烧写速度太慢了

原函数：

ret=request\_thread\_irq(ts->client->irq,null,istcore\_irq\_task,(IRQF\_TRIGGER\_LO｜IRQ\_ONESHOT), ts->client->name, ts);

修改方式一

如果handle与thread用相同的函数（原来那个放在thread里面的函数）：

ret=request\_thread\_irq(ts->client->irq,**istcore\_irq\_task**,istcore\_irq\_task,(IRQF\_TRIGGER\_LOW｜IRQ\_ONESHOT), ts->client->name, ts);

tb启动不起来。

无限循环

start boot options等待

启动到lenovo界面

修改方式二

ret=request\_thread\_irq(ts->client->irq,**my\_handler**,istcore\_irq\_task,(IRQF\_TRIGGER\_LOW｜IRQ\_ONESHOT), ts->client->name, ts);

static irqreturn\_t my\_handler (int irq, void \* handle)

{

Printk(“my\_handler run \n”);

Return IRQ\_WAKE\_THREAD;

}

结果：cat /proc/kmsg 触摸显示printk 内容

尝试将thread的处理函数替换handler，thread存在。结果如1

尝试将thread的处理函数替换handler，并且thread为null。结果如1

接下来试图寻找到底是哪些代码导致启动失败。在thread里面

测试结果是如果两个函数都有对寄存器的操作，则平板无法开机。结果如1

进一步测试表明只要在handler里面放入对寄存器的操作，都会无法开机。不管thread里面写的什么

尝试在handler里面放入延时函数msleep，开机失败，***卡死在len’ovo界面***

执行在中断上下文中的代码需要注意的一些事项：

* 中断上下文中的代码不能进入休眠。
* 不能使用mutex，只能使用自旋锁，且仅当必须时。
* 中断处理函数不能直接与用户空间进行数据交换。
* 中断处理程序应该尽快结束。
* 中断处理程序不需要是可重入的，因为相同的中断处理函数不能同时在多个处理器上运行。
* 中断处理程序可能被一个优先级更高的中断处理程序所中断。为了避免这种情况，可以要求内核将中断处理程序标记为一个快速中断处理程序（将本地CPU上的所有中断禁用），不过在采取这个动作前要慎重考虑对系统的影响

利用脚本模拟的触摸事件在getevent以及cat /proc/kmsg里面没有任何显示

搜索全文查找关于request\_threaded\_irq的例子

Out/target/product/r2\_cht\_ffd/obj/kernel/source/virt/kvm/assigned-dev.c：350

这里面的handler写的比较简单。用来spin\_lock去，没有复杂的寄存器读取

Kernel/cht/drivers/staging/iio/accel/lis3l02dq\_core.c :724

这里面的handler写的简单。调用函数 iio\_trigger\_poll

这里面的handler在.h文件里面define了另外一个名字。在另外一个文件里面进行的定义，实现

Kernel/cht/drivers/staging/imx-drm/imx-hdmi.c:1632

这个关于hdmi的中断的handler里面写了对硬件的读取 hdmi\_readb()函数的使用。Hdmi\_writeb函数的使用；**但是我没有找到最终的调用**

Kernel/cht/drivers/i2c/busses/i2c-omap.c:1228

这里面的handler调用了 omap\_i2c\_read\_reg **基本确定是可以在handler里面对寄存器进行操作的**

分析下来调用内核提供函数

原子操作

并且互斥锁不能用于中断上下文

感觉是互斥锁的锅

准备尝试将thread里面的处理更加细化。分割

仔细查看可以得出问题。互斥锁太多了。不能在中断上下文中执行

发现platform\_ist940e.c中没有写i2c-put\_adapter 加上后正常使用tablet。不明觉利

一个驱动分两部分

一类一旦开机就无法测试，

一类可以在整个程序运行的时候测试

关于输出的错误序列号 查看kernel/../include/uapi/asm-generic/errno.h