一，设备树基本框架

<1>设备树从根节点开始，每个设备都是一个节点。

<2>节点和节点之间可以互相嵌套，形成父子关系。

<3>设备的属性用key-value对(键值对)来描述，每个属性用分号结束

二，设备树语法

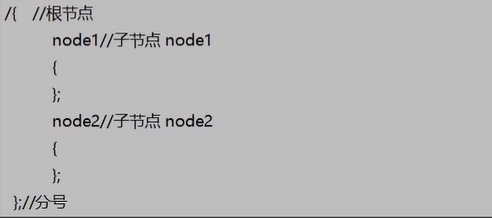
2.1节点

什么是节点呢？节点就好比一颗大树，从树的主干开始，然后有一节一节的树枝，这

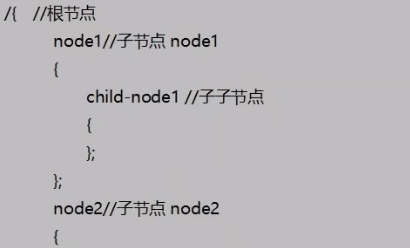
个就叫节点。在代码中的节点是什么样子的呢。我们把上面模板中的根节点摘出来，如下图所示，这个就是根节点。相当于大树的树干。

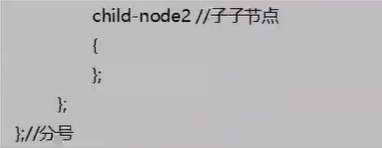


而树枝就相当于设备树的子节点，同样我们把子节点摘出来就是根节点里面的node1和node2，如下图所示:



一个树枝是不是也可以继续分成好几个树枝呢，也就是说子节点里面可以包含子子节点。所以child-node1和child-node2是node1和node1的子节点，如下图所示：





2.2节点名称

节点的命名有一个固定的格式。

IMG_256

<名称>节点的名称也不是任意起的，一般要体现设备的类型而不是特点的型号，比如网口，应该命名为ethernet，而不是随意起一个，比如111。

<设备地址>就是用来访问该设备的基地址。但并不是说在操作过程中来描述一个硬件的地址，他主要用来区分用。

注意事项：

<1>同一级的节点 只要地址不一样，名字是可以相同的。

<2>设备地址是一个可选选项，可以不写。但为了容易区分和理解，一般是都写的。

2.3节点别名

当我们找一个节点的时候，我们必须书写完整的节点路径，如果我们的节点名很长，那么我们在引用的时候就十分不方便，所以，设备树允许我们用下面的形式为节点标注引用（起别名）。

举例：

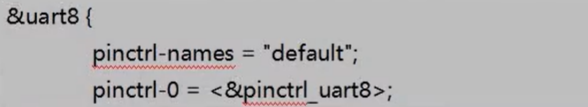
IMG_256

其中，uart8就是这个节点名称的别名，serial@02288000就是节点名称。

2,4, 节点的引用

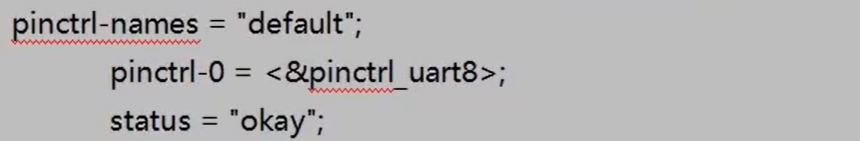
一般我们往一个节点里面添加内容的时候，不会直接把添加的内容写到节点里面，而是通过节点的引用来添加。

举例：





&uart8表示引用节点别名为uart8的节点，并往这个节点里面添加以下内容：



注意事项：

编译设备树的时候，相同的节点的不同属性信息都会被合并，相同节点的相同的属性 会被重写，使用引用可以避免移植者四处找节点。如dts和dtsi里面都有根节点，但最终会合并成一个根节点。

2.5属性

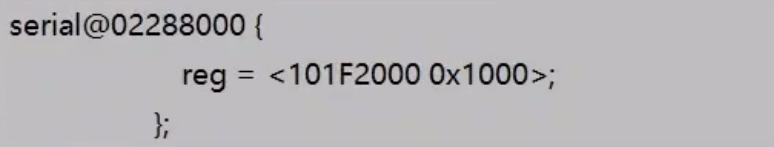
(1） reg 属性

reg属性用来描述一个设备的地址范围。

格式：

IMG_256

举例：



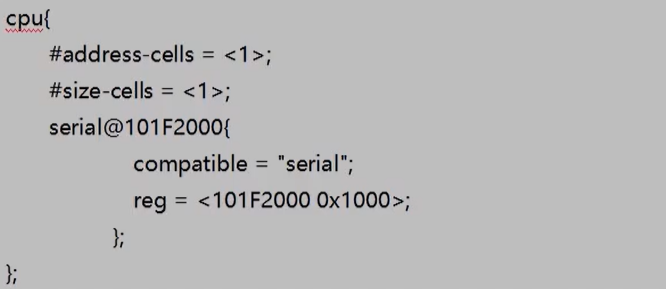
其中101F2000就是起始地址，0x1000就是长度。

1. #address-cells 和#size-cells 属性

#address-cells用来设置 子节点中 reg地址 的数量

#size-cells用来设置 子节点中 reg地址 长度的数量。

举例：



其中#address-cells和#size-cell均为1，也就是说我们子节点里面的reg属性里这个 寄存器组 的起始地址 只有一个，长度也只有一个。所以101F2000是起始地址，0x1000是长度。

(3) compatible属性

compatible是一个字符串列表，可以在代码中进行匹配用的字符串。

举例：

IMG_256

(4) status属性

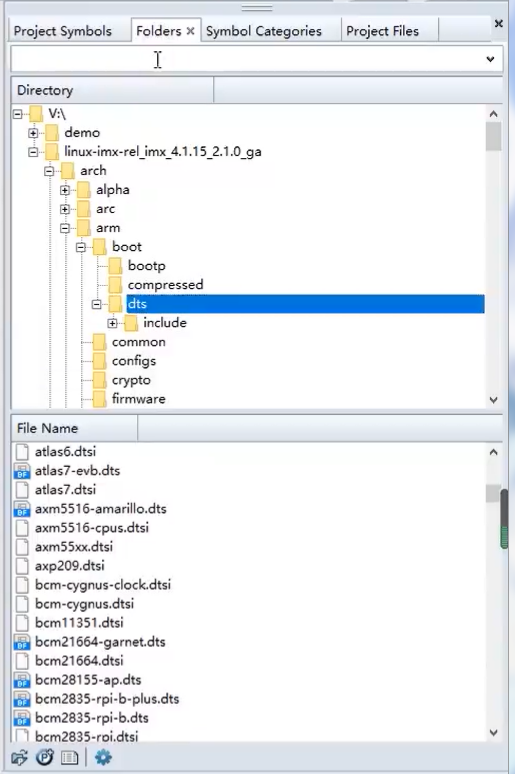
status属性的值类型是字符串，这里我们只要记住俩个常用的即可，一个是okay，表示设备可以正常使用，一个是disable，表示设备不能正常使用。

——————————————————————————————————

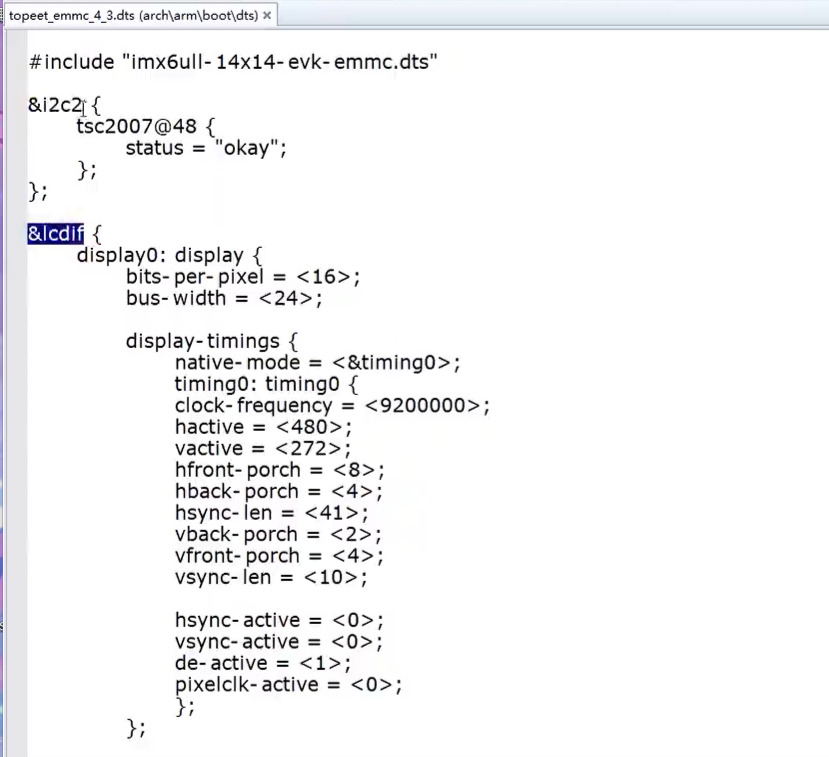
实操：

我们来看我们的设备树文件：

位于：内核源码的arch/arm/boot/dts/ 我们的设备树文件都在这个dts目录下面。



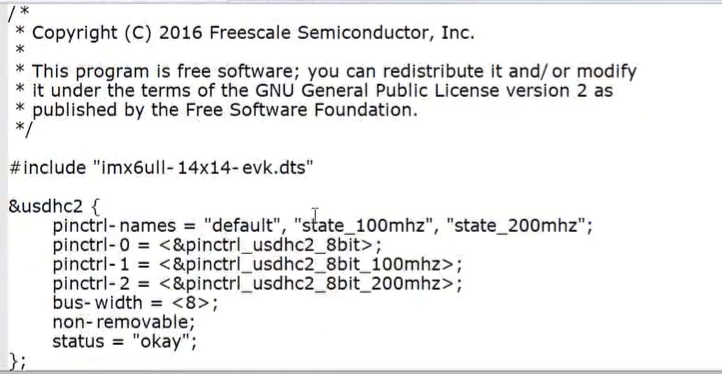
这里以topeet\_emmc\_4\_3.dts文件为例：



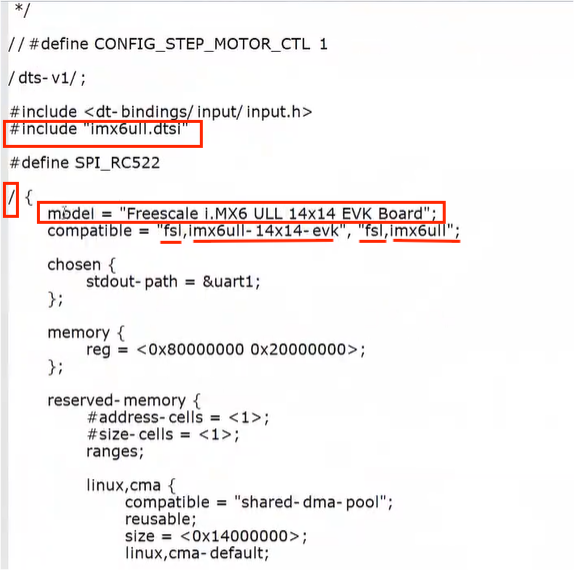
在这个设备树文件中引用了两个节点，也就是说给这两个节点追加了内容。给i2c2和lcdif节点追加了如上内容。

我们发现这个文件里面并没有根节点，那么根节点在哪里呢？

我们进入头文件：



再继续进入头文件：



在这个文件里面，我们就可以看到我们的根节点。

Model：里面是字符串，用来描述我们开发板的准确型号的。

Compatible：里面有4个部分，就说明它可以匹配多个驱动，因为我们匹配的时候要用到Compatible属性。

Chosen：这是个特殊的属性，它是用来设置u-boot环境变量的。

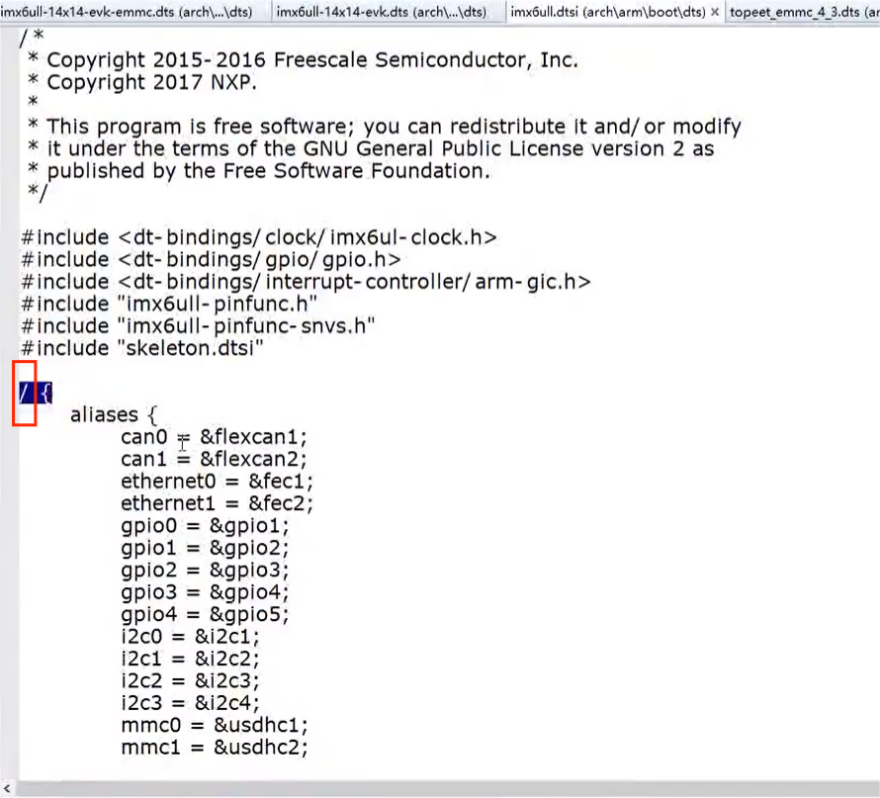
/\*

设备树看不懂很正常，因为设备树需要结合驱动去看。因为设备树代替了我们原来的device.c部分。那么当device和driver匹配成功的时候。driver就会从device里面拿取资源。那么当driver拿到资源之后，又会去完成相应的需求功能。

那么设备树里面描述的 就都是相应的一些资源。这些资源并不是指特定的 哪一个灯、蜂鸣器之类的。而是指cpu上的相关的寄存器和相关的中断、DMA、时钟等资源。 那么这些资源被描述之后，我们在使用的时候又从设备树取出来。所以具体设备树里面的某一块代码的具体的作用，需要根据实际驱动来判断。

\*/

回到开头,我们可以看到里面有一个dtsi文件：我们进入这个文件：



我们可以看到它也有一个 根节点。那么可能会问：为什么会有两个根节点？是不是哪里有问题？

我们在2.4部分其实分析过：相同的节点的相同属性会被重写。相同节点的不同属性会被合并。所以实际上，这两个根节点最终会被合并成一个根节点。