

## 实验 06 设备注册

### 6.1 本章导读

在 Linux2.6 之后，引入了平台总线的概念，所以注册设备变得更加简单。为了使大家更快的入门，先介绍如何在平台总线中注册设备。

#### 6.1.1 工具

##### 6.1.1.1 硬件工具

- 1 ) iTOP4412 开发板
- 2 ) U 盘或者 TF 卡
- 3 ) PC 机
- 4 ) 串口

##### 6.1.1.2 软件工具

- 1 ) 虚拟机 Vmware
- 2 ) Ubuntu12.04.2
- 3 ) 超级终端 ( 串口助手 )
- 4 ) Ubuntu 系统下解压生成的 Linux 源码

#### 6.1.2 预备课程

内核的编译以及烧写等相关课程

Menuconfig、Kconfig 的使用等相关课程

### 6.1.3 视频资源

本节配套视频为 “视频 06\_设备注册”

## 6.2 学习目标

本章需要学习以下内容：

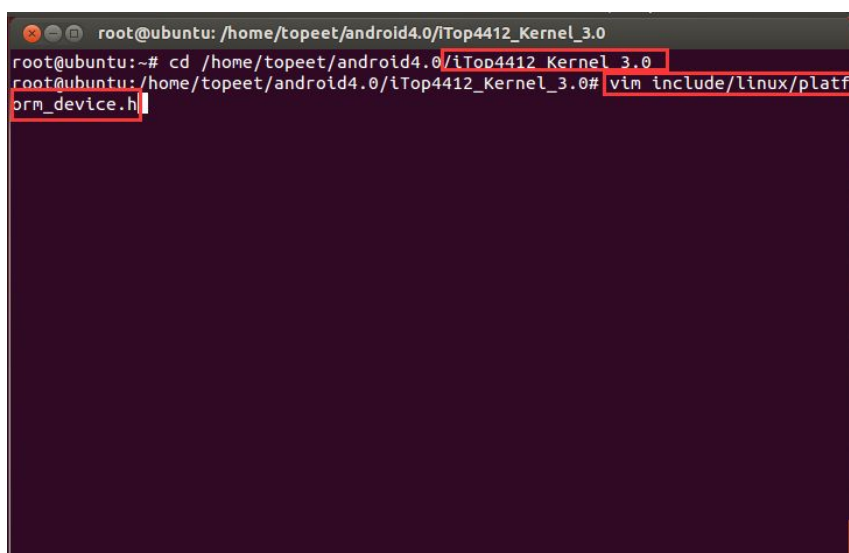
掌握在平台文件中注册设备的方法

### 6.3 在虚拟总线上注册设备

在实验 5 中介绍了注册设备的结构体 “platform\_device” ,并没有介绍 “platform\_device” 结构体的内部结构。

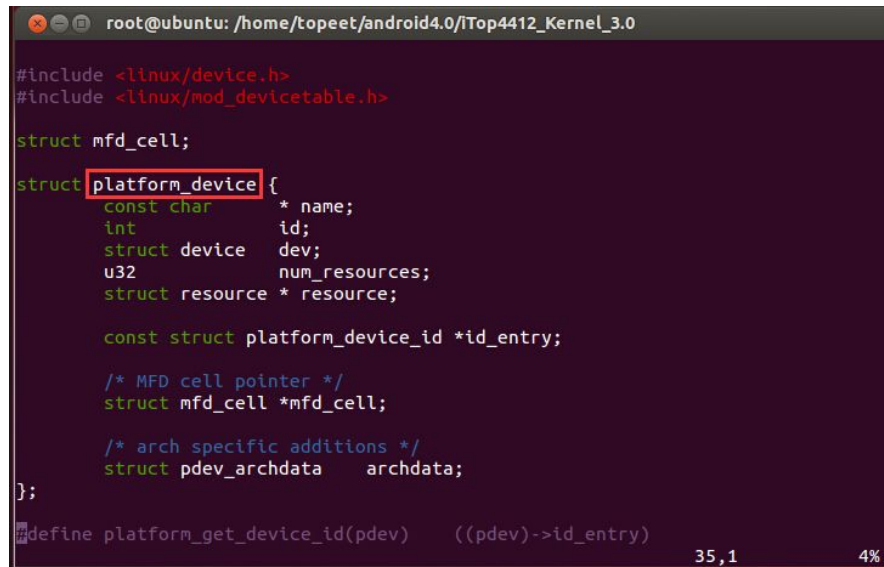
如下图所示，进入解压之后的内核文件夹 “iTop4412\_Kernel\_3.0” ，使用命令

“vim include/linux/platform\_device.h” 打开 “platform\_device” 所在文件。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:~# cd /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# vim include/linux/platform_device.h
```

如下图所示，就在第一页中，就可以看到结构体 “platform\_device” 。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
#include <linux/device.h>
#include <linux/mod_devicetable.h>

struct mfd_cell;

struct platform_device {
    const char * name;
    int id;
    struct device dev;
    u32 num_resources;
    struct resource * resource;

    const struct platform_device_id *id_entry;

    /* MFD cell pointer */
    struct mfd_cell *mfd_cell;

    /* arch specific additions */
    struct pdev_archdata archdata;
};

#define platform_get_device_id(pdev) ((pdev)->id_entry)
```

如下图所示，给结构体 “platform\_device” 添加了注释。

```
struct platform_device {
    const char * name; //设备名称，在 sys/devices 会显示
    int id; //设备 id，用于插入总线并且具有相同 name 的设备编号，
    如果只有一个设备那么-1
    struct device dev; //结构体中内嵌的 device 结构体
    u32 num_resources; //设备使用资源的数量
    struct resource * resource; //设备使用的资源数组

    const struct platform_device_id *id_entry;

    /* MFD cell pointer */
    struct mfd_cell *mfd_cell;

    /* arch specific additions */
    struct pdev_archdata archdata;
```

在结构体 “platform\_device” 中

第一个参数 “name” ，是一个字符指针，驱动初始化前需要和注册驱动的 “name” 字段匹配的参数；

第二个参数 “id” ，表示子设备编号，一个设备如果有多个子设备号，则需写入子设备号数量，如果只有一个则用-1 表示；

第三个参数 “device” ，表示结构体内嵌的设备结构体；

第四个参数 num\_resource ，表示设备使用的资源数组；

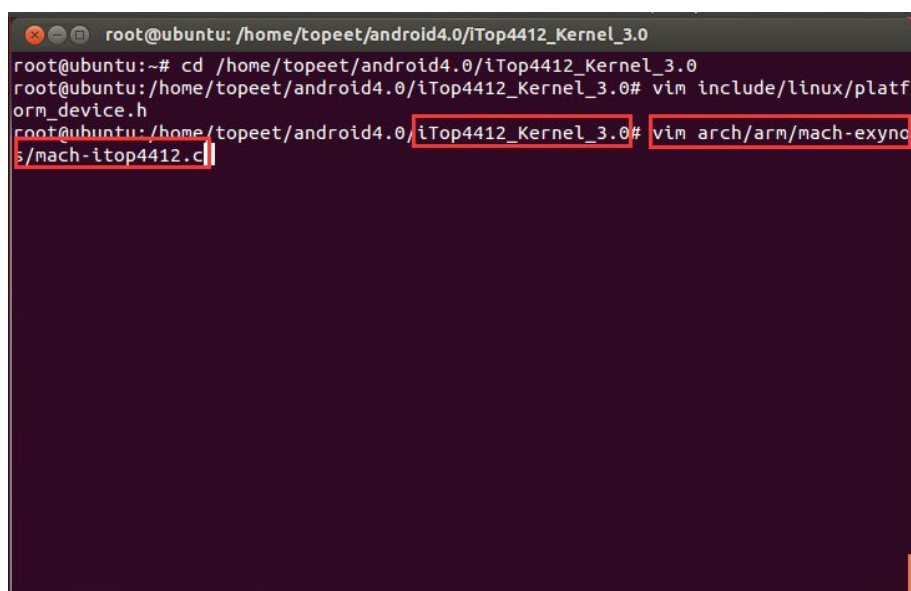
这些参数不一定全部使用，在大多数驱动中，需要写的只有设备名\*name 和设备编号 id，常用的还有资源数组\*resource，后面的参数在用到的时候再介绍。

## 6.4 添加设备到平台总线

在实验 5 中介绍了注册设备的流程，并没有详细介绍如何操作，像平台总线中添加 Linux 设备的方法很简单，下面就详细介绍一下基本步骤。

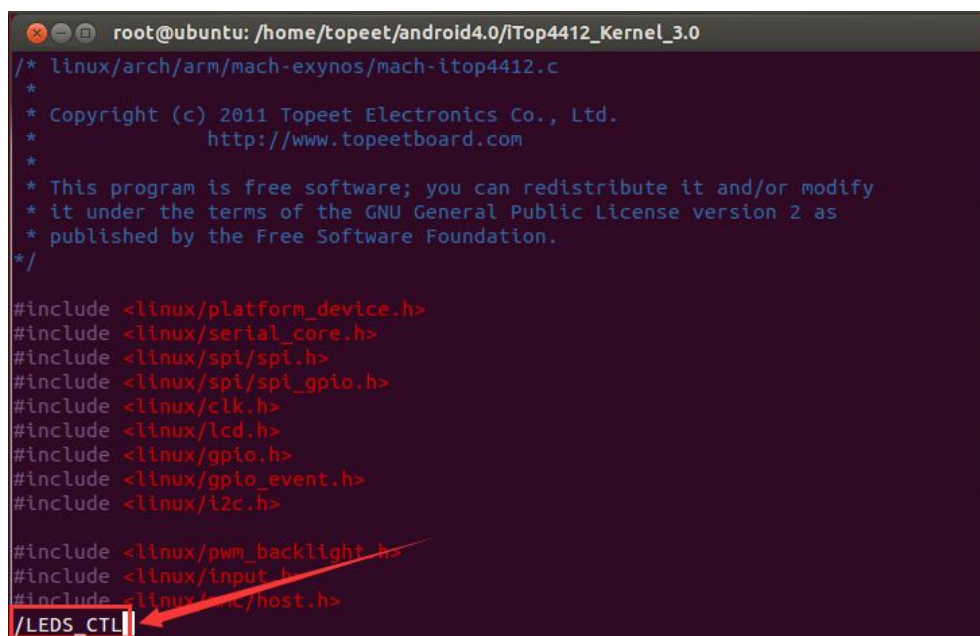
前面给大家介绍的平台文件 “arch/arm/mach-exynos/mach-itop4412.c”，并没有介绍这个文件和注册平台设备的关系。

如下图所示，使用命令 “vim arch/arm/mach-exynos/mach-itop4412.c” ，打开平台文件。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:~# cd /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# vim include/linux/platform_device.h
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# vim arch/arm/mach-exynos/mach-itop4412.c
```

现在介绍如何添加最简单的设备，led 的驱动相对来说很简单，在里面查找宏定义“LEDS\_CTL”，如下图所示。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
/* linux/arch/arm/mach-exynos/mach-itop4412.c
 *
 * Copyright (c) 2011 Topeet Electronics Co., Ltd.
 *      http://www.topeetboard.com
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 */

#include <linux/platform_device.h>
#include <linux/serial_core.h>
#include <linux/spi/spi.h>
#include <linux/spi/spi_gpio.h>
#include <linux/clock.h>
#include <linux/lcd.h>
#include <linux/gpio.h>
#include <linux/gpio_event.h>
#include <linux/i2c.h>

#include <linux/pwm_backlight.h>
#include <linux/input.h>
#include <linux/clk/host.h>
LEDS_CTL
```

如下图所示，找到 leds 注册设备的代码。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0

#ifdef CONFIG_LEDS_CTL
struct platform_device s3c_device_leds_ctl = {
    .name = "leds",
    .id = -1,
};
#endif

#ifdef CONFIG_BUZZER_CTL
struct platform_device s3c_device_buzzer_ctl = {
    .name = "buzzer_ctl",
    .id = -1,
};
#endif

struct platform_device s3c_device_keyboard = {
    .name = "samsung_keypad",
    .id = -1,
};

struct platform_device s3c_device_irq_test = {
    .name = "irq_test",
    .id = -1,
};
```

在上图红色方框中，注册平台设备结构体“platform\_device”中，只调用了两个参数“\*name”和“id”。

如下图所示，仿照着这段代码在它前面添加一个设备“hello\_ctl”。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0

    .name = "max485_ctl",
    .id = -1,
};
#endif

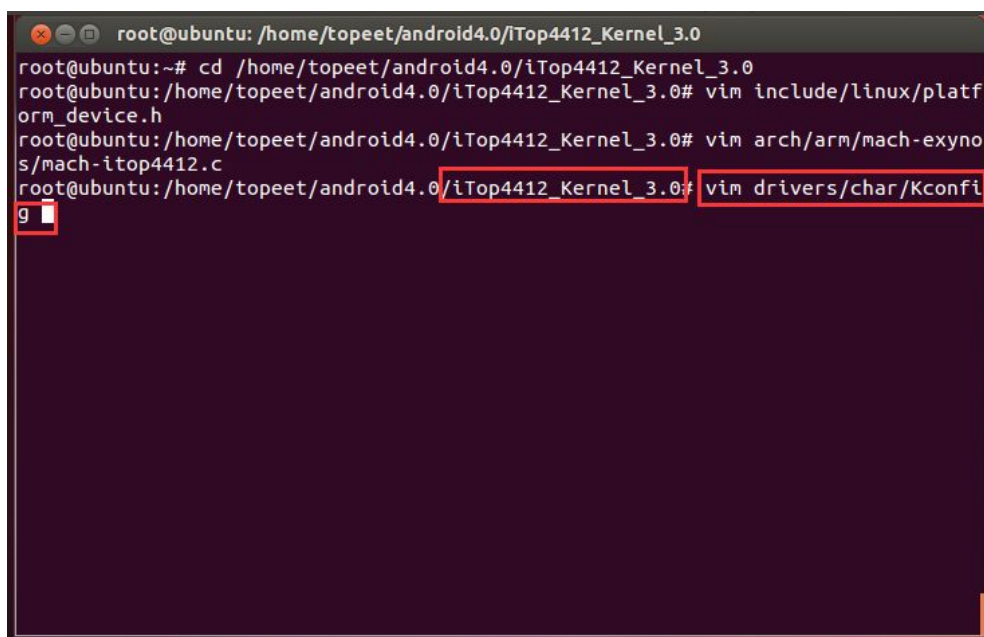
#ifdef CONFIG_HELLO_CTL
struct platform_device s3c_device_hello_ctl = {
    .name = "hello_ctl",
    .id = -1,
};
#endif

#ifdef CONFIG_LEDS_CTL
struct platform_device s3c_device_leds_ctl = {
    .name = "leds",
    .id = -1,
};
#endif

#ifdef CONFIG_BUZZER_CTL
struct platform_device s3c_device_buzzer_ctl = {
    .name = "buzzer_ctl",
    .id = -1,
};
```

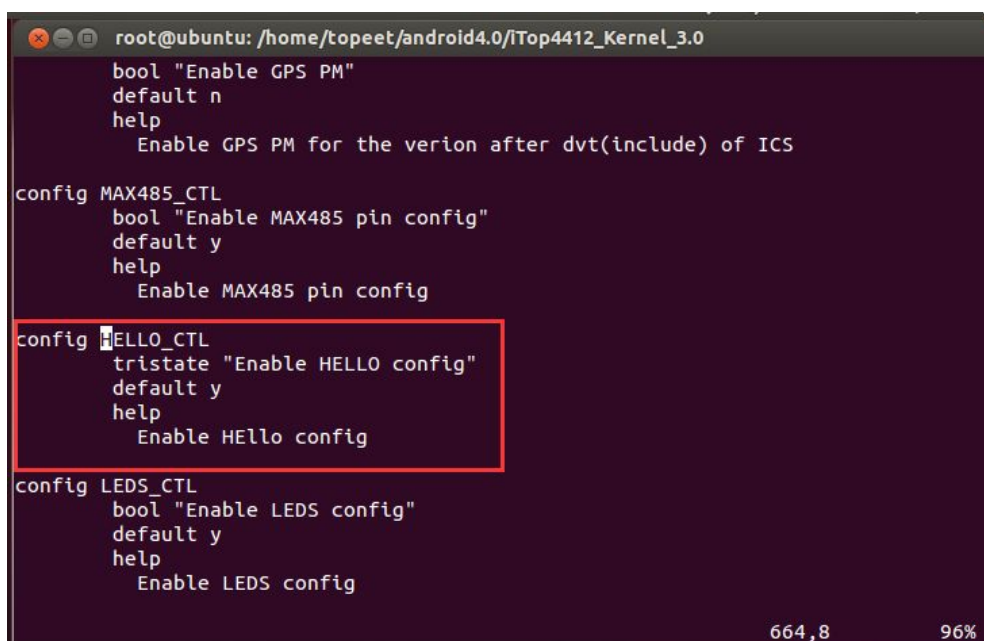
这里还需要确认一下，是否有“HELLO\_CTL”宏定义，只有定义了这个宏，在生成内核的时候才会将其编译到内核。

在前面关于 Kconfig 实验中，已经添加了“HELLO\_CTL”宏，如下图所示，使用命令“vim drivers/char/Kconfig”打开前面定义过“HELLO\_CTL”的配置文件。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:~# cd /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# vim include/linux/platform_device.h
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# vim arch/arm/mach-exynos/mach-itop4412.c
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# vim drivers/char/Kconfig
g
```

如下图所示，已经定义。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
bool "Enable GPS PM"
default n
help
    Enable GPS PM for the version after dvt(include) of ICS

config MAX485_CTL
bool "Enable MAX485 pin config"
default y
help
    Enable MAX485 pin config

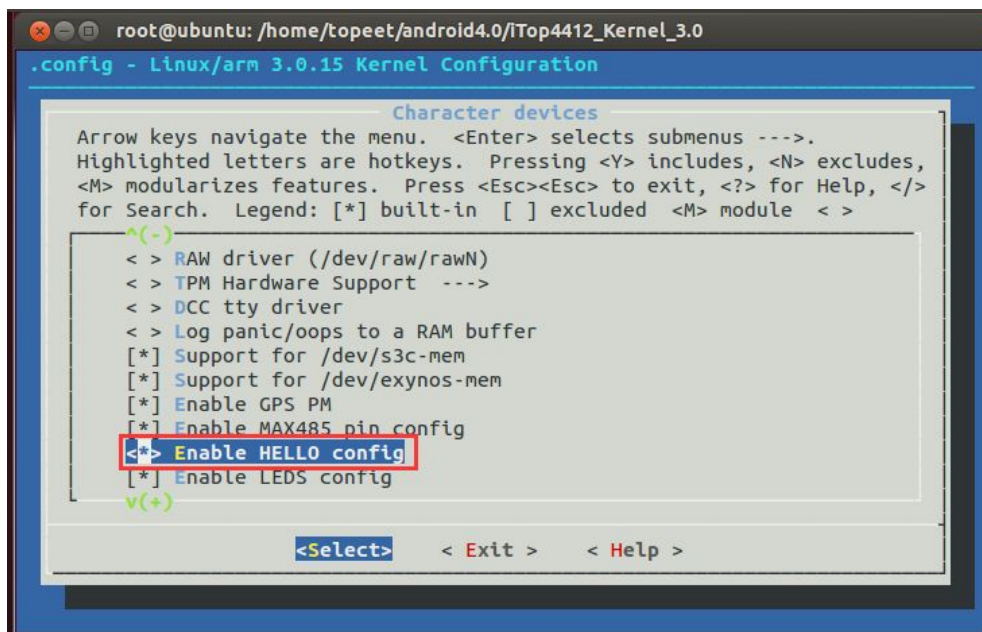
config HELLO_CTL
tristate "Enable HELLO config"
default y
help
    Enable Hello config

config LEDS_CTL
bool "Enable LEDS config"
default y
help
    Enable LEDS config

664,8 96%
```



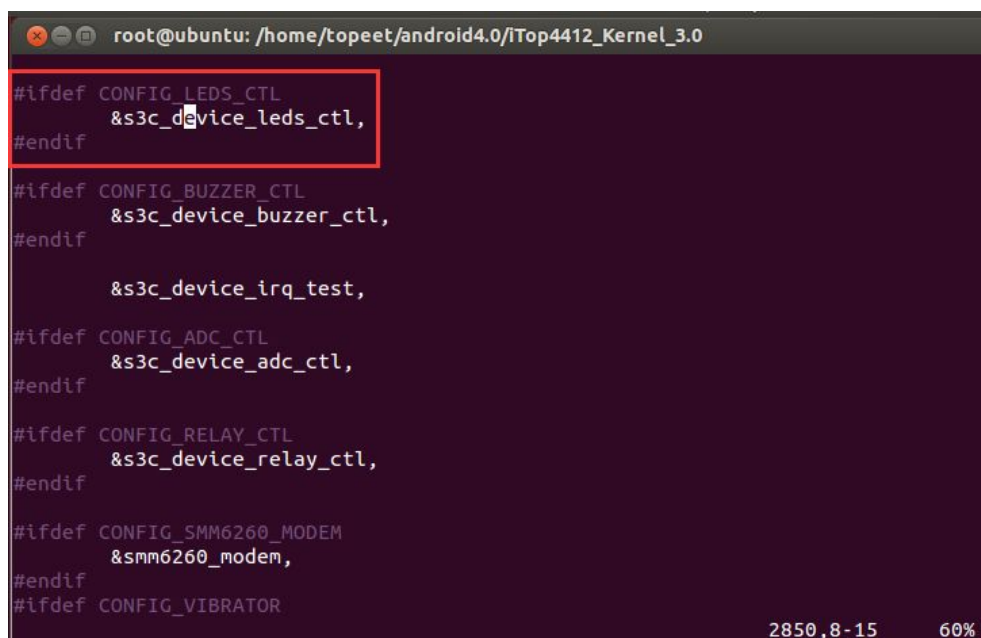
接着到 menuconfig 中将其配置上,使用命令 “make menuconfig”, 进入 Device Drivers --->” → “Character devices --->” → “Enable HELLO config” , 如下图所示, 配置上宏定义 “HELLO\_CTL” 。



配置后保存退出。这样就确认了宏定义 “HELLO\_CTL” 已经出现。

接着再次打开 “arch/arm/mach-exynos/mach-itop4412.c” 平台文件, 再搜索 “LEDS\_CTL” 。如下图所示, 查找到设备初始化的代码, 这一段比较简单, 仿照着写即可。





```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0

#ifdef CONFIG_LEDS_CTL
    &s3c_device_leds_ctl,
#endif

#ifdef CONFIG_BUZZER_CTL
    &s3c_device_buzzer_ctl,
#endif

    &s3c_device_irq_test,

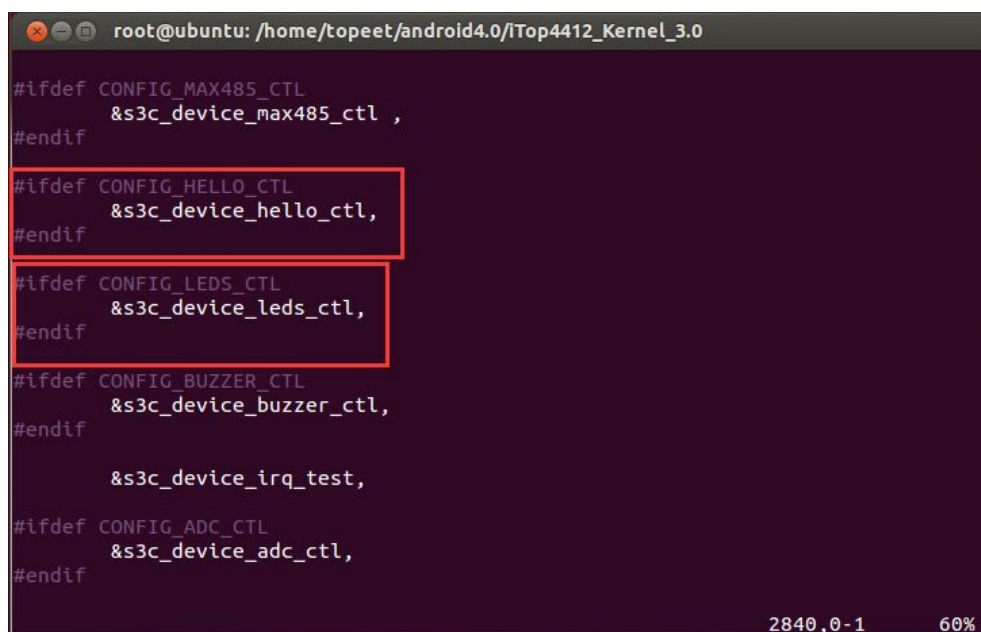
#ifdef CONFIG_ADC_CTL
    &s3c_device_adc_ctl,
#endif

#ifdef CONFIG_RELAY_CTL
    &s3c_device_relay_ctl,
#endif

#ifdef CONFIG_SMM6260_MODEM
    &smm6260_modem,
#endif
#ifdef CONFIG_VIBRATOR
```

2850,8-15 60%

如下图所示，添加 hello 设备的代码。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0

#ifdef CONFIG_MAX485_CTL
    &s3c_device_max485_ctl ,
#endif

#ifdef CONFIG_HELLO_CTL
    &s3c_device_hello_ctl,
#endif

#ifdef CONFIG_LEDS_CTL
    &s3c_device_leds_ctl,
#endif

#ifdef CONFIG_BUZZER_CTL
    &s3c_device_buzzer_ctl,
#endif

    &s3c_device_irq_test,

#ifdef CONFIG_ADC_CTL
    &s3c_device_adc_ctl,
#endif
```

2840,0-1 60%

保存退出，重新编译内核，烧写到开发板。

开发板启动之后，使用命令 “ls /sys/devices/platform/” 可以查看到新注册的 hello 设备，如下图所示。

```
[root@iTOP-4412]# ls /sys/devices/platform/
adc_ctl          relay_ctl        samsung-audio
alarm            s3c-pl1330.1    samsung-audio-idma
android_pmem.0   s3c-pl1330.2    samsung-i2s.0
android_pmem.1   s3c-sdhci.2     samsung-i2s.4
arm-pmu.0        s3c-sdhci.3     samsung-keypad
bt-sysfs         s3c-usb gadget  samsung-kmsg
buzzer_ctl       s3c2410-wdt     samsung-pd.0
dw_mmc           s3c2440-i2c.1   samsung-pd.1
exynos-busfreq   s3c2440-i2c.3   samsung-pd.2
exynos-usb-switch s3c2440-i2c.4   samsung-pd.4
exynos4412-adc   s3c2440-i2c.5   samsung-pd.5
gpio-keys        s3c2440-i2c.7   samsung-pd.6
hello_ctl        s3c24xx-pwm.1   samsung-pd.7
i2c-gpio.0       s3c64xx-rtc     samsung-rp
ion-exynos       s3c64xx-spi.2   serial8250
irq_test         s5p-ehci        si_gps
leds             s5p-pmic        snd-soc-dummy
max485_ctl       s5p-sysmmu.15   soc-audio
power            s5pv210-uart.0  switch-gpio.0
power.0          s5pv210-uart.1  tc4-regulator-consumer
reg-dummy        s5pv210-uart.2  uevent
regulatory.0     s5pv210-uart.3  wlan_ar6000_pm_dev.1
[root@iTOP-4412]#
```

