2_LED实验

ioremap函数

- 概念：将物理地址映射为虚拟地址，便于程序通过虚拟地址操作硬件寄存器
- void __iomem *ioremap(phys_addr_t paddr,unsigned long size)
  - addr：需要映射的物理地址
  - size：需要映射的字节数
  - 返回映射后的虚拟地址指针

iounmap函数

- 概念：取消之前用ioremap创建的地址映射，释放虚拟地址空间
- void iounmap(void *addr)
  - addr：取消映射的虚拟地址

标准IO读写函数

- `ioread8()` - 读1字节
- `ioread16()` - 读2字节
- `ioread32()` - 读4字节
- `iowrite8(data, addr)` - 写1字节
- `iowrite16(data, addr)` - 写2字节
- `iowrite32(data, addr)` - 写4字节

实验流程（硬件部分）

- 查看原理图，确定LED的GPIO管脚号 --- GPIO0_C7
- 配置**引脚复用**（选择引脚功能为GPIO）
  查文档，确定寄存器是PMU_GRF_GPIO0C_IOMUX_H
- 配置**引脚电平**
  查文档，确定寄存器是GPIO_SWPORT_DR_H
- 配置**引脚输出模式**
  查文档，确定寄存器是GPIO_SWPORT_DDR_H
- 配置**引脚上下拉**
  查文档，确定寄存器是GRF_GPIOA_P

实验流程（软件部分）

- 1、通过define定义寄存器物理地址（硬编码）
- 2、创建LED字符设备自定义结构体（struct led_chrdev）
  - sturct cdev dev
  - unsigned int __iomem *va_dr
  - unsigned int __iomem *va_ddr
  - unsigned int led_pin
- 3、将上述结构体，实例化为led_cdev数组，先写一个成员，且其pin=7
- 4、编写加载函数
  - ①将映射后的虚拟地址赋值给led_cdev[0]
  - ②申请设备号 alloc_chrdev_region()
  - ③创建自定义结构体的类 class_create()
  - ④遍历数组，获得各个自定义结构体
    初始化cdev：cdev_init
    分配完成的设备号：MKDEV
    添加到散列表：cdev_add
    创建设备节点：device_create（用到了类）
- 5、编写卸载函数
  - ①遍历数组，释放虚拟地址
  - ②遍历数组，释放设备节点和cdev
  - ③释放设备号
  - ④删除设备类
- 6、编写open函数
  - ①通过container_of函数获取自定义结构体
  - **②private_data指向自定义的结构体**
  - ③通过自定义结构体，操作虚拟地址，实现寄存器初始状态
- 7、编写write函数
  - ①在private_data获取自定义结构体
  - ②通过自定义结构体中的虚拟地址，读取用户传入的数据，实现电平的控制
- 8、编写release函数
  - 功能和exit一样，直接rteurn 0
- 9、创建file_operations结构体，绑定上面的函数
  - 在cdev_init会用到

cdev_init（cdev，fops）