

介绍了基于飞思卡尔芯片 i.MX27 和嵌入式 linux 系统下的触摸屏硬件的连接设计和软件的驱动设计,并依照此设计实现了触摸屏的从硬件到软件的驱动系统。该实现主要应用于家庭无线智能控制系统中。

文章首先介绍了触摸屏的实现原理,然后介绍了触摸屏芯片 AD7873 的特性,在此基础上设计了 ad7873 与 i.MX27 和触摸屏的连接 PCB 图,最后依照硬件连接图设计了嵌入式 Linux 下的驱动,并成功通过了 tclib 触摸屏专业测试软件的测试,在家庭智能网关系统的测试中也成功运行,实现了从硬件到软件的嵌入式下触摸屏的驱动系统设计。

1 引言

随着计算机技术的发展和普及,触摸屏技术得到了越来越广泛应用,在各种手持设备中,如手机、MP4、掌上游戏机、掌上 PDA 等,由于其方便、舒适,使其完全摆脱了键盘和鼠标的束缚,使人机交互更为直截了当。而在微软最新开发的 windows7 操作系统中,就有其值得骄傲并加以推广的多点触摸技术,并成为一大卖点。可见,触摸屏技术引起了上到微软,下到普通老百姓的关注。而在我们的日常生活中,无论你是在商场购物,还是在银行存取款,触摸式的自动服务器将能为你提供方便快捷的服务。这里通过对触摸屏原理的理解和分析,成功的设计出了 CPU 与触摸屏芯片之间的硬件连接,并依照硬件和驱动设计的原理,设计出了基于嵌入式 Linux 和飞思卡尔 i.MX27 芯片以及 AD7873 触摸屏芯片的驱动程序,并成功移植到内核中,实现了家庭控制器系统的触摸技术。

2 硬件系统的构成

2.1 电阻式触摸屏原理。

电阻式触摸屏是一种传感器,它将矩形区域中触摸点 (X, Y) 的物理位置转换为代表 X 坐标和 Y 坐标的电压。当触摸屏表面受到的压力 (如通过笔尖或手指进行按压) 足够大时,顶层与底层之间会产生接触。所有的电阻式触摸屏都采用分压器原理来产生代表 X 坐标和 Y 坐标的电压。如图1所示,分压器是通过将两个电阻进行串联来实现的。上面的电阻 (R1) 连接正参考电压 (VREF), 下面的电阻 (R2) 接地。两个电阻连接点处的电压测量值与下面那个电阻的阻值成正比。为了在电阻式触摸屏上的特定方向测量一个坐标,需要对一个阻性层进行偏置: 将它的一边接 VREF, 另一边接地。

同时,将未偏置的那一层连接到一个 ADC 的高阻抗输入端。当触摸屏上的压力足够大,使两层之间发生接触时,电阻性表面被分隔为两个电阻。它们的阻值与触摸点到偏置边缘的距离成正比。触摸点与接地边缘之间的电阻相当于分压器中下面的那个电阻。因此,在未偏置层上测得的电压与触摸点到接地边之间的距离成正比。

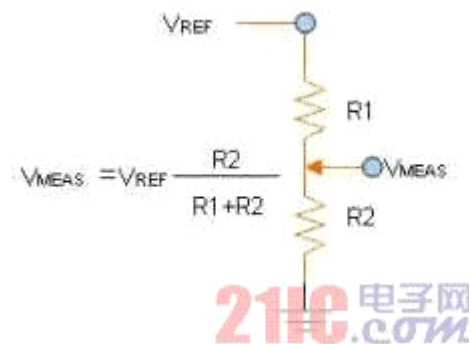


图1原理示意图

2.2 AD7873 介绍及与系统硬件原理

AD7873 是一款 12 位逐次逼近型 ADC, 具有同步串行接口以及用于驱动触摸屏的低导通电

阻开关，采用2.2V至5.25V单电源供电，吞吐量大于125KBPS。

AD7873可用于电池测量、温度测量和触摸压力测量，还具有一个2.5V片上基准电压源，可用于辅助输入、电池监控器和温度测量等模式。不使用时，可关断内部基准电压源以降低功耗。也可以使用外部基准电压，并可在1V至VCC范围内变化，模拟输入范围为0V至VREF。这款器件具有关断模式，此模式下功耗不足1μA。

2.3 AD7873与CPU和触摸屏的硬件连接图

其硬件原理框图如下，其中的X+、Y+、X-、Y-与触摸屏的相应引脚相连，接受来自触摸屏的模拟信号，然后经过AD7873芯片的内部处理成数字信号，通过SPI总线将数据传送给CPU，请求处理。CS为片选引脚，与CPU的DTR_DCE1相连，PENIRQ为中断引脚，接CPU的GPIO1_0。

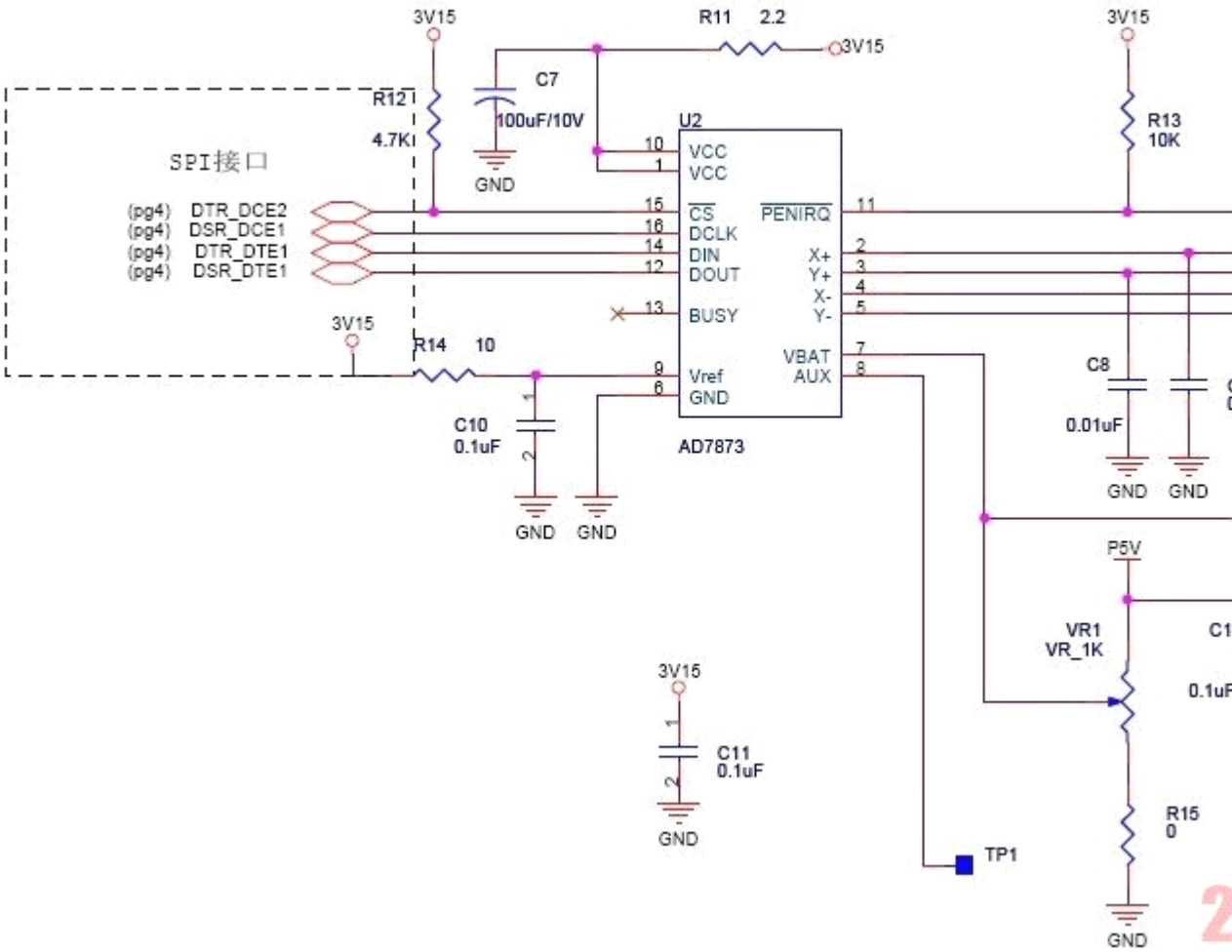


图2硬件连线图

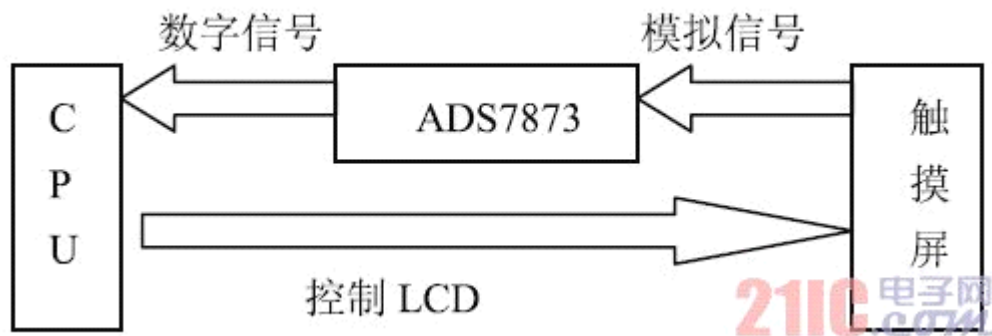


图3触摸系统框图

3软件系统

3.1Linux 设备驱动介绍

在Linux系统中，为了简化对设备的管理，所有的外围设备被归结为3类：字符设备、块设备、网络设备。Linux对所有的物理设备进行了抽象，并定义了一个统一的概念：接口。AD7873被定义为一个字符设备，采用spi接口与CPU通讯。

3.2驱动部分重要函数的设计

(1) 设备驱动程序中数据结构 struct driver 定义了一系列函数操作的接口，这个数据结构将整个驱动连为一体，由这个结构可以看出整个驱动的脉络。由于AD7873与CPU连接方式为SPI总线连接，因此将此设备注册为SPI设备，即要用到数据结构 struct spi_driver。

对应于AD7873设备，设计编写的数据结构如下：

```
static struct spi_driver ad7873_driver = {  
    .driver = {  
        .name = "ad7873",  
        .bus = &spi_bus_type,  
        .owner = THIS_MODULE,  
    },  
    .probe = ad7873_probe,  
    .remove = __devexit_p(ad7873_  
remove),  
    .suspend = ad7873_suspend,  
    .resume = ad7873_resume,  
};
```

其中，driver中定义了驱动名称、总线类型和驱动所有者。

probe函数为探测设备的函数。其主要进行初始化设备数据结构、初始化中断、向设备发送控制字等。

remove函数为设备移除函数。其主要进行移除设备文件、释放中断、释放设备。

suspend 为设备暂停函数。

resume 为设备恢复函数。

(2) 设备初始化函数 `static int __init ads7873_init (void)` 即是注册上述数据结构，也是注册一个驱动，其中主要的内容为：

`return spi_register_driver (&ads7873_driver);` (3) 还要设计一个重要的函数，就是中断函数，在触摸屏被按下的时候产生中断，在中断函数中的重要工作就是启动定时器，以判断触摸后的动作是触摸笔被提起还是继续按下。其函数原型设计为

`static irqreturn_t ads7873_irq (int irq, void *handle);` (4) 定时器函数设计的目的就是判断触摸屏是否被提起，被提起就立刻刷新设备的数据到应用层，仍然处于“按下”状态，则需要继续测量。其函数原型设计为：

`static void ads7873_timer (unsigned long handle);`

3.3 驱动的编译和加载

驱动程序加入内核有两种方式动态和静态，动态就是只是把驱动编译为模块，系统启动后执行 `insmod` 后加载，静态是指直接编译进内核，系统启动后自动加载了。由于我们的驱动需要经过测试才能加入内核，因此采用动态的方式加载驱动。

在已经安装好开发平台的主机上执行 `make`，就可以得到目标文件 `ads7873.ko`，这就是我们需要的驱动。在 `platform_data` 数据结构中设置好驱动需要的数据，如中断、触摸屏大小、触摸压力上下限等数值，并编译好内核下载到开发板中运行。然后将 `ads7873.ko` 复制到开发板，在终端下执行 `insmod ads7873.ko`，得到如下提示：



```
mx27# insmod ads7873.ko
ads7873 spi.0: touchscreen, irq 126
This sentence has been <6>input: ADS7873 Touchscreen as /class/input/input2
mx27#
```

图4驱动加载信息

显示驱动已经加载成功，中断号为126。

(注：“this sentence has been”，为调试驱动中断的测试语句) 然后用应用广泛的 `tslib` 触摸屏测试程序对驱动进行测试，测试结果显示，触摸屏工作正常。下图触摸屏上白色的字“桂林电子科技大学触摸屏”为在 `tslib` 的测试程序下用触摸笔写的字，证明了触摸屏工作是正常的。

将此设计应用到家庭智能网关系统中，证明其工作稳定，响应很快。



图5触摸屏正常响应图

4 驱动程序出现的问题及解决方法

在调试驱动的过程中，出现了很多问题，主要有以下两个：

4.1 加载时报错

在执行 `insmod ad7873.ko` 时，报错为：

```
ad7873:noversionfor"struct_module"found:kerneltainted.
```

经过检查，发现是因为 `platform_data` 里面定义的 `model="7873"` 被写成了。

`model="7883"`，造成了系统无法识别这个驱动，产生了错误，修改后执行 `insmod ad7873.ko` 成功。

4.2 使用 `tslib` 测试软件测试的时候，屏幕无反应

原以为是中断程序出了问题，因此在中断里面设置打印信息，发现可以打印，于是确定了中断时正常响应了的。然后跟踪代码检查，发现定时器函数很可疑，于是在定时器函数设置打印信息，发现定时器函数没有正确执行，找到了问题的所在。于是顺藤摸瓜，终于发现原来是在这个函数中 `fn_get_pendown_state(void)`，也就是用来得到触摸屏状态的引脚设置不正确，导致定时器函数里面无法得到触摸屏的状态，所以无响应。修改了定时器的引脚定义使其指向正确的引脚便一切 OK。

5 结论

本文从硬件到软件设计了整个触摸屏驱动系统。作为家庭无线智能控制器的一部分，触摸屏使整个系统使用更方便与高效。而未来随着对触摸技术的越来越高的要求，如多点触摸技术，需要驱动为上层应用提供更稳定的处理与更多的接口，以便上层应用可以更加高效的编写应用程序来应对复杂的机制。这就需要在内核驱动层改进驱动的结构与算法来应对。