AN78329 – CY8C20xx7/S

CapSense®デザイン ガイド

文書番号: 001-88331 Rev. *B

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
japan.cypress.com

著作権

© Cypress Semiconductor Corporation, 2012-2020. 本書面は, Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。) を含む) は, アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し, 本段落で特に記載されているものを除き, その特許権, 著作権, 商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず, かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合, Cypress は, (1) 本ソフトウェアの著作権に基づき, (a) ソースコード形式で提供されている本ソフトウェアについて, Cypress ハードウェア製品と共に用いるためにのみ, かつ組織内部でのみ, 本ソフトウェアの修正及び複製を行うこと, 並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ, (直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること, 並びに (2) 本ソフトウェア (Cypress により提供され, 修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき, Cypress ハードウェア製品と共に用いるためにのみ, 本ソフトウェアの作成, 利用, 配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用, 複製, 修正, 変換又はコンパイルを禁止する。

適用される法律により許される範囲内で, Cypress は, 本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても, 明示又は黙示をとわず, いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。いかなるコンピューティングデバイスも絶対に安全ということはない。従って, Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず, Cypress は, Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて, 本書面に記載された製品には, エラッタと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり, 公表された仕様とは異なる動作をする場合がある。適用される法律により許される範囲内で, Cypress は, 別途通知することなく, 本書面を変更する権利を留保する。Cypress は, 本書面に記載のある, いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は, 参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計, プログラム, かつテストすることは, 本書面のユーザーの責任において行われるものとする。Cypress 製品は, 兵器, 兵器システム, 原子力施設, 生命維持装置若しくは生命維持システム, 蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム, 汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用, 又は装置若しくはシステムの不具合が人身傷害, 死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計, 意図又は承認されていない。重要な構成部分とは, それの不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ, 若しくは本目的外使用に関連するいかなる請求, 損害又はその他の責任についても, Cypress はその全部又は一部をとわず一切の責任を負わず, かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求, 費用, 損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, CapSense, EZ-USB, F-RAM, 及び Traveo は, 米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは, cypress.com を参照すること。その他の名称及びブランドは, それぞれの権利者の財産として権利主張がなされている可能性がある。

目次

1. はじめに 7

1.1 概要 7
1.2 サイプレス CapSense の文書体系 7
1.3 CY8C20xx7/S CapSense ファミリの機能 9
1.4 本書の表記法 11

2. CapSense テクノロジー 12

2.1 CapSense の原理 12
2.2 CY8C20xx7/S の静電容量センシング方式 13
2.2.1 CapSense シグマ デルタ (CSD) 13
2.2.2 CapSense シグマ デルタ (CSD) PLUS 16
2.2.3 SmartSense_EMCPLUS 自動チューニング 19
2.2.4 ユーザー モジュールの選択 20

3. CapSense 設計ツール 21

3.1 概要 21
3.1.1 PSoC Designer およびユーザー モジュール 21
3.1.2 CY8C20xx7/S QuietZone スターター キット 22
3.1.3 CapSense データ表示ツール 22
3.2 ユーザー モジュール概要 23
3.3 CapSense ユーザー モジュールのグローバル アレイ 23
3.3.1 raw カウント 24
3.3.2 ベースライン 24
3.3.3 差分カウント (信号) 24
3.3.4 センサー状態 25
3.4 CSD／CSDPLUS ユーザー モジュール パラメーター 25
3.4.1 ユーザー モジュールの高レベル パラメーター 25
3.4.2 CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター 28
3.5 SmartSense_EMCPLUS ユーザー モジュールのパラメーター 32

4. ユーザー モジュールによる CapSense 性能のチューニング 34

4.1 一般的な注意事項 34
4.1.1 信号、ノイズ、および SNR (信号対ノイズ比) 34
4.1.2 充電／放電速度 35
4.1.3 ベースライン更新閾値の検証の重要性 36

4.2 CSD／CSDPLUS ユーザー モジュールのチューニング 36
4.2.1 CSD／CSDPLUS 用の推奨 C_{MOD} 値 37
4.2.2 I_{DAC} 範囲 37
4.2.3 自動校正 37
4.2.4 I_{DAC} 値 37
4.2.5 補正 I_{DAC} 値 37
4.2.6 プリチャージ源 37
4.2.7 プリスケーラ 38
4.2.8 分解能 38
4.2.9 スキャン速度 39
4.2.10 高レベル API パラメーター 40
4.2.11 高レベル パラメーターの設定 40
4.3 SmartSense_EMCPLUS ユーザー モジュールの使用 41
4.3.1 SmartSense_EMC_PLUS のガイドライン 41
4.3.2 相違点 41
4.3.3 SmartSense_EMC_PLUS 用の推奨 C_{MOD} 値 41
4.3.4 SmartSense_EMCPLUS ユーザー モジュールのパラメーター 41
4.3.5 SmartSense_EMCPLUS ユーザーモジュール固有のガイドライン 42
4.3.6 CapSense Sensor のスキャン時間 43
4.3.7 SmartSense_EMCPLUS 応答時間 44
4.3.8 SmartSense_EMCPLUS UM による最低限の SNR の確保方法 44
4.3.9 ファームウェア デザイン ガイドライン 45
4.4 CY8C20xx6A/AS から CY8C20xx7/S への設計の移行 48
4.4.1 廃止されたサポート／ユーザー モジュール 48
4.4.2 改善および新機能 48
4.4.3 ピンの互換性 48

5. 設計上の注意事項 49

5.1 オーバーレイの選択 49
5.2 ESD 保護 50
5.2.1 防止 50
5.2.2 リダイレクト 50
5.2.3 クランプ 50
5.3 電磁環境適合性 (EMC) の注意事項 50
5.3.1 放射性干渉 50
5.3.2 放射妨害波 51
5.3.3 伝導ノイズ耐性および伝導性放射 51
5.4 ソフトウェア フィルター 51
5.5 消費電力 52
5.5.1 システム設計の推奨事項 52
5.5.2 スリープ スキャン方式 52
5.5.3 応答時間対消費電力 52
5.5.4 平均消費電力の測定 53

5.6 ピン割り当て 53
5.7 GPIO の負荷瞬時変化 54
5.7.1 GPIO 負荷瞬時変化ノイズ減少用のハードウェア ガイドライン 55
5.7.2 GPIO 負荷瞬時変化ノイズ補正用のファームウェア ガイドライン 56
5.8 PCB レイアウト ガイドライン 58

6. 耐液性デザイン上の注意事項 59

6.1 シールド電極とガード センサー 59
6.1.1 シールド 59
6.1.2 ガード センサー 62
6.2 設計上の推奨事項 64

7. 近接センシング デザインに関する注意事項 65

7.1 近接センサーの種類 65
7.1.1 ボタン 65
7.1.2 ワイヤ 65
7.1.3 PCB 配線 65
7.1.4 センサー連動 66
7.2 設計上の推奨事項 66

8. 低消費電力設計上の注意事項 67

8.1 その他の省電力技術 67
8.1.1 駆動モードのアナログ HI-Z 設定 67
8.1.2 まとめ 68
8.1.3 スリープ モードの I²C スレーブの推奨実装方法 68
8.1.4 スリープ モードの混乱 68
8.1.5 保留中の割り込み 69
8.1.6 グローバル割り込みの有効化 69
8.2 ウェイクアップ後の実行シーケンス 69
8.2.1 PLL モードの有効化 69
8.2.2 グローバル割り込みの有効化の実行 69
8.2.3 スリープ モードの I²C スレーブの推奨実装方法 69
8.2.4 スリープ タイマー 70

9. リソース 72

9.1 ウェブサイト 72
9.2 データシート 72
9.3 テクニカル リファレンス マニュアル 72
9.4 開発キット 73
9.4.1 CY8C20xx7/S QuietZone スターター キット 73
9.4.2 ユニバーサル CapSense コントローラー キット 73
9.4.3 ユニバーサル CapSense モジュール基板 73

9.5 サンプル ボード ファイル 74
9.6 PSoC Programmer 76
9.7 CapSense データ表示ツール 76
9.8 PSoC Designer 76
9.9 サンプル コード 76
9.10 デザイン サポート 76

1. はじめに

1.1 概要

本文書は、静電容量センシング (CapSense®) 機能を CapSense コントローラーの CY8C20xx7/S ファミリに実装するための設計ガイドを提供します。本ガイドでは以下の項目について紹介します。

- CapSense コントローラーの CY8C20xx7/S ファミリの機能
- CapSense の動作原理
- CapSense 設計ツールの紹介
- 最適な性能用の CapSense システムのチューニング ガイド
- CapSense を使ったシステムの電気的および機械的デザインの注意事項
- CapSense の低消費電力設計における注意事項
- CapSense をシステムに組み込むための追加リソースおよびサポート

1.2 サイプレス CapSense の文書体系

図 1-1 および表 1-1 は、サイプレス CapSense の文書体系をまとめています。これらのリソースにより、CapSense 製品設計を完成させるために必要な情報にすばやくアクセスできます。図 1-1 に静電容量センシングを用いた製品設計サイクルの一般的なフローを示します。本ガイドに記載されている情報は、緑色でハイライト表示されたトピックに最も関連があります。表 1-1 には、図 1-1 で番号を付された各タスクをサポートする文書へのリンクを記載します。

図 1-1. 標準的な CapSense の製品設計フロー

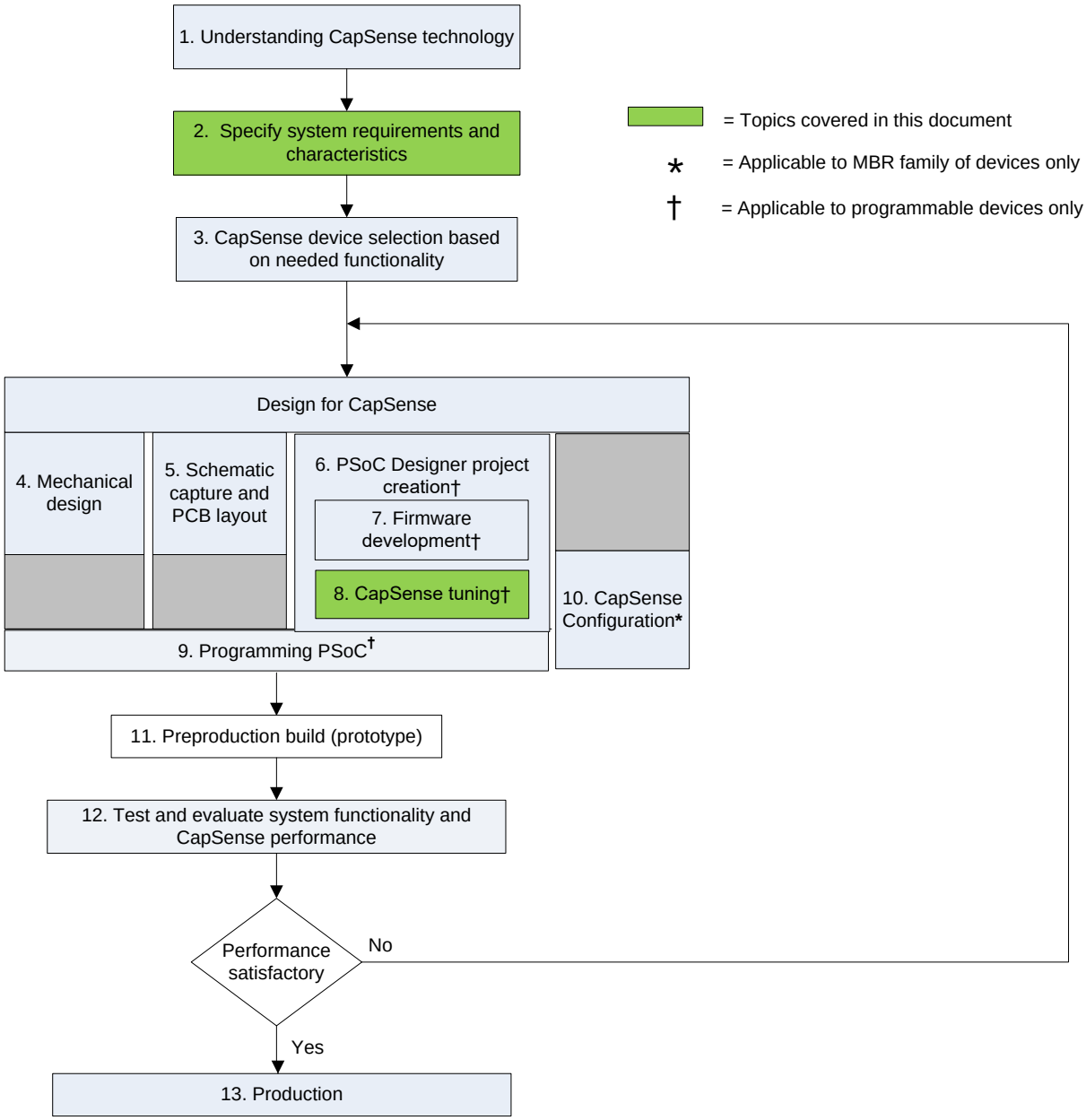


表 1-1.図 1-1 中に番号を付された設計タスクをサポートするサイプレスの文書

図 1-1 中に番号を付された設計タスク	サイプレス CapSense の関連文書
1	● CapSense 入門
2	● CapSense 入門 ● CY8C20xx7/S CapSense デバイス データシート
3	● CapSense 入門 ● CY8C20xx7/S CapSense デザイン ガイド (本書)
4	● CapSense 入門
5	● CapSense 入門
6	● PSoC® Designer™ユーザー ガイド
7	● アセンブリ言語のユーザー ガイド ● C 言語コンパイラのユーザー ガイド ● CapSense サンプル コード ● PSoC CY8C20xx7/S テクニカル リファレンス マニュアル
8	● PSoC ファミリ向け CapSense デザイン ガイド (本書) ● PSoC ファミリ向け CapSense ユーザー モジュール データシート (CSD、CSDPLUS および SmartSense_EMC_PLUS) ● PSoC CY8C20xx7/S テクニカル リファレンス マニュアル ●「 AN2397 -CapSense データ表示ツール」
9	● プログラマ ユーザー ガイド ● MiniProg3 ユーザー ガイド ● ISSP プログラミング仕様 - CY8C20045、CY8C20055、CY8C20065、CY8C20xx6A、CY8C20xx7 ●「AN59389 - CY8C20xx6A、CY8C20xx6AS、CY8C20xx6L および CY8C20xx7/S 用のホスト ソース シリアル プログラミング」
11	● CY8C20xx7/S CapSense デザイン ガイド (本書) ● CapSense サンプル コード

1.3 CY8C20xx7/S CapSense ファミリの機能

サイプレスの CY8C20xx7/S は、低電力、高性能、プログラマブル CapSense コントローラー ファミリで、以下の機能があります。

高度なタッチ センシング機能

- プログラマブル静電容量センシング要素
  - CapSense ボタン、スライダー、近接センサーの組み合わせに対応
  - ボタンとスライダーを実装する統合 API
  - 最大 31[a]の静電容量センサーまたは 6 個のスライダーに対応[b]
  - 5pF～45pF のセンサー寄生容量に対応
- SmartSense™自動チューニングにより市場投入までの時間を短縮
  - 電源投入時と実行中のチューニング パラメーターを自動的に設定および監視
  - デザインの可動性 - ユーザー インターフェース設計の変更に対して自己チューニング
  - 実行中の環境補正
  - 静電容量が最小 0.1pF までのタッチを検出
- 強化されたノイズ耐性と堅牢性

[a] 2 本のピンが I^2C 通信に、1 本のピンが C_{MOD} 接続に使用されることを前提にしています。

[b] 詳細については、CY8C20xx7/S データシートを参照してください。

- SmartSense_EMCPLUS は環境とノイズの変化を自動的に補正
- SmartSense_EMCPLUS は困難な導電性および放射ノイズ条件にあるアプリケーションに優れたノイズ耐性を提供
- 内部レギュレータは電源ノイズおよびリップルに対する安定性を実現し、最大 500mV の V_{DD} のリップルに対応可能
- SNR を向上するためのソフトウェア フィルターの統合 API

■ 超低消費電力
- 最適消費電力用の 3 つの異なる電力モード
- アクティブ モード、スリープ モード、ディープ スリープ モード (ディープ スリープ電流が 100nA)
- スリープからの復帰時間が 125ms 時のセンサーごとの消費電流が 28µA

■ 5 本の GPIO ピンにある被駆動シールド
- クラス最高の耐水性設計を提供
- 金属が存在する場合の堅牢な近接センシング
- 長い配線長をサポート
- 最大 100pF の負荷 (3MHz)

デバイスの特長

■ 高性能、低消費電力 M8C ハーバード アーキテクチャ プロセッサ
- 24MHz 内部クロックによる最大 4 MIPS

■ 柔軟性のある内蔵メモリ
- 最大 32KB のフラッシュ、最大 2KB の SRAM
- エミュレートした EEPROM をサポート

■ 高精度の、プログラマブルなクロック供給
- 内部主発振器 (IMO): 6／12／24MHz ± 5%
- 高精度 32kHz の外部水晶発振器のオプション

■ 拡張された汎用入出力 (GPIO) の特長
- ピン配置がプログラム可能な 34 の GPIO
- GPIO あたりに 25mA のシンク電流、デバイスあたりに 120mA の合計シンク電流
- すべての GPIO における内部プルアップ抵抗、HI-Z、オープン ドレイン、ストロングの駆動モード

■ ペリフェラルの特長
- 3 個の 16 ビット タイマー
- I^2C - マスター (100kHz) とスレーブ (400kHz)
- SPI - マスターとスレーブ - 46.9kHz～12MHz の設定可能範囲
- 10 ビットのインクリメンタル ADC - 0～1.2V の入力範囲

■ 動作条件
- 広い動作電圧範囲: 1.71V～5.5V
- 温度範囲: –40°C～+85°C

1.4 本書の表記法

表記法	説明
Courier New フォント	ファイルの場所、ユーザーが入力したテキスト、ソース コードを表示します。 C:\ ...cd\icc\
イタリック フォント	ファイル名および参考資料を表示します。 *PSoC Designer User Guide* にある *sourcefile.hex* ファイルを参照してください。
[角括弧、太字]	操作手順でキーボードに入力するコマンドを表示します。 [**Enter**]または[**Ctrl**] [**C**]
File > Open	メニュー パスを表示します。 File > Open > New Project
太字	操作手順でのコマンド、メニュー パス、アイコン名を表示します。 **File** アイコンをクリックして、**Open** をクリックします。
Times New Roman フォント	式を表示します。 2 + 2 = 4
灰色のボックス内のテキスト	製品の注意点やユニークな機能を説明します。

2. CapSense テクノロジー

2.1 CapSense の原理

CapSense はタッチセンシング技術で、センサーである CapSense コントローラー上の各 I/O ピンの静電容量を計測することによって機能します。図 2-1 に示すように、n 個のセンサーを備えたデザインにおいて、各センサー ピンの総容量を C_X,1～C_X,n の値を持つ等価集中コンデンサとしてモデル化できます。CY8C20xx7/S デバイスの内部回路が各 C_X の大きさを、後処理用に保存されるデジタル コードに変換します。他の構成要素である C_{MOD} は、CapSense コントローラーの内部電気回路によって使用され、「CY8C20xx7/S の静電容量センシング方式」でさらに詳しく説明します。

図 2-1. CY8C20xx7/S PSoC デバイスでの CapSense の実装

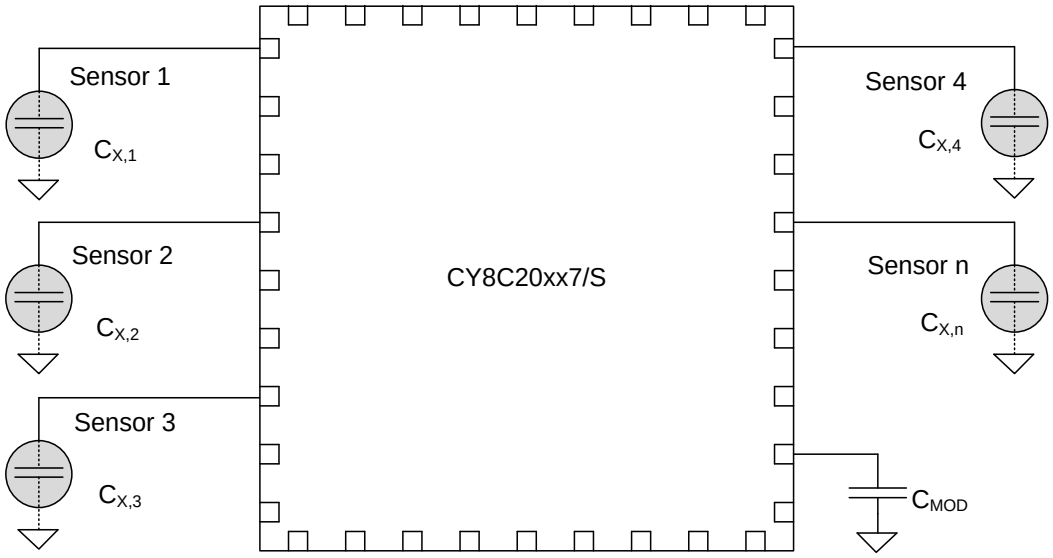


図 2-1 に示すように、各センサーI/O ピンは必要に応じて配線、ビアまたはその両方でセンサー パッドに接続されます。オーバーレイはセンサー パッドを覆う非導電性のカバーであり、製品のタッチ インターフェースを構成します。指がオーバーレイに接触すると、人体の導電性と大きな体積によりセンサー パッドに対し平行な接地された導体面となります。これを図 2-2 に示します。この配置は平行板コンデンサとなり、その静電容量は次の式で表されます。

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

ここで、

C_F = センサーを覆うオーバーレイに接触する指により生じた静電容量

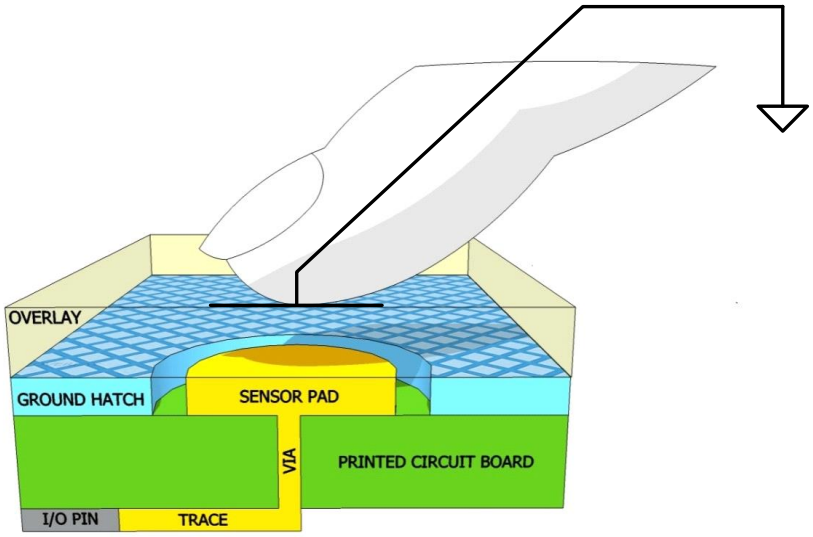
ε_0 = 真空の誘電率

ε_r = オーバーレイの比誘電率

A = 指とセンサー パッドが重なっている面積

D = オーバーレイの厚さ

図 2-2. 指でセンサーが起動したときの典型的な CapSense 基板の断面図



平行板コンデンサに加えて、オーバーレイに接触している指は、それ自身とすぐ近くにある他の導体との間に静電結合を引き起こします。このフリンジ電界の影響は平行板コンデンサと比較して通常小さく、無視することができます。

指がオーバーレイに触れなくても、センサーI/O ピンは寄生容量 (C_P) があります。C_P は、センサー (センサー パッド、配線、ビアを含む) とシステム内の他の導電体 (グランド面、配線、製品のシャーシまたは封入物など) の間で生じる電場と CapSense コントローラーの内部の寄生容量との組み合わせにより生じます。CapSense コントローラーは、センサー ピンに接続されるすべての静電容量 (C_X) を計測します。

指がセンサーに触れていない場合:

$C_X = C_P$ 式 2-2

指がセンサー パッドに触れた場合は、C_X は C_P と C_F の和に等しいです。

$C_X = C_P + C_F$ 式 2-3

一般的に、C_P は C_F より何十倍か大きい値です。C_P は通常 10pF～20pF ですが、極端な場合は 50pF まで高くなることもあります。C_F は通常 0.1pF～0.4pF です。C_P の大きさは CapSense システムのチューニング時にきわめて重要であり、「ユーザー モジュールによる CapSense 性能のチューニング」で説明されます。

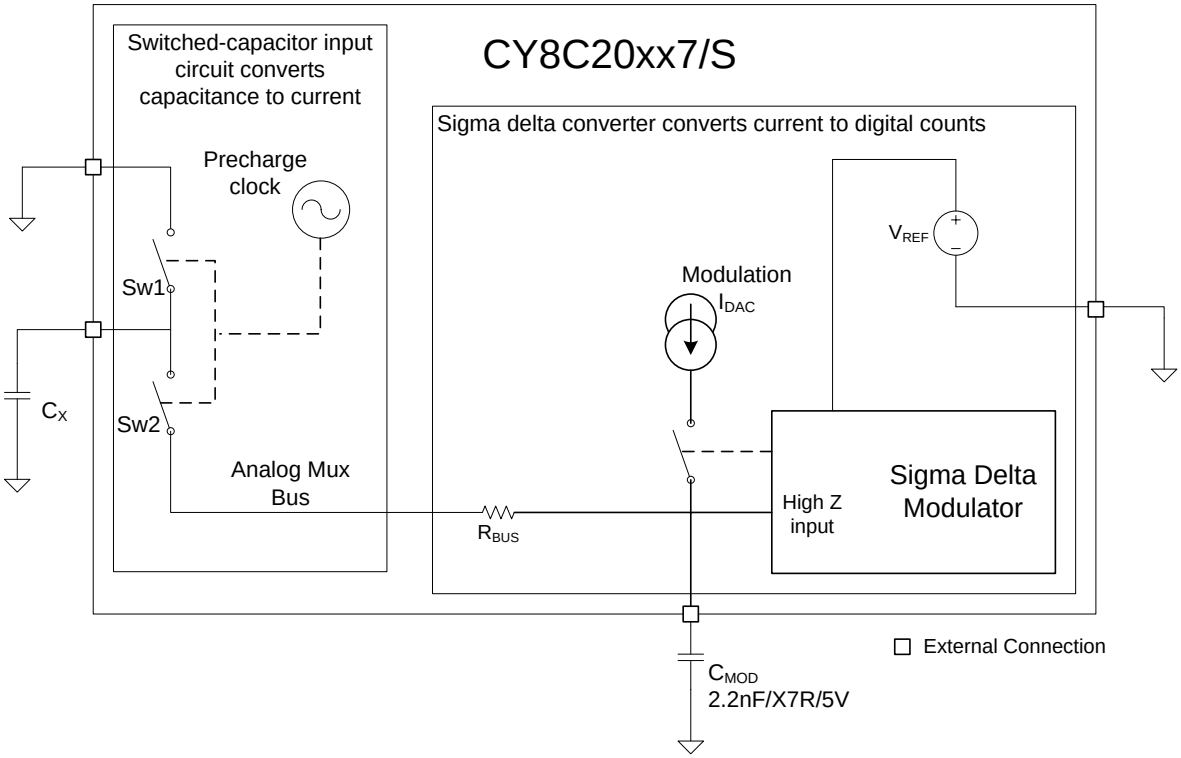
2.2 CY8C20xx7/S の静電容量センシング方式

CY8C20xx7/S デバイスは、センサー静電容量 (C_X) をデジタル カウントに変換する CSD、CSDPLUS および SmartSense_EMCPLUS CapSense 方式を対応します。CSDPLUS 方式は、CSD 方式の上位互換であり、CSD と比べて、いくつかの改善があります。これらの 2 つの方式はハードウェアで実装されます。SmartSense_EMCPLUS は、すべての CSDPLUS パラメーターを自動的に調整するためにファームウェアで実装された自動チューニング アルゴリズムを使用します。CSD、CSDPLUS および SmartSense_EMCPLUS 方式は、PSoC Designer ユーザー モジュールの形式で実装され、以下の項で説明されます。

2.2.1 CapSense シグマ デルタ (CSD)

図 2-3 は、センサー静電容量 (C_X) をデジタル カウントに変換する CSD 方式のブロック図を示します。この方式は、概念的に 2 つのブロックに分割できます。それらは静電容量を電流に変換するスイッチト キャパシタ入力、および電流をデジタルカウントに変換するシグマ デルタ コンバータです。これらのブロックは以下の項で説明されます。

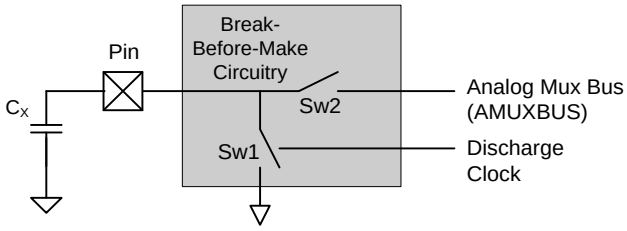
図 2-3. CSD ブロック図



2.2.1.1 スイッチト キャパシタ入力

CY8C20xx7/S デバイスの CSD 方式では、図 2-3 に示すように、C_X がスイッチト キャパシタ回路に接続されます。

図 2-4. スイッチト キャパシタ入力として設定されるピン



周波数 F_{SW} (図 2-6) の、位相が一致しない 2 つの非重複クロックが、スイッチ Sw1 および Sw2 を制御します。図 2-5 に示すように、Sw1 と Sw2 の連続切り替えにより、等価抵抗 R_S を形成します。等価抵抗 R_S の値は以下のとおりです。

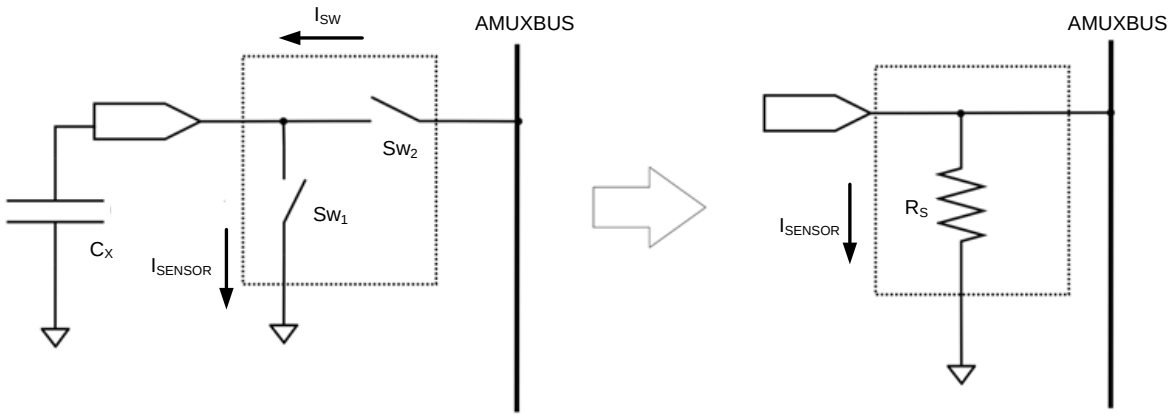
$$R_S = \frac{1}{C_X F_{SW}}$$ 式 2-4

ここで、

C_X = センサー静電容量

F_{SW} = スイッチング クロックの周波数

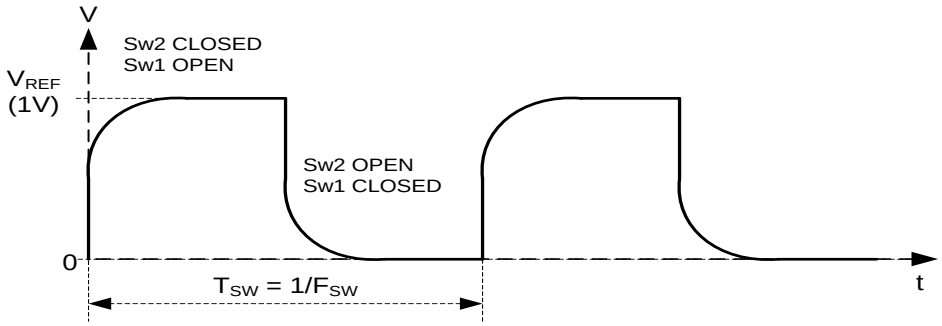
図 2-5. スイッチト キャパシタ入力が AMUXBUS から電流をシンク



シグマ デルタ コンバータは、V_{REF} 定数で AMUXBUS の電圧を維持します (この処理は「シグマ デルタ コンバータ」で説明されます)。図 2-6 はセンサー静電容量の両端における電圧波形を示します。したがって、スイッチ Sw1、Sw2 を非重複 (プリチャージ) クロックにより駆動すると、式 2-5 に示される平均電流シンク (I_{SENSOR}) が AMUXBUS で発生します。I_{SENSOR} は C_X の大きさに正比例します。

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

図 2-6. センサー静電容量の両端における電圧 (C_X)



2.2.1.2 シグマ デルタ コンバータ

シグマ デルタ コンバータは入力電流を対応するデジタル カウントに変換します。14 ページの図 2–3 に示すように、コンバータは、シグマ デルタ変調器と 1 個の電流ソーシングのデジタル アナログ コンバータ (I_{DAC})で構成されます。

シグマ デルタ変調器はオン／オフ マナーで 8 ビット I_{DAC} の電流を制御します。I_{DAC} は変調 I_{DAC} として知られており、本書で「I_{DAC}」または「変調 I_{DAC}」として呼ばれています。14 ページの図 2-3 に示すように、シグマ デルタ コンバータは外部積分コンデンサ C_{MOD} も必要とします。C_{MOD} の推奨値は 2.2nF です。

シグマ デルタ変調器は V_{REF} で C_{MOD} 電圧を維持するために、C_{MOD} の両端における少しの電圧変動に応じて変調 I_{DAC} をオンまたはオフに切り替えます。

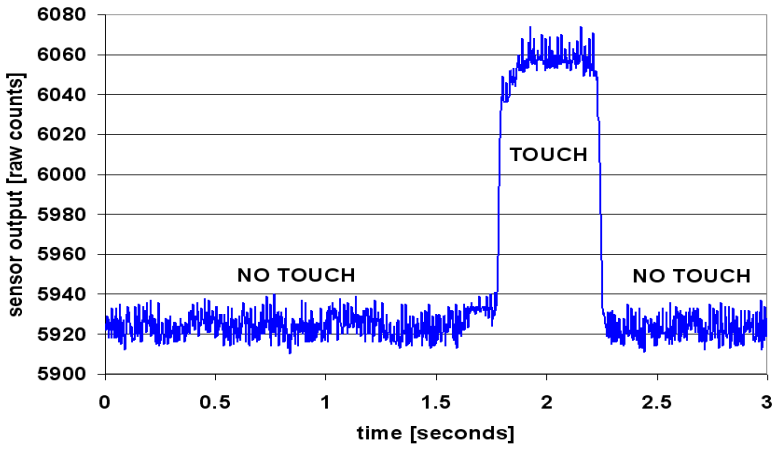
AMUX の平均電圧を安定状態の値 (V_{REF}) に維持するために、シグマ デルタ コンバータは変調ビット ストリーム デューティ サイクルを制御することで平均充電電流 (I_{DAC}) を I_{SENSOR} に一致させます。シグマ デルタ コンバータはセンサー スキャンの間ビット ストリームを保存します。その積算値は C_X に正比例した、raw カウントと呼ばれたデジタル出力値となります。

シグマ デルタ コンバータは 9 ビット～16 ビットの分解能で動作できます。「N」がシグマ デルタ コンバータの分解能であり、I_{DAC} が変調 I_{DAC} 電流の値である場合、raw カウントの近似値は以下の式から得られます。

$$\text{raw count} = (2^N - 1)\frac{V_{REF}\ F_{SW}\ C_X}{I_{DAC}}$$ 式 2-6

この raw カウントは高レベルのアルゴリズムによって解釈され、センサー状態を判断し、タッチを検出します。図 2-7 は、センサーを指で触れてから離すまで連続スキャンした CSD の raw カウントをプロットしたものです。「CapSense の原理」で説明したように、指の接触により C_X が C_F 分増加し、その結果 raw カウントが比例して増加します。定常状態の raw カウント レベルの変化を事前に設定された閾値と比較することで、センサーがオン (接触) 状態であるかオフ (非接触) 状態であるかを高レベル アルゴリズムにより判定できます。

図 2-7. 指を触れたときの CSD の raw カウント



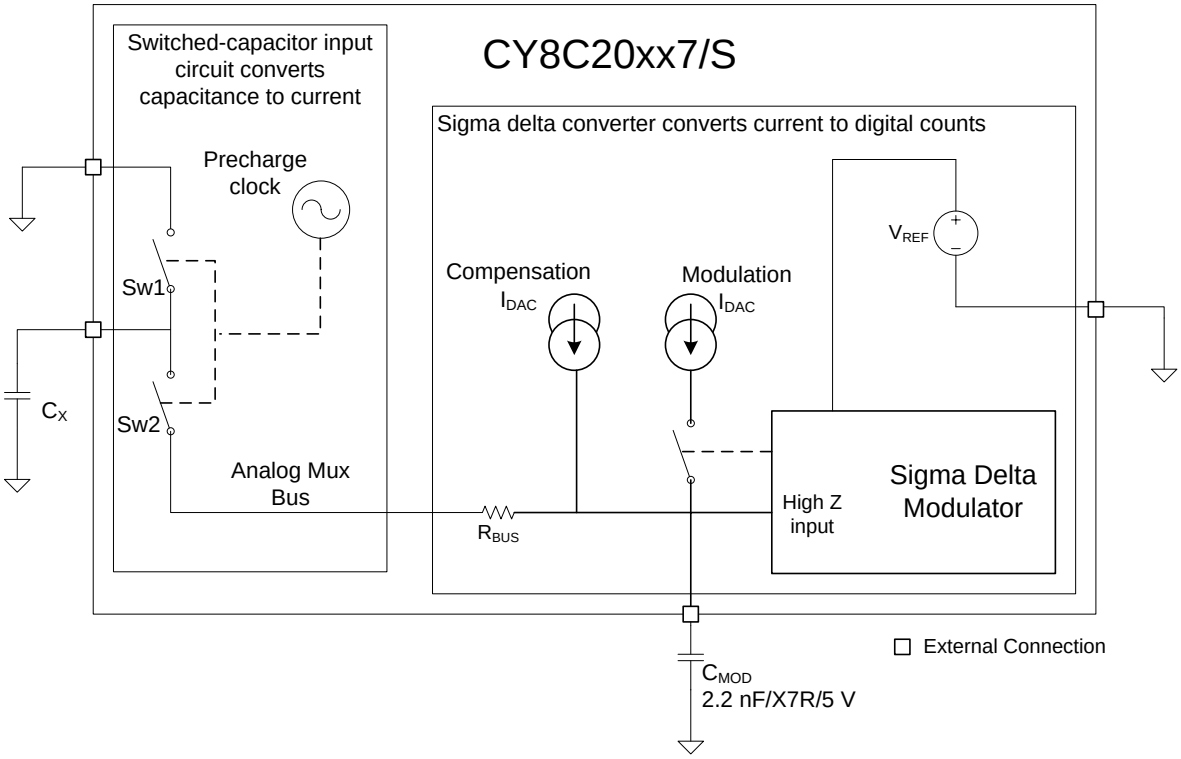
ハードウェア パラメーター (I_{DAC} や F_{SW} などの CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター) およびファームウェア パラメーター (ユーザー モジュールの高レベル パラメーター) は信頼性の高いタッチ検出のための最適値に調整する必要があります。チューニングの詳細については、「ユーザー モジュールによる CapSense 性能のチューニング」を参照してください。

2.2.2 CapSense シグマ デルタ (CSD) PLUS

図 2-8 は、センサー静電容量 (C_X) をデジタル カウントに変換する CSDPLUS 方式のブロック図を示します。CSDPLUS と CSD 方式の主な違いは使用される I_{DAC} の数です。CSDPLUS は 2 個の I_{DAC} を使用し、CSD は単一の I_{DAC} を使用します。

CSDPLUS 方式は、概念的に 2 つのブロックに分割できます。それらは静電容量を電流に変換するスイッチト キャパシタ入力、および電流をデジタル カウントに変換するシグマ デルタ コンバータです。各ブロックは以下の項で説明されます。

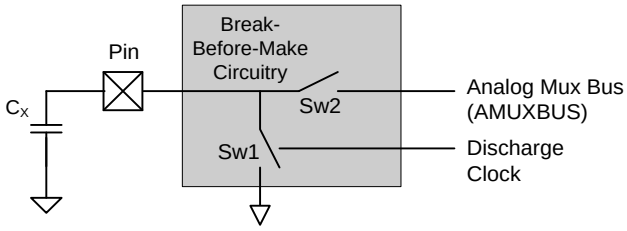
図 2-8. CSDPLUS ブロック図



2.2.2.1 スイッチト キャパシタ入力

CY8C20xx7/S デバイスの CSDPLUS 方式では、図 2-9 に示すように、C_X がスイッチト キャパシタ回路に接続されます。

図 2-9. スイッチト キャパシタ入力として設定されるピン



周波数 F_{SW} (図 2-11) の、位相が一致しない 2 つの非重複クロックが、スイッチ Sw1 および Sw2 を制御します。式 2-10 に示すように、Sw1 と Sw2 の連続切り替えにより、等価抵抗 R_S を形成します。等価抵抗 R_S の値は以下のとおりです。

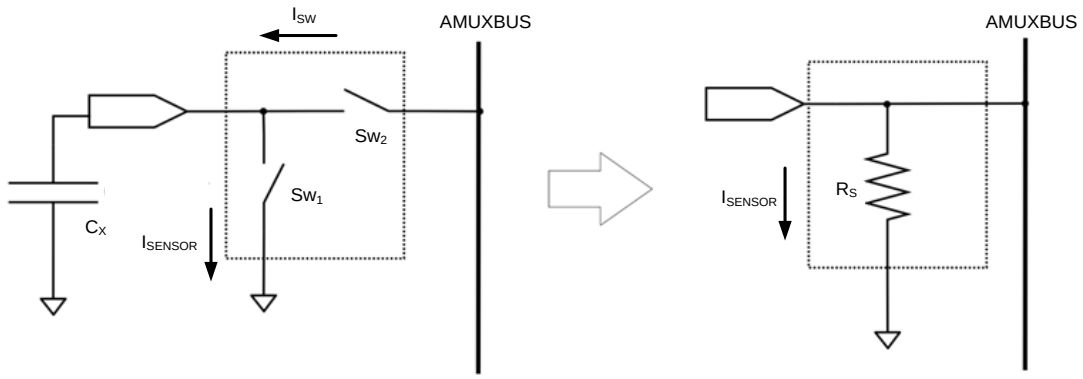
$$R_S = \frac{1}{C_X F_{SW}}$$ 式 2-7

ここで、

C_X = センサー静電容量

F_{SW} = スイッチング クロックの周波数

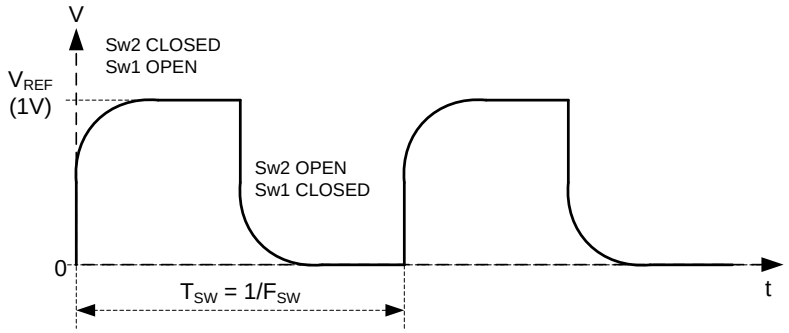
式 2-10. スイッチト キャパシタ入力が AMUXBUS から電流をシンク



シグマ デルタ コンバータは、V_{REF} 定数で AMUXBUS の電圧を維持します (この処理は「シグマ デルタ コンバータ」で説明されます)。図 2-11 は、センサー静電容量の両端における電圧波形を示します。したがって、スイッチ Sw1、Sw2 を非重複 (プリチャージ) クロックにより駆動すると、式 2-8 に示される平均電流シンク (I_{SENSOR}) が AMUXBUS で発生します。I_{SENSOR} は C_X の大きさに正比例します。

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

図 2-11. センサー静電容量の両端における電圧 (C_X)



2.2.2.2 シグマ デルタ コンバータ

シグマ デルタ コンバータは入力電流を対応するデジタル カウントに変換します。17 ページの図 2-8 に示すように、このコンバータは、シグマ デルタ変調器および 2 個の電流ソーシングのデジタル アナログ コンバータ (I_{DAC}) で構成されます。

シグマ デルタ変調器はオン／オフ 方式で 7 ビット I_{DAC} の電流を制御します。I_{DAC} は変調 I_{DAC} として知られており、本書で「I_{DAC}」または「変調 I_{DAC}」として呼ばれています。もう 1 個の 7 ビット I_{DAC} は補正 I_{DAC} として知られており、常にオンまたはオフです。本書では、この I_{DAC} は「補正 I_{DAC}」または「I_{COMP}」と呼ばれています。

17 ページの図 2-8 に示すように、シグマ デルタ コンバータは外部積分コンデンサ C_{MOD} も必要とします。C_{MOD} の推奨値は 2.2nF です。シグマ デルタ変調器は、V_{REF} で C_{MOD} 電圧を維持するために、C_{MOD} の両端における少しの電圧変動に応じて I_{DAC} 変調をオンとオフに切り替えます。

AMUX の平均電圧を安定状態の値 (V_{REF}) に維持するために、シグマ デルタ コンバータは変調ビット ストリーム デューティ サイクルを制御することで平均充電電流 (I_{DAC}) を I_{SENSOR} に一致させます。シグマ デルタ コンバータはセンサー スキャンの間ビット ストリームを保存します。その積算値は C_X に正比例した、raw カウントと呼ばれたデジタル出力値となります。

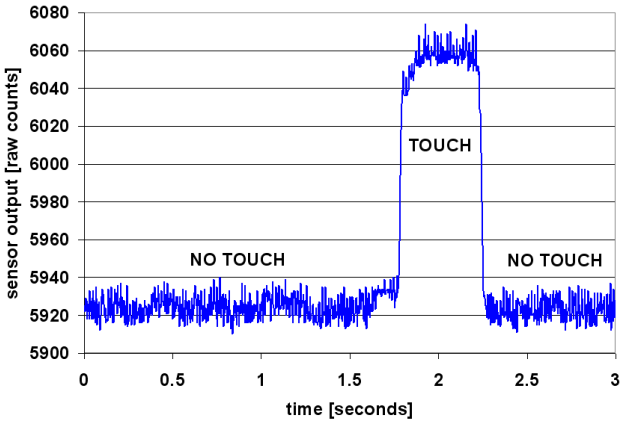
シグマ デルタ コンバータは 9 ビット～16 ビットの分解能で動作できます。「N」がシグマ デルタ コンバータの分解能であり、I_{DAC} が変調 I_{DAC} 電流の値であり、I_{COMP} が補正 I_{DAC} の電流である場合、raw カウントの近似値は以下の式から得られます。

$$\text{raw count} = (2^N - 1)\frac{V_{REF} F_{SW}}{I_{DAC}} C_X - (2^N - 1)\frac{I_{COMP}}{I_{DAC}}$$ 式 2-9

raw カウントの値は常に正であることに注意してください。そのため、I_{COMP} は常に $2^N V_{REF} F_{SW}$ 未満である必要があります。

この raw カウントは高レベルのアルゴリズムによって解釈され、センサー状態を判断し、タッチを検出します。図 2-12 は、センサーを指で触れてから離すまで連続スキャンした CSD の raw カウントをプロットしたものです。「CapSense の原理」で説明したように、指の接触により C_X が C_F 分増加し、その結果 raw カウントが比例して増加します。定常状態の raw カウント レベルの変化を事前に設定された閾値と比較することで、センサーがオン (接触) 状態であるかオフ (非接触) 状態であるかを高レベル アルゴリズムにより判定できます。

図 2-12. 指を触れたときの CSD の raw カウント



ハードウェア パラメーター (I_{DAC}、I_{COMP} や F_{SW} などの CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター) およびファームウェア パラメーター (ユーザー モジュールの高レベル パラメーター) は信頼性の高いタッチ検出のための最適値に調整する必要があります。チューニングの詳細については、「ユーザー モジュールによる CapSense 性能のチューニング」を参照してください。

2.2.3 SmartSense_EMCPLUS 自動チューニング

タッチ センシング式のユーザー インターフェースのチューニングは、正常なシステム動作と快適なユーザー操作を確実にするための重要なステップです。標準的な設計フローには、初期設計段階、システム統合中、および生産立上げ前の最終製造の微細調整でのセンサー インターフェースのチューニングが含まれます。チューニングは反復プロセスなので時間がかかることがあります。SmartSense_EMCPLUS 自動チューニングは、ユーザー インターフェースの開発サイクルを簡素化するために開発されました。このプロセスは使用方法が簡単であり、プロトタイプから大量生産までの製品開発サイクル全体からチューニング プロセスを除外して、設計サイクル時間を大幅に短縮できます。SmartSense_EMCPLUS は、電源投入時に各 CapSense センサーを自動的に調整し、その後実行中に最適なセンサー性能を監視し維持します。この技術は、プリント基板やオーバーレイによる製造のばらつきに適応し、LCD インバータ、AC ライン、スイッチング電源などのノイズ発生源を自動的に調整してノイズを取り除きます。

2.2.3.1 プロセスのばらつき

CY8C20xx7/S 用の SmartSense_EMCPLUS ユーザー モジュール (UM) は、5pF～45pF のセンサー静電容量に対応できるように設計されます (一般的にはセンサーの C_P 値は 10pF～20pF です)。各センサーの感度パラメーターは、そのセンサーの特性に基づいて自動的に設定されます。これにより、大量生産における歩留まりが上がります。理由は、指定した範囲 (5pF～45pF) 内でセンサーごとの C_P のばらつきに関係なく、すべてのセンサーから一貫性のある応答を維持できるためです。個々のセンサーの寄生容量は、プリント基板レイアウトや製造工程のばらつき、または複数の業者の供給網でベンダー間の変動によって異なることがあります。センサーの感度はその寄生容量に応じて決まります。C_P 値が高くなるとセンサー感度が低下し、その結果指のタッチ信号の振幅が低下します。場合によっては、C_P 値の変化がシステムの性能を落とし、最適なセンサー性能を得られない (過敏か感度不足になる) 結果となり、最悪の場合センサーが機能しないことがあります。いずれの場合でも、システムを再調整する必要があります。場合によっては UI サブシステムを最適化する必要があります。SmartSense_EMCPLUS 自動チューニングはこのような問題を解決します。

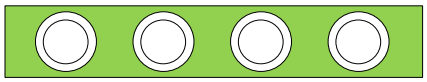
SmartSense_EMCPLUS 自動チューニングはプラットフォーム設計を可能にします。ラップトップ コンピューターの静電容量タッチ センシング式マルチメディア キーを想像してください。ボタン間の間隔はラップトップのサイズとキーボードのレイアウトで決まります。この例では、ワイド画面のマシンでは標準画面のモデルに比べてボタン間のスペースが大きくなります。ボタン間の間隔がより大きいということは、センサーと CapSense コントローラー間の配線を増やし、センサーの寄生容量を高める結果となることを意味します。つまり、同じプラットフォーム デザインであってもモデルが違えば (図 2-13 および図 2-14)、CapSense ボタンの寄生容量

は異なることがあります。これらのボタンの機能はすべてのラップトップ モデルで同じですが、センサーはモデル毎に調整する必要があります。SmartSense_EMCPLUS では、チューニングが効果的に自動で行われるため、「CapSense 入門」に記載されているプリント基板レイアウトで推奨される最良実践を活用してプラットフォーム設計ができます。

図 2-13. 21 インチ モデル用のラップトップ マルチメディア キーの設計



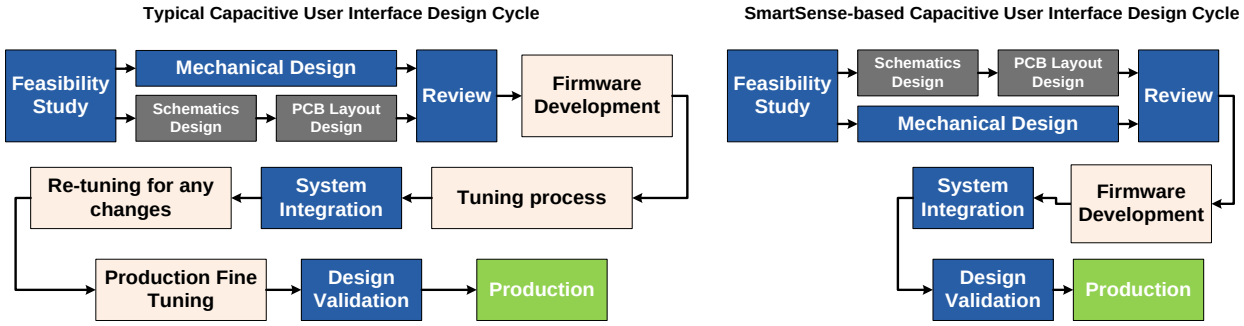
図 2-14. 機能とボタン サイズが同一の 15 インチ モデル用のラップトップ マルチメディア キーの設計



2.2.3.2 設計サイクル タイムの短縮

通常、静電容量センサーのインターフェース設計において最も時間のかかる作業は、ファームウェアの開発とセンサーのチューニングです。通常のタッチ センシング コントローラーでは、同じ設計を異なるモデルに移植したり、プリント基板やセンサー プリント基板レイアウトの機械的寸法に変更があった場合、センサーを再調整する必要があります。SmartSense_EMCPLUS 搭載の設計では、ファームウェアの開発が最小限に抑えられ、チューニングや再チューニングが必要ないため、そのような変更も問題になりません。これにより、一般的な設計サイクルがずっと速くなります。図 2-15 は、一般的なタッチ センシング コントローラーと SmartSense_EMC_PLUS ベースのデザインの設計サイクルを比較します。

図 2-15. 一般的な静電容量インターフェースの設計サイクルの比較



2.2.4 ユーザー モジュールの選択

SmartSense_EMCPLUS では、CSD と CSDPLUS UM で必要とされるチューニング プロセスは不要となります。CapSense の設計プロセスを簡素化するために SmartSense_EMCPLUS を使用することを推奨します。しかし、デバイスの消費電力を最適化する、または高い C_P があるセンサーでタッチを検出するためにセンサー パラメーター (分解能、プリスケーラなど) を制御しなければならない場合があります。その場合では、CSD／CSDPLUS UM を使用してください。

CSD と比較したときの CSDPLUS UM の利点は以下のとおりです。

- CSDPLUS UM は、指定されたセンサーの分解能 (またはスキャン時間) でより高い SNR を提供します。
- CSDPLUS UM では、センサーをスキャンするための時間は短くなります。短いスキャン時間のために、CSD UM と比べて、同じ SNR を達成するためにチューニングされる場合、CSDPLUS UM のある CapSense デバイスの平均消費電力[a]はより低いです。

しかし、センサーC_Pが 10pF 未満の場合、CSDPLUS UM は CSD UM より低い SNR を提供し、そのため CSDPLUS UM の代わりに CSD UM を使用することを推奨します。

注: I_{COMP} が「0」にセットされる場合、CSDPLUS UM の動作は CSD UM と同じようになります。

[a] デバイスがセンサーをスキャンした後にスリープ モードに入り、消費電力を減少するために周期的にウェイクアップすることが前提とされます。

3. CapSense 設計ツール

3.1 概要

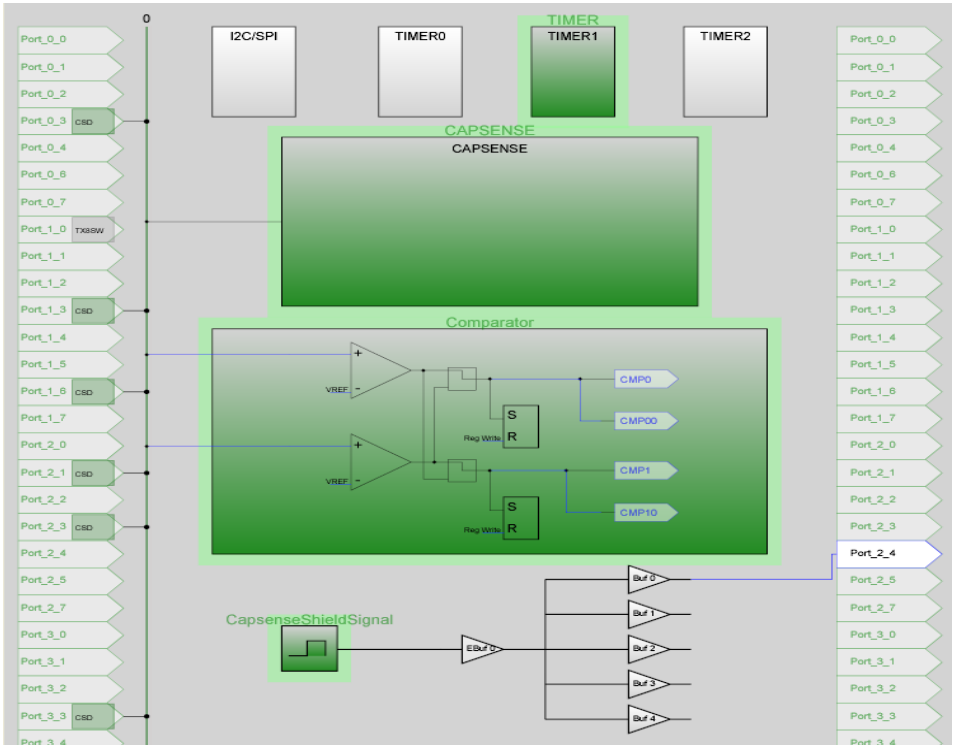
サイプレスは、CapSense 静電容量タッチ センシング アプリケーションの開発用に幅広いハードウェアおよびソフトウェアツールを提供します。注文情報は「リソース」を参照してください。

3.1.1 PSoC Designer およびユーザー モジュール

サイプレスの専用統合設計環境である PSoC Designer では、アナログとデジタル ブロックのコンフィギュレーション、ファームウェアの開発、設計のチューニングとデバッグが可能です。アプリケーションは、ドラッグ アンド ドロップ方式の設計環境でユーザー モジュールのライブラリを使用して開発されます。ユーザー モジュールは、デバイス エディタ GUI、あるいはファームウェアで特定のレジスタに書込むように設定されます。PSoC Designer には、内蔵 C コンパイラおよび組み込みのプログラマが含まれます。複雑なデザイン用には専用コンパイラがあります。

CSD および CSDPLUS ユーザー モジュールは、スイッチト キャパシタ回路、アナログ マルチプレクサ、コンパレータ、デジタル カウント機能、高レベル ソフトウェア ルーチン (API) を使用して静電容量タッチ センサーを実装します。その他のアナログとデジタル ペリフェラル用のユーザー モジュールは、I²C、SPI、TX8、タイマーなど追加機能を実装するために用意されます。

図 3-1. PSoC Designer デバイス エディタ



3.1.1.1 CapSense ユーザー モジュールの導入

PSoC Designer で新しい CY8C20xx7/S プロジェクトを作成する場合は、以下の手順を行います。

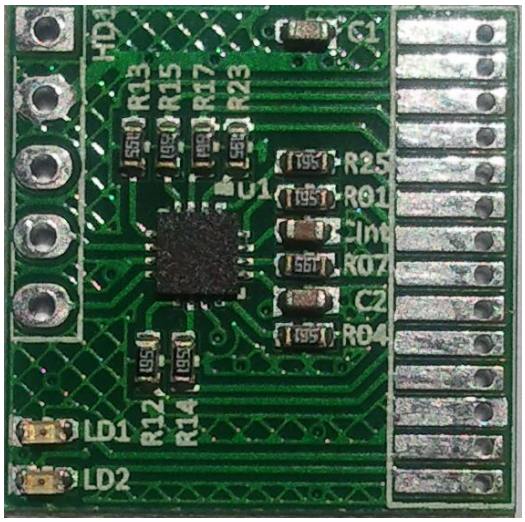
1. CY8C20xx7/S をターゲット デバイスとした新しい PSoC Designer プロジェクトを作成します。
2. CSDPLUS／SmartSense_EMCPLUS ユーザー モジュールを選択して配置します。
3. ユーザー モジュールを右クリックしてユーザー モジュール ウィザードにアクセスします。
4. ボタン センサーの数、スライダーのコンフィギュレーション、ピンの割り当ておよび接続を設定します。
5. ピンおよびユーザー モジュールのグローバル パラメーターを設定します。
6. アプリケーションを生成して、アプリケーション エディタを開きます。
7. ユーザー モジュール データシートからのサンプル コードを適用して、ボタンまたはスライダーを実装します。

PSoC Designer プロジェクトの作成とユーザー モジュール ウィザードのコンフィギュレーションの詳細な手順については、各々のユーザー モジュールのデータシートを参照してください。CapSense ユーザー モジュールのサンプル コードについては、「サンプル コード」を参照してください。

3.1.2 CY8C20xx7/S QuietZone スターター キット

CY8C20xx7/S QuietZone スターター キットは単純なプラグイン ハードウェアを含みますが、これにより、試作は容易になります。キットはサイプレスのモジュール パートナーArtaFlex のウェブサイト (www.artaflexmodules.com/quietzone) から入手できます。

図 3-2. CY8C20xx7/S QuietZone スターター キット

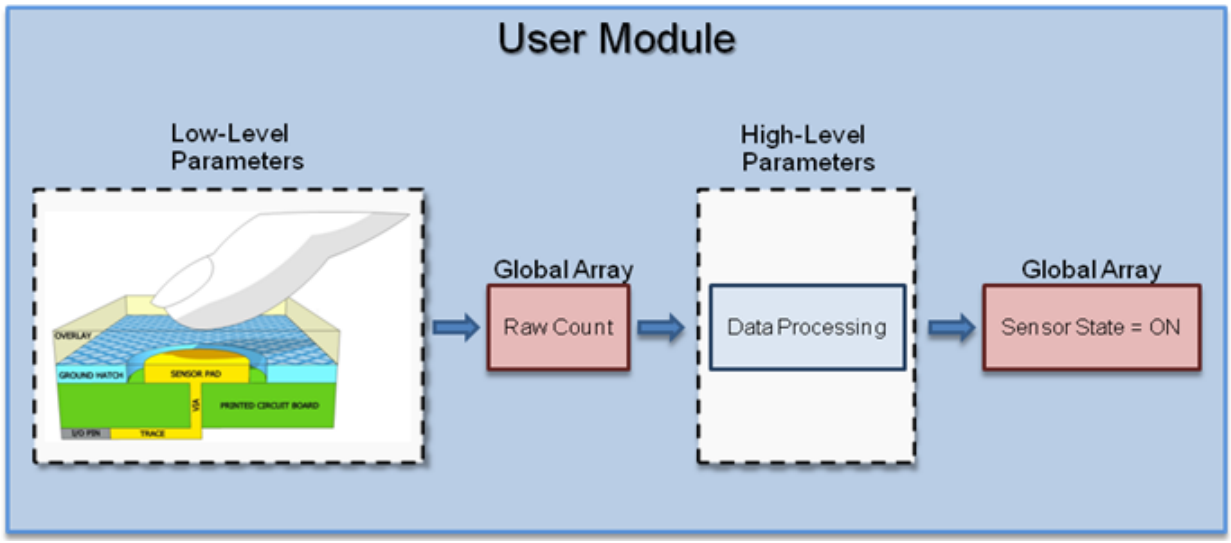


3.1.3 CapSense データ表示ツール

CapSense の設計中には、チューニングまたはデバッグ用に CapSense データ (raw カウント、ベースライン値、差分カウントなど) の監視が必要となる場合が何度もあります。CapSense データ表示ツールには MultiChart と Bridge Control Panel の 2 つがあります。これらのツールは、「AN2397 – CapSense データ表示ツール」アプリケーション ノートで詳しく説明されています。

3.2 ユーザー モジュール概要

図 3-3. ユーザー モジュールのブロック図



ユーザー モジュールには、物理的な検知からデータ処理までの全 CapSense システムが含まれます。ユーザー モジュールの動作は、さまざまなパラメーターを使用して決定します。パラメーターはセンシング システムの異なる部分に影響し、グローバルアレーを使用して相互に交信する低レベルと高レベルのパラメーターに分けられます。

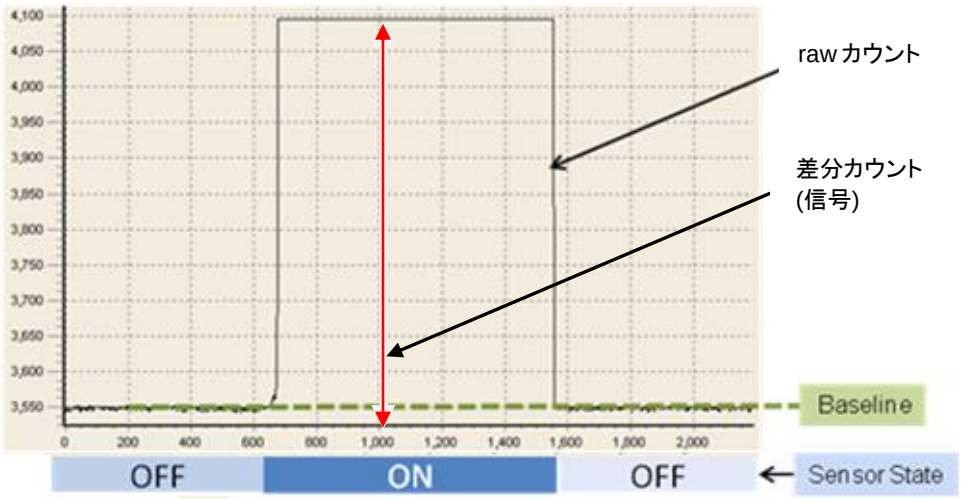
スキャン センサーの速度と分解能などの低レベル パラメーターは、物理層でのセンサー動作を決定し、静電容量から raw カウントへの変換に関連しています。低レベル パラメーターはセンシング方式によって異なり、それぞれ「CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター」および「SmartSense_EMCPLUS ユーザー モジュールのパラメーター」で説明されます。

デバウンス カウントやノイズ閾値といった高レベル パラメーターでは、raw カウントを処理する方法を定義して、センサーのオン／オフ状態やスライダー上の指の予想位置などの情報を生成します。これらのパラメーターはすべてのセンシング方式で同じであり、「ユーザー モジュールの高レベル パラメーター」で説明されます。

3.3 CapSense ユーザー モジュールのグローバル アレイ

CapSense ユーザー モジュール パラメーターについて習得する前に、CapSense システムで使用する特定のグローバルアレイに精通しておく必要があります。これらのアレイは手動で変更できませんが、デバッグ用に検査できます。

図 3-4. raw カウント、ベースライン値、差分カウントおよびセンサー状態



3.3.1 raw カウント

CapSense コントローラーの内蔵ハードウェア回路はセンサーの静電容量 (C_X) を測定します。測定値は、ユーザー モジュールの API UMname_ScanSensor()を呼び出すと、raw カウントと呼ばれるデジタル形式で回路に保存されます。UMname は CSD,CSDPLUS または SmartSenseEMC_PLUS になります。

センサーの raw カウントは、そのセンサー静電容量に比例します。センサー静電容量の値が増加すると、raw カウントが増加します。

センサーの raw カウント値は、*UMname*_waSnsResult[] 整数アレイに保存されます。このアレイは *UMname.h* のヘッダファイルで定義されます。

3.3.2 ベースライン

ベースラインは、センサーの寄生容量 (C_P) に対応する raw カウント値として見なされます。温度や湿度などの段階的な環境変化は、センサーC_P (そのため C_X) に影響を与え、その結果 raw カウントが変動します。

ユーザー モジュールは複雑なベースライン アルゴリズムを使用してそのような変化を補正します。アルゴリズムでは、ベースライン変数を使用してこの補正を行います。ベースライン変数は、raw カウント値の段階的な変動を追跡します。ベースライン変数は基本的に、raw カウント値が入力されるデジタル ローパス フィルターの出力を保持します。

ベースライン アルゴリズムは、ユーザー モジュール API *UMname*_UpdateSensorBaseline によって実行されます。*UMname* は CSD、CSDADC、または SmartSense_EMC_PLUS となります。

センサーのベースライン値は、*UMname*_waSnsBaseline[] 整数アレイに保存されます。このアレイは *UMname.h* のヘッダ ファイルで定義されます。

3.3.3 差分カウント (信号)

差分カウントはセンサーの信号としても知られ、センサーの raw カウントとベースライン値間のカウント差として定義されます。センサーがアクティブでないとき、差分カウントはゼロです。タッチしてセンサーをアクティブ化すると、差分カウント値はプラスになります。

センサーの差分カウント値は *UMname*_waSnsDiff[] (*UMname* は CSD、CSDPLUS または SmartSense_EMC_PLUS になる) 整数アレイに保存されます。このアレイは *UMname.h* のヘッダ ファイルで定義されます。

差分カウント変数はユーザー モジュールの API *UMname*_UpdateSensorBaseline() により更新されます。

3.3.4 センサー状態

センサー状態は物理的センサーのアクティブ／非アクティブな状態を示します。センサー状態は、指を触れると 0 から 1 に変わり、指を離すと 0 に戻ります。

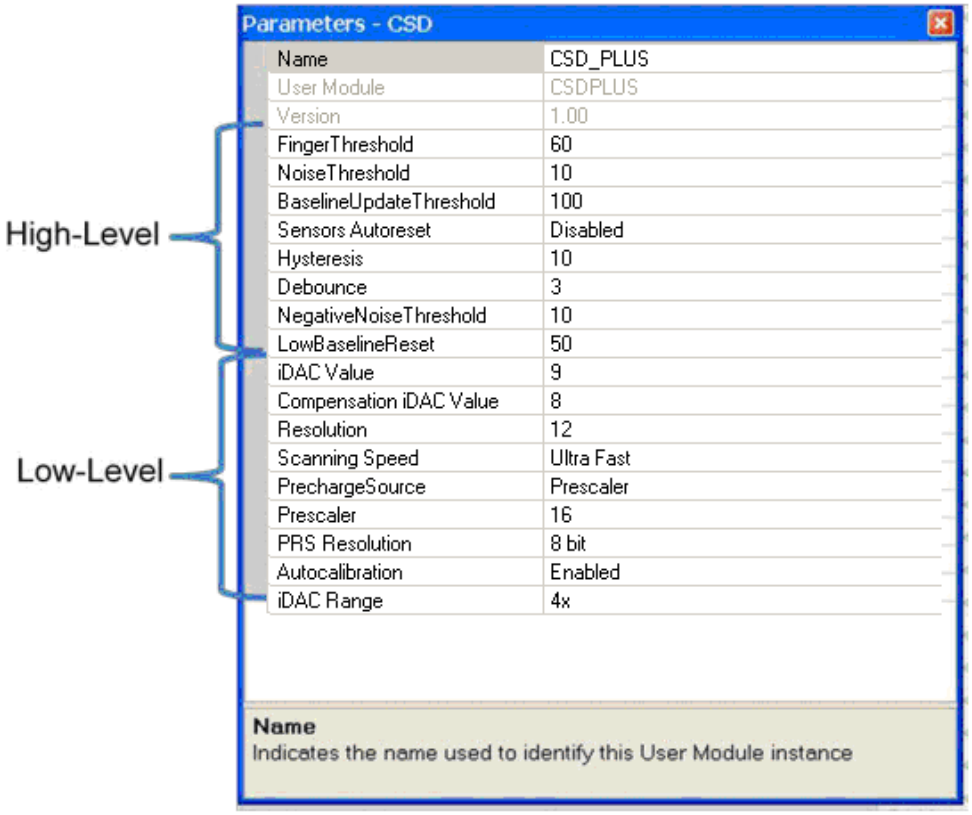
センサー状態は *UMname*_baSnsOnMask[] アレイ (*UMname* は CSD、CSDPLUS または SmartSense_EMCPLUS) というバイト アレイに保存されます。このアレイは *UMname.h* のヘッダ ファイルで定義されます。各アレイ要素は 8 つの隣接したセンサーの状態を保存します。

センサー状態はユーザー モジュールの API *UMname*_bIsAnySensorActive() で更新されます。

3.4 CSD／CSDPLUS ユーザー モジュール パラメーター

CSD／CSDPLUS ユーザー モジュールのパラメーターは高レベルと低レベルのパラメーターに分類されます。CSDPLUS ユーザー モジュールのパラメーターの一覧およびそれらの分類については、図 3-5 を参照してください。CSD と CSDPLUS UM パラメーター ウィンドウの唯一の違いは、CSD UM が補正 iDAC 値を持っていないということです。

図 3-5. PSoC Designer – CSDPLUS パラメーター ウィンドウ



3.4.1 ユーザー モジュールの高レベル パラメーター

3.4.1.1 指閾値

指閾値パラメーターはユーザー モジュールで使用し、センサーのアクティブ／非アクティブ状態を判定します。センサーの差分カウントが閾値より大きい場合、センサーはアクティブと判定されます。この定義では、ヒステリシス レベルが「0」に設定され、デバウンスが「1」に設定されたことを前提としています。

値の範囲は 3～255 です。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.2 ヒステリシス

ヒステリシスの設定により、センサーのオン状態がシステム ノイズによるチャタリングを防ぎます。ヒステリシスの関数は式 3-1 で計算されます。この式で、デバウンスが「1」に設定されていることを前提とします。

図 3-6. ヒステリシスが「0」に設定されている場合のセンサー状態対差分カウント

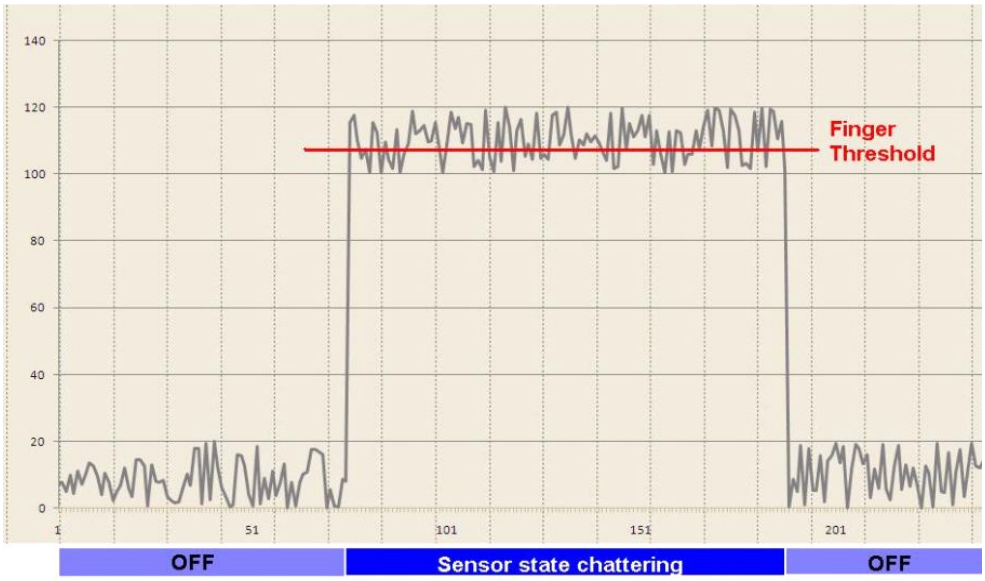
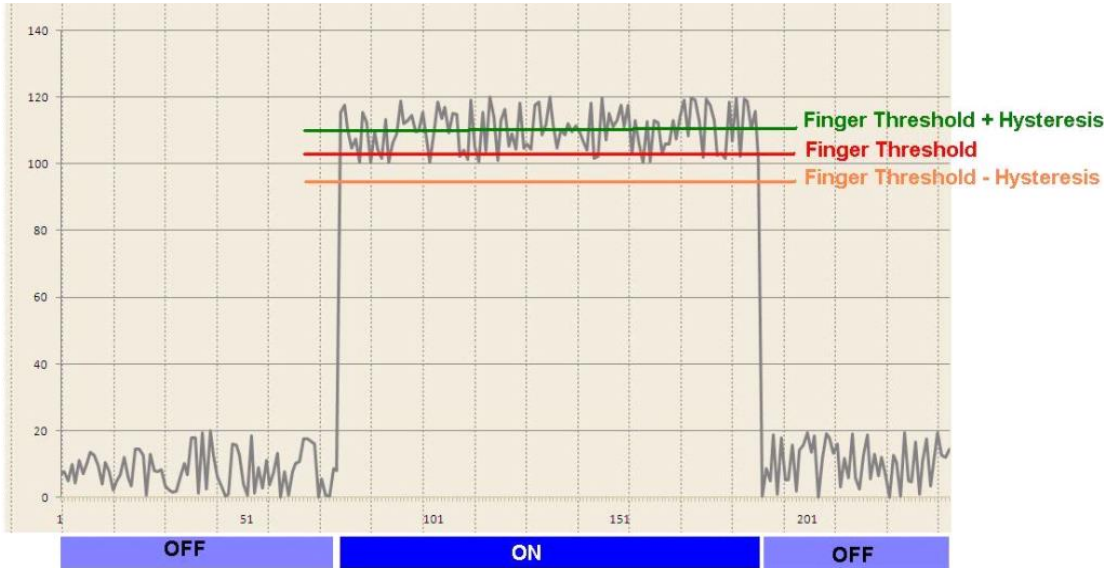


図 3-7. ヒステリシスが「0」以外に設定されている場合のセンサー状態対差分カウント



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

$$\text{ifDifferenceCount} \leq \text{FingerThreshold} - \text{Hysteresis}, \text{SensorState} = \text{OFF} \qquad \text{式 3-1}$$

値の範囲は 0～255 です。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.3 デバウンス

デバウンス パラメーターは、raw カウントのスパイクのためセンサー状態がオフからオンに変わることを防ぎます。センサー状態がオフからオンになるために、特定のサンプル数の間、差カウントは [指閾値 + ヒステリシス レベル] より大きい値を維持する必要があります。

値の範囲は 1～255 です。「1」に設定すると、デバウンスは起こりません。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.4 ベースライン更新閾値

前述したように、ベースライン変数は raw カウント値の段階的な変動を追跡します。つまり、ベースライン変数は基本的に、raw カウント値が入力されるデジタル ローパス フィルターの出力を保持します。ベースライン更新閾値パラメーターは、このローパス フィルターの時定数調整に使用されます。

ベースライン更新閾値はこのフィルターの時定数に正比例します。ベースライン更新閾値が高いほど、時定数は高くなります。

値の範囲は 0～255 です。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.5 ノイズ閾値

ユーザー モジュールはノイズ閾値を使用して raw カウント内のノイズ カウントの上限を解釈します。個々のセンサーについては、raw カウントがベースラインを上回り、これらの差がこの閾値を上回る場合、ベースライン更新アルゴリズムは停止されます。

スライダー センサーの場合、差分カウントがノイズ閾値を超えるとセントロイドの計算は停止されます。

値の範囲は 3～255 です。ユーザー モジュールが正常に動作するために、ノイズ閾値を [指閾値 － ヒステリシス] より高く設定しないでください。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.6 負のノイズ閾値

負のノイズ閾値は、ユーザー モジュールが raw カウント内のノイズ カウントの下限を知る際に役立ちます。raw カウントがベースラインを下回り、これらの差がこの閾値を上回る場合、ベースライン更新アルゴリズムは停止されます。

値の範囲は 0～255 です。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.7 低ベースライン リセット

低ベースライン リセット パラメーターは、負のノイズ閾値パラメーターと併用されます。サンプル カウント値が特定のサンプル数の間 [ベースライン － 負のノイズ閾値] を下回ると、ベースラインは新しい raw カウント値に設定されます。これは基本的に、ベースライン値をリセットする必要がある異常に低いサンプルをカウントします。サンプル値は、起動時に指が置かれている状態の補正をするために使用されます。

値の範囲は 0～255 です。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.1.8 センサー自動リセット

センサー自動リセット パラメーターにより、ベースライン値は連続して更新されるのか、それとも差分カウントがノイズ閾値を下回った場合にのみ更新されるかが決定されます。

センサー自動リセットが有効にされると、ベースライン値は常に更新されます。この設定により、センサーの最大時間は制限されますが、センサーに何も触れないのに raw カウントが誤って上昇した場合に、センサーが恒久的にオンになることを防ぎます。この突然の上昇の原因には、大幅な電源電圧の変化、高エネルギーRF ノイズ ソース、または急速な温度変化があります。

センサー自動リセットが無効にされると、ベースライン値は差分カウントがノイズ閾値パラメーターを下回った場合にのみ更新されます。

値は、Enabled (有効) と Disabled (無効) です。

推奨値は、「高レベル パラメーターの設定」を参照してください。

3.4.2 CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター

CSD／CSDPLUS ユーザー モジュールには、高レベル パラメーターに加えて低レベル パラメーターがあります。これらは CSD／CSDPLUS センシング方式固有のパラメーターであり、どのように raw カウント データをセンサーから取得するかを決定します。

3.4.2.1 I_{DAC} 値

I_{DAC} パラメーターは、容量測定 (C_X の変更による raw カウントの変更) の範囲と感度を設定します。I_{DAC} 値が高くなると、範囲は高くなりますが、感度は低くなります。以下のように説明します。

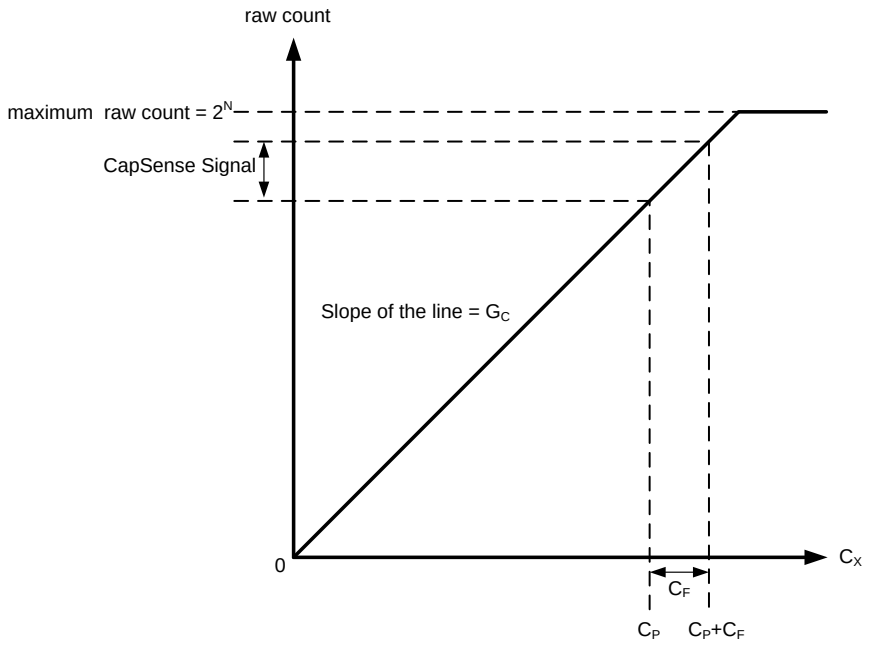
式 2-9 のとおりに、以下のように raw カウントは C_X および I_{DAC} に対応します。

$$\text{raw count} = G_C C_X - (2^N - 1)\frac{I_{COMP}}{I_{DAC}} \qquad \text{式 3-2}$$

G_C は静電容量-デジタル変換利得であり、$(2^N - 1)\frac{V_{REF} F_{SW}}{I_{DAC}}$ に等しいです。

補正 I_{DAC} 値、すなわち I_{COMP} がゼロの場合を考慮してみましょう。この場合では、raw カウントと C_X との関係 (式 3-2) は図 3-8 で示されます。

図 3-8. raw カウント対センサー静電容量



指がセンサーに触れたときの raw カウントの変化は CapSense 信号と呼ばれます。図 3-9 は変換利得に対して信号の値がどのように変化するかについて示します。

図 3-9. 異なる変換利得に対する信号値

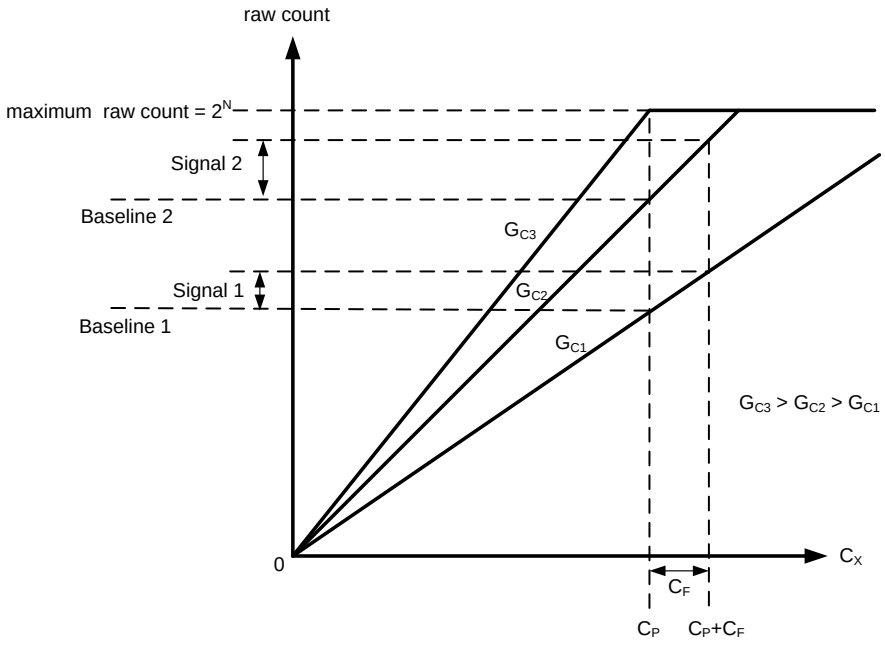
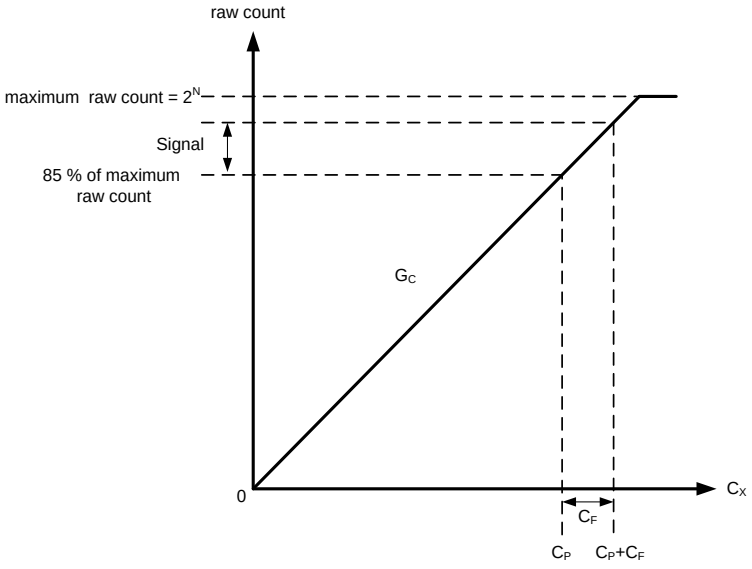


図 3-9 は 3 つの変換利得値 (G_{C3}、G_{C2} および G_{C1}) に該当する 3 つのプロットを示します。変換利得の増加により、信号の値が高くなります。しかし変換利得の増加にしたがって、C_P に該当する raw カウントも raw カウントの最大値 (2^N) の方へ近づきます。非常に高い利得値に対しては、raw カウントは G_{C3} のプロットが示すように飽和します。そのため、raw カウントの飽和を回避しながら、信号の良い値を得るように変換利得を調整する必要があります。図 3-10 に示すように、C_P に該当する raw カウントが raw カウントの最大値の 85%になるよう利得を調整することを推奨します。

図 3-10. I_{COMP} がゼロの場合の推奨チューニング



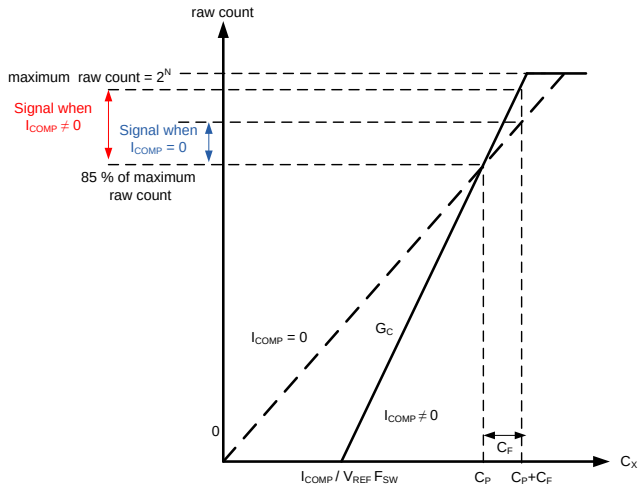
このパラメーターは、対応する API 関数を用いて実行時に変更できます。

値の範囲は 1～127 です。

3.4.2.2 補正 I_{DAC} 値

補正 I_{DAC} パラメーターは、raw カウント対 C_X のグラフのオフセットをセットします。補正 I_{DAC} は、C_P の高いセンサーに対し高い感度を達成するために使用できます。図 3-11 は、I_{COMP} が非ゼロのときのセンサー容量に対する raw カウントのプロットを示します。

図 3-11. raw カウント対 C_X (I_{COMP} が非ゼロのとき)



補正 I_{DAC} 値の増加は C_X 軸のインターセプトを $\left(\frac{I_{COMP}}{V_{REF}\ F_{SW}}\right)$ に増やします。

raw カウントを再度最大値の 85%にすると、ラインのスロープも増加します。したがって、補正 I_{DAC} 値が増加すると、信号は増加します。

補正 I_{DAC} を使用する場合は、ノイズ カウントの変動のため、信号の増加に比例して SNR が増加しないことがあることに注意してください。

追加変数 (I_{COMP}) が含まれるので、デュアル I_{DAC} モードはチューニング プロセスを複雑にします。しかし、このモードを使用して信号を増加させます。詳細については、「CSD／CSDPLUS ユーザー モジュールのチューニング」を参照してください。

このパラメーターは、対応する API 関数を用いて実行時に変更できます。

値の範囲は 1～127 です。

3.4.2.3 分解能

このパラメーターではスキャン分解能をビット数の単位で設定します。ビット数が N の場合、スキャン分解能の最大 raw カウントは 2^N–1 です。分解能を高くすると、感度が高くなりますが、スキャン時間が長くなり、その結果応答速度が低下します。

値の範囲は 9～16 です。

表 3-1. 分解能とスキャン速度

分解能	個々のボタンのスキャン速度 (µs)			
	超高速	高速	通常	低速
9	57	78	125	205
10	78	125	205	380
11	125	205	380	720
12	205	380	720	1400
13	380	720	1400	2800
14	720	1400	2800	5600
15	1400	2800	5600	11000
16	2800	5600	11000	22000

3.4.2.4 スキャン速度

このパラメーターは、シグマ デルタ コンバータへのクロックを設定することによりセンサー スキャン速度を設定します。スキャン速度が速い場合応答時間が短くなりますが、スキャン速度が低い場合以下のメリットがあります。

- 向上した SNR
- 電源と温度変化に対する改善された耐性
- システム割り込み遅延の必要性が減少し、より長い割り込みを処理可能

値は「Ultra Fast」(超高速)、「Fast」(高速)、「Normal」(通常)、「Slow」(低速) です。

3.4.2.5 シールド電極出力

シールド電極は寄生容量を減少させ、耐水性を達成するために使用します。シールド電極の詳細については、本書の「耐液性デザイン上の注意事項」節および「CapSense 入門」の「シールド電極およびガード センサー」節を参照してください。このパラメーターはシールド電極の出力のルーティング先を選択します。

値は P0[0]、P1[2]、P0[2]、P2[2]および P2[4]です。

3.4.2.6 プリチャージ源

このパラメーターは、「スイッチト キャパシタ入力」節でプリチャージ クロックとして呼ばれるプリチャージ スイッチ用のクロック ソースを選択します。

値は PRS およびプリスケーラです。プリスケーラの場合、スイッチング クロック周波数は IMO／プリスケーラにセットされます。PRS の場合、分周された IMO クロックは疑似乱数発生器に渡され、スプレッド拡散クロックが提供されます。PRS が幅広い範囲でスイッチング周波数を平均化するため、ほとんどの場合 PRS 源を使用すると、EMI 耐性が改善され、放射を減少させます。

3.4.2.7 プリスケーラ

このパラメーターではプリスケーラ比を設定し、プリチャージ スイッチの出力周波数を決定します。このパラメーターは PRS 出力周波数にも影響を与えます。

値は 1、2、4、8、16、32、64、128、256 です。

3.4.2.8 PRS 分解能

このパラメーターは PRS シーケンスの長さを変更します。

スキャンに時間がほとんどかからない場合、過度のノイズを回避するために 8 ビット PRS を使用する必要があります。スキャン時間は分解能パラメーター (PRS 分解能と混同してはならない) により定義されます。スキャン時間が 380µs 以下の場合、PRS 分解能を 8 ビットに設定し、スキャン時間が 380µs を超える場合、PRS 分解能を 12 ビットに設定する必要があります。デフォルト設定は 8 ビットです。

3.4.2.9 自動校正

自動校正が有効になった場合、変調および補正 I_{DAC} は自動的に校正され、raw カウント ベースラインは 2^N のおおよそ 85%に設定されます (「N」はセンサーの分解能設定です)。自動校正を有効にすると、I_{DAC} および I_{COMP} のデバイス エディタ設定は無効になります。

自動校正が無効になった場合、raw カウント値はデバイス エディタで設定した I_{DAC} 範囲、I_{DAC} 値、分解能、センサー静電容量、IMO 周波数、プリスケーラ、プリチャージ源および V_{REF} パラメーターに基づいて計算します。

自動校正は ROM と RAM リソースを消費し、起動時間を増加させます。自動校正は I_{DAC} の範囲値を自動的に選択しません。校正の終了後の raw カウント値が分解能範囲の半分より少ない場合、I_{DAC} 範囲を広げるか、プリチャージ周波数を減らす必要があります。自動校正の目的は、機能レベルが下限ぎりぎりのコンフィギュレーションを改善することです。

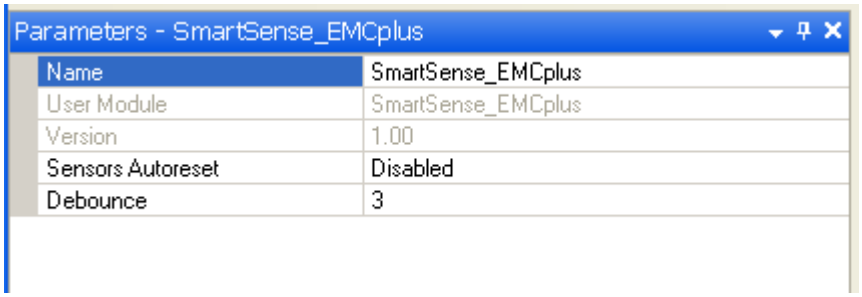
3.4.2.10 I_{DAC} 範囲

I_{DAC} 範囲パラメーターでは I_{DAC} 電流出力の範囲を設定します。たとえば、8x を選択すると I_{DAC} 出力が 4x 範囲の 2 倍に設定されます。

値は 4x および 8x です。

3.5 SmartSense_EMCPLUS ユーザー モジュールのパラメーター

図 3-12. PSoC Designer SmartSense_EMCPLUS のパラメーター

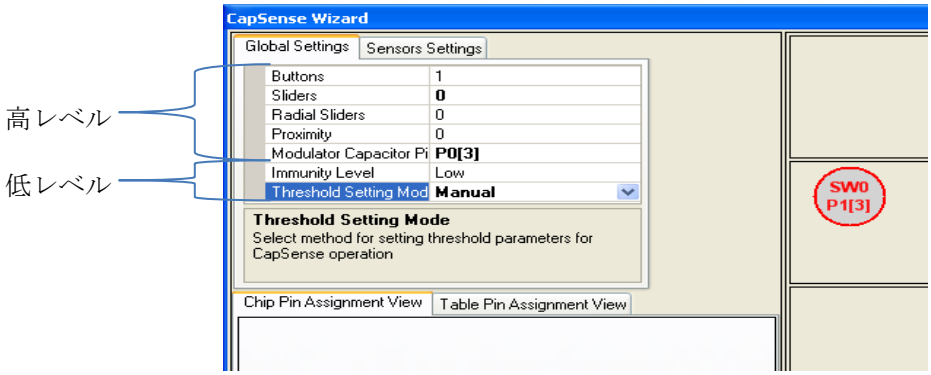


3.5.1.1 耐性レベル

このパラメーターではユーザー モジュールの外部ノイズに対する耐性レベルを定義します。高耐性レベルを選択すると、外部ノイズに対する最高の耐性が得られます。中耐性レベルを選択すると、中レベルの耐性が得られます。低耐性モードと比べて、中耐性モードはセンサーのスキャン時間とセンサー実装用の RAM メモリ使用量が 2 倍になり、高耐性モードはスキャン時間と RAM メモリ使用量が 3 倍になります。

値は Low (低)、Medium (中)、High (高) です。

図 3-13. PSoC Designer SmartSense_EMCPLUS のグローバル設定

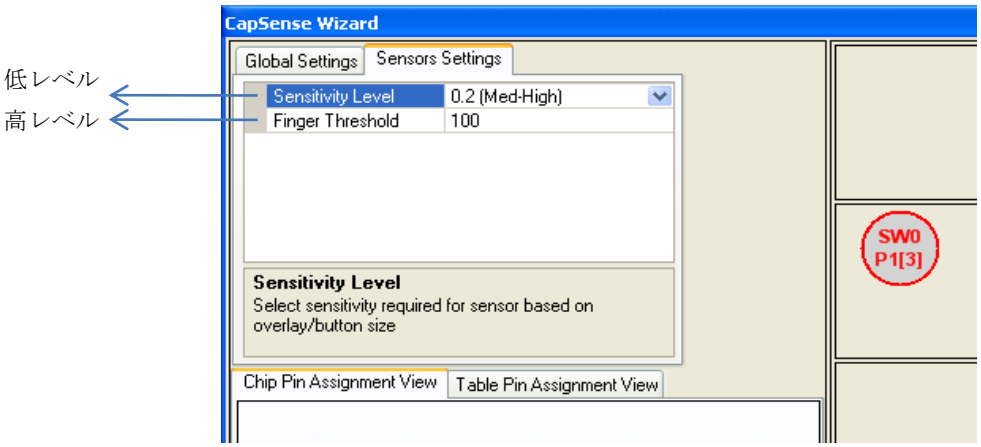


3.5.1.2 閾値設定モード

手動閾値モードを選択すると、各センサーの指閾値設定を柔軟に行えます。自動閾値モードを選択すれば、SmartSense_EMCPLUS ユーザー モジュールは個々のセンサーの閾値を自動的に設定します。自動閾値モードでは、手動閾値モードより多くの RAM が消費されます。

値は Manual (手動) と Automatic (自動) です。

図 3-14. PSoC Designer SmartSense_EMC_PLUS のセンサー設定



3.5.1.3 センサー感度

このパラメーターはセンサー感度の増減に使用されます。

値は 0.1pF、0.2pF、0.3pF、0.4pF です。

4. ユーザー モジュールによる CapSense 性能のチューニング

最適なユーザー モジュール パラメーター設定は基板レイアウトやボタン寸法、オーバーレイ素材、アプリケーションの要件によって異なります。これらの要因は「設計上の注意事項」で説明します。チューニングとは、堅牢で信頼性の高いセンサー動作のための最適なパラメーター設定を指定するプロセスです。

4.1 一般的な注意事項

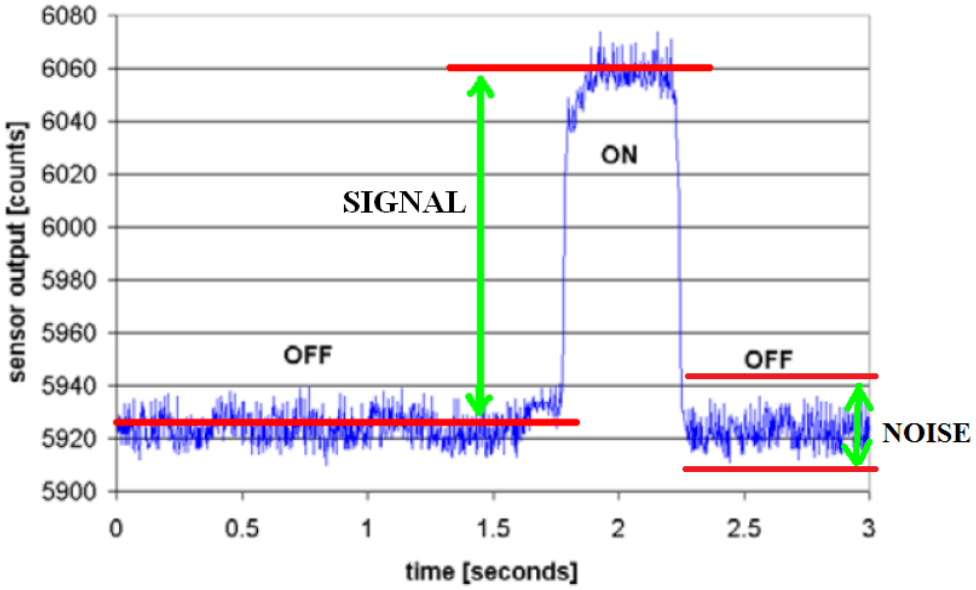
4.1.1 信号、ノイズ、および SNR (信号対ノイズ比)

適切に調整された CapSense システムでは、オンとオフのセンサー状態が確実に識別されます。この性能レベルを達成するには、CapSense 信号は CapSense ノイズよりはるかに大きくする必要があります。CapSense 信号は、信号対ノイズ比 (SNR) という量により CapSense ノイズと比較されます。CapSense の SNR についての説明を行う前に、まずタッチセンシングのコンテキストにおける信号とノイズを定義します。

4.1.1.1 CapSense 信号

CapSense 信号とは図 4-1 に示すように、センサー上に指を置いたときのセンサーraw カウントの変化です。センサーの出力は、センサー静電容量を追跡する値を表示するデジタル カウンターです。この例では、センサーに指を置かない場合の平均レベルは 5925 カウントです。センサーに指を置いた場合の平均出力は 6060 カウントに増えます。CapSense 信号は指触によるカウントの変化を追跡するため、信号 = 6060 – 5925 = 135 カウントとなります。

図 4-1. CapSense 信号とノイズ



4.1.1.2 CapSense ノイズ

図 4-1 に示すように、CapSense ノイズは指を触れない場合、センサー応答の最小から最大までの変化です。この例では、指を置かない場合の出力波形は、最小で 5912 カウント、最大で 5938 カウントです。ノイズとはこの波形の最小値と最大値の差分であるため、ノイズ = 5938 – 5912 = 26 カウントとなります。

4.1.1.3 CapSense SNR

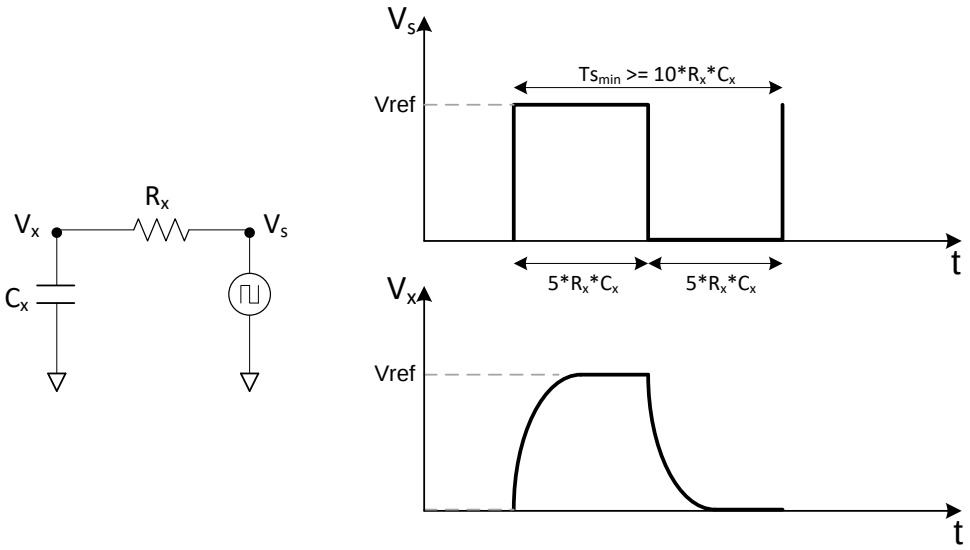
CapSense SNR とは信号とノイズの単純比です。この例では、信号が 135 カウントであり、ノイズが 26 カウントである場合、SNR は 135:26 で、5.2:1 に約分されます。CapSense の推奨最小 SNR は 5:1 で、すなわち信号値がノイズ値より 5 倍大きいです。一般的に、フィルターはノイズを減少させるためにファームウェアで実装されます。詳細は、「ソフトウェア フィルター」を参照してください。

4.1.2 充電／放電速度

チューニング プロセスにおいて最高感度を達成するには、各サイクルの間センサー コンデンサを完全に充電および放電しなければなりません。充電／放電パスは、CSDPLUS ユーザー モジュールではプリチャージ クロックと呼ばれるユーザーモジュール パラメーターにより設定された速度で、2 つの状態間で切り替わります。

充電／放電パスには、電荷移動を減速させる直列抵抗が含まれます。電荷移動の変化速度は図 4-2 に示すように、センサー コンデンサ (C) と直列抵抗 (R) から計算する RC 時定数で決まります。

図 4-2. 充電／放電波形



充電／放電速度を、RC 時定数に適合するレベルに設定します。一般的に、各遷移には 5 RC の期間を合わせて、毎期間 (充電 1 回、放電 1 回) 2 回の遷移とする必要があります。最短周期と最大周波数の式は以下のとおりです。

$Ts_{Min} = 10 \times R_X C_X$ 式 4-1

$fs_{Max} = \frac{1}{10 \times R_X C_X}$ 式 4-2

たとえば、直列抵抗に 560Ω の外部抵抗と最大 800Ω の内部抵抗が含まれており、センサー容量が標準値であることを前提とします。

R_X = 1.4kΩ

C_X = 24pF

この例では、時定数の値と最大フロントエンド スイッチング周波数は次のようになります。

Ts_{min} = 0.34µs

fs_{max} = 3MHz

4.1.3 ベースライン更新閾値の検証の重要性

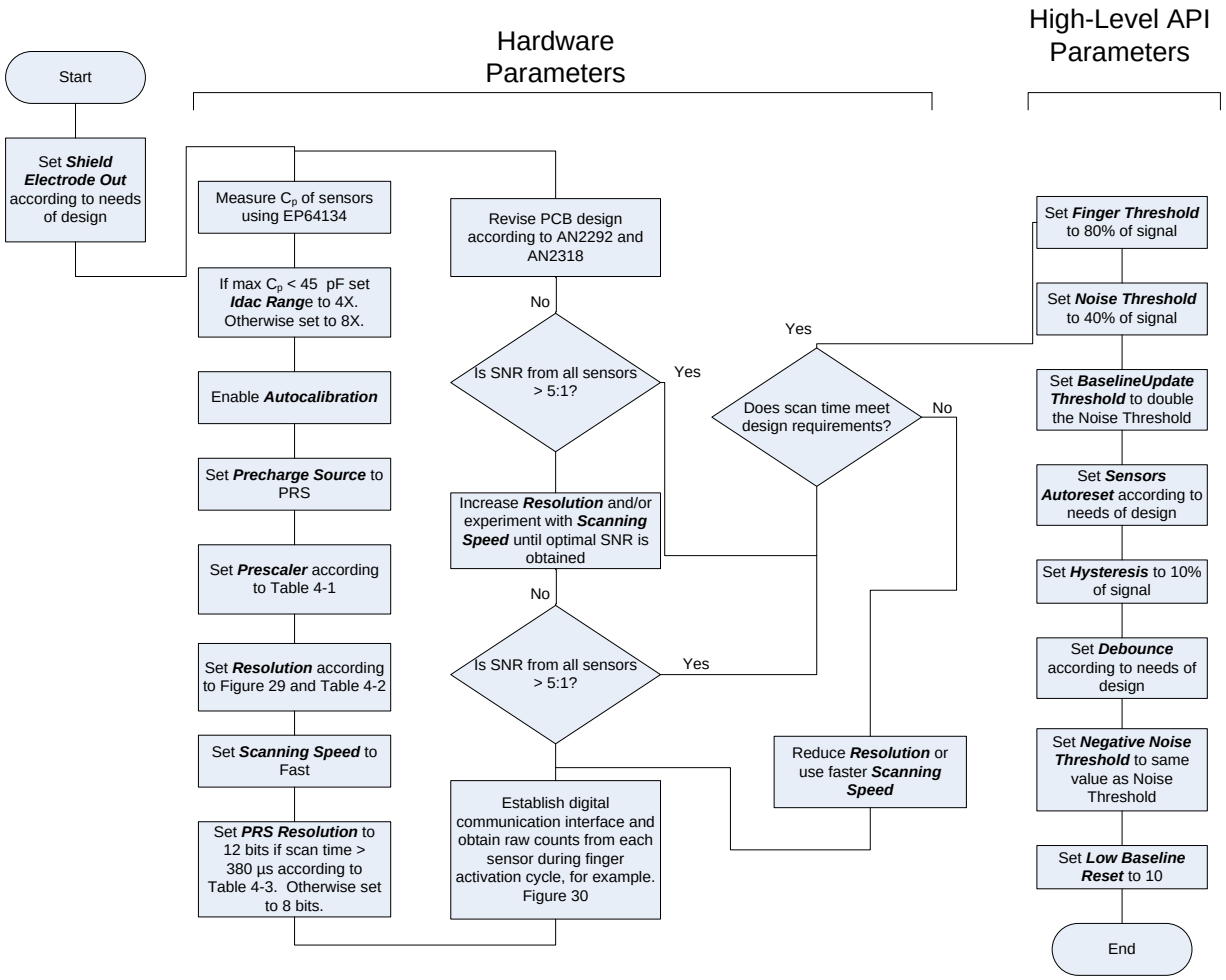
温度と湿度の両方は平均カウント数の経時的変動の原因となります。ベースラインは CapSense 測定の参照カウント レベルであり、環境影響への補正に重要な役割を果たします。指あり状態と指なし状態などの高レベル決定は、ベースラインで設定された参照レベルに基づきます。各センサーにはそれぞれ別の寄生容量があるため、各静電容量センサーには独自のベースラインがあります。

ベースラインは、ベースライン更新閾値パラメーターで設定された速度でカウントの変化を追跡します。更新速度が目的に適合するようにしてください。更新速度が速すぎると、ベースラインは指によるすべての変化を補正するため、動いている指は検出されません。更新速度が遅すぎると、比較的にゆっくりした環境変化が指と間違えられます。開発時に、ベースライン更新閾値設定を検証する必要があります。

4.2 CSD／CSDPLUS ユーザー モジュールのチューニング

図 4-3 は、CSD／CSDPLUS UM のパラメーターのチューニング プロセスを示すフロー チャートです。CSD／CSDPLUS UM のパラメーターは、低レベル (ハードウェア) パラメーターと高レベル API パラメーターの 2 つカテゴリに大きく分類できます。これらのカテゴリのパラメーターは、さまざまな方法で静電容量センシング システムの動作に影響します。ただし、ハードウェア パラメーター設定と多くの高レベル パラメーター設定で決まったように各センサー感度間には補完関係があります。ハードウェア パラメーターを変更する場合は、対応する高レベル パラメーターがそれに応じて調整されるようにこの点を考慮する必要があります。CSD／CSDPLUS ユーザー モジュール パラメーターのチューニングは、常にハードウェアパラメーターから開始する必要があります。

図 4-3. CSD／CSDPLUS ユーザー モジュールのチューニング



ハードウェア パラメーターは、個々のセンサーの物理的静電容量をデジタル コードに変換するために CSD／CSDPLUS 方式によって使用されるハードウェアを設定します。本節はこれらのパラメーターを説明し、各パラメーターをシステム特性やその他のパラメーターに基づいて調整する方法を説明します。

デフォルトでは、ハードウェア パラメーターはデザインのすべての CapSense センサーに適用されるグローバル設定です。個々のセンサーの総寄生容量 (C_P)、センサー感度、またはその両方が幅広い範囲で変化するデザインでは、すべてのセンサーに適合するグローバル ハードウェア パラメーター設定はないことがあります。そのような場合、SetIdacValue(i)、SetPrescaler(i) および SetScanMode(i) API 関数を使用して、個々のセンサーの、それぞれのハードウェア パラメーターの設定が可能です ((i) は ScanSensor(i) API 関数を呼び出す前のセンサー インデックスです)。

表 4-1 と表 4-3 は、センサーの C_P に基づいて、いくつかの主要なハードウェア パラメーターのチューニングの推奨事項を示します。C_P 値は完成品の PSoC やプリント基板レイアウトの特性、その他の部品の近接度に依存します。そのため、C_P はシステムの最終組み立て状態で、すなわちシステムと同じ筐体および同じカバーで、最初の位置で測定する必要があります。C_P 測定の最良の方法は、CapSense コントローラー サンプル コード デザイン ガイドに記載されている「Measuring Absolute Sensor Capacitance with a CY8C20xx6A CapSense Controller」サンプル コードを使用することです。本プロジェクトでは、PSoC を使用したシステム内の各センサーの絶対静電容量を測定するため、C_P に影響するすべての要因を検討することになります。設定と使用の説明は、サンプル コード関連文書を参照してください。

4.2.1 CSD／CSDPLUS 用の推奨 C_{MOD} 値

CSD ベース デザイン用の C_{MOD} の推奨値は **2.2nF** です。温度の変化時に C_{MOD} を安定化させるためには、X7R または NPO タイプのコンデンサが推奨され、コンデンサの定格電圧は 5V 以上でなければなりません。

4.2.2 I_{DAC} 範囲

最大センサーC_P が 45pF 未満のプロジェクトでは 4X を使用し、それ以外の場合は 8X を使用します。

4.2.3 自動校正

CY8C20xx7/S CSD／CSDPLUS 設計においては、自動校正を常に「有効」に設定する必要があります。自動校正アルゴリズムで I_{DAC} が正常に設定されるには、プリスケーラを適切に設定し、C_{MOD} を推奨のサイズに設定する必要があります。

4.2.4 I_{DAC} 値

このパラメーターでは自動校正が無効になったときの I_{DAC} の電流出力を設定します。自動校正が有効である場合、推奨通りにはこのパラメーターが無効にされ、何の効果もありません。自動校正が無効である場合、このパラメーターを増加すると raw カウント ベースラインが低くなり、逆もまた同様です。また、「CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター」の「IDAC 値」で説明したように、I_{DAC} 値は、raw カウントが 2^N の 85%になるように選択する必要があります。

4.2.5 補正 I_{DAC} 値

このパラメーターでは自動校正が無効になったときの補正 I_{DAC} の電流出力を決定します。このパラメーターは CSDPLUS UM のみに適用可能です。自動校正が有効である場合、推奨通りにはこのパラメーターが無効にされ、何の効果もありません。自動校正が無効である場合、このパラメーターを増加すると raw カウント ベースラインが低くなり、逆もまた同様です。自動校正が無効である場合、以下の方法を使って補正 I_{DAC} 値をチューニングします。

最初に、変調 I_{DAC} を使用して (補正 I_{DAC} を 0 に保つ)、raw カウントを最大 raw カウントの 85%に調整します。その後、補正 I_{DAC} を変調 I_{DAC} 値の 10%に増やし、変調 I_{DAC} を再調整して、最大 raw カウントの 85%を達成します。SNR をメモしてください。最大の SNR を達成するまで、より高い補正 I_{DAC} 値でこの手順を繰り返します。補正 I_{DAC} 値が高いほど、信号が高くなることに注意してください (詳細は、「CSD／CSDPLUS ユーザー モジュールの低レベル パラメーター」の「補正 IDAC 値」を参照してください)。また、一般的には最良の SNR が I_{DAC} 値 = I_{COMP} 値のとき達成されることにも注意してください。自動校正を有効にすると、I_{DAC} および I_{COMP} は、I_{DAC} = I_{COMP} であり、raw カウントが 2^N のおおよそ 85%であるように設定されます。

4.2.6 プリチャージ源

このパラメーターではセンサー スイッチング クロック源を選択します。オプションはプリスケーラ (分周器を介して IMO を使用する) および PRS (分周された IMO クロックを疑似乱数発生器に渡し、スプレッド拡散クロックを提供する) です。PRS は優れたノイズ耐性を提供し、ノイズ放射を低減するため、プリチャージ源の推奨デフォルト設定となります。場合によって、プリスケーラ プリチャージ源はより高い SNR を提供できます。しかし、銅製の回路を使用する場合、通常 SNR の上昇はわずかであり、PRS の利点を無視するほどの価値はありません。

4.2.7 プリスケーラ

プリスケーラは、プリチャージ クロックを生成するために IMO に適用される分周器です。これは、CSD デザインを適切に調整するための最も重要なハードウェア UM パラメーターです。プリスケーラーは選択したプリチャージ源、IMO、およびスキャン中のセンサーの C_P に依存します。表 4-1 はこれらのパラメーターに基づいて推奨されるプリスケーラ設定を示します。

表 4-1. プリチャージ源、IMO および C_P に基づくプリスケーラ設定

C_P (pF)	プリチャージ源 = PRS			プリチャージ源 = プリスケーラ		
	プリスケーラ IMO = 24MHz	プリスケーラ IMO = 12MHz	プリスケーラ IMO = 6MHz	プリスケーラ IMO = 24MHz	プリスケーラ IMO = 12MHz	プリスケーラ IMO = 6MHz
<6	1	注 1	注 1	2	1	1
7～11	2	1	注 1	4	2	1
12～15	2	1	注 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 1: このプリチャージ源、プリスケーラおよび C_P の組み合わせは推奨されません。

4.2.8 分解能

値の範囲は 9～16 ビットです。分解能を増やすと、感度、SNR、ノイズ耐性が向上しますが、スキャン時間が長くなります。スキャン分解能が N の場合、最大 raw カウント (全範囲) は 2^N–1 です。表 4-2 は C_P と指の静電容量 C_F に基づく推奨の分解能設定を示します。C_F はセンサーに指を置いたときにセンサー静電容量に発生する変化です。C_F は、オーバーレイの厚さ、センサー サイズ、およびセンサーの他の大きいな導電体との近接度で決まります。図 4-4 は、オーバーレイの厚さと円形センサーの直径に対する C_F 値を示します。

図 4-4. オーバーレイの厚さと円形センサーの直径に対する指静電容量 (C_F)

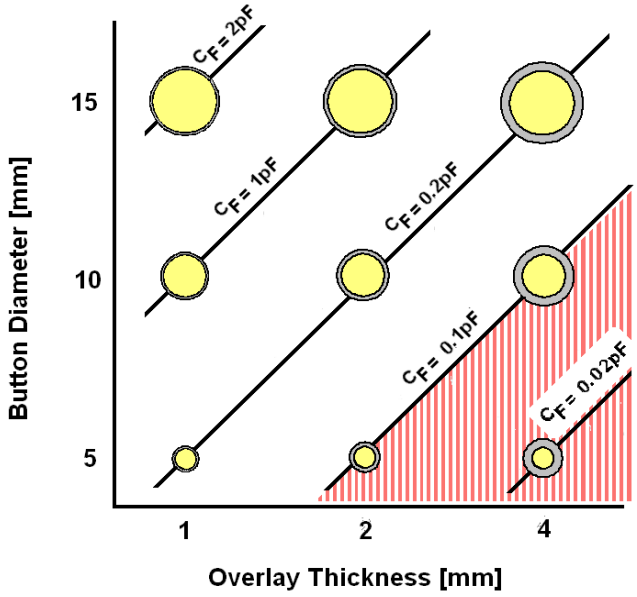


表 4-2. 指静電容量と C_P に基づく分解能設定

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

4.2.9 スキャン速度

このパラメーターにより、スキャン結果の各 LSB の積分時間を制御します。超高速、高速、普通、低速のオプションがあります。一般的には、高速で始めることを推奨します。必ずではありませんが、スキャン速度を遅くしたら SNR が高くなりますが、スキャン時間が長くなり消費電力が増加する場合があります。表 4-3 に、分解能とスキャン速度に応じた、センサーの実際のスキャン時間を、マイクロ秒単位で示します。

表 4-3. 分解能とスキャン速度に対する単一センサーのスキャン時間 (µs)

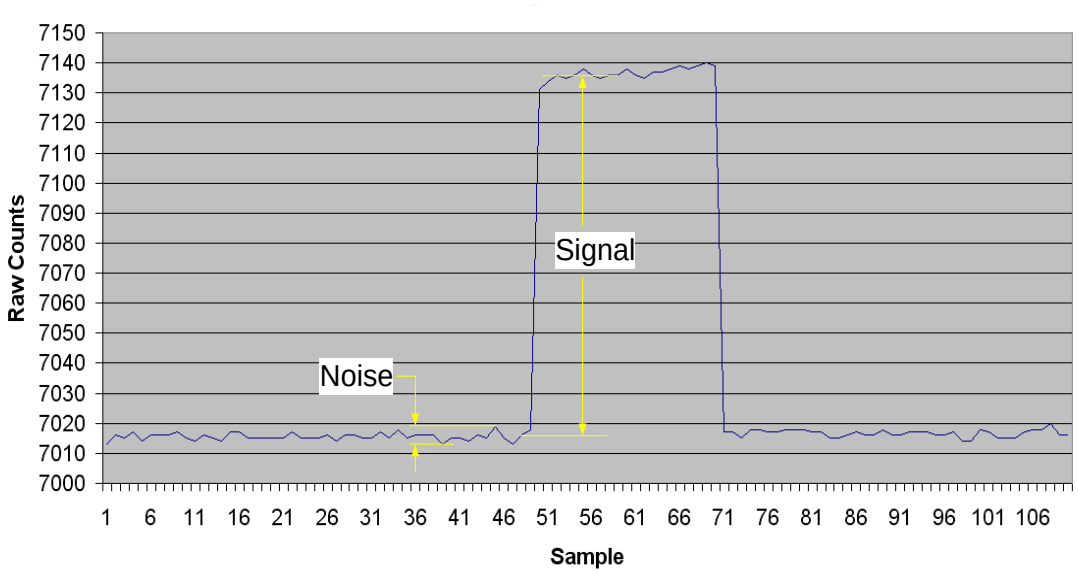
分解能 (ビット)	スキャン速度			
	超高速	高速	通常	低速
9	57	78	125	205
10	78	125	205	380
11	125	205	380	720
12	205	380	720	1400
13	380	720	1400	2800
14	720	1400	2800	5600
15	1400	2800	5600	11000
16	2800	5600	11000	22000

4.2.10 高レベル API パラメーター

高レベルの API パラメーターは、センサー アクティブ化動作とノイズを区別し環境の変化による信号ドリフトを補正する高レベルのファームウェア アルゴリズムの動作を決定します。これらのパラメーターを適切な値にするためには、システムとのデジタル通信インターフェースを確立し、各センサーの指によるアクティブ化イベント中に raw カウント、ベースラインや差分カウントを監視しなければなりません。このデータは、それぞれ CSDPLUS_waSnsBaseline[] 、CSDPLUS_waSnsResult[] および CSDPLUS_waSnsDiff[] と呼ばれるアレイに保存されます。このデータにより、高レベル API パラメーター設定は、主に周囲ノイズおよび指信号強度に基づいていることが分かります。ノイズおよび信号強度は EMI 環境、PCB レイアウト、オーバーレイの厚さやシステムのその他の物理的特性に依存します。したがって、これらのパラメーター設定のベースとなったデータはシステムの最終組み立て状態で、使用中と同じ EMI 環境において、最初の位置で取得しなければなりません。

図 4-5 は、指によるアクティブ化サイクルの間、すなわち、センサーがアクティブ化されてから非アクティブ化されるまでの間、センサーから得られた標準 raw カウントを示します。ラベルを、raw データに応じてノイズと信号を計算する方法を示すデータ上に、重ね合わせます。必要に応じて、以下の高レベル パラメーターの説明では、これらのノイズや信号の値に基づいて各パラメーターを設定する方法についての情報も示します。CapSense 設計のベスト プラクティスによれば、CapSense システムの動作の安定性のためには SNR は 5:1 以上でなければなりません。SNR が 5:1 未満の場合、「CapSense 入門」ガイドラインに記載されているようにハードウェア パラメーターを調整するか、または PCB レイアウトを変更するか、またはその両方を行って、SNR 比を 5:1 以上に引き上げる必要があります。

図 4-5. 指によるアクティブ化サイクルの間のセンサーから得られた標準 raw カウント



4.2.11 高レベル パラメーターの設定

以下の推奨事項は最適なパラメーター設定の選択の開始点です。

- **指閾値**: センサーがオンの状態で raw カウントの 80%に設定します。
- **ノイズ閾値**: センサーがオフの状態で raw カウントの 40%に設定します。
- **負のノイズ閾値**: ノイズ閾値に等しく設定します。
- **ベースライン更新閾値**: ノイズ閾値の 2 倍に設定します。
- **ヒステリシス**: センサーがオンの状態で raw カウントの 10%に設定します。
- **低ベースライン リセット**: 50 に設定します。
- **センサー自動リセット**: 設計要件に応じて設定します。
- **デバウンス**: 設計要件に応じて設定します。

4.3 SmartSense_EMCPLUS ユーザー モジュールの使用

センサー寄生容量が 5pF～45pF の範囲内で最小の指タッチ感度が 0.1pF である限り、SmartSense_EMCPLUS を利用してチューニングを必要としない CapSense デザインを作成できます。PSoC Designer 5.1 以降のバージョンに伴う SmartSense_EMCPLUS ユーザー モジュールを用いて SmartSense_EMCPLUS デザインを作成できます。また本節では、既存の CSD CapSense デザインを SmartSense_EMC_PLUS に移行する方法も説明します。

4.3.1 SmartSense_EMC_PLUS のガイドライン

アプリケーションで SmartSense_EMCPLUS ユーザー モジュールを使用するときは、以下のガイドラインに従ってください。

- SmartSense_EMCPLUS では、静電容量ユーザー インターフェースのデザインを「設計上の注意事項」節で説明したレイアウトとシステム デザインのベスト プラクティスに従って設計する必要があります。
- すべての CSD／CSDPLUS UM パラメーター (I_{DAC} 値、プリスケーラー期間、クロック分周器、スキャン速度、分解能など) は、実行中に SmartSense_EMCPLUS ユーザー モジュールで決定されます。設計にどのような影響が出るか完全に理解していなければ、ファームウェア内でこれらの CSD パラメーターを修正する API を使用しないでください。
- 既存のデザインを CSD／CSDPLUS から SmartSense_EMC_PLUS に移行するには、以下を行ってください。
  - ☐ まず、CSD／CSDPLUS パラメーターを設定または変更するすべての API はプログラムから削除されたことを確認します。
  - ☐ デザイン内のすべての CapSense センサーの寄生容量が環境と PCB 生産プロセスのばらつきに応じて、5pF～45pF の範囲内であることを確認します。
  - ☐ 推奨 C_{MOD} コンデンサ (X7R、2.2nF、定格電圧 5V 以上) が、ユーザー モジュール ウィザードで選択された C_{MOD} ポートピンに接続されていることを確認します。

4.3.2 相違点

SmartSense_EMCPLUS ユーザー モジュールと標準 CSD／CSDPLUS ユーザー モジュールの違いは以下のとおりです。

- SmartSense_EMCPLUS ユーザー モジュールと標準 CSD／CSDPLUS ユーザー モジュールは、同じ API をサポートします。そのため、ユーザー モジュール インスタンス名以外に、その他の API の配置、設定、開始や呼び出しを変更する必要はありません。
- チューニングに関連するパラメーターがすべて SmartSense_EMCPLUS ユーザー モジュールによってデバイス実行中に自動的に設定されるため、チューニングのためにユーザー モジュールのパラメーターを設定することは不要です。
- C_{MOD} コンデンサの値は 2.2nF に制限されます。定格電圧が 5V 以上の X7R または NPO コンデンサをすべての CapSense アプリケーションに使用することを推奨します。
- 性能を最大限にしながら、CapSense の安定した動作を確保するために、SmartSense_EMCPLUS アルゴリズムで各センサーの信号 SNR を 5:1～11:1 の範囲内に維持します。
- SmartSense_EMCPLUS ユーザー モジュールのスキャン時間は 24MHz 動作モードでは、センサーの寄生容量に応じて、センサーあたりに 410µs～2.8ms にアルゴリズムによって制限されます。

4.3.3 SmartSense_EMC_PLUS 用の推奨 C_{MOD} 値

SmartSense_EMC_PLUS ベース設計の推奨 C_{MOD} 値は 2.2nF です。温度の変化時に C_{MOD} を安定化させるために、X7R または NPO タイプのコンデンサの使用が推奨されます。コンデンサの定格電圧は 5V 以上である必要があります。

4.3.4 SmartSense_EMCPLUS ユーザー モジュールのパラメーター

このユーザー モジュールでは、設定する必要があるパラメーターは 4 つだけです。それらは以下のとおりです。

- センサー自動リセット
- デバウンス
- 変調コンデンサ ピン
- 感度レベル

4.3.4.1 センサー自動リセット

このパラメーターは、ベースラインが連続して更新されるか、または信号差がノイズ閾値を下回った場合にのみ更新されるかを決めます。有効になった場合、ベースラインは連続して更新されます。この設定では、センサーがオンとなる最大時間(一般的に 5～10 秒) を制限しますが、システムの故障のためタッチがなくても raw カウントが急に上昇した場合、センサーが無限にオンのままになることを防ぎます。

4.3.4.2 デバウンス

デバウンス パラメーターでは、デバウンス カウンターをセンサーのアクティブ状態への遷移に追加します。センサーが非アクティブ状態からアクティブ状態になるには、指タッチ信号がデバウンス数の分だけの連続したスキャンの間センサーにある必要があります。このパラメーターは、同じ程度ですべてのセンサーへ影響を及ぼします。

4.3.4.3 変調コンデンサ ピン

このパラメーターで 2.2nF、X7R、定格電圧 5V 以上の C_{MOD} が接続するピンを選択します。P0[1]および P0[3]が選択可能です。

注: SmartSense_EMCPLUS が正常に動作するには、外部 2.2nF コンデンサが必須です。

4.3.4.4 感度レベル

センサー信号の強度は感度により増減されます。感度の値を低くすると (0.1pF)、センサー信号の強度が増加します。オーバーレイの厚いデザインでは、デバイスが正常に動作するためにセンサーの信号が強いことを必要とします。高 (0.1pF)、中高 (0.2pF)、中低 (0.3pF) と低 (0.4pF) の感度が選択可能です。

強いセンサー信号 (高感度) を生成するには、SmartSense_EMCPLUS ユーザー モジュールは、センサー スキャン時間を延長する必要があります。つまり、感度レベルが 0.1pF (高感度) に設定されたセンサーは、感度レベルが 0.2pF (中高感度) に設定されたセンサーに比べ、スキャン時間が長いです。

ベスト プラクティスとしてのチューニング方法は、センサーで 5:1 の SNR を達成できる最高の感度値を調べることです。まず、最高の感度値 (0.4pF) を使用してチューニングをし、そして SNR が 5:1 になるまで感度値を低下してみてください。

4.3.5 SmartSense_EMCPLUS ユーザーモジュール固有のガイドライン

SmartSense_EMC ユーザー モジュールに適用可能なガイドラインすべては、SmartSense_EMCPLUS UM に適用できます。CapSense デザインと SmartSense_EMC_PLUS ベースのデザインの一般的なガイドラインは、「CapSense 入門」デザインガイドを参照してください。本節では、SmartSense_EMCPLUS UM のいくつかの重要な問題点を説明します。

4.3.5.1 センサーのスキャン時間、応答時間およびメモリの使用

SmartSense_EMCPLUS UM でセンサーを実装する場合、センサーのスキャン時間、センサーの応答時間および RAM メモリ使用量はユーザー モジュールで選択された耐性モードによって決まります。

- 中耐性モードのセンサー スキャン時間は、低耐性モードのセンサー スキャン時間の 2 倍になります。高耐性モードのセンサー スキャン時間は、低耐性モードのセンサー スキャン時間の 3 倍になります。
- スキャン時間が長くなれば、センサーの応答時間もそれに比例して長くなります。中耐性モードの応答時間は、低耐性モードの応答時間の 2 倍になります。同様に、高耐性モードの応答時間は低耐性モードの応答時間の 3 倍になります。
- SmartSense_EMCPLUS UM は RAM メモリを使用して、堅牢な電磁対応アルゴリズムを実行します。その結果、最高の耐性モード (高) で使用する RAM メモリ容量は、低耐性モードでの RAM メモリ使用容量の約 3 倍です。中耐性モードで使用する RAM メモリ容量は、低耐性モードで使用する RAM メモリ容量の約 2 倍だけです。

4.3.5.2 IMO 許容誤差とタイムクリティカルなタスク

SmartSense_EMC 対応デバイスの IMO 許容誤差は+5%と–20%です。

- タイムクリティカルなアルゴリズムとロジックを実行する場合、ファームウェア ロジックやアルゴリズムが中断しないように IMO 許容誤差を考慮する必要があります。
- プロジェクトで割り込みを使用する場合、割り込み遅延、ISR 実行時間などの解析時に IMO 許容誤差を考慮することが必要です。
- IMO に依存するタイミング解析 (例えば、IMO でクロック供給されたタイマー、ファームウェアのループにより生じる遅延、API 実行時間など) のとき、安定したアプリケーション ファームウェアを確保するために IMO 許容誤差を検討する必要があります。

4.3.5.3 I²C 動作速度

I²C インターフェースの動作周波数は、SmartSense_EMC 対応デバイスのユーザー モジュールの実際周波数の最大 80%に制限されます。この限度は 20%の IMO 許容誤差に起因して確立されます。

- つまり、I²C ユーザー モジュールでクロック周波数が 400kHz と選択された場合、I²C インターフェースの動作可能な最大クロックは 320kHz になります。同様に、I²C ユーザー モジュールにおいて、100kHz と 50kHz クロック モードが選択される場合、動作周波数は、それぞれ 80kHz と 40kHz に制限されます。
- I²C スレーブ インターフェースを使用する場合、マスター クロックは、前述のように抑えた仕様の範囲内で動作することが必要です。これに違反すれば、データの破壊や I²C バス混乱、I²C ユーザーモジュールの不整合な動作が発生します。
- I²C マスター モジュールは、インターフェースのスループットのみに影響を与えます。

I2CSBUF ユーザー モジュールを使用する場合、CPU が 32 バイトの専用バッファにアクセスするとき、およびデバイスがスリープまたはディープ スリープ モードに入ったときにオートナック モードを使用することを強く推奨します。詳細については、I2CSBUF ユーザー モジュール データシートを参照してください。

4.3.6 CapSense Sensor のスキャン時間

寄生容量の幅広い範囲で指の応答感度の整合性を維持するために、SmartSense_EMCPLUS ユーザー モジュールは、ユーザー モジュールのハードウェア パラメーターを自動的に決定します。この結果として、センサー スキャン時間が一定に保たれません。大量生産のデザインでは、PCB 寄生容量のばらつきによって異なります。

センサーの総スキャン時間は 4 つの要因で決まります。これらは、センサーの寄生容量、IMO 周波数、CPU 動作周波数、および SmartSense_EMCPLUS ユーザー モジュールの感度レベルです。

センサーのスキャン時間は式 4-3 と以下の表で計算できます。

スキャン時間 = サンプリング時間 (ST) + 処理時間 (PT) 式 4-3

以下の表はさまざまな IMO と感度レベルに対応するサンプリング時間の値を示します。

表 4-4. IMO = 24MHz の場合のセンサーのサンプリング時間

感度 = 0.2pF		感度 = 0.3pF		感度 = 0.4pF	
C_P (pF)	ST (µs)	C_P (pF)	ST (µs)	C_P (pF)	ST (µs)
8～10	340	8～17	340	8～10	170
10～23	680	17～35	680	10～23	340
23～41	1360	35～41	1360	23～41	680
41～45	2730	41～45	2730	41～45	1360

表 4-5. IMO = 12MHz の場合のセンサーのサンプリング時間

感度 = 0.2pF		感度 = 0.3pF		感度 = 0.4pF	
C_P (pF)	ST (µs)	C_P (pF)	ST (µs)	C_P (pF)	ST (µs)
8～10	680	8～17	680	8～10	340
10～23	1360	17～35	1360	10～23	680
23～41	2730	35～41	2730	23～41	1360
41～45	5460	41～45	5460	41～45	2730

表 4-6. IMO = 6MHz の場合のセンサーのサンプリング時間

感度 = 0.2pF		感度 = 0.3pF		感度 = 0.4pF	
C_P (pF)	ST (µs)	C_P (pF)	ST (µs)	C_P (pF)	ST (µs)
8～11	680	8～10	680	8～11	680
11～23	1360	10～17	1360	11～23	1360
23～42	2730	17～35	2730	23～41	2730

42～45	5460	35～41	5460	41～45	5460
		41～45	10920		

表 4-7 はさまざまな CPU 周波数に対応する処理時間の値を示します。

表 4-7. センサーの処理時間

CPU CLK	処理時間 (PT) (µs)
24	71
12	142
6	284
3	568

例えば、CapSense システムは、IMO 周波数が 24MHz、CPU クロックが 6MHz (IMO/4 に相当)、SmartSense_EMCPLUS 感度レベルが 0.3pF の場合、寄生容量が約 15 pF のセンサーのスキャン時間は、式 4-1 と上記の表の値を使用して計算できます。

前述の設定 (IMO 周波数: 24MHz、感度: 0.3pF) のサンプリング時間は、表 4-4 から選択され、680µs です。前述の設定 (CPU クロック: 6MHz) の処理時間は表 4-7 から選択され、284µs です。

したがって、本設定では、合計スキャン時間は 680 + 284 = 964µs です。すべてのセンサーのスキャン時間は、各センサーのスキャン時間の合計です。

4.3.7 SmartSense_EMCPLUS 応答時間

標準 CSD と代表的な CapSense スキャン ファームウェアを用いた以下のアプリケーションを考慮してみましょう。

- 3 個の CapSense センサー、寄生容量 5pF～10pF
- IMO 周波数 12MHz、CPU クロック 12MHz
- センサー感度レベル 0.4pF
- デバウンス = 3

上記の表によると、各センサーのスキャンは 482µs を要し、3 個のセンサーの合計スキャン時間は 1.45ms です。以下のファームウェア例では、追加のファームウェアの実行には 1ms が必要となるため、ループの実行時間は 2.45ms となります。

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

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

つまり、CapSense センサーをアクティブにした場合、ファームウェアは 7.35ms の間センサーのオン状態を生成します(連続スキャンのデバウンス数に対してセンサーをアクティブにする必要があります)。これは通常 CapSense システムの応答時間とみなされます。

一貫性の維持のために寄生容量に対応してスキャン時間が変わる場合、プロセスのばらつきのためセンサーの寄生容量が変われば、応答時間は何の影響を受けるでしょうか。この場合は応答時間が長くなる (応答が遅い) ことがあります。これは、センサー性能への不利な影響を与えます。堅牢なファームウェア デザインを作るためのガイドラインを次の節で説明します。

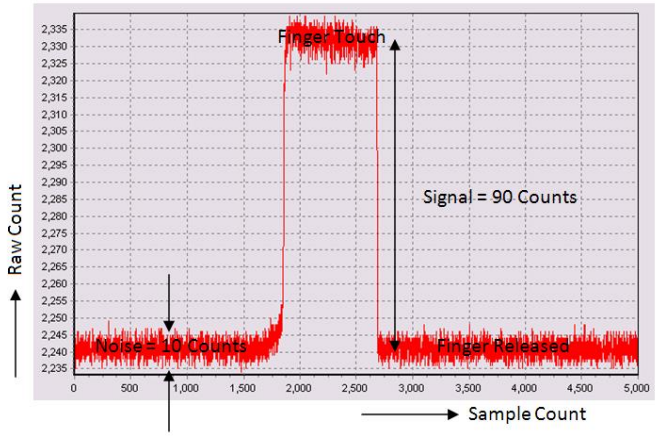
4.3.8 SmartSense_EMCPLUS UM による最低限の SNR の確保方法

SmartSense_EMCPLUS は、複雑なチューニング プロセスを適用しない CSD ベースの SmartSense ユーザー モジュールによる高度な電磁対応デザインです。しかし、SmartSense_EMCPLUS UM を使う際にデザインの堅牢さを確保するための簡単な 2 つのステップがあります。

1. センサーの信号を計測するために、CapSense ユーザー モジュール パラメーターを監視するためのリアルタイムのモニター ツールを用意します。チューニング プロセスでは、センサーの raw カウント (SmartSense_EMCPLUS_ waSnsResult)、センサーの正規化信号 (SmartSense_EMCPLUS_baSnsSignal)、センサーの指閾値 (SmartSense_EMCPLUS_baBtnFThreshold) を確認する必要があります。モニター監視として、LCD などの数値表示ディスプレイは遅く、データの変動を視覚化できないので、使用しないでください。MultiChart または I²C USB Bridge Control Panel をデータ監視に使用することを推奨します。

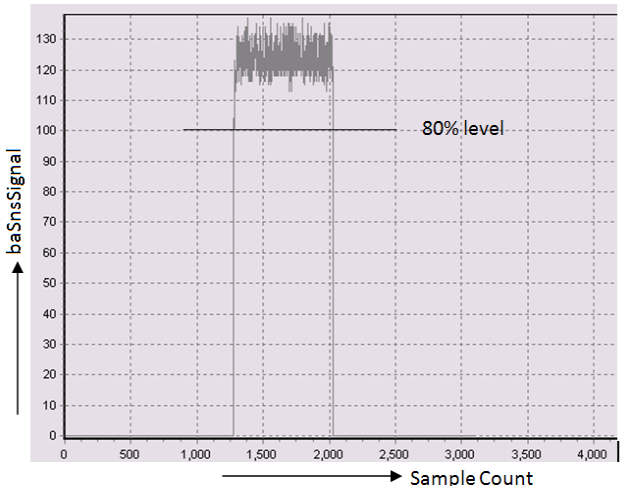
2. 感度レベルを 0.4pF (低) に設定し、SNR を計算します。図 4-6 は指タッチの標準 raw カウントのグラフを示します。CapSense ベスト プラクティスによると、デザインを堅牢にするためには SNR が 5:1 を超える必要があります。測定された SNR が 10:1 以上であれば、SNR が 5:1～10:1 の範囲内になるまで、感度レベル値を次のレベルに下げます。

図 4-6. 標準センサーへの指タッチがあるときの raw カウント グラフ



3. 設計時、自動指閾値の機能を使用する場合、このプロセスは前のステップで完了します。フレキシブルな指閾値の機能を使用する場合、各センサー用の指閾値を設定してプロセスを終了します。指閾値を設定するには、センサー信号 (SmartSense_EMCPLUS_baSnsSignal) を監視し、指閾値を指タッチがあった場合のセンサー信号値のおおよそ 80%に設定します。これでプロセスは完了です。図 4-7 は標準センサーの信号と指閾値を示します。

図 4-7. 標準センサーへの指タッチがあるときのセンサー信号



4.3.9 ファームウェア デザイン ガイドライン

CapSense センサーの応答時間は、センサーの寄生容量の増加に応じて変わることがあります。増加する可能性のあるループ実行時間 (以下のサンプル コードを参照) を監視することも重要です。すべてのセンサーの寄生容量が 10pF 以下の場合、ファームウェア ルーティンは 2.45ms の速度で実行されます。プロセスのばらつきのためセンサーの寄生容量が増加して、センサー スキャン時間が長くなると、この速度は変わります。

以下は、メイン ループ実行時間に応じてポート ピンを切り替えるサンプル コードです。

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

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

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

Port_0[1]ピン上の信号の周期は、4.9ms です (ポート ピンがトグルするとき、周期はループ時間の 2 倍になります)。センサーの寄生容量を約 15pF まで増加すれば、スキャン時間は 1.78ms になり、よって Port_0[1]上の信号の周期は 5.6ms になります。

センサーの寄生容量が SmartSense_EMCPLUS 静電容量群の限界の近く (例えば、限界が 10pF の場合、その近くは 9pF) に増加すれば、プロセスのばらつきのため、SmartSense_EMCPLUS はアプリケーションの近傍のスキャン時間を選択することがあります。したがって、同一デザインの異なるデバイス (量産中) は、2 種のループ実行時間と 2 種の応答時間がある場合があります。

上記に説明したように、その他の機能 (例えば、ソフトウェア PWM、ソフトウェア遅延など) を実行するために、ファームウェアはセンサーのスキャン時間に依存してはなりません。ウォッチドッグ タイマー (WDT) を実装するプログラムでは、WDT 終了時間を設定時にこの点を検討することが必要です。

以下は、Timer16 ユーザー モジュールを使用して同一のメイン ループ実行時間を得る簡単なファームウェア実装例です。

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

#pragma interrupt_handler myTimer_ISR_Handler;
void myTimer_ISR_Handler( void );

void main()
{
M8C_EnableGInt;

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

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

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

       /* Wait till timer expires or
        sleep here */

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

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

```
        }

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

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

上記の例で、センサーのスキャンが完了しても、プログラムはタイマー終了を待ちます。タイマー周期は、最悪ケースのメイン ループ実行時間によって選択します。これは、個々の CapSense センサーの最悪ケースのスキャン時間の合計です。センサーの寄生容量が SmartSense_EMCPLUS 静電容量群の限界に近い場合、計算のためにより高いスキャン時間を選択します (表 4-5 を使用します)。

SmartSense_EMCPLUS UM により、システムに静電容量タッチ センシング ユーザー インターフェースを容易に実装できます。チューニング プロセスを簡単にし、PCB の製造工程の変更やその他の変更があっても、量産中の歩留まりの向上に役立ちます。したがって、既存の CSD ベース CapSense デザインを SmartSense_EMCPLUS に移行し、新しいデザインには SmartSense_EMCPLUS を使用することを推奨します。

SmartSense_EMCPLUS のメイン ループ実行時間とスキャン時間は、プロセスのばらつきによって異なります。プロセスのばらつきは CapSense 性能に何の影響も与えませんが、ファームウェア開発者は SmartSense_EMCPLUS 自動チューニング機能を備えた CapSense PLUS アプリケーションを実行する際には、この点を考慮すべきです。

4.4 CY8C20xx6A/AS から CY8C20xx7/S への設計の移行

CY8C20xx7/S ファミリの CapSense コントローラーは QuiteZone™ 技術ベースの優れたノイズ耐性および 0.1pF の改善された感度 (SNR が 5:1) を提供します。本節では、既存の CY8C20xx6A/AS 設計の移行の重要なポイントをまとめます。

4.4.1 廃止されたサポート／ユーザー モジュール

- USB インターフェースは CY8C20xx7/S ファミリでは廃止されました。
- オンチップ デバッグ (OCD) のサポートは CY8C20xx7/S ファミリでは廃止されました。
- CSA および CSA_EMC ユーザー モジュールは CY8C20xx7/S ファミリではサポートされません。

4.4.2 改善および新機能

- CSD ベースの改良された CapSense センシング エンジンは 0.1pF の改善された感度を提供します。
- 32 バイトの専用バッファを備える、新規の改良された I²C スレーブ インターフェース (I2CSBUF ユーザー モジュール) はスレーブによるクロック ストレッチを削除します。
- 改良された I²C インターフェースは、I²C スレーブ アドレス一致イベントからのウェイクアップ割り込みもサポートします。

デザインに適切な I²C ユーザー モジュールを選択するために、表 4-8 を使用してください。

表 4-8. I²C 機能

システム要件	I2CSBUF	EzI2C	I2CHW	I2Cm
I²C スレーブ アドレス一致によるウェイクアップ	有	有	無	該当なし
マスターはクロック ストレッチを必要とする	無	有	有	該当なし
I²C データ バッファ サイズ	1～32 バイト	1～255 バイト	1～255 バイト	該当なし
I²C マスター	無	無	無	有

4.4.3 ピンの互換性

表 4-9. ピンの互換性

パッケージ	CY8C20xx6A/AS とのピン互換性
16-SOIC	CY8C20xx7/S の新規のパッケージ
16-QFN	ピン ツー ピン互換
24-QFN	1 ピン ― ピン 23 は、CY8C20xx7/S では Vss、CY8C20xx6A/AS では I/O
32-QFN	2 ピン ― ピン 28 とピン 29 は、CY8C20xx7/S ではそれぞれ I/O と Vss、CY8C20xx6A/AS ではそれぞれ Vss と I/O
30-WLCSP	ピン ツー ピン互換
48-QFN	2 ピン ― ピン 36 とピン 45 の両方は CY8C20xx7/S では NC、CY8C20xx6A/AS では I/O

5. 設計上の注意事項

アプリケーションに静電容量タッチ センシング技術を組み込む場合、CapSense デバイスはより大きなフレームワークを必要とすることに注意してください。PCB レイアウトからエンドアプリ環境へのユーザー インターフェースにいたるまで、すべての設計レベルにおいて慎重に設計すると、堅牢で信頼性の高いシステム性能の実現が可能になります。詳細な情報については、「CapSense 入門」を参照してください。

5.1 オーバーレイの選択

「CapSense の原理」では、式 4-1 は指の静電容量の計算公式です。

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

ここで、

ε_0 = 真空の誘電率

ε_r = オーバーレイの比誘電率

A = 指とセンサー パッドの重複面積 (mm^2)

D = オーバーレイの厚さ (mm)

CapSense 信号の強度をあげるには、比誘電率の高いオーバーレイ素材を選択し、オーバーレイの厚さを薄くし、ボタン直径を大きくします。

表 5-1. オーバーレイ素材の絶縁力

素材	□r	絶縁破壊電圧 (V/mm)	12kV でのオーバーレイの最小の厚さ (mm)
空気	1.0	1200～2800	10
木材 (乾燥)	1.2～2.5	3900	3
ガラス (通常)	7.6～8.0	7900	1.5
ガラス - ホウケイ酸塩 (Pyrex®)	6.0	13,000	0.9
PMMA プラスチック (Plexiglas®)	2.8	13,000	0.9
ABS	2.4～4.1	16,000	0.8
ポリカーボネート (Lexan®)	2.9～3.0	16,000	0.8
フォーマイカ	4.6～4.9	18,000	0.7
FR-4	4.8	28,000	0.4
PET フィルム (Mylar®)	3.2	280,000	0.04
ポリイミド フィルム (Kapton®)	2.9～3.9	290,000	0.04

導電材料は、電場パターンを妨げるため、オーバーレイとしては使用できません。このため、金属粒子を含有する塗料をオーバーレイで使用しないでください。

オーバーレイを CapSense PCB に接合するために、通常は接着剤を使用します。3M™の 200MP と呼ばれる透明なアクリル系接着剤は CapSense アプリケーションでの使用に適しています。この特殊な接着剤は裏に紙が付いたテープロール形状で販売されます (3M 商品番号: 467MP、468MP)。

5.2 ESD 保護

安定した ESD 耐性は、入念なシステム設計から生まれた当然の結果です。ユーザー インターフェースを始め製品で接触放電がどのように発生するか検討することで、CapSense コントローラーにダメージを与えることなく、18kV までの放電現象に耐えられます。

CapSense コントローラー ピンは直流 2kV 放電現象に耐えられます。ほとんどの場合、オーバーレイの素材がコントローラー ピンに十分な ESD 保護を提供します。表 5-1 は、IEC 61000-4-2 で指定されるように CapSense センサーを 12kV 放電から保護するために必要となるさまざまなオーバーレイの厚さの一覧です。オーバーレイの素材が十分な ESD 保護を提供しない場合、対応策は次の順番で適用すべきです: 防止、リダイレクト、クランプ。

5.2.1 防止

接触面のすべてのパスが、潜在的な高電圧接触よりも絶縁破壊電圧が高いことを確認してください。また、CapSense コントローラーと ESD 発生源となる可能性のある部分との間に適切な距離を保つようにシステムを設計します。適切な距離を実現できない場合は、絶縁破壊電圧の高い素材で製造される保護レイヤを ESD 発生源と CapSense コントローラーの間に設けます。厚さ 5mil の 1 層の Kapton®テープは、18kV に耐えられます。

5.2.2 リダイレクト

基板の集積密度が高い場合は、放電現象を避けることが難しいかもしれません。この場合、放電の発生場所を制御することで CapSense コントローラーを保護できます。標準的な技法としては、シャーシ グランドに接続している回路基板の周囲にガードリングを配置します。「PCB レイアウト ガイドライン」で推奨するように、ボタンまたはスライダー センサーの周囲にハッチング グランド面を作り、ESD 事象をセンサーおよび CapSense コントローラーから遠く離してリダイレクトできます。

5.2.3 クランプ

CapSense センサーは、意図的にタッチ面の近くに設置されるため、放電経路をリダイレクトすることは実用的でない場合もあります。この場合、直列抵抗と専用 ESD 保護デバイスを使用してみることを推奨します。

推奨直列抵抗は 560Ω です。

より効果的な方法は、専用 ESD 保護デバイスを脆弱な配線上に置くことです。CapSense 用の ESD 保護デバイスは低静電容量のものである必要があります。表 5-2 に、CapSense コントローラー向けの推奨デバイスを示します。

表 5-2. CapSense 用の推奨低静電容量 ESD 保護デバイス

ESD 保護デバイス		入力静電容量	リーク電流	接触放電の上限	空中放電の上限
メーカー	製品番号				
Littelfuse	SP723	5pF	2nA	8kV	15kV
Vishay	VBUS05L1-DD1	0.3pF	0.1µA <	±15kV	±16kV
NXP	NUP1301	0.75pF	30nA	8kV	15kV

5.3 電磁環境適合性 (EMC) の注意事項

5.3.1 放射性干渉

放射性電気エネルギーはシステム測定に影響を与え、さらにプロセッサ コアの動作に影響を与える可能性もあります。PCB レベルでは、干渉は、CapSense センサーの配線や、その他のあらゆるデジタル、アナログ入力を介して PSoC チップに侵入します。RF 干渉の影響を最小限にするためのレイアウト ガイドラインは以下のとおりです。

- **グランド面**: PCB にグランド面を設けます。
- **直列抵抗**: CapSense コントローラー ピンから 10mm 以内に直列抵抗を配置します。

- ☐ CapSense 入力ラインの推奨抵抗は 560Ω です。
- ☐ I^2C や SPI などの通信回線の推奨直列抵抗は 330Ω です。

- **配線の長さ**: 可能な限り配線を短くします。
- **電流ループの面積**: 電流の帰路を短くします。寄生容量の影響を軽減するために、ベタ グランドの代わりにハッチング グランドをセンサーおよび配線から 1cm 以内に配置します。
- **RF 源の位置**: LCD インバータやスイッチング電源 (SMPS) のようなノイズ源を備えたシステムを区分して、それらを CapSense 入力から隔てます。干渉を防ぐもう一つの良くある技術は電源のシールドです。

5.3.2 放射妨害波

スイッチト キャパシタ クロックを低周波数にして CapSense センサーからの放射妨害波が低減できます。このクロックは、プリスケーラ オプションを使用してファームウェアで制御されます。プリスケーラ値を高くすると、スイッチング クロックの周波数が低くなります。

5.3.3 伝導ノイズ耐性および伝導性放射

他のシステムとの相互接続によりシステムに混入したノイズは伝導ノイズと呼ばれます。相互接続には電源や通信ラインなどがあります。CapSense コントローラーは低消費電力デバイスのため、伝導性放射を回避しなければなりません。以下のガイドラインは伝導性放射を減らし伝導ノイズ耐性を高める助けになります。

- データシートで推奨されているようにデカップリング コンデンサを使用します。
- システム電源への入力に双方向性のフィルターを追加します。伝導性放射および伝導ノイズ耐性の両方に効果的です。パイ型フィルターの使用で、電源ノイズの敏感な部品への影響を防止しながら、部品それ自体のスイッチングノイズが電源プレーンに戻ってカップリングすることも防ぎます。
- CapSense コントローラーPCB がケーブルで電源に接続される場合は、ケーブルの長さを最短にして、 シールドケーブルの使用を検討します。
- 高周波ノイズを除去するために、電源や通信回線のまわりにフェライト ビーズを配置します。

5.4 ソフトウェア フィルター

ソフトウェア フィルターの使用も高レベルのシステム ノイズへの対応技術の 1 つです。表 5-3 は CapSense に役立つフィルタータイプの一覧です。

表 5-3. CapSense フィルター表

タイプ	説明	応用
平均	等しく加重された係数を持つ有限インパルス応答フィルター (フィードバックなし)	電源からの周期的ノイズ
IIR	RC フィルターに類似したステップ応答を備えた有限インパルス応答フィルター (フィードバック付き)	高周波ホワイト ノイズ (1/f ノイズ)
メジアン	サイズ N のバッファからメジアン入力値を計算する非線形フィルター	モーターおよびスイッチング電源によるスパイクノイズ
ジッタ	前の入力に基づいて現行の入力を制限する非線形フィルター	厚いオーバーレイからのノイズ (SNR < 5:1) に適用され、特にスライダーのセントロイド データに役立ちます。
イベントベース	センサー データで観察されたパターンへの事前定義された応答を発生させる非線形フィルター	CapSense データ送信をブロックするために、非接触イベント中によく使用されます。
ルールベース	センサー データで観察されたパターンへの事前定義された応答を発生させる非線形フィルター	タッチ表面の通常操作中によく使用され、誤ったマルチボタン選択などの特別なシナリオに対応します。

表 5-4 はさまざまなソフトウェア フィルターの RAM とフラッシュ要件を示します。各フィルター タイプに必要なフラッシュの大きさは、コンパイラの性能に依存します。ここに示す要件は ImageCraft コンパイラと ImageCraft Pro コンパイラの両方に該当します。

表 5-4. RAM およびフラッシュの要件

フィルタータイプ	フィルターの次数	RAM (センサーあたりのバイト数)	フラッシュ (バイト数) ImageCraft コンパイラ	フラッシュ (バイト数) ImageCraft Pro コンパイラ
平均	2～8	6	675	665
IIR	1	2	429	412
	2	6	767	622
メジアン	3	6	516	450
	5	10	516	450
raw カウントのジッタ フィルター	該当なし	2	277	250
スライダー セントロイドのジッタ フィルター	該当なし	2	131	109

5.5 消費電力

5.5.1 システム設計の推奨事項

多くのデザインでは、消費電力を最小限に抑えることは重要な目標です。CapSense 静電容量タッチ センシング システムの消費電力を削減する方法はいくつかあります。

- 低電力の GPIO 駆動モードの設定
- 高電力ブロックの電源の切断
- 低電力用に CPU の速度の最適化
- 低い V_{DD} での動作

これらの提案に加えて、スリープ スキャン方式の適用も非常に効果があります。

5.5.2 スリープ スキャン方式

標準アプリケーションでは、CapSense コントローラーをいつでもアクティブのままにする必要はありません。デバイスをスリープ状態にして、デバイスの CPU と主要ブロックを停止することができます。スリープ モードでのデバイスの消費電力はアクティブ時の電力よりもはるかに少ないです。

長期にわたってデバイスが消費した平均電流は、次の式で計算できます。

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

デバイスの平均消費電力は、次のように計算します。

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

5.5.3 応答時間対消費電力

式 5-2 から分かるように、平均消費電力は I_{AVE} または V_{DD} を減少させることにより削減できます。スリープ時間を増やして、I_{AVE} を低減することが可能です。スリープ時間をきわめて高い値にすると、CapSense ボタン応答時間は長くなります。結果として、スリープ時間はシステム要件に応じて設定する必要があります。

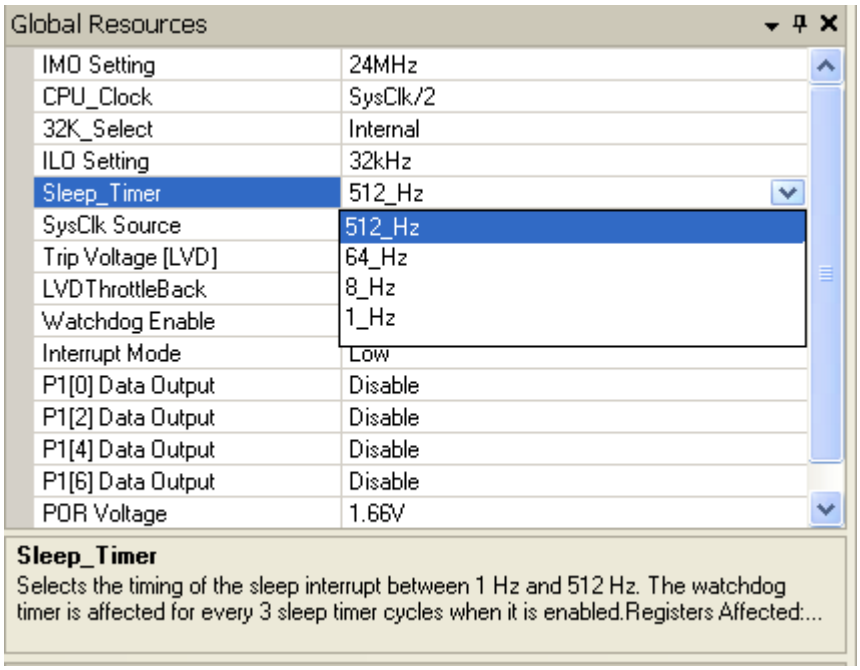
いずれのアプリケーションでも、消費電力と応答時間の双方が考慮すべき重要なパラメーターである場合、連続スキャンとスリープ スキャンモードの両方を組み合わせる方法を適用することができます。この方法では、デバイスはスリープ スキャンモードにほとんどの時間を費やします。前の節で説明したように、デバイスは定期的にセンサーをスキャンしてスリープ状態に入り、その結果、消費電力が少なくなります。ユーザーがセンサーに触れてシステムを動作させると、デバイスはスリープ状態にならず、センサーを継続的にスキャンする連続スキャン モードにジャンプするので、非常に優れた応答時間が得られます。デバイスは、指定されたタイムアウト期間の間連続スキャン モードのままになります。ユーザーがこのタイムアウト期間内にセンサーを動作させない場合、デバイスはスリープ スキャン モードに戻ります。

5.5.4 平均消費電力の測定

以下に、スリープ スキャン方式を使用するときの平均消費電力を判定する方法について説明します。

1. スリープ状態に入らずに (連続スキャン モードで) すべてのセンサーをスキャンするプロジェクトを作成します。センサーをスキャンする前に、コードにピン トグル機能を組み込みます。出力ピン状態のトグルは、オシロスコープで確認できるタイム マーカーとしての役割があります。
2. プロジェクトを CapSense デバイスにダウンロードし、消費電流を測定します。測定した消費電流を I_{ACT} に割り当てます。
3. データシートからスリープ電流情報を読み出し、I_{SLP} に割り当てます。
4. トグルしている出力ピンをオシロスコープで監視し、2 回のトグル間の時間を測定します。これはアクティブ時間です。この値を t_{ACT} に割り当てます。
5. スリープ スキャンをプロジェクトに適用します。スリープ スキャン サイクルの時間である T は、図 5-1 に示すように、グローバル リソース画面でのスリープ タイマー周波数を選択して設定します。
6. スリープ スキャン サイクル時間からアクティブ時間を差し引いて、スリープ時間を取得します。$T_{SLP} = T - t_{ACT}$。
7. 式 5-1 を使用して、平均電流を計算します。
8. 式 5-2 を使用して、平均消費電力を計算します。

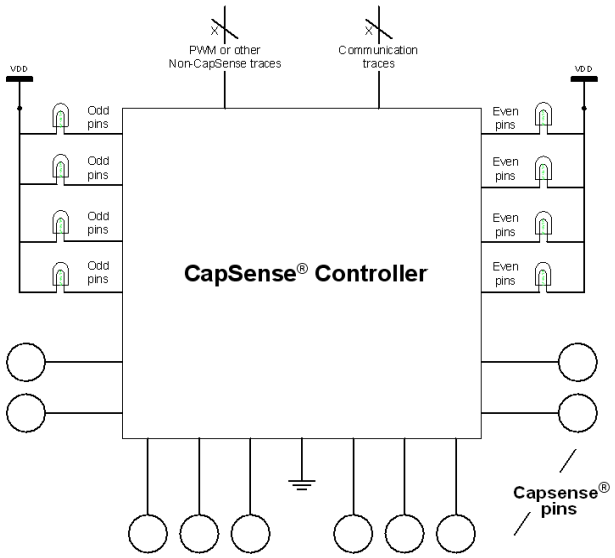
図 5-1. Global Resources ウィンドウ



5.6 ピン割り当て

CapSense センサー配線と通信配線と非 CapSense 配線間の相互影響を抑える効果的な方法としては、それぞれをポートの割り当てで隔離することです。図 5-2 に、QFN パッケージ向けの基本的な分離を示します。それぞれの機能が独立しているので、CapSense コントローラーは通信、LED およびセンシング配線が交差しないようにされます。

図 5-2. 通信、CapSense と LED の推奨ポート分離



CapSense コントローラーの全 GPIO は CapSense センサーおよび LED 駆動を実装できます。ただし、GPIO、LED 駆動、CapSense センサーが同じプロジェクトで実装される場合、最高のパフォーマンスのために、表 5-5 のように GPIO を使用することを推奨します。CapSense センサーまたは LED 駆動のみを実装するデザインの場合、表 5-5 の制限は適用されません。

表 5-5. LED 駆動および CapSense センサー用の推奨 GPIO

LED 駆動用の推奨 GPIO	CapSense センサーおよび CMOD 用の推奨 GPIO
P0.1、P2.5、P2.3、P2.1、P4.1、P3.7、P3.5、P3.3、P3.1、P1.7、P1.5、P1.3、P1.1	P1.0、P1.2、P1.4、P1.6、P3.0、P3.2、P3.4、P3.6、P4.0、P4.2、P2.0、P2.2、P2.4、P0.0、P0.2、P0.4、P0.6、P0.3

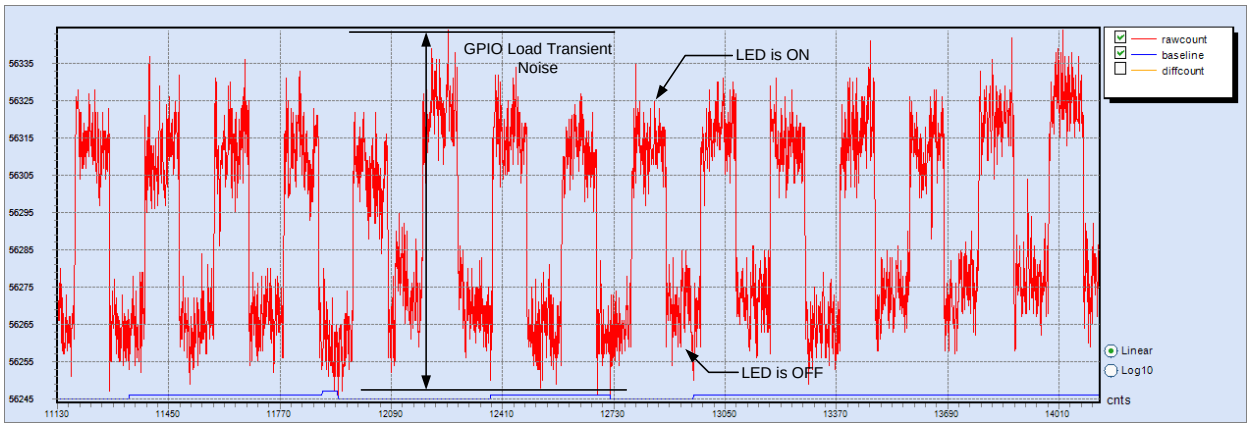
CapSense コントローラーのアーキテクチャによって偶数、奇数のポート ピン番号に対し電流量の制限が行われます。奇数ピンは、ピン番号が奇数であるどのポート ピンでも構いません。CapSense コントローラーでは、奇数ポート ピンの電流量が 100mA であれば、すべての奇数ポート ピンを通して引き出される電流量の合計は 100mA を超えないようにします。総電流量の制限に加えて、CapSense コントローラーのデータシートに定義されているように個々のポート ピンの電流制限もあります。

すべての CapSense コントローラーは、高電流シンク／ソース対応のポート ピンを備えます。ポート ピンからの高電流シンク／ソースを使用する場合、ノイズを最小限に抑えるために、デバイスのグランド ピンに最も近いポートを使用することを推奨します。

5.7 GPIO の負荷瞬時変化

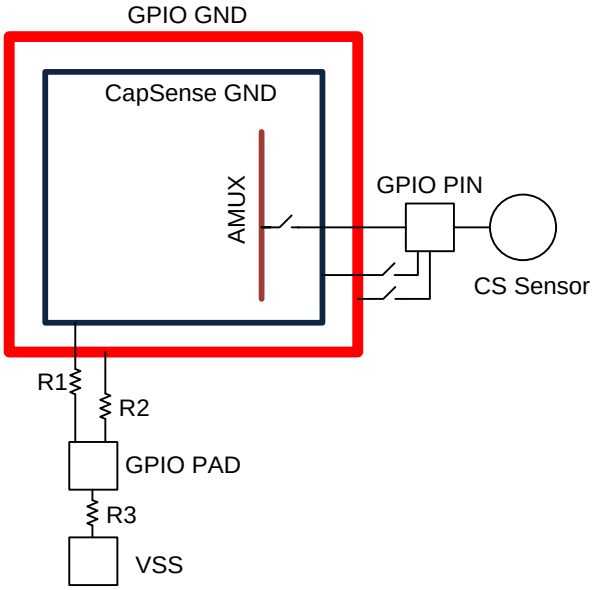
ポート ピンをストロング LOW に駆動して、GPIO が大きな電流 (>10mA) をチップのグランドに吸い込むとき、ノイズが CapSense システムに混入します。GPIO ピンを介してグランドへ流れる電流量の瞬時変化は、GPIO 負荷瞬時変化と呼ばれます。GPIO 負荷瞬時変化のため CapSense システムへ混入するノイズは、図 5-3 に示すように、GPIO 負荷瞬時変化ノイズと呼ばれます。本節では、ハードウェア技術を使って GPIO 負荷瞬時変化ノイズを低減し、ファームウェア技術を使ってこのノイズを補正する方法を説明します。

図 5-3. CapSense システム内の GPIO 負荷瞬時変化ノイズ



非ゼロのボンディング ワイヤ抵抗 R3 が存在しているため、電流が GPIO ピンを介して吸い込まれると、CapSense グランド (GPIO PAD) の電圧は 0 ではありません。非ゼロのグランド電位のため、LED が電流を吸い込むとき、センサーは完全に放電されません。そのため、センサーの raw カウントが増加します。

図 5-4. CY8C20xx7/S のグランド構造



注: R1、R2、R3 はボンディング ワイヤ抵抗です。

堅牢な CapSense デザインのために、最悪ケースの GPIO 負荷瞬時変化ノイズは指タッチ信号の 30%より小さい必要があります。CapSense システムでは、GPIO の状態が電流なしの状態 (例えば、全 LED がオフ) から最大電流の状態 (例えば、全 LED がオン) に遷移する最悪ケースにノイズが生じます。

GPIO 負荷瞬時変化ノイズはセンサーのスキャン分解能に応じて増加します。寄生容量が高い CapSense センサーや近接センサーは、SNR > 5:1 を達成するために、より高いセンサー スキャン分解能を必要とします。このようなシステムでは、GPIO 負荷瞬時変化の効果は、はるかに目立っています。いくつかの場合、GPIO 負荷瞬時変化ノイズは指タッチ信号より高く、センサーの誤トリガーを発生させることもあります。次の節では、GPIO 負荷瞬時変化ノイズを減少する方法を説明します。

5.7.1 GPIO 負荷瞬時変化ノイズ減少用のハードウェア ガイドライン

- センサーの C_P の低減

  センサーの C_P は、センサーのスキャン分解能のパラメーターを決めます。SNR > 5:1 を達成するために、C_P が大きければ、分解能のパラメーターは高くなる必要があります。高分解能のパラメーターを設定すると、GPIO 負荷瞬時変化ノイズの振幅

が増加します。そのため、「CapSense 入門」のデザイン ガイドに記載されているレイアウト ガイドラインにしたがってセンサーの C_P を最小限にすることを推奨します。

- LED のシンク電流の低減

  GPIO 負荷瞬時変化ノイズは、LED のシンク電流に正比例します。LED のシンク電流をデバイス データシートに記載されている範囲内にすることを推奨します。GPIO がデータシートに示されている最大値を超えた電流を吸い込む場合、外部トランジスタか、またはドライバーIC を使用します。

- LED に適切なピンの選択

  すべての CapSense コントローラーは、高電流シンク／ソース対応のポート ピンを提供します。ポート ピンからの高電流シンク／ソースを使用する場合、表 5-5 で推奨されているポートを使用します。

5.7.2 GPIO 負荷瞬時変化ノイズ補正用のファームウェア ガイドライン

GPIO 負荷瞬時変化に起因したセンサーの誤トリガーを防止するために、ルールベースのアルゴリズムでセンサーのベースラインを更新する必要があります。ベースラインの補正方法の 1 つは以下で説明します。

図 5-5 は GPIO 負荷瞬時変化により誤トリガーが発生する状態を示します。

1. ①では、センサー上に指タッチがなく、LED がオフです。
2. ②では、センサー上に指があり、raw カウントの変化は指閾値より高いです。
3. raw カウントの変化は指閾値より高いため、③では、LED はオンになります。
4. LED がオンになったとき、GPIO 負荷瞬時変化のため、raw カウントはさらに変化します。
5. ④では、指が離れていても、GPIO 負荷瞬時変化に起因した raw カウントの変化のため raw カウントは初期値に戻りません。この変化が指閾値より高い場合、LED は恒久的にオンになって、センサーの誤トリガーを意味します。

センサーと LED が恒久的にオンになることを防ぐために、センサーのベースラインを補正する必要があります。以下で詳しく説明します。

図 5-5. ベースラインが補正されないときの CapSense センサーの変数

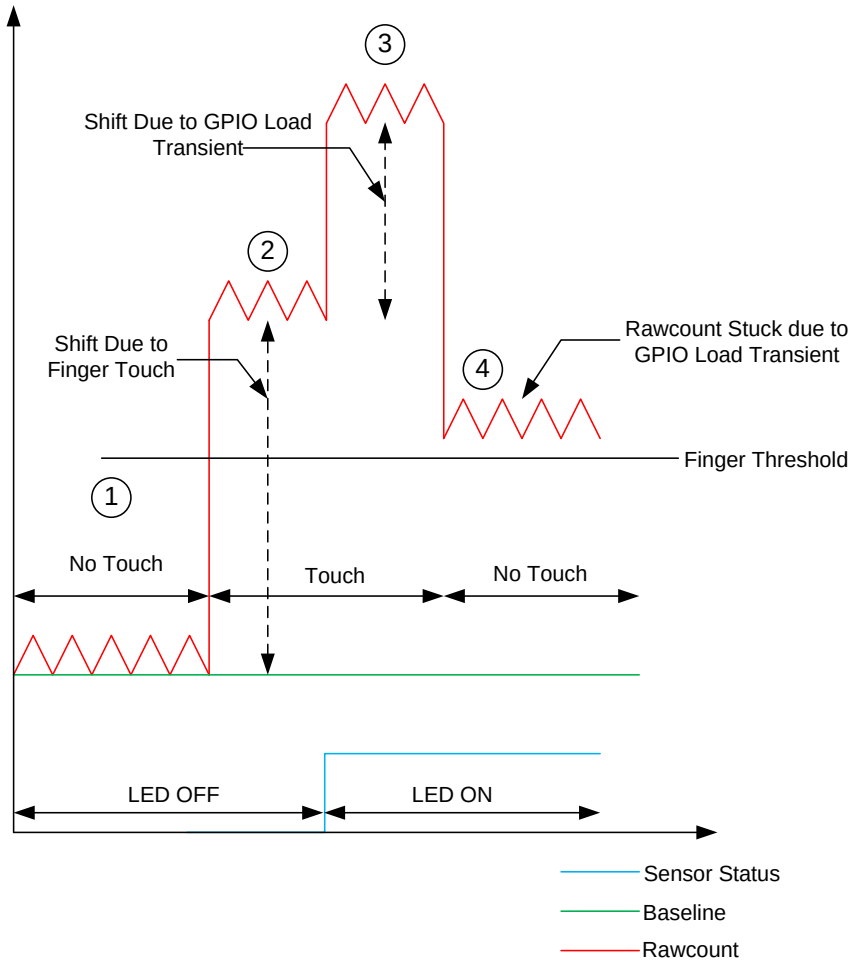
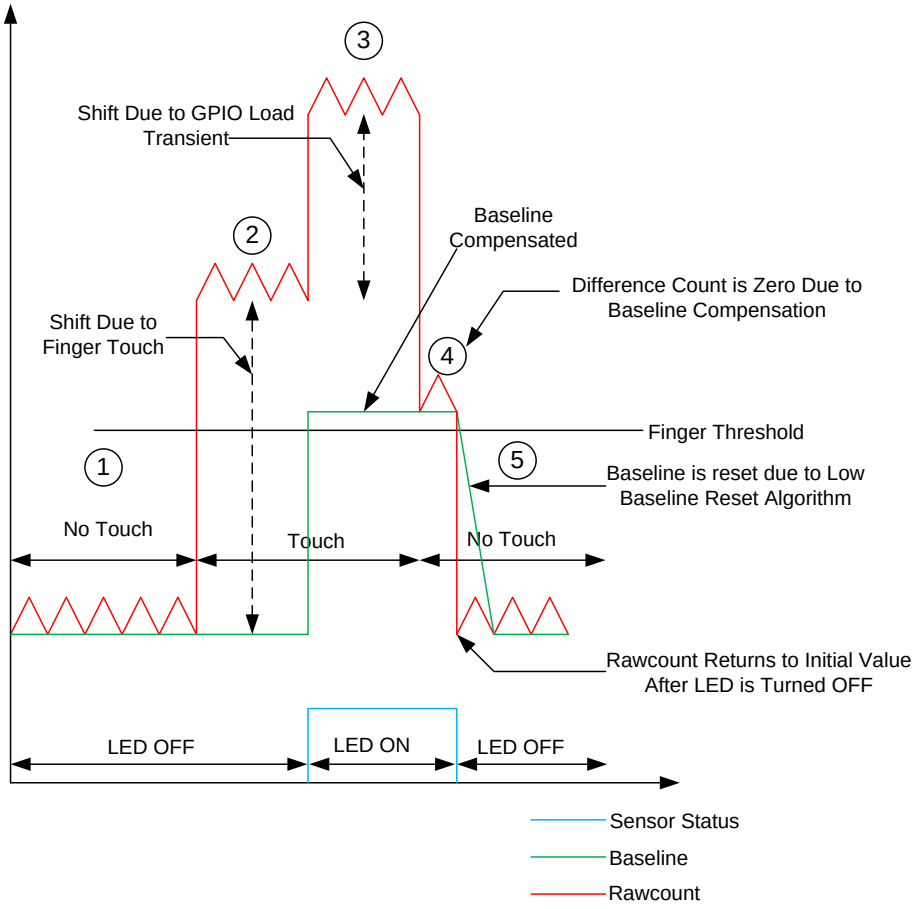


図 5-6 はセンサー ベースラインの補正により誤トリガーがなくなる状態を示します。

1. ①では、センサー上に指タッチがなく、LED がオフです。
2. ②では、センサー上に指があり、raw カウント (差分カウント) の変化は指閾値より高いです。
3. 差分カウントの変化は指閾値より高いため、③では、LED はオンになります。
4. LED がオンにされたとき、GPIO 負荷瞬時変化ノイズは以下のとおり計算されます: ノイズ = raw カウント (LED がオンの場合) – raw カウント (LED がオフの場合)。

   GPIO 負荷瞬時変化に起因したノイズ カウントはベースラインに追加され、その結果、指が離れると、差分カウントの値はゼロになり、LED はオフになります。
5. LED がオフになると、raw カウントは初期値になり、ベースラインは低ベースライン リセット アルゴリズムでリセットされます。

図 5-6. ベースラインが補正されたときの CapSense センサーの変数



5.8 PCB レイアウト ガイドライン

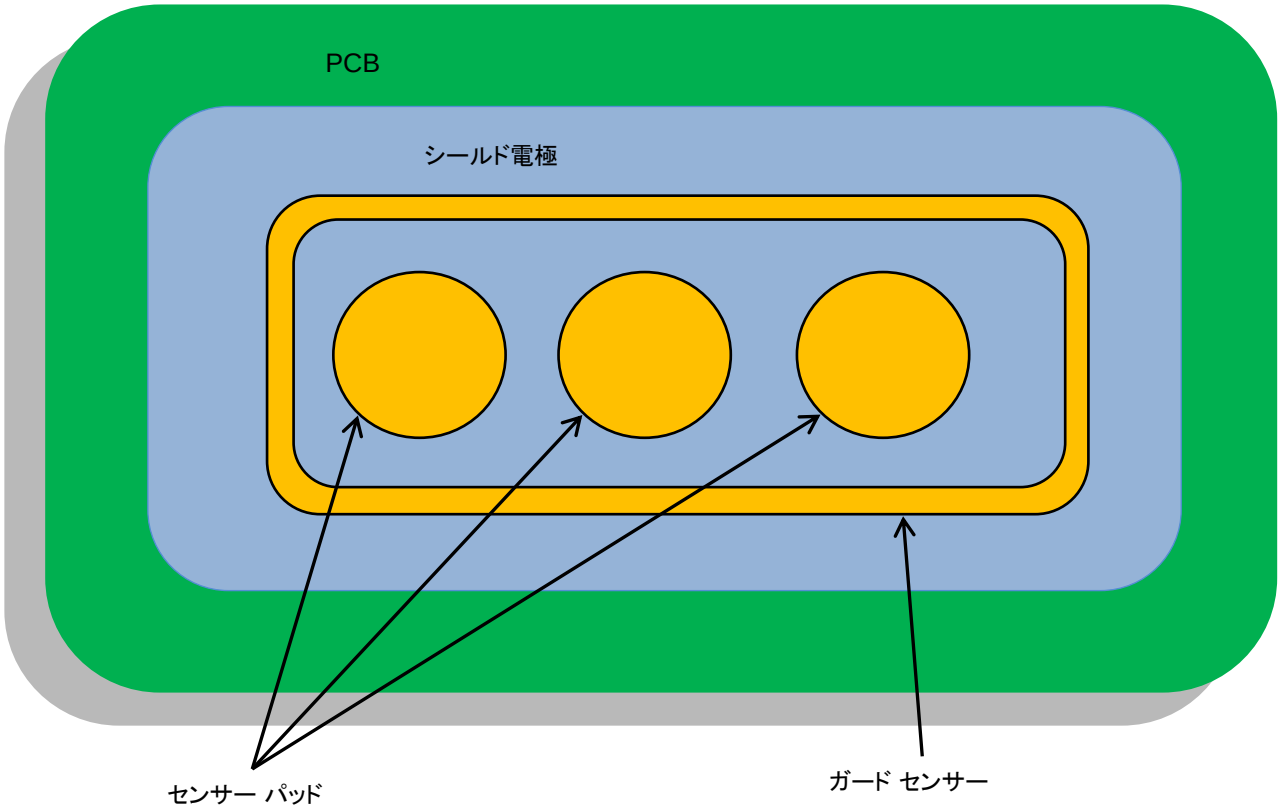
詳細な PCB レイアウト ガイドラインについては、「Getting Started withCapSense」を参照してください。

6. 耐液性デザイン上の注意事項

一部の CapSense 静電容量タッチ センシング アプリケーションは、水、またはケチャップや血などのその他の液体が存在する場合では信頼性の高い動作を必要とします。家電製品や車載アプリケーション、産業用アプリケーションは、水や氷、湿度の変化、またはその他の液体を伴う環境で動作する必要があるシステムの例です。そのようなアプリケーションには、シールド電極およびガード センサーが堅牢なタッチ センシングを提供します。

6.1 シールド電極とガード センサー

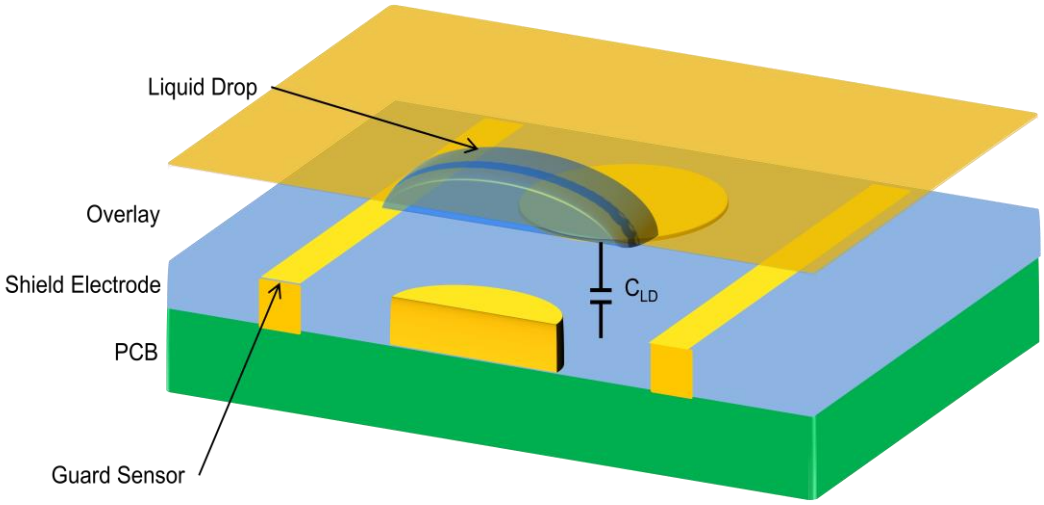
図 6-1. シールド電極とガード センサーを実装した PCB のレイアウト



6.1.1 シールド

シールド電極は、CapSense ボタン センサーが水滴や液漏れによる誤ったタッチを検出しないようにします。オーバーレイの表面に水滴が存在する場合、図 6-2 に示すように、シールド電極とセンサー パッド間のカップリングが C_{LD} の分だけ増加します。

図 6-2. 液滴が存在する場合の静電容量測定



C_{WD} = 液滴およびシールド電極間の静電容量

シールド電極の目的は、タッチ センサーの周囲に水の影響の軽減に役立つ電場を設けることです。シールド電極は、タッチ センサーの電圧をシールドに反映して動作します。

正常なシールド動作を確保するために以下のガイドラインに従ってください。

- 回路図
- レイアウト
- ファームウェア開発

6.1.1.1 回路図

シールド電極出力信号を駆動するために適切なピンを選択します。次のピンをシールド電極出力信号を駆動するために使用してください。

- ポート ピン: P0[0]、P1[2]、P0[2]、P2[2]、P2[4]

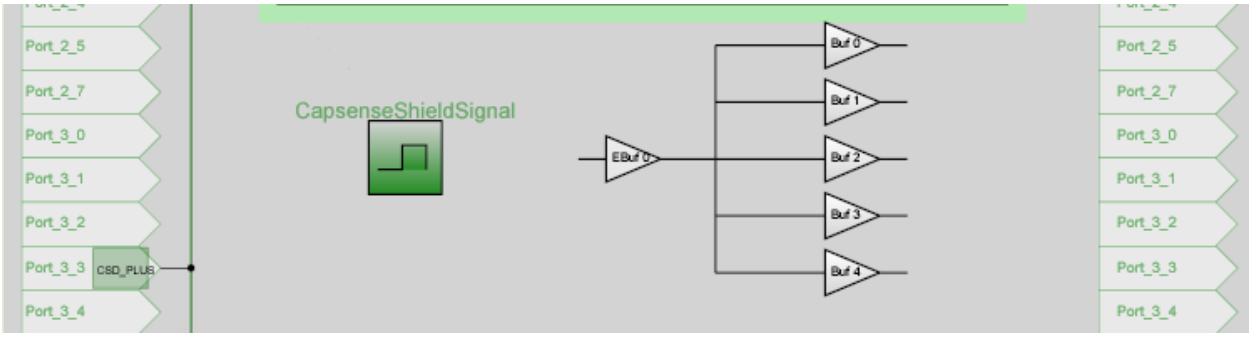
6.1.1.2 レイアウト

「CapSense 入門」に記載されているレイアウト ガイドラインに従ってください。

6.1.1.3 ファームウェア開発

図 6-3 に示すように、ファームウェア開発を最小限に抑えるシールド駆動 GUI があります。

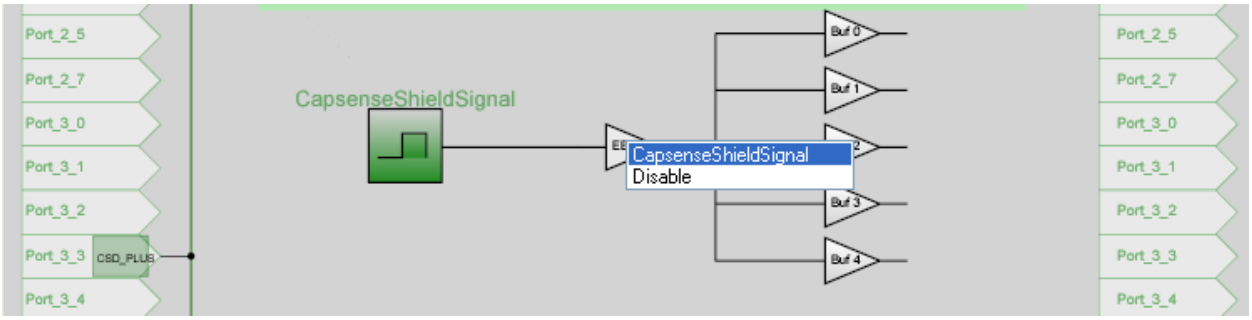
図 6-3. 被駆動シールド GUI (シールドが非アクティブ)



ステップ 1: GUI で CapSenseShieldSignal を有効にします。

図 6-4 に示すように、EnableShieldDriver スイッチである「Ebuf0」をクリックします。デフォルトでは「Disable」オプションが選択されています。**CapsenseShieldSignal** を選択してシールド ドライバーを有効します。

図 6-4. GUI での CapsenseShieldSignal の有効化



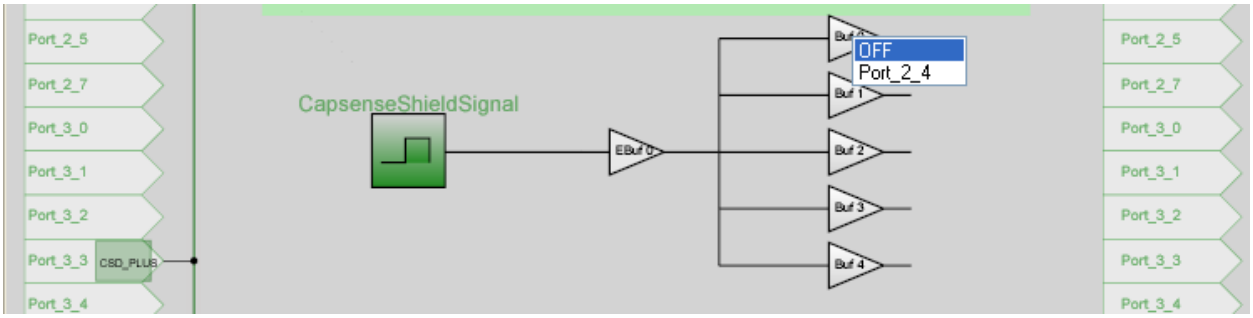
ステップ 2: シールド電極をシールド ピンに配線します。

シールド駆動バッファは Buf 0～Buf 4 の 5 つあり得ます。それらのマッピングは以下の表に示します。

シールド バッファ	出力ポート ピン
Buf 0	Port_2_4
Buf 1	Port_2_2
Buf 2	Port_0_2
Buf 3	Port_0_0
Buf 4	Port_1_2

シールド駆動バッファはデフォルトでオフです。希望のシールド駆動バッファを選択します。図 6-5 は、Buf 0 が有効になり、シールド信号が Port_2_4 で駆動されることを示します。

図 6-5. 出力選択の回路図表示



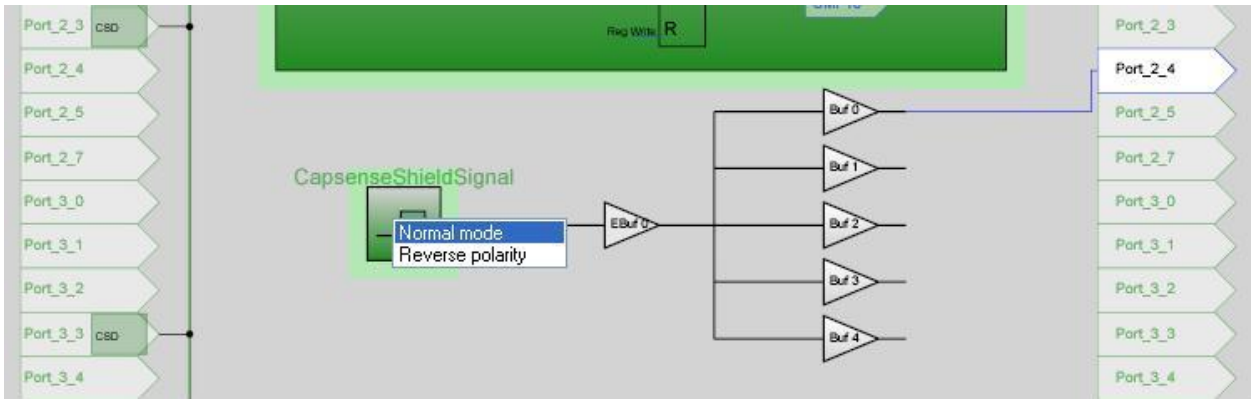
ステップ 3: シールド信号の極性を選択します。

極性の選択肢が 2 つあり、詳細については図 6-8 を参照してください。

- Normal Mode (デフォルト): シールド駆動信号はセンサー スキャン信号と同相です。
- Reverse Polarity: シールド駆動信号はセンサー スキャン信号と 180°位相がずれます。

Normal Mode は耐液性のデザインでは最高の性能を発揮します。

図 6-6. 極性選択の回路図表示



6.1.2 ガード センサー

図 6-7. シールドとガード センサーを搭載した PCB

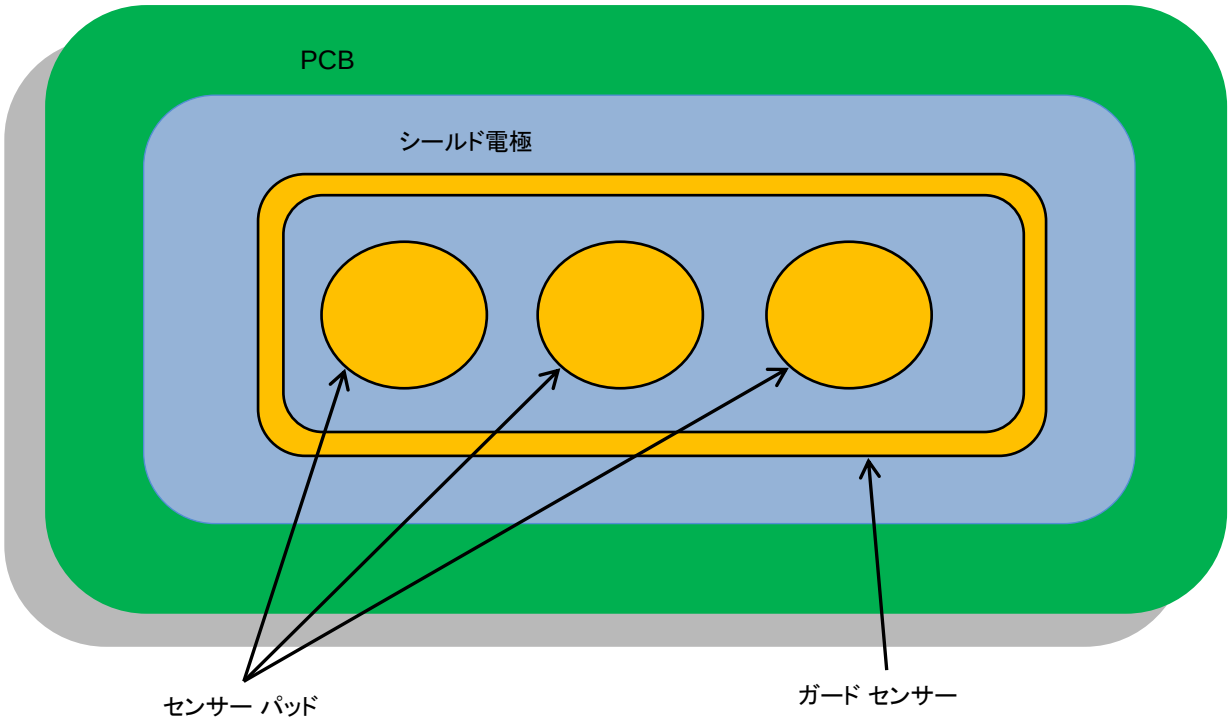
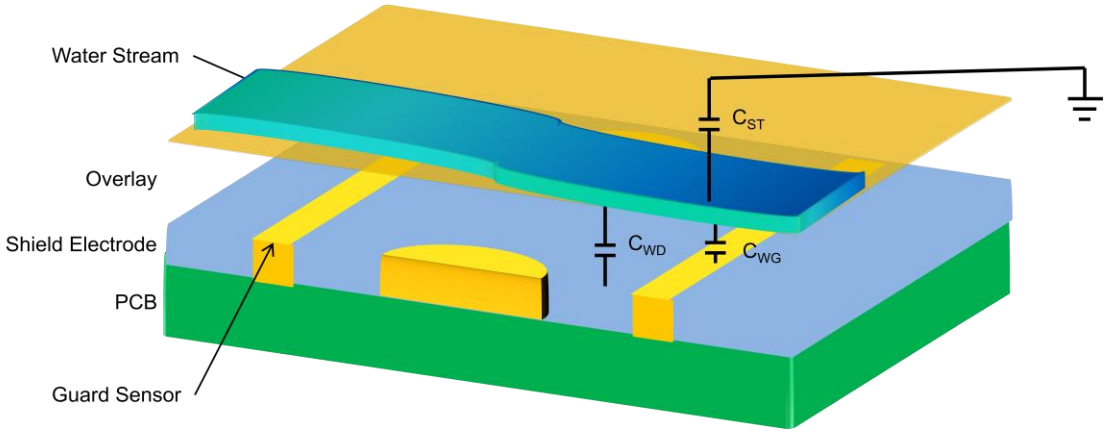


図 6-9 に示すように、ガード センサーは銅配線で、PCB 上の全センサーを囲み、連続的な水流、または液漏れの存在の検知に使用されます。感知表面に水流、または液漏れが存在する場合、図 6-8 に示すように、大静電容量 C_{ST} がシステムに追加されます。この静電容量は C_{WD} の数倍大きい可能性があります。このため、シールド電極の効果は完全にマスクされており、センサーにより測定される raw カウントは、指のタッチと同じかそれより高いです。この場合はガード センサーが役立ちます。水流を検知すると、その他のセンサーがトリガーしないようにします。

図 6-8. 水流または液漏れが存在する場合の静電容量の測定



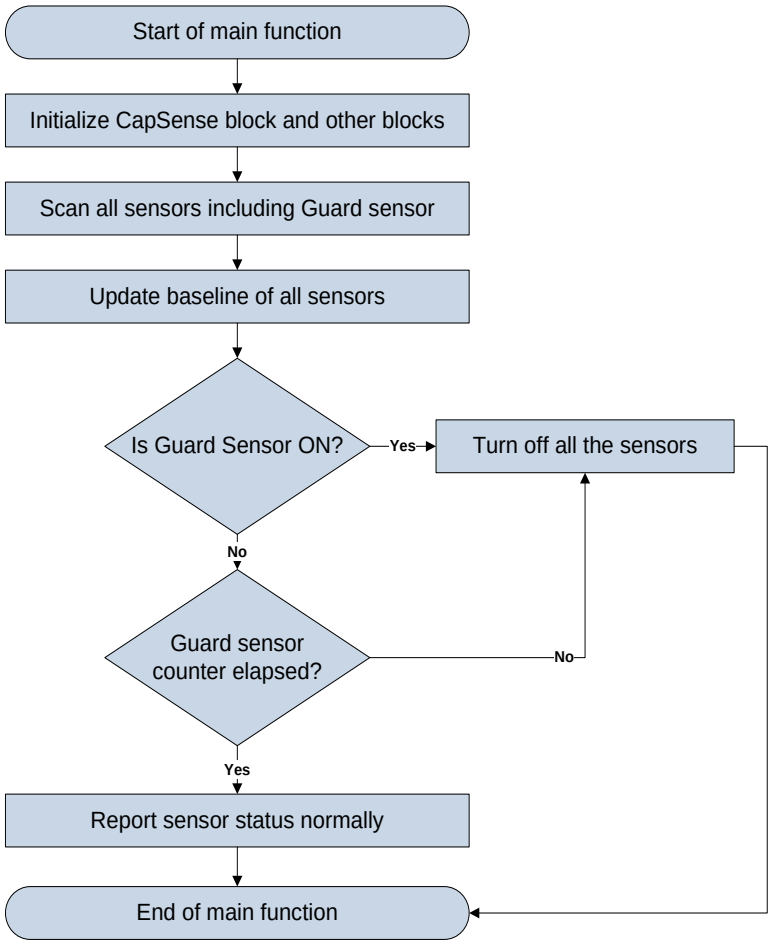
- C_{WD} - 水流／液漏れとシールド電極間の静電容量
- C_{ST} - 水流／液漏れとシステム グランド間の静電容量
- C_{WG} - 水流／液漏れとガード センサー間の静電容量

ガード センサーはファームウェアで実装する必要があります。ガード センサーを実装するために、CapSense ピンが 1 つ、カウンターが 1 つ (ハードウェア／ソフトウェア) 必要です。

基板上に水流や液漏れが存在する場合、ガード センサーがそれを検知して、タッチ センサーの処理ロジックを無効にします。追加のガード センサー「デッド」タイムにより、早過ぎるセンサーの解除を防止します。液体が無くなると、ガード カウンターはタッチ処理を短時間で抑制します(ガード センサー カウンター)。これにより、基板上に残っている水や液体による誤タッチの検出をなくします。

図 6-9 はファームウェアでのガード センサーの実装方法を示すフロー チャートです。

図 6-9. ガード センサー実装フロー チャート



6.2 設計上の推奨事項

次のシステム レベルと PCB レイアウトに関する推奨事項は、液体や湿潤環境にさらされる CapSense システムに適用されます。

- シールド電極銅ハッチングに関する推奨事項:
  - ☐ 最上層 – 7mil の配線と 45mil のグリッド (15%充填)
  - ☐ 最下層 – 7mil の配線と 70mil のグリッド (10%充填)
  - ☐ ボタン間のシールド電極の幅は少なくとも 10mm
- 水滴や液漏れがセンサーから自然に消えて、大きな水滴が蓄積しないように、センサーの表面を垂直または水平に対してある角度を持たせて設置します。
- 撥水性で非吸収性のオーバーレイ素材を使用します。これにより、デバイス パネル上の水の筋および水膜を最小限に抑えます。海水のように導電性の高い水の場合、これは特に重要です。
- アプリケーションが連続的な水流や大きい液漏れにさらされる可能性がある状況では、ガード センサーを使用する必要があります。デバイスがさらされるのは雨、蒸気や多湿環境のみである場合、ガード センサーは必要ありません。

7. 近接センシング デザインに関する注意事項

近接センサーは、静電容量のタッチ面に接触する前に、手またはその他の導電性の物質の存在を検知します。暗闇で車載用オーディオ システムを操作する場合を想像してください。近接検知は、指が近づくことによりシステムを有効にし、ライトアップします。たとえば、オーディオ システムのボタンは、ユーザーの手が近づくとバックライト LED を光らせます。

7.1 近接センサーの種類

7.1.1 ボタン

大きな C_P および小さな差分カウントを持つボタンは、近接センサーとして機能できます。ボタンとして実装された近接センサーの感度は、通常の静電容量式タッチ センシングのボタンよりはるかに高いです。

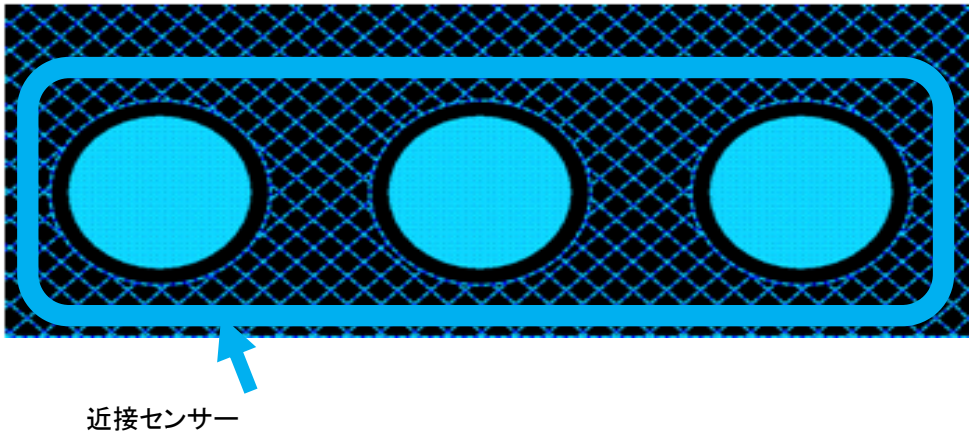
7.1.2 ワイヤ

単一のワイヤは近接センサーとして効果的に機能します。手の検知は電界の変化による静電容量の変化に依存するため、ワイヤの周囲の電界に影響を与える浮遊静電容量または物体は近接センサーの範囲に影響を与えます。ワイヤ センサーの使用は製造コストや複雑さのため、大量生産用の最適なソリューションではありません。

7.1.3 PCB 配線

長い PCB 配線は近接センサーを形成できます。配線は、直線、または図 7-1 に示すようにシステムのユーザー インターフェースの周囲を囲む場合があります。この方法は大量生産には適しますが、ワイヤ センサーほどの感度は高くありません。

図 7-1. PCB 配線を使用した近接センサー



7.1.4 センサー連動

近接センサーを実装するもう 1 つの方法は、センサーを一緒に連結することです。これは、PSoC Designer における内部アナログ マルチプレクサ バスからのセンサーを接続するファームウェアを使用して、複数のセンサーを 1 つの大きなセンサーに結合することにより実現されます。この方法を利用する際は、デザインの C_P 限度を超えないように注意してください。

7.2 設計上の推奨事項

- シールド電極を効果的に使用することにより、近接センサーの検知距離を拡大できます。これは金属が存在する場で近接センサーを動作する必要がある場合に役に立ちます。
- ワイヤ センサーは、シールド電極から遠く離れた場所に配置できるため、シールド電極の効果がより有益になります。
- 近接センサー内の大きなベタ グランドのエリアは、感度を低下させるので、それを避けてください。
- スキャン速度を遅くすると、あらゆる距離においても感度が増加します。近接センサーでは、スキャン速度はきわめて遅くしてください。

8. 低消費電力設計上の注意事項

消費電力はマイクロコントローラー デザインの重要な問題です。CapSense コントローラーの平均消費電流を低減するいくつかの技術のなかで、スリープ モードの適用は最も共通の方法です。何の機能も実行する必要がない場合、CapSense コントローラーは、携帯電話で一定のアイドル時間の後バックライトが暗くなるのと同じように、スリープ モードになります。スリープ モードへの移行は、デバイスの平均消費電流を削減するためです。これは、すべてのバッテリー使用アプリケーションでは重要な対応です。

CapSense コントローラーは、CPU_SCR0 レジスタのスリープ ビット (ビット 3) に「1」を書き込むとスリープ モードに入ります。これは、M8C_Sleep マクロを呼び出すことで実行されます。

スリープ モード中:

- 中央の CPU が停止
- IMO が無効
- バンドギャップ リファレンスの電源が切れた
- フラッシュ メモリ モデルが無効

電源電圧モニターと 32kHz 内部発振器のみが動作します。

基準のスリープ モード以外の省電力技術を以下に示します。

- I^2C スリープ モード (「I2C スリープ モードへの移行」の節を参照してください)
- CapSense (PSoC) アナログ ブロック リファレンスの無効化
- CT と SC ブロックの無効化
- CapSense (PSoC) アナログ出力バッファの無効化
- 駆動モードのアナログ HI-Z 設定

スリープ モードは、デザインに悪影響を与える場合があります。注意を怠ると、想定外の状況が発生することがあります。PSoC を必要に応じてスリープ モードから正しく復帰させる必要があり、またユーザーはデバイスがスリープ モードであり追加処理を行うことを認識しておく必要があります。

8.1 その他の省電力技術

スリープ モード適用を除き、他のすべての省電力技術はアプリケーション ベースです。その内には意外な結果を引き起こすものもあります。以下の節でそれぞれの技術を詳細に説明します。

8.1.1 駆動モードのアナログ HI-Z 設定

CapSense コントローラーの駆動モードの状態は消費電力に影響を与えることがあります。システムに悪影響を与えないピンでのみ、駆動モードの変更が可能です。ライン グリッチを発生させないように変更シーケンスを実施します。この変更シーケンスは現時点のピンの駆動モードとポート データ レジスタの状態によって決まります。CapSense コントローラーの駆動モードの構造のため、HI-Z またはストロング駆動モードの間で切り替える際に、ピンは一時的に抵抗プルアップまたは抵抗プルダウンの駆動モードのどちらかに入る必要があります。一時駆動モードは、ピンの直前の値と反対になります。したがって、ピンが HIGH に駆動されていれば、一時駆動モードは抵抗プルダウンでなければなりません。これにより、ピンの駆動モードが抵抗のものではなく、グリッチ発生の可能性を排除します。

駆動モードは、スリープ モードに入る前にソフトウェア内で手動でセットします。PRTxDM0、PRTxDM1、および PRTxDM2 の 3 つのレジスタが駆動モードを制御します。各レジスタから 1 ビットをピンに割り当てます。したがって、1 つのピンの駆動モードを変更するには、3 つのレジスタへの書き込みが必要です。しかし、同じ 3 つのレジスタの書き込みにより、ポート全体が変更可能であるため、これは便利です。アナログ HI-Z の適切なピット パターンは 110b です。次のコードを使用して、最初に抵抗プルダウンの駆動モードにしてから、ポート 0 をストロングからアナログ HI-Z に設定します。

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

8.1.2 まとめ

以下のコードは、28 ピンのデバイスの標準スリープ準備シーケンスの例です。このシーケンスでは、割り込みは無効にされ、アナログ回路はオフになり、すべての駆動モードはアナログ HI-Z に設定されてから、割り込みは再度有効にされます。

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

8.1.3 スリープ モードの I²C スレーブの推奨実装方法

I²C がスリープ モードの場合、I²C バスがロックしない、または破損したトランザクションが発生しないように、特定の実装ガイドラインに従わないといけません。

8.1.3.1 I²C スリープ モードへの移行

一般的には、I²C スレーブはスリープ モードに入る前に、FORCE_NACK モードにする必要があります。スリープ モードを正常にするために、以下の手順に従ってください。

- I2C_BP_EZ_CFG レジスタの CLK_STRETCH_EN ビットにより、動作モード (ウェイクアップのためのクロックストレッチまたはスリップ中の NACK) を選択します。
- I2C_XCFG レジスタの FORCE_NACK ビットをセットします。
- 論理「1」になるかどうかを確認するために I2C_XSTAT.READY_TO_NACK ステータス ビットをポーリングします。
- SLP_CFG2 レジスタの I2C_ON ビットをセットします。
- M8C_Sleep 関数を呼び出します (これは、CPU_SCR0 レジスタの SLEEP ビット (ビット 3) をセットします)。

注: 32 バイトの I²C バッファのスリープとディープ スリープ モード時のデータ保持は、SLP_CFG2 レジスタの I2C_ON ビットをアサートすることで、I²C ブロックへの電源が入っているときのみに保証されます。

8.1.4 スリープ モードの混乱

CapSense コントローラーは、リセットまたは割り込みによってスリープを終了できます。CapSense コントローラーのリセットは 3 種類があります: 外部リセット、ウォッチドッグ リセット、およびパワーオン リセット。これらのリセットの 1 つを使用して CapSense コントローラーをスリープ モードから復帰させます。リセットのデアサートの後、CapSense コントローラーは *Boot.asm* からコードの実行を開始します。CapSense コントローラーをウェイクアップするのに使用可能な割り込みはスリープ タイマー、低電圧モニター、GPIO、アナログ カラム、および非同期です。割り込みにより CapSense コントローラーをウェイクアップする場合と、スリープ中にデジタル通信を行おうとする場合に、スリープ モードの混乱が問題になります。これらの検討事項については、次の節で詳しく説明します。

8.1.5 保留中の割り込み

割り込みが保留され、有効化され、そして、CPU_SCR0 レジスタの SLEEP ビットへの書き込み後に発生するように予定されていれば、システムはスリープにはなりません。これでも命令は実行されますが、CapSense コントローラーは SLEEP ビットをセットしません。その代わり、割り込みが処理され、事実上 CapSense コントローラーは、スリープ命令を無視することになります。これを回避するには、スリープ準備が行われている間に割り込みを全体的に無効にし、その後 SLEEP ビットを書き込む直前に割り込みを再度有効にしなければなりません。

8.1.6 グローバル割り込みの有効化

割り込みで CapSense コントローラーをウェイクアップするには、グローバル割り込みイネーブル レジスタ (CPU_F) を有効にすることは不要です。以下の例のように、割り込みによるスリープ モード終了の唯一の要件は、INT_MSKx レジスタ内の正しい割り込みマスクを使用することです。グローバル割り込みが無効にされた場合、CapSense コントローラーをウェイクアップする ISR は実行されませんが、CapSense コントローラーはスリープ モードを終了します。

この場合、ISR を実行するには、保留中の割り込みを手動でクリアするか、またはグローバル割り込みを有効にする必要があります。割り込みは INT_CLRx レジスタ内でクリアされます。

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

8.2 ウェイクアップ後の実行シーケンス

リセットにより CapSense コントローラーをウェイクアップする場合、ブート コードの最初から実行されます。割り込みサービス ルーティンにより CapSense コントローラーをウェイクアップする場合、スリープ命令の直後の命令を最初に実行します。これは、スリープ命令直後の命令は、CapSense コントローラーが完全にスリープ モードになる前にプリフェッチされたためです。そのため、グローバル割り込みが無効な場合、命令実行はスリープが開始される前に戻って継続します。

8.2.1 PLL モードの有効化

PLL モードが有効になった場合、スリープ モードに入る前に、CPU 周波数を最小の 3MHz まで下げる必要があります。これは、CapSense コントローラーが復帰して、再度有効になった後に、PLL が再度ロックしようとして、毎回オーバーシュートの状態になるためです。さらに、デバイスが正常に動作するように、スリープ モードを終了した後、10ms 待つ必要があります。これは、スリープ モードと PLL を使用するにはソフトウェアは 3MHz で実行できなければならないことを意味します。OSC_CR0 レジスタへの簡単な書き込みによって、CPU を減速できます。しかし、このレジスタは SYSCLK の分周器をセットするだけなので、異なった SYSCLK を備えたデバイス ファミリによって CPU 速度が異なります。通常、SYSCLK は 24MHz です。

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

8.2.2 グローバル割り込みの有効化の実行

スリープ ビット書き込みの命令境界上での割り込みを回避してください。これにより、スリープ コマンドが「割り込みから戻る」(RETI) 命令で実行される場合、スリープ モード移行の準備は無視される可能性があります。これを防ぐために、割り込みをスリープ準備前に一時的に無効にしてから、スリープに入る前に再度有効にします。グローバル割り込み命令のタイミングのため、次の命令 (この場合は SLEEP ビットをセットする命令) の間は割り込みは発生しません。

8.2.3 スリープ モードの I²C スレーブの推奨実装方法

I²C がスリープ モードの場合、I²C バスがロックしない、または破損したトランザクションが発生しないように、特定の実装ガイドラインに従わないといけません。

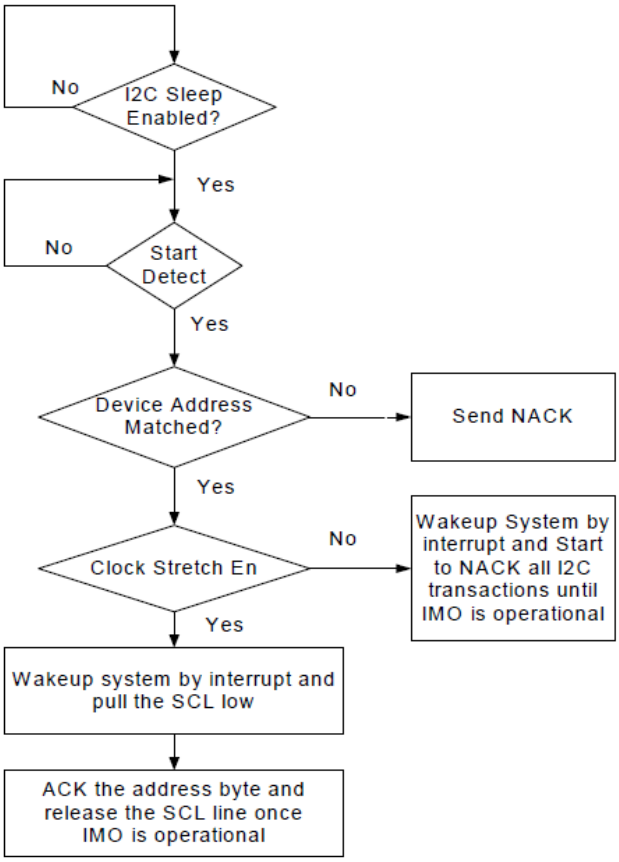
8.2.3.1 HW アドレス一致による I²C スリープ モードからのウェイクアップ

I²C によりウェイクアップを有効にするために、I²C スレーブ ブロックがアドレスが一致した場合のみにシステムをウェイクアップするように HW Addr EN ビットをセットします。I²C ブロックは、システムがスリップ モードの場合、I²C バス上のトランザクションに応答します。ただし、システムがスリップ モードになると、システム クロックがシャットダウンになります。

したがって、I²C ブロックは、I²C バス上のトランザクションに応答すれば、動作するシステム クロックがなくなります。結果として、着信 SCL クロックは、バス上のトランザクションに応答するためにブロック用のクロックとして使用されます。

アドレスが一致しているときの動作は CLK_STRETCH_EN の設定次第です。このビットが HIGH に設定された場合、SCL は IMO が動作可能になるまでプルダウンされます。このビットが LOW に設定された場合、スレーブは、アドレス指定されたすべての I^2C トランザクションを CPU がウェイクアップするまで NACK して ACK ビットをセットします。クロック ストレッチモードが無効になる場合、IRQ ウェイクアップの後、いかなる他の START 条件も IMO が動作可能になるまで、デバイスによって認識されません。この間、すべてのトランザクションは NACK を受信します。図 8-1 は I^2C によるウェイクアップシーケンスを表します。

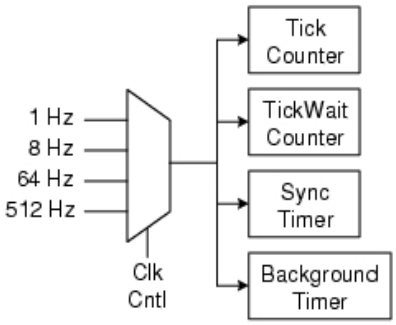
図 8-1. ウェイクアップ シーケンス



8.2.4 スリープ タイマー

CapSense コントローラーは、スリープ タイマーとスリープ タイマー ユーザー モジュールを提供します。これらは CapSense コントローラーがスリープ モードに入ったときに使用され、両方とも同様の機能を実行します。実際のスリープ タイマーは、常に電源が切られない内部低速発振器を使用します。タイマーは 1Hz、8Hz、64Hz または 512Hz の選択可能な周波数で割り込みを発生させます。CapSense コントローラーを定期的に有効にして何かの処理やアクティビティの確認を行うのに有用です。一例は、センサーをスキャンするために定期的にウェイクアップすることです。スリープ タイマー ユーザー モジュールは追加の機能を生成するためにスリープ タイマーを使用します。この機能は、定期的な割り込みを発生させるバックグランド ティック カウンター、プログラム ループ用遅延機能、設定可能なダウン カウンター、およびループ タイムを制御するループ ガバナーがあります。この機能の簡単なブロック図を図 8-2.に示します。

図 8-2.スリープ タイマー ユーザー モジュール ブロック図



9. リソース

9.1 ウェブサイト

本節で説明したすべての参照資料を入手するために、サイプレスの CapSense コントローラー ウェブサイトをご覧ください。

CapSense CY8C20xx7/S デバイス ファミリのさまざまな技術資料は、CY8C20xx7/S ウェブページをご覧ください。

9.2 データシート

CapSense CY8C20xx7/S デバイス ファミリのデータシートは、www.cypress.com にて入手できます。

- CY8C20xx7/S データシート

9.3 テクニカル リファレンス マニュアル

サイプレスは、トップ レベルのアーキテクチャ図やレジスタおよびタイミング図などの、CapSense コントローラー機能情報を迅速かつ容易に参照できることを目的として以下のテクニカル リファレンス マニュアルを作成しました。

- CY8C20xx7/S テクニカル リファレンス マニュアル (TRM)

9.4 開発キット

9.4.1 CY8C20xx7/S QuietZone スターター キット

図 9-1 に示すように CY8C20xx7/S QuietZone スターター キットは CY8C20247S の 16 ピン QFN パッケージに基づく 0.75 インチの四角形基板です。

図 9-1. CY8C20xx7/S QuietZone スターター キット



CY8C20xx7/S QuietZone スターター キット ハードウェアは以下の特長があります。

- 12 個の静電容量センシング入力
- MiniProg1/3 への簡単な接続用の 1 個の 5 ピン ヘッダ (HD1)
- 1 個の被駆動シールド出力
- ビジュアル フィッドバック用の 2 個の LED

キットはサイプレスのモジュール パートナーArtaFlex のウェブサイト (www.artaflexmodules.com/quietzone) から入手できます。

9.4.2 ユニバーサル CapSense コントローラー キット

CY8C20xx7/S ファミリはオンチップ デバッグ (OCD) 機能がありません。デバッグ プラットフォームが必要になる場合、CY8C20xx6A ユニバーサル CapSense コントローラー キットは使用できます。ユニバーサル CapSense コントローラーキットは、事前定義された制御回路およびプラグイン ハードウェアを備えており、プロトタイピングとデバッグを簡単にします。チューニングとデータ取得用にプログラミングと I²C-USB ブリッジ ハードウェアが同梱されています。

- CY3280-20xx6A ユニバーサル CapSense コントローラー

9.4.3 ユニバーサル CapSense モジュール基板

9.4.3.1 シンプル ボタン モジュール基板

CY3280-BSM シンプル ボタン モジュールは 10 個の CapSense ボタンと 10 個の LED で構成されます。このモジュールは、CY3280 ユニバーサル CapSense コントローラー基板と接続します。

9.4.3.2 マトリックス ボタン モジュール基板

CY3280-BMM マトリックス ボタン モジュールは、4×4 マトリックス形式として構成される 8 個の LED と 8 個の CapSense センサーから成ります (すなわち、物理的ボタン 16 個が形成されます)。このモジュールは、CY3280 ユニバーサル CapSense コントローラー基板と接続します。

9.5 サンプル ボード ファイル

サイプレスでは、サンプル回路図およびボード ファイルを提供しています。これらは、PCB 設計プロセスを迅速に完了するために参照用としてご利用いただけます。

- CY8C20xx7/S での I²C ヘッダ付きボタン デザイン
- CY8C20xx7/S での I²C ヘッダ付きボタンとスライダー デザイン

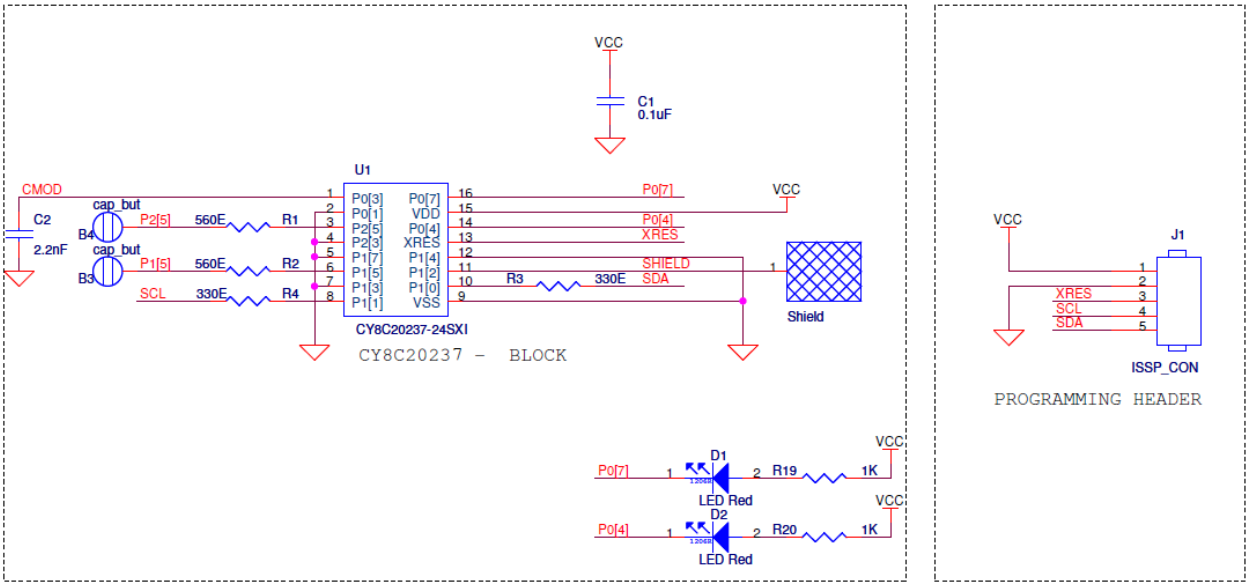
注: ボード ファイル (回路図、レイアウトおよびガーバー ファイル) は、本文書のランディング ページに含まれます。

図 9-2 と図 9-3 は基板回路図を示します。

以下をサポートをするために、この回路図は設計されています。

- CY8C20237-24SXI デバイスの P2[5]および P1[5]ピンに割り当てられる 2 個の CapSense センサー
- D1 と D2 の LED を駆動するために CY8C20237-24SXI の P0[7]と P0[4]ピンに接続される 2 個の GPIO
- シールド面を駆動するために P1[2]ピンに接続されるシールド電極
- ヘッダ J1 に接続される CY8C20237-24SXI のプログラミング ラインおよび I²C SDA／SCL 信号

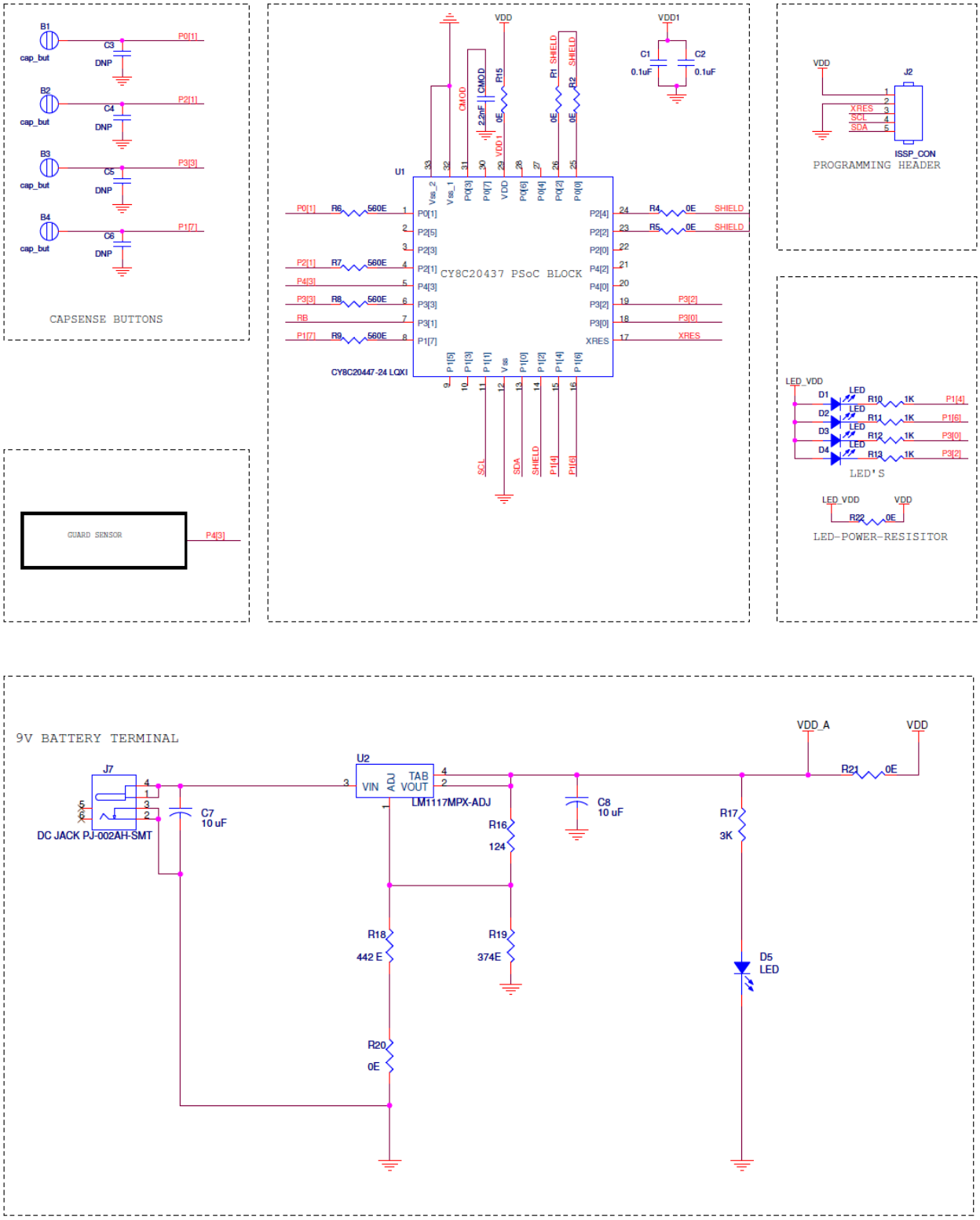
図 9-2. CY8C20237 での I²C ヘッダ付きボタン デザイン - 基板回路



以下の回路図は、次をサポートするために設計されます。

- CY8C20437-24LQXI デバイスの P0[1]、P2[1]、P3[3]、P1[7]ピンに割り当てられる 4 個の CapSense センサー
- CY8C20437-24LQXI デバイスの P4[3]ピンに接続されるガード センサー
- D1、D2、D3、D4 の LED を駆動するために CY8C20437-24LQXI の P1[4]、P1[6]、P3[0]、P3[2]に接続される 4 個の GPIO ピン
- プログラム ヘッダーJ2 による CY8C20437-24LQXI のプログラミング
- ヘッダーJ7 に接続できる 9V DC 入力

図 9-3. CY8C20437 での I²C ヘッダ付き耐液性のボタン デザイン



9.6 PSoC Programmer

PSoC Programmer は PSoC デバイスをプログラムするための柔軟性のある統合プログラミング アプリケーションです。PSoC Designer と PSoC Creator™と併用して PSoC デバイスにさまざまなデザインをプログラムできます。

PSoC Programmer には、プログラマとブリッジ デバイスを用いて専用アプリケーションを設計するための API を備えたハードウェア層が含まれます。PSoC Programmer のハードウェア層は、COM ガイド文書だけでなく、C#、C、Perl および Python の言語でサンプル コードとしても説明されています。

9.7 CapSense データ表示ツール

CapSense の設計中には、チューニングやデバッグの目的で、関連する CapSense データ (raw カウント、ベースライン、差分カウントなど) の監視が必要となる場合が何度もあります。CapSense データ表示とログに適切なツールの決定と使用に有用な情報は、アプリケーション ノート「AN2397 – CapSense Data Viewing Tools」を参照してください。

9.8 PSoC Designer

サイプレスは、専用の統合設計環境 (IDE) である PSoC Designer を提供します。PSoC Designer を使用すると、アナログおよびデジタル ブロックの構成、ファームウェアの開発および設計のチューニングが可能になります。アプリケーションは、完全に特性化されたアナログとデジタルの機能 (CapSense を含む) のライブラリを使用し、ドラッグ アンド ドロップの設計環境で開発されます。PSoC Designer には、C コンパイラとプログラマが組み込まれています。複雑なデザイン用には専用コンパイラがあります。

9.9 サンプル コード

サイプレスは、設計を速やかに完了するために大量のサンプル コードを提供します。

- CapSense コントローラー サンプル コード デザイン ガイド

9.10 デザイン サポート

サイプレスはさまざまなデザイン サポート チャネルを提供しており、お客様の CapSense ソリューションの成功を確実にしています。

- 知識ベース記事: 製品ファミリ別の技術情報記事を参照したり、CapSense についてのさまざまなトピックを検索できます。
- CapSense アプリケーション ノート: 本文書で提供された情報の上に築かれた、幅広いアプリケーション ノートを参照します。
- ホワイト ペーパー: 上級の静電容量タッチ インターフェースに関するトピックについて学びます。
- サイプレス開発コミュニティ: サイプレス テクニカル コミュニティに参加し、情報を交換します。
- CapSense 製品セレクタ ガイド: サイプレスの CapSense 製品ラインの完全な製品群をご覧になれます。
- ビデオ ライブラリ: チュートリアル ビデオで素早く学習できます。
- 品質および信頼性: サイプレスはお客様の満足を第一に考えます。品質ウェブサイトでは、信頼性および品質に関するレポートをご覧になれます。
- テクニカル サポート: 世界レベルのテクニカル サポートがオンラインで利用いただけます。

10. 用語集

AMUXBUS

入出力ピンを複数の内部アナログ信号に接続する PSoC 内にあるアナログ マルチプレクサ バスです。

SmartSense™ 自動チューニング

設計フェーズの後で最適な性能のために、センシング パラメーターを自動設定し、システム、製造および環境変化に対し連続的に補正する CapSense アルゴリズムです。

ベースライン

センサーに人の指で触らないときの raw カウントの傾向を推定しファームウェア アルゴリズムから得られる値です。ベースラインは raw カウントの突然の変化に敏感性が低く、差分カウントを計算するための基準点を提供します。

ボタン／ボタン ウィジェット

センターに対応しており、センサーのアクティブ状態または非アクティブ状態 (すなわち、2 つだけの状態) を報告するウィジェットです。例えば、センサー上の指の「タッチあり」または「タッチなし」の状態を検出できます。

差分カウント

raw カウントとベースラインの差です。差分値が負であるかまたはノイズ閾値未満である場合、差分カウントは常に 0 に設定されます。

静電容量センサー

静電容量の変化によってタッチまたは近づいている物体に反応する導電体および基板 (プリント回路基板 (PCB) 上の銅ボタンなど) です。

CapSense®

サイプレスのタッチ センシング ユーザー インターフェース ソリューションです。業界 2 位に対して、4 倍の販売実績がある業界 No.1 ソリューションです。

CapSense メカニカル ボタン リプレースメント (MBR)

メカニカル ボタンを静電容量ボタンにアップグレードするサイプレスの構成可能なソリューションであり、センサー パラメーターの設定に必要な設計工数を最小限に抑え、ファームウェアの開発も不要とします。これらのデバイスは CY8CMBR3XXX および CY8CMBR2XXX のファミリを含みます。

重心／重心位置

スライダー分解能の指定した範囲内のスライダー上の指の位置を示す数です。この数は CapSense 重心計算アルゴリズムにより算出されます。

補正 IDAC

過剰なセンサー C_P を補正するために CSD により使用されるプログラム可能な定電流源です。この IDAC は変調 IDAC と違って、CSD ブロックでシグマ-デルタ変調器によって制御されません。

CSD

CapSense シグマ デルタ (CSD) は、静電容量センシング アプリケーション用に自己容量を測定するサイプレスの特許取得済み方法です。

CSD モードでは、センシング システムは電極の自己容量を測定し、指の有無を識別するために自己容量の変化が検出されます。

デバウンス

有効なタッチとなるためにタッチがある必要な連続スキャン サンプル数を定義するパラメーターです。このパラメーターは怪しいタッチ信号を排除するために役立ちます。

指のタッチは、差分カウントがスキャン サンプルの連続デバウンス数で [指閾値 + ヒステリシス] を超える場合にのみ報告されます。

被駆動シールド

シールド電極がセンサー スイッチング信号と同じ位相および振幅を持つ信号によって駆動され、耐液性を有効にするために CSD によって使用される技術です。

電極

PCB、ITO または FPCB 上のパッドや層などの導電材料です。電極は CapSense デバイスのポート ピンに接続され、CapSense センサーとして使用されるか、または CapSense の機能に関連した特定の信号を駆動するために使用されます。

指閾値

センサーの状態を確定するためにヒステリシスと一緒に使用されるパラメーターです。センサーの状態は、差分カウントが [指閾値 + ヒステリシス] を上回る場合、オンとして報告され、差分カウントが [指閾値 – ヒステリシス] を下回る場合、オフとして報告されます。

連動センサー

複数のセンサーを連動させ、単一センサーとしてスキャンする方法です。近接センシング用のセンサーの領域を増やし、消費電力を減少させるために用いられます。

システムが低電力モードにある場合の消費電力を削減するために、センサーは個別にスキャンされず、これらのすべてを連動して単一のセンサーとしてスキャンされ、時間を短縮させます。ユーザーがセンサーに触れる場合、システムはアクティブ モードに移行し、すべてのセンサーを個別にスキャンしてアクティブになったセンサーを検出します。

PSoC はファームウェアでセンサー連動をサポートし、これにより、複数のセンサーはスキャンのために AMUXBUS と同時に接続できます。

ジェスチャー

ジェスチャーはスワイプやピンチズームなどのユーザーの行動です。CapSense は事前に定義されたタッチパターンに基づいて異なるジェスチャーを識別するジェスチャー検出機能を備えます。CapSense コンポーネントでは、ジェスチャー機能はタッチパッド ウィジェットのみによってサポートされます。

ガード センサー

ボタン センサーと同様に PCB 上のすべてのセンサーを取り囲み、液体流を検出するために使用される銅配線です。ガード センサーがトリガーされると、ファームウェアは誤ったタッチを防ぐために、すべての他のセンサーのスキャンを無効にできます。

ハッチ フィル／ハッチ グランド／ハッチド グランド

静電容量センシングの PCB を設計する際、良好なノイズ耐性のために接地した銅面をセンサーの周りに配置する必要があります。しかし、ベタ グランドはセンサーの期待されない寄生静電容量を増加させます。そのため、グランドは

特別なハッチ パターンで充填する必要があります。ハッチ パターンはメッシュのように密接に配置され、交差されるラインがあり、充填率がラインの幅および 2 本のライン間の間隔によって決まります。耐液性の場合、シールド電極 として呼ばれるこのハッチ フィルはグランドの代わりにシールド信号で駆動されます。

ヒステリシス

システム ノイズに起因してセンサー状態の出力がランダムにトグルすることを回避し、センサーの状態を決定するために指閾値と一緒に使用されるパラメーターです。「指閾値」を参照してください。

IDAC (電流出力デジタル-アナログ変換器)

CapSense および ADC 動作用の PSoC 内のプログラマブルな定電流源です。

耐液性

水滴、液体流や霧が存在する環境でも確実に動作する静電容量センシング システムの能力です。

リニア スライダー

指の物理的な位置 (単一の軸で) を検出するために特定の直線状で配置された複数のセンサーを含むウィジェットです。

低ベースライン リセット

raw カウントが負のノイズ閾値を異常的に下回るスキャン サンプルの最大数を表すパラメーターです。低ベースライン リセットの値を超える場合、ベースラインは現時点の raw カウントにリセットされます。

手動チューニング

CapSense パラメーターの手動設定 (または手動チューニング) プロセスです。

マトリックス ボタン

マトリックス状で配置された 2 つ以上のセンサーから成り、垂直方向および水平方向に配置されるセンサーの交差部に人の指 (タッチ) の有無を検出するために使用されるウィジェットです。

M を水平軸上のセンサーの数と、N を垂直軸上のセンサーの数とすれば、マトリックス ボタン ウィジェットは M + N 本のポート ピンだけを使用して合計で M x N 個の交差部を監視できます。

CSD センシング方式 (自己容量) を使用する場合、このウィジェットは同時に 1 点のみの交差位置で有効なタッチを検出できます。

変調コンデンサ (CMOD)

自己容量センシング モードでの CSD ブロックの動作のために必要な外部コンデンサです。

変調器クロック

センサーのスキャン間に CSD ブロックから変調器出力をサンプリングするために使用されるクロック ソースです。このクロックが raw カウント カウンターにも供給されます。スキャン時間 (事前および事後処理時間を除く) は [$(2^N - 1)$/変調器クロック周波数] (N はスキャンの分解能) により計算されます 。

変調 IDAC

変調 IDAC はプログラマブルな定電流源であり、この出力は V_{REF} の AMUXBUS 電圧を維持するために、CSD ブロックのシグマ デルタ変調器の出力によって制御 (オン／オフ) されます。この IDAC によって供給される平均電流は、センサー コンデンサが引き出した平均電流に等しいです。

相互容量

ある電極 (例えば TX) と他の電極 (例えば RX) 間の静電容量は相互容量として知られています。

負のノイズ閾値

負の方向に出るスプリアス信号から通常のノイズを識別するために使用される閾値です。このパラメーターは、低ベースライン リセット パラメーターとともに使用されます。

raw カウントが負のノイズ閾値を超えない (すなわち、ベースラインと raw カウントの差 (ベースライン – raw カウント) が負のノイズ閾値未満である) 限り、ベースラインは raw カウントの変化を追跡するために更新されます。

負の方向でスプリアス信号をトリガーする可能性があるシナリオは次のとおりです。電源投入時にセンサーに指が触れるとき、センサーの近く配置される金属の物体を除去するとき、耐液性のある CapSense 製品の水分を除去するとき、および他の急激な環境変化があるときです。

ノイズ (CapSense ノイズ)

センサーがオフ状態 (タッチなし) のときにピーク ツー ピーク カウントとして測定される raw カウントの変化です。

ノイズ閾値

センサー用にノイズから信号を識別するために使用されるパラメーターです。[raw カウント – ベースライン] の差分がノイズ閾値を超える場合、おそらく有効な信号を示します。差分がノイズ閾値に下回る場合、raw カウントはノイズのみを含みます。

オーバーレイ

静電容量センサーをカバーしタッチ面として機能するプラスチックやガラスなどの非導電材料です。センサーがある PCB はオーバーレイの下に直接配置されるか、またはスプリングを介して接続されます。製品の筐体は常にオーバーレイになります。

寄生容量 (C_P)

寄生容量は PCB 配線、センサパッド、ビアおよびエア ギャップによって与えられるセンサー電極の固有容量です。寄生容量は CSD の感度を減らすため、望ましくないものです。

近接センサー

あらゆる物理的な接触なしに近くの物体の存在を検知できるセンサーです。

ラジアル スライダー

指の物理的な位置を検出するために特定の円形状に配置された複数のセンサーを含むウィジェットです。

raw カウント

センサーの物理的静電容量を示す CapSense ハードウェア ブロックの未処理デジタル カウント出力です。

リフレッシュ間隔

センサーの 2 回の連続スキャンの間の時間です。

スキャン分解能

CSD ブロックによって生成される raw カウントの分解能 (ビット数) です。

スキャン時間

センサーのスキャンを完了するのに要する時間です。

自己容量

回路のグランドと電極間の静電容量です。

感度

センサー静電容量の変化に応じる raw カウントの変化であり、カウント/pF の単位で表します。センサーの感度は基板レイアウト、オーバーレイ特性、センシング方式およびチューニング パラメーターに依存します。

センス クロック

CSD センシング方式用のスイッチト キャパシタ回路のフロント エンドを実装するために使用されるクロック ソースです。

センサー

「静電容量センサー」を参照してください。

センサー自動リセット

システム故障の際、または金属物体がセンサーの近くに連続的に存在する際に、センサーが誤ったタッチ状態を無期限に報告してしまうことを防ぐための設定です。

センサー自動リセット機能が有効になった場合、ベースラインは差分カウントがノイズ閾値を超えても常に更新されます。このように、センサーが無期限のオン状態を報告しないようにします。センサー自動リセットが無効なとき、ベースラインは差分カウントがノイズ閾値を下回った場合にのみ更新されます。

センサー連動

「連動センサー」を参照してください。

シールド電極

センサーの周囲を覆う銅トレースで水または他の液体による誤タッチを防止します。シールド電極は CSD ブロックからのシールド信号出力によって駆動されます。「被駆動シールド」を参照してください。

シールド タンク コンデンサ (C_{SH})

高い寄生容量を持つ広いシールド層がある場合、CSD シールドの駆動能力を強化するために使用されるオプションの外部コンデンサ (C_{SH} タンク コンデンサ) です。

信号 (CapSense 信号)

差分カウントは信号とも呼ばれます。「差分カウント」を参照してください。

信号対ノイズ比 (SNR)

タッチしたときのセンサーの信号とタッチしないときのセンサーのノイズ信号との比率です。

スライダー分解能

スライダーが分解された指の位置の総数を示すパラメーターです。

タッチパッド

特定の水平と垂直な様式で配置された複数のセンサーから成り、タッチの X および Y 位置を検出するウィジェットです。

トラックパッド

「タッチパッド」を参照してください。

チューニング

CapSense の動作に必要なさまざまなハードウェアおよびソフトウェアまたは閾値パラメーターの最適値を決定するプロセスです。

VREF

PSoC 内にあるプログラマブル電圧リファレンス ブロックであり、CapSense および ADC の動作に使用されます。

ウィジェット

単一センサーまたは同様のセンサー グループで構成される CapSense コンポーネントのユーザー インターフェース要素です。ボタン、近接センサー、リニア スライダー、ラジアル スライダー、マトリックス ボタンおよびタッチパッドはサポートされたウィジェットです。

11. 改版履歴

11.1 改訂履歴

文書名: AN78329 – CY8C20xx7/S CapSense®デザイン ガイド 文書番号: 001-88331		
版	**発行日**	**変更内容**
**	07/04/2013	これは英語版 001-78329 Rev *C を翻訳した日本語版 001-88331 Rev. ** です。
*A	06/29/2015	これは英語版 001-78329 Rev *E を翻訳した日本語版 001-88331 Rev. *A です。
*B	01/08/2020	これは英語版 001-78329 Rev *H を翻訳した日本語版 001-88331 Rev. *B です。