll 层比较靠近硬件，主要实现了对硬件定时器的编程、低频时钟的管理和定时时间的转换（将 hal 的时间值转化为硬件定时器寄存器的值）

hal 层是对 ll 层的进一步抽象，封装了硬件的细节。主要实现了虚拟定时器序列，回调管理，校准调度，射频事件调度等。

虚定时器

链路控制器计数器。链路控制器中有一个计数器，定时器模块可利用该硬件计数器来虚拟出多个软件定时器。

时间基准。虚拟定时器的有一个特定的单位，称为系统时间单位（STU system time unit）。一个 STU = 625/256 us。在对真正的计数器进行编程之前，需要将用 STUs 表示的时间转换到硬件计时器计数单元中。

校准间隔。是一个参数，可以在初始化阶段设置，以决定该设备需要多长时间对内部振荡器的频率进行测量；当计数器使用的时钟源是外部晶振时，则不需要该参数。

射频计数器

BLUENRG-LP 提供了另外一个定时器专门用于触发射频事务（transaction）。该事务可以是一次射频发送、或射频接收。

休眠管理

定时器模块可避免系统在以下条件下进入休眠：

- 虚拟定时器已经触发但回调函数还在执行
- 低频时钟检测流程正在进行
- 下一个射频事务已经很接近要触发了
- The device is in a back-to-back communication

配对模式，用户层如何给底层提供自定义的 passkey?

多连

ST 的芯片在多连（一拖多，主从一体）特性的支持上有优化，达到行业领先水平。BLUENRG-1/-2 关于此的实现相关资料如下：

https://www.st.com/resource/en/design_tip/dm00518102-slot-allocation-and-multiple-connection-timing-strategy-for-bluenrg-bluenrgms-bluenrg1-and-bluenrg2-stmicroelectronics.pdf

休眠流程

休眠具体流程是?

休眠时，IO口的电平状态是?

唤醒源有哪些，分别是什么?

DEEPSTOP 模式下，唤醒源可能是：

- 通过运行在低频时钟的内部唤醒定时器，射频模块能产生两种事件来唤醒系统：
  - Radio wake-up time is reached
  - CPU host wake-up time is reached
- RTC 事件
- IWDG 产生的复位事件
- 多达 28 个的 GPIO 事件（PA0~15, PB0~11）

SHUTDOWN 模式下，只能通过拉高拉低复位引脚来唤醒系统

休眠时，是否所有的 IO 口都唤醒系统?

在大多数口（非全部）上可以：PA0~15, PB0~11

启动流程

启动流程是？支持哪些接口启动?

芯片上电（或复位）后，芯片内部的 ROM 代码会先运行，

当检测到 PA10 脚为低电平时，会直接跑进 FLASH 开始运行 用户固件（运行失败会停在 while(1) ）。

当检测到 PA10 脚为高电平时，会运行 UART BOOTLOADER 流程。

UART BOOTLOADER 可通过串口接收用户指令，执行一系列的操作。其特性为：

- 自动检测波特率，范围：500 – 460800
- 只能在指定的引脚：UART RX = PA8，UART TX = PA9
- 用户需要先以某个波特率给 BLUENRG-LP 发送 0x7F，当接收到 0x79 的回复时，说明成功进入 UART BOOTLOADER 流程，可开始发送指令

UART BOOTLOADER 详细的介绍，参照文档：AN5471: The BlueNRG-LP UART bootloader protocol

安全启动、固件加密?

空中升级

OTA具体流程?