根据自己的理解翻译了<http://lxr.linux.no/linux+v2.6.34/Documentation/i2c/instantiating-devices>中关于枚举建立i2c\_client的文档。有异议或疑问请参照原文，毕竟内核的文档才是真正的精华。s

**方法1：使用总线号声明设备。**

**在内核的初始化中定义设备的信息。前提是内核编译的时候已经确定有哪些i2c设备和它们的地址，还要知道连接的总线的编号。**

比如在/arch/arm/mach-xxxx/board\_xxxx.c中可以有这么一段代码来注册i2c设备的信息。

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/yuanlulu/article/details/6557901)

1. static struct i2c\_board\_info \_\_initdata h4\_i2c\_board\_info[] = {
2. {
3. I2C\_BOARD\_INFO("isp1301\_omap", 0x2d),
4. .irq = OMAP\_GPIO\_IRQ(125),
5. },
6. { /\* EEPROM on mainboard \*/
7. I2C\_BOARD\_INFO("24c01", 0x52),
8. .platform\_data = &m24c01,
9. },
10. { /\* EEPROM on cpu card \*/
11. I2C\_BOARD\_INFO("24c01", 0x57),
12. .platform\_data = &m24c01,
13. },
14. };
15. static void \_\_init omap\_h4\_init(void)
16. {
17. (...)
18. i2c\_register\_board\_info(1, h4\_i2c\_board\_info,
19. ARRAY\_SIZE(h4\_i2c\_board\_info));
20. (...)
21. }



这样注册之后，i2c\_adapter注册的时候就会扫描所有的已注册的2c\_board\_info，并为连接自己的i2c设备建立一个i2c\_client。这样在2c\_board\_info中的同名i2c\_driver注册的时候，i2c\_client就会和i2c\_driver绑定了，i2c\_driver的probe函数被调用。

**方法2:枚举设备。**

**使用**[**i2c\_new\_device**](http://lxr.linux.no/linux+v2.6.34/+code=i2c_new_device)**()或者i2c\_new\_probed\_device（）.**

方法1有诸多限制，必须必须在编译内核的时候知道i2c的总线编号和物理的连接。有时开发者面对的是一个已经存在的系统，无法修改内核。

或者内核开发者移植系统的时候也不知道有哪些i2c设备或者到底有多少i2c总线。

在这种情况下就需要用到i2c\_new\_device()了。它的原型是：

struct i2c\_client \*

i2c\_new\_device(struct i2c\_adapter \*adap, struct i2c\_board\_info const \*info);

这个函数将会使用info提供的信息建立一个i2c\_client并与第一个参数指向的i2c\_adapter绑定。返回的参数是一个i2c\_client指针。

驱动中可以直接使用i2c\_client指针和设备通信了。这个方法是一个比较简单的方法。

获取i2c\_adapter指针的函数是：

struct i2c\_adapter\* i2c\_get\_adapter(int id)；//它的参数是i2c总线编号。

使用完要释放：

void i2c\_put\_adapter(struct i2c\_adapter \*adap)；

如果连i2c设备的地址都是不固定的，甚至在不同的板子上有不同的地址，可以提供一个地址列表供系统探测。

此时应该使用的函数是i2c\_new\_probe\_device.。用法如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/yuanlulu/article/details/6557901)

1. Example (from the pnx4008 OHCI driver):
2. static const unsigned **short** normal\_i2c[] = { 0x2c, 0x2d, I2C\_CLIENT\_END };
3. static **int** \_\_devinit usb\_hcd\_pnx4008\_probe(struct platform\_device \*pdev)
4. {
5. (...)
6. struct i2c\_adapter \*i2c\_adap;
7. struct i2c\_board\_info i2c\_info;
8. (...)
9. i2c\_adap = i2c\_get\_adapter(2);
10. memset(&i2c\_info, 0, sizeof(struct i2c\_board\_info));
11. strlcpy(i2c\_info.name, "isp1301\_pnx", I2C\_NAME\_SIZE);
12. isp1301\_i2c\_client = i2c\_new\_probed\_device(i2c\_adap, &i2c\_info,
13. normal\_i2c);
14. i2c\_put\_adapter(i2c\_adap);
15. (...)
16. }



i2c\_new\_probed\_device的原型是：

struct i2c\_client \*

i2c\_new\_probed\_device(struct i2c\_adapter \*adap,

struct i2c\_board\_info \*info,

unsigned short const \*addr\_list)；

这个函数将会在指定的总线上探测addr\_list中的地址，将第一个有ACK反馈的地址赋给info->addr，

然后使用前两个参数调用i2c\_new\_device。它的返回值也是一个可用的i2c\_client指针。

i2c\_unregister\_device() 可以注销i2c\_new\_device()/i2c\_new\_probed\_device()申请的i2c\_client。

补充：驱动开发者如何知道一个物理i2c总线的编号？

[root@zlg

/]# cat /sys/class/i2c-dev/i2c-0/name

PNX4008-I2C0

[root@zlg /]# cat /sys/class/i2c-dev/i2c-1/name

PNX4008-I2C1

[root@zlg /]# cat /sys/class/i2c-dev/i2c-2/name

USB-I2C

**方法3：在所有i2c总线上探测特定设备。**

内核文档中关于方法2的限制及方法3的好处我没看懂。说一下自己的理解，那就是方法2虽然可以探测多个地址，

但是仅仅能在一个指定的总线上探测，并且探测到第一个可用的地址就停止探测了。如果之前并不确定总线的编号，

或者一次探测多个i2c设备，就需要用到方法3了。

实现方法3需要两个条件：

\*\*\*\*\*\*实现i2c\_driver的detect成员。这个成员函数原型是：

int (\*detect)(struct i2c\_client \*, int kind, struct i2c\_board\_info \*);

这个函数必须检查第二个参数的addr域是否自己支持的地址，是的话则至少填充info->type，info的其它成员也可以填充，但不应该修改addr。

如果是就返回0，否则返回-ENODEV。

\*\*\*\*\*\*初始化i2c\_driver的address\_list成员。i2c\_driver注册的时候，i2c\_core会在所有已经注册的i2c\_adapter上探测address\_list中的所有地址，硬件探测成功之后后调用i2c\_driver的detect成员，然后根据detect填充的info建立一个i2c\_client。如果两个总线上有相同的地址的设备，那么会分别建立两个i2c\_client。如果address\_list中的多个地址都有设备占用，那么会建立多个i2c\_client。

或许因为方法3太过于强大和灵活，内核文档不推荐这种方法。优先选用方法1和2。

**方法4：从用户控件枚举。**

如果编写驱动的时候实在无法知道i2c设备的地址（连可能的地址列表也不知道），那就需要系统运行后从用户空间输入了。

用户空间通过两个sysfs属性文件来建立和删除i2c\_client：new\_device和delete\_device。这两个文件都是只写的。

new\_device有两个参数：i2c设备的名字（字符串）和地址（以0x开头的16进制数）。

delete\_device只有一个参数，那就是设备的地址。

举例：

# echo eeprom 0x50 > /sys/bus/i2c/devices/i2c-3/new\_device

可以看到，此时已经指定了总线编号。

**补充--方法5：在i2c\_driver中的attach\_adapter中调用**[**i2c\_new\_device**](http://lxr.linux.no/linux+v2.6.34/+code=i2c_new_device)**()或者i2c\_new\_probed\_device（）**

**这个方法实质上和方法2类似。**

这样的例子在2.6.34内核的/sound/ppc/keywest.c中。可以参考。

**补充:**

i2c\_driver和i2c\_client的总线类型均为i2c\_bus\_type，i2c\_client的name成员（对应info->type）和i2c\_driver中的id\_table中的名字

是它们相互绑定的依据。

不同的i2c\_adapter上可以挂靠相同地址的设备，但是i2c设备的名字是全局的，因此不同设备的名字不要相同。（并不是不能相同）。