Linux下控制GPIO的三种方法

原创 四季帆 于 2022-05-16 20:00:00 发布 13237 收藏 122 版权

分类专栏： # gpio 文章标签： gpio 嵌入式 驱动开发 linux

1. 应用空间控制 gpio

1.1简介

在/sys/class/gpio/下有个export文件，向export文件写入要操作的GPIO号，使得该GPIO的操作接口从内核空间暴露到用户空间，GPIO的操作接口包括direction和value等，direction控制GPIO输入或者输出模式，而value可控制GPIO的状态或者读取状态。

```
/sys/class/gpio/目录下各个文件说明：
/sys/class/gpio/export文件用于通知系统需要导出控制的GPIO引脚编号；
/sys/class/gpio/unexport 用于通知系统取消导出；
/sys/class/gpio/gpioX/direction文件，可以写入in（设置输入方向）或out（设置输出方向）；
/sys/class/gpio/gpioX/value文件是可以读写GPIO状态；
/sys/class/gpio/gpiochipX目录保存系统中GPIO寄存器的信息，包括每个寄存器控制引脚的起始编号，寄存器名称，引脚总数；其中X表示具体的引脚编号。
```

1.2操作gpio

比如我要操作GPIO8_A6作为高电平输出有效, 那么有以下三个操作：

1. 2.1 换算对应的gpio number

可以通过/sys/kernel/debug/gpio查询信息：

```
root@rk3288:/sys/kernel/debug # cat gpio
//snip
GPIOs 184-215, platform/ff770000.pinctrl, gpio6:
 gpio-193 (?                   ) in  hi
 gpio-194 (?                   ) in  hi

GPIOs 216-247, platform/ff770000.pinctrl, gpio7:
 gpio-218 (enable              ) out hi
 gpio-219 (lcd_en              ) in  hi
 gpio-220 (lcd_cs              ) in  hi
 gpio-221 (gslX680 wake pin    ) out hi
 gpio-222 (gslX680 irq pin     ) out lo
 gpio-223 (headset_gpio        ) in  hi
 gpio-233 (?                   ) in  hi
 gpio-234 (?                   ) in  hi
 GPIOs 248-279, platform/ff770000.pinctrl, gpio8:

 GPIOs 280-311, platform/ff770000.pinctrl, gpio15:
```

可以看到gpio8是以nubmer为248开始, 那么GPIO8_A6就是 248 + 6 = 254，接下来就可以导出gpio了。

```
root@rk3288:/sys/class/gpio # echo 254 > export
root@rk3288:/sys/class/gpio # ls
export
gpio254
```

1.2.2 设置成输出

```
root@rk3288:/sys/class/gpio/gpio254 # echo out > direction
root@rk3288:/sys/class/gpio/gpio254 # cat direction
out
```

1.2.3 输出高电平

```
root@rk3288:/sys/class/gpio/gpio254 # echo 1 > value
root@rk3288:/sys/class/gpio/gpio254 # cat value
```

1.3 总结

这种方式一般不采用，为了gpio使用的安全性，一般是不将gpio的使用权暴露给应用层的，即sys/class/下没有gpio节点。

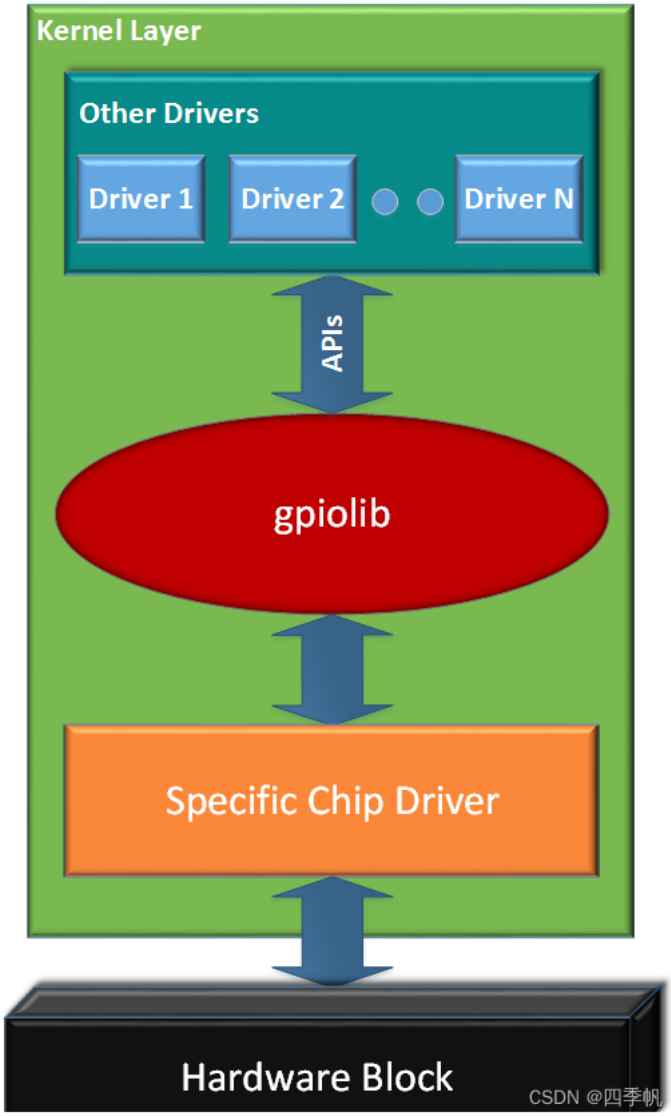
2. 内核空间控制gpio

在内核空间控制gpio有两种方法，第一种是通过调用gpiolib的接口来控制gpio；第二种是通过 ioremap 来控制gpio。

2.1 gpiolib控制gpio

2.1.1 gpiolib简介

Linux Kernel 中对 GPIO 资源进行了抽象，抽象出一个叫做 Gpiolib 的东西。



中间层是 Gpiolib，用于管理系统中的 GPIO。Gpiolib 汇总了 GPIO 的通用操作，根据 GPIO 的特性，Gpiolib 对上（其他 Drivers）提供的一套统一通用的操作 GPIO 的软件接口，屏蔽了不同芯片的具体实现。对下，Gpiolib 提供了针对不同芯片操作的一套 framework，针对不同芯片，只需要实现 Specific Chip Driver ，然后使用 Gpiolib 提供的注册函数，将其挂接到 Gpiolib 上，这样就完成了这一套东西。

2.1.2 Gpiolib 为其他驱动提供的 APIs

```
int gpio_request(unsigned gpio, const char *label);
/*向内核申请 gpio，要使用 GPIO 首先应该向内核进行申请，返回 0，代表申请成功，
 *可以进行后续操作*/

void gpio_free(unsigned gpio);
/*对应 gpio_request，是使用完gpio以后把gpio释放掉*/

int gpio_direction_input(unsigned gpio);
/*设置 GPIO 为输入*/

int gpio_direction_output(unsigned gpio, int value);
/*设置 GPIO 为输出*/

int gpio_get_value(unsigned gpio);
/*读取 GPIO 的值*/

int gpio_set_value(unsigned gpio);
/*设置 GPIO 的值*/
```

2.1.3 操作gpio

功能和1.2一样。

```
#define GPIO8_A6 254
ret = gpio_request(GPIO8_A6 , "gpio8_a6");
if (!ret) {
    printk("request for gpio8_a6 failed:%d\n", ret);
        return 0;
}
gpio_direction_output(GPIO8_A6 ,1);//设置GPIO8_A6为输出功能且输出高电平
```

2.2 ioremap控制gpio

这种方法会降低程序的可读性，不建议使用。

linux内核 空间访问的地址为虚拟地址（3~4GB）,故在内核空间操作某个寄存器时，需先通过ioremap函数将物理地址映射成虚拟地址。

用ioremap() 获取寄存器的地址：

```
unsigned int __iomem  *base_addr1;  //__iomem可选择，告诉你为虚拟地址
#define  GPIO8_REGBASE  (0x20A0000)
#define  GPIO8_A6 (*(volatile unsigned int *)(base_addr1 + 6)) //指针unsigned int为4字节,指针加1，字节加4
base_addr1 = ioremap(GPIO8_REGBASE, 0x14)
```

通过 readl() 或者 writel() 函数直接操作映射后的地址：

```
GPIO8_A6 |= (1<<8);

int temp;
temp = readl(GPIO8_A6);
temp |= (1<<8);
writel(temp, GPIO8_A6);
```

使用完后，取消映射：

```
iounmap(base_addr1);
```

3. 查看GPIO全部信息

```
cat /sys/kernel/debug/pinctrl/pinctrl/pinmux-pins
```

```
    Pinmux settings per pin
Format: pin (name): mux_owner gpio_owner hog?
pin 0 (gpio0-0): wireless-wlan (GPIO UNCLAIMED) function wireless-wlan group wifi-wake-host
pin 1 (gpio0-1): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 2 (gpio0-2): (MUX UNCLAIMED) gpio0:2
pin 3 (gpio0-3): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 4 (gpio0-4): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 5 (gpio0-5): (MUX UNCLAIMED) gpio0:5
pin 6 (gpio0-6): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 7 (gpio0-7): (MUX UNCLAIMED) gpio0:7
pin 8 (gpio0-8): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 9 (gpio0-9): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 10 (gpio0-10): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 11 (gpio0-11): ff050000.i2c (GPIO UNCLAIMED) function i2c1 group i2c1-xfer
pin 12 (gpio0-12): ff050000.i2c (GPIO UNCLAIMED) function i2c1 group i2c1-xfer
pin 13 (gpio0-13): (MUX UNCLAIMED) (GPIO UNCLAIMED)
```

根据对比实验，“MUX UNCLAIMED” 的意思好像是该复用引脚未被配置，仅个人小实验，不具备绝对准确性。