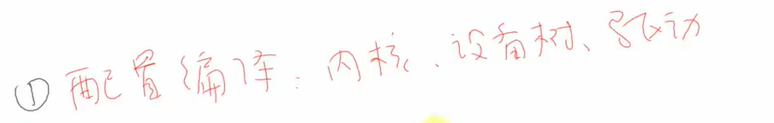
编译设备树 在linux内核目录下，即Linux-4.9.88。命令：make dtbs

板子上设备树的地址：/boot/100ask\_imx6ull-14x14.dtb

查询地址/sys/firmware/devicetree/base

/sys/devices/platform



装载驱动程序insmode \*\*.ko

查看驱动安装cat /proc/devices

lsmod

ls /dev/helloc -l

允许程序运行 chmod +x \*\*

运行程序 ./\*\*

从内核中查看打印信息

dmesg

dmesg | grep \*\*.c

cat /proc/devices

卸载驱动

rmmod \*\*

echo "7 4 1 7" > /proc/sys/kernel/printk

传统的写法：单个开发使用的框架

使用哪个引脚，怎么操作引脚，全在驱动程序中，修改硬件时，全都要修改

总线设备驱动模型：可以拓展为多个开发版共同使用的框架

分为了dev和drive

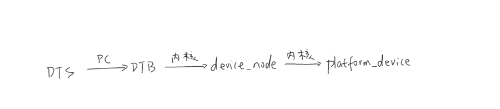
dev中有platform\_device结构体，用来表示指向哪个引脚

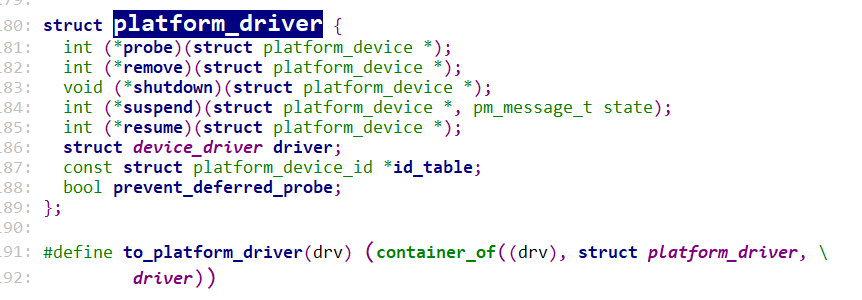
设备树驱动模型：将一些信息转化为设备树信息，这样可以减少内核的臃肿。

设备树的自动编译：gcc -E预编译，进行宏展开，生成一个\*\*.dts.tmp的临时文件，再使用设备树的编译器dtc -O dtb 生成一个设备树的二进制文件。因为使用了gcc，所以#include可以使用。

设备树的子节点有compatile属性可以被转化为platform\_device

设备树的父节点的compatile属性包含“simple-bus”等属性，也可以被转化为platform\_device





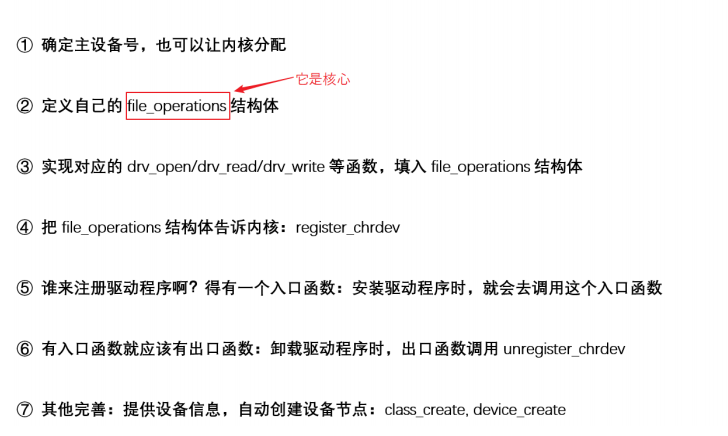
记录资源

device\_create

找不到好的例子，在ubuntu下的linux源码进行查找

在驱动程序中，其他程序想要引用，必须要EXPORT\_SYMBOL把这个函数导出

怎么写驱动设备



major=0，表示分配主设备号

# GPIO和Pinctrl子系统

pinControler控制，bsp工程师实现这种原厂的控制。

client device

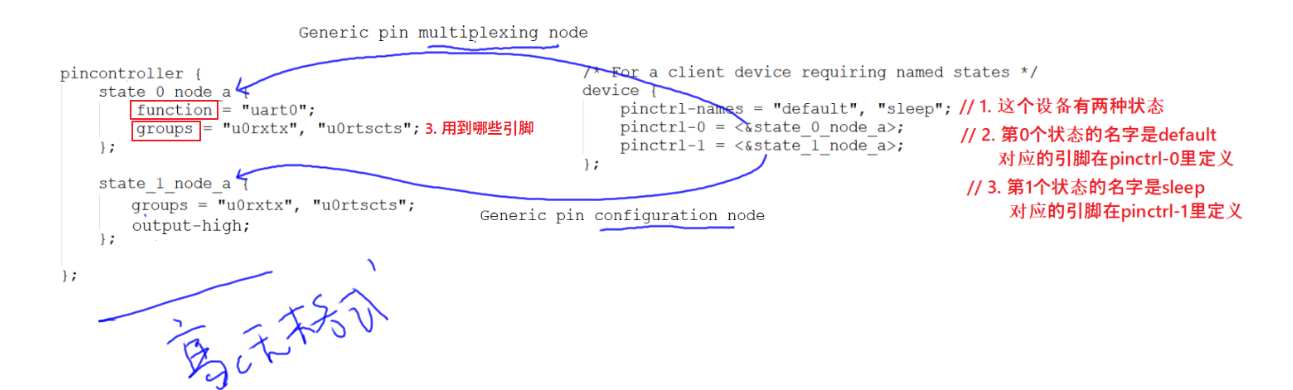
UART，

正常状态

休眠状态

配置节点，复用节点

左侧是controler， 右侧是client



不同开发版，格式不一样，但是概念是一样的：group，function，configuration。

device

哪一组，哪一个，这个的状态

# 中断和异常

对硬件中断的处理：不能嵌套，越快越好

hard irq：Linux系统对硬件等硬件中断的处理，硬件中断，每个中断都有对应的处理函数。

soft irq：

在设备树上声明，我使用哪个中断，上升沿，

interrupt-parent = <&gpio1>

interrupt = <5 RTSING>

hwirq∈domain

irq\_domain 是一个结构体，使用来解析设备树，把这个5解析成一个虚拟中断号irq，会保存在platform\_device。

compatible

在设备树中，中断控制器节点中必须有一个属性：interrupt-controller，表明它是“中断控制器”。

#interrupt-cells=<1>;

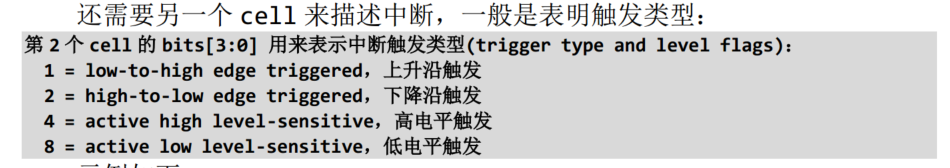
使用上一级哪个中断控制器的哪个中断

interrupts-extended = <&intcl 5 1>, <&intc2 1 0>;



第一个1表示gipo1中的第一个引脚

第二个1表示上升沿触发，可以改为2，表示下降沿触发。



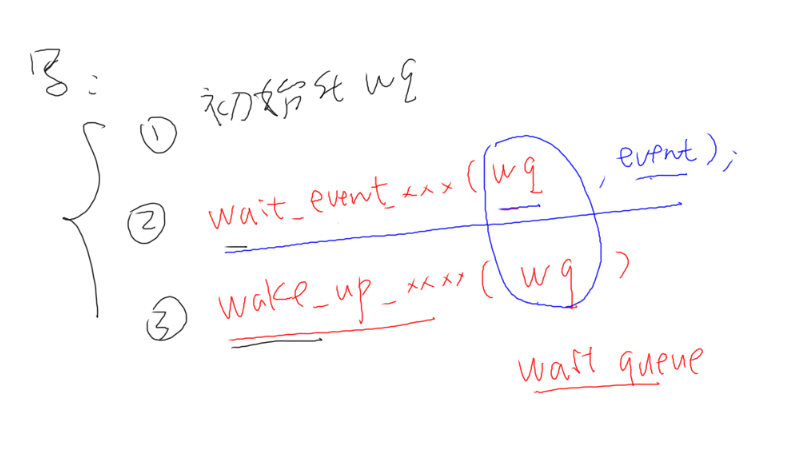
创建一个platform\_driver结构体：需要哪些函数，设备树的匹配信息结构体数组

注册，删除这个结构体

当匹配的时候，就有probe函数被调用：我要取出每一个GPIO，转化为中断号，然后去requestirq

将每个引脚取出来，转化中断号：gpio\_to\_irq()，这个需要知道每个引脚对应的编号，通过函数，of\_get\_gpio\_flags去获得这个编号。

去request irq;用一个全局的数组去保存这些局部的变量，因为以后要释放这些irq还需要知道这些申请的irq，因为不知道具体有多少个数据，使用链表来保存。链表需要提前分配空间大小



休眠唤醒使用两个函数，wait\_event, wake\_up