CapSense® デルタシグマプラス ADC データシート CSDADC V 1.40

Copyright © 2005-2012 Cypress Semiconductor Corporation. All Rights Reserved.

リソース	PSoC® ブロック			API メモリ (バイト) Typical		ピン (外部入出力ごと)
	デジタル	アナログ CT	アナログ SC	フラッシュ	RAM	
CY8C24x94、CY8CLED0xD、CY8CLED0xG、CY8CLED04。使用できる Flash、RAM、ピンの数は、センサの数と構成によって変わります。						
PRS16 ベースのユーザ モジュール (1 センサの場合)	3	1	2	1257	34	2 ー 5
PRS8- ベースのユーザ モジュール (1 センサの場合)	1	1	2	1185	32	2 ー 5
プリスケーラクロック (センサー 1 つの場合)	1	1	2	1185	32	2 ー 5
VC2 クロックを使ったユーザモジュール (1 センサの場合)	0	1	2	1170	1170	1
追加の各 CapSense® ボタン	-	-	-	2	10	1
5 つのセンサー素子を使った静電容量式スライダを使用する場合の、コード容量と RAM 使用エリアの増分	-	-	-	1197	79	5
追加の各スライダ素子	-	-	-	2	10	1
スライダでダイプレックスを使用する場合の、コード容量と RAM 使用エリアの増分	-	-	-	0	スライダ *2	-

特長および概要

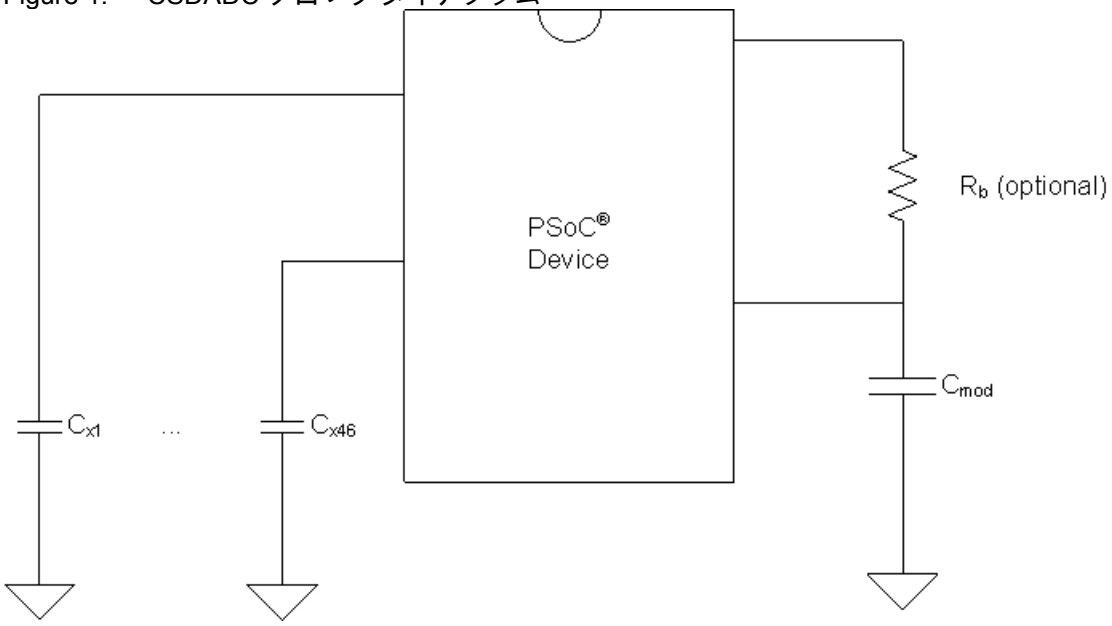
- CapSense センサのスキャンと電圧の測定を、ハードウエアの設定変更を行わず、同時に使用可能。(ダイナミックリコンフィグレーションの必要なし)
- ハードウェアに完全実装された $Sinc^N$ フィルタを使用して、CPU の負担の軽減とエイリアシングの要件を軽減
- デジタル ブロックを使わないコンフィグレーションをサポート
- ADC の特長 :
  - 2 次変調器を備えたデルタシグマ ADC
  - 符号なしまたは符号付き 2 の補数のフォーマットによるデータ
  - 10,12, 及び 14 ビットに分解能を動的に変更

  - 最大サンプリング速度は 10-bit 分解能で 31250 sps、14-bit 分解能で 7812 sps
  - 入力範囲は内部及び外部リファレンス オプションによって選択可能
  - コンフィギュラブルなゲインと基準設定を備えた内蔵プログラマブルゲインアンプ

- CapSense の特長：
  - 実績のある CSD 方式を採用
  - 2 次変調器による優れた S/N 比性能
  - 最大 46 個の静電容量式センサをスキャン
  - ガラスのオーバーレイが最大 25 mm まで検知可能
  - ワイヤベースセンサでは 30 cm の近接検知
  - AC 電源ノイズ、EMC ノイズ、電源電圧変化に対する高耐性
  - 異なる独立・スライド式静電容量式センサの組み合わせをサポート
  - ダイプレックスを使用してスライドセンサの物理的分解能を倍加
  - 補間法を使用してスライドセンサの分解能を向上
  - 2 つのスライドセンサを使ったタッチパッド サポート
  - 高抵抗伝導性材料を使った検知サポート (ITO フィルムなど)
  - 水膜や水滴がある場合にも信頼できる動作をするシールド電極サポート
  - CSDADC ウィザードを使用し、ガイドされたセンサとピンの割り当て
  - 温度、湿度、静電気放電 (ESD) に対する、統合された基底値更新アルゴリズム
  - 調整が簡単な操作パラメータ
  - PC GUI アプリケーションを使用して、実際の Raw データをモニターしながらパラメタの最適化をリアルタイムで調整可能

CSDADC は、スイッチト キャパシタ電流をデジタル値に変換するデルタシグマ変調器を使用した、スイッチト キャパシタ手法による静電容量式検知を行います。

Figure 1. CSDADC ブロック ダイアグラム



クイック スタート

1. 専用ピンを必要とするユーザモジュールを選択・配置します（例：I2C と LCD）。ポートとピンを適宜割り当てます。
2. CSDADC ユーザ モジュールを選択、配置します。
3. CSDADC ユーザ モジュールを右クリックし、CSDADC ウィザードを開きます。
4. センサ数、コンフィグレーション、ピン割り当てを設定します。
5. ピンとグローバルパラメータを設定します。すべてのパラメータ記述を読み、条件とガイドラインに準拠します。
6. アプリケーションを生成し、Application Editor を開きます。
7. 個別のセンサ、スライド式センサ、またはタッチパッドを実装するために必要なサンプルコードを使用します。
8. RS232 レベル トランスレータまたは I2C-USB ブリッジをターゲット ボードに接続し、GUI を使ってパラメータを最適化します。
9. CSDADC パラメータを変更し、アプリケーションを再構築します。
10. PSoC デバイスをプログラムし、モジュール動作を確認します。5:1 S/N 比要件を満たすように CSDADC パラメータを調整してください。詳しくは CY8C21x34/B CapSense Design Guide を参照してください。

機能説明

CSDADC　は、機能、設定を個別にロード (して組み立て) しなくとも、静電容量センシングと AD コンバータによる電圧計測を " コンビ " で提供します。CSD と ADC に共通のモジュールを再利用することで、コード容量を節約します。静電容量式検知と ADC 機能の**両方**を必要する場合は、CSDADC を使用する必要があります。いずれかのみを必要とするアプリケーションには、CSD または DELSIG ADC を使用してください。

CSDADC は、検知用スイッチト キャパシタ電流をデジタル コードに変換するデルタシグマ変調器 (CSD) を使用した、スイッチト・キャパシタ手法による静電容量式検知を行います。CSDADC は、2 次

変調器と内蔵プリアンプを備えたデルタシグマ ADC を実装します。2 次変調器を使って、より優れた S/N 比を実現します。このユーザモジュールデータシートは、CSD 及び ADC 操作に関する基本的な情報を提供します。CSD パラメータの設定、CSD 利用のヒント、CSD のトラブルシューティングに関する詳しい詳細や推奨については、CSD、DELSIG、PGA のデータシート及び関連するアプリケーションノートを参照してください。CSD 及び ADC ユーザ モジュールを初めて使用する場合は、ソリューションを実装する前に、これらの文書をよくお読みください。

CSDADC ユーザ モジュールを初めて使う際は、その前に以下の文書を読むことをお勧めします。

- 「*CY8C24x94 Series PSoC Programmable System-on-Chip Technical Reference Manual*」の以下のセクション
  - 2 コラムのアナログ
  - デジタル クロック
  - I/O アナログ マルチプレクサ
- *Understanding Switched Capacitor Analog Blocks* - AN2041

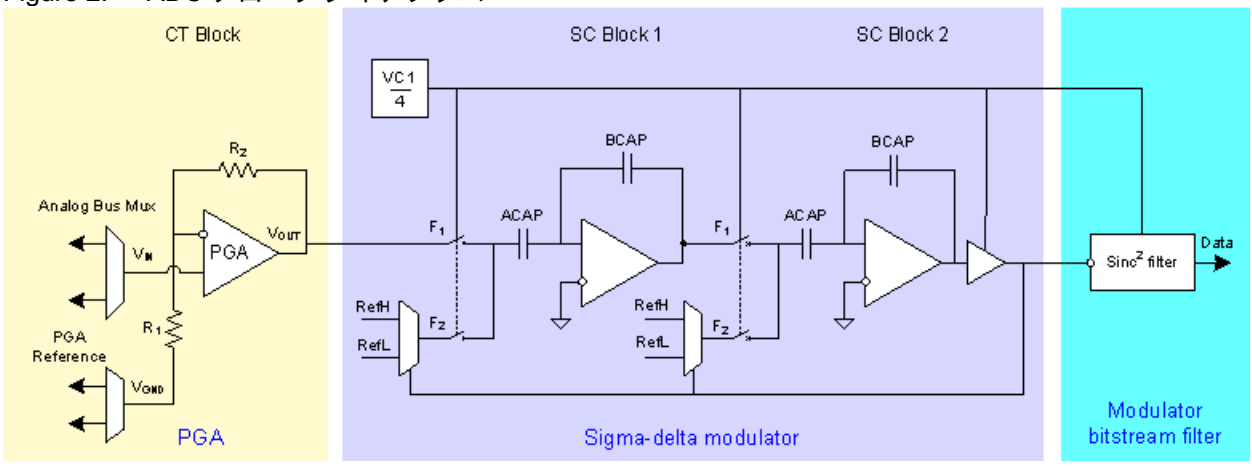
CSDADC ユーザモジュールデータシートを読んだあとに、次の設計ガイドを推奨します。これらの文書はサイプレス セミコンダクタのウェブサイト（ japan.cypress.com ）から探すことができます。

- Getting Started with CapSense
- CY8C20xx6A/H CapSense Design Guide
- CY8C21x34/B CapSense Design Guide
- CY8C20x34 CapSense Design Guide
- CY8CMBR2044 CapSense Design Guide

ADC 操作

CSDADC は、内蔵 PGA で DELSIG ADC を使用します。DELSIG ユーザ モジュールは、単一の出力サンプルを生成するために 64 ～ 256 の積分サイクルを必要とする、積分型変換器です。マルチプレックス入力を変更すると、変更後最初の 2 つのサンプルが無効になります。DELSIG ADC は、PGA、2 次変調器、Sinc^2 デシメーション フィルタという 3 つの主要モジュールから成ります。

Figure 2.　ADC ブロック ダイアグラム



プログラマブル ゲイン アンプ

ADC は、プログラマブル ゲイン アンプ (PGA) の前に配置されます。このアンプを使うと、入力信号範囲を ADC 信号範囲に一致させることにより、ADC ダイナミック レンジの利用率が向上します。PGA

入力は、アナログ バスから得るため、外部ソースに接続できます。内部信号ソースをサポートするよう再構成することもできます。

PGA 入力信号は、内部アナログ グランド、V_{ss}、その他の選択したリファレンスを基準にできます。プログラマブル ゲイン アンプのゲインは、レジスタアレイの選択可能なタップと、連続時間型アナログ PSoC ブロックのフィードバック タップをプログラムすることで設定されます。デバイス エディタでゲインとリファレンスを設定します。アンプは、以下の伝達関数を持ちます。

Equation 1

$$V_{OUT} = (V_{IN} - V_{GND}) \cdot \left(1 + \frac{R_2}{R_1}\right) + V_{GND}$$

以下のいずれかとして、リファレンスを指定できます。

- 内部リファレンスから派生した固定値
- 電源電圧に比例した値
- アナログ グランド
- 外部入力

アンプの入出力電圧範囲は、パワーサプライによって大きくなりません。(つまり、「レール to レール」オペアンプではありません)。許容入力範囲は、以下の組み合わせです。

- 入力制限
- 出力制限
- 電源電圧
- アナログ グランド値
- 選択したゲイン

変調器

変調器は 1-bit オーバーサンプリング回路で、それが生成する 1 及び 0 の密度を単位として入力電圧を表します。変調器の出力は、複数の 1-bit サンプルをより高い分解能のサンプルに変換するローパス デシメーション フィルタにより、最終のサンプリング速度に低減されます。一般に、より高い復号化レート (つまりより高いオーバーサンプリング速度) は、より高い分解能の結果を生成することができますが、変調器の次数などその他の要因も関係します。

デルタシグマ変換器の主な利点は、変調器が行う「ノイズシェーピング」です。　通常、信号サンプリングに特有の量子化ノイズは、「DC」とサンプル周波数の半分、ナイキスト周波数の間の周波数で、ほぼ均等に分散しています (「白」)。　簡単に言うと、デルタシグマ変調器は、量子化ノイズの一部を低周波数から高周波数にシフトします。これは後で、デシメーション フィルタによって減衰されます。2 つのスイッチト キャパシタ アナログ PSoC ブロックを必要とする 2 次変調器は、1 つのアナログ PSoC ブロックのみを必要とする 1 次変調器よりも、ノイズ シェーピングで優れています。256X のより高い復号化レートで 2 次変調器は、1 次変調器に比べ、実効分解能において 3.5-bit の増加が見られます。2 次変調器は、1 次変調器のアナログ出力を同様の PSoC ブロックにフィードし、2 つ目のブロックの 1-bit コンパレータ出力が前に示したように両方のブロックに戻るよう、フィードバック構成を変更することによって作成します。

アナログ コンパレータのバスは、アナログ PSoC ブロック アレイのコラムで垂直に走るため、2 次変調器のブロックは、ほかのものより 1 つ上に配置する必要があります。

DelSig ADC の範囲は、$\pm V_{Ref}$ によって規定されます。V_{Ref} は PSoC Designer ™のグローバルリソースウィンドウで設定します。固定スケールの場合、V_{Ref} は、$\pm V_{Bandgap}$ または $\pm 1.6\ V_{Bandgap}$ に設定されます。調整可能なスケールの場合は、V_{Ref} は ±Port 2[6] に設定されます。サプライレシオメトリックスケールでは、V_{Ref} は $\pm V_{DD}/2$ に設定されます。オプションの完全なリストを以下の表に示します。

Table 1. Ref Mux グローバル パラメータ設定の入力電圧範囲

RefMux の設定	V_{DD} = 5V	V_{DD} = 3.3V
$(V_{DD}/2) \pm$ BandGap	$1.2 < V_{in} < 3.8$	$0.35 < V_{in} < 2.95$
$(V_{DD}/2) \pm (V_{DD}/2)$	$0 < V_{in} < 5$	$0 < V_{in} < 3.3$
BandGap ± BandGap	$0 < V_{in} < 2.6$	$0 < V_{in} < 2.6$
(1.6*BandGap) ± (1.6*BandGap)	$0 < V_{in} < 4.16$	適用外
(2*BandGap) ± BandGap	$1.3 < V_{in} < 3.9$	適用外
(2*BandGap) ± P2[6]	$(2.6 - V_{P2[6]}) < V_{in} < (2.6 + V_{P2[6]})$	適用外
P2[4] ± BandGap	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$
P2[4] ± P2[6]	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$

Sinc^2 デシメーション フィルタ

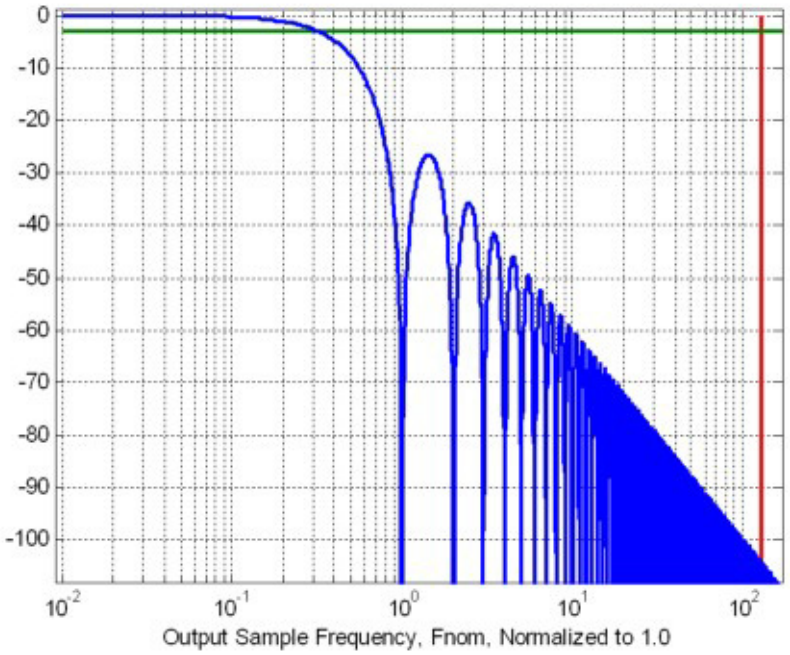
デシメーションフィルタの応答は、以下の z ドメイン関係によって得られます。

Equation 2

$$H(z) = \left[\frac{1 - z^{-n}}{1 - z^{-1}}\right]^2, \text{ where n is the decimation level.}$$

図 3 にプロットした周波数ドメイン伝達関数は、出力サンプリング速度 F_{nom} が 1.0 になるように正規化されます。-3 dB のポイントは $0.318 \times F_{nom}$ のすぐ上で発生し、関数は F_{nom} の各整数倍でゼロになります。1-bit サンプリングレートは定格出力レートよりも 64 ～ 256 高いため、ナイキスト制限は F_{nom} の 5～7 オクターブ上となり、アンチエイリアス フィルタの要件を著しく低減します。256 の復号化レートに対する 1-bit ナイキスト周波数は、グラフの右側に太い垂直線で示されています。さらに高い復号化レートも可能ですが、デバイスのノイズフロアのため、利点はほとんどありません。12-bit トポロジ、256 の復号化レートを持つ 2 次変調器の場合は、分解能が信号対雑音比により制限されます。

Figure 3. -3 dB 及びナイキスト周波数を使った Sinc^2 デシメーション フィルタの振幅特性



デシメータは、復号化レートの形成に別途タイマ ブロックを使わない、自己カウント モードで動作します。デシメータは、1-bit サンプリング速度で動作する二重積分器により、伝達関数の分母を実装します。分子は、定格出力サンプリング速度で動作する二重微分器 (2 つ目の差分演算子) によって実装されます。

静電容量測定の操作

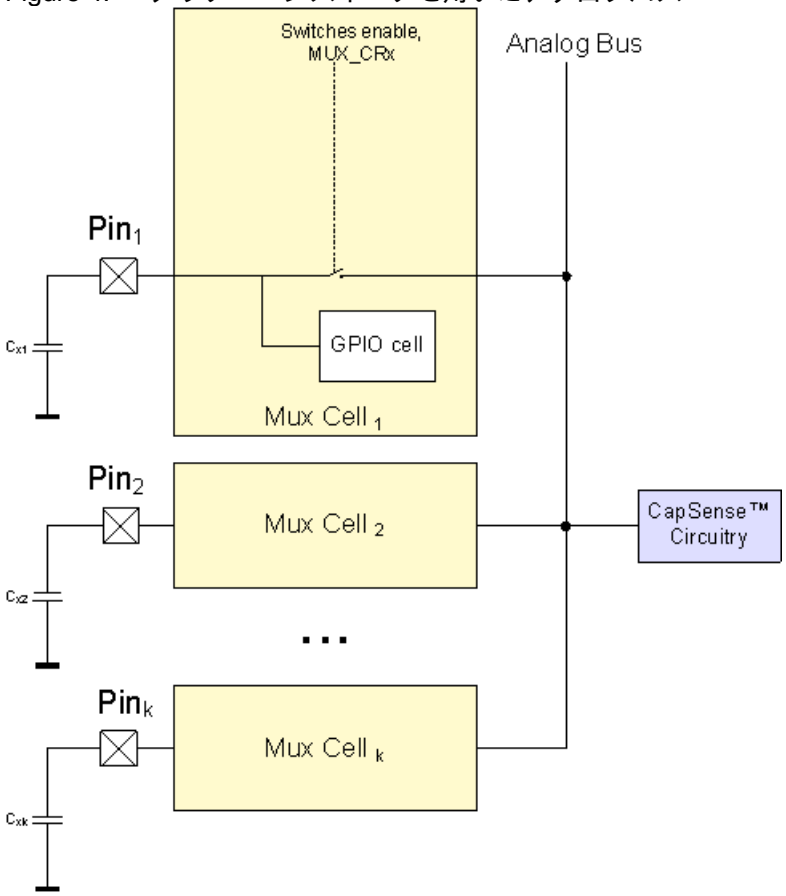
判定理論はファームウェアで実行されます。ファームウェアは、静電容量を分析し、環境要因による静電容量の変化をトラッキングし、判定理論を実行することにより、ボタンのタッチを検知し、スライダの位置を計算します。

センサのアレイをスキャンする

CY8C24x94 デバイスファミリは、2 つの内蔵アナログバスを持ちます。2 つのアナログ バスが互いに接続され、すべてのピンで接続されたセンサをスキャンします。CSDADC ユーザモジュールは、内部のプリチャージスイッチを使って、クロック信号フェーズ Ph_1 でアクティブなセンサを充電し、フェーズ Ph2 でアナログバスをセンサに接続します。Sigma-Delta 変調器の変調コンデンサとコンパレータの入力は、アナログバスに恒久的に接続されています。

ファームウェアは、MUX_CRx レジスタで該当するビットを設定して、センサのスキャンを実行します。

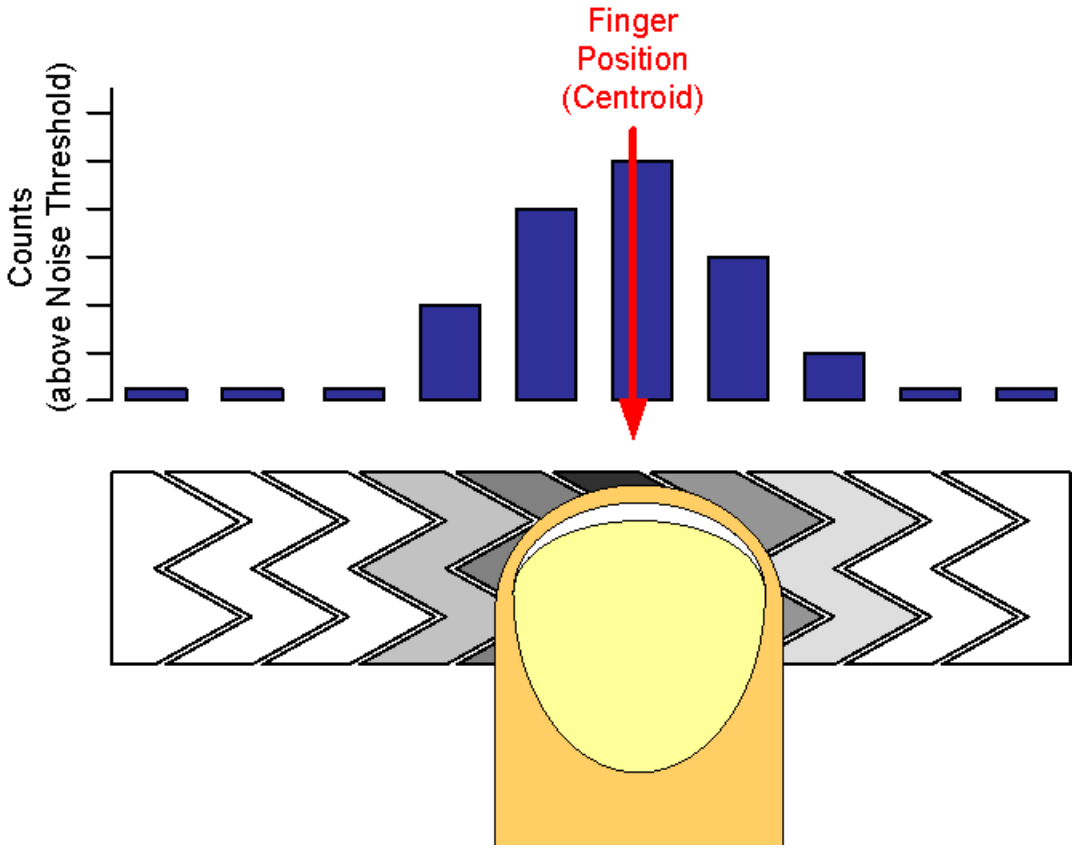
Figure 4.　プリチャージスイッチを用いたアナログバス



スライダ

スライダは、漸次的調整を必要とする制御に使用します。例としては、照明管理（調光装置）、音量管理、グラフィックイコライザ、速度管理などが挙げられます。これらのセンサは機械的に互いに隣接しており、1 つのセンサの作動は、物理的に近接するセンサの部分的な作動につながります。スライダの実際の位置は、作動したセンサ セットの重心位置を計算することによって判断できます。CSDADC ウィザードは、特定の順番を持つスライダ グループを設定することで、スライダに対応します。センサ スライダの実質的な下限は 5 で、上限は、選択した PSoC デバイスで利用できるセンサ数になります。

Figure 5.　物理的センサ位置の順序



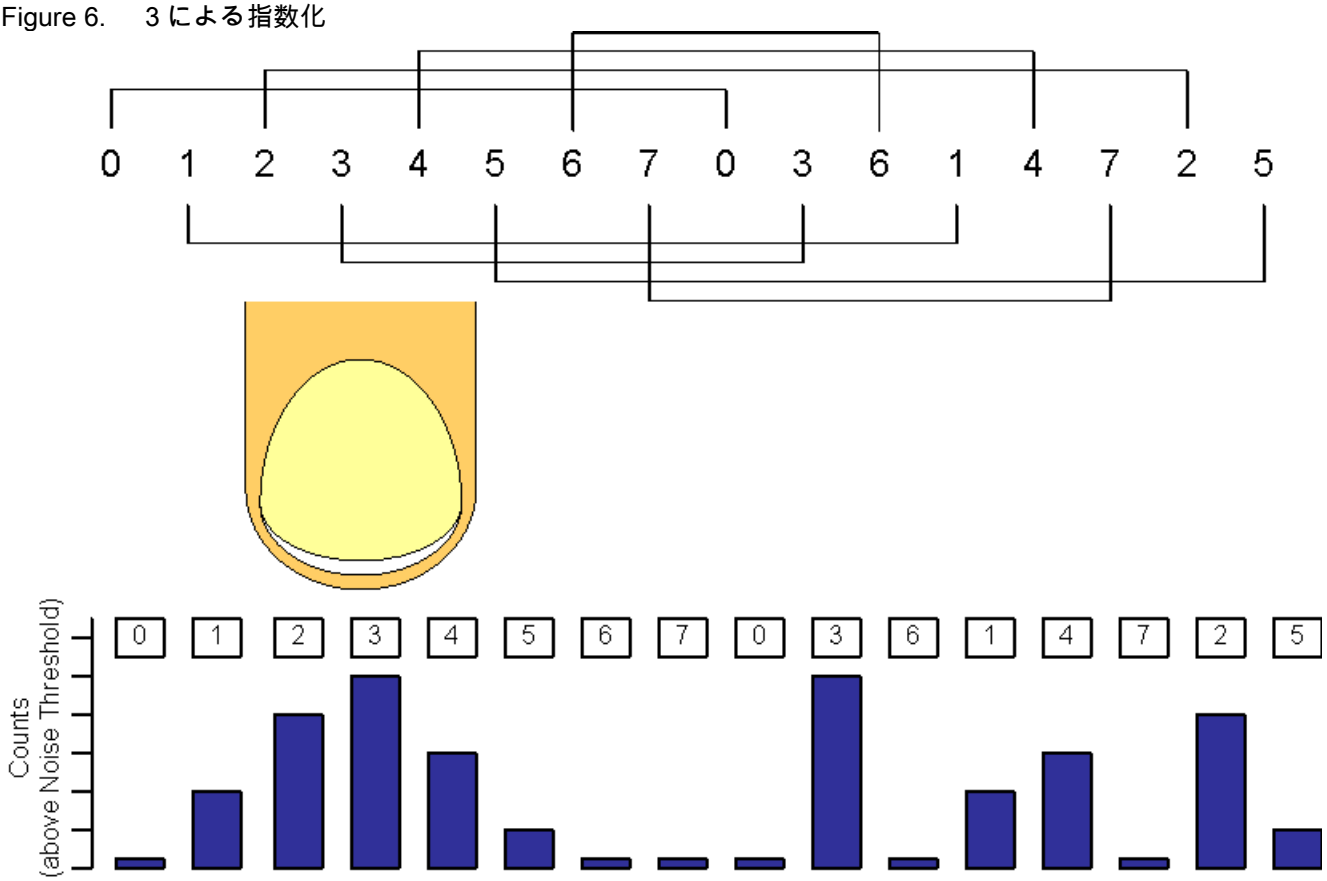
スライダの半分で強い信号を近接検知すると、残り半分に同レベルの信号を生じますが、結果は分散してしまいます。検知アルゴリズムは、強い近隣信号セットを検索して、スライダ位置を特定します。

ダイプレックス

スライダセンサの各センサは、PSoC の各ピンに 2 つづつ接続されます。物理的位置の最初の（もしくは数字が小さい）半分は、CSDADC ウイザードで設計者が割り当てたポートピンを使用して、ベース割り当てセンサに連続的にマッピングされます。残りの半分（数字が大きい）のスライダセンサーは、ウィザードのアルゴリズムによって自動的にマッピングされ、取り込みファイルに一覧表示されます。この順序は、半分内における近隣センサ起動が別の半分の近隣センサ起動を引き起こさないように設定されます。この順序の決定と、PCB 基板へのマッピングは慎重に行ってください。

物理センサ位置の後の半分に対して順序を決める方法は数多くあります。最も簡単な方法は、上半分のセンサのうち、偶数センサ全部を先に決定し、次に奇数センサ全部を決定するやり方です。他の方法では、別の指数値を使ってセンサを配置します。このユーザ モジュールで選択された方法は 3 による指数化です。

Figure 6. 3 による指数化



スライダ中のセンサ静電容量は均衡がとれていなければなりません。センサや PCB レイアウトによって、一部のセンサペアではセンサー配線が長くなる可能性があります。ダイプレックス センサ指数表は、ダイプレックスを選択すると CSDADC ウィザードによって自動的に作成されます。この表は、異なるスライダ セグメント カウントのダイプレックス シーケンスを示します。

Table 2. 異なるスライダ セグメント カウントのダイプレックス シーケンス

スライダ セグメントの総カウント	セグメント シーケンス
10	0,1,2,3,4,0,3,1,4,2
12	0,1,2,3,4,5,0,3,1,4,2,5
14	0,1,2,3,4,5,6,0,3,6,1,4,2,5
16	0,1,2,3,4,5,6,7,0,3,6,1,4,7,2,5
18	0,1,2,3,4,5,6,7,8,0,3,6,1,4,7,2,5,8
20	0,1,2,3,4,5,6,7,8,9,0,3,6,9,1,4,7,2,5,8
22	0,1,2,3,4,5,6,7,8,9,10,0,3,6,9,1,4,7,10,2,5,8
24	0,1,2,3,4,5,6,7,8,9,10,11,0,3,6,9,1,4,7,10,2,5,8,11

スライダ セグメントの総カウント	セグメント シーケンス
26	0,1,2,3,4,5,6,7,8,9,10,11,12,0,3,6,9,12,1,4,7,10,2,5,8,11
28	0,1,2,3,4,5,6,7,8,9,10,11,12,13,0,3,6,9,12,1,4,7,10,13,2,5,8,11
30	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,0,3,6,9,12,1,4,7,10,13,2,5,8,11,14
32	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0,3,6,9,12,15,1,4,7,10,13,2,5,8,11,14
34	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,0,3,6,9,12,15,1,4,7,10,13,16,2,5,8,11,14
36	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,0,3,6,9,12,15,1,4,7,10,13,16,2,5,8,11,14,17
38	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,0,3,6,9,12,15,18,1,4,7,10,13,16,2,5,8,11,14,17
40	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,0,3,6,9,12,15,18,1,4,7,10,13,16,19,2,5,8,11,14,17
42	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,0,3,6,9,12,15,18,1,4,7,10,13,16,19,2,5,8,11,14,17,20
44	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,0,3,6,9,12,15,18,21,1,4,7,10,13,16,19,2,5,8,11,14,17,20
46	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,0,3,6,9,12,15,18,21,1,4,7,10,13,16,19,22,2,5,8,11,14,17,20
48	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,3,6,9,12,15,18,21,1,4,7,10,13,16,19,22,2,5,8,11,14,17,20,23
50	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,0,3,6,9,12,15,18,21,24,1,4,7,10,13,16,19,22,2,5,8,11,14,17,20,23
52	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,0,3,6,9,12,15,18,21,24,1,4,7,10,13,16,19,22,25,2,5,8,11,14,17,20,23
54	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,0,3,6,9,12,15,18,21,24,1,4,7,10,13,16,19,22,25,2,5,8,11,14,17,20,23,26
56	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,0,3,6,9,12,15,18,21,24,27,1,4,7,10,13,16,19,22,25,2,5,8,11,14,17,20,23,26
58	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,0,3,6,9,12,15,18,21,2,27,1,4,7,10,13,16,19,22,25,28,2,5,8,11,14,17,20,23,26
60	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,0,3,6,9,12,15,18,21,24,27,1,4,7,10,13,16,19,22,25,28,2,5,8,11,14,17,20,23,26,29
62	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,0,3,6,9,12,15,18,21,24,27,30,1,4,7,10,13,16,19,22,25,28,2,5,8,11,14,17,20,23,26,29
64	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,0,3,6,9,12,15,18,21,24,27,30,1,4,7,10,13,16,19,22,25,28,31,2,5,8,11,14,17,20,23,26,29
66	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,0,3,6,9,12,15,18,21,24,27,30,1,4,7,10,13,16,19,22,25,28,31,2,5,8,11,14,17,20,23,26,29,32

スライダ セグメントの総カウント	セグメント シーケンス
68	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,0,3,6,9,12,15,18,21,24,27,30,33,1,4,7,10,13,16,19,22,25,28,31,2,5,8,11,14,17,20,23,26,29,32
70	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,0,3,6,9,12,15,18,21,24,27,30,33,1,4,7,10,13,16,19,22,25,28,31,34,2,5,8,11,14,17,20,23,26,29,32
72	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,0,3,6,9,12,15,18,21,24,27,30,33,1,4,7,10,13,16,19,22,25,28,31,34,2,5,8,11,14,17,20,23,26,29,32,35
74	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,0,3,6,9,12,15,18,21,24,27,30,33,36,1,4,7,10,13,16,19,22,25,28,31,34,2,5,8,11,14,17,20,23,26,29,32,35
76	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,0,3,6,9,12,15,18,21,24,27,30,33,36,1,4,7,10,13,16,19,22,25,28,31,34,37,2,5,8,11,14,17,20,23,26,29,32,35
78	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,0,3,6,9,12,15,18,21,24,27,30,33,36,1,4,7,10,13,16,19,22,25,28,31,34,37,2,5,8,11,14,17,20,23,26,29,32,35,38
80	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,0,3,6,9,12,15,18,21,24,27,30,33,36,39,1,4,7,10,13,16,19,22,25,28,31,34,37,2,5,8,11,14,17,20,23,26,29,32,35,38
82	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,0,3,6,9,12,15,18,21,24,27,30,33,36,39,1,4,7,10,13,16,19,22,25,28,31,34,37,40,2,5,8,11,14,17,20,23,26,29,32,35,38
84	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,0,3,6,9,12,15,18,21,24,27,30,33,36,39,1,4,7,10,13,16,19,22,25,28,31,34,37,40,2,5,8,11,14,17,20,23,26,29,32,35,38,41
86	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,1,4,7,10,13,16,19,22,25,28,31,34,37,40,2,5,8,11,14,17,20,23,26,29,32,35,38,41
88	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,1,4,7,10,13,16,19,22,25,28,31,34,37,40,43,2,5,8,11,14,17,20,23,26,29,32,35,38,41
90	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,1,4,7,10,13,16,19,22,25,28,31,34,37,40,43,2,5,8,11,14,17,20,23,26,29,32,35,38,41,44
92	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,1,4,7,10,13,16,19,22,25,28,31,34,37,40,43,2,5,8,11,14,17,20,23,26,29,32,35,38,41,44

ダイプレクススライダ用のスライダセグメント選択ガイドライン

スライダに必要なセグメント数の選択は、主にスライダの物理的な長さに基づきます。しかし、ダイプレクススライダ用にセグメント数を決定する際には特別注意する必要があります。

ダイプレクススライダ設計において、1 つのセンサは、スライダの物理的な長さを増やすために 2 つの異なる物理的スライダセグメントとして使用されます。1 回の指タッチによって完全にカバーされるセグメント数は同一のセンサから派生した 2 つのセグメント間におけるセンサ数よりも少なくなければなりません。これによってダイプレクススライダが正常に機能します。

例えば、10-segment スライダ（5 つのセンサ）の場合、センサ 3 から派生した 2 つのスライダセグメントは 2 つのセンサ（センサ 4 と 0）によってのみ分離されます。この場合、スライダを正常に機能させるために、1 回の指タッチが 3 つ以上のセンサセグメントを完全にカバーしてはなりません。

12-segment スライダについては、1 回の指タッチが 4 つ以上のセグメントをカバーしてはなりません。同様に、18-segment スライダについては、1 回の指タッチが 5 つ以上のセグメントを完全にカバーしてはなりません。

補間とスケーリング

多くの場合、スライド式センサとタッチパッド用のアプリケーションでは、個々のセンサのネイティブピッチよりも高い分解能を得られるように指 (またはその他の静電容量性物体) の位置を特定する必要があります。　スライド式センサやタッチパッドで指が触れる工エリアは、しばしば 1 個のセンサより大きくなっています。

重心を使用した補間位置の計算では、まずアレイをスキャンして、センサの位置が有効であることを確認します。ここでは、近隣センサ信号のある番号がノイズ閾値を超えていることが要件となります。最も強い信号が見つかったら、その信号と、ノイズ閾値より大きい近隣信号を使用して重心を算出します。重心は (通常)、2 つから 8 つのセンサを使用して、次の数式で計算されます。

Equation 3

$$N_{Cent} = \frac{n_{i-1}(i-1) + n_i i + n_{i+1}(i+1)}{n_{i-1} + n_i + n_{i+1}}$$

通常、計算結果は整数ではありません。たとえば 12 個のセンサに対して 0 ~ 100 という範囲である場合、重心を特定の分解能の形で報告するには、計算されたスカラー量を重心に掛けます。1 つの計算で補間とスケーリングを組み合わせ、その結果を直接、必要なスケールで報告する方が効率的です。これは高レベル API で行う処理です。

スライダセンサ カウントと分解能は、CSDADC ウィザードで設定します。スケーリング値は、ウィザードで計算され、整数ではない値として保存されます。

重心の分解能の乗数は、3 バイトに含まれ、それぞれのビット定義は以下になります。

分解能乗数 MSB								
ビット	7	6	5	4	3	2	1	0
乗数	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
分解能乗数 ISB								
乗数	128	64	32	18	16	8	4	2
分解能乗数 LSB								
乗数	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256

分解能はこの数式を用いて算出されます。

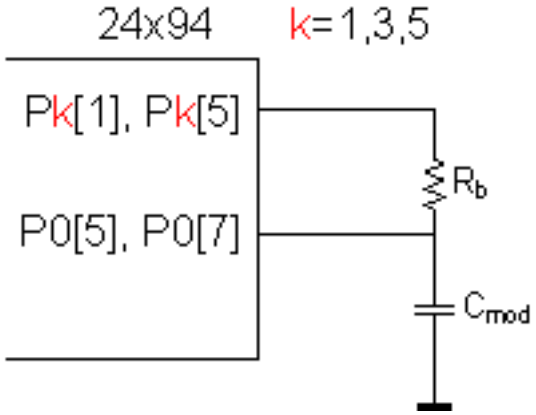
分解能 = (センサ数 – 1) x 乗数

重心は 24-bit 符号無し整数で保持され、その分解能はセンサ数と乗数の関数です。

フィードバックコンポーネント選択のガイドライン

ユーザモジュールには、外付け変調コンデンサ C_{mod} とモジュレータフィードバック レジスタ R_b が必要です。　コンデンサは、P0[5]、P0[7] ポートピンと Vss アースに接続できます。フィードバックレジスタ R_b はポートピン P1[1]、P1[5]、P3[1]、P3[5]、P5[1]、P5[5] およびコンデンサピンに接続できます。ピンは、ユーザ モジュールのパラメータ設定により選択されます。変調器コンポーネントの接続用に選択されているピンは、これ以外の目的で使用しないでください。LCD や I2CHW などの特定のピンを必要とするユーザ モジュールは、CSDADC ユーザ モジュールのポート ピン接続を確立する前に配置しなければなりません。構成の選択は、ウィザードを開いたときに表示されます。

Figure 7.　外部コンポーネントの接続



変調コンデンサ用に推奨されている値は、4.7 ~ 47.0 nF です。最適な静電容量は、最大の S/N 比を得るための実験を行うことで決定することができます。ほとんどの場合、5.6 ~ 10 nF の値で良い結果が得られます。**注** プリスケーラを使った構成を使用する場合は、より大きい変調コンデンサが必要となります。ほとんどの場合、高品質の 100 nF コンデンサで十分です。フィードバック レジスタを選択した後で、最良の S/N 比を得るためには、いくつかのコンデンサ値で実験してみるとよいでしょう。セラミック コンデンサを使用する必要があります。温度静電容量係数は重要ではありません。抵抗値は、総センサ静電容量 C_s によって異なります。抵抗値は、次の方法で選択します。

- 異なるセンサのタッチの Raw カウントをモニターする。
- 選択されたスキャン分解能でフルスケール読み値より約 30% 小さい最大読み値を提供する抵抗値を選択する。抵抗値が減ると、Raw カウントは増えます。

標準値は 500Ω ~ 10 kΩ で、センサの静電容量とプリチャージ スイッチの動作周波数によって変わります。CY3214 評価ボードを使用している場合は、2.0kΩ から開始できます。

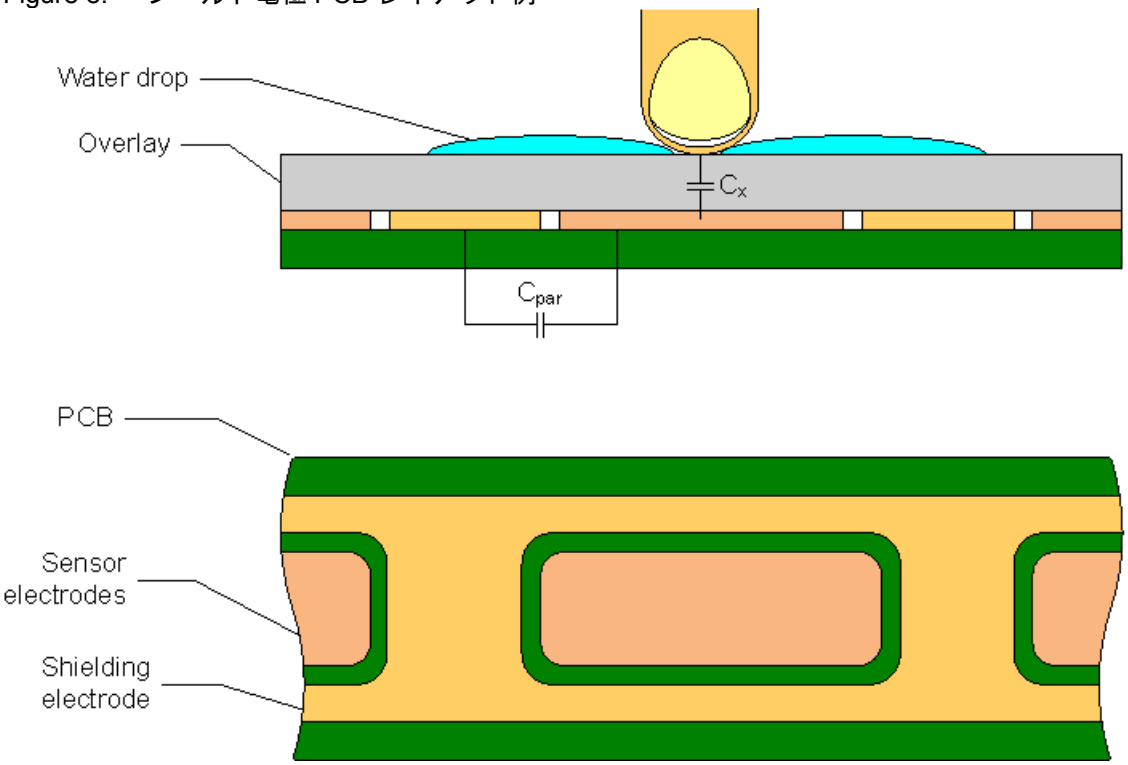
シールド電極

一部のアプリケーションでは、水膜や水滴がある場合でも動作の信頼性が要求されます。白物家電、車載アプリケーション、様々な産業用アプリケーションなどでは、水、氷、湿度変化があっても誤動作がない静電容量式センサが必要です。この場合、別個のシールド電極を使用することができます。この電極は検知電極の背部または外部に装備します。水膜がデバイスの遮断オーバーレイの表面にある場合、シールドと検知電極のカップリングが増えます。シールド電極は、寄生静電容量の影響を低減し、検知静電容量の変化の処理をするダイナミックレンジを広げます。

一部のアプリケーションでは、電極間のカップリングを増やして、検知電極の静電容量測定値のタッチ変化の逆を発生させるため、シールド電極信号と検知電極に対するその相対的位置を適切に選ぶことは

有効です。これにより、高レベルソフトウェアの API 作業が簡単になります。CSDADC ユーザ モジュールは、シールド電極の個々の出力に対応します。

Figure 8.　シールド電極 PCB レイアウト例



前の図は、ボタンのシールド電極のレイアウト構成例を示しています。シールド電極は、LCD ドライブ電極のノイズの影響を阻止し、同時に浮遊静電容量を低減するため、透明な ITO タッチパッドデバイスでは特に有用です。

この例では、ボタンはシールド電極平面で覆われています。代替案として、ボタンの下のプレーンなど、PCB の反対側のレイヤに置くことも可能です。この場合、充填率約 30 ~ 40% で、ハッチパターンを使用することが推奨されます。ここでは、グランド プレーンを追加する必要はありません。

水滴がシールドと検知電極の間にある場合、C_{par} が増え、変調器の電流が低減することがあります。実際のテストでは、水滴による Raw カウント上昇はゼロに近いか、わずかにマイナスとなる必要があるため、変調器の基準電圧は API によって上げることが可能です。これは、適切な変調器基準値を選択することによって達成できます。

シールド電極は、迂回可能ないずれの PSoC ピンにも接続できます。駆動モードを **Strong Slow** に設定し、アース ノイズと放射妨害波を軽減します。さらに、スルーを制限するレジスタを、PSoC デバイスとシールド電極の間に接続することができます。

クロック源

クロック源は、検知キャパシタ上でスイッチの制御に使用されます。ユーザ モジュールは、プリチャージスイッチのクロック源として、次の 4 つの選択オプションをサポートしています。

以下の表で、4 つの構成を比較します。

構成	動作周波数	使用するデジタル ブロック	EMC ノイズイミュニティ
PRS16	スペクトラム拡散。平均は $F_{IMO}/4$、ピークは $F_{IMO}/2$。	3	高。高感度ポイントは、PRS シーケンスの繰り返し期間の倍数で、PRS 基本周波数は F_{IMO}。
PRS8	調整可能なスペクトラム拡散、 $F_{IMO}/4 – F_{IMO}/512$	2	中。PRS 繰り返し期間が短くなるため、より多くのポイントで検知。高い EMI 耐性が必要な場合は、PRS16 構成を使用
Prescaler (プリスケーラ)	固定、IMO/(期間 + 1)	1	デバイスは、動作周波数とその調和周波数の EMC 信号に敏感です。高抵抗素材を使用した操作のみ推奨。
VC2	固定、IMO/(VC_1xVC_2)	0	デバイスは、動作周波数とその調和周波数の EMC 信号に敏感です。認定 EMC/EMI テストが予定されていない場合のみ推奨。

DC 電気的特性と AC 電気的特性

Table 3. 電源電圧

パラメータ	最小値	標準値	最大値	単位	テスト条件とコメント
値	3.0	5.0	5.25	V	

Table 4. 標準ノイズ

パラメータ[a]	高速	通常	低速	単位	テスト条件 (V_{DD} = 5V、SysClk = 24 MHz, CPU Clock = 12 MHz、 基準値 >= 分解能最大カウントの 70%) ゲイン = 4
ノイズカウント、 ピーク - ピーク	1	1	1	ピーク - ピーク	分解能 = 14
ノイズカウント、 ピーク - ピーク	1	2	2	ピーク - ピーク	分解能 = 12
ノイズカウント、 ピーク - ピーク	3	3	4	ピーク - ピーク	分解能 = 10

a. S/N 比は、スキャン速度が遅く、ベースライン値が増加するにしたがってよくなります。

Table 5. 消費電力

電源電圧	最小値	標準値	最大値	単位	テスト条件とコメント
アクティブ電流		10		mA	スキャン中の平均電流、8 センサ
スタンバイ電流		250		µA	スキャン速度 = 超高速、分解能 = 9 100 ms レポートレート、8 センサ
		1.6		mA	スキャン速度 = 高速、分解能 = 12 100 ms レポートレート、8 センサ
スリープ / ウェイク電流		10		µA	1s レポート レート、1 センサ

Table 6. 5.0V PGA DC の電気的特性

パラメータ	標準値	上限 / 下限	単位	条件および注記
公称値からのゲイン偏差				
G=48.00	3.0	--	%	
G=24.00	2.2	--	%	
G=16.00	1.5	--	%	
G=4.00	0.7	--	%	
G=1.0	0.5	--	%	
入力				
入力オフセット電圧	4.5	--	mV	
Input 電圧範囲	--	Vss ~ V_{DD}	V	
漏れ電流 [1]	1	--	nA	
入力容量 [1]	3	--	pF	
出力スイング	0.05 ~ V_{DD}-0.05	--	V	
PSRR	73	--	dB	

Table 7. 5.0V PGA AC の電気的特性

パラメータ	標準値	上限 / 下限	単位	条件および注記
スルー レート (20% ~ 80%)[2]				
Low 出力	0.6	--	V/µs	
Med 出力	2.5	--	V/µs	
High 出力	9.5	--	V/µs	
整定時間				

パラメータ	標準値	上限 / 下限	単位	条件および注記
Low 出力	13	--	µs	
Med 出力	4	--	µs	
High 出力	1	--	µs	
ノイズ [2]				入力換算
Med 出力	110		nV/√Hz	高出力（High Power）を除きオペアンプ バイアスは LOW。リファレンス入力は AGND に設定。
High 出力	100		nV/√Hz	

Table 8. 3.3V PGA DC の電気的特性

パラメータ	標準値	上限 / 下限	単位	条件および注記
公称値からのゲイン偏差				
G=48.00	4.0	--	%	
G=24.00	2.2	--	%	
G=16.00	1.2	--	%	
G=4.00	0.6	--	%	
G=1.0	0.3	--	%	
入力				
Input オフセット電圧	3.5	--	mV	
Input 電圧範囲	--	Vss ~ V_{DD}	V	
Leakage[1]	1	--	nA	
Input 静電容量 [1]	3	--	pF	
出力スイング	0.05 ~ V_{DD}-0.05	--	V	
PSRR	68	--	dB	
動作電流				
Low 出力	130	--	µA	
Med 出力	520	--	µA	
High 出力	2000	--	µA	

Table 9. 5.0VxA0;ADC 変調器の DC 電気的特性と AC 電気的特性

パラメータ	標準値	上限 / 下限	単位	条件および注記
入力				
Input 電圧範囲	---	Vss ~ V_{DD}	V	Ref Mux = $V_{DD}/2 \pm V_{DD}/2$
Input 静電容量	3	---	pF	入出力ピンを含みます。
Input インピーダンス	1/(C*clk)	---	W	
有効な分解能				
Decimate by 64	---	10	ビット	
Decimate by 128		12		
Decimate by 256		14		
サンプリングレート				
Decimate by 64	---	31,250	sps	データ クロック 8 MHz
Decimate by 128		15625		
Decimate by 256		7812		
DC 精度				
DNL				
Decimate by 64	<1	---	LSB	ソース クロック 1.5 MHz
Decimate by 128	<1			
Decimate by 256	0.6			
Offset エラー	13	---	mV	
Gain エラー	2		% FSR	リファレンス ゲイン誤差を含む
データ クロック	---	0.032 ~ 8.0	MHz	デジタル ブロックへの入力およびアナログ コラムのクロック

Table 10. 3.3V ADC 変調器の DC 電気的特性と AC 電気的特性

パラメータ	標準値	上限 / 下限	単位	条件および注記
入力				
Input 電圧範囲	---	Vss ~ Vdd	V	Ref Mux = $V_{DD}/2 \pm V_{DD}/2$
Input 静電容量	3	---	pF	入出力ピンを含みます。
Input インピーダンス	1/(C*clk)	---	W	
有効な分解能				

パラメータ	標準値	上限 / 下限	単位	条件および注記
Decimate by 64	---	10	ビット	
Decimate by 128		12		
Decimate by 256		14		
サンプリングレート				
Decimate by 64	---	31,250	sps	データ クロック 8 MHz
Decimate by 128		15625		
Decimate by 256		7812		
DC 精度				
DNL				
Decimate by 64	<1	---	LSB	データ クロック 1.5 MHz
Decimate by 128	<1			
Decimate by 256	0.5			
Offset エラー	13	---	mV	
Gain エラー	2		% FSR	リファレンス ゲイン誤差を含む
データ クロック	---	0.032 ~ 8.0	MHz	デジタル ブロックへの入力およびアナログ コラムのクロック

配置

ユーザ モジュールのブロックは、ユーザ モジュールがインスタンス化されると自動的に配置されます。他の配置は利用できません。変調器コンパレータは、ACB01 連続時間型ブロックと ASD11/ASD21 スイッチト キャパシタ ブロックに配置されます。0 ー 3 個のデジタル ブロックを使用する、異なる UM 構成があります。

この表に、使用されるデジタル リソースを示します。

構成	使用するデジタル ブロック
PRS16	3
PRS8	2
Prescaler (プリスケーラ)	1
VC2 クロック源	0

使用されないアナログ ブロック及びデジタル ブロックは、ほかの目的に利用できます。すべての UM 構成で、ハードウェア デシメータが使用されます。

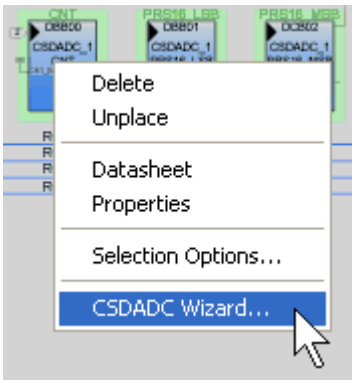
LCD や I2CHW などの特定のピンを必要とするユーザ モジュールは、CSDADC ユーザ モジュールのポート ピン接続を確立する前に配置しなければなりません。構成の選択は、ウィザードを開いたときに反映されます。

静電容量式センサの接続を配置する場合、P1[0] と P1[1] は避けてください。これらのピンは、そのデバイスのプログラミングに使用され、センサの検出感度とノイズに影響を与える過度のルーティング静電容量を持つ可能性があります。

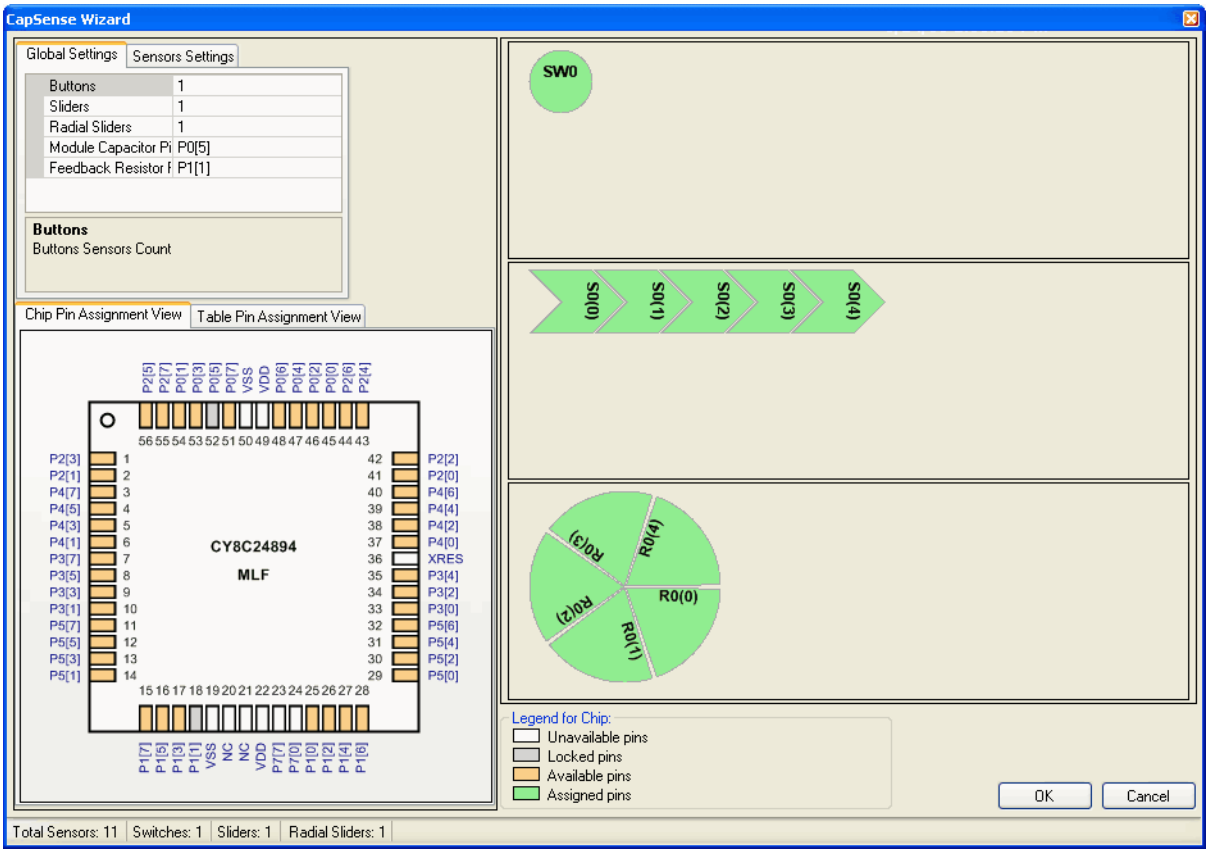
ウィザード

CSDADC ウィザードを使って、CapSense ボタン、スライダ、近接検知センサのピン配列を設定します。ドラッグ アンド ドロップ インタフェースを使って、構成を選択し、ボタンとセグメントを割り当てます。

1. ウィザードにアクセスするには、デバイス エディタの相互接続ビューで CSDADC の任意のブロックを右クリックし、次に [CSDADC Wizard] (CSDADC ウィザード) を左クリックします。



2. ウィザードが開き、センサの数とスライダセンサの数がボックスに示されます。



ウィザードのピン凡例

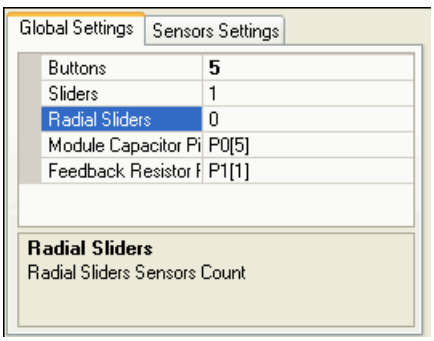
白 – このピンは CapSense の入力に使用できません。

グレー – このピンはロックされています。これには 2 つの原因が考えられます。その 1 つ目は、LCD や I^2C などの別のユーザ モジュールが、そのピンを使用している場合です。2 つ目は、ピンの名前がデフォルトから変わった場合です。ピン名をデフォルトに戻すには、Pinout ビューでそのピンの表示を広げ、**Select** メニューで **Default** を選択します。これで、ピンはウィザードで割り当てられるようになりました。

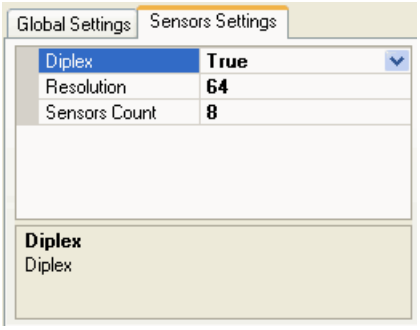
オレンジ – このピンは割り当て可能です。

グリーン – このピンは CapSense 入力に割り当てられています。

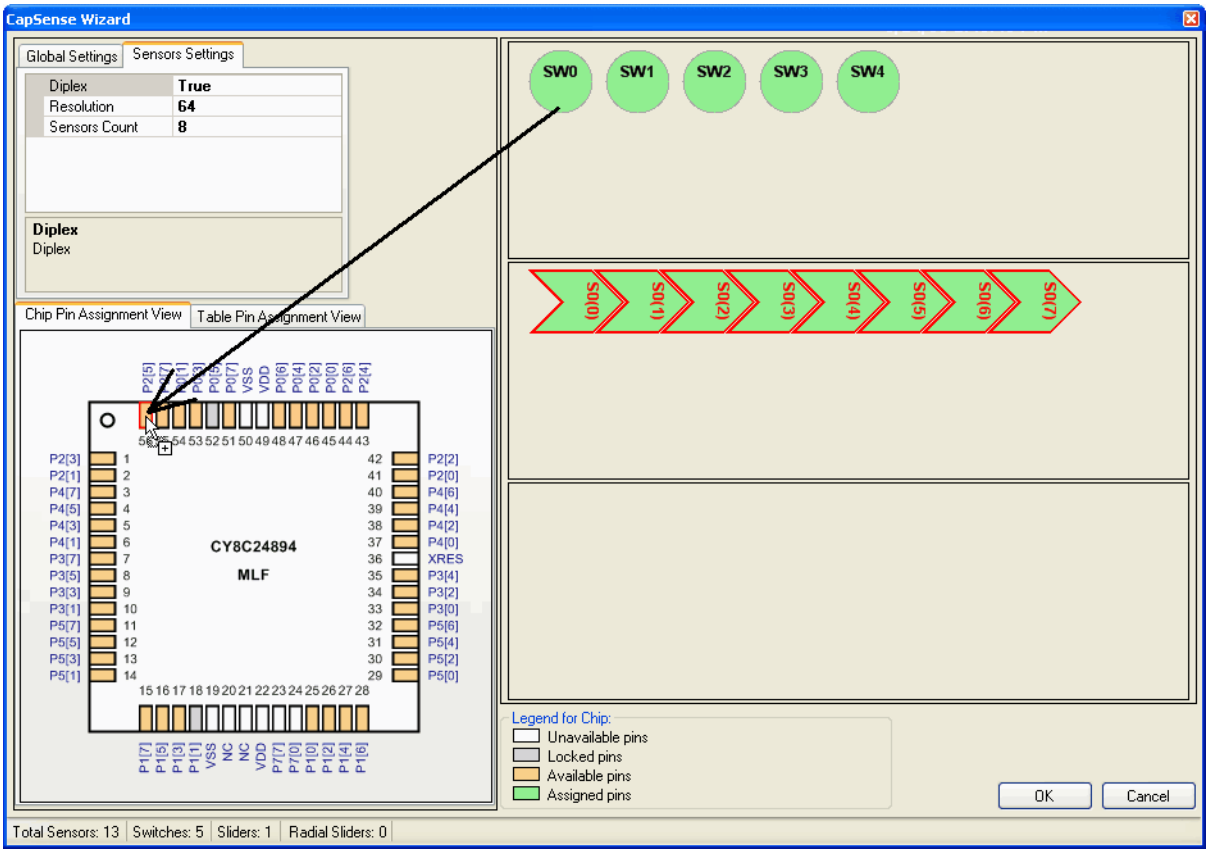
3. 独立したボタン、スライダ、ラジアル スライダの数を入力します。センサの総数は、使用できるピン数に制限されています。X-Y タッチパッドには 2 つのスライダが必要ですが、選択されるのは 1 つです。



4. [Sensor Settings] (**センサの設定**) タブを選択し、センサ数と、スライダ及びラジアル スライダに対するその他の設定を指定します。各スライダのセンサ数を入力します。スライダセンサ中の現実的な最低センサ数は 5 で、最大数はピン数によって制限されます。

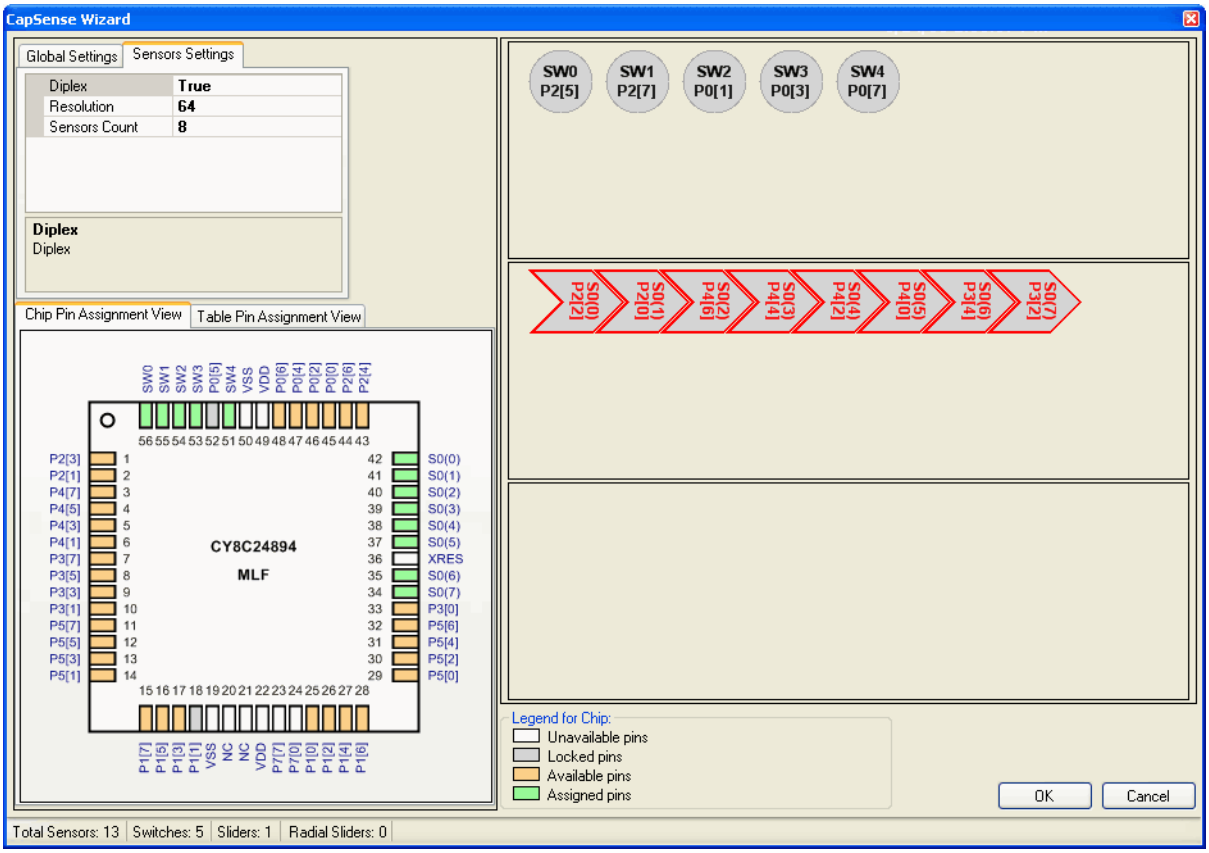


5. 出力の分解能を入力します。最低値は 5 です。最大値は、(センサ用ピン数 - 1) x 2^8、ダイプレックススライダの場合、(2 x センサ用ピン数 - 1) x 2^8 です。
6. 必要に応じて、ダイプレックスを選択します。これにより、センサ用に選択されたピンを、基板上で 2 倍の数のセンサ位置にマッピングできます。上図では、ダイプレックスセンサの上半分だけが示されています。下半分は、前述の 「ダイプレックス」の項での説明の通り、自動的にマッピングされます。 「ダイプレックス」の項で、ピン説明のダイプレックス表を参照してください。
7. ボタンを左クリックし、利用可能なピンにドラッグします。ポートピンは一旦選択するとグリーンになり、使用不可となります。ポート ピンからボタンをドラッグして外すと、センサの割り当てを変更できます。



8. 他のボタンでも同じ操作を繰り返します。

9. 個々のスライダセンサを、ボタンの場合と同様に、物理ポート ピンにマッピングします。
10. [OK] をクリックしてデータを受け入れ、PSoC Designer に戻ります。



これでセンサの配置が完了しました。Device Editor ウィンドウを右クリックし、[**Refresh**] (再表示) を選択すると、ピン接続が更新されます。

ユーザ モジュールのパラメータを選択し、アプリケーションを生成します。必要に応じて、サンプルプロジェクトを使用することもできます。

CSDADC ウィザードで数値を入力する場合は、まず古い値を削除してから、新しい値を入力してください。編集ボックスには、カーソルは表示されません。

ピン割り当てを変更するには、割り当てられているピンにカーソルを合わせてクリックし、それをスイッチボックスの外までドラッグ アンド ドロップします。これでこのピンは割り当てから外され、他に割り当てることが可能になります。

ウィザードを完了したら、[Generate Application] (アプリケーションの生成) をクリックします。入力したセンサ数、ピン割り当て、ダイプレックス、分解能に基づいて、一連の表が生成されます。これらの表は CSDADC_Table.asm に保存されています。

センサ表

センサ表は各センサに対して 2-byte エントリから構成されています。第 1 バイトはポート番号で、第 2 バイトはそのビットのビットマスクです (ビット番号ではありません)。表には、すべての独立したセンサ、次に各センサが順番に列挙されています。次に、10 のセンサを含む表の例を示します。

```
CSDADC_Sensor_Table:
_CSDADC_Sensor_Table:
    dw    0x0001    // Port 0 Bit 0
    dw    0x0002    // Port 0 Bit 1
    dw    0x0004    // Port 0 Bit 2
    dw    0x0008    // Port 0 Bit 3
    dw    0x0010    // Port 0 Bit 4
    dw    0x0101    // Port 1 Bit 0
    dw    0x0102    // Port 1 Bit 1
    dw    0x0104    // Port 1 Bit 2
    dw    0x0108    // Port 1 Bit 3
    dw    0x0110    // Port 1 Bit 4
```

この表は CSDADC_wGetPortPin() ルーチンで使用されます。

グループ表

グループ表は、ボタンセンサやスライダのグループを定義します。各スライダにつきエントリが 1 つ、フリーボタンセンサにもエントリが 1 つあります。最初のエントリは必ずフリーセンサです。各エントリは 6 バイトです。第 1 バイトはセンサ表のインデックスです。第 2 バイトはグループ内のセンサ数です。第 3 バイトは、スライダがダイプレックスであるかどうかを示します (4 はダイプレックス、0 は非ダイプレックス)。第 4、第 5、第 6 バイトは固定小数点乗数であり、スライダの計算された重心に乗算されて、CSDADC ウィザードで要求された分解能を達成します。

```
CSDADC_1_Group_Table:
_CSDADC_1_Group_Table:
; Group Table:
;    Origin    Count    Diplex    DivBtwSw(wholeMSB, wholeLSB, fractByte)
 db   0x0,      0x5,     0x00,     0x00,     0x00,     0x00 ; Buttons
 db   0x5,      0x5,     0x3,      0x0,      0x0,      0x71 ; Slider 1
```

ダイプレックス表

ダイプレックス表のスキャンオーダデータは、スライダでダイプレックスされている場合に、グループを対象として作成されます。これ以外の場合は、ラベルが作成されますが、データは記録されません。この表は、各スライダのセンサマッピングと、各スライダによるそれぞれの表のリファレンスという 2 つの部分から構成されています。ここでは、5 つのセンサ スライダを使った典型的な例を示します。

```
DiplexTable_0:
; This group is not a diplexed slider
DiplexTable_1:
    db    0,1,2,3,4,0,3,1,4,2    // 5 switch slider


 CSDADC_1_Diplex_Table:
_CSDADC_1_Diplex_Table:
    db >DiplexTable_0, <DiplexTable_0
    db >DiplexTable_1, <DiplexTable_1
```

パラメータおよびリソース

ADCEnabled

ADCEnabled パラメータは 以下の 2 つの値を持つことができます。

- Enabled (有効 : デフォルト値)
- Disabled (無効)

パラメータを Enabled に設定すると、ADC ルーチンがユーザ モジュールの互換コードに含められ、ユーザのコードから呼び出すことができるようになります。このパラメータを Disabled に設定すると、条件付きコンパイル ディレクティブを使って ADC ルーチンは含まれなくなります。設計上 ADC が必要なくなった場合、このパラメータを Disabled に設定すると便利です。これにより CSDADC は、2 次変調器を使って、標準の CY8C24x94 CSD UM より S/N 比を向上させることができます。

PGAGain

ADC への PGA ゲインを設定します。PSoC Designer または API で提供される CSDADC_SetGain ルーチンを使用したとき、ゲインの範囲は 1 ～ 48.00 です。1 未満のゲイン設定はサポートされていません。デフォルト値は 1.00 です。

PGAReference

PGA のゲインは、選択した 「グランド」値を基準とします。選択肢には、AGND (オンチップ アナログ グランド)、V_{ss}、近接連続時間 (CT) ブロック、スイッチト キャパシタ (SC) ブロックがあります。CT 及び SC ブロックの接続を使うと、制御されたリファレンス電圧として、オフセットを調整できるようになります。CSDADC_SetRef API は、ランタイムで基準を変更するときに使用できます。デフォルトの PGA リファレンスは AGND です。

AnalogBus (アナログ バス)

PGA ブロックの出力は、ローカルな相互接続のアナログ PSoC ブロック アレイのネットワークやアナログ出力バスを通して接続できます。PGA ユーザ モジュールの AnalogBus パラメータをデフォルト値の Disable (無効) に設定すると、ローカル ネットワークへの接続が限定されます。PGA AnalogBus 出力をバスで行う場合は、他のユーザ モジュールが同じバスを駆動させないように注意を払う必要があります。このパラメータは、ユーザ モジュールの CSD 及び ADC 部分の両方で使用されます。

指の閾値

この閾値は、各ボタンセンサの状態を判断するために使用します。 1 つでもセンサがアクティブな場合、bIsAnySensorActive() 関数は 1 を返します。すべてのセンサがオフの場合、bIsAnySensorActive() 関数は 0 を返します。

指検知閾値は、すべてのセンサとスライダに適用されます。個々のセンサ (スライダグループに属さないセンサ) では、これらの閾値は変数で、baBtnFThreshold[] アレイで提供されます。SetDefaultFingerThresholds() 関数を使用すると、閾値をデバイス エディタで設定されているデフォルト値に設定できます。個々のセンサの感度を調整するには、各センサの baBtnFThreshold[] 値を変更します。 (このバイトアレイのサイズは、個々の実施センサ数と同等です。)

可能な値の範囲は 5 ～ 255 です。デフォルト値は 40 です。

ノイズ閾値

個々のセンサでは、このパラメータは閾値となるカウント値を設定し、これを上回ると基準値が更新されなくなります。スライダセンサでは、この閾値を下回るカウント値は重心の計算に加えられません。可能な値は 5 から 255 です。デフォルト値は 20 です。

BaselineUpdate 閾値

新しい Raw カウント値が現在のベースライン値を上回っており、その差がノイズ閾値を下回る場合 (センサのオートリセットパラメータは無効にセットされている)、現在のベースライン値と Raw カウントの差はバケツに集められたような状態になります。バケツが満杯になると、ベースライン値はある値分増分し、バケツは空になります。このパラメータは、ベースライン値が増分するためにバケツが到達しなければならない閾値を設定します。可能な値は 0 から 255 です。パラメータ値が大きいときは、ベースライン値の更新速度を遅くします。より頻繁なベースライン値の更新が必要なときは、このパラメータを小さくします。デフォルト値は 200 です。

LowBaselineReset

LowBaselineReset パラメータは、NegativeNoiseThreshold パラメータとともに作動します。指定されたサンプル数で、サンプルカウント値が (Baseline 値 - NegativeNoiseTreshold) 以下の場合、Baseline 値は新しい Raw カウント値に設定されます。これは基本的に、異常に低い Raw カウントが認識されたときに、ベースラインをリセットする必要があるときに使われるもので、通常、指などが置かれたまま起動された時等の異常な状態をリセットする為に使用されます。。可能な値は 0 から 255 です。デフォルト値は 50 です。

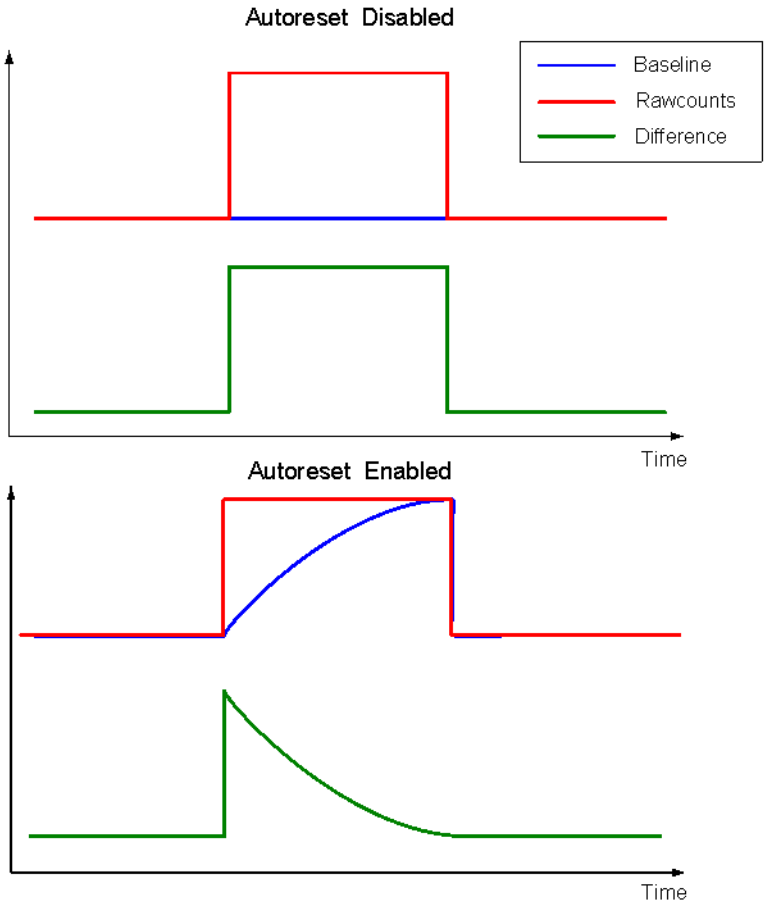
センサの Autoreset

このパラメータは、Baseline 値が常時更新されるか、信号差がノイズ閾値より低い場合のみ更新されるかを指定します。[**Enabled**] (有効) にセットされている場合、Baseline 値は常時更新されます。この設定は、センサの最大時間を制限します (標準的な値は 5 ~ 10 秒) が、何もセンサに触れずに生カウントが突然上がった際に、センサが永続的にオンになるのを妨ぐことができます。この突然の上昇の原因には、大幅な電源電圧の変化、高エネルギー RF ノイズ源、非常に速い温度変化があります。デフォルトで Enabled (有効) になっています。

パラメータが [**Disabled**] (無効) にセットされている場合、Raw 値と Baseline 値の差がノイズ閾値パラメータを下回る場合にのみ、基準値は更新されます。何もセンサに触れずに Raw カウントが突然上がった際に、センサが永続的にオンになるという問題がない限り、このパラメータは [Disabled] にしておきます。

以下の図は、このパラメータが Baseline 値更新に与える影響について示しています。

Figure 9.　センサ自動リセットパラメータ



ヒステリシス

ヒステリシスパラメータは、センサが現在動作中であるか否かに応じて、指閾値に数値を足したり、そこから引いたりします。センサが動作中でない場合、差の数は指閾値 + ヒステリシスを上回る必要があります。センサが動作中の場合、差の数は指閾値 - ヒステリシスを下回る必要があります。これは、デバウンス性と粘着性を指検知アルゴリズムに加えるために使用されます。ヒステリシスを伴う閾値は、bIsSensorActive() または bIsAnySensorActive() が呼び出されたときに評価されます。センサの状態は、戻り値 bIsSensorActive() または baSnsOnMask[] アレイを使用してモニターされます。可能な値は 0 から 255 ですが、指閾値パラメータ設定よりも低くなければなりません。デフォルト値は 10 です。

適切な高レベル判定論理パラメータを選択すると、環境要因 (温度や湿度の変化など) を効果的に補償し、ノイズ信号 (ESD、電源スパイク) を抑圧し、様々な条件下で信頼できるタッチ検知を実施できます。

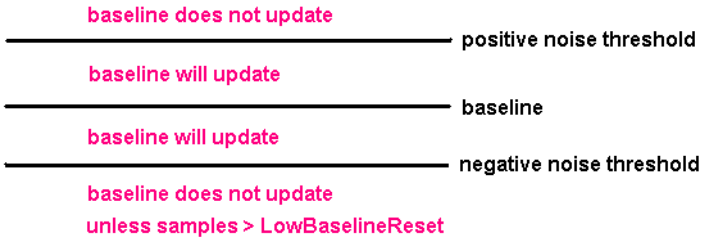
Debounce (デバウンス)

デバウンス パラメータは、センサの動作遷移のためのデバンスカウンタを設定します。センサが非作動中から動作中へ遷移するためには、指定されたサンプル数に対して、Diff カウント値が指閾値 + ヒステリシスを上回る状態を続けなければなりません。デバウンス数は、API 関数の bIsSensorActive または bIsAnySensorActive で増分されます。

可能な値は 1 ー 255 です。1 をセットするとデバウンスは起こりません。デフォルト値は 3 です。

NegativeNoiseThreshold

NegativeNoiseThreshold パラメータは、マイナスの差カウント閾値を追加します。現在の Raw カウントがベースラインを下回り、これらの差がこの閾値を上回る場合、ベースラインは更新されません。しかし、LowBaselineReset パラメータで設定されたサンプル数の間、現在の Raw カウントが低い状態で続く場合 (差は閾値より大きい)、ベースラインはリセットされます。可能な値は 0 から 255 です。デフォルト値は 20 です。



スキャン速度

このパラメータは、センサのスキャン速度に影響を及ぼします。利用可能な選択肢は、**Fast** (高速)、**Normal** (標準)、**Slow** (低速) です。デフォルト値は Normal (標準) です。スキャン速度が遅いと、次のような利点があります。

- 優れた S/N 比
- 電源と温度変化に対するより良いイミュニティ
- システム割り込みレイテンシの必要性が減少し、より長い割り込みを処理可能

分解能

このパラメータはスキャン分解能をビット数で判断します。PRS16 構成では、センサは 9 ~ 16 bits の分解能でスキャンできます。デフォルトの分解能は 12 ビットです。PRS8、プリスケーラ、VC2 構成では、センサは 10、12、14 ビットでのみスキャンできます。これらの構成でより高い分解能が必要な場合は、オーバーサンプリングを使用し、複数のサンプリングを平均化してください。ビット数が N の場合、スキャン分解能の最大 Raw カウントは 2^N-1 です。ADC 分解能の設定でも同じパラメータ値が使用されます。

分解能を高くすると、検出感度とタッチ検知の S/N 比が高くなります。近接検知用には高分解能を使用します。16-bit 分解能、低速スキャンモード、20 cm のワイヤによって、20 cm 以上離れた手を検知できます。

Table 11. 24 MHz IMO 動作におけるスキャン時間 (単位 : µs)、スキャン Speed と分解能、PRS16 構成

分解能 (単位 : ビット)	スキャン速度		
	高速	通常	低速
9	424	548	990
10	680	890	1670
11	1200	1580	3040
12	2220	2880	5800

分解能 (単位 : ビット)	スキャン速度		
	高速	通常	低速
13	4280	5680	11200
14	8400	11100	22300
15	16600	22100	44200
16	33000	44000	88000

Table 12. 24 MHz IMO 動作における、スキャン時間 (単位 : µs) 対スキャン速度と分解能、PRS8、プリスケーラ、VC2 構成

分解能 (単位 : ビット)	スキャン速度		
	高速	通常	低速
10	124	136	296
12	220	220	548
14	428	552	1060

注 : スキャン時間は、2 つのセンサスキャンの間の間隔を測定したものです。この時間には、センサのセットアップ時間、変調器安定化遅延、サンプル変換間隔、データ前処理時間が含まれています。

変調器キャパシタピン

このパラメータは、外付け変調器キャパシタ (C_{mod}) を接続するピンをセットします。利用できるピンから選択します。デフォルトピンは P0[5] です。

Feedback Resistor Pin (フィードバック レジスタ ピン)

このパラメータは、外付けフィードバック レジスタ (R_b) を接続するピンを設定します。利用できるピンから選択します。フィードバック レジスタ接続では、ISSP プログラミングに問題が発生することがあるため、P1[1] の使用は推奨できません。ヒント : これらのピンの一部が、センサ接続の割り当てなど、その他の目的で使用される場合は、UM パラメータ リストには表示されません。デフォルトピンは P1[1] です。

Prescaler Period (プリスケーラ期間)

このパラメータは、プリスケーラ期間レジスタを設定し、プリチャージ スイッチ出力周波数を決定します。このパラメータは、プリスケーラを持つ PRS8 構成でのみ利用できます。プリスケーラ期間値は、1 ～ 255 になります。デフォルトのプリスケーラ期間は 7 です。

最大信号対雑音比 (S/N 比) を得るための推奨値は 2^n-1 です。

- 1
- 3
- 7
- 15
- 31
- 63
- 127

- 255

その他の値を使うと、特に低い分解能と高いスキャン速度の場合に、ノイズが増えます。

PRS Polynomial (PRS 多項)

このパラメータは、PRS8 構成で、PRS 多項を設定します。次の 2 つのオプションがあります。

- Short (短) - 短い多項設定を使うと、S/N 比が改善されますが、繰り返し期間が短くなるため、終端デバイスが外部ノイズ源の影響を受けやすくなります。
- Long (長) - 長い多項設定を使うと、S/N 比が悪くなりますが、デバイスがノイズ信号に影響されにくくなります。それはデフォルト値です。

ShieldElectrodeOut

シールド電極信号源は、空いているデジタル行バス (Row_0_Output_0 ～ Row_0_Output_3).) のいずれかから選択できます。各行出力は、3 つのピンのいずれかに迂回できます。Row LUT 関数を A に設定します。デフォルト値は None (なし) です。

アプリケーション プログラミング インタフェース

アプリケーション プログラミング インタフェース (API) 関数は、ハイレベルでモジュールを扱うことができるように、ユーザ モジュールの一部として提供されています。このセクションでは、各関数に対するインタフェースを include ファイルによって提供される関連定数とともに示します。

注 ** ここでは、すべてのユーザ モジュール API と同様に、API 関数を呼び出すことで A と X レジスタの値は、変更される可能性があります。関数を呼び出す場合、呼出し後に A と X の値が必要になるならば、必ず呼び出し前に A と X の値を保存してください。PSoC Designer のバージョン 1.0 以降では、効率を高めるために、この「registers are volatile (レジスタの揮発性)」ポリシーが選択され、実施されています。C コンパイラは自動的にこの条件を処理します。アセンブリ言語のプログラマは、コードでこのポリシーを考慮する必要があります。一部のユーザ モジュール API 関数では A と X は変更されないこともありますが、将来も変更されないという保証はありません。

大型メモリモデルデバイスでは、CUR_PP、IDX_PP、MVR_PP、および MVW_PP レジスタにおける値を保存するのも呼び出し側の責任です。これらのレジスタのいくつかを現在修正することはできないとしても、将来のリリースにおいても現在の状態が維持されることを保証することはできません。

エントリ ポイントは、CSDADC の初期化やサンプリングの開始、CSDADC の終了を行うためのものです。どの場合でも、モジュールのインスタンス名が次のエントリ ポイントの CSDADC 接頭辞を置き換えます。間違ったインスタンス名の使用は、構文エラーの一般的な原因です。

API の機能は、様々なグローバルアレイを使用します。これらのアレイを手動で変更しないでください。ただし、デバックの目的でこれらの値を調べることが可能です。たとえば、チャート作成ツールを使用して、アレイの内容を表示することは可能です。次にいくつかのグローバルアレイを挙げます。

- CSDADC_waSnsBaseline[]
- CSDADC_waSnsResult[]
- CSDADC_waSnsDiff[]
- CSDADC_baSnsOnMask[]

CSDADC_waSnsBaseline[] – 各センサのベースラインデータを含む整数アレイです。アレイのサイズはセンサ数と同等です。CSDADC_waSnsBaseline[] アレイは以下の関数によって更新されます。

- CSDADC_UpdateAllBaselines();
- CSDADC_UpdateSensorBaseline();
- CSDADC_InitializeBaselines().

CSDADC_waSnsResult[] – 各センサの Raw データの整数アレイです。アレイのサイズはセンサ数と同等です。CSDADC_waSnsResult[] データは次の関数によって更新されます。

- CSDADC_ScanSensor();
- CSDADC_ScanAllSensors().
- CSDADC_ScanSensorsAveraging()

CSDADC_waSnsDiff [] – 各センサの Raw データとベースラインデータの差を含む整数アレイです。アレイのサイズはセンサ数と同等です。

CSDADC_baSnsOnMask[] – センサのオン・オフの状態（ボタンまたはスライダ）を維持するバイトアレイです。CSDADC_baSnsOnMask[0] は、センサ 0 ~ 7 のマスクされたビットを含んでいます（センサ 0 はビット 0、センサ 1 はビット 1)。CSDADC_baSnsOnMask[1] はセンサ 8 ~ 15 のマスクされたビットを含んでいます。必要に応じて、同様の方法で、より多くのセンサにも対応できます。このバイトアレイには、全ての配置されているセンサの要素が含まれます。ボタンがオンの場合 1 つのビットの値は 1 で、オフの場合その値は 0 です。CSDADC_baSnsOnMask[] データは、CSDADC_blsSensorActive(BYTE bSensor) 関数または CSDADC_bIsAnySensorActive() ルーチンによって更新されます。

CSDADC_Start

説明：

レジスタを初期化し、ユーザーのモジュールを起動します。他のユーザ モジュール関数を呼び出す前に、この関数を呼び出す必要があります。

C プロトタイプ

```
void  CSDADC_Start()
```

アセンブラ

```
lcall  CSDADC_Start
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_Stop

説明：

センサスキャンを停止し、内部割り込みを無効にし、CSDADC_ClearSensors() を呼び出してすべてのセンサをリセットしてオフ状態にします。

C プロトタイプ

```
void  CSDADC_Stop()
```

アセンブラ

```
lcall  CSDADC_Stop
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_SetRefMux

説明：

このルーチンは、システムの AGND、RefHi、RefLo リファレンスを設定し、PSoC Designer のデバイス エディタで設定される値を上書きします。

C プロトタイプ

```
void  CSDADC_SetRefMux(BYTE bReference)
```

アセンブラ

```
mov A, [Ref]
lcall  CSDADC_SetRefMux
```

パラメータ：

リファレンスは、以下の値のいずれかを取ります。

記号名	値	説明
REF_HALFVDD_BANDGAP	0x00	(Vdd/2) +/- Bandgap
REF_P24_P26	0x08	P2[4] +/- P2[6]
REF_HALFVDD_HALFVDD	0x10	(Vdd/2) +/- (Vdd/2)
REF_2BANDGAP_BANDGAP	0x18	BandGap +/- BandGap
REF_2BANDGAP_P26	0x20	2Bandgap +/- P2[6]
REF_P24_BANDGAP	0x28	P2[4] +/- BandGap
REF_BANDGAP_BANDGAP	0x30	BandGap +/- BandGap
REF_16BANDGAP_16BANDGAP	0x38	1.6BandGap +/- 1。6BandGap

戻り値：

なし

副作用：

**

CSDADC_SetPGAGain

説明：

PGA ブロックのゲインを設定し、デバイス エディタで設定される値を上書きします。

C プロトタイプ

```
void  CSDADC_SetPGAGain(BYTE bGainSetting)
```

アセンブラ

```
mov   A, bGainSetting
lcall  CSDADC_SetPGAGain
```

パラメータ：

bGainSetting: C とアセンブリの include ファイルで設定される記号名及び関連する値を、次の表に示します。PGA ゲインは、1 ～ 48 に設定できます。1 未満の設定はサポートされていません。この関数は、ADC モードと CSD モードで共通です。前者の場合は ADC プリアンプ ゲインを、後者の場合は変調器のゲインを設定します。

記号名	値
PGA_G48_0	0x0C
PGA_G24_0	0x1C
PGA_G16_0	0x08
PGA_G8_00	0x18
PGA_G5_33	0x28
PGA_G4_00	0x38
PGA_G3_20	0x48
PGA_G2_67	0x58
PGA_G2_27	0x68
PGA_G2_00	0x78
PGA_G1_78	0x88
PGA_G1_60	0x98
PGA_G1_46	0xA8
PGA_G1_33	0xB8
PGA_G1_23	0xC8
PGA_G1_14	0xD8
PGA_G1_06	0xE8
PGA_G1_00	0xF8

戻り値：

なし

副作用：

**

CSDADC_SetPGARef

説明：

PGA ブロックのリファレンスを設定します。PGA のゲインは、ユーザが選択した「グランド」値をリファレンスとします。選択肢には、AGND (オンチップ アナログ グランド)、V_{ss}、近接連続時間 (CT) ブロック、スイッチト キャパシタ (SC) ブロックがあります。CT 及び SC ブロックの接続を使うと、制御されたリファレンス電圧として、オフセットを調整できるようになります。

C プロトタイプ

```
void  CSDADC_SetPGARef(BYTE bRefSetting)
```

アセンブラ

```
mov   A, bRefSetting
lcall  CSDADC_SetPGARef
```

パラメータ：

bRefSetting: C とアセンブリで設定される記号名及び関連する値を、この表に示します。

記号名	値
ADC00	0x00
AGND	0x01
VSS	0x02
ASC10	0x03

戻り値：

なし

副作用：

**

CSDADC_EnableADC

説明：

UM の ADC 関数を動作させます。このルーチンを呼び出した後でのみ、ADC 関連ルーチンの呼び出しが可能になります。

C プロトタイプ

```
void  CSDADC_EnableADC (void);
```

アセンブラ

```
lcall CSDADC_EnableADC
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_EnableCapSense

説明：

このルーチンは CapSense 動作を有効にし、CapSense 関数にハードウェアの構成を設定します。このルーチンを呼び出した後で、CapSense 関連ルーチンの呼び出しが可能になります。

C プロトタイプ

```
void  CSDADC_EnableCapSense(void);
```

アセンブラ

```
lcall CSDADC_EnableCapSense
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_EnableInput

説明：

入力ポートを ADC に接続します。接続はアナログ バスを使って行われるため、利用可能なポートはすべて ADC 入力機能に対応します。

C プロトタイプ：

```
void CSDADC_EnableInput (BYTE bMask, BYTE bPort);
```

アセンブラ

```
mov A, [bMask]
mov X, [bPort]
lcall CSDADC_EnableInput
```

パラメータ：

bPort - ADC の入力ポートを設定

bMask - ポートのビット

副作用：

**

CSDADC_DisableInput

説明：

選択したポート ピンをアナログ バスから切断することで、ADC から入力ポートを切断します。CSDADC_ EnableInput と CSDADC_ DisableInput の両ルーチンは対で呼び出す必要があります。

C プロトタイプ：

```
void CSDADC_DisableInput(BYTE bMask, BYTE bPort);
```

アセンブラ

```
mov A, [bMask]
mov X, [bPort]
lcall CSDADC_DisableInput
```

パラメータ：

bPort - ADC の入力ポートを設定

bMask - ポートのビット

副作用：

**

CSDADC_StartADC

説明：

ADC をオンにし、連続して実行します。ADC が機能するためには、グローバル割り込みを有効にする必要があります。

C プロトタイプ

```
void CSDADC_StartADC(void)
```

アセンブラ

```
lcall CSDADC_StartADC
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_StopADC

説明：

ADC をオフにし、変換処理を直ちに停止します。

C プロトタイプ

```
void CSDADC_StopADC(void)
```

アセンブラ

```
lcall CSDADC_StopADC
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_fIsDataAvailable

説明：

ADC のステータスを確認します。

C プロトタイプ

```
BYTE CSDADC_fIsDataAvailable(void)
```

アセンブラ

```
lcall CSDADC_fIsDataAvailable; Return value will be in A
```

パラメータ：

なし

戻り値：

データが変換され、読み取れる状態の場合は、非ゼロ値を返します。

副作用：

**

CSDADC_wGetData

説明：

変換されたデータを返します。データ サンプルが準備できたかどうかを確認するためには、CSDADC_fIsDataAvailable() を呼び出す必要があります。

C プロトタイプ

```
WORD CSDADC_wGetData(void)
```

アセンブラ

```
lcall CSDADC_iGetData
;Data will be in A (LSB) and X(MSB) upon return
```

パラメータ：

なし

戻り値：

符号なし整数の形式で、変換されたデータ サンプルを返します。

副作用：

**

CSDADC_wGetDataClearFlag

説明：

データを取得して、データ利用可能フラグをリセットします。

C プロトタイプ

```
WORD CSDADC_wGetDataClearFlag(void)
```

アセンブラ

```
lcall CSDADC_wGetDataClearFlag
;Data will be in A (LSB) and X(MSB) upon return
```

パラメータ：

なし

戻り値：

符号なし整数の形式で、ADC 変換サンプルを返します。

副作用

**

CSDADC_ScanSensor

説明：

選択されたセンサの静電容量をスキャンします。各センサは、センサアレイ内で独自の番号を持っています。この番号は CSDADC ウィザードによって順番に割り当てられます。Sw0 は センサ 0、Sw1 はセンサ 1 などのように割り当てられます。

C プロトタイプ

```
void CSDADC_ScanSensor(BYTE bSensor);
```

アセンブラ

```
mov A,  bSensor
lcall CSDADC_ScanSensor
```

パラメータ：

A => Sensor 数字

戻り値：

なし

副作用

**

CSDADC_ScanSensorAveraging

説明：

選択したセンサを 2^{bPower} 回スキャンし、結果を平均化します。出力データは bPower ビットだけ右にシフトされるので、出力データの分解能は維持されます。複数のスキャン結果を平均化することで、より高い信号対雑音比 (S/N 比) が得られます。PRS8、プリスケーラ、VC2 プリチャージクロック構成を使用している場合、この関数は、CSDADC_ScanSensor() 関数を複数回呼び出すよりも高速です。これは、$Sinc^2$ フィルタには、最初の 2 つのサンプルをスキップする CSDADC_ScanSensor() 関数が必要であるためです。

任意の分解能及びスキャン速度で CSDADC_ScanSensor() 関数による単一のセンサをスキャンする回数を T とすると、速度の差分は以下のようになります。

- ScanSensorAveraging – $(T/3)*2^{bPower} + 2T/3$
- 複数回の ScanSensor 呼び出し – $T*2^{bPower}$

この関数は、CY8C21x34/B CapSense Design Guide で説明されているように、十分な調整を行った後でも良好な S/N 比が得られない場合のみ使用してください。たとえば、厚いオーバーレイ越しにスキャンする場合や、近接検知に CapSense を使用する場合、または ITO フィルムを使用する際などに低周波数での動作が必要な場合が挙げられます。

C プロトタイプ

```
void CSDADC_ScanSensorAveraging(BYTE bSensor, BYTE bPower);
```

アセンブラ

```
mov A,  bSensor
mov A,  bPower
lcall CSDADC_ScanSensorAveraging
```

パラメータ：

bSensor：スキャンされるセンサ番号。

bPower：センサがスキャンされる回数を定義します。センサは、2^{bPower} 回スキャンされます。許可される値は、0 ー 8 です。

戻り値：

なし

副作用

**

CSDADC_ScanAllSensors

説明：

各センサインデックスに対して CSDADC_ScanSensor() を呼び出して、構成済み静電容量式センサをすべてスキャンします。

C プロトタイプ

```
void CSDADC_ScanAllSensors();
```

アセンブラ

```
lcall CSDADC_ScanAllSensors
```

パラメータ：

なし

戻り値：

なし

副作用

**

CSDADC_UpdateSensorBaseline

説明：

この経時的カウント値は、センサ別に独立して計算され、センサのベースライン値と呼ばれます。ベースライン値はバケツメソッドを用いて更新されます。

バケツメソッドは、次のアルゴリズムを使用します。

1. CSDADC_UpdateSensorBaseline() が呼び出されるたびに、Raw カウント値を前回のベースライン値から引いて Difference 値を計算します。この差は CSDADC_waSnsDiff[] アレイに保存され、表示されます。

2. センサ Autoreset が無効な場合、CSDADC_UpdateSensorBaseline() が呼び出されるたびに、Difference 値をノイズ閾値と比較します。Diff カウント値がノイズ閾値より小さい場合、仮想バケツに入れられます。差がノイズ閾値より大きい場合、バケツは更新されません。センサ Autoreset が有効な場合、ノイズ閾値パラメータに関わらず、差分は仮想バケツに集積されます。

3. 仮想バケツで集積された差のカウントが BaselineUpdateThreshold に達すると、基準値は 1 増分され、バケツは 0 にリセットされます。

4. Difference カウントがノイズ閾値より小さい場合、waSnsDiff[] アレイに保持されている値が 0 にリセットされます。従って、このアレイには 0 より大きく NoiseThreshold より小さい値を持つ成分は含まれません。

C プロトタイプ

```
void CSDADC_UpdateSensorBaseline(BYTE bSensorNum)
```

アセンブラ

```
mov   A,   bSensorNum
lcall  CSDADC_UpdateSensorBaseline
```

パラメータ：

A => Sensor 数字

戻り値：

なし

副作用：

**

CSDADC_UpdateAllBaselines

説明：

CSDADC_bUpdateSensorBaseline() 関数を使用して、すべてのセンサのベースライン値を更新します。

C プロトタイプ

```
void  CSDADC_UpdateAllBaselines()
```

アセンブラ

```
lcall CSDADC_UpdateAllBaselines
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_bIsSensorActive

説明：

センサの Difference カウントアレイを確認し、指閾値と比較します。ここではヒステリシスを考慮します。ヒステリシス値は、センサが現在オンであるか否かに基づいて、指閾値を足したり引いたりします。アクティブ時の場合、閾値は下げられます。アクティブ時でない場合、閾値は上げられます。この関数は、CSDADC_baSnsOnMask[] アレイでセンサのビットを更新します。

C プロトタイプ

```
BYTE  CSDADC_bIsSensorActive(BYTE bSensorNum)
```

アセンブラ

```
mov  A,  bSensorNum
lcall CSDADC_bIsSensorActive
```

パラメータ：

bSensor A => センサ番号

戻り値：

戻り値は、アクティブの場合 1 で、アクティブでない場合は 0 です。

副作用：

**

CSDADC_bIsAnySensorActive

説明：

全センサの Difference カウントアレイを確認し、指閾値と比較します。各センサについて CSDADC_bIsSensorActive() を呼び出し、CSDADC_baSnsOnMask[] アレイが関数呼び出し後に最新の状態であるようにします。

C プロトタイプ

```
BYTE  CSDADC_bIsAnySensorActive()
```

アセンブラ

```
lcall CSDADC_bIsAnySensorActive
```

パラメータ：

なし

戻り値：

戻り値は、アクティブの場合 1 で、アクティブでない場合は 0 です。

副作用：

**

CSDADC_wGetCentroidPos

説明：

Difference アレイを確認し、重心を探します。1 つ存在する場合、オフセットと長さが一時変数に保存され、重心位置が CSDADC ウィザードで指定された分解能で計算されます。この関数は、スライダが CSDADC ウィザードで定義されている場合のみ利用できます。

C プロトタイプ

```
WORD  CSDADC_wGetCentroidPos(BYTE bSnsGroup)
```

アセンブラ

```
mov  A,  bSnsGroup
lcall CSDADC_wGetCentroidPos
```

パラメータ：

bSnsGroup A => グループ番号

このパラメータは、スライダとして使用されるセンサグループへのレファレンスです。グループ 0 はボタン用です。スライダはグループ 1 以降に含まれています。

戻り値：

スライダの位置値は LSB は A に、MSB は X にあります。

副作用：

このルーチンは、ノイズ閾値を引くことによって Difference カウントを調整します。差が負値となることを避けるために、各スキャン後一度だけ呼び出します。Difference カウント信号をモニターするアプリケーションの場合、Difference カウントデータ転送後にこのルーチンを呼び出します。

スライダセンサが 1 つでも動作中の場合、この関数はゼロから CSDADC ウィザードで設定された分解能の値までを返します。アクティブのセンサがない場合、関数は –1 (FFFFh) を返します。重心 / ダイプレックスアルゴリズムの実行中にエラーが発生した場合、関数は –1 (FFFFh) を返します。必要に応じて、CSDADC_bIsSensorActive() ルーチンを使用してタッチされたスライダセグメントを判定できます。

注 スライダセグメント上のノイズカウントがノイズ閾値より大きい場合、このサブルーチンは正しくない重心結果を示すことがあります。ノイズが擬似重心結果を生じないように、ノイズ閾値は慎重に設定します (ノイズレベルを十分に超える高さ)。

CSDADC_InitializeSensorBaseline

説明：

選択されたセンサをスキャンして、初期値を伴う CSDADC_waSnsBaseline[bSensor] アレイを読み込みます。Raw カウント値は、選択されたセンサの Baseline 値の配列エレメントにコピーされます。この関数は、個々のセンサのベースラインをリセットするために使用されます。

C プロトタイプ

```
void CSDADC_InitializeSensorBaseline(BYTE bSensorNum)
```

アセンブラ

```
mov A, bSensorNum
lcall CSDADC_InitializeSensorBaseline
```

パラメータ：

A => センサ番号

戻り値：

なし

副作用：

**

CSDADC_InitializeBaselines

説明：

それぞれのセンサをスキャンして、CSDADC_waSnsBaseline[] アレイに初期値をロードします。Raw カウント値は各センサのベースラインアレイにコピーされます。

C プロトタイプ

```
void CSDADC_InitializeBaselines()
```

アセンブラ

```
lcall CSDADC_InitializeBaselines
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_SetDefaultFingerThresholds

説明：

FingerThreshold パラメータ値を伴う CSDADC_baBtnFThreshold[] アレイを読み込みます。CSDADC_baBtnFThreshold[] アレイがカスタム値とともに手動で読み込まれていない場合、この関数はスキャン前に呼び出さなければなりません。

C プロトタイプ

```
void CSDADC_SetDefaultFingerThresholds()
```

アセンブラ

```
lcall CSDADC_SetDefaultFingerThresholds
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_SetScanMode

説明：

スキャン速度と分解能を設定します。この関数を実行時に呼び出し、スキャン速度と分解能を変更することができます。関数は、ユーザ モジュールのパラメータ設定を上書きします。異なるスキャン速度と分解能でスキャンする必要のあるセンサがある場合、この関数は効果的です。例としては、通常のボタンと近接検出などが挙げられます。通常のボタンは 9-bit 分解能、300 µs スキャン速度でスキャンできます。近接検知では、長い範囲の検知を実行するために、16-bit の分解能と 12 ms 以上のスキャン時間でスキャンの頻度を下げます。この関数は CSDADC_ScanSensor() 関数とともに使用できます。

C プロトタイプ

```
void CSDADC_SetScanMode(BYTE bSpeed, BYTE bResolution);
```

アセンブラ

```
mov     A, bSpeed
mov     X, bResolution
lcall    CSDADC_SetScanMode
```

パラメータ：

bSpeed：スキャン速度

次の定数は bSpeed パラメータ用です。

定数	値
CSDADC_FAST_SPEED	0x01
CSDADC_NORMAL_SPEED	0x02
CSDADC_SLOW_SPEED	0x03

bResolution：スキャン分解能。値を必要な分解能ビット数に設定します。値の範囲はユーザのモジュール設定に基づきます。

以下の定数は PRS16 構成の bResolution パラメータ用です。

定数	値
CSDADC_9_BIT_RESOLUTION	9
CSDADC_10_BIT_RESOLUTION	10
CSDADC_11_BIT_RESOLUTION	11
CSDADC_12_BIT_RESOLUTION	12

定数	値
CSDADC_13_BIT_RESOLUTION	13
CSDADC_14_BIT_RESOLUTION	14
CSDADC_15_BIT_RESOLUTION	15
CSDADC_16_BIT_RESOLUTION	16

以下の定数は PRS8、プリスケーラ、VC2 構成の bResolution パラメータ用です。

定数	値
CSDADC_10_BIT_RESOLUTION	10
CSDADC_12_BIT_RESOLUTION	12
CSDADC_14_BIT_RESOLUTION	14

戻り値：

なし

副作用：

**

CSDADC_ClearSensors

説明：

各センサに対して CSDADC_wGetPortPin() と CSDADC_DisableSensor() を続けて呼び出し、静電容量式センサをノンサンプリング状態にクリアします。

C プロトタイプ

```
void  CSDADC_ClearSensors()
```

アセンブラ

```
lcall  CSDADC_ClearSensors
```

パラメータ：

なし

戻り値：

なし

副作用：

**

CSDADC_wReadSensor

説明：

A (LSB) と X (MSB) で主要な Raw スキャン値を返します。

C プロトタイプ

```
WORD  CSDADC_wReadSensor(BYTE bSensor)
```

アセンブラ

```
mov A, bSensor
lcall CSDADC_wReadSensor
```

パラメータ：

A => Sensor 数字

戻り値：

センサのスキャンです。A では LSB、X では MSB。

副作用：

**

CSDADC_wGetPortPin

説明：

特定のセンサのポート番号とピンマスクを返します。渡されたパラメータは、CSDADC_Sensor_Table[]. からのデータをインデックスし、選択します。戻り値は、CSDADC_EnableSensor()、CSDADC_DisableSensor(). へと渡すことができます。

C プロトタイプ

```
WORD  CSDADC_wGetPortPin(BYTE bSensorNum)
```

アセンブラ

```
mov  A,  bSensorNumber
lcall     CSDADC_wGetPortPin
;Data will be in A (LSB, Sensor Bitmap) and X(MSB, Port Number) upon return
```

パラメータ：

bSensorNumber – 範囲は 0 ~ (n – 1) です。ここで n は、CSDADC ウィザードで設定されたセンサ数とスライダに含まれているセンサ数の合計です。センサ番号は、選択されたアクティブなセンサのポートとビットマスクを判定するために、CSDADC_wGetPortPin() によって使用されます。

戻り値：

A => Sensor ビットマップ

X => Port 数字

副作用：

**

CSDADC_EnableSensor

説明：

次の測定サイクルで測定するために選択されたセンサを構成します。ポートとセンサは、CSDADC_wGetPortPin() 関数を使用して選択できます。このとき、ポート番号とセンサビットマスクはそれぞれ X と A に読み込まれています。選択されたポートとピンを Analog High Z（アナログハイ Z）モードにし、正しい Analog Mux Bus（アナログ多重化バス）入力を可能にするために、駆動モードを調整します。これによりコンパレータ機能も有効になります。

C プロトタイプ

```
void  CSDADC_EnableSensor(BYTE bMask, BYTE bPort)
```

アセンブラ

```
mov  X,  bPort
mov  A,  bMask
lcall  CSDADC_EnableSensor
```

パラメータ：

A => Sensor ビットマップ

X => Port 数字

戻り値：

なし

副作用：

**

CSDADC_DisableSensor

説明：

CSDADC_wGetPortPin() 関数により選択されたセンサを無効にします。駆動モードは、Strong (001) に変更されます。これにより、センサが効果的にグランド接続されます。ポートピンから AnalogMuxBus までの接続はオフになります。この関数パラメータは、CSDADC_wGetPortPin() 関数により返されます。

C プロトタイプ

```
void  CSDADC_DisableSensor(BYTE bMask, BYTE bPort)
```

アセンブラ

```
mov  X,  bPort
mov  A,  bMask
lcall  CSDADC_DisableSensor
```

パラメータ：

A => Sensor ビットマップ

X => Port 数字

戻り値：

なし

副作用：

**

ファームウェア ソースコードの例

このコードは、ユーザ モジュールを開始し、センサを連続的にスキャンします。

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

WORD wResult;

void main(void) {
   M8C_EnableGInt;

   CSDADC_SetRefMux(CSDADC_REF_BANDGAP_BANDGAP); //BandGap +/- BandGap
   CSDADC_Start();
   CSDADC_SetDefaultFingerThresholds();
   CSDADC_InitializeBaselines();

   while (1) {
     CSDADC_SetRefMux(CSDADC_REF_BANDGAP_BANDGAP); //BandGap +/- BandGap
     CSDADC_EnableCapSense();
     CSDADC_ScanAllSensors();
     CSDADC_UpdateAllBaselines();

     //Here you can insert a code for processing of CapSence data

     CSDADC_SetRefMux(CSDADC_REF_HALFVDD_HALFVDD);//(Vdd/2) +/-(Vdd/2)
     CSDADC_SetPGAGain(CSDADC_G1_00);
     CSDADC_EnableADC();
     CSDADC_EnableInput(0x01,0x02); // use P2[0]
     CSDADC_StartADC();

     // Skip three first samples
     while (0 == CSDADC_fIsDataAvailable());
     wResult = CSDADC_wGetDataClearFlag();
     while (0 == CSDADC_fIsDataAvailable());
     wResult = CSDADC_wGetDataClearFlag();
     while (0 == CSDADC_fIsDataAvailable());
     wResult = CSDADC_wGetDataClearFlag();

     //Here you can insert a code for processing of ADC data

     CSDADC_StopADC();
     CSDADC_DisableInput(0x01, 0x02); // ruquired for normal CSD operation
   }
}
```

アセンブリ言語での同じプロジェクトは次のようになります。

```
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules


export _main

area bss(RAM)          ;inform assembler that variables follow
bCounter:  blk 1

area text(ROM,REL)     ;inform assembler that program code follows

_main:

    M8C_EnableGInt

mov    A, CSDADC_REF_BANDGAP_BANDGAP
    lcall  CSDADC_SetRefMux       ; BandGap +/- BandGap

lcall  CSDADC_Start
    lcall  CSDADC_SetDefaultFingerThresholds
    lcall  CSDADC_InitializeBaselines

loop:
    mov    A, CSDADC_REF_BANDGAP_BANDGAP
    lcall  CSDADC_SetRefMux       ; BandGap +/- BandGap

    lcall  CSDADC_EnableCapSense
    lcall  CSDADC_ScanAllSensors
 lcall  CSDADC_UpdateAllBaselines

    ;Here you can insert a code for processing of CapSence data

    mov    A, CSDADC_REF_HALFVDD_HALFVDD
    lcall  CSDADC_SetRefMux        ; (Vdd/2) +/-(Vdd/2)

    mov    A, CSDADC_G1_00
    lcall  CSDADC_SetPGAGain

    lcall  CSDADC_EnableADC

mov    A, 0x01
    mov    X, 0x02
    lcall  CSDADC_EnableInput       ; use P2[0]

    lcall  CSDADC_StartADC

    ; Skip three first samples

mov   [bCounter], 3
.scan:
lcall  CSDADC_fIsDataAvailable
    cmp    A, 0
    jz     .scan
```

```
    lcall  CSDADC_wGetDataClearFlag
    dec    [bCounter]
    jnz    .scan

; The ADC rezult is stored in A (LSB) and X(MSB)
    ; Here you can insert a code for processing of ADC data

lcall  CSDADC_StopADC

    mov    A, 0x01
    mov    X, 0x02
    lcall  CSDADC_DisableInput

    jmp    loop
```

コンフィグレーション レジスタ

Table 13. ブロック PGA、レジスタ：ACB_CONTROL0_REG (ACB01CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	ゲイン					1	リファレンス	

ゲインは、ユーザーモジュールパラメータ内の Gain 名と ADC に関連した API で制御されます。リファレンスは、PGA Ref ユーザ モジュール パラメータにより制御されます。

Table 14. ブロック PGA、レジスタ：ACB_CONTROL1 (ACB01CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	AnalogBus (アナログバス)	0	1	0	0	0	0	1

AnalogBus は、同じ名前のユーザ モジュール パラメータで制御されます。電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 15. ブロック PGA、レジスタ：ACB_CONTROL2 (ACB01CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 16. ブロック PGA、レジスタ：ACB_CONTROL3 (ACB01CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	EXGain

ExGain ビットは、24 または 48 の PGA ゲインが選択された場合に設定されます。

Table 17. ブロック ADC1、レジスタ：AtoD1cr0 (ASD11CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	0	1	0	0	0

Table 18. ブロック ADC1、レジスタ：AtoD1cr1 (ASD11CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	0

Table 19. ブロック ADC1、レジスタ：AtoD1cr2 (ASD11CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 20. ブロック ADC1、レジスタ：AtoD1cr3 (ASD11CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	1	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 21. ブロック ADC2、レジスタ：AtoD2cr0 (ASC21CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	1	0	0	0	0

Table 22. ブロック ADC2、レジスタ：AtoD2cr1 (ASC21CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	00	0	0	0	0	0	0	0

Table 23. ブロック ADC2、レジスタ：AtoD2cr2 (ASC21CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 24. ブロック ADC3、レジスタ：AtoD2cr3 (ASC21CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 25. ブロック PRS、レジスタ：関数 (DxBxxFN)、バンク 1

ビット	7	6	5	4	3	2	1	0
LSB	0	0	0	0	1	0	1	0
MSB	0	1	1	0	1	0	1	0

Table 26. ブロック PRS、レジスタ：入力 (DxBxxIN)、バンク 1

ビット	7	6	5	4	3	2	1	0
LSB	0	0	0	0	0	0	0	0
MSB	0	0	1	1	0	0	0	0

Table 27. ブロック PRS、レジスタ：出力 (DxBxxOU)、バンク 1

ビット	7	6	5	4	3	2	1	0
LSB	1	1	0	0	0	0	0	0
MSB	1	1	1	0	0	ShieldElectrodeOut		

ShieldElectrodeOut は、シールド電極からの信号を Row Output に出力します。これは、同じ名前のユーザ モジュール パラメータで制御されます。

Table 28. ブロック PRS、レジスタ：制御 (DxBxxCR0)、バンク 1

ビット	7	6	5	4	3	2	1	0
LSB	0	0	0	0	0	0	0	Enable
MSB	0	0	0	0	0	0	0	0

Enable ビットは、PRS ブロックを有効にし、CSDADC API で制御されます。

Table 29. ブロック PRS、レジスタ：シフト (DxBxxDR0)、バンク 0

ビット	7	6	5	4	3	2	1	0
LSB	PRS シフト レジスタ (LSB) - 直接アクセスなし							
MSB	PRS シフト レジスタ (MSB) - 直接アクセスなし							

Table 30. ブロック PRS、レジスタ：多項 (DxBxxDR1)、バンク 0

ビット	7	6	5	4	3	2	1	0
LSB	PRS 多項 (LSB)							
MSB	PRS 多項 (MSB)							

PRS 多項は、ScanSpeed パラメータ及び Resolution パラメータに応じて、CSDADC API によって制御されます。

Table 31. ブロック PRS、レジスタ：シード (DxBxxDR2)、バンク 0

ビット	7	6	5	4	3	2	1	0
LSB	PRS シード / 比較レジスタ (LSB)							
MSB	PRS シード / 比較レジスタ (MSB)							

このレジスタの値は、ほぼすべてのパラメータ値に応じて、API によって制御されます。

PRS8 クロック源構成レジスタを持つ CSDADC

Table 32. ブロック PGA、レジスタ：ACB_CONTROL0_REG (ACB01CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	ゲイン					1	リファレンス	

ゲインは、ユーザーモジュールパラメータ内の Gain 名と ADC に関連した API で制御されます。リファレンスは、PGA Ref ユーザ モジュール パラメータにより制御されます。

Table 33. ブロック PGA、レジスタ：ACB_CONTROL1 (ACB01CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	AnalogBus (アナログ バス)	0	1	0	0	0	0	1

AnalogBus は、同じ名前のユーザ モジュール パラメータで制御されます。電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 34. ブロック PGA、レジスタ：ACB_CONTROL2 (ACB01CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 35. ブロック PGA、レジスタ：ACB_CONTROL3 (ACB01CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	EXGain

ExGain ビットは、24 または 48 の PGA ゲインが選択された場合に設定されます。

Table 36. ブロック ADC1、レジスタ：AtoD1cr0 (ASD11CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	0	1	0	0	0

Table 37. ブロック ADC1、レジスタ：AtoD1cr1 (ASD11CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	0

Table 38. ブロック ADC1、レジスタ：AtoD1cr2 (ASD11CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンパレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 39. ブロック ADC1、レジスタ：AtoD1cr3 (ASD11CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	1	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 40. ブロック ADC2、レジスタ：AtoD2cr0 (ASC21CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	1	0	0	0	0

Table 41. ブロック ADC2、レジスタ：AtoD2cr1 (ASC21CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	00	0	0	0	0	0	0	0

Table 42. ブロック ADC2、レジスタ：AtoD2cr2 (ASC21CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 43. ブロック ADC3、レジスタ：AtoD2cr3 (ASC21CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 44. ブロック PRS、レジスタ：関数 (DxBxxFN)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	0	1	1	0	1	0	1	0

Table 45. ブロック PRS、レジスタ：入力 (DxBxxIN)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	0

Table 46. ブロック PRS、レジスタ：出力 (DxBxxOU)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	0	ShieldElectrodeOut		

ShieldElectrodeOut は、シールド電極からの信号を Row Output に出力します。これは、同じ名前のユーザ モジュール パラメータで制御されます。

Table 47. ブロック PRS、レジスタ：制御 (DxBxxCR0)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	Enable

Enable ビットは、PRS ブロックを有効にし、CSDADC API で制御されます。

Table 48. ブロック PRS、レジスタ：シフト (DxBxxDR0)、バンク 0

ビット	7	6	5	4	3	2	1	0
値	PRS シフト レジスタ - 直接アクセスなし							

Table 49. ブロック PRS、レジスタ：多項 (DxBxxDR1)、バンク 0

ビット	7	6	5	4	3	2	1	0
値	PRS Polynomial (PRS 多項)							

PRS 多項は、ScanSpeed パラメータ及び Resolution パラメータに応じて、CSDADC API によって制御されます。

Table 50. ブロック PRS、レジスタ：シード (DxBxxDR2)、バンク 0

ビット	7	6	5	4	3	2	1	0
値	PRS シード / 比較レジスタ							

このレジスタの値は、ほぼすべてのパラメータ値に応じて、API によって制御されます。

PWM8 クロック源構成レジスタを持つ CSDADC

Table 51. ブロック PGA、レジスタ：ACB_CONTROL0_REG (ACB01CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	ゲイン					1	リファレンス	

ゲインは、ユーザーモジュールパラメータ内の Gain 名と ADC に関連した API で制御されます。リファレンスは、PGA Ref ユーザ モジュール パラメータにより制御されます。

Table 52. ブロック PGA、レジスタ：ACB_CONTROL1 (ACB01CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	AnalogBus (アナログバス)	0	1	0	0	0	0	1

AnalogBus は、同じ名前のユーザ モジュール パラメータで制御されます。電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 53. ブロック PGA、レジスタ：ACB_CONTROL2 (ACB01CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 54. ブロック PGA、レジスタ：ACB_CONTROL3 (ACB01CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	EXGain

ExGain ビットは、24 または 48 の PGA ゲインが選択された場合に設定されます。

Table 55. ブロック ADC1、レジスタ：AtoD1cr0 (ASD11CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	0	1	0	0	0

Table 56. ブロック ADC1、レジスタ：AtoD1cr1 (ASD11CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	0

Table 57. ブロック ADC1、レジスタ：AtoD1cr2 (ASD11CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 58. ブロック ADC1、レジスタ：AtoD1cr3 (ASD11CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	1	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 59. ブロック ADC2、レジスタ：AtoD2cr0 (ASC21CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	1	0	0	0	0

Table 60. ブロック ADC2、レジスタ：AtoD2cr1 (ASC21CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	00	0	0	0	0	0	0	0

Table 61. ブロック ADC2、レジスタ：AtoD2cr2 (ASC21CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 62. ブロック ADC3、レジスタ：AtoD2cr3 (ASC21CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 63. ブロック PWM、レジスタ：関数 (DxBxxFN)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	0	1	1	0	0	0	0	1

Table 64. ブロック PWM、レジスタ：入力 (DxBxxIN)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	0	0	0	1	0	0	0	0

Table 65. ブロック PWM、レジスタ：出力 (DxBxxOU)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	1	1	0	0	0	1	0	0

Table 66. ブロック PWM、レジスタ：制御 (DxBxxCR0)、バンク 1

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	Enable

Enable ビットは、PRS ブロックを有効にし、CSDADC API で制御されます。

Table 67. ブロック PWM、レジスタ：カウント (DxBxxDR0)、バンク 0

ビット	7	6	5	4	3	2	1	0
値	カウント レジスタ - 直接アクセスなし							

Table 68. ブロック PWM、レジスタ：期間 (DxBxxDR1)、バンク 0

ビット	7	6	5	4	3	2	1	0
値	PrescalerPeriod (プリスケーラ期間)							

PrescalerPeriod は、同じ名前のユーザ モジュール パラメータで制御されます。

Table 69. ブロック PWM、レジスタ：比較 (DxBxxDR2)、バンク 0

ビット	7	6	5	4	3	2	1	0
値	比較							

このレジスタの値は、リファレンス ユーザ モジュールのパラメータ値に応じて、API によって制御されます。

VC2 クロック源構成レジスタを持つ CSDADC

Table 70. ブロック PGA、レジスタ：ACB_CONTROL0_REG (ACB01CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	ゲイン					1	リファレンス	

ゲインは、ユーザーモジュールパラメータ内の Gain 名と ADC に関連した API で制御されます。リファレンスは、PGA Ref ユーザ モジュール パラメータにより制御されます。

Table 71. ブロック PGA、レジスタ：ACB_CONTROL1 (ACB01CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	AnalogBus (アナログ バス)	0	1	0	0	0	0	1

AnalogBus は、同じ名前のユーザ モジュール パラメータで制御されます。電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 72. ブロック PGA、レジスタ：ACB_CONTROL2 (ACB01CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 73. ブロック PGA、レジスタ：ACB_CONTROL3 (ACB01CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	EXGain

ExGain ビットは、24 または 48 の PGA ゲインが選択された場合に設定されます。

Table 74. ブロック ADC1、レジスタ：AtoD1cr0 (ASD11CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	0	1	0	0	0

Table 75. ブロック ADC1、レジスタ：AtoD1cr1 (ASD11CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	0

Table 76. ブロック ADC1、レジスタ：AtoD1cr2 (ASD11CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 77. ブロック ADC1、レジスタ：AtoD1cr3 (ASD11CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	1	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

Table 78. ブロック ADC2、レジスタ：AtoD2cr0 (ASC21CR0)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	0	0	1	0	0	0	0

Table 79. ブロック ADC2、レジスタ：AtoD2cr1 (ASC21CR1)、バンク X

ビット	7	6	5	4	3	2	1	0
値	00	0	0	0	0	0	0	0

Table 80. ブロック ADC2、レジスタ：AtoD2cr2 (ASC21CR2)、バンク X

ビット	7	6	5	4	3	2	1	0
値	0	CompBus (コンバレータ バス)	0	0	0	0	0	0

CompBus ビットは、CSDADC API で制御されます。

Table 81. ブロック ADC3、レジスタ：AtoD2cr3 (ASC21CR3)、バンク X

ビット	7	6	5	4	3	2	1	0
値	1	1	1	0	0	0	Power (出力)	

電源は、CSDADC_Start() と CSDADC_Stop() API で制御されます。

バージョン履歴

バージョン	編集者	説明
1.1	DHA	エラーメッセージを更新しました。
1.20	DHA	1. DisableInt マクロの呼び出しをポーリング ループから ISR (PRS16 CSD モード) に移動しました。 2. AnalogComparatorColumn 及び GlobalOutput 行間の接続は、インターコネクト ビューに表示されます。
1.20.b	DHA	ウィザードにヘルプファイルを追加。
1.30	DHA	1. 「AREA UserModules」から 「AREA lit」に DiplexTable を移動。 2. デフォルトの 「DiplexTable」パラメータ値を 0x0112 に設定。 3. コードの圧縮を改善するために 「DiplexUsed」を追加。
1.40	DHA	1. Imagecraft の最適化をサポートするためにエリア宣言を更新。 2. ユーザモジュールデータシートの分解能パラメータ用に記号名を追加。 3. ダイ温度測定機能記述を追加し、SetScanMode() API 用の機能記述を更新。 4. CSDADC_StartTempMeasurement と CSDADC_GetTemperature 機能を実装するために CSDADC ユーザモジュールを更新。 5. ユーザモジュールウィザードにおけるスライダとラジアルスライダ用の分解能範囲演算を更新。 6. ユーザモジュールウィザードヘルプを更新。スライダ分解能パラメータの最小 / 最大値の記述を追加。

Note PSoC Designer 5.1、全ユーザモジュールデータシートに改訂履歴を追加。このセクションでは、ユーザー モジュールの過去のバージョンと現在のバージョンとの違いについて簡単な解説を掲載しています。

Copyright © 2005-2012 © Cypress Semiconductor Corporation. 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporation (サイプレス セミコンダクタ社) は、サイプレス 製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許またはその他の権限下で、ライセンスを譲渡または暗示することもありません。サイプレス 製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、または安全の用途のために使用することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される、生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス 製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC® は Cypress Semiconductor Corp の登録商標であり、PSoC Creator ™ および Programmable System-on-Chip ™ は Cypress Semiconductor Corp. の商標です。本文書で言及するその他すべての商標または登録商標は、各社の所有物です。

全てのソース コード (ソフトウェアおよび / またはファームウェア) はサイプレス セミコンダクタ社 (以下「サイプレス」) が所有し、全世界の特許権保護 (米国およびその他の国)、米国の著作権法ならびに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によりライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタム ソフトウェアおよび / またはカスタムファームウェアを作成する目的に限って、サイプレスのソース コードの派生著作物をコピー、使用、変更そして作成するためのライセンス、ならびにサイプレスのソース コードおよび派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソース コードを複製、変更、変換、コンパイル、または表示することは全て禁止されます。

免責条項 : サイプレス は、明示的または黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性または特定目的への適合性の黙示的な保証か゛含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品または回路を適用または使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネントとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレス ソフトウェア ライセンス契約によって制限され、かつ制約される場合があります。