RDA5876 编程指南

REV	DATE	AUTHER	CHANGE DESCRIPTION
V1.0	2012-4-13	马千里	Initial version.
V1.3	2012-10-26	马千里	Change for bb test
V1.4	2012-11-20	马千里	Add rx mode
V1.5			

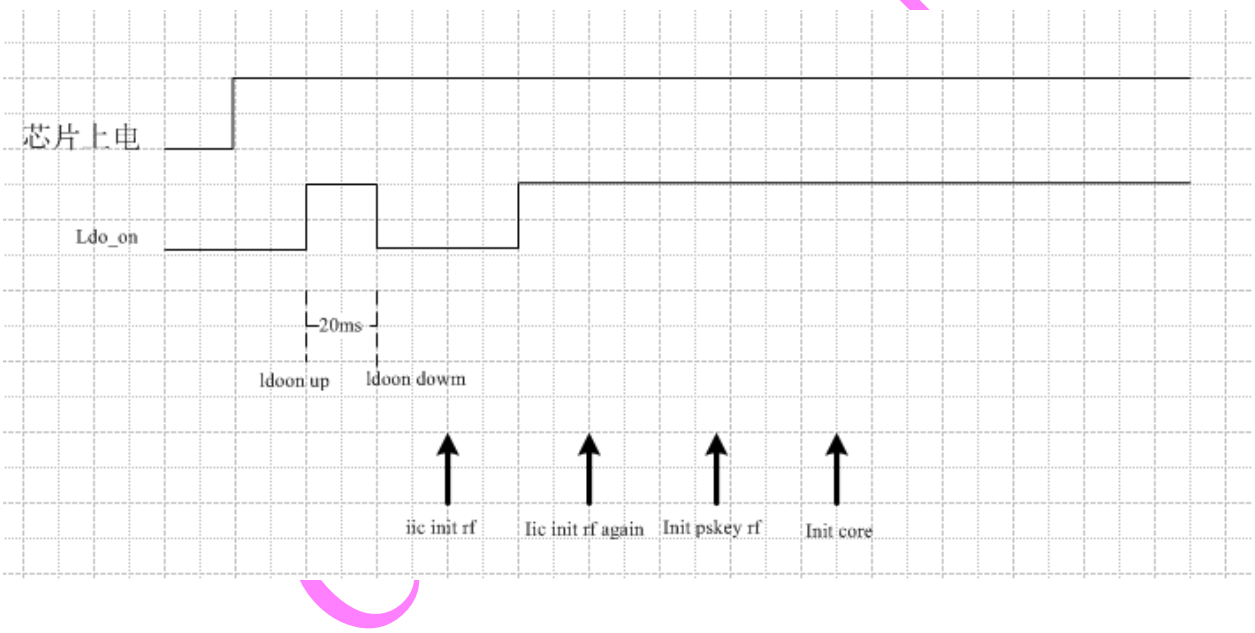
上电配置顺序：

RDA5875 的配置包括 RF 部分和 core 部分的配置，分为 2 种配置方式，

一. 使用 iic 来初始化 rf 部分，配置顺序是：

1. 芯片上电，拉高 LDOON。
2. 等待 20ms。
3. 拉低 LDOON
4. 通过 iic 配置 5875 RF 部分。
5. 拉高 LDOON，等待 20ms
6. 通过 iic 再次配置 5875 RF 部分。
7. 通过 uart 配置 5875 pskey rf 部分
8. 通过 uart 配置 5875 的 core 部分。

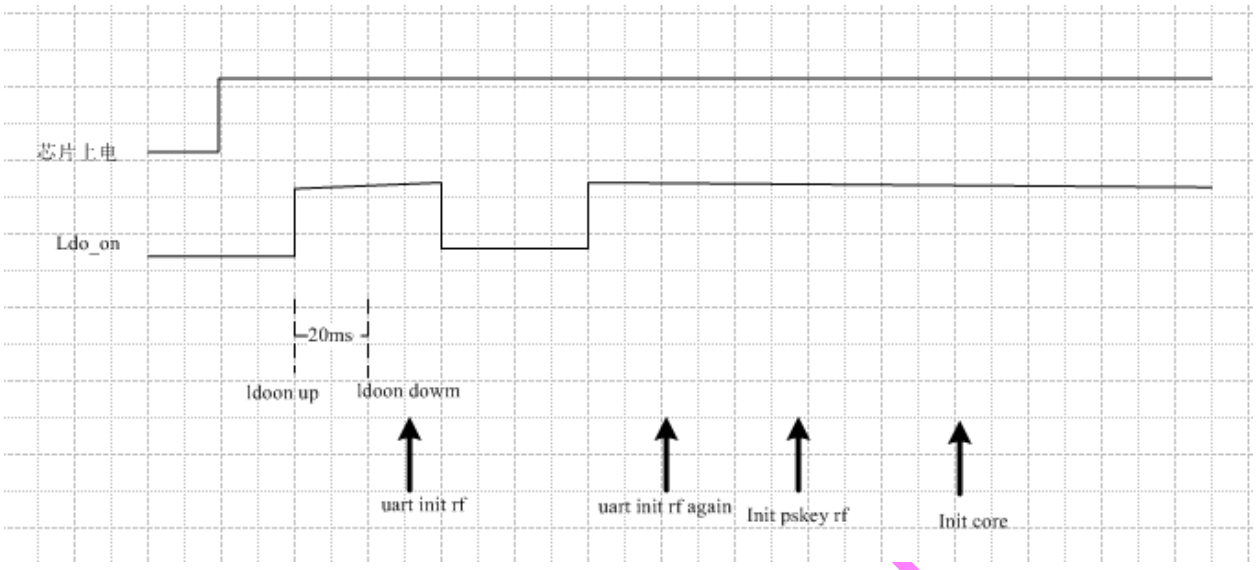
直接接电池时上电时序图如下：



二. 使用 uart 来初始化 rf 部分，配置顺序是：

1.芯片上电，拉高 LDOON。
2.等待 20ms。
3.通过 uart 配置 5875 RF 部分。
4.拉低 LDOON，core 部分随之复位，复位 core 部分。
5.拉高 LDOON，等待 20ms
6.通过 uart 配置 5875 RF 部分
7.通过 uart 配置 5875 pskey rf 部分
8.通过 UART 配置 5875 的 core 部分。

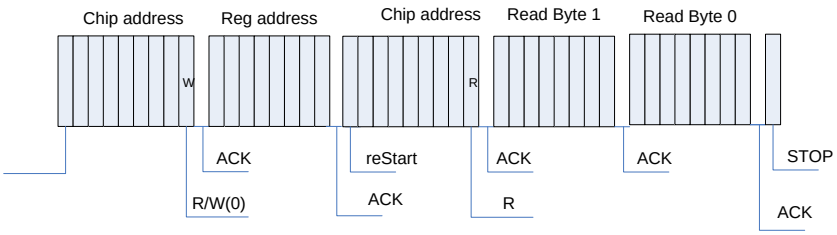
直接接电池时上电时序图如下：



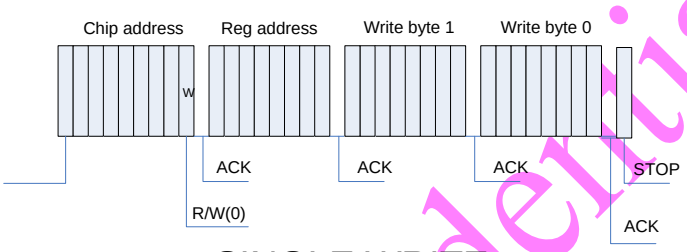
RDA5875 的电源输入 VBAT 可以直接接电池或者接在 LDO 后，推荐电压范围为 3.3v-4v

5875 RF 部分 I2C 协议：

5875 RF 部分的 I2C slave 设备地址是 0010110。寄存器地址是 8 位，数据宽度是 16 位，高 byte 在前，低 byte 在后。读写寄存器时序如下：



SINGLE READ

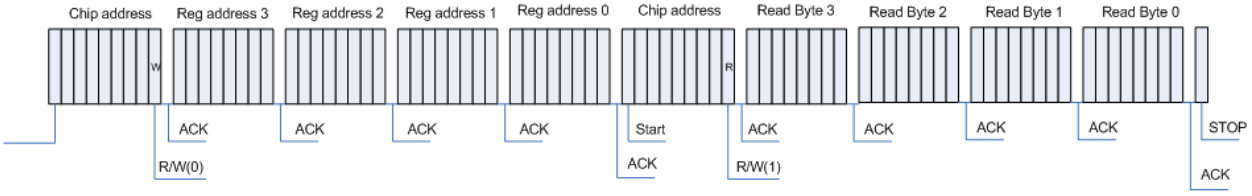


SINGLE WRITE

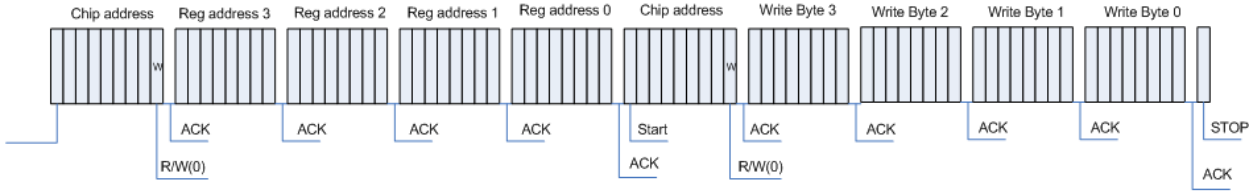
注意，read 有 restart，write 没有 restart。

5875 Core 部分 I2C 协议：

5875 core 部分的 I2C slave 设备地址是 0010101。寄存器地址和数据宽度都是 32 位，
高 byte 在前，低 byte 在后。读写寄存器时序如下：



SINGLE READ



SINGLE WRITE

标准的 I2C 时序，大致上是一个 command byte，接下来写入 32 位寄存器地址，然后 restart，再一个 command byte，最后是读的或者写的数据。

请注意 core 部分时序和 RF 部分时序读时序相似，写时序有区别，RF 部分没有中间的 restart。

I2C 地址配置：

5876 中有 RF 部分和 core 部分两个 I2C slave device，这两个 device 的 I2C 地址高两位可以通过 TM 配置为 2’b00 和 2’b11。具体 TM 和 MA[6:5]的映射关系请参考后面的”TM 配置”一节.
RF 部分地址： {MA[6:5],5’b10110}
Core 部分地址： {MA[6:5],5’b10101}
当 MA[6:5]＝2’b00 时，RF 地址和 Core 地址分别是 7’b0010110,7’b0010101。
当 MA[6:5]＝2’b11 时，RF 地址和 Core 地址分别是 7’b1110110,7’b1110101。

UART 波特率配置:

5875 缺省是 115200 波特率，最高支持到 3.2M。

通过 pskey 配置如下：

{0x80000060,0x000e1000}

{0x80000040,0x00000100}

其中，0x80000060 寄存器值为波特率的绝对值，如 0x000e1000 对应的 10 进制就是 921600

0x80000040 为 pskey 使能位，value 的每一个 bit 对应一个 pskey。具体的 pskey 说明请看对应的 pskey 配置

- **UART 格式：**

5875 默认的 UART 格式为 8 位有效数据，一位起始位、一位停止位，没有奇偶校验位。有效数据从最低位开始传输。

Uart 的格式可以进行修改，修改方式如下：

{0x8000005c, 0x03300000}

{0x80000040,0x00000080}

0x8000005c 寄存器的定义如下：

Bit0-7：hcitype 0x00 uart

0x01 bcsp

Bit8-15 uart flag 默认为 0，不修改

Bit16-23 convert_time 默认为 0x30，不修改

Bit24-31 uart setting 默认为 0x03 相应的定义如下

```
#define XR7_LCR_PEN          0x08
#define XR7_LCR_5BITS        0x0
#define XR7_LCR_6BITS        0x2
#define XR7_LCR_7BITS        0x1
#define XR7_LCR_8BITS        0x3
#define XR7_LCR_1STOP        0x0
#define XR7_LCR_2STOP        0x4
#define XR7_LCR_PODD         0x0
#define XR7_LCR_PEVEN        0x10
```

4	EPS	R/W	偶校验选择位 **Reset Value:** 0x0
3	PEN	R/W	奇偶校验使能位 **Reset Value:** 0x0
2	STOP	R/W	停止位个数 0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit **Reset Value:** 0x0
1:0	DLS	R/W	数据长度选择 00 = 5 bits

			01 = 6 bits 10 = 7 bits 11 = 8 bits **Reset Value:** 0x0

UART 流控配置：

UART 缺省关闭流控，可以通过配置 UART 寄存器开启流控功能。有的平台有用。
就目前看到的来讲，只要是平台原来有开流控的，我们也必须开。因为流控在进入睡眠时会无效，唤醒后时钟恢复好后才会重新有效。流控信号同时也有代表时钟是否恢复好的物理意义，参与睡眠握手机制，所以固定接为有效是不行的。这是合理的，睡眠时当然流控应该无效。5875 也支持睡眠时自动拉高 RTS，时钟恢复好后才拉低。

5875 有对应的流控脚 rts 和 cts
但是 5876 由于 pin 脚数量限制，没有专门的流控脚，那么如果需要开启流控配置，就需要把 gpio2 作为流控脚 cts，把 mcu 的 rts 接地，因为蓝牙的 uart 硬件 buff 比较大，所以可以默认以为 bb 任何时候都可以向蓝牙发数。

如果采用 gpio2 为流控脚的话，需要进行如下配置配置
{0x40200044,0x0000003c};

开启流控的配置是：
{0x50000010,0x00000122};

PCM 接口配置：

PCM 接口简单说明

PCM 接口用于 RDA5875 与 Host 系统之间传输蓝牙语音数据，通过 PCM 接口，在建立 SCO 连接后可以发送和接收 PCM 数据。PCM 接口共有 4 根信号，PCM_CLK, PCM_SYNC, PCM_IN 和 PCM_OUT. RDA5875 的 PCM 接口通过用户配置可支持多种格式，支持的具体格式如下表所示：

接口配置参数	支持的数据格式
接口数据类型	Linear /A law /U law
数据长度（PCM_IN/ PCM_OUT 的比特数）	Linear: 16 bits A law/U law: 8 bits
语音采样速率	8Khz samples
PCM Clock/Sync 来源	Master Mode: RDA5875 产生 Slave Mode: 来自外部
PCM Clock 时钟频率	MasterMode: 64Khz/128Khz/256Khz/512Khz/1Mhz Slave Mode: 64Khz/128Khz/256Khz/512Khz/1Mhz/2Mhz (注：Slave Mode 下的 2Mhz 时钟在 RDA5875 中支持)
PCM Sync 速率	8Khz
PCM Sync 类型	Long sync / Short Sync
Data Ordering	MSB / LSB
数据格式	2’s /1’s /sign
Sign extension	支持
Zero padding	支持
多 Slot 模式	支持

相关寄存器

PCM Configure

44	SCO_CFG1[3:0]				SCO_CFG0[3:0]			
45	VCI_CLK_SEL[1:0]		SCO_FIFO[1:0]		SCO_CFG2[3:0]			
46	2S_INPUT	1S_INPUT	SIGN_INPUT	MSB_INPUT	SYNC_LS	VCI_SYNC_DIR	PCM_CLK_SLOT	VCI_CLK_SEL[2]
47	2S_OUPUT	1S_OUPUT	SIGN_OUTPUT	MSB_OUTPUT	ADPCM_EN	S_1024_DELAY_ECO	SIGN_EXTENSION _OUTPUT	SIGN_EXTENSION _INPUT

- **Name:** PCM Configure
- **Size:** 32 bits
- **Address Offset:** 0x44
- **Read/write access:** write

Bit	Name	Function	Default & Note
31	2S_OUPUT	Data output format is 2’s set to 1, else 0	1’b1

30	1S_OUPUT	Data output format is 1's set to 1, else 0	1'b0
29	SIGN_OUTPUT	Data output format is sign set to 1, else 0	1'b0
28	MSB_OUTPUT	Data output ordering is MSB set to 1, else 0	1'b1
27	ADPCM_EN	Adpcm function enable set to 1, else 0	1'b0
26	S_1024_DELAY_ECO	PCM clock is 1Mhz or 2Mhz set to 1, else 0	1'b0
25	SIGN_EXTENSION_OUTPUT	Data output is sign extension set to 1, else 0	1'b0
24	SIGN_EXTENSION_INPUT	Data input is sign extension set to 1, else 0	1'b0
23	2S_INPUT	Data input format is 2's set to 1, else 0	1'b1
22	1S_INPUT	Data input format is 1's set to 1, else 0	1'b0
21	SIGN_INPUT	Data input format is sign set to 1, else 0	1'b0
20	MSB_INPUT	Data input ordering is MSB set to 1, else 0	1'b1
19	SYNC_LS	Short pcm_sync set to1, else 0	1'b1
18	VCI_SYNC_DIR	Master mode set to 1,else 0	1'b1
17	PCM_CLK_SLOT	Master mode and last slot is valid set to 1, else 0	1'b0
16	VCI_CLK_SEL[2]	VCI_CLK_SEL[2:0] set to 0~5 to choose VCI clock 0:Master 128Khz 1:Master 1Mhz 2:Master 64Khz	1'b1
15:14	VCI_CLK_SEL[1:0]	4:Master 256Khz 5:Master 512Khz 3:Slave mode, clock from external (64khz~~2Mhz)	2'h0
13:12	SCO_FIFO[1:0]	Choose which fifo to use, 0: fifo0 1:fifo1 2:fifo2	2'h0
11:8	SCO_CFG2[3:0]	The Third SCO connection is valid set to 1~7,13~14, else 0	4'h0
7:4	SCO_CFG1[3:0]	The Second SCO connection is valid set to 1~7,13~14, else 0	4'h0
3:0	SCO_CFG0[3:0]	The First SCO connection is valid 4'h1: A law to U law 4'h2: U law to A law 4'h3: 12bits to CVSD 4'h4: 13bits to CVSD 4'h5: 14bits to CVSD 4'h6: 15bits to CVSD 4'h7: 16bits to CVSD 4'hD: A law to CVSD 4'hE: U law to CVSD	4'h7

- **Name:** ESCO Ctrl
- **Size:** 32 bits
- **Address Offset:** 0x48
- **Read/write access:** write

Bit	Name	Function	Default & Note
25:24	PCM_SLOT_NUM[1:0]	The first slot is valid set to 0, else 1	2'h0

- **Name:** PCM_SLOT_SEL
- **Size:** 16 bits
- **Address Offset:** 0x4c
- **Read/write access:** write

Bit	Name	Function	Default & Note
15:0	PCM_SLOT_ SEL [15:0]	Choose which slot is valid	16'h1

常见的 PCM 接口配置和配置举例

Master mode +16bits Linear to CVSD + 2's + LSB + short sync + 256Khz （6600R）
配置值：101d0007

Master mode + 16bits Linear to CVSD + 2's + LSB + short sync +128Khz (6600L)
配置值：101e0007

Slave mode + 16bits Linear to CVSD +2's +LSB+ short sync + 2Mhz (VIA 2M 平台)
配置值：1418c007

Slave mode +16bits Linear to CVSD + 2's + MSB + long sync + 128Khz(64Khz~2Mhz)
配置值：9090c007

需要注意的问题

关于 SLOT 的说明，8K 的 PCM Sync，根据 PCM Clock 和 PCM IN /PCM OUT 的比特数来划分 SLOT 的个数。举例，PCM Clock 是 128Khz, PCM IN 是 16 比特，则有 1 个 SLOT; PCM Clock 是 256Khz, PCM IN 是 8 比特，则有 4 个 SLOT; 以此来推断 SLOT 的个数，哪个 SLOT 有效，则把 PCM_SLOT_SEL[15:0]寄存器的哪个比特位置 1，其余位则置 0.

在展讯6600I平台上碰到过最后一个SLOT有效的情况，需要配置ESCO Ctrl寄存器和PCM_SLOT_SEL寄存器，把 ESCO Ctrl 寄存器的 第 25：24 比特 PCM_SLOT_NUM[1:0]置 2'h1（默认是 2'h0），PCM_SLOT_SEL[15:0]置 16'h2（默认是 16'h1）。

如果是 Master mode 而且还是最后一个 SLOT 有效,还需把 PCM Configure 寄存器的第 17bit（第 0bit 开始）PCM_CLK_SLOT 置 1。

TM 配置：

射频寄存器中 0xb2 寄存器的 bit2-0 对应于 TM[2:0]，默认值为 001。
TM[2:0] 用于设置 5875 的工作模式，主要是区分不同 HCI 接口或者不同兼容平台和量产测试模式,5875 中也兼用作 RF 和 core I2C 设备地址高两位的选择。

有 UART-M,UART-C 和 Normal UART 三种 HCI 接口设置,每种 HCI 接口对应两个 TM[2:0]编码,这两个不同 TM[2:0]编码仅仅用来选择 I2C 设备地址的高两位(一种对应 2'b00,一种对应 2'b11),其它完全相同。

TM[2:0]	CHIP MODE	I2C ADDR MA[6:5]
000	UART-M	2'b11
001	UART-M (MTK platform compatible)	2'b00 (normal used)
010	UART-C (CSR platform compatible)	2'b00 (normal used)
011	Normal-UART	2'b00 (normal used)
100	UART-C	2'b11
101	Normal-UART	2'b11
110	BIST	2'b00
111	SCAN	2'b00

其它平台 porting 时建议使用 TM＝011，normal uart 模式。此时需要在初始化配置中针对不同平台加上 sleep，hostwake 这两条 pskey 配置，和 uart 自动波特率切换使能配置。TM mode 必须 ldo 拉低后，再拉高来使能，也就是说，如果通过 iic 在 ldoon 为低的时候配置 tmmode，然后拉高 ldoon，芯片就会按照新的模式工作，如果用 uart 在 ldoon 为高的时候配置 tm mode，那么需要拉低 ldoon 然后拉高 ldoon，芯片按照新的模式进行工作

共用晶体配置：

5875 支持共用晶体功能，可以和基带共用系统时钟，一般是从 GSM transciver 的 XOUT 脚分出一路，接到 5875 的 XIN 脚，这样蓝牙不需要单独系统时钟晶体，节省成本。

相关的有 XENIN 和 XENOUT pin 脚。一般应用下,XENIN 接地，XENOUT 输出 5875 自己的系统时钟请求。XENOUT 接在基带芯片的外部时钟请求输入脚或者中断管脚上。

此功能默认开启

睡眠握手机制配置：

当 5875 应用于手机平台上时，5875 和基带芯片都会在允许的情况下进入睡眠状态，以节省功耗。5875 和 host 之间有握手机制来保证只有允许时才进入睡眠和需要唤醒时能被唤醒。不同平台睡眠机制不同。这里指的睡眠状态对应于关掉系统晶体 XTAL，系统功耗处于最低的模式，5875 在这种状态下（我们习惯于叫这种状态为深睡眠）耗电约为 400uA，整个手机平台耗电约为 2mA（普通 DH 水平）。睡眠状态的进入和退出时间都比较长，尤其是退出时，需要系统时钟晶体重新起振和稳定，应该是 6ms～15ms 的时间后系统才能恢复到正常工作状态。

5875 和基带芯片工作于睡眠状态是独立无关的事情。所以睡眠握手机制有两个独立的方向，一个方向是基带芯片控制 5875 允许进入睡眠和唤醒退出睡眠，另一个方向是 5875 唤醒基带。这两个方向在控制机制上是基本对称的。

5875 睡眠和唤醒方向：

5875 在软硬件允许进入睡眠时，软件会写硬件 SCU 中的 deepsleep_mode 寄存器位，告诉硬件应该进入睡眠模式，硬件会依序关断数字时钟，停止系统晶体起振和调整。5875 从此进入 200uA 耗电的睡眠模式，此时数字部分依然有电，但只有 32K 时钟电路工作，耗电很低。数字电源也只能负载 500uA 以下电流，所以此时就算有系统时钟，相应数字电路也是不能工作的。

软件允许进入睡眠的条件是没有 pending 的任务，而且蓝牙基带 TAB 已经置于 sleep 模式。硬件允许进入睡眠的条件是没有有效的中断，而且 HCI bus 也已经 idle。判断 HCI bus 已经 idle 支持如下几种方法：根据 uart rx 为高超过一定时间；根据 uart rx 上的 break 命令。可以分别用寄存器使能。

当基带芯片需要和 5875 通信时，会估计蓝牙是否已经睡眠，如果基带芯片觉得蓝牙有可能已经进入睡眠状态，就会先唤醒 5875，等待握手信号后再进行通信。5875 检测到唤醒动作，会依序恢复数字供电电源，重新起振系统晶体和等待稳定，开启数字电路时钟。5875 从此进入正常工作模式，首先会给出和基带的握手信号，通知基带 5875 已经进入正常工作模式。

5875 支持的唤醒方法有如下几种：来自蓝牙基带 32K 计时 auxtimer 的中断；uart rx 上有低电平或者翻转；uart 上 break 撤销；uart cts 拉低；外部中断。可以分别用寄存器使能。

5875 还可以在睡眠时自动拉 uart rts 为高，主要用途是当平台上如果有硬件流控，基带唤醒蓝牙后蓝牙恢复工作前流控一直无效，以免基带发 HCI 命令下来 5875 收不到。

总的来说，5875 的睡眠是软件决定和指导进入，硬件唤醒退出。

相关寄存器说明如下：

PSKey:

SLEEP SETTING：
用低 8 位，代表最短睡眠时间，默认值为 0x40；
15~8 位表示在 standby mode 下醒的时间，默认值为 50ms；
19~16 代表在没有任务的情况下，空循环走的遍数，默认值为 0，
31 位为 LF_OSCPRESENT 标志位，30 位为 SLEEPENABLE 标志位，29 位为 DEEPSLEEP 标志位，三个标志位默认值都为 0。

HOST_WAKE_CMD1：
HOST 唤醒 Core 的 command（实际对应为 UART RX 上 1us 低脉冲），默认值为 0xfcc0。

Core 部分 SCU 寄存器：

Register 14h. DEEPSLEEP MODE REGISTER
32'b0010_0101_1000_0000_0001_1011_0001_0000

Bit	Name	Function	Default & Note
6	UART_BREAK_VALID	If 1, host side use uart_rxd to inform if 5875 could enter into deepsleep mode. If 0, not use uart_rxd in deepsleep negotiation.	1’b0
5	UART_CTS_VALID	If 1, host side use uart_cts to inform if 5875 should exit deepsleep mode. If 0, not use uart_cts in deepsleep negotiation.	1’b0
4	HOSTBUS_ACTIVITY_VALID	If 1, detect hostbus activity to know if hostbus allow enter into deepsleep. If 0, not use hostbus activity in deepsleep negotiation.	1’b1
1	DEEPSLEEP_EN	Deepsleep mode enable.	1’b0
0	DEEPSLEEP_ALLOWED	Hostbus is idle now, means host allows 5875 enter into deepsleep mode now.	Read only

Register 18h. DEEPSLEEP MODE REGISTER (HOSTBUS TIMER FOR DEEPSLEEP REGISTER)
32'b0010_0000_0001_1111_0010_0000_0000_0000

Bit	Name	Function	Default & Note

31	HOSTBUS_IS_WAKING_CLR	Clear hostbus_is_waking status. Write 1 to clear.	1'b0
29	HOSTBUS_IS_WAKING_RESP_EN_FORSW (HOSTBUS_IS_WAKING_EN)	If 1, SW will autogen response to cpu when hostbus_is_waking. If 0, SW will not autogen response to cpu when hostbus_is_waking.	1'b1
28	HOSTBUS_IS_WAKING	1: hostbus activity causes waking up. 0: TAB auxtimr causes waking up.	Readonly
27:26	HOSTBUS_STATE<1:0>	00:awake 01:asleep 10:waking 11:reserved	Readonly
24	UART_RTS_BLOCK_VALID	If 1, 5875 block uart_rts when in deepsleep state.	1'b0
15:0	HOSTBUS_ACTIVITY_IDLE_TIMER<15:0>	If hostbus idle longer than this timer, assume hostbus is asleep, and allow BT controller to enter into deepsleep. Unit is ms.	16'h2000 (8s)

Register 2Ch. HOSTBUS TIMER FOR HOSTWAKE REGISTER
32'b0000_0000_0101_1000_0001_0000_0000_0000

Bit	Name	Function	Default & Note
31:23	reserved		0
22	HOSTBUS_WAKINGSTATE_ACTEXIT_EN	If 1, hostbus activity could make hostbus state exit from waking state. If 0, could not.	1'b1
21	HOSTBUS_WAKINGSTATE_TIMEREXIT_EN	If 1, timer could make hostbus state exit from waking state. If 0, could not.	1'b0

列一下见过的这个方向上的握手机制，这些都测试过：

- MTK 平台：利用 uart rx 为高的时间判断 HCI 总线是否 idle；基带利用 uart rx 上的 1us 低电平脉冲来唤醒 5875，5875 醒来后回 fcc0 的 event 给基带，以通知基带 5875 已经可以正常工作。
- 使用 CSR 的平台：CSR 比较特殊，由于 BCSP 可以重传的缘故，基带可以一直发 BCSP 命令，不用考虑 5875 是否已经进入睡眠，如果 5875 已经睡眠，看见 uart 上的动作会醒来，醒来后就会正确处理以后收到的 HCI 命令，HCI 通信会继续。5875 也是利用 uart rx 为高的时间来判断 HCI 总线是否 idle。

唤醒 Host 方向：

基带芯片也可能进入睡眠状态，每隔 1～2s 和基站同步一次。当 5875 有主动和基带芯片通信需求时，会估计基带是否处于睡眠状态，如果认为基带芯片有可能处于睡眠状态，会首先唤醒基带芯片，等到握手信号后再和基带芯片通信。

比如 5875 被搜索和配对，5875 就需要给基带发 event 以获取名字或者做认证配对，这是一个典型的需要唤醒 host 的 case。

判断基带是否处于睡眠支持如下几种方法：根据 uart rx 为高的时间判断基带是否处于睡眠，这儿的 timer 是专门用于这个方向的，和 5875 睡眠时判断 HCI bus 的 timer 独立。或者根据基带控制 5875 允许睡眠的控制信号来判断，只要是基带允许蓝牙睡眠，就认为基带也有可能睡眠。

唤醒 host 的方法有如下几种：uart tx 上传数据或者是发 break 唤醒，这个需要基带芯片支持 uart 唤醒；单独的 hostwake 信号唤醒，一般是以中断信号进入基带。hostwake 主要有两种时序，一种是仅在认为基带睡眠时在发数据前拉一个脉冲唤醒基带，物理意义是一个提醒；另一种是 HCI 上有去基带的数据就拉起，直到 5875 进入睡眠，物理意义是这次工作状态中有 HCI 会话，需要基带处于工作状态。

总的来说，是否需要唤醒 host 是硬件判断，软件读取硬件的判断结果后来指导发出唤醒信号。

相关寄存器说明如下：

PSKey:

HOST_WAKE_SETTING：

31 24	23	22	21	20	19 10	9 0
Reserved	method	bothedge	polarity	clrmode	length	pauselength

Method: 1，软件编程 GPIO7 作为 hostwake 信号，以中断方式唤醒 host；0，软件编程硬件，在 uart tx 上产生 break 信号，用 uart 方式唤醒 host。缺省 1。

Bothedge: host cpu 中断是 bothedge 方式，所以每次 hostwake 信号只需要 toggle 一下，不用做一个脉冲。缺省 0。

Polarity: host cpu 中断的极性，1，高有效；0，低有效。BCSP 缺省 0，其它缺省 1。

Clrmode: 1,利用 host 发下的中断响应 cmd 清 hostwake 信号；0，hostwake 自动清，脉宽由 hostwake_length 决定。BCSP 缺省 0，其它缺省 1。

Length: 单位是 ms，hostwake 自动清模式下 hostwake 的长度。缺省 150。

Pauselength: 单位是 ms，hostwake 自动清模式下 cpu waking 的长度，计时从 Hostwake_length 结束开始，计时结束后 5875 才能向 host 发包。缺省 0。

HOST_WAKE_CMD2：

Core 用中断唤醒 HOST 时，HOST 响应中断的 Command，默认值为 0xfcc1。

Core 部分 SCU 寄存器：

Register 14h. DEEPSLEEP MODE REGISTER

32'b0010_0101_1000_0000_0001_1011_0001_0000

Bit	Name	Function	Default & Note

31:22	HOST_WAKE_PULSE_TIMER[9:0]	Length of pulse used to wake up host. Unit is ms.	10'h096
21:12	HOST_WAKE_PAUSE_TIMER[9:0]	Length of time for host side waking. Unit is ms.	10'h001
8	HOST_WAKE_POLARITY	If 1, high asserted; If 0, low asserted.	1'b1

这个极性设置是用于硬件自动产生的 hostwake 脉冲的，和用 GPIO7 产生的 hostwake 脉冲无关。

Register 18h. DEEPSLEEP MODE REGISTER (HOSTBUS TIMER FOR DEEPSLEEP REGISTER)
32'b0010_0000_0001_1111_0010_0000_0000_0000

Bit	Name	Function	Default & Note
30	HOSTWAKE_BOTHEDGE_INT_EN	If 1, host_wake is used as an bothedge interrupt; If 0, host_wake is not used as an bothedge interrupt.	1'b0
25	WAKEUP_TOHOST_REG	Write 1 to trigger a wakeup to host.	1'b0
23	HOST_WAKE_IO_VALID	If 1, 5875 use host_wake asserted by HW timer to wake up host.	1'b0
22	HOST_WAKE_BREAK_VALID	If 1, 5875 use uart break at uart_txd to wake up host.	1'b0

WAKEUP_TOHOST_REG 是硬件自动产生 hostwake 机制的软硬件接口。使用硬件自动拉 hostwake 机制时，设上 HOST_WAKE_BREAK_VALID 可以支持用 uart tx 上 break 信号唤醒 host 功能。

Register 2Ch. HOSTBUS TIMER FOR HOSTWAKE REGISTER
32'b0000_0000_0101_1000_0001_0000_0000_0000

Bit	Name	Function	Default & Note
31:23	reserved		0
22	HOSTBUS_WAKINGSTATE_ACTEXIT_EN	If 1, hostbus activity could make hostbus state exit from waking state. If 0, could not.	1'b1
21	HOSTBUS_WAKINGSTATE_TIMEREXIT_EN	If 1, timer could make hostbus state exit from waking state. If 0, could not.	1'b0
20	HOSTWAKE_GPIO_POLARITY	If 1, high asserted; If 0, low asserted.	1'b1
19	HOSTWAKE_GPIOINT_EN_FORSW	If 1, SW will assert a gpio interrupt to cpu when hostwake_is_need. If 0, SW will not assert a gpio interrupt	1'b1
18	HOSTWAKE_IS_NEED	1: hostbus is idle, host_wake is	Readonly

		needed. 0: hostbus is active, host_wake is not needed.	
17:16	HOSTBUS_STATE_2<1:0>	00:awake 01:asleep 10:waking 11:reserved	Readonly
15:0	HOSTBUS_ACTIVITY_IDLE_TIMER_2<15:0>	If hostbus idle longer than this timer, assume hostbus is asleep, and need assert host_wake firstly when transmit. Unit is ms.	16'h1000 (4s)

这里的 HOSTWAKE_GPIO_POLARITY 极性设置是用于 GPIO7 产生的 hostwake 脉冲的，分 TM，TM＝MTK 时，极性 1 代表高有效；TM＝CSR 时，极性 1 代表低有效。这两个 TM 时 hostwake 极性的不同是软件自动根据 TM 设置好的，所以这里希望缺省时不用配置这个极性寄存器。

HOSTWAKE_GPIOINT_EN_FORSW 寄存器在软件设置 GPIO7 产生 hostwake 脉冲时需要设置为 1，在硬件自动拉 hostwake 脉冲时需要设置为 0（硬件用了，做选择信号，所以必须为 0。但现在软件中用这个寄存器判断是否需要 hostwake 机制，不分软件或者硬件方式，所以软件这儿需要补丁）。

Core 部分 GPIO 寄存器：

Register 04h. GPIO ALTERNATE FUNCTION SELECT REGISTER

32'b0000_0000_0011_1000_0000_0000_1011_1111

Bit	Name	Function	Default & Note
20:0	GPIO_AF_SEL[20:0]	GPIO alternate function select.	

软件拉 GPIO7 产生 hostwake 脉冲时 GPIO7 需要设成 GPIO；硬件拉 hostwake 脉冲时 GPIO7 需要设成 alternate function。

Register 20h. RF5400 CONTROL SOURCE SELECT REGISTER

32'b0000_0000_0000_0000_0000_0000_0000_0000

Bit	Name	Function	Default & Note
4	GPIO7_polarity	GPIO7(hostwake) hardware polarity control.	1'b0

这个寄存器可以翻转 GPIO7 输出 data 的极性，这样在软件不动的情况下可以改变 hostwake 极性配合平台工作，需要和 HOSTWAKE_GPIO_POLARITY 协同设置。

列一下见过的这个方向上的握手机制，这些都测试过：

- MTK 平台：利用 uart rx 为高的时间判断基带是否睡眠；利用软件拉 GPIO7 产生 hostwake 脉冲以中断方式唤醒基带，基带醒来后回 HCI 命令作为中断响应和握手信号，从此可以认为基带恢复正常工作状态。
- 使用 CSR 的平台：CSR 比较特殊，由于 BCSP 可以重传的缘故，5875 可以一直发 BCSP 命令，不用考虑基带是否已经进入睡眠，如果基带已经睡眠，看见 uart 上的动作会醒来，醒来后就会正确处理以后收到的 HCI 命令，HCI 通信会继续。但实际中经常用单独的 hostwake 信号，即使基带支持 uart 唤醒也是这样，可能中断信号唤醒比较简单吧。展讯平台上有 hostwake 唤醒和 uart 上 break 唤醒的 case。VIA 平台上也是 hostwake 唤醒。用 hostwake 信号唤醒时也是用 uart rx 为高的

时间来判断基带是否已经睡眠。

蓝牙地址配置

5875 上电后默认蓝牙地址为：0xae2d22115875,可以通过扩展的 hci 命令来修改
修改命令如下：
Uart send: 01 1a fc 06 + 6byte Bluetooth address
Uart receive： 04 0e 04 01 1a fc 00
这样蓝牙地址就更改为设置的地址了

PSKEY 配置说明

5875 对蓝牙配置通常是通过 pskey 来进行配置的
Pskey 的使能寄存器是 0x80000040,每一个 bit 对应一个配置项
使能 bit 的计算方式如下：
如果配置一个寄存器 0x80000070，那么对应的 bit 为：
（0x80000070-0x80000040）/4=12
那么 80000040 写 0x00001000 就会把 0x80000070 的值写入到蓝牙 core 里面,然后 80000040 会重置为 0

常用配置如下：
1. uart setting:
{0x8000005c, 0x03300000}
{0x80000040, 0x00000080}
详细定义见 uart 格式配置
2.uart baud setting
{0x80000060, 0x000e1000}
{0x80000040, 0x00000100}
设置串口波特率，详细定义见 uart 波特率设置
3.sysconfig setting
{0x80000070, 0x00000000}
{0x80000040, 0x00001000}
Config setting 为：

```
#define SYS_CONFIG_FLAG_DISABLE_ESCO_LINK            (0x0008)
#define SYS_CONFIG_FLAG_DISABLE_SCATTERNET           (0x0010)
#define SYS_CONFIG_FLAG_IGNORE_CONNECT_PARAMETER     (0x0040)
#define SYS_CONFIG_FLAG_FORCE_SCO_VIA_HCI            (0x0080)
#define SYS_CONFIG_FLAG_DISABLE_FIX_SCO_CFG          (0x0200)
#define SYS_CONFIG_FLAG_ENABLE_SCO_AUTO_RATE_CFG     (0x0400)
#define SYS_CONFIG_FLAG_ENABLE_UNSLEEP_WITH_UNSNIFF_CFG     (0x0800)
#define SYS_CONFIG_FLAG_SHORT_NAME_EVENT              (0x10000)
#define SYS_CONFIG_FLAG_FORCE_TO_ESCO                    (0x20000)
```

```
/* bit 15~13 save the value of TM register */
#define SYS_CONFIG_TM_HEADSET                    (0x0000)
#define SYS_CONFIG_TM_HID                        (0x2000)
#define SYS_CONFIG_TM_CORDLESS                   (0x4000)
#define SYS_CONFIG_TM_USB                        (0x0000)
#define SYS_CONFIG_TM_UART_M                     (0x2000)
#define SYS_CONFIG_TM_UART_C                     (0x4000)
#define SYS_CONFIG_TM_UART                       (0x6000)
#define SYS_CONFIG_TM_SPI                        (0x8000)
#define SYS_CONFIG_TM_5400                       (0xA000)
#define SYS_CONFIG_TM_BIST                       (0xC000)
#define SYS_CONFIG_TM_SCAN                       (0xE000)
```

4.sleep setting：

{0x80000074, 0xa5025010}

{0x80000040, 0x00002000}

Bit31: LowFrequencyOscillatorAvailable 使能睡眠的时候设置为 1，关闭睡眠的时候写 0

Bit30：写 0

Bit29：sleep enable

Bit28-24 delay time 默认写 5

Bit23-16 g_loop_redundancy_counter 蓝牙循环空转次数 默认为 2

Bit15-8 standby_sleep_period 蓝牙 idle 状态下，多长时间才睡，默认为 0x50

Bit7-0 min_sleep_interval 最小睡眠间隙 默认为 0x10

5.sniff interval setting

{0x80000078, 0x0f054001}

{0x80000040, 0x00002000}

Bit7-0 sniff mini interval 设置 sniff 最小 interval

6. pcm config

{0x80000084, 0x9098c007}

{0x80000040, 0x00020000}

详细配置见 pcm 配置

7. hostwake setting

{0x800000a4, 0x08b0280a}

{0x80000040, 0x02000000}

Bit31-20 g_host_wake_flag hostwake 的基本配置

Bit19-10 g_host_wake_length 非 clear mode 下 hostwake 信号有效时间

Bit9-0 g_host_wake_pause_length 非 clear mode 下，hostwake 信号恢复 idle 后等待时间

Hostwak 配置：

```
#define HOSTWAKE_METHOD_INTERRUPT          0x008
#define HOSTWAKE_BOTHEDGE                  0x004
#define HOSTWAKE_POLARITY                  0x002
#define HOSTWAKE_CLRMODE                   0x001
#define HOSTWAKE_ENABLE                    0x080
#define HOSTWAKE_IGNORE_NEEDBIT            0x040
#define HOSTWAKE_METHOD_PUSLE              0x020
#define HOSTWAKE_GPIO_PUSLE                0x010
```

寄存器读写说明

write reg:
command id(1) + command opcode(2) +length(1)+memory type(1)+memory num(1)+memory address(4)+memory data(4*memory num)
比如写 0x80000070=0x01020304
command id 固定不变 为 0x01
command opcode 0x02 0xfd
length 0x0a (后面所跟数据长度)
memory type 00
memory num 0x1
address 和 data 需要大小端翻转，由低到高:
memory address 0x70 0x00 0x00 0x80
memory data 0x04 0x03 0x02 0x01

所以完整发出去的串口命令为 0x01 0x02 0xfd 0x0a 0x00 0x01 0x70 0x00 0x00 0x80 0x04 0x03 0x02 0x01
事件返回：event id(1)+event opcode(1)+length(1)+command num cansend(1)+command opcode(2)+status
event id 固定不变 04
event opcede 0x0e
event length 0x04
command num cansend 0x01
command opcode 0x02 0xfd
status 0x00（no error）
所以蓝牙完整的回复为 04 0e 04 01 02 fd 00

如果是连续写内存的话
memory num =写的内存数
memory data 顺序往后面累计

比如写 0x80000070=0x01020304 0x80000074=0x05060708

0x01 0x02 0xfd 0x0e 0x00 0x02 0x70 0x00 0x00 0x80 0x04 0x03 0x02 0x01 0x08 0x07 0x06 0x05

memory type 为 0 表示写 core 部分地址，即 rom，ram，tab，gpio，scu 等寄存器地址
memory type 为 1 表示写 rf 部分地址，即 540x interface 地址或者 modem 部分地址

写 rf 地址的时候，需要进行地址转换，即（rf address*4+0x200）

比如写{0x0001,0x0123}
那么用串口写的地址就是 0x0001*4+0x200=0x204

所以，实际串口命令为 0x01 0x02 0xfd 0x0a 0x01 0x01 0x04 0x02 0x00 0x00 0x23 0x01 0x00 0x00

写 rf 地址的时候需要注意地址翻页，大于 0x80 的地址需要把 0x3f 寄存器写 1，然后把实际地址-0x80
例：
写 0x80=0x4567
那么需要变成
{0x3f,0x0001}
{0x00,0x4567}

地址仍然需要重新转换，实际串口命令为
0x01 0x02 0xfd 0x0a 0x01 0x01 0xfc 0x02 0x00 0x00 0x01 0x00 0x00 0x00
0x01 0x02 0xfd 0x0a 0x01 0x01 0x00 0x02 0x00 0x00 0x67 0x45 0x00 0x00

read reg：
command id(1) + command opcode(2) +length(1)+memory type(1)+memory num(1)+memory address(4)
比如 读取 0x80000070：
command id 0x01
command opcode 0x01 0xfd
length 0x06
memory type 0x00
memory num 0x01
memory address 0x70 0x00 0x00 0x80
完整串口数据为 0x01 0x01 0xfd 0x06 0x00 0x01 0x70 0x00 0x00 0x80

蓝牙芯片回复为：
event id(1)+event opcode(1)+length(1)+command num cansend(1)+command opcode(2)+status（1）+memory num（1）+memory address(4)+memory data(4*memory num)

event id = 0x04
event opcode = 0xe
length = 0xd
command num cansend 0x01
command opcode 0x01 0xfd
status 0x00 (no error)
memory num 0x01
memory address 0x70 0x00 0x00 0x80
memory data xx xx xx xx

ps：读取 rf 部分的地址和写一样，都需要进行地址转换和翻页

ps：在读写 rf 部分的寄存器的时候，需要主要把蓝牙内部 spi clock 打开
写{0x40240000,0x0004f39c}, //; SPI2_CLK_EN PCLK_SPI2_EN
然后再进行地址读写

DUT 测试模式进入

蓝牙如果需要进入 DUT 测试模式，那么需要对射频进行初始化，执行上电初始化后，进行 pskey 的配置：

```
{0x800000a4,0x00000000},//disable hostwake function
{0x800000a8,0x0bbfbf30},//set min power level
{0x80000040,0x06000000} //PSKEY: modify flag
```

然后写入测试模式 trap：

```
{0x40180100,0x000068b8},//inc power
{0x40180120,0x000069f4},
{0x40180104,0x000066b8},//dec power
{0x40180124,0x000069f4},
//{0x40180108,0x0002ec44},////for cbt test
//{0x40180128,0x00000014},
{0x4018010c,0x0000f8c4},///all rxon
{0x4018012c,0x00026948},
{0x40180000,0x00000b00},
//{0x40180000,0x00000700},
```

然后写入 3 条命令：

```
{0x01,0x03, 0x18, 0x00};//enable dut
{0x01, 0x1a, 0x0c, 0x01, 0x03}; //enable both scan
{0x01,0x05, 0x0c, 0x03, 0x02, 0x00, 0x02};///auto accept connect request
```

这样后，就可以连接仪器进行测试了，主要支持的蓝牙综测仪有 cbt,cmu200,n4010,tc3000,8852

非调制信号的输出：

Rda5875/6 支持非调制信号的全频段输出，输出方式如下：

1. 先初始化芯片，保证射频信号正常
2. 打开射频寄存器读写 clk：

{0x40240000,0x0004F398},//enable spi2

3.写射频寄存器

{0x0000003f,0x00000000},//page 0

{0x0000002b,0x000000b2},//3//2b

{0x00000002,0x0000f180},//设置频点为 0 信道

4.打开发射开关，发射信号

{0x40200020,0x00000041},//tx on

如果需要切换频段输出那么

1. 关闭射频发射

{0x40200020,0x00000000},//tx on

2. 写入需要发射的频点：

{0x00000002,0x0000f180},//

频点设置寄存器为 bit0-bit6，范围为 0-78，一共 79 个频点

比如信道 1，{0x00000002,0x0000f181},//1

信道 41，{0x00000002,0x0000f1a9},// 41

信道 78，{0x00000002,0x0000f1ce},// 78

3.打开发射开关

{0x40200020,0x00000041},//tx on

需要注意的是：射频寄存器的读写和 core 部分寄存器的读写是不一样的，详情参考前文寄存器读写说明

调制信号输出

Rda5875/6 支持调制信号的输出，输出方式如下：

一. 先初始化芯片，保证射频信号正常，然后写入测试模式配置文件

二. 然后发送命令：

01 10 FD 18 00 00 01 00 03 00 0F 00 33 1A 3A E2 4E 7A 2C CE FF 01 55 55 00 00 12 00

三. 命令格式：

Command OGF:0x3f

Command OCF 0X110

Command length: 0x18

Command parameters struct:

```
typedef struct bt_test_entry
{
    u_int8 t_hop_frq;   /* val: [0.....78]*/
    u_int8 t_msg_sco;   default 0
    u_int8 t_msg_burst; /* 0----burst; 1----continues*/
    u_int8 t_msg_type;
    u_int8 t_tx_packet_type;
    u_int8 t_EDR_mode;
    u_int8 t_power_level;
    u_int8 align1;
    u_int32 t_syncword_low;
    u_int32 t_syncword_hight;
    u_int16 t_PRbs_init;
    u_int16 t_msg_data;
    u_int8 t_am_addr;   /* 3 bits*/
    u_int8 align2;
    u_int16 t_tx_len;
} t_bt_test;
```

1.continues mode

当 t_msg_burst 为 1（continue mode)的时候,命令中有效的参数是 t_hop_frq t_msg_type t_EDR_mode t_power_level t_msg_data

parameter	value
t_hop_frq	Default 0-79 0-78 表示发射频点 79 表示按照 0 到 78 频点依次发射信号
t_msg_type	04： prbs9 Other ： user defined patten
t_EDR_mode	Default:0x00 0x00: bdr /gfsk 0x01: edr-2M /4-DQPSK 0X02: edr-3M /8-DPSK
t_power_level	Transmit power Default:0xf Max:0xf Min：0xd
t_msg_data	User defined pattern Default 1010101010101010b

需要注意的是，

1）.如果是单频发射（t_hop_frq == 0--78），那么可以通过 stop test 的命令来终止发射，如果是调频发射（t_hop_frq == 79），那么如果需要对芯片进行操作，需要拉低 ldoon，然后出现上电初始化后再进行操作

2）.在 continue 模式下，t_msg_type==04 的时候发射 prbs9 数据，如果不等于 4，那么发射的是 t_msg_data 里面的数据，t_msg_data 是 16bit 寄存器，默认值为 1010101010101010b

3）.在 continue 模式下，是没有包类型的概念的，芯片全时发射信号。

2. burst mode

当 t_msg_burst 为 0 （burst mode）的时候，命令中有效的参数定义：

序号(0x)	描述	Value
00 t_hop_frq	Hop freq	Default：00 Channel 1:0x01 Channel 2：0x02… Channel 78: 0x4e 79： freq changed auto
00 t_msg_sco	Data path	Default: 0x00
01 t_msg_burst	Burst or continues	0x00:burst 0x01:continues
00 t_msg_type	Data pattern	0x00:00000000 0x01:11111111 0x02:10101010 0x03:11110000 0x04:prbs9 0x05:user defined
03 t_tx_packet_type		Default:0x03 0x03:DM1 packet 0x04:DH1/2DH1 packet 0x08:DV/3DH1 packet 0x0A:DM3/2DH3 packet 0x0B:DH3/3DH3 packet 0x0e:DM5/2DH5 packet 0x0f:DH5/3DH5 packet
00 t_EDR_mode	EDR mode	Default:0x00 0x00: bdr /gfsk 0x01: edr-2M /4-DQPSK 0X02: edr-3M /8-DPSK
0f t_power_level	Power level	Default:0xf Max:0xf
00 align1	Pad for word align	Pad
33 1A 3A E2 4E 7A 2C CE t_syncword_low t_syncword_hight	Sync word	Defaut
FF 01 t_PRbs_init	PRbs init	default
55 55 t_msg_data	User data	Default:55 55
00 t_am_addr	Am address	Default:00 Value：0-7
00 align2	Pad for word align	Default:00
12 00 t_tx_len	Data length	Default :0x12

需要注意的是：

1）. 如果是单频发射（t_hop_frq == 0--78），那么可以通过 stop test 的命令来终止发射，如果是调频发射（t_hop_frq == 79），那么如果需要对芯片进行操作，需要拉低 ldoon，然后出现上电初始化后再进行操作

2）. 和 continues mode 不同的是 t_msg_type 0-5 全有效 当 t_msg_type=5 发射数据采用 t_msg_data 里面的数据

3）. t_tx_packet_type 和 t_EDR_mode 是同时操作的，比如说要选择 3dh1 的包类型，则需要把 t_tx_packet_type 设为 8，把 t_EDR_mode 设为 2.要设置 2dh5 的包类型，则需要 t_tx_packet_type =0x0e，t_EDR_mode=1

包类型的定义：dm1 dm3 dm5 dh1 dh3 dh5 为 1M 包，gfsk 调制模式

2dh1 2dh3 2dh5 为 2M 包 4-DQPSK 调制模式

3dh1 3dh3 3dh5 为 3M 包，8-DPSK 调制模式

四．停止命令

如果需要停止射频发射，发送命令：

01 12 FD 00 //stop test

当处于发射模式，需要更改参数重新进行发射的时候，通过串口先发送停止命令，然后再发送调制信号输出命令

PS:

调制信号输出部分的说明，适用于 5876,5875y 芯片，要进行此测试模式，需要对应的配置文件，请联系 rda 工程师获取相关资料

接收模式

进入接收模式的方法是:

1.先初始化芯片，保证射频信号正常

2.打开射频寄存器读写 clk：

{0x40240000,0x0004F398},//enable spi2

3.写射频寄存器

{0x0000003f,0x00000000},//page 0

{0x0000002b,0x000000a2},//3//2b

{0x00000002,0x00000f80},//设置频点为 0 信道

4.打开接收开关，接收信号

{0x40200020,0x000000a7},//rx on

如果需要切换频段输出那么

1.关闭射频发射

{0x40200020,0x00000000},//rx on

2.写入需要接收的频点：

{0x00000002,0x00000f80},//

频点设置寄存器为 bit0-bit6，范围为 0-78，一共 79 个频点

比如信道 1，{0x00000002,0x00000f81},//1

信道 41，{0x00000002,0x00000fa9},// 41

信道 78，{0x00000002,0x00000fce},// 78

3.打开接收开关

{0x40200020,0x000000a7},//rx on

需要注意的是：射频寄存器的读写和 core 部分寄存器的读写是不一样的，详情参考前文寄存器读写说明

接收测试模式

现在只支持 pxi3000 的接收测试模式

1. 正常启动蓝牙初始化射频
2. 写入对应的测试补丁 patch-rx-test-uart
3. 让仪器通过固定包类型进入发射模式
4. 输入 cmd 01 77 fd 0c 01 00 08 00 00 00 00 01 e8 03 00 00

序号(0x)	描述	Value
01	Command id	Default：01
77 fd	command opcode	Default: 0x77 0xfd
0c	Length	Default：0xc
01	Enable	01：enable 00：disable
00	Freq channel	Default：00 Channel 1:0x01 Channel 2：0x02… Channel 78: 0x4e
08 00 00 00	Lap for sync word	Default:000008
00	Edr mode	Default:0x0 00：bdr mode 01：edr mode
01	Data Pattern	Default：0x1: 00： 00000000 01:10101010 02:11110000 03:11111111
e8 03 00 00	Total test packet number	Defaut :1000

5. 当芯片接收完 Total test packet number 个包后，会返回 event:

04 0e 14 01 77 fd 00 error packets received(4 bytes) total received packets(4 bytes) error bits(4 bytes) total received bits(4 bytes)