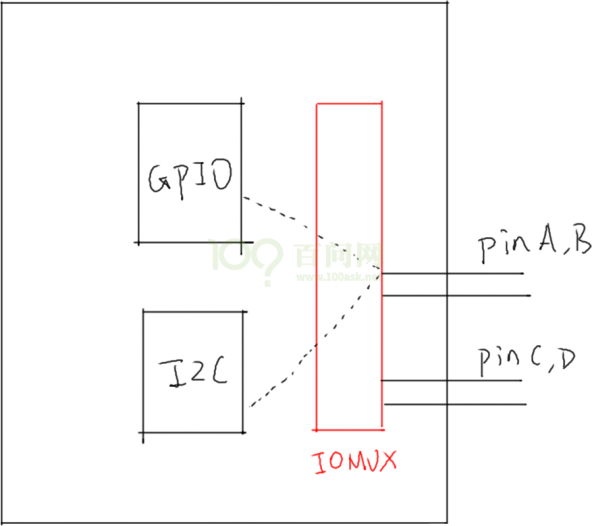
### 16.1 Pinctrl子系统重要概念

#### 16.1.1 引入

无论是哪种芯片，都有类似下图的结构：



要想让pinA、B用于GPIO，需要设置IOMUX让它们连接到GPIO模块；

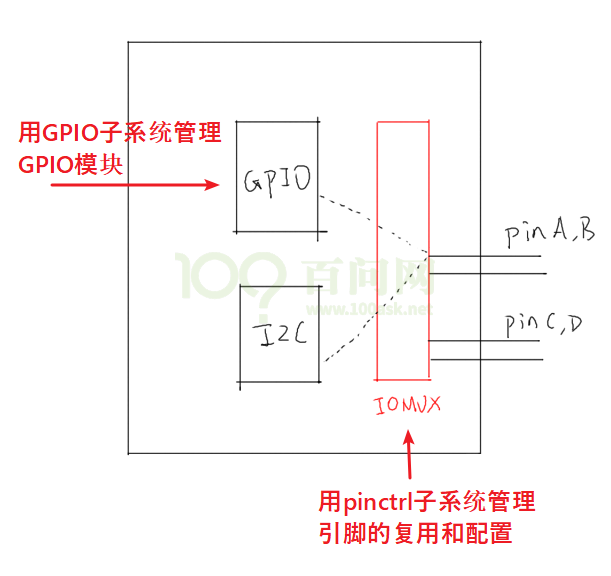
要想让pinA、B用于I2C，需要设置IOMUX让它们连接到I2C模块。

所以GPIO、I2C应该是并列的关系，它们能够使用之前，需要设置IOMUX。有时候并不仅仅是设置IOMUX，还要配置引脚，比如上拉、下拉、开漏等等。

现在的芯片动辄几百个引脚，在使用到GPIO功能时，让你一个引脚一个引脚去找对应的寄存器，这要疯掉。术业有专攻，这些累活就让芯片厂家做吧──他们是BSP工程师。我们在他们的基础上开发，我们是驱动工程师。开玩笑的，BSP工程师是更懂他自家的芯片，但是如果驱动工程师看不懂他们的代码，那你的进步也有限啊。

所以，要把引脚的复用、配置抽出来，做成Pinctrl子系统，给GPIO、I2C等模块使用。

BSP工程师要做什么？看下图：



等BSP工程师在GPIO子系统、Pinctrl子系统中把自家芯片的支持加进去后，我们就可以非常方便地使用这些引脚了：点灯简直太简单了。

等等，GPIO模块在图中跟I2C不是并列的吗？干嘛在讲Pinctrl时还把GPIO子系统拉进来？

大多数的芯片，没有单独的IOMUX模块，引脚的复用、配置等等，就是在GPIO模块内部实现的。

在硬件上GPIO和Pinctrl是如此密切相关，在软件上它们的关系也非常密切。

所以这2个子系统我们一起讲解。

#### 16.1.2 重要概念

从设备树开始学习Pintrl会比较容易。

主要参考文档是：内核Documentation\devicetree\bindings\pinctrl\pinctrl-bindings.txt

这会涉及2个对象：pin controller、client device。

前者提供服务：可以用它来复用引脚、配置引脚。

后者使用服务：声明自己要使用哪些引脚的哪些功能，怎么配置它们。

a. pin controller：

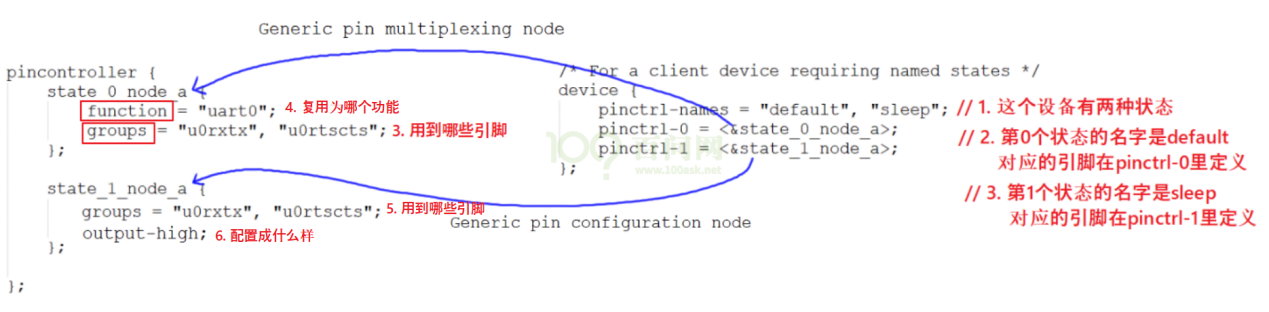
在芯片手册里你找不到pin controller，它是一个软件上的概念，你可以认为它对应IOMUX──用来复用引脚，还可以配置引脚(比如上下拉电阻等)。

注意，pin controller和GPIO Controller不是一回事，前者控制的引脚可用于GPIO功能、I2C功能；后者只是把引脚配置为输入、输出等简单的功能。即先用pin controller把引脚配置为GPIO，再用GPIO Controler把引脚配置为输入或输出。

b. client device

“客户设备”，谁的客户？Pinctrl系统的客户，那就是使用Pinctrl系统的设备，使用引脚的设备。它在设备树里会被定义为一个节点，在节点里声明要用哪些引脚。

下面这个图就可以把几个重要概念理清楚：



上图中，左边是pin controller节点，右边是client device节点：

a. pin state：

对于一个“client device”来说，比如对于一个UART设备，它有多个“状态”：default、sleep等，那对应的引脚也有这些状态。

怎么理解？

比如默认状态下，UART设备是工作的，那么所用的引脚就要复用为UART功能。

在休眠状态下，为了省电，可以把这些引脚复用为GPIO功能；或者直接把它们配置输出高电平。

上图中，pinctrl-names里定义了2种状态：default、sleep。

第0种状态用到的引脚在pinctrl-0中定义，它是state\_0\_node\_a，位于pincontroller节点中。

第1种状态用到的引脚在pinctrl-1中定义，它是state\_1\_node\_a，位于pincontroller节点中。

当这个设备处于default状态时，pinctrl子系统会自动根据上述信息把所用引脚复用为uart0功能。

当这这个设备处于sleep状态时，pinctrl子系统会自动根据上述信息把所用引脚配置为高电平。

b. groups和function：

一个设备会用到一个或多个引脚，这些引脚就可以归为一组(group)；

这些引脚可以复用为某个功能：function。

当然：一个设备可以用到多组引脚，比如A1、A2两组引脚，A1组复用为F1功能，A2组复用为F2功能。

c. Generic pin multiplexing node和Generic pin configuration node

在上图左边的pin controller节点中，有子节点或孙节点，它们是给client device使用的。

可以用来描述复用信息：哪组(group)引脚复用为哪个功能(function)；

可以用来描述配置信息：哪组(group)引脚配置为哪个设置功能(setting)，比如上拉、下拉等。

**注意**：pin controller节点的格式，没有统一的标准！！！！每家芯片都不一样。

甚至上面的group、function关键字也不一定有，但是概念是有的。

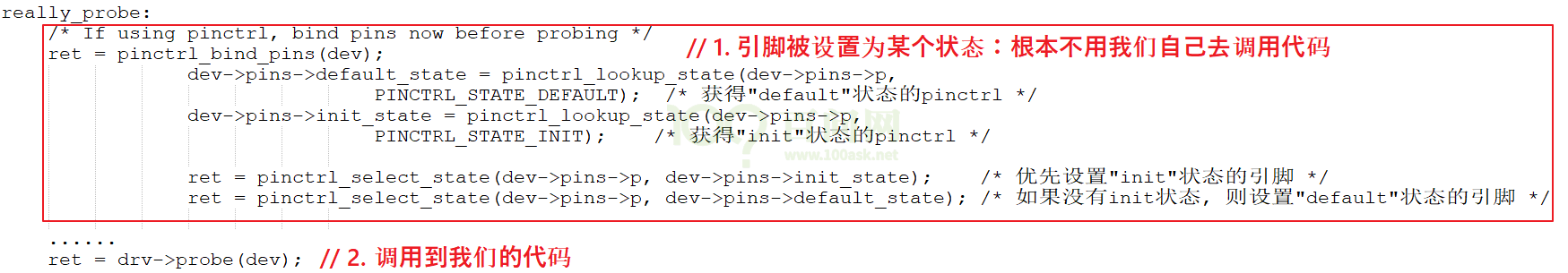
#### 16.1.3 示例



#### 16.1.4 代码中怎么引用pinctrl

这是透明的，我们的驱动基本不用管。当设备切换状态时，对应的pinctrl就会被调用。

比如在platform\_device和platform\_driver的枚举过程中，流程如下：



当系统休眠时，也会去设置该设备sleep状态对应的引脚，不需要我们自己去调用代码。

非要自己调用，也有函数：

devm\_pinctrl\_get\_select\_default(struct device \*dev); // 使用"default"状态的引脚

pinctrl\_get\_select(struct device \*dev, const char \*name); // 根据name选择某种状态的引脚

pinctrl\_put(struct pinctrl \*p); // 不再使用, 退出时调用