touch screen driver (edt-ft5x06.c)

```
module_i2c_driver(edt_ft5x06_ts_driver);

==>

#define module_i2c_driver(__i2c_driver) \
        module_driver(__i2c_driver, i2c_add_driver, \
                        i2c_del_driver)

module_driver(edt_ft5x06_ts_driver, i2c_add_driver, i2c_del_driver);

==>
#define module_driver(__driver, __register, __unregister, ...) \
static int __init __driver##_init(void) \
{ \
        return __register(&(__driver) , ##__VA_ARGS__); \
} \
module_init(__driver##_init); \
static void __exit __driver##_exit(void) \
{ \
        __unregister(&(__driver) , ##__VA_ARGS__); \
} \
module_exit(__driver##_exit);
```

```
static int __init edt_ft5x06_ts_driver_init(void)
{
    return i2c_add_driver(&edt_ft5x06_ts_driver);
}
module_init(edt_ft5x06_ts_driver_init);

static void __exit edt_ft5x06_ts_driver_exit(void)
{
    i2c_del_driver(&edt_ft5x06_ts_driver);
}
module_exit(edt_ft5x06_ts_driver_exit);
```

==========

```
/* Each module must use one module_init(). */
#define module_init(initfn)                                     \
        static inline initcall_t __inittest(void)               \
        { return initfn; }                                      \
        int init_module(void) __attribute__((alias(#initfn)));

/* This is only required if you want to be unloadable. */
#define module_exit(exitfn)                                     \
        static inline exitcall_t __exittest(void)               \
        { return exitfn; }                                      \
        void cleanup_module(void) __attribute__((alias(#exitfn)));
```

==========

module_init(edt_ft5x06_ts_driver_init);

==>

static inline initcall_t __inittest(void)

{ return edt_ft5x06_ts_driver_init; }

int init_module(void) __attribute__((alias(edt_ft5x06_ts_driver_init)));

The alias attribute causes the declaration to be emitted as an alias for another symbol, which must be specified.

即使得init_module是edt_ft5x06_ts_driver_init的一个alias.

init_module() equal edt_ft5x06_ts_driver_init().

上面是如果edt-ft5x06 driver被build成module的情况下。

in include/linux/init.h

The driver is not a module currently.

#define module_init(x) __initcall(x);

即__initcall(edt_ft5x06_ts_driver_init);

#define __initcall(fn) device_initcall(fn)

==> device_initcall(edt_ft5x06_ts_driver_init);

```
#define device_initcall(fn)             __define_initcall(fn, 6)
```

```
==> __define_initcall(edt_ft5x06_ts_driver_init, 6) ;
```

```
#define __define_initcall(fn, id) \
        static initcall_t __initcall_##fn##id __used \
        __attribute__((__section__(".initcall" #id ".init"))) = fn; \
        LTO_REFERENCE_INITCALL(__initcall_##fn##id)
```

```
==> static initcall_t __initcall_edt_ft5x06_ts_driver_init6 __used
    __attribute__((__section__(".initcall6.init"))) = edt_ft5x06_ts_driver_init;
    static __used __exit void *reference_initcall_edt_ft5x06_ts_driver_init6(void)
    {
        return &initcall_edt_ft5x06_ts_driver_init6;
    }
```

CONFIG_TOUCHSCREEN_EDT_FT5X06=y

所以edt_ft5x06_ts_driver_init()应该在kernel初始化时被调用。

driver的初始化相对比较晚。

```
asmlinkage __visible void __init start_kernel(void)
{
    .....
```

```
        rest_init();
}

static noinline void __init_refok rest_init(void)
{
    ......
    kernel_thread(kernel_init, NULL, CLONE_FS);
    ......
}

static int __ref kernel_init(void *unused)
{
    kernel_init_freeable();
    ......
}

static noinline void __init kernel_init_freeable(void)
{
    ......
    do_basic_setup();
    ......
}

/*
 * Ok, the machine is now initialized. None of the devices
```

```
 * have been touched yet, but the CPU subsystem is up and
 * running, and memory and process management works.
 *
 * Now we can finally start doing some real work..
 */
static void __init do_basic_setup(void)
{
        cpuset_init_smp();
        usermodehelper_init();
        shmem_init();
        driver_init();              初始化driver framework
        init_irq_proc();
        do_ctors();
        usermodehelper_enable();
        do_initcalls();          真正对各个内置的driver进行初始化是在这里
        random_int_secret_init();
}
```

driver的初始化是在memory and process management都OK以后开始的。

```
static void __init do_initcalls(void)
{
        int level;

        for (level = 0; level < ARRAY_SIZE(initcall_levels) - 1; level++)
```

```
        do_initcall_level(level);
}
```

初始化时要执行的是如下function

```
static int __init edt_ft5x06_ts_driver_init(void)
{
   return i2c_add_driver(&edt_ft5x06_ts_driver);
}

static struct i2c_driver edt_ft5x06_ts_driver = {
      .driver = {
            .owner = THIS_MODULE,
            .name = "edt_ft5x06",
            .of_match_table = of_match_ptr(edt_ft5x06_of_match),
            .pm = &edt_ft5x06_ts_pm_ops,
      },
      .id_table = edt_ft5x06_ts_id,
      .probe    = edt_ft5x06_ts_probe,
      .remove   = edt_ft5x06_ts_remove,
};

#define i2c_add_driver(driver) \
      i2c_register_driver(THIS_MODULE, driver)
```

即在drivers/i2c/i2c-core.c中的i2c_register_driver()中加入debug info,确认是否edt-ft5x06 driver是否register了。

```
==================================================
      printk("%s-%d\n", __func__, __LINE__);
```

```
        printk("%s: driver - %s\n", __func__, driver->name);
```

==========

log show the following message

```
i2c_register_driver-1828
i2c_register_driver: driver - edt_ft5x06
i2c_register_driver-1860
```

edt_ft5x06 driver被注册了！

难道edt_ft5x06 device node没有创建？

```
i2c_device_match-463-edt,edt-ft5x06
i2c_device_match-474: driver = edt_ft5x06
CPU: 0 PID: 1252 Comm: udevd Tainted: G           O   3.18.7-yocto-standard #28
[<c0016314>] (unwind_backtrace) from [<c00120b0>] (show_stack+0x10/0x14)
[<c00120b0>] (show_stack) from [<c044f0f0>] (dump_stack+0x80/0xc0)
[<c044f0f0>] (dump_stack) from [<c032c080>] (i2c_device_match+0xb8/0x140)
[<c032c080>] (i2c_device_match) from [<c0281888>] (__device_attach+0x28/0x44)
[<c0281888>] (__device_attach) from [<c027fa80>] (bus_for_each_drv+0x58/0x8c)
[<c027fa80>] (bus_for_each_drv) from [<c0281474>] (device_attach+0x78/0x80)
[<c0281474>] (device_attach) from [<c02809fc>] (bus_probe_device+0x84/0xa8)
[<c02809fc>] (bus_probe_device) from [<c027eda4>] (device_add+0x454/0x570)
[<c027eda4>] (device_add) from [<c032c41c>] (i2c_new_device+0x12c/0x1bc)
```

[<c032c41c>] (i2c_new_device) from [<c032ceb0>] (i2c_register_adapter+0x2dc/0x504)

[<c032ceb0>] (i2c_register_adapter) from [<bf0096b0>] (i2c_pxa_probe+0x3b4/0x4c0 [i2c_pxa])

[<bf0096b0>] (i2c_pxa_probe [i2c_pxa]) from [<c0283064>] (platform_drv_probe+0x44/0xa4)

[<c0283064>] (platform_drv_probe) from [<c0281604>] (driver_probe_device+0x138/0x394)

[<c0281604>] (driver_probe_device) from [<c0281930>] (__driver_attach+0x8c/0x90)

[<c0281930>] (__driver_attach) from [<c027f9e0>] (bus_for_each_dev+0x60/0x94)

[<c027f9e0>] (bus_for_each_dev) from [<c0280cc4>] (bus_add_driver+0x15c/0x218)

[<c0280cc4>] (bus_add_driver) from [<c0281f40>] (driver_register+0x78/0xf8)

[<c0281f40>] (driver_register) from [<c0008adc>] (do_one_initcall+0xb8/0x1e0)

[<c0008adc>] (do_one_initcall) from [<c007aa60>] (load_module+0xe7c/0xfcc)

[<c007aa60>] (load_module) from [<c007ad40>] (SyS_finit_module+0x80/0xb0)

[<c007ad40>] (SyS_finit_module) from [<c000f220>] (ret_fast_syscall+0x0/0x30)

i2c_device_match-484

对应

474 printk("%s-%d: driver = %s\n", __func__, __LINE__, drv->name);

476dump_stack();

```
    if (of_driver_match_device(dev, drv))
    {
480         printk("%s-%d\n", __func__, __LINE__);
            return 1;
    }
```

484 printk("%s-%d\n", __func__, __LINE__);

edt_ft5x06没有match!!!

但非常奇怪的是有如下log

i2c_device_match-484

edt_ft5x06 3-0038: touchscreen probe failed

edt_ft5x06: probe of 3-0038 failed with error -121

发现原来code的一个问题：

在dts中touch screen device的描述如下

```
polytouch: edt-ft5x06@38 {
    compatible = "edt,edt-ft5x06";
    reg = <0x38>;
    pinctrl-names = "default";
    interrupt-parent = <&gpio0>;
    interrupts = <92 0>;
    num-x = <1024>;
    num-y = <600>;
    invert-y = <1>;
```

```
                invert-x = <0>;
                reset-gpios = <&gpio0 93 0>;
```

但在edt-ft5x06.c中

```
static const struct i2c_device_id edt_ft5x06_ts_id[] = {
        { "edt-ft5x06", 0, },
        { /* sentinel */ }
};
MODULE_DEVICE_TABLE(i2c, edt_ft5x06_ts_id);

#ifdef CONFIG_OF
static const struct of_device_id edt_ft5x06_of_match[] = {
        { .compatible = "edt,edt-ft5206", },
        { .compatible = "edt,edt-ft5306", },
        { .compatible =       "edt,edt-ft5406", },
        { /* sentinel */ }
};
MODULE_DEVICE_TABLE(of, edt_ft5x06_of_match);
#endif

static struct i2c_driver edt_ft5x06_ts_driver = {
        .driver = {
                .owner = THIS_MODULE,
                .name = "edt_ft5x06",
```

```
        .of_match_table = of_match_ptr(edt_ft5x06_of_match),
        .pm = &edt_ft5x06_ts_pm_ops,
    },
    .id_table = edt_ft5x06_ts_id,
    .probe    = edt_ft5x06_ts_probe,
    .remove   = edt_ft5x06_ts_remove,
};
```

dts中的compatible = "edt,edt-ft5x06"无法与code中的

```
static const struct of_device_id edt_ft5x06_of_match[] = {
    { .compatible = "edt,edt-ft5206", },
    { .compatible = "edt,edt-ft5306", },
    { .compatible =       "edt,edt-ft5406", },
```

比配，但在6270（Granite2）board上touch screen一切正常。

原因是i2c-core.c

```
static int i2c_device_match(struct device *dev, struct device_driver *drv)
{
    struct i2c_client    *client = i2c_verify_client(dev);
    struct i2c_driver    *driver;

    if (!client)
        return 0;
```

```
	/* Attempt an OF style match */
	if (of_driver_match_device(dev, drv))		( 1 )
		return 1;

	/* Then ACPI style match */
	if (acpi_driver_match_device(dev, drv))
		return 1;

	driver = to_i2c_driver(drv);
	/* match on an id table if there is one */
	if (driver->id_table)
		return i2c_match_id(driver->id_table, client) != NULL;	( 2 )

	return 0;
}
```

（1）处的基于device tree的match失败了，但在（2）出legacy方式的match成功了。

client->name与

```
static const struct i2c_device_id edt_ft5x06_ts_id[] = {
	{ "edt-ft5x06", 0, },
	{ /* sentinel */ }
};
```

匹配！！！