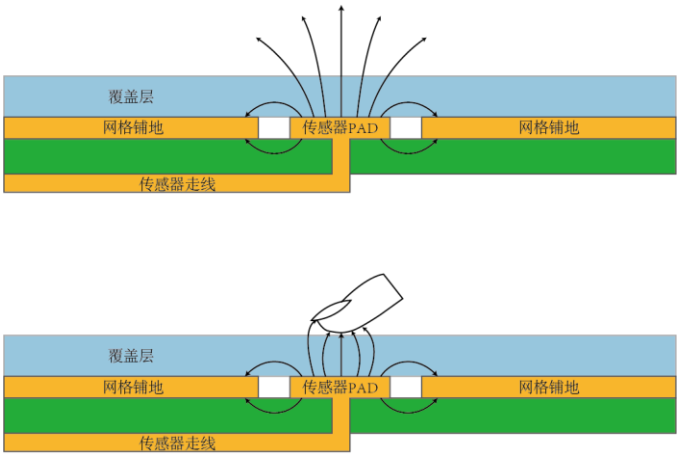
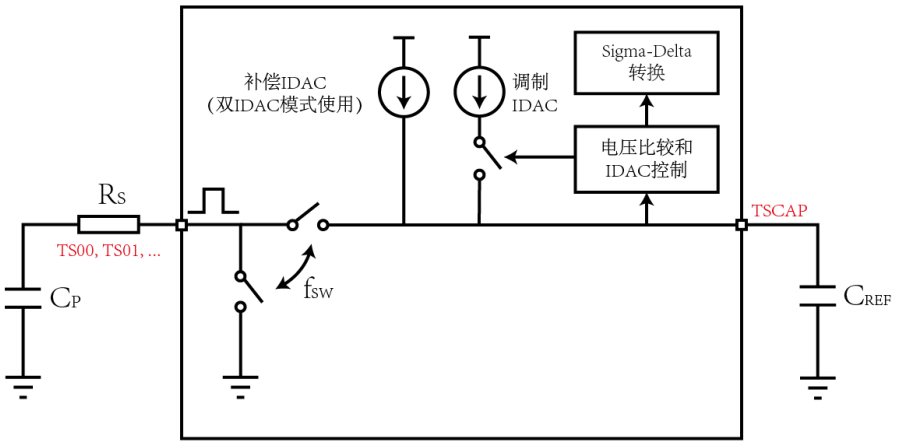
TSI 触摸检测原理

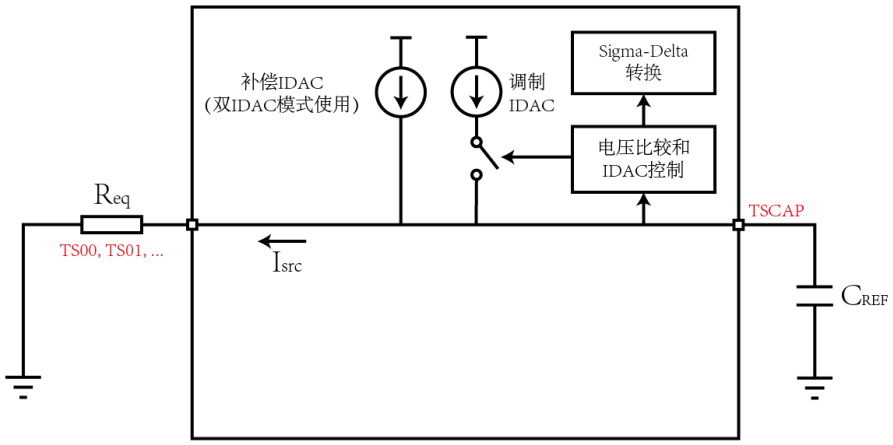
TSI 模块使用自电容的方法来检测触摸行为。自电容检测的原理如下图所示，当传感器 PAD 处于未被触摸状态的时候，传感器 PAD 和走线的电场仅能耦合到网格铺地上，形成传感器的静态电容C_S。而在有手指触摸的情况下，传感器 PAD 和手指之间就通过覆盖层形成了一个对地的电容C_F，这使得传感器 PAD 的电容值变大。因此，TSI 模块通过检测传感器的电容值的变化，可以检测到触摸行为。



TSI 模块内部使用 Sigma-Delta 方法检测传感器电容值的变化。模块首先通过采样时钟对内部开关施加频率为f_{SW}的开关信号，驱动传感器等效电容C_P进行充放电（需要注意的是，我们给的f_{SW}开关信号至少要保证能够满充满放）。充放电的电源来自于 TSCAP 引脚挂载的储能电容C_{REF}。同时，电压比较电路通过采样储能电容上的电压对 IDAC 进行闭环控制，将储能电容上的电压始终稳定在一个固定的参考电压上。简单示意图如下：



C_P每满充满放一次，相当于从储能电容C_{REF}上抽走一个C_P可以储存的电荷量。由于内部电路会控制C_{REF}上电压基本不变，我们可以将传感器电路和开关充放电路等效为从C_{REF}接一个电阻对地进行放电，如下图所示：



等效电阻R_{eq}计算如下：

$$\frac{1}{f_{SW}} \cdot \frac{V_{ref}}{R_{eq}} = V_{ref} \cdot C_S$$

$$\Rightarrow R_{eq} = \frac{1}{C_S \cdot f_{SW}}$$

Sigma-Delta 方法通过累计一个通道扫描周期内，调制 IDAC 打开的时间占整个周期的时间的比例来反映等效R_{eq}抽取电流的量。TSI 模块使用计数器来量化这个比例，可以通过分辨率 **N** 配置来改变量化的粒度。最终我们从 TSI 模块获得的原始计数值 **RawCount** 可以按照如下计算获得：

1. **单 IDAC 模式下，**

$$\frac{t_{IDAC}}{T} \cdot I_{MOD} = \frac{V_{ref}}{R_{eq}}$$

$$\Rightarrow \frac{RawCount}{2^N - 1} \cdot I_{MOD} = \frac{V_{ref}}{\frac{1}{C_S \cdot f_{SW}}}$$

$$\Rightarrow RawCount = (2^N - 1) \cdot \frac{V_{ref} \cdot C_S \cdot f_{SW}}{I_{MOD}}$$

2. **双 IDAC 模式下，**补偿 IDAC 用于降低传感器线路自身静态电容在 RawCount 中引入的无效计数值，扩大 RawCount 的有效范围，从而在相同分辨率下，可以调节其他参数获得更大的精度：

$$\frac{t_{IDAC}}{T}\cdot I_{MOD}+I_{COMP}=\frac{V_{ref}}{R_{eq}}$$

$$\Rightarrow \frac{RawCount}{2^N-1}\cdot I_{MOD}+I_{COMP}=\frac{V_{ref}}{\frac{1}{C_S\cdot f_{SW}}}$$

$$\Rightarrow RawCount=(2^N-1)\cdot\frac{V_{ref}\cdot C_S\cdot f_{SW}}{I_{MOD}}-(2^N-1)\cdot\frac{I_{COMP}}{I_{MOD}}$$

可以看出，RawCount 数值和传感器线路的总电容值成一次函数关系，因此我们可以直接用它来衡量传感器的电容值大小。

RawCount 滤波器

均值滤波器

均值滤波是使用最为广泛且简单有效的滤波方法，能够滤除周期性的噪声。TSI 库使用的均值滤波器固定为使用连续 4 个采样值求取平均值作为输出：

$$y(i)=\frac{1}{4}\left(x(i-3)+x(i-2)+x(i-1)+x(i)\right)$$

中值滤波器

TSI 库提供了一个中值滤波器来抑制毛刺。它使用连续 3 个采样值，取中位数作为输出：

$$y(i)=mid(x(i-2),\ x(i-1),\ x(i))$$

IIR 滤波器

TSI 库提供了一个一阶 IIR 滤波器用于滤除高频噪声：

$$y(i)=\frac{N\times x(i)+(256-N)\times y(i-1)}{K}$$

其中 N 为我们需要调节的 **IIR 基线系数**。N 越大，基线跟随输入就越快。注意这里 N 需要在**[1, 127]**的区间内。

需要注意的是，IIR 滤波器在提供较好的滤波效果的时候，会比较显著地提高响应时间，一般应用场景下，均值滤波器和中值滤波器已经能很好地满足需求。

高级二段式 IIR 滤波器（ADVIIR）

高级二段式 IIR 滤波器（ADVIIR）是专门设计用于接近感应数据滤波的滤波器。由于接近感应产生的 RawCount 值变化十分小，为了尽可能地增加检测的灵敏度，ADVIIR 实际包含了 2 个可以互相切换的滤波器，并设置了 2 个阈值：**激活阈值**和**检测阈值**，用于两个滤波器间的互相切换。

- 算法初始化的时候，ADVIIR 处于检测接近状态，使用滤波性能更好的滤波器（**性能滤波器**）。这使得噪声能够被更好的抑制住，不会产生误触发，且对手指处于较远位置产生的较小接近信号具有比较好的辨识度，其缺点是响应时延相对较大一些；

- 当 RawCount 值上升到激活阈值之上后，ADVIR 滤波器切换为响应时延较小的滤波器（**快速滤波器**）。其滤波性能虽然较弱一些，但可以加速对手指接近的响应，让应用可以更快产生反应，用户体验较好；

- 当 RawCount 值回落到检测阈值之下后，滤波器切换回**性能滤波器**；

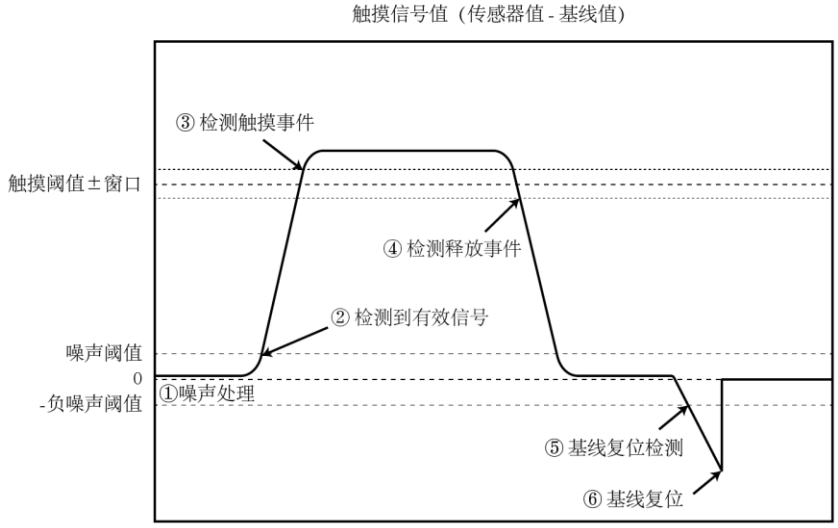
TSI 触摸检测算法

接下来，我们来详细介绍一下复旦微触摸软件库中使用的触摸检测算法。

由于触摸检测实际上是检测管脚上的总电容值，其静态数值在不同线路板、不同安装方式下都会发生变化，且随着环境变化产生漂移，没有办法使用预先计算好的数值作为基准来检测触摸事件。因此，算法需要一个动态跟踪传感器静态数值的基准线，基于它再去计算传感器的实际信号值，并通过设置的阈值来检测触摸事件。复旦微提供的触摸软件库总共实现了 2 种基准线算法：多状态切换的基准线算法和基于一阶 IIR 滤波器的基准线算法。

多状态切换的基准线算法

在每个扫描周期，传感器数值减去当前基准线数值，可以得到当前触摸信号值。算法使用正负噪声阈值、触摸阈值来划分信号强度范围，并根据触摸信号值所在区域，执行不同的基准线更新策略。下面我们介绍一下算法的具体流程。*请注意，在以下讨论中，阈值均使用正整数。*



① **触摸信号值处于区间[-负噪声阈值,噪声阈值]中**：此时的小信号我们作为噪声处理，为了应对随时间和环境变化的飘移，需要执行*水桶算法*。该算法是通过持续累加输入值（这里是触摸信号值），当其超过了水桶容量后，将输出（这里是基准线值）加一，并清空水桶，从而达到让输出持续缓慢追踪输入的效果。我们的算法对正噪声和负噪声采用了分别处理的方式，因此需要 2 套参数：**噪声阈值**和**噪声水桶容量**，**负噪声阈值**和**负噪声水桶容量**，分别控制信号值位于区间**(0, 噪声阈值]**和区间**[-负噪声阈值, 0)** 的情况。

② **触摸信号值处于区间[噪声阈值, 触摸阈值] 中**：这代表我们检测到了有效信号。如果我们没有配置**基准线始终更新**选项，那么此时基准线数值将被锁定，等待触摸事件触发；如果配置了**基准线始终更新**选项，那么我们继续使用水桶算法更新基准线值。为了能够更方便控制此时基准线追踪的速度，我们还额外引入了**误差递增量**参数来控制水桶满后基准线增加的数值。

注意：*一般情况下，只有在传感器静态数值经常发生大幅度变化时才使用***基准线始终更新***选项。该选项使得触摸传感器不会因为静态数值大幅变化而被长时间持续误触发。*

③ **触摸信号值上升到大于触摸阈值+触摸窗口**：此时启动防抖计数器，在连续 N_{ACT} 个计数后产生触摸事件，N_{ACT} 由**触发防抖计数**参数配置。

④ **触摸信号值下降到小于触摸阈值-触摸窗口**：此时启动防抖计数器，在连续 N_{RST} 个计数后产生触摸释放事件，N_{RST} 由**复位防抖计数**参数配置。程序逻辑则在触摸信号值小于等于触摸阈值时回到②执行。

⑤ **触摸信号值下降到小于-负噪声阈值**：可能是强 ESD 或者 RF 或者其他原因（比如上电时手指就触摸在传感器表面，之后手指离开）导致的。此时我们启动基线复位计数器，当触摸信号**连续基线复位计数**个扫描周期低于**-负噪声阈值**，基准线会复位到传感器数值处。

基于一阶 IIR 滤波器的基准线算法

由于设置基准线的目标是要追踪传感器静态情况下的数值变化，另外一种办法是采用一个时间系数较大的 IIR 滤波器来进行。这种方法虽然没有前述多状态切换的基准线算法控制的细致，但是却能够大幅降低我们需要调节的参数，并且也能够达到类似的效果，因此我们还是**推荐使用此算法作为基准线算法**，比较省时省心。

软件库内置 IIR 滤波器使用下式：

$$y(i)=\frac{N\times x(i)+(256-N)\times y(i-1)}{K}$$

其中 N 为我们需要调节的 **IIR 基线系数**。N 越大，基线跟随输入就越快。注意这里 N 需要在**[1, 127]**的区间内。

下面我们来介绍一下使用 IIR 基准线算法时的程序逻辑。

① **触摸信号值小于噪声阈值：**我们使用一阶 IIR 滤波器对传感器数值进行滤波得到基准线值。

② **触摸信号值处于区间[噪声阈值，触摸阈值] 中：**这代表我们检测到了有效信号。如果我们没有配置**基准线始终更新**选项，那么此时基准线数值将被锁定，等待触摸事件触发；如果配置了**基准线始终更新**选项，那么我们继续使用一阶 IIR 滤波器计算基准线。

③ **触摸信号值上升到大于触摸阈值+触摸窗口：**此时启动防抖计数器，在连续 N_{ACT} 个计数后产生触摸事件，N_{ACT} 由**触发防抖计数**参数配置。

④ **触摸信号值下降到小于触摸阈值-触摸窗口：**此时启动防抖计数器，在连续 N_{RST} 个计数后产生触摸释放事件，N_{RST} 由**复位防抖计数**参数配置。程序逻辑则在触摸信号值小于等于触摸阈值时回到②执行。