AN11023_1

基于 LPC11XX 的电容式触摸传感

Rev. 01 — 12 December 2010 应用手册

文档信息

信息	内容
关键字	LPC1112，LPC1100，Cortex M0，电容式触摸感应
摘要	该应用手册描述了应用 NXP 的 LPC1100 微控制器来设计一个简单电容式触摸传感器的方法

深圳市伟博创科技有限公司
Tel:(0755)87240703 Fax:83240724
http://www.weboch.com.cn
http://www.longdouble.com

1. 简介

该应用手册描述了使用 LPC11XX 微控制器的 ADC（模数转换）输入来实现一个简单的电容式触摸传感器的方法。

手册中使用的电容式触摸传感器位于 PCF8883 评估板的 PCB 板上的镀铜区域(见图 1)。4 个传感器中有一个通过阻容网络（RC）与 LPC11XX 微控制器 ADC 输入端连接（见图 2）。

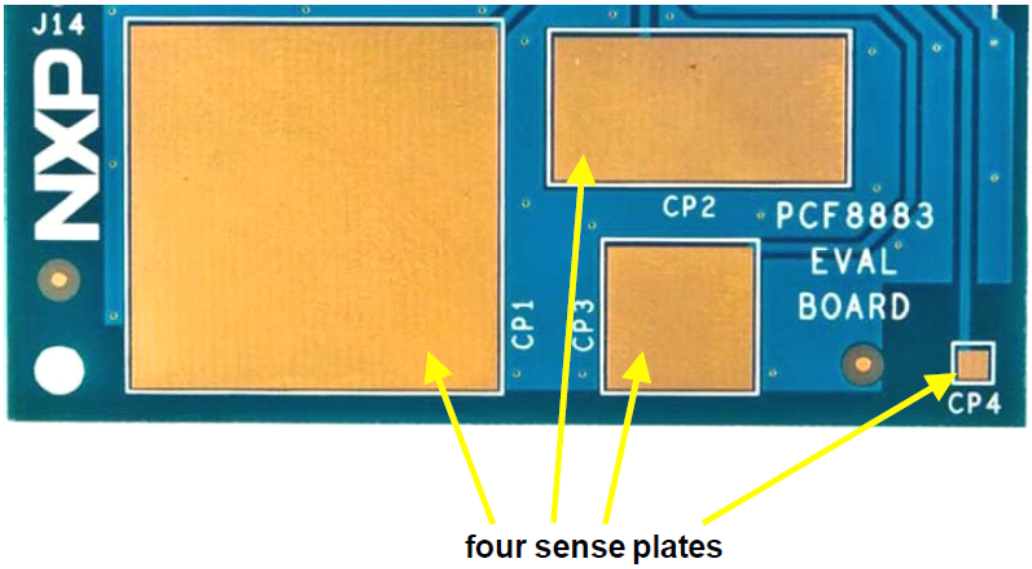

图 1

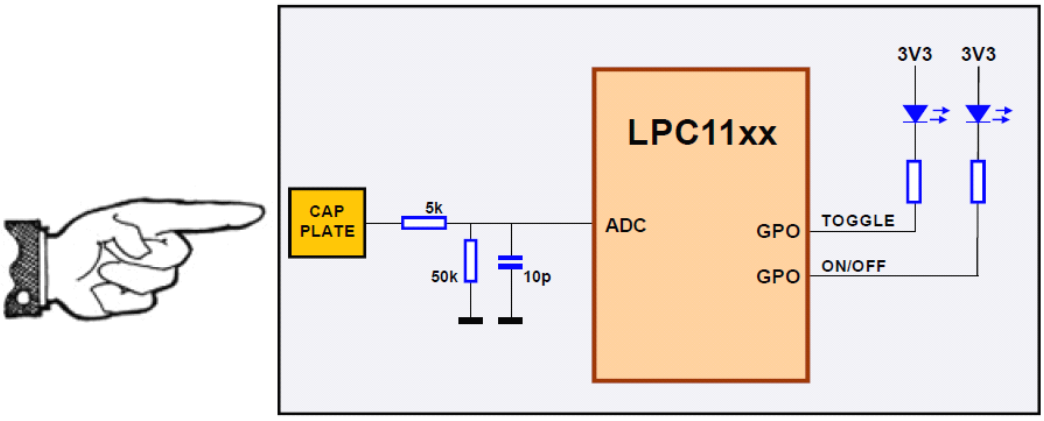

图 2

2. 工作方式

传感只需要一个 PIN 脚，ADC 输入端或者通用输出端。读的过程只有几个简单的步骤（见图 3）。

首先，把 I/0 pin 脚置高输出状态（high output），向外部的 10pF 电容和电容板（capactive plate）充电。

然后，I/O pin 脚重置为 ADC 输入状态（ADC input），这将引发外部电容和电容板通过两个电阻放电（de-charge）。例子中使用的电阻为 5K-50K。当手指触摸传感器时，总电容增加，因此放电弧度变小。

之后，ADC 转换器启动。手指触摸导致 ADC 读取量增加。在示例代码中，在无触摸时有一个稳定的平均值，触摸后产生一个能被检测到的误差值。

最后，I/O pin 重新置为“高”输出状态（”high” output），回到第一步。

传感步骤（见图 3）：

1. 驱动传感器置于 VDD，作为数字输出
2. 传感器转换为 ADC 输入，并启动 ADC 转换器。
3. ADC 采样点。采样并保持一个 ADC 时钟，之后需要至少 10 个时钟来完成 10 位 ADC 转换（完成置位，结果保存在寄存器 LPC_ADC->DR[x]中）。
4. 回到步骤 1。

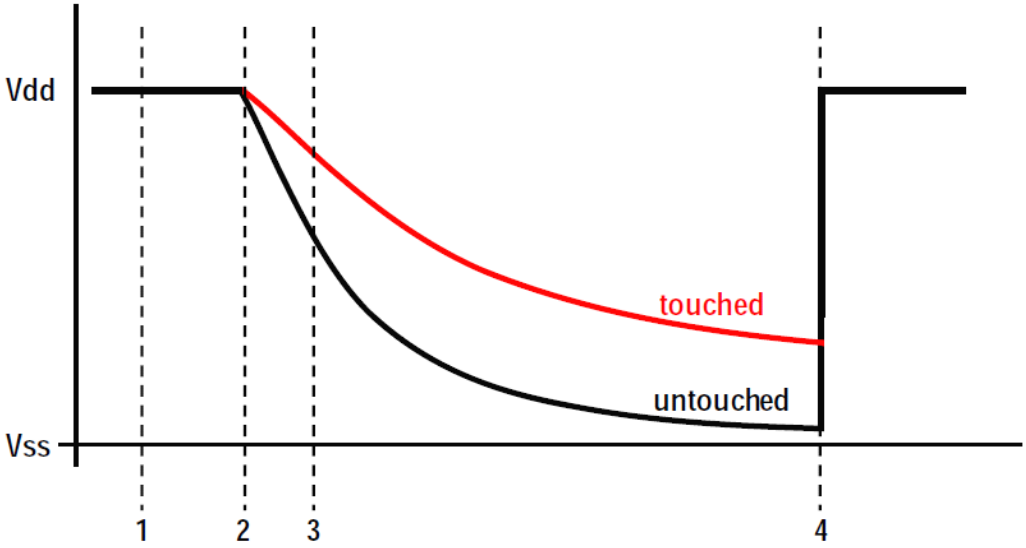

图 3

3. Demo

下面的 LPC1100 演示代码使用 ADC 输入端(PIO1_0)作为传感输入。此外，有两个输出连接到 LED 上，可以方便的观察到传感器的转换过程。

一个输出（PIO3_2）指示触摸状态（触摸，无触摸）

另一个输出（PIO3_3）指示瞬时转换模式。只要触摸事件持续，输出就会激活，LED 点亮。

这个软件示例是用 C语言编写，并使用Keil's uVision (MDKARM,V4.14)编译器编译。LPC11XX使用标准CMSIS启动代码（**startup_LPC11xx.s** and **system_LPC11xx.c**），并设置 CCLK = IRC = 12 MHz。

源代码清单：

```
1
/*****************************************************************************
***
2 * Title : LPC11xx Capacitive Touch Sensing demo program
3 * Hardware : MicroCore48 board + PCF8883 evaluation kit
4 *
5 * 1. Use Systick timer to generate a 10 msec timer tick (interrupt driven).
6 * 2. Capacitive sense plate is connected to P1.0 = AD1 input
7 * 3. Every 10 msec: use ADC to read the capacitive sense input
8 * 4. P3.3 = ON/OFF LED used to indicate press and release condition
9 * 5. P3.2 = toggle LED is used to indicate a new press condition
10
*****************************************************************************
***/
11 #include <LPC11xx.h> // LPC11xx definitions
12
13 static char k_press = 0;
14 static int average = 0;
15
16 static short ADC_ReadCH1(void) // read ADC channel AD1
17 {
18 LPC_IOCON->R_PIO1_0 = 2; // set sensor line as AD1 input
19 LPC_ADC->CR = 2 | // SEL = 2, select channel 1 on ADC
20 (3 << 8) | // ADC_CLK = Fpclk / CLKDIV = 4 MHz
21 (1 << 24); // start conversion
```

```
22
23 while (!(LPC_ADC->DR[1] & 0x80000000)) ; // wait until end of AD1 conversion
24
25 LPC_IOCON->R_PIO1_0 = 0x81; // sensor line is output high
26 LPC_ADC->CR &= 0xF8FFFFFF; // stop ADC
27 return (LPC_ADC->DR[1] >> 6) & 0x3FF; // return A/D conversion value
28 }
29
30 void SysTick_Handler(void) // SysTick Timer ISR every 10 msec
31 {
32 static char debounce = 0;
33 static char avgindex = 0;
34 char result = 0;
35 short reading;
36 reading = ADC_ReadCH1(); // read AD1 = Cap sense input
37
38 if (reading > average + (average >> 4)) // above (average + 6% of average)?
39 {
40 if (debounce == 4) // debounce, 4 triggers for press
41 {
42 k_press = 1; // reached max, indicate pressed
43 result = 1; // set result for return value
44 }
45 else
46 debounce ++; // still going toward max
47 }
48 else if (reading < average + (average >> 5)) // below (average + 3% of average)?
49 {
50 if (debounce == 0)
51 {
52 k_press = 0; // reached min, indicate release
53 result = 0; // clear result for return value
54 }
55 else
56 debounce --; // going toward min
57 }
58
59 if (result == 0 && debounce == 0) // recalculate average
60 {
61 if (++avgindex == 8) // average index delay
62 {
63 average = (reading + (15 * average)) / 16;
64 avgindex = 0;
65 }
```

```
66 }
67 }
68
69 int main (void)
70 {
71 static char toggle = 0;
72 static char ledon = 0;
73 short i;
74
75 SystemInit();
76 LPC_GPIO1->DIR |= (1<<0); // P1.0 connected to cap sense plate
77 LPC_GPIO3->DIR |= (1<<2); // P3.2 = toggle LED
78 LPC_GPIO3->DIR |= (1<<3); // P3.3 = ON/OFF LED
79
80 LPC_SYSCON->PDRUNCFG &= ~(1<<4); // disable pd bit to the ADC block
81 LPC_SYSCON->SYSAHBCLKCTRL |= (1<<13); // enable AHB clock to the ADC
82
83 for (i=0; i<200; i++) // warm up, establish average
84 {
85 average = (32 + ADC_ReadCH1() + (15 * average)) / 16;
86 }
87 SysTick_Config(SystemCoreClock/100); // generate interrupt each 10 ms
88
89 while (1)
90 {
91 __wfi(); // go to sleep
92
93 if (k_press) // key pressed ?
94 {
95 LPC_GPIO3->DATA &= ~(1<<3); // P3.3 low = LED ON
96 if (!toggle)
97 {
98 toggle = 1;
99 if (!ledon)
100 {
101 ledon = 1;
102 LPC_GPIO3->DATA &= ~(1<<2); // P3.2 low = LED ON
103 }
104 else
105 {
106 ledon = 0;
107 LPC_GPIO3->DATA |= (1<<2); // P3.2 high = LED OFF
108 }
109 }
```

```
110 }
111 else // key released
112 {
113 LPC_GPIO3->DATA |= (1<<3); // P3.3 high = LED OFF
114 if (toggle)
115 {
116 toggle = 0;
117 }
118 }
119 }
120 }
```