触摸屏学习及触控程序开发

为标准触摸屏编写驱动程序

尽管触摸屏正在迅速普及开来，但大多数开发人员以前从来没有开发过触摸屏产品。本文详细介绍了触摸屏产品的设计步骤，指导读者了解使触摸屏首次工作需要的软硬件细节。

触摸屏如今随处可见。工业控制系统、消费电子产品，甚至医疗设备上很多都装备了触摸屏输入装置。我们平时不经意间都会用到触摸屏。在 ATM 机上取款、签署包裹，办理登机手续或查找电话号码时都可能会用到触摸屏。

本文介绍了二种较新的 CPU，它们都内建了对触摸屏输入的支持。本文将介绍如何编写软件驱动程序，从而能够使用这些微处理器配置、校准触摸屏以及对触摸屏输入持续响应。最终将提供可免费下载和使用的工作代码，作为读者进一步设计的基础。

触摸屏作为输入手段的优点和缺点

没有一种输入方式是十全十美的，对某些特定的应用和产品类型来说，触摸屏不是最好的输入手段。为了让读者清楚的了解触摸屏的特性，下面先概括使用触摸屏作为输入手段的优点和缺点。

首先是优点：触摸屏不可否认的具有酷的感觉，立刻就能使产品的使用变得更有乐趣。同时触摸屏也非常直观。当用户想要选择 A 选项时，他伸出手指碰一下 A 选项就可以了。这还不够直观吗？连两岁的婴儿都知道怎样伸手去触摸他(或她)想要的东西。最后要说的是，触摸屏作为输入装置和系统固定在了一起。如果用户忘记遥控器或鼠标放的位置，就会无法进行输入。而如果具有触摸屏的设备放在用户前面，用户马上就可以用触摸屏进行输入。

再说缺点，触摸屏可能会在不合适的场合下被错误的使用。这里我是指对安全性要求严格的设备，对于这些设备，如果没有适当的预防措施，使用触摸屏会非常危险。下面我将概括一些最明显的潜在的问题，如果读者想作更进一步的了解，可以参考更多的资料。

第一个问题是视差，即屏幕上看到的对象的位置与其在触摸面板上的实际有效位置之间的差异。图 1 说明了这个问题。我能想到的最佳例子是典型的“免下车”ATM 机。这种 ATM 机不会根据汽车的高度升高或降低自己的高度，因此如果你坐在较高的 SUV 或卡车里，那么你就会从抬高的位置俯视显示屏。为了保护昂贵的显示器件免受恶意破坏，ATM 机都会在用户和显示屏之间放置几层强化玻璃。

触摸屏是不能这样保护的。如果真这样做的话，用户就无法进行触摸了。因此触摸屏放在表层上，而显示屏放在表层下的几层玻璃后面。这就造成了触摸层和显示层之间的物理隔离。如果用户以某个角度观看屏幕，就意味着用户按压触摸屏进行选择的位置会与用户接口软件预期的输入位置之间存在一定的距离偏差。

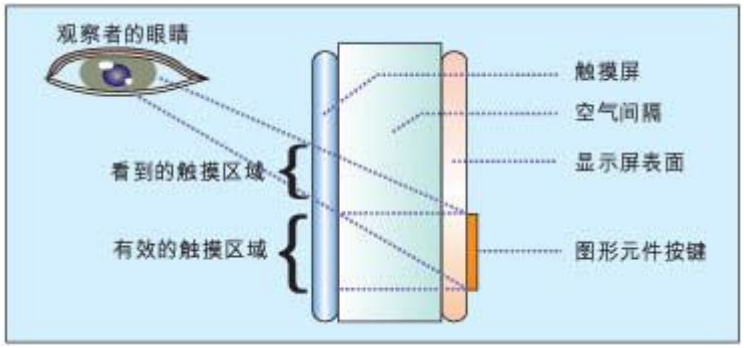

图 1：视差(横截面图)。

人们能很快适应这种偏差。经过几次尝试和错误，使用者学习在触摸屏的表面找到显示信息的映射位置，然后触摸到正确的位置。ATM 设计师也认识到这一点，他们会采用大面积的按键，并尽量使它们相互远离，因此有助于防止错误按键的误触发。当然，不小心按下错误的 ATM 按键不会使你得癌症或使你失明。但如果这样的失误发生在医疗控制设备上，并且系统设计师没有在系统内置足够的安全预防措施，那么以上两种后果确实都有可能发生。通过缩短显示层和触摸层之间的物理距离可以尽量减少视差。在 CRT 或 LCD 前面总会有玻璃存在。最好的方法是将对触摸敏感的电子元件嵌入到玻璃里，并且这层玻璃做得尽可能薄。这样就减少了触摸输入层和显示层之间的相隔距离。像 Palm 这样的手持设备就可以采用这样的策略，因为它们不必太担心机械强度不够或者遭受恶意破坏。随着相隔距离的缩小(用户觉得真的触摸到了图形元件)，精度会大大提高。

第二个明显的问题是，在用户触摸屏幕的过程中，触摸屏幕的物体(触控笔、手指)至少会遮挡屏幕上的一小部分面积，从而影响用户的观察。在工厂自动化应用中这种情况更容易发生，因为用户很可能使用手指或手套而非触控笔，即使是使用触控笔，在屏幕上做选择动作也会不时遮挡住一部分你给用户展示的信息。例如，想象一下你想展示一个滑动控制条给用户调节数值（如速度或音量），并且你将用户选择的数值以数字形式显示在滑动控制条的左边。这样做一般工作情况会很好，但当左撇子用户操作你的系统时，只有移开他的手指他才能看到所选的值。因此你必须在你的用户界面设计中考虑这类因素。

触摸屏硬件原理简介

我们在开始编写触摸屏驱动程序之前，必须对硬件的工作原理有个基本的了解。许多不同的触摸技术会把屏幕某个位置的压力或接触转换成有意义的数字坐标。典型的触摸技术包括电阻触摸屏、声表面波触摸屏、红外线触摸屏和电容触摸屏。

这里侧重介绍电阻触摸屏。电阻触摸屏非常普及，你会发现许多评估板和开发套件中都集成了电阻触摸屏。电阻触摸屏普及的主要原因是价格便宜，而且在电气上可以直接接入用户的系统中。

之所以叫电阻触摸屏，是因为它们本质上就是电阻分压器。它们由两个电阻薄层组成，这两个薄层被非常薄的绝缘层隔开，绝缘层通常以塑料微粒子的形式存在。当你触摸屏幕时，会

使两个电阻薄层变形到足以使它们之间发生电气连接。然后由软件通过检测分压器上产生的电压计算出两层的短接位置，并最终确定触摸位置。

电阻触摸屏分为几种类型，比如"四线"，"五线"和"八线"。线越多，精度就越高，温度漂移也越少，但基本的操作是一样的。在最简单的四线设计中，有一层称为"X 轴"的电阻层，上面加有一定的电压，另一个称为"Y 轴"的电阻层作为接受层测量对应 X 轴位置的电压值。这一过程再反过来执行一遍，即 Y 轴层加电，X 轴层用于电压检测。

图 2 是电阻触摸屏的简单等效电路。注意必须获取二个完全独立的读数，即 X 轴位置和 Y 轴位置数据。这些数据在四线或五线电阻触摸屏中是无法同时读取的。软件必须先读一个轴，然后再读另外一个轴。读取的顺序则无关紧要。将电阻触摸屏产生的电压转换成数字需要用到模数转换器(ADC)。直到不久前这个 ADC 几乎一直是主 CPU 的外围器件。Burr Brown NS7843 或 NS7846 就是这种 ADC 控制器。该器件为 12 位的模数转换器，其内嵌的逻辑电路通过交替给一个薄层加电，再从另外一层转换来控制触摸屏。虽然可以使用诸如 GPIO 之类的信号线来完成薄层加电的切换，但该器件能够分担许多任务，还能提供产生触摸或笔压中断的方式。

最近有几家 CPU 制造商开始在主 CPU 中集成 ADC 模块和专用的触摸屏控制电路。在消费类设备、远程信息通信或一些面向其它市场的产品中，LCD 显示屏和触摸屏非常普遍，当想把 CPU 用于这类产品中时，在 CPU 中集成 ADC 和触摸屏控制电路的做法会非常有意义。

基于两种 CPU 的参考板

本文设计两种集成了触摸屏控制功能的 CPU 的参考板。这二种 CPU 都基于 ARM 处理器架构。

第一块板是飞思卡尔的 MX9823ADS 评估板，采用了飞思卡尔的 MC9328MX1 处理器。该评估板可以直接从飞思卡尔的分销商处定购。评估套件包括 QVGA(240x320)彩色 LCD 和触摸屏。

第二块板采用了夏普 LH79524 ARM 处理器。这块夏普的参考板以及集成的显示和触摸套件都可以从 LogicPD 公司处定购。有几种可更换的显示套件供选择，分辨率范围从 QVGA 到 800x600 像素不等。

本文中不提供每个驱动程序的详细代码，而是介绍驱动程序的设计和流程，并重点介绍其中的重要部分。读者可以从 ftp://ftp.embedded.com/pub/2005/07maxwell 下载每个驱动程序的全部源代码。总的来看，软件提供的功能完成以下这些步骤：

1. 配置控制器硬件
2. 判断屏幕是否被触摸
3. 获得稳定的、去抖动的位置测量数据
4. 校准触摸屏
5. 将触摸状态和位置变化信息发送给更高层的图形软件

下面开始详细介绍每个步骤。

硬件配置

触摸驱动程序要做的第一件事是配置硬件。对这些集成控制器来说，这意味着通过向映射到存储器的寄存器中写入数据将控制器配置成某个确定状态。这一过程是由每个驱动程序中的TouchConfigureHardware()函数完成的。为了配置硬件，需要事先做好某些决定。例如，驱动程序应该使用中断驱动吗？为了获得能够响应并且精确的触摸位置信息需要什么样的转换速率？让我们看看做出这些决定的具体过程吧。

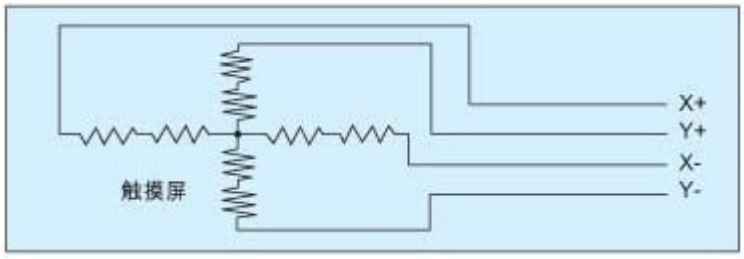

图 2：触摸屏电路简单等效电路。

关于触摸驱动程序是否应该使用中断驱动，事实上在范例的驱动程序中用的就是中断驱动方式。坦率地讲，我之所以这样做是因为使用中断很有趣。千万不要由这个例子推断出采用中断永远是最好或最正确的设计方式，也不要听信别人说不采用中断驱动方式的触摸驱动程序就是"错误的"。之所以这样说只是因为"轮询"对嵌入式系统程序员来说似乎变成了贬义词。我曾经问过一位客户，他的输入设备采用的是轮询还是中断服务方式。回答是"这是嵌入式系统，我们不做任何轮询"。我当时感觉问这个问题时我就像一个傻瓜，但进一步探讨后发现查询其实也是一种合理且值得考虑的方式。如果使用的是 RTOS，并且所有任务经常为了等待某类外部事件而被中断，处理器经常处于空闲的循环状态，没有什么有意义的事做。这种情况下使用空闲任务查询触摸屏上的输入也许是更好的设计方式。根据你的总体系统需求，查询也可能是一个值得考虑的合理的设计方式。

配置中断的方法因具体操作系统而异。读者会发现对于每一个支持的 RTOS 都有被(#ifdef)限定的代码段。在所有情况下驱动程序实际会使用二种不同的中断：
1 当屏幕被初次触摸时唤醒主机的中断，称为 PEN_DOWN 中断
2 当完成一组模数数据转换时的第二种中断信号

后文会详细介绍这些中断和它们产生的过程。

接下来的问题是我们希望以多快的速度从 ADC 接收采样输入读数。采样速度会影响我们需要如何配置时钟来驱动触摸屏和 ADC。我们希望时钟有足够快的速度来提供可响应的输入和实现精确的跟踪，但也不要太快，以至于影响转换精度，或让系统消耗超过所需的功率。根据我的经验，触摸屏至少需要以 20Hz 或 50ms 间隔的速度向更高层软件提供位置更新数据，只要高层软件跟得上，速度越快越好，我们不太担心功耗问题。如果触摸输入响应比这慢得多，那么在用户的触摸输入和显示屏上可观察到的响应之间会出现明显和烦人的迟滞现

象。 20Hz 的更新速度听起来并不是太有挑战性，但提供 20Hz 的更新速度实际上要求采样速度接近 200Hz，具体数值取决于我们在确定输入稳定之前准备采用多少读数。为了去抖动和对触摸输入位置值进行平均，我们需要进行过采样。电阻触摸屏，特别是便宜的那种，一般会有很大的噪声和抖动。在向更高层软件发送位置更新数据之前，驱动程序需要多次采样每个轴上的输入。我们提供的驱动程序默认情况下将以最少 200Hz(5ms)的采样速率配置各自处理器上的 ADC 时钟。这样就能让驱动程序对输入原始数据进行充分的去抖动和过滤，并仍能向高层用户接口软件提供 20Hz 的实际位置更新速率。

飞思卡尔 i.MX 处理器中的触摸控制器模块叫做模拟信号处理器(ASP)，i.MX 处理器提供两个由内核 CPU 时钟分频得到的外设时钟。输入 ASP 模块端口的是 PERCLK2(外设时钟 2)，它经过再分频产生 ASP 所需的最终输入时钟。需要注意的是，PERCLK2 除了驱动 ASP 模块外，还驱动包括内部 LCD 控制器在内的其它子模块，因此触摸驱动程序无法只是为了更好的配合触摸采样而对 PERCLK2 进行编程。PERCLK2 被编程为所有附属外设所要求的最高速率(在大多数情况下是 LCD 控制器)，然后通过分频产生速度较慢的外设所需的时钟。MC9328MX1 参考手册中包含一份表格，该表格定义了达到 200Hz 数据输出速率所需的时钟编程值。

夏普 LH79524 在硬件配置时要求对几个 GPIO 引脚进行编程以便给这些引脚分配 ADC 功能，并要求编程和激活 ADC 时钟，还要对 ADC 序列器编程。LH79524 ADC 本身是一个令人称奇的电路系统，能够实现完全可编程的状态机和序列器。该 ADC 无需核心 CPU 的任何干预就可以通过编程完成：驱动一个触摸层；延时；进行测量；驱动另一层；延时；进行测量等操作。理解如何对 LH79524 ADC 序列器控制单元编程可能是一个挑战，不过利用夏普(www.sharpsma.com)公司提供的应用指南可以使这项工作简单很多。本文提供的驱动程序完全符合该应用指南对如何配置夏普 ADC 序列控制器提出的建议。

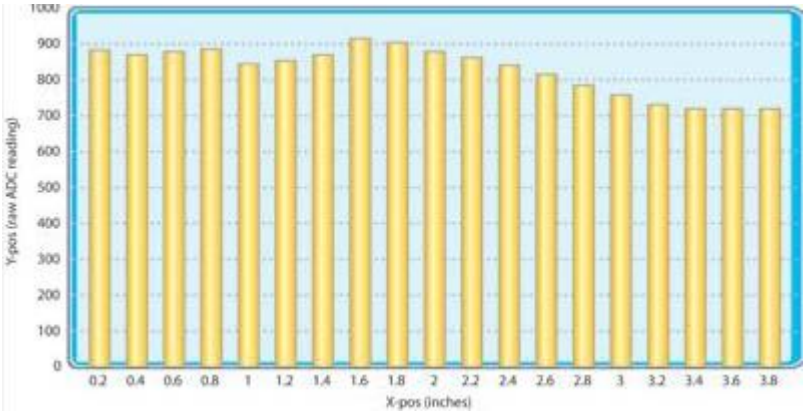

图 3：X 轴移动时 Y 轴上的偏移。

屏幕被触摸到了吗？

一旦完成了基本的硬件设置，接下来就需要一种可靠的方法判断屏幕是否被触摸了。如果用户没有触摸屏幕，那么运行 ADC 获得转换后的读数毫无意义。上述两个控制器都提供了屏幕是否被触摸的检测机制，并且当触摸事件发生时还可选择是否中断主处理器。判断屏幕是否被触摸的驱动程序的函数名叫 WaitForTouchState()。当控制器处于触摸检测模式时，Y

轴触摸层通过一个上拉电阻上拉到高电平，X 轴触摸层则连接到地。当用户触摸屏幕的任何地方时，这两层就发生短接，Y 轴层被拉到低电平。该事件可以在驱动程序内部连接到名为 PEN_OWN IRQ 的中断发生机制。在正常工作期间，当触摸事件发生时驱动程序利用 PEN_DOWN IRQ 唤醒触摸驱动任务。这样做可以让驱动程序在屏幕没有被触摸时中断自己的执行，而不消耗任何 CPU 资源，而一旦用户触摸屏幕，驱动程序就被唤醒并进入转换模式。我们也可以在转换模式没被激活时停止(disable)ADC 时钟来节省功耗。在校准和主动采样期间，驱动程序使用与上述基本相同的机制检测屏幕是否被触摸；不过在这些模式下驱动程序会屏蔽实际的中断，并通过人工方式简单的检查触摸状态。对于飞思卡尔的处理器，这时要求把控制器编程到触摸检测模式，并检查 PEN_DOWN IRQ 的数据位。对于夏普的处理器，触摸检测内建在 ADC 命令序列中，不需要额外的步骤。

读取触摸数据

在校准和正常操作期间，我们需要读取 X 和 Y 轴的原始数据并去抖动，然后确定屏幕被触摸时是否有稳定的读数。该过程在两个驱动程序中都叫 TouchScan()。该过程的要点是：

1. 检查屏幕是否被触摸；
2. 采集每个轴上的多个原始读数用于以后的过滤；
3. 检查屏幕是否仍在被触摸。

在执行模数转换时，两个控制器都提供了由编程产生延迟的方法，以在给触摸层加电和开始实际的模数转换之间插入一段时延。飞思卡尔把这段时延称作数据建立计数(DSCNT)，在两层切换后会有很多个 ASP 输入时钟长度的延时。夏普把这段时延称为预充时延。两种 CPU 都需要这种时延，因为电阻触摸面板是二块由薄绝缘层隔离的大面积导体，正好形成一个电容。当从将要执行模数转换的层切换到正在加电的层时，需要一定的延时才能保证电容达到稳定状态。

对于飞思卡尔的 i.MX1 处理器来说，一旦我们启动转换过程，那么由 ADC 产生的数据将被保存在一个 16 位宽 x12 个条目深度的 FIFO 中。ADC 产生 9 位无符号数据，因此每个 16 位条目的高 7 位将被忽略掉。这意味着这种触摸控制器的全部数据范围从 0 到 511，不过实际上没有 ADC 或电阻触摸屏会产生接近这个极限值的数据。
我们可以通过编程让处理器在 FIFO 存有任何有效数据时就产生中断，或在输入 FIFO 装满时产生中断。由于我们通常会做多次读取，因此驱动程序一般会在 FIFO 装满时产生中断。当该中断产生时，会有 12 个原始的模数转换数据等待处理，分别对应于 X 轴的 6 次读数和 Y 轴的 6 次读数。夏普 LH79524 处理器允许在产生中断前通过编程完成精确的步骤序列。在执行每个步骤时，结果同样会保存在输入 FIFO 中，等待驱动程序软件的读取。结果是以 16 位数值进行保存。每个结果的高 10 位是模数转换值，最低 4 位是序列索引。10 位转换结果意味着这种触摸控制器的最大数值范围是 0 到 1023，当然你永远也不会观察到接近极限值的结果。一旦序列器控制字在 LH79524 上被编好程，驱动程序获取原始读数所需要做的就是命令序列器执行。当 EOS（序列结束）中断产生时，我们获得的结果就可以用于采集和检查了。序列器可以被配置为当屏幕被触摸时自动触发、根据软件命令触发或连续触发三种模式。要注意原始转换器读数中经常会有一些噪声和偏差，这是正常的。你只有紧紧压住电阻触摸屏才能得到两个连续的读数，并取得一致的 9 位或 10 位原始数据。然而你会发现当触控笔或手指按上或离开触摸屏时，读数的变化要比你保持稳定压力时大得多。要记住

用户是以机械的方式连通二个平面电阻-触摸层。当用户按压和释放触摸屏时，在很短的一段时间内两层之间的电气连接处于临界状态。我们需要丢弃这些读数直到系统稳定下来，否则我们提交的触摸位置读数会产生大幅跳跃，导致更高层的软件无法进行合适的操作。这里不可避免要进行折衷考虑。如果我们要求较窄的稳定窗口，那么驱动程序将无法跟踪快速的"拖曳"操作。对于在签名输入期间发生的滑动或笔划跟踪事件来说快速拖曳是非常重要的。如果我们加宽稳定窗口，我们就可能面临着风险，这些风险包括接收到不精确的触摸数据和上文描述过的处于临界状态的层连接结果。因此需要通过实验来确定适合自己系统的最佳值。智能化的触摸控制器同样允许你通过软件命令调整这些参数。每个样值所需的读取次数、连续读取间允许的偏差以及采样速率是每个驱动程序的全部可编程参数。可以通过#defines 调整这些参数以便在你的系统上产生最佳结果。智能化的外部触摸控制器一般会以很快的速度读取数十或数百个数据用以改善精度。由于我们是用核心 CPU 完成这种过滤，因此我们需要确定有多少时间可以合理地分配给触摸采样任务。嵌入式系统包含折衷，你的任务就是想出最佳的折衷办法，以产生能使用户满意的系统。出于游戏目的，我喜欢测试日常生活中所遇到的商业触摸系统。下一次当你使用触摸屏进行购物签名或包裹签名时，你可以尝试快速大范围波浪形地移动触控笔，然后观察结果，查看屏幕跟踪你移动的程度如何。如果你能看到漂亮光滑的跟踪轨迹，你就知道驱动程序的采样速率相当快，可能在 200Hz 以上。经常你会观察到移动轨迹变成了一条直线(慢速采样)或完全丢失(由于数值改变过大而被拒绝输入)。当你在零售商店进行这种小测试的时候千万不要大呼小叫，否则人们会用异样的目光看你。正常人是不会理解什么东西会使工程师那么激动。

触摸屏的校准

到此我们已经介绍了驱动程序所支持的全部功能，这是我们进入下一步之前必须完成的繁琐工作。既然各种功能都已就绪，可以让用户实际触摸屏幕了。电阻触摸屏需要校准。我们需要一些参考值，以便我们能够将接收到的原始模数转换值转换成高层软件所需的屏幕像素坐标。理想情况下校准程序只要在产品初次加电测试过程中运行一次就可以了，参考值被存储在非易失性存储器中。我已经安排好让触摸驱动程序在一启动时就运行校准程序，但要记住，你要把参考值保存起来，以免让用户在以后的加电启动期间再做校准。不过无论如何你仍然需要向用户提供一种进入校准例程的途径，从而在由于温度漂移或其它因素造成校准不准确时进行重新校准。

校准例程的名称是 CalibrateTouchScreen()，它是一个简单的逐步操作过程，会在屏幕上向用户提供图形目标，并要求用户触摸目标，然后记录下原始的 ADC 读数，该读数将用于后面的原始数据转换到像素位置的调整例程。图形目标和用户提示通过使用便携式图形用户界面(PEG)图形软件 API 显示在屏幕上，不过这也可以通过类似的图形软件实现。

在理想情况下你只需两组(X 和 Y)原始数据，即在屏幕对角读取的最小和最大值。而在实际应用中，因为许多电阻触摸屏存在显著的非线性，因此如果在最小和最大值之间简单的插入位置数值会导致驱动程序非常的不精确。

非线性意味着在屏幕上的等距物理移动会导致原始数据的增量不等。更糟的情况下，即使我们只改变 X 轴的触摸位置，但从 Y 轴读取的数据也会发生很大的变化。为了演示这一现象，我用触控笔在一个典型的电阻触摸屏上从左到右移动，尽量保持 Y 轴位置不变，同时在图上记录 Y 轴的读数。你当然希望触控笔从左到右在 X 轴上滑动时 Y 轴读数能保持一定程度

的恒定，但从图 3 可以看出完全不是这回事。得出的结论是采用的校准点越多越好，尽量减小内插窗口的间距，才能产生可能的最佳精度。如果你能在工厂做一次校准，那么得到大量采样点并不是件难事。如果无法在工厂完成校准，那你必须确定用户需要输入多少个点才能产生足够精确的校准。本文提供的校准例程用了四个数据点，即屏幕的每个角一个。对于参考板上的 VGA 分辨率(640x480)显示屏幕来说，这样做的精度可达到一个或二个像素之内。对于更高的屏幕分辨率或其它触摸屏，要产生一个精确的驱动程序这些点也许过多，也许不够。做出准确判定的唯一途径只能是对具体的硬件进行大量反复测试。在任何情况下，我的建议是尽可能多做些校准点。对用户来说，难得做一次较长时间的校准操作总比正常状态下系统无法对触摸输入做出精确响应要好。

正常操作

一旦校准过程完成，我们就可以开始正常的操作，并开始向更高层软件发送触摸事件。我把提供的每个触摸驱动程序在每种支持的 RTOS 环境中都作为低优先级任务加以执行。任务的入口名叫 PegTouchTask，因为驱动程序需要与 PEG 图形软件进行交互操作。这些驱动程序修改后，可与其它图形软件甚至你自己编写的用户接口环境协同工作。在任何时候 PegTouchTask 总是先调用硬件配置例程，然后调用校准例程，最后进入等待触摸输入的无限循环中。在 MX1 驱动程序中，无限循环通过等待前文描述的 PEN_DOWN 中断事件中止自身循环。当屏幕被触摸时，该任务会持续读取原始数据，将他们转换成屏幕像素坐标，并将触摸位置或状态的变化发送给更高层的软件。我把这称为"活动跟踪"模式。LH79524 驱动程序以相似的方式工作。当产生 PEN_DOWN 中断时，我们命令 ADC 序列器开始进行转换。驱动程序以 20Hz 的速度工作，检查位置的变化，直到屏幕不再处于被触摸的状态。当屏幕被触摸时，我们需要对每个轴连续读取多个转换值以确定触摸位置是否稳定。如果任何两个连续读数中的增量或变化超出#defined 定义的噪声窗口范围，我们就要重新开始。我们一直这样做，直到读取的多个连续值处于#defined 定义的稳定范围内，此时我们可以调整该结果并向更高层软件报告更新。当屏幕不再被触摸时，我们又可以中断此任务，等待触摸输入事件的发生。在每个转换过程的前后，驱动程序必须检查并确认屏幕仍处于被触摸状态。我们不希望向更高层的软件报告实际上是处于"开路状态"的稳定读数。我也看到过有的驱动程序在屏幕被初始触摸后会忽略掉 N 个读数。不过对于这两块参考电路板，我没有发现忽略掉一定数量的初始读数是有必要或有益的。当屏幕被触摸时，驱动程序得到每个稳定的读数，并利用简单的线性插值法将原始数据转换成像素坐标。读取原始数据并将它们转换成屏幕坐标的例程名字叫 GetScaleTouchPosition()。

最后部分

好了,我们终于调整好驱动程序，获得了精确、调整过的、可靠的触摸信息。这些重要的数据能用来干什么呢？如果你正在运行象 PEG 这样的图形用户接口系统，大部分工作到此就结束了。你只要简单的将这些触摸数据整理成消息，并将消息发送到 PEG 消息队列。PEG 软件会对这些数据作出正确地处理。PEG 可以识别三种触摸输入事件类型，分别对应于向下触摸、向上触摸和拖曳。发送拖曳事件是可选的，但如果你希望向用户提供平滑的屏幕移动操作，那么发送拖曳事件就是必须的了。确定该发送哪种类型的消息给 PEG 消息队列的逻辑包含在所提供的源代码中名为 SendTouchMessage()的函数中。这里需要强调的一点是用于发送 drag(PM_POINTERMOVE)消息的名为 Fold()的函数的使用。这是一个方便使用的 PEG API 函数，可以防止用户接口的响应落后于用户的输入。例如，如果用户正在高

分辨率显示器上滚动一个大窗口，那么用户接口很可能在重画滚动窗口时迟滞一段时间。在用户接口的响应能跟上时，用户一释放滚动条屏幕就应该立即停止滚动。但如果消息队列已经包含了一个 PM_POINTERMOVE 消息，我们只需要将这条消息更新到最新位置，而不用再发送新的消息。这样做的效果就是用户接口滚动到最新位置，跳过了对处理器来说太快的所有中间位置更新。这就是 PEG 提供 Fold 函数的目的。它会检查这个消息类型是否已经在消息队列中，如果在，那么它只是简单的更新这条已有的消息，而不是发送全新的消息。如果你正在使用另外一种图形软件包，你也会希望实现类似的功能。

动手下载

本文主要介绍了如何为两个集成了触摸屏控制电路的主流 CPU 编写触摸屏驱动程序。你可以从 ftp://ftp.embedded.com/pub/2005/07maxwell 网站免费下载每个驱动程序的源代码，并按照你的意图使用和修改。

提供精确可靠的触摸信息显然要花费大量的处理器时间。专门设计用于支持触摸屏输入的智能化的 ADC 可以显著地减轻核心 CPU 的负担，并有效地提高触摸屏输入系统的精度。