```
pegmatite-regulator@2 {
    compatible = "pegmatite-reg";
    reg = <0x0 0xd0630030 0x0 0x8>;
    init-on = <0x1>;
    clocks = <0x33 0x34 0x35>;
    regulator-name = "pegmatite_gpu";
    dev-name = "gpu";
    supply-name = "islandpower";
    status = "disable";
};
```

在dts中对应的device node中添加status = "disable" property可以禁止该device。

原理如下：

当kernel初始化时通过of_platform_populate() function create device时(according to device tree)，如果device node的描述中的status property不是"ok"或"okay"，则该device并不会create，从而该device的driver因为没有device可以match，所以driver的probe()也不会被调用。

in arch/arm/mach-pegmatite/pegmatite.c

```
static void __init pegmatite_dt_init(void)
{
    /* Add devices not supported by device tree */
    platform_add_devices(platform_devices, ARRAY_SIZE(platform_devices));

    of_platform_populate(NULL, of_default_bus_match_table, NULL, NULL);
}
```

of_platform_populate()根据dtb中的device node来创建device。

of_platform_populate() in drivers/od/platform.c

|

|

\|/

of_platform_bus_create() in drivers/od/platform.c

|

|

\|/

of_platform_device_create_pdata() in drivers/od/platform.c

|

|

\|/

of_device_is_available()

```
static struct platform_device *of_platform_device_create_pdata(
                                        struct device_node *np,
                                        const char *bus_id,
                                        void *platform_data,
                                        struct device *parent)
{
        struct platform_device *dev;

        if (!of_device_is_available(np) ||
            of_node_test_and_set_flag(np, OF_POPULATED))
```

```
        return NULL;

    dev = of_device_alloc(np, bus_id, parent);
    if (!dev)
        goto err_clear_flag;

    ......
}

/**
 *  of_device_is_available - check if a device is available for use
 *
 *  @device: Node to check for availability
 *
 *  Returns 1 if the status property is absent or set to "okay" or "ok",
 *  0 otherwise
 */
int of_device_is_available(const struct device_node *device)
{
    unsigned long flags;
    int res;

    raw_spin_lock_irqsave(&devtree_lock, flags);
    res = __of_device_is_available(device);
    raw_spin_unlock_irqrestore(&devtree_lock, flags);
```

```
        return res;


}
EXPORT_SYMBOL(of_device_is_available);


/**
 *  __of_device_is_available - check if a device is available for use
 *
 *  @device: Node to check for availability, with locks already held
 *
 *  Returns 1 if the status property is absent or set to "okay" or "ok",
 *  0 otherwise
 */
static int __of_device_is_available(const struct device_node *device)
{
        const char *status;
        int statlen;


        if (!device)
                return 0;


        status = __of_get_property(device, "status", &statlen);
        if (status == NULL)
                return 1;                   (1)
```

```
        if (statlen > 0) {
                if (!strcmp(status, "okay") || !strcmp(status, "ok"))    (2)
                        return 1;
        }

        return 0;                   (3)
}
```

(1)

读取"status" property，如果没定义"status" property，则该device node有效

(2)

如果device node定义了"status" property，且property value为"okay" or "ok"，则该device node有效

(3)

定义了"status" property，且不等于"okay" or "ok"，该device node无效