

实验 14 LEDS 驱动一

14.1 本章导读

本节实验介绍一个完整的 GPIO 驱动，以后在 Linux 中需要处理 GPIO 驱动都可以仿照或者移植这个驱动。

14.1.1 工具

14.1.1.1 硬件工具

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

14.1.1.2 软件工具

- 1) 虚拟机 Vmware
- 2) Ubuntu12.04.2
- 3) 超级终端（串口助手）
- 4) 源码文件夹“leds”

14.1.2 预备课程

实验 12_物理地址虚拟地址

实验 13_GPIO 初始化

14.1.3 视频资源

本节配套视频为“视频 14 LEDS 驱动一”

14.2 学习目标

本章需要学习以下内容：

Led 硬件原理简单介绍

Led 管脚的调用、赋值以及配置

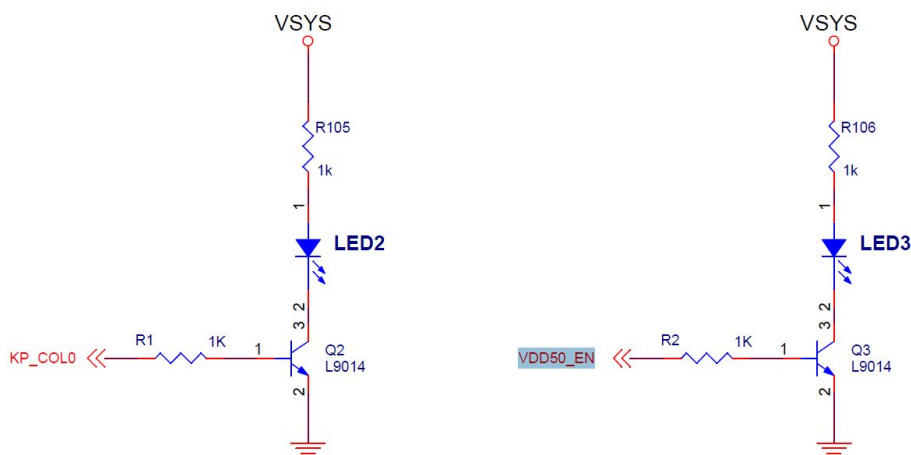
编写简单应用调用 LED 管脚，并测试

14.3 Led 硬件原理简单介绍

Led 的电路比较简单，一般是使用三极管搭建一个控制电路。

如下图所示，是原理图中两个 Led 的控制电路。KP_COL0 和 VDD50_EN 网络控制 Led 的通断。

LED



如上图所示。

当 KP_COL0 和 VDD50_EN 网络时高电平的时候，三极管 L9014 的 BE 导通，CE 导通，相当于 5V 的 VSYS 电压加到 1K 和 Led 小灯上，小灯就会亮。

当 KP_COL0 和 VDD50_EN 网络时低电平的时候，三极管 L9014 的 BE 会截止，CE 截止，相当于 5V 的 VSYS 电压加到 1K、Led 小灯和一个无限大的电阻上，电流为零，小灯就会灭。

14.4 Led 管脚的调用、赋值以及配置

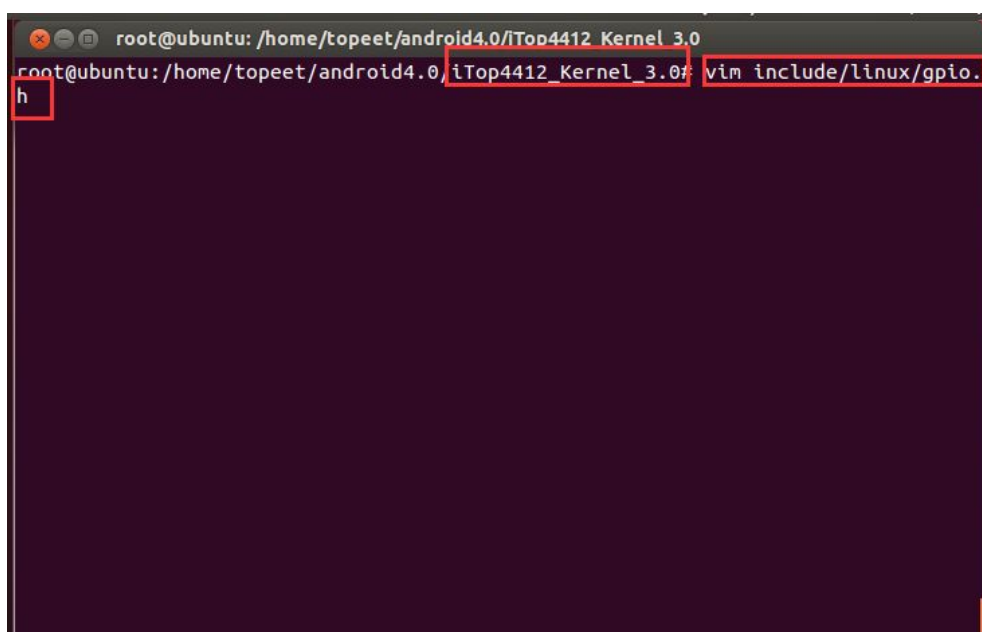
本节给大家介绍一部分涉及 GPIO 调用、赋值以及配置的函数。

14.4.1 GPIO 申请和释放函数

想用使用任何一个 GPIO 都必须先申请。

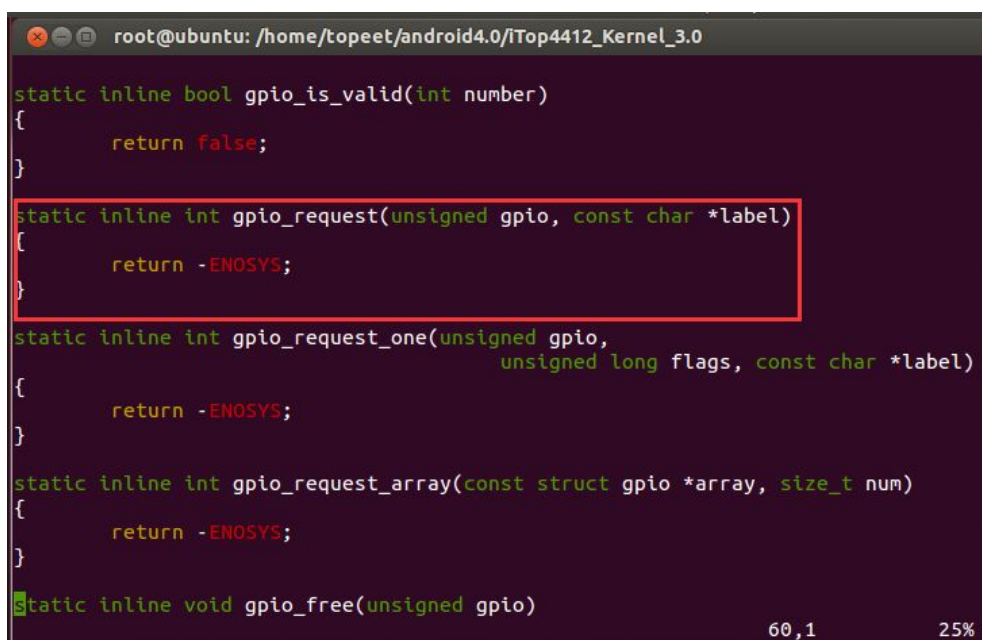
在头文件 “include/linux/gpio.h” 中有 Linux 默认的 GPIO 申请函数，这个头文件是属于嵌入式 Linux 平台，任何一个嵌入式 Linux 内核都可以这么使用。

如下图所示，在源码目录中使用命令 “vim include/linux/gpio.h” 打开该文件。



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

如下图所示，就是本节实验中需要用到的函数 `gpio_request`。



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

static inline bool gpio_is_valid(int number)
{
    return false;
}

static inline int gpio_request(unsigned gpio, const char *label)
{
    return -ENOSYS;
}

static inline int gpio_request_one(unsigned gpio,
                                   unsigned long flags, const char *label)
{
    return -ENOSYS;
}

static inline int gpio_request_array(const struct gpio *array, size_t num)
{
    return -ENOSYS;
}

static inline void gpio_free(unsigned gpio)
```

如上图所示，简单介绍一下 `gpio_request` 函数。

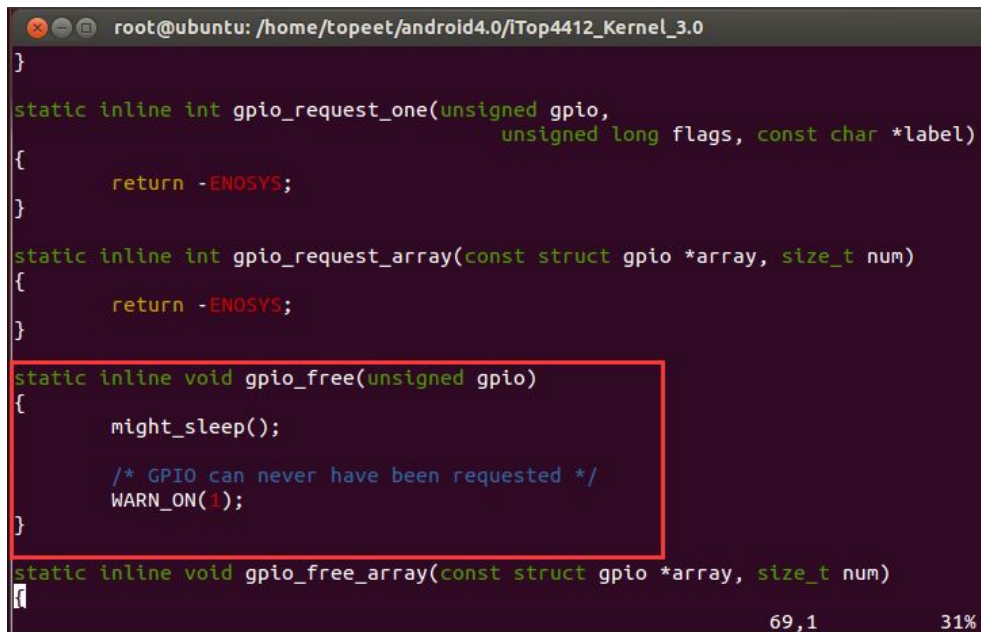
首先这个函数有一个重要的“检测”功能，就是如果其它地方申请了这个 IO，那么这里就会返回错误，提示已经被占用了，这是 Linux 中的一个标准用法。

gpio_request 函数有两个参数

unsigned gpio, 申请的那个 GPIO, 一般是 GPIO 对应的宏定义

const char *label, 为 GPIO 取个名字, 便于阅读

如下图所示, 和 gpio_request 函数对应的是 gpio_free 函数。



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

static inline int gpio_request_one(unsigned gpio,
                                   unsigned long flags, const char *label)
{
    return -ENOSYS;
}

static inline int gpio_request_array(const struct gpio *array, size_t num)
{
    return -ENOSYS;
}

static inline void gpio_free(unsigned gpio)
{
    might_sleep();

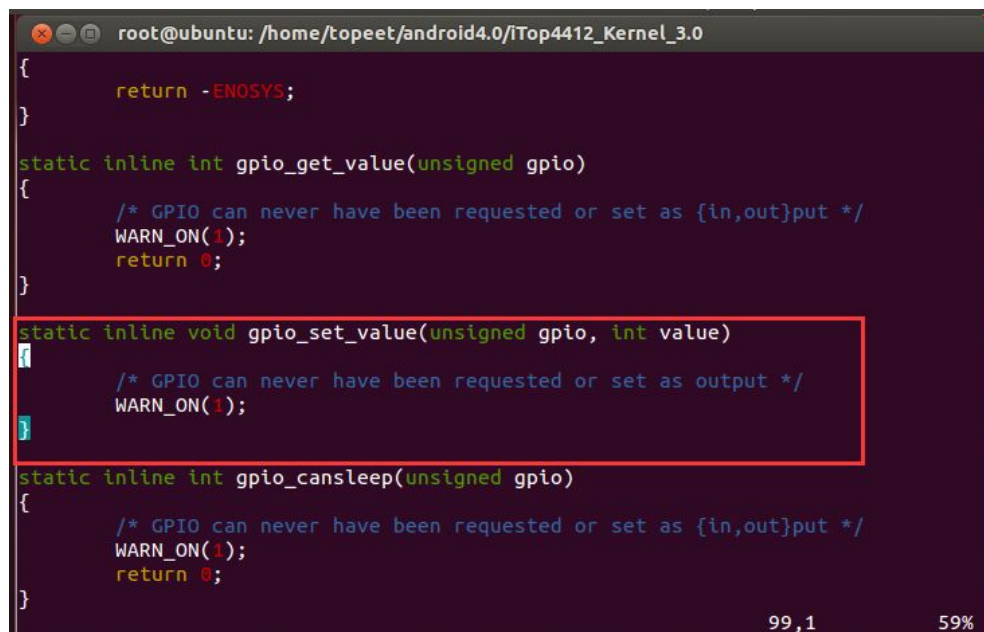
    /* GPIO can never have been requested */
    WARN_ON(1);
}

static inline void gpio_free_array(const struct gpio *array, size_t num)
{
}
```

在调用 gpio_request 函数之后, 向系统表明这个 IO 已经被占用了, 在卸载驱动的时候一般需要调用 gpio_free 函数将其释放。

gpio_free 函数的参数比较简单, 只有一个 GPIO 参数, 使用 GPIO 对应的宏定义即可。

如下图所示, 还有一个赋值函数 gpio_set_value。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
{
    return -ENOSYS;
}

static inline int gpio_get_value(unsigned gpio)
{
    /* GPIO can never have been requested or set as {in,out}put */
    WARN_ON(1);
    return 0;
}

static inline void gpio_set_value(unsigned gpio, int value)
{
    /* GPIO can never have been requested or set as output */
    WARN_ON(1);
}

static inline int gpio_cansleep(unsigned gpio)
{
    /* GPIO can never have been requested or set as {in,out}put */
    WARN_ON(1);
    return 0;
}

99,1 59%
```

在将 GPIO 配置为输出模式之后，还需要给 GPIO 赋值，一般就是高电平和低电平两种。

两个参数分别为

unsigned gpio , GPIO

int value , 高电平 1 和低电平 0。

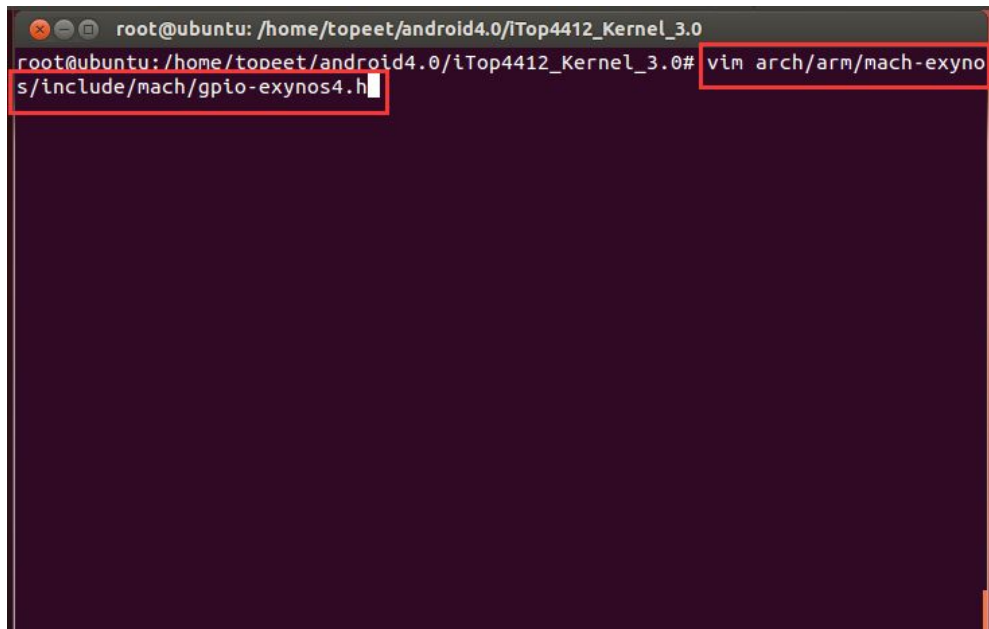
14.4.2 GPIO 配置参数宏定义

GPIO 在 Linux 初始化 进行映射之后调用 GPIO 操作函数对 GPIO 宏定义进行操作就是对 GPIO 的操作。

这个 GPIO 宏定义文件都是由原厂提供，肯定是已经做好的，属于 BSP 板级开发包。

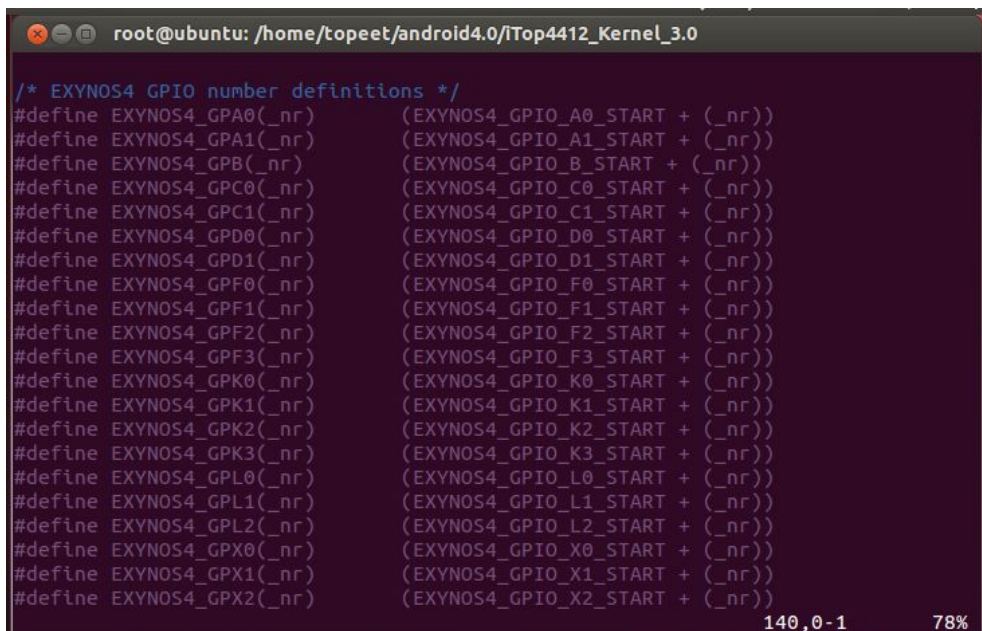
如下图所示，在源码目录中使用命令

“vim arch/arm/mach-exynos/include/mach/gpio-exynos4.h” 打开该文件。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# vim arch/arm/mach-exynos/include/mach/gpio-exynos4.h
```

如下图所示，可以看到所有的 GPIO 都已经定义了。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
/* EXYNOS4 GPIO number definitions */
#define EXYNOS4_GPA0(_nr) (EXYNOS4_GPIO_A0_START + (_nr))
#define EXYNOS4_GPA1(_nr) (EXYNOS4_GPIO_A1_START + (_nr))
#define EXYNOS4_GPB(_nr) (EXYNOS4_GPIO_B_START + (_nr))
#define EXYNOS4_GPC0(_nr) (EXYNOS4_GPIO_C0_START + (_nr))
#define EXYNOS4_GPC1(_nr) (EXYNOS4_GPIO_C1_START + (_nr))
#define EXYNOS4_GPD0(_nr) (EXYNOS4_GPIO_D0_START + (_nr))
#define EXYNOS4_GPD1(_nr) (EXYNOS4_GPIO_D1_START + (_nr))
#define EXYNOS4_GPF0(_nr) (EXYNOS4_GPIO_F0_START + (_nr))
#define EXYNOS4_GPF1(_nr) (EXYNOS4_GPIO_F1_START + (_nr))
#define EXYNOS4_GPF2(_nr) (EXYNOS4_GPIO_F2_START + (_nr))
#define EXYNOS4_GPF3(_nr) (EXYNOS4_GPIO_F3_START + (_nr))
#define EXYNOS4_GPK0(_nr) (EXYNOS4_GPIO_K0_START + (_nr))
#define EXYNOS4_GPK1(_nr) (EXYNOS4_GPIO_K1_START + (_nr))
#define EXYNOS4_GPK2(_nr) (EXYNOS4_GPIO_K2_START + (_nr))
#define EXYNOS4_GPK3(_nr) (EXYNOS4_GPIO_K3_START + (_nr))
#define EXYNOS4_GPL0(_nr) (EXYNOS4_GPIO_L0_START + (_nr))
#define EXYNOS4_GPL1(_nr) (EXYNOS4_GPIO_L1_START + (_nr))
#define EXYNOS4_GPL2(_nr) (EXYNOS4_GPIO_L2_START + (_nr))
#define EXYNOS4_GPX0(_nr) (EXYNOS4_GPIO_X0_START + (_nr))
#define EXYNOS4_GPX1(_nr) (EXYNOS4_GPIO_X1_START + (_nr))
#define EXYNOS4_GPX2(_nr) (EXYNOS4_GPIO_X2_START + (_nr))
140,0-1 78%
```

在原理图中查找 KP_COL0、VDD50_EN 网络，最终连接到 4412 上的部分如下图所示。



如上图所示，则两个 Led 的宏定义为 EXYNOS4_GPL2(0)，EXYNOS4_GPK1(1)。

14.4.3 GPIO 配置函数和参数

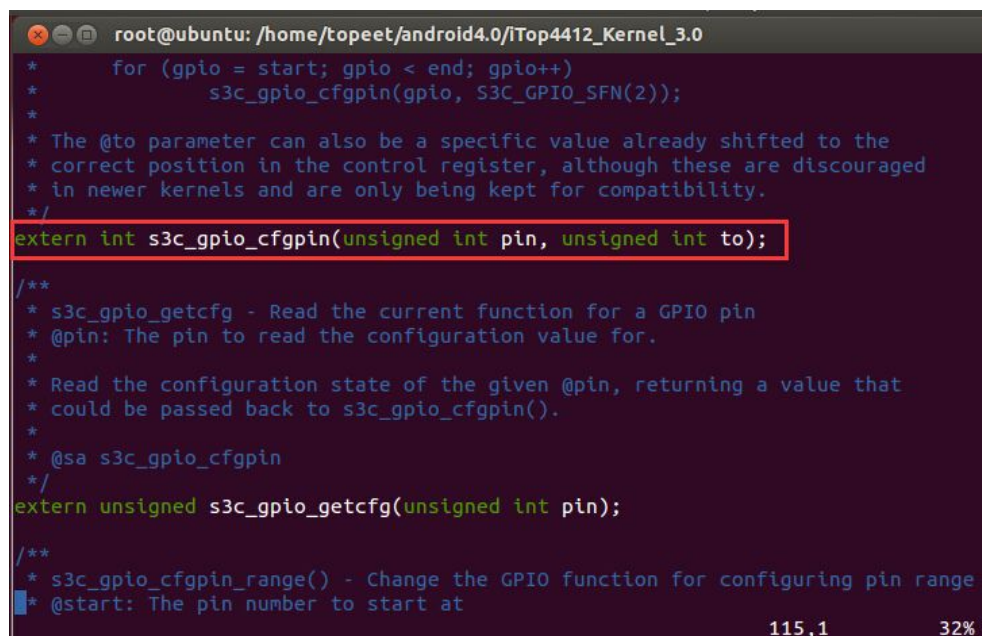
在 Linux 中，对 GPIO 的配置函数以及参数都已经集成到三星板级开发包中。

先来看一下配置函数，如下图所示，在源码目录中使用命令

“vim arch/arm/plat-samsung/include/plat/gpio-cfg.h” 打开该文件。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# vim arch/arm/plat-samsung/include/plat/gpio-cfg.h
```

如下图所示，s3c_gpio_cfgpin 函数就是本节实验需要的。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
*   for (gpio = start; gpio < end; gpio++)
*       s3c_gpio_cfgpin(gpio, S3C_GPIO_SFN(2));
*
* The @to parameter can also be a specific value already shifted to the
* correct position in the control register, although these are discouraged
* in newer kernels and are only being kept for compatibility.
*/
extern int s3c_gpio_cfgpin(unsigned int pin, unsigned int to);

/**
 * s3c_gpio_getcfg - Read the current function for a GPIO pin
 * @pin: The pin to read the configuration value for.
 *
 * Read the configuration state of the given @pin, returning a value that
 * could be passed back to s3c_gpio_cfgpin().
 *
 * @sa s3c_gpio_cfgpin
 */
extern unsigned s3c_gpio_getcfg(unsigned int pin);

/**
 * s3c_gpio_cfgpin_range() - Change the GPIO function for configuring pin range
 * @start: The pin number to start at

```

如上图所示，函数 `extern int s3c_gpio_cfgpin(unsigned int pin, unsigned int to);`;

一般来说带有 `s3cxxx` 的函数就是三星平台能够通用的函数。

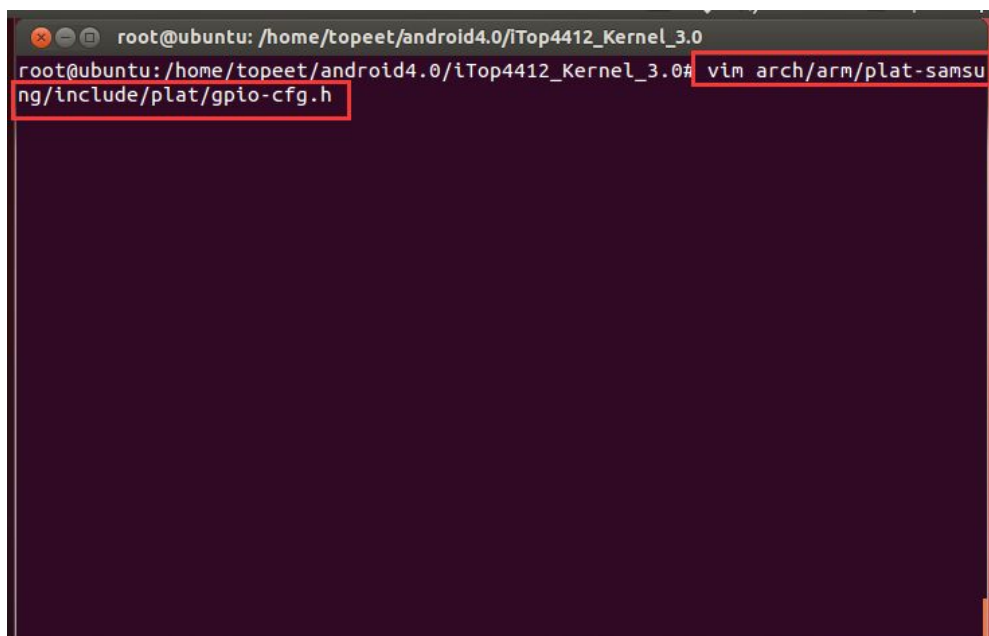
`s3c_gpio_cfgpin` 管脚配置函数有两个参数

参数 `unsigned int pin`，管脚

参数 `unsigned int to`，配置参数。

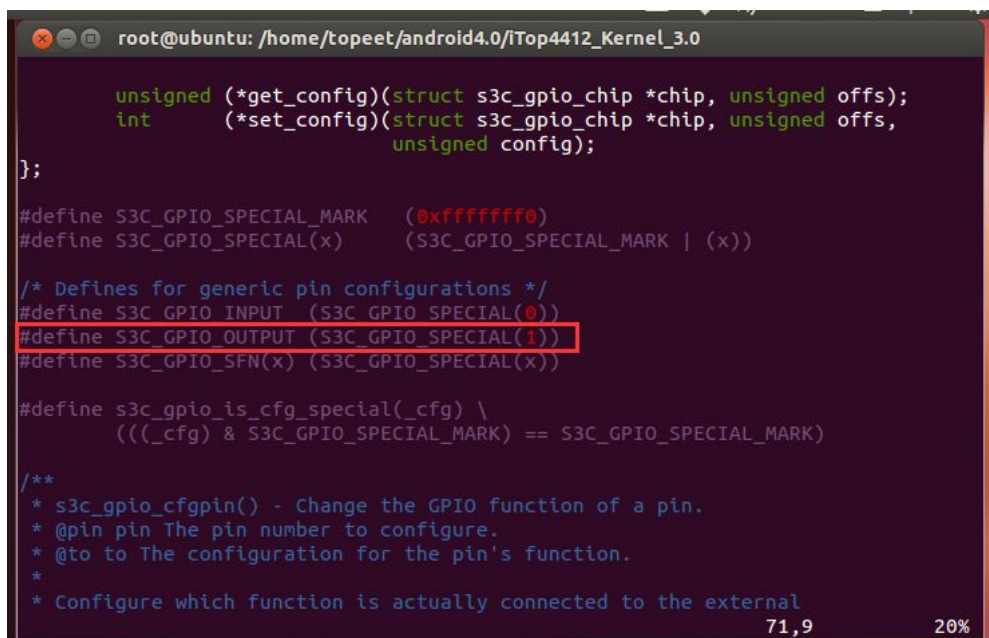
再来看一下配置参数，如下图所示，在源码目录中使用命令

“`vim arch/arm/plat-samsung/include/plat/gpio-cfg.h`” 打开该文件，配置参数和函数是在同一个函数中。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# vim arch/arm/plat-samsu
ng/include/plat/gpio-cfg.h
```

如下图所示，对于 GPIO 需要将其配置为输出模式，对应 S3C_GPIO_OUTPUT 宏定义。



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

    unsigned (*get_config)(struct s3c_gpio_chip *chip, unsigned offs);
    int      (*set_config)(struct s3c_gpio_chip *chip, unsigned offs,
                          unsigned config);
};

#define S3C_GPIO_SPECIAL_MARK    (0xffffffff)
#define S3C_GPIO_SPECIAL(x)     (S3C_GPIO_SPECIAL_MARK | (x))

/* Defines for generic pin configurations */
#define S3C_GPIO_INPUT    (S3C_GPIO_SPECIAL(0))
#define S3C_GPIO_OUTPUT  (S3C_GPIO_SPECIAL(1))
#define S3C_GPIO_SFN(x)  (S3C_GPIO_SPECIAL(x))

#define s3c_gpio_is_cfg_special(_cfg) \
    (((_cfg) & S3C_GPIO_SPECIAL_MARK) == S3C_GPIO_SPECIAL_MARK)

/**
 * s3c_gpio_cfgpin() - Change the GPIO function of a pin.
 * @pin pin The pin number to configure.
 * @to to The configuration for the pin's function.
 *
 * Configure which function is actually connected to the external
71,9      20%
```

14.5 编写简单应用调用 LED 管脚，并测试

在前面的 devicenode_linux_module.c 文件上添加代码，首先将文件名 devicenode_linux_module.c 改为 leds.c。

先处理一下编译文件 Makefile，如下图所示，将 devicenode_linux_module 改为 leds。

```
#!/bin/bash
#通知编译器我们要编译模块的哪些源码
#这里是编译itop4412_hello.c这个文件编译成中间文件itop4412_hello.o
obj-m += leds.o

#源码目录变量，这里用户需要根据实际情况选择路径
#作者是将Linux的源码拷贝到目录/home/topeet/android4.0下并解压的
KDIR := /home/topeet/android4.0/iTop4412_Kernel_3.0

#当前目录变量
PWD ?= $(shell pwd)

#make命名默认寻找第一个目标
#make -C就是指调用执行的路径
#$(KDIR) Linux源码目录，作者这里指的是/home/topeet/android4.0/iTop4412_Kernel_3.0
#$(PWD) 当前目录变量
#modules要执行的操作
all:
    make -C $(KDIR) M=$(PWD) modules

#make clean执行的操作是删除后缀为o的文件
clean:
    rm -rf *.o
```

接着修改 leds.c 文件。

首先添加需要的头文件，如下图所示，分别是申请 GPIO、配置函数、配置参数、GPIO 宏定义等的头文件。然后将设备节点名称由 hello_ctl123 修改为 hello_ctl

```
#include <linux/init.h>
#include <linux/module.h>

/*驱动注册的头文件，包含驱动的结构体和注册和卸载的函数*/
#include <linux/platform_device.h>
/*注册杂项设备头文件*/
#include <linux/miscdevice.h>
/*注册设备节点的文件结构体*/
#include <linux/fs.h>

/*Linux中申请GPIO的头文件*/
#include <linux/gpio.h>
/*三星平台的GPIO配置函数头文件*/
/*三星平台EXYNOS系列平台，GPIO配置参数宏定义头文件*/
#include <plat/gpio-cfg.h>
#include <mach/gpio.h>
/*三星平台4412平台，GPIO宏定义头文件*/
#include <mach/gpio-exynos4.h>

#define DRIVER_NAME "hello_ctl"
#define DEVICE_NAME "hello_ctl"
```

然后需要修改的就是 probe 函数，一般说来 GPIO 的初始化都是在 probe 中。如下图所示，调用配置函数以及配置函数。

```
static int hello_probe(struct platform_device *pdv){
    int ret;

    printk(KERN_EMERG "\tinitialized\n");

    ret = gpio_request(EXYNOS4_GPL2(0), "LEDS");
    if(ret < 0){
        printk(KERN_EMERG "gpio_request EXYNOS4_GPL2(0) failed!\n");
        return ret;
    }

    s3c_gpio_cfgpin(EXYNOS4_GPL2(0), S3C_GPIO_OUTPUT);

    gpio_set_value(EXYNOS4_GPL2(0), 0);

    misc_register(&hello_dev);

    return 0;
}
```

然后就是修改一下 ioctl 函数，在 Linux 中对 GPIO 的控制一般是使用 ioctl，虽然 write 函数也可以实现类似的功能，但是 ioctl 函数的效率高一些。如下图所示，根据应用传入的参数给 GPIO 赋值。

```
static long hello_ioctl( struct file *files, unsigned int cmd, unsigned long arg){
    printk("cmd is %d,arg is %d\n",cmd,arg);

    if(cmd > 1){
        printk(KERN_EMERG "cmd is 0 or 1\n");
    }
    if(arg > 1){
        printk(KERN_EMERG "arg is only 1\n");
    }

    gpio_set_value(EXYNOS4_GPL2(0),cmd);

    return 0;
}
```

如上图所示，先对于参数做一个简单的判断，然后给 led 赋值。

接着再来看一下应用，如下图所示，应用比较简单，调用延时函数，首先将 Led 点亮三秒，然后再灭掉三秒，再点亮。

```
#include <stdio.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>

main() {
    int fd;
    char *hello_node = "/dev/hello_ctl";

    /*O_RDONLY只读打开,O_NDELAY非阻塞方式*/
    if((fd = open(hello_node,O_RDONLY|O_NDELAY))<0) {
        printf("APP open %s failed",hello_node);
    }
    else{
        printf("APP open %s success",hello_node);
        ioctl(fd,1,1);
        sleep(3);
        ioctl(fd,0,1);
        sleep(3);
        ioctl(fd,1,1);

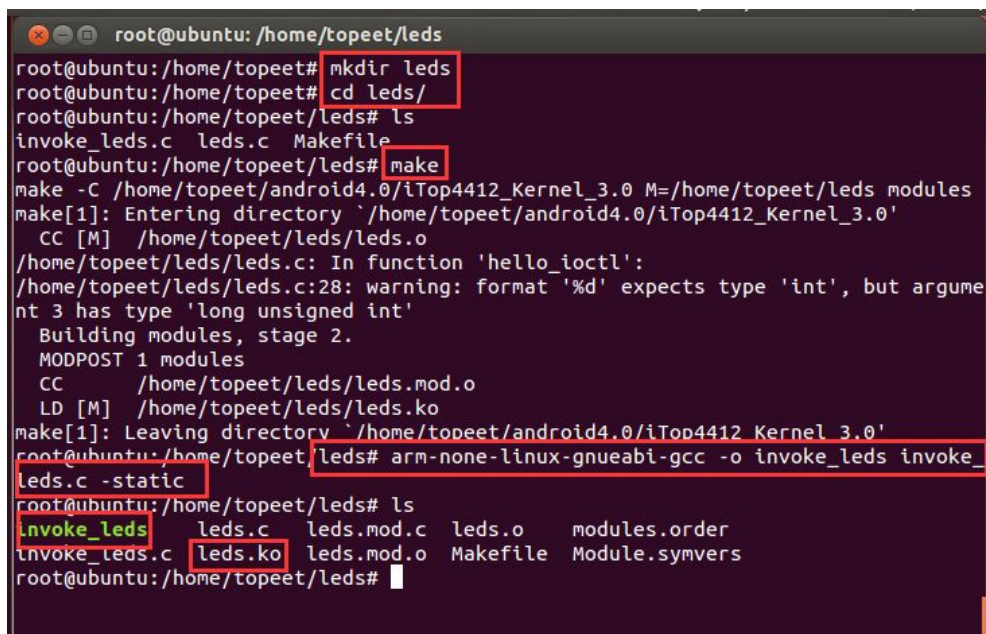
    }

    close(fd);
}
```


在 Ubuntu 系统下新建 leds 文件夹，将写好的 leds 和编译脚本拷贝到 leds 文件夹下，使用 Makefile 命令编译驱动，使用

“arm-none-linux-gnueabi-gcc -o invoke_leds invoke_leds.c -static” 命令编译应用。

如下图所示。



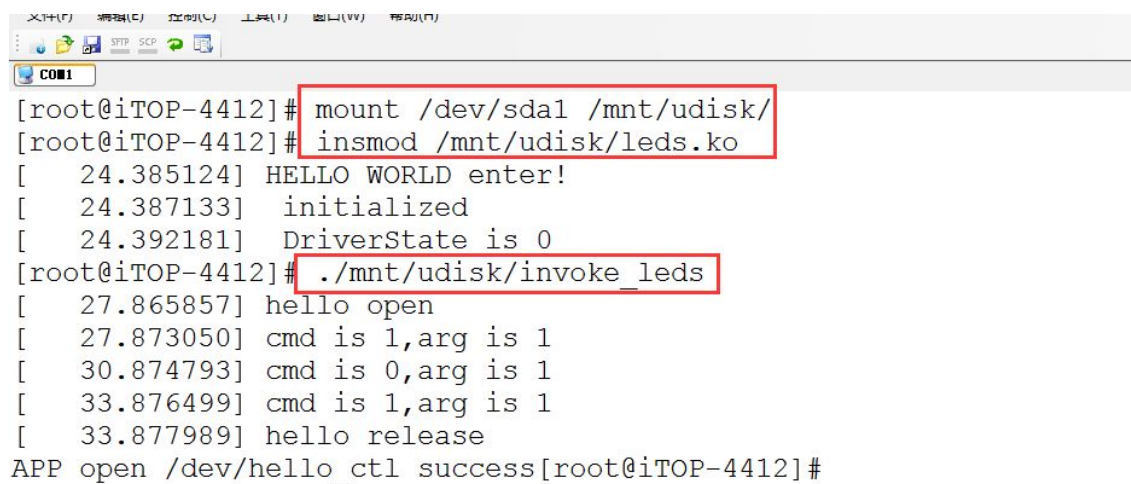
```
root@ubuntu: /home/topeet/leds
root@ubuntu:/home/topeet# mkdir leds
root@ubuntu:/home/topeet# cd leds/
root@ubuntu:/home/topeet/leds# ls
invoke_leds.c  leds.c  Makefile
root@ubuntu:/home/topeet/leds# make
make -C /home/topeet/android4.0/iTop4412_Kernel_3.0 M=/home/topeet/leds modules
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
  CC [M] /home/topeet/leds/leds.o
/home/topeet/leds/leds.c: In function 'hello_ioctl':
/home/topeet/leds/leds.c:28: warning: format '%d' expects type 'int', but argument 3 has type 'long unsigned int'
  Building modules, stage 2.
  MODPOST 1 modules
  CC /home/topeet/leds/leds.mod.o
  LD [M] /home/topeet/leds/leds.ko
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
root@ubuntu:/home/topeet/leds# arm-none-linux-gnueabi-gcc -o invoke_leds invoke_leds.c -static
root@ubuntu:/home/topeet/leds# ls
invoke_leds  leds.c  leds.mod.c  leds.o  modules.order
invoke_leds.c  leds.ko  leds.mod.o  Makefile  Module.symvers
root@ubuntu:/home/topeet/leds#
```

将上图中的文件 invoke_leds 和 leds.ko 拷贝到 U 盘。

启动开发板，将 U 盘插入开发板，使用命令 “mount /dev/sda1 /mnt/udisk/” 加载 U 盘符，

使用命令 “insmod /mnt/udisk/leds.ko ” 加载驱动 leds.ko ，

使用命令 “./mnt/udisk/invoke_leds” 运行小应用 invoke_leds，如下图所示。



```
文件(F) 编辑(E) 控制(C) 上传(U) 窗口(W) 帮助(H)
COM1
[root@iTOP-4412]# mount /dev/sda1 /mnt/udisk/
[root@iTOP-4412]# insmod /mnt/udisk/leds.ko
[ 24.385124] HELLO WORLD enter!
[ 24.387133] initialized
[ 24.392181] DriverState is 0
[root@iTOP-4412]# ./mnt/udisk/invoke_leds
[ 27.865857] hello open
[ 27.873050] cmd is 1,arg is 1
[ 30.874793] cmd is 0,arg is 1
[ 33.876499] cmd is 1,arg is 1
[ 33.877989] hello release
APP open /dev/hello_ctl success[root@iTOP-4412]#
```

经过上面的操作可观察到 led 小灯会一亮一灭一亮，中间大概间隔三秒钟。