Title：板级I2C 适配器驱动静态注册流程

Time：2016-2-26 11:46:43

以AM3359为例，整体流程：

**platform\_driver注册流程：**

1. **subsys\_initcall(****omap\_i2c\_init\_driver) )--> platform\_driver\_register(&omap\_i2c\_init\_driver)--> driver\_register(&drv->driver)**

subsys\_initcall：子系统初始化入口

platform\_driver\_register：

drv->driver.bus = &platform\_bus\_type;

if (drv->probe) drv->driver.probe = platform\_drv\_probe;

if (drv->remove) drv->driver.remove = platform\_drv\_remove;

if (drv->shutdown) drv->driver.shutdown = platform\_drv\_shutdown;

可见，platform.c的一系列函数(platform\_drv\_probe等)在这里都将赋值给传递下来的platform\_driver里的成员struct device\_driver driver里的函数指针。（后面可以看到，在id匹配成功后，会调用到此处的platform\_drv\_probe）

1. **driver\_register(&drv->driver) -->bus\_add\_driver(drv) -->driver\_attach(drv) -->**

**bus\_for\_each\_dev(drv->bus, NULL, drv, \_\_driver\_attach)**

bus\_for\_each\_dev：

while ((dev = next\_device(&i)) && !error)

error = fn(dev, data);

探测设备：fn即传递下来的\_\_driver\_attach函数指针，而dev则是内核链表挂载的设备。

1. **\_\_driver\_attach-->driver\_match\_device(drv, dev) -->driver\_probe\_device(drv, dev)**

driver\_match\_device:

drv->bus->match(dev, drv)

对于platform\_driver，在步骤1中调用的platform\_driver\_register 中：drv->driver.bus = &platform\_bus\_type; 而platform\_bus\_type的.match = platform\_match，所以drv->bus->match最终会调用platform\_match。

跳转到platform.c的platform\_match函数中：首先，进行of\_driver\_match\_device(与内核加入的设备树特性有关)，如果匹配成功直接返回，否则继续匹配id\_table。一旦匹配成功，则进行下一步：设备和驱动的绑定（probe）

1. **driver\_probe\_device(drv, dev) -->drv->probe(dev)或者dev->bus->probe(注)-->**

**platform\_drv\_probe--> drv->probe(dev)**

从步骤1中可以看到，此处的drv->probe即platform\_drv\_probe，而platform\_drv\_probe(struct device \*\_dev)中通过to\_platform\_driver(\_dev->driver)获取了struct device\_driver的父对象platform\_driver（也就是一开始platform\_driver\_register中传递下来的xxx\_platform\_driver指针），如此，最终在platform\_drv\_probe中将会实现xxx\_platform\_driver的probe函数的调用。（饶了一大圈又回到原点）

注：

If(dev->bus->probe) {

Ret = dev->bus->probe(dev);

If (ret)

Goto probe\_failed;

} else if (drv->probe) {

Ret = drv->probe(dev);

}

总结:无论是设配器驱动注册（platform\_driver\_register），还是具体的从设备驱动注册（如i2c\_driver\_registers）,最终都会调用到driver本身实现的probe函数来进行驱动和设备的绑定。

1. 对于具体的xxx\_xxx\_probe函数，其中除了主要的硬件模块初始化，中断处理程序注册外，就是万众瞩目的I2C适配器注册。

adap->nr = pdev->id; //给定总线号

r = i2c\_add\_numbered\_adapter(adap);