AN76000 – CY8CMBR2110

CapSense®デザイン ガイド

文書番号: 001-88933 Rev. *C

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 6

1.1 概要 6
1.2 サイプレスの CapSense の文書体系 6
1.3 CY8CMBR2110 CapSense Express デバイスの機能 8
1.4 本書の表記法 9
1.5 略語 10

2. CapSense 技術 11

2.1 CapSense の原理 11
2.2 静電容量の検出方法 12
2.2.1 CapSense シグマデルタ (CSD) 12
2.3 SmartSense 自動チューニング 14
2.3.1 プロセスのばらつき 14
2.3.2 設計サイクル時間の短縮 14

3. CapSense 回路デザイン 15

3.1 CapSense コントローラー ピン 15
3.1.1 CapSense のボタン (CSx) 15
3.1.2 汎用出力 (GPOx) 15
3.1.3 変調コンデンサ (CMOD) 16
3.1.4 ブザー信号出力 (BuzzerOut0、BuzzerOut1) 16
3.1.5 ホスト制御 GPO (HostControlGPO0、HostControlGPO1) 18
3.1.6 Attention/Sleep 18
3.2 CapSense コントローラーの設定 20
3.2.1 ボタン自動リセット (ARST) 20
3.2.2 ノイズ耐性 20
3.2.3 自動閾値 20
3.2.4 トグル オン／オフ 21
3.2.5 隣接センサー抑制 (FSS) 21
3.2.6 LED オン時間 22
3.2.7 LED エフェクトのパラメーター 22
3.2.8 ラッチ状態の読み出し 27
3.2.9 アナログ電圧サポート 28
3.2.10 感度制御 29
3.2.11 デバウンス制御 29
3.2.12 システム診断機能 30

3.2.13 ボタン スキャン速度 31
3.2.14 I²C 通信 32
3.3 デザイン ツールボックス 33
3.3.1 全般的なレイアウト ガイドライン 33
3.3.2 レイアウト エスティメータ 34
3.3.3 CP、消費電力、応答時間計算器 35
3.3.4 デザイン検証 37
3.4 CY8CMBR2110 設定 38
3.4.1 EZ-Click カスタマイザ ツール 40
3.4.2 ホスト プロセッサによるデバイス設定 41
3.4.3 サードパーティ プログラマ 47
3.5 CY8CMBR2110 のリセット 47
3.5.1 ハードウェア リセット 47
3.5.2 ソフトウェア リセット 47

4. 電気的／機械的設計時の考慮事項 48

4.1 オーバーレイの選択 48
4.2 ESD 保護 49
4.2.1 防止 49
4.2.2 リダイレクト 49
4.2.3 クランプ 49
4.3 EMC (電磁環境適合性) の注意点 50
4.3.1 放射性干渉 50
4.3.2 伝導耐性およびエミッション 50
4.4 PCB レイアウト ガイドライン 50

5. 低消費電力設計上の考慮事項 51

5.1 システム設計の推奨事項 51
5.2 平均消費電力の計算 51
5.2.1 ボタン スキャン速度 (TR) 52
5.2.2 スキャン時間 (TS) 53
5.2.3 非タッチ状態での平均電流 (IAVE_NT) 54
5.2.4 タッチ状態での平均電流 (IAVE_T) 54
5.2.5 アクティブ時間の割合 (P) 54
5.2.6 平均使用電流 (IAVE_U) 55
5.2.7 平均電流 (IAVE) 55
5.2.8 平均電力 (PAVE) 55
5.2.9 計算例 55
5.3 スリープ モード 56
5.3.1 低消費電力スリープ モード 56
5.3.2 ディープ スリープ モード 57

6. リソース 58

6.1 ウェブサイト 58
6.2 データシート 58

6.3 デザイン ツールボックス 58
6.4 EZ-Click™ カスタマイザ ツール 58
6.5 デザイン サポート 58

7. 付録 59

7.1 参考回路 59
7.1.1 回路図 1: 10 のボタンと 10 の GPO 59
7.1.2 回路図 2: アナログ電圧出力付きの 8 個のボタン 61
7.2 CY8CMBR2110 コンフィグレーション用の API 63
7.2.1 高レベル API 63
7.2.2 低レベル API 85

用語集 86

改訂履歴 92

改訂履歴 92

1. はじめに

1.1 概要

このドキュメントはサイプレスの CapSense® Express CY8CMBR2110 デバイスを用いて静電容量センサー機能を実装する方法を説明しています。このガイドでは以下の項目について紹介します。

- CY8CMBR2110 の特長
- CapSense の動作原理
- CY8CMBR2110 デバイスの設定オプション
- CY8CMBR2110 デザイン ツールボックスの使用
- CY8CMBR2110 システムの電気的および機械的の設計考慮事項
- CY8CMBR2110 の低消費電力設計の考慮事項
- ユーザー システムに CapSense 組込み設計するための追加のリソースとサポート

1.2 サイプレスの CapSense の文書体系

図 1-1 と表 1-1 に、CapSense の文書体系を示します。これらのリソースにより、CapSense を使った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することができます。図 1-1 に、静電容量センシングを使用した典型的な製品設計サイクルを示します。このドキュメントは、緑色でハイライト表示した項目を含みます。表 1-1 は、図 1-1 の中に番号を付けられた各タスクに関する関連ドキュメントへのリンクを提示しています。

図 1-1. 標準的な CapSense 製品の設計フロー

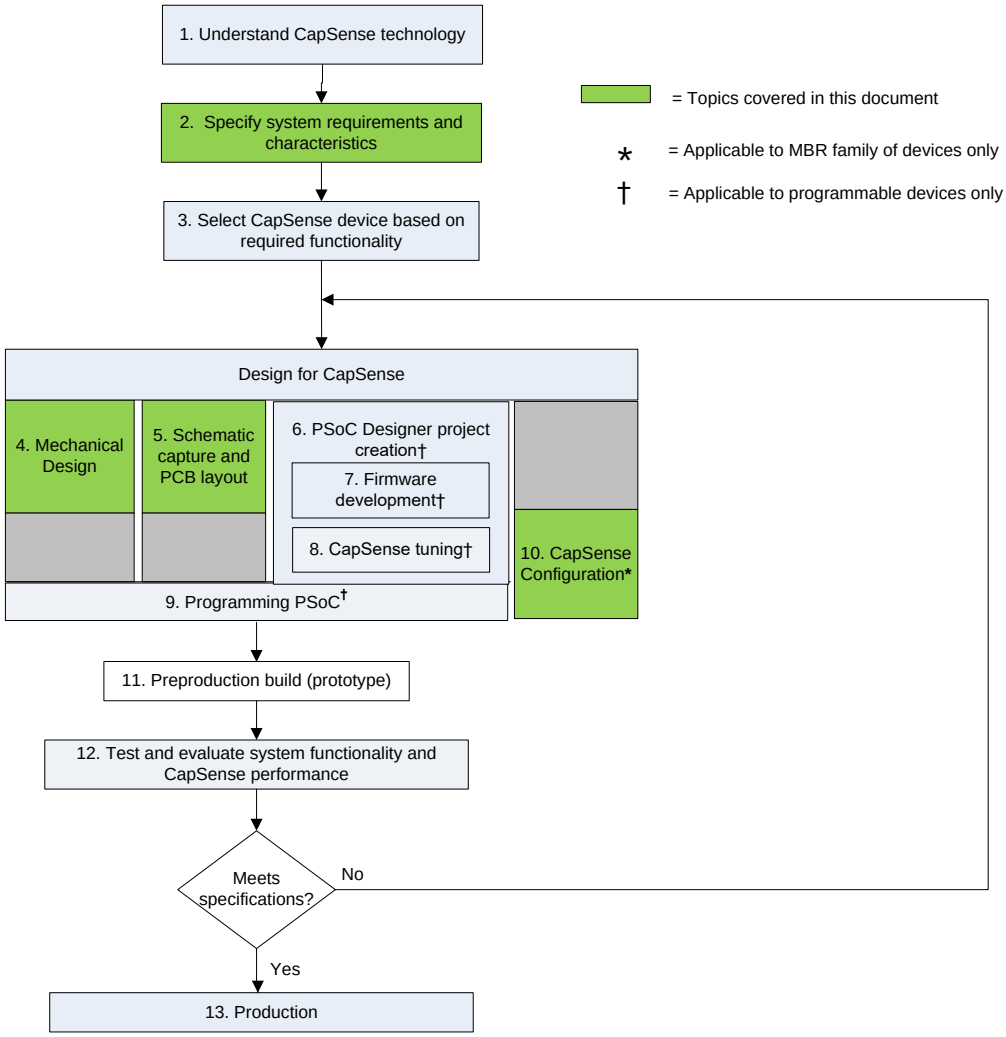


表 1-1.図 1-1 中に番号付けされた設計タスクをサポートしているサイプレスのドキュメント

図 1-1 の設計タスクの番号	サイプレス CapSense のサポートドキュメント
1	「Getting Started with CapSense」
2	「CY8CMBR2110 Device Datasheet」
3	「Getting Started with CapSense」
4	本書
5	本書
6	CY8CMBR2110 は対象外
7	CY8CMBR2110 は対象外
8	CY8CMBR2110 は対象外
9	CY8CMBR2110 は対象外
10	本書

1.3 CY8CMBR2110 CapSense Express デバイスの機能

サイプレスの低消費電力 CapSense コントローラーは、静電容量タッチ センシングをユーザー インターフェースに簡単に追加することができます。デバイスの特長は以下のとおりです。

- レジスタ設定可能 CapSense コントローラー
  - ファームウェアまたはデバイスのプログラミングが不要
  - I^2C プロトコルで設定可能な 10 個のボタン ソリューション
  - 10 個の汎用出力 (GPO)
  - GPO は CapSense ボタンに接続されている
  - GPO が直接 LED を駆動することをサポート
- SmartSense™ 自動チューニング
  - システム、製造および環境の変化に対応して連続的に補正を行う CapSense アルゴリズム
  - CapSense パラメーターを動的に設定
  - システムの手動調整が不要
  - 寄生容量 (C_P) の広い範囲 (5～40pF)
- 高度な機能
  - 隣接センサー抑制 (FSS)
    - 間隔が狭いボタンからの信号を区別
  - ユーザー設定可能な LED エフェクト
    - システム パワーオン時
    - ボタン タッチ時
    - ボタンを離した後の LED オン時間
    - スタンバイ モードの LED 輝度
  - ブザー信号出力
  - アナログ電圧出力
    - 外部抵抗ブリッジを使用
  - 任意の CapSense ボタン状態の変更を指示するホストへのアテンション ライン割込み
  - I^2C インターフェースを介した CapSense パフォーマンス データ
    - 生産ラインのテストやシステム デバッグを簡素化
- ノイズ耐性
  - 外部の放射および伝導ノイズに対して、優れたノイズ耐性を備えた特別な設計
  - 低放射性ノイズ放出
- システム診断機能
  - ボタン短絡
  - 変調コンデンサ (C_{MOD}) の不適切な値
  - 寄生容量 (C_P) の値が範囲外
- EZ-Click™ カスタマイザ ツール

- □ シンプルなグラフィックによる設定
- □ すべての機能を動的に設定
- □ 各設定は将来の再利用のために保存可能

- I²C インターフェース
  - □ クロック ストレッチなし
  - □ 最大 100kHz の速度をサポート
- 広範な動作電圧
  - □ 1.71～5.5V
  - □ 安定化および非安定化の両バッテリ アプリケーションに最適
- 低消費電力
  - □ ボタン当たりの平均消費電流が 23µA[1]
  - □ ディープ スリープ電流: 100nA
- 産業用途向け温度範囲: -40°C～+85°C
- 32 ピン QFN パッケージ (5x5x0.6mm)

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 コードを示します。 C:\ ...cd\icc\
Italics フォント	ファイル名や参考資料を示します。 「*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 コマンドを示します。 [Enter]または[Ctrl] [C]
File > Open	メニュー パスを示します。 File > Open > New Project
太字	操作手順でのコマンド、メニューパス、アイコン名を示します。 **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	式を示します。 2+2=4
灰色ボックス内のテキスト	製品の注意点や製品固有の機能を示します。

[1]ボタンが 4 個使用され、1 時間当たり 180 回のボタン タッチ、ボタン タッチの平均時間が 1000ms、ブザーが無効、ボタン タッチ LED エフェクトが無効、10pF< (すべてのボタンの C_P) <20pF、ボタン スキャン速度が 541ms、消費電力が最適化され、ノイズ耐性のレベルが「中」、CSx 感度が「中」。

1.5 略語

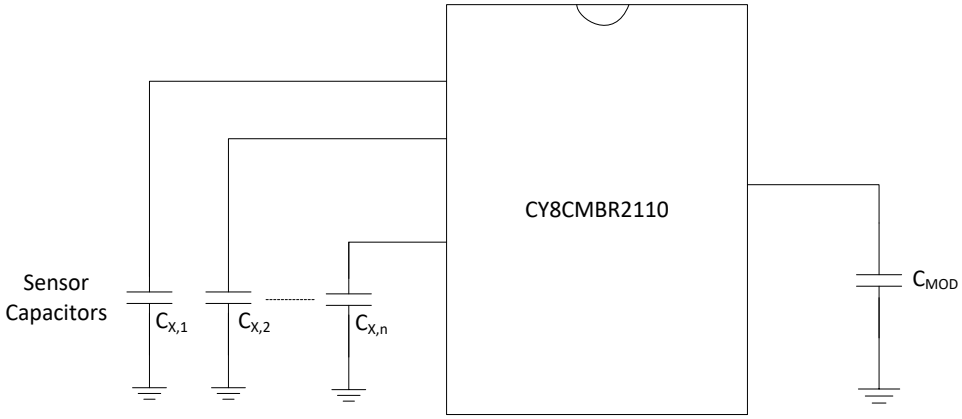
略語	説明
AC	Alternating current (交流電流)
ARST	Auto Reset (自動リセット)
C_F	Finger capacitance (指の静電容量)
C_P	Parasitic capacitance (寄生容量)
CS	CapSense
CSD	CapSense Sigma Delta (CapSense シグマデルタ方式)
EMC	Electromagnetic Compatibility (電磁環境適合性)
ESD	Electrostatic Discharge (静電放電)
FSS	Flanking Sensor Suppression (隣接センサー抑制)
GPO	汎用出力
MSB	Most significant bit (最上位ビット)
LCD	Liquid Crystal Display (液晶ディスプレイ)
LED	Light-Emitting Diode (発光ダイオード)
LSB	Least significant bit (最下位ビット)
PCB	Printed Circuit Board (プリント回路基板)
POR	Power on Reset (パワー オン リセット／電源投入時リセット)
POST	Power on Self-Test (パワー オン セルフテスト／電源投入時自己検査)
RF	Radio Frequency (無線周波数)
SNR	Signal to Noise Ratio (信号対雑音比)
SMPS	Switched Mode Power Supply (スイッチング電源)

2. CapSense 技術

2.1 CapSense の原理

CapSense はタッチ センシング技術で、CapSense コントローラー上の各センサー入力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 ピンの総合静電容量は Cx,1～Cx,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。CY8CMBR2110 デバイスに内蔵された回路が、各 CX 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。変調コンデンサ (CMOD) は、CapSense コントローラーの内部回路で使用されます。CMOD の詳細は「静電容量の検出方法」を参照してください。

図 2-1. CY8CMBR2110 デバイスの CapSense の実装



各センサーの入力ピンは、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。非導電性のオーバーレイが各センサー パッドを覆うために必要であり、製品のタッチ インターフェースを構成しています。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を構成し、その静電容量は以下の式によって与えられます。

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

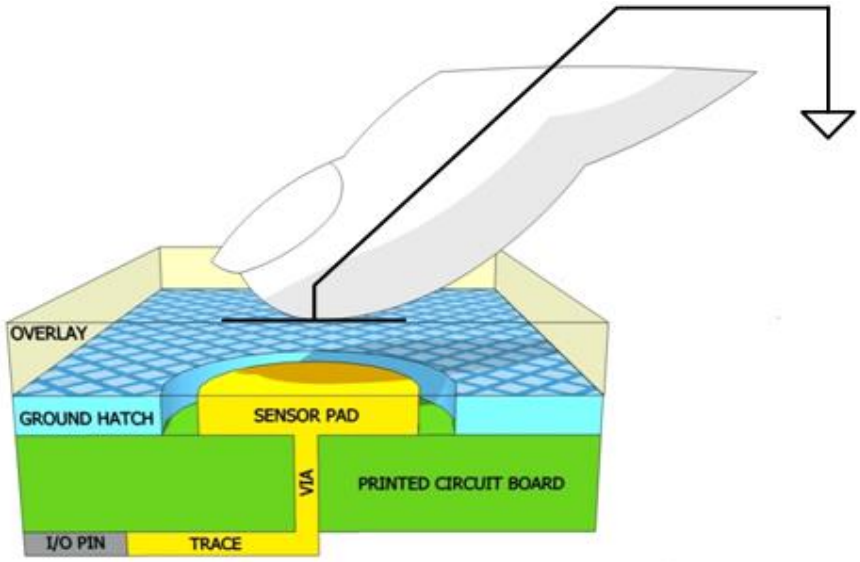
ε_0=真空の誘電率

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. 指でセンサーがアクティブになっているときの典型的な 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タッチしている指は、それ自身とすぐ近くにある他の導体とのあいだに静電結合を引き起こします。この静電結合された領域の影響は通常小さく、無視することができます。

指がオーバーレイにタッチしなくても、センサーの入力ピンには寄生容量 (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、ビアがグランドプレーン、他の配線、製品のシャーシまたは筐体に含まれる金属などの導体と静電結合した組み合わせとして生じます。CapSense コントローラーは、センサー ピンに接続している全ての電気容量 (C_X) を計測します。

指がセンサーに触れていない場合、次の式を使用します。

$$C_X = C_P \quad \text{式 2}$$

指がセンサーに触れた場合は、C_X は C_P と C_F の和になります。

$$C_X = C_P + C_F \quad \text{式 3}$$

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 の範囲ですが、極端な場合は 40pF まで高くなることもあります。C_F は通常 0.1pF～0.4pF の範囲です。

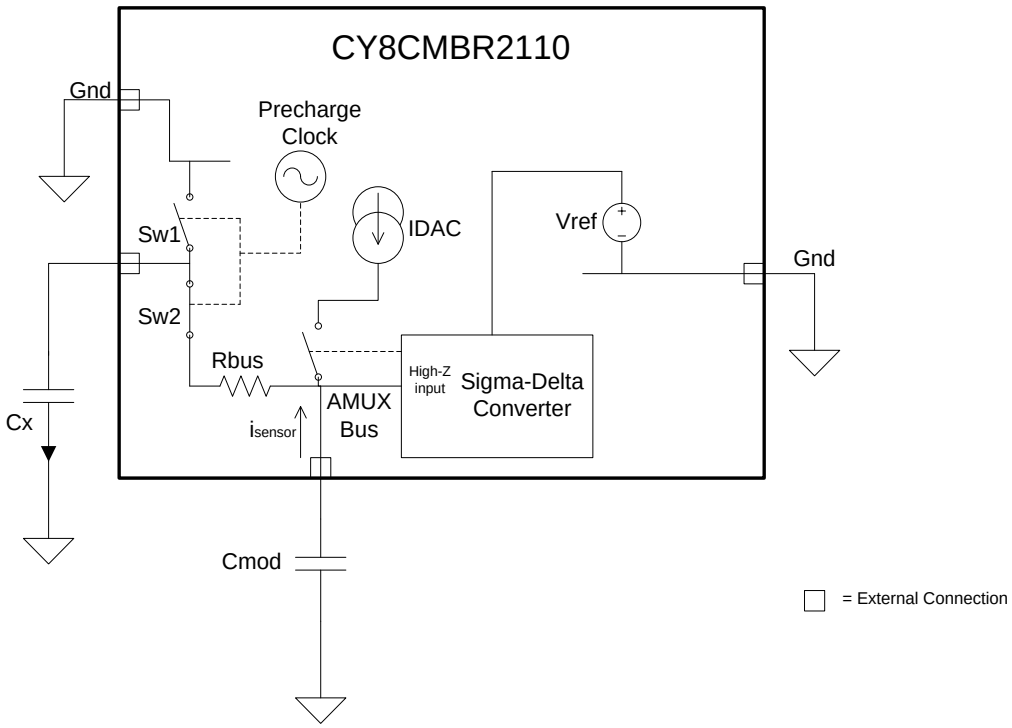
2.2 静電容量の検出方法

CY8CMBR2110 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ために、SmartSense 自動チューニングおよび CapSense シグマ デルタ (CSD) に対応しています。CSD 方法は次の節で説明します。

2.2.1 CapSense シグマデルタ (CSD)

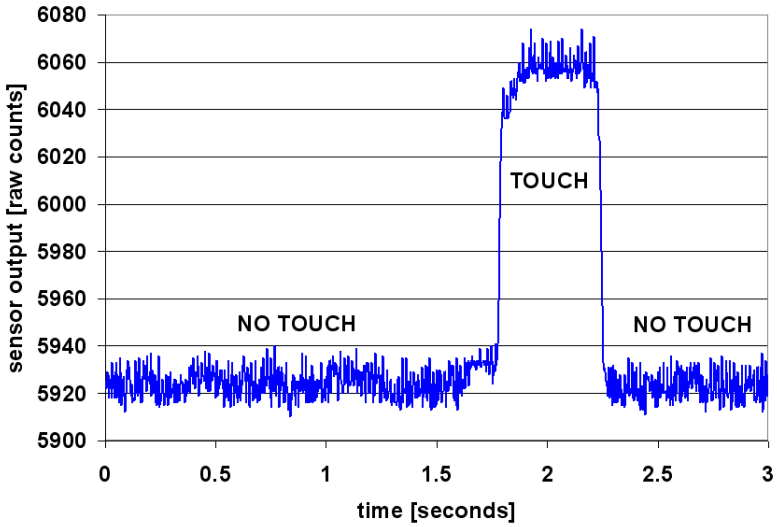
CY8CMBR2110 デバイスの CSD 方式は、図 2-3 に示すように、C_X をスイッチト キャパシタ回路によって組み込まれます。C_X は、重なっていないスイッチ SW1 と SW2 によってグランドおよび AMUX バスへ二者択一的に接続されています。Sw1 および Sw2 は、プリチャージ クロックによって駆動され、AMUX バスから電流 i_{sensor} を出力します。i_{sensor} の大きさは、C_X の大きさに正比例します。シグマデルタ コンバータは、AMUX バス電圧をサンプリングして、定電流源 (IDAC) を制御する変調ビット ストリームを生成します。IDAC は、平均 AMUX バス電圧が Vref 電圧に維持されるように AMUX を充電します。センサーは、CMOD から i_{sensor} を出力します。CMOD は Rbus と一緒にシグマデルタ コンバータの入力部でプリチャージスイッチングの過渡電流を減衰させるローパス フィルタを形成します。

図 2-3. CSD ブロック図



AMUX バス電圧を Vref 電圧に維持するために、シグマデルタ コンバータはビット ストリームのデューティ サイクルを制御することで IDAC 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 スキャン期間に渡ってビット ストリームを格納し、蓄積された結果が C_X に正比例した raw カウントであるデジタル出力となります。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、センサーに指を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 (タッチ) 状態であるかオフ (非タッチ) 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。raw カウント、指閾値および信号対雑音比 (SNR) の詳細は「Getting Started with CapSense」を参照してください。

図 2-4. 指がタッチしたときの CSD の raw カウント



2.3 SmartSense 自動チューニング

タッチ センシングのユーザー 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顧客体験のために重要です。残念ながら、チューニングは反復プロセスのため時間がかかります。一般的な開発サイクルでは、インターフェースは初期の設計段階、システム統合中および増産前にチューニングされます。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素化するために開発されました。これはシステム、製造および環境の変化に対応して連続的に補正を行う CapSense アルゴリズムです。使い易く、試作や製造段階で手動チューニングを無くすことで、設計のサイクル時間を短縮します。SmartSense 自動チューニングは電源投入時に各 CapSense ボタンを自動的に調整し、使用中の最適なボタン操作を維持します。SmartSense 自動チューニングは、プリント基板とオーバーレイの製造ばらつきに適応し、LCD インバータ、AC ライン、スイッチング電源などからのノイズを自動的に調整し取り除きます。

2.3.1 プロセスのばらつき

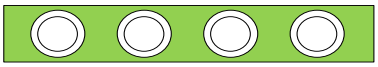
CY8CMBR2110 デバイスの SmartSense 自動チューニングは、5～40pF の範囲内の C_P の値で動作するように設計されています。各ボタンの感度パラメーターは、個々の特性に基づいて自動的に設定されます。ボタンごとに異なる C_P に関係なく一貫性のある応答を維持しているので、このパラメーターは量産時の歩留まりを向上させます。C_P は、プリント基板のレイアウトや配線の長さ、基板製造プロセスのばらつき、複数の購買サプライ チェーンのベンダー間のプリント基板のばらつきによって異なる可能性があります。ボタンの感度は C_P に依存します。C_P 値が高くなると感度が低下し、指タッチ信号の振幅が低下します。C_P の変化により、ボタンが高感度になりすぎたり、感度が十分でなくなったり、または動作しなくなってしまうことがあります。そのとき、ユーザーはシステムを再調整し、場合によってはユーザー インターフェースのサブシステムを再検証する必要があります。SmartSense 自動チューニングはこのような問題を解決します。

SmartSense 自動チューニングにより、プラットフォームの設計が可能になります。例えば、ラップトップ コンピュータ上の容量タッチ センシングのマルチメディア キーを考えてみます。CapSense ボタンの寄生容量は、ラップトップのサイズやキーボード レイアウトに応じて同じプラットフォーム設計でもモデルごとに異なります。この例では、ワイド画面のラップトップ モデルは、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。従って、ワイド画面モデルは各ボタンと CapSense コントローラー間で長い配線となります。その結果、高い C_P 値となります。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、ボタンは各モデル毎に調整する必要があります。SmartSense 自動チューニングにより、「Getting Started with CapSense」のプリント基板レイアウトの節に示されている推奨方法を使用することでプラットフォーム設計を行うことができます。

図 2-5. 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



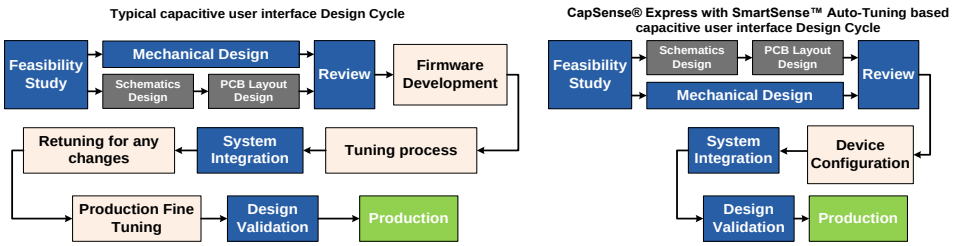
図 2-6. 機能とボタン サイズが同一の 15 インチ モデルのラップトップ マルチメディア キーの設計



2.3.2 設計サイクル時間の短縮

容量ボタン インターフェースを設計するときに、最も時間のかかる作業はファームウェア開発、レイアウト、およびボタンの調整です。一般的なタッチ 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、PCB やボタン PCB レイアウトの機械的寸法に変更があった場合、ボタンを再調整しなければなりません。SmartSense 自動チューニングによる設計は、ファームウェア開発、手動チューニングまたは再チューニングを必要としないため、これらの課題に対応しています。また、SmartSense 自動チューニングは一般的な設計サイクルを加速します。図 2-7 は、一般的なタッチ センシング コントローラーと SmartSense 自動チューニングに基づいたデザインの設計サイクルを比較します。

図 2-7. 典型的な静電容量ユーザー インターフェース設計サイクルの比較

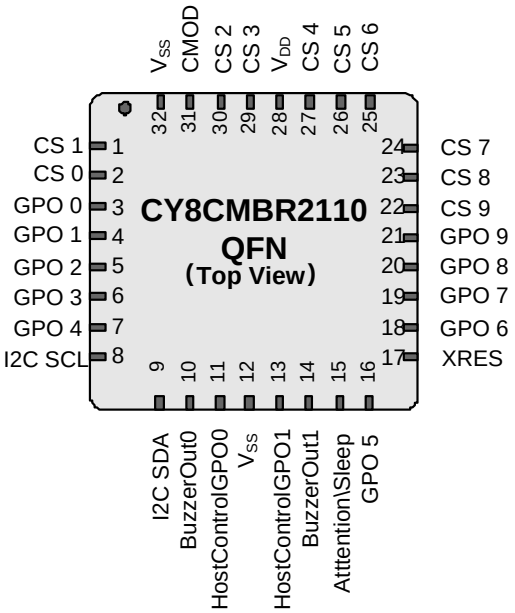


3. CapSense 回路デザイン

サイプレスの CY8CMBR2110 デバイスは、ハードウェアおよび EZ-Click カスタマイザ ツールで I²C インターフェースを介して設定されます。本節では、CapSense コントローラー ピンおよびレジスタの概要と、その設定方法について説明します。

3.1 CapSense コントローラー ピン

図 3-1. CY8CMBR2110 ピン配置図



3.1.1 CapSense のボタン (CSx)

CY8CMBR2110 コントローラーには、CS0～CS9 までの 10 個の静電容量センサー入力があります。各静電容量ボタンは、それら静電容量センサー入力 1 個との接続が必要です。ユーザーは、すべての未使用の CapSense (CSx) 入力ピンを接地しなければなりません。

3.1.2 汎用出力 (GPOx)

CY8CMBR2110 コントローラー上には、GPO0～GPO9 までの 10 個のアクティブ LOW 出力があります。各出力は、対応する静電容量センシング入力の CSx により駆動されます。GPO は直接 LED を駆動したり、または機械的スイッチを置き換えるために使われます。GPO はストロング駆動[2]モードになっています。すべての未使用 GPO ピンは、フローティングでなければなりません。

[2] ピンがストロング駆動モードにある場合、出力が HIGH レベルのとき V_{DD} にプルアップされ、出力が LOW レベルのときグランドにプルダウンされます。

3.1.3 変調コンデンサ (CMOD)

2.2nF (±10%) コンデンサを CMOD ピンに接続します。

3.1.4 ブザー信号出力 (BuzzerOut0、BuzzerOut1)

ブザー信号出力は、CapSenseボタンが触れられたときに音を出すために使用されます。これはオーディオ センサーが使用される設計には役立ちます。ブザー信号出力のために圧電ブザーを使用します。

ブザー信号出力はストロング駆動モード出力です。通常、各出力は全ての CSx ボタンによって駆動されます。ブザーが使われない場合は、BuzzerOut0 および BuzzerOut1 はホスト制御 GPO として使用されます。表 3-2 に様々なブザー設定の例を示します。

ブザー信号出力は 2 つの設定があります。

1. AC 1 ピン ブザー: 図 3-2 に示すように、ブザーはデバイスの BuzzerOut0 ピンに接続されます。指定された周波数とデューティ サイクルの矩形波がこのピンで駆動されます。 BuzzerOut1 ピンはフローティング状態にする、またはホスト制御 GPO として使用することができます。
2. AC 2 ピン ブザー: 図 3-3 に示すように、デバイスの BuzzerOut0 と BuzzerOut1 のピンに接続されます。これらのピンで、指定した周波数とデューティ サイクルを持つ 2 つの位相がずれた矩形波が駆動されます。

図 3-2. AC 1 ピン ブザーの設定

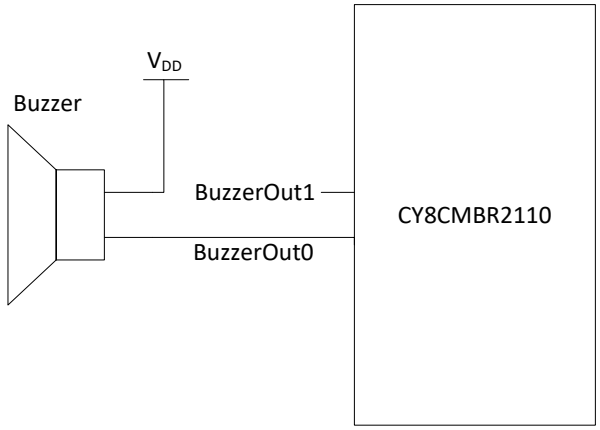
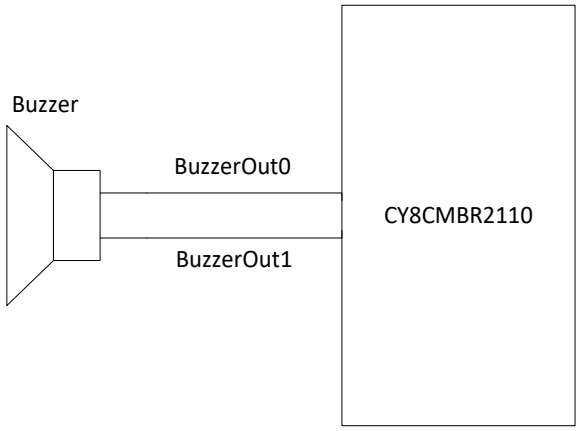


図 3-3. AC 2 ピン ブザーの設定



ブザー信号の周波数は設定可能です。表 3-1 に、様々な周波数および対応する出力デューティ サイクルを示します。

表 3-1. ブザー信号出力周波数およびデューティ サイクル

ブザー信号出力周波数 (kHz)	デューティ サイクル
1.00	50%
1.14	57.14%
1.33	50%
1.60	60%
2.00	50%
2.67	66.7%
4.00	50%

ブザー オン時間は ((1～127) x ボタン スキャン速度定数) の範囲内です。この定数の詳細についてはボタン スキャン速度を参照してください。ブザー信号出力は設定した時間の間駆動され、ボタン タッチ時間に依存しません。出力は図 3-4 に示すように、ボタンがタッチされたままであっても、ブザー オン時間が経過した後にアイドル状態になります。ブザー ピンのアイドル状態は V_{DD} とグランドのいずれかに対して設定することができます。ブザー信号出力は図 3-5 に示すように、ブザー オン時間が経過する前にボタンがタッチされた場合、直ちに再起動します。

ブザー信号出力を有効にすると (デバイス設定モードでの) Buz_Op_Duration レジスタは最小値の 1 になります。このレジスタの詳細は CY8CMBR2110 Datasheet の付録を参照してください。

図 3-4. ブザー タイムアウト

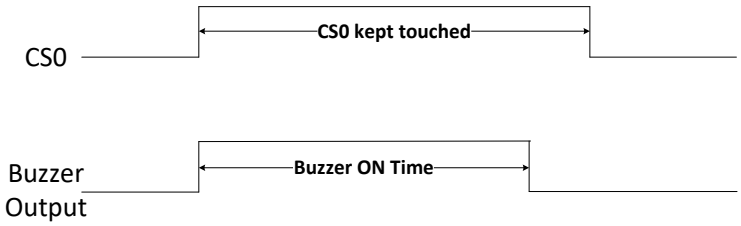
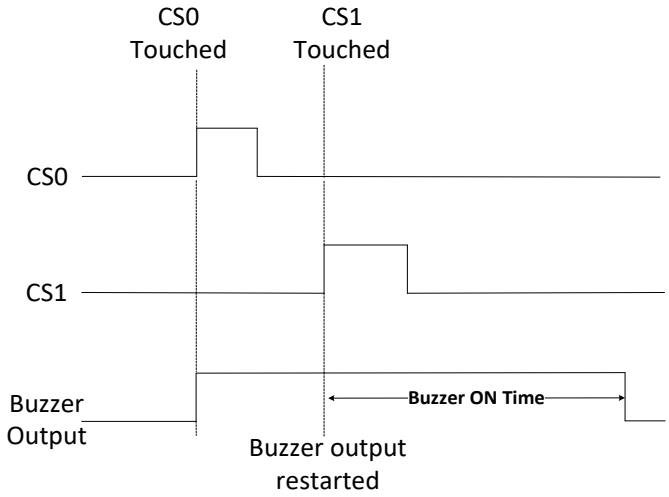


図 3-5. ブザー信号出力の再起動



3.1.5 ホスト制御 GPO (HostControlGPO0、HostControlGPO1)

ホスト制御 GPO のロジック状態はホストにより制御することができます。これらの出力はストロング駆動モードにあります。

ブザーを設計で使用しない場合、BuzzerOut0 および BuzzerOut1 のピンは ホスト制御 GPO として使用することができます。AC 1 ピン ブザーが使用される場合、BuzzerOut1 ピンはホスト制御 GPO として使用することができます。

ホストはこれらの GPO を動作モード、生産ラインのテスト モード および デバッグ データ モードで制御することができます。

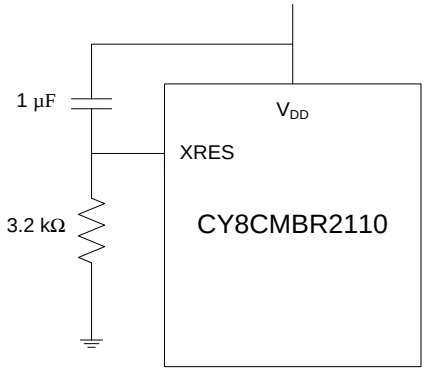
ホスト制御 GPO は電源投入時に LOW 状態になり、リセットの後で設定する必要があります。他の機能の設定と異なり、その設定はフラッシュ メモリに保存できません。

電源投入期間中、HostControlGPO1 は 16ms の立上りパルスを持っています。このパルスを除くには、図 3-6 に示すように、XRES ピンで外部 RC 回路 (±5%誤差) を使用します。これによって、電源投入期間中にデバイスは XRES リセットのままです。デバイスは 16ms 後に XRES リセットを終了して、通常通り動作します。

表 3-2. ブザーおよびホスト制御 GPO

ブザーの設定	BuzzerOut0 ピン	BuzzerOut1 ピン	使用可能なホスト制御 GPO の最大数
ブザーなし	開放／ホスト制御 GPO3	開放／ホスト制御 GPO2	4
AC 1 ピン	ブザー ピン 0	開放／ホスト制御 GPO2	3
AC 2 ピン	ブザー ピン 0	ブザー ピン 1	2

図 3-6 電源投入中の HostControlGPO1 パルスを止める XRES ピンの設定



3.1.6 Attention/Sleep

Attention/Sleep は、ホストとデバイスの両方が制御できるオープン ドレイン LOW 駆動モードでの双方向ラインです。Attention/Sleep は、デバイスの CapSense データの読み出し、低消費電力スリープおよびディープ スリープのモードへの移行およびそれらからの復帰に使用されます。

3.1.6.1 デバイス データの読み出し

ホストがデバイスのデータを読み出しには、2 段階が必要です。

1. ホストは Attention/Sleep ラインを LOW にします。
2. ホストはデバイスとの I²C 通信を初期化します。

Attention/SleepラインがHIGHに引き上げられると、(Host_Modeレジスタのディープスリープ ビットが設定された場合)デバイスは低消費電力スリープ モードまたはディープ スリープ モードになります。このとき、デバイスはI²C通信をNACKにできます。 電力を節約するため、Attention/SleepラインをHIGHに維持してください。
デバイス データを読む出すために、ホストはいつでも Attention/Sleep ラインを LOW に引き下げることができます。Attention/Sleep ラインが LOW の間、デバイスは I²C 通信を NACK にできますが、それは非常にまれです。

図3-7に示すように、任意のCapSenseボタンがタッチされると、デバイスはAttention/SleepラインがLOWになり、ホストに割込みます。その後、ホストはデバイスとのI²C通信により、CapSenseデータを読み出すことができます。図3-8に示すように、1個以上の

ボタンが同時にタッチされれば、Attention/Sleepラインがその接触時間中にLOWになります。ボタンを離すと、Attention/SleepラインはHIGHになります。

ホストは、Attention/Sleepラインの立ち上がりと立ち下がり両方のエッジ トリガ割込みである必要があり、これはボタン タッチとボタン解放の両方を認識するためのものです。立ち上がりエッジ トリガの割込みがなければ、Attention/SleepラインがLOWになった後、ホストは継続的にボタンの状態をポーリングする必要があります。そのポーリングはボタン スキャン速度定数で実行する必要があります。

図 3-7. CS0 と CS1 が個別にタッチされたときの Attention/Sleep ラインの状態

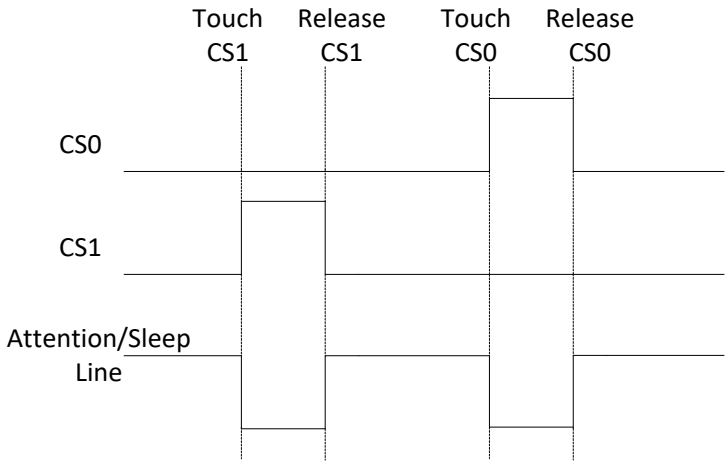
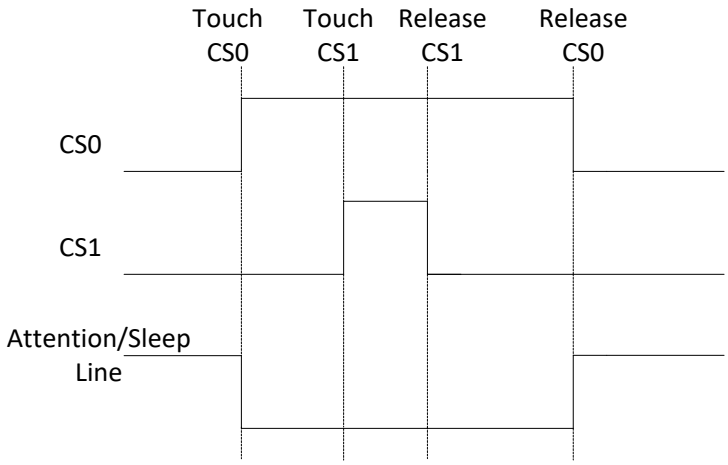


図 3-8. CS0 と CS1 が同時にタッチされたときの Attention/Sleep ラインの状態



3.1.6.2 スリープ モード

2 つの可能なスリープ モード設定があります。

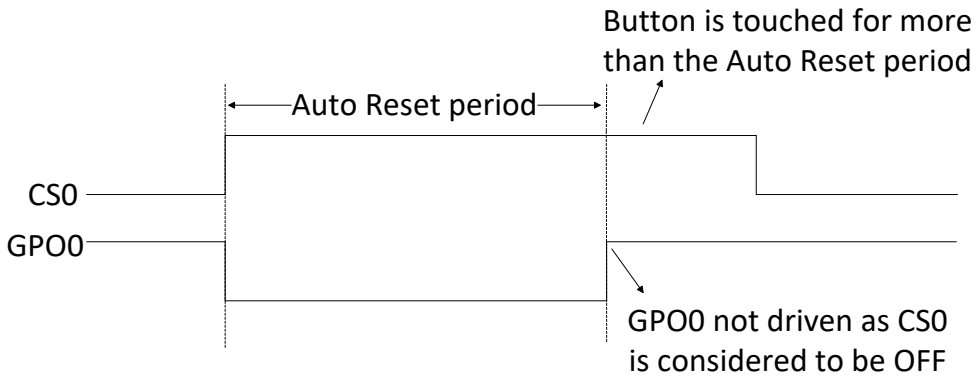
1. Attention/Sleep ラインに V_{DD} をセットし、低消費電力スリープ モードを有効にします。
2. Attention/Sleep ラインに V_{DD} をセットし、(動作モードで) Host_Mode レジスタのディープスリープ ビットをセットして、ディープ スリープ モードを有効にします。

3.2 CapSense コントローラーの設定

3.2.1 ボタン自動リセット (ARST)

ボタン自動リセットは CSx がタッチされたままのとき、ボタンがオンとみなされる最大時間を決定します。ボタンは ARST 期間後にオフになります。本機能は、金属性の物体がボタンに接近して置かれる場合にボタンが固まるのを防止します。ARST 期間は 5 秒と 20 秒のいずれかに設定することができます。ボタン自動リセットは図 3-9 に示されます。

図 3-9. ボタン自動リセット



CSx がボタン自動リセットによってオフになった後、およびボタンが離された後、ボタン スキャン速度に等しい期間にボタンにタッチしないでください。

3.2.2 ノイズ耐性

この設定は、外部の放射および伝導ノイズに対するデバイスの耐性を決定します。そのノイズは、電力増幅器からの可聴周波数ノイズ、無線送信機からの無線周波数ノイズ、静電気や電力線サージ等です。

ノイズの心配がないシステムではノイズ耐性の「中」を選択します。ノイズが酷い環境ではノイズ耐性の「高」を選択します。ノイズ耐性が「高」の場合、電力消費と応答時間が増加します。「高」のノイズ耐性で、応答時間が同じようにするには、ボタン デバウンス値を低くします。詳細はデバウンス制御を参照してください。

3.2.3 自動閾値

CapSense シグマデルタ (CSD) で説明したように、センサーの状態がオンまたはオフになるかは、raw カウントのシフト値を (指閾値という) 事前定義した閾値と比較することによって決定されます。指閾値は設定可能であり、デバイスの他の閾値を決定します。指閾値の詳細は「Getting Started with CapSense」を参照してください。

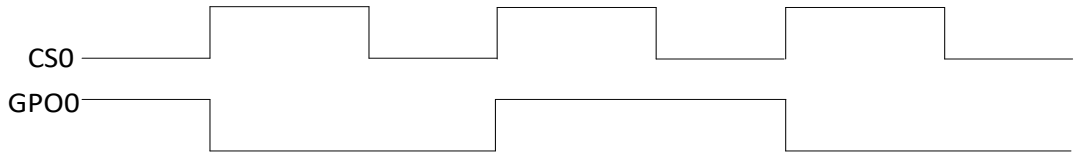
ユーザーは各ボタンに個別に指閾値を設定し、または自動閾値の機能を利用できます。環境ノイズに対応して、自動閾値は各ボタンに様々な閾値を動的に設定します。 可変ノイズの環境では、自動閾値を利用します。指閾値を手動で設定する必要があれば、自動閾値を無効にし、指閾値を望ましいレベルに設定します。

3.2.4 トグル オン／オフ

トグル オン／オフ機能が有効な場合、GPOx の状態は CSx の立ち上がりエッジで変化します。トグル オン／オフのコンフィギュレーションを図 3-10 に示します。

各 CapSense ボタンには、トグル オン／オフ機能を個別に有効にすることができます。

図 3-10. トグル オン／オフ機能の動作例



3.2.5 隣接センサー抑制 (FSS)

FSS は間隔の狭いボタンからの信号を区別し、誤ったタッチを排除します。これはシステムが最初の触れられたボタンのみを認識することを保証します。FSS では、一度に 1 個の CSx のみが「タッチ」状態となります。もし指が複数の CSx ボタンに接触しても、タッチ状態を感知した最初の 1 個のみがオンになります。

また FSS は、輝度調整 (高または低) 用に 2 個のボタンがあるインターフェースのような場合に、近くのボタンが逆のエフェクトを生み出す場合にも役に立ちます。

FSS は、各ボタンに個別に有効にすることができます。FSS コンフィギュレーションは図 3-11 と図 3-12 に示します。

洗濯機などのアプリケーションでは、ボタンは FSS が有効にされるグループと FSS が無効にされるグループの 2 つのグループに分けられます。これによって、設計の一方で近接したボタンを区別しながら、もう一方でマルチタッチ機能を実現できます。

図 3-11. 1 個のボタンだけをタッチするときの FSS



タッチの前にオンになるボタンがない

タッチしたときに CS1 は
オンとして報告される

図 3-12. 既に 1 個のボタンがオンになってから複数のボタンがタッチされるときの FSS



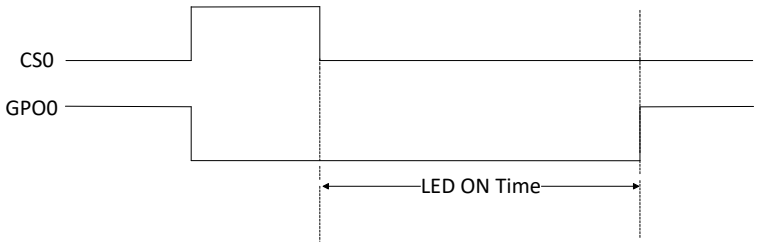
CS1 がタッチされると、
オンとして報告される

CS2 も CS1 と共にタッチされた場合、
CS1 のみはオンとして報告される

3.2.6 LED オン時間

LED オン時間は、図 3-13 に示すように、CSx 解放後に GPOx が LOW に駆動される期間を示します。LED オン時間の範囲は 0～5100ms で、その分解能は 20ms です。

図 3-13. LED オン時間

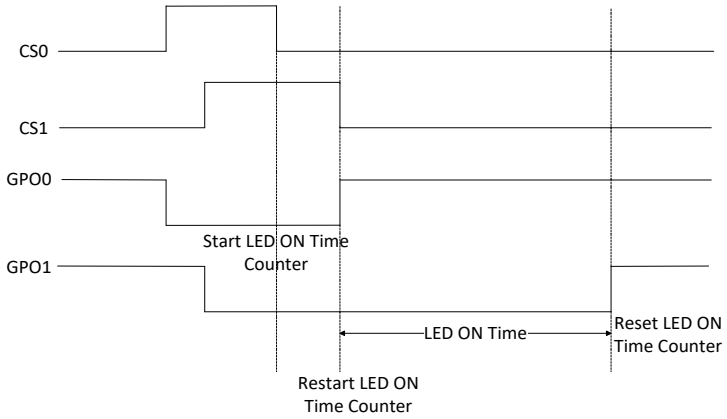


LED オン時間はデバイスによって異なります。精度は-40°C～+85°C の範囲で±10%です。

もしボタン自動リセット (ARST) が CSx でトリガされた場合、LED オン時間は GPOx に適用されません。トグル オン／オフが有効にされる場合、LED オン時間は無効になります。

LED オン時間は一度に 1 個の GPOx に適用されます。つまり、CSx がオンタッチ状態に移行するたびに LED オン時間のカウンタはリセットされます。図 3-14 に示すように、複数のボタンがタッチされたときの LED オン時間がどのようになるかを示します。CS1 が LED オン時間のカウンタをリセットしたので、GPO0 が早まってオフになっています。

図 3-14. 複数のボタンが絡む LED オン タイミング



3.2.7 LED エフェクトのパラメーター

パワーオン LED エフェクトとボタン タッチ LED エフェクトは、次のパラメーターを使用します。

- 低輝度: 最小 LED 輝度
- 低輝度時間: LED 低輝度状態の期間
- ランプアップ時間: LED が低輝度から高輝度に移行する期間
- 高輝度: 最大 LED 輝度
- 高輝度時間: LED が高輝度状態のままである期間
- ランプダウン時間: LED が高輝度から低輝度に移行する期間
- 反復回数: エフェクトが繰り返される回数

GPO は同じパラメーターを持つようにグループで設定されます。異なるグループは次のとおりです。

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

GPO0 のパラメーターは個別に設定することができます。この機能は、CS0 ボタンがパワー ボタンと同じように特別な機能を持つ設計で役立ちます。

輝度レベルは 0～100%の範囲にできます。時間範囲は 0～1600ms です。高輝度の値は低輝度の値よりも大きく設定する必要があります。

3.2.7.1 パワーオン LED エフェクト

この機能を有効にすると、GPO に接続するすべての LED はシステムの電源投入で初期時間に減光やフェージングエフェクトを表せます。エフェクトおよびその持続時間を設定することができます。この時間中に、全ての CapSense ボタンは無効になります。デバイスは、エフェクトが終了した後にボタン タッチに応答します。

このエフェクトは電源投入後のデバイス初期化後に可能です。ノイズ耐性が「中」であれば、この時間は 350ms 未満ですが、ノイズ耐性が「高」であれば、この時間は 1000ms 未満です。

電源投入後に、パワーオン セルフテストを含むシステム診断が実行されます。問題のある CapSense ボタンがある場合、それに対応する GPO に表示エフェクトは出力されません。このテストの詳細はシステム診断を参照してください。

パワーオン LED エフェクト中に、デバイスは I^2C 通信に ACK を返しますが、すべての書き込みコマンドは無視されます。ホストは動作モード データの読み出しだけができます。

図 3-15 および図 3-16 に示すように、パワーオン LED エフェクトは、すべての GPO で並行または順次発生するように設定することができます。

図 3-15. パワーオン LED エフェクトの例 (同時)[3]

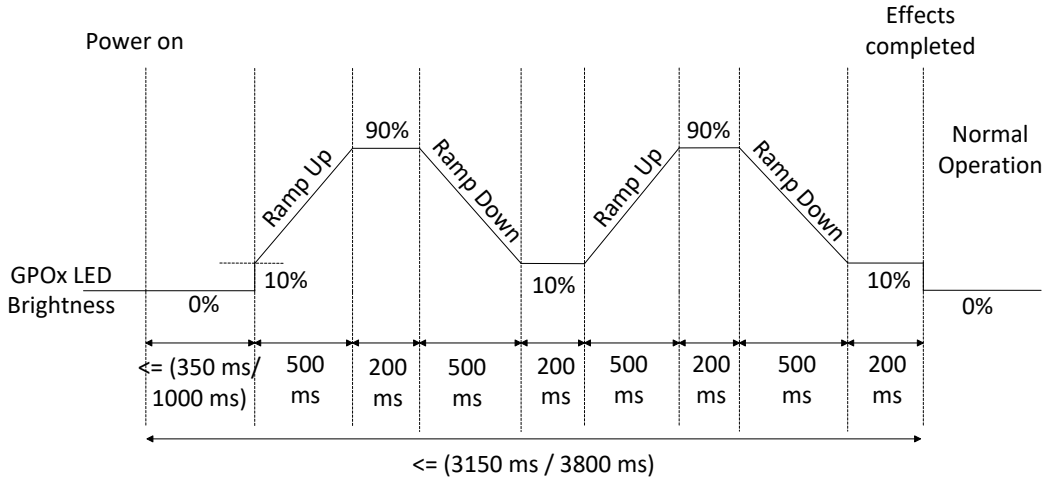
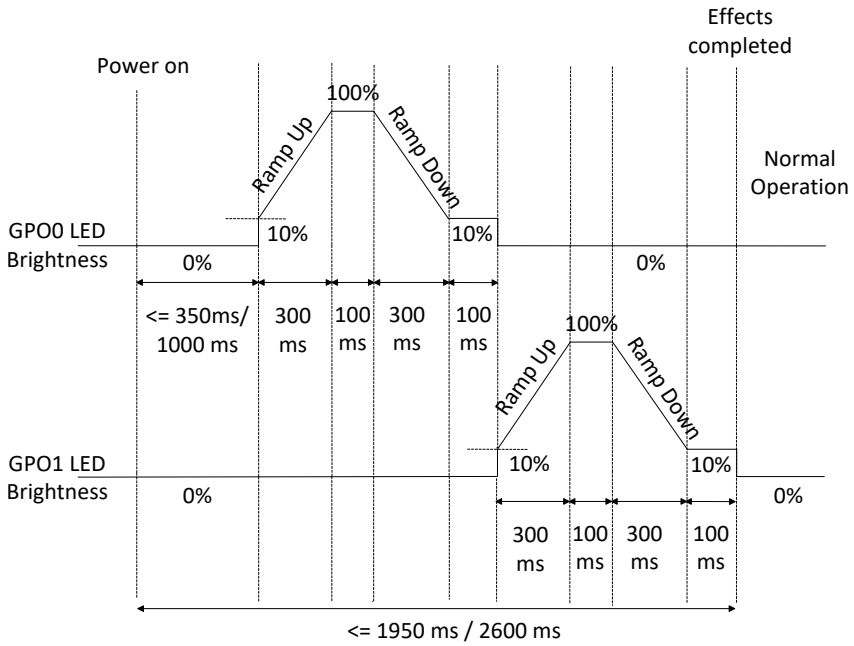


図 3-16. 2 ボタン設計でのパワーオン LED エフェクトの例 (シーケンシャル)[4]



[3] ランプ アップ時間=500ms。高輝度=90%。高輝度時間=200ms。ランプ ダウン時間=500ms。低輝度=10%。低輝度時間=200ms。繰り返し速度=1。

[4] ランプ アップ時間=300ms。高輝度=100%、高輝度時間=100ms。ランプ ダウン時間=300ms。低輝度=10%。低輝度時間=100ms。繰り返し速度=0。

3.2.7.2 ボタン タッチ LED エフェクト

ボタンにタッチすると、この機能は有効になります。そのとき、GPO に接続する関連 LED は減光およびフェージングエフェクトを表します。エフェクトおよびその持続時間を設定することができます。

ボタン制御 LED エフェクトのブリージング エフェクトは有効または無効になっています。その両方を図 3-17 に示します。

ブリージング エフェクトが有効: このとき、ボタンがタッチされると、LED 輝度は直ちにスタンバイ モード LED 輝度から低輝度に移行します。その後、LED は高輝度になり、高輝度期間はそのレベルのままです。そして、低輝度になり、低輝度期間はそのレベルのままです。ボタンがタッチされている限り、このエフェクトは繰り返します。ボタンが離されれば、ブリージング エフェクトサイクルはそれが完了するまで続きます。ブリージング エフェクト サイクルは反復回数によって繰り返すことができます。

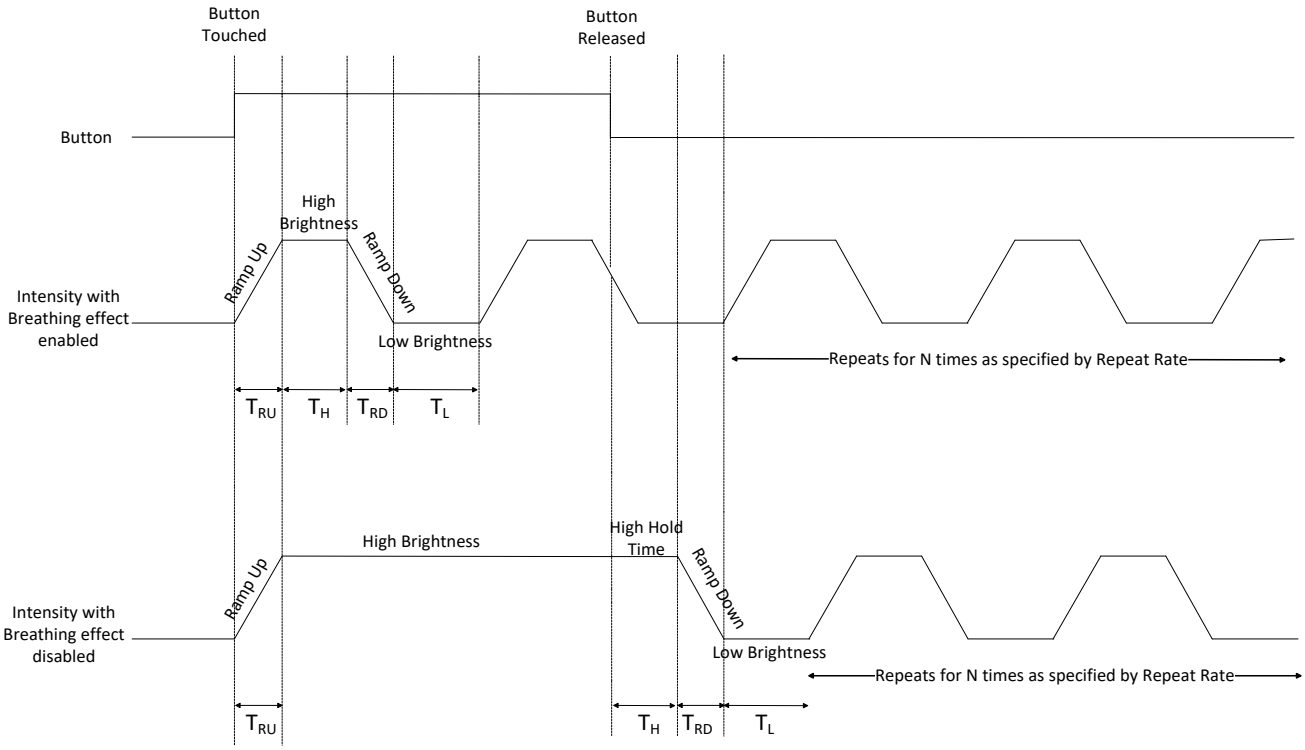
ブリージング エフェクトが無効: このとき、ボタンがタッチされると、LED 強度は直ちにスタンバイ モード LED 輝度から低輝度に移行します。その後、LED は高輝度にランプ アップし、ボタンがタッチされている限り、そのレベルのままです。ボタンが離されると、LED は高輝度期間に高輝度を維持します。その後、低輝度にランプ ダウンし、低輝度期間は低輝度のままです。このエフェクトは反復回数に応じて繰り返します。

ボタン タッチ LED エフェクトが GPOx に表示されている間、該当する CSx が再びタッチされると、パターンは GPOx で再開します。

図 3-18 に示すように、トグル オン／オフ機能が有効の場合、ボタンが連続してタッチされるとき、LED はスタンバイ モード LED 輝度と高輝度の間をトグルします。

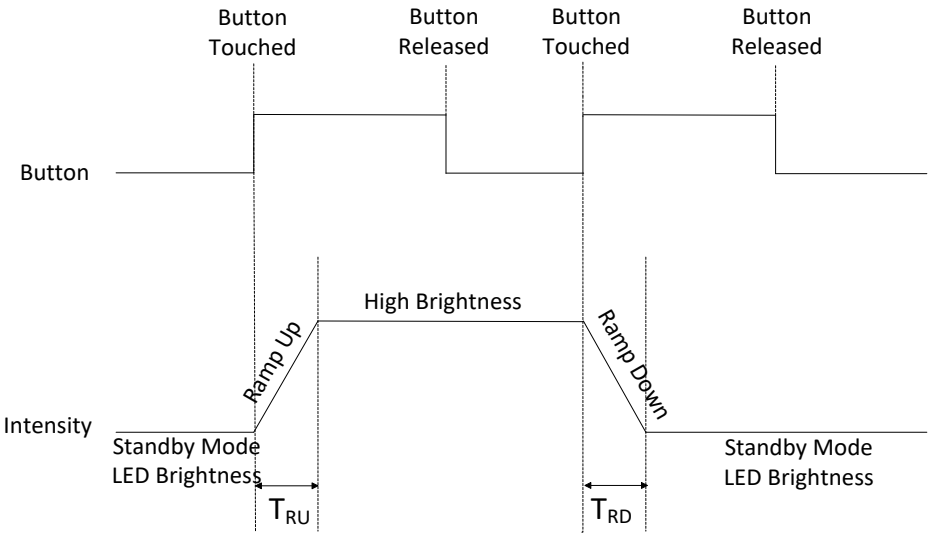
ボタン タッチ LED 効果を有効にすると、LED オン時間は自動的に無効になります。 デバイスがディープ スリープ モードに移行すると、実行中のボタン タッチ LED エフェクトは直ちに無効になります。

図 3-17. ボタン タッチ LED エフェクト[5]



[5] T_{RU} = ランプ アップ時間
T_{RD} = ランプ ダウン時間
T_H = 高輝度時間
T_L = 低輝度時間

図 3-18. トグル オン／オフが有効な場合のボタン タッチ LED エフェクト

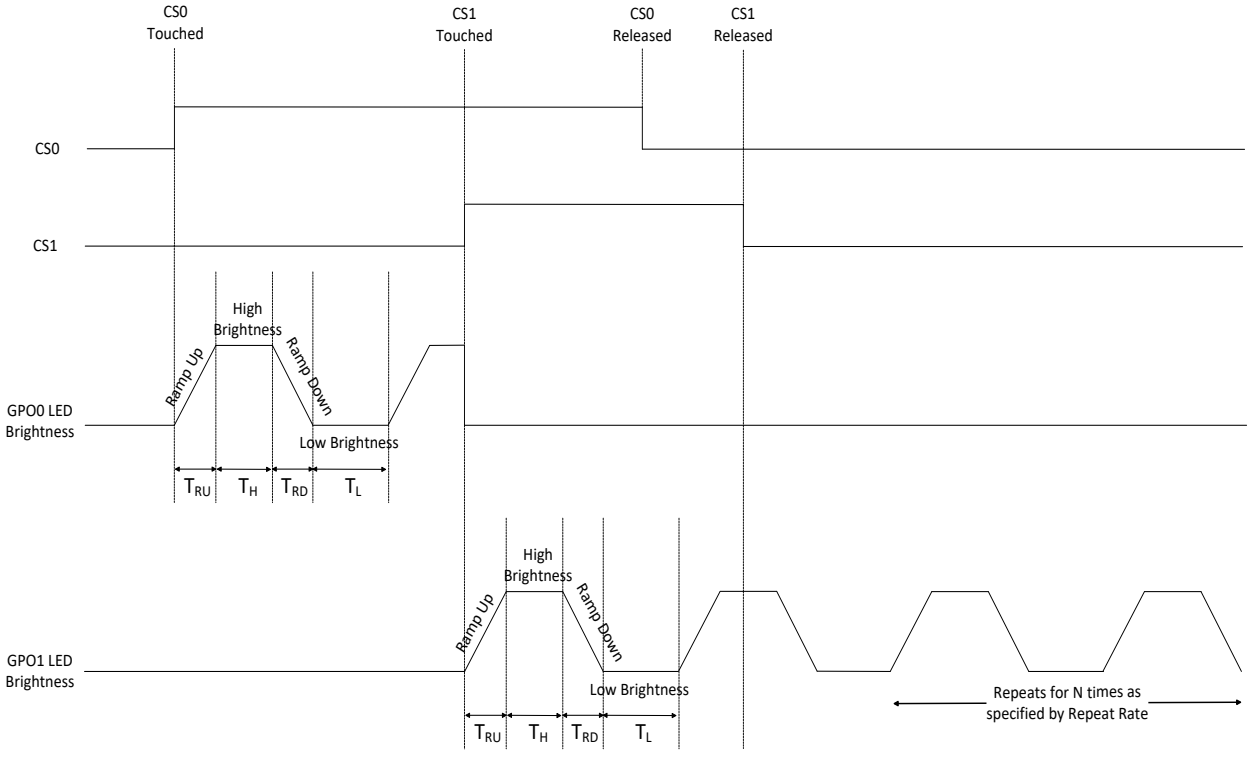


3.2.7.3 最終のボタン LED エフェクト

任意のボタンがタッチされるとボタン タッチ LED エフェクトが 1 個の GPO で中断されるように設定することができます。図 3-19 に示すように、エフェクトは最初の GPO でリセットし、最後にタッチしたボタンに対応する GPO で開始します。この機能はデフォルトで無効です。

トグル オン／オフが幾つかのボタンで有効にされると、最終ボタンLEDエフェクトは、それらのボタンで無効になります。隣接センサー抑制 (FSS) が有効であり、2個のボタンが同時にタッチされた場合は、2番目にタッチされたボタンがオンにならないので、最終ボタンLEDエフェクトは適用されません。

図 3-19. 最終ボタン LED エフェクトが有効な場合のボタン タッチ LED エフェクト (ブリージングが有効)



3.2.7.4 スタンドバイ モードでの LED の輝度

CapSense ボタン CSx がオフの場合、該当する GPOx に接続する LED を設定して、LED バックライト用のスタンドバイモード LED 輝度を得ることができます。この設定は設計の見栄えを良くします。

スタンバイ モード LED 輝度は、0%、20%、30%または 50%に設定することができます。 スタンバイ LED 輝度は低輝度と同じである必要があります。

CSx がオフになっているとき、パワーオン LED エフェクトまたはボタン タッチ LED エフェクトの終了後、GPOx に対応する LED はスタンバイ モード LED 輝度のままです。

デバイスが低消費電力スリープ モードに移行しないため、スタンバイ モードLED輝度によって、デバイスの電力消費は増加します。デバイスがディープ スリープ モードに移行すると、スタンバイ モードLED輝度は無効になります。

3.2.8 ラッチ状態の読み出し

CapSense ボタン CSx がタッチされると、デバイスは Attention/Sleep ラインを LOW にプルすることでホストに割込みを生成します。そして、ホスト プロセッサは I²C 通信を介してデバイスのレジスタ マップを読み出し、CapSense ボタンの状態を判断できます。詳細は「Attention/Sleep」を参照してください。I²C 通信の詳細は CY8CMBR2110 データシートを参照してください。

割込みがデバイスからホストに生成されるとき、ホストはその割込みを直ちに処理できないことがあります。その結果、ホストはボタン タッチを見逃すことがあります。ボタン タッチを見逃さないようにホストは任意のボタン タッチについての正確な情報を得るために、ボタン状態 (CS) とラッチ状態 (LS) の両方を読み出す必要があります。 CS は Button_Current_Stat0 および Button_Current_Stat1のレジスタに動作モードで記録されます。LSはButton_Latch_Stat0およびButton_Latch_Stat1のレジスタに動作モードで記録されます。レジスタ マップの詳細はCY8CMBR2110データシートの付録を参照してください。

ボタン ステータス ビットはボタン タッチのときにセットされ、ボタン解放のときにクリアされます。ラッチ ステータス ビットはボタン タッチのときにセットされます。ホストがボタンの状態を読み出すと、このビットは自動的にクリアされます。

表 3-3 に様々なボタン タッチのケースおよびそれらに対する確認応答を示します。これらのケースは図 3-20 と図 3-21 に示します。

表 3-3. ラッチ状態の読み出し

ボタン状態 (CS)	ラッチ状態 (LS)	解釈
0	0	CSx が現行の I²C の読み出し中にタッチされていない。 最後の I²C の読み出しのとき、ホストは以前の CSx タッチを既に確認した。
0	1	CSx が現行の I²C の読み出しの前にタッチされた。 ホストはこの CSx 接触を見逃した。
1	0	CSx が前の I²C の読み出し中にタッチされ、ホストにより確認された。 この CSx が現行の I²C の読み出し中にタッチされたままである。
1	1	CSx が現行の I²C の読み出し中にタッチされたままである。

図 3-20. ラッチ状態の読み出し、ケース 1

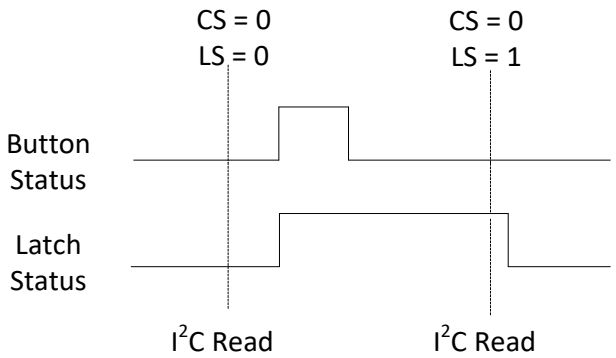
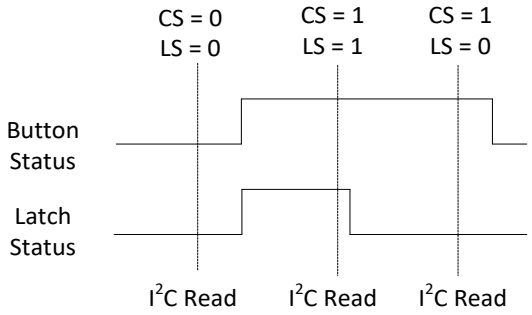


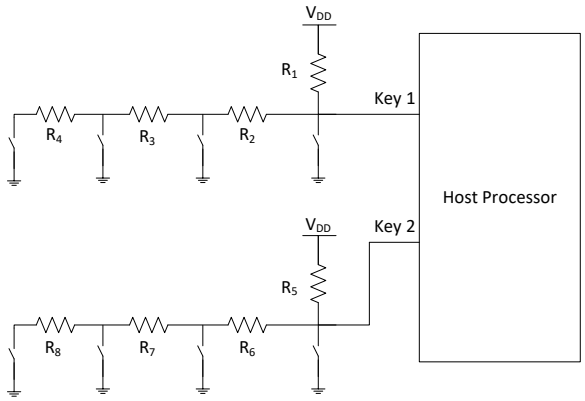
図 3-21. ラッチ状態の読み出し、ケース 2



3.2.9 アナログ電圧サポート

図 3-22 に示す例のように、ホスト プロセッサを持つ一般的な外部抵抗ネットワークでは、入力ピンで見えられる電圧レベルに基づいて異なる機能を実行するようホストを構成することができます。V_{DD} とグランドの間で抵抗とスイッチとの組み合わせを使用して、電圧レベルを変化させます。

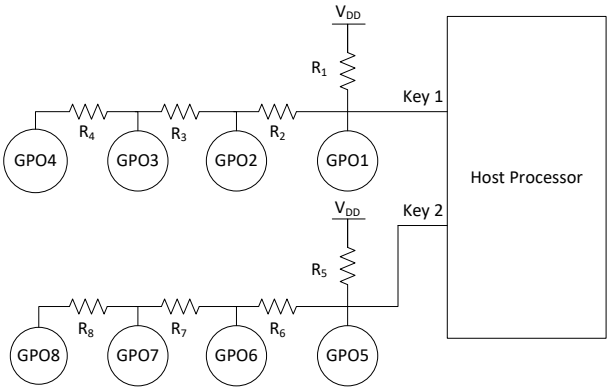
図 3-22. 外部抵抗ネットワークの例



CY8CMBR2110 のアナログ電圧サポート機能により、CapSense ボタンを使って、これらのスイッチを選択して制御することができます。各スイッチは、1 個の GPOx に置き替えることができます。1 個の CSx ボタンがタッチされたとき、対応する GPOx が LOW になり、スイッチが閉じます (グランドに短絡)。ボタンを離したとき、対応するスイッチは開放になります。これは図 3-23 に示されます。

この機能を有効にすると GPO は LED 駆動に使用できません。1 個のボタンだけをアナログ電圧サポート用にオンにする必要がある場合、FSS をこの機能と共に有効にします。通常、GPO ピンはストロング駆動モードにありますが、この機能が有効になると、GPO はオープン ドレイン LOW 駆動モードです。

図 3-23. CY8CMBR2110 のアナログ電圧サポート



3.2.10 感度制御

各 CapSense ボタンの感度は個別に設定することができます。感度は、ボタンをオンにするための最小限必要な C_F を決定します。以下の要素はボタンの感度に影響を与えます。

1. オーバーレイの厚さ: オーバーレイが厚いほど、感度要件は高くなります。
2. システム ノイズ: システム ノイズが増加するときは、ボタンによる誤ったトリガを防止するために、感度を低くする必要があります。
3. 設計の形状要因: 低い感度 (高い C_F) をサポートするために、比較的大きなボタン形状が必要です。小さい径のボタンでは、感度を高くする必要があります。
4. 消費電力: 高い感度のボタンは消費電力が高くなります。低消費電力のためには、感度を低くする必要があります。

感度の設定のレベルは「高」、「中」、および「低」です。

3.2.11 デバウンス制御

このデバウンス機能により、有効なタッチ入力となるようにボタンがタッチされる最小時間を指定することで、ボタンがノイズスパイクまたはシステム グリッチにより誤ってトリガされることを回避できます。

デバウンス時間はボタンの機能によって異なります。例えば、電源ボタンはシステム オン／オフが誤って切り替えられることを防ぐために、長いデバウンス時間にする必要があります。デバウンス時間を短くすると、ボタン タッチへのデバイスの応答が速くなります。

CS0 ボタンのデバウンス値は CS1～CS9 ボタンのデバウンス値と別々に設定することができます。この機能は、CS0 ボタンが電源ボタンなどのように特別な機能を持つ設計に役立ちます。デバウンス値の範囲は 1～255 です。

デバイスの応答時間はボタンのデバウンス値に依存します。表 3-4 に、異なるデバウンス値に応じたデバイスの応答時間の例を示します。[6]任意のデバウンス値に対し応答時間を計算する方法は応答時間を参照してください。

表 3-4.各デバウンス値に対する応答時間の例

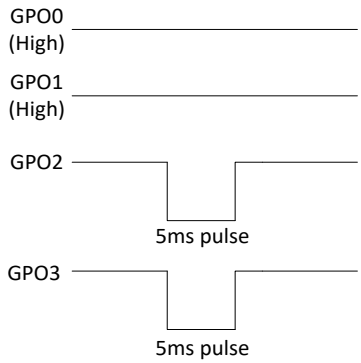
デバウンスの値	繰り返しボタン接触の応答時間 (ms)
1	70
4	105
7	140
10	175
100	1225
200	2380
255	3010

[6] 8 個のボタン、ノイズ耐性「中」レベル、最適化された応答時間の設計

3.2.12 システム診断機能

内蔵の電源投入時自己診断 (POST) メカニズムにより、電源投入時のリセット (POR) のときに、5 つのテストを行うことができます。これは製造試験において便利です。 いずれかのボタンが失敗した場合、5ms のパルスが 350ms (ノイズ耐性が「中」の場合)、または 1000ms (ノイズ耐性が「高」の場合) の間、対応する GPO から送信されます。

図 3-24. POST に CS0 と CS1 は合格し、CS2 と CS3 に問題がある例



システム診断結果を取得するには、EZ-Click カスタマイザ ツールを使用してください。ツールの詳細は「EZ-Click Customizer Tool User Guide」を参照してください。

ユーザーがデバイス データをすべて読み出す必要がある場合、デバイスのレジスタ マップ モードを「製造ライン テスト」モードに変更し、I²C ラインを介してデータを読み出してください。レジスタ マップ モードを変更する方法の詳細は「CY8CMBR2110 データシート」のレジスタ マップ モードを参照してください。デバイス データの詳細は「I2C 通信」を参照してください。

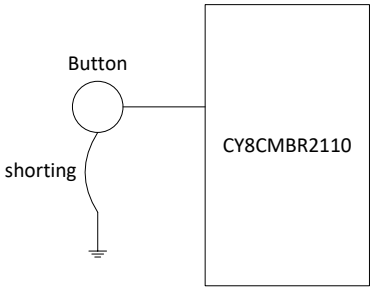
I²C を使って GPO の状態を読み出すため、GPO とホスト コントローラー ピン間のインターフェースを作成する必要がありません。

以下のテストはすべてのボタンで実行されます。

3.2.12.1 グランドに短絡したボタン

接地が検出されると、ボタンは無効になります。

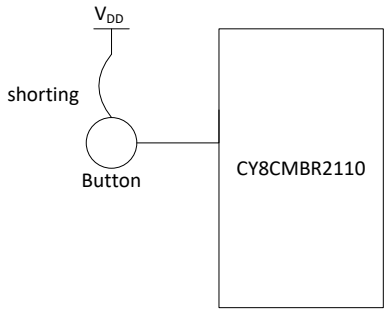
図 3-25. グランドに短絡したボタン



3.2.12.2 V_{DD} に短絡されたボタン

V_{DD} への短絡が検出されると、ボタンは無効になります。

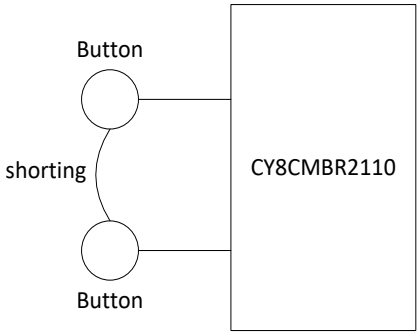
図 3-26. V_{DD} に短絡されたボタン



3.2.12.3 互いに短絡されたボタン

複数ボタンの相互短絡が検出されると、該当のボタンは無効になります。

図 3-27. 互いに短絡されたボタン



3.2.12.4 不適切な CMOD

CMOD の推奨値は 2.2nF±10%です。

CMOD の値が 1nF より小さく、または 4nF より大きいと、全てのボタンが無効になります。

3.2.12.5 ボタン C_P>40pF

任意のボタンの C_P が 40pF より大きいと、そのボタンは無効になります。

3.2.13 ボタン スキャン速度

ボタン スキャン速度は、デバイスによる連続したボタン スキャン間の時間を指定します。次式を使って、速度を計算します。

ボタン スキャン速度=ボタン スキャン速度定数+ボタン スキャン速度オフセット　　式 4

ボタン スキャン速度は 25～561ms で設定可能です。

ボタン スキャン速度の定数は、ボタンの数および選択したノイズ耐性レベルに依存します。ボタンの数は多くなるほど、定数は高くなります。同じように、ノイズ耐性が「高」の場合、定数は高くなります。

最大 5 個のボタンを使用する場合、ボタン スキャン速度の定数は、設計を最適化する方法に依存します。

応答時間の最適化: 連続したボタン スキャン間の時間は短くなります。一定の時間内で多くのスキャンが行われるため、デバイスはボタン タッチに速く対応できます。その反面消費電力が増加します。

消費電力の最適化: 連続したボタン スキャン間の時間は長くなります。一定の時間内でより少ないスキャンが行われるため、デバイスはボタン タッチの応答により長い時間がかかります。その結果として、消費電力は減少します。

ボタン スキャン速度オフセットは「EZ-Click カスタマイザ ツール」で設定することができます。ボタン スキャン速度定数は表 3-5 に示されます。

表 3-5. ボタン スキャン速度の定数

ボタン数	ボタン スキャン速度の定数			
	応答時間の最適化をした場合		消費電力の最適化をした場合	
	ノイズ耐性「中」	ノイズ耐性「高」	ノイズ耐性「中」	ノイズ耐性「高」
≤5	25	35	35	55
>5	35	55	35	55

例として、4 個のボタンおよび以下のパラメーターを持つ設計を考えてみましょう。

- すべてのボタンの C_P が 10～20pF
- すべてのボタンの感度が高い
- ノイズ耐性は「中」
- 各ボタン用のデバウンスは 10 にセット
- 1 時間当たりのボタン タッチの平均回数=200
- 平均タッチ時間=1000ms
- ブザーおよびボタン タッチ LED エフェクトが無効
- ボタン スキャン速度オフセット=0
- ボタンごとの消費電流は次のとおりです。
  - ☐ 応答時間を最適化した場合=0.3075mA
  - ☐ 消費電力を最適化した場合=0.2204mA

最初のボタン タッチおよび連続したボタン タッチの応答時間は次のとおりです。

- ☐ 応答時間を最適化した場合=125ms
- ☐ 消費電力を最適化した場合=175ms

応答時間が最適化された設計は、消費電力が最適化された設計に比べて、さらに多くの電力を消費し、ボタン タッチにさらに速く応答することに注意してください。適切な応答時間は「Design Toolbox」を参照してください。

デバイスによってボタン スキャン速度は異なり、-40°C～+85°C の温度範囲において±10%の精度です。

3.2.14 I²C 通信

I²C は、CY8CMBR2110 (I²C スレーブ) とホスト (I²C マスター) との間で通信するために使用されるインターフェースです。

プロトコルおよび通信手順の詳細は CY8CMBR2110 データシートの I²C 通信の節を参照してください。

ホストとデバイスとの適切な I²C 通信のために、以下のガイドラインに従ってください。

- 任意の I²C 通信を初期化する前に、ホスト プロセッサは Attention/Sleep ラインを LOW にする必要があります。そうでない場合、デバイスはホストを NACK する可能性があります。
- ホスト プロセッサは、以下の場合以外、デバイスとの I²C 通信が開始されないか、継続しません。
  - ☐ ホストがデバイスを設定する必要がある場合
  - ☐ デバイスがホストに割込みをする場合
  - ☐ ホストがデバイスのレジスタ マップの内容を読み出し、検証する必要がある場合
- 消費電力を低くするには、デバイスとの長時間の I²C 通信を避けてください。
- デバイスの電源投入後、任意の I²C 通信を開始する前に、ホストは 350ms (ノイズ耐性が「中」の場合) または 1000ms (ノイズ耐性が「高」の場合) 待機する必要があります。そうでなければ、デバイスはそのような通信を NACK します。
- 任意の I²C 通信の後、新しい通信を開始する前にホストは少なくとも 60ms 待機することが必要です。

- 「フラッシュに保存」または「ソフトウェア リセット」コマンドが発行された後、任意の I²C 通信を開始する前に、ホストは 350ms (ノイズ耐性が「中」の場合) または 1000ms (ノイズ耐性が「高」の場合) 待機する必要があります。
- 実行時ではデバイスは動作モードにある必要があります。
- ホストは前の I²C 通信に STOP 条件を発行することなしに、デバイスに新しい START 条件を発行してはいけません。これは、反復 START 条件とも呼ばれています。
- ホストは、連続した I²C 通信間では最短 60ms を維持する必要があります。
  - ☐ この時間中に、ホストが他の I²C 通信を開始すると、前の通信データと同じデータを受信します。
  - ☐ この時間中に、ホストが前の通信で書き込まれたレジスタに書き込むと、古いデータは失われます。
  - ☐ この時間中に、ホストが前の通信で書き込まれたレジスタ以外のレジスタに書き込むと、そのレジスタはこのデータを保持します。前の通信で書き込まれたデータは失われません。

3.3 デザイン ツールボックス

「Design Toolbox」は CY8CMBR2110 CapSense ソリューションを設計するのに役立ちます。基板のレイアウトおよび機能設定に関する基本情報を提供し、その設計が大量生産に適しているかどうかを提示します。

3.3.1 全般的なレイアウト ガイドライン

図 3-28 に、CY8CMBR2110 レイアウトのガイドラインをまとめています。このガイドラインは電気的／機械的設計時の考慮事項の章で説明します。この資料についての詳細は「Getting Started with CapSense」を参照してください。

図 3-28. 設計レイアウトの推奨事項

General Layout Guidelines

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

3.3.2 レイアウト エスティメータ

レイアウト エスティメータは、目的とする最終製品の要件および工業デザインに基づいて、最小ボタン サイズや最大配線長の推奨値を提供します。入力は、オーバーレイの素材、オーバーレイの厚さ、回路基板素材の配線容量および CapSense ボタン感度を含みます。図 3-29 および表 B には、さまざまなオーバーレイ素材の誘電率と PCB の単位長さ当たりの配線容量を示しています。表 A では、3 つのシステム ノイズの条件で最小ボタン直径および最大配線長を計算します。「低」、「中」、「高」のノイズ状態は、ボタン開発の目安とする相対的な性能指数です。ノイズ状態は、最終製品の環境に基づいたボタンによって異なります。ノイズ状態が不明な場合は、まず「中」を使用して開始します。各ボタンの実際のノイズはデザイン検証 中に決めます。

このシートの出力をボタン基板レイアウトの過程で使用してください。そして、試作する前に、「CP、消費電力、応答時間計算器」で詳しく説明している C_P、消費電力、および応答時間計算器シートを使って設計を確認してください。

図 3-29. レイアウト エスティメータ

Layout Estimator

TableA: Layout Estimator

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

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

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

TableB - Industry Standard Reference Values

Overlay Material	Dielectric constant
Plastic	2.8
Plexi glass	8
Formica	4.6-4.9
Glass (Standard)	7.6-8.0
Glass (Ceramic)	6
Mylar	3.2
ABS Plastic	3.8 – 4.5
Wood	1.2-2.5

Trace and board type	Capacitance per inch in pF
Copper trace , PCB, 2 layer, 64mil, FR4	2
Copper trace, flex PCB, 2 layer	8

入力

- オーバーレイの厚さ
- オーバーレイの比誘電率
- 基板 1 インチ当たりの配線の静電容量
- CSx 感度

出力

- 異なるノイズ状態に対する推奨最小ボタン直径および最大配線長
- ボタンとグランドの間隔

各ボタンの直径は、各ボタンのノイズ変動に基づいて変わります。

3.3.3 C_P、消費電力、応答時間計算器

基板のレイアウトが完了した後、図 3-30 に示した消費電力および応答時間計算器を使用して、ボタン基板の試作品を作る前にデザインをチェックします。各ボタンの C_P 値を検証するには、表 A にボタン直径と配線の長さを挿入します。この情報を入力すると、ツールボックスは各ボタンが指定された 5～40pF の C_P の範囲内にあるかどうかを確認します。

表 B にある消費電力計算器を用いて消費電力を最適化します。消費電力は、ボタン スキャン速度、ノイズ耐性レベルとアクティブ時間 (パーセント単位) の関数です。アクティブ時間は、1 時間当たりの平均ボタン タッチ回数をボタン タッチ時間、ブザー オン時間、およびボタン タッチの LED 効果という 3 つの値の最大値と掛けることにより計算されます。計算結果の値がアクティブ時間のパーセント単位に変換され、消費電力はそれに基づいて計算されます。以下の全ての入力セルが同時に空 (またはゼロ) でないことを確保してください。

1. 1 時間当たりのボタン タッチの平均回数
2. ボタン タッチの平均時間
3. ブザー オンの平均時間
4. ボタン タッチ LED エフェクトの平均時間

表 C は、表 A と表 B への入力に基づいてボタンの応答時間を出力します。デバウンス値はボタンの応答時間に影響します。

図 3-30. C_P、消費電力、応答時間計算器

Cp, Power Consumption and Response Time Calculator

Table A: Cp Calculator

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

Table B: Power calculator

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

Table C: Response time calculator

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

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

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

入力

- レイアウトで設計した CS0～CS9 のボタンの直径および配線の長さ
- CS0～CS9 の感度
- ボタン スキャン速度のオフセット
- 設計最適化
- ノイズ耐性レベル
- CS0 デバウンス
- CS1～CS9 デバウンス
- 1 時間当たりのボタン タッチの平均回数
- ボタン タッチの平均時間
- ブザー オンの平均時間
- ボタン タッチ LED エフェクトの平均時間
- スタンドバイ モードでの LED の輝度
- 消費電流の計算因子

出力

- 各ボタンの C_P (C_P 値が 5～40pF の指定範囲内にあるかどうかを確認)
- ボタンごとの消費電流
- ボタンの応答時間

3.3.4 デザイン検証

試作品の基板を立ち上げてテストを実施した後、EZ-Click カスタマイザ ツールを使用して、全てのボタンの raw カウント、ノイズ カウント、および C_P をキャプチャします (EZ-Click User Guide を参照してください)。この情報とデザイン検証シートを使って、デザイン検証 に詳しく記述されているようにデザインを検証できます。

表 A には、前のシートから取り込まれた様々な設計パラメーター値を示しているため、このシートに何のデータも入力する必要がありません。このシートは試作品基板の合否を判定します。デザインが不合格の場合は、システムを再設計することができます。新しい値を表 A に入力すると、更なる推奨事項や結果を得られます。デザインが合格の場合は、表 A の「新しい値」欄を空白のままにします。

表 B は C_P、消費電力、および応答時間計算器のシートから取り込まれたボタンの感度値を示します。デザインが不合格の場合は、新しい値を入力することでボタンの感度を再設計できます。デザインが合格の場合は、表 B の「新しい値」欄を空白のままにします。

図 3-31. デザイン検証

Design Validation

Table A: Actual Design values

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

Table B: Button Sensitivity

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

Table C: Reference values

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

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

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

Table D: Power consumption, Button diameter actuals

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

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

EZ-Click カスタマイザ ツールを使って、データを表 D に入力するためには、次の手順に従ってください。

1. デバイスの電源を入れた後、USB-I²C ブリッジ (CY3240-I2USB Bridge) を使用してご使用のコンピュータに接続します。USB-I²C ブリッジ (CY3240-I2USB Bridge) の詳細は AN2397 – CapSense Data Viewing Tools を参照してください。

2. EZ-Clickカスタマイザ ツールを起動して新しいプロジェクトを作成します。サイプレスのCY8CMBR2110デバイスを選択します。ポート選択のウィンドウから使用しているポートを選んで「**Connect**」をクリックします。
3. デバイス設定のタブを開いて、所望されるデザイン用のボタン数を選択します。必要な場合にはCapSenseピンを該当のボタンに割当てます。指閾値を設定するか、または自動閾値を選択します。
4. CapSense出力のタブを開いて、「Button Specific Output view」を選択します。
5. CapSense出力を表示させるボタンを選択します。「Raw Count vs Baseline」グラフを選択します。
6. rawカウントのグラフを観測して、300個のサンプルからrawカウントの平均値を記録します。C_Pボタンの値も記録します。
7. 下式に従ってノイズ カウントを計算します。
   ノイズ カウント=最大rawカウント-最小rawカウント (300個のサンプル)
8. このデータを表Dに入力して消費電流値を取得した後、このデザインが大量生産に適しているかどうかを決めます。

入力

- raw カウント
- ノイズ カウント
- ボタンの C_P
- デザインが不合格の場合、以下の値を考慮してください。
  - ☐ 各ボタンの新しいオーバーレイの厚さ、オーバーレイ素材の誘電率、ボタンの直径、および配線の静電容量
  - ☐ CSx 感度

出力

- ボタンごとの消費電流
- デザイン変更の推奨事項。デザイン ツールボックスは、ボタン サイズまたは配線の長さが最良の設計例から外れている場合、そのデザインからの実際の値に基づいた推奨事項を提供します。

ボタン基板が不合格の場合、デザイン ツールボックスは合格の結果を得るよう推奨事項を提供します。不合格デザインを改善するために次の 4 項目を変更できます: ボタン サイズ、配線の長さ、オーバーレイの素材、オーバーレイの厚さです。ボタン サイズや配線の長さを変える場合、基板の作り直しが必要ですが、オーバーレイの素材や厚さまたはその両方を変えることでデザインを合格する場合もあります。最善の解決法は、デザインの開発サイクル中の位置付け、および最終製品の要件に依存します。

3.4 CY8CMBR2110 設定

CY8CMBR2110 は、次の方法のいずれかによって設定できます。

1. EZ-Click カスタマイザ ツール
2. ホスト プロセッサによるデバイス設定
3. サードパーティ プログラマ

CY8CMBR2110 デバイスを設定するための基本的な手順は、複数のステップで示されます。すべての設定方式に対して、これらは共通の手順です。EZ-Click カスタマイザ ツールは、この手順を自動的に実行しますが、ホスト プロセッサは以下の手順に従わなければなりません。

1. デバイスのモードを LED コンフィギュレーション モードに変更します。
2. 55ms 待機します。
3. LED コンフィギュレーション モードのすべてのコンフィグレーション レジスタに書き込みます。
4. 55ms 待機します。
5. デバイス モードをデバイス コンフィギュレーション モードに変更します。
6. 55ms 待機します。
7. デバイス コンフィギュレーション モードのすべてのコンフィグレーション レジスタに書き込みます。

8. チェックサム値を計算し、その結果を (デバイス コンフィギュレーション モードの)「Checksum_MSB」(0x1E) と「Checksum_LSB」(0x1F) レジスタに入力します。

   チェックサム: これは LED コンフィギュレーション モードにあるレジスタ (0x01～0x1F) 値とデバイス コンフィギュレーション モードにあるレジスタ (0x01～0x1D) 値の合計です。チェックサムには、予約済みのレジスタ ビットの値も含まれています。ホストはこれらのビットに書き込んではなりません。そしてチェックサムを計算するときはそのようなビットに 0 を追加しなければなりません。

   (動作モードにある) Checksum_Flash_xxx レジスタはフラッシュに格納されているチェックサムを示します。

   (動作モードにある) Checksum_RAM_xxx レジスタは、現在の設定のためデバイスにより計算され、RAM に格納されているチェックサムを示します。

9. 55ms 待機します。

10. (デバイス コンフィギュレーション モードの) Host_Mode レジスタの「チェックサム一致」ビットを読み込んで、1 にセットされていることを確認します。このビットがまだセットされていない場合、ステップ 1 から再開しデバイスを再設定します。必要に応じて、ホストはコンフィギュレーション データのバックアップを保存する必要があります。

    「チェックサム一致」ビット: CY8CMBR2110 はチェックサムを計算して、ホストが入力したチェックサム レジスタの値と比較します。これら両方の値が一致する場合、Host_Mode レジスタ内の「チェックサム一致」ビットが 1 にセットされます。これら両方の値が不一致の場合、I^2C 書き込みエラーである可能性を示し、このビットは 0 にクリアされます。ホストは、(動作モードにある) Checksum_RAM_xxx レジスタを読み込んで、デバイスが計算したチェックサムを取得することができます。

11. 「チェックサム一致」ビットが 1 にセットされると、Host_mode レジスタの「フラッシュに保存」ビットをセットします。

    フラッシュに保存: 「フラッシュに保存」プロセスでは次の手順を行います。

    (i) デバイスは、(LED コンフィギュレーション モードおよびデバイス コンフィギュレーション モードの) 64 バイトのデータをフラッシュにコピーします。

    (ii) ソフトウェア リセットを行います。

    (iii) ソフトウェア リセット後に、デバイス モードは動作モードになります。

    設定の変更は、フラッシュに保存しない限り、適用されません。以降のすべての動作のためにデバイスが 1 回のみ設定される必要がある場合には、この「フラッシュに保存」プロセスが役立ちます。「フラッシュに保存」プロセスの実行中は、デバイスの電源が安定している必要があります。V_{DD} の変動は±5%まで許容されます。

12. 「フラッシュに保存」プロセスの実行後、(T_{SAVE_FLASH}+デバイス初期化) 時間を待機します。T_{SAVE_FLASH} は CY8CMBR2110 データシート内のフラッシュ書き込み時間の仕様に記述されます。デバイスの初期化時間は、350ms (ノイズ耐性が「中」の場合)、または 1000ms (ノイズ耐性が「高」の場合) です。

13. (動作モードにある) Device_Stat レジスタの「工場出荷時のデフォルト ロード」ビットを読み出します。

    工場出荷時のデフォルト ロード: リセットするたびに、デバイスがフラッシュの値を RAM にロードし、フラッシュの破損がないことを保証するために、RAM のチェックサムとフラッシュのチェックサムを検証します。それらのチェックサムが異なると、デバイスはフラッシュの破損が発生したと認識し、RAM にある工場出荷時のデフォルト値をロードして「工場出荷時のデフォルト ロード」ビットをセットします。これにより、以前ホストによって変更されたレジスタの値が全てリセットされます。各レジスタの工場出荷時のデフォルト値はレジスタ マップにあります。

    工場出荷時のデフォルト値がロードされると、デバイスの I^2C アドレスも、ホストが設定した当時のアドレスからデフォルトのアドレス (37h) に変更されます。工場出荷時のデフォルト値がロードされた後、ホストは I^2C バス上で I^2C のデフォルト アドレスを使用して、CY8CMBR2110 と通信しなければなりません。

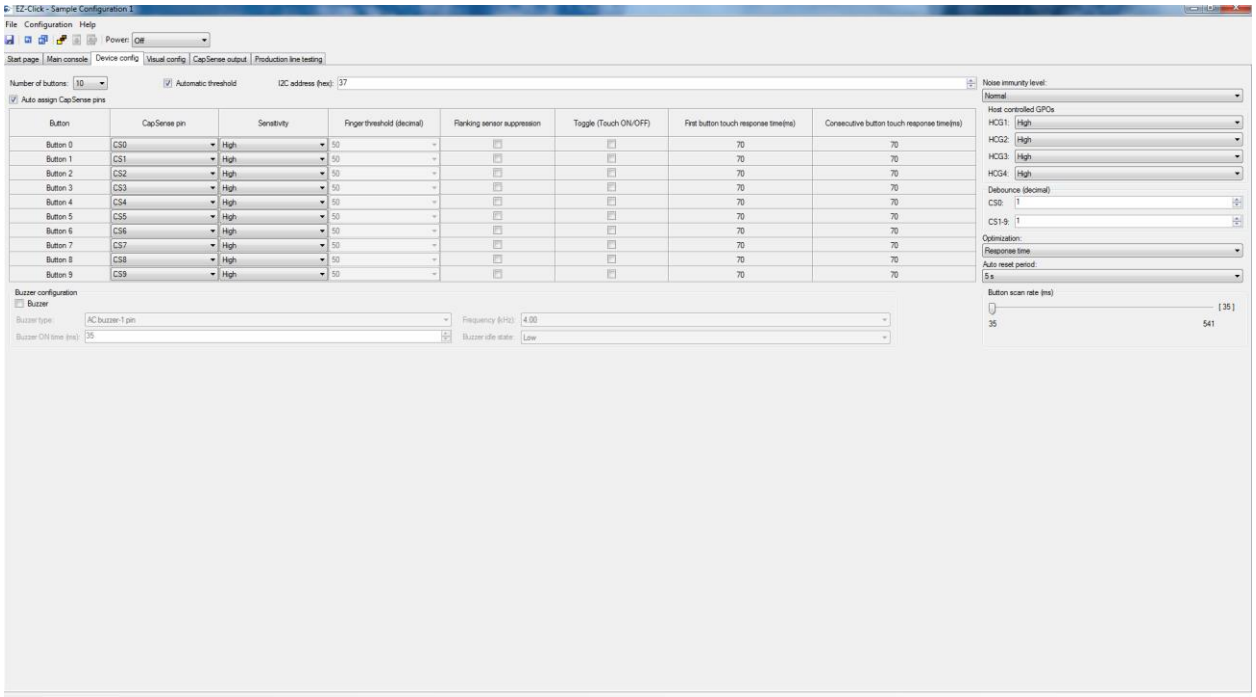
14. 「工場出荷時のデフォルト ロード」ビットがセットされていると、フラッシュが破損しているので、ホストはデバイスを最初から再設定する必要があります。このビットがクリアされている場合は、デバイスが正常に設定されたことを意味します。

注: この手順に言及される異なるモードとレジスタの詳細は CY8CMBR2110 データシートに記載されています。

3.4.1 EZ-Click カスタマイザ ツール

EZ-Click カスタマイザ ツールは、デバイスを設定するための簡単かつ直観的なグラフィカル ユーザー インターフェースです。必要なすべてのパラメーターを I²C インターフェースを使用してデバイスを設定します。

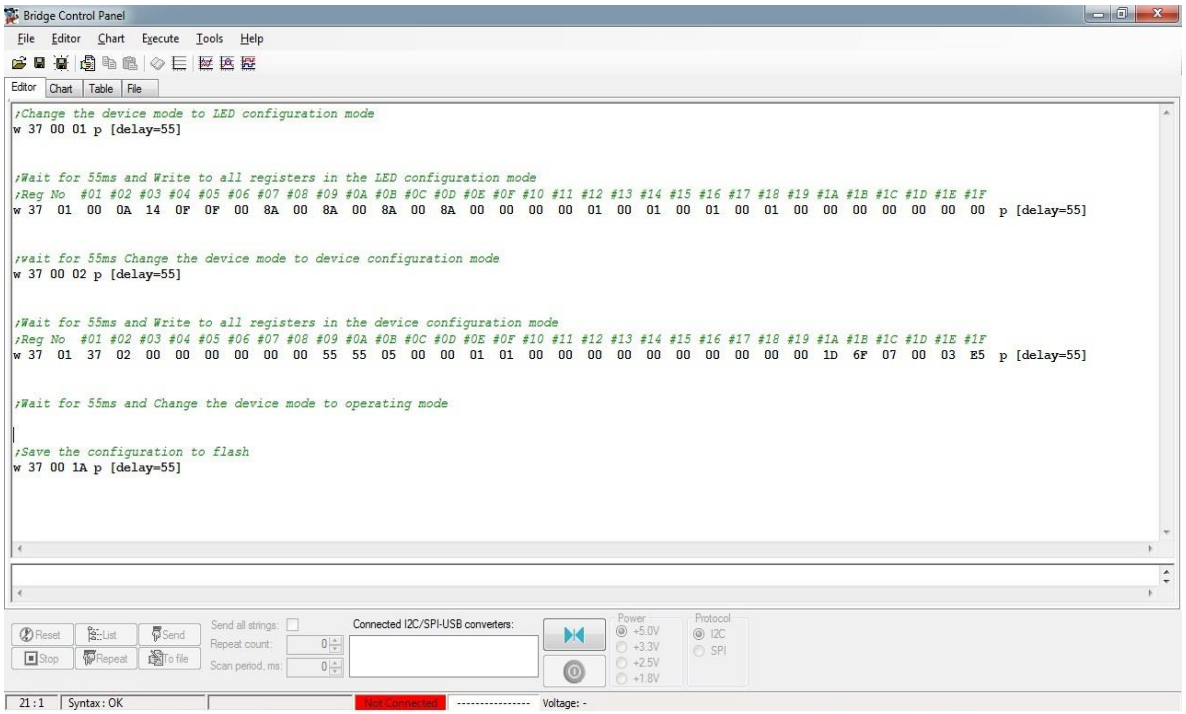
図 3-32: EZ-Click カスタマイザ ツール



EZ-Clickカスタマイザ ツールはデバイスのCapSenseデータをリアルタイムで表示します。CapSenseボタンの状態、C_P、rawカウント、指閾値、SNRを含むボタン固有およびパラメーター固有の両方のデータが参照できます。このツールは、システム診断プログラムの結果およびCapSenseボタンのSNRを表示し、そのSNRが要求を満たしたかどうかを示しているため、製造ラインのテストに使用できます。このツールの詳細はEZ-Click User Guideを参照してください。

その設定を保存して他のサンプルに使用することができます。さらに、このツールは必要なI²C命令を含む設定ファイルを生成して使用することで、デバイスを設定するためにも使用できます。そうするためには、ブリッジ コントロール パネルのコンフィギュレーション ファイルを開いて、USB-I²C リッジ (CY3240-I2USB Bridge) を使ってコマンドをデバイスに送信します (ブリッジ コントロール パネルの詳細はAN2397 - CapSense Data Viewing Toolsを参照してください)。図3-33にコンフィギュレーション ファイルの例を示します。

図 3-33. EZ-Click カスタマイザ ツールが生成したコンフィギュレーション ファイル例



3.4.2 ホスト プロセッサによるデバイス設定

ホスト プロセッサを使用して CY8CMBR2110 デバイスを設定する場合、特定の順序でホスト プロセッサから呼び出す包括的な API のリストがあります。これらの API は、I²C 通信を使用して、CapSense データの読み出し、ホスト コントロール GPO の駆動、製造ライン テストの実施、消費電力の設定などのデバイス機能を設定します。ソースコードは http://www.cypress.com/?rID=74590 からダウンロードできます。

ホスト プロセッサを使用して CY8CMBR2110 デバイスを設定すると以下の利点があります。

- インシステム コンフィギュレーション ― デバイス (チップ) を基板から外す必要がない。
- 実行時コンフィギュレーション ― ホスト プロセッサにより動的に機能が修正できる。

API は主に高レベル API と低レベル API に分けられます。高レベル API は、ハードウェア (プラットフォーム) に依存せず、すべてのホスト プロセッサで動作できます。低レベル API は、CY8C29466-24PXI デバイス向けに開発され、ハードウェア (プラットフォーム) に依存します。アプリケーションに異なるホスト プロセッサがあれば、低レベル API ファームウェアを修正する必要があります。

3.4.2.1 高レベル API

高レベル API は、ボタン タッチ LED 輝度の有効化／無効化、指閾値パラメーター設定、スキャン速度設定、I²C アドレス変更、および他の多くの機能の実行のために使用されます。

高レベルの API は、CY8CMBR2110 の該当するレジスタの読み出し／書き込みのコード、および各設定のチェックサムの計算用のコードを格納します。また、ホスト プロセッサ特有の低レベル API を呼び出し、ホスト プロセッサとデバイス間の物理的な I²C 通信を作成します。

高レベル API ヘッダ ファイル (High_Level_API.h) は高レベル API 関数プロトタイプをすべて含みます。このヘッダ ファイルは、CY8CMBR2110 デバイスの設定時に必須である.C ファイルに含む必要があります。高レベル API は、内部設定のために High_Level_API.h に定義されているマクロを使用します。マクロを変更してはいけません。

以下のような場合です。

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

3.4.2.2 低レベル API

低レベル API は、デバイスとの物理的な I²C 通信を有効化するためにホスト プロセッサで使用されます。ここでの低レベル API は、PSoC I2CHW ユーザー モジュールを使用して、読み出しと書き込みの動作を行います。I²C プロトコルの実装方法に従って、低レベル API コードを修正する場合もあります。

低レベル API ヘッダ ファイル (Low_Level_API.h) は、低レベル API のための関数プロトタイプ、および低レベル API が使用するマクロを含みます。マクロは、主に I²C 通信とソフトウェア遅延ルーチンのために使用されます。これらのマクロは CY8C29466-24PXI デバイス用に定義されます。ホストの I²C 実装に適用するには、その定義を修正する必要があります。

例えば、CY8CMBR2110 デバイスが NACK を返す場合、CY8C29466-24PXI (PSoC1) の I2CHW ユーザー モジュールは 0x00 を返します。従って、マクロ **I2C_NACK** は 0x00 として定義されます。NACK 時に違う値を返す別のホスト プロセッサを使用している場合、それに合わせるように **I2C_NACK** を適切に修正する必要があります。

ソフトウェア遅延 API はボタン スキャン速度に等しい遅延を提供するために必要です。この遅延は書き込み命令の実行後に必要です。この遅延をハードウェア リソース (タイマー) で実装する場合は、表 3-7 に示すとおり、対応するマクロをクリアしてソフトウェア遅延ルーチンを無効にすることができます。

ホスト コントローラーに依存しないマクロは表 3-6 に示されます。使用しているホスト コントローラーにより修正が必要なマクロは表 3-7 に示されます。

表 3-6. ホスト コントローラーに依存しないマクロ

マクロ名	使用法
FLASH_WRITE_TIME	「フラッシュに保存」コマンドが実行された後、CY8CMBR2110 デバイスがデータを適切に保存するために必要な時間です。
TOTAL_BUTTON_COUNT	CY8CMBR2110 デバイスの最大ボタン数です。
FACTORY_DEFAULT_CHECKSUM	CY8CMBR2110 デバイス工場出荷時のデフォルト チェックサムです。
DEFAULT_SLAVE_ADDRESS	CY8CMBR2110 デバイス工場出荷時のデフォルト I²C アドレスです。
DELAY_CONST	ソフトウェア遅延に必要な反復回数を計算するために使用されます。
SLAVE_NACK	CY8CMBR2110 デバイスが NACK を返す場合に I²C フラグをクリアするために使用されます。
SLAVE_ACK	CY8CMBR2110 デバイスが ACK を返す場合に I²C フラグをセットするために使用されます。
SLAVE_BUF_PTR	ホスト I²C バッファ ポインタをレジスタ マップ内の特定のレジスタ アドレスに設定するために使用されます。

表 3-7. ホスト コントローラーに依存するマクロ

マクロ名	使用法
I2C_WRITE_COMPLETE	CY8CMBR2110 デバイスへの I2C 書き込み動作が完了したかを確認します。書き込み動作が完了した場合、I2CHW ユーザー モジュールは 0x50 値を返します。
NACK_RETRY_LIMIT	CY8CMBR2110 デバイスが NACK を返す場合にホスト プロセッサが再試行する回数を定義します。標準値は 20 です。この値は、使用しているアプリケーションに適合するように変更できます。
DELAY_ROUTINE_USED	ソフトウェア遅延ルーチンを有効／無効にするために使用されます。「1」はソフトウェア遅延を有効にし、「0」は無効にします。**ハードウェア リソースを使用してその遅延を実装する場合は、ソフトウェア遅延ルーチンを無効にしてください。** **注:** ソフトウェア遅延ルーチンはブロッキング コードです。CPU を一定時間停止します。
I2C_NACK	CY8CMBR2110 デバイスが現在の I²C 動作を NACK したかどうかを確認するために使用されます。書き込み／読み出し動作が NACK の場合、I2CHW ユーザー モジュールは 0x00 値を返します。
I2C_READ_COMPLETE	CY8CMBR2110 デバイスへの I²C 書き込み動作が完了したかどうかを確認します。書き込み動作が完了した場合、I2CHW ユーザー モジュールは 0x15 値を返します。
NEW_SLAVE_ADDRESS	新しいスレーブ アドレス値です。ホストが MBR_SetI2CSlaveAddress API を使用して、CY8CMBR2110 デバイスのデフォルトのスレーブ アドレスを変更すると、その新しいスレーブ アドレスに対してこのマクロを再定義する必要があります。
CLOCK_FREQUENCY	ホスト コントローラーのクロック周波数 (単位: MHz) です。PSoC 1 ホスト デバイスの場合、クロック周波数は 24MHz です。
MACHINE_CYCLES	ソフトウェア遅延ルーチンの while ループを実行するために必要なマシン サイクル数です。ImageCraft コンパイラでビルドする場合、MACHINE_CYCLES 値は 97 です (MBR_Delay を参照してください)。

3.4.2.3 MBR_WriteBytes

この API は、CY8CMBR2110 デバイスとホスト プロセッサの間の I²C 書き込み動作を開始します。関数プロトタイプは 7.2.2 節に記述されています。

注: 書き込み動作の場合は、ホストで定義されるバッファがあります。高レベル API は、バイト配列の形式であるそのバッファを書き込み API に渡します (データ タイプを参照してください)。書き込むときには、最初のバイト (byte[0]) にベースのポインタを格納し、残りのバイト (byte[1]、byte[2]…) にデータを格納します。ベースのポインタが「CY8CMBR2110 のレジスタ マップで書き込まれるロケーション」にセットされるため、書き込み動作はそのロケーションから開始します。

高レベル API は I²C バッファのポインタおよび書き込まれるバイト数を渡します。MBR_WriteBytes は以下のとおり実行します。

1. CY8CMBR2110 デバイスへの I²C 書き込み動作を開始します。
2. そのトランザクションが完了するまで待機します。
3. そのトランザクションが正常に動作したかを確認します。
4. そのトランザクションが正常に動作しないと、マクロの最大 NACK_RETRY_LIMIT 値までその書き込み動作を再試行します。

3.4.2.4 MBR_ReadBytes

この API は、CY8CMBR2110 デバイスとホスト プロセッサの間の I²C 読み出し動作を開始します。関数プロトタイプは 7.2.2 節に記述されています。

注: byte[0]が 0x00 ロケーションのデータを格納し、byte[1]が 0x01 ロケーションのデータを格納する等により、読み出し時にホストのバッファは、デバイス レジスタ マップの 0x00 ロケーションからの要求データに更新されます。読み出し動作は常にすべてのレジスタ マップの 0x00 ロケーションから開始します。

高レベル API は I²C バッファと読み出されるバイト数を渡します。MBR_ReadBytes は以下のとおり実行します。

1. デバイスから読み込まれる I²C バッファ アドレスとバイト数を取り出します。
2. スレーブ ポインタを 0x00 ロケーションに設定します。
3. CY8CMBR2110 デバイスからの I²C 読み出し動作を開始します。
4. そのトランザクションが完了するまで待機します。
5. そのトランザクションが正常に動作したかを確認します。
6. そのトランザクションが正常に動作しないと、マクロの最大 NACK_RETRY_LIMIT 値までその読み出し動作を再試行します。

3.4.2.5 MBR_Delay

この API は、特定の回数で実行する while ループを使用してソフトウェア遅延を生成します。関数プロトタイプは 7.2.2 節に記述されています。ループの反復回数は次の式に従って計算されます。

$$\textit{ループ回数} = \frac{\text{必要な遅延時間(ms)} \times \text{クロック周波数(MHz)} \times 1000}{\text{while ループの実行に必要なマシン サイクル}} \quad \text{式 5}$$

ホスト マシンで while ループを実行するために必要なマシン サイクル数 (アセンブリレベルの命令サイクルの総数) を計算する必要があります。ImageCraft Pro コンパイラを使用する PSoC 1 のホストの場合、マクロ MACHINE_CYCLES は 97 です。使用しているコンパイラとホスト プロセッサに基づいてこの値を修正する必要があります。

注: 遅延時間中には CPU が完全にブロックされます。

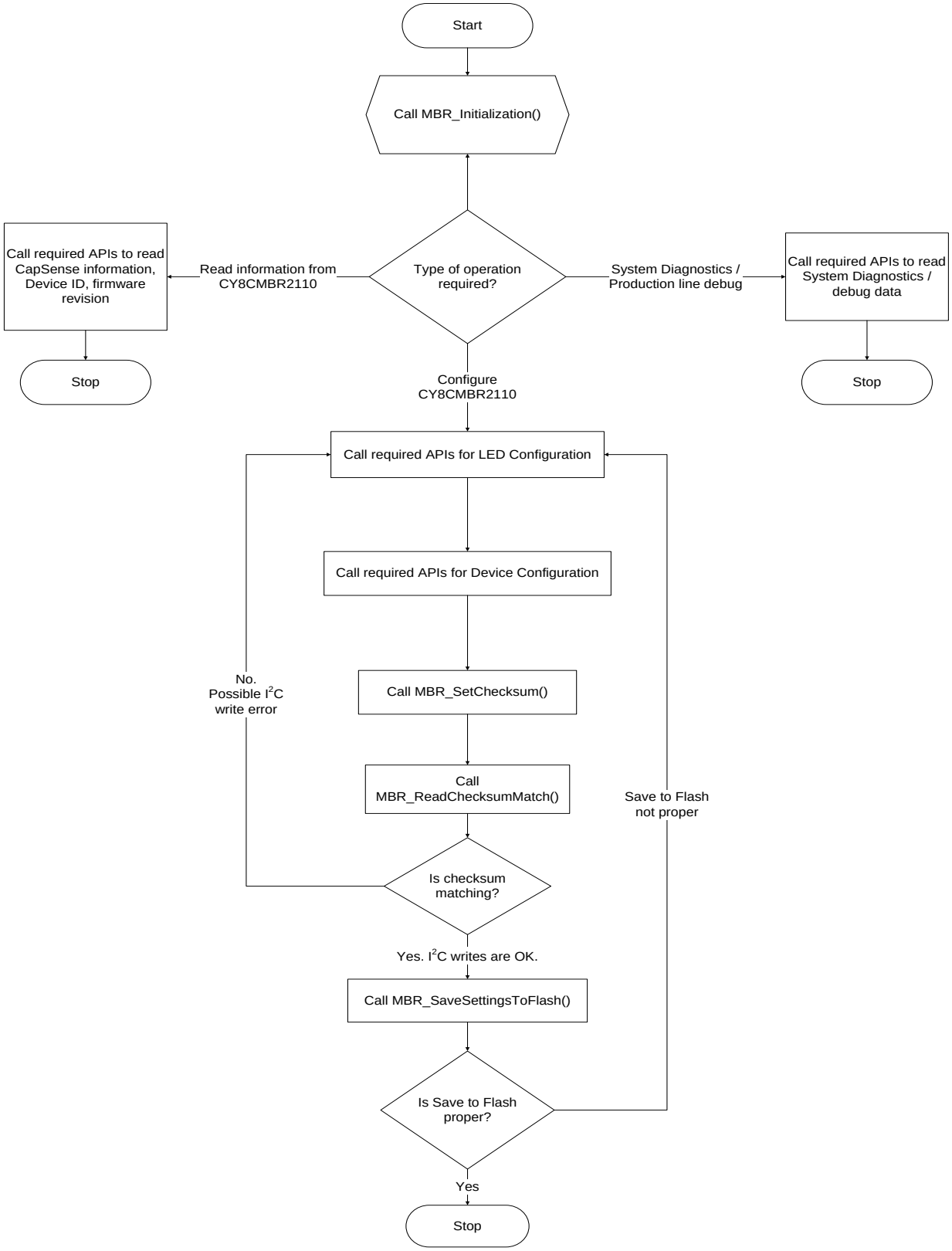
3.4.2.6 CY8CMBR2110 デバイスの設定方法

- CY8CMBR2110 デバイスを設定するときには、高レベル API が特定の順序で呼び出される必要があります。図 3-34 に正しい順序を示します。
- MBR_Initialization API を呼び出した後、ホスト プロセッサの I²C 通信の状態を確認します。この API は、他の API 呼び出しの前に呼び出す必要があります。例えば、トランザクションが ACK を受けた場合、低レベル API の「gbI2CFlag」変数が 1 にセットされ、そうでない場合は 0 にセットされます。適切なトランザクションのためにこの変数を確認できます。また、NACK または ACK の確認のために、自身のホスト プロセッサの I²C レジスタを確認します。
- レジスタ マップのモードのための全機能の設定が完了するまで、レジスタ マップのモード間を切り換えないでください。例えば、LED コンフィギュレーション モードの 1 つの機能を設定してから、デバイス コンフィギュレーション モードに移行し

て、再び LED コンフィギュレーション モードの機能設定に戻らないでください。レジスタ マップのモード間の切り替えには時間がかかります。そのため、すべての機能の設定を LED コンフィギュレーション モードで実行した後で、デバイス コンフィギュレーション モードに移行します。

- CY8CMBR2110 コンフィグレーション用の API に定義している通りに、正しい引数を高レベル API に渡します。
- 高レベル API が設定のチェックサムを計算するため、チェックサムの計算を考慮する必要はありません。
- 「フラッシュに保存」コマンドによりソフトウェア リセットが発生すると、ホストが制御する GPO コンフィギュレーションがクリアされるため、ホストが制御する GPO の設定は、「フラッシュに保存」コマンドの実行後にしてください。
- LED エフェクトは、GPO0 を除いた GPO のグループ (GPO123、GPO456、GPO789) で定義されます。設定は、グループ内の全ての GPO で一致しなければなりません。例えば、GPO1 と GPO2 に異なる LED コンフィギュレーションを渡してはいけません。同じレジスタを共有しているため、GPO1 を設定した後、そのコンフィギュレーションは GPO2 と GPO3 にも適用されます。同様に、GPO2 に対して異なる LED エフェクトを設定すると、そのコンフィギュレーションは GPO1 と GPO3 に適用されます。
- ボタンの「指閾値」を設定する場合は、MBR_SetAdaptiveThreshold API を使って自動閾値機能をクリア (無効化) します (高レベル API を参照してください)。
- LED エフェクトを使用する場合は、エフェクトの機能を設定する前にそのエフェクトを有効にます。例えば、ボタンタッチ LED エフェクトを有効にしてから、そのボタン タッチ LED エフェクトに該当するすべての機能を設定します。
- ディープ スリープ モードに関する API はディープ スリープ モードでの手順に従って呼び出してください。
- LED オン時間の設定、および トグル オン／オフの有効化は実行しないでください。トグル オン／オフが有効な場合、LED オン時間は無効になります。
- LED オン時間の設定、およびボタン タッチ LED エフェクトの有効化を実行しないでください。ボタン タッチ LED エフェクトを有効にすると、LED オン時間は無効になります。
- トグル オン／オフおよび最終ボタンの LED エフェクトを有効にしないでください。トグル オン／オフを有効にすると、最終ボタン LED エフェクトは無効になります。
- システム診断、センサーの現在状態、センサー ラッチ状態、センサーSNR、およびデバッグ データなどのすべての読み出し API は、フラッシュに保存せずに直接呼び出すことができます。

図 3-34: 高レベル API のフロー チャート



3.4.2.7 入力ヘッダ

Inputs.h には、高レベル API 入力用のマクロ定義があります。これらのマクロは、高レベル API に引数を渡すときに使ってください。例えば、1 つの機能を有効にするときに、引数として FEATURE_ENABLE マクロを渡します。幾つかの高レベル API の入力にはマクロがありません。例えば、MBR_SetScanRate() API の入力にはマクロの定義がありません。よって、その関数のパラメーターへの入力として、0～31 の 10 進値を渡す必要があります。パラメーターを渡す前に、CY8CMBR2110 コンフィグレーション用の API の節で、個々の高レベル API の関数プロトタイプを参照してください。これらのマクロの定義を変更しないでください。これらのマクロを使用すると、適切なパラメータを高レベル API に渡すのに役立ちます。

注: 各高レベル API のヘッダには、引数として渡すことができるマクロも一覧表示されています。

3.4.2.8 データ タイプ

それぞれのデータの型に対して割当てるメモリ量はコンパイラに依存します。char、 int および long 型のデータは型定義に属して、高レベル API がそれらを使用して CY8CMBR2110 デバイスを設定します。データの型は下記のとおりです。

- 符号なし char は BOOL 型に定義
- 符号なし char は BYTE 型に定義
- 符号なし int は WORD 型に定義
- 符号なし long は DWORD 型に定義
- 符号付き char は CHAR 型に定義
- 符号付き int は INT 型に定義
- 符号付き long は LONG 型に定義

これらの値は、char、int および long 型データがそれぞれ 8、16、および 32 ビットのメモリ空間を占めるという仮定に基づいています。使用しているホスト コンパイラに対してこれらの仮定が適切でない場合は、Low_Level_API.h および High_Level_API.h の型定義を修正してください。

3.4.2.9 プロジェクトの例

このサンプル プロジェクトを作成する目的は、http://www.cypress.com/?rID=74590 からダウンロードできる CY8C29466-24PXI (PSoC) をホスト デバイスとして CY8CMBR2110 デバイスを設定するためです。このコードは、CY3210-PSoC-EVAL-kit の PSoC Designer 5.2 および ImageCraft コンパイラで実装されます。サンプル コードは次の機能を設定します。

1. 動作中のセンサー数 (システム診断に合格した有効なセンサー) を読み出します。
2. 全ての GPO に対して、並行のパワーオン LED エフェクトを 600ms のランプアップ／ランプダウン時間で有効にします。
3. 600ms の高輝度時間を有効にして、パワーオン LED エフェクトの輝度レベルを 80% (GPO0、GPO123 の場合)、20% (GPO456 の場合)、100% (GPO789 の場合) にします。
4. パワーオン LED エフェクトの場合には、GPO0 に対して反復回数を 1 に設定します。
5. AC-1 ピンのブザーを LOW のアイドル状態 (ブザー周波数 4KHz およびブザー期間 200ms) で設定します。
6. CS0 ボタンに対してデバウンス値を 100 (連続したボタン タッチの応答時間が 1225ms) に設定します。
7. CS0 ボタンに対して感度値を 2 (中) に設定します。
8. CS0 ボタンにトグル機能を有効にします。
9. すべてのボタンの FSS 機能を有効にします。
10. ホストが計算したチェックサムを CY8CMBR2110 デバイスに書き込みます。
11. チェックサム一致条件を検証します。
12. チェックサム一致条件が真の場合、すべての設定をフラッシュに保存します。
13. HGPO1 状態を HIGH にセットします。

注: HGPO1 は、「フラッシュに保存」が完了した後、HIGH にセットされます。次のリセット時に、HGPO1 は LOW にクリアされます。パワーオン LED エフェクトを確認したい場合、デバイスに対して HGPO1 をクリアするハードウェアリセットを実施する必要があります。

3.4.3 サードパーティ プログラマ

数多くのデバイスを設定する場合は、弊社はサードパーティ ベンダーがデバイスの自動プログラミングを実施するよう推奨します。このために、EZ-Click カスタマイザ ツールで生成された 16 進形式のファイルを Hilo systems 社 (サードパーティプログラマの一つ) に提供しなければなりません。

詳しくは http://www.hilosystems.com.tw/en/index.aspx にお問い合わせください。

3.5 CY8CMBR2110 のリセット

CY8CMBR2110 は、ハードウェアまたはソフトウェアによりリセットできます。

3.5.1 ハードウェア リセット

ハードウェア リセットの場合、LED コンフィギュレーション モードとデバイス コンフィグレーション モードのレジスタ値はフラッシュから RAM にロードされます。すべてのデバイス ブロックが初期化され、システム診断が実施され、最初の 5ms のパルスが CSx の立ち下がりで GPOx 上に送信されます。ノイズ耐性が「中」であれば、これは 350ms 内に完了しますが、ノイズ耐性が「高」であれば、これは 1000ms 内に完了します。パワーオン LED エフェクトが有効な場合、その後に残りの全ての GPO 上に確認できます。LED エフェクトの後、デバイスは動作モードになり、通常の動作を開始します。

ハードウェア リセットは、電源または XRES により、CY8CMBR2110 のピンをパワーオンになるようトグルすることで実行されます。

3.5.1.1 電源リセット

電源リセットの場合、デバイスの V_{DD} への外部電源をオフにして、V_{DD} が 100mV よりもドロップしたことを確認してから電源を再びオンにします。電源リセットの場合では、16ms の HIGH に駆動しているパルスが HostControlGPO1 ピンで確認できます。

3.5.1.2 XRES リセット

XRES リセットの場合、デバイスの XRES ピンを HIGH にしてから LOW にします。XRES リセットの場合では、パルスは HostControlGPO1 ピンで確認できません。

3.5.2 ソフトウェア リセット

ソフトウェア リセットは、Host_Mode レジスタの「ソフトウェア リセット」ビットに 1 を書き込むことで実行されます (動作モードにある場合)。ソフトウェア リセットの場合、LED コンフィギュレーション モードとデバイス コンフィギュレーション モードのレジスタ値はフラッシュから RAM にロードされます。デバイスは「ソフトウェア リセット」ビットを自動的にクリアし、デバイスのすべてのブロックは初期化されます。ノイズ耐性が「中」であれば、これは 350ms 内に完了しますが、ノイズ耐性が「高」であれば、これは 1000ms 内に完了します。デバイスは動作モードにあり、通常の動作を開始します。システム診断は行われず、パワーオン LED エフェクトも起こりません。ユーザーがデバイスをパワーオン LED エフェクトに設定してコンフィギュレーションをフラッシュに保存すると、パワーオン LED エフェクトを確認できるようにハードウェア リセットを実行しなければなりません。

4. 電気的／機械的設計時の考慮事項

静電容量式タッチセンサー技術をアプリケーションで設計する際に、CapSense デバイスがより大きな構造物に存在していることを意識することが重要となります。プリント基板レイアウト、ユーザー インターフェースあるいはエンドユーザー操作環境等の詳細にわたり注意を払うことで、堅牢で信頼性の高いシステム性能が達成できます。詳細は「Getting Started withCapSense」を参照してください。

4.1 オーバーレイの選択

「CapSense 回路デザイン」では、式 1 は指の静電容量を示しています。

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

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=指とボタンパッドが重なる面積

D=オーバーレイの厚さ

CapSense の信号強度を高めるには、より誘電率が高いオーバーレイ素材を使用し、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。CapSense 回路デザインの章で説明したように、デザイン ツールボックスは堅牢で信頼性の高い CY8CMBR2110 ソリューションを設計する手助けをします。

表 4-1. オーバーレイ素材の絶縁耐圧

素材	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	1200～2800	10
乾燥した木材	3900	3
ガラス (一般的)	7900	1.5
ガラス (ホウケイ酸ガラス (Pyrex®))	13,000	0.9
PMMA プラスチック (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 フィルム (Mylar®)	280,000	0.04
ポリイミド フィルム (Kapton®)	290,000	0.04

導電材料は電界パターンと干渉するため、オーバーレイとしては使用できません。このため金属粒子が含まれる塗料を使用しないでください。

オーバーレイを PCB に固着
空気の絶縁定数は非常に小さいため、オーバーレイとボタンの間に隙間があると、ボタンの性能が低下します。隙間を無くすために、非導電性接着剤でオーバーレイを CapSense PCB に接着します。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。この接着剤は裏に紙がつ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 および 468MP)。

4.2 ESD 保護

高い ESD 許容度は、入念なシステムデザインにより生まれます。特にユーザー インターフェースなどの最終製品の接触放電について検討することにより、CapSense コントローラーに損傷を与えずに 18kV の放電現象に耐えることができます。

CapSense コントローラー ピンは直接的な 12kV 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 4-1 に、IEC 61000-4-2 で指定されるような 12kV の放電から CapSense ボタンを保護するのに必要な様々なオーバーレイ素材の厚さを示します。オーバーレイの素材が十分な ESD 保護を提供されない場合、対応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。

4.2.1 防止

接触面のすべての経路の絶縁破壊電圧は、接触による潜在的な高電圧よりも高いことを確認してください。また、CapSense コントローラーと想定 ESD 源との間に適切な距離を保てるようシステムを設計します。適正な距離を保つことが難しい場合は、ESD 源と CapSense コントローラーとの間に高い絶縁破壊電圧をもった保護層を置きます。例えば、厚さ 5mil の Kapton®テープは 18kV まで耐えられます。

4.2.2 リダイレクト

製品が高密度な場合は、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。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の周囲にハッチング グランド面を施すことによりボタンおよび 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。

4.2.3 クランプ

CapSense ボタンは意図的に接触面に近接して配置されているため、放電経路の転向は現実的でない場合もあります。この場合、直列抵抗や専用の ESD 保護デバイスを使用するのが適切です。

推奨する直列抵抗値は 560Ω です。

より効果的な方法は、専用の 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である必要があることに注意してください。表 4-2 は、CapSense コントローラーとの使用に推奨するデバイスの一覧です。

表 4-2. 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	製品型番				
Littelfuse	SP723	5pF	2nA	8kV	15kV
Vishay	VBUS05L1-DD1	0.3pF	0.1µA	±15kV	±16kV
NXP	NUP1301	0.75pF	30nA	8kV	15kV

4.3 EMC (電磁環境適合性) の注意点

4.3.1 放射性干渉

輻射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。干渉は、CapSense ボタンの配線、その他のデジタルまたはアナログ入力を通じて、PCB レベルで CY8CMBR2110 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 ガイドラインを以下に記します。

- **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 入力ラインで推奨する直列抵抗値は 560Ω です。
- **配線長**: 可能な限り配線を短くします。
- **電流ループ領域**: 電流の帰路を短くします。寄生容量の影響を低減させるために、ベタ グランドの代わりにハッチグランドをボタンと配線から 1cm 以内に配置します。
- **RF 源の位置**: LCD インバーターおよびスイッチトモード電源 (SMPS) のようなノイズ源を CapSense 入力から隔てるために隔離手段をとります。電源をシールドすることも干渉防止の一般的な技術です。

4.3.2 伝導耐性およびエミッション

他のシステムとの相互接続を通じてシステムに入ったノイズは伝導ノイズと呼ばれます。例えば、電源や通信ライン等です。CapSense コントローラーは低消費電力デバイであるため、伝導性放射は避けるべきです。以下のガイドラインは伝導性放射を減らし電磁波耐性を高める助けになります。

- データシートで推奨しているようにデカップリング コンデンサを使用します。
- システム電源に接続している入力に双方向フィルタを加えます。このフィルタは、伝導妨害およびイミュニティの両方に効果的です。Π 型フィルタは、電源ノイズが高感度の部分に影響を与えるのを防止でき、また、その部分自体のスイッチング ノイズが電源プレーンにカップリングして戻るのを防ぎます。
- CapSense コントローラーPCB がケーブルで電源に接続されている場合は、ケーブル長を最短にして、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、電源や通信ラインにフェライト ビーズを配置します。

4.4 PCB レイアウト ガイドライン

「一般的なレイアウト ガイドライン」で説明したように、全般的なレイアウト ガイドライン は堅牢な CY8CMBR2110 CapSense PCB レイアウトを設計するのに役立ちます。

GPO を使って電流を CapSense コントローラーに流し込むデザインで、CapSense システムにノイズが多い場合、直列抵抗をすべて GPO に使用してシンク電流を制限します。表 4-3 に示すように、シンク電流の限界はデザインにおける 5V でボタンの最大 C_P 値で決定します。

表 4-3. 低出力電圧に対する GPO シンク電流の上限

ボタンの C_P 範囲	GPO ごとのシンク電流の限界	デバイスのシンク電流の上限
$5pF \le C_P \le 12pF$	25mA	120mA
$12pF \le C_P \le 21pF$	20mA	20mA
$21pF \le C_P \le 40pF$	6mA	6mA

詳細な PCB レイアウト ガイドラインは「Getting Started with CapSense」を参照してください。

5. 低消費電力設計上の考慮事項

5.1 システム設計の推奨事項

サイプレスの CY8CMBR2110 は電池式のアプリケーションの低消費電力要件に対応しています。

消費電力を最小化するには、以下の手順で行います。

- 未使用の Capsense 入力を全て接地します。
- 「Getting Started with CapSense」のデザイン ガイドラインを使用して C_P を最小化します。
- 電源電圧を低くします。
- CSx ボタンの感度を低くします (「感度制御」を参照してください)。
- 消費電力を最適化するようデザインを設定します (「ボタン スキャン速度」を参照してください)。
- 必要な部分にのみ「高」のノイズ耐性レベルを使用します (「ノイズ耐性」を参照してください)。
- ボタン スキャン速度を高めるかディープ スリープ動作モードを使用します (「ボタン スキャン速度」を参照してください)。

5.2 平均消費電力の計算

デザイン ツールボックスは、本節で説明する電力最適化計算を自動で行います。CY8CMBR2110 の平均消費電力は、以下のパラメーターから計算することにより決まります。

- ボタン スキャン速度: T_R
- スキャン時間: T_S
- 非タッチ状態での平均電流: I_{AVE_NT}
- タッチ状態での平均電流: I_{AVE_T}
- アクティブ時間の割合、P
- 平均使用電流: I_{AVE_U}
- 平均電流、I_{AVE}
- 平均消費電力、P_{AVE}

5.2.1 ボタン スキャン速度 (T_R)

CY8CMBR2110 に設定されるレジスタ マップを介してボタン スキャン速度を制御します。レジスタ値に基づいて、オフセットを取得し、定数に追加することで、実際のボタン スキャン速度を取得します。オフセット値の範囲は 0～506ms です。

$$T_R = \textit{ボタン スキャン速度定数} + \textit{ボタン スキャン速度オフセット} \quad \text{式 6}$$

表 3-5 にボタン スキャン速度定数の決め方を示します。

5.2.1.1 応答時間

応答時間とは、デバイスが有効なボタン接触を検出し、GPOx で信号を生成するのにかかる CSx ボタンにタッチする最小時間です。

応答時間は次の式で計算します。

式 7

ノイズ耐性が「中」の場合:

$$RT_{CBT} = \textit{ボタン スキャン速度定数} + \left[\textit{ボタン スキャン速度定数} \times \left\{Round_{down}\left((\textit{デバウンス} - 1)/3\right) + 1\right\}\right]$$

$$RT_{FBT} = \textit{ボタン スキャン速度} + \left[\textit{ボタン スキャン速度定数} \times \left\{Round_{down}\left((\textit{デバウンス} - 1)/3\right) + 1\right\}\right]$$

ノイズ耐性が「高」の場合:

$$RT_{CBT} = \textit{ボタン スキャン速度定数} + \left[\textit{ボタン スキャン速度定数} \times \textit{デバウンス}\right]$$

$$RT_{FBT} = \textit{ボタン スキャン速度} + \left[\textit{ボタン スキャン速度定数} \times \textit{デバウンス}\right]$$

ここで、

RT_{CBT} = 最初のボタン タッチ後の連続したボタン タッチの応答時間

RT_{FBT} = 最初のボタン タッチの応答時間

CS1～CS9 のデバウンス=1～255

CS0 のデバウンス=1～255

$Round_{down}$ は((デバウンス-1) /3)の結果より小さい、または等しい最大の整数値

デザイン設定のノイズ耐性を「中」から「高」レベルに変更する必要がある場合は、応答時間を維持するためにデバウンス値を減らします。

5.2.2 スキャン時間 (TS)

近似値スキャン速度は次の式で計算します。

式 8

ノイズ耐性が「中」の場合:

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

ノイズ耐性が「高」の場合:

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

ここで、

K_{CSX} = CSx のボタン感度定数、表 5-1 より

T_{FW} = ファームウェア実行時間、表 5-2 より

表 5-1. ボタン感度定数

CSx 感度 (pF)	C_P (pF)[7]	ボタン感度定数 (K)
HIGH	GND に接続したボタン	0
	5pF≤C_P≤10pF	1
	10pF<C_P≤22pF	2
	22pF<C_P≤40pF	4
中	GND に接続したボタン	0
	5pF≤C_P≤18pF	1
	18pF<C_P≤38pF	2
	38pF<C_P≤40pF	4
LOW	GND に接続したボタン	0
	5pF≤C_P≤12pF	0.5
	12pF<C_P≤26pF	1
	26pF<C_P≤40pF	2

表 5-2. 平均電流パラメーター

パラメーター	Typ	Max
T_{FW}	6.00ms	6.50ms
T_S	式 7 より	Typ 値から+5%
T_R	式 5 より	Typ 値+10%
I_{SLEEP}	9.52µA	14.2µA
I_{ACTIVE}	3.4mA	4.00mA

[7] C_P の限界は近似値で、±2pF の誤差があります。

5.2.3 非タッチ状態での平均電流 (IAVE_NT)

$$I_{AVE_NT} = \left(\frac{T_R - T_S}{T_R} \times I_{スリープ}\right) + \left(\frac{T_S}{T_R} \times I_{アクティブ}\right)$$ 式 9

ここで、

T_R=ボタン スキャン速度

T_S=スキャン時間

I_{SLEEP}=低消費電力スリープ モードで CY8CMBR2110 が消費する電流、表 5-2 より

I_{ACTIVE}=アクティブ動作モードで CY8CMBR2110 の消費電流、表 5-2 より

スタンバイ モードでの LED 輝度が有効な場合:

$$I_{AVE_NT} = I_{アクティブ}$$

5.2.4 タッチ状態での平均電流 (IAVE_T)

$$I_{AVE_T} = \left(\frac{C_{BS} - T_S}{C_{BS}} \times I_{スリープ}\right) + \left(\frac{T_S}{C_{BS}} \times I_{アクティブ}\right)$$ 式 10

ここで、

T_S=スキャン時間

C_{BS}=ボタン スキャン速度定数、表 3-5 より

I_{SLEEP}=低消費電力スリープ モードで CY8CMBR2110 が消費する電流、表 5-2 より

I_{ACTIVE}=アクティブ動作モードで CY8CMBR2110 の消費電流、表 5-2 より

スタンバイ モードでの LED 輝度が有効な場合:

$$I_{AVE_T} = I_{アクティブ}$$

5.2.5 アクティブ時間の割合 (P)

ボタンにタッチすると、デバイスのアクティブ時間が 1 時間当たりのボタン タッチ回数および次の 3 つの値の最大値を使用することで計算 (単位: ms) されます。

1. ボタン タッチの平均時間
2. ブザー オンの平均時間
3. ボタン タッチ LED エフェクトの平均時間

式 11

$$アクティブ時間 = Max(ボタン\ タッチ時間、ブザー\ オン時間、ボタン\ タッチLEDエフェクト時間) \times (1\ 時間当たりのボタン\ タッチ回数)$$

アクティブ時間の割合:

$$P = \frac{アクティブ時間}{(3600 \times 1000)} \times 100$$ 式 12

この方法で P を計算する場合には、ブザー信号出力またはボタン タッチ LED エフェクトが完了した後他のボタンがタッチされないときに、各ボタンのタッチが発生すると仮定します。そうでない場合は、この計算結果を P の値にすると、消費電力の計算結果が実際の値よりも高くなります。

5.2.6 平均使用電流 (I_{AVE_U})

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

ここで、

P=アクティブ時間の割合

I_{AVG_NT}=非タッチ状態での平均電流

I_{AVG_T}=タッチ状態での平均電流

5.2.7 平均電流 (I_{AVE})

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

ここで、

T_{SA}=デバイスがディープ スリープ モードにない時間

T_{DS}=デバイスがディープ スリープ モードにある時間

5.2.8 平均電力 (P_{AVE})

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

ここで、

I_{AVE}=平均電流

V_{DD}=電源電圧

5.2.9 計算例

平均電力の計算方法の例として、良く設計された 8 個のボタンおよび以下のパラメーターを持つ CapSense ユーザーインターフェースを取り上げます。

- 全 8 個のボタンの C_P は 10～20pF
- 各ボタンの感度は「高」
- 応答時間の最適化した場合のデザイン
- ノイズ耐性は「中」
- ボタン スキャン速度のオフセットは 506ms に設定
- スタンバイ モードでの LED 輝度は無効
- 消費電流の標準値は計測

ボタン スキャン速度定数は表 3-5 から取り出します。

$C_{BS} = 35ms$

ボタン スキャン速度は式 5 で計算します。

$T_R = 35 + 506 = 541ms$

スキャン時間は式 7 で計算します。そのうち、ボタン感度定数は表 5-1 から、ファームウェア実行時間の標準値は表 5-2 から取り出します。

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

非タッチ状態の平均電流は、式 8 および表 5-2 から取り出す I_{SLEEP} と I_{ACTIVE} の最大値で計算します。

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

タッチ状態の平均電流は式 9 で計算します。

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

式 10 を使ってアクティブ時間を計算するには、1 つのボタンが 1 分当たり 1 度のみタッチされる (1 時間当たり 60 回のボタンがタッチされる) と仮定します。平均すると、ボタン タッチ時間が 1000ms、ボタン タッチ LED エフェクトの時間が 3000ms、ブザー出力はありません。

$アクティブ時間 = 3000ms \times 60 = 180s$

アクティブ時間の割合は式 11 で計算されます。

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

デザインの平均電流消費は式 16 で計算されます。

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

このデザインがディープ スリープ モードを使用せず、1.71V で動作していると仮定すると、平均電力は式 14 で計算されます。

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

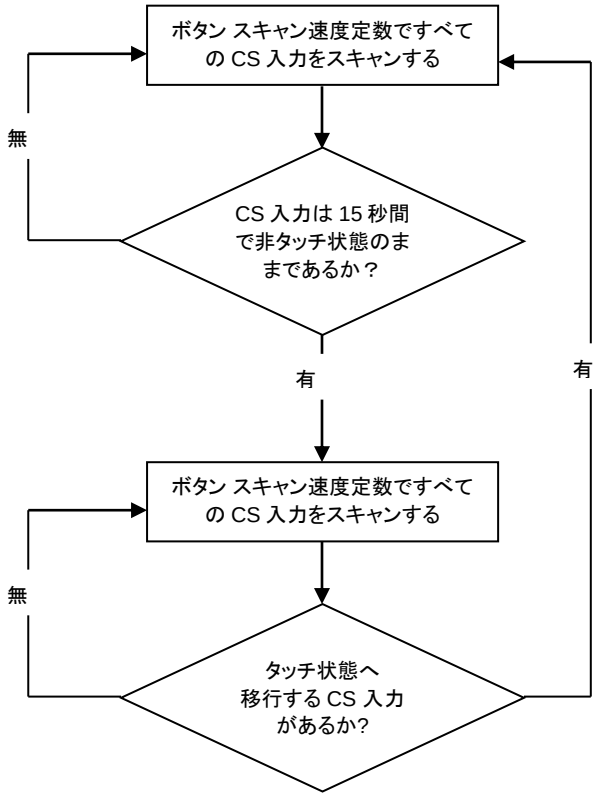
5.3 スリープ モード

サイプレスの CY8CMBR2110 は、低消費電力スリープ モード、またはディープ スリープ モードのいずれかで動作するよう設定することができます。これらのモードでは、デバイスの消費電力が低減します。

5.3.1 低消費電力スリープ モード

低電力スリープ モードでの CY8CMBR2110 コントローラーの動作は 図 5-1 で説明されます。

図 5-1. 低消費電力スリープ モード



5.3.2 ディープ スリープ モード

CY8CMBR2110 をホスト プロセッサの搭載されたシステムで使用する場合、Attention/Sleep ラインはディープ スリープモードでデバイスを動作させることができます。CY8CMBR2110 をディープ スリープ モードへ移行するためには、以下の手順を行ってください。

1. Attention/Sleep ラインを LOW にします。
2. 実行モード中の Host_Mode レジスタの「ディープ スリープ」ビットを 1 にセットします。
3. 50ms 待機します。
4. Attention/Sleep ピンを HIGH にします。

すべての通信は保留されます。ディープ スリープ モードでは、デバイスが約 0.1µA 消費します。デバイスがディープ スリープ モードに移行した後、ディープ スリープ ビットが自動的にクリアされます。これから復帰するためには、ホストが Attention/Sleep ラインを LOW にプルダウンします。復帰した後、CY8CMBR2110 はアクティブ モードに移行します。デバイスを低消費電力スリープモードに移行させるために、ホスト プロセッサは Attention/Sleep ピンを HIGH にします。ディープ スリープ モードから復帰した後、デバイスはボタン スキャンが再開する前にいくらか時間がかかります。この時間は再初期化と呼ばれます。この時間中に、ボタンにタッチしても報告されません。再初期化の時間は、ノイズ耐性が「中」の場合は 20ms、ノイズ耐性が「高」の場合は 50ms です。

6. リソース

6.1 ウェブサイト

Cypress's CapSense Controllers website にアクセスし、本書で説明したすべての参照資料がご覧になれます。

CY8CMBR2110 ウェブページで、様々な技術的リソースを見つけてください。

6.2 データシート

CapSense CY8CMBR2110 デバイス用のデータシートは www.cypress.com で入手できます。

- CY8CMBR2110

6.3 デザイン ツールボックス

対話式のデザイン ツールボックスにより、ユーザーは強固で信頼できる CY8CMBR2110 CapSense ソリューションを設計することができます。

6.4 EZ-Click™ カスタマイザ ツール

対話式の EZ-Click カスタマイザ ツールにより、ユーザーは CY8CMBR2110 CapSense ソリューションを設定することができます。

6.5 デザイン サポート

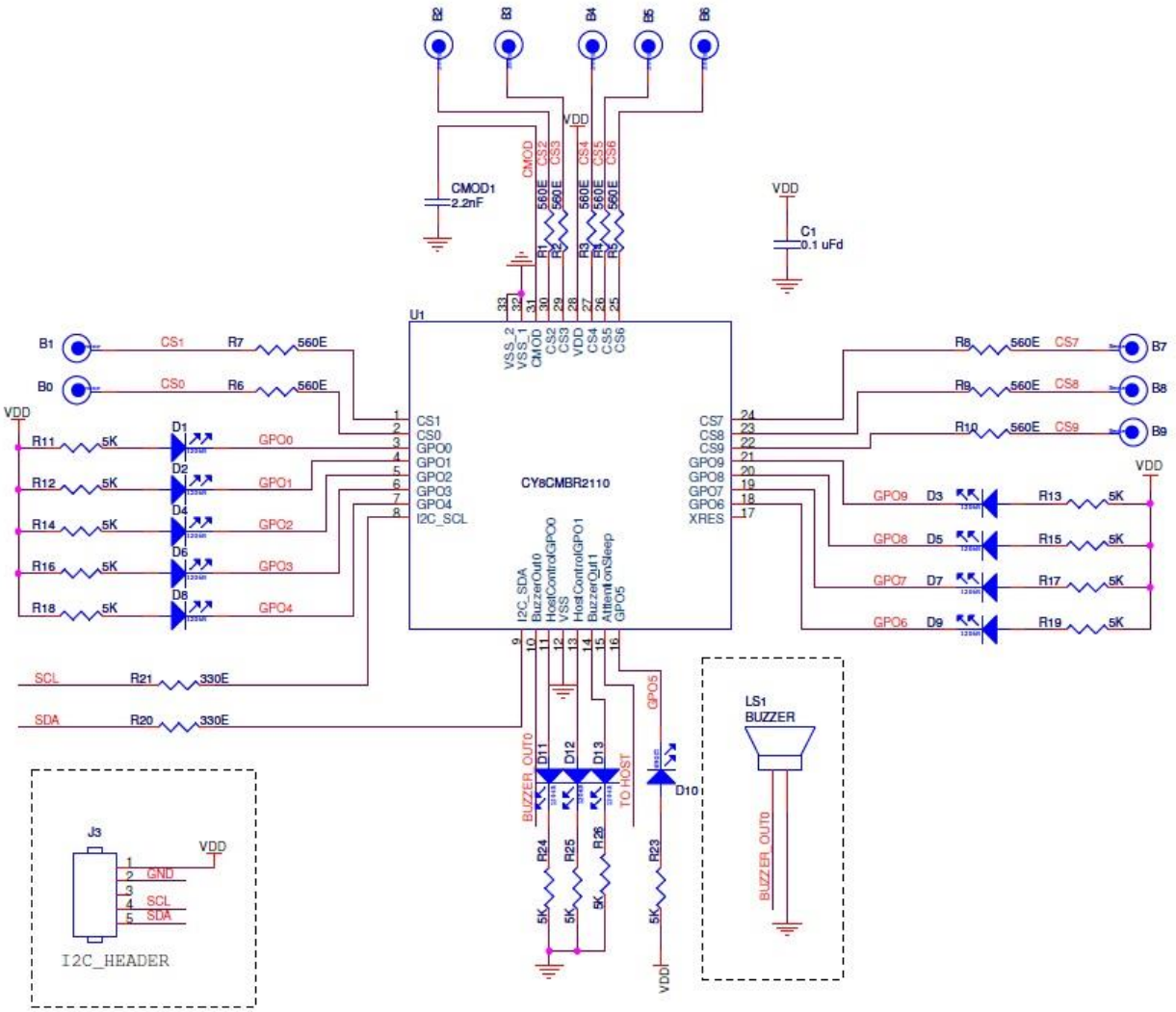
お客様の CapSense ソリューションを確実に成功させるために、サイプレスは以下の様々なデザイン サポート チャネルを持っています。

- 知識ベース記事 – 製品ファミリ別の技術情報記事を閲覧したり、CapSense についての様々なトピックスを検索できます。
- CapSense アプリケーション ノート – 本書で紹介した情報に基づいた幅広いアプリケーション ノートを詳細に調べます。
- ホワイト ペーパー – 最先端の静電容量タッチ インターフェース トピックスについて学べます。
- サイプレス開発コミュニティ – サイプレス技術コミュニティに参加し、情報交換できます。
- CapSense 製品選択ガイド – すべての CapSense 製品ラインを見ることができます。
- 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 – サイプレスは顧客満足を第一に考えます。当社の品質ウェブサイトで、信頼性および製品認定報告書をご覧ください。
- 技術サポート – 世界一流の技術サポートをオンラインで利用することができます。

7. 付録

7.1 参考回路

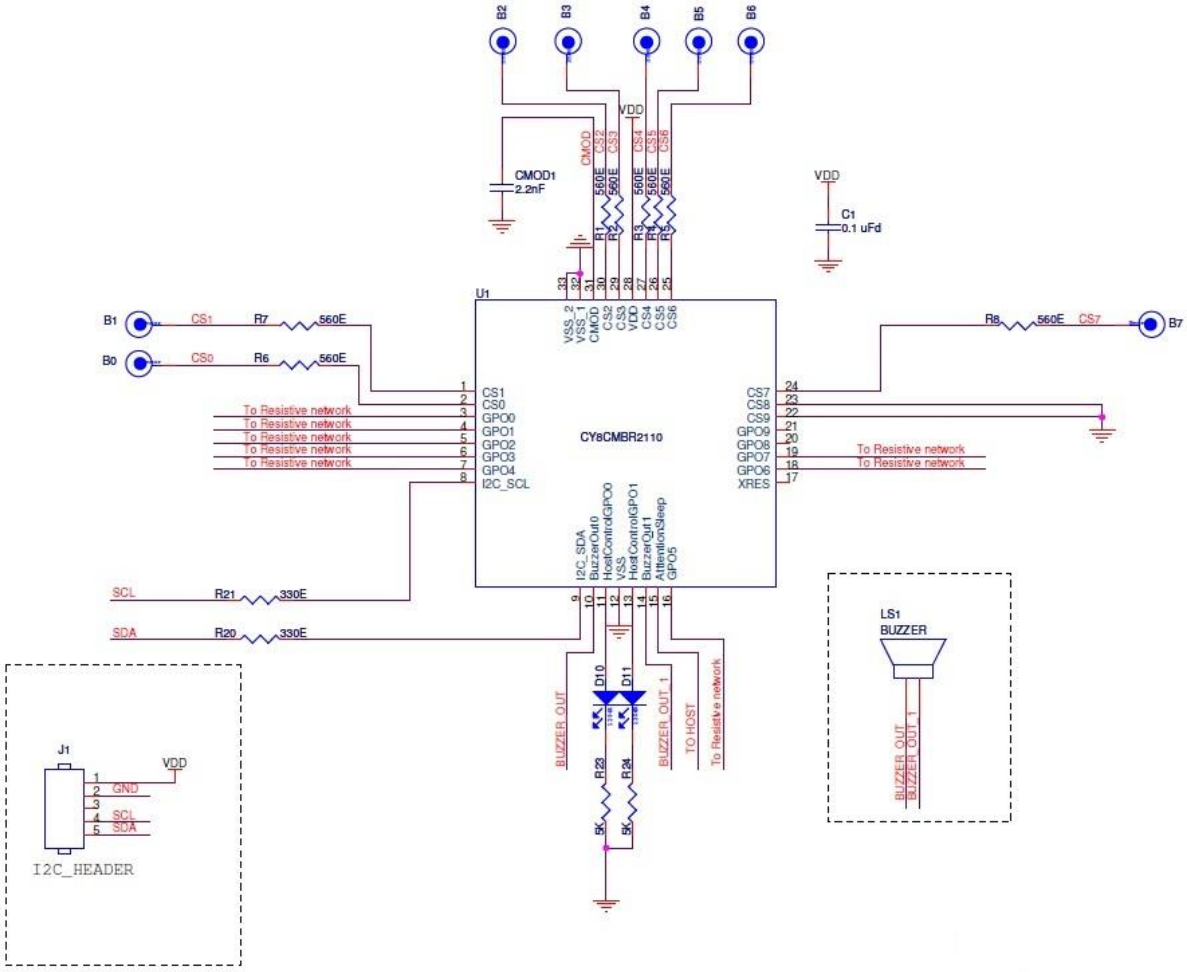
7.1.1 回路図 1: 10 のボタンと 10 の GPO



回路図 1: 10 のボタンと 10 の GPO では、CY8CMBR2110 は以下のように設定されます。

- CS0～CS9 ピン: 560Ω の抵抗を介して CapSense ボタンに接続
  - 10 個の CapSense ボタン (CS0～CS9)
- GPO0～GPO9 ピン: LED があり、5kΩ の抵抗を介して V_{DD} に接続
  - 10 個の LED (GPO0～GPO9) を駆動する CapSense ボタン
- CMOD ピン: 2.2nF のコンデンサを介して接地
  - 変調コンデンサ
- XRES ピン: 開放
  - 外部リセット用
- BuzzerOut0 ピン: ブザーに接続
  - AC ブザー (1 ピン)
  - ブザー セカンド ピンの接地
- BuzzerOut1 ピン: LED があり、5kΩ の抵抗を介して接地
  - ホスト制御 GPO として使用される
- HostControlGPO0、HostControlGPO1: LED があり、5kΩ の抵抗を介して接地
  - 2 個のホスト制御 GPO
- I2C_SDA、I2C_SCL ピン: 330Ω の抵抗を介して I^2C ヘッダに接続
  - I^2C 通信
- Attention/Sleep ピン: ホストに接続
  - I^2C 通信、消費電力、およびデバイス実行モードの制御用

7.1.2 回路図 2: アナログ電圧出力付きの 8 個のボタン



回路図 2: アナログ電圧出力付きの 8 個のボタン 、CY8CMBR2110 は以下のように設定されます。

- CS0～CS7 ピン: 560Ω の抵抗を介して CapSense ボタンに接続。CS8、CS9 ピン: 接地
  - 8 個の CapSense ボタン (CS0～CS7)
  - CS8 と CS9 は設計で使用しない
- GPO0～GPO7 ピン: 外部抵抗ネットワークへ接続
  - アナログ電圧出力用の 8 個の GPO (GPO0～GPO7)
  - GPO8 と GPO9 は設計で使用しない
- CMOD ピン: 2.2nF のコンデンサを介して接地
  - 変調コンデンサ
- XRES ピン: 開放
  - 外部リセット用
- BuzzerOut0、BuzzerOut1 ピン: AC ブザーに接続
  - AC 2 ピンのブザー

- HostControlGPO0、HostControlGPO1 ピン: LED を 5kΩ の抵抗を介して接地
  - □ 2 個のホスト制御 GPO
- I2C_SDA、I2C_SCL ピン: 330Ω の抵抗を介して I^2C ヘッダに接続
  - □ I^2C 通信
- Attention/Sleep ピン: ホストに接続
  - □ I^2C 通信、消費電力、およびデバイス実行モードの制御用

7.2 CY8CMBR2110 コンフィグレーション用の API

以下の表は 72 の高レベル API および 3 つの低レベル API を示します。この表は、I²C インターフェースを介して CY8CMBR2110 (I²C スレーブ) を設定するためにホスト プロセッサ (I²C マスター) で使用されます。これらの高レベル API は、プラットホームから独立しており、どのホスト プロセッサにおいても使用できます。適切な入力は inputs.h ファイルの多くの高レベルの API のマクロとして定義されます。

低レベル API は、プラットフォームから独立しており、デバイスとの物理的な I²C 通信を有効にするためにホスト プロセッサで使用されます。これらの低レベル API は、PSoC 1 ホスト デバイスのために開発されています。そのため、ユーザは低レベル API コードをホスト プロセッサに基づいて変更する必要があるかも知れません。これらの API で作成されたサンプル プロジェクトは、3.4.2.9 の節で説明されます。

7.2.1 高レベル API

	プロトタイプ	**void MBR_Initialization(void);**		
1	**説明**	高レベル API が使用するグローバル変数を初期化します。別の API を呼び出す前に、この API を呼び出す必要があります。		
	パラメーター	なし		
	戻り値	なし		
	例	MBR_Initialization();		
2	**プロトタイプ**	**void MBR_SetCustomData(BYTE bCustomData);**		
	説明	ユーザーが与えたデータをデバイス コンフィグレーション モードでのカスタム データ ストレージ レジスタへ書き込みます。データを永久に格納するために API 関数の **MBR_SaveSettingsToFlash** を呼び出す必要があります。		
	パラメーター	**名前**	**説明**	**可能な値**
		bCustomData	カスタム レジスタに書き込まれるデータ	0～255
	戻り値	なし		
	例	MBR_SetCustomData(200); 値 200 はカスタム データ ストレージ レジスタに格納されます。		
3	**プロトタイプ**	**void MBR_IssueSWReset(void);**		
	説明	ソフトウェア リセットを CY8CMBR2110 デバイスに発行します。ソフトウェア リセットを参照してください。		
	パラメーター	なし		
	戻り値	なし		
	例	MBR_IssueSWReset();		
4	**プロトタイプ**	**WORD MBR_ReadFlashChecksum(void);**		
	説明	CY8CMBR2110 デバイスのフラッシュに格納されるチェックサムを読み出します。		
	パラメーター	なし		
	戻り値	CY8CMBR2110 デバイスのフラッシュ チェックサム		
	例	MBR_ReadFlashChecksum();		

5	**プロトタイプ**	**WORD MBR_ReadRAMChecksum(void);**
	説明	CY8CMBR2110 デバイスの RAM に格納されるチェックサムを読み出します。
	パラメーター	なし
	戻り値	CY8CMBR2110 デバイスの RAM のチェックサム
	例	MBR_ReadRAMChecksum();
6	**プロトタイプ**	**void MBR_SetChecksum(void);**
	説明	ホストが計算したチェックサムを CY8CMBR2110 デバイスに書き込みます。ホストは自身で設定のチェックサムを計算します。
	パラメーター	なし
	戻り値	なし
	例	MBR_SetChecksum();
7	**プロトタイプ**	**BYTE MBR_ReadChecksumMatch(void);**
	説明	CY8CMBR2110 が計算した RAM チェックサムがホストが計算したチェックサムと同じかチェックします。
	パラメーター	なし
	戻り値	0 または 1 チェックサムが一致しない場合 0 を返す チェックサムが一致した場合 1 を返す
	例	MBR_ReadChecksumMatch();
8	**プロトタイプ**	**BYTE MBR_SaveSettingsToFlash(void);**
	説明	CY8CMBR2110 デバイスの現時点の設定をフラッシュに保存します (CY8CMBR2110 設定)。
	パラメーター	なし
	戻り値	0 または 1 0 – 「フラッシュに保存」が失敗 1 – 「フラッシュに保存」が成功
	例	MBR_SaveSettingsToFlash();
9	**プロトタイプ**	**BYTE MBR_SettingsLoaded(void);**
	説明	工場出荷時のデフォルト設定あるいはユーザーによる設定がロードされるかを示します。
	パラメーター	なし
	戻り値	0 または 1 0 – ユーザによる設定 1 – 工場出荷時のデフォルト設定
	例	MBR_SettingsLoaded( );

10	**プロトタイプ**	**void MBR_LoadFactoryDefaults(void);**		
	説明	工場出荷時のデフォルト設定をCY8CMBR2110デバイスのRAMにロードします。		
	パラメーター	なし		
	戻り値	なし		
	例	MBR_LoadFactoryDefaults();		
11	**プロトタイプ**	**void MBR_ReadConfigData(BYTE abConfigData[]);**		
	説明	LED コンフィギュレーションおよびデバイス コンフィギュレーションのデータを CY8CMBR2110 デバイスからロードします。		
	パラメーター	**名前**	**説明**	**可能な値**
		abConfigData	すべてのコンフィギュレーション データを格納する 64 バイト アレイへのポインタ	
	戻り値	なし		
	例	MBR_ReadConfigData(abConfigData); abConfigData は 64 バイト アレイ abConfigData[64] へのポインタです。アレイはすべてのコンフィギュレーション データで更新されます。		
12	**プロトタイプ**	**void MBR_LEDEffectsBreathing(BYTE bGPO, BYTE bBreath);**		
	説明	ボタン タッチLEDブリージング エフェクトをイネーブル／ディセーブル 注: **LEDエフェクトでは、GPOはGPO0、GPO123、GPO456、GPO789にグループ分けされ、1つのGPOを設定すると、そのグループの残りのGPOも設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bBreath	ブリージング エフェクトをイネーブル／ディセーブル	0 または 1 0 – ブリージングをディセーブル 1 – ブリージングをイネーブル
	戻り値	なし		
	例	MBR_LEDEffectsBreathing(GPO4, FEATURE_ENABLE); GPO4 は値 4 のマクロで、FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		

13	**プロトタイプ**	**void MBR_LEDEffectsRepeatRate(BYTE bGPO, BYTE bRepeatRate, BYTE bPwrOnOrBtnTch);**		
	説明	選択した GPO に対し LED エフェクトの反復回数を設定します。 注: **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされ、1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bRepeatRate	LED エフェクトの反復回数	0～7 0 – 0 の繰り返し速度 1 – 1 の繰り返し速度 2 – 2 の繰り返し速度 3 – 4 の繰り返し速度 4 – 6 の繰り返し速度 5 – 10 の繰り返し速度 6 – 15 の繰り返し速度 7 – 20 の繰り返し速度
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsRepeatRate(GPO1,REPEAT_RATE_20,POWER_ON_LED_EFFECTS); GPO1、REPEAT_RATE_20、POWER_ON_LED_EFFECTS はそれぞれの値が 1、7、1 のマクロです (inputs.h)。		

14	**プロトタイプ**	**void MBR_LEDEffectsLowBrightness(BYTE bGPO, BYTE bLowBright, BYTE bPwrOnOrBtnTch);**		
	説明	GPO に対して、LED を低輝度で設定します。 注: **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされ、1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bLowBright	レジスタ マップに対応した低輝度	0～7 0 – 低輝度 0% 1 – 低輝度 10% …… …….. 7 – 低輝度 100%
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsLowBrightness(GPO2, LOW_BRIGHT_80, BTN_TOUCH_LED_EFFECTS); GPO2、LOW_BRIGHT_80、BTN_TOUCH_LED_EFFECTS はそれぞれの値が 2、6、2 のマクロです (inputs.h)。		
15	**プロトタイプ**	**void MBR_LEDEffectsHighBrightness(BYTE bGPO, BYTE bHighBright, BYTE bPwrOnOrBtnTch);**		
	説明	GPO に対し LED の高輝度を設定します。 注: **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされ、1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bHighBright	レジスタ マップに対応した高輝度	0～7 0 – 高輝度 100% 1 – 高輝度 90% …… …….. 7 – 高輝度 0%
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsHighBrightness(GPO9, HIGH_BRIGHT_50, BTN_TOUCH_LED_EFFECTS); GPO9、HIGH_BRIGHT_50、BTN_TOUCH_LED_EFFECTS はそれぞれの値が 9、4、2 のマクロです (inputs.h)。		

16	**プロトタイプ**	**void MBR_LEDEffectsLowTime(BYTE bGPO, BYTE bLowTime, BYTE bPwrOnOrBtnTch);**		
	説明	各 GPO に対して、低輝度での LED の時間を設定します。 注 **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされている。1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bLowTime	低輝度での LED の時間の値を収得するためのグローバル期間レジスタ マップ	0～1 0 – GLOBAL_PERIOD_1 1 – GLOBAL_PERIOD_2
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsLowTime(GPO6, GLOBAL_PERIOD_1, BTN_TOUCH_LED_EFFECTS); GPO6、GLOBAL_PERIOD_1、BTN_TOUCH_LED_EFFECTS はそれぞれの値が 6、0、2 のマクロです (inputs.h)。		
17	**プロトタイプ**	**void MBR_LEDEffectsHighTime(BYTE bGPO, BYTE bHighTime, BYTE bPwrOnOrBtnTch);**		
	説明	各 GPO に対して、高輝度での LED の時間をセットします。 注: **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされ、1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bHighTime	高輝度での LED の時間の値を収得するためのグローバル期間レジスタ マップ	0～1 0 – GLOBAL_PERIOD_1 1 – GLOBAL_PERIOD_2
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsHighTime(GPO5, GLOBAL_PERIOD_2, BTN_TOUCH_LED_EFFECTS); GPO5、GLOBAL_PERIOD_2、BTN_TOUCH_LED_EFFECTS はそれぞれの値が 5、1、2 のマクロです (inputs.h)。		

18	**プロトタイプ**	**void MBR_LEDEffectsRampDown(BYTE bGPO, BYTE bRampDown, BYTE bPwrOnOrBtnTch);**		
	説明	GPO のランプ ダウン時間を設定します。 注: **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされ、1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bRampDown	ランプ ダウン時間の値を収得するためのグローバル期間レジスタ マップ	0～3 0 – GLOBAL_PERIOD_1 1 – GLOBAL_PERIOD_2 2 – GLOBAL_PERIOD_3 3 – GLOBAL_PERIOD_4
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsRampDown(GPO8, GLOBAL_PERIOD_1, POWER_ON_LED_EFFECTS); GPO8、GLOBAL_PERIOD_1、POWER_ON_LED_EFFECTS はそれぞれの値が 8、0、1 のマクロです (inputs.h)。		
19	**プロトタイプ**	**void MBR_LEDEffectsRampUp(BYTE bGPO, BYTE bRampUp, BYTE bPwrOnOrBtnTch);**		
	説明	GPO のランプ アップ時間を設定します。 注: **LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 にグループ分けされ、1 つの GPO を設定すると、そのグループの残りの GPO も設定されます。**		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		bRampUp	ランプ アップ時間の値を収得するためのグローバル期間レジスタ マップ	0～3 0 – GLOBAL_PERIOD_1 1 – GLOBAL_PERIOD_2 2 – GLOBAL_PERIOD_3 3 – GLOBAL_PERIOD_4
		bPwrOnOrBtnTch	パワー オンまたはボタン タッチ LED エフェクト	1 または 2 1 – パワー オン LED 2 – ボタン タッチ LED
	戻り値	なし		
	例	MBR_LEDEffectsRampUp(GPO8, GLOBAL_PERIOD_1, POWER_ON_LED_EFFECTS) GPO8、GLOBAL_PERIOD_1、POWER_ON_LED_EFFECTS は値が 8、0、1 のマクロです (inputs.h)。		

20	**プロトタイプ**	**void MBR_PowerONLEDEffectSeq(BYTE bPwrOnSeq);**		
	説明	パワー オン LED エフェクト シーケンス (同時またはシーケンシャル) を設定します。この API を呼び出す前にパワー オン LED エフェクトを有効にする必要があります。		
	パラメーター	**名前**	**説明**	**可能な値**
		bPwrOnSeq	パワー オン LED エフェクト シーケンスのタイプ	0 または 1 0 – 同時 1 – シーケンシャル
	戻り値	なし		
	例	MBR_PowerONLEDEffectSeq(POWER_ON_SEQUENTIAL); POWER_ON_SEQUENTIAL は値が 1 のマクロです (inputs.h)。		
21	**プロトタイプ**	**void MBR_PowerONLEDEffects(BYTE bEnable);**		
	説明	パワー オン LED エフェクトをイネーブル／ディセーブル		
	パラメーター	**名前**	**説明**	**可能な値**
		bEnable	エフェクトをイネーブル／ディセーブル	0 または 1 0 – エフェクトをディスエーブル 1 – エフェクトをイネーブル
	戻り値	なし		
	例	MBR_PowerONLEDEffects(FEATURE_ENABLE); FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		
22	**プロトタイプ**	**void MBR_ButtonLEDEffects(BYTE bEnable);**		
	説明	ボタン タッチ LED エフェクトをイネーブル／ディセーブル		
	パラメーター	**名前**	**説明**	**可能な値**
		bEnable	エフェクトをイネーブル／ディセーブル	0 または 1 0 – エフェクトをディスエーブル 1 – エフェクトをイネーブル
	戻り値	なし		
	例	MBR_ButtonLEDEffects(FEATURE_DISABLE); FEATURE_DISABLE は値が 0 のマクロです (inputs.h)。		
23	**プロトタイプ**	**void MBR_StandbyModeLEDBrightness(BYTE bLEDBrightness);**		
	説明	スタンドバイ モード LED 輝度レベルを設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bLEDBrightness	レジスタ マップに対応するスタンドバイ モードでの輝度レベル	0～3 0 – 0%輝度 1 – 20%輝度 2 – 30%輝度 3 – 50%輝度
	戻り値	なし		
	例	MBR_StandbyModeLEDBrightness(STDBY_LED_50); STDBY_LED_50 は値が 3 のマクロです (inputs.h)。		

24	**プロトタイプ**	**void MBR_LEDEffectLastButton(BYTE bEnable);**		
	説明	最後にタッチしたボタン の LED エフェクト機能をイネーブル／ディセーブル		
	パラメーター	**名前**	**説明**	**可能な値**
		BYTE bEnable	エフェクトをイネーブル／ディセーブル	0 または 1 0: エフェクトをディスエーブル 1: エフェクトをイネーブル
	戻り値	なし		
	例	MBR_LEDEffectLastButton(FEATURE_DISABLE); FEATURE_DISABLE は値が 0 のマクロです (inputs.h)。		
25	**プロトタイプ**	**void MBR_SetGlobalPeriod(BYTE bPeriodReg, WORD wPeriodValue);**		
	説明	グローバル期間レジスタの期間値を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bPeriodReg	グローバル期間レジスタ マップ	0～3 0 – GLOBAL_PERIOD_1 1 – GLOBAL_PERIOD_2 2 – GLOBAL_PERIOD_3 3 – GLOBAL_PERIOD_4
		wPeriodValue	グローバル期間値 (単位: ms)	0～1600
	戻り値	なし		
	例	MBR_SetGlobalPeriod(GLOBAL_PERIOD_1,600) GLOBAL_PERIOD_1 は値が 0 のマクロです (inputs.h)。		
26	**プロトタイプ**	**void MBR_SetAllGlobalPeriods(WORD awPeriodValue[]);**		
	説明	すべてのグローバル期間レジスタの期間値を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		awPeriodValue	ms 単位で表す期間値を格納する 4 ワード アレイへのポインタ	0～1600
	戻り値	なし		
	例	MBR_SetAllGlobalPeriods(wTestBuffer); wTestBuffe は 4 ワード アレイ wTestBuffer[4]のベース ポインタです。		

27	**プロトタイプ**	**void MBR_SetAllLEDParameters(BYTE bGPO, BYTE abParam[]);**		
	説明	任意の GPO に対して、すべての LED エフェクト パラメーターを設定します。 **注:** LED エフェクトでは、GPO は GPO0、GPO123、GPO456、GPO789 としてグループ化されます。 1 つの GPO を 1 つのグループに設定すると、そのグループの他の GPO も設定されます。		
	パラメーター	**名前**	**説明**	**可能な値**
		bGPO	GPO 番号	0～9
		awPeriodValue	設定パラメーターを格納する 9 バイト アレイへのポインタ	byte[0]: パワー オンまたはボタン タッチ エフェクト byte[1] - 高輝度レベル byte[2] - 低輝度レベル byte[3] - グローバル期間レジスタにマッピングするランプ アップ時間 byte[4] - グローバル期間レジスタにマッピングするランプ ダウン時間 byte[5] - グローバル期間レジスタにマッピングするHIGH時間 byte[6] - グローバル期間レジスタにマッピングするLOW時間 byte[7]: 反復回数 byte[8]: ブリージング エフェクトのイネーブル／ディセーブル
	戻り値	なし		
	例	MBR_SetAllLEDParameters(GPO2, bconfig); bconfig は 9 バイト アレイ bconfig[9]へのポインタです。		
28	**プロトタイプ**	**BYTE MBR_ReadDeviceID(void);**		
	説明	CY8CMBR2110 デバイス ID を読み出します。		
	パラメーター	なし		
	戻り値	CY8CMBR2110のデバイスID。IDは「0xA1」です。		
	例	MBR_ReadDeviceID();		
29	**プロトタイプ**	**BYTE MBR_ReadFWRevision(void);**		
	説明	スレーブ デバイス ファームウェア リビジョンを読み出します。		
	パラメーター	なし		
	戻り値	デバイス ファームウェア リビジョン		
	例	MBR_ReadFWRevision();		
30	**プロトタイプ**	**void MBR_SetDebugSensorNumber(BYTE bSensor);**		
	説明	デバッグ データが送信されるセンサー番号を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSensor	センサ番号	0～9
	戻り値	なし		
	例	MBR_SetDebugSensorNumber(CS0); CS0 は値が 0 のマクロです (inputs.h)。		

31	**プロトタイプ**	**void MBR_SetDebugDataParameter(BYTE bParameter);**		
	説明	デバッグ データ出力のときに送信されるパラメーター タイプをセットします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bParameter	パラメーターのタイプ	0～4 0 – C_P 1 – raw カウント 2 – 差分カウント 3 – raw カウント、ベースライン 4 – すべてのパラメーター (C_P、raw カウント、差分 カウント、ベースライン、SNR)
	戻り値	なし		
	例	MBR_SetDebugDataParameter(DEBUG_PARAM_CP); DEBUG_PARAM_CP は値が 0 のマクロです (inputs.h)。		
32	**プロトタイプ**	**void MBR_ReadDebugData(BYTE abDebugData[]);**		
	説明	デバッグ データ レジスタ マップから選択されたパラメーターのデバッグ データを読み出します。		
	パラメーター	**名前**	**説明**	**可能な値**
		abDebugData	デバッグ データを保持する 25 バイト アレイへのポインタ	
	戻り値	なし		
	例	MBR_ReadDebugData(bgetdata); bgetdata は 25 バイト アレイ bgetdata[25]へのポインタです。アレイはデバッグ データで更新されます。		
33	**プロトタイプ**	**void MBR_SetBuzzer(BYTE bEnable);**		
	説明	オーディオ フィードバック (ブザー) をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bEnable	ブザーをイネーブル／ディセーブル	0 または 1 0 – イネーブル 1 – ディセーブル
	戻り値	なし		
	例	MBR_SetBuzzer(FEATURE_ENABLE) FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		
34	**プロトタイプ**	**void MBR_SetBuzzerPins(BYTE bBuzzerPins);**		
	説明	ブザーのピン数を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bBuzzerPins	ブザー出力ピンの数	0 または 1 0 – 1 ピン式のブザー 1 – 2 ピン式のブザー
	戻り値	なし		
	例	MBR_SetBuzzerPins(BUZZER_AC_2_PIN); BUZZER_AC_2_PIN は値が 1 のマクロです (inputs.h)。		

35	**プロトタイプ**	**void MBR_SetBuzzerIdleState(BYTE bIdleState);**		
	説明	ブザー ピンのアイドル状態を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bIdleState	ブザーのアイドル状態	0 または 1 0 – LOW 1 – HIGH
	戻り値	なし		
	例	MBR_SetBuzzerIdleState(BUZZER_IDLE_HIGH); BUZZER_IDLE_HIGH は値が 1 のマクロです (inputs.h)。		
36	**プロトタイプ**	**void MBR_SetBuzzerFrequency(BYTE bFrequency);**		
	説明	ブザー出力の出力周波数を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bFrequency	ブザー出力周波数	1～7 1 – 4000Hz 2 – 2670Hz 3 – 2000Hz 4 – 1600Hz 5 – 1330Hz 6 – 1140Hz 7 – 1000Hz
	戻り値	なし		
	例	**MBR_SetBuzzerFrequency (**BUZZER_FREQ_1000); BUZZER_FREQ_1000 は値が 7 のマクロです (inputs.h)。		
37	**プロトタイプ**	**void MBR_SetBuzzerOutputDuration(WORD wDuration);**		
	説明	ブザー出力の期間を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		wDuration	ブザー出力期間 (単位: ms)	(0～127) * ボタン スキャン速度
	戻り値	なし		
	例	MBR_SetBuzzerOutputDuration(1000);		

38	**プロトタイプ**	**void MBR_SetAllBuzzerParameters(BYTE bEnable, BYTE bParameters[], WORD wOutputDuration);**		
	説明	CY8CMBR2110 デバイスのすべてのブザー パラメーターを設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bEnable	ブザーをイネーブル／ディセーブル	0 または 1 0 – ディセーブル 1 – イネーブル
		bParameters	要求される入力を保持する 3 バイト アレイへのポインタ	Byte[0] – ブザー ピンの数 Byte[1] – ブザー アイドル状態 Byte[2] – ブザー出力周波数
		wOutputDuration	ブザー出力の期間 (単位: ms)	(0～127) * ボタン スキャン速度
	戻り値	なし		
	例	MBR_SetAllBuzzerParameters(FEATURE_ENABLE , bbuzzconfig ,1000); FEATURE_ENABLE は値が 1 のマクロです (inputs.h)、bbuzzconfig はブザー ピン、アイドル状態、および周波数詳細 (1000 がブザー期間である) を含む 3 バイト アレイへのポインタ。		
39	**プロトタイプ**	**void MBR_SetI2CSlaveAddress (BYTE bNewSlaveAddress);**		
	説明	CY8CMBR2110 デバイスの I^2C アドレスを設定します。デフォルト アドレスは「37h」です。		
	パラメーター	**名前**	**説明**	**可能な値**
		bNewSlaveAddress	デバイスへの新しいアドレス値	0x00～0x7F
	戻り値	なし		
	例	MBR_SetI2CSlaveAddress(50);		
40	**プロトタイプ**	**void MBR_SetAdaptiveThreshold(BYTE bSetRest);**		
	説明	CY8CMBR2110 デバイスの自動閾値設定機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSetRest	自動閾値機能をイネーブル／ディセーブルします。	0 または 1 0 – ディセーブル 1 – イネーブル
	戻り値	なし		
	例	MBR_SetAdaptiveThreshold(FEATURE_ENABLE); FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		

41	**プロトタイプ**	**void MBR_SetSensitivity(BYTE bButtonNumber, BYTE bButtonSensitivityLevel);**		
	説明	ボタンの感度の値を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
		bButtonSensitivityLevel	ボタンの感度レベル	1～3 1 – 感度「高」 2 – 感度「中」 3 – 感度「低」
	戻り値	なし		
	例	MBR_SetSensitivity (CS9, SENSITIVITY_MEDIUM); CS9 と SENSITIVITY_MEDIUM はそれぞれの値が 9 と 2 のマクロです (inputs.h)。		
42	**プロトタイプ**	**void MBR_SetSensitivityAll(BYTE bsensitivity[]);**		
	説明	すべてのボタンの感度値を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bsensitivity	すべてのボタンの感度レベルを格納する 10 バイト アレイへのポインタ	1～3 1 – 感度「高」 2 – 感度「中」 3 – 感度「低」
	戻り値	なし		
	例	test_MBR_SetSensitivityAll(bBuffer); bBuffer は、すべてのボタンの感度レベルを格納する 10 バイト アレイ bBuffer[10]へのポインタです (第 1 バイトはボタン 0 に対応、･･･第 10 バイトはボタン 9 に対応します)。		
43	**プロトタイプ**	**void MBR_SetDebounce(BYTE bButtonNumber, BYTE bDebouncevalue);**		
	説明	ボタンのデバウンス レベルを設定します。ボタン 1～ボタン 9 は同じ値で設定されます。 これらのボタンは個別に設定することはできません。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～1 0 – ボタン 0 の場合 1 – ボタン 1～ボタン 9 の場合
		bDebouncevalue	ボタンのデバウンス値	1～255
	戻り値	なし		
	例	MBR_SetDebounce(DEBOUNCE_FOR_CS0, 200); DEBOUNCE _FOR_CS0 は値が 0 のマクロです (inputs.h)。		

44	**プロトタイプ**	**void MBR_SetFingerThreshold(BYTE bButtonNumber, BYTE bFingerthreshold);**		
	説明	ボタンの指閾値を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
		bFingerthreshold	指閾値レベル	0～15
	戻り値	なし		
	例	MBR_SetFingerThreshold(CS3, FINGER_THRESHOLD_180); CS3 および FINGER_THRESHOLD_180 はそれぞれの値が 3 と 10 のマクロです (inputs.h)。		
45	**プロトタイプ**	**void MBR_SetFingerThresholdAll(BYTE bFingerthreshold[]);**		
	説明	すべてのセンサーの指閾値を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bFingerthreshold	指閾値を格納する 10 バイト アレイへのポインタ	0～15
	戻り値	なし		
	例	MBR_SetFingerThresholdAll(Buffer); Buffer はすべてのボタンの指閾値を格納する 10 バイト アレイ Buffer[10]へのポインタです。第 1 バイトはボタン 0 に対応、・・・ 第 10 バイトはボタン 9 に対応します。		
46	**プロトタイプ**	**BYTE MBR_ReadSensorStatus(BYTE bButtonNumber);**		
	説明	ボタン タッチの現時点の状態をチェックするために、現時点のボタン状態を読み出します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
	戻り値	0 または 1 0 – ボタンは押されない (オフ) 1 – ボタンは押される (オン)		
	例	MBR_ReadSensorStatus(CS5); CS5 は値が 5 のマクロです (inputs.h)。		
47	**プロトタイプ**	**WORD MBR_ReadSensorStatusAll(void);**		
	説明	すべてのボタンの現時点の状態を読み出します。		
	戻り値	すべてのセンサーの現時点の状態を含む 2 バイト。 LSB は 0～7 のボタンです。 MSB の最初の 2 ビットはボタン 8 とボタン 9 です。 例えば、0x0301 はボタン 0、8、9 がタッチされている (オン)、残りのボタンは非タッチ (オフ) を示します。		
	例	MBR_ReadSensorStatusAll();		

48	**プロトタイプ**	**WORD MBR_ReadLatchStatusAll(void);**		
	説明	すべてのセンサーのラッチされた状態を読み出します。		
	パラメーター	なし		
	戻り値	すべてのセンサーの現時点のラッチされた状態を含む 2 バイト LSB は 0～7 のボタンです。 MSB の最初の 2 ビットはボタン 8 とボタン 9 です。 例えば、0x0301 は、I^2C の電流が読み出される前に、ボタン 0、8、9 がタッチされている (オン)、 残りのボタンは非タッチ (オフ) を示します。		
	例	MBR_ReadLatchStatusAll();		
49	**プロトタイプ**	**void MBR_EnterDeepSleep(void);**		
	説明	デバイスがディープ スリープ モードへ移行するように、動作モード レジスタのディープ スリープ ビットを 1 に設定します。ディープ スリープ モードの手順を行って、モードをディープ スリープ モードに変更してください。		
	パラメーター	なし		
	戻り値	なし		
	例	MBR_EnterDeepSleep();		
50	**プロトタイプ**	**void MBR_SetPowerOptimization(BYTE bOptimization);**		
	説明	CY8CMBR2110 デバイスの消費電力最適化または応答時間最適化の設計を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bOptimization	消費電力最適化または応答時間最適化をイネーブルにします。	0 または 1 0 – 応答時間最適化 1 – 消費電力最適化
	戻り値	なし		
	例	MBR_SetPowerOptimization(PWR_CONS_OPT); PWR_CONS_OPT は値が 1 のマクロです (inputs.h)。		
51	**プロトタイプ**	**void MBR_SetScanRate(BYTE bSetscanvalue);**		
	説明	CY8CMBR2110 デバイスのスキャン速度を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSetscanvalue	レジスタ マップに対応したスキャン速度	0～31 0～25ms 31～561ms
	戻り値	なし		
	例	MBR_SetScanRate(30);		

52	**プロトタイプ**	**void MBR_SetHGPOValue(BYTE bHGPO_Number, BYTE bDriveLogic);**		
	説明	HGPO の駆動論理を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bHGPO_Number	ホスト制御 GPO (HGPO) 番号	0～3 0 – HGPO0 1 – HGPO1 2 – HGPO2 3 – HGPO3
		bDriveLogic	駆動論理レベル	0 または 1 1 – HIGH 0 – LOW
	戻り値	なし		
	例	MBR_SetHGPOValue(HOSTGPO_3, HOSTGPO_HIGH); HOSTGPO_3 および HOSTGPO_HIGH はそれぞれの値が 3 と 1 のマクロです (inputs.h)。		
53	**プロトタイプ**	**void MBR_SetAllHGPOValue(BYTE bdriveGP0, BYTE bdriveGP1, BYTE bdriveGP2, BYTE bdriveGP3);**		
	説明	すべての HGPO の駆動論理を設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bdriveGP0	HGPO0 の駆動論理	0 または 1
		bdriveGP1	HGPO1 の駆動論理	0 または 1
		bdriveGP2	HGPO2 の駆動論理	0 または 1
		bdriveGP3	HGPO3 の駆動論理	0 または 1
	戻り値	なし		
	例	void MBR_SetAllHGPOValue(HOSTGPO_HIGH, HOSTGPO_HIGH, HOSTGPO_LOW, HOSTGPO_LOW); HOSTGPO_HIGH、HOSTGPO_HIGH、HOSTGPO_LOW および HOSTGPO_LOW はそれぞれの値が 1、1、0、0 のマクロです (inputs.h)。		
54	**プロトタイプ**	**void MBR_AnalogOutput(BYTE bSet_Reset)**		
	説明	CY8CMBR2110 デバイスのアナログ出力電圧機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSet_Reset	アナログ出力電圧機能をセット／リセットします。	0 または 1
	戻り値	なし		
	例	MBR_AnalogOutput(FEATURE_ENABLE); FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		

55	**プロトタイプ**	**void MBR_SetToggle (BYTE bButtonNumber, BYTE fSet_Reset);**		
	説明	CY8CMBR2110 デバイスのボタンのトグル機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
		fSet_Reset	トグルをイネーブル／ ディセーブルします	0 または 1 0 – ディセーブル 1 – イネーブル
	戻り値	なし		
	例	MBR_SetToggle(CS0,FEATURE_ENABLE); FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		
56	**プロトタイプ**	**void MBR_SetToggleAll(BYTE afSet_Reset[]);**		
	説明	全てのボタンのトグル機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		afSet_Reset	値を保持する 2 バイト アレイへのポインタ	Byte[1] – 0x00～0xFF Byte[2] – 0x00～0x03
	戻り値	なし		
	例	MBR_SetToggleAll(Buffer); Buffer は 2 バイト アレイのバッファへのポインタです。 Buffer[1]はボタン 0～ボタン 7 の値を保持します。 Buffer[2]の最初の 2 ビットはボタン 8 とボタン 9 の値を保持します。 例えば、Buffer[1]が 0xFF のとき、ボタン 0～7のトグルはイネーブルにされ、Buffer[2]が 0x01 のとき、 ボタン 8 のトグルがイネーブルにされ、ボタン 9 のトグルはディセーブルされます。		
57	**プロトタイプ**	**void MBR_LEDONTime(BYTE bSet_Reset);**		
	説明	CY8CMBR2110 デバイスの LED オン機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSet_Reset	機能をイネーブル／ディセーブルします。	0 または 1 0 – ディセーブル 1 – イネーブル
	戻り値	なし		
	例	MBR_LEDONTime(FEATURE_DISABLE); FEATURE_DIASBLE は値が 0 のマクロです (inputs.h)。		
58	**プロトタイプ**	**BYTE MBR_ReadValidSensors(void);**		
	説明	有効なセンサー／ボタン カウントを読み出します。 VDD への短絡、GND への短絡、不適切な C_P 値、または不適切な CMOD 値のため、ボタンが無効になることがあります。		
	パラメーター	なし		
	戻り値	有効なセンサー カウントを示す 1 バイトのデータ。 8 の返り値は、8 個のボタンが有効で、2 つのボタンが無効であることを示します。		
	例	MBR_ReadValidSensors();		

59	**プロトタイプ**	**BYTE MBR_ReadFMEAGround(BYTE bButtonNumber);**		
	説明	短絡接地のため、1 つのボタンのシステム診断データを読み出します (ボタンが接地されるかチェックする必要があります)。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
	戻り値	0 または 1 0 – ボタンは接地されない 1 – ボタンが接地される		
	例	MBR_ReadFMEAGround(CS9); CS9 は値が 9 のマクロです (inputs.h)。		
60	**プロトタイプ**	**WORD MBR_ReadFMEAGroundAll(void);**		
	説明	接地のため、すべてのボタンのシステム診断データを読み出します。		
	パラメーター	なし		
	戻り値	どのボタンが接地するか示すための 2 バイト LSB はボタン 0～7 です。 MSB の最初の 2 ビットはボタン 8 とボタン 9 です。 例えば、0x02F1 はボタン 0、4、5、6、7、9 が接地されることを示します。		
	例	MBR_ReadFMEAGroundAll();		
61	**プロトタイプ**	**BYTE MBR_ReadFMEAVDD(BYTE bButtonNumber);**		
	説明	VDD への短絡接触のため、1 つのボタンのシステム診断データを読み出します (ボタンが VDD へ接触されるかチェックする必要があります)。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
	戻り値	0 または 1 0 – ボタンは VDD と接続されない 1 – ボタンは VDD へ接続される		
	例	MBR_ReadFMEAVDD(CS8); CS8 は値が 8 のマクロです (inputs.h)。		
62	**プロトタイプ**	**WORD MBR_ReadFMEAVDDAll(void);**		
	説明	VDD との接続のため、すべてのボタンのシステム診断データを読み出します。		
	パラメーター	なし		
	戻り値	どのボタンが VDD と接続されるか示すための 2 バイト LSB は 0～7 のボタンです。 MSB の最初の 2 ビットはボタン 8 とボタン 9 です。 例えば、0x02F1 はボタン 0、4、5、6、7、9 が VDD と接続されることを示します。		
	例	MBR_ReadFMEAVDDAll();		

63	**プロトタイプ**	**WORD MBR_ReadFMEASnsToSnsAll(void);**		
	説明	ボタン間での接続のため、すべてのボタンのシステム診断データを読み出します。		
	パラメーター	なし		
	戻り値	どのボタンが別のボタンへ接続されるか示すための 2 バイト LSB は 0～7 のボタンです。 MSB の最初の 2 ビットはボタン 8 とボタン 9 です。 例えば、0x02F1 はボタン 0、4、5、6、7、9 がその他のボタンへ接続されることを示します。		
	例	MBR_ReadFMEASnsToSnsAll();		
64	**プロトタイプ**	**BYTE MBR_ReadFMEASensorCP(BYTE bButtonNumber);**		
	説明	高 Cp 値のため、1 つのボタンのシステム診断データを読み出します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
	戻り値	0 または 1 0 – ボタンの Cp 値は適切 (Cp<40pF) 1 – ボタンの Cp 値は高い (Cp>40pF)		
	例	MBR_ReadFMEASensorCP(CS0); CS0 は値が 0 のマクロです (inputs.h)。		
65	**プロトタイプ**	**WORD MBR_ReadFMEASensorCpAll(void);**		
	説明	高 C_P の値のため、すべてのボタンのシステム診断データを読み出します。		
	パラメーター	なし		
	戻り値	どのボタンが高い Cp 値を持っているか示すための 2 バイト LSB は 0～7 のボタンです。 MSB の最初の 2 ビットはボタン 8 とボタン 9 です。 例えば、0x02F1 は (Cp>40pF) のボタン 0、4、5、6、7、9 を示します		
	例	MBR_ReadFMEASensorCpAll();		
66	**プロトタイプ**	**BYTE MBR_ReadFMEACMOD(void);**		
	説明	CMOD のシステム診断を読み出します (CMOD 容量値をチェックしてください)。		
	パラメーター	なし		
	戻り値	0 – CMOD が (1～4) nF の範囲内 1 – COMD が 4nF より高い 2 – CMOD が 1nF より低い		
	例	MBR_ReadFMEACMOD();		

67	**プロトタイプ**	**BYTE MBR_ReadSensorSNR(BYTE bButtonNumber);**		
	説明	ボタンの SNR 値を読み出します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
	戻り値	ボタン SNR を示す 1 バイト。1 つのボタンに対して、SNR 値は 0～15 の範囲です。		
	例	MBR_ReadSensorSNR(CS9); CS9 は値が 9 のマクロです (inputs.h)。		
68	**プロトタイプ**	**void MBR_ReadSensorSNRAll(BYTE bSensor_SNR[TOTAL_BUTTON_COUNT]);**		
	説明	すべてのボタンの SNR 値を読み出します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSensor_SNR[TOTAL_BUTTON_COUNT]	デバイスからのリード データを保持する 10 バイト アレイへのポインタ	
	戻り値	なし		
	例	MBR_ReadSensorSNRAll(buffer); buffer はボタン 0 から始まる SNR 値で更新される 10 バイト アレイへのポインタです (第 1 バイトがボタン 0 に対応、第 2 バイトがボタン 1 に対応、・・・第 10 バイトがボタン 9 に対応します)。		
69	**プロトタイプ**	**void MBR_SetAutoResetTime(BYTE bSet_time);**		
	説明	すべてのボタンの自動リセット タイムを設定します。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSet_time	自動リセット タイムの値	1 – 制限無し 2 – 5 秒 3 – 20 秒
	戻り値	なし		
	例	MBR_SetAutoResetTime(AUTO_RESET_20S); AUTO_RESET_20S は値が 3 のマクロです (inputs.h)。		
70	**プロトタイプ**	**void MBR_SetEMC(BYTE bSet_Reset);**		
	説明	CY8CMBR2110 デバイスの EMC 機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bSet_Reset	EMC をイネーブル／ディセーブルします。	0 または 1 0 – ディセーブル 1 – イネーブル
	戻り値	なし		
	例	MBR_SetEMC(FEATURE_ENABLE); FEATURE_ENABLE は値が 1 のマクロです (inputs.h)。		

71	**プロトタイプ**	**void MBR_SetFSS(BYTE bButtonNumber, BYTE fSet_Reset);**		
	説明	1 つのボタンの FSS (隣接センサー抑制) 機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		bButtonNumber	ボタン番号	0～9
		fSet_Reset	機能をイネーブル／ ディセーブルします。	0 または 1 0 – ディセーブル 1 – イネーブル
	戻り値	なし		
	例	MBR_SetFSS(CS0,FEATURE_DISABLE); CS0 および FEATURE_DISABLE はそれぞれの値が 0 と 1 のマクロです (inputs.h)。		
72	**プロトタイプ**	**void MBR_SetFSSAllSensors(BYTE afSet_Reset[])**		
	説明	すべてのボタンの FSS (隣接センサー抑制) 機能をイネーブル／ディセーブルします。		
	パラメーター	**名前**	**説明**	**可能な値**
		afSet_Reset	設定値を格納する 2 バイト アレイへのポインタ	Byte[1] – 0x00～0xFF Byte[2] – 0x00～0x03
	戻り値	なし		
	例	MBR_SetFSSAllSensors(Buffer); Buffer は 2 バイト アレイ Buffer[2]へのポインタです。 Buffer[1]はボタン 0～7 の値を保持します。 Buffer[2]の最初の 2 ビットはボタン 8 とボタン 9 の値を保持します。 例えば、Buffer[1]が 0xFF のとき、ボタン 0～7 の FSS がイネーブルにされ、Buffer[2]が 0x01 のときにボタン 8 の機能がイネーブルにされ、Buffer[2]が 0x02 のときにボタン 9 の機能がイネーブルにされ、ボタン 8 の機能がディセーブルされます。		

7.2.2 低レベル API

1	**プロトタイプ**	**void MBR_WriteBytes(BYTE abWriteBuffer[], BYTE bNumberOfBytes)**		
	説明	バイト アレイを CapSense スレーブ デバイスに書き込みます。		
	パラメーター	**名前**	**説明**	**可能な値**
		abWriteBuffer	ホスト I²C バッファ アレイへのポインタ	1～31 バイト
		bNumberOfBytes	書き込まれたバイト数	1～31
	戻り値	なし		
	例	MBR_WriteBytes(abHostI2CBuffer, 3);		
2	**プロトタイプ**	**void MBR_ReadBytes(BYTE abReadBuffer[] , BYTE bNumberOfBytes)**		
	説明	バイト アレイを CapSense スレーブ デバイスから読み出します。		
	パラメーター	**名前**	**説明**	**可能な値**
		abReadBuffer	I²C バッファ アレイへのポインタ	1～32 バイト
		bNumberOfBytes	読み出すバイト数	1～32
	戻り値	なし		
	例	MBR_ReadBytes(bHostI2CBuffer, 6);		
3	**プロトタイプ**	**void MBR_Delay(WORD wDelayTime)**		
	説明	ソフトウェア遅延をミリ秒で実装します。		
	パラメーター	**名前**	**説明**	**可能な値**
		wDelayTime	遅延時間 (単位: ms)	
	戻り値	なし		
	例	MBR_Delay(100);		

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、センシング 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タッチされないときの raw カウントの傾向を推定するファームウェア 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。

ボタンまたはボタン 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

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) を上回る場合に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) を下回る場合に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、電力消費量を削減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させて単一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タッチする場合、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連動センサーをサポートするため、複数のセンサーはスキャンに AMUXBUS と同時に接続します。

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 ウィジェットにのみサポートされます。

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

静電容量センシングの PCB を設計するとき、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 (MxN) 個の交差部を監視することができます。

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 (TX と言う) と他の電極 (RX と言う) 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 パラメーターと共に使用されます。

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場合、センサーの近くに配置される金属の物体を除去する場合、耐液性のある CapSense 製品の水分を除去する場合、および他の急激な環境変化がある場合です。

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。この差がノイズ閾値を下回る場合、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配置されます。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、望ましくありません。

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必要な時間です。

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 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化されていると、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。

センサー連結

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。

信号対雑音比 (SNR)

タッチしたときのセンサー信号と非タッチのセンサー 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なり、タッチの X 位置と 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。

V_{REF}

PSoC 内にあるプログラマブル基準電圧ブロックであり、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、マトリクス ボタンおよびタッチパッドはサポートされるウィジェットです。

改訂履歴

改訂履歴

文書名: AN76000 - CY8CMBR2110 CapSense®デザイン ガイド 文書番号: 001-88933		
版	**発行日**	**変更内容**
**	08/26/2013	これは英語版 001-76000 Rev. *B からを翻訳した日本語版 001-88933 Rev. ** です。
*A	10/10/2014	これは英語版 001-76000 Rev. *C からを翻訳した日本語版 001-88933 Rev. *A です。
*B	01/25/2016	これは英語版 001-76000 Rev. * D からを翻訳した日本語版 001-88933 Rev. * B です。
*C	01/08/2020	これは英語版 001-76000 Rev. * G からを翻訳した日本語版 001-88933 Rev. * C です。