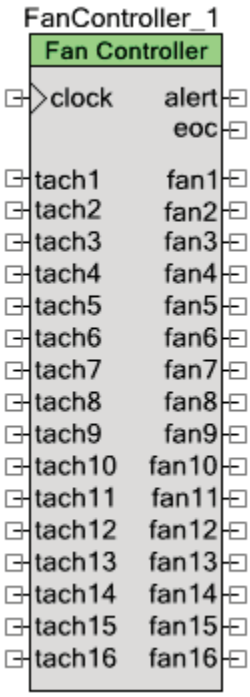
Fan Controller
2.10

特徴



- 最大 16 の PWM 制御、4 線式ブラシレス DC ファンに対応
- 回転速度計入力に対する、個別またはバンク PWM 出力
- 25kHz、50kHz、及びユーザー指定 PWM 周波数に対応
- 最大 25,000RPM のファン速度をサポート
- 4 極および 6 極モーターに対応
- すべてのファンにおいてファンストール / ローターロックを検出
- ファームウェア制御またはハードウェア制御によるファン速度制御が可能
- ファン障害報告用のカスタマイズ可能なアラートピン

概要説明

Fan Controller コンポーネントは、PSoC を使うことにより、設計者が早く、容易にファンコントローラソリューションを開発することを可能にします。コンポーネントは、PWM、回転速度計入力キャプチャタイマー、コントロールレジスタ、ステータスレジスタ、および DMA コントローラを含む、すべての必要なハードウェアブロックをカプセル化した、システムレベルのソリューションです。これにより開発期間と労力を短縮できます。

コンポーネントは GUI によってカスタマイズ可能で、デザイナーは、RPM-デューティサイクルマッピングや物理的なファンバンク構成などのファンの電気機械パラメータを入力することができます。開または閉ループ制御方式や PWM 周波数および分解能を含む性能パラメータは、同じユーザーインターフェースから設定することができます。システムパラメータが入力されると、コンポーネントは PSoC 内のリソースを節減した最適な実装を行い、他の温度管理とシステム管理機能を統合することができます。使いやすい API は、ファームウェア開発者が早く設計および実行できるように提供されています。

注： Fan コントローラコンポーネントを使用する設計では、PSoC の省電力スリープやハイバネート（休止）モードを使用しないでください。これらのモードに入ると、ファンコントローラコンポーネントは、ファンの制御や監視ができなくなります。

Fan Controller の用途

Fan Controller コンポーネントは、4 線式、PWM ベース DC 冷却ファンをドライブおよび監視する必要があるどんな温度管理アプリケーションにも使用できます。アプリケーションが 16 個以上のファンを必要とする場合、Fan Controller コンポーネントを複数回インスタンス生成することができます。同じように、アプリケーション内のファンがバンクに構成されている場合、設計者はバンク当たり 1 つの Fan Controller コンポーネントのインスタンスか、すべてのバンクを取り扱う 1 つのコンポーネントのインスタンスを選択することができます。

入出力接続

ここでは、Fan Controller のさまざまな入出力接続について説明します。I/O 項目のアスタリスク(*)は、その I/O が、説明に挙げられた条件において、回路シンボルに表示されない場合があることを示します。

clock – 入力 *

ファンコントロール PWM 用のユーザー定義クロックソース入力。外部クロックオプションがコンポーネントカスタマイザで選択されている場合にのみ機能します。

tach1～16 – 入力 *

各ファンからの回転速度信号。Fan Controller がファンの回転速度を測定することを可能にします。コンポーネントは、回転速度計出力での 1 回転当たりの 2 つのハイ-ローパルス列を生成する 4 極 DC ファンか、3 つのハイローパル列を生成する 6 極 DC ファンに対応するように設計されています。tach2～16 入力はオプションです。

fan1～16 – 出力 *

ファン速度を制御するための可変デューティサイクルの PWM 出力。ファンバンクが有効な場合は、これら出力ピンは bank1～8 の出力に置き換えられます。fan2～16 はオプションです。

ハードウェア UDB モードの場合は、ファン出力(および関連の tach 入力)の最大数は 12 に制限されており、デジタルリソースの利用を最小にします。

bank1～8 – 出力 *

ファンバンクの速度を制御するための可変デューティサイクルの PWM 出力。これら出力は、バンクが有効な場合にのみ出力されます。

alert – 出力 *

ファン障害が検出された場合にアサートされるアクティブハイ出力ピン(有効にされている場合)。

eoc – 出力 *

End-of-Cycle(サイクル終了)出力は、回転速度計ブロックがシステム内のすべてのファンを測定する度にハイパルス化されます。このピンをステータスレジスタコンポーネントまたは割り込みコンポーネントに接続することにより、ファームウェアのアルゴリズムを Fan Controller ハードウェアに同期するのに使用することができます。

コンポーネントのパラメータ

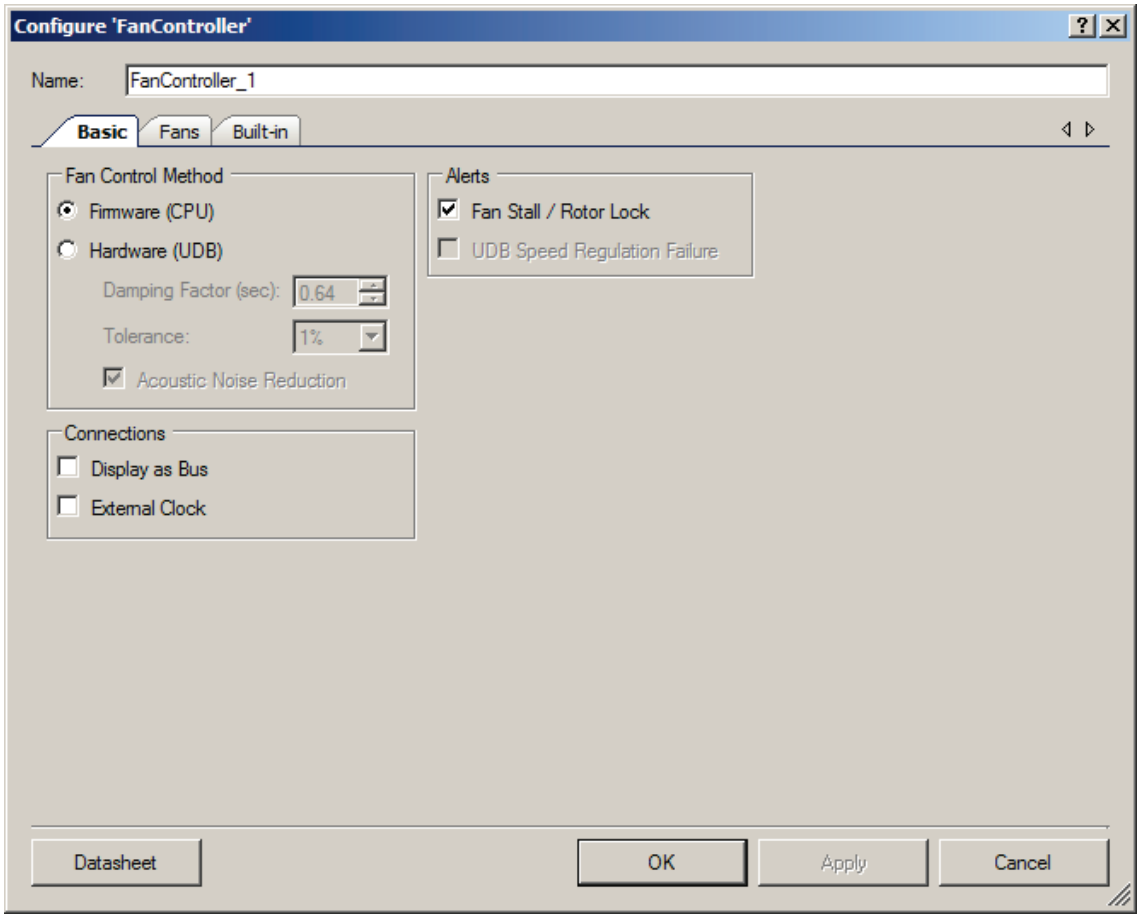
Fan Controller を設計上にドラッグし、ダブルクリックして Configure ダイアログを開きます。このタブは、コンポーネントの基本操作パラメータを設定するのに使用されます。

図 1 は Configure ダイアログです。

Basic Tab(基本タブ)オプション

このタブは、コンポーネントの基本操作パラメータを設定するのに使用されます。

図 1. Basic Configuration(基本設定)ダイアログ



Fan Control Method(ファン制御方法) – Firmware/Hardware

このパラメータはファン速度の制御方法を定義します。Firmware (CPU)設定は、ユーザーのファームウェアがファン速度を制御する場合に使用されます。ファームウエアは PWM デューティーサイクルを設定し、ファン速度を読み出すことにより適切なアクションを決定します。

Hardware (UDB)設定では、PSoC 内のハードウェアブロックは、CPU の介在なしに自動的にファン速度を制御します。ファームウエアは各ファンに希望の速度を設定し、ハードウエアは自動的に PWM デューティーサイクルを調整して指定許容誤差内で希望の速度を達成、維持します。デフォルト設定は Firmware (CPU)です。

Firmware (CPU)設定は以下の場合に選択します。

1. 設計者が PSoC 内のファームウェアで実施する、複雑なまたはカスタムのファンコントロールアルゴリズムを持っている場合。
2. 外部のホストコントローラが、ファン速度アルゴリズムの管理をし、Fan Controller コンポーネントが単純にファンへのハードウェアインタフェースとして使用される場合。
3. 複数のファンが共通の PWM 駆動信号を共有するバンクに関連付けされる場合。

Hardware (UDB)設定は、設計者が複数のファンを個別に制御し、最小のファームウェア開発でそれを行ないたい場合に選択してください。

Hardware Control Mode – Damping Factor(制動係数)

ハードウェアコントロールファンモードで、このパラメータはハードウェアコントロールループのダイナミック応答時間を制御します。このパラメータは、ハードウェアが各ファンの PWM デューティーサイクルを調整する頻度を制御します。

数個のファンしかない状況において、より高い制動係数は、コントロールループが希望の速度ターゲットの周りで発振することなく希望の速度に調整することを可能にします。多数のファンが制御されている場合、より低い制動係数は、ファン速度の変更に対して適切な応答時間を保証します。このパラメータは、選択したファンの電気機械特性に一致させるためハードウェアコントロールロジックのファン微調整を可能にします。このパラメータの範囲は 0～1.27
(秒指定)。デフォルト設定は 0.64 です。

Hardware Control Mode – Tolerance(許容値)

ハードウェアコントロールファンモードにおいて、このパラメータは、希望のファン速度を指定するとき利用可能な許容値を設定します。許容値は、希望の速度設定に対するパーセントとして指定されます。このパラメータは、選択したファンの電気機械特性に一致させるため、ハードウェアコントロールロジックの微調整を可能にします。

このパラメータの範囲は 1～10%です。デフォルト設定は 1%です。8 ビット PWM 分解能が Fan Controller の Fans タブ(図 2 参照)で選択されると、許容値パラメータは 5%に設定することを推奨します。

Hardware Control Mode – Acoustic Noise Reduction(音響ノイズ除去)

ハードウェアコントロールファンモードで、このパラメータは速度変更の増加率を制限してファンからの音響ノイズを制限します。イネーブルにされ、ファームウェアは希望のファン速度の増加を要求すると、PWM デューティーサイクルは急なステップ状に変化するのではなく、新しい設定値に徐々に増加していきます。これは、急な速度増加による騒がわしいファン回転音を除去します。デフォルト設定では選択されています。

Alerts – FanStall / RotorLock(ファン失速／回転子ロック)

ファンコントローラは、機械障害により失速したり回転を停止した場合、アクティブハイアラート信号を生成するように構成することができます。デフォルト設定では選択されています。

Alerts – UDB SpeedRegulationFailure(速度調整の失敗)

ハードウェアコントロールループが希望速度を達成することができない場合、ハードウェアコントロールファンモードにおいて、ファンコントローラは、アクティブハイアラート信号を生成するよう構成することができます。デフォルト設定では選択されていません。

Connections – Display as Bus

選択すると、tach 入力と fan/bank 出力がバスとして表示されます。選択しない場合は、個々の端子として表示されます。

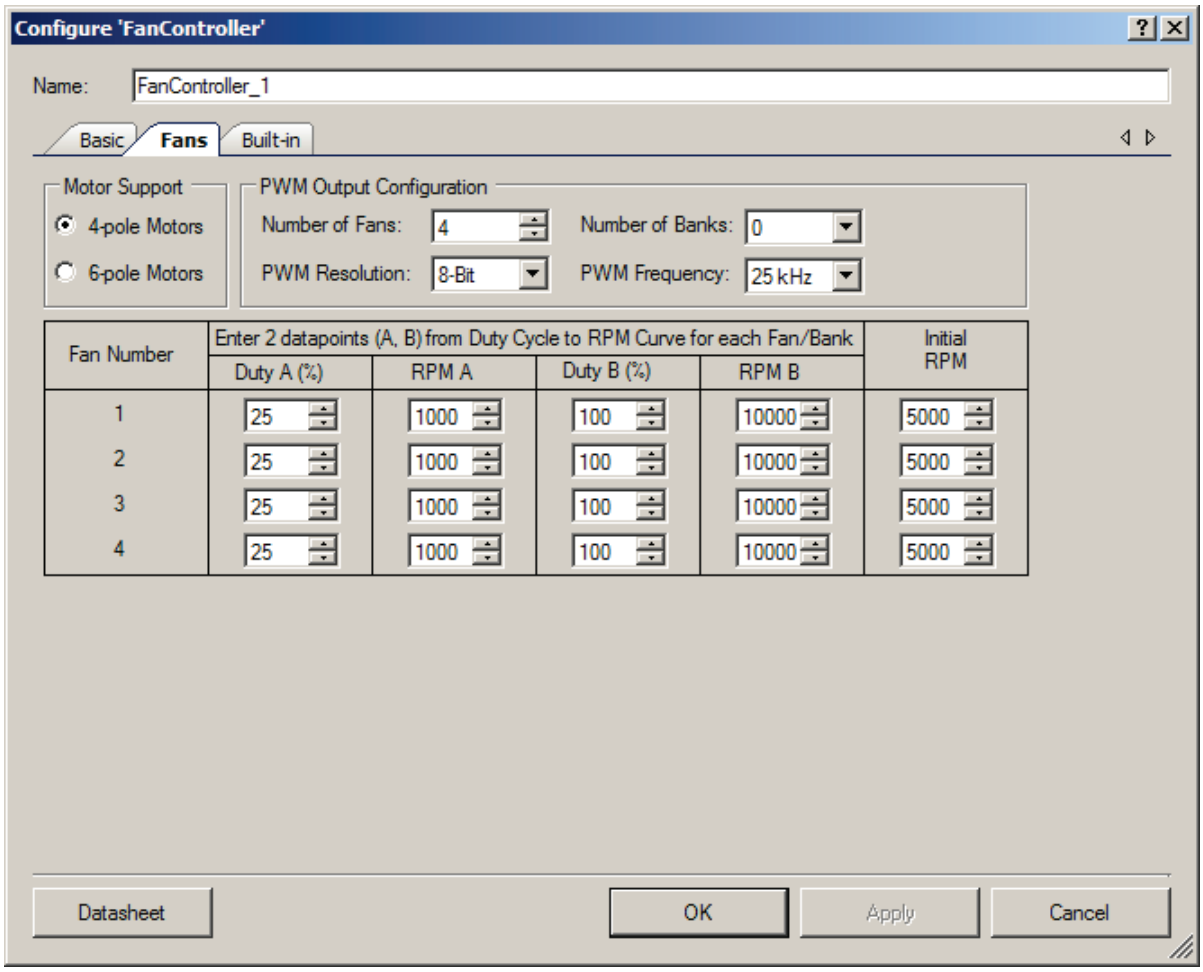
Connections – External Clock

選択すると、外部クロックソースをコンポーネントクロック入力端子に接続できます。選択しない場合は、内部クロックソースが使用されます。

Fans タブのオプション

このタブは、ファン固有のパラメータを設定するのに使用されます。

図 2. Fans Configuration ダイアログ



Motor Support

このパラメータは、ファン 1 回転当たりの回転速度計出力に現れるハイーローパルスの数を指定します。4 極オプションはファン 1 回転当たり 2 つのハイ、2 つのローパルスがあること、一方 6 極オプションはファン 1 回転当たり 3 つのハイ、3 つのローパルスがあることを示します。

PWM Output Configuration – Number of Fans

このパラメータはシステム内のファン数を指定します。このパラメータ値の許容範囲は 1～16 です。デフォルト設定は 4 です。

PWM Output Configuration – Number of Banks

このパラメータははファームウェアコントロールモードでのみ有効で、システム内のファンバンク数を指定します。ファンがバンクに構成されている場合、各バンクが同じファン数になります。したがって、ファン数はバンク数で割り切れる必要があります。値「0」はファンはバンクされていないことを示します。バンク操作の場合、このパラメータ値の有効範囲は 1～(ファン数/2)です。デフォルト設定は 0 です。

PWM Output Configuration – PWM Resolution

このパラメータは、回転速度を制御するためのファンを駆動する、変調された PWM 信号のデューティーサイクルの分解能を指定します。このパラメータの有効な値は 8 ビットか 10 ビットのいずれかです。デフォルト設定は 8 ビットです。

PWM Output Configuration – PWM Frequency

このパラメータは、ファンを駆動する変調された PWM 信号の周波数を指定します。内部クロックを使用したとき、このパラメータの有効なオプションは 25kHz または 50kHz です。デフォルト設定は 25kHz です。外部クロックを使用したときは、PWM 周波数は入力クロックソースに依存するのでこのパラメータは無効になります。詳細については、「Functional Description(機能詳細)」セクションを参照してください。

Fan Specifications – Duty A (%), RPM A

これらパラメータは、選択したファンまたはファンバンクのデューティーサイクルーRPM 伝達関数における 1 つのデータポイントを指定します。RPM A パラメータは、Duty A (%)のデューティーサイクルの PWM で駆動されたときの公称ファン速度を指定します。この情報に関しては、ファン製造メーカーのデータシートを参照してください。Fan Controller コンポーネントは、Duty A (%)にゼロ以外の値が設定されていても、PWM デューティーサイクル 0%で駆動する場合があることに注意ください。

Duty A (%)パラメータ値の有効範囲は 0～99 です。デフォルト設定は 25 です。

RPM A パラメータ値の有効範囲は 500～24,999 です。デフォルト設定は 1,000 です。

Fan Specifications – Duty B (%), RPM B

これらパラメータは、選択したファンやファンバンクのデューティーサイクルーRPM 伝達関数における別のデータポイントを指定します。RPM B パラメータは、Duty B (%)のデューティーサイクルの PWM で駆動されたときの公称ファン速度を指定します。この情報に関しては、ファン製造メーカーのデータシートを参照してください。Fan Controller コンポーネントは、Duty B (%)が 100%より低くに設定されていても、PWM デューティーサイクル最大 100%で駆動することがあることに注意ください。

Duty B (%)パラメータ値の有効範囲は 1～100 です。デフォルト設定は 100 です。

RPM B パラメータ値の有効範囲は 501～25,000 です。デフォルト設定は 10,000 です。

Fan Specifications – Initial RPM

このパラメータは個々のファンの初期 RPM を指定します。初期 RPM の値はデューティーサイクルに変換され、個々のファンに対する初期デューティーサイクルとして設定されます。Initial RPM パラメータ値は RPM A パラメータ値より低くに設定することができます。

クロック選択

コンポーネントはその操作にクロックツリーソースを使用します。以下のクロックがあります。Bus Clock、Tachometer Clock (500kHz)、PWM Clock (6、12、24MHz。構成によって異なります)。コンポーネントには、PWM Clock でなく外部クロックソースを接続するオプションがあります。

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)ルーチンにより、ファームウェアを使用してコンポーネントとやりとりできます。次の表は、各関数へのインターフェースとその説明を示しています。続くセクションでは、各関数について詳しく説明します。

PSoC Creator は、設計コンポーネントの最初のインスタンスにデフォルトで「FanController_1」というインスタンス名を割り当てます。コンポーネントのインスタンス名は、識別子の文法ルールに従って固有の名前に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。便宜上、次の表ではインスタンス名として「FanController」を使っています。

関数

関数	説明
FanController_Start()	コンポーネントを起動します
FanController_Stop()	コンポーネントを停止し、ハードウェアブロックを無効にします
FanController_Init()	コンポーネントを初期化します
FanController_Enable()	コンポーネント内のハードウェアブロックを有効にします
FanController_EnableAlert()	コンポーネントからのアラートを有効にします
FanController_DisableAlert()	コンポーネントからのアラートを無効にします
FanController_SetAlertMode()	アラートソースを設定します

関数	説明
FanController_GetAlertMode()	現在有効なアラートソースを戻します
FanController_SetAlertMask()	各ファンからのアラートのマスキングを有効にします
FanController_GetAlertMask()	各ファンのアラートマスキングステータスを戻します
FanController_GetAlertSource()	保留中のアラートソースを戻します
FanController_GetFanStallStatus()	各ファンのストールステータスを示すビットマスクを戻します
FanController_GetFanSpeedStatus()	ハードウェアコントロールモードにおいて各ファンの速度調整ステータスを示すビットマスクを戻します
FanController_SetDutyCycle()	指定したファンやファンバンクのPWMデューティーサイクルを設定します
FanController_GetDutyCycle()	指定したファンやファンバンクのPWMデューティーサイクルを戻します
FanController_SetDesiredSpeed()	ハードウェアコントロールモードにおいて指定したファンの希望ファン速度を設定します
FanController_GetDesiredSpeed()	ハードウェアコントロールモードにおいて指定したファンの希望ファン速度を戻します
FanController_GetActualSpeed()	指定したファンの実速度を戻します
FanController_OverrideHardwareControl()	ハードウェア (UDB) ファンコントロールを無効にしてファームウェアを有効にします

グローバル変数

変数	説明
FanController_initVar(static)	initVar変数は、このコンポーネントの初期設定を指定するのに使用されます。この変数はコンポーネント名とともに事前には未決です。この変数は、ゼロに初期化され、FanController_Start()が最初に呼び出されたとき1に設定されます。これにより、FanController_Start()ルーチンに続くすべての呼び出しにおいて、再初期化することなくコンポーネントの初期化ができます。 コンポーネントの再初期化が必要な場合は、FanController_Stop()ルーチンは、FanController_Init()とFanController_Enable()に先立って呼び出されます。

voidFanController_Start(void)

説明: コンポーネントをイネーブルにします。コンポーネントが以前に初期化されていない場合は、Init() APIを呼び出します。Enable() APIを呼び出します。

パラメータ: なし

返り値: なし

副作用: なし

voidFanController_Stop(void)

説明: コンポーネントを無効にします。すべてのPWM出力は100% デューティーサイクルで駆動され、コンポーネントの非稼働中確実に冷却が継続されます。

パラメータ: なし

返り値: なし

副作用: アラートピンはアサート解除されます。

voidFanController_Init(void)

説明: コンポーネントを初期化します。

パラメータ: なし

返り値: なし

副作用: なし

voidFanController_Enable(void)

説明: コンポーネント内のハードウェアブロックを有効にします。

パラメータ: なし

返り値: なし

副作用: なし

voidFanController_EnableAlert(void)

説明: アラート信号の生成を有効にします。有効にするアラートソースは、FanController_SetAlertMode()とFanController_SetAlertMask()のAPIを使用することにより明確に設定されます。

パラメータ: なし

返り値: なし

副作用: なし

voidFanController_DisableAlert(void)

説明: アラート信号の生成を無効にします。

パラメータ: なし

返り値: なし

副作用: アラートピンはアサート解除されます。

voidFanController_SetAlertMode(uint8 alertMode)

説明: コンポーネントからアラートソースを設定します。2つのアラートソースが利用できます。1) ファンストールまたはローターロック、2) ハードウェアコントロールモードの速度調整エラー。

パラメータ: uint8 alertMode

ビットフィールド	有効なアラートソース
FanController_STALL_ALERT	1=ファンストール / ローターロックアラートを有効にする
FanController_SPEED_ALERT	1=閉ループ速度調整エラーアラートを有効にする

返り値: なし

副作用: なし

uint8FanController_GetAlertMode(void)

説明: 有効なアラートソースを戻します。

パラメータ: なし

返り値: uint8 alertMode

ビットフィールド	有効なアラートソース
FanController_STALL_ALERT	1=ファンストール / ローターロックアラートを有効にする
FanController_SPEED_ALERT	1=閉ループ速度調整エラーアラートを有効にする

副作用: なし

voidFanController_SetAlertMask(uint16 alertMask)

説明: マスクを介して各ファンからのアラートを有効または無効にします。ファンストールアラートと速度調整エラーアラートの両方にマスキングを適用します。

パラメータ: uint16 alertMask

ビットフィールド	有効なアラートソース
bit0	1=Fan1のアラートを有効にする
bit1	1=Fan2のアラートを有効にする
…	…
bit15	1=Fan16のアラートを有効にする

返り値: なし

副作用: なし

uint16FanController_GetAlertMask(void)

説明: 各ファンのアラートマスクステータスを戻します。ファンストールアラートと速度調整エラーアラートの両方にマスキングを適用します。

パラメータ: なし

返り値: uint16 alertMask

ビットフィールド	有効なアラートソース
bit0	1=Fan1のアラートを有効にする
bit1	1=Fan2のアラートを有効にする
…	…
bit15	1=Fan16のアラートを有効にする

副作用: なし

uint8FanController_GetAlertSource(void)

説明: コンポーネントから保留アラートソースを戻します。このAPIは、コンポーネントのアラートステータスをポーリングするのに使用することができます。一方、アラートピンがPSoCのCPUコアへの割り込みを発生するのに使用される場合、割り込みサービスルーチンはこのAPIを使用して、アラートのソースを決定することができます。いずれかの場合で、このAPIがゼロ以外の値を戻す場合、FanController_GetFanStallStatus()およびFanController_GetFanSpeedStatus() APIは、エラーが発生しているファンに関する詳細情報を提供することができます。

パラメータ: uint8 alertMode

ビットフィールド	保留アラート
FanController_STALL_ALERT	1=ファンストール / ローターロックアラート保留
FanController_SPEED_ALERT	1=閉ループ速度調整エラーアラート保留

返り値: なし

副作用: なし

uint16FanController_GetFanStallStatus(void)

説明: すべてのファンのストール / ローターロックステータスを戻します。

パラメータ: なし

返り値: uint16 stallStatus

ビットフィールド	ステータス
bit0	Fan1ストールステータス(1=ストール、0=OK)
bit1	Fan2ストールステータス
…	…
bit15	Fan16ストールステータス

副作用: このAPIを呼び出すと、すべての保留ファンストールアラートをクリアし、アラートピンのアサートを解除します。ストールアラートが保留中の場合、アラートピンは次のサイクルの終了(eoc)パルスに同期して再アサートされます。

uint16FanController_GetFanSpeedStatus(void)

説明: すべてのファンのハードウェアファンコントロールモード速度調整ステータスを戻します。速度調整エラーは以下の2つのケースで発生します。1) 希望ファン速度が現在の実ファン速度を超え、ファンのデューティーサイクルが既に100%である場合、2) 希望ファン速度が現在の実ファン速度より低く、ファンのデューティーサイクルが既に0%である場合。

パラメータ: なし

返り値: uint16 speedStatus

ビットエラー	状態
bit0	Fan1の速度調整ステータス (1=エラー、0=OK)
bit1	Fan2の速度調整ステータス
…	…
bit15	Fan16の速度調整ステータス

副作用: このAPIを呼び出すと、すべての保留速度調整アラートをクリアし、アラートピンのアサートを解除します。速度調整アラートが保留の場合、アラートピンは次のサイクルの終了(eoc)パルスに同期して再アサートされます。

voidFanController_SetDutyCycle(uint8 fanOrBankNumber, uint16 dutyCycle)

説明: 選択したファンやファンバンクのPWMデューティーサイクルパーセント値の100倍を設定します。ハードウェアファンコントロールモードにおいて、手動デューティーサイクルコントロールが望ましい場合は、このAPIを呼び出す前にFanController_OverrideHardwareControl() APIを呼び出します。

パラメータ: uint8 fanOrBankNumber
値の有効範囲は1～16ですが、システム内のファン数やバンク数以下にする必要があります。

uint16 dutyCycle
1/100%単位のデューティーサイクル。例えば、50%デューティーサイクル = 5000。有効範囲は0～10000です。

返り値: なし

副作用: なし

uint16FanController_GetDutyCycle(uint8 fanOrBankNumber)

説明: 選択したファンやファンバンクの現在のPWMデューティーサイクルを1/100%単位でで戻します。

パラメータ: uint8 fanOrBankNumber
値の有効範囲は1～16ですが、システム内のファン数やバンク数以下にする必要があります。

返り値: 1/100%単位のデューティーサイクル。例えば、50%デューティーサイクル = 5000。

副作用: なし

voidFanController_SetDesiredSpeed(uint8 fanNumber, uint16 rpm)

説明: 指定したファンの希望速度を1分当たりの回転(RPM)で設定します。ハードウェアファンコントロールモードでは、RPMパラメータは、調整のため新しいターゲットファン速度としてコントロールループハードウェアに渡されます。ファームウェアファンコントロールモードにおいて、RPMパラメータは、カスタマイザのFansタブに入力したファンパラメータに基づいてデューティーサイクルに変換され、適切なPWMへ書き込まれます。これは、粗レベルの速度コントロールを初期化するための方法をファームウェアに提供します。細レベルファームウェア速度コントロールはFanController_SetDutyCycle() APIを使用して実現できます。

パラメータ: uint8 fanNumber
有効範囲は1～16ですが、システム内のファン数以下にする必要があります。

uint16 rpm
有効範囲は500～25,000ですが、ファンが稼働できる最大RPM以下にする必要があります。有効範囲以外の場合は速度調整エラーが発生します。

返り値: なし

副作用: なし

uint16FanController_GetDesiredSpeed(uint8 fanNumber)

説明: 選択したファンの現在の希望速度を戻します。

パラメータ: uint8 fanNumber
有効範囲は1～16ですが、システム内のファン数以下にする必要があります。

返り値: 選択したファンの現在の希望速度/RPM

副作用: なし

uint16 FanController_GetActualSpeed(uint8 fanNumber)

説明: 選択したファンの現在の実速度を戻します。

パラメータ: uint8 fanNumber
有効範囲は1～16ですが、システム内のファン数以下にする必要があります。

返り値: 選択したファンの現在の実速度/RPM

副作用: なし

注 PSoC3 デバイスでは、ハードウェアモード動作時、特定の時間窓内で FanController_GetActualSpeed()を呼び出す必要があります。詳細については、「Functional Description(機能詳細)」セクションを参照してください。

voidFanController_OverrideHardwareControl(uint8 override)

説明: ハードウェアファンコントロールモードにおいて、ファームウエアがファン制御を継承することを許可します。このAPIは、ファームウェアファンコントロールモードにおいては呼び出すことはできません。

パラメータ: uint8 override
0 = ハードウェアがファンコントロールする
1 = ファームウェアがファンコントロールする
有効範囲は0～1です。デフォルトは0です

返り値: なし

副作用: なし

ファームウェア・ソースコードのサンプル

PSoC Creator は、Find Example Project ダイアログに数多くのサンプルプロジェクトを提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、コンポーネントカタログまたは回路図に置いたコンポーネントインスタンスからダイアログを開きます。一般例については、「Start Page (スタートページ)」または **File** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

詳しくは、PSoC Creator Help の「Find Example Project(プロジェクト例を検索)」トピックを参照してください。

割り込みサービスルーチン

ファンコントローラコンポーネントは自動的には割り込みサービスルーチン API を提供していません。このコンポーネント用の割り込みを使用するには、PSoC Creator の Cypress 標準カタログから割り込みコンポーネントを追加します。

機能説明

Custom Clock(カスタムクロック)

コンポーネントには、外部クロックに接続できる機能があります。クロックソースの周波数は、PWM 出力周波数を決定します。入力クロック周波数(f_{CLK}) と出力 PWM 周波数(f_{PWM})との関係は以下の方程式で表されます。

$$f_{PWM} = f_{CLK} / 240$$、8 ビット分解能 の場合。

$$f_{PWM} = f_{CLK} / 960$$。10 ビット分解能の場合。

(240 と 960 の定数は 8 ビットと 10 ビット分解能モードそれぞれの最大周期期間です)。

Actual Speed(実速度)

このコンポーネントが PSoC 3 デザインで使用される場合、16 ビット速度値を正確に取り込むには、「eoc」パルスの生成後特定の時間フレーム(tACT)内にこの API を呼び出す必要があります。この要因は、16 ビット値を一度に 8 ビット単位で読み取る 8051 CPU コアによります。この領域外では、実速度 MSB と実速度 LSB の読み取りに 2 つの異なるファン速度測定(過去と現在)からのデータが含まれる可能性があります。

このようなことを起こらないようにするには、「eoc」パルスの生成後以下の期間内に GetActualSpeed() API を呼び出す必要があります

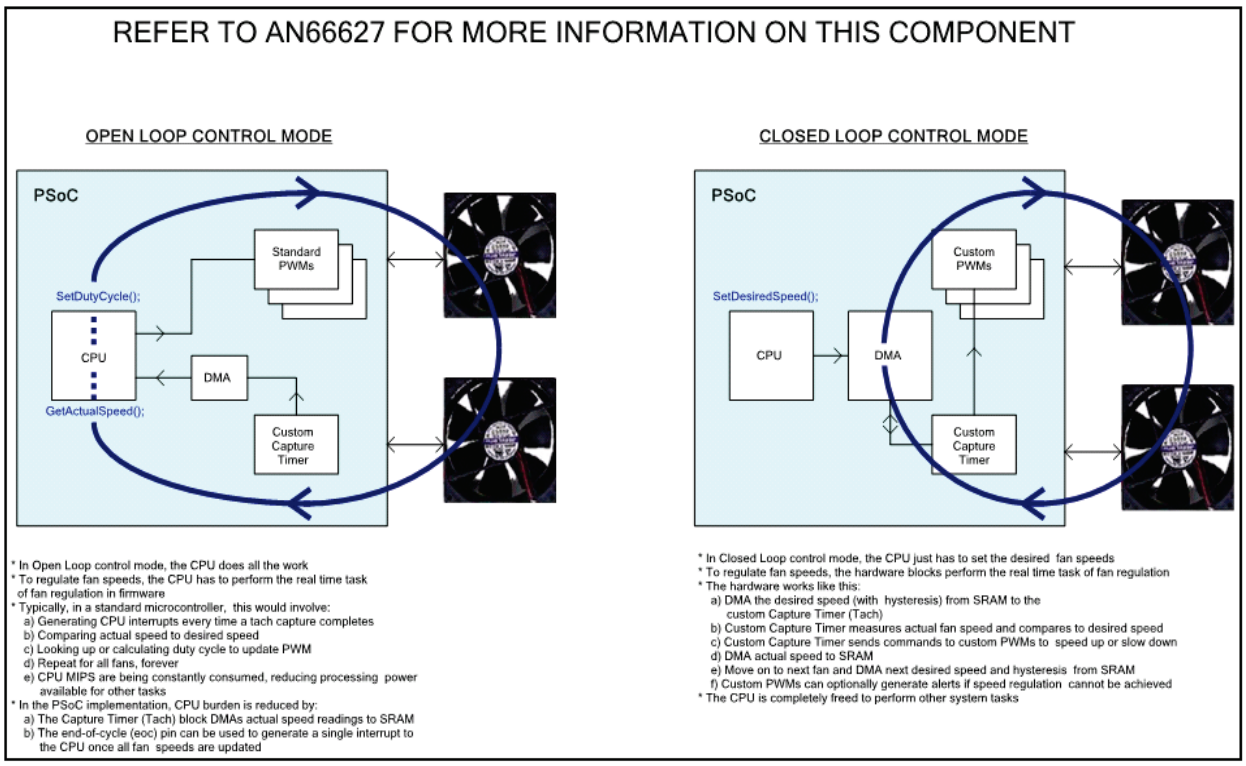
$$t_{ACT} = 2 * 60/RPMn_{MAX}$$

注　$RPMn_{MAX}$は fanN(設定で使用される 最後のファン)の最大回転速度です。

ブロックダイアグラムと設定

図 3 で示されている図式はコンポーネントの 2 つの基本操作モードの上位ブロック図です。1) 開ループコントロールモード(ファームウェアがファンを速度調整)、2) 閉ループコントロールモード(ハードウェアがファン速度を調整)。

図 3. ファンコントローラのブロック図



レジスタ

ファンコントローラには、操作を制御し、ステータスをモニターするファームウェア API によって使用される、複数のコントロール、ステータスレジスタがあります。これらレジスタは、ユーザーファームウェアによって直接アクセスすることはできません。

リソース

ファンコントローラコンポーネントは UDB アレイ全体に渡って配置されます。コンポーネントは以下のリソースを利用します。

設定[1]	リソースのタイプ					
	データパスセル	マクロセル	ステータスセル	コントロールセル	DMA チャネル	割り込み
ファームウェアモード、8ビット、4個のファン	4	28	3	4	1	–
ファームウェアモード、8ビット、8個のファン	6	36	3	4	1	–
ファームウェアモード、8ビット、12個のファン	8	46	4	5	1	–
ファームウェアモード、8ビット、16個のファン	10	56	4	5	1	–
ファームウェアモード、10ビット、4個のファン	6	28	3	4	1	–
ファームウェアモード、10ビット、8個のファン	10	36	3	4	1	–
ファームウェアモード、10ビット、12個のファン	14	46	4	5	1	–
ファームウェアモード、10ビット、16個のファン	18	56	4	5	1	–
ハードウェアモード、8ビット、4個のファン	6	57	4	8	2	–
ハードウェアモード、8ビット、8個のファン	10	93	4	12	2	–
ハードウェアモード、8ビット、12個のファン	14	132	6	17	2	–
ハードウェアモード、10ビット、4個のファン	10	57	4	8	2	–
ハードウェアモード、10ビット、8個のファン	18	93	4	12	2	–

API メモリ使用率

コンポーネントのメモリ使用率は、コンパイラ、デバイス、使用する API 数、コンポーネントの構成によって非常に異なります。以下の表は、特定のコンポーネント構成で利用可能なすべての API に対するメモリ使用率です。

測定は、最適化サイズのリリースモードで構成された関連コンパイラで行われました。特定のデザインに対しては、コンパイラで生成されたマップファイルを分析してメモリ使用率を決定することができます

1. ハードウェアモードの場合、制動係数機能によって使用されるリソースは含まれません。

設定[2]	PSoC 3 (Keil_PK51)		PSoC 5 (GCC)		PSoC 5LP (GCC)	
	フラッシュ バイト	SRAM バイト	フラッシュ バイト	SRAM バイト	フラッシュ バイト	SRAM バイト
ファームウェアモード、8 ビット、4 個のファン	1396	87	856	106	856	106
ファームウェアモード、8 ビット、8 個のファン	1396	171	840	206	840	206
ファームウェアモード、8 ビット、12 個のファン	1413	255	872	306	872	306
ファームウェアモード、8 ビット、16 個のファン	1413	339	876	406	880	406
ファームウェアモード、10 ビット、4 個のファン	1425	87	872	106	868	106
ファームウェアモード、10 ビット、8 個のファン	1425	171	852	206	852	206
ファームウェアモード、10 ビット、12 個のファン	1442	255	884	306	888	306
ファームウェアモード、10 ビット、16 個のファン	1442	339	888	406	892	406
ハードウェアモード、8 ビット、4 個のファン	2222	129	1380	163	1380	163
ハードウェアモード、8 ビット、8 個のファン	2222	253	1380	319	1380	319
ハードウェアモード、8 ビット、12 個のファン	2243	377	1428	475	1424	475
ハードウェアモード、10 ビット、4 個のファン	2239	129	1388	163	1388	163
ハードウェアモード、10 ビット、8 個のファン	2239	253	1388	319	1388	319

2. ハードウェアモードの場合、制動係数機能によって使用されるリソースは含まれません。

DC 電気的特性と AC 電気的特性

特記されているところ以外は、仕様は–40°C ≤ T_A ≤ 85°C および T_J ≤ 100°C で有効です。仕様は、特に注記した場合を除いて、1.71V～5.5V において有効です。

DC 特性(PWM クロック – 6 MHz)

パラメータ	説明	Min	Typ[3]	Max	単位
Idd	コンポーネントの消費電流(ファームウェアモード、4 個のファン)				
	8 ビット分解能	–	108	–	µA
	10 ビット分解能	–	139	–	µA
	コンポーネントの消費電流(ファームウェアモード、8 個のファン)				
	8 ビット分解能	–	194	–	µA
	10 ビット分解能	–	260	–	µA
	コンポーネントの消費電流(ファームウェアモード、12 個のファン)				
	8 ビット分解能	–	307	–	µA
	10 ビット分解能	–	395	–	µA
	コンポーネントの消費電流(ファームウェアモード、16 個のファン)				
	8 ビット分解能	–	392	–	µA
	10 ビット分解能	–	505	–	µA
	コンポーネントの消費電流(ハードウェアモード、4 個のファン)				
	8 ビット分解能	–	225	–	µA
	10 ビット分解能	–	269	–	µA
	コンポーネントの消費電流(ハードウェアモード、8 個のファン)				
	8 ビット分解能	–	417	–	µA
	10 ビット分解能	–	518	–	µA
	コンポーネントの消費電流(ハードウェアモード、12 個のファン)				
	8 ビット分解能	–	636	–	µA

3. デバイス I/O とクロック分配電流は含まれていません。ルーティング条件、tach 入力の周波数、および温度などのその他要因も消費電流に影響を与えます。値は 25°C の場合です。

DC 特性(PWM クロック – 12 MHz)

パラメータ	説明	Min	Typ	Max	単位
Idd	コンポーネントの消費電流(ファームウェアモード、4 個のファン)				
	8 ビット分解能	–	169	–	µA
	10 ビット分解能	–	233	–	µA
	コンポーネントの消費電流(ファームウェアモード、8 個のファン)				
	8 ビット分解能	–	310	–	µA
	10 ビット分解能	–	439	–	µA
	コンポーネントの消費電流(ファームウェアモード、12 個のファン)				
	8 ビット分解能	–	498	–	µA
	10 ビット分解能	–	677	–	µA
	コンポーネントの消費電流(ファームウェアモード、16 個のファン)				
	8 ビット分解能	–	641	–	µA
	10 ビット分解能	–	871	–	µA
	コンポーネントの消費電流(ハードウェアモード、4 個のファン)				
	8 ビット分解能	–	403	–	µA
	10 ビット分解能	–	492	–	µA
	コンポーネントの消費電流(ハードウェアモード、8 個のファン)				
	8 ビット分解能	–	756	–	µA
	10 ビット分解能	–	951	–	µA
	コンポーネントの消費電流(ハードウェアモード、12 個のファン)				
	8 ビット分解能	–	1162	–	µA

DC 特性(PWM クロック – 24 MHz)

パラメータ	説明	Min	Typ	Max	単位
Idd	コンポーネントの消費電流(ファームウェアモード、4 個のファン)				
	8 ビット分解能	–	290	–	µA
	10 ビット分解能	–	417	–	µA
	コンポーネントの消費電流(ファームウェアモード、8 個のファン)				
	8 ビット分解能	–	545	–	µA
	10 ビット分解能	–	798	–	µA
	コンポーネントの消費電流(ファームウェアモード、12 個のファン)				
	8 ビット分解能	–	887	–	µA
	10 ビット分解能	–	1243	–	µA
	コンポーネントの消費電流(ファームウェアモード、16 個のファン)				
	8 ビット分解能	–	1150	–	µA
	10 ビット分解能	–	1607	–	µA
	コンポーネントの消費電流(ハードウェアモード、4 個のファン)				
	8 ビット分解能	–	753	–	µA
	10 ビット分解能	–	938	–	µA
	コンポーネントの消費電流(ハードウェアモード、8 個のファン)				
	8 ビット分解能	–	1446	–	µA
	10 ビット分解能	–	1826	–	µA
	コンポーネントの消費電流(ハードウェアモード、12 個のファン)				
	8 ビット分解能	–	2220	–	µA

AC 仕様

パラメータ	説明	Min	Typ	Max	単位
f_{pwm_clk}	入力クロック周波数[4]				
	ファームウェアモード、8ビット、4個のファン	–	–	54	MHz
	ファームウェアモード、8ビット、8個のファン	–	–	46	MHz
	ファームウェアモード、8ビット、12個のファン	–	–	39	MHz
	ファームウェアモード、8ビット、16個のファン	–	–	28	MHz
	ファームウェアモード、10ビット、4個のファン	–	–	42	MHz
	ファームウェアモード、10ビット、8個のファン	–	–	42	MHz
	ファームウェアモード、10ビット、12個のファン	–	–	41	MHz
	ファームウェアモード、10ビット、16個のファン	–	–	37	MHz
	ハードウェアモード、8ビット、4個のファン	–	–	57	MHz
	ハードウェアモード、8ビット、8個のファン	–	–	57	MHz
	ハードウェアモード、8ビット、12個のファン	–	–	40	MHz
	ハードウェアモード、10ビット、4個のファン	–	–	50	MHz
	ハードウェアモード、10ビット、8個のファン	–	–	47	MHz
f_{tach_clk}	回転速度計クロック周波数	–	500	–	kHz
Tach分解能	回転速度計カウンタの分解能	–	16	–	ビット
f_{tach}	回転速度計速度	500[5]	–	25000	RPM

4. 値はファンが安全に動作する PWM 周波数の最大値を示します。コンポーネントをより高いクロック周波数で駆動することは可能ですが、タイミングの要求仕様を STA（Static Timing Analysis）の結果で検証する必要があります。

5. 速度が最小値未満の場合は中断の割り込みが発生します。

コンポーネントの変更

ここでは、以前のバージョンからコンポーネントに加えられた主な変更を示します。

バージョン	変更の説明	変更の理由 / 影響
2.10	コンポーネント特性データの更新。	
	PSoC 5LP対応の追加	
2.0	コンポーネント特性データの追加。	
	制動係数動作の変更。	旧バージョンでは、Damping Factor(制動係数)は各ファンの速度計測間の遅延を指定し、遅延の値の単位はありませんでした。このバージョンでは、すべてのファンの速度測定の開始から開始までの遅延を指定するようになりました。さらに、この遅延は秒単位で指定されるようになりました。
	Configuration(設定)に各ファンの新しいパラメータ、Initial RPMが追加されました。	このパラメータは、コンポーネント開始時の特定ファンの適切な回転速度を指定します。旧バージョンでは、すべてのファンが最大速度で初期化されていました。
	コンポーネントでサポートされている、モータのタイプを選択する機能が追加されました。	このパラメータはファンの1回転当たりのハイーローパルス数を指定します。4極モーター用の2ハイ-ローパルス、6極モーター用の3ハイ-ローパルス。
	コンポーネント記号は内部PWMを駆動する外部クロックソースを追加する機能により拡張されました。このオプションはカスタマイザのGUIで利用できます。	新規コンポーネント機能。外部クロックの接続ができるようになりました。設定されたソースクロック値とPWMの分解能に基づいて、PWM出力周波数を調整することができます。
	追加された新機能は、カスタマイザのGUIで選択でき、コンポーネント入出力をバスとして表示します。	tach1～tachN、fan1～fanN、bank1～bankNの接続セットがバスとして表示されます。このオプションはコンポーネント記号のサイズを小さくするので、説明図の領域を節減することができます。
	Keil関数の再入力可能のサポートをAPIに追加しました。	顧客が個々に生成された関数を再入可能として指定する機能を追加しました。
	削除された旧関数名、FanController_OverrideClosedLoop() (FanController_OverrideHardwareControl()の簡易#define)	前のバージョンでは廃止と明示されていたので、このバージョンで削除されました。

バージョン	変更の説明	変更の理由 / 影響
1.20	1. PSoC Creator v2.0に対応 2.「Concept」コンポーネントとして再分類化 3. SetDesiredSpeed() APIの精度改善のため修正	
1.10	1. SetDesiredSpeed() APIの修正 2. 回転速度計入力にグリッチフィルタを追加 3. ファンコントロール野用語を開/閉ループからファームウェア/ハードウェアコントロール方式に変更 4. 記号の色とサイズを変更 5. リソース使用率の更新(削減) 6. 電力管理APIに対するリファレンスの削除(未対応)	
1.0	初版	

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 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。特許又はその他の権限下で、ライセンスを譲渡又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、又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を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ツ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提供することは、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 Designer™及び Programmable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)は Cypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並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る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及び/又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複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及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、又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、明示的又は黙示的を問わず、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、商品性又は特定目的への適合性の黙示的な保証が含まれますが、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又は回路を適用又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ツ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提供することは、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。