GMF03x 触摸感应应用指南

版本：**1.0**

发布日期：**2020-04-28**

深圳市汇顶科技股份有限公司

商标声明

GOODIX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档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。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）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。GOODIX 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。

未经 GOODIX 书面批准，不得将 GOODIX 的产品用作生命维持系统中的关键组件。在 GOODIX 知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦 B 座 2 层、13 层

电话：+86-755-33338828　　传真：+86-755-33338830

网址：http://www.goodix.com

前言

编写目的

本文档主要介绍基于 GMF03x MCU 触摸感应应用的软、硬件设计要点，以帮助用户更好地使用 GMF03x MCU 进行触摸感应应用开发。

阅读本文档时，可参考《GMF03x 触摸感应控制库（TSCL）使用手册》与《GMF03x 硬件应用设计规范》。

读者对象

本文适用于以下读者：

- GMF03x 应用开发者
- 项目或产品经理
- 开发爱好者

版本说明

本手册为第 1 次发布，对应的硬件版本为 GMF03x-DK-Touch Rev.A，软件版本为 GMF0xx SDK V1.3.2。

修订记录

版本	日期	修订内容
1.0	2020-04-28	首次发布。

目录

1 简介

GMF03x MCU 的触摸感应控制器（Touch Sensor Controller，TSC）模块具有 24 个触控通道和 1 个防水通道，支持硬件检测、防水（Shielding）等功能，支持低功耗工作模式等，可满足各种触摸感应应用场景需求。

另外，还提供了一套基于 TSC 的软件解决方案—GMF03x 触摸感应控制库（Touch Sensor Controller Library，TSCL）。GMF03x TSCL 可支持多种触摸感应传感器（如按键传感器、滑条传感器、旋钮传感器）配置，按压互斥、按压超时检测、滤波等功能配置，为基于 GMF03x MCU 的触摸感应应用开发提供软件开发支持。

GMF03x MCU 触摸感应应用解决方案的框图如下：

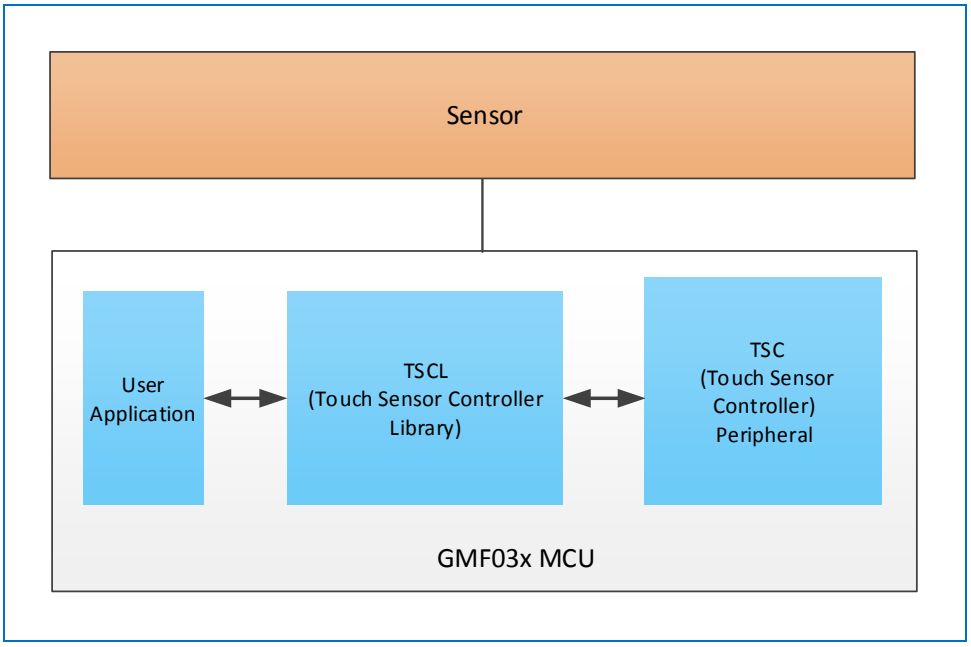

图 1-1 GMF03x MCU 触摸感应解决方案的应用框图

本文主要介绍使用 GMF03x MCU 实现触摸传感器应用时，在软、硬件设计方面的功能配置要点、注意事项、设计准则等，以指导用户快速地进行触摸传感器应用开发。

2 触摸传感器软件设计

本章主要介绍基于 GMF03x TSCL 进行软件设计时的功能配置要点、注意事项、设计原理等，以指导用户快速完成触摸传感器的软件设计。

2.1 配置触摸传感器功能

根据触摸按键的实际应用场景，GMF03x TSCL 提供多种按压事件和按压操作特性以及特殊的数据处理功能，用户可按需进行功能配置。

2.1.1 配置传感器参数

为保证传感器正常工作，用户在使用按键功能之前，需配置传感器参数并初始化传感器。

初始化传感器的步骤如下：

1. 在 tscl_capture_channel_t 中配置传感器 keypad_idx 与 channel_idx 的映射关系。
2. 在 tscl_capture_handle_t 中配置传感器的 coeff 参数、唤醒通道、低功耗模式下的刷新周期和正常模式下的刷新率。
3. 在 tscl_capture_bank_t 中配置每个数据块的通道数量，初始化通道映射关系、采集数据和滤波数据的存放地址。
4. 定义一个 tscl_algo_internal_t 结构体，用于存储滑条和旋钮的算法处理信息。先将该结构体初始化为 0，再将其传入算法调用接口函数中。
5. 在 tscl_algo_ctx_t 中设置每个传感器的算法计算参数信息，如：传感器电极类型的配置信息、按压检测阈值以及 tscl_algo_internal_t 结构体地址。其中，传感器的电极类型在结构体 tscl_algo_electrodes_mode_t 中定义。滑条和旋钮传感器需初始化一个算法计算句柄，而按键传感器仅需将其初始化为 NULL。
6. 在 tscl_capsensor_ctx_t 中配置每个传感器的上下文参数，如传感器的通道数、各状态下的防抖次数、按压检测超时时间，以及指向传感器算法参数的 tscl_algo_ctx_t 句柄。
7. 在 tscl_capsensor_t 中配置每个传感器的属性参数，如传感器编号（id）、类型，以及初始化传感器的上下文参数。如果传感器用作防护传感器，还需将相应的防护传感器标志位置 1。
8. 在 tscl_sensor_group_t 中对用户定义的传感器进行分组管理，配置传感器数量和对应指针。
9. 在 tscl_system_t 中配置 tscl_capture_handle_t 句柄和传感器校准采样次数。

完成上述配置后，用户需调用相应的初始化函数，将硬件配置参数写入 TSC 寄存器，从而实现触摸感应功能。

说明：

- GMF03x TSC 最多可配置 4 个硬件检测通道。
- 按键、滑条和旋钮传感器的电极类型配置不相同。
- *gm_tscl_config.h* 文件中包含各功能的宏定义和各参数的宏定义，用户可直接使用，无需再另外定义宏。

2.1.2 配置软件滤波功能

在噪声应用环境中，触摸感应传感器需具备较强的抗干扰能力，即不会因外部噪声干扰而导致原始数据异常波动，进而出现冒键。GMF03x TSCL 提供了软件滤波功能，主要采用均值滤波和 IIR 滤波。

软件滤波功能的配置步骤如下：

1. 使能滤波器模块，打开宏定义：#define TSCL_DENOISE_FILTER。
2. 定义一个 tscl_algo_denoise_t 结构体，用于存储滤波所需的数据。如：tscl_algo_denoise_t g_filter[TSCLP_TOTAL_CH] = {0}，其中 TSCLP_TOTAL_CH：需滤波处理的通道数。
3. 将 g_filter 通过静态数据初始化到结构体 tscl_capture_bank_t 的成员变量 p_filter_data 中。
4. 根据实际应用场景，选择滤波方式，通过接口函数 tscl_status_t tscl_sensor_capture_data(tscl_system_t *p_system, const tscl_capture_bank_t *p_banks, tscl_num_t banks_num, p_tscl_algo_filter_t p_filter)配入系统框架中。将需使用的均值滤波接口函数或 IIR 滤波接口函数写入 tscl_sensor_capture_data 的第四个参数中。

2.1.3 配置防异物覆盖功能

GMF03x TSCL 支持按压超时检测功能，可防止 Touch 面板由于表面覆盖水或其他异物而导致异常。当某个传感器的按压时间超过了按压检测超时时间时，系统将以当前采集值为基准，重新扫描该传感器通道并判断是否有按压。

防异物覆盖功能的配置步骤如下：

1. 使能按压超时检测模块（DTO），打开宏定义：#define TSCL_FEATURE_DTO。
2. 将按压检测超时时间写入结构体 tscl_capsensor_ctx_t 的成员变量 pressed_timeout 中，默认按压超时检测的时间单位为秒。
3. 初始化 TSCL 时，会将相应的初始化信息写入框架代码中。

2.1.4 配置按压互斥检测功能

GMF03x TSCL 支持按压互斥检测功能，可防止多个传感器同时处于按压态。即：当检测到一个传感器处于按压态时，其他传感器将进入触摸态，以保证同一时间仅有一个传感器响应用户操作。

按压互斥检测功能的配置步骤如下：

1. 使能按压互斥检测功能，打开宏定义：#define TSCL_FEATURE_EXCLUSIVE。
2. 先调用传感器状态机接口函数 tscl_sensor_group_process()，筛选出所有被按压的通道，再调用按压互斥功能的接口函数 tscl_sensor_arbitrate_multiple_touch()，确定一个通道（对应被按压的传感器）响应按压，并将其他被按压的传感器的状态转换为触摸态。

2.1.5 配置按压事件功能

GMF03x TSCL 支持各种按压事件（如：单击、双击、长按、左滑、右滑、顺时针和逆时针滑动等）的处理功能，以满足用户不同的应用场景需求。

按压事件功能的配置步骤如下：

1. 使能传感器的事件：若使能按键传感器事件，则需打开宏定义：#define TSCL_FEATURE_TOUCHKEY_EVENT；若使能滑条与旋钮事件，则需定义宏 TSCLP_TOTAL_LINROT_SENSOR 的值大于零。
2. 使能按压事件功能后，系统会将按压事件记录在结构体 tscl_linrot_ctx_t 和 tscl_tkey_ctx_t 的 event 成员变量中。

2.2 物理通道与逻辑通道映射

GMF03x TSC 提供 24 个触控通道，用于采集传感器的数据。TSCL 中有 24 个物理通道（TSCL_KEYPAD0 ~ TSCL_KEYPAD23）和 24 个逻辑通道（TSCL_CHANNEL0 ~ TSCL_CHANNEL23）。物理通道与底层硬件的 GPIO 引脚相对应，逻辑通道是上层软件处理数据时所使用的通道。

用户可配置物理通道与逻辑通道的映射关系。例如：将传感器的 key0 连接至芯片引脚 PA2（参见表 2-1，PA2 对应的物理通道为 PAD0），若 PAD0 映射到 channel0，则表示 channel0 中存储的数据为 key0 的采集数据。

GMF03x 的 GPIO 与物理通道的映射关系如下：

表 2-1 GPIO 与物理通道的映射

GPIO	物理通道	GPIO	物理通道
PA2	PAD0	PB13	PAD12
PA3	PAD1	PB14	PAD13
PA4	PAD2	PB15	PAD14
PA5	PAD3	PA8	PAD15
PA6	PAD4	PA9	PAD16
PA7	PAD5	PA10	PAD17
PB0	PAD6	PA11	PAD18
PB1	PAD7	PA12	PAD19
PB2	PAD8	PA15	PAD20
PB10	PAD9	PB7	PAD21
PB11	PAD10	PB8	PAD22
PB12	PAD11	PB9	PAD23

物理通道与逻辑通道的映射关系，可通过结构体 tscl_capture_channel_t 配置。该结构体的详细信息，请参考《GMF03x 触摸感应控制库（TSCL）用户手册》中的“4.1 数据通道（Channel）”。

物理通道与逻辑通道映射关系的配置步骤如下：

1. 定义一个 tscl_capture_channel_t 类型的结构体。

```
static const tscl_capture_channel_t g_capture_channels[TSCLP_TOTAL_CH] /* TSCLP_TOTAL_CH 表示通道总数*/
```

2. 根据物理通道与逻辑通道的映射关系，初始化结构体。

```
static const tscl_capture_channel_t g_capture_channels[TSCLP_TOTAL_CH] =
{
    TSCL_KEY_MAP(TSCL_KEYPAD22, TSCL_CHANNEL22),    /*KEY22 映射到 channel22*/
    TSCL_KEY_MAP(TSCL_KEYPAD21, TSCL_CHANNEL21),    /*KEY21 映射到 channel21*/
    ……
}
```

3. 调用 tscl_capture_init()函数，将映射关系的初始化信息写入 TSC 模块的寄存器中，以完成配置流程。

2.3 按压互斥应用

GMF03x TSCL 支持按压互斥功能，可防止多个传感器同时处于按压态。

按压互斥功能的实现原理为：当检测到多个传感器处于按压态时，选择一个传感器（传感器组中的第一个传感器具有最高优先级）响应并输出按压态，其他传感器被“锁定”（设置 DXSlock 标志）并进入触摸态。一旦按压态的传感器被释放，则处于触摸态的最高优先级传感器将被置为按压态。

按压互斥功能采用以下规则筛选被按压的传感器：

- 最先检测到按压的传感器（即先被按压的传感器）将作为互斥逻辑筛选出的被按压的传感器。
- 若多个传感器同时（无时间差）被按下，则选择初始化时传感器编号（id）最小的传感器作为被按压的传感器。

使用互斥功能时，需注意以下事项：

- 使用互斥功能之前，必须先调用状态机函数 tscl_sensor_group_process()确定传感器状态，然后再调用 tscl_sensor_arbitrate_multiple_touch()确定响应按压的传感器。
- 防护传感器不能用于按压互斥。
- 若没有使能按压超时检测功能，则通过按压互斥功能选择的传感器将一直处于按压态，直至该传感器被释放，然后其他处于触摸态的传感器将被转换为按压态。

2.4 防水应用

GMF03x TSCL 防护传感器（Guard Sensor）支持防水功能。该防水设计可有效保证触摸感应传感器产品在薄雾、湿气、水滴和水流等环境下仍具有良好的可靠性，避免误将水流识别为手指按压。

防护传感器（Guard Sensor）的设计框图如下：

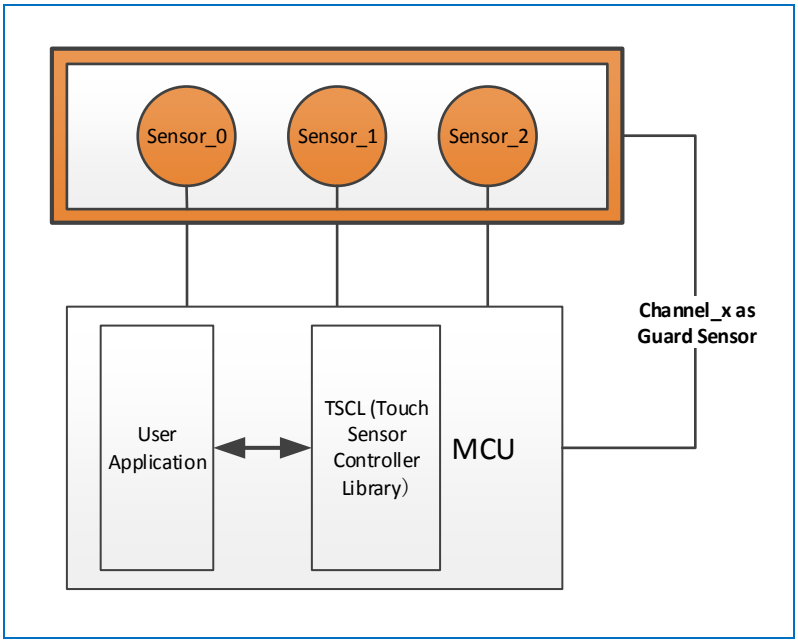

图 2-1 防护传感器设计框图

GMF03x TSCL 提供了利用防护传感器避免误触发的软件处理方法：当水滴较小时，触摸感应功能可正常使用；当出现大量水流时，通过分析防护传感器的数据，禁能其他传感器。

防护传感器功能的配置步骤如下：

1. 在 tscl_capture_channel_t 中配置防护传感器 keypad_idx 与 channel_idx 的映射关系。
2. 在 tscl_capture_handle_t 中配置每个防护传感器的 coeff 参数、唤醒通道、低功耗模式下的刷新周期和正常模式下的刷新率。

3. 在 tscl_capture_bank_t 中配置每个防护传感器数据块的通道数量，初始化通道映射关系、采集数据和滤波数据的存放地址。
4. 在 tscl_algo_ctx_t 中设置每个传感器的算法计算参数信息，如：传感器电极类型的配置信息、按压检测阈值以及 tscl_algo_internal_t 结构体地址。防护传感器的电极类型为：TSCL_ALGO_ELECTRODES_MODE_SINGLE，只有滑条和旋钮传感器需初始化一个算法计算句柄，而防护传感器仅需将其初始化为 NULL。
5. 在 tscl_capsensor_ctx_t 中配置每个防护传感器的上下文参数，如：传感器的通道数、各个状态下的防抖次数、按压检测超时时间，以及指向传感器算法参数的 tscl_algo_ctx_t 句柄。通常防护传感器的通道数为 1。
6. 在 tscl_capsensor_t 中配置每个防护传感器的属性参数，如：传感器编号、类型，以及初始化传感器的上下文参数。通常，防护传感器组号与其所保护的其他传感器的组号保持一致，类型为 TSCL_SENSOR_TOUCHKEY。另外，防护传感器标志需初始化为 1。
7. 在 tscl_sensor_group_t 中对用户定义的防护传感器进行分组管理，配置传感器数量和对应指针。
8. 在 tscl_system_t 中配置 tscl_capture_handle_t 句柄和传感器校准采样次数。

完成上述配置后，用户调用相应的初始化函数，将相应的结构体写入函数的形参中，系统将初始化硬件寄存器，把所有的参数信息写入寄存器中，从而实现防水功能。

使用 TSCL 防护传感器功能时，需注意以下事项：

- 如果用户使用防护传感器，则需配置防护传感器的数量，即配置宏定义 TSCLP_TOTAL_GUARD_SENSOR 的值。
- 防护传感器不能用于按压互斥功能检测、基准更新。
- 配置防护传感器时，其传感器组号与它所保护的传感器的组号需保持一致。传感器组号为防护传感器识别需保护哪些传感器的标志。
- 初始化防护传感器时，需将结构体 tscl_capsensor_t 中的成员变量 guard 初始化为 1，以表明该传感器为防护传感器，否则系统将无法识别防护传感器。

2.5 查询与中断应用

触摸感应控制库（TSCL）支持两种数据采集模式：查询模式和中断模式。用户可在 *gm_tscl_config.h* 文件中配置数据采集模式。

- 查询模式（关闭中断模式的宏）：//#define TSCL_INTERRUPT_MODE

在查询模式下，采集数据的方式为阻塞式采集。在该模式下，TSCL 需等待硬件完成数据采集（寄存器的完成标志位被置位，表示所有使能通道的数据采集已完成）后，才能读取数据。若完成标志位未被置位，程序将不会返回。因此，TSCL 将消耗部分时间以等待硬件完成标志被置位。

查询模式无需硬件中断支持即可完成数据采集，但由于检查条件需占用 CPU 时间，其效率低于中断模式。

- 中断模式（打开中断模式的宏）：#define TSCL_INTERRUPT_MODE

在中断模式下，采集数据的方式为非阻塞式采集。在该模式下，TSCL 主动获取数据和检查采集完成标志是否被置位，而无需等待硬件采集完成标志被置位。若采集完成标志未被置位，则立即返回，将 CPU 交还给其他应用；若采集完成标志被置位，则立即读取并处理采集的数据。另外，当硬件完成采集时，将触发中断，并在中断服务程序中缓存各通道的采集数据，以等待 TSCL 读取，从而防止各数据帧之间产生串扰。

对于硬件采集频率较低的应用场景，建议采用中断模式，将多余的时间留给 CPU 处理比较耗时的逻辑，以提升效率。

2.6 按压事件检测应用

GMF03x TSCL 触摸感应应用提供了按压事件检测功能：位置计算、手势识别。

- 按压位置计算

按压位置计算是进行手势识别的核心与基础。进行按压位置计算需先对传感器建立坐标，以线性滑条为例构建坐标，如图 2-2 所示。

假定线性滑条上一共有 n+1 个触摸按键，分别为 Key_0，Key_1，Key_2，…，Key_n。

- 将 Key_0 的起始位置设为 0，Key_n 的结束位置设为最大值 255。
- 每个 Key 的相对距离为 256/（n+1）。

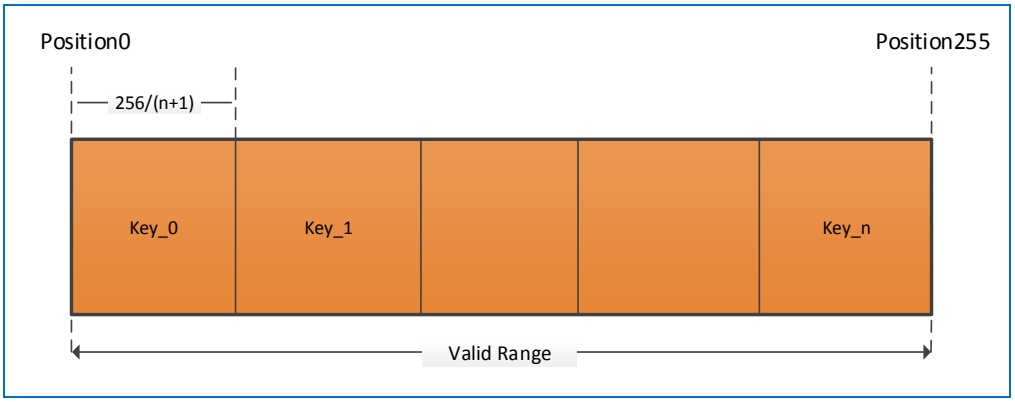

图 2-2 滑条间距坐标示意图

- 手势识别

手势识别是基于位置计算的一种手势识别方法，其原理为：记录多次按压或者非按压状态，通过按压或者非按压次数与按压位置的变化情况来识别动作。使用该方法时，用户可根据实际数据采集频率灵活设置相关参数：

- 手势识别事件参数：单击事件、双击事件、长按事件和滑动事件（左滑、右滑、顺时针滑动、逆时针滑动）。
- 事件识别判断值和阈值：单击判断值、双击判断值、长按判断值、长按阈值、滑动阈值。

当检测到两个或两个以上按键的采集值超过按压阈值，且按键的位置不相邻时，将不输出任何事件。

2.7 低功耗唤醒应用

GMF03x TSC 支持可配置的低功耗模式，可满足低频次触摸的应用场景需求，极大程度地降低设备功耗。使用该模式时，最多可配置 4 路通道用于低功耗模式唤醒检测。用户可根据使用场景配置低功耗下的检测刷新周期，以实现应用功耗最低。

- 配置低功耗通道唤醒：可从 GMF03x TSC 的 24 个通道中任选 1 ~ 4 个通道进行配置。在 TSC 的低功耗模式下，将定时对低功耗唤醒的通道进行触摸检测，而非低功耗唤醒的通道则不会进行触摸检测，即不响应任何触摸事件以降低功耗。
- 配置低功耗检测刷新时间：刷新时间可根据应用场景和功耗的要求进行配置，单位为毫秒。该刷新时间对所有配置的低功耗唤醒通道有效。

- 配置低功耗通道唤醒阈值：该阈值为低功耗唤醒检测的阀值。将低功耗下采集的通道数据与该阈值相比较：若采集值高于硬件唤醒上阈值或低于硬件唤醒下阈值，表示通过硬件检测；反之，则没有通过检测。
- 配置低功耗硬件防抖检测：用户可以配置硬件检测通过的条件。该配置包含两个参数：检测次数和通过次数。检测次数是指在低功耗模式下对通道检测多少次；通过次数是指在检测的次数中得到的检测值与预先设定的阈值相比较后，通过了阈值设定值的次数。

2.8 通道数据归一化

采集数据时，由于电容触摸传感器通道的电容差异，需对采集的通道数据进行归一化处理，以便于 TSCL 进行数据处理。例如：当系统基准校准到 2048 时，在相同按压条件下，每个通道采集的数据变化量存在不一致的情况，需对通道数据进行归一化处理。本系统采用 min-max 标准化（Min-Max Normalization）方法进行归一化，也称为离差标准化，是对原始数据的线性变换，使结果值映射到[0 - 1]之间。转换函数如下：

$$X^* = \frac{X - min}{max - min}$$

其中 max 为样本数据的最大值，min 为样本数据的最小值。

GMF03x TSCL 只接受归一化数据，以确保在同一变化量关系上基于每个通道数据对按压、滑动等动作进行识别。为便于 TSCL 处理，要求将采样数据映射到 0 ~ 300 范围之内。需注意的是，本归一化并不要求每个数据均落入该范围，但建议误差不超过 max 的 10%。通道数据采集完成后，才能开始数据归一化处理。为简化数据运算，归一化数据可由原始采样值乘上通道归一化系数获得。归一化系数由 GTouchInsight 根据 min-max 算法计算获得。

为了不引入浮点运算，TSCL 中的通道归一化系数由 tscl_coeff_t 结构体定义。

```
typedef struct _tscl_coeff_t
{
    uint8_t numerator;      /*!< Coefficient numerator */
    uint8_t denominator;    /*!< Coefficient denominator */
} tscl_coeff_t;
```

通道归一化系数由 tscl_capture_handle_t 结构体管理，每个通道对应一组归一化系数。由于不同芯片或同一芯片的不同通道均存在差异，因此，归一化系数需基于芯片使用 GTouchInsight 工具进行校准，然后生成配置文件。在对精度要求不高的应用中，可不一一校准每颗芯片，而是采用固定校准系数。

```
#define TSCL_CHANNEL_COEFF（channel, numerator, denominator） .coeff[channel]
= {numerator, denominator}
static tscl_capture_handle_t g_capture_handle =
{
    TSCL_CHANNEL_COEFF（TSCL_CHANNEL0, 1, 1）, /*channel 0 coeff*/
    ……
};
```

2.9 阈值设置

触摸感应传感器通过检测外部电容按键的电容值变化识别手指按压。当手指逐渐靠近/远离按键时，ADC 模块采样值的变化曲线如图 2-3 所示。

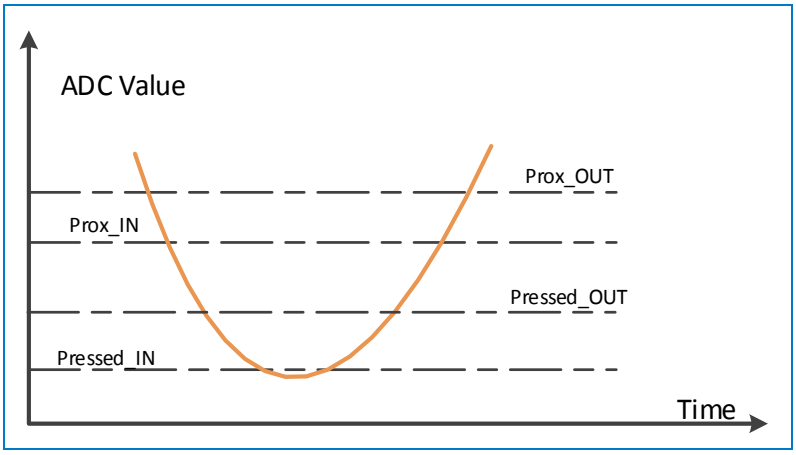

图 2-3 ADC 采样值的变化曲线

为了识别手指按压的变化，TSCL 中定义了以下四种阈值：

- 靠近阈值（Prox_IN）：判断手指处于靠近未接触状态的最大 ADC 采样值。当 ADC 采样值由高到低变化且小于 Prox_IN，大于 Pressed_IN 时，即认为手指处于靠近按键未接触状态。
- 按压阈值（Pressed_IN）：判断手指处于按压状态的最大 ADC 采样值。当 ADC 采样值由高到低变化且小于 Pressed_IN 时，即认为手指处于按压状态。
- 远离阈值（Prox_OUT）：判断手指处于远离按键未接触状态的最小 ADC 采样值。当 ADC 采样值由低到高变化且大于 Prox_OUT 时，即认为手指处于远离按键未接触状态。
- 抬手阈值（Pressed_OUT）：判断手指处于抬起状态的最小 ADC 采样值。当 ADC 采样值由低到高变化且大于 Pressed_OUT 小于 Prox_OUT 时，即认为手指处于抬起状态。

结合基准参考面，可将整个传感器采样值的变化量划分为 7 个区域（参见图 2-4），且基准更新包括正向更新（采样值高于基准值）和负向更新（采样值低于基准值）两种。

- Region A: 定时基准更新
- Region B: 正向温漂基准更新
- Region C: 负向温漂基准更新
- Region D: 正向异常漂移基准更新
- Region E: 负向异常漂移基准更新
- Region F: 异常按压基准更新、大面积按压基准更新
- Region G: 大负值基准更新

将正向 A、B、D、F 这 4 个区域标记为 5 个不同的按压等级：Level0、Level1、Level2、Level3、Level4，以适配不同的状态转换。

图 2-4 阈值区域划分

传感器阈值由结构体 tscl_algo_ctx_t 中的 detect_in_th、detect_out_th、proxy_in_th 和 proxy_out_th 指定。

```
typedef struct _tscl_algo_ctx
{
   const tscl_algo_electrodes_mode_t electrodes;      /*!< Electrode mode  */
   tscl_algo_internal_t * const internal;        /*!< Internal data context*/
   int16_t  detect_in_th;                         /*!< Detect in threshold  */
   int16_t  detect_out_th;                       /*!< Detect out threshold  */
   int16_t  proxy_in_th;                          /*!< Proxy in threshold  */
   int16_t  proxy_out_th;                        /*!< Proxy out threshold  */
} tscl_algo_ctx_t;
```

阈值设置基于归一化数据，与通道的原始数据差异无关。阈值需与实际应用场景适配，并建议按以下要求进行配置：

- detect_in_th：该值应小于归一化数据最大值的 50%。例如：若归一化数据最大值为 300，detect_in_th 应小于 150。在低功耗应用中，系统唤醒阈值由 detect_in_th 确定，若该值过大，可能会导致系统无法唤醒。
- detect_out_th = detect_in_th * 80%
- proxy_in_th = detect_in_th * 50%
- proxy_out_th = detect_in_th * 30%

总体上，需满足 detect_in_th > detect_out_th > proxy_in_th > proxy_out_th。在触摸灵敏度要求较高的应用场景中，可根据上述调校准则，调低阈值量即可。

2.10 按压超时检测

GMF03x TSCL 支持按压超时检测功能。按压超时检测机制为：从传感器检测到按压开始，直至按压检测超时时间，传感器将持续处于按压态。超时后，TSCL 自动将传感器状态从按压态转换为释放态。如图 2-5 所示的示例：当有持续按压时，通道采集了 6 个数据（存在一定范围的波动）。配置传感器的按压超时检测为 3，然后传感器将从第 4 个数据开始处理，将状态切换到释放态。

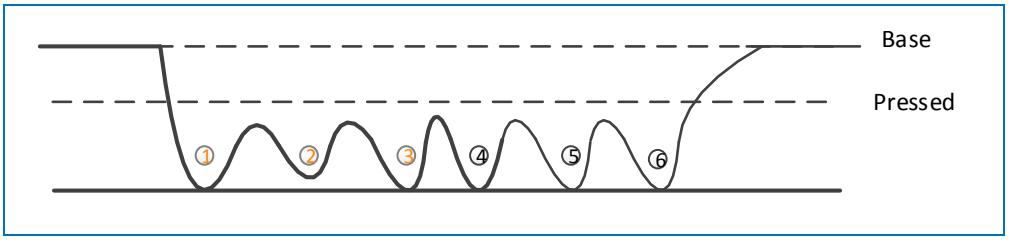

图 2-5 超时检测状态变化示例

2.11 去抖动

GMF03x TSCL 支持去抖动功能。去抖动机制为：当传感器检测到状态变化时，对采集的数据进行多次判断，符合要求后再进行后续处理。如图 2-6 所示的示例：用户配置传感器去抖动为 3，即当连续 3 个数据满足要求时才对该通道数据进行下一步处理。不处理①②③数据，从第 4 个数据开始，仅处理④⑤⑥数据，以决定按键状态。

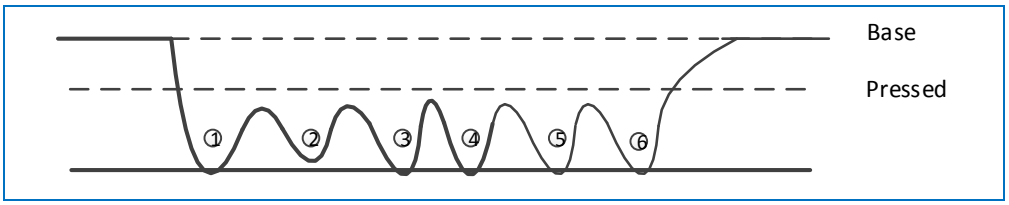

图 2-6 去抖动状态示例 1

在此示例中，如果在去抖的处理过程中，满足要求的连续数据少于 3 个，则在满足要求的下一个数据出现时重新开始去抖计数。如图 2-7 所示的示例：再次经过去抖后，仅处理④数据。

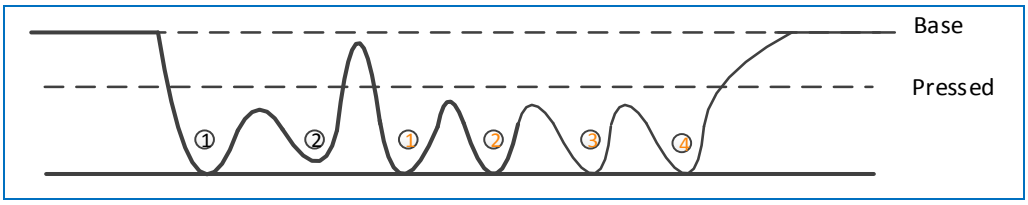

图 2-7 去抖动状态示例 2

2.12 触摸传感器半端电极位置计算

GMF03x TSCL 使用半端电极的线性传感器。

- 与全端电极的线性传感器相比，半端电极的线性传感器在传感器的两端电极具有更高的位置精度。
- 全端电极的线性传感器，如图 2-8 所示，它的起始位置（位置计算时的 Position0）在第一个电极（Key_0）的中点，结束位置（位置计算时的 Position255）在最后一个电极（Key_n）的中点。将 Key_0 到 key_n 之间的范围（Valid Range）分为 256 等份的固定值，用于 TSCL 位置计算，而 Key_0 的前半段与 Key_n 的后半段不参与位置计算（Invalid Range）。

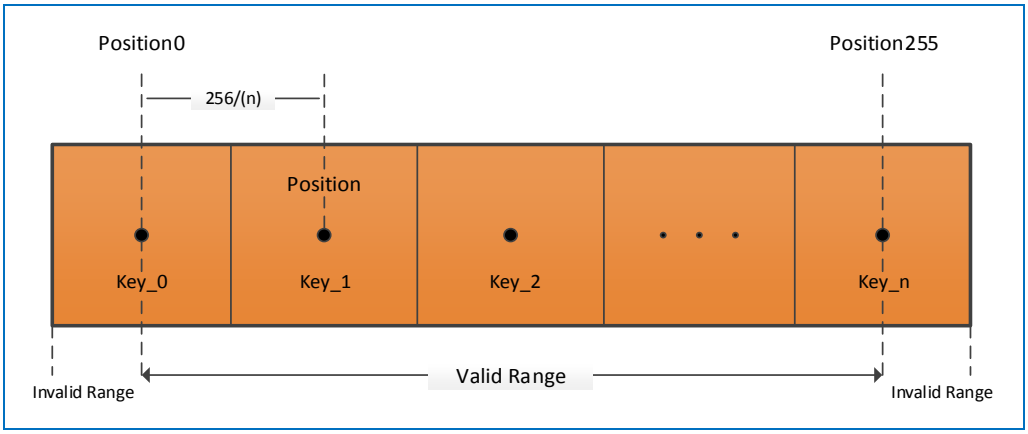

图 2-8 全端电极坐标示意图

- 半端电极的线性传感器，如图 2-9 所示，它的有效位置计算范围是从第一个电极（Key_0）的起始位置到最后一个电极（Key_n）的结束位置。半端电极线性传感器的第一个电极（Key_0）的前半段与最后一个电极（Key_n）后半段的位置可参与 TSCL 位置计算。

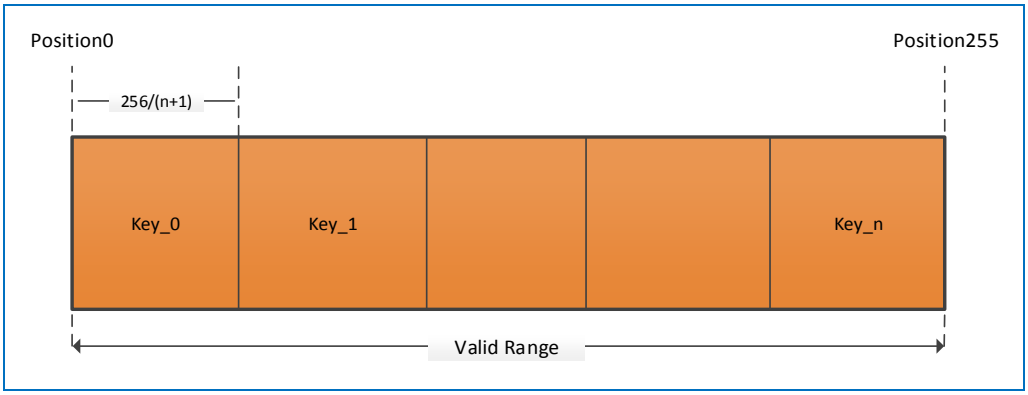

图 2-9 半端电极坐标示意图

2.13 共模噪声抑制

GMF03x TSCL 采用以下两种策略以提高共模噪声的抑制效果：

- 软件上采用滤波算法

配置方法（在 *gm_tscl_config.h* 中打开滤波器的宏定义）：#define TSCL_DENOISE_FILTER

TSCL 提供 IIR 滤波器和均值滤波器，推荐使用均值滤波器。由于均值滤波算法需记录过去多次输出值，会导致相位滞后。针对这种情况，可通过提高数据的刷新频率进行优化。采样频率可根据均值滤波算法的阶数调节，以使滤波算法不会影响正常的事件侦测。

- 硬件上增加 ADC 采样平均的次数和提高数据采集的频率

多次 ADC 采样取平均，也是一种均值滤波手段。与软件均值滤波算法不同之处在于：无需记录过去多次滤波器输出值，仅需采集多次数据求平均即可。多次 ADC 采样取平均，采用固定窗口求平均，而软件均值滤波采用的是滑动窗口求平均。

配置方法：可通过 GTouchInsight 工具配置这两项参数，并导出配置文件。

2.14 感测信号调节

GMF03x TSCL 通过调整硬件的配置可增强感测信号，从而提高触摸感应的灵敏度。主要体现在以下几

个方面：

- 充电积分次数

GMF03x TSC 采集数据时，将对硬件内部的积分电容进行充电，并分时进行多次充电，得到相对较大的电荷变化量。当手指按压按键时，将得到较大的数据变化量。增加充电积分次数可提高触摸感应的灵敏度，但将增加硬件功耗，用户需根据实际应用场景需求在灵敏度和功耗上做取舍。

- 归一化采集数据

开始采集数据之前，需调整硬件参数，以使传感器在无按压时，通道采集值调整到 2048 附近（12 位 ADC 采样的取值范围为 0 ~ 4095，建议调整到 2048，使采集值上下均有 2048 左右的裕度）。TSCL 采用二分法快速进行采集值校准，用户使能的通道数越多，校准需耗费的时间就越长。

- 灵活调整唤醒阈值

GMF03x TSC 最多支持 4 个唤醒通道，这 4 个唤醒通道的唤醒阈值可灵活配置。在不同的环境下，TSCL 的基准值可能不一致，若唤醒阈值仍然保持不变，则会导致系统唤不醒或者被误唤醒的情况。TSCL 会根据实时基准值，动态调整唤醒阈值，使得唤醒阈值与基准值的相对变化量保持不变，以提高唤醒的准确性。

3 触摸传感器硬件设计

本章主要介绍触摸传感器硬件设计的相关注意事项、设计准则等，以帮助用户快速地完成触摸传感器硬件开发。

3.1 按键传感器设计

按键传感器的典型应用是感知手指的触摸。

按键传感器的形状推荐设计为实心圆形，如图 3-1 所示。

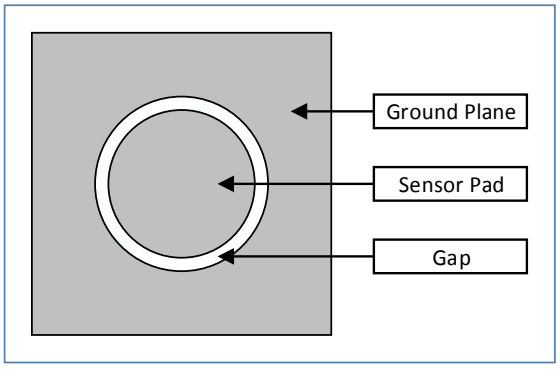

图 3-1 按键传感器

增加按键（Sensor Pad）周围的间隙（Gap）可降低寄生电容，因此，设计时需充分考虑其影响。

3.2 滑条传感器设计

滑条传感器是一个电极阵列。相邻电容元件之间的变化可用于确定按压的位置。基于重心算法，即可在按压变化中确定接触点位置。

- 滑条尺寸和形状要求

滑条段应足够小，以便于多个电极能与手指接触。滑条传感器的形状推荐设计为锯齿形，且滑条段数目最好大于等于 5。滑条的最大长度只受 GMF03x 可用 I/O 引脚的限制。

典型的滑条传感器形状如图 3-2 所示。

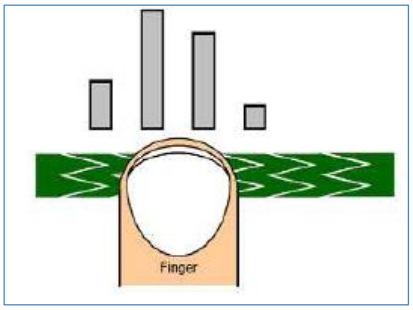

图 3-2 滑条传感器

- 滑条复用

如果 I/O 引脚数目有限，那么将两个滑条段连接至同一个 TSC PAD 脚上，即可实现感应量翻倍。TSCL 的线性滑条传感器支持半端电极位置算法，半端电极设计采用滑条复用技术实现。

3.3 防护传感器设计

GMF03x TSCL 支持防护传感器功能，其详细设计原理可参考文档《GMF03x 触摸感应控制库（TSCL）用户手册》“2.2.3 防水设计”。防护传感器是围绕所保护的传感器周围的特殊传感器。

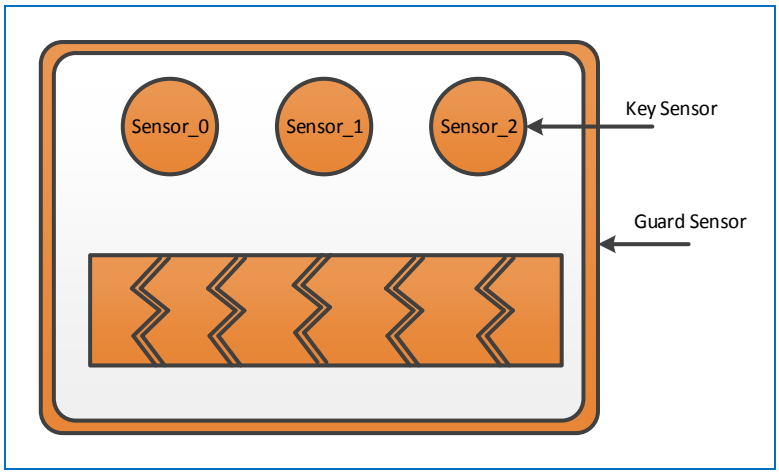

图 3-3 防护传感器示意图

设计防护传感器 PCB 时，请遵循以下原则：

- 防护传感器电极需围绕在所需被保护的电极周围。如图 3-3 所示，防护传感器围绕在按键传感器和滑条传感器周围，当触摸表面有水流覆盖时，则可以保护按键和滑条传感器不因水流而误触发。
- 防护传感器与其所需保护的传感器应保持适当的距离，以防止意外触摸到防护传感器。

设计防护传感器结构时，请遵循以下原则：

- 防护传感器的形状应为中空圆角矩形，且防护传感器应围绕所需保护的其他传感器。
- 防护传感器的厚度、以及防护传感器和 Shielding 电极之间间隙等参数的设置，请参考《GMF03x 硬件应用设计规范》。
- 建议在旋钮传感器中心位置放置防护传感器以提高防水识别精度。

3.4 硬件防水设计

传感器表面上的液体可能导致触摸或接近的错误检测，为了有效抑制液体引入的干扰，通常在硬件结构设计中需考虑驱动信号屏蔽功能。

在按压电极周围设置一层驱动信号屏蔽网状电路（如图 3-4 所示），且与电极之间的间距应不小于 2 mm。该网状电路需连接至 GMF03x 的 Shielding 通道引脚。驱动屏蔽信号与传感器开关信号具有相同的振幅、频率和相位，当液体落在传感器表面时，液体引入电容的两极保持相同的电位。因此，液体引入的电容不会使总线上产生任何附加电荷，从而起到防水作用。

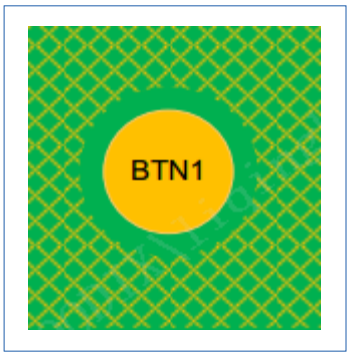

图 3-4 网状结构 Shielding

3.5 半端电极设计

半端电极传感器的结构图，如图 3-5 所示。

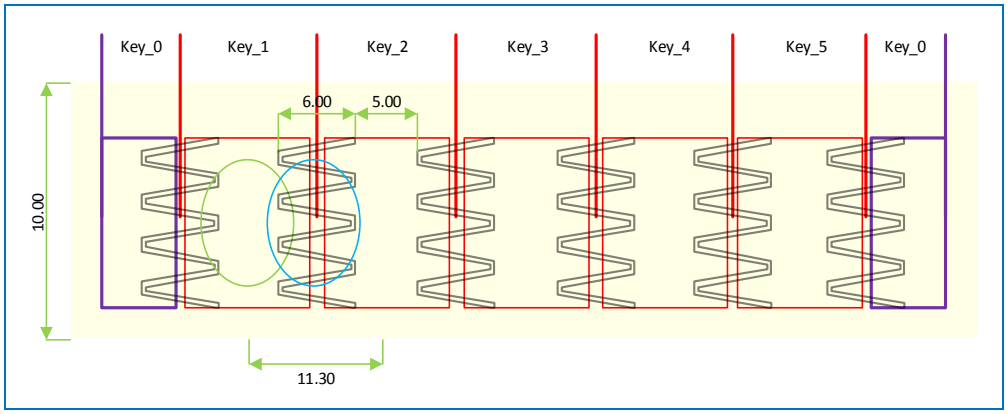

图 3-5 半端电极结构

- 图 3-5 所示的滑条传感器，为具有 6 个电极的线性半端电极传感器。需注意的是，线性半端电极传感器的起始电极（参见图 3-5 左侧的紫色方框）与末端电极（参见图 3-5 右侧的紫色方框）在电路上是连接在一起的（Key_0）。
- 起始电极与末端电极的宽度只有其他电极宽度的一半，它们连接在一起组成的 Key_0 电极与其他电极宽度一致。
- 图中的空白区域（参见图 3-5 中绿色椭圆框）是按键区，有齿的区域（参见图 3-5 中蓝色椭圆框）是耦合区（键与键之间交叠的地方）。
- 线性滑条传感器推荐尺寸：齿宽 6 mm，带宽 5 mm，键宽为 11 mm，齿数为 4 个，Sensor 高度为 10 mm，键之间的间隔为 0.3 mm。

3.6 防 RF 干扰硬件电路设计

在 Sensor 通道上串接电阻 R_S，并在近芯片端并接电容 C_P，构建一个低通滤波器，可抑制高频的 RF 干扰，如图 3-6 所示。

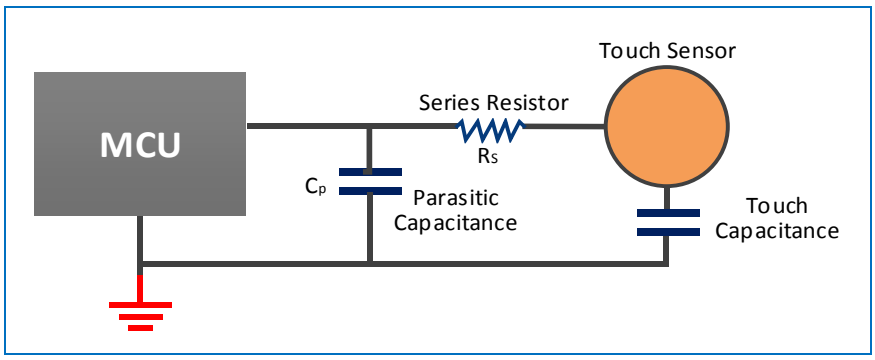

图 3-6 低通滤波器原理框图

上图为低通滤波器的原理框图，在高抗干扰应用场景中，每一个触控通道上均需配置该电路。低通滤波器的截止频率 f_0 的计算公式为：

$$f_0 = \frac{1}{2\pi R_S C_P}$$

其中，R_S 为串联电阻，C_P 为并联电容。理论上 RC 滤波器能滤掉截止频率以上的所有电磁波干扰。

表 3-1 ~ 表 3-3 分别为三种典型 RF 干扰场景下的 RC 低通滤波的对比测试。RC 低通滤波器对于不同场景下的 RF 干扰，均有比较大的抑制作用。

- GSM 干扰

表 3-1 GSM 干扰测量

GSM 干扰（33 dBm，700 ~ 1000MHz）	最大值	最小值	抖动量
无 RC 滤波	4095	1820	2275
RC 滤波（1 KΩ + 10 pF）	1991	1807	184
RC 滤波（4.7 KΩ + 20 pF）	1960	1837	123

GSM 干扰的改善量为：2275-->184-->123。

- 蓝牙干扰

表 3-2 蓝牙干扰测量

蓝牙干扰（20 dBm，2.4 ~ 2.48 G）	最大值	最小值	抖动量
无 RC 滤波	1980	1820	160
RC 滤波（1 KΩ + 10 pF）	1862	1848	14
RC 滤波（4.7 KΩ + 20 pF）	1949	1935	14

蓝牙干扰的改善量：160-->14-->14。

- 手机 NFC 干扰

表 3-3 手机 NFC 干扰测量

手机 NFC 干扰（13.65 MHz）	最大值	最小值	抖动量
无 RC 滤波	1602	1568	34
RC 滤波（1 KΩ + 10 pF）	-	-	-
RC 滤波（4.7 KΩ + 20 pF）	1827	1802	25

手机 NFC 干扰的改善量：34-->25。