一、 ioremap() 函数基础概念

几乎每一种外设都是通过读写设备上的相关寄存器来进行的，通常包括控制寄存器、状态寄存器和数据寄存器三大类，外设的寄存器通常被连续地编址。根据 CPU 体系结构的不同，CPU 对 IO 端口的编址方式有两种：

a -- I/O 映射方式（I/O-mapped）

典型地，如 X86 处理器为外设专门实现了一个单独的地址空间，称为"I/O 地址空间"或者"I/O 端口空间"，CPU 通过专门的 I/O 指令（如 X86 的 IN 和 OUT 指令）来访问这一空间中的地址单元。

b -- 内存映射方式（Memory-mapped）

RISC 指令系统的 CPU（如 ARM、PowerPC 等）通常只实现一个物理地址空间，外设 I/O 端口成为内存的一部分。此时，CPU 可以象访问一个内存单元那样访问外设 I/O 端口，而不需要设立专门的外设 I/O 指令。

但是，这两者在硬件实现上的差异对于软件来说是完全透明的，驱动程序开发人员可以将内存映射方式的 I/O 端口和外设内存统一看作是"I/O 内存"资源。

一般来说，在系统运行时，外设的 I/O 内存资源的物理地址是已知的，由硬件的设计决定。但是 CPU 通常并没有为这些已知的外设 I/O 内存资源的物理地址预定义虚拟地址范围，驱动程序并不能直接通过物理地址访问 I/O 内存资源，

而必须将它们映射到核心虚地址空间内（通过页表），然后才能根据映射所得到的核心虚地址范围，通过访内指令访问这些 I/O 内存资源。

Linux 在 io.h 头文件中声明了函数 ioremap（），用来将 I/O 内存资源的物理地址映射到核心虚地址空间（3GB－4GB）中（这里是内核空间），原型如下：

1、ioremap 函数

ioremap 宏定义在 asm/io.h 内：

#define ioremap(cookie,size)　　__ioremap(cookie,size,0)

__ioremap 函数原型为(arm/mm/ioremap.c)：

void __iomem * __ioremap(unsigned long phys_addr, size_t size, unsigned long flags);

参数：

phys_addr：要映射的起始的 IO 地址

size：要映射的空间的大小

flags：要映射的 IO 空间和权限有关的标志

该函数返回映射后的内核虚拟地址(3G-4G). 接着便可以通过读写该返回的内核虚拟地址去访问之这段 I/O 内存资源。

2、iounmap 函数

iounmap 函数**用于取消 ioremap（）所做的映射**，原型如下：

void iounmap(void * addr);

二、 ioremap() 相关函数解析

在将 I/O 内存资源的物理地址映射成核心虚地址后，理论上讲我们就可以象读写 RAM 那样直接读写 I/O 内存资源了。为了保证驱动程序的跨平台的可移植性，我们**应该使用 Linux 中特定的函数来访问 I/O 内存资源，而不应该通过指向核心虚地址的指针来访问**。

读写 I/O 的函数如下所示：

a -- writel()

writel()往内存映射的 I/O 空间上**写数据**，wirtel() I/O 上写入 32 位数据 (4 字节)。

原型：**void writel (unsigned char data , unsigned int addr)**

b -- readl()

readl() 从内存映射的 I/O 空间上**读数据**,readl 从 I/O 读取 32 位数据（4 字节)。
原型：**unsigned char readl (unsigned int addr)**