AN65973 - CY8C20xx6A/H/AS

CapSense®設計ガイド

文書番号: 002-12923 Rev. *A

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

© Cypress Semiconductor Corporation, 2010-2019. 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, CapsSense, 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 6

1.1 要約 6

1.2 サイプレス CapSense の文書体系 6

1.3 CY8C20xx6A/H/AS CapSense ファミリの特長 8

1.3.1 高性能タッチ センス機能 8

1.3.2 デバイスの特長 9

1.4 本書の表記法 10

2. CapSense 技術 11

2.1 CapSense の原理 11

2.2 CY8C20xx6A/AS/H の静電容量センス方式 12

2.2.1 CapSense シグマデルタ (CSD) 13

2.2.2 CapSense 逐次近似電磁環境適合性 (CSA_EMC) 14

2.3 SmartSense 自動チューニング 15

3. CapSense 設計ツール 17

3.1 概要 17

3.1.1 PSoC Designer およびユーザー モジュール 17

3.1.2 ユニバーサル CapSense コントローラー キット 18

3.1.3 ユニバーサル CapSense コントローラー モジュール ボード 18

3.1.4 CapSense データ表示ツール 19

3.2 ユーザー モジュール概要 19

3.3 CapSense ユーザー モジュールのグローバル アレイ 20

3.3.1 Raw カウント 20

3.3.2 ベースライン 20

3.3.3 差分カウント (信号) 20

3.3.4 センサー状態 21

3.4 CSD ユーザー モジュール パラメーター 21

3.4.1 ユーザー モジュールの高レベル パラメーター 22

3.4.2 CSD ユーザー モジュール低レベル パラメーター 24

3.4.3 CSA_EMC ユーザー モジュール 低レベル パラメーター 25

3.4.4 SmartSense ユーザー モジュール パラメーター 27

3.4.5 SmartSense_EMC ユーザー モジュール パラメーター 28

4. ユーザー モジュールによる CapSense 性能のチューニング 30
4.1 一般的な注意事項 30
4.1.1 信号、ノイズ、および SNR 30
4.1.2 充電／放電速度 31
4.1.3 ベースライン更新閾値の検証の重要性 32
4.2 CSA_EMC ユーザー モジュールのチューニング 32
4.3 CSA_EMC の推奨 C_{INT} 値 33
4.4 センサー C_P の測定 33
4.5 CSA_EMC クロックの予測 34
4.6 整定時間の設定 34
4.7 CapSense データの監視 35
4.8 SNR の向上方法 35
4.8.1 ノイズの削減 35
4.8.2 信号の増大 35
4.9 CSD ユーザー モジュールのチューニング 35
4.9.1 CSD 用の推奨 C_{MOD} 値 36
4.9.2 ShieldElectrodeOut 37
4.9.3 I_{DAC} 範囲 37
4.9.4 自動校正 37
4.9.5 I_{DAC} 値 37
4.9.6 プリチャージ源 37
4.9.7 プリスケーラ 37
4.9.8 分解能 38
4.9.9 スキャン速度 39
4.9.10 高レベル API パラメーター 39
4.9.11 高レベル パラメーターの設定 40
4.10 SmartSense ユーザー モジュールを使用 40
4.10.1 SmartSense のガイドライン 40
4.10.2 相違点 41
4.10.3 SmartSense 用の推奨される C_{CMOD} 値 41
4.10.4 SmartSense ユーザー モジュール パラメーター 41
4.10.5 SmartSense_EMC ユーザー モジュール用のガイドライン 42
4.10.6 CapSense センサーのスキャン時間 43
4.10.7 SmartSense 応答時間 44
4.10.8 SmartSense_EMC ユーザーモジュールを使って最低限の S/N 比を確保する方法 45
4.10.9 ファームウェア デザイン ガイドライン 46
5. 設計上の注意事項 48
5.1 オーバーレイの選択 48
5.2 ESD 保護 49
5.2.1 防止 49
5.2.2 リダイレクト 49
5.2.3 クランプ 49
5.3 電磁環境適合性 (EMC) の注意事項 49
5.3.1 放射性干渉 49

5.3.2 放射妨害波 50
5.3.3 伝導イミュニティおよびエミッション 50
5.4 ソフトウェアのフィルタリング 50
5.5 消費電力 51
5.5.1 システム設計の推奨事項 51
5.5.2 スリープ スキャン方式 51
5.5.3 応答時間対消費電力 51
5.5.4 平均消費電力の測定 52
5.6 ピン割り当て 52
5.7 GPIO の負荷瞬時変化 53
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 54
5.7.2 GPIO 負荷瞬時変化ノイズの補正用のファームウェア ガイドライン 55
5.8 PCB レイアウト ガイドライン 56
6. 低消費電力設計上の注意事項 57
6.1 その他の省電力技術 57
6.1.1 駆動モードをアナログ HI-Z に設定 57
6.1.2 まとめ 58
6.1.3 スリープ モードの混乱 58
6.1.4 保留中の割り込み 58
6.1.5 グローバル割り込みのイネーブル 58
6.2 ウェイクアップ後の実行シーケンス 58
6.2.1 PLL モードのイネーブル 59
6.2.2 グローバル割り込みイネーブル化の実行 59
6.2.3 スリープモードを持った I^2C スレーブ 59
6.2.4 スリープ タイマー 59
7. リソース 60
7.1 ウェブサイト 60
7.2 データシート 60
7.3 テクニカル リファレンス マニュアル 60
7.4 開発キット 60
7.4.1 ユニバーサル CapSense コントローラー キット 60
7.4.2 ユニバーサル CapSense モジュール基板 60
7.4.3 インサーキットエミュレーション (ICE) キット 61
7.5 サンプル ボード ファイル 61
7.6 PSoC Programmer 63
7.7 CapSense データ表示ツール 63
7.8 PSoC Designer 63
7.9 コード例 64
7.10 デザイン サポート 64
用語集 65
改版履歴 70
ドキュメント改版履歴 70

1. はじめに

1.1 要約

本文書は、静電容量センス (CapSense®) 機能を CapSense コントローラーの CY8C20xx6A/AS/H ファミリに使用するための設計ガイダンスを示します。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 CY8C20xx6A/AS/H ファミリの機能
- CapSense の作動原理
- CapSense 設計ツールの概要
- 最適なパフォーマンスのための CapSense システムのチューニング詳細ガイド
- 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 文書体系をまとめます。これらのリソースによって、CapSense 製品の設計を完了するために必要な情報へ迅速にアクセスできます。図 1-1 に静電容量センスを使用し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 で付番された各タスクをサポートする文書へのリンクが記載されています。

図 1-1. 標準的な CapSense の製品設計フロー

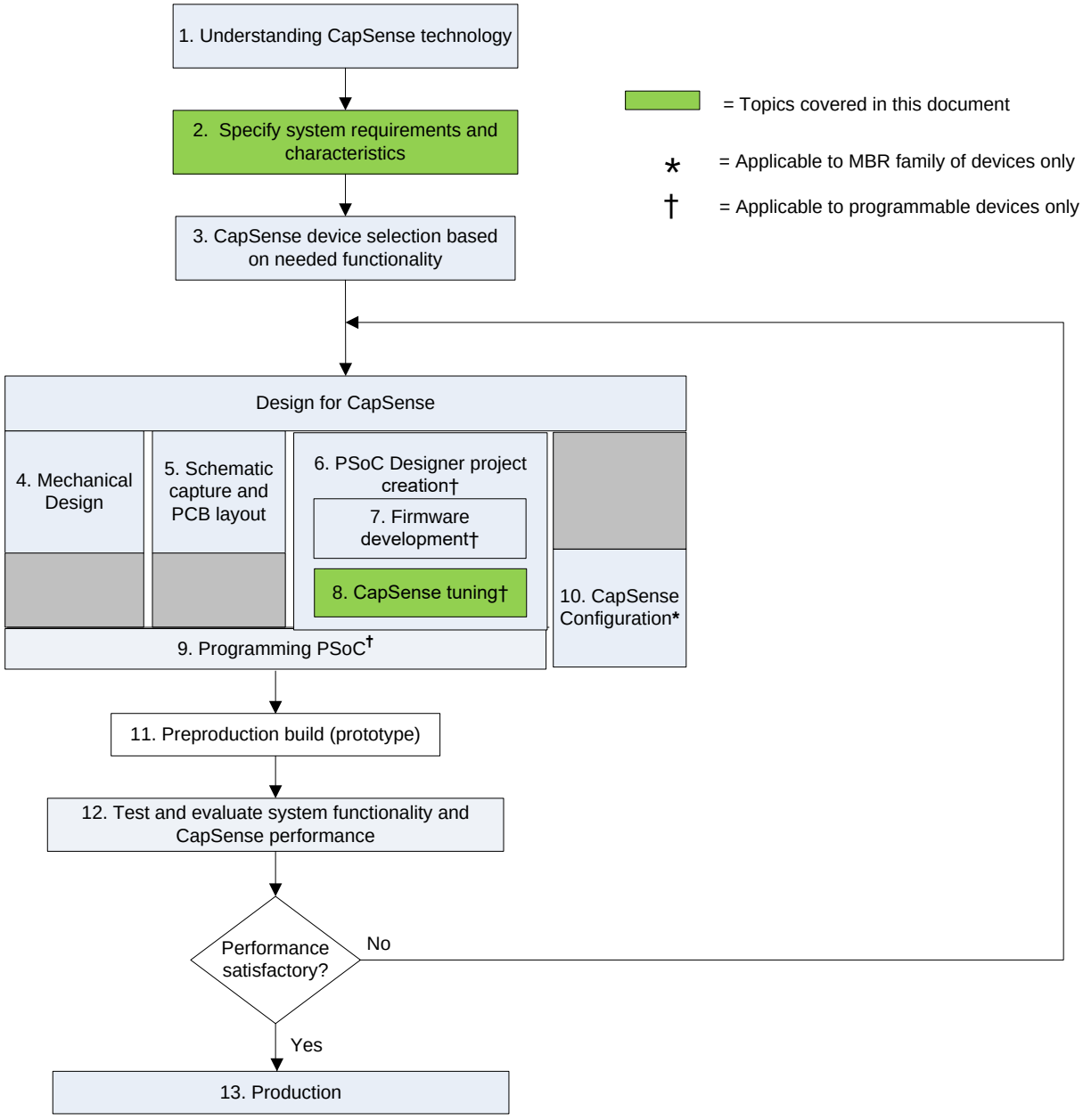


表 1-1. 図 1-1 中に番号付けされた設計タスクをサポートしているサイプレスの文書

図 1-1 中に番号付けされた設計タスク	サイプレス CapSense の関連文書
1	■ Getting Started with CapSense
2	■ Getting Started with CapSense ■ CY8C20xx6A/AS/H CapSense Device Datasheets
3	■ Getting Started with CapSense ■ PSoC ファミリ向け CapSense 設計ガイド (本文書)
4	■ Getting Started with CapSense
5	■ Getting Started with CapSense ■ PSoC Designer™ User Guides
6	■ PSoC Designer User Guides
7	■ Assembly Language User Guide ■ C Language Compiler User Guide ■ CapSense Code Examples ■ PSoC ファミリ向けテクニカル リファレンス マニュアル (CY8C20xx6A/AS/H 用)
8	■ PSoC ファミリ向け CapSense 設計ガイド (本文書) ■ PSoC ファミリ向け CapSense ユーザー モジュール データシート (CSD および SmartSense™) ■ PSoC ファミリ向けテクニカル リファレンス マニュアル (CY8C20xx6A/AS/H 用) ■ CapSense Controller Code Examples Design Guide ■ AN2397 -CapSense Data Viewing Tools
9	■ Programmer User Guide ■ MiniProg3 User Guide ■ AN2026c - In-System Serial Programming (ISSP) Protocol for CY8C20xx6, CY8C20xx6A, CY8CTMG2xx, and CY8CTST2xx, CY7C643xx, and CY7C604xx ■ AN44168 - PSoC 1 Device Programming using External Microcontroller (HSSP) ■ AN59389 - Host-Sourced Serial Programming for CY8C20xx6, CY8CTMG2xx, and CY8CTST2xx
11	■ PSoC ファミリ向け CapSense 設計ガイド (本文書) ■ CapSense Code Examples

1.3 CY8C20xx6A/H/AS CapSense ファミリの特長

サイプレスの CY8C20xx6A/H/AS は低電力、高性能、プログラム可能な CapSense コントローラー ファミリで、以下の機能があります。

1.3.1 高性能タッチ センス機能

- プログラム可能な静電容量センス要素
  - ☐ 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 36 個の静電容量センサー、36 個の GPIO またはスライダーに対応
  - ☐ 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製品化までの時間を短縮
  - ☐ 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
  - ☐ デザインの移植性 - 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
  - ☐ 実行中の環境補償

- 静電容量が最小 0.1pF までのタッチを検知

- 向上したノイズ耐性と堅牢性
  - SmartSense は自動的に環境とノイズの変化を補償
  - SmartSense_EMC 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、最高 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
  - 最適消費電力に向けた 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)
  - 125ms スキャン レートでセンサーごとに 28µA

1.3.2 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、外部水晶発振子またはクロック信号による最大 4MIPS
- 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 EEPROM
- 高精度、プログラム可能なクロック供給
  - 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
- 強化された汎用入出力 (GPIO) の特長
  - プログラム可能なピン コンフィギュレーションを備えた最大 36 個の GPIO
  - 25mA シンク電流／GPIO および 120mA 合計シンク電流／デバイス
  - すべての GPIO で内部抵抗プルアップ、high-z、オープン ドレイン、およびストロング ドライブ モードに対応
- ペリフェラルの特長
  - 3 個の 16 ビット タイマー
  - フルスピード USB - 12Mbps USB 2.0 対応
  - I^2C - マスター (100kHz) およびスレーブ (最大 400kHz)
  - SPI - マスターおよびスレーブ - 46.9kHz～12MHz の設定可能範囲
  - 最大 10 ビット ADC - 0～1.2V の入力範囲
- 作動条件
  - 広い作動電圧範囲: 1.71V～5.5V
  - 温度範囲: –40°C～+85°C

1.4 本書の表記法

表記法	使用法
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 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 *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]	手順としてキーボードのコマンドを表示 [**Enter**]または[**Ctrl**] [**C**]
File > Open	メニュー パスを表示 File > Open > New Project
太字	操作手順でコマンド、メニュー パス、アイコン名を表示 **File** アイコンをクリックして、**Open** をクリック
Times New Roman フォント	数式を表示 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

2. CapSense 技術

2.1 CapSense の原理

CapSense はタッチ センシング技術で、センサーとしてデザインされている 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動作します。図 2-1 に示すように、各センサー ピンの総静電容量は n 個のセンサーがあるデザインにおいて、C_X、1～C_X、n の値を持つ等価集中コンデンサーとしてモデル化することができます。CY8C20xx6A/AS/H デバイスの内部回路は各 C_X の大きさをデジタル コードに変換し、ポスト プロセシング用に保存します。その他のコンポーネントで、C_{MOD} は CapSense コントローラーの内部回路が使用します。この詳細は CY8C20xx6A/AS/H の静電容量センス方式で説明しています。

図 2-1. CY8C20xx6A/AS/H PSoC デバイスでの CapSense の実装

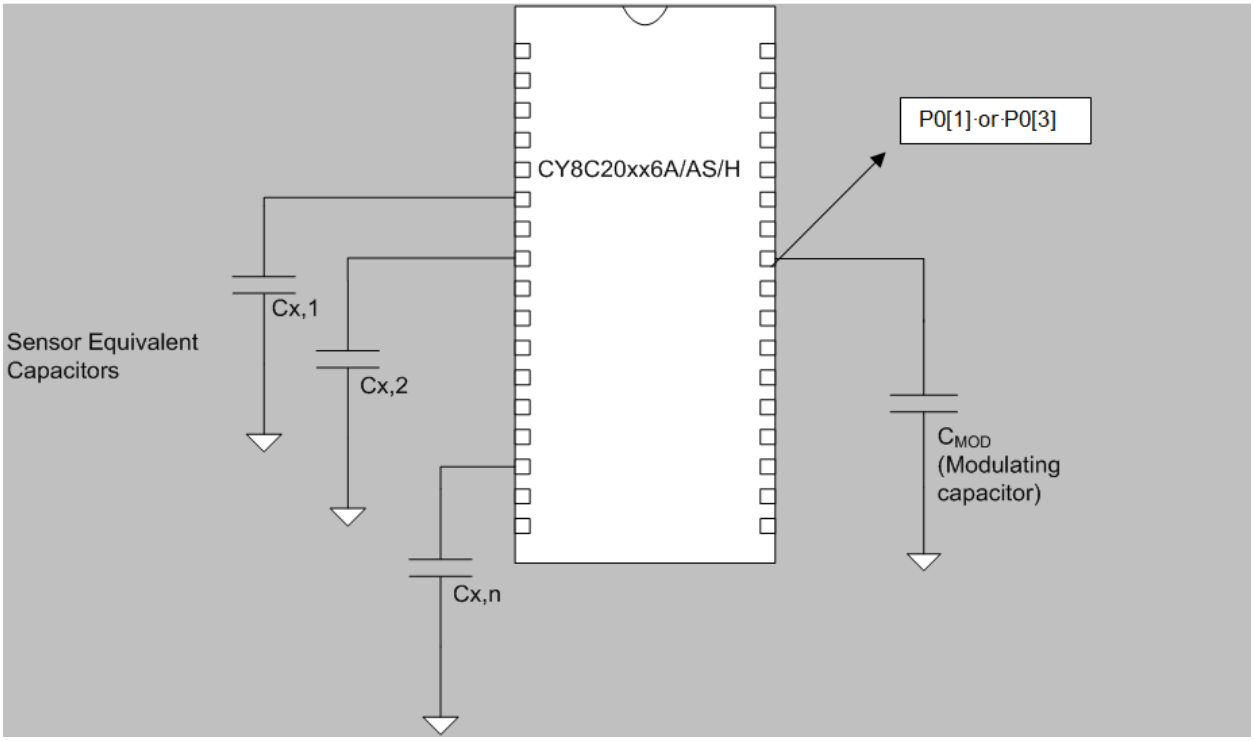


図 2-1 に示すように、センサーの各 I/O ピンは必要に応じて配線、ビアまたはその両方でセンサー パッドに接続されます。オーバーレイはセンサー パッドを覆う非導電性のカバーであり、製品のタッチ インターフェースを構成します。指がオーバーレイに接触すると、人体の導電性と大きな体積によりセンサー パッドに対し平行な接地された導体面となります。これを図 2-1 に示します。この配置は平行板コンデンサーを構成し、その静電容量は式 1 で表されます:

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

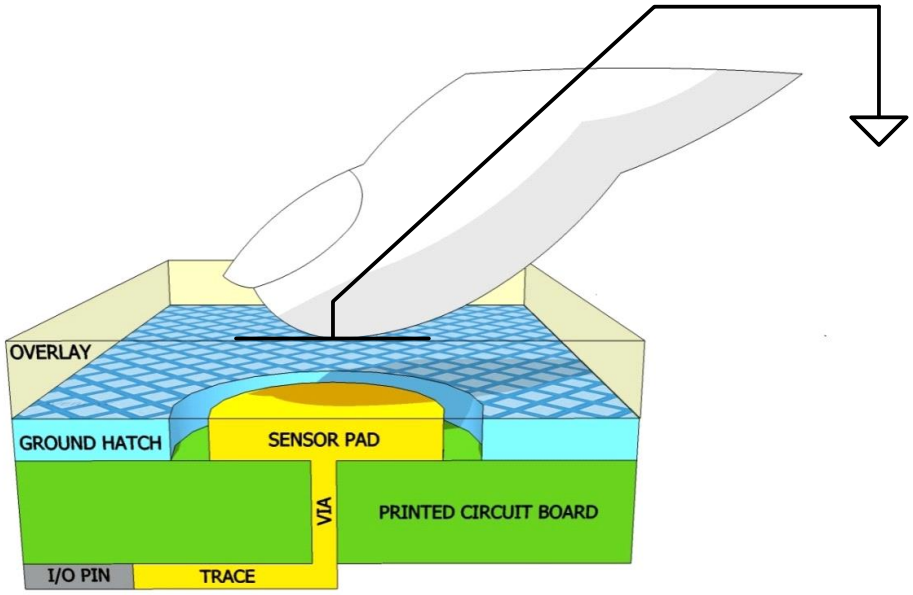
ε_0 = 真空の誘電率

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時の典型的な CapSense PCB の断面図



平行板コンデンサーに加えて、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ー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、CapSense コントローラー内部の寄生容量およびセンサー パッド、配線、ビア、およびグランド プレーン、他の配線、製品のシャーシまたは同封物に含まれる金属の間で結合した電場などとの、組み合わせの結果として生じます。CapSense コントローラーは、センサー ピンに接続してい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:

$$C_X = C_P \quad \text{式 2}$$

指がセンサー パッド上にある場合は、 C_X は C_P と C_F の和に等しい:

$$C_X = C_P + C_F \quad \text{式 3}$$

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はきわめて重要であり、これについては「ユーザー モジュールによる CapSense 性能チューニング」で説明されています。

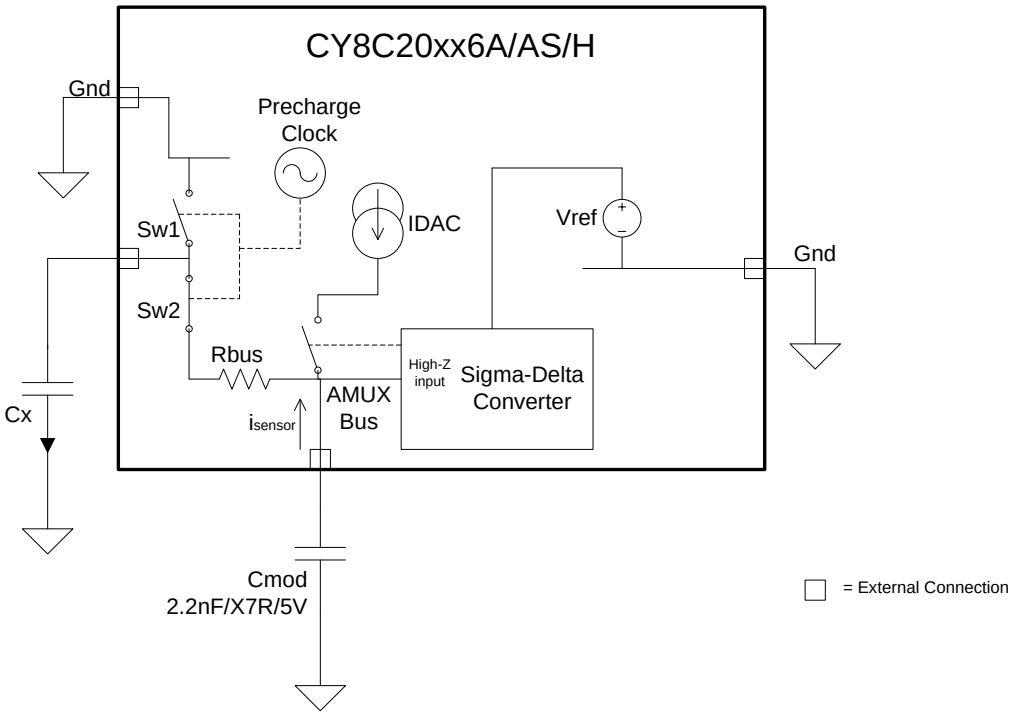
2.2 CY8C20xx6A/AS/H の静電容量センス方式

CY8C20xx6A/AS/H デバイスは、センサー容量 (C_X) をデジタル カウントに変換する複数の CapSense 方法をサポートしています。これらの方法は、CapSense シグマ デルタ (CSD)、CapSense 逐次近似電磁互換性 (CSA_EMC)、SmartSense、および SmartSense_EMC です。これらの方法は、PSoC Designer ユーザー モジュールの形式で実行され、以下のセクションで説明されています。

2.2.1 CapSense シグマデルタ (CSD)

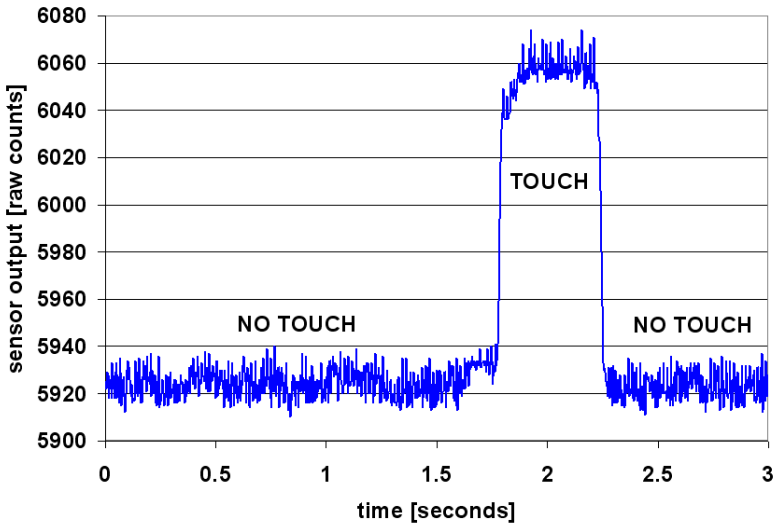
図 2-3 で示すように、CY8C20xx6A/AS/H デバイスの CapSense シグマデルタ方法は C_X をスイッチト キャパシタ回路に統合します。センサー (C_X) は、アンダーラップしたスイッチ Sw1 および Sw2 によって GND およびアナログ MUX (AMUX) バスにそれぞれ交互に接続されます。Sw1 および Sw2 はプリチャージ クロックによって駆動され、AMUX バスから電流 (I_{SENSOR}) を流します。I_{SENSOR} は C_X の大きさに正比例します。シグマデルタ コンバーターは AMUX バス電圧をサンプリングして、変調ビット ストリームを生成し、これにより AMUX をチャージする定電流源 (I_{DAC}) が制御され、AMUX バスの平均電圧を V_{REF} に維持します。センサーは、変調コンデンサー (C_{MOD}) から I_{SENSOR} を充電します。C_{MOD} は R バスと共にローパス フィルターを形成し、シグマデルタ コンバーター入力においてプリチャージ スイッチングの過渡現象を減衰させます。

図 2-3. CSD ブロック図



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 コンバーターはビット 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 コンバーターはセンサー 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。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、センサーの状態を判断します。図 2-4 は、センサーを指で触れて離した間の連続したスキャン数により CSD 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、指の接触によって、C_X が C_F に増加し、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。

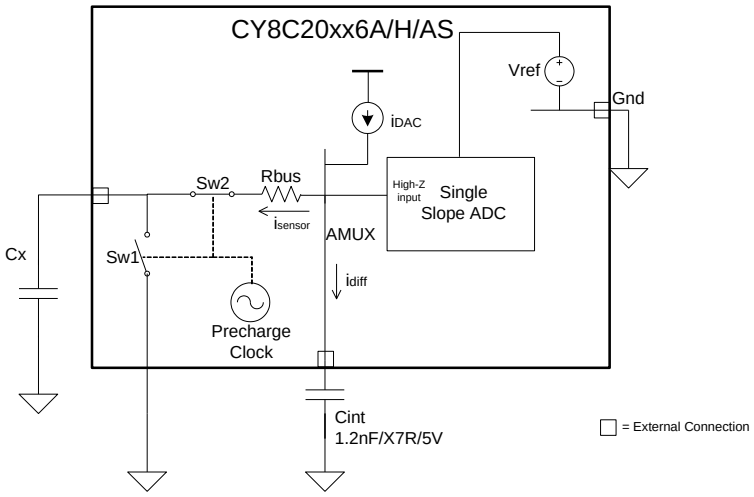
図 2-4. 指を触れた時の CSD の raw カウント



2.2.2 CapSense 逐次近似電磁環境適合性 (CSA_EMC)

図 2-5 に示すように、CY8C20xx6A デバイスで使用する CapSense 逐次近似方式電磁環境適合性 (CSA_EMC) 方式は CX をスイッチト キャパシタ回路に統合します。

図 2-5. CSA_EMC ブロック図



定電流源 (I_{DAC}) は電流 I_{DAC} 分を AMUX に供給します。スイッチ Sw1 と Sw2 によって、順に AMUX バスと GND の間で交互に接続されるセンサー (C_X)は、AMUX バスから電流 I_{SENSOR} 分を低減させます。I_{SENSOR} は C_X の大きさに正比例します。スイッチ Sw1 と Sw2 は、プリチャージ クロックとして知られる非重複クロックによりクロック制御されます。

積分コンデンサーC_{INT} は、差分電流 *iDiff* (I_{DAC} と I_{SENSOR} の差分) を積分し、その電位を上げます。この電荷の集積は、C_{INT} で上がった電位が I_{SENSOR} が I_{DAC} に等しくなる均等レベルに達するまで続きます。この積分時間を整定時間と言います。

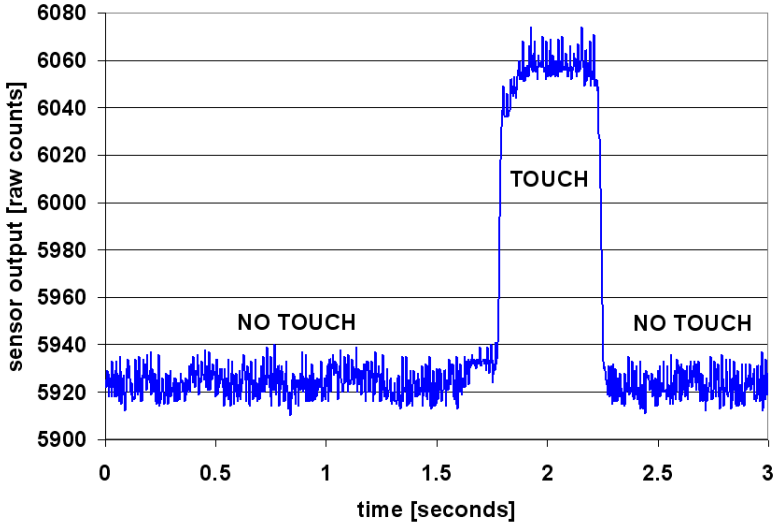
シングルスロープ ADC を使用して、C_{INT} の平衡電位を、raw カウントという C_X に比例するデジタル出力カウントに変換します。この raw カウントは高レベル アルゴリズムによって解釈され、それによりセンサーの状態が判断されます。

I_{DAC} 電流は、C_{INT} の平衡電圧が ADC の線形変換領域内になるようにする逐次比較方式を使用して設定されます。

図 2-6 は、センサーに指でタッチしてから離すまでの多くの連続スキャンの CSA_EMC 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、指の接触によって、C_X が C_F に増加し、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 レベルの動きと事前に設定された閾値を比較することで、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 2-6. 指でタッチ中の CSA_EMC raw カウント



CSA_EMC CapSense のアルゴリズムは RF 干渉が存在する場合でも正常に機能するよう拡張されました。CSA_EMC は、CapSense が導電干渉、AC ノイズ、およびインバーター、変圧器、電源などその他のノイズ源に曝されるアプリケーションで使用します。CSA_EMC ユーザー モジュール 低レベル パラメーターはこのトピックについて詳細に検討しています。

2.3 SmartSense 自動チューニング

タッチ センス式のユーザー 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ています。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 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易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 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 インバーター、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。

2.3.1.1 プロセスのばらつき

CY8C20xx6A/H/AS 用の SmartSense ユーザー モジュール (UM) は、5pF～45pF の範囲でセンサー寄生容量に対応するよう設計されています (一般的なセンサー C_P 値は 10pF～20pF の範囲です)。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 5pF～45pF の範囲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 自動チューニングがこのような問題を解決します。

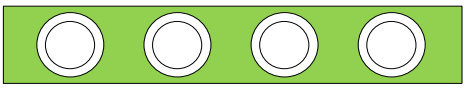
SmartSense 自動チューニングはプラットフォーム設計を実現します。ラップトップ コンピューターの静電容量タッチセンスのマルチメディア キーを想像してください。ボタン間の間隔はラップトップのサイズとキーボードのレイアウトで

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、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、センサーはモデルごとに調整する必要があります。SmartSense では、チューニングが効果的に自動で行われるため、「Getting Started with CapSense」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 2-7. 21 インチ モデル用のラップトップ マルチメディア キーのデザイン



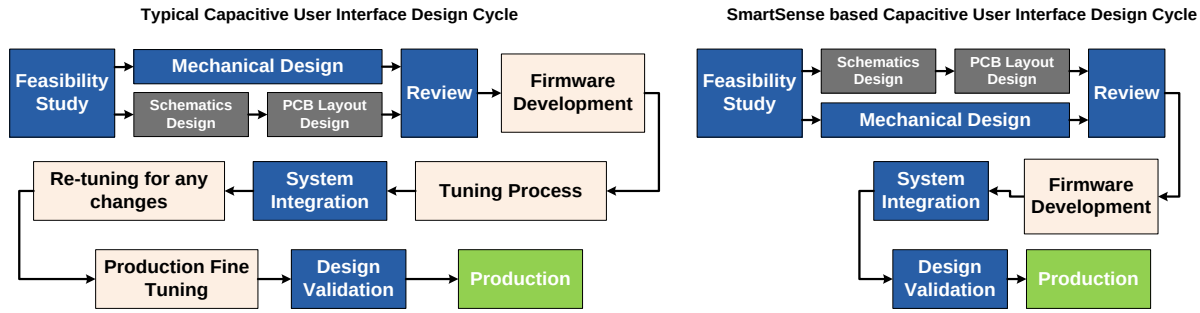
図 2-8. 機能とボタン サイズが同一の 15 インチ モデル用のラップトップ マルチメディア キーの設計



2.3.1.2 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 センス 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 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不要のため、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 2-9 は一般的なタッチ センス コントローラーと SmartSense ベースのデザインの設計サイクルを比較します。

図 2-9. 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



3. CapSense 設計ツール

3.1 概要

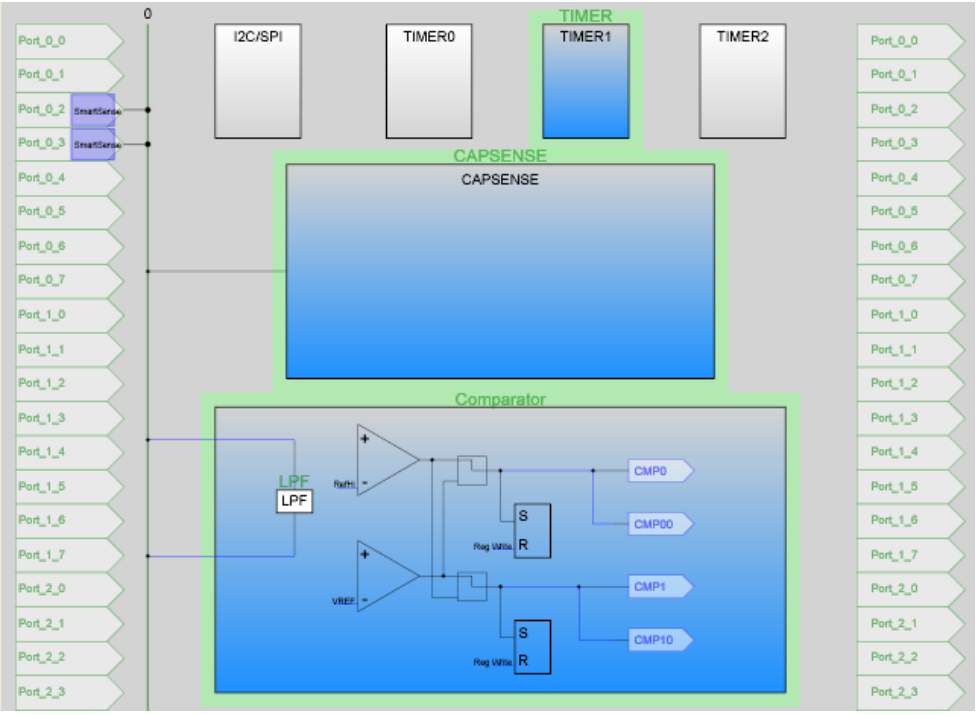
Cypress は CapSense 静電容量タッチ センス アプリケーション開発用のハードウェアとソフトウェア ツールのフルラインを提供しています。CY8C20xx6A/H/AS ファミリの基本的な開発システムには以下のコンポーネントが含まれています。注文情報はリソースを参照してください。

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 & 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 エディター 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ています。複雑な設計用には専用コンパイラがあります。

CSA_EMC 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ています。

図 3-1. PSoC Designer デバイス エディター



3.1.1.1 CapSense ユーザー モジュールの導入

PSoC Designer で新しい CY8C20xx6A/H/AS プロジェクトを作成する場合は、以下の手順を行います。

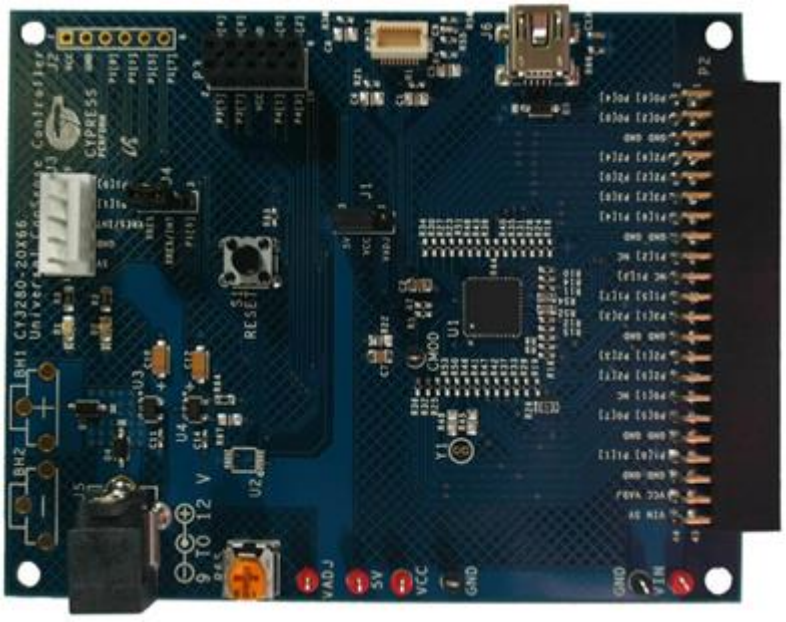
1. CY8C20xx6 をターゲット デバイスとした新しい PSoC Designer プロジェクトを作成します。
2. CSD/CSA_EMC/SmartSense ユーザー モジュールを選択し、配置します。
3. ユーザー モジュールを右クリックしてユーザー モジュール ウィザードにアクセスします。
4. ボタン センサーの数、スライダーのコンフィギュレーション、ピンの割り当ておよび接続を設定します。
5. ピンおよびユーザー モジュールのグローバル パラメーターを設定します。
6. アプリケーションを生成して、アプリケーション エディターを開きます。
7. ユーザー モジュール データ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 ウィザードの設定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 コードは、サンプル コードを参照してください。

3.1.2 ユニバーサル CapSense コントローラー キット

ユニバーサル CY3280-20xx6 CapSense コントローラー キットには、プロトタイプ作成とデバッグ処理を簡単にするための制御回路とプラグイン ハードウェアが事前定義されています。プログラミング用に MiniProg ハードウェアがキットと一緒に含まれており、I²C-USB ブリッジ ハードウェアがチューニングとデータ収集用に含まれています。

図 3-2. CY3280-20xx6 CapSense コントローラー キット



3.1.3 ユニバーサル CapSense コントローラー モジュール ボード

サイプレスのモジュール基板には、用途に応じたさまざまなセンサー、LED、インターフェースが用意されています。

- CY3280-BSM シンプル ボタン モジュール
- CY3280-BMM マトリクス ボタン モジュール
- CY3280-SLM リニア スライダー モジュール
- CY3280-SRM ラジアル スライダー モジュール

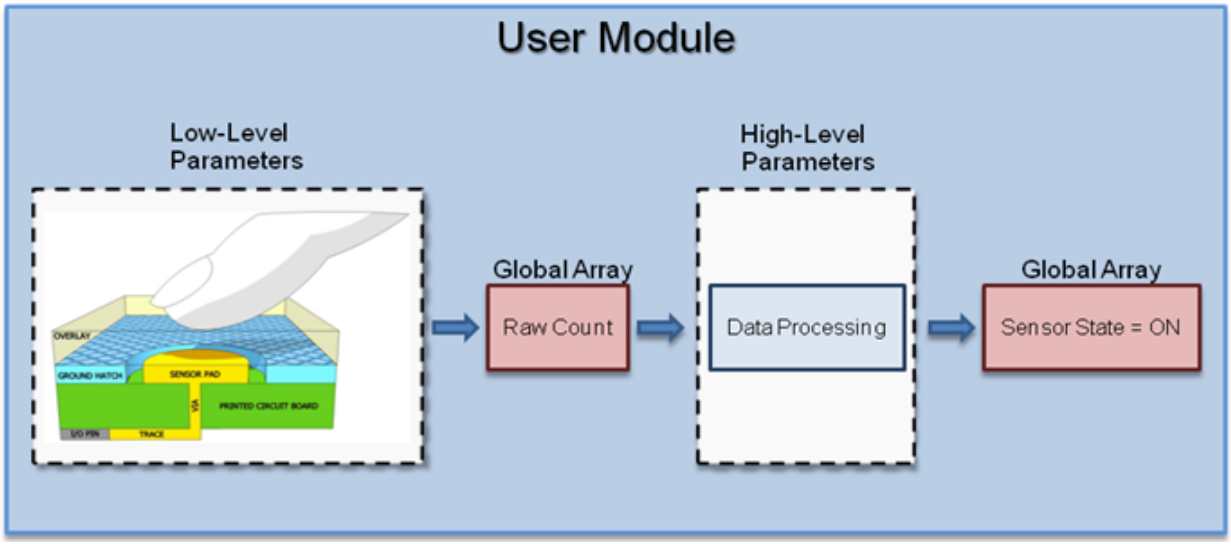
- CY3280-BBM ユニバーサル CapSense プロトタイピング モジュール

3.1.4 CapSense データ表示ツール

CapSense の設計プロセス中には、チューニングおよびデバッグ作業用に 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よくあります。これは、マルチチャートとブリッジ コントロール パネルという 2 つの CapSense データ表示ツールを使用して実現できます。アプリケーション ノート AN2397 – CapSense Data Viewing Tools ではこれらのツールを詳しく説明します。

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ス 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。

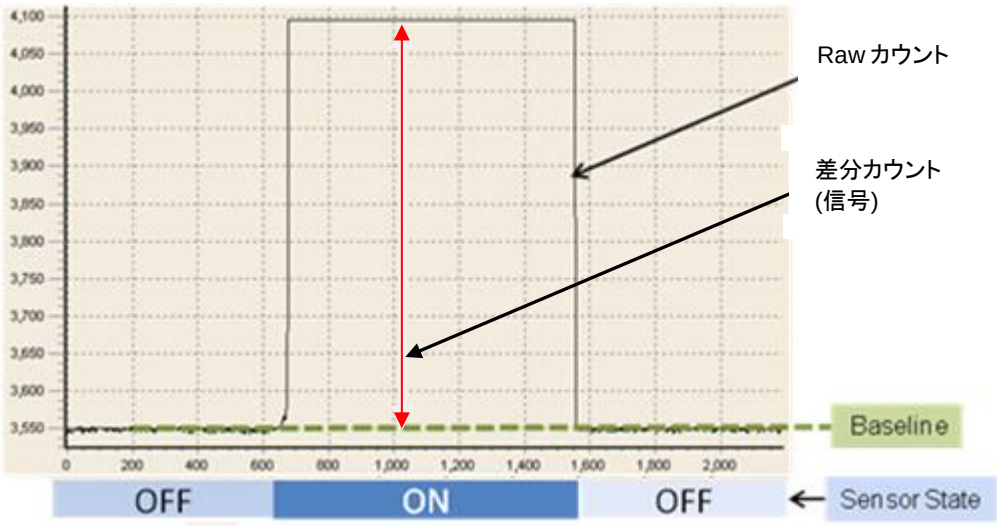
スキャン用センサーのスピードや分解能などの低レベル パラメーターは、物理レイヤでセンス方法の動作を定義し、静電容量から raw カウントへの変換に関連付けます。低レベル パラメーターは各タイプのセンス方法に固有であり、CSD ユーザー モジュール低レベル パラメーター、CSA_EMC ユーザー モジュール 低レベル パラメーターおよび SmartSense ユーザー モジュール パラメーターで説明されています。

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ス方式で同じであり、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ています。

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 コントローラーのハードウェア回路はセンサー静電容量を測定します。測定値は、ユーザー モジュールの API 関数 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、SmartSense または CSA_EMC 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ています。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 のヘッダー ファイルで決定されます。

3.3.2 ベースライン

温度や湿度などの段階的な環境変化は、センサーの raw カウントに影響を及ぼし、その結果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。つまり、ベースライン値変数は、入力 raw カウント値を入力するデジタルロー 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、SmartSense、または CSA_EMC 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 のヘッダー ファイルで決定されます。

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時に差分カウントはゼロです。センサーのアクティブ化 (触ることにより) によって、差カウントが正の値となります。

センサーの差分カウント値は *UMname*_waSnsDiff[]整数アレイに保存されます。*UMname* は CSD、SmartSense、または CSA_EMC になります。このアレイは *UMname.h* のヘッダー ファイルで決定されます。

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[]array (*UMname* は CSD、SmartSense、または CSA_EMC) というバイト アレイに保存されます。このアレイは *UMname.h* のヘッダー ファイルで決定されます。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 ユーザー モジュール パラメーター

CSD ユーザー モジュール 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。CSA_EMC ユーザー モジュール パラメーターのリストと、その分類の方法は図 3-5 を参照してください。CSD ユーザー モジュール パラメーターのリストと、その分類の方法は図 3-6 を参照してください。

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

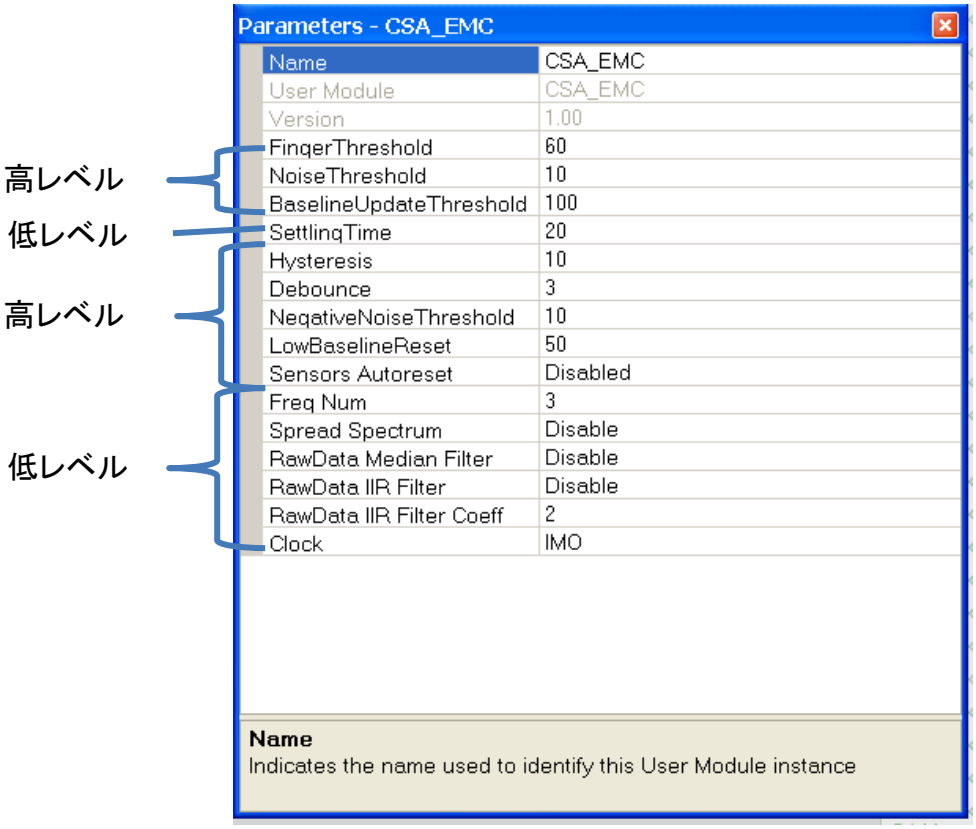
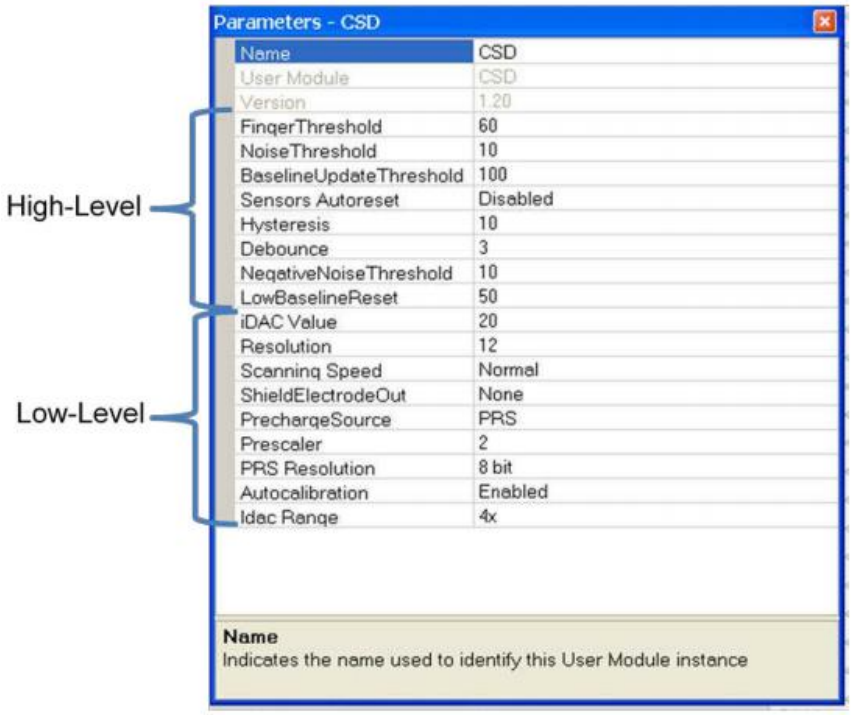


図 3-6. PSoC Designer – CSD パラメーター ウィンドウ



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

ユーザー モジュールは指閾値パラメーターを使用し、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」に設定されている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

ヒステリシス設定は、システム ノイズによるセンサーのランダム トグルの状態を防ぎます。ヒステリシス パラメーターは指しきい値とともに使用され、センサーの状態を決定します。差分カウントが特定のサンプル数の間 [指閾値 + デバウンス用ヒステリシス レベル] を上回ると、センサー状態はオフからオンになります。差分カウントが [指閾値 – ヒステリシス レベル] を下回ると、センサー状態はオンからオフになります。ヒステリシスの関数は式 4 に示します。

$$if Difference Count \geq Finger Threshold + Hysteresis, Sensor State = ON$$

$$if Difference Count \leq Finger Threshold - Hysteresis, \ Sensor State = OFF \quad \text{式 4}$$

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、センサー状態のオフからオンへの転換を防止させます。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」にセット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。すなわち、ベースライン変数は、入力 raw カウント値を受け取るデジタル 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を調整するために使用されます。

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作動するために、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。これはベースライン値のリセットに必要となる異常に低いサンプルの数をカウントします。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 センサーの自動リセット

センサー自動リセット機能を有効にすると、センサーが無期限にオンにならないようにすることができます。このパラメーターでは、ベースラインを常に更新されるか、または差分カウントがノイズ閾値パラメーター以下になった時にのみ更新するかを決めます。

センサーの自動リセットが有効化されている場合、差分カウントがノイズ閾値パラメーターを超えてもベースライン値が更新されます。これにより、センサーが連続して触れられる時にオン状態にある最大時間 (通常は 5～10 秒) が制限されますが、センサーが触れられずに raw カウントが増加してしまった時にセンサーが永久にオンにならないようにすることができます。この突然の増加は、システム内の電気的故障またはセンサーの近くに置かれる金属物体により生じることがあります。

センサーの自動リセットが無効化されていると、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。従って、センサーが触れられる限り、オン状態にあります。

値は、Enabled (有効) と Disabled (無効) です。推奨設定は高レベル パラメーターの設定を参照してください。

3.4.2 CSD ユーザー モジュール低レベル パラメーター

CSD 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、いくつかの低レベル パラメーターがあります。これらのパラメーターは CSD センス方式によって異なり、どのように raw カウント データをセンサーから取得するのかが決まります。

3.4.2.1 I_{DAC} 値

静電容量測定範囲は I_{DAC} パラメーターによって設定されます。この値が高いほど、範囲が広くなります。raw カウントが全範囲の約 50 %～70 %になるように I_{DAC} 値を調節します。このパラメーターはユーザー モジュール API CSD_SetIdacValue() を使用して、実行中に変更することができます。

値の範囲は 1～255 です。

3.4.2.2 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、スキャン時間が短縮されます。

値の範囲は 9～16 ビットです。

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

3.4.2.3 スキャン速度

このパラメーターはセンサーの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。

3.4.2.4 シールド電極出力

シールド電極は寄生容量を削減するために使用します。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。

可能な値は P0[7]または P1[2]です。

3.4.2.5 プリチャージ源

このパラメーターは、プリチャージ スイッチのクロック ソースを選択します。

可能な値は PRS とプリスケーラです。殆どの場合は、PRS 源を使用すると、より良い EMI イミュニティと低放射を実現できます。

3.4.2.6 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。

可能な値は、1、2、4、8、16、32、64、128、および 256 です。

3.4.2.7 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。

可能な値は 8 ビットと 12 ビットです。対応するシーケンスの長さは 511 および 2047 の入力クロック周期です。12 ビット設定が良好な SNR を提供しない場合には、8 ビット設定を使用します。

3.4.2.8 自動校正

自動校正機能が有効になった場合、raw カウント値が最大カウント (2^N–1、N は分解能) の割合として正規化されます。自動校正はデバイス エディターの設定を無効にします。

自動校正が無効になった場合、raw カウント値はデバイス エディターで設定した 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。

3.4.2.9 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。例えば、2x を選択すると I_{DAC} 出力が範囲の 2 倍に設定されます。

可能な値は 1x、2x、4x、および 8x です。

3.4.3 CSA_EMC ユーザー モジュール 低レベル パラメーター

CSA_EMC 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、いくつかの低レベル パラメータがあります。これらのパラメーターは CSA_EMC センス方式固有のもので、センサーから raw カウント データを取得する方法を決定します。

3.4.3.1 整定時間

整定時間パラメーターはソフトウェアの遅延を制御して、C_{MOD} コンデンサーの電圧を安定させます。各ループは、1 反復あたり 9 の CPU サイクルを持ちます。式 5 に基づいて整定時間を選択します。

$$Settling\ Time\ \geq 10 \times R_{SERIES} \times C_P \qquad \text{式 5}$$

ここで、

R_{SERIES} = 400Ω + ポート ピンとセンサー間に配置された直列抵抗 (標準値が 560Ω)

C_P = センサー ベースの静電容量

可能な値は 2～255 です。

3.4.3.2 頻度

このパラメーターは特許を取得した EMC 改善技術を実装して EMC 性能を向上させます。Freq Num = 1 は標準のスキャン アルゴリズムを、Freq Num = 3 は高度なアルゴリズムを表します。高度なスキャン アルゴリズムを有効にすると、スキャン時間と RAM の消費量がほぼ 3 倍になります。

可能な値は 1 (標準的なスキャン アルゴリズム) および 3 (高度なアルゴリズム) です。

3.4.3.3 スペクトラム拡散

このパラメーターは、スキャン中にクロック値をランダムに変更するファームウェア ベースの拡散スペクトル技術を実装し、EMC 性能を向上させます。拡散スペクトルは Freq Num を「1」に設定すると有効になります。

可能な値は 1 (有効) および 3 (無効) です。

3.4.3.4 Raw データ メディアン フィルター

このメディアン フィルターは、センサーから最新のサンプル 3 つを調べ、中央値を報告します。これは短いスパイク ノイズを除去するために使用されます。このフィルターは 1 サンプルの遅延を発生させます。このフィルターは遅延および RAM 使用量が膨大であるため、一般的にお勧めしません。このフィルターを有効にすると、RAM の (センサーの数 ×2×Freq Num) バイトとフラッシュの 100 バイトを消費します。これは初期設定では無効になっています。

可能な値は Enabled (有効) と Disabled (無効) です。

3.4.3.5 RawData IIR フィルター

この無限インパルス応答 (IIR) フィルターは、変換の結果 (raw カウント) 内のノイズを減少させます。Raw カウントをフィルターすることは XY 座標をフィルターするよりも、はるかに効果的ではあるが、より多くの RAM が必要です。このフィルターを有効にすると、フラッシュの 100 バイトが余分に消費されます。これは、初期設定では無効になっています。デフォルトの IIR 係数は 0.5 です。

可能な値は Enabled (有効) と Disabled (無効) です。

3.4.3.6 RawData IIR フィルター係数

これは、raw カウント IIR フィルターの係数です。

可能な値は 2 (½前回のサンプル + ½現在のサンプル) と 4 (¾前回のサンプル + ¼現在のサンプル) です。

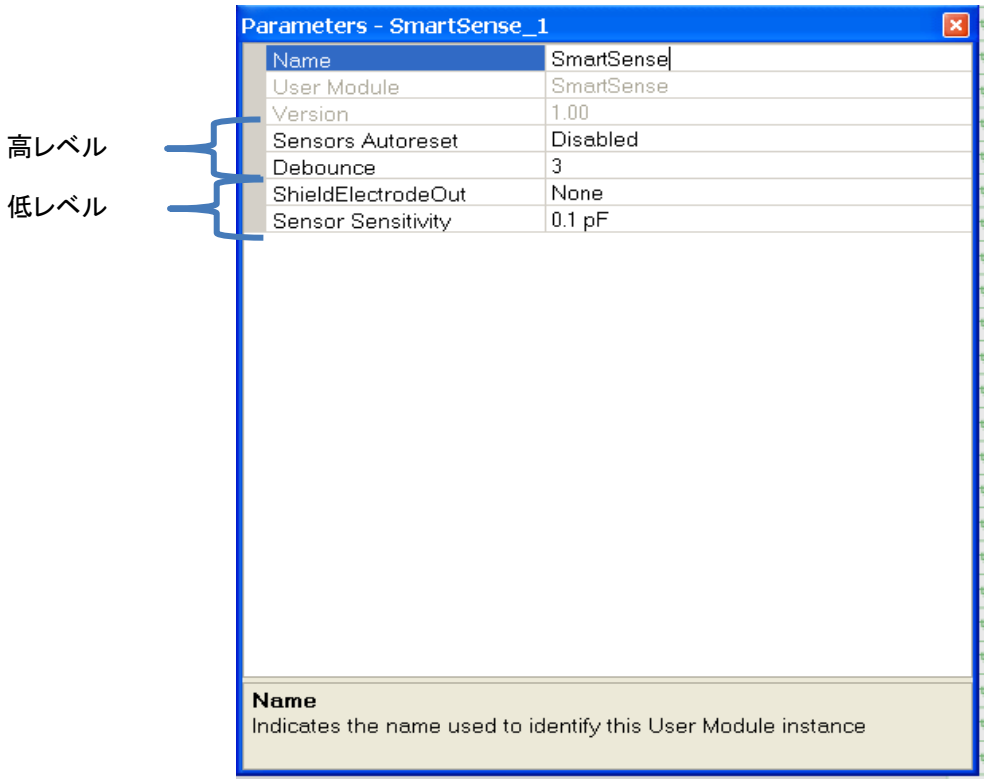
3.4.3.7 クロック

Clock パラメーターは、センサーの実効抵抗値を上げるために使用されます。センサーが占める面積が大きい場合、実効抵抗が高くなりすぎて、スイッチト キャパシタ回路の自動校正ができなくなることがあります。大型の近接センサーでは、検出感度が落ちる可能性があります。この場合、セトリング電圧がコンパレータの閾値をはるかに下回ります。内部主発振器 (IMO) の大型分周器を設定して、高静電容量を相殺する実効抵抗を増加します。

可能な値は IMO、IMO/2、IMO/4、IMO/8 です。

3.4.4 SmartSense ユーザー モジュール パラメーター

図 3-7. PSoC デザイナーSmartSense パラメーター



3.4.4.1 シールド電極出力

シールド電極は寄生容量を削減するために使用します。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。

可能な値は P0[7]または P1[2]です。

3.4.4.2 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。

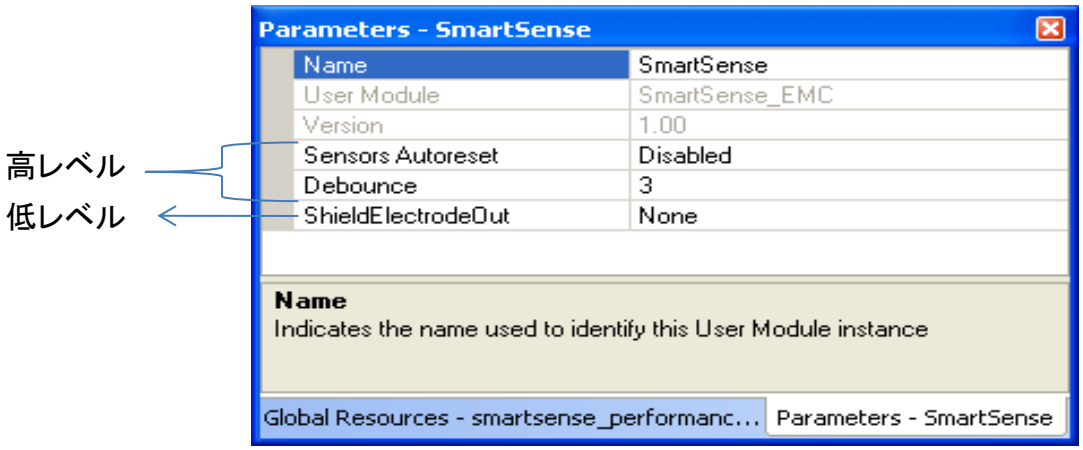
可能な値は 0.1pF、0.2pF、0.3pF、および 0.4pF です。

3.4.4.3 CapSense ユーザー モジュール パラメーター監視用の MultiChart

CapSense システムをチューニングするには、CapSense ユーザー モジュール グローバル アレイを監視する必要があります。MultiChart アプリケーションを用いると、このパラメーターの監視が容易になります。MultiChart の使用の詳細はアプリケーション ノート AN2397 を参照してください。

3.4.5 SmartSense_EMC ユーザー モジュール パラメーター

図 3-8. PSoC Designer SmartSense_EMC パラメータ



3.4.5.1 シールド電極出力

シールド電極は寄生容量を削減するために使用します。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。

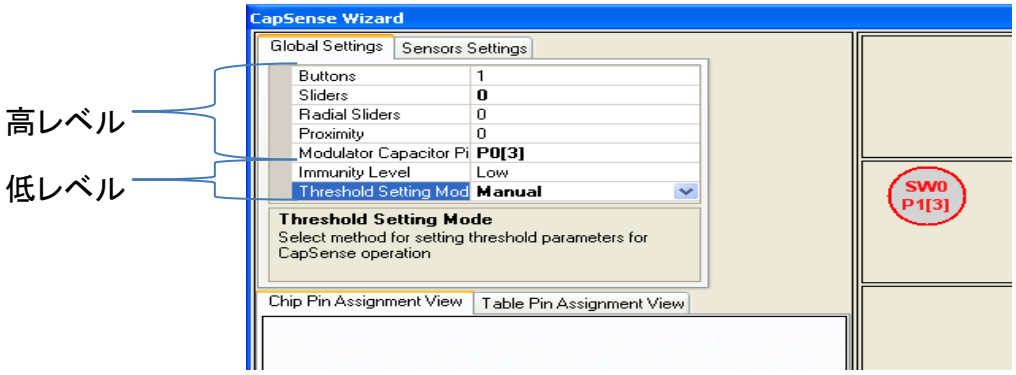
可能な値は P0[7]または P1[2]です。

3.4.5.2 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決定します。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、中度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。

可能な値は Low (低)、Medium (中)、High (高) です。

図 3-9. PSoC Designer SmartSense_EMC グローバル設定

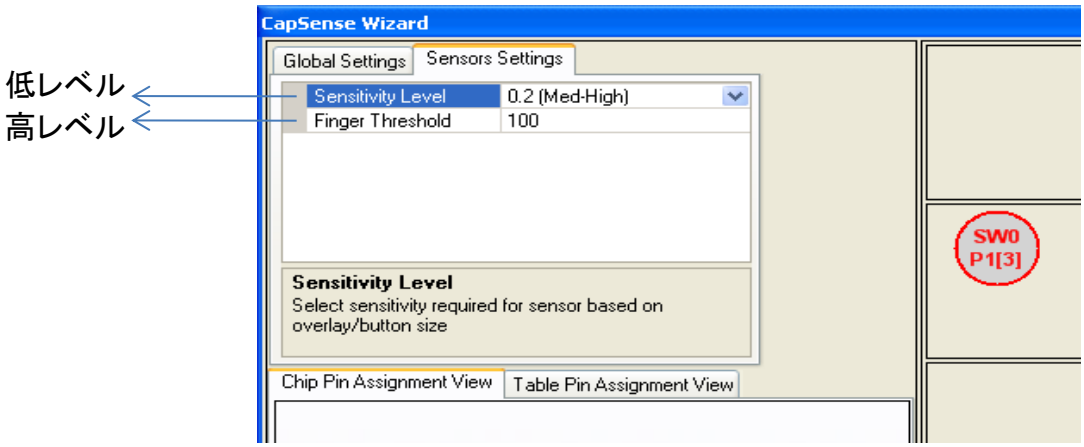


3.4.5.3 閾値設定モード

手動閾値モードを選択すると、各センサーの指閾値の設定を柔軟に行うことができます。自動閾値モードを選択すれば、SmartSense_EMC 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。

可能な値は Manual (手動) と Automatic (自動) です。

図 3-10. PSoC デザイナーSmartSense_EMC センサーの設定



3.4.5.4 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。

可能な値は 0.1pF、0.2pF、0.3pF、および 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、安定で信頼できるセンサー操作のための最適なパラメーター設定を特定するプロセスです。

4.1 一般的な注意事項

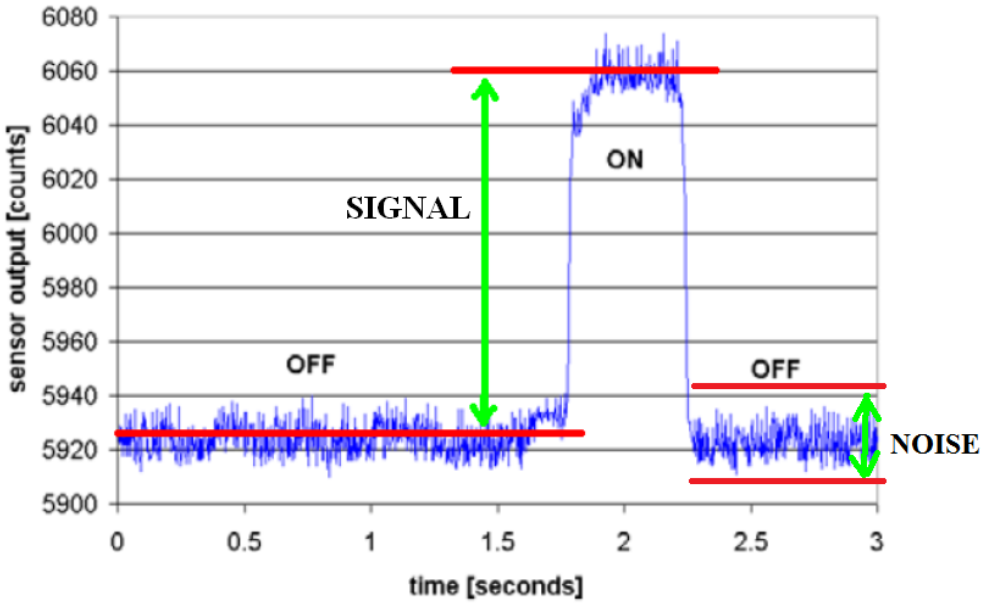
4.1.1 信号、ノイズ、および SNR

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。この性能レベルを得るには、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、タッチ センスのコンテキストにおける信号とノイズの定義づけを行います。

4.1.1.1 CapSense 信号

図 4-1 に示すように CapSense 信号は指がセンサーに触れた時に、センサー応答の変化です。センサーの出力は、センサー静電容量を追跡する値のデジタル 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、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

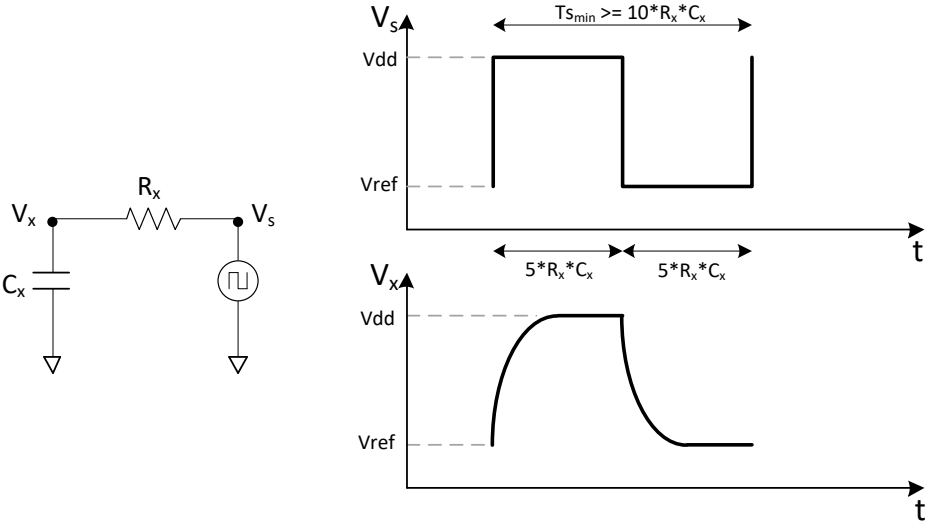
CapSense SNR とは信号とノイズの単純比です。この例では、信号が 135 カウントであり、ノイズが 26 カウントである場合、SNR は 135:26 で、5.2:1 に約分されます。CapSense の推奨最小 SNR は 5:1 で、すなわち信号値がノイズ値より 5 倍大きいです。一般的に、フィルターはノイズを減少させるためにファームウェアに実装されます。詳細はソフトウェアのフィルタリングを参照してください。

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 コンデンサー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、CSA_EMC ユーザー モジュール内のクロックと、CSD ユーザー モジュール内のプリチャージ 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、この 2 つの状態間で切り替えられます。

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。図 4-2 に示すように、電荷移動の変化速度はセンサー コンデンサーと直列抵抗から計算する 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 5RC 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。次の式に最小期間と最大周波数を示します:

$$Ts_{min} = 10 \times R_X C_X$$ 式 6

$$fs_{max} = \frac{1}{10 \times R_X C_X}$$ 式 7

例えば、直列抵抗に外部抵抗 560Ω と内部抵抗 800Ω がつながれていて、センサー静電容量が標準的であると仮定すれば:

R_X = 1.4kΩ

C_X= 24pF

この例では、時定数の値とフロントエンドの最大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

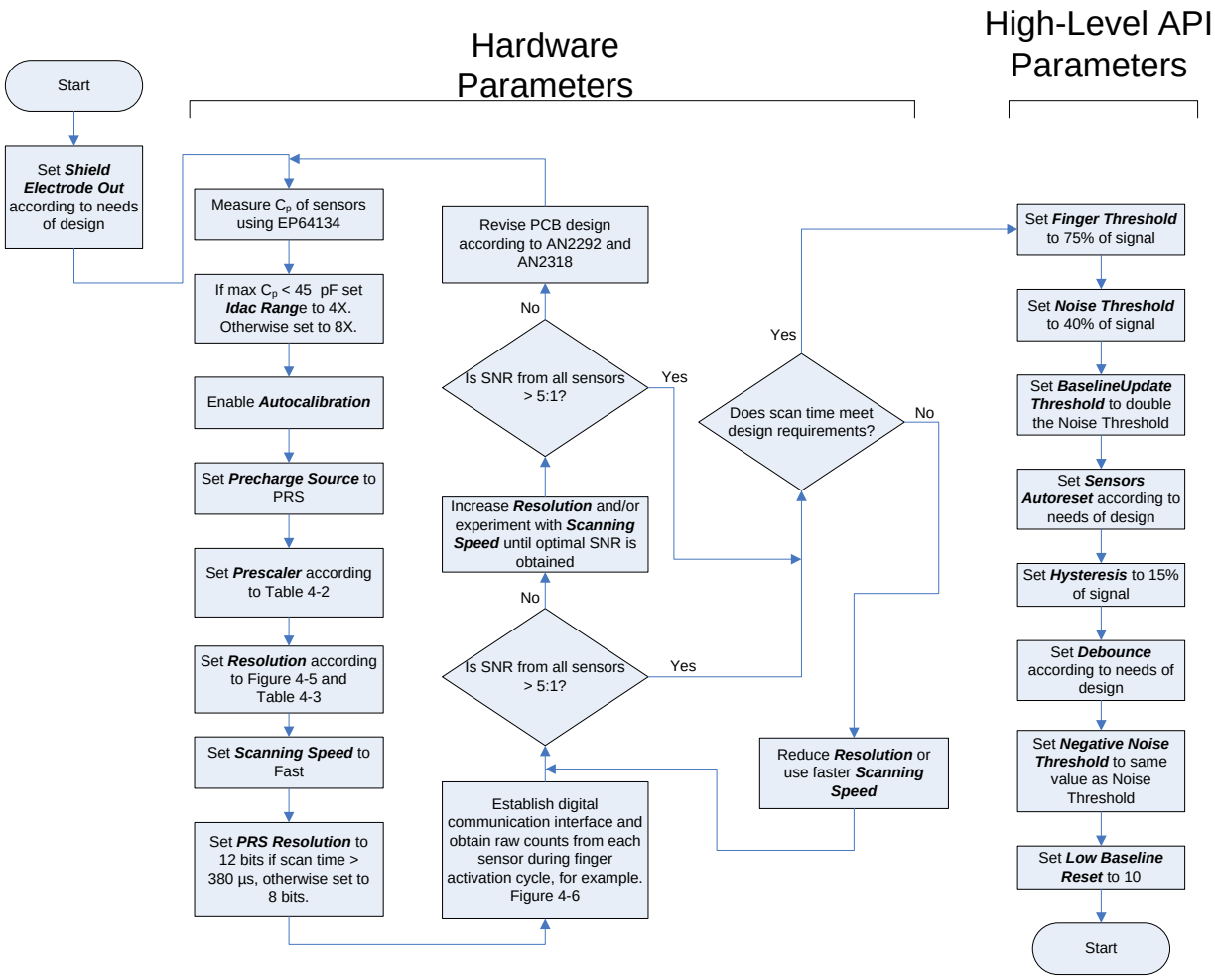
温度と湿度はどちらも平均カウント数の経時的変動の原因となります。ベースラインは CapSense 測定の参照カウント レベルであり、環境影響への補正に重要な役割を果たします。指あり状態と指なし状態などの高レベル決定は、ベースラインで設定された参照レベルに基づきます。各センサーにはそれぞれ別の寄生容量があるため、各静電容量センサーには独自のベースラインがあります。

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していることを確認します。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る可能性があります。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 CSA_EMC ユーザー モジュールのチューニング

図 4-3 は CSA_EMC パラメーターのチューニング プロセスを示したフローチャートです。CSA_EMC ユーザー モジュール 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 パラメーターの 2 つの大きなカテゴリに分類できます。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、センサー感度はハードウェア 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るため、各センサーの感度の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、対応する高レベル パラメーターが正しく調整されていることを確認する必要があります。CSA_EMC 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. CSA_EMC ユーザー モジュール パラメーターのチューニング フローチャート



4.3 CSA_EMC の推奨 C_{INT} 値

1.2nF の C_{INT} の推奨値でチューニング プロセスを開始します。チューニング プロセス中に、センサー信号が 5:1 SNR の取得に不十分であると判断する場合は、C_{INT} を増加できます。C_{INT} の推奨最大値は 5.6nF です。温度に対して C_{INT} を安定化させる場合は、X7R または NPO タイプのコンデンサーが推奨されます。また、コンデンサーの定格電圧は 5V 以上でなければなりません。

4.4 センサー C_P の測定

チューニング プロセスの最初のステップは、センサー寄生容量 (C_P) の測定です。段階を追った手順は、以下のとおりです。

1. **CPU_CLK** を SysClk/2 と等しく設定します。
2. **Clock** を IMO/8 と等しく設定します。
3. **Settling Time** を 255 と等しく設定します。
4. 特定のセンサーに対してアルゴリズムで設定された I_{DAC} コードをリードバック。値はアレイ `CSA_EMC_baDACCodeBaseline[]`に格納されます。
5. I_{DAC} コードに対応する I_{DAC} 電流を測定します。

次のコードで PSoC Designer プロジェクトを作成します。このコードでは I_{DAC} がポート ピン P1[4]に接続されます。

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

ピン P1[4]とグランドとの間に電流メーターを配置し、電流を測定します。$I_{MEASURED}$ をその値に設定します。

6. 式 $C_P = I_{MEASURED} / ((IMO/8) * 1.3)$ を使用して、C_P を計算します。

センサーCP は LCR メーターで測定することもできます。LCR メーターの端子の 1 つをセンサー ピンに接続し、もう 1 つの端子を GND に接続し、CP を測定します。

4.5 CSA_EMC クロックの予測

表 4-1はセンサーC_Pの機能としての、推奨されるプリチャージ クロック周波数を示しています。CSA_EMC クロックを、推奨されるプリチャージ クロック周波数が得られるように設定します。プリチャージ クロック周波数は選択した IMO、CSA_EMC クロック設定、および、センサーの C_P により異なります。

プリチャージ クロック周波数は、推奨される値を超えないようにしてください。

表 4-1. C_P および IMO に基づく CSA_EMC クロック設定

C_P (pF)	CSA_EMC クロック		
	IMO = 24MHz	IMO = 12MHz	IMO = 6MHz
< 5	IMO/2	IMO	IMO
5～10	IMO/4	IMO/2	IMO
10～15	IMO/4	IMO/4	IMO/2
15～20	IMO/4	IMO/4	IMO/2
20～25	IMO/16	IMO/8	IMO/4
25～30	IMO/16	IMO/8	IMO/4
30～35	IMO/16	IMO/8	IMO/4
35～40	IMO/16	IMO/8	IMO/4
40～45	IMO/16	IMO/8	IMO/4
45～50	IMO/16	IMO/8	IMO/8

4.6 整定時間の設定

整定時間パラメーターの最小値は式 8 により予測されます。

$$\text{Settling Time} = \frac{(5*\text{Cint})}{\left(\text{Clock}*\text{Cp}*25*\left(\frac{1}{\text{Fcpu}}\right)\right)} \quad \text{式 8}$$

ここで、

- C_{INT} = 積分コンデンサーの値

- クロック = プリチャージ クロックの周波数 (CSA_EMC クロック)
- C_P = センサー寄生容量値
- F_{CPU} = CPU クロック周波数

4.7 CapSense データの監視

CapSense データ表示ツールを参照してください。

4.8 SNR の向上方法

本セクションでは、SNR を向上させるいくつかの方法について説明します。

4.8.1 ノイズの削減

SNR を向上する方法の 1 つはノイズ カウントを減らすことです。本目的には、以下のオプションのいずれかを使用できます:

- ソフトウェア フィルターを使用 – 詳細はソフトウェアのフィルタリングを参照してください。
- スペクトラム拡散を有効化 – 詳細はスペクトラム拡散を参照してください。
- 耐性レベルを増加– 詳細は頻度を参照してください。

4.8.2 信号の増大

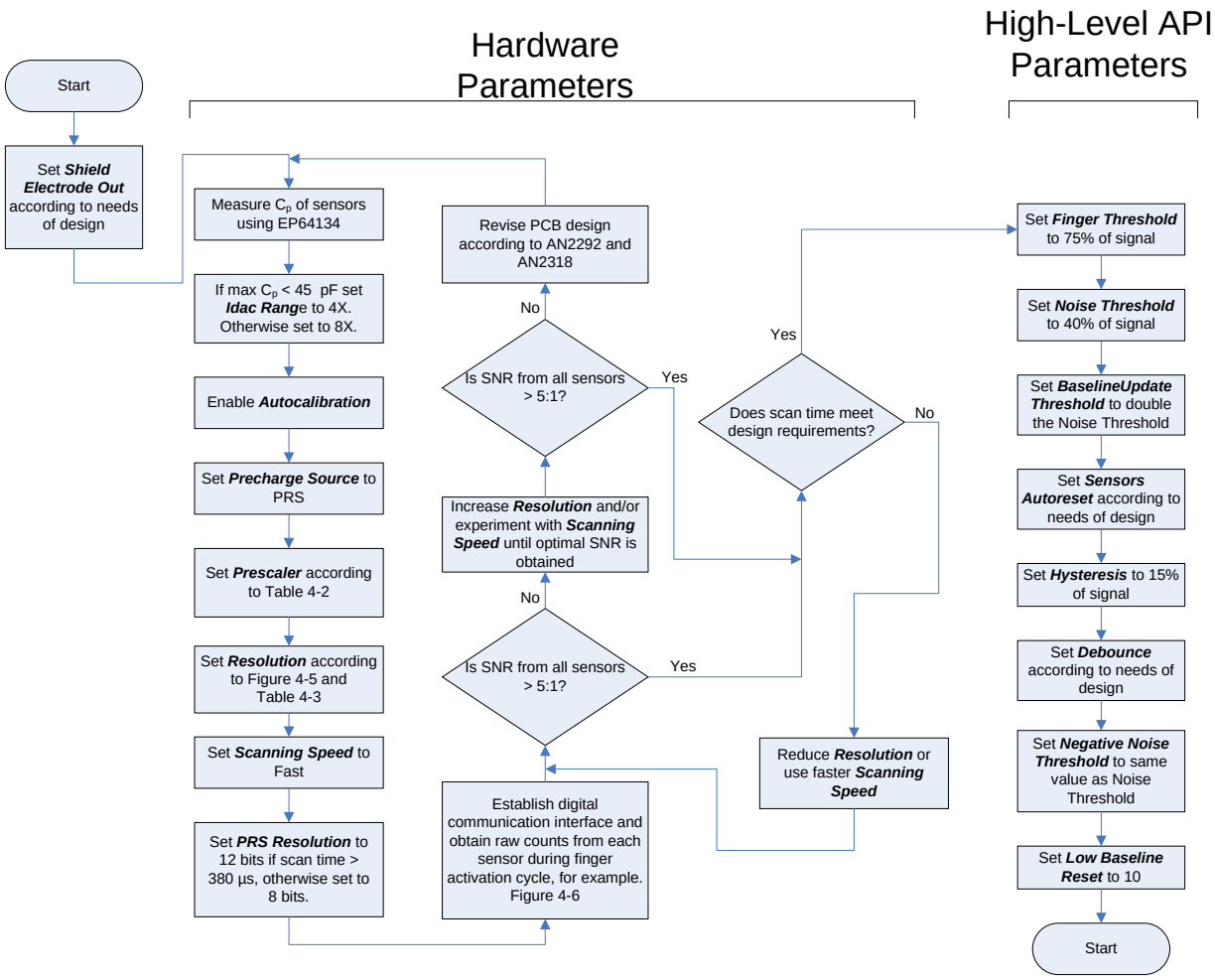
次の方法のいずれかで信号を増加して SNR を改善します:

- マクロ `CSA_EMC_BASELINE` により定義された値を増加。このマクロは *CSA_EMC.inc.*ファイルにあります。初期設定では、マクロには 0x0800 の値が割り当てられます。
- C_{INT} コンデンサーの値を増加

4.9 CSD ユーザー モジュールのチューニング

図 4-4 は CSD UM 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 UM 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 つの大きなカテゴリに分類できます。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ス 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、この点を考慮して、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ことを確認する必要があります。CSD 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 4-4. CSD 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 方法が使用す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、どのように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。

初期設定では、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設計では、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、ScanSensor() API 機能を呼び出す前に SetIdacValue()、SetPrescaler()、および SetScanMode() API 機能を呼び出して、各センサーのそれぞれのハードウェア パラメーターを設定可能です。

表 4-2 および表 4-4 はセンサーC_Pに基づくいくつかの主要なハードウェア パラメーター 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、PCB レイアウト、その他のコンポーネントの近接度に依存します。このため、C_P はシステムが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 Code Examples Design Guide に記載されている「Measuring Absolute Sensor Capacitance with a CY8C20xx6A CapSense Controller」のサンプル 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。設定と使用の説明のサンプル コード関連資料を参照してください。

4.9.1 CSD 用の推奨 C_{MOD} 値

CSD ベースのデザイン用の C_{MOD} の推奨値は 2.2nF です。温度に対して C_{INT} を安定化させる場合は、X7R または NPO タイプのコンデンサーが推奨されます。また、コンデンサーの定格電圧は 5V 以上でなければなりません。

4.9.2 ShieldElectrodeOut

本設計の ShieldElectrodeOut を有効化します。

4.9.3 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。

4.9.4 自動校正

CY8C20xx6A CSD 設計において、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。

4.9.5 I_{DAC} 値

このパラメーターでは自動校正が無効になっている時 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。

4.9.6 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)、または PRS (分周した IMO クロックを疑似ランダム発生器に渡しペクトラム拡散クロックを生成する) 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初期設定となっています時には、プリスケーラ プリチャージ源はより高い SNR を提供することがあります。しかし銅製の回路を使用する場合、通常この 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。

4.9.7 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 CP に依存します。表 4-2 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 4-2. プリチャージ源、IMO および CP 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

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
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。

4.9.8 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 4-3 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時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 4-5 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 4-5. オーバーレイの厚さと円形センサーの直径に対する指の静電容量 (C_F)

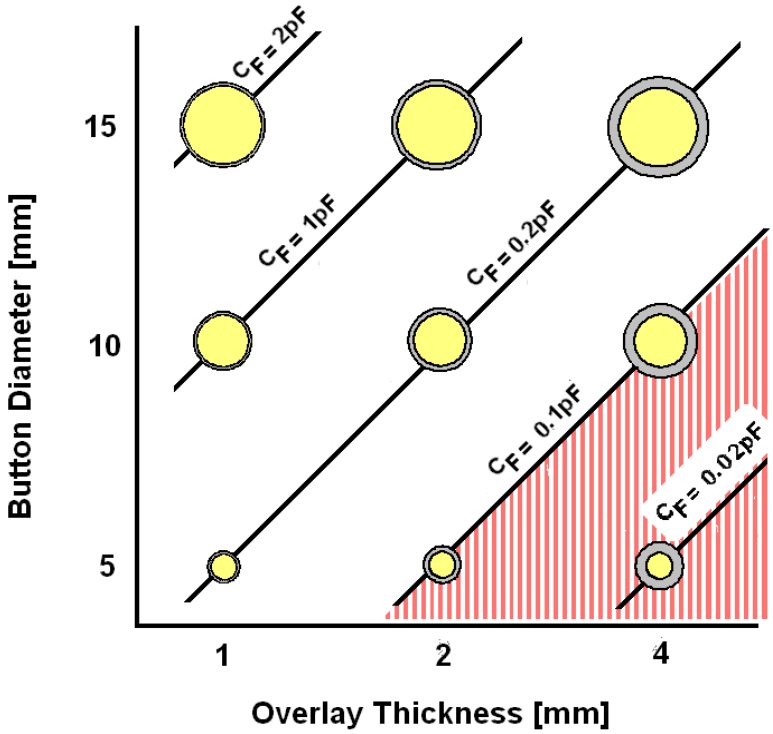


表 4-3. 指静電容量と C_P に基づく分解能設定

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

4.9.9 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お勧めします。必ずではありませんが、遅いスキャン速度により SNR 比 が高くなり、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 4-4 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 4-4. 分解能とスキャン速度に応じた、センサーのスキャン時間 (単位 µ)

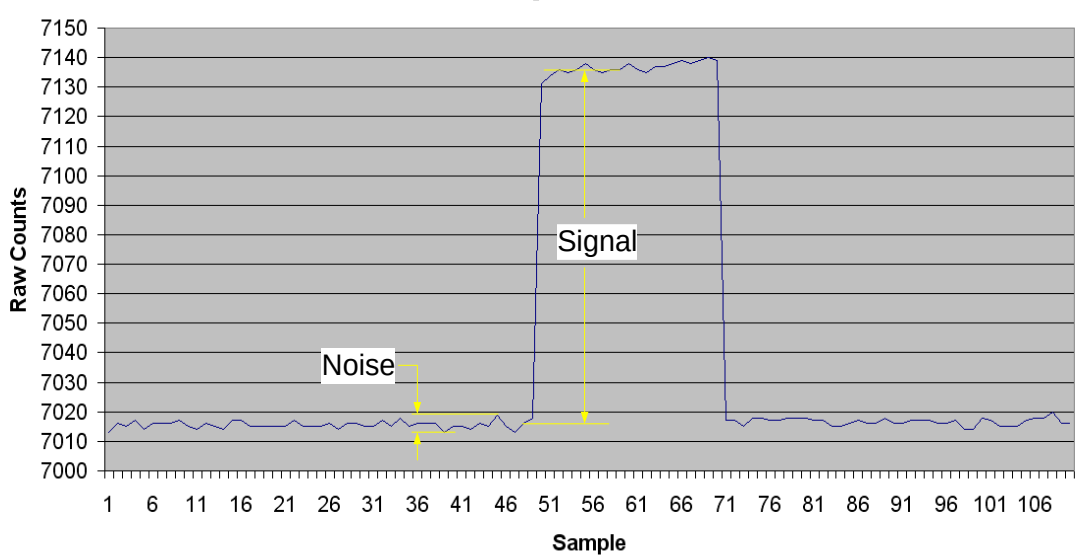
分解能 (単位: ビット)	スキャン速度			
	超高速	高速	通常	低速
9	57	78	125	205
10	78	125	205	380
11	125	205	380	720
12	205	380	720	1400
13	380	720	1400	2800
14	720	1400	2800	5600
15	1400	2800	5600	11000
16	2800	5600	11000	22000

4.9.10 高レベル API パラメーター

この高レベルの API パラメーターで、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応じ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判定します。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 CSD_waSnsBaseline[]、CSD_waSnsResult[]、および CSD_waSnsDiff[]と呼ばれるアレイに格納されます。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、指信号強度に基づいていることが分かります。ノイズ、信号の強度は 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よります。従って、これらのパラメーター設定の基礎となったデータはシステムが最終組み立て状態で、使用中と同じ EMI 環境において、初期の位置で取得しなければなりません。

図 4-6 は指タッチ有効なサイクルの間、つまり、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に、センサーから得られた標準 raw カウントを示したものです。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 システムの作動の安定性のためには信号対ノイズ比 (SNR) は 5:1 以上でなければなりません。SNR が 5:1 未満である場合、Getting Started with CapSense のガイドラインに示しているようにハードウェア パラメーターを調整する、または PCB レイアウトを変更する、またはその両方を行って、SNR を 5:1 以上に引き上げる必要があります。

図 4-6. 指のアクティブ サイクルの間にセンサーから得られた代表的な raw カウント



4.9.11 高レベル パラメーターの設定

以下は最適なパラメーター設定の選択の開始点についての推奨事項です:

- **指閾値**: センサーがオンの状態で raw カウントを 75%に設定
- **ノイズ閾値**: センサーがオフの状態で raw カウントを 40%に設定
- **負のノイズ閾値**: (ノイズ閾値／2) に等しく設定
- **ベースライン更新閾値**: ノイズ閾値の 2 倍に設定
- **ヒステリシス**: センサーがオンの状態で raw カウントを 15%に設定
- **低ベースライン リセット**: 50 に設定
- **センサー自動リセット**: 設計要件に応じて設定
- **デバウンス**: 設計要件に応じて設定

4.10 SmartSense ユーザー モジュールを使用

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 を利用して、チューニングを必要としない CapSense デザインを作成することができます。PSoC Designer 5.1 の SmartSense ユーザー モジュールを使用して、SmartSense デザインを作成することができます。また本セクションでは、既存の CSD CapSense デザインを SmartSense に変換する方法も説明します。

4.10.1 SmartSense のガイドライン

ガイドラインに従ってアプリケーションで SmartSense ユーザー モジュールを使用してください。

SmartSense では、静電容量ユーザー インターフェース設計が本デザイン ガイドの前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 ユーザー モジュール パラメーター (I_{DAC} 値、プリスケーラ周期、クロック デバイダー、スキャン速度、分解能など) は、実行中に SmartSense 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る場合を除き、ファームウェア内のこれらの CSD パラメーターを改変する API を使用してはなりません。
- 既存のデザインを CSD から SmartSense に変換、
  - ☐ まず、CSD パラメーターを設定、変更するすべての API をプログラムから削除することを確認してください。

- ☐ デザインのすべての CapSense センサーの寄生容量が環境と PCB 生産プロセスの変化に応じて、5pF～45pF の範囲内であることを確認してください。
- ☐ 推奨 C_{MOD} コンデンサー (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 ピンに接続された状態を確認してください。

4.10.2 相違点

SmartSense ユーザー モジュールと標準 CSD ユーザー モジュールの違いは:

- SmartSense ユーザー モジュールと標準 CSD ユーザーモジュールは同じ API をサポートします。従って、ユーザー モジュール インスタンス名以外に、その他の API のインストール、設定、起動や呼び出しに変更する必要はありません。
- チューニングに関連するすべてのパラメーターが SmartSense ユーザー モジュールで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} コンデンサーの値は 2.2nF に限定します。定格電圧が 5V 以上の X7R コンデンサーをすべての CapSense アプリケーションに使用することをお勧めします。
- 性能を最大限にしながら CapSense の安定した作動を確保するために、SmartSense アルゴリズムで各センサーの SNR を 5:1～11:1 の範囲内に維持します。
- SmartSense 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、アルゴリズムで 1 センサー当たり 410µs～2.8ms に制限します。

4.10.3 SmartSense 用の推奨される C_{CMOD} 値

SmartSense ベースの設計の推奨される C_{CMOD} 値は 2.2nF です。温度の変化時に C_{INT} が安定するために X7R または NPO タイプのコンデンサーの使用をお奨めします。コンデンサーの定格電圧は 5V 以上である必要があります。

4.10.4 SmartSense ユーザー モジュール パラメーター

このユーザー モジュールでは、設定する必要なパラメーターは 4 つだけです。それらは以下の通りです:

- センサーの自動リセット
- デバウンス
- 変調器コンデンサー ピン
- 感度レベル

4.10.4.1 センサーの自動リセット

このパラメーターにより、ベースラインを連続的に、または信号差がノイズ閾値以下の場合にのみ更新することを決めます。有効化されている場合、ベースラインは常に更新されます。この設定では、センサーがオンとなる最大時間を制限します (一般的に 5～10 秒) 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。

4.10.4.2 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有効状態への遷移に追加します。センサーが無効状態から有効状態になるには、指タッチ信号がデバウンス数の連続したスキャンで検知される必要があります。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。

4.10.4.3 変調器コンデンサー ピン

本パラメーターで 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]と P0[3]ピンは使用可能です。

注: SmartSense が正常に動作するには、2.2nF の外付けコンデンサーが必須です。

4.10.4.4 感度レベル

センサー信号の強さは感度により増減されます。感度の低い値 (0.1pF) により、センサーの信号が強化されます。オーバーレイの厚いデザインでは、デバイスが正常に実行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 UM 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、0.1pF (高感度) に設定したセンサーは、0.2pF (中高感度) のセンサーよりも、スキャン時間が長いです。

最善の調整方法は、センサーで 5:1 の SNR を達成することのできる最高の感度を調べることです。まず、最高の感度 (0.4pF) を使用してチューニングをし、そして SNR が 5:1 になるまで感度を低下してみてください。

4.10.5 SmartSense_EMC ユーザー モジュール用のガイドライン

SmartSense 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_EMC ユーザー モジュールに適用されます。CapSense デザインと SmartSense ベースのデザインの共通ガイドラインは、CapSense Getting Started Guide を参照してください。このセクションでは、SmartSense_EMC ユーザー モジュールのいくつかの重要な問題点を説明します。

4.10.5.1 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_EMC ユーザー モジュール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 メモリ使用はユーザー モジュールで選択されたイミュニティ モードによって決まります。

- 中イミュニティ モードのセンサーのスキャン時間は、低イミュニティ モードのセンサーのスキャン時間の 2 倍長いです。高イミュニティ モードのセンサーのスキャン時間は、低イミュニティ モードのセンサーのスキャン時間より 3 倍長いです。
- 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。中イミュニティ モードのセンサーの応答時間は、低イミュニティ モードのセンサーの応答時間の 2 倍になります。同様に、高イミュニティ モードのセンサーの応答時間は低イミュニティ モードのセンサーの応答時間よりも 3 倍長いです。
- SmartSense_EMC ユーザー モジュールは 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、最高のイミュニティ モード (高) で使用する RAM メモリ容量は、低イミュニティ モードでの RAM メモリ使用容量の約 3 倍です。中イミュニティ モードで使用する RAM メモリ容量は、低イミュニティ モードで使用する RAM メモリ容量より約 2 倍だけです。

4.10.5.2 IMO 許容誤差とスピードが必要なタスク

SmartSense_EMC が有効なデバイスの IMO 許容誤差は+5%から–20%です。

- スピードの必要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 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) の時、IMO 許容誤差を検討して、安定したアプリケーション ファームウェアを確保する必要があります。

4.10.5.3 I²C の作動速度

I²C インターフェースの作動周波数は、SmartSense_EMC が有効なデバイスのユーザー モジュールの実際周波数の最高 80 %に制限されています。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 モジュールでは、クロック周波数が 400kHz の場合、I²C インターフェースは、最高 320kHz で作動可能です。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、作動周波数は、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作動することが必要です。これに違反すれば、データの破壊、I²C バス混乱、I²C ユーザー モジュールの不整合な作動が発生します。

- 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。

4.10.6 CapSense センサーのスキャン時間

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 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。量産のデザインでは、PCB 寄生容量変動によって変動します。

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、SmartSense 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 9 と以下の表で計算できます。

$$Scan\ time = Sampling\ time\ (ST) + Processing\ time\ (PT)$$ 式 9

以下の表に、様々な IMO と感度レベルに応じるサンプリング時間を示します。

表 4-5. IMO 付属センサーのサンプリング時間 = 24MHz

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 4-6. IMO 付属センサーのサンプリング時間 = 12MHz

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 4-7. IMO 付属センサーのサンプリング時間 = 6MHz

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 4-8 に様々な CPU 周波数に対応する処理時間の値を表示します。

表 4-8. 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、ある CapSense システムが 24MHz IMO 周波数、6MHz CPU クロック(IMO/4)、そして SmartSense 感度レベル 0.3pF で設計されている場合、約 15pF 寄生容量のセンサーのスキャン時間は、式 9 を用いて前出の表から計算することができます。

前述設定のサンプリング時間は、(24MHz の IMO、感度 0.3pF) 表 4-5 から、680µs が選択されます。前述のコンフィギュレーション (CPU クロック周波数 24MHz) の処理時間は、表 4-8 から選択され、284µs です。

従って、このコンフィギュレーションの総スキャン時間は、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。

4.10.7 SmartSense 応答時間

標準 CSD と代表的な CapSense スキャンファームウェアが付いた場合は、以下のアプリケーションを考慮します。

- 3 個の CapSense センサーで、寄生容量が 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

前述の表によると、個々のセンサーのスキャンには 482µs かかり、3 個のセンサーのスキャン時間は 1.45ms です。以下のファームウェアの実例では、ファームウェアの追加の実行には 1ms が必要であるため、ループの実行時間は 2.45ms となります。

```
while (1)
{
    SmartSense_ScanAllSensors();
    SmartSense _UpdateAllBaselines();

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

つまり、CapSense センサーがアクティブ化された場合、ファームウェアによって 7.35 ms(センサーは、連続したデバウンス数のスキャンの間アクティブである必要があります)以内にセンサーが ON 状態になります。これは通常、CapSense システムの応答時間とみなされます。

プロセスのばらつきによってセンサーの寄生容量が変動する場合、整合性の維持のために、寄生容量に関わるスキャン時間が変わると、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 デザインを確立するためのガイドラインを次に説明します。

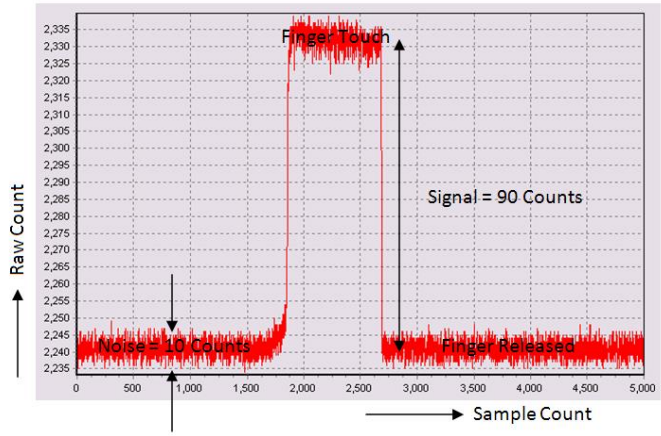
4.10.8 SmartSense_EMC ユーザーモジュールを使って最低限の S/N 比を確保する方法

SmartSense_EMC は、めんどうな調整プロセスを必要としない 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_EMC 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 パラメーターをモニター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。調整プロセスにおいて、センサー生カウント (SmartSense_EMC_waSnsResult)、センサーの正規化された信号 (SmartSense_EMC_baSnsSignal) と、センサーフィンガー閾値 (SmartSense_EMC_baBtnFThreshold)を確認します。データのモニターとして、LCD などの数値表示ディスプレイは遅く、データダイナミックス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 ブリッジ コントロール パネルをデータ監視に使用することをお勧めします。

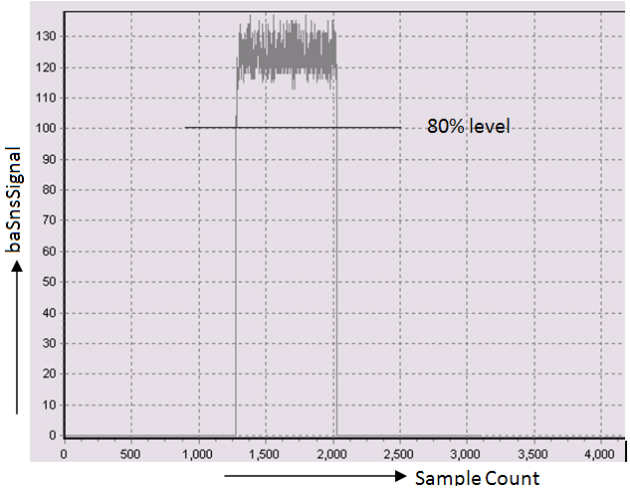
2. 感度レベルを 0.4pF (低) に設定し、S/N 比を計算します。図 4-7 にフィンガータッチの代表的生カウントグラフを示します。CapSense ベスト プラクティスによると、デザインが堅牢にするためには S/N 比が 5:1 を超える必要があります。測定された S/N 比が 10:1 以上であれば、S/N 比が 5:1～10:1 の範囲内になるまで、感度レベル値を次のレベルに下げます。

図 4-7. 標準センサーへの指タッチがある時の raw カウント グラフ



3. デザインにおいて、自動フィンガー閾値を使用している場合、プロセスは前のステップで完了します。フレキシブルフィンガー閾値を使用している場合、フィンガー閾値をセットしてプロセスが完了します。フィンガー閾値をセットする場合、センサー信号(SmartSense_EMC_baSnsSignal)をモニターし、フィンガー閾値をセンサーにタッチした場合のセンサー信号値の 80 パーセントにセットします。これでプロセスは完了です。図 4-8 に代表的センサー信号とフィンガー閾値を示します。

図 4-8. 標準センサーへの指タッチがある時のセンサー信号



4.10.9 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 (以下の実例コードを参照) 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 ルーチンは 2.45ms の速度で実行されます。プロセスの変動による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の実行時間に応じるポート ピン切り替えのコード例です。

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

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

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

Port_0[1]上の信号の期間は、4.9ms です (ポート ピンが切り替える時、この期間はループ時間の 2 倍になります)。センサーの寄生容量を約 15pF まで引き上げれば、スキャン時間は 1.78ms になり、よって Port_0[1]上の信号の期間は 5.6ms になります。

センサーの寄生容量が SmartSense 容量群の境界に近接(例えば、10pF 境界に非常に近い 9pF)していれば、あるアプリケーションにおいて、プロセス変動のために、近傍の(異なった)スキャン時間が、SmartSense によって選択されることがあります。従って、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、タイマー16 ユーザー モジュールによる主ループ実行時間の整合化向けの簡単なファームウェア実行例です。

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

#pragma interrupt_handler myTimer_ISR_Handler;
void myTimer_ISR_Handler( void );

void main()
{
M8C_EnableGInt;

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

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

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

        /* Wait till timer expires or
         sleep here */
```

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

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

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

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

この前の例で、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機します。タイマーの時間は、ワーストケースの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 容量群の境界に近接している場合、計算に上のスキャン時間を選択(表 4-6 を用いて) してください。

SmartSense ユーザー モジュールによって、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、容易に実装することが可能になります。調整プロセスの困難さを排除し、製造中の PCB の変動やその他の変動があっても、歩留まりの向上に有効です。従って、既存 CSD ベースの CapSense デザインを SmartSense に変換し、新しいデザインには SmartSense を使用することが推奨されます。

SmartSense のメインループ実行時間とスキャン時間は、プロセスの変動に対応して変化します。CapSense の性能はなんら影響されませんが、ファームウェア開発者は SmartSense 自動調整テクノロジーを備えた CapSense PLUS アプリケーションを実行する場合、この点を考慮すべきです。

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。プリント基板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。詳細な情報は、 Getting Started with CapSense 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、 式 1 は指の静電容量を示します

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

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、誘電体定数の高いオーバーレイ素材を選択し、オーバーレイ厚さを薄くし、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	絶縁破壊電圧 (V/mm)	最小値 12kV でのオーバーレイの厚さ (mm)
空気	1200 ～ 2800	10
木材 (乾燥)	3900	3
ガラス (一般的)	7900	1.5
ガラス – ホウケイ酸塩 (Pyrex®)	13,000	0.9
PMMA Plastic (プレキシグラス®)	13,000	0.9
ABS	16,000	0.8
ポリカーボネート (Lexan®)	16,000	0.8
フォーマイカ (高圧メラミン化粧板)	18,000	0.7
FR-4	28,000	0.4
PET 膜 – (Mylar®)	280,000	0.04
ポリイミド膜 – (Kapton®)	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、接着剤は使用されます。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ています (3M™商品番号: 467MP、468MP)。

5.2 ESD 保護

安定した ESD 耐性は、慎重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 放電現象に耐えることが可能です。ほとんどの場合、オーバーレイの素材がコントローラー ピンへの完全な ESD 保護を提供します。表 5-1 は、IEC 61000-4-2 で指定されているように、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を一覧表示しています。オーバーレイの素材が十分な保護を提供しない場合、 ESD 対策は次の順番で適用されます: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高い絶縁破壊電圧を備えている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。適切な距離を保つことができない場合は、絶縁破壊電圧の高い素材による保護レイヤを ESD ソースと CapSense コントローラーの間に設けてください。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

基板にはコンポーネントの密度が高い場合は、放電現象を避けることは難しいかもしれません。この場合、放電が起きる場所を制御することにより、CapSense コントローラーを保護することができます。標準的な手順としては、シャーシの接地に接続された回路ボードの外周部に保護リングを配置します。「PCB レイアウト ガイドライン」でお勧めするように、ボタンやスライダー センサーの周囲にハッチング グランド面を施すことによってセンサーと CapSense コントローラーへの ESD の影響を回避することができます。

5.2.3 クランプ

CapSense センサーは、意図的にタッチ面に近く設置されているため、放電経路をリダイレクトすることは不要になる場合もあります。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お勧めします。

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	接触放電の最高限度	空中放電の最高限度
メーカー	型番				
Littlefuse	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**: プリント基板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値は 560Ω。
- ☐ I^2C や SPI などの通信回線の推奨直列抵抗は 330Ω。

- **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**: システムを CapSense 入力と隔離するために、これらのシステムを LCD インバーターおよびスイッチング電源 (SMPS) のようなノイズ源と離れて配置します。干渉を防ぐもう 1 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 クロックに低い周波数を選択することも、CapSense センサーからの放射エミッションを低減するのに有効です。このクロックは、プリスケーラ オプションを使用するファームウェアで制御しています。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 伝導イミュニティおよびエミッション

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。これらの接続には電源や通信ラインを含みます。CapSense コントローラーは低消費電力デバイスのため、伝導エミッションを回避しなくてはいけません。以下のガイドラインに従って伝導エミッションおよびイミュニティを低減することができます。

- データシートで推奨しているようにデカップリング コンデンサーを使用します。
- システム電源への入力に双方向フィルターを追加します。伝導エミッションおよびイミュニティの両方に効果的です。パイ型フィルターで電源ノイズの鋭敏な部品への影響を防止しながら、部品のスイッチング ノイズが電源面に戻るのを防止することができます。
- CapSense コントローラーPCB がケーブルで電源に接続されている場合は、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てください。
- 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 ソフトウェアのフィルタリング

ソフトウェア フィルターの使用も高レベルのシステム ノイズ対応の技術です。表 5-3 は CapSense に役立つフィルターの種類の一覧です。

表 5-3. CapSense フィルターの表

タイプ	説明	アプリケーション
Average (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)	電源からの周期的なノイズ
IIR	RC フィルターに類似したステップ レスポンスを備えた、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)
Median (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	モーターおよび電源切り替えによるスパイクノイズ
Jitter (ジッター)	前の入力に基づいて電流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)；特にスライダーのセントロイド データに役立つ
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	タッチ以外のイベントにおいて、CapSense データの送信をブロックするために広く使用される
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、偶発的なマルチボタン選択などの特殊なシナリオに対応

表 5-4 に、様々なソフトウェア フィルターの RAM とフラッシュ要件を示します。各フィルター種類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ています。ここで一覧表示している要件は、ImageCraft コンパイラと ImageCraft Pro コンパイラ双方に該当します。

表 5-4. RAM およびフラッシュの要件

フィルターの種類	フィルターの順序	RAM (センサーごとのバイト)	フラッシュ (バイト) ImageCraft コンパイラ	フラッシュ (バイト) ImageCraft Pro コンパイラ
Average (平均)	2～8	6	675	665
IIR	1	2	429	412
	2	6	767	622
Median (メジアン)	3	6	516	450
	5	10	516	450
Raw カウントのジッター 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ー 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。

- GPIO 駆動モードを低電力にセットします。
- 高電力ブロックの電源の切断。
- 低電力の場合 CPU 速度を最適化します。
- 低い V_{DD} で動作させる。

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有効です。

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。スリープ状態のデバイスが消費する電流はアクティブ時の電流よりも小さくなります。

長期にわたってデバイスが消費した平均電流は、次の式で計算することができます。

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

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 11

5.5.3 応答時間対消費電力

式 11 に示されているように、平均消費電力は I_{AVE} または V_{DD} を低下させて、削減することができます。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。よって、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、デバイスはその時間の大半をスリープ スキャン モードで使い、前のセクションで説明したように、定期的にセンサーをスキャンしてスリープ状態に入り、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操作すると、デバイスは連続スキャンモードに移行します。連続スキャンモードでは、スリープを起動することなくセンサーを定期的にスキャンし、極めて

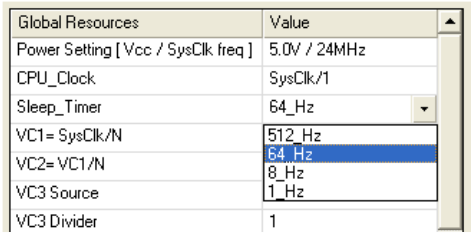
優れた応答時間を提供します。デバイスは、指定されたタイムアウト期間では、連続スキャン モードのままになります。ユーザーがこのタイムアウト期間内にセンサーを操作しない場合、デバイスはスリープ スキャン モードに戻ります。

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時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、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(連続スキャン モード)。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。出力ピン状態の切り替えは、オシロスコープで確認することが可能なタイム 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。測定された消費電流を 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. オシロスコープでの切り替え出力ピンを監視し、2 つの切り替え間の周期を測定します。これによりアクティブ時間が分かります。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、スリープ時間が求められます。$T_{SLP} = T – t_{ACT}$.
7. 式 10 により、平均電流を計算します。
8. 式 11 により、平均消費電力を計算します。

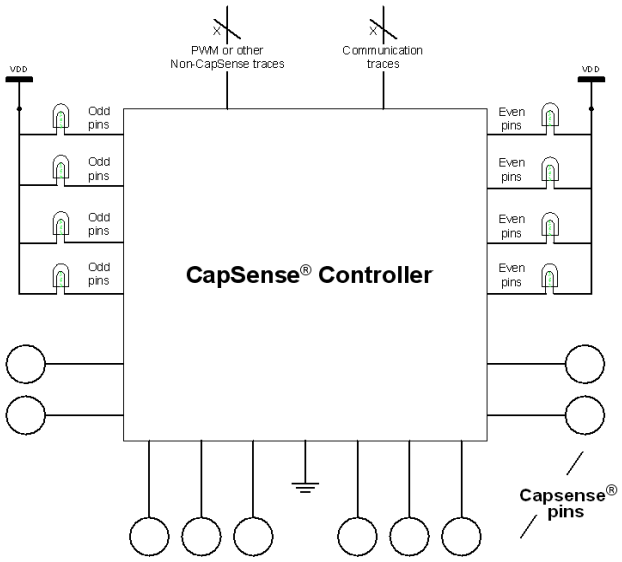
図 5-1. グローバル リソース ウィンドウ



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制限す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 は、32 ピン QFN パッケージ用のこの分離の基本バージョンを表しています。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 のポート分離 (推奨)

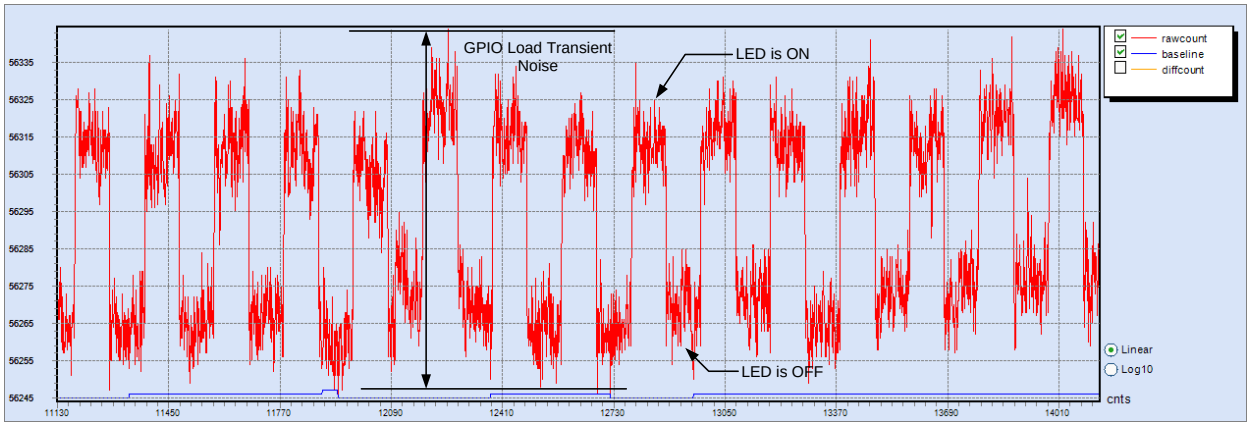


CapSense コントローラーのアーキテクチャによって偶数、奇数のポート ピン番号に対し電流量の制限が行われます。奇数ピンは、ピン番号が奇数であるどのポート ピンでも構いません。CapSense コントローラーでは、奇数ポート ピンの電流量が 100mA であれば、すべての奇数ポート ピンを通して引き出される電流量の合計は 100mA を超えないようにします。総電流量の制限に加えて、CapSense コントローラーのデータシートに定義されているように個々のポート ピンの最大電流制限もあります。

5.7 GPIO の負荷瞬時変化

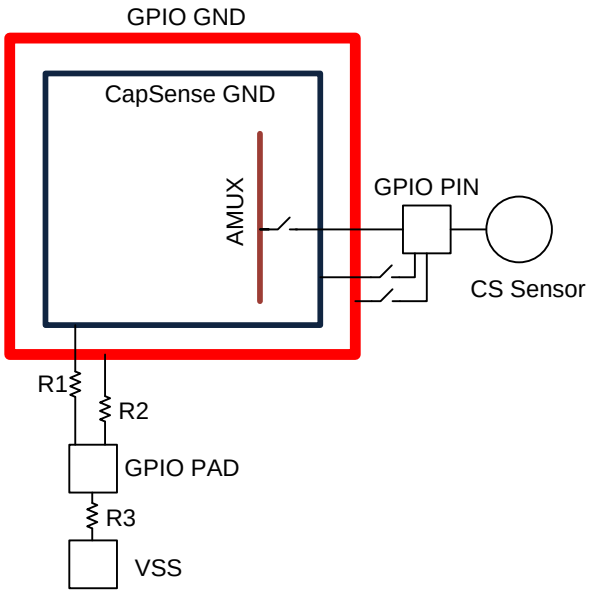
ポート ピンをストロング LOW に駆動して、GPIO が大きな電流 (>10mA) をチップのグランドに吸い込む時、ノイズが CapSense システムに混入します。GPIO ピンを介してグランドへ流れる電流量の瞬時変化は、GPIO 負荷瞬時変化と呼ばれています。GPIO 負荷瞬時変化による CapSense システムへ混入するノイズは、図 5-3 に示したように、GPIO 負荷瞬時変化ノイズと呼ばれます。このセクションでは、ハードウェア技術を使ってこのノイズを低減し、ファームウェア技術でこのノイズを補正する方法を説明します。

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 ワイヤー抵抗 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 がシンク電流である場合、センサーは完全に放電されません。これにより、センサーの raw カウントは増加します。

図 5-4. CY8C20x66A/S のグランド構造



注: R1、R2、R3 はボンディング ワイヤー抵抗です。

堅牢な CapSense デザインのために、最悪の GPIO 負荷瞬時変化ノイズが指のタッチの信号の 30%より小さいである必要があります。CapSense システムでは、GPIO の状態が電流の流れなしの状態 (例えば、すべて LED がオフ) から電流の流れの最大状態 (即ち、すべての LED がオン) に遷移した時に最悪のノイズが生じます。

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 センサーと近接センサーは、S/N 比 > 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の信号より高くて、センサーの誤トリガーを発生させることもあります。次のセクションでは、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

a. 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、「*Getting Started with CapSense*」のデザイン ガイドに記載されているレイアウト ガイドラインに従って、センサーの C_P を最小化することをお勧めします。

b. 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お勧めします。GPIO がデータシートに示される最大の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。

c. LED に適切なピンの選択
すべての CapSense コントローラーは、高電流の吸い込み／吐き出し可能なポート ピンを提供します。高電流シンクやポート ピンからのソースを使用する場合、デバイスの接地ピンに最も近いポートを使用し、GPIO 負荷瞬時変化ノイズを最小限に抑える必要があります。

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

GPIO 負荷瞬時変化ノイズによる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. インスタント 1 では、センサー上の指タッチがなく、LED がオフです。
2. インスタント 2 では、センサーには指の存在があり、raw カウントの変化は指の閾値より高いです。
3. raw カウントの変化は指の閾値より高いため、インスタント 3 では、LED はオンになります。
4. LED がオンになった時、GPIO 負荷瞬時変化ノイズのため、raw カウントはさらに変化します。
5. インスタント 4 では、指が離れていても、GPIO 負荷瞬時変化ノイズによる raw カウントの変化により raw カウントは初期値に戻りません。この変化が指の閾値より高い場合、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止するために、センサーのベースラインを補正する必要があります。ベースラインの補正は以下で説明します。

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

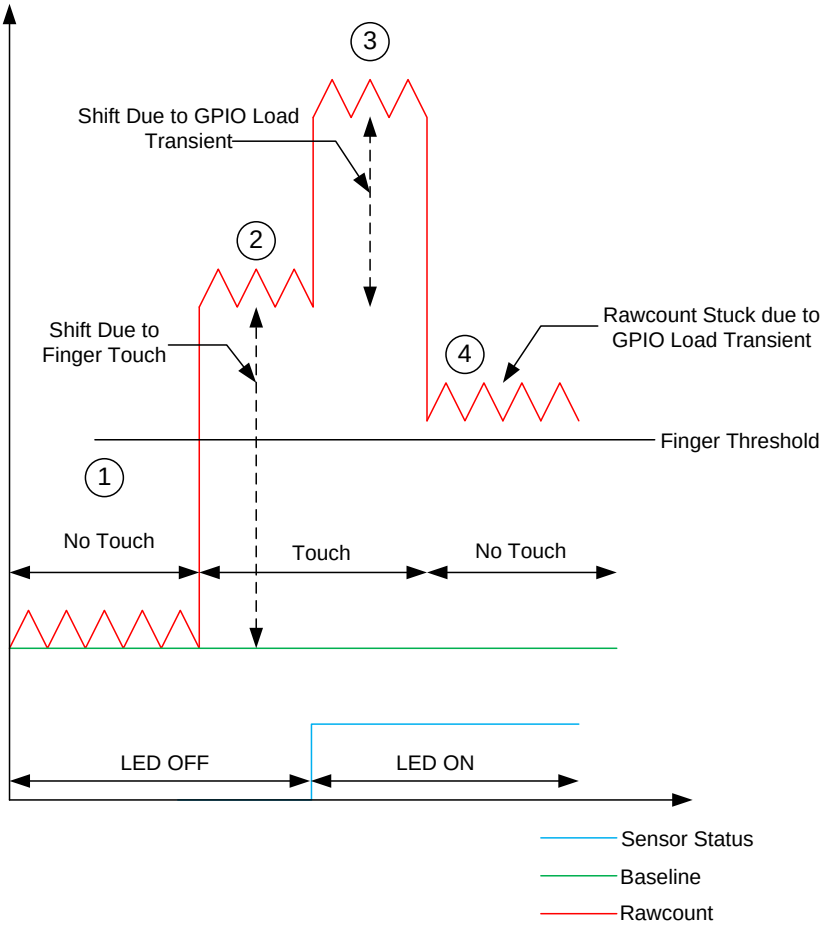


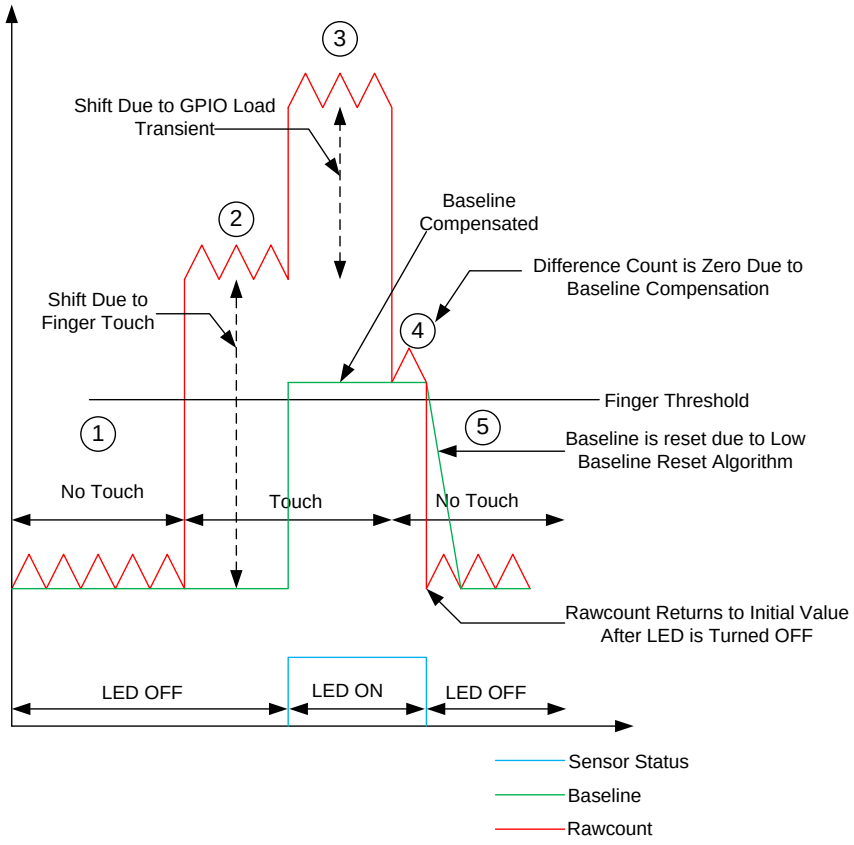
図 5-6 は、センサー ベースラインの補正により誤トリガーがなくなる状態を示します。

1. インスタント 1 では、センサー上の指タッチがなく、LED がオフです。

2. インスタント 2 では、センサーには指の存在があり、raw カウント (差分カウント) の変化は指の閾値より高いです。
3. 差分カウントの変化は指の閾値より高いため、インスタント 3 では、LED はオンになります。
4. LED がオンにされた時、GPIO 負荷瞬時変化ノイズは以下の通り計算されます。ここで、 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)です

   GPIO 負荷瞬時変化によるノイズ 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のリセット 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時の 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 with CapSense」を参照してください。

6. 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 レジスタ内のスリープビット (ビット 3) に「1」を書き込んでスリープ モードに入ります。これは、M8C_スリープマクロを呼び出すことにより実行されます。スリープモードでは。中央 CPU は停止し、内部メインオシレータ(IMO)は無効化され、バンドギャップ基準電圧は引き下げられ、フラッシュメモリモードは無効化されます。供給電圧モニターと 32kHz 内部オシレータ回路のみがオペレーション状態です。スリープモード以外の省電力技術を以下に示します:

- CapSense (PSoC) アナログ ブロック リファレンスを無効にします
- CT と SC ブロックを無効にします
- CapSense (PSoC) アナログ出力バッファを無効にします
- 駆動モードをアナログ HI-Z に設定

スリープモードには、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 は、必要な時にはスリープを解除し、またユーザーはデバイスがスリープモードで追加処理を行うことを認識しておく必要があります。

6.1 その他の省電力技術

スリープモード以外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。以下のセクションでそれぞれの技術を詳細に説明します。

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

6.1.1 駆動モードをアナログ HI-Z に設定

CapSense コントローラーの駆動モードの状態が消費電力に影響を及ぼすことがあります。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。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、CapSense コントローラーの駆動モードの構造のために、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、ピンに関わるその前の値と反対になります。従って、ピンが HIGH で駆動されていれば、一時駆動 モードは、抵抗プルダウンになります。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マニュアルでセットします。駆動モードを制御するレジスタは、PRTxDM0、PRTxDM1 と PRTxDM2 の 3 つです。各レジスタから 1 ビットをピンに割り当てます。従って、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、同一の 3 つのレジスタ書き込みにより、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 の適正なビットパターンは 110b です。次のコードを使用して、最初に抵抗プルダウンに進み、ポートゼロ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
```

6.1.2 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。この順序では、割り込みは無効化され、アナログ回路はオフになり、すべてのドライブ モードはアナログ HI-Z に設定され、割り込みが再度有効にされます。

```
void PSoC_Sleep(void){
    M8C_DisableGInt;
    ARF_CR &= 0xf8; // analog blocks Off
    ABF_CR0 &= 0xc3; // analog buffer off
    PRT0DM0 = 0x00; // port 0 drives
    PRT0DM1 = 0xff;
    PRT0DM2 = 0xff;
    PRT1DM0 = 0x00; // port 1 drives
    PRT1DM1 = 0xff;
    PRT1DM2 = 0xff;
    PRT2DM0 = 0x00; // port 2 drives
    PRT2DM1 = 0xff;
    PRT2DM2 = 0xff;
    M8C_EnableGInt;
    M8C_Sleep;
}
```

6.1.3 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から解除することができます。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 コントローラーを有効にする場合と、スリープ中にデジタル通信をしようとする場合に、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、次のセクションで詳しく説明されています。

6.1.4 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 レジスタのスリープビットへの書き込み後に実行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 コントローラーによりスリープビットが設定されありません。その代わり、割り込みが可能となり、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が全体的に無効化され、その後スリープビットを書き込む直前に割り込みを再度イネーブルにしなければなりません。

6.1.5 グローバル割り込みのイネーブル

割り込みで CapSense コントローラーをウェイクするには、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、割り込みによってスリープから解除する場合の必要条件は INT_MSKx レジスタ内の適切な割り込みマスクを使用します。グローバル割り込みが無効にされている場合、CapSense コントローラーをウェイクする ISR は実行されませんが、CapSense コントローラーはまだスリープ モードを終了します。

この場合、ISR を可能にするには、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;
```

6.2 ウェイクアップ後の実行シーケンス

リセットで CapSense コントローラーをスリープ モードから復帰させる場合、ブートコードの最初から実行されます。CapSense コントローラーを割り込みサービス ルーチンでウェイクする場合、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になる前にプリフェッチされ

ているからです。そのため、グローバル割り込みがディスエーブルされた場合、命令実行はスリープが開始される前に戻って継続されます。

6.2.1 PLL モードのイネーブル

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 周波数を最低の 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、スリープ モードを終了した後、10ms 待て、デバイスが正常に動作することを確認する必要があります。このことは、スリープモードと PLL を使用するにはソフトウェアは 3MHz で実行可能なもので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 を減速することができます。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 を備えた各デバイス ファミリに応じて CPU 速度が異なることを意味します。通常、SYSCLK は 24 MHz です。

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

6.2.2 グローバル割り込みイネーブル化の実行

スリープビット書き込み命令境界上での割り込みの回避。これにより、スリープ コマンドが割り込み命令からの戻り (reti) で実行される場合、スリープ モード移行の準備は不要になります。これを防止するために、割り込みはスリープ準備前に、一時的に無効化され、スリープに入る前に再度イネーブルにします。グローバル割り込みの命令のタイミングにより、次の命令 (この場合にスリープビットを設定する命令) の間は割り込みは発生しません。

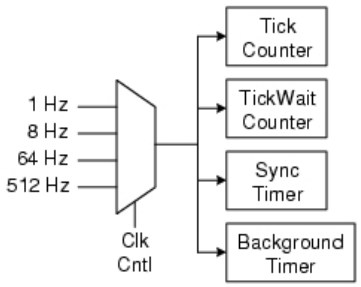
6.2.3 スリープモードを持った I²C スレーブ

I²C スレーブをスリープモードで使用するには、いくつかの混乱があります。スリープの間は、IMO と CPU が停止しているので、CapSense コントローラー内には何の処理もありません。問題は I²C アドレスで発生します。I²C スタート条件が特定のアドレスに送信された場合、CapSense コントローラーがアドレスを処理できないために、NAK で応答します。代表的な回避策は、I²C バスのクロック、または、データライン上に立下りエッジ割り込みを作る方法です。こうすればマスターが CapSense コントローラーをウェイクアップするダミーのスタート条件を送信することができます。ウェイクアップと I²C アドレスが処理可能となるまでの間に、若干のタイムラグがあるので、マスターは次の送信を最長 200µs 遅らせるか、または、ACK を受信するまで送信を継続する必要があります。このソリューションには、CapSense コントローラーがあらゆる I²C 立ち下がりエッジトラフィックでウェイクアップし、それにより合計アクティブ時間が増加してスリープ電流が高くなるという点で第 2 の問題があります。もう 1 つのソリューションでは、3 番目の GPIO ピンを使用して CapSense コントローラーをウェイクアップし、適切な遅延時間の後で初回の START 条件を送信します。

6.2.4 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になっている間に使用され、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は、常に電源が切られない内部低速オシレータをコピーします。タイマーは 1Hz、8Hz、64Hz と 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。この 1 例として、定期的にウェイクアップし、センサーをスキャンする方法があります。スリープタイマ・ユーザーモジュールは別の追加的機能を生成させるのにスリープタイマを使用します。この機能は、定期的な割り込みを発生するバックグランド ティック カウンター、ループ プログラム用遅延機能、設定可能なダウン カウンター、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 ダイヤグラムを図 6-1 に示します。

図 6-1. スリープ タイマー ユーザー モジュール ブロック図



7. リソース

7.1 ウェブサイト

サイプレスの CapSense コントローラー ウェブサイトにアクセスして、本書で説明したすべての参照資料を参照できます。

CapSense CY8C20xx6A/H/AS シリーズデバイスの様々な技術資料を CY8C20xx6A/H ウェブページで見ることができます。

7.2 データシート

CapSense CY8C20XX6A/H/AS シリーズデバイスのデータシートは www.cypress.com で入手可能です。

- CY8C20x36A、CY8C20x46A、CY8C20x66A、CY8C20x96A、CY8C20x46AS および CY8C20x66AS
- CY8C20336H、CY8C20446H

7.3 テクニカル リファレンス マニュアル

サイプレスは、以下の技術的リファレンスマニュアルを作成し、最高レベルのアーキテクチャダイヤグラム、レジスタとタイミングダイヤグラムなどの、CapSense コントローラー機能情報を迅速・容易にアクセスできるようにしました。

- PSoC® CY8C20x66 、CY8C20x66A, 、CY8C20x46/96 、CY8C20x46A/96A 、CY8C20x36 、CY8C20x36A Technical Reference Manual (TRM)

7.4 開発キット

7.4.1 ユニバーサル CapSense コントローラー キット

ユニバーサル CapSense コントローラー キットは設計済みのコントロール回路およびプラグインハードウェアを特長とし、プロトタイピングとデバッグが簡単になりました。チューニングとデータ取得用にプログラミングと I^2C-USB ブリッジ ハードウェアが含まれています。

- CY3280-20xx6 汎用 CapSense コントローラー

7.4.2 ユニバーサル CapSense モジュール基板

7.4.2.1 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 から成ります。このモジュールは、あらゆる CY3280 ユニバーサル CapSense コントローラー基板と接続します。

7.4.2.2 マトリクス ボタン モジュール基板

CY3280-BMM マトリクス ボタン モジュールは、4×4 マトリ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) 。このモジュールはあらゆる CY3280 ユニバーサル CapSense コントローラー基板と接続します。

7.4.2.3 リニア スライダー モジュール基板

CY3280-SLM リニア スライダー モジュールは 5 個の CapSense ボタン、1 個のリニア スライダー (10 個のセンサー付) および 5 個の LED から成ります。このモジュールはあらゆる CY3280 ユニバーサル CapSense コントローラーボードと接続します。

7.4.2.4 ラジアル スライダー モジュール基板

CY3280-SRM ラジアル スライダー モジュールは 4 個の CapSense ボタン、1 個のラジアル スライダー (10 個のセンサー付) および 4 個の LED から成ります。このモジュールは、あらゆる CY3280 ユニバーサル CapSense コントローラー基板と接続します。

7.4.2.5 ユニバーサル CapSense プロトタイプ モジュール

CY3280-BBM ユニバーサル CapSense プロトタイピング モジュールで、付属コントローラー基板上の 44 ピン コネクタに接続されたすべての信号へのアクセスが可能です。プロトタイプ モジュール基板はユニバーサル CapSense コントローラー基板と併用して、その他の専用のユニバーサル CapSense モジュール基板に備えていない追加機能を実行することができます。

7.4.3 インサーキットエミュレーション (ICE) キット

ICE ポッドにより、CY3215-DK イン - サーキットエミュレータとプロトタイプシステムや、PCB 内の、ターゲットの PSoC デバイスとのパッケージ専用ポッドフィートを使用し、フレキシブル ケーブルを通した相互接続が可能になります。次のポッドを使用することができます。

- CY8C20236/46A CapSense PSoC デバイス デバッグのための、CY3250-20246QFN イン - サーキット エミュレーション(ICE) ポッド キット
- CY8C20336/346A CapSense PSoC デバイス デバッグのための CY3250-20346QFN イン - サーキット エミュレーション(ICE) ポッド キット
- CY8C20636/646/666A CapSense PSoC デバイス デバッグのための CY3250-20666QFN イン - サーキット エミュレーション (ICE)ポッド キット
- CY8C20536/546/566A CapSense PSoC デバイス デバッグのための CY3250-20566 イン - サーキット エミュレーション(ICE) ポッド キット

7.5 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。これは、PCB 設計プロセスを迅速に完了するための参照用として使用されます。

- CY8C20466A の I^2C ヘッダーによるボタン デザイン
- CY8C20466A の I^2C ヘッダーによるボタンとスライダー デザイン

注: ボード ファイルは (回路図、レイアウトおよびガーバー ファイル) は、本文書の待ち受けページに含まれます。

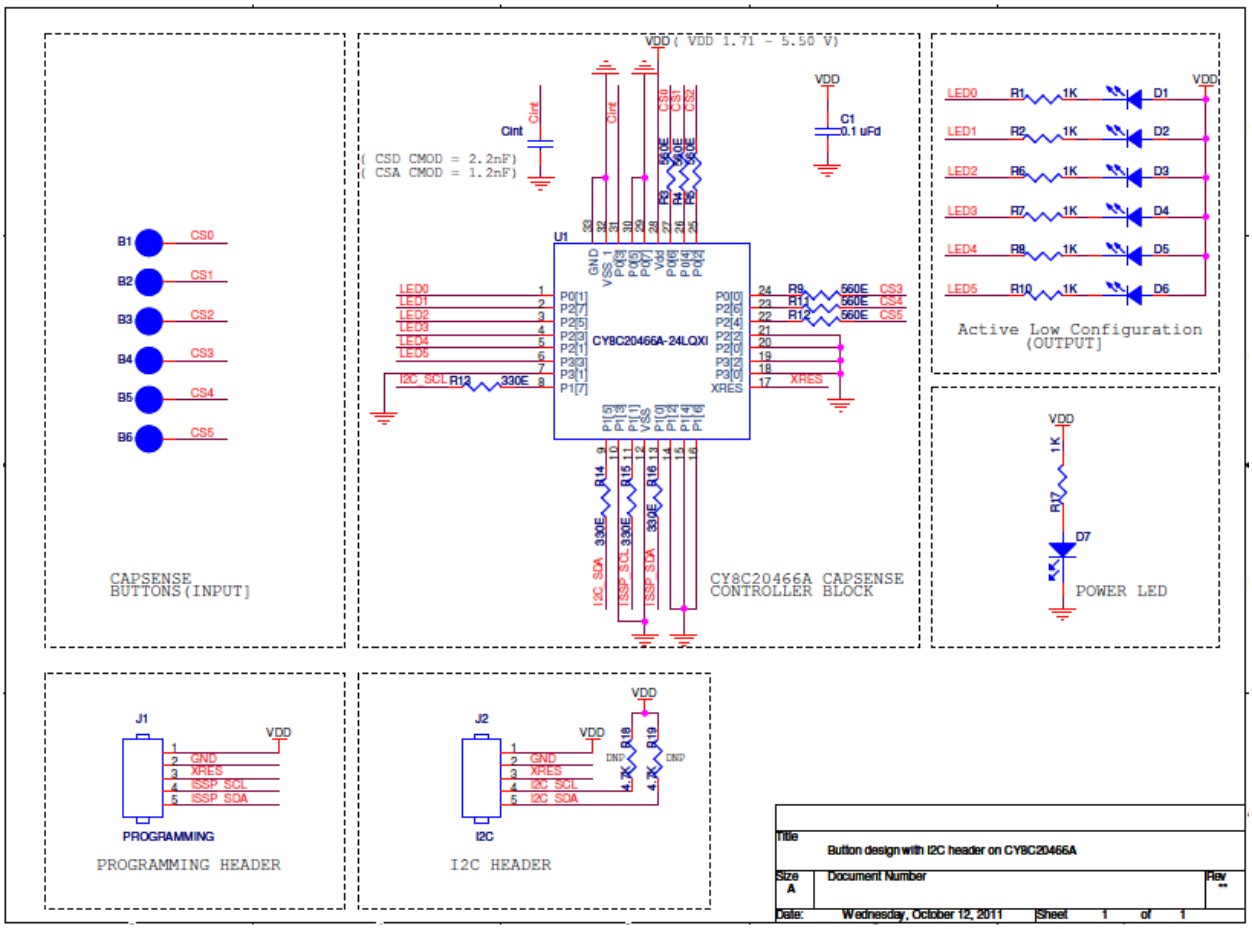
図 7-1 と図 7-2 に基板回路図を示します。

以下の回路図は、次のコンポーネントをサポートするために設計されます。

- 6 個の CapSense センサー。センサーは、CY8C20466A-24LQXI デバイスの P0[6]、P0[4]、P0[2]、P0[0]、P2[6]および P2[4]ピンに割り当てられています。

- 6 個の GPO は、CY8C20466A-24LQXI の P0[1]、P2[1]、P2[3]、P2[5]、P2[7]および P3[3]に接続され、D1、 D2、D3、D4、D5 および D6 LED を駆動します。
- プログラム ヘッダーJ1 による、CY8C20466A-24LQXI 用プログラム作成します。
- I²C の、I²C ヘッダーJ2 による CY8C20466A-24LQXI との通信

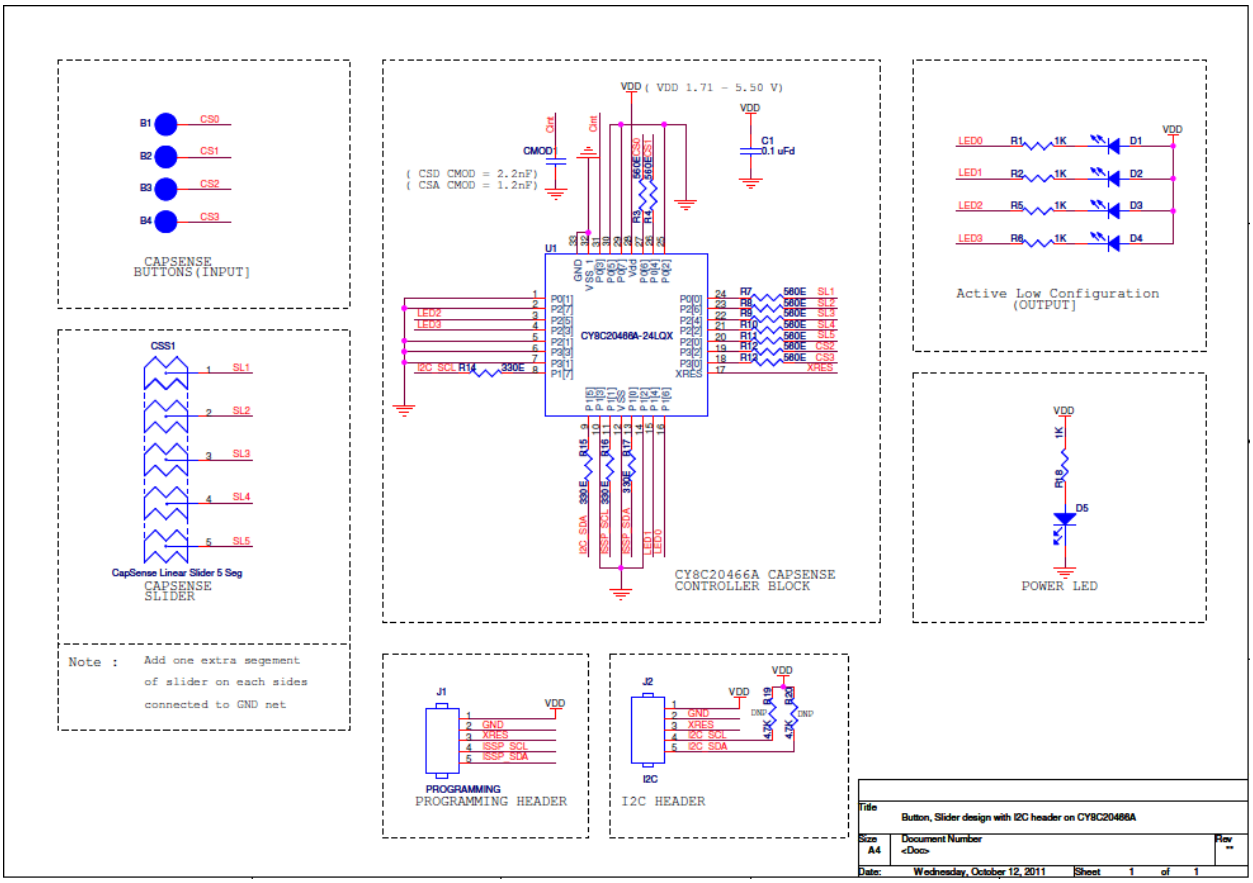
図 7-1. CY8C20466A - 基板回路図上の I²C ヘッダーによるボタン デザイン



以下の回路図は、次のコンポーネントをサポートするために設計されます。

- 4 個の CapSense センサー。センサーは CY8C20466A-24LQXI デバイスの P0[6]、P0[4]、P3[2]および P3[0]ピンに割り当てられています。
- 5 個のセグメントを持つリニア スライダー。セグメントは、CY8C20466A-24LQXI デバイスの P0[0]、P2[6]、P2[4]、P2[2]および P2[0]ピンに割り当てられています。
- 4 個の GPO は、CY8C20466A-24LQXI の P1[6]、P1[4]、P2[5]および P2[7]ピンに割り当てられ、D1、D2、D3 および D4 LED を駆動します。
- プログラム ヘッダーJ1 による、CY8C20466A-24LQXL 用プログラム作成。
- I²C の、I²C ヘッダーJ2 による CY8C20466A-24LQXI との通信

図 7-2. CY8C20466A の I²C ヘッダーによるボタンとスライダー デザイン



7.6 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ム アプリケーションです。 PSoC Designer と PSoC Creator と併用して PSoC デバイスに様々なデザインをプログラムすることが可能です。

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 API を備えたハードウェア層が含まれます。PSoC Programmer 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 コードと共に説明されています。

7.7 CapSense データ表示ツール

常に、CapSense の設計中に、調整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) をモニタリングすることが必要です。

CapSense データ表示とログに適切なツールの識別と使用の有用な情報は、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。

7.8 PSoC Designer

サイプレスは、専用の統合設計環境である PSoC Designer を提供しています。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。これらのアプリケーション (CapSense を含む) は、完全に特性化されたアナログ デジタル機能のライブラリを使用し、ドラッグ アンド ドロップ デザイン環境で開発されています。PSoC Designer には内蔵の C コンパイラと組み込みプログラマが含まれています。複雑な設計用には専用コンパイラがあります。

7.9 コード例

サイプレスは、設計を速やかに完了するために大量のコード実例を提供しています。

- 「CapSense コントローラー コード例」デザイン ガイド
- CY8C20xx6A の EzI2C スレーブによる、CSD ソフトウェア フィルター

7.10 デザイン サポート

サイプレスには様々なデザイン サポート チャネルがあり、お客様の CapSense ソリューション適用を成功にしています。

- Knowledge Base Articles – 製品グループごとの技術記事を閲覧、または、様々なトピックを検索します。
- CapSense Application Notes – 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 ノートを参照してください。
- White Papers – 高度な静電容量タッチ インターフェースに関するトピック。
- Cypress Developer Community – サイプレス技術コミュニティに参加し、情報交換可能
- CapSense Product Selector Guide – サイプレスの CapSense 製品ラインの完全な製品群
- Video Library – チュートリアル ビデオで素早く学習できます。
- Quality and Reliability ーサイプレスは顧客満足を完全に保証します。当社の品質ウェブサイトでは、信頼性および品質レポートをご覧になることが可能です。
- Technical Support – 世界レベルのテクニカル サポートがオンラインで利用可能です。

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性能のために、センス パラメーターを自動的にセットし、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時の 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リファレンス点を提供します。

ボタンまたはボタン ウィジェット

関連したセンサーを持って、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「タッチ有り」または「タッチ無し」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分です。差分が負数であるか、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。

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 のファミリを含んでいます。

重心または重心位置

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、静電容量センスの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、センス 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除去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。

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

プリント基板、ITO または FPCB 上のパッドまたは層などの導電材料です。電極は CapSense デバイス上のポート ピンに接続し、CapSense センサーとして使用されるか、または CapSense 機能に関する特定の信号を駆動するために使用されます。

指の閾値

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。

連結センサー

複数のセンサーを連動して、単一のセンサーとしてスキャンする方法です。近接センス用のセンサーの領域を増やし、電力消費量を減少するために用いられます。

システムが低消費電力モードにある場合の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に、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る場合、システムはアクティブ モードに移行し、全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ています。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りサポートされます。

ガード センサー

ボタン センサーと同様に PCB 上の全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することができます。

ハッチ フィル／ハッチ グランド／ハッチド グランド

静電容量センスの PCB を設計する時、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、ラインの幅および 2 本のライン間の間隔は充填率を指定します。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の閾値と一緒に使用されるパラメーターです。指の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ス 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。

マトリクス ボタン

マトリクス状で配置された 2 つ以上のセンサーを含んで、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 個の交差部を監視することができます。

CSD センス方式 (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。

変調コンデンサー (CMOD)

自己容量センス 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ーです。

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 コンデンサーが引き出した平均電流に等しいです。

相互静電容量

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 パラメーターと併用されます。

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通りです: 電源投入時にセンサーに指が触れる時、センサーの近く配置される金属の物体を除去する時、耐液性のある CapSense 製品の水分を除去する時、および他の急激な環境変化がある時です。

ノイズ (CapSense ノイズ)

センサーがオフ状態 (タッチなし) 時にピーク 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。

寄生静電容量 (C_P)

寄生容量は PCB 配線、センサー 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
物理的な接触無しで近接オブジェクトの存在性を検知できるセンサー。

ラジアル スライダー
指の物理位置を検出する特定の円形に配置される 1 つ以上のセンサーが含むウィジェット。

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必要な時間です。

自己静電容量
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、センス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 センス方式用のスイッチト コンデンサー回路のフロント 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
センサーがシステム故障によって誤ったタッチ状態の報告から防ぎ、または金属物体がセンサーの近くに連続的に存在する場合の設定。

常に、センサーの自動リセットが有効化されている場合、差分カウントがノイズ閾値を超えてもベースライン値が更新されます。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。センサーの自動リセットが無効化されていると、ベースライン値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。

センサー連結
連結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分による誤タッチを防止。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。駆動シールドを参照してください。

シールド タンク コンデンサー (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ー (C_{SH} タンク コンデンサー) です。

信号 (CapSense 信号)
差分カウントも信号と呼ばれる。差分カウントを参照してください。

信号対雑音比 (SNR)
タッチした時のセンサーの信号とタッチしない時のセンサーのノイズ信号との比率。

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なり、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。

調整

CapSense の動作に必要な様々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

CapSense および ADC 動作用の PSoC 内のプログラマブルな電圧リファレンス ブロック。

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、マトリクス ボタンおよびタッチパッドはサポートしたウィジェットです。

改版履歴

ドキュメント改版履歴

文書名: AN65973 - CY8C20xx6A/H/AS CapSense® デザイン ガイド 文書番号: 002-12923			
版	**発行日**	**改版者**	**変更内容**
**	07/07/2016	HZEN	これは英語版 001-65973 Rev. *H を翻訳した日本語版 002-12923 Rev. **です。
*A	03/12/2019	SSAS	これは英語版 001-65973 Rev. *J を翻訳した日本語版 002-12923 Rev. *A です。