**请注意赛普拉斯已正式并入英飞凌科技公司。**
此封面页之后的文件标注有“赛普拉斯”的文件即该产品为此公司最初开发的。请注意作为英飞凌产品组合的部分，英飞凌将继续为新的及现有客户提供该产品。

**文件内容的连续性**
事实是英飞凌提供如下产品作为英飞凌产品组合的部分不会带来对于此文件的任何变更。未来的变更将在恰当的时候发生，且任何变更将在历史页面记录。

**订购零件编号的连续性**
英飞凌继续支持现有零件编号的使用。下单时请继续使用数据表中的订购零件编号。

# AN78329 — CY8C20xx7/S

# CapSense®设计指南

文档编号：001-88330 版本*B

赛普拉斯半导体公司

198 Champion Court

San Jose，CA 95134-1709

www.cypress.com

## 版权所有

© 赛普拉斯半导体公司，2012-2020 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约归赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件没有附带许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方适用于个人的、非独占性、不可转让的许可（无转授许可权）（1）在版权保护下的软件（a）以源代码形式提供的软件,只能是在组织内部为了使用赛普拉斯的硬件去修改和复制。（b）以二进制代码形式从外部发到终端用户（直接或间接通过经销商和分销商），仅用于赛普拉斯硬件产品单元。（2）在软件（由赛普拉斯公司提供，且未经修改）侵犯赛普拉斯专利的权利主张下，仅许可在赛普拉斯硬件产品上制造、使用、提供和导入软件。禁止对软件的任何其他使用、复制、修改、翻译或编译。

赛普拉斯不对此材料提供任何类型的明示或暗示保证，包括但不限于针对特定用途的适销性和适用性的暗示保证。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权的访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规 格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的范围内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿并保护赛普拉斯免受所有索赔的损害，包括因人身伤害或死亡引起的索赔、费用、损失和其它责任。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。

# 目录


**目录 ..... 3**

**1. 简介 ..... 7**

1.1 摘要 ..... 7
1.2 赛普拉斯的 CapSense 文档体系 ..... 7
1.3 CY8C20xx7/S CapSense 系列特性 ..... 9
1.4 文档规范 ..... 11

**2. CapSense 技术 ..... 12**

2.1 CapSense 基本原理 ..... 12
2.2 CY8C20xx7/S 中的电容式感应方法 ..... 13
2.2.1 CapSense Sigma Delta（CSD） ..... 13
2.2.2 CapSense Sigma Delta（CSD）PLUS ..... 16
2.2.3 SmartSense_EMCPLUS 自动调校 ..... 19
2.2.4 选择用户模块 ..... 20

**3. CapSense 设计工具 ..... 21**

3.1 概述 ..... 21
3.1.1 PSoC Designer 和用户模块 ..... 21
3.1.2 CY8C20xx7/S QuietZone 入门套件 ..... 22
3.1.3 CapSense 数据查看工具 ..... 22
3.2 用户模块概述 ..... 23
3.3 CapSense 用户模块全局数组 ..... 23
3.3.1 原始计数 ..... 24
3.3.2 基准线 ..... 24
3.3.3 计数差值（信号值） ..... 24
3.3.4 传感器状态 ..... 24
3.4 CSD/CSDPLUS 用户模块参数 ..... 24
3.4.1 用户模块高层参数 ..... 25
3.4.2 CSD/CSDPLUS 用户模块低层参数 ..... 28
3.5 SmartSense_EMCPLUS 用户模块参数 ..... 32

**4. 使用用户模块对 CapSense 进行性能调校 ..... 34**

4.1 基本注意事项 ..... 34
4.1.1 信号、噪声和信噪比 ..... 34

4.1.2 充电/放电率 ..... 35
4.1.3 基准线更新阈值验证的重要性 ..... 35
4.2 调校 CSD/CSDPLUS 用户模块 ..... 36
4.2.1 CSD/CSDPLUS 的 $C_{MOD}$ 推荐值 ..... 37
4.2.2 $I_{DAC}$ 范围 ..... 37
4.2.3 自动校准 ..... 37
4.2.4 $I_{DAC}$ 值 ..... 37
4.2.5 补偿 $I_{DAC}$ 值 ..... 37
4.2.6 预充时钟源 ..... 37
4.2.7 预分频器 ..... 37
4.2.8 分辨率 ..... 38
4.2.9 扫描速度 ..... 39
4.2.10 高层 API 参数 ..... 39
4.2.11 设置高层参数 ..... 40
4.3 使用 SmartSense_EMCPLUS 用户模块 ..... 40
4.3.1 SmartSense_EMC_PLUS 的指南 ..... 40
4.3.2 用户模块差异 ..... 41
4.3.3 SmartSense_EMC_PLUS 的建议 $C_{MOD}$ 值 ..... 41
4.3.4 SmartSense_EMCPLUS 用户模块参数 ..... 41
4.3.5 SmartSense_EMCPLUS 用户模块的特定指南 ..... 42
4.3.6 CapSense 传感器的扫描时间 ..... 42
4.3.7 SmartSense_EMCPLUS 响应时间 ..... 43
4.3.8 使用 SmartSense_EMCPLUS 用户模块实现最低信噪比的方法 ..... 44
4.3.9 固件设计指南 ..... 45
4.4 将设计从 CY8C20xx6A/AS 移植到 CY8C20xx7/S ..... 47
4.4.1 不再支持/用户模块 ..... 47
4.4.2 改进和新特性 ..... 47
4.4.3 引脚兼容性 ..... 47

**5. 设计的注意事项 ..... 48**

5.1 覆盖层选择 ..... 48
5.2 ESD 保护 ..... 49
5.2.1 预防 ..... 49
5.2.2 重定向 ..... 49
5.2.3 钳制 ..... 49
5.3 电磁兼容性（EMC）的注意事项 ..... 49
5.3.1 辐射干扰 ..... 49
5.3.2 辐射 ..... 50
5.3.3 抗传导干扰和辐射 ..... 50
5.4 软件滤波 ..... 50
5.5 功耗 ..... 51
5.5.1 系统设计建议 ..... 51
5.5.2 睡眠-扫描方法 ..... 51
5.5.3 响应时间与功耗 ..... 51

5.5.4 测量平均功耗 ..... 52
5.6 引脚分配 ..... 52
5.7 GPIO 负载瞬态 ..... 53
5.7.1 降低 GPIO 负载瞬态噪声的硬件指南 ..... 54
5.7.2 补偿 GPIO 负载瞬态噪声的固件指南 ..... 55
5.8 PCB 布局指南 ..... 57

**6. 防水设计的注意事项 ..... 58**

6.1 屏蔽电极和保护传感器 ..... 58
6.1.1 屏蔽电极 ..... 58
6.1.2 保护传感器 ..... 61
6.2 设计建议 ..... 63

**7. 接近感应设计的注意事项 ..... 64**

7.1 接近感应传感器类型 ..... 64
7.1.1 按键 ..... 64
7.1.2 导线 ..... 64
7.1.3 PCB 走线 ..... 64
7.1.4 传感器机械连接 ..... 64
7.2 设计建议 ..... 65

**8. 低功耗设计的注意事项 ..... 66**

8.1 其他节能技术 ..... 66
8.1.1 将驱动模式设置为模拟高阻 ..... 66
8.1.2 全部放在一起 ..... 67
8.1.3 睡眠模式下建议的 I²C 从设备操作 ..... 67
8.1.4 睡眠模式复杂性 ..... 67
8.1.5 挂起中断 ..... 67
8.1.6 全局中断使能 ..... 68
8.2 唤醒后执行序列 ..... 68
8.2.1 使能 PLL 模式 ..... 68
8.2.2 执行全局中断使能 ..... 68
8.2.3 睡眠模式下建议的 I²C 从设备操作 ..... 68
8.2.4 睡眠定时器 ..... 69

**9. 资源 ..... 70**

9.1 网站 ..... 70
9.2 数据手册 ..... 70
9.3 技术参考手册 ..... 70
9.4 开发套件 ..... 71
9.4.1 CY8C20xx7/S QuietZone 入门套件 ..... 71
9.4.2 通用 CapSense 控制器套件 ..... 71
9.4.3 通用 CapSense 模块板 ..... 71

9.5 样本电路板文件 ..... 72
9.6 PSoC Programmer ..... 74
9.7 CapSense 数据查看工具 ..... 74
9.8 PSoC Designer ..... 74
9.9 代码示例 ..... 74
9.10 设计支持 ..... 74

**术语表 ..... 75**

**修订记录 ..... 80**

文档修订记录 ..... 80

# 1. 简介

## 1.1 摘要

本文档提供了使用 CapSense 控制器 CY8C20xx7/S 系列实现电容式感应（CapSense®）功能的设计指南。本指南包括以下主题：

- CapSense 控制器中 CY8C20xx7/S 系列的特性
- CapSense 的操作原理
- CapSense 设计工具简介
- 调校 CapSense 系统以获得最佳性能的指南
- CapSense 的系统电气和机械设计注意事项
- CapSense 的低功耗设计注意事项
- 在系统中设计 CapSense 所使用的附加资源和支持

## 1.2 赛普拉斯的 CapSense 文档体系

图 1-1 和表 1-1 汇总了赛普拉斯 CapSense 文档体系。这些资源有助于用户快速访问成功设计 CapSense 产品所需的信息。图 1-1 显示了利用电容式感应进行产品设计的典型流程；本指南中的信息与绿色显示的主题最相关。表 1-1 提供了与图 1-1 每个编号任务的支持文档的链接。

图 1-1. CapSense 产品的典型设计流程

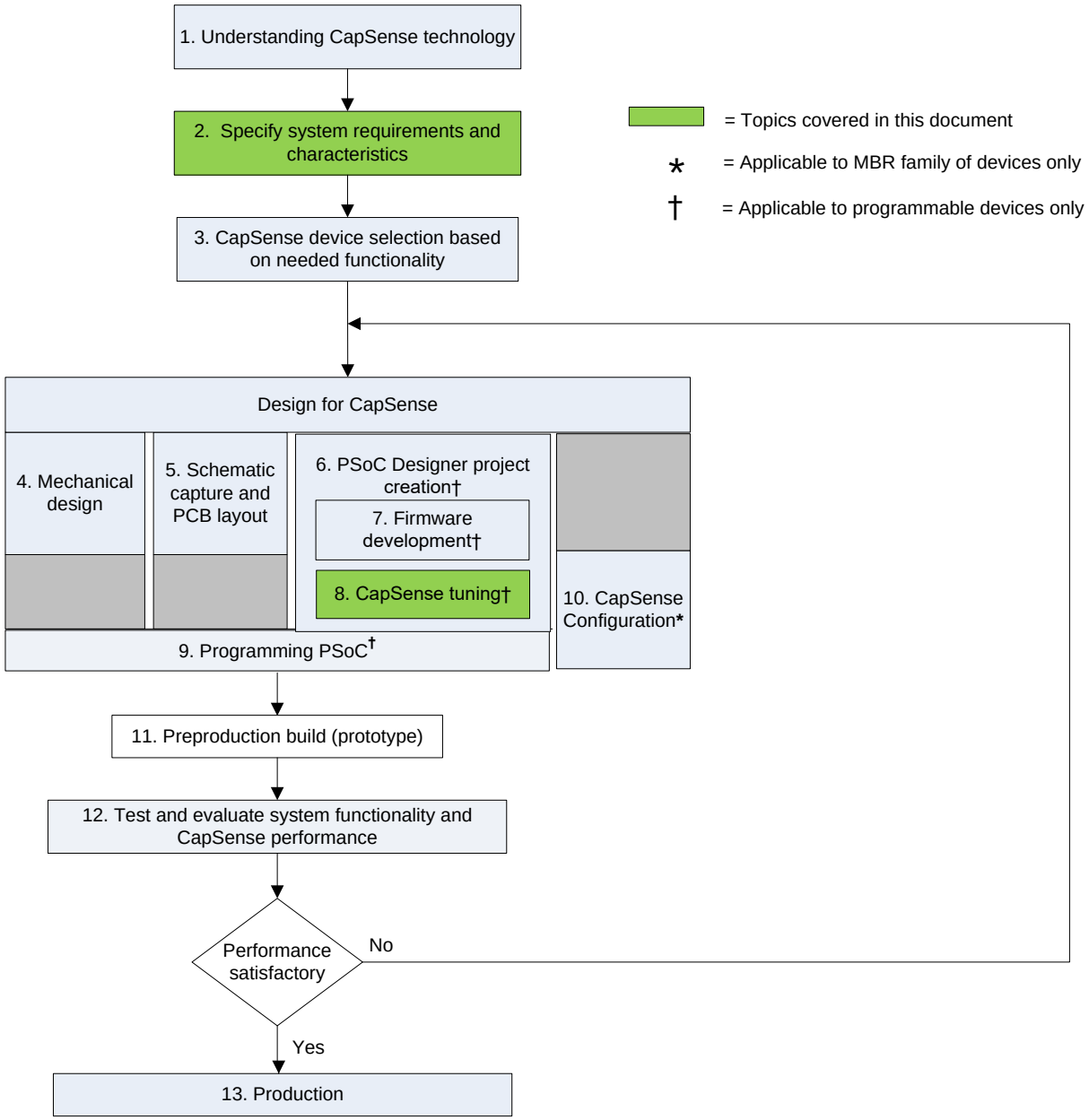


表 1-1. 图 1-1 中每个编号任务的赛普拉斯支持文档

| 图 1-1 中的编号设计任务 | 赛普拉斯的 CapSense 文档 |
|---|---|
| 1 | ● CapSense 入门 |
| 2 | ● CapSense 入门<br>● CY8C20xx7/S CapSense 器件数据手册 |
| 3 | ● CapSense 入门<br>● CY8C20xx7/S CapSense 设计指南（本文档） |
| 4 | ● CapSense 入门 |
| 5 | ● CapSense 入门 |
| 6 | ● PSoC® Designer™用户指南 |
| 7 | ● 汇编语言用户指南<br>● C 语言编译器用户指南<br>● CapSense 代码示例<br>● PSoC CY8C20xx7/S 技术参考手册 |
| 8 | ● PSoC 系列产品的 CapSense 设计指南（即本文档）<br>● PSoC 系列特定的 CapSense 用户模块数据手册（CSD、CSDPLUS 和 SmartSense_EMC_PLUS）<br>●PSoC CY8C20xx7/S 技术参考手册<br>●AN2397 — CapSense 数据查看工具 |
| 9 | ●编程器用户指南<br>●MiniProg3 用户指南<br>● ISSP 编程规范 — CY8C20045、CY8C20055、CY8C20065、CY8C20xx6A、CY8C20xx7<br>● AN59389 — CY8C20xx6A、CY8C20xx6AS、CY8C20xx6L 和 CY8C20xx7/S 的主机源串行编程 |
| 11 | ● CY8C20xx7/S CapSense 设计指南（本文档）<br>● CapSense 代码示例 |

## 1.3 CY8C20xx7/S CapSense 系列特性

赛普拉斯的 CY8C20xx7/S 是低功耗、高性能、可编程的 CapSense 控制器系列，其特性如下：

**高级触摸感应特性**

- 可编程电容式感应元件
  - ☐ 支持 CapSense 按键、滑条和接近感应传感器的组合
  - ☐ 实现按键和滑条的集成 API
  - ☐ 支持多达 31[a]个电容式传感器或 6 个滑条[b]
  - ☐ 支持传感器的寄生电容范围 5 pF 到 45 pF
- SmartSense™自动调校功能可加快上市速度
  - ☐ 在上电或运行期间自动设置和监控调校参数
  - ☐ 设计的可移植性 — 对用户界面设计中的更改自行调校
  - ☐ 运行期间的环境补偿
  - ☐ 检测低至 0.1 pF 的触摸电容变化

[a] 假设使用两个引脚进行 $I^2C$ 通信，一个引脚用于 $C_{MOD}$ 连接。

[b] 更多有关信息，请参考 CY8C20xx7/S 数据手册。

- 增强的抗噪能力和稳定性
  - SmartSense_EMCPLUS 自动补偿环境和噪声变化
  - SmartSense_EMCPLUS 为传导和辐射噪声条件较严重的应用提供出色的抗噪性能
  - 内部电压调节器提供稳定的抗电源噪声，并可以接受最大 500 mV 的供电 $V_{DD}$ 纹波
  - 提高信噪比的软件滤波器集成 API
- 超低功耗
  - 用于优化功耗的三种功耗模式
  - 活动、睡眠和深度睡眠模式（深度睡眠电流：100 nA）
  - 从睡眠模式唤醒的周期为 125 ms 时，每个传感器消耗的电流为 28 µA
- 可以在五个 GPIO 引脚上使用的驱动屏蔽
  - 提供最佳的防水设计
  - 在存在金属物体的情况下增大接近感应功能
  - 支持更大的走线长度
  - 最大负载为 100 pF（3 MHz）

**器件特性**

- 高性能又低功耗的 M8C Harvard 架构处理器
  - 内部时钟频率为 24 MHz 时，运行速度高达 4 MIPS
- 灵活的片上存储器
  - 高达 32 KB 闪存和 2 KB 的 SRAM
  - 支持仿真型 EEPROM
- 高精度的可编程时钟
  - 内部主振荡器（IMO）：6/12/24 MHz ± 5%
  - 可选择精度为 32 kHz 的外部晶体振荡器
- 增强型通用输入/输出（GPIO）功能
  - 34 个 GPIO，具有可编程引脚配置功能。
  - 单个 GPIO 支持最大 25 mA 灌电流，单个器件支持最大 120 mA 的总灌电流
  - 所有 GPIO 支持内部电阻上拉、HI-Z、开漏和强驱动模式
- 外设特性
  - 三个 16 位定时器
  - $I^2C$ — 主设备（100 kHz）和从设备（400 kHz）
  - SPI — 主设备和从设备 — 可配置范围从 46.9 kHz 到 12 MHz
  - 10 位增量型 ADC — 输入电压范围为 0 至 1.2 V
- 工作条件
  - 宽工作电压范围：1.71 V 到 5.5 V
  - 温度范围：–40 °C 到+85 °C

## 1.4 文档规范

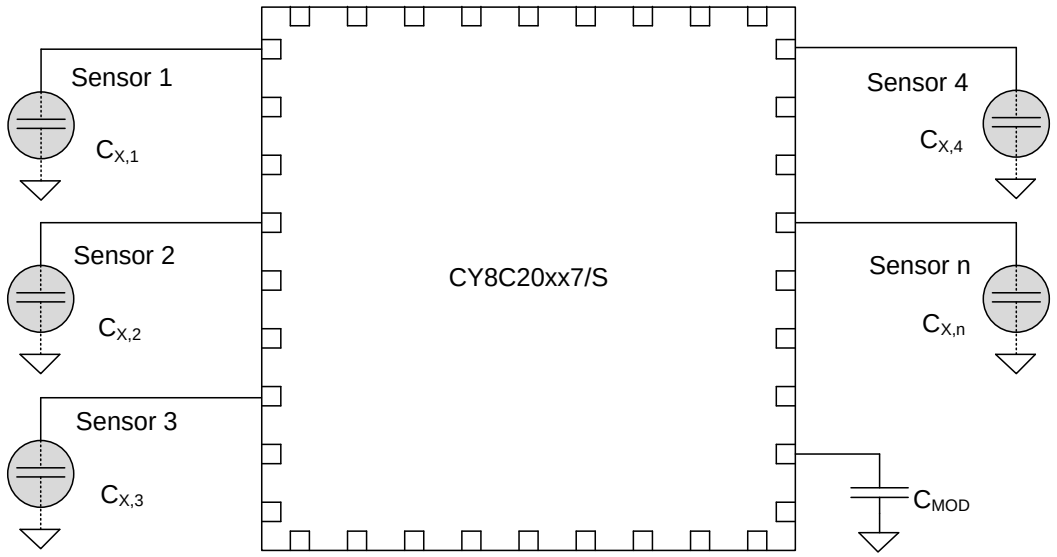
| 规范 | 使用说明 |
|---|---|
| Courier New 字体 | 显示文件位置、用户输入的文本和源代码：<br>C:\ ...cd\icc\ |
| *斜体字* | 用于显示文件名称和参考文档：<br>请阅读 *PSoC Designer 用户指南*中的 *sourcefile.hex* 文件。 |
| **[方括号、粗体]** | 用于显示程序中的键盘命令：<br>[**Enter**]或[**Ctrl**] [**C**] |
| File > Open | 表示菜单路径：<br>File > Open > New Project |
| **粗体字** | 用于显示操作过程中的各条命令、菜单路径和图标名称：<br>请点击 **File** 图标，然后点击 **Open**。 |
| Times New Roman 字体 | 用于显示公式：<br>2 + 2 = 4 |
| 灰色框中的文本 | 用于说明警告或产品的独特功能。 |

# 2. CapSense 技术

## 2.1 CapSense 基本原理

CapSense 是一种触摸式感应技术，其工作方式是测量在 CapSense 控制器上设计为传感器的每个 I/O 引脚的电容。如图 2-1 所示，对于具有 n 个传感器的设计，每个传感器引脚的总电容可以建模为值为 $C_X$,1 到 $C_X$,n 的等效集总电容。CY8C20xx7/S 器件的内部电路将每个 $C_X$ 的大小转换成数字代码，然后存储以供后期处理。CapSense 控制器的内部电路会使用其他组件（$C_{MOD}$），它的更多信息将会在 CY8C20xx7/S 电容式感应方法中讨论。

图 2-1. 在 CY8C20xx7/S PSoC 器件中实现 CapSense




如图 2-1 所示，若需要，每个传感器 I/O 引脚均通过走线、过孔或两者连接至传感器板。覆盖层是置于传感器板上的绝缘盖，也是该产品触摸界面的组成部分。当手指与覆盖层接触时，人体组织的导电性会产生一个与传感器板并行的接地导电层，如图 2-2 所示。该操作构成了平行板电容器，其电容可通过以下公式得出：

$$C_F = \frac{\varepsilon_0 \varepsilon_r A}{D}$$ 公式 2-1

其中：

$C_F$ = 手指与传感器覆盖层接触时所产生的电容值

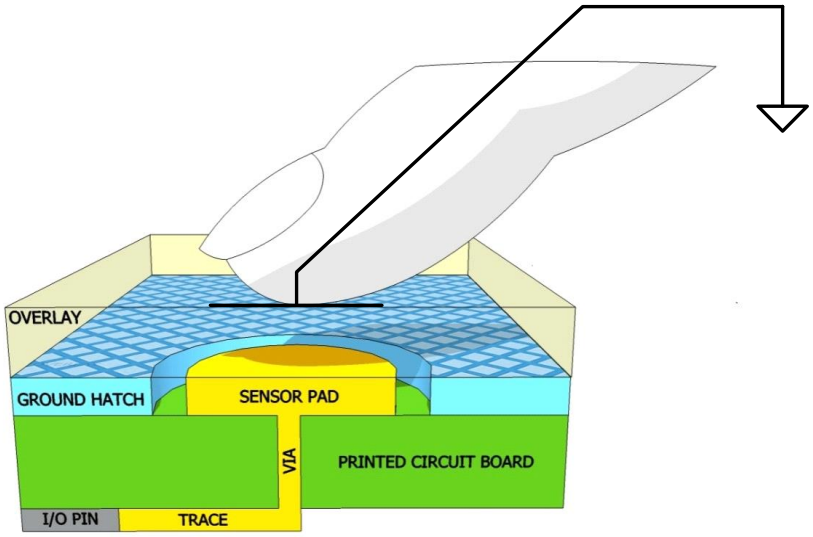
$\varepsilon_0$ = 空气介电常数

$\varepsilon_r$ = 覆盖层的绝缘常数（相对介电常数）

A = 手指与传感器板覆盖层的接触面积

D = 覆盖层的厚度

图 2-2. 典型的 CapSense PCB 与通过手指激活的传感器之间的横截面图




除了平行板电容值之外，手指与覆盖层的接触也会在其自身与附近其它导体之间产生边缘电场。与平行板电容相比，这些边缘电场的影响通常很轻微，所以通常可以忽略它们。

即使手指未触摸覆盖层，传感器 I/O 引脚也有一些寄生电容（$C_P$）。$C_P$ 是由 CapSense 控制器内部寄生电容与耦合电场共同产生的，其中电场是由传感器板、走线以及过孔与系统中其它导体（如地层、其它走线、产品机壳或外壳中的任何金属）之间的耦合产生的。CapSense 控制器可测量连接至传感器引脚的总电容 （$C_X$）。

当手指未触摸传感器时：

$$C_X = C_P$$ 公式 2-2

当手指接触传感器板上时，$C_X$ 等于 $C_P$ 与 $C_F$ 之和：

$$C_X = C_P + C_F$$ 公式 2-3

通常，$C_P$ 比 $C_F$ 大几个数量级。$C_P$ 的范围通常为 10 pF 到 25 pF，但在极端情况下可以高达 50 pF。$C_F$ 的范围通常为 0.1 pF 到 0.4 pF。调整 CapSense 系统时，$C_P$ 的数量级至关重要，该内容将在使用用户模块对 CapSense 进行性能调校中进行讨论。

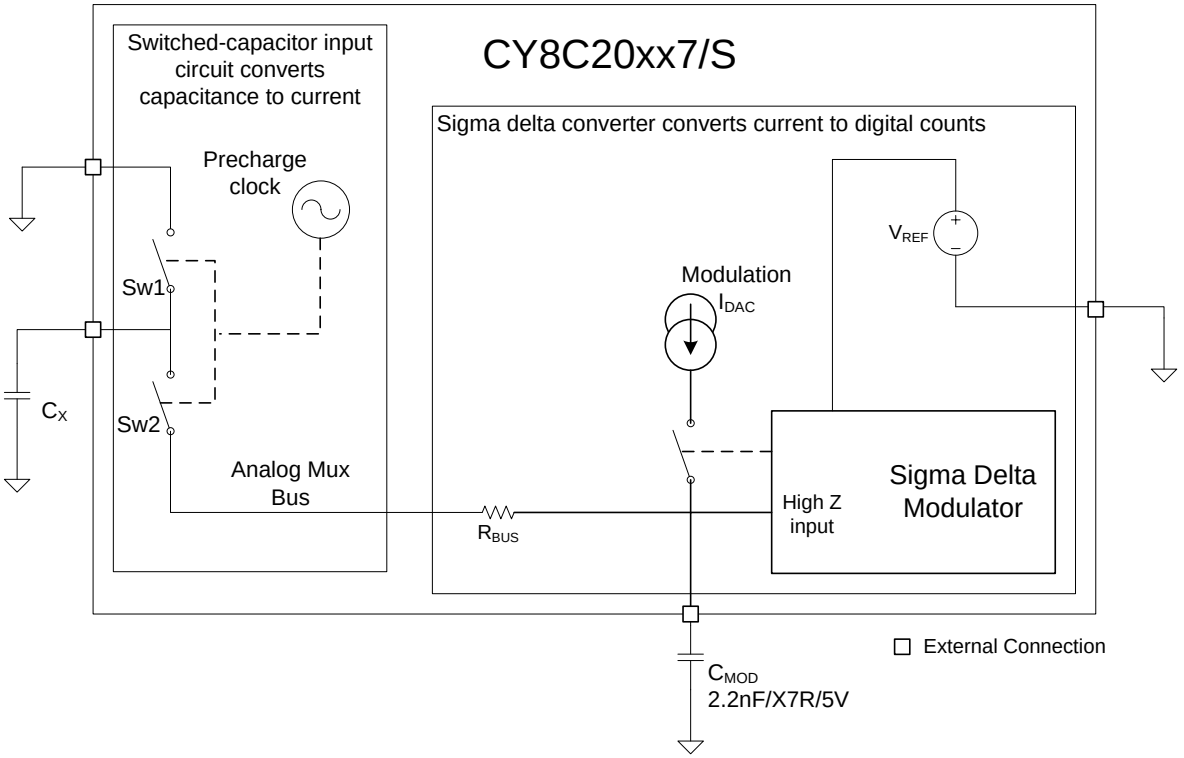
## 2.2 CY8C20xx7/S 中的电容式感应方法

CY8C20xx7/S 器件支持 CSD、CSDPLUS 和 SmartSense_EMCPLUS 等 CapSense 方法，用于将传感器电容（$C_X$）转换为数字计数。CSDPLUS 方法是 CSD 方法的超集，与 CSD 相比，该方法有了多项改进。这两种方法都是在硬件中实现的。SmartSense_EMCPLUS 使用硬件中实现的自动调校算法来自动调校所有 CSDPLUS 参数。CSD、CSDPLUS 和 SmartSense_EMCPLUS 方法在 PSoC Designer 用户模块中实现，并在以下章节中进行描述。

### 2.2.1 CapSense Sigma Delta（CSD）

图 2-3 显示的是 CSD 方法框图，该方法用于将传感器电容（$C_X$）转换为数字计数。从理论上讲，该方法可分为两个模块 — 开关电容输入前端（将电容转换为电流）和 sigma delta 转换器（将电流转换为数字计数）。以下部分介绍了这两个模块。

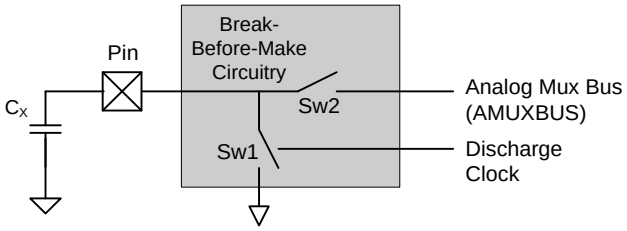
图 2-3. CSD 框图




### *2.2.1.1 带开关电容的输入*

CY8C20xx7/S 器件中的 CSD 方法将 $C_X$ 整合至开关电容电路中，如图 2-3 所示。

图 2-4. 引脚被配置为开关电容的输入




两个非重叠、非重相位时钟的频率 $F_{SW}$（请参考图 2-6）分别控制着开关 Sw1 和 Sw2。Sw1 和 Sw2 的连续切换形成一个等效电阻 $R_S$，如图 2-5 所示。相应的 $R_S$ 电阻值如下计算：

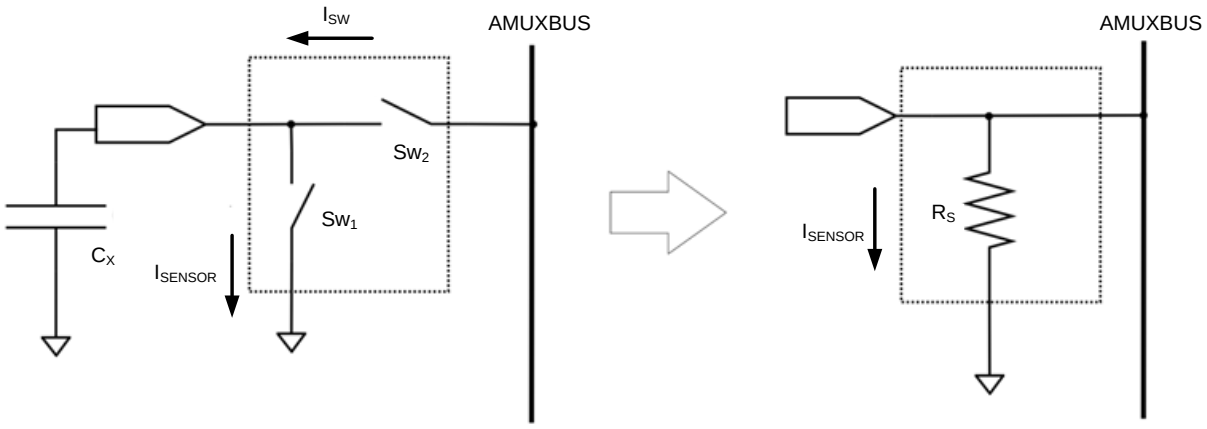
$$R_S = \frac{1}{C_X F_{SW}}$$ 公式 2-4

其中：

$C_X$ = 传感器电容

$F_{SW}$ = 开关时钟的频率

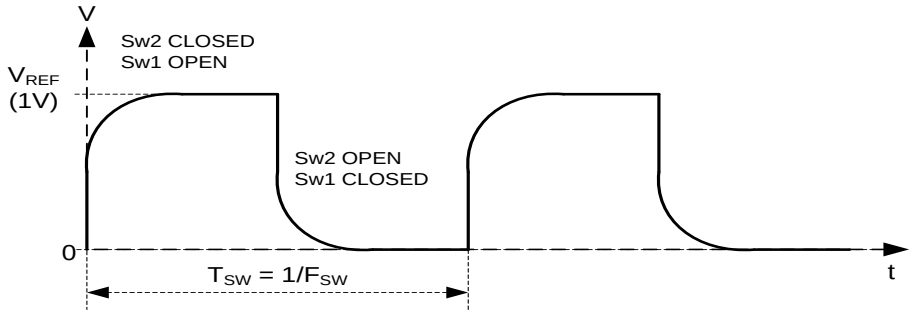
图 2-5. 开关电容电路从 AMUXBUS 吸收电流




Sigma Delta 转换器将 AMUXBUS 的电压保持为常量 $V_{REF}$（该过程如 Sigma Delta 转换器一节中所述）。图 2-6 显示的是传感器电容的电压波形。因此，使用非重叠预充电时钟来驱动开关 Sw1 和 Sw2，会产生从 AMUXBUS 输出的平均灌电流（$I_{SENSOR}$），如公式 2-5 所示。$I_{SENSOR}$ 的大小与 $C_X$ 的大小成正比。

$$I_{SENSOR} = V_{REF}/R_S = C_X F_{SW} V_{REF}$$ 公式 2-5

图 2-6. 传感器电容的电压（$C_X$）




### 2.2.1.2 Sigma Delta 转换器

Sigma Delta 转换器将输入电流转换为一个相应的数字计数。它包括一个 Sigma Delta 调制器和一个源电流 数模转换器（$I_{DAC}$），如第 14 页上的图 2-3 所示。

Sigma Delta 调制器以打开/关闭形式控制 8 位 $I_{DAC}$ 的电流。该 $I_{DAC}$ 还称为调制 $I_{DAC}$，在本文档中，它被称为“$I_{DAC}$”或“调制 $I_{DAC}$”。Sigma Delta 转换器还要求一个外部集成电容 $C_{MOD}$，如第 14 页上的图 2-3 所示。$C_{MOD}$ 的建议值为 2.2 nF。

根据 $C_{MOD}$ 上微弱的电压变化，Sigma Delta 调制器会将调制 $I_{DAC}$ 在 ON 和 OFF 状态间进行切换，从而保持 $C_{MOD}$ 的电压为 $V_{REF}$。

为了保持 AMUX 平均电压是一个稳态值（$V_{REF}$），Sigma Delta 转换器通过控制位流占空比的方式保持平均充电电流（$I_{DAC}$）与 $I_{SENSOR}$ 相互匹配。Sigma-Delta 转换器在传感器扫描期间存储位流，累积结果为数字输出值（称为原始计数），该值与 $C_X$ 成正比。

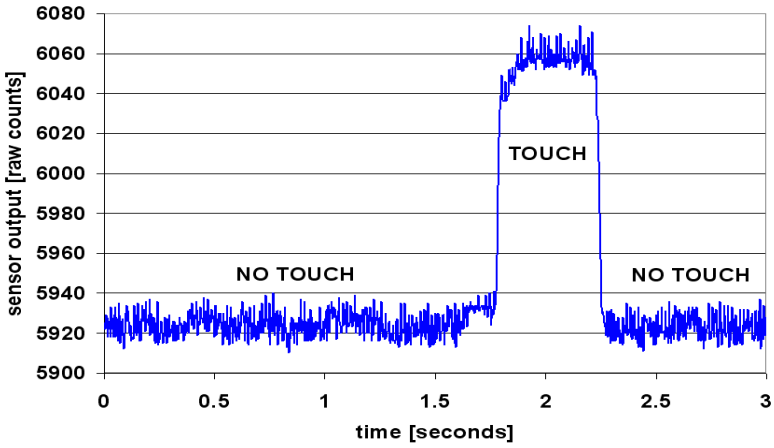
Sigma delta 转换器的工作范围为 9 位到 16 位分辨率。如果‘N’是 Sigma Delta 转换器的分辨率，并且 $I_{DAC}$ 是调制 $I_{DAC}$ 电流的值，那么计算初始计数的近似公式为：

$$\text{原始计数} = (2^N - 1)\frac{V_{REF}\ F_{SW}\ C_X}{I_{DAC}}$$ 公式 2-6

该原始计数由高级算法进行计算，以便求得传感器的状态和检测触摸。手指触摸然后释放传感器的过程中，得到若干连续扫描结果，由此绘制出 CSD 原始计数，如图 2-7。正如 CapSense 基本原理所解释，手指触摸使 $C_X$ 增加到 $C_F$，使

原始计数按比例增加。通过比较稳态下原始计数水平到预定阈值的转变，高级算法能够确定传感器是处于 ON（触摸）还是 OFF（无触摸）状态。

图 2-7. 手指触摸期间的 CSD 原始计数



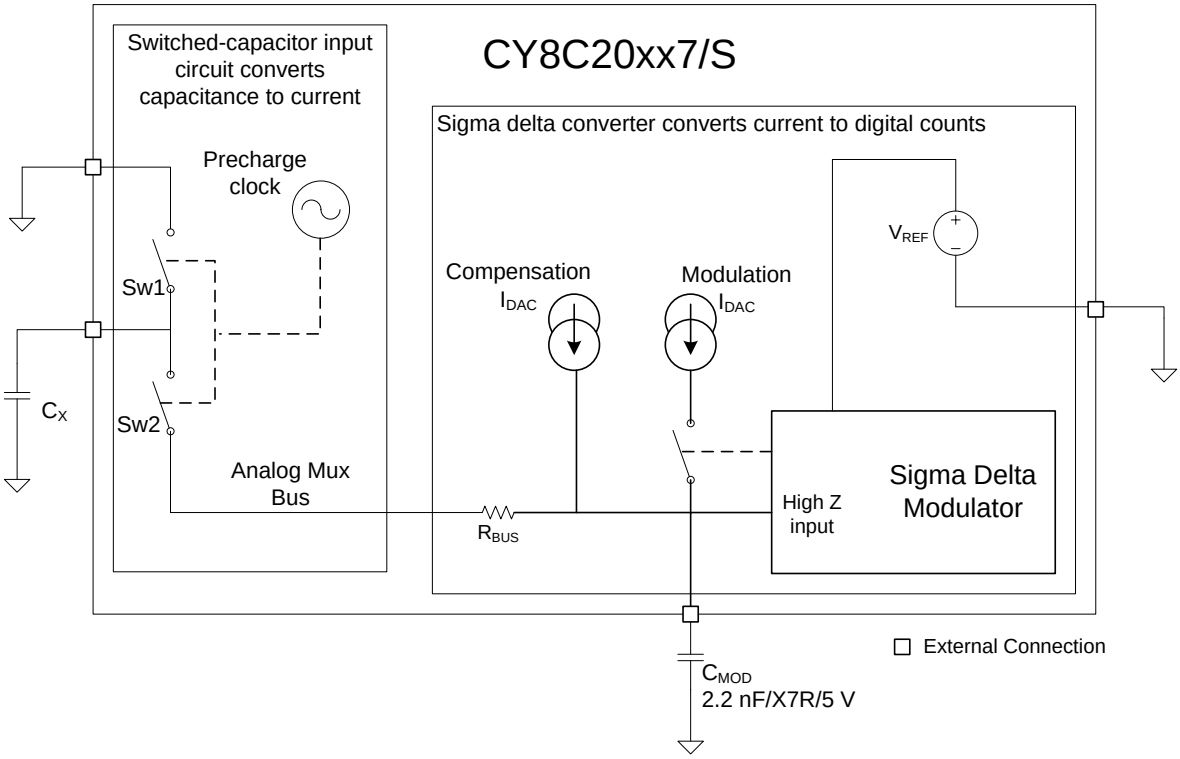

为得到可靠的触摸检测，应该以最佳值调校各个硬件参数（又称为 CSD/CSDPLUS 用户模块，如 $I_{DAC}$ 和 $F_{SW}$）和固件参数用户模块）。有关调校的更多详细信息，请参考使用用户模块对 CapSense 进行性能调校。

## 2.2.2 CapSense Sigma Delta（CSD）PLUS

图 2-8 显示了 CSDPLUS 方法的框图，该方法用于将传感器电容（$C_X$）转换为数字计数。CSDPLUS 和 CSD 方法的主要区别是在所使用的 $I_{DAC}$ 数量；CSDPLUS 使用两个 $I_{DAC}$，而 CSD 使用一个 $I_{DAC}$。

从理论上讲， CSDPLUS 方法可分为两个模块 — 带开关电容的输入（将电容转换为电流）和 sigma delta 转换器（将电流转换为数字计数）。以下部分会详细介绍每个模块。

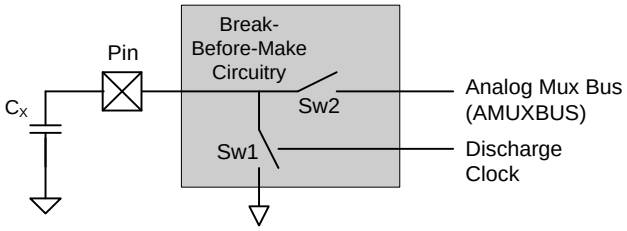
图 2-8. CSDPLUS 框图




### 2.2.2.1 带开关电容的输入

CY8C20xx7/S 器件中的 CSD PLUS 方法将 $C_X$ 整合至开关电容电路中，如图 2-9 所示。

图 2-9. 被配置为带开关电容的输入的引脚




两个非重叠、非重相位时钟的频率 $F_{SW}$（请参考图 2-11）分别控制着开关 Sw1 和 Sw2。Sw1 和 Sw2 的连续切换形成了一个等效电阻 $R_S$，如图 2-10 所示。相应的 $R_S$ 电阻值如下计算：

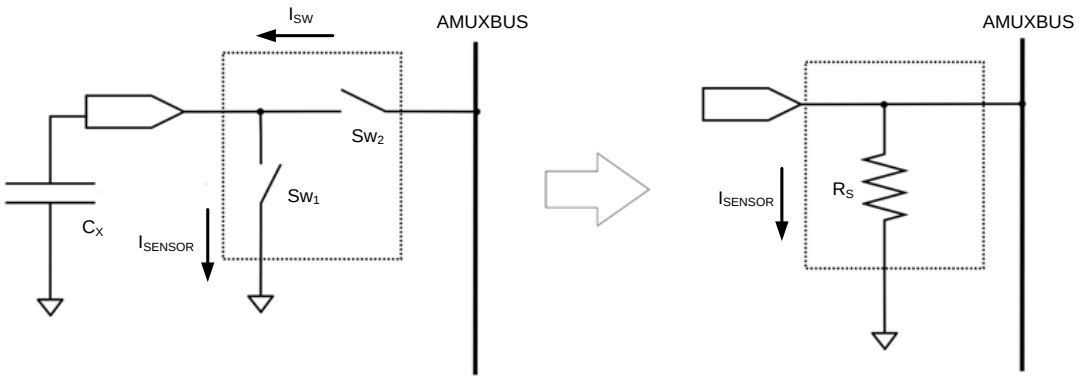
$$R_S = \frac{1}{C_X F_{SW}}$$ 公式 2-7

其中：

$C_X$ = 传感器电容

$F_{SW}$ = 开关时钟的频率

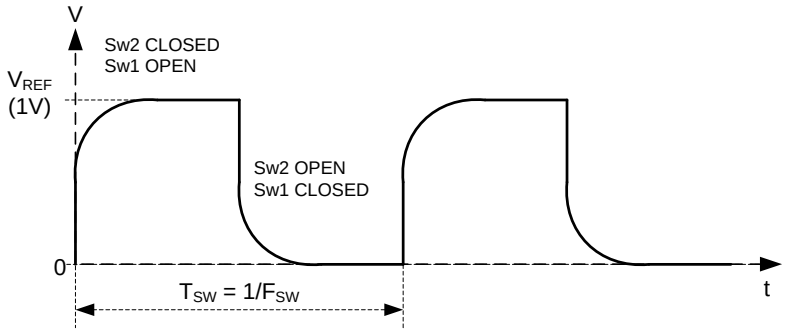
图 2-10. 从 AMUXBUS 吸收电流的带开关电容的输入




Sigma Delta 转换器将 AMUXBUS 的电压保持为常量 $V_{REF}$（该过程如 Sigma Delta 转换器一节中所述）。图 2-11 显示的是传感器电容的电压波形。因此，使用非重叠预充电时钟来驱动开关 Sw1 和 Sw2，会产生从 AMUXBUS 输出的平均灌电流（$I_{SENSOR}$），如公式 2-8 所示。$I_{SENSOR}$ 的大小与 $C_X$ 的大小成正比。

$$I_{SENSOR} = V_{REF}/R_S = C_X F_{SW} V_{REF}$$ 公式 2-8

图 2-11. 传感器电容的电压（$C_X$）




### 2.2.2.2 Sigma Delta 转换器

Sigma Delta 转换器将输入电流转换为一个相应的数字计数。它包括一个 Sigma Delta 调制器和两个源电流数模转换器（$I_{DAC}$），如第 17 页上的图 2-8 所示。

Sigma Delta 调制器以打开/关闭形式控制 7 位 $I_{DAC}$ 的电流。该 $I_{DAC}$ 还称为调制 $I_{DAC}$，在本文档中，它被称为“$I_{DAC}$”或“调制 $I_{DAC}$”。另一个 7 位 $I_{DAC}$（又称为补偿 $I_{DAC}$）的状态始终为 ON 或始终为 OFF。在本文档中，该 $I_{DAC}$ 被称为“补偿 $I_{DAC}$”或“$I_{COMP}$”。

Sigma Delta 转换器还要求一个外部集成电容 $C_{MOD}$，如第 17 页上的图 2-8 所示。$C_{MOD}$ 的建议值为 2.2 nF。根据 $C_{MOD}$ 上微弱的电压变化，Sigma Delta 调制器会将调制 $I_{DAC}$ 在 ON 和 OFF 状态间进行切换，从而保持 $C_{MOD}$ 的电压为 $V_{REF}$。

为了保持 AMUX 平均电压是一个稳态值（$V_{REF}$），Sigma Delta 转换器通过控制位流占空比的方式保持平均充电电流（$I_{DAC}$）与 $I_{SENSOR}$ 相互匹配。Sigma-Delta 转换器在传感器扫描期间存储位流，累积结果为数字输出值（称为原始计数），该值与 $C_X$ 成正比。

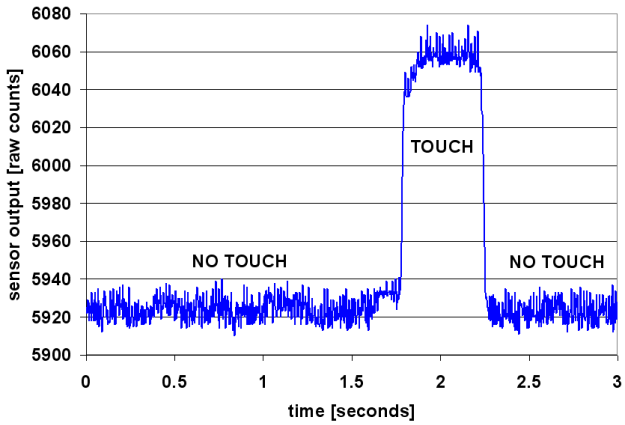
Sigma delta 转换器的工作范围为 9 位到 16 位分辨率。如果‘N’是 Sigma Delta 转换器的分辨率，并且 $I_{DAC}$ 是调制 $I_{DAC}$ 电流的值，$I_{COMP}$ 是补偿 $I_{DAC}$ 电流，那么计算初始计数的近似公式为：

$$\text{原始计数} = (2^N - 1)\frac{V_{REF} F_{SW}}{I_{DAC}} C_X - (2^N - 1)\frac{I_{COMP}}{I_{DAC}}$$ 公式 2-9

请注意，原始计数值始终为正值。因此，$I_{COMP}$ 应始终小于 $2^N V_{REF} F_{SW}$。

该原始计数由高级算法进行计算，以便求得传感器的状态和检测触摸。手指触摸然后释放传感器的过程中，得到若干连续扫描结果，由此绘制出 CSD 原始计数，如图 2-12。正如 CapSense 基本原理所解释，手指触摸使 $C_X$ 增加到 $C_F$，使原始计数按比例增加。通过比较稳态下原始计数水平到预定阈值的转变，高级算法能够确定传感器是处于 ON（触摸）还是 OFF（无触摸）状态。

图 2-12. 手指触摸期间的 CSD 原始计数




为得到可靠的触摸检测，应该以最佳值调校各个硬件参数（又称为 CSD/CSDPLUS 用户模块，如 $I_{DAC}$、$I_{COMP}$ 和 $F_{SW}$）和固件参数（又称为用户模块）。有关调校的更多详细信息，请参考使用用户模块对 CapSense 进行性能调校。

## 2.2.3 SmartSense_EMCPLUS 自动调校

调校触摸感应用户界面是确保系统正常运行和保持良好用户体验的重要步骤。典型设计流程包括，在初始设计阶段和系统整合过程中调校传感器界面，并在投入量产前进行最后的生产微调。调校是一个重复过程，可能非常耗时。SmartSense_EMCPLUS 自动调校性能用于简化用户界面开发周期。该过程非常简单易用，通过去除从原型到批量生产的整个产品开发周期中的调校过程，来大幅度缩短设计周期时间。SmartSense_EMCPLUS 在每个 CapSense 传感器加电时对其进行自动调校，从而监控并维护运行时的最佳传感器性能。这项技术适用于 PCB、覆盖层的制造性误差和噪声源（如 LCD 反相器、交流电路噪声和开关模式电源），并能够自动对其进行修正。

### *2.2.3.1 过程差异*

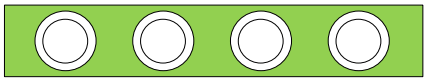
CY8C20xx7/S 的 SmartSense_EMCPLUS 用户模块（UM）能够与寄生电容大小范围为 5 pF 到 45 pF 的传感器配合使用（典型的传感器 $C_P$ 值介于 10 pF 到 20 pF 之间）。各个传感器的灵敏度参数会根据特定传感器的特性自动设置。因为在指定范围内（5 pF ~ 45 pF），无论传感器之间的 $C_P$ 怎样变动，都能保持每个传感器的响应一致，所以提高了批量生产的效率。单个传感器的寄生电容可能由于 PCB 布局、PCB 制造流程各不相同或多源供应链中的不同 PCB 供应商，有所变化。传感器的灵敏度取决于寄生电容的大小；$C_P$ 值越高，传感器灵敏度就越低，进而导致手指触摸信号振幅降低。在某些情况下，$C_P$ 值的变化会解调系统，使传感器无法达到最佳性能（过于灵敏或不够灵敏），或在最坏的情况下，传感器不能正常工作。在上述任一情况下，您都必须重新调校系统；在某些情况下，还需要重新验证 UI 子系统。SmartSense_EMCPLUS 自动调校可以解决这些问题。

通过 SmartSense_EMCPLUS 自动调校，可以实现平台设计。设想在笔记本电脑中的电容式多媒体触摸感应键；在两个按键之间的间距大小均取决于笔记本电脑尺寸和键盘布局。在本例中，宽屏机型要比标屏机型的按键间距大许多。两个按键之间的间距越大，传感器与 CapSense 控制器之间的走线就越长，进而导致传感器寄生电容越高。这意味着在相同平台上设计的不同机型（图 2-13 和图 2-14），CapSense 按键的寄生电容可能不同。虽然对于所有笔记本机型而言，这些按键的功能均是相同的，但是，传感器必须针对每种机型进行调校。使用推荐的最佳实践（参见 CapSense 入门中的 PCB 布局），SmartSense_EMCPLUS 可以帮助您实现平台设计，调校功能会自动和有效的完成。

图 2-13. 21 英寸笔记本电脑的多媒体按键设计



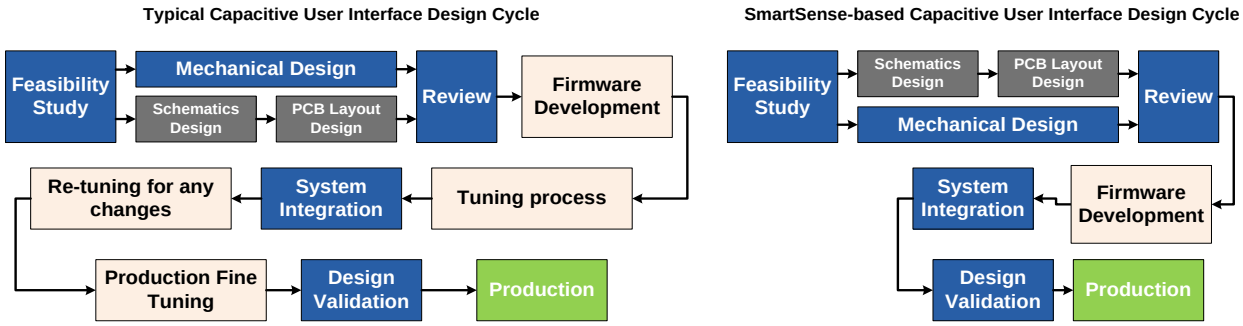
图 2-14. 15 英寸笔记本电脑的多媒体按键设计（该机型与 21 英寸机型有相同的功能和按键大小）



### 2.2.3.2 *缩短设计周期时间*

通常，电容式传感器接口设计最耗时的任务是固件开发和传感器调校。一般对于典型的触摸感应式控制器，在不同机型中进行相同的设计，或者机器的 PCB 尺寸或传感器 PCB 布局发生变化时，都需要重新调校传感器。SmartSense_EMCPLUS 设计解决了这些挑战，因为它需要较少的固件开发工程，无须调校和重新调校过程。这样明显加快了常规设计周期的进程。图 2-15 对典型的触摸感应控制器和基于 SmartSensee_EMC_PLUS 设计流程的周期进行比较。

图 2-15. 典型电容式界面设计周期的比较




## 2.2.4 选择用户模块

与 CSD 和 CSDPLUS UM 不同，SmartSense_EMCPLUS 设计不需要进行手动调校。为了简化 CapSense 设计过程，建议您采用 SmartSense_EMCPLUS。但在某些情况下，需要控制传感器参数（如分辨率和预分频器）来优化器件功耗或通过检测高 $C_P$ 检测出传感器触摸。在这种情况下，请使用 CSD/CSDPLUS UM。

CSDPLUS UM 相比 CSD UM 有以下优点：

- 对于给定的传感器分辨率（或扫描时间），CSDPLUS UM 提供了更高的 SNR。
- 调校信噪比相同的情况下，CSDPLUS UM 扫描传感器需要的时间更短，所以同等条件下使用 CSDPLUS UM 相比使用 CSD UM CapSense 器件的平均功耗[a]更低。

但是，传感器 $C_p$ 小于 10pF 时，CSDPLUS UM 提供的信噪比低于 CSD UM，此时，建议使用 CSD UM，而不是 CSDPLUS UM。

**注意：**如果将 $I_{COMP}$ 设为 0，CSDPLUS UM 与 CSD UM 没有区别。

[a] 这里，假设器件进行扫描传感器后进入睡眠模式并被周期性唤醒，以降低器件功耗。

# 3. CapSense 设计工具

## 3.1 概述

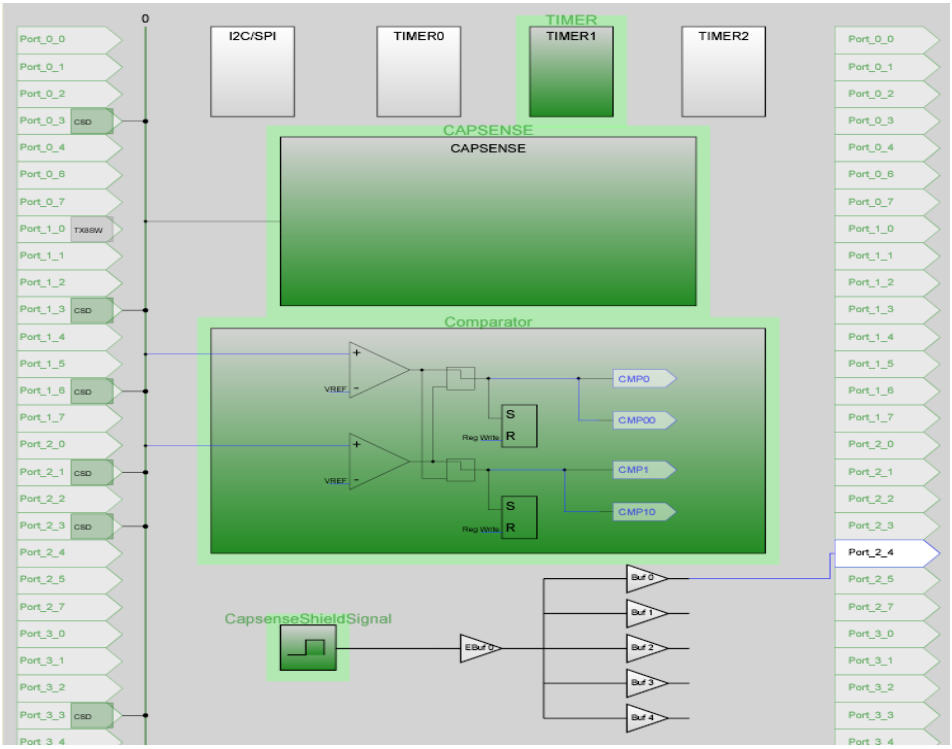
赛普拉斯可以提供用来开发 CapSense 电容式触摸传感应用的全套硬件和软件工具。有关订购信息，请参见资源。

### 3.1.1 PSoC Designer 和用户模块

赛普拉斯独有的集成设计环境（PSoC Designer）支持配置模拟和数据模块、开发固件和调校和调试设计。在拖放式设计环境中使用用户模块库开发这些应用。要配置用户模块，可通过器件编辑器 GUI 实现或通过固件写入特定的寄存器内来完成。PSoC Designer 具有内置的 C 语言编译器和嵌入式编程器。专业版编译器可用于复杂设计。

CSD 和 CSDPLUS 用户模块使用开关电容电路、模拟复用器、比较器、数字计数功能和高级软件子程序（API）来实现电容式触摸传感器。其他模拟和数字外设的用户模块可用于实现其他功能，如 I²C、SPI、TX8 和定时器。

图 3-1. PSoC Designer 器件编辑器



#### *3.1.1.1 CapSense 用户模块入门*

在 PSoC Designer 中创建一个新的 CY8C20xx7/S 项目：

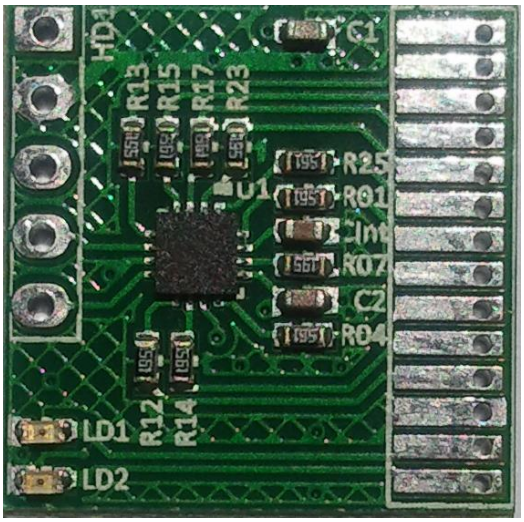
1. 使用目标器件 CY8C20xx7/S 新建 PSoC Designer 项目。
2. 选择并放置 CSDPLUS/SmartSense_EMCPLUS 用户模块。
3. 右键单击用户模块，访问用户模块向导。
4. 设置按键传感器计数、滑条配置、引脚分配及相关项。
5. 设置引脚和全局用户模块参数。
6. 生成应用，并切换到应用编辑器。
7. 根据用户模块数据手册调整示例代码，使按键或滑条生效。

欲了解创建 PSoC Designer 项目及配置用户模块向导的详细过程，请参考特定用户模块的数据手册。欲了解 CapSense 用户模块的代码示例，请参见代码示例。

### 3.1.2 CY8C20xx7/S QuietZone 入门套件

CY8C20xx7/S QuietZone 入门套件是简单的即插即用式硬件，易于进行原型设计。您可以从我们的生产模块合作伙伴 ArtaFlex 的以下链接下载该套件：www.artaflexmodules.com/quietzone

图 3-2. CY8C20xx7/S QuietZone 入门套件

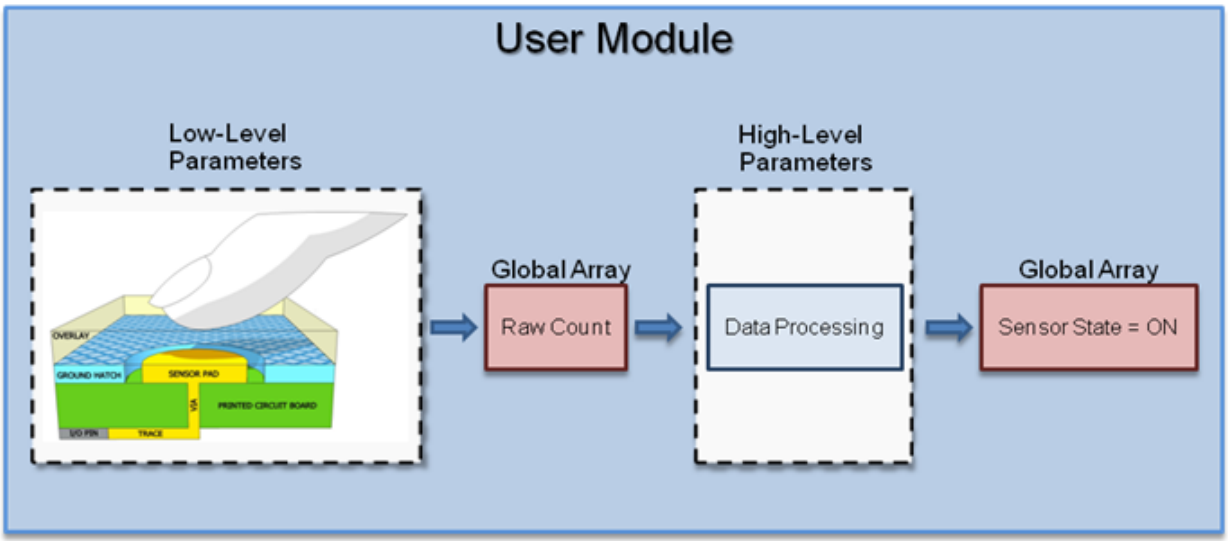



### 3.1.3 CapSense 数据查看工具

CapSense 设计过程中，您会经常需要监测 CapSense 数据（原始计数、基准线、计数差值（信号值）等），以进行调校和调试。CapSense 数据查看工具有两种：MultiChart 和 Bridge Control Panel。有关这些工具的详细说明，请参见应用笔记 AN2397 — CapSense 数据查看工具。

## 3.2 用户模块概述

图 3-3. 用户模块框图




用户模块包含整个 CapSense 系统，即从物理感应到数据处理。用户模块的行为是通过使用各种参数定义的。这些参数影响感应系统的不同器件，可分为低层参数和高层参数，在这些参数之间使用全局数组进行通信。

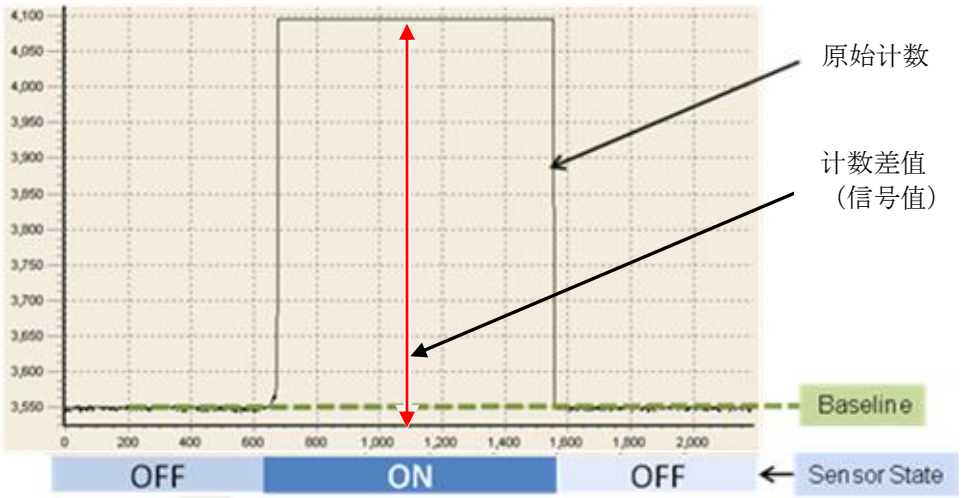
低层参数在物理层定义传感方法的行为，并涉及从电容到原始计数的转换，如扫描传感器的速度和分辨率。每种感应方法具有独特的低层参数，相关描述请参见 CSD/CSDPLUS 用户模块和 SmartSense_EMCPLUS 用户模块参数。

高层参数（如防反跳计数和噪声阈值）定义如何处理原始计数来产生相关信息，如：传感器 ON/OFF（触摸/离开）状态和手指在滑条上的大致位置。这些参数在所有的感应方法中都相同。相关描述请参见用户模块。

## 3.3 CapSense 用户模块全局数组

学习 CapSense 用户模块参数之前，您必须熟悉 CapSense 系统所使用的某些全局数组。这些数组不应通过手动方式来改变，但可以针对调试这一目的进行检测。

图 3-4. 原始计数、基准线、计数差值和传感器状态



### 3.3.1 原始计数

CapSense 控制器中的硬件电路用于测量传感器的电容 $C_X$。调用用户模块 API UMname_ScanSensor()时，该电路以数字格式存储调用的原始计数的结果，其中 UMname 可以是 CSD、CSDPLUS 或 SmartSenseEMC_PLUS。

传感器的原始计数与其电容成正比。当传感器电容值增加时，原始计数也随之增加。

传感器的原始计数值存储在 *UMname*_waSnsResult[]整数数组中。在头文件 *UMname.h* 中定义该数组。

### 3.3.2 基准线

基准线被认为是与传感器的寄生电容（$C_P$）相对应的原始计数值。环境的缓慢改变（如温度和湿度）会影响传感器 $C_P$ 以及 $C_X$，这样会引起原始计数的差异。

用户模块使用复杂的基准线算法，用来补偿这些差异。该算法使用基准线变量来完成该功能。基准线变量记录了原始计数值的所有渐变差异。实际上，基准线变量会存储数字低通滤波器的输出（该数字低通滤波器使用的输入则是原始计数值）。

基准线算法由用户模块 API *UMname*_UpdateSensorBaseline 执行（其中 *UMname* 可以为 CSD、CSDPLUS 或 SmartSense_EMC_PLUS）。

传感器的基准线值被存储在 *UMname*_waSnsBaseline[]整数型数组内。在头文件 *UMname.h* 中定义该数组。

### 3.3.3 计数差值（信号值）

计数差值也称为传感器信号，被定义为传感器原始计数和基准值之间计数的差值。当传感器处于非活动状态时，计数差值为零。激活传感器（通过触摸方式）使计数差值为正值。

传感器的计数差值存储在 *UMname*_waSnsDiff[]整数型数组内，其中 *UMname* 可以是 CSD、CSDPLUS 或 SmartSense_EMC_PLUS。在头文件 *UMname.h* 中定义该数组。

计数差值变量可通过用户模块 API *UMname*_UpdateSensorBaseline()更新。

### 3.3.4 传感器状态

传感器状态表示物理传感器的活动/非活动状态。当手指触摸传感器时，传感器的状态从 0 变为 1，而手指释放时，传感器的状态返回 0。

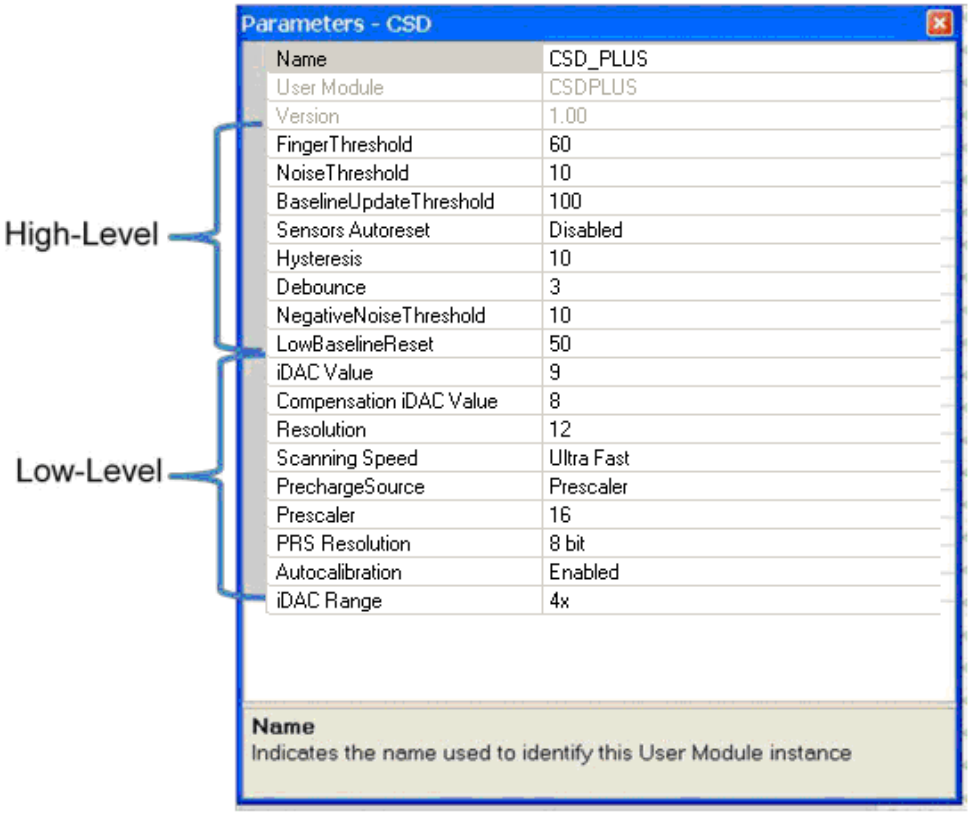
传感器的状态存储在名为 *UMname*_baSnsOnMask[]的字节数组内，其中 *UMname* 可以为 CSD、CSDPLUS 或 SmartSense_EMCPLUS。在头文件 *UMname.h* 中定义该数组。每字节存储 8 个连续传感器的状态。

传感器状态通过用户模块 API *UMname*_bIsAnySensorActive()更新。

## 3.4 CSD/CSDPLUS 用户模块参数

CSD/CSDPLUS 用户模块参数分为高层参数和低层参数。有关 CSDPLUS 用户模块参数列表和如何分类这些参数的详细内容，请参见图 3-5。CSD 和 CSDPLUS UM 参数窗口的唯一区别是：CSD UM 没有 Compensation iDAC（补偿 iDAC）值。

图 3-5. PSoC Designer — CSDPLUS 参数窗口




## 3.4.1 用户模块高层参数

### *3.4.1.1 手指阈值*

用户模块使用该手指阈值参数判断传感器是否为活动/非活动状态。如果传感器的计数差值（信号值）值超过手指阈值，则传感器被判定为活动状态。该定义假设迟滞水平为 0，去抖动值为 1。

噪声阈值的取值范围为 3 到 255。

更多有关建议值的信息，请参见设置高层参数。

### *3.4.1.2 迟滞*

该迟滞设置可防止传感器 ON 状态因系统噪声而抖动。公式 3-1 给出了迟滞函数。该公式假设去抖动值被设置为 1。

图 3-6. 迟滞值为 0 时，传感器状态与差值的关系

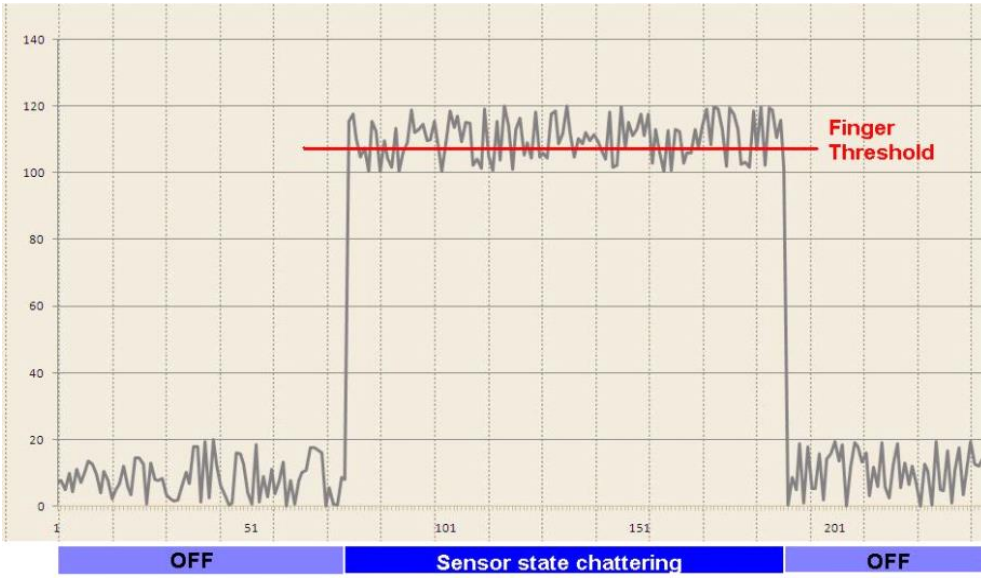
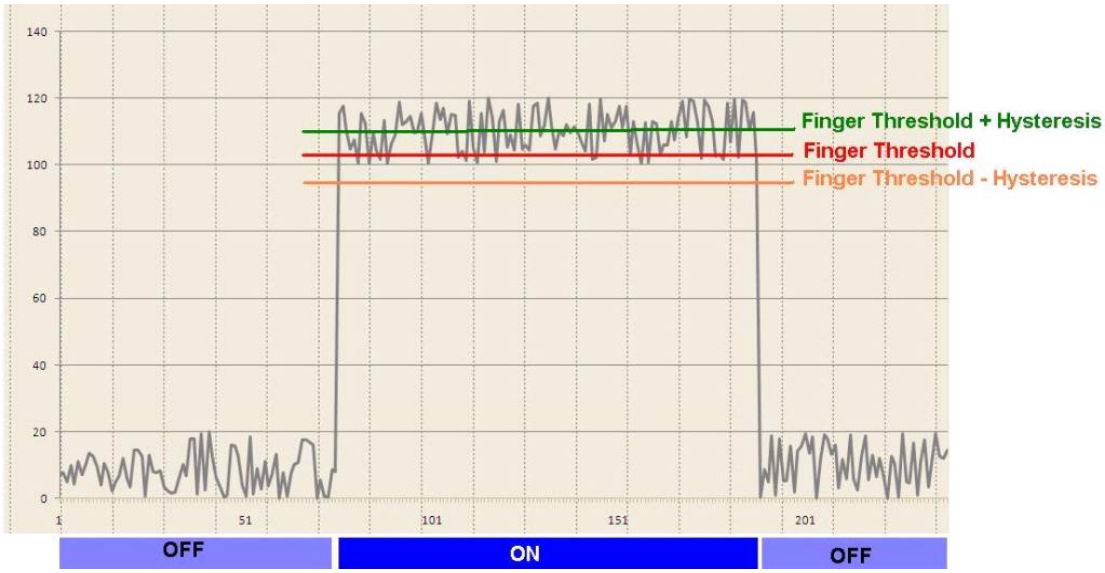



图 3-7. 迟滞值非 0 时，传感器状态与差值的关系




*如果差值计数*

*如果差值计数* ≤ *手指阈值* – *迟滞，传感器状态为OFF* 公式 3-1

该参数的取值范围为 0 到 255。

更多有关建议值的信息，请参见设置高层参数。

### *3.4.1.3 去抖动*

去抖动参数可防止原始计数中的尖峰将传感器状态从 OFF 更改为 ON。对于传感器状态从 OFF 到 ON 的跃变而言，必须保持计数差值（信号值）在指定的采样数内大于手指阈值与迟滞量之和。

取值范围为 1 到 255。如果将该参数设置为 1，将不提供去抖动。

更多有关建议值的信息，请参见设置高层参数。

#### 3.4.1.4 基准线更新阈值

如前所释，基准线变量会追踪原始计数的渐进变化。换言之，基准线变量存储数字低通滤波器的输出（该数字低通滤波器使用的输入则是原始计数值）。“基准线更新阈值”参数用于调整该低通滤波器的时间常数。

基准线更新阈值与该滤波器的时间常量成正比。基准线更新阈值越高，时间常量就越大。

取值范围为 0 到 255。

更多有关建议值的信息，请参见设置高层参数。

#### 3.4.1.5 噪声阈值

用户模块使用噪声阈值来了解原始计数中噪声计数的上限。对于单个传感器而言，当原始计数超过基准线且二者之间的差值大于该阈值时，基准线更新算法暂停。

对于滑条传感器而言，当计数差值（信号值）大于噪声阈值时，质心计算会暂停。

噪声阈值的取值范围为 3 到 255。为了保证用户模块正常操作，噪声阈值不会高于手指阈值与迟滞之差。

更多有关建议值的信息，请参见设置高层参数。

#### 3.4.1.6 负噪声阈值

负噪声阈值有助于用户模块了解原始计数中噪声计数的下限。当原始计数低于基准线，并且它们之间的差值大于负噪声阀值时，将暂停基准线更新算法。

取值范围为 0 到 255。

更多有关建议值的信息，请参见设置高层参数。

#### 3.4.1.7 低基准线复位

低基准线复位参数与负噪声阈值参数结合使用。如果采样计数值小于基准线与指定采样数量的负噪声阈值之差，则将基准线设置为新的原始计数值。该参数基本上计数了复位基准线所需的异常低的采样值。采样值用来纠正启动时手指触摸传感器的状态。

该参数的取值范围为 0 到 255。

更多有关建议值的信息，请参见设置高层参数。

#### 3.4.1.8 传感器自动复位

传感器自动复位参数指定基准线值是需要随时进行更新，还是仅在计数差值小于噪声阈值时才更新。

使能传感器自动复位后，会不断地更新基准线。这限制了传感器的最大持续时间，但可以防止在无触摸的情况下，原始计数意外升高而导致传感器处于永久性打开状态。这种突然上升可能是由电源中的大电压波动、高能射频噪声源或较快的温度变化波动所引起的。

禁用传感器自动复位时，仅在计数差值小于噪声阈值参数时才更新基准线。

可能值为“使能”和“禁用”。

更多有关建议值的信息，请参见设置高层参数。

## 3.4.2 CSD/CSDPLUS 用户模块低层参数

除高层参数外，CSD/CSDPLUS 用户模块还具有多个低层参数。这些参数是 CSD/CSDPLUS 感应方法特定的参数，用于决定如何从传感器中采集原始计数数据。

### *3.4.2.1 $I_{DAC}$ 值*

$I_{DAC}$ 参数用于设置电容测量的范围和灵敏度（原始计数与 $C_X$ 成正比）。$I_{DAC}$ 值越大，电容测量范围越宽，但灵敏度也越低，下面将详细介绍这种情况。

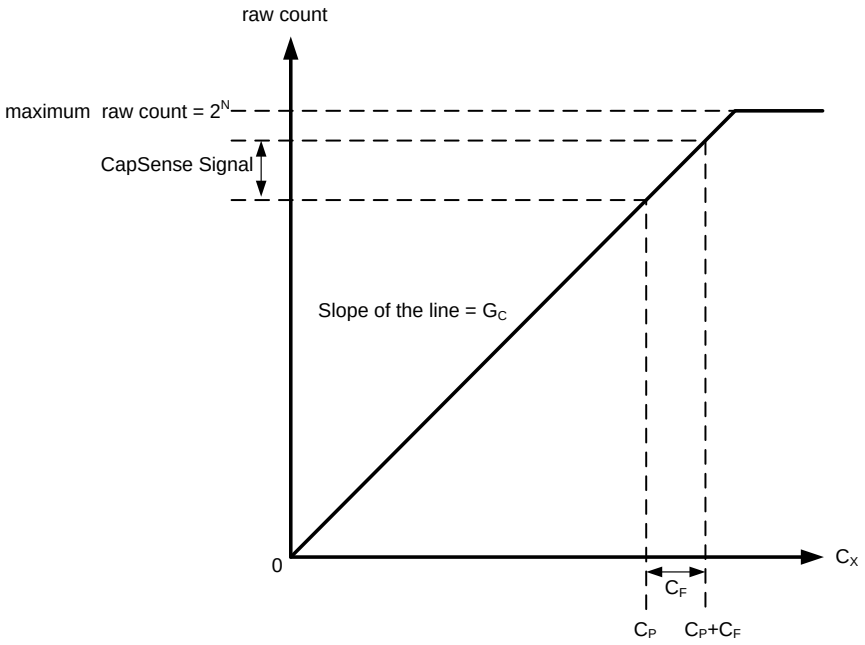
按照公式 2-9，原始计数值与 $C_X$ 和 $I_{DAC}$ 密切相关。

$$\text{原始计数} = G_C C_X - (2^N - 1)\frac{I_{COMP}}{I_{DAC}} \qquad \text{公式 3-2}$$

其中，$G_C$ 是数字转换增益的容值，并等于 $(2^N - 1)\frac{V_{REF} F_{SW}}{I_{DAC}}$。

请考虑这种情况，补偿 $I_{DAC}$ 值（即 $I_{COMP}$）为 0。在这种情况下，图 3-8 显示了原始计数和 $C_X$（公式 3-2）的关系。

图 3-8. 原始计数与传感器电容




当手指触摸传感器时，原始计数会发生变化，该过程被称为一个 CapSense 信号。图 3-9 显示了信号值随转换增益如何变化。

图 3-9. 不同转换增益中的信号值

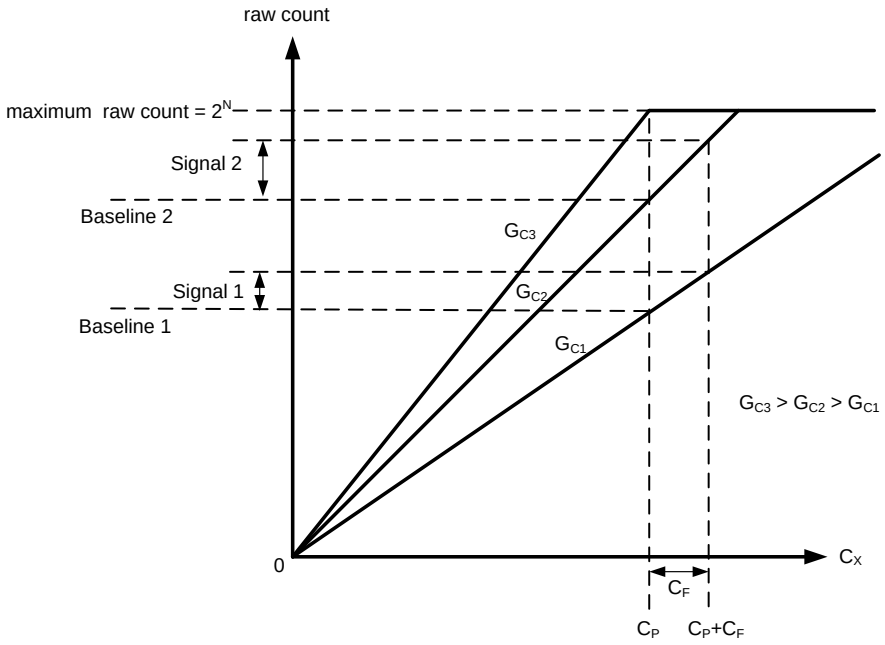
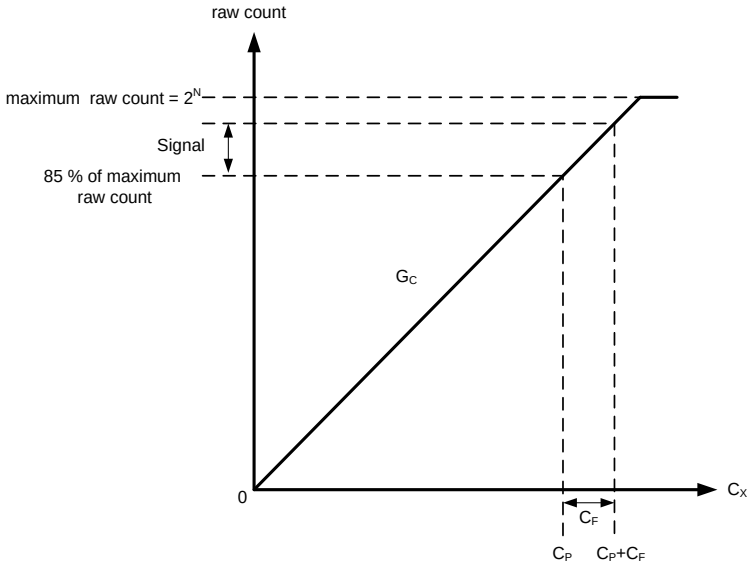



图 3-9 显示了与三个转换增益值相对应的三条关系线：$G_{C3}$、$G_{C2}$ 和 $G_{C1}$。转换增益值越大，信号值便越大。但增大转换增益值也会使与 $C_P$ 相对应的原始计数值越来越靠近原始计数最大值（$2^N$）。增益值过大，原始计数会饱和，如 $G_{C3}$ 所示。因此，为避免原始计数饱和，您应该调整转换增益值，以得到良好的信号值。建议调整增益，使原始计数对应的 $C_P$ 值等于原始计数最大值的 85%，如图 3-10 所示。

图 3-10. $I_{COMP}$ 为 0 时建议调校



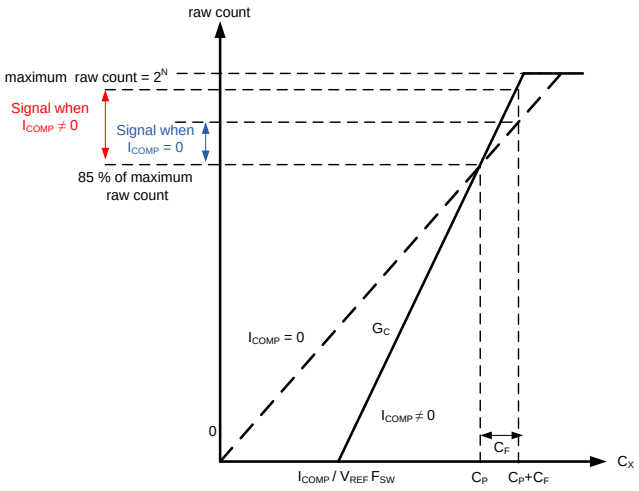

可以在运行时使用相应的 API 函数更改该参数。

取值范围为 1 到 127。

### 3.4.2.2 补偿 $I_{DAC}$ 值

补偿 $I_{DAC}$ 参数用于设置原始计数与 $C_X$ 的偏移图。还可以使用补偿 $I_{DAC}$ 来获取高 $C_P$ 传感器的高灵敏度。图 3-11 显示的是 $I_{COMP}$ 为非零时原始计数与电容式传感器的关系。

图 3-11. $I_{COMP}$ 为非零时原始计数与 $C_X$ 的关系




增大补偿 $I_{DAC}$ 值会使 $C_X$ 轴的截距变大：$\left(\frac{I_{COMP}}{V_{REF}\ F_{SW}}\right)$

当原始计数再次被增大到其最大值的 85%时，会加大直线斜率。因此，当补偿 $I_{DAC}$ 值变大时，信号值也被增大。

请注意，当使用补偿 $I_{DAC}$ 时，由于噪声计数的变化，信号的这种增加可能不会成比例地增加 SNR。

由于使用了额外变量（$I_{COMP}$），双 $I_{DAC}$ 模式使调试过程变得更复杂。但您仍能使用该模式来增大信号。详细内容如调校 CSD/CSDPLUS 用户模块所述。

还可以在运行时使用相应的 API 函数更改该参数。

取值范围为 1 到 127。

### 3.4.2.3 分辨率

该参数指定扫描分辨率（单位为“位”）。N 位的扫描分辨率的最大原始计数为 $2^N$–1。增大分辨率可提高灵敏度，但会减少扫描时间，从而降低响应速率。

取值范围为 9 到 16 位。

表 3-1. 分辨率和扫描速度

| 分辨率 | 单个按键的扫描速度（µs） | | | |
|---|---|---|---|---|
| | 超快速度 | 快速 | 正常速度 | 慢速 |
| 9 | 57 | 78 | 125 | 205 |
| 10 | 78 | 125 | 205 | 380 |
| 11 | 125 | 205 | 380 | 720 |
| 12 | 205 | 380 | 720 | 1400 |
| 13 | 380 | 720 | 1400 | 2800 |
| 14 | 720 | 1400 | 2800 | 5600 |
| 15 | 1400 | 2800 | 5600 | 11000 |
| 16 | 2800 | 5600 | 11000 | 22000 |

#### 3.4.2.4 扫描速度

该参数通过设置 Sigma Delta 转换器的时钟输入而指定传感器扫描速度。尽管更快的扫描速度可提供良好的响应时间，但较慢的扫描速度具有以下优势：

- 提高信噪比
- 更好的抗电源和温度变化的能力
- 很少需要系统中断延迟；您能够处理较长的中断

可选项为超快、快速、正常速度和慢速。

#### 3.4.2.5 屏蔽电极输出

屏蔽电极用于降低寄生电容并实现防水功能。请参考本文档中的防水设计的注意事项和 CapSense 入门中的“屏蔽电极和保护传感器”章节，以便深入了解屏蔽电极。该参数选择屏蔽电极输出的路由位置。

可能的值为 P0[0]、P1[2]、P0[2]、P2[2]和 P2[4]。

#### 3.4.2.6 预充时钟源

该参数用于选择预充电开关的时钟源，在带开关电容的输入一节中称为预充电时钟。

可选项为 PRS 和预分频器。预分频器可作为 IMO/预分频使用，用于设置开关时钟频率。PRS 的选择通过伪随机发生器传递被分频的 IMO 时钟，提供扩频时钟。在大多数情况下，使用 PRS 源能够获得更好的抗 EMI 能力和更低的辐射，因为 PRS 可对较大范围的切换频率求平均值。

#### 3.4.2.7 预分频器

该参数用于设置预分频器比例，并确定预充电开关输出频率。该参数还影响 PRS 输出频率。

可选值为 1、2、4、8、16、32、64、128 和 256。

#### 3.4.2.8 PRS 分辨率

该参数用于更改 PRS 序列的长度。

如果需要极短的扫描时间，则必须使用 8 位 PRS 来避免过大噪声。扫描时间由分辨率（不应该与 PRS 分辨率相互混淆）参数确定。对于扫描时间小于或等于 380 µs，将 PRS 分辨率设置为 8 位；对于扫描时间大于 380 µs，将 PRS 分辨率设置为 12 位。默认设置为 8 位。

#### 3.4.2.9 自动校准

如果使能自动校准参数，调制和补偿 $I_{DAC}$ 将自动得到校准，从而建立大约为 $2^N$ 的 85%的原始计数基准线，其中“N”是传感器的分辨率设置。使能自动校准参数会覆盖 $I_{DAC}$ 和 $I_{COMP}$ 的器件编辑器设置。

如果禁用自动校准参数，则原始计数值将取决于器件编辑器的 $I_{DAC}$ 范围、$I_{DAC}$ 值、分辨率、传感器电容和 IMO 频率、预分频器、预充时钟源和 $V_{REF}$ 等参数集。

自动校准参数使用 ROM 和 RAM 资源，并且会增加启动时间。自动校准参数不会自动选择 $I_{DAC}$ 范围的值。如果校准后的原始计数值小于分辨率范围的一半，则应当增大 $I_{DAC}$ 范围或降低预充电频率。自动校准参数用于提高边缘功能配置。

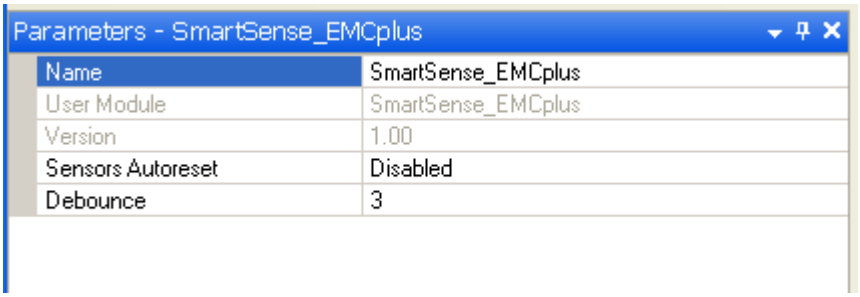
#### 3.4.2.10 $I_{DAC}$ 范围

$I_{DAC}$ 范围参数用于缩放 $I_{DAC}$ 电流输出。例如，选择 8x 会将 $I_{DAC}$ 输出缩放到 4x 范围的两倍。

可能值为 4x 和 8x。

## 3.5 SmartSense_EMCPLUS 用户模块参数

图 3-12. PSoC Designer SmartSense_EMCPLUS 参数

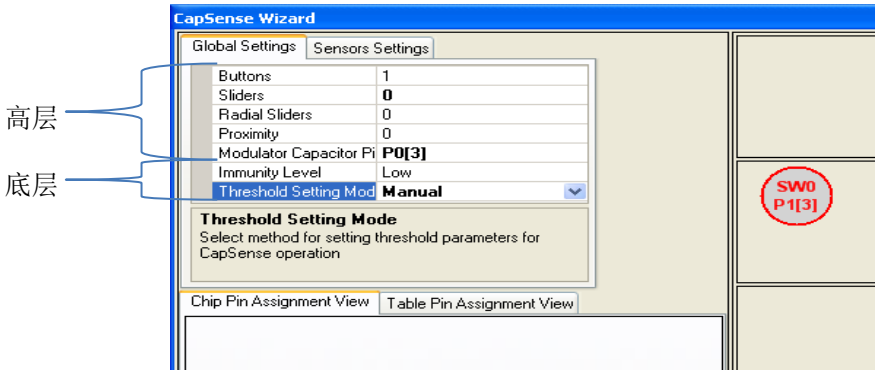



### *3.5.1.1 抗干扰级别*

该参数用于定义用户模块对外部噪声的抗干扰级别。选择高级抗干扰，可提供对外部噪声的最高抗干扰能力。中级抗干扰，可提供中等抗干扰能力。与低级抗干扰相比，设置中级抗干扰会消耗扫描时间和 RAM 存储空间的两倍，设置高级抗干扰会消耗扫描时间和 RAM 存储空间的三倍。

可选项为低、中和高。

图 3-13. PSoC Designer SmartSense_EMCPLUS 全局设置

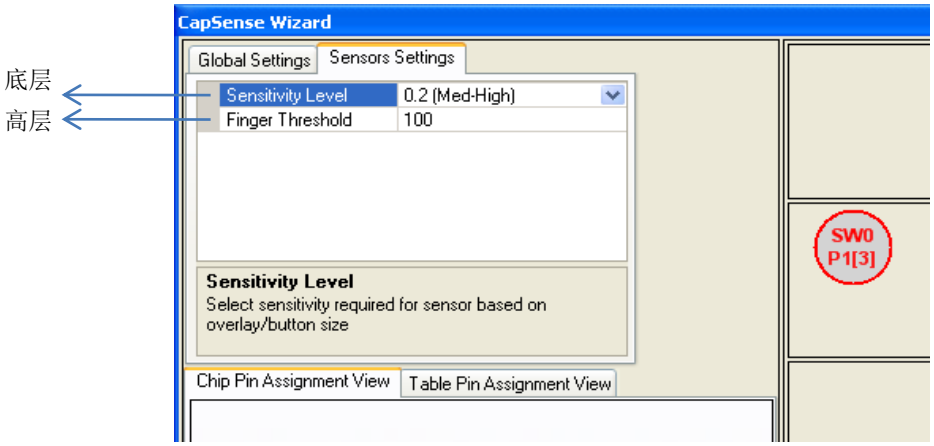



### *3.5.1.2 阈值设置模式*

选择手动阈值模式，可更加灵活地配置传感器手指阈值。选择自动阈值模式，可使 SmartSense_EMCPLUS 用户模式自动为每个传感器设置阈值。与手动阈值模式相比，自动阈值模式会消耗更多 RAM。

可选项为“手动”和“自动”。

图 3-14. PSoC Designer SmartSense_EMC_PLUS 传感器设置



#### *3.5.1.3 传感器灵敏度*

该参数用于升高或降低传感器的灵敏度。

可能值为 0.1 pF、0.2 pF、0.3 pF 和 0.4 pF

# 4. 使用用户模块对 CapSense 进行性能调校

理想的用户模块参数设置取决于电路板布局、按键尺寸、覆盖层材料和应用需求。有关这些因素的信息，请参考设计的注意事项。调校是确定最佳参数设置的过程，它的目的是保证传感器操作的稳健和可靠性。

## 4.1 基本注意事项

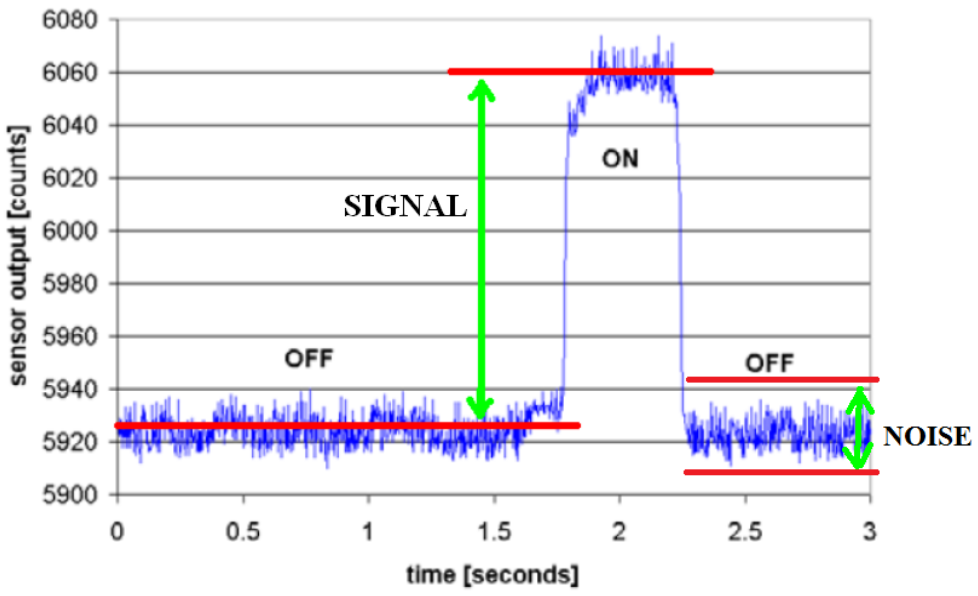
### 4.1.1 信号、噪声和信噪比

调校好的 CapSense 系统能够准确地识别传感器的 ON 和 OFF 状态。要实现该性能级别，CapSense 信号必须远远大于 CapSense 噪声。通过使用称为信噪比（SNR）的量，将 CapSense 信号与 CapSense 噪声进行比较。在讨论 CapSense SNR 的含义时，首先需要在触摸感应的背景下定义信号和噪声。

#### *4.1.1.1 CapSense 信号*

CapSense 信号是手指触摸传感器时，传感器原始计数的变化，如图 4-1 所示。传感器的输出是一个数字计数器，该计数器带有表示传感器电容的值。在该示例中，在手指没有触摸传感器时，平均输出为 5925。当手指触摸传感器时，平均输出增至 6060。由于 CapSense 信号跟踪手指产生的计数变化，因此信号值 = 6060 – 5925 = 135。

图 4-1. CapSense 信号与噪声




#### *4.1.1.2 CapSense 噪声*

CapSense 噪声表示在手指未触摸传感器时，传感器响应的峰峰值变化，如图 4-1 所示。在该示例中，手指未触摸时输出波形跳跃限幅为：最大为 5938 数值，最小则为 5912 数值。因为噪声是波形最大值与最小值之间的差别，因此噪声值 = 5938 – 5912 = 26。

#### *4.1.1.3 CapSense 信噪比*

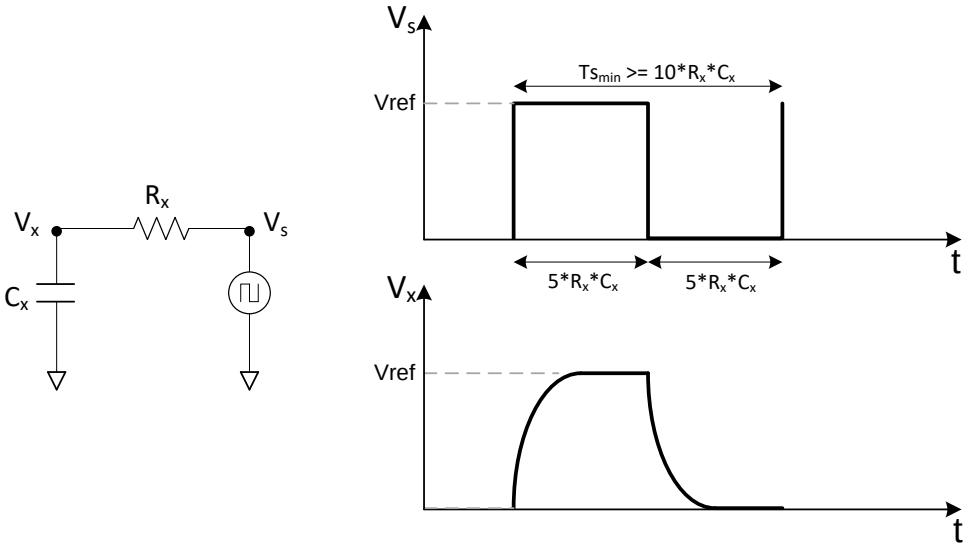
CapSense SNR 简单是信号和噪声的比例。继续该示例，如果信号值为 135、噪声值为 26，那么，SNR 为 135:26，既为 5.2:1。所推荐的 CapSense SNR 最小为 5:1，因此信号比噪声大 5 倍。滤波器通常在固件中实现，以降低噪声。更多有关信息，请参见软件滤波。

### 4.1.2 充电/放电率

在调校过程中，要获得最大灵敏度，必须在每个周期内对传感器电容器进行完全充电和放电。充电/放电路径以某个频率在两种状态的之间进行切换，该频率由 CSDPLUS 用户模块中的预充电时钟设置。

充电/放电路径包括串联电阻，这降低了电荷转移的速率。该电荷转移的变化率由 RC 时间常数描绘，该时间常数与传感器电容（C）和串联电阻（R）相关，如图 4-2 所示。

图 4-2. 充电/放电波形




将充电/放电率设置为与 RC 时间常数相兼容的级别。通常，每个切换周期为 5 RC，每个周期内发生两次切换（一次充电、一次放电）。用于计算最小值周期和最大频率的公式分别为：

$$Ts_{min} = 10 \times R_X C_X$$ 公式 4-1

$$fs_{max} = \frac{1}{10 \times R_X C_X}$$ 公式 4-2

例如，假定串联电阻包含 560 Ω 的外部电阻和 800 Ω 内部电阻，并且传感器的电容为标准电容：

$R_X$ = 1.4 kΩ

$C_X$= 24 pF

在本示例中，时间常数值和前端最大切换频率为：

$Ts_{min}$ = 0.34 µs

$fs_{max}$ = 3 MHz

### 4.1.3 基准线更新阈值验证的重要性

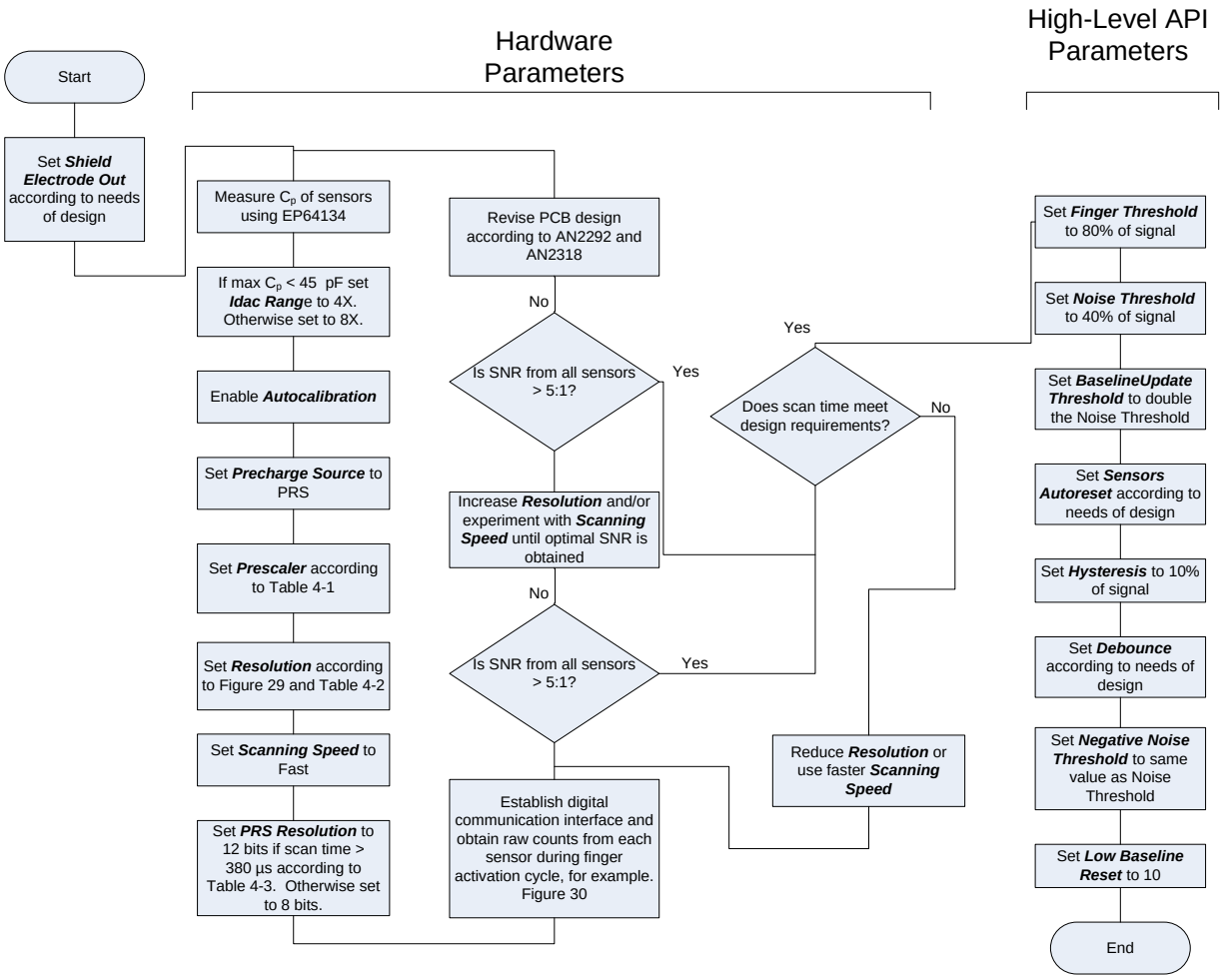
温度和湿度都会引起平均计数随时间浮动。基准线是 CapSense 测量值的参考计数量，这对于环境影响补偿而言起着重要作用。高级决策（例如，手指触摸和手指离开状态）是基于基准线建立的参考级别。由于每个传感器都具有与之相关的独特寄生电容，因此每个电容式传感器都有自己的基准线。

基准线以基准线更新阈值参数所设置的速率来跟踪计数变化。确保更新速度与预期应用相互匹配。如果更新速度过快，那么，基准线将补偿手指触摸产生的所有变量，并不予检测手指移动的情况。如果更新速度过慢，那么，相对慢的环境变化可能导致误触摸。开发期间，应对基准线更新阈值设置进行验证。

## 4.2 调校 CSD/CSDPLUS 用户模块

图 4-3 是展示 CSD/CSDPLUS UM 参数调校过程的流程图。CSD/CSDPLUS UM 参数可以分为两大类：低层（硬件）参数和高层 API 参数。这两种类别的参数以不同的方式影响电容式感应系统的行为。但是，由硬件参数设置和许多高层参数设置确定的每个传感器的灵敏度之间存在互补关系。更改任何硬件参数时，必须考虑这一点，以便保证相关的高层参数得到相应的调整。调校 CSD/CSDPLUS 用户模块参数应始终从硬件参数开始。

图 4-3. 调校 CSD/CSDPLUS 用户模块




硬件参数配置 CSD/CSDPLUS 方法用于将每个传感器的物理电容转换为数字代码的硬件。该部分介绍了这些参数，并提供有关如何根据系统特性和其他参数调整每个参数的指导。

默认情况下，硬件参数是适用于设计中所有 CapSense 传感器的全局设置。在每个传感器（CP）的总寄生电容，传感器灵敏度或两者都在很大范围内变化的设计中，可能没有适合所有传感器的全局硬件参数设置。在这种情况下，SetIdacValue(i)、 SetPrescaler(i)和 SetScanMode(i)等 API 函数均能够用于配置每个传感器的相应硬件参数，其中(i)是调用 ScanSensor(i) API 函数前所需要的传感器索引。

表 4-1 和表 4-3 提供了传感器 CP 为一些关键硬件参数的推荐调校值。CP 值取决于 PSoC 的特征、PCB 布局以及组装产品中其他组件的接近程度。根据上述原因，CP 值必须在系统最终组装原始状态测量；即在与系统提供服务时一样，具有相同的外壳以及覆盖层。测量 CP 值的最佳方法是使用 CapSense 控制器代码示例设计指南中提供的“使用

CY8C20xx6A CapSense 控制器测量传感器电容绝对值”代码示例。该项目使用 PSoC 测量系统中每个传感器的电容绝对值，因此要考虑影响 $C_P$ 的所有因素。请参见代码示例相关的文档以查阅安装和使用的步骤。

### 4.2.1 CSD/CSDPLUS 的 $C_{MOD}$ 推荐值

基于 CSD 的设计的 $C_{MOD}$ 推荐值为 **2.2 nF**。推荐使用 X7R 或 NPO 类型的电容，从而可以保证 $C_{MOD}$ 在不同的温度条件下保持稳定状态，同时电容电压不要低于 5 V。

### 4.2.2 $I_{DAC}$ 范围

对于最大传感器 $C_P$ 小于 45 pF 的项目，可以使用 4X；否则，使用 8X。

### 4.2.3 自动校准

在 CY8C20xx7/S CSD/CSDPLUS 设计中，应将自动校准始终设置为使能状态。自动校准算法成功设置 $I_{DAC}$ 的能力依赖于正确设置预分频器并且 $C_{MOD}$ 是建议的大小。

### 4.2.4 $I_{DAC}$ 值

禁用自动校准后，该参数将决定 $I_{DAC}$ 的当前输出。使能自动校准时（如上面的建议），该参数将被覆盖并失效。禁用自动校准时，提高该参数会降低原始计数基准线，反之亦然。此外，根据 CSD/CSDPLUS 用户模块中的 IDAC 值建议，应选择 $I_{DAC}$ 值，使原始计数为 $2^N$ 的 85%。

### 4.2.5 补偿 $I_{DAC}$ 值

禁用自动校准后，该参数将决定补偿 $I_{DAC}$ 的当前输出。该参数仅适用于 CSDPLUS 用户模块。使能自动校准时，会使用推荐设定，该参数将被覆盖，并且失效。禁用自动校准时，提高该参数会降低原始计数基准线，反之亦然。如果禁用了自动校准，那么可以通过以下方法来调校补偿 $I_{DAC}$ 值。

刚开始，仅使用调制 $I_{DAC}$ 将原始计数调校为其最大值的 85%（保持补偿 $I_{DAC}$ 值为 0）。然后将补偿 $I_{DAC}$ 值增大到调制 $I_{DAC}$ 值的 10%，并重新将调制 $I_{DAC}$ 值调到最大原始计数值的 85%。记录这时的信噪比（SNR）。使用一个更大的补偿 $I_{DAC}$ 值重复上述过程，直到获得最大信噪比为止。请注意，补偿 $I_{DAC}$ 值越大，信号值也越大（请参考 CSD/CSDPLUS 用户模块中的补偿 IDAC 值部分以获取更多详细信息）。此外，还要注意，当 $I_{DAC}$ 值等于 $I_{COMP}$ 值时，通常可以达到最佳信噪比。使能自动校准也会设置 $I_{DAC}$ 和 $I_{COMP}$，从而使 $I_{DAC}$ 等于 $I_{COMP}$，并且原始计数值大约为 $2^N$ 的 85%。

### 4.2.6 预充时钟源

该参数用于选择传感器开关时钟源。可用选项为预分频器（使用经过分频器的 IMO）或 PRS（将被分频的 IMO 时钟传递到伪随机发生器，从而提供扩频时钟）。PRS 提供了出色的抗噪能力和较低的噪音辐射，因此是预充时钟源的推荐默认设置。在一些情况下，预分频器预充时钟源可提供更高的信噪比。然而，使用铜线路时，信噪比的改善通常比较细微，并且也不能明显看出 PRS 的优点。

### 4.2.7 预分频器

预分频器是适用于 IMO 的分频器，用于开发预充电时钟。这是合理调校 CSD 设计时的最重要硬件 UM 参数。预分频器取决于所选的预充时钟源、IMO 和扫描传感器的 $C_P$ 值。表 4-1 显示了基于这些参数的预分频器推荐设置。

表 4-1. 基于预充时钟源、IMO 和 $C_P$ 的预分频器设置

| $C_P$（pF） | 预充时钟源 = PRS | | | 预充时钟源 = 预分频器 | | |
|---|---|---|---|---|---|---|
| | 预分频器 IMO = 24 MHz | 预分频器 IMO = 12 MHz | 预分频器 IMO = 6 MHz | 预分频器 IMO = 24 MHz | 预分频器 IMO = 12 MHz | 预分频器 IMO = 6 MHz |
| < 6 | 1 | 注释 1 | 注释 1 | 2 | 1 | 1 |
| 7–11 | 2 | 1 | 注释 1 | 4 | 2 | 1 |
| 12–15 | 2 | 1 | 注释 1 | 4 | 2 | 1 |
| 16–19 | 4 | 2 | 1 | 8 | 4 | 2 |
| 20–22 | 4 | 2 | 1 | 8 | 4 | 2 |
| 23–26 | 4 | 2 | 1 | 8 | 4 | 2 |
| 27–30 | 4 | 2 | 1 | 8 | 4 | 2 |

| $C_P$（pF） | 预充时钟源 = PRS | | | 预充时钟源 = 预分频器 | | |
|---|---|---|---|---|---|---|
| | 预分频器 IMO = 24 MHz | 预分频器 IMO = 12 MHz | 预分频器 IMO = 6 MHz | 预分频器 IMO = 24 MHz | 预分频器 IMO = 12 MHz | 预分频器 IMO = 6 MHz |
| 31–34 | 4 | 2 | 1 | 8 | 4 | 2 |
| 35–37 | 8 | 4 | 2 | 16 | 8 | 4 |
| 38–41 | 8 | 4 | 2 | 16 | 8 | 4 |
| 42–45 | 8 | 4 | 2 | 16 | 8 | 4 |
| 46–49 | 8 | 4 | 2 | 16 | 8 | 4 |
| 50–52 | 8 | 4 | 2 | 16 | 8 | 4 |
| 53–56 | 8 | 4 | 2 | 16 | 8 | 4 |
| 57–60 | 8 | 4 | 2 | 16 | 8 | 4 |

**注释 1**：不建议使用这种预充时钟源、预分频器和 $C_P$ 的组合。

## 4.2.8 分辨率

可选范围为 9 到 16 位。提高分辨率可提高传感器的灵敏度、信噪比和抗噪性能，但扫描时间被延长。扫描分辨率为 N 时，最大原始计数值（满量程范围）为 $2^N$-1。表 4-2 显示了基于 $C_P$ 和手指电容值 $C_F$ 的建议分辨率设置。$C_F$ 是手指触摸传感器时电容值的变化。$C_F$ 值取决于覆盖层厚度、传感器大小及传感器与其他大型导体的接近程度。图 4-4 显示了 $C_F$ 值，它可作为覆盖层厚度和圆形传感器直径的函数。

图 4-4. 基于覆盖层厚度和圆形传感器直径的手指电容值（$C_F$）

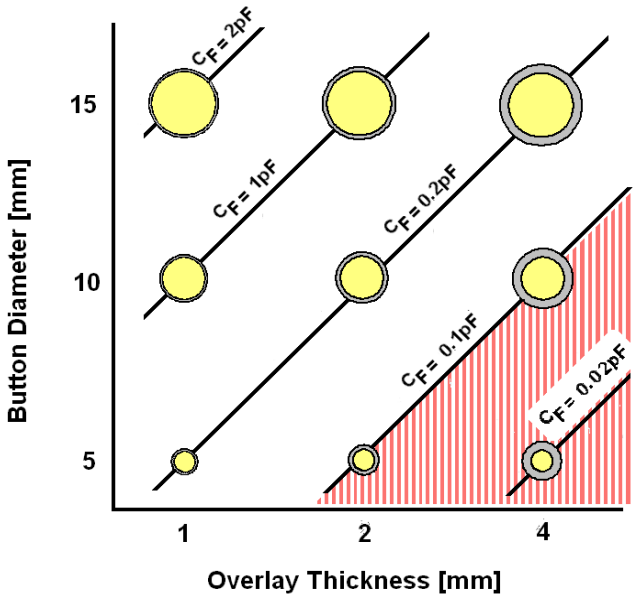



表 4-2. 基于手指电容值和 $C_P$ 的分辨率设置

| $C_P$（pF） | $C_F$ = 0.1 pF | $C_F$ = 0.2 pF | $C_F$ = 0.4 pF | $C_F$ = 0.8 pF |
|---|---|---|---|---|
| < 6 | 11 | 10 | 9 | 8 |
| 7–12 | 12 | 11 | 10 | 9 |
| 13–24 | 13 | 12 | 11 | 10 |
| 25–48 | 14 | 13 | 12 | 11 |
| > 49 | 15 | 14 | 13 | 12 |

### 4.2.9 扫描速度

该参数控制扫描结果的 LSB 集成时间。扫描速度选项包括：超快、快速、正常速度和慢速。建议初始值选择“快速”。在某些情况下（但非所有情况），较慢的扫描速度可以获得更高的信噪比，但扫描时间更长且功耗更大。表 4-3 显示了在不同分辨率和扫描速度下，单传感器的实际扫描时间（单位为微秒）。

表 4-3. 单传感器在不同分辨率和扫描速度下的扫描时间 (µs)

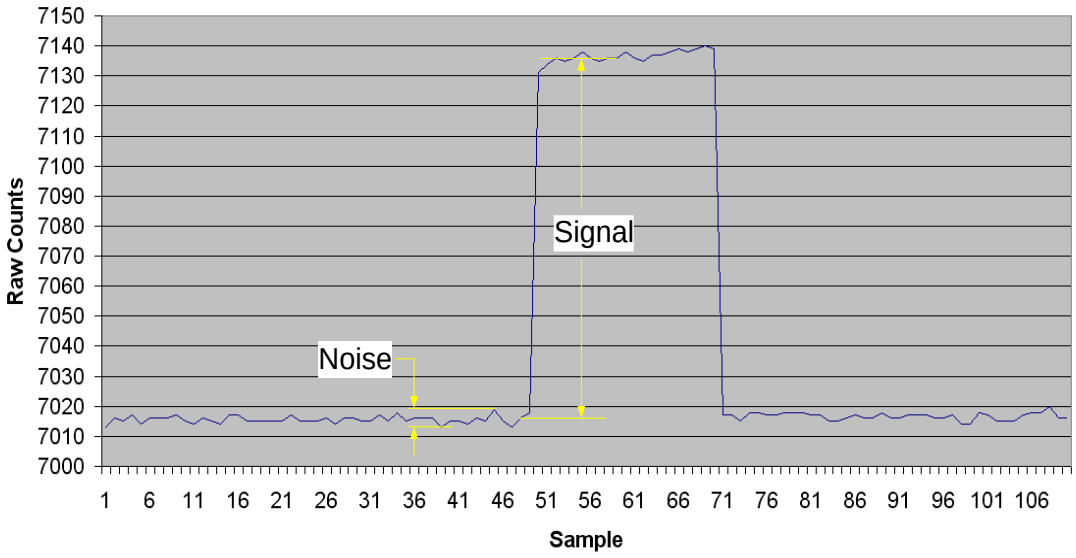
| 分辨率（位） | 扫描速度 | | | |
|---|---|---|---|---|
| | 超快速度 | 快速 | 正常速度 | 慢速 |
| 9 | 57 | 78 | 125 | 205 |
| 10 | 78 | 125 | 205 | 380 |
| 11 | 125 | 205 | 380 | 720 |
| 12 | 205 | 380 | 720 | 1400 |
| 13 | 380 | 720 | 1400 | 2800 |
| 14 | 720 | 1400 | 2800 | 5600 |
| 15 | 1400 | 2800 | 5600 | 11000 |
| 16 | 2800 | 5600 | 11000 | 22000 |

### 4.2.10 高层 API 参数

高层 API 参数决定了高级固件算法的性能，该算法用于区分传感器的激活与噪声，并补偿由环境因素所引起的信号漂移。为了确定适当的参数取值，您必须在系统中建立起数字通信接口，以监控每个传感器手指激活事件期间的原始计数、基准线和差值。数据分别存储于如下三个数组：CSDPLUS_waSnsBaseline[]、CSDPLUS_waSnsResult[] 和 CSDPLUS_waSnsDiff[]。如该数据所示，高层 API 参数的设置主要基于环境噪声和手指信号强度。噪声与信号强度取决于 EMI 环境、PCB 布局、覆盖层厚度及系统的其他物理特性。因此，该数据作为参数设置基础，须从系统的原始位置中获取，并且该系统须处于最终组装状态，并与未来实际应用时的 EMI 环境相同。

图 4-5 显示了传感器在一个手指激活周期中获取的典型原始计数值，即传感器被激活再取消激活。标签叠加在数据上，表示如何根据原始数据计算噪声和信号。在适当的情况下，后面的高层参数描述包括有关如何基于这些噪声和信号值设置每个参数的信息。根据 CapSense 设计最佳实践，SNR 必须至少为 5:1 才能实现稳健的 CapSense 系统操作。如果信噪比低于 5:1，则需调整硬件参数，或根据 CapSense 入门中的说明更改 PCB 布局，至少将信噪比提高至 5:1；亦可同时对二者进行调整。

图 4-5. 传感器在一个手指激活周期中的典型原始计数值




### 4.2.11 设置高层参数

为获得最佳参数设置，建议采用以下初始值：

- **手指阈值**：传感器为 ON 时设置为原始计数的 80%
- **噪声阈值**：传感器为 OFF 时设置为原始计数的 40%
- **负噪声阈值**：设置为噪声阈值
- **基准线更新阈值**：设置为噪声阈值的两倍
- **迟滞**：传感器为 ON 时设置为原始计数的 10%
- **低基准线复位**：设置为 50
- **传感器自动复位**：根据设计要求
- **去抖动**：根据设计要求

## 4.3 使用 SmartSense_EMCPLUS 用户模块

只要传感器寄生电容在 5pF 至 45pF 的范围内，手指触摸最小为 0.1pF，便可用 SmartSense_EMCPLUS 创建一个无需调校的 CapSense 设计。可使用 PSoC Designer 5.1 或更高版本中的 SmartSense_EMCPLUS 用户模块创建一个 SmartSense_EMCPLUS 设计。本节也介绍如何将现有的 CSD CapSense 设计导入到 SmartSense_EMC_PLUS 中。

### 4.3.1 SmartSense_EMC_PLUS 的指南

在应用中使用 SmartSense_EMCPLUS 用户模块时，请遵循这些指南。

- SmartSense_EMCPLUS 要求，设计电容式用户界面时，要遵循设计的注意事项章节中记录的布局和系统设计最佳实践。
- 所有 CSD/CSDPLUS 用户模块参数（例如：$I_{DAC}$ 值、预分频器周期、时钟分频器、扫描速度、分辨率等）由 SmartSense_EMCPLUS 用户模块在运行时确定。除非您了解使用 API 修改固件中的 CSD 参数对您的设计产生的具体影响，否则请勿进行此类修改。
- 要想将现有的 CSD/CSDPLUS 设计导入到 SmartSense_EMC_PLUS 中，
  - ☐ 请确保首先从程序中移除所有设置或者修改 CSD/CSDPLUS 参数的 API。

- ☐ 确保设计中所有 CapSense 传感器的寄生电容在环境和 PCB 生产工艺变化范围内介于 5pF 和 45 pF 之间。
- ☐ 请确保建议 $C_{MOD}$ 电容（X7R、2.2-NF 和额定电压超过 5 V）连接到用户模块向导中已选的 $C_{MOD}$ 端口引脚。

### 4.3.2 用户模块差异

SmartSense_EMCPLUS 用户模块和标准 CSD/CSDPLUS 用户模块之间的不同是：

- SmartSense_EMCPLUS 用户模块和标准 CSD/CSDPLUS 用户模块支持同样的 APIs。因此，除用户模块实例名称外，放置、配置、启动或调用其他 API 时都无需更改。
- 无需为调校设置任何用户模块参数，因为与调校相关的所有参数均由 SmartSense_EMCPLUS 用户模块在运行时自动设置。
- $C_{MOD}$ 电容值限制为 2.2 nF。 建议在所有 CapSense 应用中使用额定电压高于 5 V 的 X7R 或 NPO 电容。
- SmartSense_EMCPLUS 算法将每个传感器的信噪比维持在 5:1 到 11:1 之间，从而在使 Capsense 获得最佳性能的同时，确保其运行稳定。
- 根据传感器的寄生电容，算法将 SmartSense_EMCPLUS 用户模块的扫描时间限制为：在 24 MHz 运行模式下，每个传感器的扫描时间为 410 µs 到 2.8 ms。

### 4.3.3 SmartSense_EMC_PLUS 的建议 $C_{MOD}$ 值

基于 SmartSense_EMC_PLUS 的设计的建议 $C_{MOD}$ 值为 2.2 nF。建议使用 X7R 或者 NPO 类型的电容，以保证 $C_{MOD}$ 在不同的温度条件下保持稳定状态。电容的额定电压应不低于 5 V。

### 4.3.4 SmartSense_EMCPLUS 用户模块参数

该用户模块中，只需要设置四个参数。它们为：

- 传感器自动复位
- 去抖动
- 调制器电容引脚
- 灵敏度级别

#### *4.3.4.1 传感器自动复位*

该参数可确定更新基准线的时间：随时更新或只有信号差值低于噪声阈值时才更新。当设置为使能时，基准线随时更新。此设置限制传感器保持 ON 的最长时间（通常为 5 到 10 秒），但是当原始计数突然升高而没有任何东西接触传感器时，它会阻止传感器因系统的任何故障而永久 ON。

#### *4.3.4.2 去抖动*

去抖动参数为传感器的有效转换添加去抖动计数。为使传感器从非活动状态声明为活动状态，传感器应在连续扫描的去抖动次数期间保持手指触摸信号。该参数影响所有类似传感器。

#### *4.3.4.3 调制器电容引脚*

该参数选择的引脚连接着 2.2 nF/X7R/额定电压大于 5 V 的 $C_{MOD}$ 电容。可选引脚为 P0[1]和 P0[3]。

**注意**：必须有一个大小为 2.2 nF 的外部电容，SmartSense_EMCPLUS 才能正常工作。

#### *4.3.4.4 灵敏度级别*

灵敏度用于增加或减少传感器发出的信号强度。灵敏度值越低（0.1 pF），传感器发出信号越强。具有较厚覆盖层的设计需要更强的传感器信号才能正确实施。灵敏度选项有：“高（0.1 pF）”、“中高（0.2 pF）”、“中低（0.3 pF）”和“低（0.4 pF）”。

要产生更强的传感器信号（高灵敏度），SmartSense_EMCPLUS 用户模块必须消耗更多的传感器扫描时间。这意味着：传感器灵敏度等级设置为 0.1 PF（“高”）与设置为 0.2 PF（“中高”）相比，前者消耗更多的扫描时间。

最佳调校方法是查找产生要求的 5:1 信噪比时的最高灵敏度。可从灵敏度最高值（0.4 pF）开始调校，并按需要减少该值直至满足 5:1 信噪比

## 4.3.5 SmartSense_EMCPLUS 用户模块的特定指南

适用于 SmartSense_EMC 用户模块的所有指南均适用于 SmartSense_EMCPLUS 用户模块。有关 CapSense 设计和基于 SmartSense_EMC_PLUS 设计的通用说明，请参阅 CapSense 入门指南。本节介绍 SmartSense_EMCPLUS 用户模块的几个重要方面。

### *4.3.5.1 传感器扫描时间、响应时间和内存使用*

当使用 SmartSense_EMCPLUS 用户模块实现传感器时，传感器的扫描时间、响应时间以及 RAM 内存使用均取决于用户模块中选择的抗噪模式。

- 抗噪模式设置为“中”时的传感器扫描时间比设置为“低”时大一倍。抗噪模式设置为“高”时，传感器扫描时间比设置为“低”时大两倍。
- 随着扫描时间的增加，传感器的响应时间也同比增加。抗噪模式设置为“中”时，传感器响应时间比它被设置为“低”时大一倍。同样，抗噪模式设置为“高”时的传感器响应时间比设置为“低”时大两倍。
- 为实现稳健的电磁兼容算法，SmartSense_EMCPLUS 用户模块使用 RAM 存储器。因此，最高抗噪音模式（“高”）需要的 RAM 内存大约比模式为“低”时大两倍。抗噪音模式为“中”时所需要的 RAM 存储器大约比模式为“低”时大一倍。

### *4.3.5.2 IMO 容差和时间关键任务*

SmartSense_EMC 使能部分的 IMO 容差为+5%和–20%。

- 使用时间关键算法和逻辑时，须考虑 IMO 容差，以确保避免破坏固件的逻辑或算法。
- 如果项目使用中断，在分析中断延迟、ISR 执行时间等时，须考虑 ISO 容差。
- 每个基于 IMO 的时序分析（例如，IMO 定时器、固件中使用循环而导致的延迟，API 执行时间）必须考虑 IMO 容差，以确保应用固件的稳健。

### *4.3.5.3 I²C 工作速度*

I²C 接口工作频率被限制为 SmartSense_EMC 使能部分中用户模块实际工作频率的 80%。该限制由 20%的 IMO 容差导致。

- 这意味在 I²C 用户模块中选择 400 kHz 的时钟频率时，I²C 接口的最大工作频率为 320 kHz。同样，在 I²C 用户模块中选择 100 kHz 和 50 kHz 的时钟模式时，则其最大工作频率分别为 80 kHz 和 40 kHz。
- 使用 I²C 从设备接口时，主设备时钟需在前面提到的降频范围内运行。否则会导致数据损坏、I²C 总线连接或与 I²C 用户模块性能不一致。
- 使用 I²C 主设备模块只影响该接口的吞吐量。

在使用 I2CSBUF 用户模块的情况下，当 CPU 访问 32 字节专用缓冲区或者器件进入睡眠或深度睡眠模式时，强烈建议使用自动取消应答模式。更多详细信息，请参考 I2CSBUF 用户模块数据手册。

## 4.3.6 CapSense 传感器的扫描时间

为了保证在多种寄生电容值范围下手指响应灵敏度的一致性，SmartSense_EMCPLUS 用户模块自动选择用户模块的硬件参数。因此，传感器扫描时间并不固定。为大批量生产设计时，传感器扫描时间可根据 PCB 寄生电容值的变化而变化。

传感器的总扫描时间由四个因素决定。分别为：传感器寄生电容值、IMO 频率、CPU 的工作频率和 SmartSense_EMCPLUS 用户模块的灵敏度水平。

可以使用公式 4-3 和下表查找传感器的扫描时间。

$$\textit{扫描时间} = \textit{采样时间}(ST) + \textit{处理时间}(PT) \qquad \text{公式 4-3}$$

下表显示的是在不同 IMO 和灵敏度水平下的采样时间值。

表 4-4. IMO = 24 MHz 的传感器采样时间

| 灵敏度 = 0.2 pF | | 灵敏度 = 0.3 pF | | 灵敏度 = 0.4 pF | |
|---|---|---|---|---|---|
| $C_P$（pF） | ST (µs) | $C_P$（pF） | ST（µs） | $C_P$（pF） | ST（µs） |
| 8 到 10 | 340 | 8 到 17 | 340 | 8 到 10 | 170 |
| 10 到 23 | 680 | 17 到 35 | 680 | 10 到 23 | 340 |
| 23 到 41 | 1360 | 35 到 41 | 1360 | 23 到 41 | 680 |
| 41 到 45 | 2730 | 41 到 45 | 2730 | 41 到 45 | 1360 |

表 4-5. IMO = 12 MHz 的传感器采样时间

| 灵敏度 = 0.2 pF | | 灵敏度 = 0.3 pF | | 灵敏度 = 0.4 pF | |
|---|---|---|---|---|---|
| $C_P$（pF） | ST（µs） | $C_P$（pF） | ST（µs） | $C_P$（pF） | ST（µs） |
| 8 到 10 | 680 | 8 到 17 | 680 | 8 到 10 | 340 |
| 10 到 23 | 1360 | 17 到 35 | 1360 | 10 到 23 | 680 |
| 23 到 41 | 2730 | 35 到 41 | 2730 | 23 到 41 | 1360 |
| 41 到 45 | 5460 | 41 到 45 | 5460 | 41 到 45 | 2730 |

表 4-6. IMO = 6 MHz 的传感器采样时间

| 灵敏度 = 0.2 pF | | 灵敏度 = 0.3 pF | | 灵敏度 = 0.4 pF | |
|---|---|---|---|---|---|
| $C_P$（pF） | ST（µs） | $C_P$（pF） | ST（µs） | $C_P$（pF） | ST（µs） |
| 8 到 11 | 680 | 8 到 10 | 680 | 8 到 11 | 680 |
| 11 到 23 | 1360 | 10 到 17 | 1360 | 11 到 23 | 1360 |
| 23 到 42 | 2730 | 17 到 35 | 2730 | 23 到 41 | 2730 |
| 42 到 45 | 5460 | 35 到 41 | 5460 | 41 到 45 | 5460 |
| | | 41 到 45 | 10920 | | |

表 4-7 显示不同 CPU 频率下的处理时间值。

表 4-7. 传感器处理时间

| CPU CLK | 处理时间（PT），单位：毫秒 |
|---|---|
| 24 | 71 |
| 12 | 142 |
| 6 | 284 |
| 3 | 568 |

例如，如果 CapSense 系统的 IMO 频率为 24 MHz 、CPU 时钟为 6 MHz （IMO/4）并且 SmartSense_EMCPLUS 灵敏度水平为 0.3 pF，则寄生电容值大约为 15 pF 的传感器扫描时间可根据之前的表中的数据、使用公式 4-1 进行计算。之前提到配置（IMO 为 24 MHz，灵敏度为 0.3 pF）的采样时间根据表 4-4 进行选择，采样时间为 680 µs。之前提到配置（CPU 时钟为 6 MHz）的处理时间根据表 4-7 进行选择，处理时间为 284 µs。

因此，这种配置的总扫描时间为 680 + 284 = 964 µs。多个传感器的扫描时间是单个传感器扫描时间的总和。

## 4.3.7 SmartSense_EMCPLUS 响应时间

考虑以下使用标准 CSD 和典型 CapSense 扫描固件的应用。

- 三个 CapSense 传感器，它们的寄生电容值在 5 pF 到 10 pF 的范围内
- IMO 频率为 12 MHz，CPU 时钟频率为 12 MHz
- 传感器灵敏级别为 0.4 pF

- 去抖动值 = 3

根据之前的表格，每个传感器的扫描需要 482 µs ，三个传感器的总扫描时间为 1.45 ms。以下固件样本需要 1 ms 的额外固件执行时间；因此，循环执行时间为 2.45 ms。

```
while (1)
{
    SmartSense_EMC_PLUS_ScanAllSensors();
    SmartSense_EMCPLUS _UpdateAllBaselines();

    if(SmartSense_EMCPLUS _bIsAnySensorActive() )
    {
        //1ms firmware routines
    }
}
```

这意味着：当 CapSense 传感器被激活时，固件在 7.35 ms 内开启传感器（为了接收连续扫描的去抖动数量，传感器应处于活动状态）。这通常被称为 CapSense 系统的响应时间。

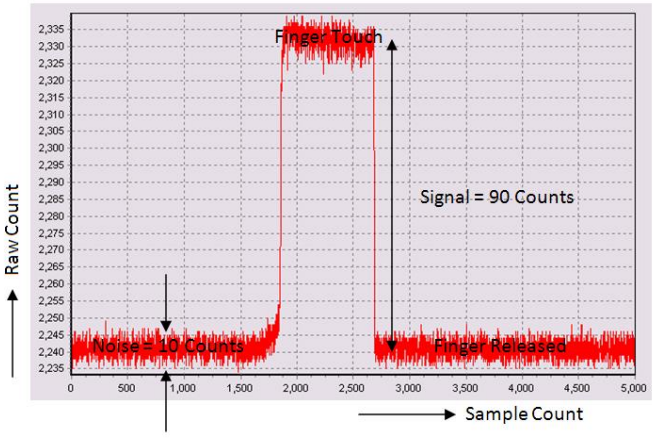
如果扫描时间随着寄生电容的变化发生相应变化，那么由于程序变化而导致的传感器寄生电容变化对响应时间会产生怎样的影响？在这种情况下，响应时间可能会增加（响应变慢）。这样可能会对传感器的性能产生负面影响。在下一节中将介绍如何实现稳健的固件设计。

## 4.3.8 使用 SmartSense_EMCPLUS 用户模块实现最低信噪比的方法

SmartSense_EMCPLUS 是先进的电磁器件设计，该设计基于 CSD 的 SmartSense 用户模块，并且不需要繁琐的调校过程。然而，使用 SmartSense_EMCPLUS 用户模块时，有两个简单的步骤能确保设计的稳健。

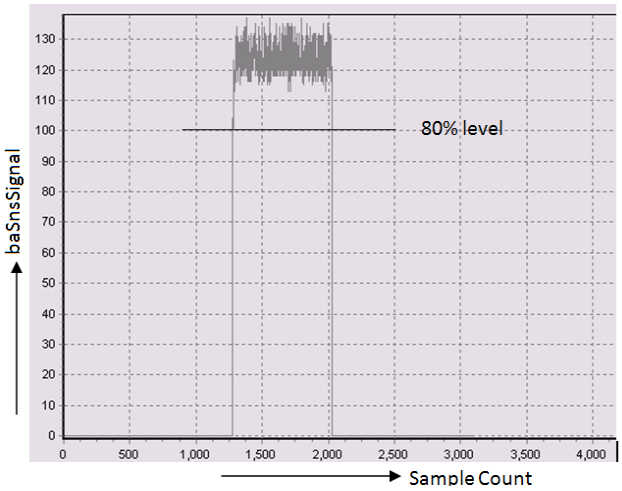
1. 建立实时监测工具，监测 CapSense 用户模块参数以测量传感器信号。在调校期间，可以观察到传感器原始计数（SmartSense_EMCPLUS_waSnsResult）、传感器标准化信号（SmartSense_EMCPLUS_baSnsSignal）和传感器手指阈值（SmartSense_EMCPLUS_baBtnFThreshold）。不要使用 LCD 或其他任何数字显示屏来监测计数，因为它们太慢，您无法观察到数据的动态变化。建议使用的数据监测工具是多图或 I²C USB 桥控制面板。
2. 将灵敏度设置到 0.4 pF（低），并计算 SNR。图 4-6 显示了手指触摸时的典型原始计数图。根据 CapSense 最佳实践，稳健设计的信噪比应大于 5:1。如果测量得到的信噪比大于 10:1，则会使灵敏度的值下降到下一可能的操作，直到获得的信噪比大于 5:1 并小于 10:1 为止。

图 4-6. 典型传感器（有手指触摸）的原始计数图




3. 如果您在设计中使用自动手指阈值功能，那么执行上一步后即完成了。如果您正在使用灵活手指阈值功能，您也应该为每个传感器设置单独的手指阈值以完成该操作。为了设置手指阈值，需要监测传感器信号（SmartSense_EMCPLUS_baSnsSignal），并在触摸传感器时，设置手指阈值为传感器信号的 80%左右。该操作就此完成。图 4-7 显示了典型传感器信号和手指阈值。

图 4-7. 典型传感器（有手指触摸）的传感器信号




## 4.3.9 固件设计指南

CapSense 传感器响应时间可根据传感器寄生电容值的增加而变化。观察循环执行时间（请看以下示例代码）也很重要，这也可能会增加。当所有传感器的寄生电容值小于 10 pF 时，固件程序执行率是 2.45 ms。如果增加传感器扫描时间，则频率将会增加，因为此传感器寄生电容的提高是基于处理过程变化的。

下面显示的是根据主循环执行时间切换端口引脚的示例代码。

```
while (1)
{
    SmartSense_EMC_PLUS_ScanAllSensors();
    SmartSense_EMC_PLUS_UpdateAllBaselines();

    if(SmartSense_EMC_PLUS_bIsAnySensorActive() )
        {
            //1ms firmware routines
        }

    PRT0DR_Shadow ^= 0x01;
    PRT0DR = PRT0DR_Shadow;
}
```

Port_0[1]引脚上的信号周期为 4.9 ms（此时间是端口引脚切换的循环时间的两倍）。如果一个传感器的寄生电容提高到 15 pF，则扫描时间是 1.78 ms；因此 Port_0[1]上的信号周期将为 5.6 ms。

如果该传感器的寄生电容接近 SmartSense_EMCPLUS 电容组的边界（例如：9 pF 非常接近 10 pF 边界），那么 SmartSense_EMCPLUS 可能会因工艺变化而在应用中选择相邻扫描时间。鉴于此，同一设计的不同产品部件可能有两个不同的主循环执行次数和响应时间。

基于以上讨论，固件不应该依赖于传感器的扫描时间以实现其他的功能（例如，软件 PWM、软件延迟等）。在实现看门狗定时器（WDT）程序中设定 WDT 到期时间时就应该考虑上述事实。

下面显示的是一个简单的固件实现示例，说明了如何使用 Timer16 用户模块来获取一致的主循环执行时间。

```
// Main program
BYTE bTimerTicks = 0;

#pragma interrupt_handler myTimer_ISR_Handler;
void myTimer_ISR_Handler( void );

void main()
{
M8C_EnableGInt;
```

```
SmartSense_EMC_PLUS_Start();
SmartSense_EMC_PLUS_ScanAllSensors();
SmartSense_EMC_PLUS_SetDefaultFingerThresholds() ;

Timer16_EnableInt();
Timer16_SetPeriod (TIMEOUT_10MS) ;
Timer16_Start();

while( 1 )
{
       /* Scan all 3 sensors and update
        Baseline */
       SmartSense_EMC_PLUS_ScanAllSensors();
       SmartSense_EMC_PLUS_UpdateAllBaselines();

       /* Wait till timer expires or
        sleep here */

       while (bTimerTicks != 1) ;
       bTimerTicks = 0 ;

       if(CSDAUTO_bIsAnySensorActive() )
       {
           //1ms firmware routines
       }

       // Toggle Port_0[1]
       PRT0DR_Shadow ^= 0x01 ;
       PRT0DR = PRT0DR_Shadow ;
   }
}

// Timer16 ISR program
void myTimer_ISR_Handler(void)
{
bTimerTicks++;
}
```

在上述示例中，即使传感器完成扫描，程序仍会等待定时器到期。应该考虑最差情况下主循环执行时间来选择定时器周期。该值是最差情况下每个 CapSense 传感器扫描时间的总和。如果传感器的寄生电容接近 SmartSense_EMCPLUS 电容组的边界，则选择较高的扫描时间（使用表 4-5）来计算。

使用 SmartSense_EMCPLUS 用户模块，您可以在系统中轻松实现电容式触摸感应用户界面。面对 PCB 生产工艺的变化和其他的变化，它消除了调校过程中的困难，也提高了生产力。因此，最好的选择是将现有的基于 CSD 的 CapSense 设计移植到 SmartSense_EMCPLUS，并在新设计中使用 SmartSense_EMCPLUS。

SmartSense_EMCPLUS 的主循环执行时间和扫描时间都根据工艺的变化而改变。尽管这不会影响 CapSense 的性能，但在利用 SmartSense_EMCPLUS 自动调校技术来实现 CapSense PLUS 应用时，固件开发人员应该将此考虑在内。

## 4.4 将设计从 CY8C20xx6A/AS 移植到 CY8C20xx7/S

CapSense 控制器的 CY8C20xx7/S 系列提供了基于 QuiteZone™ 技术的优越抗噪能力以及 0.1 pF 灵敏度（信噪比为 5:1）本节汇总了移植现有 CY8C20xx6A/AS 设计的关键点。

### 4.4.1 不再支持/用户模块

- CY8C20xx7/S 系列不再支持 USB 接口。
- CY8C20xx7/S 系列不再支持片上调试（OCD）。
- CY8C20xx7/S 系列不支持 CSA 和 CSA_EMC 用户模块。

### 4.4.2 改进和新特性

- 基于 CSD 的改进型 CapSense 传感引擎可提供 0.1 pF 的灵敏度。
- 新型的改进 I²C 从设备接口（I2CSBUF 用户模块）使用专用的 32 字节缓冲区且可消除由从设备实现的时钟延展。
- 改进的 I²C 接口还支持 I²C 从设备地址匹配事件发生时引起的唤醒中断。

使用表 4-8 为您的设计选择正确的 I²C 用户模块。

表 4-8. I²C 特性

| 系统要求 | I2CSBUF | EzI2C | I2CHW | I2Cm |
|---|---|---|---|---|
| 在 I²C 从设备地址匹配时唤醒 | 支持 | 支持 | 不支持 | N/A |
| 主设备需要时钟延展 | 不支持 | 支持 | 支持 | N/A |
| I²C 数据缓冲区的大小 | 1 到 32 个字节 | 1 到 255 个字节 | 1 到 255 个字节 | N/A |
| I²C 主设备 | 不支持 | 不支持 | 不支持 | 支持 |

### 4.4.3 引脚兼容性

表 4-9. 引脚兼容性

| 封装 | 与 CY8C20xx6A/AS 相比的引脚兼容性 |
|---|---|
| 16-SOIC | CY8C20xx7/S 提供的新封装 |
| 16-QFN | 引脚-引脚兼容 |
| 24 QFN | 一个引脚 — 在 CY8C20xx7/S 系列中引脚 23 是 Vss，而在 CY8C20xx6A/AS 中它则是一个 I/O。 |
| 32-QFN | 两个引脚 — 在 CY8C20xx7/S 系列中，引脚 28 是 I/O 而引脚 29 是 Vss；在 CY8C20xx6A/AS 系列中，引脚 28 是 Vss 而引脚 29 是 I/O。 |
| 30-WLCSP | 引脚-引脚兼容 |
| 48-QFN | 两个引脚 — 引脚 36 和引脚 45 是 NC，在 CY8C20xx6A/AS 系列中，两个引脚都是 I/O |

# 5. 设计的注意事项

当您的应用设计中采用电容式触摸感应技术时，必须将 CapSense 元件置入较大的框架中。认真考虑从 PCB 布局到用户界面到最终使用操作环境的各个细节层次，可以保证稳健可靠的系统性能。有关 EMC 注意事项的更多信息，请参见 CapSense 入门一节。

## 5.1 覆盖层选择

在 CapSense 基本原理部分中，使用公式 4-1 计算手指电容，如下所示：

$$C_F = \frac{\varepsilon_0 \varepsilon_r A}{D}$$

其中：

$\varepsilon_0$ = 空气介电常数

$\varepsilon_r$ = 覆盖层的介电常数

A = 手指与传感器板覆盖层的接触面积（$mm^2$）

D = 覆盖层厚度（mm）

若要增大 CapSense 信号的强度，请选择具有较高介电常数的覆盖层材料，降低覆盖层的厚度，并增大按键直径。

表 5-1. 覆盖层材料介电强度

| 材料 | $\varepsilon_r$ | 击穿电压（V/mm） | 电压为 12 kV 时的<br>最小覆盖层厚度（mm） |
|---|---|---|---|
| 空气 | 1.0 | 1200–2800 | 10 |
| 干木材 | 1.2–2.5 | 3900 | 3 |
| 普通玻璃 | 7.6–8.0 | 7900 | 1.5 |
| 硼硅酸盐玻璃（Pyrex®） | 6.0 | 13,000 | 0.9 |
| PMMA 塑料（Plexiglas®） | 2.8 | 13,000 | 0.9 |
| ABS | 2.4–4.1 | 16,000 | 0.8 |
| 聚碳酸酯（Lexan®） | 2.9–3.0 | 16,000 | 0.8 |
| 福米卡 | 4.6–4.9 | 18,000 | 0.7 |
| FR-4 | 4.8 | 28,000 | 0.4 |
| PET 薄膜 – （Mylar®） | 3.2 | 280,000 | 0.04 |
| 聚酰亚胺薄膜 – （Kapton®） | 2.9-3.9 | 290,000 | 0.04 |

导电材料不能用做覆盖层，因为它会与电场模式相干扰。因此，请勿在覆盖层中使用包含金属颗粒的油漆。

粘合剂常用来将覆盖层粘贴到 CapSense PCB 上。3M™有一种产品号为 200MP 的透明丙烯酸粘合膜，可用于 CapSense 应用。这种特殊的粘合剂是从纸背的胶带卷上抽取（3M 产品号为 467MP 和 468MP）的。

## 5.2 ESD 保护

考虑周全的系统设计自然会具有稳健的抗 ESD 能力。考虑到在您的产品（特别是用户界面产品）中发生的接触放电方式，可能需要承受 18 kV 的放电事件，并且不对 CapSense 控制器产生任何损害。

CapSense 控制器引脚可承受 2 kV 的直接电流。在大多数情况下，覆盖层材料能为控制器引脚提供充分的 ESD 保护。表 5-1 列出了为使 CapSense 传感器避免经受 12 kV 放电所需要的各种覆盖层材料的厚度（按照 IEC 61000-4-2 规定）。如果覆盖层材料无法提供足够的保护，将按以下顺序应用 ESD 对策：防止、重定向、钳制。

### 5.2.1 预防

确保触摸表面上所有路径的击穿电压均大于潜在高电压接触。此外，设计您的系统以确保 CapSense 控制器和 ESD 源之间保持适当的距离。如果无法保持足够的距离，可在 ESD 源和 CapSense 控制器之间放置一个高击穿电压的保护层。厚度为 5 mil 的一层 Kapton®胶带可承受 18 kV 电压。

### 5.2.2 重定向

如果您的产品空间密集，则可能无法防止放电事件。在这种情况下，您可以通过控制放电发生的位置来保护 CapSense 控制器。标准做法是在连接到底盘接地的电路板周界上放置一个保护环。按照 PCB 布局指南中的建议，在按键或滑条传感器周围提供一个网格接地层可重定向 ESD 事件，使其远离传感器和 CapSense 控制器。

### 5.2.3 钳制

当必须将 CapSense 传感器放置在触摸表面附近时，重新定向路径的方法可能并不实用。在这种情况下，合理的做法是添加串联电阻或者专用 ESD 保护器件。

建议串联电组值为 560 Ω。

更有效的方法是在易受影响的走线上提供特殊用途 ESD 保护器件。CapSense 的 ESD 保护器件必须是低电容的。表 5-2 列出了针对 CapSense 控制器使用的建议器件。

表 5-2. 建议用于 CapSense 的低电容 ESD 保护器件

| ESD 保护器件 | | 输入电容 | 漏电流 | 接触放电的最大限制 | 空气放电的最大限制 |
|---|---|---|---|---|---|
| 制造商 | 器件型号 | | | | |
| Littelfuse（力特） | SP723 | 5 pF | 2 nA | 8 kV | 15 kV |
| Vishay | VBUS05L1-DD1 | 0.3 pF | 0.1 µA < | ±15 kV | ±16 kV |
| NXP | NUP1301 | 0.75 pF | 30 nA | 8 kV | 15 kV |

## 5.3 电磁兼容性（EMC）的注意事项

### 5.3.1 辐射干扰

辐射电能可能影响系统测量，并可能影响处理器内核的运行过程。这种干扰进入 PCB 级别上的 PSoC 芯片，穿过 CapSense 传感器走线，然后穿过其他所有数字和模拟输入。最小化射频干扰影响的布局指南如下：

- **接地层**：在 PCB 上提供一个接地层。
- **串联电阻**：在 CapSense 控制器引脚的 10 mm 内安装串联电阻。
  - ☐ 建议 CapSense 输入线路串联电阻值为 560 Ω。
  - ☐ 建议为通信线（I²C 和 SPI）使用 330 Ω 的串联电阻。
- **走线长度**：应该尽量缩短走线长度。
- **电流环路区域**：尽量减少电流的返回路径。应在传感器和走线的 1 cm 范围内提供网格接地（而不是实体填充）来减小寄生电容的影响。

- **RF 源位置**：隔离带有噪声源（如 LCD 反相器和开关电源（SMPS））的系统，以使此干扰与 CapSense 输入相分隔。电源屏蔽是另一种防止干扰的通用方法。

### 5.3.2 辐射

选择低频率的开关电容时钟，可以减少 CapSense 发出的辐射。在固件中使用预分频器选项控制此时钟。增加预分频器的值将降低开关时钟的频率。

### 5.3.3 抗传导干扰和辐射

通过与其他系统的互连而进入系统的噪声被称为传导噪声。这些互连包括电源和通信线。因为 CapSense 控制器是低功耗器件，所以必须避免传导辐射。以下指南可帮助减少传导辐射和干扰：

- 按照数据手册的建议使用去耦电容。
- 在系统电源的输入上添加双向滤波器。这非常有助于抗传导辐射和干扰。π 型滤波器能够防止电源噪声影响敏感器件，但同时也会阻隔该敏感器件耦合返回到电源层的开关噪声。
- 如果 CapSense 控制器 PCB 通过线缆连接到电源，请尽量减小线缆长度并考虑使用屏蔽线缆。
- 请在电源或者通信线周围放置一个铁氧体磁珠，以滤除高频噪声。

## 5.4 软件滤波

软件滤波是处理高级系统噪音的技术之一。表 5-3 列出了对 CapSense 有用的滤波器类型。

表 5-3. CapSense 滤波器表

| 类型 | 描述 | 应用 |
|---|---|---|
| 均值 | 具有同样的加权系数的有限脉冲响应滤波器（无反馈路径） | 来自电源的周期性噪声 |
| IIR | 具有与 RC 滤波器类似的阶跃响应的无限脉冲响应滤波器（有反馈路径） | 高频白噪声（1/f 噪声） |
| 中值滤波器 | 从大小为 N 的缓冲区计算中值输入值的非线性滤波器 | 来自电机和开关式电源的噪声毛刺 |
| 抖动 | 根据之前输入来限制当前输入的非线性滤波器 | 来自厚覆盖层的噪声（SNR < 5:1），对滑条中心数据非常有用。 |
| 基于事件的滤波器 | 对传感器数据中观察到的模式进行预定义响应的非线性滤波器 | 通常在非触摸事件中使用，以阻止 CapSense 数据传输 |
| 基于规则的滤波器 | 对传感器数据观察到的样本进行预定义响应的非线性滤波器 | 通常在触摸表面的正常操作过程中使用，以响应特殊情况，例如，意外选择多个按键 |

表 5-4 详细说明了不同软件滤波器的 RAM 和闪存要求。每种滤波器所需的闪存空间取决于编译器的性能。这里列出的要求适用于 ImageCraft 编译器和 ImageCraft Pro 编译器。

表 5-4. RAM 和闪存要求

| 滤波器类型 | 滤波器的阶位 | RAM（每个传感器的字节） | 闪存（字节）ImageCraft 编译器 | 闪存（字节）ImageCraft Pro 编译器 |
|---|---|---|---|---|
| 均值滤波器 | 2–8 | 6 | 675 | 665 |
| IIR | 1 | 2 | 429 | 412 |
| | 2 | 6 | 767 | 622 |
| 中值滤波器 | 3 | 6 | 516 | 450 |
| | 5 | 10 | 516 | 450 |
| 用于原始计数的抖动滤波器 | N/A | 2 | 277 | 250 |
| 用于滑条中心的抖动滤波器 | N/A | 2 | 131 | 109 |

# 5.5 功耗

## 5.5.1 系统设计建议

对于许多设计而言，最小化功耗是重要目标。有多种方法可降低 CapSense 电容式触摸感应系统的功耗。

- 将 GPIO 驱动模式设置为低功耗
- 关闭高功耗模块
- 优化 CPU 速度以减低功耗
- 在较低 $V_{DD}$ 下操作

除了这些方法外，睡眠-扫描方法也很有效。

## 5.5.2 睡眠-扫描方法

在一般应用中，CapSense 控制器不需要一直处于活动状态。可将器件置于睡眠状态，以停止器件的 CPU 和主要模块。在睡眠状态下，该器件所消耗的电流远低于有效电流。

使用以下公式可计算器件在较长周期中所消耗的平均电流。

$$I_{AVE} = \frac{(I_{Act} \times t_{Act}) + (I_{Slp} \times t_{Slp})}{T}$$ 公式 5-1

器件的平均功耗可按以下公式计算：

$$P_{AVE} = V_{DD} \times I_{AVE}$$ 公式 5-2

## 5.5.3 响应时间与功耗

如公式 5-2 所述，可以通过降低 $I_{AVE}$ 或 $V_{DD}$ 来减少平均功耗。增加睡眠时间可减少 $I_{AVE}$。延长睡眠时间到较高值会导致 CapSense 按键响应时间差。因此，睡眠时间必须符合系统要求。

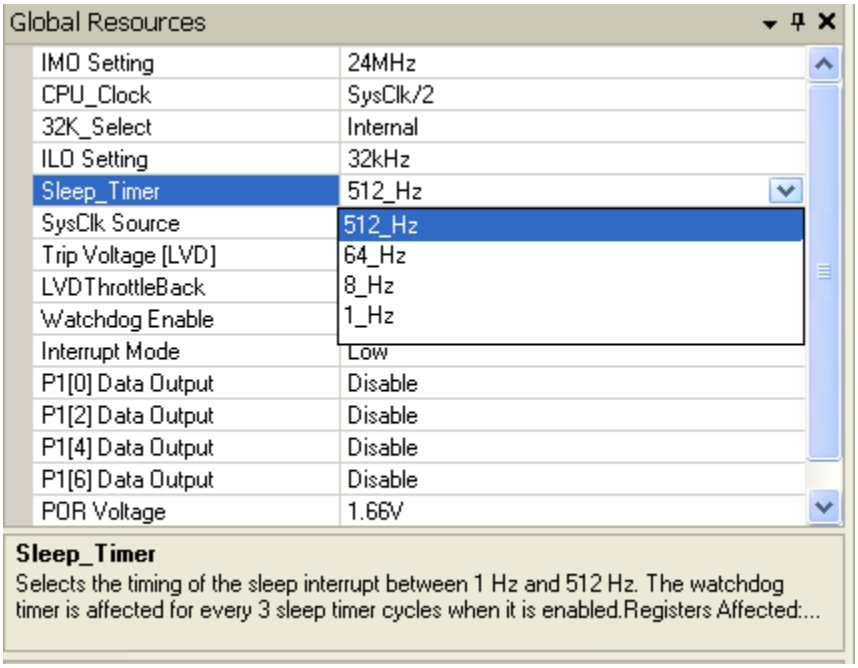
在任何应用中，若功耗和响应时间都是需要考虑的重要参数，则可使用下述优化方法，即同时采用连续扫描和睡眠-扫描模式。使用该方法时，器件在大部分时间将处于睡眠扫描模式。如先前章节所述，器件会周期性地扫描传感器并进入睡眠，从而使用较低的功耗。用户通过触摸传感器来操作系统时，器件将切换到连续扫描模式，传感器便连续扫描而不调用睡眠模式，从而大量缩短响应时间。在指定的超时周期内，器件仍然处于持续扫描模式。如果在该超时周期内，用户未触摸任何传感器，则器件会切换回睡眠-扫描模式。

### 5.5.4 测量平均功耗

下列指南介绍了使用睡眠-扫描方法时确定平均功耗的各个步骤：

1. 编译一个扫描所有传感器而不进入睡眠模式（持续扫描模式）的项目。扫描传感器前，代码需要具备引脚切换功能。输出引脚的状态切换可当做时间标记，该时间标记可通过示波器跟踪。
2. 将项目加载到 CapSense 器件中，并测量电流消耗。将测量好的电流分配给 $I_{ACT}$。
3. 从数据手册中获取睡眠电流信息，该电流是 $I_{SLP}$。
4. 通过示波器监控切换的输出引脚，并测量两次切换之间的时间间隔。这给出活动时间。将该值分配给 $t_{ACT}$。
5. 对项目应用睡眠-扫描模式。通过在全局资源窗口中选择睡眠定时器频率来设置睡眠扫描周期的时间周期 T，如图 5-1 所示。
6. 从睡眠-扫描周期时间中减去活动时间就可得出睡眠时间。$T_{SLP} = T – t_{ACT}$。
7. 使用公式 5-1 计算平均电流。
8. 使用公式 5-2 计算平均功耗。

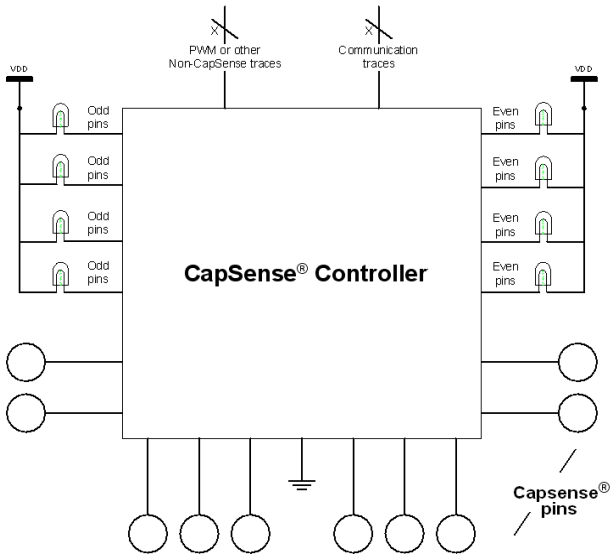
图 5-1. 全局资源窗口




## 5.6 引脚分配

减少 CapSense 传感器走线和通信线以及非 CapSense 走线之间相互作用的一种有效方法是使用端口分配将它们隔离。图 5-2 显示了针对 QFN 封装进行此隔离的基本示例。由于各个功能已被隔离，CapSense 控制器得到定向，因此保证通信、LED 和感应走线之间不存在交叉。

图 5-2. 通信、CapSense 和 LED 的建议端口隔离




CapSense 控制器的所有 GPIO 均能实现 CapSense 传感器和驱动 LED。但如果 GPIO 在某个项目中同时实现 CapSense 传感器并驱动 LED 时，建议根据表 5-5 使用 GPIO，从而保证最佳性能。如果 GPIO 在设计中仅作为 CapSense 传感器使用或者仅作为 LED 驱动使用，那么不用考虑表 5-5 中的限制。

表 5-5. 实现 LED 驱动和 CapSense 传感器的推荐 GPIO

| 适用于 LED 驱动 | 适用于 CapSense 传感器和 CMOD |
|---|---|
| P0.1、P2.5、P2.3、P2.1、P4.1、P3.7、P3.5、P3.3、P3.1、P1.7、P1.5、P1.3、P1.1 | P1.0、P1.2、P1.4、P1.6、P3.0、P3.2、P3.4、P3.6、P4.0、P4.2、P2.0、P2.2、P2.4、P0.0、P0.2、P0.4、P0.6、P0.3 |

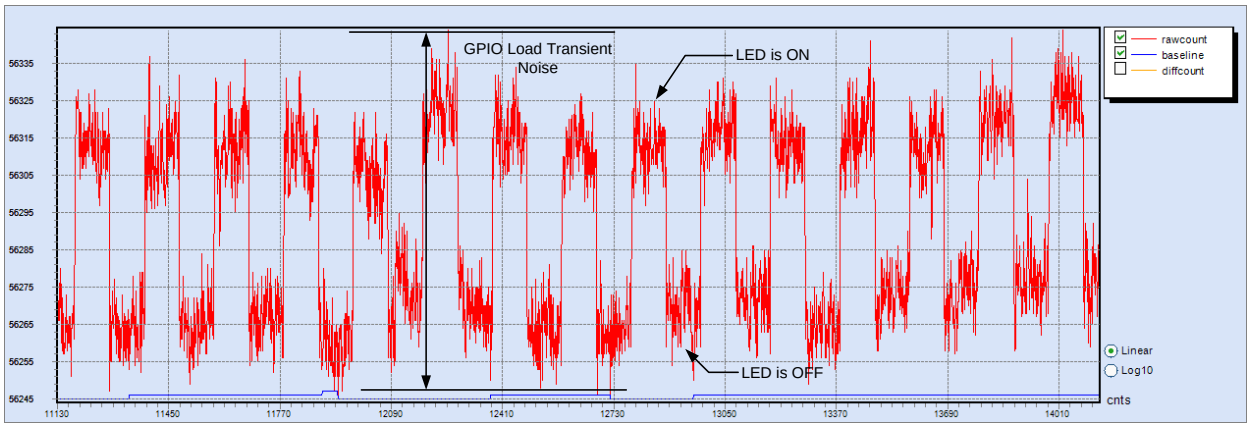
CapSense 控制器结构限制了奇偶端口引脚上的电流预算。奇数引脚指以奇数作为引脚号的端口引脚。对于 CapSense 控制器，如果奇数端口引脚的电流预算为 100 mA，那么所有奇数端口引脚的总电流不应该超过 100 mA。除了总电流预算限制外，CapSense 控制器数据手册中定义的每个端口引脚还有最大电流限制。

所有的 CapSense 控制器提供了适用于高灌电流和源电流的端口引脚。使用来自端口引脚的高灌电流或源电流时，应使用距离器件接地引脚最近的端口，以最小化噪声。

## 5.7 GPIO 负载瞬态

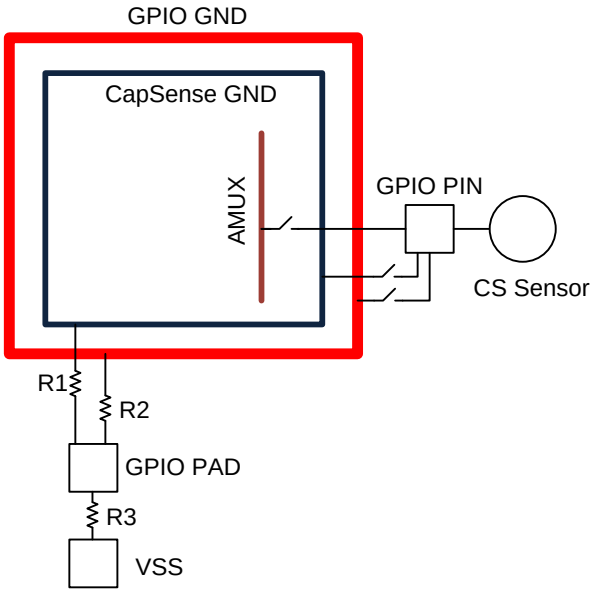
当 GPIO 通过将端口引脚驱动为强驱动低电平，并将大电流（> 10 mA）输入到该芯片的接地面时，会有噪声被引入到 CapSense 系统内。通过 GPIO 流入地面的电流量的瞬间改变被视为 GPIO 负载瞬态。因 GPIO 负载瞬态引入 CapSense 系统的噪声被称为 GPIO 负载瞬态噪声，如图 5-3 所示。本节介绍了如何使用硬件技术降低噪声以及如何使用固件技术补偿噪声。

图 5-3. CapSense 系统中的 GPIO 负载瞬态噪声




通过 GPIO 引脚输入电流时，由于非零接合线电阻 R3 的存在，CapSense 接地面（GPIO PAD）上的电压为非零值。由于非零接地电位的存在，因此当 LED 输入电流时，传感器将不会完全放电；这样会使传感器原始计数递增。

图 5-4. CY8C20xx7/S 中的接地结构




**注意：**R1、R2 和 R3 均是接合线电阻

对于稳健的 CapSense 设计，最大的 GPIO 瞬态噪声应该小于手指触摸信号的 30%。当 GPIO 状态从无电流状态（例如，所有 LED 关闭）到最大电流状态（例如，所有 LED 打开）时，最大的噪声将发生在 CapSense 系统内。

GPIO 负载瞬态噪声与传感器扫描分辨率同步增加。带有高寄生电容的 CapSense 传感器或接近感应传感器需要更高的传感器扫描分辨率，以使信噪比 > 5:1。在这些系统中，GPIO 负载瞬态的影响更明显。在某些情况下，由 GPIO 负载瞬态造成的噪声会比由手指触摸造成的信号更高，从而导致传感器的误触发。以下内容介绍了如何降低 GPIO 负载瞬态噪声。

## 5.7.1 降低 GPIO 负载瞬态噪声的硬件指南

- 降低传感器 $C_P$

  传感器 $C_P$ 确定传感器的扫描分辨率参数。为了保证信噪比 > 5:1，当 $C_P$ 越大，分辨率参数越高。将分辨率参数设置为高会使 GPIO 负载瞬态噪声的振幅增加。因此，建议按照 CapSense 入门设计指南所介绍的布局指南尽量减少传感器 $C_P$。

- 降低 LED 灌电流

  GPIO 负载瞬态噪声与 LED 灌电流直接成正比。建议将 LED 的灌电流保持为器件数据手册中所指定的范围内。如果 GPIO 必须输入超过数据手册中所指定的最大值的电流，则应该使用外部晶体管或驱动器 IC。

- 为 LED 选择合适引脚

  所有的 CapSense 控制器提供了可以使用高灌电流和源电流的端口引脚。使用来自端口引脚的高灌电流或源电流时，应该使用表 5-5 中所推荐的端口。

## 5.7.2 补偿 GPIO 负载瞬态噪声的固件指南

为了防止由 GPIO 负载瞬态噪声导致的传感器误触发，可以使用符合规则的算法更新传感器基准线。下面介绍了补偿基准线的一种方法。

图 5-5 显示了由 GPIO 负载瞬态造成的误触发的情况。

1. 在第一个事件内，传感器上没有手指触摸而且 LED 为关闭状态。
2. 在第二个事件内，传感器上发生手指触摸而且原始计数的移位大于手指阈值。
3. 由于原始计数的移位大于手指阈值，因此在第三个事件内，LED 被打开。
4. 当 LED 打开时，由于 GPIO 负载瞬态噪声的存在，因此原始计数将再次移位。
5. 在第四个事件内，如果移除手指，由于 GPIO 负载瞬态噪声导致的原始计数的移位存在，因此原始计数将不返回初始值。如果该移位值大于手指阈值，LED 将永远保持为打开状态，造成传感器的误触发。

为了防止传感器和 LED 永远保持打开状态，需要补偿传感器基准线，以下步骤介绍了该内容。

图 5-5. 基准线未被补偿时的 CapSense 传感器变量

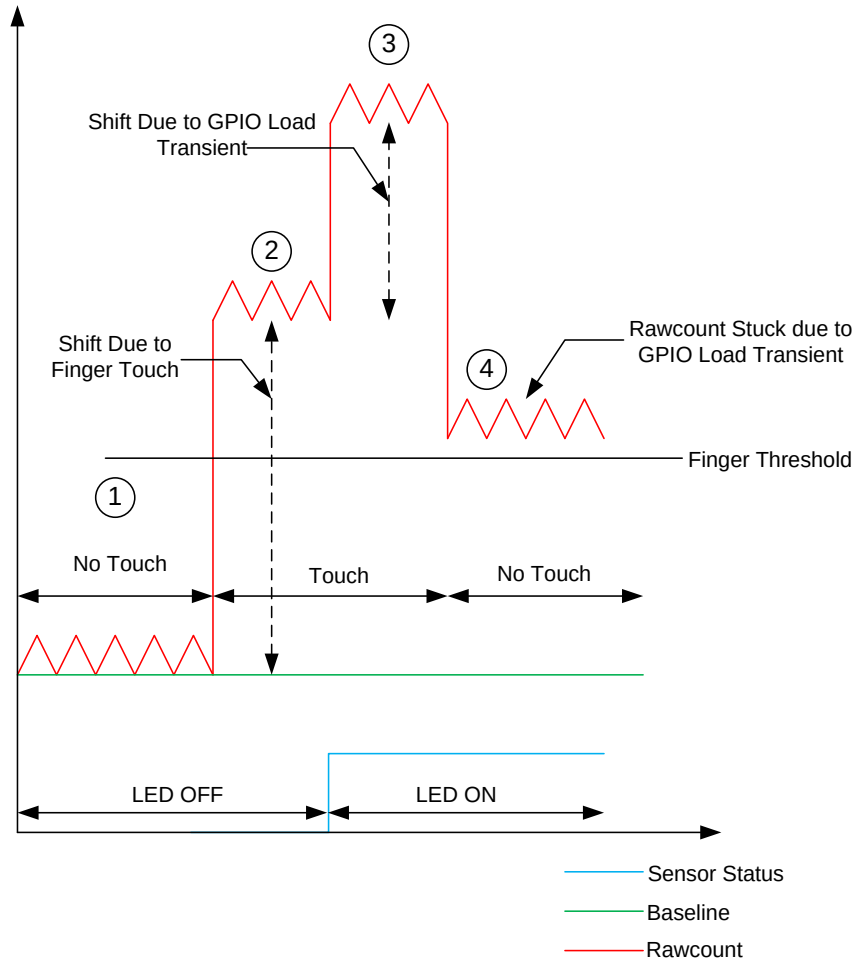
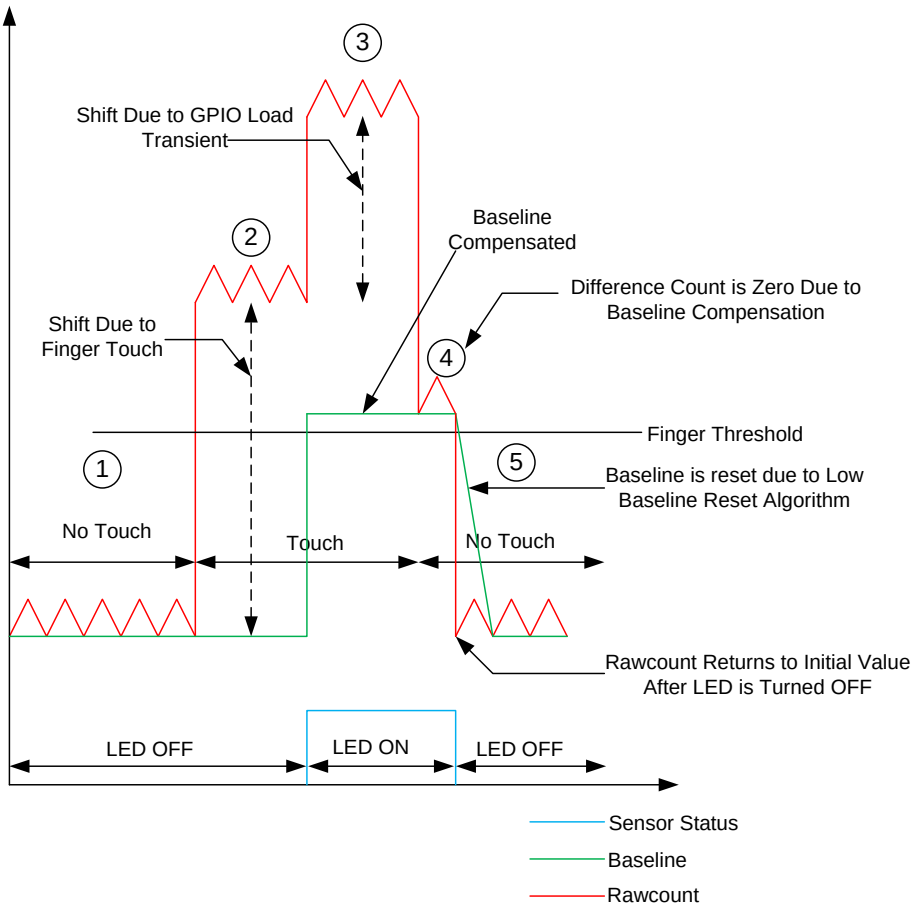



图 5-6 显示了通过补偿传感器基准线消除了误触发的情况。

1. 在第一个事件内，传感器上没有手指触摸而且 LED 为关闭状态。
2. 在第二个事件内，传感器上发生手指触摸而且原始计数的移位（计数差值（信号值））大于手指阈值。
3. 由于计数差值（信号值）的移位大于手指阈值，因此在第三个事件内，LED 被打开。
4. 当打开 LED 时，将计算由 GPIO 负载瞬态造成的噪声。即为：噪声 = 原始计数（LED 打开时） - 原始计数（LED 关闭时）

   由 GPIO 负载瞬态造成的噪声计数被添加到基准线内，因此，当移除手指触摸时，计数差值（信号值）将为 0 而且 LED 被关闭。
5. LED 关闭后，原始计数将返回初始值而且基准线通过低基准线复位算法复位。

图 5-6. 基准线被补偿时的 CapSense 传感器变量




## 5.8 PCB 布局指南

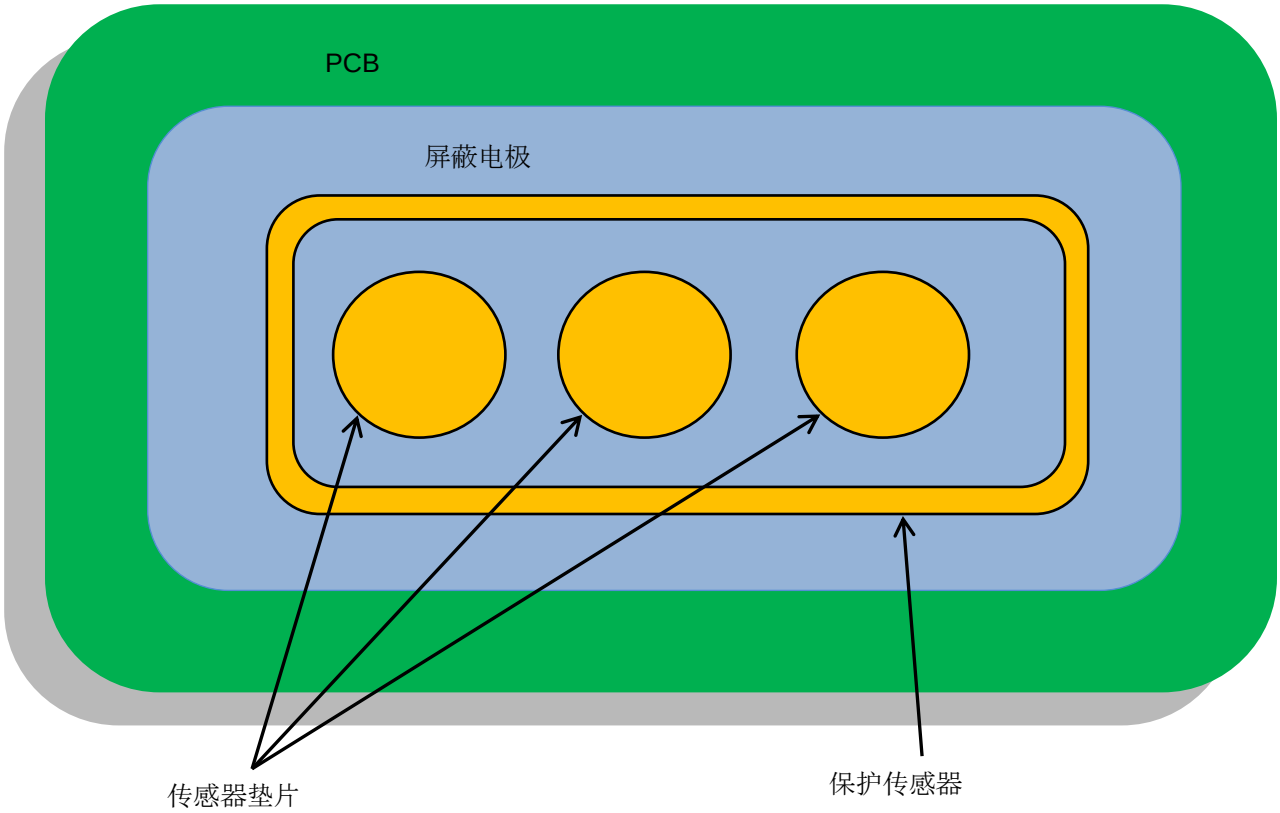
CapSense 入门中提供了有关 PCB 布局的详细指南。

# 6. 防水设计的注意事项

某些 CapSense 电容式触摸感应应用需要在带有水份环境进行可靠的工作。白色家电、汽车应用和工业应用是需要在存在水、冰、湿度变化或其他液体的环境下工作的系统。对于这样的应用环境而言，屏蔽电极和保护传感器可以提供强健的触摸感应系统。

## 6.1 屏蔽电极和保护传感器

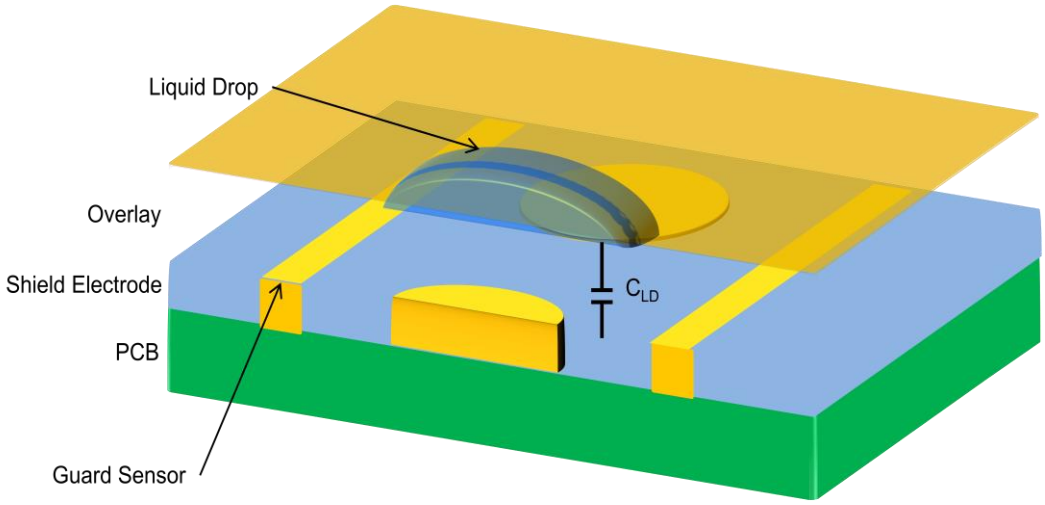
图 6-1. 使用屏蔽电极和保护传感器的 PCB 布局




### 6.1.1 屏蔽电极

屏蔽电极防止 CapSense 按键传感器检测由水滴引起的误触摸。当水滴落到覆盖层表面上时， 屏蔽电极和传感器板之间的耦合电容增加了 $C_{LD}$，如图 6-2 所示。

图 6-2. 存在水滴时的电容测量




$C_{WD}$ — 水滴与屏蔽电极之间的电容

屏蔽电极的目的是在触摸传感器周围建立电场，用来消弱水滴产生的影响。屏蔽电极的工作方式是对照屏蔽电极上的触摸传感器的电压。

请遵循以下准则以确保正确的屏蔽操作：

- 原理图
- 布局
- 固件开发

### 6.1.1.1 *原理图*

选择正确的引脚以驱动屏蔽电极输出信号。应该使用以下引脚来驱动屏蔽电极输出信号。

- 端口引脚：P0[0]、P1[2]、P0[2]、P2[2]或 P2[4]

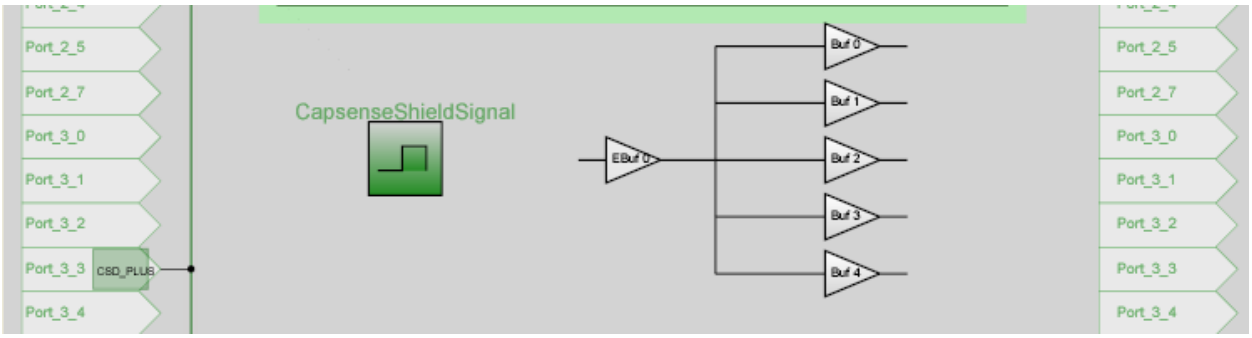
### 6.1.1.2 *布局*

遵循 CapSense 入门中提供的布局指南。

### 6.1.1.3 *固件开发*

通过使用图 6-3 中的屏蔽驱动 GUI，从而最小化固件开发。

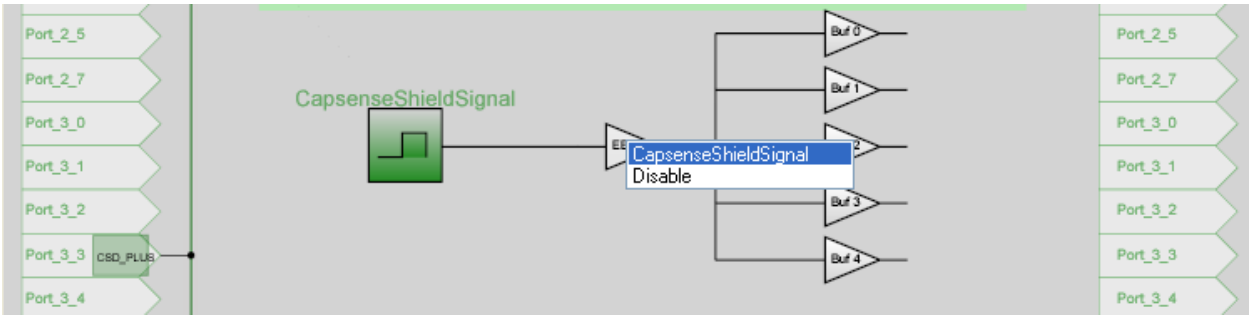
图 6-3.驱动屏蔽 GUI（屏蔽无效）




**第一步：**在 GUI 中使能 CapSense 屏蔽信号。

点击“Ebuf0”，即 EnableShieldDriver 开关（如图 6-4 所示）；默认情况下，“Ebuf0”选项被设置为“Disable”。选择 **CapsenseShieldSignal** 以使能屏蔽驱动器。

图 6-4. 在 GUI 中使能 CapsenseShieldSignal



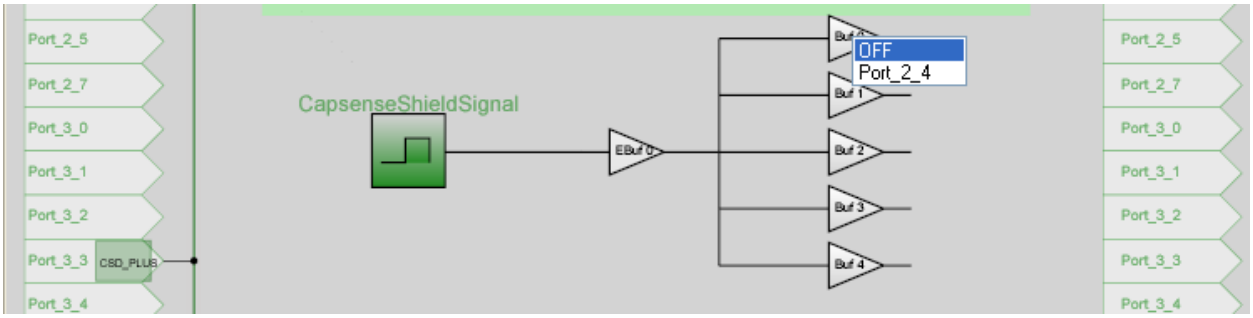

**第二步：**将屏蔽电极输出路由到屏蔽引脚。

共有五个屏蔽驱动缓冲器，分别被标示为 Buf 0 到 Buf 4，缓冲器的映射情况如下表所示：

| 屏蔽缓冲器 | 输出端口引脚 |
|---|---|
| Buf 0 | Port_2_4 |
| Buf 1 | Port_2_2 |
| Buf 2 | Port_0_2 |
| Buf 3 | Port_0_0 |
| Buf 4 | Port_1_2 |

屏蔽驱动缓冲器的默认状态是 OFF；请选择需要使用的屏蔽驱动缓冲器。如图 6-5 所示，Buf 0 已被使能，因此屏蔽信号在 Port_2_4 上被驱动。

图 6-5. 输出选择原理图



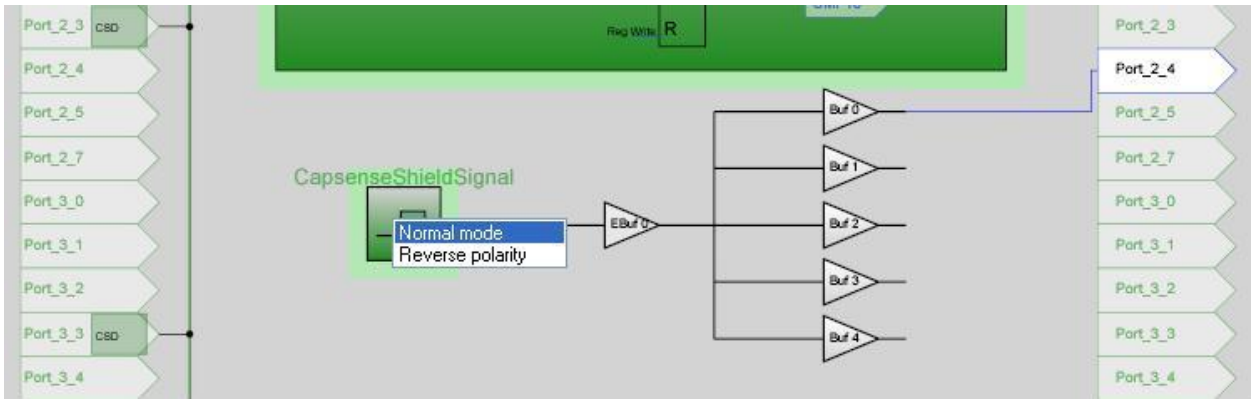

**第三步：**选择屏蔽信号的极性。

共有两个极性选项，如图 6-8 所示：

- 普通模式（默认模式）：屏蔽驱动器信号与传感器扫描信号的相位相同。
- 反向极性：屏蔽驱动器信号与传感器扫描信号的相位间相差 180°。

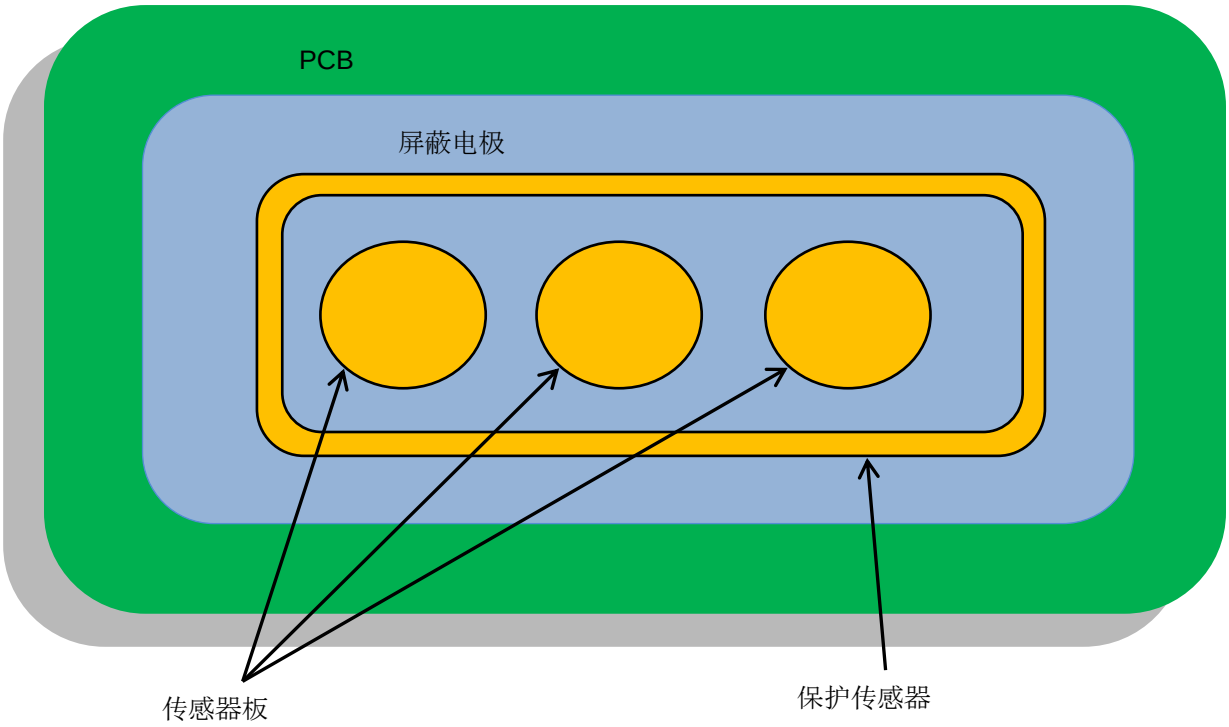
普通模式为防水设计提供了最好的性能。

图 6-6. 极性选择原理图



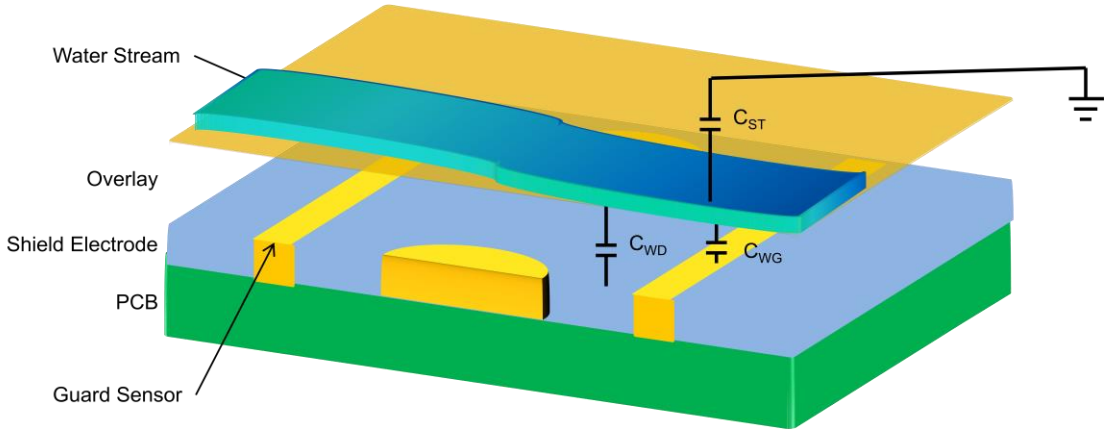

## 6.1.2 保护传感器

图 6-7. 带有屏蔽和保护传感器的 PCB




保护传感器是一条铜走线，如图 6-9 所示，该走线环绕 PCB 上的所有传感器，用来检测是否出现了连续水流或大量液体溢出。当在感应表面上存在水流或大量液体时，系统中的电容将增加 $C_{ST}$，如图 6-8 所示。此电容可能比 $C_{LD}$ 大数倍。正因为如此，屏蔽电极的影响完全被消除，传感器测量的原始计数将与手指触摸值相同，甚至更高。在这种情况下，保护传感器将有所帮助；当检测水流时，该传感器将防止其他传感器触发。

图 6-8. 存在大量液体或水流时的电容测量



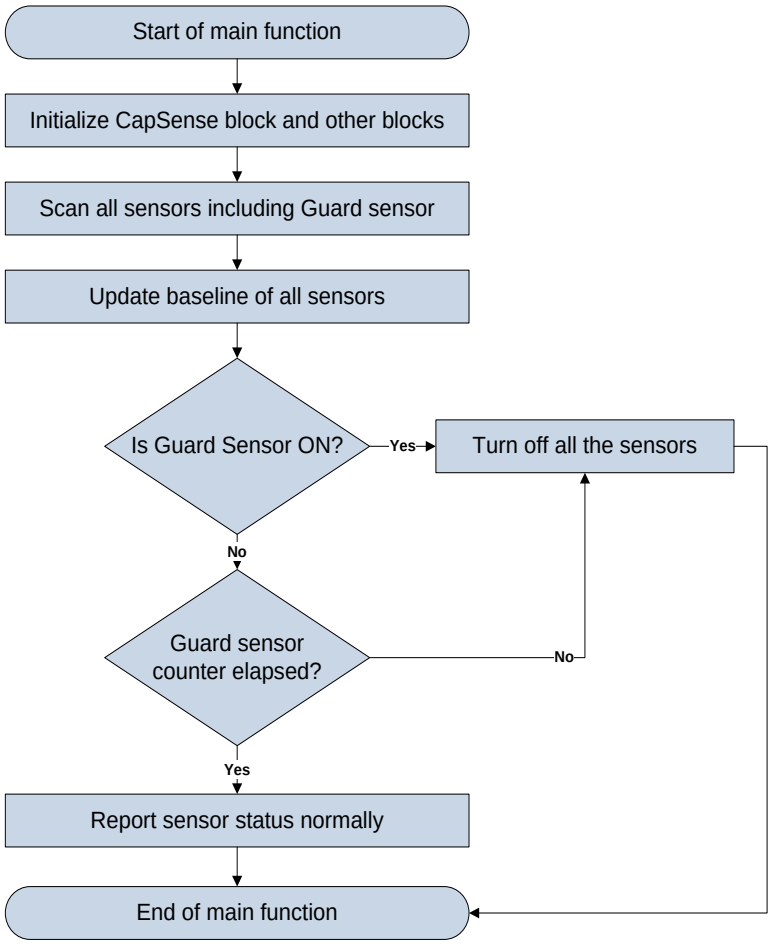

- $C_{LD}$ — 水流/液体与屏蔽电极之间的电容
- $C_{ST}$ — 水流/液体与系统接地之间的电容
- $C_{WG}$ — 水流/液体与保护传感器之间的电容

保护传感器应在固件中实现。实施保护传感器需要一个 CapSense 引脚和一个计数器（硬件/软件）。

当水流进入触摸板时，保护传感器将检测到这个事件，然后禁用触摸传感器的处理逻辑。附加保护传感器的“死”时间阻止传感器被过早解锁。水流消失时，保护计数器会在短时间暂时抑制触摸处理系统（保护传感器计数器）。这样消除了遗留在电路板上的水流触发的假触摸检测。

图 6-9 显示了如何在固件中实现保护传感器的流程图。

图 6-9. 实现保护传感器的流程图




## 6.2 设计建议

以下的系统级和 PCB 布局建议适用于暴露于水份的 CapSense 系统。

- 屏蔽电极铜网格填充建议：
  - 顶层 — 走线宽度为 7 mil，栅格为 45 mil（填充 25%）
  - 下层 — 走线宽度为 7 mil，栅格为 70 mil（填充 17%）
  - 按键之间的屏蔽电极宽度至少应为 10 mm
- 传感器表面垂直放置，或与平面成一个角度，这样，水流自然滑离传感器，而不会积存大量水滴。
- 使用防水和不吸水的覆盖层材料。这样可以最小化器件屏幕上的水纹和薄膜。这特别适用于具有较高导电性的水，例如，海水。
- 如果应用在存在连续水流的条件下运行，必须使用保护性传感器。如果器件仅受雨、薄雾或潮湿条件的影响，则不需要保护传感器。

# 7. 接近感应设计的注意事项

接近感应传感器能够在手指或其他导电性物体接触电容式触摸表面前检测到它们的存在。假设在黑暗中，一只手伸出以打开汽车音频系统。接近检测功能在手指接近时能够使系统发亮。例如，当用户的手靠近时，音响系统按键的背光灯 LED 将点亮。

## 7.1 接近感应传感器类型

### 7.1.1 按键

具有较大 $C_P$ 和较小计数差值（信号值）的按键可以作为接近感应传感器使用。实施为按键的接近感应传感器的灵敏度相当高，远远超过了常规电容式触摸感应按键的灵敏度。

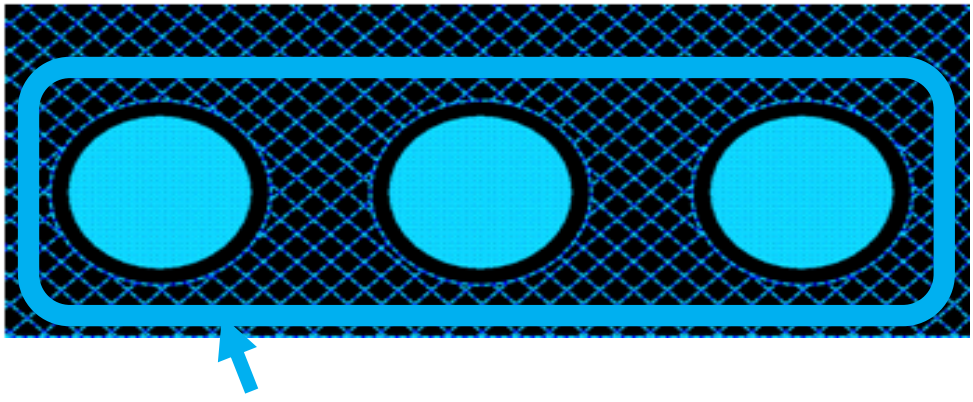
### 7.1.2 导线

一根导线段也可以用作良好的接近感应传感器。由于手指检测取决于随电场变化而变化的电容，影响到导线周围的电场的任何杂散电容或物体都会影响接近感应传感器的感应范围。由于制造成本和复杂性较高，所以使用导线传感器对于批量生产而言并不是理想的解决方案。

### 7.1.3 PCB 走线

一根较长的 PCB 走线可以构成一个接近感应传感器。走线可能是直线型，或环绕系统的用户接口周边，如图 7-1 所示。这种方法适用于批量生产，但灵敏性不如导线传感器。

图 7-1. 使用 PC 走线的接近感应传感器




### 7.1.4 传感器机械连接

实现接近感应传感器的另一种方法是机械性连接多个传感器。实施方法是将多个传感器组合到一个大型传感器中，使用固件将传感器连接到 PSoC Designer 内部模块复用器总线上。使用这种方法时，请注意不要超过 $C_P$ 设计限制。

## 7.2 设计建议

- 使用屏蔽电极有效延长接近感应传感器的检测距离。当接近感应传感器周围存在金属物体时，这种方法特别有用。
- 导线传感器使屏蔽电极的作用得到更大的发挥，因为它可以较大程度地远离屏蔽电极。
- 避免接近感应传感器内部范围较大且较坚实的接地填充区域的存在，因为这样会使灵敏度降低。
- 扫描速度越慢，所有位置的灵敏度越高。对于接近感应传感器，扫描速度应该非常低。

# 8. 低功耗设计的注意事项

功耗是微控制器设计中面临的重要问题。在降低 CapSense 控制器使用的平均电流的多种技术当中，睡眠模式的使用最为普遍。在不需要执行任何功能时，CapSense 控制器会进入睡眠模式，这与手机背光灯在空闲期间变暗的情况相同。这样可以降低器件消耗的平均电流，进而满足所有电池应用的节能要求。

CapSense 控制器向 CPU_SCR0 寄存器（位 3）中的 SLEEP（睡眠）位写入‘1’来进入睡眠模式。通过调用 M8C_Sleep 宏，可以实现该功能。

在睡眠模式下：

- 中央 CPU 被停止
- IMO 被禁用
- 带隙电压参考被断电
- 闪存模式被禁用

保持工作的唯一电路是电源电压监视器和 32 kHz 的内部振荡器。

睡眠模式以外的其他节省功耗技术：

- $I^2C$ 睡眠模式（请参见进入 I2C 睡眠模式）
- 禁用 CapSense（PSoC）模拟模块参考
- 禁用 CT 和 SC 模块
- 禁用 CapSense（PSoC）模拟输出缓冲区
- 将驱动模式设置为模拟高阻态

睡眠模式对设计造成负面影响。若在使用时不注意，可能会造成无法预见的操作。如有必要，必须将 PSoC 从睡眠模式中正确唤醒，用户要认识到器件处于睡眠时容许的额外处理。

## 8.1 其他节能技术

除睡眠模式之外，所有节能技术均是基于应用的技术。它们当中的某些技术可能产生不良结果。以下章节对各种技术进行了详细讨论。

### 8.1.1 将驱动模式设置为模拟高阻

CapSense 控制器驱动模式的状态可能影响功耗程度。您只能改变不会对系统造成负面影响引脚上的驱动模式。需要按准确顺序进行更改，这样可以避免产生线路故障。该顺序由当前的驱动模式和端口数据寄存器状态决定。对于 CapSense 控制器驱动模式结构，在高阻态或强驱动模式之间切换时，引脚必须暂时处于电阻上拉或电阻下拉驱动模式。临时性驱动模式与引脚上的上一个值相反。因此，如果引脚以前的驱动模式为电阻上拉，那么当前的模式应为电阻下拉。这就确保引脚的驱动模式不是电阻驱动模式，排除一切可能产生的故障。

进入睡眠模式之前，在软件中手动设置驱动模式。共有三个寄存器，分别是 PRTxDM0、PRTxDM1 和 PRTxDM2，用于控制驱动模式。每个寄存器向一个引脚分配一个位。因此，要想更改每个引脚的驱动模式，便需要对三个寄存器进行三次写操作。但由于整个端口通过三个相同的寄存器写操作更改，因此该操作很容易实现。HI-Z 模拟的正确位模式为 110b。使用以下代码，通过调至电阻下拉模式，将端口 0 从强驱动模式转换为。

```
PRT0DM0 = 0x00; // low bits
PRT0DM1 = 0xff; // med bits
PRT0DM2 = 0xff; //high bits
```

### 8.1.2 全部放在一起

下列代码是 28 引脚器件准备进入睡眠模式的典型序列的例子。在该序列中，中断被禁用，模拟电路被关闭，所有驱动模式均被设置为模拟高阻态，然后重新使能中断。

```
void PSoC_Sleep(void)
{
    M8C_DisableGInt;
    PRT0DM0 = 0x00; // port 0 drives
    PRT0DM1 = 0xff;
    PRT0DM2 = 0xff;
    PRT1DM0 = 0x00; // port 1 drives
    PRT1DM1 = 0xff;
    PRT1DM2 = 0xff;
    PRT2DM0 = 0x00; // port 2 drives
    PRT2DM1 = 0xff;
    PRT2DM2 = 0xff;
    M8C_EnableGInt;
    M8C_Sleep;
}
```

### 8.1.3 睡眠模式下建议的 I²C 从设备操作

在睡眠模式下使用 I²C 时，为了防止 I²C 总线的锁定状态，或损坏数据操作的发生，需要遵循一定的执行指南。

#### *8.1.3.1 进入 I²C 睡眠模式*

I²C 从设备在进入睡眠模式之前，通常需处于 FORCE_NACK 模式。为了保证准确进入睡眠模式，需要按以下步骤进行操作：

- 通过 I2C_BP_EZ_CFG 寄存器中的 CLK_STRETCH_EN 位选择操作模式（时钟延长或睡眠到唤醒期间保持 NACK 的模式）。
- 设置 I2C_XCFG 寄存器中的 FORCE_NACK 位
- 轮询 I2C_XSTAT.READY_TO_NACK 是否为逻辑‘1’
- 设置 SLP_CFG2 寄存器中的 I2C_ON 位
- 调用 M8C_Sleep 宏（该操作会设置 CPU_SCR0 寄存器中的 SLEEP 位（位 3））

**注意：**只有通过激活 SLP_CFG2 寄存器中的 I2C_ON 位使能提供给 I²C 模块的电源时，才能保证 32 字节的 I²C 缓冲器中的数据在睡眠和深度睡眠模式下得以保留。

### 8.1.4 睡眠模式复杂性

CapSense 控制器可以通过复位或中断退出睡眠模式。CapSense 控制器有三种复位方式：外部复位、看门狗复位和上电复位。任意一种复位方式均可使 CapSense 控制器退出睡眠模式。取消确认复位后，CapSense 控制器将从 *Boot.asm* 开始执行代码。用于唤醒 CapSense 控制器的可用中断有：睡眠定时器、低电压监视器、GPIO、模拟列以及异步。使用中断唤醒 CapSense 控制器，或在睡眠状态中执行数字通信时，都会增大睡眠模式的复杂性。这些注意事项将在后面的章节中进行讨论。

### 8.1.5 挂起中断

如果某个中断被挂起、被使能，并预定在写入到 CPU_SCR0 寄存器中的 SLEEP 位后发生，那么系统不会进入睡眠模式。指令仍然执行，但 CapSense 控制器并不设置 SLEEP 位。相反，中断得到服务，这样会导致 CapSense 控制器完全忽略睡眠指令。为了避免这种情况，准备进入睡眠模式时，应全部禁用中断并在写入 SLEEP 位之前重新使能中断。

### 8.1.6 全局中断使能

通过中断唤醒 CapSense 控制器时，无需使能“全局中断使能”寄存器（CPU_F）。通过中断从睡眠模式唤醒的唯一要求是使用 INT_MSKx 寄存器中的正确中断掩模，如下面示例所述。如果禁用全局中断，系统将不执行唤醒 CapSense 控制器的 ISR，但 CapSense 控制器仍然退出睡眠模式。

在这种情况下，您必须手动清除挂起中断，或使能全局中断以处理 ISR。在 INT_CLRx 寄存器中清除中断。

```
//Set Mask for GPIO Interrupts
M8C_EnableIntMask(INT_MSK0, INT_MSK0_GPIO)
// Clear Pending GPIO Interrupt
INT_CLR0 &= 0x20;
```

## 8.2 唤醒后执行序列

如果通过复位唤醒 CapSense 控制器，程序会从启动代码的前段开始执行。如果通过中断服务子程序唤醒 CapSense 控制器，首先执行的指令是紧随睡眠指令之后的指令。这是因为 CapSense 控制器完全进入睡眠模式之前，已预先提取该指令。因此，如果禁用全局中断，指令执行将从进入睡眠前的中断位置继续。

### 8.2.1 使能 PLL 模式

如果使能 PLL 模式，则 CPU 在进入睡眠模式前，频率必须降至 3 MHz（最小值）。这是因为 PLL 在 CapSense 控制器唤醒并重新启用后尝试重新锁定时总会出现过冲。此外，为了确保正确操作，您应在唤醒后等待 10 ms，然后再开始让 CPU 正常工作。这表示软件必须能够在 3 MHz 的频率下运行，才能使用睡眠模式和 PLL。对 OSC_CR0 寄存器进行简单的写操作可以降低 CPU 的速度。然而，该寄存器仅设置 SYSCLK 分频器，这意味着 CPU 速度将由于器件系列的不同 SYSCLK 而有所不同。一般情况下，SYSCLK 为 24 MHz

```
OSC_CR0 &= 0xf8; // CPU = 3 Mhz IMO = 24 Mhz
```

### 8.2.2 执行全局中断使能

避免在写入 SLEEP 位的指令边界上使能中断。如果在中断（reti）指令后执行睡眠指令，则会忽略固件的所有睡眠准备。要避免发生这种情况，请在准备进入睡眠前暂时禁用中断，然后重新使能中断。由于全局中断指令的时序要求，中断不会发生在下一指令执行时，也就是设置睡眠位时。

### 8.2.3 睡眠模式下建议的 I²C 从设备操作

在睡眠模式下使用 I²C 时，为了防止 I²C 总线的锁定状态，或损坏数据操作的发生，需要遵循一定的执行指南。

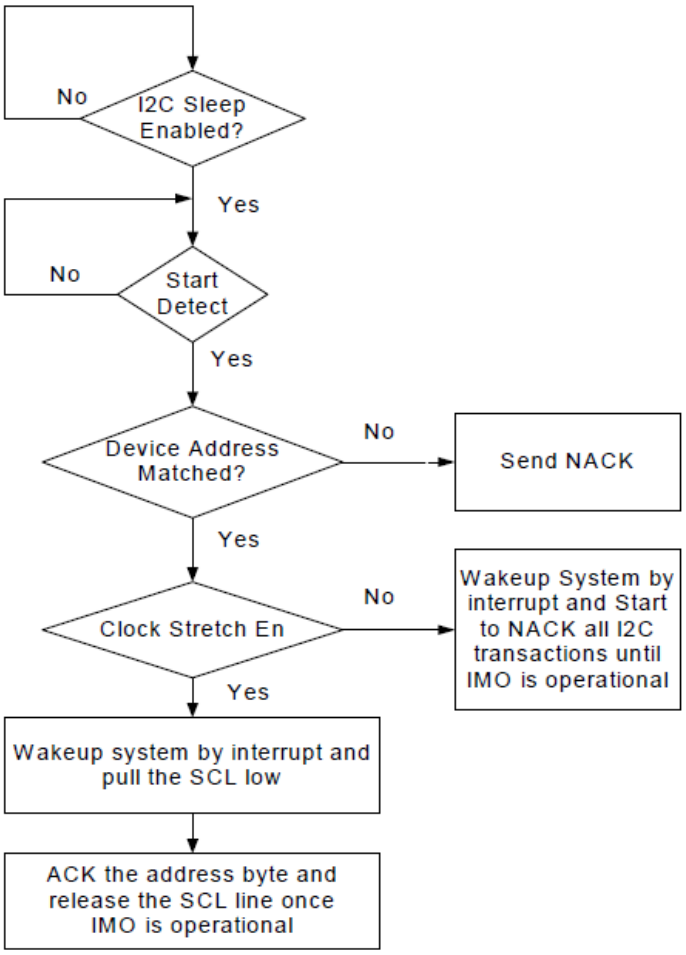
#### *8.2.3.1 通过硬件地址匹配从 I²C 睡眠模式唤醒*

要想通过 I²C 使能唤醒，需要设置硬件地址 EN 位，以保证 I²C 从设备模块仅在地址匹配时唤醒系统。系统处于睡眠模式时，I²C 模块对 I²C 总线上的数据操作响应。但是，系统时钟在该模式下已被停止。

因此，I²C 模块缺少了用于对 I²C 总线上的数据操作响应时所需的系统时钟。输入 SCL 时钟由此成为了该模块所需的时钟。

地址匹配时，该模块的响应由 CLK_STRETCH_EN 的设置决定。该位处于高电平时，SCL 将为低电平，直到 IMO 运行为止。该位为低电平时，从设备将否认向其发送的所有 I²C 数据操作，直到 CPU 唤醒并设置 ACK 位为止。时钟延长模式被禁用时，唤醒 IRQ 后，器件将忽略所有其他启动条件，直到 IMO 运行为止。这段时间内发生的所有数据操作都会收到一个 NACK。图 8-1 显示了使用 I²C 的唤醒流程。

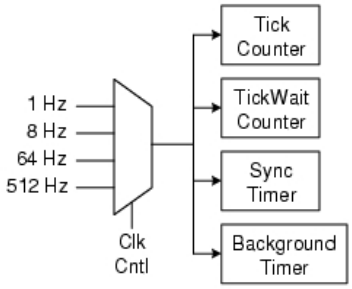
图 8-1. 唤醒序列




## 8.2.4 睡眠定时器

CapSense 控制器提供了睡眠定时器和睡眠定时器用户模块。当 CapSense 控制器处于睡眠模式时，两者将得以使用，并执行类似功能。实际的睡眠定时器由永远不关闭的内部低速振荡器控制。定时器在 1 Hz、8 Hz、64 Hz 和 512 Hz 等可选频率下生成一个中断。定期唤醒 CapSense 控制器有助于执行某些处理或检查操作。例如，定期唤醒以扫描传感器。睡眠定时器用户模块使用睡眠定时器来执行其他功能。这些功能包括后台钟表计数器（用来生成定期中断）、程序环路延迟功能、可变倒计时计数器和控制回路时间的回路控制器。该功能的简化框图如图 8-2.所示。

图 8-2.睡眠定时器用户模块框图



# 9. 资源

## 9.1 网站

访问赛普拉斯的 CapSense 控制器网站可获取本节中讨论的所有参考材料。

如需查看 CapSense CY8C20xx7/S 系列器件丰富的技术资源，请查看 CY8C20xx7/S 网页。

## 9.2 数据手册

CapSense CY8C20xx7/S 器件系列的数据手册可以从 www.cypress.com 上获取。

- CY8C20xx7/S 数据手册

## 9.3 技术参考手册

赛普拉斯创建了下列技术参考手册，用于对 CapSense 控制器功能的信息（包括顶级架构图、寄存器和时序图）进行快速而简易的访问。

- CY8C20xx7/S 技术参考手册（TRM）

# 9.4 开发套件

## 9.4.1 CY8C20xx7/S QuietZone 入门套件

CY8C20xx7/S QuietZone 入门套件（如图 9-1 所示）是一个基于 CY8C20247S 16-QFN 封装的 0.75 英寸大小的方形 PCB。

图 9-1. CY8C20xx7/S QuietZone 入门套件




CY8C20xx7/S QuietZone 入门套件硬件具有以下特性：

- 十二个电容式感应输入
- 一个 5 引脚的插座（HD1），用于快速连接至 MiniProg1/3
- 一个驱动屏蔽输出
- 两个 LED，用于视觉反馈

您可以从我们的生产模块合作伙伴 ArtaFlex 的以下链接下载该套件：www.artaflexmodules.com/quietzone

## 9.4.2 通用 CapSense 控制器套件

CY8C20xx7/S 系列不具有片上调试（OCD）功能；如果需要调校平台，需要提供 CY8C20xx6A 通用 CapSense 控制器套件。通用 CapSense 控制器套件的特性包括预定义控制电路和即插即用硬件，易于进行原型设计和调校。该套件还包含编程硬件和 I²C 至 USB 桥接器硬件，用于调校和数据采集。

- CY3280-20xx6A 通用 CapSense 控制器

## 9.4.3 通用 CapSense 模块板

### *9.4.3.1 简单按键模块板*

CY3280-BSM 简单按键模块由 10 个 CapSense 按键和 10 个 LED 组成。该模块可以连接至任何 CY3280 通用 CapSense 控制器电路板

### *9.4.3.2 矩阵按键模块板*

CY3280-BMM 阵列按键模块由 8 个 LED 和 8 个 CapSense 传感器组成（以 4×4 阵列格式组织，从而构成 16 个物理按键）。该模块可连接至任何 CY3280 通用 CapSense 控制器电路板。

## 9.5 样本电路板文件

赛普拉斯提供原理图和电路板样本文件，您可将其用作参考以快速完成您的 PCB 设计流程。

- CY8C20xx7/S 上带有 I²C 插座的按键设计
- CY8C20xx7/S 上带有 I²C 插座的按键和滑条设计

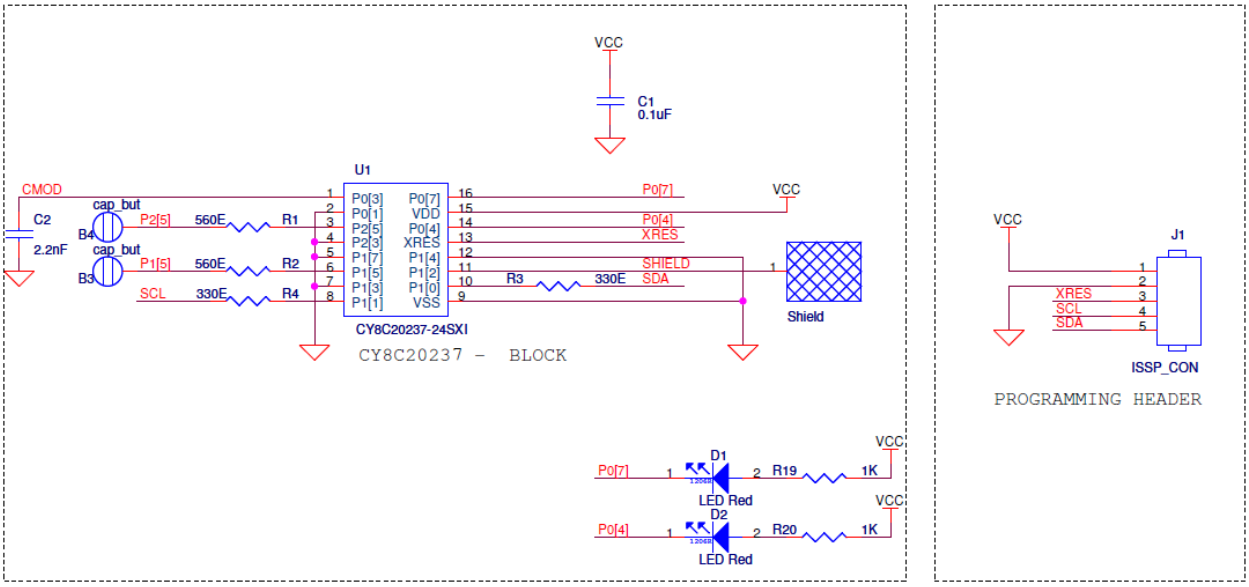
**注意：**电路板文件（原理图、布局与Gerber文件）将置于本文档的登陆页面。

图 9-2 和图 9-3 显示了电路板原理图。

以下原理图用于支持：

- 两个 CapSense 传感器。这些传感器分别被分配给 CY8C20237-24SXI 器件的引脚 P2[5]和 P1[5]。
- 两个 GPIO 分别被连接到 CY8C20237-24SXI 器件的引脚 P0[7]和 P0[4]上，用于驱动 LED D1 和 D2。
- 屏蔽电极被连接到引脚 P1[2]上，用于驱动屏蔽平台。
- CY8C20237-24SXI 的编程线和 I²C SDA/SCL 信号将被连接到插座 J1 上。

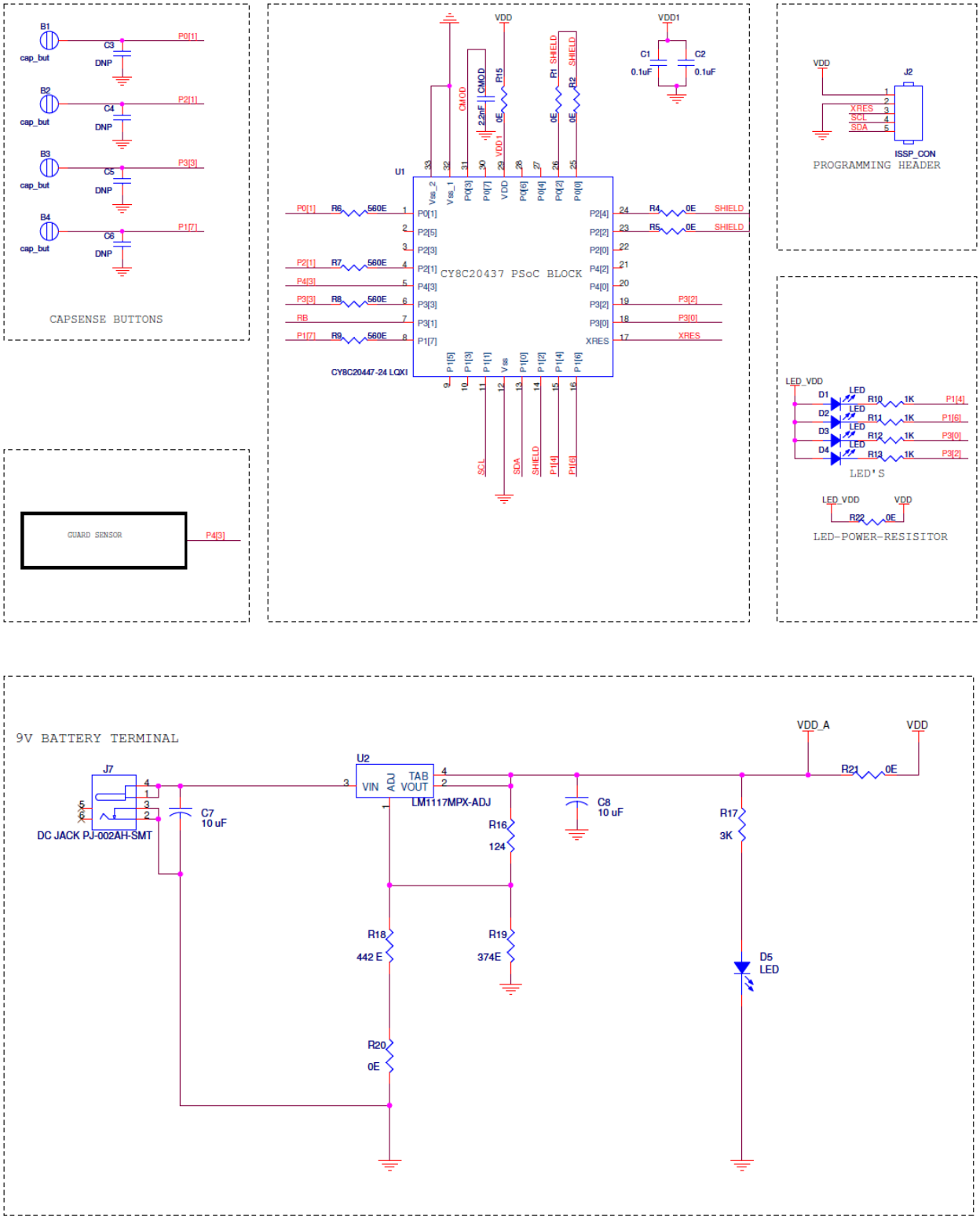
图 9-2. CY8C20xx7/S 上带有 I²C 插座的按键设计 — 电路板原理图




以下原理图仅支持下列条件：

- 4 个 CapSense 传感器。这些传感器被分别分配到 CY8C20437-24LQXI 器件的引脚 P0[1]、P2[1]、P3[3]和 P1[7]。
- 保护传感器被连接到 CY8C20437-24LQXI 器件的引脚 P4[3]上。
- 四个 GPIO 分别被连接到 CY8C20437-24LQXI 器件的引脚 P1[4]、P1[6]、P3[0]和 P3[2]上，用于驱动 LED D1、D2、D3 和 D4。
- 通过编程插座 J2 对 CY8C20437-24LQXI 进行编程。
- 可将 9 V 直流输入连接到插座 J7 上。

图 9-3. CY8C20437 上带有 I²C 插座的防水按键设计



## 9.6 PSoC Programmer

PSoC Programmer 是用于编程 PSoC 器件的灵活度且集成度较高的编程应用。它可与 PSoC Designer 和 PSoC Creator™搭配使用，进行 PSoC 器件的任何设计编程。

PSoC Programmer 包含带有 API 的硬件层，可用编程器与桥接器件设计特定的应用。COM 指导文档中包含了 PSoC Programmer 硬件层的详细描述及以下所有语言的示例代码：C#、C、Perl 和 Python。

## 9.7 CapSense 数据查看工具

在 CapSense 设计过程中，您可能要多次监控 CapSense 相关数据（原始计数、基准线、计数差值（信号值）等），以实现调校和调试目的。应用笔记 AN2397 – CapSense 数据查看工具为您提供了有助于识别和使用正确的 CapSense 数据查看和记录工具的信息。

## 9.8 PSoC Designer

赛普拉斯具备独有的集成设计环境（IDE）— PSoC Designer。使用 PSoC Designer，您可以配置模拟和数字模块、开发固件和调校设计。在拖放式设计环境中使用全特性模拟和数字功能库（包括 CapSense）来开发您的应用。PSoC Designer 带有内置 C 语言编译器和嵌入式编程器。现已推出用于复杂设计的专业版编译器。

## 9.9 代码示例

赛普拉斯提供了大量代码示例，这样您能够轻松快速进行设计。

- CapSense 控制器代码示例设计指南

## 9.10 设计支持

赛普拉斯具有各种设计支持渠道，以确保 CapSense 解决方案成功。

- 知识库文章 — 参见产品系列的技术文章或搜索各种 CapSense 相关主题。
- CapSense 应用笔记 — 参见本文档中说明信息所涉及的广泛应用笔记。
- 白皮书 — 了解电容式触摸接口的高级主题。
- 赛普拉斯开发社区 — 与赛普拉斯技术社区联系并交换信息。
- CapSense 产品选择器指南 — 参见赛普拉斯 CapSense 产品系列的全部产品。
- 视频资料库 — 通过教程视频快速掌握相关知识
- 质量和可靠性 — 赛普拉斯承诺满足客户的要求。在我们的质量网站上，可以找到可靠性和产品资质报告。
- 技术支持 — 在线提供一流的技术支持。

# 术语表

**AMUXBUS**

指的是 PSoC 中的模拟复用器总线，通过它可将 I/O 引脚连接至多个内部模拟信号。

**SmartSense™ 自动调校**

设计阶段结束后，CapSense 算法自动设置各个感应参数以得到最佳性能，然后连续补偿由于系统、生产过程和环境不同引起的变化。

**基准线**

指的是从固件算法得到的数值。当传感器上没有手指触摸时，该算法将估计原始计数的值。基准线对原始计数突变的灵敏度较低，另外它还为计数差值提供了参考点。

**按键或按键 widget**

指的是带有相关传感器的 widget，它会报告传感器的活动或非活动状态（即仅两种状态）。例如，它可以检测到传感器上是否有手指触摸。

**计数差值（信号值）**

指的是原始计数与基准线间的差值。如果该差值为负，或如果它低于噪声阈值，则计数差值总是被设置为‘0’。

**电容式传感器**

导体和基板（如印刷电路板（PCB）上的铜质按键）会对触摸事件或接近电容变化物体作出反应。

**CapSense®**

赛普拉斯的触摸感应用户界面的解决方案。这是行业排名第一的解决方案，销量是排名第二的方案的四倍。

**CapSense 机械按键替换（MBR）**

将机械按键升级到电容式按键的赛普拉斯可配置解决方案仅需要很少的工程开发工作，并且不需要固件开发。这些器件包括 CY8CMBR3XXX 和 CY8CMBR2XXX 系列。

**质心或质心位置**

是指在滑条分辨率所给定的范围内，表示滑条上的手指位置的数字。该数字由 CapSense 质心计算算法计算得出。

**补偿 IDAC**

指的是可编程的恒流源，CSD 通过使用该恒流源补偿多余的传感器 $C_P$。与调制 IDAC 不同，该 IDAC 没有受 CSD 模块中 Sigma-delta 调制器的控制。

**CSD**

CapSense Sigma Delta（CSD）是赛普拉斯专利方法，用于测量电容式感应应用的自电容。

在 CSD 模式下，感应系统测量电极的自电容，且检测自电容的变化，从而确定是否有手指触摸。

**去抖动**

用于定义连续扫描样本数量的参数，只有存在手指触摸时，该参数才有效。该参数有助于抑制误触摸触摸信号。

对于连续扫描样本的去抖动数量，仅在计数差值大于手指阈值+迟滞时，手指触摸才被报告。

**驱动屏蔽**

指的是 CSD 所使用的一种技术，用于使能防水功能，其中屏蔽电极由一个信号驱动，该信号的相位和幅度与传感器开关信号的相等。

**电极**

指的是导电材料，如 PCB 板、ITO 或 FPCB 板上的垫片或物理层。电极连接到 CapSense 器件的端口引脚，并作为 CapSense 传感器使用或用于驱动与 CapSense 功能相关的特定信号。

**手指阈值**

与 Hysteresis（迟滞）一起使用的参数，旨在确定传感器的状态。如果计数差值高于手指阈值+迟滞，传感器状态将显示‘ON’；如果计数差值低于手指阈值-迟滞，则传感器状态将显示‘OFF’。

**组合传感器**

这是将多个传感器连接在一起，并将它们作为单个传感器进行扫描的方法。该方法用于扩大接近感应的传感器面积，并降低功耗。

当系统处于低功耗模式时，为了降低功耗，需要将所有传感器连接在一起并将其作为单个传感器进行扫描（而不是单独扫描所有传感器），这样可以缩短扫描时间。当用户触摸任何传感器时，系统会进入活动模式，在该模式中，它会单独扫描所有传感器，以检测哪个传感器被激活。

PSoC 通过固件支持传感器组合，这意味着，可以将多个传感器同时连接到 AMUXBUS，以进行扫描。

**手势**

手势是一个由用户执行的动作，如滑动和线捏/缩放等等。CapSense 具有手势检测功能，即根据预定义的触摸格式来识别不同的手势。在 CapSense 组件中，只有触摸板 widget 支持手势功能。

**保护传感器**

指的是 PCB 板上围绕所有传感器的铜线，它类似于按键传感器并用于检测水流。触发保护传感器时，固件会禁用对所有其它传感器进行的扫描，以防止误触摸。

**网格填充、网格地填充或网格铺地**

当设计一个拥有电容式感应功能的 PCB 板时，应将铜制接地层放置在传感器周边，以获取良好的抗噪能力。但是实心接地层会使传感器的寄生电容增加（这种电容是不需要的）。因此，应该以特殊网格方式填充接地层。网格图案被紧密放置、纵横交错，同丝网一样，线宽度和两条线间的距离决定了填充百分比。要求具有防水功能时，将通过屏蔽信号（而不是接地层）驱动该网格填充（作为屏蔽电极使用）。

**迟滞**

用于防止由系统噪声产生随机切换造成传感器状态的参数，它与手指阈值一起使用，以确定传感器状态。请查看手指阈值。

**IDAC（电流输出的数模转换器）**

PSoC 中的可编程恒流源，用于 CapSense 和 ADC 操作。

**防水功能**

存在水滴、水流或薄雾时，电容感应系统仍能够正常工作的能力。

**线性滑条**

指的是至少包含一个传感器的 Widget。这些传感器以特殊的线性方式安排以检测手指的物理位置（在单轴上）。

**低基准线复位**

表示扫描样本最大数量的参数，其中原始计数异常低于负噪声阈值。如果超过了低基准线复位值，基准线将被复位到当前的原始计数。

**手动调校**

指的是手动设置（或调校）CapSense 参数的过程。

**矩阵按键**

指的是至少包含两个传感器（这些传感器以矩阵方式安排）的 widget。通过使用它可以在各个传感器（这些传感器以垂直方向和横向安排）的交点上检测是否有手指（触摸）。

如果 M 是横轴上的传感器数量，且 N 是纵轴上的传感器数量，那么矩阵按键 Widget 只需要使用 M + N 端口引脚就可以监控 M x N 总交叉点。

使用 CSD 感应方法（自电容）时，该 Widget 一次只能检测一个交叉点位置上的有效触摸。

**调制电容（CMOD）**

在自电容感应模式下 CSD 模块操作所需要的外部电容。

**调制器时钟**

指的是一个时钟源，在传感器扫描过程中用于采样从 CSD 模块输出的调制器。该时钟也是原始计数计数器的源。扫描时间（不包括前处理和后处理时间）的计算公式为$(2^N – 1)$/调制器的时钟频率，其中 N 是扫描分辨率。

**调制 IDAC**

调制 IDAC 是可编程的恒流源，它的输出由 CSD 模块中的 Sigma-delta 控制器输出控制（ON/OFF），以保持 AMUXBUS 电压始终为 $V_{REF}$。该 IDAC 提供的平均电流等于传感器电容引出的平均电流。

**互电容**

一个电极（假设为 TX）与另一个电极（假设为 RX）间的相对电容被称为互电容。

**负噪声阈值**

用于区分通常噪声与不想要的杂散信号的阈值。该参数与低基准线复位参数结合使用。

通过更新基准线，可以跟踪原始计数和负噪声阈值范围内的原始计数的变化，也就是基准线与原始计数之差（基准线–原始计数）小于负噪声阈值。

负方向的杂散信号可被触发的场合包括：上电时传感器上有手指触摸，除去传感器附近的金属物体，移除带有防水功能的 CapSense 产品上的水滴，以及突然发生其它的环境变化。

**噪声（CapSense 噪声）**

传感器处于‘OFF’状态（无触摸）时原始计数的变量，使用峰至峰计数来测量。

**噪声阈值**

用于区分传感器的信号和噪声的参数。如果原始计数 – 基准线的值大于噪声阈值，该参数将表示信号可能有效。如果差值小于噪声阈值，则该原始计数仅包括噪声。

**覆盖层**

指的是覆盖电容式传感器，并用作触摸表面的非导电材料（如塑胶和玻璃）。将带有多个传感器的 PCB 直接放置在覆盖层下面，或通过弹簧连接。产品的外壳常作为覆盖层使用。

**寄生电容（$C_P$）**

寄生电容是由 PCB 走线、传感器焊盘、过孔以及气隙组成的传感器电极的内部电容。这是不想要的情况，因为它会使 CSD 的灵敏度降低。

**接近感应传感器**

指的是不需要物理接触却能够检测到附近的物体的传感器。

**辐射滑条**

指的是包含多于一个传感器的 Widget。这些传感器以特殊的圆形方式设置，以检测手指的物理位置。

**原始计数**

代表传感器物理电容的 CapSense 硬件模块的未处理数值输出。

**刷新间隔**

传感器两次连续扫描间的时间。

**扫描分辨率**

由 CSD 模块生产的原始计数分辨率（单位为位）。

**扫描时间**

完成传感器的扫描过程所需要的时间。

**自电容**

与电路接地和电极相关的电容。

**灵敏度**

指的是原始计数随传感器电容的变化，用计数/pF 来表示。传感器灵敏度取决于电路板布局、覆盖层属性、感应方法以及调校参数。

**感应时钟**

用来实现 CSD 感应方法的开关电容前端的时钟源。

**传感器**

请参见电容式传感器。

**传感器自动复位**

指的是一种设置，用于防止传感器无限期地报告由系统故障或金属物体连续出现在传感器附近时造成的误触摸状态。

使能传感器自动复位时，即使计数差值大于噪声阈值，也更新基准线。这样将防止传感器无限期地报告‘ON’状态。禁用传感器自动复位时，只有计数差值小于噪声阈值时才能更新基准线。

**传感器组合**

请参见组合传感器。

**屏蔽电极**

传感器周围填充铜，以便防止水滴或其它液体引起的误触摸。屏蔽电极由 CSD 模块输出的屏蔽信号驱动。请参见驱动屏蔽。

**屏蔽槽电容（$C_{SH}$）**

指的是（当有一个带有高的寄生电容的大屏蔽层时，）用于增强 CSD 屏蔽的驱动能力的可选外部电容（$C_{SH}$ 槽电容）。

**信号（CapSense 信号）**

计数差值还被称为信号。请参见计数差值。

**信噪比（SNR）**

有手指触摸时的传感器信号与无手指触摸时的传感器信号间的比例。

**滑条分辨率**

表示滑条上需要处理的手指位置总数的参数。

**触摸板**

指的是包含多个传感器的 Widget（这些传感器以特殊的横向和纵向安排），用于检测一个触摸的 X 和 Y 位置。

**触摸板**

请参见触摸板。

**调校**

“调校”是使 CapSense 操作中所需的各种硬件和软件或阈值参数达到最佳值的过程。

**$V_{REF}$**

PSoC 中的可编程参考电压模块，用于 CapSense 和 ADC 操作。

**Widget**

指的是 CapSense 组件中包括一个传感器或一组类似传感器的用户界面元素。受支持的 widget 包括按键、接近感应传感器、线性滑条、辐射滑条，矩阵按键和触摸板。

# 修订记录

## 文档修订记录

| 文档标题：AN78329 — CY8C20xx7/S CapSense®设计指南<br>文档编号：001-88330 | | |
|---|---|---|
| **版本** | **提交日期** | **变更说明** |
| ** | 07/05/2012 | 本文档版本号为 Rev**，译自英文版 001-78329 Rev*C。 |
| *A | 06/23/2015 | 本文档版本号为 Rev*A，译自英文版 001-78329 Rev*E。 |
| *B | 03/31/2020 | 本文档版本号为 Rev*B，译自英文版 001-78329 Rev*H。 |