



# AN65973 - CY8C20xx6A/H/AS

## CapSense®设计指南

文档编号: 001-78419 版本\*B

赛普拉斯半导体公司  
198 Champion Court  
San Jose, CA 95134-1709  
电话（美国）：800.858.1810  
电话（国际）：408.943.2600  
[www.cypress.com](http://www.cypress.com)

版权所有

## 版权所有

© 赛普拉斯半导体公司，2010-2016。此处所包含的信息可随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路外，赛普拉斯半导体公司不对任何其它电路的使用承担任何责任。也不会根据专利权或其它权利以明示或暗示方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯不保证产品能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于有可能因为发生运转异常和故障对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

## 商标

PSoC Designer™、Programmable System-on-Chip™ 和 SmartSense™ 是赛普拉斯半导体公司的商标，PSoC® 和 CapSense® 是赛普拉斯半导体公司的注册商标。此处引用的所有其他商标或注册商标均归其各自所有者所有。

## 源代码

所有源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途外，未经赛普拉斯明确的书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

## 免责声明

赛普拉斯不针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。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于合理预计可能发生运转异常和故障，并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能适用于赛普拉斯软件许可协议的限制。

# 目录



<b>1. 简介</b>	<b>6</b>
1.1 摘要	6
1.2 赛普拉斯的 CapSense 文档体系	6
1.3 CY8C20xx6A/H/AS CapSense 系列特性	8
1.3.1 高级触摸式感应功能	8
1.3.2 器件特性	9
1.4 文档规范	10
<b>2. CapSense 技术</b>	<b>11</b>
2.1 CapSense 基本原理	11
2.2 CY8C20xx6A/AS/H 中的电容式感应方法	12
2.2.1 CapSense Sigma-Delta (CSD)	12
2.2.2 CapSense 逐次逼近电磁兼容性 (CSA_EMC)	14
2.3 SmartSense 自动调校	15
<b>3. CapSense 设计工具</b>	<b>17</b>
3.1 概况	17
3.1.1 PSoC Designer 和用户模块	17
3.1.2 通用的 CapSense 控制器套件	18
3.1.3 通用 CapSense 控制器模块电路板	18
3.1.4 CapSense 数据查看工具	19
3.2 用户模块概述	19
3.3 CapSense 用户模块全局阵列	20
3.3.1 原始计数	20
3.3.2 基准线	20
3.3.3 计数差值 (信号)	20
3.3.4 传感器状态	20
3.4 CSD 用户模块参数	21
3.4.1 用户模块高级参数	22
3.4.2 CSD 用户模块低级参数	24
3.4.3 CSA_EMC 用户模式低级参数	25
3.4.4 SmartSense 用户模块参数	27
3.4.5 SmartSense_EMC 用户模块参数	28

<b>4. 使用用户模块对 CapSense 性能进行调校</b>	<b>30</b>
4.1 基本注意事项	30
4.1.1 信号、噪声和信噪比 (SNR)	30
4.1.2 充电/放电率	31
4.1.3 基准线更新阈值验证的重要性	32
4.2 调校 CSA_EMC 用户模块	32
4.3 CSA_EMC 的建议 C <sub>INT</sub> 值	33
4.4 测量传感器 C <sub>P</sub>	33
4.5 估算 CSA_EMC 时钟	34
4.6 设置建立时间	34
4.7 监测 CapSense 数据	35
4.8 增大信噪比的方法	35
4.8.1 降低噪声	35
4.8.2 提高信号	35
4.9 调校 CSD 用户模块	35
4.9.1 CSD 的建议 C <sub>MOD</sub> 值	36
4.9.2 ShieldElectrodeOut (屏蔽电极输出)	36
4.9.3 I <sub>DAC</sub> 范围	37
4.9.4 自动校准	37
4.9.5 I <sub>DAC</sub> 值	37
4.9.6 预充电源	37
4.9.7 预分频器	37
4.9.8 分辨率	38
4.9.9 扫描速度	38
4.9.10 高级 API 参数	39
4.9.11 设置高级参数	40
4.10 使用 SmartSense 用户模块	40
4.10.1 SmartSense 指南	40
4.10.2 理解差异	40
4.10.3 SmartSense 的建议 C <sub>CMOD</sub> 值	41
4.10.4 SmartSense 用户模块参数	41
4.10.5 SmartSense_EMC 用户模块的专用指南	41
4.10.6 CapSense 传感器的扫描时间	42
4.10.7 SmartSense 响应时间	43
4.10.8 使用 SmartSense_EMC 用户模块尽量降低信噪比的方法	44
4.10.9 固件设计指南	45
<b>5. 设计注意事项</b>	<b>47</b>
5.1 覆盖层选择	47
5.2 ESD 保护	48
5.2.1 预防	48
5.2.2 重定向	48
5.2.3 钳制	48
5.3 电磁兼容性 (EMC) 的注意事项	48
5.3.1 辐射干扰	48

## 目录

5.3.2	辐射 .....	49
5.3.3	抗传导干扰和辐射 .....	49
5.4	软件滤波 .....	49
5.5	功耗 .....	50
5.5.1	系统设计建议 .....	50
5.5.2	睡眠-扫描方法 .....	50
5.5.3	响应时间与功耗 .....	50
5.5.4	测量平均功耗 .....	51
5.6	引脚分配 .....	51
5.7	GPIO 负载瞬态 .....	52
5.7.1	降低 GPIO 负载瞬态噪声的硬件指南 .....	53
5.7.2	补偿 GPIO 负载瞬态噪声的固件指南 .....	53
5.8	PCB 布局指南 .....	55
<b>6.</b>	<b>低功耗设计的注意事项.....</b>	<b>56</b>
6.1	其他节能技术 .....	56
6.1.1	将驱动模式设置为模拟高阻 .....	56
6.1.2	全部放在一起 .....	57
6.1.3	睡眠模式复杂性 .....	57
6.1.4	挂起中断 .....	57
6.1.5	全局中断使能 .....	57
6.2	唤醒后执行序列 .....	58
6.2.1	使能 PLL 模式 .....	58
6.2.2	执行 Global Interrupt Enable（全局中断使能） .....	58
6.2.3	I <sup>2</sup> C 从设备处于睡眠模式 .....	58
6.2.4	睡眠定时器 .....	58
<b>7.</b>	<b>资源 .....</b>	<b>59</b>
7.1	网站 .....	59
7.2	数据手册 .....	59
7.3	技术参考手册 .....	59
7.4	开发套件 .....	59
7.4.1	通用的 CapSense 控制器套件 .....	59
7.4.2	通用 CapSense 模块板 .....	59
7.4.3	在线仿真（ICE）套件 .....	60
7.5	样本电路板文件 .....	60
7.6	PSoC Programmer .....	62
7.7	CapSense 数据查看工具 .....	62
7.8	PSoC Designer .....	62
7.9	代码示例 .....	63
7.10	设计支持 .....	63
	<b>术语表 .....</b>	<b>64</b>
	<b>修订记录.....</b>	<b>69</b>
	文档修订记录 .....	69

# 1. 简介



## 1.1 摘要

本文档介绍的是如何利用 CapSense 控制器中的 CY8C20xx6A/AS/H 系列来实现电容式感应（CapSense®）功能。本指南包含以下主题：

- CapSense 控制器中 CY8C20xx6A/AS/H 系列的特性
- CapSense 的操作原理
- CapSense 设计工具简介
- 调校 CapSense 系统的性能到最佳状态的详细指南
- 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。通过这些资源，用户可以快速访问所需要的信息，从而成功完成 CapSense 产品设计。图 1-1 显示了利用电容式感应进行产品设计的典型循环流程；本指南中与该主题密切相关的信息用绿色文字高亮显示。表 1-1 提供了与图 1-1 中所列出的每个已编号任务相对应的文档链接。

图 1-1. 典型的 CapSense 产品设计流程

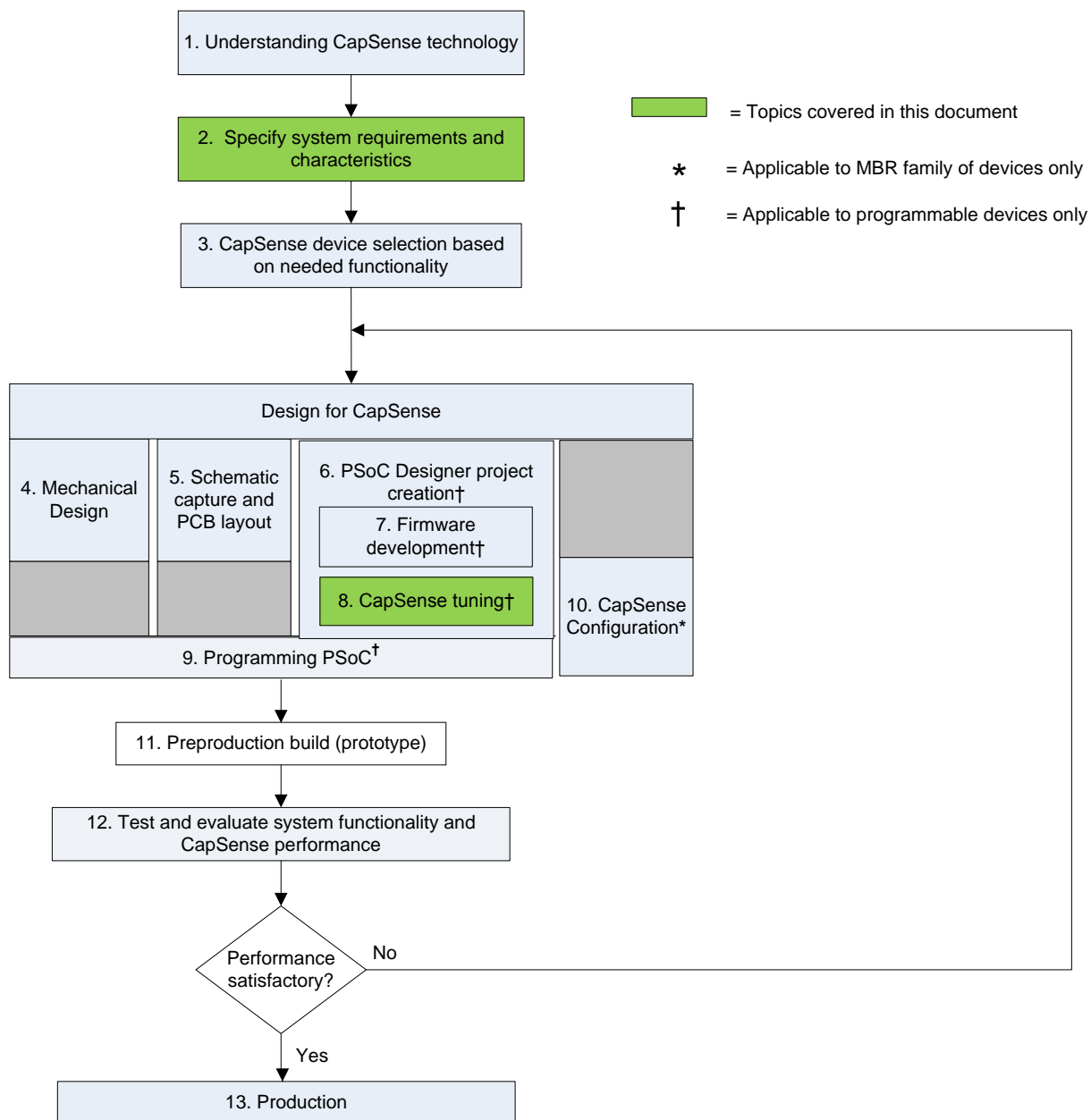


表 1-1. 图 1-1 中每个已编号的任务所对应的赛普拉斯文档

图 1-1 中已编号的设计任务	赛普拉斯的 CapSense 文档
1	<ul style="list-style-type: none"> <li>• <a href="#">CapSense 入门</a></li> </ul>
2	<ul style="list-style-type: none"> <li>• <a href="#">CapSense 入门</a></li> <li>• <a href="#">CY8C20xx6A/AS/H CapSense 器件数据手册</a></li> </ul>
3	<ul style="list-style-type: none"> <li>• <a href="#">CapSense 入门</a></li> <li>• <a href="#">PSoC 系列的特定 CapSense 设计指南（本文档）</a></li> </ul>
4	<ul style="list-style-type: none"> <li>• <a href="#">CapSense 入门</a></li> </ul>
5	<ul style="list-style-type: none"> <li>• <a href="#">CapSense 入门</a></li> <li>• <a href="#">PSoC Designer™ 用户指南</a></li> </ul>
6	<ul style="list-style-type: none"> <li>• <a href="#">PSoC Designer 用户指南</a></li> </ul>
7	<ul style="list-style-type: none"> <li>• <a href="#">汇编语言用户指南</a></li> <li>• <a href="#">C 语言编译器用户指南</a></li> <li>• <a href="#">CapSense 代码示例</a></li> <li>• <a href="#">PSoC 系列的特定技术参考手册（适用于 CY8C20xx6A/AS/H）</a></li> </ul>
8	<ul style="list-style-type: none"> <li>• <a href="#">PSoC 系列的特定 CapSense 设计指南（本文档）</a></li> <li>• <a href="#">PSoC 系列的特定 CapSense 用户模块数据手册（CSD 和 SmartSense™）</a></li> <li>• <a href="#">PSoC 系列的特定技术参考手册（适用于 CY8C20xx6A/AS/H）</a></li> <li>• <a href="#">CapSense 控制器代码示例设计指南</a></li> <li>• <a href="#">AN2397 — CapSense 数据查看工具</a></li> </ul>
9	<ul style="list-style-type: none"> <li>• <a href="#">编程器用户指南</a></li> <li>• <a href="#">MiniProg3 用户指南</a></li> <li>• <a href="#">AN2026c — 用于 CY8C20xx6、CY8C20xx6A、CY8CTMG2xx、CY8CTST2xx、CY7C643xx 和 CY7C604xx 的系统内串行编程（ISSP）协议</a></li> <li>• <a href="#">AN44168 — 使用外部微控制器（HSSP）的 PSoC 1 器件编程</a></li> <li>• <a href="#">AN59389 — CY8C20xx6、CY8CTMG2xx 和 CY8CTST2xx 的主机源串口编程</a></li> </ul>
11	<ul style="list-style-type: none"> <li>• <a href="#">PSoC 系列的特定 CapSense 设计指南（本文档）</a></li> <li>• <a href="#">CapSense 代码示例</a></li> </ul>

## 1.3 CY8C20xx6A/H/AS CapSense 系列特性

赛普拉斯 CY8C20xx6A/H/AS 是一个低功耗、高性能、可编程的 CapSense 控制器系列，其功能如下：

### 1.3.1 高级触摸式感应功能

- 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使用
  - ☐ 实现按键和滑条的集成 API
  - ☐ 支持多达 36 个电容传感器或 36 个 GPIO/滑条
  - ☐ 支持寄生电容为 5 pF 到 45 pF 的传感器
- 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 — 自动对用户界面设计中的更改进行调校

## 简介

- ☐ 运行期间的环境补偿
- ☐ 检测低至 0.1 pF 的触摸
- 增强的抗噪能力和稳定性
  - ☐ SmartSense 自动补偿环境和噪声变化
  - ☐ SmartSense\_EMC 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
  - ☐ 内部调节器提供抗电源噪声的稳定性，并可接受纹波最高为 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）
  - ☐ 在扫描速率为 125 ms 的情况下，每个传感器的电流为 28  $\mu$ A

### 1.3.2 器件特性

- 高性能、低功耗 M8C Harvard 架构处理器
  - ☐ 运行速度高达 4 MIPS，拥有 24 MHz 的内部时钟、外部晶体谐振器或时钟信号
- 灵活的片上存储器
  - ☐ 高达 32 KB 的闪存和 2 KB 的 SRAM
  - ☐ 模拟 EEPROM
- 高精度的可编程时钟
  - ☐ 内部主振荡器（IMO）：6/12/24 MHz  $\pm$  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
  - ☐ 高达 36 个 GPIO，具有可编程引脚配置
  - ☐ 25 mA 灌电流/GPIO 和 120 mA 的总灌电流/器件
  - ☐ 所有 GPIO 上均可采用内部电阻上拉、高阻态、开漏驱动和强驱动模式
- 外设功能
  - ☐ 三个 16 位定时器
  - ☐ 全速 USB — 符合 12 Mbps USB 2.0 标准
  - ☐ I<sup>2</sup>C — 主设备（100 kHz）和从设备（高达 400 kHz）
  - ☐ SPI — 主设备和从设备 — 可配置范围从 46.9 kHz 到 12 MHz
  - ☐ 高达 10 位的 ADC — 输入范围为 0 到 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： C:\ ...cd\icc\
斜体字	用于显示文件名称和参考文档： 请阅读 <i>PSoC Designer 用户指南</i> 中的 <i>sourcefile.hex</i> 文件。
[方括号、粗体]	用于显示程序中的键盘指令： <b>[Enter]</b> 或 <b>[Ctrl] [C]</b>
File（文件）> Open（打开）	表示菜单路径： File > Open > New Project
粗体字	用于显示操作过程中的各条指令、菜单路径和图标名称： 请点击 <b>File</b> 图标，然后点击 <b>Open</b> 。
Times New Roman 字体	用于显示公式： $2 + 2 = 4$
灰色框中的文本	用于说明警告或产品的独特功能。

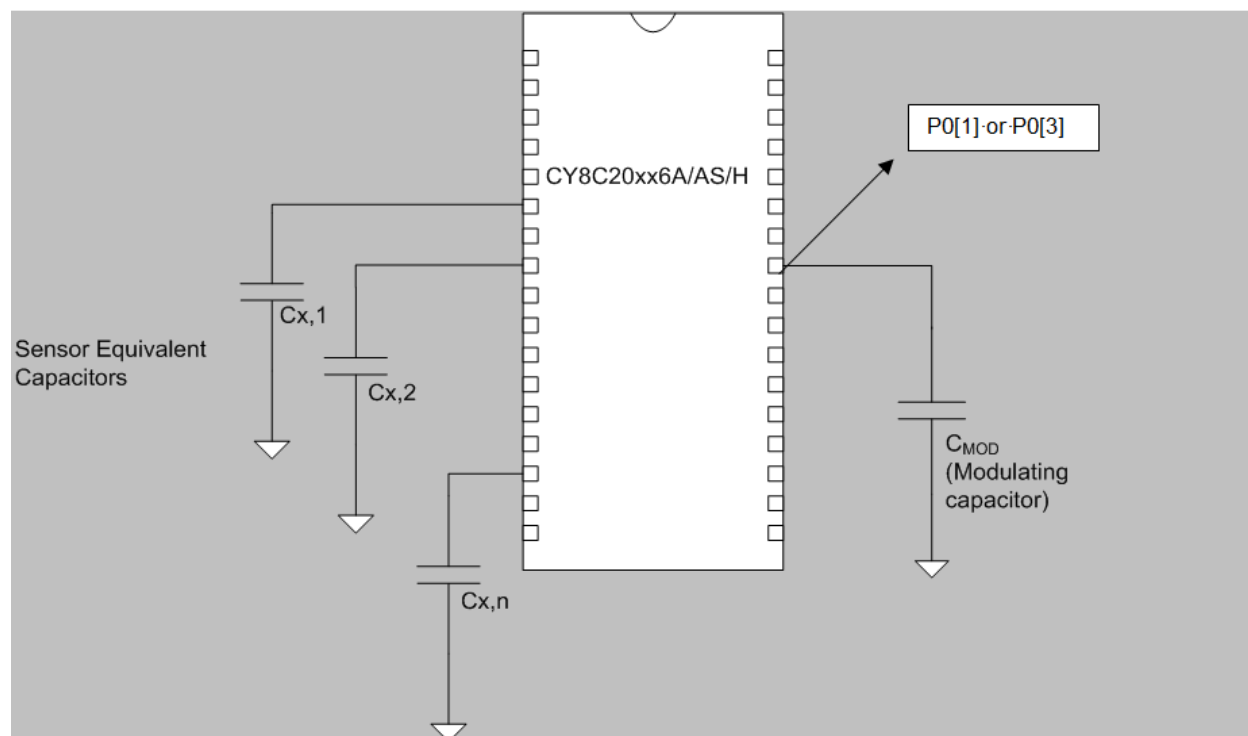
## 2. CapSense 技术



### 2.1 CapSense 基本原理

CapSense 是一种触摸式感应技术，其工作方式是在预先指定为传感器的 CapSense 控制器上测量每个 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模拟为数值为  $C_{x,1}$  到  $C_{x,n}$  的等效总电容值。CY8C20xx6A/AS/H 器件的内部电路将各个  $C_x$  的大小转换为数字代码进行保存，以供后期处理。CapSense 控制器的内部电路会使用其他组件  $C_{MOD}$ ，更多信息会在 CY8C20xx6A/AS/H 中的电容式感应方法电容式感应方法中讨论。

图 2-1. 在 CY8C20xx6A/AS/H PSOC 器件中实现 CapSense



如图 2-1 所示，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。覆盖层是传感器板上面的非导电性覆盖层，这便构成了该产品的触摸式接口。当手指触摸到覆盖层时，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。该操作构成了一个平行板电容；其容值可通过公式 1 计算得到：

$$C_F = \frac{\epsilon_0 \epsilon_r A}{D} \quad \text{公式 1}$$

其中：

$C_F$  = 手指触摸传感器覆盖层时所产生的电容值

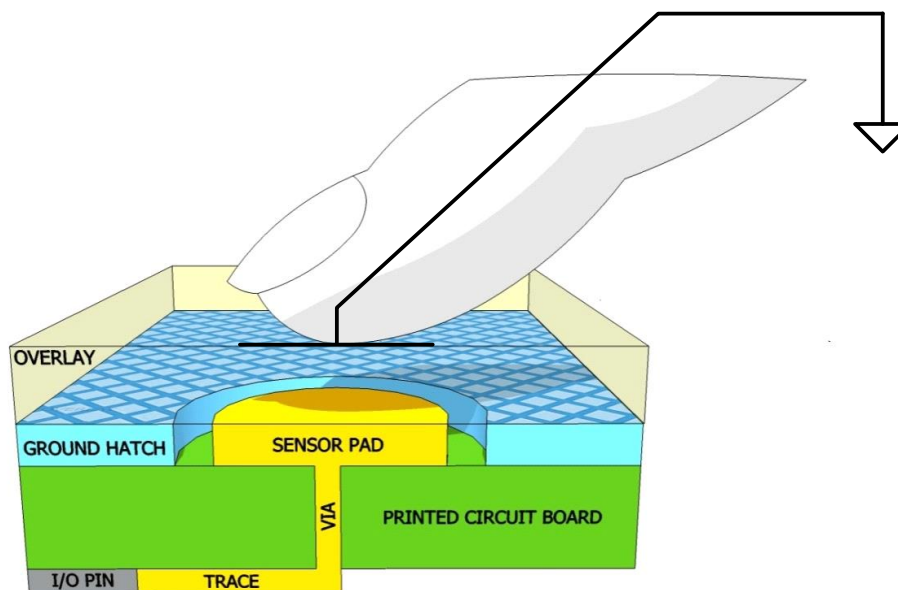
$\epsilon_0$  = 空气介电常数

$\epsilon_r$  = 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



除了平行板电容值外，当手指触摸到覆盖层时，也会在其自身与附近其他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，其中电场是在传感器板、走线和过孔以及系统中其他导体（如接地层、其他走线、产品机壳或外壳中的任何金属）之间耦合产生的。CapSense 控制器可测量连接至传感器引脚的总电容 ( $C_X$ )。

当手指未接触传感器时：

$$C_X = C_P \quad \text{公式 2}$$

当手指触摸传感器板时， $C_X$  等于  $C_P$  与  $C_F$  之和：

$$C_X = C_P + C_F \quad \text{公式 3}$$

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。调校 CapSense 系统时， $C_P$  的数量级至关重要，这将在[通过用户模块调校 CapSense 性能](#)中进行讨论。

## 2.2 CY8C20xx6A/AS/H 中的电容式感应方法

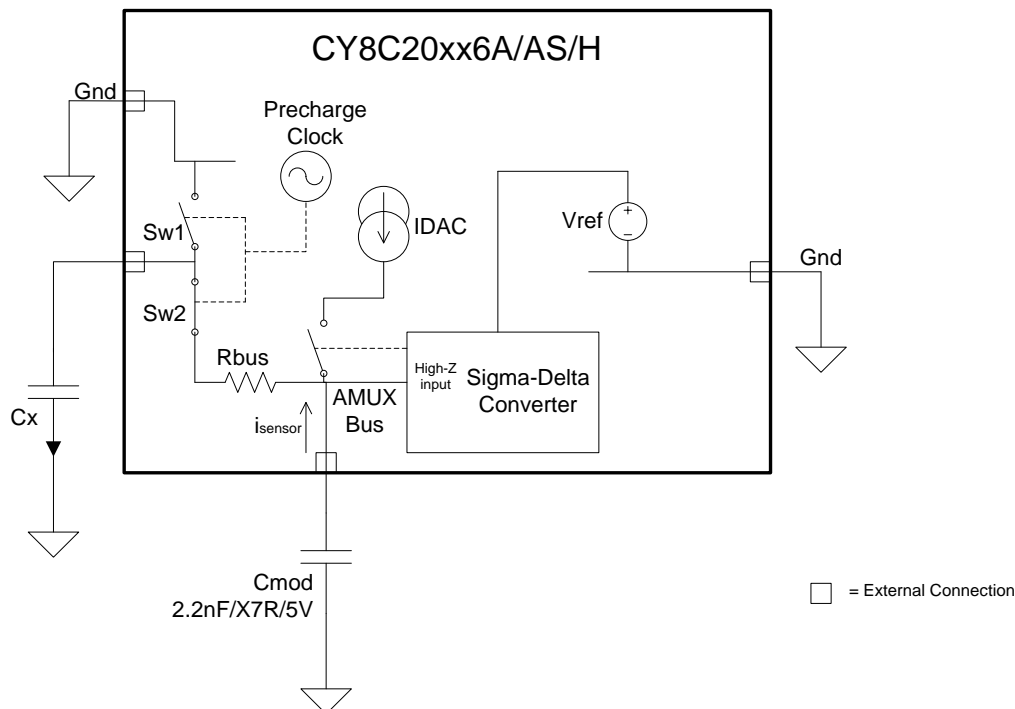
CY8C20xx6A/AS/H 器件支持多个用来将传感器电容 ( $C_X$ ) 转换成数字计数的 CapSense 方法。这些方法包括 CapSense Sigma Delta (CSD)、CapSense 逐次逼近电磁兼容 (CSA\_EMC)、SmartSense 和 SmartSense\_EMC。这些方法在 PSoC Designer 用户模块中实现，并在以下章节中进行描述。

### 2.2.1 CapSense Sigma-Delta (CSD)

如图 2-3 所示，CY8C20xx6A/AS/H 器件中的 CapSense Sigma-Delta 方法将  $C_X$  整合到开关电容电路中。该传感器 ( $C_X$ ) 分别通过开关 Sw1 和 Sw2 交替连接到 GND 和模拟复用器 (AMUX) 总线。Sw1 和 Sw2 由预充电时钟驱动，通过模拟复用器 (AMUX) 总线释放电流 ( $I_{\text{SENSOR}}$ )。 $I_{\text{SENSOR}}$  的数量级与  $C_X$  的数量级成正比。Sigma-Delta 转换器采

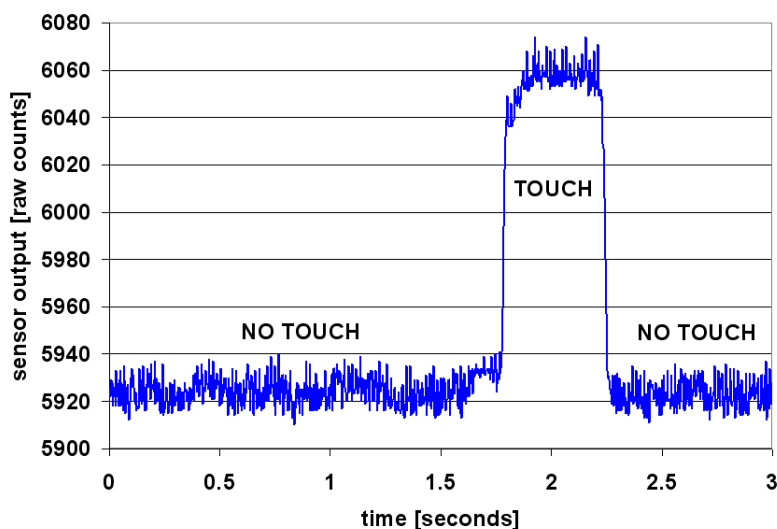
集 AMUX 总线电压样本并生成一个用来控制恒流源 ( $I_{DAC}$ ) 的调制位流, 该恒流源对 AMUX 充电以使 AMUX 总线的平均电压便保持为  $V_{REF}$ 。传感器释放调制电容器 ( $C_{MOD}$ ) 中的电荷  $I_{SENSOR}$ 。 $C_{MOD}$  与  $R_{bus}$  共同构成一个低通滤波器, 该滤波器消弱了 Sigma-Delta 转换器输入的预充电开关跃变电压。

图 2-3. CSD 框图



保持 AMUX 平均电压为稳态值 ( $V_{REF}$ ) 时, Sigma-Delta 转换器通过控制位流占空比的方式保持平均的泄漏电流 ( $I_{DAC}$ ) 与  $I_{SENSOR}$  相互匹配。Sigma-Delta 转换器在传感器扫描期间内存储位流, 其累积结果为数字输出值, 称为原始计数, 该值与  $C_x$  成正比。原始计数由高级算法进行解析, 用于确定传感器的状态。在手指触摸传感器然后再释放的过程中, 得到若干连续扫描结果, 可根据扫描结果绘制出 CSD 原始计数, 如图 2-4 所示。正如 CapSense 基本原理所解释, 手指触摸使  $C_x$  增加到  $C_F$ , 依次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。

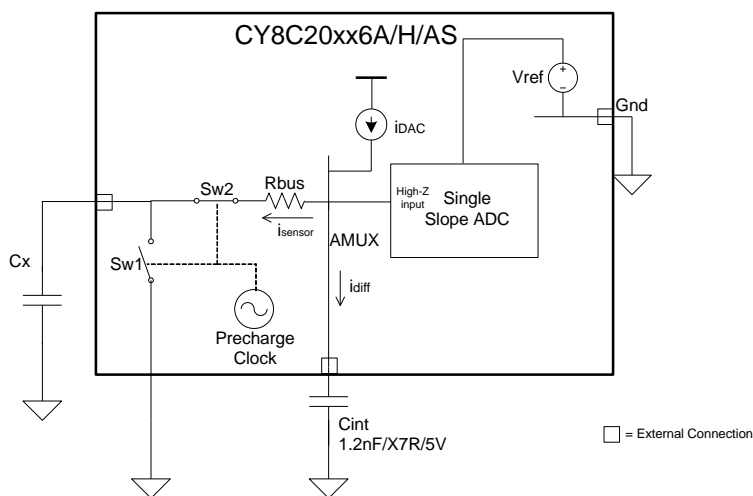
图 2-4. 手指触摸期间的 CSD 原始计数



## 2.2.2 CapSense 逐次逼近电磁兼容性 (CSA\_EMC)

如图 2-5 所示，CY8C20xx6A 器件应用 CapSense 逐次逼近电磁兼容 (CSA\_EMC) 方法，将  $C_x$  整合到开关电容电路上。

图 2-5. CSA\_EMC 框图



恒流源 ( $I_{DAC}$ ) 为模拟复用器 (AMUX) 提供的电流量为  $I_{DAC}$ 。传感器 ( $C_x$ ) 分别通过开关  $Sw1$  和  $Sw2$  选择性地连接到 AMUX 总线和 GND，耗尽 AMUX 总线流出的  $I_{SENSOR}$  电流量。 $I_{SENSOR}$  的数量级与  $C_x$  的数量级成正比。 $Sw1$  和  $Sw2$  的时钟来自非重叠时钟（称为预充电时钟）。

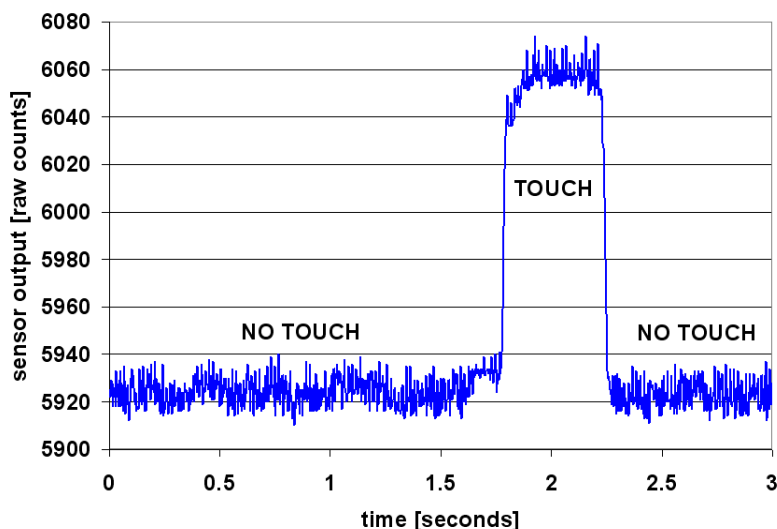
积分电容  $C_{INT}$  集成差动电流  $i_{Diff}$  ( $I_{DAC}$  和  $I_{SENSOR}$  的差值)，并增强其电位。此电荷积分持续进行，直到展开的潜能超过  $C_{INT}$  达到平衡水平，即  $I_{SENSOR}$  等于  $I_{DAC}$ 。此积分时间称为建立时间。

请用单斜 ADC 将  $C_{INT}$  的平衡电位转换为数字输出计数，即与  $C_x$  成比例的原始计数。原始计数由高级算法进行解析，以确定传感器的状态。

用逐次逼近方法设置  $I_{DAC}$  电流，以确保  $C_{INT}$  的均衡电压在 ADC 的线性转换区域。

图 2-6 表示从手指触摸传感器到释放这段时间内多次连续扫描得出的 CSA EMC 原始计数。正如 CapSense 基本原理所解释，手指触摸使  $C_x$  增大到  $C_F$ ，依次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 2-6. 手指触摸期间的 CSA EMC 原始计数



CSA EMC CapSense 算法适用于存在射频干扰的情况。当 CapSense 遭遇传导干扰、交流噪声以及变换器、变压器和电源等其他噪声源时，可使用 CSA EMC。CSA EMC 用户模式低级参数中详细讨论了该主题。

## 2.3 SmartSense 自动调校

调校触摸感应用户接口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 自动调校性能用于简化用户界面开发周期。SmartSense 易于使用，通过省去从原型到批量生产的整个产品开发周期中的调校过程，可大幅缩短设计周期时间。SmartSense 在每个 CapSense 传感器上电时自动对其进行调校，从而监控并维护运行时最佳传感器性能。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。

### 2.3.1.1 过程差异

CY8C20xx6A/H/AS SmartSense 用户模块 (UM) 设计与寄生电容在 5 pF 到 45 pF 范围内的传感器工作（典型的传感器  $C_P$  寄生电容值介于 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 5 pF 到 45 pF 内，无论传感器之间的  $C_P$  怎样变动，都保持每个传感器的响应一致，所以提高了批量生产的产品率。

个别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，从而导致手指触摸信号振幅降低。在某些情况下， $C_P$  值的变化会使系统失谐，使传感器无法达到最佳性能（过于灵敏或不够灵敏）；在最坏的情况下，传感器不能正常工作。在上述任一情况下，您都须要重新调校系统。在某些情况下，还需要重新验证 UI 子系统。SmartSense 自动调校可解决这些问题。

SmartSense 自动调校可以实现平台设计。设想在笔记本电脑上设置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。在本示例中，宽屏设备的按键间距要比标准屏幕机型的大许多。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。这意味着在同一个平台上设计的不同模型，CapSense 按键的寄生电容也可能不同。虽然对于所有笔记本模型而言，这些按键的功能都相同，但是，必须根据不同模型来调校传感器。使用所推荐的最佳实践（参见 CapSense 入门手册中的 PCB 布局），SmartSense 可以帮助您实现平台设计，调校功能会自动并有效地完成。

图 2-7. 21 英寸笔记本电脑模型的多媒体按键设计



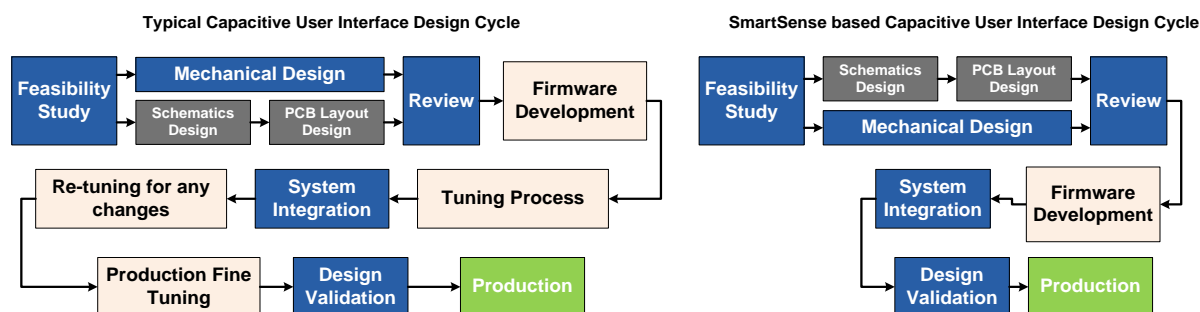
图 2-8. 15 英寸笔记本电脑模型的多媒体按键设计（该模型与 21 英寸模型具有相同的功能和按键大小）



### 2.3.1.2 缩短了设计周期时间

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。使用典型触摸感应控制器时，如果将相同的设计移植到不同的模型，或者 PCB 或传感器 PCB 布局的机械尺寸发生了变化，那么需要重新调校传感器。SmartSense 设计解决了这些问题，因为 SmartSense 需要较少的固件开发工程，无须进行调校和重新调校过程。这大大加快了常规设计周期的进程。图 2-9 对典型触摸感应控制器和 SmartSense 的设计周期进行了比较。

图 2-9. 典型电容式界面设计周期的比较



## 3. CapSense 设计工具



### 3.1 概况

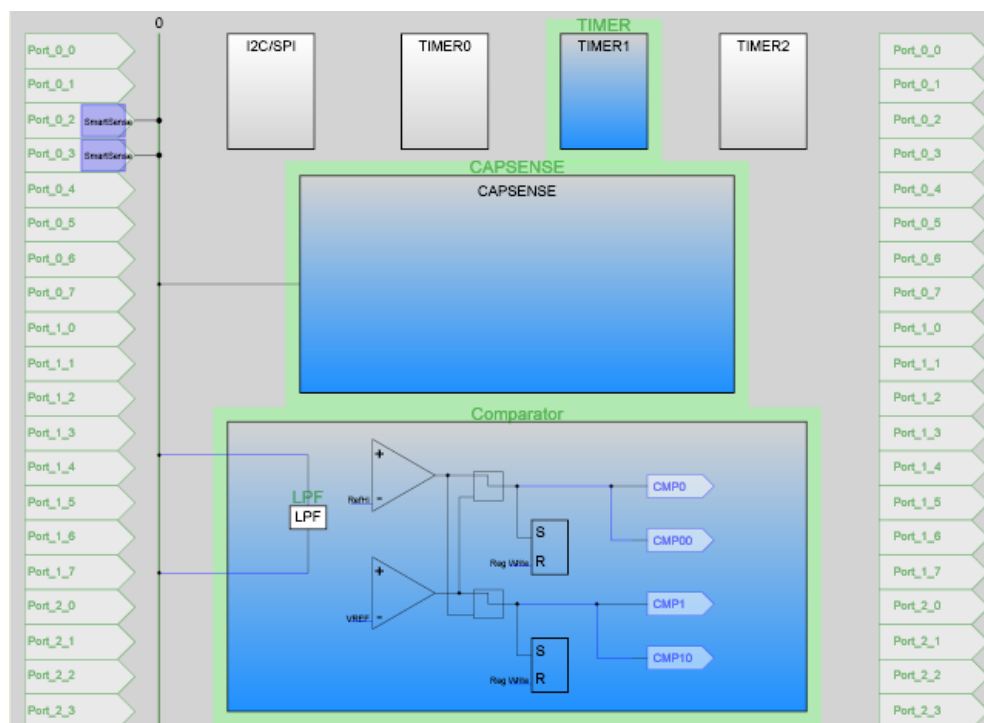
赛普拉斯提供用于开发 CapSense 电容式触摸感应应用的全套硬件和软件开发工具。CY8C20xx6A/H/AS 系列的基本开发系统包括以下部分。有关订购信息，请参见[资源](#)。

#### 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](#)）支持配置模拟和数字模块、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程序。如果想要配置用户模块，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实现。PSoC Designer 附带内置的 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。

CSA\_EMC 用户模块使用开关式电容电路、模拟复用器、电压比较器、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 I<sup>2</sup>C、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 CY8C20xx6A/H/AS 项目：

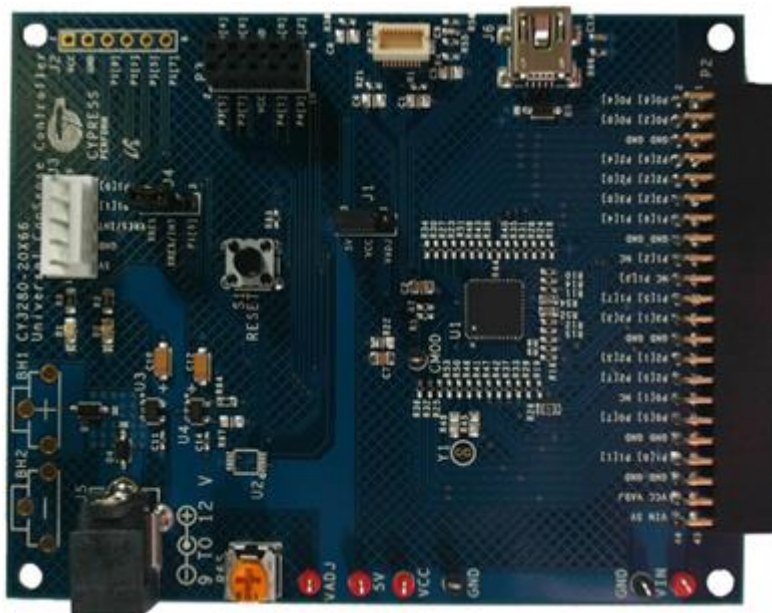
1. 创建一个带有 CY8C20xx6 的 PSoC Designer 项目，并将它作为目标器件。
2. 选择并放置 CSD/CSA\_EMC/SmartSense 用户模块。
3. 右键单击用户模块，以访问用户模块向导。
4. 设置按键传感器计数、滑条配置、引脚分配及相关项。
5. 设置引脚和全局用户模块参数。
6. 生成应用，并切换到应用编辑器。
7. 根据用户模块数据手册调整示例代码，使按键或滑条生效。

欲了解创建 PSoC Designer 项目及配置用户模块向导的详细分步过程，请参见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 通用的 CapSense 控制器套件

通用的 **CY3280-20xx6** CapSense 控制器评估套件带有预定义的控制电路和插入硬件，用来简化原型设计和调试系统。该套件包含用于编程的 MiniProg 硬件以及用于调校和数据采集的 I<sup>2</sup>C-USB 桥接器硬件。

图 3-2. CY3280-20xx6 CapSense 控制器套件



### 3.1.3 通用 CapSense 控制器模块电路板

赛普拉斯模块电路板具有一系列的传感器、LED 和接口，以满足您应用的要求。

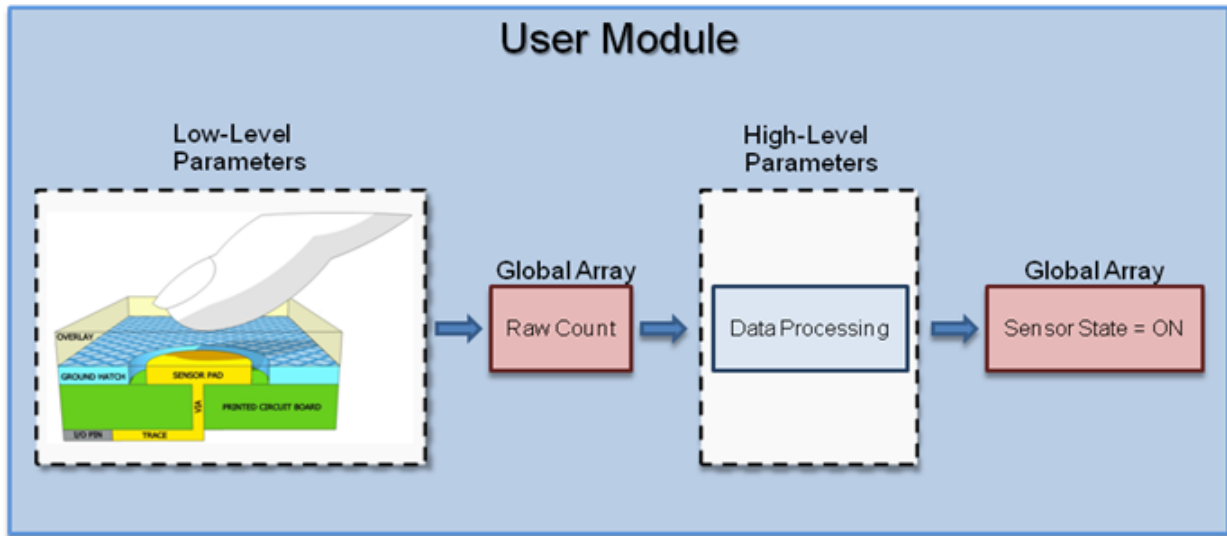
- **CY3280-BSM** 的简单按键模块
- **CY3280-BMM** 矩阵按键模块
- **CY3280-SLM** 线性滑条模块
- **CY3280-SRM** 辐射滑条模块
- **CY3280-BBM** 通用 CapSense 原型设计模块

### 3.1.4 CapSense 数据查看工具

在 CapSense 设计过程中，您需要监测用于调校和调试的 CapSense 数据（原始计数、基准线、计数差值等）。通过两种 CapSense 数据查看工具（即 MultiChart 和桥接控制面板（BCP））可实现该操作。应用笔记 [AN2397 — CapSense 数据查看工具](#)详细介绍了这些工具。

## 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。通过使用各种参数定义用户模块的行为。这些参数影响感应系统的不同器件，可分为低级参数和高级参数，在这些参数之间使用全局阵列进行通信。

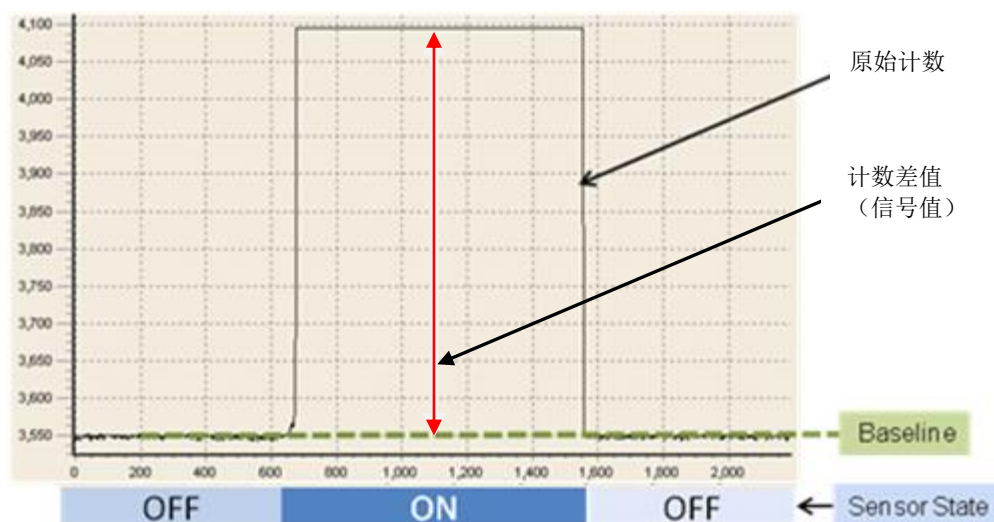
低级参数（如扫描传感器的速度和分辨率）定义物理层上感应方法的性能，并与电容到原始计数的转换相关。每种感应方法都具有单独的低级参数，相关描述请参见 [CSD 用户模块低级参数](#)、[CSA EMC 用户模式低级参数](#)和 [SmartSense 用户模块参数](#)等部分。

高级参数（如去抖动计数和噪声阈值）定义如何处理原始计数，以产生相关信息，例如：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[用户模块高级参数](#)。

### 3.3 CapSense 用户模块全局阵列

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阵列。不能手动更改这些阵列，但如果需要进行调试，您可以对其进行检查。

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。在调用用户模块 API `UMname_ScanSensor()` 函数（`UMname` 可以是 CSD、SmartSense 或 CSA\_EMC）之后，以数字形式（称为原始计数）存储测量结果。

传感器的原始计数与其传感器电容成比例。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` 中定义该阵列。

#### 3.3.2 基准线

渐变的环境改变（如温度和湿度）会影响传感器原始计数，这会导致计数差异。

用户模块使用复杂的基准线算法来补偿这些差异。该算法使用基准线变量来完成该功能。基准线变量记录原始计数值的所有渐变差异。实际上，基准线变量存储数字低通滤波器的输出，而输入的原始计数值被馈送到数字低通滤波器中。

基准线算法由用户模块 API `UMname_UpdateSensorBaseline` 执行（`UMname` 可以为 CSD、SmartSense 或 CSA\_EMC）。

传感器的基准线值被存储在 `UMname_waSnsBaseline[]` 整数型阵列中。在头文件 `UMname.h` 中定义该阵列。

#### 3.3.3 计数差值（信号）

计数差值也被称为传感器信号，被定义为传感器原始计数和基准线值的计数差值。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。

传感器的计数差值存储在 `UMname_waSnsDiff[]` 整数型阵列中，这里 `UMname` 可以是 CSD、SmartSense 或 CSA\_EMC。在头文件 `UMname.h` 中定义该阵列。

差值变量通过用户模块 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，传感器的状态返回到 0。

传感器的状态存储在名为 `UMname_baSnsOnMask[]` 的字节阵列，这里 `UMname` 可以是 `CSD`、`SmartSense` 或 `CSA_EMC`。在头文件 `UMname.h` 中定义该阵列。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_blsAnySensorActive()` 更新。

### 3.4 CSD 用户模块参数

CSD 用户模块参数分为高级参数和低级参数两种。有关 `CSA_EMC` 用户模块参数列表及分类的信息，请参见图 3-5。

有关 `CSD` 用户模块参数列表及分类的信息，请参见图 3-6。

图 3-5. PSoC Designer — CSA\_EMC 参数窗口

高级

低级

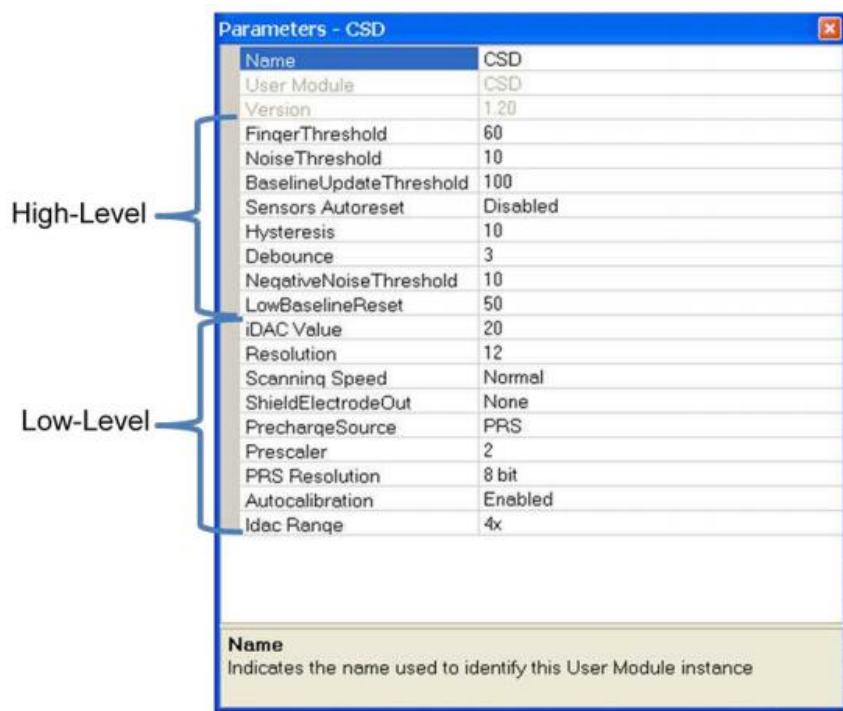
高级

低级

Parameters - CSA_EMC	
Name	CSA_EMC
User Module	CSA_EMC
Version	1.00
FingerThreshold	60
NoiseThreshold	10
BaselineUpdateThreshold	100
SettlingTime	20
Hysteresis	10
Debounce	3
NegativeNoiseThreshold	10
LowBaselineReset	50
Sensors Autoreset	Disabled
Freq Num	3
Spread Spectrum	Disable
RawData Median Filter	Disable
RawData IIR Filter	Disable
RawData IIR Filter Coeff	2
Clock	IMO

**Name**  
Indicates the name used to identify this User Module instance

图 3-6. PSoC Designer — CSD 参数窗口



Name	CSD
User Module	CSD
Version	1.20
FingerThreshold	60
NoiseThreshold	10
BaselineUpdateThreshold	100
Sensors Autoreset	Disabled
Hysteresis	10
Debounce	3
NegativeNoiseThreshold	10
LowBaselineReset	50
iDAC Value	20
Resolution	12
Scanning Speed	Normal
ShieldElectrodeOut	None
PrechargeSource	PRS
Prescaler	2
PRS Resolution	8 bit
Autocalibration	Enabled
Idac Range	4x

**Name**  
Indicates the name used to identify this User Module instance

### 3.4.1 用户模块高级参数

#### 3.4.1.1 手指阈值

用户模块使用此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超过手指阈值，则表示该传感器处于活动状态。该定义假设迟滞水平为 0，去抖动值为 1。

取值范围为 3 到 255。

更多有关建议值的信息，请参见[设置高级参数](#)。

#### 3.4.1.2 迟滞

该迟滞设置防止传感器状态由于系统噪声而发生随机切换。迟滞参数与手指阈值一起使用，可确定传感器的状态。如果在规定的样本去抖动数量内，计数差值保持大于手指阈值与迟滞值之和，则传感器状态将从 OFF 切换到 ON。去抖动如果计数差值小于手指阈值与迟滞值之和，则传感器状态将从 ON 切换到 OFF。公式 4 显示了迟滞函数。

$$\text{if } \text{DifferenceCount} \geq \text{FingerThreshold} + \text{Hysteresis}, \text{SensorState} = \text{ON}$$

$$\text{if } \text{DifferenceCount} \leq \text{FingerThreshold} - \text{Hysteresis}, \text{SensorState} = \text{OFF} \quad \text{公式 4}$$

更多有关建议值的信息，请参见[设置高级参数](#)。

#### 3.4.1.3 去抖动

去抖动参数防止原始计数尖峰脉冲将传感器状态从 OFF 更改为 ON。对于传感器状态从 OFF 到 ON 的跃变而言，计数差值必须保持在指定的采样数内大于手指阈值与迟滞量之和。

取值范围介于 1 到 255 之间。如果将该参数设置为 1，则不提供去抖动。

更多有关建议值的信息，请参见[设置高级参数](#)。

#### 3.4.1.4 基准线更新阈值

如上面所述，基准线变量会追踪原始计数中的渐进变化。也就是说，基准线变量保持数字低通滤波器的输出，而输入原始计数值也馈送到此滤波器中。基准线更新阈值参数用于调整该低通滤波器的时间常量。

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[设置高级参数](#)。

#### 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，并且它们之间的差值大于该阈值时，基准线更新算法将暂停。

对于滑条传感器而言，当计数差值大于噪声阈值时，质心计算将被暂停。

取值范围为 3 到 255。对于正确的用户模块设置，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[设置高级参数](#)。

#### 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阈值时，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[设置高级参数](#)。

#### 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。对于已指定的采样数量而言，如果采样计数值小于基值与负噪声阈值之差，则将基准线值设置为新的原始计数值。它计算复位基准线时所需要的异常低采样数量。它用来修正启动时手指在传感器上触摸的情况。

取值范围为 0 到 255。

更多有关建议值的信息，请参见[设置高级参数](#)。

#### 3.4.1.8 传感器自动复位

通过使能传感器自动复位性能可以防止传感器在无限的时间周期内进入 ON 状态。该参数用于确定是否始终更新基准线，或者仅确定计数差值低于噪声阈值参数的时间。

使能传感器自动复位时，即使计数差值大于噪声阈值参数，也始终能够更新基准线。此设置限制了传感器被连续触摸时可报告 ON 状态的最大持续时间（典型值为 5 至 10 秒），但可在无手指触摸传感器而原始计数突然上升的情况下，阻止传感器始终报告为 ON 状态。系统中的电气损坏或在某个金属物体靠近传感器时都会导致原始计数突然上升的现象。

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可以更新基准线。因此，如果触摸该传感器，它将处于 ON 状态。

可能项为‘使能’和‘禁用’。更多有关建议设置的信息，请参见[设置高级参数](#)。

### 3.4.2 CSD 用户模块低级参数

除高级参数外，CSD 用户模块还具有多个低级参数。这些参数是 CSD 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

$I_{DAC}$  参数设置电容测量范围。值越高，范围越大。调整  $I_{DAC}$  值，以获取大约为整个范围的 50 到 70% 的原始计数。可以使用用户模块 API `CSD_SetIdacValue()`，在运行状态下更改此参数。

取值范围介于 1 到 255 之间。

#### 3.4.2.2 分辨率

该参数确定扫描分辨率（以“位”为单位）。N 位的扫描分辨率最大原始计数为  $2^N - 1$ 。增大分辨率可提高灵敏度，但会增加扫描时间。

这些值可能为 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 ( $\mu$ s)			
	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

#### 3.4.2.3 扫描速度

该参数用于设置传感器扫描速度。虽然较快的扫描速度可以提供优化的响应时间，但较慢的扫描速度也具有下列优势：

- 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、正常和慢速。

#### 3.4.2.4 屏蔽电极输出

屏蔽电极用于降低寄生电容。该参数用于选择将屏蔽电极输出路由至何处。

可选值为 P0[7] 和 P1[2]。

#### 3.4.2.5 预充电电源

此参数选择预充电开关的时钟源。

可选项为 PRS 和预分频器。大多数情况下，使用 PRS 源可获得更好的抗 EMI 能力和更低的辐射。

### 3.4.2.6 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。

### 3.4.2.7 PRS 分辨率

此参数更改 PRS 序列的长度。

可选项为 8 位和 12 位。对应的序列长度为 511 和 2047 输入时钟周期。如果 12 位设置无法提供较好的信噪比，请使用 8 位设置。

### 3.4.2.8 自动校准

使能“自动校准”（Autocalibration）之后，原始计数值将校准为最大值（ $2^N - 1$ ）的百分比，其中 N 是分辨率。“自动校准”（Autocalibration）会覆盖器件编辑器中的设置。

如果禁用“自动校准”（Autocalibration），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、预充电源和  $V_{REF}$  参数。

“自动校准”（Autocalibration）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。自动校准参数用于部分提高功能配置。

### 3.4.2.9 $I_{DAC}$ 范围

$I_{DAC}$  范围参数用于标度  $I_{DAC}$  电流输出量程。例如，选择 2x 会将  $I_{DAC}$  的输出范围为范围的两倍。

可选值为 1x、2x、4x 和 8x。

## 3.4.3 CSA\_EMC 用户模式低级参数

除高级参数外，CSA\_EMC 用户模块还具有多个低级参数。这些参数是 CSA\_EMC 感应方法的特定参数，用于确定如何从传感器中采集原始计数数据。

### 3.4.3.1 建立时间

“建立时间”参数控制软件延迟，此延迟允许  $C_{MOD}$  电容上的电压趋于稳定。每个循环的每个迭代有 9 个 CPU 周期。根据公式 5 选择建立时间。

$$Settling\ Time \geq 10 \times R_{SERIES} \times C_P$$

公式 5

其中：

$R_{SERIES}$  = 400  $\Omega$  + 端口引脚与传感器之间的串联电阻（典型值为 560  $\Omega$ ）

$C_P$  = 传感器基本电容值

可选值范围为 2 到 255。

### 3.4.3.2 Freq Num

该参数通过一项拥有专利的 EMC 提升技术来提高 EMC 的性能。Freq Num = 1 对应于标准扫描算法，Freq Num = 3 将使能高级算法。使能高级扫描算法会将扫描时间和 RAM 消耗增加三倍。

可选值为 1（标准扫描算法）和 3（高级算法）。

### 3.4.3.3 扩频

该参数通过基于固件的扩频技术在扫描时随机改变时钟值，从而提高 EMC 性能。Freq Num 值为 1 时，将使能扩频。

可选值为 1（使能）和 3（禁用）。

#### 3.4.3.4 原始数据中值滤波器

该中值滤波器将查看传感器的最近三个样本并报告中值。它用于消除短的噪音毛刺。该滤波器会生成一个样本的延迟。由于这种延迟以及 RAM 消耗，通常建议不要启用该滤波器。使能该滤波器会消耗（传感器数量  $\times$  2  $\times$  Freq Num）字节的 RAM 以及 100 字节的闪存。默认情况下，它被禁用。

可选项为“使能”和“禁用”。

#### 3.4.3.5 原始数据 IIR 滤波器

此无限脉冲响应（IIR）滤波器可减少转换结果（原始计数）中的噪声。对原始数据进行滤波会比对 XY 坐标进行滤波更有效，但需要消耗更多 RAM。使能此滤波器会额外消耗 100 字节的闪存。默认情况下，它被禁用。默认 IIR 系数为 0.5。

可选项为“使能”和“禁用”。

#### 3.4.3.6 原始数据 IIR 滤波系数

这是原始计数 IIR 滤波器的系数。

可选值为 2（ $\frac{1}{2}$  上一个样本 +  $\frac{1}{2}$  当前样本）和 4（ $\frac{3}{4}$  上一个样本 +  $\frac{1}{4}$  当前样本）。

#### 3.4.3.7 时钟

时钟参数可用于增加传感器的有效电阻值。如果传感器面积较大，有效电阻可能过高，导致无法自动校准开关电容电路。较大接近感应传感器的灵敏度可能会下降。在该情况下，稳定电压会远远低于比较器阈值。设置较大的内部主振荡器（IMO）分频器可增加有效电阻，以补偿高电容值。

可选值为 IMO、IMO/2、IMO/4 和 IMO/8。

### 3.4.4 SmartSense 用户模块参数

图 3-7. PSoC Designer SmartSense 参数

高级  
低级

Parameters - SmartSense_1	
Name	SmartSense
User Module	SmartSense
Version	1.00
Sensors Autoreset	Disabled
Debounce	3
ShieldElectrodeOut	None
Sensor Sensitivity	0.1 pF

**Name**  
Indicates the name used to identify this User Module instance

#### 3.4.4.1 屏蔽电极输出

屏蔽电极用于降低寄生电容。该参数用于选择将屏蔽电极输出路由至何处。  
可选项为 P0[7]和 P1[2]。

#### 3.4.4.2 传感器灵敏度

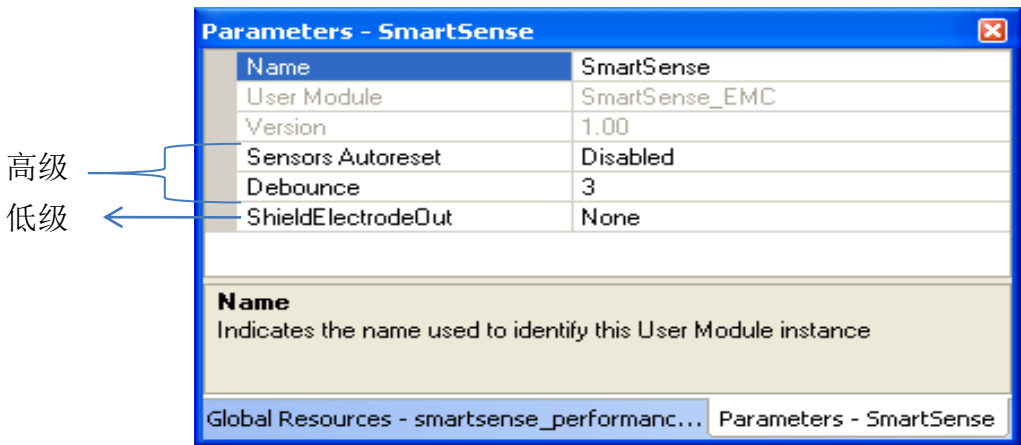
该参数用于提高或降低传感器的灵敏度。  
可选值为 0.1 pF、0.2 pF、0.3 pF 和 0.4 pF。

#### 3.4.4.3 MultiChart 用于监控 CapSense 用户模块参数

调校 CapSense 系统需要您监控 CapSense 用户模块的全局阵列。MultiChart 应用有助于更轻松地监控该参数。关于如何使用 MultiChart 的更多细节，请见应用笔记 [AN2397](#)。

### 3.4.5 SmartSense\_EMC 用户模块参数

图 3-8. PSoC Designer SmartSense\_EMC 参数



#### 3.4.5.1 屏蔽电极输出

屏蔽电极用于降低寄生电容。该参数用于选择将屏蔽电极输出路由至何处。

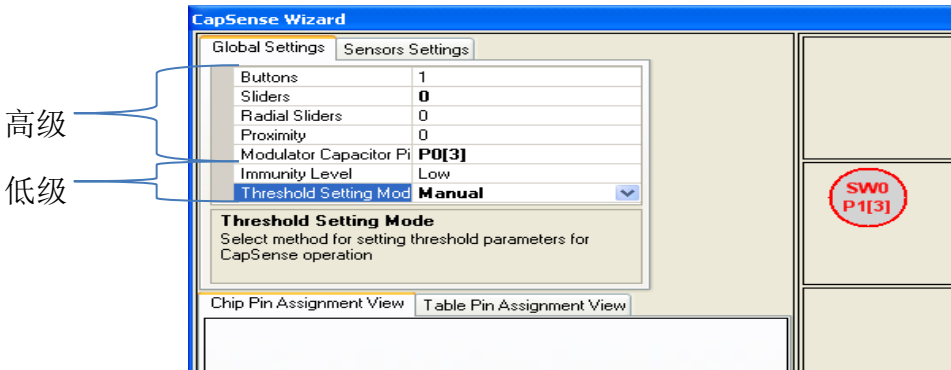
可选项为 P0[7]和 P1[2]。

#### 3.4.5.2 抗噪声级别

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 3-9. PSoC Designer SmartSense\_EMC 全局设置

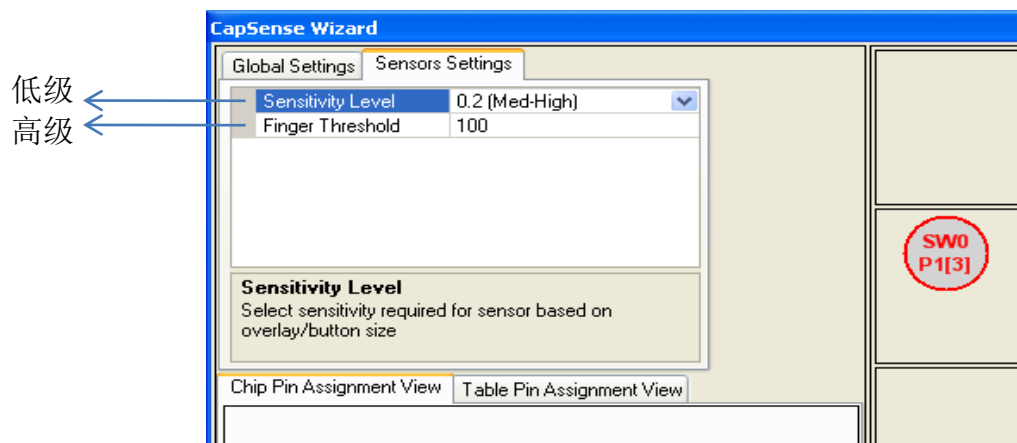


#### 3.4.5.3 阈值设置模式

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\_EMC 用户模式自动为传感器设置阈值。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 3-10. PSoC Designer SmartSense\_EMC 传感器设置



#### 3.4.5.4 传感器灵敏度

该参数用于提高或降低传感器的灵敏度。

可选值为 0.1 pF、0.2 pF、0.3 pF 和 0.4 pF。

## 4. 使用用户模块对 CapSense 性能进行调校



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[设计注意事项](#)。调校是确定最佳参数设置的过程，它的目的为保证传感器操作更强健和可靠。

### 4.1 基本注意事项

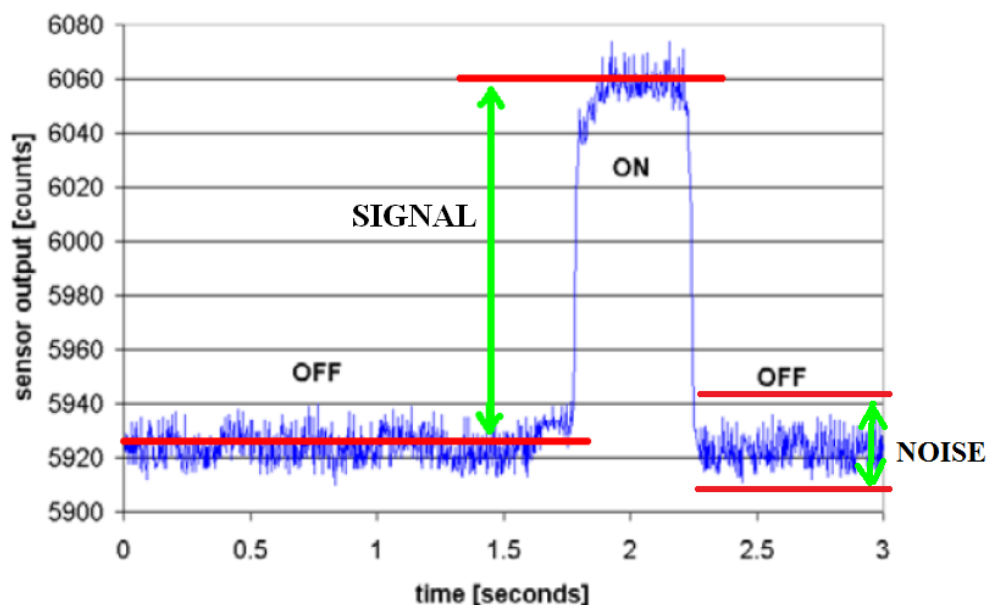
#### 4.1.1 信号、噪声和信噪比（SNR）

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。可以使用一个称为信噪比（SNR）的数值，将 CapSense 信号与 CapSense 噪声进行比较。在讨论 CapSense SNR 的含义时，首先需要定义触摸感应环境中信号和噪声的概念。

##### 4.1.1.1 CapSense 信号

CapSense 信号表示在手指触摸传感器时，传感器响应的变化，如图 4-1。传感器输出是一个带有表示传感器电容值的数字计数器。在该示例中，在手指没有触摸传感器时，平均输出为 5925 次计数。当手指触摸传感器时，平均输出增至 6060 次计数。由于 CapSense 信号跟踪手指产生的计数变化，因此信号 =  $6060 - 5925 = 135$  次计数。

图 4-1. CapSense 信号与噪声



使用用户模块对 CapSense 性能进行调校

### 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-峰的变动，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 次计数和最小为 5912 次计数。因为噪声为该波形最大计数与最小计数之间的差，所以噪声 = 5938 – 5912 = 26 次计数。

### 4.1.1.3 CapSense 信噪比

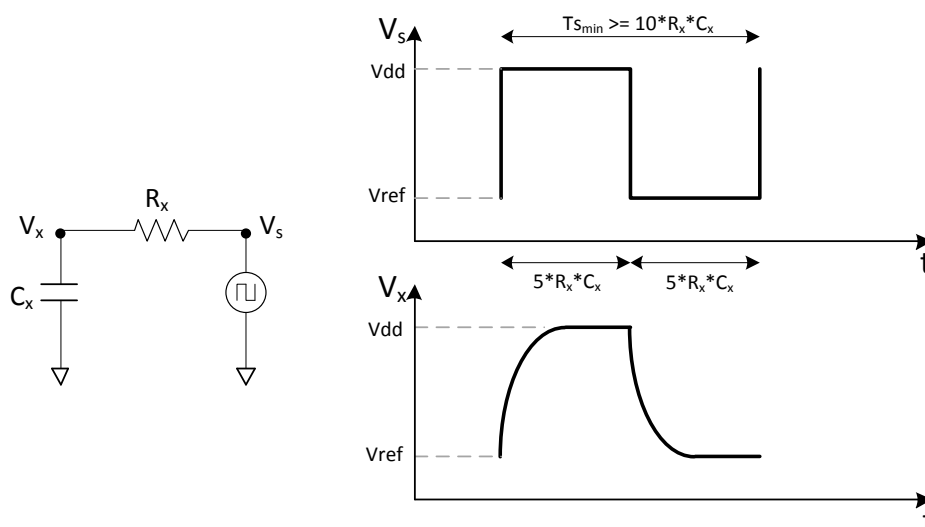
CapSense 信噪比是信号和噪声间简单的比例。继续该示例，如果信号为 135 次计数，噪声为 26 次计数，那么 SNR 将为 135:26，这样可使 SNR 降低到 5.2:1。CapSense 的最小建议 SNR 为 5:1，表示信号比噪声大 5 倍。滤波器通常在固件中实现，以降低噪声。更多有关信息，请参见[软件滤波](#)。

## 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之间进行切换，该频率由 CSA\_EMC 用户模块中的时钟和 CSD 用户模块中的预充电时钟设置。

充电/放电路径包含串联电阻，这降低了电荷转移的速度。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和串联电阻相关，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 时间常数兼容的级别。您应该将每个转变的时间周期设置为 5RC，每个周期有两次切换（一次充电、一次放电）。最小时间周期和最大频率的计算公式如下：

$$T_{smin} = 10 \times R_X C_X \quad \text{公式 6}$$

$$f_{smax} = \frac{1}{10 \times R_X C_X} \quad \text{公式 7}$$

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

$$R_X = 1.4 \text{ k}\Omega$$

$$C_X = 24 \text{ pF}$$

在本示例中，时间常数值和前端最大切换频率为：

$$T_{smin} = 0.34 \text{ }\mu\text{s}$$

$$f_{smax} = 3 \text{ MHz}$$

### 4.1.3 基准线更新阈值验证的重要性

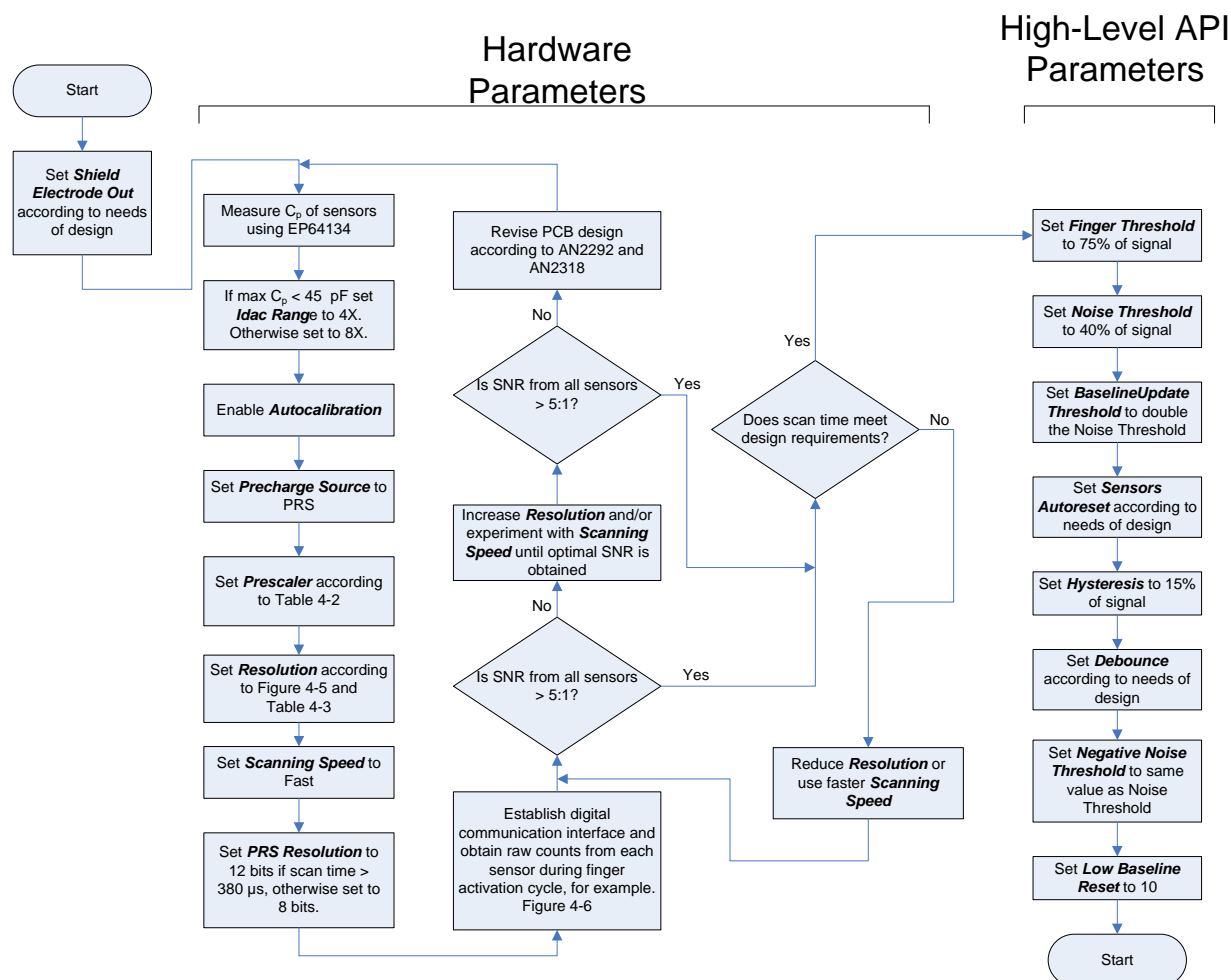
温度和湿度都会引起平均计数随时间浮动。基准线是 CapSense 测量值的参考计数量，这对于环境影响补偿而言起着重要作用。高级决策（例如，手指触摸和手指离开状态）是基于基准线建立的参考级别。由于每个传感器具有相应的独特寄生电容，因此所有电容式传感器均有各自的基准线。

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。确保更新速率与预期应用相互匹配。如果更新速率过快，那么，基准线将补偿手指产生的所有变量，并不予检测手指移动的情况。如果更新速率过慢，那么，相对较慢的环境变化可能导致手指出错。开发期间，应验证基准线更新阈值的设置情况。

## 4.2 调校 CSA\_EMC 用户模块

图 4-3 是展示调校 CSA\_EMC 参数过程的流程图。CSA\_EMC 用户模块分两大类：低级（硬件）参数和高级参数。这两种类别的参数以不同的方式影响电容式感应系统的性能。然而，每个传感器的灵敏度之间都有互补关系，由硬件参数设定和若干高级参数设置决定。当硬件参数改变时，必须确保相应的高级参数也相应作出调整。调校 CSA\_EMC 用户模块参数应该先从硬件参数开始。

图 4-3. CSA\_EMC 用户模块参数调校流程图



### 4.3 CSA\_EMC 的建议 C<sub>INT</sub> 值

使用 1.2 nF 的建议值 C<sub>INT</sub> 开始调校过程。在调校过程中，如果发现传感器信号不足以达到 5:1 的信噪比，则可以增加 C<sub>INT</sub>。C<sub>INT</sub> 的最大建议值为 5.6 nF。推荐使用 X7R 或者 NPO 类型的电容，以保证 C<sub>INT</sub> 在不同的温度条件下保持稳定状态，同时电容的电压不应该低于 5 V。

### 4.4 测量传感器 C<sub>P</sub>

调校程序的第一步是测量传感器寄生电容（C<sub>P</sub>）。测量的分步步骤如下：

1. 将 CPU\_CLK 的值设置为等于 SysClk/2。
2. 将 Clock（时钟）的值设置为等于 IMO/8。
3. 将 Settling Time（建立时间）设置为 255。
4. 回读由特定传感器算法设置的 I<sub>DAC</sub> 代码。读值将被存储在 CSA\_EMC\_baDACCodeBaseline[] 阵列内
5. 测量 I<sub>DAC</sub> 代码相应的 I<sub>DAC</sub> 电流。

使用用户模块对 CapSense 性能进行调校

使用以下代码创建 PSoC Designer 项目。该代码将 IDAC 路由到端口引脚 P1[4]。

```
//configure P1[4] to HI-Z
PRT1DM0 &= ~ (1<<4);
PRT1DM1 |= (1<<4);
//connect P1[4] to analog mux bus
MUX_CR1 = (1<<4);
// set IDAC to read back IDAC Code
IDAC_D = <IDAC CODE>
// turn ON IDAC
CS_CR2 = 0xD0;
```

将电流仪放在引脚 P1[4]与地之间，然后测量电流。它的值为  $I_{\text{MEASURED}}$ 。

6. 使用公式  $C_P = I_{\text{MEASURED}} / ((I_{\text{MO}}/8) * 1.3)$  计算  $C_P$

传感器  $C_P$  的数值也可通过 LCR 测量仪测量得到。将 LCR 仪表的一个终端连接到传感器引脚上，另一个终端连接到 GND，以测量  $C_P$ 。

## 4.5 估算 CSA\_EMC 时钟

表 4-1 展示了所推荐的预充电时钟频率，它是基于传感器  $C_P$  的函数。设定 CSA\_EMC 时钟以获取推荐预充电时钟频率。预充电时钟频率由所选 IMO、CSA\_EMC 时钟设置和传感器的  $C_P$  决定。

确保预充电时钟频率不超过推荐值。

表 4-1. 基于  $C_P$  和 IMO 的 CSA\_EMC 时钟设置

$C_P$ (pF)	CSA_EMC 时钟		
	IMO = 24 MHz	IMO = 12 MHz	IMO = 6 MHz
< 5	IMO/2	IMO	IMO
5 到 10	IMO/4	IMO/2	IMO
10 到 15	IMO/4	IMO/4	IMO/2
15 到 20	IMO/4	IMO/4	IMO/2
20 到 25	IMO/16	IMO/8	IMO/4
25 到 30	IMO/16	IMO/8	IMO/4
30 到 35	IMO/16	IMO/8	IMO/4
35 到 40	IMO/16	IMO/8	IMO/4
40 到 45	IMO/16	IMO/8	IMO/4
45 到 50	IMO/16	IMO/8	IMO/8

## 4.6 设置建立时间

使用公式 8 估计出建立时间参数的最小值。

$$\text{Settling Time} = \frac{(5 * C_{\text{int}})}{\left( \text{Clock} * C_P * 25 * \left( \frac{1}{F_{\text{cpu}}} \right) \right)} \quad \text{公式 8}$$

其中：

- $C_{\text{INT}}$  = 积分电容器的值

使用用户模块对 CapSense 性能进行调校

- 时钟 = 预充电时钟频率 (CSA\_EMC 时钟)
- $C_P$  = 传感器的寄生电容值
- $F_{CPU}$  = CPU 时钟频率

## 4.7 监测 CapSense 数据

请参考 [CapSense 数据查看工具](#)。

## 4.8 增大信噪比的方法

此部分叙述了增大信噪比的方法。

### 4.8.1 降低噪声

增大信噪比的方法是减小噪声计数。要减小噪声计数，可以使用以下任一种方法：

- 使用软件滤波器 — 更多细节，请参见 [软件滤波](#)。
- 使能扩频功能 — 更多细节，请参见 [扩频](#)。
- 增加抗噪级别 — 更多细节，请参见 [Freq Num](#)。

### 4.8.2 提高信号

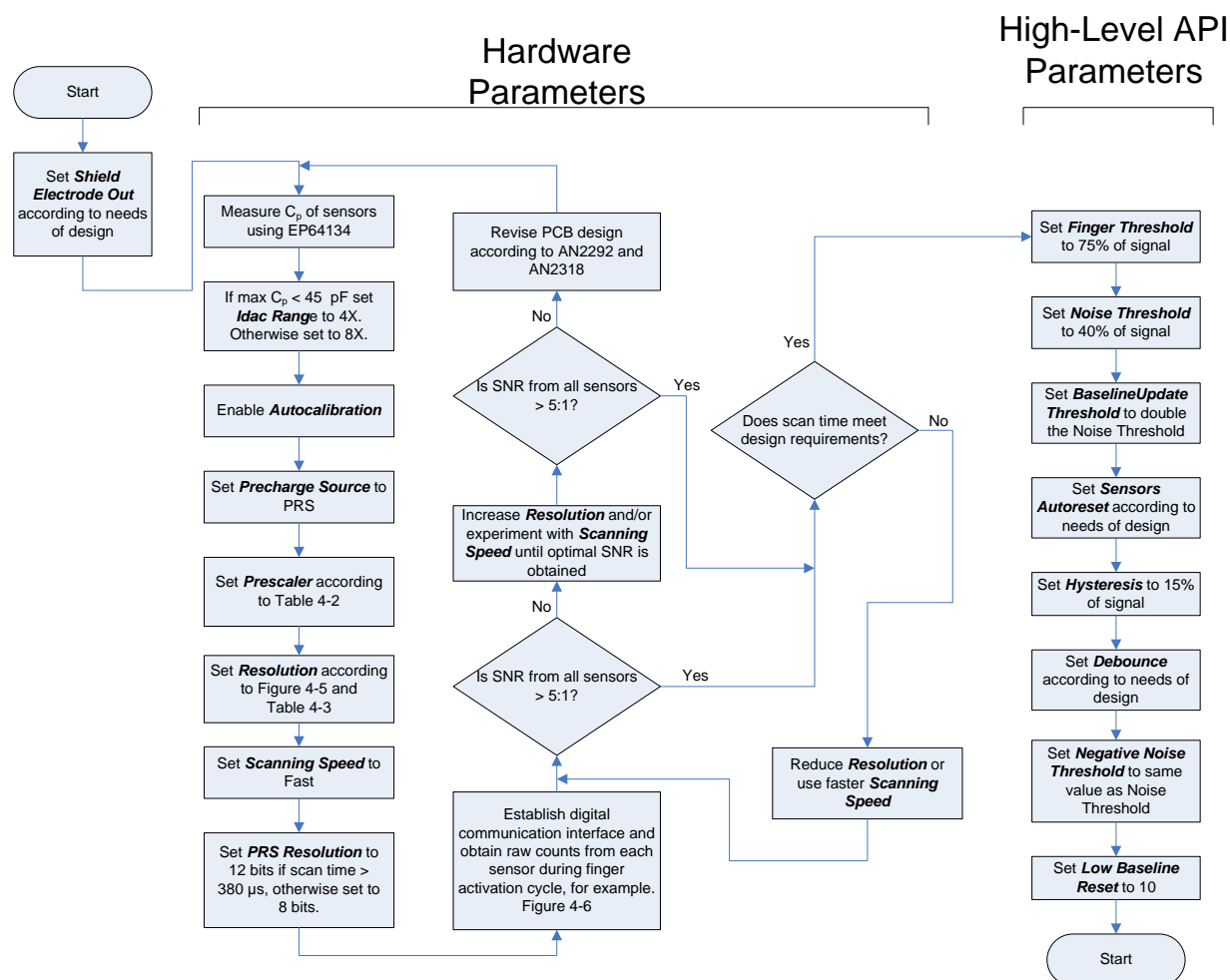
通过以下两种方法增加信号可以提高信噪比：

- 增加由宏 `CSA_EMC_BASELINE` 定义的值。该宏位于 `CSA_EMC.inc` 文件内。默认情况下，将 `0x0800` 值分配给该宏。
- 增加 `CINT` 电容的值

## 4.9 调校 CSD 用户模块

图 4-4 显示的是 CSD UM 参数调校过程的流程图。CSD UM 参数可以分为两大类：低级（硬件）参数和高级 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性能。然而，各个传感器的灵敏性之间存在互补关系，这由硬件参数设置和许多高级参数设置所决定。更改任何硬件参数时，您必须考虑这种情况，由此确保相应地调整相对的高级参数。调校 CSD 用户模块参数应始终从硬件参数开始。

图 4-4. 调校 CSD 用户模块



硬件参数配置硬件，CSD 方法使用该硬件将每个传感器的物理电容值转换为数字代码。该部分描述了这些参数，并为如何根据硬件特征和其他参数调校提供指导。

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。在设计中，传感器的寄生电容的总值 ( $C_P$ ) 和/或传感器的灵敏度在一个较大范围内浮动，那么全局硬件参数设置可能会不适合所有传感器。在这些情况下，通过调用 `SetIdacValue()`、`SetPrescaler()` 和 `SetScanMode()` API 函数，然后调用 `ScanSensor()` API 函数可以为每个传感器设置相应的硬件参数。

表 4-2 和表 4-4 根据传感器的  $C_P$  提供了几个关键硬件参数的建议调校值。 $C_P$  的值是由 PSoC 的特征、PCB 布局和产品组装板附近元件决定的。因上述原因， $C_P$  值必须在系统最终组装状态原位测量，即：在与系统提供服务时一样，具有相同的外围或覆盖层。测量  $C_P$  值的最佳方法是使用 CapSense 代码示例设计指南中提供的“使用 CY8C20xx6A CapSense 控制器测量绝对传感器电容”代码示例。此项目使用 PSoC 测量系统中每个传感器的绝对电容值，因此会考虑影响  $C_P$  的所有因素。请参考代码示例相关的文档，以查阅有关安装和使用的步骤。

### 4.9.1 CSD 的建议 $C_{MOD}$ 值

基于 CSD 的设计的建议  $C_{MOD}$  值为 2.2 nF。推荐使用 X7R 或者 NPO 类型的电容，以保证  $C_{INT}$  在不同的温度条件下保持稳定状态，同时电容的电压不应低于 5 V。

### 4.9.2 ShieldElectrodeOut（屏蔽电极输出）

使能设计中的屏蔽电极输出。

### 4.9.3 I<sub>DAC</sub> 范围

对于小于 45 pF 的最大传感器 C<sub>P</sub> 项目，可以使用 4X；否则，则使用 8X。

### 4.9.4 自动校准

在 CY8C20xx6A CSD 设计中，应将自动校准始终设置为使能状态。如果合理设置预分频器并将 C<sub>MOD</sub> 设置为所建议的大小，则自动校准算法可以成功设置 I<sub>DAC</sub>。

### 4.9.5 I<sub>DAC</sub> 值

禁用自动校准后，此参数将决定 I<sub>DAC</sub> 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，递增该参数会降低原始计数基准线，反之亦然。

### 4.9.6 预充电源

该参数用于选择传感器切换时钟源。可用选项为预分频器（通过分频器使用 IMO）或 PRS（可通过任意发生器传输被分频的 IMO 时钟），这些传感器切换时钟源提供了扩频时钟。PRS 提供了高级抗噪能力和较低的噪音辐射，因而它也成为默认的预充电源推荐设置。在某些情况下，预分频器预充电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小，并且也不能明显看出是否有利于前面提到的 PRS。

### 4.9.7 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定的预充电源、IMO 和扫描传感器的 C<sub>P</sub> 值。表 4-2 显示的是基于这些参数的预分频器推荐设置。

表 4-2. 根据预充电源、IMO 和 C<sub>P</sub> 的预分频器设置

C <sub>P</sub> (pF)	预充电源 = PRS			预充电源 = 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
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电源、预分频器和 C<sub>P</sub> 的组合。

## 4.9.8 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、加大信噪比，并延长降噪的扫描时间。扫描分辨率为  $n$  时，最大原始计数值（所有范围内）为  $2^n - 1$ 。表 4-3 显示的是基于  $C_P$  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量。 $C_F$  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 4-5 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 4-5. 基于覆盖层厚度和圆形传感器直径的手指电容值 ( $C_F$ )

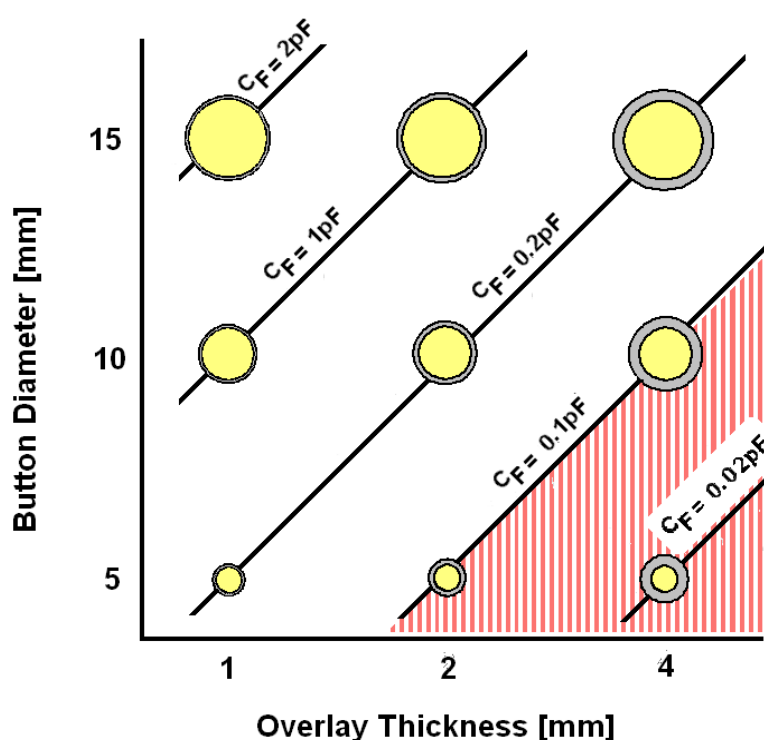


表 4-3. 基于手指电容值和  $C_P$  的分辨率设置

$C_P$ (pF)	$C_F = 0.1$ pF	$C_F = 0.2$ pF	$C_F = 0.4$ pF	$C_F = 0.8$ pF
< 6	12	11	10	9
7 到 12	13	12	11	10
13 到 24	14	13	12	11
25 到 48	15	14	13	12
> 49	16	15	14	13

## 4.9.9 扫描速度

该参数控制各个扫描结果的 LSB 集成时间。扫描速度选项包括：超快、快速、正常和慢速。建议将初始值选择为“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 4-4 显示的是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 4-4. 单传感器在不同分辨率和扫描速度下的扫描时间 (μs)

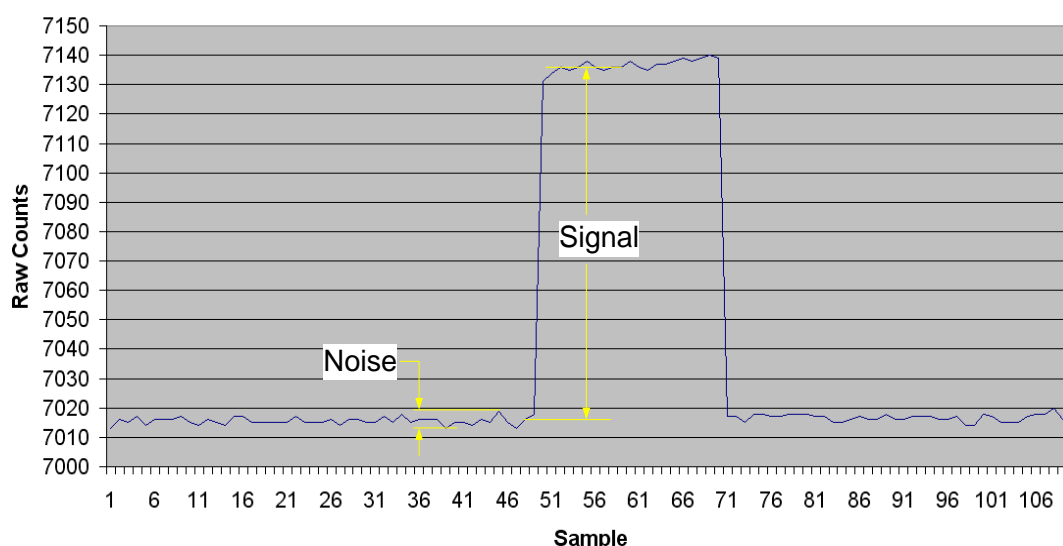
分辨率 (位)	扫描速度			
	超快速度	快速	正常	慢速
9	57	78	125	205
10	78	125	205	380
11	125	205	380	720
12	205	380	720	1400
13	380	720	1400	2800
14	720	1400	2800	5600
15	1400	2800	5600	11000
16	2800	5600	11000	22000

### 4.9.10 高级 API 参数

高级 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。要想为这些参数确定一个合适的取值，您必须在系统中建立起一个数字通信接口，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。数据分别存储在如下三个阵列：CSD\_waSnsBaseline[]、CSD\_waSnsResult[] 和 CSD\_waSnsDiff[]。如该数据所示，高级 API 参数的设置主要取决于环境噪声和手指信号的强度。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，用于设置这些参数的基础数据必须取自最终装配状态并与未来实际应用时的 EMI 环境相同。

图 4-6 显示了传感器在一个手指激活周期中所获取的典型原始计数值，即传感器被激活后再取消激活。叠加在数据上的标签说明了如何根据原始数据计算噪声和信号。在适当的情况下，下面的高级参数描述包括了有关如何根据噪声和信号值进行设置每参数的信息。根据 CapSense 设计的最佳做法，为了确保 CapSense 系统操作的健壮性，信噪比（SNR）至少要为 5:1。如果信噪比低于 5:1，则需要调整硬件参数和/或根据 CapSense 入门更改 PCB 布局，以保证信噪比最少为 5:1。

图 4-6. 传感器在一个手指激活周期中的典型原始计数值



### 4.9.11 设置高级参数

为了获取最佳参数设置，建议采用以下初始值：

- **手指阈值：**传感器处于 ON 状态时设置为原始计数的 75%
- **噪声阈值：**传感器为 OFF 时将其设置为原始计数的 40%
- **负噪声阈值：**设置为（噪声阈值/2）
- **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，将其设置为原始计数的 15%
- **低基准线复位：**将其设置为 50
- **传感器自动复位：**根据设计要求
- **去抖动：**根据设计要求

## 4.10 使用 SmartSense 用户模块

只要传感器寄生电容值在 5 pF 到 45 pF 之间，且触摸电容值最小为 0.1 pF 时，便能够使用 SmartSense 来创建一个无需调校的 CapSense 设计。可使用 PSoC Designer 5.1 中的 SmartSense 用户模块创建一个 SmartSense 设计。本节也介绍了如何将一个现有的 CSD CapSense 设计移植到 SmartSense。

### 4.10.1 SmartSense 指南

在应用中使用 SmartSense 用户模块时，请遵循以下指南：

- SmartSense 要求电容式用户界面设计遵循本设计指南之前章节中所记录的布局 and 系统设计最佳实践。
- 所有 CSD 用户模块参数（例如：IdAC 值、预分频器周期、时钟分频器、扫描速度、分辨率等）由 SmartSense 用户模块在运行时确定。除非您已经了解使用 API 修改固件中的 CSD 参数对您的设计产生的具体影响，否则请勿进行该修改操作。
- 要想将现有的 CSD 设计移植到 SmartSense 中，
  - ☐ 请确保先从程序中移除所有设置或修改 CSD 参数的 API。
  - ☐ 请确保在环境和 PCB 生产过程发生变化时，设计中所有 CapSense 传感器的寄生电容值都处于 5 pF 到 45 pF 的范围内。
  - ☐ 请确保建议将 C<sub>MOD</sub> 电容（X7R，2.2 nF，额定电压超过 5 V）连接到用户模块向导中已选的 C<sub>MOD</sub> 端口引脚上。

### 4.10.2 理解差异

SmartSense 用户模块和标准 CSD 用户模块间的区别包括：

- SmartSense 用户模块所支持的 API 与标准 CSD 用户模块所支持的一样。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 用户模块在运行时被自动设置。
- C<sub>MOD</sub> 的电容值被限制为 2.2 nF。在所有 CapSense 应用中，建议使用额定电压高于 5 V 的 X7R 电容。
- SmartSense 算法将每个传感器的信噪比维持在 5:1 到 11:1 之间，这样可确保在使 Capsense 获得最佳性能的同时，它仍能稳定运行。
- 根据传感器的寄生电容，算法将 SmartSense 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。

### 4.10.3 SmartSense 的建议 $C_{MOD}$ 值

基于 SmartSense 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  $C_{INT}$  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。

### 4.10.4 SmartSense 用户模块参数

在该用户模块中，只需要设置四个参数。它们为：

- Sensors Autoreset（传感器自动复位）
- Debounce（去抖动）
- Modulator Capacitor Pin（调制器电容引脚）
- Sensitivity Level（灵敏度级别）

#### 4.10.4.1 Sensors Autoreset（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。设置为“使能”时，基准线随时会被更新。虽然该设置限制了传感器运行的最长时间（通常是 5 至 10 秒），但是可以防止下述情况的发生：由于某种系统故障，虽然未触摸传感器，但原始计数却突然上升，从而导致传感器一直运行。

#### 4.10.4.2 Debounce（去抖动）

去抖动参数为传感器的有效转换添加去抖动计数。对于从无效状态转换到有效状态的传感器，手指信号应连续显示在传感器上去抖计数次扫描。该参数影响所有类似传感器。

#### 4.10.4.3 Modulator Capacitor Pin（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：**必须使用一个 2.2 nF 的外部电容，这样 SmartSense 才能正常工作。

#### 4.10.4.4 Sensitivity Level（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），传感器发出的信号便越强。为了满足要求，覆盖层越厚，传感器信号要更强。可用的灵敏度选项包括：“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 用户模块需要更长的传感器扫描时间。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，前者消耗更久的扫描时间。

最佳调校方法是查找传感器的最高灵敏度值，用于生成所需要的 5:1 信噪比。可以使用最大的灵敏度值（0.4 pF）进行调校，并按要求降低该值，以满足 5:1 的信噪比。

### 4.10.5 SmartSense\_EMC 用户模块的专用指南

适用于 SmartSense 用户模块的所有指南都适用于 SmartSense\_EMC 用户模块。有关 CapSense 设计和基于 SmartSense 设计的通用说明，请参阅《CapSense 入门指南》。本节介绍 SmartSense\_EMC 用户模块的几个重要方面。

#### 4.10.5.1 传感器扫描时间、响应时间和存储器的使用情况

当使用 SmartSense\_EMC 用户模块运行传感器时，传感器的扫描时间、传感器的响应时间以及 RAM 存储器大小均取决于用户模块中所选择的抗噪模式。

- 抗噪级别被设置为“中”时，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，传感器响应时间比设置为“低”时大两倍。
- 为实现强大的电磁兼容算法，SmartSense\_EMC 用户模块使用 RAM 存储器。因此，最高抗噪音模式（“高”）所需要的 RAM 存储器大约比模式为“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。

#### 4.10.5.2 IMO 容差和时间严格的任务

SmartSense\_EMC 使能部分的 IMO 容差为+5%和-20%。

- 使用严格时间算法和逻辑时，必须考虑 IMO 容差，以确保固件的逻辑或避免破坏算法。
- 如果项目使用中断，则分析中断延迟、ISR 执行时间等其他操作时，必须考虑 IMO 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定性。

#### 4.10.5.3 I<sup>2</sup>C 工作速度

I<sup>2</sup>C 接口工作频率的最大为 SmartSense\_EMC 使能部分中用户模块实际工作频率的 80%。该限制由 IMO 容差的 20% 导致。

- 这意味着在 I<sup>2</sup>C 用户模块中选择 400 kHz 的时钟频率时，I<sup>2</sup>C 接口的最大工作频率为 320 kHz。同样，在 I<sup>2</sup>C 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 I<sup>2</sup>C 从设备接口时，主设备时钟需要运行于前面所提到的降频范围内。否则会导致数据损坏、I<sup>2</sup>C 总线连接或与 I<sup>2</sup>C 用户模块性能不一致。
- 使用 I<sup>2</sup>C 主设备模块只影响该接口的吞吐量。

### 4.10.6 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 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 用户模块的灵敏度水平。

可以使用公式 9 和下表查找传感器的扫描时间。

$$\text{Scan time} = \text{Sampling time (ST)} + \text{Processing time (PT)}$$

公式 9

下表显示的是在不同 IMO 和灵敏度水平下的采样时间值。

表 4-5. 传感器 IMO = 24 MHz 时的采样时间

灵敏度 = 0.2 pF		灵敏度 = 0.3 pF		灵敏度 = 0.4 pF	
C <sub>P</sub> (pF)	ST (μs)	C <sub>P</sub> (pF)	ST (μs)	C <sub>P</sub> (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

使用用户模块对 CapSense 性能进行调校

表 4-6. 传感器 IMO = 12 MHz 时的采样时间

灵敏度 = 0.2 pF		灵敏度 = 0.3 pF		灵敏度 = 0.4 pF	
C <sub>P</sub> (pF)	ST (μs)	C <sub>P</sub> (pF)	ST (μs)	C <sub>P</sub> (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 4-7. 传感器 IMO = 6 MHz 时的采样时间

灵敏度 = 0.2 pF		灵敏度 = 0.3 pF		灵敏度 = 0.4 pF	
C <sub>P</sub> (pF)	ST (μs)	C <sub>P</sub> (pF)	ST (μs)	C <sub>P</sub> (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 4-8 显示的是不同 CPU 频率下的处理时间值。

表 4-8. 传感器处理时间

CPU CLK	处理时间 (PT)，单位为 μs
24	71
12	142
6	284
3	568

例如，如果 CapSense 系统的 IMO 频率为 24 MHz、CPU 时钟频率为 6 MHz (IMO/4) 且灵敏度水平为 0.3 pF，则寄生电容值大约为 15 pF 的传感器的扫描时间可使用前面表中的数据通过公式 9 计算得到。

上述所配置 (IMO 频率为 24 MHz，灵敏度为 0.3 pF) 的采样时间是从表 4-5 中选出的，该值为 680 μs。上面所述配置 (CPU 时钟频率为 6 MHz) 的处理时间是从表 4-8 中选出的，该值为 284 μs。

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

#### 4.10.7 SmartSense 响应时间

考虑使用标准 CSD 与典型 CapSense 扫描固件的以下应用。

- 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
- 传感器灵敏等级为 0.4 pF
- 去抖动值 = 3

使用用户模块对 CapSense 性能进行调校

根据前面的表格，每个传感器的扫描时间为 482  $\mu$ s，三个传感器的总扫描时间为 1.45 ms。以下固件示例需要 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_ScanAllSensors();
    SmartSense_UpdateAllBaselines();

    if(SmartSense_bIsAnySensorActive() )
    {
        //1ms firmware routines
    }
}
```

这意味着：当 CapSense 传感器被激活时，固件在 7.35 ms 内开启传感器状态（传感器应连续处于活动状态去抖数量的扫描时间）。这通常被称为 CapSense 系统的响应时间。

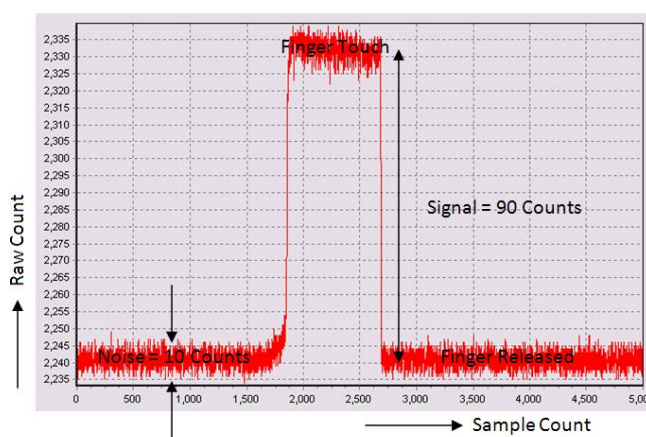
如果扫描时间随着寄生电容的变化发生相应变化，那么由于程序变化而导致的传感器寄生电容变化对响应时间会产生怎样的影响？在这种情况下，响应时间可能会增加（响应变慢）。这样可能会对传感器的性能产生负面影响。在下一节中将介绍如何设计稳定的固件。

#### 4.10.8 使用 SmartSense\_EMC 用户模块尽量降低信噪比的方法

SmartSense\_EMC 是先进的电磁器件设计，该设计以 CSD 的 SmartSense 用户模块为基础，并且不需要繁琐的调校过程。然而，使用 SmartSense\_EMC 用户模块时，有两个简单的步骤能确保设计的稳定性。

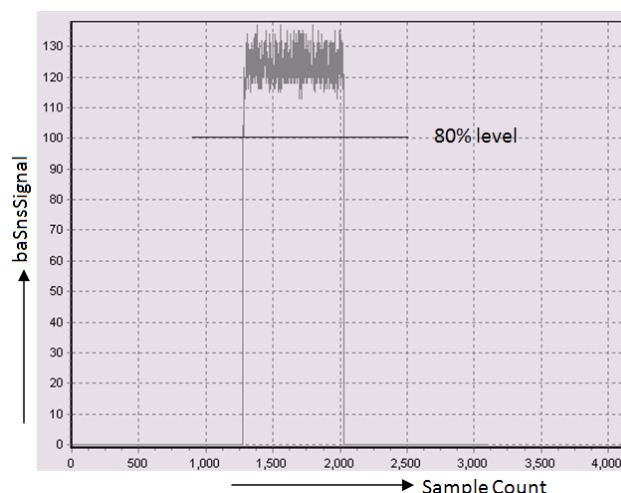
1. 建立实时监测工具来监测 CapSense 用户模块参数，从而测量传感器信号。在调校期间，可以观察到传感器原始计数（SmartSense\_EMC\_waSnsResult）、传感器标准化信号（SmartSense\_EMC\_baSnsSignal）和传感器手指阈值（SmartSense\_EMC\_baBtnFThreshold）。不要使用 LCD 或其他任何数字显示屏来监测计数，因为这些设备太慢，您无法观察到数据的动态变化。建议使用 Multi-chart 或 I<sup>2</sup>C USB Bridge Control panel。
2. 将灵敏度设置为 0.4 pF（低），并计算 SNR。图 4-7 显示的是手指触摸时的典型原始计数图。根据 CapSense 最佳实践，强大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，则使灵敏度的值下降到下一个可能值，直到获得的信噪比大于 5:1 或小于 10:1 为止。

图 4-7. 手指触摸典型传感器的原始计数图



3. 如果您在设计中使用自动手指阈值，那么该操作与上一步同时完成。如果您使用灵活手指阈值，您也应该设置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\_EMC\_baSnsSignal），并在触摸传感器时，将手指阈值设置为传感器信号的 80%。这样，便完成了该操作。图 4-8 显示了典型传感器信号和手指阈值。

图 4-8. 手指触摸典型传感器时的传感器信号



#### 4.10.9 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。查看循环执行时间同样重要（请看以下示例代码），因为它也会增加。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_ScanAllSensors();
    SmartSense_UpdateAllBaselines();

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

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

Port\_0[1]引脚上的信号时间段为 4.9 ms（由于端口引脚的切换，此时间是循环时间的两倍）。如果一个传感器的寄生电容递增到 15 pF 左右，则扫描时间将为 1.78 ms，因此，Port\_0[1]上的信号时间将为 5.6 ms。

如果该传感器的寄生电容接近 SmartSense 电容组的边界（例如：9 pF 非常接近 10 pF 边界），那么由于处理过程所发生的变化，SmartSense 可能在应用中选择临近的扫描时间。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。

根据上述内容，在实现其他功能时，固件不应该依赖于传感器的扫描时间（例如，软件 PWM、软件延迟等）。实现看门狗定时器（WDT）的程序在设置 WDT 到期时间时应该考虑到这一点。

下面显示的是一个简单的固件实现示例，说明了如何使用定时器 16 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()
{

```

使用用户模块对 CapSense 性能进行调校

```
M8C_EnableGInt;

SmartSense_Start();
SmartSense_ScanAllSensors();
SmartSense_SetDefaultFingerThresholds();

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

while( 1 )
{
    /* Scan all 3 sensors and update
    Baseline */
    SmartSense_ScanAllSensors();
    SmartSense_UpdateAllBaselines();

    /* Wait till timer expires or
    sleep here */

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

    if(CSDAUTO_bIsAnySensorActive())
    {
        //lms 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。应该根据最差情况下主循环执行的时间来选择定时器的时间。这是在最差情况下，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 电容组的界限，则应选择较高的扫描时间（使用表 4-6）来计算。

通过使用 SmartSense 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 生产程序的变化和其他变化，它避免了调校过程中存在的困难，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，并在新设计中使用 SmartSense。

主循环执行时间和 SmartSense 扫描时间都会根据程序的变化而改变。尽管不会影响 CapSense 的性能，但在利用 SmartSense 自动调校技术来实现 CapSense PLUS 应用时，固件开发人员应该考虑到这一点。

## 5. 设计注意事项



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。请认真考虑从 PCB 布局到用户界面到最终使用操作环境等各项细节，以实现既稳定又可靠的系统性能。更多有关信息，请参见 [CapSense 入门](#) 一节。

### 5.1 覆盖层选择

在 [CapSense 基本原理](#) 部分，使用公式 1 计算手指电容，如下所示：

$$C_F = \frac{\epsilon_0 \epsilon_r A}{D}$$

其中：

$\epsilon_0$  = 空气介电常数

$\epsilon_r$  = 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<sup>2</sup>)

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	击穿电压 (V/mm)	12 kV 时的覆盖层最小厚度 (mm)
空气	1200 – 2800	10
木材 – 干	3900	3
玻璃 – 普通	7900	1.5
玻璃 — 硼硅酸盐 (Pyrex®)	13,000	0.9
PMMA 塑料 (Plexiglas®)	13,000	0.9
ABS	16,000	0.8
聚碳酸酯 (Lexan®)	16,000	0.8
福米卡	18,000	0.7
FR-4	28,000	0.4
PET 薄膜 — (Mylar®)	280,000	0.04
聚酰亚胺薄膜 — (Kapton®)	290,000	0.04

传导材质不能用做覆盖层，因为它会与电场模式相干扰。出于这个原因，覆盖层不能使用包含金属颗粒的油漆。

粘合剂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作胶带卷抽取 (3M™ 编号为 467MP 和 468MP 的产品) 的。

## 5.2 ESD 保护

精密系统设计自然要有强大的抗 ESD 能力。经过考虑在最终产品（尤其是用户界面）中可能发生的接触放电，我们的设计可承受 18 kV 放电事件，并且不会对 CapSense 控制器造成任何损害。

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，将按以下顺序采用 ESD 对策：预防、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，系统设计需要确保 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机壳接地的电路板周边放置一个保护环。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。

建议使用 560  $\Omega$  的串联电阻。

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。用于 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				
Littlefuse	SP723	5 pF	2 nA	8 kV	15 kV
Vishay	VBUS05L1-DD1	0.3 pF	0.1 $\mu$ A <	$\pm$ 15 kV	$\pm$ 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。
  - CapSense 输入线的建议串联电阻值为 560  $\Omega$ 。
  - 通信线（I<sup>2</sup>C 和 SPI）的建议串联电阻值为 330  $\Omega$ 。
- **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。使用预分频器选项在固件中控制该时钟。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：

- 按照数据手册中的建议使用去耦电容。
- 在系统电源输入端添加双向滤波器。这对于传导辐射和干扰都有效。 $\pi$  型滤波器能够防止电源噪声影响敏感器件，但同时也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	说明	应用
均值滤波器	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（有反馈回路）	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发出预定义响应的非线性滤波器	通常用于非触摸事件中，用于阻止 CapSense 数据传输。
基于规则的滤波器	对传感器数据中观察到的模式发出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。

- 将通用 I/O 驱动设置为低耗能模式。
- 关闭高功耗模块。
- 优化 CPU 速度，以降低功耗。
- 在较低的  $V_{DD}$  下工作。

除了上述建议外，还可应用睡眠-扫描方法。

### 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。

$$I_{AVE} = \frac{(I_{Act} \times t_{Act}) + (I_{Slp} \times t_{Slp})}{T} \quad \text{公式 10}$$

器件的平均功耗可按以下方式计算：

$$P_{AVE} = V_{DD} \times I_{AVE} \quad \text{公式 11}$$

### 5.5.3 响应时间与功耗

如公式 11 所示，可通过降低  $I_{AVE}$  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，在大部分时间内器件将处于睡眠扫描模式。如先前章节所述，器件会周期性扫描传感器，并进入睡眠，从而消耗较低的功率。用户通过触摸传感器操作系统时，器件将切换到连续扫描模式。在该模式下，传感器连续扫描而不会进入睡眠模式，从而大大缩短响应时间。在指定的超时期间内，器件仍然处于持续扫描模式。如果在该超时周期内，用户未对传感器进行操作，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. 将项目下载到 **CapSense** 器件中，并测量当前功耗。将测量好的电流分配给  $I_{ACT}$ 。
3. 从数据手册中查询有关睡眠电流的信息，然后将其分配给  $I_{SLP}$ 。
4. 在示波器中监测输出引脚的切换，并测量两次切换间的时间间隔。这样可提供有效时间。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. 睡眠扫描周期减去有效时间便得出睡眠时间。 $T_{SLP} = T - t_{ACT}$ 。
7. 公式 10 用于计算平均电流。
8. 公式 11 用于计算平均功耗。

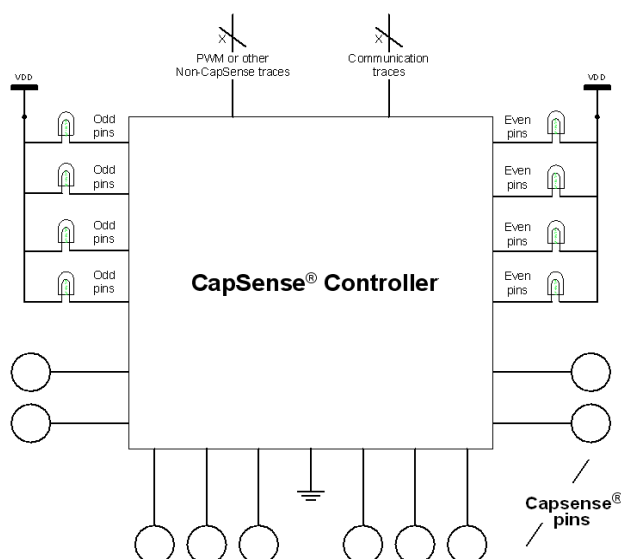
图 5-1. 全局资源窗口

Global Resources	Value
Power Setting [ Vcc / SysClk freq ]	5.0V / 24MHz
CPU_Clock	SysClk/1
Sleep_Timer	64_Hz
VC1= SysClk/N	512_Hz
VC2= VC1/N	64_Hz
VC3 Source	8_Hz
VC3 Divider	1_Hz
	1

## 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 显示的是针对 32-QFN 封装进行此隔离的基本版本。由于各个功能已被隔离，因此 **CapSense** 控制器得到定向，为了确保通信、LED 和感应走线之间不应存在交叉。

图 5-2. 通信、CapSense 和 LED 的建议端口隔离

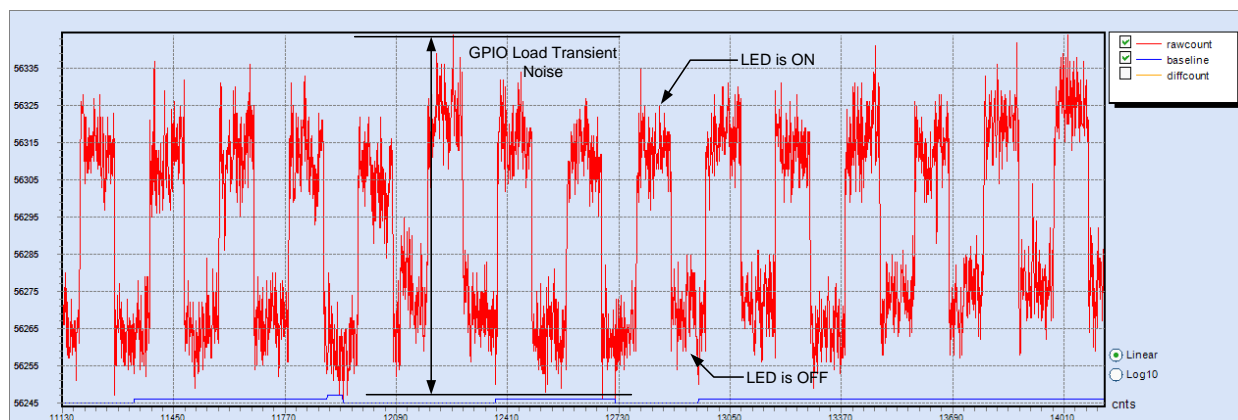


**CapSense** 控制器架构限制了奇偶端口引脚上的电流预算。奇数引脚是指以奇数作为引脚编号的端口引脚。对于 **CapSense** 控制器，如果奇数端口引脚的电流预算为 100 mA，那么所有奇数端口引脚的总电流不应超过 100 mA。除了总电流预算限制外，每个端口引脚也有最大的电流限制，相关定义请见 **CapSense** 控制器数据手册中的相关内容。

## 5.7 GPIO 负载瞬态

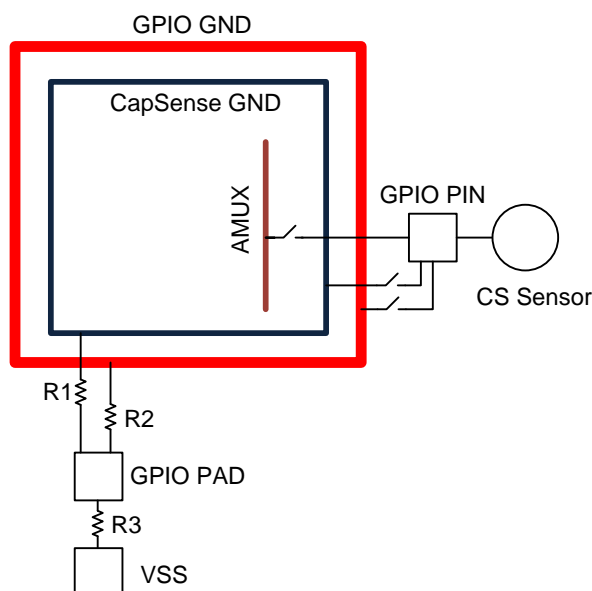
当 GPIO 通过将端口引脚驱动为强驱动低电平，并将一个大电流 ( $> 10\text{ mA}$ ) 输送给该芯片的接地面时，会有噪声被引入到 CapSense 系统内。通过 GPIO 流入地面的电流量的瞬间更改被视为 GPIO 负载瞬态。因 GPIO 负载瞬态引入 CapSense 系统的噪声被称为 GPIO 负载瞬态噪声，如图 5-3 所示。本节介绍的是如何使用硬件技术降低噪声以及如何使用固件技术补偿噪声。

图 5-3. CapSense 系统中的 GPIO 负载瞬态噪声



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

图 5-4. CY8C20x66A/S 中的接地结构



**注意：**R1、R2、R3是接合线电阻

对于强大的CapSense设计，最差情况下的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，最差情况下的噪声将出现在CapSense系统内。

GPIO负载瞬态噪声根据传感器扫描分辨率而递增。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，从而导致传感器误触发。以下内容介绍了降低GPIO负载瞬态噪声的方法。

## 5.7.1 降低 GPIO 负载瞬态噪声的硬件指南

### a) 降低传感器 $C_P$

传感器  $C_P$  确定传感器扫描分辨率参数。 $C_P$  越大，分辨率参数越高，以便得到的信噪比  $> 5:1$ 。将分辨率参数设置为高会使 GPIO 负载瞬态噪声的振幅递增。因此，建议按照 [CapSense](#) 设计指南所介绍的布局指南尽量减少传感器  $C_P$ 。

### b) 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，则使用外部晶体管或驱动器 IC。

### c) 为 LED 选择合适引脚

所有 CapSense 控制器都提供高灌电流和拉电流的端口引脚。使用来自端口引脚的高灌电流或拉电流时，应该选择最接近器件接地引脚的端口，以最小化 GPIO 负载瞬间噪声。

## 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，可以根据规则算法来更新传感器基准线。下面介绍了一种补偿基准线的方法。

图 5-5 显示的是由 GPIO 负载瞬态造成的误触发情况。

1. 在第一个事件内，传感器上没有手指而且 LED 为关闭状态。
2. 在第二个事件内，传感器上有手指而且移入原始计数大于手指阈值。
3. 由于移入原始计数大于手指阈值，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 负载瞬态噪声的移入原始计数的存在，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，以指出传感器的误触发。

为了防止传感器和 LED 永远保持打开状态，需要补偿传感器基线，以下步骤介绍的是这方面的内容。

图 5-5. 未补偿基准线的 CapSense 传感器变量

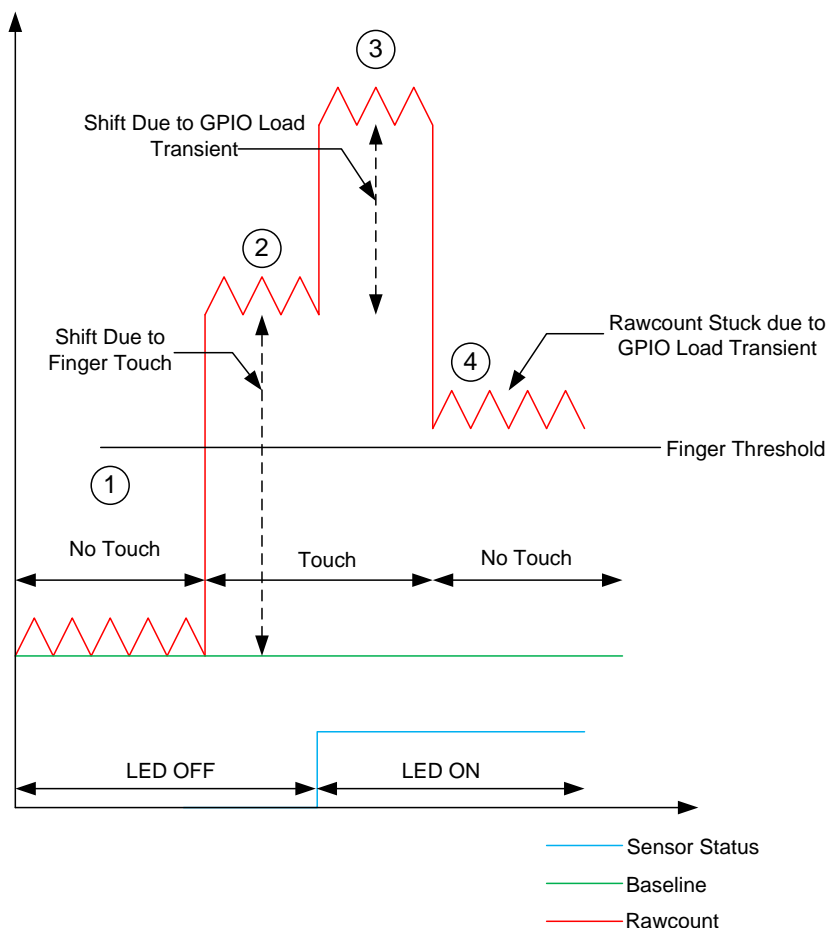
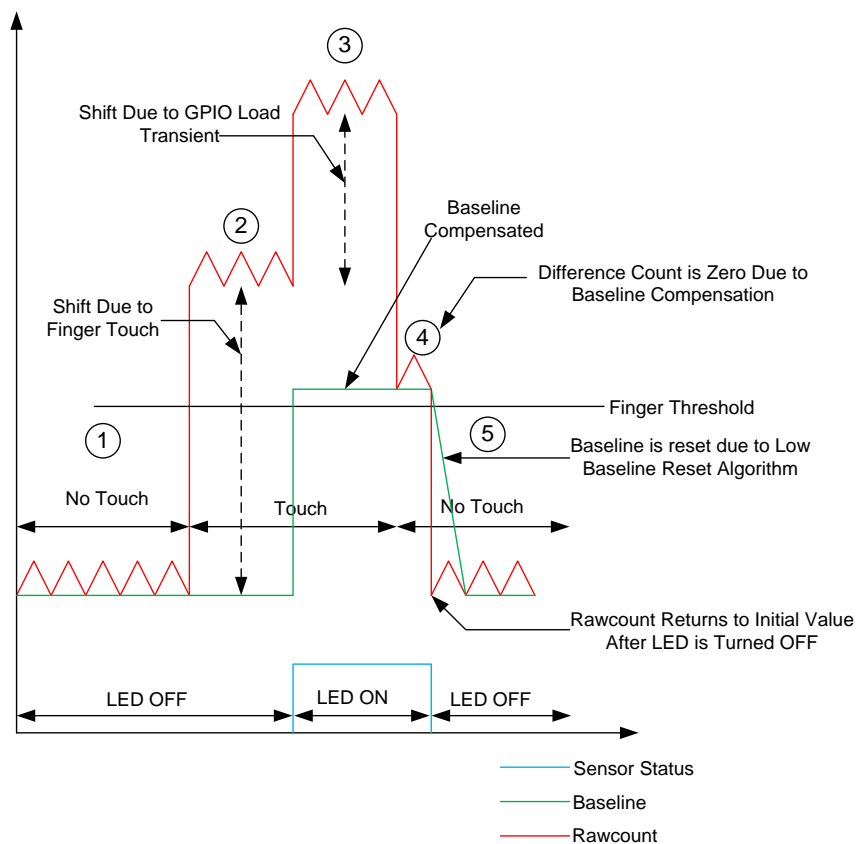


图 5-6 显示的是通过补偿传感器基线消除误触发的情况。

1. 在第一个事件内，传感器上没有手指而且 LED 为关闭状态。
2. 在第二个事件内，传感器上有手指，并且移入原始计数（计数差值）大于手指阈值。
3. 由于移入计数差值大于手指阈值，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。在这里，噪声 = 原始计数（LED 为打开） - 原始计数（LED 为关闭）  
 由 GPIO 负载瞬态造成的噪声计数被添加到基准线内，因此，当移除手指时，计数差值将为 0，而且 LED 被关闭。
5. LED 被关闭后，原始计数将返回初始值，并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 入门手册》中提供了详细的 PCB 布局指南。

## 6. 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，可以完成该功能。进入睡眠模式后，中央 CPU 停止运行，内部主振荡器（IMO）被禁用，带隙电压参考断电，闪存存储器模型也被禁用。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。除睡眠模式外，各节省功耗技术包括：

- 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

睡眠模式对设计造成负面影响。若在使用时不注意，可能会造成无法预测的操作。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。

### 6.1 其他节能技术

除睡眠模式外，所有节能技术均是基于应用的技术。有些会产生不良结果。以下章节将详述每种技术。

```
ABF_CR0 &= 0xc3; // Buffer Off
```

#### 6.1.1 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，如果引脚被驱动为高电平，那么当前的驱动模式应为电阻下拉。这样可以确保引脚驱动模式非电阻性，从而消除所有可能产生的故障。

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。模拟高阻的正确位模型为 110b。0 端口暂时进入电阻下拉驱动模式，然后使用下列代码从强驱动切换为模拟高阻态。

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

## 6.1.2 全部放在一起

下列代码是 28 引脚器件的典型睡眠准备序列的样本。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模式，而且中断重新被使能。

```
void PSoC_Sleep(void){
    M8C_DisableGInt;
    ARF_CR &= 0xf8; // analog blocks Off
    ABF_CR0 &= 0xc3; // analog buffer off
    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;
}
```

## 6.1.3 睡眠模式复杂性

CapSense 控制器可以通过复位或中断的方式退出睡眠模式。CapSense 控制器中有三种复位方式：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。

## 6.1.4 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，睡眠准备发生时，应全局禁用中断并在写入 SLEEP 位之前重新使能中断。

## 6.1.5 全局中断使能

从中断中唤醒 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码，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;
```

## 6.2 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，则需要执行的是睡眠指令后的第一条指令。这是因为 CapSense 控制器完全进入睡眠模式前，已提前提取了该指令。因此，如果禁用了全局中断，指令将在睡眠启动前从它所停止的位置继续执行。

### 6.2.1 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 控制器唤醒及重新使能后重新尝试锁定时经常过冲。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。通常情况下，SYSCLK 为 24 MHz。

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

### 6.2.2 执行 Global Interrupt Enable（全局中断使能）

避免在写入 SLEEP 位的指令边界上使能中断。如果睡眠指令在发生中断返回（reti）指令时执行，则会忽略固件的所有睡眠准备。要避免发生这种情况，请在准备进入睡眠时暂时禁用中断，然后重新使能中断。由于全局中断指令时序，中断在执行下一个指令过程中不能发生，这种情况可以设置睡眠位。

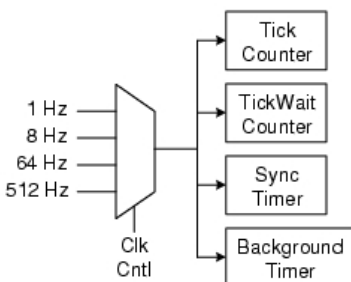
### 6.2.3 I<sup>2</sup>C 从设备处于睡眠模式

在睡眠模式下使用 I<sup>2</sup>C 从设备有许多复杂情况。由于在睡眠过程中，IMO 和 CPU 被关闭，因此 CapSense 控制器不再执行任何操作。I<sup>2</sup>C 地址会引起这些问题。当将 I<sup>2</sup>C START 条件发送到特定地址时，CapSense 控制器无法处理该地址，因此发出 NAK 响应。典型的解决办法是在 I<sup>2</sup>C 总线的时钟或数据线上设置下降沿中断。然后，主设备向其发送虚拟的 START 条件以唤醒 CapSense 控制器。唤醒与实现处理 I<sup>2</sup>C 地址之间有一大段延迟时间，因此主设备可能需要长达 200 μs 的时间延迟，然后再执行下一次发送或继续发送，直到收到 ACK 为止。该解决方案还存在另一个问题，即 CapSense 控制器将在所有 I<sup>2</sup>C 下降沿上被唤醒，这样会加长总有效时间并增大睡眠电流。另一个解决方案是使用第三个 GPIO 引脚来唤醒 CapSense 控制器，在适当的延迟时间后发送初始的 START 条件。

### 6.2.4 睡眠定时器

CapSense 控制器提供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，这些器件被使用，并执行类似的功能。实际的睡眠定时器由永远不被关闭的内部低速振荡器控制。可选频率间隔为 1 Hz、8 Hz、64 Hz 和 512 Hz 时，定时器发生一个中断。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（用于生成定期中断）、程序环路延迟功能、可设置的递减计数器和控制回路时间的回路循环调节器。图 6-1 显示了该功能的简单框图。

图 6-1. 睡眠定时器用户模块框图



## 7. 资源



### 7.1 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 CY8C20xx6A/H/AS 器件系列丰富的技术资源，请通过 [CY8C20xx6A/H](#) 网页查找。

### 7.2 数据手册

CapSense CY8C20XX6A/H/AS 器件系列的数据手册可在 [www.cypress.com](http://www.cypress.com) 上获取。

- [CY8C20x36A](#)、[CY8C20x46A](#)、[CY8C20x66A](#)、[CY8C20x96A](#)、[CY8C20x46AS](#) 和 [CY8C20x66AS](#)
- [CY8C20336H](#)、[CY8C20446H](#)

### 7.3 技术参考手册

赛普拉斯已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）进行快速而简明的访问。

- [PSoC® CY8C20x66](#)、[CY8C20x66A](#)、[CY8C20x46/96](#)、[CY8C20x46A/96A](#)、[CY8C20x36](#)、[CY8C20x36A](#) 技术参考手册（TRM）

### 7.4 开发套件

#### 7.4.1 通用的 CapSense 控制器套件

通用 CapSense 控制器套件具有预定义控制电路和即插即用硬件，易于进行原型设计和调试。包含用于调校和数据采集的编程硬件和 I<sup>2</sup>C 至 USB 桥接器。

- [CY3280-20xx6](#) 通用 CapSense 控制器

#### 7.4.2 通用 CapSense 模块板

##### 7.4.2.1 简单按键模块板

[CY3280-BSM](#) 简单按键模块由十个 CapSense 按键和十个 LED 构成。该模块可以连接到任何 CY3280 通用 CapSense 控制器电路板

##### 7.4.2.2 矩阵触摸按键模块板

[CY3280-BMM](#) 阵列按键模块由 8 个 LED 和 8 个 CapSense 传感器组成（8 个 CapSense 传感器以 4x4 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。

### 7.4.2.3 线性滑条模块板

**CY3280-SLM** 线性滑条模块由五个 CapSense 按键、一个线性滑条（带十个传感器）和五个 LED 组成。该模块可连接至任何 CY3280 通用 CapSense 控制器电路板。

### 7.4.2.4 辐射滑条模块板

**CY3280-SRM** 辐射滑条模块由四个 CapSense 按键、一个辐射滑条（带十个传感器）和四个 LED 组成。该模块可以连接至任何 CY3280 通用 CapSense 控制器电路板。

### 7.4.2.5 通用 CapSense 原型设计模块

**CY3280-BBM** 通用 CapSense 原型设计模块提供了与路由至附带控制器电路板上 44 引脚连接器的各个信号的访问。原型设计模块电路板与通用的 CapSense 控制器电路板一起使用，以实施其他功能（该功能不是其他单用途通用 CapSense 模块电路板所拥有的）。

## 7.4.3 在线仿真（ICE）套件

ICE 转接板通过柔性线缆，为 CY3215-DK 在线仿真器与原型系统中的 PSoC 目标器件（或带有特定封装转接板支脚的 PCB）提供互连。以下转接板可用。

- 用于调试 CY8C20236/46A CapSense PSoC 器件的 CY3250-20246QFN 在线仿真（ICE）转接板套件
- 用于调试 CY8C20336/346A CapSense PSoC 器件的 CY3250-20346QFN 在线仿真（ICE）转接板套件
- 用于调试 CY8C20636/646/666A CapSense PSoC 器件的 CY3250-20666QFN 在线仿真（ICE）转接板套件
- 用于调试 CY8C20536/546/566A CapSense PSoC 器件的 CY3250-20566 在线仿真（ICE）转接板套件

## 7.5 样本电路板文件

赛普拉斯提供了原理图和电路板样本文件，您可以参考它来完成您的 PCB 设计流程设计。

- CY8C20466A 上带有 I<sup>2</sup>C 插座的按键设计
- CY8C20466A 上带有 I<sup>2</sup>C 插座的按键与滑条设计

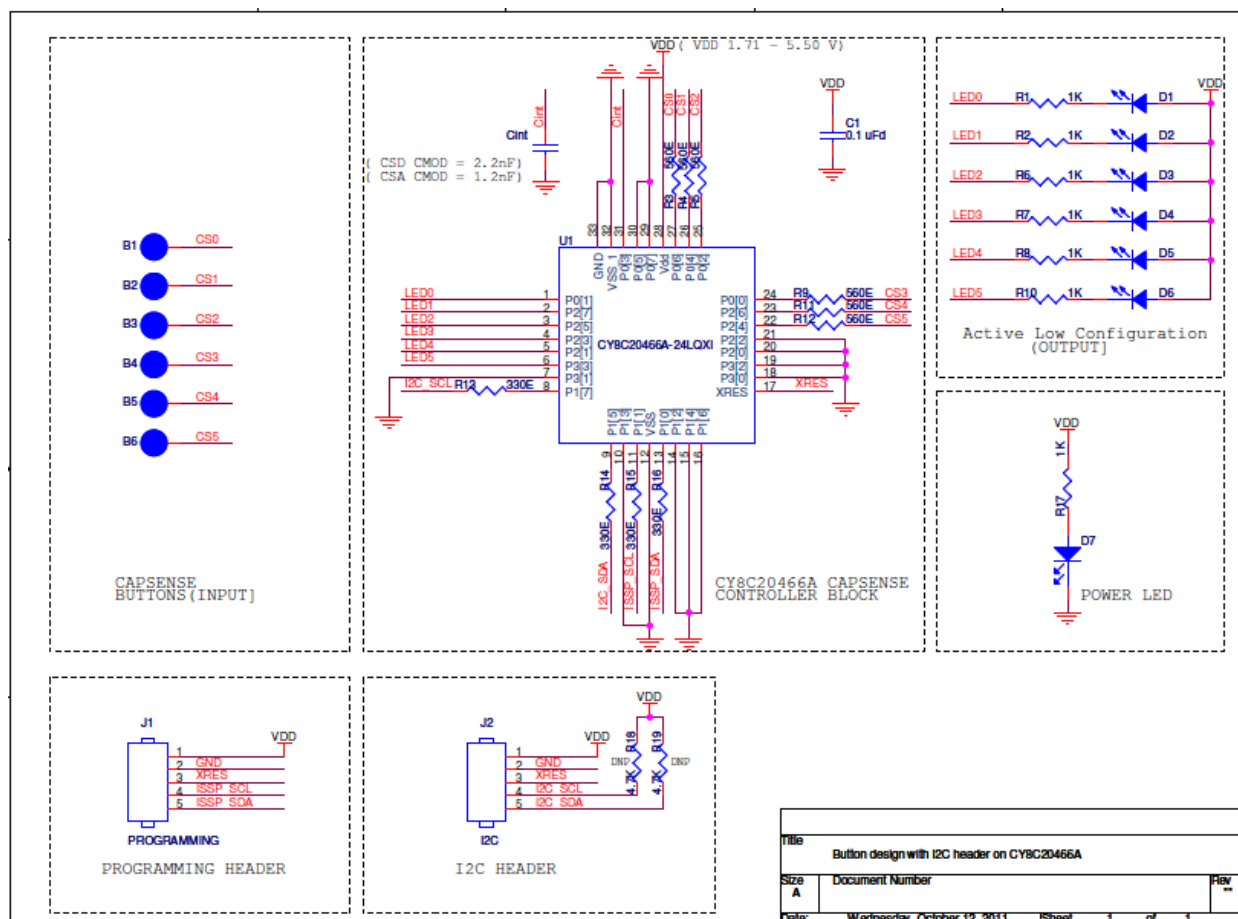
**注意：**板文件（原理图、布局与 Gerber 文件）位于本文档的登陆页面上。

图 7-1 和图 7-2 显示了板原理图。

以下原理图用于支持：

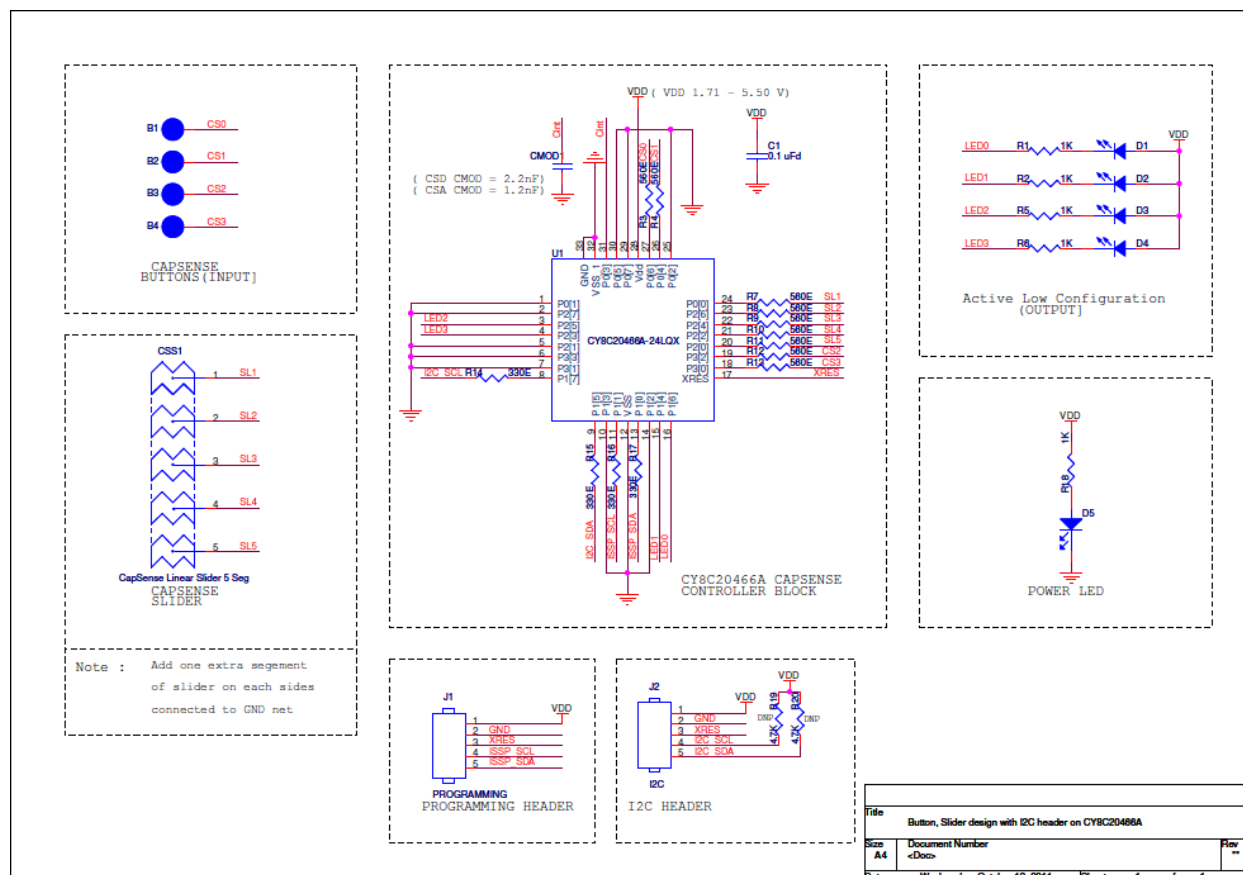
- 6 个 CapSense 传感器。这些传感器被分配给 CY8C20466A-24LQXI 器件的引脚 P0[6]、P0[4]、P0[2]、P0[0]、P2[6]及 P2[4]。
- 6 个 GPO 连接到 CY8C20466A-24LQXI 的引脚 P0[1]、P2[1]、P2[3]、P2[5]、P2[7]及 P3[3]，以驱动 D1、D2、D3、D4、D5 及 D6 等 LED。
- 通过编程插座 J1 对 CY8C20466A-24LQXI 进行编程。
- 通过 I<sup>2</sup>C 插座 J2，I<sup>2</sup>C 与 CY8C20466A-24LQXI 进行通讯。

图 7-1. CY8C20466A — 板原理图上带有 I²C 插座的按键设计



以下原理图用于支持:

- 4 个 CapSense 传感器。这些传感器被分配给 CY8C20466A-24LQXI 器件的引脚 P0[6]、P0[4]、P3[2]和 P3[0]。
- 5 段线性滑条。这些滑条段被分配给 CY8C20466A-24LQXI 器件的引脚 P0[0]、P2[6]、P2[4]、P2[2]和 P2[0]。
- 连接至引脚 P1[6]、P1[4]、P2[5]和 P2[7] CY8C20466A-24LQXI 的四个 GPO，用于驱动 LED D1、D2、D3 和 D4。
- 通过编程插座 J1 对 CY8C20466A-24LQXI 进行编程。
- 通过 I²C 插座 J2，I²C 与 CY8C20466A-24LQXI 进行通讯。

图 7-2. CY8C20466A 上带有 I<sup>2</sup>C 插座的按键与滑条设计

## 7.6 PSoC Programmer

**PSoC Programmer** 是用来为 PSoC 器件编程的既灵活又具有高集成度的编程应用程序。当它与 PSoC Designer 和 PSoC Creator 配合使用时，可以将任何设计编程到 PSoC 器件上。

PSoC Programmer 为您提供了带有 API 的硬件层，以便通过使用编程器和桥接器来设计特定的应用程序。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。

## 7.7 CapSense 数据查看工具

在 CapSense 设计过程中，您将要监测用于调校和调试的 CapSense 数据（原始计数、基准线、计数差值等）。

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。

## 7.8 PSoC Designer

赛普拉斯提供了专有的集成设计环境 — **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。

## 7.9 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](#)
- [CY8C20xx6A 上带有 EzI2C 从设备的 CSD 软件滤波器](#)

## 7.10 设计支持

为了确保成功实现 CapSense 解决方案，赛普拉斯提供了各种设计支持渠道。

- [知识库文章](#) — 按产品系列浏览技术文章，或根据不同 CapSense 主题进行搜索。
- [CapSense 应用笔记](#) — 以本文档提供的信息为基础的各种应用笔记。
- [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。
- [视频库](#) — 使用视频教程提高学习速度。
- [质量和可靠性](#) — 赛普拉斯致力于满足客户的所有要求。在我们的“质量”页面上，您可找到可靠性与产品质量鉴定的相关报告。
- [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

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功耗，并且不需要固件开发。这些器件包括 CY8CMBR3XXX 和 CY8CMBR2XXX 系列。

## 中心或中心位置

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 中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。该参数有助于抑制假的触摸信号。

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 (Driven-Shield)

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他传感器进行的扫描，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。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  $(2^N - 1) / \text{调制器的时钟频率}$ ，其中 N 是扫描分辨率。

## Modulation IDAC（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，以及突然发生其他的环境变化。

**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<sub>p</sub>）**

寄生电容是由 PCB 走线、传感器垫片、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。

**刷新闻隔**

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**

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显示在传感器附近时造成的误触摸状态的设置。

使能传感器自动复位时，即使计数差值大于噪声阈值，也可以更新基准线。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，以便防止水滴或其他液体引起的误触摸。屏蔽电极由 CSD 模块输出的屏蔽信号驱动。请参见[驱动屏蔽 \(Driven-Shield\)](#)。

### 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: AN65973 — CY8C20xx6A/H/AS CapSense®设计指南

文档编号: 001-78419

版本	提交日期	变更者	变更说明
**	04/26/2012	HHLL	首次翻译。
*A	05/06/2015	HHLL	本文档版本号为 Rev*A, 译自英文版 001-65973 Rev*G。
*B	06/15/2016	HHLL	本文档版本号为 Rev*B, 译自英文版 001-65973 Rev*H。