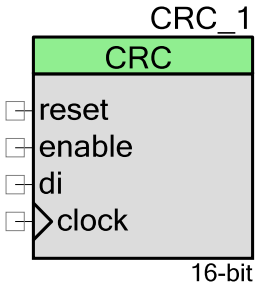
巡回冗長チェック（CRC）

2.0

特長



- 1～64 ビット
- 時間分割多重化モード
- シリアル ビット ストリーム入力用のクロックおよびデータが必要
- シリアルデータ入力、パラレル出力
- 標準 [CRC-1 (パリティ ビット)、CRC-4 (ITU-T G.704)、CRC-5-USB など] 、またはカスタム多項式
- 標準またはカスタムのシード値
- イネーブル入力が、その他のコンポーネントとの同期動作を提供

概要説明

巡回冗長チェック (CRC) コンポーネントのデフォルトでの使用方法は、あらゆる長さのシリアル ビット ストリームから CRC を計算するためです。データ クロックの立ち上がりエッジで、入力データがサンプリングされます。CRC 値は、開始前に 0 に設定されるか、オプションで初期値をシードとして使用できます。ビットストリームが完了すると、計算された CRC 値が読み出されます。

CRC を使用する場合

デフォルトの CRC コンポーネントは、送信やストレージ中にデータの変化を検出するため、チェックサムとして使用されます。CRC は、バイナリ ハードウェアに簡単に実装できる、数学的に簡単に解析が可能、送信チャネルでのノイズによって引き起こされる共通エラー検出に特に優れているなどの点から、よく利用されます。

入出力接続

ここでは、CRC のさまざまな入出力接続について説明します。I/O リストのアスタリスク (*) は、I/O が、その I/O の説明でリストされている条件において、シンボルに隠れている可能性があることを示します。

clock – 入力

CRC は、CRC の計算に使用される、シリアル ビットストリームを提供するデータ入力を必要とします。シリアル データ入力を正しくサンプリングするために、データ クロック入力も必要です。データ クロックの立ち上がりエッジで、入力データがサンプリングされます。

reset – 入力

リセット入力は、CRC を非同期リセットするための信号です。

enable – 入力

CRC コンポーネントは、開始後からイネーブル信号が HIGH である限り実行されます。この入力が、その他のコンポーネントとの同期動作を提供します。

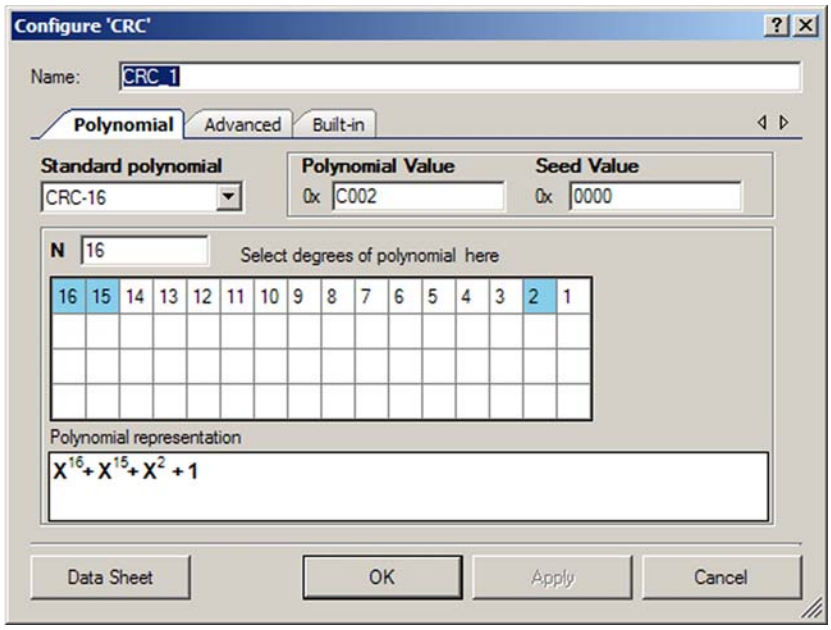
di – 入力

CRC の計算に使用される、シリアル ビットストリームを提供するデータ入力。

パラメータおよび設定

CRC コンポーネントをデザイン上にドラッグし、ダブルクリックして [Configure] (設定) ダイアログを開きます。このダイアログには、CRC コンポーネントのセットアップをガイドする複数のタブがあります。

[Polynomial (多項式)] タブ



Standard Polynomial (標準多項式)

[Standard polynomial (標準多項式)] コンボボックスに提供されている標準の CRC 多項式のいずれかを選択するか、カスタム多項式を生成できます。各標準多項式に関するその他の情報は、ツールのヒントに表示されます。デフォルトは CRC-16 です。

多項式名	多項式	使用
カスタム	ユーザ定義	全般
CRC-1	x + 1	パリティ
CRC-4-ITU	$x^4 + x + 1$	ITU G.704
CRC-5-ITU	$x^5 + x^4 + x^2 + 1$	ITU G.704
CRC-5-USB	$x^5 + x^2 + 1$	USB
CRC-6-ITU	$x^6 + x + 1$	ITU G.704
CRC-7	$x^7 + x^3 + 1$	電話通信システム、MMC

多項式名	多項式	使用
CRC-8-ATM	$x^8 + x^2 + x + 1$	ATM HEC
CRC-8-CCITT	$x^8 + x^7 + x^3 + x^2 + 1$	1 ワイヤ バス
CRC-8-Maxim	$x^8 + x^5 + x^4 + 1$	1 ワイヤ バス
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	全般
CRC-8-SAE	$x^8 + x^4 + x^3 + x^2 + 1$	SAE J1850
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	全般
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	電話通信システム
CRC-15-CAN	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$	CAN
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$	XMODEM、X.25、V.41、Bluetooth、PPP、IrDA、CRC-CCITT
CRC-16	$x^{16} + x^{15} + x^2 + 1$	USB
CRC-24-Radix64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$	全般
CRC-32-IEEE802.3	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	Ethernet、MPEG2
CRC-32C	$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$	全般
CRC-32K	$x^{32} + x^{30} + x^{29} + x^{28} + x^{26} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{11} + x^{10} + x^7 + x^6 + x^4 + x^2 + x + 1$	全般
CRC-64-ISO	$x^{64} + x^4 + x^3 + x + 1$	ISO 3309
CRC-64-ECMA	$x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$	ECMA-182

Polynomial Value (多項値)

16 進値で表されます。標準多項式のいずれかが選択されている場合は、自動的に計算されます。手動で入力することもできます (「カスタム多項」を参照)。

Seed Value (シード値)

16 進値で表されます。最大値は 2^N-1 です。

N

多項式の全次数を定義します。可能な値は 1〜64 ビットです。表での数字は、多項式に含まれる全次数を示しています。選択した数のセルは青、その他は白になっています。アクティブなセルの数は N に等しくなります。数は、降順に配列されています。セルをクリックし、数を選択または選択解除できます。

Polynomial Representation (多項式表現)

数学的な式として、多項式の結果を表示します。

Custom Polynomials (カスタム多項式)

3 つの異なる方法で、カスタム多項式を入力できます。

標準の多項式に多少の変更を加える

- 標準多項式のいずれかを選択します。
- 該当するセルをクリックし、表で必要な次数を選択します。**[Standard Polynomial (標準多項式)]** が [Custom (カスタム)] に変わります。
- 多項値が、多項式を基にして自動的に再計算されます。

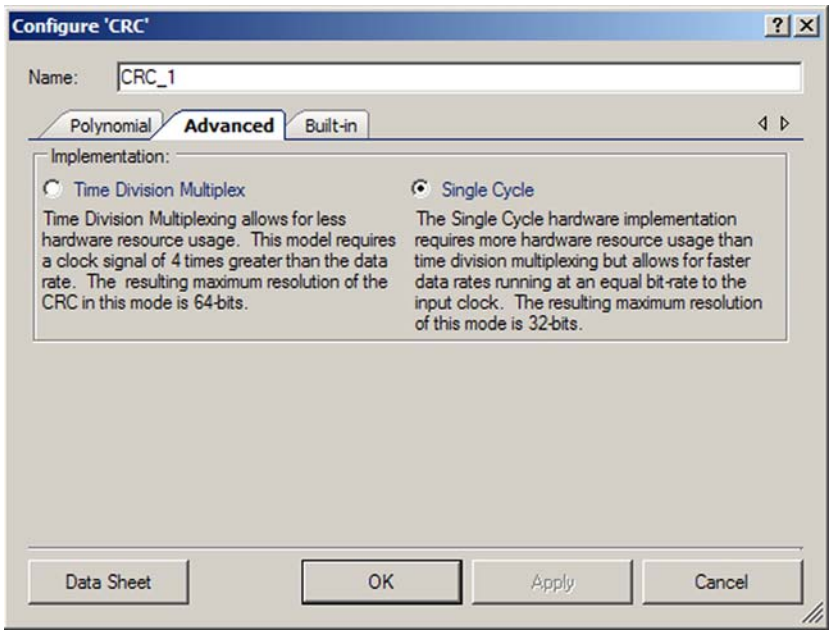
Use Polynomial Degrees (多項式の次数を使用)

- **[N]** テキストボックスにカスタム多項式を入力します。**[Standard Polynomial (標準多項式)]** が [Custom (カスタム)] に変わります。
- 表の数で必要な次数を選択します。
- **[Polynomial Representation (多項式)]** で多項式を確認します。
- 多項値が、多項式を基にして自動的に再計算されます。

Use Hexadecimal Format (16 進法形式の使用)

- **[Polynomial Value (多項値)]** テキスト ボックスに、16 進法形式で多項値を入力します。
- [Enter] を押すか、別のコントロールに切り替えます。**[Standard Polynomial (標準多項式)]** が [Custom (カスタム)] に変わります。
- N 値と多項式の次数は、入力した多項値を基にして再計算されます。

Advanced (詳細) タブ



Implementation (実装)

これは、CRC コンポーネントの実装を定義します。時間分割多重化またはシングル サイクル。デフォルトは、シングル サイクルです。

Local Parameters (ローカル パラメータ) (API の利用向け)

これらのパラメータは、API によって使用され、GUI には表示されません。

- **PolyValueLower (uint32) –** 16 進値による多項値の下位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0xB8h (LFSR= [8,6,5,4]) になります。
- **PolyValueUpper (uint32) –** 16 進値による多項値の上位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0x00h になります。
- **SeedValueLower (uint32) –** 16 進値によるシード値の下位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0xFFh になります。
- **SeedValueUpper (uint32) –** 16 進値によるシード値の上位半分が含まれます。デフォルトの分解能は 8 であるため、デフォルトは 0 になります。

クロック選択

このコンポーネントには、内部クロックはありません。クロック源を取り付ける必要があります。

注 Impremention (実装) パラメータで時間分割多重化を選択した場合に、8 を超える**分解能**で正しい CRC シーケンスを生成するには、データ レートの 4 倍のクロック信号が必要となります。

配置

CRC は、UDB アレイ全体に配置され、すべての配置情報は、*cyfitter.h* ファイルを通して API に提供されます。

リソース

シングル サイクル実装

リソース	リソースのタイプ			API メモリ (バイト数)		ピン (外部入出力ごと)
	データパス セル	PLD	制御/Count 7 セル	Flash	RAM	
1〜8 ビット分解能	1	1	1	256	6	4
9〜16 ビット分解能	2	1	1	317	9	4
7〜24 ビット分解能	3	1	1	436	15	4
25〜32 ビット分解能	4	1	1	447	15	4

時間分割多重化の実装

リソース	リソースのタイプ			API メモリ (バイト数)		ピン (外部入出力ごと)
	データパス セル	PLD	制御/Count 7 セル	Flash	RAM	
9〜16 ビット分解能	1	3	1	483	13	4
7〜24 ビット分解能	2	3	1	873	23	4
25〜32 ビット分解能	2	3	1	1097	23	4
33〜40 ビット分解能	3	3	1	1345	43	4
41〜48 ビット分解能	3	3	1	1509	43	4

49〜56 ビット分解能	4	3	1	1742	43	4
57〜64 ビット分解能	4	3	1	1956	43	4

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 インターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。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。その次のセクションでは、各関数について詳しく説明します。

デフォルトで、PSoC Creator は、インスタンス名「CRC_1」を、特定の設計における最初のコンポーネント インスタンスに割り当てます。インスタンス名は、識別子の構文ルールに従った固有の値に変更できます。インスタンス名は、すべてのグローバル関数名、変数、定数記号の接頭辞になります。分かりやすいよう、次の表では、インスタンス名「CRC」を使用しています。

関数	説明
void CRC_Start(void)	初期値で、シードおよび多項レジスタを初期化します。CRC の計算は、入力クロックの立ち上がりエッジで開始されます。
void CRC_Stop(void)	CRC の計算を停止します。
void CRC_Wakeup(void)	CRC 設定を復元し、入力クロックの立ち上がりエッジで CRC の計算を開始します。
void CRC_Sleep(void)	CRC の計算を停止し、CRC 設定を保存します。
void CRC_Init(void)	初期値で、シードおよび多項レジスタを初期化します。
void CRC_Enable(void)	入力クロックの立ち上がりエッジで CRC の計算を開始します。
void CRC_SaveConfig(void)	シードおよび多項レジスタを保存します。
void CRC_RestoreConfig(void)	シードおよび多項レジスタを復元します。
void CRC_WriteSeed(uint8/16/32 seed)	シード値を書き込みます。
void CRC_riteSeedUpper(uint32 seed)	シード値の上位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。
void CRC_WriteSeedLower(uint32 seed)	シード値の下位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。
uint8/16/32 CRC_ReadCRC(void)	CRC 値を読み取ります。
uint32 CRC_ReadCRCUpper(void)	CRC 値の上位半分を読み込みます。33〜64 ビットの CRC でのみ生成されます。

関数	説明
uint32 CRC_ReadCRCLower(void)	CRC 値の下位半分を読み込みます。33〜64 ビットの CRC でのみ生成されます。
void CRC_WritePolynomial(uint8/16/32 polynomial)	CRC 多項値を書き込みます。
void CRC_WritePolynomialUpper(uint32 polynomial)	CRC 多項値の上位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。
void CRC_WritePolynomialLower(uint32 polynomial)	CRC 多項値の下位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。
uint8/16/32 CRC_ReadPolynomial(void)	CRC 多項値を読み取ります。
uint32 CRC_ReadPolynomialUpper(void)	CRC 多項値の上位半分を読み取ります。33〜64 ビットの CRC でのみ生成されます。
uint32 CRC_ReadPolynomialLower(void)	CRC 多項値の下位半分を読み取ります。33〜64 ビットの CRC でのみ生成されます。

グローバル変数

変数	説明
CRC_initVar	CRC が初期化されたかどうかを示します。変数は、0 に初期化され、最初に CRC_Start() が呼び出されると 1 に設定されます。これで、CRC_Start() ルーチンを最初に呼び出した後で、再初期化を行うことなく、コンポーネントを再起動できます。 コンポーネントの再初期化が必要な場合は、CRC_Int() 関数を CRC_Start() または CRC_Enable() 関数の前に呼び出すことができます。

void CRC_Start(void)

説明： 初期値で、シードおよび多項レジスタを初期化します。CRC の計算は、入力クロックの立ち上がりエッジで開始されます。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_Stop(void)

説明： CRC の計算を停止します。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_Stop(void)

説明： CRC の計算を停止し、CRC 設定を保存します。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_Stop(void)

説明： CRC 設定を復元し、入力クロックの立ち上がりエッジで CRC の計算を開始します。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_Start(void)

説明： 初期値で、シードおよび多項レジスタを初期化します。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_Sleep(void)

説明： 入力クロックの立ち上がりエッジで CRC の計算を開始します。

パラメータ： なし

戻り値： なし

副作用： なし

void LCD_Char_SaveConfig(void)

説明： 初期のシードおよび多項レジスタを保存します。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_SaveConfig(void)

説明： 初期のシードおよび多項レジスタを復元します。

パラメータ： なし

戻り値： なし

副作用： なし

void CRC_WriteSeed(uint8/16/32 seed)

説明： シード値を書き込みます。

パラメータ： seed: uint8/16/32 – シード値。

戻り値： なし

副作用： シード値は、マスク = 2 分解能 - 1 に準じてカットされます。
例えば、CRC 分解能が 14 ビットの場合は、マスク値が次のようになります。マスク = $2^{14} – 1$ = 0x3FFFu。
シード値 = 0xFFFFu はカットされ、シードおよびマスク = 0xFFFFu & 0x3FFFu = 0x3FFFu となります。

void CRC_WriteSeedUpper(uint32 seed)

説明： シード値の上位半分を書き込みます。33～64 ビットの CRC でのみ生成されます。

パラメータ： seed: uint32 – シード値の上位半分。

戻り値： なし

副作用： シード値の上位半分は、マスク = 2 分解能 - 1 に準じてカットされます。
例えば、CRC 分解能が 35 ビットの場合は、マスク値が次のようになります。
$2^{(35 – 32)} – 1 = 2^{\wedge 3} – 1$ = 0x0000 0007u。
シード値の上位半分 = 0x0000 00FFu はカットされ、
シードの上位半分 & マスク = 0x0000 00FFu & 0x0000 0007u = 0x0000 0007u

void CRC_WriteSeedLower(uint32 seed)

説明： シード値の下位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。

パラメータ： seed: uint32 – シード値の下位半分。

戻り値： なし

副作用： なし

uint8 ControlReg_Read(void)

説明： CRC 値を読み取ります。

パラメータ： なし

戻り値： uint8/16/32: CRC 値を返します。

副作用： なし

uint32 CRC_ReadCRCUpper(void)

説明： CRC 値の上位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。

パラメータ： なし

戻り値： uint32: CRC 値の上位半分を返します。

副作用： なし

uint32 CRC_ReadCRCLower(void)

説明： CRC 値の下位半分を読み込みます。33〜64 ビットの CRC でのみ生成されます。

パラメータ： なし

戻り値： uint32: CRC 値の下位半分を返します。

副作用： なし

void CRC_WriteSeed(uint8/16/32 seed)

説明： CRC 多項値を書き込みます。

パラメータ： polynomial: uint8/16/32 – CRC 多項式。

戻り値： なし

副作用： 多項値は、マスク = $2^{\text{分解能} - 32} - 1$ に準じてカットされます。例えば、CRC 分解能が 14 ビットの場合は、マスク値が次のようになります。マスク = $2^{14} – 1$ = 0x3FFFu。
多項値 = 0xFFFFu はカットされ、
多項式 & マスク = 0xFFFFu & 0x3FFFu = 0x3FFFu

void CRC_WritePolynomialUpper(uint32 polynomial)

説明： CRC 多項値の上位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。

パラメータ： polynomial: uint32 – CRC 多項値の上位半分。

戻り値： なし

副作用： 多項値の上位半分は、マスク = $2^{(分解能 - 32)} - 1$ に準じてカットされます。例えば、CRC 分解能が 35 ビットの場合は、マスク値が次のようになります。
$2^{(35 - 32)} - 1 = 2^3 - 1$ = 0x0000 0007u。
多項値の上位半分 = 0x0000 00FFu はカットされ、
多項式の上位 & マスク = 0x0000 00FFu & 0x0000 0007u = 0x0000 0007u

void CRC_WritePolynomialLower(uint32 polynomial)

説明： CRC 多項値の下位半分を書き込みます。33〜64 ビットの CRC でのみ生成されます。

パラメータ： polynomial: uint32 – CRC 多項値の下位半分。

戻り値： なし

副作用： なし

uint8/16/32 CRC_ReadCRC(void)

説明： CRC 多項値を読み取ります。

パラメータ： なし

戻り値： int8/16/32: CRC 多項値を返します。

副作用： なし

uint8/16/32 CRC_ReadPolynomial(void)

説明： CRC 多項値の上位半分を読み取ります。33〜64 ビットの CRC でのみ生成されます。

パラメータ： なし

戻り値： uint32: CRC 多項値の上位半分を返します。

副作用： なし

uint32 CRC_ReadPolynomialUpper(void)

説明： CRC 多項値の下位半分を読み取ります。33〜64 ビットの CRC でのみ生成されます。

パラメータ： なし

戻り値： uint32: CRC 多項値の下位半分を返します。

副作用： なし

ファームウェア サンプルソースコード

PSoC Creator では、[Find Example Project (サンプル プロジェクトの検索)] ダイアログに回路図とサンプル コードを含む数々のサンプル プロジェクトが提供されています。コンポーネント固有の例については、[Component Catalog (コンポーネント カタログ)] または回路図のコンポーネント インスタンスからダイアログを開きます。一般的なサンプルを見るには、開始ページまたは **[File (ファイル)]** メニューからダイアログを開いてください。必要に応じてダイアログにある [Fileter Options (フィルタのオプション)] 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 ヘルプの「Find Example Project (例プロジェクトを検索)」を参照してください。

機能説明

CRC は、リニア フィードバック シフト レジスタ (LFSR) として実装されます。シフト レジスタは、LFSR 関数を計算します。多項レジスタは、LFSR 多項式を定義する多項値を保持し、シード レジスタは、開始データの初期化を有効にします。

このコンポーネントでは、シードおよび多項レジスタを開始する前に初期化する必要があります。

N ビット LFSR 結果の計算は、X^0=1 となる X0 項 が最後の項となる N+1 項の多項式によって指定されます。例えば、広く利用されている CRC–CCITT 16 ビットの多項式は、$X^{16}+X^{12}+X^5+1$ です。CRC アルゴリズムは、X^0 項が存在すると想定するため、N ビット結果の多項式は、N+1 ビットの仕様ではなく、N ビットによって表現できます。

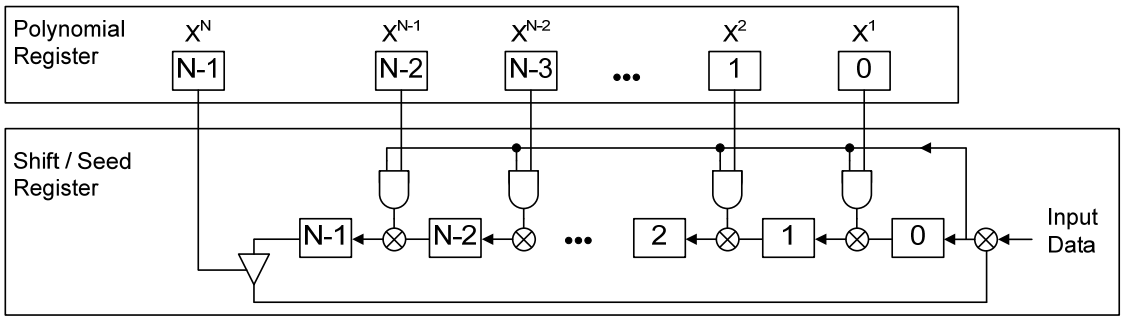
多項式の仕様を指定するには、各項の存在を 1 で表す、完全な多項式に該当する N+1 ビットのバイナリ数を書き込みます。CRC–CCITT 多項値は、10001000000100001b のようになります。次に、一番右のビット (X^0 項) をドロップし、CRC 多項値を取得します。CRC–CCITT 例を実装するため、多項レジスタが 8810h の値とともにロードされます。

入力クロックの立ち上がりエッジで、各ビットがシフトされます。これは、指定された CRC アルゴリズムを計算するシフト レジスタを通し、入力データ ストリームの MSB から始まります。入力データの各バイトで CRC を計算するには、8 つのクロックが必要です。

初期シード値が失われることに注意してください。シード値は、シフト レジスタをデータセットにつき一度だけ初期化するためにのみ使用されるため、初期シード値が失われることで悪影響が出ることはありません。

ブロック図と設定

使用されるデータ パスと、どのようにしてレジスタがデータ パスの中で使用されるかの情報をここに追加します。また、レジスタの書き込みによって起こる結果についても説明します。



タイミング図

図 1. 時間分割多重化の実装モード

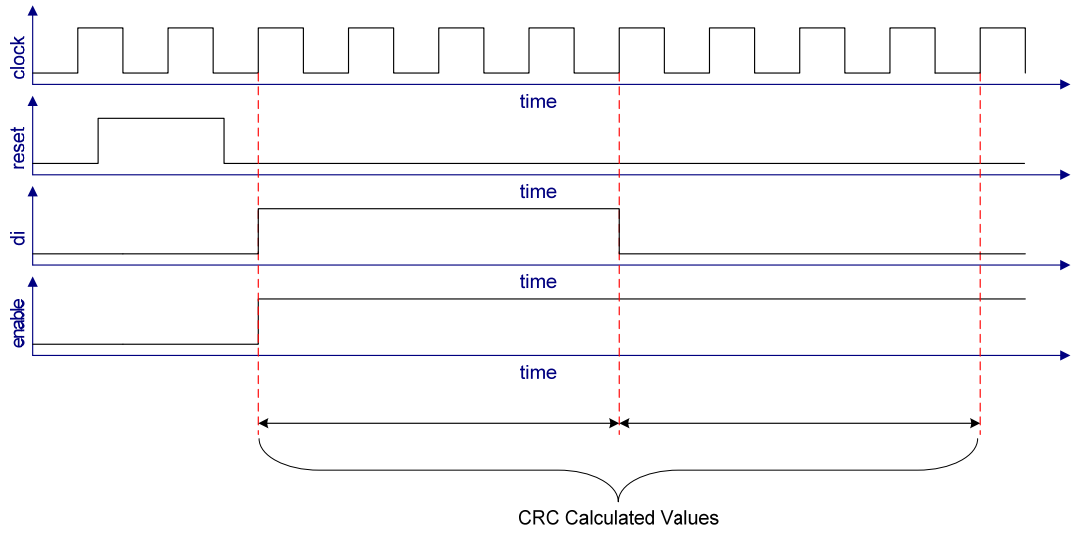
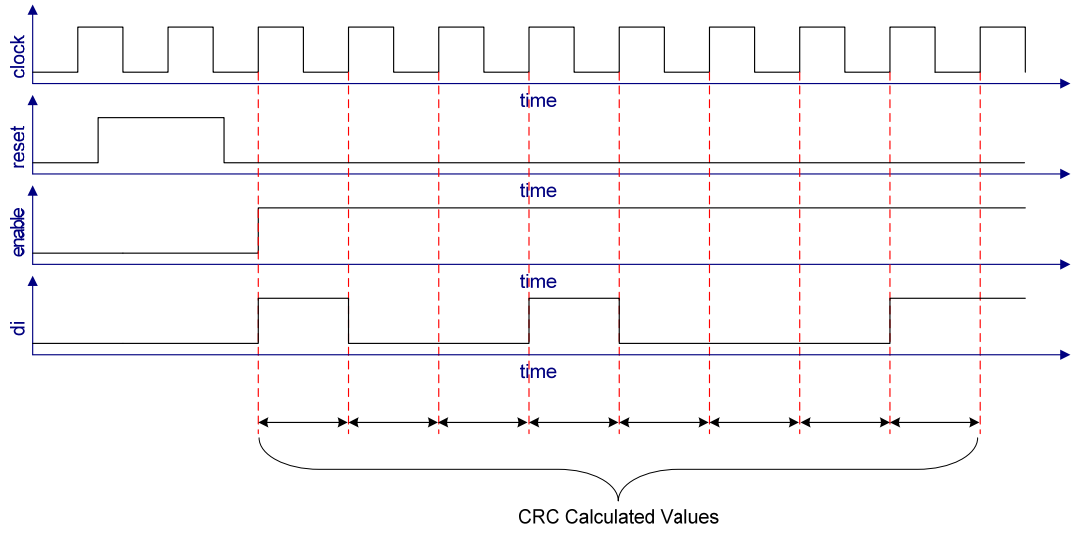


図 2. シングル サイクル実装モード



DC 電気的特性と AC 電気的特性

以下の値は、期待されるパフォーマンスを示しており、初期特性データを基にしています。

「公称ルーティングでの最大」タイミング特性

パラメータ	説明	設定[1]	最小値	標準値	最大値	単位
f_{clock}	コンポーネントのクロック周波数 [2]	設定 1			45	MHz
		設定 2			30	MHz
		設定 3			41	MHz
		設定 4			24	MHz
		設定 5			35	MHz
		設定 6			21	MHz
t_{clockH}	入力クロック HIGH 時間 [3]	該当なし		0.5		$1/f_{clock}$
t_{clockL}	入力クロック LOW 時間	該当なし		0.5		$1/f_{clock}$
Inputs (入力)						
t_{PD_ps}	入力パスの遅延、同期するピン [4]	1			STA [5]	ns
t_{PD_ps}	入力パスの遅延、同期するピン [6]	2			8.5	ns

[1] 設定:

設定 1:
分解能: 8 ビット
実装: シングル サイクル

設定 2:
分解能: 16 ビット
実装: シングル サイクル

設定 3:
分解能: 16 ビット
実装: 時間分割多重化

設定 4:
分解能: 32 ビット
実装: シングル サイクル

設定 5:
分解能: 32 ビット
実装: 時間分割多重化

設定 6:
分解能: 64 ビット
実装: 時間分割多重化

[2] 時間分割多重化の実装が選択された場合は、コンポーネント クロック周波数が、データ レートの 4 倍である必要があります。

[3] $t_{CY_clock} = 1/f_{clock}$ - 1 回の周期クロックの周期時間。

[4] t_{PD_ps} は、後述する静的タイミングの結果にあります。ここに記載されている値は、多くの入力での STA 解析を基にした公称値です。

[5] t_{PD_ps} および t_{PD_si} は、ルート パス遅延です。ルーティングが動的であるため、これらの値は変わり、最大コンポーネント クロックおよび同期クロック周波数に直接影響を与えます。値は、静的タイミング解析結果で見つける必要があります。

パラメータ	説明	設定[1]	最小値	標準値	最大値	単位
t_{PD_si}	入力パスの遅延への同期出力 (ルート)	1,2,3,4			STA	ns
t_{I_clk}	clockX および clock のアライメント	1,2,3,4	0		1	t_{CY_clock}
t_{PD_IE}	コンポーネント クロックへの入力パス遅延 (エッジ・センシティブ・入力)	1,2	t_{PD_ps} + t_{sync} + t_{PD_si}		t_{PD_ps} + t_{sync} + t_{PD_si} + t_{I_clk}	ns
t_{PD_IE}	コンポーネント クロックへの入力パス遅延 (エッジを感知する入力)	3,4	t_{sync} + t_{PD_si}		t_{sync} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	入力 HIGH 時間	1,2,3,4	t_{CY_clock} [7]			ns
t_{IL}	入力 LOW 時間	1,2,3,4	t_{CY_clock}			ns

設定 2 の [6] t_{PD_ps} は、デバイスのピンごとに定義される固定値です。ここに記載されている値は、デバイスで利用できるすべてのピンの公称値です。

時間分割多重化の実装が選択された場合は、[7] $t_{CY_clock} = 4^* [1/f_{clock}]$

「すべてのルーティングでの最大」タイミング特性

パラメータ	説明	設定 [1]	最小値	標準値	最大値 [2]	単位
f_{clock}	コンポーネントのクロック周波数 [3]	設定 1			23	MHz
		設定 2			15	MHz
		設定 3			21	MHz
		設定 4			12	MHz
		設定 5			18	MHz
		設定 6			11	MHz
t_{clockH}	入力クロック HIGH 時間 [4]	該当なし		0.5		$1/f_{clock}$
t_{clockL}	入力クロック LOW 時間	該当なし		0.5		$1/f_{clock}$
Inputs (入力)						
t_{PD_ps}	入力パスの遅延、同期するピン [5]	1			STA [6]	ns
t_{PD_ps}	入力パスの遅延、同期するピン [7]	2			8.5	ns

[1]設定:

設定 1:
分解能: 8 ビット
実装: シングル サイクル

設定 2:
分解能: 16 ビット
実装: シングル サイクル

設定 3:
分解能: 16 ビット
実装: 時間分割多重化

設定 4:
分解能: 32 ビット
実装: シングル サイクル

設定 5:
分解能: 32 ビット
実装: 時間分割多重化

設定 6:
分解能: 64 ビット
実装: 時間分割多重化

[2]「すべてのルーティング」の最大値は、<公称値>/2 で最近似の整数に切り上げ/切り下げられます。この値により、このコンポーネント周波数以下で実行される場合に、ユーザがタイミング条件を気にする必要がなくなります。

[3] 時間分割多重化の実装が選択された場合は、コンポーネント クロック周波数が、データ レートの 4 倍である必要があります。

[4] $t_{CY_clock} = 1/f_{clock}$ - 1 回の周期クロックの周期時間。

[5] t_{PD_ps} は、後述する静的タイミングの結果にあります。ここに記載されている値は、多くの入力での STA 解析を基にした公称値です。

[6] t_{PD_ps} および t_{PD_si} は、ルート パス遅延です。ルーティングが動的であるため、これらの値は変更され、最大コンポーネント クロックおよび同期クロック周波数に直接影響を与えます。値は、静的タイミング解析結果で見つける必要があります。

パラメータ	説明	設定 [1]	最小値	標準値	最大値 [2]	単位
t_{PD_si}	入力パスの遅延への同期出力 (ルート)	1,2,3,4			STA [6]	ns
t_{I_clk}	clockX および clock の整列	1,2,3,4	0		1	t_{CY_clock}
t_{PD_IE}	コンポーネント クロックへの入力パス遅延 (エッジを感知する入力)	1,2	t_{PD_ps} + t_{sync} + t_{PD_si}		t_{PD_ps} + t_{sync} + t_{PD_si} + t_{I_clk}	ns
t_{PD_IE}	コンポーネント クロックへの入力パス遅延 (エッジを感知する入力)	3,4	t_{sync} + t_{PD_si}		t_{sync} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	入力 HIGH 時間	1,2,3,4	t_{CY_clock} [8]			ns
t_{IL}	入力 LOW 時間	1,2,3,4	t_{CY_clock}			ns

設定 2 の [7] t_{PD_ps} は、デバイスのピンごとに定義される固定値です。ここに記載されている値は、デバイスで利用できるすべてのピンの公称値です。

時間分割多重化の実装が選択された場合は、[8] $t_{CY_clock} = 4^* [1/f_{clock}]$

特性データ用の STA 結果の使用方法

公称ルーティング最大値は、静的タイミング分析 (STA) を使って、複数のテストから収集されます。最大値は、以下の方法で、STA 結果を使って、それぞれの設計用に計算できます。

f_{clock}　最大コンポーネント クロック周波数は、名前の付いた外部クロックとして、クロックサマリのタイミング結果に含まれます。*_timing.html* ファイルからの、クロックの制限例を、以下の図に示します。

-Clock Summary

Clock	Actual Freq	Max Freq	Violation
BUS_CLK	24.000 MHz	118.683 MHz	
clock	24.000 MHz	56.967 MHz	

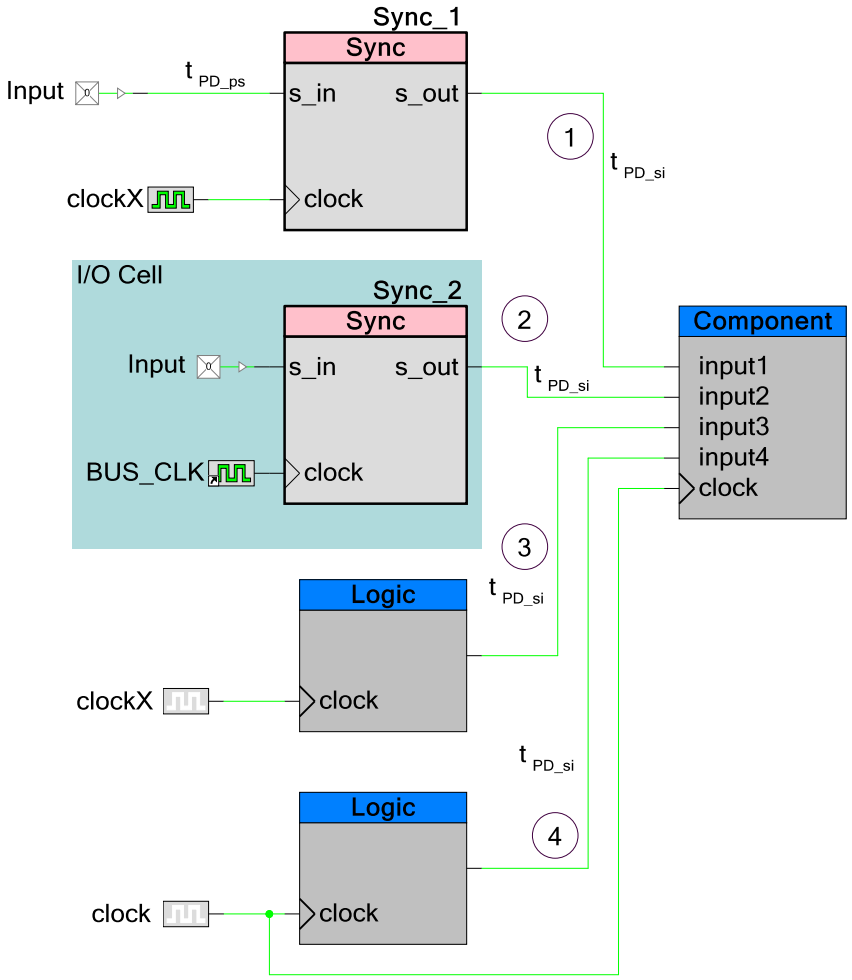
入力パス遅延とパルス幅

入力の機能を特性化する場合は、どのように構成しても、すべての入力は、図 3 に示されるように、4 つの可能な設定のいずれかになります。

すべての入力は同期させる必要があります。同期メカニズムは、コンポーネントへの入力によって異なります。システムがどのように動作するかを完全に理解するには、各入力に対して、どの入力設定をしたか、およびシステムのクロック設定を理解する必要があります。ここでは、

状態タイミング解析 (STA) 結果を使用して、システムの特性を決定する方法について説明します。

図 3. コンポーネント タイミング仕様に対する入力設定



コンフィグレーション	コンポーネント クロック	シンクロナイザ クロック (周波数)	図
1	master_clock	master_clock	図 8
1	clock (クロック)	master_clock	図 6
1	clock (クロック)	clockX = clock [1]	図 4
1	clock (クロック)	clockX > clock	図 5
1	clock (クロック)	clockX < clock	図 7

[1] クロック周波数は同じですが、立ち上がりエッジのアライメントは保証されません。

コンフィグレーション	コンポーネント クロック	シンクロナイザ クロック (周波数)	図
2	master_clock	master_clock	図 8
2	clock (クロック)	master_clock	図 6
3	master_clock	master_clock	図 13
3	clock (クロック)	master_clock	図 11
3	clock (クロック)	clockX = clock	図 9
3	clock (クロック)	clockX > clock	図 10
3	clock (クロック)	clockX < clock	図 12
4	master_clock	master_clock	図 13
4	clock (クロック)	clock (クロック)	図 9

1. 入力は、デバイスのピンによって駆動され、「sync」コンポーネントで内部的に同期されます。このコンポーネントは、コンポーネントが使用するクロックではない別の内部クロックを使用します。

   この方法で設定される入力を特性化する場合、clockX は、コンポーネント クロックより高速、低速、または同じになります。また、図 4、図 5、図 7、図 8 に示されているように、特性化パラメータを生成する、master_clock にも等しくなる場合があります。

2. 入力は、デバイスピンによって駆動され、master_clock を使ってピンで同期化されます。

   この方法で設定される入力を特性化する場合、master_clock は、コンポーネント クロックより高速、または同じになります (低速にはならない)。これは、図 および図 658 に示されているように、特性化パラメータを生成します。

図 4. 入力設定 1 と 2、同期クロック周波数 = コンポーネント クロック周波数 (clock および clockX のエッジアライメントは保証されません。)

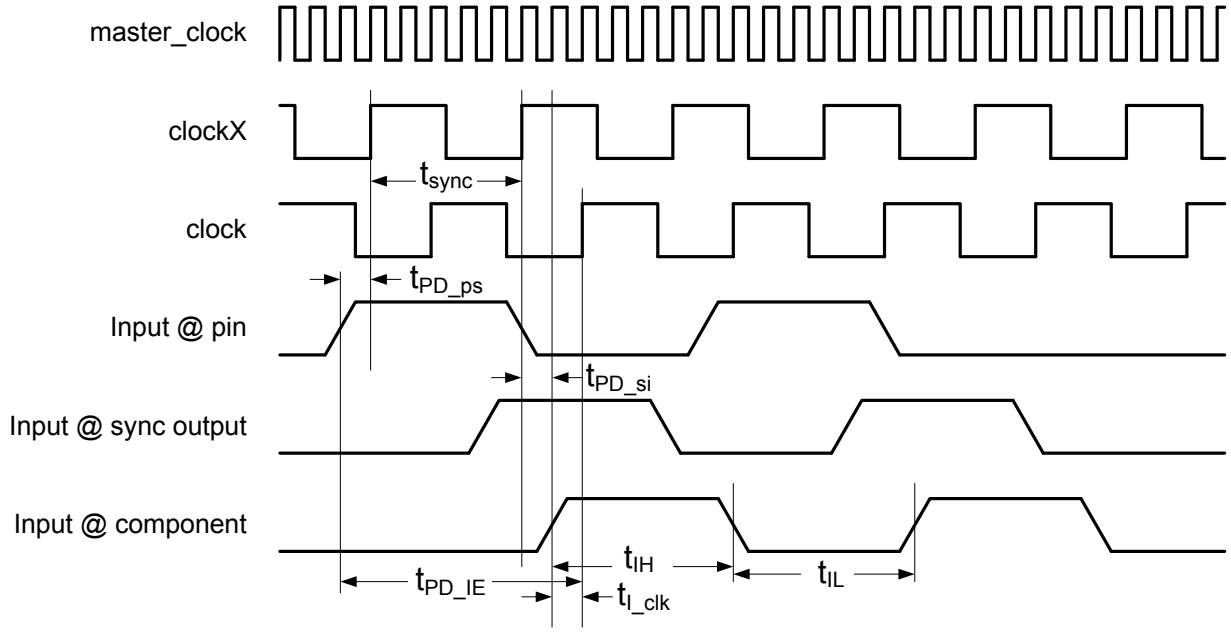


図 5. 入力設定 1 と 2、同期 クロック周波数 > コンポーネント クロック周波数

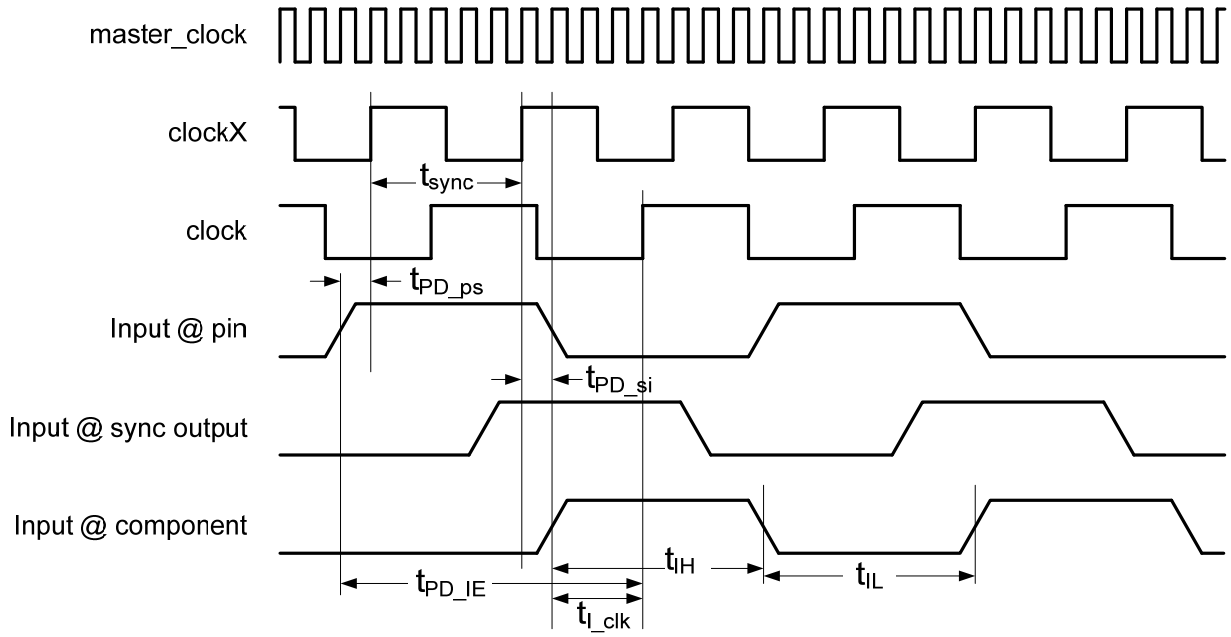


図 6. 入力設定 1 と 2、[同期 クロック周波数 == master_clock] > コンポーネント クロック周波数

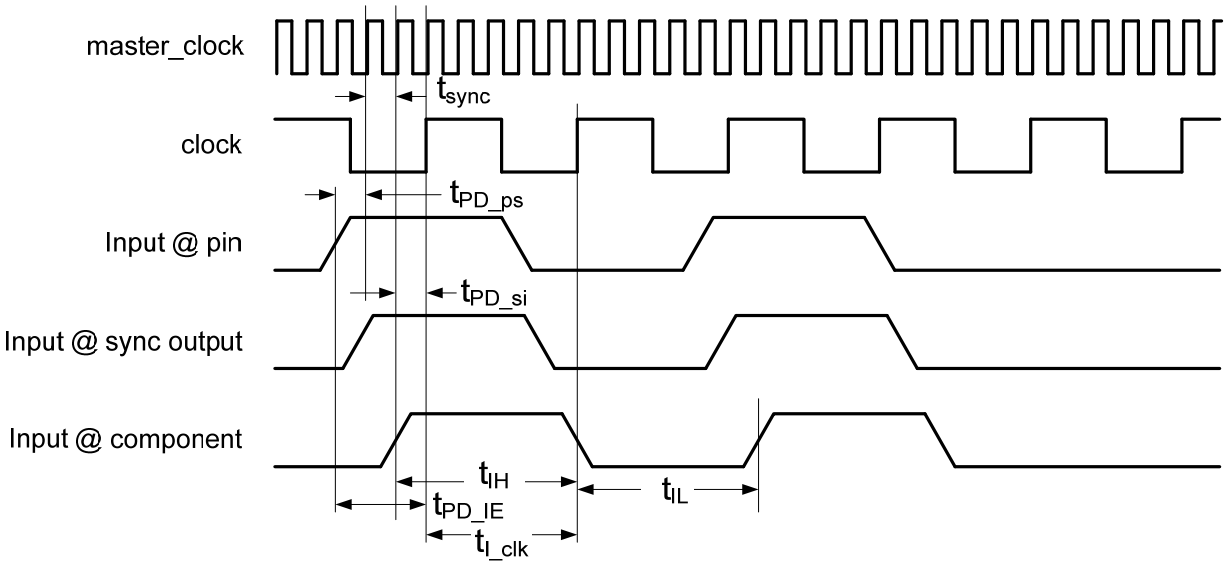


図 7. 入力設定 1、同期 クロック周波数 < コンポーネント クロック周波数

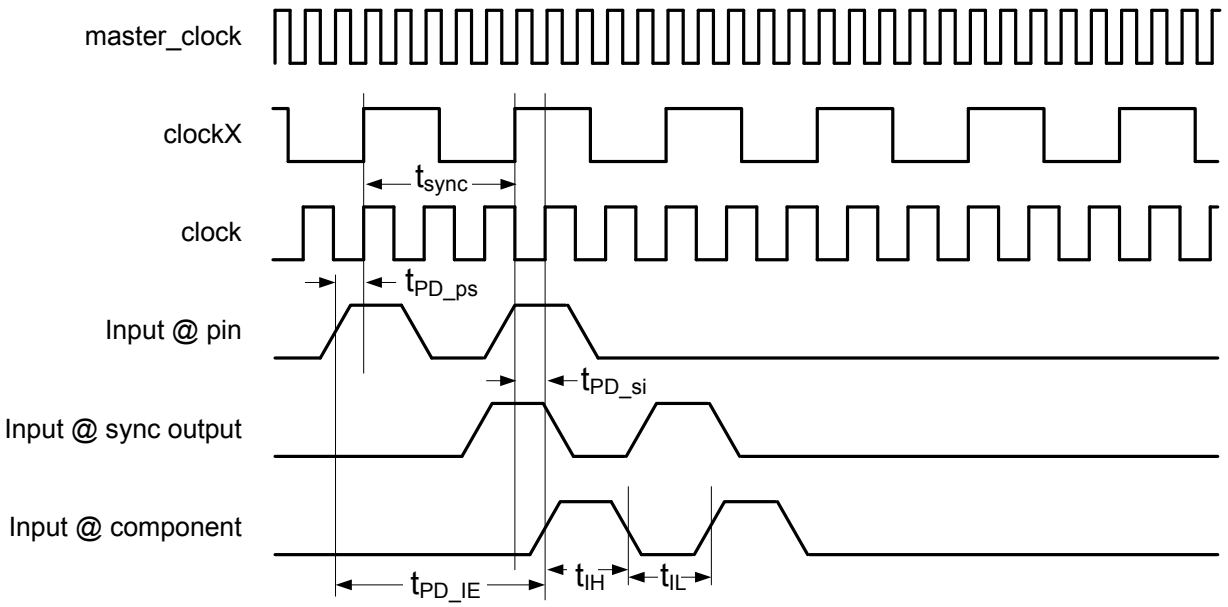
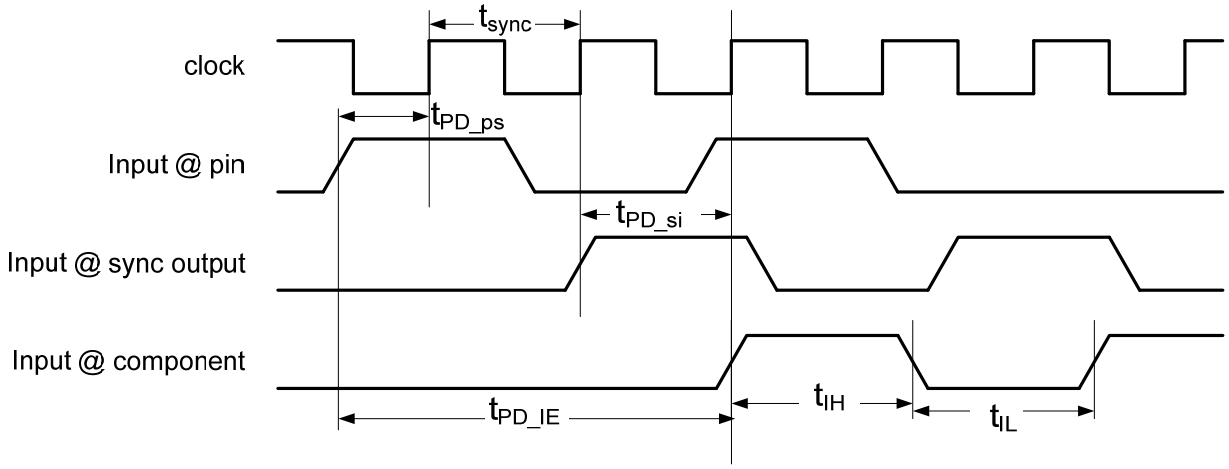


図 8. 入力設定 1 と 2、同期 クロック = コンポーネント クロック = master_clock



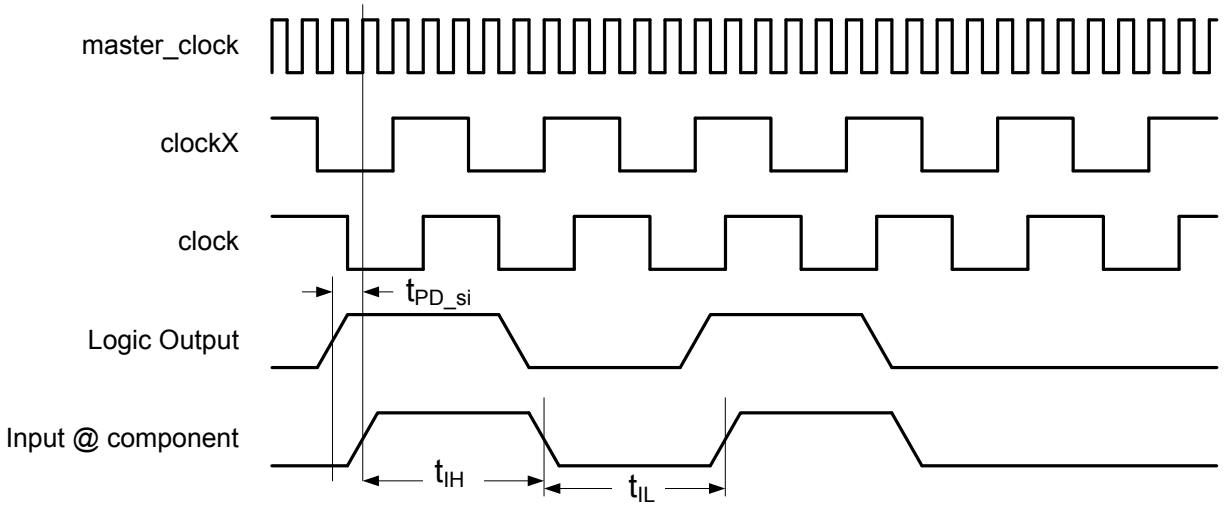
3. この入力は、コンポーネントが使用するクロックではない別の内部クロックを基にして同期され、PSoC への内部ロジックによって駆動されます。(全内部クロックは、 master_clock から生成されます。）

   この方法で設定された入力を特性化する場合、同期クロックは、図 9、図 10、図 12 に示される特性化パラメータを生成するコンポーネントのクロックより高速、低速、または等しくなります。

4. この入力は、コンポーネントが使用する同じクロックを基にして同期化され、PSoC への内部ロジックによって駆動されます。

   この方法で設定された入力を特性化する場合、同期クロックは、図 13 に示される特性化パラメータを生成する、コンポーネントのクロックに等しくなります。

図 9. 入力設定 3、同期 クロック周波数 = コンポーネントのクロック周波数 (clock および clockX のエッジアライメントは保証されません。)



この図は、クロックに対する静的タイミング解析を示しています。デジタル クロック ドメインの全クロックは、master_clock に同期されます。ただし、同じ周波数を持つ 2 つのクロックで、立ち上がりエッジがアラインしない可能性があります。このため、静的タイミング解析ツールは、クロックが同期されているのがどちらのエッジかを知ることはできません。また、少なくとも 1 回の master_clock 周期があると想定する必要があります。これは、t_{PD_si} がシステムの master_clock に制限付きの影響を与えることを意味します。このパス遅延が長すぎると、master_clock のセットアップ時間違反が発生します。システムの同期クロックを変更するか、master_clock をより低い周波数で実行する必要があります。

図 10. 入力設定 3、同期 クロック周波数 > コンポーネント クロック周波数

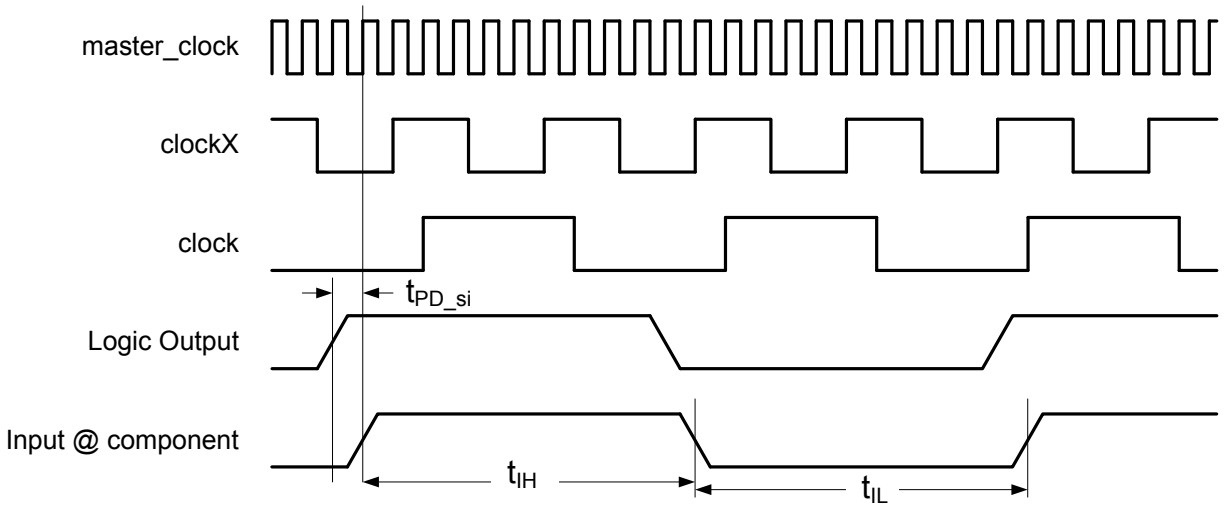


図 9 に示されているのとほぼ同じ方法で、すべてのクロックは、master_clock から生成されます。STA は、この構成で、1 回の master_clock 周期において、master_clock に対する t_{PD_si} の制限を示します。このパス遅延が長すぎると、master_clock のセットアップ時間違反が発生します。システムの同期クロックを変更するか、master_clock をより低い周波数で実行する必要があります。

図 11. 入力設定 3、シンクロナイザ クロック周波数 = master_clock > コンポーネント クロック周波数

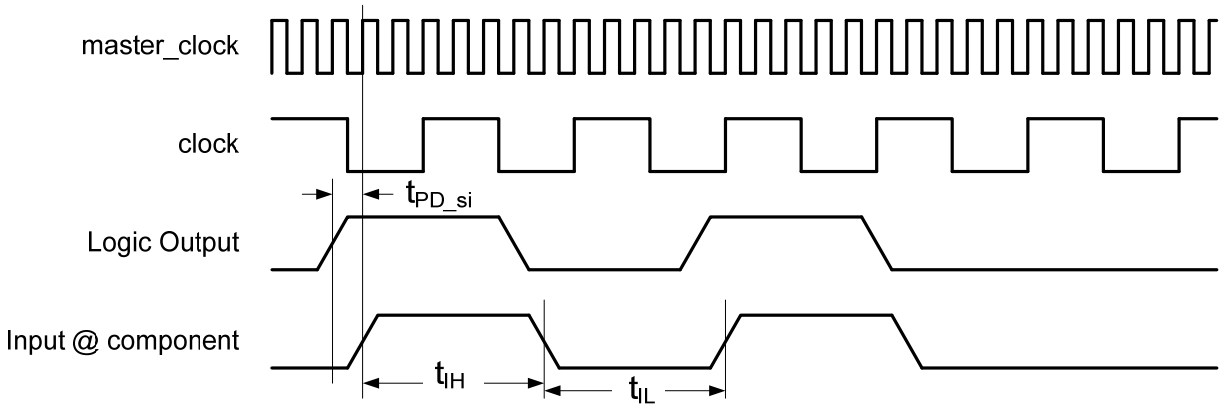


図 12. 入力設定 3、シンクロナイザ クロック周波数 < コンポーネント クロック周波数

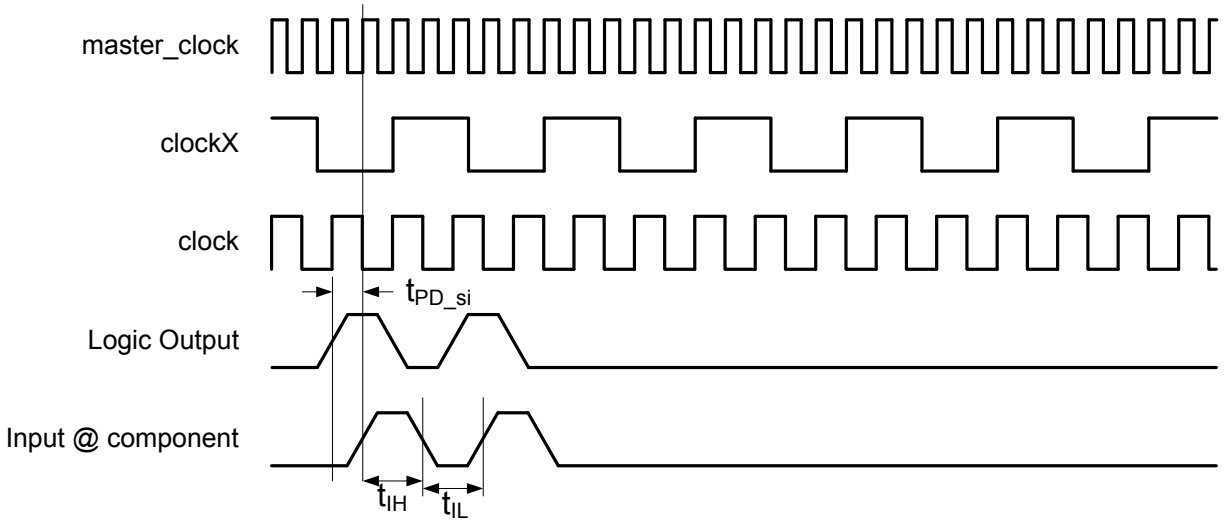
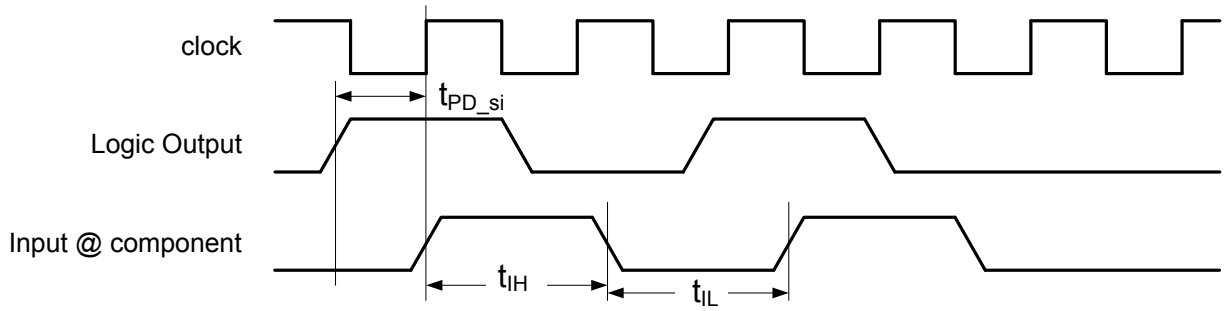


図 9 に示されているのとほぼ同じ方法で、すべてのクロックは、master_clock から生成されます。SAT は、1 回の master_clock 周期において、master_clock に対する t_{PD_si} の制限を示します。このパス遅延が長すぎると、master_clock のセットアップ時間違反が発生します。システムの同期クロックを変更するか、master_clock をより低い周波数で実行する必要があります。

図 13. 入力設定 4 のみ、 シンクロナイザー クロック = コンポーネント クロック



ここに記載されているすべての図で、実装を理解するために使用するほとんどの重要なパラメータは、f_{clock} と t_{PD_IE} です。t_{PD_IE} は、t_{PD_ps} および t_{sync} (設定 1 と 2 のみ)、t_{PD_si}、 および t_{I_Clk} によって定義されます。重要な点は、t_{PD_si} がコンポーネントの最大クロック周波数を定義することです。t_{I_Clk} は、STA 結果から取得されませんが、 t_{PD_IE} がレジスタされる時期を表すために使用されます。これは、シンクロナイザとコンポーネントのクロック間のルートを終えた後に残るマージンです。

t_{PD_ps} および t_{PD_si} は、STA 結果に含まれます。

t_{PD_ps} を見つけるには、*_timing.html* ファイルで定義されている入力セットアップ時間を見てください。この入力のファンアウトは 1 以上であるため、これらのパスの最大を評価する必要があります。

-Setup times

-Setup times to clock BUS_CLK

Start	Register	Clock	Delay (ns)
input1(0):iocell.pad_in	input1(0):iocell.ind	BUS_CLK	16.500

t_{PD_si} は、レジスタ-レジスタ間で定義されます。*_timing.html* ファイルを使用するため、ネットの名前を知る必要があります。このパスのファンアウトは 1 以上であると考え、これらのパスの最大を評価する必要があります。

-Register-to-register times

-Destination clock clock

Destination clock clock (Actual freq: 24.000 MHz)

+Source clock clock

-Source clock clock_1

Source clock clock_1 (Actual freq: 24.000 MHz)
Affected clock: BUS_CLK (Actual freq: 24.000 MHz)

Start	End	Period (ns)	Max Freq	Frequency	Violation
\Sync_1:genblk1[0]:INST\:synccell.syncq	\PWM_1:PWMUDB:runmode_enable\:macrocell.mc_d	7.843	127.508 MHz	24.000 MHz	

出力パス遅延

出力のパス遅延を特性化する場合、STA 結果のどこでデータを見つけることができるかがわかるよう、出力がどこに行くかを考慮する必要があります。このコンポーネントでは、すべての出力がコンポーネント クロックに同期化されます。出力は、次のいずれかのカテゴリになります。出力は、デバイス内の別のコンポーネントまたはデバイス外をドライブするピンに到達します。前者の場合に、上記のロジック-入力の説明に示されているレジスタ-レジスタ間を見る必要があります (ソース クロックはコンポーネント クロックです)。2 つ目の場合、*_timing.html* STA 結果のクロック-出力時間を見る必要があります。

コンポーネントの変更

ここでは、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
2.0.a	データシートに特性データを追加	
	データシートのマイナーな編集と更新	

バージョン	変更の説明	変更の理由 / 影響
2.0	PSoC ES3 シリコンでのサポートが追加されました。変更は以下のとおりです。 • 時間分割多重化の実装モードに 4x クロックを追加 • 1x クロックでのシングル サイクル実装が 1〜32 ビットで利用可能になりました。 • 4x クロックでの時間分割多重化の実装モードが 9〜64 ビットで利用可能になりました。 • 非同期入力信号のリセットが追加されました。 • 同期入力信号のイネーブルが追加されました。 • [Implementation (実装)] (時間分割多重化、シングル サイクル) パラメータの [Configure (設定)] ダイアログに新しい [Advanced (詳細)] ページが追加されました。	PSoC 3 ES3 デバイスをサポートする新しい要求により、CRC コンポーネントの新しい 2.0 バージョンが作成されました。
	CRC_Sleep()/CRC_Wakeup() および CRC_Init()/CRC_Enable() API が追加されました。	低電力モードをサポートし、ほとんどのコンポーネントの初期化と有効化を別に制御する共通のインターフェースを提供します。
	関数 CRC_WriteSeed() および CRC_WriteSeedUpper() が更新されました。	マスク パラメータは、シード値をカットして、書き込み中に CRC の分解能を定義するために使用されました。
	Resolution (分解能) パラメータへの検証が追加されました。	CRC の分解能は 1〜64 ビットです。検証が、入力値を制限するために追加されました。
	リセット DFF のトリガが、次の多項式書き込み関数に追加されました。CRC_WritePolynomial()、CRC_WritePolynomialUpper()、CRC_WritePolynomialLower()。	CRC の計算を開始する前に、DFF トリガを正しい状態に設定する必要があります (多項式の最上位ビットは常に 1)。この条件を満たすため、シードまたは多項レジスタへの書き込みは、DFF トリガをリセットします。
	次のパラメータで、Expression View が見れるよう、[Configure (設定)] ダイアログが更新されました。PolyValueLower、PolyValueUpper、SeedValueLower、SeedValueLower	Expression View は、記号パラメータに直接アクセスするために使用されます。このビューを使うと、必要に応じて、コンポーネント パラメータと外部パラメータを接続できるようになります。
	[Configure (設定)] ダイアログが更新され、さまざまなパラメータにエラー アイコンが追加されました。	テキスト ボックスに不正な値を入力すると、問題を説明するツールのヒントとともに、エラー アイコンが表示されます。これにより、別途エラー メッセージを表示するより、使いやすくなりました。
1.20	API 生成の方法が変更されました。バージョン 1.10 では、API はカスタマイザの設定から生成されていました。1.20 では、API は、他のほとんどのコンポーネントと同様に、.c および .h ファイルによって提供されます。	この変更により、生成された API を表示して、変更を加えることができるようになりました。また、次のビルドで上書きされることはありません。

バージョン	変更の説明	変更の理由 / 影響
	シードおよび多項パラメータが、16 進法形式に変更されました。	変更は、サイプレスの標準に準拠するためのものです。

© Cypress Semiconductor Corporation, 2011. 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、サイプレス セミコンダクタ社の登録商標であり、PSoC Creator™ およびプログラマブル System-on-Chip™ は、サイプレス セミコンダクタ社の商標です。本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。