在kernel中要访问i2c adapter，必须知道该adapter被赋值的编号。

1. struct i2c_adapter *i2c_get_adapter(int nr)
这里的nr就是编号

2. 由step 1获得i2c_adapter object
int i2c_transfer(struct i2c_adapter *adap, struct i2c_msg *msgs, int num)
i2c_transfer()支持多个i2c message的操作

3. void i2c_put_adapter(struct i2c_adapter *adap)
release该adaptor

所以这里的关键是知道对应i2c adaptor的编号！

in i2c-pxa.c/i2c_pxa_probe()

```
        /* Default adapter num to device id; i2c_pxa_probe_dt can override. */
        i2c->adap.nr = dev->id;

        ret = i2c_pxa_probe_dt(dev, i2c, &i2c_type);
```

即初始由dev->id设定，但可以由i2c_pxa_probe_dt()改写

```
static int i2c_pxa_probe_dt(struct platform_device *pdev, struct pxa_i2c *i2c,
                            enum pxa_i2c_types *i2c_types)
{
        struct device_node *np = pdev->dev.of_node;
        const struct of_device_id *of_id =
                        of_match_device(i2c_pxa_dt_ids, &pdev->dev);

        if (!of_id)
                return 1;

        /* For device tree we always use the dynamic or alias-assigned ID */
        i2c->adap.nr = -1;

        if (of_get_property(np, "mrvl,i2c-polling", NULL))
                i2c->use_pio = 1;
        if (of_get_property(np, "mrvl,i2c-fast-mode", NULL))
                i2c->fast_mode = 1;
        *i2c_types = (u32)(of_id->data);
        return 0;
}
```

如果是由device-tree设定的i2c，则或是动态分配编号或是通过alias设定

in i2c_pxa_probe()

```
ret = i2c_add_numbered_adapter(&i2c->adap);
```

```
int i2c_add_numbered_adapter(struct i2c_adapter *adap)
{
        if (adap->nr == -1) /* -1 means dynamically assign bus id */
                return i2c_add_adapter(adap);

        return __i2c_add_numbered_adapter(adap);
}
EXPORT_SYMBOL_GPL(i2c_add_numbered_adapter);
```

```
int i2c_add_adapter(struct i2c_adapter *adapter)
{
        struct device *dev = &adapter->dev;
        int id;

        if (dev->of_node) {
                id = of_alias_get_id(dev->of_node, "i2c");      ①
                if (id >= 0) {
                        adapter->nr = id;
                        return __i2c_add_numbered_adapter(adapter);
                }
        }

        mutex_lock(&core_lock);
        id = idr_alloc(&i2c_adapter_idr, adapter,
                       __i2c_first_dynamic_bus_num, 0, GFP_KERNEL);
        mutex_unlock(&core_lock);
        if (id < 0)
                return id;

        adapter->nr = id;

        return i2c_register_adapter(adapter);
}
```

①

```
        aliases {

                ethernet0 = "/stmmac@d0700000";

                serial0 = "/uart@d4030000";

                serial1 = "/uart@d4017000";

                serial2 = "/uart@d4018000";

                serial3 = "/uart@d4016000";

                i2c0 = "/i2c@d4011000";

                i2c1 = "/i2c@d4031000";

                i2c2 = "/i2c@d4032000";

                i2c3 = "/i2c@d4033000";

                i2c4 = "/i2c@d4033800";

                i2c5 = "/i2c@d4034000";

                spi0 = "/ssp@0xd4035000";

                spi1 = "/ssp@0xd4036000";

        };
```

所以在Gemstone2上注册了6个I2C adapter，其编号分别为0,1,2,3,4,5。