サイプレスはインフィニオン テクノロジーズになりました

この表紙に続く文書には「サイプレス」と表記されていますが、これは同社が最初にこの製品を開発したからです。新規および既存のお客様いずれに対しても、引き続きインフィニオンがラインアップの一部として当該製品をご提供いたします。

文書の内容の継続性

下記製品がインフィニオンの製品ラインアップの一部として提供されたとしても、それを理由としてこの文書に変更が加わることはありません。今後も適宜改訂は行いますが、変更があった場合は文書の履歴ページでお知らせします。

注文時の部品番号の継続性

インフィニオンは既存の部品番号を引き続きサポートします。ご注文の際は、データシート記載の注文部品番号をこれまで通りご利用下さい。

PSoC 4000S ファミリ

PSoC® 4 アーキテクチャ テクニカル リファレンス マニュアル (TRM)

文書番号 : 002-15788 Rev. *A

January 24, 2020

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

Copyrights

概要

Section A: 概要 **10**
1. はじめに ... 11
2. 開発サポート ... 16
3. 文書の構成 ... 17

Section B: CPUシステム **21**
4. Cortex-M0+ CPU ... 22
5. 割り込み ... 27

Section C: システムリソース サブシステム (SRSS) **35**
6. I/Oシステム ... 37
7. クロック供給システム ... 57
8. 電源と電圧監視 ... 64
9. チップ動作モード ... 68
10. 消費電力モード ... 69
11. ウォッチドッグ タイマー ... 73
12. リセット システム ... 78
13. デバイス セキュリティ ... 81

Section D: デジタル システム **83**
14. シリアル通信ブロック (SCB) ... 84
15. タイマー、カウンター、およびパルス幅変調器 ... 128

Section E: アナログ システム **153**
16. 低消費電力コンパレータ ... 154
17. CapSense ... 160
18. LCDダイレクト ドライブ ... 161

Section F: プログラムおよびデバッグ **174**
19. プログラムおよびデバッグ インターフェース ... 175
20. 不揮発性メモリ プログラム ... 182

用語集 **198**

目次

Section A: 概要 **10**

改訂履歴 10

1. はじめに **11**

1.1 トップ レベル アーキテクチャ 11
1.2 特長....12
1.3 CPUシステム 12
1.3.1 プロセッサ 12
1.3.2 割り込みコントローラー 13
1.4 メモリ 13
1.4.1 フラッシュ 13
1.4.2 SRAM....13
1.5 システムワイド リソース 13
1.5.1 クロック システム 13
1.5.2 電源システム....13
1.5.3 GPIO....13
1.6 固定機能デジタル 13
1.6.1 タイマー／カウンター／PWMブロック 13
1.6.2 シリアル通信ブロック 13
1.7 アナログ システム....14
1.7.1 低消費電力コンパレータ 14
1.8 特殊機能ペリフェラル 14
1.8.1 LCDセグメント駆動....14
1.8.2 CapSense 14
1.9 プログラムおよびデバッグ....14
1.10 デバイス機能の要約....14

2. 開発サポート **16**

2.1 サポート....16
2.2 製品アップグレード 16
2.3 開発キット 16
2.4 アプリケーション ノート 16

3. 文書の構成 **17**

3.1 主要な節....17
3.2 本書の表記法 17
3.2.1 レジスタの表記法 17
3.2.2 数値の表記法 17
3.2.3 測定単位....18
3.2.4 略語....18

Section B: CPUシステム **21**

トップ レベル アーキテクチャ 21

4. Cortex-M0+ CPU 22
4.1 特長 22
4.2 ブロック図 23
4.3 動作原理 23
4.4 アドレス マップ 23
4.5 レジスタ 24
4.6 動作モード 25
4.7 命令セット 25
4.7.1 アドレス アライメント 26
4.7.2 メモリ エンディアン 26
4.8 Systickタイマー 26
4.9 デバッグ 26

5. 割り込み 27
5.1 特長 27
5.2 動作原理 27
5.3 割り込みと例外の動作 28
5.3.1 割り込み／例外の処理 28
5.3.2 レベルおよびパルス割り込み 28
5.3.3 例外ベクタ テーブル 29
5.4 例外ソース 29
5.4.1 リセット例外 29
5.4.2 マスク不可能割り込み (NMI) 例外 30
5.4.3 HardFault例外 30
5.4.4 スーパバイザ コール (SVCall) 例外 30
5.4.5 PendSV例外 30
5.4.6 SysTick例外 31
5.5 割り込みソース 31
5.6 例外の優先度 31
5.7 割り込みの有効と無効 32
5.8 例外状態 32
5.8.1 保留中の例外 33
5.9 例外のスタック使用量 33
5.10 割り込みと低消費電力モード 33
5.11 例外の初期化とコンフィギュレーション 34
5.12 レジスタ 34
5.13 関連文書 34

Section C: システムリソース サブシステム (SRSS) 35
トップ レベル アーキテクチャ 36

6. I/Oシステム 37
6.1 特長 37
6.2 GPIOインターフェースの概要 37
6.3 I/Oセルのアーキテクチャ 38
6.3.1 デジタル入力バッファ 40
6.3.2 デジタル出力ドライバー 40
6.4 高速I/Oマトリックス 42
6.5 スマートI/O 43
6.5.1 概要 43
6.5.2 ブロック コンポーネント 43
6.5.3 配線 51
6.5.4 動作 51

6.6 電源投入時のI/O状態 52
6.7 省電力モードでの動作 53
6.8 割り込み 53
6.9 ペリフェラルの接続 54
6.9.1 ファームウェアで制御されるGPIO 54
6.9.2 アナログI/O 54
6.9.3 LCD駆動 54
6.9.4 CapSense 55
6.9.5 シリアル通信ブロック (SCB) 55
6.9.6 タイマー／カウンター／パルス幅変調器 (TCPWM) ブロック 55
6.10 レジスタ 55

7. クロック供給システム 57

7.1 ブロック図 57
7.2 クロック ソース 58
7.2.1 内部主発振器 (IMO) 58
7.2.2 内部低速発振器 (ILO) 59
7.2.3 外部クロック (EXTCLK) 60
7.2.4 時計用水晶発振器 (WCO) 60
7.3 クロック分配 60
7.3.1 HFCLK入力の選択 60
7.3.2 LFCLK入力の選択 61
7.3.3 SYSCLKプリスケーラのコンフィギュレーション 61
7.3.4 ペリフェラル クロック分周器のコンフィギュレーション 61
7.4 低消費電力モード動作 63
7.5 レジスター覧 63

8. 電源と電圧監視 64

8.1 ブロック図 64
8.2 電源供給のシナリオ 65
8.2.1 単一1.8V～5.5V非安定化電源 65
8.2.2 1.71V～1.89V安定化電源の直接接続 65
8.3 動作原理 66
8.3.1 レギュレータの概要 66
8.4 電圧監視 66
8.4.1 パワーオンリセット (POR) 66
8.5 レジスター覧 67

9. チップ動作モード 68

9.1 ブート 68
9.2 ユーザー 68
9.3 特権 68
9.4 デバッグ 68

10. 消費電力モード 69

10.1 アクティブ モード 70
10.2 スリープ モード 70
10.3 ディープスリープ モード 70
10.4 消費電力モードの概要 71
10.5 低消費電力モードへの移行および終了 72
10.6 レジスター覧 72

11. ウォッチドッグ タイマー 73

11.1 特長 73

11.2 ブロック図 73
11.3 動作原理 73
11.3.1 WDTの有効化と無効化 74
11.3.2 WDT割り込みと低消費電力モード 75
11.3.3 WDTリセット モード 75
11.4 追加タイマー 75
11.4.1 WDT0とWDT1 75
11.4.2 WDT2 76
11.4.3 カスケード 76
11.5 レジスター覧 77

12. リセット システム 78

12.1 リセット ソース 78
12.1.1 パワーオン リセット 78
12.1.2 電圧低下リセット 78
12.1.3 ウォッチドッグ リセット 78
12.1.4 ソフトウェア初期化リセット 79
12.1.5 外部リセット 79
12.1.6 保護フォルト リセット 79
12.2 リセット ソースの識別 79
12.3 レジスター覧 80

13. デバイス セキュリティ 81

13.1 特長 81
13.2 動作原理 81
13.2.1 デバイス セキュリティ 81
13.2.2 フラッシュ セキュリティ 82

Section D: デジタル システム 83

トップ レベル アーキテクチャ 83

14. シリアル通信ブロック (SCB) 84

14.1 特長 84
14.2 シリアル ペリフェラル インターフェース (SPI) 84
14.2.1 特長 84
14.2.2 概要 85
14.2.3 SPI動作モード 86
14.2.4 スレーブへのクロック供給用のSPIマスターの使用 91
14.2.5 イージーSPIプロトコル 91
14.2.6 SPIレジスタ 93
14.2.7 SPI割り込み 94
14.2.8 SPIの有効化と初期化 94
14.2.9 内部と外部クロック供給SPI動作 96
14.3 UART 98
14.3.1 特長 98
14.3.2 概要 99
14.3.3 UARTの動作モード 99
14.3.4 UARTレジスタ 107
14.3.5 UART割り込み 107
14.3.6 UARTの有効化と初期化 108
14.4 インター インテグレーテッド回路 (I2C) 110
14.4.1 特長 110
14.4.2 概要 110
14.4.3 用語および定義 111

14.4.4 I2C動作モード ... 111
14.4.5 イージーI2C (EZI2C) プロトコル ... 113
14.4.6 I2Cレジスタ ... 114
14.4.7 I2C割り込み ... 116
14.4.8 I2Cの有効化および初期化 ... 116
14.4.9 I2Cにおける内部および外部クロック動作 ... 117
14.4.10 スリープから復帰 ... 119
14.4.11 マスター モード転送の例 ... 120
14.4.12 スレーブ モード転送の例 ... 122
14.4.13 EZスレーブ モード転送の例 ... 124
14.4.14 マルチマスター モード転送の例 ... 126

15. タイマー、カウンター、およびパルス幅変調器 128

15.1 特長 ... 128
15.2 ブロック図 ... 129
15.2.1 TCPWMブロックにおけるカウンターの有効／無効 ... 129
15.2.2 クロッキング ... 129
15.2.3 トリガー入力に基づいたイベント ... 130
15.2.4 出力信号 ... 132
15.2.5 電力モード ... 134
15.3 動作モード ... 134
15.3.1 タイマー モード ... 135
15.3.2 キャプチャ モード ... 138
15.3.3 直交デコーダー モード ... 141
15.3.4 パルス幅変調モード ... 144
15.3.5 デッドタイム付きパルス幅変調モード ... 148
15.3.6 疑似乱数パルス幅変調モード ... 150
15.4 TCPWMレジスタ ... 152

Section E: アナログ システム 153

トップ レベル アーキテクチャ ... 153

16. 低消費電力コンパレータ 154

16.1 特長 ... 154
16.2 ブロック図 ... 154
16.3 動作原理 ... 155
16.3.1 入力コンフィギュレーション ... 155
16.3.2 出力および割り込みコンフィギュレーション ... 156
16.3.3 消費電力モードおよび動作速度のコンフィギュレーション ... 157
16.3.4 ヒステリシス ... 157
16.3.5 低消費電力モードからの復帰 ... 158
16.3.6 コンパレーター クロック ... 158
16.3.7 オフセット調整 ... 158
16.4 レジスタ要約 ... 159

17. CapSense 160

18. LCDダイレクト ドライブ 161

18.1 特長 ... 161
18.2 LCDセグメント ドライブの概要 ... 161
18.2.1 駆動モード ... 162
18.2.2 ドライブ モードの推奨使用方法 ... 170
18.2.3 デジタル コントラスト制御 ... 170
18.3 ブロック図 ... 171

18.3.1 動作原理 ... 172
18.3.2 高速および低速のマスター信号発生器 ... 172
18.3.3 マルチプレクサおよびLCDピン回路 ... 172
18.3.4 ディスプレイ データ レジスタ ... 172
18.4 レジスター覧 ... 173

Section F: プログラムおよびデバッグ 174

トップ レベル アーキテクチャ ... 174

19. プログラムおよびデバッグ インターフェース 175

19.1 特長 ... 175
19.2 機能説明 ... 175
19.3 シリアル ワイヤ デバッグ (SWD) インターフェース ... 176
19.3.1 SWDタイミングの詳細 ... 177
19.3.2 ACK応答の詳細 ... 177
19.3.3 ターンアラウンド (Trn) 期間の詳細 ... 177
19.4 Cortex-M0+デバッグおよびアクセス ポート (DAP) ... 178
19.4.1 デバッグ ポート (DP) レジスタ ... 178
19.4.2 アクセス ポート (AP) レジスタ ... 178
19.5 PSoC 4デバイスのプログラム ... 179
19.5.1 SWDポートの開通 ... 179
19.5.2 SWDプログラム モードへの移行 ... 179
19.5.3 SWDプログラム ルーチンの実施 ... 179
19.6 PSoC 4 SWDデバッグ インターフェース ... 180
19.6.1 デバッグ制御およびコンフィギュレーション レジスタ ... 180
19.6.2 ブレークポイント ユニット (BPU) ... 180
19.6.3 データ ウォッチポイント (DWT) ... 180
19.6.4 PSoC 4デバイスのデバッグ ... 181
19.7 レジスタ ... 181

20. 不揮発性メモリ プログラム 182

20.1 特長 ... 182
20.2 機能説明 ... 182
20.3 システム コールの搭載 ... 183
20.4 ブロッキングと非ブロッキングのシステム コール ... 183
20.4.1 システム コールの実行 ... 183
20.5 システム コール ... 184
20.5.1 シリコンID ... 185
20.5.2 クロックの設定 ... 185
20.5.3 ロード フラッシュ バイト ... 186
20.5.4 列書き込み ... 187
20.5.5 列プログラム ... 188
20.5.6 全消去 ... 189
20.5.7 チェックサム ... 190
20.5.8 書き込み保護 ... 191
20.5.9 非ブロッキング列書き込み ... 192
20.5.10 非ブロッキング列プログラム ... 193
20.5.11 レジューム非ブロッキング ... 194
20.6 システム コール ステータス ... 195
20.7 非ブロッキング システム コール疑似コード ... 196

用語集 198

セクション A: 概要

このセクションは次の章を含みます :

- 11 ページのはじめに
- 16 ページの開発サポート
- 17 ページの文書の構成

改訂履歴

版	発行日	変更内容
**	10/17/2016	これは英語版 002-10129 Rev. *A を翻訳した日本語版 002-15788 Rev. ** です。
*A	01/24/2020	これは英語版 002-10129 Rev. *C を翻訳した日本語版 002-15788 Rev. *A です。

1. はじめに

PSoC® 4 は Arm® Cortex®-M0+ CPU を備えたプログラマブル組み込みシステム コントローラーです。PSoC 4000S は PSoC 4000 ファミリを改良した製品であり、PSoC 4 のより大きなメンバーと上位互換性があります。

PSoC 4 デバイス には以下の特長があります :

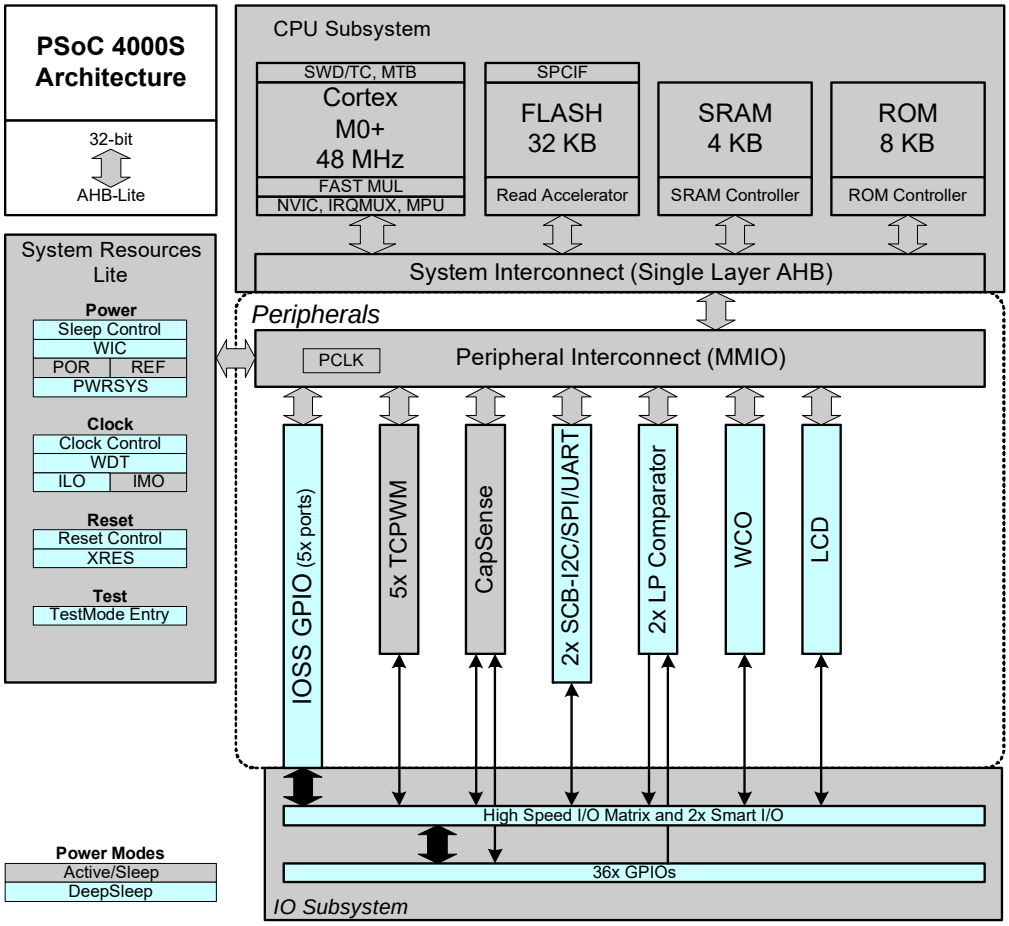
- 高性能、32 ビットのシングル サイクル Cortex-M0+ CPU コア
- 高性能アナログ システム
- 自己および相互静電容量式タッチ センシング (CapSense®)
- コンフィギュレーション可能なタイマー／カウンター／ PWM ブロック
- コンフィギュレーション可能な通信ブロック : I^2C、SPI および UART 動作モード
- 低消費電力動作モード : スリープ およびディープスリープ

本書は PSoC 4000S デバイスの各機能ブロックについて詳しく説明します。この情報は設計者がシステムレベル設計を作成するために役に立ちます。

1.1 トップ レベル アーキテクチャ

図 1-1 に PSoC 4000S アーキテクチャの主なコンポーネントを示します。

図 1-1. PSoC 4000S ファミリ ブロック図



1.2 特長

PSoC 4000S ファミリは以下の主なコンポーネントがあります :

- 最大 0.9DMIPS/MHz を供給するシングルサイクル乗算器を搭載した 32 ビット Cortex-M0+ CPU
- 最大 32KB フラッシュおよび 4KB SRAM
- 5 つの 中央揃えパルス幅変調器 (PWM) コンプリメンタリ、デッドバンド プログラマブル出力を搭載
- 2 つの低消費電力コンパレータ
- 2 つのシリアル通信ブロック (SCB) SPI、UART、I^2C およびローカル インターコネクト ネットワーク (LIN) スレーブ シリアル通信チャネルとして動作可能
- スマート I/O ブロック、I/O 信号パスでブール関数を実行可能
- CapSense
- セグメント LCD ダイレクト ドライブ
- 低消費電力動作モード : スリープ およびディープスリープ
- シリアル ワイヤ デバッグ (SWD) を介したプログラミングとデバッグ システム
- PSoC Creator™ IDE ツールによる完全サポート

1.3 CPU システム

1.3.1 プロセッサ

PSoC 4 の心臓部は 32 ビット Cortex-M0+ CPU コアです。PSoC 4000S は最大 48MHz で動作します。このプロセッサは広範なクロックゲーティングを搭載し、低消費電力動作に最適化されています。16 ビット命令を使用し、Thumb-2 命令セットを実行します。この命令セットにより、バイナリコードは完全な上位互換性があり、Cortex M3 や M4 等の高性能プロセッサに使用できます。

CPU は 1 サイクルで 32 ビット結果を出すハードウェア乗算器を搭載しています。

1.3.2 割り込みコントローラー

CPU サブシステムには、16 の割り込み入力を持つネスト型ベクタ割り込みコントローラー (NVIC) とプロセッサをディープスリープ モードから復帰できるウェイクアップ割り込みコントローラー (WIC) があります。

1.4 メモリ

PSoC 4 メモリ サブシステムはフラッシュと SRAM で構成されています。ブートおよびコンフィギュレーション ルーチンを格納する監視 ROM が提供されています。

1.4.1 フラッシュ

PSoC 4 はフラッシュ ブロックからの平均アクセス時間を改善するCPUに密に結合したフラッシュ アクセラレータ付きのフラッシュ モジュールを搭載しています。フラッシュ アクセラレータはシングルサイクル SRAM のアクセス性能平均の 85% を達成します。

1.4.2 SRAM

PSoC 4 は SRAM を提供し、これはデバイスのすべての消費電力モードでデータを保持します。

1.5 システムワイド リソース

1.5.1 クロック システム

クロック システムは内部クロックとして、内部主発振器 (IMO) および内部低速発振器 (ILO) を構成し、外部クロックおよび時計用水晶発振器 (WCO) に対応できます。

IMO は ±2%の精度を持ち、デバイスの内部クロックの主要供給源です。IMO の初期設定周波数は 24MHz であり、4MHz 単位で 24MHz ～ 48MHz の間で調整できます。アプリケーションの要件を満たすために、複数のクロック周波数はメイン クロック周波数から生成されます。

ILO は低消費電力ですが精度の低い発振器であり、ディープスリープモードで周辺機能を動作させるためのクロックを生成する LFCLK の信号源として使用されます。そのクロック周波数は 40kHz で ±60 パーセントの精度です。

1MHz ～ 48MHz の外部クロック ソースは、IMO の代わりに機能ブロック用の派生クロックを生成するために使用することができます。

1.5.2 電源システム

デバイスは 1.71V ～ 5.5V の単一外部電源で動作します。これは複数電源供給ドメインを提供します – デジタル セクション電源用 V_{DDD} とアナログ セクション用にノイズ隔離する V_{DDA} とを外部で短絡する必要があります。

デバイスはスリープ およびディープスリープの 2 つの低消費電力モードがあり、デフォルトはアクティブ モードです。アクティブ モードでは、すべてのデジタル回路に電源が供給され CPU が動作します。スリープ モードでは、すべての他のペリフェラル機能を含めて、CPU は電源オフです。ディープスリープ モードでは、CPU、SRAM および高速ロジックはデータ保持状態に入ります ; メイン システム クロックがオフにされ、低速クロックはオンで、低速ペリフェラルは動作し続けます。

各種消費電力モードで電源供給に対応するため、多数の内部レギュレータがシステムで利用可能です。

1.5.3 GPIO

すべての GPIO は以下の特性を持ちます :

- 8 つの駆動能力モード
- 入力と出力のディスエーブルの個別制御
- 直前の状態をラッチするための保持モード
- 選択可能なスルー レート
- 割り込み生成 – エッジ トリガー

更に、デバイスはポート I/O でブール関数を実行することができる 2 つのスマート I/O ブロックがあります。スマート I/O ブロックは、低消費電力モードを含むすべてのデバイス消費電力モードで使用可能です。

ピンは 8 ビット幅のポートで構成されます。高速 I/O マトリックスは、I/O ピンに接続できる複数の信号を多重化するために使用されます。固定機能ブロックのピン位置も固定されています。

1.6 固定機能デジタル

1.6.1 タイマー／カウンター／ PWM ブロック

タイマー／カウンター／PWMブロックはユーザープログラム可能な周期の 5 つの 16 ビット カウンターで構成されます。TCPWM ブロックはキャプチャ レジスタ、周期レジスタおよび比較レジスタを持ちます。ブロックはコンプリメンタリ、デッドバンド プログラマブル出力をサポートします。これは所定の状態にする強制的な出力を行うキル入力も有ります。ブロックの他の機能では中央揃え PWM、クロック プリスケール、擬似ランダム PWM および直交デコーダーの機能を含みます。

1.6.2 シリアル通信ブロック

デバイスは 2 つの SCB を持ちます。各 SCB は I^2C、UART、ローカル インターコネクト ネットワーク (LIN) スレーブ、または SPI となるシリアル通信インターフェースを搭載することができます。

各 SCB の 機能を以下に示します :

- 標準 I2C マルチマスターとスレーブの機能
- Motorola、Texas Instruments、National Semiconductor (MicroWire) のプロトコルと互換性のある標準的な SPI マスターとスレーブの機能
- SmartCardリーダー (ISO7816)、IrDAおよびLINプロトコルと互換性のある標準的な UART 送受信機能
- 標準 LIN スレーブは LIN V1.3 および LIN V2.1/2.2 仕様規格に準拠
- 32バイト バッファを利用するEZ機能モードのサポート

1.7 アナログ システム

1.7.1 低消費電力コンパレータ

PSoC 4 は一対の低消費電力コンパレータを持ち、これはすべてのデバイスの消費電力モードで動作することができます。この機能は、低消費電力モード時に外部電圧レベルの監視能力を保持しながら、CPU と他のシステム ブロックを無効にすることが可能です。2 つの入力電圧は、両方ともピンから配線されるか、1 つが AMUXBUS を介して内部信号から配線されます。

1.8 特殊機能ペリフェラル

1.8.1 LCD セグメント駆動

PSoC 4 は最大 8 つのコモンとそれぞれの GPIO をコモンまたはセグメントで駆動するために設定できる LCD コントローラを搭載しています。内部で LCD 電圧を生成する必要のないフル デジタル方式 (デジタル相関および PWM) を使用して LCD セグメントを駆動します。

1.8.2 CapSense

PSoC 4000S デバイスは第 4 世代 CapSense であり、これは以下の機能を持ちます :

- 自己静電容量および相互静電容量ベースのタッチ センシング
- 自己静電容量および相互静電容量ベースのタッチ センシングのそれぞれはクラス最高の SNR を提供する堅牢な CapSense シグマデルタ (CSD) および CapSense クロスポイント (CSX) センシング技術
- ADC として CapSense ブロックを再構成でき、すべての GPIO ピンで ADC 入力をサポート
- プログラマブル基準電圧 (VREF) を利用する、優れた SNR
- 低EMIのためにスペクトラム拡散およびプログラマブル抵抗スイッチをサポート
- CapSense シーケンサーに初期化および構成プロセスの負荷軽減によるスキャン中の CPU オーバヘッドの削減
- シールド信号駆動を使用する耐水性 CapSense 動作
- すべてのGPIOピンでの静電容量式タッチ センシングとシールド

1.8.2.1 IDAC とコンパレータ

CapSense ブロックは 1 つの調整可能な基準電圧で 2 つの IDAC と 1 つのコンパレータを持ち、これらは CapSense を使用しない場合は一般目的に使用することができます。

1.9 プログラムおよびデバッグ

PS0C 4 デバイスはオンチップ SWD インターフェース経由で、デバイスのプログラミングとデバッグの機能をサポートします。PSoC Creator IDE は、完全に統合されたプログラミングとデバッグのサポートを提供します。SWD インターフェースは業界標準のサードパーティ製ツールとも完全互換です。

1.10 デバイス機能の要約

表 1-1 に PSoC 4000S デバイスの要約を示します。

表 1-1. PSoC 4000S デバイスの要約

特長	PSoC 4000S
最大 CPU 周波数	48MHz
フラッシュ	32KB
SRAM	4KB
GPIO (最大)	36
スマート I/O	2 ポート
CapSense	使用可能
LCD ドライバー	使用可能
タイマー、カウンター、PWM (TCPWM)	5

表 1-1. PSoC 4000S デバイスの要約 (続表)

特長	PSoC 4000S
シリアル通信ブロック (SCB)	2
IDAC (CapSense 用)	2
低消費電力コンパレータ (LPCOMP)	2
時計用水晶発振器 (WCO)	使用可能
消費電力モード	アクティブ、スリープおよびディープスリープ

注意事項 : PSoC 4100S Plus デバイスには、2 つの異なるフラッシュおよび SRAM サイズファミリがあります。データシートでは、PSoC 4100S Plus および PSoC 4100S Plus 256 KB と表記されています。PSoC 4100S Plus ファミリには、128 KB のフラッシュ、16 KB の RAM、3 つのスマート I/O ポート、および最大 57 のプログラム可能な GPIO があります。ただし、真の乱数発生器 (TRNG) およびコントローラー エリア ネットワーク (CAN) 機能はありません。他の機能と周辺機器は、PSoC 4100S Plus ファミリと同じです。

2. 開発サポート

2.1 サポート

PSoC® 4 製品の無料サポートを www.cypress.com/psoc4 からオンラインでご利用になれます。トレーニング セミナー、ディスカッション フォーラム、アプリケーション ノート、PSoC コンサルタント、CRM テクニカル サポートの電子メール、知識ベース、アプリケーション サポート エンジニアのリソースがあります。

アプリケーションについて支援が必要な場合は www.cypress.com/support/ をご覧になるか、1-800-541-4736 までお電話ください。

2.2 製品アップグレード

サイプレスは無償 PSoC Creator の予定したアップグレートと改良したバージョンを提供しています。アップグレードは DVD-ROM で代理店から入手できます。また www.cypress.com/psoccreator から直接ダウンロードすることもできます。システム ドキュメントの重要な更新もドキュメントのセクションで提供されます。

2.3 開発キット

サイプレス オンライン ストアでも開発キット、C コンパイラおよび PSoC プロジェクトを失敗なく開発するための必要なアクセサリをお求めいただけます。サイプレス オンライン ストア ウェブサイト www.cypress.com/cypress-store をご覧ください。**製品**ごとに利用可能なアイテムをご覧いただくため、**Programmable System-on-Chip** をクリックしてください。開発キットは Digi-Key、Avnet、Arrow、Future の各社より入手することも可能です。

2.4 アプリケーション ノート

アプリケーション ノート「 AN79953 - Getting Started with PSoC 4」を参照してください。PSoC 4 デバイスの機能および PSoC Creator と PSoC 4 開発キットを使用した簡単な PSoC アプリケーションを迅速に作成する詳細情報があります。

3. 文書の構成

この文書は次の節を含みます :

- 21 ページのセクション B:CPU システム
- 35 ページのセクション C: システムリソース サブシステム (SRSS)
- 83 ページのセクション D: デジタル システム
- 153 ページのセクション E: アナログ システム
- 174 ページのセクション F: プログラムおよびデバッグ

3.1 主要な節

使い易さのため、情報はデバイスの機能によって分けられるセクションと章で構成されます。

- 節 – トップレベル アーキテクチャ、開始方法および規則と製品概要の情報を説明します。
- 章 – セクション トピックに特有の事項を個々に説明します。搭載と使用に関する詳細な情報です。
- 用語集 – テクニカル リファレンス マニュアル (TRM) で使用される専門用語を定義します。用語はボールド イタリック のフォントで表示されます。
- レジスタ テクニカル リファレンス マニュアル – テクニカル リファレンス マニュアルに、すべてのデバイス レジスタの詳細を提供します。これは追加文書です。

3.2 本書の表記法

本書では見出しのフォントの他に、4 つの特徴的なフォントを使用しています。

- 1 番目は文書名またはファイル名に言及する時に使用されるイタリックフォントです。
- 2 番目は本書の用語集で説明される用語に言及する時に使用される**ボールド イタリック**フォントです。
- 3 番目は式の例を示すために使用される Times New Roman フォントです。
- 4 番目はコード例を示すために使用される Courier New フォントです。

3.2.1 レジスタの表記法

レジスタ規則は、PSoC 4000S Family: PSoC 4 Registers TRM にて詳述されます。

3.2.2 数値の表記法

16 進数はすべて大文字で表記し、小文字の「h」を付記しています (例えば「14h」「3Ah」)。*C* のコーディング規則に基づき、接頭語「0x」を使用して 16 進数を表現している場合もあります。2 進数には小文字の「b」を付記しています (例えば「01010100b」や「01000011b」) 。「h」も「b」も付いていない数は 10 進数です。

3.2.3 測定単位

次の表に本書で使用する測定単位を示します。

表 3-1. 測定単位

略語	測定単位
bps	ビット毎秒
°C	摂氏温度
dB	デシベル
fF	フェムトファラッド
Hz	ヘルツ
k	キロ、1000
K	キロ、2^10 (1024)
KB	1024 バイトまたは約 1000 バイト
Kbit	1024 ビット
kHz	キロヘルツ (32.000)
kΩ	キロオーム
MHz	メガヘルツ
MΩ	メガオーム
µA	マイクロアンペア
µF	マイクロファラッド
µs	マイクロ秒
µV	マイクロボルト
µVrms	マイクロボルト (実効値)
mA	ミリアンペア
ms	ミリ秒
mV	ミリボルト
nA	ナノアンペア
ns	ナノ秒
nV	ナノボルト
Ω	オーム
pF	ピコファラッド
pp	ピーク ツー ピーク
ppm	100 万分の 1
SPS	サンプル数毎秒
s	シグマ : 標準偏差値を 1 単位とした表記
V	ボルト

3.2.4 略語

次の表に本書で使用する略号を示します。

表 3-2. 略語

略語	定義
abus	analog output bus (アナログ出力バス)
AC	alternating current (交流電流)
ADC	analog-to-digital converter (アナログ - デジタル変換器)
AHB	AMBA (アドバンスト マイクロコントローラー バス アーキテクチャ) 高性能バス、Arm データ転送バスの一種
API	application programming interface (アプリケーション プログラミング インターフェース)
APOR	analog power-on reset (アナログ パワーオン リセット)
BC	broadcast clock (ブロードキャスト クロック)
BOD	brownout detect (電圧低下検出)
BOM	bill of materials (部品表、員数表)
BR	bit rate (ビット レート)
BRA	bus request acknowledge (バス要求認識)
BRQ	bus request (バス要求)
CAN	controller area network (コントローラー エリア ネットワーク)
CI	carry in (キャリーイン)
CMP	compare (比較)
CO	carry out (キャリー アウト)
COM	LCD common signal (LCD コモン信号)
CPU	central processing unit (中央演算処理装置)
CRC	cyclic redundancy check (巡回冗長検査)
CSD	CapSense Sigma Delta (CapSense シグマ デルタ方式)
CT	continuous time (連続時間)
DAC	digital-to-analog converter (デジタル - アナログ変換器)
DAP	debug access port (デバッグ アクセス ポート)
DC	direct current (直流)
DI	digital or data input (デジタル入力またはデータ入力)
DMA	direct memory access (直接メモリ アクセス)
DMIPS	Dhrystone million instructions per second (ドライストーン 100 万命令毎秒)
DO	digital or data output (デジタル出力またはデータ出力)
DSI	digital signal interface (デジタル信号インターフェース)
DSM	deep-sleep mode (ディープスリープ モード)
DW	data wire (データ線)
ECO	external crystal oscillator (外部水晶発振器)

表 3-2. 略語 (続表)

略語	定義
EEPROM	electrically erasable programmable read only memory (電気的消去プログラム可能な読み出し専用メモリ)
EMIF	external memory interface (外部メモリ インターフェース)
FB	feedback (フィードバック)
FIFO	first in first out (先入れ先出しメモリ)
FSR	Full scale range (フル スケール範囲)
GPIO	general purpose I/O (汎用 I/O)
HCI	host-controller interface (ホストコントローラー インターフェース)
HFCLK	high-frequency clock (高周波クロック)
HSIOM	high-speed I/O matrix (高速 I/O マトリクス)
I^2C	inter-integrated circuit (インターインテグレーテッド サーキット)
IDE	integrated development environment (統合開発環境)
ILO	internal low-speed oscillator (内部低速発振器)
ITO	indium tin oxide (インジウム錫酸化物)
IMO	internal main oscillator (内部主発振器)
INL	integral nonlinearity (積分非直線性)
I/O	input/output (入出力)
IOR	I/O read (I/O 読み出し)
IOW	I/O write (I/O 書き込み)
IRES	initial power on reset (初期パワーオン リセット)
IRA	interrupt request acknowledge (割り込み要求認識)
IRQ	interrupt request (割り込み要求)
ISR	interrupt service routine (割り込みサービス ルーチン)
IVR	interrupt vector read (割り込みベクタ読み出し)
LCD	liquid crystal display (液晶ディスプレイ)
LFCLK	low-frequency clock (低周波クロック)
LPCOMP	low-power comparator (低消費電力コンパレータ)
LRb	last received bit (最後に受信したビット)
LRB	last received byte (最後に受信したバイト)
LSb	least significant bit (最下位ビット)
LSB	least significant byte (最下位バイト)
LUT	lookup table (ルックアップ テーブル)
MISO	master-in-slave-out (マスターインスレーブアウト)
MMIO	memory mapped input/output (メモリマップ入出力)
MOSI	master-out-slave-in (マスターアウトスレーブイン)
MPU	memory protection unit (メモリ保護ユニット)

表 3-2. 略語 (続表)

略語	定義
MSb	most significant bit (最上位ビット)
MSB	most significant byte (最上位バイト)
MSP	main stack pointer (メイン スタック ポインタ)
NMI	non-maskable interrupt (マスク不可割り込み)
NVIC	nested vectored interrupt controller (ネスト型ベクタ割り込みコントローラー)
PC	program counter (プログラム カウンター)
PCB	printed circuit board (プリント回路基板)
PCH	program counter high (プログラム カウンター上位バイト)
PCL	program counter low (プログラム カウンター下位バイト)
PD	power down (パワー ダウン)
PGA	programmable gain amplifier (プログラマブル ゲイン アンプ)
PM	power management (電源管理)
PMA	PSoC memory arbiter (PSoC メモリ アービタ)
POR	power-on reset (パワーオン リセット)
PPOR	precision power-on reset (高精度パワーオン リセット)
PRS	pseudo random sequence (疑似乱数列)
PSoC®	Programmable System-on-Chip (プログラマブル システムオンチップ)
PSP	process stack pointer (プロセス スタック ポインタ)
PSRR	power supply rejection ratio (電源電圧変動除去比)
PSSDC	power system sleep duty cycle (電源システム スリープ デューティ サイクル)
PWM	pulse width modulator (パルス幅変調器)
RAM	random-access memory (ランダム アクセス メモリ)
RETI	return from interrupt (割り込みから復帰)
RF	radio frequency (無線周波数)
ROM	read only memory (読み出し専用メモリ)
RMS	root mean square (二乗平均平方根)
RW	read/write (読み出し / 書き込み)
SAR	successive approximation register (逐次比較レジスタ)
SEG	LCD segment signal (LCD セグメント信号)
SC	switched capacitor (スイッチド キャパシタ)
SCB	serial communication block (シリアル通信ブロック)
SIE	serial interface engine (シリアル インターフェース エンジン)
SIO	special I/O (特殊 I/O)
SE0	single-ended zero (シングルエンド ゼロ)

表 3-2. 略語 (続表)

略語	定義
SNR	signal-to-noise ratio (信号対雑音比)
SOF	start of frame (フレームの開始)
SOI	start of instruction (命令の開始)
SP	stack pointer (スタック ポインタ)
SPD	sequential phase detector (順次位相検出器)
SPI	serial peripheral interconnect (シリアル ペリフェラル インターコネクト)
SPIM	serial peripheral interconnect master (シリアル ペリフェラル インターコネクト マスター)
SPIS	serial peripheral interconnect slave (シリアル ペリフェラル インターコネクト スレーブ)
SRAM	static random access memory (スタティック ランダム アクセス メモリ)
SROM	supervisory read only memory (監視用読み出し専用メモリ)
SSADC	single slope ADC (シングル スロープ ADC)
SSC	supervisory system call (スーパバイザ システム コール)
SYSCLK	system clock (システム クロック)
SWD	single wire debug (シングル ワイヤ デバッグ)
TC	terminal count (ターミナル カウント)
TCPWM	timer、counter、PWM (タイマー／カウンター／ PWM)
TD	transaction descriptors (トランザクション記述子)
TIA	trans-impedance amplifier (トランスインピーダンス アンプ)
UART	universal asynchronous receiver/transmitter (汎用非同期レシーバー／トランスミッタ)
UDB	universal digital block (汎用デジタル ブロック)
USB	universal serial bus (ユニバーサル シリアル バス)
USBIO	USB I/O (USB 入出力)
VTOR	vector table offset register (ベクタ テーブル オフセット レジスタ)
WCO	watch crystal oscillator (時計用水晶発振器)
WDT	watchdog timer (ウォッチドッグ タイマー)
WDR	watchdog reset (ウォッチドッグ リセット)
XRES	external reset (外部リセット)
XRES_N	external reset、active low (外部リセット、アクティブ LOW)

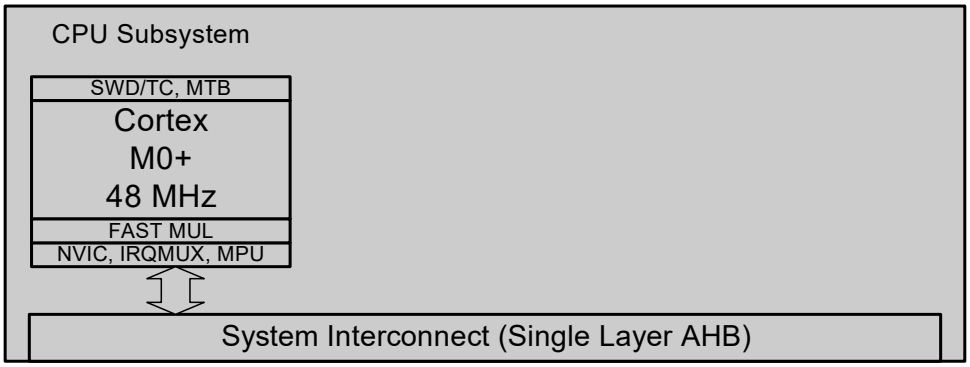
セクション B: CPU システム

このセクションは次の章を含みます。

- 22 ページの Cortex-M0+ CPU の章
- 27 ページの割り込みの章

トップ レベル アーキテクチャ

CPU システム ブロック図



4. Cortex-M0+ CPU

PSoC® 4 Arm Cortex-M0+ コアは低消費電力動作に最適化された 32 ビット CPU です。高効率の 2 段パイプラインおよび固定 4GB メモリ マップがあり、Armv6-M Thumb 命令セットに対応します。Cortex-M0 ＋は 32 ビットのシングル サイクル乗算命令およびレイテンシの短い割り込み処理にも対応します。CPU コアに密接にリンクされるその他のサブシステムはネスト型ベクタ割り込みコントローラー (NVIC)、SYSTICK タイマー、およびデバッグ機能を含みます。

この節では Cortex-M0 ＋プロセッサの概要を説明します。詳細については www.arm.com に掲載されている Arm Cortex-M0+ のユーザー ガイドまたは、テクニカル リファレンス マニュアルを参照してください。

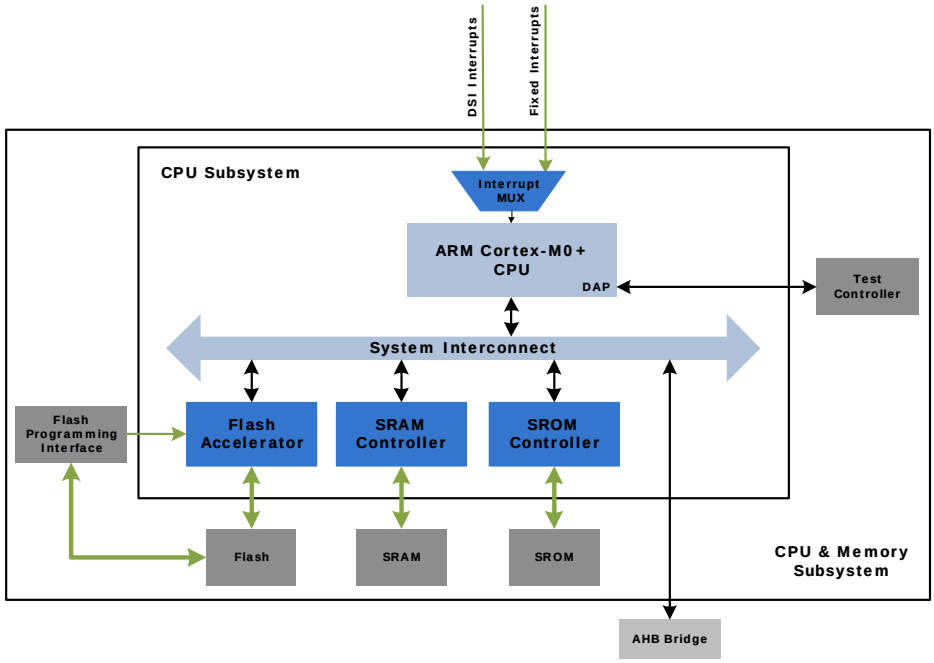
4.1 特長

PSoC 4 Cortex-M0+ は以下の特徴があります :

- プログラムとデバッグが容易であり、8 ビット／ 16 ビットプロセッサからの移行がより簡単
- 最大 0.9DMIPS/MHz で動作します。これは実行速度の向上、消費電力の低減に役立つ
- コード密度を高める Thumb 命令セットに対応し、メモリの効率的な使用を確保
- NVIC ユニットは突発的で確定的な割り込みと例外に対応
- 設計時、コンフィギュレーション可能なメモリ保護ユニット (MPU) を実装
- 非特権モードおよび特権モードの実行をサポート
- 任意のベクタ テーブル オフセット レジスタ (VTOR) をサポート
- 機能強化されたデバッグ機能 :
  - SWD ポート
  - ブレークポイント
  - ウォッチポイント

4.2 ブロック図

図 4-1. CPU サブシステム ブロック図



4.3 動作原理

Cortex-M0+ は 32 ビットのデータ バス、32 ビットのレジスタおよび 32 ビットのメモリ インターフェースを持つ 32 ビットのプロセッサです。Thumb 命令セットのほとんどの 16 ビット命令と Thumb 2 命令セットの一部の 32 ビット命令に対応します。

プロセッサは 2 つの動作モードに対応します (25 ページの動作モードを参照してください)。シングル サイクル 32 ビット乗算命令があります。

4.4 アドレス マップ

Arm Cortex-M0 ＋には固定アドレス マップがあります。これを使用すると、簡単なメモリ アクセス命令でメモリとペリフェラルにアクセスできます。32 ビット (4GB) のアドレス空間は表 4-1 に示す領域に分けられています。コードと SRAM の領域からコードを実行することができることにご注意ください。

表 4-1. Cortex-M0 ＋アドレス マップ

アドレス範囲	名称	用途
0x00000000 ～ 0x1FFFFFFF	コード	プログラム コード領域。データも配置可能。アドレス 0 から始まる例外ベクタ テーブルを保持。
0x20000000 ～ 0x3FFFFFFF	SRAM	データ領域。この領域からのコードを実行することもできる
0x40000000 ～ 0x5FFFFFFF	ペリフェラル	すべてのペリフェラル レジスタ。この領域からのコードを実行することができない。
0x60000000 ～ 0xDFFFFFFF		未使用。
0xE0000000 ～ 0xE00FFFFF	PPB	CPU コア内のペリフェラル レジスタ
0xE0100000 ～ 0xFFFFFFFF	デバイス	PSoC 4 固有の実装

4.5 レジスタ

表 4-2 に示すように、Cortex-M0+ は 16 の 32 ビット レジスタがあります :

- R0 ～ R12: 汎用レジスタ。すべての命令で R0 ～ R7 にアクセスでき、サブセットの命令で他のレジスタにアクセスできます。
- R13: スタック ポインタ (SP)。スタック ポインタは 2 つあり、1 度に 1 つのみ使用可能です。スレッド モードでは、CONTROL レジスタが主スタック ポインタまたはプロセス スタック ポインタ、どちらを使用するかを示します
- R14: リンク レジスタ。関数呼び出し中、戻りプログラム カウンターを保存します。
- R15: プログラム カウンター。このレジスタは制御プログラム フローに書き込むことが可能です。

表 4-2. Cortex-M0+ レジスタ

名称	タイプ[a]	リセット値	説明
R0 ～ R12	RW	未定義	R0 ～ R12 はデータ操作用の 32 ビットの汎用レジスタ
MSP (R13) PSP (R13)	RW	0x00000000	スタック ポインタ (SP) はレジスタ R13。スレッド モードでは、コントロール レジスタのビット 1 は使用するスタック ポインタを示す : 0 = 主スタック ポインタ (MSP)。これはリセット値 1 = プロセス スタック ポインタ (PSP) リセット時、プロセッサはアドレス 0x00000000 からの値で MSP をロード
LR (R14)	RW	未定義	リンク レジスタ (LR) はレジスタ R14。サブルーチンと関数呼び出し、例外のために返し情報を保存
PC (R15)	RW	[0x00000004]	プログラム カウンター (PC) はレジスタ R15。現在のプログラム アドレスを保持。リセット時、プロセッサはアドレス 0x00000004 からの値で PC をロード。ビット 0 はリセット時 EPSR T ビットにロードされ、1 でなければならない
PSR	RW	未定義	プログラム ステータス レジスタ (PSR) は以下を含む : アプリケーション プログラム ステータス レジスタ (APSR) 実行プログラム ステータス レジスタ (EPSR) 割り込みプログラム ステータス レジスタ (IPSR)
APSR	RW	未定義	APSR は直前の命令を実行後の条件フラグの状態を保持
EPSR	RO	[0x00000004].0	リセット時、EPSR にレジスタ [0x00000004] のビット 0 の値がロードされる
IPSR	RO	0	IPSR は現在の ISR の例外番号を保持する
PRIMASK	RW	0	PRIMASK レジスタは優先順位を設定することですべての例外が有効になることを防止する
CONTROL	RW	0	コントロール レジスタはプロセッサがスレッド モードの時に、使用したスタックを制御する

a. スレッド モードとハンドラ モードのプログラム実行中のアクセス タイプです。デバッグ アクセスは異なることがあります。

表 4-3 に PSR ビットの割り当て方法を示します。

表 4-3. Cortex-M0+ PSR ビット割り当て

ビット	PSR レジスタ	名称	使用法
31	APSR	N	負のフラグ
30	APSR	Z	ゼロ フラグ
29	APSR	C	キャリーまたはボロー フラグ
28	APSR	V	オーバーフロー フラグ

表 4-3. Cortex-M0+ PSR ビット割り当て

ビット	PSR レジスタ	名称	使用法
27 ～ 25	–	–	予約済み
24	EPSR	T	Thumb 状態のビット。常に 1 でなければならない。T ビットが 0 である時の命令実行は HardFault 例外になる
23 ～ 6	–	–	予約済み
5 ～ 0	IPSR	該当なし	現在の ISR 例外番号 : 0 = スレッド モード 1 = 予約 2 = NMI 3 = ハードフォルト 4 – 10 = 予約 11 = スーパバイザ コール 12, 13 = 予約 14 = PendSV 15 = SysTick 16 = IRQ0 … 47 = 32

MSR または CPS 命令を使用して、PRIMASK レジスタのビット 0 をセットあるいはクリアします。ビットが 0 であると例外は有効になります。ビットが 1 であると、設定された優先順位の例外、つまり HardFault、NMI、リセットを除く例外は無効になります。例外一覧については 27 ページの割り込みの章を参照してください。

4.6 動作モード

Cortex-M0＋プロセッサは以下の2つの動作モードに対応します :

- スレッド モード : 通常のアプリケーションはこれを使用します。このモードでは MSP または PSP を使用することが可能です。CONTROL レジスタ ビット 1 は使用するスタック ポインタを決定します。
  - 0 = MSP は現在のスタック ポインタ。
  - 1 = PSP は現在のスタック ポインタ。
- ハンドラ モード : これで例外のハンドラを実行します。MSP は常に使用されます。

スレッド モードでは、MSR 命令を使用して、CONTROL レジスタのスタック ポインタ ビットを設定します。スタック ポインタを交換する時、MSR 命令の後は直ちに ISB 命令を使用します。この動作で ISB 以後の命令に新しいスタック ポインタが確保されます。

ハンドラ モードでは MSP が常に使用されるので、CONTROL レジスタへの明示的な書き込みは無視されます。例外のエントリと戻りメカニズムは CONTROL レジスタを自動的に更新します。

4.7 命令セット

Cortex-M0+ は表 4-4 に示すように、Thumb 命令セットのバージョンを備えています。詳細については Cortex-M0+ の全般的なユーザー ガイドを参照してください。

命令オペランドとして定数、その他の命令固有のパラメーター、Arm レジスタを取ります。命令はオペランドに作用し、結果を転送先レジスタに保存します。PC または SP をオペランド､転送先レジスタとして使用することは多くの命令でできないか或いは制限があります。

表 4-4. Thumb 命令セット

ニーモニック	概要
ADCS	キャリー付き加算
ADD{S}[a]	加算
ADR	PC 相対アドレスからレジスタまで
ANDS	ビット単位 AND
ASRS	算術右シフト
B{cc}	ブランチ { 条件付き }
BICS	ビット クリア
BKPT	ブレークポイント
BL	リンク付きブランチ
BLX	リンク付きブランチ インダイレクト
BX	ブランチ インダイレクト
CMN	比較否定
CMP	比較
CPSID	プロセッサ状態を変更、割り込みを無効化
CPSIE	プロセッサ状態を変更、割り込みを有効化
DMB	データ メモリ バリア

表 4-4. Thumb 命令セット

ニーモニック	概要
DSB	データ同期化バリア
EORS	排他的 OR
ISB	命令同期化バリア
LDM	複数レジスタをロード、インクリメント アフター
LDR	PC 相対アドレスからレジスタをロード
LDRB	ワード単位でレジスタをロード
LDRH	ハーフワード単位でレジスタをロード
LDRSB	符号付きバイト単位でレジスタをロード
LDRSH	符号付きハーフワード単位でレジスタをロード
LSLS	論理左シフト
LSRS	論理右シフト
MOV{S}[a]	移動
MRS	特殊レジスタから汎用レジスタに移動
MSR	汎用レジスタから特殊レジスタに移動
MULS	乗算、32 ビット出力
MVNS	ビット単位 NOT
NOP	なにもしない
ORRS	論理 OR
POP	スタックからレジスタのポップ
PUSH	スタックにレジスタのプッシュ
REV	バイト逆ワード
REV16	バイト逆パック ハーフワード
REVSH	バイト逆符号付きハーフワード
RORS	右ローテート
RSBS	逆減算
SBCS	キャリー付き減算
SEV	イベント送信
STM	複数レジスタの保存、インクリメント アフター
STR	ワード単位レジスタ保存
STRB	バイト単位レジスタ保存
STRH	ハーフワード単位レジスタ保存
SUB{S}[a]	減算
SVC	監視プログラム呼び出し
SXTB	符号拡張バイト
SXTH	符号拡張ハーフワード
TST	論理 AND でテスト
UXTB	1 バイトのゼロ拡張
UXTH	1 ハーフワードのゼロ拡張
WFE	イベントの待機
WFI	割り込みの待機

a. 「S」修飾子により、ADD、SUB または MOV 命令が APSR 条件フラグを更新します。

4.7.1 アドレス アライメント

アラインされたアクセスとは、ワード アライン アドレスがワード アクセスに、ハーフワード アライン アドレスがハーフワードアクセスに使用される動作です。バイト アクセスは常にアラインされます。

Cortex-M0+ プロセッサはアラインされていないアクセスに対応しません。アラインされていないメモリ アクセス動作の実行は HardFault の例外になります。

4.7.2 メモリ エンディアン

Cortex-M0+ は、ワードの最下位バイトが最下位アドレスに保存され、最上位バイトが最上位アドレスに保存されるリトルエンディアン フォーマットを使用します。

4.8 Systick タイマー

Systick タイマーは NVIC と統合され、SYSTICK 割り込みを生成します。この割り込みはリアルタイム システムのタスク管理に使用することができます。タイマーにはカウントダウン値として使用可能な24ビットのリロード レジスタがあります。Systick タイマーは Cortex-M0+ 内部クロックまたは低周波数クロック (LF_CLK) のいずれかを信号源として使用します。

4.9 デバッグ

PSoC 4 は SWD に基づくデバッグ インターフェースを持ちます。4 つのブレークポイント (アドレス) コンパレーターおよび 2 つのウォッチポイント (データ) コンパレーターを備えています。

5. 割り込み

PSoC® 4 の Arm Cortex-M0+ (CM0+) CPU は割り込みと例外をサポートしています。割り込みとは、タイマー、シリアル通信ブロック、ポート ピン信号などの CPU の外部ペリフェラルによって生成されるイベントのことです。例外とは、メモリ アクセス フォルトと内部システム タイマー イベントなどの CPU によって生成されるイベントのことです。割り込みと例外が発生すると、いずれも現時点のプログラム実行が停止され、例外ハンドラーまたは割り込みサービス ルーチン (ISR) が CPU によって実行されます。デバイスは割り込みハンドラー／ ISR と例外ハンドラーの両方が統合された例外ベクタ テーブルを持っています。

5.1 特長

PSoC 4 が対応する割り込み機能は次の通り :

- 16 個の割り込みをサポート
- ネスト型ベクタ割り込みコントローラー (NVIC) が CPU コアに統合され、低い割り込みレイテンシの実現
- ベクタ テーブルは、フラッシュまたは SRAM のいずれかに配置することが可能
- 各割り込みに 0 から 3 までの優先度を設定可能
- レベル トリガーおよびパルス トリガー割り込み信号

5.2 動作原理

図 5-1. PSoC 4 割り込みブロック図

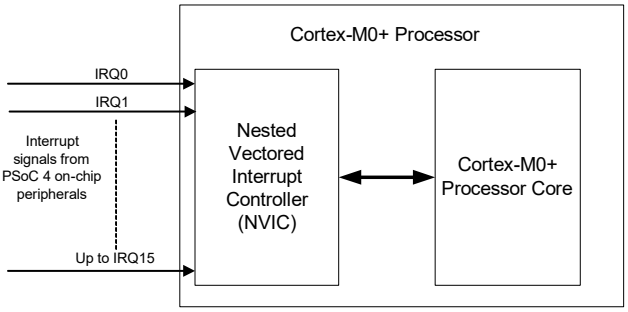


図 5-1 に割り込み信号と Cortex-M0+ CPU との関係を示します。PSoC 4 は 16 個の割り込みがあり、これらの割り込み信号は NVIC によって処理されます。NVIC は個々の割り込みの有効／無効の制御、優先度管理、CPU コアとの通信を担当します。CPU の外部周辺機器によって生成される割り込みと違い、例外は CM0+ のコアで生成されるイベントの一部であるため、図 5-1 に表示されていません。

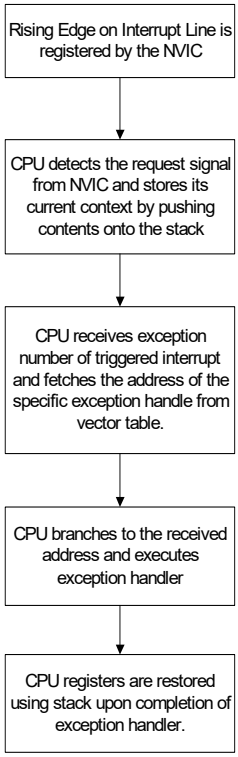
5.3 割り込みと例外の動作

5.3.1 割り込み／例外の処理

割り込みまたは例外のイベントがトリガーされると、以下の一連のイベントが発生します :

1. すべての割り込み信号がアイドルまたは非アクティブの状態であり、プロセッサがメイン コードを実行している状態で、いずれかの割り込みラインが立ち上がったことが NVIC に認識されます。割り込みラインは CPU の処理を待機し、保留状態にします。
2. NVIC からの割り込み要求信号を検出する際に、CPU はスタック上に CPU レジスタの内容をプッシュすることで現在のコンテキストを保存します。
3. また、CPU はトリガーされた割り込みの例外番号を NVIC から受け取ります。表 5-1 に示すように、すべての割り込みと例外は固有の例外番号を持っています。この例外番号を使用することで CPU は具体的な例外ハンドラー アドレスをベクタ テーブルからフェッチします。
4. CPU はこのアドレスへ分岐し、あとに続く例外ハンドラーを実行します。
5. 例外ハンドラーが終了すると、CPU レジスタはスタック ポップ動作を行い元の状態に復元されます。CPU はメイン コードの実行を再開します。

図 5-2. トリガーされた割り込みの処理



NVIC が割り込みの処理中に別の割り込み要求を受ける場合、または複数の割り込み要求を同時に受ける場合、これらのすべての割り込み優先度を評価し、CPU に最も高い優先度の割り込みの例外番号を送信します。従って、優先度の高い割り込みは優先度の低い ISR の実行をいつでもブロックすることができます。

例外は割り込みが処理される場合と同じ方法で処理されます。各イベントには固有の例外番号があり、その番号は適切な例外ハンドラーを実行するために CPU で使用されます。

5.3.2 レベルおよびパルス割り込み

NVIC は、割り込みライン (IRQ0 ～ IRQ15) で、レベルとパルス信号の両方に対応します。割り込みソースに応じてレベルおよびパルスのどちらかに分類されます。

図 5-3. レベル割り込み

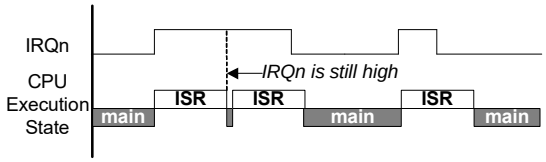


図 5-4. パルス割り込み

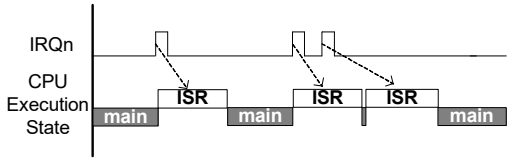


図 5-3と図 5-4はレベルとパルス割り込みの動作を示しています。以下のイベント シーケンスは、割り込みラインが最初に非アクティブ (論理 LOW) であるとして、レベルとパルス割り込みの処理を示しています。

1. 割り込み信号の立ち上がりエッジで、NVIC は割り込み要求を登録します。割り込みは保留状態になります。その割り込み要求が CPU で未処理であることを意味します。
2. NVIC は割り込み要求信号とともに例外番号を CPU に送信します。CPU が ISR の実行を開始すると、割り込みの保留状態がクリアされます。
3. ISR が CPU によって実行されている間に割り込み信号の立ち上がりエッジが検出されると、それは 1 つの保留された割り込み要求として記録されます。現在の ISR の実行が完了すると保留中の割り込みが処理されます (パルス割り込みについては図 5-4 を参照してください)。
4. ISR の完了後に割り込み信号がまだ HIGH である場合、一度保留された後、続いて ISR が実行されます。図 5-3 は割り込み信号が HIGH である限り ISR が実行されるレベル トリガー割り込みを示しています。

5.3.3 例外ベクタ テーブル

例外ベクタ テーブル (表 5-1) はすべての例外ハンドラーに対応するエントリ ポイント アドレスを格納します。CPU は例外番号に基づき、適切なアドレスをフェッチします。

表 5-1. 例外ベクタ テーブル

例外番号	例外	例外の優先度	ベクタ アドレス
–	初期スタック ポインタの値	該当なし (NA)	Base_Address - 0x00000000 (フラッシュ メモリの開始) または 0x20000000 (SRAM の開始)
1	リセット	–3、最優先	Base_Address + 0x0C
2	マスク不可能割り込み (NMI)	–2	Base_Address + 0x08
3	HardFault	–1	Base_Address + 0x0C
4-10	予約済み	該当なし	Base_Address + 0x10 to Base_Address + 0x28
11	スーパバイザ コール (SVCall)	コンフィギュレーション可能 (0 ～ 3)	Base_Address + 0x2C
12-13	予約済み	該当なし	Base_Address + 0x30 to Base_Address + 0x34
14	PendSupervisory (PendSV)	コンフィギュレーション可能 (0 ～ 3)	Base_Address + 0x38
15	システム タイマー (SysTick)	コンフィギュレーション可能 (0 ～ 3)	Base_Address + 0x3C
16	外部割り込み (IRQ0)	コンフィギュレーション可能 (0 ～ 3)	Base_Address + 0x40
…	…	コンフィギュレーション可能 (0 ～ 3)	…
31	外部割り込み (IRQ15)	コンフィギュレーション可能 (0 ～ 3)	Base_Address + 0x7C

表 5-1 では最初のワード (4 バイト) を例外番号 0 として記載していません。例外テーブルの最初のワードは、メイン スタック ポインタ (MSP) の値をデバイスリ セットで初期化するために使用されるためです。例外とみなされません。ベクターテーブルは、ベクターテーブル オフセットレジスタ (VTOR) を変更することにより、メモリマップ (フラッシュまたは SRAM) の任意の場所に配置できます。このレジスタは、0xE000ED08 にある CM0+ のシステム制御空間の一部です。このレジスタは、ベクタテーブル アドレスのビット 31:8 を取ります。ビット 7:0 は予約済みです。したがって、ベクターテーブルのアドレスは 256 バイトに揃える必要があります。SRAM にベクタ テーブルを移動させる利点は、SRAM ベクタ テーブルの内容を変更することにより例外ハンドラーのアドレスを動的に変更することができることです。しかし、不揮発性フラッシュ メモリのベクタ テーブルをフラッシュ メモリ書き込みによって変更する必要があります。

CPUSS_SYSREQレジスタのDIS_RESET_VECT_RELビットが設定されていない限り、スタックポインタを取り出し、ベクタをリセットするために、フラッシュアドレス 0x00000000 および 0x00000004 の読み出しは SROM の最初の 8 バイトにリダイレクトされます。リセット時のこのビットのデフォルト値は0で、リセットベクタが常にSROMから取り出されるようにします。アドレス 0x00000000 および0x00000004からのフラッシュ読み取りを許可するためには、DIS_RESET_VECT_REL ビットを「1」に設定する必要があります。スタック ポインタのベクタは、スタック ポインタがリセット時にロードされるアドレスを保持します。リセット ベクタはブート シーケンスのアドレスを保持します。このマッピングは、デバイス リセットが解除される時に SROM からのスタック ポインタおよびリセット ベクタのデフォルト アドレスを使用するために行われます。リセットのために、まず SROM 内のブート コードを実行し、その後 CPU がフラッシュにハンドラーを実行するためにフラッシュの 0x00000004 アドレスにジャンプします。SRAM ベクタテーブルのリセット例外アドレスは使用されません。

また、CPUSS_SYSREQ レジスタの SYSREQ ビットがセットの場合、0x00000008 フラッシュ アドレスの読み出しは、NMI ベクタ アドレスを取り出すために、フラッシュの代わりに SROM にリダイレクトされます。0x00000008 アドレスでフラッシュを読み出すために CPUSS_SYSREQ をリセットします。

例外ソース (1 ～ 15 までの例外番号) は 5.4 例外ソースで説明します。表 5-1 で予約としてマークされている例外は、使用されていませんが、ベクタ テーブルにそれらのための予約アドレスを持っています。例外ソース (16 ～ 31 までの例外番号) は 5.5 割り込みソースで説明します。

5.4 例外ソース

この節では表 5-1 に記載されている (例外番号 1 ～ 15) 例外ソースについて説明します。

5.4.1 リセット例外

デバイス リセットは PSoC 4 で例外として扱われます。最も優先度の高い例外である -3 の優先度に常に有効にされています。デバイス リセットは、パワーオンリセット (POR)、XRES ピンによる外部リセット信号またはウォッチドッグ

リセットなど複数の原因で発生する可能性があります。デバイスがリセットされると、デバイス コンフィグレーションのための初期ブート コードが監視読み出し専用メモリ (SROM) から実行されます。SROM メモリのブート コードやその他のデータはサイプレスよってプログラムされ、外部ユーザーは読み書きができません。SROM のブート シーケンスが完了した後、CPU のコード実行はフラッシュ メモリへジャンプします。フラッシュ メモリ アドレス 0x00000004 (表 5-1 例外 #1) はフラッシュ メモリ内の起動コードの位置を格納します。CPU はこのアドレスからのコードの実行を開始します。リセット解除時点でデバイスはフラッシュ ベクタ テーブルが選択された状態となるため、SRAM ベクタ テーブルのリセット例外アドレスが使用されることはないことにご注意ください。リセットがアサート解除された後、SRAM ベクタ テーブルを選択するためのレジスタ コンフィギュレーションはフラッシュの起動コードの一部として実行されます。

5.4.2 マスク不可能割り込み (NMI) 例外

マスク不可能な割り込み (NMI) はリセットに次いで最も優先度の高い例外です。これは -2 の固定優先度によって常に有効にされています。デバイスで NMI 例外をトリガーするものは 2 つあります :

- **NMIPENDSET ビット (ユーザー NMI 例外) をセットすることによる NMI 例外 :** NMI 例外は割り込み制御状態レジスタ (CM0P_ICSR レジスタ) に NMIPENDSET ビットを立てることにより、ソフトウェアでトリガーすることができます。このビットを立てることでアクティブなベクタ テーブル (フラッシュまたは SRAM ベクタ テーブル) に示された NMI ハンドラーが実行されます。
- **システム コール NMI 例外 :** フラッシュ書き込み動作とフラッシュ チェックサム動作などの不揮発プログラム動作のために使用されます。CPUSS_SYSREQ レジスタに SYSCALL_REQ ビットを立てることによってトリガーされます。SYSCALL_REQ ビットによってトリガーされた NMI 例外は、SROM に内蔵された NMI 例外ハンドラーのコードを常に実行します。フラッシュまたは SRAM 例外ベクタ テーブルはシステム コール NMI 例外のために使用されません。SROM の NMI ハンドラー コードは、ユーザーによって変更されてはならない不揮発性プログラミング ルーチンが含まれているため、読み出し／書き込みのアクセスができません。

5.4.3 HardFault 例外

HardFault は、通常または例外処理中のエラーが原因で発生する常に有効な例外です。HardFault は -1 の優先度を持ち、優先度が設定できる他の例外よりも高い優先度を持っています。HardFault 例外は、異なる種類のフォルト条件 (未定義命令の実行や無効なメモリ アドレスへのアクセスなど) のためのキャッチ オール例外です。CM0+ CPU は、HardFault 例外ハンドラーにフォルト ステータス情報を提供していません。ただし、ソフトウェアがフォルト状態から回復する能力を有している場合、ハンドラーに例外からの復帰と実行の継続ができるようにしています。

5.4.4 スーパバイザ コール (SVCall) 例外

スーパバイザ コール (SVCall) は CPU が SVC 命令をアプリケーション コードの一部として実行することにより発生する常に有効な例外です。アプリケーション ソフトウェアは SVC 命令を使用して、オペレーティングシステムへの呼び出しを行い、サービスを提供させます。これはスーパバイザコールとして知られています。SVC 命令はアプリケーションがシステムへの特権アクセスを要求するスーパバイザ コールを発行することを可能にします。PSoC 4 の CM0+ がシステム呼び出し NMI 例外に対して SVCall 例外に関連しない特権モードを使用することにご注意ください。(特権モードの詳細は 68 ページのチップ動作モードの章 を参照してください)。デバイスではアーキテクチャレベルで SVCall をサポートする他の特権モードはありません。アプリケーション開発者は、エンド アプリケーションの要件に応じて SVCall 例外ハンドラーを定義する必要があります。

SVCall 例外の優先度は、システム ハンドラー優先度レジスタ 2 (SHPR2) の 2 ビット フィールド PRI_11[31:30] に書き込むことによって、0 と 3 の間の値に設定することができます。SVC 命令が実行された時、SVCall 例外が保留状態に入り、CPU による処理を待ちます。システム ハンドラー制御および状態レジスタ (SHCSR) の SVCALLPENDED ビットは、SVCall 例外の保留状態を確認、変更に使用することができます。

5.4.5 PendSV 例外

PendSV は SVCall に類似するスーパバイザ コールの例外です。通常ソフトウェアで生成されます。PendSV は常に有効であり、その優先度を設定できます。PendSV 例外は、割り込み制御状態レジスタである CM0P_ICSR の PENDSVSET ビットをセットすることでトリガーされます。このビットをセットすると、PendSV 例外が保留状態に入り、CPU による処理を待ちます。PendSV 例外の保留状態は、割り込み制御状態レジスタである CM0P_ICSR の PENDSVCLR ビットを立てることによりクリアすることができます。PendSV 例外の優先度は、システム ハンドラー優先度レジスタ 3 (CM0P_SHPR3) の 2 ビット フィールド PRI_14[23:22] に値を書き込むことによって、0 と 3 の間に設定できます。詳細については Armv6-M Architecture Reference Manual を参照してください。

5.4.6 SysTick 例外

PSoC 4 の CM0+ CPU は、SysTick と呼ばれるシステム タイマーをその内部アーキテクチャの一部として搭載しています。SysTick は、RTOS ティック タイマー、高速アラーム タイマー、単純なカウンターなどの様々な時間管理の目的のために簡単な24ビット デクリメントのカウンターを提供しています。SysTick タイマーはそのカウントがゼロに達した時に割り込みを生成するよう設定することができ、その割り込みを SysTick 例外と呼びます。この例外は、SysTick 制御およびステータスレジスタ (CM0P_SYST_CSR) の TICKINT ビットをセットすることで有効になります。SysTick 例外の優先度は、システム ハンドラー優先度レジスタ 3 (CM0_SHPR3) の 2 ビット フィールド PRI_15[31:30] に値を書き込むことによって、0 と 3 の間に設定できます。SysTick 例外は割り込み制御状態レジスタ CM0P_ICSR の PENDSTSETb ビットに 1 を書き込むことによって、どの瞬間においても常にソフトウェアで生成することができます。同様に、SysTick 例外の保留状態は、割り込み制御状態レジスタである CM0P_ICSR の PENDSTCLR ビットをセットすることによってクリアすることができます。

5.5 割り込みソース

PSoC 4 は、周辺機器からの 16 個の割り込み (IRQ0 ～ IRQ15 または例外番号 16 ～ 31) に対応しています。各割り込みのソースは表 5-3 に記載されています。PSoC 4 は割り込みラインそれぞれのための柔軟な選択機構を提供しています。割り込みは、TCPWM およびシリアル通信ブロックなどのオンチップ周辺機器からの標準の割り込みを含みます。生成された割り込みは、通常異なるペリフェラル ステートとの論理 OR です。ペリフェラル ステータス レジスタは、割り込みを生成した条件を検出するために ISR で読み出されます。割り込みは通常、ペリフェラル ステータス レジスタが割り込みのクリアのために ISR で読み出されることを必要とするレベル割り込みです。ステータス レジスタが ISR で読み出されないと、割り込みがアサートされたままとなり、ISR が継続して実行されます。

GPIO 割り込みの詳細については 37 ページの I/O システムの章 をご覧ください。

表 5-2. PSoC 4 割り込みソースの一覧

割り込み	Cortex-M0+ 例外番号	割り込みソース
NMI	2	SYS_REQ
IRQ0	16	GPIO 割り込み - ポート 0
IRQ1	17	GPIO 割り込み - ポート 1
IRQ2	18	GPIO 割り込み - ポート 2
IRQ3	19	GPIO 割り込み - ポート 3
IRQ4	20	GPIO 割り込み - すべてのポート
IRQ5	21	LPCOMP (低消費電力コンパレータ)
IRQ6	22	WDT (ウォッチドッグ タイマー)
IRQ7	23	SCB0 (シリアル通信ブロック 0)
IRQ8	24	SCB1 (シリアル通信ブロック 1)
IRQ9	25	SPCIF 割り込み
IRQ10	26	CSD (CapSense)
IRQ11	27	TCPWM0 (タイマー／カウンター／ PWM 0)
IRQ12	28	TCPWM1 (タイマー／カウンター／ PWM 1)
IRQ13	29	TCPWM2 (タイマー／カウンター／ PWM 2)
IRQ14	30	TCPWM3 (タイマー／カウンター／ PWM 3)
IRQ15	31	TCPWM4 (タイマー／カウンター／ PWM 4)

5.6 例外の優先度

CPU によって処理される必要がある複数の例外がある時、例外の優先度は例外裁定に有用です。PSoC 4 は異なる例外に優先度を設定することを柔軟に行えます。リセット、NMI および HardFault 以外のすべての例外に優先度を割り当てることができます。リセット、NMI および HardFault 例外はそれぞれ –3、–2、–1 の固定された優先度を持っています。PSoC 4 では優先度番号が小さい方がより高い優先度を表します。よってリセット、NMI、HardFault の例外は最上位優先度になります。他の例外は優先度 0 ～ 3 を割り当てることができます。

PSoC 4は優先度の高い例外が現在アクティブな例外ハンドラーを阻止 (割り込み) 可能なネスト例外をサポートします。後から発生した例外の優先度がアクティブな例外と同じであれば、このプリエンプションは発生しません。CPU は優先度の高い例外を処理した後、優先度の低い例外ハンドラーの実行を再開します。PSoC 4 の CM0+ CPU は最大 4 つの例外ネストを可能にします。CPU は同じ優先度の複数の例外要求を受信した場合、最小の例外番号のものが最初に処理されます。

1 から 15 までの例外番号の優先度を設定するためのレジスタは 29 ページ の「例外ソース」に説明されています。

割り込み優先順位レジスタ (CM0P_IPR) に書き込むことによって、16 個の割り込み (IRQ0 ～ IRQ15) の優先度を設定することができます。表 5-3 に示すように、これは 32 ビット レジスタのグループであり、各レジスタは 4 つの割り込みの優先度を格納します。レジスタの他のビットフィールドは使用されません。

表 5-3. 割り込み優先度レジスタのビット定義

ビット	名称	説明
7:6	PRI_N0	割り込み番号 N の優先度
15:14	PRI_N1	割り込み番号 N+1 の優先度
2322	PRI_N2	割り込み番号 N+2 の優先度
31:30	PRI_N3	割り込み番号 N+3 の優先度

5.7 割り込みの有効と無効

NVIC はソフトウェアで 16 個の割り込みを個別に有効／無効にするためのレジスタを提供します。割り込みが有効にされない場合、NVIC はその割り込みラインの割り込み要求を処理しません。割り込み有効セット レジスタ (CM0P_ISER) と割り込み有効クリア レジスタ (CM0P_ICER) はそれぞれの割り込みを有効／無効にするために使用されます。これらは 32 ビット幅のレジスタで、各ビットは同じ番号の割り込みに対応しています。また、これらの割り込みのイネーブル ステータスを取得するために、レジスタをソフトウェアで読み出すことができます。表 5-4 はこの 2 つのレジスタのためのレジスタ アクセス プロパティを示します。これらのレジスタにゼロを書き込んでも影響がないことにご注意ください。

表 5-4. 割り込みイネーブル／ディスエーブル レジスタ

レジスタ	動作	ビット値	説明
割り込みイネーブル セット レジスタ (CM0P_ISER)	書き込み	1	割り込みを有効にする
		0	何もしない
	読み出し	1	割り込みが有効
		0	割り込みが無効
割り込みイネーブル クリア レジスタ (CM0P_ICER)	書き込み	1	割り込みを無効にする
		0	何もしない
	読み出し	1	割り込みが有効
		0	割り込みが無効

CM0P_ISER と CM0P_ICER レジスタは割り込み (IRQ0 ～ IRQ15) のみに適用可能です。これらのレジスタは例外番号 1 ～ 11 までを有効／無効にするためには使用できません。29 ページ の「例外ソース」で説明されているように、15 個の例外は別の手段で有効／無効にされます。

有効にされているかどうかに関わらず、Cortex-M0+ (CM0+) CPU の PRIMASK レジスタは優先度を設定できる例外をマスクするためのグローバル イネーブル レジスタ として使用されます。優先度を設定できる例外は表 5-1 に記載されているリセット、NMI、HardFault を除くすべての例外を含みます。これらは優先度を 0 から 3 に設定することができ、0 が最高の優先度で、3 が最低の優先度です。PRIMASK レジスタの PM ビット (ビット 0) がセットされると、優先度を設定できる例外のいずれも CPU によって処理されることがありませんが、PM ビットがクリアされた後、それらは CPU によって処理待ちの保留状態になります。

5.8 例外状態

各例外は以下のいずれかの状態にあります。

表 5-5. 例外状態

例外状態	意味
非アクティブ	例外は非アクティブまたは保留中。例外が無効になっているか、有効な例外はトリガーされていない
保留中	例外要求がCPU/NVICによって受信されており、例外が CPU による処理を待機中
アクティブ	CPU で処理されているが、例外ハンドラーの実行が完了していない例外。優先度の高い例外は、優先度の低い例外の実行に割り込むことができる。この場合両方の例外がアクティブ状態になる
アクティブ・保留	例外がプロセッサによって処理されているが、その例外ハンドラーの実行中に、同じソースからの保留中の要求がある

割り込み制御状態レジスタ (CM0P_ICSR) はさまざまな例外状態を説明するステータス ビットが含まれています。

- CM0P_ICSR の VECTACTIVE ビット ([8:0]) は現在実行中の例外番号を格納します。CPU が例外ハンドラーを (CPUはスレッド モードにある) を実行していない場合、この値はゼロです。VECTACTIVE ビット フィールドが、アクティブ例外番号を格納するために使用される割り込みプログラム ステータス レジスタ (IPSR) のビット [8:0] の値と同じであることにご注意ください。
- CM0P_ICSR の VECTPENDING ビット ([20:12]) は最上位優先度の保留中の例外の例外番号を格納します。保留中の例外が存在しない場合、この値はゼロです。
- CM0P_ICSRのISRPENDINGビット (ビット22) はNVICが生成した割り込み (IRQ0 ～ IRQ15) が保留状態にあることを示します。

5.8.1 保留中の例外

周辺装置が NVIC へ割り込み要求信号を生成するが、または例外イベントが発生する時、対応する例外は保留状態に入ります。CPU は対応する例外ハンドラー ルーチンの実行を開始すると、例外が保留状態からアクティブ状態へ変更されます。

割り込みの保留状態を設定またはクリアするための異なるレジスタ ビットを提供することにより、NVIC は 16 個の割り込みラインのソフトウェア保留を可能にします。割り込み保留セットレジスタ (CM0P_ISPR) と割り込み保留クリア レジスタ (CM0P_ICPR) は割り込みラインの保留状態をセットまたはクリアするために使用されます。これらは 32 ビット幅のレジスタで、各ビットは同じ番号の割り込みに対応しています。表 5-6 にこの 2 つのレジスタのためのレジスタ アクセス プロパティを示します。これらのレジスタにゼロを書き込んでも影響がないことにご注意ください。

表 5-6. 割り込み保留セット／クリア レジスタ

レジスタ	動作	ビット値	説明
割り込み保留セット レジスタ (CM0P_ISPR)	書き込み	1	割り込みを保留状態にする
		0	何もしない
	読み出し	1	割り込みが保留中
		0	割り込みが保留中でない
割り込み保留クリア レジスタ (CM0P_ICPR)	書き込み	1	保留の割り込みをクリア
		0	何もしない
	読み出し	1	割り込みが保留中
		0	割り込みが保留中でない

既にセットされているビットに保留ビットを設定しても結果は 1 つのみの ISR の実行です。対応する割り込みが有効になっているかどうかに関わらず、保留ビットの更新は可能です。割り込みが有効にされていない場合、割り込みラインが CM0P_ISER レジスタに書き込んで有効化されるまで、保留状態に移行しません。

CM0P_ISPR と CM0P_ICPR レジスタは 16 個のペリフェラル割り込み (16 ～ 31 の例外番号) のみに使用されることにご注意ください。これらのレジスタは例外番号 1 ～ 11 までを保留するためには利用できません。29 ページ の「例外ソース」で説明されているように、15 個の例外は別の手段で保留されます。

5.9 例外のスタック使用量

CPU はメイン コード (スレッド モードで) を実行し、例外要求が発生すると、CPU は汎用レジスタの状態をスタックに格納します。次に対応する例外ハンドラーの実行 (ハンドラー モード) を開始します。CPU は 8 個の 32 ビットの内部レジスタ をスタックにプッシュします。これらのレジスタはプログラム ステータス レジスタ (PSR)、戻りアドレス、リンク レジスタ (LR または R14)、R12、R3、R2、R1 および R0 です。Cortex-M0+ には MSP と PSP の二種類のスタック ポインタがあります。一度にアクティブにできるのは 1 種類のスタック ポインタのみです。スレッド モードでは、制御レジスタのアクティブ スタック ポインタ ビットが現在のアクティブ スタック ポインタを定義するために使用されます。ハンドラー モードでは、MSP は常にスタック ポインタとして使用されます。Cortex-M0+ のスタック ポインタは常に下向きに進み、最後にプッシュ されたデータを持つアドレスを示します。

CPU がスレッド モードにあり、例外要求が発生した場合、CPU は汎用レジスタの内容を格納するために制御レジスタで定義されたスタック ポインタを使用します。スタック プッシュの動作後、例外ハンドラーの実行をするために、CPU はハンドラー モードに入ります。現在の例外の実行中に別の優先度の高い例外が発生すると、CPU がハンドラー モード中であるため、MSP はスタックのプッシュ／ポップ動作に使用されます。詳細は 22 ページの Cortex-M0+ CPU の章 をご覧ください。

Cortex-M0+ はサービス割り込みにおける待ち時間を削減するためにテールチェーンと後着の 2 つの技術を使用しています。これらの技術は外部ユーザーには見えず、内部プロセッサアーキテクチャの一部です。テール チェインと後着のメカニズムの詳細については Arm Infocenter をご覧ください。

5.10 割り込みと低消費電力モード

特定の周辺割り込み要求が生成された時に PSoC 4 は、低消費電力モードからのデバイス復帰を可能にします。ウェイクアップ割り込みコントローラー (WIC) ブロックは、1 つまたは複数のウェイクアップ ソースが割り込み信号を生成した時に、デバイスがアクティブ モードに移行するために、ウェイクアップ信号を生成します。アクティブ モードに入った後、周辺割り込みの ISR が実行されます。

CM0+ CPU で実行される割り込み待機 (WFI) 命令は、スリープ およびディープ スリープ モードへの遷移をトリガーします。異なる低消費電力モードに入るシーケンスの詳細は 69 ページの消費電力モードの章 を参照してください。チップ低消費電力モードは固定機能割り込みソースの 2 つの分類があります :

- ディープスリープおよびハイバネート モードのみで使用可能な固定機能の割り込みソース (ウォッチドッグ タイマー割り込み、)
- アクティブ モードのみで使用可能な固定機能の割り込みソース (他のすべての固定機能割り込み)

5.11 例外の初期化とコンフィギュレーション

この節は PSoC 4 での例外の初期化とコンフィグレーションに関連のあるさまざまなステップを説明します。

1. 例外ベクタ テーブル位置のコンフィギュレーション : 例外を使用する最初のステップは要求に基づき、フラッシュ メモリまたは SRAM のいずれかにベクタ テーブルを配置することです。このコンフィギュレーションは、VTOR レジスタのビット 31:28 に、ベクタ テーブルが存在するフラッシュまたは SRAM アドレスの値を書き込むことによって行われます。このレジスタ書き込みはデバイスの初期化コードの一部として実行されます。

   アプリケーションがベクタ アドレスを動的に変更する必要がある場合、ベクタ テーブルが SRAM で利用できるようにすることをお勧めします。テーブルがフラッシュに配置されている場合には、フラッシュ書込み動作でベクタ テーブルの内容を変更する必要があります。PSoC Creator IDE はデフォルトで SRAM にベクタ テーブルを配置しています。
2. 個別例外のコンフィグレーション : 次のステップはアプリケーションに必要な個別の例外を設定することです。
   a. 例外または割り込みソースのコンフィグレーションについては、割り込みの発生条件の設定が含まれています。レジスタ コンフィギュレーションは要求された具体的な例外に依存します。
   b. 例外ハンドラー機能を定義し、例外ベクタ テーブルへ機能のアドレスを書き込みます。表 5-1 に例外ベクタ テーブル形式を示します。例外ハンドラーのアドレスがテーブル内の適切な例外番号エントリに書き込まれる必要があります。
   c. 31 ページ の「例外の優先度」で説明したように例外の優先度を設定します。
   d. 32 ページ の「割り込みの有効と無効」で説明したように例外を有効にします。

5.12 レジスタ

表 5-7. レジスター覧

レジスタ名	説明
CM0P_ISER	割り込みイネーブル セット レジスタ
CM0P_ICER	割り込みイネーブル クリア レジスタ
CM0P_ISPR	割り込み保留セット レジスタ
CM0P_ICPR	割り込み保留クリア レジスタ
CM0P_IPR	割り込み優先度レジスタ
CM0P_ICSR	割り込み制御状態レジスタ
CM0P_AIRCR	アプリケーション割り込みおよびリセット制御レジスタ
CM0P_SCR	システム制御レジスタ
CM0P_CCR	コンフィギュレーションおよび制御レジスタ
CM0P_SHPR2	システム ハンドラー優先度レジスタ 2
CM0P_SHPR3	システム ハンドラー優先度レジスタ 3
CM0P_SHCSR	システム ハンドラー制御および状態レジスタ
CM0P_SYST_CSR	Systick 制御およびステータスレジスタ
CPUSS_CONFIG	CPU サブシステム コンフィギュレーションレジスタ
CPUSS_SYSREQ	システム要求レジスタ

5.13 関連文書

- Armv6-M Architecture Reference Manual ― この資料は Arm Cortex-M0+ アーキテクチャを説明し、その中には命令セット、NVIC アーキテクチャおよび CPU レジスタの説明が含まれます。

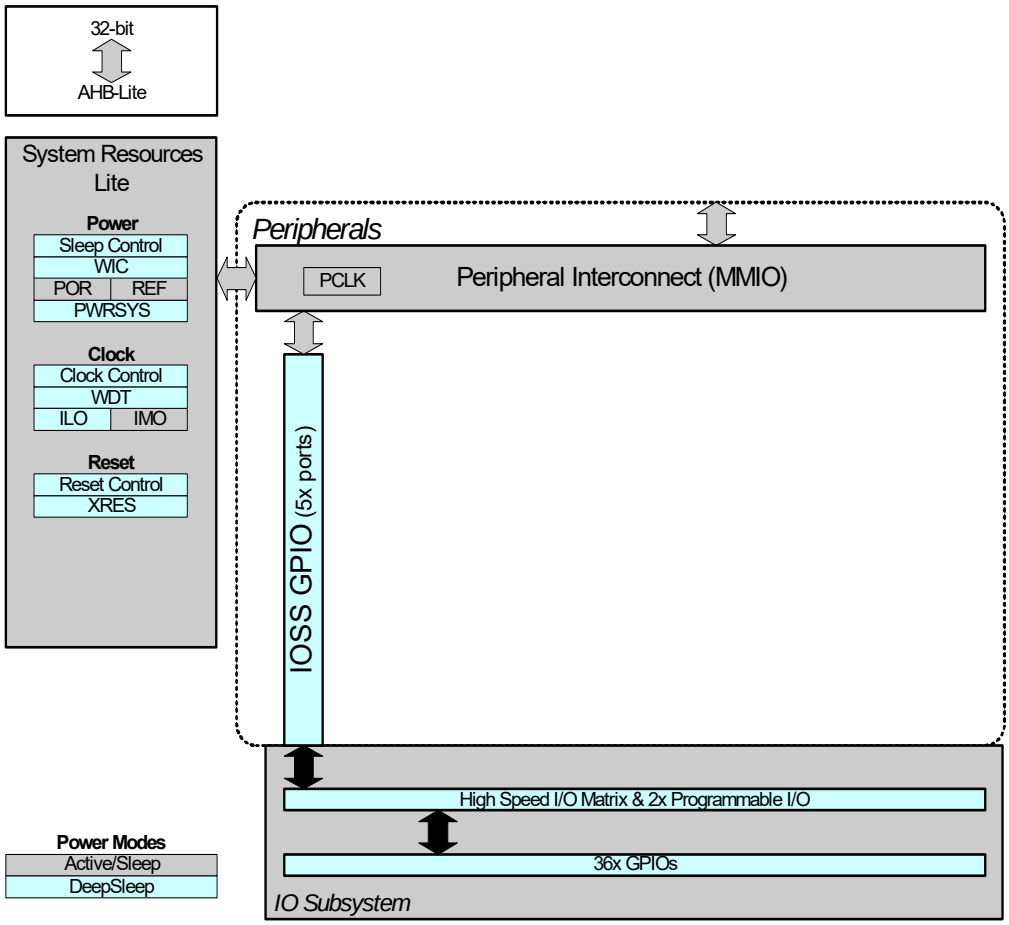
セクション C: システムリソース サブシステム (SRSS)

このセクションは次の章を含みます :

- 37 ページの I/O システム
- 57 ページのクロック供給システム
- 64 ページの電源と電圧監視
- 68 ページのチップ動作モード
- 69 ページの消費電力モード
- 73 ページのウォッチドッグ タイマー
- 78 ページのリセット システム
- 81 ページのデバイス セキュリティ

トップ レベル アーキテクチャ

システムリソース サブシステム ブロック図



6. I/O システム

本章では、PSoC® 4 の I/O システムの特長、アーキテクチャ、動作モードおよび割り込みについて説明します。PSoC 4 の GPIO ピンはポートにグループ分けされ、それぞれのポートは最大 8 本の GPIO を持っています。PSoC 4000S ファミリは 5 つのポートに配置される最大 36 本の GPIO を備えています。

6.1 特長

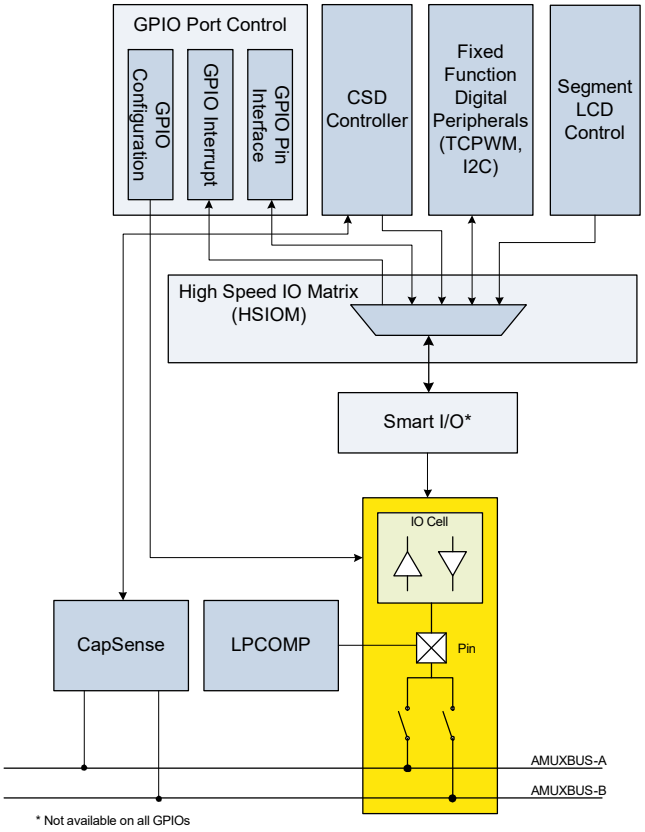
PSoC 4 の GPIO は以下の特長を持っています。

- アナログ - デジタル入出力機能
- 8 つの駆動能力モード
- 個々のピンでエッジ (立ち上がりエッジ、立ち下がりエッジまたは両方のエッジ) でトリガーする割り込み
- スルー レート制御
- 前の状態をラッチするための保持モード (ディープ スリープ モードで I/O 状態を維持するため)
- CMOS および低電圧 LVTTL 入力バッファ モードを選択可能
- スマート I/O ブロックは I/O 信号経路にブール関数を実行する能力を提供
- CapSense サポート
- セグメント LCD 駆動サポート
- アナログ信号を多重化するために使用される 2 つのアナログ マルチプレクサ バス (AMUXBUS-A と AMUXBUS-B)

6.2 GPIO インターフェースの概要

PSoC 4 はアナログおよびデジタル ペリフェラルを搭載しています。図 6-1 はペリフェラルとピン間の配線の概要を示しています。

図 6-1. GPIO インターフェースの概要

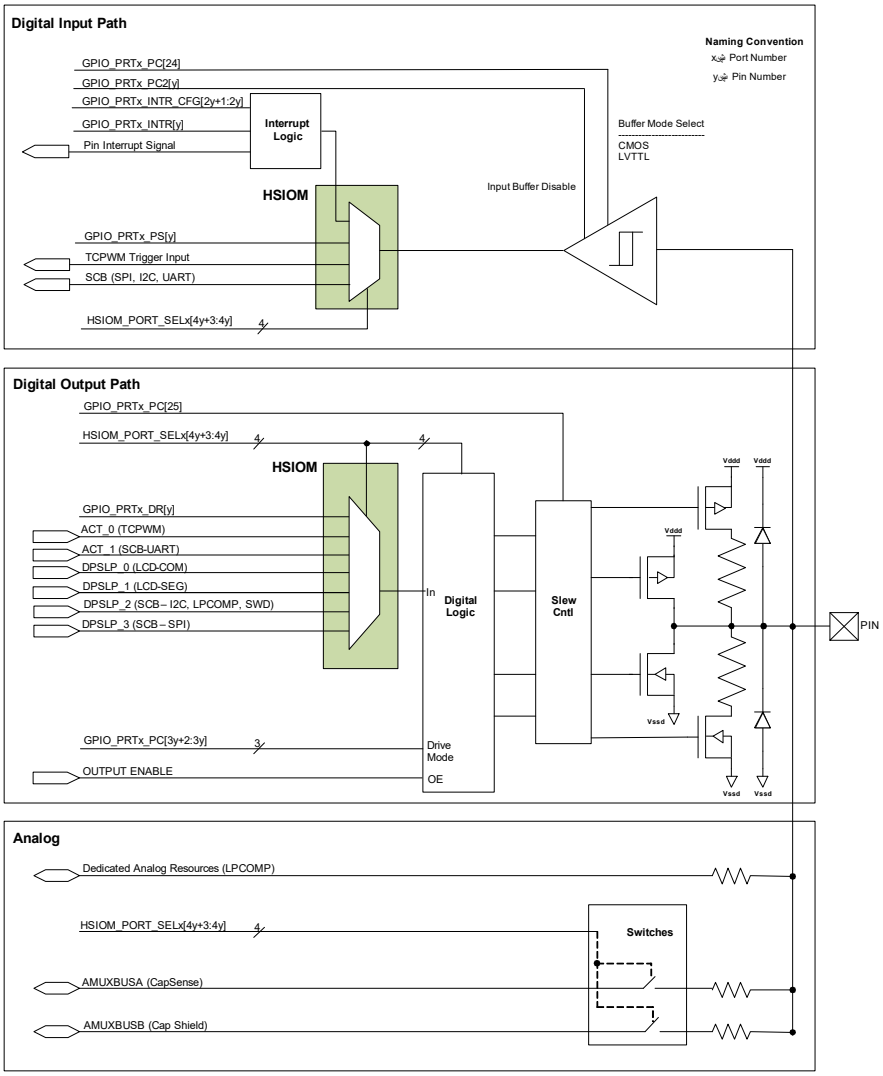


GPIO ピンは I/O セルに接続されます。これらのセルには、より高い入力インピーダンスを実現するためのデジタル入力用入力バッファ、およびデジタル出力信号用ドライバーが搭載されています。デジタル ペリフェラルは高速 I/O マトリク (HSIOM) を介して I/O セルに接続します。HSIOM はユーザーにより選択されるペリフェラルをピンに接続するためのマルチプレクサを含んでいます。幾つかのポート ピンは、HSIOM とピン間にスマート I/O ブロックを搭載しています。スマート I/O ブロックはピン信号上の論理演算を可能にします。アナログ ペリフェラルおよびアナログ マルチプレクサ バスとの接続は GPIO セル内で直接行われます。CapSense ブロックは AMUX バスを通じて GPIO ピンに接続されます。

6.3 I/O セルのアーキテクチャ

図 6-2 は I/O セルのアーキテクチャを示します。これは入力バッファと出力ドライバーから構成されています。このアーキテクチャはあらゆる GPIO セルで共通です。デジタル入出力信号用に HSIOM マルチプレクサ／スマート I/O ブロッに接続します。

図 6-2. GPIO のブロック図



6.3.1 デジタル入力バッファ

このデジタル入力バッファは、外部のデジタル入力に対して高インピーダンスのバッファを提供します。そのバッファはポート コンフィギュレーション レジスタ 2 (GPIO_PRTx_PC2、「x」はポート番号) の INP_DIS ビットにより有効化または無効化されます。そのバッファは次のモードにてコンフィギュレーション可能です。

- CMOS
- LVTTL

これらのバッファ モードはポート コンフィギュレーション レジスタの PORT_VTRIP_SEL ビット (GPIO_PRTx_PC[24]) により選択されます。

表 6-1. 入力バッファ モード

PORT_VTRIP_SEL	入力バッファ モード
0b	CMOS
1b	LVTTL

それぞれのモードの閾値はデバイス データシートから取得することができます。入力バッファの出力は選択されたペリフェラルに配線するために HSIOM に接続します。HSIOM ポート選択レジスタ (HSIOM_PORT_SELx) に書き込むことで周辺機器を選択します。HSIOM のデジタル入力周辺機器は、図に示されているように、ピンに依存しています。それぞれのピンで利用可能な機能についてはデバイス データシートを参照してください。

6.3.2 デジタル出力ドライバー

ピンはデジタル出力ドライバーにより駆動されます。このドライバーは異なる駆動モードを実装するための回路、およびデジタル出力信号用のスルー レート制御を含んでいます。ペリフェラルはHSIOMを介してデジタル出力ドライバーに接続します。HSIOM ポート選択レジスタ (HSIOM_PORT_SELx) に書き込むことで特定のペリフェラルを選択できます。

PSoC 4000S では、I/O は V_{DDD} 電源で駆動されます。各 GPIO ピンは、ピンの電圧が V_{DDD} 電源電圧を上回らないようにするための ESD ダイオードを備えています。ピンの電圧が I/O 電源電圧 V_{DDD} を上回らず、VSSD を下回らないことを確認してください。GPIO 電圧の絶対最大値・最小値については、デバイス データシートをご覧ください。デジタル出力ドライバーは、ペリフェラルからの DSI 信号、または出力ピンに紐付いているデータ レジスタ (GPIO_PRTx_DR) を使用して有効化・無効化されます。データに対するペリフェラル ソースの選択およびソース選択制御の有効化・無効化については 6.4 高速 I/O マトリックスを参照してください。

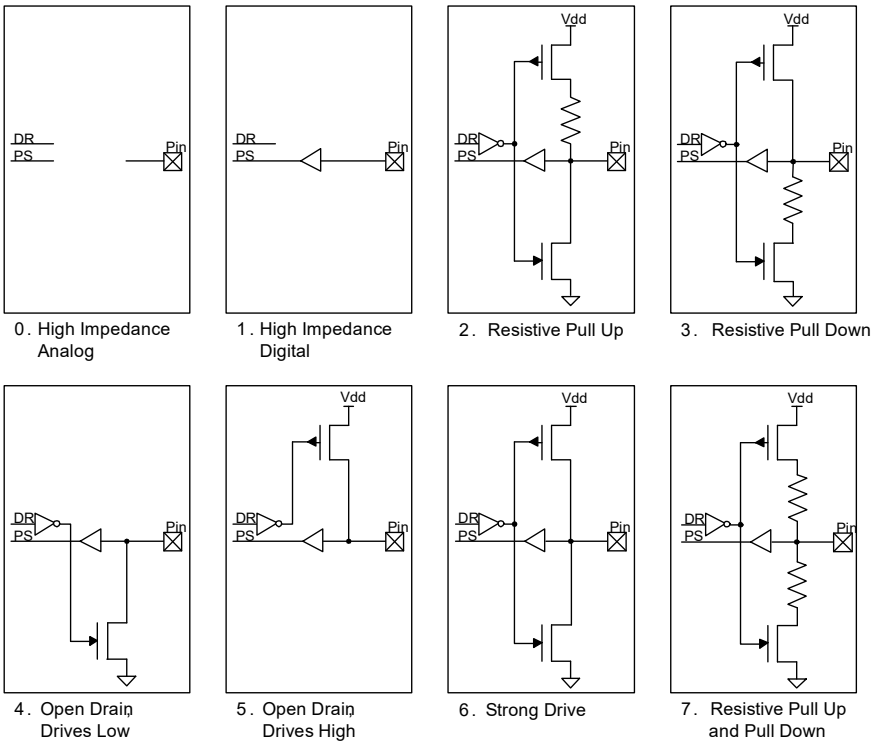
6.3.2.1 駆動モード

各 I/O は、ポート コンフィギュレーション レジスタの GPIO_PRTx_PC を用いて 8 駆動モードのいずれかに個別にコンフィギュレーション可能です。表 6-2 は駆動モードを一覧化したものです。図 6-2 は、それぞれのモードのピン配置を示す出力ドライバーの簡素化された図です。

表 6-2. 駆動モード設定

GPIO_PRTx_PC (「x」はポート番号、「y」はピン番号を示す)				
ビット	駆動モード	値	データ = 1	データ = 0
3y+2: 3y	SEL'y'	ピン「y」に対して駆動モードを選択 (0 ≤ y ≤ 7)		
	高インピーダンス アナログ	0	High Z	High Z
	高インピーダンス デジタル	1	High Z	High Z
	抵抗プルアップ	2	弱 1	強 0
	抵抗プルダウン	3	強 1	弱 0
	オープン ドレイン、LOW に駆動	4	High Z	強 0
	オープン ドレイン、HIGH に駆動	5	強 1	High Z
	ストロング駆動	6	強 1	強 0
	抵抗プルアップとプルダウン	7	弱 1	弱 0

図 6-3. I/O 駆動モードのブロック図



■ 高インピーダンス アナログ

高インピーダンス アナログ モードはデフォルトのリセット状態です。出力ドライバーとデジタル入力バッファの両方がオフになります。この状態は外部電圧による電流のデジタル入力バッファへの流入を防ぎます。この駆動モードは開放しているピンまたはアナログ電圧をサポートしているピンに対して推奨されます。高インピーダンス アナログ ピンをデジタル入力に使用することはできません。データ レジスタの値に関わらず、ピン状態レジスタを読み出すと 0x00 を返します。省電力モードで最も低いデバイス電流を実現するために、未使用の GPIO を高インピーダンスアナログ モードにコンフィギュレーションする必要があります。

■ 高インピーダンス デジタル

高インピーダンス デジタル モードは、デジタル入力に対して推奨される標準の高インピーダンス (High Z) 状態です。この状態では、入力バッファはデジタル入力信号に対して有効になります。

■ 抵抗プルアップまたは抵抗プルダウン

抵抗モードは、一方のデータ状態では直列抵抗を、残り一方の状態ではストロング駆動を提供します。ピンはこれらのモードにおいて、デジタル入力またはデジタル出力のいずれにも使用できます。抵抗プルアップをする場合は、そのピンのデータ レジスタ ビットに「1」を書き込みます。抵抗プルダウンをする場合は、そのピンのデータ レジスタに「0」を書き込みます。メカニカル スイッチへのインターフェースが、これらの駆動モードの一般的な用途です。抵抗モードは、PSoC をオープン ドレイン駆動配線とインターフェースするためにも使用されます。入力がオープン ドレインが LOW の場合は抵抗プルアップ、オープン ドレインが HIGH の場合は抵抗プルダウンを使用します。

■ オープン ドレイン - HIGH に駆動および LOW に駆動

オープン ドレイン モードは、一方のデータ状態では高インピーダンスを、残り一方の状態ではストロング駆動を提供します。ピンはこれらのモードにおいて、デジタル入力またはデジタル出力として使用できます。従ってこのモードは双方向デジタル通信で広く使われています。信号が外部でプルダウンされた場合はオープン ドレイン HIGH 駆動モード、外部でプルアップされた場合はオープン ドレイン LOW 駆動モードを使用します。オープン ドレイン LOW 駆動モードの一般的な用途は、I^2C バス信号ラインを駆動することです。

■ ストロング駆動

ストロング駆動モードは、ピンの標準的なデジタル出力モードです。HIGH 状態と LOW 状態の両方で強い CMOS 出力駆動を提供します。ストロング駆動モードのピンは、通常の状況下で入力として使用しないでください。このモードは多くの場合、デジタル出力信号または外部トランジスタを駆動するために使用されます。

■ 抵抗プルアップおよび抵抗プルダウン

抵抗プルアップおよび抵抗プルダウン モードにおいて、GPIO は、論理 1 と論理 0 の両方の出力状態で直列抵抗を持っています。HIGH データ状態はプルアップになり、一方 LOW データ状態はプルダウンになります。このモードは、バスの短絡を生じる可能性のある他の信号によって駆動される場合に使用します。

6.3.2.2 スルー レート制御

GPIO ピンは、ストロング駆動モードにおいて出力スルー レートが高速および低速の 2 つのオプションがあります。これはポート コンフィギュレーション レジスタ (GPIO_PRTx_PC[25]) を用いてコンフィギュレーションされます。スルー レートは各ピンに対して個別にコンフィギュレーション可能です。このビットはデフォルトでクリアされ、ポートは高速のスルー レートで動作します。低速のスルー レートが必要な場合にこのビットをセットします。スルー レートが低くなると、EMI とクロストークもそれに伴って低下します。そのため、低速オプションは低周波数信号または厳格なタイミング制限のない信号に対して推奨されます。

6.4 高速 I/O マトリックス

高速 I/O マトリック (HSIOM) はデバイス内のペリフェラルへ GPIO を配線する高速スイッチのグループです。GPIO は複数の機能で共有されるため、HSIOM はピンを多重化し、ユーザーにより選択された特定のペリフェラルに接続します。PSoC 4000S、スマート I/O ブロックはポート 2、ポート 3 のピンと HSIOM 間の架け橋としての役割を担います。他のポートは直接 HSIOM に接続します。HSIOM_PORT_SELx レジスタはペリフェラルを選択するために提供されます。これは 1 本のピンが 4 ビットを持つ各ポートで利用可能な 32 ビット幅レジスタです。このレジスタは、表 6-3 に一覧表示されるように、1 本のピンに対して最大 16 の異なるオプションを提供します。

表 6-3. PSoC 4000S HSIOM ポート設定

HSIOM_PORT_SELx (「x」はポート番号、「y」はピン番号を示す)			
ビット	**名称 (SEL'y')**	**値**	**説明 (ピン「y」ソース選択 (0 ≤ y ≤ 7))**
4y+3 : 4y	DR	0	ピンはファームウェアで制御される普通の I/O ピンまたは専用のハードウェア ブロックに接続
	CSD_SENSE	4	ピンは CSD 検出ピン (アナログ モード)
	CSD_SHIELD	5	ピンは CSD シールド ピン (アナログ モード)
	AMUXA	6	ピンは AMUXBUS-A に接続
	AMUXB	7	ピンは AMUXBUS-B に接続。このモードは CSD I/O 充電にも使用。CSD I/O 充電が CSD_CONTROL で有効にされると、デジタル I/O ドライバーは csd_charge 信号に接続 (ピンは依然として AMUXBUS-B に接続)
	ACTIVE_0	8	ピン特有のアクティブ ソース #0 (TCPWM 出力)
	ACTIVE_1	9	ピン特有のアクティブ ソース #1 (SCB-UART)
	ACTIVE_2	10	ピン特有のアクティブ ソース #2 (予約済み)
	ACTIVE_3	11	ピン特有のアクティブ ソース #3 (TCPWM 入力)
	DEEP_SLEEP_0	12	ピン特有のディープスリープ ソース #0 (LCD - COM)
	DEEP_SLEEP_1	13	ピン特有のディープスリープ ソース #1 (LCD - SEG)
	DEEP_SLEEP_2	14	ピン特有のディープスリープ ソース #2 (SCB-I^2C、SWD、LPCOMP)
	DEEP_SLEEP_3	15	ピン特有のディープスリープ ソース #3 (SCB-SPI)

注 アクティブおよびディープスリープのソースはピンに依存しています。各ピンが対応する特長の詳細については、デバイス データシート の「ピン配置」節を参照してください。

6.5 スマート I/O

スマート I/O ブロックはプログラマブルな論理を I/O ポートに追加します。このプログラマブルな論理は AND、OR および XOR 等の基板レベルのブール論理関数をポートに統合します。スマート I/O ブロックは以下の特長を持っています :

- 基板レベルのブール論理関数をポートに統合
- GPIO ポート ピンからの HSIOM 入力信号を前処理する能力
- GPIO ポート ピンへの HSIOM 出力信号を後処理する能力
- あらゆるデバイス電源モードで対応可能
- I/O パッドに密接に統合し、プログラマブルな最短の信号経路を実現

PSoC 4000Sは、ポート2とポート3の2つのポートでスマートI/Oをサポートします。レジスタ命名法として「PRGIO_PRT0」はポート 2 のスマート I/O レジスタを、「PRGIO_PRT1」はポート 3 のスマート I/O レジスタを示します。一般的なスマート I/O レジスタを説明する際は「PRGIO_PRTx」命名法を使用します。

6.5.1 概要

スマート I/O ブロックは HSIOM と I/O ポート間にある信号経路に位置します。HSIOM は固定機能のペリフェラル および CPU から特定のポート ピンへと (逆の場合も同様) 出力信号を多重化します。図 6-4 に示すように、スマート I/O ブロックはこの信号経路に位置し、ポート ピンおよび HSIOM 両方からの信号を処理できるブリッジとして動作します。

図 6-4. スマート I/O のインターフェース

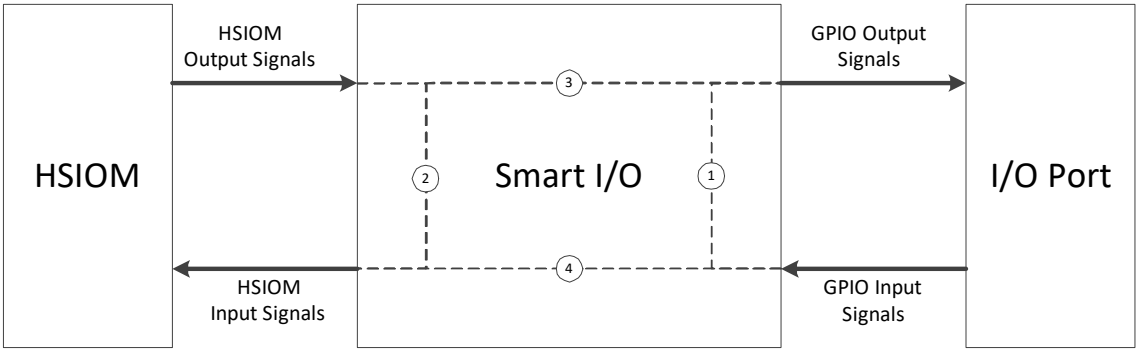


図 6-4 に示すように、スマート I/O ブロック経由でサポートされる信号経路は以下の役割を果たします。

1. I/O ポート信号で直接動作する自己完結型の論理関数を搭載
2. HSIOM 信号または HSIOM 信号と I/O ポート信号の両方で動作する自己完結型の論理関数を搭載
3. HSIOM 出力信号で動作し、その出力信号を変更したり、変更された信号を I/O ポートの信号に配線
4. I/O ポートの信号で動作し、その信号を変更したり、変更された信号を HSIOM 入力信号に配線

次の節ではスマート I/O ブロックのコンポーネント、配線およびコンフィギュレーションを詳細に説明します。それらの節においては、GPIO 信号 (io_data) は I/O ポートからの入出力信号を、デバイスまたはチップ (chip_data) 信号は HSIOM からの入出力信号を指します。

6.5.2 ブロック コンポーネント

スマートI/Oの内部論理には以下のコンポーネントが含まれます :

- クロック／リセット コンポーネント
- シンクロナイザ
- LUT3 コンポーネント
- データ ユニット コンポーネント

6.5.2.1 クロックとリセット

クロックとリセット コンポーネントはスマート I/O ブロックのクロック (clk_block) およびリセット信号 (rst_block_n) を選択します。単一のクロックとリセット信号はブロック内のすべてのコンポーネントに使用されます。クロックとリセットのソースは PRGIO_PRTx_CTL レジスタの CLOCK_SRC[4:0] ビット フィールドにより判定されます。選択されるクロックは、ブロック コンポーネントの I/O 入力シンクロナイザ、LUT およびデータ ユニットの同期ロジックに使用されます。選択されるリセットは LUT とデー

タ ユニット コンポーネントの同期ロジックを非同期でリセットするために使用されます。

ブロックの同期ロジックに選択されるクロック (clk_block) は、デバイスにおける同じクロックで動作する他の同期ロジックと位相が揃っていないことに注意してください。そのため、スマート I/O と他の同期ロジックとの通信は非同期として扱う必要があります。

以下のクロック ソースが利用可能です :

- GPIO 入力信号「io_data_in[7:0]」: これらのクロックに関連するリセットはありません。
- HSIOM 出力信号「chip_data[7:0]」: これらのクロックに関連するリセットはありません。
- スマート I/O クロック (clk_prgio): これはペリフェラル クロック分周期を使用してシステム クロック (clk_sys) から取得されます。ペリフェラル クロック分周期の詳細については 57 ページのクロック供給システムを参照してください。このクロックはアクティブ モードとスリープ モードでのみ利用可能です。このクロックに関連するリセットは rst_sys_act_n か rst_sys_dpslp_n のいずれかです。これらのリセットはブロック同期状態がどのシステム電源モードでリセットされるかを判定します。例えば、rst_sys_act_n リセットはアクティブ モードにあるスマート I/O の同期機能向けのものであり、ディープスリープ モードで有効化されます。
- 低周波数 (40kHz) のシステム クロック (clk_lf): このクロックはディープスリープ モードで利用可能です。このクロックに関連するリセットは rst_lf_dpslp_n です。

ブロックが有効な場合、選択されたクロック (clk_block) および関連リセット (rst_block_n) はファブリック コンポーネントにリリースされます。ファブリックが無効化されると、クロックはファブリック コンポーネントにリリースされず、リセットは有効になります (LUT とデータ ユニット コンポーネントはリセット値の「0」にセットされる)。

I/O 入力シンクロナイザは 2clk_block サイクルの遅延を生じます (シンクロナイザが有効の場合)。その結果として、最初の 2 サイクルでは、ブロックはシンクロナイズ出力から古いデータを受信する可能性があります。従って、最初の 2 サイクル中に、リセットはアクティブ化され、ブロックはバイパス モードにあります。

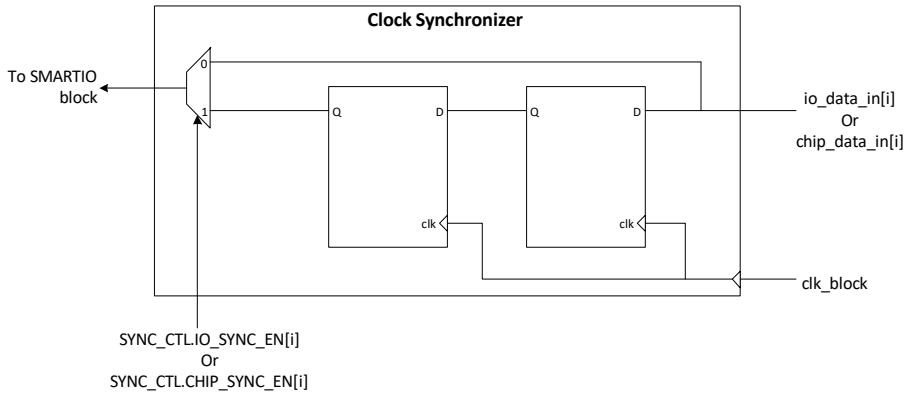
表 6-4. クロックとリセット レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRT0_CTL[12:8]	CLK_SRC[4:0]	クロック (clk_block) ／リセット (rst_block_n) のソース選択 : 「0」: io_data[0]/'1' ... 「7」: io_data[7]/'1' 「8」: chip_data[0]/'1' ... 「15」: chip_data[7]/'1' 「16」: clk_prgio/rst_sys_act_n ; アクティブ モード以外のすべての電源モードでリセットをアサートする。即ち、スマート I/O はペリフェラル分周器からクロック供給され、アクティブ モードでのみ有効である。 「17」: clk_prgio/rst_sys_dpslp_n。スマート I/O はペリフェラル分周期からクロック供給され、すべての電源モードにおいて有効である。ただし、そのクロックは ディープスリープ モードでは無効である。 「19」: clk_lf/rst_lf_dpslp_n。スマート I/O は ILO からクロック供給され、すべての電源モードにおいて有効である。 「20」～「30」: クロック ソースは定数「0」である。低消費電力を確保するために、IP が無効になるとこれらのクロック ソースのいずれかが選択される。 「31」: clk_sys/'1'。この選択は「clk_sys」動作のためのものではない。しかし、非同期動作の場合、IP は有効になって 3 つの「clk_sys」サイクル後に完全に機能するようになる (リセットは非アクティブ)。非同期 (クロックなし) ブロック機能に使用される。

6.5.2.2 シンクロナイザ

各 GPIO 入力信号とデバイス入力信号 (HSIOM 入力) は非同期または同期で使用できます。信号を同期で使用するには、図 6-5 に示すように、信号をスマート I/O クロック (clk_block) に同期化するために、ダブル フリップフロップ シンクロナイザを両方の信号経路に配置します。各ピン／入力の同期化は、PRGIO_PRT0_SYNC_CTL レジスタにある IO_SYNC_EN[i] ビット フィールド (GPIO 入力信号の場合) および CHIP_SYNC_EN[i] (HSIOM 信号の場合) をセットまたはクリアすることで有効化または無効化されます (「i」がピン番号を示す)。

図 6-5. スマート I/O クロック シンクロナイザ



6.5.2.3 LUT3

1 つのスマート I/O ブロックには 8 個のルックアップ テーブル (LUT3) コンポーネントが含まれます。LUT3 コンポーネントは 1 個の 3 入力 LUT と 1 個のフリップフロップから構成されます。各 LUT3 ブロックは 3 つの入力信号を持っており、PRGIO_PRTx_LUT_CTLy レジスタで設定されたコンフィギュレーションに基づいて出力を生成します (y は LUT3 番号を示す)。各 LUT3 のコンフィギュレーションは、PRGIO_PRTx_LUT_CTLy レジスタにある 8 ビットのルックアップ ベクター LUT[7:0] および 2 ビットのオペコード OPC[1:0] により定められます。8 ビットのベクターは 3 つの入力信号に対するルックアップ テーブルとして使用されます。2 ビットのオペコードはフリップフロップの使用を決めます。異なるオペコードに対する LUT3 コンフィギュレーションは図 6-6 に示されます。

PRGIO_PRTx_LUT_SELy レジスタは各 LUT3 に入ってくる 3 つの入力信号 (tr0_in、tr1_in と tr2_in) を選択します。その入力は以下のソースから取得できます :

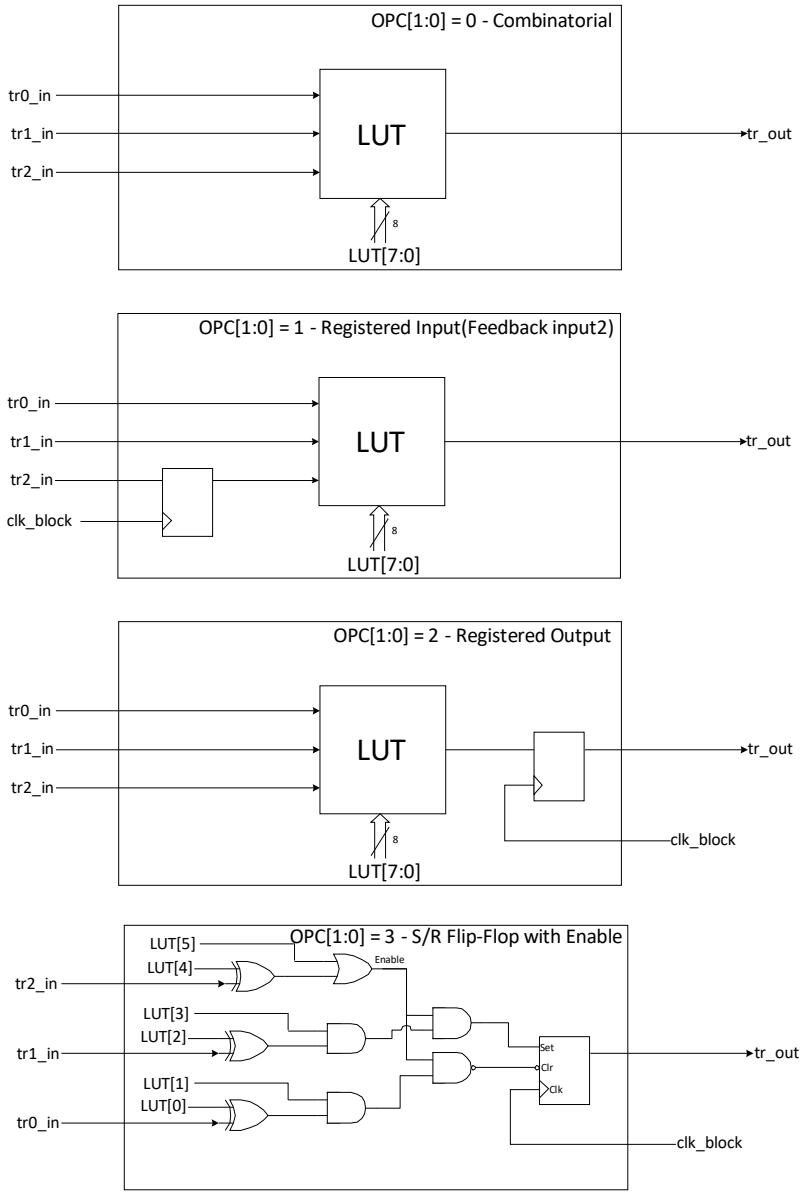
- データ ユニットの出力
- 他の LUT3 の出力信号 (tr_out)
- HSIOM 出力信号 (chip_data[7:0])
- GPIO 入力信号 (io_data[7:0])

PRGIO_PRTx_LUT_SELy レジスタの LUT_TR0_SEL[3:0] ビットは y 番目の LUT3 の tr0_in 信号を選択します。同様に、LUT_TR1_SEL[3:0] ビットおよび LUT_TR2_SEL[3:0] ビットはそれぞれ tr1_in 信号と r2_in 信号を選択します。詳細は表 6-5 を参照してください。

表 6-5. LUT3 レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_LUT_CTLy[7:0]	LUT[7:0]	LUT のコンフィギュレーションである。LUT オペコード (LUT_OPC)、内部状態、および LUT 入力信号の tr0_in、tr1_in、tr2_in に応じて、LUT コンフィギュレーションは LUT 出力信号と次の連続状態を判定するために使用される
PRGIO_PRTx_LUT_CTLy[9:8]	LUT_OPC[1:0]	図 6-6 に示すように LUT オペコードは LUT の動作を特定する
PRGIO_PRTx_LUT_SELy[3:0]	LUT_TR0_SEL[3:0]	LUT 入力信号「tr0_in」のソース選択 : 「0」: データ ユニットの出力 「1」: LUT 1 の出力 「2」: LUT 2 の出力 「3」: LUT 3 の出力 「4」: LUT 4 の出力 「5」: LUT 5 の出力 「6」: LUT 6 の出力 「7」: LUT 7 の出力 「8」: chip_data[0] (LUT 0、1、2、3 用) ; chip_data[4] (LUT 4、5、6、7 用) 「9」: chip_data[1] (LUT 0、1、2、3 用) ; chip_data[5] (LUT 4、5、6、7 用) 「10」: chip_data[2] (LUT 0、1、2、3 用) ; chip_data[6] (LUT 4、5、6、7 用) 「11」: chip_data[3] (LUT 0、1、2、3 用) ; chip_data[7] (LUT 4、5、6、7 用) 「12」: io_data[0] (LUT 0、1、2、3 用) ; io_data[4] (LUT 4、5、6、7 用) 「13」: io_data[1] (LUT 0、1、2、3 用) ; io_data[5] (LUT 4、5、6、7 用) 「14」: io_data[2] (LUT 0、1、2、3 用) ; io_data[6] (LUT 4、5、6、7 用) 「15」: io_data[3] (LUT 0、1、2、3 用) ; io_data[7] (LUT 4、5、6、7 用)
PRGIO_PRTx_LUT_SELy[11:8]	LUT_TR1_SEL[3:0]	LUT 入力信号「tr1_in」のソース選択 : 「0」: LUT 0 の出力 「1」: LUT 1 の出力 「2」: LUT 2 の出力 「3」: LUT 3 の出力 「4」: LUT 4 の出力 「5」: LUT 5 の出力 「6」: LUT 6 の出力 「7」: LUT 7 の出力 「8」: chip_data[0] (LUT 0、1、2、3 用) ; chip_data[4] (LUT 4、5、6、7 用) 「9」: chip_data[1] (LUT 0、1、2、3 用) ; chip_data[5] (LUT 4、5、6、7 用) 「10」: chip_data[2] (LUT 0、1、2、3 用) ; chip_data[6] (LUT 4、5、6、7 用) 「11」: chip_data[3] (LUT 0、1、2、3 用) ; chip_data[7] (LUT 4、5、6、7 用) 「12」: io_data[0] (LUT 0、1、2、3 用) ; io_data[4] (LUT 4、5、6、7 用) 「13」: io_data[1] (LUT 0、1、2、3 用) ; io_data[5] (LUT 4、5、6、7 用) 「14」: io_data[2] (LUT 0、1、2、3 用) ; io_data[6] (LUT 4、5、6、7 用) 「15」: io_data[3] (LUT 0、1、2、3 用) ; io_data[7] (LUT 4、5、6、7 用)
PRGIO_PRTx_LUT_SELy[19:16]	LUT_TR2_SEL[3:0]	LUT 入力信号「tr2_in」のソース選択。エンコーディングは LUT_TR1_SEL と同様

図 6-6. スマート I/O LUT3 コンフィギュレーション



6.5.2.4 データ ユニット

1 つのスマート I/O ブロックは 1 つのデータ ユニット (DU) コンポーネントを含みます。データ ユニットには簡単な 8 ビットのデータパスが含まれます。そのデータパスは簡単なインクリメント、デクリメント、インクリメント／デクリメント、シフト、および AND/OR 動作を実行することができます。DU が実行する動作は、PRGIO_PRTx_DU_CTL レジスタの 4 ビットのオペコード DU_OPC[3:0] を用いて選択されます。

LUT3 コンポーネントと同様に、データ ユニット コンポーネントは最大 3 つの入力トリガー信号 (tr0_in、tr1_in、tr2_in) をサポートします。これらの信号は DU オペコードにより定義される動作を開始するために使用されます。また、データ ユニットには、8 ビットの内部状態 (data[7:0]) を初期化する、またはリファレンスを提供するために使用される 2 つの 8 ビット入力データ (data0_in[7:0] と data1_in[7:0]) も含まれています。これら 8 ビット データへの入力は以下のソースから取得できます :

- 定数「0x00」
- io_data_in[7:0]

- chip_data_in[7:0]
- PRGIO_PRTx_DATA レジスタの DATA[7:0] ビット フィールド

トリガー信号は、PRGIO_PRTx_DU_SEL レジスタの DU_TRx_SEL[3:0] ビット フィールドを使用して選択されます。PRGIO_PRTx_DU_SEL レジスタの DUT_DATAx_SEL[1:0] ビットは 8 ビット入力データのソースを選択します。DU のサイズ (データパスにより使用されるビット数) は PRGIO_PRTx_DU_CTL レジスタの DU_SIZE[2:0] ビットにより定義されます。レジスタ制御の詳細については表 6-6 を参照してください。

表 6-6. データ ユニット レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_DU_CTL[2:0]	DU_SIZE[2:0]	データ ユニットのサイズ／幅 (ビット単位) は DU_SIZE+1 である。例えば、DU_SIZE が 7 の場合、データ ユニットの幅が 8 ビットである
PRGIO_PRTx_DU_CTL[11:8]	DU_OPC[3:0]	データ ユニット オペコードはデータ ユニットの動作を特定する : 「0」: カウントアップ 「1」: カウントダウン 「2」: カウントアップ ラップ 「3」: カウントダウン ラップ 「4」: カウントアップ / ダウン 「5」: カウントアップ / ダウン ラップ 「6」: 右回転 「7」: 右にシフト 「8」: DATA0 & DATA1 「9」: Majority 3 「10」: DATA1 と一致 その他 : 未定義
PRGIO_PRTx_DU_SEL[3:0]	DU_TR0_SEl[3:0]	データ ユニット入力信号「tr0_in」のソース選択 : 「0」: 定数「0」 「1」: 定数「1」 「2」: データ ユニットの出力 「10 ～ 3」: LUT 7 ～ 0 の出力 その他 : 未定義
PRGIO_PRTx_DU_SEL[11:8]	DU_TR1_SEl[3:0]	データ ユニット入力信号「tr1_in」のソース選択。エンコーディングは DU_TR0_SEL と同様
PRGIO_PRTx_DU_SEL[19:16]	DU_TR2_SEl[3:0]	データ ユニット入力信号「tr2_in」のソース選択。エンコーディングは DU_TR0_SEL と同様
PRGIO_PRTx_DU_SEL[25:24]	DU_DATA0_SEL[1:0]	データ ユニット入力信号「data0_in」のソース選択 : 「0」: 定数「0」 「1」: data[7:0] 「2」: gpio[7:0] 「3」: DU Reg
PRGIO_PRTx_DU_SEL[29:28]	DU_DATA1_SEL[1:0]	データ ユニット入力信号「data1_in」のソース選択。エンコーディングは DU_DATA0_SEL と同様
PRGIO_PRTx_DATA[7:0]	DATA[7:0]	データ ユニットの入力データ ソース

データ ユニットは単一の出力トリガー信号 (“tr_out”) を生成します。内部状態 (du_data[7:0]) は clk_block からクロックを要求するフリップフロップに取り込まれます。

次の疑似コードは DU オペコードがサポートするデータパスの様々な動作を説明します。「Comb」は組み合わせ機能のことを指すことにご注意ください。即ち、前の出力状態と無関係に動作する機能です。「Reg」は登録された機能のことを指します。即ち、入力と直前の出力状態で動作する機能です (フリップフロップを用いて登録される)。

```
// The following is shared by all operations.
```

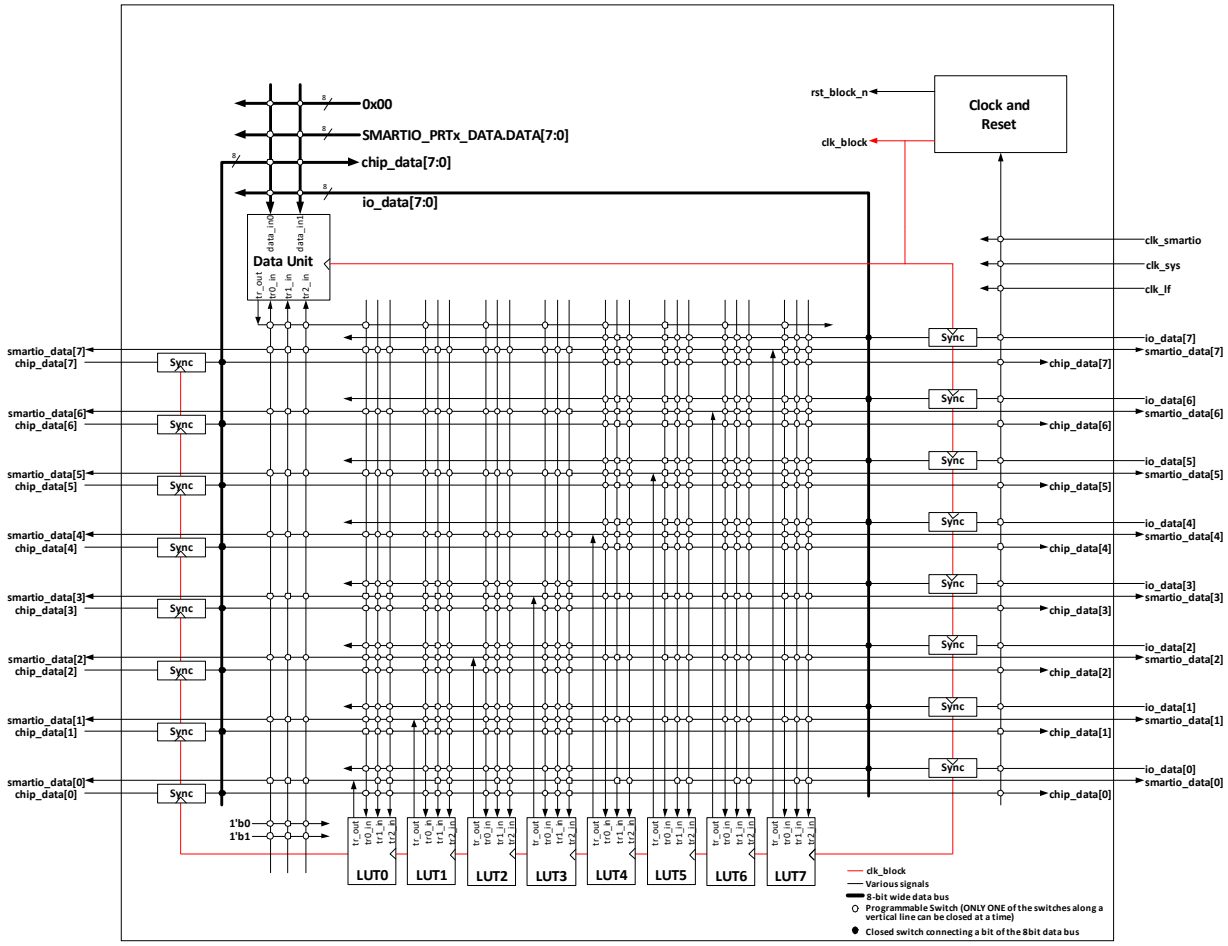
```
mask = (2 ^ (DU_SIZE+1) - 1)
data_eql_data1_in = (data & mask) == (data1_in & mask));
data_eql_0        = (data & mask) == 0);
data_incr         = (data + 1) & mask;
data_decr         = (data - 1) & mask;
data0_masked      = data_in0 & mask;

// INCR operation: increments data by 1 from an initial value (data0) until it reaches a
// final value (data1).
Comb:tr_out = data_eql_data1_in;
Reg:  data <= data;
      if (tr0_in)      data <= data0_masked; //tr0_in is reload signal - loads masked data0
                                             // into data
      else if (tr1_in) data <= data_eql_data1_in ? data : data_incr; //increment data until
                                                                     // it equals data1

// INCR_WRAP operation: operates similar to INCR but instead of stopping at data1, it wraps
// around to data0.
Comb:tr_out = data_eql_data1_in;
Reg:  data <= data;
      if (tr0_in)      data <= data0_masked;
      else if (tr1_in) data <= data_eql_data1_in ? data0_masked : data_incr;

// DECR operation: decrements data from an initial value (data0) until it reaches 0.
Comb:tr_out = data_eql_0;
Reg:  data <= data;
      if (tr0_in)      data <= data0_masked;
      else if (tr1_in) data <= data_eql_0        ? data : data_decr;

// DECR_WRAP operation: works similar to DECR. Instead of stopping at 0, it wraps around to
// data0.
Comb:tr_out = data_eql_0;
Reg:  data <= data;
      if (tr0_in)      data <= data0_masked;
      else if (tr1_in) data <= data_eql_0        ? data0_masked: data_decr;

// INCR_DECR operation: combination of INCR and DECR. Depending on trigger signals it either
// starts incrementing or decrementing. Increment stops at data1 and decrement stops at 0.
Comb:tr_out = data_eql_data1_in | data_eql_0;
Reg:  data <= data;
      if (tr0_in)      data <= data0_masked; // Increment operation takes precedence over
                                             // decrement when both signal are available
      else if (tr1_in) data <= data_eql_data1_in ? data : data_incr;
      else if (tr2_in) data <= data_eql_0  ? data : data_decr;

// INCR_DECR_WRAP operation: same functionality as INCR_DECR with wrap around to data0 on
// touching the limits.
Comb:tr_out = data_eql_data1_in | data_eql_0;
Reg:  data <= data;
      if (tr0_in)      data <= data0_masked;
      else if (tr1_in) data <= data_eql_data1_in ? data0_masked : data_incr;
      else if (tr2_in) data <= data_eql_0  ? data0_masked : data_decr;

// ROR operation: rotates data right and LSB is sent out. The data for rotation is taken from
// data0.
Comb:tr_out = data[0];
Reg:  data <= data;
      if (tr0_in)      data          <= data0_masked;
```

```
      else if (tr1_in) {
                        data          <= {0, data[7:1]} & mask; //Shift right operation
                        data[du_size] <= data[0]; //Move the data[0] (LSB) to MSB
      }

// SHR operation: performs shift register operation. Initial data (data0) is shifted out and
// data on tr2_in is shifted in.
Comb:tr_out = data[0];
Reg:  data <= data;
      if (tr0_in)       data          <= data0_masked;
      else if (tr1_in) {
                        data          <= {0, data[7:1]} & mask; //Shift right operation
                        data[du_size] <= tr2_in; //tr2_in Shift in operation
      }

// SHR_MAJ3 operation: performs the same functionality as SHR. Instead of sending out the
// shifted out value, it sends out a '1' if in the last three samples/shifted-out values
// (data[0]), the signal high in at least two samples. otherwise, sends a '0'. This function
// sends out the majority of the last three samples.
Comb:tr_out =   (data == 0x03)
              | (data == 0x05)
              | (data == 0x06)
              | (data == 0x07);
Reg:  data <= data;
      if (tr0_in)       data          <= data0_masked;
      else if (tr1_in) {
                        data          <= {0, data[7:1]} & mask;
                        data[du_size] <= tr2_in;
      }

// SHR_EQL operation: performs the same operation as SHR. Instead of shift-out, the output is
// a comparison result (data0 == data1).
Comb:tr_out = data_eql_data1_in;
Reg:  data <= data;
      if      (tr0_in) data          <= data0_masked;
      else if (tr1_in) {
                        data          <= {0, data[7:1]} & mask;
                        data[du_size] <= tr2_in;
      }

// AND_OR operation: ANDs data1 and data0 along with mask; then, ORs all the bits of the
// ANDed output.
Comb:tr_out = | (data & data1_in & mask);
Reg:  data <= data;
      if (tr0_in) data <= data0_masked;
```

6.5.3 配線

スマート I/O ブロックは、ブロックに／から信号を配線したり、ブロック内の様々なコンポーネント間に配線したりするために使用される数多くのスイッチを含みます。その配線スイッチは PRTGIO_PRTx_LUT_SELy と PRGIO_PRTx_DU_SEL レジスタを通じて処理されます。詳細はレジスタ TRM を参照してください。スマート I/O の内部配線は図 6-7 に示されています。この図では、LUT7?LUT4 は io_data/chip_data[7]?io_data/chip_data[4] で動作することに対して、LUT3?LUT0 は io_data/chip_data[3]?io_data/chip_data[0] で動作することに注意してください。

図 6-7. スマート I/O の配線



6.5.4 動作

スマート I/O ブロックは下記のように構成し、動作する必要があります。

1. 43 ページの「ブロック コンポーネント」で説明しているように、ブロックを有効にする前に、すべてのコンポーネントを構成し、配線を選択しておく必要があります。
2. コンポーネントのコンフィギュレーションや配線の他に、所望の動作を実現できるように幾つかのブロック レベルの設定を適切にコンフィギュレーションすることが必要です。
   a. バイパス制御 : スマート I/O 経路は、PRGIO_PRTx_CTL レジスタの BYPASS[i] ビット フィールドを設定することにより特定の GPIO 信号に対してバイパスされます。BYPASS[7:0] ビット フィールドでビット「i」が設定されると、「i」番目の GPIO 信号は直接 HSIOM 信号経路にバイパスされます。この場合、スマート I/O 論理はその信号経路に存在しません。これは、スマート I/O 機能が選択 I/O でのみ必要とされる場合には特に有用です。
   b. パイプライン トリガー モード : LUT3 入力マルチプレクサと LUT3 コンポーネント自体はいかなる組み合わせループも含んでいません。同様に、データ ユニットも組み合わせループを含みません。ただし、LUT3 が他の LUT3 またはデータ ユニットとやり取りをする場合は、意図しない組み合わせループが発生することは有り得ます。この制限を受

けないようにするために、PRGIO_PRTx_CTL レジスタの PIPELINE_EN ビット フィールドが使用されます。このビット フィールドが設定される場合、すべての出力 (LUT3 とデータ ユニット) は他のコンポーネントに分岐する前に登録 (フロップ) されます。PIPELINE_EN ビットがクリアされると、出力はフロップされなくなります。

3. スマート I/O ブロックは希望の機能に応じてコンフィギュレーションされた後、PRGIO_PRTx_CTL レジスタの ENABLED ビット フィールドを設定することで有効化されます。ENABLED ビット フィールドがクリアされると、スマート I/O ブロックはバイパス モードに入ります。そのモードにおいては、GPIO 信号は直接 HSIOM 信号により制御されます (逆の場合も同様)。スマート I/O ブロックをコンフィギュレーションしなければなりません。即ち、レジスタの更新中にグリッチを防止するために、ブロックを有効化する前にすべてのレジスタ設定を更新しなければなりません。

表 6-7. スマート I/O ブロック制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_CTL[25]	PIPELINE_EN	パイプライン レジスタの有効化 : 「0」: 無効 (レジスタがバイパスされる) 「1」: 有効
PRGIO_PRTx_CTL[31]	ENABLED	スマート I/O を有効化する。スマート I/O が完全にコンフィギュレーションされた時にのみ「1」に設定するべきである。 「0」: 無効化。(信号がバイパスされる ; BYPASS[7:0] が 0xFF であるかのように振る舞う) 無効の場合、ブロック (データ ユニットと LUT) リセットがアクティブ化される。 ブロックが無効の場合 : 低消費電力を確保するために、PIPELINE_EN レジスタを「1」に設定する必要がある。 低消費電力をを確保するために、CLOCK_SRC レジスタ フィールドを 20 ～ 30 に設定する必要がある (クロックは定数「0」)。 「1」: 有効化。有効科化された場合、ブロック リセットが非アクティブにされ、ブロックが完全に機能できるようになるまで 3 つの「clk_block」クロック サイクルが必要である。この動作により、ブロックが完全に機能できるようになる時に I/O ピンの入力シンクロナイザの状態がフラッシュされることを確保する
PRGIO_PRTx_CTL[7:0]	BYPASS[7:0]	スマート I/O のバイパスであり、1 ビットは 1 つの I/O ピンに対応する : BYPASS[i] は I/O ピン「i」に対応する。ENABLED が「1」の場合、このフィールドは使用される。ENABLED が「0」の場合、このフィールドは使用されず、スマート I/O は常にバイパスされる。 「0」: バイパスなし (スマート I/O が信号経路に存在する) 「'1」: バイパスあり (スマート I/O が信号経路に存在しない)

6.6 電源投入時の I/O 状態

電源投入時に、すべての GPIO は高インピーダンス アナログ状態になり、入力バッファは無効にされます。実行中、GPIO は関連レジスタに書き込むことで設定することができます。デバッグ アクセス ポート (DAP) の接続 (SWD ライン) をサポートするピンは電源投入時に常に SWD ラインとして有効化されることにご注意ください。ただし、DAP の接続は HSIOM を介する様々な用途のために無効化、またはリコンフィギュレーションすることができます。しかしながら、このリコンフィギュレーションはデバイスが起動し、コーディングを開始した後にのみ行われます。

6.7 省電力モードでの動作

表 6-8 は、省電力モードにおける GPIO の状態を示します。

表 6-8. 低消費電力モードの GPIO

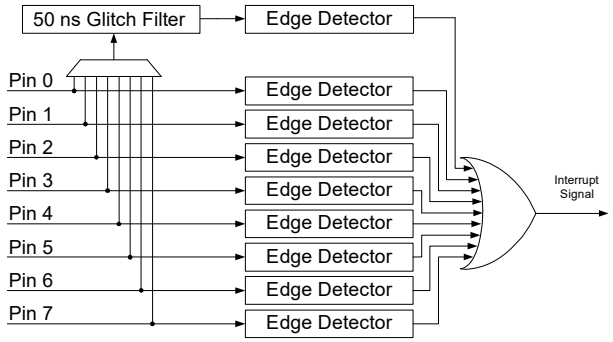
低消費電力モード	状態
スリープ	■ GPIO が有効であり、CapSense、CTBm、TCPWM、SCB などのペリフェラル、およびスリープ モードで動作できる低消費電力コンパレータにより駆動される ■ 入力バッファが有効であるため、あらゆる I/O 上の割り込みを CPU をウェイクアップするために使用できる ■ AMUXBUS 接続は利用可能である
ディープスリープ	■ I^2C と SPI ピンを除いて、GPIO 出力ピンの状態はラッチされ、凍結状態のままである。SCB (I^2C と SPI) ブロックはディープスリープ モードで動作し、アドレス一致時または SPI スレーブ選択イベント時に CPU をウェイクアップすることができる。低電力コンパレータはその専用ピンから信号を受信し、CPU をウェイクアップすることができる。CTBm も専用ピンを持っており、このモードで機能する。 ■ 入力バッファはこのモードでも有効であり；ピン割り込みが機能する ■ AMUXBUS 接続は利用不可である

6.8 割り込み

PSoC 4 デバイスにおいては、すべてのポート ピンが割り込みを生成する能力を有します。図 6-2 に示すように、ピン信号は GPIO エッジ検出ブロックを介して割り込みコントローラに配線されます。

図 6-8 は、GPIO エッジ検出ブロックのアーキテクチャを示しています。

図 6-8. GPIO エッジ検出ブロックのアーキテクチャ



エッジ検出器は各ピンに搭載されます。この検出器により、リコンフィギュレーションすることなく立ち上がりエッジ、立ち下がりエッジ、そしてその両方のエッジを検出することが可能です。表 6-9 に示す通りに、エッジ検出器はポート割り込みコンフィギュレーション レジスタ GPIO_PRTx_INTR_CFG の EDGE_SEL ビットに書き込むことでコンフィギュレーションされます。

表 6-9. エッジ検出器のコンフィギュレーション

EDGE_SEL	コンフィギュレーション
00	割り込みが無効
01	立ち上がりエッジの割り込み
10	立ち下がりエッジの割り込み
11	両エッジの割り込み

ピン以外に、エッジ検出器はグリッチフィルタ出力にも存在します。このフィルタはポートのいずれかのピンで使用されます。表 6-10 に示すように、そのピンは GPIO_PRTx_INTR_CFG レジスタの FLT_SEL フィールドに書き込むことで選択されます。

表 6-10. グリッチフィルタの入力選択

FLT_SEL	選択されるピン
000	ピン 0 が選択される
001	ピン 1 が選択される
010	ピン 2 が選択される
011	ピン 3 が選択される
100	ピン 4 が選択される
101	ピン 5 が選択される
110	ピン 6 が選択される
111	ピン 7 が選択される

ポートのエッジ検出器の出力は OR され、割り込みコントローラ (CPU サブシステムの NVIC) に配線されます。従っ

て、1 つのポートに対して 1 つの割り込みベクターしかありません。ピンの割り込みについては、どのピンが割り込みを発生したかを調べることが必要です。ポート割り込みステータス レジスタの GPIO_PRTx_INTR を読み出すことで情報が得られます。このレジスタには、どのピンが割り込みをトリガーしたかという情報だけでなく、ピンのステータス情報も含まれており、CPU が単一の読み出し動作で両方の情報を読み出すことを可能にします。また、このレジスタにはより重要な役割があります。それは割り込みをクリアすることです。該当するステータス ビットに「1」を書き込むことで割り込みをクリアできます。割り込みステータス ビットをクリアすることが重要です。そうしないと、割り込みは単一のトリガーに対して繰り返して発生するか、または複数のトリガーに対して 1 回だけ応答します。それらの関連事項は本章の後半で説明されます。さらに、該当するポートで割り込みが発生した時にポート割り込み制御ステータス レジスタが読み出されると、割り込みが適切に検出されない原因にもなることにご注意ください。そのため、GPIO 割り込みを使用する場合、コードの他の部分ではなく、該当する割り込みサービス ルーチン内でのみステータス レジスタを読み出すことを推奨します。表 6-11 はポート割り込みステータス レジスタのビット フィールドを示しています。

表 6-11. ポート割り込みステータス レジスタ

GPIO_PRTx_INTR	説明
0000b ～ 0111b	ピン 0 ～ピン 7 の割り込みステータスである。該当するビットに「1」を書き込むことで割り込みをクリアする
1000b	グリッチフィルタからの割り込みステータス
10000b ～ 10111	ピン 0 ～ピン 7 のステータス
11000b	グリッチフィルタの出力ステータス

28 ページ の図 5-3 に示すように、エッジ検出器ブロックの出力が割り込みソース マルチプレクサに配線され、レベルによる検出オプション及び立ち上がりエッジによる検出オプションを提供します。レベル オプションが選択される場合、ポート割り込みステータス レジスタ ビットが設定されている限り、割り込みは繰り返してトリガーされます。立ち上がりエッジ検出オプションが選択される場合、ポート割り込みステータス レジスタがクリアされていなければ、割り込みは一回だけトリガーされます。従って、エッジ検出ブロックが使用される場合、割り込みステータス ビットをクリアすることが重要です。

6.9 ペリフェラルの接続

6.9.1 ファームウェアで制御される GPIO

ファームウェアで制御される GPIO の HSIOM 設定について見てみます。GPIO_PRTx_DR は、GPIO の出力データを読み書きするためのデータ レジスタです。このレジスタへの書き込み動作により、GPIO 出力を書き込まれた値に変更します。読み出し動作は、GPIO の現時点での状態ではなく、このレジスタへ書き込まれた出力データに影響を与えることにご注意ください。このレジスタを使用することにより、入力と出力 GPIO の両方を持つポートで、読み出し - 修正 - 書き込みシーケンスを確実に実行することができます。

データ レジスタ以外に、GPIO_PRTx_DR_SET、GPIO_PRTx_DR_CLR 及び GPIO_PRTx_INV という 3 個のレジスタも提供されており、それぞれのレジスタは他のピンに影響を与えることなくポートの特定のピンの出力データをセット、クリア、そして反転します。これらのレジスタに「1」を書き込むことでデータをセット、クリア、反転します。「0」を書き込む場合はピンの状態に対して何の影響も与えません。

GPIO_PRTx_PS は I/O パッド レジスタであり、読み出されると GPIO の状態を返します。このレジスタに書き込んでも何の反応もありません。

6.9.2 アナログ I/O

LPCOMP等の低インピーダンス配線を必要とするアナログ リソースは専用ピンを持っています。専用アナログ ピンは特定のアナログ ブロックへ直接接続が可能です。専用ピンは

性能の向上に役立てっており、これらのアナログ リソースを使用する際は他のピンよりも優先されます。これらの専用ピンの詳細についてはデバイス データシート を参照してください。

GPIO を専用アナログ I/O としてコンフィギュレーションするためには、高インピーダンス アナログ モードでコンフィギュレーションし (表 6-2 を参照)、特定のアナログ リソースで該当する接続を有効にする必要があります。これは該当のアナログ リソースに関連付けられるレジスタを介して実行されます。

GPIOをAMUXBUSに接続するアナログ ピンとしてコンフィギュレーションするためには、高インピーダンス アナログ モードでコンフィギュレーションし、HSIOM_PORT_ SELx レジスタを用いて AMUXBUS に配線する必要があります。

6.9.3 LCD 駆動

あらゆる GPIO は、LCD コモンまたはセグメントを駆動する能力を有しています。HSIOM_PORT_SELx レジスタは、LCD 駆動ピンを選択するために使用されます。詳細は 161 ページの LCD ダイレクト ドライブを参照してください。

6.9.4 CapSense

CSD をサポートするピンは CapSense ウィジェット (ボタン、スライダー要素、タッチパッドまたは近接センサー等) としてコンフィギュレーションすることができます。CapSense には外部のタンク コンデンサとシールド線も必要です。表 6-12 は CapSense に必要な GPIO と HSIOM 設定を示しています。詳細は、160 ページの CapSense を参照してください。

表 6-12. CapSense 設定

CapSense ピン	GPIO 駆動モード (GPIO_PRTx_PC)	デジタル入力バッファの設定 (GPIO_PRTx_PC2)	HSIOM 設定
センサー	高インピーダンス アナログ	バッファを無効化	CSD_SENSE
シールド	高インピーダンス アナログ	バッファを無効化	CSD_SHIELD
CMOD (通常動作)	高インピーダンス アナログ	バッファを無効化	AMUXBUS A または CSD_COMP
CMOD (GPIO プリチャージ、選択 GPIO でのみ利用可能)	高インピーダンス アナログ	バッファを無効化	AMUXBUS B または CSD_COMP
CSH TANK (GPIO プリチャージ、選択 GPIO でのみ利用可能)	高インピーダンス アナログ	バッファを無効化	AMUXBUS B または CSD_COMP

6.9.5 シリアル通信ブロック (SCB)

UART、I^2C および SPI としてコンフィギュレーション可能な SCB はピンへの専用接続を持っています。これらの専用ピンの詳細についてはデバイス データシートを参照してください。UART と SPI モードが使用される場合、入力ピンを高インピーダンス状態に維持するために、SCB はそのピンに対してデジタル出力バッファ駆動モードを制御します。つまり、SCB ブロックは、SPI マスターとしてコンフィギュレーションされる場合は UART Rx ピンと MISO ピンの出力バッファを無効化し、SPI スレーブとしてコンフィギュレーションされる場合は MOSI と選択ラインの出力バッファを無効化します。この機能は GPIO_PRTx_PC レジスタを用いて行われる駆動モードの設定を上書きします。

6.9.6 タイマー／カウンター／パルス幅変調器 (TCPWM) ブロック

TCPWM はピンへの専用接続を持っています。これらの専用ピンの詳細についてはデバイス データシートを参照してください。開始や停止などの TCPWM ブロック入力がピンから取得される場合、TCPWM ブロックはその入力ピンの出力バッファを無効にするため、駆動モードは High-Z デジタルのみであることにご注意ください。

6.10 レジスタ

表 6-13. I/O レジスタ

名称	説明
GPIO_PRTx_DR	ポート出力データ レジスタ
GPIO_PRTx_DR_SET	ポート出力データ セット レジスタ
GPIO_PRTx_DR_CLR	ポート出力データ クリア レジスタ
GPIO_PRTx_DR_INV	ポート出力データ反転レジスタ
GPIO_PRTx_PS	ポート ピン ステート レジスタ : I/O ピンの論理状態を読み出す
GPIO_PRTx_PC	ポート コンフィギュレーション レジスタ : 出力駆動モード、入力閾値、スルー レートを設定
GPIO_PRTx_PC2	ポート セカンダリ コンフィギュレーション レジスタ : I/O ピンの入力バッファを設定
GPIO_PRTx_INTR_CFG	ポート割り込みコンフィギュレーション レジスタ
GPIO_PRTx_INTR	ポート割り込みステータス レジスタ
HSIOM_PORT_SELx	HSIOM ポート選択レジスタ
PRGIO_PRTx_CTL	スマート I/O ポート制御レジスタ
PRGIO_PRTx_SYNC_CTL	スマート I/O 同期制御レジスタ
PRGIO_PRTx_LUT_SELy	スマート I/O の y 番目の LUT コンポーネント入力選択レジスタ

表 6-13. I/O レジスタ

名称	説明
PRGIO_PRTx_LUT_CTLy	スマート I/O の y 番目の LUT コンポーネント制御レジスタ
PRGIO_PRTx_DU_SEL	スマート I/O デーた ユニットの入力選択レジスタ
PRGIO_PRTx_DU_CTL	スマート I/O データ ユニット制御レジスタ
PRGIO_PRTx_DATA	スマート I/O データ ユニットの入力データ ソース レジスタ

注： GPIO レジスタ名の「x」はポート番号を示します。例えば、GPIO_PTR1_DR はポート 1 の出力データ レジスタを示します。スマート I/O レジスタ名の「x」はスマート I/O ポート番号を示します。スマート I/O のポート番号と実際のポート番号が異なる場合があります。詳細は 6.543 ページのスマート I/O を参照してください。

7. クロック供給システム

PSoC® 4 クロック システムは以下のクロック ソースを含んでいます :

- 2 つの内部クロック ソース :
  - 24 ～ 48MHz 内部主発振器 (IMO) ±2% の精度 (全周波数範囲に渡り、調整可能)
  - 40kHz 内部低速発振器 (ILO) ± 60 パーセントの精度 (IMO を使って校正可能)
- 2 つの外部クロック ソース :
  - I/O ピン上の 1 つの信号により生成された外部クロック (EXTCLK)
  - 外部 32kHz 時計用水晶発振器 (WCO)
- IMO または 外部クロックから選択された最大 48MHz の高周波数クロック (HFCLK)
- ILO から供給された低周波数クロック (LFCLK)
- HFCLK によって供給される最大 48MHz のシステム クロック (SYSCLK) の専用プリスケーラ
- 6 つの 16 ビット ペリフェラル クロック分周器
- 正確なクロック生成のための 2 つの 分数分周器
- 11 個のデジタルおよびアナログ ペリフェラル クロック

7.1 ブロック図

図 7-1 に PSoC 4 デバイスのクロック システムの概要を示します。

図 7-1. クロック システム ブロック図

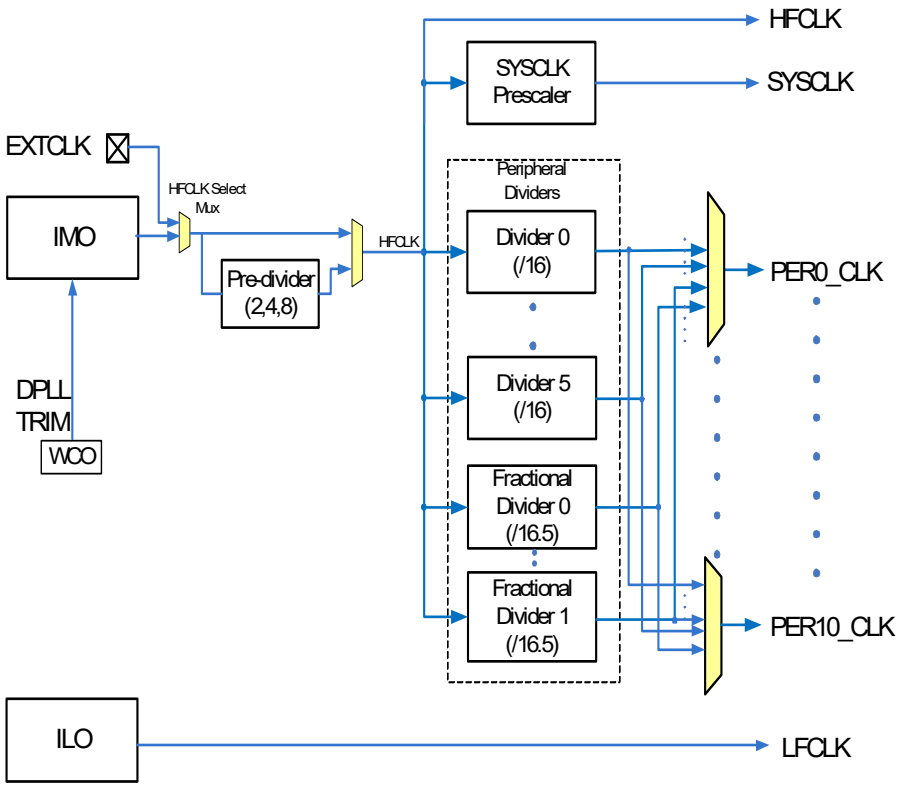


図 7-1 に示すように、デバイスの 4 つのクロック ソースは IMO、EXTCLK、WCO および ILO です。HFCLK マルチプレクサは EXTCLK または IMO から HFCLK ソースを選択します。HFCLK 周波数は最大 48MHz です。

7.2 クロック ソース

7.2.1 内部主発振器 (IMO)

内部主発振器 (IMO) は、アクティブおよびスリープ モードの間で、主クロック ソースとして利用可能な正確で高速な内部 (水晶なし) 発振器です。これはデバイスでデフォルトのクロック ソースです。周波数は精度が±2%であり、24MHz～48MHzの間を4MHzステップで変更することができます。

IMO 周波数は CLK_IMO_SELECT レジスタを使用し、変更されます。デフォルト周波数は 24MHz です。

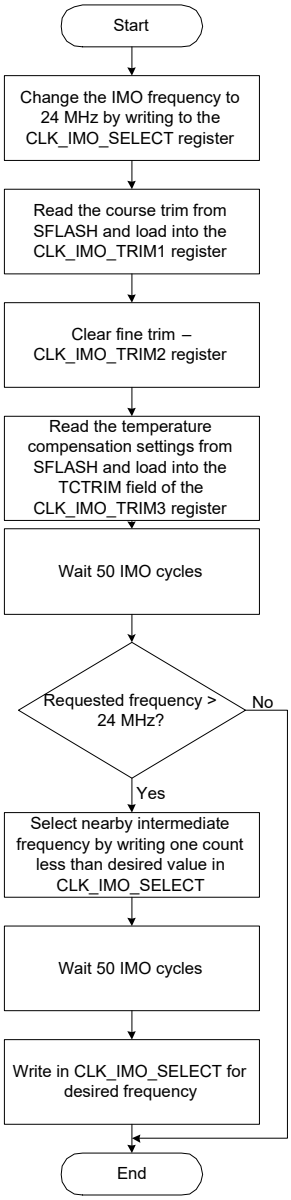
表 7-1. IMO 周波数

CLK_IMO_SELECT[2:0]	公称 IMO 周波数
0	24MHz
1	28MHz
2	32MHz
3	36MHz
4	40MHz
5	44MHz
6	48MHz

正確な IMO 周波数を取得するために、調整レジスタが提供されます – CLK_IMO_TRIM1 が 120kHz ステップ サイズの粗い調整を提供し、CLK_IMO_TRIM2 は、15kHz ステップ サイズの微調整であり、CLK_IMO_TRIM3 で TCTRIM フィールドは温度補償用です。調整設定は CLK_IMO_SELECT によって選択することができるすべての周波数用に、製造中に行われます。これらの調整設定は SFLASH に格納されます。

調整設定はデバイス スタートアップ時にロードされます ; しかし、ファームウェアは新しいトリム値のロードおよび実行時に周波数を変更することができます。IMO 周波数の変更は図 7-2 のアルゴリズムに従います。

図 7-2. IMO 周波数の変更



7.2.1.1 スタートアップ動作

リセット後、IMO は 24MHz 動作にコンフィギュレーションされます。スタートアップ プロセスの「ブート」時間には、調整値はフラッシュ メモリから読み込まれ、IMO はデータシートに指定した精度を達成するように設定されます。

7.2.1.2 プログラミング クロック (36MHz)

IMO はフラッシュをプログラムするために 48MHz に設定されます。これはフラッシュのプログラム／消去のタイミング目的用に、チャージ ポンプを駆動するために使用されます。

7.2.2 内部低速発振器 (ILO)

内部低速発振器は、外部コンポーネントなく動作し、公称 40kHz の安定したクロックを出力します。ILO の消費電力は比較的小さく、精度は良くありません。より高い精度の高周波クロックを使用して定期的に較正し、精度を向上させることができます。ILO はハイバネートとストップ モードを除きすべての消費電力モードで使用可能です。デバイスで、ILO はシステ

ムの低周波数クロック LFCLK として使用されます。ILO は低周波数のクロックを生成するための比較的不正確 (±60% の過電圧と温度) な発振器です。動作中に IMO を校正する場合、ILO は安定した温度および電圧によって、±10% の精度になります。ILO は、CLK_ILO_CONFIG レジスタの ENABLE ビットにより有効／無効化されます。

7.2.3 外部クロック (EXTCLK)

外部クロック (EXTCLK) は、MHz 帯のクロックで、PSoC 4 の指定したピンの信号から生成されます。このクロックは IMO のかわりにシステム高周波数クロック HFCLK の供給源として使用される場合があります。外部クロックの許容される周波数範囲は 1 ～ 48MHz です。デバイスは常に IMO を使用して起動しますが、外部クロックはユーザー モードで使用可能にする必要があります。そのためデバイスは外部クロックからクロック供給されるリセットから起動することはできません。

特定のピンを EXTCLK の入力として設定する場合、デジタル入力バッファーを有効にするために、ピンの駆動モードをハイインピーダンスのデジタル モードにする必要があります。詳細は 37 ページの I/O システムの章を参照してください。

7.2.4 時計用水晶発振器 (WCO)

PSoC デバイスは、32.768kHz の時計用水晶を駆動するための発振器が搭載されています。ILO と同様、WCO はすべての消費電力モードで使用可能です。WCO は、WCO_CONFIG レジスタの ENABLE ビットにより有効／無効化されます。

WCO は、WCO_CONFIG[0] ビットをセットすることで低消費電力モードに移行します。一方、ブロックはデバイスがディープスリープ モードに移行する時にのみ、低消費電力モード移行が発生する自動モードに移行できます。このモードは WCO_CONFIG[1] をセットすることで有効となります。ブロックは WCO_CONFIG[0] をセットすることで低消費電力モードに移行される場合、自動モードは無視されることにご注意ください。

正常モードおよび低消費電力モード間の違いはアンプ ゲインです。低消費電力モードは、電力削減を有効にするため、より低いアンプ ゲインを持つことが期待されます。2 つのモードのアンプ ゲインは WCO_TRIM レジスタで設定されます。

IMO は WCO にロックする機能をサポートします。WCO には IMO クロックの測定／比較、および IMO トリミング回路があります。WCO は ±2% のクロック精度に対応するために、デジタル位相同期回路 (PLL) 方式を搭載します。WCO 内の IMO トリミング回路は、WCO_CONFIG レジスタの DPLL_ENABLE ビットの設定によって有効／無効化されます。この機能を使用する場合、ユーザー ファームウェアは、WCO 有効時点から DPLL_ENABLE イベントが発生するまでの最小時間を 500ms 確保する必要があります。

7.3 クロック分配

図 7-1 に示すように PSoC 4 のクロックは生成され、あらゆるデバイスに分配されます。この分配コンフィギュレーションオプションは次のとおりです。

- HFCLK 入力の選択
- LFCLK 入力の選択
- SYSCLK プリスケーラのコンフィギュレーション
- ペリフェラル分周器のコンフィギュレーション

7.3.1 HFCLK 入力の選択

PSoC 4 の HFCLK の 2 つの入力オプションは、IMO と EXTCLK です。HFCLK 入力は、表 7-2 に示すように、CLK_SELECT レジスタの HFCLK_SEL ビットを使用することで選択されます。

表 7-2. HFCLK 入力選択ビット HFCLK_SEL

名称	説明
HFCLK_SEL[2:0]	HFCLK 入力クロックの選択 0: IMO。IMO を HFCLK の信号源として使用 1: EXTCLK。EXTCLK を HFCLK の信号源として使用 2 ～ 7: 予約済み。使用禁止

プレ分周器は、デバイスのピーク電流を制限するために HFCLK 用に設けられています。分周器オプションは CLK_SELECT レジスタの HFCLK_DIV ビットを使用して設定する 2、4 および 8 です。デフォルト分周値は 4 です。

7.3.2 LFCLK 入力の選択

ILO のみを PSoC 4000S デバイスで LFCLK に供給することができます。

7.3.3 SYSCLK プリスケーラのコンフィギュレーション

SYSCLK プリスケーラによりデバイスは HFCLK を分周して SYSCLK とすることが可能になります。これによりペリフェラル クロックとシステム クロックの周波数を整数倍の関係から外すことができます。SYSCLK は HFCLK から生成されたデバイス内の他のクロックより高速である必要があります。SYSCLK プリスケーラは、HFCLK を 2^0 = 1 から 2^7 = 128 の範囲内の2のべき乗の値で分周できます。プリスケーラの分周値は表 7-3に示すように、CLK_SELECTレジスタのSYSCLK_DIV ビットにより設定されます。プリスケーラの初期設定分周値は 1 です。

表 7-3. SYSCLK プリスケーラ分周値ビット SYSCLK_DIV

名称	説明
SYSCLK_DIV[3:0]	SYSCLK プリスケーラ分周値 0: SYSCLK = HFCLK 1: SYSCLK = HFCLK/2 2: SYSCLK = HFCLK/4 3: SYSCLK = HFCLK/8 4: SYSCLK = HFCLK/16 5: SYSCLK = HFCLK/32 6: SYSCLK = HFCLK/64 7: SYSCLK = HFCLK/128

7.3.4 ペリフェラル クロック分周器のコンフィギュレーション

PSoC 4 は 8 個のクロック分周器 (6 個の 16 ビット クロック分周器 と 2 個の 16.5 ビット分数クロック分周器) を持っています。分数クロック分周器により、クロック除数には 0 ～ 31/32 の分数があり得ます。分数分周器の出力周波数のための式は Fout = Fin / (INT16_DIV + (FRAC5_DIV/32)) です。例えば、16.5 ビット分数クロック分周器の整数分周値が 2 の場合 (INT16_DIV=3、FRAC5_DIV=0)、その分周器は 48MHz の HFCLK から 16MHz のクロックを生成します。例えば 16.5 ビット分数クロック分周器の整数分周値が 3 (INT16_DIV=3、FRAC5_DIV=0) の場合、その分周器は 48MHz の HFCLK から 12MHz のクロックを生成します。16.5 ビット分数クロック分周器の整数分周値が 2 (INT16_DIV=3) で、分数分周値が 16 (FRAC5_DIV=16) の場合、48MHz の HFCLK から 13.7MHz のクロックを生成します。13.7MHz クロックの周期はすべて等しい訳ではありません。その半数が 3HFCLK サイクルで、残りの半数が 2 HFCLK サイクルです。

分数分周器は UART/SPI シリアル インターフェースなど高精度なクロックが求められる場合に有用です。クロック周期のジッタが 1 HFCLK サイクルのため、低ジッタ クロックが求められる場合には分数分周器を使用しません。

各 16 ビット整数クロック分周器の分周値は PERI_DIV_16_CTLx レジスタにて設定されますが、4 個の 16.5 ビット分数クロック分周器のは PERI_DIV_16_5_CTLx レジスタにて設定されます。表 7-4 と表 7-5 はこれらのレジスタのコンフィギュレーションを示しています。

表 7-4. 非分数ペリフェラル クロック分周器コンフィギュレーション レジスタ PERI_DIV_16_CTLx

ビット	名称	説明
0	ENABLE_x	分周器が有効。ENABLE コマンドの実行結果として、HW はこのフィールドを「1」に設定。DISABLE コマンドの実行結果として、HW はこのフィールドを「0」に設定。
23:8	INT16_DIV_x	(1+INT16_DIV) の値で分周。[1、65536] 範囲内の整数分周が実行可能

表 7-5. 分数ペリフェラル クロック分周器 コンフィギュレーション レジスタ PERI_DIV_16_5_CTLx

ビット	名称	説明
0	ENABLE_x	分周器が有効。ENABLE コマンドの実行結果として、HW はこのフィールドを「1」に設定。DISABLE コマンドの実行結果として、HW はこのフィールドを「0」に設定。
7:3	FRAC5_DIV_x	(FRAC5_DIV/32) 分数比分周。[0、31/32] 範囲内の分数比分周を実行可能にする。 クロック周期の一部が他のクロック周期に比べ 1「clk_hf」サイクル長くなる場合があるため、分数比分周によりクロック ジッタを生じることに注意
23:8	INT16_DIV_x	(1+INT16_DIV) の値で分周。[1、65536] 範囲内の分周が実行可能

各分周器は PERI_DIV_CMD レジスタにより有効になります。このレジスタは、すべての 16 個の整数分周器と 4 個の分数分周器に対して、コマンド レジスタとして機能します。PERI_DIV_CMD レジスタのフォーマットは次の通りです。

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	Enable	Disable															PA_SEL_TYPE		PA_SEL_DIV						SEL_TYPE		SEL_DIV					

SEL_TYPE フィールドは、設定される分周器の種類を示しています。このフィールドを「1」に設定すると 16 ビット整数分周器となり、「2」に設定すると 16.5 ビット分数分周器となります。

SEL_DIV フィールドは設定される分周器数を指定します。整数分周器の場合、この値は 0 ～ 15 になります。分数分周器の場合、このフィールドの値は 0 ～ 3 になります。SEL_TYPE = 63 と SEL_TYPE = 3 である場合、いずれの分周器も指定しません。

(PA_SEL_TYPE、PA_SEL_DIV) フィールドの対により、ある分周器と他の分周器の位相を揃えることができます。PA_SEL_DIV は位相を揃えるデバイスを指定します。有効な分周器はいずれも基準分周器として使用できます。PA_SEL_TYPE は位相を揃える分周器の種類を指定しています。PA_SEL_DIV = 63 と PA_SEL_TYPE = 3 の場合、HFCLK は基準クロックとして使用されます。

48MHz の HFCLK から、12MHz 分周クロック A と 8MHz 分周クロック B を生成するケースを検討します。クロック A は 16 ビットの整数分周器 0 を使用して、それを HF_CLK ((PA_SEL_TYPE, PA_SEL_DIV) が (3, 63)) クロックとの位相を揃えて生成され、DIV_16_CTL0 レジスタの INT16_DIV の値が 3 です。クロック B は整数分周器 1 を使用して、それをクロック A ((PA_SEL_TYPE, PA_SEL_DIV) が (1、0)) との位相を揃えて生成され、DIV_16_CTL1 レジスタの INT16_DIV の値が 5 です。これによりクロック B はクロック A と位相が揃います。これらのクロック周期の最小公倍数が 12 HFCLK サイクルのため、クロック A とクロック B は 12 HFCLK サイクルごとに位相が揃います。クロック B はクロック A と位相を揃えますが、それ自体の分周値のために、依然として HFCLK を基準クロックとして使用することにご注意ください。

PSoC 内の各ペリフェラル ブロックは、それに関係する固有のペリフェラル クロック (PERI#_CLK) を持っています。各ペリフェラル クロックはそれぞれ、既存のクロック分周器のいずれか 1 つから入力クロックを取得できる多重化入力があります。

表 7-6 は (図 7-1 に示すように) 該当するペリフェラル ブロックへのマルチプレクサ出力の接続を示しています 。いずれかのペリフェラル クロック分周器は表 7-7 に示すように、それらの該当する PERI_PCLK_CTLx レジスタを使用することで、特定のペリフェラルに接続することができます。

表 7-6. ペリフェラル クロック マルチプレクサ出力マッピング

PERI#_CLK	ペリフェラル
0	SCB0
1	SCB1
2	CSD
3	TCPWM0
4	TCPWM1
5	TCPWM2
6	TCPWM3

表 7-6. ペリフェラル クロック マルチプレクサ出力マッピング

PERI#_CLK	ペリフェラル
7	TCPWM4
8	スマート I/O
9	スマート I/O
10	LCD

表 7-7. プログラマブル クロック制御レジスタ - PERI_PCLK_CTLx

ビット	名称	説明
5:0	SEL_DIV	SEL_TYPE で指定した同じ種類の中から 1 個の分周器を指定。SEL_DIV が「4」と SEL_TYPE が「1」の場合、5 番目 (0 は 1 番目) の 16 ビットのクロック分周器は、マルチプレクサ出力に接続され、ペリフェラルのクロック clock_x を構成する。同様に SEL_DIV が「0」と SEL_TYPE が「2」の場合、1 番目の 16.5 ビットのクロック分周器はマルチプレクサ出力に接続される
7:6	SEL_TYPE	0: 使用禁止 1: 16.0 (整数) クロック分周器 2: 16.5 (分数) クロック分周器 3:

7.4 低消費電力モード動作

IMO、EXTCLK、HFCLK、SYSCLK およびペリフェラル クロックを含む高周波数クロックは、アクティブとスリープ モードでのみ動作します。ILO、WCO および LFCLK はすべての消費電力モードで動作します。

7.5 レジスター覧

表 7-8. クロック供給システムのレジスター覧

レジスタ名	説明
CLK_IMO_TRIM1	IMO トリム レジスタ - このレジスタは粗調整用の IMO 調整値を含む
CLK_IMO_TRIM2	IMO トリム レジスタ - このレジスタは微調整用の IMO 調整値を含む
CLK_IMO_TRIM3	IMO 調整レジスタ - このレジスタは IMO 用の温度補償および IMO 周波数の粗調整と微調整のステップ サイズを調整するための調整設定を含む
PWR_BG_TRIM1	バンドギャップ調整レジスタ - これらのレジスタはバンドギャップ基準の調整を制御し、デバイスの基準電圧調整を可能にする
PWR_BG_TRIM2	
CLK_ILO_CONFIG	ILO コンフィギュレーション レジスタ - このレジスタは ILO コンフィギュレーションを制御
CLK_IMO_CONFIG	IMO コンフィギュレーション レジスタ - このレジスタは IMO コンフィギュレーションを制御
CLK_SELECT	クロック選択 - このレジスタはクロック ツリー コンフィギュレーションを制御し、異なる信号源からシステム クロックを選択
WCO_CONFIG	WCO イネーブル。このレジスタは外部の時計用水晶発振器を有効／無効にする
PERI_DIV_16_CTLx	ペリフェラル クロック分周器制御レジスタ - これらのレジスタは、ペリフェラル クロック分周器のコンフィギュレーション、整数分周値の設定、および分周器の有効／無効化を行う
PERI_DIV_16_5_CTLx	ペリフェラル クロック分数分周器制御レジスタ - これらのレジスタは、ペリフェラル クロック分周器のコンフィギュレーション、分数分周値の設定、および分周器の有効／無効化を行う
PERI_PCLK_CTLx	プログラマブル クロック制御レジスタ - これらのレジスタはペリフェラルの入力クロックを選択するために使用される

8. 電源と電圧監視

PSoC® 4 は 1.71V ～ 5.5V の電源電圧で動作可能です。電圧範囲は以下の 2 つに分かれます。

- 内部レギュレータを利用する 1.80V ～ 5.50V 入力
- 直接供給を利用する 1.71V ～ 1.89V[1] 入力

各消費電力モードをサポートする 2 つの内部レギュレータ - アクティブ デジタル レギュレータ、ディープスリープ レギュレータ - 搭載されています

8.1 ブロック図

図 8-1. 電力システムのブロック図

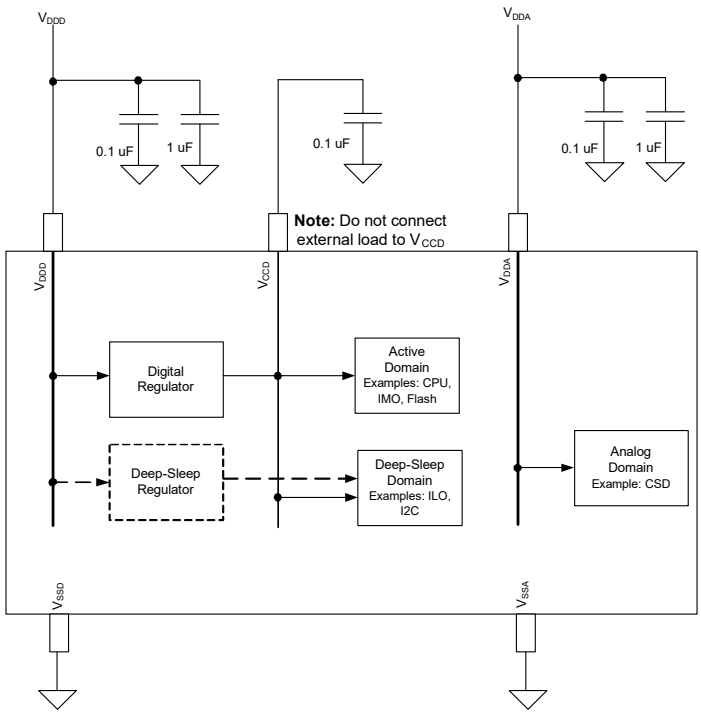


図 8-1 は電力システム図とすべての電力供給端子を示します。システムは、アクティブ モードで動作するデジタル回路用レギュレータがあります。アナログ レギュレータはありません。アナログ回路は V_{DDA} 電源により動作します。ディープスリープ モード用の個別レギュレータがあります。

電源電圧範囲は 1.71V ～ 5.5V で、すべての機能と回路がその範囲において動作します。デバイスには、非安定化外部電源供給と安定化外部電源供給モードの 2 つの電源供給モードがあります。

1. システム電源の電圧範囲が 1.80V ～ 1.89V である場合、直接供給と内部レギュレータの両方のオプションを使用することができます。どちらが使用されるかをユーザーのシステム機能に基づいて選択します。直接供給のオプションに対し、デバイスを破壊しないように電源電圧を 1.89V よりも高く印加しないことにご注意ください。内部レギュレータのオプションに対し、電源電圧を1.80Vよりも下げないで下さい。そうでなければ、レギュレータはオフとなります。

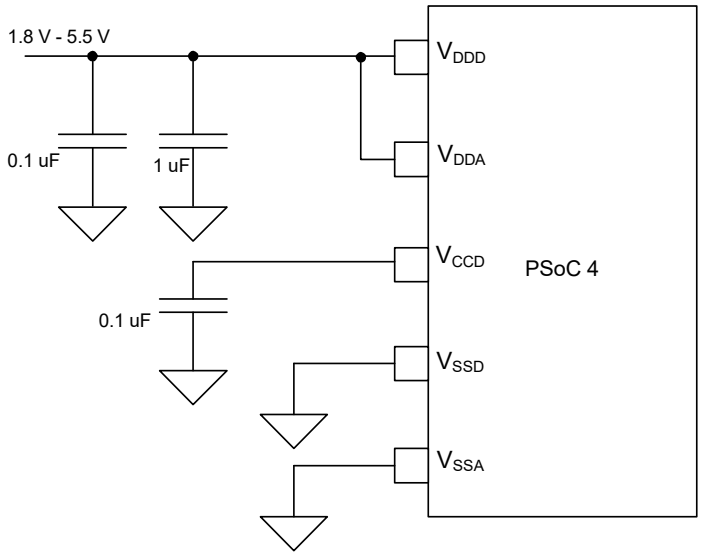
8.2 電源供給のシナリオ

下図はデバイスに電源供給される様々な方法を示しています。

8.2.1 単一 1.8V ～ 5.5V 非安定化電源

1.8V ～ 5.5V の供給が非安定化電源入力として使用する場合、図 8-2 に示すように、接続する必要があります。

図 8-2. 単一安定化 V_{DDD} 電源



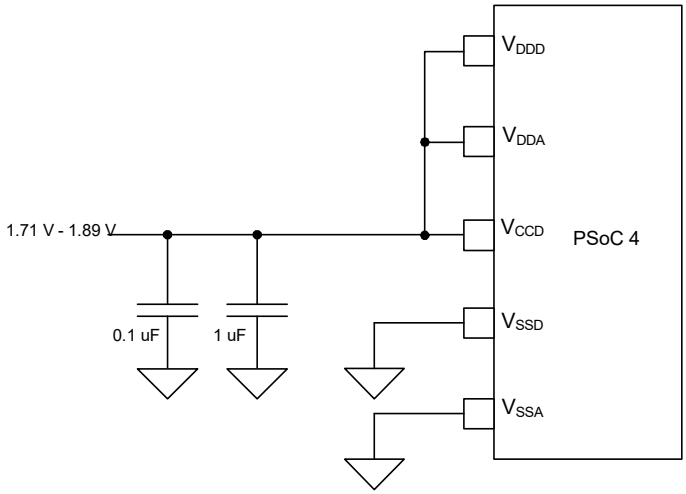
このモードでは、デバイスは 1.8V ～ 5.5V までの範囲で任意に、外部電源から供給されます。この範囲はバッテリ駆動で動作するようにも設計されています。例えば、チップは 3.5V から始まり、1.8V まで動作し、電池システムから電力を供給することができます。このモードでは、内部レギュレータは内部ロジックに供給します。V_{CCD} 出力は 0.1µF 外部セラミック コンデンサを経由し、グランドにバイパスされる必要があります。

バイパス コンデンサは V_{DDD} とグランド間の接続も必要です。この周波数範囲でのシステムの標準的な実践としては、互いに並列に接続する 1µF ～ 10µF のバルク コンデンサとそれより小さいセラミック コンデンサ (例えば、0.1µF) を使用します。これらは単なる経験則であり、重要なアプリケーションに対しては、最適なバイパスを得るために、設計の際には PCB レイアウト、リード インダクタンス、寄生バイパス コンデンサをシミュレーションする必要があることにご注意ください。

8.2.2 1.71V ～ 1.89V 安定化電源の直接接続

直接電源の構成で、V_{CCD} および V_{DDD}は 互いに短絡さ せ、1.71V～ 1.89Vの 電源に接続さ れます。この安定化電源は図 8-3 に示すようにデバイスに接続します。

図 8-3. 単一非安定化 V_{DDD} 電源



このモードで、V_{CCD} および V_{DDD} 端子は互いに短絡され、バイパスされます。内部レギュレータはファームウェアで無効にする必要があります。詳細は 66 ページの 8.3.1.1 アクティブ デジタル レギュレータを参照してください。

8.3 動作原理

図 8-1 に示すレギュレータは、デバイスの種々のドメインに電源を供給します。すべてのコア レギュレータは、V_{DDD} 電源端子から電力を取り込みます。アナログ回路は V_{DDA} 入力から直接電源供給されます。

8.3.1 レギュレータの概要

8.3.1.1 アクティブ デジタル レギュレータ

表 8-1. 消費電力モード別レギュレータの状態

モード	アクティブ デジタル レギュレータ	ディープスリープ レギュレータ
ディープスリープ	オフ	オン
スリープ	オン	オン
アクティブ	オン	オン

1.8V ～ 5.5V の外部電源の場合、アクティブ デジタル レギュレータはアクティブおよびスリープ モードで主なデジタル回路に電源を供給します。このレギュレータの出力は V_{CCD} 端子に接続され、外部にデカップリング コンデンサ (1µF X5R) を接続します。

1.8V 以下の電源の場合、V_{CCD} を直接供給する必要があります。この場合は、図 8-3 に示すように V_{CCD} および V_{DDD} を互いに短絡させる必要があります。

アクティブ デジタル レギュレータは、PWR_CONTROL レジスタ内の EXT_VCCD ビットをセットして無効にすることができます。この操作では直接電源モードでの消費電力が削減されます。アクティブ デジタル レギュレータは、アクティブおよびスリープ モードで使用できます

8.3.1.2 ディープスリープ レギュレータ

このレギュレータはディープスリープ モードで電源供給される ILO、WCO および SCB (I^2C/SPI) の回路および低消費電力コンパレータに電源供給します。ディープスリープ レギュレータはすべての消費電力モードで使用できます。アクティブおよびスリープ モードで、このレギュレータのメイン出力はアクティブ デジタル レギュレータ (V_{CCD}) に接続されます。

8.4 電圧監視

電圧監視システムはパワーオンリセット (POR)、電圧低下検出 (BOD) を含みます。

8.4.1 パワーオンリセット (POR)

POR 回路は、電源を投入してから電源電源が上昇する間にリセット パルスを発生します。POR 回路は V_{CCD} 電圧を監視します。POR 回路のトリップ ポイントは一般的にあまり正確ではありません。POR 回路は電源投入時に使用され、以後無効になります。

8.4.1.1 電圧低下検出 (BOD)

BOD 回路は、デバイスをリセットすることで、動作中またはデータ保持中の回路を安全ではない電源供給条件から保護します。BOD 回路は V_{CCD} 電圧を監視します。BOD 回路は、電圧変動が安全な動作に必要な最小 V_{CCD} 電圧を下回った場合にリセットを生成します (詳細はデバイス データシートを参照してください) 。システムは電源電圧が有効な範囲に回復するまでリセット状態を継続します。

信頼性の高いデバイス動作を保証するために、ウォッチドッグ タイマーはすべてのデザインで使用する必要があります。ウォッチドッグ タイマーは、CPU 機能を損なう可能性がある異常な電圧低下状態に対して保護します。詳細は 73 ページのウォッチドッグ タイマーの章 を参照してください。

8.5 レジスター覧

表 8-2. 電源と電圧監視レジスタの一覧

レジスタ名	説明
PWR_CONTROL	消費電力モード制御レジスタ – このレジスタにより、デバイスの消費電力モードとレギュレータの動作をコンフィギュレーションする

9. チップ動作モード

PSoC® 4 は 4 つの異なるモードでファームウェアを実行することができます。これらのモードは異なるハードウェア特権レベルでフラッシュおよび ROM での異なる場所からの実行を決定します。これらのモードの内 3 つだけが最終アプリケーションに使用されます。デバッグ モードは、ファームウェア開発で設計をデバッグするためだけに使用されます。

PSoC 4 の動作モードは以下の通りです :

- ブート
- ユーザー
- 特権
- デバッグ

9.1 ブート

ブート モードはデバイスがデバイス中の SROM にハードコードされた命令により設定される動作モードです。リセット終了後にデバイスにデバッグ取得のシーケンスが受信されなければ、このモードに入ります。ブート モードは特権モードです。このモードでは割込みが無効にされるため、ブート ファームウェアは割り込まれずにデバイスをセットアップすることができます。ブート モードでは、電源投入時に適切な動作を確保するためにハードウェア調整の設定がフラッシュからロードされます。ブート処理修了後、デバイスはユーザー モードに入り、フラッシュからのコード実行が始まります。フラッシュ内のこのコードは、デバイスをさらに設定する Psoc Creator IDE が自動的に生成する命令を含むことがあります。

9.2 ユーザー

ユーザー モードはフラッシュから通常のユーザー ファームウェアが実行される動作モードです。ユーザー モードでは SROM からのコードを実行することができません。このモードで実行されるファームウェアは Psoc Creator IDE に自動的に生成されるファームウェアとユーザーに書かれるファームウェアを含みます。自動的に生成されるファームウェアはファームウェア起動と通常動作の部分を制御することができます。ブート プロセスはそのタスクが終わったら、このモードへ制御権を移転します。

9.3 特権

特権モードはデバイスの ROM に保存される特別なサブルーチンの実行を可能にする動作モードです。これらのサブルーチンはユーザーによる修正が不可で、割り込みや観察されることを嫌う独自のコードを実行するために使用されます。特権モードではデバッグ処理は不可能です。

CPU はシステム コールを実行することで特権モードに移行することができます。システム コールを実行する方法の詳細については 183 ページの「システム コールの実行」を参照してください。このモードを終了すると、デバイスはユーザー モードに戻ります。

9.4 デバッグ

デバッグ モードは PSoC 4 の操作パラメーターの観察を可能にする動作モードです。このモードは開発中にファームウェアをデバッグするために使用されます。規定の接続時間内に SWD デバッガーがデバイスに接続すると、デバイスはデバッグ モードに入ります。これはデバイス リセットの間に発生します。デバッグ モードは PSoC Creator や Arm MDK などの IDE がファームウェアをデバッグすることを可能にします。デバッグ モードはオープン モード (4 つの保護モードの 1 つ) にあるデバイスのみで利用可能です。デバッグ インターフェースの詳細については 175 ページのプログラムおよびデバッグ インターフェースの章を参照してください。

保護モードの詳細については 81 ページのデバイス セキュリティの章を参照してください。

10. 消費電力モード

PSoC® 4 はアプリケーションでの平均消費電力を最小にするため 3 つの消費電力モードを提供します。消費電力の大きい順に電力モードは以下の通りです :

- アクティブ
- スリープ
- ディープスリープ

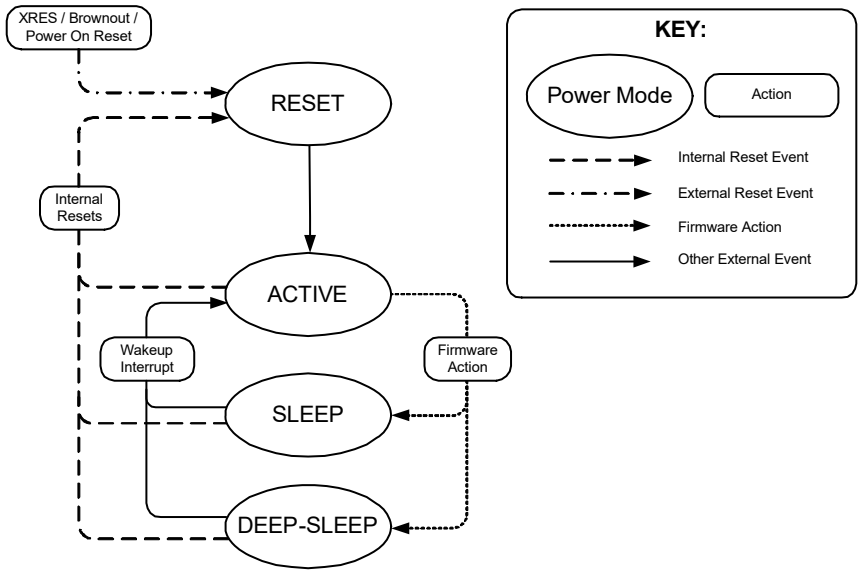
アクティブ、スリープおよびディープスリープは Arm 標準による定義された電力モードであり、Arm CPU によりサポートされます。

それぞれの電力モードの消費電力は以下の方法で制御されます :

- ペリフェラルを有効／無効
- 内部レギュレータの電源供給をオン／オフ
- クロック ソースの電源供給をオン／オフ
- PSoC 4 の他の部分の電源供給をオン／オフ

図 10-1 に消費電力モードの状態遷移図を示します。

図 10-1. 消費電力モードの状態遷移図



注 : Arm はディープスリープ モードを「SLEEPDEEP」と名づけます。

表 10-1 は PSoC 4 に用意された電力モードについて説明します。

表 10-1. PSoC 4 消費電力モード

消費電力モード	説明	移行条件	復帰ソース	アクティブクロック	復帰動作	使用可能なレギュレータ
アクティブ	主要な動作モード。すべてのペリフェラルは利用可能 (プログラム可能)	他の消費電力モードから復帰、内部と外部がリセット、電圧低下、パワー オン リセット	該当なし	すべて (プログラム可能)	該当なし	すべてのレギュレータが使用可能。外部のレギュレータ機能を使用している場合は、アクティブ デジタル レギュレータを無効にすることが可能
スリープ	CPU はスリープ モードに入り、SRAM は保持状態になる。すべてのペリフェラルを使用可能 (プログラム可能)	マニュアル レジスタ書き込み	任意の有効な割り込み	すべて (プログラム可能) CPU クロックを除く	割り込み	すべてのレギュレータが使用可能。外部のレギュレータ機能を使用している場合は、アクティブ デジタル レギュレータを無効にすることが可能
ディープスリープ	すべての内部電源はディープスリープ レギュレータにより駆動される。IMO と高速ペリフェラルはオフになる。低速 (32kHz) のクロックのみが使用可能 低速、非同期あるいは低消費電力アナログ ペリフェラルの割り込みにより復帰が可能	マニュアル レジスタ書き込み	GPIO 割り込み、低消費電力コンパレータ、SCB ウォッチドッグ タイマー	ILO (40kHz)、WCO (32kHz)	割り込み	ディープスリープ レギュレータ

表 10-1 に記載した復帰ソースに加え、外部リセット (XRES) と電圧低下リセットはいずれかの電力モードからアクティブ モードにデバイスを移行させます。

10.1 アクティブ モード

アクティブ モードは PSoC デバイスの主要な電力モードです。このモードはデバイスのサブシステム／ペリフェラルを全て使用するための前提となります。このモードでは CPU が動作し、すべてのペリフェラルが給電されます。ファームウェアは電力消費量を削減するために、使用していないペリフェラルを無効にすることがあります。

10.2 スリープ モード

このモードは CPU の消費電力を重視する電力モードです。このモードでは、Cortex-M0+ CPU がスリープ モードに入り、CPU のクロックが無効になります。低消費電力を達成するために、デバイスは CPU がアイドリングすると頻繁にこのモードに移行します。ペリフェラルについてはアクティブ モードと同じです。任意の有効な割り込みにより、スリープ モードから復帰させることができます。

10.3 ディープスリープ モード

ディープスリープ モードでは、CPU、SRAM および高速な回路がデータ保持状態に入ります。高周波数クロック (HFCLK と SYSCLK を含む) は無効になります。任意で内部低速発振器 (40kHz) をオンのままにし、低速ペリフェラルを動作させ続けることもできます。クロックを必要としない、または外部インターフェースからクロックを受けるデジタル ペリフェラル (例 : I^2C スレーブ) は動作し続けます。低速、非同期あるいは低消費電力アナログ ペリフェラルの割り込みにより、ディープスリープ モードから復帰させることができます。

利用可能な復帰ソースを表 10-3 に載せています。

10.4 消費電力モードの概要

表 10-3 はそれぞれの低消費電力モードで有効にすることができるペリフェラルについて説明します。表 10-3 は各消費電力モードで利用可能な復帰ソースについて説明します。

表 10-2. 利用可能なペリフェラル

ペリフェラル	アクティブ	スリープ	ディープスリープ
CPU	使用可能	保持[a]	保持
SRAM	使用可能	保持	保持
高速ペリフェラル	使用可能	使用可能	保持
低速ペリフェラル	使用可能	使用可能	使用可能 (オプション)
内部主発振器 (IMO)	使用可能	使用可能	使用不可
内部低速発振器 (ILO、40kHz)	使用可能	使用可能	使用可能 (オプション)
非同期ペリフェラル (内部クロックで動作しないペリフェラル)	使用可能	使用可能	使用可能
パワーオン リセット、電圧低下検出	使用可能	使用可能	使用可能
アナログ MUX バス接続	使用可能	使用可能	使用可能
低消費電力コンパレータ	使用可能	使用可能	使用可能
GPIO 出力状態	使用可能	使用可能	使用可能

a. ペリフェラルのコンフィギュレーションと状態は保持されます。ペリフェラルはデバイスがアクティブ モードへ移行しても動作し続けます。

表 10-3. 復帰ソース

消費電力モード	復帰ソース	ウェイクアップ動作
スリープ	任意の有効な割り込みソース	割り込み
ディープスリープ	GPIO 割り込み	割り込み
	I2C アドレス一致	割り込み
	ウォッチドッグ タイマー	割り込み／リセット
	低消費電力コンパレータ	割り込み

注 : 表 10-3 に記載した復帰ソースに加え、外部リセット (XRES) と電圧低下リセットはいずれかの消費電力モードからアクティブ モードにデバイスを移行させます XRES と電圧低下は完全なシステムの再起動をトリガーします。凍結された GPIO を含むすべての状態を失います。この場合、復帰の原因をデバイスが再起動した後で読み込むことはできません。

10.5 低消費電力モードへの移行および終了

Cortex-M0+ (CM0+) からの「割り込み待機 (WFI)」命令はスリープ およびディープスリープ モードへの移行をトリガーします。Cortex-M0+ は最も低い優先度の ISR が終了するまで低消費電力モードへの移行を遅延することができます (CM0 システム制御レジスタの SLEEPONEXIT ビットがセットされた場合)。

スリープ およびディープスリープ モードへの移行は、CM0P システム制御レジスタ (CM0P_SCR) 上の SLEEPDEEP フラグによって制御されます。

- WFI 命令が実行され、SLEEPDEEP = 0 の時にスリープに移行します。
- WFI 命令が実行され、SLEEPDEEP = 1 の時にディープスリープ モードに移行します。

PWR_CONTROL レジスタの LPM READY ビットは、ディープスリープ レギュレータの状態を表します。ファームウェアがレギュレータの準備が整う前にディープスリープ モードへ遷移する場合、まず PSoC 4 はスリープ モードへ遷移し、レギュレータの準備が整った後ディープスリープ モードへ遷移します。この動作はハードウェアで自動的に行います。

スリープおよびディープスリープ モードでは、ペリフェラルの選択が有効になり (表 10-3 を参照してください)、ファームウェアは関連する割り込みを有効にすることができます。有効にされた割り込みにより、低消費電力モードからアクティブ モードに復帰させることができます。さらに、任意の RESET はアクティブ モードにシステムを復帰させます。詳細は 27 ページの割り込みと 78 ページのリセット システムを参照してください。

10.6 レジスター覧

表 10-4. 電力モードのレジスター覧

レジスタ名	説明
CM0P_SCR	システム制御 ー システム制御データを設定または回復
PWR_CONTROL	消費電力モード制御 ー デバイス電力モードの詳細を設定し、現在の状態を監視

11. ウォッチドッグ タイマー

ウォッチドッグ タイマー (WDT) は、ファームウェアが予期しない実行経路へのイベント、または CPU 機能を損なう電圧低下のイベントでデバイスを自動的にリセットするために使用されます。WDT は ILO が生成する LFCLK により動作します。リセットを回避するためにファームウェアで定期的にタイマーをクリアする必要があります。放置するとタイマーが終了し、デバイス リセットを発生させます。WDT は省電力モードでは割り込みソースまたは復帰ソースとして使用されます。

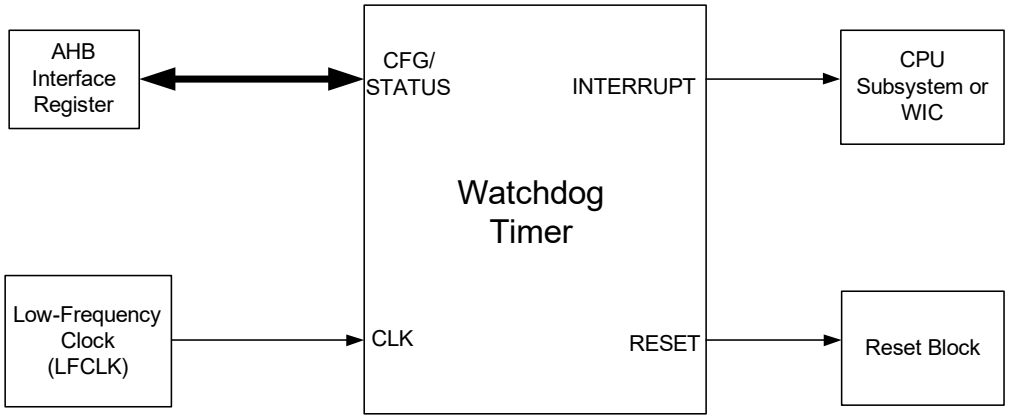
11.1 特長

WDT は以下の特長があります :

- コンフィギュレーション可能な経過時間の後にシステムリセットを生成可能
- アクティブ、スリープ、ディープスリープの消費電力モードで定期的な割り込み／復帰トリガーを生成可能
- 16 ビット フリーランニング カウンター機能

11.2 ブロック図

図 11-1. ウォッチドッグ タイマー ブロック図



11.3 動作原理

ファームウェアで定期的にクリアされない限り、WDT は 3 番目の WDT 一致イベントでデバイスのハードウェア リセットをアサートします。WDT は、16 ビット分解能の最大値を有するフリーランニング ラップアラウンド アップカウンターです。この分解能は設定可能であり、本節の後半で説明します。

WDT_COUNTER レジスタは WDT のカウント値を提供します。WDT は、WDT_COUNTER でのカウント値が WDT_MATCH レジスタに格納されている比較値と等しい時に割り込みを生成しますが、カウントを 0 にリセットしません。そして、WDT はカウントし続け、オーバーフローするまで (分解能を 16 ビットに設定されている時、0xFFFF の後) 0 にロール バックします。カウント値が再び比較値になると、別の割り込みが生成されます。カウンターが動作している時、一致カウントは変更可能なことにご注意ください。

WDT 割り込みが発生する度に、SRSS_INTR レジスタの WDT_MATCH という名前のビットが設定されます。この割り込みは、ウォッチドッグをリセットするために SRSS_INTR の WDT_MATCH ビットに「1」を書き込むことによりクリアする必要があります。ファームウェアが 2 つの連続割り込みで WDT をリセットしない場合、3 番目の一致イベントでは、ハードウェア リセットが発生します。

WDT_MATCH レジスタの IGNORE_BITS は、WDT カウンター全体の周期を減少させるために使用することができます。無視するビットは、破棄する必要がある MSB の数を指定することができます。例えば、IGNORE_BITS 値が 3 の場合、WDT カウンターは 13 ビット カウンターになります。詳細については、PSoC 4000S ファミリ : PSoC 4 レジスタ TRM での WDT_COUNTER、WDT_MATCH および SRSS_INTR レジスタの節を参照してください。

システム クラッシュからの保護に WDT が使用される時、ウォッチドッグをリセットするために WDT 割り込みと直接関係がないコードの部分から WDT 割り込みビットをクリアしなければなりません。そうしないと、ファームウェアのメイン関数がクラッシュしたり無限ループに陥ったとしても、WDT 割り込みベクタは変わらないままで、周期的にその WDT を送出し続けます。

システム クラッシュから保護するために WDT を使用する 1 番安全な方法は以下の通りです :

- 1 番長いファームウェアの遅延パスの間でも、ファームウェアが周期中に最低 1 回ウォッチドッグをリセットすることができるようにウォッチドッグ リセット周期を設定します。
- SRSS_INTR レジスタの WDT_MATCH に「1」を書き込むことにより、ファームウェア コードの本体で定期的に割り込みビットをクリアしてウォッチドッグをリセットします。
- WDTをクラッシュからシステムを保護するためのリセットソースとして使用する場合、WDT割り込みサービス ルーチン (ISR) でウォッチドッグをリセットすることをお勧めしません。したがって、一緒に WDT リセット機能と ISR を使用することは推奨されません。

WDT を割り込みタイマーとして使用するためには以下の手順に従ってください :

1. カウンター分解能を設定するために、WDT_MATCH レジスタに希望する IGNORE_BITS を書き込みます。
2. WDT_MATCH レジスタに希望する一致値を書き込みます。
3. 保留の WDT 割り込みをクリアするために SRSS_INTR の WDT_MATCH ビットをクリアします。
4. SRSS_INTR_MASK の WDT_MATCH ビットの設定によって、WDT 割り込みを有効にします
5. CM0_ISER レジスタ内のグローバル WDT 割り込みを有効にします (詳細は 27 ページの割り込みを参照してください)。
6. ISR では、WDT 割り込みをクリアし、既存の一致値に希望する一致値を追加します。これにより、カウンターは新しい一致値に達した時、他の定期的な割り込みが生成されます。

割り込みの詳細は 27 ページの割り込みを参照してください。

11.3.1 WDT の有効化と無効化

ウォッチドッグ カウンターは無効にできないフリーランニング カウンターです。しかし、これは WDT_DISABLE_KEY レジスタに「0xACED8865」キーを書き込むことにより、ウォッチドッグ リセットを無効にできます。このレジスタへの他の値の書き込みはウォッチドッグ リセットを有効にします。ウォッチドッグ システム リセットを無効にした場合、ファームウェアはシステム リセットを回避するウォッチドッグの定期的なリセットは必要ありません。ウォッチドッグ カウンターはまだ、割り込みソースまたはウェイクアップ ソースとして使用することができます。このカウンターを停止する唯一の方法は、CLK_ILO_CONFIG レジスタの ENABLE ビットをクリアし、ILO を無効にすることです。ウォッチドッグ リセットは ILO を無効にする前に無効にする必要があります。それ以外の場合、ILO を無効にする任意のレジスタへの書き込みは無視されます。ウォッチドッグ リセットの有効化により、自動的に ILO は有効になります。

注 : 以下の場合、WDT リセットの無効化は推奨されません :

- ファームウェア クラッシュに対して保護を必要とする
- 電源供給が CPU 機能を損なうような突然の電圧低下を起こし得る

11.3.2 WDT 割り込みと低消費電力モード

ウォッチドッグ カウンターはアクティブ モードで CPU へまたはディープスリープモードで復帰割り込みコントローラー (WIC) へ割り込み要求を送信することができます。ウォッチドッグ カウンターは以下の通りに動作します :

- **アクティブ モード :** このモードでは WDT は CPU へ割り込みを送信することができます。CPU は割り込み要求をアクノリッジし、ISR を実行します。ISR がファームウェアに入った後に、割り込みをクリアする必要があります。
- **スリープまたはディープスリープ モード :** このモードでは CPU サブシステムの電源が遮断されます。よって WDT からの割り込み要求は WIC に直接に送信され CPU を復帰させます。CPU は割り込み要求をアクノリッジし、ISR を実行します。ISR がファームウェアに入った後に、割り込みをクリアする必要があります。

デバイスの消費電力モードは 69 ページの消費電力モードを参照してください。

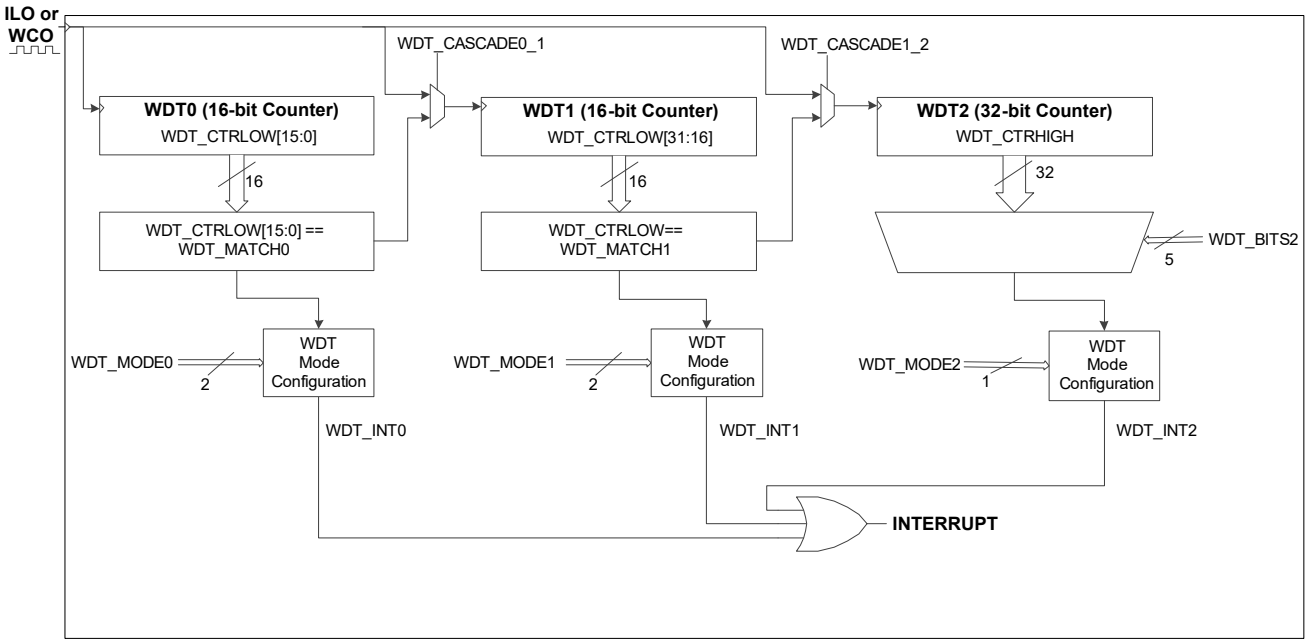
11.3.3 WDT リセット モード

RES_CAUSE レジスタの RESET_WDT ビットは WDT が生成するリセットを示します。このビットはクリアされるか、パワーオン リセット (POR)、電圧低下リセット (BOD) もしくは外部リセット (XRES) が発生するまでセットされたままになります。他のすべてのリセットはこのビットに影響を与えません。詳細については 78 ページのリセット システムを参照してください。

11.4 追加タイマー

WDT の他に、汎用使用のための 3 つの追加アップカウント タイマーがあります : WDT0、WDT1 および WDT2。これらの 3 つのタイマーは、WCO_WDT_CLKEN レジスタへの書き込みによって ILO または WCO のいずれかからクロック供給されます。これらのタイマーはアクティブ、スリープおよびディープスリープのモードで動作し、割込み生成ができます。

図 11-2. WDT 追加タイマー ブロック図



11.4.1 WDT0 と WDT1

これらは 16 ビット タイマーです。これは 2 つのコンフィギュレーションで動作します :

- フリーラン
- クリア オン マッチ (設定可能な周期)

フリーラン モードでは、タイマーは 16 ビット範囲でカウントします。65535 (216–1) に到達すると、タイマーは 0 にリセットし、再びカウントを開始します。クリア オン マッチ モードでは、WCO_WDT_MATCH レジスタの WDT_MATCH0 と WDT_MATCH1 に書き込まれたマッチ カウントは WDT0 と WDT1 のそれぞれの周期を決定します。タイマー カウントがマッチ値に到達すると、タイマーは 0 にリセットし、再びカウントを開始します。これらの 2 つのコンフィギュレーションのいずれかが、WCO_WDT_CONFIG レジスタの WDT_CLEAR0 および WDT_CLEAR1 ビットを使用して選択します。WDT_CLEARx に「1」を書き込むと、一致時クリアモードが選択されます。このビットに「0」を書き込むと、マッチ カウントでタイマーのクリアが無効にされ、フリーランニング モードが設定されます。デバイスをディープ スリープにする前に、マッチ カウントの更新後の少なくとも 1 入力クロック サイクルの遅延を確保します。

割込みは WCO_WDT_CONFIG レジスタの WDT_MODE ビットへの書き込みで、マッチまたはタイマー オーバフローで生成することができます。割込みの発生時、WCO_WDT_CONTROL レジスタの <t0/>WDT_INTx<t1/> ビットがセットされます。このビットは次の割込みトリガーを許可するためにファームウェアでクリアする必要があります。すべての 3 つのタイマーからの割込みは、CPU に単一のトリガー信号を生成するために論理和が取られます。割込みを起こしたタイマーを特定するために、WDT_INTx ビットを読み出します。

タイマーは WCO_WDT_CONTROL レジスタの WDT_ENABLEx ビットへの「1」の書き込みによって有効化します。これは有効となるまで 3 クロック サイクルかかることに注意してください。この期間中にこのビットを複数回トグルすることは推奨しません。タイマーが有効になった後、コンフィギュレーション レジスタ (WCO_WDT_CONFIG) への書き込みは推奨しません。タイマーの現在値は WDT_CTRLOW レジスタから読み出すことができます。これは WCO_WDT_CONTROL レジスタの WDT_RESETx ビットへの「1」の書き込みによってリセットすることができます。

11.4.2 WDT2

これは以下の違いを持って WDT0 と WDT1 と同様です :

- WDT2 は 32 ビットのカウントアップ タイマーです。
- カウント範囲が 0 ～ (2^{32}–1) のフリーランニング コンフィギュレーションのみをサポートします。
- 割込みは、カウント中に 32 ビットの 1 ビットがトグルするとトリガーします。ビット位置は WCO_WDT_CONFIG レジスタの 5 ビット WDT_BITS2 フィールドで設定します。「0」にセットすると、すべての入力クロックで割込みが発生します。「1」にセットすると、代替クロックで割込みが発生します。「31」にセットすると、2^{31} クロックごとに割込みが発生します。

11.4.3 カスケード

カスケード オプションは以下のとおりです :

- WDT0 と WDT1 のタイマーは WCO_WDT_CONFIG レジスタの WDT_CASCADE0_1 ビットへの書き込みによってカスケードすることができます。カスケードした時、WDT1 は WDT0 がマッチ カウントに到達した後、インクリメントします。
- WDT1 と WDT2 のタイマーは WCO_WDT_CONFIG レジスタの WDT_CASCADE1_2 ビットへの書き込みによってカスケードすることができます。カスケードした時、WDT2 は WDT1 がマッチ カウントに到達した後、インクリメントします。
- すべての 3 つのタイマーは、WDT_CASCADE0_1 および WDT_CASCADE1_2 のビットがセットされた時カスケードします。

11.5 レジスター覧

表 11-1. WDT レジスタ

レジスタ名	説明
WDT_DISABLE_KEY	値 0XACED8865 を書き込むと、WDT は無効です。他の値では WDT は普通に動作
WDT_COUNTER	WDT のカウント値を提供
WDT_MATCH	WDT の比較値を格納
SRSS_INTR	WDT のリセット回避をサービス

表 11-2. WDT レジスタ

レジスタ名	説明
WDT_DISABLE_KEY	値 0XACED8865 を書き込むと、WDT は無効。他の値では WDT は正常に動作
WDT_COUNTER	WDT のカウント値を提供
WDT_MATCH	WDT のマッチ値を維持
SRSS_INTR	WDT のリセット回避をサービス
WCO_WDT_CTRLOW	現時点の WDT0 および WDT1 タイマー値を保存
WCO_WDT_CTRHIGH	現時点の WDT2 タイマー値を保存
WCO_WDT_MATCH	WDT0 と WDT1 のマッチ カウントを保持
WCO_WDT_CONFIG	WDT0、WDT1 および WDT2 を設定 – クロック ソースの選択、フリー ランニングまたはクリア オン マッチの選択、割込み生成、およびカスケード
WCO_WDT_CONTROL	タイマーの有効化およびリセットに使用
WCO_WDT_CLKEN	タイマーで使用するクロック (ILO ／ WCO) を有効化

12. リセット システム

PSoC® 4 は、電源投入時にエラーのない動作を保証し、ユーザーが供給する外部ハードウェアまたは内部ソフトウエアによるリセット信号に応じてデバイスのリセットを可能にする複数のリセット タイプをサポートします。PSoC 4 は特定のリセット検出を可能にするために、ハードウェアも内蔵しています。

リセット システムには以下のリセットソースがあります :

- パワー オン リセット (POR): 電源電圧の立ち上がり時、デバイスをリセット状態に維持
- 電圧低下リセット (BOD): デバイスの動作中に電源電圧が動作仕様を下回る時にデバイスをリセット
- ウォッチドッグ リセット (WRES): ファームウェアの実行がウォッチドッグ タイマーに間に合わない場合にデバイスをリセット
- ソフトウェア リセット (SRES): 必要に応じてファームウェアを使用してデバイスをリセット
- 外部リセット (XRES): 外部電気信号を使用してデバイスをリセット
- 保護フォルト リセット (PROT_FAULT): 保護違反が発生する場合、デバイスをリセット

12.1 リセット ソース

次の節は PSoC 4 で利用可能なリセット ソースについて説明します。

12.1.1 パワーオン リセット

パワー オン リセットは、電源投入時にシステム リセットのために提供されます。POR は、電源電圧 V_{DDD} がデータシート記載の電圧に到達するまでデバイスをリセット状態に保持します。POR は電源投入時に自動的にアクティブになります。

POR イベントはリセット要因ステータスビットをセットしませんが、他のリセットソースを確認することで部分的に推測できる場合があります。他のリセット イベントが検出されない場合、POR、BOD または XRES によりリセットを発生させます。

12.1.2 電圧低下リセット

電圧低下リセットは、デジタル チップ電源電圧 V_{CCD} を監視し、V_{CCD} はデバイス データシートに記載される最低動作電圧を下回る場合リセットを発生させます。BOD はすべての電力モードで使用可能です。

12.1.3 ウォッチドッグ リセット

ウォッチドッグ タイマーがユーザーの指定する時間内にクリアされない場合、ウォッチドッグ リセット (WRES) が発生し、コード実行に問題があったことを検出します。この機能はデフォルトで有効です。これは WDT_DISABLE_KEY レジスタに「0xACED8865」を書き込むことによって無効にできます。

ウォッチドッグ リセットが発生する時、RES_CAUSE レジスタの RESET_WDT ステータス ビットがセットされます。このビットはクリアされるまで、または POR、XRES または BOD のリセットになるまでセットされたままです。例えば、デバイスのパワー サイクルの場合です。他のすべてのリセットはこのビットに影響を与えません。

詳細については 73 ページのウォッチドッグ タイマーを参照してください。

12.1.4 ソフトウェア初期化リセット

ソフトウェア初期化リセット (SRES) はソフトウェアでリセットを可能にするメカニズムです。「1」が SYSRESETREQ ビットに書き込まれる時、Cortex-M0+ のアプリケーション割り込みおよびリセット制御レジスタ (CM0P_AIRCR) により、デバイスは強制的にリセットされます。書き込みを有効にするために、値 05FA を CM0P_AIRCR レジスタの上位 2 バイトの位置に書き込む必要があります。つまり、リセットのために A05F0004 を書き込みます。

ソフトウェア リセットが発生する時、RES_CAUSE レジスタの RESET_SOFT ステータス ビットがセットされます。このビットはクリアされるまで、または POR、XRES または BOD のリセットになるまでセットされたままです。例えば、デバイスのパワー サイクルの場合です。他のすべてのリセットはこのビットに影響を与えません。

12.1.5 外部リセット

外部リセット (XRES) はユーザー供給リセットで、アサートされると直ちにシステム リセットを発生させます。XRES ピンはアクティブ LOW ***active low (アクティブ LOW)*** です。このピンの電圧が HIGH の場合、何も発生しませんが、LOW の時はリセットを発生させます。このピンはデバイス内で HIGH にプル アップされます。XRES はほとんどのデバイスで、専用ピンとして使用可能です。ピン配置の詳細については、デバイス データ シートのピン配置の節を参照してください。

XRES ピンはアクティブの間、デバイスをリセット状態に保持します。ピンが解放されると、デバイスは通常のブート シーケンスに従います。XRES の論理閾値および他の電気的特性については、デバイス データ シートの電気的仕様の節を参照してください。

XRES のイベントは、リセット要因ステータス ビットをセットしませんが、部分的に他のリセット ソースの不存在によって推測される場合があります。他のリセット イベントが検出されない場合、POR、BOD または XRES により、リセットを発生させます。

12.1.6 保護フォルト リセット

保護フォルト リセット (PROT_FAULT) は保護違反を検出し、違反が発生する場合デバイスをリセットさせます。保護フォルトの一例としては、特権コードの実行中にデバッグ ブレーク ポイントに達する場合です。特権コードの詳細については 68 ページの「特権」を参照してください。

保護フォルトが発生すると、RES_CAUSE レジスタの RESET_PROT_FAULT ビットがセットされます。このビットはクリアされるまで、または POR、XRES または BOD がリセットになるまでセットされたままです。例えば、デバイスのパワー サイクルの場合です。他のすべてのリセットはこのビットに影響を与えません。

12.2 リセット ソースの識別

デバイスがリセットを完了する時点で、直近またはそれ以前のリセット原因が分かるとその後の処理にしばしば都合の良い場合があります。これは主に RES_CAUSE レジスタを通じて可能になります。このレジスタは、いくつかのリセット ソースを示す特定のステータス ビットを持っています。RES_CAUSE レジスタはウォッチ ドッグ リセットの検出、ソフトウェア リセットおよび保護フォルト リセットの検出をサポートします。これは POR、BOD または XRES の発生を記録しません。ビットは関連したリセットが発生した時にセットされ、POR リセット、外部リセットまたは電圧低下検出の後でクリアされるかデータを失うまで、セットされたままです。

RES_CAUSE レジスタでリセットの原因を検出できない場合、それは記録されないリセットとデータが保持されないリセット (BOD、POR、XRES) のいずれかになります。これらのリセットはオンチップのリソースを使用して区別できません。

12.3 レジスター覧

表 12-1. リセット システム レジスター覧

レジスタ名	説明
WDT_DISABLE_KEY	値 0XACED8865 を書き込むと、WDT は無効です。他の値では WDT は普通に動作
CM0P_AIRCR	Cortex-M0+ アプリケーション割り込みおよびリセット制御レジスタ - このレジスタは他の Cortex-M0+ 機能の中で、ソフトウェア リセットを許可
RES_CAUSE	リセット原因レジスタ - このレジスタは直近のリセット原因を保存

13. デバイス セキュリティ

PSoC® 4 は無許可のアクセスあるいはコピーからユーザー設計を保護する多数の選択肢を提供します。デバッグ機能を無効にするとともにフラッシュ保護を有効にすることにより高レベルのセキュリティを提供します。

デバッグ回路はデフォルトで有効にされており、ファームウェアでのみ無効にすることができます。一度無効にした後に再び有効にするには、デバイス全体を消去し、フラッシュ保護を解除し、デバッグ処理を有効にする新しいファームウェアでデバイスをプログラムし直します。さらに、悪意を持ってデバイスを再プログラムすることによるフィッシング攻撃、またはフラッシュ プログラミング シーケンスを開始して割り込むことでセキュリティ システムを打破することが懸念されるアプリケーションについては、すべてのデバイス インターフェースを恒久的に無効にすることが可能です。インターフェースの恒久的な無効化は、設計者がデバイスにアクセスできなくなるため、ほとんどのアプリケーションにおいて推奨されません。フラッシュ列とチップ保護の説明と同様、詳細情報については PSoC 4100M、PSoC 4200M、PSoC 4200D、PSoC 4400、PSoC 4000S、PSoC 4100S、PSoC 4700S プログラミング仕様を参照してください。

注 : 最大限のデバイス セキュリティが有効な時にはすべてのプログラミング、デバッグ、テスト インターフェースが無効にされるため、デバイスの完全なセキュリティが有効にされた PSoC 4 デバイスは不具合解析ができない場合があります。

13.1 特長

PSoC 4 デバイスのセキュリティ システムは以下の特長があります :

- ユーザー選択可能な保護レベル
- 最も厳しい保護レベルでは、チップはテスト／デバッグの通信ができず、消去サイクルに入れないように「ロック」されます。消去サイクルへの割り込みは、ハッカーがチップを未定義の状態にして観察のためにチップを開く方法として知られています。
- マスク不可能な割り込み (NMI) を使用することにより、特権モードでの CPU 実行は可能です。特権モードでは、セキュリティ リークを発生させる割り込み命令による予期しないリターンを回避するために NMI はアサートされたままです。

更に、デバイスは個別のフラッシュ列データの保護も提供します。

13.2 動作原理

13.2.1 デバイス セキュリティ

CPU は通常のユーザー モードまたは特権モードで動作し、デバイスは BOOT (ブート)、OPEN (オープン)、PROTECTED (保護)、KILL (キル) という 4 つの保護モードの 1 つで動作します。各モードは CPU ソフトウェアおよびデバッグの特定の機能を提供します。CPUSS_PROTECTION レジスタに書き込むことによってモードを変更できます。

- **BOOT モード** : デバイスがリセット状態を抜けると BOOT モードに入ります。デバイスの保護状態が、監視フラッシュから保護コントロール レジスタ (CPUSS_PROTECTION) にコピーされるまでデバイスはこのモードに留まります。このコピー動作が行われるまでデバッグ アクセス ポートはストールされます。BOOT は、デバイスを指定した保護状態に設定するために必要な一時的なモードです。BOOT モードの間、CPU は常に特権モードで動作します。
- **OPEN モード** : これは工場出荷時のデフォルト モードです。CPU はユーザー モードまたは特権モードで動作します。ユーザー モードではフラッシュはプログラム可能で、デバッガ機能がサポートされます。特権モードではアクセス制限が適用されます。
- **PROTECTED モード** : ユーザーは OPEN モードから PROTECTED モードに変更できます。このモードはユーザー コードまたはメモリへのすべてのデバッグ アクセスを無効にします。PROTECTED モードでは、少数の限られたレジスタにのみアクセスできます。フラッシュを再プログラムするためのレジスタへのデバッグ アクセスは不可能です。フラッシュを完全に消去した後にのみ、デバイスを OPEN モードに戻すことができます。

- **KILL モード** : ユーザーは OPEN モードから KILL モードに変更できます。このモードはユーザー コードまたはメモリへのすべてのデバッグ アクセスは解除され、フラッシュは消去できないようになります。ほとんどのレジスタへのアクセスはまだ可能です。フラッシュを再プログラムするためのレジスタへのデバッグ アクセスは不可能です。デバイスは KILL モードの終了はできません。KILL モードにあるデバイスは不具合解析ができない場合があります。

13.2.2 フラッシュ セキュリティ

PSoC 4 デバイスはフラッシュ メモリへのアクセスを制御する柔軟なフラッシュ保護システムを備えています。この機能はユーザー独自のコードを保護するために利用されますが、フラッシュのブートローダ部分への予期しない上書きから保護するためにも使用されます。

フラッシュ メモリは列で構成されます。それぞれの列に保護レベルを割り当てることができます。表 13-1 を参照してください。フラッシュの保護レベルは、フラッシュの完全消去を実行することによってのみ変更できます。

詳細については 182 ページの不揮発性メモリ プログラムを参照してください。

表 13-1. フラッシュ保護レベル

保護設定	可能	不可
非保護	External read and write, Internal read and write	–
完全保護	外部読み出し[a] 内部読み出し	外部書き込み、 内部書き込み

a. 外部読み出し動作からデバイスを保護するためにデバイスの保護設定を PROTECTED に変更する必要があります。

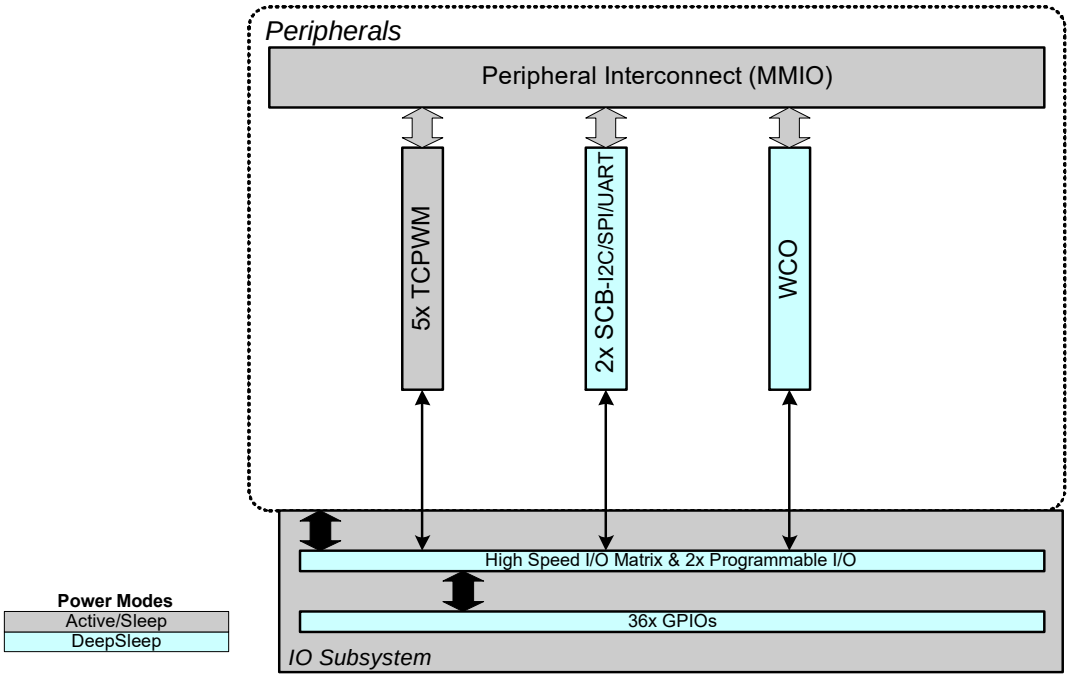
セクション D: デジタル システム

このセクションは次の章を含みます :

- 84 ページのシリアル通信ブロック (SCB)
- 128 ページのタイマー、カウンター、およびパルス幅変調器

トップ レベル アーキテクチャ

デジタル システム ブロック図



14. シリアル通信ブロック (SCB)

PSoC® 4 のシリアル通信ブロック (SCB) は SPI、UART、I²C の 3 つのシリアル インターフェース プロトコルに対応します。SCB は任意の時点で 1 つのプロトコルのみに対応します。PSoC デバイスは 2 つの SCB を備えています。

14.1 特長

このブロックは以下の機能をサポートします :

- Motorola、Texas Instruments、National Semiconductor のプロトコルと互換性のある標準的な SPI マスターと SPI スレーブ機能
- SmartCard リーダー、ローカル相互接続ネットワーク (LIN)、IrDA プロトコルと互換性のある標準的な UART 機能
- 標準的な I²C マスターとスレーブ機能
- 標準 LIN スレーブ機能は LIN V1.3 および LIN V2.1/2.2 仕様規格に準拠
- SPI と I2C 用の EZ モード、このモードでは CPU の介入なしに動作可能
- SPI と I2C プロトコル用の低消費電力 (ディープスリープ) 動作モード (外部のクロックを使用)

3 つのプロトコルはそれぞれ以下の節で説明します。

14.2 シリアル ペリフェラル インターフェース (SPI)

SPI プロトコルは同期シリアル インターフェース プロトコルです。デバイスはマスター モードまたはスレーブ モードで動作します。マスターはデータ転送を開始します。SCB は SPI プロトコルに対してシングルマスターマルチスレーブ トポロジをサポートします。個別のスレーブ セレクト ラインに対して複数のスレーブがサポートされます。

PSoC が 1 つ以上の SPI スレーブ デバイスに通信する必要がある時に SPI マスター モードを使用できます。PSoC が 1 つの SPI マスター デバイスに通信する必要がある場合に SPI スレーブ モードは使用可能です。

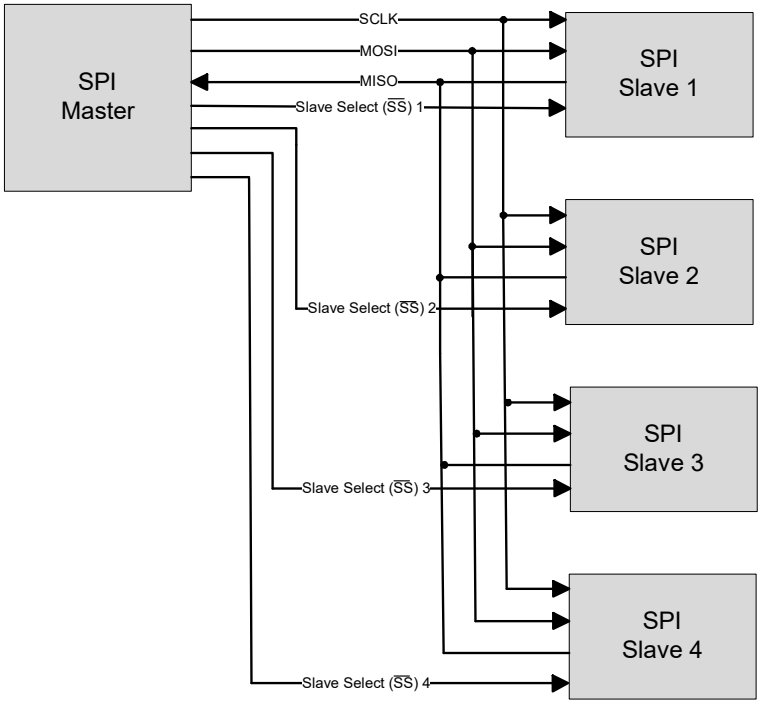
14.2.1 特長

- マスターとスレーブ機能をサポート
- 3 つの種類の SPI プロトコルに対応 :
  - Motorola 社 SPI: モード 0、1、2、3
  - Texas Instruments 社 SPI: モード 1 用のデータ フレームの同時インジケータと先行インジケータ機能付き
  - National Semiconductor (MicroWire) 社 SPI: モード 0
- 最大 4 本のスレーブ選択ラインをサポート
- データ フレーム サイズは 4 ビット～ 16 ビットにプログラム可能
- 割り込みまたはポーリングによる CPU インターフェース
- プログラマブルなオーバーサンプリング
- EZ 動作モードに対応 (イージー SPI プロトコル)
  - EZSPI モードでは CPU の介入なしに動作可能
- 外部からクロック供給されるスレーブ動作をサポート :
  - スレーブはアクティブ、スリープ、ディープスリープのシステム消費電力モードで操作

14.2.2 概要

図 14-1 に SPI マスターと 4 つのスレーブの例を示します。

図 14-1. SPI の例



標準 SPI インターフェースは以下の 4 つの信号を含んでいます。

- SCLK: シリアル クロック (マスターからのクロック出力、スレーブへの入力)
- MOSI: マスター アウト スレーブ イン (マスターからのデータ出力、スレーブへの入力)
- MISO: マスター イン スレーブ アウト (マスターへのデータ入力、スレーブからの出力)
- スレーブ選択 (SS): 通常はアクティブ LOW 信号 (マスターからの出力、スレーブへの入力)

簡単な SPI データ転送は次の動作を含みます : マスターはその SS ラインを駆動することでスレーブを選択してから、MOSI ライン上のデータと SCLK ライン上のクロックを駆動します。スレーブは MOSI ライン上のデータを取り込むために、コンフィギュレーションに応じて SCLK エッジのいずれかを使用します。また、マスターによって取り込まれる MISO ライン上のデータを駆動します。

デフォルトで SPI インターフェースは 8 ビット (1 バイト) のデータ フレーム サイズに対応します。データ フレーム サイズは 4 ビット～ 16 ビット範囲内の任意の値に設定できます。シリアル データは最上位ビット (MSb) ファーストまたは最下位ビット (LSB) ファーストの方式で送信されます。

SPI プロトコルの 3 つの種類は SCB によってサポートされます :

- Motorola 社 SPI: これはオリジナルの SPI プロトコルです。
- Texas Instruments 社 SPI: オリジナルの SPI プロトコルの変形です。このプロトコルではデータ フレームは SS ライン上のパルスで識別されます。
- National Semiconductor 社 SPI: オリジナルの SPI プロトコルの半二重変形です。

14.2.3 SPI 動作モード

14.2.3.1 Motorola 社 SPI

オリジナルの SPI プロトコルは Motorola 社によって定義されました。これは全二重プロトコルです。$\overline{SS}$ ラインが「0」に維持される間に複数のデータ転送が行えます。したがって、スレーブ デバイスは個別のデータ フレームを分離するためにデータ転送の進捗を確認しなければなりません。データを送信しない時に、$\overline{SS}$ ラインは「1」に維持され、SCLK は通常、LOW にプルダウンされます。

Motorola 社 SPI モード

Motorola 社の SPI プロトコルは、データを MOSI と MISO ラインで駆動し取り込む方法に基づいて 4 つの異なるモードがあります。これらのモードはクロック極性 (CPOL) とクロック位相 (CPHA) によって決められます。

クロック極性はデータの送信中でない時、SCLK ラインの値を決定します。CPOL =「0」はデータの送信中でない時、SCLK が「0」であることを示します。CPOL = 「1」はデータの送信中でない時、SCLK が「1」であることを示します。

クロック位相はデータが駆動され取り込まれる時点を決めます。クロック エッジが立ち上がりエッジであるか立ち下がりエッジであるかに関わらず、CPHA=0 は立ち上がり (最初の) クロック エッジでサンプリングする (データを取り込む) ことを意味し CPHA=1 は立ち下り (2 番目の) クロック エッジでサンプリングすることを意味します。CPHA=0 の場合、データは最初のクロック サイクルの前のセットアップ時間で安定する必要があります。

- モード 0: CPOL = 0、CPHA = 0: データは SCLK の立ち下がりエッジで駆動されます。データは SCLK の立ち上がりエッジで取り込まれます。
- モード 1: CPOL = 0、CPHA = 1: データは SCLK の立ち上がりエッジで駆動されます。データは SCLK の立ち下がりエッジで取り込まれます。
- モード 2: CPOL = 1、CPHA = 0: データは SCLK の立ち上がりエッジで駆動されます。データは SCLK の立ち下がりエッジで取り込まれます。
- モード 3: CPOL = 1、CPHA = 1: データは SCLK の立ち下がりエッジで駆動されます。データは SCLK の立ち上がりエッジで取り込まれます。

図 14-2 に CPOL と CPHA の組み合わせに応じた MOSI/MISO データの駆動と取り込みを示します。

図 14-2. Motorola 社 SPI、4 モード

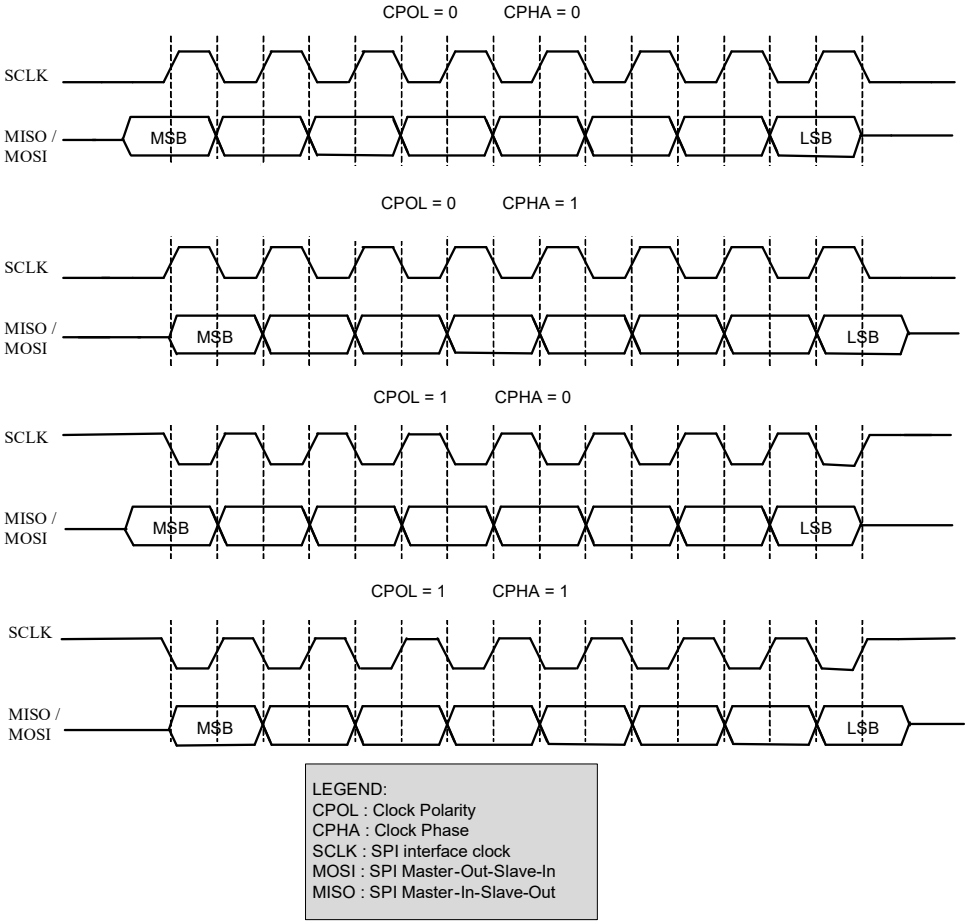
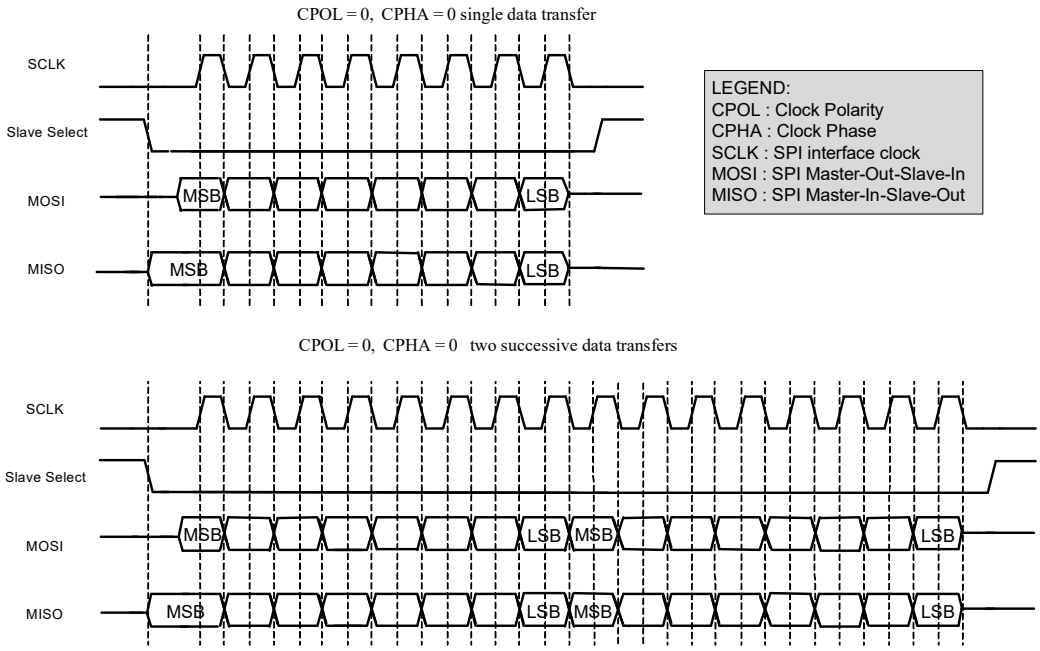


図 14-3 にモード 0 (CPOL = 0、CPHA = 0) での単一の 8 ビット データ転送と 2 つの連続した 8 ビット データ転送を示します。

図 14-3. Motorola 社 SPI のデータ転送例



SPI Motorola 社モード用の SCB の構成

SPI Motorola 社モード用に SCB を構成するには様々なレジスタ ビットを次の順序で設定してください :

1. SCB_CTRL レジスタの MODE ビット (ビット [25:24]) に「01」を書き込むことで SPI を選択
2. SCB_SPI_CTRL レジスタの MODE ビット (ビット [25:24]) に「00」を書き込むことで SPI Motorola 社モードを選択
3. SCB_SPI_CTRL レジスタの CPHA と CPOL フィールド (それぞれビット 2 とビット 3) に書き込むことで Motorola 社の動作モードを選択
4. 94 ページの「SPI の有効化と初期化」で述べたステップ 2 ～ステップ 4 を行う

PSoC Creator が GUI を利用してこれらのすべてを自動的に行うことにご注意ください。これらのレジスタの詳細は「PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

14.2.3.2 Texas Instruments 社 SPI

Texas Instruments 社の SPI プロトコルは $\overline{SS}$ 信号の使用を再定義します。このプロトコルはデータ転送の開始を示すために Motorola 社 SPI の場合のアクティブ LOW スレーブ選択信号ではなくこの信号を使用します。したがって、スレーブ デバイスは個別のデータ フレームを分離するためにデータ転送の進捗を確認する必要はありません。転送開始は単一ビット転送周期のアクティブ HIGH パルスで示されます。このパルスは最初のデータ ビットの送信の 1 サイクル前に発生するか、または最初のデータ ビットの送信と同時に発生します。TI 社の SPI プロトコルはモード 1 (CPOL = 0、CPHA = 1) のみをサポートします : データは SCLK の立ち上がりエッジで駆動され、SCLK の立ち下がりエッジで取り込まれます。

図 14-4 に単一の 8 ビット データ転送と 2 つの連続した 8 ビット データ転送を示します。SELECT パルスは最初のデータ ビットに先行します。2 番目のデータ転送の SELECT パルスが最初のデータ転送の最終データ ビットと同時に発生する点にご注意ください。

図 14-4. TI 社 SPI のデータ転送例

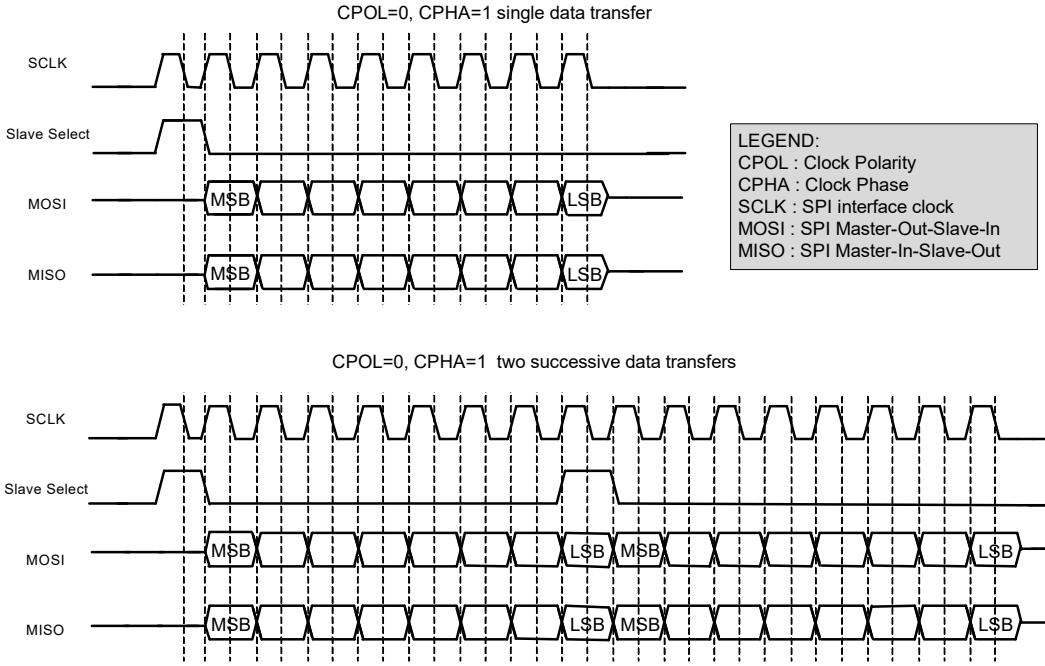
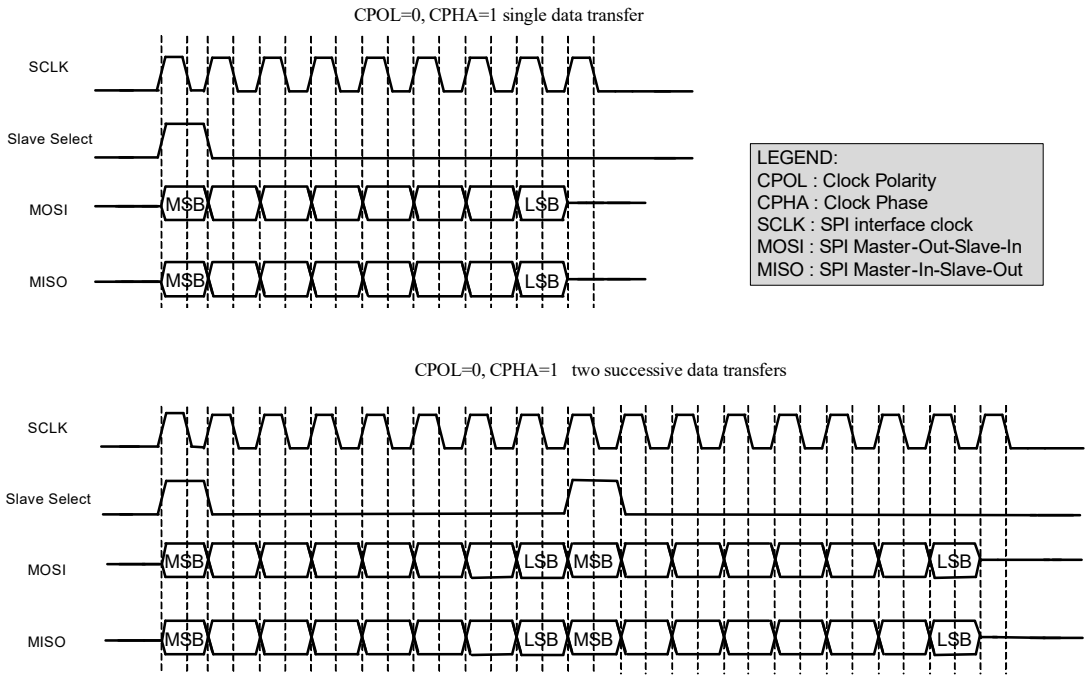


図 14-5 に単一の 8 ビット データ転送と 2 つの連続した 8 ビット データ転送を示します。SELECT パルスはフレームの最初のデータ ビットと同時に発生します。

図 14-5. TI 社 SPI のデータ転送例



SPI TI モード用の SCB の構成

SPI TI モード用に SCB を構成するには様々なレジスタ ビットを次の順序で設定してください :

1. SCB_CTRL レジスタの MODE ビット (ビット [25:24]) に「01」を書き込むことで SPI を選択
2. SCB_SPI_CTRL レジスタの MODE ビット (ビット [25:24]) に「01」を書き込むことで SPI TI モードを選択
3. SCB_SPI_CTRL レジスタの SELECT_PRECEDE フィールド (ビット 1) に書き込むことで TI の動作モードを選択 (「1」を書き込むと SELECT パルスが次のフレームの最初のビットに先行するように設定され「0」を書き込むと残りのモードとなります)
4. 94 ページの「SPI の有効化と初期化」で述べたステップ 2 ～ステップ 5 を行う

PSoC Creator が GUI を利用してこれらのすべてを自動的に行うことにご注意ください。これらのレジスタの詳細は「PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

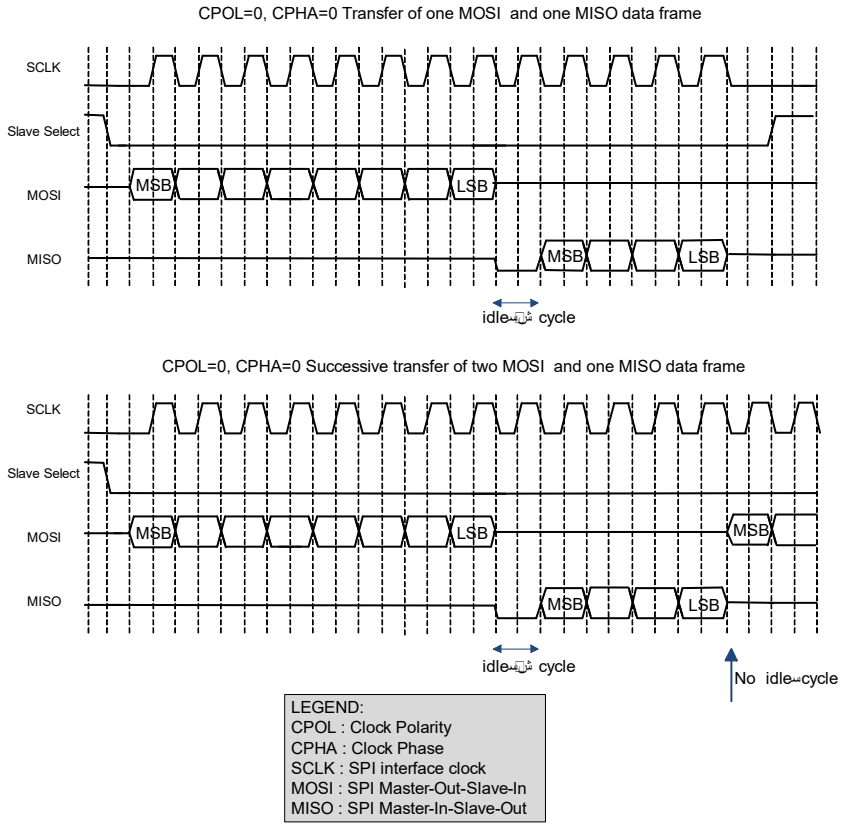
14.2.3.3 National Semiconductor 社 SPI

National Semiconductor 社の SPI プロトコルは半二重プロトコルです。送信と受信を同時に行わず、順番に行います。送信と受信のデータ サイズが異なることがあります。単一の「アイドル」ビット転送周期が送信と受信を分離します。ただし連続したデータ転送は「アイドル」ビット転送周期で分離されません。

National Semiconductor 社の SPI プロトコルはモード 0 のみをサポートします : データは SCLK の立ち下がりエッジで駆動され SCLK の立ち上がりエッジで取り込まれます。

図 14-6 に単一のデータ転送と 2 つの連続したデータ転送を示します。両方の場合、送信データ転送サイズは 8 ビットで受信データ転送サイズは 4 ビットです。

図 14-6. NS 社 SPI のデータ転送例



SPI NS モード用の SCB の構成

SPI NS モード用に SCB を構成するには様々なレジスタ ビットを次の順序で設定してください。

1. SCB_CTRL レジスタの MODE ビット (ビット [25:24]) に「01」を書き込むことで SPI を選択
2. SCB_SPI_CTRL レジスタの MODE ビット (ビット [25:24]) に「10」を書き込むことで SPI NS モードを選択
3. 94 ページの「SPI の有効化と初期化」で述べたステップ 2 ～ステップ 5 を行う

PSoC Creator がコンポーネント カスタマイザを利用してこれらのすべてを自動的に行うことにご注意ください。これらのレジスタの詳細は「PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

14.2.4 スレーブへのクロック供給用の SPI マスターの使用

通常の SPI マスター モードの送信では SCLK は SCB が有効にされかつデータが送信中の時にのみ生成されます。SCB が有効な限り SCLK ライン上にクロックを常に生成するように変更できます。スレーブが SPI 機能だけでなく他の機能でも SCLK を使用する時にこのオプションが使用されます。これはSCB_SPI_CTRLレジスタのSCLK_CONTINUOUS (ビット 5) に「1」を書き込むことで可能となります。

14.2.5 イージー SPI プロトコル

イージー SPI (EZSPI) プロトコルはすべてのモード (0、1、2、3) で動作する Motorola 社の SPI に基づいたものです。このプロトコルは個別のフレームのレベルで CPU の介入なしにマスターとスレーブ間の通信を可能にします。

EZSPIプロトコルはスレーブ デバイスに位置するメモリ アレイ (各エントリが 8 ビットを含む 32 エントリのアレイがサポートされる) をインデックスする 8 ビットの EZ アドレスを定義します。EZ アドレスの下位の 5 ビットはこの 32 個の位置をアドレス指定するために使用されます。すべてのEZSPIデータ転送は8ビットのデータ フレームを有します。

注 : SCB はバイト単位の書き込みが可能な 16 ワード * 16 ビットの SRAM である FIFO メモリを搭載しています。EZ と非 EZ 機能へのアクセス方法は異なります。非 EZ モードで FIFO は TXFIFO と RXFIFO に分けられます。それぞれは 8 エントリを持ち各エントリが 16 ビットです。エントリごとの 16 ビット幅はコンフィギュレーション可能なデータ幅に対応するために使用されます。EZ モードでは固定した 8 ビット幅データのみが使用されるため、これは単一の 32x8 ビットの EZFIFO として使用されます。

EZSPI は 3 つの転送タイプがあります。マスターからスレーブへの EZ アドレスの書き込み、マスターからアドレス指定したスレーブ メモリ位置へのデータの書き込み、アドレス指定したスレーブ メモリ位置からのマスターによる読み出しです。

14.2.5.1 EZ アドレス書き込み

EZ アドレスの書き込みは EZ アドレスを書き込むマスターの意図を示す MOSI ライン上のコマンド バイト (0x00) から始まります。その後、スレーブはコマンドが監視される (0xFE) または監視されない (0xFF) ことを示すために MISO ライン上に返信バイトを駆動します。MOSI 上の 2 番目のバイトは EZ アドレスです。

14.2.5.2 メモリ アレイ書き込み

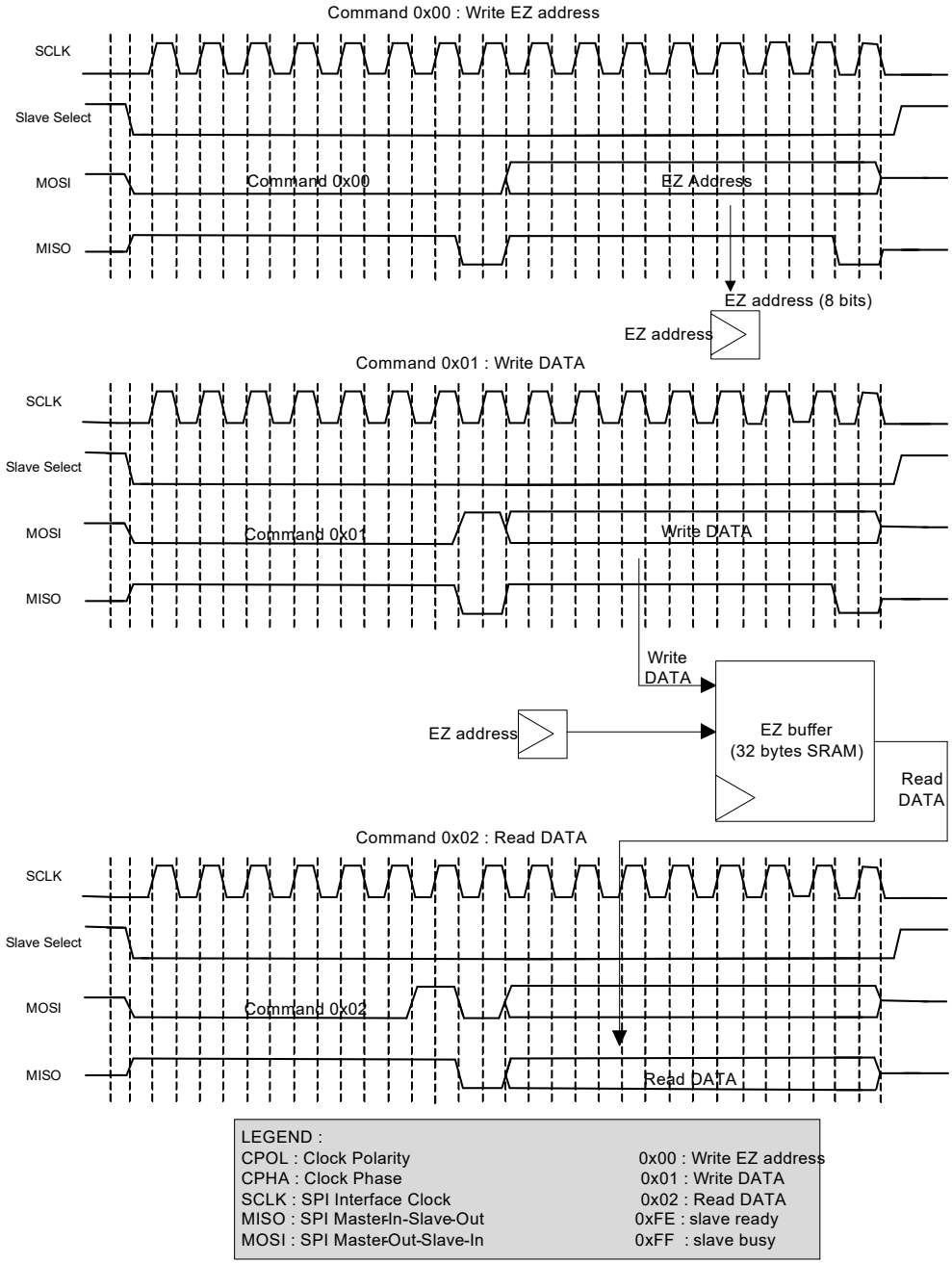
メモリ アレイ インデックスへの書き込みはメモリ アレイに書き込むマスターの意図を示す MOSI ライン上のコマンド バイト (0x01) から始まります。その後、スレーブはコマンドが登録された (0xFE) または非登録の (0xFF) であることを示すために MISO ライン上に返信バイトを駆動します。MOSI 上の追加書き込みのデータ バイトは通信した EZ アドレスによって示される位置でメモリ アレイに書き込まれます。バイトがメモリ アレイに書き込まれると EZ アドレスはスレーブによって自動的にインクリメントされます。EZ アドレスがメモリ エントリの最大数 (32) を超えるとその値を維持し、0 にラップ アラウンドしません。

14.2.5.3 メモリ アレイの読み出し

メモリ アレイ インデックスからの読み出しはメモリ アレイから読み出すマスターの意図を示す MOSI ライン上のコマンド バイト (0x02) から始まります。その後、スレーブはコマンドが登録 (0xFE) または非登録 (0xFF) である ことを示すために MISO ライン上に返信バイトを駆動します。MISO 上の追加読み出しのデータ バイトは通信した EZ アドレスによって示される位置でメモリ アレイから読み出されます。バイトがメモリ アレイから読み出されると、EZ アドレスはスレーブによって自動的にインクリメントされます。EZ アドレスがメモリ エントリの最大数 (32) を超えるとその値を維持し、0 にラップ アラウンドしません。

図 14-7 に EZSPI プロトコルでの EZ アドレスの書き込み、メモリ アレイへの書き込み、メモリ アレイからの読み出し動作を示します。

図 14-7. EZSPI の例



14.2.5.4 EZSPI モード用の SCB の構成

SCB はデフォルトで非 EZ の動作モードに構成されます。EZSPI モード用に SCB を構成するにはレジスタ ビットを次の順序で設定してください :

1. SCB_CTRL レジスタの EZ_MODE ビット (ビット 10) に「1」を書き込むことで EZ モードを選択
2. SCB_SPI_CTRL レジスタの CONTINUOUS ビットに「1」を書き込むことでトランスミッターに連続送信モードを使用
3. 94 ページの「SPI の有効化と初期化」で述べるステップ 2 ～ステップ 5 を行う

PSoC Creator がコンポーネント カスタマイザを利用してこれらのすべてを自動的に行うことにご注意ください。これらのレジスタの詳細は「 PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

14.2.6 SPI レジスタ

SPI インターフェースは表 14-1 に示す一連の 32 ビットのコントロールとステータス レジスタで制御されます。これらのレジスタの詳細は「 PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

表 14-1. SPI レジスタ

レジスタ名	動作
SCB_CTRL	SCB を有効にし、シリアル インターフェースのタイプ (SPI、UART、I^2C) を選択し、内部と外部クロック供給動作、EZ と非 EZ のモード動作を選択する
SCB_STATUS	EZ モードではこのレジスタは外部からクロック供給されるロジックが EZ メモリを使用している可能性があるかを示す
SCB_SPI_CTRL	SPI をマスターまたはスレーブとして構成し SPI プロトコル (Motorola、TI、National) と Motorola 社 SPI でのクロック ベースのサブモード (モード 0、1、2、3) を選択し、TI 社 SPI での SELECT 信号のタイプを選択。SPI がスレーブモードとして機能する場合、最初のチップ選択ピン SPI_SELECT [0] のみがスレーブモードで使用できる。
SCB_SPI_STATUS	SPI バスがビジーであるかを示し内部クロック供給モードで SPI スレーブ EZ アドレスを設定
SCB_TX_CTRL	データ フレーム幅を指定し送信の最初のビットが MSB であるか LSB であるかを指定
SCB_RX_CTRL	SCB_TX_CTRL レジスタと同じ機能を実行。ただし対象はレシーバー。メジアン フィルターが入力インターフェース ライン上に使用されるかどうかも決定
SCB_TX_FIFO_CTRL	トリガー レベルを指定しトランスミッター FIFO とシフト レジスタをクリアしトランスミッター FIFO のフリーズ動作を実行
SCB_RX_FIFO_CTRL	SCB_TX_FIFO_CTRL レジスタと同じ機能を実行。ただし対象はレシーバー
SCB_TX_FIFO_WR	トランスミッター FIFO に書き込まれるデータ フレームを格納。動作はプッシュの動作と同じ
SCB_RX_FIFO_RD	レシーバー FIFO から読み出されるデータ フレームを格納。データ フレームを読み出すとそのデータ フレームが FIFO から削除される。POP 動作と同じ。このレジスタはソフトウェアによって読み出される時にデータ フレームが FIFO から削除されるという副作用がある
SCB_RX_FIFO_RD_SILENT	レシーバー FIFO から読み出されるデータ フレームを格納。データ フレームを読み出してもそのデータ フレームが FIFO から削除されない。PEEK 動作と同じ
SCB_RX_MATCH	スレーブ デバイス アドレスとマスク値を格納
SCB_TX_FIFO_STATUS	トランスミッター FIFO に格納されているバイト数、データ フレームがハードウェアに読み出される位置 (読み出しポインタ)、新しいデータ フレームが書き込まれる位置 (書き込みポインタ) を指定し、トランスミッター FIFO が有効なデータを格納しているかを確定
SCB_RX_FIFO_STATUS	SCB_TX_FIFO_STATUS レジスタと同じ機能を実行。ただし対象はレシーバー
SCB_EZ_DATA	EZ メモリ位置にデータを格納

14.2.7 SPI 割り込み

SPI は内部割り込み要求にも外部割り込み要求にも対応しています。内部割り込みイベントはここで示されています。PSoC Creator はバッファ管理割り込みの処理用に必要な割り込みサービス ルーチン (ISR) を生成します。外部割り込みコンポーネントを SPI コンポーネント (外部割り込みが有効) の割り込み出力に接続することでカスタム ISR も使用可能です。

SPI の事前定義の割り込みは TX 割り込みと RX 割り込みに分類できます。TX 割り込みの出力はすべての可能な TX 割り込みソースのグループの論理和 (OR) です。この信号は任意の有効な TX 割り込みソースが真の場合、HIGH になります。RX 割り込みの出力はすべての可能な RX 割り込みソースのグループの論理和 (OR) です。この信号は任意の有効な Rx 割り込みソースが真の場合、HIGH になります。様々な割り込みレジスタは割り込みの実際のソースを決定するために使用されます。

SPI は以下のイベントで割り込みに対応します :

- SPI マスター転送が完了
- SPI バス エラー : SPI 転送間の予期しない時点にスレーブの選択を解除
- EZSPI 転送が発生した後、SPI スレーブの選択を解除
- EZSPI 書き込み転送が発生した後、SPI スレーブの選択を解除
- TX
  - TX FIFO のエントリ数が SCB_TX_FIFO_CTRL レジスタの TRIGGER_LEVEL ビットで指定する値より少ない
  - TX FIFO が満杯でない
  - TX FIFO が空
  - TX FIFO オーバーフロー
  - TX FIFO アンダーフロー
- RX
  - RX FIFO が満杯
  - RX FIFO が空でない
  - RX FIFO オーバーフロー
  - RX FIFO アンダーフロー
- SPI 外部からのクロック供給
  - スレーブ選択時の復帰要求の生成
  - 各転送の終了時の SPI STOP の検出
  - 書き込み転送の終了時の SPI STOP の検出
  - 読み出し転送の終了時の SPI STOP の検出

注 : SPI 割り込み信号は Cortex-M0 NVIC に固定接続され、外部ピンに接続できません。

14.2.8 SPI の有効化と初期化

SPI は次の順でプログラムしなければなりません :

1. 表 14-3 に従って SCB_SPI_CTRL レジスタを使用し、プロトコル固有の情報をプログラムします。これはプロトコルのサブモードの選択とマスター スレーブ機能の選択動作を含んでいます。EZSPI はスレーブ モードのみで使用できます
2. 表 14-4 に示すように SCB_TX_CTRL と SCB_RX_CTRL レジスタを使用し、一般的なトランスミッターとレシーバーの情報をプログラムします :
   a. データ フレームの幅を指定。EZSPI の場合、この幅は 8 でなければなりません
   b. 最初に送信／受信されるビットが MSB であるか LSB であるかを指定。EZSPI の場合、最初に送受信されるビットが MSB でなければなりません
3. 表 14-5 に示すように SCB_TX_FIFO_CTRL と SCB_RX_FIFO_CTRL レジスタそれぞれを使用し、トランスミッターとレシーバー FIFO をプログラムします :
   a. トリガー レベルを設定
   b. トランスミッターとレシーバー FIFO、シフト レジスタをクリア
   c. TX と RX FIFO をフリーズ

4. SCB ブロックを有効にするために SCB_CTRL レジスタをプログラムします。また動作モードを選択。これらのレジスタ ビットは表 14-2 に示します
5. ブロックを有効にします (SCB_CTRL レジスタの ENABLED ビットに「1」を書き込みます)。ブロックを有効にした後、コントロール ビットを変更してはいけません。変更 (例えば、Motorola モードから TI モードに動作モードを変更するか、外部クロック供給動作から内部クロック供給動作に変更すること) はブロックを無効にした後に行わなければなりません。変更はブロックを再び有効にしなければ有効になりません。ブロックを再び有効にすると再初期化およびその関連の状態 (FIFO 内容など) が消失することに注意してください

表 14-2. SCB_CTRL レジスタ

ビット	名前	値	説明
[25:24]	MODE	00	I^2C モード
		01	SPI モード
		10	UART モード
		11	予約済み
31	ENABLED	0	SCB ブロックは無効
		1	SCB ブロックは有効

表 14-3. SCB_SPI_CTRL レジスタ

ビット	名前	値	説明
[25:24]	MODE	00	SPI Motorola サブモード (これは EZSPI サポートのモードのみ)
		01	SPI Texas Instruments サブモード
		10	SPI National Semiconductors サブモード
		11	予約済み
31	MASTER_MODE	0	スレーブ モード。(これは EZSPI サポートのモードのみ)
		1	マスター モード

表 14-4. SCB_TX_CTRL/SCB_RX_CTRL レジスタ

ビット	名称	説明
[3:0]	DATA_ WIDTH	「DATA_WIDTH + 1」は送信または受信データ フレーム内のビット数。有効な範囲は [3、15]。スタート、ストップとパリティのビットを含まない EZSPI 用には、この値は 0b0111 でなければならない
8	MSB_FIRST	1 = MSB ファースト 0 = LSB ファースト EZSPI 用には、この値は 1 でなければならない
9	MEDIAN	これは SCB_RX_CTRL 専用。 3 タップのデジタルメジアン フィルターが入力インターフェース ラインに適用されるかどうかを決定。このフィルターはエラーの影響を低下させるが、より高いオーバーサンプリングレートを必要とする 1 = 有効 0 = 無効

表 14-5. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL レジスタ

ビット	名称	説明
[7:0]	TRIGGER_LEVEL	トリガー レベル。このフィールドの値に比べてトランスミッターFIFO のエントリ数がより少ないまたはレシーバーFIFO のエントリ数がより多い場合、それぞれトランスミッターまたはレシーバー トリガー イベントが生成される
16	CLEAR	「1」の時トランスミッターまたはレシーバー FIFO とシフト レジスタはクリアされる
17	FREEZE	「1」の時トランスミッターまたはレシーバー FIFO へのハードウェア読み出し／書き込みは無効フリーズは TX または RX FIFO の読み出し／書き込みポインタを進めない

14.2.9 内部と外部クロック供給 SPI 動作

SCB は SPI と I^2C 機能用に内部クロック供給動作も外部クロック供給動作もサポートします。内部クロック供給動作はチップが供給するクロックを利用します。外部クロック供給動作はシリアル インターフェースが供給するクロックを利用します。外部クロック供給動作ではディープスリープのシステム消費電力モードで動作が可能です。

内部クロック供給動作はシステムの高周波数クロック (HFCLK) を使用します。システムのクロック供給の詳細については 57 ページのクロック供給システムの章を参照してください。オーバーサンプリングもサポートされています。オーバーサンプリングは高周波数クロックに対して行われます。SCB_CTRL レジスタの OVS ビット (ビット [3:0]) はオーバーサンプリングを指定します。

SPI マスター モードではオーバーサンプリングの有効な範囲は 4 ～ 16 です。したがってクロック速度が 48MHz の場合、最大ビット レートは 12Mbps です。しかし、I/O セルと配線遅延を考慮すれば、オーバーサンプリングは正常の動作のために 6 ～ 16 の範囲内で設定する必要があります。そのため、最大ビット レートは 8Mbps になります。**注：**可能な最大ビットレートを達成するために、LATE_MISO_SAMPLE は SPI マスター モードで「1」に設定されなければなりません。このビットのデフォルト値は「0」です。

SPI スレーブ モードでは SCB_CTRL レジスタの OVS フィールド (ビット [3:0]) は使用されません。ただしインターフェース クロック (SCLK) に対する SCB クロックの周波数の条件があります。この条件は SCLK に対する SCB クロックの比で表されます。この比率は SCB_RX_CTRL レジスタの MEDIAN と SCB_CTRL レジスタの LATE_MISO_SAMPLE という 2 つのフィールドに左右されます。外部 SPI マスターが MISO の遅いサンプリングに対応し、かつメディアン ビットが「0」に設定されている場合、達成可能な最大データ レートは 16Mbps です。外部 SPI マスターが MISO の遅いサンプリングに対応していない場合、最大のデータ レートは 8Mbps に制限されます (メディアン ビットは「0」に設定)。これらのビットに基づく最大ビットレートを表 14-6 に示します。

表 14-6. SPI スレーブの最大データ速度

48MHz のペリフェラル クロックでの最大ビット速度	比率の要件	SCB_RX_CTRL のメジアン	SCB_CTRL の LATE_MISO_SAMPLE
8Mbps	³6	0	1
6Mbps	³8	1	1
4Mbps	³12	0	0
3Mbps	³16	1	0

外部クロック供給動作は以下に制限されます :

- スレーブ機能
- EZ機能。EZ機能はブロックのSRAMをメモリ構造として使用します。非EZ機能はブロックのSRAMをTXとRX FIFOとして使用します。外部クロック供給動作では FIFO がサポートされません。
- Motorola 社モード 0、1、2、3

外部クロック供給の EZ 動作モードでは (インターフェース クロックが 48MHz の時) 48Mbps のデータ レートをサポートできます。

内部と外部クロック供給動作は SCB_CTRL レジスタの以下の 2 つのフィールドによって決められます :

- **EC_AM_MODE:** SPI スレーブの選択が内部クロック供給 (「0」) であるか外部クロック供給 (「1」) であるかを示します。SPI スレーブの選択はプロトコル動作の最初の部分を含んでいます。
- **EC_OP_MODE:** プロトコル動作の残りの部分 (SPI スレーブ選択以外) は内部クロック供給 (「0」) であるか外部クロック供給 (「1」) であるかを示します。前述のように外部クロック供給動作は非 EZ 機能をサポートしません。

この 2 つのレジスタ フィールドは SPI の機能動作を決定します。これらのレジスタ フィールドはアクティブ、スリープ、ディープスリープのシステム消費電力モードでの必要な動作に基づいて設定する必要があります。不正な設定はいくつかのシステム電力モードでの誤った動作を引き起こす可能性があります。表 14-7 と表 14-8 に非 EZ モードと EZ モードでの SPI の設定を示します。

14.2.9.1 非 EZ の動作モード

非 EZ モードでは 2 つの可能な設定があります。外部クロック供給動作が非 EZ 機能にサポートされない (FIFO がサポートされない) ため、EC_OP_MODE は常に「0」にセットする必要があります。ただし EC_AM_MODE は「0」にも「1」にもセットできます。表 14-7 に可能なオプションをまとめます。

表 14-7. 非 EZ モードでの SPI 動作

SPI (非 EZ) モード				
システム消費電力モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
アクティブとスリープ	内部クロックを使用する選択 内部クロックを使用する動作	外部クロックを使用する選択 : 内部クロックを使用する動作 アクティブ モードでは復帰割り込みソースは無効 (MASK = 0)。 スリープ モードで MASK ビットはユーザーによって設定可能。	非対応	非対応
ディープスリープ	非対応	外部クロックを使用する選択 : 復帰割り込みソースは有効 (MASK = 1) 0xFF を送信		

EC_OP_MODE が「0」、EC_AM_MODE が「0」: この設定はアクティブとスリープのシステム消費電力モードでのみ機能します。ブロック機能全体は内部クロック供給ドメインで提供されます。

EC_OP_MODE が「0」、EC_AM_MODE が「1」: この設定はアクティブとスリープのシステム消費電力モードで機能しディープスリープのシステム消費電力モードで制限された (ウェイクアップ) 機能を提供します。SPI スレーブ選択は外部クロック供給回路で行われます : アクティブのシステム消費電力モードでは内部と外部クロック供給回路の両方が有効で、ディープスリープのシステム消費電力モードでは外部クロック供給回路のみが有効です。外部クロック供給回路はスレーブ選択動作を検出すると、割り込みを生成し、CPU を復帰させるために使用できる復帰割り込みソース ビットをセットします。

- アクティブのシステム消費電力モードでは、CPU とブロックの内部クロック供給動作が有効で復帰割り込みソースが無効です (対応する MASK ビットが「0」です)。しかしスリープ モードではアプリケーションによって復帰割り込みソースが有効の場合も無効の場合もあります (MASK ビットが「1」または「0」です)。スリープ モードでの残りの動作はアクティブ モードと同じです。内部クロック供給動作は進行中の SPI 転送を処理します。
- ディープスリープのシステム消費電力モードでは CPU を復帰させる必要があり復帰割り込みソースが有効 (MASK ビットが「1」) です。復帰に時間がかかりますので進行中の SPI 転送は否定応答され (「1」ビットまたは「0xFF」バイトが MISO ラインから送信され)、内部クロック動作は復帰した時に次の SPI 転送を処理します。

14.2.9.2 EZ 動作モード

EZ モードでは 3 つの可能な設定があります。EC_OP_MODE が「0」の時 EC_AM_MODE は「0」または「1」にセットでき EC_OP_MODE が「1」の時 EC_AM_MODE は「1」にセットしなければなりません。表 14-8 に可能なオプションをまとめます。灰色のセルは可能であるが推奨されない設定を示します。推奨しない理由はこの設定が外部クロック回路 (スレーブ選択) から内部クロック供給回路 (残りの動作) への切り替えを引き起こすためです。EC_AM_MODE=0 と EC_OP_MODE=1 の組み合わせオプションは無効でブロックは応答しません。

表 14-8. EZ モードでの SPI 動作

SPI、EZ モード				
システム消費電力モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
アクティブとスリープ	内部クロックを使用する選択 内部クロックを使用する動作	外部クロックを使用する選択 内部クロックを使用する動作 アクティブ モードでは復帰割り込みソースは無効 (MASK = 0)。 スリープ モードでMASKビットはユーザーによって設定可能	無効	外部クロックを使用する選択 外部クロックを使用する動作
ディープスリープ	非対応	外部クロックを使用する選択 : 復帰割り込みソースは有効 (MASK = 1) 0xFF を送信		外部クロックを使用する選択 外部クロックを使用する動作

EC_OP_MODE が「0」、EC_AM_MODE が「0」: この設定はアクティブとスリープのシステム消費電力モードでのみ機能します。ブロック機能全体は内部クロック供給ドメインで提供されます。

EC_OP_MODE が「0」、EC_AM_MODE が「1」: この設定はアクティブとスリープのシステム消費電力モードで機能しディープスリープのシステム消費電力モードで制限された (ウェイクアップ) 機能を提供します。SPI スレーブ選択は外部クロック供給回路で行われます : アクティブのシステム消費電力モードでは内部と外部クロック供給回路の両方が有効でディープスリープのシステム消費電力モードでは外部クロック供給回路のみが有効です。外部クロック供給回路はスレーブ選択動作を検出すると、割り込みを生成し、CPU を復帰させるために使用できる復帰割り込みソースビットをセットします。

- アクティブのシステム消費電力モードでは、CPU とブロックの内部クロック供給動作が有効で復帰割り込みソースが無効です (対応する MASK ビットが「0」です)。しかしスリープ モードではアプリケーションによって復帰割り込みソースは有効か無効です (MASK ビットが「1」または「0」です)。スリープ モードでの残りの動作はアクティブ モードと同じです。内部クロック供給動作は進行中の SPI 転送を処理します。
- ディープスリープのシステム消費電力モードでは CPU を復帰させる必要があり復帰割り込みソースが有効 (MASK ビットが「1」) です。復帰に時間がかかりますので進行中の SPI 転送は否定応答され (「1」ビットまたは「0xFF」バイトが MISO ラインから送信され)、内部クロック動作は復帰した時に次の SPI 転送を処理します。

EC_OP_MODE が「1」、EC_AM_MODE が「1」: この設定はアクティブ、スリープとディープスリープのシステム消費電力モードで機能します。SCB 機能は外部クロック供給ドメインで提供されます。この設定はブロックの SRAM への外部クロック供給アクセスにつながることにご注意ください。このアクセスはデバイスからの内部クロック供給アクセスと衝突する可能性があります。この衝突はウェイト ステートまたはバス エラーにつながることがあります。SCB_CTRL レジスタの FIFO_BLOCK フィールドはウェイト ステート (「1」) またはバス エラー (「0」) を生成するかを決めます。

14.3 UART

汎用非同期レシーバー／トランスミッター (UART) プロトコルは非同期のシリアルインターフェース プロトコルです。UART 通信は通常ポイント ツー ポイントです。UART インターフェースは 2 つの信号を含んでいます :

- TX: トランスミッター出力
- RX: レシーバー入力

14.3.1 特長

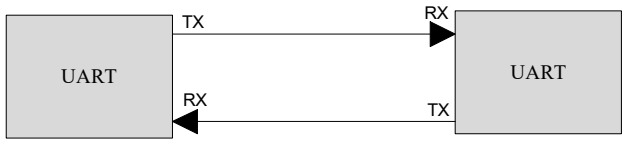
- 非同期トランスミッターとレシーバー機能
- 最大 3Mbps までのデータ レートをサポート
- UART プロトコルをサポート
  - 標準 UART
  - SmartCard (ISO7816) リーダー
  - IrDA
- ローカル インターコネクト ネットワーク (LIN) をサポート
  - ブレーク検出
  - ボーレート検出
  - 衝突検出 (駆動するビット値がバスに反映されず、他のコンポーネントが同じバスを駆動していることを検出する)
- マルチプロセッサ モード
- データ フレーム サイズは4ビット～9ビットにプログラム可能
- プログラマブルなSTOPビット数 (1から4までのビット半周期単位で設定可能)

- パリティをサポート (奇数パリティと偶数パリティ)
- 割り込みまたはポーリングによる CPU インターフェース
- プログラマブルなオーバーサンプリング

14.3.2 概要

図 14-8 に標準 UART TX と RX を示します。

図 14-8. UART 例



標準 UART 転送は「スタート ビット」から始まりその後に複数の「データ ビット」と「パリティ ビット」(任意) が続き 1 つ以上の「ストップ ビット」で終わります。スタートとストップ ビットはそれぞれデータ送信の開始と終了を示します。パリティ ビットはトランスミッターによって送信されシングル ビット エラーを検出するためにレシーバーによって使用されます。インターフェースがクロックを持たない (非同期) ため、トランスミッターとレシーバーは自身のクロックを使用し 1 ビット転送周期について同調する必要があります。

3 つの異なるシリアル インターフェース プロトコルがサポートされます :

- 標準 UART プロトコル
  - マルチプロセッサ モード
  - ローカル インターコネクト ネットワーク (LIN)
- UART と同じであるが、否定応答 (NACK) 信号を送信することが可能な SmartCard
- 変調スキームで UART から変更された IrDA

UART はデフォルトで 8 ビットのデータ フレーム幅をサポートします。ただしこの幅は 4 ～ 9 の範囲内の任意の値に設定できます。これはスタート、ストップとパリティ ビットを含んでいません。ストップ ビット数は 1 ～ 4 の範囲内にあります。パリティ ビットは有効であるかまたは無効です。有効な場合、パリティのタイプは偶数パリティまたは奇数パリティに設定できます。パリティ ビットは標準 UART モードと SmartCard UART モードでのみ使用可能です。IrDA UART モードではパリティ ビットは自動的に無効化されます。図 14-9 に SCB の UART インターフェースのデフォルト コンフィギュレーションを示します。

注 UART インターフェースは外部クロック供給動作をサポートしません。そのため UART はアクティブとスリープのシステム消費電力モードでのみ動作します。

14.3.3 UART の動作モード

14.3.3.1 標準プロトコル

標準 UART 転送はスタート ビットから始まりその後に複数のデータ ビットとパリティ ビット (任意) が続き 1 つ以上のストップ ビットで終わります。スタート ビットの値は常に「0」でデータ ビットの値は転送されるデータに依存し、パリティ ビットの値はデータ ビット上の偶数または奇数パリティを保証する値にセットされストップ ビットの値は「1」です。パリティ ビットはトランスミッターによって生成されシングル ビット送信エラーを検出するためにレシーバーによって使用可能です。データの送信中でない時、TX ラインは「1」(ストップ ビットと同じ値) です。

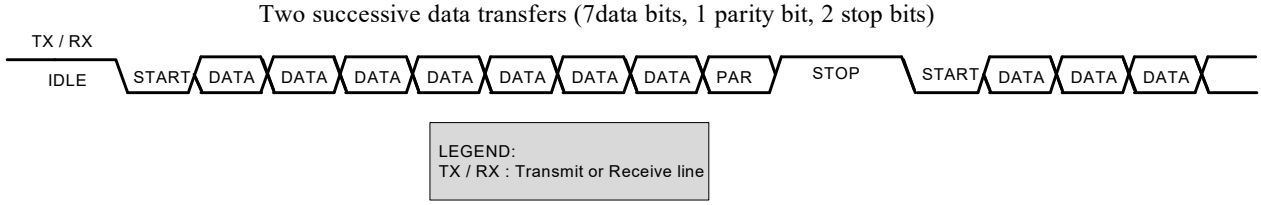
インターフェースがクロックを持たないため、トランスミッターとレシーバーはビット転送の周期について合意する必要があります。トランスミッターとレシーバーはそれ自身の内部クロックを持ちます。レシーバー クロックがビット転送周波数より高い周波数で動作するため、レシーバーは受信信号をオーバーサンプリングすることがあります。

ストップ ビットからスタート ビットへの移行は TX ライン上の「1」から「0」への変更で示されます。レシーバーはこの移行を使用してトランスミッター クロックと同期化することが可能です。各データ転送の開始時の同期化によりトランスミッターとレシーバー クロック間で周波数ドリフトが存在する場合でもエラーのない転送が可能です。必要なクロック精度はデータ転送サイズに依存します。

連続したデータ転送間のストップ周期とストップ ビット数は通常トランスミッターとレシーバー間で合意され 1 ～ 3 ビットの転送周期の範囲内にあります。

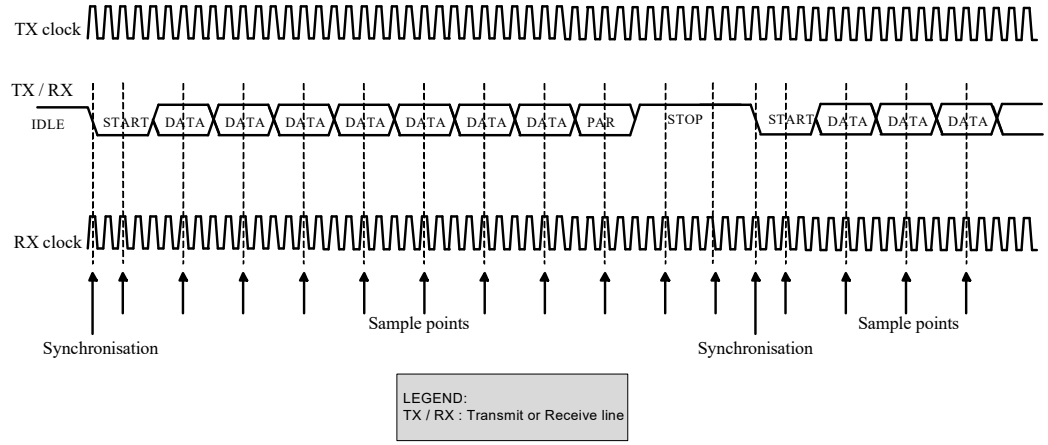
図 14-9 に UART プロトコルを示します。

図 14-9. UART の標準プロトコル例



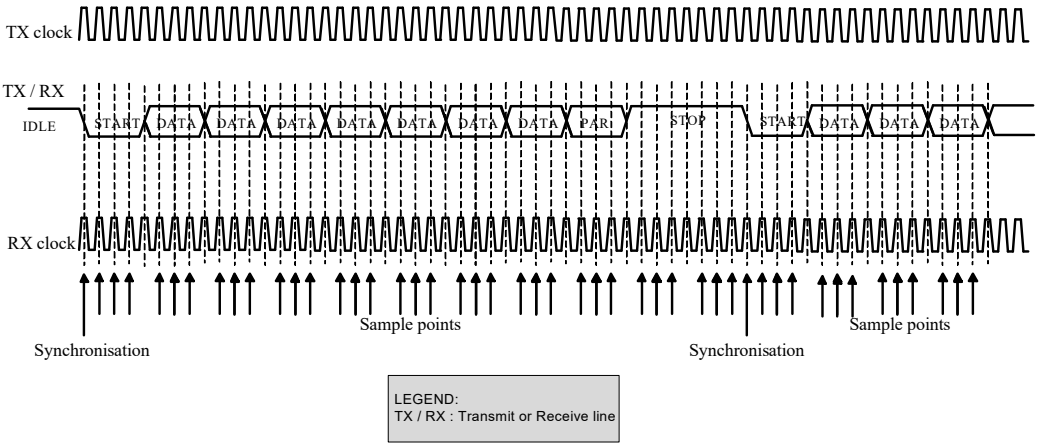
レシーバーは受信信号をオーバーサンプリングします。ビット転送周期の中央にある (レシーバー クロック上の) サンプル ポイントの値が使用されます。図 14-10 にそれを示します。

図 14-10. UART の標準プロトコル例 (シングル サンプル)



ビット転送周期の中央前後の (レシーバー クロック上の) 3 サンプルは精度を高めるために多数決で使用されます。図 14-11 にそれを示します。

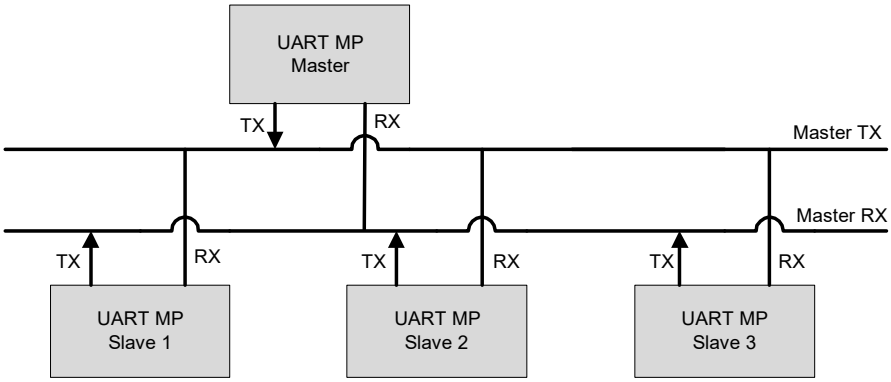
図 14-11. UART の標準プロトコル例 (複数サンプル)



UART マルチプロセッサ モード

UART_MP (マルチプロセッサ) モードは図 14-12 に示すようにシングルマスターマルチスレーブ トポロジで定義されています。データ フィールドが 9 ビット幅であるため、このモードは UART 9 ビットプロトコルとも呼ばれています。UART_MP は標準 UART モードの一部です。

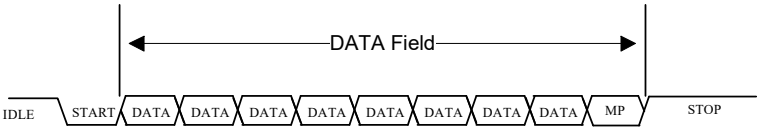
図 14-12. UART MP モードのバス接続



UART_MP モードの主なプロパティは次の通りです :

- シングル マスター マルチ スレーブ接続形態 (マルチドロップ ネットワーク)
- 各スレーブは固有のアドレスで識別
- 9 ビット データ フィールドを使用。その 9 番目のビットはアドレス／データ フラグ ビット (MP ビット)。このビットは HIGH にセットされた時にアドレス バイトを示し LOW にセットされた時にデータ バイトを示す。図 14-13 にデータ フレームを示す
- パリティ ビットは無効

図 14-13. UART MP データ フレーム

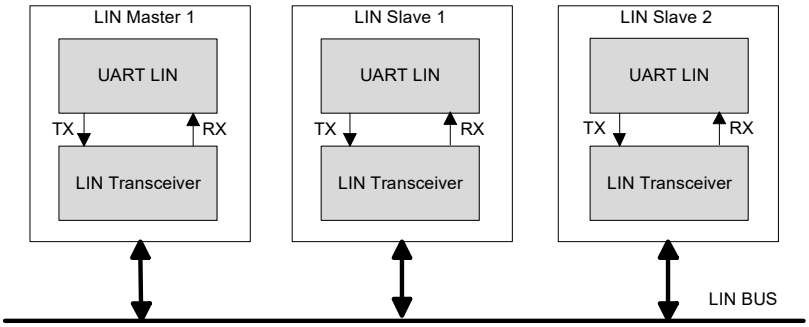


SCB は UART_MP モードでマスターまたはスレーブ デバイスとして使用可能です。SCB_TX_CTRL と SCB_RX_CTRL レジスタは両方とも 9 ビットのデータ フレーム サイズに設定できます。SCB が UART_MP マスター デバイスとして動作する時にファームウェアはすべてのアドレスまたはデータ フレーム用の MP フラグを変更します。このブロックが UART_MP スレーブ デバイスとして機能する時に SCB_UART_RX_CTRL レジスタの MP_MODE フィールドを「1」にセットする必要があります。SCB_RX_MATCH レジスタはスレーブ アドレスとアドレス マスク用にセットしなければなりません。SCB_CTRL レジスタの ADDR_ACCEPT フィールドが「1」にセットされた時に一致したアドレスが RX_FIFO に書き込まれます。受信したアドレスがそれ自身のアドレスに一致しない場合、インターフェースは次にアドレスが受信されるまで後続のデータを無視します。

UART ローカル インターコネクト ネットワーク (LIN) モード

LIN プロトコルは SCB によって標準 UART の一部としてサポートされます。LIN はシングルマスターマルチスレーブ トポロジで設計されています。LIN バス上に 1 つのマスター ノードと複数のスレーブ ノードがあります。SCB UART は LIN マスターとスレーブ機能を両方ともサポートします。LIN 仕様は物理層 (層 1) とリンク層 (層 2) の両方を定義します。図 14-14 に UART_LIN と LIN トランシーバーを示します。

図 14-14. UART_LIN と LIN トランシーバー

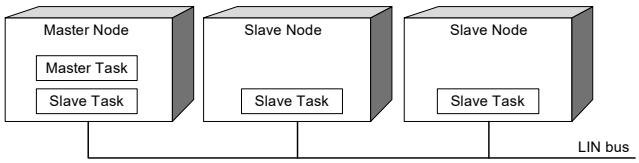


LIN プロトコルは以下の 2 タスクを定義します :

- マスター タスク : このタスクはヘッダー パケットを送信し、LIN 転送を開始することを含む
- スレーブ タスク : このタスクは応答の送信または受信を含む

図 14-15 に示すようにマスター ノードはマスター タスクとスレーブ タスクをサポートしスレーブ ノードはスレーブ タスクのみをサポートします。

図 14-15. LIN バスのノードとタスク

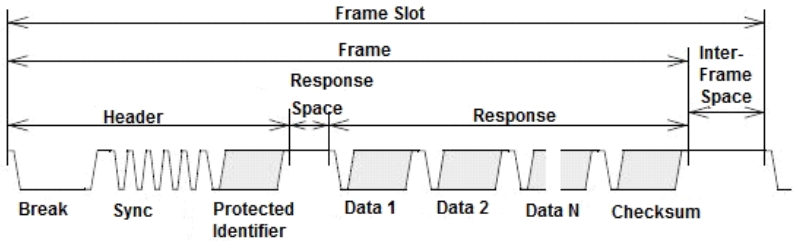


LIN フレームの構造

LIN は事前に定めた時間割にしたがってフレームを送信します。フレームは図 14-16 に示すようにヘッダーと応答フィールドに分けられます。

- ヘッダー フィールドは以下のものから構成されます :
  - ブレーク フィールド (少なくとも「0」値の 13 ビット周期)
  - 同期フィールド (0x55 バイトのフレーム)。同期フィールドはスレーブ タスクのクロックをマスター タスクのクロックと同期化するために使用可能
  - 識別子フィールド (特定のスレーブを指定するフレーム)
- 応答フィールドはデータとチェックサムから構成されます。

図 14-16. LIN フレームの構造



LIN プロトコル通信ではデータの最下位ビット (LSB) が最初に送信され最上位ビット (MSB) が最後に送信されます。スタートビットが 0 としてエンコードされストップ ビットが 1 としてエンコードされます。次の節で LIN フレーム内のすべてのバイトフィールドについて説明します。

ブレーク フィールド

すべての新しいフレームはマスターによって常に生成されるブレーク フィールドから始まります。ブレーク フィールドは論理 0 の最小 13 ビット タイムがありその後にブレーク デリミタが続きます。ブレーク フィールドの構造を図 14-17 に示します。

図 14-17. LIN ブレーク フィールド

同期フィールド

このフィールドはヘッダー フィールド内でマスターによって送信される 2 番目のフィールドです。その値は 0x55 です。同期フィールドは自動的なボーレート検出用にスレーブ タスクのクロックをマスター タスクのクロックと同期化するために使用可能です。図 14-18 に LIN 同期フィールドの構造を示します。

図 14-18. LIN 同期フィールド

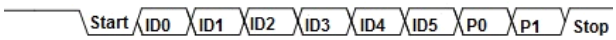
保護識別子 (PID) フィールド

保護識別子フィールドはフレーム識別子 (ビット 0 ～ 5) とパリティ (ビット 6 と 7) という 2 つのサブフィールドから構成されます。PID フィールド構造を図 14-19 に示します。

- フレーム識別子 : フレーム識別子は 3 つのカテゴリに分けられています。
  - 値 0 ～ 59 (0x3B) は信号を含むフレームに使用されます。
  - 60 (0x3C) と 61 (0x3D) は診断とコンフィギュレーションのために使用されます。
  - 62 (0x3E) と 63 (0x3F) は将来のプロトコル拡張に予約済みです。
- パリティ : フレーム識別子ビットがパリティを計算するために使用されます。

図 14-19 に PID フィールド構造を示します。

図 14-19. PID フィールド



データ . LIN では各フレームは 1 バイト～ 8 バイトのデータを含むことができます。ここでデータ バイトの LSB が最初に送信され MSB が最後に送信されます。

チェックサム

チェックサムは LIN フレーム内の最後のバイト フィールドです。すべてのデータ バイトのみの 8 ビットのキャリー付加算またはすべてのデータ バイトと PID フィールドの 8 ビットのキャリー付加算を反転することで計算されます。LIN フレーム内のチェックサムは 2 種類あります。それらは以下の通りです :

- 伝統的チェックサム : すべてのデータ バイトのみで計算されるチェックサムです (LIN 1.x スレーブで使用されます)。
- 拡張チェックサム : すべてのデータ バイトと保護識別子で計算されるチェックサムです (LIN 2.x スレーブで使用されます)。

LIN フレーム タイプ

フレーム タイプはフレームを送信するために有効でなければならない条件について言及します。LIN 仕様によって LIN フレームは5つの異なる種類がありますノードまたはクラスタはすべてのフレーム タイプをサポートする必要があるわけではありません。

無条件フレーム

これらのフレームは信号とフレーム識別子 (0x00 ～ 0x3B の範囲) を含みます。サブスクライバーはフレームを受信してアプリケーションに使用可能にします。フレームのパブリッシャーはヘッダーへの応答を提供します。

イベントトリガー フレーム

イベントトリガー フレームの目的は、まれに発生するイベントに対して複数のスレーブ ノードをポーリングさせてバスの帯域幅を多く消費させないことにより、LIN クラスタの応答性を向上させることです。イベントトリガー フレームは 1 つ以上の無条件フレームの応答信号を含みます。イベント トリガー フレームに対応する無条件フレームは以下の条件を満たす必要があります :

- 同じ長さ
- 同じチェックサム モデル (伝統的または拡張モデル) を使用する
- 保護識別子の最初のデータ フィールドを予約する
- 異なるスレーブ ノードによって発行される
- イベント トリガー フレームと同じスケジュール表に直接含めない

散発フレーム

散発フレームの目的はスケジュール表の残りの部分に影響せずにいくつかの動的動作をスケジュール表に統合することです。これらのフレームはフレーム スロットを共有する無条件フレームのグループを含んでいます。散発フレームを送信しようとする時に無条件フレームが更新した信号を含むかチェックされます。更新した信号がなければフレームが送信されずフレーム スロットが空です。

診断フレーム

診断フレームは常にトランスポート層を持って 8 データ バイトを含んでいます。

診断フレーム用のフレーム識別子は以下の通りです :

- マスター要求フレーム (0x3C) または
- スレーブ応答フレーム (0x3D)

マスター要求フレームを送信する前にマスター タスクはその診断モジュールへの問いかけを行って、フレームが送信されるかまたはバスに信号がないかを確認します。スレーブ応答フレーム ヘッダーは無条件で送信されます。スレーブ タスクはその診断モジュールに応じて応答を発行するかまたは受け入れます。

予約フレーム

これらのフレームは将来のために予約済みです。そのフレーム識別子は 0x3E と 0x3F です。

LIN のスリープへの移行と復帰

LIN プロトコルはマスターが「Go-to-sleep (スリープ状態に移行)」コマンドを送信した場合に LIN バスをスリープ モードに維持する機能を備えています。Go-to-sleep コマンドは最初のバイト フィールドが 0x00 で残りのバイトが 0xFF にセットされるマスター要求フレーム (ID = 0x3C) です。Go-to-sleep コマンドが受信された後にスレーブ ノード アプリケーションはアクティブのままであることがあります。この動作はアプリケーション固有のものです。LIN バスが 4 秒以上使用されないと LIN スレーブ ノードは自動的にスリープ モードに入ります。

復帰はバスを 250µs ～ 5ms の期間でドミナントにさせることで LIN バスに接続した任意のノード (LIN マスターまたは任意の LIN スレーブ) によって開始されます。各スレーブは 100ms 以内に復帰要求を検出し、ヘッダー処理の準備ができている必要があります。マスターも復帰要求を検出し、スレーブ ノードがアクティブになる時にヘッダーの送信を開始します。

LIN をサポートするには専用の (チップ外) ライン ドライバー／レシーバーが必要となります。LIN バス上の電源供給範囲は 7V ～ 18V です。通常 LIN ライン ドライバーは SCB TX ラインで提供される値で LIN ラインを駆動しその値を LIN ラインに移動してから SCB RX ラインに移動します。SCB 内の TX と RX ラインを比較することでバス衝突を検出できます (SCB_INTR_TX レジスタの SCB_UART_ARB_LOST フィールドで示されます)。

標準 UART インターフェースとして SCB を構成

標準 UART インターフェースとして SCB を構成するには様々なレジスタ ビットを次の順序で設定してください :

1. SCB_CTRL レジスタの MODE フィールド (ビット [25:24]) に「10」を書き込むことで SCB を UART インターフェースとして構成
2. SCB_UART_CTRL レジスタの MODE フィールド (ビット [25:24]) に「00」を書き込むことで標準プロトコルとして動作するように UART インターフェースを構成
3. UART MP モードまたは UART LIN モードを有効にするにはそれぞれ SCB_UART_RX_CTRL レジスタの MP_MODE (ビット 10) または LIN_MODE (ビット 12) への「1」の書き込み
4. 108 ページの「UART の有効化と初期化」で述べたステップ 2 ～ステップ 5 を行う

PSoC Creator が GUI を利用してこれらのすべてを自動的に行うことにご注意ください。これらのレジスタの詳細は「PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

14.3.3.2 SmartCard (ISO7816)

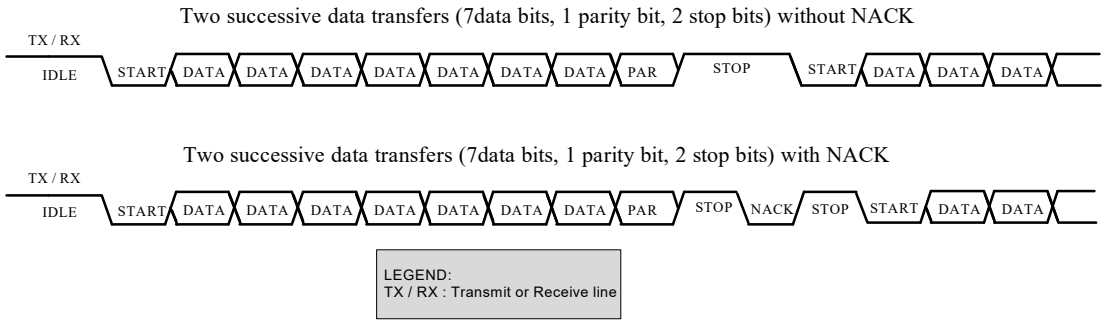
ISO7816 はシングル マスター シングル スレーブ トポロジで定義される非同期シリアルインターフェースです。ISO7816 はリーダー (マスター) とカード (スレーブ) 機能の両方を定義します。詳細については ISO7816 仕様を参照してください。マスター (リーダー) 機能のみは SCB によってサポートされます。このブロックは非同期文字送信を使用し、基本物理層のサポートを提供します。UART_TX と UART_RX コントロール モジュール間の内部多重化で UART_TX ラインが SmartCard I/O ラインに接続されます。

SmartCard 転送は UART 転送と同じですがレシーバーからトランスミッターへの否定応答 (NACK) 信号の送信が追加されます。NACK は常に「0」です。マスターとスレーブは同じラインを駆動できますがその駆動を同時に行うことはできません。

SmartCard 転送ではトランスミッターがスタート ビット、データ ビット (および、任意でパリティ ビット) を駆動します。これらのビットの駆動後にバスを解放することでストップ期間に入ります。バス解放によりラインの値が「1」 (ストップ ビットの値) となります。1 ビット転送周期からストップ周期に遷移した後にレシーバーはライン (値が「0」) 上の NACK 信号を 1 ビット転送周期で駆動することがあります。この NACK がトランスミッターによって観察され、ストップ周期を 1 ビット転送周期延長する反応をします。このプロトコルが動作するようにストップ周期は 1 ビット転送周期以上でなければなりません。NACK 付きのデータ転送は NACK なしのデータ転送より 1 ビット転送周期が長いことにご注意ください。通常プルアップ抵抗付きのトライステート ドライバーを搭載・使用しますので、ラインはデータまたはストップ ビットの送信中でない時に値が「1」です。

図 14-20 に SmartCard プロトコルを示します。

図 14-20. SmartCard 例



ISO7816 の通信ボーレートは以下のように計算されます :

ボーレート = f_{7816} × (D/F)

ここで f_{7816} はクロック周波数で、F はクロック レート変換整数で、D はボーレート調整整数です。

デフォルトで F = 372、D = f1、最大クロック周波数 = 5MHz となります。したがって最大ボーレートは 13.4kbps です。通常 3.57MHz のクロックが選択されます。ボーレートの標準値は 9.6kbps です。

UART SmartCard インターフェースとして SCB を構成

UART SmartCard インターフェースとして SCB を構成するには様々なレジスタ ビットを次の順で設定してください。PSoC Creator が GUI を利用してこれらのすべてを自動的に行うことに注意してください。これらのレジスタの詳細は「 PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。。

1. SCB_CTRL レジスタの MODE フィールド (ビット [25:24]) に「10」を書き込むことで SCB を UART インターフェースとして構成
2. SCB_UART_CTRL レジスタの MODE フィールド (ビット [25:24]) に「01」を書き込むことで SmartCard プロトコルとして動作するように UART インターフェースを構成
3. 108 ページの「UART の有効化と初期化」で述べるステップ 2 ～ステップ 5 を行う

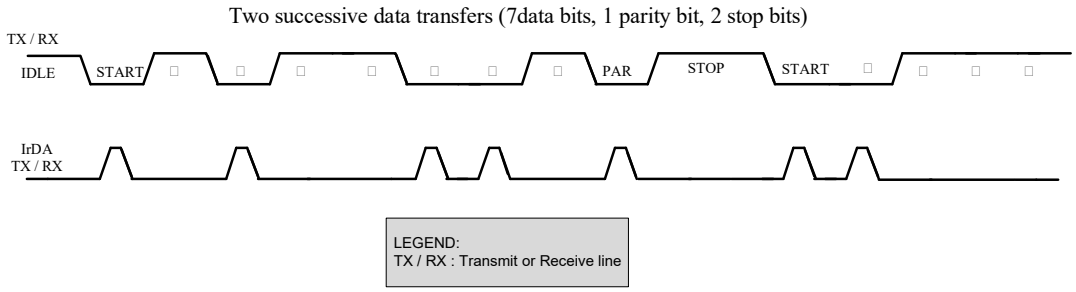
14.3.3.3 IrDA

SCB は UART インターフェースを使用し、赤外線データ協会 (IrDA) プロトコルをサポートして最大 115.2kbps のデータ レートを得ます。このブロックはデータ レートが 115.2kbps 未満の IrDA プロトコルの基本物理層のみをサポートします。そのため、このブロックを簡略化したシステムではその他の可能なシステム リソースを使って完全な IrDA 通信システムを搭載する方法を考慮する必要があります。

IrDA プロトコルは UART 信号に変調スキームを追加したものです。トランスミッターでビットが変調されます。レシーバーでビットが復調されます。変調スキームは Return-to-Zero-Inverted (ゼロ復帰逆転 - RZI) フォーマットを使用します。「0」のビット値がライン上の短い「1」パルスで示され、「1」のビット値がラインを「0」に維持することで示されます。これらのデータ レート (115.2kbps 以下) には RZI 変調スキームが使用されパルスの期間はビット周期の 16 分の 3 です。サンプリング クロック周波数は SCB_CTRL レジスタの SCB_OVS フィールドを構成することで選択したボーレートの 16 倍に設定する必要があります。

対応するブロックのクロック周波数を構成することで 115.2kbps 未満の異なる通信速度を得られます。追加可能な速度は 2.4kbps、9.6kbps、19.2kbps、38.4kbps と 57.6kbps です。IrDA シリアル赤外線インターフェースは 9.6kbps で動作します。図 14-21 に UART 転送の IrDA による変調方法を示します。

図 14-21. IrDA 例



UART IrDA インターフェースとして SCB を構成

UART IrDAインターフェースとしてSCBを構成するには様々なレジスタ ビットを次の順で設定してください。PSoC Creator が GUI を利用してこれらのすべてを自動的に行うことにご注意ください。これらのレジスタの詳細は「 PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

1. SCB_CTRL レジスタの MODE フィールド (ビット [25:24]) に「10」を書き込むことで SCB を UART インターフェースとして構成
2. SCB_UART_CTRL レジスタの MODE フィールド (ビット [25:24]) に「10」を書き込むことで IrDA プロトコルとして動作するように UART インターフェースを構成
3. SCB_RX_CTRL レジスタの MEDIAN フィールド (ビット 9) に「1」を書き込むことで入力インターフェース ライン上のメディアン フィルターを有効化
4. 108 ページの「UART の有効化と初期化」で説明したように SCB を構成

14.3.4 UART レジスタ

UART インターフェースは表 14-9 に示す一連の 32 ビットレジスタで制御されます。これらのレジスタの詳細は「 PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

表 14-9. UART レジスタ

レジスタ名	動作
SCB_CTRL	SCB を有効にしシリアル インターフェースのタイプ (SPI、UART、I^2C) を選択
SCB_UART_CTRL	UART のサブモード (標準 UART、SmartCard、IrDA) を選択するために使用 ; ローカル ループバック制御にも使用
SCB_UART_RX_STATUS	ビット周期を決定する BR_COUNTER 値を指定するために使用。これは SCB クロックの精度を設定するために使用。この値は SCB_CTRL レジスタの OVS ビットより高い精度を与える
SCB_UART_TX_CTRL	ストップ ビット数の指定、パリティの有効化、パリティ タイプの選択、NACK 時の再送信の有効化用に使用
SCB_UART_RX_CTRL	SCB_UART_TX_CTRL と同じ機能を実行 ; ただしマルチ プロセッサ モード、LIN モード、パリティ エラー時のドロップ、フレーム エラー時のドロップを有効にするためにも使用
SCB_TX_CTRL	データ フレーム幅を指定し送信の最初のビットが MSB であるか LSB であるかを指定するために使用
SCB_RX_CTRL	SCB_TX_CTRL レジスタと同じ機能を実行。ただし対象はレシーバー。またメジアン フィルターが入力インターフェース ライン上に使用されるかどうかを決定
SCB_UART_FLOW_CONTROL	UART トランスミッター用のフロー制御を構成

14.3.5 UART 割り込み

UART は内部割り込み要求にも外部割り込み要求にも対応しています。内部割り込みイベントはこの節で示されています。PSoC Creator はバッファ管理割り込みの処理用に必要な割り込みサービス ルーチン (ISR) を生成します。外部割り込みコンポーネントを UART コンポーネント (外部割り込みが有効) の割り込み出力に接続することでカスタム ISR も使用可能です。

UART の事前定義の割り込みは TX 割り込みと RX 割り込みに分類できます。TX 割り込みの出力はすべての可能な TX 割り込みソースのグループの論理和 (OR) です。この信号は任意の有効な TX 割り込みソースが真の場合、HIGH になります。RX 割り込みの出力はすべての可能な RX 割り込みソースのグループの論理和 (OR) です。この信号は任意の有効な Rx 割り込みソースが真の場合、HIGH になります。UART は以下のイベントで割り込みを提供します :

- TX
  - TX FIFO のエントリ数が SCB_TX_FIFO_CTRL レジスタの TRIGGER_LEVEL ビットで指定する値より少ない
  - TX FIFO が満杯でない
  - TX FIFO が空
  - TX FIFO オーバーフロー
  - TX FIFO アンダーフロー
  - TX が SmartCard モードで NACK を受信
  - TX 完了
  - アービトレーション ロスト (LIN または SmartCard モードの時)

- RX
  - RX FIFO のエントリ数が SCB_RX_FIFO_CTRL レジスタの TRIGGER_LEVEL ビットで指定する値より少ない
  - RX FIFO が満杯
  - RX FIFO が空でない
  - RX FIFO オーバーフロー
  - RX FIFO アンダーフロー
  - 受信したデータ フレームでフレーム エラーが発生
  - 受信したデータ フレームでパリティ エラーが発生
  - LIN ボーレート検出完了
  - LIN ブレーク検出が正常に行われる

14.3.6 UART の有効化と初期化

UART を以下の順序でプログラムしてください :

1. 表 14-10 に記載する SCB_SPI_CTRL レジスタ情報を使用し、プロトコル固有の情報をプログラムします。これはプロトコルのサブモードの選択とトランスミッター レシーバー機能の選択動作を含んでいます
2. 表 14-11 に記載する SCB_TX_CTRL と SCB_RX_CTRL レジスタ情報を使用し、一般的なトランスミッターとレシーバーの情報をプログラムします
   a. データ フレームの幅を指定します
   b. 最初に送信／受信されるビットが MSB であるか LSB であるかを指定します
3. 表 14-12 に記載する SCB_TX_FIFO_CTRL と SCB_RX_ FIFO_CTRL レジスタ情報それぞれを使用し、トランスミッターとレシーバー FIFO をプログラムします。
   a. トリガー レベルを設定
   b. トランスミッターとレシーバー FIFO、シフト レジスタをクリア
   c. TX と RX FIFO をフリーズ
4. SCB ブロックを有効にするために SCB_CTRL レジスタをプログラムします。また動作モード (表 14-13) を選択します。
5. ブロックを有効にします (SCB_CTRL レジスタの ENABLED ビットに「1」を書き込みます)。ブロックを有効にした後、コントロール ビットを変更してはいけません。動作モード (SmartCard から IrDA へ) の変更等はブロックを無効にしてから行います。変更はブロックを再び有効にしなければ有効になりません。ブロックを再び有効にすると再初期化およびその関連の状態 (FIFO 内容など) が消失することにご注意ください

表 14-10. SCB_UART_CTRL レジスタ

ビット	名前	値	説明
[25:24]	MODE	00	標準 UART
		01	SmartCard
		10	IrDA
		11	予約済み
16	LOOP_BACK	ループ バック制御 : このビットは SCB UART トランスミッターがその相手となるレシーバーと通信することを許可	

表 14-11. SCB_TX_CTRL/SCB_RX_CTRL レジスタ

ビット	名称	説明
[3:0]	DATA_ WIDTH	「DATA_WIDTH + 1」は送信または受信データ フレーム内のビット数。有効な範囲は [3、15]。スタート、ストップとパリティのビットを含まない
8	MSB_FIRST	1= MSB ファースト 0= LSB ファースト
9	MEDIAN	これは SCB_RX_CTRL 専用。 3 タップのデジタルメジアン フィルターが入力インターフェース ラインに適用されるかどうかを決定。このフィルターはエラーの影響を低下させるがより高いオーバーサンプリング レートを必要とする UART IrDA モードの場合このビットは常に「1」でなければならない 1 = 有効 0 = 無効

表 14-12. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL レジスタ

ビット	名称	説明
[7:0]	TRIGGER_LEVEL	トリガー レベル。このフィールドの値に比べてトランスミッターFIFO のエントリ数がより少ないまたはレシーバー FIFO のエントリ数がより多い場合、それぞれトランスミッターまたはレシーバー トリガー イベントが生成される
16	CLEAR	「1」の時トランスミッターまたはレシーバー FIFO とシフト レジスタはクリアまたは無効化される
17	FREEZE	「1」の時トランスミッターまたはレシーバー FIFO へのハードウェア読み出し／書き込みは無効フリーズは TX または RX FIFO の読み出し／書き込みポインタを進めない

表 14-13. SCB_CTRL レジスタ

ビット	名前	値	説明
[25:24]	MODE	00	I^2C モード
		01	SPI モード
		10	UART モード
		11	予約済み
31	ENABLED	0	SCB ブロックは無効
		1	SCB ブロックは有効

14.4 インター インテグレーテッド回路 (I2C)

この節は PSoC での I2C 実装を説明します。I2C プロトコル仕様の詳細については、NXP ウェブサイトに掲載される I2C バス仕様を参照してください。

14.4.1 特長

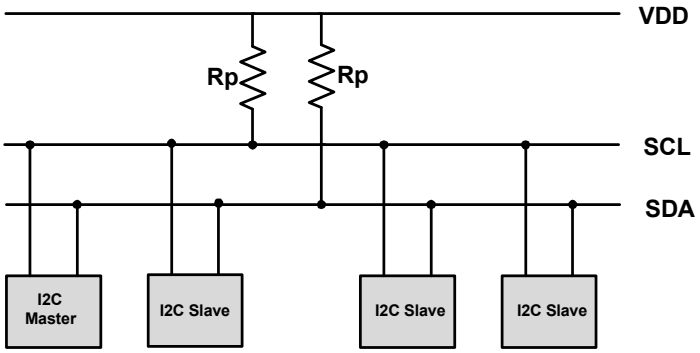
このブロックは以下の機能をサポートします :

- マスター、スレーブ、マスター／スレーブ モード
- 低速モード (50kbps)、標準モード (100kbps)、高速モード (400kbps)、高速モードプラス (1000kbps) のデータ レート
- 7 または 10 ビットのアドレス指定 (10 ビットのアドレス指定はファームウェア サポートが必要)
- クロック ストレッチおよび衝突検出
- I2C クロック信号 (SCL) のプログラマブルなオーバーサンプリング
- I2C データ信号 (SDA) の入力パスにデジタル メジアン フィルターを搭載することでエラーを削減
- アナログ グリッチ フィルターを使用することでグリッチなしの信号送信を実現
- 割り込みまたはポーリングによる CPU インターフェース

14.4.2 概要

図 14-22 に I2C 通信ネットワークの例を示します。

図 14-22. I2C インターフェース ブロック図



標準 I2C バスは以下のラインを含む 2 線式インターフェースです :

- シリアル データ (SDA)
- シリアル クロック (SCL)

I2C デバイスはオープン コレクタまたはオープンドレイン出力ステージを使用し、これらのラインに接続され、プルアップ抵抗 (Rp) が取り付けてあります。デバイス間に簡単なマスター／スレーブ関係があります。マスターとスレーブはトランスミッターかレシーバーのどちらでも動作できます。バスに接続する各スレーブ デバイスは、他と異なる 7 ビット アドレスをソフトウェアによりアドレス指定します。また、PSoC はファームウェアにより、I2C 向けの 10 ビット アドレス マッチングをサポートします。

14.4.3 用語および定義

表 14-14 に I²C 通信ネットワークにおいて一般に使用する用語を説明します。

表 14-14. I²C バス用語の定義

用語	説明
トランスミッター	データをバスに送信するデバイス
レシーバー	データをバスから受信するデバイス
マスター	転送を開始しクロック信号を生成し転送を終了するデバイス
スレーブ	マスターによってアドレス指定されるデバイス
マルチマスター	メッセージを破損せずにバスを同時に制御できる 2 つ以上のマスター
アービトレーション	2 つ以上のマスターが同時にバスを制御しようとする場合に、1 つのマスターだけに制御権を持たせ、メッセージが破損されないことを確保する手順
同期化	2 つ以上のデバイスのクロック信号を同期化する手順

14.4.3.1 クロック ストレッチ

スレーブ デバイスはデータ処理の準備ができていない場合には SCL ラインを拘束するため「0」に駆動することがあります。I/O 信号インターフェースの搭載により、マスターまたはスレーブが SCL ラインで駆動する値に関わらず、SCL ラインは常に「0」になります。これはクロック ストレッチといわれます。スレーブが SCL ラインを駆動することを許されるのはこの状態だけです。マスター デバイスは SCL ラインを監視し、スレーブが SCL ラインで正のクロック パルス (「1」) を生成できない時にストレッチを検出します。その後マスター デバイスはSCLラインでのポジティブ エッジの生成を延期し、クロックをストレッチしているスレーブ デバイスと実質的に同期します。

14.4.3.2 バス アービトレーション

I²C プロトコルはマルチマスター、マルチスレーブのインターフェースです。バス アービトレーションは SDA ラインを監視することで搭載します。バス衝突はマスターが SDA ラインで駆動している値と一致しない値を観測した時に検出されます。例えば、マスター 1 が SDA ラインで「1」を駆動していおり、マスター 2 が SDA ラインで「0」を駆動している時は、実際のライン値は I/O 信号インターフェースの搭載により「0」になります。マスター 1 が不一致を検出し、バスの制御を放棄します。マスター 2 は不一致を検出せずバスの制御を保持します。

14.4.4 I²C 動作モード

I²C は同期のシングル マスター、マルチマスター、マルチスレーブのシリアル インターフェースです。デバイスはマスター モード、スレーブ モード、マスター／スレーブ モードのいずれかで動作します。マスター／スレーブ モードでは、デバイスは指定されるとマスターからスレーブ モードに切り替えます。データ転送中はアクティブになるのは 1 個のシングル マスターのみです。アクティブなマスターは SCL ラインでのクロック駆動を担当します。表 14-15 に I²C 動作モードを示します。

表 14-15. I²C モード

モード	説明
スレーブ	スレーブ専用の動作 (デフォルト)
マスター	マスター専用の動作
マルチマスター	バス上で 2 つ以上のマスターをサポート
マルチマスタースレーブ	スレーブおよびマルチマスターの同時動作

I²Cバスを介したデータ転送は一定のフォーマットに従います。表 14-16 に I²C データ転送の一部となる一般的なバス イベントを幾つか記載します。書き込み転送と読み出し転送節でデータ転送時の I²C バス上のビット フォーマットを説明します。

表 14-16. I²C バス イベントの用語

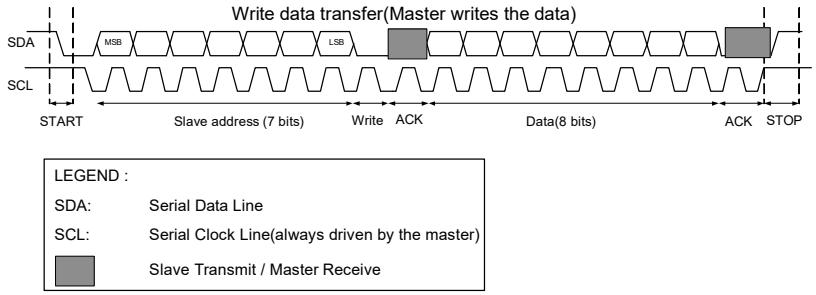
バス イベント	説明
START	SCL が HIGH の間の SDA ライン上の HIGH から LOW への遷移
STOP	SCL が HIGH の間の SDA ライン上の LOW から HIGH への遷移
ACK	トランスミッターが各バイトを送信した後、レシーバーは SDA ラインを LOW にプルダウンし、クロック パルスの HIGH 期間中に LOW の状態に維持。このイベントはレシーバーがバイトを正常に受信したことをトランスミッターに通知
NACK	トランスミッターが各バイトを送信した後、レシーバーは SDA ラインを LOW にプルダウンせずクロック パルスの HIGH 期間中に HIGH の状態に維持。このイベントはレシーバーがバイトを正常に受信したことをトランスミッターに通知
繰り返し START	転送の終了時に STOP 条件の代わりにマスターによって生成される START 条件
DATA	SDA の状態は SCL が LOW (データ変更) の間に変化し、SCLがHIGH (データ有効) の間は変えない

マルチマスター モードで動作している時、バスはビジーであるか確認される必要があります (他のマスターがスレーブと通信しているかもしれません)。この場合、マスターは START 信号を発信する前に現在の処理が完了するまで待機しなければなりません (表 14-16、図 14-23、図 14-24 を参照してください)。マスターは、データ転送開始の合図として STOP 信号を探します。

マルチスレーブ モードで動作している時、マスターがデータ転送中にアービトレーションを失う場合、ハードウェアはスレーブ モードに戻り、デバイスがバス上の他のマスターに応答できるようにバイト受信でスレーブ アドレス割り込みを生成します。これらのモードでは、読み出しと書き込みの 2 種の転送があります。書き込み転送では、マスターがスレーブにデータを送信します。読み出し転送では、マスターがスレーブからデータを受信します。書き込みと読み出し転送の例は 120 ページの「マスター モード転送の例」、122 ページの「スレーブ モード転送の例」および 126 ページの「マルチマスター モード転送の例」に記載します。

14.4.4.1 書き込み転送

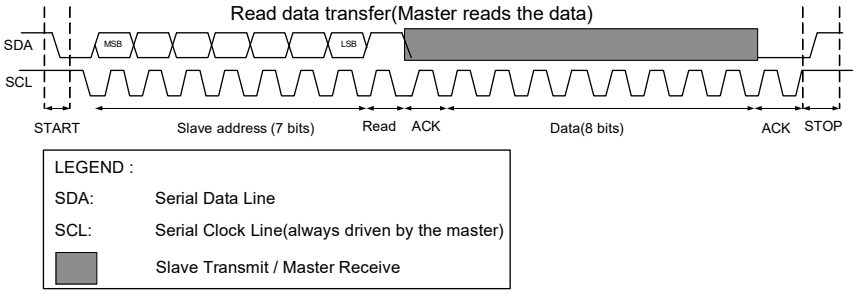
図 14-23. マスター書き込みデータ転送



- 通常の書き込み転送は I²C バス上で START 条件を生成するマスターから始まります。次にマスターは START 条件の後、7 ビットの I²C スレーブ アドレスと書き込みビット (「0」) を書き込みます。アドレス指定されたスレーブは 9 番目のビット時間の間にデータ ラインを LOW に引き下げることで確認応答バイトを送信します。
- スレーブ アドレスがスレーブ デバイスのいずれにも一致しない場合、またはアドレス指定されたデバイスが要求に確認応答しない場合、そのデバイスは送信 SDA ラインを LOW に引き下げないことにより、確認応答なし (NACK) を送信します。確認応答がない場合、プルアップ抵抗搭載により SDA ラインの値は「1」になります。
- NACK がスレーブによって送信された場合、マスターは STOP イベントで書き込み転送を終了することができます。またマスターは再転送のために、繰り返し START 条件を生成することができます。
- マスターは ACK を受信した場合、バスにデータを転送することができます。アドレス指定されたスレーブは書き込まれたデータの各バイトの受信を確認するために ACK を送信します。この ACK を受信すると、マスターは他のデータ バイトを送信することができます。
- 転送が完了すると、マスターは STOP 条件を生成します。

14.4.4.2 読み出し転送

図 14-24. マスター読み出しデータ転送



- 通常の読み出し転送は I²C バス上で START 条件を生成するマスターから始まります。次にマスターは START 条件の後、7 ビットの I²C スレーブ アドレスと読み出しビット (「1」) を書き込みます。アドレス指定されたスレーブは 9 番目のビット時間の間にデータ ラインを LOW に引き下げることで確認応答バイトを送信します。
- スレーブ アドレスが接続しているスレーブ デバイスに一致しない場合、またはアドレス指定されたデバイスが要求に確認応答しようとしない場合、送信 SDA ラインを LOW に引き下げないことにより、確認応答なし (NACK) が送信されます。確認応答がない場合、プルアップ抵抗搭載により SDA ラインの値は「1」になります。
- NACK がスレーブによって送信された場合、マスターは STOP イベントで読み出し転送を終了することができます。またマスターは再転送のために、繰り返し START 条件を生成することができます。
- スレーブ アドレスを認識した場合は、ACK 信号の後にデータの転送を開始します。マスターはスレーブから送信された各データ バイトの受信を確認するために ACK 信号を送信します。この ACK を受信すると、マスターは他のデータ バイトを送信することができます。

- マスターは、スレーブのデータ バイト転送を停止するためにスレーブに NACK 信号を送信することができます。これにより読み出し転送が完了します。
- 転送が完了すると、マスターは STOP 条件を生成します。

14.4.5 イージー I2C (EZI2C) プロトコル

イージーI2C (EZI2C) プロトコルは I²C プロトコル上に構築されたサイプレス独自の通信方式です。このプロトコルはインデックス付きメモリ転送を使用してI²Cスレーブと通信するために、標準的な I²C プロトコルを包含するソフトウェア ラッパーを使用しています。これにより個々のフレームのレベルでの CPU の介入が不要になります。

EZI2C プロトコルはスレーブ デバイスにあるメモリ アレイ (8 ビット幅 32 個の位置) をインデックスする 8 ビットのアドレスを定義します。EZ アドレスの下位 5 ビットはこの 32 個の位置をアドレス指定するために使用されます。EZI2C メモリ アレイから／へ転送されたバイト数はSTARTイベント時の EZ アドレスと STOP イベント時の EZ アドレスを比較することで分かります。

注 : I²C ブロックは、書き込みイネーブル バイトを持つ 16 ビット幅および深さ 16 のハードウェア FIFO メモリを有します。EZ と非 EZ 機能へのアクセス方法は異なります。非 EZ モードで FIFO は TXFIFO と RXFIFO に分けられます。それぞれは 16 ビット幅 8 個の配置を有します。EZ モードでは、FIFO は 8 ビット幅 32 個の配置のシングル メモリ ユニットとして使用されます。

EZI2C は 2 つの転送タイプがあります。これらはマスターからアドレス指定したスレーブ メモリ位置へのデータ書き込み、アドレス指定したスレーブ メモリ位置からのマスターによる読み出しです。

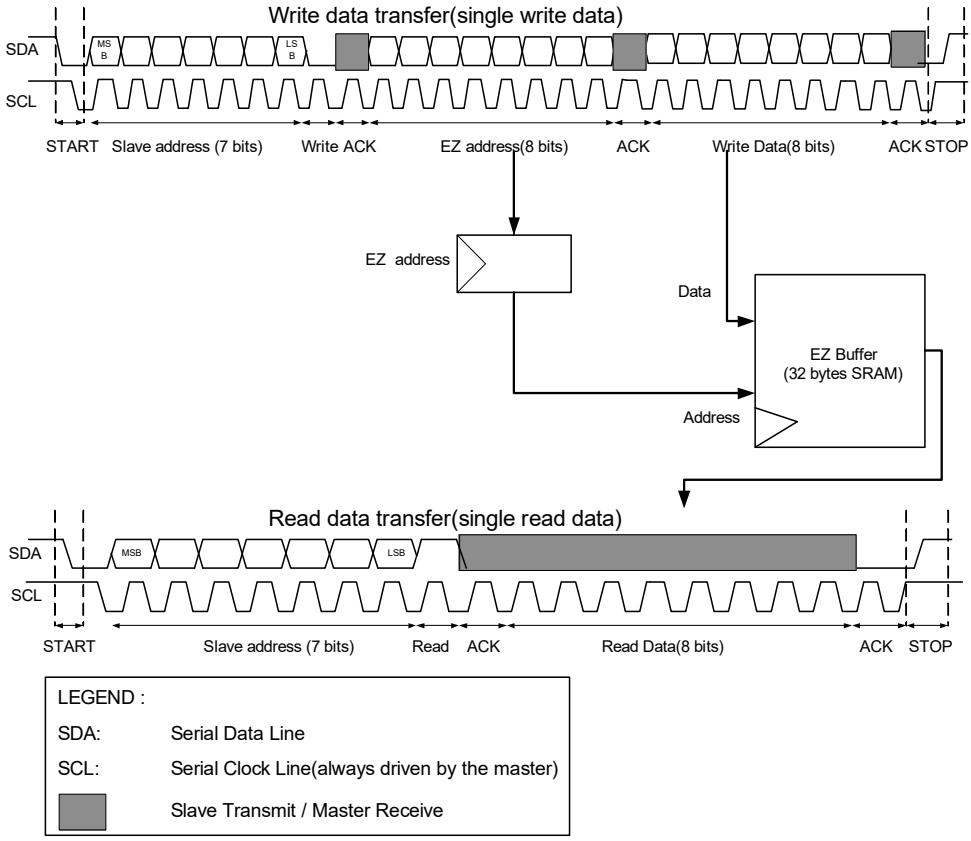
14.4.5.1 メモリ アレイ書き込み

メモリ アレイ インデックスへの EZ 書き込みは I²C 書き込み転送により行われるものです。最初に送信される書き込みデータは、マスターからスレーブに EZ アドレスを送信するために使用されます。書き込みデータの最下位 5 ビットがスレーブ側での「新しい」EZ アドレスとして使用されます。書き込み転送の他の書き込みデータ要素はすべてメモリ アレイに書き込まれるバイトです。バイトがメモリ アレイに書き込まれると、EZ アドレスはスレーブによって自動的にインクリメントされます。EZI2C バッファに書き込まれる連続データ バイトの数が EZI2C バッファの境界を超える場合には、次のバイトは最後の位置に上書きされます。

14.4.5.2 メモリ アレイの読み出し

メモリ アレイ インデックスからの EZ 読み出しは I²C 読み出し転送により行われます。EZ の読み出しは以前の EZ 書き込みがスレーブでEZアドレスを設定したことに依存します。最初に受信した読み出しデータは EZ アドレス メモリ位置のメモリ アレイから読み出されたバイトです。バイトがメモリ アレイから読み出されると、EZ アドレスは自動的にインクリメントされます。最終のメモリ位置になると、アドレスはゼロに戻ります。

図 14-25. EZI2C 書き込みおよび読み出しデータ転送



14.4.6 I2C レジスタ

表 14-17 に記載されるように、I2C インターフェースは、コンフィギュレーション、制御およびステータスレジスタのセットを読み書きすることで制御されます。

表 14-17. I2C レジスタ

レジスタ	機能
SCB_CTRL	I2C ブロックを有効にし、シリアル インターフェースのタイプ (SPI、UART、I2C) を選択。内部または外部からのクロック供給動作と EZ モードまたは非 EZ モードを選択するためにも使用
SCB_I2C_CTRL	モード (マスター、スレーブ) を選択しレシーバーの FIFO 状態に応じて ACK または NACK 信号を送信
SCB_I2C_STATUS	バスのビジー状態の検出、スレーブ／マスターの読み出し／書き込み転送状態を示し、EZ スレーブ アドレスを格納
SCB_I2C_M_CMD	マスターが START、STOP と ACK/NACK 信号を生成することを可能にする
SCB_I2C_S_CMD	スレーブが ACK/NACK 信号を生成することを可能にする
SCB_STATUS	外部からクロック供給されるロジックが EZ メモリを使用しているかどうかを示す。このビットは EZ メモリへのソフトウェア アクセスを発行しても安全であるかどうかを決定するためにソフトウェアによって使用することができる
SCB_I2C_CFG	SDA と SCL ラインからグリッチを取り除くフィルターを構成
SCB_TX_CTRL	データ フレーム幅を指定；送信の最初のビットが MSB であるか LSB であるかを指定するためにも使用
SCB_TX_FIFO_CTRL	トリガー レベル、トランスミッター FIFO とシフト レジスタのクリア、トランスミッター FIFO のフリーズ動作を指定

表 14-17. I2C レジスタ

レジスタ	機能
SCB_TX_FIFO_STATUS	トランスミッター FIFO に格納されているバイト数、データ フレームがハードウェアに読み出される位置 (読み出しポインタ)、新しいデータ フレームが書き込まれる位置 (書き込みポインタ) を指定し、トランスミッター FIFO が有効なデータを格納しているかを確定
SCB_TX_FIFO_WR	トランスミッター FIFO に書き込まれるデータ フレームを格納。動作はプッシュの動作と同じ
SCB_RX_CTRL	SCB_TX_CTRL レジスタと同じ機能を実行。ただし対象はレシーバー。またメジアン フィルターが入力インターフェース ライン上に使用されるかどうかを決定
SCB_RX_FIFO_CTRL	SCB_TX_FIFO_CTRL レジスタと同じ機能を実行。ただし対象はレシーバー
SCB_RX_FIFO_STATUS	SCB_TX_FIFO_STATUS レジスタと同じ機能を実行。ただし対象はレシーバー
SCB_RX_FIFO_RD	レシーバー FIFO から読み出されるデータを格納。データ フレームを読み出すとそのデータ フレームが FIFO から削除される。POP 動作と同じ。このレジスタはソフトウェアによって読み出される時にデータ フレームが FIFO から削除されるという副作用がある
SCB_RX_FIFO_RD_SILENT	レシーバー FIFO から読み出されるデータを格納。データ フレームを読み出してもそのデータ フレームが FIFO から削除されない。PEEK 動作と同じ
SCB_RX_MATCH	スレーブ デバイスのアドレスを格納しスレーブ デバイス アドレス MASK としても使用
SCB_EZ_DATA	EZ メモリ位置内のデータを格納

注 : I^2C レジスタ ビットの詳細説明については 「PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

14.4.7 I2C 割り込み

固定機能 I2C ブロックは、以下の条件のために割り込みを生成します。

- I2C マスター
  - I2C マスターがアービトレーションを喪失した
  - I2C マスターが NACK を受信した
  - I2C マスターが ACK を受信した
  - I2C マスターが STOP を送信した
  - I2C バス エラー (予期しない STOP/START 条件が検出された)
- I2C スレーブ
  - I2C スレーブがアービトレーションを喪失した
  - I2C スレーブ NACK を受信した
  - I2C スレーブ ACK を受信した
  - I2C スレーブ STOP を受信した
  - I2C スレーブ START を受信した
  - I2C スレーブ アドレスが一致
  - I2C バス エラー (予期しない STOP/START 条件が検出された)
- TX
  - TX FIFO のエントリ数が SCB_TX_FIFO_CTRL レジスタの TRIGGER_LEVEL ビットで指定する値より少ない
  - TX FIFO が満杯でない
  - TX FIFO が空
  - TX FIFO オーバーフロー
  - TX FIFO アンダーフロー
- RX
  - RX FIFO のエントリ数が SCB_RX_FIFO_CTRL レジスタの TRIGGER_LEVEL ビットで指定する値より少ない
  - RX FIFO が満杯
  - RX FIFO が空でない
  - RX FIFO オーバーフロー
  - RX FIFO アンダーフロー
- I2C 外部からのクロック供給
  - アドレス一致時の復帰要求
  - 各転送終了時の I2C STOP 検出
  - 書き込み転送終了時の I2C STOP 検出
  - 読み出し転送終了時の I2C STOP 検出

I2C 割り込み信号は Cortex-M0 NVIC に固定接続され、外部ピンに接続できません。

割り込みの出力はすべての可能な割り込みソースのグループの論理和です。いずれかの有効な割り込み条件が満たされた時に割り込みがトリガーされます。割り込みステータス レジスタは割り込みの実際のソースを判定するために使用されます。割り込みレジスタの詳細は「 PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

14.4.8 I2C の有効化および初期化

本節では、標準 (非 EZ) モードと EZI2C モードのために I2C ブロックを構成する方法を説明します。

14.4.8.1 I2C 標準 (非 EZ) モード用の構成

I2C インターフェースは次の順序でプログラムしなければなりません。

1. 表 14-18 に記載する SCB_I2C_CTRL レジスタ情報を使用し、プロトコル固有の情報をプログラム。これはマスター スレーブ機能の選択動作を含んでいます
2. 表 14-19 に記載する SCB_TX_CTRL と SCB_RX_CTRL レジスタ情報を使用し、一般的なトランスミッターとレシーバーの情報をプログラム
   a. データ フレームの幅を指定
   b. 最初に送信／受信されるビットを MSB に指定
3. 表 14-20 に記載する SCB_TX_FIFO_CTRL と SCB_RX_FIFO_CTRL レジスタ情報それぞれを使用し、トランスミッターとレシーバー FIFO をプログラム
   a. トリガー レベルを設定
   b. トランスミッターとレシーバー FIFO、シフト レジスタをクリア
4. I2C ブロックを有効にするために SCB_CTRL レジスタをプログラムし、I2C モードを選択。これらのレジスタ ビットを表 14-21 に示します。I2C レジスタの完全な説明については「PSoC 4000S Family: PSoC 4 Registers TRM」を参照してください。

表 14-18. SCB_I2C_CTRL レジスタ

ビット	名前	値	説明
30	SLAVE_MODE	1	スレーブ モード
31	MASTER_MODE	1	マスター モード

表 14-19. SCB_TX_CTRL/SCB_RX_CTRL レジスタ

ビット	名称	説明
[3:0]	DATA_ WIDTH	「DATA_WIDTH + 1」は送信または受信データ フレーム内のビット数。I2C の場合この値は常に 7
8	MSB_FIRST	1= MSB ファースト (I2C の場合は常に真)
		0= LSB ファースト
9	MEDIAN	これは SCB_RX_CTRL 専用。 3 タップのデジタル メジアン フィルターが入力インターフェース ラインに適用されるかどうかを決定。このフィルターはエラーの影響を低下させるが、より高いオーバーサンプリング レートを必要とする 1= 有効 0= 無効

表 14-20. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL

ビット	名称	説明
[7:0]	TRIGGER_LEVEL	トリガー レベル。このフィールドの値に比べてトランスミッター FIFO のエントリ数がより少ないまたはレシーバー FIFO のエントリ数がより多い場合、それぞれトランスミッターまたはレシーバー トリガー イベントが生成される
16	CLEAR	「1」の時トランスミッターまたはレシーバー FIFO とシフト レジスタはクリアされる
17	FREEZE	「1」の時トランスミッターまたはレシーバー FIFO へのハードウェア読み出し／書き込みは無効フリーズは TX または RX FIFO の読み出し／書き込みポインタを進めない

表 14-21. SCB_CTRL レジスタ

ビット	名前	値	説明
[25:24]	MODE	00	I2C モード
		01	SPI モード
		10	UART モード
		11	予約済み
31	ENABLED	0	SCB ブロックは無効
		1	SCB ブロックは有効

14.4.8.2 EZI2C モード用の構成

EZI2C モードのために I2C ブロックを構成するには、次の I2C レジスタ ビットを設定します。

1. SCB_CTRL レジスタの EZ_MODE ビット (ビット 10) に「1」を書き込むことで EZI2C モードを選択
2. I2C 標準 (非 EZ) モード用の構成で述べたステップ 2 ～ステップ 4 を実行
3. S_READY_ADDR_ACK (ビット 12) と S_READY_DATA_ACK (ビット 13) という SCB_I2C_CTRL レジスタのビットをセット

14.4.9 I2C における内部および外部クロック動作

I2C ブロックはデータ レート生成のために内部からクロックを供給される動作も外部からクロックを供給される動作もサポートします。内部クロック動作は PSoC システム バス クロックに引き出したクロック信号を使用します。外部クロック動作はユーザーから供給されるクロックを使用します。外部クロック動作により、オンチップ クロックがアクティブにならないディープスリープ消費電力モードで制限された機能が可能になります。システムのクロック供給の詳細については 57 ページのクロック供給システムの章を参照してください。

外部クロック供給動作は以下に制限されます :

- スレーブ機能

- EZ 機能

TX と RX FIFO は外部クロック動作をサポートしないため、非 EZ 機能に使用されません。

内部と外部クロック動作は SCB_CTRL レジスタの以下の 2 つのフィールドによって決められます :

- **EC_AM_MODE (外部クロック アドレス マッチング モード)**: I2C アドレス マッチング動作が内部 (「0」) か外部 (「1」) からクロックを供給されるかを示します
- **EC_OP_MODE (外部クロック動作 モード)**: プロトコル動作の残りの部分 (I2C スレーブ選択以外) が内部クロック供給 (「0」) であるか外部クロック供給 (「1」) であるかを示します 前述のように外部クロック供給動作は非 EZ 機能をサポートしません。

この 2 つのレジスタ フィールドは I2C の機能動作を決定します。これらのレジスタ フィールドはアクティブ、スリープ、ディープスリープのシステム消費電力モードでの必要な動作に基づいて設定する必要があります。誤った設定はいくつかのシステム消費電力モードでの誤った動作を引き起こす可能性があります。表 14-22 と表 14-23 に非 EZ モードと EZ モードでの I2C の設定を示します。

14.4.9.1 I2C の非 EZ 動作モード

このモードでは FIFO サポートがないため、外部クロック動作は非 EZ 機能に対してサポートされません。そして、EC_OP_MODE は非 EZ モードのために常に「0」に設定される必要があります。ただし、EC_AM_MODE は「0」にも「1」にもセットできます。表 14-22 に可能なオプションをまとめます。EC_AM_MODE = 0 と EC_OP_MODE = 1 の組み合わせオプションは無効でブロックが応答しません。

EC_AM_MODE が「0」で、EC_OP_MODE が「0」。

この設定はアクティブとスリープのシステム消費電力モードでのみ有効です。SCB 機能は外部クロックドメインで提供されます。

EC_AM_MODE が「1」で、EC_OP_MODE が「0」。

この設定はアクティブ、スリープ、ディープスリープのシステム消費電力モードで有効です。I2C アドレス マッチングはアクティブ、スリープおよびディープスリープの消費電力モードで外部クロック回路によって行われます。外部クロック供給回路はアドレス一致を検出すると、割り込みを生成し、CPU を復帰させるために使用できる復帰割り込みソース ビットをセットします。

表 14-22. 非 EZ モードでの I2C 動作

I2C (非 EZ) モード				
システム消費電力モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
アクティブとスリープ	内部クロックを使用するアドレス一致 内部クロックを使用する動作	外部クロックを使用するアドレス一致 内部クロックを使用する動作	非対応	
ディープスリープ	非対応	外部クロックを使用するアドレス一致 内部クロックを使用する動作		

- アクティブのシステム消費電力モードでは、CPU は有効で復帰割り込みソースは無効です (対応する MASK ビットは「0」です)。外部クロック供給回路はアドレス マッチングを担当し、内部クロック供給回路は、残りの I2C 転送を担当します。
- しかしスリープ モードではアプリケーションによって復帰割り込みソースが有効の場合も無効の場合もあります。残りの動作はアクティブ モードに似ています。
- ディープ スリープ モードでは、CPUがシャットダウンされ、復帰割り込み要因が有効になっている場合はI2Cのアクティビティの際に復帰します。CPU の復帰には時間がかかり、進行中の I2C 転送が確認応答なし (NACK) とされるかクロックがストレッチされます。NACK の場合、内部クロック供給回路は復帰後の最初の I2C 転送を担当します。クロック ストレッチの場合、内部クロック供給ロジックは復帰時の進行中／ストレッチされた転送を担当します。SCB_I2C_CTRL レジスタのレジスタ ビット S_NOT_READY_ADDR_NACK (ビット 14) は外部クロック供給回路が NACK (「1」) かクロック ストレッチを行うかを決定します (「0」)。

14.4.9.2 EZ モードでの I2C 動作

EZ モードでは 3 つの可能な設定があります。EC_OP_MODE が「0」の時 EC_AM_MODE は「0」または「1」にセットでき、EC_OP_MODE が「1」の時 EC_AM_MODE は「1」にセットする必要があります。表 14-23 に可能なオプションをまとめます。灰色のセルは可能であるが推奨されない設定を示します。推奨しない理由はこの設定が外部クロック回路 (スレーブ選択) から内部クロック供給回路 (残りの動作) への切り替えを引き起こすためです。EC_AM_MODE = 0 と EC_OP_MODE = 1 の組み合わせオプションは無効でブロックが応答しません。

表 14-23. EZ モードでの I2C 動作

I2C、EZ モード				
システム消費電力モード	EC_OP_MODE= 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
アクティブとスリープ	内部クロックを使用するアドレス一致 内部クロックを使用する動作	外部クロックを使用するアドレス一致 内部クロックを使用する動作	無効	外部クロックを使用するアドレス一致 外部クロックを使用する動作
ディープスリープ	非対応	外部クロックを使用するアドレス一致 内部クロックを使用する動作		外部クロックを使用するアドレス一致 外部クロックを使用する動作

- **EC_AM_MODE が「0」で、EC_OP_MODE が「0」**。この設定はアクティブとスリープのシステム消費電力モードでのみ有効です。
- **EC_AM_MODE が「1」で、EC_OP_MODE が「0」**。この設定は I2C の非 EZ モードと同じ効果があります。
- **EC_AM_MODE が「1」で、EC_OP_MODE が「1」**。この設定はアクティブとディープスリープのシステム消費電力モードで有効です。

I2C ブロックの機能は外部クロック供給ドメインで提供されます。この設定はブロックの SRAM への外部クロック供給アクセスにつながることにご注意ください。このアクセスはデバイスからの内部クロック供給アクセスと衝突する可能性があります。この衝突はウェイト ステートまたはバス エラーにつながることがあります。SCB_CTRL レジスタの FIFO_BLOCK (ビット 17) フィールドはウェイト ステート (「1」) またはバス エラー (「0」) を生成するかを決めます。

14.4.10 スリープから復帰

I2C アドレスの一致が発生すると、システムはスリープまたはディープスリープ システム消費電力モードから復帰します。固定機能 I2C ブロックはアドレスが一致した後に、アドレス ACK またはアドレス NACK の 2 つ動作のいずれかを行います。

アドレス ACK - I2C スレーブはクロック ストレッチを行い、デバイスが復帰するまで待機してアドレスを ACK します。

アドレス NACK - I2C スレーブは直ちにアドレスを NACK します。デバイスの復帰時間が経過した後、マスターは再びスレーブをポーリングする必要があります。このオプションはスレーブまたはマルチマスター スレーブ モードでのみ有効です。

注 : スリープ モードに切り替えている時に、I2C がスレーブ アドレス一致イベントでデバイスを復帰させるために、SCB_INTR_I2C_EC レジスタの割り込みビット WAKE_UP (ビット 0) を有効にする必要があります。

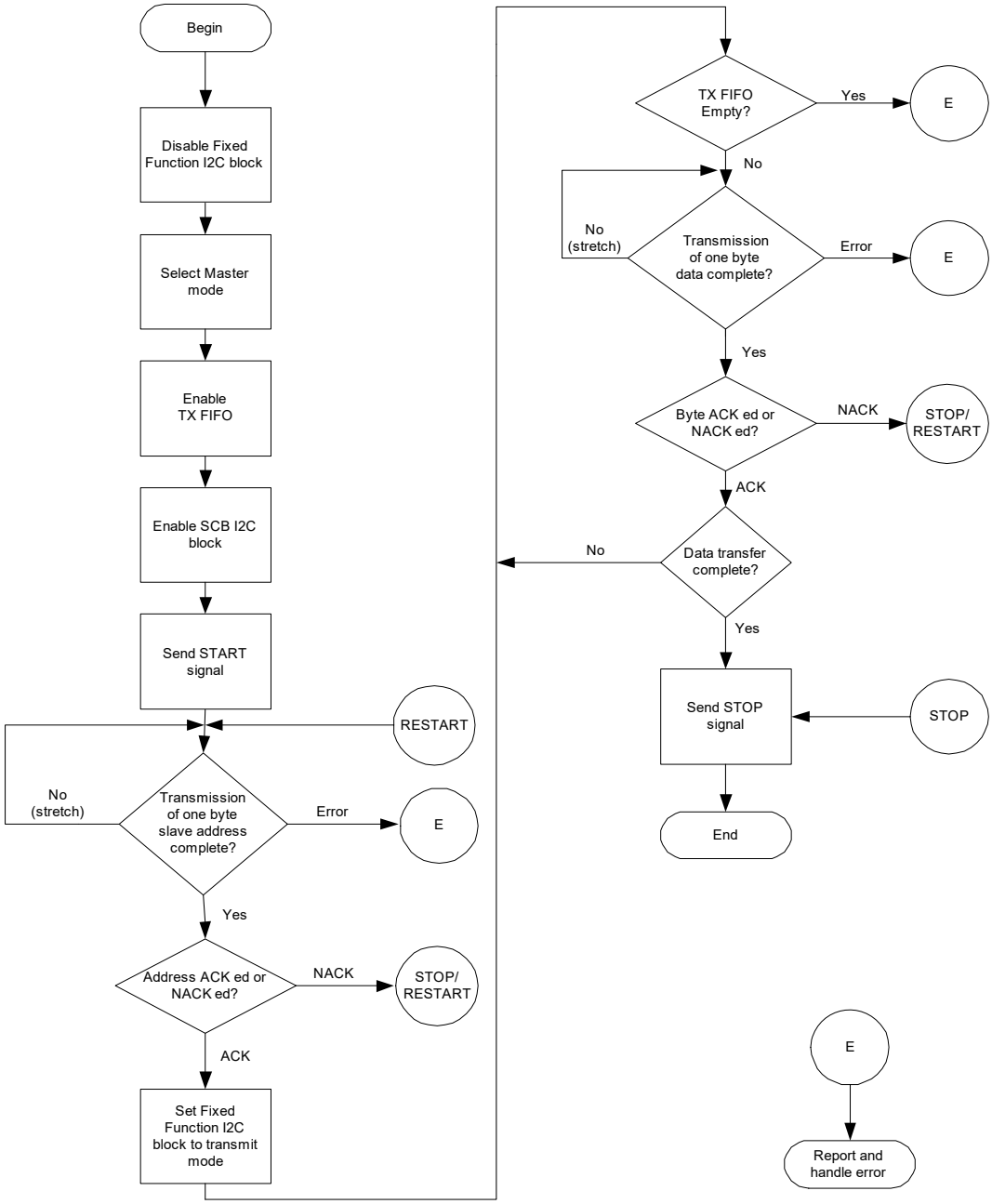
注 : デバイスが I2C スレーブ モードに設定されている場合、SCB へのクロックは、ディープ スリープ消費電力モードに入るときに無効にする必要があります。ディープ スリープ モードから復帰する時にクロックを有効にします。

14.4.11 マスター モード転送の例

マスターがデータを送信または受信します。

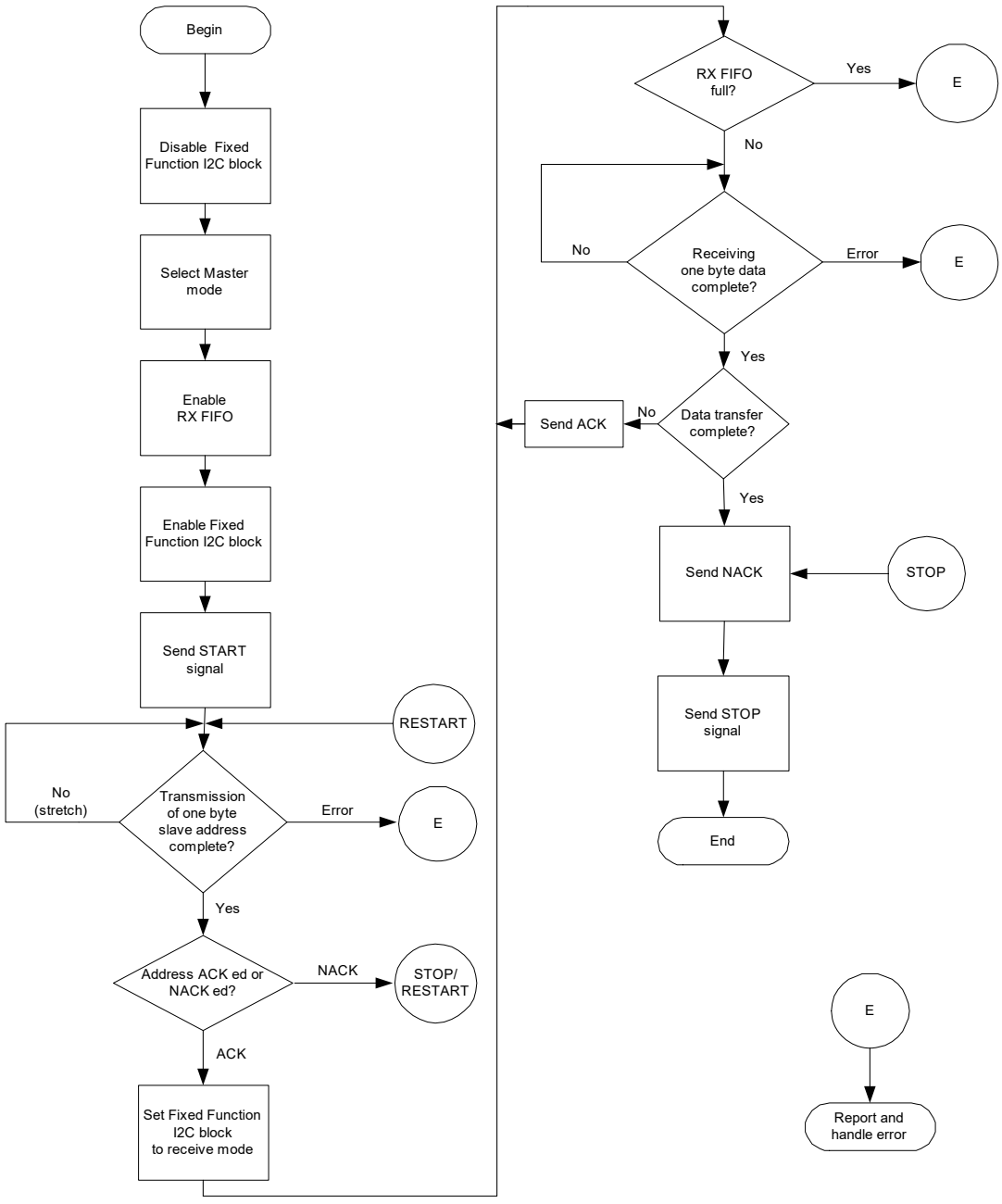
14.4.11.1 マスター送信

図 14-26. シングル マスター モードでの書き込み動作のフローチャート



14.4.11.2 マスター受信

図 14-27. シングル マスター モードでの読み出し動作のフローチャート

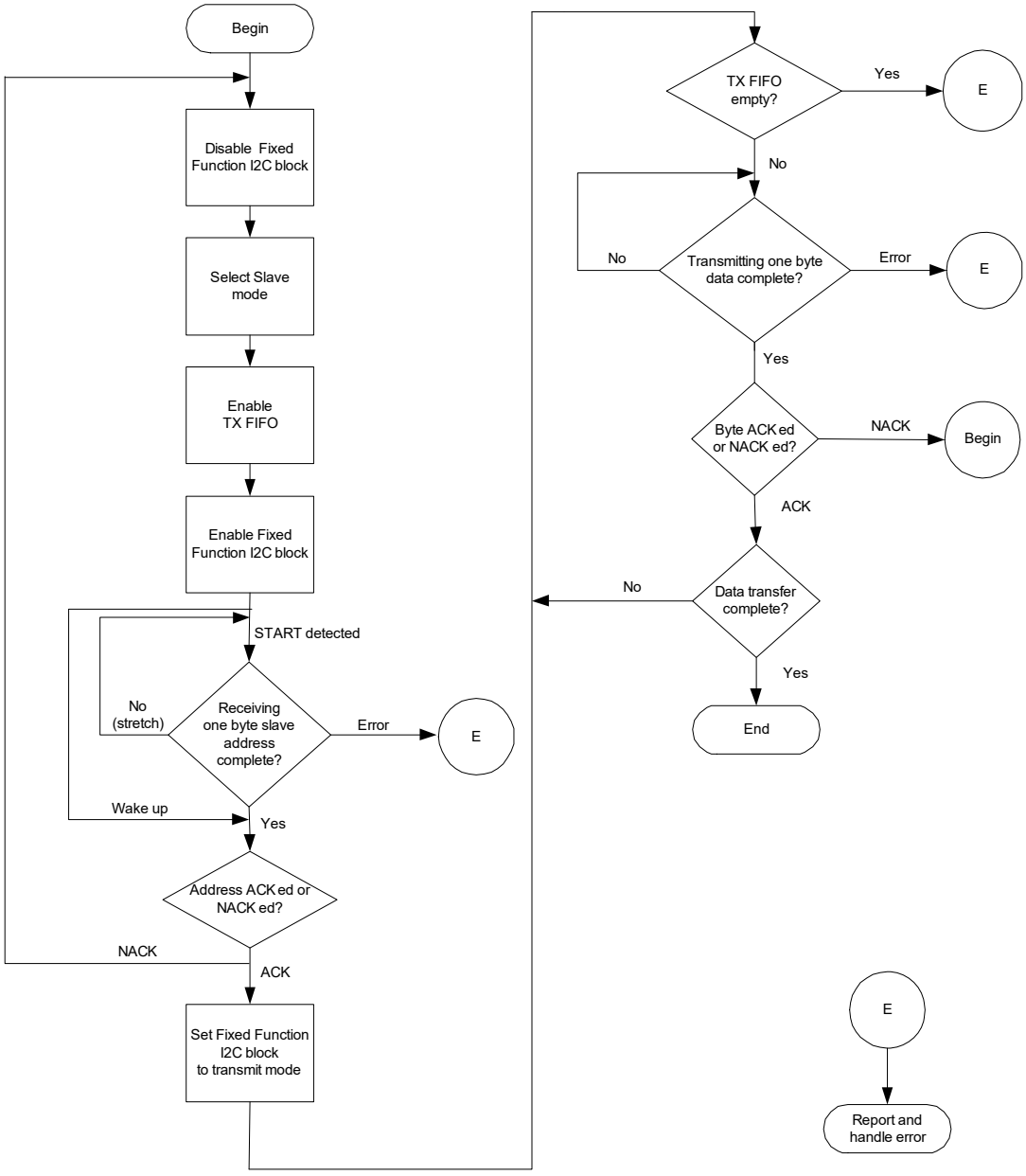


14.4.12　スレーブ モード転送の例

スレーブがデータを送信または受信します。

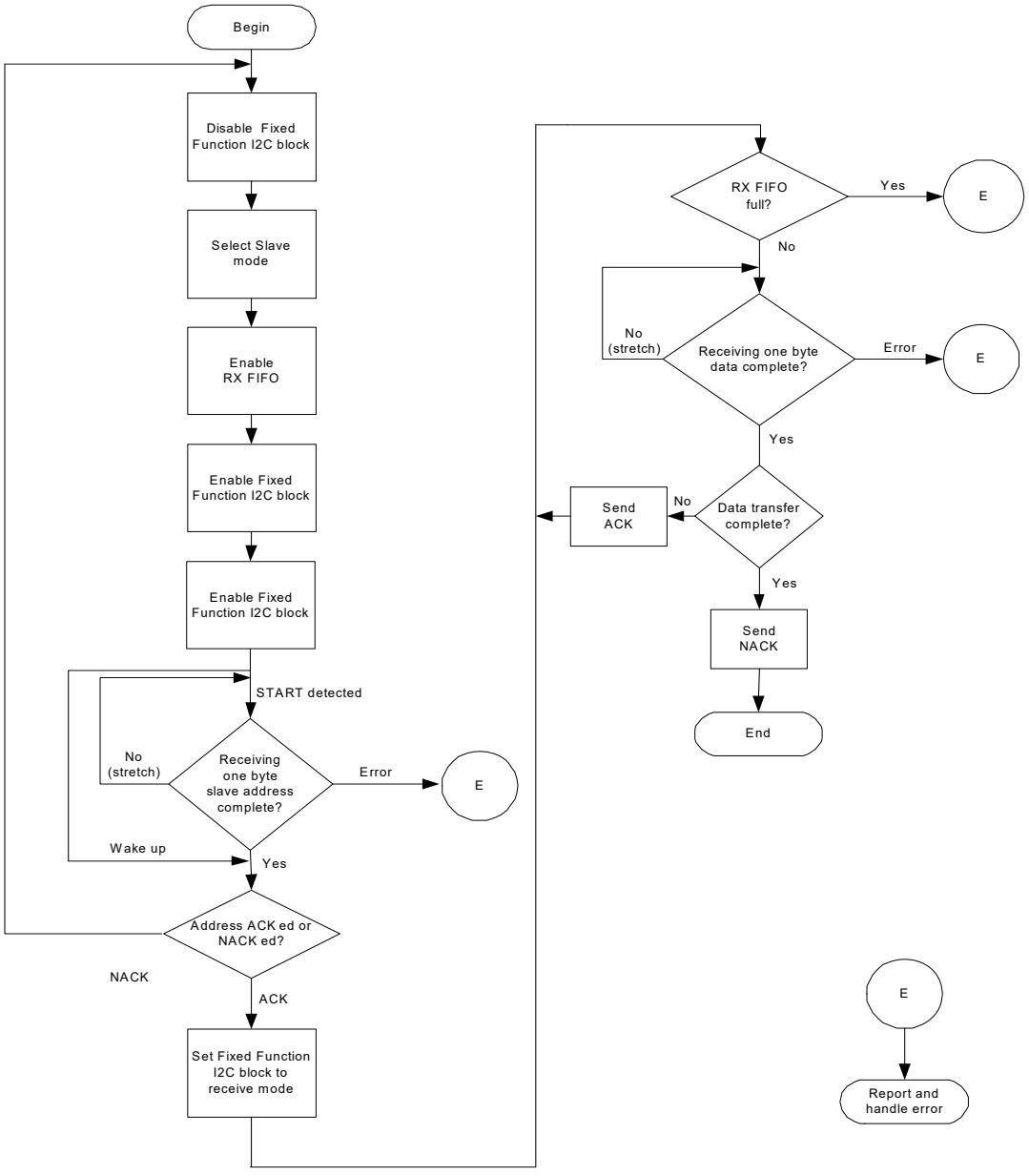
14.4.12.1　スレーブ送信

図 14-28. スレーブ モードでの書き込み動作のフローチャート



14.4.12.2 スレーブ受信

図 14-29. スレーブ モードでの読み出し動作のフローチャート

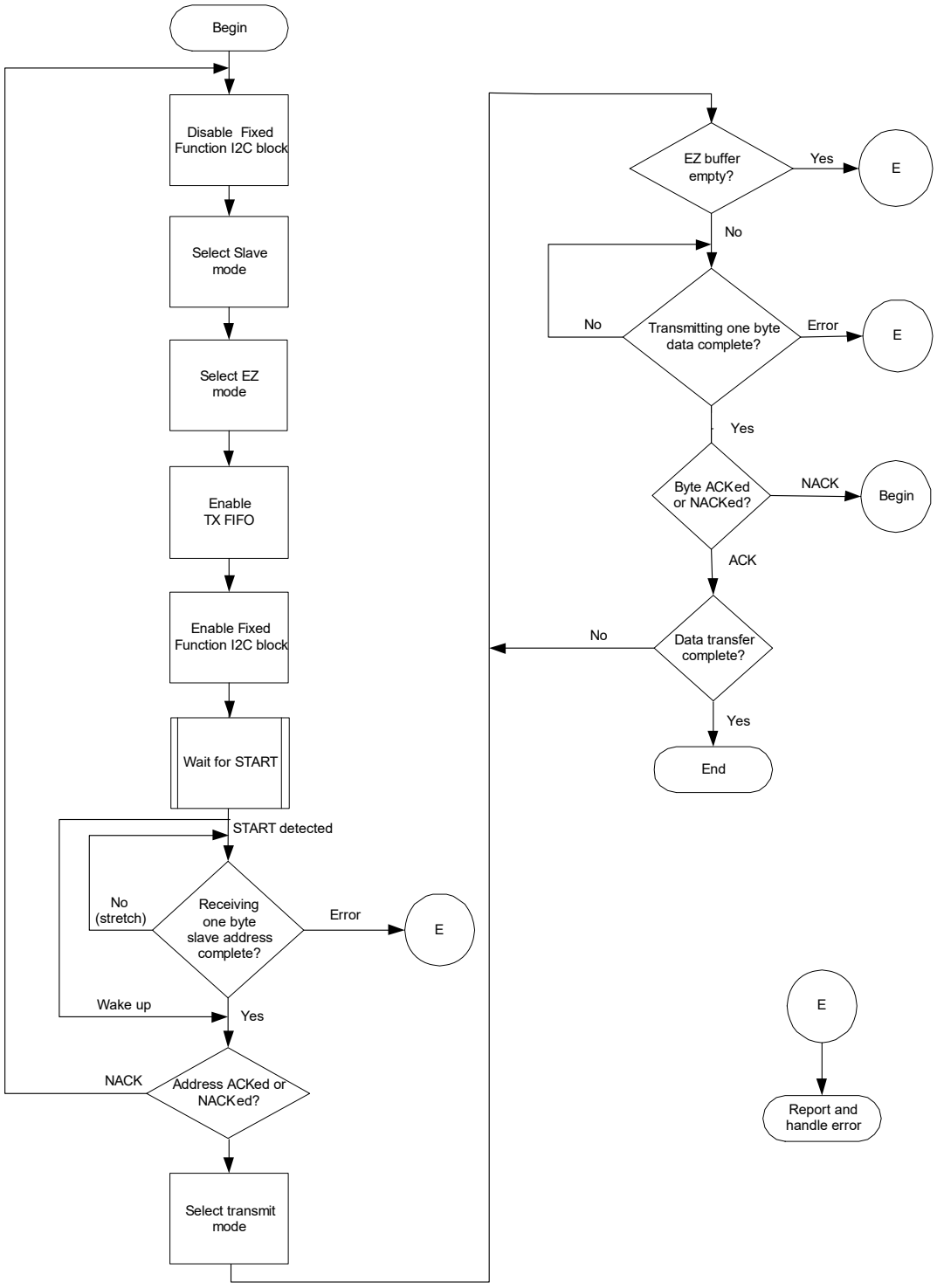


14.4.13 EZ スレーブ モード転送の例

EZ スレーブがデータを送信または受信します。

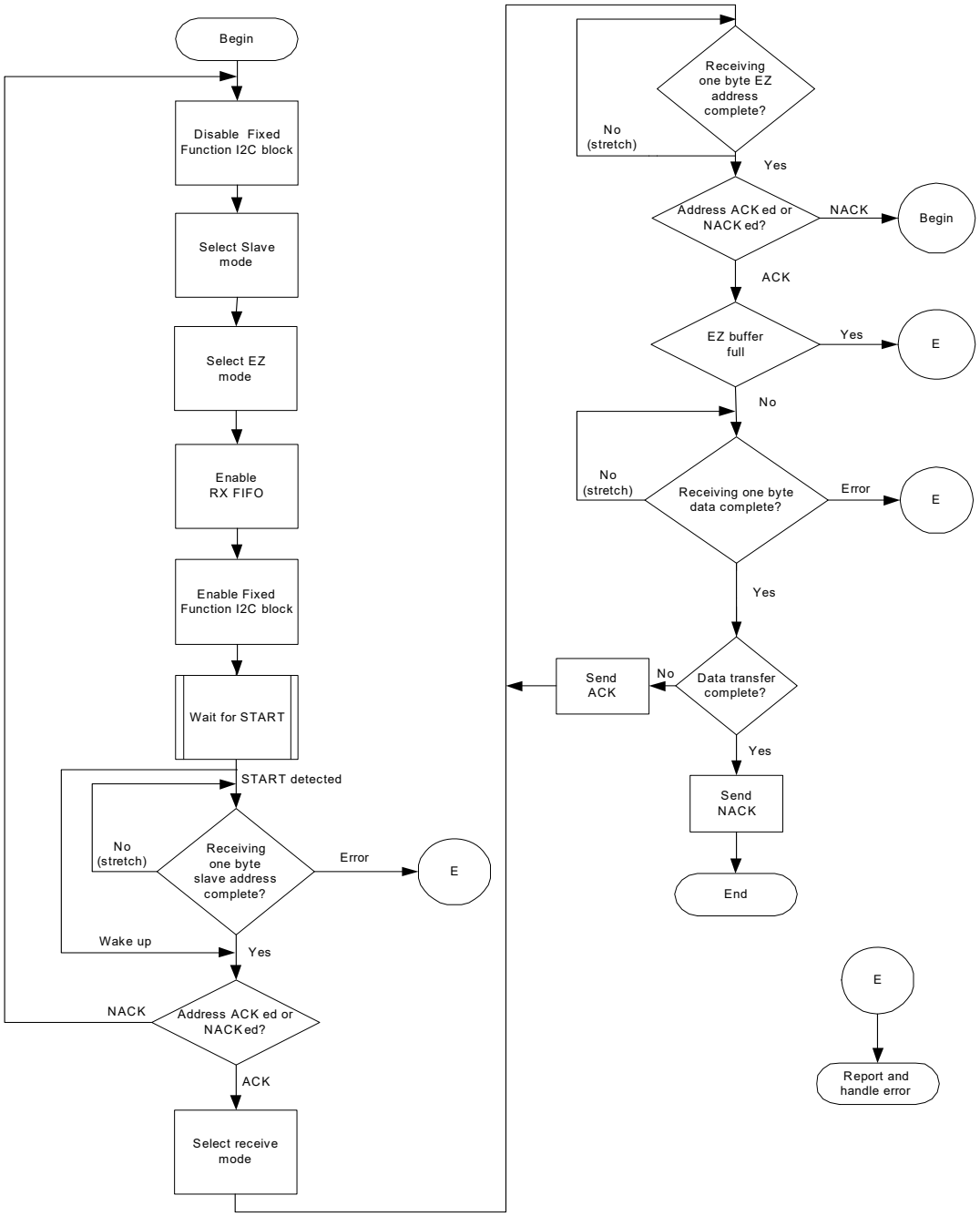
14.4.13.1 EZ スレーブ送信

図 14-30. EZI2C スレーブ モードでの書き込み動作のフローチャート



14.4.13.2 EZ スレーブ受信

図 14-31. EZI2C スレーブ モードでの読み出し動作のフローチャート

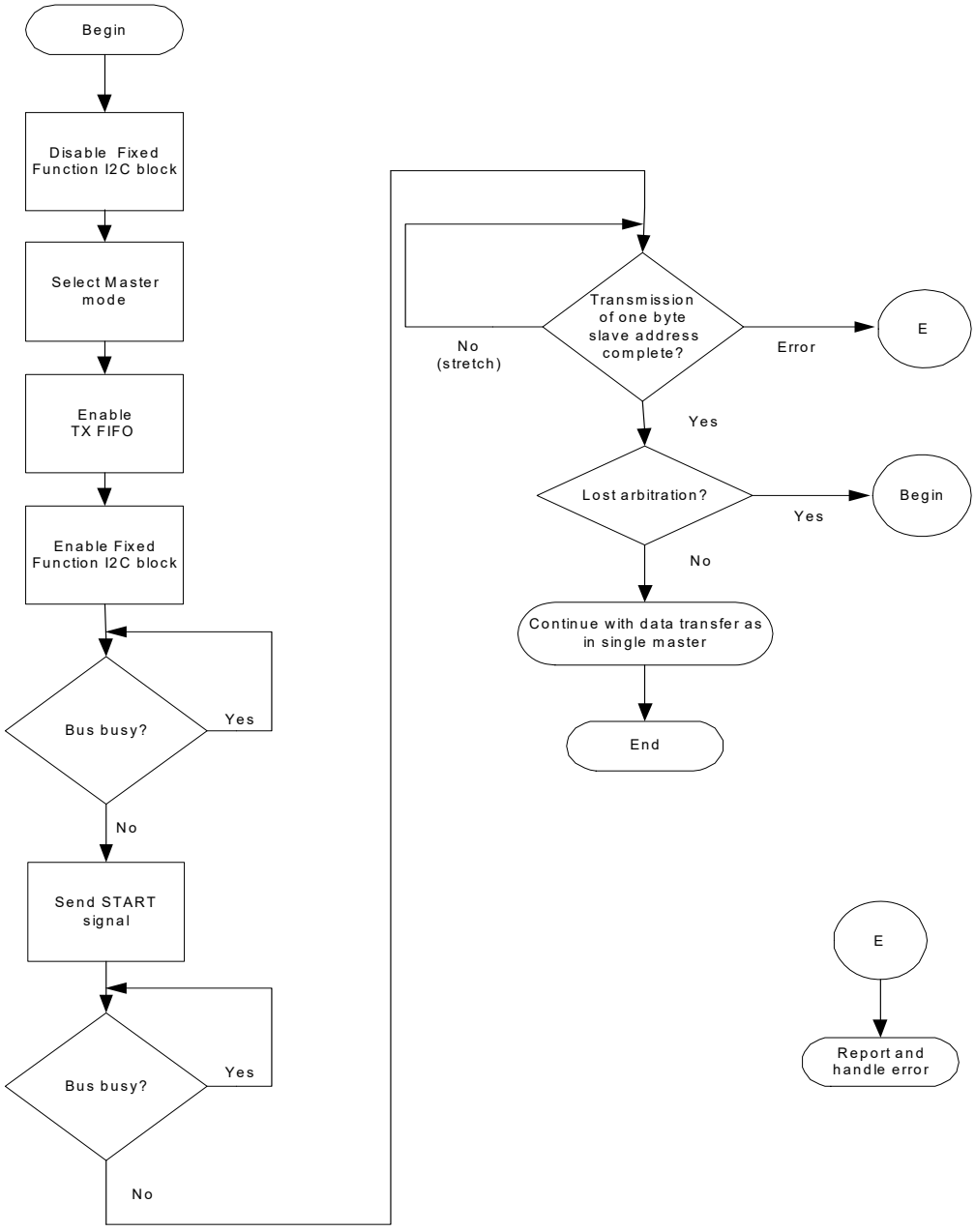


14.4.14　マルチマスター モード転送の例

マルチマスター モードでは、スレーブ モードが有効でも無効でもデータ転送が可能です。

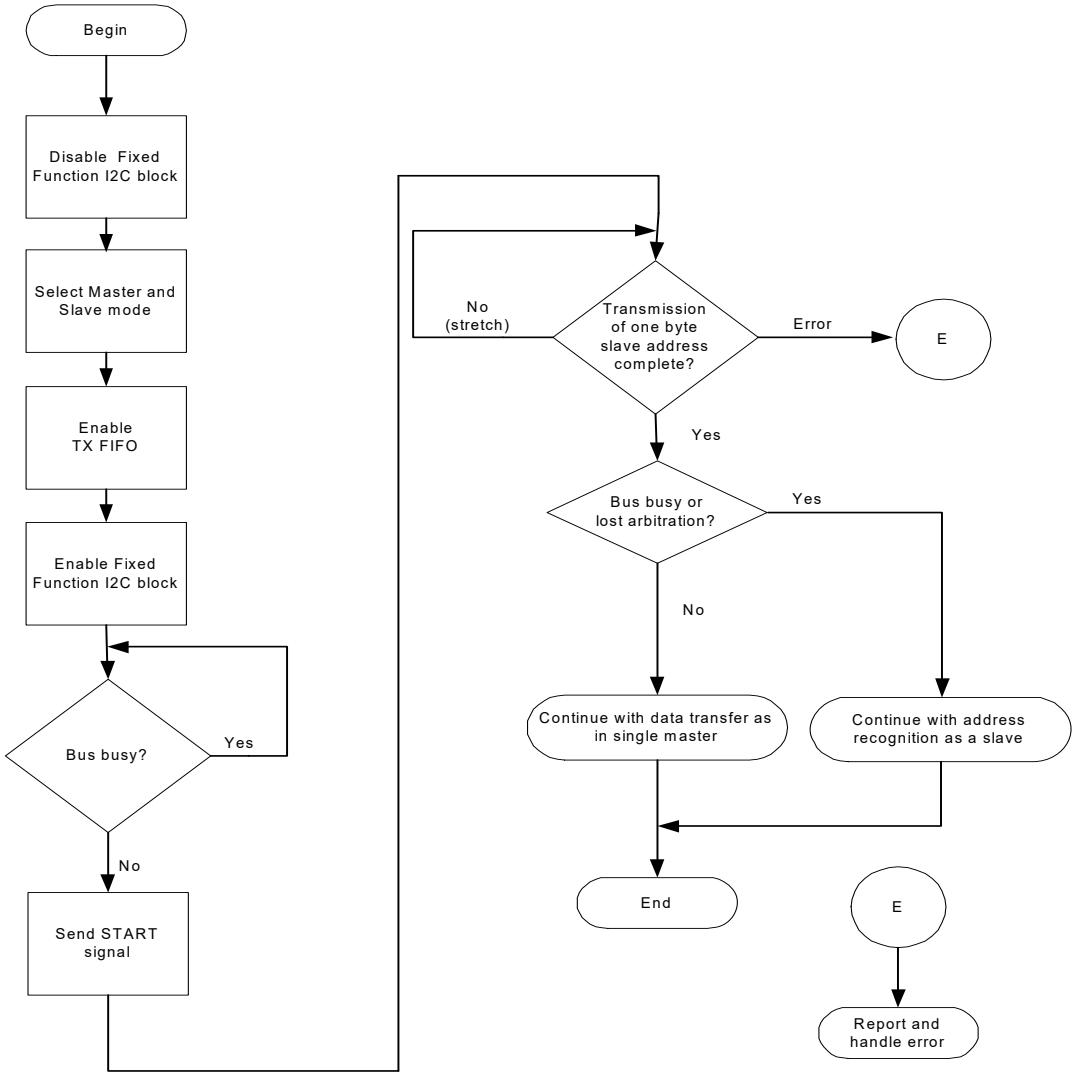
14.4.14.1　マルチマスター - スレーブが無効

図 14-32. 「マルチマスター、スレーブが無効」のフローチャート



14.4.14.2 マルチマスター - スレーブが有効

図 14-33. 「マルチマスター、スレーブが有効」のフローチャート



15. タイマー、カウンター、およびパルス幅変調器

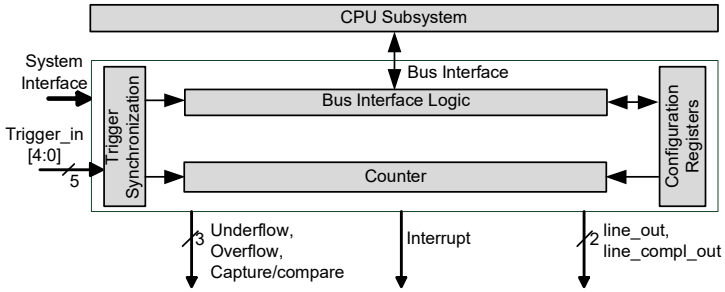
PSoC® 4 内のタイマー、カウンター、パルス幅変調器 (TCPWM) ブロックは 16 ビット タイマー、カウンター、パルス幅変調器 (PWM)、直交デコーダーの機能を搭載しています。ブロックは入力信号の周期とパルス幅の測定 (タイマー) や特定のイベントが発生する回数のカウント (カウンター)、PWM 信号の生成、直交信号の復号に使用します。本章は TCPWM ブロックの機能、実装、動作モードを説明します。

15.1 特長

- 5 個の 16 ビット タイマー、カウンターまたはパルス幅変調器 (PWM)
- TCPWM ブロックが対応している動作モード :
  - タイマー
  - キャプチャ
  - 直交復号
  - パルス幅変調
  - 疑似乱数 PWM
  - デッドタイム付き PWM
- 複数のカウント モード : アップ、ダウン、アップ／ダウン
- クロック分周 (1、2、4、...64、128 分周)
- 比較／キャプチャと周期の二重バッファ
- 以下のイベントの発生時の割り込みに対応 :
  - ターミナル カウント – カウンター レジスタの最終値に到達
  - キャプチャ／比較 – カウントがキャプチャ／比較レジスタに取り込まれる、またはカウントが比較値と一致
- アンダーフロー、オーバーフローおよびキャプチャ／比較の出力信号
- PWM のコンプリメンタリ ライン出力
- 他の TCPWM の TCPWM アンダーフロー、オーバーフローおよびキャプチャ／比較の信号による選択可能なスタート、リロード、ストップ、カウントおよびキャプチャのイベント信号、LPCOMP および立ち上がりエッジ、立ち下りエッジ、両エッジおよびトリガー レベル オプションの専用 GPIO からの出力

15.2 ブロック図

図 15-1. TCPWM ブロック図



ブロックには以下のインターフェースがあります :

- バス インターフェース : ブロックを CPU サブシステムに接続
- I/O 信号インターフェース : 入力トリガーを接続 (リロード、スタート、ストップ、カウントおよびキャプチャ)
- 割り込み : ターミナル カウント (TC) または CC 条件をベースとするカウンターからの割り込み要求信号の提供
- システム インターフェース : システム リソース サブシステムからのクロックやリセットなどの制御信号を包含

TCPWM ブロックは TCPWM レジスタに書き込むことで設定します。ブロックに必要なすべてのレジスタの詳細については 152 ページの「TCPWM レジスタ」を参照してください。

15.2.1 TCPWM ブロックにおけるカウンターの有効／無効

カウンターは制御レジスタ TCPWM_CTRL の COUNTER_ENABLED フィールド (ビット 0) をセットすることで有効にします。

注 : カウンターは有効にする前に設定する必要があります。カウンターを設定した後、有効にするとレジスタは新しいコンフィギュレーションで更新されます。カウンターが無効になると、再び有効になる (再設定される) までレジスタ値はそのままです。

15.2.2 クロッキング

TCPWM はシステム インターフェースを介して HFCLK を受信してブロック内のすべてのイベントを同期化します。カウンターが有効になると生成されたカウンター イネーブル信号 (counter_en) はカウンター固有のクロック (counter_clock) を供給するために HFCLK をゲート制御します。本章の後半で説明する出力トリガーも HFCLK と同期化されます。

クロック分周 : counter_clock は 1、2、4… 64、128 の分周比で分周できます。この分周は表 15-1 に示すようにカウンター制御 (TCPWM_CNT_CTRL) レジスタの GENERIC フィールドを変更することで行います。

表 15-1. カウンター クロック分周のビット フィールド設定

GENERIC[10:8]	説明
0	1 分周
1	2 分周
2	4 分周
3	8 分周
4	16 分周
5	32 分周
6	64 分周
7	128 分周

注 : クロック分周は、直交モードおよび PWM-DT モードでは行えません。

15.2.3 トリガー入力に基づいたイベント

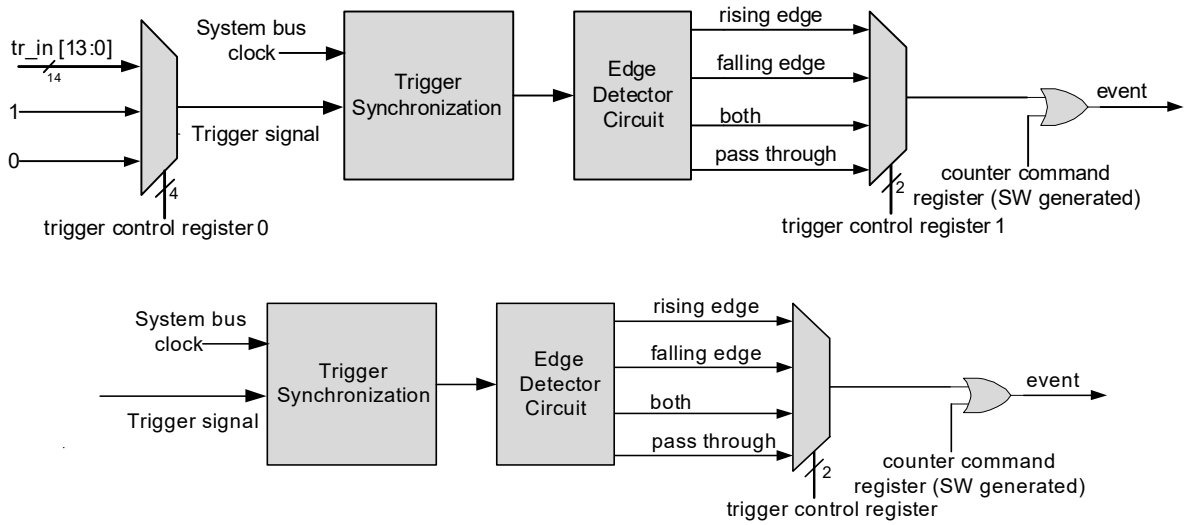
以下はハードウェアまたはソフトウェアによってトリガーされるイベントです。

- リロード
- スタート
- ストップ
- カウント
- キャプチャ／切り替え

ハードウェア トリガーはレベル信号、立ち上がり、立ち下がり、または両エッジにすることができます。図 15-2 に、イベント トリガー信号にエッジ検出タイプの選択を示します。

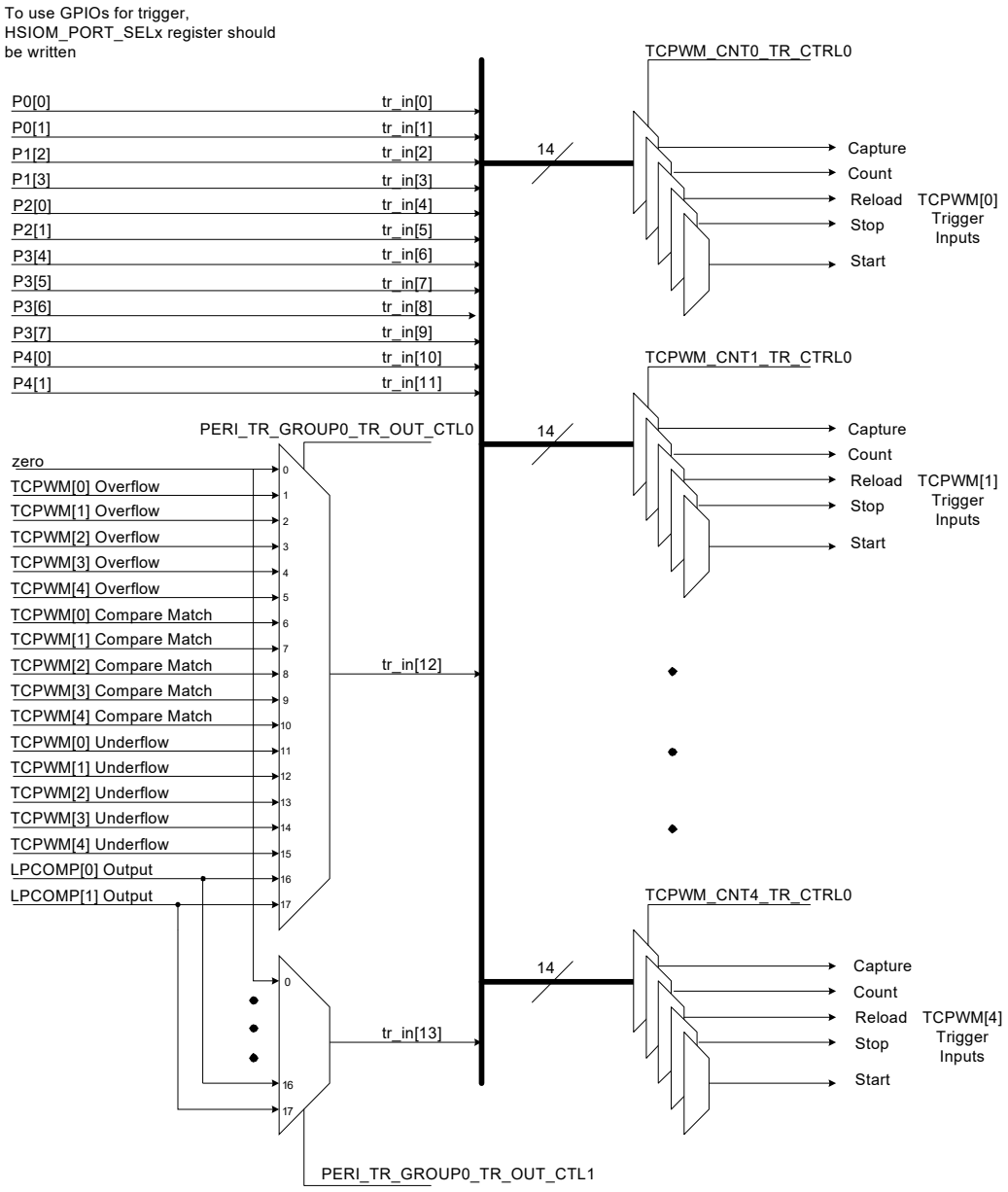
トリガー制御レジスタ 1 (TCPWM_CNT_TR_CTRL1) を設定することでエッジ (立ち上がりエッジ、立ち下がりエッジ、両エッジ) またはレベル (HIGH) のいずれをもイベント発生に選択することができます。このエッジ／レベル コンフィギュレーションはトリガーごとに選択できます。あるいは図 15-2 に示すようにカウンター コマンド レジスタ (TCPWM_CMD) に書き込むことでファームウェアでイベントを生成することができます。

図 15-2. トリガー信号エッジ検出



イベントを生成するためのトリガー信号は GPIO 信号、TCPWM のアンダーフロー、比較一致またはオーバーフロー信号、または低消費電力コンパレータ (LPCOMP) 出力信号にすることができます。図 15-3 に、すべてのイベントによるトリガー信号選択を示します。

図 15-3. トリガー マルチプレクサ



これらのトリガーから派生するイベントは TCPWM ブロックのモードによって定義が異なることがあります。

- **リロード**：リロード イベントはカウンターを初期化し、スタート
  - カウントアップ　モードでは、カウント　レジスタ (TCPWM_CNT_COUNTER) が「0」で初期化
  - カウント ダウン モードでは、カウンターが TCPWM_CNT_PERIOD レジスタに格納された周期で初期化
  - カウント アップ／ダウン モードでは、カウント レジスタが「0」で初期化
  - 直交モードでは、リロード イベントは直交インデックス イベントとして機能インデックス／リロード イベントは完了した回転を示し、直交復号を同期化するために使用可能
- **スタート**: スタート イベントはカウントを開始するために使用されます。ソフトウェアでストップ イベントまたはカウンター レジスタのある値への再初期化の後に使用できます。カウント レジスタはこのイベントでは初期化されないことにご注意ください。
  - 直交モードではスタート　イベントは 141 ページの「直交デコーダー モード」で説明する直交位相入力 phiB として使えます
- **カウント**: カウント イベントではカウンターはコンフィギュレーションに応じてインクリメント／デクリメントします。
  - 直交モードではカウント　イベントは直交位相入力 phiA として使えます
- **ストップ**: ストップ イベントはカウンターのインクリメント／デクリメントを停止します。スタート イベントでカウントは再び開始します。
  - PWM モードではストップ イベントはキル イベントとなります。キル イベントではすべての PWM 出力ラインが無効となります。
- **キャプチャ**：キャプチャ イベントでカウンター レジスタの値がキャプチャ レジスタにコピーされ、キャプチャ レジスタの値がバッファ キャプチャ レジスタにコピーされます。PWM モードではキャプチャ イベントはスイッチ イベントとなります。キャプチャ／比較レジスタと周期レジスタに付属するバッファレジスタの値と切り替えます。この機能はパルス幅および周波数を変調するために使用できます。

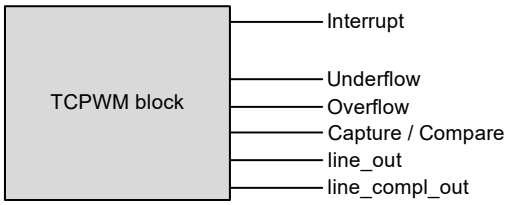
注

- すべてのトリガー入力は HFCLK と同期します。
- 複数のイベントが同一のカウンター　クロック周期で発生する時、1 つ以上のイベントを見逃すことがあります。これは高周波イベント (カウンター周波数に近い周波数) と分周 (分割) カウンター クロックが使用されるタイマー コンフィギュレーションのために発生することがあります。

15.2.4　出力信号

図 15-4 に示すように TCPWM ブロックは複数の出力信号を生成します。

図 15-4. TCPWM 出力信号



15.2.4.1　トリガー条件発生時の信号

- カウントアップ モードでカウント レジスタが周期値に達するとカウンターは内部オーバーフロー (OV) 条件を生成します。
- カウントダウン モードでカウント レジスタが 0 に達するとカウンターは内部アンダーフロー (UN) 条件を生成します。
- カウンターが実行中で、以下のいずれかの条件が発生すると TCPWM はキャプチャ／比較 (CC) を生成します :
  - カウント一値が比較値と等しい
  - キャプチャ イベントが発生 ー キャプチャ イベントが発生すると TCPWM_CNT_COUNTER レジスタの値はキャプチャ レジスタにコピーされ、キャプチャ レジスタの値がバッファ キャプチャ レジスタにコピーされます。

注 : これらの信号が発生すると HFCLK の 2 サイクルの間論理 HIGH を維持します。信頼できる動作のためにはトリガーを発生させる条件は HFCLK の 1/4 未満である必要があります。たとえば HFCLK の周波数が 24MHz である場合、トリガーが発生する条件は 6MHz 未満の周波数となります。

15.2.4.2 割り込み

TCPWM ブロックはカウンターから専用割り込み出力信号を提供します。TC 条件または CC 条件で割り込みが生成されます。これらの条件の正確な定義はモードによって異なります。

表 15-2 に示すようにこのブロックでは 4 本のレジスタが割り込み処理に用いられます。

表 15-2. 割り込みレジスタ

割り込みレジスタ	ビット	名称	説明
TCPWM_CNT_INTR (割り込み要求レジスタ)	0	TC	ターミナル カウント検出時、このビットは「1」にセット。このビットのクリアは「1」を書き込む
	1	CC_MATCH	カウント値がキャプチャ／比較のレジスタ値と一致時、このビットは「1」にセット。このビットのクリアは「1」を書き込む
TCPWM_CNT_INTR_SET (割り込みセット要求レジスタ)	0	TC	割り込み要求レジスタの対応するビットのセットは「1」を書き込む。読み出し時、このレジスタは割り込み要求レジスタの状態を示す
	1	CC_MATCH	割り込み要求レジスタの対応するビットのセットは「1」を書き込む。読み出し時、このレジスタは割り込み要求レジスタの状態を示す
TCPWM_CNT_INTR_MASK (割り込みマスク レジスタ)	0	TC	割り込み要求レジスタの対応する TC ビットのマスク ビット
	1	CC_MATCH	割り込み要求レジスタの対応する CC_MATCH ビットのマスク ビット
TCPWM_CNT_INTR_MASKED (割り込みマスク要求レジスタ)	0	TC	対応する TC 要求とマスク ビットの論理積
	1	CC_MATCH	対応する CC_MATCH 要求とマスク ビットの論理積

15.2.4.3 出力

TCPWM には line_out と line_compl_out (line_out のコンプリメンタリ) の 2 本の出力があります。必要に応じて TCPWM_CNT_TR_CTRL2 レジスタを設定することで OV、UN、CC 条件を使って line_out と line_compl_out を駆動することができることにご注意ください (表 15-3 を参照してください)。

表 15-3. OV、UN、CC 条件の発生時の出力ライン コンフィギュレーション

フィールド	ビット	値	イベント	説明
CC_MATCH_MODE デフォルト値 = 3	1:0	0	line_out を「1」にセット	比較一致 (CC) イベントの発生時に出力ラインを設定
		1	line_out を「0」にクリア	
		2	line_out を反転	
		3	変更なし	
OVERFLOW_MODE デフォルト値 = 3	3:2	0	line_out を「1」にセット	オーバーフロー (OV) イベントの発生時に出力ラインを設定
		1	line_out を「0」にクリア	
		2	line_out を反転	
		3	変更なし	
UNDERFLOW_MODE デフォルト値 = 3	5:4	0	line_out を「1」にセット	アンダーフロー (UN) イベントの発生時に出力ラインを設定
		1	line_out を「0」にクリア	
		2	line_out を反転	
		3	変更なし	

15.2.5 電力モード

TCPWM ブロックはアクティブとスリープ モードで動作します。TCPWM ブロックは V_{CCD} から電源供給されます。コンフィギュレーション レジスタおよびその他の回路はコンフィギュレーション レジスタの状態を維持するためにディープスリープ モードでも電源供給されます。詳細は表 15-4 を参照してください。

表 15-4. TCPWM ブロックの消費電力モード

消費電力モード	ブロックの状態
アクティブ	ブロックは完全に動作可能で、クロックが供給され、電源がオン
スリープ	すべてのカウンター クロックがオンであるが、バス インターフェースにはアクセス不可能
ディープスリープ	ブロックへの電源がオンであるがバス クロックがないためブロックが動作しない。すべてのコンフィギュレーションレジスタは状態を維持

15.3 動作モード

表 15-5 に示すようにカウンター ブロックは 6 つの動作モードで機能します。カウンター制御レジスタ (TCPWM_CNTx_CTRL) の MODE[26:24] フィールドはカウンターを特定の動作モードに設定します。

表 15-5. 動作モード コンフィギュレーション

モード	MODE フィールド [26:24]	説明
タイマー	000	タイマーまたはカウンターを搭載。カウント イベントが検出されるカウンター クロック サイクルごとに「1」ずつインクリメント／デクリメント
キャプチャ	010	キャプチャ入力を備えるタイマーまたはカウンターを搭載。カウント イベントが検出されるカウンター クロック サイクルごとに「1」ずつインクリメント／デクリメント。キャプチャ イベントが発生するとカウントはキャプチャ レジスタにコピーされる
直交デコーダー	011	選択した (X1、X2、X4) 符号化スキームに応じて二相入力に基づいてカウンターがデクリメント／インクリメントする直交デコーダーを搭載
PWM	100	8 ビット クロック プリスケーラおよびバッファ付き比較／周期レジスタを備えるエッジ／中央揃え PWM を搭載
PWM-DT	101	設定可能な 8 ビット デッド タイム (両出力) およびバッファ付き比較／周期レジスタを備えるエッジ／中央揃え PWM を搭載
PWM-PR	110	16 ビット線形帰還シフトレジスタ (LFSR) を用いる疑似乱数 PWM を搭載

表 15-6 に示すように TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE[17:16] フィールドを設定することでカウンターをカウントアップ、ダウン、アップ／ダウンに設定することができます。

表 15-6. カウント モード コンフィギュレーション

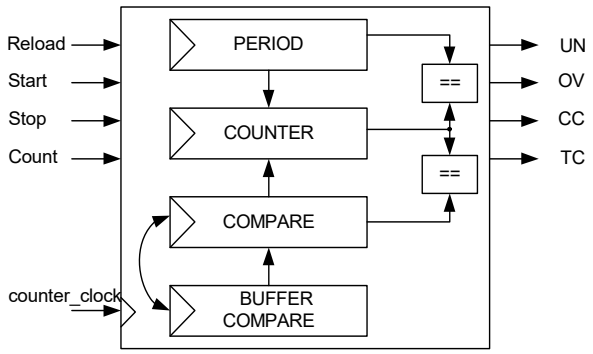
カウント モード	UP_DOWN_MODE[17:16]	説明
カウントアップ モード	00	周期値に達するまでカウンターをインクリメント。カウントが周期値に達すると、ターミナル カウント (TC) 条件が生成される
カウントダウン モード	01	「0」に達するまで周期値からカウンターをデクリメント。カウンターが「0」に達すると、TC 条件が生成される
カウントアップ／ダウン モード 0	10	周期値に達するまでカウンターをインクリメントしてから、「0」に達するまでカウンターをデクリメント。「0」に達する時にのみ TC 条件が生成される
カウント アップ／ダウン モード 1	11	カウント アップ／ダウン モード 0 に似ているが、カウンターが「0」に達する時にもカウンターが周期値に達する時にも TC 条件が生成される

15.3.1 タイマー モード

一般的にタイマー モードはイベントの継続時間を測定するか、2 つのイベント間の時間差を測定するために使用されます。

15.3.1.1 ブロック図

図 15-5. タイマー モード ブロック図



15.3.1.2 動作原理

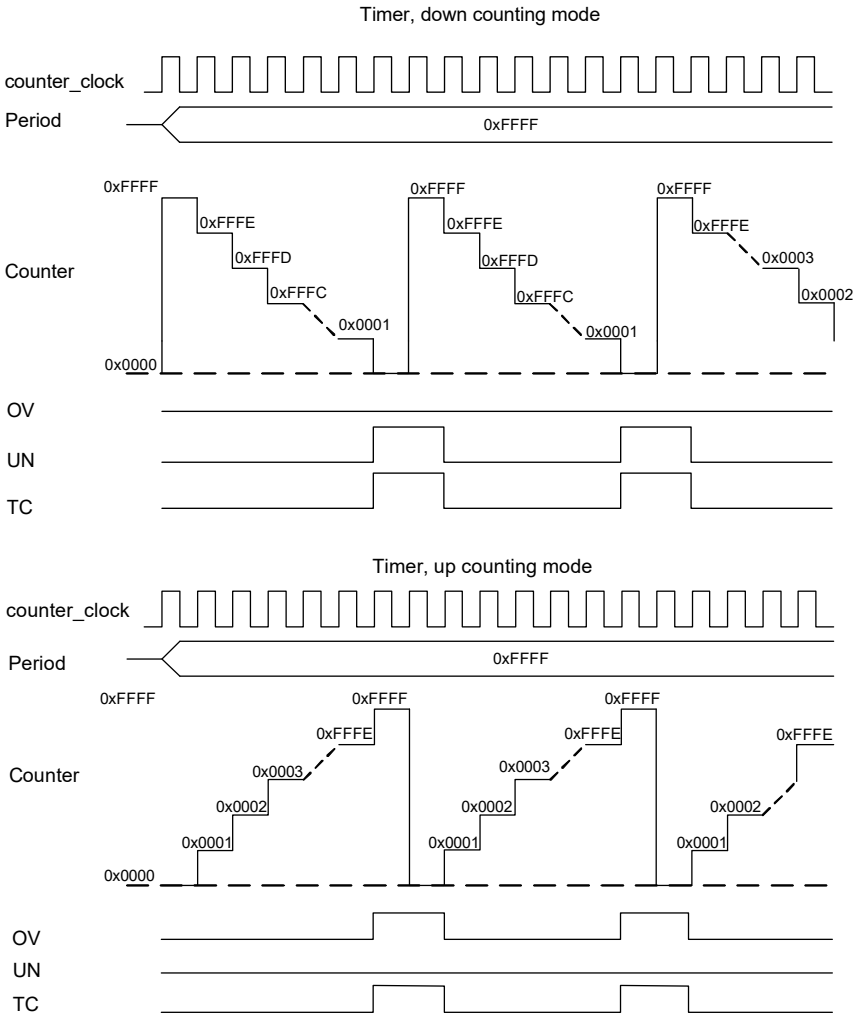
タイマーはカウントアップ、ダウン、アップ／ダウン モードに設定できます。さらに連続モードまたはワンショット モードに設定することもできます。以下にタイマーの動作原理を説明します :

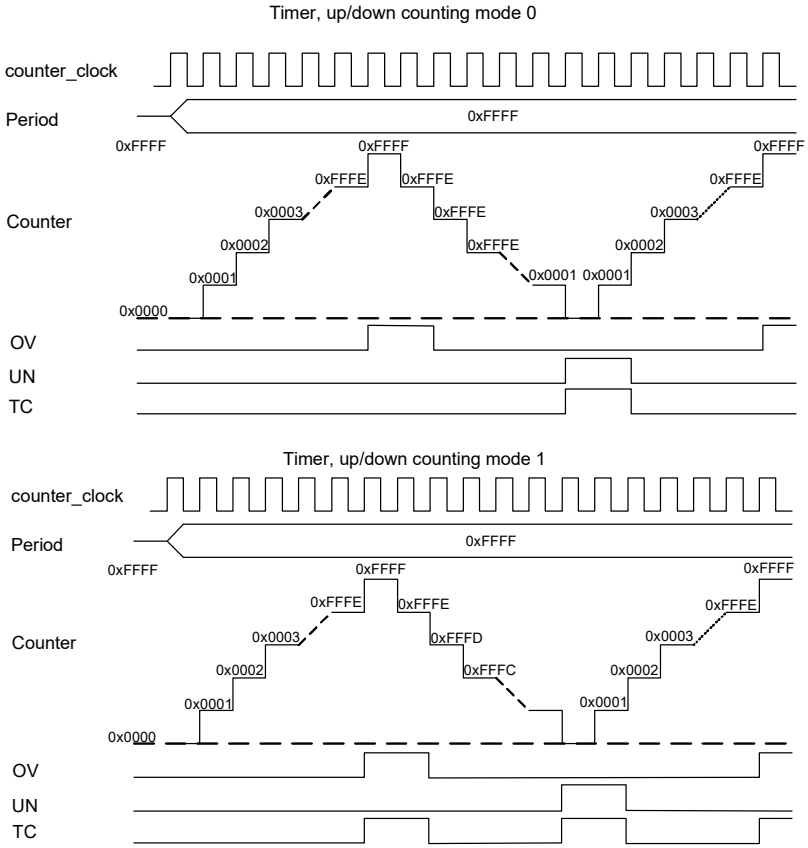
- タイマーはアップ、ダウン、アップ／ダウンのカウンターです。
  - 現時点のカウント値はカウント レジスタ (TCPWM_CNTx_COUNTER) に格納されます。
    注 : カウンターの動作中、このレジスタに書き込まないでください。
  - タイマーの周期は周期レジスタに格納されます。
- カウンターは以下のように異なるモードで再初期化されます :
  - カウントアップ モードでは、カウント レジスタは周期値に達すると自動的に 0 にリロードされます
  - カウントダウン モードでは、カウント レジスタが 0 に達すると周期レジスタの値にリロードされます
  - カウントアップ／ダウン モードでは、カウント レジスタは最終値に達した時更新されません。その代わりにカウント値が 0 または周期値に達するとカウント方向が変化します
- カウント レジスタの値が比較レジスタの値と等しいと CC 条件が生成されます。この条件の発生時に、カウンター制御レジスタ (TCPWM_CNT_CTRL) の AUTO_RELOAD_CC ビット フィールドが有効であれば、比較レジスタとバッファ比較レジスタは値を切り替えます。この条件は割り込み要求を生成するために使用できます。

図 15-6 に 4 つのカウント モードにおけるタイマーの動作モードを示します。周期レジスタは最大カウントを格納します。

- カウントアップ モードでは周期 A は A+1 カウンター サイクル (0 から A まで) となります。
- カウントダウン モードでは周期 A は A+1 カウンター サイクル (A から 0 まで) となります。
- 2つのカウントアップ／ダウン モード (モード0と1) では周期Aは2*Aカウンター サイクル (0からAまで、そして0に戻る) となります。

図 15-6. 各カウント モードにおけるタイマーのタイミング図





注 : 132 ページの「トリガー条件発生時の信号」で説明したように、OV と UN 信号は HFCLK の 2 サイクルの間、論理 HIGH を維持します。本章の図は HFCLK とカウンター クロックが同じと想定しています。

15.3.1.3 タイマー モードの設定方法

以下にカウンターをタイマー動作モードに設定する手順および影響されるレジスタ ビットを示します。

1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「000」を書き込んでタイマー モードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期を設定します。
4. TCPWM_CNT_CC レジスタに 16 ビット比較値を設定し、TCPWM_CNT_CC_BUFF レジスタにバッファ比較値を設定します。
5. CC 条件が発生すると値を切り替える必要がある場合、TCPWM_CNT_CTRL レジスタの AUTO_RELOAD_CC フィールドをセットします。
6. 表 15-1 に示すように、TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込んでクロック分周を設定します。
7. 表 15-6 に示すように TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE[17:16] フィールドに書き込んでカウント方向を設定します。
8. TCPWM_CNT_CTRL レジスタの ONE_SHOT[18] フィールドに 0 か 1 を書き込んでタイマーを連続モードかワンショット モードに設定します。
9. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (リロード、スタート、ストップ、キャプチャ、カウント) を発生させるトリガーを選択します。
10. TCPWM_CNT_TR_CTRL1 レジスタを設定してイベント (リロード、スタート、ストップ、キャプチャ、カウント) を発生させるトリガーのエッジを選択します。

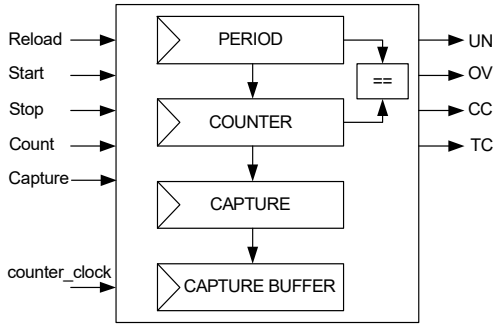
11. 必要に応じて 133 ページの「割り込み」に示すように、TC または CC 条件の発生時の割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドへの「0」の書き込みによりカウンターを有効にします。ハードウェア開始信号が有効になっていない場合、カウンターを開始するためにファームウェア (TCPWM_CMD レジスタ) でスタート トリガーを与える必要があります。

15.3.2 キャプチャ モード

キャプチャ モードではコマンドレジスタ (TCPWM_CMD) へのファームウェア書き込みまたはキャプチャ トリガー入力によりカウントをいつでも取り込むことができます。このモードは周期値とパルス幅の測定に使用されます。

15.3.2.1 ブロック図

図 15-7. キャプチャ モード ブロック図



15.3.2.2 動作原理

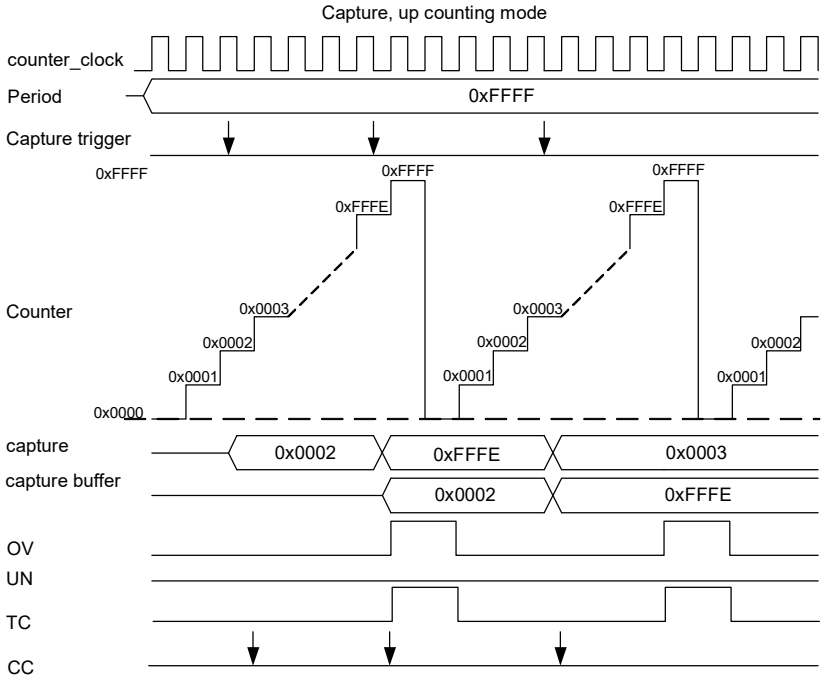
カウンター制御レジスタ (TCPWM_CNT_CTRL) の UP_DOWN_MODE[17:16] ビット フィールドを設定することでカウンターをアップ、ダウン、アップ／ダウンのモードに設定することができます。

キャプチャ モードにおける動作は以下の通りです :

- ハードウェアまたはソフトウェアによって生成されたキャプチャ イベント中に、現時点のカウント レジスタの値はキャプチャ レジスタ (TCPWM_CNT_CC) にコピーされ、キャプチャ レジスタの値はバッファ キャプチャ レジスタ (TCPWM_CNT_CC_BUFF) にコピーされます。
- カウントがキャプチャ レジスタにコピーされると CC 出力信号にパルスが生成されます。この条件は割り込み要求を生成するために使用できます。

図 15-8 にカウントアップ モードにおけるキャプチャ動作を示します。

図 15-8. キャプチャ モード - カウントアップ モードのカウンターのタイミング図



図に示すように、以下のことが分かります :

- 周期レジスタは最大カウント値を格納します。
- カウンターが周期値に達すると、内部オーバーフロー (OV) と TC 条件が生成されます。
- キャプチャ イベントの発生はエッジまたはソフトウェアによってのみ可能です。トリガー制御レジスタ1を使用してエッジ検出を設定します。
- 1 クロック サイクルに複数のキャプチャ イベントが発生した場合キャプチャ イベントは以下のように処理されます :
  - 偶数のキャプチャ イベントがある場合 : イベントは監視されません。
  - 奇数のキャプチャ イベントがある場合 : 1 つのイベントが監視されます。

これはキャプチャ信号周波数が counter_clock 周波数より高い場合実行されます。

15.3.2.3 キャプチャ モードの設定方法

以下にカウンターをキャプチャ動作モードに設定する手順および影響されるレジスタ ビットを示します。

1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「010」を書き込んでキャプチャ モードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期を設定します。
4. 表 15-1 に示すように、TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込んでクロック分周を設定します。
5. 表 15-6 に示すように TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE[17:16] フィールドに書き込んでカウント方向を設定します。
6. TCPWM_CNT_CTRL レジスタの ONE_SHOT[18] フィールドに 0 か 1 を書き込んでカウンターを連続モードかワンショット モードに設定します。
7. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (リロード、スタート、ストップ、キャプチャ、カウント) を発生させるトリガーを選択します。
8. TCPWM_CNT_TR_CTRL1 レジスタを設定してイベント (リロード、スタート、ストップ、キャプチャ、カウント) を発生させるエッジを選択します。

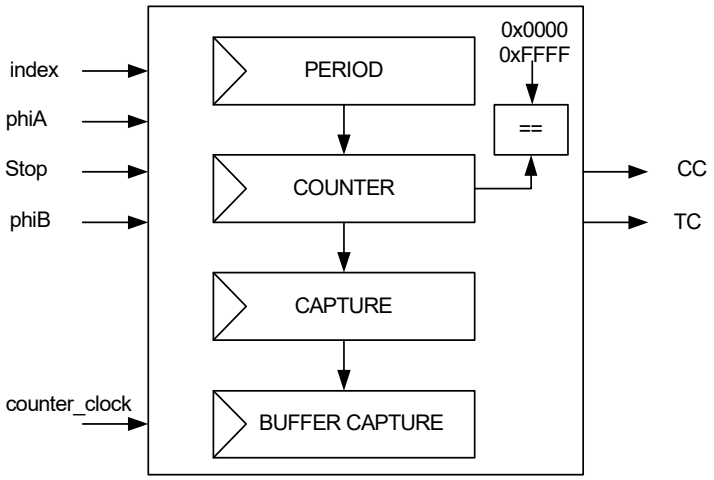
9. 必要に応じて 133 ページの「割り込み」に示すように、TC または CC 条件の発生時の割り込みを設定します。
10. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを有効にします。ハードウェア開始信号が有効になっていない場合、カウンターを開始するためにファームウェア (TCPWM_CMD レジスタ) でスタート トリガーを与える必要があります。

15.3.3 直交デコーダー モード

直交デコーダーは回転機器 (サーボモーター、ボリューム コントロール部、PC マウスなど) の速度と位置を判定するために使用されます。直交デコーダー信号はデコーダーの phiA と phiB 入力が用いられます。

15.3.3.1 ブロック図

図 15-9. 直交モード ブロック図



15.3.3.2 動作原理

直交デコーダーは counter_clock にのみ従って動作します。X1、X2、X4 の 3 つのサブモードで動作できます。これらの符号化モードはカウンター制御レジスタ (TCPWM_CNT_CTRL) の QUADRATURE_MODE[21:20] フィールドで制御します。このモードは二重バッファ キャプチャ レジスタを使用します。

直交モードにおける動作は以下の通りです :

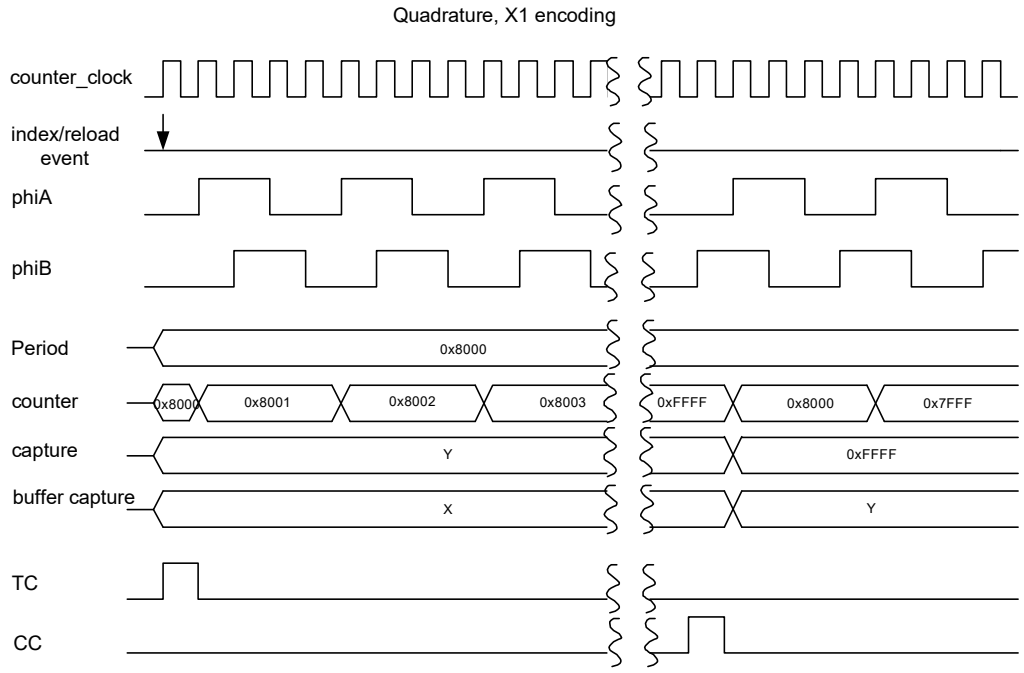
- 直交位相phiAとphiB: カウント方向はphiAとphiBの位相関係で決まります。2 つの入力信号はそれぞれデコーダーのハードウェア入力のカウント入力とスタート トリガー入力に接続します。
- 直交インデックス信号 : これはハードウェア入力のリロード信号に接続します。このイベントは図 15-10 に示すように TC 条件を生成します。

  TC 条件の発生時にカウンターは 0x0000 (カウントアップ モードの場合) または周期値 (カウントダウン モード) に設定されます。

  注 : カウントダウン モードでは周期値を 0x8000 (中点値) に設定することが推奨されます。
- カウント レジスタが 0x0000 または 0xFFFF に達すると CC 出力信号にパルスが生成されます。CC 条件の発生時、カウント レジスタは周期値 (この場合 0x8000) に設定されます。
- TC または CC 条件の発生時 :
  - カウント レジスタの値はキャプチャ レジスタにコピーされます。
  - キャプチャ レジスタの値はバッファ キャプチャ レジスタにコピーされます。
  - この条件は割り込み要求を生成するために使用できます。
- キャプチャ レジスタの値はイベントを発生させる条件および以下のことを判定します :
  - カウンター アンダーフローが発生したか (値 0)
  - カウンター オーバーフローが発生したか (値0xFFFF)
  - インデックス／TC イベントが発生したか (0か0xFFFFと等しくない値)
- カウンター ステータス レジスタ (TCPWM_CNTx_STATUS) の DOWN ビット フィールドを読み出して現時点のカウント方向を判定することができます。「0」はこれまでインクリメント動作であることを示し、「1」はデクリメント動作であることを示します。図 15-10 に X1 符号化モードでの直交動作を示します。
  - phiA の立ち上がりエッジで phiB が「0」であればカウンターはインクリメントし、phiB が「1」であればカウンターはデクリメントします。
  - カウント レジスタはインデックス／リロード イベントの発生時に周期値に初期化されます。
  - カウンターがインデックス イベントで初期化されるとターミナル カウントが生成されます。このイベントは割り込みを生成するために使用できます。

- カウント レジスタが 0xFFFF (カウント レジスタの最大値) に達するとカウント レジスタの値はキャプチャ レジスタにコピーされ、カウント レジスタは 周期値 (この場合 0x8000) に初期化されます。

図 15-10. 直交モード -X1 符号化のタイミング図

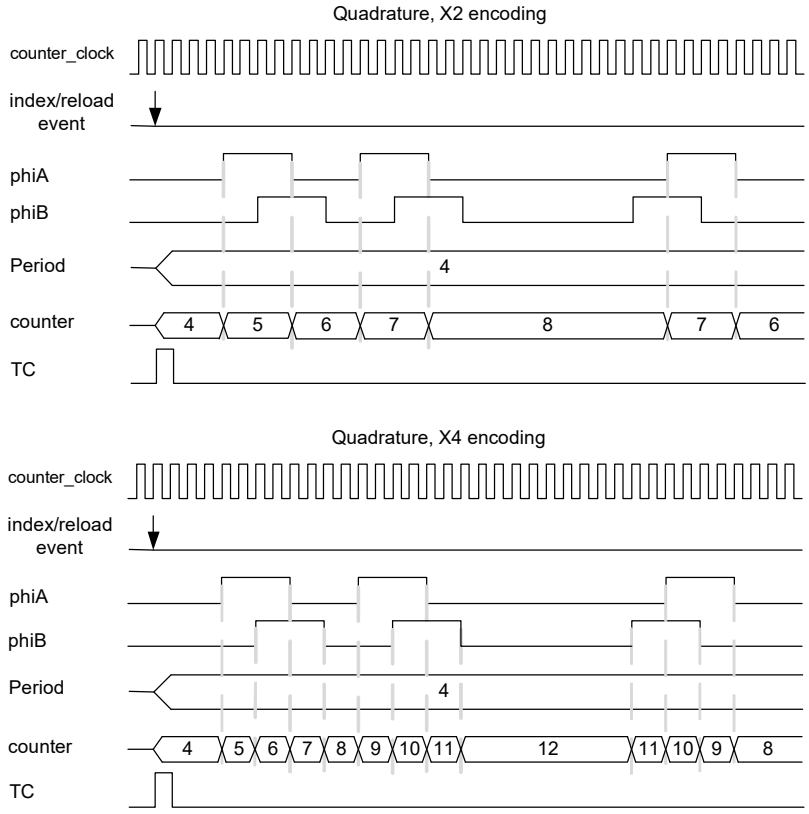


直交位相は counter_clock で検出されます。位相は 1 つの counter_clock 周期以内に 1 回以上値が変化してはいけません。

X2 と X4 直交符号化モードでは X1 符号化モードの 2 倍と 4 倍の速度でカウントします。

図 15-11 に X2 と X4 符号化モードでの直交動作を示します。

図 15-11. 直交モード -X2 と X4 符号化のタイミング図



15.3.3.3 カウンタを直交モードに設定する方法

以下にカウンターを直交動作モードに設定する手順および影響されるレジスタ ビットを示します。

1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「011」を書き込んで直交モードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期値を設定します。
4. TCPWM_CNT_CTRL レジスタの QUADRATURE_MODE[21:20] フィールドに書き込んで必要な符号化モードを設定します。
5. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (インデックス、ストップ) を発生させるトリガーを選択します。
6. TCPWM_CNT_TR_CTRL1 レジスタを設定してイベント (インデックス、ストップ) を発生させるエッジを選択します。
7. 必要に応じて 133 ページの「割り込み」に示すように、TC または CC 条件の発生時の割り込みを設定します。
8. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを有効にします。

15.3.4 パルス幅変調モード

PWM モードはデジタル コンパレータ モードとも呼ばれています。比較出力は PWM 信号です。この信号の周期は周期レジスタ値に依存し、デューティ比は比較および周期レジスタ値に依存します。

左および右揃えモードの場合 : PWM 周期 = 周期値 / カウンター クロック周波数

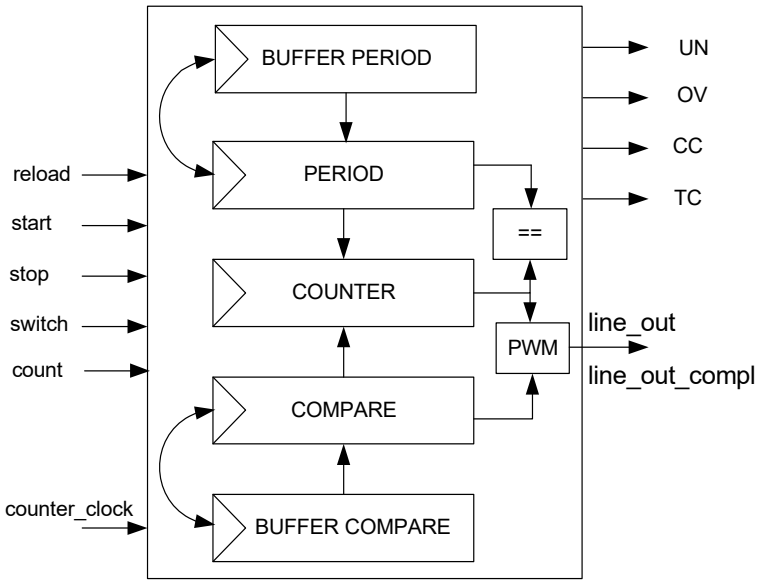
中央揃えモードの場合 : PWM 周期 = 2 × (周期値 / カウンター クロック周波数)

左および右揃えモードの場合 : デューティ比 = 比較値 / 周期値

中央揃えモードの場合 : デューティ比 = (周期値 - 比較値) ／周期値

15.3.4.1 ブロック図

図 15-12. PWM モード ブロック図



15.3.4.2 動作原理

PWM モードは左、右、中央揃えまたは非対称的な PWM 信号を出力できます。表 15-6 に示すように TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE [17:16] ビットにより選択したカウンターのカウントアップ、ダウン、アップ／ダウンのモードを使用して希望する出力アライメント設定を実現します。

この CC 信号は OV と UN 信号と共に PWM 出力ラインを制御します。TCPWM_CNT_TR_CTRL2 レジスタを設定することでこれらの信号は出力ラインを反転させるか、論理「0」か「1」に設定することができます。信号が出力ラインに関与する方法を設定することで希望するPWM出力アライメント設定を実現することができます。

デューティ比を変更する推奨方法は以下の通りです :

- バッファ周期レジスタとバッファ比較レジスタを新しい値に更新します。
- TC 条件の間アクティブな切り替えイベントが発生すると、周期レジスタと比較レジスタは自動的にバッファ周期レジスタとバッファ比較レジスタに更新されます。カウンター制御レジスタの AUTO_RELOAD_CC と AUTO_RELOAD_PERIOD フィールドは「1」にセットされます。切り替えイベントが検出されると次の TC イベントまで維持されます。パススルー信号 (イベント検出の設定中に選択) は切り替えイベントをトリガーできません。
- 図 15-14に示すようにバッファ周期レジスタとバッファ比較レジスタの更新は次の TC 条件 (アクティブな切り替えイベントを持つ) が来る前に完了する必要があります。そうしない場合切り替えはレジスタ更新を反映しません。

中央揃えモードでは、出力ラインはターミナル カウントで「0」に設定され、CC 条件でトグルされます。

リロード イベントの発生時、カウント レジスタは初期化され適切なモードでカウントを開始します。カウントの度にカウント レジスタは比較レジスタ値と比較され、一致すると CC 信号を生成します。

図 15-13 にバッファ周期レジスタとバッファ比較レジスタを搭載した中央揃え PWM (カウントアップ／ダウン モード 0) を示します。

図 15-13. 中央揃え PWM のタイミング図

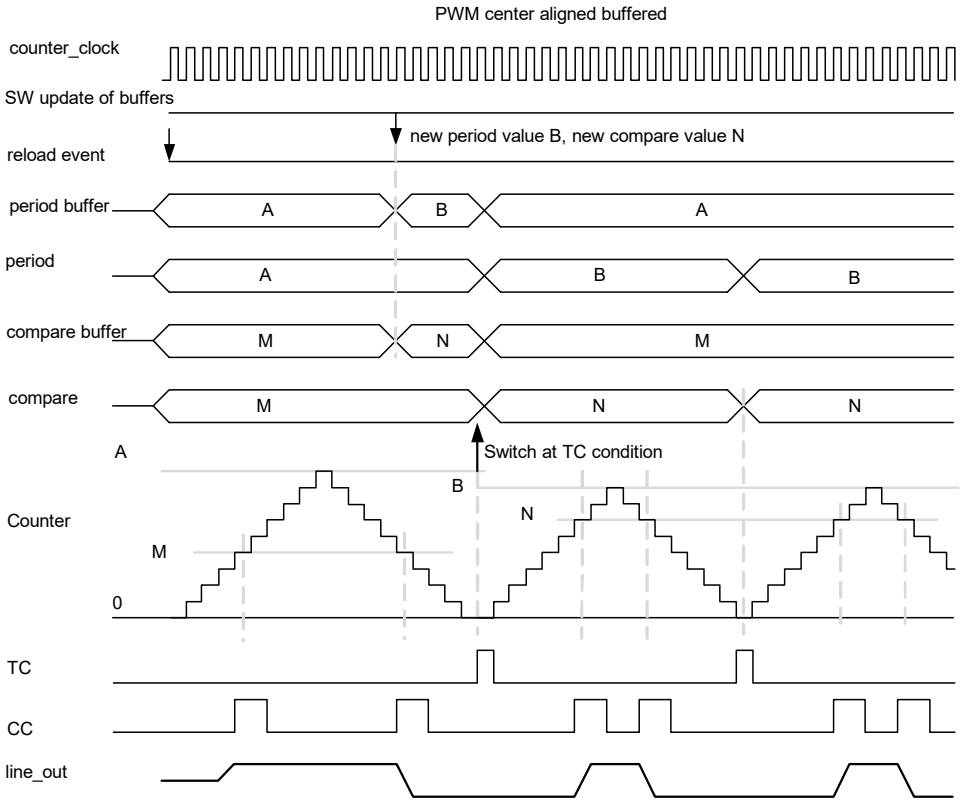
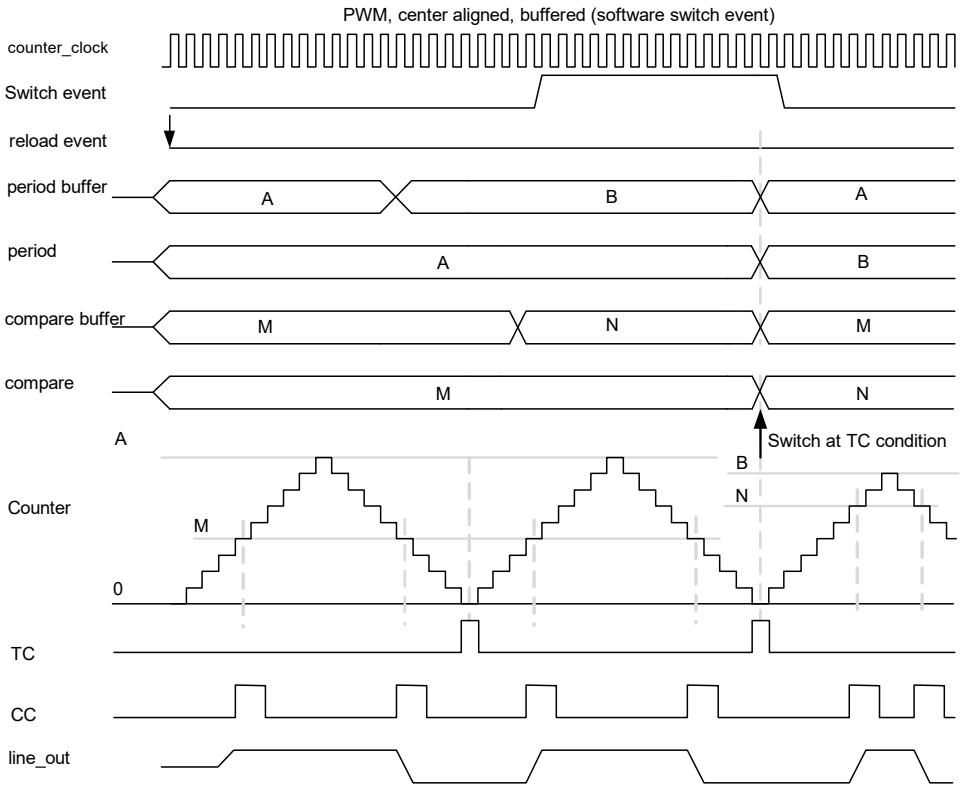


図 15-13 にソフトウェアで生成された切り替えイベントのある中央揃え PWM を示します :

- 周期バッファと比較バッファ両方のレジスタが更新された後にだけ、ソフトウェアが切り替えイベントを生成します。
- 2 番目の PWM パルスが遅れて到着する (ターミナル カウントの後) ため 1 番目の PWM パルスが繰り返されます。
- 切り替えイベントは発効後自動的にハードウェアでクリアされることにご注意ください。

図 15-14. 中央揃え PWM (ソフトウェア切り替えイベント) のタイミング図



15.3.4.3 その他のコンフィギュレーション

- 非対称的 PWM ではカウントアップ／ダウン モード 1 を使用する必要があります。これによりカウンターが「0」または周期値に達すると TC が発生します。非対称的 PWM を作成するために比較レジスタを TC 条件ごとに (カウンターが「0」または周期値に達する時) 変更し、周期レジスタを1つのTC条件おきに (カウンターが「0」に達する時にのみ) 変更します。
- 左揃え PWM ではカウントアップ モードを使用します。OV 条件を設定して出力ラインを「1」にセットし、CC 条件を設定して出力ラインを「0」にリセットします。表 15-3 を参照してください。
- 右揃え PWM ではカウントダウン モードを使用します。UN 条件を設定して出力ラインを「0」にクリアし、CC 条件を設定して出力ラインを「1」にセットします。表 15-3 を参照してください。

15.3.4.4 キル機能

キル機能により両方の出力ラインを直ちに無効にすることが可能です。表 15-7 に示すようにカウンター制御レジスタの PWM_STOP_ON_KILL と PWM_SYNC_KILL フィールドを変更することでプログラムからカウンターを停止することができます。

表 15-7. ストップ オン キル機能のフィールド設定

PWM_STOP_ON_KILL フィールド	説明
0	キル トリガーでは PWM 出力ラインを一時的にブロックするが、カウンターは動作を継続
1	キル トリガーでは PWM 出力ラインを一時的にブロックし、カウンターは停止される

表 15-8 に示すようにキル イベントは非同期または同期にプログラムできます。

表 15-8. 同期／非同期キルのフィールド設定

PWM_SYNC_KILL フィールド	説明
0	非同期キル イベントは存在する限り継続する。このイベントにはパス スルー モードが必要
1	同期キル イベントは次の TC イベントまで出力ラインを無効にする。このイベントには立ち上がりエッジ モードが必要

同期キルでは PWM は次の TC が来るまで開始できません。キル入力が解除された直後に PWM を再開するためには、キル イベントが非同期でなければなりません (表 15-8 を参照してください)。生成されたストップ イベントは両方の出力ラインを無効にします。この場合リロード イベントは同じトリガー入力信号を利用できますが、立ち下がり検出モードで使用する必要があります。

15.3.4.5 PWM モードのカウンター設定方法

以下にカウンターを PWM 動作モードに設定する手順および影響されるレジスタ ビットを示します。

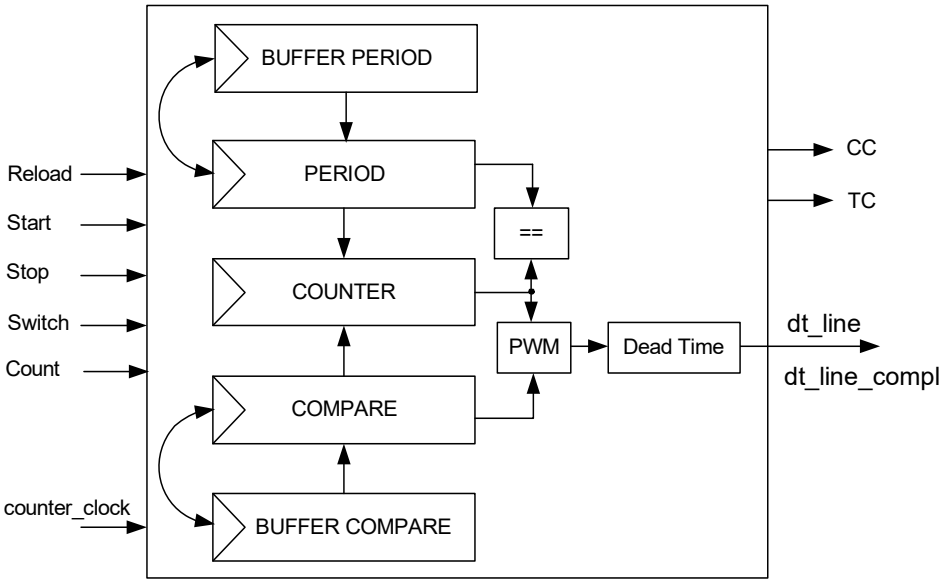
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「100」を書き込んで PWM モードを選択します。
3. 表 15-1 に示すように、TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込んでクロック分周を設定します。
4. TCPWM_CNT_PERIOD レジスタに 16 ビット周期を設定し、必要に応じて TCPWM_CNT_PERIOD_BUFF レジスタに切り替え用の周期値を設定します。
5. TCPWM_CNT_CC レジスタに 16 ビット比較値を設定し、必要に応じて TCPWM_CNT_CC_BUFF レジスタに切り替え用の比較値を設定します。
6. 表 15-6 に示すように TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE[17:16] フィールドに書き込んでカウント方向を設定し、PWM を左揃え、右揃えまたは中央揃えに設定します。
7. 必要に応じて TCPWM_CNT_CTRL レジスタの PWM_STOP_ON_KILL と PWM_SYNC_KILL フィールドを設定します。
8. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (リロード、スタート、キル、切り替え、カウント) を発生させるトリガーを選択します。
9. TCPWM_CNT_TR_CTRL1 レジスタを設定してイベント (リロード、スタート、キル、切り替え、カウント) を発生させるエッジを選択します。
10. TCPWM_CNT_TR_CTRL2 レジスタで line_out と line_out_compl レジスタを制御して CC、OV、UN 条件でセット、リセットまたは反転させることができます。
11. 必要に応じて 133 ページの「割り込み」に示すように、TC または CC 条件の発生時の割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを有効にします。ハードウェア開始信号が有効になっていない場合、カウンターを開始するためにファームウェア (TCPWM_CMD レジスタ) でスタート トリガーを与える必要があります。

15.3.5 デッドタイム付きパルス幅変調モード

デッドタイムは line_out と line_out_compl 両方の信号の遷移を遅延させるために使用されます。2 信号の遷移に指定の時間間隔を挿入します。dt_line と dt_line_compl の 2 本のコンプリメンタリ出力信号はこれらの 2 ラインから派生します。デッドバンド期間中に、比較出力とコンプリメンタリ比較出力は指定された期間論理「0」です。デッドバンド機能により 2 つの非重複 PWM パルスを生成することが可能です。この機能を使うと最大 255 クロックのデッドタイムを生成できます。

15.3.5.1 ブロック図

図 15-15. PWM-DT モード ブロック図



15.3.5.2 動作原理

デッドタイム モード付きの PWM の動作は以下の通りです :

- PWM line_out の立ち上がりエッジで UN、OV、CC 条件に応じてデッドタイム ブロックは dt_line と dt_line_compl を「0」に遷移させます。
- デッドバンド周期がロードされ、レジスタで設定された期間カウントされます。
- デッドバンド期間が終わると dt_line は「1」に遷移します。
- PWM line_out の立ち下がりエッジで UN、OV、CC 条件に応じてデッドタイム ブロックは dt_line と dt_line_compl を「0」に遷移させます。
- デッドバンド周期がロードされ、レジスタで設定された期間カウントされます。
- デッドバンド期間が終わると dt_line_compl は「1」に遷移します。
- デッドバンド期間を 0 に設定する場合、dt_line および line_out に影響を与えません。
- デッドタイム期間がパルス幅以上である場合パルスが無視されます。

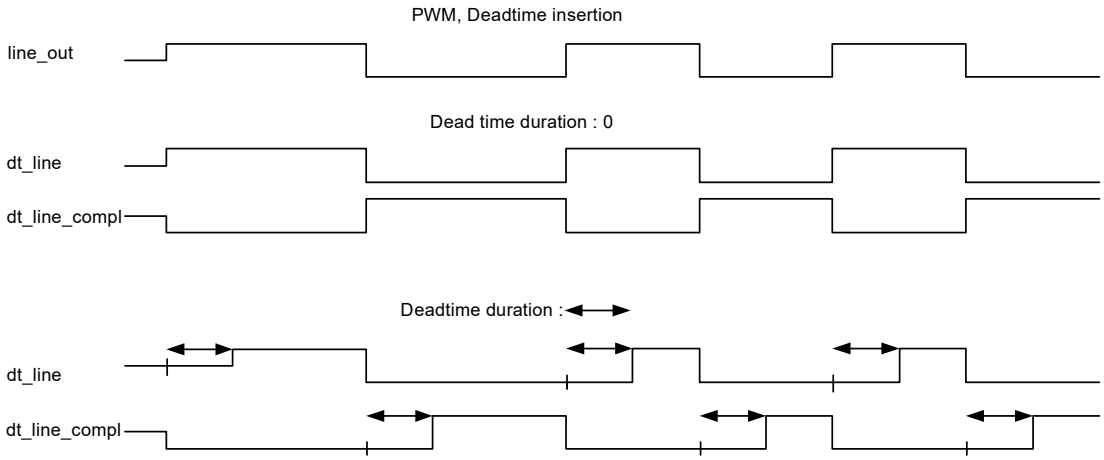
このモードは PWM モードに従い、以下の機能をサポートします :

- さまざまな出力アライメントモード
- PWM line_out と line_out_compl それぞれから派生する dt_line と dt_line_compl の 2 本のコンプリメンタリ出力ライン
  - 同期と非同期モードに対応するストップ／キル イベント
  - 比較レジスタとバッファ比較レジスタおよび周期レジスタとバッファ周期レジスタの条件付き切り替えイベント

このモードはクロック分周をサポートしません。

図 15-16 にコンプリメンタリ出力ライン dt_line と dt_line_compl がどのように PWM 出力ライン line_out から生成されるかを示します。

図 15-16. デッドタイム付き PWM のタイミング図



15.3.5.3 デッドタイム付き PWM モードのカウンター設定方法

以下にカウンターをデッドタイム付きPWM動作モードに設定する手順および影響されるレジスタ ビットを示します :

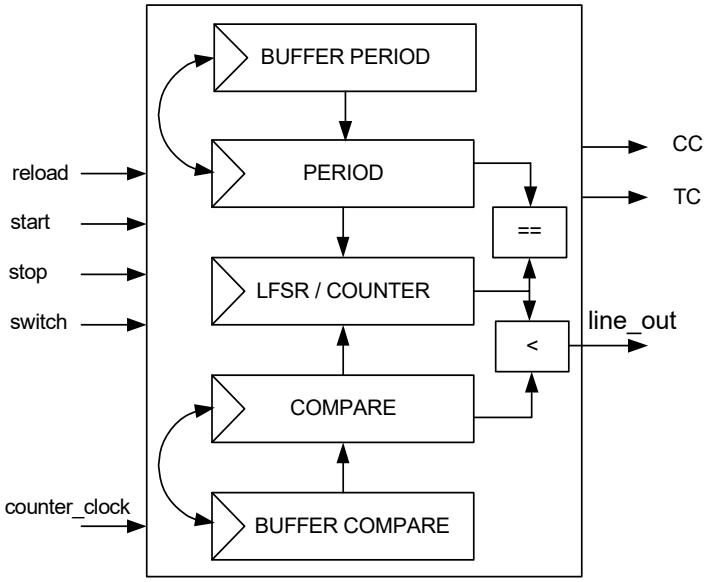
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「101」を書き込んでデッドタイム付き PWM モードを選択します。
3. 表 15-1 に示すように TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込んで必要なデッドタイムを設定します。
4. TCPWM_CNT_PERIOD レジスタに 16 ビット周期を設定し、必要に応じて TCPWM_CNT_PERIOD_BUFF レジスタに切り替え用の周期を設定します。
5. TCPWM_CNT_CC レジスタに 16 ビット比較値を設定し、必要に応じて TCPWM_CNT_CC_BUFF レジスタに切り替え用の比較値を設定します。
6. 表 15-6 に示すように TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE[17:16] フィールドに書き込んでカウント方向を設定し、PWM を左揃え、右揃えまたは中央揃えに設定します。
7. 必要に応じて 144 ページの「パルス幅変調モード」に示すように TCPWM_CNT_CTRL レジスタの PWM_STOP_ON_KILL と PWM_SYNC_KILL フィールドを設定します。
8. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (リロード、スタート、キル、切り替え、カウント) を発生させるトリガーを選択します。
9. TCPWM_CNT_TR_CTRL1 レジスタを設定してイベント (リロード、スタート、キル、切り替え、カウント) を発生させるエッジを選択します。
10. TCPWM_CNT_TR_CTRL2レジスタでdt_lineとdt_line_compl レジスタを制御して CC、OV、UN 条件でセット、リセットまたは反転させることができます。
11. 必要に応じて 133 ページの「割り込み」に示すように、TC または CC 条件の発生時の割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを有効にします。ハードウェア開始信号が有効になっていない場合カウンターを開始するためにファームウェア (TCPWM_CMDレジスタ) でスタート トリガーを与える必要があります。

15.3.6 疑似乱数パルス幅変調モード

このモードは線形帰還シフトレジスタ (LFSR) を使用します。LFSR は入力ビットが前の状態の線形関数であるシフトレジスタです。

15.3.6.1 ブロック図

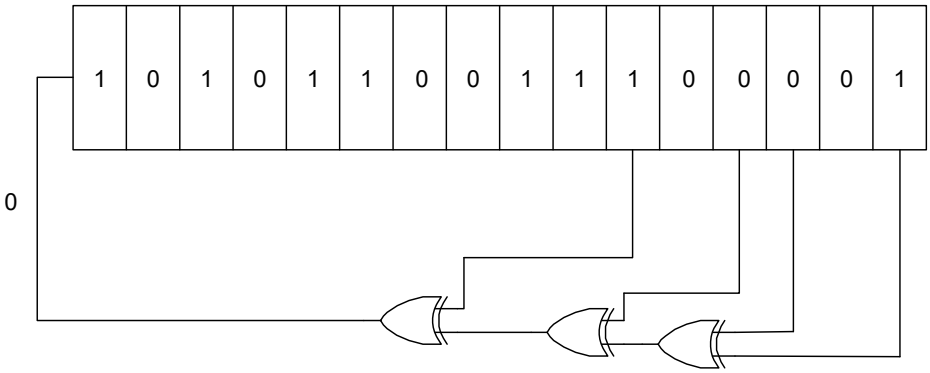
図 15-17. PWM-PR モード ブロック図



15.3.6.2 動作原理

図 15-18 に示すようにカウンター レジスタは次の多項式で LFSR を搭載するために使用されます : $x^{16}+x^{14}+x^{13}+x^{11}+1$。1 ～ 0xFFFF 範囲内のすべての値を疑似乱数系列で生成します。カウンター レジスタを 0 以外の値に初期化する必要があることにご注意ください。

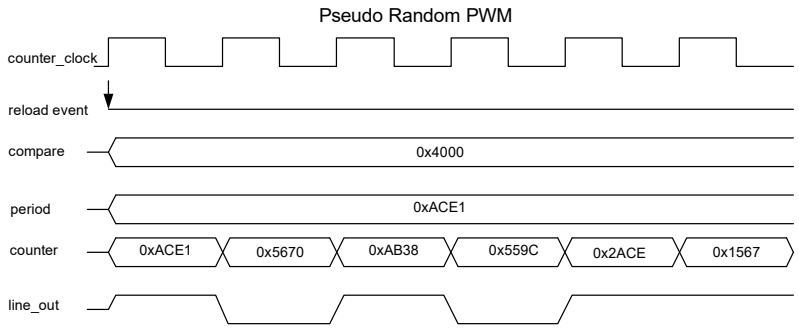
図 15-18. カウンター レジスタを用いた疑似乱数系列の生成



このプロセスの手順は以下の通りです :

- カウンター レジスタの下位 15 ビット値が比較レジスタの値より小さくなる (カウンター[14:0] < 比較 [15:0]) と PWM 出力ライン line_out が「 1」に駆動されます。「0x8000」以上の比較値では常に PWM 出力ラインが「1」になります。「0」の比較値では常に PWM 出力ラインが「0」になります。
- リロード イベントはスタート イベントと同様に機能しますが、カウンターを初期化しません。
- カウントが周期値に等しくなるとターミナル カウントが生成されます。特定の初期値の場合 LFSR は予測可能なカウントのパターンを生成します。この予測可能性により特定の LFSR の反復回数「n」後にカウントを計算できます。計算したカウントは周期値として使用され、TC は「n」の反復回数後に生成されます。
- TC の発生時に切り替え／キャプチャ イベントは条件付きで (カウンター制御レジスタの AUTO_RELOAD_CC と AUTO_RELOAD_PERIOD フィールドに基づいて) 比較と周期レジスタペアを切り替えます。
- 前述したようにキル イベントをプログラムしてカウンターを停止することができます。
- カウンター制御レジスタの ONE_SHOT フィールドを設定することでワンショット モードを設定します。ターミナル カウントでカウンターはハードウェアにより停止されます。
- このモードではアンダーフロー、オーバーフロー、トリガー条件イベントは発生しません。
- カウンターが実行中で、カウントが比較値と等しくなると CC 条件が発生します。図 15-19 に疑似乱数ノイズ動作を示します。
- 比較値 0x4000 はデューティ比 50% となります (16 ビット カウンターの下位 1 ビットだけが比較レジスタ値と比較するために使用されます)。

図 15-19. 疑似乱数 PWM のタイミング図



キャプチャ／切り替え入力信号で比較レジスタと比較バッファレジスタおよび周期レジスタと周期バッファ レジスタの値を切り替えることがあります。トリガー入力信号で変調を制御することでこの機能は 2 つの異なる比較値を変調するために使用できます。

注 : キャプチャ／切り替え入力信号はエッジ (立ち上がり、立ち下がり、両方) でのみトリガーできます。この入力信号は次のターミナル カウントまで維持されます。

15.3.6.3 疑似乱数 PWM モードのカウンター設定方法

以下にカウンターを疑似乱数 PWM 動作モードに設定する手順および影響されるレジスタ ビットを示します。

1. TCPWM_CTRL レジスタの COUNTER_ENABLED に「0」の書き込みによりカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「110」を書き込んで疑似乱数 PWM モードを選択します。
3. TCPWM_CNT_PERIOD レジスタに 16 ビット周期を設定し、必要に応じて TCPWM_CNT_PERIOD_BUFF レジスタに切り替え用の周期を設定します。
4. TCPWM_CNT_CC レジスタに 16 ビット比較値を設定し、TCPWM_CNT_CC_BUFF レジスタに切り替え用の比較値を設定します。
5. 必要に応じて TCPWM_CNT_CTRL レジスタの PWM_STOP_ON_KILL と PWM_SYNC_KILL フィールドを設定します。
6. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (リロード、スタート、キル、切り替え) を発生させるトリガーを選択します。
7. TCPWM_CNT_TR_CTRL1 レジスタを設定してイベント (リロード、スタート、キル、切り替え) を発生させるエッジを選択します。
8. TCPWM_CNT_TR_CTRL2 レジスタで line_out と line_out_compl レジスタを制御して CC、OV、UN 条件でセット、リセットまたは反転させることができます。

9. 必要に応じて 133 ページの「割り込み」に示すように、TC または CC 条件の発生時の割り込みを設定します。
10. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」の書き込みによりカウンターを有効にします。

15.4 TCPWM レジスタ

表 15-9. TCPWM レジスタの一覧

レジスタ	説明	機能
TCPWM_CTRL	TCPWM 制御レジスタ	カウンター ブロックを有効にする
TCPWM_CMD	TCPWM コマンド レジスタ	ソフトウェア イベントを生成
TCPWM_INTR_CAUSE	TCPWM カウンター割り込み原因レジスタ	集約された割り込み信号のソースを判定
TCPWM_CNT_CTRL	カウンター制御レジスタ	カウンター モード、符号化モード、ワンショット モード、スイッチング、キル機能、デッド タイム、クロック分周、カウント方向を設定
TCPWM_CNT_STATUS	カウンター ステータス レジスタ	カウントの方向、デッド タイム期間、クロック分周を読み出し、カウンターが動作中であるかをチェック
TCPWM_CNT_COUNTER	カウント レジスタ	16 ビット カウンター値を格納
TCPWM_CNT_CC	カウンター比較／キャプチャ レジスタ	カウントを取り込むまたはカウンター値と比較
TCPWM_CNT_CC_BUFF	カウンター バッファ比較／キャプチャ レジスタ	カウンター CC レジスタのバッファ レジスタであり、周期を切り替える
TCPWM_CNT_PERIOD	カウンター周期レジスタ	カウンターの上限値を格納
TCPWM_CNT_PERIOD_BUFF	カウンター バッファ周期レジスタ	カウンター周期レジスタのバッファ レジスタであり、比較値を切り替える
TCPWM_CNT_TR_CTRL0	カウンター トリガー制御レジスタ 0	特定のカウンター イベントのトリガーを選択
TCPWM_CNT_TR_CTRL1	カウンター トリガー制御レジスタ 1	特定のカウンター入力信号のエッジ検出
TCPWM_CNT_TR_CTRL2	カウンター トリガー制御レジスタ 2	CC、OV、UN 条件の発生時にカウンター出力ラインを制御
TCPWM_CNT_INTR	割り込み要求レジスタ	TC または CC 条件が検出されるとレジスタ ビットをセット
TCPWM_CNT_INTR_SET	割り込み要求セット レジスタ	割り込み要求レジスタの対応するビットをセット
TCPWM_CNT_INTR_MASK	割り込みマスク レジスタ	割り込み要求レジスタのマスク
TCPWM_CNT_INTR_MASKED	マスクされた割り込み要求レジスタ	割り込み要求とマスク レジスタのビット論理積

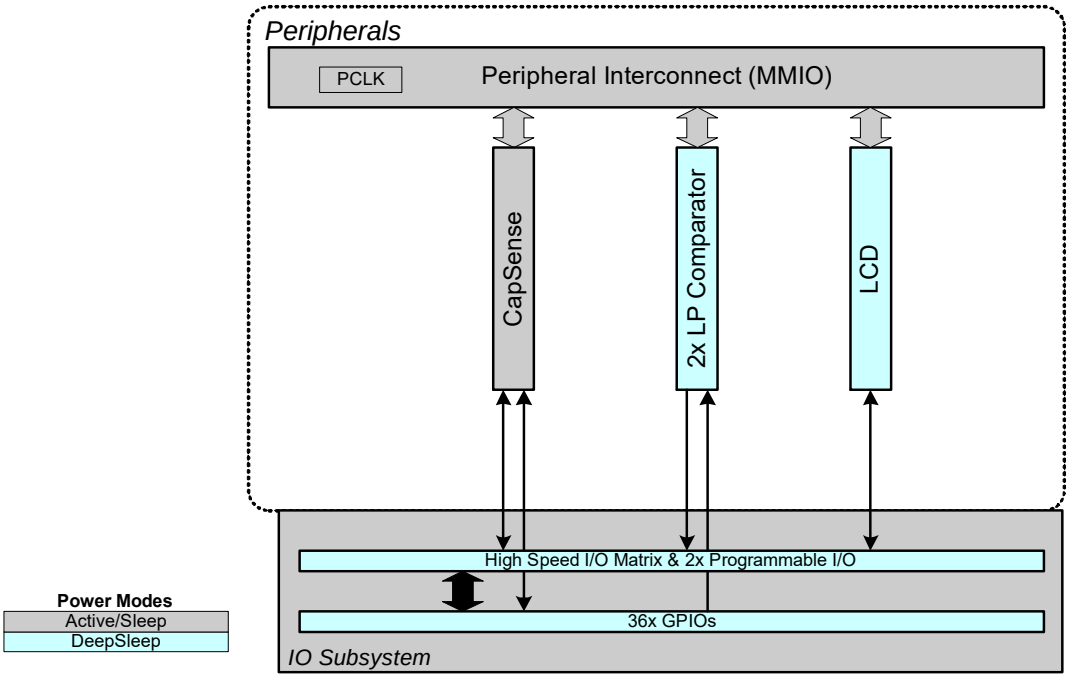
セクション E: アナログ システム

このセクションは次の章を含みます。

- 154 ページの低消費電力コンパレータの章
- 161 ページの LCD ダイレクト ドライブの章
- 160 ページの CapSense の章

トップ レベル アーキテクチャ

アナログ システム ブロック図



16. 低消費電力コンパレータ

PSoC® 4 デバイスは 2 個の低消費電力コンパレータを備えています。これらのコンパレーターはすべての消費電力モードで高速アナログ信号の比較を可能にします。デバイスの消費電力モードの詳細については 69 ページの消費電力モードを参照してください。正と負の入力は専用の GPIO 端子または AMUXBUS-A/AMUXBUS-B に接続することができます。コンパレーター出力は、ステータス レジスタを介して CPU に読み込みができ、割り込みまたは復帰ソースとして使用、または GPIO に配線されます。

16.1 特長

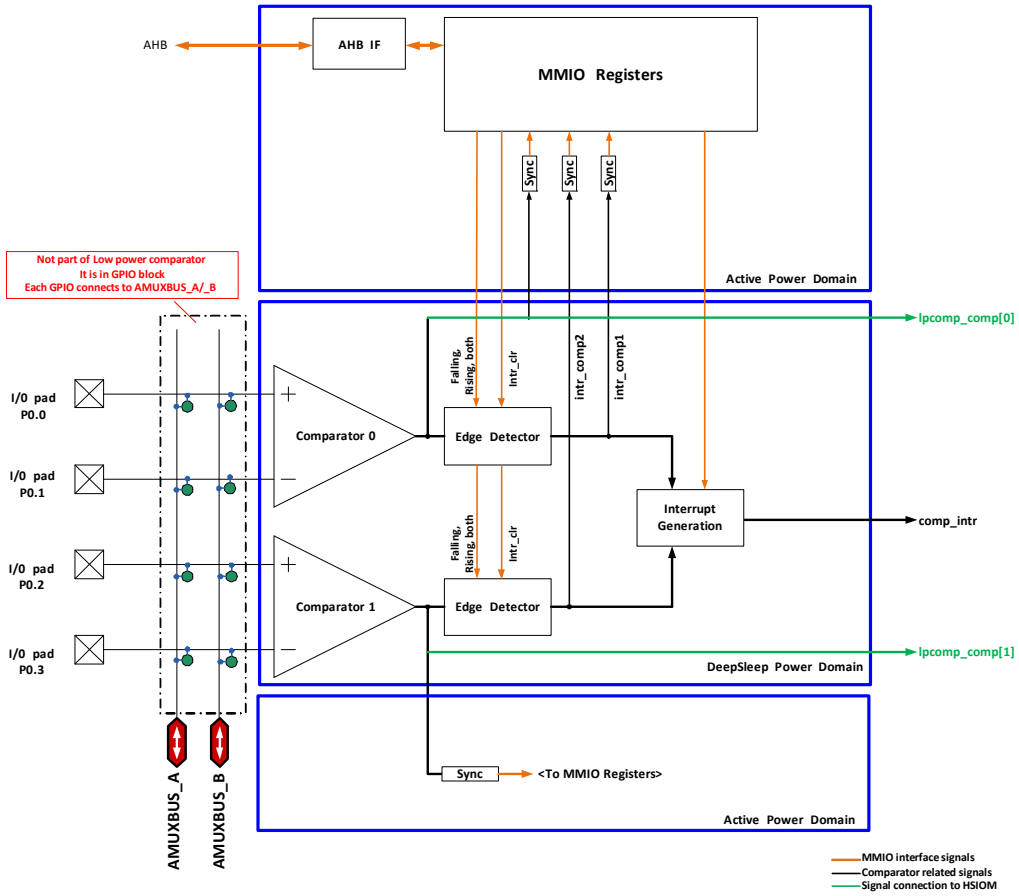
PSoC 4 コンパレータは次の機能があります :

- コンフィギュレーション可能な正と負の入力
- プログラム可能な消費電力および動作速度
- 超低消費電力モードに対応 (4µA 未満)
- 10mV の入力ヒステリシス オプション
- 小さな入力オフセット電圧 (調整後 4mV 未満)
- ディープスリープ モードでの復帰ソース

16.2 ブロック図

図 16-1 に低消費電力コンパレーターのブロック図を示します。

図 16-1. 低消費電力コンパレータ ブロック図



16.3 動作原理

以下の節に、PSoC 4 の低消費電力コンパレータの動作、入力コンフィギュレーション、電力と動作速度モード、出力と割り込みコンフィギュレーション、ヒステリシス、低消費電力モードから復帰、コンパレータ クロックおよびオフセット調整を説明します。

16.3.1 入力コンフィギュレーション

コンパレータへの入力は以下に示す通りです :

- 専用入力端子から 2 つの正と負の入力
- AMUXBUS を介し、任意の端子から 2 つの正と負の入力 (ディープスリープ モードでは使用不可)
- 外部端子からの入力と、内部で生成された信号を利用する入力。外部信号と内部信号は、コンパレーターの正と負の入力のどちらにも接続することができる。内部で生成された信号はアナログ AMUXBUS を介して、コンパレーターの入力に接続される
- 内部で生成された信号を正と負に入力。内部で生成された信号は AMUXBUS-A/AMUXBUS-B を介して、コンパレーターの入力に接続される

図 16-1 で、P0.0 と P0.1 はコンパレーター 0 の正と負の入力に接続され、P0.2 と P0.3 はコンパレーター 1 の入力に接続されることにご注意ください。AMUXBUS ネットはコンパレーターの入力に直接に接続されないことにもご注意ください。このようにしてコンパレーターの接続は、対応する入力端子を介して AMUXBUS ネットに配線されます。コンパレーターの接続に AMUXBUS を使用する場合、これらの入力端子はその他の目的に使用することができません。コンパレーター入力の接続に AMUXBUS を使用する設計では、それらをオープンの状態にすべきです。AMUXBUS 接続はディープスリープ モードで使用できないことにご注意ください。もし、ディープスリープ動作で必要な場合、低消費電力コンパレータは専用端子で接続する必要があります。この制限事項は、接続に AMUXBUS を使用し、内部で生成されたいずれの信号配線も含みます。GPIO の AMUXBUS A/B への接続、またはコンパレーター入力の GPIO の設定の詳細については 37 ページの I/O システムを参照してください。

16.3.2 出力および割り込みコンフィギュレーション

コンパレーター0 とコンパレーター1 の出力は、OUT1 ビット [6] と OUT2 ビット [14] で、LPCOMP_CONFIG レジスタでそれぞれ使用可能です (表 16-1)。コンパレーター出力は、LPCOMP_CONFIG レジスタの OUTx ビットにラッチする前に、SYSCLK に同期されます。各コンパレーターの出力は対応するエッジの検出ブロックに接続されます。このブロックは割り込みをトリガーするエッジを決めます。エッジの選択および割り込みの有効化は、LPCOMP_CONFIG レジスタの INTTYPE1 ビット [5:4] と INTTYPE2 ビット [13:12] を使用して行います。INTTYPEx ビットでは、割り込みのタイプを表 16-1. に示すように、無効、立ち上がりエッジ、立ち下がりエッジまたはその両エッジから選択します。

各コンパレータの出力は HSIOM を介して、直接に GPIO 端子に配線されます。コンパレータ出力は HSIOM でディープスリープ ソース 2 接続として使用できます。HSIOM の詳細は 42 ページの高速 I/O マトリックスを参照してください。低消費電力コンパレータ出力をサポートする端子の詳細はデバイス データシートを参照してください。これらの端子での出力はコンパレータから直接に出力され、同期化されません。これらは端子でディープスリープ ソースとして動作するためであり、コンパレータ出力をディープスリープ電力モードで利用するためです。

エッジ イベント中、コンパレーターは割り込みをトリガーします (図 16-1 の intr_comp1/intr_comp2 信号)。割り込み要求は、コンパレーター 0 とコンパレーター 1 それぞれに、LPCOMP_INTR レジスタの COMP1 ビット [0] と COMP2 ビット [1] に保存されます。コンパレーター 0 とコンパレーター1 の両方は割り込みを共用します (図 16-1 の comp_intr 信号)。それは 2 つの割り込みの論理 OR であり、CPU NVIC の低消費電力コンパレーター ブロックの割り込みとしてマップされます。詳細は 27 ページの割り込みを参照してください。両方のコンパレーターが使用される場合、LPCOMP_INTR レジスタの COMP1 と COMP2 ビットを、どちらが割り込みをトリガーしたかが分かるように、割り込みサービス ルーチンで読み出す必要があります。もしくは LPCOMP_INTR_MASK レジスタの COMP1_MASK ビット [0] と COMP2_MASK ビット [1] を、CPU へのコンパレーター 0 とコンパレーター 1 の割り込みマスクに使用することができます。マスクされた割り込みのみが CPU により処理されます。割り込みが処理された後は、ファームウェア内の LPCOMP_INTR レジスタの COMP1 と COMP2 ビットに「1」を書き込みクリアします。割り込みがクリアされないと次の比較イベントは割り込みをトリガーせず、CPU はイベントを処理することができなくなります。

LPCOMP 割り込み (comp1_intr/comp2_intr) は SYSCLK に同期されます。comp1_intr/comp2_intr のクリアはすべて同期されます。

LPCOMP_INTR_SET レジスタのビット [1:0] は、ソフトウェア デバッグ処理のために、割り込みのアサートに使用することができます。

ディープスリープ モードでは、復帰割り込みコントローラー (WIC) は、CPU を復帰させるコンパレーターのエッジイベントにより有効化されることがあります。従って、LPCOMP は低消費電力モードで指定の信号をモニターすることが可能です。

表 16-1. LPCOMP_CONFIG レジスタの出力と割り込みコンフィギュレーション

レジスタ [ビット位置]	ビット名	説明
LPCOMP_CONFIG[6]	OUT1	コンパレーター 0 の現在の出力値
LPCOMP_CONFIG[14]	OUT2	コンパレーター 1 の現在の出力値
LPCOMP_CONFIG[5:4]	INTTYPE1	コンパレーター 0 が IRQ をトリガーするエッジを設定 00: 無効 01: 立ち上がりエッジ 10: 立ち下がりエッジ 11: 立ち上がりと立下りエッジ
LPCOMP_CONFIG[13:12]	INTTYPE2	コンパレーター 1 が IRQ をトリガーするエッジを設定 00: 無効 01: 立ち上がりエッジ 10: 立ち下がりエッジ 11: 立ち上がりと立下りエッジ
LPCOMP_INTR[0]	COMP1	コンパレーター 0 の割り込み : コンパレーター 0 がトリガーする時にハードウェアはこの割り込みをセット。「1」を書き込んで、割り込みをクリア
LPCOMP_INTR[1]	COMP2	コンパレーター 2 の割り込み : コンパレーター 1 がトリガーする時にハードウェアはこの割り込みをセット。「1」を書き込んで、割り込みをクリア
LPCOMP_INTR_SET[0]	COMP1	「1」の書き込みにより、コンパレーター 0 のソフトウェア割り込みをトリガー
LPCOMP_INTR_SET[1]	COMP2	「1」の書き込みにより、コンパレーター 1 のソフトウェア割り込みをトリガー

16.3.3 消費電力モードおよび動作速度のコンフィギュレーション

低消費電力コンパレーターは次の 3 つのモードで動作可能です

- 高速
- 低速
- 超低消費電力

コンパレーター 0 の電力または速度は、LPCOMP_CONFIG レジスタの MODE1 ビット [1:0] により設定することが可能です。コンパレーター 1 の電力または速度は、同じレジスタの MODE2 ビット [9:8] により設定することが可能です。電力の消費量および応答時間は選択する消費電力モードによって異なります。電力の消費量は、高速のモードで最も多く、超低消費電力のモードで最も少なくなります。応答時間は、高速のモードで最も短く、超低消費電力のモードで最も長くなります。種々の電力設定による応答時間と電力消費量の詳細については、デバイス データシートを参照してください。

表 16-2 に示すように、コンパレーターは LPCOMP_CONFIG レジスタの ENABLE1 ビット [7] と ENABLE2 ビット [15] を使用して有効／無効にします。

注 コンパレーターが有効にされた時、消費電力モードを変更すると、コンパレーターの出力にグリッチが出ることがあります。これを防止するため、消費電力モードを変更する前に、コンパレーターを無効にします。

表 16-2. コンパレーター動作モードの選択ビット

レジスタ [ビット位置]	ビット名	説明
LPCOMP_CONFIG[1:0]	MODE1	コンパレーター 0 の消費電力モード選択 00: 低速動作モード (消費電力小) 01: 高速動作モード (消費電力大) 10: 超低消費電力動作モード (消費電力最小)
LPCOMP_CONFIG[9:8]	MODE2	コンパレーター 1 の消費電力モード選択 00: 低速動作モード (消費電力小) 01: 高速動作モード (消費電力大) 10: 超低消費電力動作モード (消費電力最小)
LPCOMP_CONFIG[7]	ENABLE1	コンパレーター 0 の有効ビット 0: コンパレーター 0 が無効 1: コンパレーター 0 が有効
LPCOMP_CONFIG[15]	ENABLE2	コンパレーター 1 の有効ビット 0: コンパレーター 1 が無効 1: コンパレーター 1 が有効

16.3.4 ヒステリシス

電位差のない信号と変化の遅い信号を比較するアプリケーションで、ヒステリシスは信号にノイズがある時にコンパレーター出力の振動の回避に役立ちます。そのようなアプリケーションには、コンパレーター ブロックで 10mV のヒステリシスを適用することがあります。

表 16-3 で説明したように、10mV のヒステリシス レベルは、LPCOMP_CONFIG レジスタの HYST1 ビット [2] と HYST2 ビット [10] を使用して適用します。

表 16-3. ヒステリシス制御ビット HYST1 と HYST2

レジスタ [ビット位置]	ビット名	説明
LPCOMP_CONFIG[2]	HYST1	コンパレーター 0 に 10mV のヒステリシスを適用する - 0: ヒステリシスを適用 - 1: ヒステリシスなし
LPCOMP_CONFIG[10]	HYST2	コンパレーター 1 に 10mV のヒステリシスを適用する - 0: ヒステリシスを適用 - 1: ヒステリシスなし

16.3.5 低消費電力モードからの復帰

コンパレータはスリープ およびディープスリープ モードを含むデバイスの低消費電力モードで動作します。コンパレーター出力の割り込みはスリープ およびディープスリープ モードからデバイスを復帰させることができます。低消費電力モードからデバイスを復帰させるコンパレーターについて、LPCOMP_CONFIG レジスタで有効にし、INTTYPEx ビットを無効以外に設定し、INTR_MASKx ビットを LPCOMP_INTR_MASKレジスタ1に設定します。AMUXBUS 接続が関係する比較はディープスリープ モードで使用できません。

ディープスリープ電力モードでは、コンパレーター 0 またはコンパレーター 1 の比較イベントは復帰割り込みを生成します。LPCOMP_CONFIG レジスタの INTTYPEx ビットを、必要に応じて、省電力モードからデバイスを復帰させコンパレーターについて設定します。LPCOMP_INTR_MASK レジスタのマスク ビットは、CPU により処理されるように、1つかそれとも2つのコンパレーターの割り込み選択に使用されます。

16.3.6 コンパレーター クロック

コンパレーターはシステムの主クロックである SYSCLK を割り込み同期用のクロックに使用します。

16.3.7 オフセット調整

コンパレーター オフセットは工場出荷時に 4.0mV 以内に調整されます。この調整は 2 段階で行われ、最初にコモン モード電圧は0.1Vに調整され、その後コモン モード電圧はV_{DD}–0.1V に調整されます。オフセット電圧は 0.1V から V_{DD}–0.1 V の入力電圧範囲で 10.0mV 未満であることが保証されます。通常の動作では更にトリム値の調整はお勧めしません。

特定の入力同相モード電圧でより高精度な調整が必要とされる場合、その入力同相モード電圧で調整を行ってください。コンパレーター オフセット調整は、LPCOMP_TRIM1/2/3/4 のレジスタを使用して行います。コンパレーター 0 の調整にLPCOMP_TRIM1 とLPCOMP_TRIM2 を使用します。コンパレーター 1 の調整に LPCOMP_TRIM3 と LPCOMP_TRIM4 を使用します。調整値を変更するビット フィールドは、LPCOMP_TRIM1 と LPCOMP_TRIM3 の TRIMA ビット [4:0] およびLPCOMP_TRIM2 とLPCOMP_TRIM4 の TRIMB ビット [3:0] です。TRIMA ビットはオフセットの粗調整に、TRIMB ビットは微調整に使用します。TRIMB ビットが使用できるのはコンパレータ動作の低速モードの場合だけです。

任意の標準的なコンパレーター オフセット調整手順で調整を実行します。特定のリファレンス／同相モード電圧入力でオフセットを改善するために、以下の方法を適用することが可能です。

1. コンパレーター入力を外部で短絡し、V_{ref} の基準電圧を入力に接続します。
2. コンパレーターを設定し、ヒステリシスをオフにし、出力を確認します。
3. 出力が High の場合、オフセットは正になります。出力が High でない場合、オフセットは負になります。以下の手順に従ってオフセットを調整してください :
   a. TRIMA ビット [4:0] を出力が反転するまで調整します。TRIMA ビット [3:0] はオフセット量を、TRIMA ビット [4] はオフセットの極性を制御します (「1」は正オフセットに、「0」は負オフセットを示します)。
   b. TRIMA ビット調整の完了後、引き続き、TRIMB ビット [3:0] を出力が反転するまで調整します。TRIMB ビットの調整はコンパレーター動作が低速モードの場合にのみ有効です。TRIMB ビット [3] はオフセットの極性を制御します。TRIMB ビット [2:0] はオフセット量です。
   c. 3-b のステップが完了した後、TRIMA と TRIMB ビットの値はその特定の V_{ref} での最良の調整値になります。

16.4 レジスタ要約

表 16-4. 低消費電力コンパレータ レジスタ要約

レジスタ	機能
LPCOMP_ID	LPCOMP コントローラー ID とリビジョン番号を保持
LPCOMP_CONFIG	LPCOMP コンフィギュレーション レジスタ
LPCOMP_INTR	LPCOMP 割り込みレジスタ
LPCOMP_INTR_SET	LPCOMP 割り込みセット レジスタ
LPCOMP_INTR_MASK	LPCOMP 割り込み要求マスク レジスタ
LPCOMP_INTR_MASKED	LPCOMP マスクされた割り込み出力レジスタ
LPCOMP_TRIM1	コンパレーター 0 の調整領域
LPCOMP_TRIM2	コンパレーター 0 の調整領域
LPCOMP_TRIM3	コンパレーター 1 の調整領域
LPCOMP_TRIM4	コンパレーター 1 の調整領域

17. CapSense

CapSense システムは、電極の自己容量または一対の電極間の相互容量を測定できます。静電容量センシングに加えて、CapSense システムは ADC として機能し、CapSense 機能をサポートする GPIO ピンの電圧を測定できます。

自己静電容量を検出する PSoC 4 MCU の CapSense タッチセンシング方式は、CapSense シグマデルタ (CSD) として知られています。同様に、相互容量センシング方式は CapSense クロスポイント (CSX) として知られています。CSD と CSX タッチ センシング方式は、産業界最高の信号対ノイズ比 (SNR)、高いタッチ感度、低消費電力動作、および優れた EMI 性能を実現しています。

CapSense タッチセンシングは、ハードウェアとファームウェアの技術の組み合わせです。したがって、PSoC Creator IDE が提供する CapSense コンポーネントを使用して、CapSense デザインを実装してください。詳細については、PSoC 4 および PSoC 6 MCU CapSense 設計ガイドを参照してください。

18. LCD ダイレクト ドライブ

PSoC® 4 液晶ディスプレイ (LCD) ドライブ システムは、PSoC が STN および TN 型セグメント LCD を直接駆動できるようにする、高度にコンフィギュレーション可能なペリフェラルです。

18.1 特長

PSoC 4 LCD セグメント ドライブ ブロックは以下の特長を持っています :

- 最大 28 セグメントおよび 8 コモンに対応
- タイプ A (標準) およびタイプ B (低消費電力) の駆動波形に対応
- コモンとセグメントを任意の GPIO ピンに配線可能
- 5 つの駆動方法に対応 :
  - デジタル相関
  - 1/2 バイアス PWM
  - 1/3 バイアス PWM
  - 1/4 バイアス PWM
  - 1/5 バイアス PWM
- デジタル相関 モードでは 1.8V の V_{DD} で 3V ディスプレイを駆動可能
- アクティブ、スリープおよびディープスリープ モードで動作可能
- デジタル コントラスト制御

18.2 LCD セグメント ドライブの概要

セグメント LCD パネルは 1 対の電極の間に液晶材料があり、多様な偏光板と反射板を持ちます。1 対の電極は、1 つをコモン (COM) またはバックプレーン電極と呼び、もう 1 つをセグメント電極 (SEG) と呼びます。電気的に LCD セグメントは容量性負荷と見なされます。COM/SEG 電極はセグメントの行列要素と見なされます。LCD セグメントの遮光性は、対応する COM/SEG の組にかかる実効値電圧 (RMS) を変更することによって制御されます。

LCD の駆動方法を説明するために、この章では以下の用語、電圧を使用します :

- **V_{RMSOFF}**: セグメントがオフと見なされる LCD ドライブ電圧
- **V_{RMSON}**: セグメントがオンと見なされる LCD ドライブ電圧
- **識別比率 (D)**: V_{RMSON} と V_{RMSOFF} の比率。これは LCD パネルに適用される波形タイプに依存する。識別比率が高いとコントラストが高くなります。

液晶材料は長時間の DC 電圧に耐えられません。そのためパネルに加えられる波形は、すべてのセグメント (オンまたはオフ) について DC 成分をなくさなければなりません。一般的に LCD ドライバーは、複数電圧を切り替えることにより発生させる波形を COM および SEG 電極に加えます。これらの波形を定義するため、以下の用語を使用します :

- **デューティ**: あるドライバーがCOM電極をM個駆動するとき、このドライバーは1/Mデューティで動作すると言われます。各 COM 電極は実効的に 1/M の時間ドライブされます。
- **バイアス** : ドライブ波形が (1/B) × VDRV の電圧ステップを使用するとき、1/B バイアスを使用すると言われます。VDRV はシステムで最も高い駆動電圧 (PSoC 4 での V_{DD} と同じ) です。PSoC 4 は PWM 駆動モードで 1/2、1/3、1/4 および 1/5 バイアスに対応します。
- **フレーム** : フレームはすべてのセグメントを駆動するために必要な時間の長さです。フレームの間に、ドライバーは順序に従ってコモンを通じて信号を変化させます。フレーム全体を測定するとき、すべてのセグメントで 0V DC (RMS 電圧は 0 ではない) となります。

PSoC 4 は全ての駆動モードで 2 種類の駆動波形をサポートします。それらは以下の通りです :

- **タイプ A 波形**: ドライバーは M サブフレームにフレームを構成します。M は COM 電極の数です。各 COM はフレームの間 1 回のみアドレスされます。例えば COM[i] は i サブフレームでアドレスされます。
- **タイプ B 波形**: ドライバーは 2M サブフレームにフレームを構成します。2 つのサブフレームはお互いの逆のものです。各 COM はフレームの間 2 回アドレスされます。例えば COM[i] は i サブフレームと M+i フレームでアドレスされます。タイプ B 波形はフレーム内の遷移が少ないため、電力効率が若干高まります。

18.2.1 駆動モード

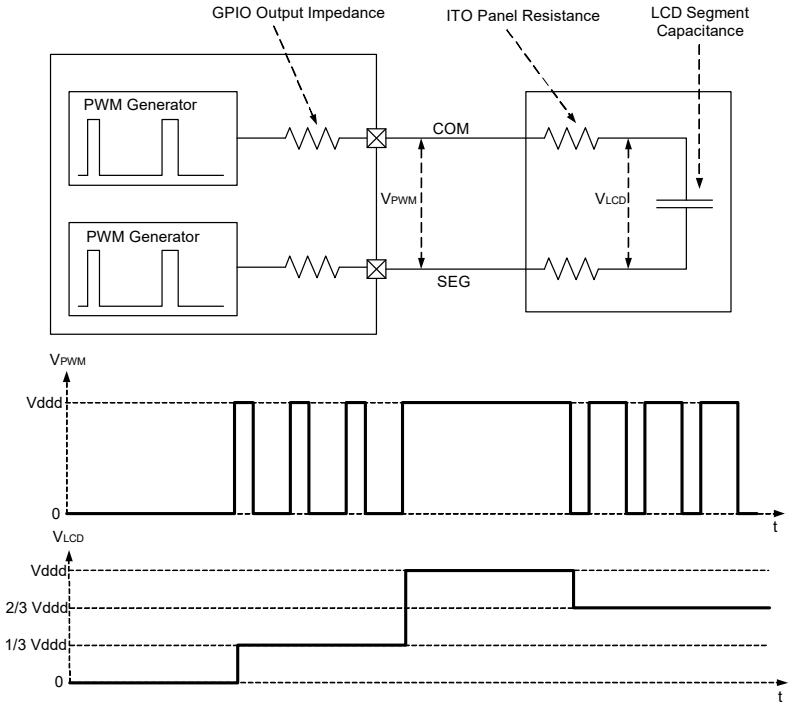
PSoC 4 は以下の駆動モードに対応します。

- 1/2 バイアス PWM
- 1/3 バイアス PWM
- 高周波数クロック入力と 1/4 バイアス PWM
- 高周波数クロック入力と 1/5 バイアス PWM
- デジタル相関

18.2.1.1 PWM ドライブ

PWM 駆動モードでは、複数電圧ドライブ信号を LCD の固有抵抗と静電容量を組み合わせて PWM 出力信号を使用することで生成します。図 18-1 はこれを説明する図です。

図 18-1. PWM ドライブ (1/3 バイアス)



ドライブ回路の出力波形は PWM 波形です。PWM 波形を平滑化するために、酸化インジウムスズ (ITO) パネル抵抗とセグメント静電容量を利用することによって、LCD セグメント間電圧は図 18-1 に示すようなアナログ電圧になります。この図は 1/3 バイアス波形 (4 つのコモンと $V_{DD}/3$ の電圧ステップ) の生成を説明しています。

PWM は ILO (32kHz、低速動作) または IMO (高速動作) から生成されます。生成されたアナログ電圧はセグメント LCD をドライブするために一般的に非常に低い周波数 (50Hz 程度) で駆動します。

図 18-2 および図 18-3 は 1/2 バイアス ー 1/4 デューティの、COM と SEG 電極用のタイプ A 波形およびタイプ B 波形を説明しています。COM0/COM1 および SEG0/SEG1 のみをデモンストレーションのために描画しました。同様に図 18-4 および図 18-5 は 1/3 バイアス ー 1/4 デューティの、COM と SEG 電極用のタイプ A 波形およびタイプ B 波形を説明しています。

図 18-2. PWM 1/2 タイプ A 波形の例

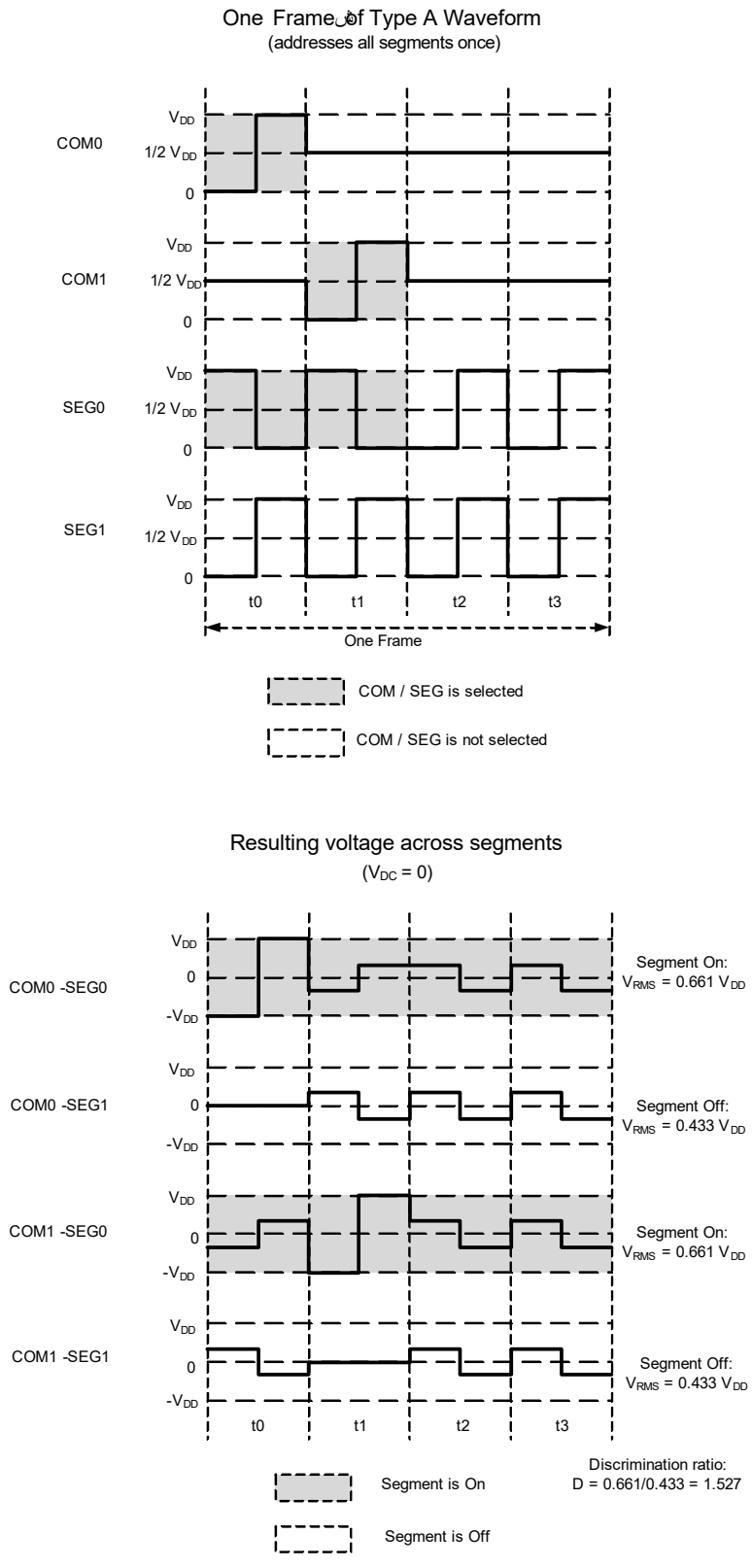
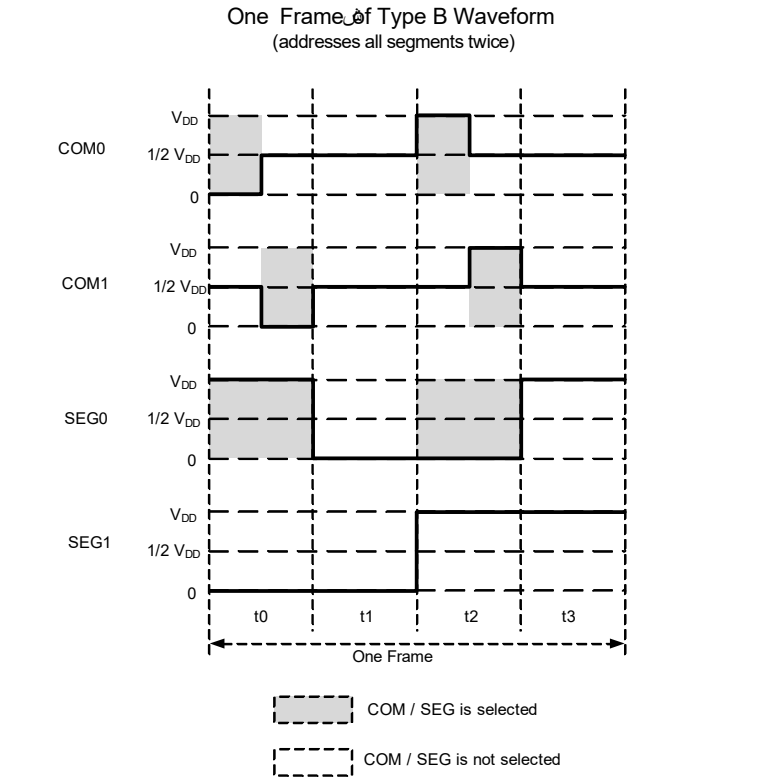


図 18-3. PWM 1/2 タイプ B 波形の例



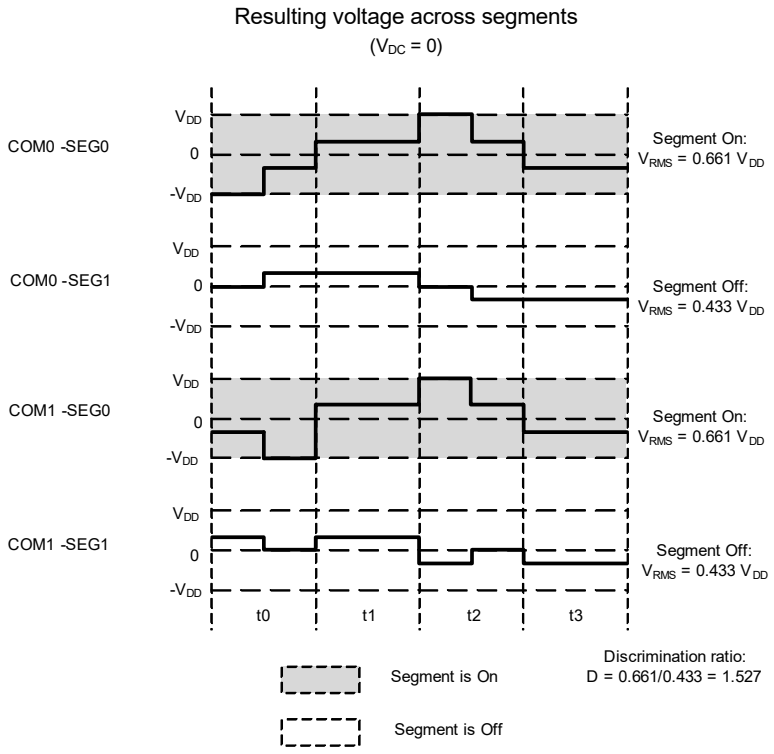


図 18-4. PWM 1/3 タイプ A 波形の例

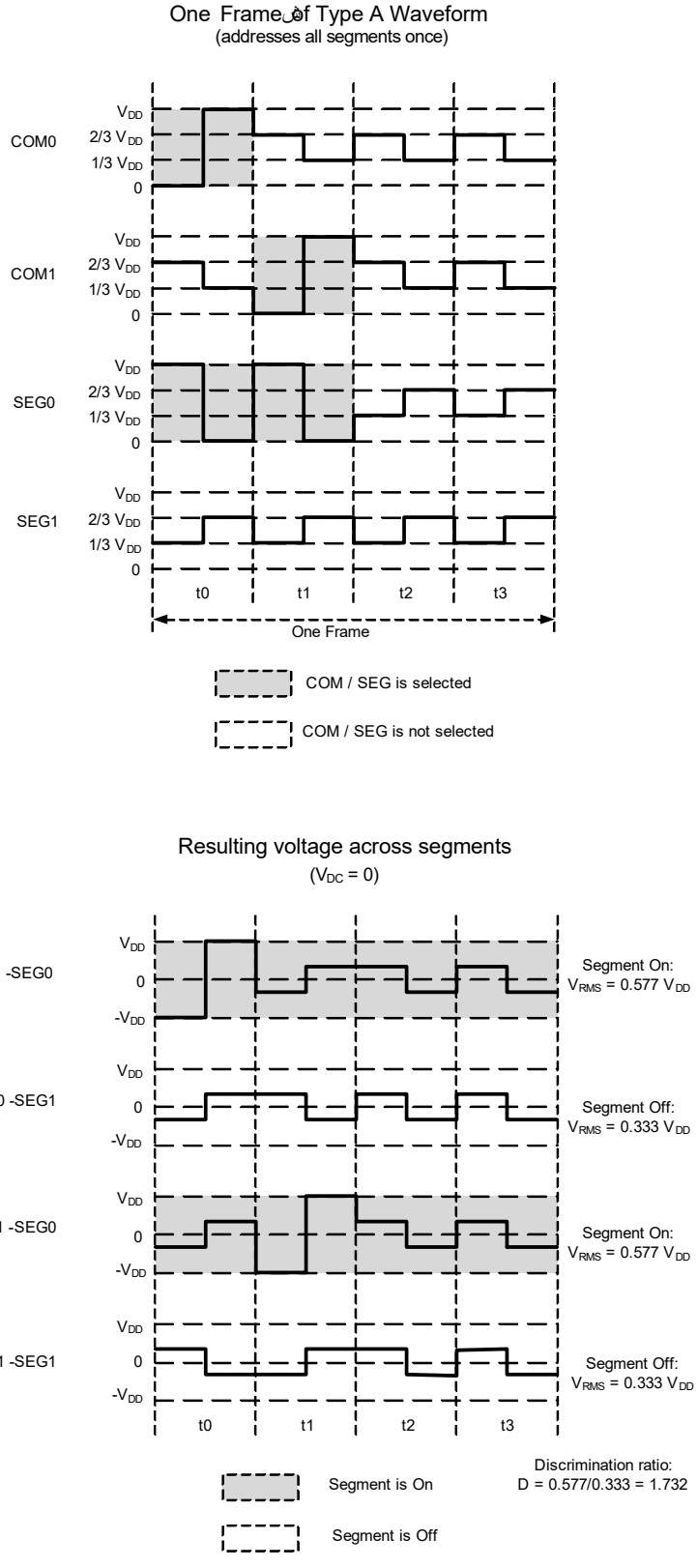
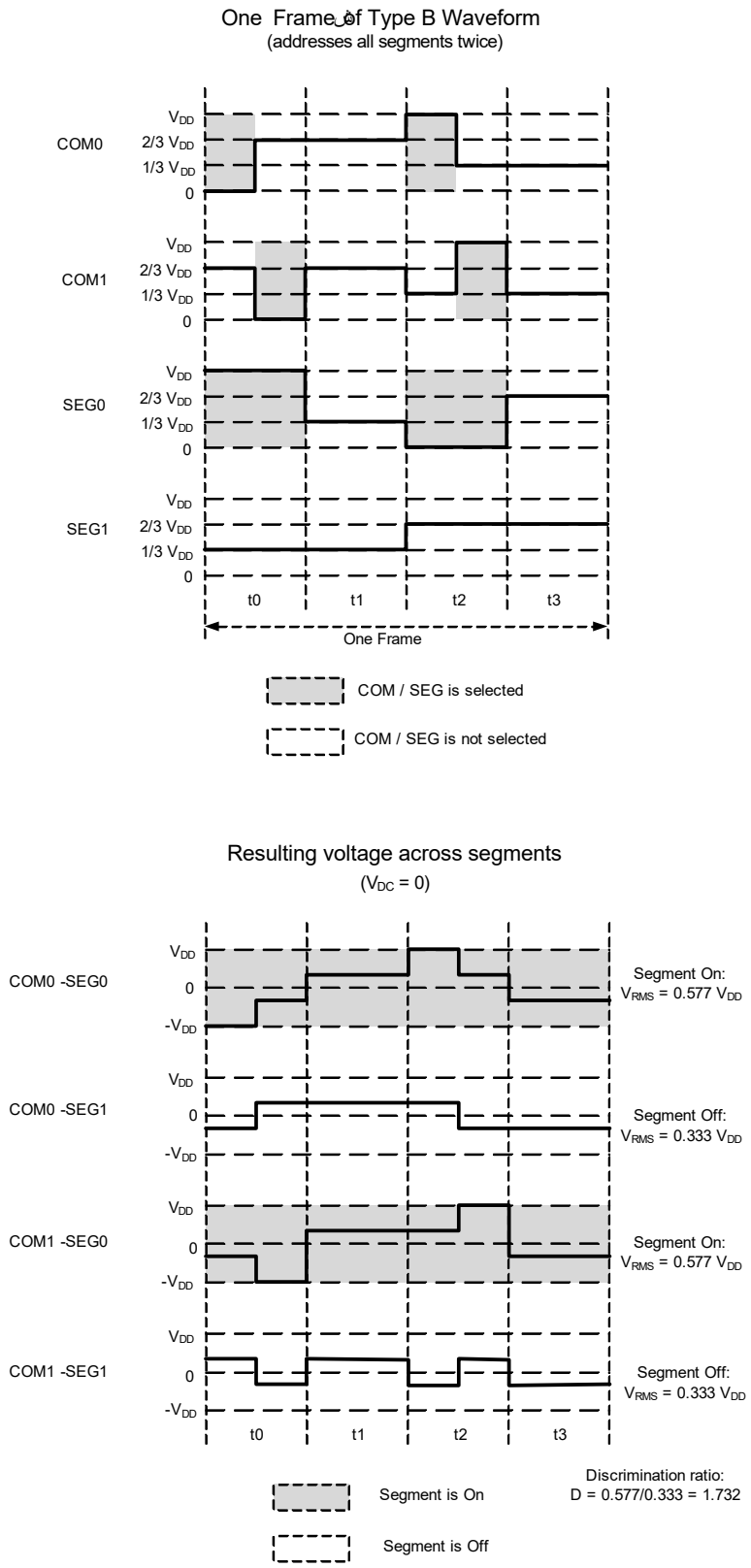


図 18-5. PWM 1/3 タイプ B 波形の例



ON および OFF セグメントの実効的な RMS 電圧は以下の式を使用して容易に計算されます :

$$V_{RMS(OFF)} = \sqrt{\frac{2(B-2)^2 + 2(M-1)}{2M}} x\left(\frac{V_{DRV}}{B}\right)$$

式 18-1

$$V_{RMS(ON)} = \sqrt{\frac{2B^2 + 2(M-1)}{2M}} x\left(\frac{V_{DRV}}{B}\right)$$

式 18-2

ここで B はバイアスであり、M はデューティ (COM の数) です。

例えば COM が 4 つの時、1/2 および 1/3 バイアスの識別比率 (D) はそれぞれ 1.528、1.732 となります。1/3 バイアスは COM ドライブ数が 2 つと 3 つの場合でもより良い識別比率を示します。従って 1/3 バイアスは 1/2 バイアスより良いコントラストを提供し、ほとんどのアプリケーションに推奨されます。1/4 と 1/5 バイアスは LCD の高速動作でのみ利用可能です。これは、COM の多い設計 (4 つ以上) で使用される場合、特に良い識別比率を示します。

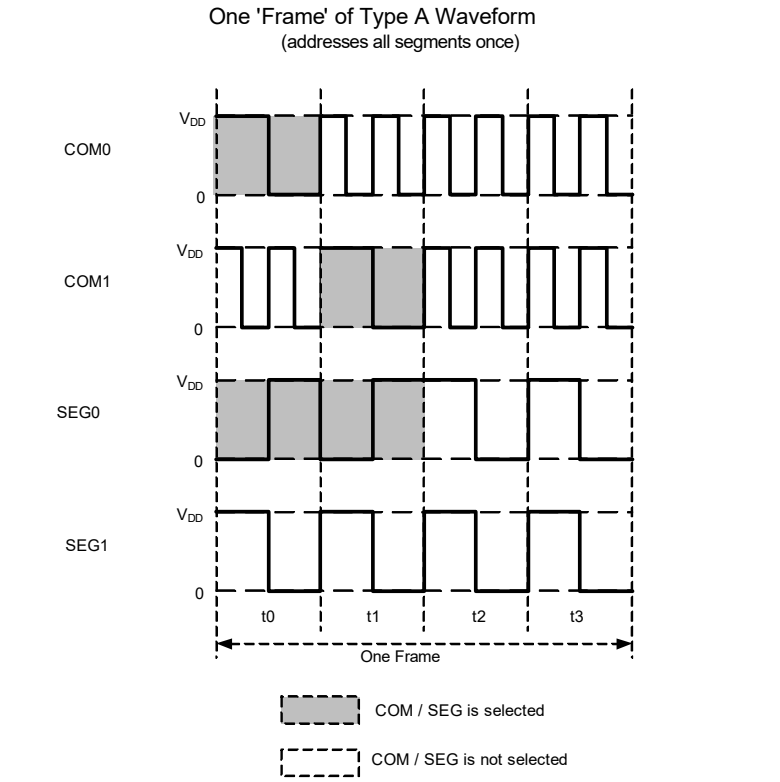
LCD を低速動作させるとき、PWM 信号は ILO から生成されます。32kHz PWM を使用して、許容可能なリップルと立ち上がり、立ち下がり時間で低静電容量ディスプレイを駆動するために、100k から 1MΩ の抵抗を外部で直列に使用します。外部抵抗は、1MHz 程度より大きい PWM 周波数の場合必要ではありません。理想的な PWM 周波数はディスプレイの静電容量および ITO 配線の抵抗に依存します。

1/2 バイアス モードの長所は、COM 信号のみに PWM が必要とされ、SEG 信号は図 18-2 および図 18-3 に示すようにロジック レベルを使用することです。

18.2.1.2 デジタル相関

デジタル相関モードは、バイアス電圧を生成するのではなく、LCD セグメントのコントラストがセグメントにかかる RMS 電圧によって決まるという LCD の性質を利用しています。この手法によって、任意の COM と SEG 信号ペアの相関係数が、対応する LCD セグメントがオンかオフかを決定します。非アクティブのサブフレームで COM 信号のベース ドライブ周波数を 2 倍にします。それにより COM と SEG ドライブ信号の位相関係を、セグメントをオン、オフさせるために変化させることができます。これは PWM ドライブの手法で信号の DC レベルを変更するものとは違います。図 18-8 および図 18-9 は動作原理を説明する波形例です。

図 18-6. デジタル相関タイプ A 波形



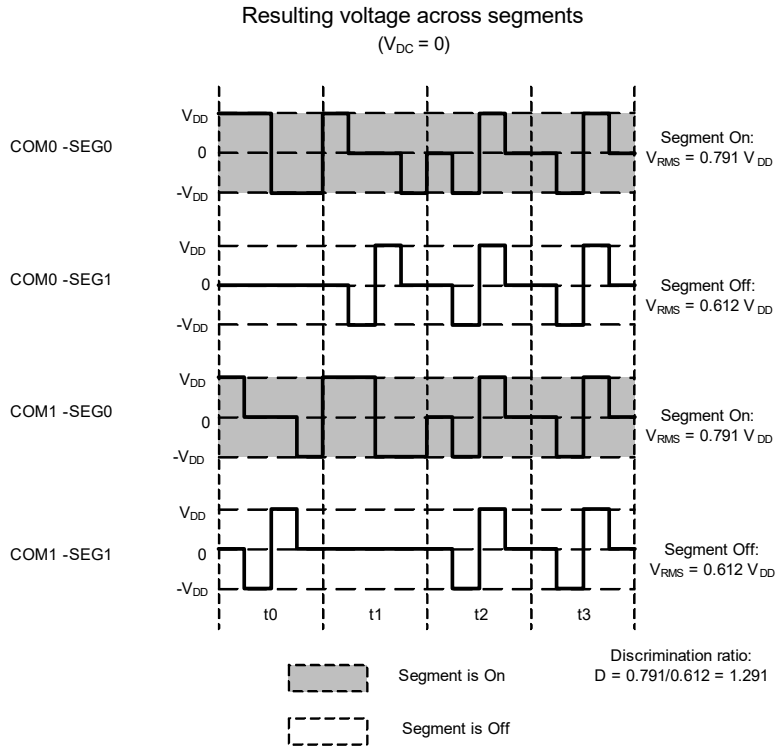
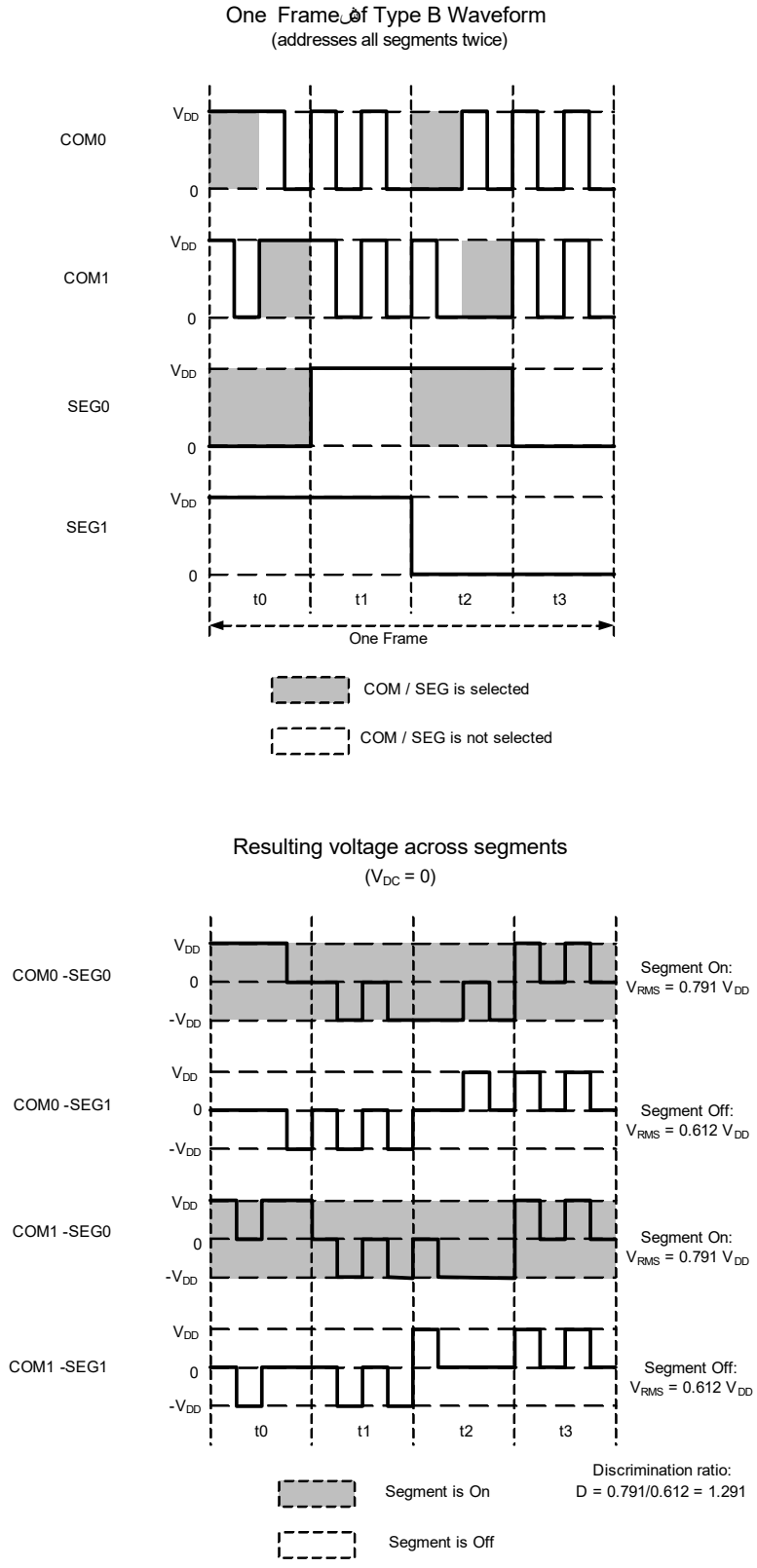


図 18-7. デジタル相関タイプ B 波形



セグメントをオン、オフするために適応される RMS 電圧は下記のように計算できます :

$$V_{RMS(OFF)} = \sqrt{\frac{(M-1)}{2M}} x(V_{DD})$$

$$V_{RMS(ON)} = \sqrt{\frac{2+(M-1)}{2M}} x(V_{DD})$$

ここで B はバイアスであり、M はデューティ (COM の数) です。これは 4 つの COM 用の識別比率の 1.291 をもたらします。デジタル相関モードでは 1.8V の V_{DD} で 3V ディスプレイを駆動できます。

18.2.2 ドライブ モードの推奨使用方法

18.2.1.1 PWM ドライブおよび 18.2.1.2 デジタル相関で説明するように、PWM 駆動モードはデジタル相関モードと比べ、より高い識別比率を得られます。従って、デジタル相関手法のコントラストは PWM 手法のコントラストより低くなりますが、デジタル相関は信号を低周波数でトグルするため電力消費量がより少なく済みます。

デジタル相関モードは TN ディスプレイに対して少しコントラストが低くなります。しかし、よりコントラストの高い STN ディスプレイに対しては、コントラスト、視野角に大差はありません。各モードには長所と短所があります。推奨される使用方法は以下の通りです。

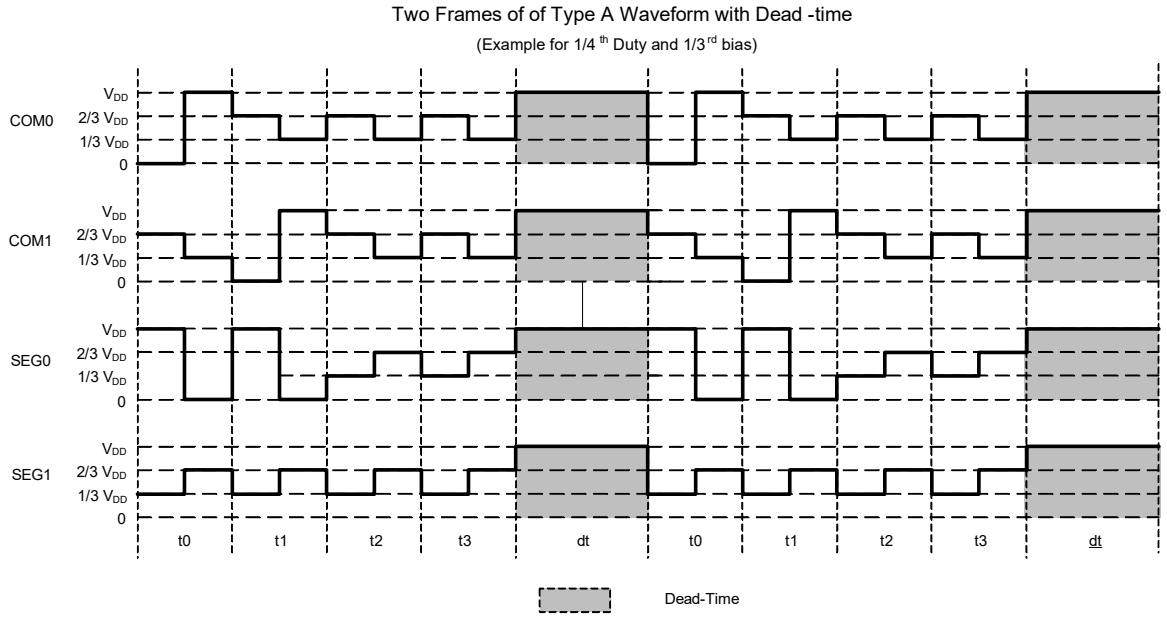
表 18-1. 駆動モードの推奨使用方法

ディスプレイ タイプ	ディープスリープ モード	スリープ／アクティブ モード	注 :
4つのCOMがあるTN ガラス	デジタル相関	PWM 1/3 バイアス	ファームウェアは、ディープ スリープ モードへの遷移またはアクティブ モードへの復帰前に LCD 駆動モードを切り替えねばならない
4 つの COM がある STN ガラス	デジタル相関		PWM ドライブはコントラストに関して STN ガラスに対する利点がない
8 つの COM, STN	非対応	1/4 バイアスと 1/5 バイアス PWM	高速 LCD モードでのみサポートする。低速クロックは PWM を高いマルチプレックス比で動作するために十分に早くない

18.2.3 デジタル コントラスト制御

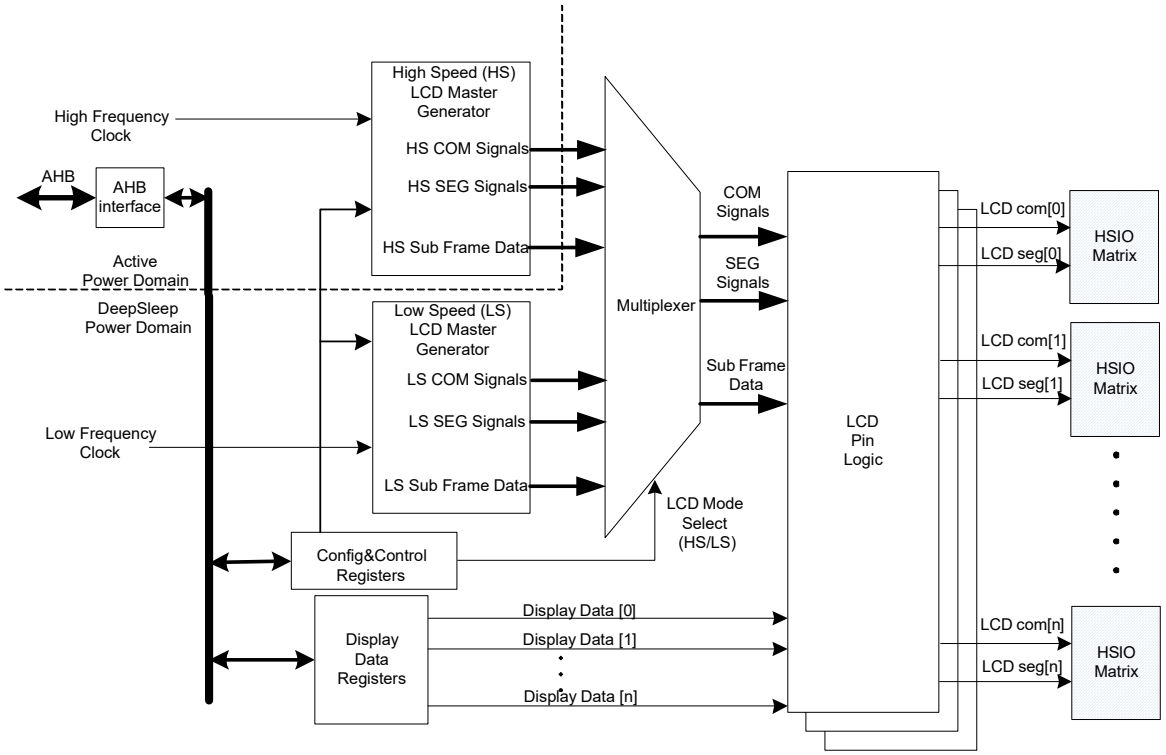
すべての駆動モードで、デジタル コントラスト制御はセグメントのコントラストを変更するために使用されます。この方法は、セグメントの駆動時間を減少させることによってコントラストを減少させます。これは各フレーム後にデッド タイムを入れることによって実行されます。デッド タイム中にすべての COM および SEG 信号は論理 1 に駆動されます。デッド タイムは高分解能で制御されます。図 18-8 は 1/3 バイアスと 1/4 デューティ搭載のデッド タイム コントラスト制御方法を説明します。

図 18-8. デッド タイム コントラスト制御



18.3 ブロック図

図 18-9. LCD ダイレクト ドライブ システムのブロック図



18.3.1 動作原理

LCD コントローラー ブロックは 2 つの信号発生器を持ちます。高速クロック ソース HFCLK を持つものと ILO に駆動される低速クロック ソースを持つものです。これらはそれぞれ高速LCD マスター信号発生器および低速LCD マスター信号発生器と呼ばれています。両方の信号発生器は PWM モードおよびデジタル相関 ドライブ モードをサポートします。低速信号発生器の PWM ドライブ モードは 162 ページの PWM ドライブに述べるように外部抵抗が必要です。

マルチプレクサは、ファームウェアによって設定され、この 2 つの信号発生器出力の 1 つを選択します。LCD ピン回路ブロックは COM 出力および SEG 出力を、信号発生器から対応する I/O マトリックスに配線します。任意の GPIO を COM または SEG として使用することができます。COM または SEG のピン割り付けの設定は、GPIO および I/O マトリックスで行なわれます。42 ページの高速 I/O マトリックスを参照してください。これらの 2 つの信号発生器は同じコンフィギュレーション レジスタを共有します。I/O レジスタをマッピングするこれらのメモリは、AHB インターフェースを使うシステム バス (AHB) に接続されます。

LCD コントローラーはアクティブ、スリープ、ディープ スリープの 3 つの電力モードで動作します。高速動作はアクティブおよびスリープ モードで動作します。低速動作はアクティブ、スリーブおよびディーブスリーブ モードで動作します。LCD コントローラーはハイバネートおよびストップ モードでは電力が供給されません。

18.3.2 高速および低速のマスター信号発生器

高速および低速のマスター信号発生器は相互に似ています。1 つの違いは、高速バージョンがフレームとサブフレーム周期を発生させるために、分周比の大きな分周器を利用することです。これは、高速ブロック (HFCLK) のクロックが、低速ブロックに送られる ILO クロックの周波数に比べ一般的に 30 ～ 100 倍である IMO から生成されることによります。高速信号発生器はアクティブ電力モードで動作し、低速信号発生器はディープ スリープモードで動作します。1 組のコンフィギュレーション レジスタが、高速ブロックと低速ブロックの両方を制御するために提供されます。各マスター信号発生器は以下の機能および特長があります :

- タイプ A またはタイプ B ドライブ波形用のブロックを構成するレジスタ ビット (LCD_CONTROL レジスタにある LCD_MODE ビット)
- COM の数を選択するためのレジスタ ビット (LCD_CONTROL レジスタにある COM_NUM フィールド) 指定できる値は 2、3、4 です。
- 選択する動作モードのコンフィギュレーションビットを有効にする :
  - デジタル相関
  - PWM 1/2 バイアス
  - PWM 1/3 バイアス
  - 1/4 バイアスでの PWM (低速信号発生器でサポートしない)
  - 1/5 バイアスでの PWM (低速信号発生器でサポートしない)
  - オフ／ディセーブル。2 つある信号発生器の内 1 つがオフになるように通常設定される

  LCD_CONTROL ビットにある OP_MODE および BIAS フィールドで駆動モードを選択する
- サブフレームのタイミングを生成するためのカウンター。LCD_DIVIDER レジスタにある SUBFR_DIV フィールドは各サブフレーム時間を決定する。このカウンターに書き込まれる分周値が C であれば、サブフレーム周期は 4 × (C+1) である。低速信号発生器は 8 ビット カウンターを持っている。このカウンタは ILO クロックから最大 8 ms 周期の信号を発生する。高速信号発生器は 16 ビット カウンターを持っている
- デッド タイム周期を生成するためのカウンター。これらのカウンターはサブフレーム周期カウンターと同じビット数があり、同じクロックを使用する。LCD_DIVIDER レジスタにある DEAD_DIV フィールドはデッド タイム周期を制御する

18.3.3 マルチプレクサおよび LCD ピン回路

マルチプレクサは高速または低速マスター信号発生器ブロックの出力信号を選択し、LCD ピン回路に送ります。この選択は設定および制御レジスタによって行います。LCD ピン回路はマルチプレクサからのサブフレーム信号を使ってディスプレイ データを選びます。このピン回路は各 LCD ピン用に複製されます。

18.3.4 ディスプレイ データ レジスタ

各 LCD セグメント ピンは、ディスプレイ データ レジスタ LCD_DATAnx を持つ LCD ポートの一部です。デバイスにはこの LCD ポートが 8 つあります。これらのポートは実際のピン ポートではありませんが、セグメントをコモンにマッピングするために LCD ハードウェアで利用できるポート／コネクションであることにご注意ください。設定された各 LCD セグメントは、これら LCD ポートのピンと見なされます。LCD_DATAnx レジスタは 32 ビット幅であり、このデザインで有効にされる SEG-COM の組み合わせのすべての ON/OFF データを保持します。LCD_DATA0x は COM0 ～ COM3 の SEG-COM データを保持し、LCD_DATA1x は COM4 ～ COM7 の SEG-COM データを保持します。LCD_DATA0x レジスタのビット [4i+3:4i] (「i」がピン数) は、Pin[i] について Port[x] と COM[3,2,1,0] の組み合わせの ON/OFF データを表します。それを表 18-2 に示します。LCD_DATAnx レジスタは、フレーム単位のディスプレイ データによってプロ

グラムします。ディスプレイ データ レジスタは I/O をマッピングしたメモリ (MMIO) であり、AHB スレーブ インターフェースを通じてアクセスされます。

表 18-2. LCD_DATA0x レジスタの SEG-COM マッピング (各 SEG は LCD ポートのピン)

BITS[31:28] = PIN_7[3:0]				BITS[27:24] = PIN_6[3:0]			
PIN_7-COM3	PIN_7-COM2	PIN_7-COM1	PIN_7-COM0	PIN_6-COM3	PIN_6-COM2	PIN_6-COM1	PIN_6-COM0
BITS[23:20] = PIN_5[3:0]				BITS[19:16] = PIN_4[3:0]			
PIN_5-COM3	PIN_5-COM2	PIN_5-COM1	PIN_5-COM0	PIN_4-COM3	PIN_4-COM2	PIN_4-COM1	PIN_4-COM0
BITS[15:12] = PIN_3[3:0]				BITS[11:8] = PIN_2[3:0]			
PIN_3-COM3	PIN_3-COM2	PIN_3-COM1	PIN_3-COM0	PIN_2-COM3	PIN_2-COM2	PIN_2-COM1	PIN_2-COM0
BITS[7:3] = PIN_1[3:0]				BITS[3:0] = PIN_0[3:0]			
PIN_1-COM3	PIN_1-COM2	PIN_1-COM1	PIN_1-COM0	PIN_0-COM3	PIN_0-COM2	PIN_0-COM1	PIN_0-COM0

18.4 レジスター覧

表 18-3. LCD ダイレクト ドライブ レジスター覧

レジスタ名	説明
LCD_ID	このレジスタは LCD コントローラー ID とリビジョン番号を保持
LCD_DIVIDER	このレジスタはサブフレームおよびデッド タイム周期を制御
LCD_CONTROL	このレジスタは高速信号発生器および低速信号発生器を設定するのに使用される
LCD_DATA0x	COM0 から COM3 までの LCD ポート ピン データ レジスタ ; x = ポート番号、8 ポートを使用可能
LCD_DATA1x	COM4 から COM7 までの LCD ポート ピン データ レジスタ ; x = ポート番号、8 ポートを使用可能

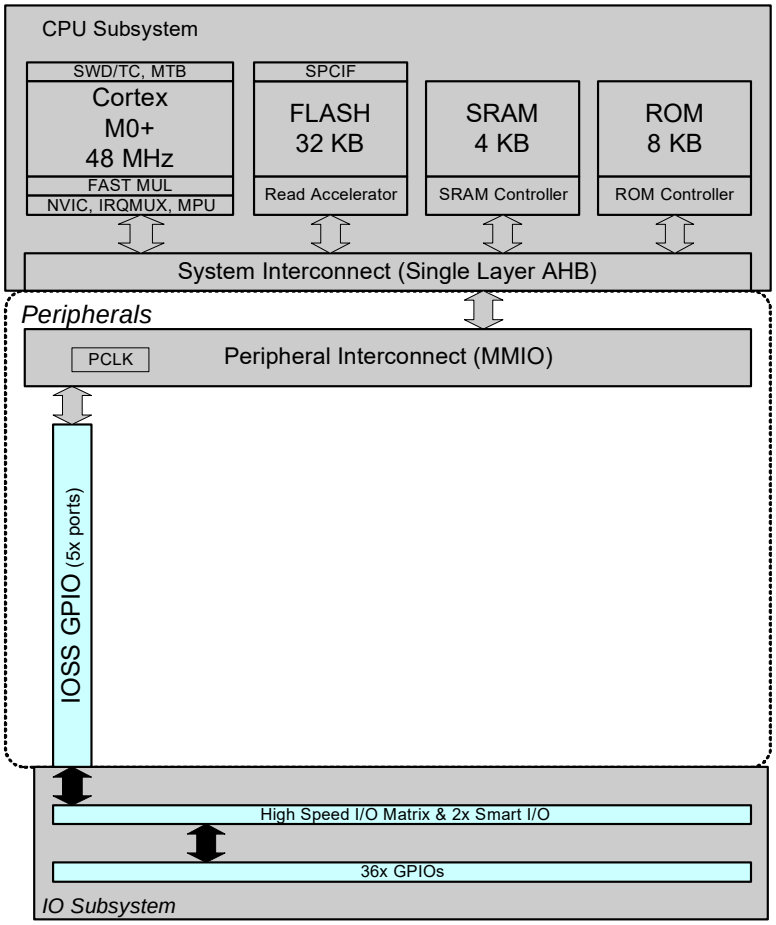
セクション F:プログラムおよびデバッグ

このセクションは次の章を含みます :

- 175 ページのプログラムおよびデバッグ インターフェース
- 182 ページの不揮発性メモリ プログラム

トップ レベル アーキテクチャ

プログラムおよびデバッグ ブロック図



19. プログラムおよびデバッグ インターフェース

PSoC® 4 のプログラムおよびデバッグ インターフェースは、外部デバイスがプログラムとデバッグ処理を実行するための通信ゲートウェイを提供します。その外部デバイスは、サイプレスが提供するプログラマやデバッガまたはプログラムおよびデバッグに対応するサードパーティのデバイスです。シリアル ワイヤ デバッグ (SWD) インターフェースは、外部デバイスと PSoC 4 間の通信プロトコルとして使用されます。

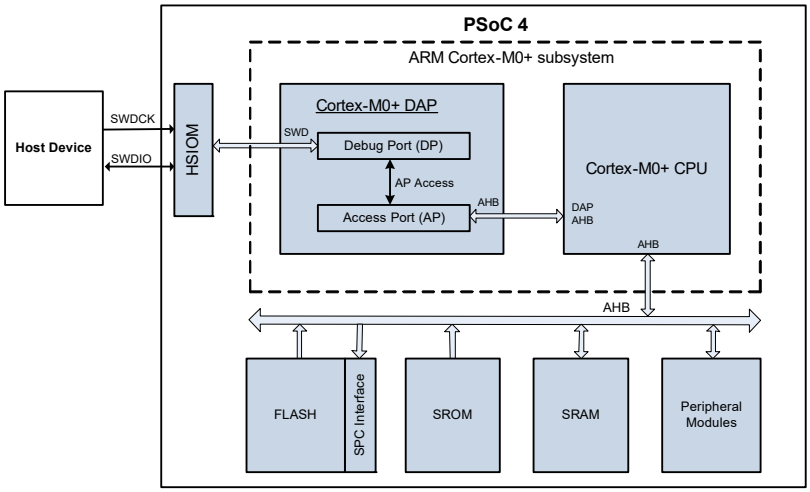
19.1 特長

- SWD インターフェースを介したプログラムおよびデバッグ
- デバッグ間に 4 点のハードウェア ブレークポイントおよび 2 点のハードウェア ウォッチポイントを搭載
- デバッグ間にシステム内のすべてのメモリとレジスタ (コアが動作中または停止した時の Cortex-M0+ レジスタ バンクも含む) へ読み書きアクセス可能

19.2 機能説明

図 19-1 に PSoC 4 におけるプログラムおよびデバッグ インターフェースのブロック図を示します。Cortex-M0+ のデバッグとアクセス ポート (DAP) は、プログラムおよびデバッグ インターフェースとして機能します。外部プログラマまたはデバッガはホストとしてターゲットである PSoC 4 の DAP と通信します。通信は SWD インターフェースの双方向データ ピン (SWDIO) とホストが駆動するクロック ピン (SWDCK) の 2 つで行います。部品上の SWD ポート ピン (SWDIO と SWDCK) は高速 I/O マトリックス (HSIOM) を通して DAP と通信します。HSIOM の詳細は 37 ページの I/O システムの章 を参照してください。

図 19-1. プログラムおよびデバッグ インターフェース



DAP は、Arm 固有の AHB インターフェース (advanced high-performance bus) を使用して Cortex-M0+ CPU と通信します。AHB はデバイス内部で使用されるシステム相互接続プロトコルであり、AHB マスターによるメモリとペリフェラル レジスタへのアクセスを円滑にします。デバイスは、Arm CM0 CPU コアと DAP の 2 つの AHB マスターを備えています。外部デバイスは、プログラムおよびデバッグ処理を実行するために、DAP を介してデバイス全体を効率的に管理できます。

19.3 シリアル ワイヤ デバッグ (SWD) インターフェース

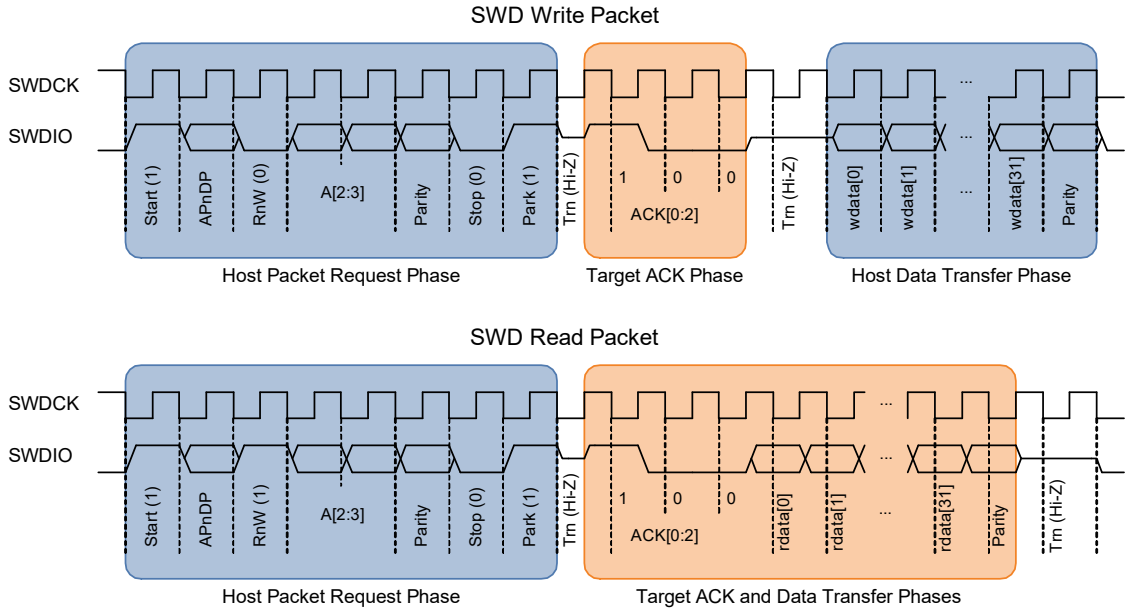
PSoC 4 の Cortex-M0+ は、SWD インターフェースを介したプログラムおよびデバッグに対応しています。SWD プロトコルは、パケット ベースのシリアル トランザクション プロトコルです。ピン レベルでは双方向データ信号 (SWDIO) と一方向クロック信号 (SWDCK) を使用します。クロック ラインは常にホスト プログラマが駆動し、データ ラインはホストまたはターゲットのいずれかが駆動します。完全なデータ転送 (1 つの SWD パケット) は 46 クロックを必要とし、次の 3 つの段階から成り立っています。

- **ホスト パケット要求段階** – ホストはPSoC 4 ターゲットに要求を発行します。
- **ターゲット ACK応答段階** – PSoC 4 ターゲットはホストに確認応答を送信します。
- **データ転送段階** – 転送方向に応じてホストまたはターゲットがバスにデータを書き込みます。

SWDIO ラインの制御がホストからターゲットに (逆もまた同様) 方向を変える場合、ターンアラウンド期間が発生します。その期間中デバイスはラインを駆動せず高インピーダンス (Hi-Z) 状態になります。ターンアラウンド期間はクロック サイクルの 2 分の 1 または 1 サイクル半となります。

図 19-2 に SWD パケットの読み書きタイミング図を示します。

図 19-2. SWD 読み書きパケットのタイミング図



SWD 読み書きパケットを送信するシーケンスは下記の通りです :

1. ホスト パケット要求段階 : SWDIO はホストによって駆動されます。
   a. スタート ビットで転送を開始します。スタート ビットは常に論理 1 です。
   b. 「AP not DP」(APnDP) ビットは、転送が AP アクセス (1b1) であるか、それとも DP アクセス (1b0) であるかを決定します。
   c. 「Read not Write」(RnW) ビットは、データの転送方向を制御します。1b1 はターゲットからの読み出しを、1b0 はターゲットへの書き込みを意味します。
   d. アドレス ビット (A[3:2]) は APnDP ビットの値に応じて、AP または DP に対応するレジスタ選択ビットです。定義については表 19-3 および表 19-4 を参照してください。**注** アドレス ビットは、最下位ビットから送信されます。
   e. パリティ ビットには APnDP、RnW および ADDR ビットのパリティが含まれています。これは偶数パリティであり、他のビットとの XOR (排他的論理和) がとられると結果は 0 になります。
   パリティ ビットが正しくない時ヘッダーは PSoC 4 によって無視され、ACK 応答はありません (ACK = 3b111)。プログラム動作は中断されデバイス リセットを行うことで再試行します。
   f. ストップ ビットは常に論理 0 です。
   g. パーク ビットは常に論理 1 です。
2. ターゲット ACK 応答段階 : SWDIO はターゲットによって駆動されます。
   a. ACK[2:0] ビットはターゲットからホストへの応答を表し、他の結果の中でとりわけ通信に失敗した或いは成功したことを示します。定義については表 19-1 を参照してください。**注** ACK ビットは、最下位ビットから送信されます。
3. データ転送段階 : SWDIO は転送方向に応じて、ターゲットまたはホストによって駆動されます。

a. 読み書き用のデータは最下位ビットからバスに書き込まれます。
b. データのパリティ ビットは読み書きされるデータのパリティを示します。これは偶数パリティであり、データ ビットとの XOR (排他的論理和) がとられると結果は 0 になります。
パリティ ビットがデータ エラーを示している場合は訂正処理を行う必要があります。パケットを読み出す場合、ホストがパリティ エラーを検出するとプログラム動作を中断させ再起動します。パケットを書き込む場合、ターゲットがパリティ エラーを検出すると次のパケットで FAULT 応答 (否定応答) を生成します。

SWD プロトコルでは、ホストは 2 パケット間で SWDIO を LOW にした状態であれば SWDCK クロック サイクルをいくつ生成しても構いません。クロックがフリーランしていない場合 2 つの SWD パケット間に 3 以上のダミー クロック サイクルを生成すること、またはクロックを IDLE モードでフリーランさせることを推奨します。

SWD インターフェースは、SWDIO を HIGH にして 50 サイクル以上 SWDCK ラインをクロッキングすることによってリセットすることができます。アイドル状態に復帰させるために、SWDIO を LOW にしてもう一回クロックインします。

19.3.1 SWD タイミングの詳細

SWDIO ラインは通信方向に応じて異なる時間で読み書きされます。ホスト パケット要求段階およびデータ転送段階 (ホストがターゲットにデータを書き込んでいる場合) において、ホストは SWDIO ラインを駆動します。ホストが SWDIO ラインを駆動している時、新しいビットは SWDCK 立ち下がりエッジでホストによって書き込まれ、SWDCK 立ち上がりエッジでターゲットによって読み出されます。ターゲット Ack 応答段階およびデータ転送段階 (ターゲットがデータを読み出している場合) において、ターゲットは SWDIO ラインを駆動します。ターゲットが SWDIO ラインを駆動している時、新しいビットは SWDCK 立ち上がりエッジでターゲットによって書き込まれ、SWDCK 立ち下がりエッジでホストによって読み出されます。

表 19-1 および図 19-2 に SWDIO ビットの読み書きタイミングを示します。

表 19-1. SWDIO ビットの書き込みおよび読み出しタイミング

SWD パケットの段階	SWDIO エッジ	
	立ち下がり	立ち上がり
ホスト パケット要求	ホスト書き込み	ターゲット読み出し
ホスト データ転送		
ターゲット ACK 応答	ホスト読み出し	ターゲット書き込み
ターゲット データ転送		

19.3.2 ACK 応答の詳細

確認応答 (ACK) ビット フィールドは、前の転送の状態を示すために使用されます。OK 応答は前のパケットが正常に配信されたことを意味します。WAIT 応答はデータ フェーズを必要とします。FAULT 状態の場合プログラム動作を直ちに中断させる必要があります。表 19-2 は ACK ビット フィールド コードの詳細を示します。

表 19-2. SWD 転送 ACK 応答のコード

応答	ACK[2:0]
OK	3b001
WAIT	3b010
FAULT	3b100
NO ACK	3b111

以下は WAIT 応答および FAULT 応答の詳細です。

- WAIT 応答において、トランザクションが読み出しの場合、ホストはデータ フェーズで読み出されたデータを無視する必要があります。ターゲットはラインを駆動せず、ホストもパリティ ビットをチェックしません。
- WAIT 応答において、トランザクションが書き込みの場合、データ フェーズは PSoC 4 によって無視されます。ただしホストはパケット送信を完了させるために依然として書き込みデータを送信します。データに対応するパリティ ビットもホストによって送信される必要があります。
- WAIT 応答において、PSoC 4 が以前のトランザクションを処理していることを意味します。OK 応答が受信されたか確認するためにホストは連続して最大 4 回の WAIT 応答を送信できます。失敗した場合はプログラム動作を中断させ再試行します。
- FAULT 応答の場合デバイス リセットを実行することによりプログラム動作を中断させ再試行します。

19.3.3 ターンアラウンド (Trn) 期間の詳細

ホスト書き込み転送の場合図 19-2 に示す通り、パケット要求段階と ACK 応答段階の間および ACK 応答段階とデータ転送段階の間にターンアラウンド期間があります。SWD プロトコルに従って、Trn 期間は、ホストとターゲットの両方によってそれぞれの SWDIO ラインで駆動モードを変更するために使用されます。パケット要求後の最初の Trn 期間中に、ターゲットは SWDCK の立ち上がりエッジで SWDIO ライン上の ACK データを駆動し始めます。この動作によりホストが次の立ち下がりエッジで ACK データを読み出すことを確保します。従って最初の Trn の持続時間はサイクルの半分しかありません。SWD パケットの 2 番目の Trn 期間は 1 サイクル半です。Trn 期間中はホストも PSoC 4 も SWDIO ラインを駆動しません。

19.4 Cortex-M0+ デバッグおよびアクセス ポート (DAP)

Cortex-M0+ プログラムおよびデバッグ インターフェースにはデバッグ ポート (DP) とアクセス ポート (AP) があります。これが DAP を形成しています。デバッグ ポートは、ホスト デバイスとの通信を可能にする SWD インターフェース プロトコルのステートマシンを搭載しています。またアクセス ポート コンフィギュレーション用のレジスタ、DAP 識別コード等も含んでいます。アクセス ポートには外部デバイスが Cortex-M0+ DAP-AHB インターフェースにアクセスすることを許可するレジスタがあります。一般的に DP レジスタは、ワンタイム プログラムまたはエラー検出の目的に使用され、AP レジスタはプログラムおよびデバッグ動作を実行するために使用されます。DAP アーキテクチャの詳細については、Arm® Debug Interface v5 Architecture Specification を参照してください。

19.4.1 デバッグ ポート (DP) レジスタ

表 19-3 はプログラムおよびデバッグに使用する Cortex-M0+ DP レジスタを、対応する SWD アドレス ビットと共に示しています。DP レジスタ アクセスの場合、APnDP ビットは常に 0 です。2 つのアドレス ビット (A[3:2]) は、異なる DP レジスタの中から選択するために使用されます。同じアドレス ビットでも読み出し動作か書き込み動作かに応じて、異なる DP レジスタをアクセスすることにご注意ください。すべての DP レジスタの詳細は Arm® デバッグ インターフェース v5 アーキテクチャ仕様を参照してください。

表 19-3. 主なデバッグ ポート (DP) レジスタ

レジスタ	APnDP	アドレス A[3:2]	RnW	レジスタ名	レジスタの機能
ABORT	0 (DP)	2b00	0 (W)	AP アボート レジスタ	このレジスタは DAP を強制中断させ、エラー フラグと自動クリアされないフラグをクリアするために使用される
IDCODE	0 (DP)	2b00	1 (R)	識別コード レジスタ	このレジスタは 0x0BC11477 である Cortex-M0+CPU の SWD ID を保持する
CTRL/STAT	0 (DP)	2b01	X (R/W)	制御と状態のレジスタ	このレジスタは DP の制御を可能にし、DP の状態情報を含む
SELECT	0 (DP)	2b10	0 (W)	AP 選択レジスタ	このレジスタは AP を選択するために使用される。PSoC 4 に APは1つしかなく、それがDAP AHBとインターフェースする
RDBUFF	0 (DP)	2b11	1 (R)	読み出しバッファ レジスタ	このレジスタは最後の AP 読み出し動作の結果を保持する

19.4.2 アクセス ポート (AP) レジスタ

表 19-4 はプログラムおよびデバッグに使用する主な Cortex-M0+ AP レジスタを、対応する SWD アドレス ビットと共に示しています。AP レジスタ アクセスの場合、APnDP ビットは常に 1 です。2 つのアドレス ビット (A[3:2]) は、異なる AP レジスタを選択するために使用されます。

表 19-4. 主なアクセス ポート (AP) レジスタ

レジスタ	APnDP	アドレス A[3:2]	RnW	レジスタ名	レジスタの機能
CSW	1 (AP)	2b00	X (R/W)	制御と状態のワード レジスタ (CSW)	このレジスタはメモリ アクセス ポートを介して接続されたメモリ システム (PSoC 4 メモリ マップ) へのアクセスをコンフィギュレーションし、制御する
TAR	1 (AP)	2b01	X (R/W)	転送アドレス レジスタ	このレジスタに読み書き操作の対象とする 32 ビット メモリ アドレスを指定
DRW	1 (AP)	2b11	X (R/W)	データ読み書きレジスタ	このレジスタは TAR レジスタで指定されたアドレスに読み書きする 32 ビットのデータを保持

19.5 PSoC 4 デバイスのプログラム

PSoC 4 は下記のシーケンスでプログラムされます。プログラム動作に必要となるプログラム アルゴリズム、タイミング仕様、ハードウェア コンフィギュレーションの完全な詳細情報については、PSoC 4 デバイス プログラミング仕様、PSoC 4-BLE プログラミング仕様、PSoC 4100M、PSoC 4200M、PSoC 4200D、PSoC 4400、PSoC 4000S、PSoC 4700S のデバイス プログラミング仕様、PSoC 4200L デバイス プログラミング仕様を参照してください。

1. PSoC 4 で SWD ポートを開通させます。
2. プログラム モードに移行します。
3. シリコン ID 確認、フラッシュ プログラム、フラッシュ検証、チェックサム検証などといったデバイス プログラム ルーチンを実行します。

19.5.1 SWD ポートの開通

19.5.1.1 SWD ポートの開通シーケンス

デバイス プログラムにおける最初の手順は、ホストがターゲットの SWD ポートを開通させることです。ホストはまず外部リセット (XRES) ピンをアサートしデバイスをリセットします。XRES 信号を解除した後、ホストは DAP の SWD インターフェースに接続するために、開通タイムアウト以内に SWD 接続シーケンスを送信しなければなりません。以下はそのシーケンスの疑似コードです。

コード 1. SWD ポート開通の擬似コード

```
ToggleXRES(); // Toggle XRES pin to reset device

//Execute Arm's connection sequence to acquire SWD-port
do
{
	SWD_LineReset(); //perform a line reset (50+ SWDCK clocks with SWDIO high)
	ack = Read_DAP ( IDCODE, out ID); //Read the IDCODE DP register

}while ((ack != OK) && time_elapsed < ms); //retry connection until OK ACK or timeout

if (time_elapsed >= ms) return FAIL; //check for acquire time out

if (ID != CM0P_ID) return FAIL; //confirm SWD ID of Cortex-M0+ CPU. (0x0BC11477)
```

擬似コードでの SWD_LineReset() は、デバッグ アクセス ポートをリセットするための標準 Arm コマンドです。SWDIO が SWDCK クロック サイクルで 49 以上の期間 HIGH になるよう構成されています。SWDIO を LOW にアサートして、少なくとも 1 つの SWDCK クロック サイクルを送信することにより、トランザクションを完了しなければなりません。このシーケンスでプログラマとチップを同期します。Read_DAP() は、デバッグ ポートでの IDCODE レジスタの読み出しを意味します。ライン リセットおよび IDCODE 読み出しのシーケンスは、IDCODE 読み出しに対して OK 応答を受信するかまたはタイムアウト (ms) が発生するまで繰り返します。時間枠以内に OK 応答を受信し、IDCODE 読み出しが Cortex-M0+ DAP の IDCODE と一致する場合、SWD ポートは開通状態になったと見なされます。

19.5.2 SWD プログラム モードへの移行

SWD ポートを開通させた後、ホストは特定の時間内にデバイスのプログラム モードに移行する必要があります。これはテスト モード制御レジスタ (MODE レジスタ) 内の TEST_MODE ビット (ビット 31) を設定することにより行われます。またデバイスのプログラム モードに入る前に、デバッグ ポートもコンフィギュレーションする必要があります。プログラム モードに移行するためのタイミング仕様および擬似コードは、PSoC 4100M、PSoC 4200M、PSoC 4200D、PSoC 4400、PSoC 4000S、PSoC 4700S、デバイスのプログラム仕様の資料に詳しく記載されています。ポート開通の手段とこの手段を実行するために最低限必要なクロック周波数は 1.5MHz です。

19.5.3 SWD プログラム ルーチンの実施

デバイスがプログラム モードの時、外部プログラマは、フラッシュ消去、フラッシュ プログラム、チェックサム検証などといったプログラム動作を実行するために、SWD パケット シーケンスを送信し始めます。プログラム ルーチンは 182 ページの不揮発性メモリ プログラムの章 で説明されています。プログラム ルーチンを呼び出す正確なシーケンスについては PSoC 4 デバイス プログラミング仕様、PSoC 4-BLE プログラミング仕様、PSoC 4100M、PSoC 4200M、PSoC 4200D、PSoC 4400、PSoC 4000S、PSoC 4700S のデバイス プログラミング仕様、PSoC 4200L デバイス プログラミング仕様の資料に記載されています。

19.6 PSoC 4 SWD デバッグ インターフェース

Cortex-M0+ DAP デバッグ機能は、侵襲性デバッグおよび非侵襲性デバッグの 2 種類に分けられています。侵襲性デバッグには、プログラムのホールトとステップ実行、ブレークポイントおよびデータ ウォッチポイントが含まれています。非侵襲性デバッグには、命令アドレス プロファイリング、そしてフラッシュ メモリ、SRAM および他のペリフェラル レジスタを含むデバイス メモリ アクセスがあります。

DAP は以下の 3 つの主なデバッグ サブシステムを備えています。

- デバッグ制御およびコンフィギュレーション レジスタ
- ブレークポイント ユニット (BPU) – ブレークポイント サポートを提供
- デバッグ ウォッチポイント (DWT) – ウォッチポイント サポートをサポートしますが、トレース実行は Cortex-M0+ デバッグで未サポートです。

デバッグ アーキテクチャの詳細情報については Armv6-M アーキテクチャ リファレンス マニュアルを参照してください。

19.6.1 デバッグ制御およびコンフィギュレーション レジスタ

デバッグ制御およびコンフィギュレーション レジスタは、ファームウェア デバッグを実行するために使用されます。以下にレジスタとその主な機能を記載します。これらレジスタのビット レベルの定義の詳細はArmv6-M アーキテクチャ リファレンス マニュアル を参照してください。

- デバッグ ホールト制御およびステータス レジスタ (CM0P_DHCSR) – このレジスタは、デバッグを有効にし、CPU を一時停止し、そしてシングルステップ動作を実行するための制御ビットを含んでいます。さらにプロセッサのデバッグ状態を示すためのステータス ビットも含みます。
- デバッグ フォルト ステータス レジスタ (CM0P_DFSR) このレジスタは、デバッグ イベントが発生した理由を説明します。これは CPU の一時停止、ブレークポイント イベントまたはウォッチポイント イベントに起因したデバッグ イベントを含んでいます。
- デバッグ コア レジスタ セレクタ レジスタ (CM0P_DCRSR) – このレジスタは、Cortex-M0 + CPU で汎用レジスタを選択するために使用されます。Cortex-M0 CPU では、読み出し動作または書き込み動作が外部デバッガによって実行される必要があります。
- デバッグ コア レジスタ データ レジスタ (CM0P_DCRDR) – このレジスタは、CM0P_DCRSR レジスタで選択したレジスタに転送するデータを格納するために使用されます。
- デバッグ例外およびモニター制御レジスタ (CM0P_DEMCR) – このレジスタは、グローバル デバッグ ウォッチポイント (DWT) ブロック イネーブル、リセット ベクタ キャッチおよびハード フォルト例外キャッチに対応したイネーブル ビットを含んでいます。

19.6.2 ブレークポイント ユニット (BPU)

BPU は命令取り出し段階でブレークポイント機能を提供します。PSoC 4 での Cortex-M0+ DAP は、最大 4 点のハードウェア ブレークポイントをサポートしています。ハードウェア ブレークポイントに加え、Cortex-M0+ の BKPT 命令を使用することによりソフトウェア ブレークポイントを任意の数作成することができます。BPU には 2 種類のレジスタがあります。

- ブレークポイント制御レジスタ (CM0P_BP_CTRL) は BPU を有効にし、デバッグ システムが対応するハードウェア ブレークポイントの数 (PSoC 4 での CM0 DAP の場合は 4 点) を保持します。
- それぞれのハードウェア ブレークポイントには1つのブレークポイント比較レジスタ (CM0P_BP_COMPx) があります。ブレークポイント比較レジスタはブレークポイントのイネーブル ビット、比較アドレス値およびブレークポイント デバッグ イベントのトリガ条件を含んでいます。一般的な使用例としては、命令取り出しアドレスがブレークポイントの比較アドレスと一致する場合、ブレークポイント イベントが生成され、プロセッサが一時停止します。

19.6.3 データ ウォッチポイント (DWT)

DWT はデータ アドレス アクセスまたはプログラム カウンター (PC) 命令アドレスでウォッチポイント サポートを提供しています。DWT は 2 ウォッチポイントをサポートします。また PC サンプル レジスタを使用して外部プログラム カウンター サンプリングも提供しています。このサンプル レジスタは、プログラム カウンターの非侵襲性のおおまかなプロファイリングに使用されます。以下は DWT における最も重要なレジスタです。

- ウォッチポイント比較 (CM0P_DWT_COMPx) レジスタは、ウォッチポイント イベント生成用のウォッチポイント コンパレータによって使用される比較値を格納します。各ウォッチポイントに対して関連する DWT_COMPx レジスタを持っています。
- ウォッチポイント マスク (CM0P_DWT_MASKx) レジスタは、関連するウォッチポイントに一致するアドレス範囲に適用される除外マスクを格納します。
- ウォッチポイント機能 (CM0P_DWT_FUNCTIONx) レジスタは、ウォッチポイント イベントのトリガ条件を格納します。イベントはプログラム カウンター ウォッチポイント イベントまたはデータ アドレス読み書きアクセス ウォッチポイント イベントです。また関連する

ウォッチポイント イベントが発生するとステータス ビットを設定します。

- ウォッチポイント コンパレータ PC サンプル レジスタ (CM0P_DWT_PCSR) は、プログラム カウンターの現行値を格納します。このレジスタはプログラム カウンター レジスタの非侵襲性のおおまかなプロファイリングに使用されます。

19.6.4 PSoC 4 デバイスのデバッグ

ホストは、デバッグ制御およびコンフィギュレーションレジスタ、BPU レジスタおよび DWT レジスタにアクセスすることにより、ターゲットの PSoC 4 をデバッグします。すべてのレジスタはSWD インターフェースを介してアクセスされます。Cortex-M0+ DAP の SWD デバッグ ポート (SW-DP) は、DAP-AHB インターフェース経由で SWD パケットを適切なレジスタ アクセスに変換します。

PSoC 4 をデバッグする最初の手順は SWD ポートを開通させることです。開通シーケンスは、SWD ライン リセット シーケンスおよび SWD インターフェースを介した DAP SWDID 読み出しから成り立っています。正しい CM0 DAP SWDID がターゲット デバイスから読み出された時、SWD ポートが開通します。SWD インターフェースでデバッグを行う場合、対応するピンを他の目的に使用しないでください。SWD ポート ピンを SWD インターフェース専用に使用する、または GPIO 等のような他の機能として使用する場合のコンフィギュレーションについては 37 ページの I/O システムの章 を参照してください。デバッグが必要とされた場合は、SWD ポート ピンをそれ以外の目的に使用しないでください。プログラム サポートのみが必要な場合は、SWD ピンを他の目的にも使用できます。

SWD ポートが開通した時、外部デバッガはデバッグを有効にするために、DHCSR レジスタでの C_DEBUGEN ビットを設定します。そして、デバッグ システムに適切なレジスタを書き込むことで、ステップ実行、ホールト、ブレークポイント コンフィギュレーションおよびウォッチポイント コンフィギュレーション等といった異なるデバッグ動作を実行します。

ターゲット デバイスのデバッグは 81 ページのデバイス セキュリティの章 に説明されているデバイス全体の保護設定によっても影響を受けます。OPEN 保護モードのみがデバイス デバッグに対応します。デバイスがアクティブ モードからディープスリープ モードまたはスリープ モードのいずれかに移行する場合、外部デバッガとターゲット デバイスの接続は失われません。デバイスがディープスリープ モードまたはスリープ モードからアクティブ モードに移行した場合、デバッガは接続シーケンスを再び初期化することなく、その動作を再開することができます。

19.7 レジスタ

表 19-5. レジスター覧

レジスタ名	説明
CM0P_DHCSR	デバッグ ホールト制御およびステータス レジスタ
CM0P_DFSR	デバッグ フォルト ステータス レジスタ
CM0P_DCRSR	デバッグ コア レジスタ セレクタ レジスタ
CM0P_DCRDR	デバッグ コア レジスタ データ レジスタ
CM0P_DEMCR	デバッグ例外およびモニター制御レジスタ
CM0P_BP_CTRL	ブレークポイント制御レジスタ
CM0P_BP_COMPx	ブレークポイント比較レジスタ
CM0P_DWT_COMPx	ウォッチポイント比較レジスタ
CM0P_DWT_MASKx	ウォッチポイント マスク レジスタ
CM0P_DWT_FUNCTIONx	ウォッチポイント機能レジスタ
CM0P_DWT_PCSR	ウォッチポイント コンパレータ PC サンプル レジスタ

20. 不揮発性メモリ プログラム

不揮発性メモリ プログラムの詳細については PSoC® 4 デバイスのフラッシュ メモリのプログラムを参照してください。この章はデバイス プログラムを実行する消去、書き込み、プログラムおよびチェックサム計算の関数について説明します。サイプレスが提供するプログラマおよびサードパーティーのプログラマは、これらの関数を用いて、hex ファイルのアプリケーションデータで PSoC 4 デバイスをプログラムすることができます。CPU がフラッシュ メモリの一部を更新するブートロード処理の実行にも使用することが可能です。

20.1 特長

- デバッグアクセス ポート (DAP) および Cortex-M0+ CPU を介するプログラムに対応します
- ブロッキングと非ブロッキングのフラッシュ プログラムおよび Cortex-M0+ CPU からの消去動作の両方に対応

20.2 機能説明

フラッシュ プログラム動作はシステム コールとして搭載されています。システムコールは特権モードで SROM から実行されます。ユーザーは SROM コードを読み込んだり、変更したりすることはできません。DAP または CM0+ CPU は、コントローラー インターフェース (SPCIF) 入力レジスタに関数オペコードと関数パラメーターを書き込んで、SROM が関数を実行するよう要求することで、システム コールの要求を行います。関数オペコードに基づいて、システム性能コントローラー (SPC) は SROM から対応するシステム コールを実行して、SPCIF ステータス レジスタを更新します。DAP または CPU は関数実行の成否を取得するためにこのステータス レジスタを読み出す必要があります。関数実行の一部である SROM のコードは、SPCIF とやり取りすることにより、実際のフラッシュ プログラム動作を実行します。

PSoC 4 フラッシュはプログラム消去のプログラム (PEP) シーケンスによりプログラムされます。すべてのフラッシュ セルは既知の状態にプログラムされ、消去されてから、選択したビットがプログラムされます。このシーケンスは蓄積電荷のバランスを取ることで、フラッシュの寿命を長期化します。フラッシュに書き込む時には、データは最初にページ ラッチ バッファにコピーされます。フラッシュ書き込み関数は後程このデータをフラッシュに転送するために使用されます。

外部のプログラマは、コマンドのデバッグアクセス ポート (DAP) への送信と SWD プロトコルの使用により PSoc 4 のフラッシュ メモリをプログラムします。外部プログラマが行う PSoC 4 デバイスのプログラム シーケンスは PSoC 4100M、PSoC 4200M、PSoC 4200D、PSoC 4400、PSoC 4000S、PSoC 4100S、PSoC 4700S のプログラム仕様に掲載されています。CM0+ CPU は AHB インターフェースを介した関連レジスタにアクセスすることで、フラッシュ メモリをプログラムすることができます。このようなプログラムは、通常ブートロード動作またはフラッシュ メモリに格納されたルックアップテーブルの更新などのその他のアプリケーション要求の一部として、フラッシュ メモリの一部の更新に使用されます。DAP からフラッシュ メモリへの書き込み動作または CPU からフラッシュ メモリへの書き込み動作はすべて SPCIF により完了します。

注 : フラッシュ メモリへの書き込みは最大 20 ミリ秒かかることがあります。この期間中のデバイスのリセットは禁止です。予期しない変更がフラッシュに発生する可能性があります。リセット ソース (78 ページのリセット システムの章を参照してください) は、XRES ピン、ソフトウェア リセットおよびウォッチドッグを含みます。これらが誤ってアクティブにならないようご確認ください。また低電圧検出回路はリセットではなく割り込みが発生するように設定される必要があります。

20.3 システム コールの搭載

システム コールは以下のものから構成されます

- オペコード : 一意の 8 ビット オペコード
- パラメーター: 2つの8ビットのパラメーターはすべてのシステム コールに必須のものです。これらのパラメーターは key1 と key2 と呼ばれ、以下のように定義されます :

  key1 = 0xB6

  key2 = 0xD3 + オペコード

  2 つのキーをパスすることでユーザー システム コールが誤って開始されないように確認します。key1 と key2 のパラメーターが正しくない場合、SROM は関数を実行せずエラー コードを戻します。この 2 つのパラメーターと別に、呼び出される特定の関数によって追加のパラメーターを必要とすることがあります。
- 戻り値 : いくつかのシステム コールは実行完了時に、シリコン ID あるいはチェックサムなどの値を戻します。
- 完了ステータス: 各システム コールは32ビットのステータスを戻し、CPU あるいは DAP はそれを読み込み成否を検証します。

20.4 ブロッキングと非ブロッキングのシステム コール

システム コールの関数は実行の性質に基づいて、ブロッキングまたは非ブロッキングとして分類されることが可能です。CPU がシステム コールの実行以外に同時にその他のタスクを実行できない時、ブロッキング システム コールが呼び出されます。ブロッキング システム コールがプロセスから呼び出される時、CPU は SROM の対応するコードにジャンプします。実行が完了すると、オリジナル スレッドの実行は再開します。非ブロッキング システム コールの使用により、CPU はその他のいくつかのコードを同時に実行することができます。非ブロッキング システム コールは割り込みを介してCPUに中間システム コールのタスク完了を通信することができます。

CPU がシステム コールを開始する場合のみに、非ブロッキング システム コールは使用されます。DAP はプログラム モードの中のみにシステム コールを使用します。そのプロセスでは CPU が停止します。

非ブロッキング システム コールは非ブロッキング列書き込み、非ブロッキング列プログラムとレジューム 非ブロッキングの 3 つがあります。その他のシステム コールはすべてがブロッキング システム コールです。

CPU がフラッシュで消去あるいはプログラム動作を実行する時にフラッシュからのコードを実行できないため、SRAM から実行されるコードからのみ非ブロッキング システム コールを呼び出すことができます。非ブロッキング関数がフラッシュ メモリから呼び出されれば、フラッシュ フェッチの動作が完了する時に結果は未定義で、バス エラーを戻し、ハード フォルトをトリガーします。

システム性能コントローラー (SPC) は、フラッシュ メモリの消去およびプログラム動作の実行に必要な順序で正しい高電圧のパルスを生成するブロックです。非ブロッキング関数が SRAM から呼び出される場合、SPC タイマーは書き込みまたはプログラム動作の各サブ動作が完了する時に、自らの割り込みをトリガーします。SPC 割り込みサービス ルーチン (ISR) からレジューム非ブロッキング関数を呼び出すことにより、システム コールの後続ステップが完了したかを確認します。CPU が非ブロッキング書き込みまたは非ブロッキング プログラム動作の完了時に SRAM からのコードのみを実行できるため、SPC ISR は SRAM に位置する必要があります。SPC 割り込みは、非ブロッキング プログラムの関数の場合に 1 回、非ブロッキング書き込み動作の場合に 3 回トリガーされます。SPC ISR で完了するレジューム非ブロッキングの関数コールは、非ブロッキング プログラム動作の場合に 1 回、非ブロッキング書き込み動作の場合に 3 回呼び出されます。

非ブロッキング書き込みシステム コールの使用と SRAM からのユーザー コードを実行する疑似コードについては後に説明します。

20.4.1 システム コールの実行

システム コールの開始手順は以下の通りです :

1. 関数パラメーターの設定 : 関数パラメーター (key1、key2、追加のパラメーター) を準備する方法は 2 つあり、詳細は以下の通りです :
   a. CPUSS_SYSARG レジスタへの関数パラメーターの書き込み : この方法は、CPUSS_SYSARG レジスタからのパラメーターを回収する関数に適用されます。32 ビットの CPUSS_SYSARG レジスタは、個別のシステム コール テーブルで特定されたシーケンスのパラメーターで書き込む必要があります。
   b. SRAM への関数パラメーターの書き込み : この方法は、SRAM からのパラメーターを回収する関数に適用されます。パラメーターは連続的な SRAM 位置への特定のシーケンスに最初に書き込みます。それから、最初のパラメーターのアドレスである SRAM の開始アドレスはCPUSS_SYSARGレジスタに書き込みます。この開始アドレスは、いつもワードアライン (32 ビット) のアドレスです。システム コールはこのアドレスを使用して、パラメーターをフェッチします。
2. オペコードによるシステム コールの指定およびシステム コールの開始: 8ビットのオペコードはCPUSS_SYSREQ レジスタの SYSCALL_COMMAND ビット ([15:0]) に書き込まれます。オペコードは下位の 8 ビット [7:0] に位置し、0x00 は上位の 8 ビット [15:8] に書き込みます。システム コールを開始するために CPUSS_SYSREG レジスタの SYSCALL_REQ ビット (31) をセットします。このビットを立てることで、オペコードパラメーターが

参照する SROM コードに CPU がジャンプするマスク不可能割り込みをトリガーします。

3. システム コールの実行完了の待機：システム コールは実行を開始する時に、CPUSS_SYSREQ レジスタの PRIVILEGED ビットをセットします。このビットは CPU または DAP ではなく、システム コールのみによりセットされます。DAP は CPUSS_SYSREG レジスタの PRIVILEGED と SYSCALL_REQ ビットを継続的にポーリングし、システム コールが完了したかどうかを確認します。システム コールの完了時にこれらのビットはクリアされます。最大の実行時間は 1 秒です。この 2 つのビットが 1 秒後にクリアされなければ動作が失敗と判断され、次のステップを実行せず中止します。DAP と違って、CPU アプリケーション コードは、システム コールの実行中にこれらのビットをポーリングできないことにご注意ください。それは CPU がシステム コールの実行中に SROM からのコードを実行するからです。アプリケーション コードは、実行が SROM から返った後で、最後の関数の成功／失敗ステータスのみ確認できます。
4. 完了ステータスの確認： システム コールが完了して、PRIVILEGED と SYSCALL_REQ ビットがクリアになった後、CPUSS_SYSARG レジスタを読み込んで、システム コールのステータスを確認します。CPUSS_SYSARG レジスタから読み出された 32 ビット値が 0xAXXXXXXX (「X」がドント ケア hex 値を示します) である場合、システム コールの実行は成功です。システム コールの実行が失敗した場合、ステータス コードは 0xF00000YY であり、その中で YY は失敗の原因を示します。ステータス コードの一覧表とその説明については 表 20-1 を参照してください。
5. 戻り値の回収: シリコン ID やチェックサムなどの値を戻すシステム コールの場合、CPU または DAP は CPUSS_SYSREG と CPUSS_SYSARG のレジスタを読み出して戻り値をフェッチします。

20.5 システム コール

表 20-1 は PSoC 4 が対応するすべてのシステム コールを説明とデバイスの保護モードの区分と共に示した一覧です。デバイスの保護モードの設定の詳細については 81 ページのデバイス セキュリティの章を参照してください。表に示すように、CPU が呼び出すことができないシステム コールがあることにご注意ください。各システム コールの詳細情報については表を参照してください。

表 20-1. システム コール一覧

システム コール	説明	DAP アクセス			CPU アクセス
		オープン	保護	キル	
シリコン ID	デバイス シリコン ID、ファミリ ID およびリビジョン ID を戻します	✔	✔	–	✔
ロード フラッシュ バイト	フラッシュ列にプログラムするデータをページ ラッチ バッファにロードします。1 バイトの単位で最大列サイズ 128 バイトまでです。	✔	–	–	✔
列書き込み	1 列のフラッシュを消去してからページ ラッチ バッファ内のデータでプログラム	✔	–	–	✔
列プログラム	ページ ラッチ バッファ内のデータで 1 列のフラッシュをプログラム	✔	–	–	✔
全消去	フラッシュ アレイのすべてのユーザー コード、監視フラッシュ領域にあるフラッシュ列レベル保護データを消去	✔	–	–	
チェックサム	フラッシュ メモリ全体 (ユーザーおよび監視領域) または指定したフラッシュの 1 列についてチェックサムを計算	✔	✔	–	✔
書き込み保護	フラッシュ 列単位の保護設定とチップ単位の保護設定の両方を監視フラッシュ (行 0) にプログラム	✔	✔	–	
非ブロッキング列書き込み	1 列のフラッシュを消去してからページ ラッチ バッファ内のデータでプログラム。プログラムまたは消去中にユーザーは SRAM からのコードを実行することが可能。この関数は CPU アクセスのみに使用	–	–	–	✔
非ブロッキング列プログラム	ページ ラッチ バッファ内のデータで 1 列のフラッシュをプログラム。プログラムまたは消去中にユーザーは SRAM からのコードを実行することが可能。この関数は CPU アクセスのみに使用	–	–	–	✔
レジューム非ブロッキング	非ブロッキング列書き込みまたは非ブロッキング 列プログラムを再開。この関数は CPU アクセスのみに使用	–	–	–	✔

20.5.1 シリコン ID

この関数 は 12 ビットのファミリ ID、16 ビットのシリコン ID、8 ビットのリビジョン ID および現在のデバイスの保護モードを戻します。これらの値は CPUSS_SYSARG と CPUSS_SYSREQ レジスタに戻されます。パラメーターは CPUSS_SYSARG と CPUSS_SYSREQ のレジスタに渡されます。

パラメーター

アドレス	書き込まれる値	説明
CPUSS_SYSARG レジスタ		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xD3	Key2
ビット [31:16]	0x0000	未使用
CPUSS_SYSREQ レジスタ		
ビット [15:8]	0x0000	シリコン ID オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [7:0]	シリコン ID Lo	
ビット [15:8]	シリコン ID Hi	2500～25FF
ビット [19:16]	マイナー リビジョン Id	これらの値については、PSoC 4100M, PSoC 4200M, PSoC 4200D, PSoC 4400, PSoC 4000S, PSoC 4100S, PSoC 4700S Programming Specifications を参照してください
ビット [23:20]	メジャー リビジョン Id	
ビット [27:24]	0xXX	未使用 (ドント ケア)
ビット [31:28]	0xA	成功ステータス コード
CPUSS_SYSREQ レジスタ		
ビット [11:0]	ファミリ ID	PSoC 4000S のファミリ ID は 0xA9 です
ビット [15:12]	チップ保護	詳細については 81 ページのデバイス セキュリティの章
ビット [31:16]	0xXXXX	未使用

20.5.2 クロックの設定

この関数はフラッシュ プログラミングおよび消去動作に必要なクロックを初期化します。この API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) がフラッシュ書き込みとフラッシュ消去 API の呼び出しのために、48MHz で IMO に設定されていることを確認するために使用されます。IMO はチャージ ポンプ クロックのソースであり 48MHz でない場合は、フラッシュ書き込みと消去 API はフラッシュに作用せずに終了し、「無効ポンプ クロック周波数」ステータスを戻します。

20.5.3 ロード フラッシュ バイト

この関数 は、1 列のフラッシュにプログラムされるデータで、ページ ラッチ バッファをロードします。ロード サイズの範囲は、1 バイトからフラッシュ 1 列の最大バイト数 128 バイトです。ページ ラッチ バッファにロードされるデータは「Byte Addr」の入力パラメーターにより指定される位置から開始します。ページ ラッチ バッファにロードされるデータは、ページ ラッチのコンテンツをクリアするプログラム動作の実行時まで残ります。ページ ラッチ バッファにロードされるデータを含み、この関数用のパラメーターは SRAM に、SRAM データの開始アドレスは CPUSS_SYSARG レジスタに書き込まれます。開始パラメーター アドレスはワードアラインのアドレスであることにご注意ください。

パラメーター

アドレス	書き込まれる値	説明
SRAM アドレス - 32'hYY (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xD7	Key2
ビット [23:16]	バイト アドレス	データの書き込み用のページ ラッチ バッファの開始アドレス 0x00 – ラッチ バッファのバイト 0 0x7F – ラッチ バッファのバイト 127
ビット [31:24]	フラッシュ マクロの選択	0x00 – フラッシュ マクロ 0 0x01 – フラッシュ マクロ 1 (デバイスのフラッシュ マクロの番号について 22 ページの Cortex-M0+ CPU の章を参照してください)
SRAM アドレス - 32'hYY + 0x04		
ビット [7:0]	ロード サイズ	ページ ラッチ バッファに書き込まれるバイト数 0x00 – 1 バイト 0x7F – 128 バイト
ビット [15:8]	0xXX	ドント ケア パラメーター
ビット [23:16]	0xXX	ドント ケア パラメーター
ビット [31:24]	0xXX	ドント ケア パラメーター
SRAM アドレス - (32'hYY + 0x08) ～ (32'hYY + 0x08 + ロード サイズ)		
バイト 0	データ バイト [0]	ロードされる最初のデータ
.	.	.
.	.	.
バイト (ロード サイズ –1)	データ バイト [ロード サイズ –1]	ロードされる最後のデータ
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0004	フラッシュ バイト オペコードをロード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.5.4 列書き込み

この関数は、ページ ラッチ バッファ内のデータで 1 列のフラッシュを消去してから、プログラムします。ページ ラッチ バッファのすべてのデータが 0 になると、プログラムはスキップされます。この関数用のパラメーターは SRAM に保存されます。保存されたパラメーターの開始アドレスは CPUSS_SYSARG レジスタに書き込まれます。この関数は、列がプログラムされた後で、ページ ラッチ コンテンツをクリアします。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前にロードフラッシュ バイト関数を呼び出します。この関数は対応するフラッシュ列が書き込み保護されていない場合にのみ、書き込み動作を行うことができます。

詳細情報については「 PSoC 4000S Family: PSoC 4 Registers TRM, PSoC 4700S Family: PSoC 4 Registers TRM」の CLK_IMO_CONFIG レジスタを参照してください。

パラメーター

アドレス	書き込まれる値	説明
SRAM アドレス - 32'hYY (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xD8	Key2
ビット [31:16]	列 ID	書き込む列番号 0x0000 – 列 0
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードアライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0005	列書き込みオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.5.5 列プログラム

この関数は、ページ ラッチ バッファ内のデータで、フラッシュのアドレス付けされた列をプログラムします。ページ ラッチ バッファのすべてのデータが 0 になると、プログラムはスキップされます。この関数を呼び出す前に、列が消去状態である必要があります。列がプログラムされた後でページ ラッチバッファの内容をクリアします。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前にロード フラッシュ バイト関数を呼び出します。この関数を呼び出す前に、列が消去状態である必要があります。この関数は対応するフラッシュ列が書き込み保護されていない場合にのみ、プログラム動作を行うことができます。

パラメーター

アドレス	書き込まれる値	説明
SRAM アドレス - 32'hYY (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xD9	Key2
ビット [31:16]	列 ID	プログラムする列番号 0x0000 – 列 0
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードアライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0006	列プログラムオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.5.6 全消去

この関数は、フラッシュの主アレイのすべてのユーザー コード、各フラッシュ マクロのスーパバイザ フラッシュの列 0 の低レベル保護データを消去します。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。チップの保護モードが OPEN であり DAP からのプログラム モードの場合のみに、この API は呼び出すことができます。チップの保護モードが PROTECTED である場合、保護の設定を OPEN に変更するために、DAP は書き込み保護 API を使用する必要があります。保護設定を PROTECTED から OPEN に変更すると自動的に全消去が行われます。

パラメーター

アドレス	書き込まれる値	説明
SRAM アドレス - 32'hYY (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xDD	Key2
ビット [31:16]	0xXXXX	ドント ケア
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードアライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x000A	すべてのオペコードを消去
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.5.7 チェックサム

この関数は、フラッシュ メモリの全体とフラッシュの 1 列のどちらかを読み出し、そのフラッシュ領域で読み出される各バイトの合計の 24 ビットを戻します。フラッシュ全体のチェックサムが計算される場合、ユーザー コードおよびスーパバイザ フラッシュ領域が含まれます。フラッシュ 1 列のチェックサムが計算される場合、フラッシュの列数がパラメーターとして渡されます。パラメーターのバイト 2 とバイト 3 は、チェックサムがフラッシュ メモリの全体又はユーザー コード フラッシュのどこで実行されるかを選択します。

パラメーター

アドレス	書き込まれる値	説明
CPUSS_SYSARG レジスタ		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xDE	Key2
ビット [31:16]	列 ID	チェックサム演算が完了したフラッシュ列の番号を選択 列の番号 – 16 ビットのフラッシュ列の番号 または 0x8000 – フラッシュ メモリ全体のチェックサムが計算されます
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x000B	チェックサム オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:24]	0xX	未使用 (ドント ケア)
ビット [23:0]	チェックサム	選択したフラッシュ領域の 24 ビット チェックサム値

20.5.8 書き込み保護

この関数は、フラッシュ列レベル保護の設定とデバイスの保護の設定の両方をスーパバイザ フラッシュ列でプログラムします。フラッシュ列レベル保護の設定は、デバイスの各フラッシュ マクロに個別にプログラムされます。各列には 1 つの保護ビットがあります。保護バイトの総数は、8 で除算されたフラッシュ列の数です。チップレベル保護の設定 (1 バイト) は、スーパバイザ フラッシュの列 0 に位置する最後のバイトのフラッシュ マクロ ゼロに保存されます。スーパバイザ フラッシュ列のサイズはユーザー コード フラッシュ列のサイズと同じです。

注意事項 : この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。ロード フラッシュ バイト関数は、フラッシュ マクロのフラッシュ保護バイトをマクロに対応したページ ラッチ バッファにロードするために使用されます。ロード関数用の開始アドレス パラメーターは 0 にする必要があります。フラッシュ マクロの数はプログラムの対象となる数でなければならず、ロードするバイト数はそのマクロのフラッシュ保護バイトの数でなければなりません。

次に書き込み保護関数が呼び出され、ページ ラッチ バッファから対応するフラッシュ マクロの監視列にフラッシュ保護バイトがプログラムされます。デバイス保護の設定も保存するフラッシュ マクロ ゼロでは、デバイス レベル保護の設定は CPUSS_SYSARG レジスタのパラメーターとして渡されます。

パラメーター

アドレス	書き込まれる値	説明
CPUSS_SYSARG レジスタ		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xE0	Key2
ビット [23:16]	デバイス保護バイト	フラッシュ マクロ 0 のみに適用可能なパラメーター 0x01 – OPEN モード 0x02 – PROTECTED モード 0x04 – KILL モード
ビット [31:24]	フラッシュ マクロの選択	0x00 – フラッシュ マクロ 0 0x01 – フラッシュ マクロ 1
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x000D	書き込み保護オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:24]	0xX	未使用 (ドント ケア)
ビット [23:20]	0x000000	

20.5.9 非ブロッキング列書き込み

この関数は、フラッシュ列が非ブロッキング方式で CM0+ CPU に書き込まれる必要時に使用され、CPU は書き込み動作が継続中でも SRAM からのコードを実行することができます。非ブロッキング システム コールの説明については 183 ページのブロッキングと非ブロッキングのシステム コール を参照してください。

非ブロッキング列書き込みのシステム コールには、プログラム前、消去、プログラムの 3 段階があります。プログラム前の段階では、消去動作の準備のためフラッシュ列のすべてのビットに 1 が書き込まれます。消去動作の段階では列のすべてのビットが消去され、プログラム動作の段階では新しいデータを列に書き込みます。

各段階の実行中、CPU は SRAM からのコードを実行できます。非ブロッキング列書き込みのシステム コールが開始されると、ユーザーは非ブロッキング書き込み動作の完了に必要なレジューム 非ブロッキング関数以外のシステム コールを呼び出すことができません。各段階が完了した後、SPC は自らの割り込みをトリガーします。この割り込みはレジューム非ブロッキング システム コールを呼び出します。

注意事項: デバイス ファームウェアは非ブロッキング列書き込みの間デバイスをスリープ モードにしてはなりません。この動作は、ページ ラッチ バッファをリセットし、フラッシュはすべてゼロが書き込まれます。

注意事項 : この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前に、列プログラムするデータ バイトをロードするためロード フラッシュ バイト関数を呼び出します。また非ブロッキング列書き込み関数は SRAM のみから呼び出すことが可能です。それは CM0+ CPU がフラッシュ消去プログラム動作を行う時に、フラッシュからのコードを実行できないからです。この関数がフラッシュ メモリから呼び出されると、フラッシュ フェッチの動作が完了する時点で未定義の結果になり、バス エラーを返し、ハード フォルトをトリガーします。

パラメーター

アドレス	書き込まれる値	説明
32'hYY の SRAM アドレス (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xDA	Key2
ビット [31:16]	列 ID	書き込む列番号 0x0000 – 列 0
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードアライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0007	非ブロッキング列書き込みオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.5.10 非ブロッキング列プログラム

この関数はフラッシュ列が非ブロッキング方式で CM0+ CPU にプログラムされる必要時に使用され、CPU はプログラム動作が継続中でもSRAMからのコードを実行することができます。非ブロッキング システム コールの説明については183ページのブロッキングと非ブロッキングのシステム コール を参照してください。プログラム動作が継続中でも CPU は SRAM からのコードを実行することができます。非ブロッキング列プログラムのシステム コールが呼び出される時、ユーザーは非ブロッキング書き込み動作の完了に必要なレジューム非ブロッキング関数の以外のシステム コール関数を呼び出すことができません。

非ブロッキング列書き込みのシステム コールと異なり、プログラム システム コールには単一の段階しかありません。従ってレジューム非ブロッキング関数は、非ブロッキング プログラム システム コールの使用時に、SPC 割り込みから 1 回だけ呼び出す必要があります。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前に、列プログラムするデータ バイトをロードするためロード フラッシュ バイト関数を呼び出します。また、非ブロッキング列プログラム関数は SRAM のみから呼び出すことが可能です。それは CM0+ CPU がフラッシュ プログラム動作を行う時に、フラッシュからのコードを実行できないからです。この関数がフラッシュ メモリから呼び出されると、フラッシュ フェッチの動作が完了する時点で未定義の結果となり、バス エラーを戻し、ハード フォルトをトリガーします。

パラメーター

アドレス	書き込まれる値	説明
32'hYY の SRAM アドレス (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xDB	Key2
ビット [31:16]	列 ID	書き込む列番号 0x0000 – 列 0
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードアライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0008	非ブロッキング 列プログラム オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.5.11 レジューム非ブロッキング

この関数は非ブロッキング列書き込みと非ブロッキング列プログラムのシステム コールにより開始される消去とプログラムの追加段階を完了します。この関数は非ブロッキング列書き込みの呼び出しの後に 3 回、非ブロッキング 列プログラムの呼び出しの後に 1 回呼び出す必要があります。プログラム動作または消去動作のすべての段階が完了する前に、その他のシステム コールは実行できません。非ブロッキング関数の使用手順の詳細情報は 183 ページのブロッキングと非ブロッキングのシステム コール を参照してください。

パラメーター

アドレス	書き込まれる値	説明
32'hYY の SRAM アドレス (32 ビット幅、ワードアラインの SRAM アドレス)		
ビット [7:0]	0xB6	Key1
ビット [15:8]	0xDC	Key2
ビット [31:16]	0xXXXX	ドント ケア SROM には使用されない
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメーター (key1) を保存する SRAM の 32 ビットのワードアライン アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0009	レジューム非ブロッキング オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xXXXXXXX	未使用 (ドント ケア)

20.6 システム コール ステータス

それぞれのシステム コールを実行すると、ステータス コードが CPUSS_SYSARG レジスタの引数に上書きされます。成功のステータスは 0xAXXXXXXX であり、その中の X はシステム コールが値を戻す場合に「ドント ケア」値または戻り値を示します。失敗のステータスは 0xF00000XX と示され、その中の XX が失敗コードです。

表 20-2. システム コール ステータス コード

ステータス コード (CPUSS_SYSARG レジスタ内の 32 ビット値)	説明
AXXXXXXXh	成功 : 「X」は、API が CPUSS_SYSARG レジスタに直接パラメータを戻さない限り、SROM によって戻された「0」の値をもつ「ドント ケア」値を示します。
F0000001h	無効なチップ保護モード : この API は現在のチップ保護モード中は使用不可
F0000003h	無効なページ ラッチ アドレス : ページ ラッチ バッファ内のアドレスは範囲外にあるかまたは指定したサイズがページ アドレスより過大
F0000004h	無効なアドレス : 指定した列 ID またはバイト アドレスは利用可能なメモリ範囲外
F0000005h	保護された列 : 指定した列 ID は保護されている
F0000007h	レジューム完了 : 非ブロッキング API は全て実行完了レジューム API は次の非ブロッキング API が実行されるまで呼び出すことが不可
F0000008h	ペンディング レジューム : 非ブロッキング API は開始された後、他の API が呼び出される前に「レジューム API」を呼び出して完了させる必要があります
F0000009h	システム コール実行中 : レジュームまたは非ブロッキング動作がまだ実行中 SPC ISR が次のレジューム動作を試みる前に発生する必要があります
F000000Ah	チェックサム ゼロ不良 : 計算したチェックサムはゼロではありません
F000000Bh	無効なオペコード : オペコードは有効な API オペコードではありません
F000000Ch	キー オペコード不一致 : 指定したオペコードは key1 と key2 が一致しません
F000000Eh	無効な開始アドレス : 開始アドレスが指定した終了アドレスよりも大きい
F0000012h	無効なポンプ クロック周波数 : フラッシュ書き込みまたは消去が実行される前に IMO を 48MHz、HF クロック ソースを IMO クロック ソースに設定することが必要です

20.7 非ブロッキング システム コール疑似コード

この節は、フラッシュ プログラム動作中の非ブロッキング システム コールの設定と SRAM からのコードを実行する疑似コードを示します。

```
#define REG(addr)          (*((volatile uint32 *) (addr)))
#define CM0_ISER_REG            REG( 0xE000E100 )
#define CPUSS_CONFIG_REG           REG( 0x40100000 )
#define CPUSS_SYSREQ_REG           REG( 0x40100004 )
#define CPUSS_SYSARG_REG           REG( 0x40100008 )

#define ROW_SIZE_         ()
#define ROW_SIZE        (ROW_SIZE_)

/*Variable to keep track of how many times SPC ISR is triggered */
__ram int iStatusInt = 0x00;

__flash int main(void)
{
  DoUserStuff();

  /*CM0+ interrupt enable bit for spc interrupt enable */
  CM0_ISER_REG |= 0x00000040;

  /*Set CPUSS_CONFIG.VECS_IN_RAM because SPC ISR should be in SRAM */
  CPUSS_CONFIG_REG |= 0x00000001;

  /*Call non-blocking write row API */
  NonBlockingWriteRow();

  /*End Program */
  while(1);
}
__sram void SpcIntHandler(void)
{
  /* Write key1, key2 parameters to SRAM */
  REG( 0x20000000 ) = 0x0000DCB6;

  /*Write the address of key1 to the CPUSS_SYSARG reg */
  CPUSS_SYSARG_REG = 0x20000000;

  /*Write the API opcode = 0x09 to the CPUSS_SYSREQ.COMMAND
  * register and assert the sysreq bit
  */
  CPUSS_SYSREQ_REG = 0x80000009;

  /* Number of times the ISR has triggered */
  iStatusInt ++;
}
__sram void NonBlockingWriteRow(void)
{
  int iter;

  /*Load the Flash page latch with data to write*/
  * Write key1, key2, byte address, and macro sel parameters to SRAM
  */
  REG( 0x20000000 ) = 0x0000D7B6;
```

```
  //Write load size param (128 bytes) to SRAM
  REG( 0x20000004 ) = 0x0000007F;


  for(i = 0; i < ROW_SIZE/4; i += 1)
  {
     REG( 0x20000008 + i*4 ) = 0xDADADADA;
  }

  /*Write the address of the key1 param to CPUSS_SYSARG reg*/
  CPUSS_SYSARG_REG = 0x20000000;

  /*Write the API opcode = 0x04 to CPUSS_SYSREQ.COMMAND
  * register and assert the sysreq bit
  */
  CPUSS_SYSREQ_REG = 0x80000004;

  /*Perform Non-Blocking Write Row on Row 200 as an example.
  * Write key1, key2, row id to SRAM row id = 0xC8 -> which is row 200
  */
  REG( 0x20000000 ) = 0x00C8DAB6;

  /*Write the address of the key1 param to CPUSS_SYSARG reg */
  CPUSS_SYSARG_REG = 0x20000000;

  /*Write the API opcode = 0x07 to CPUSS_SYSREQ.COMMAND
  * register and assert the sysreq bit
  */
  CPUSS_SYSREQ_REG = 0x80000007;

  /*Execute user code until iStatusInt equals 3 to signify
  * 3 SPC interrupts have happened. This should be 1 in case
  * of non-blocking program System Call
  */
  while( iStatusInt != 0x03 )
  {
     DoOtherUserStuff();
  }

  /* Get the success or failure status of System Call*/
  syscall_status = CPUSS_SYSARG_REG;
}
```

コードでは CPUSS_CONFIG レジスタに 0x01 を書き込むことで、CM0+ の例外テーブルを SRAM 内に設定しています。SRAM の例外テーブルには *SpcIntHandler()* 関数のアドレスのような SPC 割り込みのベクター アドレスが必要です。CM0+ の例外テーブルを SRAM 内に設定することの詳細は 27 ページの割り込みの章を参照してください。関数オペコードとパラメーターが異なり、iStatusInt の変数が 3 回ではなく 1 回ポールされることを除き、非ブロッキング システム コールの疑似コードも同様です。非ブロッキング プログラム システム コールでは SPC ISR は 1 回しかトリガーされません。

用語集

用語集の節では、本テクニカル リファレンス マニュアルで使用されている専門用語を説明します。用語は、本マニュアルの全体にわたって、**太字やイタリック体**で表記されています。

A

accumulator (アキュムレータ)	CPU において、中間結果を格納するレジスタ。アキュムレータがないと、各演算 (加算、減算、シフト等) の結果をメイン メモリに書き込んで読み戻す必要がある。メイン メモリへのアクセスは、算術論理演算装置 (ALU) との直接のパスを持っているアキュムレータへのアクセスより遅くなる
active high (アクティブ HIGH)	1. アサート状態を論理値「1」状態とするロジック信号 2. 2 つの状態のうち、高い電圧側を論理値「1」状態とするロジック信号
active low (アクティブ LOW)	1. アサート状態を論理値「0」状態とするロジック信号 2. 2 つの状態のうち、低い電圧側を論理値「1」状態とするロジック信号 : 反転論理
address (アドレス)	情報ユニットを格納するメモリ位置 (RAM、ROM またはレジスタ) を識別するためのラベルや番号
algorithm (アルゴリズム)	操作の繰り返しをしばしば含む、有限個の手順で数学の問題を解くための手順
ambient temperature (周囲温度)	指定領域、特に PSoC デバイスの周囲領域の温度
analog (アナログ)	***analog signals (アナログ信号)*** を参照してください
analog blocks (アナログ ブロック)	基本はプログラム可能なオペアンプ回路。SC (スイッチド キャパシタ) および CT (連続時間) ブロック。これらのブロックは相互接続して、ADC、DAC、多極フィルター、ゲイン ステージなどを提供可能
analog output (アナログ出力)	論理 1 と論理 0 だけでなく、電源電圧間のあらゆる電圧を駆動する出力
analog signals (アナログ信号)	時間的に離散 (不連続) した形で表現されるデジタル信号に対して、時間的に連続した形で表現される信号

analog-to-digital (ADC) (アナログ - デジタル変換器) アナログ信号を、振幅に対応したデジタル信号に変換するデバイス。一般的に ADC は電圧をデジタル数値に変換。デジタル - アナログ変換器 (DAC) は ADC の逆の動作を行う

AND (論理積) ブール代数を参照してください

API (アプリケーション プログラミング インターフェース) コンピューター アプリケーションと低レベルのサービスと関数 (ユーザー モジュール、ライブラリなど) 間のインターフェースからなる一連のソフトウェア ルーチン。API は、ソフトウェア アプリケーションを作成するプログラマ向けのビルディング ブロックとして機能

array (アレイ、配列) ベクターまたはリストともいわれる配列は、コンピューター プログラミングにおける最も単純なデータ構造の 1 つ。配列はサイズが等しく、データ タイプもほとんど同じデータ要素の固定数を保持。連想配列とは対照的に、個別要素は連続した整数の範囲を使用してインデックスによってアクセスされる。殆どの高位プログラミング言語は配列をデータ タイプとして内蔵。一部の配列は多次元であり、即ち、固定された数の整数 (例えば 2 つの整数のグループ) によってインデックスされる。1 次元と 2 次元配列が最も一般的。また、いくつかの一般的な形で接続したコンデンサや抵抗のグループをアレイということもある

assembly (アセンブリ) 特定プロセッサの機械語を記号化した言語。アセンブラによってアセンブリ言語を機械コードに変換。通常、アセンブリ コードの各行は 1 つの機械命令を生成するが、マクロの使用が一般的である。アセンブリ言語は低位言語と見なされるのに対して、C 言語は高位言語と見なされている

asynchronous (非同期) どのクロック信号にも関係なく直ちに認識され、作用するデータを持つ信号

attenuation (減衰) エネルギーが消散して検出器へのパス外に分散することが原因で、信号の強度が低下する (幾何学的拡大による減衰を含まない) こと。減衰は通常 dB で表される

B

bandgap reference (バンドギャップ リファレンス) V_T の正の温度係数と V_{BE} の負の温度係数を一致させ、ゼロ温度係数 (理想) のリファレンス電圧を生成する、安定したリファレンス電圧回路

bandwidth (帯域幅)
1. ヘルツを単位として計測されるメッセージまたは情報処理システムの周波数範囲
2. 増幅器 (または減衰器) に実質的なゲイン (またはロス) があるスペクトル領域の幅。より具体的に (例えば最大値の 1/2 になる点の幅として) 示されることがある

bias (バイアス)
1. リファレンス値からの意図的な偏差の値
2. リファレンス値と一連の値の平均値の間の誤差
3. デバイスを動作させるリファレンス レベルを確立するために、デバイスに適用される電気的、機械的、磁気的、その他の力 (フィールド)

bias current (バイアス電流) アンプにおいて安定的な動作を生成するために使用される一定の低レベル DC 電流。この電流は、時にはアンプの帯域幅を変えるために変更することが可能

binary (2 進数)
2 を基数として表現した記数法の名称。最も一般的な記数法は 10 を基数とした 10 進数。数値を表現するシステムにおける基数は、そのシステムの数内の特定の位置に存在できる値の数を示す。例えば、2 を基数とした 2 進数の場合、各位置には 2 つの値 (0 または 1) のいずれかを持つ。10 を基数とした 10 進数の場合、各位置には 10 つの値 (0、1、2、3、4、5、6、7、8、9) のいずれかを持つ

bit (ビット)
2 進数の 1 桁を表す。従って、1 ビットの値は「0」または「1」のいずれかになる。8 ビットのグループは 1 バイトとして解釈される。PSoC の M8CP は 8 ビット マイクロコントローラーであるため、PSoC デバイスのネイティブ データのチャンク サイズは 1 バイト。

bit rate (BR) (ビット レート)
ビットストリーム中に単位時間あたりに発生したビットの数。通常、ビット毎秒 (bps) 単位で表現

block (ブロック)
1. 単一機能を実行する機能ユニット (発振器など)
2. いくつかの機能のいずれかを実行するためにコンフィギュレーション可能な機能ユニット (デジタル PSoC ブロックやアナログ PSoC ブロックなど)

Boolean Algebra (ブール代数)
数学やコンピューター サイエンスにおいて、ブール代数またはブール束は、論理演算 (論理積の AND、論理和の OR、否定の NOT) および集合演算 (和結合の UNION 演算子、積結合の INTERSECT 演算子、COMPLEMENT 補数演算子) の「本質をとらえる」代数的構造である。またブール代数は、ブール方程式の操作方法を説明する定理も定義する。例えば、これらの定理がブール方程式を簡単にするために使用される。従って、方程式を実行するのに必要な論理素子の数を減少させる。

ブール代数の演算子は、様々な方法で表現することが可能。多くの場合、それらは単に AND、OR および NOT として記述されている。ここで述べている回路では、NAND (NOT AND)、NOR (NOT OR)、XNOR (排他的 NOT OR) および XOR (排他的 OR) も使用可能。数学者たちは通常、OR に対しては + (例 : A+B)、AND に対しては• (例 : A*B) (いくつかの場合、それらの演算は他の代数的構造における加算と乗算に類似している) を使用している。そして、否定演算の上に横棒を引くことで NOT を表現する (例 : ~A, A_, !A)

break-before-make (ブレークビフォアメーク)
スイッチブレードが、新しい接続状態 (メーク) に入る前に、一旦遮断状態 (ブレーク) を経過すること

broadcast net (ブロードキャスト ネット)
マイクロコントローラーを通って配線され、複数のブロックまたはシステムによってアクセス可能な信号

buffer (バッファ)
1. 1 つのデバイスから他のデバイスへデータを転送する際に、速度差を補うために使用されるデータ ストレージ領域。通常、データが読み書きされる I/O 操作のために予約された領域を示す
2. 外部デバイスに送信されるデータや外部デバイスから受信されたばかりのデータを格納するメモリ部分
3. システムの出力インピーダンスを下げるために使用される増幅器

bus (バス)
1. 複数ラインの名前付き接続。回路同士をバスにバンドルすることにより、類似したルーティング パターンを持つネットの配線は容易になる
2. 共通機能を実行し、同様のデータを運ぶ信号一式。一般的にベクトル表記で表される (例 : アドレス [7:0])
3. 関連するデバイスのグループの共通接続として機能する 1 つまたは複数の導電体

byte (バイト)	8 ビットから成るデジタル ストレージ ユニット

C

C	高位プログラミング言語
capacitance (静電容量)	絶縁体によって分離された 2 つの隣接する導体に、電位差が印加された時にどのくらい電荷が蓄えられるかを表す量。容量はファラッドの単位で測定される
capture (キャプチャ)	コンピューター ファイルにデータを手入力するのに対して、ソフトウェアまたはハードウェアを使用して自動的に情報を抽出すること
chaining (連結)	2 つ以上の 8 ビット デジタル ブロックを繋いで、16 ビット、24 ビット、さらに 32 ビットの関数を形成。連結により、Compare (コンペア)、Carry (キャリー)、Enable (イネーブル)、Capture (キャプチャ) および Gate (ゲート) 等のような特定信号をあるブロックから別のブロックに転送することが可能
checksum (チェックサム)	データの個々のワードの値を総計 (sum) に加算することで、データ セットのチェックサムを生成。実際のチェックサムは、単に総計結果であるか、所定値を生成するために総計に加算しなければならない値である
clear (クリア)	ビット／レジスタの値を論理「0」にする
clock (クロック)	一定の周波数およびデューティ比で周期信号を生成するデバイス。クロックは時々、異なる論理ブロックを同期化するために使用される。
clock generator (クロック ジェネレータ)	クロック信号を生成するための回路
CMOS	相補的に接続された *MOS* トランジスタを用いて構成された論理ゲート。CMOS は相補型金属酸化膜半導体の頭字語
comparator (コンパレータ)	2 つの入力レベルが同時に所定の振幅要件を満たすたびに、出力電圧または電流を生成する電子回路
compiler (コンパイラ)	C のような高位言語を機械言語に変換するプログラム
configuration (コンフィギュレーション)	コンピューター システムにおける、機能ユニットの性質、数および特性に応じた配置。コンフィギュレーションはハードウェア、ソフトウェア、ファームウェアおよび各種文書に直接関係がある。コンフィギュレーションはシステムの性能に影響を与える
configuration space (コンフィギュレーション空間)	PSoC デバイスでの、CPU_F レジスタ内の XIO ビットが「1」にセットされた時にアクセスされるレジスタ空間
crowbar (クローバ回路)	過電圧保護回路の一種。出力電圧が制限値を超えると、急速に低抵抗 (通常 SCR) を配置し、その信号から電源供給レールのいずれかに短絡
CPUSS	CPU サブシステム

crystal oscillator (水晶発振器)	周波数が圧電性水晶によって制御される発振器。一般的に、圧電性水晶は他の回路コンポーネントほど周囲温度に敏感ではない
cyclic redundancy check (CRC) (巡回冗長検査)	一般的に線形フィードバック シフト レジスタを使用して行われるデータ通信でエラーを検出するために使用される計算。同様の計算はデータ圧縮など他の多くの用途に使用可能

D

data bus (データ バス)	メモリ位置から中央演算処理装置へ、またはその逆で、情報を伝えるためにコンピュータによって使用される一組の双方向信号。より一般的には、デジタル機能間でデータを伝えるために使用される信号一式
data stream (データ ストリーム)	転送中の情報を表すために使用するデジタル符号化信号のシーケンス
data transmission (データ転送)	チャネル上の信号によってデータをある場所から別の場所へ転送すること
debugger (デバッガー)	ユーザーが開発中のシステムの動作を分析することを可能にするハードウェアおよびソフトウェア システム。通常、開発者はデバッガーにより、ファームウェアを一段階ずつ手順を追って実行したり、ブレークポイントをセットしたり、メモリを分析したりすることが可能
dead band (デッド バンド)	2 つまたは複数の信号の内いずれもアクティブ状態や遷移中でない期間
decimal (10 進法)	基数を 10 とした記数法であり、数値を表現するために、0、1、2、3、4、5、6、7、8、9 の記号 (桁と呼ばれる) を小数点および符号 + (プラス) と - (マイナス) と共に併用する
default value (デフォルト値)	ユーザーの指定なしでシステムが想定、使用、実行する既定値、初期値または特定の設定、条件、値あるいは操作を指す
device (デバイス)	特記がない限り、本マニュアルで記述されているデバイスは PSoC デバイスを指す
die (ダイ)	通常、ウェハから切断される未パッケージ集積回路 (IC)
digital (デジタル)	振幅が 2 つの離散値「0」か「1」のいずれかによって特徴付けられる信号または機能
digital blocks (デジタル ブロック)	カウンターやタイマー、シリアル レシーバー、シリアル トランスミッター、CRC ジェネレータ、疑似乱数ジェネレータ、SPI として機能する 8 ビットの論理ブロック
digital logic (デジタル ロジック)	回路またはシステム操作を表す二状態変数を含む表現に対応する技法
digital-to-analog (DAC) (デジタル - アナログ変換器)	デジタル信号を対応する振幅を持っているアナログ信号に変換するデバイス。アナログ - デジタル変換器 (ADC) は DAC の逆の動作を行う
direct access (直接アクセス)	データのシーケンスで (データ間の相対位置は互いに独立している)、アドレスを用いてデータの物理的な位置を示すことにより、記憶装置に対してデータを取得したり入力したりする機能

duty cycle (デューティ比)	クロック周期の HIGH 時間と LOW 時間の関係。パーセント単位で表される

E

External Reset (XRES_N) (外部リセット)	PSoC デバイスに駆動されるアクティブ HIGH 信号。これにより、CPU およびブロックのすべての動作が停止し、事前定義された状態に戻る

F

falling edge (立ち下がりエッジ)	論理 1 から論理 0 への遷移。ネガティブ エッジともいわれる
feedback (フィードバック)	アクティブ デバイスの出力の一部または処理された出力の一部を入力へ戻すこと
filter (フィルター)	信号の特定の周波数コンポーネントを減衰させるデバイスまたはプロセス
firmware (ファームウェア)	ハードウェア デバイスに組み込まれ、CPU によって実行されるソフトウェア。エンド ユーザーによっては実行可能であるが、変更不可
flag (フラグ)	状態またはイベントの認識に使用される多種多様なインジケータのいずれかである (例えば送信の終了を通知するための文字)
Flash (フラッシュ)	EPROM のプログラマビリティとデータ ストレージおよびインシステム消去性をユーザーに提供する、電気的にプログラマブルで消去可能な*不揮発性*の技術。不揮発性とは、電源がオフになってもデータは依然として保持されるということ
Flash bank (フラッシュ バンク)	フラッシュ ROM ブロックのグループ。個々のフラッシュ バンクにおいて、フラッシュ ブロックの番号は常に「0」で始まる。フラッシュ バンクは、独自のブロック レベルの保護情報も持っている
Flash block (フラッシュ ブロック)	一度にプログラムできるフラッシュ ROM の最小容量および保護できるフラッシュ メモリの最小領域。1 つのフラッシュ ブロックは 64 バイトを保持
flip-flop (フリップフロップ)	2 つの安定した状態および 2 つの入力端子 (または入力信号の種類) を持つデバイス。それぞれの入力端子が 2 つの状態のいずれかに対応する。回路は、該当する入力信号を適用することによって残りの状態にさせられるまで、2 つの状態のいずれかのままになる
frequency (周波数)	周期的な機能の、時間単位当たりのサイクルまたは発生するイベントの数

G

gain (ゲイン、利得)	出力電流、電圧または電力対入力電流、電圧または電力の比率。ゲインは通常 dB で表される

gate (ゲート)

1. 1 つのデバイスには、1 つの出力チャネルと 1 つ以上の入力チャネルを持っている。そのように、出力チャネルの状態は完全に入力チャネルの状態によって決定される (状態切り替えの過渡期以外)。
2. 複数の組み合わせ論理要素のいずれかは、少なくとも 2 つの入力を持つ (例えば AND、OR、NAND および NOR (ブール代数も参照してください))

ground (グランド)

1. 周囲の大地と同じ電位を有する電気中性線
2. DC 電源供給のマイナス側
3. 電気的システムのリファレンス点
4. 電気回路や機器と地面との導電路または地面と接している導電部分

H

hardware (ハードウェア)

コンピューターや組み込みシステムが操作するデータおよびハードウェアがタスクを完了させるために命令を与えるソフトウェアとは区別され、コンピューターや組み込みシステムのすべての物理的な部分を呼ぶのに用いられる包括的な用語

hardware reset (ハードウェア リセット)

回路によって発生したリセット (POR、ウォッチドッグ リセット、外部リセット等)。ハードウェア リセットは、デバイスを初めて電源投入される時の状態に復帰する。従って、すべてのレジスタは本書全体にわたるレジスタ表に記載されている POR の値に設定される

hexadecimal (16 進数)

基数を 16 とした数値の表現方法 (しばしば省略され、「hex」と呼ばれている)、通常、0 ～ 9 と A ～ F 記号で書かれている。4 ビットの数字を 16 進数の 1 桁に変換するのが簡単であるため、16 進数はコンピューターにおける有用な記数システムになる。従って、1 バイトを連続した 16 進数の 2 桁で表すことが可能。2 進数、16 進数および 10 進数の表記を比較 :

bin = hex = dec

0000b = 0x0 = 0

0001b = 0x1 = 1

0010b = 0x2 = 2

...

1001b = 0x9 = 9

1010b = 0xA = 10

1011b = 0xB = 11

...

1111b = 0xF = 15

このように、10 進数の 79 は、2 進数で 0100 1111b、16 進数で 4Fh (0x4F) として表記される

high time (HIGH 時間)

周期的デジタル信号が一定の期間中に「1」の値を有する時間

I

I²C	Philips Semiconductors 社 (現 NXP Semiconductors 社) の 2 線式シリアル コンピュータ バス。I²C はインター インテグレーテッド サーキット (内部集積回路) 。組み込みシステムの低速ペリフェラルを接続するために使用。オリジナル システムはバッテリ制御インターフェースとして 1980 年代初頭に作成された。その後、制御電子回路を構築するための単純な内部バス システムとして使用。I²C2 つの双方向のピン (クロックおよびデータ) のみを使用。双方とも +5V で動作し、抵抗を介して HIGH にプルアップされる。バスは標準モードでは 100kbps、高速モードでは 400kbps で動作
idle state (アイドル状態)	ユーザーのメッセージが送信されていないものの、サービスをいつでも利用可能な状態
impedance (インピーダンス)	1. 回路における抵抗、容量性または誘導性素子による電流の流れにくさ 2. 電流の流れに対する受動抵抗の総量。インピーダンスは所定の回路において、抵抗、誘導性リアクタンス、容量性リアクタンスの特定の組み合わせによって決定されることに注意
input (入力)	デバイス、プロセスまたはチャネルにおいて、データを読み込む箇所
input/output (I/O) (入力／出力)	システムへデータを導入したり、システムからデータを抽出するデバイス
instruction (命令)	C やアセンブリ等のプログラミング言語で実行する操作を指定して、そのオペランド (もしあれば) を指定する式
instruction mnemonics (命令のニーモニック)	アセンブリ言語の命令のそれぞれのオペコードを表す略語一式 (例 : ADD、SUBB、MOV)
integrated circuit (IC) (集積回路)	抵抗、コンデンサ、ダイオードおよびトランジスタ等のコンポーネントが、1 枚の半導体基板上に形成するデバイス
interface (インターフェース)	2 つのシステムやデバイス同士が接続し、互いに影響し合うための手段
interrupt (割り込み)	外部イベントによって引き起こされ、プロセスを再開することができるような方法で行ったプロセス (コンピュータ プログラムの実行など) の一時停止
interrupt service routine (ISR) (割り込みサービス ルーチン)	M8CP がハードウェア割り込みを受信した時に通常のコードの実行から転向させられるコード ブロック。多くの割り込みソースが、それ独自の優先順位および個別の ISR コード ブロックを持っている。各 ISR コード ブロックは RETI 命令で終了し、正常のプログラム実行を終了したポイントにデバイスを戻す

J

jitter (ジッタ)

1. 遷移の理想的な位置からのタイミング誤差。シリアル データ ストリームで発生する破損の典型的形式
2. 連続パルス、連続サイクルの振幅または連続サイクルの周波数あるいは位相間の間隔など、1 以上の信号特性の急激および不要な変動

L

latency (レイテンシ)
信号が所定の回路またはネットワークを通るのにかかる時間または遅延時間のこと

least significant bit (LSb) (最下位ビット)
最下位値 (一般的には、右側のビット) を表す 2 進数の桁またはビットのこと。ビットとバイトを区別するために、ビットの場合は小文字の「b」(LSb) を使用

least significant byte (LSB) (最下位バイト)
最下位値 (一般的には、右側のバイト) を表すマルチバイト文字内のバイト。ビットとバイトを区別するために、バイトの場合は大文字の「B」(LSB) を使用

Linear Feedback Shift Register (LFSR) (リニア フィードバック シフト レジスタ)
データ入力がレジスタ チェーンの 1 つ以上の要素間の *XOR* として生成されるシフト レジスタのこと

load (負荷)
電力 (ワット)、電流 (アンプ) または抵抗 (オーム) として表しているプロセスの電力需要

logic function (論理関数)
デジタル データを用いてデジタル演算を実行して、デジタル値を返す数学関数のこと

lookup table (LUT) (ルックアップ テーブル)
複数の論理関数を実行するためのロジック ブロック。論理関数は選択ラインで選択され、ブロックの入力に適用される。例えば : 2 入力の LUT (4 本のスレーブ選択ライン付き) は、2 つの入力で 16 つの論理機能のいずれかを実行して、単一の論理出力をもたらすために使用される。LUT は組み合わせデバイスであるため、入力と出力との関係は連続的であり、即ち、サンプリングしない

low time (LOW 時間)
周期的デジタル信号が一定の期間中に「0」の値を有する時間

low-voltage detect (LVD) (低電圧検出)
V_{DDD} を感知し、V_{DDD} が既定の閾値を下回るとシステムへ割り込みを生成する回路

M

M8CP
8 ビット ハーバード アーキテクチャ マイクロプロセッサ。マイクロプロセッサをフラッシュ、SRAM、レジスタ空間へインターフェースで接続することにより PSoC デバイスのすべての内部動作を整理

macro (マクロ)
プログラミング言語マクロは抽象的である。そこで、特定したテキスト パターンは、定義されたルールに応じて置き換えられる。マクロ インスタンスが発生した時、インタプリタまたはコンパイラは、自動的にマクロ インスタンスをマクロ コンテンツで置き換える。従って、マクロが 5 回使用され、マクロ定義が 10 バイトのコード空間を要する場合、合計 50 バイトのコード空間が必要となる

mask (マスク)
1. 情報が信号から生成されることを隠蔽、非表示、あるいは防止すること。これは通常、他の信号 (ノイズ、スタティック、ジャミングまたは干渉の他の形態など) との相互作用の結果。
2. コンピューティングやデータ処理システムにおいて、他のビット パターンのセグメントを保持または抑制するために使用されるビット パターン

master device (マスター デバイス)
2 つのデバイス間のデータ交換のタイミングを制御するデバイス。または、デバイスがいくつかカスケード接続されている場合、マスター デバイスは、カスケード接続されたデバイスと外部インターフェース間のデータ交換のタイミングを制御するもの。制御されるデバイスは*スレーブ デバイス*と呼ばれている

microcontroller (マイクロコントローラー)
主に制御システムおよび製品のために設計された集積回路デバイス。通常、マイクロコントローラーは CPU に加え、メモリやタイミング回路、I/O 回路を内蔵。理由は、最小量のデバイスを使用してコントローラーの実現を可能にするため。このようにして、最大の可能性の小型化を達成。これにより、コントローラーの寸法を低減し、コストを削減。マイクロコントローラーは通常、マイクロプロセッサとして汎用演算処理には使用されない

mnemonic (ニーモニック)
メモリを支援するように用意されるツール。ニーモニックは、物事を覚えるための繰り返しにだけでなく、覚えやすい構造とデーター覧との関係作りにも依存している。2 ～ 4 の文字列がマイクロプロセッサの命令を表す

mode (モード)
ソフトウェアやハードウェアの動作に対応する明確な方法。例えば、デジタル PSoC ブロックはカウンター モードまたはタイマー モードのいずれかである

modulation (変調)
キャリア信号、特に正弦波信号の情報をエンコードするための技術範囲。変調を行うデバイスはいわゆる変調器と呼ばれている

Modulator (変調器)
キャリア上の信号を変換するデバイス

MOS
金属酸化膜半導体の頭字語

most significant bit (MSb) (最上位ビット)
最上位値 (一般的には、左側のビット) を表す 2 進数の桁またはビットのこと。ビットとバイトを区別するために、ビットの場合は小文字の「b」(MSb) を使用

most significant byte (MSB) (最上位バイト)
最上位値 (一般的には、左側のバイト) を表すマルチバイト文字内のバイト。ビットとバイトを区別するために、バイトの場合は大文字の「B」(MSB) を使用

multiplexer (mux) (マルチプレクサ)

1. 複数の入力から選び出して、データを選択した入力から出力へ伝送するために、2 進数の値またはアドレスを使用する論理関数。
2. 外部信号によって制御され、異なる入力 (または出力) 信号が異なる時間で同じラインを使用できるようにする技術。この多重化技術は、配線と I/O ポートを節約するために使用される

N

NAND (否定論理積) ブール代数を参照してください

negative edge (ネガティブ エッジ) 論理 1 から論理 0 への遷移。立ち下がりエッジともいわれる

net (ネット) デバイス間の配線

nibble (ニブル) 4 ビットのグループ、即ち 1 バイトの半分に相当

noise (ノイズ、雑音)

1. 信号に影響を与える、およびその信号によって運ばれた情報を歪める可能性がある妨害
2. 電圧や電流、データなど実体の 1 つ以上の特性のランダムなばらつき

NOR (否定論理和) ブール代数を参照してください

NOT (論理否定) ブール代数を参照してください

O

OR (論理和) ブール代数を参照してください

oscillator (発振器) クロック周波数を生成するために使用される回路。水晶制御のものもある

output (出力) アナログ ブロックまたはデジタル ブロックによって生成された電気信号

P

parallel (パラレル) 一度に複数ビットをまとめてデジタル データを転送する通信方式。それぞれのビットは個々の信号線を用いて同時に転送される

parameter (パラメーター) 特性取りされて決められたか、設計者によって定義された所定ブロックの特性

parameter block (パラメーター ブロック) SSC 命令のパラメーターが実行前に配置されているメモリ上の番地

parity (パリティ) 送信データをテストする技術。通常、2 進データに 2 進桁数を追加して、データのすべての桁数の合計が常に偶数 (偶数パリティ) または常に奇数 (奇数パリティ) になるようにする

path (パス)	1. コンピューターによって実行される命令の論理的なシーケンス 2. 電気信号が回路を通る流れ
pending interrupts (保留中の割り込み)	トリガーされたが処理されていない割り込みのこと。原因としては、プロセッサが別の割り込みを処理中であるためかグローバル割り込みが無効になっているため
phase (位相)	(通常、周波数が同じ) 2 つの信号間の遅延時間を決定するそれらの関係。信号間の遅延は時間または角度 (度) によって測定される
pin (ピン)	ハードウェア コンポーネントの端子。リードとも呼ばれている
pinouts (ピン配置)	ピン番号割り当て : PSoC デバイスの論理入力および出力とそれらのプリント回路基板 (PCB) パッケージ内の物理的な対応関係。ピン配置は回路図と PCB 設計 (両方ともコンピューター生成ファイル) 間のリンクとしてのピン番号を含み、ピン名も含む場合がある
port (ポート)	通常 8 本のピンのグループ。
positive edge (ポジティブ エッジ)	論理 0 から論理 1 への遷移。立ち上がりエッジともいわれる
posted interrupts (通知された割り込み)	マスク ビットを適用する前のハードウェアによって検出された割り込み。通知されたがマスクされていない割り込みは保留中の割り込みになる
Power On Reset (POR) (パワーオン リセット)	電圧が事前設定したレベルを下回ると、PSoC デバイスを強制的にリセットさせる回路。これはハードウェア リセットの一種
program counter (プログラム カウンター)	命令ポインタ (プログラム カウンターとも呼ばれている) は、CPU が命令を実行しているメモリ上の番地を指し示すコンピューター プロセッサのレジスタ。特定マシンの詳細に応じて、実行されている命令のアドレスまたは次に実行すべき命令のアドレスを保存
protocol (プロトコル)	規約の集合。特にネットワーク上の通信を監視するルール
PSoC®	サイプレスのプログラマブル システムオンチップ (PSoC®) デバイス
PSoC blocks (PSoC ブロック)	アナログ ブロックおよびデジタル ブロックを参照してください
PSoC Creator™	サイプレスの次世代のプログラマブル システムオンチップ技術のソフトウェア
pulse (パルス)	信号のいくつかの特性 (例えば位相や周波数) が、ベースライン値から急峻に変化し、その後ベースライン値に急峻に復帰すること
pulse width modulator (PWM) (パルス幅変調器)	関数としてデューティ比を変化させる出力

R

RAM ランダム アクセス メモリ (random access memory) の頭字語。データを読み出したり、新しいデータを書き込んだりすることができるデータ ストレージ デバイス

register (レジスタ) ビットやバイトなど、特定の容量を持つストレージ デバイス

reset (リセット) システムを既知の状態に戻す手段。ハードウェア リセットおよびソフトウェア リセットを参照してください

resistance (抵抗) 導電体に対する電流の流れへの抵抗 (オーム単位で測定)

revision ID (リビジョン ID) PSoC デバイス独自の識別コード

ripple divider (リップル分周器) フリップフロップで構成される非同期のリップル カウンター。クロック信号はカウンターの第 1 段に供給される。n 段のフリップフロップから成る n ビットの 2 進数カウンターは、2 進数で 0 から 2^n - 1 までカウントすることが可能

rising edge (立ち上がりエッジ) ポジティブ エッジを参照してください

ROM 読み出し専用メモリ (read only memory) の頭字語。データを読み出すことはできるが、新しいデータを書き込むことはできないデータ ストレージ デバイス

routine (ルーチン) 共有または頻繁に使用されている別のコード ブロックによって呼び出されるコード ブロック

routing (配線) 参照ライブラリで設定された設計ルールに従って、設計におけるオブジェクトを物理的に接続すること

ラント パルス デジタル回路において、信号の変化が閾値を超えず、有効な HIGH または LOW レベルに達しない狭パルス。例えば、非同期クロックを切り替える際、または信号が回路を介して 2 つの個別の経路を割り当てるという競合状態の結果としてラント パルスが発生する。競合状態は異なる遅延を持っており、グリッチを形成するためにまたはフリップフロップの出力が準安定状態になる時に再結合される

S

sampling (サンプリング) アナログ信号を一連のデジタル値 (またはその逆) に変換するプロセスのこと

schematic (回路図) 電子回路の要素やコンピューターの論理図の要素などのシステム要素を詳細に記述する図、図面または見取図

seed value (シード値) リニア フィードバック シフト レジスタまたは乱数発生器にロードされた初期値

serial (シリアル)	1. すべてのイベントが相次いで発生するプロセスを示す 2. 単一のデバイスまたはチャネルにある 2 つ以上の関連するアクティビティの逐次的または連続的発生を示す
set (セット)	ビット／レジスタの値を論理「1」にする
settling time (セトリング時間)	入力が変化した後に、出力の信号または値が安定化するのに要する時間
shift (シフト)	ワード内の各ビットを 1 つ右または左に移動させること。例えば 16 進値の 0x24 を左に 1 つずらすと、0x48 となる。16 進数の 0x24 を右に 1 つずらすと、0x12 となる
shift register (シフト レジスタ)	シリアル データ ストリームを出力するために、ワードを連続して右方移動または左方移動するメモリ ストレージ デバイス
sign bit (符号ビット)	符号つき 2 進数の最上位の桁またはビット。論理 1 に設定された場合は負数を表す
signal (信号)	情報を伝達するために使用される検出可能なエネルギー。電気機器に適用されている場合、信号は任意の送信された電気インパルス
silicon ID (シリコン ID)	PSoC シリコン独自の識別コード
skew (スキュー)	パラレル伝送において伝送されるビット間の到達時間差
slave device (スレーブ デバイス)	他のデバイスに、2 つのデバイス間のデータ交換のタイミングを制御させるデバイス。または、デバイスがいくつかカスケード接続されている場合、スレーブ デバイスは、他のデバイスにカスケード接続されたデバイスと外部インターフェース間のデータ交換のタイミングを制御させるもの。制御するデバイスは、マスター デバイスと呼ばれる
software (ソフトウェア)	データ処理システムの動作に関連するコンピューター プログラム、手順および各種文書 (例えばコンパイラ、ライブラリ ルーチン、マニュアルおよび回路図)。ソフトウェアはまずソースコードとして書かれ、そしてコードが実行するデバイスに特定の 2 進数形式で変換される
software reset (ソフトウェア リセット)	システムの一部を既知の状態に復帰させる、ソフトウェアによって実行される部分リセット。ソフトウェア リセットは、M8CP (PSoC ブロック、システム、ペリフェラルまたはレジスタではない) を元の状態に復帰させる。ソフトウェア リセットにより、CPU レジスタ (CPU_A、CPU_F、CPU_PC、CPU_SP および CPU_X) は 0x00 に設定される。従って、コードの実行はフラッシュ アドレス 0x0000 から開始する
SRAM	スタティック ランダム アクセス メモリ (static random access memory) の頭字語。ユーザーが高速にデータを格納および取得することを可能にするメモリ デバイス。「スタティック」という用語が使用される理由は、値が SRAM セルにロードされた時、明示的に変更されるかデバイスの電源が切られるまで変わらないため
SROM	監視用読み出し専用メモリ (supervisory read only memory) の頭字語。SROM は、デバイスを起動し、回路を較正し、フラッシュ動作を実行するために使用されるコードを保持。SROM の機能は、フラッシュ メモリから実行される通常ユーザー コードでアクセスすることが可能
stack (スタック)	Last In First Out (LIFO) という特徴を持つデータ構造の一種。これは即ち、最後に入力したデータが先に出力されるということ

stack pointer (スタック ポインタ)	スタックはコンピューター内部のメモリ セルのブロックで表示されている。スタックの底部アドレスは固定され、最上部セルのポインタが変化する
state machine (ステート マシン)	ある機能の (ハードウェアまたはソフトウェアにおける) 実際の実行。その機能は、状態とその状態間で切り替えるシーケンスの集合を含んでいると見なされる
sticky (スティッキー)	レジスタのビットで、あるイベントが発生するときに遷移が発生し、イベントが完了しても戻らずその状態を維持するようなビットのこと
stop bit (ストップ ビット)	受信デバイスが次の文字またはブロックを受信するように文字またはブロックの後に続く準備通知信号
switching (スイッチング)	論理演算や算術演算を実行するまたはネットワーク内の特定のポイント間でデータを転送するための、回路における信号の制御または配線
switch phasing (スイッチ位相)	スイッチ コンデンサ (SC) ブロックの特定のスイッチ (PHI1 または PHI2) を制御するクロックのこと。PSoC SC ブロックには、2 グループのスイッチを持つ。その中の 1 つのグループは、通常、PHI1 の間は閉じられ、PHI2 の間は開かれる。残りのグループは、PHI1 の間は開かれ、PHI2 の間は閉じられる。これらのスイッチは通常動作または PHI1 と PHI2 クロックが逆転された場合に逆転モードでも制御可能
synchronous (同期)	1. クロック信号の次のアクティブ エッジまで動作したり、受け取られることのないデータを持つ信号 2. 動作がクロック信号によって同期されるシステム

T

tap (タップ)	シフト レジスタや抵抗分圧器など、複数のブロック／コンポーネントを一連に接続することにより作成されたデバイスの 2 ブロック間の接続
terminal count (ターミナル カウント)	カウンターが 0 までカウントする状態
threshold (閾値)	検討中のシステムまたはセンサーによって検出される信号の最小値
Thumb-2 命令セット	Thumb-2 は、使いやすさ、コード サイズおよび性能の面で大幅なメリットを提供する効率的かつ強力な命令セット。Thumb-2 命令セットは、以前の 16 ビットの Thumb 命令セットのスーパーセットとなり、32 ビットの命令に加え、追加の 16 ビットの命令を持つ
transistors (トランジスタ)	トランジスタは増幅またはスイッチ動作をさせるための固体の半導体デバイスであり、3 つの端子を持つ。1 つの端子に印加された少量の電流や電圧は、残りの 2 つの端子間の電流を制御する。これはあらゆる近代の電子工学における主力素子である。デジタル回路では、トランジスタは超高速の電気スイッチとして使用されており、トランジスタの配置は論理ゲート、RAM 型のメモリおよび他のデバイスとして機能することが可能。アナログ回路では、トランジスタは基本的に増幅器として使用されている
tristate (トライステート)	出力が 0、1、Z (高インピーダンス) の 3 つの状態となる機能。この機能は Z 状態ではどんな値も駆動せず、多くの面では、回路の残りの部分から切断された状態として考慮されるため、他の出力が同じ *ライン*を駆動することが可能

U

UART	UART またはユニバーサル非同期レシーバー トランスミッターは、データのパラレル ビットとシリアル ビット間での変換を行う
user (ユーザー)	PSoC デバイスを使用したり、本マニュアルを購読したりする人
user modules (ユーザー モジュール)	低位のアナログおよびデジタル PSoC ブロックを管理およびコンフィギュレーションする、事前構築されたテスト済みのハードウェア／ファームウェアのペリフェラル機能。ユーザー モジュールはペリフェラル機能に高位の API (アプリケーション プログラミング インターフェース) も提供
user space (ユーザー空間)	レジスタ マップのバンク 0 空間。このバンクのレジスタは、初期化中にだけではなく、通常のプログラム実行中にも変更される可能性が高い。バンク 1 のレジスタはプログラムの初期化フェーズでのみ変更される可能性が最も高い

V

V_{DDD}	「ドレイン電圧」を意味する電源ラインの名称。正の最大の電源供給信号。通常 5 または 3.3 ボルト
volatile (揮発性)	範囲外の場合、同じ値またはレベルに維持することは保証されない
V_{SS}	「ソース電圧」を意味する電源ラインの名称。負の最小の電源供給信号

W

watchdog timer (ウォッチドッグ タイマー)	定期的に点検整備される必要があるタイマー。点検されない場合、CPU は一定時間経過後にリセットされる
waveform (波形)	信号を 振幅対時間のプロットとして表現したもの

X

XOR (排他的論理和)	ブール代数を参照してください