2928平台RDA5875Y蓝牙模块调试心得

Authe:李东汉

Email:ldh@rock-chips.com

Date:2012-12-19

目录

[2928平台RDA5875Y蓝牙模块调试心得 1](#_Toc343709001)

[一、 代码移植 1](#_Toc343709002)

[1) 补丁<RDA5876\_patch\_11\_22>的目录结构如下 1](#_Toc343709003)

[2) 移植注意点 2](#_Toc343709004)

[二、 软件简单流程 2](#_Toc343709005)

[三、 硬件调试 4](#_Toc343709006)

[四、 软件调试方法 5](#_Toc343709007)

[五、 总结 6](#_Toc343709008)

# 代码移植

## 补丁<RDA5876\_patch\_11\_22>的目录结构如下

├<device>

│ ├<rockchip>

│ │ ├<rk29sdk>

│ │ │ └init.rk29board.rc

├<external>

│ ├<bluetooth>

│ │ ├<bluez>

│ │ │ ├<tools>

│ │ │ │ ├Android.mk

│ │ │ │ ├hciattach.c

│ │ │ │ └hciattach\_rda5875\_5870E.c

├<frameworks>

│ ├<base>

│ │ ├<core>

│ │ │ ├<java>

│ │ │ │ ├<android>

│ │ │ │ │ ├<server>

│ │ │ │ │ │ └BluetoothAdapterStateMachine.java

│ │ ├<data>

│ │ │ ├<etc>

│ │ │ │ └handheld\_core\_hardware.xml

│ │ ├<services>

│ │ │ ├<java>

│ │ │ │ ├<com>

│ │ │ │ │ ├<android>

│ │ │ │ │ │ ├<server>

│ │ │ │ │ │ │ └SystemServer.java

├<kernel>

│ ├<drivers>

│ │ ├<bluetooth>

│ │ │ └hci\_ldisc.c

│ │ ├<misc>

│ │ │ ├Kconfig

│ │ │ ├Makefile

│ │ │ └tcc\_bt\_dev.c

│ ├<net>

│ │ ├<bluetooth>

│ │ │ └hci\_core.c

├<system>

│ ├<bluetooth>

│ │ ├<bluedroid>

│ │ │ ├Android.mk

│ │ │ └bluetooth.c

## 移植注意点

可阅读附带的readme.txt。另我在init.rk2928board.rc中chmod 0777 /dev/tcc\_bt\_dev后发现无效，随后移到ueventd.rc中去后OK。

# 软件简单流程

调试一个设备或者功能模块最重要的，莫过于清楚其走过的每一行代码，当然很多的系统调用没有必要去深究，也不会对特定模块的调试有任何影响，但必须清楚其实现的功能，输入输出数据含义。

**App（应用层）**

**Framework（系统框架层）**

**HAL层，再下面就是设备驱动层**

上面的应用层和框架层公共的东西暂且不需理会，重点看下HAL层。通过APP，framwork下来启动的函数为bluetooth.c中的int bt\_enable()，其中static int set\_bluetooth\_power(int on)为上下电操作函数，通过设备节点，最终会调用到kernel层tcc\_bt\_dev.c中的int tcc\_bt\_power\_control(int on\_off)来对IO进行设置。

property\_set("ctl.start", "hciattach")启动蓝牙服务。随后转到Hciattach.c 中的int main(int argc, char \*argv[])中，首先解析init.rc中定义的服务字段参数:即解析init.rk2928board.rc

service hciattach /system/bin/hciattach -n -s 115200 /dev/ttyS0 rda 1500000 flow

user bluetooth

disabled

oneshot

中的-n -s 115200 /dev/ttyS0 rda 1500000 flow

-s后为设备初始速度

/dev/ttyS0为设备节点

rda 相当于设备识别码

1500000 为设定的设备速度

flow 为使用流控

获得一些重要的参数后调用设备的初始化函数init\_uart(dev, u, send\_break, raw),再通过if (u->init && u->init(fd, u, &ti) < 0)调用static int rda\_init(int fd, struct uart\_t \*u, struct termios \*ti)—> int RDABT\_core\_Intialization(int fd)向设备写入厂商提供的初始化数据。

如果到此没有出现错误，说明设备初始化成功。回到bluetooth.c的bt\_enable()中，接下的代码是一个for循环，先建立一个bluetooth的套接字，然后通过ioctl来和bluez的代码来打开蓝牙设备，可以重试1000次。接下来的代码就要跑到内核的BlueZ了。Kernel/net/bluetooth/hci\_sock.c中的static int hci\_sock\_ioctl(struct socket \*sock, unsigned int cmd, unsigned long arg)

……

{

switch (cmd) {

case HCIDEVUP:

if (!capable(CAP\_NET\_ADMIN))

return -EACCES;

return hci\_dev\_open(arg);

……

}

……

}

hci\_dev\_open位于Kernel/net/bluetooth/hci\_core.c中，其调用的hdev->open(hdev)即我们在注册设备驱动时的open函数，位于driver/Bluetooth/Hci\_ldisc.c中的static int hci\_uart\_open(struct hci\_dev \*hdev)。代码到此，蓝牙算是真正打开了。

# 硬件调试

1).确认连接是否正确。UART的TX,RX是否接对，BT\_REG\_ON是否拉高? VBATT是否供电正常？26MHZ时钟还有32K时钟是否供给?

**2).** **用独立RTC IC输出来的CLK因为是开漏输出所以要加10K上拉，否则无法驱动5875Y。PMU是IO口给出来的所以不用上拉。（硬件曾峥解释）**

刚开始看了原厂给的文档说32K时钟可以不用，结果正是因为这个原因，调了差不多半个月。

3).只能发送不能接收。将蓝牙中断输出拉高，两个串口拉高

4). 流控主要是针对接收/发射缓冲而言,只要你AP的UART内部硬FIFO size小于480bytes,就需要接UART Flow control。

接法：

BT\_CTS接到AP\_RTS.

AP\_CTS接到地，BT\_RTS悬空。

注：流控低有效，两个可直接拉低先调试.

# 软件调试方法

 BT手动测试命令：  
1. 停止以下服务:  
  setprop ctl.stop bluetoothd  
  setprop ctl.stop hciattach    
2. 关闭BT电源  
  echo 0 > /sys/class/rfkill/rfkill0/state   
3. 打开BT电源  
  echo 1 > /sys/class/rfkill/rfkill0/state    
4. 运行hciattach服务  
　setprop ctl.start hciattach  
5. 打开hci0接口  
  hciconfig hci0 up

6. hciconfig –a

获取蓝牙设备的一些重要信息，如：蓝牙的物理地址，总线类型，协议类型等  
7. 扫描  
 hcitool scan

以上都是标准的蓝牙系统调用接口，具体的设备上下电的节点可能不一样，RDA5875Y的上下电方式为

关闭电源：echo 0 > dev/tcc\_bt\_dev

打开电源：echo 1 > dev/tcc\_bt\_dev

上面的命令是在调试的时候直接屏蔽app及framwork层，如同直接在终端直接调用HAL层的函数进行操作。以此排除framwork层带来的影响。所以这些命令都可以在HAL中找到对应的代码打开关闭服务即property\_set("ctl.start","hciattach")和property\_set("ctl.stop", hciattach)，上下电操作：set\_bluetooth\_power，……

# 总结

这个RDA蓝牙调了将近两周，所以有很深刻的体会。希望把一些调试过程中遇到的问题和调试的方法写出来，免得以后再有人调试时少走弯路。由于水平有限，难免有不当之处，若有发现请指出。一起进步！