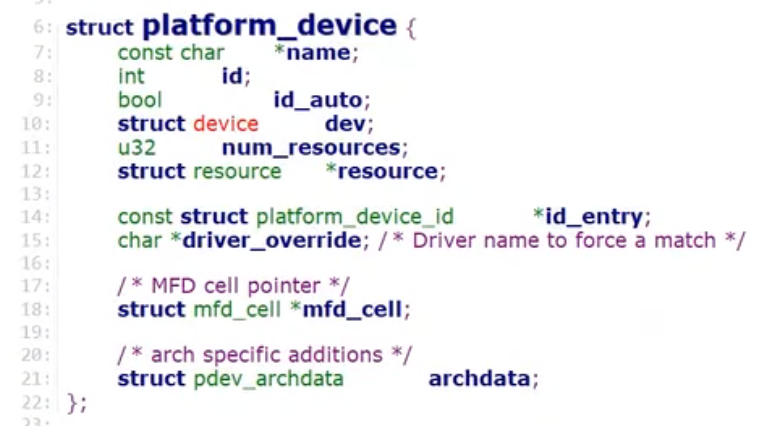
1. 平台总线注册一个device

device.c里面写的是硬件资源，这里的硬件资源是指寄存器的地址，中断号，时钟等硬件资源。

注意：硬件资源并不是指具体的硬件资源。比如说：我们要描述LED灯的硬件资源，它描述的并不是LED灯，而是LED实际上使用了哪些管脚。管脚就又涉及哪些寄存器资源。

那么在linux里面我们是如何描述硬件资源的呢？

在linux内核里面，我们是使用一个结构体来描述硬件资源的。platform\_device这个结构体位于：include/linux/platform\_device.h ：



这个结构体里面描述的成员变量，我们只需要了解用到的其中几个就可以了：

name：平台总线进行匹配的时候用到的name。

（也就是说在我们的device部分，有一个name，那么在driver里面也要有一个name，如果这两个名字一样，那么就会匹配到一起，如果不一样，就不能匹配到一起。同时，在加载device的时候，会在/sys/bus...目录下生成对应的name名称的文件）

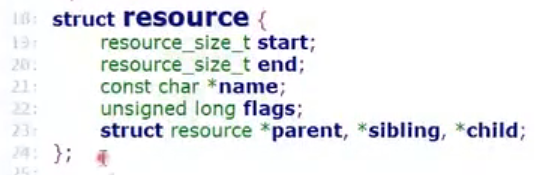
id：设备ID。一般填 -1 即可。它是用来给相同name的设备进行编号的。

dev：结构体在内嵌的device结构体。它是我们设备通用属性的部分。

num\_resources：资源的数量。

resource：device里面的硬件资源。

resource结构体，位于include/linux/ioport.h 文件中：

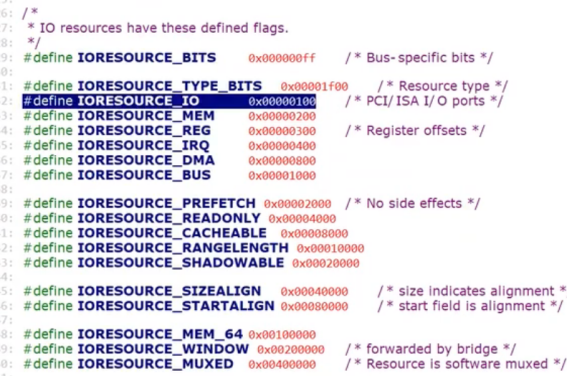


Start：资源的起始

End：资源的结束

Name：资源的名字（可选）

Flages：资源的类型：也位于include/linux/ioport.h 文件中:



Resource\* 用作节点（类似于双向链表之类的）

IORESOURCE\_IO： IO的内存

IORESOURCE\_MEM： 表述一段寄存器的地址

IORESOURCE\_IRQ： 表示中断

IORESOURCE\_DMA： 表示DMA的地址

IORESOURCE\_BUS： 表示总线号，比如说：IIC、SPI

（我们主要关注IRQ和MEM即可）

有人可能会问：为啥这里我们只控制它的数据寄存器就可以了，不用设置其电气属性以及复用关系？是不是少了一些东西？

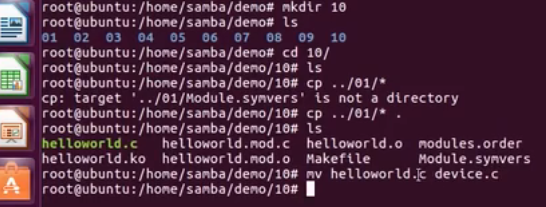
实际上，系统在启动的时候，他会去加载设备树，设备树的概念在后续会讲解。他会把一系列的配置给配好，如果你的开发板没有配这些，就需要老老实实地去配置其复用级文件、电气属性以及方向寄存器等，

———————————————————————————————————————

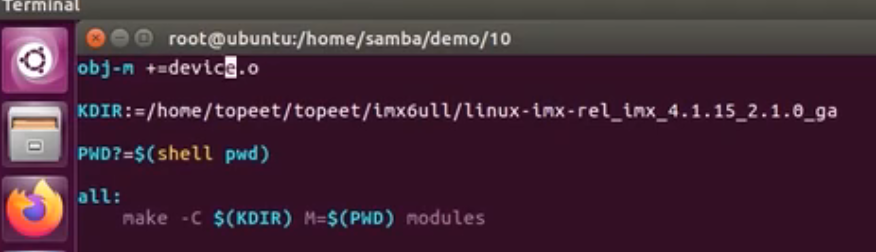
实操：

本节课为第10个实验：

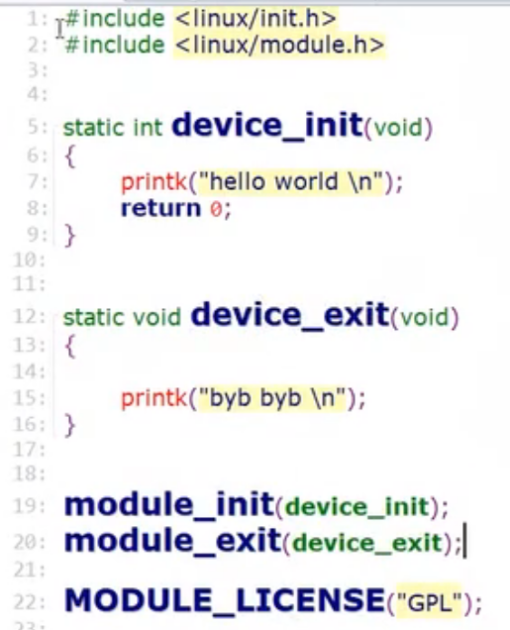
复制01的HelloWorld实验代码：将.c文件重命名为device.c



修改Makefile文件的代码：

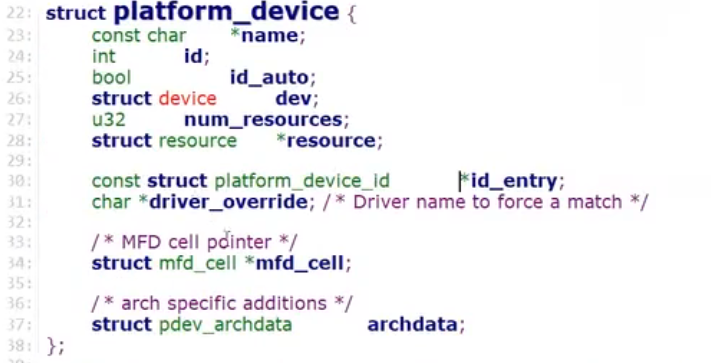


修改device.c文件：



添加头文件：

1740658015864



Name：平台总线进行匹配时用到的name，在/sys/bus...下生成对应name地设备。

Id：设备ID,一般为-1。是用来给具有相同name地设备进行编号的

Device dev：内嵌的device结构体

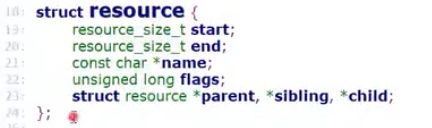
Num\_resource：资源的个数

Resource:硬件资源:寄存器地址、中断号、时钟等

/\*

Resource:

位于：include/linux/ioport.h：



Start:资源的起始地址

End:资源的结束地址

Name:资源的名字

Flags:资源的类型

Flags:

IORESOURCE\_IO：IO的内存

IORESOURCE\_MEM:表述一段物理内存

IORESOURCE\_IRQ:表示中断号

IORESOURCE\_DMA:表示DMA的地址

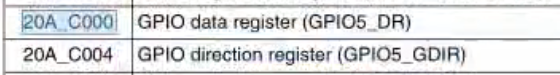
IORESOURCE\_BUS：表示总线号

我们这里为啥在resource结构体里面只控制数据寄存器就可以了？为什么不用去控制其电气属性，设备服用关系？是不是少了一些东西？

实际上，在开发板启动的时候，他会去加载设备树，设备树会去把一系列的配置都给配好。

如果你的开发板不会给你配置这些，你就需要自己去配置复用寄存器、电气寄存器、方向寄存器等，最后再去控制其数据寄存器，给管脚赋值高低电平。实际上到这时，实际上与单片机是一样的。

我们由手册得：蜂鸣器的数据寄存器DR为：20A\_c000，所以起始地址为：0x20AC000



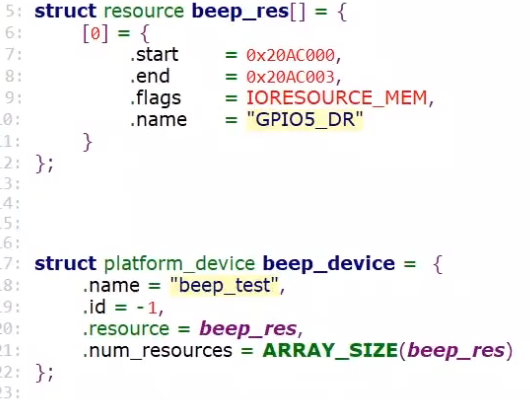
并且其GDIR寄存器地址为：20A\_C004,所以结束地址到3即可：0x20AC003

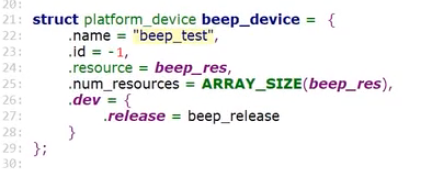
Flags：类型需要使用MEM类型，因为这里描述的一段物理地址。

Name:资源的名称，这里描述的是DR寄存器，所以名称写为GPIO5\_DR。（随意）

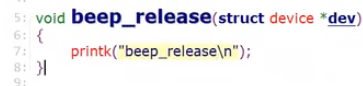
\*/

由上面的信息填充platform\_device结构体，以及 resource资源结构体(二者位于：include/linux/ioport.h)：





实现release:

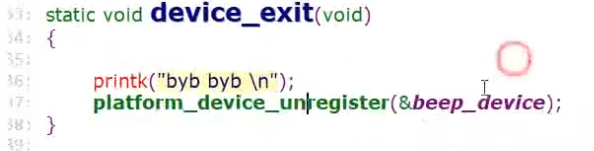


写到这里，device的就写好了。

写好后，我们就需要将其注册到内核。注册使用的是：platform\_device\_register:



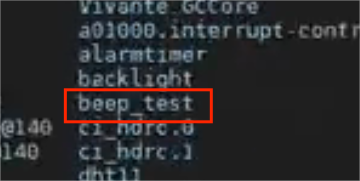
注销：使用的是：platform\_device\_unregister:



编译拷贝执行：

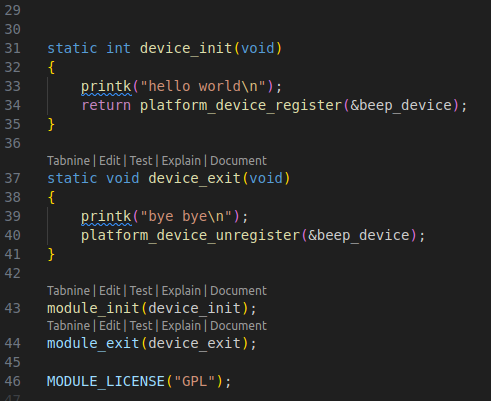


加载驱动前，/sys/bus/platform/devices/ 目录下查找没有beep\_test文件，加载后：



程序：





总结：

device.c文件里：我们描述的是硬件资源，但是硬件资源指的是我们寄存器的地址、中断号、时钟。

我们在描述的时候，就需要在resource结构体中 写其相关的寄存器地址。由于设备树将其他寄存器已经配置好了，所以我们直接就可以只描述DR寄存器。但如果使用的不同的开发板，就需要去找相关的寄存器。同时注意release释放device也是在这个结构体的dev成员中。并且我们要另外实现这个release函数。

resource描述完成之后，我们就可以描述platform\_device 的beep\_device设备了。

platform\_device描述完成之后，就可以在init函数里面直接使用：platform\_device\_register加载到内核中。

加载成功后就可以在 /sys/bus/platform\_device/ 目录下生成一个和 beep\_device结构体中name = “beep\_test”同名的文件。同时这个名字也是平台总线进行匹配的名字。后续driver.c里面也需要根据这个名字进行匹配。