

# FR801xH 系列芯片 FAQ

Bluetooth Low Energy SOC

[www.freqchip.com](http://www.freqchip.com)



## 目录

1.	概述 .....	3
2.	工具以及开发环境使用 .....	4
2.1.	Fr801xH 的芯片是否有入门指南，简单概况芯片的开发和使用? .....	4
2.2.	如何使用 SWD 进行 JLINK 仿真和使用 UART 串口打印? .....	4
2.3.	如何使用 PC 工具进行程序下载 .....	4
2.4.	如何使用手机 app 进行 OTA 在线升级 .....	4
2.5.	如何使用烧录板进行工厂模式量产烧录 .....	4
2.6.	Jlink 在线调试功能为什么不能使用 .....	4
3.	外设驱动 .....	5
3.1.	Fr801xH 系列芯片 GPIO 默认状态 .....	5
3.2.	Fr801xH 芯片如何开始/停止保持链接的休眠和进入完全的 powerdown 模式 .....	5
3.3.	Fr801xH 系列芯片 IO 口驱动能力说明 .....	6
3.4.	如何获取外设驱动代码和驱动使用的示例代码 .....	6
3.5.	Fr801xH 芯片的 LED1、LED2 管脚能否用作普通 GPIO，能否用作 pwm 输出? .....	6
3.6.	芯片处于掉电模式下，电源低电压报警能否唤醒芯片? .....	6
4.	协议栈说明 .....	6
4.1.	Fr801xH 芯片的蓝牙事件处理回调函数中各个分支代表着怎样的事件？在什么时候发生? .....	6
4.2.	如何在链接状态下获取 RSSI 信号质量? .....	7
4.3.	可否调用接口获取真随机数? .....	7
4.4.	Fr801xH 芯片如何进行多连接的操作? .....	7
4.5.	是否有 hid 键盘和鼠标的示例工程代码? .....	8
4.6.	Fr801xH 系列芯片支持 5.0 的各种广播类型吗? .....	8
4.7.	有没有禁止和使能 latency 的接口函数 .....	8
4.8.	如何使用 ANCS 的 profile .....	8
4.9.	Fr801xH 是否支持扫描以及使用方法 .....	9
5.	电路以及电气参数 .....	9
5.1.	Fr801xH 系列芯片的参考外围设计电路在哪里获取? .....	9
5.2.	Fr801xH 芯片如何进行频偏和天线匹配校准? .....	9
5.3.	Fr801xH 芯片调试功耗时如何测试底电流? .....	9
5.4.	Fr801xH 芯片的运行供电电压范围是多少? .....	9
5.5.	Fr801xH 芯片的 IO 高电平是的电压范围是多少? .....	9
5.6.	该如何选择合适的发射功率 .....	10
5.7.	如何切换 CPU 的工作频率 .....	10
5.8.	如何在项目中配置和使用充电功能 .....	10
5.9.	做 IEC 标准的 ESD 测试时，产品设计有哪些注意事项 .....	11
6.	版本历史 .....	12

## 1. 概述

该文章总结了 Fr801xH 芯片开发和生产阶段客户提到的问题及其解答。

主要分为如下几个章节：

1. 工具以及开发环境使用
2. 外设驱动
3. 协议栈说明
4. 电路以及电器参数

本文为“Fr801xH 系列芯片 FAQ”针对的是 Fr801xH 系列芯片都存在的特性和使用方法，并不针对某个特定的芯片。虽然每个芯片都有这样的特性，但是实现方式上会略有区别，这里可能只是以某个芯片来概括说明，具体到某个芯片还要灵活运用。一般而言对于已经存在的芯片如果其他芯片的实现方式不一样，本文会提及。

## 2. 工具以及开发环境使用

### 2.1. Fr801xH 的芯片是否有入门指南，简单概况芯片的开发和使用？

A: 请看文章《Fr801xH 快速入门.pdf》和《Fr801xH 如何构建系统.pdf》，这两篇文章简单介绍了 Fr801xH 芯片的开发过程以及 SDK 的使用方法，该文章可以从 Fr801xH 的 SDK\docs\Application Notes 上拿到。

### 2.2. 如何使用 SWD 进行 JLINK 仿真和使用 UART 串口打印？

A: 请看文章《Fr801xH 快速入门.pdf》里面第 7 节和第 8.2 节，简单介绍了 Fr801xH 芯片如何查看串口 log 和使用 jlink 进行下载调试，该文章可以从 Fr801xH 的 SDK\docs\Application Notes 上拿到。

### 2.3. 如何使用 PC 工具进行程序下载

A: Fr801xH 的 PC 工具采用串口下载，具体的操作步骤参见《FREQ BLE SDK User Guide.pdf》内部 1.6.2 节。该文章可以从 Fr801xH 的 SDK\docs/上拿到。

### 2.4. 如何使用手机 app 进行 OTA 在线升级

A: 第一步，在手机上安装 Fr801xH 的 OTA 升级 app 软件；第二步，在工程中引用 ota service profile 的文件，文件在 sdk\components\ble\profiles\ble\_ota 内；第三步，在工程初始化时，调用 ota\_gatt\_add\_service() 添加 OTA 的服务。第四步，打开手机 OTA 的 app 软件，链接设备后，选择要下载的 bin 文件，开始进行 OTA 下载。

### 2.5. 如何使用烧录板进行工厂模式量产烧录

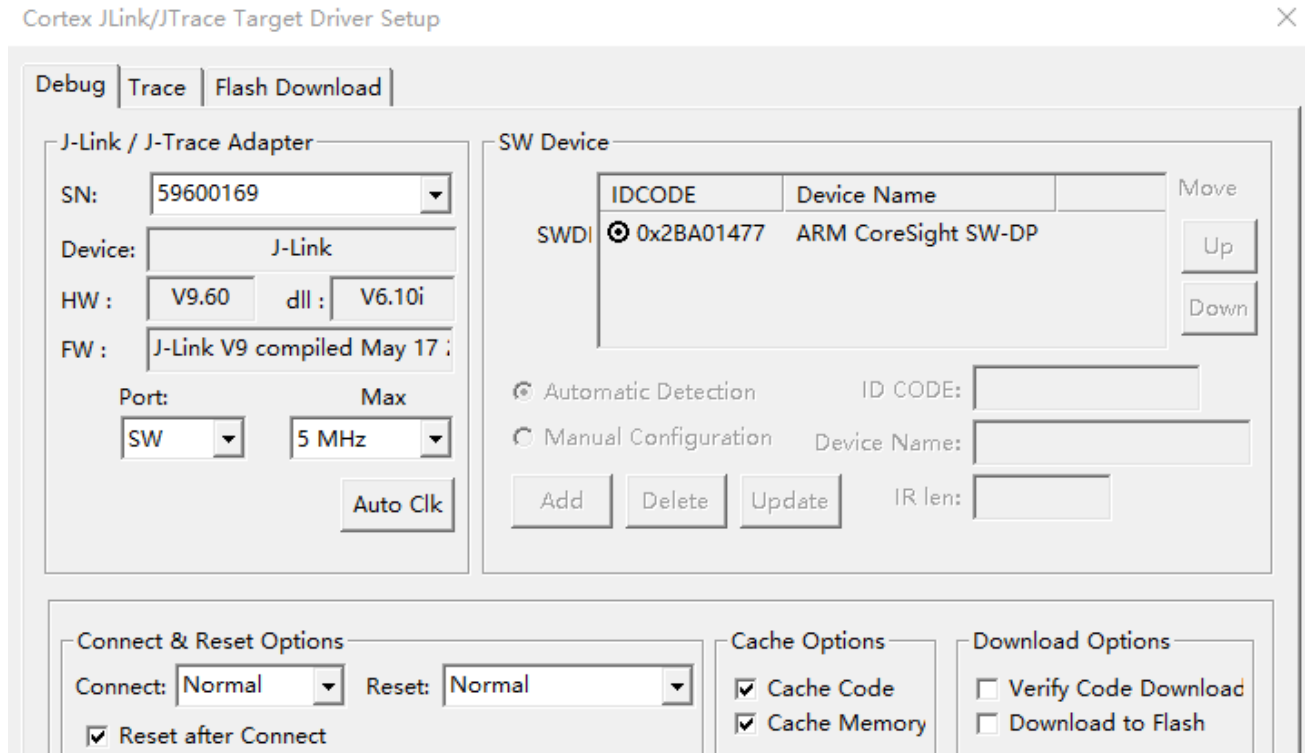
A: Fr801xH 系列芯片量产烧录工具，可以支持烧录螺片，也可烧录 PCBA，具体实施方式可以联系代理商或应用工程师。

一般烧录步骤如下：

- 1 插上 USB 线，接到 PC 机上，PC 上会识别出一个 U 盘，把要烧录的 bin 文件，放入 U 盘。
- 2 拔插 USB 线，让烧录板识别要烧录的 bin 文件。
- 3 将烧录板的右侧插针中的 VBAT/GND/TXD/RXD 分别接到被烧录芯片的 VBAT/GND/PA3/PA2 管脚。
- 4 测试烧录，按下做下方的按钮，开始。工厂机台需要发送 100ms 高电压脉冲信号到下面排针中的 start 脚。
- 5 烧录结束后，如果烧录成功，亮绿灯，同时下面排针中的 OK 脚出现 100ms 高电平脉冲。烧录失败，亮红灯，同时下面排针中的 NG 脚出现 100ms 高电平脉冲。

### 2.6. Jlink 在线调试功能为什么不能使用

A: 首先在 keil 工程的 Options for Target 页面，Debug 页面检查 Jlink 是否找到 arm 设备。如下图。



然后在 patch.c 屏蔽掉如下两个 patch 入口点。

```

99  #if 0
100      [4] = {
101          .patch_pc = 0x00004aae,
102          .replace_function = frspim_rd,
103      },
104      [3] = {
105          .patch_pc = 0x00004b02,
106          .replace_function = frspim_wr,
107      },
108  #endif

```

调试完毕后，正常代码不能屏蔽以上两个 patch 入口点。

## 3. 外设驱动

### 3.1. Fr801xH 系列芯片 GPIO 默认状态

A: Fr801xH 系列芯片所有的 GPIO 管脚除了 PA2 和 PA3 外上电默认都处于低电压。芯片的 Rom Code 将 PA2, PA3 默认初始化为 UART1 串口的 RXD 脚和 TXD 脚（波特率 115200）。并且 PA3 脚会打印一次 freq 的字符

### 3.2. Fr801xH 芯片如何开始/停止保持链接的休眠和进入完全的 powerdown 模式

A: 芯片的 sdk 自带的 lib 库默认会开启可以保持链接的休眠，在无广播和无链接的情况下，芯片每隔 10 钟会自动唤醒一次，然后继续休眠。开启广播，保持链接，或者启动了软件定时器的情况下，芯片的唤醒时间由这三个事件的执行时间决定。

在项目中调用 `system_sleep_disable()`; 停止保持链接的休眠。调用 `system_sleep_enable()`; 开始保持链接的休眠。

在项目中调用 `void system_power_off(bool aldo_bypass)` 后系统进入 `powerdown` 模式，功耗在 2uA。在进入 `powerdown` 模式之前，设置了 `gpio` 按键唤醒的 `gpio` 能唤醒。

### 3.3. Fr801xH 系列芯片 IO 口驱动能力说明

A: Fr801xH 芯片的 IO 口在输出高电平是默认的最大驱动电流为 12mA。使用芯片 ALDO 管脚做为电源输出时，ALDO 管脚的最大驱动电流为 100mA。

### 3.4. 如何获取外设驱动代码和驱动使用的示例代码

A: 芯片的外围设备大部分的驱动文件在 `SDK\components\driver` 内，驱动的使用示例代码在 `SDK\examples\none_evm\ble_drivers_demo` 示例工程内。

### 3.5. Fr801xH 芯片的 LED1、LED2 管脚能否用作普通 GPIO，能否用作 pwm 输出？

A: 芯片的 LED1、LED2 管脚可以在管脚紧张的情况下，用作普通的 GPIO 输出，或 pwm 的输出，但不能做为输入使用。具体的使用示例代码可以参考 `SDK\examples\none_evm\ble_drivers_demo` 示例工程里 `led` 部分。

### 3.6. 芯片处于掉电模式下，电源低电压报警能否唤醒芯片？

A: 低电压报警属于 `pmu` 的逻辑模块，在芯片处于掉电模式时，是持续工作的，这种情况下，低电压报警可以通过 `pmu` 中断唤醒芯片。低电压报警的最高电压为 2.5V，不适用于锂电池供电场景。

## 4. 协议栈说明

### 4.1. Fr801xH 芯片的蓝牙事件处理回调函数中各个分支代表着怎样的事件？在什么时候发生？

A: 要获取协议栈底层的蓝牙事件，首先要调用函数 `gap_set_cb_func(proj_ble_gap_evt_func)`; 设置一个接收事件的回调函数。协议栈如果监听到底层蓝牙的状态发生变化时，将通过回调函数通知应用层，传回的变量为 `gap_event_t` 类型的结构体。通过查看结构体的 `type` 元素获得事件类型。

```
typedef enum
{
    GAP_EVT_ALL_SVC_ADDED,          ///< All GATT service have been added

    GAP_EVT_SLAVE_CONNECT,         ///< Connected as slave role
    GAP_EVT_MASTER_CONNECT,        ///< Connected as master role
    GAP_EVT_DISCONNECT,            ///< Disconnected
    GAP_EVT_LINK_PARAM_REJECT,     ///< Parameter update rejected
    GAP_EVT_LINK_PARAM_UPDATE,     ///< Parameter update successful
    GAP_EVT_PHY_REJECT,            ///< Parameter update rejected
    GAP_EVT_PHY_UPDATE,            ///< PHY update indication
    GAP_EVT_ADV_END,               ///< Advertising ended
    GAP_EVT_SCAN_END,              ///< Scanning ended
    GAP_EVT_PER_SYNC_ESTABLISHED,  ///< Per_sync is established
    GAP_EVT_PER_SYNC_END,          ///< Periodic adv sync event ended
    GAP_EVT_ADV_REPORT,            ///< Scan result report
    GAP_EVT_CONN_END,              ///< Connection action is ended
    GAP_EVT_PEER_FEATURE,          ///< Got peer device supported features
    GAP_EVT_MTU,                   ///< MTU exchange event
    GAP_EVT_LINK_RSSI,             ///< Got RSSI value of link peer device

    GAP_SEC_EVT_MASTER_AUTH_REQ,   ///< As master role, got authentication request from slave
    GAP_SEC_EVT_MASTER_ENCRYPT,     ///< Link is encrypted as master role
    GAP_SEC_EVT_SLAVE_ENCRYPT,      ///< Link is encrypted as slave role
} gap_event_type_t;
```

## 4.2. 如何在链接状态下获取 RSSI 信号质量？

A: 第一步在链接建立的时间回调内，使能 rssi\_report 功能。例如在协议栈事件回调函数内分支：连接成功 运行如下代码：

```
case GAP_EVT_SLAVE_CONNECT:
{
    gap_set_link_rssi_report(true);
    gap_get_link_rssi(event->param.slave_connect.conidx);
```

调用一次查询函数 void gap\_get\_link\_rssi(uint8\_t conidx)，底层就会上报一次当前链接对端设备的 rssi 值。该值通过协议栈事件回调函数分支：得到 rssi 值 来获取。如下代码所示：

```
case GAP_EVT_LINK_RSSI:
    co_printf("link rssi %d\r\n",event->param.link_rssi);
    break;
```

## 4.3. 可否调用接口获取真随机数？

A: 可以调用接口获取真随机数，接口如下：uint32\_t rand(void)。每调用一次返回一次真随机的 32bit 的数。协议栈在初始化时会获取真随机的种子数值，将该数值做为 rand 函数的种子，决定 rand 函数返回的是真随机数。

## 4.4. Fr801xH 芯片如何进行多连接的操作？

A: Fr801xH 芯片最多支持 20 个链接，可以在链接的同时进行广播和扫描的操作。同时支持最多 2 个广播。用户在进行多链接的工程时，可以参考文档《Fr801xH 如何构建多从机工程.pdf》。文档在 SDK\docs\Application Notes 获取。

另可以参考多链接的示例工程：SDK\examples\none\_evm\ble\_multi\_role。该工程同时使能广播和扫描，如果

扫描到要连设备名字，进行主动连接。如果被主机设备连接上后，能再次发送广播，再次被连接上。

也可参考 AT 的示例工程，该工程稍微复杂。

#### 4.5. 是否有 hid 键盘和鼠标的示例工程代码?

A: Fr801xH SDK 内部有 hid 键盘和鼠标的示例工程代码。工程目录为:

键盘工程, SDK\examples\developing\ble\_hid\_kbd\_mice。

鼠标工程, SDK\examples\developing\ble\_hid\_mouse\_mice。

键盘和鼠标的工程, 均能在主流的 IOS, 安卓, 和 win7/10 平台对接运行。

#### 4.6. Fr801xH 系列芯片支持 5.0 的各种广播类型吗?

A: Fr801xH 系列芯片支持 5.0 协议规定的各种扩展广播和周期性广播, 同时支持广播的白名单过滤。可以参考示例工程, SDK\examples\none\_evm\ble\_aux\_adv\_demo 内定义的各种广播类型示例代码。

一共有 14 种广播类型的示例代码

```
#define TEST_ADV_MODE_UNDIRECT (0) //undirect connectable & scannable adv
#define TEST_ADV_MODE_DIRECT (0) //Low duty direct connectable & non-scannable adv
#define TEST_ADV_MODE_HDC_DIRECT (0) //High duty direct connectable & non-scannable adv. 1.5ms interval

#define TEST_ADV_MODE_EXTEND_CONN_UNDIRECT (0) //extended undirect connectable & non-scannable adv
#define TEST_ADV_MODE_EXTEND_CONN_DIRECT (0) //extended direct connectable & non-scannable adv
#define TEST_ADV_MODE_EXTEND_NON_CONN_SCAN (0) //extended non-connectable & scannable adv
#define TEST_ADV_MODE_EXTEND_CONN_UNDIRECT_LONGRANGE (0) //extended longrange undirect connectable & non-scannable adv
#define TEST_ADV_MODE_EXTEND_CONN_DIRECT_LONGRANGE (0) //extended longrange direct connectable & non-scannable adv
#define TEST_ADV_MODE_EXTEND_NON_CONN_SCAN_LONGRANGE (0) //extended longrange non-connectable & scannable adv
#define TEST_ADV_MODE_PER_ADV_UNDIRECT (0) //periodic undirect non-connectable & scannable adv
#define TEST_ADV_MODE_PER_ADV_DIRECT (0) //periodic direct non-connectable & scannable adv

#define TEST_ADV_MODE_UNDIRECT_WHITE_LIST (0) //undirect connectable & scannable adv with withlist
#define TEST_ADV_MODE_UNDIRECT_DOUBLE_ADV (1)

#define TEST_BOARD_ADV (1)
#define TEST_BOARD_SCAN_OR_CONN (0)
```

使用时, 打开其中一个广播的宏, 然后最下面的广播板宏打开和扫描\_连接板的宏打开后, 分别编译链接。将生成的广播板的 bin 文件烧录到第一块开发板, 将扫描\_连接板的 bin 文件烧录到第 2 块开发板, 然后对两块板上电开始运行。

#### 4.7. 有没有禁止和使能 latency 的接口函数

A: 可以调用接口禁止和使能链接的 latency, 接口如下: void patch\_set\_latency(uint16\_t latency)。在链接参数启用 latency 的情况下, 可以通过调用 patch\_set\_latency(0);和 patch\_set\_latency(org\_latency)来禁止和使能 latency。禁止 latency 的功能可以在大吞吐量传输的情况下使用, 例如, 在传输语音之前禁止 latency, 语音传输结束后, 在使能 latency。

#### 4.8. 如何使用 ANCS 的 profile

A: Fr801xH 芯片的 SDK 内部提供苹果 IOS 平台 ANCS 服务的 profile 代码。代码文件在 SDK\components\ble\profiles\ble\_ANCS 获取。使用时引用改 profile 文件到工程中, 然后参考该文件夹下的《说明.txt》文件进行调用。



## 4.9. Fr801xH 是否支持扫描以及使用方法

A: Fr801xH 的 BLE 芯片支持扫描功能, 扫描的示例代码参考示例工程, SDK\examples\none\_evm\ble\_aux\_adv\_demo 内的扫描代码。

启动扫描前, 如果要获取正确的 rssi 值, 需要在启动扫描前调用 `gap_set_link_rssi_report(true)` 开启 rssi 换算功能。在扫描结束后调用 `gap_set_link_rssi_report(false)` 以提高射频性能。

扫描的搜到的广播包通过协议栈事件回调函数分支事件: `GAP_EVT_ADV_REPORT` 进行获取。

扫描结束后会触发协议栈事件回调函数分支事件: `GAP_EVT_SCAN_END`。

## 5. 电路以及电气参数

### 5.1. Fr801xH 系列芯片的参考外围设计电路在哪里获取?

A: Fr801xH 芯片的参考电路设计文档在 SDK\docs\Hardware Reference Design 内部, 一共有 Fr8012H, Fr8016H 和 Fr8018H 三种型号的参考外围电路。

### 5.2. Fr801xH 芯片如何进行频偏和天线匹配校准?

A: 通常情况下用户在画完 PCB 板后, 需要对射频性能进行一次性的优化。优化分为频偏校准和天线匹配校准。具体请联系我们的代理商或 AE 工程师。

### 5.3. Fr801xH 芯片调试功耗时如何测试底电流?

A: 在工程内调用 `system_sleep_enable()` 后, 芯片运行时就会使能保持链接的休眠。用电流表串接到电源线, 然后给芯片上电, 运行时将电流表打到毫安档, 然后突然打到微安档后, 看到电流跳变的最小值就是芯片休眠时的底电流。长时间运行在微安档, 可能导致芯片重启。

另外推荐使用 EFM32 Kit(Leopard GECKO) 开发板进行电流测试。

### 5.4. Fr801xH 芯片的运行供电电压范围是多少?

A: Fr801xH 芯片的运行电压范围是 1.8V~4.2V。不在此范围的电压供电是, 芯片不能工作。

### 5.5. Fr801xH 芯片的 IO 高电平是的电压范围是多少?

A: Fr801xH 芯片的 IO 口做为高电平输出时的电压与 ALDO 管脚的电压一致, 可以在工程内调用函数 `void pmu_set_aldo_voltage(enum pmu_aldo_work_mode_t mode, enum pmu_aldo_voltage_t value)` 进行设置, 能够

设置的范围是：2.1V~3.5V。

注意：该值设置越大，芯片运行时的功耗越大。如果设置的电压值比 VBAT 的电压高时，IO 口的实际高电平电压值就是 VBAT 的值。比如：VBAT 当前值是 3V，调用上面函数设置 IO 口高电平电压值为 3.5V，实际的 IO 口高电平电压值为 3V。

## 5.6. 该如何选择合适的发射功率

A: 用户可以在系统初始化时的入口函数内调用 `void system_set_tx_power(enum rf_tx_power_t tx_power);` 调节射频的发射功率，范围从 -16db~+10db。默认的发射功率是 1db。要提高远距离的射频收发性能时，需要提高发射功率。

## 5.7. 如何切换 CPU 的工作频率

A: 用户可以在 `void user_custom_parameters(void)` 的入口函数内决定系统默认的 CPU 运行频率。示例代码如下： `__jump_table.system_clk = SYSTEM_SYS_CLK_48M;` 表示 CPU 运行频率是 48MHz。

另外在程序运行过程中要动态的切换 CPU 的运行频率，可以定义一个如下的函数进行。

```
void user_set_cpu_clk(uint8_t clk)
{
    if(clk == 12)
        __jump_table.system_clk = SYSTEM_SYS_CLK_12M;
    else if (clk == 24)
        __jump_table.system_clk = SYSTEM_SYS_CLK_24M;
    else if (clk == 48)
        __jump_table.system_clk = SYSTEM_SYS_CLK_48M;
    system_set_pclk(__jump_table.system_clk);
}
```

## 5.8. 如何在项目中配置和使用充电功能

A: 用户可以在入口函数 `void user_entry_before_ble_init(void)` 内部调用如下示例代码，开启充电功能和中断。

```
void user_entry_before_ble_init(void)
{
    /* set system power supply in BUCK mode */
    pmu_set_sys_power_mode(PMU_SYS_POW_BUCK);
    //enable charge isr
    pmu_enable_irq(PMU_ISR_BIT_ACOK
                  | PMU_ISR_BIT_ACOFF
                  | PMU_ISR_BIT_BAT);
    NVIC_EnableIRQ(PMU_IRQn);
    //enable charge and set current and terminal voltage
    pmu_enable_charge(CHG_CUR_113MA, CHG_VOL_4_20V);

    system_set_port_pull(GPIO_PC4, true);
    system_set_port_mux(GPIO_PORT_C, GPIO_BIT_4, PORTC4_FUNC_UART1_RXD);
    system_set_port_mux(GPIO_PORT_C, GPIO_BIT_5, PORTC5_FUNC_UART1_TXD);
    uart_init(UART1, BAUD_RATE_921600);

    ool_write(PMU_REG_ADKEY_ALDO_CTRL, ool_read(PMU_REG_ADKEY_ALDO_CTRL) & (~(1<<3)));
}
```

同时重定义充电中断函数如下来获取充电事件：

```
__attribute__((section("ram_code"))) void charge_isr_ram(uint8_t type)
{
    if(type == 2)    //charge full isr
    {
        co_printf("charge full\r\n");
        pmu_disable_irq(PMU_ISR_BIT_BAT);
        pmu_enable_irq(PMU_ISR_BIT_ACOFF);
    }
    else if(type == 1) //charge plug out
    {
        pmu_disable_irq(PMU_ISR_BIT_BAT);
        pmu_enable_irq(PMU_ISR_BIT_ACOK);
        co_printf("charge out\r\n");
    }
    else if(type == 0) //charge plug in
    {
        pmu_disable_irq(PMU_ISR_BIT_ACOK);
        pmu_enable_irq(PMU_ISR_BIT_ACOFF|PMU_ISR_BIT_BAT);
        co_printf("charge in\r\n");
    }
}
```

## 5.9. 做 IEC 标准的 ESD 测试时，产品设计有哪些注意事项

A: 用户使用 Fr801xH 系列芯片，如果需要通过类似 IEC 标准的 ESD 测试，做产品设计时需要注意如下事项。

1. 蓝牙芯片的 pin16 VBAT 脚放置双向 3.3V TVS 管，参考型号：PESDNC2FD3V3B
2. 蓝牙芯片的 pin5 RF 脚放置双向 3.3V 低 Cj 值的 TVS 管，参考型号：PESDUC2FD3V3B
3. 蓝牙芯片的 pin14 reset 脚按照参考电路放置 1K 下拉电阻。
4. 模组需加上屏蔽罩。
5. 模组底部不要放置测试点，通过边上的邮票孔焊盘进行烧录测试。
6. 模组在主板上的安装位置应注意避空，应尽量避开产品外壳的金属。
7. 模组与主板 MCU 的通信之间串联 100R 电阻。
8. 主板上也需要做 ESD 防护。
9. 软件打开 watchdog。

## 6. 版本历史

<b>Vers ion</b>	<b>Date</b>	<b>Author</b>	<b>Description</b>
0.1	2020-08-05	Dong Youcai	Draft