Linux 兼容多型号触摸屏的实现

Author：李东汉
Email：ldh@rock-chips.com

一、需求与目的

在我们进行产品开发维护过程中，经常碰到客户有这样的需求：因为硬件市场行情或定位的变化，需要不时更换产品硬件，如触摸屏，LCD 等。此时需要我们重新添加或更改驱动发布固件以支持新的硬件，但支持新的硬件的固件在前面旧的硬件机器上就不能再使用，造成产品固件版本的混乱和管理的复杂，为了解决这一问题，我们需要做这样一个事情：在发布支持新的硬件固件时，新的固件同时可以向下兼容旧的硬件的机器，简言之：一个固件可以在多个不同硬件的机器上运行。

二、预备理论知识

其实只要理解了 Linux 设备驱动的架构，实现上面的需求是一个很简单的事情，只需要修改少量的驱动代码就可以，下面介绍一下 Linux 设备驱动相关的一些知识（节选自 CSDN）：

设备与驱动需要挂载在总线上，需要指明驱动与设备是属于哪条总线的，所以设备与驱动需要注册。而总线在 linux 系统中也是属于设备，所以总线也要注册，同时要先有总线而后才能注册设备和驱动，所以总线要先注册。

总 线在 linux 系统中有俩种，一是实际存在的总线 pci usb 等等，还有一类是虚拟存在的总线 platform ，platform 总线主要是用于集成在 SoC 系统的设备，使得每一个设备都属于一条总线，相应的设备称为 platform_device，而驱动成为 platform_driver。如我们平常

见的 I2C 总线，I2S 总线等，linux 驱动中 platform 总线用的非常多，以 platform 总线说明总线，设备，驱动的注册顺序，注意这里是以先调加设备为例。

通过以下三步，即可以将总线、设备、驱动关联起来

1）platform bus 先被 kenrel 注册。这一步我们基本上不需要做任何修改。

2）系统初始化过程中调用 platform_add_devices 或者 platform_device_register ，将平台设备(platform devices) 注册到平台总线中(platform_bus_type)

这一步我们就能经常接触到，给内核添加新设备硬件时都必须添加新的硬件信息，至于怎么注册，步骤等，我们一般不需要管。如添加触摸屏 GT801 的设备信息,只需要在数组 board_i2c2_devices 中添加

```
#if defined(CONFIG_CT360_TS)
{
    .type       ="ct360_ts",                    //设备名称
    .addr           =0x01,                      //i2c 从地址
    .flags          =0,                         //i2c 标识
    .irq            = TOUCH_INT_PIN,            //设备中断 pin
    .platform_data = &ct360_info,               //附加的一些信息
},
#endif
```

即可。

3）平台驱动 (platform driver) 与平台设备 (platform device) 的关联是在 platform_driver_register 或者 driver_register 中实现，一般这个函数在驱动的初始化过程调用。

我们知道，设备的驱动程序是在系统运行起来的时候静态加载上去的，只要将驱动程序编进系统内核中，不管其有没有对应的设备，都会一一地加载，然后会试着和总线上的设备信息去匹配，如果加载的驱动程序和总线上的设备信息匹配，就会去调用该驱动程序的相关初始化接口，注册设备文件节点，分配 IO 内存空间，注册中断等，但是不是完成了以上这些动作之后，设备就能正常运行了呢？答案是：不一定。一会下面详细解释。

三、功能实现方法与步骤

由上面的预备理论知识可知，我们要实现的其实就是以下三个步骤，以项目 N12为例，兼容三个触摸屏的驱动 GT801、FT5306、CT360：

1.添加设备

如上面的第二小点，将以上三个设备的信息添加到 board_i2c2_devices 中

```
static struct i2c_board_info __initdata board_i2c2_devices[] = {
  ……
#if defined (CONFIG_TOUCHSCREEN_GT801_IIC)
    {
        .type           = "gt801_ts",
```

```
        .addr           = 0x55,
        .flags          = 0,
        .irq            = TOUCH_INT_PIN,
        .platform_data = &gt801_info,
    },
#endif
#if defined(CONFIG_TOUCHSCREEN_FT5306_IIC)
{
    .type           = "ft5x0x_ts",
    .addr           = 0x38, // 0x70>>1,
    .flags          = 0,
    .irq            = RK29_PIN0_PA2,
    .platform_data  = &FT5306_info,
},
#endif
#if defined(CONFIG_CT360_TS)
{
    .type       ="ct360_ts",
    .addr       =0x01,
    .flags      =0,
    .irq        = TOUCH_INT_PIN,
    .platform_data = &ct360_info,
},
#endif
……
};
```

2.编译驱动

这个很简单，只要我们保证每个单独的触摸屏驱动都能正常工作，直接在 make menuconfig--->Devices Driver--->Input device support------>Touchscreen 中选中 GT801，FT5306,CT360 即可，此处可能有个问题，就是三个驱动中使用的变量名称有冲突，如果这样，直接修改变量名字就可以了，然后编译通过。

3.隔离驱动

这里使用了“隔离”这词不知是否正确，顾名思义，就是让三个驱动程序互不形成干扰。刚才在第二大点中说当驱动和设备信息匹配后，然后创建节点，申请内存，注册中断成功之后设备可能还是不能正常运行，是因为即使设备（硬件）不存在，当我们完成上面两个步骤添加设备，编译驱动后，设备的信息已经存在，当驱动加载后，是靠这些信息（一般是 name）去匹配设备的，而这个过程，完全无视硬件的存在与否，直到试图和硬件进行通信，才会出现错误。而此时，可能已经完成创建节点，申请内存，注册中断等动作了。而创建的节点可能会带来影响。

在我们系统中，一般 input0 节点对应 keyboard，input1 节点对应 Touchscreen，input2 节点对应 gsensor，上层对相应输入设备的操作其实就是对以上对应的设备节点进行操作，如 read，write 等。如果三个触摸屏的驱动都加载而只有后面加载的两个其中之一正好匹配且设备存在，那第一个驱动加载时虽然设备不存在，但由于信息匹配而创建了 input1 节点，那到真正匹配的驱动运行时，可能创建的节点名就变成了 input2，input3 了，这就不仅是影响了触摸屏，而且还影响了 gsensor 的正常工作。所以这里所谓的隔离驱动，就是在设备驱动初始化的 XX_probe 中，在创建设备节点之前，添加测试硬件设备是否存在的代码，如果测试通过，说明设备存在，反之不存在，可以直接跳出初始化。

在这里我们使用 i2c 的通信是否成功来判断，增加代码如下：

```
ret = i2c_master_reg8_recv(client, 0x00, buf, 2, FT5406_IIC_SPEED);
if(ret<0){
    printk("%s:i2c_transfer fail =%d\n",__FUNCTION__,ret);
    goto err_other;
}
```

到这里，实现的功能一般就可以使用了。

四、小结

严格来说这不能说是实现新功能，因为本身强大的 Linux 已经完成，也许有人会问那为什么 Linux 不把已知的所有硬件驱动一股脑全部编译添加进内核呢？这个我想大家都很清楚，如果将所有已知硬件的驱动都编译进内核，那编译出来的内核固件肯定大得伤不起。此文目的是想和大家交流一下 Linux 设备驱动的一些原理和思想，可能会存在很多不妥或措辞不当之处，希望大家指出，一起进步。

参考网站：

1.http://blog.chinaunix.net/space.php?uid=7859076&do=blog&id=2552439
2.http://hi.baidu.com/serial_story/blog/item/d40d49c6907212129c163ded.html
3.http://blog.csdn.net/yimu13/article/details/6762964