AN1103

电容触摸传感的软件处理

作者： *Tom Perme*
Microchip Technology Inc.

介绍

本应用笔记说明了使用电容触摸传感来检测按钮按压的各种方法。我们假定读者已掌握了触摸传感的基本知识并且建议在阅读本应用笔记之前，请先阅读 AN1101《电容触摸传感简介》，以便理解硬件知识。

目前市场上的一些电容触摸传感解决方案，提供的仅仅是“黑盒子”式的解决方案，你购买的是一块 IC， IC 发出信号表明是否有按钮按下，但可配置性有限。而 Microchip 的解决方案提供了最大限度的灵活性，这是因为检测按钮按下的软件程序可以完全由用户编写。这并不是说用户必须开发自己的软件程序，Microchip 有提供电容触摸传感程序，您马上就可以用在您的解决方案中。

软件介绍

所有的检测方法的工作基本原理都相同，将测量值与频率计数的滑动平均值进行比较，频率计数值下降就表明有按钮按下。扫描按钮的基本物理过程是要设置对电容变化敏感的振荡器，在按钮焊盘上以固定的时钟周期振荡。在固定周期之后，测量频率并检查与正常情况是否相同。然后，将振荡器移向下一个按钮焊盘并进行扫描。对众多按钮的扫描皆按顺序依次完成。

用户必须创建两段主要代码。图 1 的流程图给出了电容触摸传感程序的基本流程。电容触摸传感代码的第一段是“电容的初始化”，初始化工作包括使能振荡器、设定端口引脚方向以及所有相应的初始化设置。第二段重要代码包含一系列程序处理代码块，称作“Cap ISR”，它们是在中断时（T0IF 标志位位置位的话）将执行的代码块。这些代码块的功能是判断某个按钮是否按下，并且依次扫描所有的按钮。在后面的段落中将详细描述这些重要的代码块。

初始化

首先，必须恰当地对硬件进行初始化。如何正确地设置 PIC16F88X 系列器件，详见**附录 A：“PIC16F88X 系列的寄存器设置”**。不同系列的器件可能在寄存器设置值上有细微差别，但关键位和信号通道的设置都是相同的。因此如果使用其他系列的器件，寄存器存在差异的话，**附录 A：“PIC16F88X 系列的寄存器设置”**可作为设置寄存器位时的指导。

以下是一个简短的清单，以确保一切都设置妥当：

- 引脚方向和模拟 / 数字选择
- 振荡器信号通道使能
- 使能定时器和设置 Timer0 预分频数
- 使能中断

中断服务

电容触摸传感是基于中断的， Timer0 发生中断，置位标志位 T0IF。中断服务程序（Interrupt Service Routine，ISR）首先要检查中断标志是 Timer0 中断还是其他中断。如果是 Timer0 中断，那么必须提供电容触摸传感的中断响应服务。

如果是其他中断，则ISR是由相应中断向量指向的程序，ISR 响应该中断服务，但在中断程序结束时必须确认 Timer0在ISR执行期间是否翻转。如果Timer0翻转了，那么 T0IF 标志位会被置位，但它代表的采样将被认为是失败采样，然后应清零 T0IF 标志位，并重新启动定时器，进行另一次采样。采样失败是因为基于 Timer0 的固定测量周期，如果不在中断时立刻响应那么之后情况可能会发生变化。

下面各小节将依次对流程图中 ISR 的各程序段进行说明，首先是“读 TMR1”。

中断服务：进行一次读操作

扫描刚好完成时，为了获得当前传感器的读数，就必须读取 Timer1 的值。需要一个无符号整型变量来储存原始值。获得读数的代码如例 1 所示：

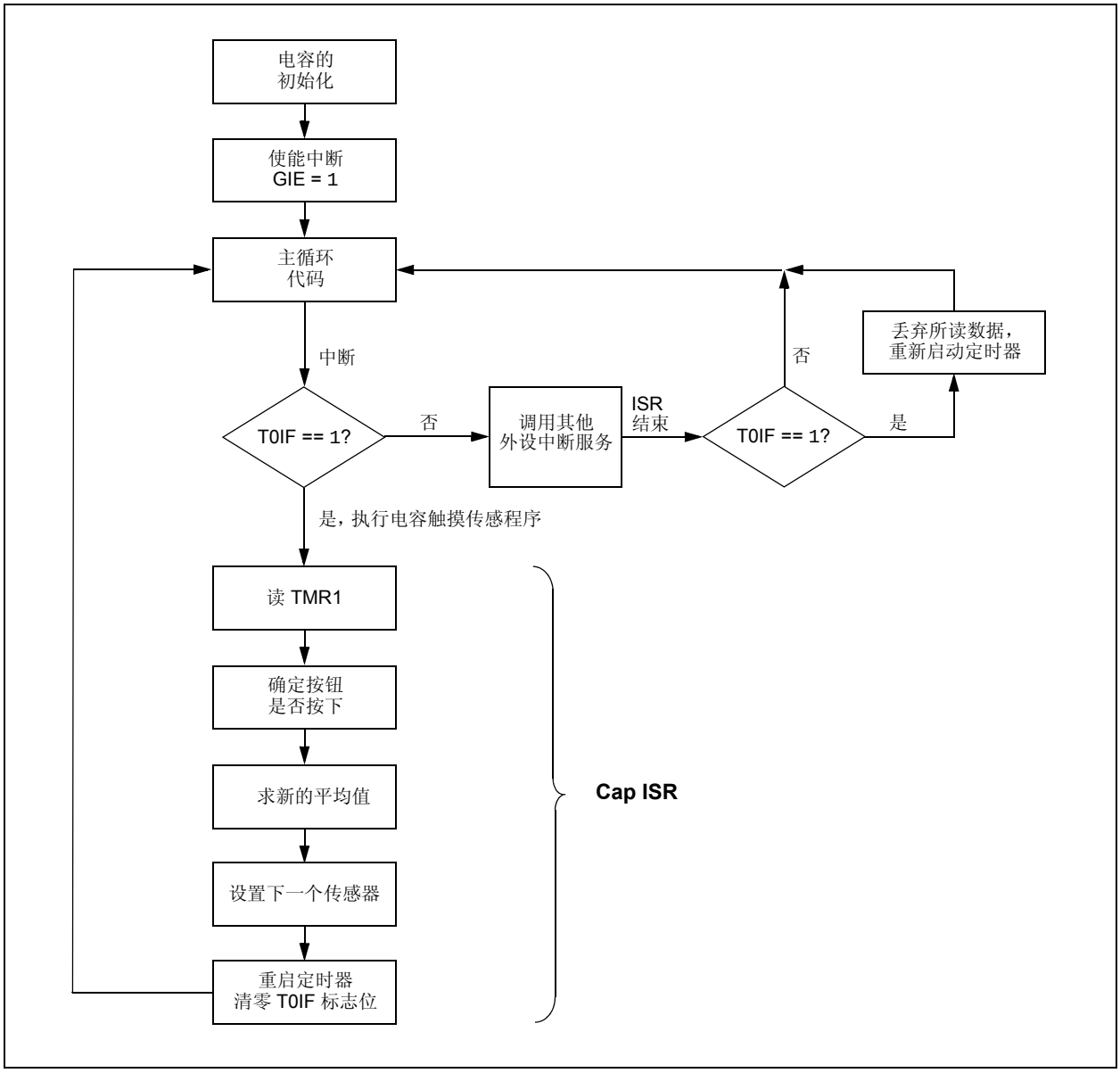
例 1：读取频率

```
unsigned int value;
value = TMR1L + (unsigned int)(TMR1H << 8);
```

© 2008 Microchip Technology Inc.

变量 value 的结果将是当前传感器的读数，当前传感器是在上一次电容触摸传感服务程序执行时设置为即将扫描的传感器。随即变量 value 的值将会与 16 个点的平均值进行比较，以确定频率计数是否有显著下降。

图 1： 软件流程



© 2008 Microchip Technology Inc.

中断服务：确定按钮是否按下

本节将介绍一种非常简单的按钮按下检测算法，如例 2 所示。

例 2：简单的按钮检测

```
if (raw < average-trip))
    // 按钮按下
else
    // 按钮未按下
```

变量 trip 存放用来衡量频率与平均值的差值，要判断按钮已按下，则 raw 下降的幅度必须至少是这一值。举个简单的例子，如果滑动平均值是 9000，而 trip 设为 800，那么在一个按钮被认为是按下之前，raw 必须降为 8200。

所有的检测方法，其工作的基本原理都相同，将测量值与频率计数的滑动平均值进行比较，频率计数值下降表明有按钮按下。其他更为复杂方法详见 **“按钮检测算法”** 一节，其中还包括若干细节资料和可能需要的补偿。

中断服务：求平均值

对当前所读的数据求平均值是相当简单的一步。为了使平均算法更有效率，程序并不存储 16 个变量来做 16 点平均。而是对当前读取值取 1/16 的加权，而滑动平均值取 15/16 的加权。计算 16 点平均值的代码如例 3 所示，数组下标代表按钮的编号。

例 3：计算平均值

```
average[index] = average[index] + \
    (raw – average[index])/16;
```

求 N 点平均时，N 取 2 的幂可以节省处理时间，因为右移可以取代软件除法。如果使用汇编的话，应该使用右移指令，但在使用像 HI-TECH PICC™ 这样的智能 C 编译器时，编译器会识别出除数是 2 的情形，从而使用右移指令。

中断服务：准备下一个要扫描的传感器

一旦程序完成 Timer0 中断服务响应，就应设定下一个要检测的传感器。这包括设定 index 指向下个传感器以及为比较器输入设定适当的连接。

1. 设定 index。
2. 设置比较器输入通道。
3. （可选的）设定外部多路复用线选控制。

index 变量增一并在恰当的时候翻转归零。这是假设用了一个平均值数组来保存所有按钮的平均值。在例 4 中，假设使用了四个按钮，因为具有 SR 锁存的 PIC® 器件的标准按钮数量是四个。

例 4： INDEX 顺序循环

```
if (index < 3)
    index++;
else
    index = 0;
```

注： 当使用 2 的幂时，与（AND）运算可以简单地把 $2^n - 1$ 翻转到零。对 4 个按钮的代码是：

```
index = (++index) & 0x03;
```

然后使用预定义的常量来配置比较器输入。这些常量都源自于正确的 CM1CON0 和 CM2CON0 寄存器设置。它们存放的是正确的信号设置，并在 CMxCON0<1:0> 位上有不同的值对应内部多路复用器通道，这样可以确定使用的是哪个比较器的负输入端（C12INx-）。预定义的常量如例 5 所示：

例 5：预定义常量

```
// C12INx-    0     1     2     3
COMP1[4] = {0x94, 0x95, 0x96, 0x97};
COMP2[4] = {0xA0, 0xA1, 0xA2, 0xA3};
```

必须根据 index 变量把比较器的寄存器设置为下例所示常量之一。当使用某器件的四个固有按钮时，按钮的序号直接映射成比较器输入通道。

例 6：按钮下标的映射

```
CM1CON0 = COMP1[index];
CM2CON0 = COMP2[index];
```

然而在处理多个按钮时需要小心地根据所扫描按钮的下标来正确设置比较器的输入通道。当采用外部 MUX 大量扩展按钮数量时这一点很重要，因为此时下标和它代表的比较器输入通道不再直接相关。关于这个问题的更多详细信息，请参阅 AN1104 《配置多个电容触摸传感按钮》。

© 2008 Microchip Technology Inc.

中断服务：重启定时器和清零 T0IF 标志位

为了保持连续读数，Timer0 和 Timer1 在每次读数据开始前都要清零。置位 TMR1ON 来重新使能 Timer1。最后，必须清零 T0IF 中断标志位，否则程序会立即返回到中断服务程序中。

例 7：中断服务

```
TMR1L   = 0;
TMR1H   = 0;
TMR1ON  = 1;
TMR0    = 0;
T0IF    = 0;
```

此时下一个传感器已准备好，在下一次 Timer0 引发中断时，电容触摸传感服务程序将会再次运行，完成对该传感器的扫描。

扫描速率

现在将依次扫描每个按钮。此时的问题是：“扫描需要多久？”单个按钮的扫描速率主要是由以下公式确定：

公式 1：

$$T_{SCAN} = 256 \times (4 \times T_{OSC}) \times PS$$

这是Timer0翻转归零并产生溢出中断所需要的时间。三个主要的参数是 Timer0 溢出的计数数值（256）、每条指令的指令周期（4 x TOSC）和Timer0预分频数（PS）。如果预分频为 1:4，则 PS 值就是 4。这是理想情况下的公式，因为它假定不存在任何额外的时间损失。例如，当使用计算密集的方法（如下文所述的百分比方法）时，将添加额外的时间开销。

按钮检测算法

在编写固件之前要创建一个良好的系统，首先要做的就是要使检测更为容易，通过使用寄生电容小的传感器从而能够检测到较大的电容变化。这将使系统功能更容易实现并减少开发时间。

对合理的系统而言，即使是非常小的变化也能被检测到。能够控制如何去检测出变化量是衡量应用系统表现的一个很好的观察点。我们所期望的系统是具有简单的按钮并能在一定距离内感应到人的触摸。

Microchip 已经开发了几项软件技术，能够透过窗户玻璃、Plexiglas® 或其他绝缘材料表面检测到按钮按下。

正如介绍中所述，算法总是要用到求平均值并且通常而言放慢求平均值的速度是有益的，因为微处理器能在手指靠近焊盘尚未接触之前就完成求平均值计算，这样的话按钮按下永远不能被检测到。放慢求平均值的速度可以在中断服务程序完成，比如可以每两次采样求一次平均值，或者每四次采样求一次平均值。做法是在每次按钮切换后，如果 if 语句判断通过，表明有按钮按下，但若还不到采样次数，就不进行计算而是继续检测。

三种按钮检测算法是：

- 方法 1：直接判断，固定滞后量
- 方法 2：百分比判断
- 方法 3：一次计算一个百分数，不断求平均

方法 1：直接判断，固定滞后量

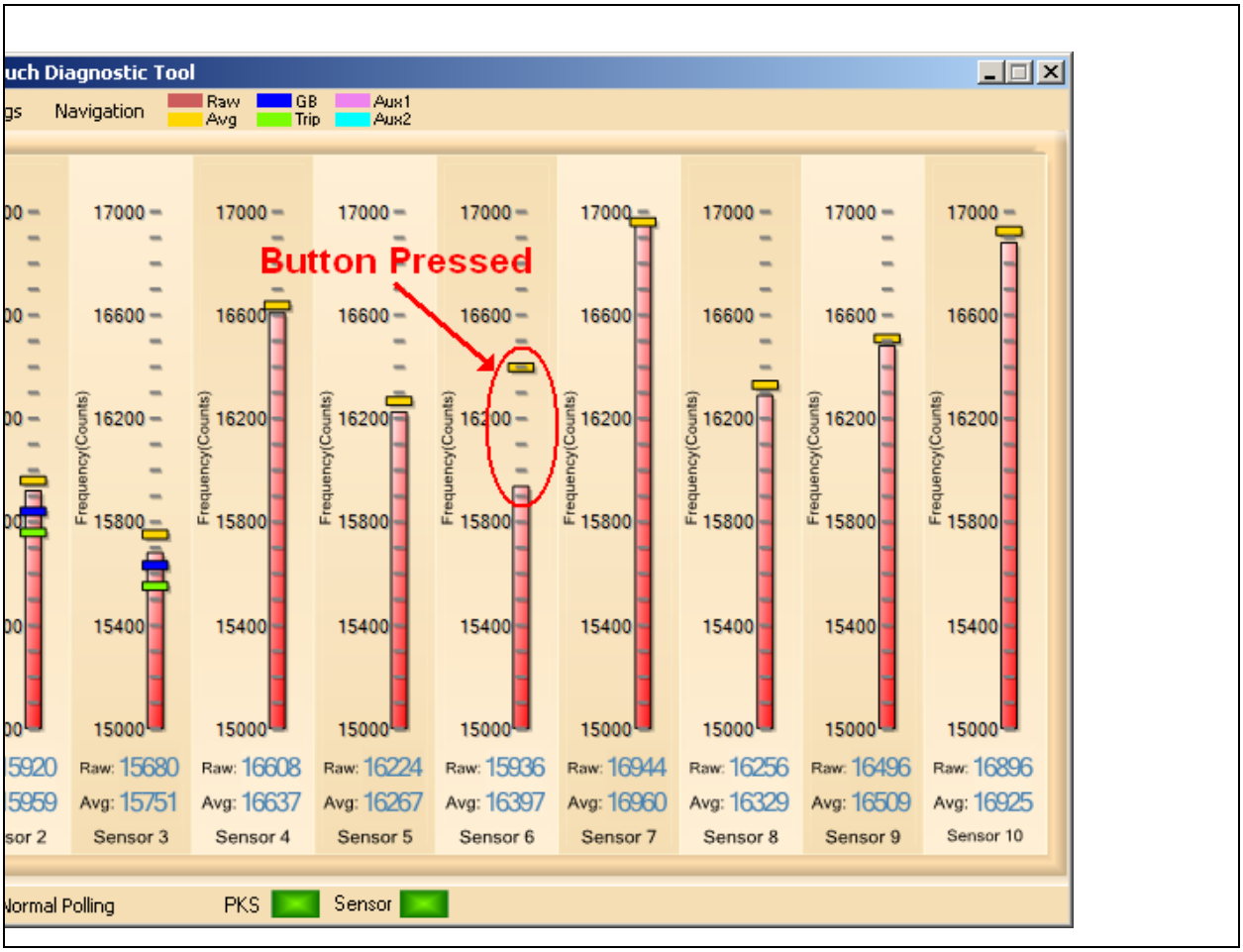
考虑到执行周期和直接处理 raw 值，这种按钮检测算法是最快的。它依赖于三个关键因素：慢速计算平均值、在每次平均值计算停止之外提供一个小的滞后量以及找到并设置一个恰当的 trip 阈值。必须根据试验来找到恰当的阈值，不过对于已给定的系统，阈值变化不大。Microchip 提供了 mTouch 诊断工具来辅助分析您的电容触摸传感系统。使用 PICkit™ 串行分析器，诊断工具可以通过 I^2C™ 接口与系统进行通信，它还能显示实时数据来描述系统特征。由于显示更加直观，这将更有助于理解电容触摸传感系统的运行情况。

为了确定trip阈值和可接受的滞后量，下面使用mTouch诊断工具的练习将演示如何为 Microchip 演示板挑选恰当的数值。

图 2 显示传感器 6 有手指按在按钮上。接近 16,400 的金黄色条是平均值，位于 16,000 刻度下面一点的红色条是当前的 raw 值。尽管没有在诊断工具上表示出来，但平均值已停止跟踪 raw 值，因为它已经超过了 trip 阈值。传感器 2 和 3 上的蓝色和绿色条是信息条，用户可以通过设置它们来改变 trip 阈值的显示刻度，从而看得更清楚（设置信息条不会对单片机进行数据读写）。

© 2008 Microchip Technology Inc.

图 2： mTouch 诊断工具截图



现在对于正在测试的系统，传感器 6 有一个正常的、未被按下的平均值，大约是 Average = 16397。这个值可以在图下方读到。按下时的值是 Raw = 15936，与平均值的差值是 461。因此，恰当的 trip 阈值应该是差值的 80%，大致为 370。

选择：

```
trip = 370
```

若将 trip 准确地设为 461，那么对于环境变化或未知变化就没有留下任何容许误差，这可能导致无法产生按压的判断。而 trip 设置得太小，将使得系统极为敏感，可能在实际接触按钮之前就认为按钮已经按下。如果设定过于敏感，也可能无意中触发相邻的按钮。

现在只剩下一个重要问题的选择了，即提供多大滞后量。实际只需要一个很小的滞后量，它可由试验来确定。这里需要滞后来简单地防止抖动，也可以防止按钮失效。最简单的例子是，假定在按压之后 raw 值会一直上升到先前的平均值。如果另一个按钮被按下，或由于金属、水或其他环境因素的变化，包括有大的拉电流流过单片机，读数值都可能下降， raw 值未必能完全上升到之前的平均值。在一次读数过程中，由于手指在按钮上挪动而造成的抖动是相当小的，且如果手指不动，那么抖动通常是在计数 16 次以内。因此取 64 的滞后量将给抖动的手指按压（或其他来源的抖动）提供计数量为 48 至 64 之间的滞后。

选择：

```
hysteresis = 64
```

最后对滞后进行检验，它不能大于平均值。在手指按下时变化非常小的系统中会发生这样的情况。当前系统有一个变化量为 461 的中等改变，且 trip 值为 370，再减去滞后量的 64，余下在滑动平均值以内的平均范围为 306。现在设计已经完成就等着实现了。例 8 给出了关键设计部分的代码实现。

© 2008 Microchip Technology Inc.

例 8：方法 1 代码：直接判断

```
// 假定我们为下标为 6 的传感器编写代码，一共有 16 个按钮。
unsigned int  average [16];
unsigned int  trip [16];

CapInit()
{

   // 初始化代码未完全给出

   trip[6] = 370;     // 设定传感器的判决水平
}

CapISR()
{

   GetReading();

   // index = 6 的示例
   if (raw < (average[index] – trip[index]) ) {
       // 按钮按下
       // 1. 为下标为 # 的传感器置位按钮标志位
       // 2. 不进行平均值计算（不需要任何操作）

       // 1
       switch(index) {
           case 0:            Buttons.BTN0 = 1; break;
           case 1:            Buttons.BTN1 = 1; break;
                              …
           case 6:            Buttons.BTN6 = 1; break;
                              …
           default   :    break;
       }
} else if (raw > (average[index] – trip[index] + 64)){
       // 按钮未按下

       // 1. 为下标为 # 的传感器清零按钮标志位
       // 2. 执行慢速平均值运算

       // 1
       switch(index) {
           case 0:            Buttons.BTN0 = 0; break;
           case 1:            Buttons.BTN1 = 0; break;
                              …
           case 6:            Buttons.BTN6 = 0; break;
                              …
           default   :        break;
       }

       // 2
       if (AvgIndex < 2)     AvgIndex++;
       else                  AvgIndex = 0;

       if (AvgIndex == 2)
            average[index] = average[index] + ((long)raw-(long)average[index])/16;

   }

   SetNextSensor();
   RestartTimers();

}
```


© 2008 Microchip Technology Inc.

注：	使用 PC 软件 mTouch 诊断工具和 PICkit 串行分析器会引入边缘电容，但只要系统不在极限情况下运行，其影响不大。如果可能，请采用实际系统元件来测试系统而非模拟测试。

方法 2：百分比判断法

从平均值和原始值总结出来的一个好方法是在相对平均值改变的百分数的基础上来进行检测。这样做可以使调校电容触摸传感系统变得更为容易并且提供更高的可靠性。本方法需要更多的处理时间来进行必要的计算，但是和方法 1 相比能减少存储器的占用，因为不用再保存每个按钮单独的 trip 值组成的数组。其他任务，例如缓慢平均值计算和设置传感器等，仍然是一样的。

在做数学计算求平均值时，结果通常是小数，如 0.05 等。而对单片机来说处理整数要更加容易，因此可以将计算结果乘上 100，那么 5% 就可得到 5。如果乘上 1000 的话就可以包含百分数的第一个十分位，那么可以用 52 来代表 5.2%。执行百分数运算的代码如后面例 10 所示。

注意，如果百分数是负的，则 raw 值比平均值大。在设定平均值时应考虑到这一点，因此负值都被忽略并置为零，这样将使平均值计算的逻辑更简单。

现在的判断数值是一个百分数，“ON”代表有按钮按下，对应百分数是 PCT_ON。这个阈值像方法 1 中的 trip 一样，也源于试验数据，但可以是在微弱按压的 1% 和强力按压的 25% 之间的任何值。还应有某个值对应的百分数小于代表按钮释放的情况 “OFF”对应百分数 PCT_OFF，所以必须对 PCT_OFF 进行设置，把滞后考虑进来后，它将与 PCT_ON 不同。最后像前面那样进行慢速平均值计算，一直到平均值达到按钮释放时的阈值为止。

例如，假定有如下数值。这些数值都被处理成整数偶数值，但它们都是实数。

未按压平均值	15000
按压平均值	13500
差值	1500
百分比差值	10%

因此，在选择安全的按钮按下对应百分数时，恰当的 PCT_ON 值应该设为大约 8%，或者说是 10% 变化量的 80%。对于未按下按钮对应的百分数而言， 1% 的滞后已经足够，因为绝对改变量的大小正好；对本例而言，1% 的滞后是 150 次计数。这是基于用百分比计算方法的典型例子，只是在极低频计数的读操作会有差别。因此 PCT_OFF 应为 7%。如果需要更大的滞后量，可以选择 6% 或 5%。

选择：

```
PCT_ON = 8
PCT_OFF = 7
```

例 9：百分比计算

```
long percent;

CapISR()
{
...
     percent = ((long)average[index] - (long)raw[index]);
     if (percent < 0){
          percent = 0;
     } else {
          percent = percent * 100;
          percent = percent / average[index];
}
...
}
```

现在设计已经完成就等着实现了，实现的代码如例10所示。

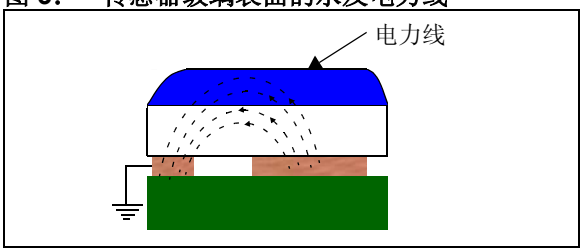
例 10：百分比算法的实现

```
// 定义 on/off 的百分数
#define PCT_ON    8
#define PCT_OFF   7

CapISR()
{
 GetReading();

 // GetPercentage() 调用前面例子中的代码。
 percent = GetPercentage();

 if (percent < PCT_OFF) {

    // 1. 按钮标志位清零
    // 2. 执行慢速平均值计算

    // 1 （同方法 1）
    switch(index) {
      case 0:   Button.BTN0 = 0; break;
                …
      default:  break;
    }

    // 2 （同方法 1）
    if (AvgIndex < 2)  AvgIndex++;
    else               AvgIndex = 0;

 } else if (percent > PCT_ON) {

    //  1. 按钮标志位置位
    //  2. 不进行平均值计算 （不做任何处理）

    // 1 （同前面一样）
    switch (index) {
      case 0:   Button.BTN0 = 1; break;
                …
      default:  break;
    }
 }

 SetNextSensor();
 RestartTimers();
}
```

方法 3：百分比 “选择”法

还有第三种方法，主要用于环境变化剧烈的场合，比如按钮附近持续有水的情况等。在本方法中，任何时刻只能有一个按钮按下，选出按压最大的那个按钮，认为它就是被按下的按钮。这就是通过 “选择”选出按压最大的按钮。如果有大量的水洒在传感器的表面，它会增加与附近地面介电系数，就会导致虚假的按钮按下信息。这是因为水的介电常数比较高，从而在水覆盖的整个表面，形成了更大的电容。

图 3： 传感器玻璃表面的水及电力线



如果仅是由于物理原因有水洒在传感器表面上，没有一种完美的电容触摸传感解决方案来处理这一问题。水溅落到传感器表面，可看做是一个手指按下，但如果水一直在传感器表面呢？第三种方法的提出，就是为了试图处理这种情况。

一般来说，当水溅到传感器表面并停留在那里时，透过水的手指按压仍然可以检测到。在这种情况下，必须根据现实情况来设置滑动平均值的大小，即有水在传感器表面的平均值。这要求不断计算滑动平均值，而按钮按下只有一段很短的持续时间，那么就可以计算出新的平均值来反应这种较低的按压值。

如果水不止洒到一个按钮上时，按压最大的那个按钮往往是正确的按钮。比如，假定水覆盖了三个按钮，称为按钮 1、 2 和 3，如图 4 所示。系统已经计算出它们在有水状态下的平均值，并有人按下了按钮 3，由于水在三个焊盘上都产生电容，所有三个按钮的频率都会下降而且比没有水的时候下降得多。但是按钮 3 上频率计数的下降将可能比按钮 1 和 2 下降得更多。现在该算法指出，应该从三个按钮中选出百分比最高的那个按钮，百分比算法如方法 2 所述。

方法比较

这三种方法各有优缺点，但都是适合使用的。如果程序和 RAM 存储器空间有限，如 PIC16F610，百分比方法或百分比选择方法有可能不合适，因为它们会消耗大量的器件资源。使用存储空间更大的器件，如 RAM 更大的 PIC16F887 系列或 PIC16F690 系列器件，就能使用百分比算法。表 1 给出了三种方法之间的优缺点。

© 2008 Microchip Technology Inc.

表 1： 软件方法比较

	RAM/PGM 存储器	易于实现	外部物质 （H_20）
直接判断法	+	-	-
百分比法	-	+	-
百分比 “选择”法	- -	+	+

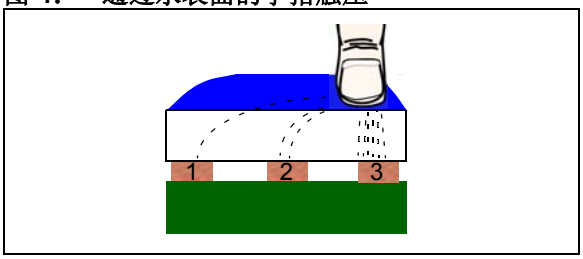
基于百分比法的程序使用更多数学计算来确定按钮是否按下，因此需要更大的存储器。百分比选择系统需要两个额外的数组来排序选出哪个按钮按压最大，它在百分比法数学计算程序的基础上需要更多的存储器。

基于百分比法的程序的主要优点是易于实现。只需在 raw 值基础上多算一步总结出百分比，对多块电路板，甚至对整个设计，都能更方便地得到合适的工作百分比值。尽管实际差值可能在不同设计或在一个特定设计的不同按钮上都可能有显著不同，但百分比变化大致是相同的。因此，与方法 1 “直接判断”确定合适的 trip 值比较起来，百分比法有助于更快地得到可用的解决方案。

此外，使用百分比选择系统，有助于处理感应区附近存在外部导电物体的情况。它也非常有助于处理表面一层薄水膜的情况，而其他两种方法都很可能检测出所有有水膜覆盖的按钮都被按下，但当水汇集的情况发生，或者金属又或是另一种高介电常数或导电材料在附近时，它也会像其他两种方法那样失去其可靠性。

前面两种方法：直接判断和百分比判断，可以作为很好的入门方法，并可为针对具体应用进行定制。其他特性，例如按钮时间超时、在按下和释放传感器焊盘的情况下激活按钮，以及其他应用需要的特性，都可以整合到这些方法中去。

图 4： 通过水表面的手指触压



在这种情况下，无法确定地判断一个按钮是否按下，但是在面板上只有小到中量的水时，还是能够提供适当的可靠性。如果感应表面只有一小片水膜仍将正常工作，对有凝结或露水的情况也一样。然而，当表面严重浸水时，该方法即不再可靠。

处理水问题的最好方法是设计难于洒落，滑过或者停留水的感应表面物理系统。例如，安装按钮时使表面形成倾角，从而水无法停留在表面；这样就只可能出现水的薄膜。当积水不再是一个影响因素的时候，可靠性将大大提高，于是只有水滴溅落才可能导致虚假检测问题。

随本应用笔记提供的软件中，给出了方法 3 的电容触摸传感服务程序实现代码。相比其他方法，代码要更复杂些，其关键在于始终进行平均值计算，只要按钮的百分数超过阈值时，根据各百分数而确定的按压最大的按钮被认为是按下的按钮。

注意事项

Timer1 溢出

由于主要测量是读取自 TMR1 的值，在 Timer1 溢出之前， Timer0 必须溢出并产生一个中断去读 Timer1。这是由 Timer0 的预分频数和系统振荡频率决定的。通常安全合适的预分频数是从将 OPTION 寄存器中的 PS<2:0>设置为 0x2 开始。更长的 Timer0 周期将允许更大的计数，与非常快的周期相比较，读数更平稳，但是这样做的代价是扫描单个按钮的周期更长。

Timer1 是 16 位定时器，只要频率计数读数远低于最大无符号整数 65,535，那么 Timer0 和预分频位就是可以接受的。

按钮失效

这里所说的按钮失效是指已按下但释放后没有关断的按钮。“失效”按钮通常是由两种原因引起。第一个原因是未调节好的按钮，可能会导致拉电流。第二个原因是突发强拉电流，这将在**“突发的强拉电流”**一节中讨论。

作为设计人员，不断调整 trip 阈值是不可取的，调整次数越少越好，最好不要调整。对于已经完成的量产设计而言，一旦找到系统 trip 值或百分比阈值的最优配置，通常并不需要对生产的每个系统进行改动。前面所述的 trip 值和百分比阈值按钮检测方法可能会遇到不当设置所带来的问题，但绝对值的 trip 阈值的问题会更多。

通常原因是在按钮按下时，按压的反作用导致拉电流流过器件，如 LED，这会使频率计数降低。当按钮被释放时，即使人的手指已经拿开，该电流仍然会使频率下降一些。如果 trip 设置过小，以致非常轻的按压都会开启按钮 （同时有拉电流的反作用），频率降低的幅度可能足够使频率计数保持在 trip 阈值之下。同样，如果 trip 阈值太小，相邻按钮的按压可能触发其邻近按钮的虚假按压。

修正方法是在不损失触摸检测可靠性的条件下，尽可能地让 trip 阈值取最大值。在前面的例子中，建议使用手指按压时的频率计数改变量的 80% 是合理的，因为较轻的手指按压和较重的按压相比引起的改变更小。因此，应该测试可预计的最轻按压，然后应为该按压提供一定的安全余地，任何较重的按压都将被检测到。

按钮失效在正常无误用的情况下是容易防止的。**“突发的强拉电流”**一节讨论的话题是关于突发强拉电流流过单片机时会产生的更多问题，并且还需要非常好的设置来解决这一问题。

突发的强拉电流

流向 PIC MCU 电流的显著增加会导致振荡器频率略有下降。如果预先没有考虑到，这可能成为按钮失效的潜在因素。然而和一些竞争对手相比，Microchip 能够在进行电容触摸传感的同时，为各种 LED 提电，这就是强拉电流的简单例子。如果必须接通非常大的电流，那么可能的话应逐步地增加电流消耗。拉电流的大小并不是问题，而拉电流的突然改变才是问题。

按钮按压扫描的时间和拉电流作用的时间交错开来，是解决这两项工作的好方法。

如果没有交错工作，通常不需补偿的，但是对某些对象（如在非常邻近区域有很多按钮的滑动条），邻近按钮的电容“串扰”和 LED 驱动可能导致按钮失效。一个按钮按下后将略微拉低相邻按钮的频率，这也是不期望发生的情况。

当单个按钮需要电流损耗补偿且常出现突变的强电流时，有一种方法是手动调整平均值，调整到一个比当前静止状态更接近 trip 阈值的点。如果这样做可行，那么它将取决于应用，但在很多情况下，突变电流的变化能够根据设计来预估。如果在按钮按压时预计瞬态出现 50 mA负载电流，那么这个问题就能被解决。手动降低平均值可让按钮被正常释放。

对像滑动触摸条这样，一个按钮导致另一按钮失效的情形，如果可以在应用中重新设置失效按钮的平均值为当前读取的 raw 值话，就没有问题了。此时先前失效的按钮将被校准，即根据系统当前配置（具有 LED 拉电流和 / 或手指仍旧按下等）来进行校准，这样改变滑动触摸的阈值，那么按压另一按钮就更加容易。

最后，如果器件接通时的拉电流太大，可使用外部驱动器来避免电流流过单片机。

滞后量不足

如果给定的滞后非常小，电容触摸按钮接通了像 LED 这样的电流负载，那么就可能进入振荡状态。通常按钮成功按下，但由于有负载拉电流，就会使按钮的开关状态存在很大的不确定性。简单的修正办法是在按压和释放的阈值间提供更大的滞后量。

© 2008 Microchip Technology Inc.

厨房中调味品所引起的问题

家庭厨房环境有一系列其他环境下不太可能会遇到的问题。一种衡量厨房用具安全性和质量的方法是：当厨房用具溅上蕃茄酱时，观察是否有按钮按下的虚假指示。本节中只讨论番茄酱问题，但也适用于其他调味品。

一大片喷洒的番茄酱，比如在界面面板上挤番茄酱，将有可能导致电容触摸传感系统的频率下降。厨房用具的安全性要求是在这种情况下不会指示虚假触发，因为这是番茄酱而非人的手指。期望溅出物的量足够小并不是个好的解决方案。由于没有办法去改变实际情况，因此需要利用所给的硬件设置设计出能适当地检测出番茄酱喷洒物的固件。

因为无法始终防止频率的下降，所以在一定时间内监视按压和释放是个可行的解决办法。这需要人在固定的时间内按压和释放按钮，比如说一秒钟。在固件内部，这被看成是频率在允许时间内的一次下降和上升。如果溅上了番茄酱，它将粘附并保留在表面，那么释放的动作就不会发生。固件甚至能做得更先进，即要求按钮在它释放前能保持一定的时间，这样可以防止非常快速的按压。

更小的泼溅影响 （如从平底锅或旺火炒菜溅出的油），其影响要小得多。就目前情况来考虑，通常这些泼溅物对系统没有明显的影响，而只有大的变化才会显著地增加按钮上的电容。这些物品都是典型的水混合物，如番茄酱、芥末以及水本身。所有这些物品都有非常高的介电常数，这也是它们接触面板表面时会使电容增加的物理原因。

总结

与本应用笔记一起提供的软件可帮助您理解并加快设计。实现电容触摸传感的软件可以是非常简单的，也可以是能够处理按钮检测的复杂算法。由于软件的改动更加容易，因此用户可以有更大的灵活性来决定其电容触摸传感系统的工作方式。

其他参考资料包括：

AN1101 《电容触摸传感简介》

AN1102 《电容触摸传感器布板和物理设计指南》

AN1104 《配置多个电容触摸传感按钮》

附录 A： PIC16F88X 系列的寄存器设置

下列寄存器可以在《PIC16F882/883/884/886/887 数据手册》（DS41291D_CN）中找到。每一位的详细解释可以在器件的数据手册中找到。这些寄存器设置也可作为其他系列寄存器设置的指导。

寄存器 8-1： CM1CON0：比较器 C1 控制寄存器 0

R/W-0	R-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
C1ON	C1OUT	C1OE	C1POL	—	C1R	C1CH1	C1CH0
bit 7							bit 0
1	0	0	1	—	1	0	0

寄存器 8-2： CM2CON0：比较器 C2 控制寄存器 0

R/W-0	R-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
C2ON	C2OUT	C2OE	C2POL	—	C2R	C2CH1	C2CH0
bit 7							bit 0
1	0	1	0	—	0	0	0

寄存器 8-3： CM2CON1：比较器 C2 控制寄存器 1

R-0	R-0	R/W-0	R/W-0	U-0	U-0	R/W-1	R/W-0
MC1OUT	MC2OUT	C1RSEL	C2RSEL	—	—	T1GSS	C2SYNC
bit 7							bit 0
0	0	1	1	—	—	1	0

寄存器 8-4： SRCON：SR 锁存器控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/S-0	R/S-0	U-0	R/W-0
SR1(1)	SR0(1)	C1SEN	C2REN	PULSS	PULSR	—	FVREN
bit 7							bit 0
1	1	1	1	0	0	—	0

注 1：要使 SR 锁存器输出到引脚，必须正确配置 CxOE 和 TRIS 位。

寄存器 8-5： VRCON：参考电压控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VREN	VROE	VRR	VRSS	VR3	VR2	VR1	VR0
bit 7							bit 0
1	0	0	0	0	1	1	1

寄存器 8-6： ANSEL：模拟选择寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7(1)	ANS6(1)	ANS5(1)	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0
0	0	0	0	0	1	1	1

注 1：在 PIC16F883/886 上不可用。

© 2008 Microchip Technology Inc.

寄存器 8-7：　　ANSELH：模拟选择高位寄存器

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
bit 7							bit 0
—	—	0	0	0	1	1	0

ANSEL 位选择可以使能全部四个比较器的输入。

注：

© 2008 Microchip Technology Inc.

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 Accuron、 dsPIC、 KEELOQ、 KEELOQ 徽标、 MPLAB、 PIC、 PICmicro、 PICSTART、 PRO MATE、 rfPIC 和 SmartShunt 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

FilterLab、 Linear Active Thermistor、 MXDEV、 MXLAB、 SEEVAL、SmartSensor 和 The Embedded Control Solutions Company 均为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、 Application Maestro、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 dsPICworks、 dsSPEAK、 ECAN、 ECONOMONITOR、 FanSense、 In-Circuit Serial Programming、 ICSP、 ICEPIC、 Mindi、 MiWi、 MPASM、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 mTouch、 PICkit、 PICDEM、 PICDEM.net、 PICtail、 PIC^{32} 徽标、 PowerCal、 PowerInfo、 PowerMate、 PowerTool、 REAL ICE、 rfLAB、 Select Mode、 Total Endurance、 UNI/O、WiperLock 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 是 Microchip Technology Inc. 在美国的服务标记。

在此提及的所有其他商标均为各持有公司所有。

© 2008, Microchip Technology Inc. 版权所有。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08

© 2008 Microchip Technology Inc.