AN76000 - CY8CMBR2110

CapSense®设计指南

文档 编号：001-89147 版本*C

赛普拉斯半导体公司
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

© 赛普拉斯半导体公司，2012-2018 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

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

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

目录

1. 简介 6

1.1 摘要 6
1.2 赛普拉斯的 CapSense 文档系统 6
1.3 CY8CMBR2110 CapSense Express 器件特性 8
1.4 文档规范 9
1.5 缩略语 10

2. CAPSENSE 技术 11

2.1 CapSense 基本原理 11
2.2 电容式感应方法 12
2.2.1 CapSense Sigma-Delta （CSD） 12
2.3 SmartSense 自动调试 14
2.3.1 过程差异 14
2.3.2 缩短了设计周期时间 14

3. CAPSENSE 原理图设计 15

3.1 CapSense 控制器引脚 15
3.1.1 CapSense 按键（CSx） 15
3.1.2 通用输出（GPOx） 15
3.1.3 调制器电容（CMOD） 16
3.1.4 蜂鸣器信号输出（BuzzerOut0、BuzzerOut1） 16
3.1.5 主机控制 GPO （HostControlGPO0、HostControlGPO1） 18
3.1.6 提醒/睡眠线 18
3.2 CapSense 控制器配置 20
3.2.1 按键自动复位（ARST） 20
3.2.2 抗噪能力 20
3.2.3 自动阈值 20
3.2.4 切换 ON/OFF 21
3.2.5 侧翼传感器抑制（FSS） 21
3.2.6 LED 点亮保持时间 22
3.2.7 LED 效果参数 22

3.2.8 锁存状态读取 27
3.2.9 模拟电压支持 28
3.2.10 灵敏度的控制 29
3.2.11 去抖动的控制 29
3.2.12 系统诊断 30
3.2.13 按键扫描速率 31
3.2.14 I²C 通信 32
3.3 设计工具箱 33
3.3.1 通用布局指导 33
3.3.2 布局估计 34
3.3.3 C_P、功耗以及响应时间计算器 35
3.3.4 设计验证 37
3.4 配置 CY8CMBR2110 38
3.4.1 EZ-Click 定制工具 40
3.4.2 使用主机处理器配置器件 41
3.4.3 第三方的编程器 47
3.5 CY8CMBR2110 复位 47
3.5.1 硬件复位 47
3.5.2 软件复位 47

4. 电气和机械设计中的注意事项 48

4.1 覆盖层选择 48
4.2 ESD 保护 49
4.2.1 预防 49
4.2.2 重定向 49
4.2.3 钳制 49
4.3 电磁兼容性（EMC）注意事项 50
4.3.1 辐射干扰 50
4.3.2 抗传导干扰和辐射 50
4.4 PCB 布局指导方针 50

5. 低功耗设计中的注意事项 51

5.1 系统设计建议 51
5.2 计算平均功耗 51
5.2.1 按键扫描速率 （T_R） 52
5.2.2 扫描时间（T_S） 53
5.2.3 NO TOUCH（无触摸）状态下的平均电流（I_{AVE_NT}） 54
5.2.4 TOUCH（触摸）状态下的平均电流（I_{AVE_T}） 54
5.2.5 工作时间的比例（P） 54
5.2.6 平均使用电流（I_{AVE_U}） 55

5.2.7 平均电流（I_{AVE}）.....55
5.2.8 平均功耗（P_{AVE}）.....55
5.2.9 示例计算.....55
5.3 睡眠模式.....56
5.3.1 低功耗睡眠模式.....56
5.3.2 深度睡眠模式.....57

6. 资源.....58

6.1 网站.....58
6.2 数据手册.....58
6.3 设计工具箱.....58
6.4 EZ-CLICK™定制工具.....58
6.5 设计支持.....58

7. 附录.....59

7.1 原理图示例.....59
7.1.1 原理图 1：十个按键和十个 GPO.....59
7.1.2 原理图 2：八个按键和模拟电压输出.....61
7.2 用于 CY8CMBR2110 配置的 API.....63
7.2.1 高层 API.....63
7.2.2 低层 API.....82

术语表.....83

修订记录.....88

1. 简介

1.1 摘要

本文档介绍了如何使用赛普拉斯的 CapSense® Express™ CY8CMBR2110 器件的电容感应功能。本指南包括以下主题：

- CY8CMBR2110 特性
- CapSense 的操作原理
- CY8CMBR2110 器件的配置选项
- 使用 CY8CMBR2110 的设计工具箱
- CY8CMBR2110 的系统的电气和机械设计的注意事项
- CY8CMBR2110 的低功耗设计注意事项
- 附加资源和支持予系统中设计 CapSense

1.2 赛普拉斯的 CapSense 文档系统

图 1-1 和表 1-1 汇总了 CapSense 文档体系。这些资源有助于用户快速访问需要成功完成 CapSense 产品设计所需的信息。图 1-1 显示了典型的电容式感应产品设计的周期流程；本指南所提供的内容在下表中用绿色标出。表 1-1 中提供的链接用于支持图 1-1 中每个编号任务。

图 1-1. 典型的 CapSense 产品设计流程

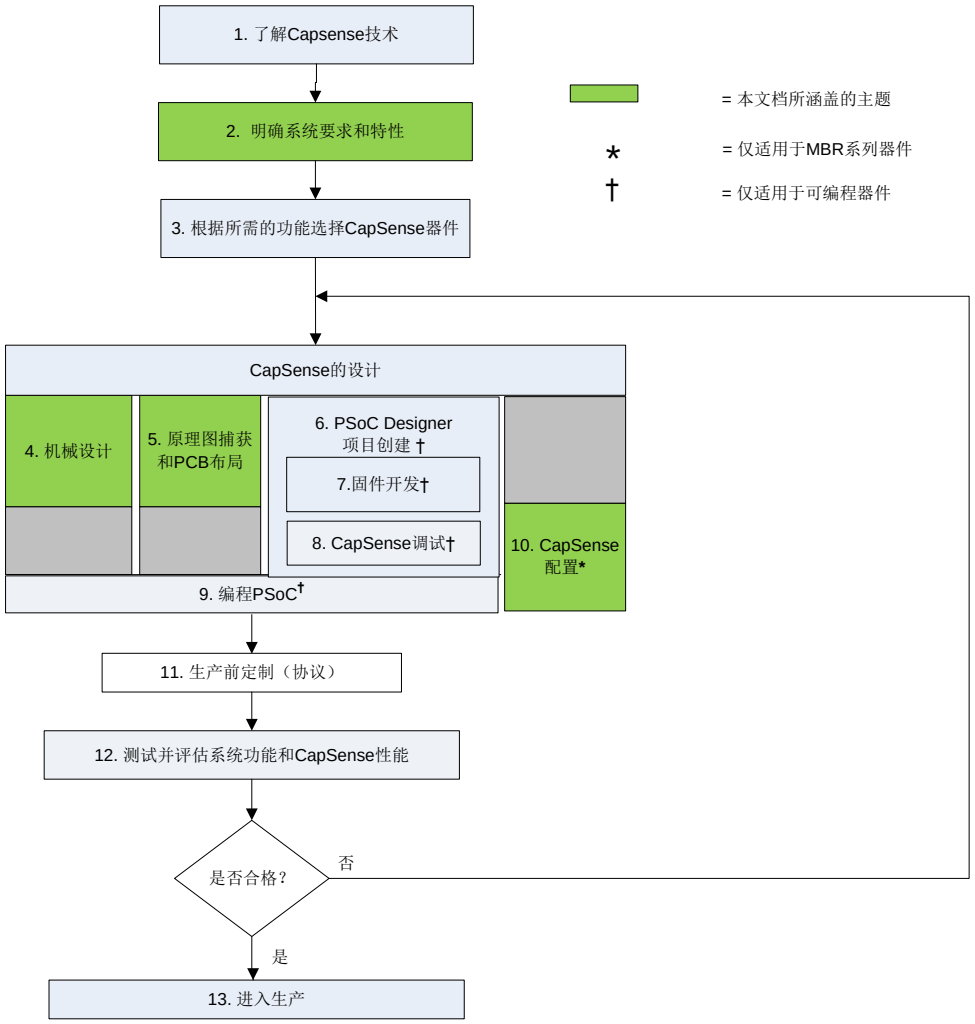


表 1-1. 图 1-1 中标注编号的设计任务，对应的赛普拉斯支持文档

图 1-1 中设计任务编号	赛普拉斯 CapSense 支持文档
1	Capsense 入门手册
2	CY8CMBR2110 器件数据手册
3	Capsense 入门手册
4	本文档
5	本文档
6	不适用于 CY8CMBR2110
7	不适用于 CY8CMBR2110
8	不适用于 CY8CMBR2110
9	不适用于 CY8CMBR2110
10	本文档

1.3 CY8CMBR2110 CapSense Express 器件特性

通过赛普拉斯的低功耗 CapSense 控制器可以很轻松地将电容式感应技术应用到您的用户界面中。该器件的功能包括：

- 寄存器可配置的 CapSense 控制器
  - 不要求固件或器件编程
  - 通过 I^2C 协议可以配置十个按键解决方案
  - 十个通用输出（GPO）
  - GPO 链接到 CapSense 按键
  - GPO 支持直接驱动 LED
- SmartSense™自动调试
  - CapSense 算法可以连续补偿由系统、生产过程和环境的变化引起的影响
  - 可动态设置各个 CapSense 参数
  - 不需要手动调试系统
  - 宽广的寄生电容（C_P）范围（5–40 pF）
- 高级特性
  - 侧翼传感器抑制（FSS）
    - 区分紧密排列按键时的信号
  - 用户可配置的 LED 效果
    - 当系统上电时
    - 当有按键触摸时
    - 按键释放后的 LED ON 时间
    - 待机模式下的 LED 亮度
  - 蜂鸣器信号输出
  - 模拟电压输出
    - 使用外部电阻桥接器
  - 通过中断连线提示主机，用于表示 CapSense 按键的状态更改
  - CapSense 通过 I^2C 接口进行传输数据的性能
    - 简化生产线测试和系统调试的流程
- 抗噪能力
  - 抵抗外部辐射和传导噪声的能力更加优越
  - 低噪声辐射
- 系统诊断
  - 按键短接
  - 错误的调制电容值（C_{MOD}）
  - 超出范围的寄生电容（C_P）

- EZ-Click™ 定制器工具
  - 简单直观的配置
  - 可以动态配置所有功能
  - 可以保存配置以备将来使用
- I²C 接口
  - 无需时钟延展
  - 支持高达 100 kHz 的速度
- 宽广的工作电压范围
  - 1.71 - 5.5 V
  - 适用于稳压和非稳压的电池应用
- 低功耗
  - 每个按键平均消耗的电流为 23 µA[1]
  - 深度睡眠电流：100 µA
- 工业级温度范围：–40 °C 到+85 °C
- 32 引脚 QFN 封装（5 mm x 5 mm x 0.6 mm）

1.4 文档规范

规范	用法
Courier New 字体	用于显示文件位置、用户输入的文本和源代码： C:\ ...cd\icc\
斜体	用于显示文件名称和参考文档： 阅读 *PSoC Designer 的用户指南*文档中的 *sourcefile.hex* 文件。
[方括号、粗体]	用于显示程序中的键盘指令： [Enter]或[Ctrl] [C]
File（文件）> Open（打开）	表示菜单路径： File > Open > New Project（新建项目）
粗体字	显示程序中的各条指令、菜单路径和图标名称： 请点击 **File** 图标，然后点击 **Open**。
Times New Roman 字体	用于显示公式：2 + 2 = 4
灰色框中的文本	说明警告或产品的独特功能。

[1] 使用四个按键，每小时有 180 次的按键触摸，平均按键触摸时间为 1000 毫秒，禁用蜂鸣器，禁用按键触摸 LED 效果，10 pF < （所有按键的 C_P） < 20 pF，按键扫描速率为 541 毫秒，最优化功耗，抗噪级别为“正常”，CSx 的灵敏度为“中等”。

1.5 缩略语

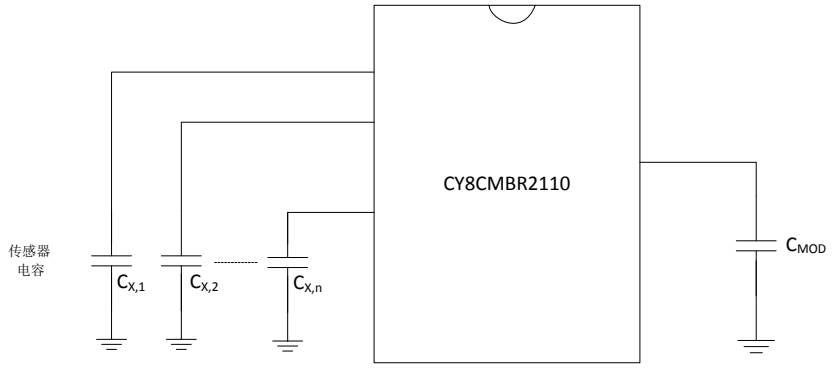
缩略语	说明
AC	交流电
ARST	自动复位
C_F	手指电容
C_P	寄生电容
CS	CapSense
CSD	CapSense Sigma Delta
EMC	电磁兼容性
ESD	静电释放
FSS	侧翼传感器抑制
GPO	通用输出
MSB	最高有效位
LCD	液晶显示器
LED	发光二极管
LSB	最低有效位
PCB	印刷电路板
POR	加电复位
POST	上电自测试
RF	射频
SNR	信噪比
SMPS	切换模式电源供应

2. CapSense 技术

2.1 CapSense 基本原理

CapSense 是一种触摸式感应技术，它的工作方式是在 CapSense 控制器上测量每个传感器输入引脚的电容值。每个传感器引脚的所有容值可以模拟为等效的一个总容值，从 CX,1 到 CX,n 如图 2-1 所示。CY8CMBR2110 器件的内部电路将各个 CX 的大小转换为数字量，然后保存以供后期处理。CapSense 控制器的内部电路中使用了调制电容器（CMOD）。电容式感应方法一节中将会更加详细的论述 CMOD。

图 2-1. CapSense 在 CY8CMBR2110 器件中的实现



根据要求使用走线、过孔或同时使用二者将每个传感器输入引脚连接至传感器板。覆盖每个传感器板的非导电性外覆层构成了该产品的触摸式界面。当手指与外覆层接触时，人体组织的导电性会产生一个与传感器板并行的接地导电层， 如图 2-2 所示。此操作构成一个平行板电容器，其容值可通过下面公式计算得到：

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

其中：

C_F = 手指与传感器外覆层接触时所产生的电容值

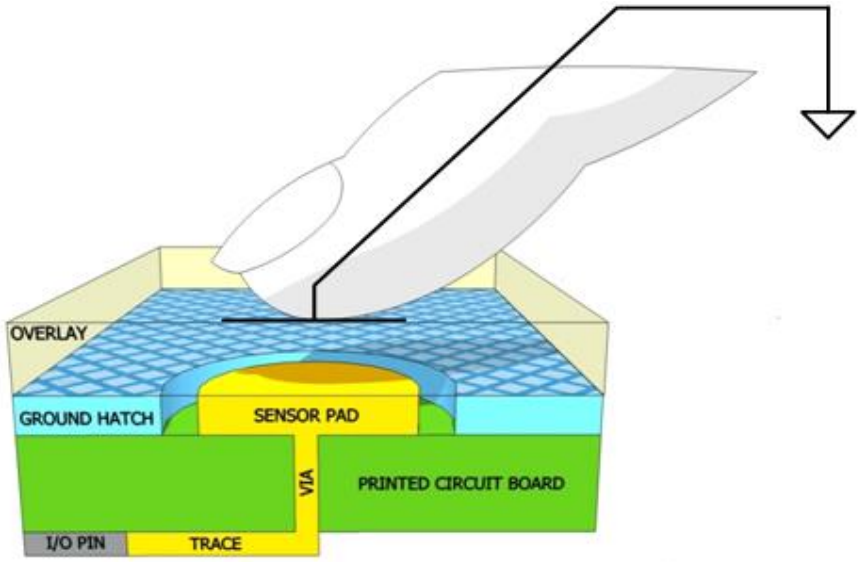
ε_0 = 空气介电常数

ε_r = 外覆层的绝缘常数（相对介电常数）

A = 手指与传感器板外覆层的接触面积

D = 外覆层的厚度

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，手指与外覆层的接触也会在其自身与附近其他导体之间产生边缘电场。通常情况下，这些边缘电场很微弱，可以忽略不计。

即使手指未触摸外覆层，传感器输入引脚也会有一些寄生电容（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，其中后者是在传感器板、走线和过孔以及系统中的其他导体（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，请使用下面公式：

$$C_X = C_P$$ 公式 2

当手指在传感器板上时，C_X 等于 C_P 和 C_F 之和：

$$C_X = C_P + C_F$$ 公式 3

通常，C_P 比 C_F 大一个数量级以上。C_P 的范围值通常为 10 至 20 pF，但在极端情况下它的值会高达 40 pF。C_F 的取值范围通常为 0.1 至 0.4 pF。

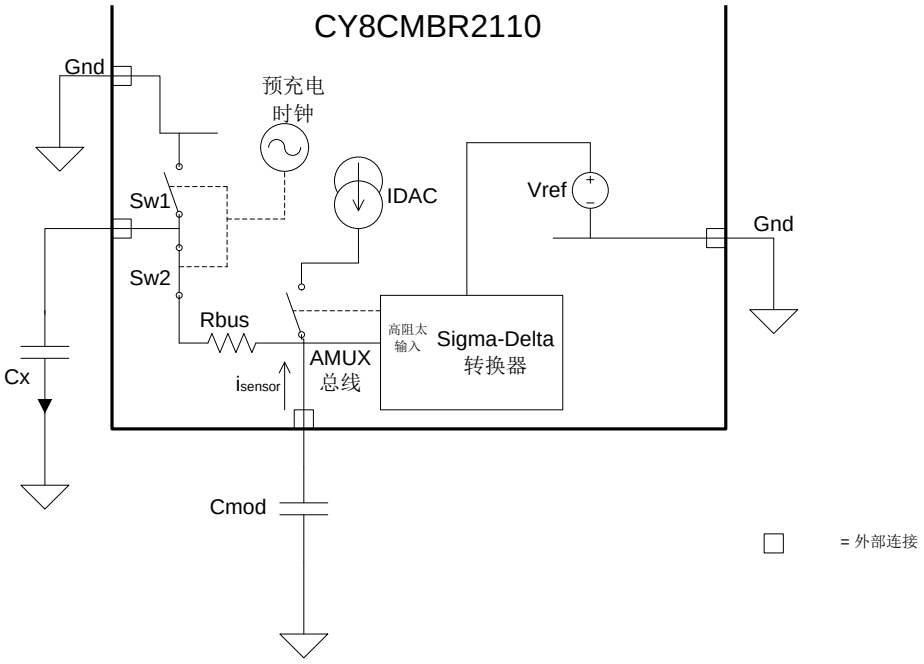
2.2 电容式感应方法

CY8CMBR2110 器件支持使用配有 SmartSense Auto-Tuning 的 CapSense Sigma Delta（CSD）将传感器电容（C_X）转换为数字值。下面各部分将介绍 CSD 方法。

2.2.1 CapSense Sigma-Delta（CSD）

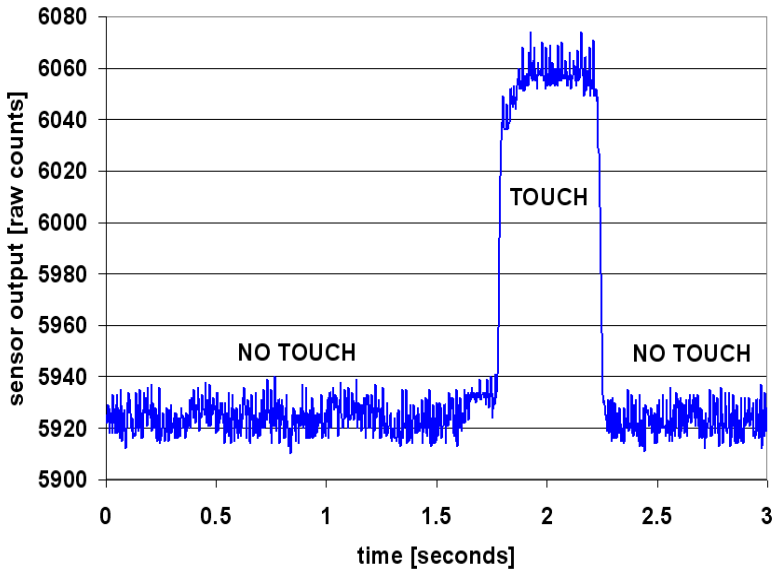
CY8CMBR2110 器件中的 CSD 方法将 C_X 整合至开关电容电路中，如图 2-3 所示。通过非重叠开关 Sw1 和 Sw2，选择性地将 C_X 连接到 GND 和 AMUX 总线。Sw1 和 Sw2 由预充电时钟驱动，以通过模拟复用器（AMUX）总线充电，电流是（i_{sensor}）。i_{sensor} 的大小与 C_X 的大小成正比。Sigma-Delta 转换器对 AMUX 总线电压进行采样，并产生一个调制的比特流控制的恒流源，IDAC。IDAC 对 AMUX 充电，这样，AMUX 总线的平均电压便保持为 Vref。传感器释放 CMOD 中的电荷 i_{sensor}。CMOD 与 Rbus 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



为了维持 AMUX 总线的电压为 Vref，Sigma-Delta 转换器通过控制比特流占空比将 IDAC 与 i_{sensor} 进行匹配。Sigma-Delta 转换器在传感器扫描期间内存储位流，累积结果为数字输出，原始计数，该值与 C_X 成正比。此原始计数由高级算法进行计算，以便求得传感器的状态。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 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。通过比较基于稳定状态下原始计数的变化量和预定阈值，高级算法能够确定传感器是在 ON（触摸）还是 OFF（未触摸）状态。欲了解更多有关原始计数、手指阀值以及信噪比（SNR）的信息，请参考《CapSense 入门手册》。

图 2-4. 手指触摸期间的 CSD 原始计数



2.3 SmartSense 自动调试

调试触摸感应用户界面对于确保系统正常运行和用户的良好体验非常重要。但不幸的是，调试非常费时，因为它是一个重复的过程。在典型的开发周期中，会在初始设计阶段,系统集成过程中，大批量量产前时去调试这个界面。 SmartSense 自动调试性能用于简化用户界面开发周期。CapSense 算法可以连续补偿由系统、生产过程和环境的变化引起的影响。它易于使用，并且在原型和制造阶段通过消除手动调试缩短了设计周期。SmartSense 自动调试功能在每个 CapSense 按键加电时对其进行自动调试，并维持运行时的最佳按键性能。SmartSense 自动调试适用解决 PCB、外覆层的制造误差，并且自动消除噪声源（如 LCD 反相器、交流电路和开关模式电源）噪声。

2.3.1 过程差异

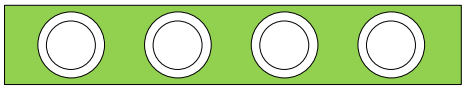
CY8CMBR2110 器件的 SmartSense 自动调试功能适用的 C_P 取值范围为 5~40 pF。各个按键的灵敏度参数会根据自身的特性自动设置。无论各个按键的 C_P 变化如何，每个按键均保持了一致的响应，因此该参数可以在大批量生产中提高产量。C_P 可能由于 PCB 布局和走线的长度、PCB 制造流程各不相同或多源供应链中不同的 PCB 供应商，而有所差异。 按键的灵敏度取决于 C_P 的大小；C_P 值越高，灵敏度就越低，进而导致手指触摸的信号幅度被降低。C_P 值的变化，会使某个按键过于灵敏、灵敏度不够或者不能正常工作。当发生这种情况时，您必须重新调整系统，有时候需要重新认证用户界面子系统。SmartSense 自动调试可以解决这些问题。

通过 SmartSense 自动调试使平台化设计成为可能。例如，研究一下在笔记本电脑上的电容式接触感应多媒体按键。 由于笔记本电脑大小和键盘布局的不同，因此在相同设计但不同机型的平台上，CapSense 按键的寄生电容值也会不一样。在本例中，按键间隔在宽屏笔记本电脑要比在标屏笔记本电脑的大。因此，宽屏机型上的每个按键与 CapSense 控制器间的走线相对较长，这样会使 C_P 的值增大。对于所有笔记本机型而言，虽然这些按键的功能都是相同的，但是，必须针对每种机型进行调试按键。使用建议的最佳实践（参见《CapSense 入门》中的 PCB 布局），SmartSense Auto-Tuning 可以帮助您实现平台化设计。

图 2-5. 21 英寸笔记本电脑的多媒体按键设计



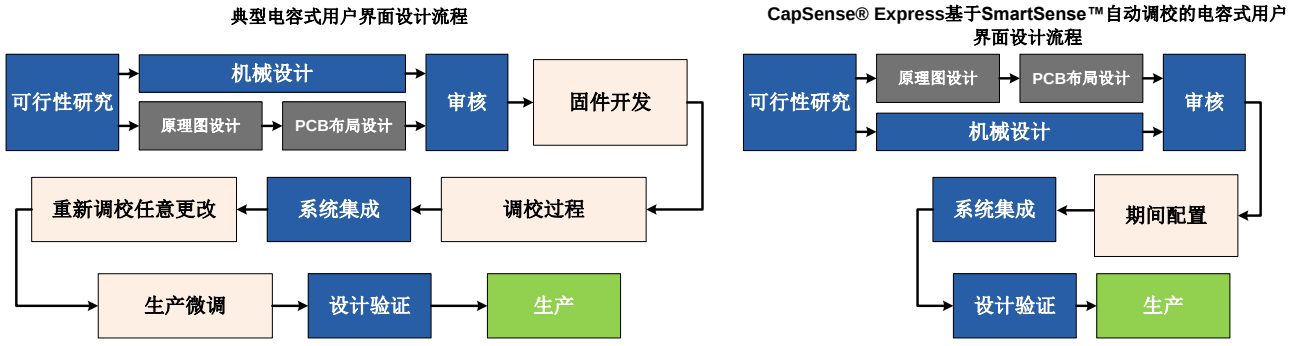
图 2-6. 15 英寸笔记本电脑的多媒体按键设计（该机型与 21 英寸机型有相同的功能和按键大小）



2.3.2 缩短了设计周期时间

当您设计电容按键界面时，最耗时的任务是固件开发、布局和按键调整。对于典型的触摸感应式控制器，当设计被移植到不同型号时，或者机器的 PCB 尺寸或按键 PCB 布局发生变化时，均须重新调试按键。带有 SmartSense 自动调试功能的设计可以适应这些挑战，因为它不需要固件开发、手动调试或重新调试。另外，SmartSense 自动调试功能还可以缩短典型的设计周期。图 2-7 比较了典型的触摸感应式控制器和基于 SmartSense 自动调试设计的设计周期。

图 2-7. 典型电容式用户界面的设计周期对照

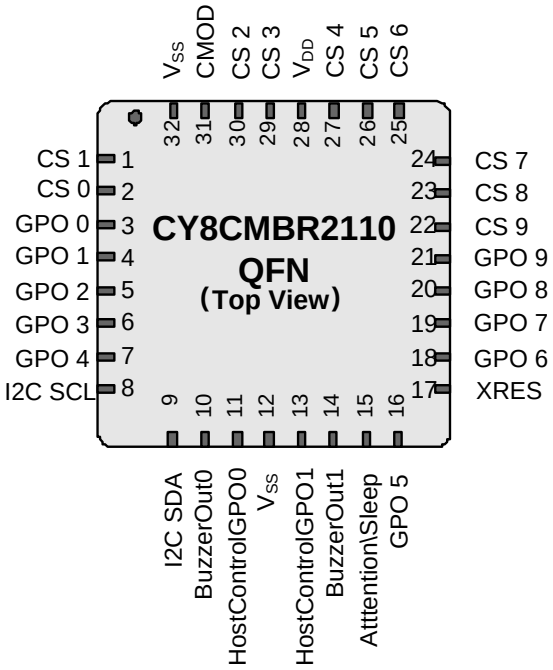


3. CapSense 原理图设计

赛普拉斯的 CY8CMBR2110 器件是使用硬件和 EZ-Click 定制器工具通过 I²C 接口来配置的。本节介绍了 CapSense 控制器引脚和寄存器以及如何配置它们。

3.1 CapSense 控制器引脚

图 3-1. CY8CMBR2110 引脚框图



3.1.1 CapSense 按键（CSx）

CY8CMBR2110 控制器具有十个电容式感应输入，即 CS0 至 CS9。每个电容式按键都需要连接到一个电容式感应输入。您必须将所有未使用的 CapSense （CSx）输入引脚接地。

3.1.2 通用输出（GPOx）

CY8CMBR2110 控制器上共有十个低电平有效输出，即 GPO0 至 GPO9。每个输出均由它的相应电容式感应输入（即 CSx）驱动。 您可以使用 GPO 直接驱动 LED 或替换机械开关。各个 GPO 处于强驱动[2]模式。所有未使用的 GPO 引脚都要悬空。

[2] 引脚在强驱动模式时，如果输出为 HIGH 时，该引脚被上拉为 V_{DD}；如果输出为 LOW 时，该引脚被下拉为 Ground。

3.1.3 调制器电容（CMOD）

将一个大小为 2.2 nF（±10%）的电容器连接到 CMOD 引脚。

3.1.4 蜂鸣器信号输出（BuzzerOut0、BuzzerOut1）

触摸CapSense按键时，通过蜂鸣器信号输出可以发出音频反馈。这样有助于使用音频传感器的设计。用于压电蜂鸣器的蜂鸣器信号输出。

蜂鸣器信号输出是强驱输出。这些输出通常由所有 CSx 按键驱动。如果不使用蜂鸣器，可以将 BuzzerOut0 和 BuzzerOut1 作为主机控制的 GPO 使用。显示的是各种蜂鸣器设置。

蜂鸣器信号输出有两种配置：

1. AC 单引脚蜂鸣器：蜂鸣器连接至器件的 BuzzerOut0 引脚，如图 3-2 所示。在该引脚上，会驱动一个拥有指定的频率和占空比的方波信号。 可以将 BuzzerOut1 引脚悬空，也可以将它作为主机控制的 GPO 使用。
2. AC 双引脚蜂鸣器：蜂鸣器连接至器件的 BuzzerOut0 和 BuzzerOut1 引脚，如图 3-3 所示。在这两个引脚上，会驱动两个拥有指定的频率和占空比的不同相位的方波信号。

图 3-2. AC 单引脚蜂鸣器配置

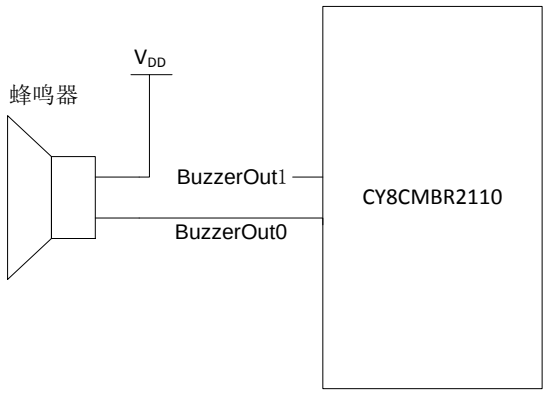
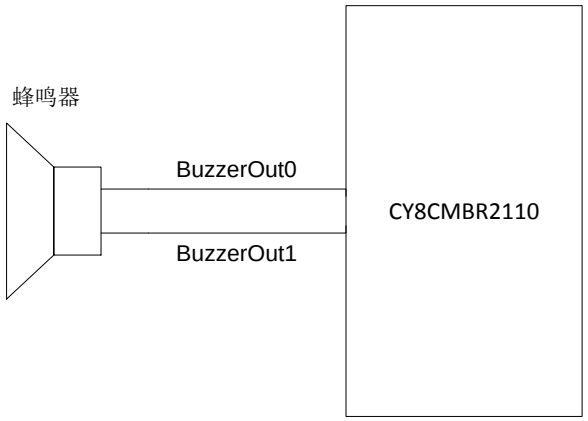


图 3-3. AC 双引脚蜂鸣器配置



蜂鸣器信号的频率是可配置的。表 3-1 列出了不同频率及其相应的输出占空比。

表 3-1. 蜂鸣器信号输出的频率和占空比

蜂鸣器信号输出的频率（kHz）	占空比
1.00	50%
1.14	57.14%
1.33	50%
1.60	60%
2.00	50%
2.67	66.7%
4.00	50%

蜂鸣器开启的时间范围值为：（1 至 127） x 按键扫描的速率常量。更多有关该常量的信息，请参考按键扫描速率部分内容。鸣器信号输出由所配置的时间驱动，并且它并不依赖于按键的触摸时间。蜂鸣器开启时间结束后，即使仍有按键触摸，该输出也会进入空闲状态，如图 3-4 所示。可将空闲状态的蜂鸣器引脚配置为 V_{DD} 或接地（Ground）。如果在蜂鸣器开启的时长结束前发生按键触摸，蜂鸣器信号输出立即重启，如图 3-5 所示。

使能蜂鸣器信号输出时，器件配置模式中的 Buz_Op_Duration 寄存器具有最小值（1）。有关该寄存器的信息，请查看 CY8CMBR2110 数据手册的附录部分。

图 3-4. 蜂鸣器超时

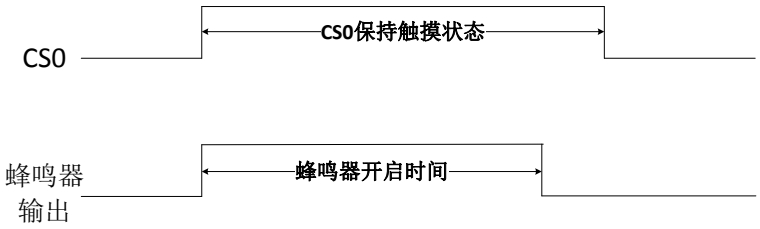
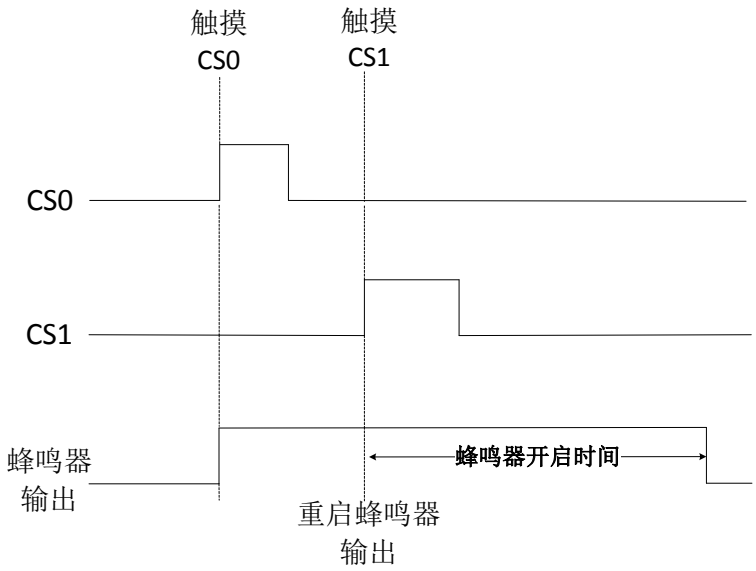


图 3-5. 蜂鸣器信号输出重启



3.1.5 主机控制 GPO（HostControlGPO0、HostControlGPO1）

通过主机可以控制主机控制 GPO 的逻辑状态。 这些输出均为强驱模式。

如果您的设计中没有使用蜂鸣器，可以将 BuzzerOut0 和 BuzzerOut1 引脚作为主机控制 GPO 使用。如果使用了一个 AC 单引脚蜂鸣器，便可以将 BuzzerOut1 引脚作为主机控制 GPO 使用。

主机可以在工作模式、生产线测试模式和调试数据模式下控制这些 GPO 引脚。

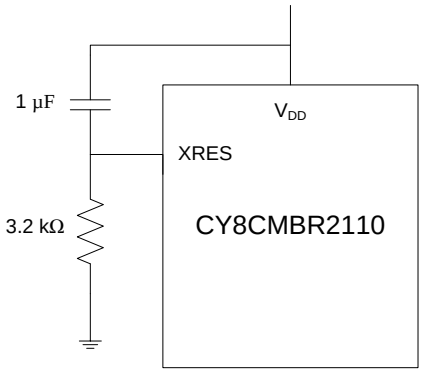
上电时，主机控制 GPO 处于低电平状态，并且复位后必须进行配置。与其它特性的配置不同，这些配置不会被保存在闪存存储器内。

HostControlGPO1 在上电时具有 16 毫秒时长的正脉冲。为了消除这个脉冲，在 XRES 引脚上使用了一个外部 RC 网络（误差为± 5%），如图 3-6 所示。这样，每当上电时，器件将处于 XRES 复位状态。退出 XRES 复位后，再经过 16 毫秒，器件将正常操作。

表 3-2. 蜂鸣器与主机控制的 GPO

蜂鸣器配置	BuzzerOut0 引脚	BuzzerOut1 引脚	可用的主机控制 GPO 的最大数量
无蜂鸣器	悬空/主机控制 GPO3	悬空/主机控制 GPO2	4
AC 单引脚	蜂鸣器引脚 0	悬空/主机控制 GPO2	3
AC 双引脚	蜂鸣器引脚 0	蜂鸣器引脚 1	2

图 3-6. 上电时避免 HostControlGPO1 脉冲的 XRES 引脚配置



3.1.6 提醒/睡眠线

提醒/睡眠是开漏低电平驱动模式的可由主机和器件控制的双向线。通过使用提醒/睡眠线，可以读取器件的 CapSense 数据，另外可以进入或退出低功耗睡眠和深度睡眠模式。

3.1.6.1 读取器件数据

主机想要读取器件上的数据，需要进行下面两个步骤的操作。

1. 主机下拉提醒/睡眠线到低电平。
2. 主机启动与器件进行的 I²C 通信。

提醒/睡眠线被拉高时，器件便处于低功耗睡眠或深度睡眠模式（如果置位了Host_Mode寄存器中的深度睡眠位）。这时，器件可以使用NACK（非应答信号）终止I²C通信。 保持提醒/睡眠线拉高可节能。

为了读取器件数据，主机可以随时拉低提醒/睡眠线。提醒/睡眠线为低电平时，器件可以使用 NACK（非应答信号）终止 I²C 通信，但这种情况很少见。

如果触摸任何CapSense按键，器件将拉低提醒/睡眠线以便中断主机，如图3-7所示。然后，主机可以与器件进行I²C通信，读取CapSense数据。如果同时触摸多个按键，提醒/睡眠线将被拉低一段时间，如图3-8所示。释放按键后，提醒/睡眠线将变为高电平。

主机应该具备产生提醒/睡眠线的下降沿和上升沿触发的中断的能力，这样便可识别按键触摸和按键释放操作。如果上升沿触发中断不可用，提醒/睡眠线拉低后，主机需要连续检查按键的状态。轮询操作需要在按键扫描率的固定时间内完成。

图 3-7. CS0 和 CS1 分别触摸时的提醒/睡眠线的状态

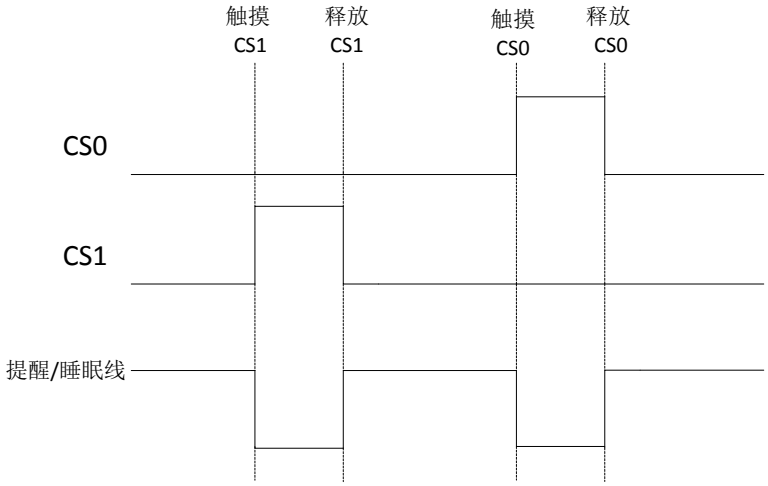
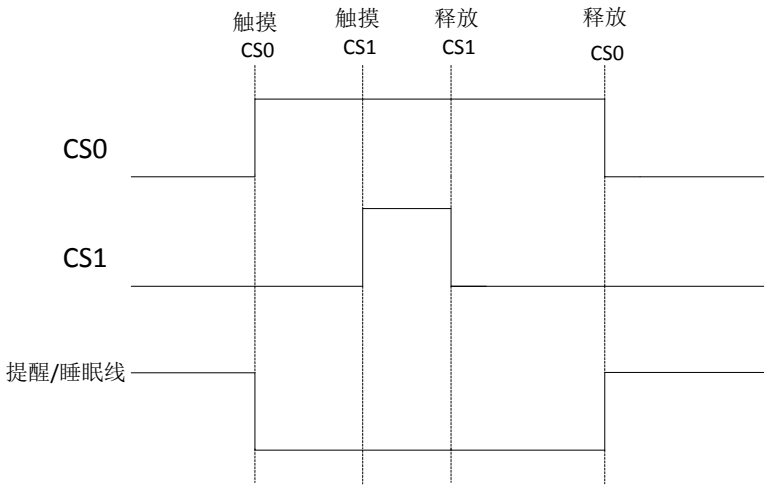


图 3-8. CS0 和 CS1 同时触摸时的提醒/睡眠线的状态



3.1.6.2 睡眠模式

可以配置两个睡眠模式：

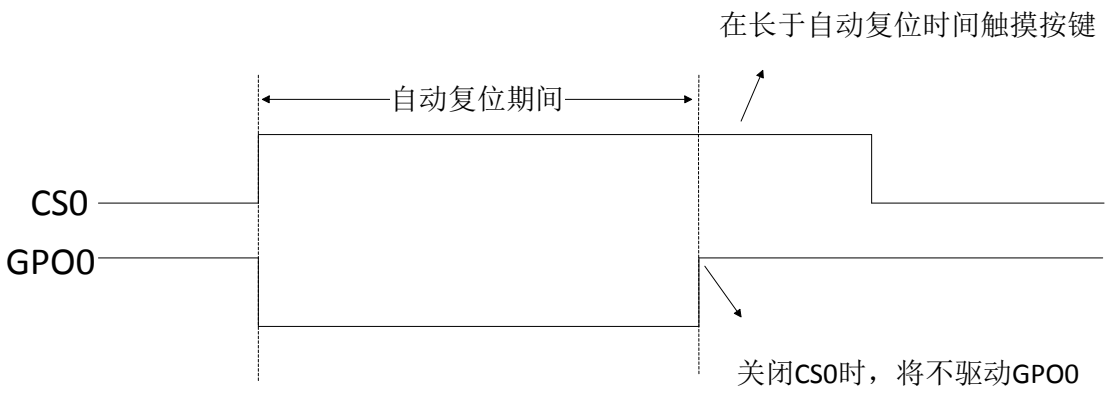
1. 将提醒/睡眠线拉到 V_{DD}，使能低功耗睡眠模式。
2. 将提醒/睡眠线拉到 V_{DD} 并置位工作模式下的 Host_Mode 寄存器中的深度睡眠位，使能深度睡眠模式。

3.2 CapSense 控制器配置

3.2.1 按键自动复位（ARST）

当持续触摸 CSx 时，按键自动复位决定一个按键被确认为 ON 状态的最大时间。ARST 时间过后，按键转为 OFF 状态。如果有一个金属物体靠近某个按键，该功能可阻止该按键被卡住。可以配置 ARST 时间为 5 秒或 20 秒。按键自动复位功能如图 3-9 所示。

图 3-9. 按键自动复位



CSx 因按键自动复位被关闭并释放后，在等于按键扫描速率的时间内请勿触摸按键。

3.2.2 抗噪能力

该设置决定了器件对外部辐射和传导噪声的抵抗能力，如功率放大器的音频频率噪声、无线发射器的射频噪声、ESD 以及电力线浪涌。

在一个没有大噪声的系统中，选择“Normal”抗噪能力。而在一个有大噪声的环境里，选择“High”抗噪能力。当选择抗噪能力为“High”时，功耗和响应时间会增加。如果要求“High”抗噪能力的响应时间与“Normal”抗噪能力的相同，需要降低按键的去抖值。更多有关信息，请参考去抖控制部分。

3.2.3 自动阈值

如 CapSense Sigma-Delta（CSD）部分中所解释的内容，通过比较原始信号的转变和预定阈值（称为手指阈值），可以确定传感器的“ON”或“OFF”状态。手指阈值是可配置的，并且它会影响到器件的其它阈值。更多有关手指阈值的信息，请参考《CapSense 入门手册》。

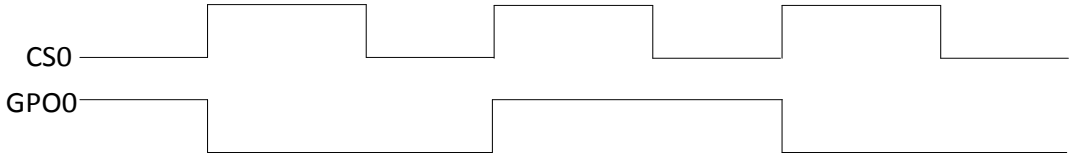
可以为每个按键配置独立的手指阈值，或使用自动阈值功能。自动阈值会根据工作环境中噪声的大小，为每个按键动态设置不同的阈值。对于可变的噪声环境，请使用自动阈值。如果需要手动调整手指阈值，请禁用自动阈值功能并设置手指阈值为所需的水平。

3.2.4 切换 ON/OFF

当启用了切换 ON/OFF 时，GPOx 的状态会在 CSx 的每个上升沿上发生变化。切换 ON/OFF 的配置如图 3-10 所示。

可以为每个 CapSense 按键单独使能切换 ON/OFF 功能。

图 3-10. 切换 ON/OFF 功能



3.2.5 侧翼传感器抑制（FSS）

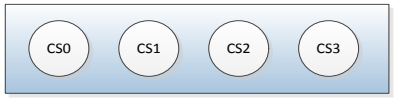
该特性用于区分来自紧密间隔的按键的信号，以避免误触摸。这样能够确保系统仅识别被触摸的第一个按键。在某一时间内，FSS 仅允许有一个 CSx 的状态为 TOUCH。如果手指触摸多个 CSx 按键，则只有第一个感应到“触摸”操作的按键被打开。

FSS 也适用于产生相反效果的相邻按键，如界面上具有两个亮度控制反向（UP 或 DOWN）的按键。

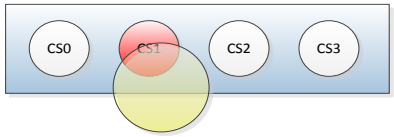
可以为每个按键单独使能 FSS。FSS 配置如图 3-11 和图 3-12 所示。

在实际应用场合中，如洗衣机面板，可以将各个按键分为两组：一组启用了 FSS，另一组禁用了 FSS。这样，您可以将紧密靠近的按键放在设计的一端，并在另一端设计多个单点的触摸功能。

图 3-11. 只有一个按键被按下时的 FSS

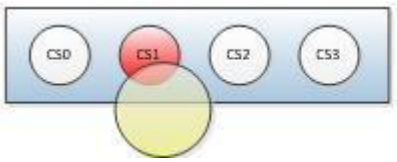


被按下前，没有按键的状态为 ON。

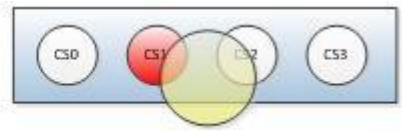


按下时，报告 CS1 的状态为 ON。

图 3-12. 有多个按键被按下时的 FSS（其中，只有第一个按键的状态为 ON）



CS1 被按下，并被报告为 ON

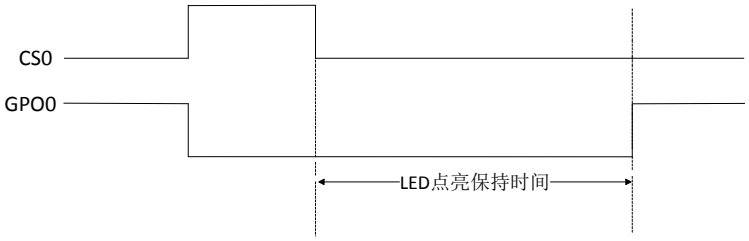


CS2 在 CS1 被按下后也被按下，
只有 CS1 被报告为 ON

3.2.6 LED 点亮保持时间

LED 点亮保持时间指的是释放 CSx 后 GPOx 被拉低的一段时间，如图 3-13 所示。LED 点亮保持时间的取值范围为 0-5100 毫秒，分辨率为 20 毫秒。

图 3-13. LED 点亮保持时间

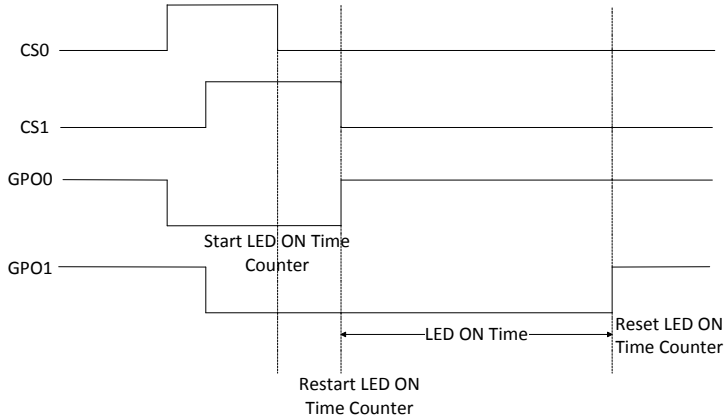


LED 点亮保持时间因器件而异。在-40 °C 到+85 °C 的条件下，精度为±10%。

如果为 CSx 触发了一个按键自动复位（ARST），则 LED 点亮保持时间不能应用于 GPOx。如果切换 ON/OFF 被启用，那么 LED 点亮保持时间功能被禁止。

LED 点亮保持时间在一定时期内仅用于一个 GPOx，意思是说每次有 CSx 处于“无触摸”（NO TOUCH）状态时，LED 点亮保持时间的计数器都被复位。说明当多个按键被按下时，LED 点亮保持时间是如何操作的。CS1 复位 LED 点亮保持时间计数器，从而导致 GPO0 提前关闭。

图 3-14. 多个按键的 LED 点亮保持的时序



3.2.7 LED 效果参数

上电 LED 效果和按键触摸 LED 效果使用了下面各参数：

- 低亮度：最小的 LED 亮度
- 低亮度时间：LED 处于低亮度状态的时长
- 上升时间：LED 从低亮度状态转换到高亮度状态的时长。
- 高亮度：最大的 LED 亮度
- 高亮度时间：LED 处于高亮度状态的时长
- 下降时间：LED 从高亮度状态转换到低亮度状态的时长。
- 重复频率：效果重复的次数。

可以分组配置 GPO，以便得到同样的参数。 有以下各组：

- {GPO1, GPO2, GPO3}
- {GPO4, GPO5, GPO6}
- {GPO7, GPO8, GPO9}

可以单独配置 GPO0 的参数。该功能适用于配置 CS0 按键为特殊功能（如电源按键）的设计。

亮度级别的取值范围为 0-100%。时长范围为 0-1600 毫秒。 高亮度的亮度水平要高于低亮度的。

3.2.7.1 上电 LED 效果

如果该功能被使能，系统上电时，所有连接到 GPO 的 LED 显示了初始时间的调光和渐褪效果。您可以配置这些效果和效果时长。这一过程中，所有 CapSense 按键均被禁用。 只在效果完成后，器件才响应任何按键触摸。

在器件上电的初始化时间过后，便可以观察效果。如果抗噪能力为“Normal”，初始化时间小于 350 毫秒；如果抗噪能力为“High”，则该时间小于 1000 毫秒。

上电后，将执行系统诊断，包括上电自测试。如果有任何 CapSense 按键测试失败，那么在相应的 GPO 上将看不到效果。了解更多关于该测试的信息，请参考《系统诊断》中的内容。

在上电 LED 效果过程中，器件会应答（ACK）I²C 通信，但是所有写指令均被忽略。主机只能读取工作模式下的数据。

可以配置上电 LED 效果，以便在所有 GPO 上同时发生或顺序发生，如图 3-15 和图 3-16 所示。

图 3-15. 上电 LED 效果的示例（同时发生）[3]

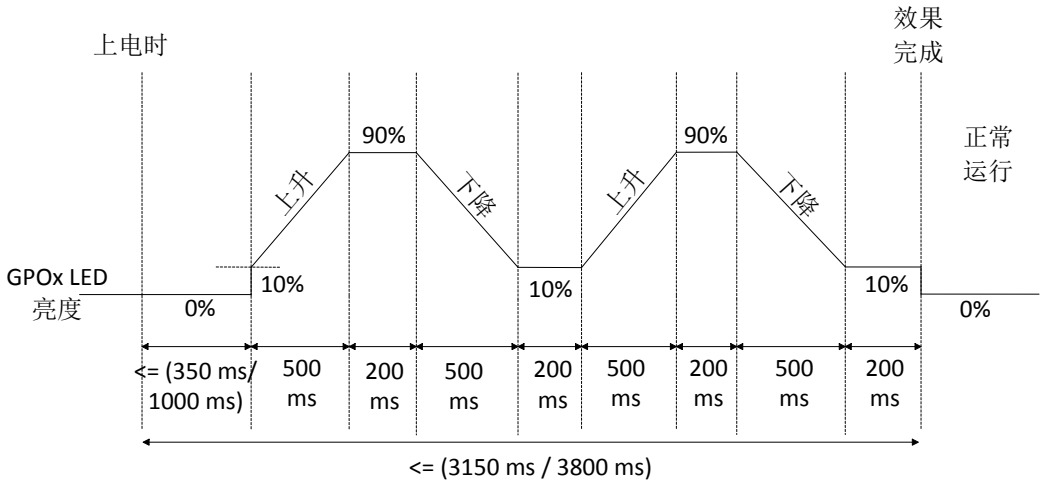
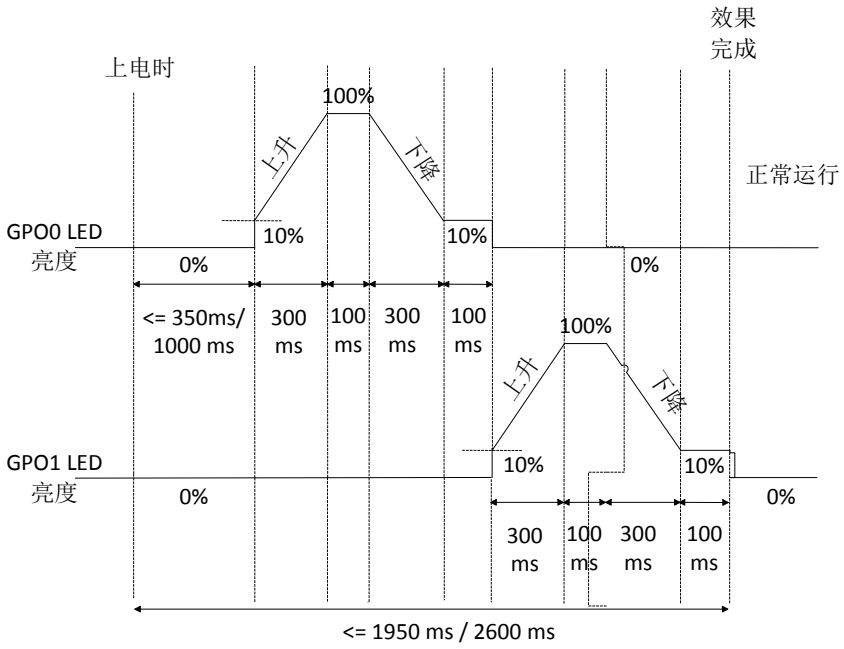


图 3-16. 两按键设计中的上电 LED 效果示例（顺序发生）[4]



[3] 上升时间 = 500 ms；高亮度 = 90%；高亮度时间 = 200 ms；下降时间 = 500 ms；低亮度 = 10%；低亮度时间 = 200 ms；重复频率 = 1

[4] 上升时间 = 300 ms；高亮度 = 100%；高亮度时间 = 100 ms；下降时间 = 300 ms；低亮度 = 10%；低亮度时间 = 100 ms；重复频率 = 0

3.2.7.2 触摸按键时的 LED 效果

该功能有效时，如果某个按键被按下，连接到 GPO 的相应 LED 会显示调光和渐褪效果。您可以配置这些效果和效果的时长。

按键控制 LED 效果可以启用或禁用呼吸效果， 如图 3-17 所示。

呼吸效果被启用：呼吸效果被启用时，如果有一个按键被按下，LED 亮度便立即从待机模式下的 LED 亮度转为低亮度状态。LED 上升到高亮度，并在高亮度时长保持该亮度水平。然后，LED 下降到低亮度，并在低亮度时长内保持这个亮度水平。只要有按键触摸，此效果将一直重复。 释放按键后，呼吸效果周期会继续实现，直到完成为止。呼吸效果重复的周期取决于重复频率。

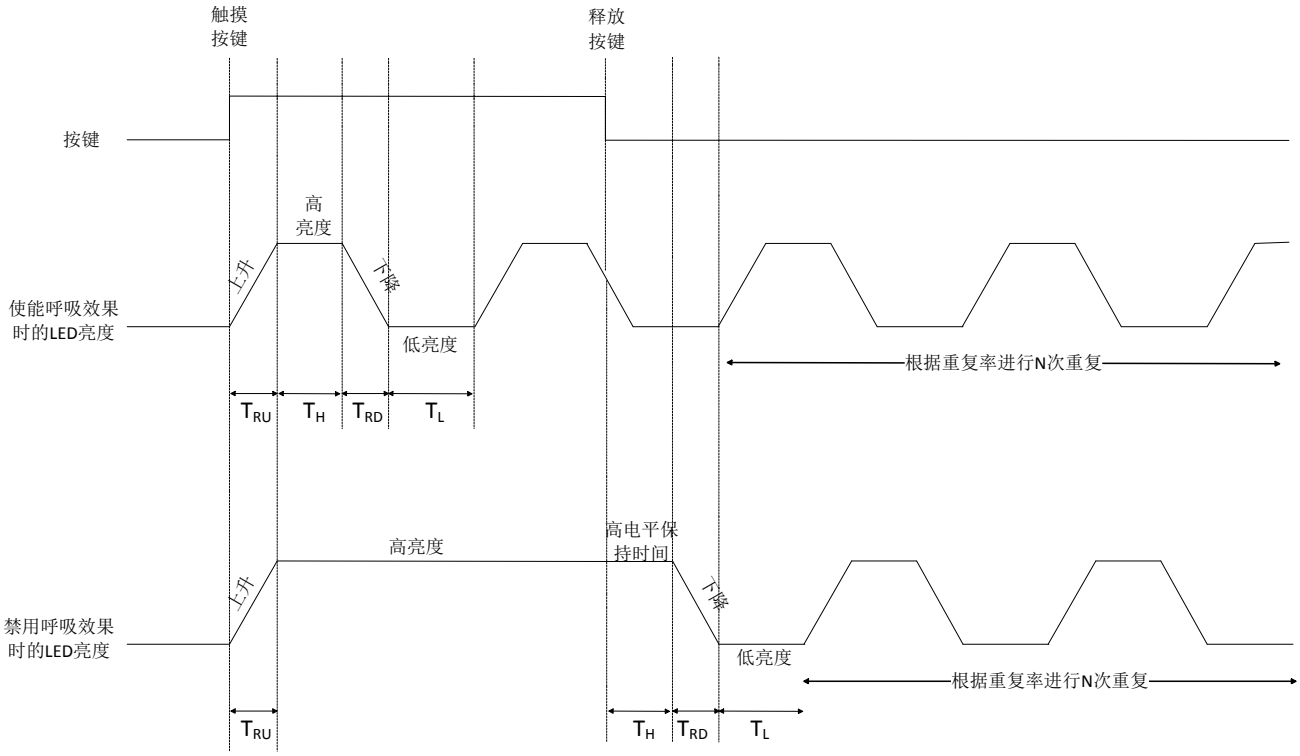
呼吸效果被禁用：呼吸效果被禁用时，如果有一个按键被按下，LED 亮度便立即从待机模式下的 LED 亮度转为低亮度状态。LED 上升到高亮度，并保持这个亮度级，直到释放按键位置。释放按键后，LED 便在高亮度时长内保持高亮度水平，然后下降到低亮度，并在低亮度时长内保持这个亮度水平。此效果重复的周期取决于重复频率。

如果正在 GPOx 上实现按键触摸 LED 效果，同时相应的 CSx 再次被按下，那么此效果周期将在 GPOx 上重新开始。

如果使能了切换 ON/OFF 功能，当有连续的按键触摸时，LED 在待机模式 LED 亮度和高亮度之间进行切换，如图 3-18 所示。

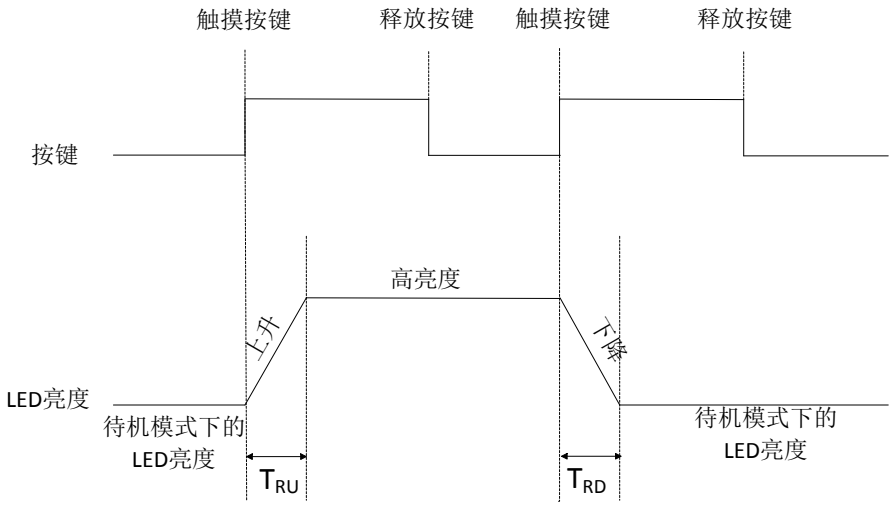
当使能按键触摸 LED 效果时，便自动禁用 LED 点亮保持时间。器件进入深度睡眠模式时，会立即禁用正在执行的按键触摸 LED 效果。

图 3-17. 触摸按键时的 LED 效果[5]



[5] T_{RU} = 上升时长
T_{RD} = 下降时长
T_H = 高亮度
T_L = 低亮度

图 3-18. 使能 ON/OFF 切换时按键触摸 LED 效果

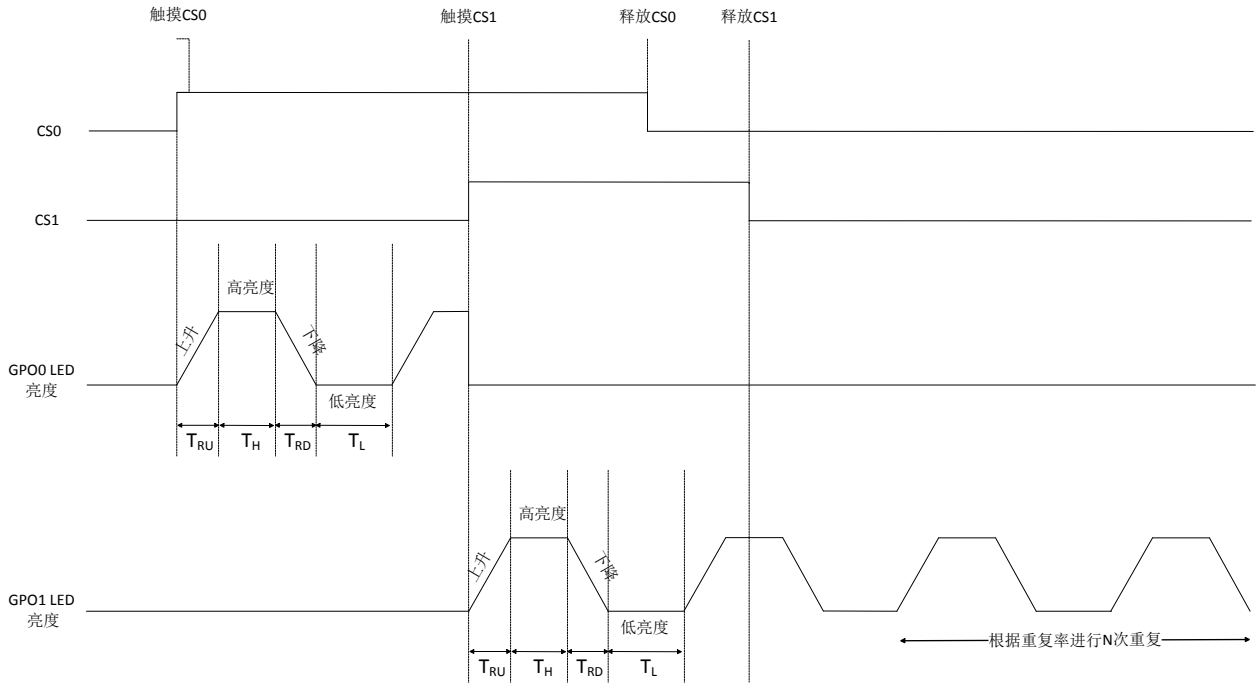


3.2.7.3 *最后按键 LED 效果*

如果其他按键被触摸，可以在一个 GPO 上中断按键触摸 LED 效果。该效果在第一个 GPO 上复位，并在与最后的触摸按键相关的 GPO 上开始生效，如图 3-19 所示。 默认情况下，该特性被禁用。

如果某些按键使能了切换ON/OFF功能，那么其最后按键LED效果将被禁用。如果侧翼传感器抑制（FSS），同时触摸两个按键，将不能使用最后按键LED效果，因为触摸的第二个按键不会有效。

图 3-19. 使能最后按键 LED 效果时的按键触摸 LED 效果（呼吸效果被使能）



3.2.7.4 待机模式下的 LED 亮度

当不触摸 CapSense 按键 CSx（即 OFF 状态）时，您可配置与 GPOx 相关的 LED 具有待机模式的 LED 亮度，用于 LED 背光。该配置可以提高该设计的界面外观。

可以分别将待机模式下的 LED 亮度配置为：0%、20%、 30%或 50%。 待机模式下的 LED 亮度应该与低亮度相同。

未触摸 CSx 时，在上电 LED 效果或按键触摸 LED 效果结束后，与 GPOx 相关的 LED 将保持待机模式中的 LED 亮度。

由于器件未进入低功耗睡眠模式，因此待机模式中的LED亮度使该器件的功耗递增。当该器件进入深度睡眠模式时，待机模式中的LED亮度将被禁用。

3.2.8 锁存状态读取

触摸 CapSense 按键 CSx 时，该器件通过将“提醒注意/睡眠”线置于低电平来为主机生成中断。然后，主机处理器通过 I²C 通信读取器件的寄存器映射，以识别 CapSense 按键的状态。 更多有关信息，请参考提醒/睡眠线部分中的内容。 更多有关 I²C 通信的信息，请参考《CY8CMBR2110 数据手册》中的内容。

当器件发送中断信号给主机时，主机不能立即响应该中断。因此，主机会遗漏按键触摸。为了避免遗漏按键触摸，主机需要读取按键状态（CS）和锁存状态（LS），以获取所有按键触摸正确的信息。在操作模式下，CS 被存储在 Button_Current_Stat0 和 Button_Current_Stat1 寄存器上。在操作模式下，LS 被存储在 Button_Latch_Stat0 和 Button_Latch_Stat1寄存器上。 有关寄存器映射的详细信息，请参考《CY8CMBR2110数据手册》附录中的内容。

按键状态位在按键被触摸时被置位，并在释放按键时被清除。锁存状态位在按键被触摸时被置位。主机读取按键状态时，将自动清除该位。

表 3-3 列出了一个按键被触摸时的各种情况及其代表的含义。这些情况如图 3-20 和图 3-21 所示。

表 3-3. 锁存状态读取

按键状态（CS）	锁存状态（LS）	说明
0	0	读取当前的 I²C 时，不触摸 CSx 主机已经确认了最后 I²C 读取时的任何前一个 CSx
0	1	当前的 I²C 读取之前，CSx 已被触摸 主机遗漏了该 CSx 触摸
1	0	前一个 I²C 读取时，CSx 被触摸并被主机确认了 读取当前 I²C 时，CSx 仍保持触摸的状态
1	1	在读取当前 I²C 时，CSx 被触摸

图 3-20. 锁存状态读取的第一种情况

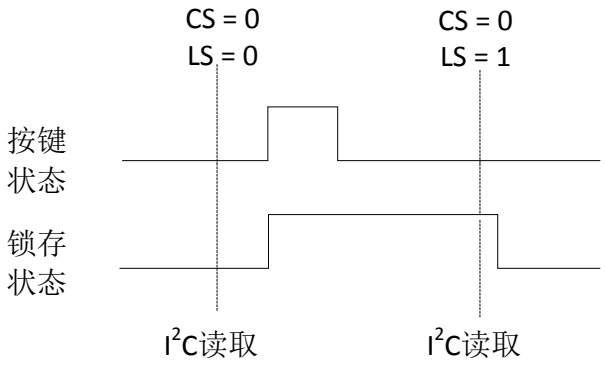
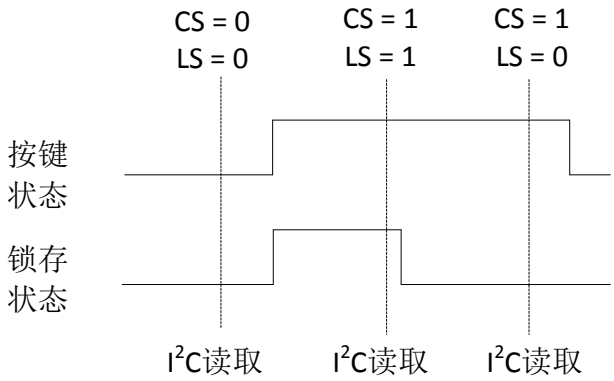


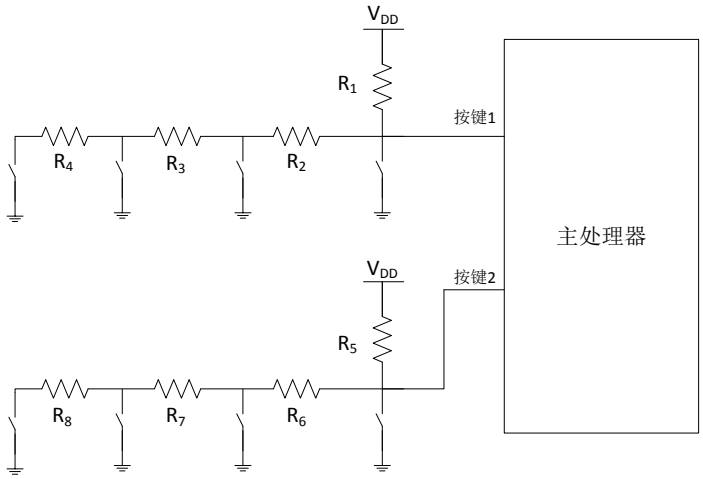
图 3-21. 锁存状态读取的第二种情况



3.2.9 模拟电压支持

一个带有主机处理器的通用外部电阻网络（如图 3-22 所示）可以根据输入引脚上的电压值来配置主机，使该主机执行不同的功能。 通过使用电阻组合以及连接在 V_{DD} 和地端的开关，可以调整该电压。

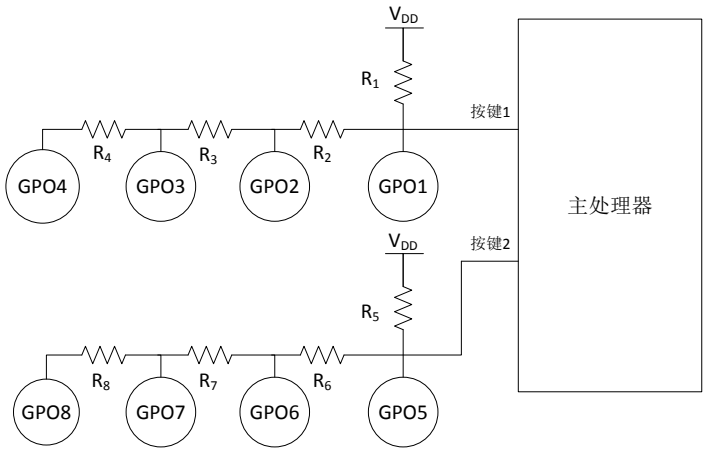
图 3-22. 通用的外部电阻网络



通过 CY8CMBR2110 的模拟电压支持特性，您可以使用 CapSense 按键控制这些开关。每一个开关都可以被替换为一个 GPOx。当一个 CSx 按键被触摸时，相应的 GPOx 变低，因此开关被关闭（短路接地）。当该按键被释放时，相应的开关便为打开状态， 如图 3-23 所示。

如果启用了该功能，便不能将 GPO 同时用于外部电阻网络和驱动 LED。如果仅需要一个按键的状态为 ON 提供模拟电压支持，则该特性会使能 FSS。一般情况下，GPO 引脚处于强驱动模式，然而使能该特性时，GPO 引脚均处于开漏低电平的驱动模式。

图 3-23. CY8CMBR2110 的模拟电压支持



3.2.10 灵敏度的控制

可以单独设置每个 CapSense 按键的灵敏度。灵敏度确定了触摸一个按键所需要的最小 C_F。影响到按键的灵敏度有下列因素：

1. 外覆层厚度：外覆层越厚，会需要更高的灵敏度。
2. 系统噪声：如果系统的噪声增加，则需要下降灵敏度来避免按键的误触发。
3. 设计的外形：要想支持一个低灵敏度（即较高的 C_F），则需要使用较大的按键。对于小直径的按键，则需要较高的灵敏度。
4. 功耗：按键的灵敏度提高，则功耗将增加。 如果需要低功耗，灵敏度必须为低。

灵敏度的设置可分别为：“高”、“中”、“低”三个级别。

3.2.11 去抖动的控制

通过指定最短时间使某个按键因发生有效触摸输入而被接触，防抖动特性可避免由于噪声毛刺或系统故障而造成的错误按钮触发。

去抖动时间根据按键的功能而改变。例如，电源按键需要一个较长的去抖动时间，以避免系统意外切换 ON/OFF 状态。 较短的去抖动时间可以加快器件对按键接触的响应。

可以从 CS1 - CS9 按键单独设置 CS0 按键的去抖动值。该功能可用于配置 CS0 按键具有特殊性能（如电源按键）的设计。去抖动的取值范围为：1-255。

器件的响应时间取决于按键的去抖动值。显示的是具有不同去抖动值的器件的响应时间示例[6]。为了计算去抖动值的响应时间，请参考响应时间部分中介绍的内容。

表 3-4. 去抖动值的响应时间示例

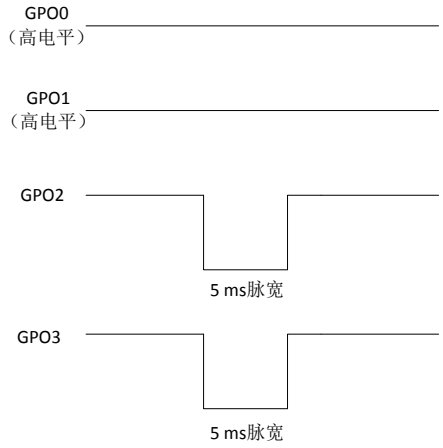
去抖动值	连续按键触摸的响应时间（单位为毫秒）
1	70
4	105
7	140
10	175
100	1225
200	2380
255	3010

[6] 8 个按键、抗干扰能力为“标准”、优化响应时间的设计

3.2.12 系统诊断

内置的加电自检（POST）机制在加电复位（POR）时执行五次测试，这样对生产测试过程十分有用。如果有任何按键上的诊断失败，抗干扰能力为“标准”时，相应的 GPO 会在 350 ms 内发出 5 ms 的脉冲；抗噪能力为“High” 时，则在 1000 ms 内发出 5 ms 的脉冲。

图 3-24. 示例：CS0、CS1 通过 POST，CS2、CS3 时失败



要想查找系统诊断的结果，请使用 EZ-Click 定制工具。欲了解有关该工具的详细信息，请参考《EZ-Click 定制工具的用户指南》中的内容。

如果需要读取整个器件的数据，您可以将该器件的寄存器映射模式修改为“生产线测试”模式，然后通过使用 I²C 数据线来读取数据。有关修改寄存器映射模式的信息，请参考《CY8CMBR2110 数据手册》中的寄存器映射模式。有关器件数据的更详细信息，请参考 I2C 通信一节中介绍的内容。

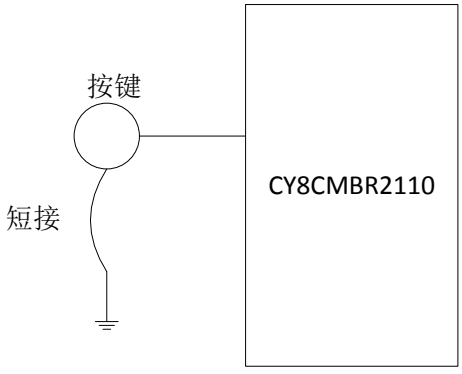
由于您可以使用 I²C 来读取 GPO 的状态，因此不需要将这些 GPO 与主机控制器引脚相连。

需要对所有按键执行下列测试。

3.2.12.1 按键短路接地

所有短路接地的按键都被禁用。

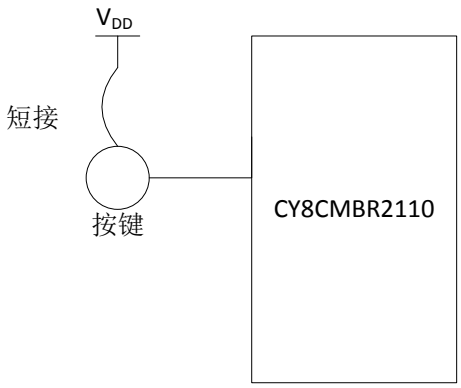
图 3-25. 按键短路接地



3.2.12.2 按键短路连接 V_{DD}

所有短路连接 V_{DD} 的按键都被禁用。

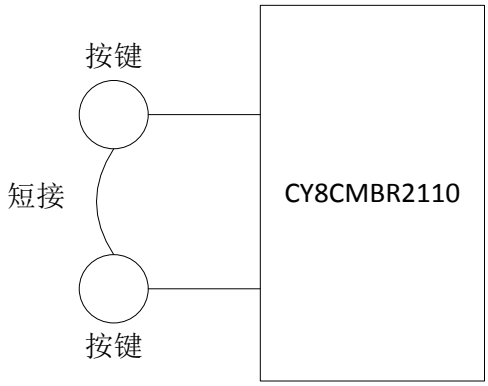
图 3-26. 按键短路连接 V_{DD}



3.2.12.3 按键与按键短路

如果有两个或更多的按键被相互短路连接，则这些按键都被禁用。

图 3-27. 按键与按键短路



3.2.12.4 CMOD 的错误值

建议 CMOD 的值为 2.2 nF ±10%。

如果发现 CMOD 的值小于 1 nF 或大于 4 nF，那么所有的按键都将被禁用。

3.2.12.5 按键 C_P > 40 pF

如果发现有任何按键 C_P 的值大于 40 pF，则该按键被禁用。

3.2.13 按键扫描速率

按键扫描速率指定了器件进行连续按键扫描的时间间隔。使用下面公式可以计算该速率：

按键扫描速率 = 按键扫描速率常量 + 按键扫描速率偏移量　　公式 4

可将按键扫描速率配置为 25 - 561 毫秒。

按键扫描速率常量由按键数量和所选的抗干扰能力级别确定。如果按键数量越多，则该常量越高。同样，如果抗干扰能力为“高”，则该常量也会更高。

如果您最多使用五个按键，则按键扫描速率常量将取决于优化设计的方式。

响应时间优化：连续按键扫描之间的时间被缩短。在固定的时间内发生的扫描更多时，器件会更快地响应按键触摸。 然而，功耗也会相应增加。

功耗优化：连续按键扫描之间的时间更长。在固定时间内进行的扫描次数更少，则器件将需要更长的时间来响应按键触摸。这样可以降低功耗。

您可以使用 EZ-Click 定制工具来配置按键扫描速率的偏移量。表 3-5 显示的是按键扫描速率常量。

表 3-5. 按键扫描速率常量

按键数量	按键扫描速率常量			
	响应时间的优化		功耗的优化	
	抗干扰能力“标准”	抗干扰能力“高”	抗干扰能力“标准”	抗干扰能力“高”
≤ 5	25	35	35	55
> 5	35	55	35	55

例如，假设一个设计中包括四个按键，其参数如下：

- 所有按键的 C_P 值为 10 到 20 pF
- 所有按键的灵敏度均为“高”
- 抗干扰能力为“标准”
- 每个按键的去抖动值为 10
- 每小时的平均按键触摸次数为 200
- 平均触摸时间为 1000 毫秒
- 蜂鸣器和按键触摸 LED 效果被禁用
- 按键扫描速率偏移量为 0
- 每个按键的电流消耗为：
  - 优化的时间响应电流 = 0.3075 mA
  - 优化功耗电流 = 0.2204 mA

第一个按键触摸以及连续按键触摸的响应时间为：

  - 优化的时间响应时间 = 125 ms
  - 优化功耗时间 = 175 ms

请注意，与功耗优化的设计相比，响应时间优化的设计消耗更多的功耗，但可以更快地响应一个按键触摸。要想查找您设计的响应时间，请参考设计工具箱中的内容。

按键扫描速率在不同的器件上会有所改变，另外在-40 °C 到+85 °C 的温度范围内，它的精度为±10%。

3.2.14 I²C 通信

I²C 是 CY8CMBR2110 （I²C 从器件）和主机（I²C 主器件）之间的通信接口。

欲了解更多有关该协议和通信流程的信息，请参考《CY8CMBR2110 数据手册》中的“I²C 通信”部分。

为了将主机和器件之间的 I²C 配置为更合适的通信，请按照下列指南进行操作：

- 在初始化任何 I²C 通信之前，主机处理器需要将“注意/睡眠”线置于低电平，否则该器件将取消确认主机的通信。
- 主机处理器不应该初始化或继续进行与该器件的 I²C 通信，除非发生下列情况：
  - 主机需要配置该器件
  - 该器件中断主机
  - 主机需要读取并验证该器件寄存器映射的内容

- 要想降低功耗，需要避免延长与该器件进行 I²C 通信的时间。
- 在初始化任何 I²C 通信之前，主机需要在器件通电后等待 350 毫秒（如果抗干扰能力为“标准”）或 1000 毫秒（如果抗干扰能力为“高”）。 否则，该器件将取消确认该通信。
- 在初始化新的数据操作之前，主机应该在任何 I²C 数据操作后至少等待 60 毫秒。
- 在初始化任何 I²C 通信之前，主机需要在发送“保存到闪存内”或“软件复位”指令后等待 350 毫秒（如果抗干扰能力为“标准”）或 1000 毫秒（如果抗干扰能力为“高”）。
- 该器件应该在操作模式下运行。
- 主机初始化器件的新 START 条件前，需要初始化 I²C 通信的 STOP 条件。 它同样被称为重启（Repeat Start）条件。
- 主机应该保持各连续 I²C 数据操作的时间间隔至少 60 毫秒。
  - ☐ 如果在 60 毫秒内，主机初始化了另一个 I²C 数据操作，则它将收到与前一数据操作相同的数据。
  - ☐ 如果在该期间，主机将新数据写入到与前一数据操作相同的寄存器内，则旧数据将被丢失。
  - ☐ 如果在该期间内，主机将新数据写入到与前一数据操作不同的寄存器内，则寄存器会保留该数据。前一数据操作中的数据不被丢失。

3.3 设计工具箱

设计工具箱有助于设计一个 CY8CMBR2110 CapSense 解决方案。它提供了有关电路板布局和功能设置的基本信息，并对设计是否适合批量生产提出了建议。

3.3.1 通用布局指导

图 3-28 总结了 CY8CMBR2110 的布局指南。 这些指南在电气和机械设计中的注意事项中进行了说明。有关该材料的详情，可以在 CapSense 入门手册中找到。

图 3-28. 设计布局建议

General Layout Guidelines

Sl. No.	Category	Min	Max	Recommendations/Remarks
1	Button shape			Solid round pattern, round with LED hole, rectangle with round corners
2	Button size	5 mm	15 mm	Given in Layout Estimator sheet
3	Button-button spacing	equal to button ground clearance		8 mm
4	Button-ground clearance	0.5 mm	2 mm	Given in Layout Estimator sheet
5	Ground flood - top layer			Hatched ground 7 mil trace and 45 mil grid (15% filling)
6	Ground flood - bottom layer			Hatched ground 7 mil trace and 70 mil grid (10% filling)
7	Trace length from sensor pad to device pin		450 mm	450 mm is for FR4 PCB, with a button diameter of 5 mm and a pin capacitance of 7 pF. For a different design, refer to Layout Estimator sheet.
8	Trace width	0.17 mm	0.20 mm	0.17 mm (7 mil)
9	Trace routing			Traces should be routed on the non sensor side. If any non CapSense trace crosses CapSense trace, ensure that the intersection is orthogonal.
10	Via position for the sensors			Via should be placed near the edge of the button to reduce trace length thereby increasing sensitivity.
11	Via hole size for sensor traces			10 mil
12	Number of via on sensor trace	1	2	1
13	CapSense series resistor placement		10 mm	Place CapSense series resistors close to the device for noise suppression. CapSense resistors have highest priority compared to LED resistors. Place them first.
14	Distance between any CapSense trace to ground flood	10 mil	20 mil	20 mil
15	Device placement			Mount the device on the layer opposite to the sensor. The CapSense trace length between the device and the sensors should be minimum (see trace length above)
16	Placement of components in two layer PCB			Top layer - sensors and bottom layer - device, other components and traces.
17	Placement of components in four layer PCB			Top layer-sensors, 2nd Layer – CapSense traces & Vdd and avoid the Vdd traces below the sensors, 3rd Layer-hatched ground, Bottom layer- device other components and non CapSense traces
18	Overlay thickness	0 mm	5 mm	Use layout Estimator sheet to decide on overlay, given maximum limit is for plastic overlay.
19	Overlay material			Should be non-conductive material. Glass, ABS Plastic, Formica, wood etc. No air gap should be there between PCB and overlay. Use adhesive to stick the PCB and overlay.
20	Overlay adhesives			Adhesive should be non conductive and dielectrically homogenous. 467MP and 468MP adhesives made by 3M are recommended.
21	LED backlighting			Cut a hole in the sensor pad and use rear mountable LEDs.
22	Board thickness			Standard board thickness for CapSense FR4 based designs is 1.6 mm.

3.3.2 布局估计

布局估计根据预期终端系统的要求和工业设计提供最小的按键尺寸和最大的走线长度。输入包括覆层材料、覆层厚度、电路板材料的走线电容及 CapSense 按键的灵敏度。图 3-29 中的列表 B 显示的是不同覆层材料的绝缘常数，以及不同 PCB 的单位长度的走线电容。表 A 根据三个系统噪声条件，计算了设计的最小按键直径和最大走线长度。“低”、“中”和“高”噪声条件是有助于按键开发的相关质量指标。根据终端系统的环境，各按键的噪声条件会有所不同。 如果未知噪声条件，请使用中等噪声条件作为起点。每个按键的实际噪声在设计验证阶段决定。

使用此表的输出来指导按键电路板的布局过程；然后在进行原型设计之前，使用 C_P、功耗和响应时间计算器表来检查该设计，具体如 CP、功耗以及响应时间计算器节中所述。

图 3-29. 布局估计

Layout Estimator

TableA: Layout Estimator

Input Parameters	Value	Units	Comments
Overlay Thickness	1.5	mm	
Overlay - Dielectric constant	4	farad/m	
Capacitance of trace per inch	2	pF	
CSx Sensitivity	High		If the power consumption is critical, select "Low" sensitivity. If the board form factor is critical, select "High" sensitivity.
Minimum Recommended Button Diameter			
Noise Conditions - Low (0.05 pF Noise)	5	mm	
Noise Conditions - Medium (0.075 pF Noise)	6	mm	
Noise Conditions - High (0.1 pF Noise)	7	mm	
Maximum Trace length			
Noise Conditions - Low (0.05 pF Noise)	412	mm	
Noise Conditions - Medium (0.075 pF Noise)	406	mm	
Noise Conditions - High (0.1 pF Noise)	400	mm	
Button to Ground clearance	1.5	mm	

	input cells, edit with actuals
	output cells, based on inputs

Note: Button diameter of all the buttons CS1 to CS9 will be same with respect to overlay thickness, but can differ with respect to noise conditions

TableB - Industry Standard Reference Values

Overlay Material	Dielectric constant
Plastic	2.8
Plexi glass	8
Formica	4.6-4.9
Glass (Standard)	7.6-8.0
Glass (Ceramic)	6
Mylar	3.2
ABS Plastic	3.8 – 4.5
Wood	1.2-2.5
Trace and board type	**Capacitance per inch in pF**
Copper trace , PCB, 2 layer, 64mil, FR4	2
Copper trace, flex PCB, 2 layer	8

输入

- 外覆层厚度
- 外覆层绝缘常数
- 电路板每英寸走线的电容
- CSx 灵敏度

输出

- 不同噪声条件下所建议的最小按键直径和最大走线长度
- 按键离地间隙

随着每个按键上噪声的变化，这些按键的直径将有所不同。

3.3.3 C_P、功耗以及响应时间计算器

在完成电路板布局后，可以在构建按键电路板原型前通过功耗和响应时间计数器（如图 3-30 所示）来检查该设计。要验证每个按键的 C_P 值，请在表 A 中输入按键的直径以及走线长度。然后，工具箱会确认各个按键的 C_P 值是否都在指定的范围（5 pF 至 40 pF）内。

表 B 中的功耗估计量用于计算功耗。 功耗是一个由按键扫描速率、抗干扰能力等级以及工作时间的比例组成的函数。 通过将每小时的平均按键触摸数次数乘以按键触摸时间所得到的时长、蜂鸣器为 ON 的时间和按键触摸 LED 效果三个值中的最大值，可以计算工作时间。 该值将被转换为工作时间的百分比，然后计算功耗。 需要确保下列输入单元不同时为空（或为 0）：

1. 每小时的平均按键触摸次数
2. 平均按键触摸时间
3. 蜂鸣器为 ON 的平均时间
4. 按键触摸 LED 效果的平均时间

根据表 A 和表 B 的输入，表 C 将输出按键的响应时间。去抖动值对按键响应时间产生影响。

图 3-30. CP、功耗以及响应时间计算器

Cp, Power Consumption and Response Time Calculator

Table A: Cp Calculator

Sensor	Button diameter		Trace length		Sensitivity	Parasitic capacitance (Cp) of sensors (Approx)		Comments
CS0		mm		mm	Medium	0	pF	
CS1		mm		mm	Medium	0	pF	
CS2		mm		mm	Medium	0	pF	
CS3		mm		mm	Medium	0	pF	
CS4		mm		mm	Medium	0	pF	
CS5		mm		mm	Medium	0	pF	
CS6		mm		mm	Medium	0	pF	
CS7		mm		mm	Medium	0	pF	
CS8		mm		mm	Medium	0	pF	
CS9		mm		mm	Medium	0	pF	
Total No of buttons	0	Nos						

Table B: Power calculator

Button Scan Rate offset	506	ms
Design optimization	Response Time	
Noise Immunity level	High	
Approximate Button Scan Rate value	541	ms
Average number of button touch per hour	50	
Average button touch time	500	ms
CS0 Debounce	1	
CS1 - CS9 Debounce	1	
Average Buzzer ON time	0	ms
Average Button Touch LED Effects time	1000	ms
Standby Mode LED Brightness	Disabled	
Current consumption calculation factor	Typical	
Sleep Current	0.00952	mA
Active Current	3.4	mA
Average Current without Finger	0	mA
Average Current with Finger	0	mA
Actual average current consumption	0	mA
Actual average current consumption per butt	0	mA

Table C: Response time calculator

CS0 First button press	576	ms
CS0 Consecutive button press	70	ms
CS1-CS9 First button press	576	ms
CS1-CS9 Consecutive button press	70	ms

	input cells, edit with actuals
	output cells, based on inputs

Note: The power values given here are for the worst case, the actual power values will be lower.

输入

- CS0 到 CS9 的按键直径和走线长度，如布局中的设计
- CS0 - CS9 的灵敏度
- 按键扫描速率偏移量
- 设计优化
- 抗干扰能力级别
- CS0 的去抖动
- CS1 - CS9 的去抖动
- 每小时的平均按键触摸次数
- 平均按键触摸时间
- 蜂鸣器为 ON 的平均时间
- 按键触摸 LED 效果的平均时间
- 待机模式下的 LED 亮度
- 电流消耗的计算因素

输出

- 每个按键的 C_P 值（确认 C_P 值是否在 5 pF 到 40 pF 的范围内）
- 每个按键的电流消耗
- 按键的响应时间

3.3.4 设计验证

在编译和测试原型电路板后，请使用 EZ-Click 定制工具来捕获所有按键的原始信号、噪声信号和 C_P（请参考《EZ-Click 用户指南》）。您可以使用该信息和设计验证表来验证设计，如设计验证部分中的介绍。

表 A 显示的是来自上表的不同设计参数值，因此您不需要向该表输入任何数据。此表为原型电路板提供通过/失败等级。 如果设计失败，您可以通过向表 A 输入新的数值来修改系统，并将收到其他的建议和结果。如果设计通过，则使表 A 中的 New value（新值）列为空。

表 B 显示的是来自 C_P、功耗和响应时间计算器表的按键灵敏度的值。如果设计失败，您可以通过输入新值重新设计该按键的灵敏度。如果设计通过，则使表 B 中的 New value（新值）列为空。

图 3-31. 设计验证

Design Validation

Table A: Actual Design values

Input Parameters	Initial value	New value	Units
Overlay Thickness (in mm)	1.5		mm
Dielectric constant, overlay	4		farad/m
Capacitance of trace per inch i	2		pF
Button Scan Rate offset	506	506	ms
Design Optimization	Response Time	Response Time	
Noise Immunity Level	High	High	
Button Scan Rate Value	541	541	ms
Average number of button touch	50	50	
Average button touch time	500	500	ms
Average Buzzer ON time	0	0	ms
Average Button Touch LED Effect	1000	1000	ms
Standby Mode LED Brightness	Disabled	Disabled	
Current consumption calculatio	Typical	Typical	
No of buttons	0	0	Nos
CS0 Button diameter actual			mm
CS1 Button diameter actual			mm
CS2 Button diameter actual			mm
CS3 Button diameter actual			mm
CS4 Button diameter actual			mm
CS5 Button diameter actual			mm
CS6 Button diameter actual			mm
CS7 Button diameter actual			mm
CS8 Button diameter actual			mm
CS9 Button diameter actual			mm

Table B: Button Sensitivity

Button	Initial value	New value
CS0	Medium	
CS1	Medium	
CS2	Medium	
CS3	Medium	
CS4	Medium	
CS5	Medium	
CS6	Medium	
CS7	Medium	
CS8	Medium	
CS9	Medium	

Table C: Reference values

Overlay Material	Dielectric constant
Plastic	2.8
Plexi glass	2.6-3.5
Formica	4.6-4.9
Glass (Standard)	7.6-8.0
Glass (Ceramic)	6
Mylar	3.2
ABS Plastic	3.8 – 4.5
Wood	1.2-2.5
Trace and board type	**Capacitance per inch in pF**
copper trace , PCB, 2 layer, 64mil,FR4	2
copper trace , flex, 2 layer	8

	input cells, edit with actuals
	output cells, based on inputs

For Table A: The Initial values of "Input Parameters" are the ones you have entered in the previous sheets. If your design passes, leave the "New value" column blank. If your design fails, enter the New values for the

Table D: Power consumption, Button diameter actuals

Sensor	Values taken from I2C						Average Current		Improvement Recommendations					
	Noise		Cp		Raw				Minimum			Maximum		
CS0		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS1		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS2		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS3		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS4		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS5		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS6		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS7		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS8		counts		pF	0	counts	0	mA	0	mm		0	mm	
CS9		counts		pF	0	counts	0	mA	0	mm		0	mm	
			Actual average current consumption				0	mA						

Note: While logging debug data for this sheet, make sure there is no finger present on the sensors for the log duration

要想使用 EZ-Click 定制工具将数据输入到表 D 内，请进行下列操作：

1. 通电器件并通过 USB-I²C 桥接器（CY3240-I2USB 桥接器）将该器件连接到您的计算机。有关 USB-I²C 桥接器（CY3240-I2USB 桥接器）的详细信息，请参考《AN2397 — CapSense 数据的查看工具》中介绍的内容。
2. 打开EZ-Click定制工具并创建新的项目。选择赛普拉斯CY8CMBR2110器件。从“端口选择”窗口中选择您正在使用的端口，然后点击**Connect**（连接）。
3. 打开“Device Config”（器件配置）选项卡，并选择用于设计的按键数量。将CapSense分配给相应的按键（若需要）。设置手指阈值或选择自动阈值。
4. 打开CapSense输出选项卡，并选择“Button Specific Output”（按键特殊输出）窗口。
5. 选择您想查看CapSense输出的按键。选择Raw Count vs Baseline（原始信号与基线）图。
6. 观察原始信号图并注释300个采样的平均原始信号和按键CP。此外，请注意按键C_P。
7. 使用下列公式计算噪声信号：

   噪声信号 = 最大的原始信号 - 最小的原始信号（在300个采样内选出）
8. 通过将该数据输入到表D查找电流消耗值，并确定能否批量生产该设计。

输入

- 原始信号
- 噪声信号
- 按键 C_P
- 如果设计失败，请注意下面的内容：
  - ☐ 每个按键新的外覆层厚度、外覆层材料介电常数和按键直径，以及走线电容
  - ☐ CSx 的灵敏度

输出

- 每个按键的电流消耗
- 设计更改建议。如果按键大小或走线长度超过了最佳设计范围，则设计工具箱根据设计的实际值提供建议。

如果按键电路板未通过，则设计工具箱将提供建议以指导您实现可通过的结果。您可以通过更改四个因素来纠正失败设计：按键大小、走线长度、外覆层材料和外覆层厚度。更改按键大小或走线长度需要重做电路板，而更改外覆层材料或厚度（或两者）则可能通过设计完成。最佳的解决方案依赖于您当前处于设计周期的哪个阶段，以及您终端系统的要求。

3.4 配置 CY8CMBR2110

可以使用下列方法来配置 CY8CMBR2110：

1. EZ-Click 定制工具
2. 使用主机处理器配置器件
3. 第三方的编程器

配置 CY8CMBR2110 器件的通用流程列出了各个具体步骤。此流程适用于所有配置方式。EZ-Click 定制工具自动遵照该流程，但主机处理器必须遵照下列步骤：

1. 将器件模式修改为 LED 配置模式。
2. 等待 55 毫秒。
3. 写入 LED 配置模式中的所有配置寄存器。

4. 等待 55 毫秒。
5. 将器件模式修改为器件配置模式。
6. 等待 55 毫秒。
7. 写入器件配置模式中的所有配置寄存器。
8. 计算该校验和并将该值输入到“Checksum_MSB” （0x1E）和“Checksum_LSB” （0x1F）寄存器内（这两个寄存器都处于器件配置模式）。

   校验和：该校验和是寄存器（0x01—0x1F）（处于 LED 配置模式）的值和寄存器（0x01—0x1D）（处于器件配置模式）的值之和。该校验和还包含任何保留寄存器位的值。计算该校验和时，主机不应写入这些位而应当向每一位添加 0。

   处于操作模式下的 Checksum_Flash_xxx 寄存器指出存储在闪存内的校验和。

   处于操作模式下的 Checksum_RAM_xxx 寄存器指出器件为当前配置计算并被存储在 RAM 内的校验和。
9. 等待 55 毫秒。
10. 读取 Host_Mode 寄存器（处于器件模式）中的“校验和匹配”位，以验证它被置 1。如果该位未被置 1，则从第一步重新操作，然后重新配置该器件。 主机应该保留该配置数据的备份（若需要）。

    “校验和匹配”位：CY8CMBR2110 计算该校验和，并将该值与主机输入的校验和寄存器值进行比较。如果这两个值相匹配，Host_Mode 寄存器中的“校验和匹配”位将被置 1。如果它们不匹配，则表示 I^2C 的写操作可能发生了错误，而且该位将被清零。主机可以通过读取处于操作模式的 Checksum_RAM_xxx 寄存器来使该器件计算校验和。
11. 如果“校验和匹配”位被置 1，则置位 Host_mode 寄存器中的“存储到闪存”位。

    存储到闪存内：在“存储到闪存内”过程中，将执行下列操作：

    (i) 该器件将处于 LED 配置模式和器件配置模式的 64 字节数据复制到闪存内。

    (ii) 执行软件复位。

    (iii) 软件复位完成后，该器件处于操作模式。

    如果还未将配置的修改内容保存到闪存内，则修改内容不可用。 当器件仅被配置一次，用于将来的所有操作，“存储到闪存内”的过程则非常有用。 存储到闪存过程中，器件的电源需要稳定，而且 V_{DD} 的波动限制为±5%。
12. “存储到闪存内”后，等待（T_{SAVE_FLASH} + 器件初始化）的一段时间。T_{SAVE_FLASH} 在 CY8CMBR2110 数据手册的“闪存写入时间规格”一节中进行了介绍。 如果抗干扰能力为“标准”，器件的初始化时间为 350 毫秒；如果抗干扰能力为“高”，则该时间为 1000 毫秒。
13. 读取操作模式中 Device_Stat 寄存器的“Factory defaults loaded”（加载出厂默认）位。

    Factory Defaults Loaded（加载出厂默认）：执行每次复位时，该器件将闪存内的内容加载到 RAM 内，并使用闪存校验和来验证 RAM 校验和，以确保没有闪存损坏。如果该校验和有异，则器件指出有闪存损坏，并加载 RAM 中的工厂默认值，然后置位“Factory defaults loaded”（加载出厂默认）位。这样会复位主机以前所修改的任何寄存器值。每个寄存器的工厂默认值被存储在寄存器映射中。

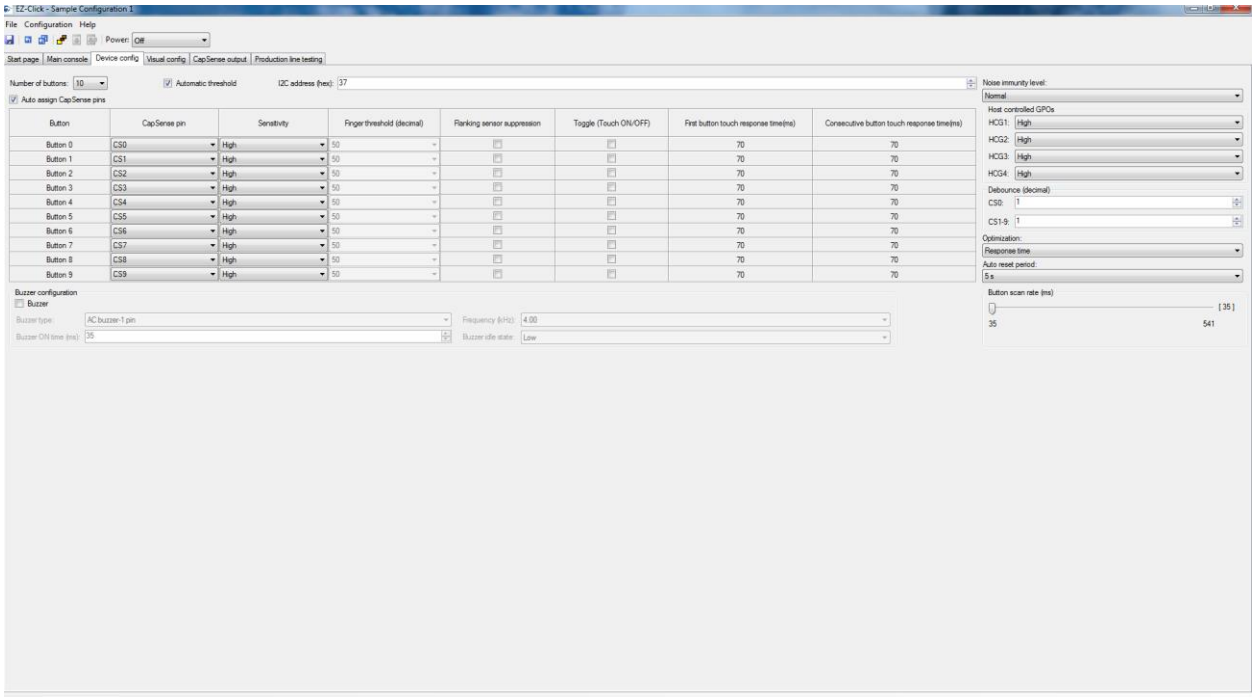
    如果加载了工厂默认值，器件的 I^2C 地址也从主机设置的当前地址修改为默认地址 37h。加载工厂默认值后，主机必须使用 I^2C 总线上的默认 I^2C 地址与 CY8CMBR2110 通信。
14. 如果置位了“Factory defaults loaded”位，则会损坏闪存，而且主机需要从第一步骤重新配置该器件。如果将该位清零，将成功地配置器件。

注意：有关这些步骤使用的不同模式和寄存器的详细信息，请参考《CY8CMBR2110 数据手册》。

3.4.1 EZ-Click 定制工具

EZ-Click 定制工具是一个用于配置器件的简单和直观的图形用户界面。它包含了所有需要的参数并通过使用 I²C 接口来配置器件。

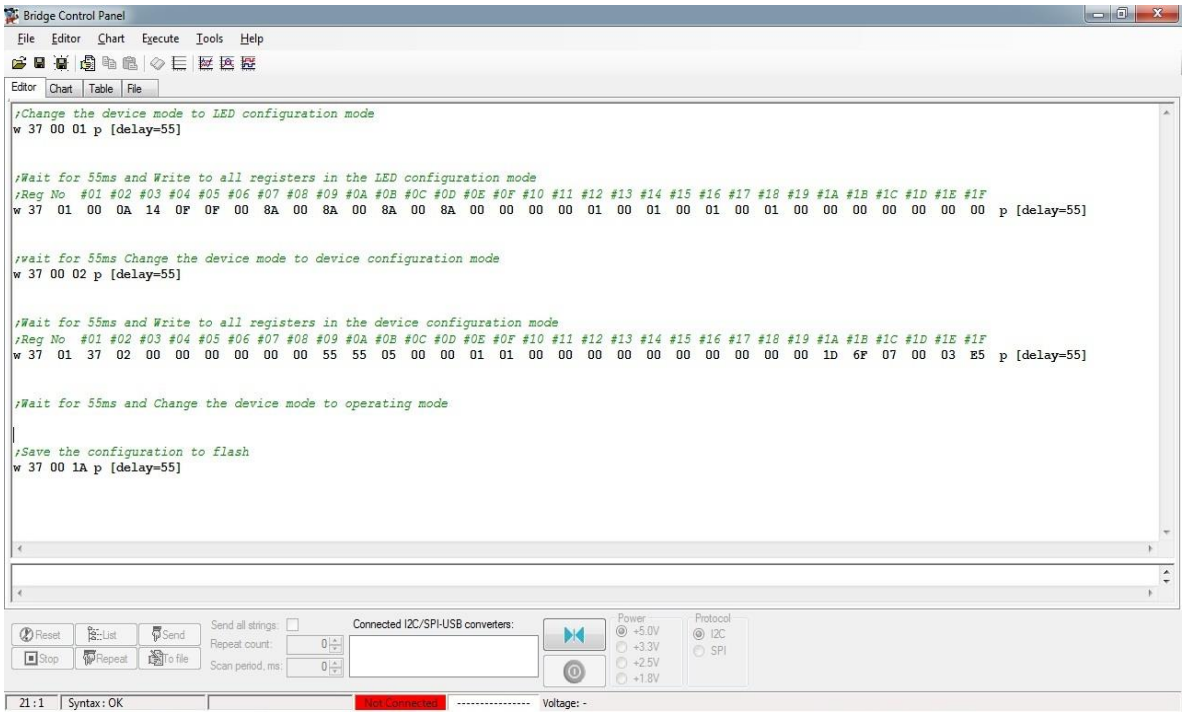
图 3-32：EZ-Click 定制工具



EZ-Click定制工具显示了器件的实时CapSense数据。您可以查看按键和参数的特定数据，包括CapSense按键状态、C_P、原始信号、手指阈值和SNR。由于该工具显示系统诊断结果和CapSense按键的SNR，并指出该SNR是否满足您的要求，因此它可用于生产线测试。更多有关该工具的信息，请参考《EZ-Click用户指南》中介绍的内容。

您可以保存该配置，以用于其他采样。您也可以使用该工具来生成包含所需要的I²C指令的配置文件，用于配置器件。 为此，需要打开桥接器控制面板中的配置文件（有关桥接器控制面板的信息，请查看AN2397 — CapSense数据查看工具）并通过USB-I²C桥接器将各指令发送到器件。显示的是一个示例配置文件。

图 3-33. 由 EZ-Click 定制器工具生成的配置文件示例



3.4.2 使用主机处理器配置器件

使用主机处理器来配置 CY8CMBR2110 器件时，主机处理器需要根据特定的顺序来调用完整列表中的 API。这些 API 使用 I²C 通信来配置器件特性、读取 CapSense 数据、驱动主机控制 GPO、执行生产线测试、配置功耗设置等。您可以从 http://www.cypress.com/?rID=74590 网站下载源代码。

使用主机处理器来配置 CY8CMBR2110 器件的优点如下：

- 系统内的配置 — 不需要从电路板中取出器件（芯片）
- 运行时间配置 — 主机处理器自动修改各特性

这些 API 主要分为高层 API 和低层 API。高层 API 独立于硬件（平台）并可以在任何主机处理器上运行。低层 API 是为 CY8C29466-24PXI 器件而开发的，而且也依赖于硬件（平台）。如果您的应用中有其他的主机处理器，则需要修改低层 API 的固件。

3.4.2.1 高层 API

高层 API 可用于使能或禁用按键触摸 LED 亮度、设置手指阈值参数、配置扫描速率、修改 I²C 地址和许多其他功能。

高层 API 包含一个代码。该代码可用于读取或写入到 CY8CMBR2110 的相应寄存器上，并计算各配置的校验和。它们调用特定于主机处理器的低层 API，并执行主机处理器和器件之间的物理 I²C 通信。

高层 API 头文件（High_Level_API.h）包含用于所有高层 API 的函数原型。 当配置 CY8CMBR2110 器件时，所需要的“.C”文件必须包含该头文件。高层 API 将使用定义在 High_Level_API.h 文件中的宏来配置内部。您不能修改这些宏。

例如：

```
#define  I2C_CFG_REG   (0x01)
```

3.4.2.2 低层 API

低层 API 用于使能主机处理器与器件的物理 I²C 通信。这些低层 API 使用 PSoC I2CHW 用户模块来执行读写操作。根据您实现 I²C 协议的方式，则可能会需要修改低层 API 代码。

低层 API 头文件（Low_Level_API.h）包含用于低层 API 的函数原型和低层 API 使用的宏。这些宏主要用于 I²C 通信和软件延迟的子程序， 而且也被定义以用于 CY8C29466-24PXI 器件。如果将这些宏用于主机 I²C 实现，则需要修改它们的定义。

例如，如果 CY8CMBR2110 器件取消确认，CY8C29466-24PXI （PSoC1）中的 I2CHW 用户模块将返回 0x00。因此，宏 **I2C_NACK** 被定义为 0x00。如果您使用其他主机处理器，当器件取消确认时返回其他值，则需要修改 **I2C_NACK** 为匹配值。

需要有软件延迟 API，以提供等于按键扫描速率的延迟。每次执行写指令后都需要该延迟。如果您想使用硬件资源（计时器）来实现该延迟，则要清除相应宏，从而禁用软件延迟子程序，如表 3-7 中所介绍。

表 3-6 中列出了这些独立于主机控制器的宏。取决于主机控制器的宏如表 3-7 所示。

表 3-6：独立于主机控制器的宏

宏名称	用途
FLASH_WRITE_TIME	发送存储到闪存内的指令后，CY8CMBR2110 器件准确地存储数据操作需要一段时间
TOTAL_BUTTON_COUNT	CY8CMBR2110 器件中的最大按键数量
FACTORY_DEFAULT_CHECKSUM	CY8CMBR2110 器件的工厂默认校验和
DEFAULT_SLAVE_ADDRESS	CY8CMBR2110 器件的工厂默认 I²C 地址
DELAY_CONST	用于计算软件延迟需要的迭代次数
SLAVE_NACK	CY8CMBR2110 器件取消确认时，用于清除 I²C 标志
SLAVE_ACK	CY8CMBR2110 器件确认时，用于设置 I²C 标志
SLAVE_BUF_PTR	用于设置指向寄存器映射上特殊寄存器地址的主机 I²C 缓冲区的指针

表 3-7：取决于主机控制器的宏

宏名称	用途
I2C_WRITE_COMPLETE	检查对 CY8CMBR2110 器件进行的 I2C 写操作是否完成。 写操作完成时，I2CHW 用户模块将返回 0x50。
NACK_RETRY_LIMIT	CY8CMBR2110 器件取消确认时，定义主机处理器重新尝试的次数。 典型值为 20。 可以将该值修改为适用于您的应用程序的值。
DELAY_ROUTINE_USED	用于使能/禁用软件延迟子程序。 如果该值为 1，将使能软件延迟。如果该值为 0，则禁用软件延迟。**如果您使用硬件资源来实现该延迟，需要禁用软件延迟子程序。** **注意：** 软件延迟子程序是一个阻塞代码。 该代码在确定的时间内停止 CPU。
I2C_NACK	用于检查 CY8CMBR2110 器件是否取消确认当前的 I²C 操作。 读/写操作被取消确认时，I2CHW 用户模块将返回 0x00。
I2C_READ_COMPLETE	检查对 CY8CMBR2110 器件进行的 I²C 写操作是否完成。 写操作完成时，I2CHW 用户模块将返回 0x15。
NEW_SLAVE_ADDRESS	新的从器件地址的值。 如果主机使用 MBR_SetI2CSlaveAddress API 作为 CY8CMBR2110 器件的默认从器件地址，它需要重新定义该宏为新的从器件地址。
CLOCK_FREQUENCY	主机控制器的时钟频率（单位为 MHz）。 PSoC 1 主器件的时钟频率为 24 MHz。
MACHINE_CYCLES	在软件延迟子程序中执行“while”循环时所需要的机器周期数量。 与 ImageCraft 编译器进行构建时，MACHINE_CYCLES 的值为 97（请参考 MBR_Delay）。

3.4.2.3 MBR_WriteBytes

此 API 初始化 CY8CMBR2110 器件和处理器之间的 I²C 写操作。该函数原型如 7.2.2 节中所述。

注意：对于写操作，在主机内定义了一个缓冲区。高层 API 将该缓冲区传送到写 API，该缓冲区的格式为一个 BYTE 阵列（请参考数据类型中的内容）。进行写操作时，第一个 BYTE（字节[0]）保存基本指针，剩下各字节（字节[1]、字节[2]…）保存数据。由于这个基本指针被设置为“CY8CMBR2110 的寄存器映射中将要写入的位置”，因此该写操作将从该位置开始。

高层 API 发送 I²C 缓冲区指针和将被写入的字节数量。 MBR_WriteBytes 将进行下列操作：

1. 对 CY8CMBR2110 器件初始化 I²C 写操作
2. 等待该数据操作结束
3. 检查该数据操作是否正常运行
4. 如果该数据操作没有正常运行，它将重新尝试宏 NACK_RETRY_LIMIT 的值进行写操作。

3.4.2.4 MBR_ReadBytes

此 API 初始化 CY8CMBR2110 器件和处理器之间的 I²C 写操作。该函数原型如 7.2.2 节中所述。

注意：由于字节[0]包含位置 0x00 中的数据，字节[1]包含位置 0x01 中的数据并依此类推，因此在进行读操作时，主机缓冲区将更新器件寄存器映射的位置 0x00 中的所需数据。读操作始终从所有寄存器映射的位置 0x00 开始。

高层 API 发送 I²C 缓冲区和将被读取的字节数量。MBR_ReadBytes 将进行下列操作：

1. 从器件内获取 I²C 缓冲区地址和将要读取的字节数量
2. 将从器件指针设置为指向位置 0x00
3. 对 CY8CMBR2110 器件初始化 I²C 读操作
4. 等待该数据操作结束
5. 检查该数据操作是否正常运行
6. 如果该数据操作没有正常运行，它将重新尝试使用宏 NACK_RETRY_LIMIT 的值进行读操作。

3.4.2.5 MBR_Delay

通过使用被执行指定次数的“while”循环，该 API 将生成一个软件延迟。该函数原型如 7.2.2 节中所述。 通过以下公式可以计算循环的重复次数：

$$number\ of\ loop\ iterations = \frac{\text{required delay time (ms)} \times \text{clock frequency (MHz)} \times 1000}{\text{machine cycles required to execute the while loop}}$$ 公式 5

您必须计算在主机中执行 while 循环时所需要的机器周期数量（汇编级指令周期的总和）。在使用 ImageCraft 专业版编译器的 PSoC 1 主机中，宏 MACHINE_CYCLES 的值为 97。您需要根据当前所使用的主机处理器和编译器修改此值。

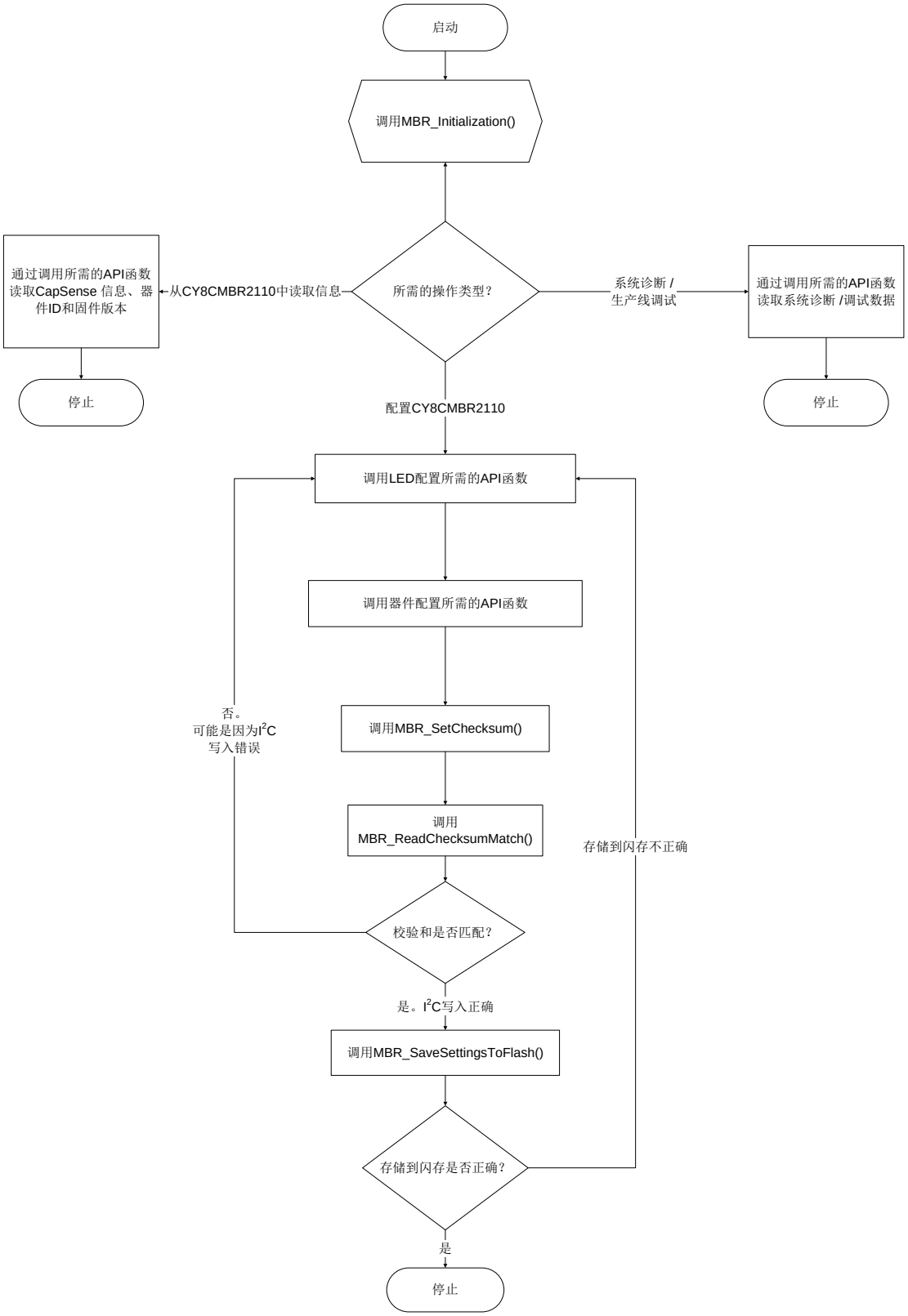
注意：在整个延迟时间内，CPU 完全被锁住。

3.4.2.6 配置 CY8CMBR2110 器件的指南

- 配置 CY8CMBR2110 器件时，必须按照某个特定的顺序调用高层 API。 说明了正确的顺序。
- 调用 MBR_Initialization API 后，请检查主机处理器中的 I²C 通信状态。调用任何其他 API 前，都需要调用此 API。例如：如果传输操作得到确认（ACK），低层 API 中的变量“gbI2CFlag”被设置为 1；否则，它将被设置为 0。您可以检查该变量以确保正确的传输操作。 另外，还可以检查主机处理器中的寄存器 I²C，以确定使用否认（NACK）还是确认（ACK）来表示。

- 在配置好一个模式下的所有特性前，请勿在各个寄存器映射模式间进行切换。例如：请勿先在 LED 配置模式下配置一个特性，然后切换器件配置模式，最后再返回 LED 配置模式配置其他特性。各个寄存器映射模式之间的切换很耗时。因此，先在 LED 配置模式下配置好所有特性，然后再切换到器件配置模式。
- 向高层 API 传递正确的参数，如用于 CY8CMBR2110 配置的 API 节中所定义。
- 由于高层 API 本身可计算配置的校验和，因此您不需要再计算它们了。
- 将 LED 配置和器件配置存储到闪存内后将引起一个软件复位，这样会清除主机控制 GPO 的配置，因此必须实现保存后才配置主机控制 GPO。
- LED 效果是按 GPO 组（如 GPO123、GPO456 和 GPO789；GPO0 除外）定义的。其配置必须与某一组中的所有 GPO 相匹配。 例如，请勿为 GPO1 和 GPO2 进行不同的 LED 配置。您配置 GPO1 后，该配置将适用于 GPO2 和 GPO3，因为它们共用了一个寄存器；另外如果您再次为 GPO2 重新配置了不同的 LED 效果，则该配置便应用于 GPO1 和 GPO3。
- 设置按键的手指阈值时，请通过使用 MBR_SetAdaptiveThreshold API（参考高层 API 一节）来清除或禁止自动阈值的特性。
- 当使用 LED 效果时，配置此效果的特性前需要使能此效果。例如：先使能按键触摸 LED 效果，然后配置所有相应的特性。
- 必须使用一节中的各步骤来调用深度睡眠 API。
- 请勿在配置 LED ON 时长的同时使能切换 ON/OFF。如果切换 ON/OFF 被使能，那么 LED ON 时长被禁止。
- 请勿在配置 LED ON 时间的同时使能按键触摸 LED 效果。如果按键触摸 LED 效果被使能，则 LED ON 时长被禁止。
- 请勿同时使能切换 ON/OFF 和最后按键 LED 效果。如果切换 ON/OFF 被使能，那么最后按键 LED 效果将被禁止。
- 可以直接调用所有读取 API（如系统诊断、传感器当前状态、传感器锁存状态、传感器信噪比和调试数据），而不需保存到闪存内。

图 3-34：高层 API 流程图



3.4.2.7 输入头文件

Inputs.h 包括高层 API 输入的宏定义。将各参数传递给高层 API 时需要使用这些宏。例如，当您使能某个特性时需要将 FEATURE_ENABLE 宏作为参数输入。某些高层 API 没有宏输入。例如，MBR_SetScanRate() API 没有任何宏定义作为输入，所以您需要输入 0 至 31 范围内的十进制数值作为函数参数。输入参数前，请参考用于 CY8CMBR2110 配置的 API 节中有关每个高层 API 的函数原型的内容。不要修改这些宏定义。这些宏有助于为高层 API 输入准确的参数。

注意：每个高层 API 的头文件都罗列了所有能够作为参数输入的宏。

3.4.2.8 数据类型

每个数据类型的内存大小取决于编译器。“char”、“int”和“long”等各种数据类型被类型定义，并由高层 API 用于配置 CY8CMBR2110 器件。 数据类型如下分类：

- 无符号的 char 数据类被定义为 BOOL
- 无符号的 char 数据类被定义为 BYTE
- 无符号的 int 数据类被定义为 WORD
- 无符号的 long 数据类被定义为 DWORD
- 有符号的 char 数据类被定义为 CHAR
- 有符号的 int 数据类被定义为 INT
- 有符号的 long 数据类被定义为 LONG

这些值均基于以下假定：char、int 和 long 数据类型分别占用存储器的 8、16 和 32 位。如果上述假定不符合您的主机编译器，请修改 Low_Level_API.h 和 High_Level_API.h 中的类型定义。

3.4.2.9 示例项目

创建一个示例项目，用以配置 CY8CMBR2110 器件（该器件将 CY8C29466-24PXI（PSoC）作为主机器件使用），且可通过 http://www.cypress.com/?rID=74590 下载该示例项目。该代码可由 CY3210-PSoC-EVAL 套件中的 PSoC Designer 版本 5.2 和 ImageCraft 编译器实现。示例代码配置了以下各特性：

1. 读取工作传感器的数量（即为通过系统诊断的有效传感器数量）
2. 在上升和下降时间的 600 ms 时间内，同时使能所有 GPO 上的上电 LED 效果，其中上升和下降时间为 600 ms。
3. 上电 LED 效果的高 l 亮度时间为 600 ms，其中：GPO0、GPO123 的亮度为 80%，GPO456 的亮度为 20%，GPO789 的亮度为 100%
4. 为 GPO0 设置上电 LED 效果的重复率为 1
5. 配置 AC 1 引脚蜂鸣器处于低闲置状态；蜂鸣器频率为 4 KHz，时长为 200 ms。
6. 将 CS0 按键的去抖动值设置为 100（连续按键接触时的响应时间为 1225 ms）
7. 将 CS0 按键的灵敏度值设置为 2（即中等级别）
8. 使能 CS0 按键的切换功能
9. 使能所有按键的 FSS 功能
10. 将主机所计算的校验和写入到 CY8CMBR2110 器件内
11. 验证校验和匹配的条件
12. 如果校验和匹配的条件为 true（真），将这些配置存储在闪存内
13. 将 HGPO1 状态设置为 HIGH（高电平状态）

注意：存储到闪存内后配置 HGPO1 为 HIGH（高电平状态）。在下一次复位时，它被清除为 LOW（低电平状态）。如果您想查看上电 LED 效果，必须为器件进行硬件复位，这样会清除 HGPO1。

3.4.3 第三方的编程器

要配置大量器件时，赛普拉斯建议使用第三方供应商的工具自动对器件进行编程。换句话说，您必须向希洛（Hilo）系统（即第三方的编程器）提供所配置的十六进制文件（该文件由 EZ-Click 定制器工具生成）。

请访问 http://www.hilosystems.com.tw/en/index.aspx 网址了解更详细的信息。

3.5 CY8CMBR2110 复位

通过使用硬件或软件，可以复位 CY8CMBR2110。

3.5.1 硬件复位

硬件复位时，可从闪存内将 LED 配置模式和器件配置模式的各个寄存器值加载到 RAM 内。初始化所有器件模块，执行了系统诊断，并传送一个最初 5 毫秒的脉冲到任何 GPOx 上（该 GPOx 与诊断失败的 CSx 相应）。如果抗噪能力被选为“Normal”，这个时间小于 350 毫秒；如果抗噪能力被选为“High”，则该时间小于 1000 毫秒。如果使能了上电 LED 效果，则可在剩余的所有 GPO 上看到此效果。LED 效果发生后，器件将处于操作模式，并开始正常的工作。

使用电源电压或 XRES 来切换 CY8CMBR2110 引脚上的电源，从而实现硬件复位。

3.5.1.1 电源复位

为了进行电源复位，关闭提供给设备的 V_{DD} 线的外部电源，（要保证此 V_{DD} 下降低于 100 mV），然后再打开电源。 复位电源时，可在 HostControlGPO1 引脚上看到变为高电平的 16 毫秒的脉冲。

3.5.1.2 XRES 复位

为了进行 XRES 复位，先上拉器件的 XRES 引脚为高电平（HIGH），然后再将它下拉为低电平（LOW）。XRES 复位时，在 HostControlGPO1 引脚上看不到脉冲。

3.5.2 软件复位

通过将 1 写入到 Host_Mode 寄存器中的“软件复位”位（在操作模式下），可以复位软件。软件复位时，可从闪存内将 LED 配置模式和器件配置模式的各个寄存器值加载到 RAM 内。器件将自动清除“软件复位”位，并且所有器件模块均被初始化。如果抗噪能力被选为“Normal”，则该时长小于 350 毫秒；如果抗噪能力被选为“High”，则该时长小于 1000 毫秒。器件处于操作模式，并开始正常工作。没有执行任何系统诊断操作，且上电 LED 效果也不会发生。 如果用户已经将器件配置为上电 LED 效果，并将此设置存储到闪存内，则必须进行硬件复位才能看到上电 LED 效果。

4. 电气和机械设计中的注意事项

在您的应用中设计电容式触摸感应技术时，一定要将 CapSense 器件配置在较大规模的框架中，这一点至关重要。要认真考虑到所有的细节，包括 PCB 布局、用户界面以及最终用户的工作环境，以实现强劲可靠的系统性能。更多详细信息，请参见《CapSense 入门手册》中的内容。

4.1 覆盖层选择

在 CapSense 原理图设计一节中，公式 1 介绍的是手指容值：

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

其中：

ε_0 = 空气介电常数

ε_r = 外覆层的绝缘常数

A = 手指与按键板外覆层的接触面积

D = 外覆层的厚度

想提高 CapSense 的信号强度，请选择介电常数较高的外覆层材料，降低外覆层的厚度，并增加按键的直径。设计工具箱有助于设计一个既稳定又可靠的 CY8CMBR2110 解决方案，如 CapSense 原理图设计章节中所述。

表 4-1. 覆盖层材料绝缘强度

材料	击穿电压（V/mm）	电压为 12 kV 条件下的最小外覆层厚度（mm）
空气	1200–2800	10
干木材	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
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

由于导电材料会干扰电场图形，因此不能将它用作外覆层。 因此，请勿使用含有金属颗粒的油漆。

将覆盖层粘接至 PCB

因为空气的介电常数很低，外覆层和按键之间的气隙会降低按键的性能。为了消除气隙，请使用不导电的粘合剂将外覆层粘贴到 CapSense PCB 上。名称为 200 MP 的 3M™透明丙烯酸酯胶膜可用于 CapSense 应用。这种粘合膜是从纸背的胶带卷上抽取出来（3M™ 产品号为 467 MP 和 468 MP）的。

4.2 ESD 保护

考虑周全的系统设计自然会具有强大的抗 ESD 能力。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 的放电事件不会对 CapSense 控制器造成任何损坏。

CapSense 控制器引脚可以承受 12 kV 的直接放电事件。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 4-1 列出了用于保护 CapSense 按键免受 12 kV 放电冲击（如 IEC 61000-4-2 中所述）所需的各种外覆层材料的厚度。如果外覆层材料未提供充分的 ESD 保护，则按照以下的顺序采取防静电措施：预防、重定向并钳制

4.2.1 预防

确保触摸表面上的所有路径的击穿电压都大于潜在的高电压触点。此外，设计系统时，使 CapSense 控制器和可能的 ESD 源之间保持合适的距离。 如果无法保持合适的距离，请在 ESD 源与 CapSense 控制器之间放置一个耐高电压击穿的材料作为保护层。 一层 5 密耳厚的 Kapton®胶带可承受 18 kV 的电压。

4.2.2 重定向

如果您的产品空间密集，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的位置来保护 CapSense 控制器。在连接到底盘接地的电路板周界上放置一个保护环。正如 PCB 布局指导方针中的建议，在按键或滑块周围使用一个网格地层可以重新定向 ESD 事件，使其远离按键和 CapSense 控制器。

4.2.3 钳制

因为有意将 CapSense 按键放置在触摸表面附近，所以重新定向路径的方法可能并不实用。在这种情况下，要考虑添加串联电阻或专用的 ESD 保护器件。

建议系列电容值为 560 Ω。

更有效的方法是在易受影响的走线上放置专用 ESD 保护器件。请注意：用于 CapSense 的 ESD 保护器件必须是低电容的。表 4-2 列出了可用于 CapSense 控制器的推荐组件。

表 4-2. 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 最大限制	空气放电 最大限制
制造商	器件型号				
Littelfuse	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

4.3 电磁兼容性（EMC）注意事项

4.3.1 辐射干扰

辐射电能可能会影响系统的测量以及处理器内核的运行。这种干扰会通过 CapSense 按键的走线和任何其他数字或模拟输入进入 CY8CMBR2010 芯片的 PCB 层。最大限度减小射频干扰作用的布局指南如下：

- **接地层**：在 PCB 上提供一个接地层。
- **串联电阻**：在 CapSense 控制器引脚的 10 mm 内安装串联电阻。
  - 建议 CapSense 输入线的串联电阻值为 560 Ω。
- **走线长度**：请尽可能缩短走线长度。
- **电流环路区域**：最小化电流的回路路径。为减小寄生电容的影响，要在按键和走线的 1 cm 空间内提供网格接地，而不是用实体填充。
- **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。屏蔽电源也是另一种防止干扰的通用技术。

4.3.2 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。由于 CapSense 控制器是低功耗器件，因此您必须避免传导辐射。下列各指南有助于降低传导辐射和干扰：

- 按照数据手册中的建议使用去耦电容器。
- 在连至系统电源的输入端上添加双向滤波器。该滤波器对传导辐射和干扰均有效。“Pi”滤波器可以防止电源噪声影响敏感器件，同时还可防止器件的开关噪声反耦合到电源层。
- 如果 CapSense 控制器 PCB 通过电缆连接至电源，请尽量减小电缆的长度，并考虑使用屏蔽电缆。
- 要想过滤出高频噪声，请在电源或通信线的周围放置一个铁氧体磁珠。

4.4 PCB 布局指导方针

设计工具箱有助于设计一个稳定的 CY8CMBR2110 CapSense PCB 布局，如通用布局指导中所述。

如果您设计使用 GPO 将电流输入到 CapSense 控制器，并且在 CapSense 系统中存在大量噪声，那么在所有 GPO 上使用的串联电阻用于限制灌电流。在您的设计中，工作电压为 5 V 时，灌电流限制取决于最大的按键 C_P，如表 4-3 所示。

表 4-3. GPO 低输出电压灌电流限制

按键 C_P 的取值范围	每个 GPO 的灌电流限制	每个器件的灌电流限制
5 pF ≤ C_P ≤ 12 pF	25 mA	120 mA
12 pF ≤ C_P ≤ 21 pF	20 mA	20 mA
21 pF ≤ C_P ≤ 40 pF	6 mA	6 mA

《CapSense 入门手册》中提供了详细的 PCB 布局指南。

5. 低功耗设计中的注意事项

5.1 系统设计建议

赛普拉斯 CY8CMBR2110 的设计可满足电池供电应用中的低功耗要求。

按照下面步骤最小化功耗：

- 将所有未使用的 CapSense 输入接地
- 按照《CapSense 入门手册》中的设计指南最小化 C_P。
- 降低供电电压
- 要降低 CSx 按键的灵敏度，请参考灵敏度的控制一节中介绍的内容。
- 针对优化功耗的要求配置设计时，请参见按键扫描速率节中的内容。
- 只在需要时才使用“High”抗噪等级，请参考抗噪能力节中的内容。
- 使用较高的按键扫描速率或深度睡眠操作模式，请参考按键扫描速率一节中的内容。

5.2 计算平均功耗

设计工具箱自动计算本节中所描述的电源优化。通过计算下面各参数可以确定 CY8CMBR2110 的平均功耗：

- 按键扫描速率，T_R
- 扫描时间，T_S
- NO TOUCH（无触摸）状态下的平均电流，I_{AVE_NT}
- TOUCH（接触）状态下的平均电流，I_{AVE_T}
- 工作时间的比例，P
- 平均使用电流，I_{AVE_U}
- 平均电流，I_{AVE}
- 平均功耗，P_{AVE}

5.2.1 按键扫描速率 （T_R）

在 CY8CMBR2110，您可通过设置寄存器映射来控制按键扫描速率。根据寄存器值，获取一个偏移量，并将其加上一个常量，以获得实际的按键扫描速率。偏移量的取值范围为 0 至 506 毫秒。

$$T_R = Button\ Scan\ Rate\ Constant + Button\ Scan\ Rate\ offset$$ 公式 6

表 3-5 显示的是确定按键扫描速率常数的方法。

5.2.1.1 响应时间

响应时间是按下按键 CSx 并且被器件视为有效按键触摸的最短时间。这时，器件将在 GPOx 上产生一个信号。

可以使用下面的公式计算响应时间：

公式 7

如果抗噪能力为“Normal”：

$$RT_{CBT} = Button\ Scan\ Rate\ constant + \left[Button\ Scan\ Rate\ constant \times \left\{Round_{down}\left((Debounce - 1)/3\right) + 1\right\}\right]$$

$$RT_{FBT} = Button\ Scan\ Rate + \left[Button\ Scan\ Rate\ constant \times \left\{Round_{down}\left((Debounce - 1)/3\right) + 1\right\}\right]$$

如果抗噪能力为“High”：

$$RT_{CBT} = Button\ Scan\ Rate\ constant + \left[Button\ Scan\ Rate\ constant \times Debounce\right]$$

$$RT_{FBT} = Button\ Scan\ Rate + \left[Button\ Scan\ Rate\ constant \times Debounce\right]$$

其中：

RT_{CBT} = 第一个按键触摸随后的按键触摸的响应时间

RT_{FBT} = 第一个按键触摸的响应时间

CS1 至 CS9 的去抖动取值范围为（1 - 255）

CS0 的去抖动取值范围为（1 - 255）

$Round_{down}$ 是一个不超过((去抖动值 – 1)/3)的最大整数值

如果想将抗噪能力从“Normal”改为“High”，请减小去抖动值来保持响应时间。

5.2.2 扫描时间（TS）

可以使用以下公式来计算近似扫描时间：

公式 8

当抗噪能力为“Normal”时：

$$T_S = [0.375\ ms \times (K_{CS0} + K_{CS1} + K_{CS2} + \cdots + K_{CS9})] + T_{FW}$$

当抗噪能力为“High”时：

$$T_S = [0.375\ ms \times (K_{CS0} + K_{CS1} + K_{CS2} + \cdots + K_{CS9}) \times 3] + T_{FW}$$

其中：

K_{CSX} = CSx 的按键灵敏度常量，来自表 5-1。

T_{FW} = 固件执行时间，如表 5-2 所示。

表 5-1. 按键灵敏度常量

CSx 灵敏度（pF）	C_P（pF）[7]	按键灵敏度常量（K）
高	按键接触 GND	0
	5 pF ≤ C_P ≤ 10 pF	1
	10 pF < C_P ≤ 22 pF	2
	22 pF < C_P ≤ 40 pF	4
中	按键接触 GND	0
	5 pF ≤ C_P ≤ 18 pF	1
	18 pF < C_P ≤ 38 pF	2
	38 pF < C_P ≤ 40 pF	4
低	按键接触 GND	0
	5 pF ≤ C_P ≤ 12 pF	0.5
	12 pF < C_P ≤ 26 pF	1
	26 pF < C_P ≤ 40 pF	2

表 5-2. 平均电流参数

参数	典型值	最大值
T_{FW}	6.00 ms	6.50 ms
T_S	如公式 7 所示	典型值加上 5%
T_R	如公式 5 所示	典型值加上 10%
I_{SLEEP}	9.52 µA	14.2 µA
I_{ACTIVE}	3.4 mA	4.00 mA

[7] C_P 限制是近似的，且其变化幅度为±2 pF。

5.2.3 NO TOUCH（无触摸）状态下的平均电流（I_{AVE_NT}）

$$I_{AVE_NT} = \left(\frac{T_R - T_S}{T_R} \times I_{SLEEP}\right) + \left(\frac{T_S}{T_R} \times I_{ACTIVE}\right)$$ 公式 9

其中：

T_R = 按键扫描速率

T_S = 扫描时间

I_{SLEEP} = CY8CMBR2110 在低功耗睡眠模式期间所消耗的电流，如表 5-2 所示。

I_{ACTIVE} = CY8CMBR2110 在有效工作期间所消耗的电流，来自表 5-2。

如果待机模式下的 LED 亮度被使能：

$$I_{AVE_NT} = I_{ACTIVE}$$

5.2.4 TOUCH（触摸）状态下的平均电流（I_{AVE_T}）

$$I_{AVE_T} = \left(\frac{C_{BS} - T_S}{C_{BS}} \times I_{SLEEP}\right) + \left(\frac{T_S}{C_{BS}} \times I_{ACTIVE}\right)$$ 公式 10

其中：

T_S = 扫描时间

C_{BS} = 按键扫描速率常量，来自表 3-5。

I_{SLEEP} = CY8CMBR2110 在低功耗睡眠模式期间所消耗的电流，如表 5-2 所示。

I_{ACTIVE} = CY8CMBR2110 在有效工作期间所消耗的电流，来自表 5-2。

如果待机模式下的 LED 亮度被使能：

$$I_{AVE_T} = I_{ACTIVE}$$

5.2.5 工作时间的比例（P）

触摸某个按键时，通过将每小时的按键触摸数量乘以以下三个值中的最大值，可以计算出器件的工作时间（单位为 ms）：

1. 平均按键触摸时间
2. 蜂鸣器为 ON 的平均时间
3. 按键触摸 LED 效果的平均时间

公式 11

$$Active\ time\ = Max(Button\ touch\ time, Buzzer\ ON\ time, Button\ Touch\ LED\ Effects\ time) \times (Number\ of\ button\ touches\ per\ hour)$$

工作时间的比例为：

$$P = \frac{Active\ time}{(3600 \times 1000)} \times 100$$ 公式 12

使用该方式计算 P，条件为蜂鸣器信号输出或按键触摸 LED 效果结束后才触摸按键并且没有触摸任何其他按键。否则，使用该 P 值会使计算的功耗超出实际值。

5.2.6 平均使用电流（IAVE_U）

$$I_{AVE_U} = \left(\frac{100-P}{100} \times I_{AVE_NT}\right) + \left(\frac{P}{100} \times I_{AVE_T}\right)$$ 公式 13

其中：

P = 工作时间的比例

IAVG_NT = NO TOUCH（无触摸）状态下的平均电流

IAVG_T = 处于 TOUCH（触摸）状态时的平均电流

5.2.7 平均电流（IAVE）

$$I_{AVE} = \left[I_{AVE_U} \times \left(\frac{T_{SA}}{T_{DS}+T_{SA}}\right)\right] + 0.1\ \mu A$$ 公式 14

其中：

TSA = 器件并非处于深度睡眠模式下的时间

TDS= 器件处于深度睡眠模式的时间

5.2.8 平均功耗（PAVE）

$$P_{AVE} = V_{DD} \times I_{AVE}$$ 公式 15

其中：

IAVE = 平均电流

VDD = 电源电压

5.2.9 示例计算

在如何计算平均功耗的示例中，考虑带有八个设计良好的按键的 CapSense 用户界面以及下述各参数：

- 8 个按键的 CP 值为 10 到 20 pF
- 每个按键的灵敏度为“High”
- 设计中的响应时间被优化
- 抗噪能力为“Normal”
- 将按键扫描速率偏移量设为 506 ms
- 禁止待机模式下的 LED 亮度
- 测量了典型电流消耗值

按键扫描速率常量可从表 3-5 获得：

$$C_{BS} = 35\ ms$$

按键扫描速率可通过公式 5 计算得到：

$$T_R = 35 + 506 = 541\ ms$$

扫描时间可通过公式 7 计算得到，其中按键灵敏度常量可从表 5-1 获得，另外固件执行时间的典型值可从表 5-2 获得。

$$T_S = [0.375 \times (8 \times 2)] + 6.00 = 12.0\ ms$$

使用公式 8 以及表 5-2 中 ISLEEP 和 IACTIVE 的最大值计算 NO TOUCH（无触摸）状态下的平均电流：

$$I_{AVE_NT} = \left(\frac{541-12}{541} \times 9.52\ \mu A\right) + \left(\frac{12}{541} \times 3.4\ mA\right) = 84.7\ \mu A$$

按照如下所示，使用公式 9 计算 TOUCH（触摸）状态下的平均电流：

$$I_{AVE_T} = \left(\frac{35-12}{35} \times 9.52\ \mu A\right) + \left(\frac{12}{35} \times 3.4\ mA\right) = 1172\ \mu A$$

使用公式 10 计算工作时间，假设每分钟触摸按键一次（即每小时触摸按键 60 次）。平均计算，按键触摸时间为 1000 毫秒，按键触摸 LED 效果时间为 3000 毫秒，并且没有蜂鸣器输出。

$$Active\ time\ = 3000ms \times 60 = 180\ s$$

通过使用公式 11，可计算出工作时间的比例：

$$P = \frac{180}{3600} \times 100 = 5\%$$

使用公式 12 如下计算设计的平均电流消耗：

$$I_{AVE_U} = \left(\frac{100-5}{100} \times 84.7\ \mu A\right) + \left(\frac{5}{100} \times 1172\ \mu A\right) = 139.1\ \mu A$$

假设此设计没有使用深度睡眠模式并且它在 1.71 V 下工作，则使用公式 14 来计算平均功耗，如下所述：

$$P_{AVE} = 1.71 \times 139.1\ \mu A = 237.8\ \mu W$$

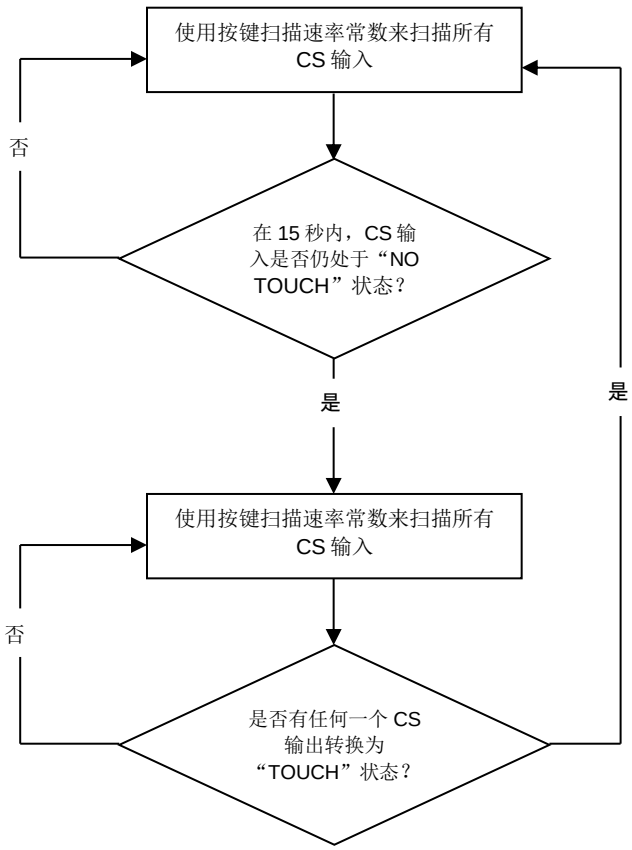
5.3 睡眠模式

赛普拉斯的 CY8CMBR2110 可配置为在低功耗睡眠或深度睡眠模式下工作。这些模式可以降低器件的功耗。

5.3.1 低功耗睡眠模式

表 5-1 中介绍的是 CY8CMBR2110 控制器在低功耗睡眠模式下的行为。

图 5-1. 低功耗睡眠模式



5.3.2 深度睡眠模式

如果 CY8CMBR2110 位于具有主机处理器的系统中，则使用“注意/睡眠”线可以使器件在深度睡眠模式下工作。为了使 CY8CMBR2110 进入深度睡眠模式，请执行下列步骤：

1. 下拉注意/睡眠线到低电平。
2. 在操作模式下，将 Host_Mode 寄存器上的 Deep Sleep（深度睡眠）位设置为 1
3. 等待 50 毫秒
4. 上拉注意/睡眠线到高电平。

暂停所有通信。在深度睡眠模式下，器件消耗大约 0.1 µA 的电流。器件进入深度睡眠模式后，会自动清除深度睡眠位。为从深度睡眠模式唤醒，主机会将“注意/睡眠”线置于低电平状态。器件唤醒后，CY8CMBR2110 将进入工作模式。然后，主机处理器可将“注意/睡眠”引脚置于高电平状态，以使器件处于低功耗睡眠模式。从深度睡眠模式唤醒后到按键恢复扫描前，器件将占用一段时间，这段时间被称为重新初始化。在这段时间内，不会报告任何按键触摸。如果抗噪能力为“Normal”，重新初始化时间为 20 ms；如果抗噪能力为“High”，则该时间为 50 ms。

6. 资源

6.1 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。

在 CY8CMBR2110 网页上可以找到各种技术资源。

6.2 数据手册

www.cypress.com 上提供了 CapSense CY8CMBR2110 器件的数据手册。

- CY8CMBR2110

6.3 设计工具箱

该交互设计工具箱允许您设计一个稳定可靠的 CY8CMBR2110 CapSense 解决方案。

6.4 EZ-Click™定制工具

交互 EZ-Click 定制工具有助于配置您的 CY8CMBR2110 CapSense 解决方案。

6.5 设计支持

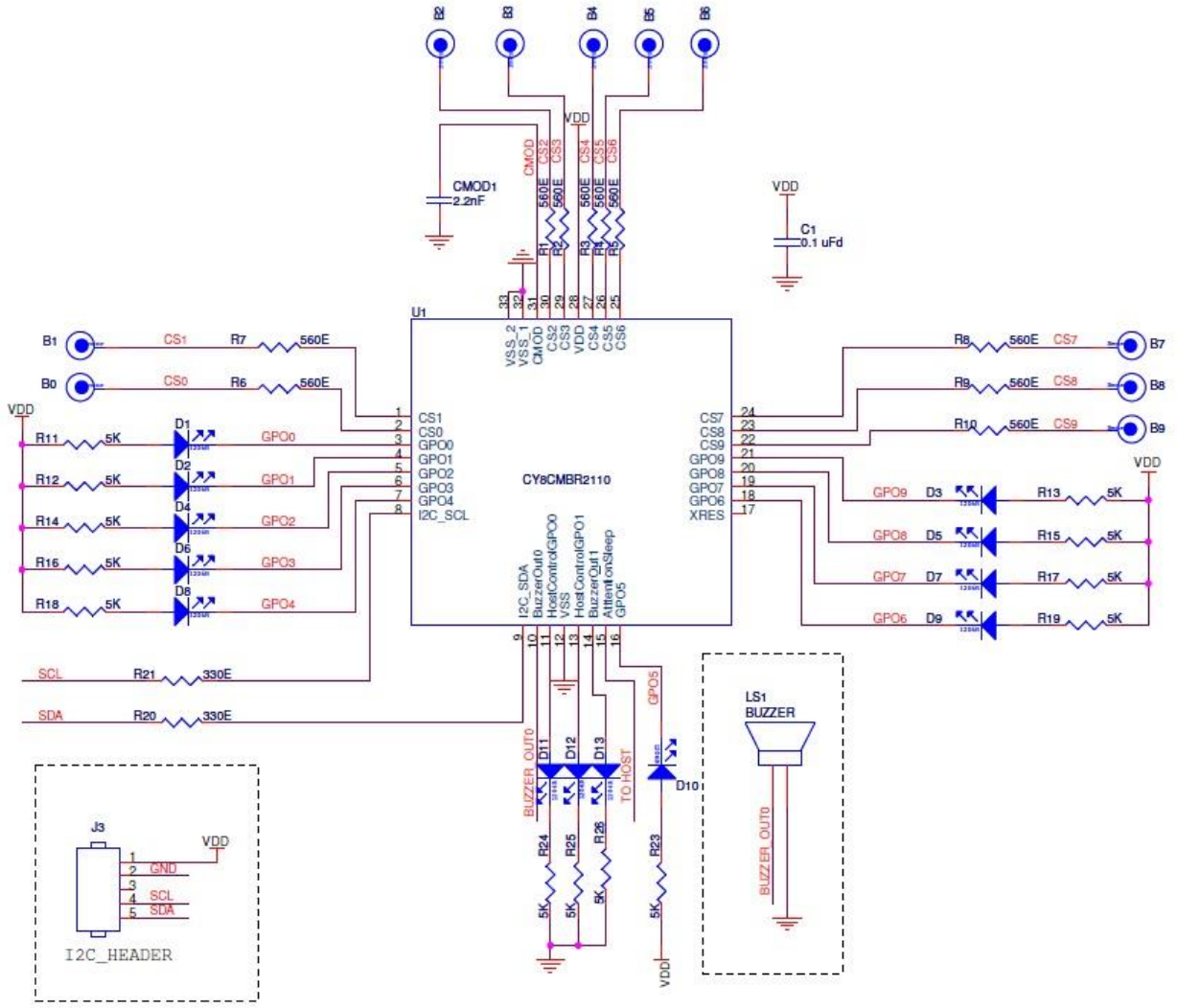
为了确保 CapSense 解决方案的成功，赛普拉斯具有各种设计支持渠道。

- 知识库文章 — 按产品系列浏览技术文章或对各种 CapSense 主题执行搜索。
- CapSense 应用手册 — 以本文档提供的信息为基础的各种应用手册。
- 白皮书 — 了解高级电容式触摸接口主题。
- 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 产品选择指南 — 请参见完整的 CapSense 产品线。
- 视频资料库 — 使用视频教程提高学习速度
- 质量和可靠性 — 赛普拉斯致力于实现客户满意度。在我们的“质量”网站，您可以找到可靠性和产品资质报告。
- 技术支持 — 世界级在线技术支持。

7. 附录

7.1 原理图示例

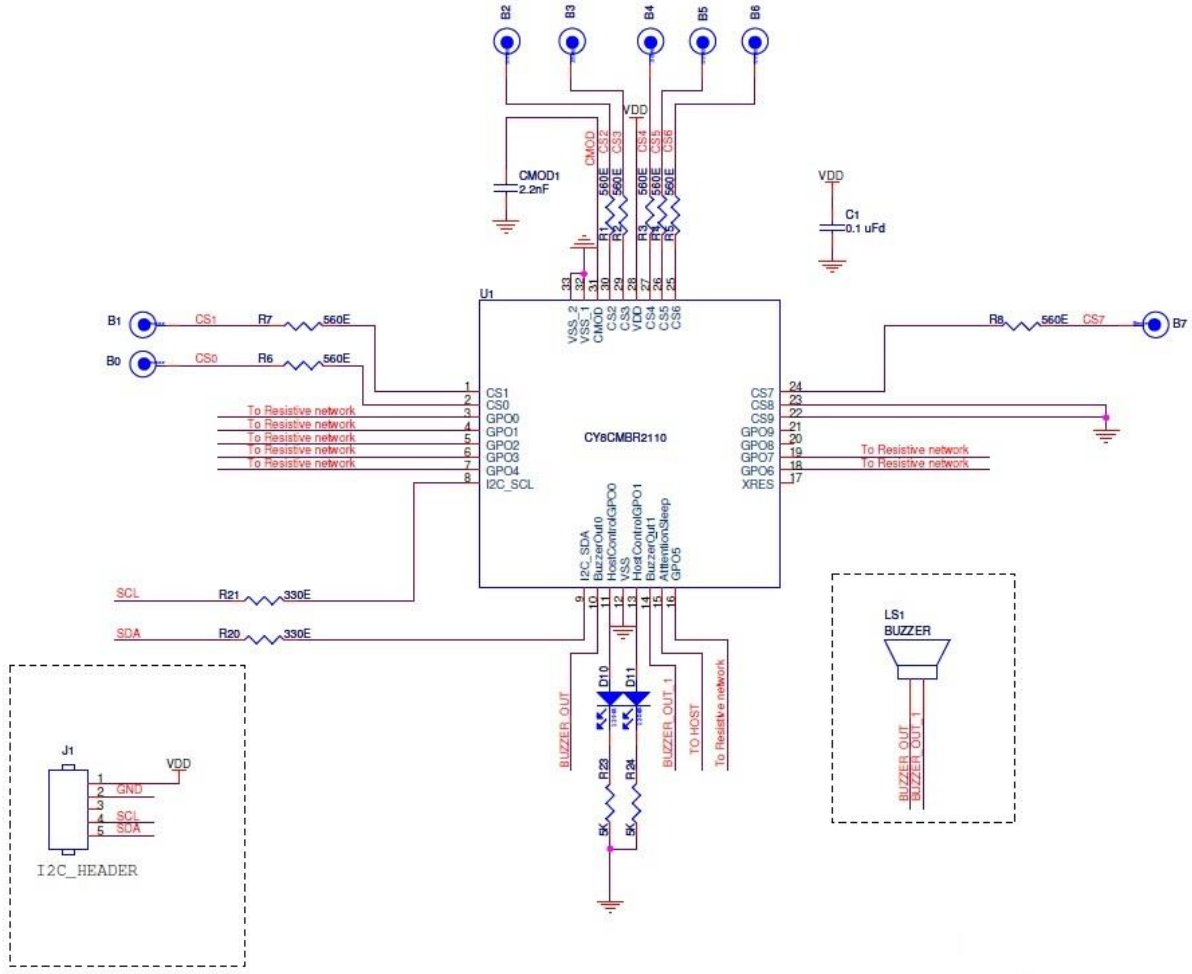
7.1.1 原理图 1：十个按键和十个 GPO



在原理图 1：十个按键和十个 GPO 中，CY8CMBR2110 的配置如下：

- CS0—CS9 引脚：通过 560 Ω 的电阻连接 CapSense 按键
  - 十个 CapSense 按键（CS0 - CS9）
- GPO0—GPO9 引脚：LED 和 5 kΩ 的电阻连接到 V_{DD}
  - CapSense 按键驱动十个 LED（GPO0 - GPO9）
- CMOD 引脚： 通过 2.2 nF 的电容接地
  - 调制电容
- XRES 引脚：悬空
  - 用于外部复位
- BuzzerOut0 引脚：连接至蜂鸣器
  - AC 蜂鸣器（单引脚）
  - 蜂鸣器秒引脚接地
- BuzzerOut1 引脚：通过 LED 和 5 kΩ 的电阻接地
  - 作为主机控制的 GPO 使用
- HostControlGPO0、HostControlGPO1：通过 LED 和 5 kΩ 的电阻接地
  - 两个主机控制的 GPO
- I2C_SDA、I2C_SCL 引脚：通过 330 Ω 的电阻与 I^2C 插槽连接
  - 用于 I^2C 通信
- 提醒/睡眠引脚：连接至主机
  - 控制 I^2C 通信、功耗以及器件工作模式

7.1.2 原理图 2：八个按键和模拟电压输出



在原理图 2：八个按键和模拟电压输出中，CY8CMBR2110 的配置如下：

- CS0 – CS7 引脚：通过 560 Ω 的电阻连接 CapSense 按键；CS8、CS9 引脚接地
  - 八个 CapSense 按键（CS0 – CS7）
  - 在设计中没有使用 CS8 和 CS9 按键
- GPO0 – GPO7 引脚：用于连接外部电阻网络
  - 八个 GPO（GPO0 – GPO7）用于模拟电压输出
  - 在设计上不使用 GPO8 和 GPO9
- CMOD 引脚：通过 2.2 nF 的电容接地
  - 调制电容
- XRES 引脚：悬空
  - 用于外部复位
- BuzzerOut0、BuzzerOut1 引脚：连接到交流蜂鸣器
  - 双引脚交流蜂鸣器

- HostControlGPO0、HostControlGPO1 引脚：通过 LED 和大小为 5 kΩ 的电阻接地
  - 两个主机控制的 GPO
- I2C_SDA、I2C_SCL 引脚：通过 330 Ω 的电阻与 I²C 插槽连接
  - 用于 I²C 通信
- 提醒/睡眠引脚：连接至主机
  - 控制 I²C 通信、功耗和器件工作模式

7.2 用于 CY8CMBR2110 配置的 API

下表列出了 72 个高层 API 和 3 个低层 API，主机处理器（I²C 主控）会使用这些 API 通过 I²C 接口配置 CY8CMBR2110（I²C 从器件）。这些高层 API 独立于平台，并可以在任何主机处理器上使用。在 inputs.h 文件内，相关的输入被定义为多个高层 API 的宏。

低层 API 与平台相关联，并作用于主机处理器以启用与器件进行的物理 I²C 通信。这些低层 API 是针对 PSoC 1 主机器件开发的；因此，您可能需要根据所使用的主机处理来修改低层 API 代码。3.4.2.9 部分说明了使用这些 API 来创建示例项目。

7.2.1 高层 API

	原型	void MBR_Initialization(void);		
1	说明	初始化高层 API 所使用的全局变量。 调用任何其他 API 之前，您必须先调用该 API。		
	参数	无		
	返回值	无		
	示例	MBR_Initialization();		
2	原型	void MBR_SetCustomData(BYTE bCustomData);		
	说明	在器件配置模式下，将用户提供的数据写入到自定义数据存储寄存器内。 用户应该调用 **MBR_SaveSettingsToFlash** API 以永久存储数据。		
	参数	名称	说明	可能值
		bCustomData	需要写入自定义寄存器中的数据。	0 到 255
	返回值	无		
	示例	MBR_SetCustomData(200); 数值 200 被存储在自定义数据存储寄存器中。		
3	原型	void MBR_IssueSWReset(void);		
	说明	对 CY8CMBR2110 器件进行软件复位。 请参考软件复位部分内容		
	参数	无		
	返回值	无		
	示例	MBR_IssueSWReset();		
4	原型	WORD MBR_ReadFlashChecksum(void);		
	说明	读取存储在 CY8CMBR2110 器件上闪存内的校验和。		
	参数	无		
	返回值	CY8CMBR2110 器件的闪存校验和		
	示例	MBR_ReadFlashChecksum();		
5	原型	WORD MBR_ReadRAMChecksum(void);		
	说明	读取存储在 CY8CMBR2110 器件上 RAM 内的校验和。		
	参数	无		
	返回值	CY8CMBR2110 器件的 RAM 校验和		
	示例	MBR_ReadRAMChecksum();		

6	**原型**	**void MBR_SetChecksum(void);**		
	说明	将主机计算的校验和写入 CY8CMBR2110 器件内。 主机自己计算所配置的校验和		
	参数	无		
	返回值	无		
	示例	MBR_SetChecksum();		
7	**原型**	**BYTE MBR_ReadChecksumMatch(void);**		
	说明	检查 CY8CMBR2110 器件计算的 RAM 校验和是否与主机输入的校验和相同。		
	参数	无		
	返回值	0 或 1 如果校验和不匹配，返回值为 0 如果校验和匹配，返回值为 1		
	示例	MBR_ReadChecksumMatch();		
8	**原型**	**BYTE MBR_SaveSettingsToFlash(void);**		
	说明	将 CY8CMBR2110 器件的当前配置保存到闪存（请参考配置 CY8CMBR2110）内		
	参数	无		
	返回值	0 或 1 0 — 未保存到闪存内 1 — 已保存到闪存内		
	示例	MBR_SaveSettingsToFlash();		
9	**原型**	**BYTE MBR_SettingsLoaded(void);**		
	说明	指示加载的是出厂默认设置还是用户配置设置		
	参数	无		
	返回值	0 或 1 0 — 用户配置设置 1 — 出厂默认设置		
	示例	MBR_SettingsLoaded( );		
10	**原型**	**void MBR_LoadFactoryDefaults(void);**		
	说明	将出厂默认设置配置加载到CY8CMBR2110器件的RAM内。		
	参数	无		
	返回值	无		
	示例	MBR_LoadFactoryDefaults();		
11	**原型**	**void MBR_ReadConfigData(BYTE abConfigData[]);**		
	说明	从 CY8CMBR2110 器件加载 LED 配置和器件配置数据。		
	参数	**名称**	**说明**	**可能值**
		abConfigData	指向包含所有配置数据的 64 字节阵列的指针	
	返回值	无		
	示例	MBR_ReadConfigData(abConfigData); abConfigData 是指向 64 字节阵列 abConfigData[64]的指针。使用所有配置数据对阵列进行更新。		

12	**原型**	**void MBR_LEDEffectsBreathing(BYTE bGPO, BYTE bBreath);**		
	说明	启用或禁用按键触摸LED效果闪烁。 **注意**：对于LED效果，GPO被分组为GPO0、GPO123、GPO456以及GPO789。在同一组中，配置一个GPO也便配置了其他GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bBreath	启用/禁用效果	0 或 1 0 — 禁用闪烁 1 — 启用闪烁
	返回值	无		
	示例	MBR_LEDEffectsBreathing(GPO4, FEATURE_ENABLE); GPO4 和 FEATURE_ENABLE 均为宏，它门的值分别为 4 和 1（inputs.h）。		
13	**原型**	**void MBR_LEDEffectsRepeatRate(BYTE bGPO, BYTE bRepeatRate, BYTE bPwrOnOrBtnTch);**		
	说明	对选定的 GPO 进行设置 LED 效果的重复频率。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bRepeatRate	LED 效果的重复频率	0 至 7 0 — LED 效果的重复频率为 0 1 — LED 效果的重复频率为 1 2 — LED 效果的重复频率为 2 3 — LED 效果的重复频率为 4 4 — LED 效果的重复频率为 6 5 — LED 效果的重复频率为 10 6 — LED 效果的重复频率为 15 7 — LED 效果的重复频率为 20
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsRepeatRate(GPO1、REPEAT_RATE_20、POWER_ON_LED_EFFECTS); GPO1、REPEAT_RATE_20 和 POWER_ON_LED_EFFECTS 均是宏，它们的值分别为 1、7、1（inputs.h）。		

14	**原型**	**void MBR_LEDEffectsLowBrightness(BYTE bGPO, BYTE bLowBright, BYTE bPwrOnOrBtnTch);**		
	说明	设置 GPO 的 LED 低亮度。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bLowBright	每个寄存器映射的低亮度级别	0 至 7 0 — 低亮度为 0% 1 — 低亮度为 10% …… ……... 7 — 低亮度为 100%
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsLowBrightness(GPO2、LOW_BRIGHT_80、BTN_TOUCH_LED_EFFECTS); GPO2、LOW_BRIGHT_80、BTN_TOUCH_LED_EFFECTS 是宏，它们的值分别为 2、6、2（inputs.h）。		
15	**原型**	**void MBR_LEDEffectsHighBrightness(BYTE bGPO, BYTE bHighBright, BYTE bPwrOnOrBtnTch);**		
	说明	设置 GPO 的 LED 高亮度。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bHighBright	每个寄存器映射的高亮度级别	0 至 7 0 — 高亮度为 100% 1 — 高亮度为 90% …… ……... 7 — 高亮度为 0%
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsHighBrightness(GPO9、HIGH_BRIGHT_50、BTN_TOUCH_LED_EFFECTS); GPO9、 HIGH_BRIGHT_50、BTN_TOUCH_LED_EFFECTS 是宏，它们的值分别为 9、4、2（inputs.h）。		

16	**原型**	**void MBR_LEDEffectsLowTime(BYTE bGPO, BYTE bLowTime, BYTE bPwrOnOrBtnTch);**		
	说明	设置 GPO 的 LED 低亮度时长。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bLowTime	确定低亮度时长时需要的全局周期寄存器映射图	0 或 1 0 — GLOBAL_PERIOD_1 1 — GLOBAL_PERIOD_2
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsLowTime(GPO6、GLOBAL_PERIOD_1、BTN_TOUCH_LED_EFFECTS); GPO6、GLOBAL_PERIOD_1、BTN_TOUCH_LED_EFFECTS 是宏，它们的值分别为 6、0、2（inputs.h）。		
17	**原型**	**void MBR_LEDEffectsHighTime(BYTE bGPO, BYTE bHighTime, BYTE bPwrOnOrBtnTch);**		
	说明	设置 GPO 的 LED 高亮度时长。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bHighTime	确定高亮度时长时需要的全局周期寄存器映射图	0 或 1 0 — GLOBAL_PERIOD_1 1 — GLOBAL_PERIOD_2
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsHighTime(GPO5, GLOBAL_PERIOD_2, BTN_TOUCH_LED_EFFECTS); GPO5、GLOBAL_PERIOD_2、BTN_TOUCH_LED_EFFECTS 是宏，它们的值分别为 5、1、2（inputs.h）。		

18	**原型**	**void MBR_LEDEffectsRampDown(BYTE bGPO, BYTE bRampDown, BYTE bPwrOnOrBtnTch);**		
	说明	设置 GPO 的下降时长。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bRampDown	用于确定下降时长值的全局周期寄存器映射图	0 至 3 0 — GLOBAL_PERIOD_1 1 — GLOBAL_PERIOD_2 2 — GLOBAL_PERIOD_3 3 — GLOBAL_PERIOD_4
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsRampDown(GPO8、GLOBAL_PERIOD_1、POWER_ON_LED_EFFECTS); GPO8、GLOBAL_PERIOD_1、POWER_ON_LED_EFFECTS 是宏，它们的值分别为 8、0、1（inputs.h）。		
19	**原型**	**void MBR_LEDEffectsRampUp(BYTE bGPO、BYTE bRampUp、BYTE bPwrOnOrBtnTch);**		
	说明	设置 GPO 的上升时长。 **注意**：对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456 以及 GPO789。在同一组中，配置一个 GPO 也便配置了其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		bRampUp	用于确定上升时长值的全局周期寄存器映射图	0 至 3 0 — GLOBAL_PERIOD_1 1 — GLOBAL_PERIOD_2 2 — GLOBAL_PERIOD_3 3 — GLOBAL_PERIOD_4
		bPwrOnOrBtnTch	上电或按键触摸 LED 效果	1 或 2 1 — 上电 LED 2 — 按键触摸 LED
	返回值	无		
	示例	MBR_LEDEffectsRampUp(GPO8、GLOBAL_PERIOD_1、POWER_ON_LED_EFFECTS) GPO8、GLOBAL_PERIOD_1、POWER_ON_LED_EFFECTS 是宏，它们的值分别为 8、0、1（inputs.h）。		
20	**原型**	**void MBR_PowerONLEDEffectSeq(BYTE bPwrOnSeq);**		
	说明	设置上电 LED 效果的序列（同时或顺序）。要确保调用此 API 之前您已经启用了上电 LED 效果。		
	参数	**名称**	**说明**	**可能值**
		bPwrOnSeq	上电 LED 效果序列的类型	0 或 1 0 — 同时 1 — 顺序
	返回值	无		
	示例	MBR_PowerONLEDEffectSeq(POWER_ON_SEQUENTIAL); POWER_ON_SEQUENTIAL 是一个宏，它的值为 1（inputs.h）。		

21	**原型**	**void MBR_PowerONLEDEffects(BYTE bEnable);**		
	说明	启用或禁用上电的 LED 效果。		
	参数	**名称**	**说明**	**可能值**
		bEnable	启用或禁用效果	0 或 1 0 — 禁用效果 1 — 启用效果
	返回值	无		
	示例	MBR_PowerONLEDEffects(FEATURE_ENABLE); FEATURE_ENABLE 是值为 1 的宏（inputs.h）。		
22	**原型**	**void MBR_ButtonLEDEffects(BYTE bEnable);**		
	说明	启用或禁用按键触摸 LED 效果		
	参数	**名称**	**说明**	**可能值**
		bEnable	启用或禁用效果	0 或 1 0 — 禁用效果 1 — 启用效果
	返回值	无		
	示例	MBR_ButtonLEDEffects(FEATURE_DISABLE); FEATURE_DISABLE 是值为 0 的宏（inputs.h）。		
23	**原型**	**void MBR_StandbyModeLEDBrightness(BYTE bLEDBrightness);**		
	说明	设置待机模式的 LED 亮度级别。		
	参数	**名称**	**说明**	**可能值**
		bLEDBrightness	每个寄存器映射图的待机模式亮度级别	0 至 3 0 — 亮度为 0% 1 — 亮度为 20% 2 — 亮度为 30% 3 — 亮度为 50%
	返回值	无		
	示例	MBR_StandbyModeLEDBrightness(STDBY_LED_50); STDBY_LED_50 是数值为 3 的宏（inputs.h）		
24	**原型**	**void MBR_LEDEffectLastButton(BYTE bEnable);**		
	说明	启用或禁用在最后按键触摸功能的 LED 效果。		
	参数	**名称**	**说明**	**可能值**
		BYTE bEnable	启用或禁用效果	0 或 1 0 — 禁用效果 1 — 启用效果
	返回值	无		
	示例	MBR_LEDEffectLastButton(FEATURE_DISABLE); FEATURE_DISABLE 是值为 0 的宏（inputs.h）。		

25	**原型**	**void MBR_SetGlobalPeriod(BYTE bPeriodReg, WORD wPeriodValue);**		
	说明	设置全局周期寄存器中的周期值		
	参数	**名称**	**说明**	**可能值**
		bPeriodReg	全局周期寄存器映射图	0 至 3 0 — GLOBAL_PERIOD_1 1 — GLOBAL_PERIOD_2 2 2 — GLOBAL_PERIOD_3 3 — GLOBAL_PERIOD_4
		wPeriodValue	全局周期值以 ms 为单位	0 至 1600
	返回值	无		
	示例	MBR_SetGlobalPeriod(GLOBAL_PERIOD_1,600) GLOBAL_PERIOD_1 是值为 0 的宏（inputs.h）。		
26	**原型**	**void MBR_SetAllGlobalPeriods(WORD awPeriodValue[]);**		
	说明	设置所有全局周期寄存器的周期值。		
	参数	**名称**	**说明**	**可能值**
		awPeriodValue	指向包含周期值（单位为 ms）的 4 字阵列的指针	0 至 1600
	返回值	无		
	示例	MBR_SetAllGlobalPeriods(wTestBuffer); wTestBuffe 是 4 字阵列 wTestBuffer[4]的基本指针。		
27	**原型**	**void MBR_SetAllLEDParameters(BYTE bGPO, BYTE abParam[]);**		
	说明	为所有 GPO 设置所有 LED 效果的参数。 **注意：**对于 LED 效果，GPO 被分组为 GPO0、GPO123、GPO456、GPO789。 配置组内的一个 GPO 也便配置了组内的其他 GPO。		
	参数	**名称**	**说明**	**可能值**
		bGPO	GPO 编号	0 至 9
		awPeriodValue	指向保存配置参数的 9 字节阵列的指针	字节[0] — 上电或按键触摸效果 字节[1] — 高亮度水平 字节[2] — 低亮度水平 字节[3] — 映射到全局周期寄存器的上升时长 字节[4] — 映射到全局周期寄存器的下降时长 字节[5] — 映射到全局周期寄存器的高亮度时长 字节[6] — 映射到全局周期寄存器的低亮度时长 字节[7] — 重复频率 字节[8] — 喘息效应使能/禁止
	返回值	无		
	示例	MBR_SetAllLEDParameters（GPO2, bconfig）； bconfig 为 9 字节阵列 bconfig[9]的指针。		
28	**原型**	**BYTE MBR_ReadDeviceID(void);**		
	说明	读取 CY8CMBR2110 器件的 ID 号。		
	参数	无		
	返回值	CY8CMBR2110的器件ID。此ID号为0xA1。		
	示例	MBR_ReadDeviceID();		

29	**原型**	**BYTE MBR_ReadFWRevision(void);**		
	说明	读取从属器件的固件版本。		
	参数	无		
	返回值	器件的固件版本		
	示例	MBR_ReadFWRevision();		
30	**原型**	**void MBR_SetDebugSensorNumber(BYTE bSensor);**		
	说明	设置传感器的编号（通过此传感器传输调试数据）		
	参数	**名称**	**说明**	**可能值**
		bSensor	传感器编号	0 至 9
	返回值	无		
	示例	MBR_SetDebugSensorNumber(CS0)； CS0 是值为 0 的宏（inputs.h）。		
31	**原型**	**void MBR_SetDebugDataParameter(BYTE bParameter);**		
	说明	设置调试数据中被发送走的参数类型。		
	参数	**名称**	**说明**	**可能值**
		bParameter	参数类型	0 至 4 0 — C_P 1 — 原始信号 2 — 差值 3 — 原始信号，基准线 4 — 所有参数（C_P、原始信号、差值、基准线以及信噪比）
	返回值	无		
	示例	MBR_SetDebugDataParameter(DEBUG_PARAM_CP)； DEBUG_PARAM_CP 是值为'0'的宏（inputs.h）。		
32	**原型**	**void MBR_ReadDebugData(BYTE abDebugData[]);**		
	说明	从调试数据寄存器映射图读取所选参数的调试数据		
	参数	**名称**	**说明**	**可能值**
		abDebugData	指向保存调试数据的 25 字节阵列的指针	
	返回值	无		
	示例	MBR_ReadDebugData(bgetdata)； bgetdata 为 25 字节阵列 bgetdata[25]的指针。 使用调试数据更新了该阵列。		

33	**原型**	**void MBR_SetBuzzer(BYTE bEnable);**		
	说明	使能或禁用音频反馈（蜂鸣器）。		
	参数	**名称**	**说明**	**可能值**
		bEnable	使能或禁用蜂鸣器	0 或 1 0 — 使能 1 — 禁用
	返回值	无		
	示例	MBR_SetBuzzer(FEATURE_ENABLE) FEATURE_ENABLE 是值为 1 的宏（inputs.h）。		
34	**原型**	**void MBR_SetBuzzerPins(BYTE bBuzzerPins);**		
	说明	设置蜂鸣器的引脚数。		
	参数	**名称**	**说明**	**可能值**
		bBuzzerPins	蜂鸣器输出引脚数量	0 或 1 0 — 1 引脚蜂鸣器 1 — 2 引脚蜂鸣器
	返回值	无		
	示例	MBR_SetBuzzerPins(BUZZER_AC_2_PIN)； BUZZER_AC_2_PIN 是值为 1 的宏（inputs.h）。		
35	**原型**	**void MBR_SetBuzzerIdleState(BYTE bIdleState);**		
	说明	设置蜂鸣器引脚的闲置状态。		
	参数	**名称**	**说明**	**可能值**
		bIdleState	蜂鸣器闲置状态	0 或 1 0 — 低电平 1 — 高电平
	返回值	无		
	示例	MBR_SetBuzzerIdleState(BUZZER_IDLE_HIGH)；BUZZER_IDLE_HIGH 是值为 1 的宏（inputs.h）		
36	**原型**	**void MBR_SetBuzzerFrequency(BYTE bFrequency);**		
	说明	设置蜂鸣器输出的频率。		
	参数	**名称**	**说明**	**可能值**
		bFrequency	蜂鸣器输出频率	1 至 7 1 — 4000 Hz 2 — 2670 Hz 3 — 2000 Hz 4 — 1600 Hz 5 — 1330 Hz 6 — 1140 Hz 7 — 1000 Hz
	返回值	无		
	示例	**MBR_SetBuzzerFrequency (**BUZZER_FREQ_1000)； BUZZER_FREQ_1000 是数值为 7 的宏（inputs.h）。		

37	**原型**	**void MBR_SetBuzzerOutputDuration(WORD wDuration);**		
	说明	设置蜂鸣器输出的持续时间。		
	参数	**名称**	**说明**	**可能值**
		wDuration	蜂鸣器输出时长，单位为毫秒（ms）	（0 至 127） * 按键扫描速率
	返回值	无		
	示例	MBR_SetBuzzerOutputDuration(1000);		
38	**原型**	**void MBR_SetAllBuzzerParameters(BYTE bEnable, BYTE bParameters[], WORD wOutputDuration);**		
	说明	设置 CY8CMBR2110 器件的所有蜂鸣器参数。		
	参数	**名称**	**说明**	**可能值**
		bEnable	使能或禁用蜂鸣器	0 或 1 0 — 禁用 1 — 使能
		bParameters	指向保存所需输入的 3 字节阵列的指针	字节[0] — 蜂鸣器引脚的数量 字节[1] — 蜂鸣器闲置状态 字节[2] — 蜂鸣器输出频率
		wOutputDuration	蜂鸣器输出的持续时间，单位为毫秒（ms）	（0 至 127） * 按键扫描速率
	返回值	无		
	示例	MBR_SetAllBuzzerParameters(FEATURE_ENABLE , bbuzzconfig ,1000)； FEATURE_ENABLE 是值为 1 的宏（inputs.h），bbuzzconfig 为 3 字节阵列（包含蜂鸣器引脚数量、闲置状态以及频率的详细信息）的指针（1000 为蜂鸣器的持续时间）。		
39	**原型**	**void MBR_SetI2CSlaveAddress (BYTE bNewSlaveAddress);**		
	说明	设置 CY8CMBR2110 器件的 I²C 地址。 默认地址为 37h。		
	参数	**名称**	**说明**	**可能值**
		bNewSlaveAddress	器件的新地址值	0x00 至 0x7F
	返回值	无		
	示例	MBR_SetI2CSlaveAddress(50);		
40	**原型**	**void MBR_SetAdaptiveThreshold(BYTE bSetRest);**		
	说明	使能或禁用 CY8CMBR2110 器件的自动阈值功能。		
	参数	**名称**	**说明**	**可能值**
		bSetRest	使能或禁用自动阈值功能	0 或 1 0 — 禁用 1 — 使能
	返回值	无		
	示例	MBR_SetAdaptiveThreshold(FEATURE_ENABLE)； FEATURE_ENABLE 是值为 1 的宏（inputs.h）。		

41	**原型**	**void MBR_SetSensitivity(BYTE bButtonNumber, BYTE bButtonSensitivityLevel);**		
	说明	设置一个按键的灵敏度值。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
		bButtonSensitivityLevel	按键的灵敏度水平	1 至 3 1 — 高级灵敏度 2 — 中级灵敏度 3 — 低级灵敏度
	返回值	无		
	示例	MBR_SetSensitivity (CS9, SENSITIVITY_MEDIUM)； CS9 和 SENSITIVITY_MEDIUM 分别是数值为 9 和 2 的宏（inputs .h）。		
42	**原型**	**void MBR_SetSensitivityAll(BYTE bsensitivity[]);**		
	说明	设置所有按键的灵敏度值。		
	参数	**名称**	**说明**	**可能值**
		bsensitivity	指向 10 字节阵列的指针，该阵列中保存所有按键的灵敏度水平	1 至 3 1 — 高级灵敏度 2 — 中级灵敏度 3 — 低级灵敏度
	返回值	无		
	示例	test_MBR_SetSensitivityAll(bBuffer)； bBuffer 为 10 字节阵列 bBuffer[10]的指针，该阵列中保存所有按键的灵敏度值（第一字节与按键 0 相应，…，第十字节与按键 9 相应）。		
43	**原型**	**void MBR_SetDebounce(BYTE bButtonNumber, BYTE bDebouncevalue);**		
	说明	设置按键的去抖动级别。 按键 1 至 9 具有同样的值。 不能单独配置这些值。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 或 1 0 — 按键 0 的级别 1 — 按键（1 - 9）的级别
		bDebouncevalue	按键的去抖动值	1 至 255
	返回值	无		
	示例	MBR_SetDebounce(DEBOUNCE_FOR_CS0, 200)； DEBOUNCE _FOR_CS0 是值为 0 的宏（inputs.h）。		
44	**原型**	**void MBR_SetFingerThreshold(BYTE bButtonNumber, BYTE bFingerthreshold);**		
	说明	设置一个按键的手指阈值级别。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
		bFingerthreshold	手指阈值级别	0 至 15
	返回值	无		
	示例	MBR_SetFingerThreshold(CS3, FINGER_THRESHOLD_180)； CS3 和 FINGER_THRESHOLD_180 分别为值是 3 和 10 的宏（inputs.h）。		

45	**原型**	**void MBR_SetFingerThresholdAll(BYTE bFingerthreshold[]);**		
	说明	设置所有传感器的手指阈值级别。		
	参数	**名称**	**说明**	**可能值**
		bFingerthreshold	指向包含手指阈值的 10 字节阵列的指针	0 至 15
	返回值	无		
	示例	MBR_SetFingerThresholdAll(Buffer)； Buffer 为 10 字节阵列 Buffer[10]的指针，该阵列中包含所有按键的手指阈值范围。 第一字节与按键编号 0 相应，...，第十字节与按键编号 9 相应）。		
46	**原型**	**BYTE MBR_ReadSensorStatus(BYTE bButtonNumber);**		
	说明	读取某个按键的当前状态（为了检查当前的按键触摸状态）。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
	返回值	0 或 1 0 — 按键未被按下（OFF） 1 — 按键被按下（ON）		
	示例	MBR_ReadSensorStatus(CS5)； CS5 是值为 5 的宏（inputs.h）。		
47	**原型**	**WORD MBR_ReadSensorStatusAll(void);**		
	说明	读取所有按键的当前状态。		
	返回值	表示所有传感器的当前状态的两个字节。 LSB 为按键 0 到 7 MSB 的头两位为按键 8 和 9 例如，0x0301 表示：按键 0、8 和 9 被按下（ON），剩下的按键未被按下（OFF）。		
	示例	MBR_ReadSensorStatusAll();		
48	**原型**	**WORD MBR_ReadLatchStatusAll(void);**		
	说明	读取所有传感器的锁定状态。		
	参数	无		
	返回值	表示所有传感器的当前组团为锁定的两个字节。 LSB 表示按键 0 到 7 MSB 的前两位表示按键 8 和 9 例如，0x0301 表示：当前读取 I^2C 前，按键 0、8 和 9 被按下（ON），剩下的按键未被按下（OFF）。		
	示例	MBR_ReadLatchStatusAll();		
49	**原型**	**void MBR_EnterDeepSleep(void);**		
	说明	设置工作模式寄存器中的深度睡眠位为 1，以便器件进入深度睡眠模式。 请按照深度睡眠模式一节中介绍的各步骤来进入深度睡眠模式。		
	参数	无		
	返回值	无		
	示例	MBR_EnterDeepSleep();		

50	**原型**	**void MBR_SetPowerOptimization(BYTE bOptimization);**		
	说明	设置 CY8CMBR2110 器件的功耗优化或响应时间优化设计。		
	参数	**名称**	**说明**	**可能值**
		bOptimization	使能功耗优化或响应时间优化	0 或 1 0 — 表示响应时间优化 1 — 表示功耗优化
	返回值	无		
	示例	MBR_SetPowerOptimization(PWR_CONS_OPT); PWR_CONS_OPT 是值为 1 的宏（inputs.h）。		
51	**原型**	**void MBR_SetScanRate(BYTE bSetscanvalue);**		
	说明	设置 CY8CMBR2110 器件的扫描速率。		
	参数	**名称**	**说明**	**可能值**
		bSetscanvalue	每个寄存器映射的扫描速率值	0 至 31 0 – 25 ms 31 – 561 ms
	返回值	无		
	示例	MBR_SetScanRate(30);		
52	**原型**	**void MBR_SetHGPOValue(BYTE bHGPO_Number, BYTE bDriveLogic);**		
	说明	设置一个 HGPO 的驱动逻辑。		
	参数	**名称**	**说明**	**可能值**
		bHGPO_Number	主机控制 GPO（HGPO）的编号	0 至 3 0 — HGPO0 1 — HGPO1 2 — HGPO2 3 — HGPO3
		bDriveLogic	驱动逻辑电平	0 或 1 1 — 高电平 0 — 低电平
	返回值	无		
	示例	MBR_SetHGPOValue(HOSTGPO_3、HOSTGPO_HIGH); HOSTGPO_3 和 HOSTGPO_HIGH 分别是值为 3 和 1 的宏（inputs.h）。		

53	**原型**	**void MBR_SetAllHGPOValue(BYTE bdriveGP0、BYTE bdriveGP1、BYTE bdriveGP2、BYTE bdriveGP3);**		
	说明	设置所有 HGPO 的驱动逻辑。		
	参数	**名称**	**说明**	**可能值**
		bdriveGP0	HGPO0 的驱动逻辑	0 或 1
		bdriveGP1	HGPO1 的驱动逻辑	0 或 1
		bdriveGP2	HGPO2 的驱动逻辑	0 或 1
		bdriveGP3	HGPO3 的驱动逻辑	0 或 1
	返回值	无		
	示例	void MBR_SetAllHGPOValue(HOSTGPO_HIGH、HOSTGPO_HIGH、HOSTGPO_LOW、HOSTGPO_LOW); HOSTGPO_HIGH、HOSTGPO_HIGH、HOSTGPO_LOW 和 HOSTGPO_LOW 分别是值为 1、1、0、0 的宏（inputs.h）。		
54	**原型**	**void MBR_AnalogOutput(BYTE bSet_Reset)**		
	说明	使能或禁用 CY8CMBR2110 器件的模拟输出电压功能。		
	参数	**名称**	**说明**	**可能值**
		bSet_Reset	设置或复位模拟输出电压功能	0 或 1
	返回值	无		
	示例	MBR_AnalogOutput(FEATURE_ENABLE); FEATURE_ENABLE 是值为 1 的宏（inputs.h）。		
55	**原型**	**void MBR_SetToggle (BYTE bButtonNumber, BYTE fSet_Reset);**		
	说明	使能或禁用 CY8CMBR2110 器件按键的切换功能。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
		fSet_Reset	使能或禁用切换	0 或 1 0 — 禁止 1 — 使能
	返回值	无		
	示例	MBR_SetToggle(CS0,FEATURE_ENABLE); FEATURE_ENABLE 是值为 1 的宏（inputs.h）。		
56	**原型**	**void MBR_SetToggleAll(BYTE afSet_Reset[]);**		
	说明	使能或禁用所有按键的切换功能		
	参数	**名称**	**说明**	**可能值**
		afSet_Reset	指向包含值的 2 字节阵列的指针	字节[1] — 0x00 至 0xFF 字节[2] — 0x00 至 0x03
	返回值	无。		
	示例	MBR_SetToggleAll(Buffer); 缓冲区为 2 字节阵列的指针缓冲区 缓冲区[1]包含按键 0 至 7 的值 缓冲区[2]的头 2 位包含按键 8 和 9 的值。 例如，缓冲区[1] = 0xFF 使能按键 0 至 7 的切换；缓冲区[2] = 0x01 使能按键 8 的切换，并禁止按键 9 的切换。		

57	**原型**	**void MBR_LEDONTime(BYTE bSet_Reset);**		
	说明	使能或禁用 CY8CMBR2110 器件的 LED ON 时长功能。		
	参数	**名称**	**说明**	**可能值**
		bSet_Reset	使能或禁用功能	0 或 1 0 — 禁止 1 — 使能
	返回值	无		
	示例	MBR_LEDONTime(FEATURE_DISABLE); FEATURE_DIASBLE 是值为 0 的宏（inputs.h）。		
58	**原型**	**BYTE MBR_ReadValidSensors(void);**		
	说明	读取有效的传感器/按键计数 由于短接到 VDD、短接到 GND、C_P 值错误和 CMOD 值错误，按键可能被使能。		
	参数	无		
	返回值	一个数据字节表示有效的传感器计算 返回值为 8 表示八个按键被使能和两个按键被禁用。		
	示例	MBR_ReadValidSensors();		
59	**原型**	**BYTE MBR_ReadFMEAGround(BYTE bButtonNumber);**		
	说明	读取一个接地短路按键的系统诊断数据（检查一个按键是否接地短路）。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
	返回值	0 或 1 0 — 按键没有短路接地 1 — 按键被短路接地		
	示例	MBR_ReadFMEAGround(CS9); CS9 是值为‘9’的宏（inputs.h）。		
60	**原型**	**WORD MBR_ReadFMEAGroundAll(void);**		
	说明	读取所有接地短路按键的系统诊断数据。		
	参数	无		
	返回值	两个字节表示接地短路的按键 LSB 为按键 0 至 7 MSB 前两位为按键 8 至 9。 例如，0x02F1 表示按键 0、4、5、6、7、9 均接地短路。		
	示例	MBR_ReadFMEAGroundAll();		
61	**原型**	**BYTE MBR_ReadFMEAGround(BYTE bButtonNumber);**		
	说明	读取按键短路连接 VDD 的系统诊断数据（检查一个按键是否对 VDD 短路）。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
	返回	0 或 1 0 — 按键没有短路接到 VDD 1 — 按键短路接到 VDD。		
	示例	MBR_ReadFMEAGround(CS9); CS9 是值为 9 的宏（inputs.h）。		

62	**原型**	**WORD MBR_ReadFMEAVDDAll(void);**		
	说明	读取所有短路至 VDD 按键的系统诊断数据。		
	参数	无		
	返回	两个字节表示短路接到 VDD 的按键。 LSB 为按键 0 至 7 MSB 前两位为按键 8 和 9。 例如，0x02F1 表示按键 0、4、5、6、7、9 均短路至 VDD。		
	示例	MBR_ReadFMEAVDDAll();		
63	**原型**	**WORD MBR_ReadFMEASnsToSnsAll(void);**		
	说明	读取所有按键与按键短路的系统诊断数据。		
	参数	无		
	返回	两个字节表示短路至另一个按键的按键。 LSB 为按键 0 至 7 MSB 前两位为按键 8 和 9。 例如，0x02F1 表示按键 0、4、5、6、7、9 均短路至另一个按键。		
	示例	MBR_ReadFMEASnsToSnsAll();		
64	**原型**	**BYTE MBR_ReadFMEASensorCP(BYTE bButtonNumber);**		
	说明	读取高 Cp 值按键的系统诊断数据。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
	返回	0 或 1 0 — 按键的 Cp 值较合适（Cp < 40 pF） 1 — 按键的 Cp 值较高（Cp > 40 pF）。		
	示例	MBR_SetDebugSensorNumber(CS0); CS0 是值为 0 的宏（inputs.h）。		
65	**原型**	**WORD MBR_ReadFMEASensorCpAll(void);**		
	说明	读取高 C_P 值按键的系统诊断数据。		
	参数	无		
	返回	两个字节表示 Cp 值较高的按键。 LSB 为按键 0 至 7 MSB 前两位为按键 8 和 9。 例如，0x02F1 表示按键 0、4、5、6、7、9 的 Cp 值都大于 40 pF。		
	示例	MBR_ReadFMEASensorCpAll();		
66	**原型**	**BYTE MBR_ReadFMEACMOD(void);**		
	说明	读取 CMOD 的系统诊断（检查 CMOD 电容值）。		
	参数	无		
	返回值	0 — 若 CMOD 值为 1 至 4nF 1 — 若 COMD 值高于 4 nF 2 — 若 CMOD 值低于 1 nF。		
	示例	MBR_ReadFMEACMOD();		

67	**原型**	**BYTE MBR_ReadSensorSNR(BYTE bButtonNumber);**		
	说明	读取按键的 SNR 值。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
	返回值	一个字节表示按键的信噪比（SNR）。 每个按键的 SNR 取值范围为 0 至 15。		
	示例	MBR_ReadSensorSNR(CS9)； CS9 是值为 9 的宏（inputs.h）。		
68	**原型**	**void MBR_ReadSensorSNRAll(BYTE bSensor_SNR[TOTAL_BUTTON_COUNT]);**		
	说明	读取所有按键的 SNR 值。		
	参数	**名称**	**说明**	**可能值**
		bSensor_SNR[TOTAL_BUTTON_COUNT]	10 字节阵列的指针，用以保持器件的读取值。	
	返回值	无		
	示例	MBR_ReadSensorSNRAll(buffer)； buffer 是一个 10 字节阵列的指针（从按键 0 开始，使用 SNR 值来对该阵列进行更新；第一字节与按键 0 相应，第二字节与按键 1 相应，… 第十字节与按键 9 相应）。		
69	**原型**	**void MBR_SetAutoResetTime(BYTE bSet_time);**		
	说明	设置所有按键的自动复位时间。		
	参数	**名称**	**说明**	**可能值**
		bSet_time	自动复位的时间值。	1 — 无限制 2 — 5 秒 3 — 20 秒
	返回值	无		
	示例	MBR_SetAutoResetTime(AUTO_RESET_20S)； AUTO_RESET_20S 是值为 3 的宏（inputs.h）。		
70	**原型**	**void MBR_SetEMC(BYTE bSet_Reset);**		
	说明	使能或禁止 CY8CMBR2110 器件的 EMC 功能。		
	参数	**名称**	**说明**	**可能值**
		bSet_Reset	使能或禁止 EMC	0 或 1 0 — 禁用 1 — 使能
	返回值	无		
	示例	MBR_SetBuzzer(FEATURE_ENABLE)； FEATURE_ENABLE 是值为 1 的宏（inputs.h）。		

71	**原型**	**void MBR_SetFSS(BYTE bButtonNumber, BYTE fSet_Reset);**		
	说明	使能或禁止一个按键的 FSS 功能（侧翼传感器抑制）。		
	参数	**名称**	**说明**	**可能值**
		bButtonNumber	按键编号	0 至 9
		fSet_Reset	使能或禁止功能	0 或 1 0 — 禁止 1 — 使能
	返回值	无		
	示例	MBR_SetFSS（CS0，FEATURE_DISABLE）； CS0 和 FEATURE_DISABLE 是值分别为 0 和 1 的宏（inputs.h）。		
72	**原型**	**void MBR_SetFSSAllSensors(BYTE afSet_Reset[])**		
	说明	使能或禁止所有按键的 FSS（侧翼传感器抑制）功能。		
	参数	**名称**	**说明**	**可能值**
		afSet_Reset	指向保存配置直的 2 字节阵列的指针。	字节[1] — 0x00 至 0xFF 字节[2] — 0x00 至 0x03
	返回值	无		
	示例	MBR_SetFSSAllSensors(Buffer)； Buffer 是 2 字节阵列 Buffer[2]的指针。 Buffer[1]保持按键 0 至 7 的值 Buffer[2]前两位保持按键 8 至 9 的值。 示例：Buffer[1] = 0xFF 使能按键 0 至 7 的 FSS 功能，Buffer[2] = 0x01 使能按键 8 的功能，Buffer[2] =0x02 使能按键 9 的功能并禁止按键 8 的功能。		

7.2.2 低层 API

1	**原型**	**void MBR_WriteBytes(BYTE abWriteBuffer[], BYTE bNumberOfBytes)**		
	说明	将各字节的阵列写入到 CapSense 的从属器件上。		
	参数	**名称**	**说明**	**可能值**
		abWriteBuffer	指向主机 I²C 缓冲区阵列的指针	1 至 31 个字节
		bNumberOfBytes	要写入的字节数量	1 到 31
	返回值	无		
	示例	MBR_WriteBytes(abHostI2CBuffer, 3);		
2	**原型**	**void MBR_ReadBytes(BYTE abReadBuffer[] , BYTE bNumberOfBytes)**		
	说明	从配备 CapSense 的从属器件读取各字节的阵列。		
	参数	**名称**	**说明**	**可能值**
		abReadBuffer	指向主机 I²C 缓冲区阵列的指针	1 至 32 个字节。
		bNumberOfBytes	要读取的字节数量	1 至 32
	返回值	无		
	示例	MBR_ReadBytes(bHostI2CBuffer, 6);		
3	**原型**	**void MBR_Delay(WORD wDelayTime)**		
	说明	实现软件延迟（单位为毫秒）。		
	参数	**名称**	**说明**	**可能值**
		wDelayTime	延迟的时间（单位为毫秒）	
	返回值	无		
	示例	MBR_Delay(100);		

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$(2^N – 1)$/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_P）

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。

刷新间隔

传感器两次连续扫描间的时间。

扫描分辨率

由 CSD 模块生产的原始计数分辨率（单位为位）。

扫描时间

完成传感器的扫描过程所需要的时间。

自电容

与电路接地和电极相关的电容。

灵敏度

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

感应时钟

用来实现 CSD 感应方法的开关电容前端的时钟源。

传感器

请参见电容式传感器。

传感器自动复位

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：AN76000 - CY8CMBR2110 CapSense®设计指南

文档编号： 001-89147

版本	发布日期	变更人	更改说明
**	09/12/2013	GOX	本文档版本号为 Rev**，译自英文版 001-76000 Rev *B。
*A	11/20/2014	GOX	本文档版本号为 Rev*A，译自英文版 001-76000 Rev *C。
*B	11/09/2015	XZNG	本文档版本号为 Rev*B，译自英文版 001-76000 Rev *D。
*C	01/11/2018	XITO	本文档版本号为 Rev. *C，译自英文版 001-76000 Rev. *G。