CY8C20xx6A/H/AS

CapSense® 设计指南

Doc No: 001-78419 Rev. **

赛普拉斯半导体

198 Champion Court

圣荷塞, CA 95134-1709

电话（美国）： 800.858.1810

电话（国际）： 408.943.2600

http://www.cypress.com

版权所有

© 赛普拉斯半导体公司，2010-2012。 此处所包含的信息可能会随时更改，恕不另行通知。 除赛普拉斯产品的内嵌电路之外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任， 也不根据专利或其他权利以明示或暗示的方式授予任何许可。 除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证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。

商标

PSoC Designer™ 和 Programmable System-on-Chip™ 以及 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。 除上述指定的用途之外，未经赛普拉斯的明确书面许可，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（但不仅限于）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。

产品使用可能受适用的赛普拉斯软件许可协议限制。

目录

1. 简介 7

1.1 摘要 7

1.2 赛普拉斯 CapSense® 文档体系 7

1.3 CY8C20xx6A/H/AS 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 CY8C20x6A/AS/H 中的电容式感应方法 14

2.2.1 CapSense Sigma-Delta (CSD) 14

2.2.2 CapSense 逐次逼近电磁兼容 (CSA_EMC) 15

2.3 SmartSense™ 自动调校 16

3. CapSense 设计工具 18

3.1 概述 18

3.1.1 PSoC Designer 和用户模块 18

3.1.2 通用 CapSense 控制器套件 19

3.1.3 通用的 CapSense 控制器模块板 19

3.1.4 CapSense 数据查看工具 20

3.2 用户模块概述 20

3.3 CapSense 用户模块全局数组 21

3.3.1 原始计数 21

3.3.2 基准线 21

3.3.3 差值（符号） 21

3.3.4 传感器状态 22
3.4 CSD 用户模块参数 22
3.4.1 用户模块高级参数 23
3.4.2 CSD 用户模式低级参数 26
3.4.3 CSA_EMC 用户模式低级参数 28
3.4.4 SmartSense 用户模块参数 29
3.4.5 SmartSense_EMC 用户模块参数 30
4. 使用 用户模块对 CapSense 进行性能调节 32
4.1 通用注意事项 32
4.1.1 信号，噪声，信噪比 32
4.1.2 充电/放电频率 33
4.1.3 验证基准线更新阈值的重要性 34
4.2 调校 CSA_EMC 用户模块 35
4.3 建议使用的 CSA_EMC 中的 C_{INT} 值 35
4.4 测量传感器 C_P 36
4.5 估算 CSA_EMC 时钟 36
4.6 设置建立时间 37
4.7 监控 CapSense 数据 37
4.8 增大信噪比的方法 37
4.8.1 降噪 37
4.8.2 增大信号强度 37
4.9 调校 CSD 用户模块 38
4.9.1 推荐 CSD 所需的 C_{CMOD} 值 39
4.9.2 屏蔽电极输出 (ShieldElectrodeOut) 39
4.9.3 I_{DAC} 范围 39
4.9.4 Autocalibration 39
4.9.5 I_{DAC} 值 39
4.9.6 预充源 39
4.9.7 预分频器 39
4.9.8 分辨率 40
4.9.9 扫描速度 (Scanning Speed) 41
4.9.10 高层 API 参数 42
4.9.11 高层 API 参数 42

4.10 使用 SmartSense 用户模块 43
4.10.1 SmartSense 指南 43
4.10.2 差异理解 43
4.10.3 SmartSense 的 C_{CMOD} 推荐值 44
4.10.4 SmartSense 用户模块参数 44
4.10.5 SmartSense_EMC 用户模块的专用指南 44
4.10.6 CapSense 传感器的扫描时间 45
4.10.7 SmartSense 响应时间 47
4.10.8 使用 SmartSense_EMC 用户模块以确保低信噪比的方法 47
4.10.9 固件设计指南 48

5. 设计考虑事项 51

5.1 选择覆盖层 51
5.2 ESD 保护 52
5.2.1 预防 52
5.2.2 重新定向 52
5.2.3 钳制 52
5.3 电磁兼容性 (EMC) 注意事项 53
5.3.1 辐射干扰 53
5.3.2 辐射 53
5.3.3 传导抗扰与辐射 53
5.4 软件滤波 53
5.5 功耗 54
5.5.1 系统设计建议 54
5.5.2 睡眠扫描方法 55
5.5.3 响应时间与功耗 55
5.5.4 测量平均功耗 55
5.6 引脚分配 56
5.7 PCB 布局指南 56

6. 低功耗设计注意事项 57

6.1 其他节能技术 57
6.1.1 将 HI-Z 模拟设为驱动模式 57
6.1.2 实例应用 58
6.1.3 睡眠模式问题 58

6.1.4 待处理中断 58
6.1.5 全局中断启用 58
6.2 后唤醒执行序列 59
6.2.1 PLL 模式启用 59
6.2.2 执行全局中断启用 59
6.2.3 睡眠模式下的 I^2C 从器件 59
6.2.4 睡眠定时器 59

7. 资源 61

7.1 网站 61
7.2 数据手册 61
7.3 技术参考手册 61
7.4 开发套件 61
7.4.1 通用 CapSense 控制器套件 61
7.4.2 通用 CapSense 模块板 62
7.4.3 在线仿真 (ICE) 套件 62
7.5 样版文件 62
7.6 PSoC Programmer 64
7.7 CapSense 数据查看工具 65
7.8 PSoC Designer 65
7.9 代码示例 65
7.10 设计支持 65

1. 简介

1.1 摘要

该内容是关于如何利用 CapSense 控制器中的 CY8C20xx6A/AS/H 系列，来执行电容式感应 (CapSense®) 功能的操作指南。 本指南包含以下主题：

- CapSense 控制器中 CY8C20xx6A/AS/H 系列的特点
- CapSense 操作守则
- CapSense 设计工具简介
- 调校 CapSense 系统至最佳性能的方法
- CapSense 系统的电气及机械设计注意事项
- CapSense 低功耗设计注意事项
- 在系统中设计 CapSense 所需的附加资源及支持

1.2 赛普拉斯 CapSense® 文档体系

图 1-1 及表 1-1 对赛普拉斯 CapSense 的文档体系进行了概述。 这些文档体系资源可以让操作人员快速获取所需信息，成功完成 CapSense® 的产品设计。 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 中的链接找到。每个编号任务的帮助信息都可以通过图 1-1

图 1-1. 典型的 CapSense 项设计流程

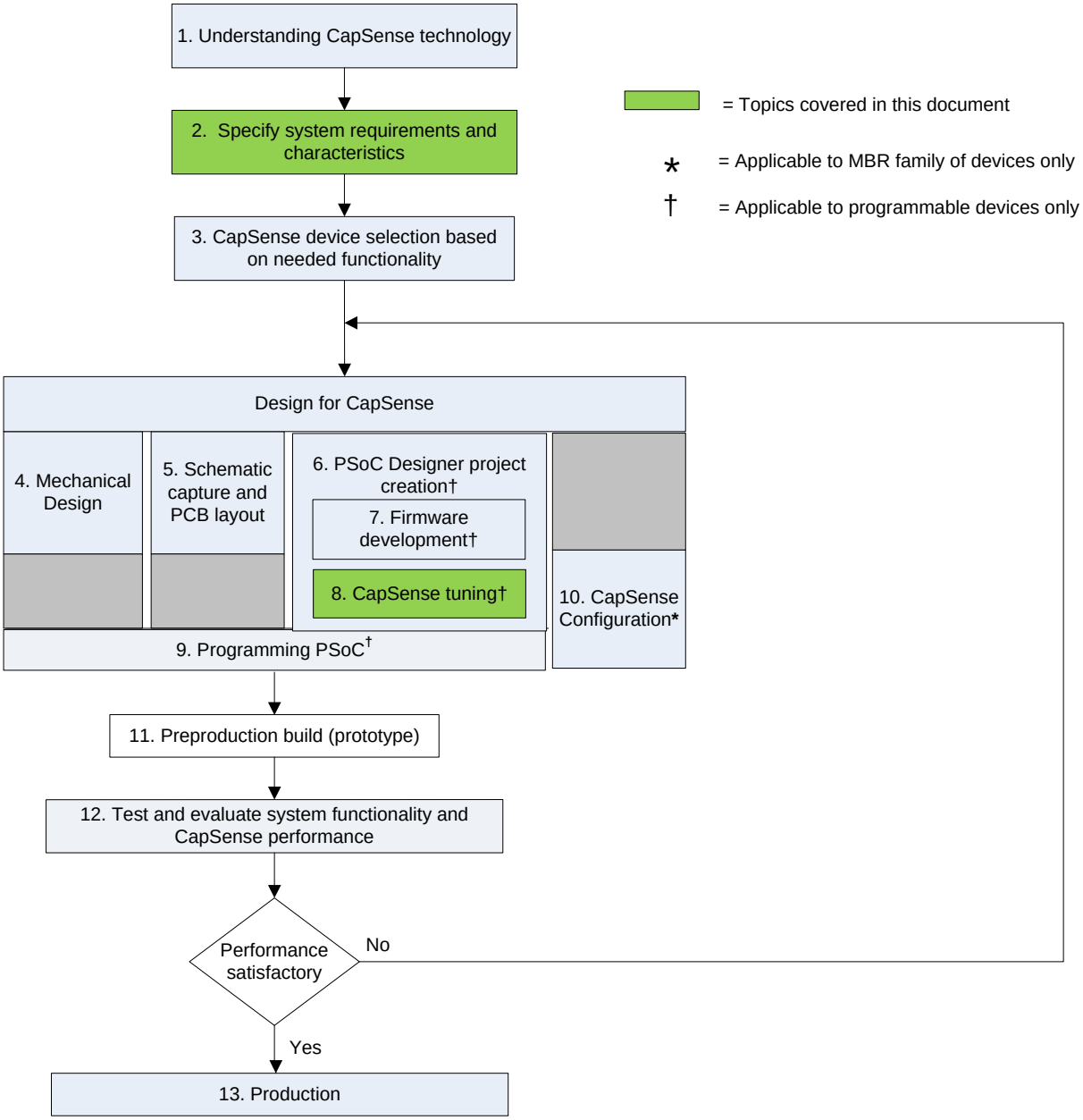


表 1-1. 赛普拉斯文档支持图 1-1 中的设计任务编号。

图 1-1 中的设计任务编号	支持赛普拉斯 CapSense® 的文档
1	●CapSense 入门
2	●CapSense 入门 ●CY8C20xx6A/AS/H CapSense 器件数据手册
3	●CapSense 入门 ● PSoC 特定系列 CapSense® 设计指南（本文档）
4	●CapSense 入门
5	●CapSense 入门 ●PSoC Designer 用户指南
6	●PSoC Designer 用户指南
7	●汇编语言用户指南 ●C 语言编译器指南 ●CapSense 代码示例 ● PSoC 系列专用技术参考手册（用于 CY8C20xx6A/AS/H）
8	● PSoC 系列专用 CapSense 设计指南（本文档） ● PSoC 系列专用 CapSense 用户模块数据手册（CSD 与 SmartSense） ● PSoC 系列专用技术参考手册（用于 CY8C20xx6A/AS/H） ● AN2397-CapSense 数据查看工具
9	●编程器用户指南 ●MiniProg3 用户指南 ● AN2026c - 用于 CY8C20xx6、CY8C20xx6A、CY8CTMG2xx、 CY8CTST2xx、CY7C643xx 和 CY7C604xx 系统内串行编程 (ISSP) 协议 ●AN44168 - 使用外部微控制器 (HSSP) PSoC 1 元件编程 ●AN59389 - 用于 CY8C20xx6、 CY8CTMG2xx 和 CY8CTST2xx 主机源串口编程
11	● PSoC 系列专用 CapSense 设计指南（本文档） ●CapSense 代码示例

1.3 CY8C20xx6A/H/AS CapSense系列特性

赛普拉斯 CY8C20xx6A/H/AS 属于 CapSense 控制器系列，低功耗、高性能、可编程 ，具有以下功能：

高级触摸感应功能：

- 可编程的电容式感应元件
  - 支持组合使用 CapSense 按钮、滑条和接近传感器

- ☐ 集成 API 以实现按钮和滑条功能
- ☐ 支持超过 36 个电容传感器或者 36 个 GPIO 或者滑条
- ☐ 支持电容范围在 5 pF 到 45 pF 之间的寄生传感器

■ SmartSense 的自动调校功能可加快上市速度

- ☐ 在加电或运行期间自动设置、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
- ☐ 运行期间环境补偿
- ☐ 检测触摸低至 0.1 pF

■ 抗噪和健壮性增强

- ☐ SmartSense 自动补偿环境和噪声变化
- ☐ SmartSense_EMC 可在传导噪声和辐射噪声等复杂的条件下为应用提供出色的抗噪能力。
- ☐ 内部调节器提供抗电源噪声稳定性，可接受纹波最高为 500 mV 的供电 V_{DD} 纹波
- ☐ 软件滤波器的集成 API 改进了信噪比

■ 超低功耗

- ☐ 三种不同的功耗模式对功耗进行优化
- ☐ 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 情况下，每个传感器电流为 28 µA

器件功能

■ 高性能、低功耗的 M8C 哈佛架构处理器

- ☐ 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

- ☐ 最高 32 KB 的闪存和 2 KB 的 SRAM
- ☐ 支持仿真的 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

- ☐ 多达 36 个通用 I/O (GPIOs)，拥有可编程引脚配置。
- ☐ 25-mA 灌电流/GPIO 和 120 mA 的总灌电流/器件
- ☐ 所有的 GPIO 中皆有电阻上拉、 high-z、开漏和强驱动模式

- 外设功能
  - ☐ 3 个 16 位定时器
  - ☐ 全速 USB - 符合 12 Mbps USB 2.0 标准
  - ☐ I^2C - 主控 (100 kHz) 和从器件（最高 400 kHz）
  - ☐ SPI - 主控和从器件 - 可配置范围从 46.9 kHz 到 12 MHz 之间
  - ☐ 最高 10 位的模数转换器 - 输入范围为 0 到 1.2 V
- 工作条件
  - ☐ 宽工作电压： 1.71 V 至 5.5 V
  - ☐ 温度范围： –40 °C 至 +85 °C

1.4 文档规范

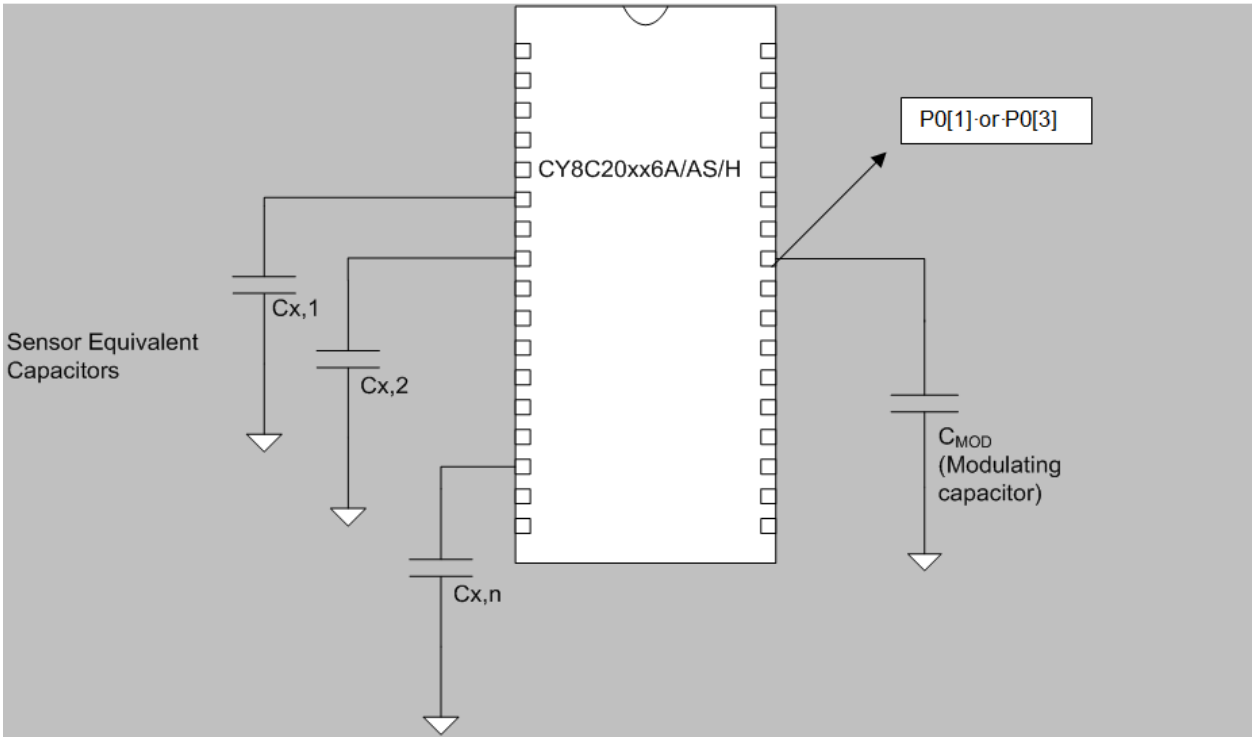
格式	用途
Courier New	显示文件位置、用户输入的文本和源代码： C:\ ...cd\icc\
斜体	显示文件名和参考文档： 阅读 *PSoC Designer 用户指南*中的 *sourcefile.hex* 文件
[方括号、粗体]	显示程序中的键盘命令： **[**Enter**] 或 [**Ctrl**] [**C**]**
“文件” > “打开”	表示菜单路径： “文件” > “打开” > “新建项目”
粗体	显示程序中的命令、菜单路径以及图标名称： 单击 **File（文件）**图标，然后再单击 **Open（打开）**。
Times New Roman 字体	显示等式： 2 + 2 = 4
文本显示在灰色框中	描述警告或者产品的独特功能。

2. CapSense 技术

2.1 CapSense 基本原理

CapSense 是一种触摸感应技术，这种技术通过测量 CapSense 控制器（已指定为传感器）上的每个 I/O 引脚电容来工作。 如图 2-1 所示，每个传感器引脚上的总电容都可以模拟成等效集总电容器，通过用 n 个传感器 $C_{X,1}$ 到 $C_{X,n}$ 的值进行设计。 电路内部的 CY8C20xx6A/AS/H 器件，把每个 C_X 的大小转换成一个数字代码，并将其存储为后期处理做准备。 CapSense 控制器的内部电路会使用其他器件 C_{MOD}，其更多信息将会在 CY8C20x6A/AS/H 中讨论。

图 2-1. 在 CY8C20x6A/AS/H PSoC 器件中实现 CapSense



如图 2-1 所示，每个传感器 I/O 引脚都可通过走线、过孔（必要时两者齐用），连接到传感器板。 覆盖层是传感器板上的绝缘层，组成产品的触摸界面。 当手指接触到覆盖层，接触部位的导电性和大小会能有效地将接地导电平面连接至传感器板。这将会在 图 2-2 中显示出来。 这种排列组成了一种平行板电容器，该电容器的容量可通过如下方式加以确定：

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

其中：

C_F = 手指接触传感器覆盖层的电容

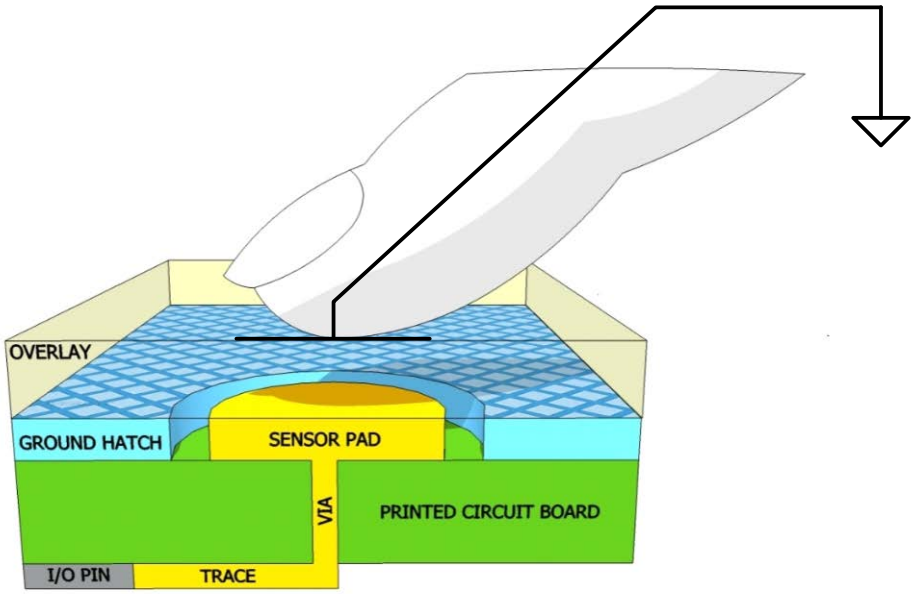
ε_0 = 真空介电常数

ε_r = 覆盖层介电常数（相对介电常数）

A = 手指和传感器板重叠区域

D = 覆盖层厚度

图 2-2. 手指激活带有传感器的 CapSense PCB 典型值部分



除平行板电容外，与覆盖层接触的手指会在其与附近其他导体之间产生边缘电场。 与平行板电容器的电场相比，这类边缘电场的作用通常极为微弱，因此一般情况下可忽略不计。

即使没有手指触摸覆盖层，传感器 I/O 引脚也具有一定的寄生电容 (C_P)。 C_P 是两个元素共同产生的：一是 CapSense 控制器内部寄生元件；二是传感器板、走线、过孔和系统内其他导电器，如接地层、其他走线、产品底盘或外壳的任何金属等之间的电场耦合程度。 CapSense 控制器测量连接到传感器引脚的总电容 (C_X)。

当手指不触摸传感器时：

$$C_X = C_P$$ 公式 2

手指与传感器板接触时，C_X 等于 C_P 和 C_F 的和：

$$C_X = C_P + C_F$$ 公式 3

通常，C_P 大于 C_F 一个数量级。 C_P 通常在 10 pF 到 20 pF 之间变动，但在特殊情况下可以高达 50 pF。 C_F 通常在 0.1 pF 到 0.4 pF 之间变动。 调校 CapSense 系统时，C_P 的大小至关重要，有关调校 CapSense 系统的知识， 会在使用用户模块调校 CapSense 性能调校中予以讨论。

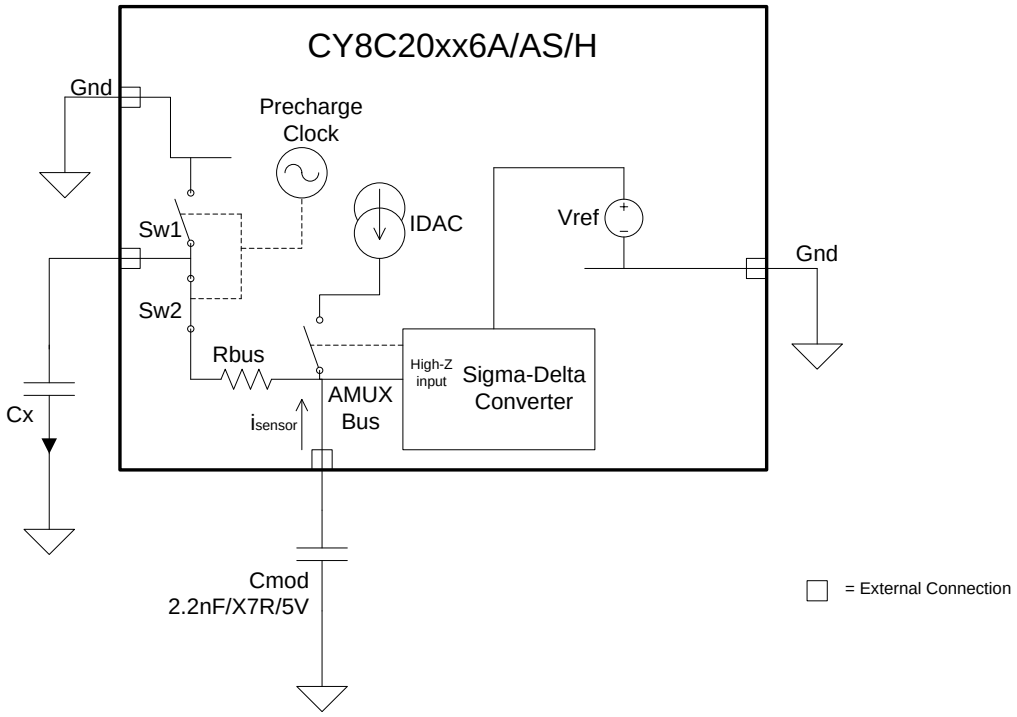
2.2 CY8C20x6A/AS/H 中的电容式感应方法

CY8C20x6A/AS/H 器件支持用多种 CapSense 方法将传感器电容 (C_X) 转化为数字计数。 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，CY8C20x6A/AS/H 器件中的 CapSense Sigma-Delta (CSD) 方法将 C_X 并入开关电容电路。 传感器 (C_X) 也可以分别通过不互相重叠的 Sw1 和 Sw2 连接到模拟复用器 (AMUX) 总线。Sw1 和 Sw2 由预充电时钟驱动由 AMUX 总线分走电流（i_{sensor}）。 i_{sensor} 的数量级与 C_X 的数量级成正比。 Sigma-Delta 转换器对 AMUX 总线电压进行采样，并产生调制位流，控制 AMUX 充电的恒流源 (I_{DAC})，使 AMUX 总线的平均电压保持为 V_{ref}。 传感器从调制电容 (C_{mod}) 泄露电荷 i_{sensor}。 C_{mod} 结合 Rbus 形成低通滤波器，衰减在 Sigma-Delta 转换器输入处的预充电开关瞬态。

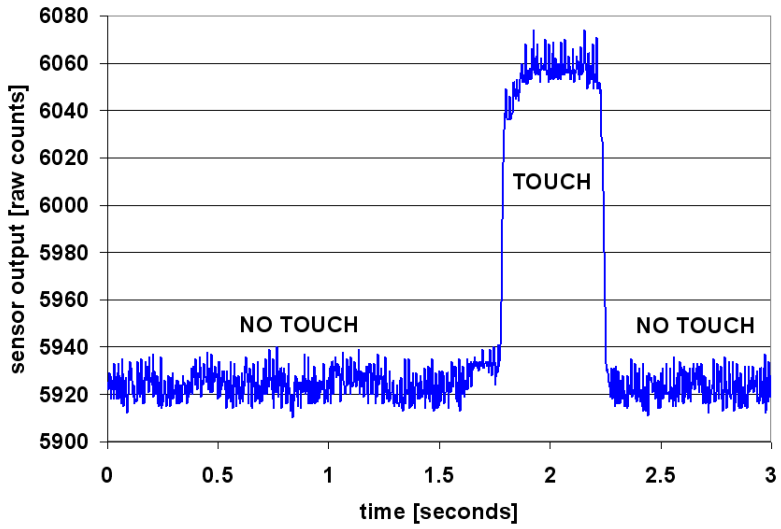
图 2-3. CSD 框图



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充电电流 (I_{DAC}) 与 i_{sensor} 相互匹配。 Sigma-Delta 转换器在传感器扫描周期中存储位流，累计结果即为数字输出值，被称为原始计数，并与 C_X 成正比。 原始计数由高级算法加以解释，以解决传感器状态问题。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 2-4。 根据 CapSense 基本原理中的解释，手指触摸传感器导致 C_X 增加 C_F，进而导致原始计数按比例增加。 通过比较稳定状态下原始计数的偏移量与预定阈值，高级算法可以确定传感器是处于 On（触摸）状态还是 Off（释放）状态。

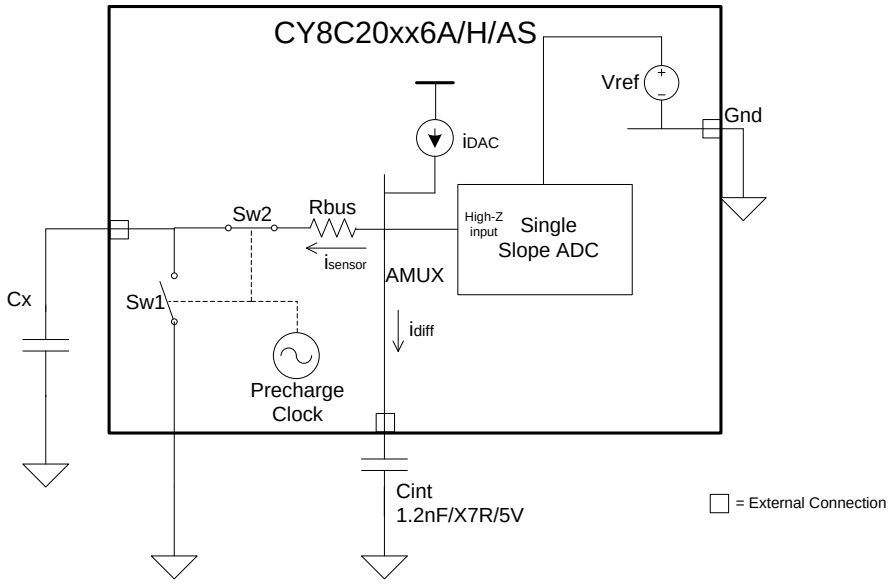
图 2-4. 手指触摸过程中的 CSD 原始计数



2.2.2 CapSense 逐次逼近电磁兼容 (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 并入开关电容电路。

图 2-5. CSA_EMC 框图



恒流源 (iDAC) 提供 I_{DAC} 量的电流至模拟开关(AMUX)。 传感器 (C_X) 也可以分别通过开关 Sw1 和 Sw2 连接至模拟开关 AMUX 总线和 GND 之间，通过 AMUX 总线释放 i_{sensor} 量的电流。 i_{sensor} 的大小与 C_X 的大小成正比。 Sw1 和 Sw2 的钟来自非重叠时钟，称作预充电时钟。

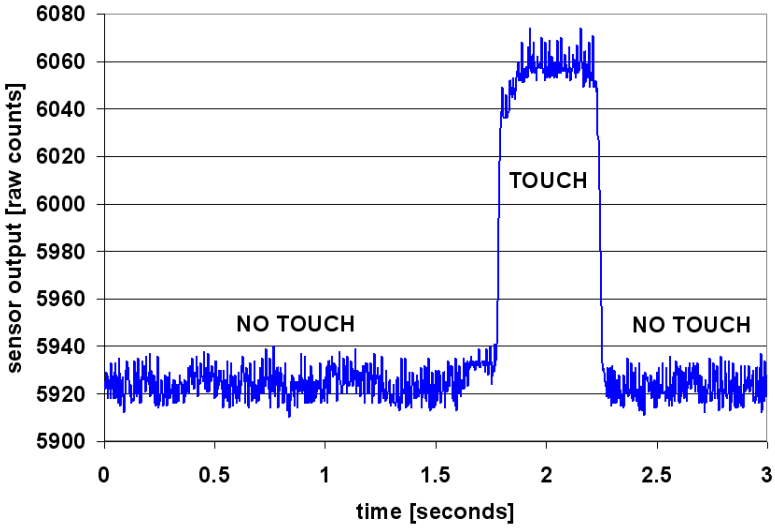
积分电容 C_{int} 集成差动电流 i_{diff} (I_{DAC} 和 i_{sensor} 的差值)，并增强其电位。 电荷持续积累，直到电位超过 C_{int}，达到与 i_{sensor} 和 I_{DAC} 等同的均衡水平。 积分时间是指建立时间。

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加以解释，以解决传感器状态问题。

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。

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 CSA_EMC 原始计数，如图 2-6。 根据 CapSense 基本原理中的解释，手指触摸传感器导致 C_X 增加 C_F，进而导致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 （未触摸）状态。

图 2-6. 手指触摸过程中的 CSA_EMC 原始计数



增强 CSA_EMC CapSense 算法，可使其在 RF 干扰下也能正常运作。 当 CapSense 受到传导干扰，以及 AC 噪音和其他噪声源，如反相器、变压器、开关电源等干扰的情况下，会应用 CSA_EMC。 CSA_EMC 用户模式低级参数将详细讨论该主题。

2.3 SmartSense™ 自动调校

为确保系统的正确运行和用户的愉快体验，调校触摸感应用户界面是至关重要的一步。 典型设计流程包括在产品初始阶段、系统集成期间调校传感器接口，最后在产品投入生产时，对其进行微调。 调校是一个反复、费时的过程。 SmartSense 自动调校用于简化用户界面开发周期。 自动调校易于使用，并且在从原型设计到大量生产的整个产品开发周期中，通过消除调校过程，显著减少设计周期。 SmartSense 在加电时，自动调校每个 CapSense 传感器，然后在运行期间，监测并保持传感器，使其达到最佳性能。 该技术适用于解决 PCB、外覆层和噪声发生器中的生产偏离，其中噪声发生器包括 LCD 反相器、AC 传输线噪声、开关电源等，并能自动解除偏离。

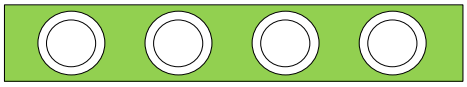
2.3.1.1 工艺差异

CY8C20xx6A/H/AS 的 SmartSense 用户模块 (UM) 在传感器寄生电容值在 5pF 到 45pF 的情况适用（典型电容值 C_P 在 10 pF 到 20 pF 之间）。 每个特定传感器基于自身特性，自动设置灵敏度。 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出率。 每个传感器的寄生电容值都能够根据 PCB 布局、PCB 生产工艺偏离而变化，或在 PCB 多源供应链内，随供应商的 PCB 偏离而变化。 传感器的灵敏度取决于寄生电容；较高的 C_P 值降低传感器灵敏度，造成手指触摸信号振幅降低。 在某些情况下，C_P 值的变化使系统失调，造成传感器性能欠佳（过于灵敏或不够灵敏），甚至导致传感器不能运作。 在这两种情况下，您都必须重调系统，并在某些情况下重新验证用户界面子系统。 SmartSense 自动调校能解决这些问题。

SmartSense 自动调校使平台设计变为可能。 设想笔记本电脑的电容触摸感应多媒体按键；按钮之间的空间取决于笔记本电脑的大小和键盘布局。 在此例中，宽屏机按钮间的空间要大于标准比例屏幕模型的按钮空间。 增大按钮空间意味着延长传感器和 CapSense 控制器之间的走线，从而导致更高的传感器电容值。 这意味着在同一平台设计的不同模型下，CapSense 按钮的寄生电容值不同。 虽然所有笔记本电脑模型的按钮功能相同，但必须对每个模型的传感器分别进行调校。 Smartsense 能够实现高效、自动调校，使您在做平台设计时，能采用最佳建议做法，该做法在 CapSense 入门中的 PCB 布局中有所显示。

图 2-7. 21- 英寸模型笔记本电脑多媒体按键的设计

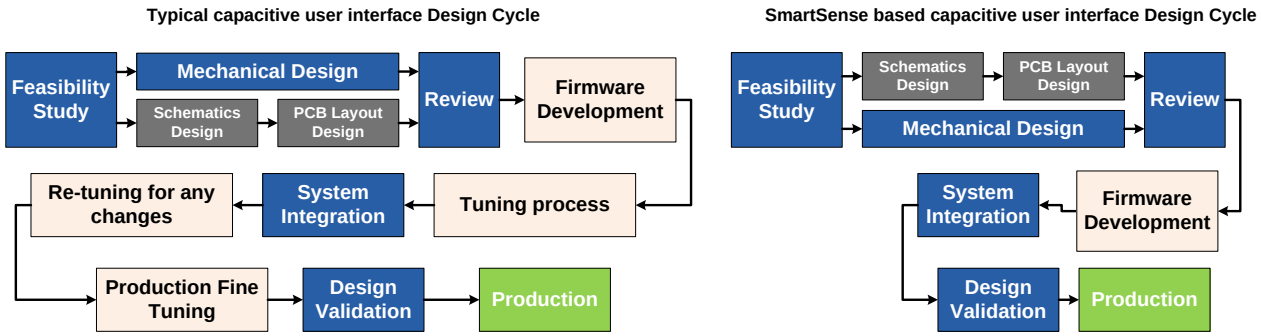
图 2-8. 15- 英寸模型笔记本电脑多媒体按键的设计，该模型与 21- 英寸模型有相同的功能和按钮大小



2.3.1.2 缩减设计周期

通常情况下，电容传感器接口设计最费时的任务是固件开发和传感器调校。 当相同的设计移植到不同模型，PCB 的机械尺寸或传感器 PCB 布局发生变化时，带有典型触摸感应控制器的传感器必须重调。 但基于 SmartSense 的设计解决了这些挑战，因为该设计需要较少固件开发工作，无需调校，也无需重调。 这显著缩短典型设计周期。 图 2-9 对典型触摸感应控制器和基于 SmartSense 的设计周期进行了比较。

图 2-9. 典型电容界面设计周期比较



3. CapSense 设计工具

3.1 概述

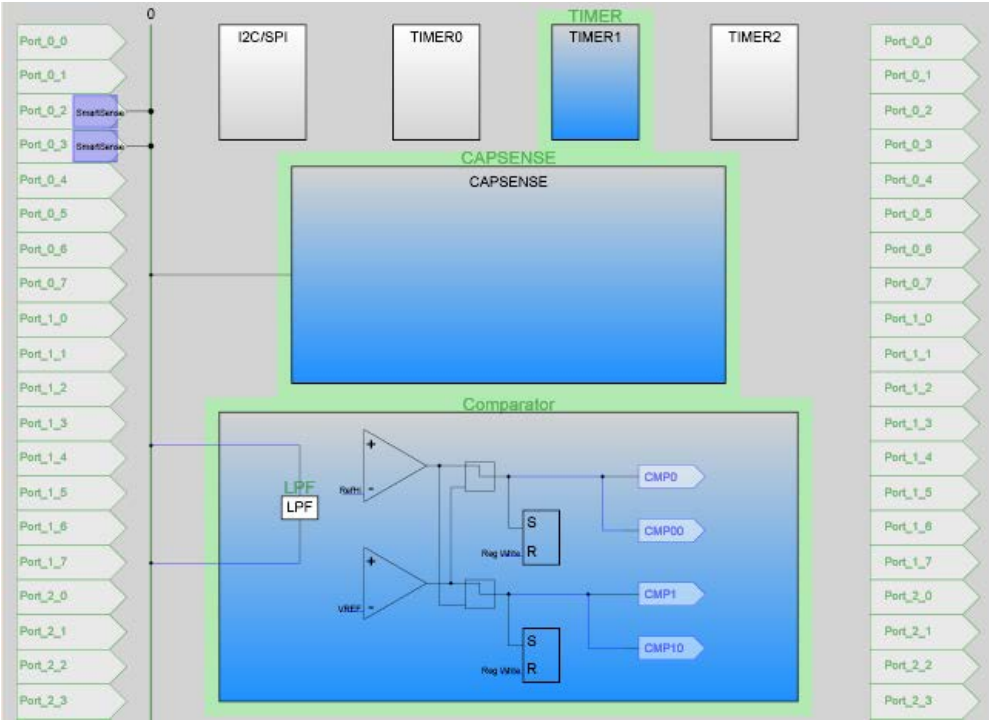
赛普拉斯提供一系列硬件和软件工具来开发 CapSense 电容式触摸感应应用。 CY8C20xx6A/H/AS 系列的基本开发系统包括以下部分。 有关订购信息，请参见 资源 。

3.1.1 PSoC Designer 和用户模块

您可利用赛普拉斯专有的集成设计环境（PSoC Designer)，配置模拟和数字模块、开发固件、调校及调试设计。 在拖放式设计环境中，使用用户模块库开发应用。 要配置用户模块，可通过器件编辑器 GUI 完成或通过固件写入特定的寄存器中。 PSoC Designer 带有内建 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。

CSA_EMC 使用开关式电容电路、模拟复用器、比较器、数字计数功能和高层软件程序 (API) 实现了电容式触摸传感器。 其他模拟和数字外设的用户模块可用于实现附加功能，如 I^2C、SPI、TX8 和定时器。

图 3-1. PSoC Designer 器件编辑器



CapSense 用户模块入门

在 PSoC Designer 中创建一个新的 CY8C20xx6A/H/AS 项目：

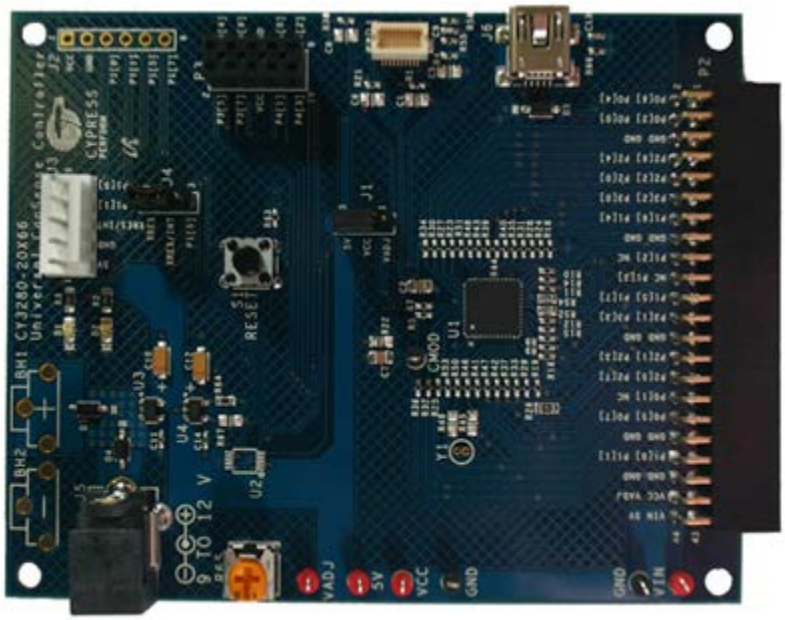
1. 使用 CY8C20xx6 创建一个新的 PSoC Designer 项目作为目标器件。
2. 选定并放置 CSD/CSA_EMC/SmartSense 用户模块。
3. 右键单击用户模块以访问用户模块向导。
4. 设置按键传感器个数、滑条配置和引脚分配及相关项。
5. 设置引脚和全局用户模块参数。
6. 生成应用，并切换到应用编辑器。
7. 根据用户模块数据手册调整示例代码，使按键或滑条生效。

如需了解创建 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册。 如需了解 CapSense 用户模块的代码示例，请参见代码示例。

3.1.2 通用 CapSense 控制器套件

通用的 CY3280-20xx6 CapSense 控制套件使用预定义的控制电路和插入硬件，简化了原型设计和调试过程。 套件包括 MiniProg，用作编程；也包括 I^2C-to-USB 桥接器硬件，用作调校和信息获取。

图 3-2. CY3280-20xx6 CapSense 控制套件



3.1.3 通用的 CapSense 控制器模块板

赛普拉斯模块板的特色是采用不同类型的传感器、LED 及界面，以满足应用所需。

- CY3280-BSM 简单按钮模块

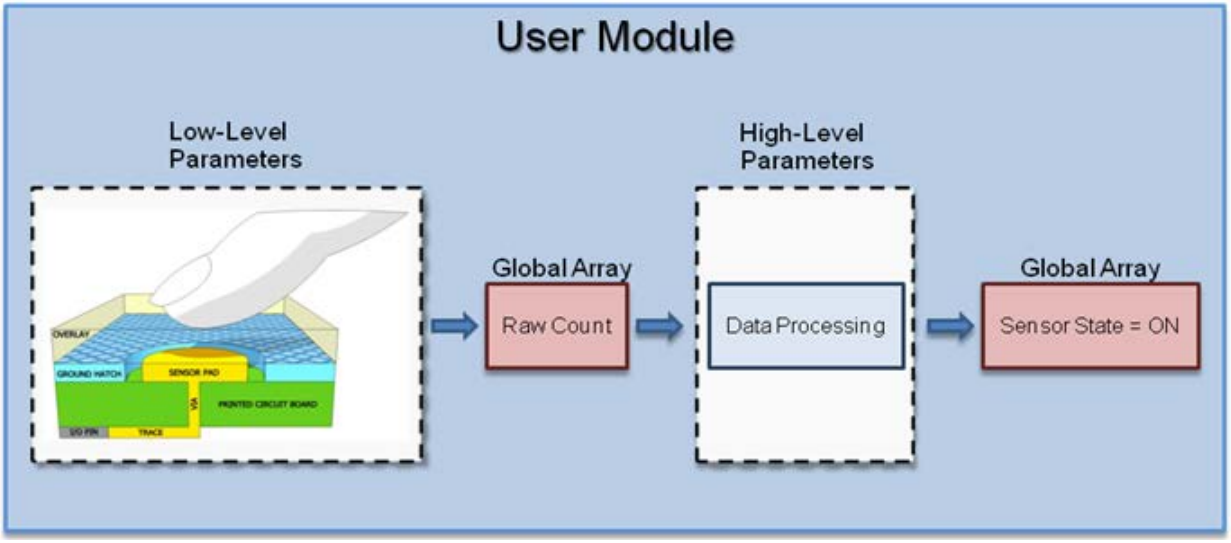
- CY3280-BMM 矩阵按钮模块
- CY3280-SLM 线性滑条模块
- CY3280-SRM 辐射状滑条模块
- CY3280-BBM 通用 CapSense 原型设计模块

3.1.4 CapSense 数据查看工具

在 CapSense 设计中，出于调校及调试目的，您经常需要监控 CapSense 数据（原始计数、基准线、差值等等）。CapSense 数据查看工具有两种：MultiChart 和 Bridge Control Panel。 有关这些工具的详细介绍，请参阅应用笔记 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



用户模块包含从物理传感到数据处理的整个 CapSense 系统。 用户模块的行为通过使用不同的参数进行定义。 这些参数影响传感系统的不同部分，可分为低级参数和高级参数，使用全局数组进行彼此沟通。

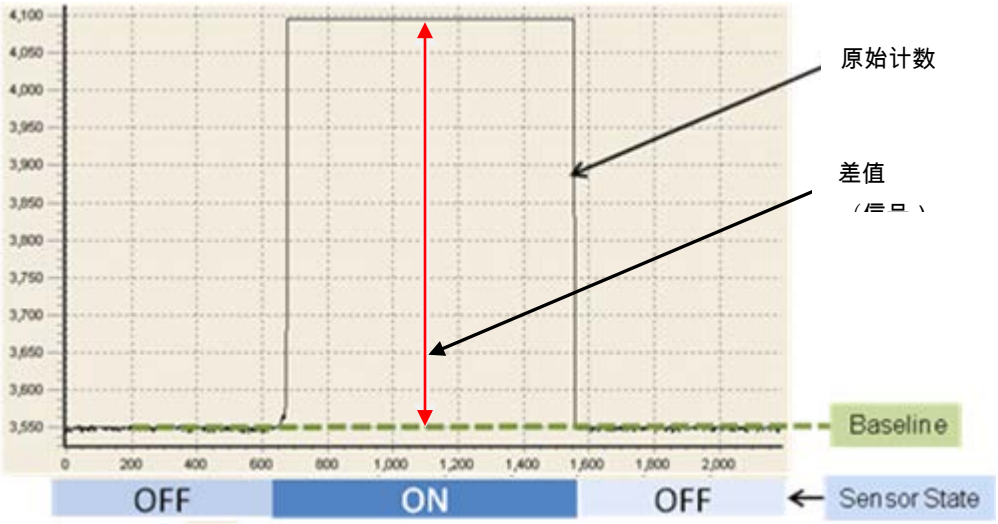
低级参数在物理层定义传感方法的行为，并涉及从电容到原始计数的转换，如扫描传感器的速度和分辨率。 每一类型的感测方法有唯一的低级参数。在 CSD 用户模式低级参数、 CSA_EMC 用户模式低级参数和 SmartSense 用户模块参数中会有详细描述。

高级参数，如去抖动计数和噪声阈值，定义了如何处理原始计数，以生成信息的。生成的信息包括：传感器 ON/OFF 状态和手指在滑条上的预估位置等。 这些参数在所有的感测方法中都相同。在用户模块高级参数中会有详细描述。

3.3 CapSense 用户模块全局数组

在开始学习 CapSense 用户模块参数之前，您必须熟悉 CapSense 系统使用的特定全局数组。 这些数组不能手动更改，但如需调试，您可对其进行检查。

图 3-4. 原始计数、基准线、差值、传感器状态



3.3.1 原始计数

CapSense 控制器中的硬件电路测量传感器的电容。 硬件电路以数字形式存储结果，这样称为原始计数，它调用了用户模块 API UMname_ScanSensor() 函数，这里 UMname 可以是 CSD、SmartSense 或者 CSA_EMC。

传感器的原始计数与其电容成正比。 原始计数随着传感器电容值的增加而增加。

传感器的原始计数值存储在 *UMname*_waSnsResult[] 整数数组中。 此数组定义在头文件 *UMname.h* 中。

3.3.2 基准线

环境渐变（如温度、湿度变化）会影响传感器的原始计数，从而导致计数发生变化。

用户模块使用了复杂的基准线算法来补偿这些变化。 该算法是通过使用基准线变量实现的。 这些基准线变量记录原始计数值中的所有渐变。 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又馈送到数字低通滤波器中。

基准线算法通过用户模块 API *UMname*_UpdateSensorBaseline 执行，这里 *UMname* 可以是 CSD、SmartSense 或者 CSA_EMC。

传感器基准线值存储在 *UMname*_waSnsBaseline[] 整数数组中。 此数组定义在头文件 *UMname.h* 中。

3.3.3 差值（符号）

差值，也被称为传感器信号，定义为传感器原始计数值与基准线值的差。 当传感器出于非活动状态，差值是零。 激活的传感器（通过触摸）产生正差值。

传感器的差值存储在 *UMname*_waSnsDiff[] 整数数组中，这里 *UMname* 可以使 CSD、 SmartSense 或 CSA_EMC。此数组定义在头文件 *UMname.h* 中。

差值变量通过用户模块 API*UMname*_UpdateSensorBaseline() 更新。

3.3.4 传感器状态

传感器状态代表了物理传感器的活动/非活动状态。 手指触摸时，传感器状态从 0 到 1 ， 手指放开时，回到 0 状态。

传感器状态存储在 *UMname*_baSnsOnMask[]字节数组中，这里 *UMname* 可以是 CSD、SmartSense 或 CSA_EMC。此数组定义在头文件 *UMname.h* 中。 每个数组元素存储八个连续传感器状态。

传感器状态通过用户模块 API *UMname*_bIsAnySensorActive() 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。 图 3-5 列出了 CSA_EMC 用户模块参数及其分类。 图 3-6 列出了 CSD 用户模块参数及其分类。

图 3-5. PSoC Designer – CSA_EMC 参数窗口

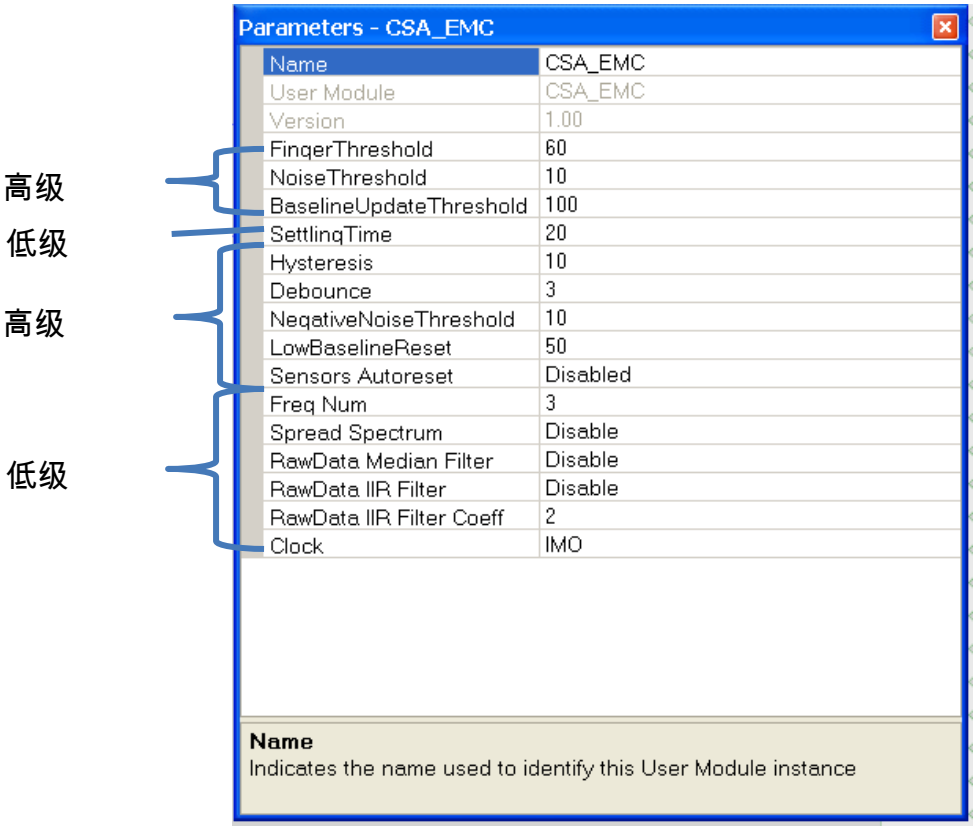
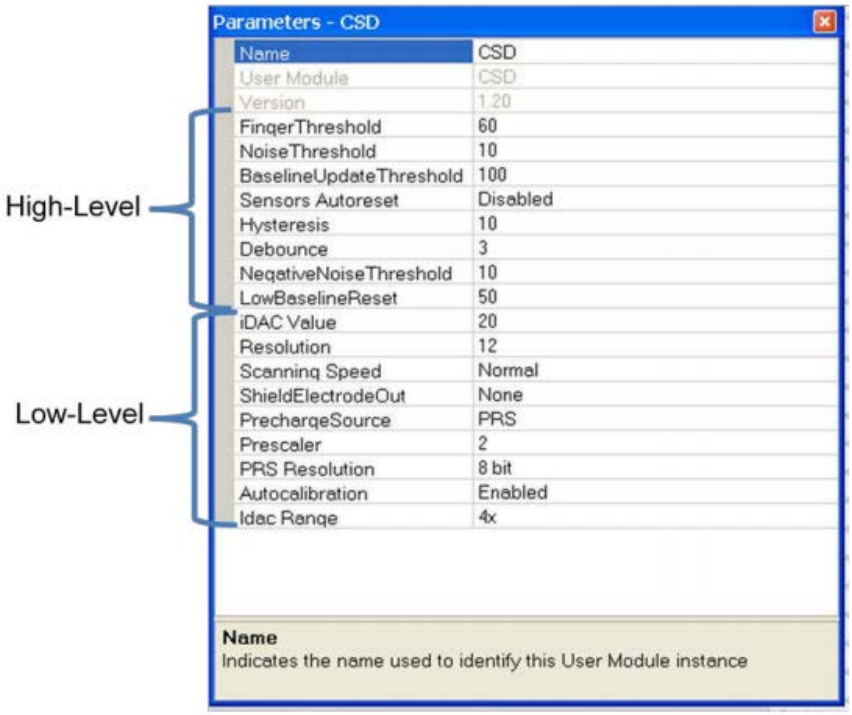


图 3-6. PSoC Designer – CSD 参数窗口



3.4.1 用户模块高级参数

3.4.1.1 手指阈值

用户模块通过手指阈值参数来判断传感器是否处于激活状态。 若传感器的差值大于手指阈值，则传感器为激活状态。该定义假设迟滞水平为 0，去抖值为 1。

可能值为 3 到 255。

更多建议值，请参见高层 API 参数。

3.4.1.2 迟滞

设定迟滞可防止传感器的激活状态受系统噪声影响，保持状态稳定。 公式 4 中给出了迟滞函数。 该公式假设去抖值为 1。

图 3-7. 迟滞值为 0 时，传感器状态与差值的关系

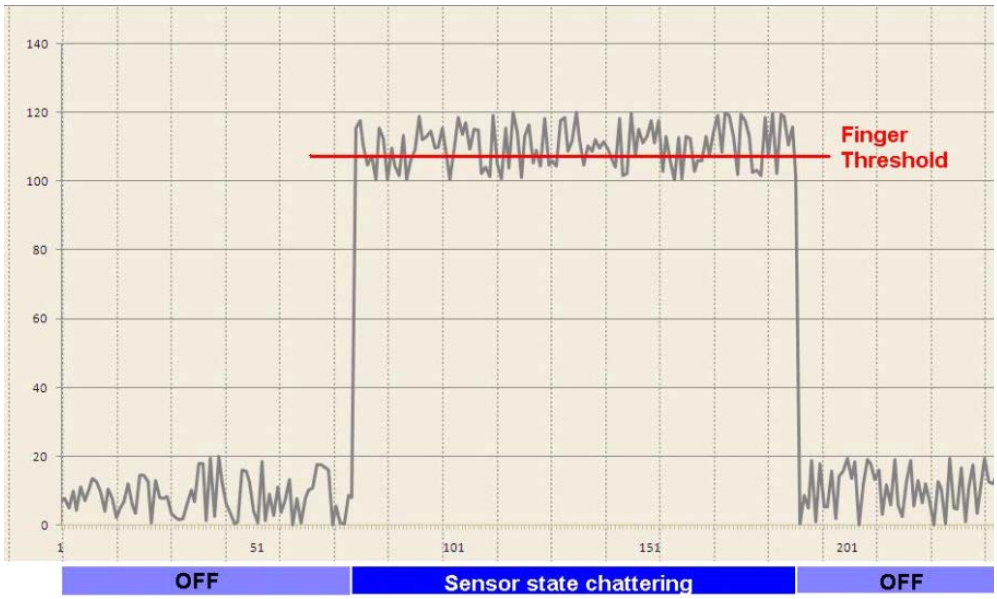
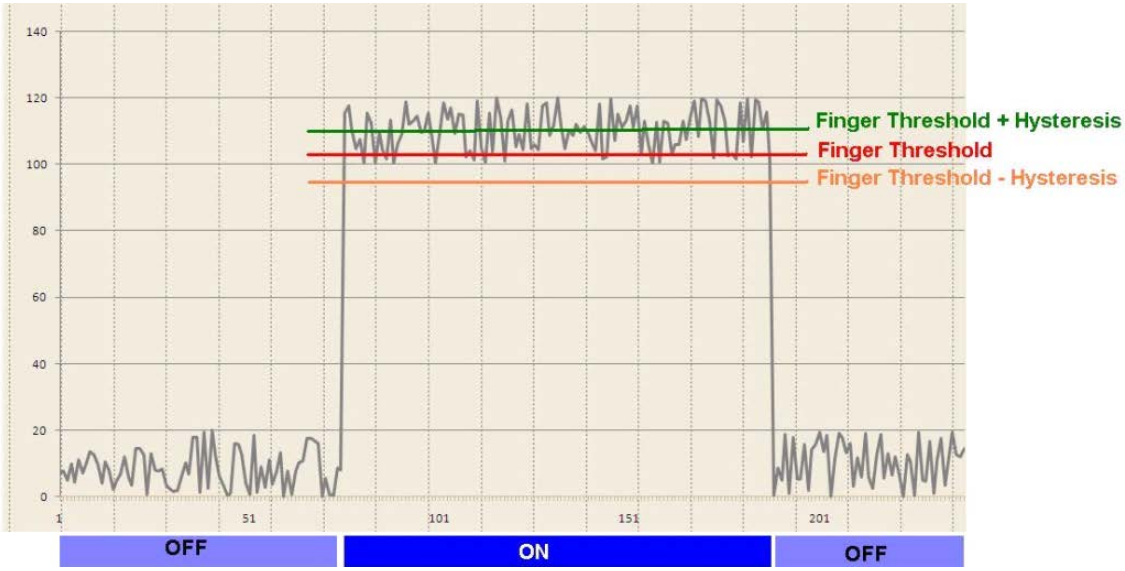


图 3-8. 迟滞值不为 0 时，传感器状态与差值的关系



若差值 ≥ 手指阈值 + 迟滞，传感器状态 = 开

若差值 ≤ 手指阈值 − 迟滞，传感器状态 = 关 Error! Bookmark not defined.

公式Error! Bookmark not defined.4

可能值为 0 到 255。

更多建议值，请参见高层 API 参数高层 API 参数

3.4.1.3 去抖

去抖值用于防止原始计数中的尖峰脉冲将传感器状态由关闭变为开启。 若要使传感器状态从关变为开，差值必须大于手指阈值与迟滞水平之和保持去抖指定的次数。

可能值为 1 到 255。 设置为 1 则不提供去抖。

更多建议值，请参见高层 API 参数。

3.4.1.4 基准线更新阈值

如前所释，基准线变量会追踪记录原始计数中的渐进变化。 换言之，基准线变量记录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，时间常量就越高。

可能值为 0 到 255。

更多建议值，请参见高层 API 参数。

3.4.1.5 噪声阈值

用户模块使用噪声阈值来解析原始计数中噪声计数的上限值。 对于单个传感器，当原始计数大于基准线且两者差值大于该阈值时，基准线更新算法将暂停。

对于滑条传感器，当差值大于噪声阈值时，质心计算将暂停。

可能值为 3 到 255。 若用户模块操作正确，噪声阀值将不会高于手指阀值与迟滞的差值。

更多建议值，请参见高层 API 参数。

3.4.1.6 负噪声阀值

负噪声阀值使户模块获取原始计数中噪声计数的下限值。 当原始计数低于基准线且两者差值大于负噪声阀值时，基准线更新算法将暂停。

可能值为 0 到 255。

更多建议值，请参见高层 API 参数。

3.4.1.7 低基准线复位

低基准线复位参数与负噪声阀值参数协同工作。 如果采样到的计数值低于基准线与负噪声阈值之差，且次数达到指定的采样次数，基准线会将设置为新的原始计数值。 此参数实际上是对重设基准线所需的异常低的采样数值进行计数。 用于更正启动时手指已经在传感器上面的情况。

可能值为 0 到 255。

更多建议值，请参见高层 API 参数。

3.4.1.8 传感器自动复位

传感器自动复位参数决定了基准线是随时更新还是只有在差值小于噪声阀值时更新。

如果传感器自动复位为启用状态，基准线将随时更新。 此设置限制传感器的最大持续时间，但可在无任何物体碰触传感器而原始计数突然上升的情况下，阻止传感器始终打开。 较大的电源电压波动、高能射频噪声源或极强烈的温度变化都会导致计数突然上升。

如果传感器自动复位为禁用状态，只有在差值小于噪声阀值参数时进行基准线更新。

可能值为“启用”和“禁用”。

更多建议值，请参见高层 API 参数。

3.4.2 CSD 用户模式低级参数

除高级参数外，CSD 用户模式还包括一些低级参数。 这些参数是 CSD 感测方法所特有的，决定了如何从传感器中获取原始计数数据。

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，使原始计数占整个范围的 50% 到 70%。 可以使用用户模块 API CSD_SetIdacValue() ，在运行状态下更改此参数。

可能值为 1 到 255。

3.4.2.2 分辨率

此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}。 增大分辨率可提高触摸检测的灵敏度，但会减少扫描时间。

可能值为 9 到 16 位。

表 3-1. 分辨率和扫描速度

分辨率	单个按钮扫描速度 (µs)			
	超快	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

3.4.2.3 扫描速度 (Scanning Speed)

该参数设置传感器扫描速度。 尽管扫描速度快能使反应时间缩短，但慢速扫描有以下优势：

- 信噪比提高

- 更好地应对电源和温度的变化
- 很少需要系统中断延迟；可以处理较长的中断

可能值为超快、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值。 该参数用于选择将屏蔽电极输出路由至何处。

可能值为 P0[7] 和 P1[2]。

3.4.2.5 预充源

此参数选择预充电开关的时钟源。

可能值为 PRS 和预分频器。 大多数情况下，使用 PRS 源可获得较好的抗 EMI 能力和较低的辐射。

3.4.2.6 预分频器

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

可能值为 1、2、4、8、16、32、64、128 和 256。

3.4.2.7 PRS 分辨率

此参数更改 PRS 序列长度。

可能值为 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 位设置无法提供很好的信噪比，请使用 8 位设置。

3.4.2.8 Autocalibration

启用“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)，则根据设备编辑器中设置的 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} 参数计算原始计数。

“自动校准” (Autocalibration) 使用 ROM 和 RAM 资源，会增加启动时间。 “自动校准” (Autocalibration) 不会自动选择 I_{DAC} 范围值。 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”(Autocalibration) 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 会标度 I_{DAC} 的输出范围为范围的两倍。

可能值为 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 用户模式还包括一些低级参数。 这些参数是 CSA_EMC 感测方法所特有的，决定如何从传感器中获取原始计数数据。

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。 基于公式 5 来选择建立时间。

$$\textit{建立时间} \geq 10 \times R_{SERIES} \times C_P \qquad \text{公式 4}$$

其中：

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

C_P = 传感器基础电容值

可能值为 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 将启用高级算法。 启用高级扫描算法会将扫描时间和 RAM 消耗增加三倍。

可能值为 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，启用扩频。

可能值为 1（启用）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。 启用该滤波器会消耗（传感器数 × 2 × Freq Num）字节的 RAM 以及 100 字节的闪存。 它默认为禁用。

可能值为“启用”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（原始数据）中的噪音。 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。 启用此滤波器会额外消耗 100 字节的闪存。 它默认为禁用。 默认 IIR 系数为 0.5。

可能值为“启用”和“禁用”。

3.4.3.6 原始数据 IIR 滤波系数

此为原始计数 IIR 滤波器的系数。

可能值为 2 （½ 上一样本 + ½ 当前样本）和 4 （¾ 上一样本 + ¼ 当前样本）。

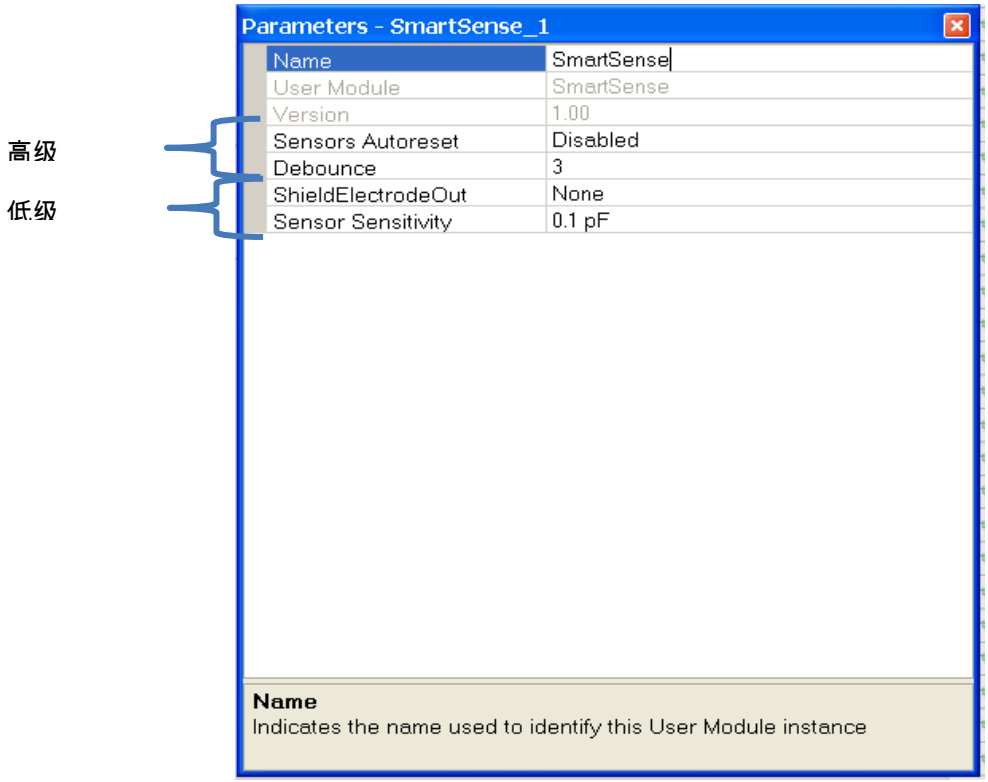
3.4.3.7 时钟

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。 较大的接近传感器的灵敏度可能会下降。 此种情况下，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。

可能值为 IMO、IMO/2、IMO/4 和 IMO/8。

3.4.4 SmartSense 用户模块参数

图 3-9. PSoC Designer SmartSense 参数



3.4.4.1 屏蔽电极输出

屏蔽电极用于降低寄生电容值。 该参数用于选择将屏蔽电极输出路由至何处。

可能值为 P0[7] 和 P1[2]。

3.4.4.2 传感器灵敏度

该参数用于提升或降低传感器的灵敏度。

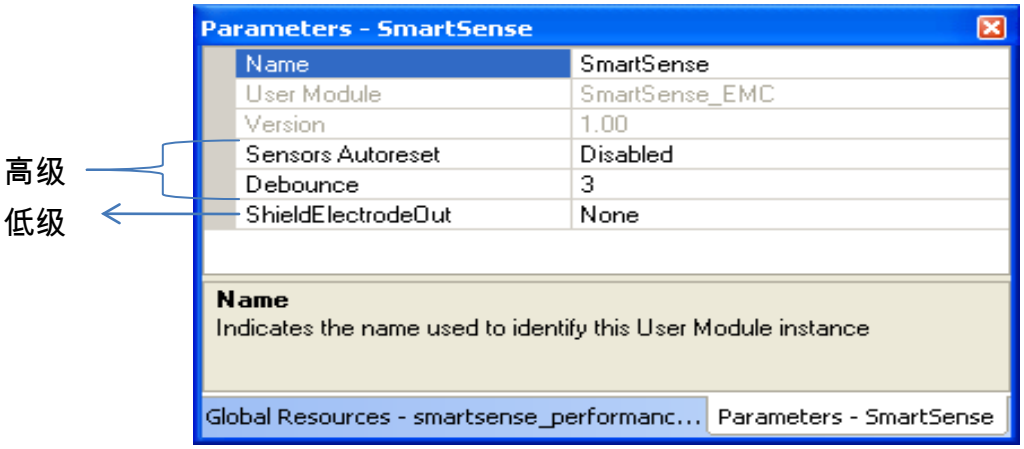
可能值为 0.1 pF、0.2 pF、0.3 pF 和 0.4 pF。

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 用户模块的全局数组。 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 3-10. 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值。 该参数用于选择将屏蔽电极输出路由至何处。

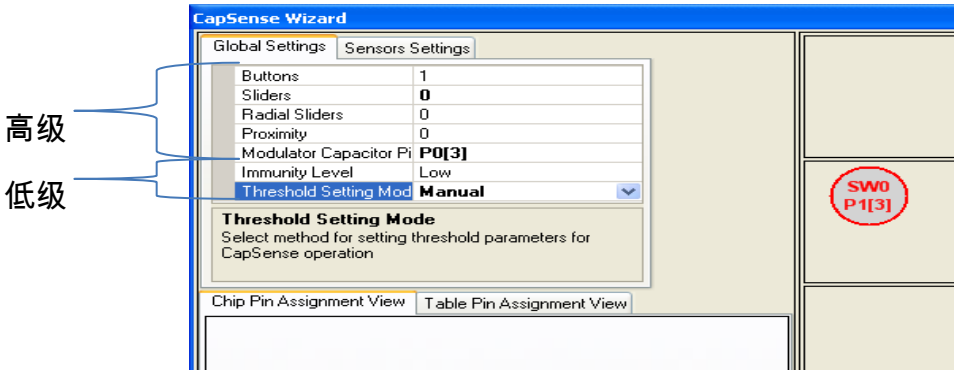
可能值为 P0[7] 和 P1[2]。

3.4.5.2 抗干扰级别

该参数定义用户模块对于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。 相较于低级抗干扰，设置中级抗干扰会消耗两倍的扫描时间和 RAM 存储空间，设置高级抗干扰会消耗三倍的扫描时间和 RAM 存储空间。

可能值为低、中和高。

图 3-11. PSoC Designer SmartSense_EMC 全局设置

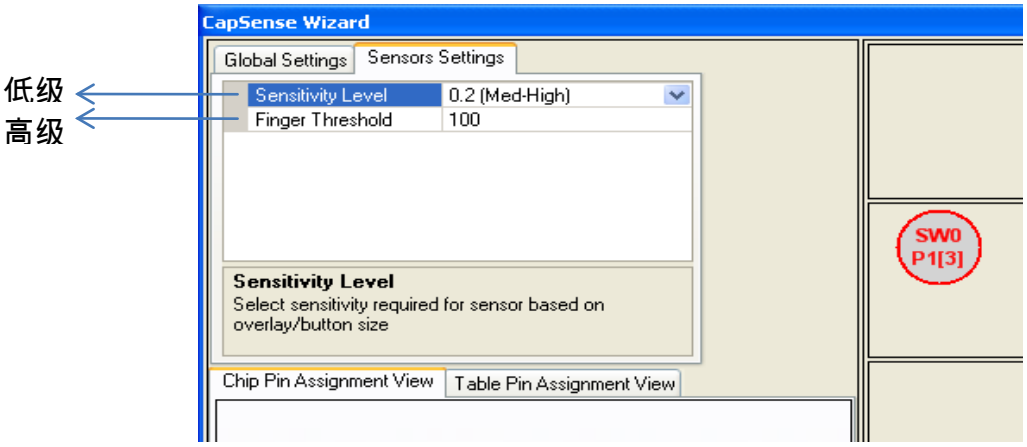


3.4.5.3 阈值设置模式

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。 相较于手动阈值模式，自动阈值模式会消耗更多 RAM。

可能值为手动和自动。

图 3-12. PSoC Designer SmartSense_EMC 传感器设置



3.4.5.4 传感器灵敏度

该参数用于提升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 进行性能调节

最佳的用户模块参数是根据面版布局、按钮尺寸、外覆层材料和应用需求设置的。 有关这些因素，请参见设计考虑事项。 调校是确定最佳的参数设置，以创造健壮可靠的传感器运行环境。

4.1 通用注意事项

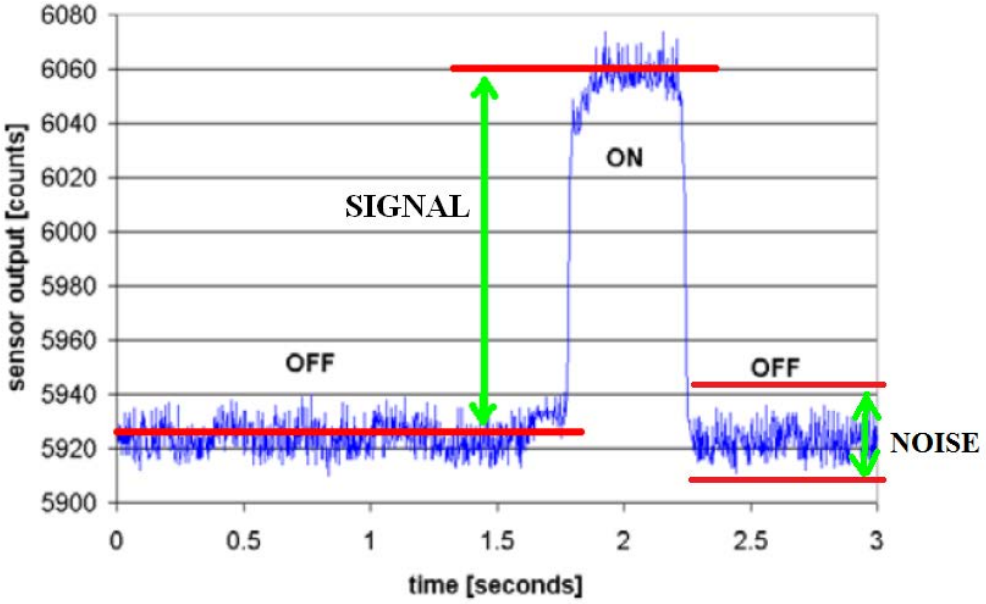
4.1.1 信号，噪声，信噪比

调校较好的 CapSense 系统能可靠地识别传感器的开关状态。 要达到此种性能，CapSense 信号强度必须明显大于 CapSense 噪声强度。 可以使用一个叫做信噪比 (SNR) 的数值比较 CapSense 信号和 CapSense 噪声。 在探讨信噪比的含义之前，首先需要定义触摸感应下的信号和噪声。

4.1.1.1 CapSense 信号

CapSense 信号是手指放置在传感器时，传感器响应的变化，如图 4-1 所示。 传感器的输出值为电子计数值，由计数器追踪记录感应器电容而来。 在本例中，没有手指触摸时，传感器的平均计数值是 5925。 手指触摸时，平均输出数值增大到 6060。 传感器信号追踪手指触摸前后数值的变动，信号值 = 6060 - 5925 = 135。

图 4-1. CapSense 信号和噪声



4.1.1.2 CapSense 噪声

CapSense 噪声是手指不在传感器时，传感器响应的峰至峰的变动，如图 4-1 所示。 在本例中，没有手指触摸时的输出波形在最大 5938 和最小 5912 的值之间。 CapSense 噪声是波形最大值和最小值的差，因而噪声值 = 5938 - 5912 = 26。

4.1.1.3 CapSense 信噪比

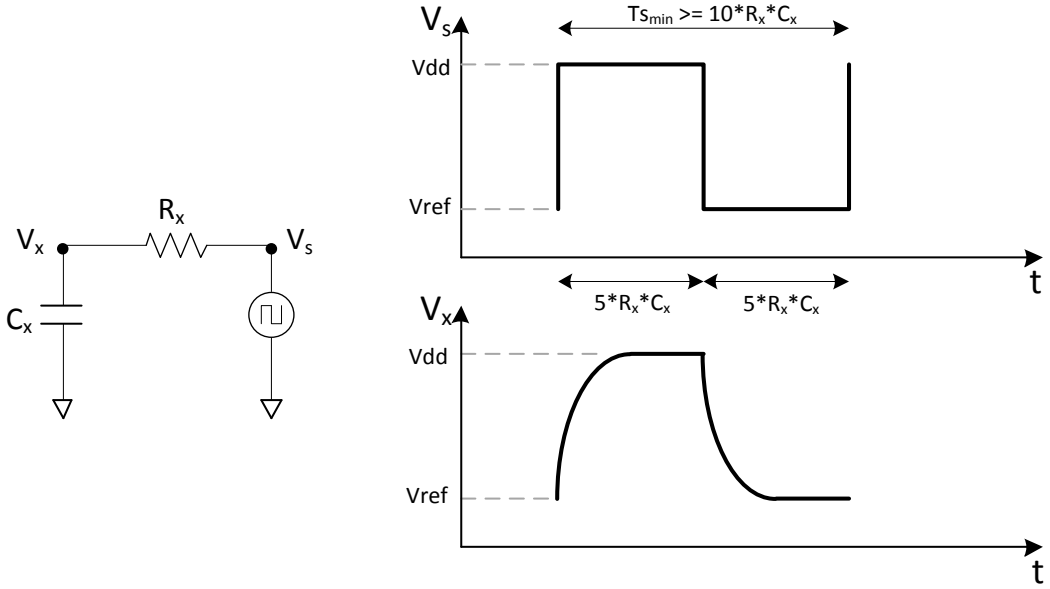
CapSense 信噪比是信号和噪声的简单比率。 继续此例，如果信号值是 135，噪声值是 26，那么信噪比是 135:26，约分为 5.2:1。 CapSense 信噪比的最低推荐值为 5:1，这意味着信号的强度是是噪声强度的 5 倍。 通常会在固件中植入滤波器以降低噪音。 更多信息，请参见软件滤波。

4.1.2 充电/放电频率

要在调校过程中达到最高灵敏度，传感器电容在每个循环中必须满充满放。 两种充电/放电状态路径以某种频率在两种状态之间转换，此频率由两个参数设定，分别为 CSA_EMC 用户模块中的“时钟”参数和 CSD 用户模块中的“预充电时钟”参数。

充电/放电路径包含会降低充放转换速度的串联电阻。 充放转换的变动频率以一个 RC 时间常数为特征。此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



把充电/放电频率设定到与 RC 时间常数相容的水平。 总体来说，应该设定每个转变的时间周期为 5RC，每个周期有两个转变（一个充电，一个放电）。最短时间按周期和最大频率的公式为：

$$Ts_{min} = 10 \times R_X C_X$$ 公式 5

$$fs_{max} = \frac{1}{10 \times R_X C_X}$$ 公式 6

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 kΩ

C_X= 24 pF

此例中时间常数和最大前端开关频率为：

Ts_{min} = 0.34 µs

fs_{max} = 3 MHz

4.1.3 验证基准线更新阈值的重要性

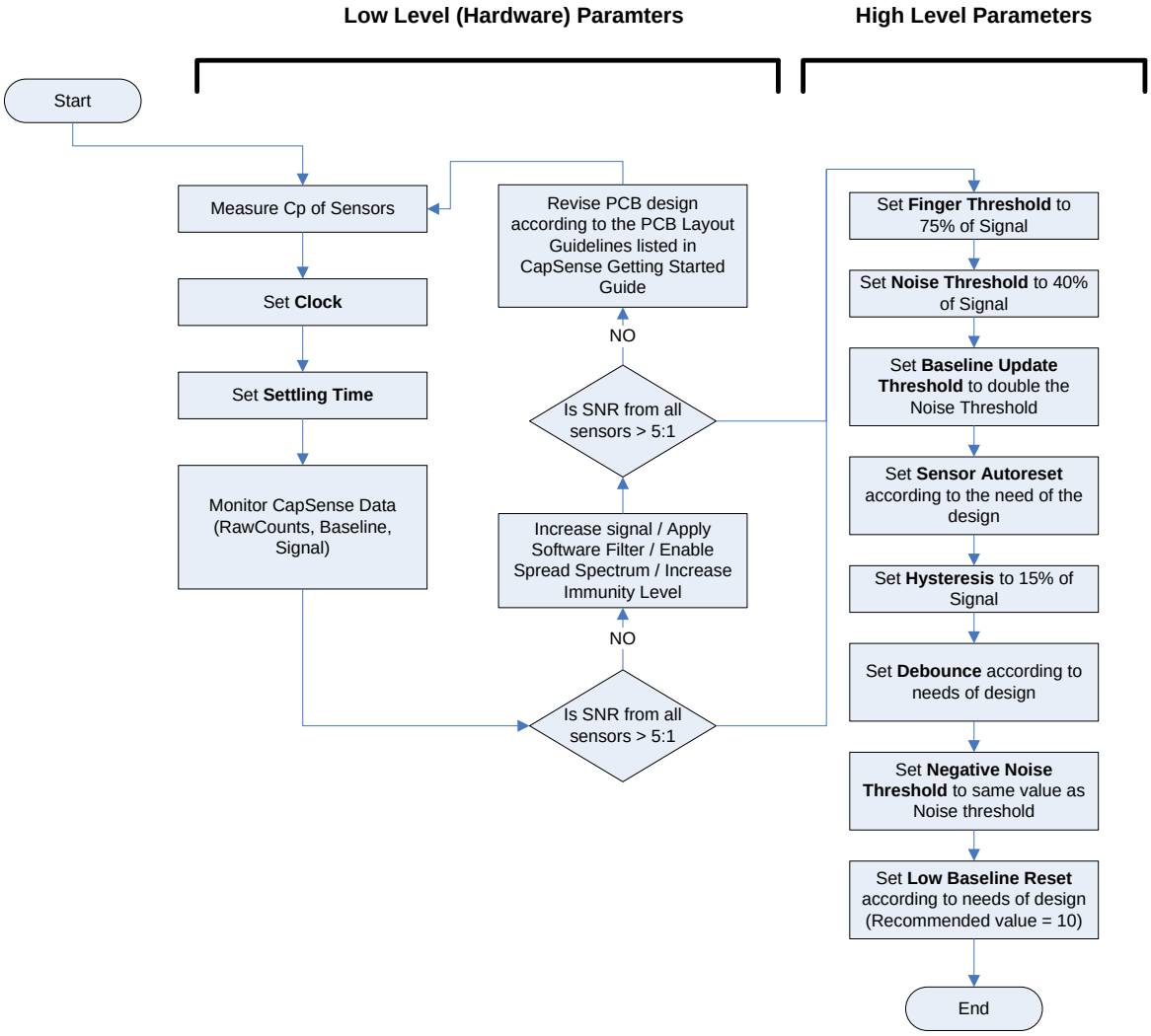
温度和湿度都会造成平均计数随时间浮动。 基准线是 CapSense 测量的参照计数水平，是补偿环境因素影响的重要因素。 诸如手指触摸和手指离开的状态的高级判定，都依照基准线建立的参照水平而定。 由于每个传感器都有与配套的寄生电容，每个电容传感器都独立拥有基准线。

基准线以特定的频率追踪计数器的变化。此频率由基准线更新阈值参数设定。 请确保更新频率与预期应用相匹配。 如果更新频率太快，基准线会抵消掉任何手指触摸所造成的变动，导致检测不到手指的移动。 如果更新频率太慢，相对缓慢的环境变动可能会被误以为是手指的动作。 开发过程中，应当验证基准线更新阈值的设置。

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 用户模块参数可以分为两类：低级（硬件）参数和高级参数。 这些类别中的参数以不同的方式影响电容式感应系统的行为。 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 用户模块参数应首先从硬件参数开始。

图 4-3. CSA_EMC 用户模块参数调校流程图



4.3 建议使用的 CSA_EMC中的 C_{INT} 值

调校起始时建议使用的 C_{INT} 值为 1.2 nF。 调校过程中，若发现传感器信号不足，信噪比不足 5:1，可以增大 C_{INT} 值。建议 C_{INT} 的最大值为 5.6 nF。 推荐使用 X7R 或者 NPO 类型的电容以保证不同温度条件下 C_{INT} 的稳定性，同时电容的电压应该不低于 5 V。

4.4 测量传感器 C_P

调校过程的首要步骤是测量传感器寄生电容 (C_P)。 测量的分步步骤如下：

第一步： 设定 **CPU_CLK** 等于 SysClk/2。

第二步： 设定**时钟**等于 IMO/8。

第三步： 设定**建立时间**为 255。

第四步： 回读由特定传感器的算法设定的 I_{DAC} 代码。 读取值将存储在 `CSA_EMC_baDACCodeBaseline[]` 数组中。

第五步：根据 Idac 设置测量 Idac 电流。
使用以下代码创建一个 PSoC 设计者项目。 代码将 I_{DAC} 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;
```

将测量仪放置于 P1[4] 和接地之间，并测量当前电流。 设此电流值为 $I_{MEASURED}$。

第六步： 利用下面的公式计算出 C_P。

$C_P = I_{MEASURED} / ((IMO/8) * 1.3)$

传感器 C_P 的数值也可通过 LCR 测量仪测量得出。 把 LCR 测量仪的一端连接至传感器针脚，另一端连接至 GND 以测量 C_P。

4.5 估算 CSA_EMC 时钟

表 4-1 展示了推荐的预充电时钟频率，是基于传感器 C_P 的方程。 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

	CSA_EMC 时钟		
C_P (pF)	**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。

$$\text{设置时间 Settling Time} = \frac{(5*\text{Cint})}{\left(\text{Clock}*\text{Cp}*25*\left(\frac{1}{\text{Fcpu}}\right)\right)}$$ 公式 7

其中：

- C_{INT} = 积分式电容的值
- 时钟 = 预充电时钟频率 (CSA_EMC Clock)
- C_P = 传感器寄生电容的值
- F_{CPU} = CPU 时钟频率

4.7 监控 CapSense 数据

查看 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 降噪

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：

1. 使用软件过滤器 — 更多细节，请参见软件滤波。
2. 激活扩频功能 — 更多细节，请参见扩频。
3. 增加抗噪级别 — 更多细节，请参见 Freq Num。

4.8.2 增大信号强度

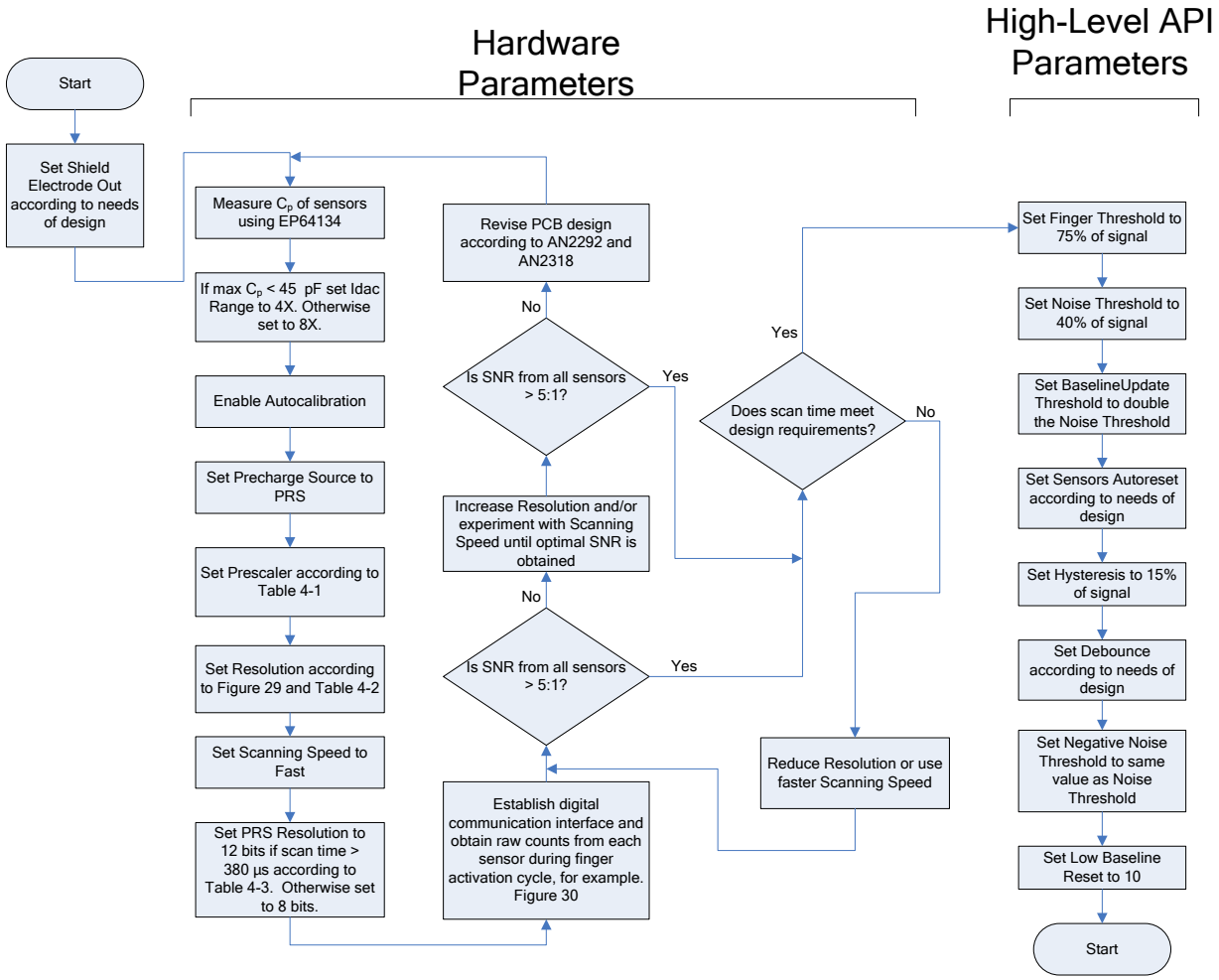
也可使用增大信号强度的方法增加信噪比。 也可使用下面两种方法之一来增大信号强度：

- 增大由宏定义的值 `CSA_EMC_BASELINE`。 这个宏位于 *CSA_EMC.inc 文件中*。 默认条件下，指定宏的值为 0x0800
- 增大 C_{INT} 电容的值

4.9 调校 CSD 用户模块

图 4-4 展示的是 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。 这些类别中的参数以不同的方式影响电容式感应系统的行为。 然而，每个传感器的灵敏度之间有些补充的关系，它们是硬件参数设定和许多高级参数设置决定的。 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 CSD 用户模块参数应首先从硬件参数开始。

图 4-4. 调校 CSD 用户模块



硬件参数配置硬件，这些硬件中，使用 CSD 方法用来把每个传感器的物理电容值转换为数字代码。 此部分描述这些参数，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，硬件参数是设计中应用到所有 CapSense 传感器的全局设置。 设计中，如果的传感器的寄生电容的总值 (C_P) ，或者是传感器的灵敏度，或者两者都在一个较大范围内浮动，全局硬件参数设置可能会不适合所有的传感器。 此类例子中，设置 IDAC 值(SetIdacValue(i))，(SetPrescaler(i))，和 扫描模式(SetScanMode(i)) 的 API 函数能够用来配置每个传感器各自的硬件参数，而 (i) 可调用 ScanSensor(i) API 函数传感器的索引。

表 4-2 和 表 4-4 提供了 C_P 为基准的几个关键硬件参数的推荐调校值。 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，具有的同样的外围或外覆层。 测量 C_P 值的最佳方式是使用以 CE64134 - 使用 CY8C20xx6A CapSense® 控制器测量传感器绝对电容为标题的代码案例。 此项目使用 PSoC 测量系统中每个传感器的绝对电容值，会考虑所有影响 C_P 的因素。 参见代码案例相关的文件以查阅安装和使用步骤。

4.9.1 推荐 CSD 所需的 C_{CMOD} 值

推荐 CSD 为基准的 C_{CMOD} 值为 2.2 nF。 推荐使用 X7R 或者 NPO 类型的电容以保证不同温度条件下 C_{INT} 的稳定性，同时电容的电压应该不低于 5 V。

4.9.2 屏蔽电极输出 (ShieldElectrodeOut)

激活设计中的屏蔽电极输出。

4.9.3 I_{DAC} 范围

对于 少于 45 pF 的最大传感器 C_P 项目，可使用 4X；否则，则使用 8X。

4.9.4 Autocalibration

在 CY8C20xx6A CSD 设计中，应将自动校准始终设置为激活状态。 自动校准算法能力能否成功设置 I_{DAC} 取决于预分频器的合理设置和分频器是否处于推荐大小的状态 。

4.9.5 I_{DAC} 值

当自动校准禁用后，此参数将决定当前 I_{DAC} 输出。 自动校准启用后，会使用推荐设定，该参数将被覆盖，并且失效。 自动校准禁用后，提高该参数会降低原始计数基准线，反之亦然。

4.9.6 预充源

此参数选择传感器开关时钟源。 可用选项是预分频器，使用 IMO 通过分频器，或使用 PRS，使用 IMO 通过伪随机序列发生器，提供扩频时钟。 PRS 提供高级降噪，释放更少的噪音，因而它也成为推荐预充源的默认设置。 在有些情况下，预分频器预充源可提供更高的信噪比。 然而，使用铜线路时，信噪比的提升通常比较细微，并且也不能明显看出是否益于前面提到的 PRS。

4.9.7 预分频器

预分频器是应用于 IMO 以建立预充电时钟的分频器。 合理调校 CSD 设计时，此为最重要的硬件 UM 参数。 预分频器依据所选预充源、IMO 和传感器的 C_P 值而定。 表 4-2 显示了基于这些参数的预分频器推荐设置。

表 4-2. 基于预充源、IMO 和 C_P 的预分频器设置

C_P (pF)	预充源 = PRS			预充源 = 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	注解 1	注解 1	2	1	1
7-11	2	1	注解 1	4	2	1
12-15	2	1	注解 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

注解 1 不推荐使用这种预充源、预分频器和 C_P 的组合。

4.9.8 分辨率

可选范围为 9 到 16 位。 提高分辨率会加强传感器的灵敏度，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 显示了基于 C_P 和手指电容值 C_F 的分辨率推荐设置。 C_F 是手指放置在传感器上时电容值的变化。 C_F 值取决于外覆层厚度、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，它可作为外覆层厚度和圆形传感器直径的函数。

图 4-5. 基于外覆层厚度和圆形传感器直径的手指电容值(C_F)

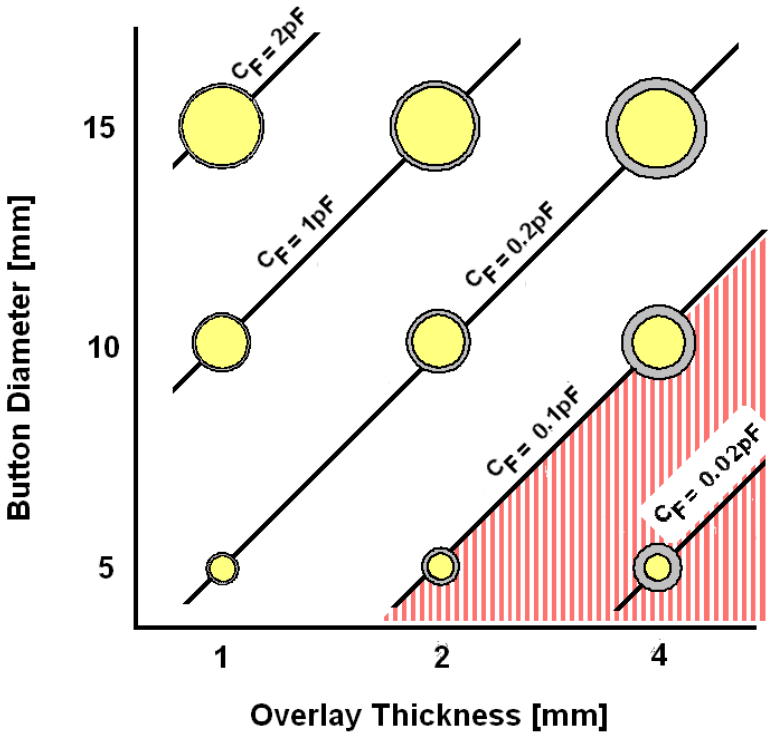


表 4-3. 基于手指电容值和 C_P 的分辨率设置

C_P (pF)	C_F = 0.1pF	C_F = 0.2 pF	C_F = 0.4pF	C_F = 0.8pF
<6	12	11	10	9
7-11	13	12	11	10
–0.24	14	13	12	11
25.00	15	14	13	12
>49	16	15	14	13

4.9.9 扫描速度 (Scanning Speed)

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 4-4 以微秒为单位，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。

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 (Scanning Speed)			
	超快	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

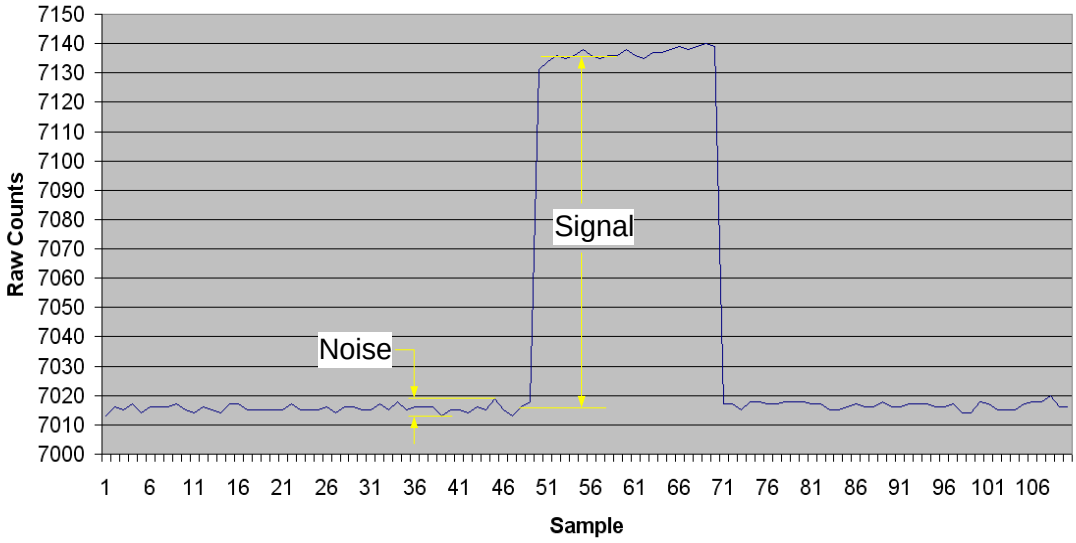
13	380	720	1400	2800
14	720	1400	2800	5600
15	1400	2800	5600	11000
16	2800	5600	11000	22000

4.9.10 高层 API 参数

高层 API 参数决定高层固件算法行为，此算法用于区分传感器的激活与噪声，并补偿由环境因素引起的信号漂移。 为了确定适当的参数取值，须与系统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：CSD_waSnsBaseline[]， CSD_waSnsResult[]，和 CSD_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、覆盖层厚度及其他系统特性。 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获取的典型原始计数值，即传感器被激活再取消激活。 叠加在数据上的标签说明了如何根据原始数据计算噪声与信号。 在适当情况下，遵循的高层参数描述包括基于噪声与信号值设置参数的方法的信息。 根据 CapSence 设计的最佳做法，为了 CapSence 系统操作的健壮性，信噪比 (SNR) 应至少为 5:1。 若信噪比低于 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 4-6. 传感器在一个手指激活周期中的典型原始计数值



4.9.11 高层 API 参数

为获得最佳参数设置，建议采用以下初始值：

- **手指阈值**： 传感器开启，原始计数设为 75%
- **噪声阈值**： 传感器关闭，原始计数设为 40%
- **负噪声阈值**： 取中值（噪声阈值的 1/2）

- **基准线更新阈值**： 设为噪声阈值的两倍
- **迟滞**： 传感器开启，原始计数设为 15%
- **低值基准线复位**： 设为 50
- **传感器自动复位**： 根据设计需求进行设置
- **去抖**： 根据设计需求进行设置

4.10 使用 SmartSense 用户模块

只要传感器寄生电容值为 5 pF - 45 pF 之间，且触摸电容值大于等于最小值 0.1 pF，便可用 SmartSense 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。 本节也介绍如何将现有的 CSD CapSense 设计导入到 SmartSense 中。

4.10.1 SmartSense 指南

在应用中使用 SmartSense 用户模块时，请遵循这些指南。

- SmartSense 要求，设计电容式用户界面时，要遵循本设计指南前面章节中记录的布局和系统设计最佳实践。
- 所有 CSD 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 用户模块运行时确定。 因为 API 会在固件上修改这些 CSD 参数，所以除非确切知道它对设计的影响，其他情况下都不应使用。
- 要将现有的 CSD 设计导入到 SmartSense 中,
  - 请确保首先从程序中移除所有设置或者修改 CSD 参数的 API。
  - 请确保设计中所有 CapSense 传感器的寄生电容值在 5 pF 和 45 pF 之间，不随环境或 PCB 生产过程变化超过此范围。
  - 请确保推荐 C_{MOD} 电容（X7R，2.2-NF，额定电压超过 5 伏）连接到用户模块向导中选择的端口引脚 C_{MOD} 。

4.10.2 差异理解

SmartSense 用户模块和标准 CSD 用户模块之间的不同是：

- SmartSense 用户模块支持的 APIs 与标准 CSD 用户模块支持的一样。 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 用户模块在运行时自动设置。
- C_{MOD} 的电容值限制为 2.2 nF。 在所有 CapSense 应用中，建议使用额定电压高于 5 V 的 X7R 电容器。
- SmartSense 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 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 C_{CMOD} 推荐值

基于 SmartSense 设计的 C_{CMOD} 推荐值为 2.2 nF。 为了保证有效工作温度下 C_{INT} 的稳定性，建议使用 X7R 或者 NPO 型电容。 电容的额定电压应不低于 5 V。

4.10.4 SmartSense 用户模块参数

该用户模块中，只有四个参数必须设置。 它们为：

- 传感器自动复位
- 去抖
- 调制器电容引脚 (Modulator Capacitor Pin)
- 灵敏度等级

4.10.4.1 传感器自动复位

此参数决定基准线的更新频率：任何时候都更新，或只有信号差值低于噪声阈值时才更新。 当设置为启用时，基准线随时更新。 虽然此设置限制传感器运行的最长时间（通常是 5 至 10 秒），但是防止了下述情况的发生：由于某种系统故障，虽未触碰传感器，原始计数却突然上升，导致传感器一直运行为 On。

4.10.4.2 去抖

去抖参数为传感器的有效转换添加去抖计数器。从释放到触发的转变，手指触摸信号应保持连续扫描超过去抖次数。该参数影响所有类似传感器。

4.10.4.3 调制器电容引脚 (Modulator Capacitor Pin)

此参数选择的引脚连接了 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]。

请注意： 必须有一个外部 2.2-nF 电容，SmartSense 才能正常工作。

4.10.4.4 灵敏度等级

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。 为了能够有效实现，设计的覆盖层越厚，所需的传感器信号就越强。 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 UM 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 信噪比时传感器灵敏度的最高值。 可从灵敏度最高值 (0.4 pF) 开始调校，并按需减少值直至满足 5:1 信噪比

4.10.5 SmartSense_EMC 用户模块的专用指南

所有适用于 SmartSense 用户模块的指南均适用于 SmartSense_EMC 用户模块。 有关 CapSense 设计和基于 SmartSense 设计的一般性说明，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、响应时间和存储器使用

当使用 SmartSense_EMC 用户模块的运行传感器时，传感器的扫描时间、传感器的响应时间以及 RAM 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”时多两倍。 抗噪模式设置为“高”时的传感器扫描时间比设置为“低”时多三倍。
- 随着扫描时间增加，传感器的响应时间也同比增加。 抗噪模式设置为“中”时的传感器响应时间比设置为“低”时多两倍。 同样，抗噪模式设置为“高”时的传感器响应时间比设置为“低”时多三倍。
- 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”）需要的 RAM 内存大约比模式为“低”时大三倍。 中等抗噪音模式需要的 RAM 内存大约比模式为“低”时大两倍。

4.10.5.2 IMO 容差和时间临界任务

SmartSense_EMC-enabled 部分的 IMO 容差为 +5% 与 –20%。

- 使用临界时间算法和逻辑时，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，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，以确保能应用固件的健壮性。

4.10.5.3 I^2C 工作速度

I^2C 接口工作频率最大为 SmartSense_EMC-enabled 部分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 I^2C 用户模块中选择 400 kHz 的时钟频率时，I^2C 接口的最大工作频率为 320 kHz。 同样，在 I^2C 用户模块中选择 100 kHz 和 50 kHz 的时钟模式，则其最大工作频率分别为 80 kHz 和 40 kHz 。
- 使用 I^2C 从控接口时，主控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、I^2C 总线连接或与 I^2C 用户模块动作不一致。
- 使用 I^2C 主控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。

$$\textit{扫描时间} = \textit{采样时间}\ (ST) + \textit{处理时间}\ (PT) \qquad \text{公式 8}$$

下表显示不同 IMO 和灵敏度水平下的采样时间值。

表 4-5. 传感器 IMO = 24 MHz 的采样时间

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 4-6. 传感器 IMO = 12 MHz 的采样时间

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 4-7. 传感器 IMO = 6 MHz 的采样时间

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 4-8 显示不同 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 : 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)、且灵敏度水平为 0.3 pF，则寄生电容值大约为 15 pF 的传感器扫描时间可根据之前的表中的数据、使用公式 9 进行计算。

之前提到配置（IMO 为 24 MHz IMO，灵敏度为 0.3 pF）的采样时间根据表 4-5 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 4-8 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。

4.10.7 SmartSense 响应时间

考虑以下使用标准 CSD 与典型 CapSense 扫描固件的应用。

- 三个 CapSense 传感器，其传感器寄生电容值为 5 pF 到 10 pF 之间
- IMO 为 12 MHz，CPU 时钟为 12 MHz
- 传感器灵敏等级为 0.4 pF
- 去抖值为 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_ScanAllSensors();
    SmartSense _UpdateAllBaselines();

    if(SmartSense _bIsAnySensorActive() )
    {
        //1 ms 固件程序例程
    }
}
```

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数，传感器应处于活动状态）。 这通常被称为 CapSense 系统的响应时间。

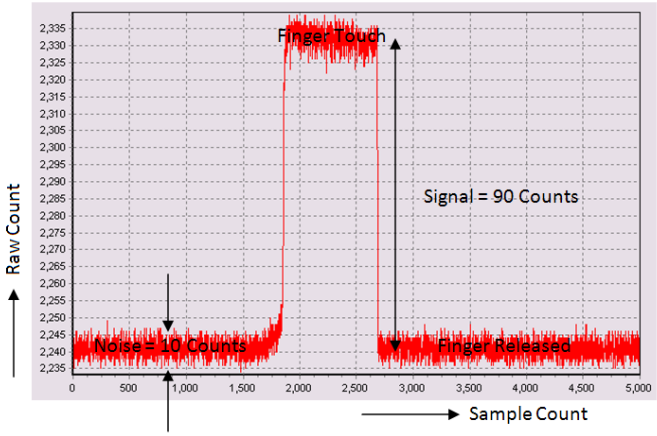
如果扫描时间随着寄生电容的变化发生相应的变化，那么由于进程变化而导致的传感器寄生电容变化，对于响应时间会产生怎样的影响？ 在这种情况下，响应时间可能会增加（反应变慢）。 这对于传感器的性能可能产生负面影响。 在下一节中将介绍如何进行健壮的固件设计。

4.10.8 使用 SmartSense_EMC 用户模块以确保低信噪比的方法

SmartSense_EMC 是先进的电磁兼容设计，该设计基于 CSD 的 SmartSense 用户模块，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健壮性。

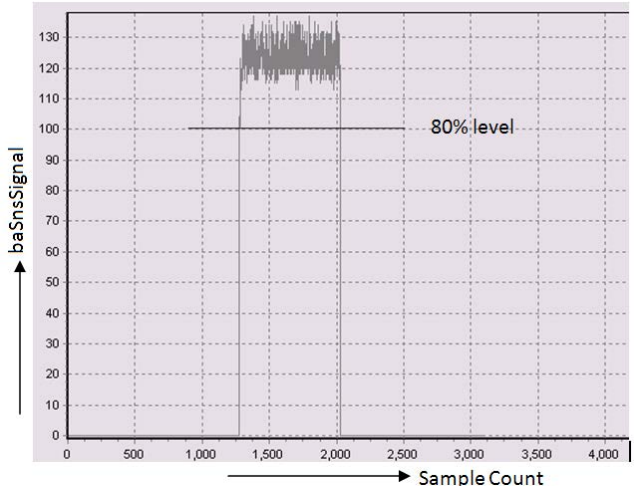
1. 建立实时监测工具，监测 CapSense 用户模块参数以测量传感器信号。 在调校期间，可以观察到传感器原始计数 (SmartSense_EMC_waSnsResult)，传感器归一化信号 (SmartSense_EMC_baSnsSignal)，和传感器手指阈值 (SmartSense_EMC_baBtnFThreshold)。 不要使用 LCD 或其他任何数字显示屏来监控计数，因为它们太慢，您无法观察到数据的动态变化。 推荐使用的数据监测工具 multi-chart 或 I2C USB Bridge Control panel。
2. 将灵敏度设置到 0.4 pF（低），并计算 SNR。 图 4-7 显示了典型手指触摸原始计数图。 根据 CapSense 最佳做法，好的设计的 SNR 应大于 5:1。 如果测得的 SNR 大于 10:1，就降低灵敏度的值到下一可能的操作，直到 SNR 的值大于 5:1，小于 10:1。

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。 为设置手指阈值，需监测传感器信号 (SmartSense_EMC_baSnsSignal)，并在触摸传感器时，设置手指阈值为传感器信号的 80% 。 该操作就此完成。 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（见以下示例代码），因为这也可能增加。 当所有传感器的寄生电容值小于 10 pF 时，固件程序执行速率是 2.45 ms。由于工艺导致寄生电容增加，扫面时间也因此增加，所以固件执行速率也会变化。

以下是一个切换端口引脚的示例代码，它基于主循环执行时间。

```
while (1)
{
        SmartSense_ScanAllSensors();
        SmartSense_UpdateAllBaselines();

        if(SmartSense_bIsAnySensorActive() )
        {
            //1 ms 固件程序例程
```

```
        }

        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提高到约为 15 pF，则扫描时间会变成 1.78 ms；因此 Port_0[1] 上的信号时间将为 5.6 ms。

如果此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），那么由于处理过程中的变化，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。

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) 的程序中设定 WDT 到期时间时就应该考虑上述事实

以下是一个简单的固件实现示例，说明如何使用定时器 16 用户模块，以得到一致的主循环执行时间。

```
// 主程序
BYTE bTimerTicks = 0;

#pragma interrupt_handler myTimer_ISR_Handler;
void myTimer_ISR_Handler( void );

void  main()
{
M8C_EnableGInt;

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

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

while( 1 )
{
        /* 扫描全部三个传感器并更新
         基准线 */
        SmartSense_ScanAllSensors();
        SmartSense_UpdateAllBaselines();

        /* Wait till timer expires or
         sleep here */

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

        if(CSDAUTO_bIsAnySensorActive() )
        {
            //1 ms 固件程序例程
        }

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

```
}

//定时器 16 ISR 程序
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。 定时器的时间应该基于最差情况下主循环执行时间。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 电容组的边界，则选择较高的扫描时间（使用表 4-6）来计算。

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 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。

5. 设计考虑事项

当设计电容式触摸感应技术的应用时，很重要的一点就是须将 CapSense 元件置入较大的框架之中。 从 PCB 布局、用户界面到终端用户操作环境，每一个细节性的层面都须注意，因为这将决定系统性能的可靠性和稳定性。欲获取更多信息，请参见 CapSense 入门.

5.1 选择覆盖层

在 CapSense 基本原理, 公式 1 为提出了手指电容

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

其中：

ε_0 = 真空介电常数

ε_r = 覆盖层的介电常数

A = 手指和传感器板重叠区域(mm^2)

D = 外覆层厚度 (mm)

如果要提高 CapSense 的信号强度，可选取高绝缘常数的覆盖层材料，减少外覆层厚度同时增大按钮尺寸。

表 5-1. 覆盖层材料绝缘强度

材料	击穿电压 (V/mm)	最小值 12 kV 下的外覆层厚度 (mm)
空气	1200–2800	10
干木块	3900	3
普通玻璃	7900	1.5
硼硅酸盐玻璃（Pyrex®）)	13,000	0.9
PMMA 塑料（Plexiglas®）	13,000	0.9
（苯乙烯共聚物板）ABS	16,000	0.8
聚碳酸酯（Lexan®）	16,000	0.8
富美家材料	18,000	0.7
FR-4 基板	28,000	0.4
聚酯薄膜—（Mylar®）	280,000	0.04

聚酰亚胺薄膜 — (Kapton®)	290,000	0.04

导电材料作不能用做覆盖层，因为它对电场图形产生干扰。 鉴于此，在覆盖层中不使用包含金属颗粒的油漆。

通常的做法是使用粘合剂将覆盖层粘合到 CapSense PCB 上。 3M™ 生产的透明丙烯酸粘合剂薄膜 200MP 符合 CapSense 的应用。 而且，这一特殊粘合剂有别于纸质胶带卷。（3M™ 产品型号：467 MP & 468 MP ）

5.2 ESD 保护

细致的系统设计中包含着健壮的 ESD 容差。 考虑到终端产品的接触放电情况，特别是用户界面上的接触放电，18-kV 的放电事件可能不会对 CapSense 控制器造成任何损坏。

CapSense 控制器引脚能够经受 2-kV 的直接放电事件。 大多数情况下，覆盖层材料能为控制器引脚提供足够的 ESD 保护。 表 5-1 列举了保护 CapSense 传感器避免 12-kV 放电所需的不同覆盖层材料的厚度，符合 IEC 61000-4-2 标准。 如果覆盖层材料无法提供足够的保护，则按照以下步骤采取 ESD 对策： 预防，重新定向，钳制。

5.2.1 预防

确保触摸表面所有路径上的击穿电压高于可能的高压接触放电。 同时，在设计系统时，请保证 CapSense 控制器和可能的 ESD 源之间离开合适的距离。 如果无法保持合适的距离，则请在 ESD 源和 CapSense 控制器之间放置一个高击穿电压材料保护层。 一层 5 mil 厚的 Kapton®胶带可以经受住 18 kV。

5.2.2 重新定向

若产品包装严密，则可能无法预防放电事件。 在这种情况下，您可以通过控制放电位置来保护 CapSense 控制器。 一种标准的做法是在连接接地线的电路板周围放置一个保护环。 按照 PCB 布局指南中的建议，在按钮或滑条传感器周围提供一个开口接地层可以将 ESD 事件重新定向，使之远离传感器和 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。 在此情况下，包含串联电阻或特殊用途的 ESD 静电保护器件可能适用。

推荐的串联电阻值为 560 Ω。

一个更有效地方法是为脆弱的走线安装专用 ESD 保护器件。 CapSense 的 ESD 保护器件必须是低电容的。 表 5-2 列出推荐使用的 CapSense 控制器保护器件。

表 5-2. 推荐的 CapSense 低电容 ESD 保护器件

ESD 保护器件		输入电容	漏电流	接触放电最大限制	空气放电最大限制
制造商	部件编号				
Littlefuse	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) 注意事项

5.3.1 辐射干扰

辐射电能会影响系统测量，并可能影响处理器内核工作。 干扰会通过 CapSense 传感器走线和任何其他数字或者模拟输入，在 PCB 水平影响 PSoC® 芯片。 有关最大限度减小 RF 干扰影响的布局指南包括：

- **接地层**： 为 PCB 提供一个接地层。
- **串联电阻**： 在 CapSense 控制器引脚 10mm 的范围内安装串联电阻。
  - 推荐的 CapSense 输入线串联电阻值为 560 Ω。
  - 推荐的通信线（如 I^2C 和 SPI）串联电阻值为 330 Ω。
- **走线长度**： 如有可能，请最大限度减少走线长度。
- **电流循环区域**： 最大限度减小电流的返回路径。 应提供规划接地而非实体填充，其与传感器和走线的距离须在 1cm 以内，以减少寄生电容的影响。
- **射频源位置**： 带有噪声源（如 LCD 反相器和开关电源 (SMPS) ）的分区系统将其与 CapSense 输入分离。 屏蔽电源是预防干扰的另一个常用技巧。

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。 提高预分频器的值会减小开关时钟的频率。

5.3.3 传导抗扰与辐射

传导噪声指通过与其他系统互连进入一个系统的噪声。 这些互连的媒介包括电线和通信线。 由于 CapSense 控制器是低功耗器件，必须避免传导辐射。 下列指南将有助于降低传导辐射和抗干扰；

- 根据数据手册，推荐使用去耦电容。
- 在系统电源上添加一个双向滤波器， 这对传导辐射和抗扰性很有效。 π 型滤波器能够防止电源噪声影响敏感部件，但同时也会阻隔该敏感部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，则请最大限度减少线缆长度并考虑使用带屏蔽层的线缆。
- 请在电源或者通信线周围放置一个铁氧体磁珠，以过滤高频噪声。

5.4 软件滤波

使用软件滤波也是在上层系统中处理噪声的技术之一。表 5-3 列出了 CapSense 中有用的过滤器类型。

表 5-3. CapSense 过滤器表

类型	说明	应用
一般	带有同等加权系数的有限脉冲响应滤波器（无反	电源周期性噪声

	馈）	
IIR	带有阶跃响应（类似 RC 过滤器）的无限脉冲响应滤波器（有反馈）	高频率白噪声（1/f 噪声）
中值	从缓存中计算中值输入值的非线性滤波器（缓存大小为 N 的）	电机和开关电源的噪音毛刺
时序抖动	基于先前输入来限制电流输入的非线性滤波器	厚覆盖层 (SNR < 5:1) 中的噪声，特别适用于滑条质心数据
基于事件	非线性滤波器，该滤波器引起对从传感器数据中获取的模式的预定义响应	通常用于非触摸事件中，用来阻碍 CapSense 数据传输。
基于规则	非线性滤波器，该滤波器引起对从传感器数据中获取的模式的预定义响应	通常用于触摸表面的常规操作中，用于特殊情形的响应（如意外选中了多个按钮）。

表 5-4 详细阐述了不同软件滤波器对 RAM 和闪存大小的要求。 每款滤波器所需的闪存大小，取决于编译器的性能。此处列出了 ImageCraft 编译器和 ImageCraft Pro 编译器所需的闪存大小。

表 5-4. RAM 和闪存需求

过滤器类型	过滤器阶	RAM（字节/每个传感器）	闪存（字节）ImageCraft 编译器	闪存（字节）ImageCraft Pro 编译器
一般	2–8	6	675	665
IIR	1	2	429	412
	2	6	767	622
中值	3	6	516	450
	5	10	516	450
原始计数的时序抖动滤波器	不可用	2	277	250
滑条质心的时序抖动滤波器	不可用	2	131	109

5.5 功耗

5.5.1 系统设计建议

很多设计的重要目标之一是最大限度减少功耗。 有多种减少 CapSense 电容式触摸感应系统功耗的方法。

- 设置通用 I/O 驱动为低耗能模式。
- 关闭高功耗模块。
- 优化低功耗状态下的 CPU 速度。
- 在较低的 V_{DD} 下操作。

除这些建议外，应用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。 可将此器件可以置于睡眠状态，以停止 CPU 和其他主要组件的工作。 睡眠状态下消耗的电流比活动状态下少很多。

一段时间内，此器件平均消耗的电流可以使用以下公式计算。

$$I_{AVE} = \frac{(I_{Act} \times t_{Act}) + (I_{Slp} \times t_{Slp})}{T}$$ 公式 9

此器件的平均功耗可以使用以下公式计算：

$$P_{AVE} = V_{DD} \times I_{AVE}$$ 公式 10

5.5.3 响应时间与功耗

如公式 11 所述，通过降低 I_{AVE} 或 V_{DD} 可以减小功耗。增加睡眠时间可能减少 I_{AVE}。 如果睡眠时间值过高，将会使 CapSense 按钮响应时间变慢。 因此，睡眠时间须基于系统需求进行设定。

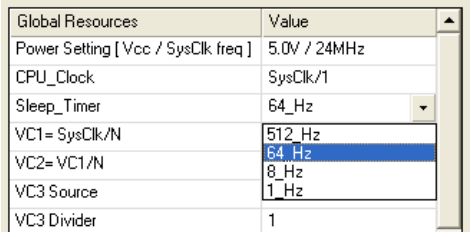
在任何应用中，若功耗和响应时间都是需要考虑的重要参数，则可使用下述优化方法，即同时采用连续扫描和睡眠扫描模式。 该方法中，器件的大部分时间分配给睡眠扫描。如先前章节所述，器件周期性地扫描传感器并进入睡眠，从而产生较低的功耗。 用户通过触摸传感器操作系统时，此器件切换至连续扫描模式，传感器连续扫描，并防止进入睡眠模式，从而大大缩短响应时间。 此器件会在一定的超时周期内保持连续扫描模式。 如果在超时周期内，用户未对传感器进行操作，则器件会切换回睡眠扫描模式。

5.5.4 测量平均功耗

以下内容介绍使用睡眠扫描方法时，如何确定平均功耗：

1. 建立一个项目，使其扫描所有的传感器且防止切换至睡眠模式（连续扫描模式）。 扫描传感器之前，在代码中包含一个引脚切换特征。 切换输出引脚的状态并进行时间标记，该时间标记可用示波器跟踪。
2. 下载项目到 CapSense 器件，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，并分配给 I_{SLP}。
4. 在示波器中监控切换输出引脚，并测量两次切换之间的时间段。 这样便获得了活动时间。 将此数值分配给 t_{ACT}。
5. 此项目中应用睡眠扫描模式。 如图 5-1 所示：在“全局资源”窗口中选择睡眠定时器频率，将会设置睡眠扫描循环的周期，T。
6. 睡眠扫描循环时间段减去活动时间就是睡眠时间。 $T_{SLP} = T - t_{ACT}$.
7. 公式 10 用于计算平均电流。
8. 公式 11 用于计算平均功耗。

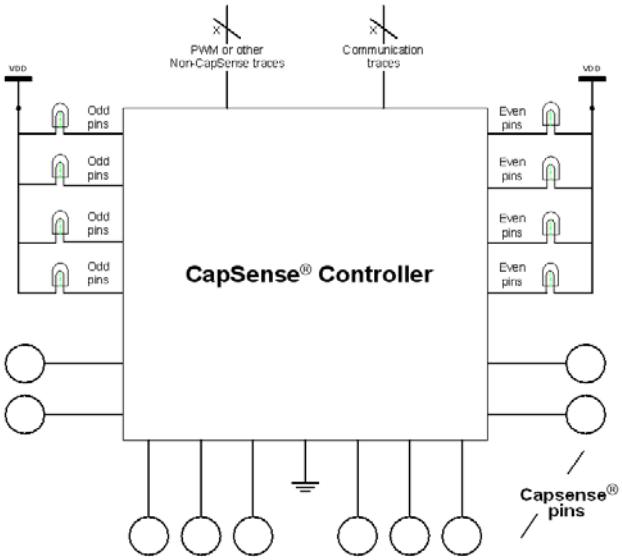
图 5-1. “全局资源”窗口



5.6 引脚分配

一个减少 CapSense 传感器走线、通信和非 CapSense 走线之间交互的有效方法是通过端口分配，将三者分离。 图 5-2 显示的是针对 32 引脚 QFN 封装进行分离的基础版本。 由于各个功能是相互分离的，须确定 CapSense 控制器方向，使得通信、LED 和感应轨迹之间没有交叉。

图 5-2. 推荐的通信、CapSense 和 LED 端口隔离



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 控制器提供了可以使用高灌电流和源电流的端口引脚。 使用端口引脚中的高灌电流或源电流时，为了最大限度减少噪声，推荐使用距器件接地引脚最近的端口。

5.7 PCB 布局指南

CapSense 入门中提供了详细的 PCB 布局指南。

6. 低功耗设计注意事项

功耗是微控制器设计的一个重要方面。 当前多种降低 CapSense 控制器所耗平均电流的可用技术中，睡眠模式应用最广。 无需执行任何功能时，CapSense 控制器将开启睡眠模式，正如手机闲置一段时间后，背景灯将变暗。 这是为了降低设备平均电流消耗，也是所有电池应用的必备特征。 CapSense 控制器向 CPU_SCR0 （位 3） 寄存器中的睡眠位写入“1”，进入睡眠模式。 实现方式是调用 M8C_Sleep 宏。 处于睡眠模式时，CPU 停止工作，内部主振荡器 (IMO) 为禁用，带隙电压参考断电，闪存模式也处于禁用状态。 此时，仅有供应电压监视器与 32 kHz 内部振荡器电路仍处于工作状态。 除睡眠模式外，其他节能技术包括：

- 禁用 CapSense (PSoC) 模拟模块参考
- 禁用 CT 与 SC 模块
- 禁用 CapSense (PSoC) 模拟缓冲区
- 将 HI-Z 模拟设为驱动模式

睡眠模式会给设计带来不利影响。 如果未谨慎使用，则可能产生无法预知的操作问题。 如果需要将 PSoC 从睡眠模式中唤醒，则必须以正确方式进行，且用户必须注意设备处于睡眠模式，可进行额外处理。

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 HI-Z 模拟设为驱动模式

CapSense 控制器驱动模式的状态会影响功耗。 您仅可以在不会对系统造成不利影响的引脚上改变驱动模式。 模式改变必须按一定顺序进行，避免造成线路故障。 顺序由当前的引脚驱动模式与端口数据寄存器状态决定。 依据 CapSense 控制器驱动模式结构，引脚必须处于电阻上拉或电阻下拉驱动模式，才能转换为 HI-Z 或强驱动模式。 当前驱动模式与先前引脚上的值相反。 因此，如果之前引脚电阻拉高，那么当前的驱动模式应为电阻下拉。 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。

进入睡眠模式前，驱动模式需要在软件中手动设置。 驱动模式由 PRTxDM0，PRTxDM1 与 PRTxDM2 三个寄存器控制。 寄存器每一位分配给一个引脚。 因此，改变一个引脚的驱动模式需要三次写入寄存器。 然而，由于三次相同的写入改变了整个端口，因此操作很便捷。 模拟 HI-Z 正确的 pit pattern 是 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 HI-Z 模拟。

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

6.1.2 实例应用

下面的代码是一个典型的 28 引脚部件的睡眠准备顺序。 在这个顺序，中断被禁用，模拟电路被关闭，所有驱动模式均设置为 Analog HI-Z，中断被重新启用。

```
void PSoC_Sleep(void){
   M8C_DisableGInt;
   ARF_CR &= 0xf8; // analog 冻结 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 睡眠模式问题

CapSense 控制器可通过复位或中断退出睡眠模式。 CapSense 控制器共有三种复位方式： 外部复位、看门狗定时器复位及加电复位。 任意一种复位方式均可使 CapSense 控制器退出睡眠模式。 复位解除后，CapSense 控制器将从 *Boot.asm* 开始执行代码。 唤醒 CapSense 控制器的可用中断有： 睡眠定时器、低电压监控器、通用 I/O、模拟列及异步。 通过中断唤醒 CapSense 控制器，或在睡眠状态中连接数字通信，都将产生睡眠模式问题。 以下章节将对这些注意事项进行详细说明。

6.1.4 待处理中断

如果某个中断处于待处理状态，已启用，并预定在 CPU_SCR0 寄存器中对睡眠位进行写入后发生，系统将不会进入睡眠状态。 指令继续执行，但 CapSense 控制器不会设置睡眠位。 中断指令可有效地使 CapSense 控制器忽略睡眠指令。 为避免这一问题，在睡眠模式准备期间，中断需全局禁用，并且在写入睡眠位之前重新启用。

6.1.5 全局中断启用

要将 CapSense 控制器从中断中唤醒，无需启用全局中断启用寄存器 (CPU_F)。 用中断从睡眠状态中唤醒控制器的唯一要求是，在 INT_MSKx 寄存器中使用正确的中断掩码，如下例所示。 如果全局中断被禁用，唤醒 CapSense 控制器的 ISR （中断服务子程序） 并未执行，但 CapSense 控制器仍可退出睡眠模式。

在这种情况下，必须手动清除待处理中断，或启用全局中断，以便进行对 ISR 的处理。 中断在 INT_CLRx 寄存器内清除。

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

6.2 后唤醒执行序列

如果 CapSense 控制器是通过复位唤醒，则从启动代码开始执行。 如果 CapSense 控制器是通过中断服务子程序唤醒，则需执行的第一条指令是睡眠指令的下一条。 原因是睡眠指令的下一条指令在 CapSense 控制器完全进入睡眠状态前已被预取。 因此，如果全局中断被禁用，在睡眠开始前中断的指令将继续执行。

6.2.1 PLL 模式启用

如果 PLL 模式启用，则进入睡眠前，CPU 频率必须降至最小值 3 MHz。 原因是 CapSense 控制器退出睡眠重新启用后，PLL 尝试重新锁定，但常常过冲。 此外，为确保执行无误，在唤醒后与常规 CPU 操作开始前，应等待 10 ms。 这意味着，要使用睡眠模式和 PLL，该软件必须可在 3 MHz 的频率下执行。 OSC_CR0 寄存器只需一次简单写入，便可降低 CPU 速度。 然而，该寄存器仅设一个 SYSCLK 的分频器，这就表示 CPU 速度在不同的 SYSCLK 的部件系列中有所不同。 通常， SYSCLK 为 24 MHz。

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

6.2.2 执行全局中断启用

避免在写入睡眠位的指令边界上启用中断。 如果睡眠指令从中断 (reti) 指令返回执行，则会绕过所有进入睡眠的固件准备工作。 为避免这一情况，睡眠准备开始前，中断暂时被禁用，进入睡眠前，中断重新启用。 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。

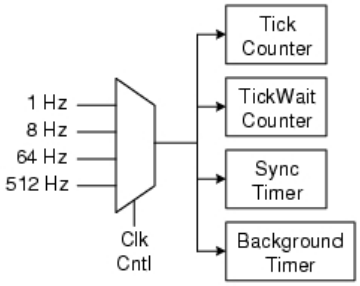
6.2.3 睡眠模式下的I^2C 从器件

在睡眠模式下使用在 I^2C 从器件会有几个问题。 由于睡眠期间 IMO 与 CPU 不工作，CapSense 控制器内部将停止处理工作。 这就产生有关 I^2C 地址的问题。 当 I^2C START 条件发送至某一特定地址时，CapSense 控制器无法处理地址，因此以 NAK 作为回复。 通常的解决方法是，在时钟或 I^2C 总线的数据线上生成下降沿中断。 主控便可发送虚拟 START 条件，从而唤醒 CapSense 控制器。 唤醒后 CapSense 无法立即处理 I^2C 地址，期间存在一定的延迟。因此，主控需延迟 200 µs 才可进行下一传输操作，或继续发送直至接收到 ACK。 该解决方案将产生另一问题：任意 I^2C 下降沿流量都将唤醒 CapSense 控制器，这会增加总活动时间并增大睡眠电流。 另一种解决方案是，使用第三个通用 I/O 引脚，唤醒 CapSense 控制器，并在适当的延时后发送初始 START 条件。

6.2.4 睡眠定时器

CapSense 控制器提供一个睡眠定时器和一个睡眠定时器用户模块。 二者在 CapSense 控制器睡眠时使用，并执行类似的功能。 实际的睡眠定时器使用内部低速振荡器，此振荡器从不关闭。 定时器可选间隔有 1 Hz、8 Hz、64 Hz 和 512 Hz，任意一种均可生成中断。 周期性地唤醒 CapSense 控制器进行一定的处理或检查活动状态是很有用的。 例如，周期性地唤醒 CapSense 进行传感器扫描。 睡眠定时器用户模块利用睡眠定时器提供额外功能。 该功能包括：后台计数器形成周期性中断，程序循环延迟，计数器可设置，以及循环调节器控制循环时间。 该功能可由一张简洁的框图体现，如 图 6-1.

图 6-1. 睡眠定时器用户模块框图



7. 资源

7.1 网站

如需获取本章讨论的所有参考资料，请访问赛普拉斯 CapSense 控制器网站。

如需查看 CapSense CY8C20xx6A/H/AS 系列器件丰富的技术资源，请查看 CY8C20xx6A/H 网页。

7.2 数据手册

CapSense CY8C20XX6A/H/AS 系列器件的数据手册可在 www.cypress.com 上获取。

- CY8C20x36A, CY8C20x46A, CY8C20x66A, CY8C20x96A, CY8C20x46AS, 及 CY8C20x66AS
- CY8C20336H, CY8C20446H

7.3 技术参考手册

赛普拉斯撰写以下技术参考手册，使用户能快速简便地获取 CapSense 控制器功能的相关信息，包括顶级架构图、寄存器图及时序图。

- 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 CapSense 控制器套件

通用 CapSense 控制器套件特含预先定义的控制电路和插入硬件插件，使原型设计与调试简单轻松。 程序设计和 I^2C-to-USB 桥接器硬件也包含在内，以便进行调校和数据采集。

- CY3280-20xx6 通用 CapSense 控制器

7.4.2 通用 CapSense 模块板

7.4.2.1 简单按钮模块板

CY3280-BSM 简单按钮模块由 10 个 CapSense 按钮和 10 个 LED 组成。 该模板与任意 CY3280 通用 CapSense 控制器板连接

7.4.2.2 矩阵按钮模块板

CY3280-BMM 矩阵按钮模块由 8 个 LED 和 8 个 CapSense 传感器组成，以 4×4 的矩阵形式排列，形成 16 个物理按钮。 该模板与任意 CY3280 通用 CapSense 控制器板连接。

7.4.2.3 线性滑条模块板

CY3280-SLM 线性滑条模块由 5 个 CapSense 按钮、1 个线性滑条（带 10 个传感器）以及 5 个 LED 组成。 该模板与任意 CY3280 通用 CapSense 控制器板连接。

7.4.2.4 辐射状滑条模块板

CY3280-SRM 辐射状滑条模块由 4 个 CapSense 按钮、1 个辐射状滑条（带 10 个传感器）以及 4 个 LED 组成。 该模板与任意 CY3280 通用 CapSense 控制器板连接

7.4.2.5 通用 CapSense 原型设计模块

CY3280-BBM 通用 CapSense 原型设计模块提供获取信号的路径，这些信号路由至所连接的控制器板上的 44 引脚连接器。 原型设计模块板与通用 CapSense 控制器板协同工作，实现额外的功能，该功能不属于其他单用途的通用 CapSense 模块板

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）提供互联。 现已推出以下转接板。

- 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 样版文件

赛普拉斯提供原理图与 PCB 板文件，可作为参考，帮助您快速完成 PCB 设计。

- CY8C20466A 上带有 I^2C 数据头的按钮设计
- CY8C20466A 上带有 I^2C 数据头的按钮与滑条设计

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

通过 I2C 数据头 J2，I2C 与 CY8C20466A-24LQXI 进行通讯。

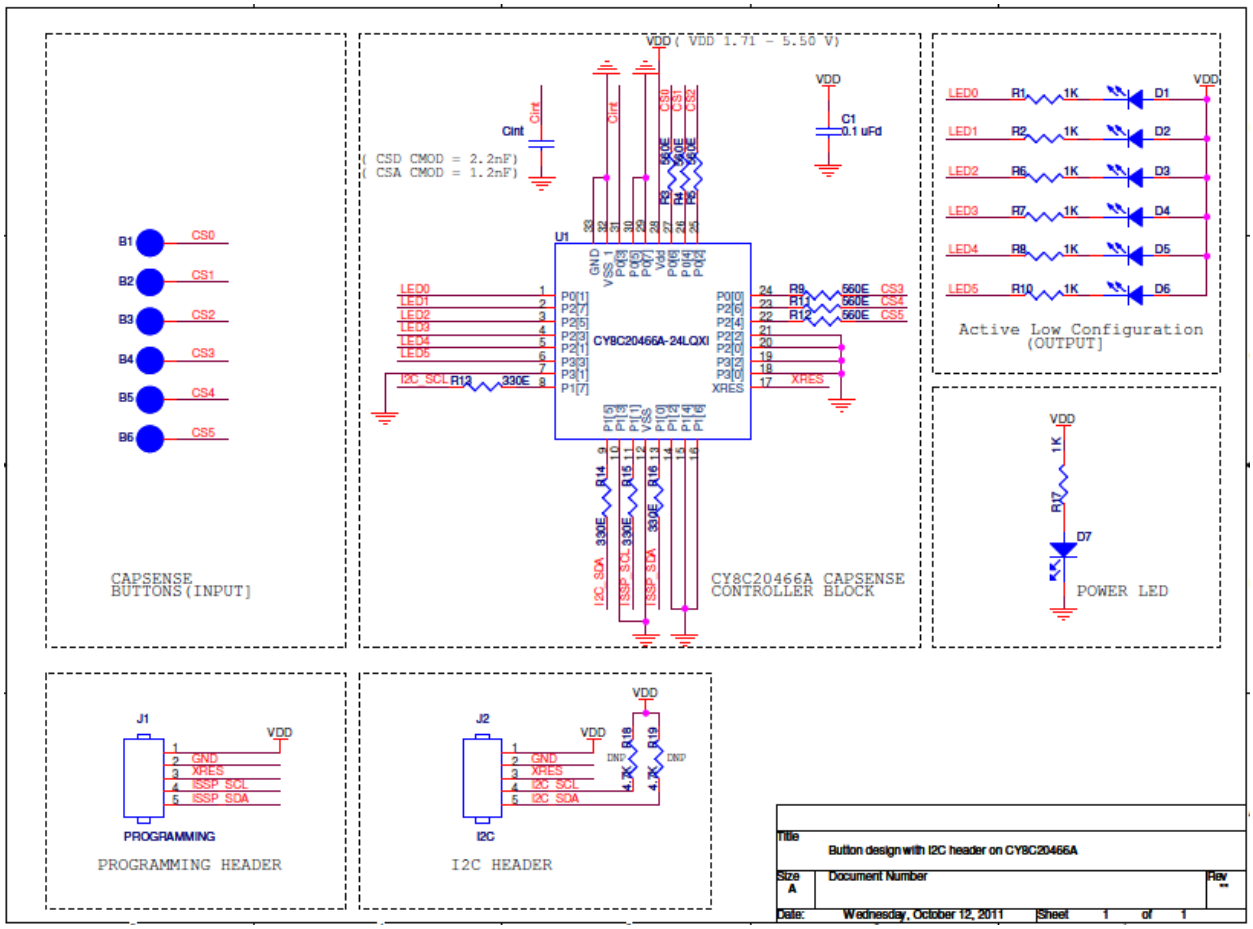
图 7-1 与通过 I2C 数据头 J2，I2C 与 CY8C20466A-24LQXI 进行通讯。

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]，驱动 D1、D2、D3、D4、D5 及 D6 等 LED。
- 通过编程数据头 J1 对 CY8C20466A-24LQXI 进行编程。
- 通过 I^2C 数据头 J2，I^2C 与 CY8C20466A-24LQXI 进行通讯。

图 7-1. CY8C20466A - 板原理图上带有 I^2C 数据头的按钮设计

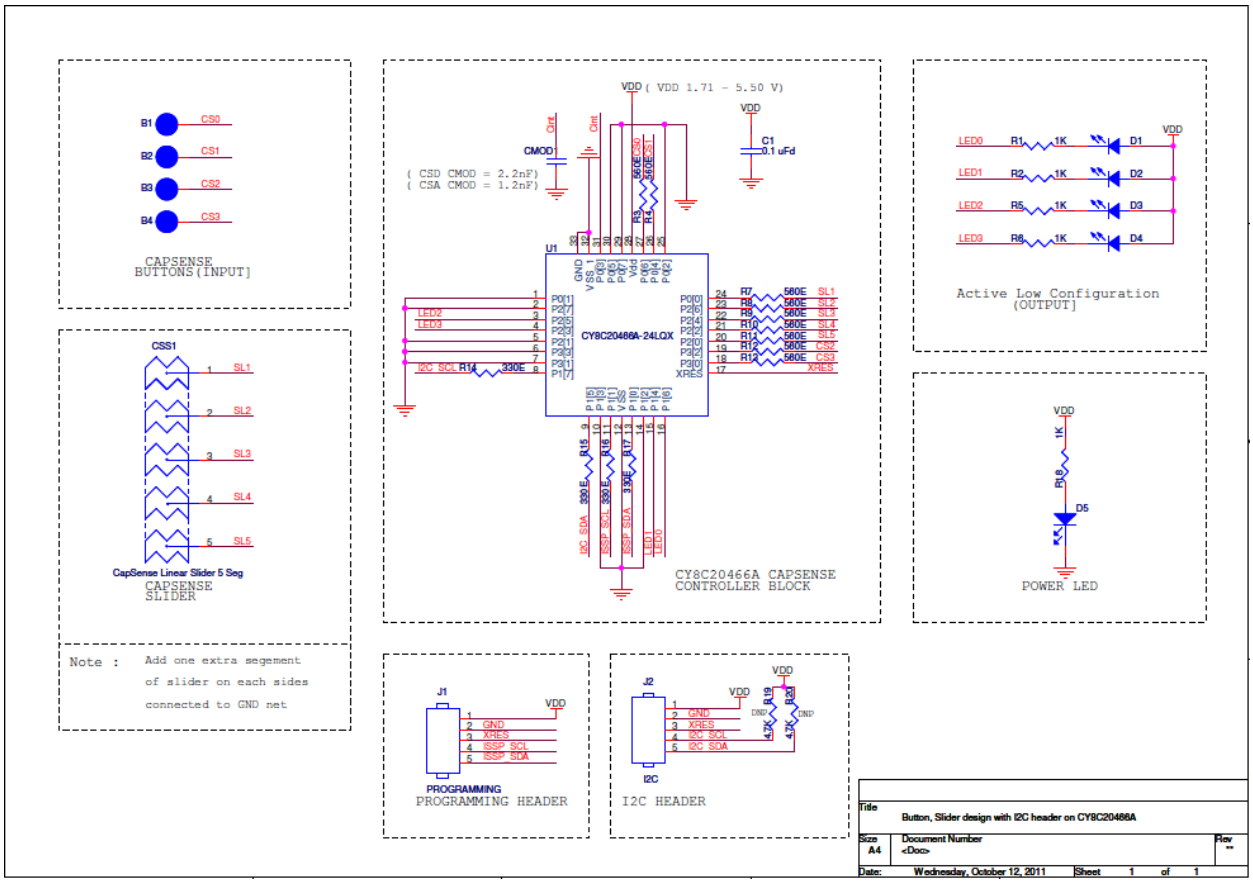


以下原理图用于支持：

- 4 个 CapSense 传感器。 这些传感器分配给 CY8C20466A-24LQXI 器件的引脚 P0[6]、P0[4]、P3[2] 及 P3[0]。

- 分为 5 段的线性滑条。 这些滑条段分配给 CY8C20466A-24LQXI 器件的引脚 P0[0]、P2[6]、P2[4]、P2[2] 及 P2[0] 。
- 4 个 GPO 连接到 CY8C20466A-24LQXI 的引脚 P1[6]、P1[4]、P2[5] 及 P2[7]，驱动 D1、D2、D3 及 D4 等 LED。
- 通过编程数据头 J1 对 CY8C20466A-24LQXIby 进行编程。
- 通过 I^2C 数据头 J2，I^2C 与 CY8C20466A-24LQXI 进行通讯。

图 7-2. CY8C20466A 上带有 I^2C 数据头的按钮与滑条设计



7.6 PSoC Programmer

PSoC Programmer 是进行 PSoC 器件编程的一款灵活的集成编程应用软件。 可结合 PSoC Designer 和 PSoC Creator 使用，为 PSoC 器件进行任意的程序设计。

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。 PSoC Programmer 硬件层在 COM 指南文档与代码示例中有详尽说明。代码编写语言包括： C#、C、Perl 和 Python。

7.7 CapSense 数据查看工具

在 CapSense 设计中，出于调校及调试目的，您经常需要监控相关的 CapSense 数据（原始计数、基准线值、差值等等）。

应用笔记 AN2397 - CapSense 数据查看工具可帮助您找到并使用正确的工具，查看和记录 CapSense 数据。

7.8 PSoC Designer

赛普拉斯提供专有的集成设计环境，PSoC Designer。 使用 PSoC Designer，您可配置模拟和数字模块，开发固件，以及调校设计。 应用程序在拖放式设计环境中开发，使用完整设定的模拟和数字功能库，包括 CapSense。 PSoC Designer 带有内建 C 语言编译器与嵌入式编程器。 现已推出用于复杂设计的专业版编译器。

7.9 代码示例

赛普拉斯提供大量代码示例，使您轻松快速进行设计。

- 在 CY8C20xx6A 上使用 SmartSense 的 CapSense 按钮
- 在 CY8C20xx6A 上使用 SmartSense 的 CapSense 按钮与滑条
- 在 CY8C20xx6A 上使用 SmartSense 的 CapSense 矩阵按钮
- 在 CY8C20xx6A 上使用 CapSense CSD 降低功耗
- 在 CY8C20xx6A 上使用 CapSense CSA 降低功耗
- CY8C20xx6A 上带有 EzI2Cs 从器件的 CSD 软件过滤器
- CY8C20xx6A 上带有 LED 背光调节的 CSA
- CY8C20xx6A 上带 TX8SW 的 CSD

7.10 设计支持

赛普拉斯提供多种设计支持渠道，保证为您提供成功的 CapSense 解决方案。

- 知识库文章 - 您可按产品类别浏览技术文章，或对 CapSense 的不同话题进行搜索。
- CapSense 应用笔记 - 您可查看本文档中出现过的各种应用笔记。
- 白皮书 – 了解电容式触摸接口的高级主题。
- 赛普拉斯开发者社区 – 与赛普拉斯技术社区沟通和交流信息。
- CapSense 产品选择器指南 – 参阅赛普拉斯 CapSense 产品系列的完整产品供应。
- 视频资料库 - 您可通过教学视频快速掌握设计技术
- 质量和可靠性 - 赛普拉斯竭诚为您服务，让您满意。 您可以在赛普拉斯的质量网站上找到可靠性和产品合格报告。
- 技术支持 - 您可在线获取高水平的技术支持。

“本文档版本号为Rev**，译自英文版 001-65973 Rev*D。