Ralink RT3070 WiFi 驱动配置指南

版本号	修改日期	修改人	描　　述
R01	2010-12-27	赖永乐	提供文档以方便用户配置基于 Ralink RT3070 的 USB WiFi。
R02	2010-12-28	赖永乐	添加在 RK2818 中 USB HOST 1.1 使用 RT3070 WiFi 的支持。

目录

1 版本说明

1.1 Version 2.07

- 改进内存分配机制，避免出现内存分配失败而引起驱动出错的可能性。
- 添加了对隐藏 SSID 的支持。(**需要更新 wpa_supplicant 到最新的 1.07 版本**)
- 支除一些无用的代码，使驱动更加精简。
- 添加 USB 切换相关的代码，以便 WIFI 可以和 USB 切换紧凑结合在一起，使 WIFI 的开、关逻辑更加合理。这需要改动 USB 驱动中的代码。
- 将 USB 设备的 VID PID 数据结构，放出来，以便用户可以自己添加。

2 程序编译

2.1 WiFi 驱动

请将 WiFi 驱动文件，如 rt3070-2.07-20101227.tgz，拷贝到 linux/drivers/net/wireless 目录下，如果原来已经有 rt3070 的目录，则请将其改名，如 rt3070.old。

将 WiFi 驱动文件解压，会创建一个新的 rt3070 目录。

如果原来就已经有配置过 Ralink RT3070 的驱动，则请将原来的 wifi_power.c 拷贝到新的 rt3070 目录中，这样可以跳过 wifi_power.c 的重新配置。

注意，在拷贝覆盖前，请比较两个版本的 wifi_power.c，确认是可以覆盖的。

2.2 USB 驱动

由于 Ralink RT3070 驱动是基于 USB 接口，且要求 USB 工作在 HOST 模式下。为了让 WiFi 的开关可以和 USB 模式的切换紧密结合起来，可能需要我们修改 USB 驱动中相关的代码。

如果不想修改 USB 驱动，则请修改 wifi_power.c 接口，将相关的函数屏蔽，具体请参考“驱动配置”章节。

2.2.1 RK2808

- 修改 linux/drivers/usb/dwc_otg/dwc_otg_driver.c，在函数 force_usb_mode_store 之后添加如下内容：

```
#ifdef CONFIG_RALINK_RT3070

extern int usb_wifi_status;

static ssize_t usb_wifi_status_show(struct device_driver *_drv, char *_buf)
{
    return sprintf (_buf, "%d\n", usb_wifi_status);
}
```

```
int usb_force_usb_for_wifi(int mode)
{
     int ret = 0;

    dwc_otg_device_t *otg_dev = lm_get_drvdata(g_lmdev);
    dwc_otg_core_if_t *core_if = otg_dev->core_if;

    if(core_if->usb_mode != mode)
         ret = 1;

     if (mode == USB_MODE_NORMAL)
         force_usb_mode_store(NULL, (const char *)"0", 0);
     else if (mode == USB_MODE_FORCE_DEVICE)
         force_usb_mode_store(NULL, (const char *)"2", 0);
     else if (mode == USB_MODE_FORCE_HOST)
         force_usb_mode_store(NULL, (const char *)"1", 0);

     return ret;
}
EXPORT_SYMBOL(usb_force_usb_for_wifi);

Static DRIVER_ATTR(usb_wifi_status, 0666, usb_wifi_status_show, NULL);

#endif
```

- 修改 linux/drivers/usb/dwc_otg/dwc_otg_driver.c 中的函数 dwc_otg_driver_init：

```
#ifdef DWC_BOTH_HOST_SLAVE
     retval = driver_create_file(&dwc_otg_driver.drv, &driver_attr_force_usb_mode);
#ifdef CONFIG_RALINK_RT3070
     retval = driver_create_file(&dwc_otg_driver.drv, &driver_attr_usb_wifi_status);
#endif
#endif
```

2.2.2 RK2818 OTG

- 修改 linux/drivers/usb/dwc_otg/dwc_otg_driver.c，在函数 force_usb_mode_store 之后添加如下内容：

```
#ifdef CONFIG_RALINK_RT3070

extern int usb_wifi_status;

static ssize_t usb_wifi_status_show(struct device_driver *_drv, char *_buf)
{
     return sprintf (_buf, "%d\n", usb_wifi_status);
}

int usb_force_usb_for_wifi(int mode)
```

```
{
    int ret = 0;

    dwc_otg_device_t *otg_dev = g_otgdev;
    dwc_otg_core_if_t *core_if = otg_dev->core_if;

    if(core_if->usb_mode != mode)
        ret = 1;

    if (mode == USB_MODE_NORMAL)
        force_usb_mode_store(NULL, (const char *)"0", 0);
    else if (mode == USB_MODE_FORCE_DEVICE)
        force_usb_mode_store(NULL, (const char *)"2", 0);
    else if (mode == USB_MODE_FORCE_HOST)
        force_usb_mode_store(NULL, (const char *)"1", 0);

    return ret;
}
EXPORT_SYMBOL(usb_force_usb_for_wifi);

Static DRIVER_ATTR(usb_wifi_status, 0666, usb_wifi_status_show, NULL);

#endif
```

- 修改 linux/drivers/usb/dwc_otg/dwc_otg_driver.c 中的函数 dwc_otg_driver_init：

```
#ifdef DWC_BOTH_HOST_SLAVE
    retval = driver_create_file(&dwc_otg_driver.drv, &driver_attr_force_usb_mode);
#ifdef CONFIG_RALINK_RT3070
    retval = driver_create_file(&dwc_otg_driver.drv, &driver_attr_usb_wifi_status);
#endif
#endif
```

上面蓝色字体部分是 RK2818 的 OTG 驱动中添加的代码，与 RK2808 区别的地方。

2.2.3 RK2818 HOST 1.1

要使用 HOST 1.1 接口支持 WiFi，在打开 WIFI 前我们需要使用 HOST 1.1 接口：

在变量定义语句：static int s_host11_enable = 0; 的后面，添加如下内容：

```
#ifdef CONFIG_RALINK_RT3070
int usb_switch_usb_host11_for_wifi(int enabled)
{
    if (enabled == s_host11_enable)
        return 0;

    if (enabled == 1)
```

```
            rk28_host11_driver_enable();
        else
            rk28_host11_driver_disable();

        s_host11_enable = enabled;

        return 0;
    }
    EXPORT_SYMBOL(usb_switch_usb_host11_for_wifi);
    #endif
```

2.3 Android 应用

2.3.1 Android 1.5

为了在 USB WIFI 正在工作时，我们不去切换 OTG 的模式，以造成驱动逻辑上的混乱，我们可以修改设备中的 USB 模式切换工具：

- 修改 Settings.apk，其源码中的 USBmodeselect.java 需要修改：

  添加变量 private File wififile;

  打开文件 wififile = new File("/sys/devices/lm0/driver/usb_wifi_status");

  检查 WiFi 是否开启：

```
try{
    fin= new FileInputStream(wififile);
    reader= new BufferedReader(new InputStreamReader(fin));
    line = reader.readLine();
}catch(IOException re){
}
Log.d(TAG, "usb wifi status="+line);
int status = Integer.valueOf(line);
if(status == 1)
    cb_mode[0].setEnabled(false);
```

2.3.2 Android 2.1

为了在 USB WIFI 正在工作时，我们不去切换 OTG 的模式，以造成驱动逻辑上的混乱，我们可以修改设备中的 USB 模式切换工具：

- 修改 Settings.apk，其源码中的 USBmodeselect.java 需要修改：

  添加变量 private File wififile;

  打开文件 wififile = new File("/sys/devices/platform/dwc_otg/driver/usb_wifi_status");

  检查 WiFi 是否开启：

```
try{
    fin= new FileInputStream(wififile);
    reader= new BufferedReader(new InputStreamReader(fin));
    line = reader.readLine();
}catch(IOException re){
}
Log.d(TAG, "usb wifi status="+line);
```

```
int status = Integer.valueOf(line);
if(status == 1)
    cb_mode[0].setEnabled(false);
```

3 驱动配置

3.1 内核配置

3.1.1 DMA 空间配置

在 RK2818/RK2808 平台上测试，需要修改 include/asm/memory.h：

#define CONSISTENT_DMA_SIZE SZ_4M

否则会出现错误：[3843] Status = 3 after NICInitTransmit

3.1.2 USB HOST 1.1

如果在 RK2818 上要使用 HOST 1.1 连接 USB WIFI，则需要通过 make menuconfig，在 USB 驱动中将下面的选项选上：

RockChip USB Host 1.1 support

3.2 Wifi_power.c

3.2.1 WiFi 电源

有些产品上，会为 WIFI 使用一个 GPIO 来控制其供电，通常在 HOST 11 的情况下，这个是有必要的，即 HOST 11 会供电，但当 WiFi 没用时，我们可以不让 WiFi 上电，以达到节电的目的。

如果有使用 GPIO 控制电源，则需要修改 wifi_power.c 中的数据结构：

```
struct wifi_power power_gpio =
{
    0, 0, 0, 0, 0, 0
};
```

将其修改为正确的 GPIO 配置。

3.2.2 USB 模式切换

如果想实现在 WiFi 被打开时，会自动去切换 USB 的模式，则需要将这一功能打开，默认情况下，这一功能是关闭的。

```
#define DONT_SWITCH_USB 0    /* Don't switch USB automaticately. */
#define WIFI_USE_OTG    1    /* WiFi will be connected to USB OTG. */
#define WIFI_USE_HOST11 2    /* WiFi will be connected to USB HOST 1.1. */

#define WIFI_USE_IFACE    0
```

将其值定义为 0，即 DONT_SWITCH_USB，是指不在 WiFi 开、关时进行 USB 模式的自动切换。为这个常量设置正确的值，然后重新编译。

当然这一功能需要 USB 驱动中添加支持，具体看程序编译中的“USB 驱动”一节。

3.3 Wifi_config.c

4 驱动测试