SC8810 蓝牙驱动介绍

课程名称： SC8810 蓝牙驱动介绍

课程类别： 客户培训课程

课程目标:

1.Android中蓝牙体系结构
2.蓝牙移植与调试要点
3.客户化定制要点

对象： 客户
讲师： 驱动软件工程师

课时数： 1H
教学法： 面授

主要内容

蓝牙系统介绍

移植与调试

系统集成与客户化定制

1 Android 蓝牙简介

1. 系统简介
2. 系统架构
3. 相关代码
4. 移植内容

1.1 Android中蓝牙系统介绍

Android中蓝牙系统围绕bluez实现，他是linux平台上一套完整的蓝牙协议栈开源实现，Bluez的协议栈在Linux2.6内核中已经包含，而Bluez的用户空间实现，Android已经移植并嵌入到自身的平台上。

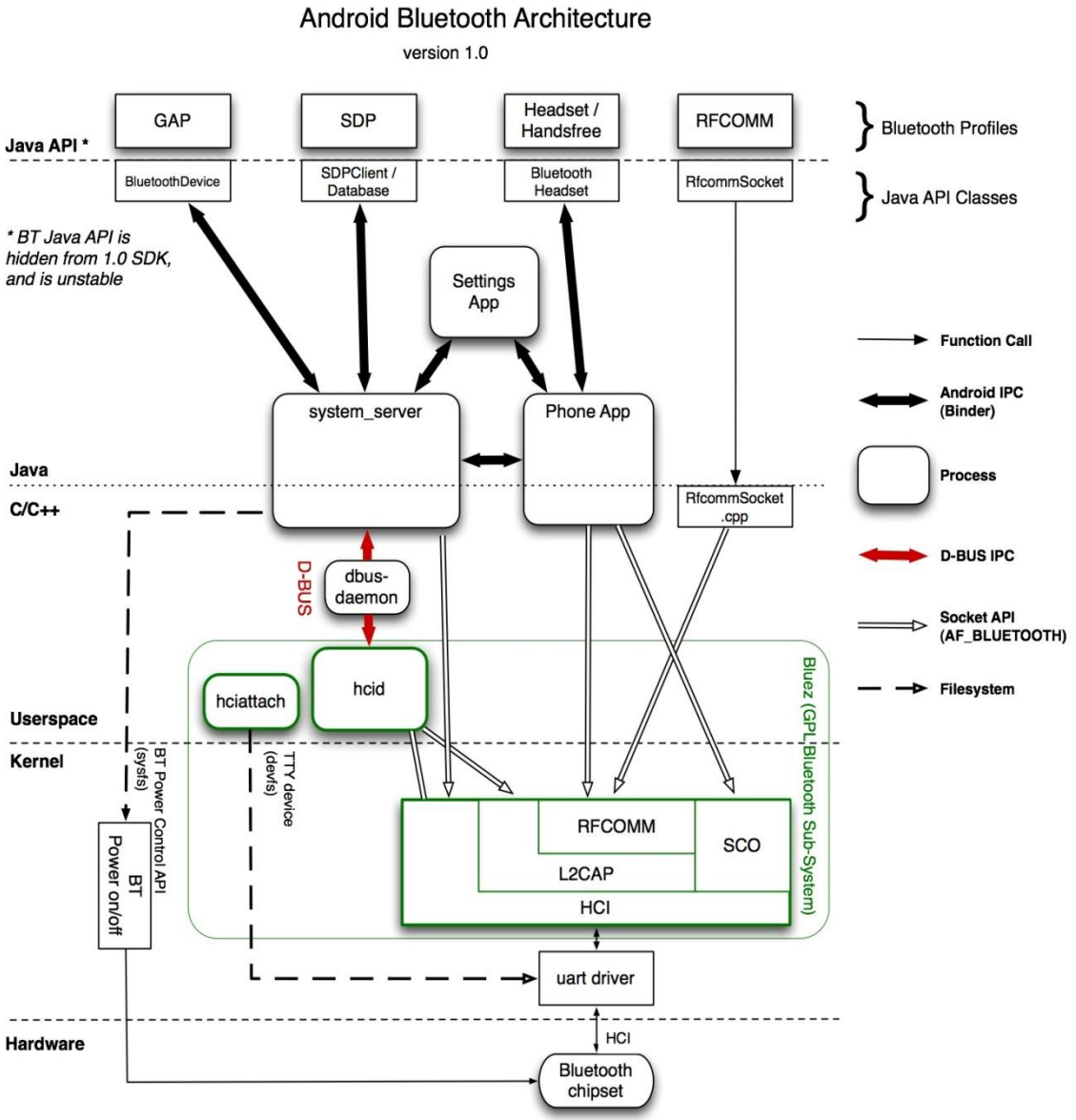
bluez 在 Android 中使用,需要经过 Android 的 bluez适配层的封装,bluez 适配层源代码及头文件路径如下所示:

system/bluetooth/

该目录除了包含生成适配层库 libbluedroid.so 的源码之外,还包含 bluez 头文件, bluez 配置文件等目录。

由于 bluez 使用 D-BUS 作为与上层沟通的接口,适配层构造比较简单,封装了蓝牙的开关功能,以及射频开关。

1.3 Bluetooth架构



2 移植和调试要点

串口驱动
电源管理
实现蓝牙的休眠与唤醒
编译与测试

2.1 串口驱动

串口驱动本身是CPU芯片相关的，和蓝牙本身没有关系，蓝牙系统利用串口和主芯片进行通讯。

BlueZ内核子系统通过守护进程hciattach与特定硬件的UART驱动进行绑定。

SC8810提供了一个可以使用的串口驱动

kernel/drivers/serial/serial_sc8810.c

这个串口驱动会用户空间生成一个名为/dev/ttyS0的设备节点，我们就可以对它进行操作，实现蓝牙功能

现有8810标准驱动

kernel/drivers/serial/serial_sc8810.c

```
static struct platform_driver serialsc8800_driver = {
        .probe  = serialsc8800_probe,
        .remove = serialsc8800_remove,
        .suspend = serialsc8800_suspend,
        .resume  = serialsc8800_resume,
        .driver  = {
                .name = "serial_sp",
                .owner = THIS_MODULE,
        },
};
```

流控的THRESHOLD

由于BCM4330使用了流控，THRESHOLD定义为0x40最为合理

```
/*flow control */
#define RX_HW_FLOW_CTL_THRESHOLD  0x40
#define RX_HW_FLOW_CTL_EN         (0x1<<7)
#define TX_HW_FLOW_CTL_EN         (0x1<<8)
```

这里如果不对，难以实现高波特率的传输

suspend


```
static int serialsc8800_suspend(struct platform_device *dev, pm_message_t
state)
{
    /*when the uart going to sleep,config the RTS pin of hardware flow
    control as the SPICLK to make the pin can be set to high*/
    unsigned long fc = 0;
    struct uart_port *port = &(serialsc8800_ports[0]);
    unsigned long u0rts_cfg[]= {
        MFP_CFG_X(U0RTS, AF2, DS1, F_PULL_UP, S_PULL_UP, IO_OE), //  BT_RTS
          };
    fc=serial_in(port,ARM_UART_CTL1);
    fc &=~(RX_HW_FLOW_CTL_EN|TX_HW_FLOW_CTL_EN);
    serial_out(port,ARM_UART_CTL1,fc);
    sprd_mfp_config(u0rts_cfg, ARRAY_SIZE(u0rts_cfg));
    return 0;
}
```

resume

```
static int serialsc8800_resume(struct platform_device *dev)
{
    /*when the uart waking up,reconfig the RTS pin of hardware flow control work in
    the hardware flow control mode to make the pin can be controlled by hardware*/
    unsigned long fc = 0;
    struct uart_port *port = &(serialsc8800_ports[0]);
    unsigned long u0rts_cfg[]= {
        MFP_CFG_X(UORTS, AF0, DS1, F_PULL_DOWN, S_PULL_UP, IO_OE),  //  BT_RTS
     };
     sprd_mfp_config(u0rts_cfg, ARRAY_SIZE(u0rts_cfg));
     fc=serial_in(port,ARM_UART_CTL1);
     fc |=(RX_HW_FLOW_CTL_EN|TX_HW_FLOW_CTL_EN);
     serial_out(port,ARM_UART_CTL1,fc);
     if(1)
     {
         wake_lock_timeout(&uart_rx_lock, HZ / 5);     // 0.2s
     }
     return 0;
}
```

2.2 电源管理

Android使用标准的Linux rfkill接口进行蓝牙芯片的电源管理。使用这种方式需要实现一个平台设备platform_device，并实现rfkill的控制逻辑。
rfkill的接口在下面的文件中定义。
./include/linux/rfkill.h

电源管理（续）

首先，要为蓝牙设备分配控制结构，

```
struct rfkill * __must_check rfkill_alloc(const char *name,
                                          struct device *parent,
                                          const enum rfkill_type type,
                                          const struct rfkill_ops
*ops,
                                          void *ops_data);
```

rfkill_type配置为 RFKILL_TYPE_BLUETOOTH

rfkill_ops实现对设备的操作

```
struct rfkill_ops {
        void    (*poll)(struct rfkill *rfkill, void *data);
        void    (*query)(struct rfkill *rfkill, void *data);
        int     (*set_block)(void *data, bool blocked);
};
```

简单的电源操作只需要实现 set_block接口即可，一般在这里需要实现蓝牙设备的具体开关工作。

电源管理（续） SC8810的实现

下面我们看看SC8810的实现
具体文件./arch/arm/mach-sc8810/rfkill_dummy.c

```
static struct rfkill_ops rfkill_ops = {
    .set_block = rfkill_set_power,
};
static int rfkill_set_power(void *data, bool blocked)
{
//    printk("rfkill to %d\n", blocked);
       return 0;
}
    rfkill = rfkill_alloc(rfkill_pdev->name, &rfkill_pdev->dev,
                                RFKILL_TYPE_BLUETOOTH,
&rfkill_ops,NULL);
```

从代码我们也可以看到这里直接return 0，没有具体操作。
要说明的是我们的蓝牙供电并不在这里实现。一般情况下，开机后都是一直供电的。

2.3 蓝牙的休眠与唤醒

bluesleep.c：A new driver is implemented to actively manage the bluetooth module power. bluesleep also tries to manage the power of the transport used.Two signals (GPIOs) are used to manage the power events.

蓝牙的休眠与唤醒

1、休眠过程：首先初始化一个定时器。使用bluesleep_tx_timer_expire函数定时检测是否可以睡眠 ，用来检测蓝牙是否有数据传输，如果没有数据传输，则关闭蓝牙相应的设备，使蓝牙进入sleep状态。

2、 唤醒过程：我们知道BT芯片控制host_wake管脚。它的基本原则是当我们蓝牙芯片不工作的时候，host_wake为1，当我们蓝牙芯片工作的时候这个芯片的管脚置0。

如果蓝牙开始工作，host_wake由高变低，中断函数会调用bluesleep_sleep_work主功能函数，清除BT_ASLEEP标志位，同时拉低

引脚的定义

kernel/arch/arm/mach-sc8810/common.c

BT2AP_WAKE: signal from BT chip to HOST to intimate HOST should wakeup/activate the transport modules required for BT communication.

AP2BT_WAKE: signal from HOST to BT chip to intimate BT chip can sleep.

```
#define BT2AP_WAKE 101
#define AP2BT_WAKE 103
```

编译与测试

实现了串口驱动和电源管理后，我们就进行系统编译，并进行简单的测试了

编译开关

编译Android打开蓝牙支持

code：3rdparty/products/sp8810ga/BoardConfig.mk

添加一下两个宏

```
BOARD_HAVE_BLUETOOTH := true
BOARD_HAVE_BLUETOOTH_BCM := true
```

简单手动测试

broadcom公司实现一个自己的工具，brcm_patchram_plus
/system/bluetooth/brcm_patchram_plus/brcm_patchram_plus.c
这个文件由broadcom公司提供，这里面涉及一些参数，比如PCM的设置等等，这个需要原厂配合调试。
还有一个 hcd文件，我们这里是bcm4330.hcd，这个文件也是和原厂沟通提供。

测试步骤

```
brcm_patchram_plus --enable_hci --enable_lpm --
no2bytes \
    --tosleep 50000 --baudrate 3000000
    use_baudrate_for_download  \
    --patchram /system/bin/bcm4330.hcd /dev/ttyS0
```

这里面涉及的参数，大家可以看看brcm_patchram_plus.c文件，里面有详细的说明。

hciconfig hci0 up 初始化完成后将其激活

hcitool scan 扫描当前激活的其他蓝牙，如果能发现，说明正常

系统集成与客户化定制要点

1. 系统集成
2. 客户化定制要点

```
# brcm bcm4330
    chmod 777 /dev/ttyS0
    chmod 666 /proc/bluetooth/sleep/proto

service hciattach /system/bin/brcm_patchram_plus --enable_hci
--enable_lpm --no2bytes \
       --tosleep 50000 --baudrate 3000000 --
use_baudrate_for_download  \
       --patchram /system/bin/bcm4330.hcd /dev/ttyS0
    class main
    user bluetooth
    group bluetooth net_bt_admin
    disabled
    oneshot
```

谢谢