Shanghai Fudan Microelectronics Group Company Limited

复旦微电子

TSI 设计指南

V1.3.0

本资料是为了让用户根据用途选择合适的上海复旦微电子集团股份有限公司（以下简称复旦微电子）的产品而提供的参考资料，不转让属于复旦微电子或者第三者所有的知识产权以及其他权利的许可。

在使用本资料所记载的信息最终做出有关信息和产品是否适用的判断前，请您务必将所有信息作为一个整体系统来进行评价。

采购方对于选择与使用本文描述的复旦微电子的产品和服务全权负责，复旦微电子不承担采购方选择与使用本文描述的产品和服务的责任。除非以书面形式明确地认可，复旦微电子的产品不推荐、不授权、不担保用于包括军事、航空、航天、救生及生命维持系统在内的，由于失效或故障可能导致人身伤亡、严重的财产或环境损失的产品或系统中。

今后日常的产品更新会在适当的时候发布，恕不另行通知。在购买本资料所记载的产品时，请预先向复旦微电子在当地的销售办事处确认最新信息，并请您通过各种方式关注复旦微电子公布的信息，包括复旦微电子的网站(http://www.fmsh.com/)。

如果您需要了解有关本资料所记载的信息或产品的详情，请与上海复旦微电子集团股份有限公司在当地的销售办事处联系。

商标

目录

1. 引言

1.1 摘要

TSI 设计指南指导用户如何在复旦微系列 MCU 中进行电容触摸传感的开发。MCU 支持多种传感器，如按钮、滑块以及接近传感器等。本指南将从 TSI 触摸检测原理、硬件设计、软件设计、上位机 TSI TUNER 使用指南以及测试注意事项等方面进行详细介绍。

1.2 电容感应特点

FM33FT0xxA/FM33FR0xx 中电容感应有以下特点：

1.支持基于自电容的触摸传感。

2.稳定的 Sigma-Delta 传感技术，为触摸传感提供了较高的信噪比。

3.支持各种覆盖材料和不同厚度的触摸传感。

4.自动调优算法。

5.支持防液体误触功能。

6.伪随机系列（Pseudo random sequence,PRS）的时钟源可有效抑制电磁干扰（electromagnetic interference）.

7.多通道支持电容感应，可满足大多实际应用。

FM33HT0xxA/FM33FH0xx 相比 FM33FT0xxA/FM33FR0xx 系列中电容感应有以下特点：

1.同时支持基于自电容以及互电容的触摸传感。

2.多频点测量方法可有效抑制电磁干扰（electromagnetic interference）.

3.多通道支持电容感应以及屏蔽电极使用，可满足大多实际应用。

4.针对不同面积的屏蔽电极设计不同的驱动方式。

1.3 设计流程

下图 1.1 为基于复旦微系列 FM33FT0xxA/FM33FR0xx MCU 设计的通用流程图。

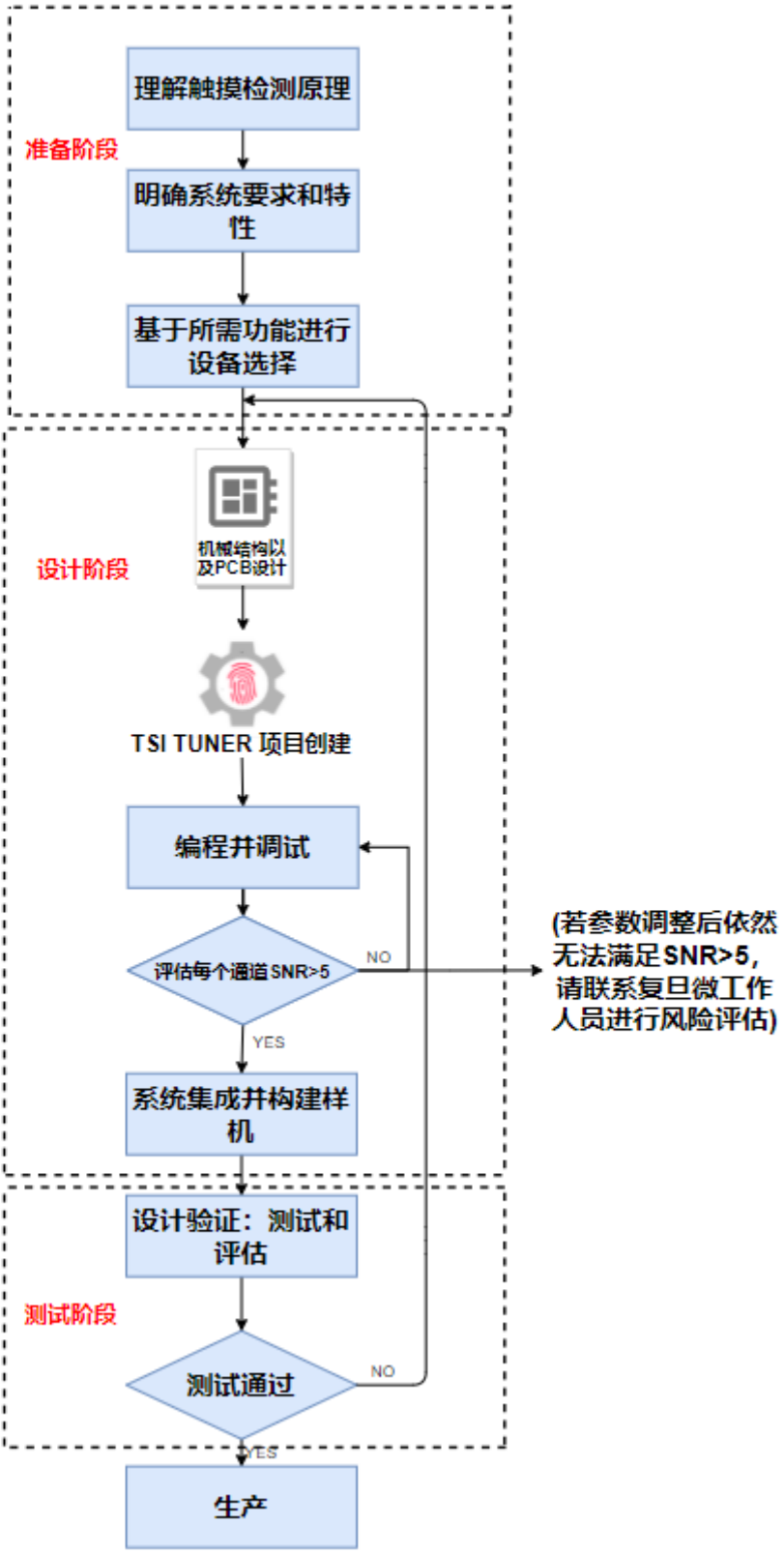

准备阶段
理解触摸检测原理
明确系统要求和特性
基于所需功能进行设备选择
设计阶段
机械结构以及PCB设计
TSI TUNER 项目创建
编程并调试
评估每个通道SNR>5
NO
(若参数调整后依然无法满足SNR>5，请联系复旦微工作人员进行风险评估)
YES
系统集成并构建样机
设计验证：测试和评估
测试阶段
测试通过
NO
YES
生产

图 1.1 设计流程

1.4 相关支持文件

设计流程	支持文档
理解触摸检测原理	TSI 触摸检测原理和算法
可行性研究（基于所需功能进行设备选择）	FM33FT0xxA 系列说明书 FM33FR0xx 系列说明书 FM33FH0xx 系列说明书 FM33HT0xxA 系列说明书
机械结构及 PCB 设计	TSI 硬件注意事项
TSI TUNER 项目创建	TSI Tuner 快速入门指南
编程并调试	TSI 软件库介绍 TSI 硬件参数调节过程
量产前建议	TSI 量产前建议测试流程

2. TSI 触摸检测原理

2.1 自电容感应

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。

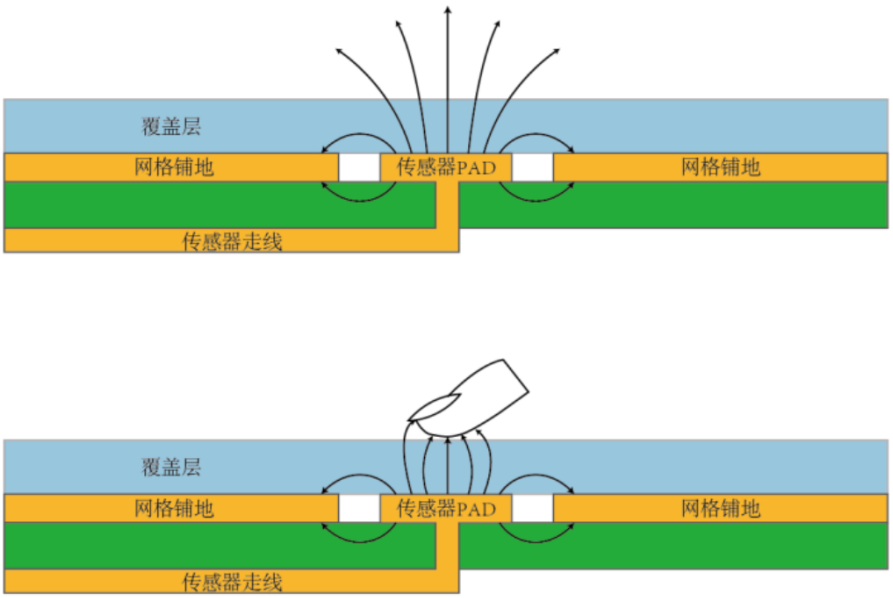


图 2.1 自电容感应

2.2 互电容感应

TSI 模块使用互电容的方法来检测触摸行为。互电容检测的原理如下图所示，互电容传感测量两个电极之间的电容，这两个电极被称为发射（Tx）和接收（Rx）电极。在互电容检测系统中，在 VDD 和 GND 之间切换的数字电压信号被施加到 Tx 引脚，并且在 Rx 引脚上接收的电荷量被测量。在 Rx 电极上接收的电荷量与两个电极之间的互电容（C_M）成正比。当手指放置在 Tx 和 Rx 电极之间时，互电容减小。由于互电容的减小，在 Rx 电极上接收的电荷也减小。TSI 模块测量在 Rx 电极上接收的电荷量，以检测触摸/无触摸情况。

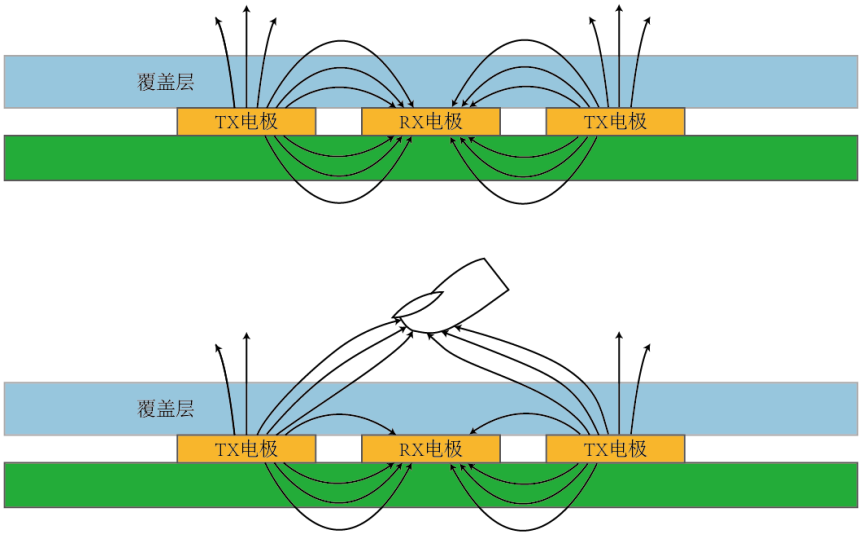


图 2.2 互电容感应

2.3 电容式触摸感应方法

2.3.1 自电容感应方法

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：

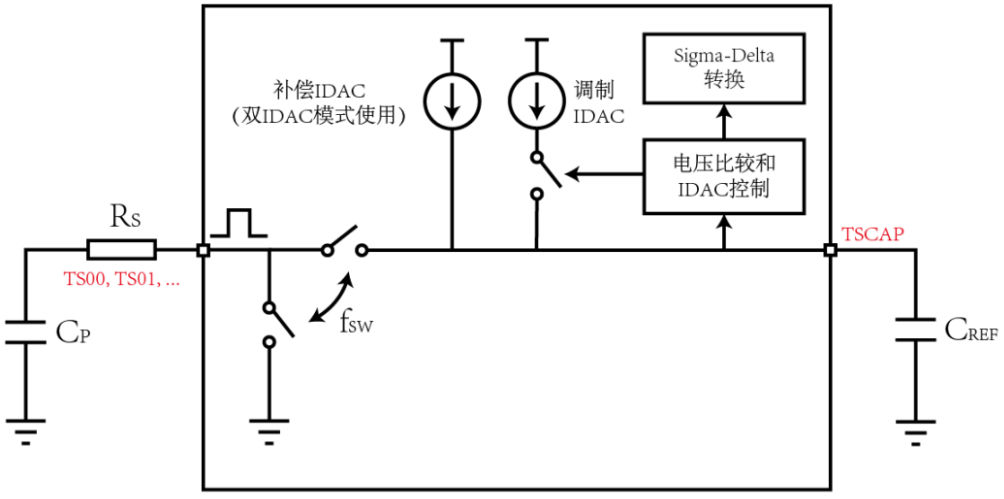


图 2.2 Sigma-Delta 示意图

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：

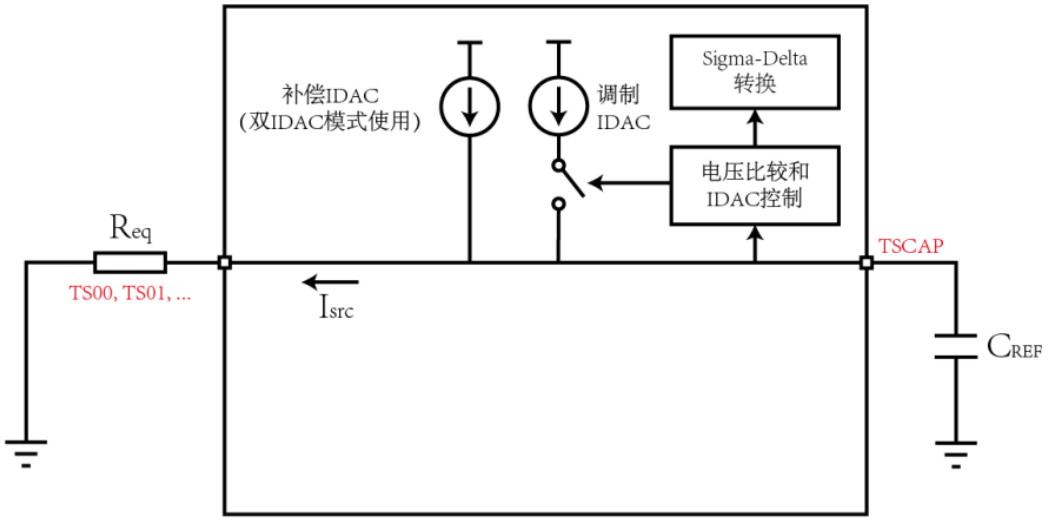


图 2.3 Sigma-Delta 等效图

等效电阻计算如下：

$$\frac{1}{f_{sw}} \bullet \frac{V_{ref}}{R_{eq}} = V_{ref} \bullet C_S$$

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

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,其中单 IDAC 模式下计算公式如下：

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

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

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

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：计算公式如下：

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

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

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

可以看出， RawCount 数值和传感器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。

2.3.2 互电容感应方法

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，发射节点并发射特定频率的方波，对传感器互容节点 C_M 充放电,接收节点通过 IDAC 吸收或补充电荷，使得总线电压稳定到基本电压上，示意图如下：

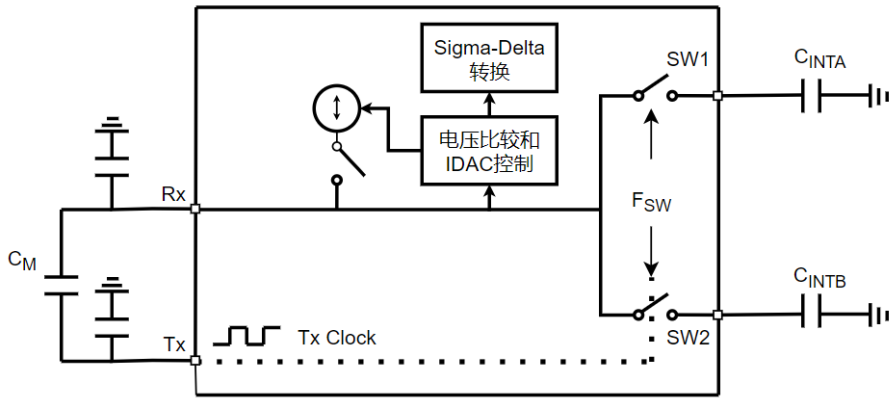


图 2.4 Sigma-Delta 互容等效图

图 2-5 显示的是 Tx 电极、 CINTA 以及 CINTB 电容上的波形。一个采样中所有子转换的总和作为结果，并被称为“原始计数”。一个子转换是用于计算在一个 Tx 时钟周期中执行的转换次数的电容值。在一个子转换过程中，SW1 和 SW2 的开关将在与 Tx 时钟的同一个相位进行。在 Tx 时钟的上升沿上，SW1 处于关闭状态（在这段时间内， SW2 处于打开状态），并且电荷从 Tx 电极流动到 Rx 电极。将该电荷集成到 CINTA 电容上，从而使 CINTA 上的电压递增。将 IDAC 配置为灌电流模式，使 CINTA 上的电容等于 VREF 电压。在 Tx

时钟的下降沿上， SW2 处于关闭状态（在该时间内， SW1 处于打开状态），并且电荷从 Rx 电极流动到 Tx 电极。这样会使 CINTB 上的电压小于 VREF。将 IDAC 配置为源电流模式，以使 CINTB 上的电压等于 VREF。在两个周期中， Tx 和 Rx 电极间传输的电荷与电极间的互电容 C_M 成正比。 IDAC 对外部电容进行充电或放电时，比较器输出将使能计数器。计数器将计算一个子转换中的调制器时钟周期的数量。执行多个子转换，并将该结果存储在同一个计数器内，从而生成传感器的“原始计数” 。

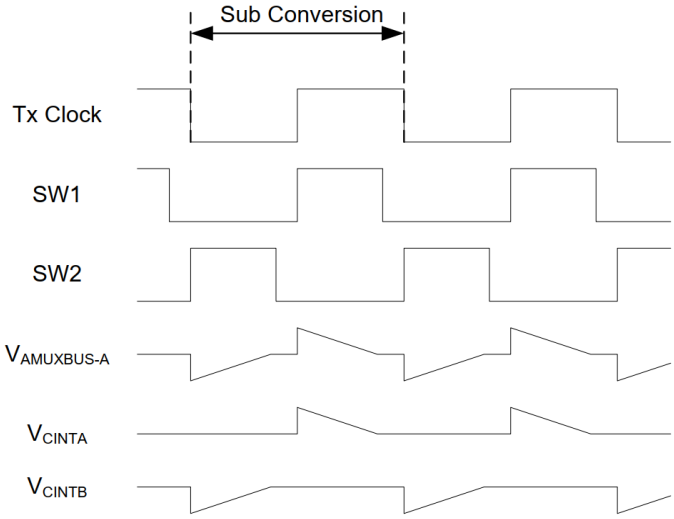


图 2.5 互容扫描波形

通过调制器时钟，可以测量到在一个 Tx 时钟周期内对外部电容进行充电/放电所需的时间。因此，调制器时钟频率必须始终大于 Tx 时钟频率；调制器时钟的频率越高，准确度也越高。为了正常工作，需要设置 IDAC 电流，使 CINTA 和 CINTB 电容能在一个 Tx 时钟周期内完成充电或放电。Rawcount 与互电容之间的对应关系如下：

$$\begin{cases} \text{Rawcount} = \dfrac{2V_{TX}F_{TX}C_M MaxCount}{IDAC} \\ MaxCount = \dfrac{F_{MOD}N_{SUB}}{F_{TX}} \end{cases}$$

可以看出来，Rawcount 数值与传感器的互电容值成正比，当手指放在互容按钮上时，Rx 与 Tx 电极间的互电容减小，Rawcount 则减小。

2.4 电容传感组件

TSI 模块中传感器部件大致分为三类：按钮、滑条和接近传感器，如下图所示。本节将解释不同传感器部件的基本概念。

图 2.6 传感器部件（左至右依次为按钮、滑条和接近感应）

2.4.1 按钮（Buttons）

电容感应式按钮存在两种状态：检测到手指\未检测到手指，分别为 ON 和 OFF。对于自电容的 Sigma-Delta 传感方法，一个简单的传感器由圆形铜箔与 TSI 通道口相连组成，按钮周围被网状铺地所包围，隔离与其他按钮的连接，并每个按钮与网状铺地有一个圆形间隙用于分割按钮与铺地，如下图所示：

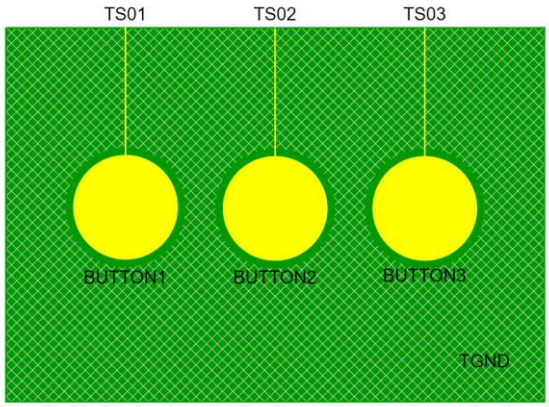


图 2.7 电容感应式按钮示意图

对于互电容方法，每个按键要求使用一个被配置为 Tx 电极的 GPIO 引脚和一个被配置为 Rx 电极的 GPIO 引脚。多个按键可共享 Tx 引脚，如下图所示：

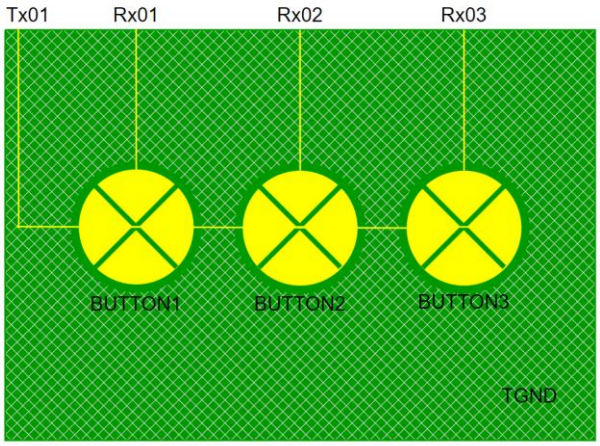


图 2.8 矩阵按钮示意图

2.4.2 滑条（Sliders）

当所需的输入需要以逐步递增或者递减形式出现时，将使用滑条。TSI 模块支持基于自电容的滑条。一段滑条由若干个电容式传感器排列组成，相邻放置，触摸一个片段也会导致相邻触摸段产生计数，通过处理此计数大小以此来判断手指触摸的几何中心位置，即质心位置。

计算出的质心位置的实际分辨率远远高于滑块中的段数，因此当手指滑过时可使得质心位置平滑过渡并定位。

在滑条的设计中，每个滑块段连接一个 TSI 通道口，滑块段建议采用锯齿形图案（双人字形）。这种布局可确保一个滑块段被接触时，相邻的滑块段也被部分解除，有助于更为精确地判断位置，滑条示意图如下所示：

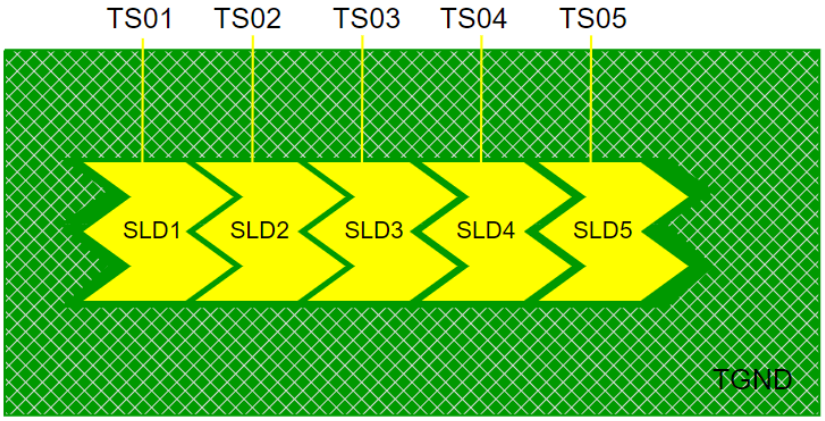


图 2.9 滑条传感器示意图

2.4.3 触摸板（TouchPad）

下图所示为经典的触摸板设计示意图，可做为自电容和互电容的触摸板使用，自电容通过两组滑条的方式实现，互电容通过 Tx 电极与 Rx 电极耦合的方式实现。

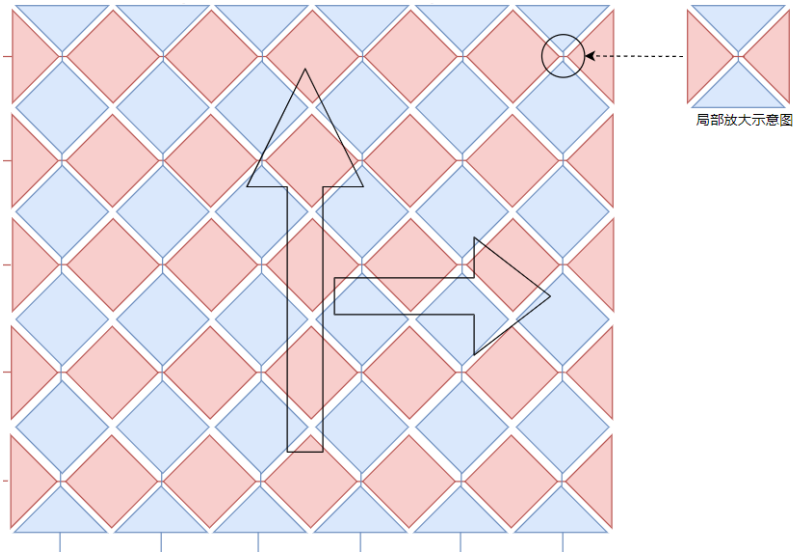


图 2.10 触摸板示意图

2.4.4 接近感应（Proximity）

接近传感器用于检测在传感器周围的三维空间中手的存在，实际输出类似于电容感应式按钮。根据传感器的结构以及 MCU 的驱动能力，接近感应式传感器可以检测 0-5cm 左右的手。TSI 模块支持基于自电容的接近感应传感器。

接近感应传感器需要将电场投射到比按钮和滑条更大的距离，因此也就需要一个很大的传感器区域，然而较大的传感器对应为更大的寄生电容 C_P，检测会存在困难，因此需要一个感应范围大，寄生电容小的传感器，如下图所示：

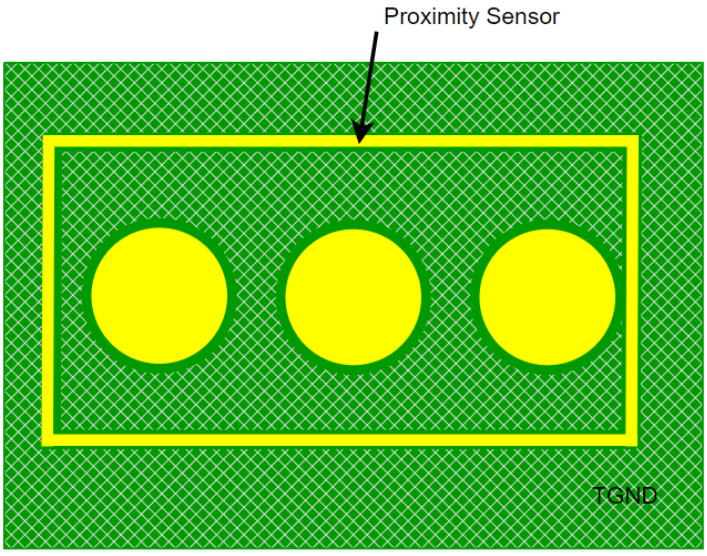


图 2.11 接近感应传感器示意图

2.5 防水性能

为了应对由于水、湿度和温度而引起 RawCount 的变化，TSI 模块不断调整 baseline 以防止误触发，同时为了补偿由于水滴在传感器表面流动的 RawCount 的变化，TSI 模块提供屏蔽电极以及保护传感器功能，当实现屏蔽电极功能且传感器表面存在小水滴，MCU 可准确检测触发状态，当存在液体流动时，保护传感器将检测是否存在流动液体，并不进行扫描。下图为当屏蔽电极功能和保护传感器功能同时使用时的示意图：

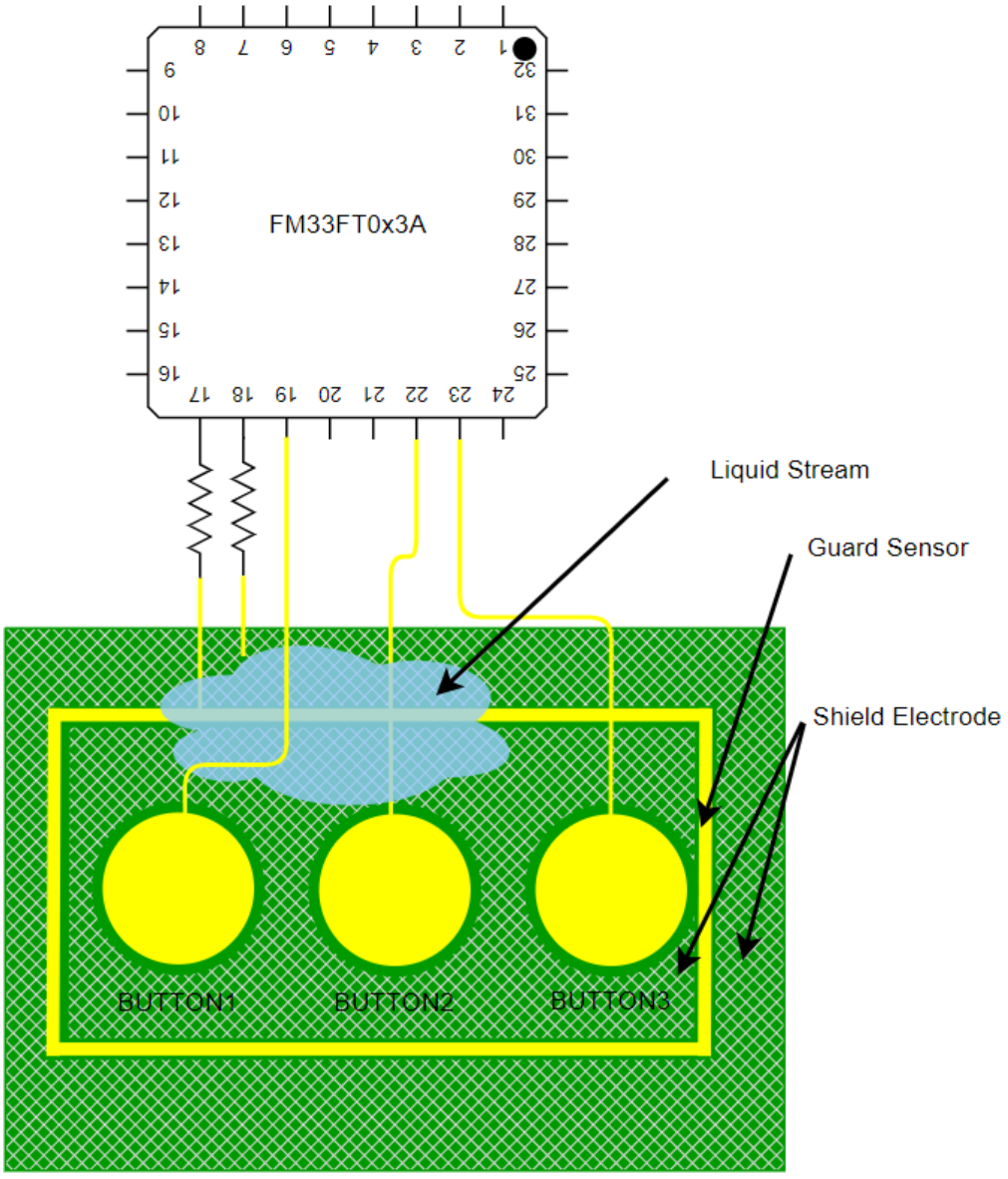


图 2.12 屏蔽电极以及保护传感器示意图

2.5.1 液体对传感器的影响

下图为网状铺铜连接到 TGND 时的按钮设计，与此同时也会提高传感器的抗噪能力，该传感器的寄生电容可以等效为右图：

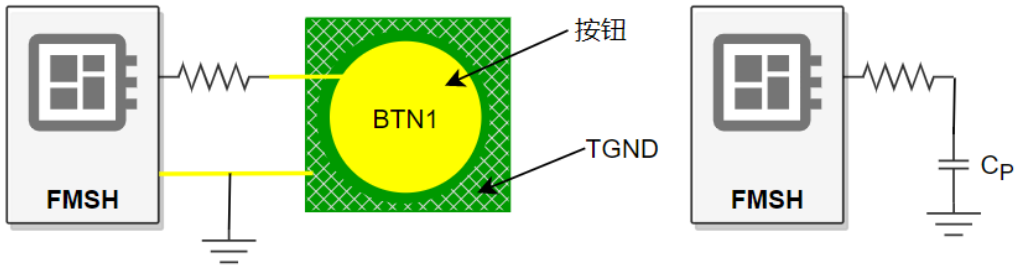


图 2.13 典型电容式感应按钮

当水滴在传感器表面时，由于其导电性，为电场与 TGND 之间提供了强耦合路径，便增加了与 C_P 平行的 C_{LD},增加的电容回吸取额外的电荷，由检测原理可知，RawCount 则会增加，在某些情况水滴引起的 RawCount 增加将大于等于手指触发引起的增加，从而会发生误触。

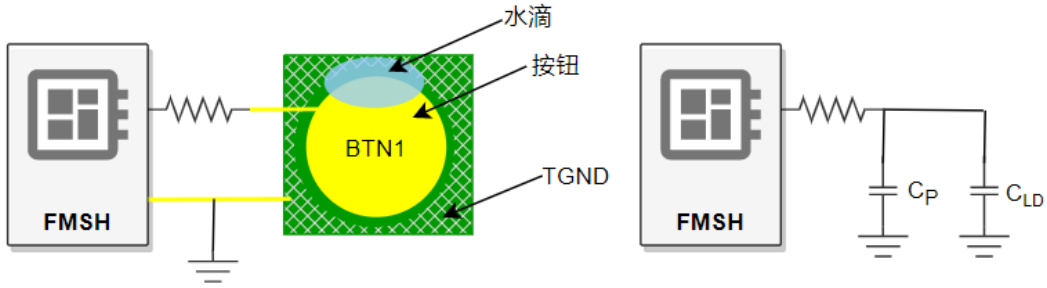


图 2.14 水滴在按钮表面示意图

为了消除水滴对传感器电容的影响，需使用 TSI 模块提供的屏蔽电极功能，将屏蔽信号填充在传感器周围。当水滴存在于传感器表面时，由于屏蔽信号的连接，水滴两侧电压保持在相同电位，因此由水滴增加的电容 C_{LD} 不会产生额外的电荷，因此影响被抵消，不会产生误触。

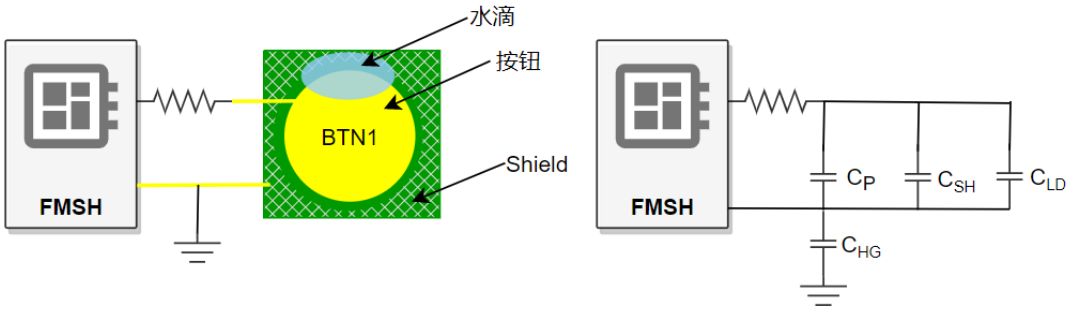


图 2.15 传感器周围铺铜连接至 Shield

2.5.2 屏蔽信号与屏蔽电极

驱动的屏蔽信号是传感器开关信号的缓冲版本，与传感器开关信号振幅、频率和相位相同，缓冲器为屏蔽信号提供足够的电流，用于驱动网状铺铜的高寄生电容。当传感器周围的铺铜连接驱动的屏蔽信号时，则称为屏蔽电极。

屏蔽电极一般有以下用途：

1. 防水滴干扰。
2. 当屏蔽电极放置在接近感应传感器与浮空或者接地的导电物体之间时，可以减少这些物体对接近感应距离的影响，有助于实现较大的接近感应距离。
3. 当接近感应传感器过长或者过大时，传感器电场到周围地的耦合增加，C_P 会非常高，使用屏蔽电极，可减少电场线耦合，从而降低 C_P。

2.5.3 保护传感器

当传感器表面存在流动且连续的水滴时，会使得传感器增大电容（C_{ST}）,这个电容远大于 C_{LD},因此屏蔽电极的效果会被完全掩盖，RawCount 会大于或等于手指触摸时的 RawCount，从而产生误触，在这种情况下，保护传感器有助于防止错误触发。

保护传感器是围绕在所有传感器周围的一圈铜箔，其类似于按钮传感器，用于检测流动液体是否存在，当保护传感器检测到流动液体时将禁止除自己之外其他传感器的扫描，以防止错误触发。

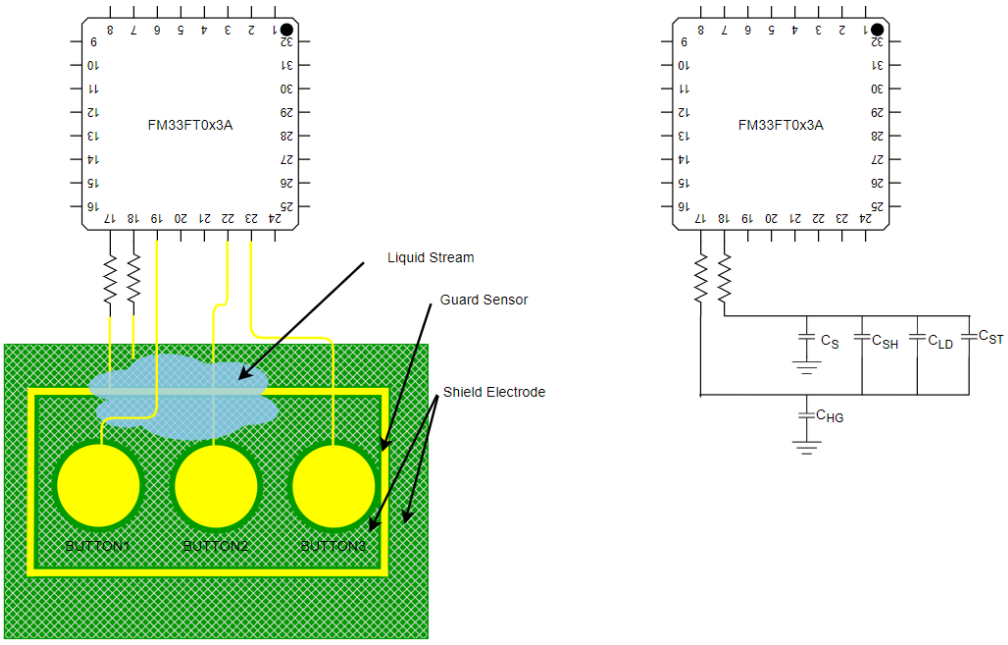


图 2.16 保护传感器

2.5.4 防水设计建议

如果需要设计可以防水请参考以下步骤：

1.如果需要防护水滴，请打开屏蔽电极，如果同时需要防护水滴以及流动液体，请实现屏蔽电极以及保护传感器。

2.在 TSI TUNER 上位机中勾选“使用屏蔽电极”。

3. TSI 传感器硬件设计

3.1 传感器结构

根据应用要求，可以使用不同的材料来构建电容式传感器。在典型的传感器构造中，传感器铜箔连接至 TSI 模块通道，整个结构放置在覆盖层下面，用户在覆盖层顶部进行交互，平面图如下所示：

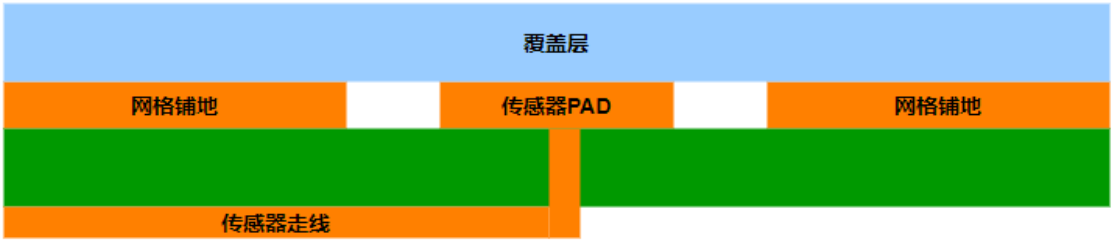


图 3.1 电容式传感器构造

PCB 表面的铜箔为传感器 PAD，非导电的覆盖层作为触摸表面，该覆盖层还可以保护传感器免收环境影响，并防止手指直接接触。

在某些情况下，弹簧可以作为电容感应传感器，形成一个距离增高的传感器，允许覆盖层放置弹簧上面，如下图所示：

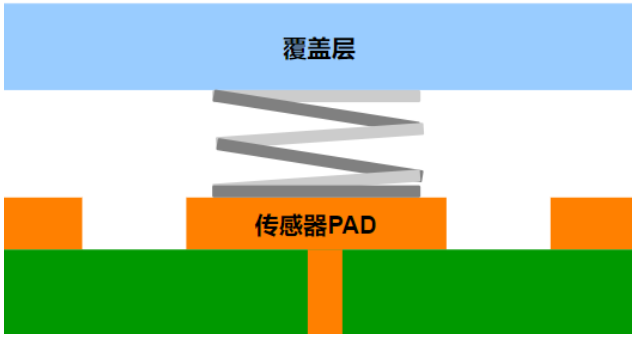


图 3.2 弹簧作为传感器

3.2 覆盖层选择

3.2.1 覆盖层材料

覆盖层是传感器的重要组件，决定手指电容的大小，其电容与覆盖材料的相对介电常数成正比，其相对介电常数在 2.0-8.0 之间比较适用于 TSI 模块，常见材料的相对介电常数如下表：

表 3.1 常见材料的相对介电常数

材料	相对介电常数
空气	1.0
塑料	4.6-4.9
玻璃（标准）	7.6-8.0
玻璃（陶瓷）	6.0
PET Flim	3.2
Polycarbonate	2.9-3.0
Acrylic	2.8
ABS	2.4-4.1
Wood Table and Desktop	1.2-2.5
Gypsum（Drywall）	2.5-6.0

覆盖材料中不建议选择导电材料，会干扰传感器电场。

3.2.2 覆盖层厚度

手指的电容大小与覆盖层厚度成反比，薄的覆盖层相比较于厚的覆盖层能提供更多的信号，在进行厚度选择时，应该结合材料相对介电常数以及灵敏度需求进行覆盖层厚度选择。

3.2.3 覆盖层粘合

覆盖层需要与 PCB 有良好的机械接触，在粘合的时候应该使用非导电性粘合剂进行连接。该粘合剂可通过消除覆盖层与传感器的空气间隙来提高系统灵敏度。

3.3 PCB 设计指南

3.3.1 寄生电容

C_P 的大小主要与走线和传感器电容有关，在以下几种情况下 C_P 会增加：

1、传感器 PAD 尺寸增加。

2、走线长度和宽度增加。

3、传感器 PAD 与网络铺地之间的间隙增加。

传感器铜箔尺寸若过小会影响可感应手指电容的大小，可通过增加传感器 PAD 与网格铺地之间的距离，但同时也会降低其抗噪声性，也可以通过驱动屏蔽电极来降低 CP。

3.3.2 板层

在大多数应用中为两层板，传感器在顶部，其他组件在底部。基于 FR4 的 PCB 设计表现良好，板的厚度为 0.5mm 到 1.6mm。

3.3.3 自电容按钮设计

在自电容按钮设计时，推荐使用圆形的铜箔，带有圆角的矩形形状铜箔是次选，应避免使用尖角（小于 90°），会使电场分布不均匀，示意图如下所示：

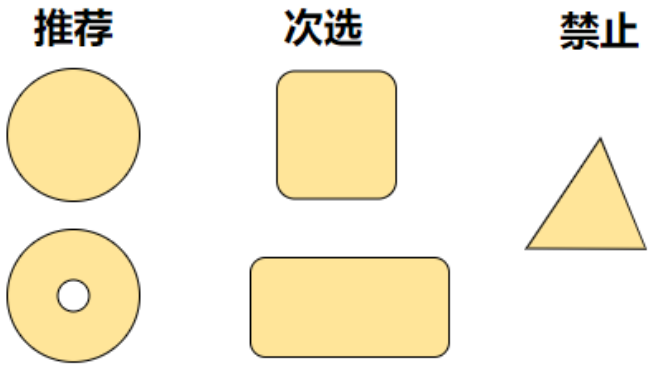


图 3.3 按钮设计图

按钮直径多为 5mm-15mm，10mm 适用于大多数应用，传感器 PAD 与网络铺地之间的距离大约等于覆盖厚度，多为 0.5mm-2mm。

两个不同按钮之间的间距应当保证手指在触摸其中一个按钮时另一个按钮不会产生错误检测。

3.3.3 互电容按钮设计

在互电容按钮设计时，互电容按钮测量两个电极间的电容耦合的变化，针对覆盖层厚度小于 1.5mm 的互容传感器可按如下图的方式进行设计参考：

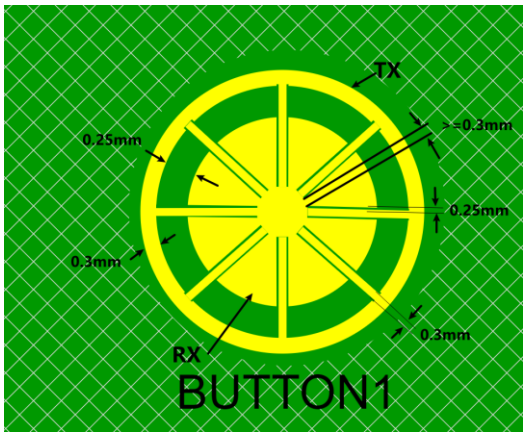


图 3.4 薄覆盖层的互容按钮设计图

针对覆盖层厚度大于 1.5mm 的覆盖层的互电容设计如下：

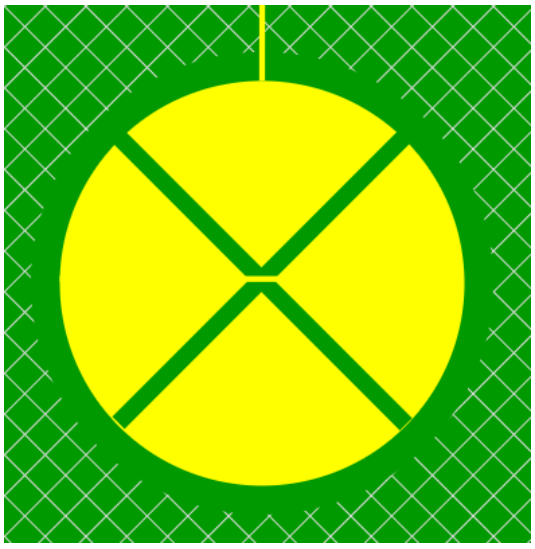


图 3.5 厚覆盖层的互容按钮设计图

最小电极大小要等于覆盖层厚度的 2 倍，确保信号良好。

3.3.4 滑条设计

滑条尺寸设计推荐如下，H 为滑条段的高度推荐 7mm-15mm，宽度 W 推荐 2-6mm，每个滑条段之间的间隙应为 0.5mm-2mm，多为 0.5mm。

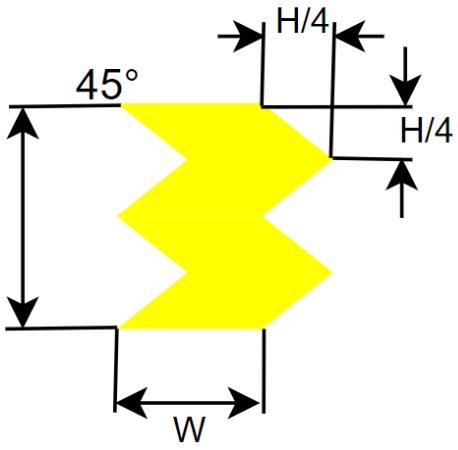


图 3.6 滑条段设计图

3.3.5 触摸板设计

触摸板尺寸设计推荐如下，Pitch 为行/列单通道尺寸，推荐为 4mm-10mm，Separation 为 XY 间隙，推荐为 0.25mm-1mm，行*列至少为 3*3。

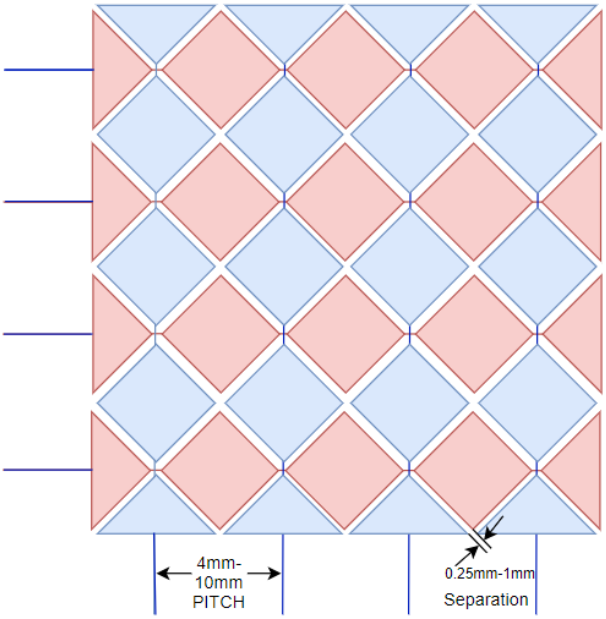


图 3.7 触摸板设计图

PITCH 和 SEPARATION 对应关系如下表：

表 3.2 PITCH\SEPARATION 对应关系

	最小值	典型值	最大值
行/列 PITCH	4mm	6mm	10mm
XY SEPARATION	0.25mm	0.5mm	1mm

3.3.6 布线(适用于 FM33FT0xxA/FM33FR0xx)

PCB 设计可以参考以下注意事项：

1. 按钮直径范围为 5 毫米到 15 毫米，其中 10 毫米适用于大多数应用。大直径按键在覆盖层较厚的情况下表现更好；传感器中心可以挖一个透光孔用于布设背光 LED；如果使用触摸弹簧，弹簧区域内不要有其他走线；
2. 传感器到芯片的走线要在非传感器布置层走线以降低手指到走线的耦合；过孔尽量打在传感器焊盘边沿，降低走线总长度；
3. 传感器到芯片的走线长度尽量短，推荐小于 100mm， 走线宽度推荐 7mil（0.18mm）；
4. 传感器到芯片的走线周围尽量由铺铜包络，到走线间距要保持在 0.5mm 到 2mm 之间，走线在触摸部分的铺铜使用 17%覆盖率（7mil 线宽，70mil 间距）网状铺铜并接地；走线在接近芯片部分的铺铜使用实心铺铜，若为双层板建议在走线背部实心铺铜，若为四层板建议走线在中间层，并将其它层实心铺铜覆盖走线。传感器周围使用 25%覆盖率（7mil 线宽，45mil 间距） 网状铺铜：如果不使用防水功能，将其接地，如果需要使用防水功能，应该连接到 Shield 引脚；

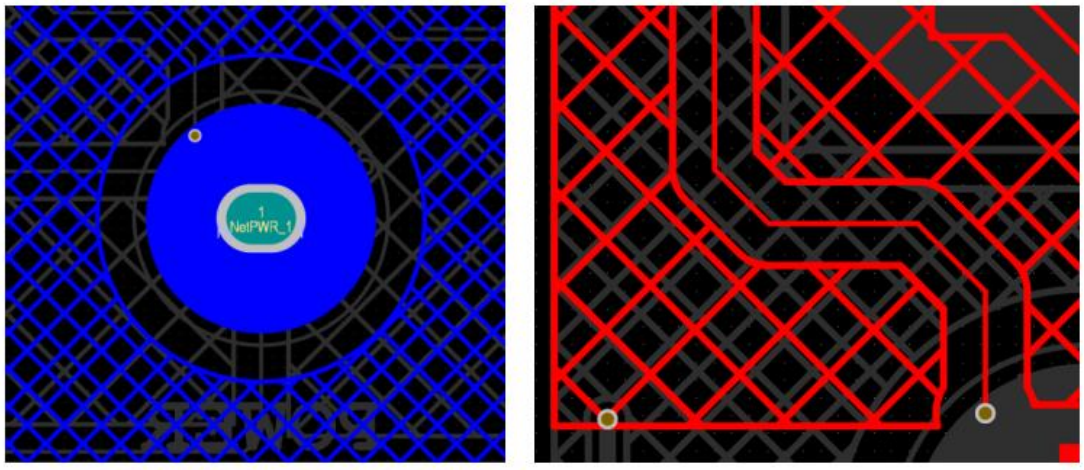


图 3.8 （a）触摸部分走线以及传感器周围铺铜

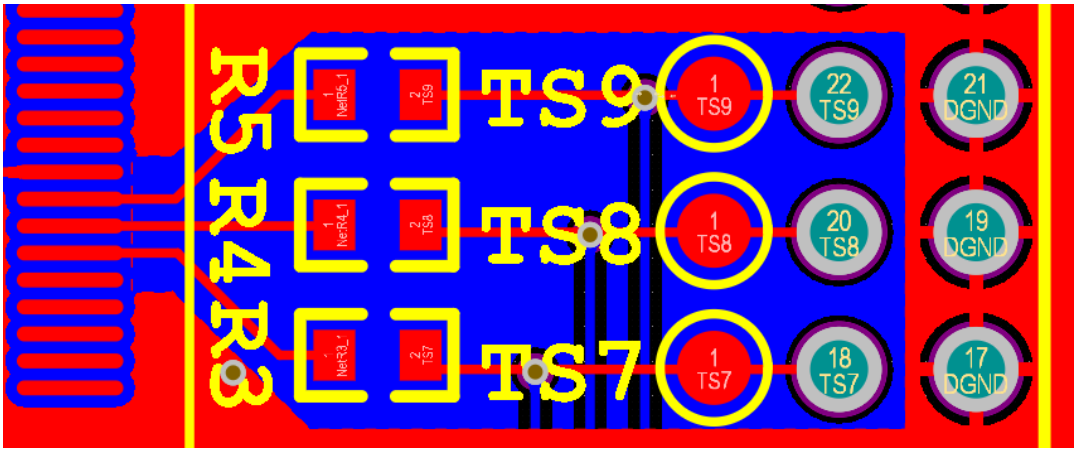


图 3.9（b）双层板的接近芯片部分走线

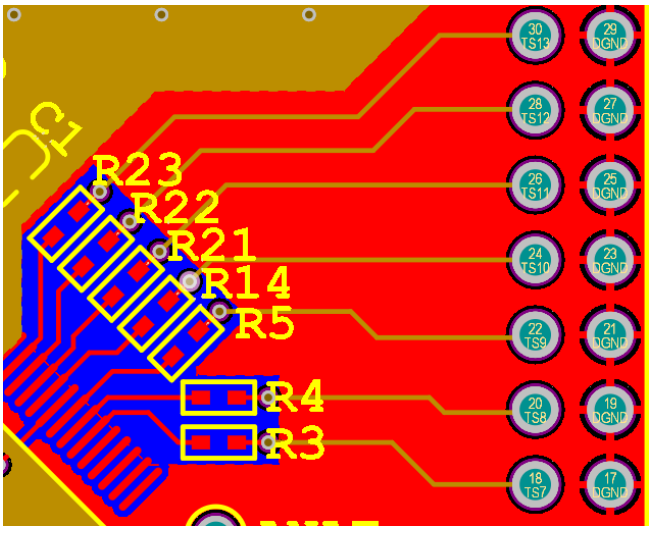


图 3.10（c）四层板的接近芯片部分走线

5. 走线应该远离其他数字信号（UART、 I2C、 SPI 等），如果不能避免，应该相互之间正交走线；传感器走线之间距离至少要为走线宽度的 2 倍；

6. 两个相邻按钮之间的间距应该足够大，如果按下一个按钮，手指不应触及另一个按钮；

7. 触摸覆盖层厚度建议以 1-5mm 为宜；
8. 传感器走线周围的覆铜和 TSCAP 脚电容的地选择与芯片地共地；

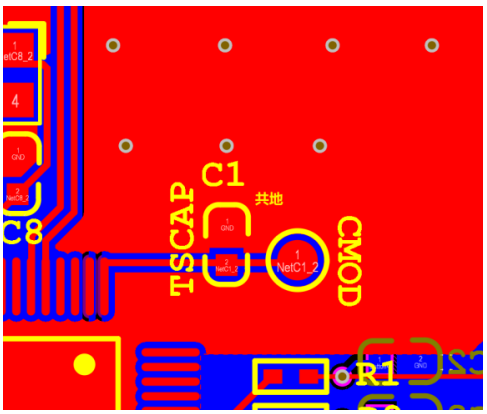


图 3.11TSCAP 与芯片共地

9. 触摸通道电阻选择 **3.3K** 为宜，尽量靠近芯片管脚；
10. 参考电容尽量选用温度稳定性高的 X7R 或 NP0 材质电容，靠近芯片放置，其他对地走线要尽量短；
11.为了提高 EMC 性能可以在触摸通道与对地端（**建议选择在 MCU 引脚端**）预留一个 **15pF-20pF** 的电容位置。

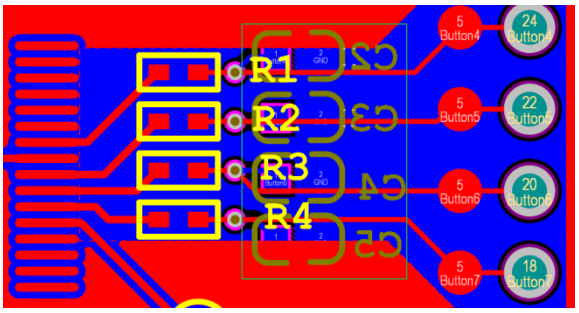


图 3.12 预留电容位置

为了和 TSITuner 上位机进行交互调试， 用户设计的 PCB 至少需要留出一个全双工串口。这个串口可以和其他串口复用，预留相应接插接口即可。我们的上位机软件会为用户的设计产生独立的，应用不相关的调试专用软件工程，待用户调试完成后，再将库和参数数据加入到实际应用工程中。

3.3.6 电源设备布局建议

在电源的设计中请注意以下几点：

1. 电源纹波过大，可能会影响触控按键的稳定性，应尽量减少电源纹波；
2. 注意芯片供电范围；
3. 电源线可通过串接磁珠增加 EFT 性能，磁珠应尽量靠近接插件接口位置；
4. 电源线宽不能低于 1MM；

5. 电源线上的去耦电容应计量靠近芯片的电源和地管脚；
6. 连到触控芯片上的电源线不要再引出去驱动其他负载；

3.3.7 原理图规则检查表(适用于 FM33FT0xxA/FM33FR0xx)

原理图设计完成后，可参考此表进行检查：

表 3.3 原理图 CheckList

编号	项目	推荐参数或方法
1	电源去耦电容	0.1uF
2	电源储能电容	10uF
3	TSCAP 外接电容	2.2nF
4	管脚排布	不要让触控传感和通信 IO 混排
5	触控管脚电阻	3.3K

表 3.4 PCB CheckList

编号	项目	子项目	最小值	最大值	推荐参数或方法
1	去耦电容（小）		0.01uF	1uF	0.1uF 靠近芯片管脚
2	储能电容（大）		10uF	100uF	10uF 靠近芯片管脚，瓷片电容或者电解电容
3	TSCAP 外接电容	容值	-	-	2.2nF
		位置	-	-	靠近 TSCAP 管脚
		材质	-	-	X7R NP0
4	按键	形状	NA	NA	实心圆/圆形长方形
		尺寸	5mm	15mm	10mm
5	滑条	宽度	4mm	-	8mm，至少影响覆盖两极滑条的电极，边缘电极未咬

					合处宽度为 4mm
		高度	7mm	15mm	10mm
6	各类传感器和铺地的间距	-	0.5mm	2mm	一般间距与覆盖物厚度成正比，覆盖物越厚，间距越大
7	传感器走线	宽度	-	-	7mil
		与铺地的间距	0.5mm	2mm	1mm
		转角	-	-	没有尖锐转角
		走线规则	-	-	如果有交叉走线，走垂直方向
8	铺地	-	-	-	传感器走线周围 17%覆盖率（7mil 线宽，70mil 间距）网状铺铜，传感器周围 25%覆盖率（7mil 线宽，45mil 间距）网状铺铜

4. 工程设计和调试

4.1 TSI Tuner(适用于 FM33FT0xxA/FM33FR0xx)

复旦微电子自主开发 TSI Tuner 软件，用于用户快速建立触摸应用原型设计并进行参数调试，此小节将进行详细介绍。

4.1.1 需求确定

为了能够全方面的展现 TSI Tuner 软件的各项功能，我们的 DEMO 工程计划设计实现一个可以使用触摸滑条和按键进行调色调亮度的小彩灯。该设计需要如下控件：

1、2 个按键，一个用作开关，一个用作模式选择；

2、1 个触摸滑条，用于调节亮度或者颜色；

主控芯片选择复旦微触摸 DEMO 板使用的 FM33FR0510。最终我们制作的 DEMO PCB 如图所示：

图 4.1 复旦微样机

4.1.2 建立工程

打开软件后，在首页点击 New 按钮，将会打开新建工程对话框。我们需要填写工程名称，选择芯片系列和芯片型号。确认无误后，点击创建按钮，完成新建工程操作。

图 4.2 新建工程

创建工程后，可以通过**文件⇨保存工程**对工程文件进行保存，或者可以使用快捷键 Ctrl+S。这里我们将工程文件保存至 D:\TSIDemo 文件夹中。

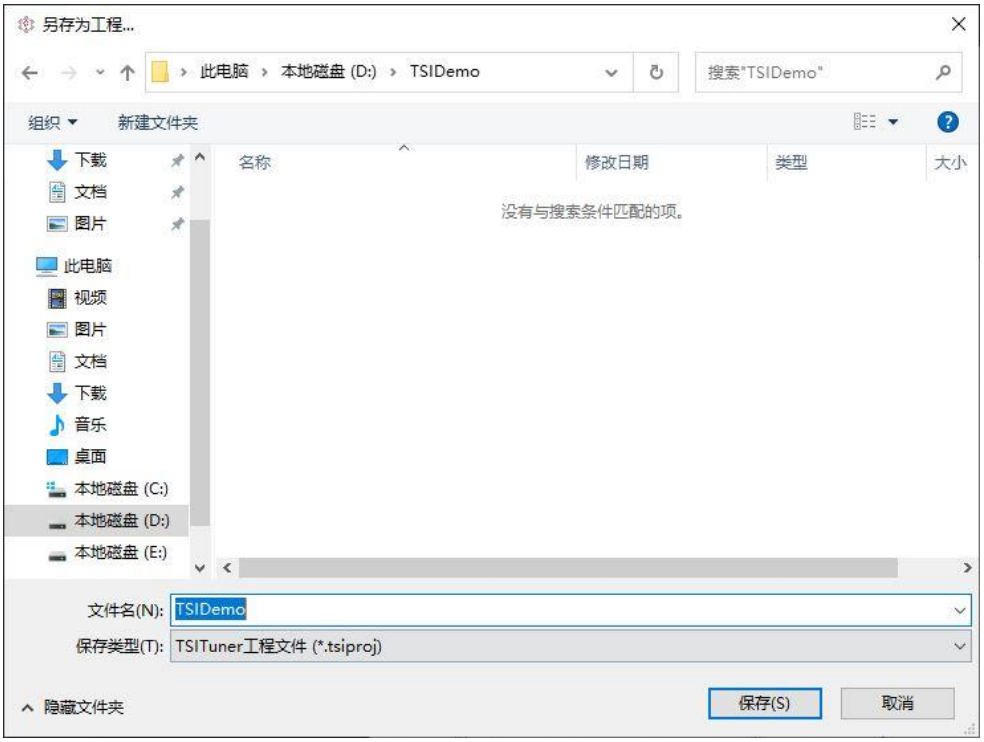


图 4.3 保存工程

4.1.3 工程设计

在创建工程完成后，进入到设计界面。在该界面，我们可以通过图形化的方式，规划布置触摸控件（按键、接近感应、滑条等）。下面对设计页面做一下简单介绍。

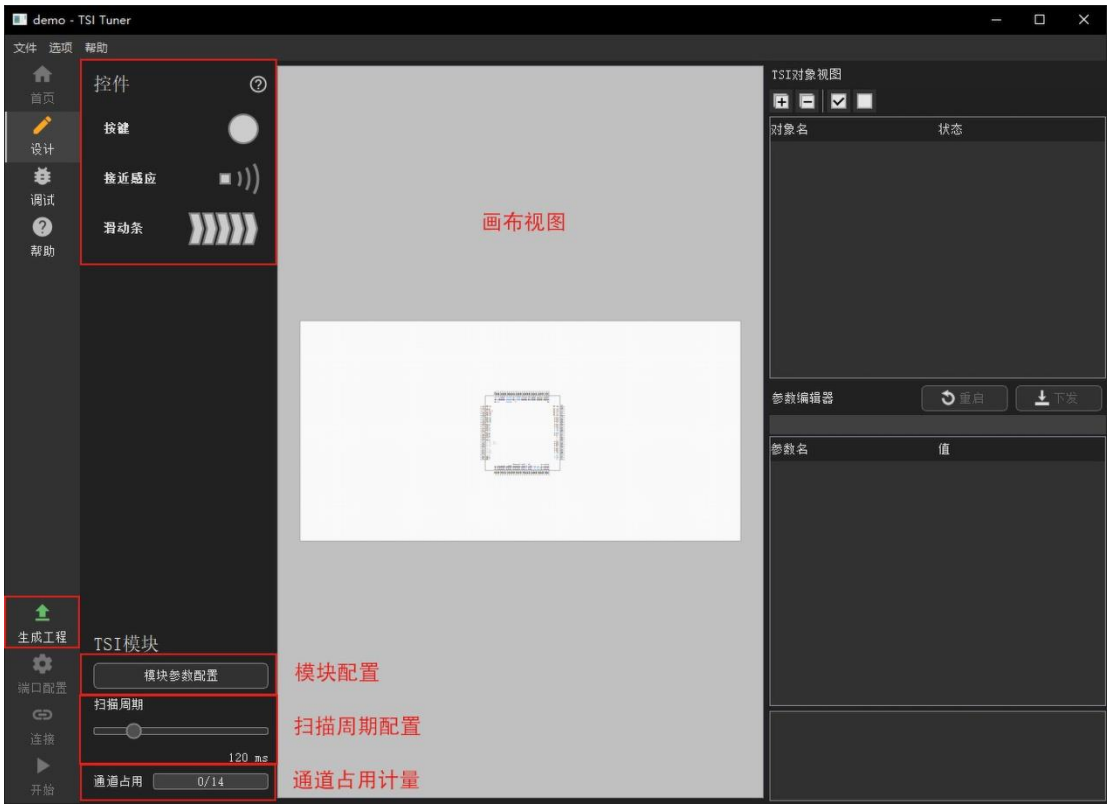


图 4.4 主界面

位于设计页面中央的是画布视图，用于浏览操作设计图纸。图纸设计操作按键如下：

- **浏览：**
  - 右键按住移动鼠标可以平移画布视图；
  - 鼠标滚轮上下滚动可以缩放画布视图；
- **选择和编辑：**
  - 鼠标左键单击：用于选择控件和连线。控件和连线选中状态下，按 Delete 键可删除；
  - 鼠标左键双击：对控件有效。双击控件会弹出一个浮动窗口，可以配置控件的属性（名称，滑条控件可配置段数等等）；

- **绘制连线**：画布中芯片封装的引脚，以及控件的引出脚都是可以进行连接的端口。将鼠标移动到需要连线的一个端口上，如果出现红色”X“光标，此时单击鼠标左键可以开始连线。连线时可以使用空格键控制拐弯方向，鼠标左键按下可以创建新的路点。当走线到达目标端口时，鼠标放在目标端口上，同样如果出现红色"X"光标，则单击鼠标左键可以结束连线绘制。如果在连线过程中需要放弃连线重新连接，可单击鼠标右键，这将退出连线模式，并删除当前绘制的连线。

注意：*如果鼠标放在端口上后，鼠标变成了禁止的图标，说明该端口不可连线。*

页面左侧的面板上有两个栏目：控件列表和模块配置栏目。控件列表栏目内容为当前可用的控件，我们可以在想要加入设计的控件项目上左键按下并拖动来将其加入设计；模块配置栏目内容包括模块参数配置，扫描周期配置和通道占用计量条。

页面右侧面板有两个窗口，分别是 TSI 对象视图和参数配置视图。这两个窗口在设计模式和调试模式都存在，但用途会有一些差别。设计模式中，TSI 对象视图用于展示当前画布中存在的控件信息，以及他们的状态。参数编辑器视图则用于配置对象的参数。通过在 TSI 对象视图单击想要编辑参数的对象，来在参数编辑器视图中编辑参数。

4.1.3.1 添加控件到设计

依据先前制定的需求，我们将按键和滑条加入到设计画布中。在左侧控件列表中分别按住按键和滑条项目，拖动到画布中，即可完成添加。双击控件，会弹出控件属性列表，我们可以为它重新起一个便于记忆的名字。这里我们分别命名按键为 Power 和 Mode，命名滑条为 Slider。

图 4.5 按钮控件示例

通过在画布上连线，可以为控件分配管脚。我们将 Power 按键分配到通道 10，将 Mode 按键分配到通道 4，将 Slider 滑条的第 1~5 段依次连接到通道 5~9。最终配置如下图所示：

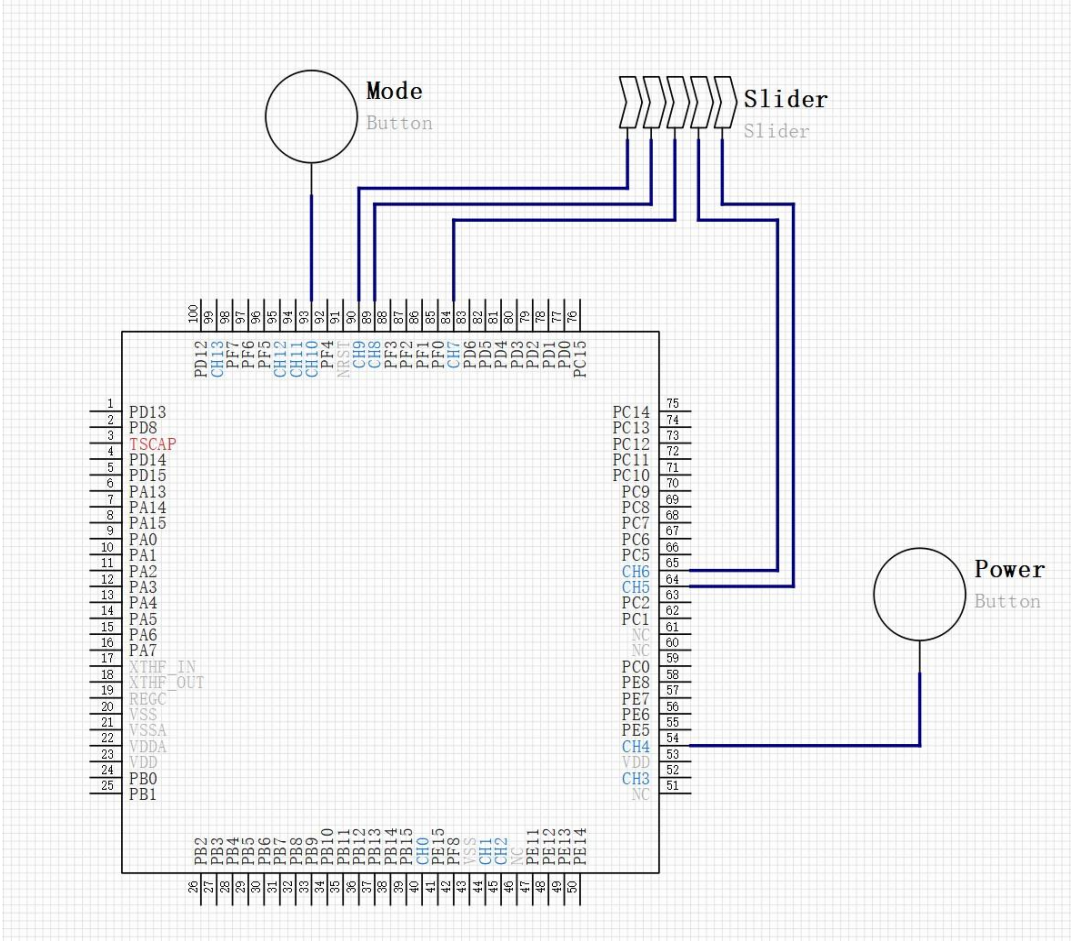


图 4.6 TSI 设计图

4.1.3.2 配置模块参数

我们首先需要进行触摸模块（TSI）的基本参数配置。单击左侧面板的模块参数配置按钮，打开模块参数配置界面。

为了使得模块能够正常的工作，我们需要先配置时钟。这里时钟需要设定的和程序计划使用的时钟一致。推荐让 TSI 工作在当前时钟能够到达的最高频率，这样比较容易得到理想的灵敏度。在示例应用中，我们使用 24MHz 作为系统主频（PLL 倍频至 48MHz，2 分频至 24MHz），FM33FR0 系列的 TSI 模块使用 APB 作为总线和工作时钟源。我们切换到**时钟和功能配置**页面，配置 TSI 时钟源为 APB，频率为 24MHz。TSI 采样时钟源一般建议使用 PRS，EMC 性能较好。

然后，我们需要配置功能：勾选**使用补偿 IDAC** 项目，可以在保持灵敏度情况下有效的提高传感器有效信号范围，我们推荐您一直勾选此项；其他保持默认即可。

接下来，我们还需要配置软件算法。切换到**触摸检测配置**页面，可以配置传感通道使用的滤波器：

- 普通传感通道（按键、滑条等非接近感应控件）：推荐您开启**均值滤波器**和**中值滤波器**。如果干扰较大，需要关闭均值滤波器并开启 IIR 滤波器，IIR 滤波器的系数建议设置为 32 以上；
- 接近传感通道（接近感应控件）：推荐您开启**中值滤波器**和**高级二段式 IIR 滤波器**。对于干扰较大的情况，可以选择开启均值滤波器；

允许 Baseline 始终更新选项会使得基线算法在触摸信号大于触发阈值后继续更新。大多数情况下，您并不需要勾选此项。

使用 IIR Baseline 选项能够配置固件使用 IIR 滤波器作为 Baseline 算法。在此模式下需要调校的参数较少，实现设计较为省心，**建议勾选该项**。

参数调校方式选项提供了 3 种方式：**手动调校**、**硬件参数自动调校**和**补偿 IDAC 自动调校**（仅在使能补偿 IDAC 时可选）。我们**推荐您仅在原型设计的时候使用硬件参数自动调校**，这将使得触摸库自动把硬件参数（采样时钟分频、分辨率、IDAC 电流）调整到一个相对适宜的范围内；而当您开始细调配置的时候，强烈建议您修改为使用**手动调校**或者**补偿 IDAC 自动调校；当您准备量产时建议修改为补偿 IDAC 自动调校**：

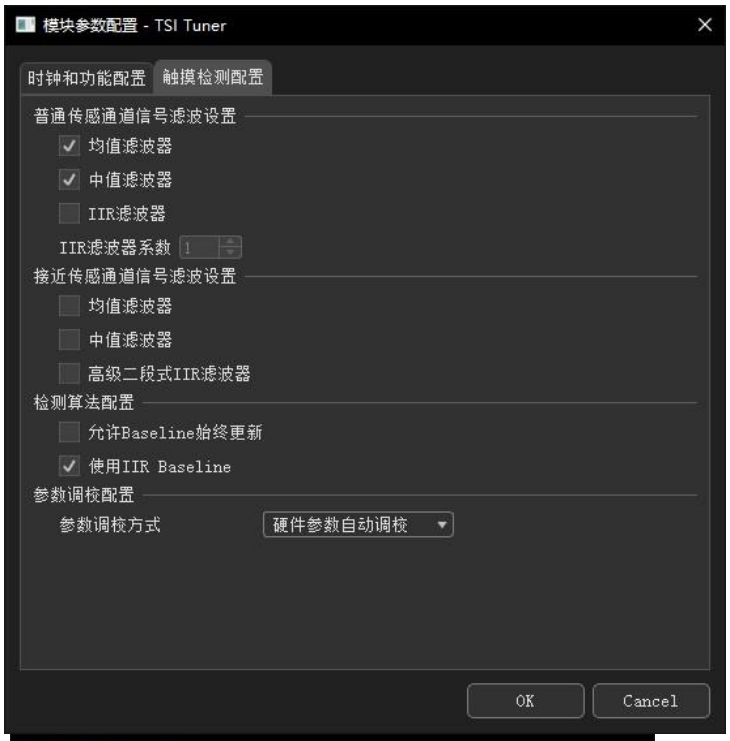
- 手动调校会使触摸库完全按照您指定的数值来配置硬件参数，从而保证调参时和实际测试生产环境参数的一致性；
- 补偿 IDAC 自动调校在手动调校之上加入了对补偿 IDAC 的自动校准，**在灵敏度不变前提下增强对环境的适应性**；

综上所述，对于我们的示例应用，我们配置了如下项目：

- 时钟和功能配置页面：
- TSI 时钟源：APB
- TSI 时钟源频率：24MHz
- TSI 采样时钟源：PRS
- 勾选**使用补偿 IDAC** 选项

- **档位选择有 1.2uA 和 300nA（注:300nA 只有 FT/FR 系列芯片 U05 版本有此档位，其他版本只可配置 1.2uA）**

- **触摸检测配置**页面：
  - 普通传感通道信号滤波设置中，勾选**均值滤波器**和**中值滤波器**；
  - 勾选**使用 IIR Baseline**；
  - 参数调校方式选择**硬件参数自动调校**；



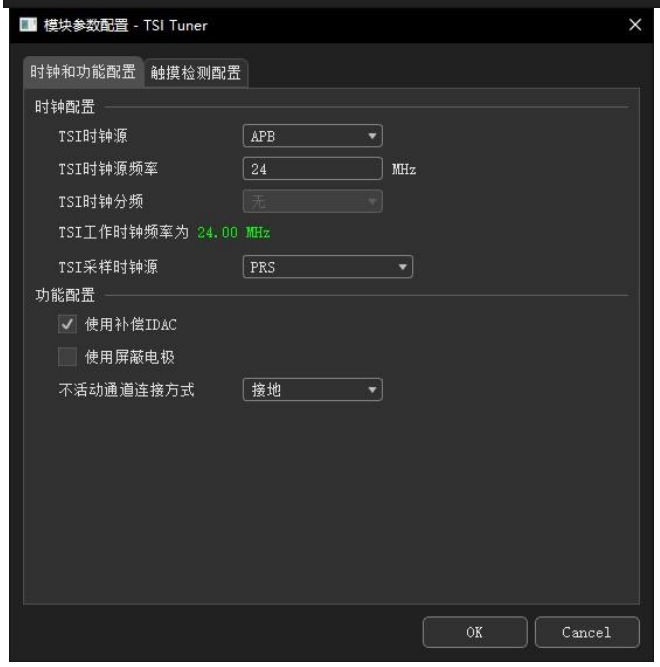


图 4.7 工程配置示例图

4.1.3.3 配置扫描周期

扫描周期代表了每一轮触摸按键扫描的周期，也即每次数据更新的最小时间单位。示例工程使用的 FM33FR0 系列芯片，其扫描周期最小为 20ms，并以 20ms 为单位步进。在示例工程使用的场景下，由于不需要低功耗，为了使按键响应尽可能快，我们将扫描周期配置为最短的 20ms。

图 4.8 扫描周期

4.1.3.4 生成调试工程

在完成设计后，我们就可以生成调试工程并下载到芯片中，进行调试工作了。单击左侧工具栏中的**生成工程**按钮，将弹出导出对话框。下面介绍一下导出对话框内各个输入框和选项：

图 4.9 调试工程

- **生成目标：**
  - 可以选择生成完整调试工程还是仅生成库；
  - 路径：指定生成路径；
- **工程设置：**
  - 工程名称：生成工程的名称；
  - 工程路径：生成工程所在的路径，工程会直接生成在该路径中；该输入框无法直接编辑，您可以通过右侧的**浏览**按钮选择目标文件夹；
  - 工具链/IDE、版本、调试器：选择您打开工程使用的工具链及其版本，以及仿真和下载使用的调试器类型；
- **生成设置：**
  - **生成调试端口初始化代码模版**选项：TSITuner 软件提供了一些调试端口配置模版，方便您快速配置工程，**一般推荐您首次导出的时候打开该选项**；生成调试端口配置后，可以根据实际情况再修改端口使用的 IO 口等来匹配您 PCB 的实际情况；
  - **尝试根据配置的 TSI 模块时钟自动生成时钟初始化代码**选项：推荐勾选，TSITuner 软件会根据您的模块参数配置，尝试自动生成符合要求的时钟初始化代码，免去手动配置的烦恼（大多数情况都能正确配置，少数特殊情况无法自动生成，需要手动在代码中添加）。

对于示例工程，我们最终配置如下（仅供参考，您可以根据自己的实际需要进行配置）：

图 4.10 导出工程

单击**导出**按钮，可以将工程导出到指定文件夹。

注意：*如果在之前您没有保存过工程文件，单击导出按钮后会先弹出窗口让您选择工程文件的保存位置，再执行导出操作。*

在弹出导出成功提示后，我们点击**打开**按钮打开调试工程：

图 4.11 打开工程

调试工程目录结构如下图所示：

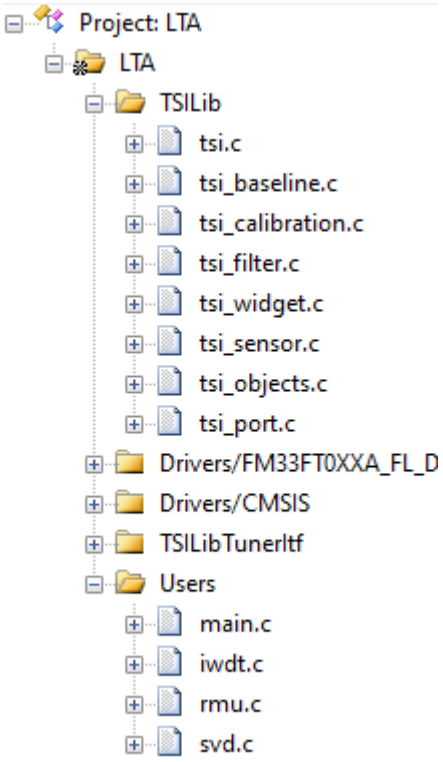


图 4.12 KEIL 工程

TSILib 目录下内容为 TSI 软件库；**TSILibTunerItf** 为 TSI 软件库调试接口实现，仅用作调试使用，实际应用中不需要添加这些文件；**Drivers/FM33FR0xx_FL_Driver** 和 **Drivers/CMSIS** 为您选择的芯片对应的 FL 驱动库；**Users** 文件夹中的 **main.c** 包含主程序入口。

4.1.3.5 补充代码并下载

要让生成的调试工程能够正常运行，我们还需要补充两处代码：

- **调试端口配置代码：TSILibTunerItf/tsi_debug_io.c**，需要实现相应接口函数；如果在上一步没有勾选**生成调试端口初始化模版**的选项，这里需要手动补充；建议首次配置的时候先勾选生成端口配置代码，再根据模板修改成需要的配置；

注意：*如果您使用生成的配置代码作为模板进行了修改来适配设计，请注意之后如果还需要导出代码到该工程的时候，不要再勾选生成调试端口初始化代码选项，否则会将您编写的代码覆盖；*

- **时钟初始化配置代码**：位于 Users/main.c 中的 SystemClockInit 函数；如果您在上一步没有手动勾选**尝试根据配置的 TSI 模块时钟自动生成时钟初始化代码**选项，需要手动配置时钟初始化使其符合您设定的模块工作时钟；

生成工程中，main.c 和 tsi_debug_io.c 中包含形如下方的用户代码块：

```
/* USER … BEGIN */

…

/* USER ... END */
```

这些代码块在重新生成工程的时候内容会被保留，可以用于添加一些用户想要额外做的工作。**请您务必把自己额外编写的代码放置在这些代码块中，否则可能丢失。**

完成上述代码补充后，编译并下载工程到芯片中，我们就可以进行参数调试了。

4.1.3.6 调试

单击窗口左侧的**调试**按钮，进入到调试界面。在该界面，我们可以观察目标板各个触摸按键的信号强度和当前状态，并根据情况来调节参数。下面对调试页面做一下简单介绍。

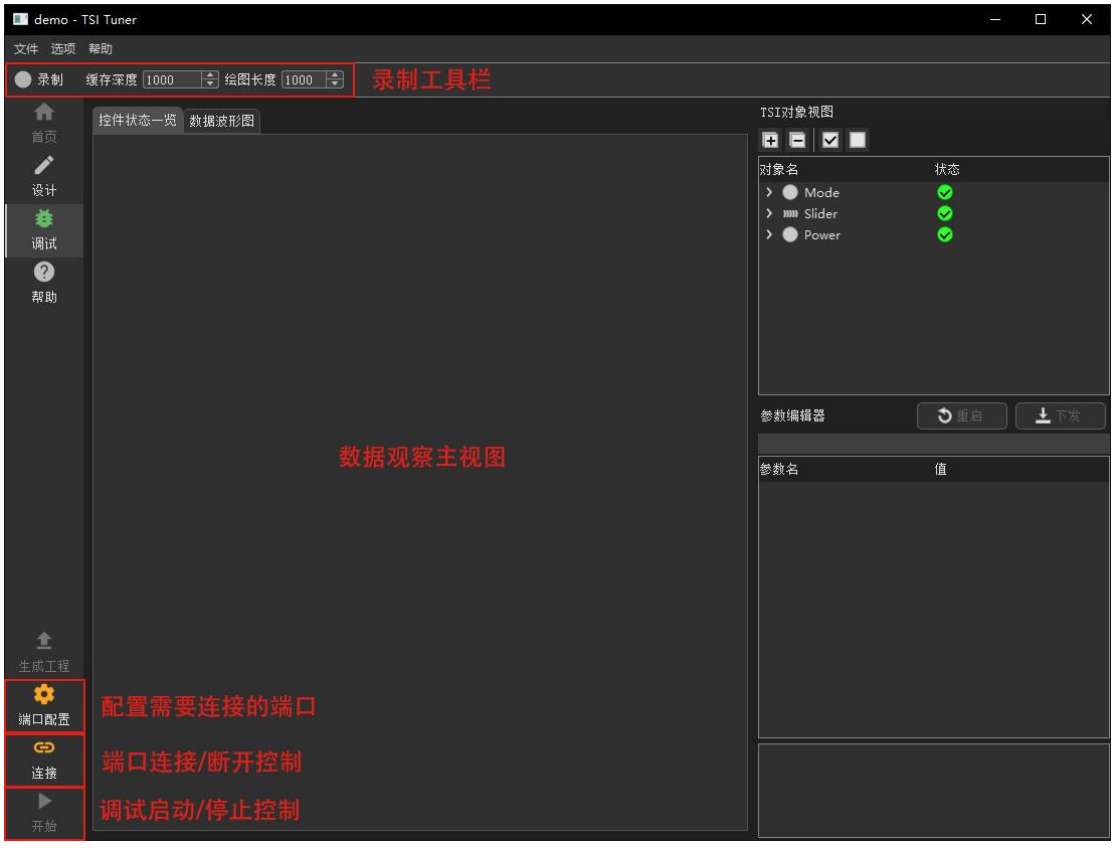


图 4.13 调试界面

位于调试界面中央的是**控件状态一览**界面和**数据波形图**界面，可以通过上方的标签切换。控件状态以比较直观的方式展现所有观察控件的状态，数据波形图则用于详细观察每个传感器的信号波形以及控件的状态。数据波形图界面里的每个波形的右上角都有两个按钮：小尺寸/全尺寸按钮用于使波形图在大图和小图之间切换，方便同时观察多个波形；隐藏/显示按钮则用于隐藏该波形，方便挑选感兴趣的波形进行观察。

页面顶部有一个工具栏，可以操作数据录制，以及配置数据缓存深度和波形图的绘图长度。

连接芯片并开始调试

要连接到芯片调试接口，可以单击左侧工具栏的**端口配置**按钮，会弹出一个对话框。在选择好需要连接的端口和配置之后，单击确定离开对话框。之后单击**连接**按钮进行连接，如果芯片程序正常运行，**开始**按钮将会变为活动状态，同时**连接**按钮变为**断开连接**按钮，这代表连接芯片的操作已经完成。

接下来，我们单击**开始**按钮，如果运行成功，**开始**按钮会变为**暂停**按钮，这代表着已经开始调试流程。

选择观察对象

在连接到芯片后，可以注意到右侧的 **TSI 对象视图**面板的列表会多出一栏：**观察**。这个栏目是用于选择需要观察哪些控件和传感器对象的，只有我们勾选了的对象，它的波形和状态才会显示在视图中。

在示例应用中，我们来观察一下 Power 按键的波形和状态：在 **TSI 对象视图**窗口勾选 Power 按键的**观察**选项，它所包含的传感器 **Power_Sns0** 的观察选项也会被自动勾选。我们切换到**数据波形图**界面，就可以看到相应的波形了。可以尝试按下按键，对应波形图会发生相应变化。

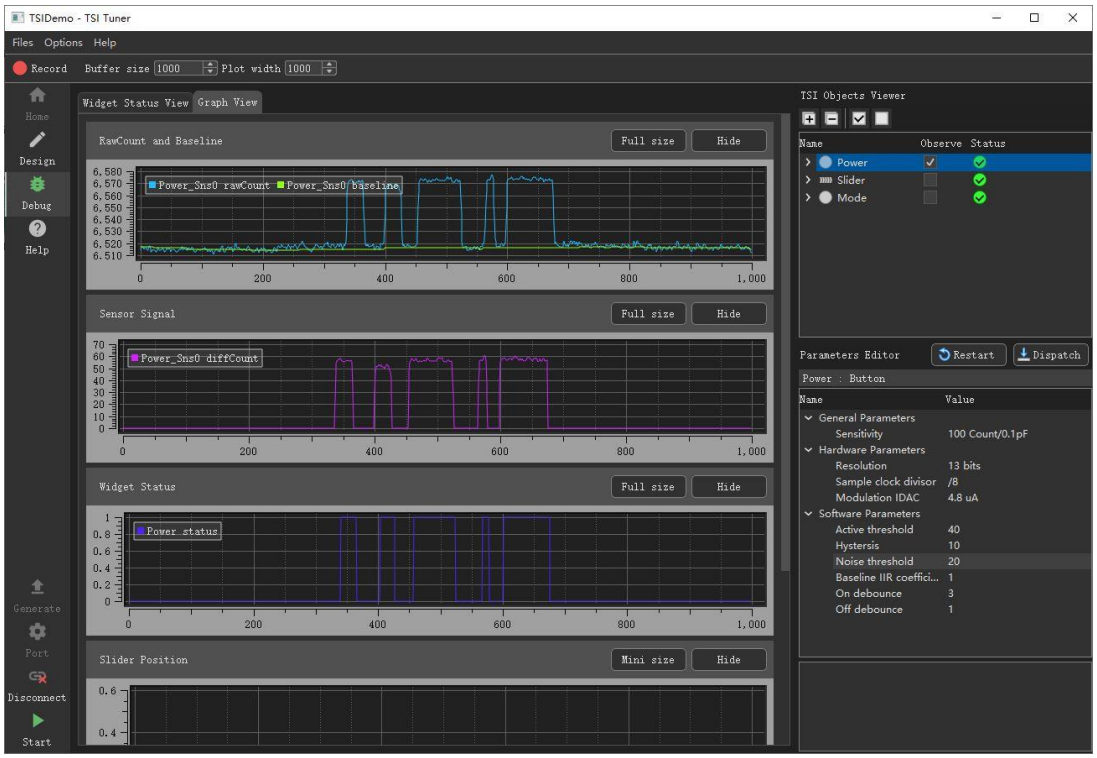


图 4.14 调试窗口

调节参数

在调试过程中，如果观察到的波形和状态不符合预期，我们可以对参数进行调节。在窗口右侧的 **TSI 对象视图**面板中，选择需要调节参数的对象，该对象的参数会展示在下方的**参数编辑器**面板中。单击想要修改的参数值，即可进行修改。在所有参数修改完成后，单击右上角的**下发**按钮，可以将参数下发到芯片。参数编辑的展现效果如下：

- 修改后的参数值如果和原来的值相比有变化，会加粗显示；
- 参数编辑框右侧的撤销按键可以用于撤销当前作出的更改（可以撤销时，箭头会变为红色）；

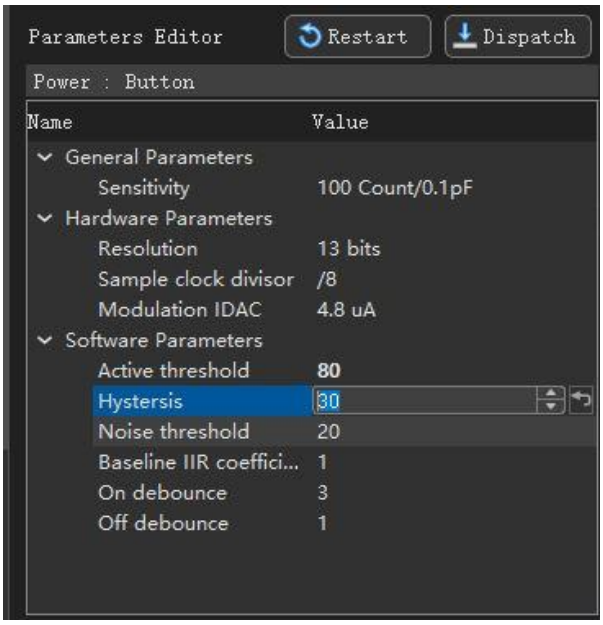


图 4.15 参数框

关于每个参数具体的含义，详见 **TSI 触摸检测原理和算法**文档。

在调节到满意参数后，我们可以返回设计工程来保存参数。点击左下角断开连接按钮断开调试连接，然后点击**设计**按钮返回设计视图。这里如果您在调试过程中修改了参数，会弹出一个参数比较对话框让您确认参数变动。如果您不想修改某项参数，可以取消勾选这项参数的更新选择框，这样参数就不会更新到工程中。

在每次调参之后，可能当前配置不能满足您的需求。在这种情况下，您可以在设计模式中重新配置模块参数。**需要注意的是，在重新配置完成后，您也需要重新生成调试工程，编译并下载到芯片中，这样新的参数才能够生效。**

导出软件库

在调节完成参数后，我们可以导出包含参数的，可直接用于实际应用工程的软件库。和导出调试工程类似，我们还是单击左侧工具栏中的**生成工程**按钮，打开导出对话框。之前在 **4.1.3.4** 生成调试工程中已经介绍了导出对话框，我们这里直接选择导出目标为**库**，并配置导出路径，然后直接单击**导出**按钮即可将包含您当前参数的软件库导出到指定文件夹中。您可以将库嵌入到您的应用工程中，加入对应函数调用即可使用。

我们将调好的示例工程参数导出到一个独立的 TSIDemoCode 文件夹，如下图所示：

图 4.16 生成库

打开导出文件夹，可以看到 TSI 软件库已经导出完成（位于 TSI\Lib 文件夹内）。接下来我们只需要为我们的工程添加 TSI 软件库和应用代码逻辑，就能实现所需的功能了。我们建立一个 FM33FR0510 的工程，编写一下按键背光 LED 和状态 LED 的驱动。然后，将导出的 TSI 软件库添加到工程。最终工程目录结构如下：

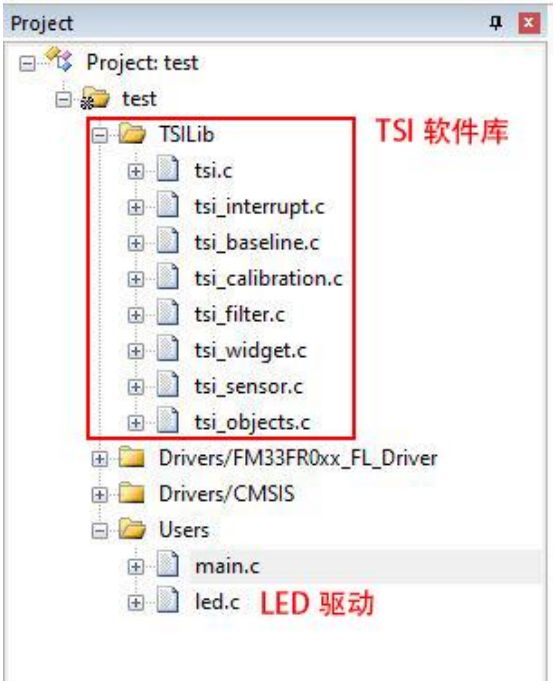


图 4.17 工程图

接下来，我们编写 main.c 文件，调用 TSI 软件库接口，完成 TSI 软件库的初始化/启动，以及按键状态读取、背光控制和三色彩灯调色调亮度的功能。我们将部分代码片段展示如下，完整工程请见 **TSI 彩灯 DEMO 工程**。

- **TSI 软件库初始化和启动：**

```
/* Init TSI */

TSI_Init();


/* Enable all widgets */

TSI_Widget_EnableAll();


/* Start TSI */

TSI_Start();
```

- **读取按键状态：**

```
if(TSI_WidgetList.mode.buttonStatus)

{
```

```
    /* 按键按下*/
}
```

- **读取滑条状态：**

```
if(TSI_WidgetList.slider.base.status)
{
    /*滑条被按下，读取当前位置数据*/
    uint8_t center = TSI_WidgetList.slider.centerPos;

    /* 根据滑条位置数据 center 进行处理 */
}
```

4.2 硬件参数调节

4.2.1 手动调节

手动调节过程直接与触摸性能相关，请在调试前仔细阅读此章节。

4.2.1.1 采样分频

采样分频的选择，首先要保障 **Sensor 电容可以满充满放（0V - 参考电压 1V）**。下图 1 为测试中工作时钟 24MHz，采样分频为 8 时的充放电波形，由图可知充放电呈下降趋势，并未达到满充满放；下图 2 为测试中工作时钟 24MHz，采样分频为 16 时的充放电波形，由图可知在此次采样过程中，已达到满充满放，所以将采样分频设为 16。

经过测试，**我们建议将分频设置为 16 或者 32**，通常可以使 Sensor 电容满充满放。

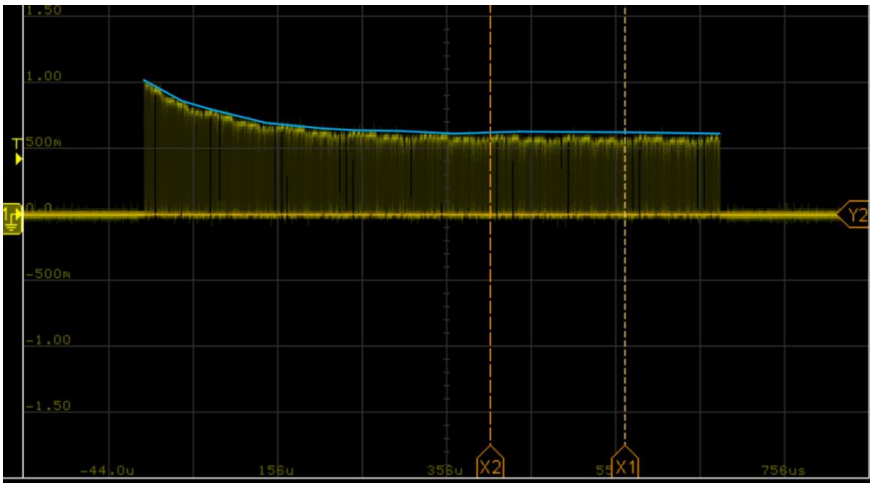


图 4.18 工作时钟 24MHz，采样分频为 8 的充放电波形

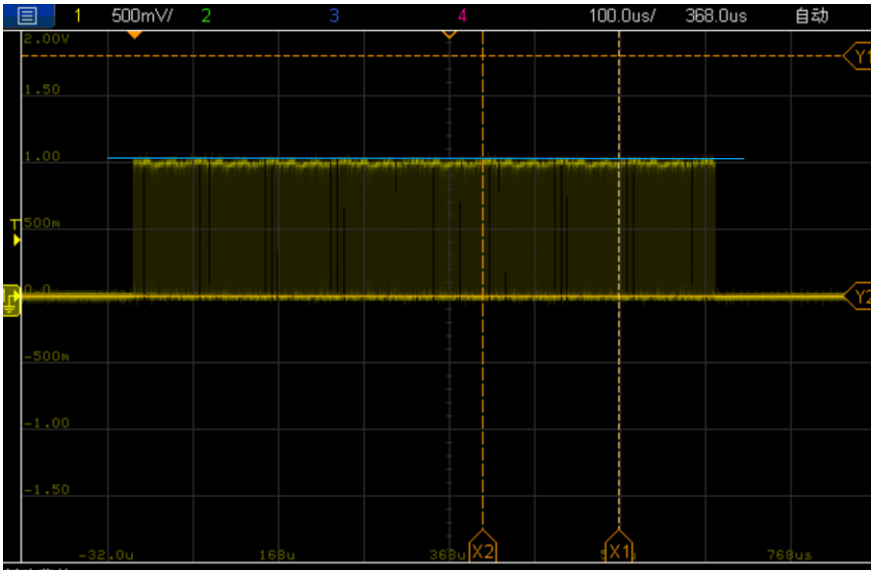


图 4.19 工作时钟 24MHz，采样分频为 16 的充放电波形

4.2.1.2 分辨率

使用 PRS 时钟的情况下，分辨率通常选用 **12 位**或是 **13 位**，可满足大多数应用场景。

4.2.1.3 调节 I_{Mod}

若挡位为 1.2uA,IMod 不可设为 0，最小为 1。在实际应用中，不推荐设置为 1，此时触摸增益最大，但同时噪声以及抖动幅值相对应也很大，影响触摸整体性能，**建议设置为 2、3、4、5**。

若挡位为 300nA,IMod 不可设为 0，最小为 4。在实际应用中，不推荐设置为 4，此时触摸增益最大，但同时噪声以及抖动幅值相对应也很大，影响触摸整体性能，**建议设置为 8、12、16、20**。

IMod 的大小与灵敏度成反比，在采样分频以及分辨率不变的前提下，**IMod** 是影响灵敏度的主要因素之一，**IMod** 越大灵敏度越低，**IMod** 越小灵敏度越高，例图如下所示：

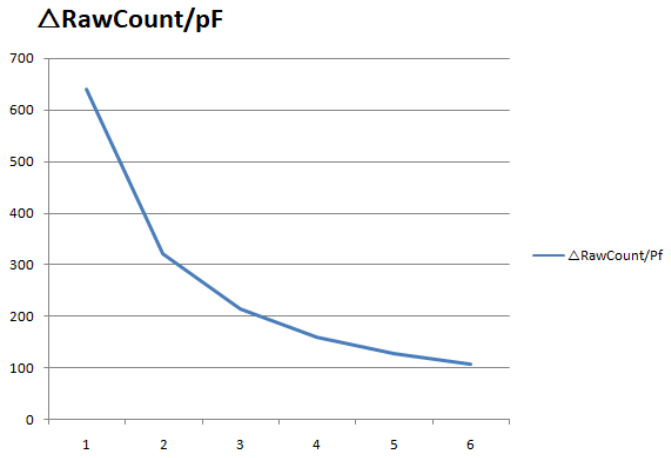


图 4.20 Imod 与灵敏度关系图

上图，横坐标为 **IMod**，由图可知，**IMod** 越大，灵敏度越低，即相同电容大小变化下，**RawCount** 变化更小。图中数值仅用为示例，具体大小请根据实际开发计算。

4.2.1.4 调节 I_{Comp}（单 IDAC 可以省略此步）

在调节 **IMod** 之后，得到合适的灵敏度的情况下，进行 **IComp** 调节，**IComp** 不影响灵敏度的大小，但会产生偏移补偿，即会影响 **RawCount** 的大小，一般在双 IDAC 模式下，需将 **RawCount** 初始值设定一般不超过最大值的 70%，留有较为充足的信号裕度。

4.2.1.5 触发阈值调制

在设立触发阈值时，请先进行仿真测试，模拟实际触摸情况，观察 diffCount 的变化区间，以此为参考设置触发阈值。

如下图所示，红线为 **RawCount**，蓝线为 **Baseline**，黄线为设置的 **activeTh**，在连续触摸测试中，**activeTh** 的设置需确保每一次触摸的 **diffCnt**（**RawCount** - **Baseline**）大于 **activeTh**。

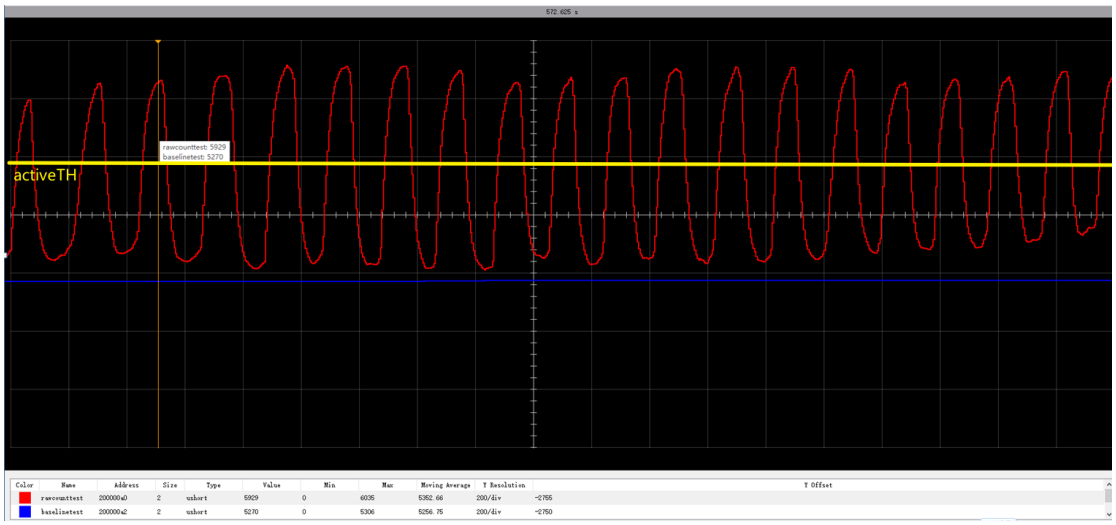


图 1.21 触发阈值设置区间

4.2.1.6 触发窗口

触发窗口的选择一般在完全按下按键时进行观察选择，观察其按下时上下抖动幅度，如下图所示，为按键按下状态，**其中黄色线为抖动范围**。通常可以将 **activeHys** 设置为抖动范围除以 2 左右，不宜过大，否则会影响触发效果。

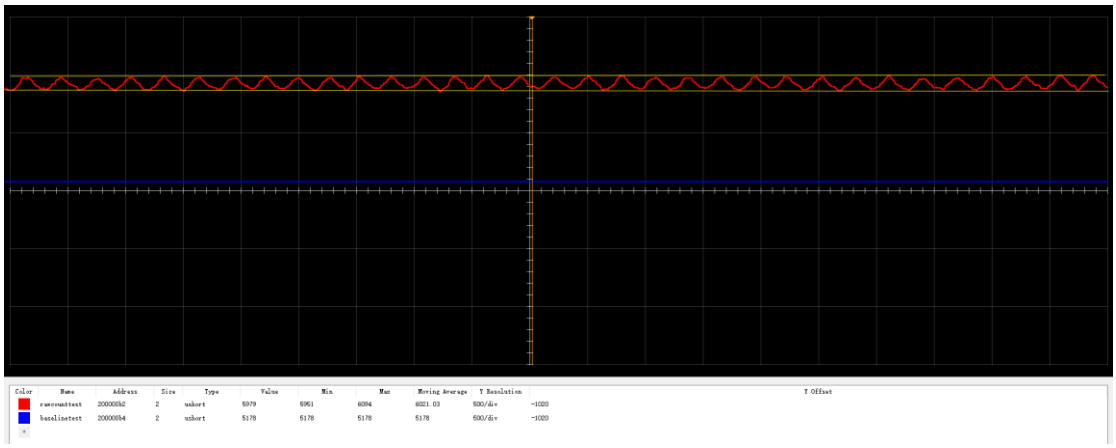


图 4.22 按键按下时抖动范围

4.2.1.7 噪声阈值

噪声阈值的设定首先需要涵括 RawCount 在未触发下的正常抖动，如下图所示，**其中黄色线为抖动范围**，**噪声至少设置为其抖动范围乘以 2 以上**，可以根据实际应用情况具体设置。

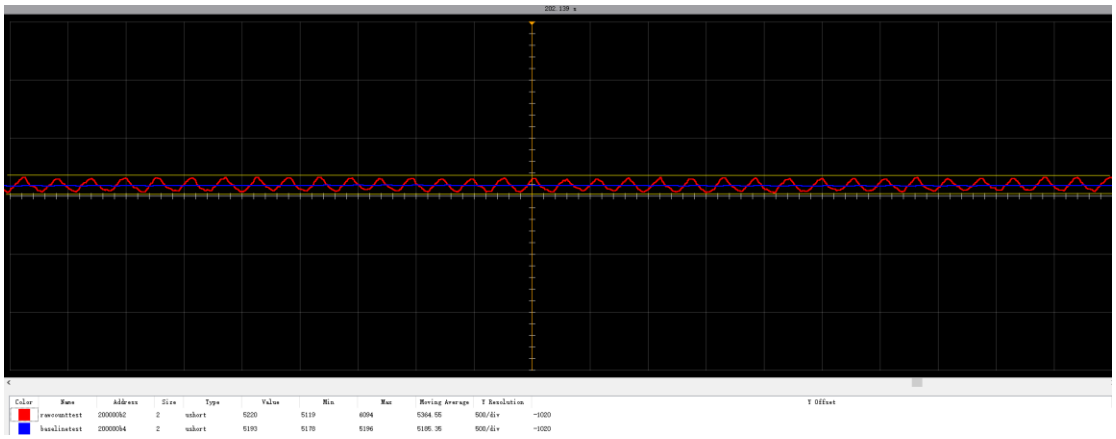


图 4.23 按键未按下时抖动范围

4.2.1.8 触发防抖与复位防抖计数

触发防抖与复位防抖计数不得设置为 0，至少为 1。一般将触发防抖计数设置为 3，代表当连续三次扫描 **diffCount>activeTh** 之后，触发状态置 1；复位防抖计数设置为 1，代表当 **diffCount<activeTh** 一次之后，触发状态复位为 0，根据开发实际需求可自行更改。

如下图示所示，**蓝色为 RawCount**，**绿色为 Baseline**，**红色为按键状态**。

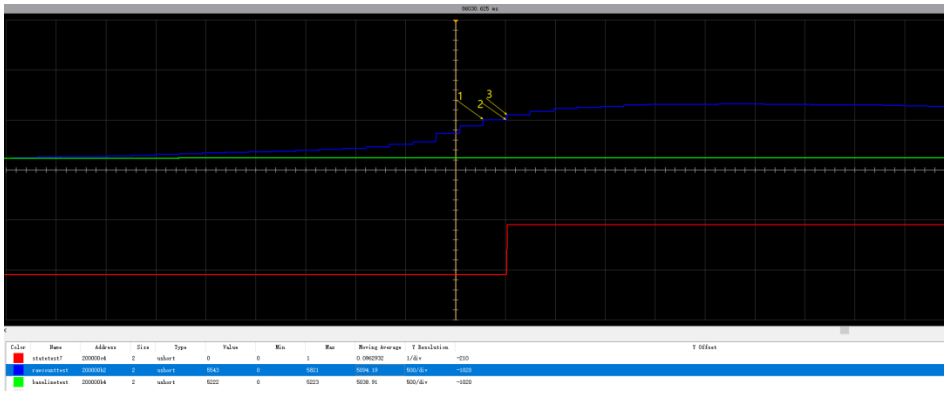


图 4.24 触发防抖计数

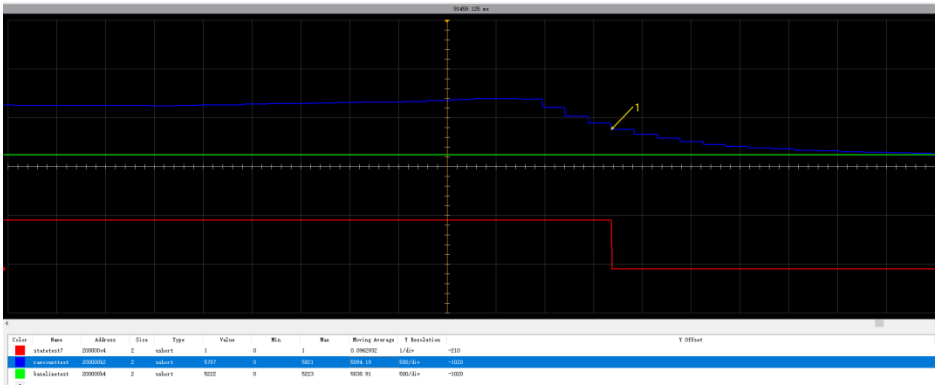


图 4.25 复位防抖计数

4.2.2 自动调节

TSI 库提供了两种参数自动调节机制：硬件参数自动调校和补偿 IDAC 自动调校。其中：

1. 硬件

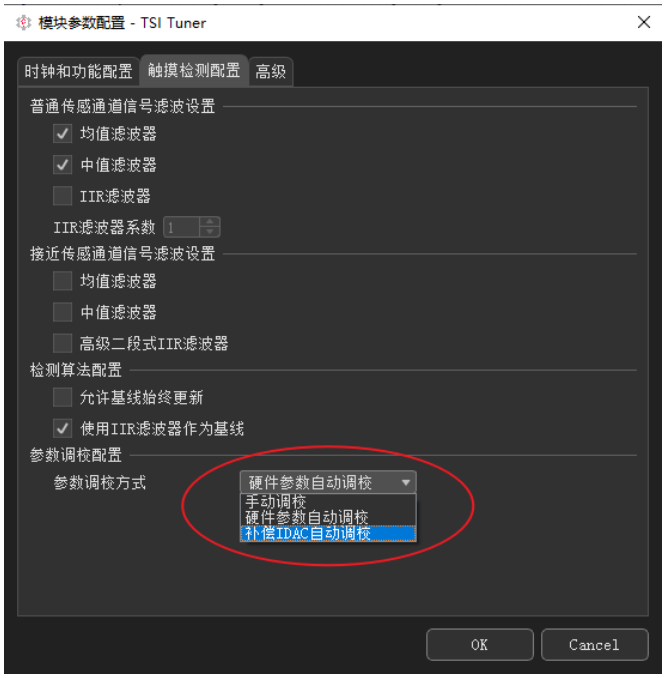


图 4.26 校准选择

硬件参数自动调节将对时钟分频、分辨率、IDAC 进行调校，在原型设计中可以使用此配置，不推荐在量产中使用。

补偿 IDAC 自动调校模式主要使用一些特殊场合（如高低温变化），此模式在其他硬件参数都固定的情况下，自动校准补偿 IDAC 至表现最优值，推荐在量产或是细调参数时打开此配置。

5.系统信噪比评估

5.1 评估背景

电容式触摸在实际应用中，主要关键的评估因素为信噪比（SNR），SNR 的大小将直接影响触摸性能，若系统信噪比过低，系统\环境噪声带来的误触事件发生概率会大幅度上升，**故在触摸参数调节完成之后请进行 SNR 评估**。

SNR 评估原理如下图所示：

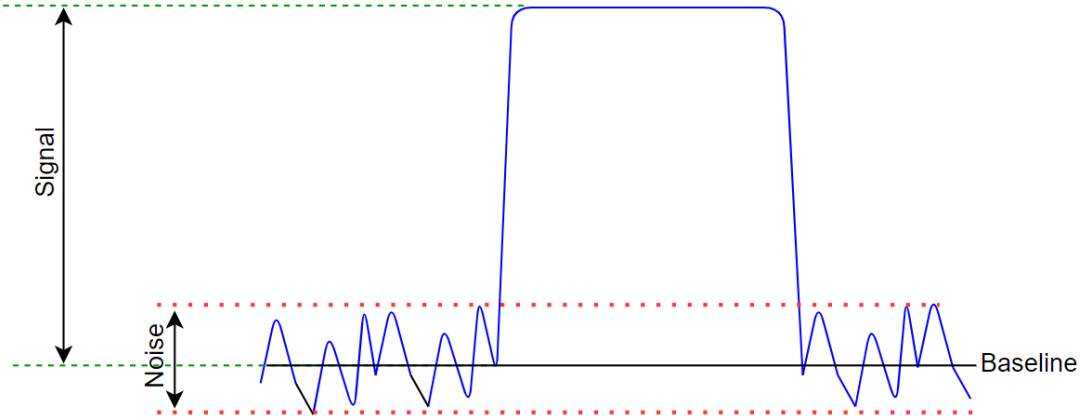


图 5.1 SNR 示意图

其中 Signal 计算如下：

$$\text{Signal} = \text{Rawcount\_AVR} - \text{Baseline}$$

Rawcount_AVR 为未触摸时的平均值；

Baseline 为基线，与软件库 Baseline 相对应；

其中 Noise 为一段时间的 Rawcount 最大值减去 Rawcount 最小值：

$$\text{Noise} = \text{Rawcount}_{\text{Max}} - \text{Rawcount}_{\text{Min}}$$

SNR 计算如下：

$$\text{SNR} = \text{Signal}/\text{Noise}$$

对每个通道 SNR 都需进行评估，需确保 SNR 大于 5,若不满足请调整触摸参数，调整完参数再次进行评估，若依然无法满足需求，请联系复旦微工作人员评估风险。

5.2 SNR 测量

测量信号请根据实际应用场景中的下限进行测量，如 8mm 铜棒，基本可涵盖触摸按钮所有的人体触摸场景。

5.2.1 TSI_TUNER 上位机评估 SNR

1. 进入调试界面，使用 USB 转串口工具将上位机与 MCU 进行连接，并点击开始按钮（SNR 测量之前务必打开开始按钮）；

图 5.2 SNRTEST 按钮

2. 点击 SNR TEST,进入 SNR TEST 测量模式下，右边控件框必须且只能选择一个测量通道，若选择错误则提示如下：

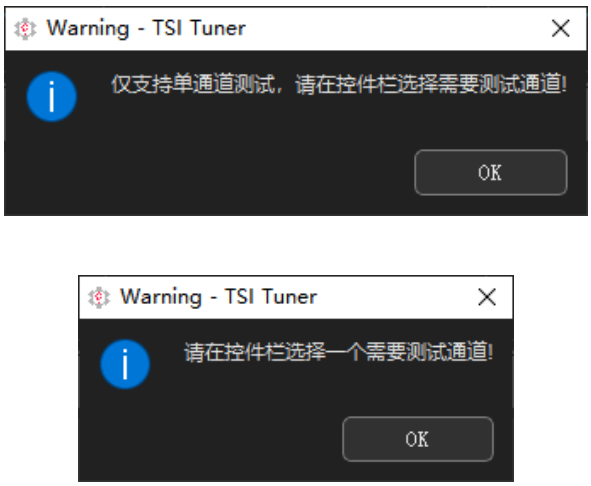


图 5.3 错误警告

3. 进入 SNR TEST 测量模式,默认处于测量阶段 1（准备测量噪声，不要去触摸面板）；

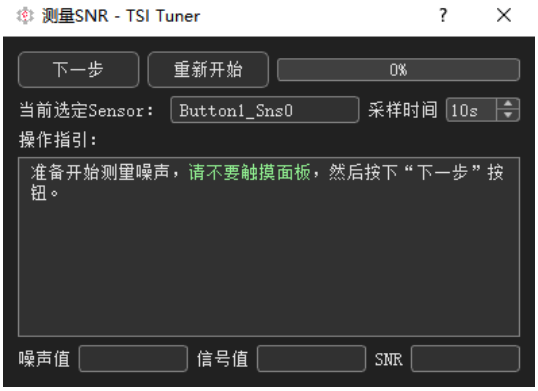


图 5.4 测量阶段 1

4. 点击下一步进入噪声测量阶段，采样时间 10s；

图 5.5 测量阶段 2

5. 测量噪声完成，准备测量信号，请在开始之前触摸面板，然后点击下一步；

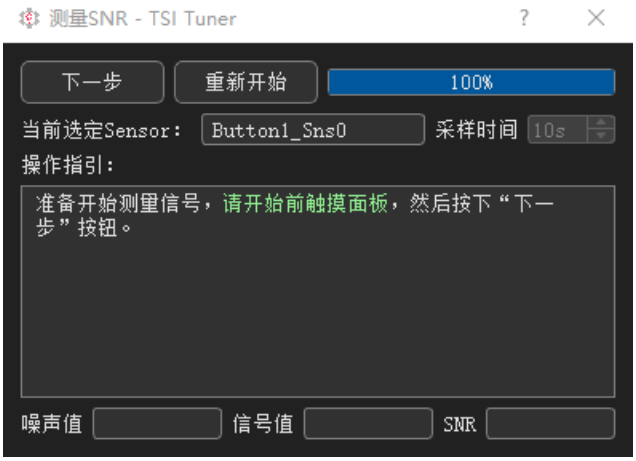


图 5.6 测量阶段 3

6. 正在测量信号值，请不要放开触摸；

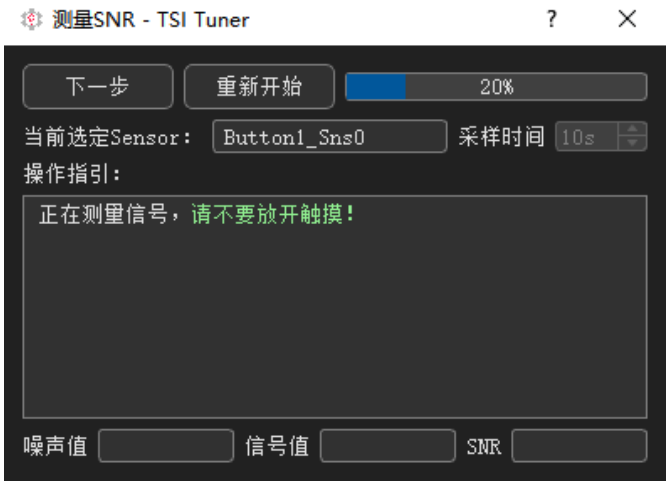


图 5.7 测量阶段 4

7.测量完成，SNR 具体数值可见下方文本框，若需测量其他通道，关闭窗口，进行重新选择后再次打开 SNR TEST;

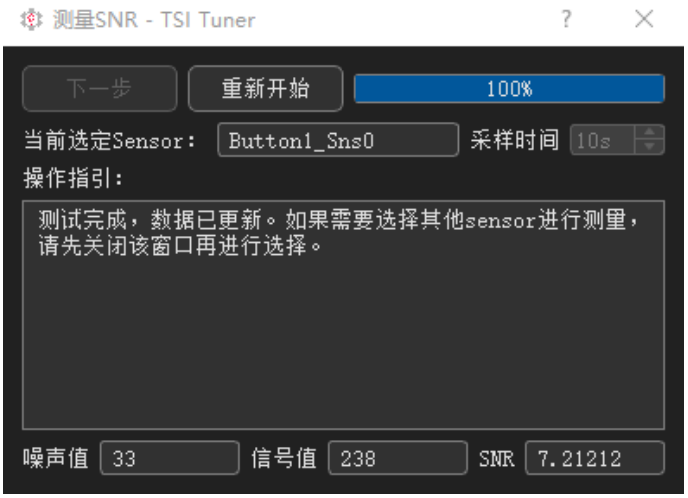


图 5.8 测量阶段 5

5.2.2 其他方法评估 SNR

参照 SNR 计算原理，使用 J_Scope/Canoe/Tsmaster 等工具，按如下步骤进行评估：

1. 首先未触摸时统计 10s 内软件滤波之后 Rawcount 的最大最小值之间的差值，记为 Noise；
2. 触摸 Sensor 表面统计 10sRawcount 的动态数据，对其取平均值，减去 Baseline 得到 Signal；
3. 使用 Signal/Noise 得到 SNR,评估 SNR 是否满足要求。

6. 软件库和算法

复旦微电子提供的 TSI 模块配套软件库提供一套完整的基于中断的按键扫描应用库，结合复旦微的 TSITuner©上位机软件，为用户提供了快速实现触摸按键的方式。目前软件库支持的控件包括按键控件、接近感应控件（可以用来制作非接触按键）和滑条控件，可以覆盖大多数常用应用。

(适用于 FM33FT0xxA/FM33FR0xx)

6.1 TSI 软件库文件结构

TSI 软件库为单目录结构，库中的文件罗列如下：

1. **tsi.c，tsi.h**：包含库的运行控制函数（启动、停止等）；
2. **tsi_baseline.c，tsi_baseline.h**：包含基准线算法相关实现；
3. **tsi_calibration.c，tsi_calibration.h**：包含触摸硬件参数自动校准算法的相关实现；
4. **tsi_conf.h**：用户配置文件，TSI 软件库的一些可配参数都罗列于此。我们仅提供对应该文件的模板**tsi_conf_template.h**，用户修改后重新命名为**tsi_conf.h**即可使用；
5. **tsi_def.h**：包含库所需的一些定义，包括一些配置可选参数值的定义；
6. **tsi_filter.c，tsi_filter.h**：包含 RawCount 滤波器的实现；
7. **tsi_objects.c，tsi_objects.h**：tsi_objects.c 包含全部的 TSI 模块控件定义和参数，tsi_objects.h 包含全部 TSI 软件库数据结构定义；**tsi_objects.c** 我们仅提供对应的模板 **tsi_objects_template.c**，用户修改后重新命名为 **tsi_objects.h** 即可使用；
8. **tsi_sensor.c，tsi_sensor.h**：包含传感器使用的配置、初始化、更新状态函数；
9. **tsi_widget.c，tsi_widget.h**：包含控件使用的配置、初始化、更新状态函数；

6.2 TSI 软件库使用方法

用户需要首先完成软件库基本配置以及控件配置，才能正常使用软件库。需要配置的文件为 tsi_objects.c 和 tsi_conf.h。接下来我们结合相关文件来梳理一下 TSI 软件库的基本数据结构和运行逻辑，方便您快速上手。特别建议您使用我们推出的 TSITuner©上位机软件，可以使用图形化方式自动生成这配置文件和对应的调参专用工程，可以极大地方便您的控件配置和调参工作，加快产品设计的速度。

tsi_objects.c 文件中包含所有 TSI 软件库中用到对象的定义。TSI 软件库在 RAM 和 ROM 中分别占据一块连续的空间，下图详细的展现了 TSI 软件库在 RAM 和 ROM 中的内存分布以及各数据结构的定义：

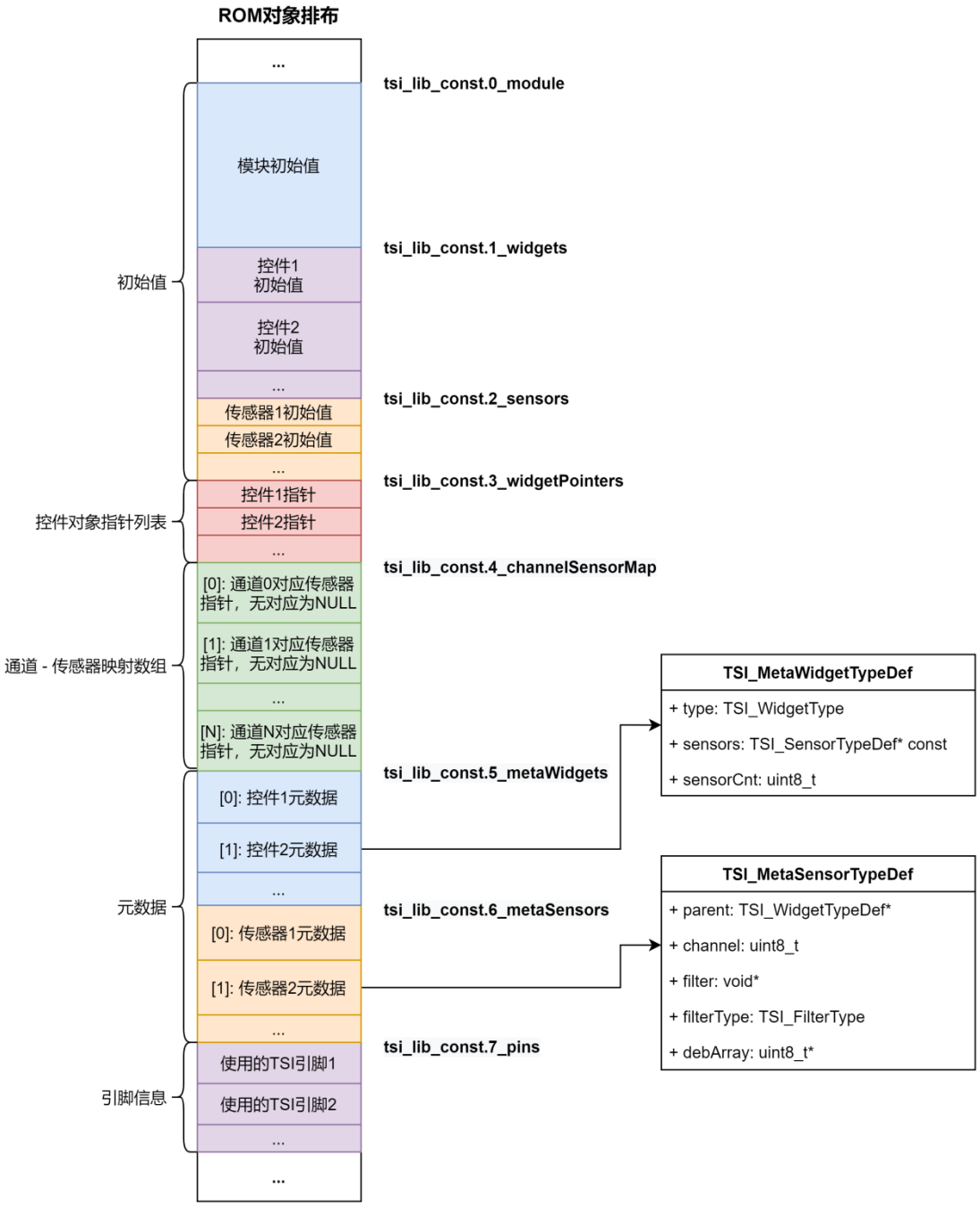


图 6.1 TSI 软件库在 RAM 和 ROM 中的内存分布

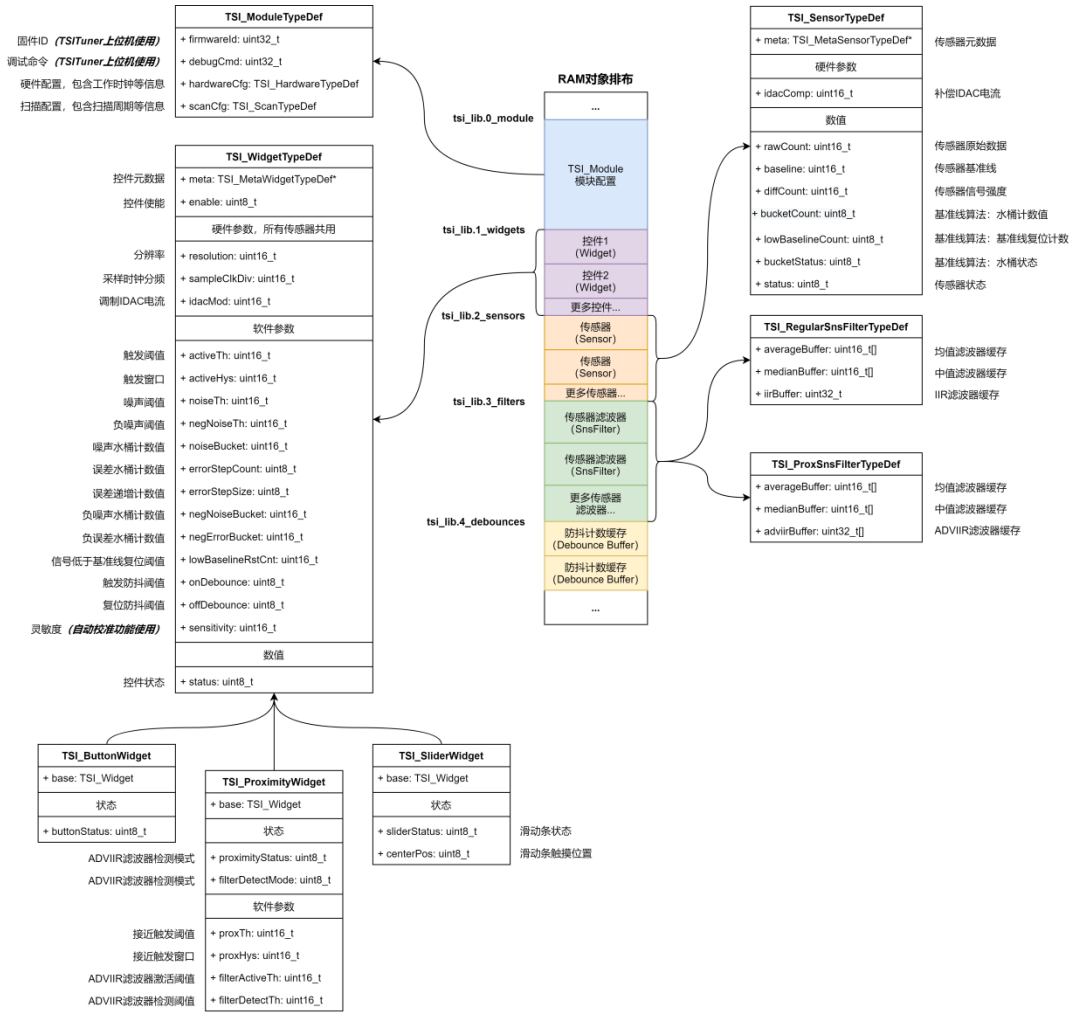


图 6.2 TSI 软件库的数据结构

上图中所有的数据结构定义可以在 tsi_objects.h 中找到，而 tsi_objects.c 中则定义了所有的 RAM/ROM 对象。

TSI 软件库的核心对象为模块（**TSI_Module**）、控件（列表，**TSI_WidgetList**）和传感器对象（列表，**TSI_SensorList**），它们在逻辑上拥有层次关系，一个模块对象拥有一个或多个控件对象，每个控件对象又拥有一个或多个传感器对象。**tsi_objects.c** 中，对每个对象都有相应的初始值定义，这些初始值在 TSI 软件库初始化的时候加载到位于 RAM 中的对应对象中。

为了维护核心对象间的层次关系和提供可拓展性，每个控件（Widget）和传感器（Sensor）对象都有相应的元数据信息：**控件的元数据**包括**控件类型**和**所拥有的传感器对象**等；**传感器的元数据**则包括其**父对象（控件对象）**和**滤波器指针**

和类型信息、**消抖缓存指针**和**通道**信息。控件通过元数据信息实现**多态**，不同控件间使用控件类型区分，具有很好的拓展性。

tsi_conf.h 文件中包含 TSI 软件库的功能配置、控件和通道使用信息以及控件和传感器列表的数据结构定义。

TSI 软件库共有 4 种执行状态：复位状态，初始状态，运行状态，暂停状态。状态机如下图所示：

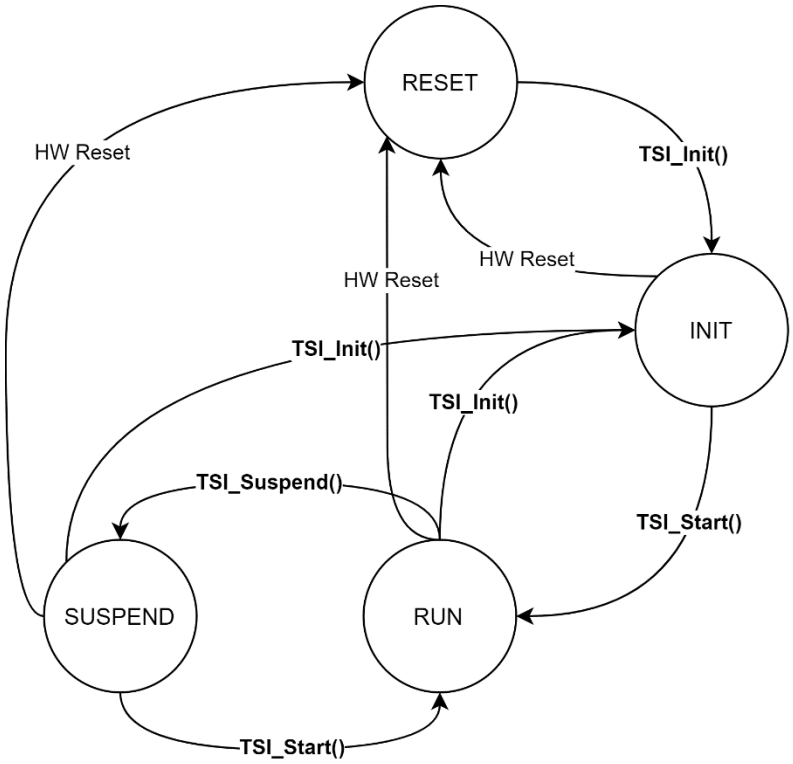


图 6.3 执行状态

- **TSI_Init()函数：初始化**
  - 首次执行初始化时，初始化时钟和使用的引脚；
  - 使用初始化参数初始化各个 TSI 对象；
  - 根据 TSI 对象的配置，将硬件参数应用到硬件；
  - 执行硬件参数自动校准（在使能校准情况下）
  - 执行初次扫描，初始化基准线和滤波器；

- **TSI_Start()函数：启动**
  - 使能中断并开始扫描
- **TSI_Suspend()函数：暂停**
  - 关闭扫描并禁止中断
- **TSI_Resume()函数：继续，功能同 TSI_Start()函数**

TSI 软件库在启动后便会持续进行扫描，TSI 软件库使能了 TSI 硬件的序列完成中断和通道完成中断，用于接收和处理数据。相关代码可以在 **tsi_interrupt.c** 中找到。在 TSI 序列完成中断触发后，TSI 软件库会置位**扫描完成标志**，可以通过执行 **TSI_GETSTAT_SCAN_CPLT()**获取该标志位状态，并使用 **TSI_CLRSTAT_SCAN_CPLT()**来清除该标志位。当用户获取到**扫描完成标志**置位之后，需要执行 **TSI_Widget_UpdateAll()**函数，该函数会依据获取到的扫描数据对所有控件进行一轮更新。之后，用户就可以根据当前控件的状态来执行操作了。各类控件获取状态的方法如下：

1. 按钮控件（Button）：通过 **TSI_WidgetList.[控件名].buttonStatus** 获取状态，0 为无事件，1 为有触摸事件；
2. 滑条控件（Slider）：通过 **TSI_WidgetList.[控件名].sliderStatus** 获取状态，0 为无事件，1 为有触摸事件；通过 **TSI_WidgetList.[控件名].centerPos** 获取滑条触摸位置，范围为 **0-255**；
3. 接近感应控件（Proximity）：通过 **TSI_WidgetList.[控件名].proximityStatus** 获取状态，0 为无事件，1 为有接近事件，2 为有触摸事件；

6.3 TSI 相关算法

6.3.1 RawCount 滤波器

6.3.1.1 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：

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

6.3.1.2 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：

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

6.3.1.3 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。

6.3.1.4 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**；

6.3.2 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。

6.3.2.1 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。*

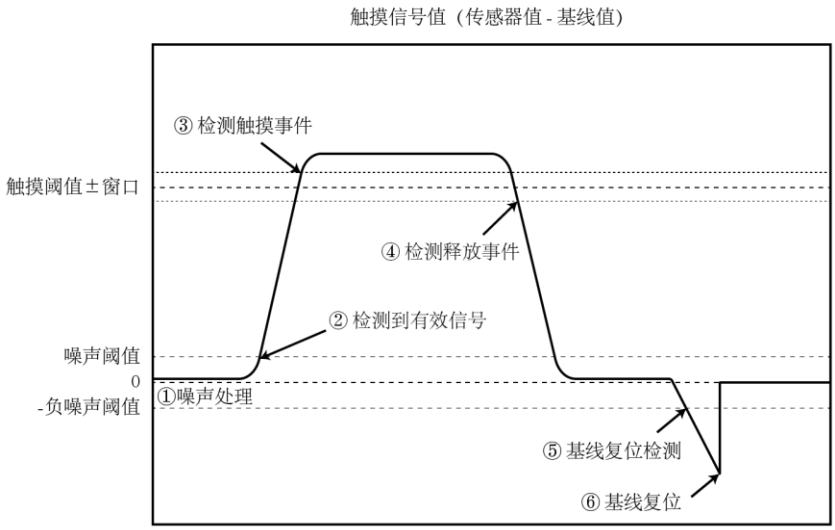


图 5.4 触摸 RawCount 变化图

10. **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。

11. **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。*

12. **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。
13. **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。
14. **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。

6.3.2.2 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。

修订历史

版本	时间	修改
V1.0	2023.08.04	首次发布
V1.1	2024.03.15	1.修改通道电阻推荐值大小 2.新增小档位配置说明（上位机工程配置以及参数调节推荐） 3.添加量产前建议测试流程相关文档
V1.2	2024.05.16	1.添加互容实现原理 2.添加互容传感器应用设计
V1.3	2024.07.07	1.添加 TSI SNR 评估指南 2.修改手动调节建议配置参数（1.2uA 建议配置 2、3、4、5|300nA 建议配置 8、12、16、20）