一、在单片机和裸机中我们操作硬件是如何操作的？

裸机上和单片机上可以这样实现操作:

比如说：此时想要操作寄存器，就可以定义一个指针：

unsigned int\* p = 0x12345678; 指向这个寄存器

\*p = 87654321; 实现对寄存器的操作。

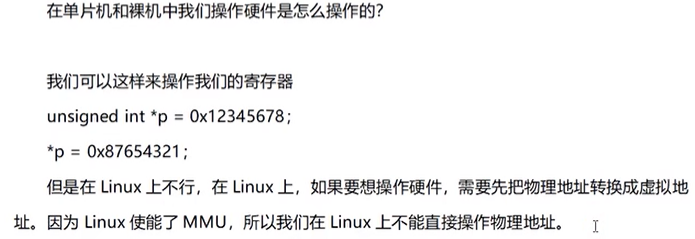
但是，在linux操作系统上我们想要对硬件进行操作，这种方法是不行的。

因为linux不能直接对寄存器的物理地址进行读写操作的。

但学习驱动不能避免操作硬件。操作寄存器就又避免不了操作物理内存地址。

所以要操作物理地址就需要先将物理地址转化为虚拟地址。

（Linux不能直接操作物理地址的原因是：其使能了MMU（Memory Management Unity）内存管理单元）



二、使能了MMU之后有什么好处？

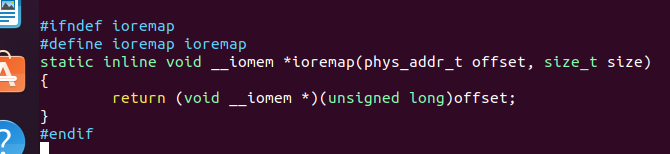
1. 让虚拟地址成为了可能：因为MMU完成了虚拟地址到物理地址的映射。
2. 让系统更加安全：有了MMU之后，上层应用看到的内存都是虚拟内存。由于我们上层应用看到的都是虚拟内存，所以上层应用就不能直接访问硬件设备。从而保障了系统的安全。
3. 我们是如何完成 物理地址 到 虚拟地址 的转换的？

MMU实质上是非常的复杂的。但我们的使用上是非常方便的，因为内核已经提供了相关的许多函数：例如：ioremap 和 iounmap

1. ioremap:将物理地址转换成了虚拟地址。

static inline void \_\_iomem \*ioremap(phys\_addr\_t offset, size\_t size)

其定义路径为： include/asm-generic/io.h 文件中:



这个函数的第一个参数：phys\_addr\_t offset就是映射物理地址的起始地址。

第二个参数：size\_t size 就是要映射多大的内存空间。

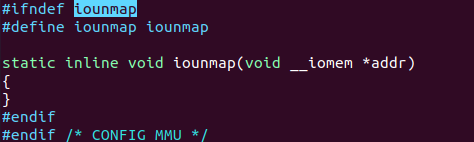
其返回值：映射成功了的话，其将返回虚拟地址的首地址。

失败了的话，就会返回NULL。

1. iounmap:释放掉ioremap映射的地址。

static inline void iounmap(void \_\_iomem \*addr)

其参数是：要取消映射的虚拟地址的首地址。

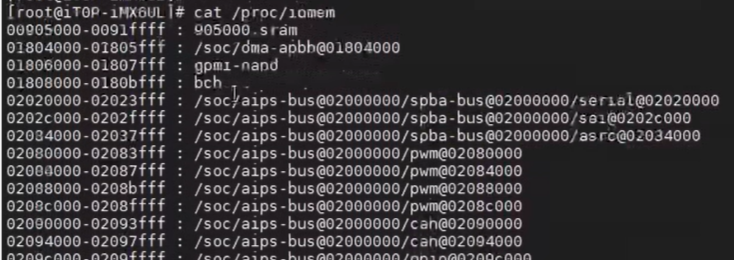


注意：物理地址只能被映射一次。多次映射将会失败。

1. 既然多次映射会失败，我们如何查看哪些物理地址被映射了？

我们可以连接开发板，然后通过命令行进行查看哪些物理地址被映射过了：

执行命令：cat /proc/iomem



那么我们看到的这些都是被映射过的。这些大部分都是自带的。如果是想要给它取消掉的话，只能去找到其对应的相关驱动，然后把对印的驱动去掉，才可以取消这些映射。