



SLASH_24

码龄5年 暂无认证

13

47万+

67万+

1万+



原创

周排名

总排名

访问

等级

260

5

3

0

18

积分

粉丝

获赞

评论

收藏

私信

关注

搜博主文章

Q

- 热门文章
- 在Linux终端输出带颜色的文字的方法

4398
- 常用的GPIO的标准接口函数

4040
- ARM体系结构（一）

901
- 信号量与自旋锁的区别

575
- Linux内核模块

490

您愿意向朋友推荐“博客详情页”吗？











强烈不推荐

不推荐

一般般

推荐


强烈推荐

- 最新文章
- 信号量与自旋锁的区别
- TCP和UDP的区别
- Linux内核定时器
- 2017年 13篇

常用的GPIO的标准接口函数

原创 SLASH_24 2017-02-09 12:58:45 4043 收藏 12 版权

分类专栏：Linux驱动 文章标签：内核模块 linux驱动 GPIO 内核 LED驱动

Linux驱动 专栏收录该内容

0 订阅 6 篇文章 订阅专栏

一、GPIO的标准接口函数

为了使得GPIO具有更好的可移植性，在Linux内核中，有一些基本的模块可以使用标准的接口函数来操作，如：GPIO、INT、Timer、Clock等基本模块。使用该GPIO口必须先对其进行申请(GPIO也是一个资源，一个GPIO只能申请一次，再次申请会报错。)，在内核源码中，每一个GPIO口都对应一个唯一的"ID"，根据硬件平台的不同，可在内核源码目录下的arch/对应的硬件平台找到相应的定义。如S5PV210：arch/arm/machs5pv210/include/mach/gpio.h。然后需设置GPIO口的功能，如基本的输入输出、中断等，接着，根据具体的情况，控制GPIO口的高低电平，实现对GPIO的控制。

二、常用的GPIO的标准接口函数

1. GPIO的标准接口函数包含头文件

```
#include <linux/gpio.h>
```

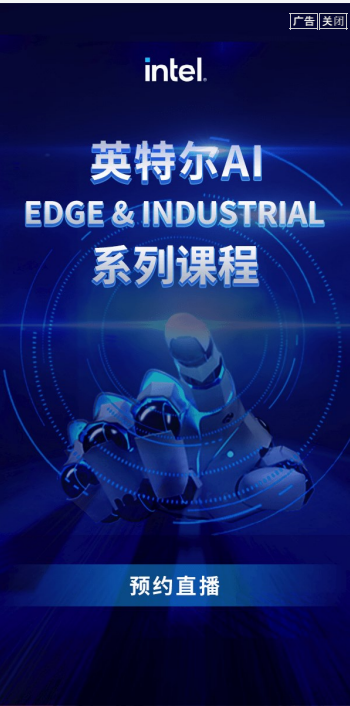
2. GPIO的申请与释放函数

GPIO申请函数：
函数原型：int gpio_request(unsigned gpio, const char *label);
参数说明：
unsigned gpio：GPIO对应的"ID"
const char *label：自定义GPIO口的名字
返回值：
成功返回0，失败返回一个负的错误码。

GPIO释放函数：
函数原型：void gpio_free(unsigned gpio);
参数说明：
unsigned gpio：GPIO对应的"ID"
返回值：
无返回值。

3. 设置GPIO的输入输出功能函数

设置GPIO的输入功能：
函数原型：int gpio_direction_input(unsigned gpio);
参数说明：
unsigned gpio：GPIO对应的"ID"
返回值：



- 分类专栏
- 

ARM体系结构与编程
- 

算法

1篇
- 

数据结构
- 

C语言

2篇
- 

Linux

1篇
- 

Linux驱动

6篇
- 

网络编程

1篇

成功返回0，失败返回一个负的错误码。

设置GPIO的输出功能：

函数原型：int gpio_direction_output(unsigned gpio, int value);

参数说明：

unsigned gpio：GPIO对应的"ID"

int value：初始电平，0为低电平，1为高电平

返回值：

成功返回0，失败返回一个负的错误码。

4. 获取GPIO的输入值和设置GPIO的输出值

获取GPIO的输入值函数：

函数原型：int gpio_get_value(unsigned gpio);

参数说明：

unsigned gpio：GPIO对应的"ID"

返回值：

成功返回0，失败返回一个负的错误码。

设置GPIO的输出值函数：

函数原型：void gpio_set_value(unsigned gpio, int value);

参数说明：

unsigned gpio：GPIO对应的"ID"

int value：设置GPIO口的高低电平

返回值：

无返回值

三、GPIO的标准接口函数实例

驱动程序(led_dev.c)：

```
1  #include <linux/kernel.h>
2  #include <linux/module.h>
3  #include <linux/uaccess.h>
4  #include <linux/fs.h>
5  #include <linux/cdev.h>
6  #include <linux/ioport.h>
7  #include <asm/io.h>
8  #include <linux/device.h>
9  #include <linux/gpio.h>    //GPIO标准接口函数
10
11 #define LED_GPIO(x) S5PV210_GPJ2(x)
12
13 static dev_t led_dev_num;
14 static unsigned int led_dev_major = 120;
15 static unsigned int led_dev_minor = 0;
16
17 static struct class *led_dev_class;
18 static struct device *led_dev_device;
19
20 //1. 定义一个字符设备的cdev
21 static struct cdev led_cdev;
22
```

```

23 static ssize_t led_dev_write(struct file *flip, const char __user *buf, size_t size, loff_t *ops)
24 {
25     int ret;
26     char kbuf[1];
27
28     if(size != 2)
29         return -EINVAL;
30
31     ret = copy_from_user(kbuf, buf, size);
32     if(ret != 0)
33         return -EFAULT;
34
35     if(kbuf[0]!=0 && kbuf[0]!=1){
36         printk("cmd error\n");
37         return -EINVAL;
38     }
39
40     if(kbuf[1]<0 || kbuf[1]>4){
41         printk("args error\n");
42         return -EINVAL;
43     }
44
45     if(kbuf[0] == 0)    //LED ON
46         gpio_set_value(LED_GPIO(kbuf[1]-1), 0);
47     else    //LED OFF
48         gpio_set_value(LED_GPIO(kbuf[1]-1), 1);
49
50     return size;
51 }
52
53 //2. 定义Led_dev的文件操作集并初始化
54 static struct file_operations led_fops = {
55     .owner = THIS_MODULE,
56     //.open = led_dev_open,
57     .write = led_dev_write,
58     //.release = led_dev_release,
59 };
60
61 static int __init led_dev_init(void)
62 {
63     int i;
64     int ret;
65
66     //3. 申请设备号(这里用静态申请)
67     led_dev_num = MKDEV(led_dev_major, led_dev_minor);
68
69     ret = register_chrdev_region(led_dev_num, 1, "led_dev");
70     if(ret < 0){
71         printk("register led chrdev failed\n");
72         return ret;
73     }
74
75     //4. 初始化Led_dev
76     cdev_init(&led_cdev, &led_fops);
77
78     //5. 将led_dev加入内核
79     ret = cdev_add(&led_cdev, led_dev_num, 1);
80     if(ret < 0){
81         printk("led_dev add to kernel failed\n");
82         goto cdev_add_error;
83     }

```

```

84
85 //6. 创建Led_dev的class
86 led_dev_class = class_create(THIS_MODULE, "led_class");
87 if(led_dev_class == NULL){
88     printk("led_dev's class create failed\n");
89     ret = -EBUSY;
90     goto class_create_error;
91 }
92
93 //7. 创建Led_dev的device
94 led_dev_device = device_create(led_dev_class, NULL, led_dev_num, NULL, "led_drv");
95 if(led_dev_device == NULL){
96     printk("led_dev's device create failed\n");
97     ret = -EBUSY;
98     goto class_device_error;
99 }
100
101 //8. 申请GPIO口并初始化
102 for(i=0; i<4; i++){
103     ret = gpio_request(LED_GPIO(i), "led_gpio");
104     if(ret < 0){
105         printk("gpio request failed\n");
106         goto gpio_request_error;
107     }
108
109     gpio_direction_output(LED_GPIO(i), 1);
110 }
111
112 printk("led_dev init\n");
113
114 return 0;
115
116 gpio_request_error:
117     device_destroy(led_dev_class, led_dev_num);
118 class_device_error:
119     class_destroy(led_dev_class);
120 class_create_error:
121     cdev_del(&led_cdev);
122 cdev_add_error:
123     unregister_chrdev_region(led_dev_num, 1);
124
125
126     return ret;
127 }
128
129 static void __exit led_dev_exit(void)
130 {
131     int i;
132     unregister_chrdev_region(led_dev_num, 1);
133     cdev_del(&led_cdev);
134     device_destroy(led_dev_class, led_dev_num);
135     class_destroy(led_dev_class);
136     for(i=0; i<4; i++){
137         gpio_free(LED_GPIO(i));
138     }
139
140
141     printk("led_dev exit\n");
142 }
143
144 module_init(led_dev_init);

```

```
145 module_exit(led_dev_exit);
146 MODULE_LICENSE("GPL");
```

编译文件(Makefile) :

```
1 obj-m += led_dev.o
2 KERN_DIR=/home/gec/linux-2.6.35.7-gec-v3.0-gt110
3
4 modules:
5 $(MAKE) -C $(KERN_DIR) M=$(PWD) modules
6 clean:
7 $(MAKE) -C $(KERN_DIR) M=$(PWD) modules clean
```

测试文件(test.c) :

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4
5 int main(void)
6 {
7     int fd;
8     char buf[2];
9     fd = open("/dev/led_drv", O_WRONLY);
10    if(fd < 0){
11        perror("open led_drv");
12        return -1;
13    }
14
15    while(1){
16        buf[0]=1; buf[1]=2;//D2 亮
17        write(fd,buf,2);
18        sleep(1);
19        buf[0]=0; buf[1]=2;//D2 灭
20        write(fd,buf,2);
21        sleep(1);
22    }
23
24    close(fd);
25
26    return 0;
27 }
```

linux内核里的GPIO操作函数

肖恒的专栏 4005

1.GPIO_set_value(unsigned gpio, int value)用来设置gpio寄存器的值 2.GPIO_direction_output(unsigned gpio, int value)用来设置gpio为输出功能,同时设...

STM32总结之GPIO 常用库函数

peijian1998的专栏 4085

配置相关函数 1.void GPIO_Init (GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct) 函数解释: GPIO的初始化函数,该函数的作用是对io进行...



请发表有价值的评论，博客评论不欢迎灌水，良好的社区氛围需大家一起维护。

抢沙发



评论

gpio_direction_output 和 gpio_set_value之间的使用关系 在linux驱动中常常会碰到gpio_set_value (port_num,0/1) 或gpio_direction_output (port_num,0/1) 这两者有什么关系呢gpio_set_value (port_num,0/1...	BlackSmith 1万+
GPIO常用操作函数 最新发布 (1) void GPIO_Delnit(GPIO_TypeDef* GPIOx) 功能：将GPIOx外设寄存器初始化为默认值 注释：判断GPIOx具体为哪个端口，再通过操作RCC_APB2R...	423
二.GPIO输入输出及延迟函数 一. GPIO 八种模式，四种输入，四种输出 1.GPIO输入 初始化 void GPIO_Config(void){ GPIO_InitTypeDef GPIO_INT; RCC_APB2PeriphClockCmd(RCC...	m0_50217870的博客 409
gpio_direction_output 与 gpio_set_value gpio_set_value (port_num,0/1) 一般只是在这个GPIO口的寄存器上写上某个值，至于这个端口是否设置为输出，它就管不了！而gpio_direction_o...	程序员笔记 1058
基于树莓派的python GPIO编程-常用函数综合整理 Hello，朋友们大家好，欢迎大家来到LJUSE网络。 随着Mini型PC越来越多，与之而来的创客也丰富起来，比如说Arduino就是一个非常好里例子。 不过，...	Python达人 1万+
GPIO通用输入输出 GPIO通用输入输出 一、GPIO的功能概述 用途：GPIO是通用输入输出（ General Purpose I/O ）的简称，主要用于工业现场需要用到数字量输入/输出的场...	dq_zhanghaifang的博客 1461
GPIO库函数介绍 文章目录1.一个初始化函数1.1技巧使用方法1.2代码1.3宏定义1.3.1 assert_param(IS_GPIO_ALL_PERIPH(GPIOx));1.3.2 assert_param(IS_GPIO_PIN(G...	流痕的博客 452
linux内核中GPIO的使用（二）-标准接口函数 在linux内核中，有一些基本模块可以使用标准的接口函数来操作，比如GPIO、interrupt、clock，所谓的标准接口函数是指一些与硬件平台无关的、linux下...	Huang-Fly的博客 9025
gpio1节点：GPIO子系统常用函数 gpio1节点：GPIO子系统初窥 GPIO1控制器的寄存器基地址 类似的还有gpio2~5 gpio1节点 imx6ull.dtsi gpio1: gpio@209c000 { compatible = "fsl,imx6ul-g...	十七阿哥的博客 201
gpio函数 GPIO_Delnit 重新初始化外围设备GPIOx相关寄存器到它的默认复位值 GPIO_AFIODelnit 初始化交错功能(remap, event control和 EXTI 配置) 寄存器 GPI...	dyz402026753的专栏 698
Linux GPIO 驱动（gpio lib） 目录 1、简述 2、Gpiolib 相关数据结构分析 2.1 gpio_chip 结构 2.2 gpio_desc 结构 2.3 gpio_device 结构 3、Gpiolib 对接芯片底层 3.1、注册 GPIO 资源...	StephenZhou 1万+
GPIO标准函数 这里写目录标题一、GPIO的标准接口函数二、GPIO标准接口函数三、参考源码实例四、GPIO口号五、检查PIN是否被其他外设使用一、GPIO的标准接...	weixin_42832472的博客 311
Linux内核中GPIO操作函数的使用方法 http://blog.sina.com.cn/s/blog_a6559d9201015vx9.html	Bruce.yang的嵌入式之旅 1842
gpio_direction_output vs gpio_set_value之间的使用关系 在linux驱动中常常会碰到gpio_set_value (port_num,0/1) 或gpio_direction_output (port_num,0/1) 这两者有什么关系呢 gpio_set_val...	shuwu 2万+
Linux GPIO子系统分析 在设备驱动中对GPIO的操作是非常普遍的，linux内核为我们提供了GPIO子系统，方便用户使用，它为用户提供了GPIO的统一操作接口，用户不需要关心...	u012830148的博客 2320
Linux内核驱动之GPIO子系统(一)GPIO的使用 热门推荐 分类：Linux内核驱动2012-10-31 21:12 162人阅读 评论(1) 收藏 举报 目录(?)[+] 一 概述 Linux内核中gpio是最简单，最常用的资源(和 interrupt,dma,ti...	kerson的专栏 6万+
gpio_direction_output 和 gpio_set_value用法的区别 最近做的项目，bring up NFC时，I2C设备挂载成功，但是写数据时失败，提示SLAVE无响应。 [45.074021] [1: AsyncTask #1: 2429] i2c-msm-v2 78b...	yt_999的博客 1万+

