**Linux驱动中的platform**

**作者：李智敏,**[华清远见嵌入式学院](http://www.embedu.org/)上海中心讲师。

从Linux2.6内核起，引入一套新的驱动管理和注册机制：platform\_device 和 platform\_driver 。Linux 中大部分的设备驱动，都可以使用这套机制，设备用 platform\_device 表示；驱动用 platform\_driver 进行注册。

Linux platform driver 机制和传统的device driver机制（即：通过 driver\_register 函数进行注册）相比，一个十分明显的优势在于platform机制将设备本身的资源注册进内核，由内核统一管理，在驱动程序中用使用这些资源时，通过platform device提供的标准接口进行申请并使用。

platform 是一个虚拟的地址总线，相比 PCI、USB，它主要用于描述SOC上的片上资源。platform 所描述的资源有一个共同点：在CPU 的总线上直接取址。平台设备会分到一个名称（用在驱动绑定中）以及一系列诸如地址和中断请求号（IRQ）之类的资源。

platform 总线下驱动的开发步骤是：

1、 设备

需要实现的结构体是：platform\_device 。

1）初始化 resource 结构变量  
        2）初始化 platform\_device 结构变量  
        3）向系统注册设备：platform\_device\_register。

以上三步，必须在设备驱动加载前完成，即执行platform\_driver\_register（）之前，原因是驱动注册时需要匹配内核中所有已注册的设备名。platform\_driver\_register（）中添加device到内核最终还是调用的device\_add函数。Platform\_device\_add和device\_add最主要的区别是多了一步insert\_resource(p, r)，即将platform资源(resource)添加进内核，由内核统一管理。

2、驱动

驱动注册中，需要实现的结构体是：platform\_driver 。

在驱动程序的初始化函数中，调用了platform\_driver\_register()注册 platform\_driver 。需要注意的是：platform\_driver 和 platform\_device 中的 name 变量的值必须是相同的 。这样在 platform\_driver\_register() 注册时，会将当前注册的 platform\_driver 中的 name 变量的值和已注册的所有 platform\_device 中的 name 变量的值进行比较，只有找到具有相同名称的 platform\_device 才能注册成功。当注册成功时，会调用 platform\_driver 结构元素 probe 函数指针。

platform\_driver\_register()的注册过程：

1.platform\_driver\_register(&s3c2410fb\_driver)   
        2.driver\_register(&drv->driver)  
        3.bus\_add\_driver(drv)   
        4.driver\_attach(drv)   
        5.bus\_for\_each\_dev(drv->bus, NULL, drv, \_\_driver\_attach)  
        6.\_\_driver\_attach(struct device \* dev, void \* data)  
        7.driver\_probe\_device(drv, dev)  
        8.really\_probe(dev, drv)

在really\_probe()中：为设备指派管理该设备的驱动：dev->driver = drv, 调用probe()函数初始化设备：drv->probe(dev)。