PSoC 4100M/4200M 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 : 001-98439 Rev. **

2015 年 10 月 5 日

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone (USA): 800.858.1810
Phone (Intnl): 408.943.2600
japan.cypress.com

ライセンス

© 2014-2015, Cypress Semiconductor Corporation. 無断複写、転載を禁止。このソフトウェアおよび関連ドキュメントまたは資料はサイプレスセミコンダクタ社 (以下サイプレス) が所有し、全世界の特許保護 (米国およびその他の国)、米国の著作権法ならびに国際協定の条項により保護され、かつそれらに従います。サイプレスとお客様の別のライセンス契約書に特記されていない限り、お客様は他の著作物のように資料を扱うものとします。

お客様は資料を機密に扱い、サイプレスから書面による承認を得なければ、資料の使用または開示してはいけません。お客様はサイプレスとのあらゆる秘密保持契約に従うものとします。

資料は第三者のライセンスを受ける可能性がある項目を含んでいる場合、そのライセンスに従うものとします。

著作権

Copyright © 2014-2015 Cypress Semiconductor Corporation. 無断複写、転載を禁止。

PSoC と CapSense は登録商標であり、PSoc Creator、Cypress® および Cypress Semiconductor™ はサイプレス セミコンダクタ社 (サイプレス) の商標です。本書で言及するその他のすべての商標または登録商標は各社の所有物です。

I^2C コンポーネントをサイプレスまたはサブライセンスを持つ関連業者から購入すると、Philips I^2C の特許権の下でライセンスが付与されます。このライセンスにより、システムが Philips の指定する I^2C の標準仕様を満たす限り、I^2C システムでこれらのコンポーネントを使用できます。2006 年 10 月 1 日以降、Philips Semiconductors 社は新社名 NXP Semiconductors を使用しています。

本文書に記載される情報は、予告なく変更されることがあり、サイプレスによる誓約として解釈されるべきではありません。合理的な予防策を講じているため、サイプレスは本書に表示するエラーに対して一切の責任を負いません。サイプレスの書面による事前の許可なく、いかなる形式、方法、手段によっても本書の内容の一部または全部を無断複写・複製することを禁じます。米国製

免責条項

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

フラッシュコードの保護

サイプレス製品は、該当する特定のサイプレス データ シートに記載されている仕様を満たします。サイプレスは、市販されている同様の PSoC 製品ファミリにおいて、製品ファミリが、使用方法にかかわらず最高水準の安全性を有すると確信しています。コード保護機能を侵害する方法があるかも知れません。サイプレスの知る限り、そのような方法はすべて不正で、かつ違法と考えられます。サイプレスまたはその他の半導体メーカのいずれも、自社のコードのセキュリティを保証することはできません。コードの保護は、サイプレスが製品の「解読不能」を保証していることを意味するものではありません。

サイプレスには、自社コードの完全性に関心があるユーザーと協力する意思があります。コードの保護は絶えず進化しております。サイプレスは当社製品のコード保護機能の継続的改善に努めています。

章目次

セクション A: 概要 **15**

1. はじめに 17
2. 開発サポート 23
3. 文書の構成 25

セクション B: CPU システム **29**

4. Cortex-M0 CPU 31
5. DMA コントローラー モード 37
6. 割り込み 47

セクション C: システムワイドリソース **57**

7. I/O システム 59
8. クロック供給システム 67
9. 電源と電圧監視 75
10. チップの動作モード 79
11. 電力モード 81
12. ウォッチドッグ タイマー 85
13. リセット システム 89
14. デバイス セキュリティ 93

セクション D: デジタル システム **95**

15. シリアル通信 (SCB) 97
16. UDB 137
17. CAN 177
18. タイマー、カウンター、PWM 193

セクション E: アナログ システム **215**

19. 高精度リファレンス 217
20. SAR ADC 221
21. 低消費電力コンパレーター 255
22. 連続時間ブロックミニ (CTBm) 261
23. LCD ダイレクト ドライブ 271
24. CapSense 283
25. 温度センサー 293

セクション F: プログラムおよびデバッグ **297**
26. デバッグ インターフェース 299
27. 不揮発性メモリ プログラム 307

用語集 **321**

インデックス **337**

詳細目次

セクション A: 概要 **15**

ドキュメント改訂履歴 15

1. はじめに **17**

1.1 トップ レベル アーキテクチャ 18
1.2 特長....19
1.3 CPU システム....19
1.3.1 プロセッサ....19
1.3.2 割り込みコントローラー....20
1.3.3 Direct Memory Access (ダイレクト メモリ アクセス)....20
1.4 メモリ....20
1.4.1 フラッシュ....20
1.4.2 SRAM....20
1.5 システムワイド リソース....20
1.5.1 クロック システム....20
1.5.2 電源システム....20
1.5.3 GPIO....20
1.6 プログラマブル デジタル....20
1.7 固定機能デジタル ブロック....21
1.7.1 タイマー／カウンタ／ PWM ブロック....21
1.7.2 シリアル通信ブロック....21
1.7.3 コントローラー エリア ネットワーク....21
1.8 アナログシステム....21
1.8.1 SAR ADC....21
1.8.2 連続時間ミニブロック (CTBm)....21
1.8.3 低消費電力コンパレータ....21
1.9 特殊周辺機能....21
1.9.1 LCD セグメント ドライブ....21
1.9.2 CapSense....22
1.10 プログラムおよびデバッグ....22

2. 開発サポート **23**

2.1 サポート....23
2.2 製品アップグレード....23
2.3 開発キット....23

3. 文書の構成 **25**

3.1 大セクション....25
3.2 本書の表記法....25
3.2.1 レジスタの表記法....25
3.2.2 数値の表記法....25
3.2.3 測定単位....26
3.2.4 略語....26

セクション B: CPU システム 29
トップ レベル アーキテクチャ 29
4. Cortex-M0 CPU 31
4.1 特長 31
4.2 ブロック図 32
4.3 動作原理 32
4.4 アドレス マップ 32
4.5 レジスタ 33
4.6 動作モード 34
4.7 命令セット 34
4.7.1 アドレス アライメント 35
4.7.2 メモリ エンディアン 35
4.8 Systick タイマー 35
4.9 デバッグ 35
5. DMA コントローラー モード 37
5.1 ブロック図の解説 37
5.2 ディスクリプタ 39
5.2.1 アドレス コンフィギュレーション 39
5.2.2 転送サイズ 40
5.2.3 ピン／ポン ディスクリプタ 41
5.2.4 動作モード 41
5.2.5 動作とタイミング 44
5.3 レジスター覧 46
6. 割り込み 47
6.1 特長 47
6.2 動作原理 47
6.3 割り込みと例外の動作 48
6.3.1 PSoC4 における割り込みと例外処理 48
6.3.2 レベルおよびパルス割り込み 48
6.3.3 例外ベクタ テーブル 49
6.4 例外ソース 49
6.4.1 リセット例外 49
6.4.2 マスク不可能割り込み (NMI) 例外 49
6.4.3 HardFault 例外 50
6.4.4 監視呼び出し (SVCall) 例外 50
6.4.5 PendSV 例外 50
6.4.6 SysTick 例外 50
6.5 割り込みソース 50
6.6 例外の優先度 52
6.7 割り込みの有効化 52
6.8 例外状態 53
6.8.1 保留中の例外 53
6.9 例外のスタック使用量 54
6.10 割り込みと低消費電力モード 54
6.11 例外の初期化とコンフィギュレーション 54
6.12 レジスタ 55
6.13 関連文書 55
セクション C: システムワイドリソース 57
トップ レベル アーキテクチャ 57

7. I/O システム 59

7.1 特長 59
7.2 ブロック図 60
7.3 GPIO 駆動モード 61
7.3.1 ハイインピーダンスアナログ 62
7.3.2 ハイインピーダンスデジタル 62
7.3.3 抵抗プルアップまたは抵抗プルダウン 62
7.3.4 オープン ドレイン - HIGH に駆動および LOW に駆動 62
7.3.5 ストロング駆動 62
7.3.6 抵抗プルアップおよび抵抗プルダウン 62
7.4 スルーレート制御 63
7.5 CMOS LVTTL レベルの制御 63
7.6 高速 I/O マトリックス 63
7.7 ファームウェアで制御される GPIO 64
7.8 アナログ IO 64
7.9 LCD 駆動 64
7.10 CapSense 64
7.11 I/O ポートの再コンフィギュレーション 64
7.12 電源投入時の I/O 状態 65
7.13 省電力モードでの動作 65
7.14 GPIO 割り込み 65
7.14.1 特長 65
7.14.2 割り込みコントローラー ブロック図 65
7.14.3 機能とコンフィギュレーション 66
7.15 入力と出力の同期 66
7.16 レジスタ 66

8. クロック供給システム 67

8.1 ブロック図 67
8.2 クロック源 69
8.2.1 内部主発振器 (IMO) 69
8.2.2 内部低速発振器 (ILO) 70
8.2.3 外部クロック (EXTCLK) 70
8.2.4 時計用水晶発振器 (WCO) 70
8.3 クロック分配 70
8.3.1 HFCLK 入力の選択 71
8.3.2 LFCLK 入力の選択 71
8.3.3 SYSCLK プリスケーラのコンフィギュレーション 71
8.3.4 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 71
8.4 低消費電力動作 73
8.5 レジスター覧 74

9. 電源と電圧監視 75

9.1 ブロック図 75
9.2 動作原理 76
9.2.1 レギュレータの概要 76
9.3 電圧監視 76
9.3.1 パワーオン リセット（POR） 76
9.4 レジスタの一覧 77

10. チップの動作モード 79

10.1 ブート 79
10.2 ユーザー 79
10.3 特権 79
10.4 デバッグ 79

11. 電力モード 81
11.1 アクティブ モード 82
11.2 スリープ モード 82
11.3 ディープスリープ モード 82
11.4 ハイバネート モード 83
11.5 ストップモード 83
11.6 電力モードの概要 83
11.7 省電力モードへの遷移および復帰 83
11.8 レジスター覧 84

12. ウォッチドッグ タイマー 85
12.1 特長 85
12.2 ブロック図 85
12.3 動作原理 86
12.3.1 WDT の有効化と無効化 87
12.3.2 WDT 動作モード 87
12.3.3 WDT 割り込みと省電力モード 88
12.3.4 WDT リセット モード 88
12.4 レジスター覧 88

13. リセット システム 89
13.1 リセット ソース 89
13.1.1 パワーオン リセット 89
13.1.2 ブラウンアウトリセット 89
13.1.3 ウォッチドッグ リセット 89
13.1.4 ソフトウェア初期化リセット 90
13.1.5 外部リセット 90
13.1.6 保護フォルト リセット 90
13.1.7 ハイバネート復帰リセット 90
13.1.8 ストップモード復帰 リセット 90
13.2 リセット ソースの識別 90
13.3 レジスター覧 91

14. デバイス セキュリティ 93
14.1 特長 93
14.2 動作原理 93
14.2.1 デバイス セキュリティ 93
14.2.2 フラッシュのセキュリティ 94

セクション D: デジタル システム 95
トップ レベル アーキテクチャ 95

15. シリアル通信 (SCB) 97
15.1 特長 97
15.2 シリアル ペリフェラル インターフェース (SPI) 97
15.2.1 特長 97
15.2.2 概要 98
15.2.3 SPI の動作モード 98
15.2.4 スレーブへのクロック供給用の SPI マスターの使用 103
15.2.5 イージー SPI プロトコル 103
15.2.6 SPI レジスタ 106
15.2.7 SPI 割り込み 106
15.2.8 SPI の有効化と初期化 107
15.2.9 内部と外部クロック供給 SPI 動作 108

15.3 UART 110
15.3.1 特長 110
15.3.2 概要 111
15.3.3 UART の動作モード 111
15.3.4 UART レジスタ 117
15.3.5 UART 割り込み 118
15.3.6 UART の有効化と初期化 118
15.3.7 特長 120
15.3.8 概要 120
15.3.9 用語および定義 120
15.3.10 I2C 動作モード 121
15.3.11 イージー I2C (EZI2C) プロトコル 123
15.3.12 I2C レジスタ 124
15.3.13 I2C 割り込み 125
15.3.14 I2C の有効化および初期化 126
15.3.15 I2C における内部および外部クロック動作 127
15.3.16 スリープから復帰 128
15.3.17 マスター モード転送の例 129
15.3.18 スレーブ モード転送の例 131
15.3.19 EZ スレーブ モード転送の例 133
15.3.20 マルチマスター モード転送の例 135

16. UDB 137

16.1 特長 137
16.2 動作原理 138
16.2.1 PLD 138
16.2.2 データパス 140
16.2.3 ステータスおよび制御モジュール 157
16.2.4 リセットおよびクロック制御モジュール 164
16.2.5 UDB アドレス指定 170
16.2.6 システム バス アクセスの一貫性 170
16.3 ポート アダプタ ブロック 171
16.3.1 PA データ入力ロジック 171
16.3.2 PA ポート ピン クロック マルチプレクサ ロジック 172
16.3.3 PA 出力ロジック 172
16.3.4 PA 出力イネーブル ロジック 173
16.3.5 PA クロック マルチプレクサ 174
16.3.6 PA リセット マルチプレクサ 174

17. CAN 177

17.1 特長 177
17.2 ブロック図 178
17.3 CAN メッセージ フレーム 178
17.3.1 データ フレーム 178
17.3.2 リモート フレーム 179
17.3.3 エラー フレーム 179
17.3.4 オーバーロード フレーム 180
17.4 CAN におけるメッセージ送信 180
17.4.1 メッセージ アービトレーション 180
17.4.2 メッセージ送信プロセス 180
17.4.3 メッセージ中止 181
17.4.4 シングル ショット送信 181
17.4.5 拡張データ フレームの送信 181

17.5 CAN におけるメッセージ受信 182
17.5.1 メッセージ受信プロセス 182
17.5.2 アクセプタンス フィルター 182
17.5.3 DeviceNet フィルター処理 183
17.5.4 拡張データ フレームのフィルター処理 184
17.5.5 受信メッセージ バッファの連結 184
17.6 リモート フレーム 185
17.6.1 要求ノードによるリモート フレームの送信 185
17.6.2 リモート フレームの受信 185
17.6.3 RTR 自動応答 186
17.6.4 拡張フォーマットでのリモート フレーム 186
17.7 時間起動 CAN 186
17.7.1 TTCAN タイマー 186
17.8 ビット タイム コンフィギュレーション 186
17.8.1 取り得るビット レートとステム クロック (SYSCLK) 186
17.8.2 ビット レート TSEG1 と TSEG2 の設定 187
17.9 CAN におけるエラー処理と割り込み 189
17.9.1 エラーの種類 189
17.9.2 エラー キャプチャ レジスタ 189
17.9.3 CAN におけるエラー状態 189
17.9.4 CAN における割り込みソース 190
17.10 CAN における動作モード 192
17.10.1 実行／停止モード 192
17.10.2 リッスンオンリー モード 192
17.10.3 ループバック テスト モード 192

18. タイマー、カウンター、PWM 193

18.1 特長 193
18.2 ブロック図 194
18.2.1 TCPWM ブロックにおけるカウンターのイネーブル 194
18.2.2 クロック 194
18.2.3 トリガー入力に基づいたイベント 195
18.2.4 出力信号 196
18.2.5 電力モード 197
18.3 動作モード 197
18.3.1 タイマー モード 198
18.3.2 キャプチャ モード 201
18.3.3 直交デコーダー モード 203
18.3.4 パルス幅変調モード 206
18.3.5 デッドタイム付きパルス幅変調モード 209
18.3.6 疑似乱数パルス幅変調モード 211
18.4 TCPWM レジスタ 213

セクション E: アナログ システム 215

トップ レベル アーキテクチャ 215

19. 高精度リファレンス 217

19.1 特長 217
19.2 ブロック図 217
19.3 動作原理 218
19.3.1 高精度バンドギャップ 218
19.3.2 調整値バッファ 218
19.3.3 低消費電力バッファ 218

19.3.4 カレントミラー....219
19.3.5 温度制御電圧発生器....219
19.3.6 温度制御電流発生器....219
19.4 コンフィギュレーション....219

20. SAR ADC 221

20.1 特長....221
20.2 ブロック図....222
20.3 動作原理....223
20.3.1 SAR ADC コア....223
20.3.2 SARMUX....226
20.3.3 SARREF....233
20.3.4 SARSEQ....234
20.3.5 割り込み....237
20.3.6 トリガー....239
20.3.7 SAR ADC ステータス....240
20.3.8 低消費電力モード....240
20.3.9 システム動作....240
20.3.10 レジスタ モード....242
20.3.11 DSI モード....244
20.3.12 アナログ配線のコンフィギュレーション例....248
20.3.13 温度センサー コンフィギュレーション....251
20.4 レジスタ....252

21. 低消費電力コンパレーター 255

21.1 特長....255
21.2 ブロック図....255
21.3 動作原理....256
21.3.1 入力コンフィギュレーション....256
21.3.2 出力および割り込みコンフィギュレーション....256
21.3.3 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....258
21.3.4 ヒステリシス....259
21.3.5 省電力モードからの復帰....259
21.3.6 コンパレーター クロック....259
21.3.7 オフセット調整....259
21.4 レジスタの概要....260

22. 連続時間ブロックミニ (CTBm) 261

22.1 特長....261
22.2 ブロック図....261
22.3 動作原理....262
22.3.1 電力モードのコンフィギュレーション....262
22.3.2 ドライブレベルコンフィギュレーション....263
22.3.3 位相補償....264
22.3.4 スイッチ制御....264
22.4 レジスタの概要....269

23. LCD ダイレクト ドライブ 271

23.1 特長....271
23.2 LCD セグメント ドライブの概要....271
23.2.1 ドライブ モード....272
23.2.2 ドライブ モードの推奨使用方法....280
23.2.3 デジタル コントラスト制御....280

23.3 ブロック図 281
23.3.1 動作原理 281
23.3.2 高速および低速のマスター信号発生器 281
23.3.3 マルチプレクサおよび LCD ピン回路 282
23.3.4 ディスプレイ データ レジスタ 282
23.4 レジスター覧 282

24. CapSense 283

24.1 特長 283
24.2 ブロック図 283
24.3 動作原理 284
24.4 CapSense CSD のセンシング 285
24.4.1 GPIO セルの静電容量／電流コンバータ 285
24.4.2 CapSense クロック ジェネレータ 287
24.4.3 シグマデルタ コンバータ 287
24.5 CapSense CSD シールド 289
24.5.1 CMOD プリチャージ 290
24.6 汎用リソース : IDAC とコンパレータ 291
24.7 レジスター覧 291

25. 温度センサー 293

25.1 特長 293
25.2 動作原理 293
25.3 温度センサー コンフィギュレーション 294
25.4 アルゴリズム 295
25.5 レジスタ 296

セクション F: プログラムおよびデバッグ 297

トップ レベル アーキテクチャ 297

26. デバッグ インターフェース 299

26.1 特長 299
26.2 機能の詳細 299
26.3 シリアル ワイヤ デバッグ (SWD) インターフェース 300
26.3.1 SWD タイミングの詳細 301
26.3.2 ACK 応答の詳細 301
26.3.3 ターンアラウンド (Trn) 期間の詳細 301
26.4 Cortex-M0 デバッグおよびアクセス ポート (DAP) 301
26.4.1 デバッグ ポート (DP) レジスタ 302
26.4.2 アクセス ポート (AP) レジスタ 302
26.5 PSoC 4 デバイスのプログラム 302
26.5.1 SWD ポートの開通 302
26.5.2 SWD プログラム モードへの移行 303
26.5.3 SWD プログラム ルーチンの実施 303
26.6 PSoC 4 SWD デバッグ インターフェース 303
26.6.1 デバッグ制御およびコンフィギュレーション レジスタ 303
26.6.2 ブレークポイント ユニット (BPU) 304
26.6.3 データ ウォッチポイント (DWT) 304
26.6.4 PSoC 4 デバイスのデバッグ 304
26.7 レジスタ 305

27. 不揮発性メモリ プログラム 307

27.1 特長.....307
27.2 機能の詳細.....307
27.3 システム コールの実装.....308
27.4 ブロッキングと非ブロッキングのシステム コール.....308
27.4.1 システム コールの実行.....308
27.5 システム コール.....309
27.5.1 シリコン ID.....309
27.5.2 ロード フラッシュ バイト.....310
27.5.3 列書き込み.....311
27.5.4 列プログラム.....312
27.5.5 全消去.....313
27.5.6 チェックサム.....313
27.5.7 書き込み保護.....314
27.5.8 非ブロッキング列書き込み.....315
27.5.9 非ブロッキング 列プログラム.....316
27.5.10 レジューム非ブロッキング.....316
27.6 システム コール ステータス.....317
27.7 非ブロッキング システム コール疑似コード.....317

用語集 321

インデックス 337

セクション A: 概要

このセクションは次の章を含みます。

- はじめに (17 ページ)
- 開発サポート (23 ページ)
- 文書の構成 (25 ページ)

ドキュメント改訂履歴

版	発行日	変更者	変更内容
**	10/05/2015	HZEN	これは英語版 001-95223 Rev. *A を翻訳した日本語版 001-98439 Rev. ** です。

1. はじめに

PSoC® 4 は ARM® Cortex™-M0 CPU 付きのプログラマブルな組み込みシステム コントローラーです。これは、高性能の ARM Cortex-M0 サブシステムとプログラマブルなアナログ ブロック、ユーザー設定可能なデジタル回路および汎用の固定機能ブロックを組み合わせたものです。

PSoC 4100M/4200M は PSoC 4100/4200 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
- 固定機能ブロックとコンフィギュレーション可能なデジタル ブロック
- デジタル回路がプログラム可能
- 高性能アナログ システム
- 柔軟でプログラマブルな相互接続
- 静電容量タッチ センシング (CapSense®)
- スリープ、ディープスリープ、ハイバネートおよびストップモードを含む省電力モード

本書は PSoC4 デバイスの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 デザインを作成するのに役に立ちます。

1.1 トップ レベル アーキテクチャ

図 1-1 と図 1-2 に PSoC 4100M と PSoC 4200M それぞれの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 4100M ファミリ ブロック図

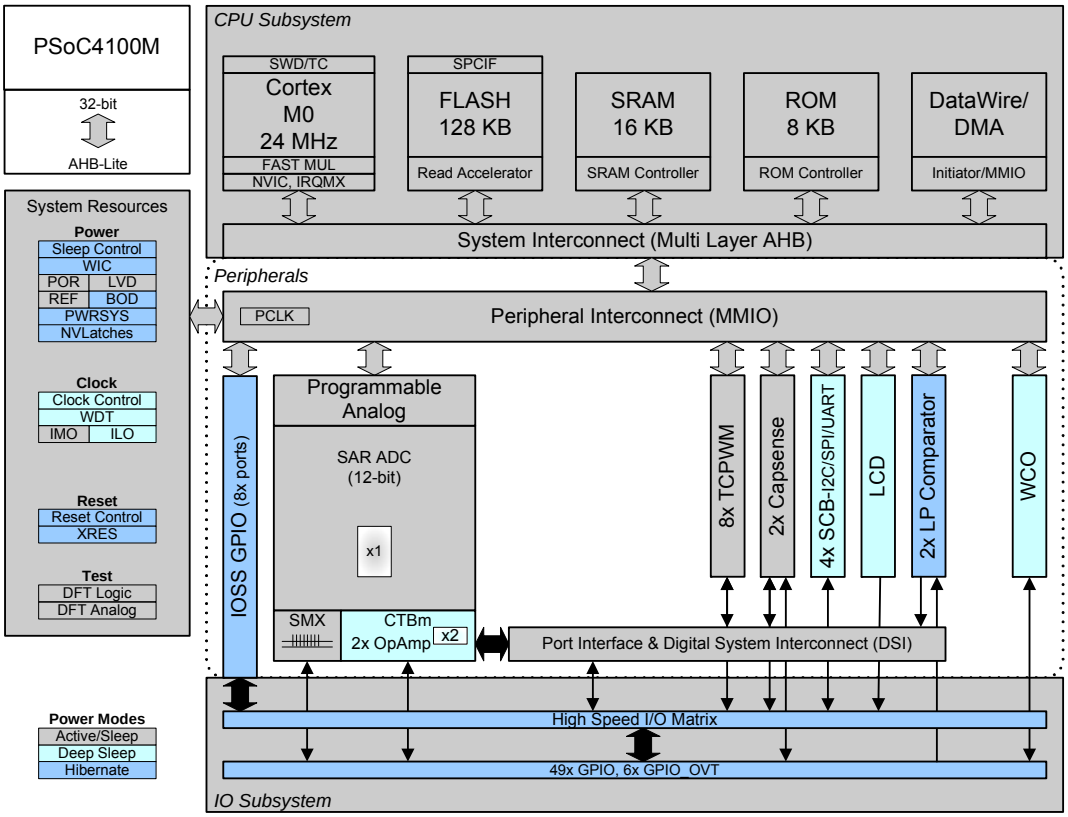
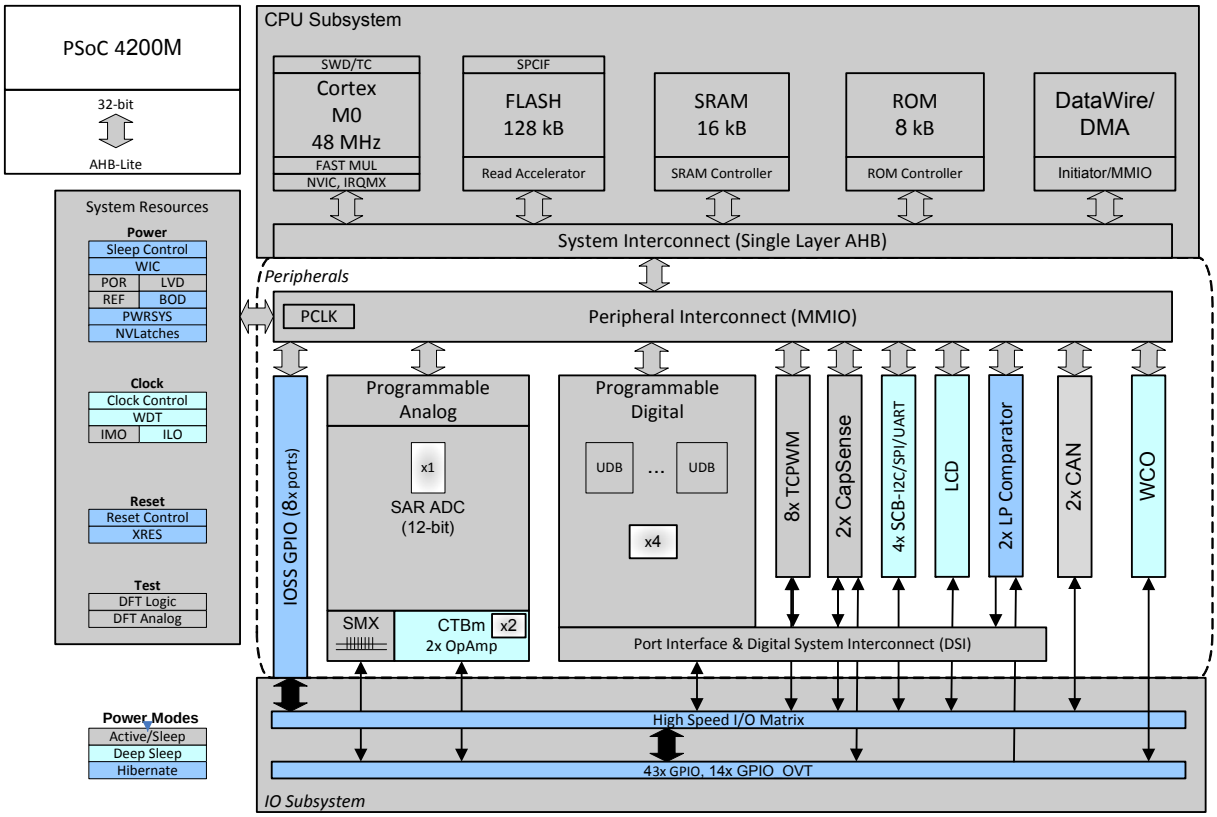


図 1-2. PSoC 4200M ファミリ ブロック図



1.2 特長

PSoC 4100M/4200M ファミリには以下の主要コンポーネントがあります :

- シングルサイクル乗算に対応した 32 ビット -M0 CPU (PSoC 4200Mは48 MHzで最高43 DMIPS、PSoC 4100Mは 24MHz で最高 21 DMIPS)
- 最大 128KB フラッシュおよび 16KB SRAM
- ダイレクト メモリ アクセス (DMA)
- 8 つの中央揃えパルス幅変調器 (PWM、デッドバンドがプログラム可能なコンプリメンタリ出力)
- ゼロオーバーヘッドのシーケンス機能付きのサンプルホールド (S&H) 機能を備えた 12 ビット ADC (サンプリングレートが PSoC 4200M では 1Msps、PSoC) 4100M では 806Msps)
- コンパレータモードと逐次比較型 AD コンバーター (SAR ADC) 入力バッファリング機能を備えた最大 4 つのオペアンプ
- 2 つの低消費電力コンパレータ
- 4 つのシリアル通信ブロック (SCB) が SPI/UART/I2C シリアル通信チャネルとして動作
- PSoC 4200M 内の 2 つのコントローラー エリア ネットワーク (CAN) ブロック
- PSoC 4200M ではユニバーサル デジタル ブロック (UDB) と呼ばれる最大 4 つのプログラマブルな論理ブロック
- CapSense
- セグメント LCD ダイレクト ドライブ
- 省電力動作モード : スリープ、ディープスリープ、ハイバネートおよびストップ
- シリアル ワイヤ デバッグ (SWD) を介してのシステムのプログラムおよびデバッグ
- 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

PSoC4 の心臓部は 32 ビット Cortex-M0 CPU コアです。PSoC 4200Mでは最大48MHz、PSoC4100Mでは最大24MHz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備え、省電力動作に最適化されています。これは 16 ビット命令を使用し、Thumb-2 命令セットを実行します。これにより、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。

PSoC 4 には、1 サイクルで 32 ビットの結果を出すハードウェア乗算器を実装しています。

1.3.2 割り込みコントローラー

PSoC 4 の CPU サブシステムには、32 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させられる復帰割り込みコントローラー (WIC) があります。PSoC 4 の Cortex-M0 CPU は、デジタル配線に結合できるマスク不可能な割り込み (NMI) 入力を実装しています。

1.3.3 Direct Memory Access (ダイレクト メモリ アクセス)

DMA エンジンは、プログラム可能なディスクリプタ チェーンを使用して、メモリマップ内の任意の場所に 独立したデータ転送が可能です (周辺機能から周辺機能への転送、メモリと周辺機能間の転送)。

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 デバイスは、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ために、CPU に緊密に結合されたフラッシュ アクセラレータ付きのフラッシュ モジュールを持っています。フラッシュ アクセラレータはシングル サイクル SRAM のアクセス性能の平均 85% を達成します。必要に応じて、EEPROM 動作をエミュレートするためにフラッシュ モジュールの一部を使用することができます。

1.4.2 SRAM

PSoC 4 はハイバネート モード中にデータが保持される SRAM を提供します。

1.5 システムワイド リソース

1.5.1 クロック システム

PSoC 4 デバイスのクロック システムは、内部クロックとしての内部主発振器 (IMO) および内部低速発振器 (ILO)、外部クロック入力、時計用水晶発振器 (WCO) から構成されます。

IMO は ±2%の精度を持ち、PSoC 4 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、1MHz の単位で 3 ～ 48MHz の間に調整できます。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ています。そのクロック周波数は 32kHz で ±60 パーセントの精度です。

DC から 48MHz のまでの範囲の外部クロック入力は、IMO の代わりPSoC 4機能ブロックで使用する各種クロックを生成するために利用することができます。

WCO は LFCLK のクロック源の 1 つとして使用されます。ディープスリープ モードでは時間間隔 を正確に維持するため WCO を使用します。ILO と同様、WCO はハイバネート モードとストップ モードを除くすべてのモードで使用可能です。

1.5.2 電源システム

PSoC 4 は、1.71 ～ 5.5 V の一系統の外部供給電圧により動作します。PSoC 4 は、通常動作を行うアクティブ モードの他に 4 つの省電力モードを持ちます : スリープ、ディープスリープ、ハイバネートおよびストップ

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、CPU はクロックが停止され機能しません。ディープスリープ モードでは、CPU、SRAM および高速な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ますが、低速クロックがオンにされ低速の周辺機能が動作します。ハイバネート モードでは、低速クロックもオフにされ、低速の周辺機能の動作も停止します。

さまざまな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

PSoC 4 の GPIO は以下の特長があります :

- 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
- 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
- 割り込みの生成 : エッジ トリガー
- CapSense と LCD ドライブ

PSoC 4100M/4200M は過電圧耐性ポート (ポート 6) を有します。これにより I2C 高速モードパワーダウン仕様への準拠を可能にし、より低い VDD で動作しながらより高い電圧バスに接続する機能を有します。ピンは8ビット幅のポートに構成されます。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のに使用されます。固定機能ブロックのピン位置も固定されています。

1.6 プログラマブル デジタル

PSoC 4200M は最大 4 つの UDB を持っています。各 UDB は、フレキシブル相互接続を有する構造データパス回路とデ

ジタル回路が含まれています。UDB アレイは、デジタルシステムインターコネクト (DSI) と呼ばれるスイッチ配線機構を提供します。DSI は、周辺機能とポートから UDB への信号配線、または UDB 内の信号配線を可能にします。

PSoC 4200M の UDB アレイは、カスタムロジック、タイマ / PWM の追加、I2C、SPI、I2S、UART などの通信インターフェースの追加を可能にします。

1.7 固定機能デジタル ブロック

1.7.1 タイマー／カウンタ／ PWM ブロック

タイマー／カウンター／ PWM ブロックは、ユーザーが周期をプログラムできる 8 個 の 16 ビットカウンタから成ります。これらのカウンタの動作を同期させることができます。各ブロック 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があります。ブロックは、デッドバンドをプログラムできるコンプリメンタリ出力をサポートしています。またキル入力もあり、強制的に出力を既定の状態にすることができます。他に、中央揃え PWM、クロックの分周、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。

1.7.2 シリアル通信ブロック

PSoC 4100M/4200M は I2C、汎用非同期レシーバ／トランスミッタ (UART) またはシリアル ペリフェラル インタフェース (SPI) などのシリアル通信インターフェースを実装できる 4 つの SCB を有しています。

各 SCB の機能は以下のとおりです :

- 標準的な I2C マルチマスタとスレーブ機能
- Motorola、TI、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 SPI スレーブ機能
- 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
- SPIおよびI2C用に32バイト バッファを利用するEZ機能モードのサポート

1.7.3 コントローラー エリア ネットワーク

2 つの CAN ブロックが、CAN の 2.0A と 2.0B をサポートする PSoC4200M で提供されます。これらのブロックは、専用のメッセージ フィルターが付属する16個の受信バッファと 8 つの送信バッファがあります。PHY インターフェースは業界標準のフィリップスのCAN PHYをサポートしています。

1.8 アナログシステム

1.8.1 SAR ADC

PSoC 4200M は、コンフィギュレーション可能な 12 ビット 1Msps の SAR ADC を持ちます。PSoC 4100M は同様の 806ksps12 ビット SAR ADC を持ちます。利得誤差は ±0.1 パーセント、積分非直線性 (INL) は 1 LSB 以下、微分非直線性 (DNL) は 1 LSB 以下、信号対雑音比 (SNR) は 68dB 以上の特性により、このコンバータは様々なアナログ用途に対応できます。

ADC はリファレンス電圧として、3 つの内部電圧リファレンス (V_{DDA}、$V_{DDA}/2$ および V_{REF}) と GPIO ピンを経由する外部リファレンス電圧から選択できます。SARは8入力シーケンサを介して固定されたピンの組に接続されます。シーケンサは、CPU 割り込みサービスの要件を軽減するために、各チャネルのデータをバッファリングすることができます。

1.8.2 連続時間ミニブロック （CTBm）

CTBm ブロックは、アナログ サブシステムの入出力点で連続時間機能を提供します。CTBm は、スイッチ配線マトリックスを持つ高度な設定が可能な高性能オペアンプを 2 つ持っています。オペアンプはコンパレータ モードでも動作します。PSoC 4100M/4200M は 2 個の CTBm ブロックがあります。

このブロックは、外部コンポーネント不要でオープン ループ オペアンプ、リニア バッファ、コンパレータ機能を実装することができます。PGA、電圧バッファ、フィルター、トランスインピーダンス アンプは外部コンポーネントを使用して実装することができます。CTBm ブロックは 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します。

1.8.3 低消費電力コンパレータ

PSoC 4100M/4200M は、ディープスリープモードとハイバネート モードで動作できる 2 つの低消費電力コンパレータを内蔵しています。これにより、低消費電力モード中に外部電圧レベルを監視する能力を維持しながらアナログ システム ブロックを無効にすることができます。

2 つの入力電圧は、2 つ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。

1.9 特殊周辺機能

1.9.1 LCD セグメント ドライブ

PSoC 4100M/4200M は、最大 4 コモン、すべての GPIO をコモンまたはセグメントに割り当てることができる LCD コントローラを備えています。内部で LCD 電圧を生成する必要のないフル デジタル方式を使用してLCDセグメントを駆動します。

1.9.2 CapSense

PSoC4 デバイスは CapSense 機能を持ちます。指による静電容量を利用してボタンのトグル、スライダ、ホイールなどの機能を実装できます。CapSense の機能は、CapSense シグマ - デルタ (CSD) ブロックにより PSoC4 内のすべて の GPIO ピンで利用可能です。CSD は耐水性能も提供します。

1.9.2.1 IDAC とコンパレータ

PSoC 4100M/4200M は汎用に 4 つの IDAC との 2 つのコンパレータを備えています。このうち 2 つの IDAC とコンパレータは CapSense のブロックの一部です。

1.10 プログラムおよびデバッグ

PS0C4 はオンチップ SWD インタフェースを使用して、デバイスのプログラミングとデバッグ機能をサポートしています。PSoC Creator IDE は、PsoC4 デバイス用の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 インターフェースも業界標準のサードパーティ製ツールと完全互換です。

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 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 まで電話してください。

2.2 製品アップグレード

サイプレスは無料で PSoc Creator の定期的なアップグレートと拡張バージョンを提供します。アップグレードは CD-ROM の形で代理店から入手できます。また www.cypress.com のソフトウェアセクション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

開発キットは Digi-Key、Avnet、Arrow、Future の各社より入手可能です。サイプレスのオンライン ストアでも開発キット、C コンパイラ、PSoC プロジェクトを失敗なく開発するために必要なアクセサリをお求めいただけます。サイプレスのオンライン ストアについては www.cypress.com/shop/ をご覧ください。入手可能な品目を見るには、製品カテゴリの下にある **Programmable System-on-Chip** をクリックしてください。

3. 文書の構成

本書の以下のセクションは次のようなトピックを含んでいます。

- セクション B: CPU システム (29 ページ)
- セクション C: システムワイドリソース (57 ページ)
- セクション D: デジタル システム (95 ページ)
- セクション E: アナログ システム (215 ページ)
- セクション F: プログラムおよびデバッグ (297 ページ)

3.1 大セクション

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。

- セクション : トップレベルアーキテクチャ、使い方、当該分野についての表記法と概要を記載し、製品の構成と構築に関する情報を読者に提供します。
- 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。実装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。
- PSoC® 4 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 マニュアルでまとめられるデバイスのレジスタの詳細な情報を提供します。これらは別文書です。

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

レジスタ表記法は「PSoC 4100M/4200M Family: PSoC 4 Registers TRM」で詳細に説明されます。

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

記号	測定単位
°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. 略号

記号	測定単位
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)
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)
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)
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)
DNL	differential nonlinearity (微分非直線性)
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)
ECO	external crystal oscillator (外部水晶発振器)

表 3-2. 略号 (続き)

記号	測定単位
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)
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)
IMO	internal main oscillator (内部主発振器)
INL	integral nonlinearity (積分非直線性)
I/O	input/output (入力／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)
L2CAP	logical link control and adaptation protocol (論理リンク制御および適応プロトコル)
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 (メモリマップ入力／出力)
MOSI	master-out-slave-in (マスター アウト スレーブ イン)

表 3-2. 略号 (続き)

記号	測定単位
MSb	most significant bit (最上位ビット)
MSB	most significant byte (最上位バイト)
PC	program counter (プログラム カウンター)
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)
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)
RW	read/write (読み出し／書き込み)
SAR	successive approximation register (逐次比較レジスタ)
SC	switched capacitor (スイッチト 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)
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)

表 3-2. 略号 (続き)

記号	測定単位
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 (監視システム 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)
TD	transaction descriptors (トランザクション記述子)
UART	universal asynchronous receiver/transmitter (汎用非同期レシーバ／トランスミッタ)
UDB	universal digital block (ユニバーサル 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
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 (外部リセット、アクティブ LOW)

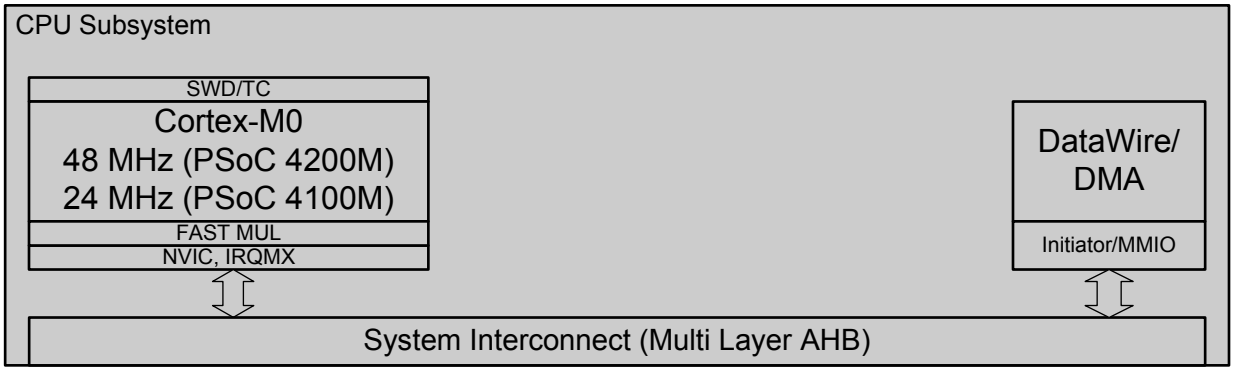
セクション B: CPU システム

このセクションは次の章を含みます。

- Cortex-M0 CPU (31 ページ)
- DMA コントローラー モード (37 ページ)
- 割り込み (47 ページ)

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 ビットの CPU です。高い効率の 3 段パイプラインおよび 4GB の固定メモリがあり、ARMv6-M Thumb 命令セットに対応します。Cortex-M0 はシングル サイクル乗算命令およびレイテンシの短い割り込みサービス ルーチン (ISR) の処理にも対応します。

Cortex-M0 プロセッサは CPU コアに密にリンクされるその他のコンポーネントを含みます。これら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、デバッグ機能です。

このセクションでは 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。

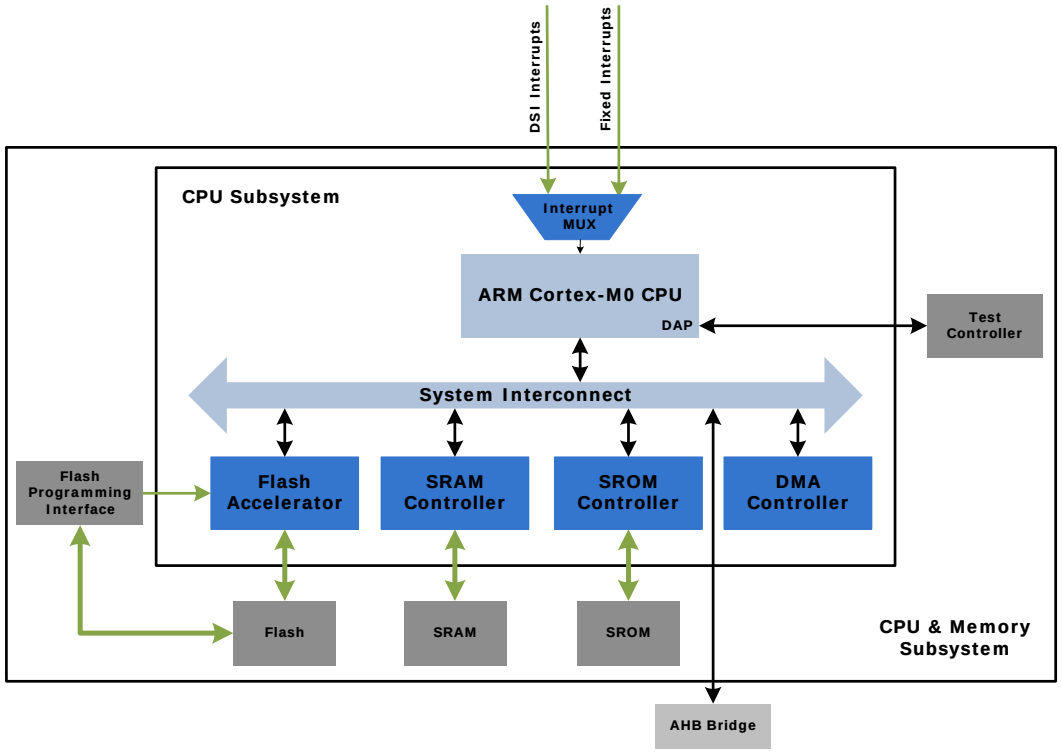
4.1 特長

PSoC 4 Cortex-M0 は下記の特長があります。

- 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にできます。
- 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ちます。
- 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します。
- NVIC ユニットは突発的で確定的な割り込みと例外に対応します。
- 機能強化されたデバッグ機能
  - シリアル ワイヤー デバッグ (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. PSoC 4 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 (34 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注意してください。

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 コードの実行可能領域。データをここに置くことも可能。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	データの実行可能領域。コードをここに置くことも可能
0x40000000 ～ 0x5FFFFFFF	ペリフェラル	すべてのペリフェラル レジスタ。この領域外ではコードは実行不可
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 には 16 ビットと 32 ビットの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	R13 はスタック ポインタ レジスタ。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
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
CONTROL	RW	0	CONTROL 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

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	APSE	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 = 予約済み 2 = NMI 3 = HardFault 4 – 10 = 予約済み 11 = SVCall 12、13 = 予約済み 14 = PendSV 15 = SysTick 16 = IRQ0 … 47 = IRQ31

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 47 ページの割り込み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。スレッド 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。これで 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実装します。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、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	キャリー付加算
ADD{S}	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}	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	バイト逆符号ハーフワード
RORS	右回転
RSBS	逆減算
SBCS	キャリー付き減算
SEV	イベントの送信
STM	複数のレジスタの保存、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}	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 TEST
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

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

PSoC 4 Cortex-M0 は、ワードの最下位バイトが最下位アドレスに、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 の内部クロック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) コンパレーターを実装しています。

5. DMA コントローラー モード

DMA コントローラーはデータ ワイヤー (DW) およびダイレクト メモリ アクセス (DMA) 機能を提供します。DMA コントローラーは以下の特長があります :

- チャネルごとに 4 段階の優先度
- チャネルごとに 3 種類の動作モード
- チャネルごとのピンポン形式で動作する 2 組のディスクリプタ
- 8 ビット、16 ビット、32 ビットのデータ幅

DMA コントローラーは 3 つの動作モードに対応します。単一のトリガー信号に対する動作が異なります。動作モードは以下の通りです

- トリガー当たり単一のデータ要素の転送を実行
- トリガー当たり全部のディスクリプタを実行
- トリガー当たり全部のディスクリプタ チェーンを実行

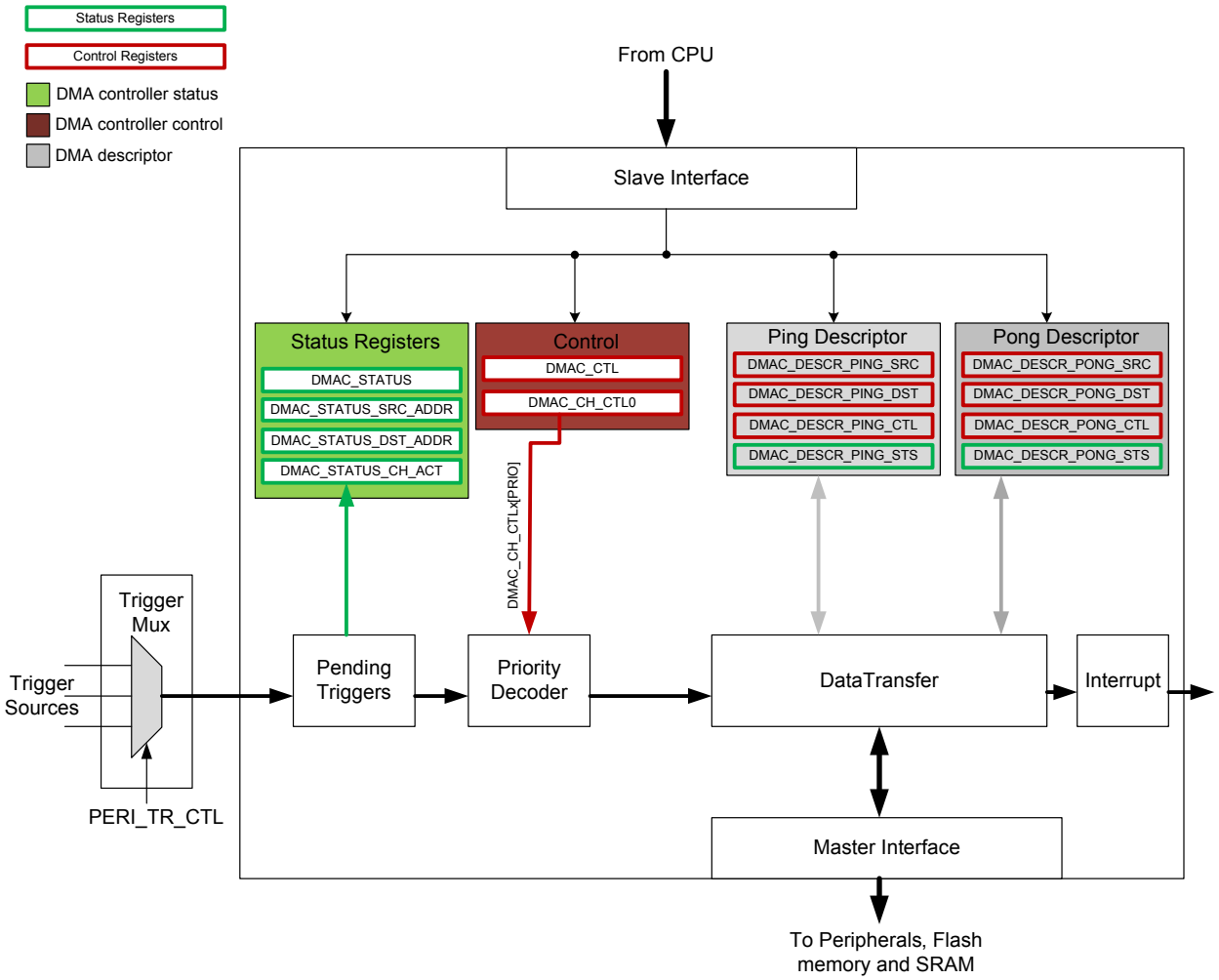
転送元および転送先のアドレス位置や転送サイズなどのデータ転送の詳細は、ディスクリプタ構造体によって指定されます。各チャネルは独立したディスクリプタ構造体を持ちます。

DMA コントローラーはアクティブ、スリープモードで機能を提供し、ディープスリープとハイバネートのモードで使用できません。

5.1 ブロック図の解説

図 5-1 にブロック レベルの DMA コントローラーの概要を示します。

図 5-1. DMA コントローラーのブロック図



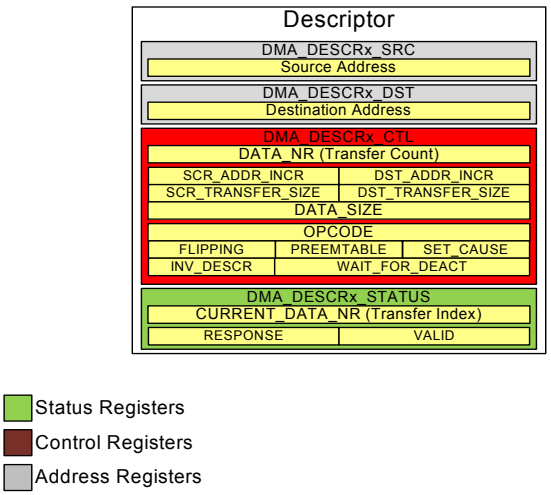
- トリガー マルチプレクサ ブロックは DMA コントローラーの外部にあり、各チャネルを特定の 1 つのトリガーに接続します。選択されたトリガー ラインが論理「1」になると、トリガーが起動されたことを示し、チャネルデータ転送が始まります。
- トリガー保留ブロックは、保留中ビットの形態でローカルにトリガーを保存することで、起動されたトリガーを監視します。データ転送エンジンは 1 回に 1 つのチャネルにしか対応できません。複数のチャネル トリガーが同時に起動される可能性がある状況で不可欠な機能です。このブロックでレベル トリガーかパルス トリガーを選択します。
  保留中のトリガーはステータス レジスタ (DMAC_STATUS_CH_ACT) に記録されます。
- 優先度デコーダーは起動されたトリガーで最高の優先度チャネルを判定します。優先度はチャネル制御レジスタ (DMAC_CH_CTL) の PRIO フィールドにより、各チャネルに設定されます。
- データ転送エンジンは転送元の位置から転送先の位置までのデータ転送の責任があります。アイドルの時、データ転送エンジンは最高の優先度で起動されたチャネルを受け入れられる状態にあります。ディスクリプタはデータ転送のコンフィギュレーションを指定します。
- ピンとポンのディスクリプタ : 各チャネルにはピンとポンの 2 組のディスクリプタ があります。ディスクリプタは、転送コンフィギュレーションを記述する 4 つの 32 ビットのワード レジスタです。ディスクリプタは、転送元アドレス、転送先アドレス、使用される転送モードおよびその他転送に関連する事項からなります。詳細は 39 ページディスクリプタを参照ください。
- マスターI/F は AHB-Lite バス マスターです。これで DMA コントローラーはAHB-Liteを起動して、転送元の位置から転送先の位置までデータを転送します。
- スレーブ I/F はAHB-Liteバス スレーブです。これでPSoC の主な CPU は DMA コントローラーの制御／ステータス レジスタおよびディスクリプタ構造体にアクセスします。
- 割り込み回路は各チャネルの割り込みステータスを含みます。

5.2 ディスクリプタ

1 つのチャネル内の転送元と転送先の間でデータ転送はディスクリプタとして記述、設定されます。DMA の各チャネルにはピンとポンというディスクリプタが 2 つあります。ディスクリプタは関連チャネル内での転送のコンフィギュレーションを含む 4 つの 32 ビットのレジスタです。

図 5-2 にディスクリプタ構造体を示します。

図 5-2. ディスクリプタ構造体



5.2.1 アドレス コンフィギュレーション

図 5-3 に 1 回の転送のアドレス コンフィギュレーションのディスクリプタ設定を示します。

転送元と転送先のアドレス ：転送元と転送先のアドレスはディスクリプタの各レジスタに設定されます。これらは転送のために、転送元と転送先の開始位置アドレスを指定します。ディスクリプタが 1 つの要素を転送するように設定される場合、このフィールドはデータ要素の転送元／転送先のアドレスそのものとなります。ディスクリプタがインクリメンタルモードの転送元アドレスあるいは転送先アドレスまたはその両方で複数の要素を転送するように設定される場合、このフィールドは転送される最初の要素のアドレスを保持します。

データ数 (DATA_NR): これは転送回数を数えるパラメーターです。DATA_NR は、ディスクリプタが完了する前に転送される要素の数を指定する 16 ビットの値です。典型的な使用例では、この設定は転送のバッファ サイズです。

転送元アドレス インクリメント (SCR_ADDR_INC): これは転送される各データ要素の間で転送元アドレスをインクリメントするかどうかを決める制御レジスタのビット設定です。データの転送元がバッファであり、各転送要素をメモリの後続の位置からフェッチすることが必要な場合、この機能を有効にします。この場合、転送元アドレスの設定はベース アドレスのみ設定し、後続の転送はインクリメントします。

転送先アドレス インクリメント (SCR_ADDR_INC): これは転送される各要素の間で転送先アドレスをインクリメントするかどうかを決める制御レジスタのビット設定です。データの転送先がバッファであり、各転送要素がメモリの後続の位置まで転送することが必要な場合、この機能を有効にします。この場合、転送先アドレスのレジスタはベース アドレスのみ設定し、後続の転送はインクリメントします。

無効ディスクリプタ (INV_DESCR): このビットがセットされる時、ディスクリプタはすべてのデータ要素を転送し、ディスクリプタの VALID ビットをクリアすることで無効となります。この機能は DMA_DESCRx_STATUS レジスタの VALID ビットに影響を与えます。

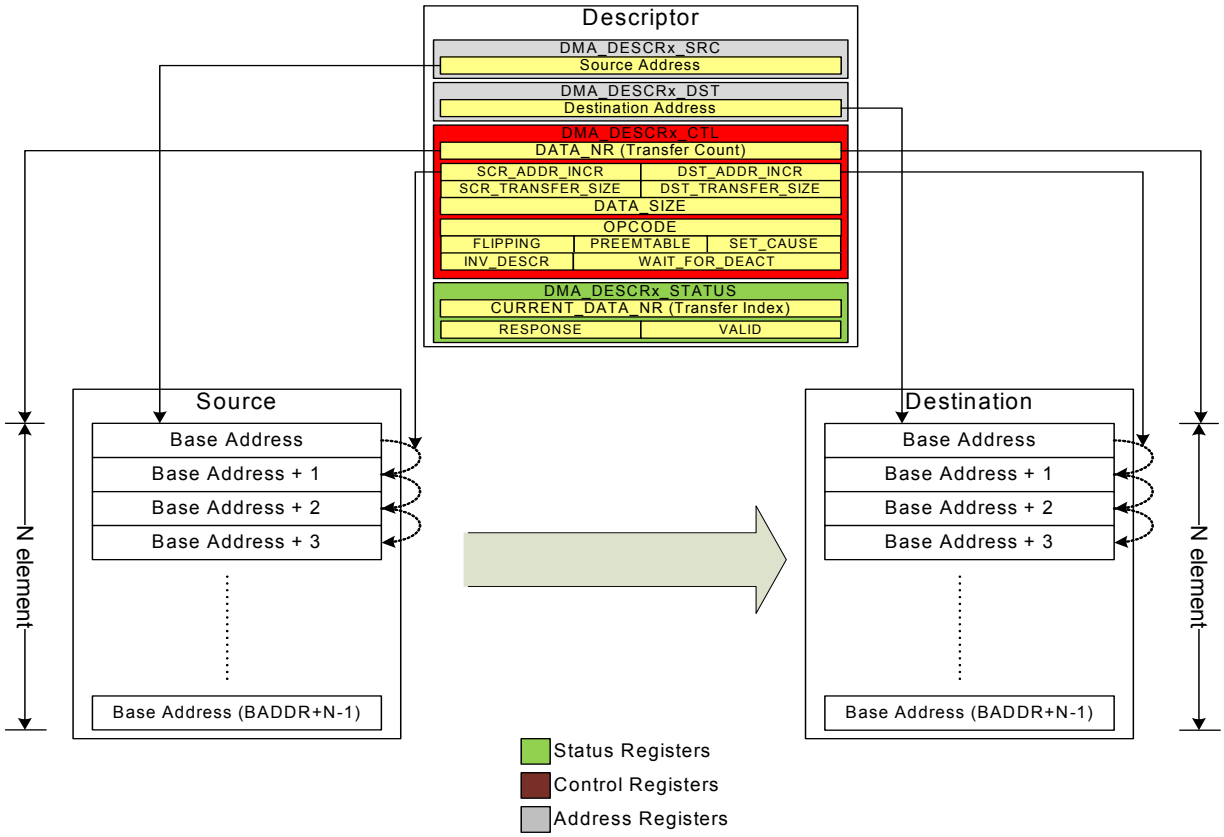
プリエンプタブル (PREEMPTABLE): これを無効にすると、動作モードで設定された現在の転送を何の通知もなく完了させることが可能です。これを有効にすると、動作モードで設定された現在の転送は優先度の高い DMA チャネルにより先取り、割り込みをされます。このチャネルが先取りされた場合保留ビットがセットされ、次回に最高の優先度で動作します。

割り込み要因セット (SET_CAUSE): ディスクリプタはすべてのデータ要素の転送を完了した後、割り込み要求を発生します。この割り込み要求はすべての DMA チャネル間に共用されます。

トリガー タイムアウト (WAIT_FOR_DEACT): これを用いて、DMA の入力トリガーのタイプを設定します。レベル センシティブ トリガーはDMAを再びトリガーする前に待機することができます。このフィールドは次の 4 つの設定があります。

- 0 - パルス トリガー
- 1 - 4 SYSCLK サイクルを待機するレベルセンシティブ : レベル トリガー信号が 4 サイクルで検出された後、DMA 転送が開始されます。
- 2 - 8 SYSCLK サイクルを待機するレベルセンシティブ : レベル トリガー信号が 8 サイクルで検出された後、DMA 転送が開始されます。
- 3 - パルス トリガーは無期限に非アクティブ化を待機します。トリガー信号が非アクティブになった後、DMA 転送が開始されます。

図 5-3. DMA 転送 : アドレス コンフィギュレーション



5.2.2 転送サイズ

転送用の転送ワード幅はディスクリプタのパラメーターの転送／データ サイズにより設定することが可能です。この設定は転送元の転送サイズ、転送先の転送サイズおよびデータサイズにわかれます。データ サイズ パラメーター (DATA_SIZE) は転送のためのバスの幅を設定します。SCR_TRANSFER_SIZE および DST_TRANSFER_SIZE により設定された転送元と転送先のサイズは DATA_SIZE の値または 32 ビットの値を有することができます。DATA_SIZE は 32 ビット、16 ビットまたは 8 ビットに設定することができます。

表 5-1 に転送サイズ設定の組み合わせとその動作を説明します。

表 5-1. 転送サイズの設定

DATA_SIZE	SCR_TRANSFER_SIZE	DST_TRANSFER_SIZE	説明
8 ビット	8 ビット	8 ビット	データの操作なし
8 ビット	32 ビット	8 ビット	転送元より送信される上位の 24 ビットは欠落
8 ビット	8 ビット	32 ビット	転送先で上位の 24 ビットに 0 を補填
8 ビット	32 ビット	32 ビット	転送元より送信される上位の 24 ビットが欠落しその部分に 0 を補填
16 ビット	16 ビット	16 ビット	データの操作なし
16 ビット	32 ビット	16 ビット	転送元より送信される上位の 16 ビットは欠落
16 ビット	16 ビット	32 ビット	転送先で上位の 16 ビットに 0 を補填
16 ビット	32 ビット	32 ビット	転送元より送信される上位の 16 ビットが欠落しその部分に 0 を補填
32 ビット	32 ビット	32 ビット	データの操作なし

5.2.3 ピン／ポン ディスクリプタ

全てのチャネルに、転送設定を有するピンとポンのディスクリプタがあります。アクティブのディスクリプタは、個別のチャネル制御レジスタ (DMAC_CH_CTL) の PING_PONG ビットに示されます。ピンとポンのディスクリプタの機能はディスクリプタのリンクリストを作成することです。これは CPU の介入なく、異なる転送コンフィギュレーションに遷移するのに役立ちます。その上、2 つのディスクリプタがあると、CPU はポン レジスタがアクティブの時にピン レジスタを自由に変更することができます。逆の場合も同様です。

ディスクリプタのフリッピング ビットが有効になると、次にカウンターパートのディスクリプタに接続します。このフィールドはオペコード 2 の転送モードと共に使用されます。従って、フリッピング ビットがピン ディスクリプタ内で有効にされオペコード 2 用に設定された場合、チャネルはピン ディスクリプタの次にポン ディスクリプタを自動的に実行します。オペコード 0 またはオペコード 1 のコンフィギュレーションの場合、ポン ディスクリプタを開始するのに新規のトリガーが必要です。

ピン ポンディスクリプタの用法は動作モードに関連します。

5.2.4 動作モード

ディスクリプタ実行中のチャネルの動作はオペコードの設定により定義されます。DMA コントローラーの各チャネルで使用可能なオペコードは 3 つあります。

5.2.4.1 データ ワイヤー機能 (オペコード 0)

オペコード 0 が設定される時、データ ワイヤー機能が実行されます。データ ワイヤは、単一のデータ要素を各トリガー信号で転送元の位置から転送先の位置まで転送します。この機能は、転送元と転送先のインクリメントなどのディスクリプタのその他の設定と共に使用することが可能です。

図 5-4 はデータ ワイヤー転送の典型的な使用例を示します。ここでは、ADC のデータ レジスタは転送元であり、SRAM のメモリ位置は転送先です。転送先はバッファのように、インクリメンタルアドレスにすることが可能です。この場合においても、転送は 1 つのトリガー当たり 1 つのデータ要素に限定されるので、バッファは 1 つのトリガー当たり 1 つのデータ要素で満たされます。トリガーは AD 変換の終了信号のこともあります。トリガーが受け入れられた場合、転送エンジンはデータを ADC 位置からロードし、下位の 16 ビットを転送先のメモリ位置に保存します。連続トリガーは同様の動作をもたらします。

図 5-4. 転送元から転送先への単純な DW 転送

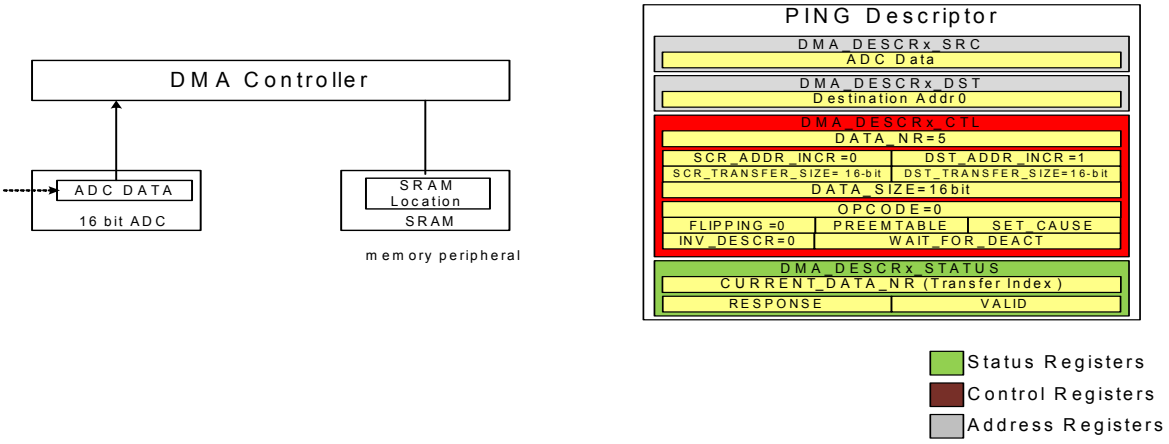
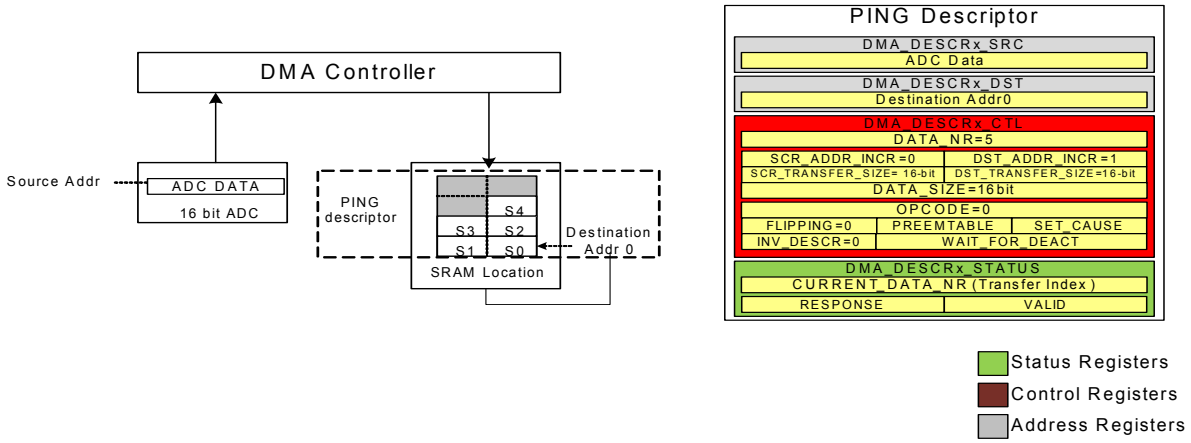


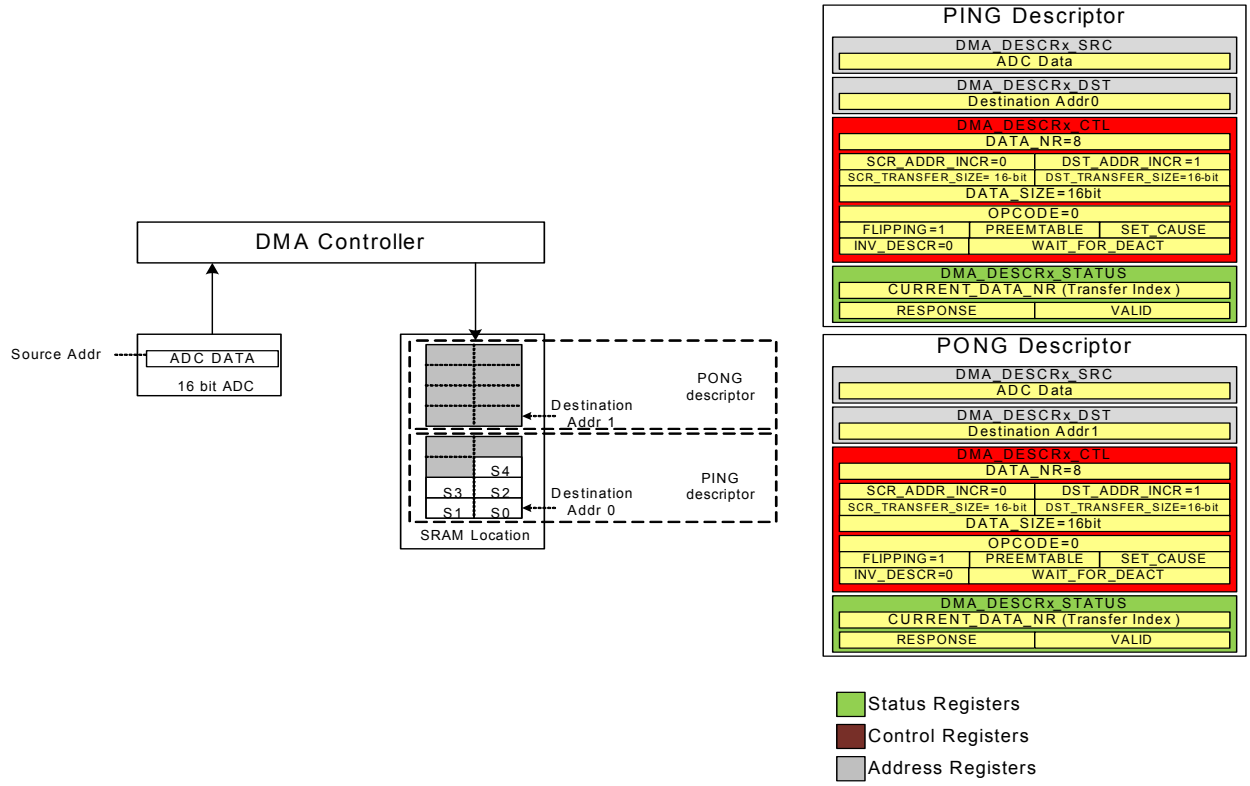
図 5-5 はデータ転送がADCデータ レジスタとバッファの間にあるケースを示します。使用ケースは、ADC である転送元からデータを取り出しながら、転送先をインクリメントするために設定されるピン レジスタを示します。トリガーが受け入れられた場合、転送エンジンはデータを ADC 位置からロードし、それを図 5-5 の「S0」のブロックで表示された転送先アドレス 0 のメモリ位置に保存します。後続のトリガーは、S1、S2、S3 などで表示された転送先アドレス 0 からの連続位置に、ピン ディスクリプタ バッファ サイズ (DATA_NR field) が満たされたまで ADC データを続けて保存します。

図 5-5. 転送先アドレス インクリメント機能による DW 転送



同様の使用ケースを図 5-6 に示します。これはピンとポンのディスクリプタの使用例です。ピン ディスクリプタの完了後、コントローラーはポン ディスクリプタを実行します。従って、2 つのバッファ転送が次々と実行されます。ただし 1 つのトリガー当たり 1 つの要素で転送が完了することに注意してください。

図 5-6. フリッピング機能を使用する DW 転送



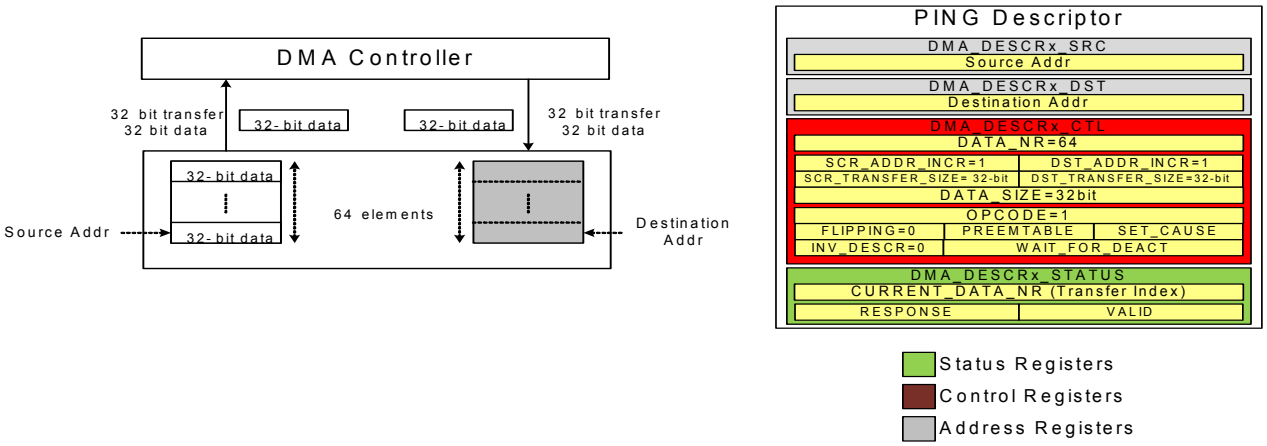
5.2.4.2 DMA 機能 (オペコード 1 と 2)

DMA は転送元の位置から転送先の位置まで複数のデータ要素を転送します。DMA 転送は descriptor_CTRL のオペコード 1 およびオペコード 2 により設定されます。

オペコード 1 の場合、コントローラーは単一のトリガーで全部のディスクリプタを実行します。この機能はメモリ間のバッファ転送に役に立ちます。

図 5-7 は転送元バッファの中味全てを、転送先のバッファへ転送するオペコード 1 の転送を示します。全部の転送は単一のピン ディスクリプタの一部であり、単一のトリガーで完了します。

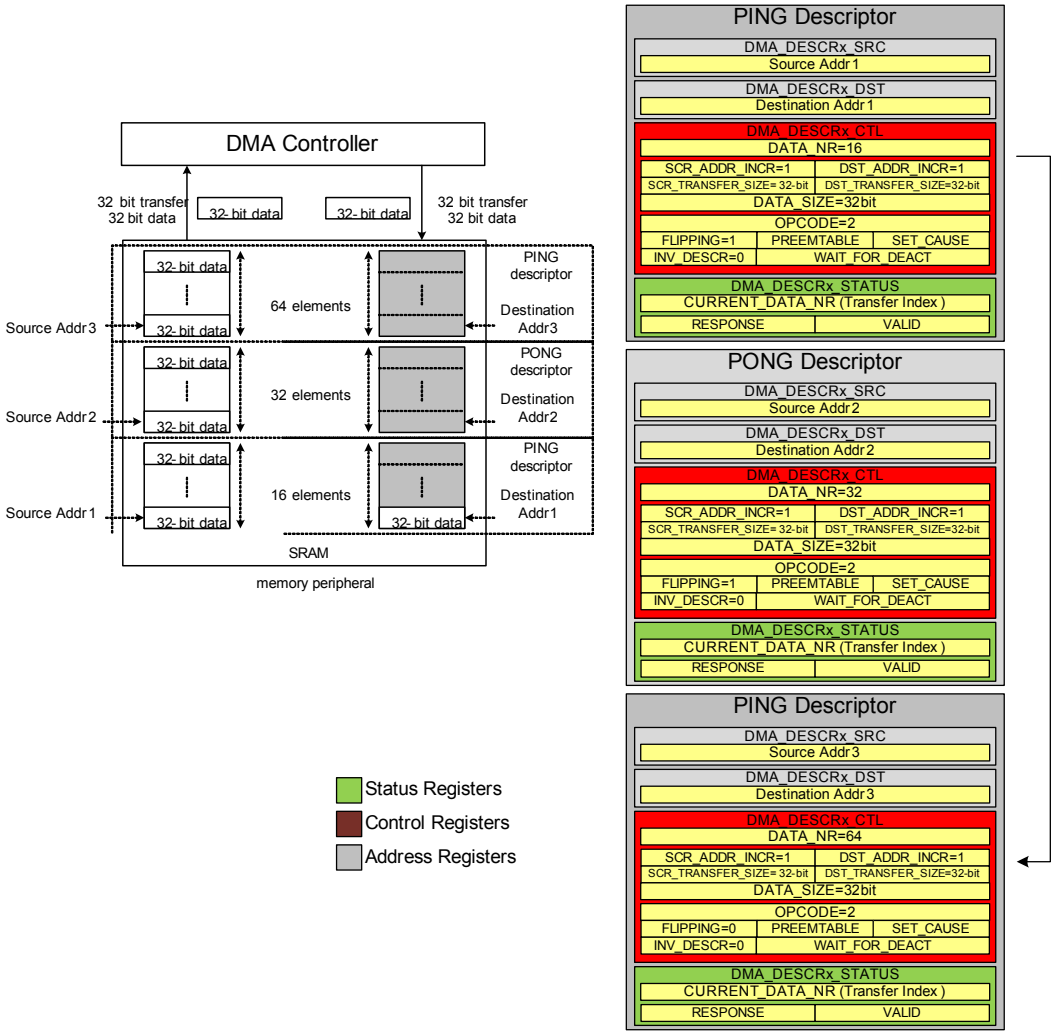
図 5-7. オペコード 1 の DMA 転送の例



オペコード 2 はフリッピング フィールドと共に使用されます。フリッピングを有効にしたピン ディスクリプタでオペコード 2 を使用する場合、単一のトリガーでピン ディスクリプタを実行し、自動的にポン ディスクリプタに遷移し実行を完了することができます。ポン ディスクリプタがオペコード 2 と共に使用される場合、ピンとポン間のサイクルは 1 つのディスクリプタが無効になるか CPU により変更されるまで継続します。

図 5-8 はピンとポンのディスクリプタがオペコード 2 の動作として設定され、ピン レジスタの 2 回目の反復でフリッピングが CPU により無効にされるケースを示します。

図 5-8. オペコード 2 の DMA 転送の例



5.2.5 動作とタイミング

DMA ブロックでは次の動作段階があります。

- チャネル設定段階: チャネルがディスクリプタ コンフィギュレーションに基づいて設定されます。この段階はチャネルの選択とディスクリプタからのデータのフェッチも含みます。この段階には SYSCLK の 8 クロック サイクルが通常かかります。
- データ転送段階 : データ転送エンジンはマスター インターフェースを用いて、転送元からデータをロードします。この段階において、インターフェースはデータロード待機の状態になることがあります。データロードの後、データ転送エンジンはマスター インターフェースを使用して、データを転送先に保存します。

  この段階には SYSCLK の 3 クロック サイクルが通常かかります。データロードとデータ保存待機の状態のため、または AHB インターフェースを使用するマスターに保留中の優先度の高いものがある場合、サイクル数は増加します。
  - データ ロード待機状態 : これは転送元アドレスからロードされるデータの待機状態です。
  - データ保存待機状態 : これは転送先に保存されるデータの待機状態です。
- ステータス ライトバック段階 : データ転送エンジンがデータ転送を反映するために、チャネルのディスクリプタ構造体を更新し、ディスクリプタ ステータス レジスタにそれを保存します。この段階には SYSCLK の 1 クロック サイクルがかかります。

データワイヤー タイミング .

1 回のデータワイヤー転送にかかる総サイクル数 = 8 (チャネル設定) + 3 (データ転送段階) + データ ロード待機状態 + データ保存待機状態 + 1 ステータス ライトバック

DMA タイミング .

DMA 転送の場合、チャネル設定段階とステータス ライトバック段階は単一のトリガーで転送される最初のデータ要素についてのみ発生します。後続のデータ要素についてはデータ転送段階だけがあります。

複数のデータ要素を DMA 転送する最初のデータ要素にかかる総サイクル数 = 8 (チャネル設定) + 3 (データ転送段階) + データ ロード待機状態 + データ保存待機状態 + 1 (ステータス ライトバック)

複数のデータ要素を DMA 転送する後続のデータ要素にかかる総サイクル数 = 3 (データ転送段階) + データ ロード待機状態 + データ保存待機状態

N 個のデータ要素のデータ転送にかかる総サイクル数 = 12 + データ ロード待機状態 + データ保存待機状態 + (N - 1)* (3 + データ ロード待機状態 + データ保存待機状態)

5.3 レジスター覧

レジスタ名	コメント	特長
DMAC_CTL	ブロック制御	DMA コントローラーのイネーブル ビット
DMAC_STATUS	ブロック ステータス	DMA コントローラーのステータス情報を提供
DMAC_STATUS_SRC_ADDR	現在の転送元アドレス	現在ロードされている転送元アドレスの詳細を提供
DMAC_STATUS_DST_ADDR	現在の転送先アドレス	現在ロードされている転送先アドレスの詳細を提供
DMAC_STATUS_CH_ACT	チャネル起動のステータス	ソフトウェアはこのフィールドを読み込んで、保留中のすべてのチャネルの情報 (保留中かデータ転送エンジンで処理中か) を取得
DMAC_CH_CTLx	チャネル制御レジスタ	チャネル x についてチャネル イネーブル、ピン / ポンおよび優先度設定の機能を提供
DMAC_DESCRx_PING_SRC	ピンディスクリプタの転送元アドレス	チャネル x 用の転送元の位置のベース アドレス
DMAC_DESCRx_PING_DST	ピンディスクリプタの転送先アドレス	チャネル x 用の転送先の位置のベース アドレス
DMAC_DESCRx_PING_CTL	ピンディスクリプタの制御ワード	ピン ディスクリプタのすべての制御設定
DMAC_DESCRx_PING_STATUS	ピンディスクリプタのステータス ワード	有効性、応答およびリアルタイム Data_NR のインデックス ステータス
DMAC_DESCRx_PONG_SRC	ポンディスクリプタの転送元アドレス	チャネル x 用の転送元の位置のベース アドレス
DMAC_DESCRx_PONG_DST	ポンディスクリプタの転送先アドレス	チャネル x 用の転送先の位置のベース アドレス
DMAC_DESCRx_PONG_CTL	ポンディスクリプタの制御ワード	ポン ディスクリプタのすべての制御設定
DMAC_DESCRx_PONG_STATUS	ポンディスクリプタのステータス ワード	有効性、応答およびリアルタイム Data_NR のインデックス ステータス
DMAC_INTR	割り込みレジスタ	
DMAC_INTR_SET	割り込みセットレジスタ	読み込む時、このレジスタは割り込み要求レジスタを反映
DMAC_INTR_MASK	割り込みマスク	INTR レジスタの対応するフィールド用のマスク
DMAC_INTR_MASKED	マスクされた割り込みレジスタ	読み込む時、このレジスタは割り込み要求レジスタにマスク レジスタを適用した値を反映。このレジスタによりソフトウエアは、ロード動作を 2 回 (割り込み要因とマスクのロードを各 1 回) 行わず、1 回ですべてのマスクされた割り込み要因ステータスを読み込むことが可能。ファームウェア開発が容易になる

6. 割り込み

PSoC® 4 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 CPU の外部周辺機器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。PSoC 4 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 テーブルを持つています。

6.1 特長

PSoC4 が対応する割り込み機能は次の通り :

- 32 個の割り込みに対応
- 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を示します
- 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できます
- 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

6.2 動作原理

図 6-1. PSoC 4 割り込みブロック図

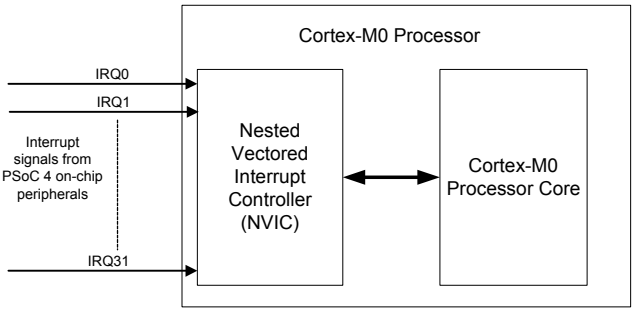


図 6-1 に割り込み信号と Cortex-M0 CPU との関係を示します。PSoC4 は 32 の割り込みがあり、この割り込み信号は NVIC によって処理されます。NVIC は個々の割り込みの有効制御、優先度管理、CPU コアとの通信を担当します。CPU の外部周辺機器によって生成される割り込みと違い、例外は CM0 のコアで生成されるイベントの一部であるため、図 6-1 に表示されていません。

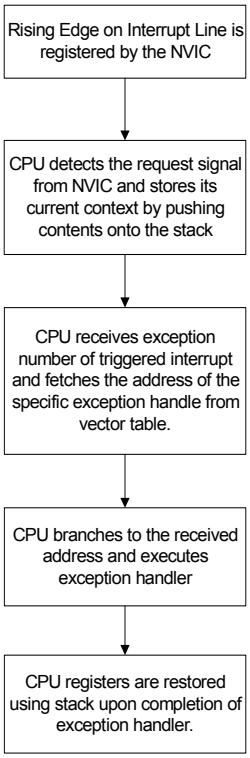
6.3 割り込みと例外の動作

6.3.1 PSoC4 における割り込みと例外処理

割り込みまたは例外イベントがトリガーされると、次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、保留状態になっています。
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 6-1 に示すように、PSoC4 の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 6-2. 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の優先度を評価して、CPU に最も優先度の高い割り込みの例外番号を送信します。こうし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の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 に使用されます。

6.3.2 レベルおよびパルス割り込み

PSoC4 NVIC は、割り込みライン (IRQ0 から IRQ31 まで) についてレベル信号とパルス (エッジ) 信号の両方に対応しています。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 6-3. レベル割り込み

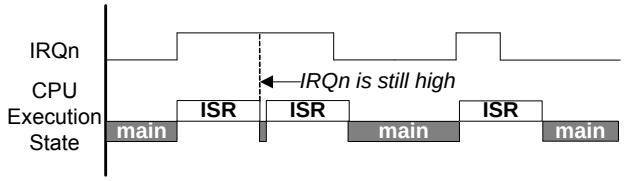


図 6-4. パルス割り込み

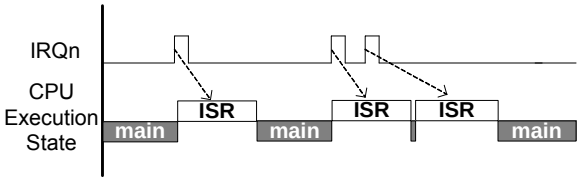


図 6-3と図 6-4はレベルとパルス割り込みの動作を示しています。以下のイベント シーケンスは、割り込みラインが最初に非アクティブ (論理 LOW) であるとして、レベルとパルス割り込みの処理を示しています。

1. 割り込み信号の立ち上がりエッジで、NVIC は割り込み要求をレジスタします。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 6-4 を参照)。
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 6-3 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。

6.3.3 例外ベクタ テーブル

例外ベクタ テーブル (表 6-1) は PSoC4 の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 は例外番号に基づいて適切なアドレスをフェッチします。

表 6-1. PSoC4 の例外ベクタ 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)[a]
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	監視呼び出し (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)	…
47	外部 割り込み (IRQ31)	コンフィギュレーション可能 (0 ～ 3)	Base_Address + 0xBC

a. 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注意してください。

表 6-1 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。PSoC 4 では、ベクタ テーブルは、フラッシュ メモリ (0x00000000 のベース アドレス) または SRAM (0x20000000 のベース アドレス) のいずれかに配置することができます。コンフィギュレーションはCPUSS_CONFIG レジスタの VECT_IN_RAM ビットフィールド (ビット 0) に書き込むことによって行われます。VECT_IN_RAM ビット フィールドが 「1」である時、CPU は例外ハンドラー アドレスを SRAM ベクタ テーブルからフェッチします。このビット フィールドが「0」(リセット ステート) である時、フラッシュ メモリのベクタ テーブルが例外アドレス フェッチのために使用されます。SRAM にベクタ テーブルを設定するには、デバイスブート コードの一部として VECT_IN_RAM ビットフィールドを設定する必要があります。SRAM にベクタ テーブルを移動させることの利点は、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。

例外ソース (1 から 15 までの例外番号) は 6.4 例外ソースで説明します。表 6-1 で予約済みと記載されている例外は PSoC 4 で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 から 47 までの例外番号) は 6.5 割り込みソースで説明します。

6.4 例外ソース

この節では表 6-1 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。

6.4.1 リセット例外

デバイス リセット は PSoC4 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 の優先度によって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 6-1 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注意してください。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行われます。

6.4.2 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 の優先度によって常に有効にされています。PSoC4 で NMI 例外をトリガーするものは 3 つあります、

- **ハードウェア信号 (ユーザーNMI 例外) による NMI 例外:** PSoC4 はデジタル信号を用いて NMI 例外をトリガーする手段を提供しています。このデジタル信号は表 6-3

で irq_out[0] と呼ばれています。irq_out[0] によってトリガーされた NMI 例外は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) で示された NMI ハンドラーを実行します。

- **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 (CM0_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 PSoC4 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、書き込みアクセスができません。

6.4.3 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、未定義命令の実行や無効なメモリ アドレスにアクセスすることを含む異なる種類のフォルト条件のためのキャッチ 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。

6.4.4 監視呼び出し (SVCall) 例外

監視呼び出し (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。これは監視呼び出し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する監視コールを発行することを可能にします。PSoC4 CM0 は SVCall 例外に関連しないシステム呼び出し NMI 例外のための特権モードを使用することに注意してください。特権モードの詳細については 79 ページのチップの動作モードを参照してください。PSoC4 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がありません。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 によって処理されるのを待ちます。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。

6.4.5 PendSV 例外

PendSV は SVCall に類似する監視コールの例外です。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 CM0_ICSR の PENDSVSET ビットをセットすることでトリガーされます。このビットを立てると PendSV 例外が保留状態に入り、CPU が処理するのを待ちます。PendSV 例外の保留状態は、割り込み制御状態レジスタである CM0_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 (CM0_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。

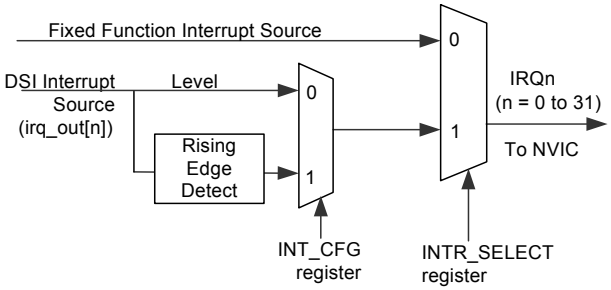
6.4.6 SysTick 例外

PSoC4 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持っています。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維持の目的のための簡単な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 (CM0_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 CM0_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 CM0_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。

6.5 割り込みソース

PSoC4 は周辺機器からの 32 個の割り込み (IRQ0 - IRQ31 または例外番号 16 - 47) に対応しています。各割り込みのソースは表 6-3 に記載されています。PSoC4 は 32 本の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。図 6-5 は割り込みソースの多重化機構を示しています。各割り込みは 2 つのソースがあります : 固定機能割り込みソースと DSI 割り込みソース。CPUSS_INTR_SELECT レジスタはこれらのソースの間を選択するために使用されます。

図 6-5. 割り込みソースの多重化



(irq_out[n]) と名付けられた DSI 割り込み信号は容易にアクセス可能ではありません。しかし PSoC Creator IDE が DSI 割り込みパスからデジタル信号の配線を実行することでタスクを簡素化することに注意してください。手動で DSI パスを設定する必要がありません。

固定機能割り込みは、TCPWM、シリアル通信ブロック および CSD 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。生成された固定機能割り込みは、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 レジスタを割り込みを生成した条件を検出するために ISR で読み出します。固定機能割り込みは通常レベル割り込みであり、割り込みをクリアするために周辺ステータス レジスタをISRで読み出すことが必要です。ステータス レジスタを ISR で読み出さないと、割り込みがアサートされたままになり、ISR が継続して実行されます。

割り込みソースの第 2 のカテゴリーは DSI 割り込み信号です。Cortex M0 には 8 DSI チャネルがあり、32 の割り込みソースに拡大するために、各チャネルは 4 にデマルチプレクスされています。UDB からのデジタル出力またはピン上のデジタル入力信号などのようなチップ上の任意のデジタル信号は DSI 割り込みソースとして配線することができます。このことにより割り込みソースを柔軟に選択することができます。図 6-5 に示したように、立ち上がりエッジの検出回路を介して、DSI 信号を配線するオプションもあります。このエッジ検出回路は DSI ライン上の立ち上がりエッジを、2 システムクロック幅を持つパルス信号に変換します。これにより割り込みが DSI ライン上の立ち上がりエッジのたびにトリガーされることを保証します。NVIC に適合したレベル割り込み信号を生成することができない割り込みソースに便利です。UDB_INT_CFG レジスタは、直接 DSI パスとエッジ検出パスを選択するのに使用されます。

DSI 割り込みチャネルはデマルチプレクスされるため、最大 DSI 割り込み数は 8 に限られます。

表 6-2. PSoC4 割り込みソース一覧

割り込み	Cortex-M0 例外番号	固定機能割り込みソース	DSI 割り込みソース
NMI (49 ページの例外ソースを参照)	2	SYS_REQ	udb.interrupts[0]:4
IRQ0	16	GPIO 割り込み - ポート 0	udb.interrupts[0]:0
IRQ1	17	GPIO 割り込み - ポート 1	udb.interrupts[1]:0
IRQ2	18	GPIO 割り込み - ポート 2	udb.interrupts[2]:0
IRQ3	19	GPIO 割り込み - ポート 3	udb.interrupts[3]:0
IRQ4	20	GPIO 割り込み - ポート 4	udb.interrupts[4]:0
IRQ5	21	GPIO 割り込み - すべてのポート[a]	udb.interrupts[5]:0
IRQ6	22	LPCOMP （低消費電力コンパレータ）	udb.interrupts[6]:0
IRQ7	23	WDT (ウォッチドッグ タイマー)	udb.interrupts[7]:0
IRQ8	24	SCB0 (シリアル通信ブロック 0)	udb.interrupts[0]:1
IRQ9	25	SCB1 (シリアル通信ブロック 1)	udb.interrupts[1]:1
IRQ10	26	SCB2 (シリアル通信ブロック 2)	udb.interrupts[2]:1
IRQ11	27	SCB3 (シリアル通信ブロック 3)	udb.interrupts[3]:1
IRQ12	28	CTBm 割り込み (すべての CTBm)	udb.interrupts[4]:1
IRQ13	29	DMA 割り込み	udb.interrupts[5]:1
IRQ14	30	SPCIF 割り込み	udb.interrupts[6]:1
IRQ15	31	SRSS LVD 割り込み	udb.interrupts[7]:1
IRQ16	32	SAR (逐次比較型 ADC)	udb.interrupts[0]:2
IRQ17	33	CSD0 (CapSense)	udb.interrupts[1]:2

表 6-2. PSoC4 割り込みソース一覧

割り込み	Cortex-M0 例外番号	固定機能割り込みソース	DSI 割り込みソース
IRQ18	34	CSD1 (CapSense)	udb.interrupts[2]:2
IRQ19	35	TCPWM0 (タイマー / カウンター /PWM 0)	udb.interrupts[3]:2
IRQ20	36	TCPWM1 (タイマー / カウンター /PWM1)	udb.interrupts[4]:2
IRQ21	37	TCPWM2 (タイマー / カウンター /PWM2)	udb.interrupts[5]:2
IRQ22	38	TCPWM3 (タイマー / カウンター /PWM3)	udb.interrupts[6]:2
IRQ23	39	TCPWM4 (タイマー / カウンター /PWM4)	udb.interrupts[7]:2
IRQ24	40	TCPWM5 (タイマー / カウンター /PWM5)	udb.interrupts[0]:3
IRQ25	41	TCPWM6 (タイマー / カウンター /PWM6)	udb.interrupts[1]:3
IRQ26	42	TCPWM7 (タイマー / カウンター /PWM7)	udb.interrupts[2]:3
IRQ27	43	CAN0 割り込み (PSoC 4200M のみ)	udb.interrupts[3]:3
IRQ28	44	CAN1 割り込み (PSoC 4200M のみ)	udb.interrupts[4]:3
IRQ29	45	<DSI のみ >	udb.interrupts[5]:3
IRQ30	46	<DSI のみ >	udb.interrupts[6]:3
IRQ31	47	<DSI のみ >	udb.interrupts[7]:3

a. ポート 5、ポート 6、およびポート 7 は専用ポート割り込みベクタ番号がありません。ベクタ IRQ5 を共用しています。

GPIO 割り込みの詳細については 59 ページの I/O システムをご覧ください。

6.6 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が柔軟にできます。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。PSoC4 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。

PSoC4 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妨害 (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 CM0 CPU は最大 4 つの例外の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する場合、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 49 ページの例外ソースに説明されています。

優先度レジスタ (CM0_IPR) に書き込むことによって、32 個の割り込み (IRQ0 - IRQ31) の優先度を設定することができます。表 6-3 で示すように、これは 8 つの 32 ビットレジスタのグループであり、レジスタ 1 つは 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 6-3. 割り込み優先度レジスタ 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

6.7 割り込みの有効化

NVIC はソフトウェアで 32 個の割り込みを個別に有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。割り込みイネーブル セット レジスタ (CM0_ISER) と割り込みイネーブル クリア レジスタ (CM0_ICER) はそれぞれの割り込みを有効にするために使用されます。これらのレジスタは 32 ビット幅であり、各々の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 6-4 はこの 2 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注意してください。

表 6-4. 割り込みイネーブル レジスタ

レジスタ	動作	ビット値	コメント
割り込みイネーブルセットレジスタ (CM0_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 (CM0_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

CM0_ISER と CM0_ICER レジスタは割り込み (IRQ0 - IRQ31) のみに適用可能です。これらのレジスタは例外番号 1 から 11 までを有効にするためには利用できません。49 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 6-1 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 が最高優先度で、7 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。

6.8 例外状態

各例外は次のいずれかの状態にあります。

表 6-5. 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で、保留中ではない例外が無効になっているか、有効な例外が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 (CM0_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。

- CM0_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 は例外ハンドラーを (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注意してください。
- CM0_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。
- CM0_ICSR の ISRPENDING ビット (ビット 22) は NVIC が生成した割り込み (IRQ0 から IRQ31 へ) が保留状態にあることを示します。

6.8.1 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 32 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 (CM0_ISPR) と割り込み保留クリア レジスタ (CM0_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のに使用されます。これらのレジスタは 32 ビット幅であり、各々のビットは同じ番号の割り込みに対応しています。表 6-6 にこの 2 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注意してください。

表 6-6. 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	コメント
割り込み保留セット レジスタ (CM0_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 (CM0_ISPR)	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 CM0_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。

CM0_ISPR と CM0_ICPR レジスタは 32 個のペリフェラル割り込み (16 ～ 47 の例外番号) のみに使用されます。これらのレジスタは例外番号 1 から 11 までを保留するためには利用できません。49 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。

6.9 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 つの 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、例外要求が発生する場合、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 プッシュの動作の後、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 プッシュ / ポップ動作に使用されます。詳細は 31 ページの Cortex-M0 CPU 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として動作します (infocenter.arm.com/help/topic/com.arm.doc.ddi0419c/index.html)。

6.10 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 PSoC4 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する時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 WFI (割り込み待機) 命令は、スリープ、ディープ スリープおよびハイバネート 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 81 ページの電力モード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について 3 分類あります :

- アクティブ、ディープスリープおよびハイバネート モード (GPIO 割り込み、低消費電力コンパレータ) で使用可能の固定機能の割り込みソース。
- 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の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、シリアル通信ブロック割り込み)
- アクティブ モードのみで使用可能の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)

6.11 例外の初期化とコンフィギュレーション

この節は PSoC4 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されたように、フラッシュ メモリまたは SRAM のいずれかにベクタ テーブルを設置することです。このコンフィギュレーションは CPUSS_CONFIG レジスタの VECT_IN_RAM ビット フィールドへ「1」(SRAM ベクタ テーブル) または「0」(フラッシュ ベクタ テーブル) のいずれか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施されています。

   アプリケーションが動的にベクタアドレスを変更する必要があるなら、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位置している場合には、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 は初期設定で SRAM にベクタ テーブルを設置しています。
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。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 6-1 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. 52 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. 52 ページの割り込みの有効化で説明したように例外を有効にします。

6.12 レジスタ

表 6-7. レジスター覧

レジスタ名	説明
CM0_ISER	割り込みイネーブルセット レジスタ
CM0_ICER	割り込みイネーブルクリア レジスタ
CM0_ISPR	割り込み保留セット レジスタ
CM0_ICPR	割り込み保留クリア レジスタ
CM0_IPR	割り込み優先度レジスタ
CM0_ICSR	割り込み制御状態レジスタ
CM0_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
CM0_SCR	システム制御レジスタ
CM0_CCR	構成および制御レジスタ
CM0_SHPR2	システム ハンドラー優先度レジスタ 2
CM0_SHPR3	システム ハンドラー優先度レジスタ 3
CM0_SHCSR	システムハンドラー制御および状態レジスタ
CM0_SYST_CSR	Systick 制御およびステータス
CPUSS_CONFIG	CPU サブシステム コンフィギュレーション
CPUSS_SYSREQ	システム要求レジスタ
CPUSS_INTR_SELECT	割り込み多重選択レジスタ
UDB_INT_CFG	UDB サブシステム割り込みコンフィギュレーション

6.13 関連文書

- ARMv6-M Architecture Reference Manual ー 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。

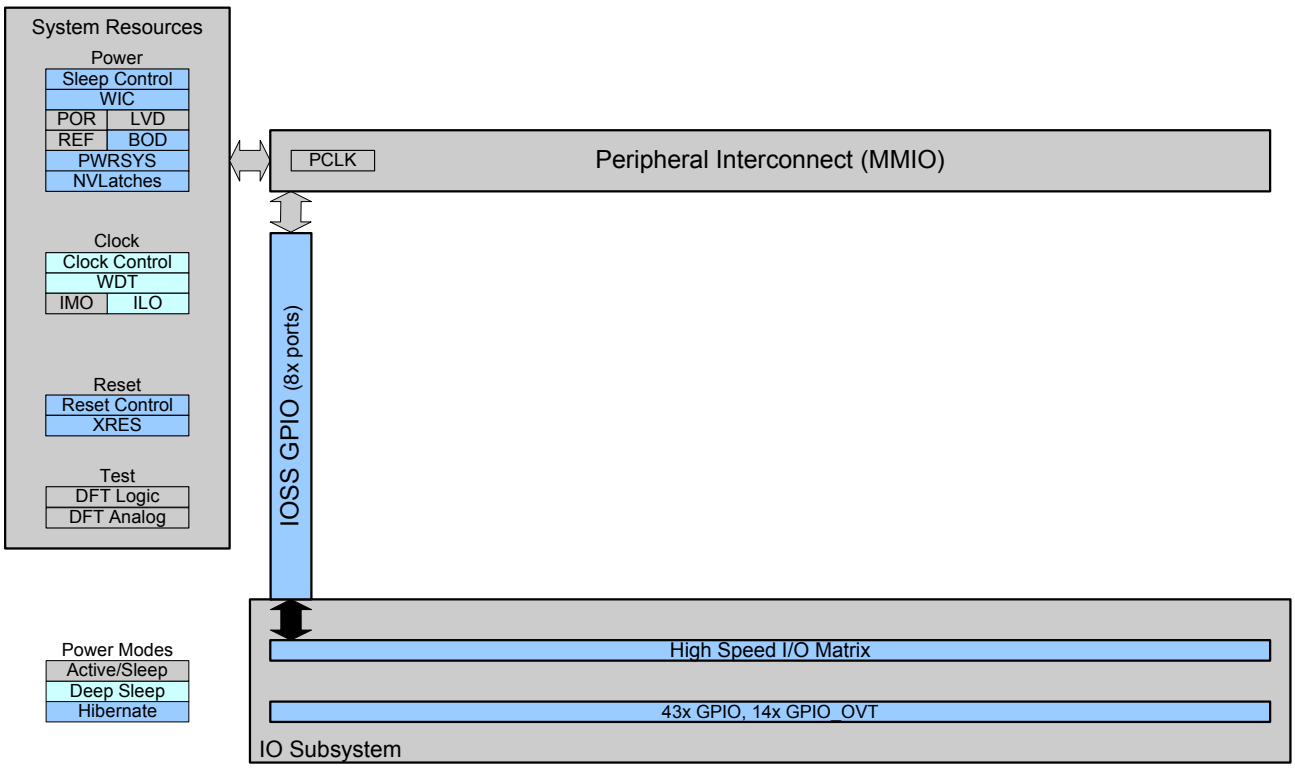
セクション C: システムワイドリソース

このセクションは次の章を含みます。

- I/O システム (59 ページ)
- クロック供給システム (67 ページ)
- 電源と電圧監視 (75 ページ)
- チップの動作モード (79 ページ)
- 電力モード (81 ページ)
- ウォッチドッグ タイマー (85 ページ)
- リセット システム (89 ページ)
- デバイス セキュリティ (93 ページ)

トップ レベル アーキテクチャ

システムワイドリソースブロック図



7. I/O システム

本章では、PSoC® 4 の I/O システム (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 の汎用 I/O (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 4100M/4200M ファミリは、8 ポートに配置されている最大 55 本の GPIO を備えており、6 本の過電圧耐性ピンが含まれています。

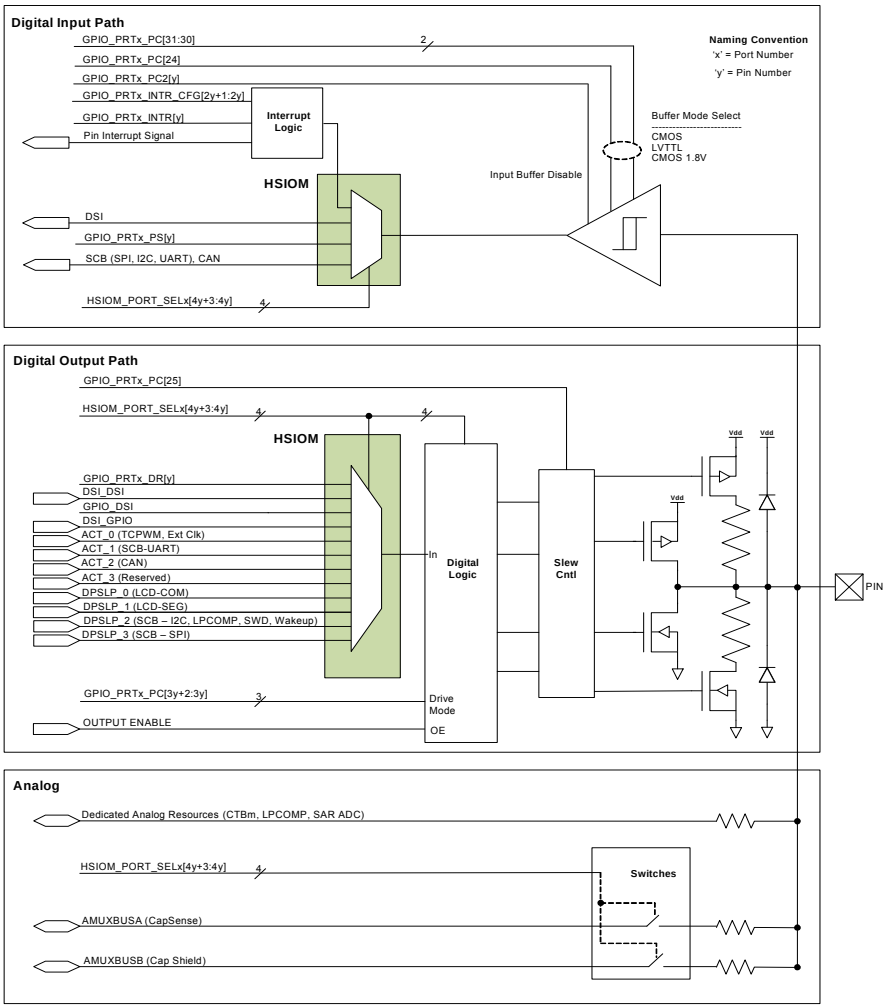
7.1 特長

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
- CapSense のサポート
- 独立したポート読み出し (PS) とポート書き込み (DR) データ レジスタによる、読み出し、修正、書き込み操作によるエラー回避
- 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
- CMOS および低電圧 LVTTL 入力バッファ モードを選択可能
- I2C に準拠するよう 6 本の過電圧耐性 (OVT_GPIO) ピンを搭載

7.2 ブロック図

図 7-1. GPIO のブロック図



注 : この図に示される GPIO の機能がすべてのピンで利用できるわけではありません。詳細は表 7-4 を参照ください。

7.3 GPIO 駆動モード

各 I/O は、表 7-1 に列挙される 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。図 7-2 は 8 駆動モードの簡略化したピン配置図です。

PSoC 4 の GPIO をコンフィギュレーションするための 2 つのポート コンフィギュレーション レジスタ : ポート コンフィギュレーション レジスタ (GPIO_PRTx_PC) およびポート セカンダリ コンフィギュレーション レジスタ (GPIO_PRTx_PC2)。すべての PSoC 4 デバイスは、専用の GPIO_PRTx_PC と GPIO_PRTx_PC2 レジスタを持つポートを備えています。

GPIO_PRTx_PC は、ポートのプロパティをコンフィギュレーションするために使用されます。

- 各ピンの出力駆動モード (3 ビットがピンに対する駆動モードを選択)
- ポート全体のスルーレート (63 ページのスルーレート制御 を参照)
- ポート全体の入力閾値選択 (63 ページの CMOS LVTTL レベルの制御 を参照)

GPIO_PRTx_PC2 は、GPIO_PRTx_PC でコンフィギュレーションされた駆動モードに関係なく、ポート上の各ピンの入力バッファを有効にするために使用されます。アナログ信号がピンに存在している時は、ビットを「1」にセットして入力バッファを無効にする必要があります。

図 7-2. I/O 駆動モードのブロック図

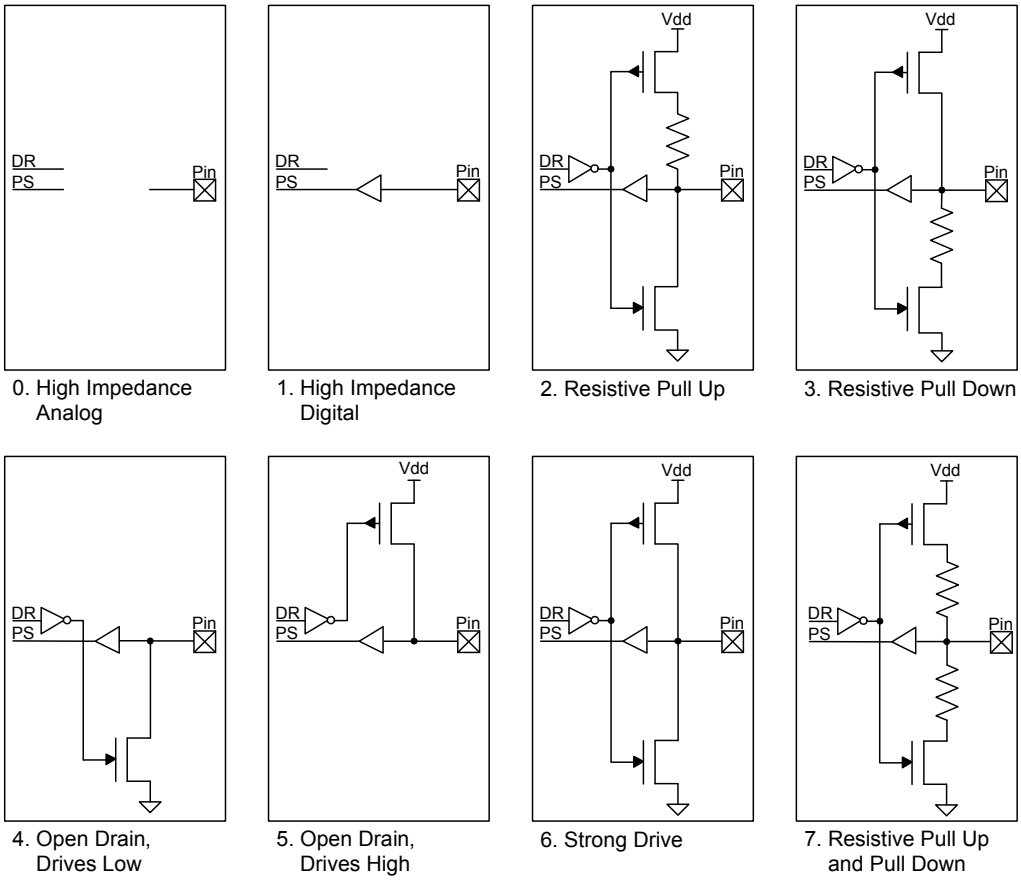


表 7-1. 駆動モード設定

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)		
	ハイインピーダンスアナログ	0	High Z	High Z
	ハイインピーダンス 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
	ストロングドライブ	6	強 1	強 0
	抵抗プルアップとプルダウン	7	弱 1	弱 0

表 7-2. 入力バッファ無効 (ポート コンフィギュレーション 2)

GPIO_PRTx_PC2 (「x」はポート番号、「y」はピン番号を示す)		
ビット	名称	説明
7:0	INP_DIS	ポート制御駆動モードとは無関係に入力バッファを無効にする。アナログ信号をピンで扱う時は、このビットをセットする必要がある

7.3.1 ハイインピーダンスアナログ

ハイインピーダンス 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が電流をデジタル入力バッファに流入させることを防ぎます。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。ハイ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 をハイ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。

7.3.2 ハイインピーダンスデジタル

ハイインピーダンス 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。

7.3.3 抵抗プルアップまたは抵抗プルダウン

抵抗モードは、Vdd または Vss との間に直列抵抗を挿入し、他方でストロング駆動をします。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 LOW の場合は抵抗プルアップ、オープン ドレイン HIGH の場合は抵抗プルダウンを使用します。

7.3.4 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 モードでは、Vdd または Vss との間が高インピーダンスになり、他方がストロング駆動になります。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。

7.3.5 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。

7.3.6 抵抗プルアップおよび抵抗プルダウン

このモードは7.3.3 抵抗プルアップまたは抵抗プルダウンで説明した駆動モードと同様です。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、バスが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。

7.4 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 GPIO_PRTx_PC[25] ビットを用いてコンフィギュレーションすることができます。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。高速のスルー レートは 1 MHz より高い信号に対して推奨されます。低速スルーレートにより EMI とクロストークが低減されるため 、低速オプションは速度が重要ではない (一般に 1MHz 未満の) 信号に推奨されます。

7.5 CMOS LVTTL レベルの制御

I/O ピンは 2 つの電圧レベルで動作できます。これらのレベルは、GPIO_PRTx_PC[24] ビットへ書き込むことで選択することが可能です。

入力レベルは、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 CMOS モードで動作します。ポートを LVTTL モードに再コンフィギュレーションするために、このビットをセットします。

CMOS は殆どの場合で使用可能であるのに対して、LVTTL はより低い電圧レベルで動作するカスタム インターフェースの要求に使用されます。モードに対応する入力電圧の閾値 (VIH と VIL) については、デバイス データシートを参照してください。

デバイスのポート 6 は過電圧耐性 (OVT) ピンを備えています。普通の GPIO ピンと同様ですが以下の特徴が追加されています。

- 過電圧許容
- より良いプルダウン駆動強度を提供
- I^2C にコンフィギュレーションされたシリアル通信ブロック (SCB) のラインを GPIO-OVT ピンに配線すると以下の I^2C 仕様に準拠 :
  - 高速モード ホット スワップ
  - 高速モード プラス IOL 仕様
  - 高速モードと高速モード プラス ヒステリシスおよび最小立ち下がり時間の仕様

仕様の詳細についてはデバイス データシートをご覧ください。

7.6 高速 I/O マトリックス

高速 I/O マトリックス (HSIOM) は、PSoC 内のリソースへ GPIO を配線する高速スイッチのグループです。リソースは CapSense、TCPWM、I2C および CPU です。HSIOM はピンに対してアクティブおよびディープスリープ電源ドメインのソースを選択します。HSIOM_PORT_SELx は GPIO の配線を制御する 32 ビット幅レジスタです。各レジスタは 1 つのポートを制御し、4 つの専用ビットはポート内の各 GPIO に割り当てられます。これにより、GPIO 配線は 16 までの異なるオプションから選択できます。この選択は表 7-3 に一覧表示されているように、異なるピン機能を提供します。

表 7-3. 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
	DR_DSI	1	出力がファームウェアで制御されるが、OE は DSI から制御
	DSI_DSI	2	出力と OE は両方とも DSI から制御
	DSI_DR	3	出力が DSI から制御されるが、OE がファームウェアで制御
	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)
	ACT_0	8	特定ピンのアクティブ ソース #0 (TCPWM、EXT CLOCK)
	ACT_1	9	特定ピンのアクティブ ソース #1 (SCB-UART)
	ACT_2	10	特定ピンのアクティブ ソース #2 (CAN - PSoC 4200M でのみ)
	ACT_3	11	予約済み
	DPSLP_0	12	特定ピンのディープ スリープ ソース #0 (LCD - COM)
	DPSLP_1	13	特定ピンのディープ スリープ ソース #1 (LCD - SEG)
	DPSLP_2	14	特定ピンのディープ スリープ ソース #2 (SCB-I2C、SWD、LPCOMP、スリープ復帰)
	DPSLP_3	15	特定ピンのディープ 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徴の詳細については、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。

UART と SPI 入力 (SCB モード) がピンに接続している場合、SCB はデジタル出力バッファの駆動モードを制御することに注意してください。例えば、SCB ブロックは、入力インピーダンスを高くするために UART Rx ピンで出力バッファを無効にします。これは 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。

7.7 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 設定については表 7-3 を参照してください。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注意してください。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 レジスタに加え、ポートで特定の GPIO の出力データをセット／クリアするために他の 2 つのレジスタ (GPIO_PRTx_DR_SET および GPIO_PRTx_DR_CLR) も提供しています。さらに GPIO_PRTx_INV レジスタは、特定の GPIOの出力データを反転するためにも使用することができます。

GPIO_PRTx_PS は、ポート I/O パッド レジスタであり、読み出すと GPIO の状態を提供します。このレジスタに書き込んでも何の反応もありません。

これらのレジスタの詳細情報については PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM を参照してください。

7.8 アナログ IO

LPCOMP、SARMUX や CTBm 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を提供します。専用ピンは性能を向上させ、これらのアナログ リソースを使用する際に他のピンよりも優先させます。PSoC 4 の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、ハイ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 7-1 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、ハイ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て (表 7-1 を参照)、HSIOM_PORT_SELx レジスタを用いて AMUXBUS に配線する必要があります (表 7-3)。

7.9 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 駆動のピンを選択するために使用されます。詳細は 271 ページの LCD ダイレクト ドライブをご覧ください。

7.10 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) 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 7-3 には CapSense に必要な GPIO と HSIOM 設定を示しています。詳細については 283 ページの CapSense を参照してください。

表 7-4. 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	ハイ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	ハイ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)	ハイ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)	ハイ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)	ハイ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

7.11 I/O ポートの再コンフィギュレーション

GPIO_PRTx_PC および HSIOM_PORT_SELx レジスタの値を変更することで、実行時に駆動モードと GPIO を再コンフィギュレーションすることができます。ピンがデジタル周辺機器に直接接続されている場合、ピンの再コンフィギュレーション中にピンの状態を保持するように注意してください。ポートがデータ レジスタによって駆動されている場合、ピンの状態は自動的に保持されます。しかしポートがバイパスされ、DSI によって駆動された場合、再コンフィギュレーションを開始する前に現行の値をデータ レジスタ (GPIO_PRTx_DR) に読み書きする必要があります。ポート

のコンフィギュレーション中に、現行のコンフィギュレーションは以下のように保存されます。

1. ソフトウェアで GPIO ピンの状態 (GPIO_PRTx_PS) を読み出します。
2. GPIO_PRTx_PS の値をデータ レジスタ (GPIO_PRTx-_DR) に書き込みます。
3. データ レジスタ (GPIO_PRTx_DR) によってピンを駆動するために、HSIOM_PORT_SELx での該当するフィールドを変更します。

注 : GPIO が AMUXBUS A/AMUXBUS B に接続するようにコンフィギュレーションされた場合、DSI 信号は GPIO スイッチを制御するために使用されます。そのため、DSI モードと AMUXBUS モードの間で GPIO を再コンフィギュレーションして使用することは不可能です。

7.12 電源投入時の I/O 状態

初期設定では、電源投入時にすべての GPIO はハイ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。チップが電源供給され、関連するレジスタに書き込むことで、GPIO は必要なアプリケーションに応じてコンフィギュレーションすることができます。

7.13 省電力モードでの動作

GPIO はスリープ モード中に、現行のピン状態を維持します。スリープ モードでは、すべての GPIO はアクティブであり、CapSense、TCPWM や I2C などのアクティブな周辺機器により駆動されます。

ディープスリープ モードでは、ディープ スリープ ドメインにある周辺機器に接続しているGPIOは依然として動作します (表 7-4 を参照)。

ハイバネート モードとストップ モードでは、すべての I/O ピンはラッチされ凍結状態になります。詳細は 81 ページの電力モードを参照してください。

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 I/O をハイ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。

7.14 GPIO 割り込み

本節では、PSoC 4 GPIO の割り込みの機能について説明します。

7.14.1 特長

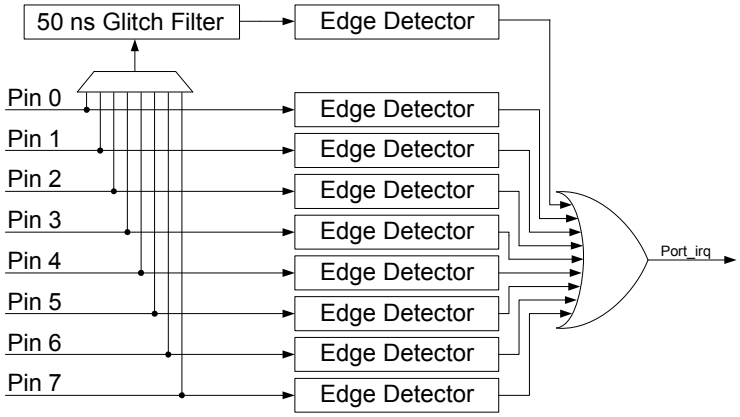
GPIO の割り込みは以下の特長を持っています。

- 各ポート インターフェースでの 8 本のピンはすべて、割り込みおよび附随する割り込みベクタを搭載
- ピン ステータス ビットは、ピンの電圧レベルに応じて割り込みソースを容易に決定します。
- 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を処理
- ピン割り込みを個別に有効化
- レジスタへ読み書きするための AHB インターフェース
- ポートでのすべての GPIO から生成された単一のポート割り込み要求 (PIRQ) 信号を割り込みコントローラーに送信

7.14.2 割り込みコントローラー ブロック図

各ポートは個別の割り込み要求と関連する割り込み要求 (IRQ) ベクタおよび割り込みサービス ルーチン (ISR) を持っています。さらに各ポートで 1 本のピンを選択して、50ns のグリッチ フィルタを通って配線し、ポートに対してグリッチ耐性割り込みを形成することができます。詳細を図 7-3 に示します。

図 7-3. 割り込みジェネレーター



7.14.3 機能とコンフィギュレーション

ポートの各ピンは、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割り込みを生成するために、GPIO_PRTx_INTR_CFG に書き込むことにより独立してコンフィギュレーションすることができます。レベルトリガの割り込みには対応していません。GPIO_PRTx_INTR_CFG は、特定のチャネルをグリッチ フィルタに配線し、9 番目のグリッチ耐性割り込みを生成するためにも使用されます。

GPIO 割り込みが割り込み有効ポート ピン上の信号によってトリガーされると、GPIO_PRTx_INTR レジスタ (ポート割り込みステータス レジスタ) が更新されます。ファームウェアはこのレジスタを読み出して、どの GPIO が割り込みをトリガーしたかを判定します。そして、ファームウェアは対応するビットに「1」を書き込んで IRQ ビットをクリアします。

さらに、対応するポートで割り込みが発生していると同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可能性があります。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対応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ように推奨します。

詳細は PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM の GPIO_PRTx_INTR_CFG および GPIO_PRTx_INTR を参照してください。

7.15 入力と出力の同期

デジタル入力／出力信号に対して、I/O は内部クロックまたはクロックのようなデジタル信号との同期を提供します。デフォルトでは、同期に HFCLK を使用しますが、他の任意のクロックを使用することもできます。

この機能およびクロックとリセット機能は、UDB ポート アダプタと I/O ブロックを組み合わせることにより実装されます。ポート アダプタの詳細については137 ページのユニバーサル デジタル ブロック (UDB) を参照してください。

7.16 レジスタ

表 7-5. 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 ピンの論理を読み出すために使用
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

注 : レジスタ名での「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。

8. クロック供給システム

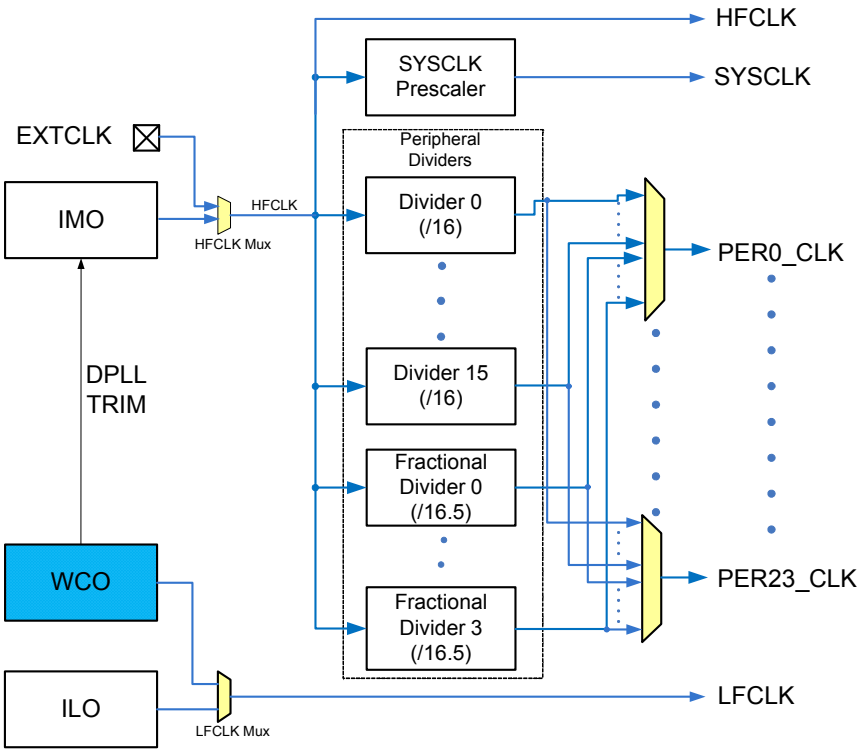
PSoC® 4 のクロック システムは以下のクロック リソースを含んでいます。

- 2 つの内部クロック源 :
  - 3MHz ～ 48MHz 内部主発振器 (IMO) ±2% の許容誤差 (全周波数範囲に渡り、調整あり)
  - 32kHz 内部低速発振器 (ILO) ±60% の許容誤差 (調整あり)
- 2 つの外部クロック源 :
  - 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)
  - 外部の 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 までの高周波数クロック (HFCLK)
- ILO または WCO からクロック供給された低周波数クロック (LFCLK)
- HFCLK からクロック供給され、PSoC 4200M では最大 48MHz まで、PSoC 4100M では 24MHz までのシステム クロック (SYSCLK) 専用のプリスケーラ
- 16 個の 16 ビット ペリフェラル クロック分周器
- 4 個の分数分周器 (精度の良いクロック生成のため)
- 24 個のデジタルおよびアナログ ペリフェラル クロック

8.1 ブロック図

図 8-1 は、PSoC 4 デバイスの一般的なクロック供給システムを示しています。

図 8-1. クロック供給システムのブロック図



デバイスのこの4種類のクロック源は図 8-1の左側に配置されています。HFCLK マルチプレクサは外部クロック源 (EXTCLK) または IMO から HFCLK の供給源を選択します。SYSCLK プリスケーラは SYSCLK を生成し、ペリフェラル分周器は個々のペリフェラル クロックを生成します。

8.2 クロック源

8.2.1 内部主発振器 (IMO)

内部主発振器は外部コンポーネントなくに動作し、3 ～ 48MHz までの周波数範囲を 1MHz 単位で設定し安定したクロックを生成します。周波数は、CLK_IMO_TRIM2 レジスタ に周波数を設定し、CLK_IMO_TRIM1 レジスタに IMO の調整値を設定してから、PWR_BG_TRIM4 と PWR_BG_TRIM5 レジスタにバンドギャップ リファレンスのトリムを設定することで選択されます。CLK_IMO_TRIM2 で設定した周波数は IMO の出力周波数を決定します。表 8-1 は IMO の出力周波数に対応する設定を示しています。CLK_IMO_TRIM2 内の周波数設定と共に、CLK_IMO_TRIM1、PWR_BG_TRIM4、および PWR_BG_TRIM5 内の調整値を読み込む必要があります。各 PSoC デバイスは、データシート仕様に合わせるために、製造中に測定されたそれぞれ IMO トリムの設定があります；調整値は SFLASH 内の製造コンフィグレーション データに格納されています。それらは、ユーザーが選択した周波数に応じた TRIM 値です。SFLASH 内の TRIM 値は、該当するトリム レジスタ – CLK_IMO_TRIM1、PWR_BG_TRIM4、および PWR_BG_TRIM5 に格納されます。所望のコンフィギュレーションを得るために、起動時にはこれらの値を読み込むことができます。ファームウェアは、これらの調整値を取得し、実行中にデバイスのコンフィギュレーションを変更して周波数を変更することができます。

表 1: IMO 周波数設定

CLK_IMO_TRIM2						周波数 (MHz)
ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	
0	0	0	0	1	1	3
0	0	0	1	0	0	4
0	0	0	1	0	1	5
0	0	0	1	1	0	6
0	0	0	1	1	1	7
0	0	1	0	0	0	8
0	0	1	0	0	1	9
0	0	1	0	1	0	10
0	0	1	0	1	1	11
0	0	1	1	0	0	12
0	0	1	1	1	0	13
0	0	1	1	1	1	14
0	1	0	0	0	0	15
0	1	0	0	0	1	16
0	1	0	0	1	0	17
0	1	0	0	1	1	18
0	1	0	1	0	0	19
0	1	0	1	0	1	20
0	1	0	1	1	0	21
0	1	0	1	1	1	22
0	1	1	0	0	0	23
0	1	1	0	0	1	24
0	1	1	0	1	1	25
0	1	1	1	0	0	26
0	1	1	1	0	1	27
0	1	1	1	1	0	28
0	1	1	1	1	1	29
1	0	0	0	0	0	30
1	0	0	0	0	1	31
1	0	0	0	1	0	32
1	0	0	0	1	1	33
1	0	0	1	0	1	34
1	0	0	1	1	0	35
1	0	0	1	1	1	36
1	0	1	0	0	0	37
1	0	1	0	0	1	38
1	0	1	0	1	0	39
1	0	1	0	1	1	40
1	0	1	1	1	0	41
1	0	1	1	1	1	42
1	1	0	0	0	0	43
1	1	0	0	0	1	44
1	1	0	0	1	0	45
1	1	0	0	1	1	46
1	1	0	1	0	0	47
1	1	0	1	0	1	48

8.2.1.1 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。

8.2.1.2 IMO 周波数拡散

IMO はスペクトラム拡散モードで動作でき、IMO の中心動作周波数で生じたノイズの振幅を小さくすることができます。このモードでは、レジスタにより選択された 4 つの拡散オプションのいずれか応じてIMOの周波数が変化します。4 つの拡散オプションは、固定周波数、三角波、疑似ランダム、および DSI 入力です。DSI 入力モードでは、デジタル信号で拡散オプションの形状を指定できます。拡散オプションは、表 8-1 に示すように、CLK_IMO_SPREAD レジスタの SS_MODE ビットにより選択されます。拡散の限界値は、表 8-2 に示すように、CLK_IMO_SPREAD レジスタの SS_RANGE ビットにより定義されます。すべての拡散オプションはダウンスプレッド方式、つまり瞬時クロック周波数の値は常にコンフィギュレーションされた周波数以下です。

表 8-1. IMO スペクトラム拡散モード ビット SS_MODE

名称	説明
SS_MODE[1:0]	IMO スペクトラム拡散モードスペクトラム拡散の周波数分布の形状を定義する 0: 無効。IMO 周波数の変化なし 1: 三角波 IMO 周波数は中心周波数から三角形の分布を形成する。カウントの限界値は SS_MAX ビットにより指定される 2: LFSR レジスタ使用の疑似ランダム シーケンス。IMO 周波数は中心周波数から疑似ランダム分布を形成する 3: DSI。IMO 周波数の分布は DSI 入力信号により決定される

表 8-2. IMO スペクトラム拡散分布範囲ビット SS_RANGE

名称	説明
SS_RANGE[1:0]	IMO スペクトラム拡散の最大範囲。スペクトラム拡散カウンターの最大カウント値において公称周波数からの拡散範囲を定義。 0: 1%。最大カウント値でスペクトラム拡散の周波数変化は 0 ～ –1% 1: 2%。最大カウント値でスペクトラム拡散の周波数変化は 0 ～ –2% 2: 4%。最大カウント値でスペクトラム拡散の周波数変化は 0 ～ –4% 3: 予約済み。使用禁止

CLK_IMO_SPREAD レジスタの SS_MAX フィールドは、スペクトラム拡散カウンターの最大カウント値を設定します。SS_MODE が三角波スペクトラム拡散に設定された場合、この値を増やすと三角波の拡散のサイクル タイム全体を増やします。

IMO スペクトラム拡散機能を有効にするには、それに接続するクロックが必要です。この回路はペリフェラル クロック 0 を自分のクロックとして使用します。IMO スペクトラム拡散のペリフェラル クロック 0 は、ペリフェラル クロック分周器からの該当するクロックに接続する必要があります。このクロックの周波数は、スペクトラム拡散回路の動作速度を決定し、従って周波数の変更速度も決定します。

8.2.1.3 プログラミング クロック (36MHz)

IMO ブロックは、フラッシュ プログラミング ブロックのクロックとして使用される 36MHz 出力があります。このクロックは、フラッシュ プログラミング ブロックのみに対応し、クロック分周器またはクロック ツリーのクロック源として使用できません。

8.2.2 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 32kHz 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、精度も良くありません。これは ハイバネートとストップ モードを除きすべての電力モードで使用可能です。PSoC 4 では、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、CLK_ILO_CONFIG レジスタの ENABLE ビットにより有効／無効化されます。

8.2.3 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。外部クロックの使用可能な周波数範囲は 0 ～ 48MHz です。PSoC 4 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 59 ページの I/O システムをご覧ください。

8.2.4 時計用水晶発振器 (WCO)

時計用水晶発振器 (WCO) は LFCLK のクロック源の 1 つとして使用されます。ILO と同様、WCO はハイバネートとストップ モードを除きすべての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。

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 ±1% 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 280ms 確保する必要があります。

8.3 クロック分配

図 8-1 に示すように PSoC 4 のクロックは発生され、あらゆるデバイスに配布されます。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

8.3.1 HFCLK 入力の選択

PSoC 4 内の HFCLK の入力オプションは IMO と EXTCLK です。HFCLK 入力は表 8-3 に示すように、CLK_SELECT レジスタの DIRECT_SEL ビットを使用することで選択されます。

表 8-3. HFCLK 入力選択ビット DIRECT_SEL

名称	説明
DIRECT_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

8.3.2 LFCLK 入力の選択

PSoC 4 内の LFCLK は、ILO と WCO の 2 つの入力オプションがあります。LFCLK は表 8-4 に示すように、WDT_CONFIG レジスタの LFCLK_SEL ビットを使用することで選択されます。

表 8-4. LFCLK 入力選択ビット LFCLK_SEL

名称	説明
LFCLK_SEL[1:0]	LFCLK 入力クロックの選択 0: ILO。ILO を LFCLK の信号源として使用 1: WCO。時計用水晶発振器を LFCLK の信号源として使用 2 ～ 3: 予約済み。使用禁止

8.3.3 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 8-5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。

8.3.4 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 20 個のクロック分周器 (16 個の 16 ビット クロック分周器と 4 個の 16.5 ビット分数クロック分周器) を備えています。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。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 の場合、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 4 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 3 で、分数分周値が 16 の場合、48MHz の HFCLK から 48 / (3 + 16/32) = 48 / 3.5 = 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 3 HFCLK サイクルで、残りの半数 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 8-5 と表 8-6 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 8-5. 整数ペリフェラル クロック分周器のコンフィギュレーション 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。[2、65,536] 範囲内の分周が実行可能

表 8-6. 分数ペリフェラル クロック分周器のコンフィギュレーション 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。[2、65,536] 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 のいずれか 1 つです。SEL_TYPE = 63 と SEL_TYPE = 3 であれば何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 はリファレンス 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 (0, 1)) 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 をリファレンス クロックとして使用することに注意してください。

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 (PER#_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 8-7 は (図 8-1 に示している) 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。20 個のペリフェラル クロック分周器のいずれか 1 つは、表 8-8 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 8-7. ペリフェラル クロック マルチプレクサ出力マッピング

ペリフェラル クロック番号	ペリフェラル
0	IMO (スペクトラム拡散)
1	CLOCK_PUMP
2	SCB0
3	SCB1
4	SCB2
5	SCB3
6	CSD0_ CLK0
7	CSD0_ CLK1
8	CSD1_CLK0
9	CSD 1_CLK1
10	SAR
11	TCPWM0

表 8-7. ペリフェラル クロック マルチプレクサ出力マッピング

ペリフェラル クロック番号	ペリフェラル
12	TCPWM1
13	TCPWM2
14	TCPWM3
15	TCPWM4
16	TCPWM5
17	TCPWM6
18	TCPWM7
19	UDB0 (PSoC 4200M 専用)
20	UDB1 (PSoC 4200M 専用)
21	UDB2 (PSoC 4200M 専用)
22	UDB3 (PSoC 4200M 専用)
23	LCD

表 8-8. プログラム可能な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: 使用禁止

8.4 低消費電力動作

PSoC 4 のクロック動作は電力モードに応じて異なります。IMO、EXTCLK、HFCLK、SYSCLK、およびペリフェラルクロックの MHz 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、ハイバネートとストップ モードを除いてすべての電力モードで動作します。

8.5 レジスター覧

表 8-9. 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 IMO 調整値を含み、その周波数の微調整を可能にする
CLK_IMO_TRIM2	IMO 周波数選択レジスタ - このレジスタは IMO の周波数範囲を制御し、その周波数の粗調整を可能にする
PWR_BG_TRIM4	バンドギャップ トリム レジスタ - これらのレジスタはバンドギャップ リファレンスの調整を制御し、デバイスの電圧リファレンスの調整を可能にする
PWR_BG_TRIM5	
CLK_IMO_SPREAD	IMO スペクトラム拡散制御レジスタ - このレジスタは、IMO スペクトラム拡散機能を制御
BLE_BLERD_DBUS	ECO イネーブル。このレジスタは ECO を有効／無効にするビットを含む
BLE_BLERD_BB_XO_CAPTRIM	X1 と X2 コンデンサ トリム レジスタ - このレジスタは ECO の X1 と X2 ノードのコンデンサ値を制御
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_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	プログラム可能な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

9. 電源と電圧監視

PSoC®4 は、1.71 ～ 5.5V の一系統の外部供給電圧により動作します。電圧範囲は以下の 2 つにわかれます。

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

PSoC4 のデバイスは様々な省電力モードをサポートするために、多くの内部レギュレータがありますそれはアクティブ デジタル レギュレータ、低雑音レギュレータ、ディープスリープ レギュレータおよびハイバネート レギュレータです。

9.1 ブロック図

図 9-1. パワー システムのブロック図

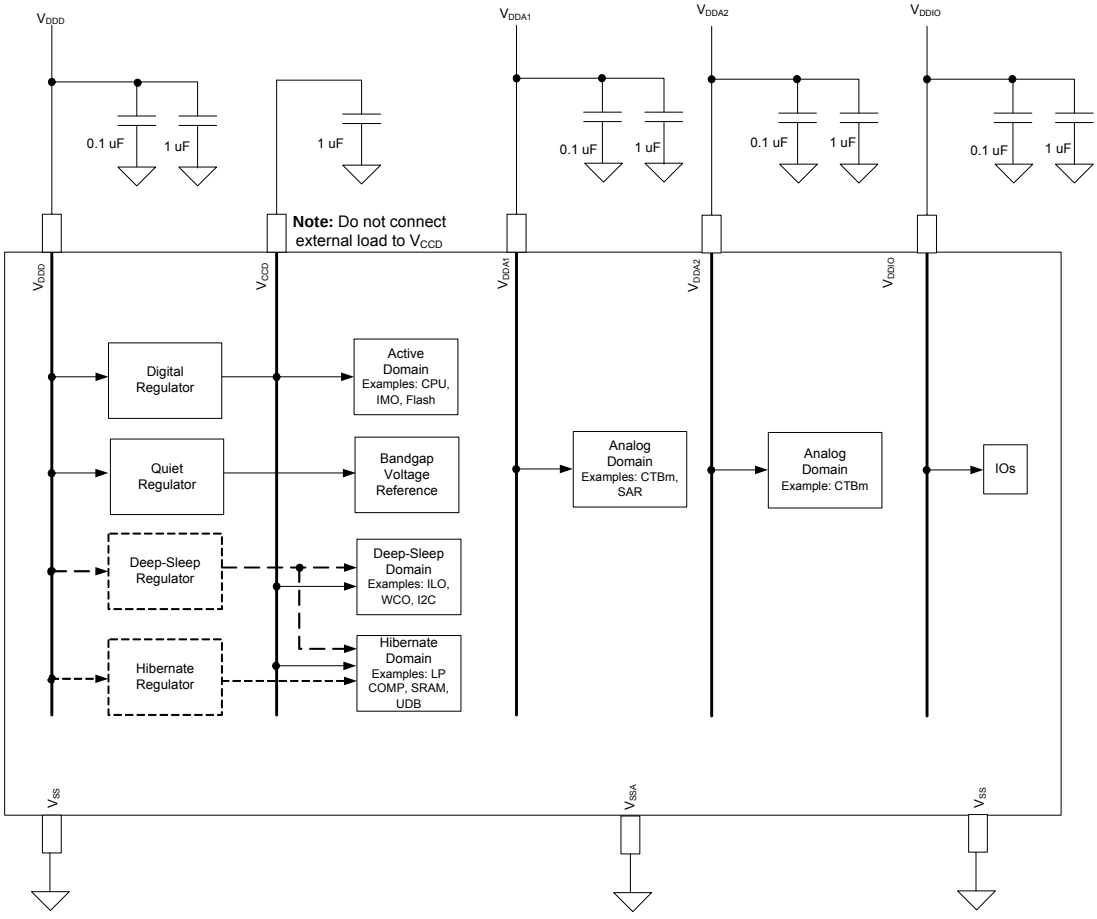


図 9-1 にPSoC 4の電源システム図およびすべての電源ピンを示します。システムには、アクティブ モードで動作するデジタル回路用レギュレータが 1 個ありますアナログ レギュレータはありません。アナログ回路は VDDA 電源により動作します。ディープ スリープモードとハイバネート (電源供給を減らすがメモリを保持する) モード用にそれぞれ独立したレギュレータ があります。バンドギャップ電圧リファレンス用に独立した低雑音レギュレータがあります。電源電圧範囲は 1.71V ～ 5.5V で、すべての機能と回路がその範囲において動作します。PSoC 4 には、非安定化外部電源供給と安定化外部電源供給モードの 2 つの電源供給モードがあります。電源接続の詳細については デバイス データシートの「電力」セクションを参照してください。

9.2 動作原理

図 9-1 に示すレギュレータは、デバイスの様々のドメインに電源を供給します。すべてのコア レギュレータおよびデジタル I/O は、V_{DD} の電源ピンから電力を取り込みます。デジタル I/O は V_{DDIO} から供給されます。アナログ回路は V_{DDA} 電源により直接動作します。

9.2.1 レギュレータの概要

アクティブ デジタル レギュレータと低雑音レギュレータは、アクティブまたはスリープ モードで有効になります。ディープ スリープおよびハイバネート モードではオフになります (表 11-1 と 図 9-1 を参照) ディープスリープとハイバネート モード レギュレータは、低消費電力モードでデバイスの電源供給条件を満たすよう設計されています。

表 9-1. 電力モード別レギュレータの状態

モード	アクティブレギュレータ	低雑音レギュレータ	ディープスリープ レギュレータ	ハイバネートレギュレータ
ストップ	オフ	オフ	オフ	オフ
ハイバネート	オフ	オフ	オフ	オン
ディープ スリープ	オフ	オフ	オン	オン
スリープ	オン	オン	オン	オン
アクティブ	オン	オン	オン	オン

9.2.1.1 アクティブ デジタル レギュレータ

1.8V ～ 5.5V の範囲の外部電源の場合、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} ピン に接続され、外部にデカップリング コンデンサ (1 µF X5R) を接続します。

1.8 V 以下の電源の場合、VCCD を直接給電する必要があります。この場合、V_{CCD}、V_{DDD}、V_{DDA1}、V_{DDA2} および V_{DDIO} を相互に接続する必要があります。

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。これにより直接電源モードでの消費電力が減少します。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

9.2.1.2 低雑音レギュレータ

このレギュレータは、デジタル スイッチング ノイズや電源ノイズがない低雑音電源を必要とする、バンドギャップ リファレンスや静電容量型検出サブシステムのようなアナログ回路に電源を供給します。このレギュレータは高い電源電圧変動除去比があります。低雑音レギュレータは、アクティブとスリープ モードで使用可能です。

9.2.1.3 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回路 (ILO、WCO、と SCB など) に電源供給します。ディープ スリープ レギュレータはハイバネート モード以外、すべての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は、このレギュレータの出力は、デジタル レギュレータの出力 (V_{CCD}) に接続されます。このレギュレータは低速クロックリソースに安定した電圧を供給する独立した電圧複製出力も持ちます。この出力はアクティブとスリープ モードで V_{CCD} に接続されません。

9.2.1.4 ハイバネート レギュレータ

このレギュレータはスリープ コントローラー、低消費電力コンパレータおよび SRAM のようなハイバネート モードで電源供給される回路に電源供給します。ハイバネート レギュレータはすべての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は、このレギュレータの出力は、デジタル レギュレータの出力に接続されます。ディープ スリープ モードでは、このレギュレータの出力は、ディープスリープ レギュレータの出力に接続されます。

9.3 電圧監視

電圧監視のシステムはパワーオンリセット (POR)、ブラウンアウト検出 (BOD) および低電圧検出 (LVD) を含みます。

9.3.1 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。

9.3.1.1 ブラウンアウト検出 (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 V_{CCD} (min) 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。

ファームウェアがブラウンアウト イベントと通常のパワーサイクルを区別できるように、BOD がリセットを発生した後にクリアされない特別なレジスタ (PWR_BOD_KEY) が提供されます。しかしこのレジスタはデバイスが POR または XRES されるとクリアされます。BOD はストップモード以外、すべての電力モードで使用可能です。

9.3.1.2 低電圧検出 (LVD)

LVD 回路は外部電源電圧を監視し、エネルギーソースの減少を正確に検出します。LVD 検出器は、システムが事態に対処できるように割り込みを生成します。

LVD はアクティブとスリープ モードだけで使用できます。ディープスリープ モードで LVD が必要な場合、WDT を復帰ソースとして使用してチップを定期的にディープスリープ モードから復帰させるよう構成する必要があります。LVD 監視はアクティブ モードで行う必要があります。LVD 回路のトリップ ポイントは、安全な動作電圧の範囲内でプログラム可能です。検出時に割り込みを生成します。LVD のトリップ ポイントは、PWR_VMON_CONFIG レジスタ内の LVD_SEL フィールドを使用して 1.75V ～ 4.5V の範囲内に設定可能です。

LVD 回路を有効にする時、初期のセトリング時間中に誤った割り込みが発生する場合があります。ファームウェアは PWR_VMON_CONFIG レジスタの LVD_EN ビットをセットした後、1 µs 待機することでこれをマスクすることができます。LVD 機能を有効にするには、以下の推奨ファームウェア手順に従ってください。

1. 誤割り込みの伝達を防止するために、PWR_INTR_MASK レジスタ内の LVD ビットが 0 であることを確認します。
2. PWR_VMON_CFG レジスタの LVD_SEL フィールドに必要なトリップポイントを設定します。
3. PWR_VMON_CFG レジスタ内の LVD_EN ビットをセットして LVD を有効にします。この時点で誤 LVD イベントが発生することがあります。
4. 回路が安定するまで、少なくとも 1µs 待ちます。
5. PWR_INTR レジスタ内の LVD ビットに 1 を書き込んで誤イベントを削除します。LVD 条件が実際に存在する場合、このビットはクリアされません。
6. PWR_INTR_MASK レジスタ内の LVD ビットを使用して割り込みマスクを解除します

9.4 レジスタの一覧

表 9-2. 電源と電圧監視レジスタの一覧

レジスタ名	説明
PWR_CONTROL	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
PWR_INTR	電源システム割り込みレジスタ – このレジスタは電源システムの割り込み状態を示す
PWR_INTR_MASK	電源システム割り込みマスク レジスタ – このレジスタはどの割り込みが CPU の割り込みコントローラーへ伝わるか制御する
PWR_VMON_CONFIG	電源システム監視電圧設定レジスタ – このレジスタは電圧監視システムの監視電圧設定などの設定ビットを格納する

10. チップの動作モード

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。これらのモードはハードウェアの異なる特権レベルでフラッシュおよび ROM での異なるロケーションからの実行を決定します。これらのモードの内、3 つだけは最終のアプリケーションで使用されます。デバッグモードはファームウェア開発中に設計のデバッグ処理に排他的に使用されます。

PSoC 4 の動作モードは以下のとおりです。

- ブート
- ユーザー
- 特権
- デバッグ

10.1 ブート

ブート モードはデバイスがデバイスの 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。電源投入のフェーズでは、適切な動作を確保するためにハードウェアの調整設定が不揮発性 (NV) ラッチからロードされます。ブート処理が終わった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。

10.2 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。

10.3 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 308 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。

10.4 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。規定の開通時間内に 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 299 ページのデバッグ インターフェースを参照してください。

保護モードの詳細については 93 ページのデバイス セキュリティを参照してください。

11. 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化するために、複数の電力モードを用意しました。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
- ハイバネート
- ストップ

アクティブ、スリープおよびディープスリープは ARM 標準により定義された電力モードであり、ARM CPU および命令セットアーキテクチャ (ISA) によりサポートされます。ハイバネート モードおよびストップ モードは PSoC 4 によりサポートされた追加の省電力モードです。これらのモードはディープスリープと同じようにファームウェアにより遷移しますが、復帰の時、CPU とすべてのペリフェラルはリセットされて復帰します。

それぞれの電力モードの電力消費量は以下の方法で制御されます。

- ペリフェラルへのクロックを有効／無効
- 内部レギュレータの電源オン／オフ
- クロック ソースの電源オン／オフ
- その他の部分の電源オン／オフ

図 11-1 に電力モードの状態遷移図を示します。

図 11-1. 電力モードの状態遷移図

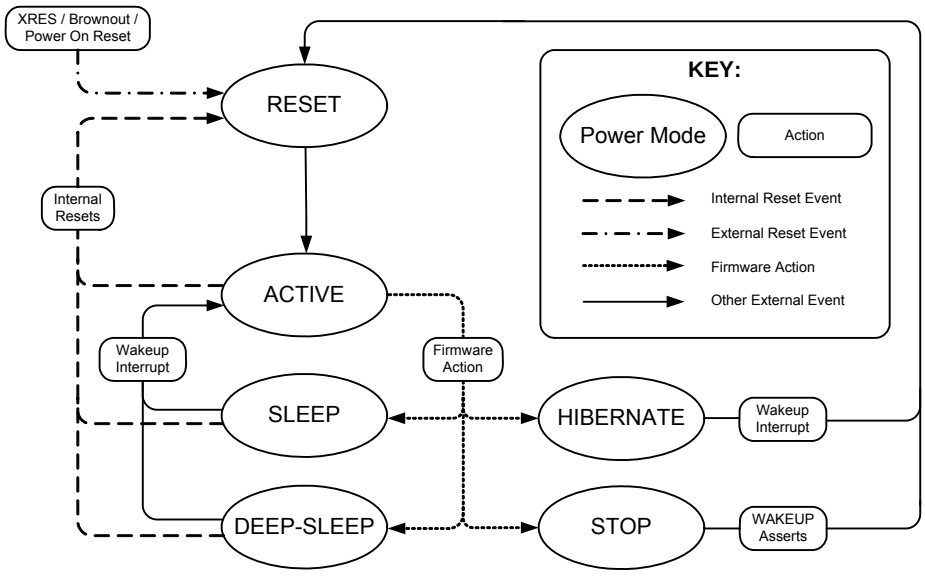


表 11-1 はPSoC 4に用意された電力モードについて説明します。

表 11-1. PSoC 4 電力モード

電力モード	説明	遷移条件	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を利用可能 (プログラム可能)	他の省電力モードから復帰、内部と外部がリセット、ブラウンアウト、パワー オン リセット	該当なし	任意 (プログラム可能)	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
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)	レジスタ書き込み	任意の割り込み	任意 (プログラム可能)	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。低周波 (32kHz) のクロックのみが使用可能 低速度、非同期あるいは低電力アナログ ペリフェラルの割り込みにより復帰が可能	レジスタ書き込み	GPIO 割り込み、低電力コンパレータ、SCB、ウォッチドッグ タイマー	ILO (32kHz)、WCO (32kHz)	割り込み	ディープスリープ レギュレータおよびハイバネート レギュレータ
ハイバネート	SRAM および UDB が保持される。ハイバネート以外、すべての内部電源がオフになる。復帰はピン割り込みまたは低電力コンパレータにより可能	レジスタ書き込み	GPIO 割り込み、低電力コンパレータ、SCB、ウォッチドッグ タイマー	なし	リセット (割り込み状態保持)	ハイバネート レギュレータ
ストップ	すべての内部電源がオフになる。GPIO の状態のみが保持される。復帰は、XRES または WAKEUP ピンのみにより可能	レジスタ書き込み	WAKEUP ピン	なし	リセット	なし

表 11-1 に記載した復帰ソースに加え、外部リセット (XRES) とブラウンアウト リセットは省電力モードからアクティブ モードにデバイスを遷移させます。

11.1 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。

11.2 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。低電力消費を達成するために、CPU がアイドリングすると同時に頻繁に遷移するようなモードです。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。

11.3 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、UDB および高速な回路がデータ保持状態に入ります。MHz 帯のクロック (HFCLK と SYSCLK を含む) は無効になります。内部用低周波発振器 (32kHz) および時計用水晶発振器 (WCO) の動作を維持し、低周波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か、I2C スレーブなどの外部インターフェースからクロックを受けるデジタル ペリフェラルは動作し続けます。低速度、非同期あるいは低電力のアナログ 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。CTBm も消費電力と帯域幅を減少させれば、このモードで動作させることができます。消費電力および CTBm の帯域幅についての詳細な情報はデバイス データシートを参照してください。

利用可能な復帰ソースを表 11-2 に載せています。

11.4 ハイバネート モード

SRAM データを保持する PSoC 4 の最小電力モードです。このモードは、すべてのクロックを停止し、CPU とすべてのペリフェラルへの電源を停止することにより実行されます。ただし外部イベントによりシステムを復帰させる一部の (非同期の) ペリフェラルは例外です。CPU とすべてのペリフェラルは、このモードで動作状態を失うことに留意してください。

ハイバネート モードでは、電力容量を制限されたハイバネート レギュレータが極めて低い電力消費を達成するために使用されます。またハイバネート モードの間、入力ピンに接続される信号の最大周波数が制限されます。すべての I/O ピンで、合計トグル レート (すべての入力ピンと出力ピンでの信号の合計周波数) は 10kHz を超えてはいけません。

高いレートでトグルする信号があるシステムは、総合消費電力の差はありますが、ディープスリープ モードを使用します。

ハイバネート モードからの復帰は、ピン割り込みまたは低電力コンパレータによるものだけです。ハイバネートからの復帰は、割り込みによる復帰と異なり、リセットを発生させます。ハイバネートからの復帰は、CPU とほとんどのペリフェラルがリセット状態に入り、ファームウェアはリセット ベクトルでスタートします。I/O ピンはリセットの後でトライステートにされます。しかし、ハイバネート モードに遷移する前にこれらのピンがファームウェアによって明示的に凍結された場合はこの限りではありません。割り込み原因が分かるように、PWR_STOP レジスタ上の TOKEN ビットを 83 ページの省電力モードへの遷移および復帰に述べてある通りに使用します。

外部リセット (XRES) は完全なシステム再起動を行います。この場合は、原因をデバイスが再起動した後に読み込むことができません。そして I/O ピンはそれらの凍結状態を保持しません。

11.5 ストップモード

ストップ モードでは、CPU、すべての内部レギュレータ、すべてのペリフェラルの電源が切断されます。ストップモードからの復帰はシステム リセットであり、XRES または WAKEUP ピンのみにより可能です。I/O ピンはリセットの後でトライステートにされます。しかし、ストップモードに遷移する前にこれらのピンがファームウェアによって明示的に凍結された場合はこの限りではありません。割り込み原因が分かるように、PWR_STOP レジスタ上の TOKEN ビットを 83 ページの省電力モードへの遷移および復帰に述べてある通りに使用します。

外部リセット (XRES) は完全なシステム再起動を行います。この場合は、原因をデバイスが再起動した後に読み込むことができません。そして I/O ピンはそれらの凍結状態を保持しません。

11.6 電力モードの概要

表 11-2 はそれぞれの省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 11-2 は各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 11-2. 復帰ソース

電力モード	復帰ソース	復帰動作
スリープ	任意の割り込みソース	割り込み
	任意のリセット ソース	リセット
ディープスリープ	GPIO 割り込み	割り込み
	低電力コンパレータ	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
	XRES (外部リセット ピン)[a]、ブラウンアウト	リセット
	CTBm	割り込み
	BLESS	割り込み
ハイバネート	GPIO 割り込み	リセット
	低電力コンパレータ	リセット
	XRES (外部リセット ピン)[a]、ブラウンアウト	リセット
ストップ	WAKEUP ピン	リセット
	XRES (外部リセット ピン)[a]、ブラウンアウト	リセット

a. XRES は完全にシステムを再起動します。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。

11.7 省電力モードへの遷移および復帰

Cortex-M0 (CM0) からの「割り込み待機 (WFI)」命令はスリープ、ディープスリープおよびハイバネート モードへの遷移をトリガーします。Cortex-M0 は、最も低い優先度の ISR が実行から復帰するまで、省電力モードへの遷移を延期することができます。(CM0 システム コントロール レジスタ上の SLEEPONEXIT ビットがセットされる場合)

スリープ、ディープスリープおよびハイバネートのモード

への遷移は、CM0 システム コントロール レジスタ (CM0_SCR) 上の SLEEPDEEP フラグおよびシステム リソース パワー サブシステム (PWR_CONTROL) 上の HIBERNATE によって制御されます。

- WFI 命令が実行され、SLEEPDEEP = 0 および HIBERNATE = x である時、スリープ モードに遷移します。
- WFI 命令が実行され、SLEEPDEEP = 1 および HIBERNATE = 0 である時、ディープスリープ モードに遷移します。
- WFI 命令が実行され、SLEEPDEEP = 1 および HIBERNATE = 1 である時、ハイバネート モードに遷移します。

PWR_CONTROL レジスタ上の LPM READY ビットは、ディープスリープおよびハイバネート 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またはハイバネート モードへ遷移しようとしたら、まず PSoC 4 はスリープ モードへ遷移され、レギュレータの準備が整った後ディープスリープまたはハイバネート 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 11-2 を参照)、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、省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。

これらの省電力モードで GPIO 状態を凍結するのに PWR_STOP レジスタを使います。ハイバネート モードおよびストップ モードから復帰する時はシステム リセットを発生するので、この機能はこれらのモードに推奨されます。

システム リソース パワー サブシステムの PWR_STOP レジスタを使って直接ストップ モードに遷移します。これはシステムのすべての低電圧ロジックから電源を切断します。I/O 状態および PWR_STOP レジスタ コンテンツのみが保持されます、そして復帰 (リセット) は XRES または固定の WAKEUP ピンのトグル操作で発生します。

PWR_STOP レジスタのフィールドは :

- TOKEN – このフィールドは、一般の RESET イベントからの WAKEUP を区別するために、ファームウェアによって使用される STOP/WAKEUP シーケンスを通して保持される 8 ビットのトークンが入っています。XRES を使って STOP から復帰させるとこのレジスタがリセットされることに注意してください。
- UNLOCK – ストップ モードを利用可能にする場合 0x3A を書き込みます。このフィールドが他の設定値であれば、ハードウェアは STOP ビットを無視します。
- POLARITY – このビットは WAKEUP ピン入力の極性を設定します。デバイスは、 WAKEUP ピンの入力が POLARITY ビットの値に一致する時、復帰します。
- FREEZE – このビットをセットするとシステムのコンフィギュレーション、モードおよびすべての GPIO の状態を凍結します。
- STOP – ストップ モードに遷移するためにこのビットをセットします。

ストップ モードへ遷移するための推奨手順は :

1. TOKEN = < 任意のアプリケーションの規定値 > を書き込む
2. UNLOCK = 0x3A を書き込む
3. POLARITY = < アプリケーションの指定極性 > を書き込む
4. FREEZE = 1 を書き込む
5. STOP = 1 を書き込む

3 番目の書き込みの後、NOP サイクルを 2 つ加えることが推奨されます。XRES または WAKEUP ピンがトグルされると、ストップ モードが終了します。どちらのイベントも PWR_STOP レジスタの STOP ビットをクリアし、その後 POR をトリガーします。復帰イベントは PWR_STOP レジスタの他のビットをクリアしませんが、XRES イベントはすべてのビットをクリアします。

ストップ モードまたはハイバネート モードから復帰する、ファームウェアの手順は以下の通りです。

1. 必要なら、アプリケーション指定のブランチ用の TOKEN を読み出します。
2. 必要なら、要求された設定に I/O 駆動モードおよび出力データ レジスタを書き込みます。デジタル出力ポートの標準的手順は、ピンを出力として設定し、凍結値を読み出し、そして出力データ レジスタにその値を設定します。
3. I/O を解凍します。

11.8 レジスター覧

表 11-3. 電力モードのレジスター覧

レジスタ名	説明
CM0_SCR	システム コントロールのレジスタ – システム コントロール データを設定
PWR_CONTROL	電力モード コントロール – 電力モードの詳細を設定し、現在の状態を監視
PWR_STOP	このレジスタはストップモードへの遷移を制御

12. 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に陥つた時、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 または WCO が生成する LFCLK により動作します。有効にされる場合、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満了し、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。

12.1 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
- 経過時間を増加させるために、カスケード接続できる独立した16ビット カウンター (2個) と32ビット カウンター (1個) をサポート

12.2 ブロック図

図 12-1. ウォッチドッグ タイマー ブロック図

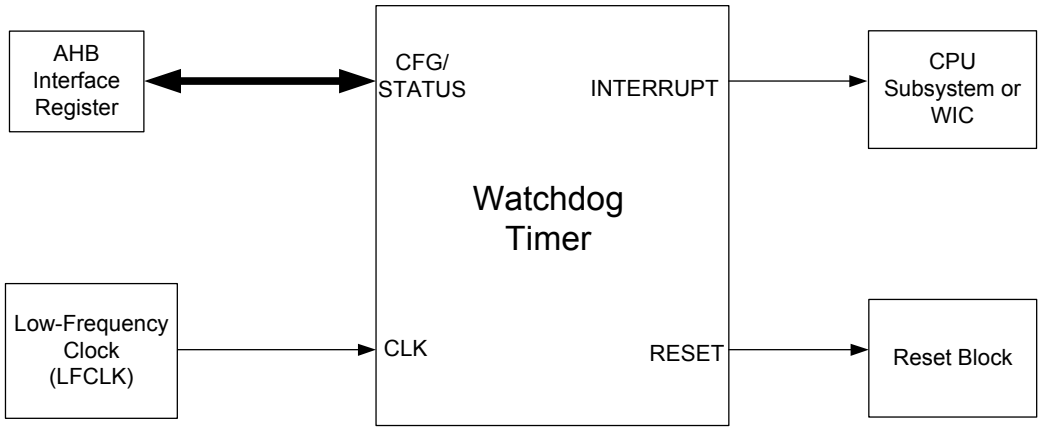
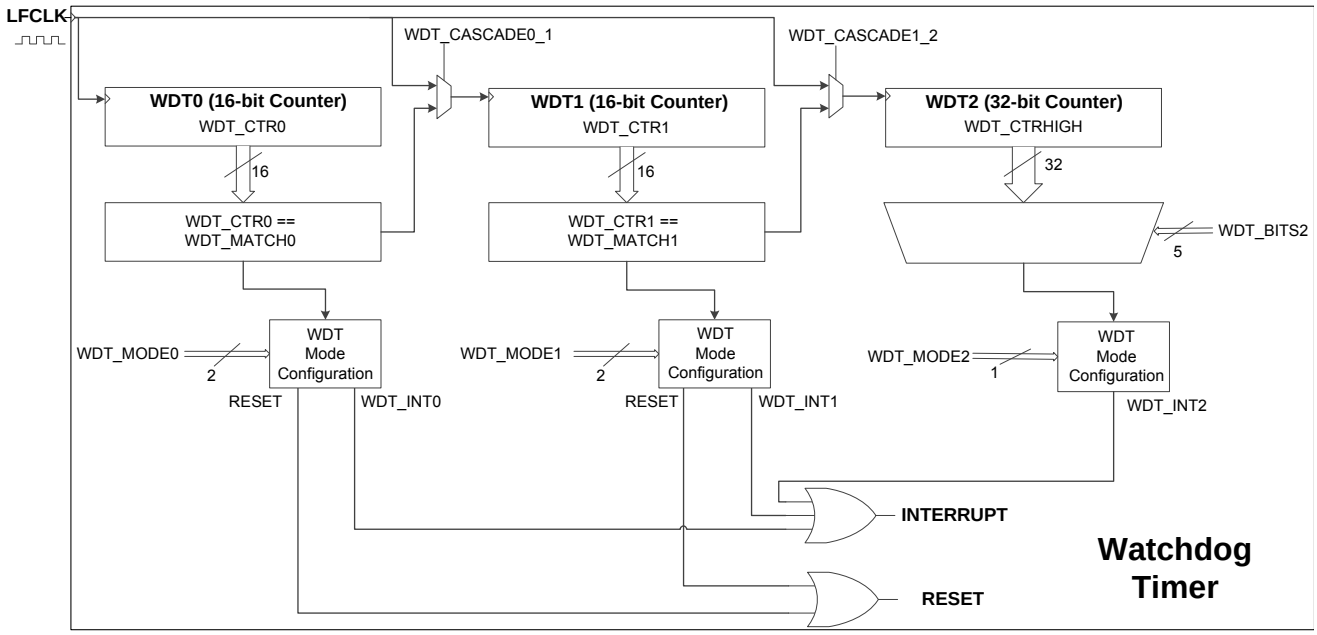


図 12-2. ウォッチドッグ タイマーの内部ブロック図



12.3 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 はプログラム可能な経過時間の後にデバイスへの割り込みまたはハードウェア リセットをアサートします。WDT は 2 つの 16 ビット カウンター (WDT0 と WDT1) と 1 つの 32 ビット カウンター (WDT2) があります。これらのカウンターが単独でまたはカスケードで動作するよう構成できます。

WDT0 と WDT1 は一致イベント、つまりカウンター値と比較値が同じ時、割り込みを生成するよう構成されます。WDT0 と WDT1 カウンターは、一致イベント時または 3 回連続する未処理の一致イベント (一致イベント割り込みはクリアされない) の後にリセットを生成するように構成することもできます。

WDT2はWDT_CONFIGレジスタ内のWDT_BITS2[4:0] ビットの設定に応じて割り込みを生成するよう構成されます。WDT2 はシステム リセットを生成できません。また、WDT1 や WDT0 のように値の一致で割り込みを生成することもできません。WDT2 は、32 ビットカウンターのいずれかのビットの立ち上がりエッジで割り込みを生成します。WDT_CONFIG レジスタの WDT_BITS2[4:0] ビットで割り込みを生成するビットを設定します。詳細については 「PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM 」に記載される WDT_CONFIG レジスタのセクションを参照してください。

上記の通りに、WDT_MODEx ビットはウォッチドッグ カウンターを設定するために使用されます。

図 12-2 に示すカスケード接続コンフィギュレーションは、リセットまたは割り込みが起こる経過時間を増やす手段を提供します。2 つの 16 ビット カウンターをカスケード接続しても32ビット カウンターが形成されないことに注意してください。その代わりに 16 ビット分周器出力に対して 16 ビット周期カウンターを取れます。例えば、WDT0 と WDT1 をカスケード接続する時、WDT0 は WDT1 の分周器として動作して、分周値は WDT_MATCH レジスタの WDT_MATCH0[15:0] ビットに定義されます。WDT1 の周期は WDT_MATCHレジスタ内のWDT_MATCH1[31:16]ビットに定義されます。WDT1 と WDT2 のカスケード接続でも同じ理屈を使用します。

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 WDT に合わせます。

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。
- 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。一致イベントでリセットを生成するよう設定されると、WDTx カウンターをクリアすることによりウォッチドッグをリセットします。WDTx カウンターは WDT_CONFIG レジスタ内の WDT_RESETx ビットをセットすることでクリアできます。詳細については「 PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM」の WDT_CONFIG レジスタ セクションを参照してください。

- 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なら、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。このため、システム リセットと割り込みを生成するために同じウォッチドッグ カウンターを使用することをお勧めしません。例えば、クラッシュからシステムを保護することを目的としてシステム リセットを生成するために WDT0 を使用する場合、WDT1 と WDT2 は定期的に割り込みを生成する割り込みタイマーとして使用される必要があります。

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. WDT0 または WDT1 が一致イベントで関連したウォッチドッグ カウンターを「0」にリセットするように WDT_CONFIG レジスタ内の WDT_CLEAR0 または WDT_CLEAR1 ビットをセットします。
2. WDT0/WDT1 の場合は WDT_MATCH レジスタに比較値を書き込み、WDT2 の場合は WDT_CONFIG レジスタに WDT_BITS2 値を書き込みます。
3. 保留の割り込みをクリアするために WDT_CONTROL 内の WDT_INTx ビットをクリアします。
4. WDT_CONFIG レジスタ内の WDT_MODEx ビットを設定して WDT 割り込みを有効にします。WDT0 または WDT1 の場合、WDT_CONFIG レジスタ内の WDT_MODE0 または WDT_MODE1 ビットを「1] (一致時割り込み) または「3 」(一致時割り込み、3 番目の未処理の一致時リセット) に設定します。WDT2 の場合、WDT_CONFIG レジスタ内の WDT_MODE2 ビットをセットします。
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。(詳細については 47 ページの割り込み を参照してください)。
6. ISR で WDT 割り込みをクリアします。

WDT_MATCH レジスタを変更するには、3 つの LFCLK サイクルが必要です。WDT_MATCH を変更した後、WDT が新しい設定に更新することを保証するために LFCLK サイクルを待った後ディープスリープ モードに入ります。

12.3.1 WDT の有効化と無効化

WDT カウンターは WDT_CONTROL レジスタの WDT_ENABLEx ビットをセットして有効になり、またこのビットをクリアして無効になります。WDT を有効、無効にするには 3 LFCLK サイクルの時間がかかります。従って WDT_ENABLEx ビットをこの期間内に変化させてはいけません。

WDT を有効にした後、WDT コンフィギュレーション (WDT_CONFIG) への書き込みと (WDT_CONTROL) レジスタの制御をお勧めしません。CLK_SELECT レジスタの WDT_LOCK[15:14] ビットをセットすることにより WDT レジスタの予想外の破損を予防できます。WDT が実行している間に WDT_MATCH を更新する必要がある場合、WDT_LOCK ビットをクリアする必要があります。両方の WDT_LOCK ビットをクリアするために 2 つの異なる書き込みが必要です。ビットに「1」を書き込むと、ビット 0 がクリアされます。「2 」を書き込むと、ビット 1 がクリアされます。「3」を書き込むと、両方のビットがセットされます。「 0」は書き込んでも何の影響はありません。詳細については「 PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM」内の CLK_SELECT レジスタ を参照してください。

12.3.2 WDT 動作モード

WDT0 と WDT1 は、システムのハングアップ状態を終了させるリセットを生成したり、スリープ／ディープスリープ モードからシステムを復帰させる割り込みを生成するために使用することができます。WDT_CTRx レジスタ ビット内のカウントが WDT_MATCH レジスタ内の比較値 (WDT_MATCHx) と同じ場合、WDT_CONFIG レジスタ内の WDT_MODEx[1:0] ビット フィールドを設定して必要なアクションを選択することができます。WDT_CTRHIGH、WDT_CTRLOW、WDT_MATCH レジスタの詳細については「 PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM 」を参照してください。

WDT2 は WDT_BITS2[4:0] レジスタ ビットの状態に応じて割り込みを生成するために使用されます。

表 12-1. WDT0 と WDT1 モード

ビットフィールド名	説明
WDT_MODE0[1:0] または WDT_MODE1[1:0]	ウォッチドッグ カウンターが一致するときの動作 (WDT_CTR0=WDT_MATCH0) または (WDT_CTR1=WDT_MATCH1): 00: 何もしない 01: WDT_INT0 または WDT_INT1 をアサート 10: WDT リセットをアサート 11: WDT_INT0 または WDT_INT1 をアサートし、別の未処理の割り込みの後に WDT リセットをアサート

表 12-2. WDT2 モード

ビットフィールド名	説明
WDT_MODE2	0: 割り込み要求なしのフリーランの カウンター 1: 割り込み要求が、WDT_CONFIG レジスタ内の WDT_BITS2 ビットで指定されるビットの立ち上がりエッジで発生するフリーランの カウンター

注 : ウォッチドッグ カウンターは LFCLK サイクル毎に割り込みを生成するよう設定された時、(WDT_CONTROL レジスタでの WDT_INTx をセットすることにより) ウォッチドッグ割り込みをクリアした後 WDT_CONTROL レジスタを読み出さなければなりません。そうでない場合、次の割り込みを取りこぼす可能性があり、LFCLK/2 の割り込みサイクルも発生します。

12.3.3 WDT 割り込みと省電力モード

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 が割り込み要求をアクノリッジして、ISR を実行します。ファームウェア内の ISR 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サブシステムは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 が割り込み要求をアクノリッジして、ISR を実行します。ファームウェア内の ISR に入った後に、割り込みをクリアする必要があります。

詳細については 81 ページの電力モードを参照してください。

12.3.4 WDT リセット モード

RES_CAUSE レジスタ内の 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)、ブラウンアウトリセット (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89ページのリセット システムを参照してください。

12.4 レジスター覧

表 12-3. WDT レジスタ

レジスタ名	説明
WDT_CTRLOW	ウォッチドッグ カウンター 0 と 1
WDT_CTRHIGH	ウォッチドッグ カウンター 2
WDT_MATCH	ウォッチドッグ カウンター 0 と 1 用の比較値
WDT_CONFIG	WDT コンフィギュレーション ビット
WDT_CONTROL	WDT カウンターの動作制御

13. 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 は一定のリセット検出を可能にするために、ハードウェアを内蔵しています。

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): 電源電圧が立ち上がる時、デバイスをリセット状態に維持する
- ブラウンアウトリセット (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
- ソフトウェア初期化リセット (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): PSoC 4 外部の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
- ハイバネート復帰リセット : デバイスをハイバネート モードから復帰させる
- ストップ復帰リセット : デバイスをストップ モードから復帰させる

13.1 リセット ソース

次のセクションでは、PSoC4 で利用可能なリセットソースについて説明します。

13.1.1 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 リセットのために供給されます。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 はリセットを発生させます。

13.1.2 ブラウンアウトリセット

ブラウンアウト リセットは、デジタル電源電圧 V_{CCD} がデバイス 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 はストップモード以外すべての電力モードで使用可能です。

BOD イベントはリセット原因ステータス ビットをセットしませんが、検出できる場合もあります。BOD イベントでは、V_{CCD} が最低動作電圧を下回るが最低データ保持電圧を超えた状態になる場合があります。そのため BOD イベントをデータ保持の状況によって POR イベントと区別できる場合があります。これは 90 ページのリセット ソースの識別で詳しく説明します。

13.1.3 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が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、WDT_CONTROL レジスタ内の WDT_ENABLEx ビットをセットして有効にします。

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 や検出不可能な BOD が発生す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。

詳細は、を参照してください。85 ページのウォッチドッグ タイマー

13.1.4 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 (CM0_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、A05F を CM0_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 や検出不可能な BOD が発生す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。すべての他のリセットはこのビットに影響を与えません。

13.1.5 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_N ピンは ***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_N 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、デバイスのデータ シートのピン配置のセクションを参照してください。

XRES ピンがアクティブの間、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セクション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、リセットは POR、検出不可能な BOD または XRES によって発生します。

13.1.6 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 79 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 や検出不可能な BOD が発生す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。

13.1.7 ハイバネート復帰リセット

ハイバネート復帰 リセットは、ハイバネート復帰ソースを検出し、デバイスをリセットしてアクティブ モードに復帰させます。ハイバネート 復帰 リセットは割り込みにより発生します。ピンとコンパレーター割り込みのいずれもハイバネート モードで使用可能です。ハイバネート復帰リセットの後、SRAM と UDB レジスタの内容は保持されますが、リセットの後のコード実行は他のリセット ソースにより発生したリセットと同様に行われます。

ハイバネート リセットはコンパレーターとピンの割り込みレジスタをチェックして検出されます。これらの割り込みレジスタのステータスはハイバネート復帰リセットの間に保持されます。

詳細は 83 ページのハイバネート モードを参照してください。

13.1.8 ストップモード復帰 リセット

ストップ モード復帰リセットは、ストップ モード復帰ソースを検出し、デバイスをリセットしてアクティブ モードに復帰させます。ストップ モード復帰リセットは、XRES ピンまたは WAKEUP ピンで発生させます。ストップ モード復帰リセットではメモリの内容は保持されません。リセットの後のコード実行は他のリセット ソースにより発生したリセットと同様に行われます。

一部のストップ モード復帰リセットは、PWR_STOP レジスタ内の TOKEN (ビット 0:7) ビット フィールドを確認して検出されます。ストップ モードに移行した場合、このビット フィールドはキーで入力されます。デバイスは WAKEUP ピンで復帰される場合、その内容は保持されます。デバイスは XRES ピンで復帰する場合は、復帰ソースが検出されません。詳細は 83 ページのストップモードを参照してください。

13.2 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に、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、XRES またはハイバネートとストップモード復帰リセット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、そのリセット (POR リセットまたはデータ保持電圧以下のブラウンアウト) 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。

ハイバネート復帰リセットは、ハイバネート モードからデバイスを復帰させるために構成されるコンパレーターとピン割り込みレジスタを検査して検出されます。WAKEUP ピンのイベントにより発生するストップモード復帰リセットは、PWR_STOP レジスタを検査して検出されます。XRES で発生するストップモード復帰リセットは検出不可です。他のリセット ソースは表 13-1 に示される RES_CAUSE レジスタのステータスに基づきます。

表 13-1. リセット ソースを検出するためのリセット原因ビット

ビット	名称	説明
0	RESET_WDT	最後のパワー サイクル以降にウォッチ ドッグ タイマーのリセットが発生した
3	RESET_PROT_FAULT	リセットが必要な保護違反が発生した
4	RESET_SOFT	Cortex-M0 は SYSRESETREQ を通じてシステムリセットを要求した

ブラウンアウトのイベントはデータが保持されるリセットと保持されないリセットの 2 種類に分けられます。v_{CCD} が最小動作電圧を下回るものの最小データ保持電圧以上の場合、BOD リセットが発生してもレジスタの内容が保持されます。V_{CCD} が最小動作電圧と最小データ保持電圧の両方を下回る場合、BOD リセットが発生するとレジスタは保持されません。このレジスタのデータ保持は、特別の PWR_BOD_KEY レジスタで検出します。PWR_BOD_KEY レジスタは、ファームウェアにより書き込まれるか、データが保持されない BOD、XRES または POR イベントなどのデータが保持されないリセットの時に値を変更します。このレジスタは、ファームウェアにより初期化され、その後データが保持される BOD が発生したかを判断するために、後に続くスタートアップ コードの実行でチェックされます。

これらの方法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 (データが保持されない BOD、POR、XRES またはストップ モード復帰リセット) 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。

13.3 レジスター覧

表 13-2. リセット システム レジスター覧

レジスタ名	説明
WDT_CONTROL	ウォッチドッグ タイマー制御レジスタ。このレジスタはデバイスのウォッチドッグ タイマーを設定する
CM0_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。このレジスタの機能の 1 つにソフトウェア リセットの発行がある
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する
PWR_STOP	このレジスタはストップ モードへの遷移を制御

14. 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堅牢なフラッシュ保護により高レベルのセキュリティを提供します。PSoC 4200M では、ファームウェアに代わりユニバーサル デジタル ブロック (UDB) にカスタム機能を実装することによってセキュリティ強化が得られます。UDB に実装されたハードウェアの設計をリバースエンジニアリングすることはオブジェクトコードをリバースエンジニアリングすることより困難です。
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、詳細な情報については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の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。

14.1 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。

- ユーザーが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の不注意による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。

それに加えて PSoC 4 は個別のフラッシュ列データの保護も提供します。

14.2 動作原理

14.2.1 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 モードの間に、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。この変更によりユーザー コードまたはメモリへのデバッグ アクセスが無効になります。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。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。この変更によりユーザー 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 モードの終了ができません。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。

14.2.2 フラッシュのセキュリティ

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不注意な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 14-1 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。

表 14-1. フラッシュ保護レベル

保護設定	可能	不可
非保護	外部読み出しおよび書き込み、 内部読み出しおよび書き込み	-
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PSoC 4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。

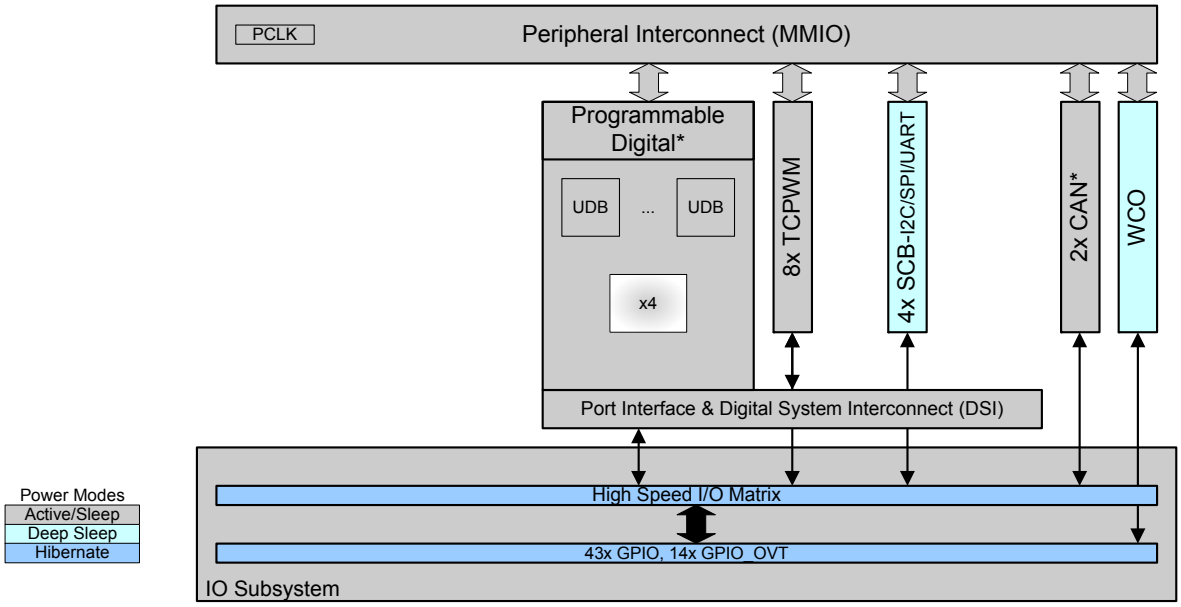
セクション D: デジタル システム

このセクションは次の章を含みます。

- シリアル通信 (SCB) (97 ページ)
- ユニバーサル デジタル ブロック (UDB) (137 ページ)
- コントローラー エリア ネットワーク (CAN) (177 ページ)
- タイマー、カウンター、PWM (193 ページ)

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



* Available only PSoC 4200M

15. シリアル通信 (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 4 デバイスは 4 つの SCB を備えています。シリアル ペリフェラル インターフェース (SPI) と UART プロトコルの追加インスタンスは PSoC 4200M 内のユニバーサル デジタル ブロック (UDB) を使って実装できます。

15.1 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
- 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 プロトコル用の省電力 (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。

15.2 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。

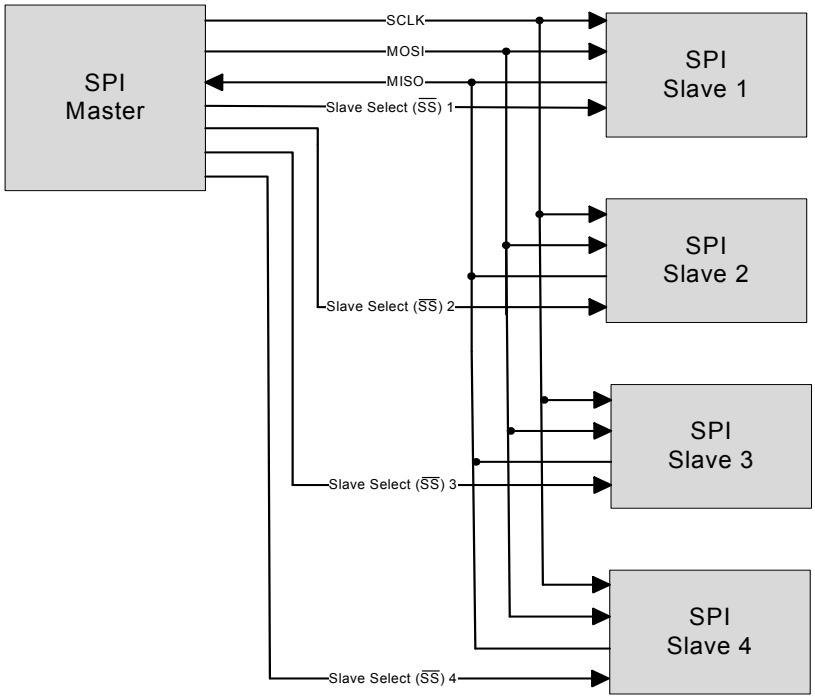
15.2.1 特長

- 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
  - TI 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
- 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電力モードで操作

15.2.2 概要

図 15-1 に SPI マスターと 4 つのスレーブの例を示します。

図 15-1. 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 SCLK のエッジを使って MOSI ライン上のデータを取り込みます。またマスターによって取り込まれる MISO ライン上のデータも駆動します。

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 つの変種が 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。

15.2.3 SPI の動作モード

15.2.3.1 Motorola 社 SPI

オリジナルの SPI プロトコルは Motorola によって定義されました。これは全二重プロトコルです。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 SS ラインは「1」に維持され、SCLK は通常オフです。

Motorola 社 SPI のモード

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の 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の 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の 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 15-2 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 15-2. Motorola 社 SPI、4 モード

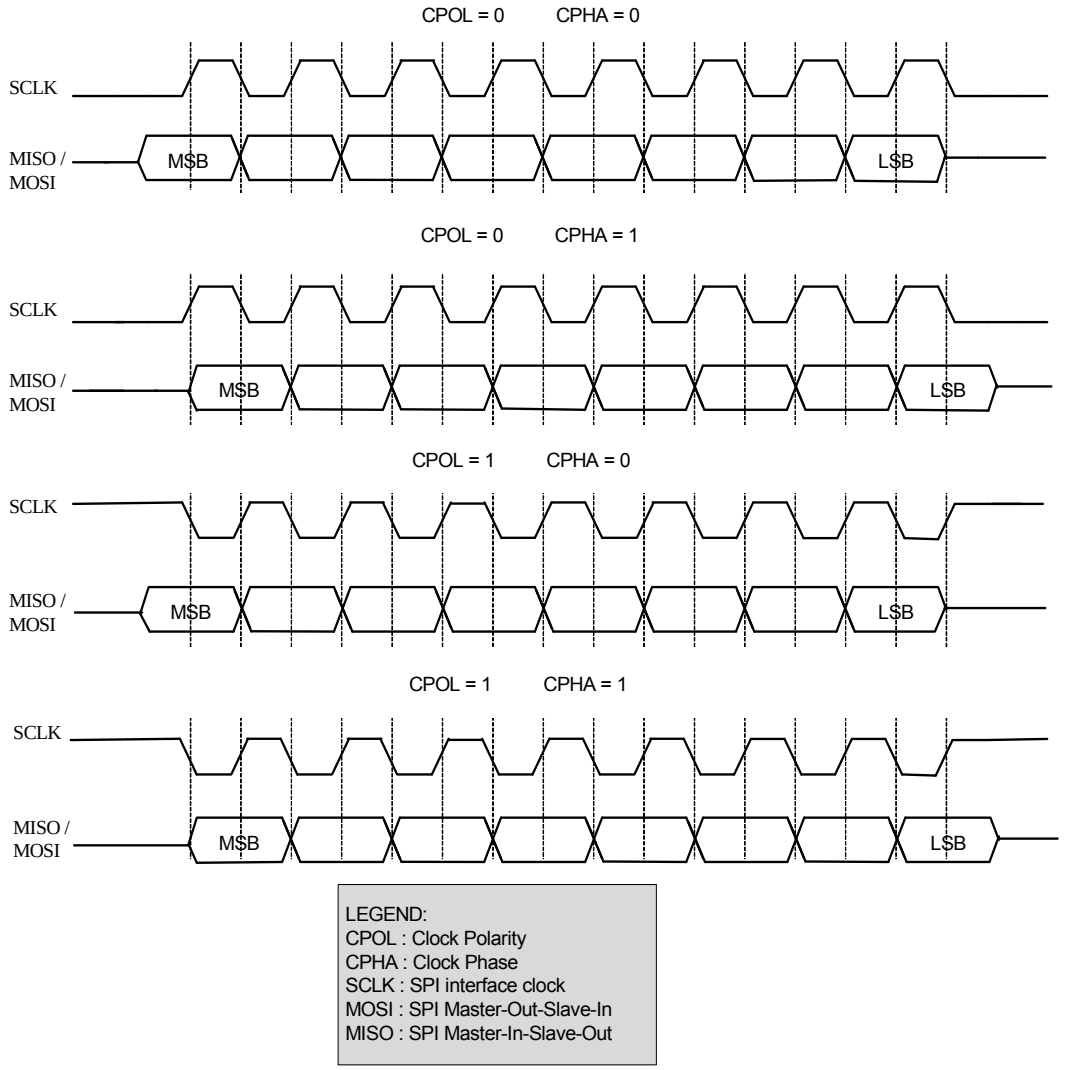
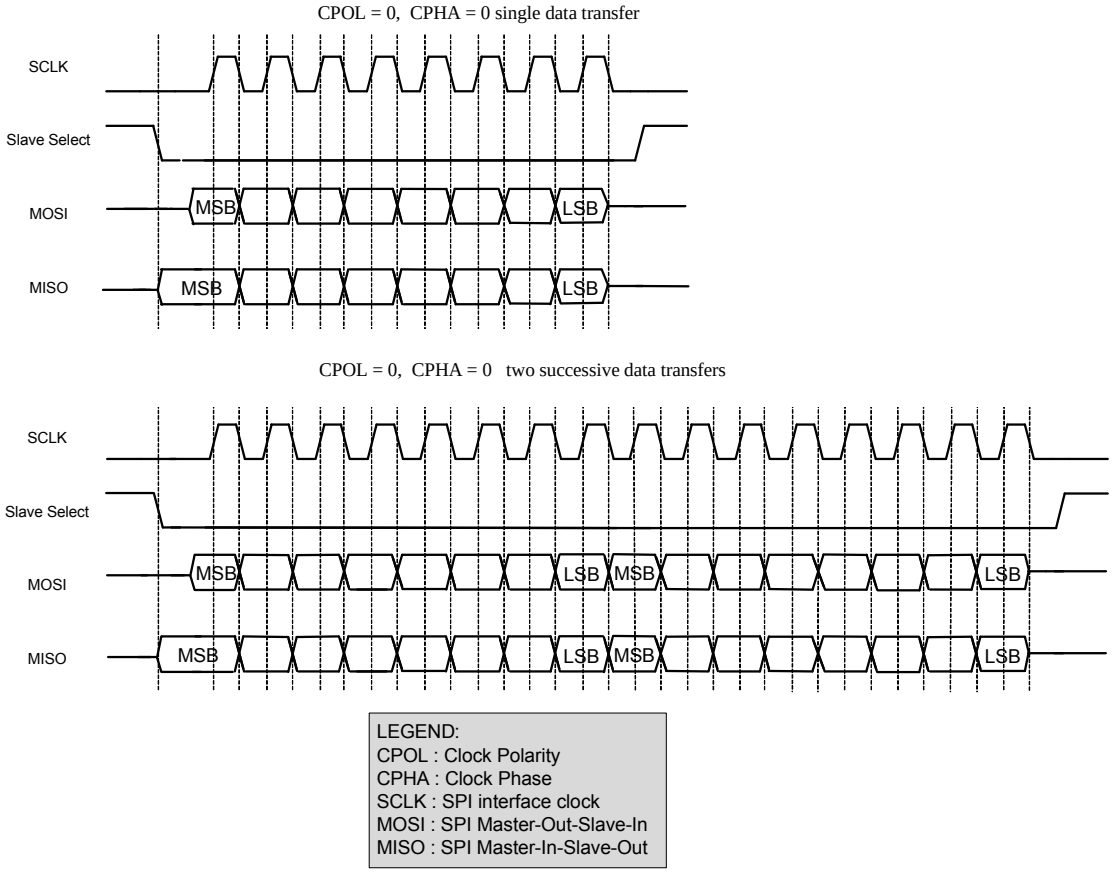


図 15-3 にモード 0 (CPOL = 0、CPHA = 0) での単一の 8 ビット データ転送と 2 つの連続した 8 ビット データ転送を示します。

図 15-3. Motorola 社 SPI のデータ転送例



SPI Motorola モード用の SCB の構成

SPI Motorola 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 を選択します。
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 モードを選択します。
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 の動作モードを選択します。
4. 107 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 を行います。

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については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

15.2.3.2 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しなければなりません。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 15-4 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注意してください。

図 15-4. TI 社 SPI のデータ転送例

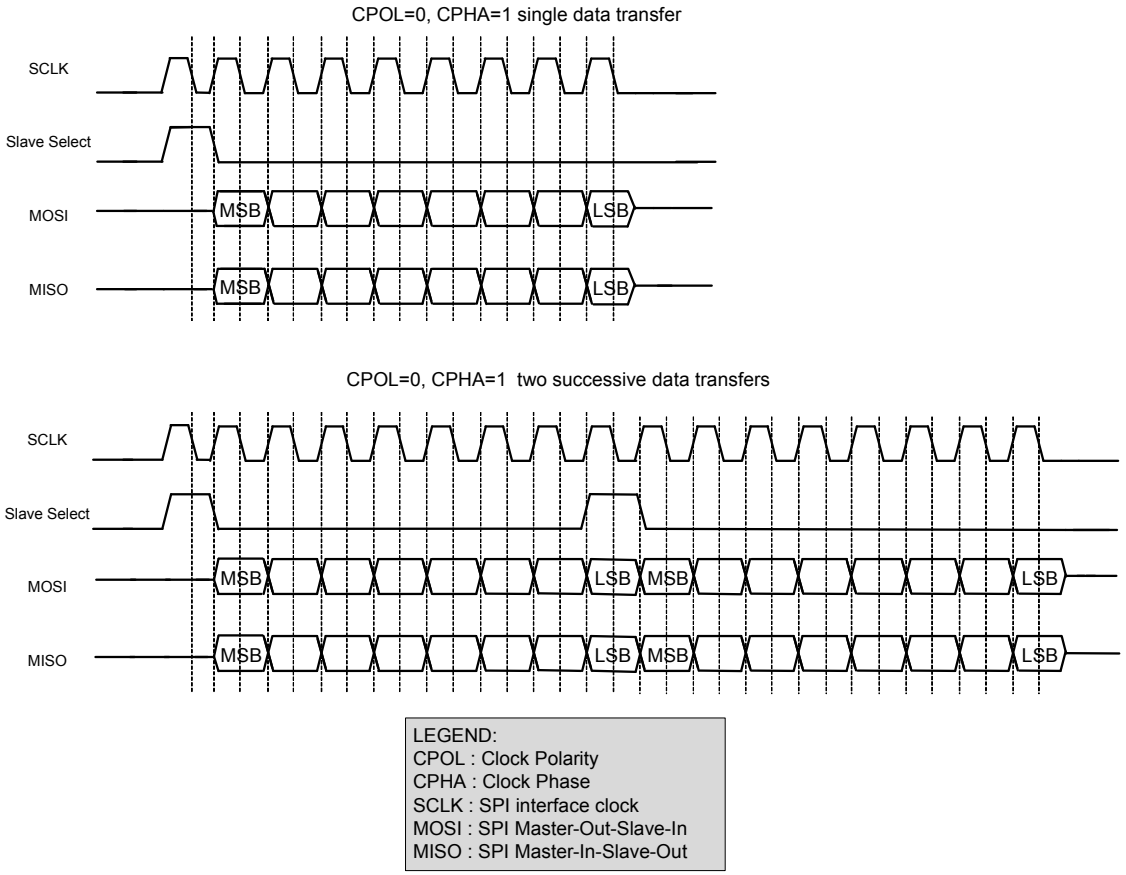
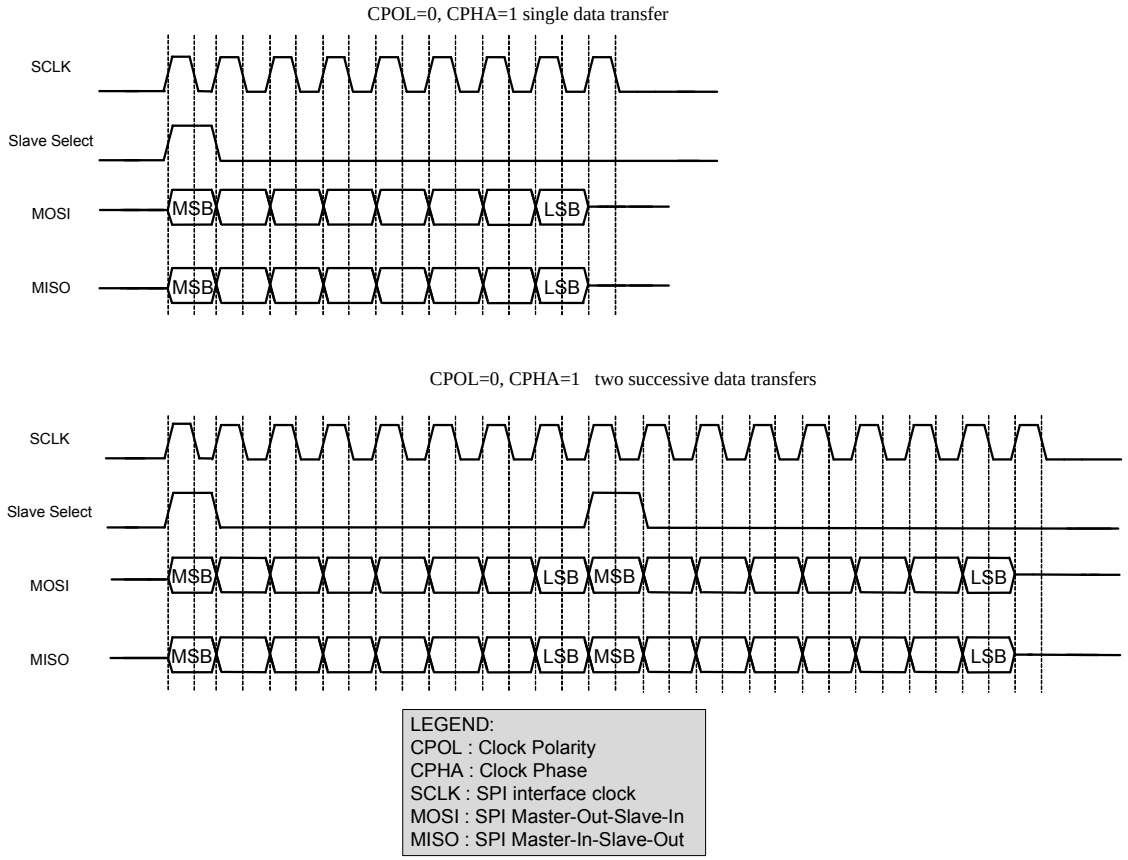


図 15-5 に単一の 8 ビット データ転送と 2 つの連続した 8 ビット データ転送を示します。SELECT パルスはフレームの最初のデータ ビットと同時に発生します。

図 15-5. 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 を選択します。
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 モードを選択します。
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します (「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. 107 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 を行います。

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については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

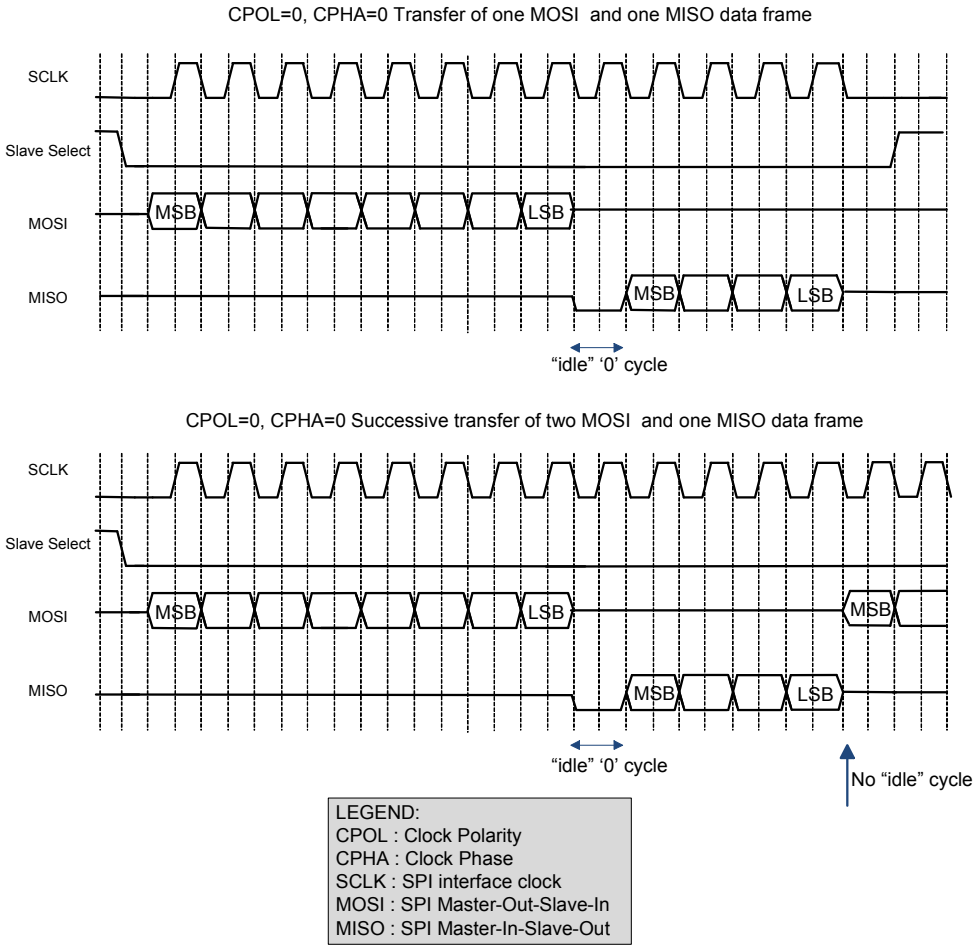
15.2.3.3 National Semiconductor 社 SPI

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 15-6 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 15-6. 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 を選択します。
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 モードを選択します。
3. 107 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 を行います。

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については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

15.2.4 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。

15.2.5 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 エントリのロケーション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 メモリを備えています。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。エントリ毎の 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。

15.2.5.1 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。

15.2.5.2 メモリ アレイ書き込み

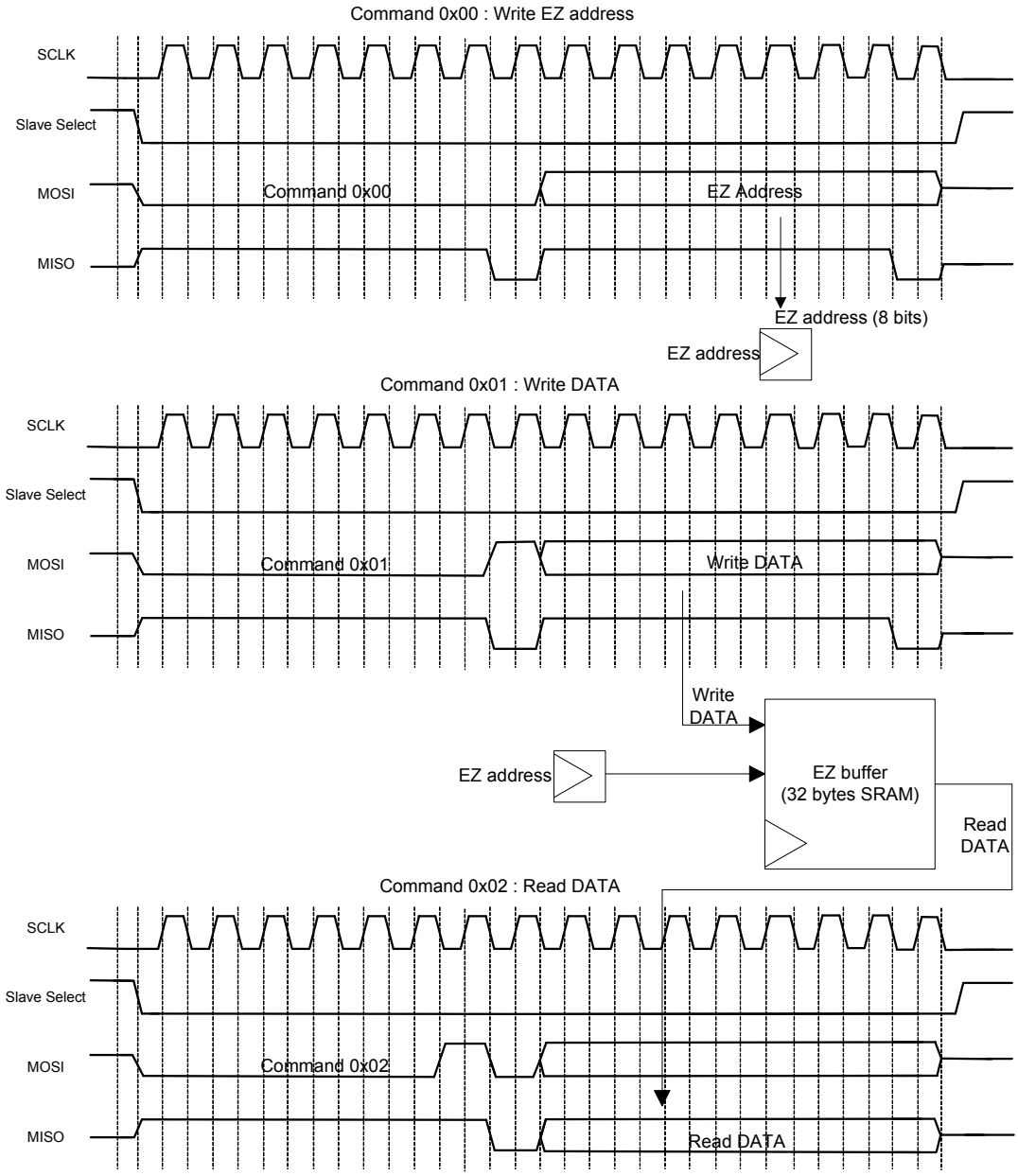
メモリ アレイ インデックスへの書き込みはメモリ アレイに書き込むマスターの意図を示す MOSI ライン上のコマンド バイト (0x01) から始まります。その後、スレーブはコマンドが監視された (0xFE) または監視されなかった (0xFF) ことを示すために MISO ライン上に返信バイトを駆動します。MOSI 上の追加書き込みのデータ バイトは通信した EZ アドレスによって示されるロケーションでメモリ アレイに書き込まれます。バイトがメモリ アレイに書き込まれると EZ アドレスはスレーブによって自動的にインクリメントされます。EZ アドレスがメモリ エントリの最大数 (32) を超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

15.2.5.3 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監視された (0xFE) または監視されなかった (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。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ロケーションでメモリ 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て 0 にラップ アラウンドしません。

図 15-7 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 15-7. EZSPI の例



15.2.5.4 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 を構成するには様々なレジスタ ビットを次の順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 モードを選択します。
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します。
3. 107 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 を行います。

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については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

15.2.6 SPI レジスタ

SPI インターフェースは表 15-1 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については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

表 15-1. 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、I2C) を選択し内部と外部クロック供給動作を選択
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
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 から除去される。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 から除去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 から除去されない。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ロケーション (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ロケーション (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 メモリ位置内のデータを格納

15.2.7 SPI 割り込み

SPI は内部割り込み要求にも外部の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

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。

15.2.8 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 15-2 に従って SCB_SPI_CTRL レジスタを使用して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 15-3 に示すように SCB_TX_CTRL と SCB_RX_CTRL レジスタを使用して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。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 であるかを指定します。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 15-4 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て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 レベルを設定します。
   b. トランスミッターとレシーバー FIFO、シフト レジスタをクリアします。
   c. TX と RX FIFO をフリーズします。
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します。にこれらのレジスタ ビットを示します。表 15-5
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 15-2. 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
		01	SPI Texas Instruments サブモード
		10	SPI National Semiconductors サブモード
		11	予約済み
31	MAS-TER_-MODE	0	スレーブ モード
		1	マスター モード

表 15-3. 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 ビット、ストップ ビットとパリティ 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 15-4. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL レジスタ

ビット	名称	説明
[7:0]	TRIG-GER_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 15-5. 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

15.2.9 内部と外部クロック供給 SPI 動作

SCB は SPI と I2C 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電力モードでの動作が可能です。

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67 ページのクロック供給システム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。ただし 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の有効な範囲は 6 ～ 16 となります。そのため最大ビットレートは 8Mbps になります。**注:** SPIマスター モードではLATE_MISO_SAMPLE を「1」にセットする必要があります。このビットの初期設定値は「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。MEDIAN ビットが「0」にセットされ LATE_MISO_SAMPLE ビットが「1」にセットされる時、SCB は 16Mbps の最大ビットレートに達することが可能です。ただし I/O セルと配線遅延を考慮すれば達成できる最大データ レートは 8Mbps となります。これらのビットに基づく最大ビットレートを表 15-6 に示します。

表 15-6. 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 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 15-7 と表 15-8 に (非 EZ モードと EZ モード) での SPI の設定を示します。

15.2.9.1 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 15-7 に可能なオプションをまとめます。

表 15-7. SPI 動作

	EC_OP_MODE = 0		EC_OP_MODE = 1	
システムの電力モード	**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		
ハイバネート	これらのモードで SCB は使用不可 (81 ページの電力モードを参照)			
ストップ				

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電力モードで機能しディープスリープのシステム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、割り込みを生成して 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。

15.2.9.2 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 15-8 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 15-8. EZ モードでの SPI 動作

SPI、EZ モード				
	EC_OP_MODE = 0		EC_OP_MODE = 1	
システムの電力モード	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
ハイバネート	これらのモードで SCB は使用不可 (81 ページの電力モードを参照)			
ストップ				

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電力モードで機能しディープスリープのシステム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、割り込みを生成して 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注意してください。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。

15.3 UART

汎用非同期レシーバ／トランスミッタ (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

15.3.1 特長

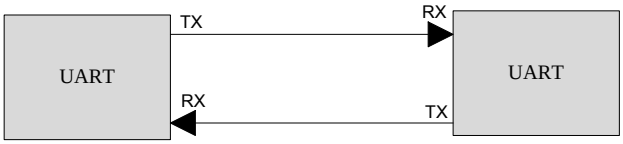
- 非同期トランスミッターとレシーバー機能
- 最大 1Mbps までのデータ レートをサポート
- UART プロトコルをサポート
  - 標準 UART
  - SmartCard (ISO7816) リーダー
  - IrDA
- ローカル インターコネクト ネットワーク (LIN) をサポート
  - ブレーク検出
  - ボーレート検出
  - 衝突検出 (駆動するビット値がバスに反映されず、他のコンポーネントが同じバスを駆動していることを検出する)
- マルチ プロセッサ モード

- 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

15.3.2 概要

図 15-8 に標準 UART TX と RX を示します。

図 15-8. 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その自身のクロックを使用し 1 ビット転送周期について合意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 15-9 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電力モードでのみ動作します。

15.3.3 UART の動作モード

15.3.3.1 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。

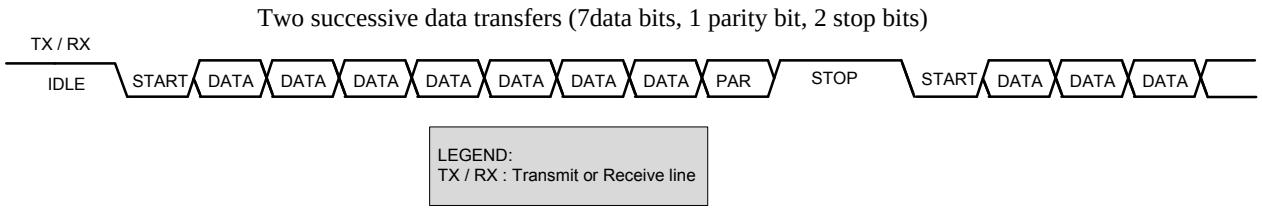
インターフェースがクロックを持たないため、トランスミッターとレシーバーはビット転送の周期について合意する必要があります。トランスミッターとレシーバーはその自身の内部クロックを持ちます。レシーバー クロックがビット転送周波数より高い周波数で動作するため、レシーバーは受信信号をオーバーサンプリングすることがあります。

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。

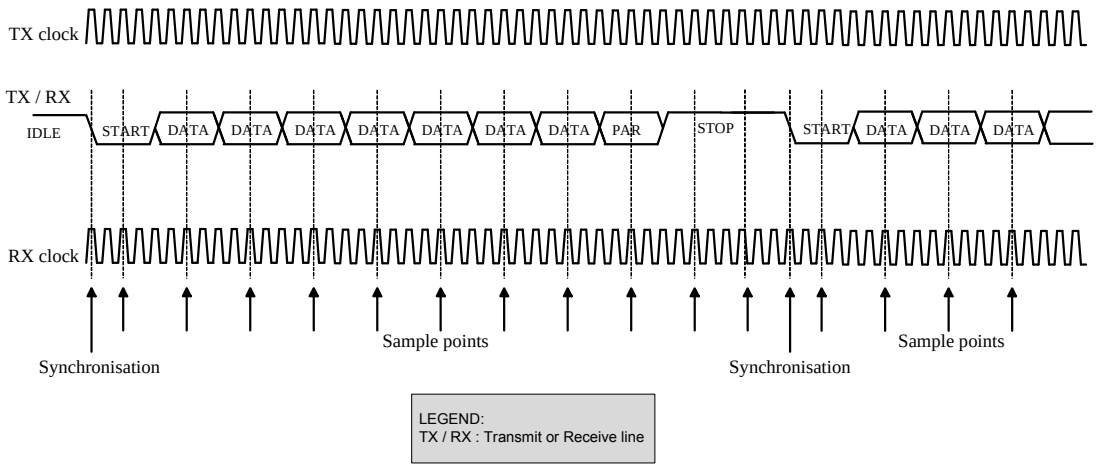
図 15-9 に UART プロトコルを示します。

図 15-9. 標準プロトコルの例である UART



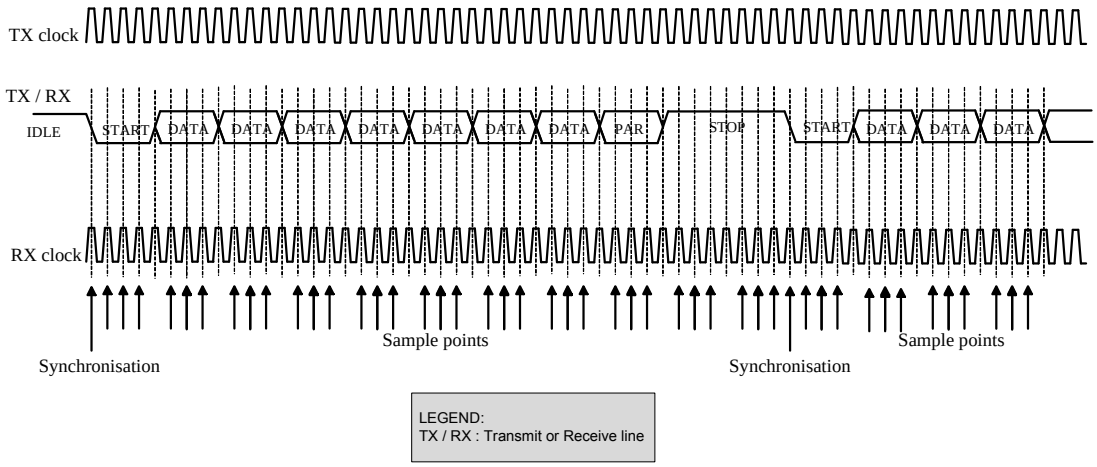
レシーバーは受信信号をオーバーサンプリングします。ビット転送周期の中央にある (レシーバー クロック上の) サンプル ポイントの値が使用されます図 15-10 にそれを示します。

図 15-10. 標準プロトコルの例である UART (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 15-11 にそれを示します。

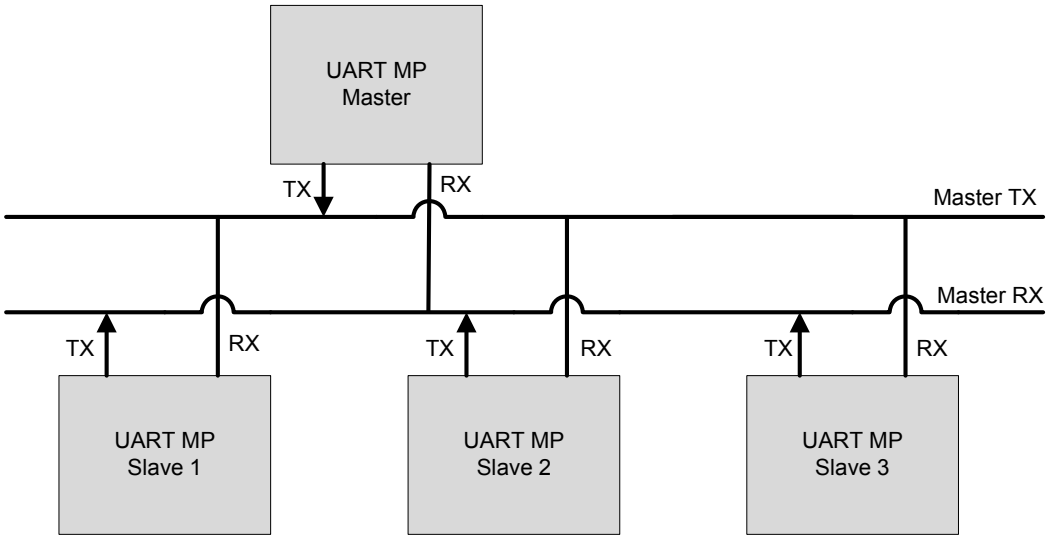
図 15-11. 標準プロトコルの例である UART (複数サンプル)



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 15-12 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。

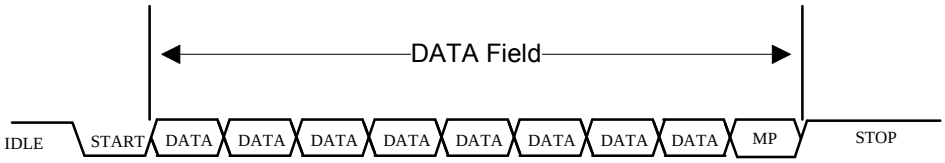
図 15-12. UART MP モードのバス接続



UART_MP モードの主なプロパティは次のとおりです。

- 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される
- 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 15-13 にデータ フレームを示す。
- パリティ ビットは無効

図 15-13. UART MP データ フレーム

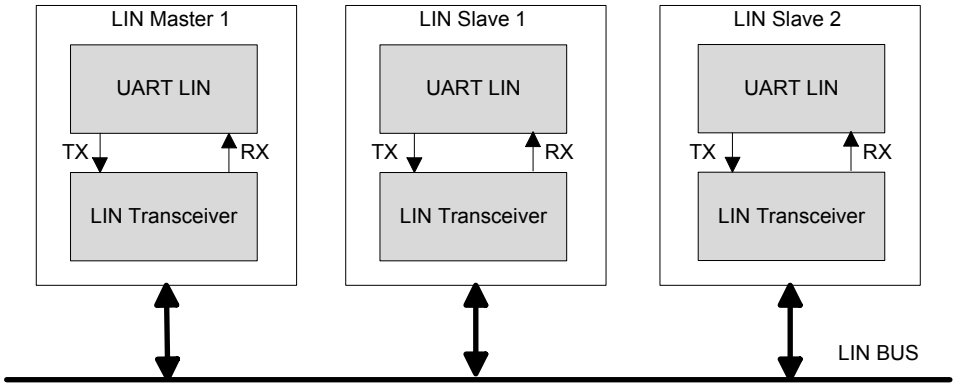


SCB は UART_MP モードでマスターまたはスレーブ デバイスとして使用可能です。SCB_TX_CTRL と SCB_RX_CTRL レジスタは両方とも 9 ビットのデータ フレーム サイズに設定できます。SCB が UART_MP マスター デバイスとして動作する時にファームウェアはすべてのアドレスまたはデータ フレーム用の MP フラグを変更します。このブロックが UART_MP スレーブ デバイスとして機能する時に SCB_UART_RX_CTRL レジスタの MP_MODE フィールドを「1」にセットする必要があります。SCB_RX_MATCH レジスタはスレーブ アドレスとアドレス 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の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 トポロジで定義されています。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 15-14 に UART_LIN と LIN トランシーバーを示します。

図 15-14. UART_LIN と LIN トランシーバー

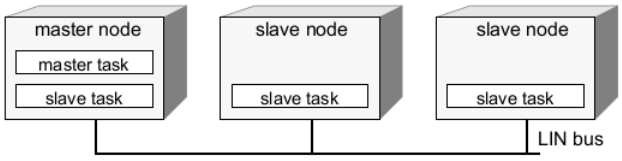


LIN プロトコルは以下の 2 タスクを定義します。

- マスター タスク : このタスクはヘッダー パケットを送信して LIN 転送を開始することを含みます。
- 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んでいます。

図 15-15 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 15-15. LIN バスのノードとタスク

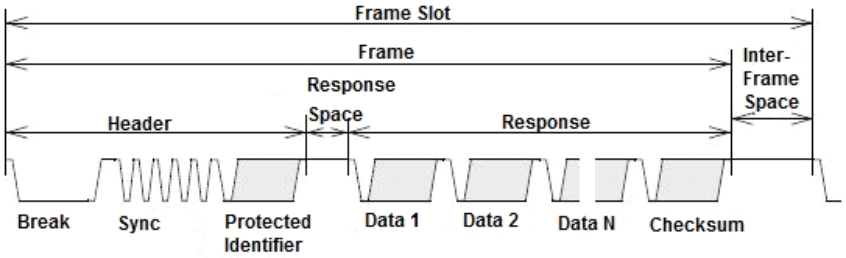


LIN フレームの構造

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 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です。
  - 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成ります。

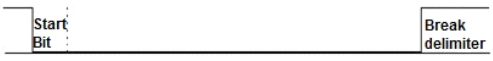
図 15-16. LIN フレームの構造



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 15-17 に示します。

図 15-17. 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 15-18 に LIN 同期フィールドの構造を示します。

図 15-18. LIN 同期フィールド

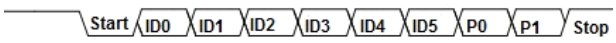


保護識別子 (PID) フィールド： 保護識別子フィールドはフレーム識別子 (ビット 0 ～ 5) とパリティ (ビット 6 と 7) という 2 つのサブフィールドから構成されます。PID フィールド構造を図 15-19 に示します。

- 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) は診断と構成データを含む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 15-19 に PID フィールド構造を示します。

図 15-19. 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を言及します。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で、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とおりです。

- 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のクエリーを行って、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。

予約済みフレーム： 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て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／レシーバが必要となります。LIN バス上の電源電圧範囲は 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します。
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します。
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) に「1」を書き込みます。
4. 118 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 を行います。

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については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

15.3.3.2 SmartCard (ISO7816)

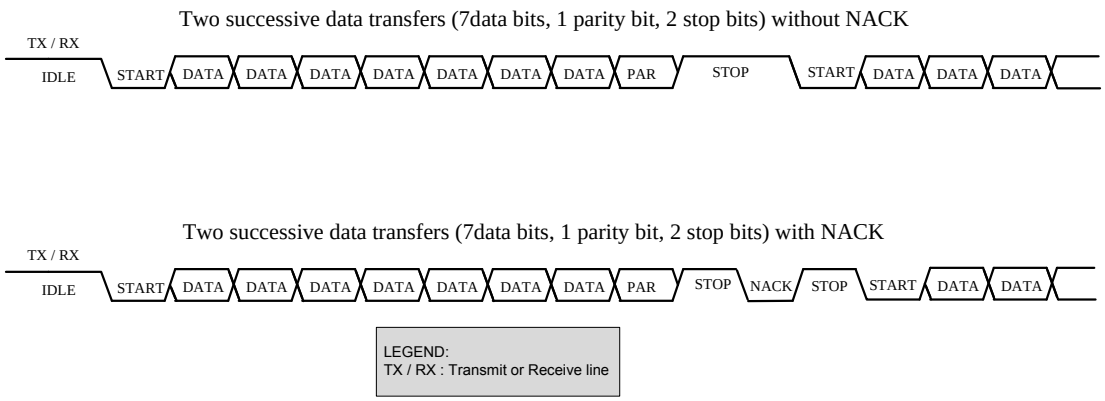
ISO7816 はシングル マスター シングル スレーブ トポロジで定義される非同期シリアル インターフェースです。ISO7816 はリーダ (マスター) とカード (スレーブ) 機能の両方を定義します。詳細については ISO7816 仕様を参照してください。マスター (リーダ) 機能のみは SCB によってサポートされます。このブロックは非同期文字送信を使用して基本物理層のサポートを提供します。UART_TX と UART_RX コントロール モジュール間の内部多重化で UART_TX ラインが SmartCard IO ラインに接続されます。

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 (と任意でパリティ 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 ビット転送周期長いことに注意してください。実装は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 15-20 に SmartCard プロトコルを示します。

図 15-20. 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については「 PSoC 4100M/4200M 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します。
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します。
3. 118 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 を行います。

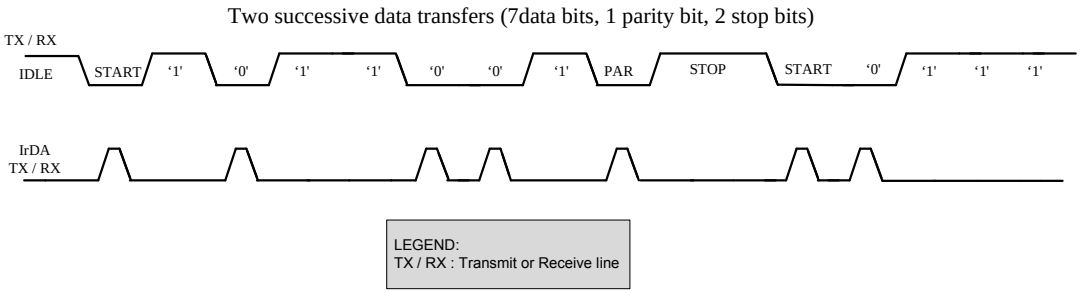
15.3.3.3 IrDA

SCB は UART インターフェースを使用して赤外線データ協会 (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、このブロックをインスタンス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 通信システムを実装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 15-21 に UART 転送の IrDA による変調方法を示します。

図 15-21. 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注意してください。これらのレジスタの詳細については「 PSoC 4100M/4200M 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します。
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します。
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 フィルターを有効にします。
4. 118 ページの UART の有効化と初期化 で説明したように SCB を構成します。

15.3.4 UART レジスタ

UART インターフェースは表 15-9 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について

は「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

表 15-9. 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、I2C) 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_C-TRL	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

15.3.5 UART 割り込み

UART は内部割り込み要求にも外部の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

15.3.6 UART の有効化と初期化

UART を以下の順でプログラムしてください。

1. 表 15-10 に記載する SCB_SPI_CTRL レジスタ情報を使用して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 レシーバ機能の選択動作を含んでいます。
2. 表 15-11 に記載する SCB_TX_CTRL と SCB_RX_CTRL レジスタ情報を使用して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 15-12 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て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 レベルを設定します。
   b. トランスミッターとレシーバーFIFO、シフト レジスタをクリアします。
   c. TX と RX FIFO をフリーズします。
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。表 15-13 に示すように動作モードを選択します。

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ってください。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 15-10. 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 15-11. 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 ビット、ストップ ビットとパリティ 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 15-12. 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 15-13. 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

本節は PSoC 4 における 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 バス仕様書を参照してください。

15.3.7 特長

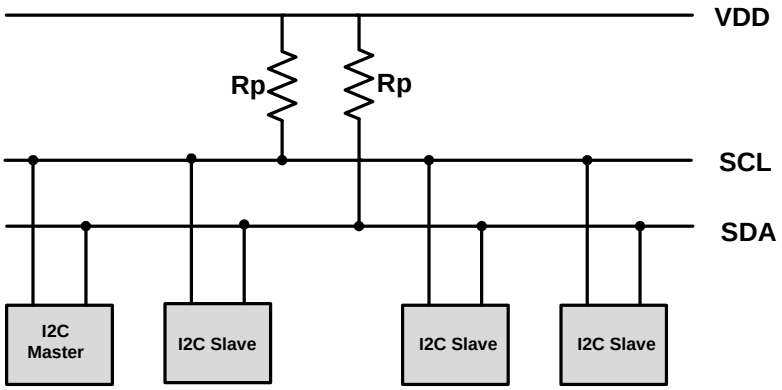
このブロックは以下の機能をサポートします。

- 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 フィルターを実装することでエラーを減少
- 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

15.3.8 概要

図 15-22 に I2C 通信ネットワークの例を示します

図 15-22. 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てこれらのラインに接続されて、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どちらとしても動作できます。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 4 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。

15.3.9 用語および定義

表 15-14 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 15-14. 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	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

15.3.9.1 バス ストール (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実装により、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。それからマスター 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。

15.3.9.2 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実装します。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発見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実装により「0」になります。マスター 1 が不一致を検出して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。

15.3.10 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 15-15 に I2C 動作モードを示します。

表 15-15. I2C モード

モード	説明
スレーブ	スレーブ専用の動作 (初期設定)
マスター	マスター専用の動作
マルチ マスタ	バス上で 2 つ以上のマスタをサポート
マルチ マスタ スレーブ	スレーブおよびマルチマスタの同時動作

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 15-16 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。書き込み転送と書き込み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 15-16. 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 にプルダウンしてクロック 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

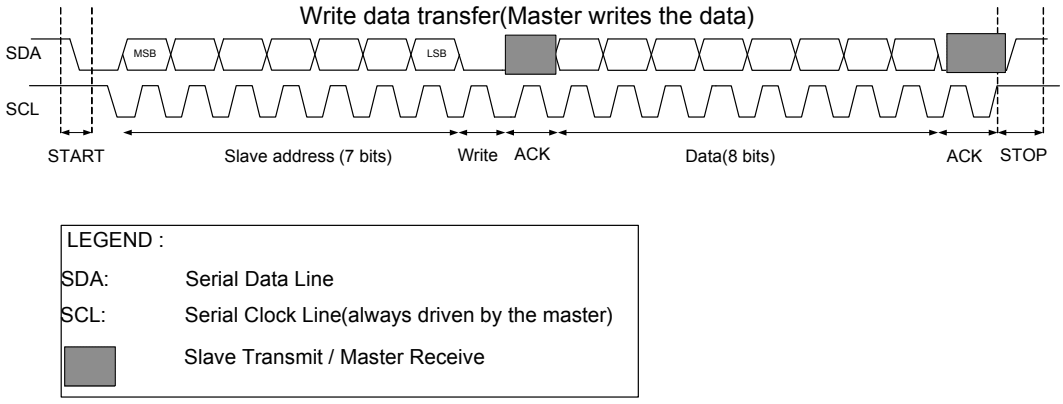
マルチマスター モードで動作している時、バスはビジーであるか確認される必要があります (他のマスターがスレーブと通信しているかもしれません)。この場合、マスターは START 信号を発信する前に現在の処理が完了するまで待機しなければなりません (表 15-16、図 15-23、図 15-24) を参照)。マスターは、データ転送開始の合図として STOP 信号を探します。

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 129 ページのマスター モード転送の例 、131 ページのスレーブ モード転送の例 および 135 ページのマルチマスター モード転送の例 に記載します。

15.3.10.1 書き込み転送

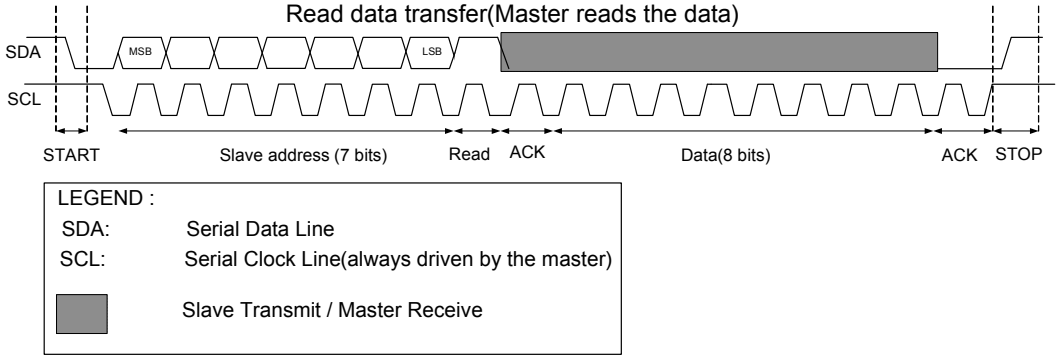
図 15-23. マスター書き込みデータ転送



- 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、プルアップ抵抗実装により 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。

15.3.10.2 書き込み転送

図 15-24. 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 アドレスと書き込みビット (「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、プルアップ抵抗実装により 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。

15.3.11 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 プロトコルを囲むソフトウェア 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 位置) 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 5 ビットはこの 32 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 16 ワードのハードウェア 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 位置を有します。EZ モードでは、FIFO は 8 ビット幅 32 位置のシングル 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 EZ 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。

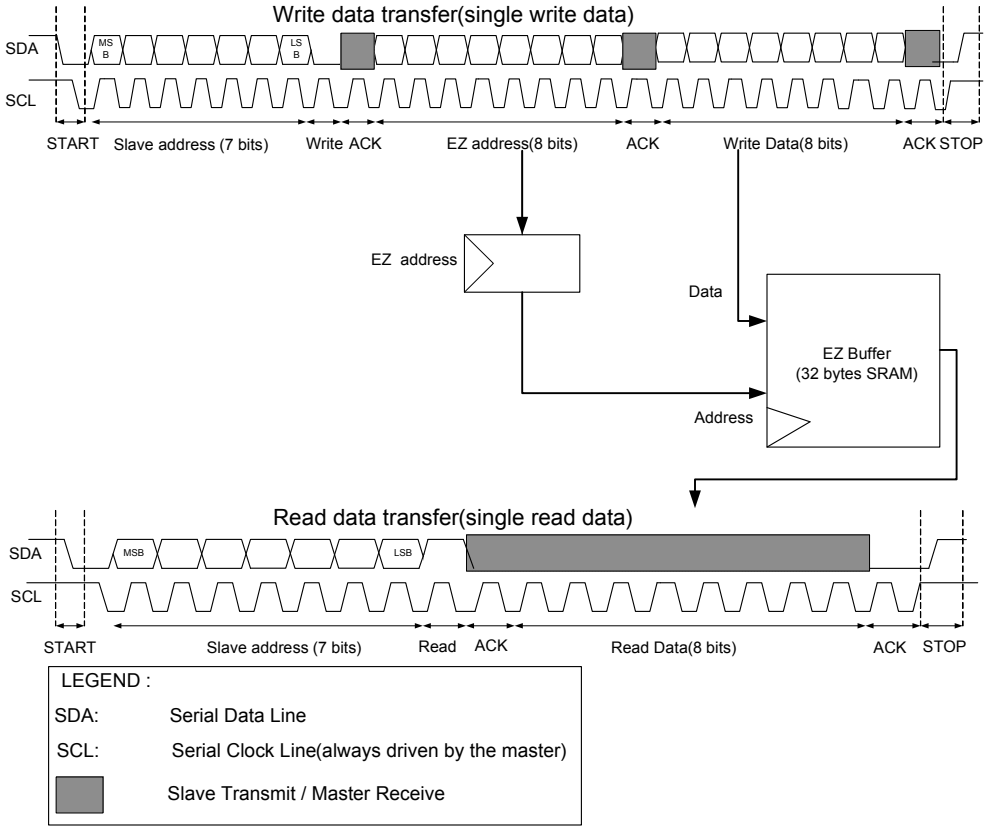
15.3.11.1 メモリ アレイ書き込み

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。

15.3.11.2 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るものです。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。

図 15-25. EZI2C 書き込みおよび読み出しデータ転送



15.3.12 I2C レジスタ

表 15-17 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 15-17. I2C レジスタ

レジスタ	機能
SCB_CTRL	SCBI2C 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 15-17. 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ロケーション (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ロケーション (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 から除去される。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 から除去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 から除去されない。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: I2Cレジスタ ビットの詳細説明については、「PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

15.3.13 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については「PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

15.3.14 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。

15.3.14.1 I2C 標準 (非 EZ) モード

I2C は次の順でプログラムしなければなりません。

1. 表 15-18 に記載する SCB_I2C_CTRL レジスタ情報を使用してプロトコル固有の情報をプログラムします。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 15-19 に記載する SCB_TX_CTRL と SCB_RX_CTRL レジスタ情報を使用して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／受信されるビットを MSB に指定します。
3. 表 15-20 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て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 レベルを設定します。
   b. トランスミッターとレシーバーFIFO、シフト レジスタをクリアします。
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 モードを選択します。これらのレジスタ ビットは表 15-21 に示します。I2C レジスタの完全な説明については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」を参照してください。

表 15-18. 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 15-19. 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 15-20. 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 15-21. 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

15.3.14.2 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 モードを選択します。
2. I2C 標準 (非 EZ) モードで述べたステップ 2 ～ステップ 4 を行います。
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します。

15.3.15 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 67 ページのクロック供給システム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 15-22 と表 15-23 に非 EZ モードと EZ モードでの I2C の設定を示します。

15.3.15.1 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 機能にサポートされません。それで EC_OP_-MODE は非 EZ モードのために「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 15-22 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、スリープディープスリープの電力モードで外部クロック供給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発見すると、割り込みを生成して 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。

表 15-22. EZ モード以外での I2C 動作

I2C (EZ 以外) モード				
システムの電力モード	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		
ハイバネート	SCB はこれらのモードで使用不可 (81 ページの電力モードを参照)			
ストップ				

- 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」)。

15.3.15.2 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」にセットしなければなりません。表 15-23 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 15-23. EZ モードでの I2C 動作

I2C、EZ モード				
システムの電力モード	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注意してください。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。

15.3.16 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。

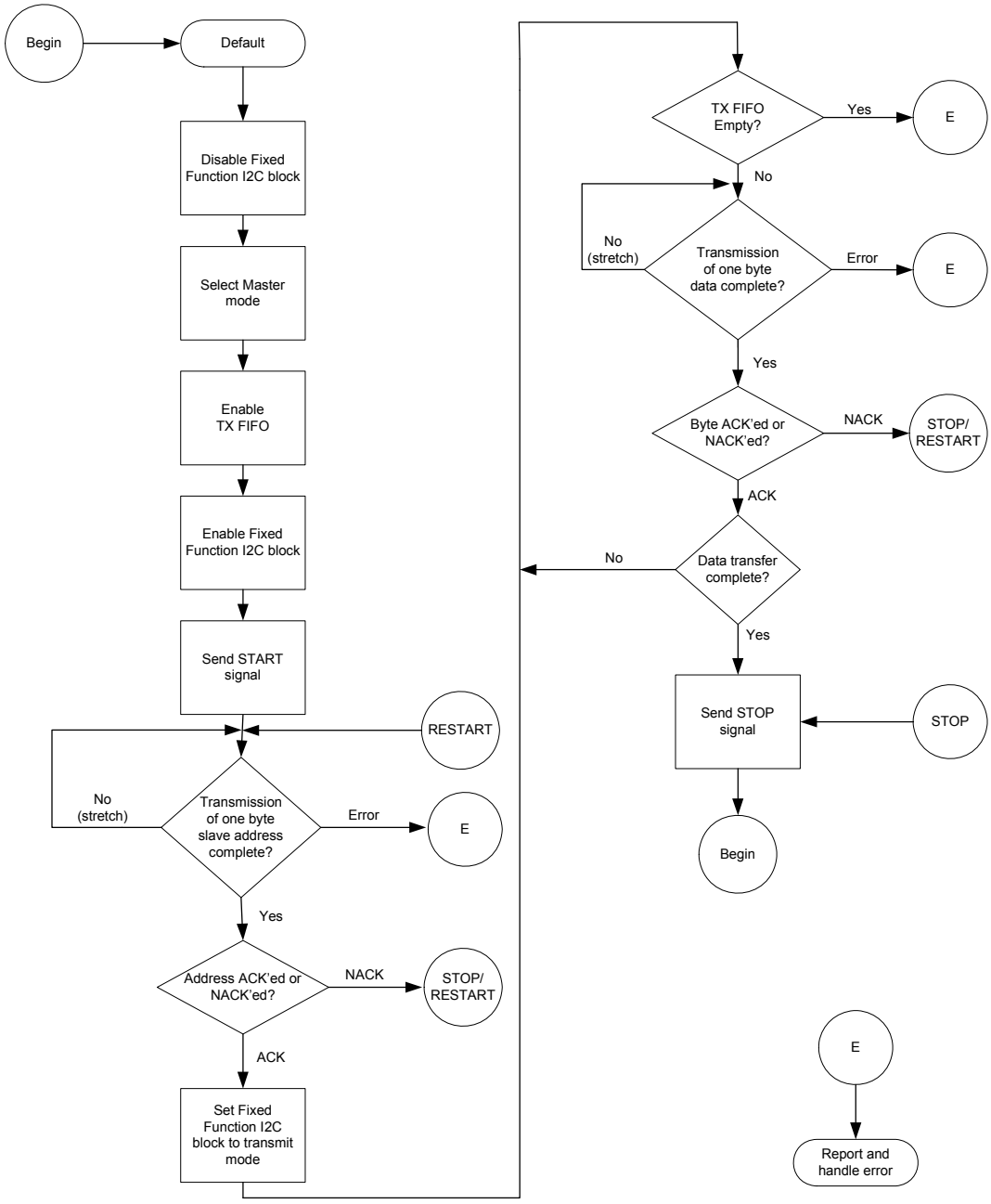
注 : スリープ モードに切り替えている時に、I2C がスレーブ アドレス一致イベントでデバイスを復帰させるために、SCB_INTR_I2C_EC レジスタの割り込みビット WAKE_UP (ビット 0) を有効にする必要があります。

15.3.17　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。

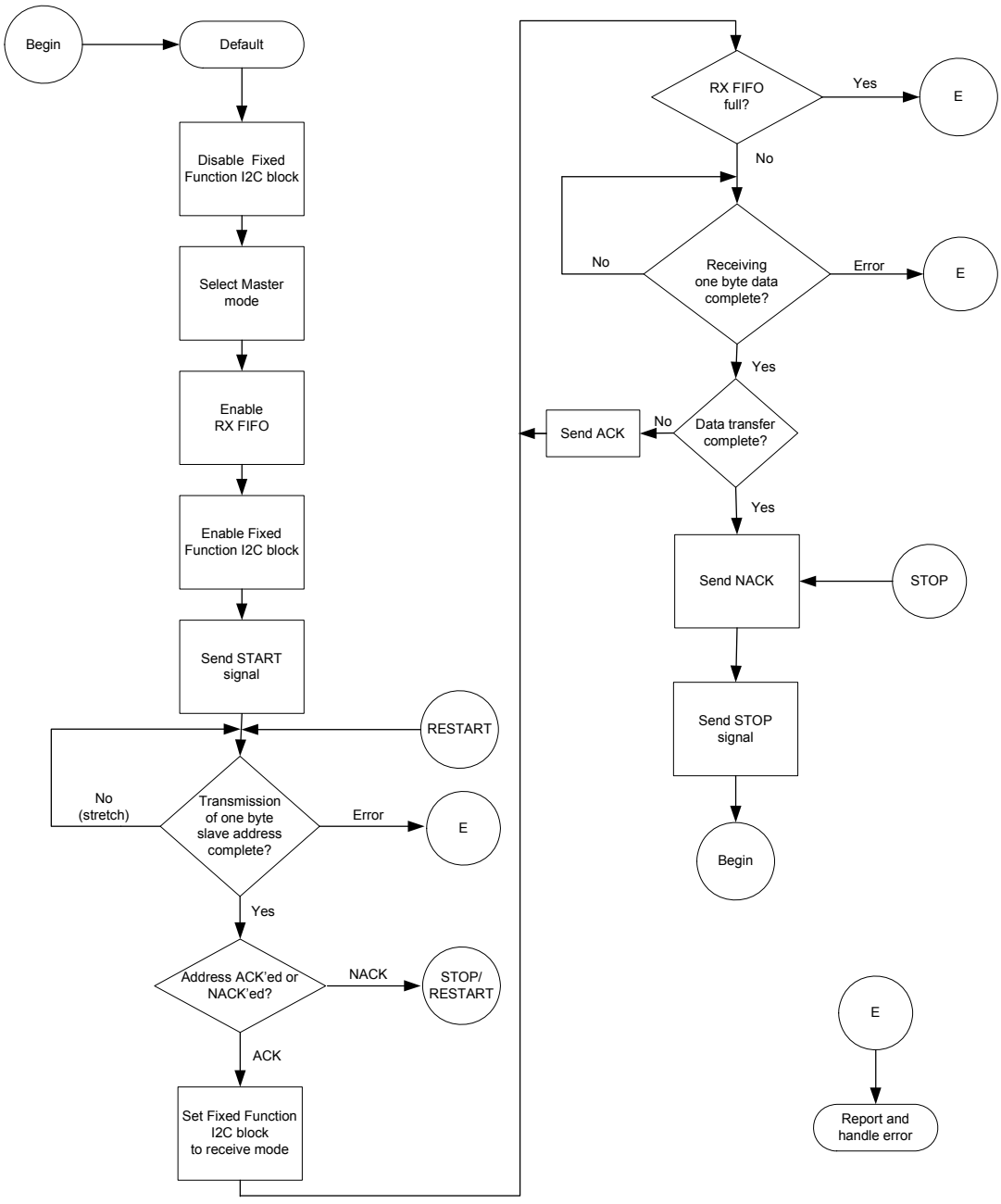
15.3.17.1　マスター送信

図 15-26. 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



15.3.17.2 マスター受信

図 15-27. 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

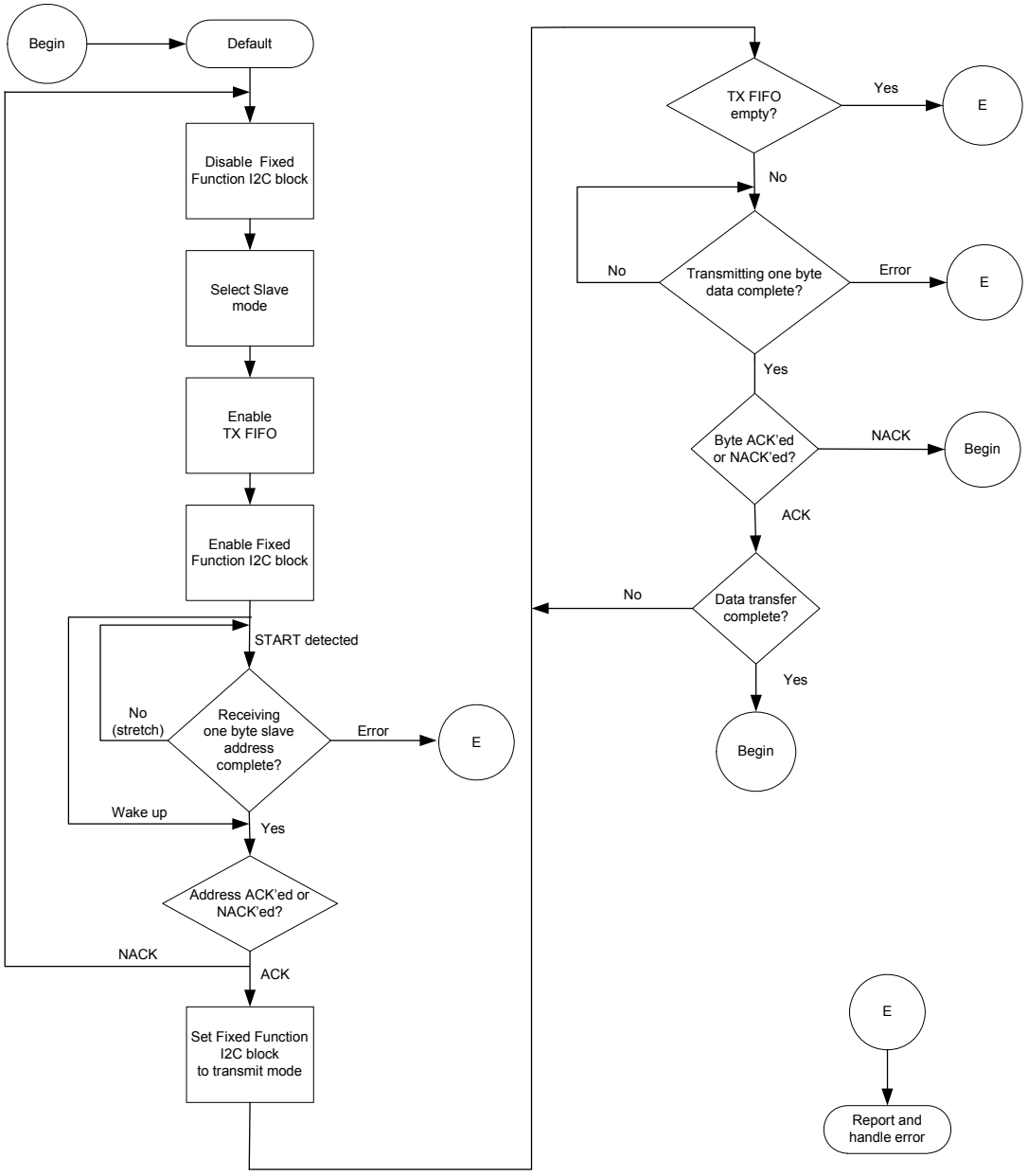


15.3.18　スレーブ モード転送の例

マスターがデータを送信または受信します。

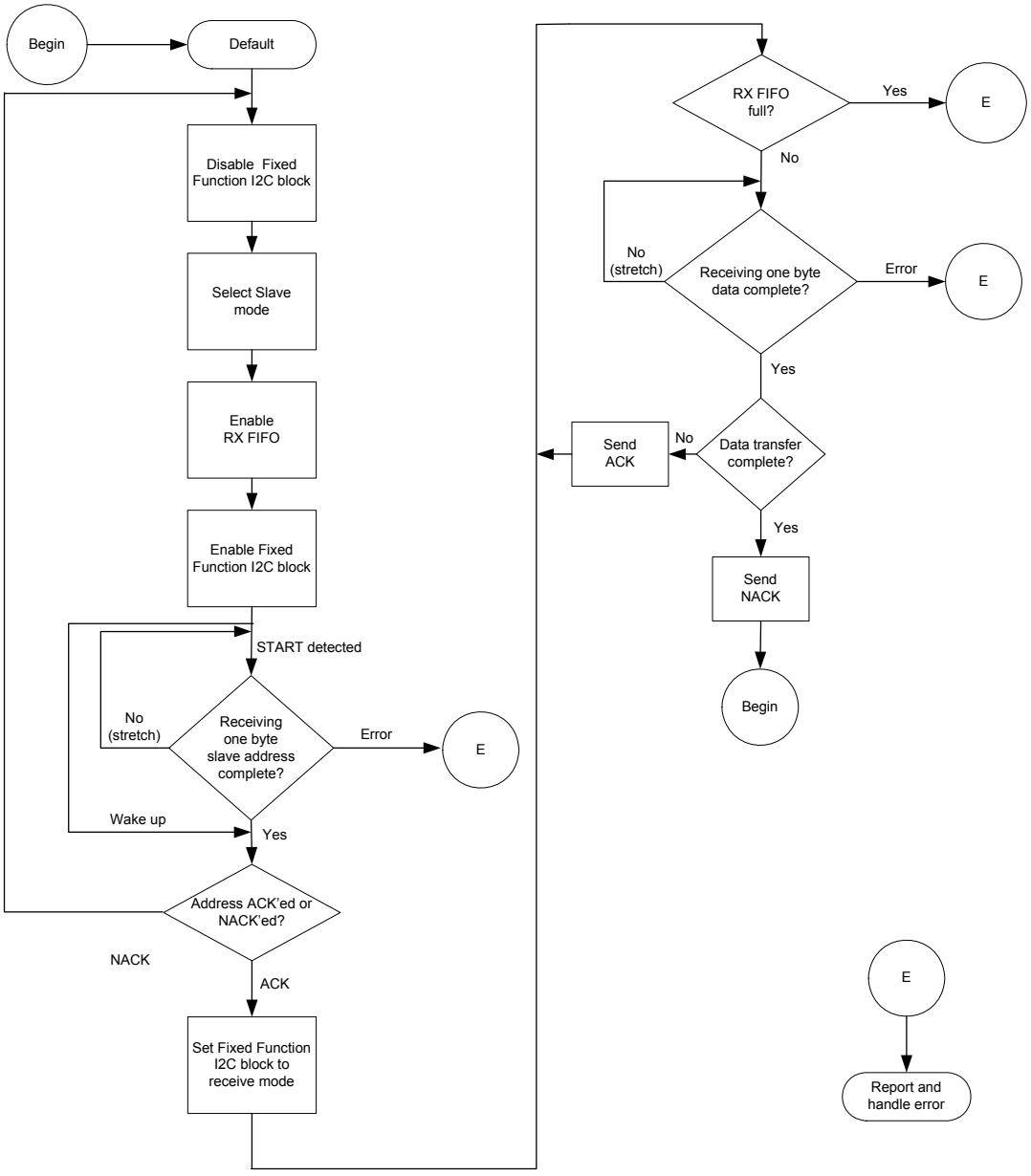
15.3.18.1　スレーブ送信

図 15-28. スレーブ モードでの書き込み動作のフローチャート



15.3.18.2 スレーブ受信

図 15-29. スレーブ モードでの読み出し動作のフローチャート

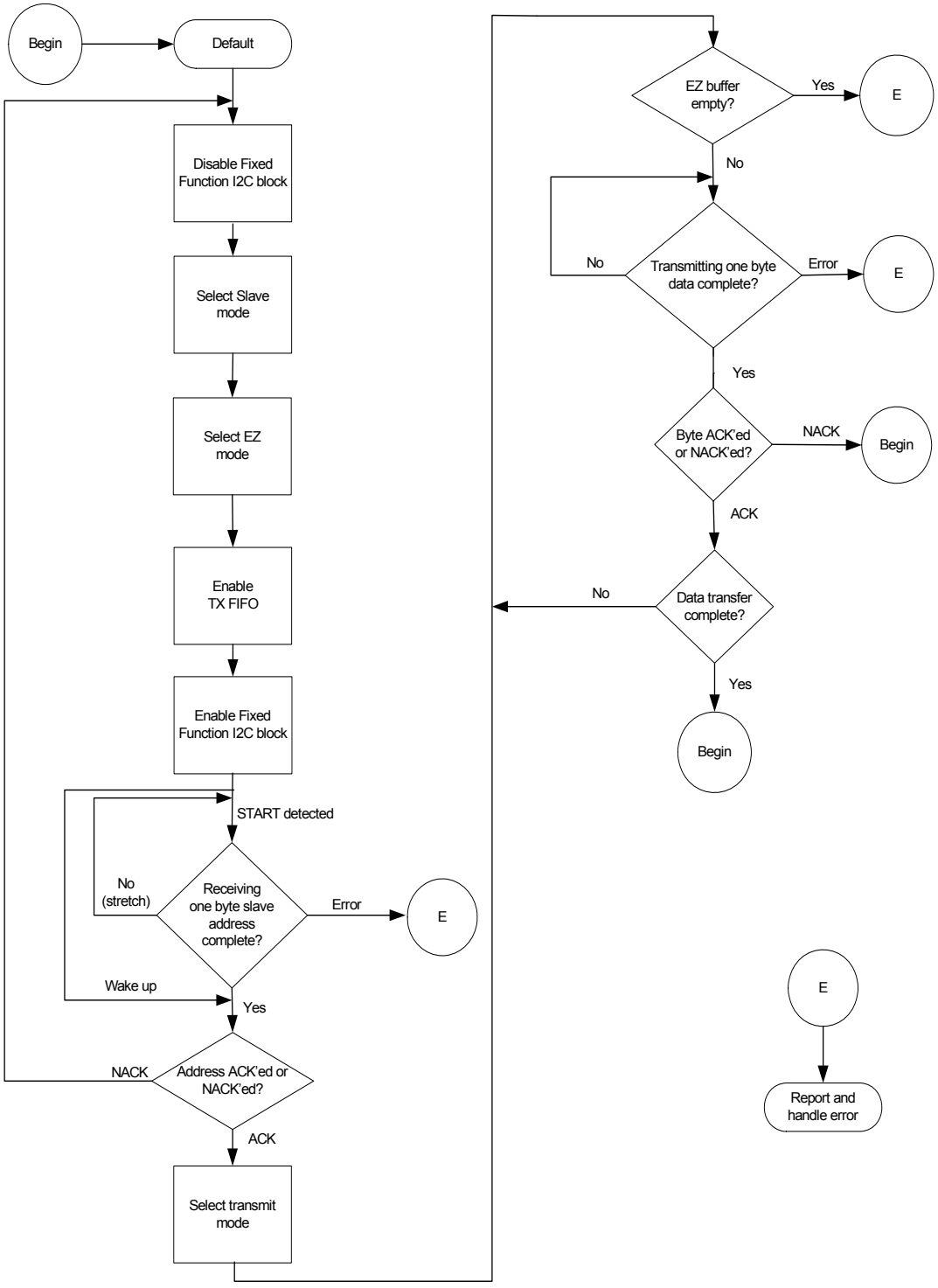


15.3.19　EZ スレーブ モード転送の例

EZ スレーブがデータを送信または受信します。

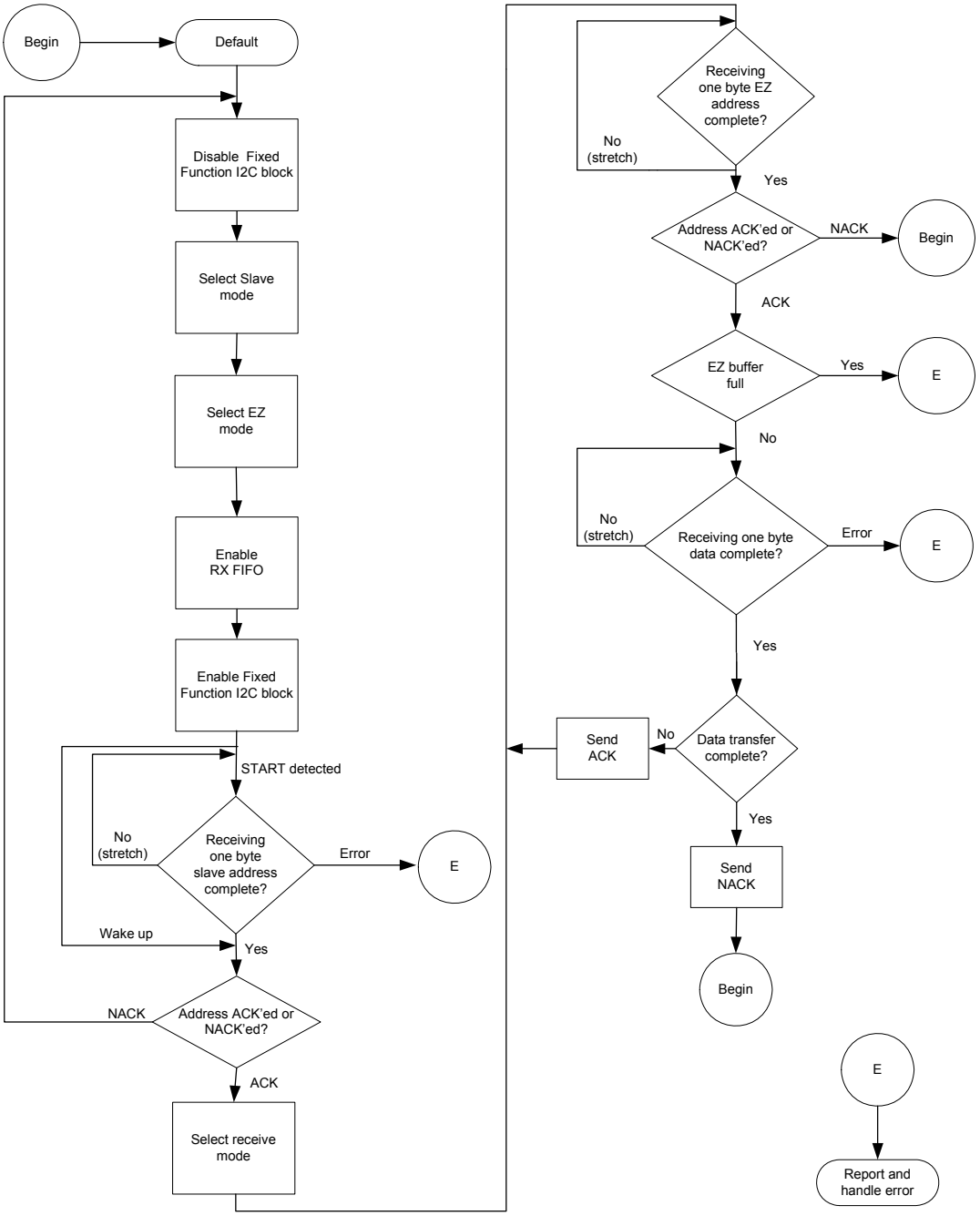
15.3.19.1　EZ スレーブ送信

図 15-30. EZI2C スレーブ モードでの書き込み動作のフローチャート



15.3.19.2 EZ スレーブ受信

図 15-31. EZI2C スレーブ モードでの読み出し動作のフローチャート

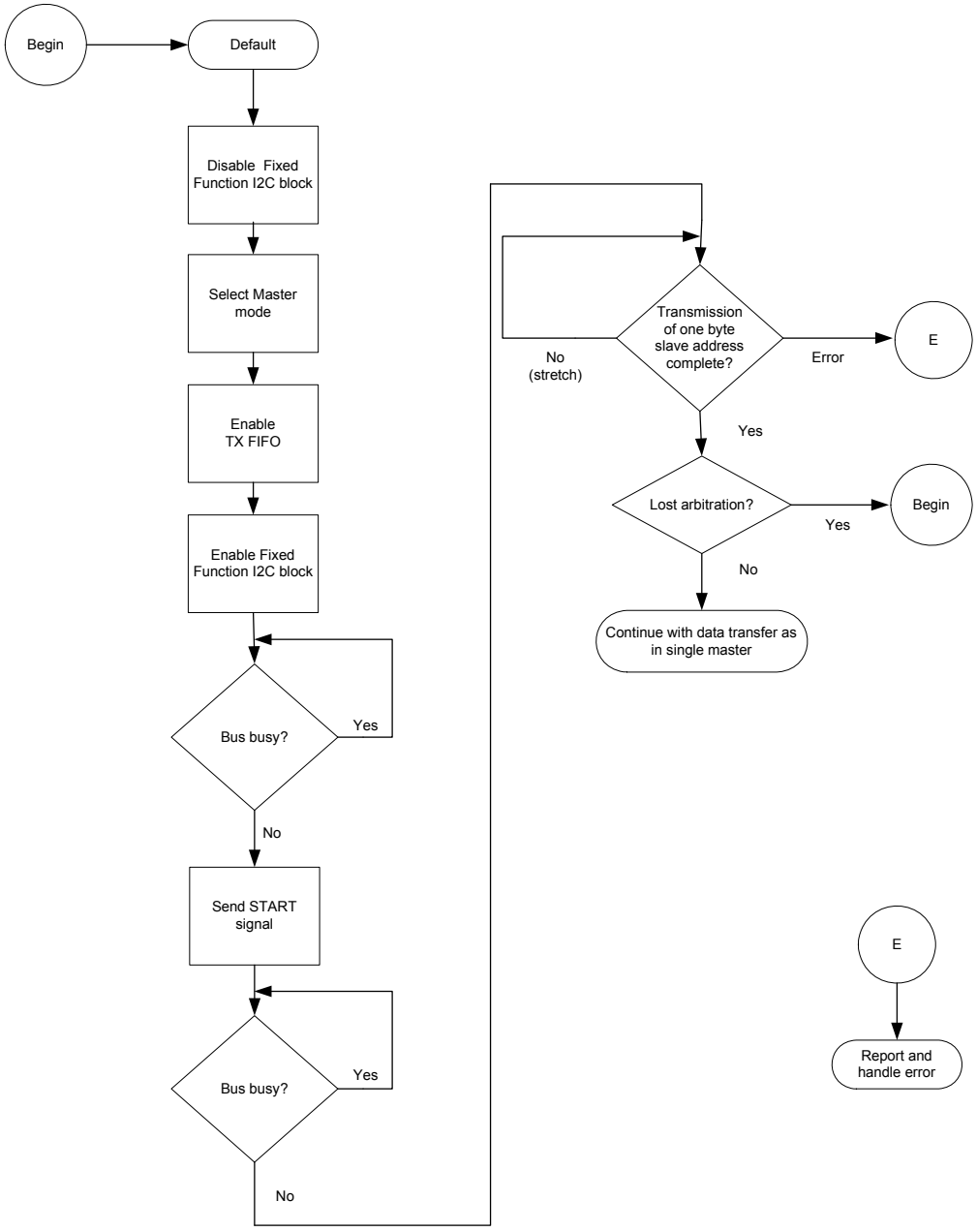


15.3.20　マルチマスター モード転送の例

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。

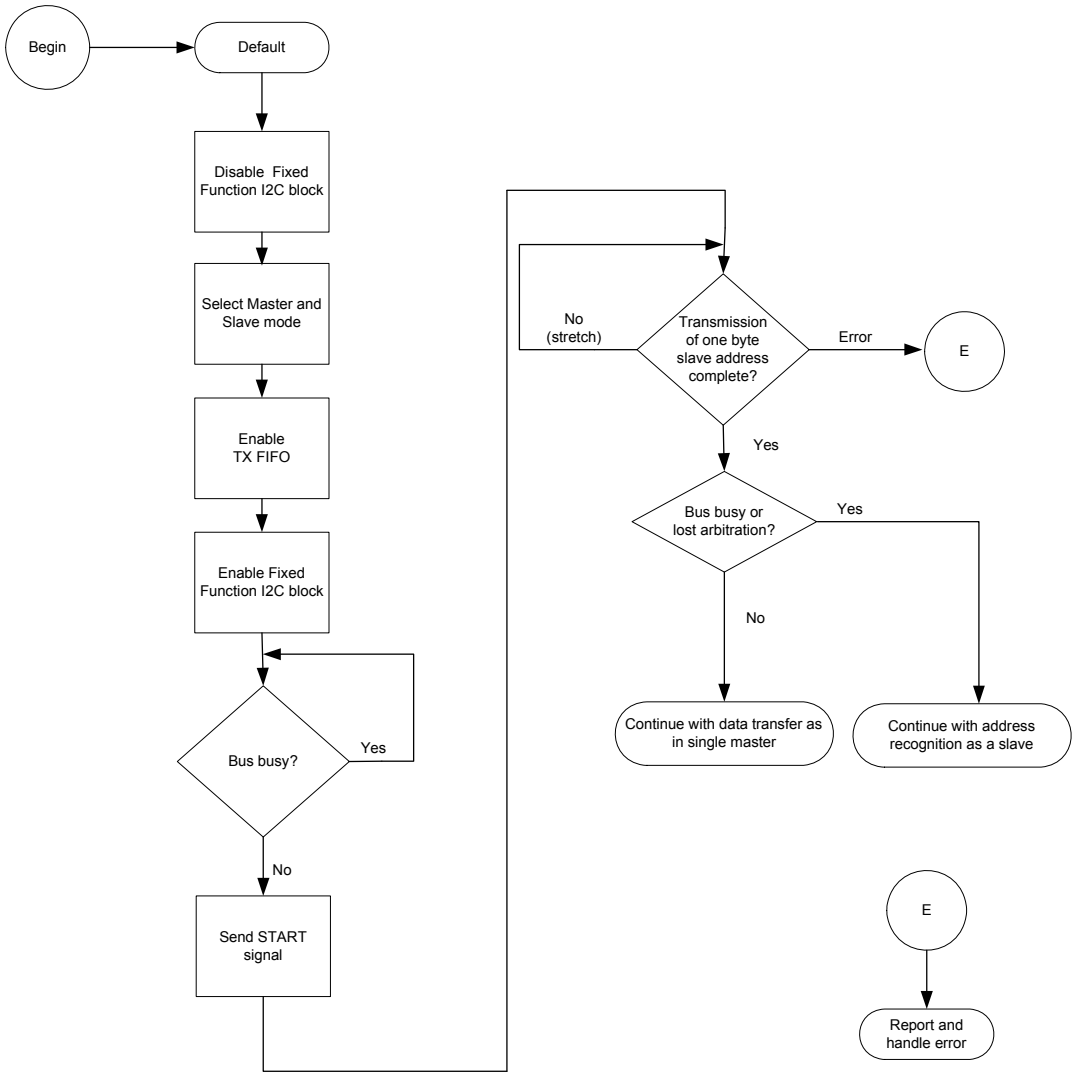
15.3.20.1　マルチマスター - スレーブが無効

図 15-32. 「マルチマスター、スレーブが無効」のフローチャート



15.3.20.2 マルチマスター - スレーブが有効

図 15-33. 「マルチマスター、スレーブが有効」のフローチャート



16. UDB

本章は PSoC® 4 のユニバーサル デジタル ブロック (UDB) について設計の詳細を説明します。新しい UDB アーキテクチャでは、コンフィギュレーションのきめ細かさと実装の効率との間に最適なバランスが成り立っています。UDB は、プログラマブル ロジック デバイス (PLD)、構造ロジック (データパス) および柔軟な配線機構の組み合わせで構成されています。

注 : PSoC 4100M ファミリには UDB がありません。

16.1 特長

- PSoC 4 には 4 個の UDB からなるアレイが含まれています。
- 柔軟性を最適化するために、それぞれの UDB には次のコンポーネントを含んでいます。
  - 複数のレジスタと FIFO、1 つの 8 ワード命令ストア レジスタを備える ALU ベースの 8 ビット データパス (DP)
  - 12 入力、8 本の積項 (プロダクトターム)、4 個のマクロセル 出力を備える 2 個の PLD
  - 制御とステータス モジュール
  - クロックとリセット モジュール
- UDB アレイを通して柔軟な配線が可能
- 大きいビット幅の機能に対応できる UDB の共有または連結
- タイマー、カウンター、PWM (デッド バンド ジェネレータ付き) 、UART、SPI、CRC 生成／ CRC チェックなど複数のデジタル機能を柔軟に実装可能
- レジスタ ベースの CPU インターフェース

図 16-1 には UDB のコンポーネント (2 個の PLD、1 個のデータパス、制御、ステータス、クロックとリセット機能) を示しています。図 16-2 には 4 個の UDB からなるアレイが PSoC 4 の他の部分にインターフェース接続される様子を示しています。

図 16-1. UDB ブロック図

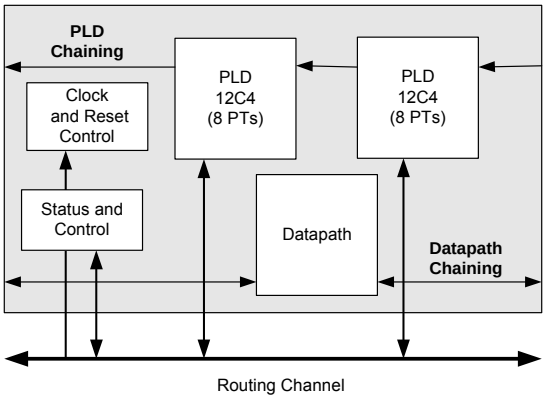
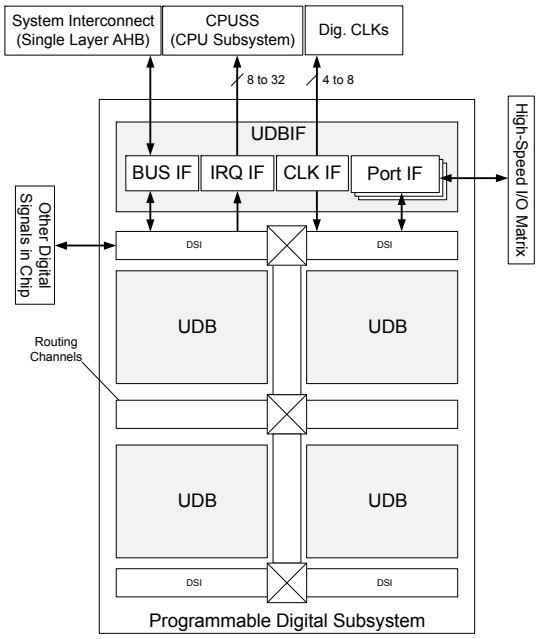


図 16-2. PSoC 4 UDB アレイ



16.2 動作原理

UDB の主なコンポーネントは次の通りです。

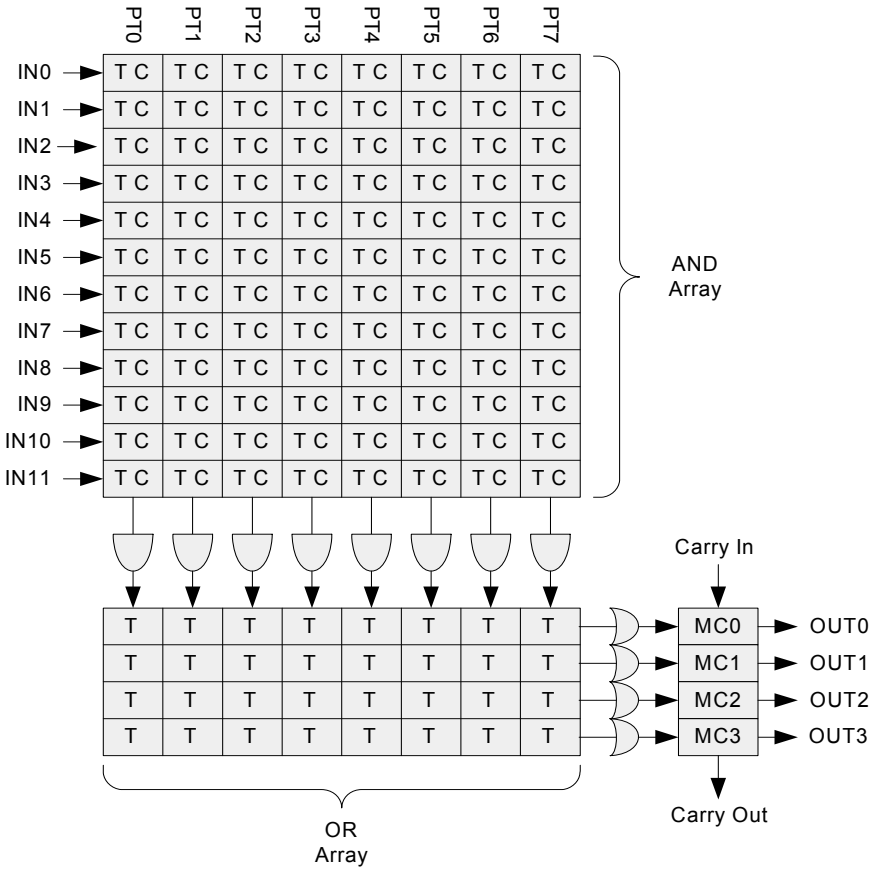
- **PLD (2個) :** 配線チャネルから入力を得て、レジスタ回路や組み合わせ積和回路を形成し、ステートマシン、データパス動作の制御、条件検出および出力の駆動を実現します。
- **データパス :** 動的にプログラム可能なALU、4個のレジスタ、2個の FIFO、コンパレーター、条件生成回路が含まれています。
- **制御とステータス :** CPU ファームウェアが、UDB とやり取りし同期する手段を提供します。
- **リセットとクロック制御 :** UDB 内の他のブロックにクロック選択とクロック イネーブル機能およびリセット選択機能を提供します。
- **信号連結 :** PLD とデータパスは信号連結機能を備えており、近隣の UDB を互いに連結してより高精度の機能を実現できます。
- **配線チャネル :** UDB はプログラム可能なスイッチ マトリクスを通して配線チャネルと接続し、UDB 内のブロック間接続あるいはアレイ内の他の UDB との接続を可能にします。
- **システム バス インターフェース :** 各UDB内蔵のレジスタと RAM はすべてシステム アドレス空間に統合され、8、16、32ビット アクセスでCPUからアクセス可能です。

16.2.1 PLD

UDB はそれぞれ 2 個の「12C4」PLD を内蔵しています。図 16-3 に示すように、PLD ブロックは、ステートマシンの実装、入出力データの条件検出、ルックアップ テーブル (LUT) の作成に使用できます。また PLD を算術的関数の実行、データパスの順序制御、ステータスの生成にも利用できます。汎用 RTL により論理合成と PLD ブロックへのマッピングができます。本節は PLD 設計の概要を説明します。

PLD には 12 の入力があり、これらは **AND (論理積)** アレイにある 8 本の積項 (PT) に供給されます。各積項で入力の真 (T) または補数 (C) を選択できます。積項の出力は OR アレイへの入力となります。「12C4」の「C」は和項 (OR ターム) があらゆる入力に一定であることを意味し、それぞれの OR アレイの入力は任意の PT にプログラムによりアクセスできます。この構造によって最大限の柔軟性が得られ、またすべての入力と出力が入れ替え可能になります。

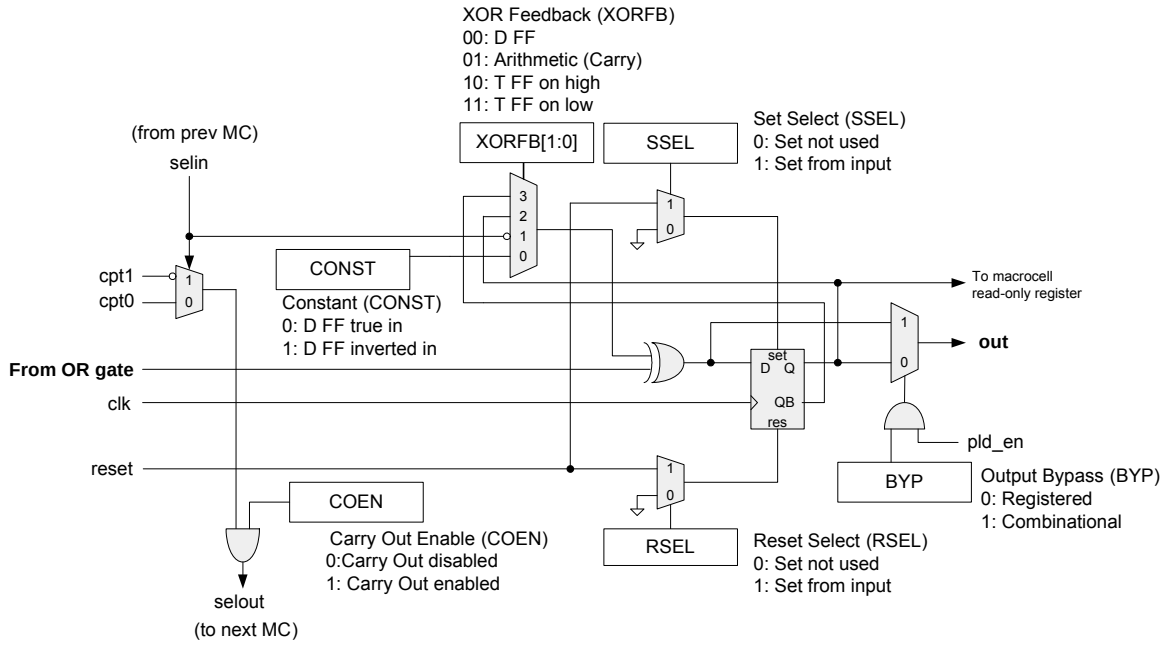
図 16-3. PLD 12C4 の構造



16.2.1.1 PLD マクロセル

図 16-4 にマクロセルのアーキテクチャを示します。出力は配線アレイを駆動し、レジスタ付きロジック回路か組み合わせロジック回路になることができます。レジスタモードは 4 つのモードがあり、、入力そのままか反転する D フリップ フロップ (DFF)、入力が HIGH か LOW の場合のトグル フリップ フロップ (TFF) です。出力レジスタは初期化のためにセット、リセットでき、また動作中に外部信号により非同期にセット、リセットできます。

図 16-4. PLD マクロセルのアーキテクチャ



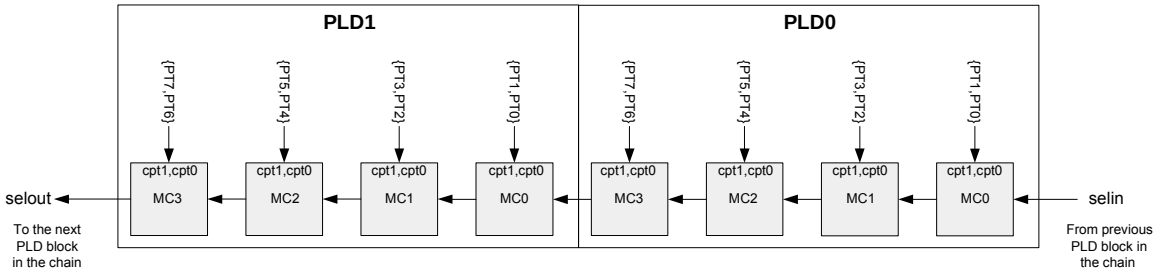
PLD マクロセル読み出し専用レジスタ

2 つの PLD にある 8 マクロセル出力は 8 ビットの読み出し専用レジスタとして CPU からアクセスすることができます。2 つ以上の UDB にまたがるマクロセルは 16 ビットか 32 ビットの読み出し専用レジスタとしてアクセスできます。170 ページの UDB アドレス指定を参照してください。

16.2.1.2 PLD キャリー連結

PLD は UDB のアドレス順に連結されます。図 16-5 に示すように、キャリー 連結入力「selin」は、連結体の 1 つ前の UDB から得られ、両方の PLD 内のそれぞれのマクロセルを通して、キャリー連結出力「selout」として次の UDB に配信されます。算術的関数のマッピング処理がより効果的になるために、特別な積項が生成され、キャリー連結と共にマクロセル内に使用されます。

図 16-5. PLD キャリー連結と特別な積項の入力



16.2.1.3 PLD コンフィギュレーション

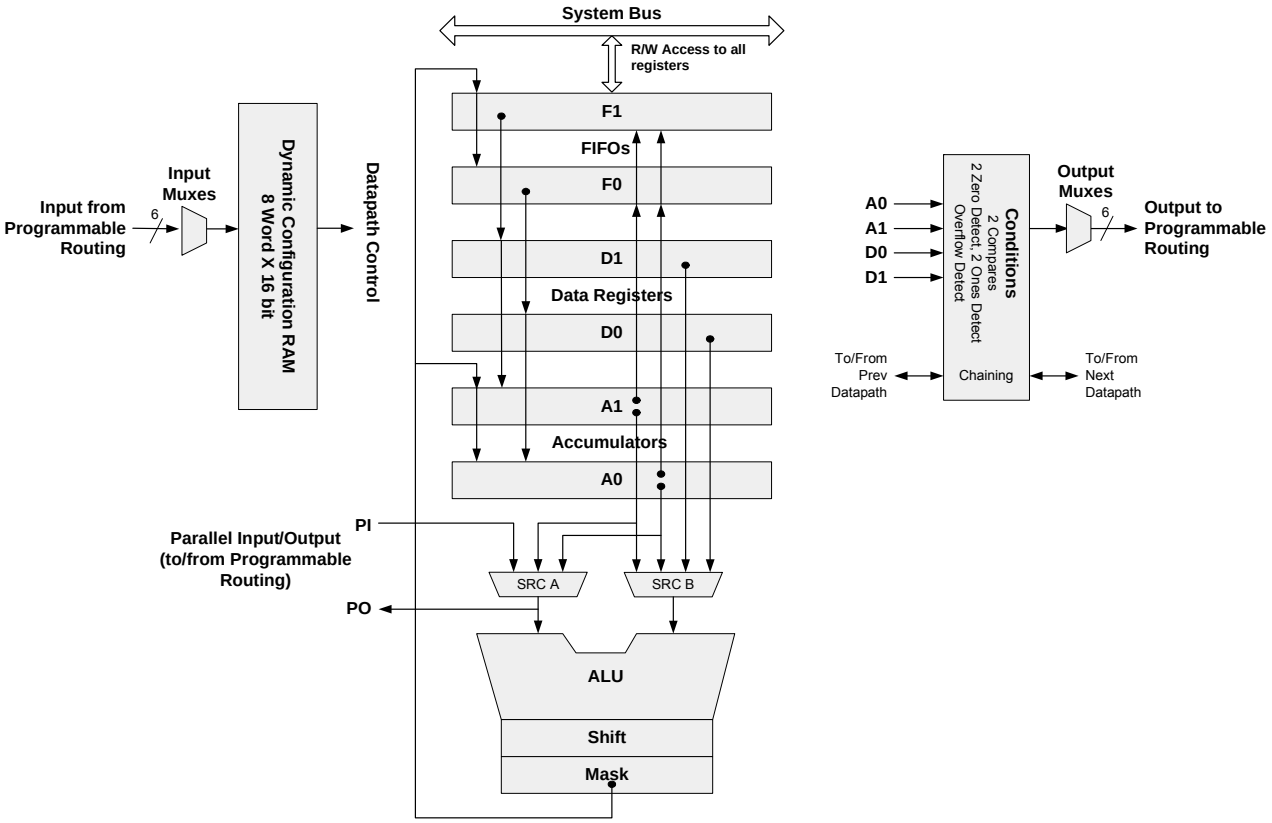
16 ビットか 32 ビットレジスタ セットにアクセスすることで PLD を構成できます。170 ページの UDB アドレス指定をご覧ください。

16.2.2 データパス

図 16-6 に示すように、データパスには 8 ビット シングル サイクル ALU および関連する比較回路と条件生成回路が含まれています。データパスは隣接する UDB の他のデータパスと連結することで、より高精度機能が実行可能になります。データパスには動的コンフィギュレーション RAM を内蔵しています。これにより特定のサイクルで動作を動的に選択できます。

データパスは、タイマー、カウンター、PWM、PRS、CRC、シフタ、デッドバンド ジェネレータなど一般的な組み込み機能を実現するために最適化されています。加算と減算機能により、デジタルデルタシグマ動作が実行可能になります。

図 16-6. データパス　トップレベル図



16.2.2.1　概要

これからデータパスの主な特長について説明します。

動的コンフィギュレーション

動的コンフィギュレーションとは、シーケンサの制御下で、サイクルごとにデータパスの機能およびその相互接続を変更する機能のことです。これはコンフィギュレーション RAM を使用して実現され、この RAM に他と重複しない 8 つのコンフィギュレーションが保存されています。この RAM へのアドレス入力は、配線構造に接続した任意のブロックから得ることができます。ブロックの一般的なものは PLD ロジック、I/O ピンや他のデータパスです。

ALU

ALU は、インクリメント、デクリメント、加算、減算、AND、OR、XOR および PASS の 8 種の機能を実行できます。機能の選択は、コンフィギュレーション RAM によってサイクルごとに制御されます。独立したシフト (左シフト、右シフト、ニブル スワップ) およびマスク演算は ALU の出力で可能です。

条件検出

データパスはそれぞれ 2 個のビット マスク処理オプション付きコンパレータを備え、比較のためにデータパス レジスタ入力の選択を様々に設定できます。その他の検出可能な条件として、すべて 0、すべて 1 およびオーバフローがあります。この条件検出出力がデータパス出力選択を形成します。デジタル配線構造に接続されて他の機能の入力とされます。

内蔵 CRC、PRS

データパスには、シングル サイクル巡回冗長検査 (CRC) の計算および任意の段数、生成多項式の疑似ランダム シーケンス (PRS) の生成に対応する機能が組み込まれています。8 ビット以上の段数を持つ CRC、PRS を得るために、データパスが連結されることがあります。この特長は動的に制御されるので、他の機能とインターリーブされることがあります。

可変 MSB

算術関数およびシフト関数の最上位ビット (MSB) は、プログラムによる指定が可能です。これにより、可変幅の CRC、PRS 機能をサポートし、ALU の出力マスキングと組み合わせて、任意の幅のタイマー、カウンターおよびシフト ブロックを実現することができます。

入出力 FIFO

各データパスには 4 バイトの FIFO が 2 つ含まれます。これらは入力バッファ (CPU が FIFO に書き込み、データパスが FIFO を読み出す) または出力バッファ (データパスが FIFO にの書き込み、CPU が FIFO からの読み出す) として個別に

設定できます。これらの FIFO は FULL または EMPTY のステータス信号を生成し、シーケンサーや割り込みと連動するために利用できます。

連結

データパスは近隣のデータパスと、条件検出出力と信号を連結するように設定できます。シフト、キャリー、キャプチャおよび他の条件検出信号を連結して、より高精度の算術、シフト、CRC、PRS 機能を形成することができます。

時分割多重化

オーバーサンプリングされたアプリケーションまたは高いクロック速度を必要としないアプリケーションでは、データパス内の単一 ALU ブロックを、2 組のレジスタと条件発生器で効率的に共用することができます。ALU とシフトの出力はレジスタに記憶され、後続のサイクルで入力として使用することができます。使用例としては、1 個の (8 ビット) データパスで 16 ビット機能をサポートすること、または CRC 生成動作をデータ シフト動作とインターリーブすることです。

データパス入力

データパスは 3 種類の入力があります : コンフィギュレーション、制御およびシリアルとパラレル データコンフィギュレーション入力は動的コンフィギュレーション RAM アドレスを選択します。制御入力は FIFO からデータ レジスタをロードし、**accumulator (アキュムレータ)** 出力を FIFO に取り込みます。シリアル データ入力にはシフト インとキャリー インがあります。1 個のパラレル データ入力ポートで最大 8 ビットのデータが配線から得られます。

データパス 出力

データパスから合計 16 個の信号が生成されます。これらの信号は条件検出信号 (例えばコンペア信号)、ステータス信号 (例えば FIFO ステータス信号)、データ信号 (例えばシフト アウト信号) からなります。これらの 16 信号は 6 個のデータパス出力に多重化されて配線マトリックスを駆動します。デフォルト状態では、出力はシングル同期化 (パイプライン化) されます。これらの出力に対して組み合わせ出力オプションは依然として有効です。

データパス ワークレジスタ

各データパス モジュールは 6 個の 8 ビット ワークレジスタを含んでいます。すべてのレジスタは CPU による読み出しと書き込みが可能です。

表 16-1. データパス ワークレジスタ

種類	名称	説明
アキュムレータ	A0、A1	アキュムレータは ALU のソースとデスティネーションになることができる。データ レジスタか FIFO からロードされる場合もある。アキュムレータはカウント、CRC、シフトなどの機能の現行値を通常含む。これらのレジスタはデータが保持されないレジスタで、スリープ モードではその値を失い、ウェイクアップ時に「0x00」の値にリセットされる
データ	D0、D1	データ レジスタは PWM 比較値、タイマー周期、CRC 多項式などの機能の定数値を通常含む。これらのレジスタはスリープ期間中にも保持される
FIFO	F0、F1	2 本の 4 バイト FIFO は、バッファされたデータのソースとデスティネーションになることができる。これらの FIFO は、両方とも入力または出力バッファ、あるいはそれぞれ入力バッファと出力バッファになるように構成される。ステータス信号は FIFO レジスタの満杯 (FULL) ／空 (EMPTY) の状態を示す。使用例として、SPI や UART での送信と受信データのバッファ、PWM 比較データのバッファ、タイマー周期データのバッファがある。これらのレジスタはデータが保持されないレジスタで、スリープ モードではその値を失い、ウェイクアップ時に「0x00」の値にリセットされる

16.2.2.2 データパス FIFO

FIFO のモードと構成

各 FIFO は様々な動作モードと構成があります。

表 16-2. FIFO のモードと構成

モード	説明
入力／出力	入力モードでは、CPU はデータを FIFO に書き込んで、データパス ブロックは内部でデータを読み出して処理する。出力モードでは、データパスは内部でデータを FIFO に書き込んで、CPU はデータ読み出しと計算処理を行う
シングル バッファ	FIFO はステータスなしの 1 バイト バッファとして動作する。FIFO に書き込まれたデータは直ちに読み出し可能になり、即時に上書き可能になる
レベル／エッジ	レベル トリガーかエッジ トリガーによりデータパスから FIFO へのロードを制御する

表 16-2. FIFO のモードと構成 (続き)

モード	説明
通常／高速	データパス ソースから FIFO にロードする制御が、選択されているデータパス クロック (通常) か HFCLK (ファスト) でサンプリングされる。これによりデータ取り込みを、データパス クロックから独立して、システムの最高の周波数速度 (HFCLK) で行うことができる
ソフトウェア 取り込み	このモードが有効で FIFO が出力モードの場合、CPU が関連するアキュムレータ (F0 には A0、F1 には A1) からデータを読み出すと、アキュムレータ内の値が同時に FIFO に送信される。取り込まれたデータはその後 FIFO から即時に読み出すことができる。連結機能が有効であれば、動作は連結された結合体を最上位ブロックまで移動し、データパスがマルチバイトの値をアトミックな読み出しとして行う
非同期	データパスが HFCLK に対して非同期にクロック供給される場合、FIFO のステータス信号は、直接 (データパス クロックへの 1 回サンプル) または非同期データパス クロックへの 2 回サンプルのいずれかの方法で、データパスの残りの部分まで送信される
独立したクロック極性	FIFO はデータパス クロックに対応した FIFO クロックの極性を反転する制御ビットをそれぞれ 1 つずつ持つ

図 16-7 は入出力モードで制御される FIFO 構成を示しています。TX ／ RX モードでは、1 つの FIFO は入力モードで、残りの FIFO は出力モードです。この構成の代表例は SPI です。デュアル キャプチャ構成では、A0 と A1 を独立に取り込みあるいは A0 と A1 のいずれか 1 つに対して 2 個個別に制御される取り込み方式を提供します。最後に、デュアル バッファ モードはバッファされた周期と比較レジスタまたは 2 個の独立した周期か比較レジスタを提供します。

図 16-7. FIFO の構成

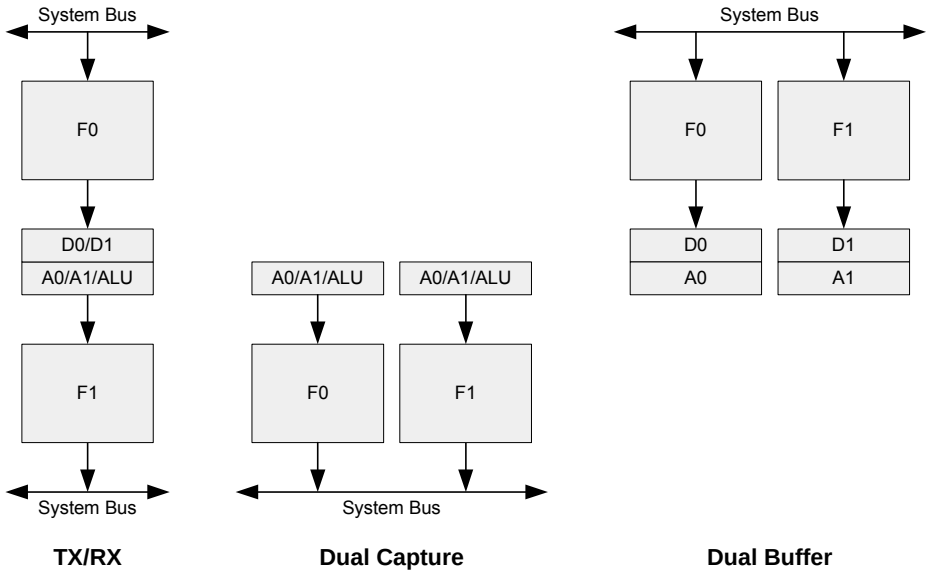
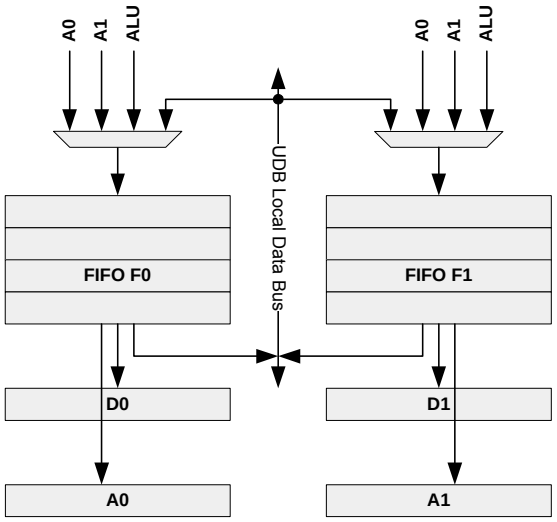


図 16-8 に FIFO のソースとシンクを詳しく示します。

図 16-8. FIFO のソースとシンク



FIFO が入力モードの場合、そのソースはシステム バスで、シンクは Dx と Ax レジスタです。FIFO が出力モードの場合、そのソースは Ax レジスタと ALU で、シンクはシステム バスです。F0_INSEL[1:0] や F1_INSEL[1:0] に対して設定する表 16-3 のとおりに、マルチプレクサ選択は UDB コンフィギュレーション レジスタ CFG15 にてスタティックに設定されます。

表 16-3. UDB CFG15 レジスタによる FIFO マルチプレクサ設定

Fx_INSEL[1:0]	説明
00	**入力モード** - システム バスからデータが FIFO に書き込まれ、FIFO の出力デスティネーションは Ax か Dx
01	**A0 出力モード** - FIFO の入力ソースは A0 で、FIFO の出力デスティネーションはシステム バス
10	**A1 出力モード** - FIFO の入力ソースは A1 で、FIFO の出力デスティネーションはシステム バス
11	**ALU 出力モード** - FIFO の入力ソースは ALU の出力で、FIFO の出力デスティネーションはシステム バス

FIFO ステータス

各 FIFO は「バス」と「ブロック」という 2 つのステータス信号を生成します。これらの信号はデータパス出力マルチプレクサを通して UDB 配線に送信されます。「バス」ステータス信号は、FIFO 読み出し、書き込みの割り込み要求をアサートするために使用されます。「ブロック」ステータス信号は主に FIFO ステータスを UDB 内部に提供するために使用されます。ステータス ビットの意味は、FIFO の設定された方向 (UDB CFG15 レジスタの Fx_INSEL[1:0] ビット) と FIFO レベル ビットに依存します。FIFO レベル ビット (Fx_LVL) はワークレジスタ空間の補助制御ワークレジスタ (ACTL) にて設定されます。表 16-4 はそのステータス オプションを示しています。

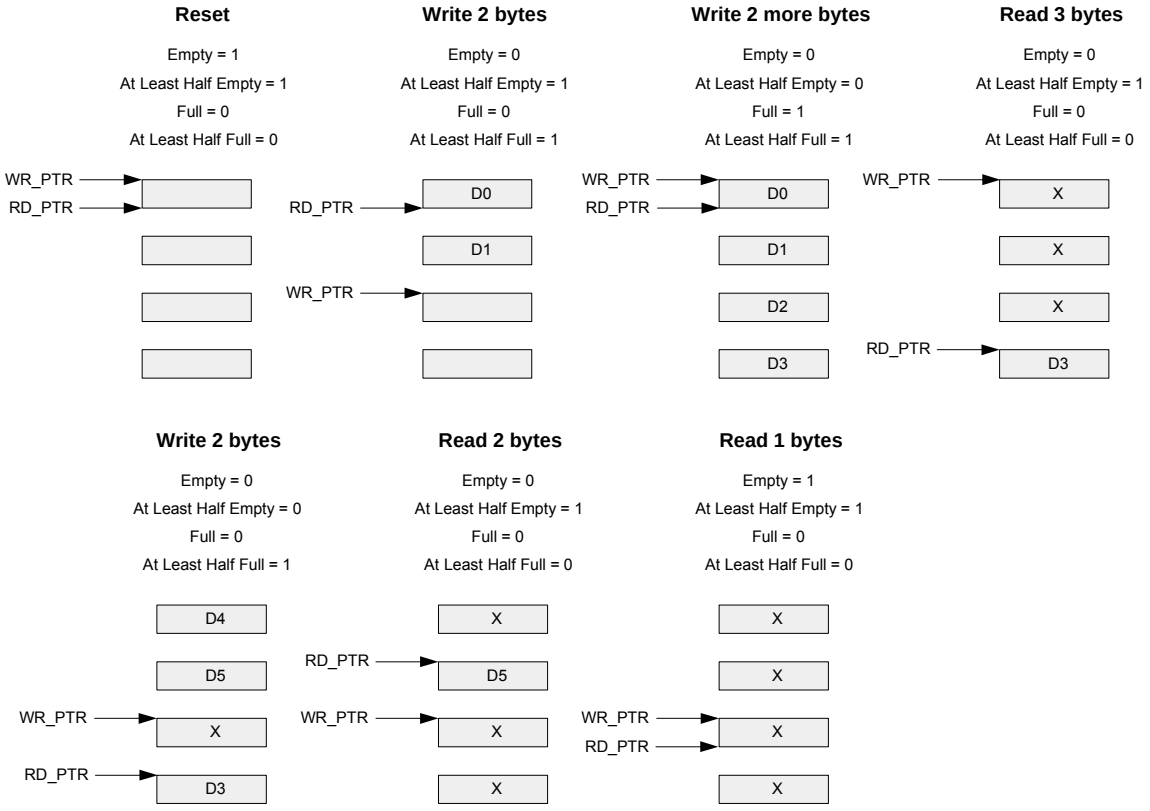
表 16-4. FIFO ステータス

Fx_INSEL [1:0]	Fx_LVL	FIFO ステータス	FIFO ステータス信号	説明
入力	0	満杯 (FULL) でない	バスステータス	FIFO に最低限 1 バイトの空間がある場合にアサート
入力	1	最低限半分が空 (EMPTY)	バスステータス	FIFO に最低限 2 バイトの空間がある場合にアサート
入力	該当なし	空 (EMPTY)	ブロックステータス	FIFO に使用可能なバイトがない時にアサート。空でない時に、データパスは内部でバイトを使用可能。空の場合、データパスはアイドル状態になるかアンダーラン状態を生成
出力	0	空 (EMPTY) でない	バスステータス	FIFO から読み出し可能なビットが最低限 1 バイトある場合にアサート
出力	1	最低限半分が空 (EMPTY)	バスステータス	FIFO から読み出し可能なビットが最低限 2 バイトある場合にアサート
出力	該当なし	満杯 (FULL)	ブロックステータス	FIFOが満杯の場合にアサート満杯でない時に、データパスは内部でデータ バイトを FIFO に書き込み可能満杯の場合、データパスはアイドル状態になるかオーバーラン状態を生成

FIFO 動作

図 16-9 は基本的な読み出し、書き込みシーケンスおよび関連するステータス生成を示しています。この図では、読み出しと書き込みが異なる時点で発生したと示されていますが、読み出しと書き込みが同時に発生しても構いません。

図 16-9. シンク側 FIFO の詳細動作

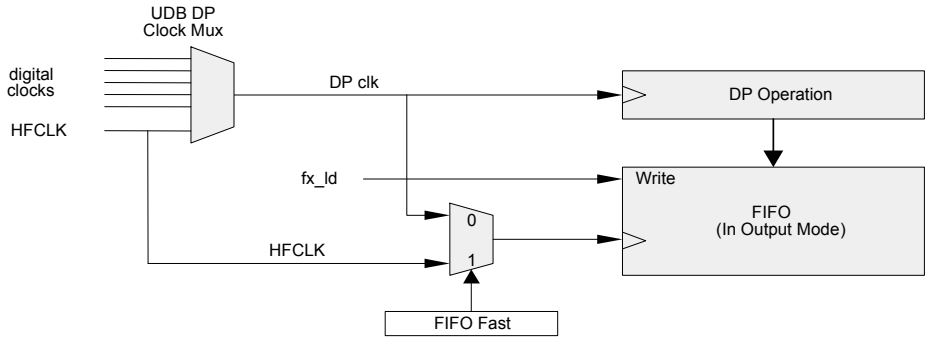


FIFO ファスト モード (FIFO FAST)

FIFO が出力に構成されると、FIFO ロード動作は通常、当時に選択されたデータパス クロックを使用して書き込み信号のサンプル処理を行います。図 16-10 に示すように、FIFO ファスト モードが有効にされる状態で、HFCLK はこの動作のためにオプションとして選択されることがあります。エッジ センシティブ モードと共に使用されると、この動作はアキュムレータから ＦＩＦＯ までの転送の待ち時間を、データパス クロックの遥かに高い分解能から HFCLK の分解能に削減することができます。これにより、CPU は最小限の待ち時間で FIFO 内に取り込まれた結果を読み込んでいます。

図 16-10 は、ファスト ロード動作が現時点に選択されているデータパス クロックに独立しますが、HFCLK を使用すると消費電力が高くなることを示しています。着信信号 fx_ld は HFCLK のタイミングに適合する必要があるため、再びローカル同期が必要になることに注意してください。

図 16-10. FIFO ファスト コンフィギュレーション シンク



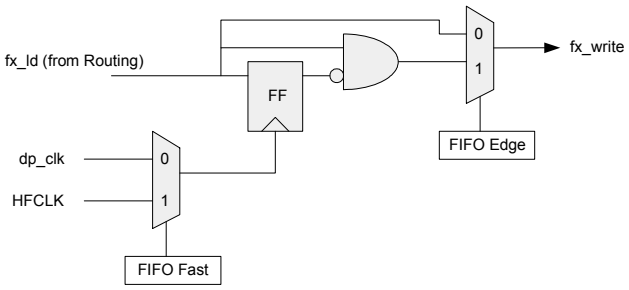
FIFO レベル／エッジ書き込みモード

データパスから FIFO へのデータ書き込みモードは 2 つ使用可能です。第 1 モードでは、データがアキュムレータから FIFO に同期転送されます。書き込み動作の制御信号 (fx_ld) は基本的にステートマシンから生成されるか、またはデータパス クロックと同期検出信号により生成されます。FIFO への書き込みは入力ロード制御信号の値が「1」のサイクルで実行されます。

第 2 モードでは、FIFO は fx_ld 信号の立ち上がりエッジに応じてアキュムレータの値を取り込むために使用されます。このモードでは波形のデューティ サイクルが任意です (ただし、その幅は少なくとも 1 データパス クロック サイクルでなければなりません)。このモードの例としては、外部ピンの入力をトリガーとして使用してアキュムレータの値を取り込むことです。このモードの制限は、他の立ち上がりエッジが検出される 1 サイクル前に入力制御を「0」に反転する必要があることです。

図 16-11 は fx_ld 制御の入力上のエッジ検出オプションを示しています。このオプションでは、1 ビットが UDB 内の両方の FIFO のモードを設定します。エッジ検出は選択した FIFO クロックの速度でサンプリングされます。

図 16-11. 内部 FIFO 書き込み用のエッジ検出オプション



FIFO ソフトウェア取り込みモード

動作中にアキュムレータの内容を確実に読み出す能力を CPU が持つことは重要な要件です。このことはソフトウェア取り込みにより実現され、FIFO キャップ コンフィギュレーション ビット (UDB CFG16 レジスタ内の FIFO_CAP ビット) を設定することで有効になります。このビットは UDB の両方の FIFO に適用しますが、FIFO が出力モードである場合のみ動作できます。ソフトウェア取り込みを使用する場合、F0 と F1 がそれぞれ A0 と A1 からデータをロードするように設定する必要があります。

図 16-12 に示すように、アキュムレータの読み出しは、そのアキュムレータから FIFO への書き込みをトリガーします。この信号は連結されているため、任意バイトを読み出すと、すべての連結されている UDB のアキュムレータが同時に取り込まれます。これにより、CPU は同時に 16 ビットかそれ以上のビットを確実に読み込めます。アキュムレータ読み出しから返されてきたデータを無視する必要があります; 取り込まれたデータは直ちに FIFO から読み出すことができます。

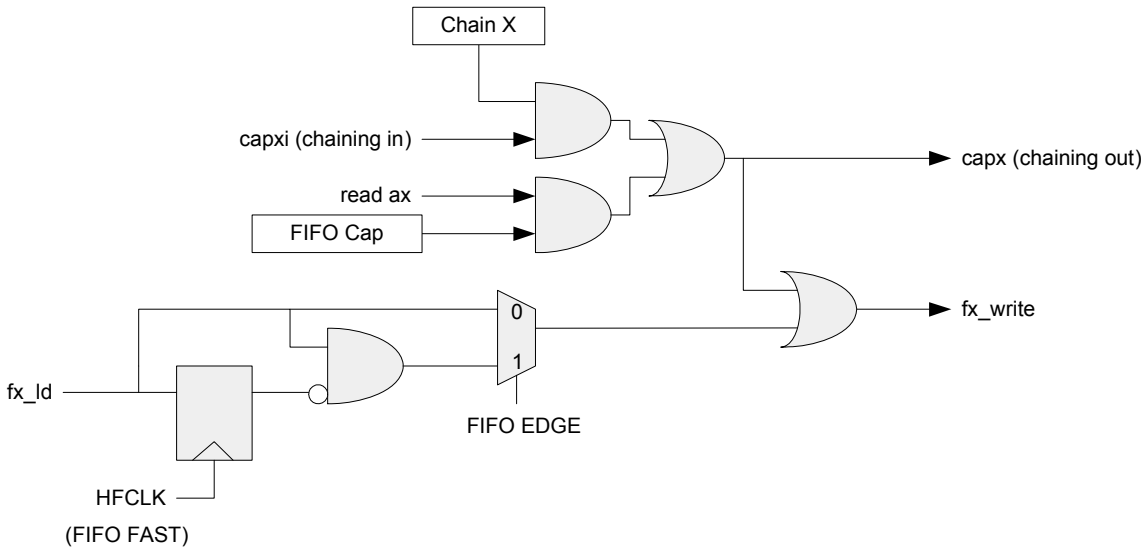
FIFO ロードを生成する fx_ld 信号はソフトウェア取り込み信号と OR で結合されます; ハードウェア取り込みとソフトウェア取り込みが同時に適用されるとその結果は予測つかなくなる可能性があります。原則として、これらの機能は互いに排他的機能です ; ただし、以下の設定で、ハードウェアとソフトウェア取り込みを同時に使用できます。

- FIFO 取り込みクロック供給モードは FIFO FAST に設定
- FIFO 書き込みモードは FIFO EDGE に設定

この設定によって、ハードウェアとソフトウェア取り込みの動作は基本的に同様で、特定の HFCLK サイクルで、これらの信号のいずれか 1 つがアサートされると取り込みが開始します。

ソフトウェア取り込みを開始する前にファームウェア (UDB ACTL レジスタ) により目標の FIFO をクリアすることも推奨します。これは FIFO 読み出しと書き込みポインターを既知の状態に初期化します。

図 16-12. ソフトウェア取り込みのコンフィギュレーション



FIFO 制御ビット

補助制御レジスタ (ACTL) には、CPU ファームウェアが通常動作中に FIFO を制御するために使用可能な 4 ビットがあります。
FIFO0 CLR と FIFO1 CLR ビットは FIFO をリセット又はフラッシュするために使用されます。これらのビットのいずれか 1 ビットに「1」の値を書き込むと、対応する FIFO がリセットされます。FIFO 動作が継続するためにはこのビットを再度「0」に書き込む必要があります。このビットがアサートされたままになると、選択した FIFO は無効になり、状態なし 1 バイト バッファーとして動作します。データを FIFO に書き込めますが、そのデータは直ちに読み出し可能になり、随時に上書きされる可能性があります。Fx INSEL[1:0] (UDB CFG15 レジスタ) コンフィギュレーション ビットを用いて設定したデータ方向は依然として有効です。
4 バイト FIFO はバスステータスをアサートします。FIFO0 LVL と FIFO1 LVL ビットは FIFO がアサートするレベルを設定します。表 16-5 に示す通り、FIFO バスステータスの意味は入出力により異なります。

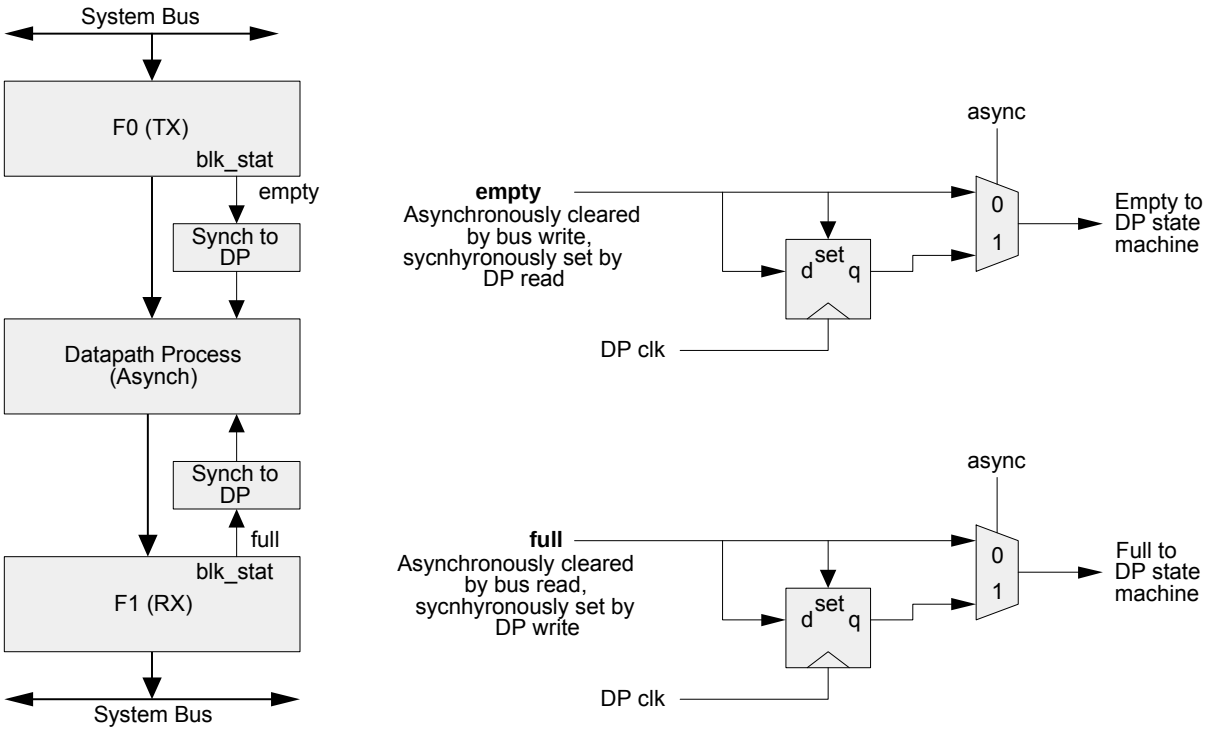
表 16-5. UDB ACTL レジスタによる FIFO レベル制御ビット

FIFOx LVL	入力モード (バスから FIFO にデータ書き込み)	出力モード (FIFO からバスにデータ読み出し)
0	満杯 (FULL) でない 書き込み可能なビットが最低限 1 バイトある	空 (EMPTY) でない 読み出し可能なビットが最低限 1 バイトある
1	最低限半分が空 (EMPTY) 書き込み可能なビットが最低限 2 バイトある	最低限半分が満杯 (FULL) 読み出し可能なビットが最低限 2 バイトある

FIFO の非同期動作

図 16-13 は FIFO の非同期動作の概念を示します。1 つの例として、F0 を入力モード、F1 を出力モードに設定するとします。これはTXとRXレジスタの一般的なコンフィギュレーションです。
TX 側では、データパスのステートマシンは「EMPTY」を使って、使用可能なバイトがあるかを決定します。「EMPTY」は DP のステートマシンと同期してセットされますが、バスからの書き込みに応じて非同期にクリアされます。クリアの際には、その状態は再び DP のステートマシンと同期化されます。
RX 側では、データパスのステートマシン「FULL」を使って、FIFO に書き込み可能な空間が残っているかを決定します。「FULL」は DP のステートマシンと同期してセットされますが、バスからの読み出しに応じて非同期にクリアされます。クリアの際には、その状態は再び DP のステートマシンと同期化されます。
この方式を有効にするためには UDB CFG16 レジスタの 1 つの FIFO ASYNCH ビットを使用します。このビットがセットされると両方の FIFO に適応します。バスステータスは必然的に割り込みプロセスにより同期化されることを前提としたため、このビットはブロックステータスにのみ適応されます。

図 16-13. FIFO の非同期動作



FIFO のオーバーフロー動作

内部 (データパス) と外部 (CPU) 読み出し／書き込みの両方を確実に実装するためには FIFO 状態通知機能を利用します。アンダーフローとオーバーフロー状態に対する内蔵保護はありません。FIFO が満杯で後続の書き込みが発生する (オーバーフロー) と、新しいデータは FIFO の頭部 (当時に出力されているデータ、または次に読み出されるデータ) を上書きします。FIFO が空の状態で後続の読み出しが発生する (アンダーフロー) と、読み出し値は未定義値です。FIFO ポインタはアンダーフローやオーバーフロー状態にも関わらず依然として正確です。

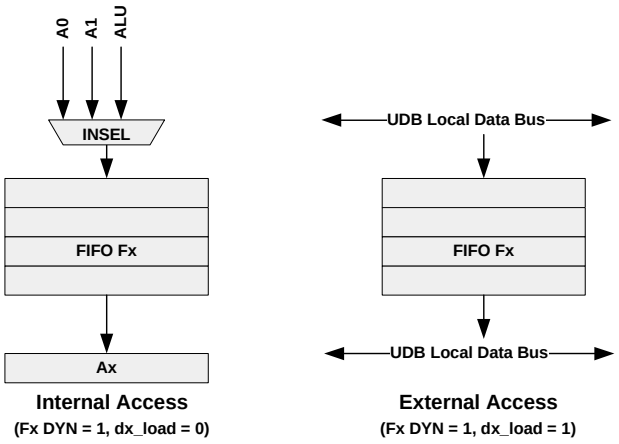
FIFO クロック反転オプション

各 FIFO は UDB CFG16 レジスタ内の Fx CK INV といった 1 個の制御ビットがあります。このビットは DP クロックの極性に対して FIFO クロックの極性を制御します。デフォルト状態で、FIFO は DP クロックと同様の極性で動作します。このビットがセットされると、FIFO は DP クロックと対立して反対の極性で動作します。これは SPI のような「両クロック エッジ」通信プロトコルにサポートします。

FIFO の動的制御

通常、FIFO は入力か出力モードになるようにスタティックに設定されます。代替の方法として、各 FIFO は特定のモードに設定されます。このモードでは FIFO の方向は配線信号により動的により制御されます。1 個の FIFO にあたり 1 つあるコンフィギュレーション ビット (UDB CFG17 レジスタの Fx DYN ビット) はモードを有効にします。図 16-14 は動的な FIFO モードで可能なコンフィギュレーションを示しています。

図 16-14. FIFO の動的モード



内部アクセス モードで、データパスは FIFO との読み出しと書き込みを実行できます。このコンフィギュレーションで、FIFO への書き込みソースを選択するために Fx INSEL ビットを設定する必要があります。Fx INSEL = 00 (CPU バス ソース) はこのモードで無効です ; そのビットのあり得る値は 01、10、または 11 (A0、A1、または ALU) だけです。単一の可能な読み出しアクセスは関連するアキュムレータへの読み出しです ; データ レジスタのデスティネーションはこのモードで不可であることに注意してください。

外部アクセス モードで、CPU は FIFO との読み出しと書き込みを実行できます。内部と外部アクセスのコンフィギュレーションはデータパスの配線信号により動的に切り替え可能です。データパス入力信号 d0_load と d1_load はこれの制御に使用されます。動的制御モードでは、d0_load と d1_load は、D0/D1 レジスタの値を F0/F1 からロードする通常の用途に使用できないことに注意してください。dx_load 信号は定数を含む任意の配線信号で駆動されます。

使用例の 1 つとしては、外部アクセス (dx_load == 1) から開始して、CPU は 1 バイトかそれ以上のデータ バイトを FIFO に書き込みます。次に内部アクセス (dx_load == 0) に切り替えて、データパスはデータに対して動作できます。また外部アクセスに切り替えると、CPU は計算結果を読み出すことができます。

Fx INSEL は常に 01、10、または 11 (A0、A1、または ALU) に設定される必要がある、要するに通常動作の「出力モード」であるため、FIFO ステータス信号の定義は次の通りになります (Fx LVL 制御にも依存します) 。

表 16-6. FIFO の状態

ステータス信号	意味	Fx LVL = 0	Fx LVL = 1
fx_blk_stat	書き込み状態	FIFO が満杯 (FULL)	FIFO が満杯 (FULL)
fx_bus_stat	読み出し状態	FIFO が空 (EMPTY) でない	最低限半分が満杯 (FULL)

データパスと CPU は両方とも FIFO に対して書き込みと読み出しを実行可能のため、これらの信号の意味は「ブロック」とバスステータスではなくなります。blk_stat 信号は書き込み状態、bus_stat 信号は読み出し状態を示すために使用されます。

16.2.2.3 FIFO の状態

FIFO のステータス信号は 4 つあり、それぞれの FIFO が 2 個の信号を持っています : fifo0_bus_stat、fifo0_blk_stat、fifo1_bus_stat、fifo1_blk_stat。これらの信号の意味は、指定した FIFO の方向 (スタティック コンフィギュレーションにより決定される方向) に依存します。

16.2.2.4 データパス ALU

ALU コアには 3 つの独立した 8 ビット プログラム可能な機能からなります。それらは、算術／論理ユニット、シフター ユニットおよびマスク ユニットです。詳細には UDB データパス アーキテクチャのブロック図 (図 16-6) をご覧ください。

算術論理演算

ALU 機能はコンフィギュレーション RAM により動的に設定され、表 16-7 に示されています。

表 16-7. UDB DCFG レジスタによる ALU 機能

Func[2:0]	機能	演算
000	PASS	srca
001	INC	++srca
010	DEC	--srca
011	ADD	srca +srcb
100	SUB	srca – srcb
101	XOR	srca ^ srcb
110	AND	srca and srcb
111	OR	srca | srcb

srca = ALU の入力ソースは「A」で、srcb = ALU の入力ソースは「B」です。図 16-6 を参照してください。

キャリーイン

キャリーインは算術演算に使用されます。表 16-8 は特定の機能に対するデフォルトのキャリーイン値を示しています。

表 16-8. キャリーイン機能

機能	演算	デフォルトのキャリーイン値
INC	++srca	srca + 00h + ci (ci を 1 に指定する)
DEC	--srca	srca + ffh + ci (ci を 0 に指定する)
ADD	srca + srcb	srca + srcb + ci (ci を 0 に指定する)
SUB	srca – srcb	srca + ~srcb + ci (ci を 1 に指定する)

キャリー動作用のこれらのデフォルト算術モードに加え、追加オプションとして 3 つのキャリーオプションがあります。CFG13 レジスタ内の CI SELA と CI SELB コンフィギュレーション ビットは特定のサイクルに対してキャリーインを決めます。動的コンフィギュレーション RAM はサイクルごとに A か B コンフィギュレーションのどれかを選択します。それらのオプションは表 16-9 に定義されています。

表 16-9. UDB CFG13 レジスタによる追加キャリーイン機能

CI SEL A CI SEL B	キャリーモード	説明
00	デフォルト	表 16-8 に示しているデフォルトの算術モード
01	レジスタ出力	キャリー フラグ、1 つ前のサイクルからのキャリー結果。このモードは「キャリー付き演算」と「ボロー付き減算」の実行に使用される。倍精度演算をエミュレートするために連続サイクルで使用可能
10	配線	キャリーは別の場所で生成され、この入力に配線される。このモードは制御可能カウンターに使用可能
11	連結	キャリーは前のデータパスに連結される。このモードは 2 つ以上のデータパスからなるより高精度のシングル サイクル演算を行うために使用可能

配線キャリーを適用する場合、それぞれの算術的関数に関する意味は表 16-10 に示しています。デクリメントと減算関数の場合に、キャリーはアクティブ LOW (反転) です。

表 16-10. 「配線」モードのキャリーイン機能

機能	キャリーイン極性	キャリーイン有効	キャリーイン無効
INC	真	++srca	srca
DEC	反転	--srca	srca
ADD	真	(srca + srcb) + 1	srca + srcb
SUB	反転	(srca – srcb) – 1	(srca – srcb)

キャリーアウト

キャリーアウトは選択可能なデータパス出力で、当時定義された MSB の位置 (スタティックにプログラム可能な位置) から派生されます。また、この値はオプションのキャリーイン値として次の最上位ブロックにも連結されます。デクリメントと減算関数の場合に、キャリーアウトは反転されます。

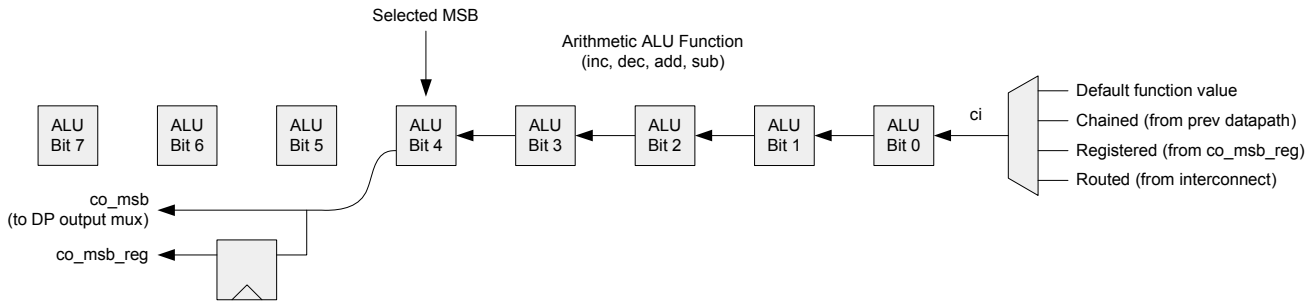
表 16-11. キャリーアウト機能

機能	キャリーアウト極性	キャリーアウト有効	キャリーアウト無効
INC	真	++srca == 0	srca
DEC	反転	--srca == –1	srca
ADD	真	srca + srcb > 255	srca + srcb
SUB	反転	srca – srcb < 0	(srca – srcb)

キャリーの構造

図 16-15 はキャリーインのオプションおよびキャリーアウト生成用の MSB 選択を示しています。レジスタされたキャリーアウト値は後続の算術演算のキャリーイン値として選択されることができます。この特長は、多くのサイクルでより高精度の機能を実装するために使用されます。

図 16-15. キャリー動作



シフト動作

シフト動作は ALU 動作と独立して表 16-12 に従って発生します。

表 16-12. UDB DCFG レジスタによるシフト動作機能

Shift[1:0]	機能
00	通過 (PASS)
01	左シフト
10	右シフト
11	ニブル スワップ

シフトアウト値をデータパス出力として使用可能です。右シフトアウト (sor) と左シフトアウト (sol_msb) の両方はその出力選択を共有します。スタティック コンフィギュレーション ビット (UDB CFG15 レジスタの SHIFT SEL ビット) はデータパス出力として使用されるシフト出力を決定します。シフトが発生しない場合、sor と sol_msb 信号はそれぞれ ALU 機能の LSB か MSB として定義されます。

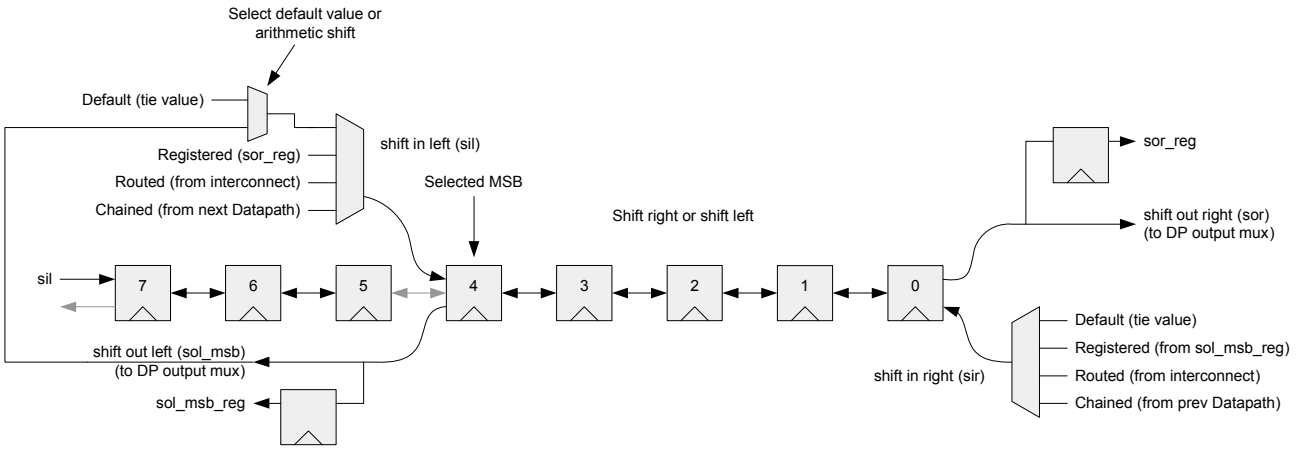
SI SELA と SI SELB コンフィギュレーション ビットは特定の動作のシフト イン データを決定します。動的コンフィギュレーション RAM はサイクルごとに A コンフィギュレーションか B コンフィギュレーションを選択します。シフト イン データは左シフトと右シフトのみに使用可能です ; PASS とニブル スワップにはそのデータを使用できません。表 16-13 は両方の左シフトと右シフトに適応する選択肢とその使用法を示しています。

表 16-13. UDB CFG15 レジスタによるシフト イン機能

SI SEL A SI SEL B	シフト イン ソース	説明
00	デフォルト／算術	デフォルト入力は DEF SI コンフィギュレーション ビットの値 (1 または 0 に固定済み)。ただし、MSB SI ビットが設定されると、そのデフォルト入力は現時点定義済み MSB の値になる (右シフト専用)
01	レジスタ出力	シフト イン値は現時点登録済みシフト アウト値 (1 つ前のサイクルからの値) により駆動される。左シフト動作は最後の左シフト アウト値を利用する。右シフト動作は最後の右シフト アウト値を利用する
10	配線	シフト インは配線チャネル (SI 入力) から選択される
11	連結	左シフト インは右近隣のデータパスから受信され、右シフト インは左近隣のデータパスから受信される

左シフトアウト データは当時定義されている MSB (CFG14 レジスタの MSB_EN と MSB_SEL ビット) 位置から配線され、(右シフト動作で) 左からシフト インされたデータは当時定義されている MSB 位置にロードされます。両方 (左または右) のシフトアウト データはレジスタ出力データで、後続のサイクルで使用可能になります。この特長は、多くのサイクルでより高精度のシフトを実装するために使用されます。

図 16-16. シフト動作



MSB 選択により分離したビットは依然としてシフトされることに注意してください。上記の例を見ると、右シフトの場合にビット 7 は依然として sil の値をシフト インし、左シフトの場合にビット 5 はビット 4 の値をシフト インします。分離したビットから右シフトアウトか左シフトアウトの値が失われます。

ALU マスク動作

UDB スタティック コンフィギュレーション レジスタ空間 (CFG9) 内の 8 ビット マスク レジスタ (AMASK) はマスク動作を定義します。この動作では、ALU の出力はマスクレジスタの値とマスク処理 (AND) されます。ALU マスク機能の基本的な使用法はフリーランニング タイマーとカウンターを分解能の 2 のべき乗で実装することです。

16.2.2.5 データパス入力と多重化

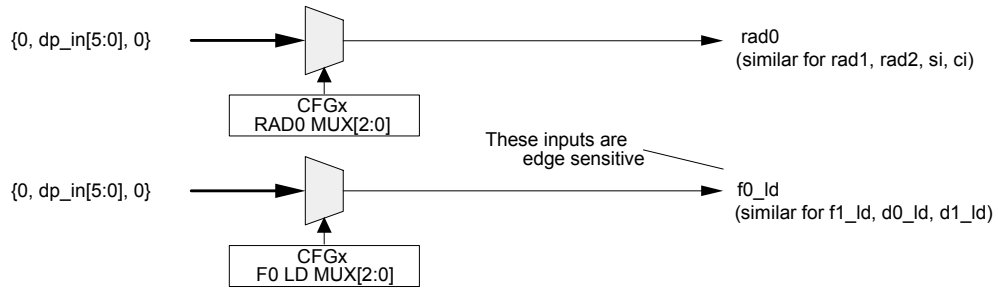
データパスは表 16-14 に示している通りに、合計で 9 個の入力があり、その中 6 個の入力はチャネル配線から得たものです。これらは、コンフィギュレーション RAM アドレス、FIFO とデータ レジスタ ロード制御信号およびデータ入力シフト インとキャリー インからなります。

表 16-14. データパス入力

入力	説明
RAD2 RAD1 RAD0	非同期ダイナミック コンフィギュレーション RAM アドレス。ユーザー プログラム可能な 16 ビット ワードを 8 つ含む。それぞれのワードには、当時のサイクル専用のデータパス制御ビットがある。一連の命令はこれらのアドレス入力により制御される
F0 LD F1 LD	特定のサイクルでアサートされた場合、選択した FIFO に A0 や A1 アキュムレータからあるいは ALU 出力からデータが引き渡される。そのデータ ソースは Fx INSEL[1:0] コンフィギュレーション ビットにより選択される。この入力はエッジ センシティブである。データパス クロックに応じてサンプル処理が行われる ; 「0」→「1」遷移が検出されると、次のクロック エッジでデータ引き渡しが行われる
D0 LD D1 LD	特定のサイクルでアサートされた場合、Dx レジスタの値は関連する FIFO Fx から書き込まれる。この入力はエッジ センシティブである。データパス クロックに応じてサンプル処理が行われる ; 「0」→「1」遷移が検出されると、次のクロック エッジでデータ引き渡しが行われる
SI	これはデータ入力値で、左シフト インと右シフト インのいずれか 1 つに使用される
CI	これはキャリー イン値で、キャリー イン選択制御信号が配線キャリーに設定される場合に使用される

図 16-17 に示すように、それぞれの入力は 6 : 1 マルチプレクサを備えるため、すべての入力は入れ替え可能になります。入力はレベル センシティブかエッジ センシティブといった 2 つの方法からどれか 1 つに従って処理されます。RAM アドレス、シフトイン値、データ入力値はレベル センシティブで、FIFO とデータ レジスタ ロード信号はエッジ センシティブです。

図 16-17. データパス入力選択



16.2.2.6 CRC、PRS サポート

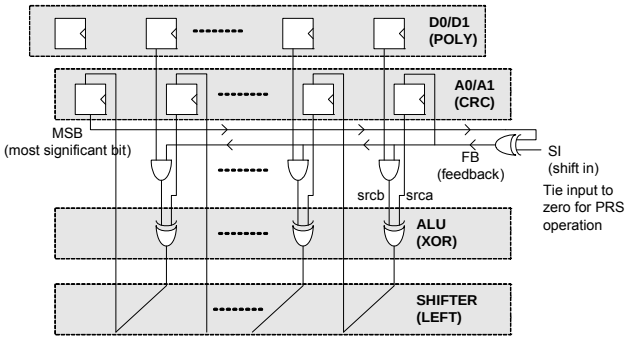
データパスは巡回冗長検査 (CRC) と疑似ランダム シーケンス (PRS) 生成をサポートできます。連結されている信号はデータパス ブロック間に接続されており、8 ビット以上の CRC、PRS ビット長さに対応できます。

CRC、PRS 計算の際には最上位ブロック内の最上位ビット (MSB) が選択され、連結された全てのブロックを通り最下位ブロックに配線されます。その後、MSB はデータ入力 (SI データ) と XOR されてフィードバック信号 (FB) を形成します。FB 信号はまた、連結された全てのブロックを通り最上位ブロックに配線されます。このフィードバック値は全てのブロックに使用され、アキュムレータの現在値を (Data0 または Data1 レジスタからの) 多項式の XOR ゲートに入れます。

図 16-18 は CRC 動作の構造を示しています。PRS コンフィギュレーションは、シフトイン (SI) が「0」に固定されることの他、同じです。PRS コンフィギュレーションで、D0 または D1 は多項式の値を含んでいますが、A0 または A1 は初期値 (シード値) と計算後の CRC 残留値を含んでいます。

CRC 動作を有効にするためには、動的コンフィギュレーション RAM 内の CFB_EN ビットを「1」に設定する必要があります。これにより、SRCB ALU 入力を CRC フィードバック信号と AND することが可能になります。「0」に設定されると、フィードバック信号は「1」に駆動され、通常の算術演算が可能になります。サイクルごとにこのビットを動的制御すると、CRC、PRS 動作を他の算術演算とインターリーブすることが可能になります。

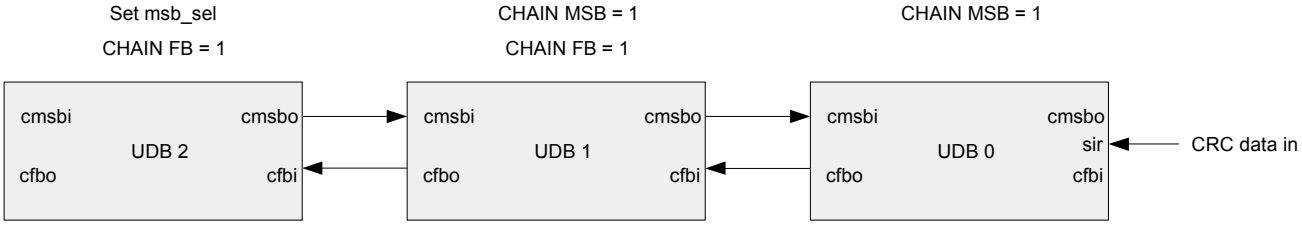
図 16-18. CRC の構造



CRC、PRS 連結

図 16-19 は 3 つの UDB を連結する CRC、PRS の例を説明します。このシナリオでは、17 ～ 24 ビット動作に対応します。連結の制御ビットは、図に示す通り、連結体のデータパスの位置に従って設定されます。

図 16-19. CRC、PRS 連結コンフィギュレーション



CRC、PRS フィードバック信号 (cfbo、cfbi) は次の通りに連結されます。

- 当該ブロックが最下位ブロックの場合、フィードバック信号は、そのブロック内の内蔵のロジック (右のシフトイン (sir) を MSB 信号と XOR するロジック) から生成されます。(PRS の場合、「sir」信号は「0」の値に固定されます。)
- 当該ブロックが最下位ブロックでない場合、CHAIN FB コンフィギュレーション ビットを設定する必要があり、フィードバック信号は連結体の前のブロックから送信されてきます。

CRC、PRS の MSB 信号 (cmsbo、cmsbi) は次の通りに連結されます。

- 当該ブロックが最上位ブロックの場合、(選択した多項式に従う) MSB ビットは UDB CFG14 レジスタの MSB_SEL コンフィギュレーション ビットによって設定されます。
- 当該ブロックが最上位ブロックでない場合、UDB CFG14 レジスタの CHAIN CMSB コンフィギュレーション ビットを設定する必要があり、MSB 信号は連結体の後続のブロックから送信されてきます。

CRC、PRS 多項式の指定

関連する D0 ／ D1 レジスタに多項式をプログラムする例として、「$x^{16} + x^{12} + x^5 + 1$」と定義される CCITT CRC-16 多項式を取り上げます。データ フォーマットを多項式から生成する方法を図 16-20 に示します。

X^0 項は本質的に常に「1」であるため、プログラムする必要がありません。多項式内の残りの項はそれぞれ、以下に示している整列の該当する位置を「1」の値に設定します。

注：この多項式のフォーマットは 16 進数で指定されるフォーマットと若干異なっています。例えば、CCITT CRC16 多項式は通常 1021H と示しています。データパス動作に必要なフォーマットに変換するためには、右に 1 シフトして、MSB ビットに「1」の値を追加します。この場合には D0 か D1 レジスタにロードする多項式の正しい値は 8810H です。

図 16-20. CCITT CRC16 多項式フォーマット

X^{16}	X^{15}	X^{14}	X^{13}	X^{12}	X^{11}	X^{10}	X^9	X^8	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
X^{16}		+		X^{12}		+					X^5		+			1
1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	

CCITT 16-Bit Polynomial is 0x8810

CRC、PRS コンフィギュレーションの例

次に CRC、PRS コンフィギュレーションの要件をまとめます。D0 は多項式で、CRC、PRS は A0 に計算するものとします。

1. 適切な多項式を選択して D0 に書き込みます。
2. 適切なシード値 (例えば CRC に「全て 0」、PRS に「全て 1」) を選択して A0 に書き込みます。
3. 必要に応じて連結設定をします。
4. 多項式に定義した MSB 位置をスタティック コンフィギュレーション レジスタの MSB_SEL ビットから選択し、UDB CFG14 レジスタ内の MSB_EN レジスタ ビットを設定します。
5. 次の通りに動的コンフィギュレーション RAM のワードフィールドを設定します :
   a. D0 を ALU 「SRCB」(ALU B 入力ソース) として選択
   b. A0 を ALU 「SRCA」(ALU A 入力ソース) として選択
   c. ALU 機能のために「XOR」を選択
   d. シフト機能のために「SHIFT LEFT」を選択
   e. CRC、PRS サポートを有効にするために「CFB_EN」を選択
   f. ALU を A0 書き込みソースとして選択

CRC 動作の場合、配線からの入力データに「右シフトイン」を設定し、クロックごとにその入力を提供します。PRS 動作の場合、「右シフトイン」を「0」に固定します。

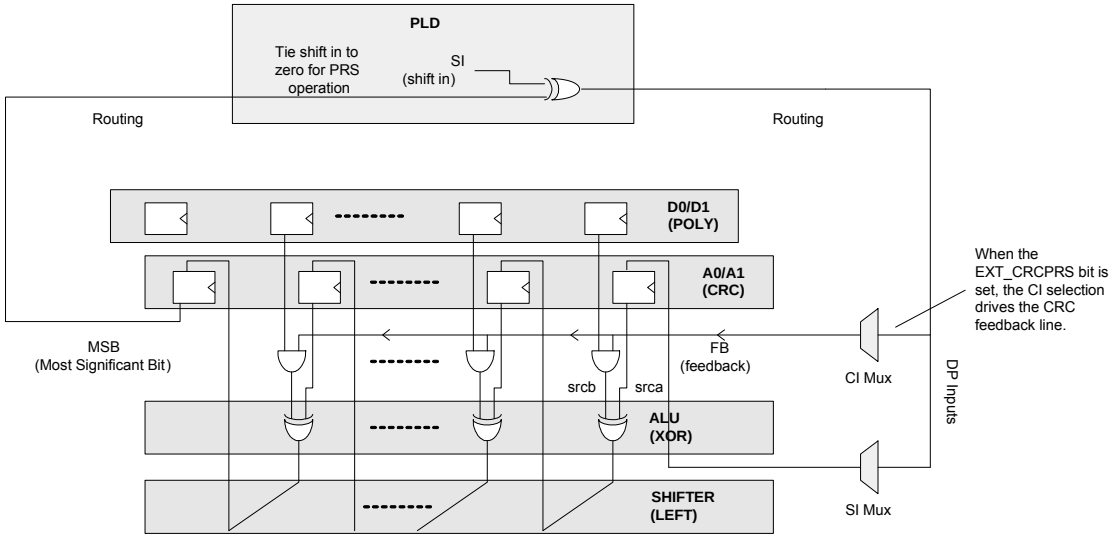
このコンフィギュレーションでは、UDB のクロック供給は必要となる CRC を生成するか、または PRS シーケンスのために配線に出力される MSB を出力します。

外部 CRC、PRS モード

CRC または PRS の外部計算が可能になるためには、スタティック コンフィギュレーション ビット (UDB CFG16 レジスタの EXT CRCPRS) を設定することができます。図 16-21 に示すように、CRC フィードバックの計算は PLD ブロック内で実現します。このビットがセットされると、CRC フィードバック信号は内部計算せずに CI (キャリーイン) データパス入力選択マルチプレクサから直接駆動されます。この図は最大 8 ビット CRC または PRS をサポートする簡単なコンフィギュレーションを示しています。通常は内蔵の回路が使用されますが、時分割多重化モードを使用することで、この機能は、1 つの UDB 内に最大 16 ビットの CRC、PRS 機能など比較的複雑なコンフィギュレーションを可能にさせます。

このモードでも、UDB DCFG0 レジスタ内の動的コンフィギュレーション RAM ビット CFB_EN は、CRC フィードバック信号が SRCB ALU 入力と AND されたかどうかを制御します。そのため内蔵の CRC、PRS 動作と同様に、この機能は必要に応じて他の機能とインターリーブすることができます。

図 16-21. 外部 CRC、PRS モード



16.2.2.7 データパス出力と多重化

各条件はレジスタされたアキュムレータ値、ALU 出力と FIFO 状態から生成されます。これらの条件は、他の UDB ブロックに使用されるかあるいは割り込みとして使用されるためにデジタル配線に駆動されるか、または I/O ピンに駆動されます。表 16-15 には 16 個の可能な条件を示しています。

表 16-15. データパス条件生成

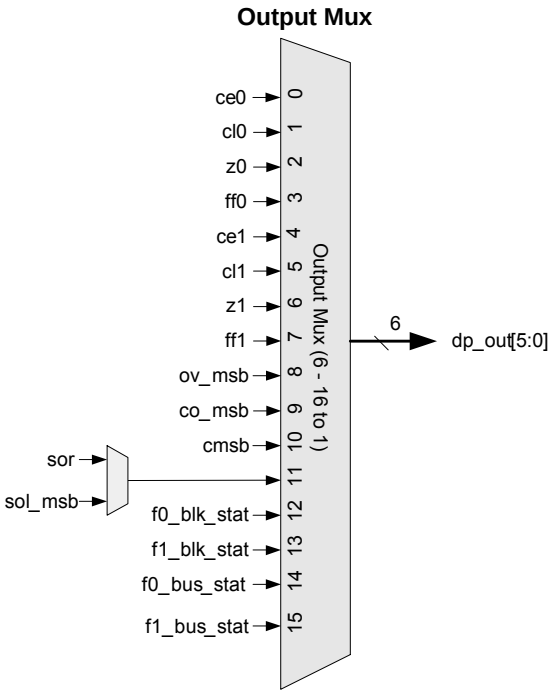
名称	条件	連結	説明
ce0	「等しい」比較	有	A0 == D0
cl0	「小なり」比較	有	A0 < D0
z0	0 検出	有	A0 == 00h
ff0	「1」検出	有	A0 == FFh
ce1	「等しい」比較	有	A1 または A0 == D1 または A0 (ダイナミック選択)
cl1	「小なり」比較	有	A1 または A0 < D1 または A0 (ダイナミック選択)
z1	0 検出	有	A1 == 00h
ff1	「1」検出	有	A1 == FFh
ov_msb	オーバーフロー	無	Carry(msb) ^ Carry(msb–1)
co_msb	キャリーアウト	有	MSB キャリー アウト定義ビット
cmsb	CRC MSB	有	CRC、PRS 機能の MSB
So	シフトアウト	有	シフト出力選択

表 16-15. データパス条件生成

f0_blk_stat	FIFO0 ブロック ステータス	無	FIFO コンフィギュレーションに従って定義される
f1_blk_stat	FIFO1 ブロック ステータス	無	FIFO コンフィギュレーションに従って定義される
f0_bus_stat	FIFO0 バス ステータス	無	FIFO コンフィギュレーションに従って定義される
f1_bus_stat	FIFO1 バス ステータス	無	FIFO コンフィギュレーションに従って定義される

データパス出力は合計で 6 個あります。図 16-22 に示すように各出力は 1 個の 16 : 1 マルチプレクサを備えるため、これらの 16 信号の 1 つが任意のデータパス出力に配線できます。

図 16-22. 出力マルチプレクサの接続



比較

2 個のコンパレータがあり、その内 1 つは固定ソースを持ち (コンパレータ 0) 、残りの比較は動的選択可能なソース (コンパレータ 1) を持っています。それぞれのコンパレータはスタティックにプログラムされた 8 ビット マスク レジスタを持っており、比較が指定したビット フィールドで実行可能になります。デフォルトではマスク機能がオフ (すべてのビットが比較される)にされますがそれを有効にする必要があります。

コンパレータ 1 の入力は動的にコンフィギュレーションされます。表 16-16 に示す通りにコンパレータ 1 には 4 つのオプションがあり、「小なり」と「等しい」条件が適用できます。UDB CFG12 レジスタ内の CMP SELA と CMP SELB コンフィギュレーション ビットは可能な比較コンフィギュレーションを決めます。UDB DCFG0 レジスタの動的コンフィギュレーション RAM ビット CMP SEL はサイクルごとに A コンフィギュレーションか B コンフィギュレーションを選択します。

表 16-16. コンペア コンフィギュレーション

CMP SEL A CMP SEL B	コンパレーター 1 の コンペア コンフィギュレーション
00	A1 を D1 と比較
01	A1 を A0 に比較
10	A0 を D1 に比較
11	A0 を A0 に比較

コンパレータ 0 とコンパレータ 1 は、前のデータパスにて生成された条件と (アドレス順番に従って) 単独で連結できます。比較の連結の決定は UDB CFG14 レジスタの CHAIN0 と CHAIN1 ビットによりスタティックに指定されます。図 16-23 は「等しい」比較の連結を示しています。これは単にこのブロックの「等しい」比較と前のブロックの連結された入力を AND したものです。

図 16-23. 「等しい」比較の連結

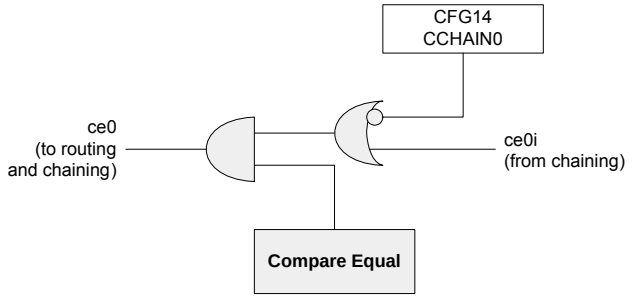
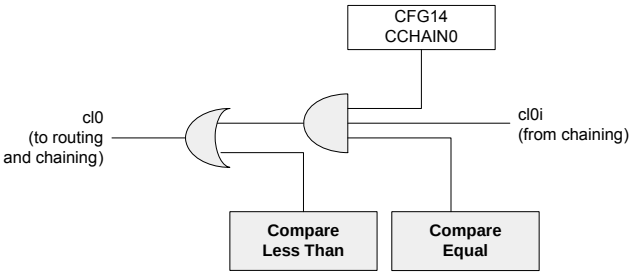


図 16-24 は「小なり」比較の連結を説明します。この場合には、「小なり」はこのブロックの「小なり」比較の出力 (無条件) から形成します。それを、このブロックが「等しい」比較と前のブロックから連結された入力が「小なり」としてアサートされた条件と OR します。

図 16-24. 「小なり」比較の連結



すべて 0 とすべて 1 の検出

アキュムレータはそれぞれ専用のすべて0検出とすべて1検出があります。これらの条件は UDB コンフィギュレーション レジスタにて指定した通りにスタティックに連結できます。これらの条件の連結の決定は UDB コンフィギュレーション レジスタにてスタティックに指定されます。0 検出の連結は「等しい」比較と同様です。連結が有効の場合、連続連結のデータが AND されます。

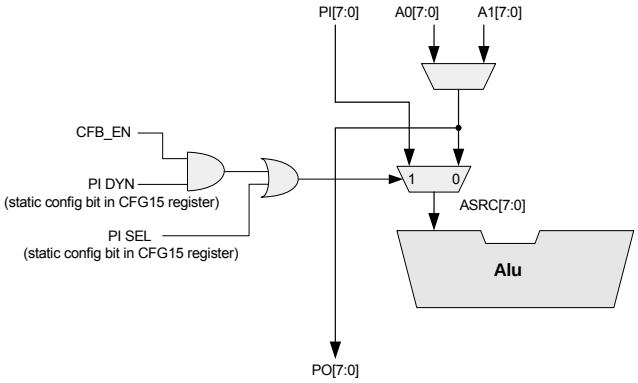
オーバーフロー

オーバーフローは MSB に入るキャリーインと MSB のキャリーアウトを XOR したものとして定義されます。その計算は MSB_SEL ビットによって指定された通りに、現時点の MSB にて実現されます。この条件は連結できませんが、キャリーが各ブロックを通して連結されている限り、高精度機能により最上位データパスで実行されるとその計算は有効になります。

16.2.2.8 データパスのパラレル入出力

図 16-25 に示すように、データパスのパラレル入力 (PI) とパラレル出力 (PO) 信号により、条件付きで、配線データを直接データパスとやりとりできます。パラレル出力信号は A0 か A1 の ALU asrc 選択として常に送信可能です。

図 16-25. データパスのパラレル入力／出力

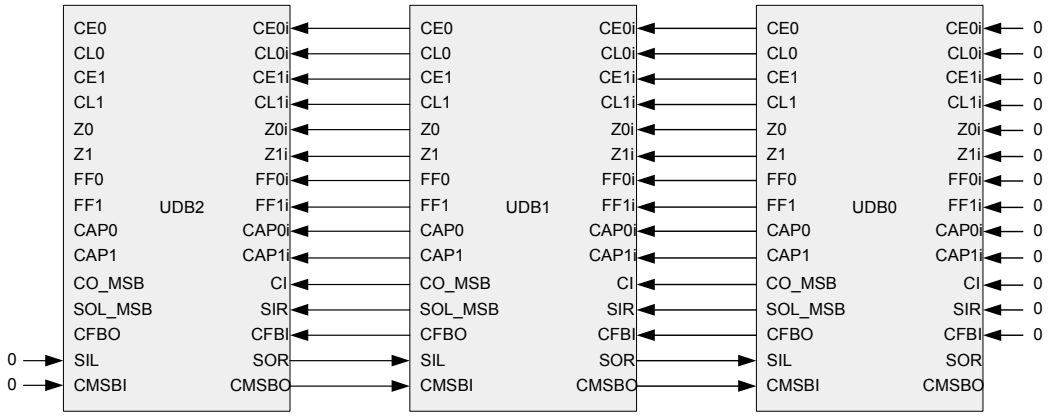


パラレル入力は ALU への入力のために選択される必要があります。2 つの可能なオプションはスタティック動作と動的動作です。スタティック動作の場合、UDB CFG15 レジスタの PI SEL ビットは ALU asrc を PI になるように強制します。UDB CFG15 レジスタの PI DYN ビットは PI の動的動作を有効にするためです。これが有効になって PI SEL が 0 と仮定すると、PI マルチプレクサは (UDB DCFG0 レジスタの) 動的制御ビット CFB_EN により制御されます。CFB_EN の主な機能は CRC、PRS 機能を有効にすることです。

16.2.2.9 データパスの連結

各データパス ブロックは 8 ビット ALU を内蔵しています。この ALU は、キャリー、シフト データ、取り込みトリガーおよび条件付き信号を最も近いデータパスに連結してより高精度の算術機能とシフターを実現します。これらの連結信号 (専用信号) により、チャネル配線供給源の不確実性に左右されずに 16 ビット／24 ビット／32 ビット シングル サイクル関数を効果的に実現できます。さらに、取り込み連結により、連結されるブロック内のアキュムレータへのアトミックな読み出しを実現できます。図 16-26 に示すように、すべての生成された条件付き信号とキャプチャ信号は、最下位ブロックから最上位ブロックへの方向で連結されます。左シフトも最下位から最上位ブロックへの方向で連結されます。右シフトは最上位から最下位ブロックへの方向で連結されます。フィードバックのための CRC、PRS 連結信号は最下位から最上位に連結されますが、MSB 出力は最上位から最下位への方向で連結されます。

図 16-26. データパスの連結フロー

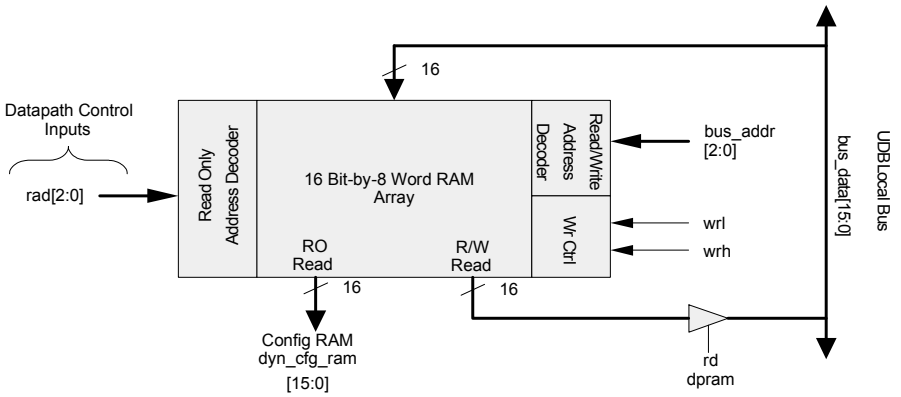


16.2.2.10 動的コンフィギュレーション RAM

図 16-27 に示すように各データパスには 8 本の 16 ビット ワード コンフィギュレーション RAM を備えます。この RAM の目的は、該当するデータパスのクロックに応じてサイクルごとにデータパス コンフィギュレーション ビットを制御するためです。この RAM はシステム バスを通してコンフィギュレーションをロードするための同期読み出しと書き込みポートがあります。

非同期読み出しポートは、これらの 16 ビット ワードをデータパスへの制御ビットとして出力する高速パスとして提供されます。非同期アドレス入力はデータパス入力から選択され、I/O ピン、PLD 出力、制御ブロック出力または他のデータパス出力を含むチャネル配線上の可能な信号のいずれか 1 つから生成されます。非同期読み出しパスの主な目的はデータパス制御ビットのために高速なシングル サイクルの複号化を提供することです。

図 16-27. コンフィギュレーション RAM I/O



次はこの動的コンフィギュレーション RAM ワードのフィールドを説明します。それぞれのフィールドの使用法の説明も下記に搭載しています。

レジスタ	アドレス	15	14	13	12	11	10	9	8
CFGRAM	61h ～ 6Fh (奇数)	FUNC[2:0]			SRCA	SRCB[1:0]		SHIFT[1:0]	

レジスタ	アドレス	7	6	5	4	3	2	1	0
CFGRAM	60h ～ 6Eh (偶数)	A0 WR SRC[1:0]		A1 WR SRC[1:0]		CFB EN	CI SEL	SI SEL	CMP SEL

表 16-17. ダイナミック コンフィギュレーションのクイック リファレンス

フィールド	ビット	パラメーター	値
FUNC[2:0]	3	ALU 機能	000 PASS 001 INC SRCA 010 DEC SRCA 011 ADD 100 SUB 101 XOR 110 AND 111 OR
SRCA	1	ALU A 入力ソース	0 A0 1 A1
SRCB	2	ALU B 入力ソース	00 D0 01 D1 10 A0 11 A1
SHIFT[1:0]	2	SHIFT 機能	00 PASS 01 左シフト 10 右シフト 11 ニブル スワップ
A0 WR SRC[1:0]	2	A0 書き込みソース	00 なし 01 ALU 10 D0 11 F0
A1 WR SRC[1:0]	2	A1 書き込みソース	00 なし 01 ALU 10 D1 11 F1
CFB EN	1	CRC フィードバック イネーブル	0 有効 1 無効
CI SEL	1	キャリー イン コンフィギュレーションの選択	0 ConfigA 1 ConfigB[a]
SI SEL	1	シフト イン コンフィギュレーションの選択	0 ConfigA 1 ConfigB[a]
CMP SEL	1	コンペア コンフィギュレーションの選択	0 ConfigA 1 ConfigB[a]

a. CI、SI、CMP に対して、RAM フィールドは、事前定義された 2 つのスタティック設定からいずれか 1 つ選択します。スタティック レジスタ コンフィギュレーションをご覧ください。

16.2.3 ステータスおよび制御モジュール

図 16-28 はステータスおよび制御モジュールの概要について示します。制御レジスタは UDB 動作のために配線に駆動しファームウェア制御入力を提供します。配線によるステータス レジスタからの読み出しはUDB動作の状態を監視する方法をファームウェアに提供します。

図 16-29 はステータスおよび制御モジュールの詳細を示します。このブロックの主な目的は、CPU ファームウェアと内部 UDB 動作とのやりとりを調整することです。ただし配線マトリクスとの様々な接続により、このブロックは他の機能を実現するためにも設定できます。

図 16-28. ステータスおよび制御レジスタ

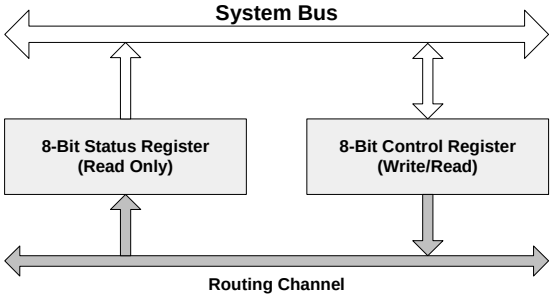
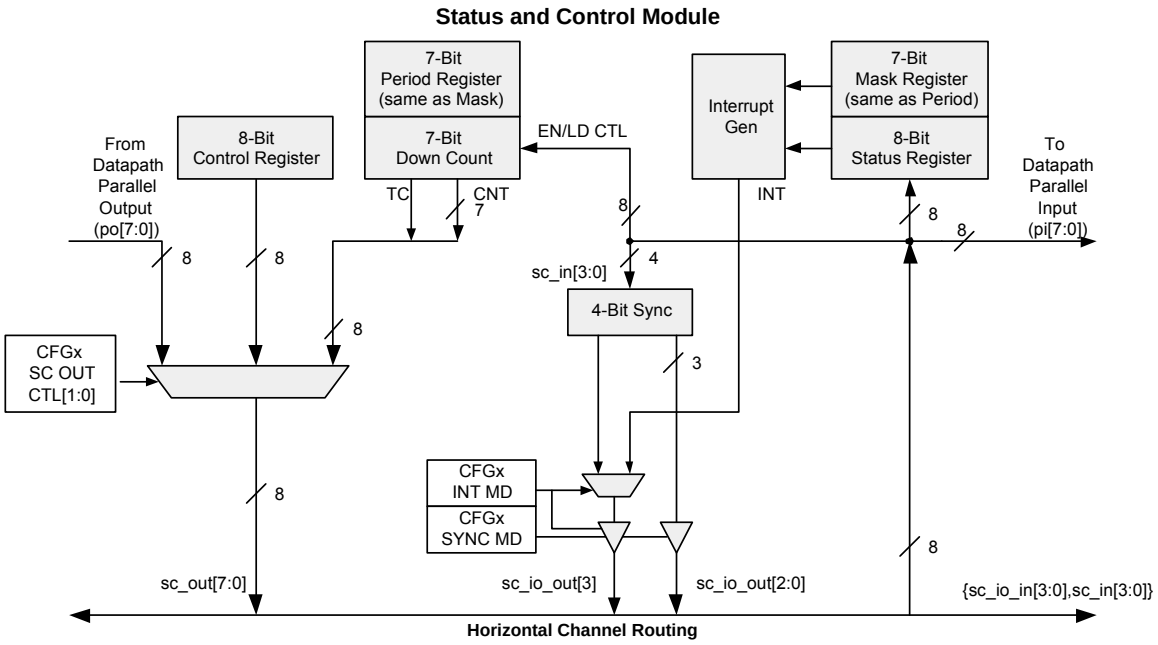


図 16-29. ステータスおよび制御モジュール



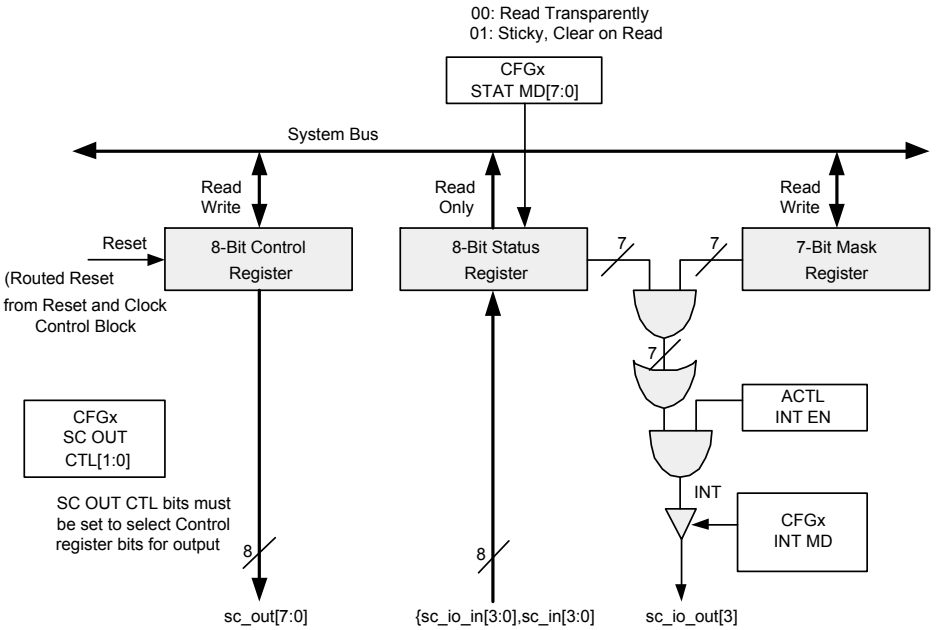
動作モードは以下の通りです。

- **ステータス入力** – 配線信号の状態は入力で、ステータスとして取り込まれて CPU により読み出されます。
- **制御出力** – CPU は制御レジスタに書き込んで配線の状態を駆動することができます。
- **パラレル入力** – データパスのパラレル入力への信号
- **パラレル出力** – データパスのパラレル出力からの信号
- **カウンター モード** – このモードでは、制御レジスタは 7 ビット ダウン カウンターとして動作し、周期プログラム可能かつ自動リロード機能を持っています。配線入力はカウンターの有効化とそのリロードの両方を制御するために設定可能です。このモードが有効の場合、制御レジスタの動作は使用不可です。
- **同期モード** – このモードではステータス レジスタは 4 ビット ダブル シンクロナイザーとして動作します。このモードが有効の場合、ステータス レジスタとしての動作は利用不可です。

16.2.3.1 ステータスおよび制御モード

図 16-30 のコンフィギュレーションの通りに、ステータスおよび制御モードで動作する場合に、このモジュールはステータス レジスタ、割り込みマスク レジスタおよび制御レジスタとして機能します。

図 16-30. ステータスおよび制御の動作



ステータス レジスタの動作

各 UDB は 1 個の 8 ビット読み出し専用ステータス レジスタを内蔵しています。このレジスタの入力はデジタルの配線構造の任意の信号です。ステータス レジスタはデータが保持されず、スリープ期間中にその状態がなくなり、ウェイクアップ時に 0x00 にリセットされます。それぞれのビットは独立してプログラムされて表 16-18 に示すように 2 つの方法のいずれかに従って動作できます。

表 16-18. UDB CFG20 レジスタによるステータス レジスタ モード選択

STAT MD	説明
0	トランスペアレント読み出しモード。読み出し結果として、配線された信号の現在値を返す
1	スティッキー モードで、読み出し時にクリアされる。入力 HIGH がサンプル処理されて取り込まれる。これはレジスタが読み出される時クリアされる

ステータス レジスタのクリア動作の重要な特長として、ステータスのクリアはセットされたビットのみに適応されることがあります。これにより、セットされない他のビットのステータスを引き続き取り込むことができ、プロセスの一貫性を維持できます。

トランスペアレント ステータスの読み出し

デフォルトでは、このレジスタの CPU 読み出しは関連する配線の状態を透明的に読み出します。このモードは、UDB 内部で計算しレジスタされた過渡状態のために使用されます。

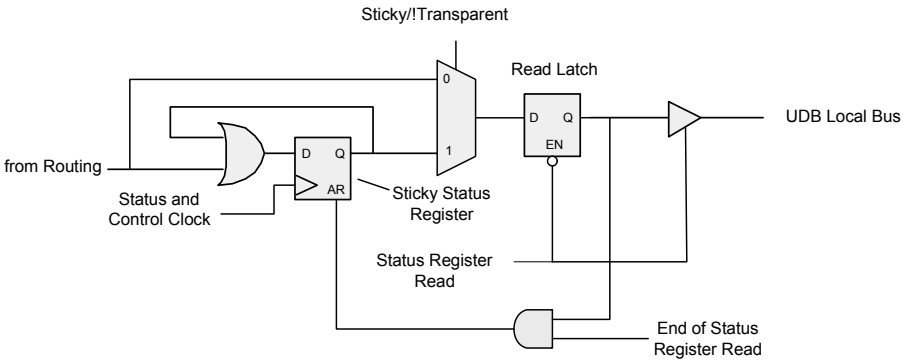
スティッキー状態、読み出すとクリア

このモードでは、ステータス レジスタ入力はステータスおよび制御クロックの各サイクルごとにサンプルされます。所定のサンプルで信号が HIGH の場合、ステータス ビットに取り込まれ、入力のその後の状態に関係なく HIGH のままです。CPU がステータス レジスタを読み出すと、このビットがクリアされます。ステータス レジスタのクリアは、モードに依存せず、UDB クロックが無効の場合でも発生します。HFCLK に基づいて動作し、読み出し動作の一部として発生します。

読み出し中のステータス ラッチ

図 16-31 はステータス読み出しロジックの構造を示します。スティッキー ステータス レジスタはラッチ回路の前に位置します。このラッチ回路はステータス レジスタ データをラッチし、所定の読み出しに対するウェイト ステート数に関わらず読み出しサイクル中にそのデータを安定した状態に保持します。

図 16-31. ステータス読み出しロジック



割り込みの生成

殆どの機能では、割り込み生成はステータス ビットの設定と結びついています。図 16-31 に示すように、この特長はマスク レジスタとステータス レジスタ削減とのAND OR後の結果としてステータス レジスタ ロジックに内蔵します。ステータス入力の下位 7 ビットのみが内蔵の割り込み生成回路に使用されます。最上位ビットは基本的に割り込み出力として使用され、デジタル配線により割り込みコントローラーに配線されます。このコンフィギュレーションでは、ステータス レジスタのMSBは割り込みビットの状態として読み出されます。

16.2.3.2 制御レジスタの動作

各 UDB は 1 個の 8 ビット制御レジスタを内蔵しています。このレジスタはシステム バス上に標準の読み出し／書き込みレジスタとして動作し、それらのレジスタ ビットの出力はデジタル配線構造のドライバーとして選択可能です。

制御レジスタはデータが保持されず、スリープ期間中にその値がなくなり、ウェイクアップ時に 0x00 にリセットされます。

制御レジスタの動作モード

使用可能なモードは 3 つあり、ビット単位で設定できます。そのコンフィギュレーションは 2 個の 8 ビット レジスタ (UDB CFG18 と CFG19 レジスタの CTL_MD1[7:0] と CTL_MD0[7:0]) のビットを連結することで制御されます。例えば、表 16-19 に示すように {CTL_MD1[0],CTL_MD0[0]} は制御レジスタのビット 0 のモードを制御します。

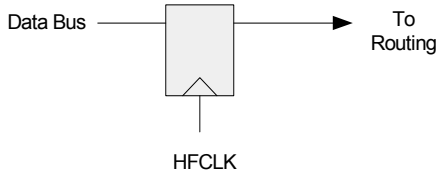
表 16-19. UDB CFG18 と CFG19 レジスタによる制御レジスタ ビット 0 のモード

CTL MD	説明
00	直接モード
01	同期モード
10	ダブル同期モード
11	パルス モード

制御レジスタの直接モード

デフォルト モードは直接モード図 16-32 に示すように、制御レジスタの値が CPU から書き込まれる時に、制御レジスタの出力はその書き込みサイクルに配線まで直接駆動されます。

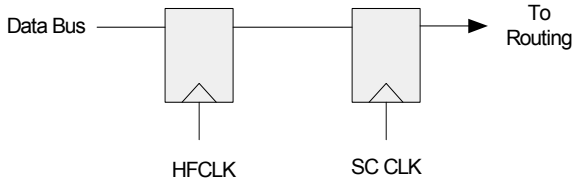
図 16-32. 制御レジスタの直接モード



制御レジスタの同期モード

図 16-33 に示すように、同期モードでは、制御レジスタの出力は現在選択されているステータスおよび制御 (SC) クロックによりクロック供給され、再サンプル レジスタで駆動されます。これにより、出力のタイミングは HFCLK ではなく選択した SC クロックにより制御されます。

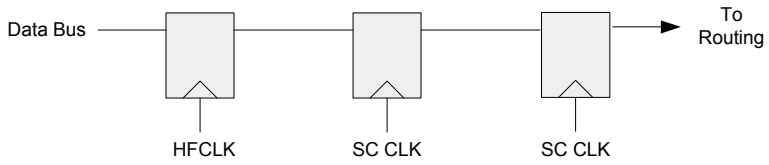
図 16-33. 制御レジスタの同期モード



制御レジスタのダブル同期モード

図 16-34 に示す通りに、ダブル同期モードでは選択した SC クロックでクロック供給された 2 番目のレジスタが再サンプル レジスタの後に追加されます。これにより、HFCLK と SC クロックが非同期の場合にも回路は確実に実行できます。

図 16-34. 制御レジスタのダブル Sync モード



制御レジスタのパルス モード

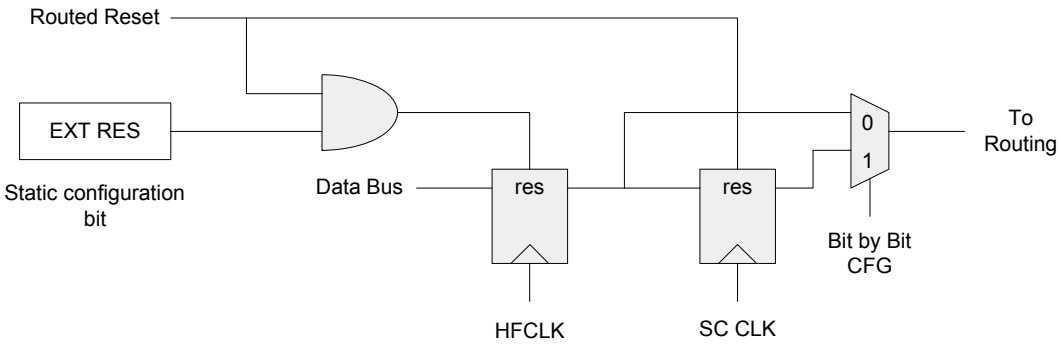
制御ビットが SC クロックにより再サンプルされる点で、パルス モードは同期モードと同様です。パルスはバス書き込みサイクルの次のサイクル (最初の SC クロック サイクル) で開始します。制御ビットの出力は、1 SC クロックサイクルの間アサートされます。このクロック サイクルの終わりに、制御ビットは自動的にリセットされます。

この動作モードで、ファームウェアは制御レジスタに「1」の値を書き込んでパルスを生成することが可能になります。「1」として書き込まれた後、パルス完了までファームウェアによりまた「1」として読み出され、その後「0」として読み出されます。ファームウェアは次に別の「1」の値を書き込んで他のパルスを開始することができます。実行中のパルスが完成していなければ、新しいパルスを生成できません。そのため、パルス生成の最大周波数は隔 SC クロックサイクルになります。

制御レジスタのリセット

図 16-35 に示すように、制御レジスタは 2 個のリセット モードがあり、EXT RES コンフィギュレーション ビットにより制御されます。EXT RES が 0 (デフォルト値) の場合、同期モードかパルス モードに配線リセット入力は実際の制御ビットではなく同期した出力をリセットします。EXT RES が 1 の場合、配線リセット入力は制御ビットと同期された出力の両方をリセットします。

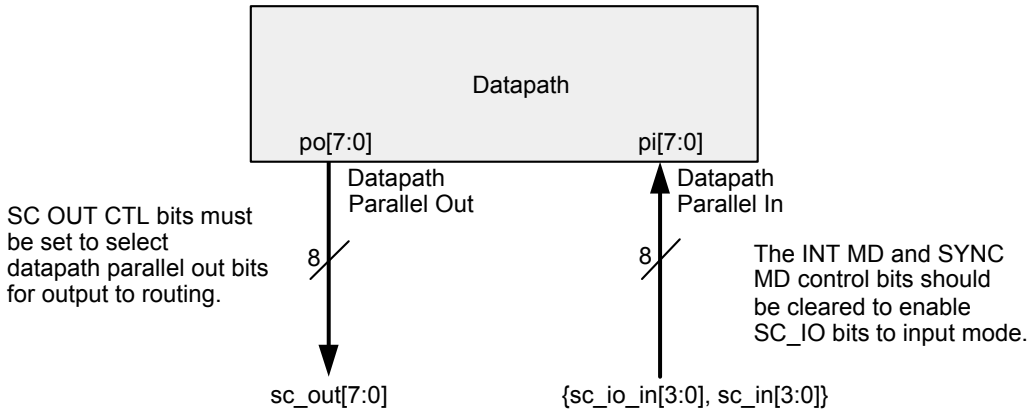
図 16-35. 制御レジスタのリセット



16.2.3.3 パラレル入力／出力モード

図 16-36 に示すように、このモードでは、ステータスおよび制御の配線はデータパスのパラレル入力とパラレル出力信号に接続されます。このモードを有効にするために、UDB CFG22 レジスタの SC OUT コンフィギュレーション ビットをセットしてデータパスのパラレル出力を選択します。パラレル入力の接続は常に有効ですが、これらの配線接続はステータス レジスタ入力、カウンター制御入力および割り込み出力と共有します。

図 16-36. パラレル入力／出力モード



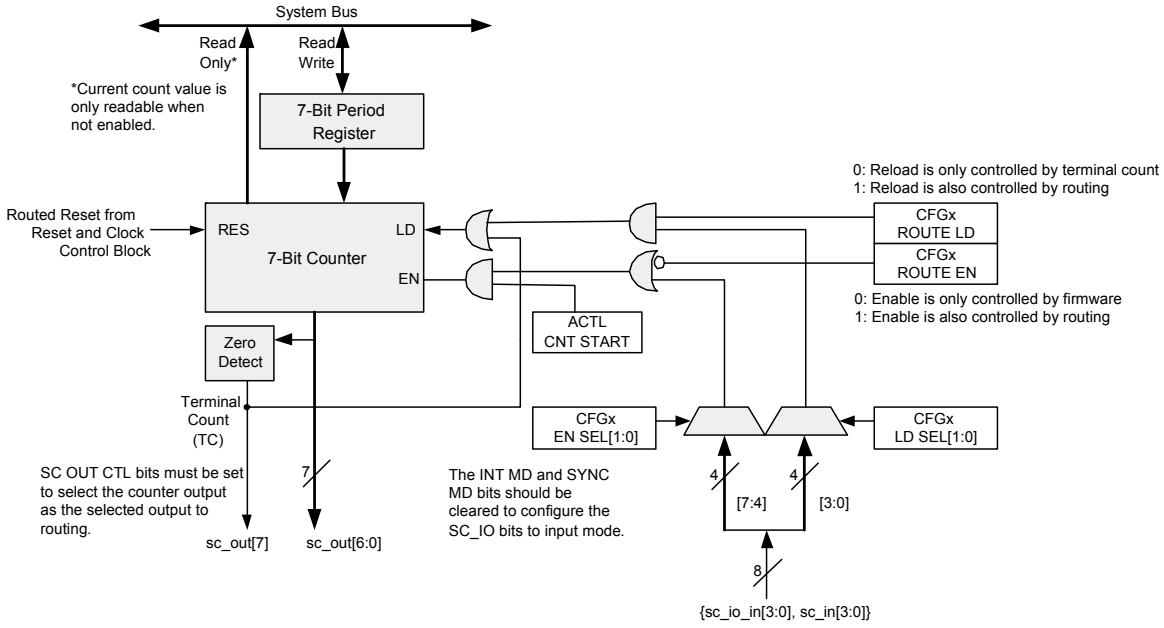
16.2.3.4 カウンター モード

図 16-37 に示すように、ブロックがカウンター モードの場合、UDB の内部動作かファームウェア アプリケーションにより、7 ビット ダウン カウンターとして機能します。このカウンターは次の特長を持っています。

- 7 ビット読み出し／書き込み周期レジスタ
- 7 ビット読み出し／書き込みカウントー レジスタカウンターが無効の場合にのみアクセス可能
- カウント終了時点 (0) でカウント レジスタに周期の値を自動的に再ロード
- カウンター開始／終了用の補助制御ワーキングレジスタ (ACTL0) の CNT START というファームウェア制御ビット (この信号は他の信号をオーバーライドします。外部から配線されるイネーブル信号が動作するためにセットする必要があります。)
- カウンター有効化とロード機能をオプションとして動的制御するために配線から選択可能なビット :
  - EN、カウントを開始／終了させる配線イネーブル信号
  - LD、周期を強制的に再ロードさせる配線ロード信号 この信号がアサートされると、保留中のターミナル カウントをオーバーライドする。それはレベル センシティブで、アサートされている間でも周期を引き続きロードする。
- 7ビット カウントはsc_out[6:0]として配線構造まで駆動可能
- ターミナル カウントは sc_out[7] として配線構造まで駆動可能
- デフォルト モードでは、ターミナル カウントはレジスタ回路出力。予備モードでは、ターミナル カウントは組み合わせ回路出力。
- デフォルト モードで配線イネーブルとされる場合、配線ロード信号動作のために配線イネーブル信号はアサートされる必要がある。予備モードでは、配線イネーブル信号と配線ロード信号は独立して動作する。

カウンター モードを有効にするために、UDB CFG22 レジスタのSC_OUT_CTL[1:0]ビットをカウンター出力にセットする必要があります。このモードでは制御レジスタの通常動作はできません。ステータス レジスタを読み出すことができますが、マスク レジスタがカウンター周期レジスタとして再利用されるため、ステータス レジスタを割込み生成に使用しないでください。周期レジスタはデータが保持されるレジスタとして実装され、スリープから復帰してもその状態を保持します。N クロックの周期に、N–1 の周期値をロードする必要があります。N = 1 (0 周期) はクロック分周値として使用されず、ターミナル カウントの出力は 1 の定数になります。同期モードの使用は動的制御入力 (LD ／ EN) が使用されるかどうかに依存します。それらの信号が使用されない場合、同期モードは影響を受けない状態です。それらの信号が使用されると、同期モードは無効になります。

図 16-37. カウンター モード

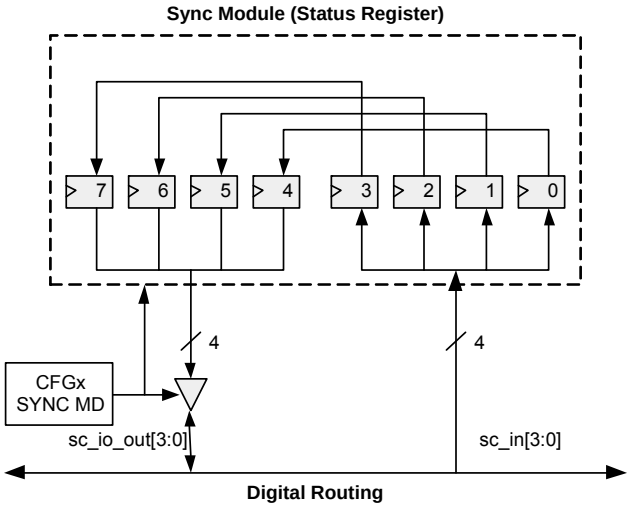


16.2.3.5 同期モード

図 16-38 に示すように、UDB CFG22 レジスタの SYNC MD ビットがセットされると、ステータス レジスタは現在の SC_CLK によりクロック供給される 4 ビット ダブル シンクロナイザーとして動作できます。このモードは GPIO 入力など非同期信号のローカル同期実装のためにも使用できます。有効の場合、同期化される信号は SC_IN[3:0] から選択され、その出力は SC_IO_OUT[3:0] ピンに駆動されて、SYNC MD は自動的に SC_IO ピンを出力モードに切り替えます。このモードではステータス レジスタの通常動作はできません。制御設定にかかわらず、ステータス スティッキー ビット モードは強制的にオフにされます。制御レジスタはこのモードから影響を受けません。カウンターは制限付きで使用され

ます。このモードにはカウンターへの動的入力 (LD ／ EN) が許可されません。

図 16-38. 同期モード



16.2.3.6 ステータスおよび制御のクロック供給

ステータスおよび制御レジスタは次のいずれか 1 つの動作モードのためにクロックが必要になります。

- 「スティッキー、読み出すとクリア」モードのステータス レジスタ
- カウンター モードの制御レジスタ
- 同期モード

この場合のクロックはリセットおよびクロック制御モジュールに配置されます。164 ページのリセットおよびクロック制御モジュールを参照してください。

16.2.3.7 補助制御レジスタ

読み出し／書き込み補助制御レジスタは UDB の固定機能のハードウェアを制御する特別なレジスタです。このレジスタにより、CPU は割り込み、FIFO およびカウンター動作を動的に制御できます。レジスタ ビットとその説明は以下のとおりです。

補助制御レジスタ							
7	6	5	4	3	2	1	0
		CNT START	INT EN	FIFO1 LVL	FIFO0 LVL	FIFO1 CLR	FIFO0 CLR

FIFO0 クリア、FIFO1 クリア

FIFO0 CLR と FIFO1 CLR ビットは関連する FIFO の状態をリセットするために使用されます。これらのビットに「1」の値を書き込むと、対応する FIFO の状態はリセットされます。FIFO 動作が継続するためにはこれらのビットを再度「0」に書き込む必要があります。これらのビットをアサートのままにすると、FIFO はステータスがなく単なる 1 バイト バッファとして動作します。

FIFO0 レベル、FIFO1 レベル

4 バイト FIFO はバスステータスをアサートします。FIFO0 LVL と FIFO1 LVL ビットは FIFO がアサートするレベルを設定します。表 16-20 に示す通り、FIFO バスステータスの意味は入出力により異なります。

表 16-20. FIFO レベル制御ビット

FIFOx LVL	入力モード (バスから FIFO にデータ書き込み)	出力モード (FIFO からバスにデータ読み出し)
0	満杯 (FULL) でない 書き込み可能なビットが最低限 1 バイトある	空 (EMPTY) でない 読み出し可能なビットが最低限 1 バイトある
1	最低限半分が空 (EMPTY) 書き込み可能なビットが最低限 2 バイトある	最低限半分が満杯 (FULL) 読み出し可能なビットが最低限 2 バイトある

割り込みイネーブル

ステータス レジスタの割り込み生成回路が有効の場合、INT EN ビットは結果の割り込み信号をゲートで制御します。

カウント開始

CNT START ビットはカウンターを有効化／無効化するために使用されます (SC_OUT_CTL[1:0] ビットがカウンター出力モードに設定された場合にのみ有効) 。

16.2.3.8 ステータスおよび制御レジスタのまとめ

表 16-21 はステータスおよび制御レジスタの機能をまとめています。制御とマスク レジスタはカウントと周期レジスタと共有され、これらのレジスタの意味もモードに依存することに注意してください。

表 16-21. ステータス／制御レジスタ機能のまとめ

モード	制御 / カウント	ステータス／同期	マスク／周期
制御	制御出力	ステータス入力または同期	ステータス マスク
カウント	カウント出力		カウント周期
ステータス	制御出力またはカウント出力	ステータス入力	ステータス マスク
同期		同期	該当なし[b]

a. カウンター モードでは、マスク レジスタは周期レジスタとして動作して、マスク レジスタとして機能できないことに注意してください。そのため、カウンター モードが有効の時に出力割り込みを使用できません。

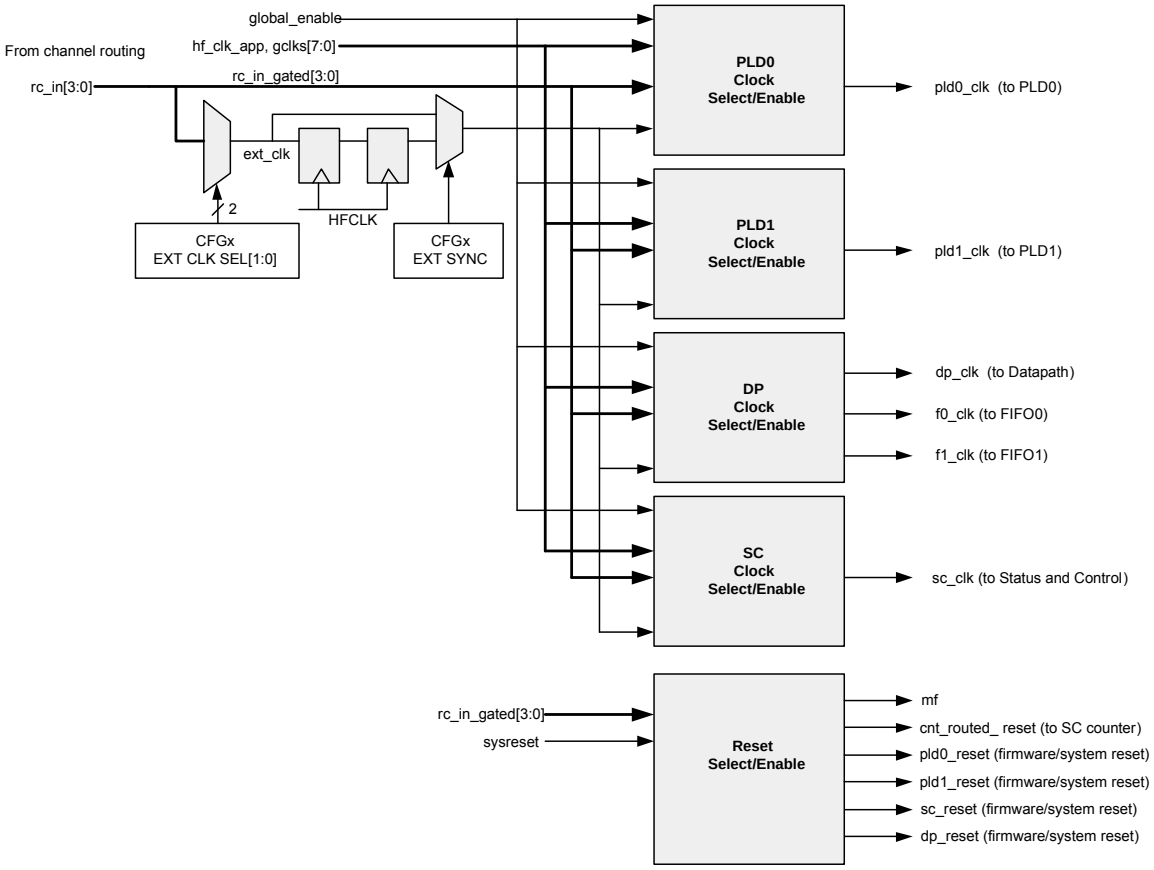
b. 同期モードでは、ステータス レジスタの機能は無効のため、マスク レジスタは使用不可になります。ただしカウント モードのために周期レジスタとして動作できます。

16.2.4 リセットおよびクロック制御モジュール

リセットおよびクロック ブロックの主な機能は各 PLD、データパス、ステータスおよび制御ブロックに、可能なグローバル システム クロックや HFCLK からクロックを選択することです。また UDB ブロックに動的かつファームウェアに基づいたリセットを提供します。図 16-39 に示すように、クロック制御ブロックは 4 個あり、リセット ブロックは 1 個あります。使用可能な入力は 4 個で配線マトリクスから提供されます (RC_IN[3:0]) 。クロック制御ブロックはこれらの配線入力から有効なクロック源を選択できます。また配線入力の中から 1 個を外部クロックの供給源として選択するマルチプレクサも装備します。以下の図のように、外部クロック供給源の選択は任意に同期化することができます。それぞれの UDB コンポーネントが選択可能なクロック数は合計で 6 個です。それは 4 個の UDB ペリフェラル クロック、HFCLK、と選択した外部クロック (ext clk) 。配線入力信号 (rc_in) はすべてレベル センシティブかまたはエッジ センシティブ有効として使用可能です。このブロックのリセット機能として、PLD ブロックと SC カウンターに配線リセットを提供し、再コンフィギュレーションのために各ブロックにファームウェア リセット機能をサポートします。

リセットおよびクロック制御用の HFCLK 入力はシステム HFCLK に独立します。このクロックは UDB ペリフェラル クロックと同様にゲートで制御され、UDB アプリケーションに使用されるため「hf_clk_app」と呼ばれます。システム HFCLK は I/O アクセスにのみ使用され、アクセスごとに自動的にゲートで制御されます。データパス クロック ジェネレーターは 3 つのクロックを生成します : データパス全般に 1 つのクロックおよびそれぞれの FIFO に 1 つのクロックです。

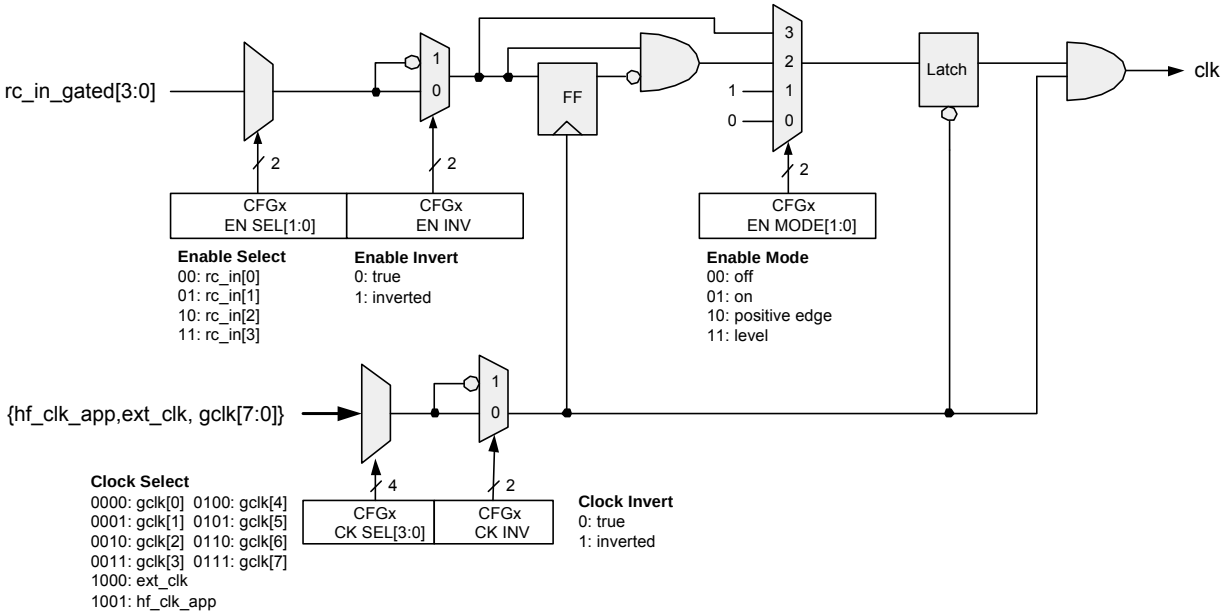
図 16-39. リセットおよびクロック制御



16.2.4.1 クロック制御

図 16-40 はクロック選択とクロック有効化回路の例を説明します。各 UDB は 4 個の回路があります : 各 PLD ブロックにそれぞれ 1 個、データパスに 1 個、ステータスおよび制御ブロックに 1 個です。この回路の主なコンポーネントはグローバル クロック選択マルチプレクサ、クロック反転、クロック有効化、クロック有効化選択マルチプレクサ、クロック有効化反転、とエッジ検出ロジックからなります。

図 16-40. クロック選択／有効化制御



クロック選択

gclk[0] ～ gclk[3] の 4 個の UDB ペリフェラル クロック (67 ページのクロック供給システムを参照してください) はすべての UDB に接続されます。残り 4 個のクロック コンフィギュレーション (gclk[4] ～ gclk[7]) は PSoC 4 ファミリのデバイスでは未対応です。これらのクロックはすべて選択不可になります。UDB ペリフェラル クロックはユーザ選択可能なクロック分周器の出力です。他の選択肢は HFCLK で、システム全体の最高周波数で動作します。「hf_clk_app」という信号はシステムの HFCLK と独立して配線されます。その他、SPI などの直接クロック供給機能に対応するために外部配線された信号もクロック入力として選択可能です。アプリケーション機能は UDB 全体の任意の境界に配置されたため、それぞれの UDB サブコンポーネント ブロックに個別のクロックを選択することは高精度のプログラムを支援します。

クロック反転

選択したクロックは任意に反転できます。サイクル タイミング パスの半分が存在するため、動作の最大周波数は制限されます。内部クロックが反転されその周波数が HFCLK と同じ時に、同時バス書き込みと内部書き込み (例えば、カウンターが動作中に新しいカウント値を書き込むこと) はサポートされません。この制限は A0、A1、D0、D1 およびカウンター モードでの制御レジスタに影響を与えます。

クロック イネーブルの選択

クロック イネーブル信号は任意の同期信号に接続され、配線マトリクスからこのブロックへの可能な 4 個の入力のいずれか 1 個から選択されます。

クロック イネーブルの反転

クロック イネーブル信号は任意に反転できます。この特長によりクロック イネーブル信号はどの極性にも生成されます。

クロック イネーブル モード

デフォルトでは、クロック イネーブル信号がオフの状態です。目標ブロックの動作を設定した後、図 16-40 に示すように UDB CFG24 レジスタの EN MODE[1:0] ビットを使用することで、そのモードを以下のいずれか 1 つのモードに設定できます。

表 16-22. UDB CFG24 レジスタによるクロック イネーブル モード

クロック有効化モード	説明
OFF	クロックが OFF の状態
ON	クロックが ON の状態。選択したグローバル クロックはフリー ランニング
ポジティブ エッジ	ゲートされたクロックはクロック イネーブル入力信号でポジティブ エッジが検出されると生成します。イネーブル入力信号の最大周波数は選択したグローバル クロックを 2 の分周比で分周した値になる
レベル	クロック イネーブル入力信号が HIGH (「1」の値) の間で、クロックは生成される

クロック イネーブルの使用法

クロック イネーブルの一般的な使用法は次の 2 つのシナリオです。

ファームウェア イネーブル – 殆どの機能はファームウェアによるクロック イネーブル信号を使用して機能の動作を開

始／停止すると仮定します。UDB アレイにマップされる機能の境界は任意 (多くの UDB および／または各 UDB の部分まで及ぼす) のため、関連の機能だけを有効化する対策が必要になります。このことは基本的に 1 つか複数のクロック イネーブル入力信号に接続する制御レジスタの 1 ビットにより実現されます。このシナリオは複数の無関連ブロックを同時に有効化させる必要があるアプリケーションにも対応できます。

エミュレート ローカル クロックの生成 – この機能により、あるUDBから他のUDBに直接クロック供給するよりも、同期クロック イネーブル生成方式を適用することで、ローカル クロックはUDBから生成され特定のアレイにて他のUDBに配布されます。クロック イネーブル モードのポジティブ エッジ機能を使用すると、クロック イネーブル波形のデューティ比にかかわる制限がなくなります。

特別な FIFO クロック供給

データパスFIFOには考慮すべき特別なクロック機能があります。デフォルトでは、FIFO クロックはデータパス クロックと同様のコンフィギュレーションに従います。しかしFIFOにはクロック コンフィギュレーションを変更する特別な制御ビットがあります。

- 各 FIFO クロックは、選択したデータパス クロックについて、極性を反転することができます。
- UDB CFG16レジスタにてFIFO FASTモードをセットすると、HFCLKはFIFOが通常使用しているデータパス クロックをオーバーライドします。

16.2.4.2 リセット制御

リセット制御のモードは互換モードと代替モードという 2 モードがあります。これらのモードは各 UDB CFG31 レジスタの ALT RES ビットにより制御されます。このビットが「0」の場合、互換モードが適用されます。このビットが「1」の場合、代替モードが適用されます。

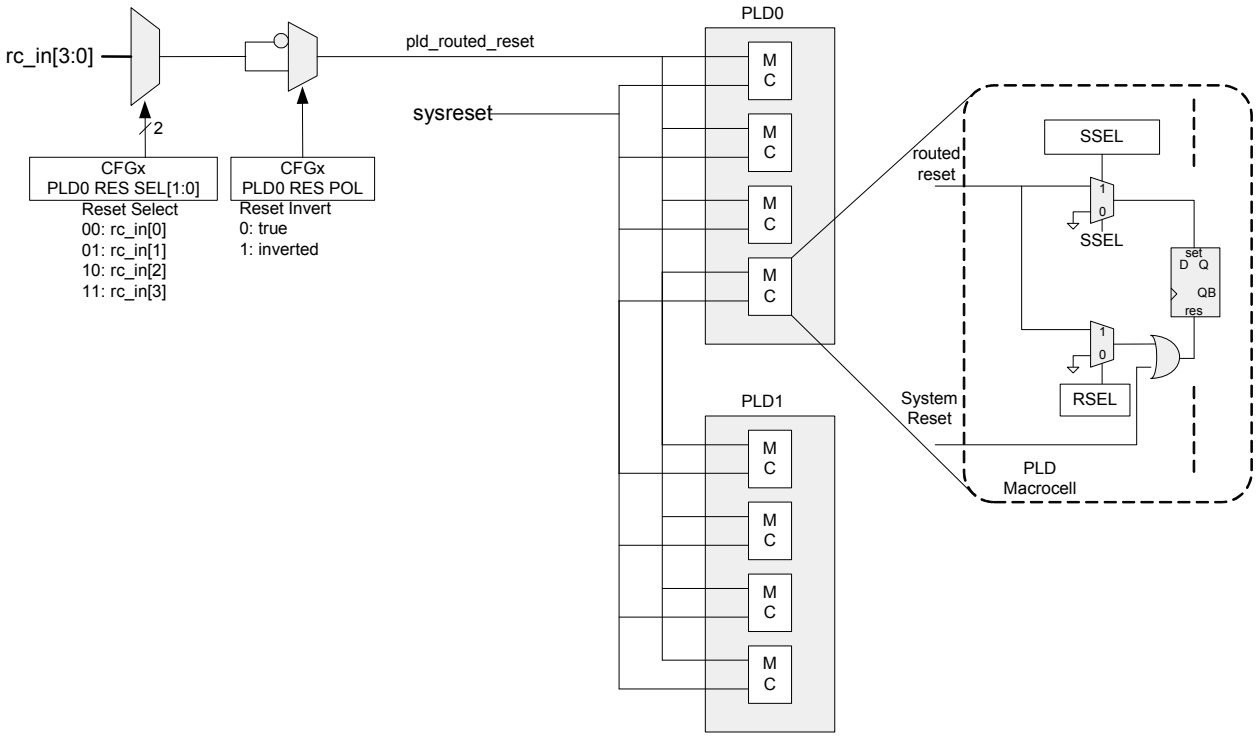
互換リセット方式

この方式はリセット配線を装備することにより、ブロックの組み込み状態を動的にリセットします。これは各 PLD マクロセルと SC カウンターに適用できます。

互換 PLD リセットの制御

図 16-41 は配線による動的リセットを使用する互換 PLD リセット システムを示します。

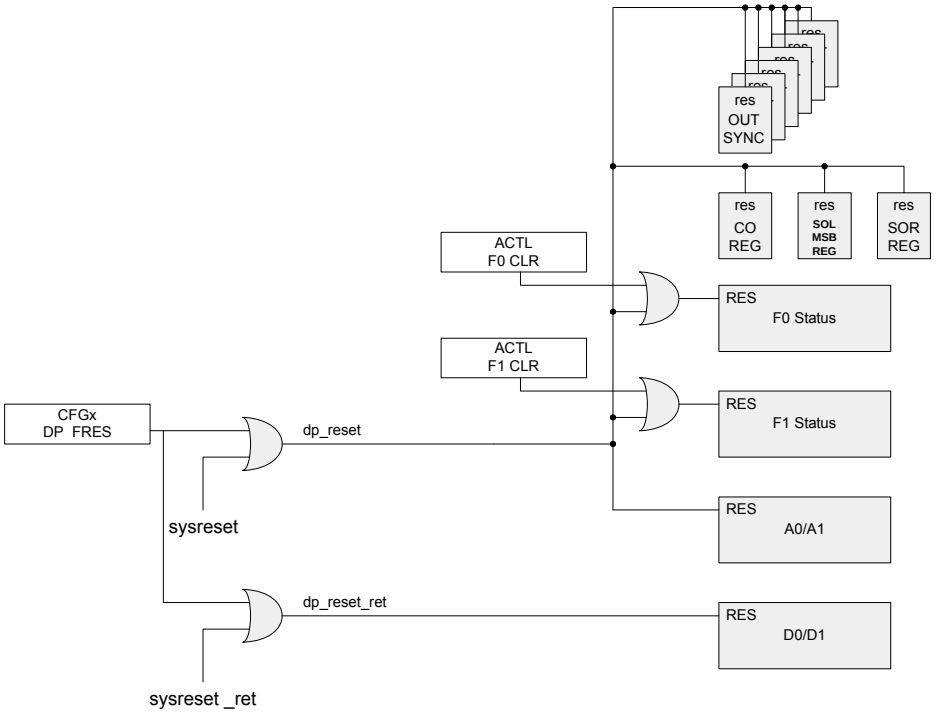
図 16-41. 互換 PLD リセットの構造



互換データパスリセットの制御

図 16-42 はファームウェアによるリセットを使用する互換データパス リセット システムを示します。ファームウェアによるリセットは DP 出力レジスタ、キャリーとシフト アウト フラグ、FIFO 状態、アキュムレータおよびデータ レジスタの値を非同期にクリアします。DO と D1 レジスタはデータが保持されるレジスタとして設定され、スリープから復帰してもその状態を保持します。FIFO データは RAM に基づいたため不定とされます。

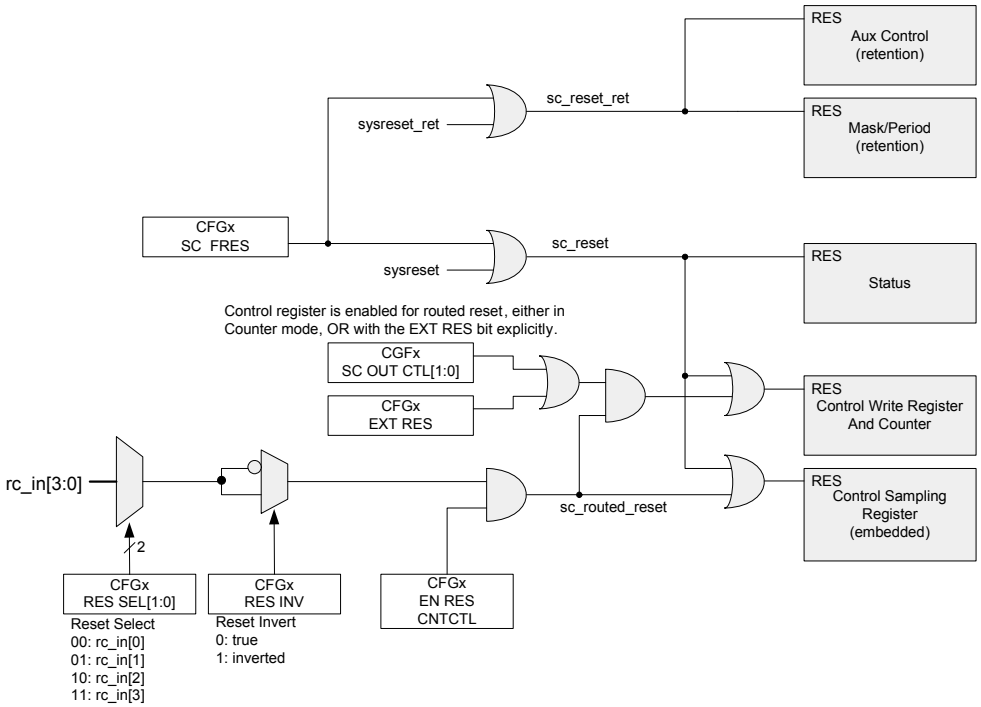
図 16-42. 互換データパスリセットの構造



互換ステータスおよび制御リセットの制御

図 16-43 は互換ステータスおよび制御ブロック リセットを示します。マスク／周期および補助制御レジスタはデータが保持されるレジスタです。

図 16-43. 互換ステータスおよび制御リセットの制御



代替リセット方式

表 16-23 は互換リセット方式と代替リセット方式との相違点をまとめます。

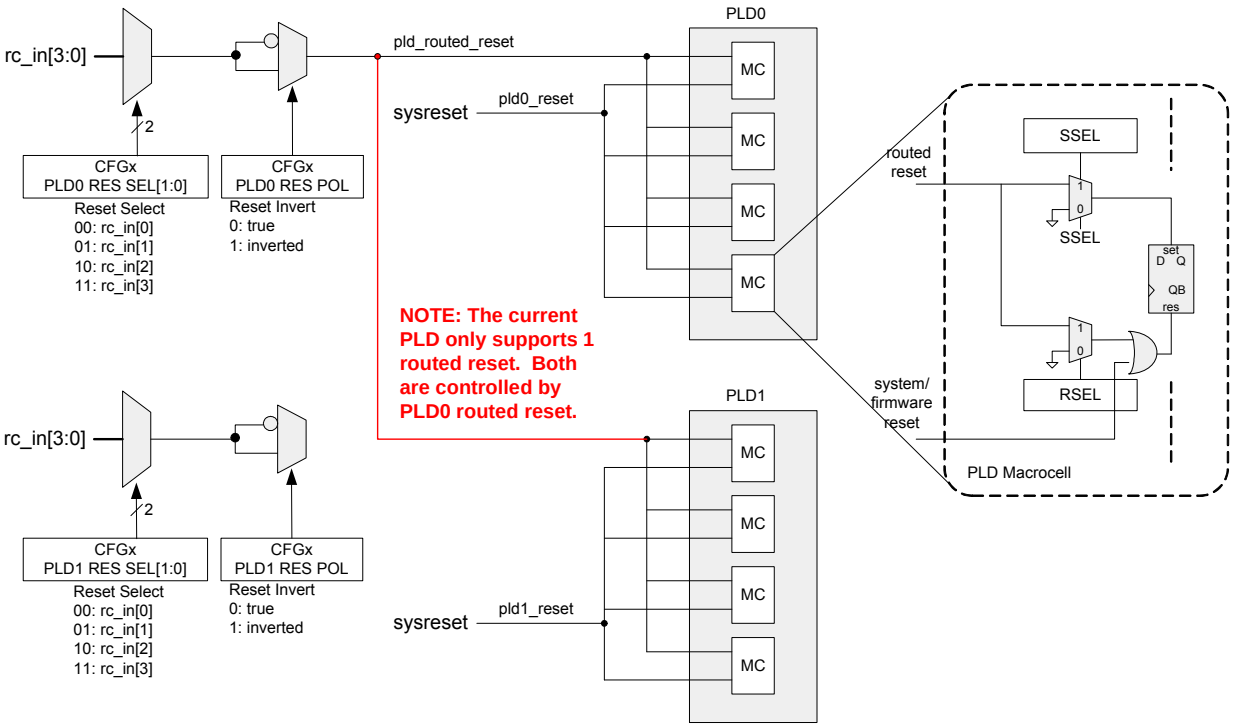
表 16-23. リセット方式

特長	互換	代替
精度	UDB 内のすべてのブロックが 1 つの配線リセットを共有する	UDB コンポーネント ブロックが個別にリセットを設定できる
ステータス レジスタ	配線リセット機能なし	選択したステータス／制御 (SC) レジスタから配線リセットをオプションとして利用可能
データパス	配線リセット機能なし	選択したデータパス (DP) から配線リセットをオプションとして利用可能

代替 PLD リセットの制御

図 16-44 は代替 PLD リセット システムを示します。これは各 PLD の個別リセットの用意がありますが、PLD ブロックには対応されません。そのため、代替リセット方式では、PLD0 リセット制御の設定は両方の PLD に適応します。

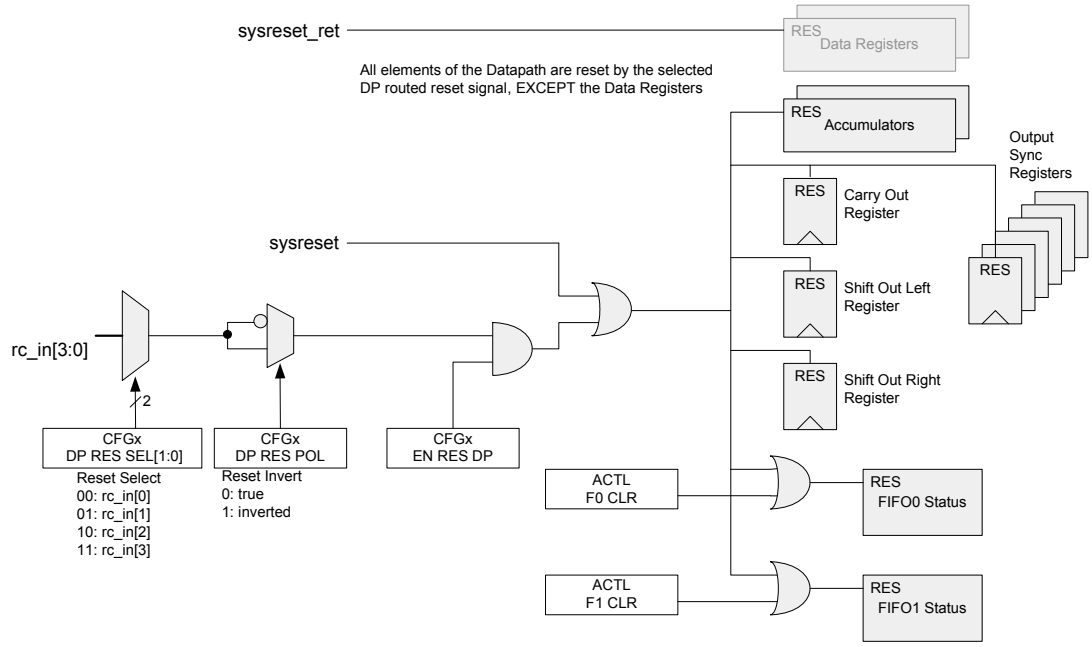
図 16-44. 代替 PLD リセット構造



代替データパス リセットの制御

図 16-45 は代替データパス リセット システムを示します。データパスの配線リセットは、データが保持されるレジスタとして設定されたデータ レジスタを除いてデータパスのすべての形態に適用します。

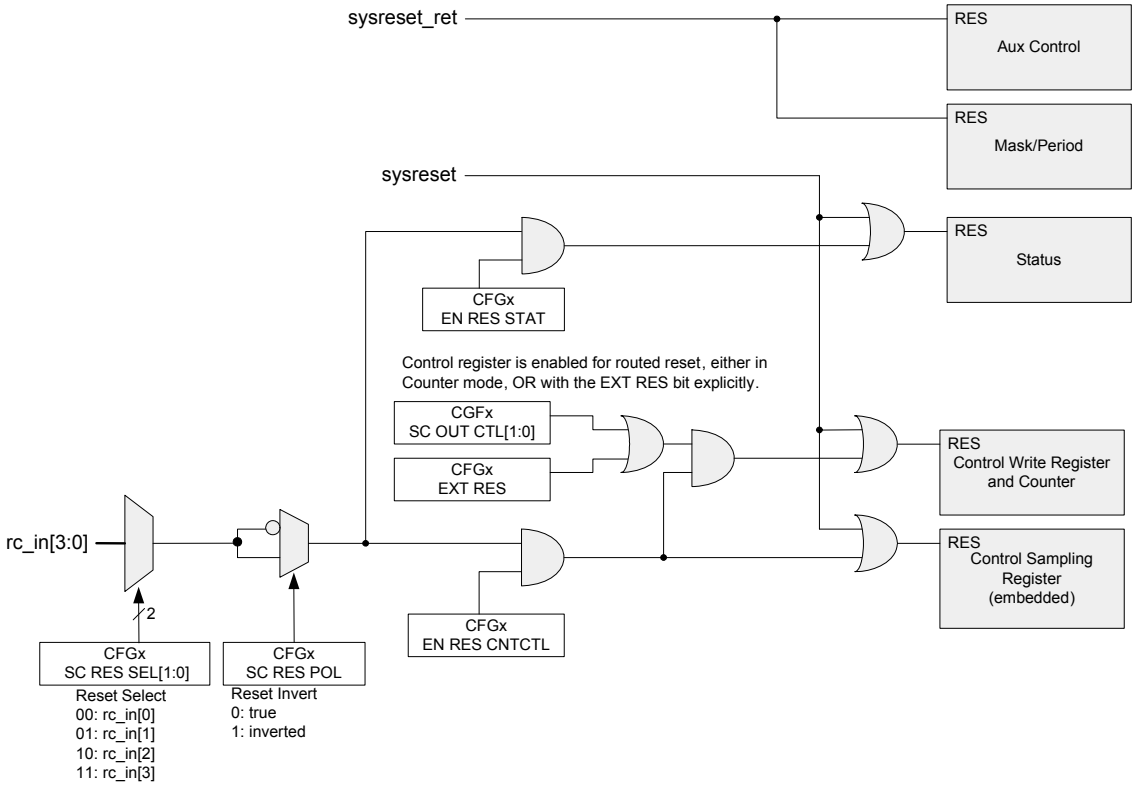
図 16-45. 代替データパス リセットの構造



代替ステータスおよび制御リセットの制御

図 16-46 は代替ステータスおよび制御ブロック リセットを示します。マスク／周期および補助制御レジスタはデータが保持されるレジスタです。

図 16-46. 代替ステータスおよび制御リセットの制御



16.2.4.3 UDB POR 時の初期化

レジスタと状態の初期化

表 16-24. UDB POR 状態の初期化

ステートの要素	ステートの要素	POR 時の状態
コンフィギュレーション ラッチ	CFG 0 ～ 31	0
Ax、Dx、CTL、ACTL、MASK	アキュムレータ、データ レジスタ、補助制御レジスタ、マスク レジスタ	0
ST、マクロセル	ステータスおよびマクロセル読み出し専用レジスタ	0
DP CFG RAM と Fx (FIFO)	データパス コンフィギュレーション RAM と FIFO RAM	不定
PLD RAM	PLD コンフィギュレーション RAM	不定

配線の初期化

POR 時に、入力と出力配線の状態は次の通りになります。

- UDB から配線マトリクスに駆動される出力はすべて「0」に保持します。
- 配線から出て UDB 入力へのドライバーはすべて最初に「0」にゲート制御されます。

この初期化の結果として、配線上の駆動状態の衝突を回避でき、初期構成は順番に依存しないシーケンスに従って発生します。

16.2.5 UDB アドレス指定

UDB は (A0、A1、D0、D1、FIFO など) ワークレジスタとコンフィギュレーション レジスタともに 8、16、32 ビット アクセスで多くのアドレス空間を通してアクセスされることができます。

- 8 ビット ワークレジスタ – このアドレス空間で単一 UDB 内のワークレジスタが個別にアクセスされます。
- 16 ビットの連続したワークレジスタ – このアドレス空間で 2 個の連続した UDB 内の同じワークレジスタ (例えば、UDBn の D0 と UDBn+1 の D0) へのアクセスが可能になります。
- 16 ビット 対のワークレジスタ – このアドレス空間で同じ UDB 内の 2 個のワークレジスタ (例えば、A0 と A1) へのアクセスが可能になります。
- 32 ビット ワークレジスタ – このアドレス空間で 4 個のあらゆる UDB 内の同じワークレジスタ (例えば、A1) へのアクセスが可能になります。
- 8、16または32ビット コンフィギュレーション レジスタ – このアドレス空間で単一 UDB 内のコンフィギュレーション レジスタへのアクセスが可能になります。

16.2.6 システム バス アクセスの一貫性

UDB のレジスタはデュアル アクセス モードがあります。

- システム バス アクセス、CPU は UDB レジスタに対して書き込み／読み出しを行っています。
- UDB 内部アクセス、UDB 機能がレジスタの内容を更新／使用しています。

16.2.6.1 システム バスの同時アクセス

表 16-25 は可能な同時アクセス イベントと必要な動作を示します。

表 16-25. システム バスの同時アクセス

レジスタ	UDB 書き込み バス書き込み	UDB 書き込み バス読み出し	UDB 読み出し バス書き込み	UDB 読み出し バス読み出し
Ax	未定義結果	直接は不可[a,b]	UDB は直前の値を読み出す	現時点の値は両方に読み出される
Dx				
Fx	未対応 (UDB とバスは対立するアクセスであることが必要)	FIFO のステータス フラグが使用されると、同じ位置の同時読み出し／書き込みは実行不可		未対応 (UDB とバスは対立するアクセスであることが必要)
ST	未対応、バスは書き込みしない	以前の値がバスに読み出される	未対応、UDB は読み出しをしない	
CTL	未対応、UDB は書き込みしない		UDB は直前の値を読み出す	現時点の値は両方に読み出される
CNT	未定義結果	直接は不可[c]		
ACTL	未対応、UDB は書き込みしない			
MASK				
PER				
マクロセル (RO)	未対応、バスは書き込みしない	直接は不可[d]	未対応、バスは書き込みしない	

a. Ax レジスタは、FIFO へのソフトウェア取り込み機能を利用することで無事に読み出されます。
b. Dx レジスタは FIFO からの動的な書き込みだけが可能です。このモードがプログラムされると、Dx レジスタの直接読み出しは許可されません。
c. CNT レジスタが無効の時にのみその読み出しは無事に実行できます。CNT 値を動的に読み出す代替の方法は、出力を (トランスペアレント モードでの) SC レジスタに配線することです。
d. マクロセル レジスタ ビットは確実にデータを読み出すために、(トランスペアレント モードでの) ステータス レジスタ入力にも配線される場合があります。

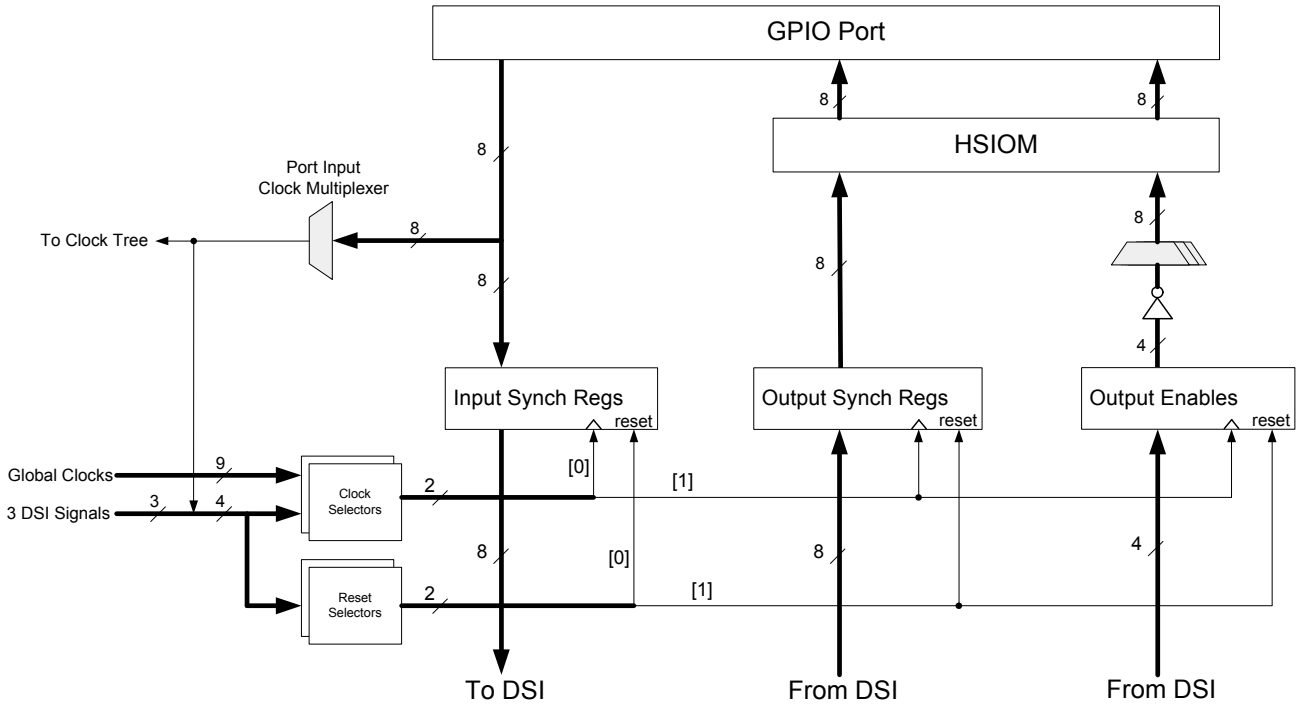
16.2.6.2 アキュムレータの一貫性アクセス (アトミックな読み出しと書き込み)

UDB のアキュムレータはデータ計算の最初の対象です。そのため、表 16-25 に示すように通常動作中にこれらのレジスタを直接読み出すと、未定義結果を出します。ただし、アトミック読み出しに対応するためにソフトウェア取り込みの形式で内蔵のサポートを備え、連結されたブロックに実現されます。この使用法では、最下位アキュムレータの読み出しにより、連結されたあらゆるブロックからのデータを該当する FIFO に転送します。アキュムレータへのアトミックな書き込みはプログラムにより実現されます。個々の書き込みは FIFO 入力に行われ、FIFO の「FIFO の最後に書き込まれた」ステータス信号が関連するブロックに送信される一方、その FIFO データを Dx か Ax レジスタに同時に転送します。

16.3 ポート アダプタ ブロック

63 ページの高速 I/O マトリックスに示すように、ポート アダプタ ブロックは UDB を拡張して高速 I/O マトリクス (HSIOM) を通して GPIO とのインターフェースを提供します。HSIOM はレジスタを配置して DSI 信号の GPIO 出力と出力イネーブルへの配線をより早くさせます。また、HSIOM により GPIO が多くのブロック (例えばポート データ レジスタと I2C などのペリフェラル) により共有されることができます。図 16-47 はその概要図を示します。

図 16-47. ポート アダプタ ブロック図



各 8 ビットの GPIO ポートは 1 個のポート アダプタ (PA) を持っています。GPIO データ入力から 8 個の入力、GPIO データ出力から 8 個の出力および 8 個の出力イネーブル (OE) 接続信号があります。PA 内のレジスタは入力、出力、出力イネーブル信号の同期化に使用されます。

他の特長はポート入力クロック マルチプレクサです。このマルチプレクサは 1 つのポート入力をクロックとして選択します。このクロックは PA 内のローカルで使用され、グローバル クロックに配線されます (67 ページのクロック供給システムをご覧ください)。

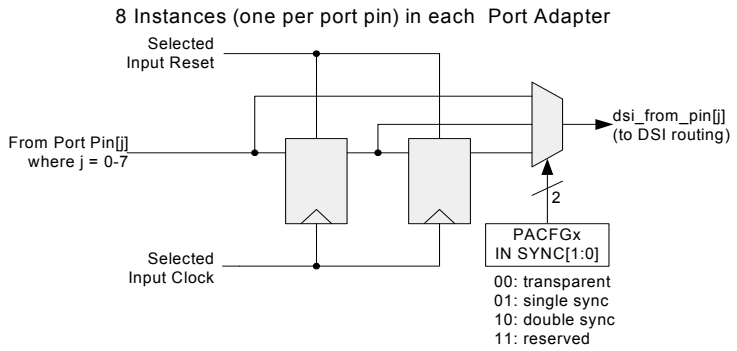
入力と出力同期レジスタに個別のクロックを提供するために 2 個のプログラム可能なクロック セレクタが用意されます。OE レジスタは出力レジスタと同じクロックを使用します。

また、2 個のプログラム可能なリセット セレクタもあり、クロック セレクタの場合と同様の方法で提供されます。

16.3.1 PA データ入力ロジック

図 16-48 はデータ入力ロジックの構造を示します。入力信号は I/O ポートの各ピンからです。この信号はシングル同期化されるかダブル同期化されます。非同期入力には同期回路がバイパスされます。同期は選択したポート入力クロックに同期します。この回路の出力は DSI 配線に接続します。

図 16-48. GPIO 入力ロジックの詳細



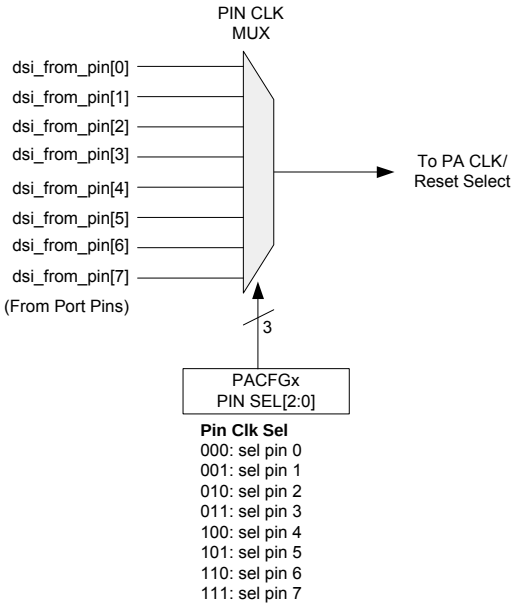
16.3.2 PA ポート ピン クロック マルチプレクサ ロジック

図 16-49 はポート ピン マルチプレクサを示します。各ポートは 8 個のデータ入力信号、その中 1 個はクロックとして選択されます。この選択は次の目的に配線されます。

- ポート アダプタのプログラマブルなクロック
- UDB クロック ツリーの供給源
- ポート アダプタのプログラマブルなリセット
- ポート アダプタのクロック イネーブル信号

選択した信号は同期回路を通過せず、ブロック内の他のクロック ドメインに非同期であることに注意してください。選択した機能に注意して使用する必要があります。

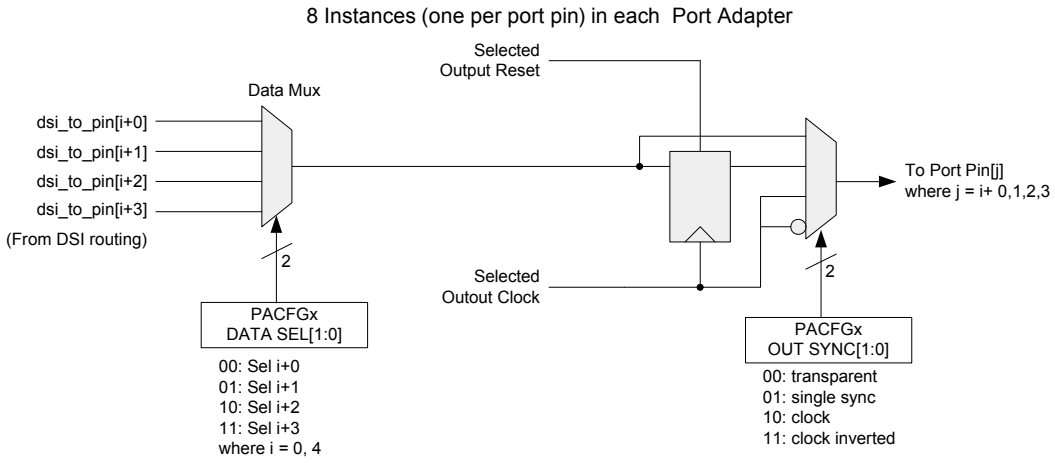
図 16-49. GPIO ピン選択の詳細



16.3.3 PA 出力ロジック

図 16-50 はデータ出力ロジックの構造を示します。出力信号は (HSIOM を通して)I/O ポートの各ピンに送信されます。この信号はシングル同期化されるか、非同期出力には同期がバイパスされます。他のオプションは、選択したクロックかクロックの反転したもののいずれか 1 つを出力する能力を含みます。

図 16-50. GPIO 出力データ ロジックの詳細



16.3.4 PA 出力イネーブル ロジック

図 16-51 は出力イネーブル (OE) ロジックを示します。この回路はデータ出力に関連するクロックとリセット信号を共有します。図 16-52 に示すように、OE に関連する DSI 出力は 4 個ありますが、すべては I/O ポート ピンへの合計 4 個の OE 接続を含む 1 つの信号に多重化される点で、この接続は独特です。

図 16-51. GPIO 出力イネーブル (OE) 同期ロジック

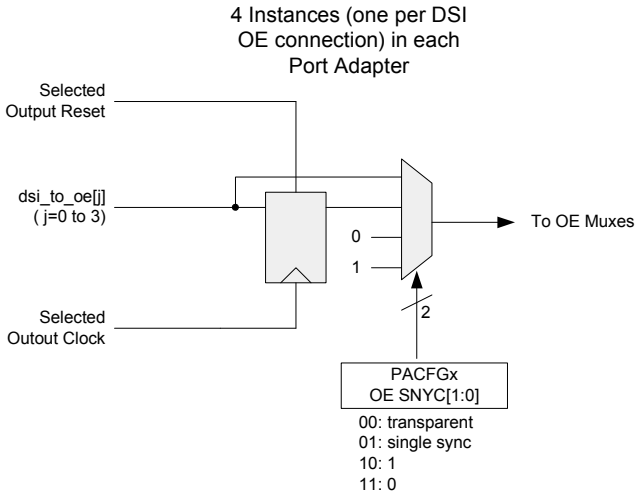
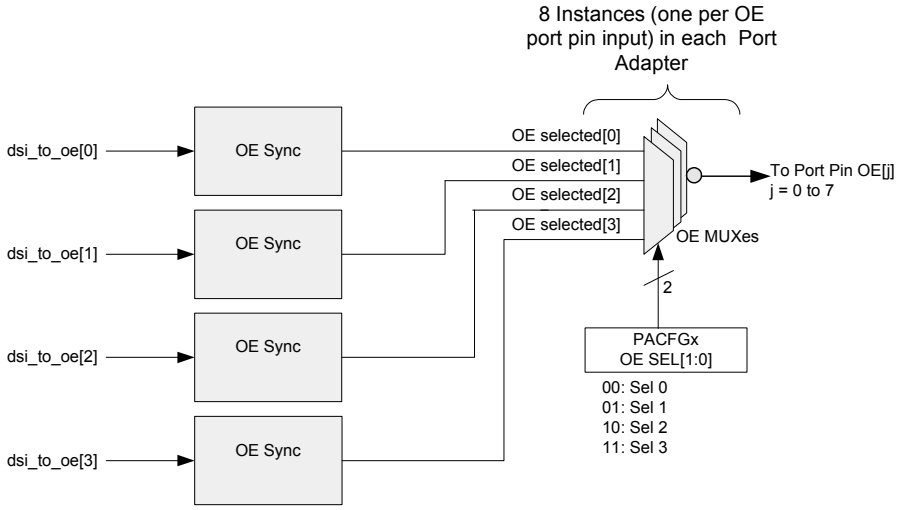


図 16-52. GPIO 出力イネーブル (OE) マルチプレクサ

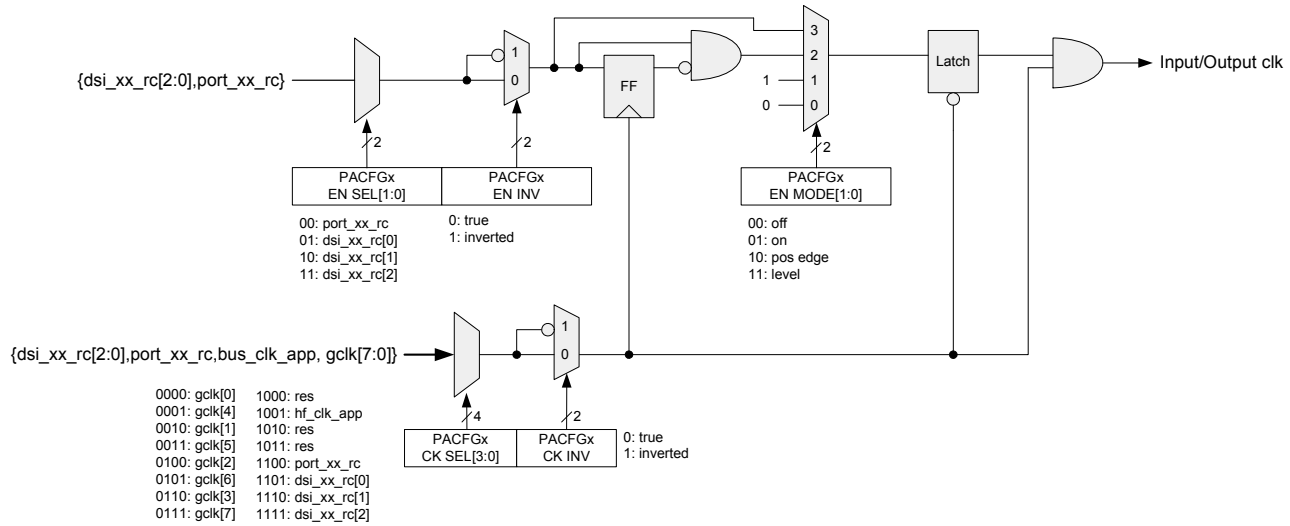


OE 信号がポートにて LOW アクティブのため、OE 同期ロジックと OE マルチプレクサとの間で追加の反転パスがあります。

16.3.5 PA クロック マルチプレクサ

図 16-53 は PA クロック マルチプレクサの構造を示します。前述の通りに、各 PA は 2 個のプログラム可能なクロック セレクタを備え、ポート入力と出力および出力 イネーブル (OE) のために個別のクロックを提供します。

図 16-53. PA クロック マルチプレクサ



16.3.6 PA リセット マルチプレクサ

PA リセット マルチプレクサの構造は図 16-54 に示します。

図 16-54. PA リセット マルチプレクサの詳細

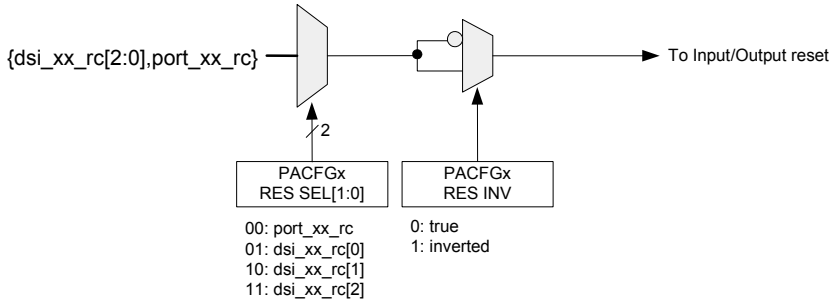
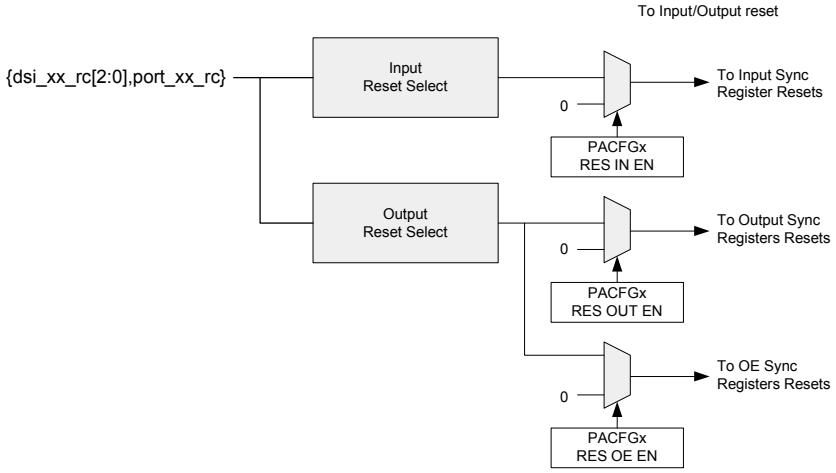


図 16-55 に示すように、リセット選択ロジックは重複され、1 個は入力のためで、残り 1 個は両方の出力と出力イネーブルのためです。これらのリセットはそれぞれ 1 個のイネーブル信号があり、関連する区分の全ての 8 ビットに使用します。

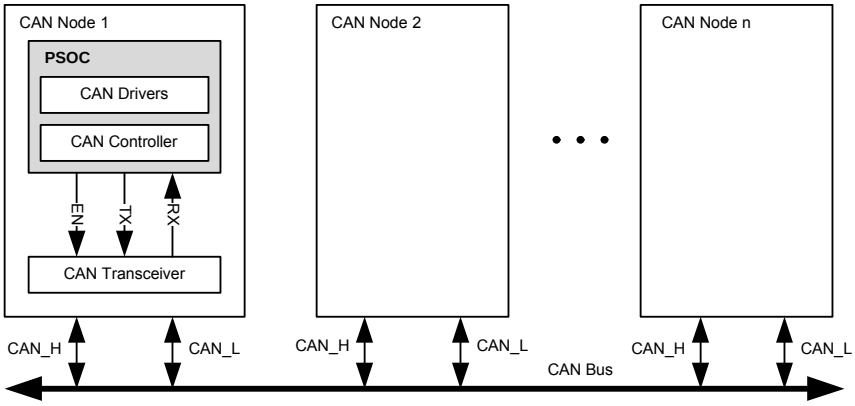
図 16-55. PA リセット システム



17. CAN

CAN ペリフェラルは、最大 1Mbps の通信ボーレートに対応できるフル機能のコントローラー エリア ネットワーク (CAN) です。PSoC 4200M デバイス ファミリは 2 個の同一の CAN コントローラー ブロックを内蔵しており、異ったピンに接続することができます。これらの CAN コントローラーは ISO-11898 規格による CAN2.0A と CAN2.0B に準拠しています。CAN プロトコルはもともと高位のエラーの検出と回復に重点を置いた車載アプリケーション向けに設計されたものです。これによって信頼性の高い通信が低コストで実現されます。CAN は車載アプリケーションで成功したため、動き重視の組み込み制御ネットワーク (CANOpen) およびファクトリ オート メーション用途 (DeviceNet) 向けの標準通信プロトコルとして採用されています。CAN の機能によってマイクロコントローラーの性能に影響を与えずに高位のプロトコルを効率的に実現できます。

図 17-1. CAN バス システムの実装



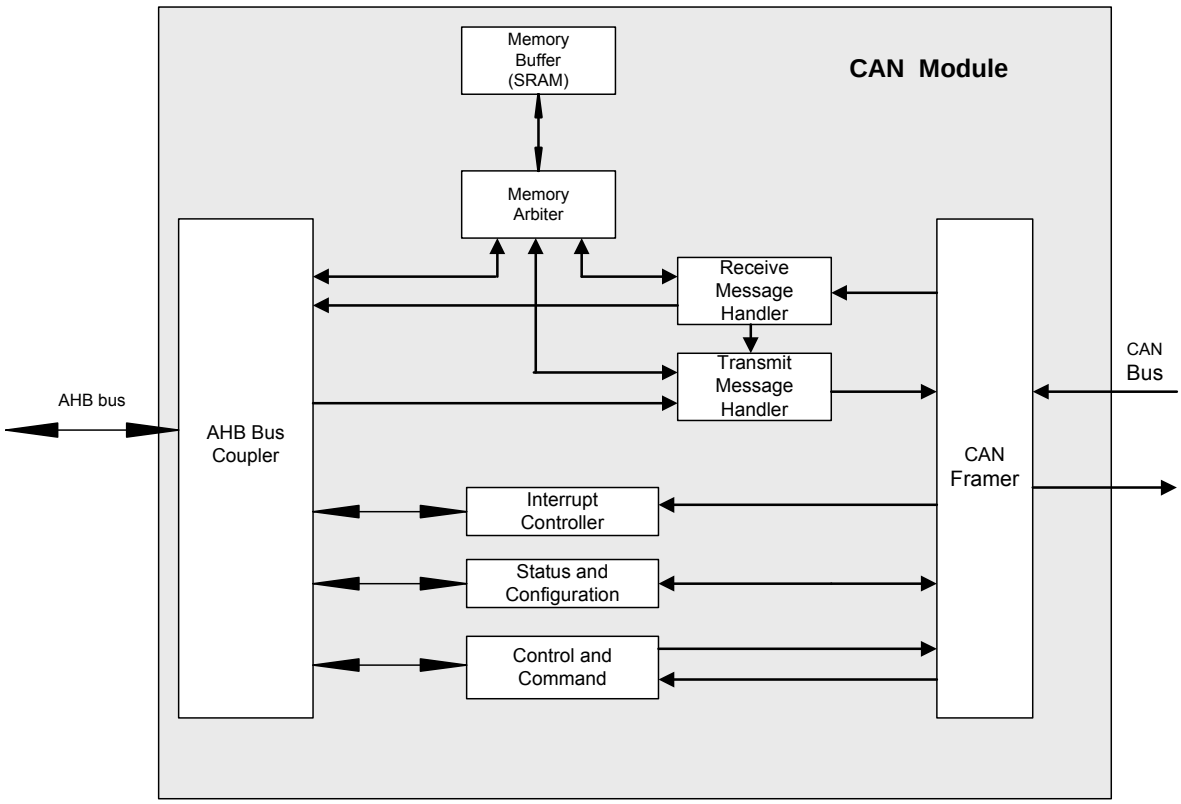
17.1 特長

- CAN2.0A/B プロトコル仕様に準拠 :
  - 標準と拡張フレーム
  - リモート送信要求 (RTR) をサポート
  - 最大 1Mbps のプログラム可能なビット レート
- 受信パス :
  - 16 個の受信メッセージ バッファ
  - 16 個のアクセプタンス フィルターとアクセプタンス マスク
  - DeviceNet のアドレス指定をサポート
  - ハードウェア FIFO を形成するために複数の受信バッファを連結するオプション
- 送信パス :
  - 8 個の送信メッセージ バッファ
  - 各送信メッセージ バッファ用のプログラム可能な優先順位
  - メッセージのシングル ショット送信をサポート
- 自動ボーレート検出のリッスンオンリー モード
- ブロック レベル テストのための内部と外部ループバック モード
- バス動作の発生時にデバイスをスリープ モードから復帰させる機能
- 時間起動 CAN を実装するためのカウンター

17.2 ブロック図

メッセージを送信するには、ホスト コントローラーはメッセージを送信メッセージ バッファに格納し、メッセージを送信する送信メッセージ ハンドラーに通知します。メッセージは受信されるとメモリ バッファに格納され、ホスト コントローラーは要求に応じてそれを処理します。送信と受信は主にステータス レジスタとコンフィギュレーション レジスタで制御されます。割り込みコントローラー ユニットは CAN モジュールの様々な割り込みを処理します。図 17-2 にこのプロセスを示します。

図 17-2. CAN ブロック図



17.3 CAN メッセージ フレーム

CAN では 4 種類の主なフレームでメッセージの送受信を制御します。

- データ フレーム
- リモート フレーム
- エラー フレーム
- オーバーロード フレーム

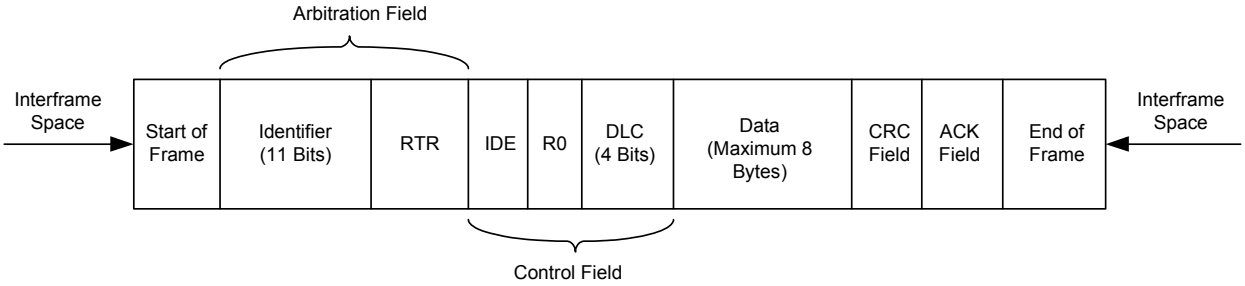
17.3.1 データ フレーム

データ フレームは主にトランスミッターとレシーバー間のデータ転送に用いられます。CAN は 2 種類の主なデータ フレームに対応しています : 標準データ フレームと拡張データ フレーム。CAN フレームでは「0」はドミナント ビット、「1」はリセッシブ ビットと呼ばれます。

17.3.1.1 標準データ フレーム

図 17-3 に CAN の標準データ フレームを示します。

図 17-3. 標準データ フレーム



フレームの開始： データ フレームの始まりはフレームの開始ビットで示されます。これは単一のドミナント ビットです。

識別子： 基本的な CAN データ フレームでは識別子の長さは 11 ビットです。これは主に受信側でデータをフィルターするために用いられます。

リモート送信要求ビット (RTR): RTR ビットは、データ フレームには「0」(ドミナント)、リモート フレームには「1」(リセッシブ ビット) にセットします。識別子と RTR ビットはアービトレーション フィールドと呼ばれます。

拡張識別子ビット (IDE): このビットは CAN の標準データ フレームには「0」(ドミナント)、拡張データ フレームには「1」(リセッシブ) にセットする必要があります。

R0: 予約済みビット。

データ長コード (DLC): これらの 4 ビットはデータ フィールドのデータ バイト数を示します。IDE、R0 および DLC ビットは制御フィールドを構成します。

データ フィールド：このフィールドはメッセージ データを格納します。この長さは最大 8 バイトまで可変です。

巡回冗長検査 (CRC): フレーム検査は巡回冗長検査 (CRC) 方式で行われます。このフィールドは 15 ビット CRC コードおよびそれに続く CRC デリミターから成ります。

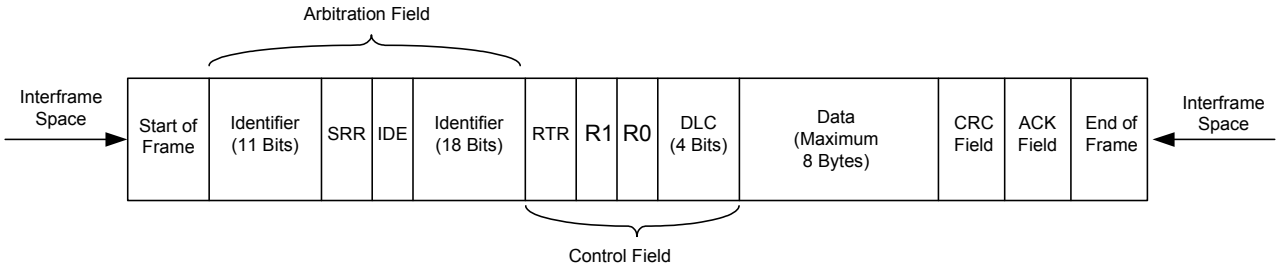
アクノリッジ フィールド (ACK): ACK フィールドは 2 ビット長で、初期設定ではリセッシブです。レシーバーはメッセージを正しく受信すると、ドミナント ビットで ACK フィールドを上書きします。

フレームの終了：各フレームの終わりはフレームの終了フィールドで示され、7 個のリセッシブ ビットから成ります。

17.3.1.2 拡張データ フレーム

拡張 CAN フレームのフォーマットは図 17-4 に示されます。拡張 CAN は 29 ビットの識別子を含みます。これは代替リモート要求 (SRR) ビットと IDE ビットによって分離される 11 ビットと 18 ビットの識別子フィールドに配置されます。SRR ビットは標準フレーム内 RTR ビットと同じ位置にあり、リセッシブです。拡張フレームでは IDE ビットがセットされます。拡張データ フレームの制御フィールドは、標準データ フレームと比べて予約済みビット「R1」が追加されます。

図 17-4. 拡張データ フレーム



17.3.2 リモート フレーム

CAN バスによりデスティネーション ノードがリモート フレームを送信することでソースからのデータを要求することができます。データ フレームとリモート フレームの相違点は 2 つあります: リモート フレームでは RTR ビットがリセッシブ ビットとして送信され、データ フィールドがありません。

拡張リモート フレームでは SRR ビットもリセッシブ ビットとして送信されます。

フレーム間のスペース：フレーム間のスペースはデータ フレームとリモート フレームを前のフレームから分離します。

17.3.3 エラー フレーム

ノードはバス エラーを検出するとエラー フレームを生成します。エラー フレームはエラー フラグとエラー デリミターから成ります。エラー フラグは 2 種に分類されています : エラー アクティブ フラグとエラー パッシブ フラグ。

エラー アクティブ フラグ： エラー アクティブ ノードはエラーを検出すると、6 個のドミナント ビットをアクティブ エラー フラグとして送信します。エラー フラグのフォーマットはビット スタッフのルールに違反します。これにより他のすべてのノードがエラー フラグを送信し、その結果バス上に一連の6～12個のドミナント ビットが発生します。

エラー パッシブ フラグ：エラー パッシブ フラグは 6 個の

リセッシブ ビットから成ります。エラー パッシブ ノードはエラーを検出すると、パッシブ エラー フラグを送信します。パッシブ エラーはその他のノードに影響せず、送信ノードがバス エラーを検出した場合にのみ検出されます。

エラー デリミター: エラー デリミターは 8 個のリセッシブ ビットから成ります。エラー フラグの送信後に、各ステーションはリセッシブ ビットを送信し、リセッシブ ビットを検出するまでバスを監視します。その後、もう 7 個のリセッシブ ビットを送信します。

17.3.4 オーバーロード フレーム

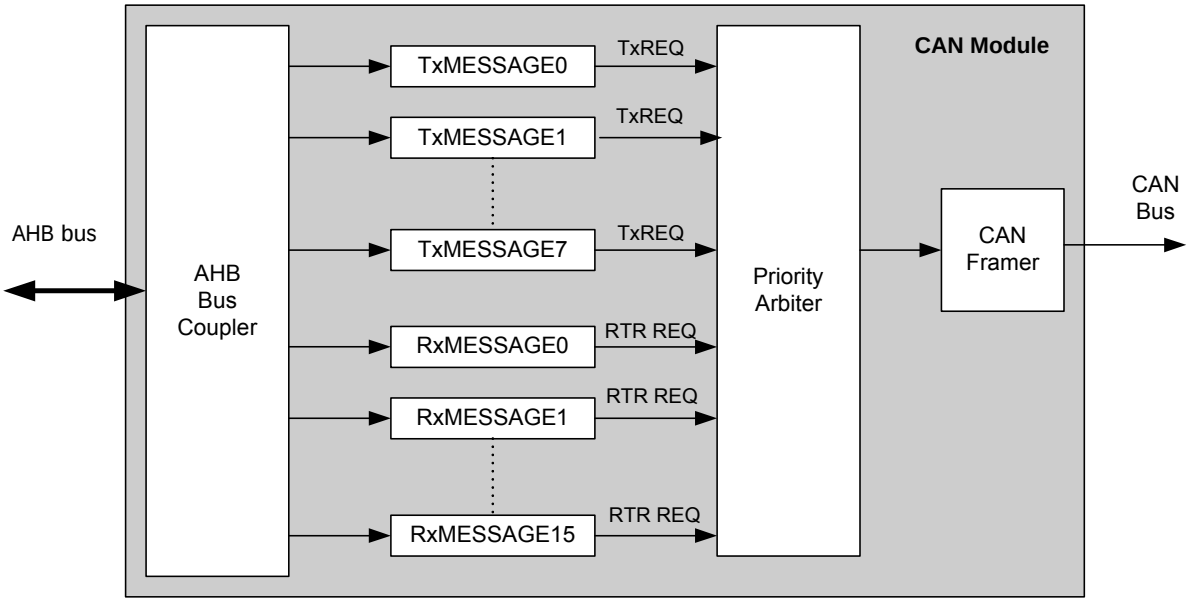
オーバーロード フレーム (EOF) はオーバーロード フラグおよびオーバーロード デリミターから成ります。PSoC 4200M CAN コントローラーは、以下の条件が発生した時にアクティブにされる、エラーに反応可能なオーバーロード フレームをサポートしています。

- 休止の最初 2 ビットの中でドミナント ビットを検出
- レシーバーは EOF の最後ビットがドミナント ビットであることを検出
- 任意のノードはエラー デリミターまたはオーバーロード デリミターの最後ビットがドミナント ビットであることを検出

17.4 CAN におけるメッセージ送信

CAN モジュールは 8 個の送信メッセージ保持バッファをサポートしています。内部優先順位アービターは、選んだアービトレーション方式に応じてメッセージを選択します。アービトレーション方式はラウンド ロビンまたは固定優先順位です。メッセージが送信される時またはメッセージ アービトレーション ロストがある時、優先順位アービターは次のメッセージの優先順位を再評価します。受信メッセージ バッファは本章の後半で説明するようにリモート送信要求を送信することもできます。

図 17-5. 送信 (Tx) ブロック図



17.4.1 メッセージ アービトレーション

優先順位アービターはラウンド ロビンと固定優先順位方式をサポートしています。アービトレーション モードはコンフィギュレーション レジスタで選択されます。

ラウンド ロビン : ラウンド ロビン方式では、最初にバッファ 0、次にバッファ 1 と続き最後にバッファ 7 を選択します。これがバッファ 0 に続いてサイクルを形成します。特定のバッファはその TX_REQ ビットがセットされた場合にのみ選択されます。この方式はすべてのバッファが同じ確率でメッセージを送信できることを保証します。

固定優先順位 : バッファ 0 は最高の優先順位を持っています。重要なメッセージが先に送信されること保証するためにバッファ 0 を重要なメッセージ用のバッファと指定します。優先順位アービトレーションはコンフィギュレーション レジスタの CFG_ARBITER ビット (CAN_CONFIG[12]) で選択されます。

注 : RTR メッセージ要求は TxMessage バッファが処理される前に処理されます。

17.4.2 メッセージ送信プロセス

図 17-6 に送信されるメッセージに関連するレジスタを示します。

図 17-6. 送信 (Tx) メッセージ レジスタ

REGISTERS											
COMMAND REGISTER (CAN_Txn_CMD)	Reserved [31:24]	WPN2 [23]	Reserved 1 [22]	RTR [21]	IDE [20]	DLC [19:16]	Reserved [15:4]	WPN1 [3]	Tx INT ENBL [2]	Tx ABORT [1]	Tx REQ [0]
IDENTIFIER (CAN_Txn_ID)	ID [31:3]									Reserved [2:0]	
DATA REGISTER High (CAN_Txn_DH)	D0 [63:56]			D1 [55:48]			D2 [47:40]			D3 [39:32]	
DATA REGISTER Low (CAN_Txn_DL)	D4 [31:24]			D5 [23:16]			D6 [15:8]			D7 [7:0]	

n = 0,1,…,7

標準データ フレームの送信手順は以下の通りです。

1. メッセージを空の送信メッセージ保持バッファに書き込みます。空のバッファは TX_REQ ビットが 0 であることで示されます。
   a. 標準データ フレームでは「0」(ドミナント) を RTR と IDE ビットに書き込みます。
   b. 転送するデータ バイト数を指定するためにDLCビットを適切に書き込みます。最大データ バイト数は 8 です。MSB (最上位ビット) が先に転送されるデータ バイトはそれぞれ D0、D1、…、D7 の位置に書き込みます。
   c. 11 ビット メッセージ識別子は ID[31:21] ビット フィールドに書き込みます。
2. 適切な優先順位アービトレーション方式を選択します。内部優先順位アービターは選んだアービトレーション方式に応じてメッセージを選択します。
3. 送信を要求するために対応する TX_REQ ビットを「1」にセットします。
4. メッセージ送信要求が保留中である限り、TX_REQ ビットはセットされたままです。TX_REQ ビットがセットされている時、メッセージ バッファの内容を変更してはいけません。

メッセージが送信された後、TX_REQ ビットはクリアされ、TX_MSG 割り込みステータス ビット (割り込みステータス レジスタ CAN_INT_STATUS の [CAN_INT_STATUS[11]) はアサートされます。割り込みステータス ビットは、TxINT ENBL (CAN_TX[n].CONTROL[2]) が「1」にセットされた時にのみアサートされます。

17.4.3 メッセージ中止

CAN_TX[n]_CONTROL レジスタの TX_ABORT ビット (CAN_TX[n]_CONTROL[1]) をセットすると、メッセージが中止されます。メッセージが中止されると、このビットは自動的にハードウェアによってクリアされます。

注 :

- 送信要求が保留中であるかにかかわらず、CAN バッファ レジスタ (CAN_BUFFER_STATUS) は読み出しに使用されます。
- 書き込み保護ビット wpn2 (CAN_TX[n].CONTROL[23]) が「0」である場合、コマンド レジスタのビット [21:16] は保護され、読み出されると未定義の値を返すため、変更不可能です。
- 書き込み保護ビット wpn1 (CAN_TX[n].CONTROL[3]) が「0」である場合、コマンド レジスタのビット [2] は変更不可能です。このビットは読み出されると「0」を返します。
- WPN フラグ (wpn1 と wpn2) を使用すると、特殊なフラグ (RTR、IDE、DLC、TxINTENBL) に注意せずに TX_REQ ビットをセットするだけで同じメッセージの再送信を簡単に行えます。

17.4.4 シングル ショット送信

シングル ショット送信モードは、アービトレーション ロストやバス エラーに起因した CAN メッセージの再送信を防止する必要のあるシステムでは使用されます。これは特にすべてのメッセージが定時に送信される時間起動 CAN システムに役立ちます。

CAN_TX.CONTROL.TX_REQ と TX_ABORT ビットを同時にアサートするとシングル ショット送信要求が発行されます。メッセージの送信が成功すると、両ビットはクリアされます。

送信中にアービトレーション ロストやバス エラーが発生すると、TX_REQ ビットはクリアされますが、TX_ABORT ビットはアサートされたままです。同時に、シングル ショット送信失敗 (sst_failure) 割り込みがアサートされます。

17.4.5 拡張データ フレームの送信

拡張データ フレームを送信するために、標準データ フレームと比較して以下のように特定のレジスタ設定を変更する必要があります。

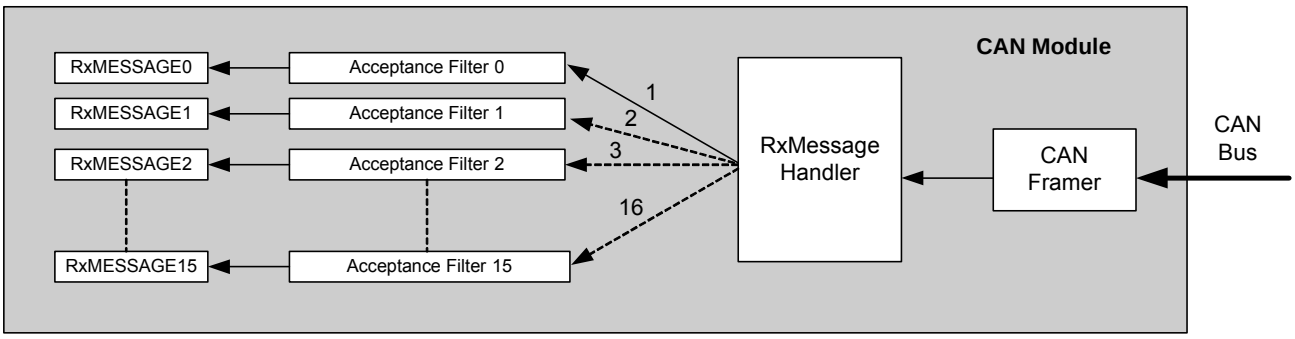
- 拡張データ フレームでは「1」(リセッシブ) を IDE ビットに書き込みます。
- メッセージ識別子を ID[31:3] ビット フィールドに書き込みます。

17.5 CAN におけるメッセージ受信

図 17-7 に示すように CAN モジュールには 16 個の受信メッセージ バッファがあります。各メッセージ バッファには 1 個の専用アクセプタンス フィルターがあります。CAN メッセージは CAN フレームで受信されます。受信されたメッセージは同時にすべてのアクセプタンス フィルターで比較され、受容されたメッセージは対応する受信メッセージ バッファに格納されます。メッセージ バッファのメッセージ アベイラビリティ (MSG_AV) ビットはセットされて新しいメッセージの存在を示します。別のメッセージを受信するために、MSG_AV ビットをクリアして受信されたメッセージをアクノリッジする必要があります。

アクセプタンス フィルターはアクセプタンス マスク レジスタ (AMR) とアクセプタンス コード レジスタで設定されます。

図 17-7. 受信 (Rx) ブロック図



17.5.1 メッセージ受信プロセス

図 17-8 に受信されるメッセージに関連するレジスタを示します。

図 17-8. 受信 (Rx) メッセージ レジスタ

REGISTERS															
COMMAND REGISTER (CAN_RXn_CMD)	Reserved [31:24]	WPNH [23]	Reserved1 [22]	RTR [21]	IDE [20]	DLC [19:16]	Reserved [15:8]	WPNL [7]	LINK FLAG [6]	RX INT ENBL [5]	RTR REPLY [4]	BUFFER EN [3]	RTR ABORT [2]	RTR REPLY PNDG [1]	MSG AV [0]
IDENTIFIER (CAN_RXn_ID)	ID [31:3]													Reserved [2:0]	
DATA REGISTER High (CAN_RXn_DH)	D0 [63:56]					D1 [55:48]			D2 [47:40]				D3 [39:32]		
DATA REGISTER Low (CAN_RXn_DL)	D4 [31:24]					D5 [23:16]			D6 [15:8]				D7 [7:0]		

n = 0,1,…,15

メッセージの受信手順は以下の通りです。

1. 新しいメッセージの受信後、RxMessageHandler ハードウェア (図 17-7 に示した) は有効なバッファを見つけるまでRxMessage0からすべての受信バッファを検索します。有効なバッファは以下のことで示されます。
   a. 受信バッファは BUFFER EN = 「1」 (CAN_RX[n].CONTROL[3]) で示されます。
   b. 受信バッファのアクセプタンス フィルターは着信するメッセージと一致します。
2. RxMessageHandler で検索された有効なバッファが空である場合、メッセージはそのバッファに格納され、その MSG AV ビットが「1」にセットされます。
3. RX INT ENBL ビットがセットされると、割り込みコントローラーRX_MSG フラグ (CAN_INT_STATUS[12]) はアサートされます。
4. 受信バッファがすでにメッセージを格納している場合 (MSG AV = 「1」かつ LINK_FLAG ビットがセットされていないことで示される)、RX_MSG_LOSS 割り込みフラグ (CAN_INT_STATUS[10]) はアサートされます。既存のメッセージは新しいメッセージで上書きされます。

注 : 受信メッセージ バッファが存在しているかは CAN バッファ レジスタ (CAN_BUFFER_STATUS) で決まります。

17.5.2 アクセプタンス フィルター

各受信バッファには、それぞれ着信するメッセージのフィルター処理に使用する独自のアクセプタンス フィルターが備えられています。アクセプタンス フィルターはアクセプタンス マスク レジスタ (AMR) とアクセプタンス コード レジスタ (ACR) で設定されます。AMR は、メッセージを受信するために着信するメッセージのどのビットが対応する ACR ビットと一致するかを定義します。

AMR: 「0」。着信ビットは対応する ACR ビットと照合されます。着信ビットが対応する ACR ビットと一致しない場合、メッセージは受容されません。
AMR: 「1」。着信ビットはドントケアです。
以下のメッセージ フィールドはフィルター処理されます。

- 識別子
- IDE
- RTR
- データ バイト 1(D0) とデータ バイト 2(D1) (DATA[63:48])[1]

標準 CAN メッセージでは、IDE = 0 の時、11 ビット識別子は AMR と ACR のビット [31:21] となります。

17.5.2.1 例

図 17-9 にメッセージおよびそれを受容するためのアクセプタンス フィルター設定を示します。

1. 184 ページの DeviceNet フィルター処理に示すように DeviceNet フィルターに役立ちます。

図 17-9. アクセプタンス フィルター

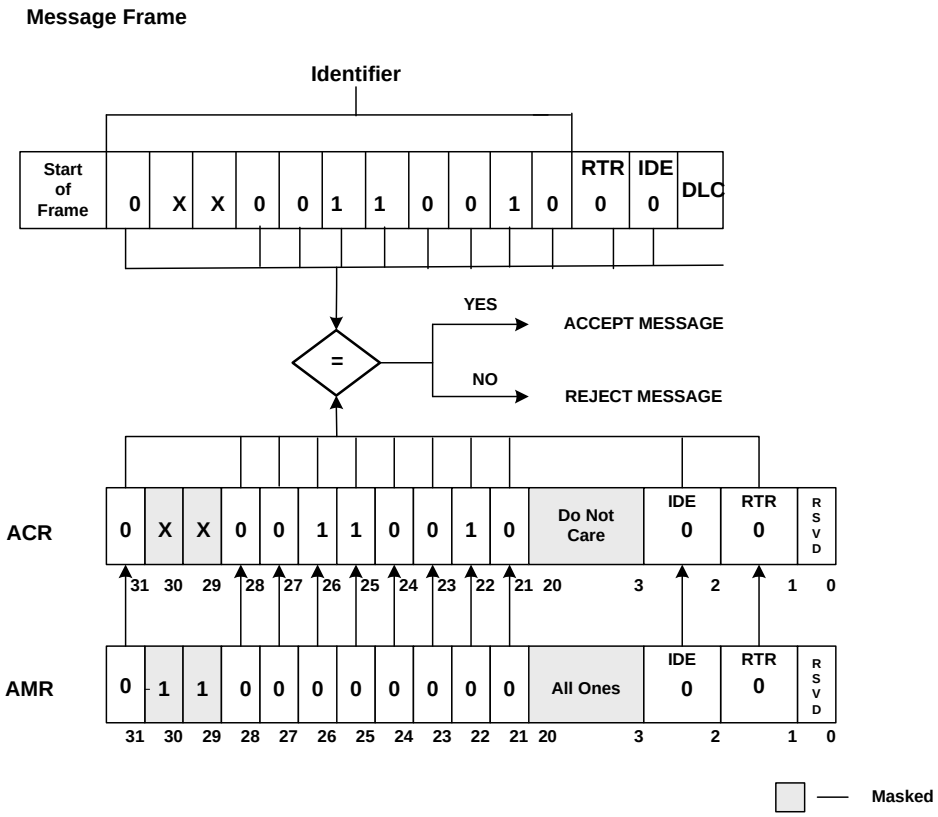


図 17-9 に示すようにグレーのエリアはマスクされたビットです。AMR レジスタであるビットが「1」にセットされると、ACR レジスタでの対応するビットは受信メッセージ フレームと照合されません。この例ではビット 30 と 29、3 ～ 20 が「1」にセットされマスクされています。AMR レジスタの残りのビットに「0」が書き込まれているため、図 17-9 に示すように、ACR レジスタでの対応するビットはメッセージと比較されます。ACR のビットがメッセージでの対応するビットと一致する場合、メッセージは受信メッセージ バッファに格納されます。ACR のビットがメッセージでの対応するビットと一致しない場合、着信したメッセージは拒否されます。

AMR の設定

ID[31]、ID[28:21] = 0
ID[30]、ID[29] = 1
ID[20:3] = 全ビット 1
IDE = 0
RTR = 0

ACR の設定

ID[31:21] = 182h
ID[20:3] = ドントケア
IDE = 0
RTR = 0

17.5.3 DeviceNet フィルター処理

DeviceNet などいくつかの CAN の高位プロトコルでは、プロトコルに関する情報が 1 と 2 番目のデータ バイトに格納されています。アクセプタンス フィルターはこれらの 2 バイトに追加のフィルター処理を行い、プロトコルの実装がより効果的になります。着信するメッセージの最初の 2 バイトのデータ ビットは ACR_DATA レジスタ (CAN_RX[n]_ACR_DATA) と比較され、比較される対応ビットは AMR_DATAレジスタ (CAN_RX[n]_AMR_DATA) で指定されます。10 ページの例を使用し、DeviceNet フィルター処理を図 17-10 に示します。

図 17-10. DeviceNet フィルター

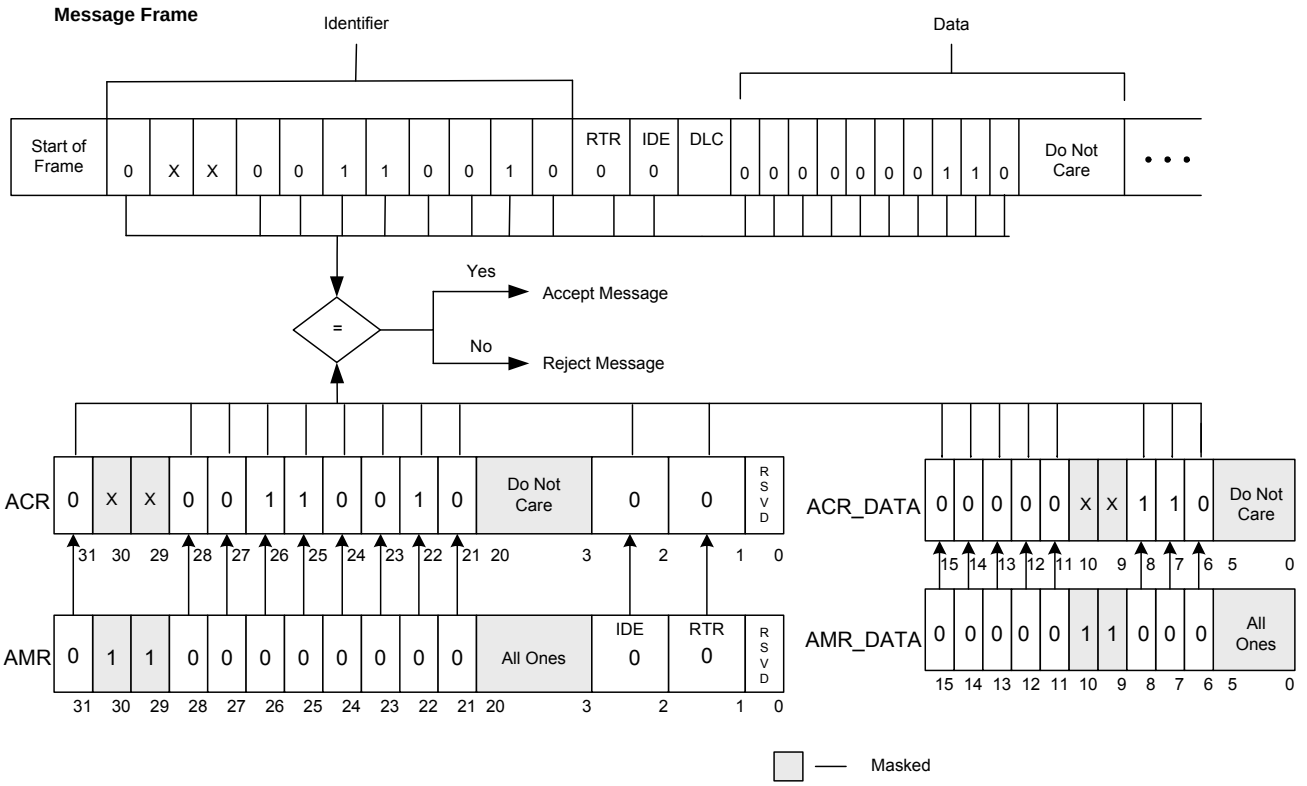


図 17-10 では、メッセージ フレームのデータ フィールドは AMR_DATA レジスタによってマスクされない ACR_DATA レジスタの対応するビットと比較されます。

このメッセージを受容するために、アクセプタンス フィルターを以下のように設定します。

AMR の設定 :

ID[28:21]、ID[31] = 0

ID[30]、ID[29] = 1

ID[20:3] = 全ビット 1

IDE = 0

RTR = 0

AMR_DATA[15:11]、AMR_DATA[8:6] = 0

AMR_DATA[10:9]、AMR_DATA[5:0] = 全ビット 1

ACR の設定 :

ID[31:21] = 182h

ID[20:3] = ドントケア

IDE = 0

RTR = 0

ACR_DATA[15:6] = 06h

ACR_DATA[5:0] = ドントケア

図 17-10の例は10データ ビットを用いたフィルター処理を示します。AMR_DATA を使うと、最大 16 個のデータ ビットをフィルター処理に用いられます。

17.5.4 拡張データ フレームのフィルター処理

拡張データ フレームのフィルター処理は標準データ フレームと似ていますが、拡張データ フレームをチェックするために AMR と ACR レジスタの IDE ビットをセットする必要がある点が異なります。

17.5.5 受信メッセージ バッファの連結

複数の受信バッファを連結して受信FIFOのような役割を果たす受信バッファ配列を形成することができます。これを実施するには次の手順を行います。

- 連結するバッファの CAN_RX[n].CONTROL[6] の LINK_FLAG ビットをセットします。
- 同じ配列ではすべてのバッファが同じメッセージ フィルター設定 (AMR と ACR は同一) を持っていることを保証します。
- 配列の最後のバッファの LINK_FLAG ビットをセットしないでください。

受信バッファがすでにメッセージを格納している場合

(MSG AV = 「1」)、新しいメッセージはバッファに着信すると放棄されます (RX_MSG_LOSS 割り込み)。この状況を防ぐには複数の受信バッファを相互に連結します。CAN コントローラーが新しいメッセージを受信すると RxMessageHandler は有効な受信バッファを検索します。バッファが満杯と判定され (MSG AV = 「1」)、LINK_FLAG がセットされていれば、有効な受信バッファの検索が引き続きます。有効な受信バッファが検索されるとメッセージはそのバッファに転送され、その結果として配列が形成されます。その他のバッファがなければ、RX_MSG_LOSS 割り込みがセットされます。

複数のメッセージ配列を形成することが可能です。各配列は同じ AMR と ACR を使用する必要があります。

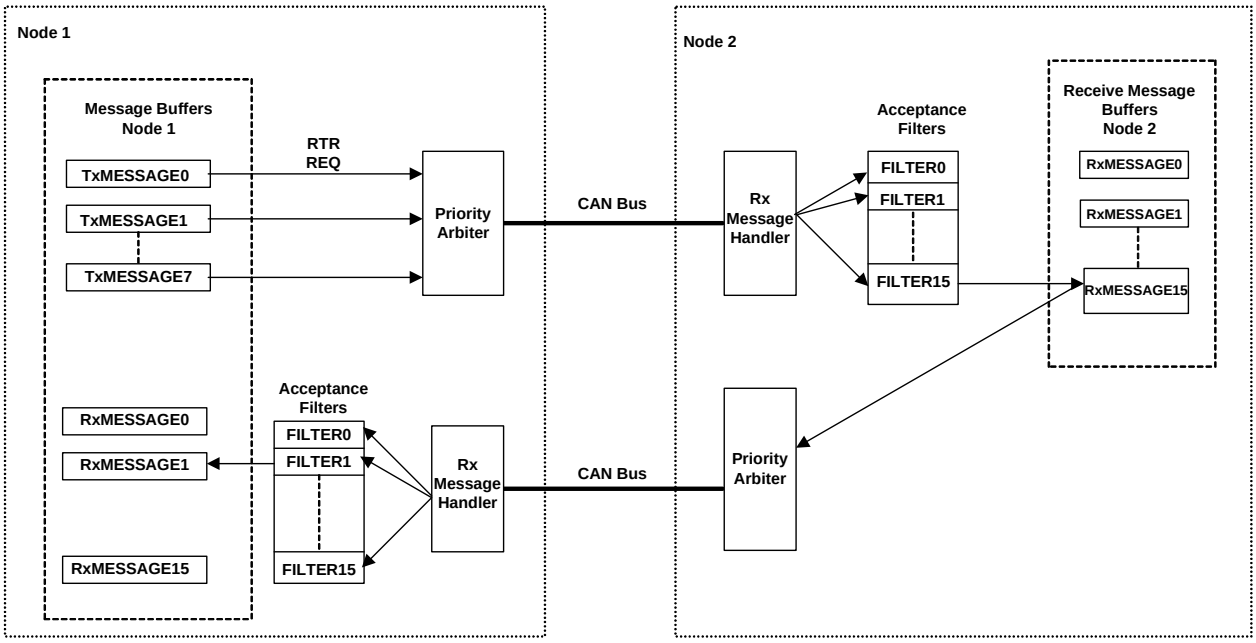
17.6 リモート フレーム

リモート フレームは 1 つのノード間の送信を開始するために使用されます。レシーバーとして機能するノードはリモート フレームを送信します。リモート フレームは標準または拡張のフォーマットを用いられます。リモート フレームは、RTR ビットが常に「1」で DLC フィールドの値にかかわらずデータ フィールドがないという点でデータ フレームと異なります。図 17-11 にリモート送信要求の流れを示します。

図 17-11 に示すように :

- ノード 1 のメッセージ バッファ 0 はリモート フレームを CAN バスに送信します。
- RTR 要求はノード 2 の RxMessageHandler によって受信され、アクセプタンス フィルターに送信されます。
- 受信メッセージ バッファ 15 のアクセプタンス フィルター設定はメッセージのアクセプタンス フィルター設定と一致すれば、メッセージは受信メッセージ バッファ 15 に移動されます。
- RTR 自動応答機能が有効になった場合、受信メッセージ バッファ 15 は受信された時の識別子を持つメッセージを送信します (CPU 介入なし)。
- RTR の送信が成功すると、CAN_RX.CONTROL レジスタの応答ビット MSG_AV_RTR_SENT および割り込みステータス レジスタの RTR_MSG ビットがセットされます。
- ノード 1 の受信メッセージ バッファ 1 のアクセプタンス フィルターはノード 1 の送信されたメッセージと同じ識別子設定を持っています。したがって RTR メッセージはノード 1 の受信メッセージ バッファ 1 に格納されます。

図 17-11. リモート送信要求



17.6.1 要求ノードによるリモート フレームの送信

要求ノード (図 17-11 に示したノード 1) によりリモート フレームを送信するプロセスは以下の通りです。

1. メッセージを空の送信バッファに書き込みます。空のバッファは TX_REQ = 「0」(CAN_TX[n]_CONTROL[0]) で示されます。
2. RTR ビットを (CAN_TX[n]_CONTROL[21]) 「1」にセットします。
3. 適切な優先順位アービトレーション方式を選択します。
4. 送信要求フラグをセットして送信を開始します。
5. メッセージで送信された識別子は受信メッセージの識別子と同じである必要があります。

17.6.2 リモート フレームの受信

リモート フレームの受信手順は以下の通りです。

1. アクセプタンス フィルターは期待するメッセージ ID を受信するよう設定する必要があります。
2. RTR REPLY ビットを「1」にセットして自動 RTR メッセージ処理を有効にします。
   a. 有効になっている時同じ識別子を持つリモート フレームを自動的に送信します。
   b. 有効になっていない時リモート　フレームはデータ フレームと同じように標準ルーチンに続いて送信する必要があります。
3. 応答された RTR メッセージを受信する要求ノードを、通常のメッセージを受信するよう設定します。RTR REPLY ビットをセットしないでください。

17.6.3 RTR 自動応答

CAN モジュールでは RTR メッセージ要求の自動応答をサポートしています。全 16 受信バッファはこの機能に対応しています。RTR REPLY FLAG がセットされている時 RTR メッセージが受信バッファで受容されると、バッファはメッセージに対して受信バッファの内容で自動的に応答します。RTR メッセージ要求が受信されると RTR REPLY PNDG FLAG はセットされます。メッセージが送信された時またはメッセージ バッファが無効にされた時リセットされます。保留中の RTR 応答メッセージを中止するために RTR ABORT コマンドを使用します。

17.6.4 拡張フォーマットでのリモート フレーム

拡張フォーマットでのリモート フレームの送受信は以下の点を除いて標準フォーマットに似ています。

- 拡張データ フレームにするために IDE ビット (CAN_TX[n].CONTROL [20]) を「1」にセットします。
- 識別子は標準データ フレームでは 11 ビット、拡張データ フレームでは 29 ビットです。

17.7 時間起動 CAN

時間起動 CAN (TTCAN) は CAN プロトコルよりも高位のプロトコル レイヤーです。TTCAN は CAN メッセージが周期的であるアプリケーションには役立ちます。TTCAN システムではメッセージの同期化はネットワーク内の時間マスターからの基準メッセージに基づいて行われます。以下の機能は CAN コントローラーに備わっており、それをレベル 1 の TTCAN システムの実装に適させます。

- 1.4.4 節で説明したようにシングル ショット送信に対応
- TTCAN メッセージのタイミング用の内部タイマー

注: リモート フレームの受信に設定された TTCAN 受信バッファでは CAN_RX.CONTROL.RTR_REPLY ビットをクリアして自動メッセージ応答を無効にする必要があります。

17.7.1 TTCAN タイマー

CAN ブロックに含まれている 16 ビット タイマー サブブロックは TTCAN システムのタイマーとして使用できます。このブロックは CAN 制御レジスタの設定で有効にします (CNTL.TT_ENABLE)。タイマーは公称の CAN ビット タイムをカウントし、CAN バス上にフレームの開始 (SOF) の検出時にタイマー カウントを取り込みます。以下は TTCAN タイマーに関連するレジスタです。

- TTCAN_COUNTER

16 ビットのローカル タイマー カウンター レジスタ (TTCAN_COUNTER.LOCAL_TIME)。TTCAN_TIMING レジスタのタイミング設定に基づいて公称のビット タイムをカウントします。

- TTCAN_COMPARE

時間イベントを生成するためのローカル カウンターの比較値です。TTCAN_COUNTER.LOCAL_TIME が TTCAN_COMPARE レジスタ値までカウントすると tt_compare ハードウェア イベントがトリガーされます。割り込みレジスタ (INTR_CAN_SET) の TT_COMPARE ビットが有効になると割り込みステータス レジスタ (INTR_CAN) での対応するビットがセットされます。詳細は 1.9.4.2 節を参照してください。

- TTCAN_CAPTURE

SOF が CAN バス上 (CAN Rx 入力) で検出されるとこのレジスタは TTCAN_COUNTER の値を取り込みます。またこれは tt_capture ハードウェア イベントをトリガーします。割り込みレジスタ (INTR_CAN_SET) の TT_CAPTURE ビットが有効になると割り込みステータス レジスタ (INTR_CAN) での対応するビットがセットされます。詳細は 1.9.4.2 節を参照してください。

基準フレームの SOF は TTCAN システムの同期化信号として使用されます。

AMR ／ ACR レジスタは基準メッセージのメッセージ ID をフィルタ処理するために使用できます。基準メッセージの時間を同期化するために基準メッセージの検出時に TTCAN_CAPTURE レジスタ値を読み出すことができます。

- TTCAN_TIMING

このレジスタは TTCAN カウンターのクロックを生成するために公称のビット タイミングを設定します。CAN コンフィギュレーション レジスタと同じビット タイム設定にする必要があります。

注: アプリケーション レベルで TTCAN のシステム デザインに注意を払わなければなりません。システム設計者は TTCAN システムを構築するために利用可能なハードウェア リソースをどう活用するかを決める必要があります。

17.8 ビット タイム コンフィギュレーション

CAN モジュールは単一のクロック入力 SYSCLK に基づいて動作します。本節は所望のビット レートを達成するためにプログラム可能なビット レート分周器を設定する方法および SYSCLK との関係を説明します。

17.8.1 取り得るビット レートとステム クロック (SYSCLK)

業界全般にわたりほとんどの CAN バスの実装には、次の 10 種類のビット レートのいずれかが使用されています。

- 1Mbps
- 800Kbps
- 500Kbps
- 250Kbps
- 125Kbps
- 100Kbps
- 50Kbps
- 20Kbps
- 10Kbps
- 5Kbps

SYSCLK が 8MHz またはその倍数である場合、これらのビット レートが設定可能です。SYSCLK が 10MHz またはその倍数である場合、800Kbps を除いてこれらが設定可能です。いくつかの例外を除き、SYSCLK が 1MHz の偶数倍でない場合、10 種類全てのビット レートの実現は不可能です。ビット レート生成のため SYSCLK 精度は 125Kbps 以下のビット レートでは少なくとも 1.58%、125Kbps より高いビット レートでは 0.5% でなければなりません。CAN ブロックの高精度クロック要件を満たすには、32kHz WCO PLL ロックを備えた IMO を使用することもでき、高精度の外部クロックをデバイスに接続して IMO クロック ソースとして使用することもできます。PSoC 4200M デバイス ファミリでは、MHz 帯域の外部水晶をクロック ソースとして使用することに対応していません。図 17-12 に 8MHz ～ 100MHz の特定の fclk でに対応する 10 種類のビット レートの表を示します。PSoC 4200M の最大周波数は 48MHz です。

図 17-12. ビット レートと SYSCLK

clk_bus Freq (MHz)	1 Mb	800 Kb	500 Kb	250 Kb	125 Kb	100 Kb	50 Kb	20 Kb	10 Kb	5 Kb
8	■	■			■	■	■	■	■	■
9					■	■	■	■	■	■
10	■		■	■	■	■	■	■	■	■
11					■	■	■	■	■	■
12			■	■	■	■	■	■	■	■
13					■	■	■	■	■	■
14				■	■	■	■	■	■	■
15			■	■	■	■	■	■	■	■
16	■	■	■	■	■	■	■	■	■	■
17					■	■	■	■	■	■
18				■	■	■	■	■	■	■
19					■	■	■	■	■	■
20	■		■	■	■	■	■	■	■	■
21					■	■	■	■	■	■
22				■	■	■	■	■	■	■
23					■	■	■	■	■	■
24	■	■	■	■	■	■	■	■	■	■
25			■	■	■	■	■	■	■	■
26				■	■	■	■	■	■	■
27					■	■	■	■	■	■
28			■	■	■	■	■	■	■	■
29					■	■	■	■	■	■
30	■		■	■	■	■	■	■	■	■
31					■	■	■	■	■	■
32	■	■	■	■	■	■	■	■	■	■
33					■	■	■	■	■	■
34				■	■	■	■	■	■	■
35			■	■	■	■	■	■	■	■
36			■	■	■	■	■	■	■	■
37					■	■	■	■	■	■
38				■	■	■	■	■	■	■

clk_bus Freq (MHz)	1 Mb	800 Kb	500 Kb	250 Kb	125 Kb	100 Kb	50 Kb	20 Kb	10 Kb	5 Kb
39					■	■	■	■	■	■
40	■	■	■	■	■	■	■	■	■	■
41					■	■	■	■	■	■
42				■	■	■	■	■	■	■
43					■	■	■	■	■	■
44			■	■	■	■	■	■	■	■
45			■	■	■	■	■	■	■	■
46				■	■	■	■	■	■	■
47					■	■	■	■	■	■
48	■	■	■	■	■	■	■	■	■	■
49					■	■	■	■	■	■
50	■		■	■	■	■	■	■	■	■
51					■	■	■	■	■	■
52			■	■	■	■	■	■	■	■
53					■	■	■	■	■	■
54				■	■	■	■	■	■	■
55			■	■	■	■	■	■	■	■
56	■	■	■	■	■	■	■	■	■	■
57					■	■	■	■	■	■
58				■	■	■	■	■	■	■
59					■	■	■	■	■	■
60	■		■	■	■	■	■	■	■	■
61					■	■	■	■	■	■
62				■	■	■	■	■	■	■
63					■	■	■	■	■	■
64	■	■	■	■	■	■	■	■	■	■
65			■	■	■	■	■	■	■	■
66				■	■	■	■	■	■	■
67					■	■	■	■	■	■
68			■	■	■	■	■	■	■	■
69					■	■	■	■	■	■

clk_bus Freq (MHz)	1 Mb	800 Kb	500 Kb	250 Kb	125 Kb	100 Kb	50 Kb	20 Kb	10 Kb	5 Kb
70	■		■	■	■	■	■	■	■	■
71					■	■	■	■	■	■
72	■	■	■	■	■	■	■	■	■	■
73					■	■	■	■	■	■
74				■	■	■	■	■	■	■
75			■	■	■	■	■	■	■	■
76			■	■	■	■	■	■	■	■
77					■	■	■	■	■	■
78				■	■	■	■	■	■	■
79					■	■	■	■	■	■
80	■	■	■	■	■	■	■	■	■	■
81					■	■	■	■	■	■
82				■	■	■	■	■	■	■
83					■	■	■	■	■	■
84			■	■	■	■	■	■	■	■
85			■	■	■	■	■	■	■	■
86				■	■	■	■	■	■	■
87					■	■	■	■	■	■
88	■	■	■	■	■	■	■	■	■	■
89					■	■	■	■	■	■
90	■		■	■	■	■	■	■	■	■
91					■	■	■	■	■	■
92			■	■	■	■	■	■	■	■
93					■	■	■	■	■	■
94				■	■	■	■	■	■	■
95			■	■	■	■	■	■	■	■
96	■	■	■	■	■	■	■	■	■	■
97					■	■	■	■	■	■
98				■	■	■	■	■	■	■
99					■	■	■	■	■	■
100	■		■	■	■	■	■	■	■	■

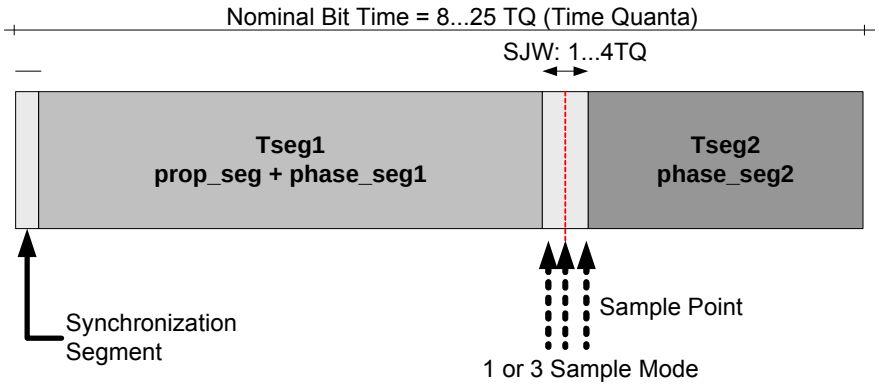
■ Configurable Bit Rates

Non Configurable Bit Rates

17.8.2 ビット レート TSEG1 と TSEG2 の設定

ビット レートはCANバスで毎秒送信されるビット数と定義されます。ビット タイムはビット レートの逆数です。図 17-13 に示すようにビット タイムは 3 つのセグメントに分割されています。各セグメントはシステム クロック (SYSCLK) から得られるタイム クォンタム (Time Quanta ; TQ) と呼ばれる一定の時間単位で表されます。

図 17-13. ビット タイム



$$BitTime = (1 + tseg1 + 1 + tseg2 + 1)TQ$$ **式 1**

$$TQ = \frac{BRP + 1}{SYSCLK}$$ **式 2**

注：ビット レート プリスケーラは CAN モジュール用のクロックを生成するために SYSCLK に対して分周機能を実行するレジスタです。図 17-14 を参照してください。

同期化セグメント： これは 1 TQ 長の最初のセグメントであり、同期化に用いられます。このセグメント内に、立ち下がりエッジが発生すると予想されます。

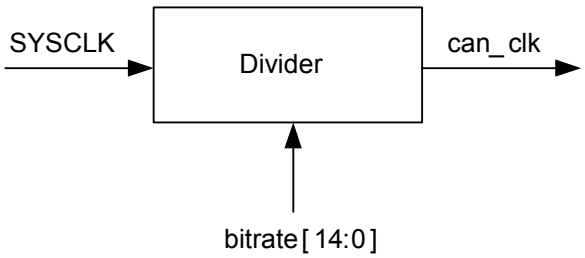
Tseg1、Tseg2: これらのセグメントはエッジ位相誤差を補正するものです。tseg1 はネットワークにおける遅延時間を含む伝播時間を含みます。セグメントの長さはエッジの位相誤差を補正するために増減されます。これは再同期化と呼ばれています。

サンプル ポイント ：このポイントではバス状態が読み出され、ビットが解釈されます。これは tseg1 の終わりに位置します。

同期化ジャンプ幅：再同期化すると、tseg1 が延長されるか tseg2 が短縮されます。この再同期化は同期化ジャンプ幅によって限界が設けられます。tseg2 の長さは同期化ジャンプ幅より長いものでなければなりません。

コンフィギュレーション レジスタ CAN_CONFIG はビット レート プリスケーラ (BRP)、tseg1、tseg2、同期化ジャンプ幅をセットするために用いられます。CAN ペリフェラル クロック (CAN_CLK) はシステム クロック (SYSCLK) を (BRP+1) で割ることで生成されます。システム クロック生成のオプションについてはクロック供給の節を参照してください。ビット タイムに N のタイム クォンタムがある場合、CAN ペリフェラル クロック周波数を CAN バス ビット レートの N 倍に設定する必要があります。

図 17-14. ビット タイミング ブロック図



17.8.2.1 例

以下は 40MHz で動作する際に 1Mbps を達成する例です。

周波数は 1MHz で、ビット タイムは 1µs です。

ビット タイムに 8 TQ (最小 TQ 数) を選択し、1TQ = 0.125µs です。

BRP = ((タイム クォンタム * SYSCLK) – 1) = 4

したがって「4」をコンフィギュレーション レジスタの CFG_BITRATE ビットに書き込みます。

サンプリング ポイントをビット タイムの約 60%、すなわち約 5TQ と選択します。サンプリング ポイントが tseg1 の終わりにあるため、(tseg2+1) = 3TQ、すなわち tseg2 = 2TQ です。

サンプリング ポイント同期化ジャンプ幅を固定するために CFG_SJW ビットを「1」にセットします。

ビット cfg_tseg2 に「2」を書き込んで tseg2 の値を 2TQ にセットします。

次の式で tseg1 を計算します: tseg1 = ((ビット タイム - (1TQ + tseg2 + 1TQ)) - 1TQ

ここに tseg1 = 3TQ。

したがってコンフィギュレーション レジスタのビット cfg_tseg1 に「3」を書き込みます。

この手順は図 17-12 に示したようにクロック周波数を用い

て標準ビット レートを達成するために適用されます。

次の tseg1 と tseg2 の設定条件を適用します。

- tseg1 = 0 または tseg1 = 1 は許可されません。
- tseg2 = 0 は許可されず、tseg2 = 1 はダイレクト サンプリング モードでのみ許可されます。

注 1: コンフィギュレーション レジスタ (CAN_CONFIG) の Sampling_mode ビットはレシーバー パスで 1 つのサンプリング ポイントを使用するか、多数決論理による 3 つのサンプリング ポイントを使用するかを指定します。

注 2: コンフィギュレーション レジスタ (CAN_CONFIG) の Edge_mode ビットは立ち下がりエッジか両エッジが同期化に使用するかを指定します。

17.9 CAN におけるエラー処理と割り込み

CAN プロトコル仕様によると、5 種類のエラーがあります。バスに接続する各 CAN ノードはエラーを検出しようとし、検出するとエラー フレームを送信します。以下の節は各種エラーおよびエラー処理プロセスについて説明します。

17.9.1 エラーの種類

17.9.1.1 BIT エラー

バスにビットを送信する CAN ユニットはバスを監視します。監視したビット値が送信したビット値と違うと BIT エラーが検出されます。1 つの例外としては、アービトレーション フィールドをスタッフするビット ストリームの間または ACK スロットの間、「リセッシブ ビット」を送信することです。パッシブ エラー フラグを送信し「ドミナント ビット」を検出したトランスミッターはこれを BIT エラーと解釈しません。

17.9.1.2 FORM エラー

CAN メッセージ フォーマットにエラーがあると FORM エラーが検出されます。フレームの終了やフレーム間のスペースなどメッセージ内の固定フォーマット フィールドには不正ビットがあります。

17.9.1.3 ACKNOWLEDGE エラー

ACK スロットの間にリセッシブ ビットを送信するトランスミッターは、ドミナント ビットの有無を確認するために ACK スロットを監視します。レシーバーがメッセージを正しく受信すると、ドミナント ビットが ACK に書き込まれます。したがって送信後にトランスミッターが ACK スロットでドミナント ビットを見つけないと ACKNOWLEDGE エラーが検出されます。

17.9.1.4 CRC エラー

送信ノードは特定の計算を行って CRC コードを生成し、それを CRC フィールドで送信します。受信ノードは同じ計算を行って CRC コードを生成します。レシーバーによって生成されたコードが送信されたコードと一致しないと CRC エラーは検出されます。

17.9.1.5 STUFF エラー

ビット スタッフのメッセージで符号化されたメッセージ フィールドに同じレベルの 6 つの連続したビットがある場合、6 番目のビットのビット タイム中に STUFF エラーが検出されます。

17.9.2 エラー キャプチャ レジスタ

PSoC 4200M CAN コントローラーには CAN バス診断に使用する専用のエラー キャプチャ レジスタ (ECR) があります。図 17-15 にエラー キャプチャ レジスタを示します。

図 17-15. エラー キャプチャ レジスタ

Reserved [31:17]	FIELD [16:12]	BIT [11:6]	TX MODE [5]	RX MODE [4]	ERROR TYPE [3:1]	ECR STATUS [0]

エラー キャプチャ レジスタは 2 つのモードがあります。

フリーラン モード : このモードでは ECR は現時点の CAN フレーム内のフィールドとビット位置を取り込みます。

エラー キャプチャ モード : このモードでは CAN エラーが検出されると ECR はフィールドとビット位置をサンプリングします。このようなイベントをサンプリングするには、書き込みアクセスを実行することで ECR を起動する必要があります。ECR は起動されると 1 つのみのエラー イベントを取り込みます。連続したエラー キャプチャのためには ECR に「1」を書き込むことで再度設定をする必要があります。

注 : IDE ビットは ECR.FIELD のアービトレーション フィールドとして見なされます。

17.9.3 CAN におけるエラー状態

CAN は 3 つのエラー状態があります。

- **エラー アクティブ :** エラー アクティブのノードは通常のバス通信に参加することができます。エラーを検出するとエラー アクティブ フラグを送信します。
- **エラー パッシブ :** エラー パッシブのノードはバス通信に参加します。エラーを検出するとエラー パッシブ フラグを送信します。エラー パッシブ フラグを送信した後、次の送信を開始するまで待機します。エラー パッシブのノードはフレーム間のスペース中に 8 リセッシブ ビットを送信します。この期間は送信が行われないため「送信の一時停止」と呼ばれます。
- **バス オフ :** この状態のノードはバス通信に参加しません。またバスには影響を与えません。

CAN におけるエラー状態はエラー ステータス レジスタ (CAN_ERROR_STATUS) で示されます。ERROR_STATE ビット (CAN_ERROR_STATUS[17:16]) は CAN ノードがどのエラー状態にあるかを示します。CAN におけるエラー状態は 2 つのカウンターの値によって判定されます。

- 送信エラー カウンター (CAN_ERROR_STATUS[7:0])
- 受信エラー カウンター (CAN_ERROR_STATUS[15:8])

エラー カウンターは CAN 2.0B 仕様にしたがって更新されます。

送信エラー カウンターと受信エラー カウンターが 127 (10 進) 以下の場合ノードは「エラー アクティブ」状態となります。送信または受信エラー カウンターが 128 (10 進) 以上の場合ノードは「エラー パッシブ」状態となります。送信エラー カウンターが 256 (10 進) 以上の場合ノードは「バス オフ」状態となります。

両方のエラー カウンターが 127 以下の場合「エラー パッシブ」のノードは「エラー アクティブ」のノードに戻ります。

バス上に連続した 11 リセッシブ ビットの発生が 128 回検出された後、両方のエラー カウンターが「0」にリセットされ、「バス オフ」のノードが「エラー アクティブ」となります。

エラー ステータス レジスタはtxgte96 (CAN_ERROR_STATUS[18]) と rxgte96 (CAN_ERROR_STATUS[19]) の 2 つのビットがあります。これらのビットは送信エラー カウンターと受信エラー カウンターそれぞれが 96 (10 進) 以上であるかを示します。エラー カウント値が 96 以上ならバスの負荷が高いことを示すためこの機能はエラー警告として使えます。

17.9.4 CAN における割り込みソース

CAN 割り込みソースは CAN コア割り込みと TTCAN 割り込みの 2 つのカテゴリに分けられています。制御レジスタ (CAN.CNTL) の TT_ENABLE ビットは割り込みの送信を制御します。

- TT_ENABLE = 0 の場合割り込みは CAN コア割り込み (INT_STATUS と INT_ENBL) から直接送信されます。
- TT_ENABLE = 1 の場合、割り込みはコア割り込みと TTCAN タイマー割り込みから生成されます。

17.9.4.1 CAN からのコア割り込み

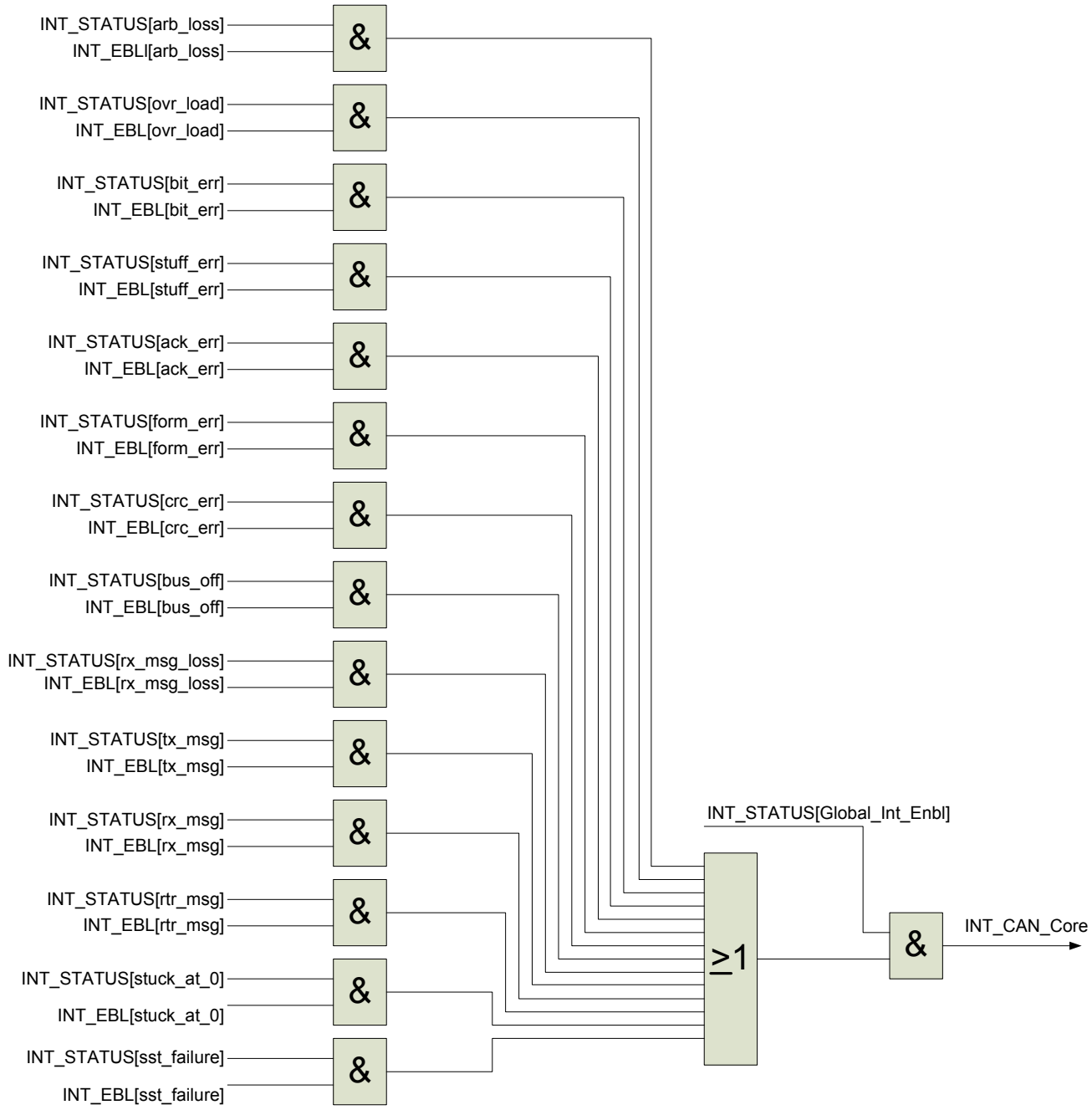
図 17-16 に示すようにコア割り込みは割り込みステータス レジスタ (CAN_INT_STATUS) と割り込みイネーブル レジスタ (CAN_ INT_EBL) で制御されます。割り込みステータス レジスタはコア割り込みイベントを格納します。ビットは一旦セットされると、「1」が書き込まれてクリアされるまで、セットされたままです。割り込みイネーブル レジスタは割り込みステータス レジスタに影響を与えません。

割り込みイネーブル レジスタ (INT_EBL) は割り込みステータス レジスタのどのビットを割り込み出力 (INT_CAN_Core) をアサートするために使用するかを制御します。INT_CAN_Core は、特定の割り込みステータス ビットと対応するイネーブル ビットがセットされるとアサートされます。INT_CAN_Core の送信は CAN_CNTL.TT_ENABLE ビットで制御されます。TT_ENABLE ビットが「0」であれば INT_CAN_Core 信号は CAN ブロック割り込み出力に直接接続されます。TT_ENABLE が 1 にセットされると割り込みは図 17-11 に示すように INTR_CAN_SET レジスタ設定に応じて送信されます。

CAN におけるさまざまなコア割り込みソースは以下の通りです。

- rx_msg: メッセージが受信された。
- tx_msg: メッセージが送信された。
- rx_msg_loss: 新しいメッセージが着信するがMSG AVフラグがセットされる場合にセットされる。
- bus_off: CAN はバス オフ状態になった。
- crc_err: CAN CRC エラーが検出された。
- form_err: CANメッセージ フォーマット エラーが検出された。
- ack_err: CAN メッセージ アクノリッジ エラーが検出された。
- stuff_err: ビット スタッフ エラーが検出された。
- bit_err: ビット エラーが検出された。
- ovr_load: オーバーロード フレームが受信された。
- arb_loss: メッセージの送信中にアービトレーションに負けた。
- stuck_at_0: ドミナント (0) エラーが検出された時に中止される。
- rtr_msg: RTR 自動応答メッセージが送信された。
- sst_failure: シングル ショット送信。

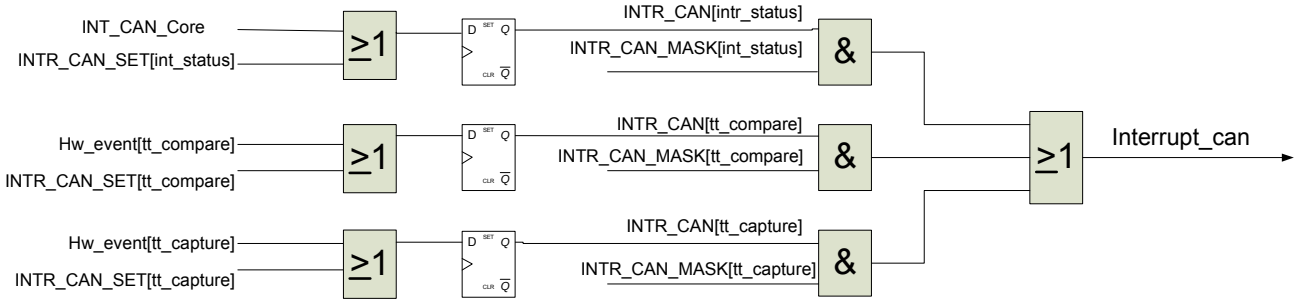
図 17-16. CAN コアからの割り込み



17.9.4.2 TT_ENABLE = 1 による割り込みルーティング

CNTL.TT_ENABLE ビットが「1」にセットされた場合、CAN 割り込み信号は図 17-17 に示すように INT_CAN_SET と INT_CAN_MASK レジスタ設定に応じて CAN コア割り込みまたは追加の TTCAN タイマー ハードウェア割り込みから生成されます。

図 17-17. CAN_CNTL.TT_ENABLE = 1 による CAN 割り込み



INTR_CAN_SET と INTR_CAN_MASK の論理積は INTR_-CAN_MASKED レジスタで取ります。CAN コア割り込みおよび TTCAN タイマー比較とキャプチャ割り込みが、INTR_-CAN_SET レジスタで有効にされた場合 INTR_CAN レジスタはそれらの状態を格納します。

17.10 CAN における動作モード

CAN モジュールは 3 つのモードで動作します。動作モードの選択は、コマンド レジスタ CAN_COMMAND の各モードに対応するビットをセットして行います。3 つの動作モードは以下の通りです。

- 実行／停止モード : CAN_COMMAND[0]
- リッスンオンリー モード : CAN_COMMAND[1]
- ループバック テスト モード : CAN_COMMAND[2]

17.10.1 実行／停止モード

実行モードでは CAN コントローラーは正常に動作します。CAN コントローラーは CAN_COMMAND[0] ビットをセットして実行モードに移行させ、ビットをクリアして停止させます。

17.10.2 リッスンオンリー モード

リッスンオンリー モードでは、CAN コントローラーはバス上で受信されたメッセージをアクノリッジせずに CAN 受信ラインをモニターします。このモードではメッセージを送信しません。しかしエラーがなくなるまでビット タイミングを調整するためにエラー フラグが更新されます。

自動ボーレート検出の手順は以下の通りです。

1. CAN コントローラーはすべてのメッセージを受容するために初期化されます (グローバル／ローカル マスクが「0」にセットされます)。
2. 最初のあり得る CANOpen ビット レート (10Kbps) がロードされ、コントローラーは「リッスンオンリー」モードに切り替わります。
3. ネットワーク上のトラフィックがあり、ビット レートが正しければ、メッセージは受容されます。
4. エラー レジスタは変化せず、メッセージ受信のフラグが CAN コントローラー内にセットされます。正しいビットレートが検出されたことを意味します。
5. ビット レートが正しくなければ、エラー フラグが更新されます (スタッフ、CRC またはフォーマット エラー)。
6. CAN コントローラーはオフになり、次のあり得るビットタイミング値がビット レート テーブルからロードされます。

17.10.3 ループバック テスト モード

ループバック モードはテストのために使用されます。PSoC 4200M CAN コントローラー ブロックはコマンド レジスタの LOOPBACK ビット (CAN_COMMAND[2]) と LISTEN ビット (CAN_COMMAND[1]) の設定に応じて 2 種類のループバック モードをサポートします。

17.10.3.1 外部ループバック モード

外部ループバック モードは COMMAND_LOOPBACK = 1、COMMAND_LISTEN = 0 の場合有効になります。このモードでは CAN_TX 出力ピンは外部を経由して CAN_RX 入力ピンに接続します。これにより送信されたデータを受信することができます。

17.10.3.2 内部ループバック モード

内部ループバック モード COMMAND_LOOPBACK = 1、COMMAND_LISTEN = 1 の場合有効になります。このモードでは送信されたデータは内部でレシーバー ロジックに戻って送信されます。

18. タイマー、カウンター、PWM

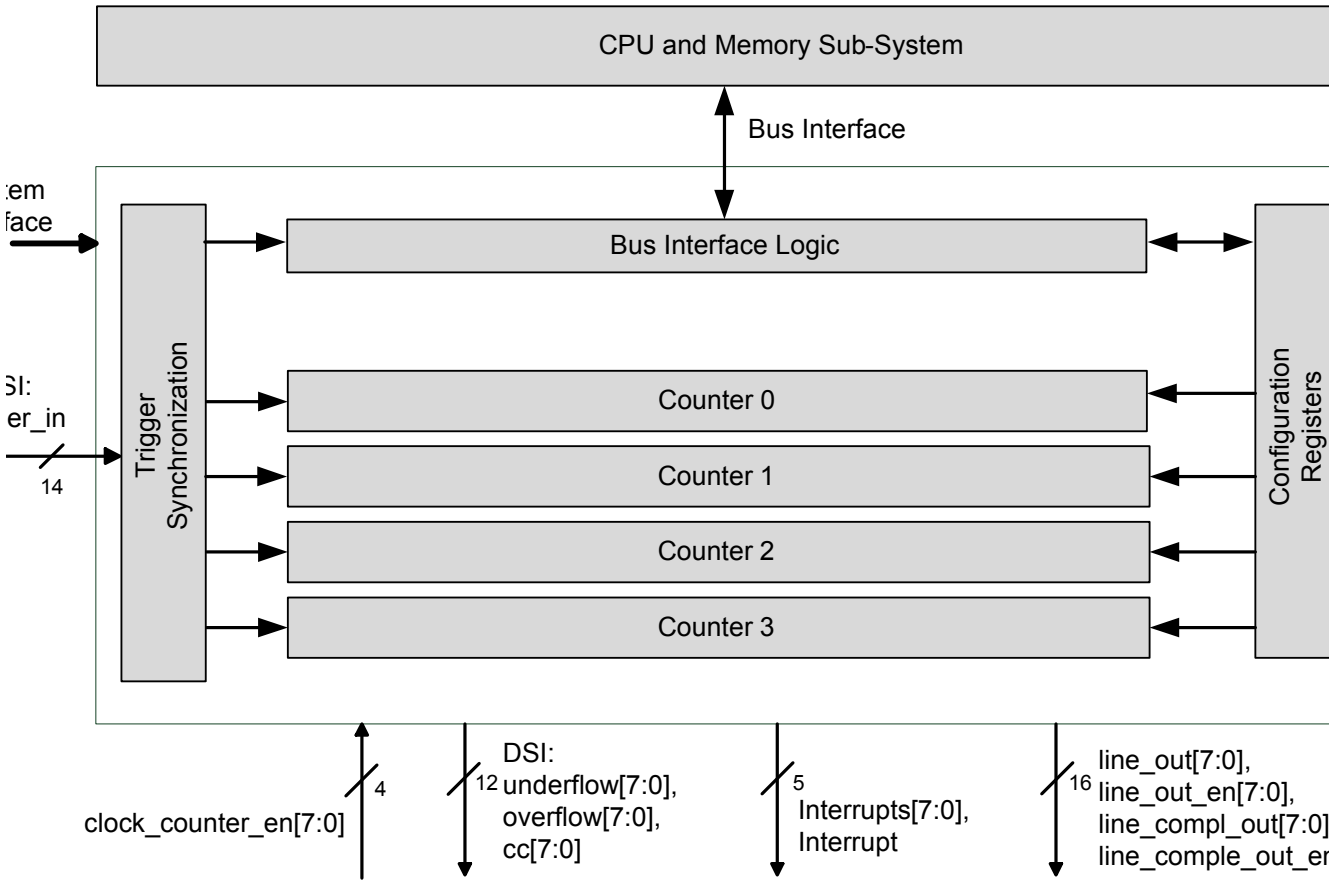
PSoC® 4 内のタイマー、カウンター、パルス幅変調器 (TCPWM) ブロックは 16 ビット タイマー、カウンター、パルス幅変調器 (PWM)、直交デコーダーの機能を実装しています。ブロックは入力信号の周期とパルス幅の測定 (タイマー) や特定のイベントが発生する回数のカウント (カウンター)、PWM 信号の生成、直交信号の復号に使用します。本章は TCPWM ブロックの機能、実装、動作モードを説明します。

18.1 特長

- 8 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達する
  - 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
- 同期化カウンター : 複数のカウンターが同時にリロード、開始、停止、カウント可能
- PWM のコンプリメンタリ出力

18.2 ブロック図

図 18-1. 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します。
- DSI による I/O 信号インターフェース : 汎用デジタル ブロック (UDB) および TCPWM ブロックと信号を配線します。入力トリガー (リロード、開始、停止、カウント、キャプチャなど) と出力信号 (オーバーフロー (OV)、アンダーフロー (UN)、キャプチャ／比較 (CC)) を含みます。任意の GPIO が入力トリガー信号に使用できます。
- 割り込み: ターミナル カウント (TC) またはCC条件に基づいて各カウンターから割り込み要求信号およびすべての 8 本の割り込み要求信号の論理和で生成された結合割り込み信号を提供します。
- 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含みます。

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 213 ページの TCPWM レジスタを参照してください。

18.2.1 TCPWM ブロックにおけるカウンターのイネーブル

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に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。ステータス レジスタはカウンターが無効になった後にクリアされます。

18.2.2 クロック

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。これは表 18-1 に示すようにカウンター制御レジスタ (TCPWM_CNT_CTRL) の GENERIC フィール

ドを変更することで行います。

表 18-1. 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	分周なし
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デッドタイム付きパルス幅変調モード (PWM-DT) では行えません。

18.2.3 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
- 開始
- 停止
- 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、立ち上がりエッジ、立ち下がりエッジまたは両エッジです。

図 18-2. TCPWM トリガー選択およびイベント検出

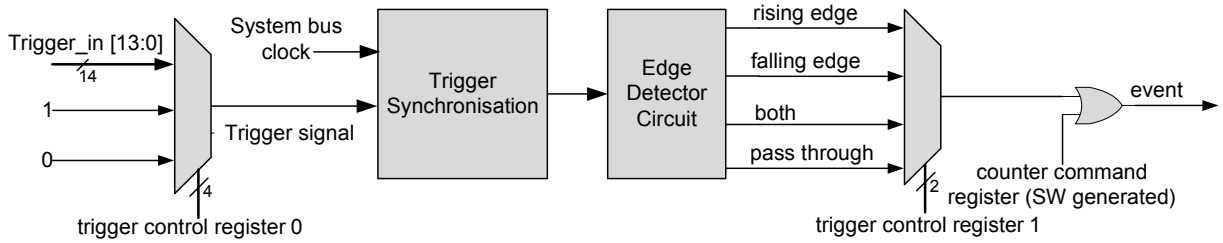


図 18-2 はTCPWMブロックのトリガー選択およびイベント検出を示します。トリガー制御レジスタ 0 (TCPWM_CNT_TR_CTRL0) は 14 つのトリガー入力から 1 つをイベント信号と選択します。さらにイベント信号に使用できる定数「0」と「1」信号があります。

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 か LOW) を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 18-2 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。

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。

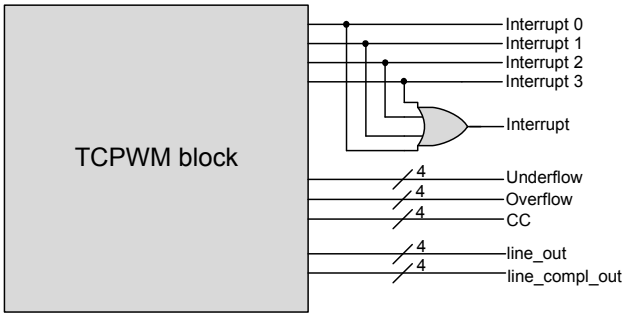
- **リロード：**リロード イベントはカウンターを初期化し、開始します。
  - カウントアップ モードではカウント レジスタ (TCPWM_CNT_COUNTER) が「0」で初期化されます。
  - カウントダウン モードではカウンターが TCPWM_CNT_PERIOD レジスタに格納された周期で初期化されます。
  - カウントアップ／ダウン モードではカウント レジスタが「1」で初期化されます。
  - 直交モードではリロード イベントは直交インデックス イベントとして機能します。インデックス／リロード イベントは完了した回転を示し、直交復号を同期化するために使用できます。
- **開始：**開始イベントはカウントを開始するために使用されます。ソフトウェアで停止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注意してください。
  - 直交モードでは開始イベントは 203 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。
- **停止：**停止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。開始イベントでカウントは再び開始します。
  - PWM モードでは停止イベントはキル イベントとなります。キル イベントではすべての PWM 出力ラインが無効になります。
- **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。
- 直交モードではエッジ検出はカウンター クロックに従ってで行われます。残りの 5 つのモードではエッジ検出はゲート制御された HFCLK で行われます。

18.2.4 出力信号

図 18-3 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 18-3. TCPWM 出力信号



18.2.4.1 トリガー条件の発生時の信号

- 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。
  - カウントが比較値と等しい
  - キャプチャ イベントが発生 : 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システム クロックの 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。

18.2.4.2 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。8 個の TCPWM からの全 8 本の割り込み出力信号の論理和が取られ、単一の割り込み出力信号を生成します。

表 18-2 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 18-2. 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 カウントが検出されるとこのビットは「1」にセットされる。クリアするには「1」を書き込む
	1	CC_MATCH	カウントがキャプチャ／比較レジスタ値と一致するとこのビットは「1」にセットされる。クリアするには「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をセットするには「1」を書き込む。読み出されると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をセットするには「1」を書き込む。読み出されると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

18.2.4.3 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注意してください (表 18-3 を参照してください)。line_out と line_compl_out はバッファの line_out_en と line_compl_out_en で有効にされます。

表 18-3. 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 初期設定値 = 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 初期設定値 = 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 初期設定値 = 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	

18.2.5 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 18-4 を参照してください。

表 18-4. TCPWM ブロックの電力モード

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
ハイバネート	ブロックの電源がオフ。コンフィギュレーション レジスタは状態がクリアされる
ストップ	ブロックの電源がオフ。コンフィギュレーション レジスタは状態がクリアされる

18.3 動作モード

表 18-5 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 18-5. 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実装。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実装。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実装
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 を実装
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 を実装
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 を実装

表 18-6 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 18-6. カウント モード コンフィギュレーション

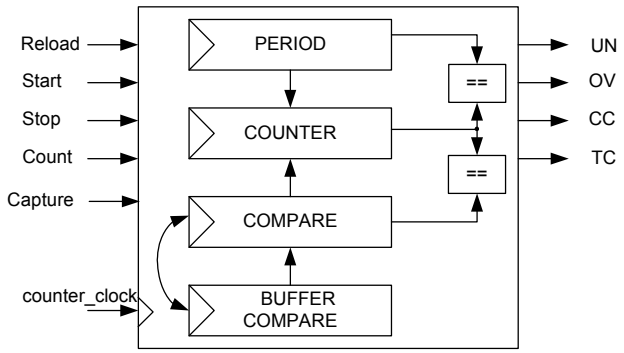
カウント モード	UP_DOWN_MODE [17:16]	説明
カウントアップ モード	00	周期に達するまでカウンターをインクリメント。カウントが周期に達するとターミナル カウント (TC) 条件が生成される
カウントダウン モード	01	「0」に達するまでカウンターを周期からデクリメント。カウンターが「0」に達すると TC 条件が生成される
カウントアップ／ダウン モード 0	10	周期に達するまでカウンターをインクリメントしてから、「0」に達するまでカウンターをデクリメント。「0」に達する時にのみ TC 条件が生成される
カウントアップ／ダウン モード 1	11	カウント アップ／ダウ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

18.3.1 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。

18.3.1.1 ブロック図

図 18-4. タイマー モード ブロック図



18.3.1.2 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。

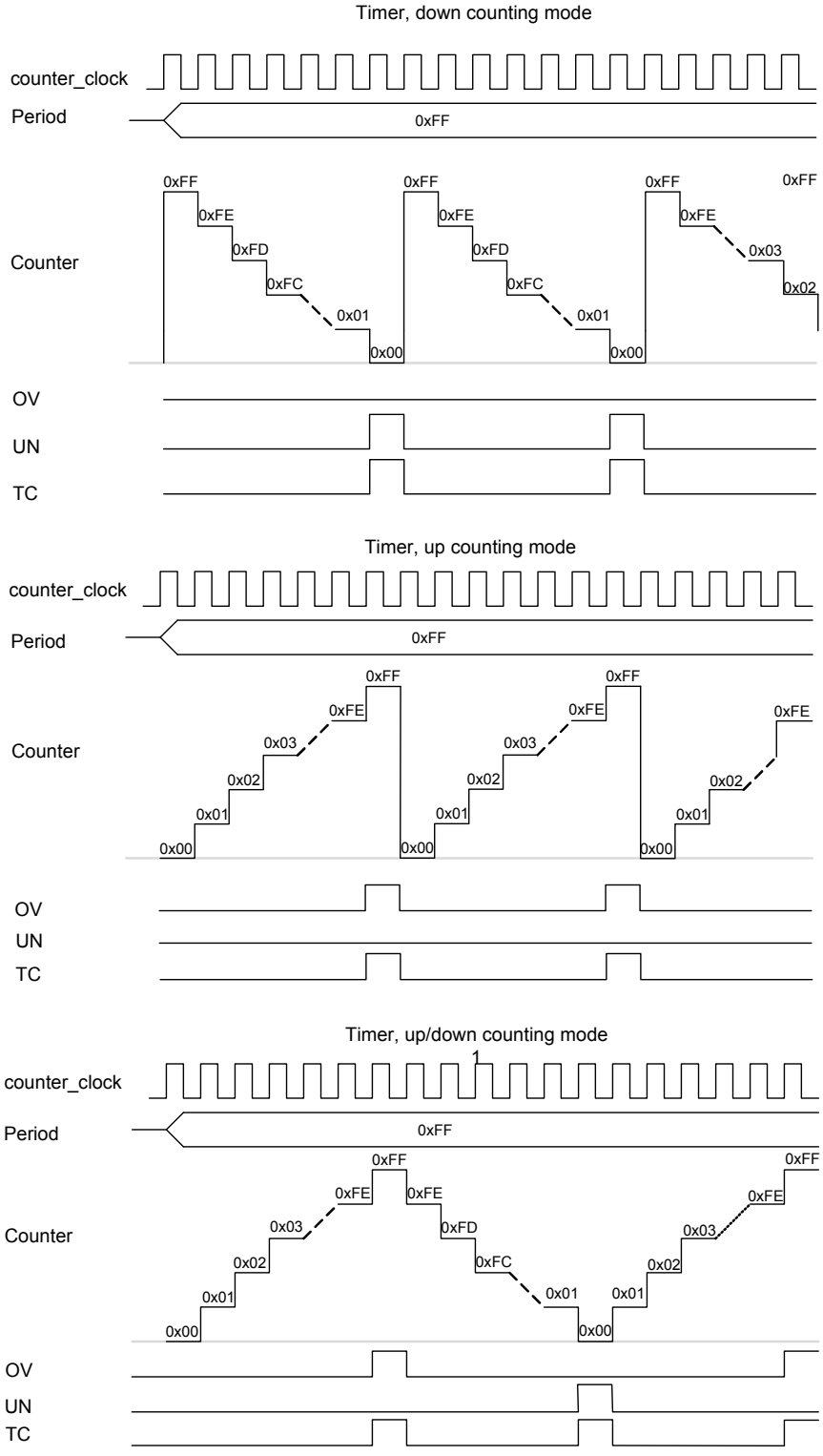
以下にタイマーの動作原理を説明します。

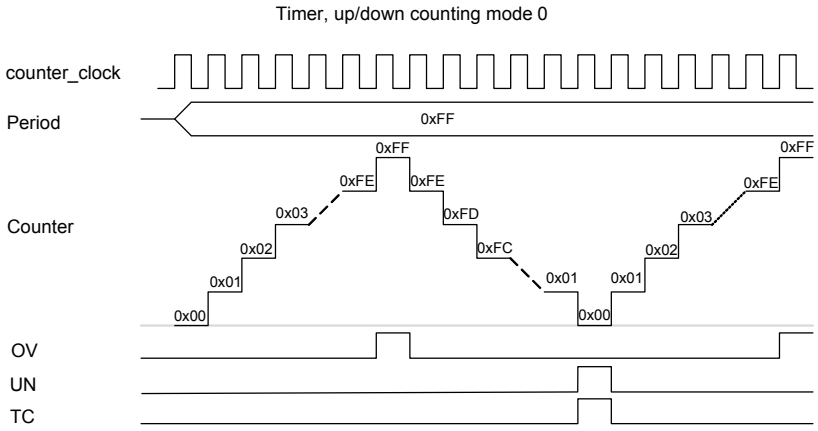
- タイマーはアップ、ダウン、アップ／ダウン カウンターです。
  - 現時点のカウント値はカウント レジスタ (TCPWM_CNTx_COUNTER) に格納されます。
    注：カウンターの動作中、このレジスタに書き込まないでください。
  - タイマーの周期は周期レジスタに格納されます。
- カウンターは以下のように異なるモードで再初期化されます。
  - カウントアップ モードではカウント レジスタは周期に達すると自動的に 0 にリロードされます。
  - カウントダウン モードではカウント レジスタが 0 に達すると周期レジスタの値にリロードされます。
  - カウントアップ／ダウン 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 18-5 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 (1 から A まで、そして 0 に戻る) となります。

図 18-5. 各カウント モードにおけるタイマーのタイミング図





注 : 196 ページのトリガー条件の発生時の信号で説明したように OV と UN 信号は HFCLK の 1 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。

18.3.1.3 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。

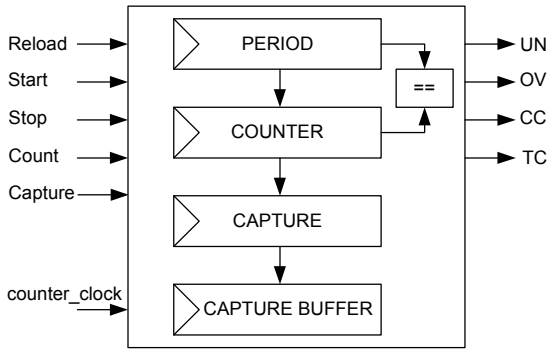
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドの対応するビットに「0」を書き込んでカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「000」を書き込んでタイマー モードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期を設定します。
4. TCPWM_CNT_CC レジスタに 16 ビット比較値を設定し、TCPWM_CNT_CC_BUFF レジスタにバッファ比較値を設定します。
5. CC 条件が発生すると値を切り替える必要がある場合 TCPWM_CNT_CTRL レジスタの AUTO_RELOAD_CC フィールドをセットします。
6. 表 18-1 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 18-6 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セットします。
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、開始、停止、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、開始、停止、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 196 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 フィールドの対応するビットに「1」を書き込んで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) で開始トリガーを提供する必要があります。

18.3.2 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。このモードは周期とパルス幅の測定に使用されます。

18.3.2.1 ブロック図

図 18-6. キャプチャ モード ブロック図



18.3.2.2 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カウントアップ、ダウン、アップ／ダウン 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。

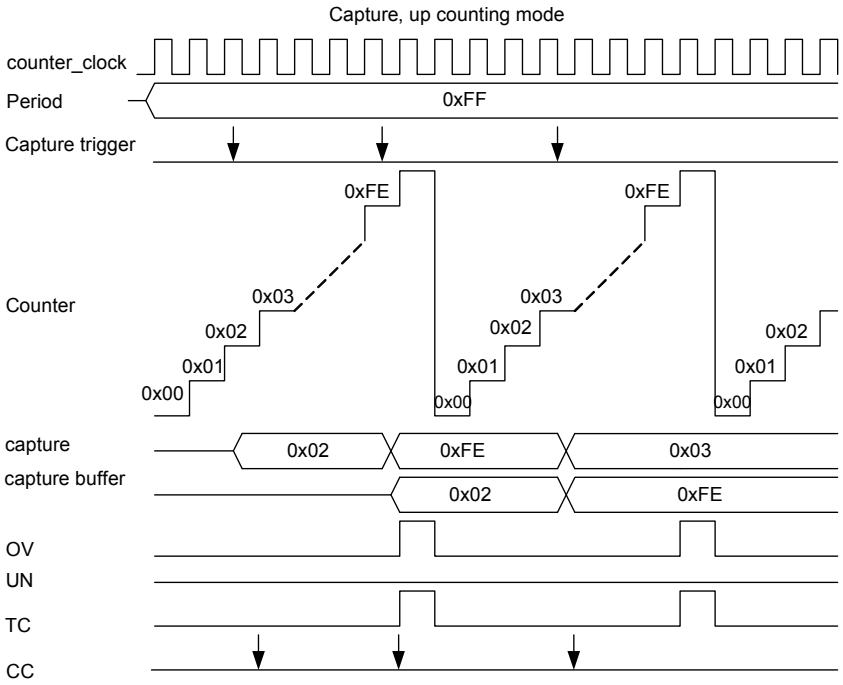
- ハードウェアまたはソフトウェアによって生成されたキャプチャ イベント中に、現時点のカウント レジスタ

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 18-7 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 18-7. 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 イベントを発生させるのはエッジまたはソフトウェアだけです。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行われます。

18.3.2.3 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 フィールドの対応するビットに「0」を書き込んで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 18-1 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 18-6 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、開始、停止、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、開始、停止、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 196 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。

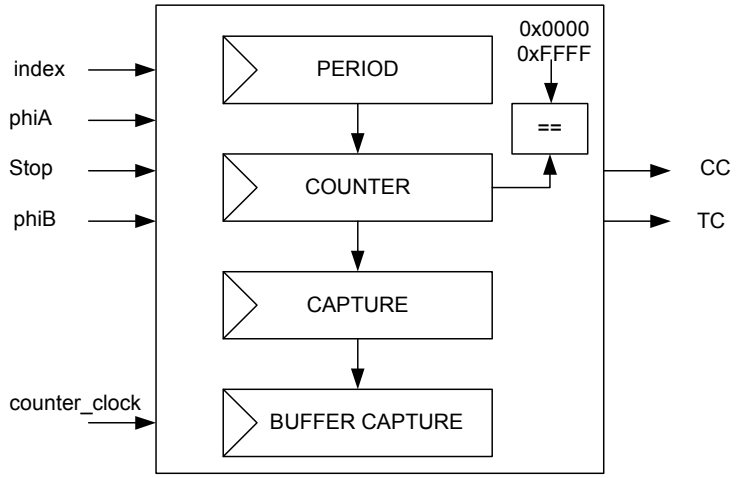
10. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドの対応するビットに「1」を書き込んでカウンターを有効にします。ハードウェア開始信号が有効になっていない場合カウンターを開始するためにファームウェア (TCPWM_CMDレジスタ) で開始トリガーを提供する必要があります。

18.3.3 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。

18.3.3.1 ブロック図

図 18-8. 直交モード ブロック図



18.3.3.2 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。

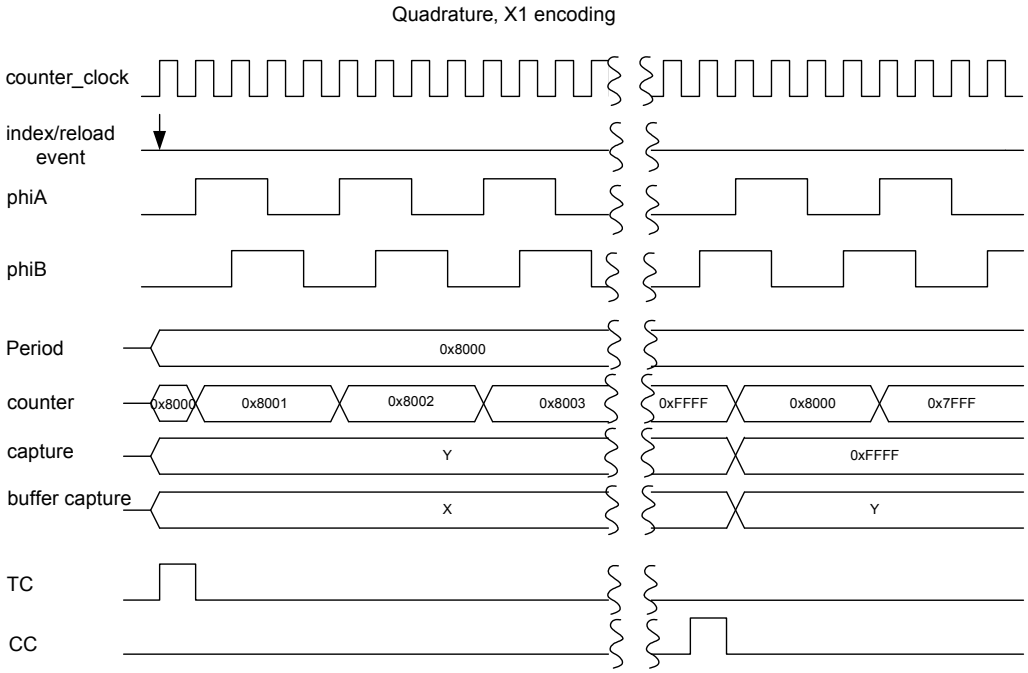
- 直交位相phiAとphiB: カウント方向はphiAとphiBの位相関係で決まります。2 つの入力信号はそれぞれデコーダーのハードウェア入力のカウント入力と開始トリガー入力に接続します。
- 直交インデックス信号 : これはハードウェア入力のリロード信号に接続します。このイベントは図 18-9 に示すように TC 条件を生成します。

  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 (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 モードでは周期を 0x8000 (中点値) に設定することが推奨されています。
- 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 条件の発生時にカウント レジスタは 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 18-9 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 カウントは生成されます。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 0x8000 に初期化されます。

図 18-9. 直交モード -X1 符号化のタイミング図

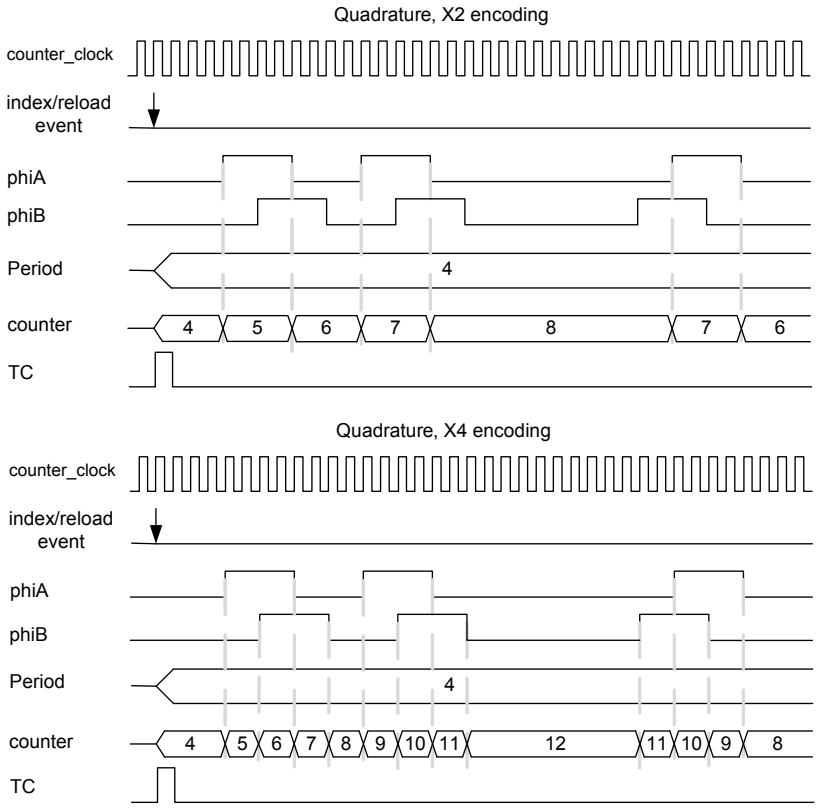


直交位相は counter_clock で検出されます。位相は 1 つの counter_clock 周期以内に 1 回以上値が変化してはいけません。

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 18-10 に X2 と X4 符号化モードでの直交動作を示します。

図 18-10. 直交モード -X2 と X4 符号化のタイミング図



18.3.3.3 直交モードの設定方法

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 フィールドの対応するビットに「0」を書き込んで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、停止) 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、停止) を発生させるエッジを選択します。
7. 必要に応じて 196 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 フィールドの対応するビットに「1」を書き込んでカウンターを有効にします。

18.3.4 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 / カウンター クロック周波数

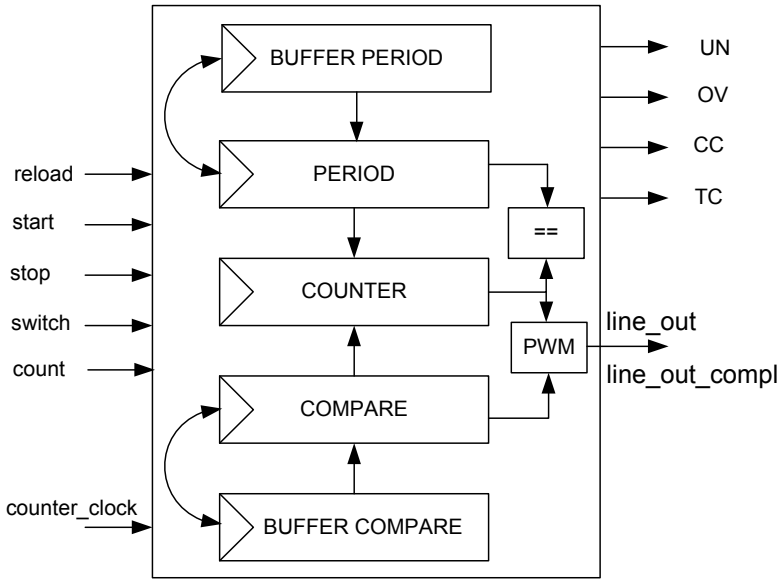
中央揃えモードの場合 : PWM 周期 = 2 × (周期 / カウンター クロック周波数)

左および右揃えモードの場合 : デューティ比 = 比較値 / 周期

中央揃えモードの場合 : デューティ比 = (周期 - 比較値)/ 周期

18.3.4.1 ブロック図

図 18-11. PWM モード ブロック図



18.3.4.2 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 18-6 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 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 18-13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次のように設定します: アンダーフロー = クリア、オーバーフロー = セット、CC = 反転。

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 18-12 にバッファ周期レジスタとバッファ比較レジスタを備えた中央揃え PWM (カウントアップ／ダウン モード 0) を示します。

図 18-12. 中央揃え PWM のタイミング図

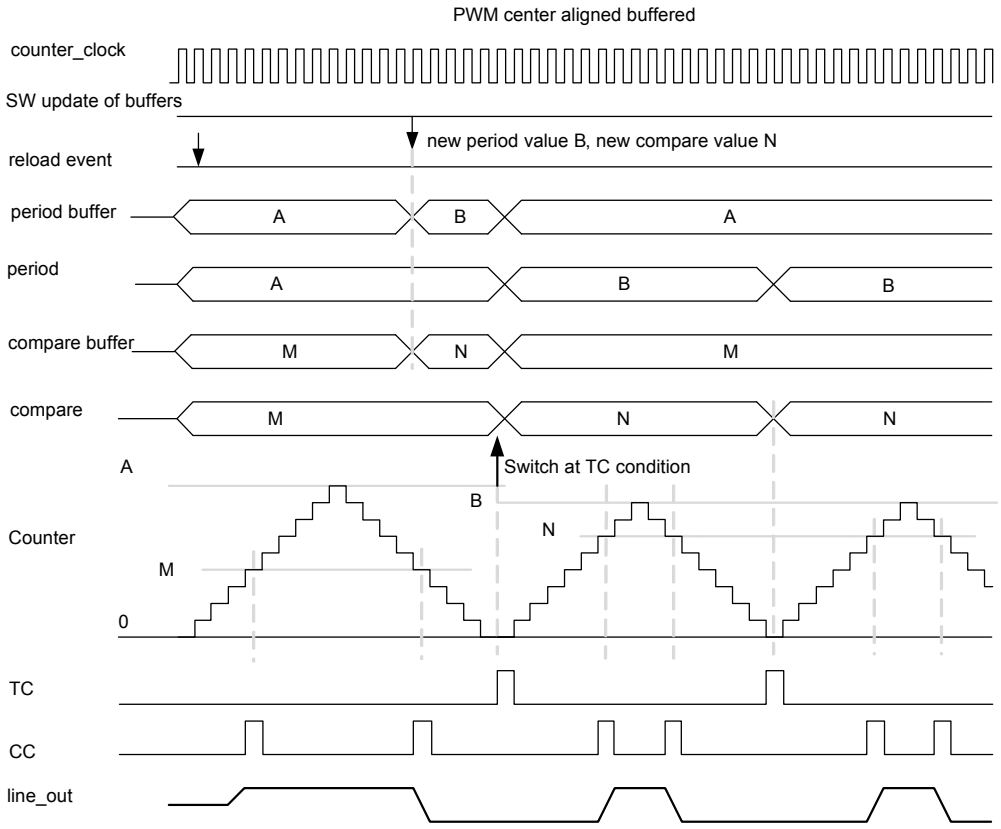
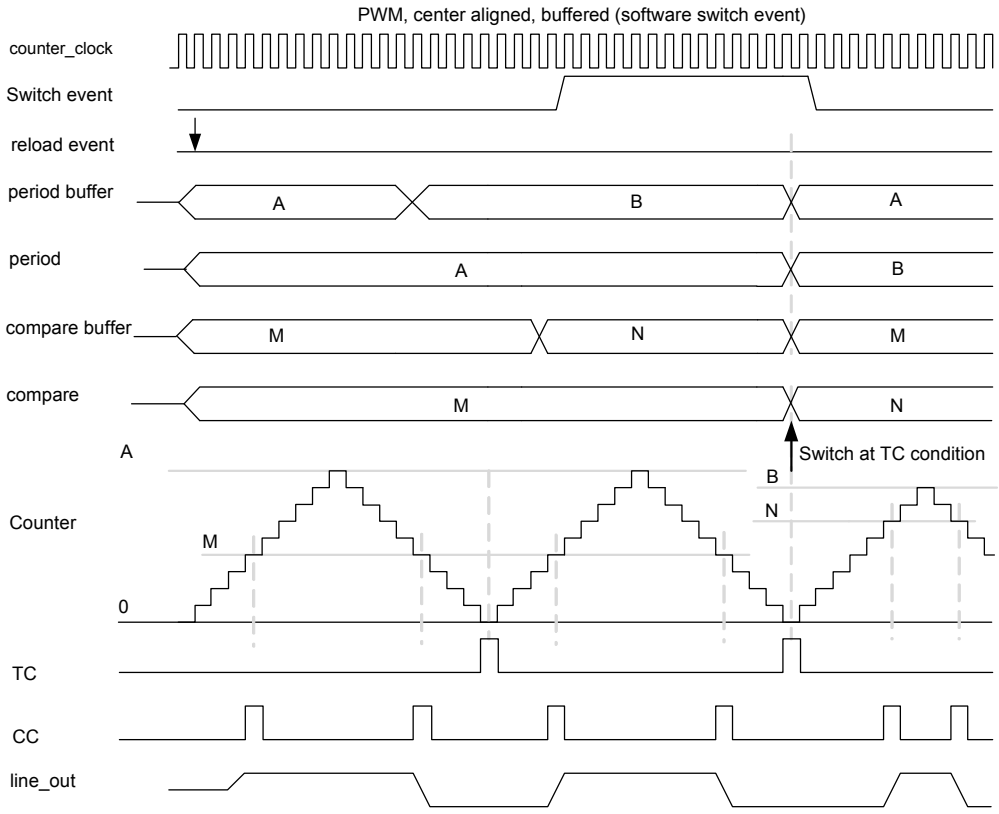


図 18-12 にソフトウェアで生成された切り替えイベントのある中央揃え PWM を示します。

- 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 パルスが遅く到着する (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した後自動的にハードウェアでクリアされることに注意してください。

図 18-13. 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



18.3.4.3 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。1 つの PWM 周期内で周期が同じであることを確認してください。
- 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 18-3 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 18-3 を参照してください。

18.3.4.4 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 18-7 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 18-7. ストップ オン キル (Stop on Kill) 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 18-8 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 18-8. 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 18-8 を参照してください)。生成された停止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。

18.3.4.5 PWM モードの設定方法

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。

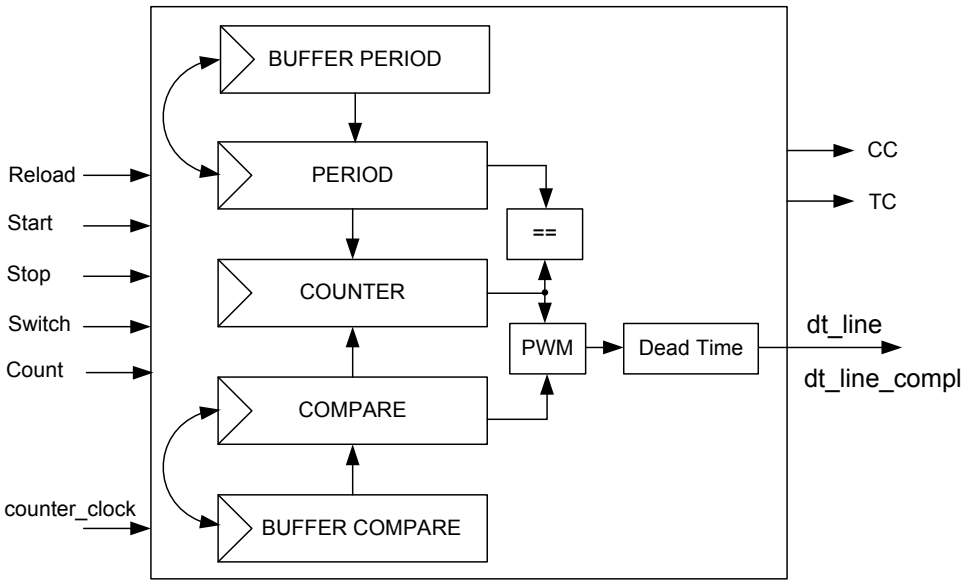
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドの対応するビットに「0」を書き込んでカウンターを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「100」を書き込んで PWM モードを選択します。
3. 表 18-1 に示すように TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込んでクロック分周を設定します。
4. TCPWM_CNT_PERIOD レジスタに 16 ビット周期を設定し、必要に応じて TCPWM_CNT_PERIOD_BUFF レジスタに切り替え用の周期を設定します。
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 18-6 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、開始、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、開始、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 196 ページの割り込みに示すように TC または CC 条件発生時の割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドの対応するビットに「1」を書き込んで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) で開始トリガーを与える必要があります。

18.3.5 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。

18.3.5.1 ブロック図

図 18-14. PWM-DT モード ブロック図



18.3.5.2 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。

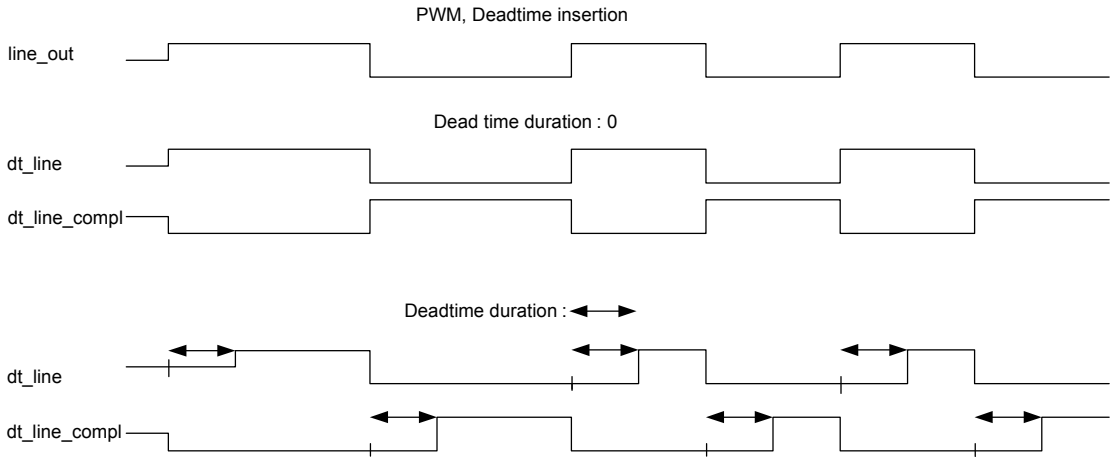
このモードは PWM モードに従い、以下の機能をサポートします。

- 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停止／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 18-15 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 18-15. デッドタイム付き PWM のタイミング図



18.3.5.3 デッドタイム付き PWM モードの設定方法

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 フィールドの対応するビットに「0」を書き込んで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 18-1 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 18-6 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 206 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、開始、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、開始、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。

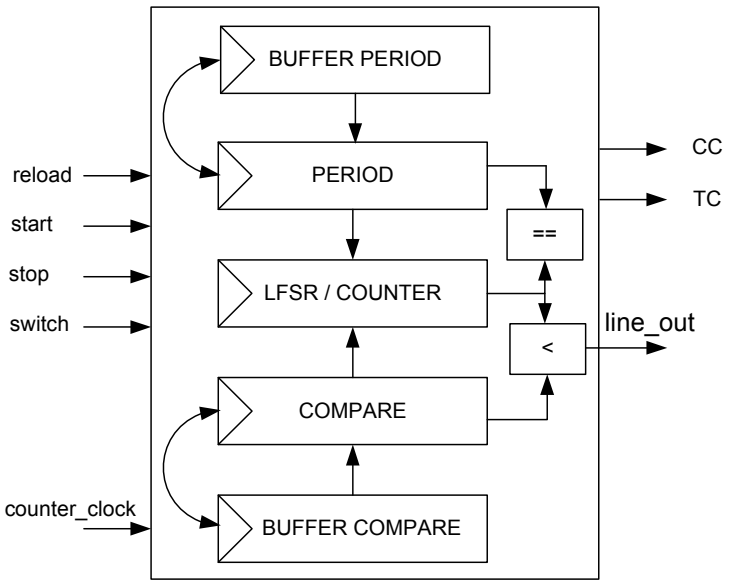
10. TCPWM_CNT_TR_CTRL2レジスタでdt_lineとdt_line_compl レジスタを制御して CC、OV、UN 条件でセット、リセットまたは反転させることができます。
11. 必要に応じて 196 ページの割り込みに示すように TC または CC 条件の発生時の割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドの対応するビットに「1」を書き込んでカウンターを有効にします。ハードウェア開始信号が有効になっていない場合カウンターを開始するためにファームウェア (TCPWM_CMDレジスタ) で開始トリガーを与える必要があります。

18.3.6 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。

18.3.6.1 ブロック図

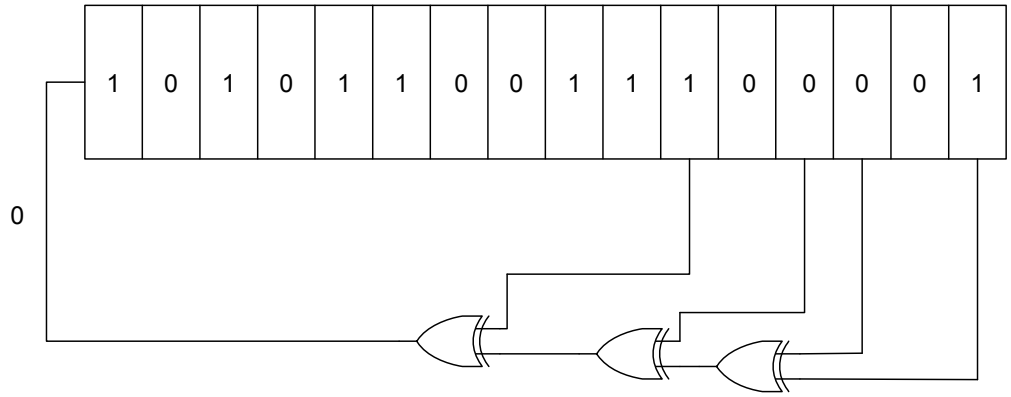
図 18-16. PWM-PR モード ブロック図



18.3.6.2 動作原理

図 18-17 に示すようにカウンター レジスタは次の多項式で LFSR を実装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注意してください。

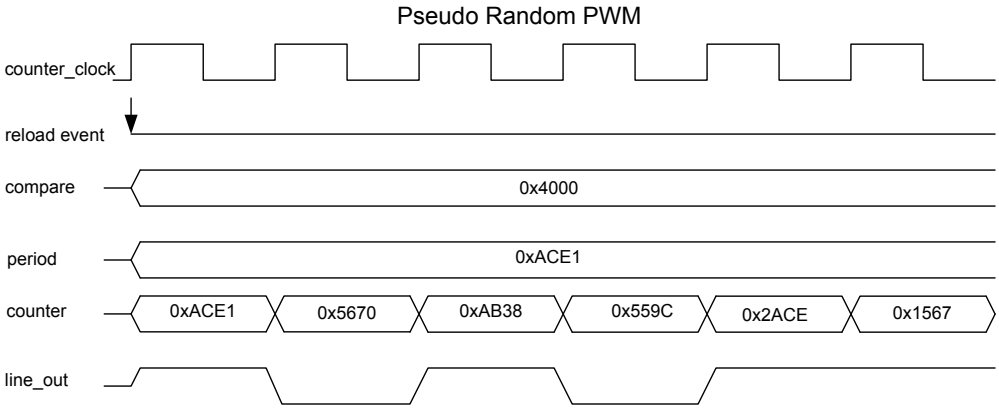
図 18-17. カウンター レジスタを用いた疑似乱数系列の生成



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 イベントは開始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で停止されます。
- 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 18-18 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ビットのみは比較レジスタ値と比較するために使用されます)。

図 18-18. 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。

18.3.6.3 疑似乱数 PWM モードの設定方法

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 フィールドの対応するビットに「 0」を書き込んで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、開始、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、開始、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 196 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 フィールドの対応するビットに「1」を書き込んでカウンターを有効にします。

18.4 TCPWM レジスタ

表 18-9. 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_CNTx_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_CNTx_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_CNTx_COUNTER	カウント レジスタ	16 ビット カウント
TCPWM_CNTx_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トと比較
TCPWM_CNTx_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、比較値を切り替える
TCPWM_CNTx_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_CNTx_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、周期を切り替える
TCPWM_CNTx_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_CNTx_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_CNTx_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_CNTx_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_CNTx_INTR_SET	割り込みセット要求レジスタ	割り込み要求レジスタの対応するビットをセット
TCPWM_CNTx_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_CNTx_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

注 : レジスタ名の「x」は TCPWM 番号を示します。たとえば TCPWM0 の割り込みマスク レジスタは TCPWM_CNT0_INTR_MASK です。

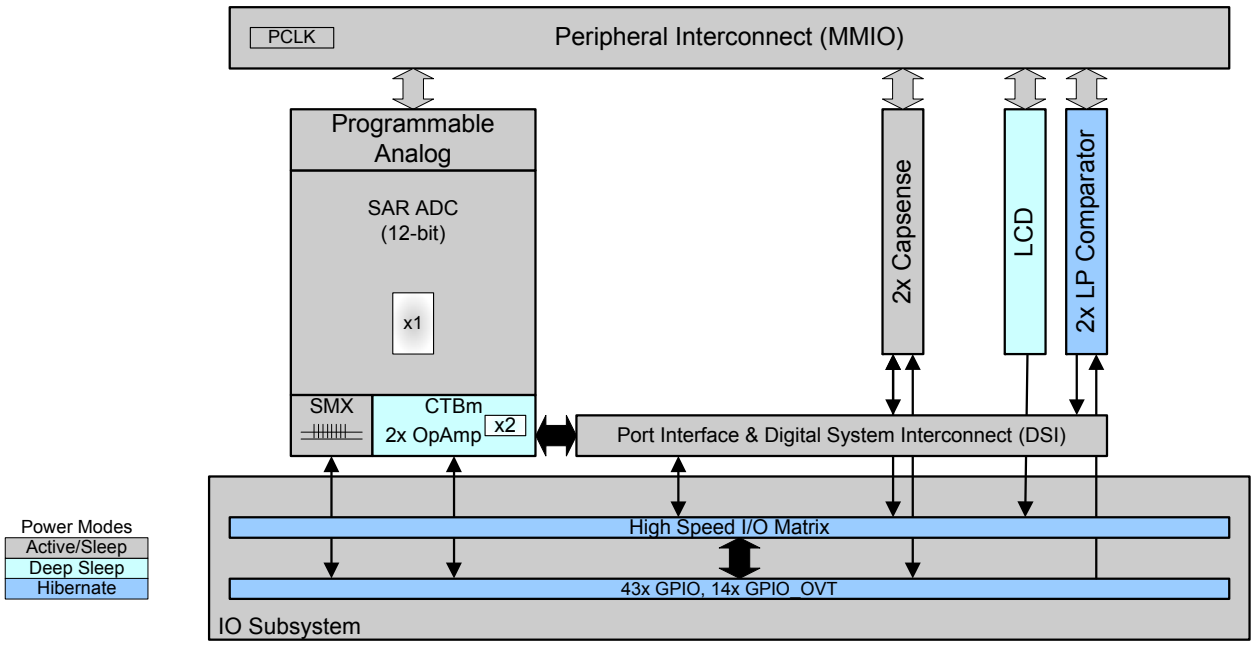
セクション E: アナログ システム

このセクションは次の章を含みます。

- 高精度リファレンス (217 ページ)
- SAR ADC (221 ページ)
- 低消費電力コンパレーター (255 ページ)
- 連続時間ブロックミニ (CTBm) (261 ページ)
- LCD ダイレクト ドライブ (271 ページ)
- CapSense (283 ページ)
- 温度センサー (293 ページ)

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



19. 高精度リファレンス

PSoC® 4 には複数のリファレンス バイアス電圧と電流を生成し、チップ全体に供給する高精度なリファレンス ブロックがあります。またこのブロックはデバイスの動作温度範囲にわたって、正確な IMO 出力周波数とエラーなしのフラッシュ読み出し／書き込み動作のために、それぞれ内部主発振器 (IMO) とフラッシュ メモリ ブロックに温度依存リファレンスも提供します。

19.1 特長

高精度リファレンス ブロックは以下の特長があります。

- 1.024V および 2.4µA リファレンスを生成するためのバンドギャップ回路
- バンドギャップ回路からの入力を使用して 1.2V、1.024V、0.8V の異なる出力電圧のレベルを生成するための調整値バッファ
- 様々なリファレンス出力の駆動能力を強化するたけでなく、相互にノイズを分離する複数の高速と低速の低消費電力バッファ
- 複数の高速と低速のカレントミラー回路
- フラッシュ メモリ用の温度依存のリファレンス電圧
- IMO 用の温度依存リファレンス電流

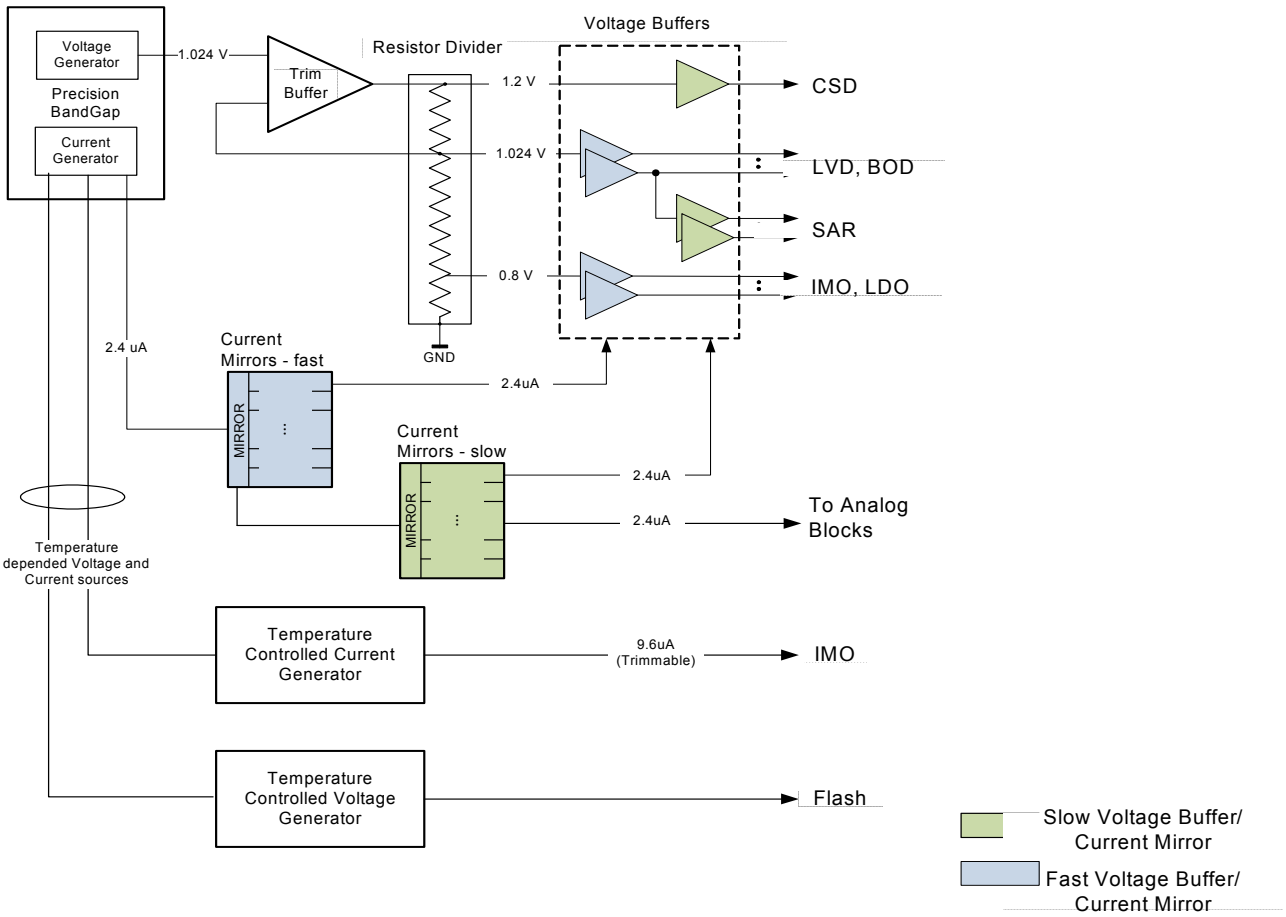
19.2 ブロック図

図 19-1 にブロック図を示します。

高精度リファレンスは主に以下のブロックから構成されます。

- 高精度なリファレンス電圧と電流を生成する高精度バンドギャップ ブロック
- 様々なアプリケーション用に異なる出力リファレンス電圧を生成し、1.024V 出力の電圧振幅を調整する調整値バッファ
- 様々なリファレンス出力の駆動能力を強化するたけでなく、相互にノイズを分離する一群の高速と低速の低消費電力バッファ
- それぞれ高速と低速のドメインでリファレンス電流の複数のコピーを作成する高速リーフ セルと低速リーフ セルのグループ
- フラッシュ システム用の温度制御電圧発生器 ブロック
- IMO 用の温度制御電流源

図 19-1. 電圧リファレンス ブロック図



19.3 動作原理

主なコンポーネントの動作原理をこの節で詳しく説明します。

19.3.1 高精度バンドギャップ

この回路は高精度リファレンス ブロックで生成されるすべてのリファレンスのソースです。これは二次近似により補正される 1.024V リファレンス電圧と 2.4uA リファレンス電流を提供します。リファレンス電圧は調整値バッファに接続され、リファレンス電流はカレントミラー回路に接続されます。

19.3.2 調整値バッファ

調整値バッファはオペアンプネットワークで、バンドギャップからの入力を使用して 3 つの異なるリファレンス (1.2V、1.024V、0.8V) を生成します。リファレンスはフィードバック ネットワークの抵抗アレイから取り出すため、出力インピーダンスが高くなります。したがってリファレンスを駆動するためにバッファが必要となります。

19.3.3 低消費電力バッファ

PSoC 4 は 2 つのグループ (高速と低速) に分けられる複数の低消費電力バッファがあります。これらのバッファは調整値バッファ回路から入力を取り込み、供給先のブロックを駆動します。高速なバッファは 9us 以内に電圧の最終値 (1% の誤差) に達することができます。低速バッファは 40us 以内に最終値に達することができます。複数のバッファは低いリファレンスラインの静電容量を確保します。したがってセトリング時間を短縮します。高速電圧バッファはシステムの起動に重要なブロックに供給されるリファレンスのために使用されます。これらのブロックは IMO、フラッシュ、低ドロップアウト (LDO) レギュレータ、低電圧検出 (LVD)、ブラウンアウト検出 (BOD) 回路です。

高速バッファの出力は低速バッファを駆動します。これは起動に関連しないブロックによる負荷の増加を、高速バッファが駆動するブロックから分離することを確保します。低速バッファは SAR ADC や CapSense CSD などの機能ブロックを駆動します。

高速バッファは常にバンドギャップ ブロックと共に有効にされます。低速バッファは PWR_BG_CONFIG レジスタの

VREF_EN ビットを使用して、ユーザーに個別に有効または無効にされます。

19.3.4 カレントミラー

カレントミラー回路はバンドギャップ回路の入力から 2.4uA リファレンスの複数のコピーを生成するために使用されます。電圧バッファと同様に、高速と低速カレントミラーの 2 つのタイプがあります。高速カレントミラー回路は電圧の最終値 (1% の誤差) に達するまで 9us のセトリング時間があります。低速カレントミラー回路は 40us のセトリング時間があります。高速カレントミラーは高速電圧バッファにバイアスを提供するために使用されます。低速カレントミラーの出力は SAR、CTBm、CSD、LPCOMP などのアナログブロックを駆動するために使用されます。

19.3.5 温度制御電圧発生器

このブロックに生成されるバイアス信号は温度によってフラッシュメモリのリファレンスを制御します。これは高精度バンドギャップ ブロックから入力を受け取ります。温度依存リファレンス電圧はフラッシュ メモリ ブロックで生成されデバイスの動作温度範囲にわたって正常な読み出し／書き込み動作に必要であるポンプ電圧を補償します。

19.3.6 温度制御電流発生器

このブロックはデバイスの動作温度範囲にわたって IMO のクロック周波数の誤差を ±2% 以内に維持するための温度依存リファレンス電流を生成します。

表 19-1. リファレンス電圧および電流

リファレンス電圧または電流	バッファ	目標精度	用途
1.024V	高速	±2%	LVD – 外部電源の低電圧検出
1.2V	低速	±2%	Capsense リファレンス
1.024V	低速	±1%	SAR ADC
1.024V	高速	±2%	BOD – 内部電圧のブラウンアウト検出
0.8V	高速	±2%	IMO – 弛張発振器のコンパレータの閾値
0.8V	高速	±2%	LDO – V_{CCD} および V_{CCA} レギュレータ リファレンス
2.4µA	低速	±2.5%	アナログ回路用のバイアス電流 (CSD - IDAC、SAR ADC、LPCOMP、CTBm)
9.6µA	–	±5%	IMO 用のプログラム可能な温度補償機能を備える IREF

19.4 コンフィギュレーション

電源投入時、高精度リファレンス ブロックは不揮発性ラッチ (NVL) と SFLASH に保存される初期調整値に初期化されます。これらの設定は製造中にプログラムされ、フィールドでの調整は不要です。

20. SAR ADC

PSoC® 4 は 1 個の逐次比較型アナログ - デジタル変換器 (SAR ADC) を内蔵しています。SAR ADC は中度の分解能および高いデータ速度を必要とするアプリケーション向けに設計されています。以下のブロックを含みます。(ブロック図は 図 20-1):

- SARMUX
- SAR ADC コア
- SARREF
- SARSEQ

SAR ADC コアは高速 12 ビット ADC であり、サンプリング レートが PSoC 4200M では 1Msps、PSoC 4100M では 806Msps です。SAR ADC の前に SARMUX があります。これは外部ピンと内部ピン (AMUXBUS-A/-B、CTBm、温度センサー出力) を SAR ADC の 8 本のチャネルに接続することができます。SARREF は異なるリファレンス電圧の選択に使用されます。シーケンサー コントローラー SARSEQ は SARMUX と SAR ADC を制御し、CPU の介入なく有効になったチャネルに対して自動スキャンを行い、出力データの平均化など前処理タスクを実行します。

9 番目のチャネルは内部温度センターなどピンと信号の不定期で偶発的なサンプリングのためにファームウェアで使用する注入チャネルです。

各チャネルの結果は二重バッファされ、スキャンの完了で割り込みを生成することもできます。あるいは CPU を介さず後処理をするために、データをプログラマブルなデジタル ブロック (UDB) に接続することもできます。シーケンサーは、割り込みをアサートできるオーバーフロー、衝突、飽和エラーを警告するよう設定できます。

柔軟性をもっと高めるために、UDB またはファームウェアにより SARMUX 内のアナログ スイッチを含むほとんどのアナログ スイッチを制御することが可能です。これにより UDB またはファームウェアで代替シーケンサーを実装することができます。

20.1 特長

- デバイスの全電源電圧範囲で動作可能
- 最大サンプリング レートが PSoC 4200M では 1Msps、PSoC 4100M では 806Msps
- 個別に設定可能なチャネル 8 本と注入チャネル 1 本
- 各チャネルの持つ特長 :
  - 外部ピンまたは内部信号からの入力 (AMUXBUS ／ CTBm ／温度センサー)
  - プログラム可能な取得時間
  - 選択可能な 8、10、12 ビット分解能
  - シングルエンド入力または差動入力
  - 平均化
  - 結果の二重バッファリング
  - 結果が左または右アライメント
- スキャンはファームウェア、タイマー、ピンまたは UDB でトリガー
  - ワンショット、定期または連続モード
- ハードウェアによる平均化に対応
  - 一次の蓄積
  - 2 ～ 256 (2 のべき乗) 回数のサンプル平均化
- 結果が 16 ビット符号拡張値で表される
- 選択可能な電圧リファレンス

- 内部 V_{DDA} と $V_{DDA}/2$ リファレンス
- 内部 1.024V リファンレス (バッファあり)
- 外部リファレンス

■ 割り込みの生成
- 完了したスキャン変換
- 各チャネルで飽和検出とオーバーレンジ (設定可能) 検出
- スキャン結果オーバーフロー
- 衝突検出

■ 注入チャネルの設定
- ファームウェアでトリガー
- 2 つのスキャン シーケンス間に挟まれる (テールゲート)
- 選択可能なサンプリング時間、分解能、シングルエンド入力か差動入力、平均化

■ CPU の負荷を軽減するためのプログラマブルなデジタル ブロックでデータを処理するオプション

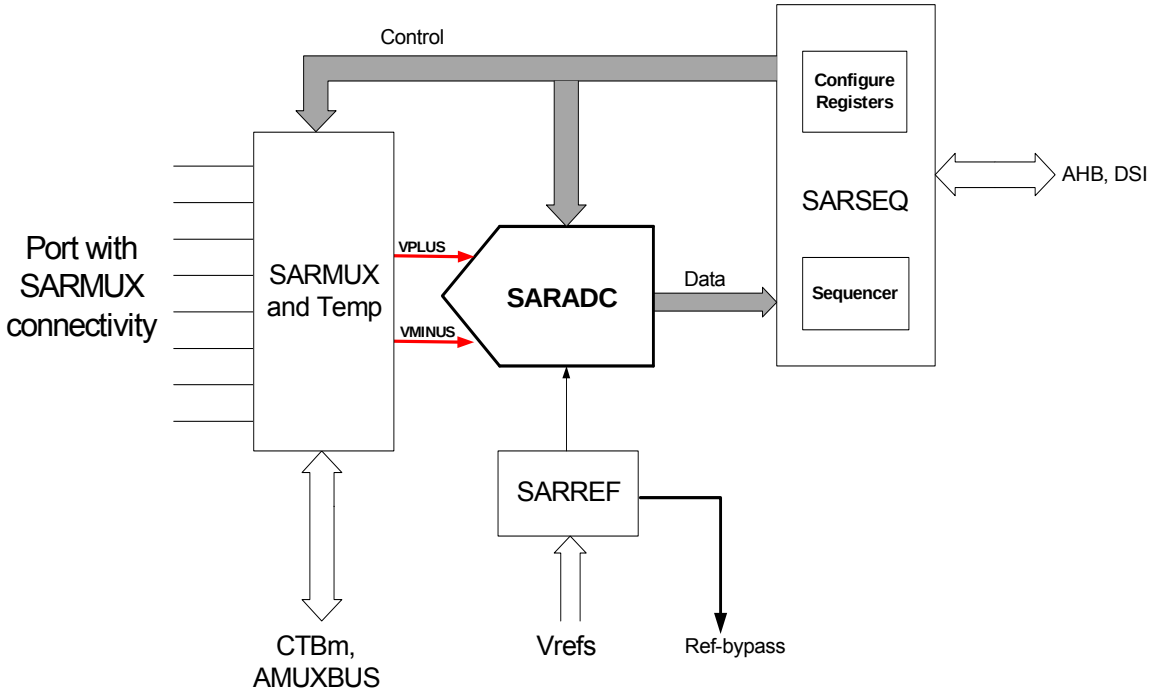
■ プログラマブルなデジタル ブロックでスイッチを制御するオプション

■ プログラマブルなデジタル ブロックで SAR ADC とスイッチを制御するオプション
- 代替 SAR シーケンサーを実装
- 1Msps を達成可能

■ 低消費電力モード
- ADC コアおよびリファレンス電圧モジュールは別に低消費電力モードがある

20.2 ブロック図

図 20-1. ブロック図



20.3 動作原理

本節で以下のことを説明します。

- 各ブロックの概要: SAR ADCコア、SARMUX、SARREF、SARSEQ
- SAR ADCシステム リソース: 割り込み、低消費電力モード、SAR ADC の状態
- システム動作モード
  - レジスタ モード
  - DSI モード
- コンフィギュレーション例

20.3.1 SAR ADC コア

SAR ADC コアの特長は以下の通りです。

- 完全な差動アーキテクチャ、シングルエンド モードにも対応
- 12 ビット分解能および選択可能な予備分解能 : 8 ビットまたは 10 ビット
- プログラム可能な取得時間
- プログラム可能な電力モード (最大、1/2、1/4)
- シングルと連続変換モードに対応

20.3.1.1 シングルエンドと差動モード

PSoC 4 SAR ADC はシングルエンドと差動モードで動作できます。完全な差動アーキテクチャで設計され、差動動作モードで 12 ビット精度を実現するよう最適化されています。$–V_{REF}$ ～ $+V_{REF}$ の差動入力の全範囲出力 (0 ～ 4095) を提供します。SAR ADC は－入力を固定することでシングルエンド モードに設定することができます。差動またはシングルエンド モードはチャネル コンフィギュレーション レジスタ SAR_CHANx_CONFIG で設定できます。

－入力のシングルエンド モード オプションは V_{SSA}、V_{REF}、SARMUX を介した 8 本のピンのいずれかからの外部入力です。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。このモードはグローバル コンフィギュレーション レジスタ SAR_CTRL で設定されます。Vminus が SARMUXピンに接続されるとシングルエンド モードは差動モードに相当します。しかし各差動ペアの奇数ピンが共通の代替グランドに接続すると変換の分解能は 11 ビットとなります。理由は測定された信号値 (SARMUX.vplus) がグランド電圧を下回ることができないからです。

12ビット シングルエンド変換を行うにはV_{REF}をSAR ADCの－入力に接続する必要があり、入力範囲は 0 ～ 2 × V_{REF} です。

温度センサーはシングルエンドでのみ使用でき、SAR_CTRL [11:9] を 0 で上書きすることに注意してください。温度センサーは差動変換に対応しておらず、結果は不定となります。

20.3.1.2 入力範囲

すべての入力はV_{SSA}～V_{DDA}の範囲である必要があります。また入力電圧範囲は V_{REF} によって制限されています。－入力の電圧が Vn で、ADC リファレンス電圧が V_{REF} である場合、＋入力の電圧は Vn ± V_{REF} となります。これはシングルエンドと差動両方のモードに適用されます。

Vn ± V_{REF} は V_{SSA} ～ V_{DDA} の範囲内にある必要があることに注意してください。たとえば－入力が V_{SSA} に接続した場合、＋入力の範囲は $–V_{REF}$ ～ V_{REF} ではなく 0 ～ V_{REF} です。理由は信号が V_{SSA} を下回ることができないからです。＋入力信号がV_{SS}を下回ることができないためADC範囲の半分にのみ対応可能で、実効的に 11 ビットの結果を生成します。

20.3.1.3 結果データのフォーマット

結果データのフォーマットは 2 つの側面で設定可能です。

- 符号付き／符号無し
- 左／右アライメント

結果が符号付きの場合変換の最上位ビットは 16 ビットまでの符号拡張に使用されます。符号無しの変換の場合結果は 16 ビットまでの 0 拡張です。SAR_SAMPLE_CTRL [3:2] で差動またはシングルエンド変換に設定することができます。

サンプル値は結果レジスタの 16 ビット内で右アライメントまたは左アライメント です。初期設定ではデータはデータ [11:0] 内で右アライメントで、必要に応じて 16 ビットまでの符号拡張があります。分解能が低く、右アライメントの場合下位ビットが 0 にされます。

以下に符号付きと符号無し、左アライメントと右アライメント、12、10、8 ビットの変換の結果データ フォーマットを示します。

表 20-1. 結果データのフォーマット

アライメント	符号付き／符号無し	分解能	結果レジスタ															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
右	符号無し	12	–	–	–	–	11	10	9	8	7	6	5	4	3	2	1	0
		10	–	–	–	–	–	–	9	8	7	6	5	4	3	2	1	0
		8	–	–	–	–	–	–	–	–	7	6	5	4	3	2	1	0

表 20-1. 結果データのフォーマット

アライメント	符号付き／符号無し	分解能	結果レジスタ															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
右	符号付き	12	11	11	11	11	11	10	9	8	7	6	5	4	3	2	1	0
		10	9	9	9	9	9	9	9	8	7	6	5	4	3	2	1	0
		8	7	7	7	7	7	7	7	7	7	6	5	4	3	2	1	0
左	–	12	11	10	9	8	7	6	5	4	3	2	1	0	–	–	–	–
		10	9	8	7	6	5	4	3	2	1	0	–	–	–	–	–	–
		8	7	6	5	4	3	2	1	0	–	–	–	–	–	–	–	–

20.3.1.4 －入力選択

－入力接続の選択は電圧範囲、SNR、有効な分解能に影響を与えます (表 20-2 を参照してください)。シングルエンド モードでは SAR ADC の－入力は V_{SSA}、V_{REF} または SARMUX を介して 8 本のピンのいずれかに接続できます。

表 20-2. －入力選択の比較表

シングルエンド／差動	符号付き／符号無し	SARMUX Vminus	SARMUX Vplus 範囲	結果レジスタ	最大 SNR
シングルエンド	該当なし[a]	V_{SSA}	$+V_{REF}$ V_{SSA} = 0	0x7FF 0x000	より良い
シングルエンド	符号無し	V_{REF}	+2 × V_{REF} V_{REF} V_{SSA} = 0	0xFFF 0x800 0	良い
シングルエンド	符号付き	V_{REF}	+2 × V_{REF} V_{REF} V_{SSA} = 0	0x7FF 0x000 0x800	良い
シングルエンド	符号無し	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0xFFF 0x800 0	最も良い
シングルエンド	符号付き	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0x7FF 0x000 0x800	最も良い
差動	符号無し	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0xFFF 0x800 0	最も良い
差動	符号付き	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0x7FF 0x000 0x800	最も良い

a. Vminus が V_{SSA} に接続されたシングルエンド モードでは変換の分解能が実効的に 11 ビットです。理由はすべての PSoC 4 で電圧が V_{SSA} を下回ることができないからです。そのためグローバル コンフィギュレーション ビット SINGLE_ENDED_SIGNED (SAR_SAMPLE_CTRL[2]) は無視され、結果は常に 0x000 ～ 0x7FF の範囲内です。

12ビット シングルエンド変換を行うにはV_{REF} を SAR ADC の－入力に接続する必要があり、入力範囲は 0 ～ 2 × V_{REF} です。

Vminus が SARMUX を介してピンに接続したシングルエンド変換は電気的に差動モードに相当します。しかし各差動ペアの奇数ピンが共通の代替グランドに接続すると変換の分解能は 11 ビットとなります。理由は測定された信号値 (SARMUX.vplus) がグランド電圧を下回ることができないからです。

20.3.1.5 分解能

PSoC 4 は 12 ビット分解能 (初期設定) および選択可能な予備分解能 : チャネルごとに 8 ビットまたは 10 ビット

分解能は以下のように変換時間に影響を与えます。

変換時間 (sar_clk) = 分解能 (ビット) + 2

取得時間と変換時間の和 (sar_clk) = 取得時間 + 分解能 (ビット) + 2

12 ビット変換で、取得時間 = 4 の場合、18 sar_clk が必要となります。たとえば sar_clk が 18MHz の場合変換は 18 sar_clk を要し、変換速度は 1Msps となります。分解能が低いほど変換速度は高くなります。

20.3.1.6 取得時間

取得時間は SAR ADC 内のサンプル ホールド (S/H) 回路がセトリングされるのに要する時間です。取得時間が経過した後入力信号ソースは SARADC コアから切断され、S/H 回路の出力は変換に使用されます。各チャネルはグローバル コンフィギュレーション レジスタ SAR_SAMPLE_TIME01 と SAR_SAMPLE_TIME23 で定義された 4 ～ 1023 SAR クロック サイクルの 4 つの取得時間オプションの 1 つを選択できます。

図 20-2. 取得時間

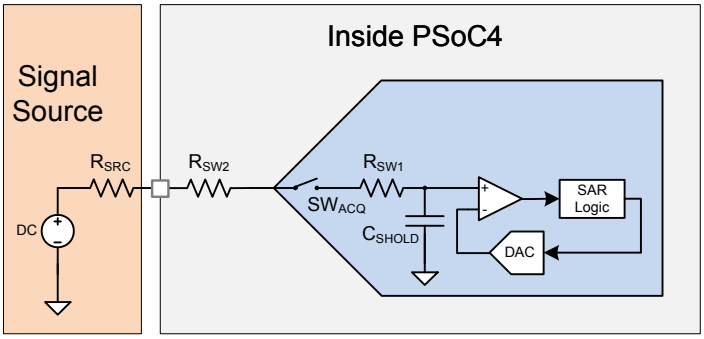


図 20-2 に示すように取得時間は配線経路の抵抗を介して ADC の内部ホールド コンデンサを充電するのに十分である必要があります。取得時間の推奨値 :

$t_{ACQ} >= 9 \times (R_{SCR} + R_{SW2} + R_{SW1}) \times C_{SH}$

ここで、

C_{SH} = 約 10pF

$R_{SW2} + R_{SW1}$ = 約 500 ～ 1000Ω、配線経路に応じる (詳細は 226 ページのアナログ配線を参照してください)。

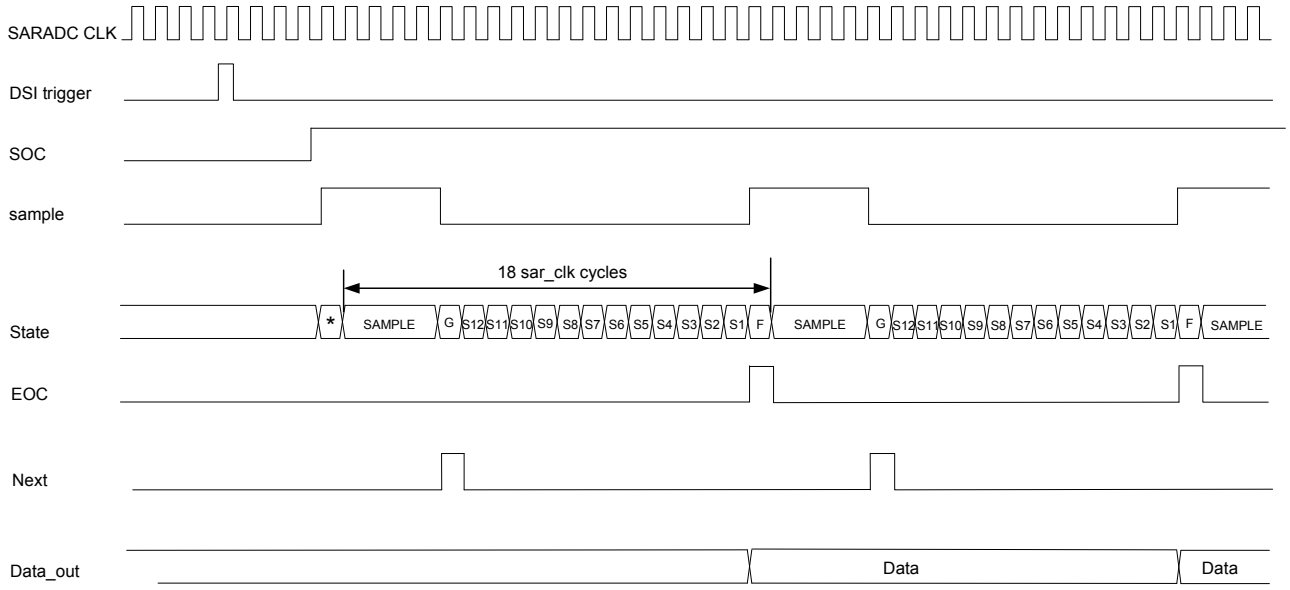
R_{SRC} = 信号ソースの直列抵抗

20.3.1.7 SAR ADC クロック

SAR ADC クロック周波数は PSoC4200M では 1MHz ～ 18MHz で、PSoC 4100M では 1MHz ～ 14.5MHz です。クロック周波数はクロック分周器により IMO から生成されます。SAR ADC は分数分周器をサポートしていないことに注意してください。PSoC 4200M で 1Msps のサンプル速度を達成するために 18MHz SAR ADC クロックが必要となります。このためにはシステム クロック (IMO) を 48MHz に代わり 36MHz に設定する必要があります。PSoC 4100M で 806ksps のサンプル速度を達成するために IMO を 29MHz に設定する必要があります。取得時間が 4 クロックの 12 ビット ADC 変換は完了するのに 18 クロックを要します。10 ビットと 8 ビット変換それぞれは 16 と 14 クロックを要します。

20.3.1.8 SAR ADC タイミング

図 20-3. SAR ADC タイミング



タイミング図に示すように変換の開始 (SOC) の前にsar_clk遅延があります。12 ビット分解能の変換は 14 クロックを要します (1ビットは1 sar_clk、GとFステートはもう2 sar_clkを要します)。初期設定では取得時間が 4 sar_clk であるため、ADC の取得と変換に 18 sar_clk クロック サイクルを要します。サンプル (開通) 後に次のパルス (dsi_sample_done) が出力されます。この時 SARMUX は別のピンと信号に接続できます。これはシーケンサー コントローラーで自動的に行われます (詳細は 234 ページの SARSEQ を参照してください)。

20.3.2 SARMUX

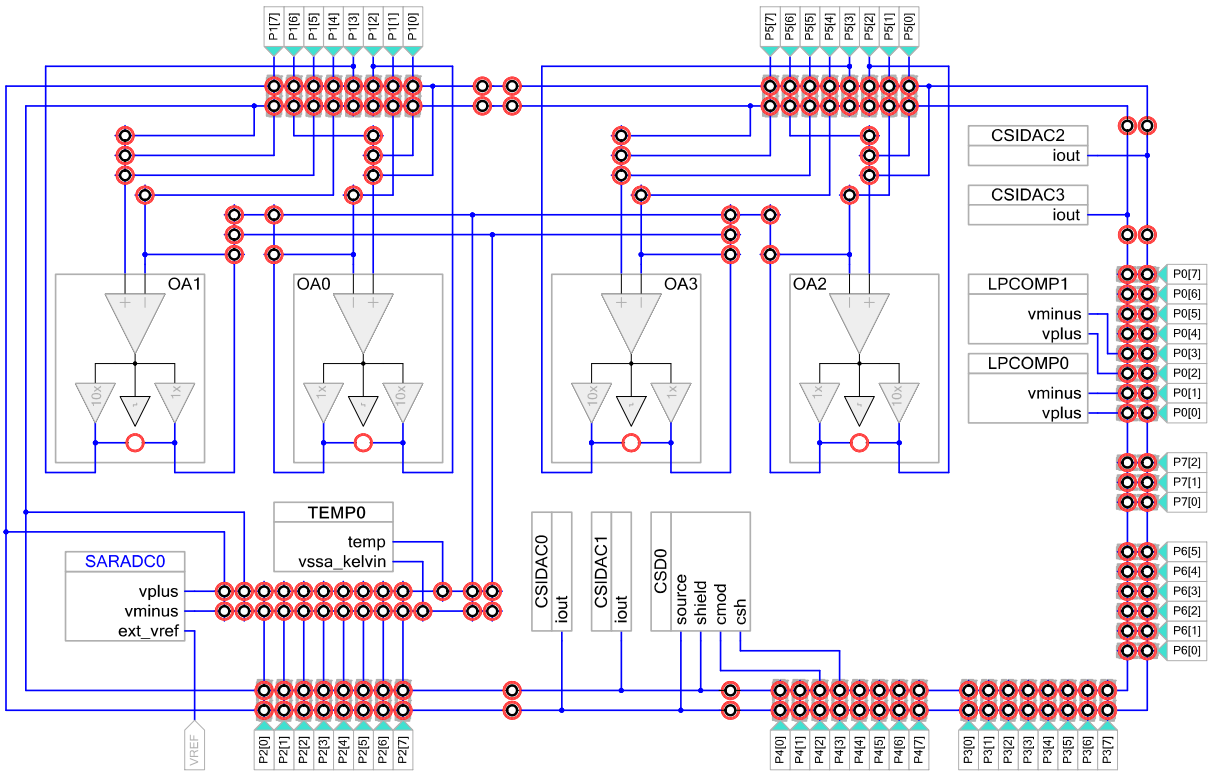
SARMUXはプログラム可能な専用アナログ マルチプレクサです。SARMUX の特長は以下の通りです。

- スイッチのオン抵抗 : 600Ω (最大値)
- 内部温度センサー
- シーケンサー コントローラー ブロック (SARSEQ)、ファームウェアまたは UDB で制御
- 内部チャージ ポンプ :
  - V_{DDA} < 4.0V の場合、スイッチ抵抗を低減させるためにチャージ ポンプはオン
  - V_{DDA} >= 4.0V の場合、チャージ ポンプはオフで、V_{DDA} を出力
- 複数の入力 :
  - ピンからのアナログ信号 (ポート 2)
  - 温度センサー出力
  - sarbus0/1 を介した CTBm 出力 (1Msps のサンプル速度を達成するのに十分ではない)
  - AMUXBUS_A/B (1Msps のサンプル速度を達成するのに十分ではない)

20.3.2.1 アナログ配線

SARMUXにはSARSEQブロック (シーケンサー コントローラー)、ファームウェアまたは DSI が制御する多くのスイッチがあります。シーケンサーと DSI はハードウェア制御方式であり、SAR_MUX_SWITCH_HW_CTRL レジスタのハードウェア制御ビットでマスクできます。各種の制御方式はスイッチに対する異なる制御機能を持っています。図 20-4 を参照してください。

図 20-4. SARMUX スイッチおよび制御機能



シーケンサー制御 : スイッチは SARSEQ ブロック内のシーケンサーで制御されます。各チャネルのアナログ配線の設定後、CPU の介入なくラウンドロビン方式でマルチチャネル自動スキャンを有効にします。どのスイッチもシーケンサーで制御できるわけではありません (図 20-4 を参照してください)。関連するレジスタは SAR_CHANx_CONFIG、SAR_MUX_SWITCH0、SAR_CTRL、SAR_MUX_SWITCH_HW_CTRL です。詳細なコンフィギュレーションはレジスタ モードにあります (246ページのファームウェア アナログ配線を参照してください)。

ファームウェア制御 : プログラム可能なレジスタは VPLUS/VMINUS 接続を直接定義します。SARMUX ですべてのスイッチを制御することができます (図 20-4 を参照してください)。たとえばファームウェア制御では、2 本の隣接したピンだけでなく (シーケンサー制御とは異なって) 任意の 2 本のピンまたは信号間の差動測定を実行することが可能です。ただしマルチチャネルの取得には CPU の介入が必要となります。関連するレジスタは SAR_MUX_SWITCH0、SAR_MUX_SWITCH_HW_CTRL、SAR_CTRL です。詳細なコンフィギュレーションはレジスタ モードにあります (246 ページのファームウェア アナログ配線を参照してください)。

DSI 制御 : スイッチは UDB からの DSI 信号で制御されます。UDB はカスタマイズ設計の二次シーケンサーとして機能できます。DSI はほとんどのスイッチを制御できます。そのためファームウェア制御と同じように任意の 2 本のピンまたは信号間で差動測定を実行することができます。詳細なコンフィギュレーションは DSI モードにあります (242 ページの SARMUX アナログ配線を参照してください)。

20.3.2.2 アナログ相互接続

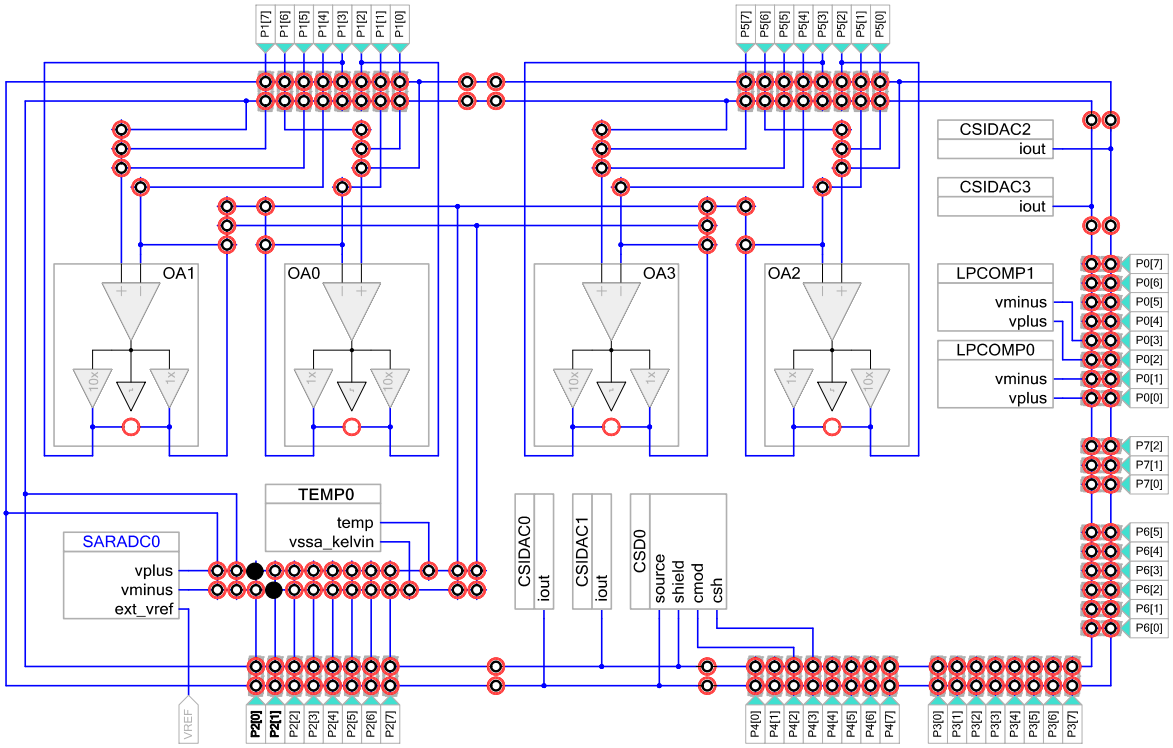
PSoC 4 のアナログ相互接続は非常に柔軟です。SAR ADC は SARMUX を介して外部ピンと内部信号を含む複数の入力に接続できます。たとえば CTBm など隣接したブロックに接続できます。また AMUXBUS_A/B を介してポート 2 以外のピンに接続できますが、スキャン性能が低下します (寄生カップリングが高くなり、セトリングされるまでの RC 時間が長くなります)。

アナログ相互接続を深く理解できるように以下のケースを説明します。

外部ピンからの入力

図 20-5 に SARMUX に対応している GPIO を差動ペア (Vpuls/Vminuse) としてスイッチを介して SAR ADC に接続する方法を示します。2 個のスイッチはシーケンサー、ファームウェアまたは DSI で制御できます。ピンは隣接したペアとして配置されています。たとえば SARMUX ポート P2[0] と P2[1]、P2[2] と P2[3] などです。P2[1] と P2[2] など一対になっていないピンを差動ペアとして使用する必要がある場合、シーケンサーの代わりにファームウェアまたは DSI を使用してください。

図 20-5. 外部ピンからの入力

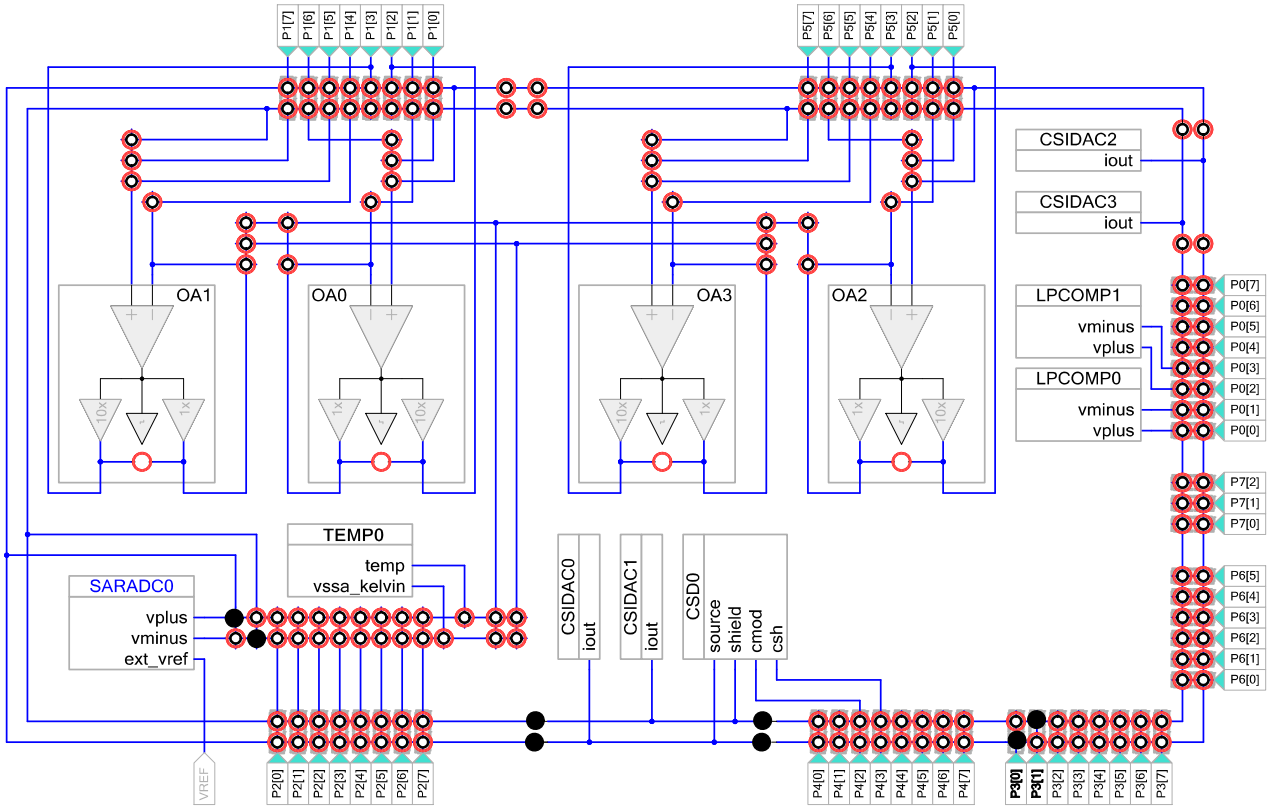


アナログバスからの入力 (AMUXBUS_A/B)

図 20-6 に SARMUX 接続に対応していない 2 本のピンを差動ペアとして ADC に接続する方法を示します。AMUXBUS_A と AMUX-BUS_B の 2 本のピンに追加のスイッチを接続してから、AMUXBUS_A と AMUXBUS_B を ADC に接続します。

追加のスイッチはスキャン性能を低下させる (寄生カップリングが高くなり、セトリングされるまでの RC 時間が長くなる) ため、1Msps のサンプル速度を達成するのに十分ではありません。これは外部信号に推奨されておらず、その代わりに専用 SARMUX ポートを使用してください (可能である場合)。

図 20-6. アナログバスからの入力



sarbus を介した CTBm 出力からの入力

SAR ADC は sarbus 0/1 を介して CTBm 出力に接続できます。図 20-7 にオペアンプ (フォロワーとして設定される) 出力をシングルエンド SAR ADC に接続する方法を示します。一端子は V_{REF} に接続されます。図 20-8 に 2 本のオペアンプ出力を差動ペアとして SAR ADC に接続する方法を示します。オペアンプ出力を sarbus 0/1 に接続してから SAR ADC 入力を sarbus 0/1 に接続する必要があります。複数のスイッチがあるため 1Msps のサンプル速度を達成するのに十分ではありません。しかし内蔵オペアンプは多くのアプリケーションで付加価値があります。

図 20-7. sarbus を介した CTBm 出力からの入力

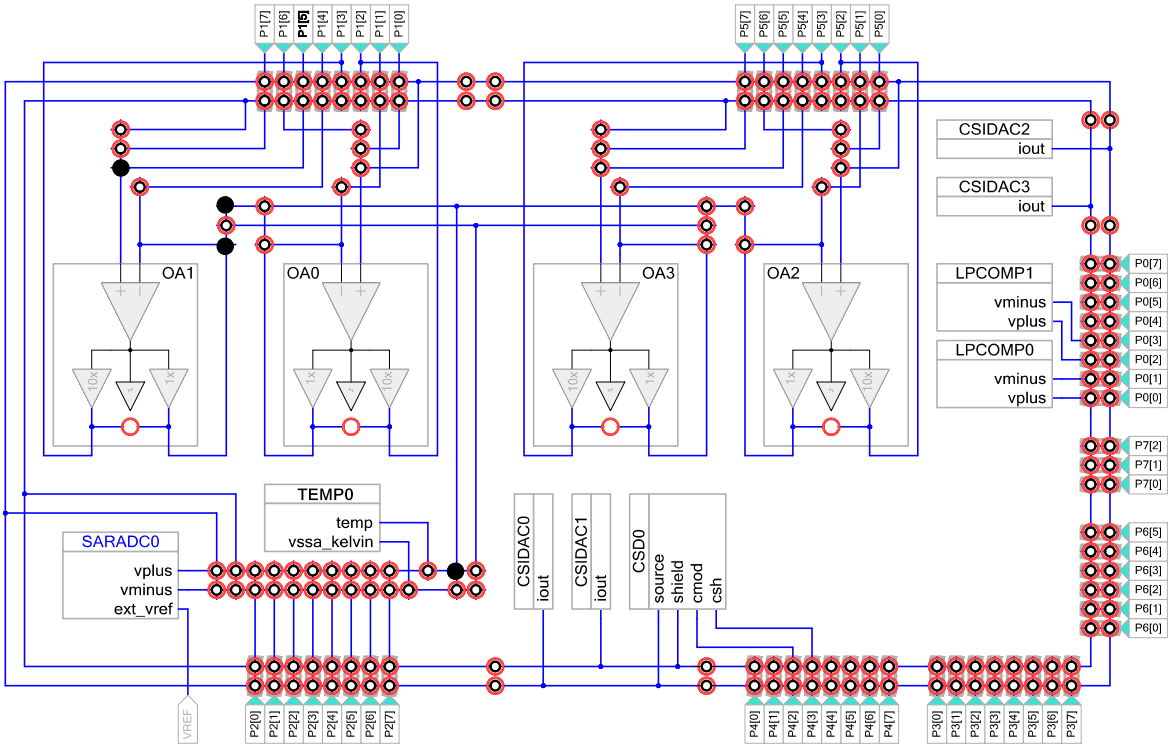
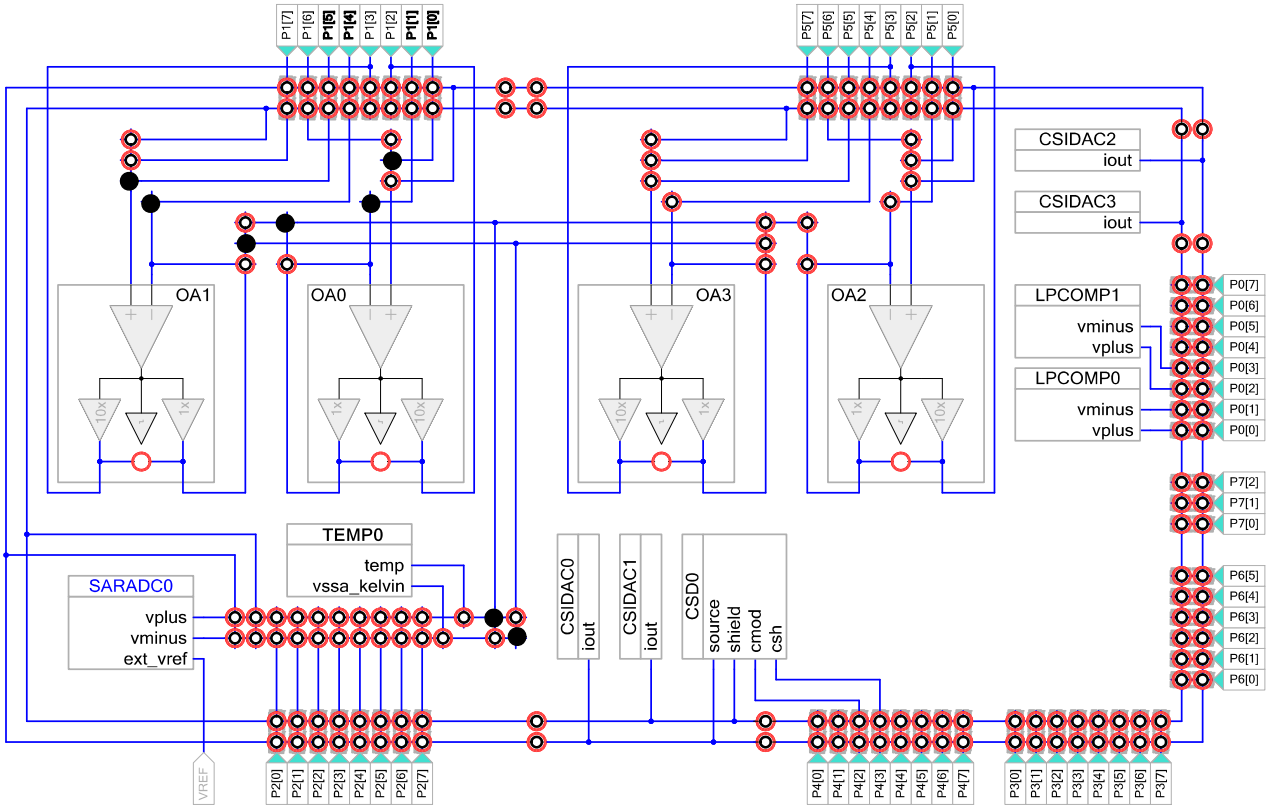


図 20-8. sarbus0 と sarbus1 を介した CTBm 出力からの入力

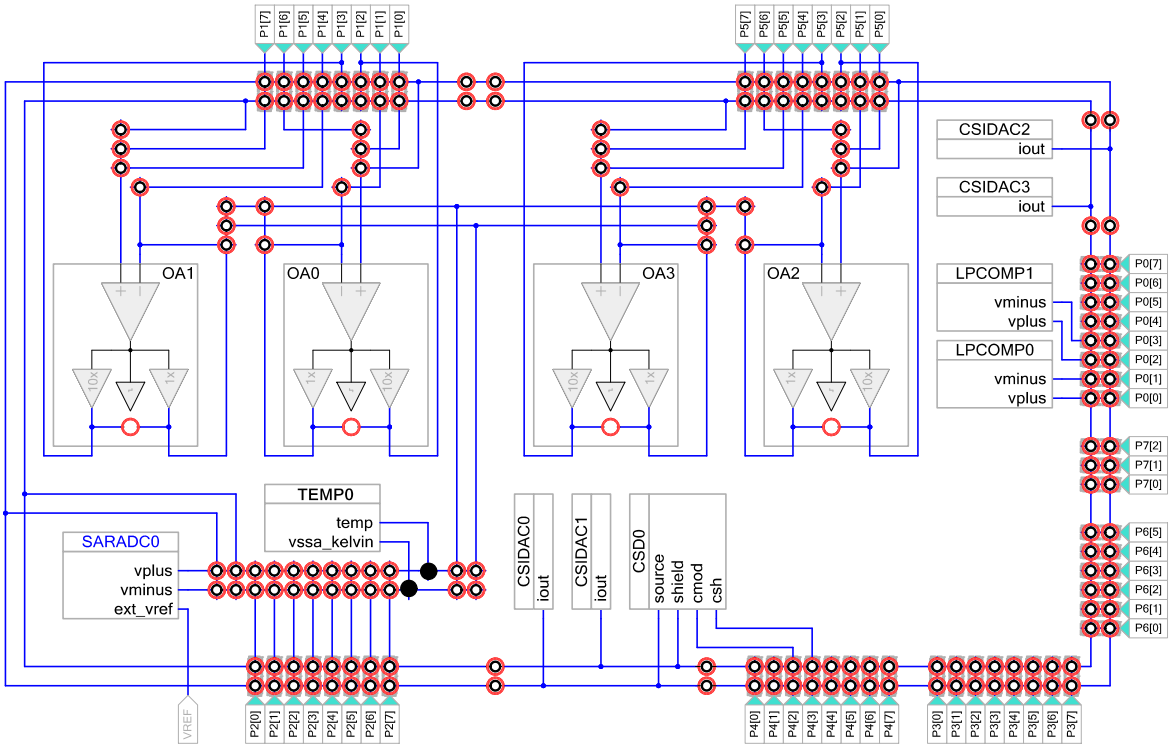


温度センサーからの入力

温度センシングおよび温度ベースの較正用に 1 個の内蔵温度センサーがあります。温度センサーは差動変換に対応していない (変換結果が未定義) ため、常にシングルエンド モードで使用されます。リファレンス電圧は内部 1.024V から供給されます。

図 20-9 に示すように温度センサーは、シーケンサー、ファームウェアまたは DSI で制御できるスイッチを介して SAR ADC の＋入力に接続することができます。MUX_FW_TEMP_VPLUS ビット (SAR_MUX_SWITCH0[17]) をセットすると温度センサーが有効になり、出力が SAR ADC の VPLUS に接続されます。このビットをクリアすると温度センサーのバイアス電流が遮断され、温度センサーが無効になります。

図 20-9. 温度センサーからの入力

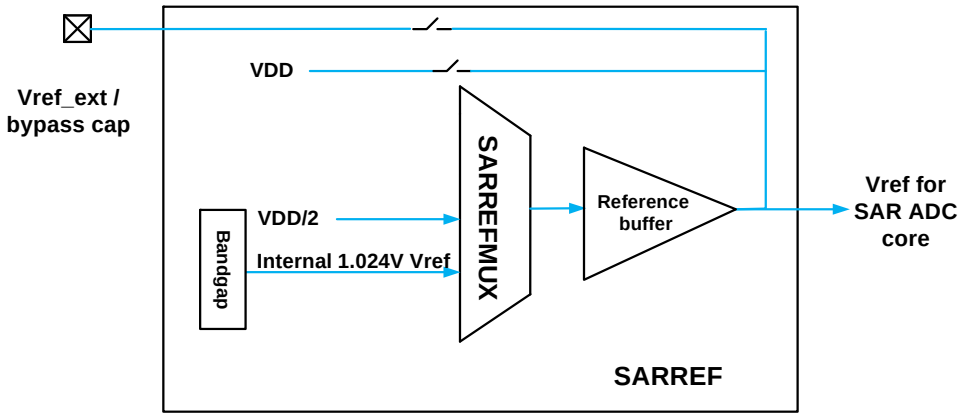


20.3.3 SARREF

SARREF の特長は以下の通りです。

- リファレンス電圧オプション : V_{DDA}、$V_{DDA}/2$、1.024V バンドギャップ (±1%)、外部リファレンス電圧
- 内部リファレンスの駆動能力を強化するためのリファレンス バッファ + バイパス コンデンサ

図 20-10. SARREF ブロック図



20.3.3.1 リファレンスのオプション

SAR ADCのリファレンス電圧の選択はSARREF内のリファレンス マルチプレクサとスイッチに基づいています。リファレンス電圧は V_{DDA}、$V_{DDA}/2$、バンドギャップからの 1.024V 内部リファレンスまたは外部 Vref ／ SAR バイパス ピンに接続した外部 V_{REF} から 1 つ選択できます (詳細はデバイス データシートを参照してください)。SARREF 内のリファレンス マルチプレクサの制御はグローバル コンフィギュレーション レジスタ SAR_CTRL [6:4] で設定されます。

20.3.3.2 バイパス コンデンサ

内部リファレンス 1.024V バンドギャップまたは $V_{DDA}/2$ はリファレンス バッファでバッファされます。このリファレンスは Ext Vref/SAR バイパス ピンに接続できます。このピンでは外部コンデンサを使用してリファレンス信号に存在する可能性のある内部ノイズをフィルター処理することができます。

外部リファレンス バイパス コンデンサがなければ SAR ADC サンプル速度は 100ksps を超えられません。たとえばバイパス コンデンサなしで 1.024V 内部 V_{REF} を使用する場合、SAR ADC の最大クロック周波数は 1.6MHz です。外部リファレンス電圧を使用する場合は外部コンデンサを使用することが推奨されています。バイパス コンデンサは SAR_CTRL [7] をセットすることで有効にします。

表 20-3 に PSoC 4200M における各種のリファレンス モードおよび 12 ビット連続モード動作の最大周波数／サンプル速度を示します。

表 20-3. リファレンス モード

リファレンス モード	リファレンス SAR_CTRL [6:4]	バイパス コンデンサ SAR_CTRL[7]	バッファ	最大周波数	最大サンプル速度
バイパス コンデンサを使わない場合の内部 1.024V V_{REF}	4	0	有	1.6MHz	100ksps
バイパス コンデンサを使う場合の 1.024V 内部 V_{REF}	4	1	有	18MHz	1Msps
外部 V_{REF}	5	X	無	18MHz	1Msps
バイパス コンデンサを使わない場合の $V_{DDA}/2$	6	0	有	1.6MHz	100ksps
バイパス コンデンサを使う場合の $V_{DDA}/2$	6	1	有	18MHz	1Msps
V_{DDA}	7	X	無	9MHz	500ksps

内部 1.024V V_{REF} の立ち上がり時間はバイパス コンデンサ サイズによって異なります。表 20-4 に一般的な 2 つのバイパス コンデンサの値および立ち上がり時間仕様を示します。スキャン間でリファレンスの選択が変更されれば、SAR ADC がサンプリングを開始する時内部 1.024V V_{REF} がセトリングされることを保証してください。

表 20-4. バイパス コンデンサ静電容量

内部 V_{REF} 立ち上がり時間	最大仕様
1uF の外部コンデンサを使う場合のリファレンス電圧の立ち上がり時間	2ms
100nF の外部コンデンサを使う場合のリファレンス電圧の立ち上がり時間	200µs

20.3.3.3 入力範囲とリファレンス

すべての入力は V_{SSA} と V_{DDA} の範囲内である必要があります。ADC の入力範囲は V_{REF} の選択で制限されます。－入力が Vn で、ADC リファレンス電圧が V_{REF} である場合、＋入力の電圧は Vn ± V_{REF} となります。この基準は－入力と＋入力が V_{SSA} と V_{DDA} の範囲内にあればシングルエンドと差動両方のモードに適用できます。

20.3.4 SARSEQ

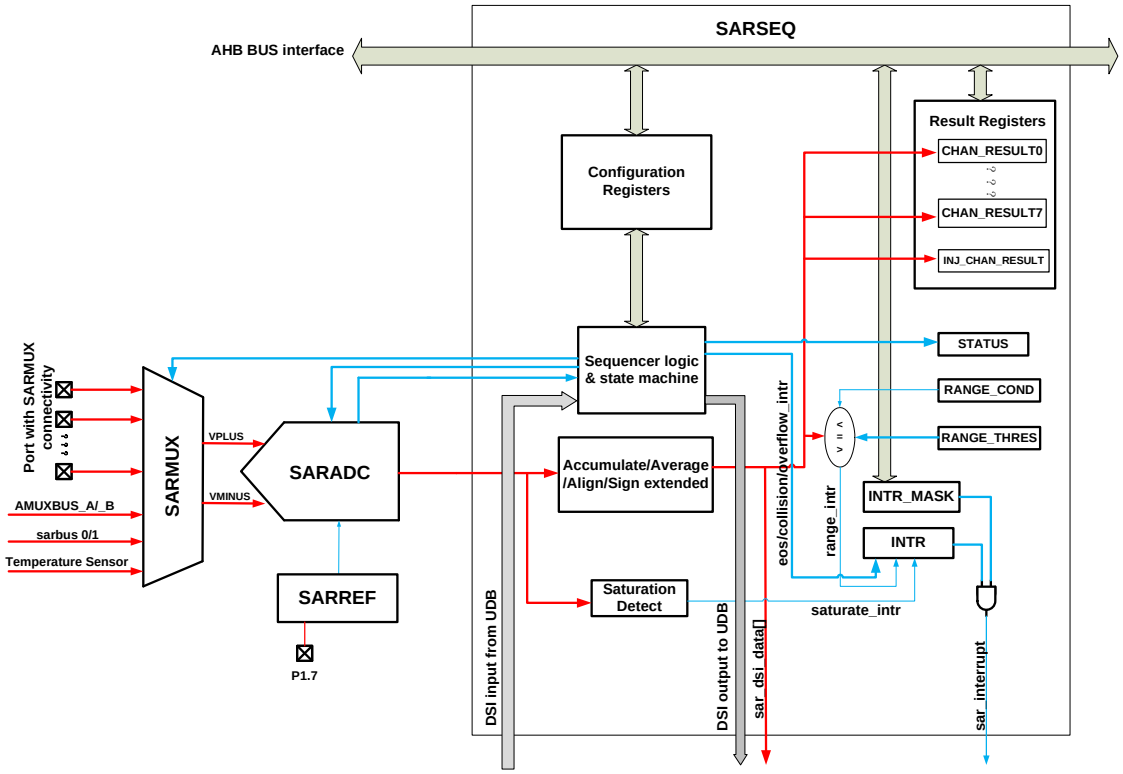
SARSEQ は専用シーケンサー コントローラーであり、自動的に入力マルチプレクサをチャネルからチャネルまでスキャンし、結果をチャネルごとのレジスタ アレイに格納します。

- CPU の介入なく自動的に SARMUX アナログ配線を制御
- SAR ADC コアを制御 (分解能、取得時間、リファレンス電圧など)
- SAR ADC からデータを受信し、前処理を行う (平均化、範囲検出)
- 次のスキャンが進行中に CPU が前のスキャン結果を安全に読み出すため結果は二重バッファされる

SARSEQ の特長は以下の通りです。

- 8本のチャネルはCPUの介入なく個別に自動スキャンとして設定可能
- 不定期な信号を自動スキャンに挿入するための 9 番目のチャネル (注入チャネル)
- 各チャネルの持つ特長 :
  - 外部ピンまたは内部信号からの入力 (AMUXBUS ／ CTBm ／温度センサー)
  - 4 つのプログラム可能な取得時間オプション
  - 12 ビットの初期設定分解能、選択可能な予備分解能 : 8 ビットまたは 10 ビット
  - シングルエンド入力または差動入力
  - 結果平均化
- スキャン トリガー
  - ワンショット、定期または連続モード
  - GPIO ピンからのデジタル信号または入力でトリガー
  - 内部 UDB または機能固定ブロック
  - ソフトウェアでトリガー
- ハードウェアによる平均化に対応
  - 一次の蓄積
  - 2 ～ 256 (2 のべき乗) 回数のサンプル平均化
  - 結果が 16 ビットで表される
- 出力データの二重バッファ
  - 結果が左アライメントまたは右アライメント
  - 結果がワーク レジスタと結果レジスタに格納される
- 割り込みの生成
  - 完了したスキャン変換
  - すべての制御モードでのチャネル飽和検出
  - 各チャネルでオーバーレンジ (設定可能) 検出
  - スキャン結果オーバーフロー
  - 衝突検出
- 注入チャネルの設定
  - ファームウェアでトリガー
  - 2 つのスキャン シーケンス間に挟まれる (テール ゲート)
  - 選択可能なサンプリング時間、分解能、シングルエンド入力か差動入力、平均化

図 20-11. SARSEQ ブロック図



20.3.4.1 平均化

SARSEQ ブロックは平均化を実装するための 20 ビット アキュムレータとシフト レジスタがあります。平均化のビットは符号拡張ビットの後にあります。グローバル コンフィギュレーション レジスタ SAR_SAMPLE_CTRL は平均化の詳細を指定します。

レジスタ制御モードではチャネル コンフィギュレーション レジスタ SAR_CHAN_CONFIG には平均化を有効にするイネーブル ビット (AVG_EN) があります。DSI 制御モードでは平均化は dsi_cfg_average 信号で有効にします。

グローバル コンフィギュレーションで AVG_CNT (SAR_SMAPLE_CTRL [6:4]) は以下の式でサンプル数 (N) を指定します。

N=2^(AVG_CNT+1) N range = [2..256]

たとえば AVG_CNT (SAR_SMAPLE_CTRL [6:4]) = 3 の場合、N = 16 となります。

AVG_SHIFT ビット (SAR_SAMPLE_CTRL[7]) は結果をシフトして平均化するために使用されます。平均化が有効にされた場合ビットをセットする必要があります。

特定のチャネルが平均化される場合、SARSEQ は各スキャンでチャネルの N 個の連続したサンプルを取ります。変換結果が 12 ビットで、N の最大値が 256 (8 ビット左シフト) であるため、20 ビット アキュムレータはオーバーフローしません。

SAR_SAMPLE_CTRLレジスタのAVG_SHIFTビットがセットされると SAR シーケンサーは符号拡張に続いて蓄積を実行します。蓄積された結果は、(AVG_CNT + 1) ビット右シフトされ平均化されます。そうしないと結果は 16 ビットに合わされるように右シフトさせられます。右シフトされるビット数は 0 ～ AVG_CNT - 3 ビットです。サンプル数が 16 (AVG_CNT > 3) より多い場合、蓄積結果は AVG_CNT - 3 ビット右シフトされます。AVG_CNT < 3 の場合、結果はシフトされません。この場合平均化結果は予測より大きく、AVG_SHIFT ビットをセットすることが推奨されることに注意してください。このモードは常に ADC の選択された分解能 (12、10、8 ビット) を使用します。

20.3.4.2 範囲検出

SARSEQ は範囲検出に対応しており、CPU の介入なくプログラム可能な 2 つの閾値と比較された結果の自動検出が可能です。範囲検出は SAR_RANGE_THRES レジスタで定義します。RANGE_LOW フィールド (SAR_RANGE_THRES [15:0]) は範囲の下限を定義し、RANGE_HIGH フィールド (SAR_RANGE_THRES [31:16]) は上限を定義します。

SAR_RANGE_COND ビットはチャネル マスク可能範囲検出割り込み (RANGE_INTR) をトリガーする条件を定義します。以下の条件を選択できます。

0: 結果 < RANGE_LOW (下限未満)

1: RANGE_LOW ≤ 結果 < RANGE_HIGH (範囲内)

2: RANGE_HIGH ≤ result (上限超)

3: 結果 <RANGE_LOW || RANGE_HIGH <= 結果 (範囲外)

詳細は 238 ページの範囲検出の割り込みを参照してください。

20.3.4.3 二重バッファ

二重バッファはファームウェアが次のスキャンが進行中に完了したスキャンの結果を読み出すために使用されます。SAR ADCのスキャン結果はスキャンが完了するまでワークレジスタに書き込まれます。この時データは 2 番目のレジスタにコピーされ、ユーザー アプリケーションで読み出されます。これにより現時点のスキャンが完了する前にファームウェアが前のスキャン結果を読み出すのに十分な時間がとれます。注入チャネルを除き、すべての入力チャネルは 16 のレジスタで二重バッファされます。注入チャネルは通常のチャネル スキャンの一部ではないため、二重バッファは必要ありません。

20.3.4.4 注入チャネル

注入チャネルは通常のスキャンの一部ではないという点を除き、他のチャネルに似ています。注入チャネルは偶発的または稀な変換に使用されます。たとえば 2 秒ごとに温度センサーをサンプリングすることです。SAR が連続モードで動作する場合注入チャネルを有効にするとサンプル速度が変更されます。

注入チャネルはファームウェア トリガー (ワンショット) により制御されます。注入チャネルは連続またはDSIトリガーに対応していません。またデータや割り込みの DSI バスへの出力をサポートしていません。ファームウェア トリガーがワンショットであるため二重バッファまたはオーバーフロー割り込みは不要です。

注入チャネルの変換は SAR_INJ_CHAN_CONFIG レジスタをセットすることで通常チャネルと同じように設定することができます。以下のものをサポートします。

- ピンまたは信号の選択
- シングルエンド入力または差動入力
- 12 ビットまたはグローバル指定された SUB_RESOLUTION の分解能の選択
- 4 つのグローバルに指定されたサンプル時間オプションから選択可能なサンプル時間
- 平均化の選択

注入チャネルはオーバーフロー割り込みを除いて通常チャネルと同じ割り込みに対応しています。

- マスク可能変換の終了割り込み INJ_EOC_INTR
- マスク可能範囲検出割り込み INJ_RANGE_INTR
- マスク可能飽和検出割り込み INJ_SATURATE_INTR
- マスク可能衝突割り込み INJ_COLLISION_INTR

対応するレジスタは SAR_INTR、SAR_INTR_MASK、SAR_INTR_MASKED、SAR_INTR_SET です。

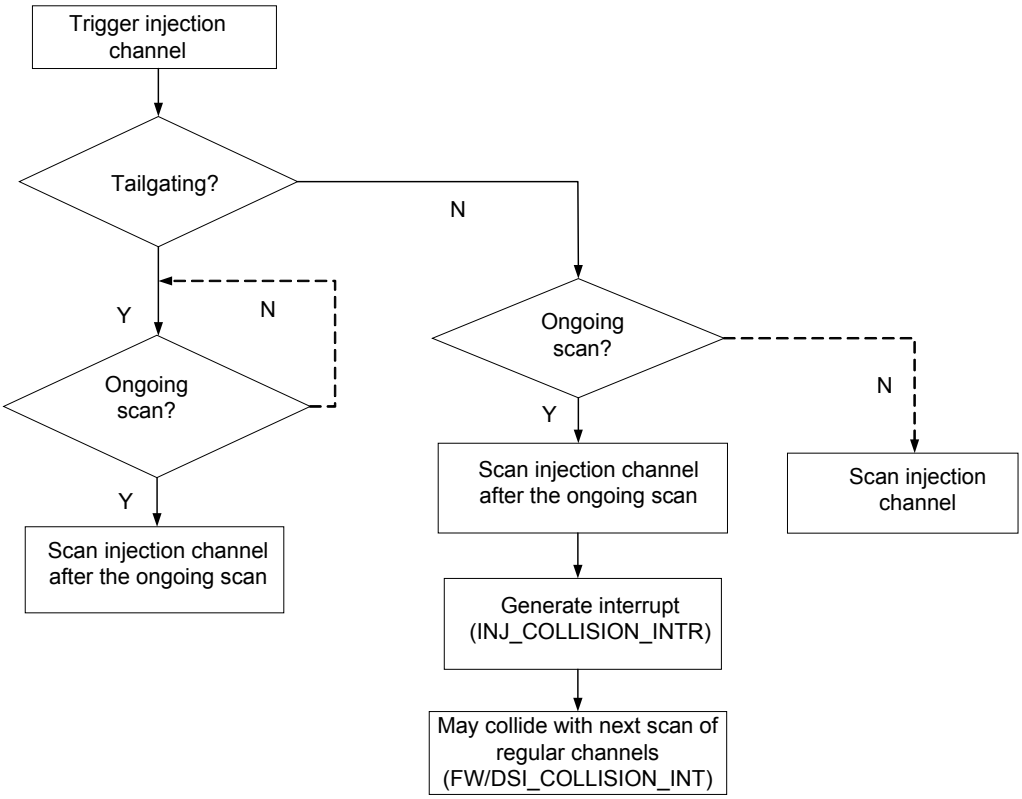
これらの機能を 242 ページのグローバル SARSEQ コンフィギュレーション、243 ページのチャネル コンフィギュレーション、237 ページの割り込みで詳細に説明します。

テールゲート

注入チャネルの変換は開始ビット (イネーブル ビット) INJ_START_EN (SAR_INJ_CHAN_CONFIG [31]) をセットすることでトリガーします。進行中のスキャンがあれば、INJ_TAILGATING = 1 (SAR_INJ_CHAN_CONFIG [30]) にセットすることでテールゲートを選択することが推奨されます。注入チャネルは進行中の通常チャネルのスキャンが終わるとスキャンされ、衝突はありません。しかし進行中のスキャンがないか SAR ADC がアイドルの時、テールゲートを選択すると、INJ_START_EN は次の通常チャネル スキャンの終わりに注入チャネルのスキャンを行います。この場合テールゲートは不要です。

テールゲートを選択しないと、注入チャネルは進行中の通常チャネルのスキャンが終わるとスキャンされますが、衝突が発生し、衝突割り込み (INJ_COLLISION_INTR) が生成されます。テールゲートを選択しない場合のもう一つの問題は、次の通常チャネルのスキャンが注入チャネルの変換と衝突する可能性があることです。(FW/DSI_COLLISION_INTR が HIGH にセットされます)。通常チャネルのスキャンは注入チャネルのスキャンが終わるまで延期され、通常のスキャンにジッタを発生します。連続トリガーおよび DSI トリガーレベル モードは衝突割り込みをトリガーしないことに注意してください。

図 20-12. 注入チャネルのフロー図



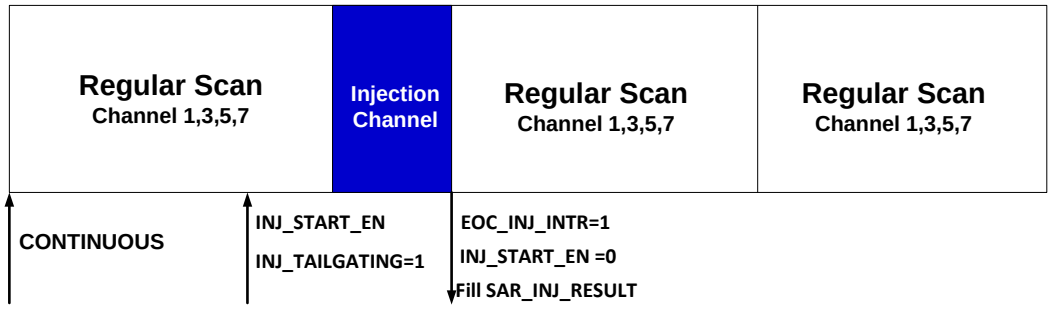
テールゲートの欠点は次のトリガーが発生するまで長時間かかる可能性があることです。通常チャネルに衝突やジッタが発生する可能性がなければ、注入チャネルはテールゲートを使わずに安全に使用できます。

注入チャネルの変換が完了した後変換の終了割り込み (INJ_EOC_INTR) がセットされ、INJ_START_EN ビットはクリアされます。注入チャネルの変換データは SAR_INJ_RESULT レジスタに格納されます。SAR_CHAN_RESULT と同様に、レジスタは「有効」(= INJ_EOC_INTR)、範囲検出、飽和検出割り込み、衝突割り込み (INJ_COLLISSION_INTR) のミラー ビットがあります。

図 20-13 はテールゲートが有効な場合、注入チャネルが連続スキャン (チャネル 1、3、5、7 が有効) 中に有効になる例です。

SAR が無効になると (それまでに有効であった場合) 直ちに INJ_START_EN ビットがクリアされることに注意してください。

図 20-13. テールゲートが有効な注入チャネル



20.3.5 割り込み

本節で説明される割り込みは、それぞれ SAR_INTR_MASK レジスタ内に割り込みマスクを持っています。割り込みマスクを LOW に設定すると、対応する割り込みソースが無視されます。割り込みマスクのビットが HIGH であり、対応する割り込みソースが保留中の場合では、SAR 割り込みは生成されます。

割り込みのサービスの時、割り込みサービス ルーチン (ISR) はデータを読み出してから、割り込みビットに 1 を書き込むことにより、割り込みソースをクリアします。

SAR_INTR_MASKED レジスタは、割り込みソースと割り込みマスクの論理 AND です。これはファームウェアが割り込みソースを判定するのに使用します。

検証およびデバッグのために、セット ビット (SAR_IN-

TR_SET レジスタの EOS_SET など) は各割り込みのトリガーに使用されます。これによりファームウェアは実際にイベントが発生しなくても割り込みを生成できます。

20.3.5.1 End-of-Scan (スキャンの終了の) 割り込み (EOS_INTR)

スキャンの終了後、スキャンの終了の割り込み (EOS_INTR) が立ち上がります。ファームウェアは RESULT レジスタからのデータを得た後、この割り込みをクリアします。

または、EOS_INTR は SAR_SAMPLE_CTRL [31] の EOS_DSI_OUT_EN ビットをセットすることにより、DSI バス上に送信することもできます。EOS_INTR 信号は、DSI バス上に 2 つのシステム クロック サイクルで保持されます。これらのサイクルは、スキャンの最後のチャネル (選択された場合) で、data_valid 信号と同時に発生します。

EOS_INTR は SAR_INTR_MASK レジスタ内の EOS_MASK ビット 0 の設定で、マスクすることが可能です。SAR_INTR_MASKED レジスタの EOS_MASKED ビットは、割り込みフラグおよび割り込みマスクの論理 AND です。SAR_INTR_SET レジスタの EOS_SET ビットに 1 を書き込むと、デバッグおよび検証を目的とする EOS_INTR を設定することが可能です。

20.3.5.2 オーバーフロー割り込み

新規のスキャンが終了し、ハードウェアが EOS_INTR をセットしようとするものの EOS_INTR がまだ HIGH である (ファームウェアがまだクリアしていない) 場合、オーバーフロー割り込み (OVERFLOW_INTR) がハードウェアによって生成されます。これは現在のスキャンが完了するまでに、ファームウェアが前の結果を読み出していないということです。この場合、古いデータは上書きされます。

OVERFLOW_INTR は、SAR_INTR_MASK レジスタ内の OVERFLOW_MASK ビット 0 の設定で、マスクすることが可能です。SAR_INTR_MASKED レジスタの OVERFLOW_MASKED ビットは、割り込みフラグおよび割り込みマスクの論理 AND であり、ファームウェアが簡易になります。SAR_INTR_SET レジスタの OVERFLOW_SET ビットに1を書き込むと、デバッグおよび検証を目的とするOVERFLOW_SET を設定することが可能です。

20.3.5.3 衝突割り込み

前のトリガーでスキャンが開始して、SARSEQ が取り込み中に、新規のトリガーを生成することが可能です。従って新規のトリガーのスキャンは進行中のスキャンが完了するまで遅延されます。新規のサンプルが無効であることをファームウェアに通知することは大切です。これは連続トリガーではなく、新しいトリガーで立ち上がった衝突割り込みにより行われます。

衝突割り込みは 3 つがあり、ファームウェア トリガー (FW_COLLISION_INTR) 用、DSI トリガー (DSI_COLLISION_INTR) 用および注入チャネル (INJ_COLLISION_INTR) 用です。これによりファームウェアはどのトリガーが進行中のスキャンと衝突したかを特定することができます。

DSI トリガーがレベル モードで使用される場合、DSI_COLLISION_INTR は決してセットされません。

3 つの衝突割り込みは、SAR_INTR_MASK レジスタ内で対応するビット 0 を設定することにより、マスクされます。SAR_INTR_MASKED レジスタの対応するビットは、割り込みフラグおよび割り込みマスクの論理 AND です。SAR_INTR_SET レジスタの対応するビットに 1 を書き込むと、デバッグおよび検証を目的とする衝突割り込みを設定することが可能です。

20.3.5.4 注入変換の終了の割り込み (INJ_EOC_INTR)

注入チャネルの変換の終了後、注入変換の終了の割り込み (INJ_EOC_INTR) が立ち上がります。このファームウェアは、INJ_RESULT レジスタからのデータを得た後、この割り込みをクリアします。

注入チャネルがスキャンの後に続く場合、EOS_INTR が立ち上がり注入チャネルの変換も開始します。注入チャネルはスキャンの一部とみなされません。

INJ_EOC_INTR は、SAR_INTR_MASK レジスタの INJ_EOC_MASK ビット 0 の設定によりマスクされます。SAR_INTR_MASKED レジスタの INJ_EOC_MASKED ビットは、割り込みフラグおよび割り込みマスクの論理 AND です。SAR_INTR_SET レジスタの INJ_EOC_SET ビットに 1 を書き込むと、デバッグおよび検証を目的とする INJ_EOC_INTR を設定することが可能です。

20.3.5.5 範囲検出の割り込み

範囲検出の割り込みフラグは、平均化、アライメントおよび符号拡張の後にセットすることができます (妥当な場合)。チャネル変換がオーバーレンジであるかどうかを判断するために、すべてのスキャンが完了するまで待機する必要がないという意味です。閾値は結果データのデータ フォーマットと同様のものを必要とします。

指定されたチャネル用の範囲検出の割り込みは、SAR_RANGE_INTR_MASK レジスタの指定ビットを 0 にクリアすることでマスクされます。SAR_RANGE_INTR_MASKED レジスタは、割り込み要求とマスク レジスタのビット単位の AND を反映します。値が 0 でなければ、NVIC への SAR 割り込み信号は HIGH です。

SAR_RANGE_INTR_SET はデバッグと検証に使用することが可能です。1 を書き込んで、割り込み要求レジスタの対応するビットをセットします。範囲検出の割り込みは読み出す時に、割り込み要求レジスタを反映します。

チャネルそれぞれに範囲検出の割り込みがあります (RANGE_INTR および INJ_RANGE_INTR)。

20.3.5.6 飽和検出の割り込み

飽和検出は全ての変換に適用されます。サンプル値が各分解能の最大最小値に等しい場合を検出します。その場合、マスク可能な割り込みフラグが対応するチャネルにセットされます。これにより SAR ADC が飽和する時、ファームウェアは結果の廃棄などの対処を行うことができます。サンプル値は変換の直後および平均化の前に検査されます。すなわち割り込みは、データ レジスタの平均化された結果が最大最小値に等しくなくてもセットされるということです。

チャネルに 10 ビットまたは 8 ビットの分解能が選択される時、飽和検出は 10 ビットまたは 8 ビットで行われます。

飽和割り込みフラグは、フル スキャンと平均化の前に、飽和に対して速く応答できるように、直ちにセットされます。指定されたチャネルの飽和検出の割り込みは、SAR_SATURATE_INTR_MASK レジスタの指定ビットを 0 にクリアすることによりマスクされます。SAR_SATURATE_INTR_MASKED レジスタは、割り込み要求とマスク レジスタのビット単位の AND を反映します。値が 0 でなければ、NVIC への SAR 割り込み信号は HIGH です。

SAR_SARTURATE_INTR_SET はデバッグと検証に使用す

ることが可能です。1 を書き込んで、割り込み要求レジスタの対応するビットをセットします。範囲検出の割り込みは読み出す時に、割り込み要求レジスタを反映します。

20.3.5.7 割り込み要因のあらまし

INTR_CAUSE レジスタは、あらゆる待機中の SAR 割り込みを反映します。割り込みの要因を判定は ISR が行います。レジスタは SAR_INTR_MASKED のミラー コピーで構成されます。レジスタは範囲検出割り込みと飽和割り込みを全チャネルについて統合した 2 つのビットを持っています。RANGE_INTR_MASKED および SATURATE_INTR_MASKED レジスタ内のすべてのビットの論理 OR を含みます (INJ_RANGE_INTR と INJ_SATURATE_INTR を含みません)。

20.3.6 トリガー

スキャンのトリガーの方法は 3 つあり、以下の通りです。

- ファームウェアが SAR_START_CTRL レジスタの FW_TRIGGER ビットに書き込む時、ファームウェアまたはワンショット トリガーが生成されます。スキャンの終了後、SARSEQ は FW_TRIGGER ビットをクリアし、アイドル モードに戻り、次のトリガーを待機します。FW_TRIGGER ビットは SAR が無効化された後直ちにクリアされます。
- 周期トリガーは DSI 接続 (dsi_trigger) に呼応します。このトリガーは TCPWM の出力に接続されます。ただし GPIO ピンまたは UDB に接続することも可能です。UDB にイベントの特定のシーケンスを探すステート マシンを実装することができます。
- 連続トリガーは SAR_SAMPLE_CTRL レジスタの CONTINUOUS ビットをセットすることにより有効にされます。このモードでは SARSEQ はあるスキャンの終了後直ちに次のスキャンを行うのでいつも BUSY です。結果としてその他のトリガーは無視されます。FW_TRIGGER は次のスキャンの完了で、ハードウェアによってクリアされることに注意してください。

ハードウェアの要件はありませんが、3 つ のトリガーは相互に排他的です。DSI トリガーがファームウェア トリガーと同時に行われる場合、DSI トリガーが先に処理され、ファームウェア トリガーにより個別のスキャンが行われます (衝突割り込みが設定されます)。DSI トリガーが連続トリガーと同時に行われる場合、両方は実効的に同時に処理されます (衝突トリガーが DSI トリガーによりセットされることがあります)。

ファームウェア または連続トリガーの場合では、シーケンサーが SAR ADC にサンプルを開始させる前に、1 SAR ADC クロック サイクルがかかります (シーケンサーがアイドルの場合)。DSI トリガーの場合では、トリガー コンフィギュレーション設定によります。

20.3.6.1 DSI トリガー コンフィギュレーション

- DSI 同期化

SARSEQ の DSI インターフェースは、システムのクロック周波数 (clk_sys) で実行します。詳細は 67 ページのクロック供給システムを参照してください。着信 DSI トリガー信号は、AHB クロックと同期化されない場合、ダブルフリップフロップ (初期設定) を用いてその信号を同期化する必要があります。ただし DSI トリガー信号が AHB クロックと既に同期化された場合、このダブルフリップフロップを無視することができます。SAR_SAMPLE_CTRL レジスタ内の DSI_SYNC_TRIGGER コンフィギュレーション ビットは無視されるダブルフリップフロップを制御します。DSI_SYNC_TRIGGER は、DSI パルス トリガー信号のトリガー幅 (TW) とトリガー間隔 (TI) の要求に影響を及ぼします。

- DSI トリガー レベル

DSI トリガーは、SAR_SAMPLE_CTRL レジスタ内の DSI_TRIGGER_LEVEL コンフィギュレーション ビットによって示され、パルスかレベルになります。DSI トリガー信号がレベル入力である場合、SAR は DSI トリガー信号が HIGH である限り、新規のスキャンを開始します。DSI トリガー信号がパルス入力である場合、DSI トリガー信号上のポジティブ エッジは新規のスキャンをトリガーします。

- 送信時間

dsi_trigger が立ち上がった後、SAR ADC がサンプルを開始する前に、送信時間がかかります。異なる DSI_SYNC_TRIGGER と DSI_TRIGGER_LEVEL コンフィギュレーションにつれて、送信時間が異なります。表 20-5 は最大の送信時間を示します。2 つのトリガー パルス間隔は送信時間より長くなる必要があります。そうでなければ、第 2 のトリガーが無視されます。

SAR が無効である時 (ENABLED=0)、DSI トリガーは無視されます。

表 20-5. DSI トリガー最大時間

最大 DSI_TRIGGER 送信時間	同期回路をバイパス、DSI_SYNC_TRIGGER=0	同期回路をイネーブル、DSI_SYNC_TRIGGER=1 (初期設定)
パルス トリガー: DSI_TRIGGER_LEVEL=0 (初期設定)	1 clk_sys+2 clk_sar	3 clk_sys+2 clk_sar
レベル トリガー: DSI_TRIGGER_LEVEL=1	2 clk_sar	2 clk_sys+2 clk_sar

表 20-6. トリガー信号の要件

トリガー仕様	要件
トリガー幅 (TW)	トリガーをロックできるように TW は十分に大きい必要がある。DSI_SYNC_TRIGGER=1 の場合 TW >= 2 clk_sys サイクル。DSI_SYNC_TRIGGER=0 の場合 TW >= 1 SAR クロック サイクル
トリガー間隔 (TI)	DSI パルス トリガー信号のトリガー間隔は表 20-5 で指定した送信時間より長い必要がある。そうしないと次のトリガー パルスが無視される

20.3.7 SAR ADC ステータス

SAR の状態は、SAR_STATUS レジスタ内の BUSY および CUR_CHAN の領域により監視することが可能です。SAR がチャネルをサンプルし、またはそれを変換するたびに、BUSY ビットは HIGH になります。CUR_CHAN [4:0] ビットは、サンプルされた現在のチャネル数を示します (チャネル 16 が注入チャネルを示します)。SW_VREF_NEG ビットは、V_{REF} 入力と NEG を接続する SAR ADC 内のスイッチの DSI とレジスタの制御を含む、現在のスイッチ状態を示します。

最後のスキャンでサンプルされたワーク データが有効であれば、CHAN_WORK_VALID レジスタ内の CHAN_WORK_VALIDがセットされます。CHAN_RESULT_VALIDレジスタ内の CHAN_RESULT_VALID がセットされることは、RESULT データが有効であることを意味します。その後対応する CHAN_WORK_VALID ビットがクリアされます。SAR_AVG_STAT レジスタ内の CUR_AVG_ACCU と CUR_AVG_CNT の領域は、現在の平均化アキュムレーターの内容および現在のサンプル カウントを示します (カウントダウン)。

SAR_MUX_SWITCH_STATUS レジスタは、MUX_SWITCH0 レジスタの現在のスイッチ状態を提供します。

これらのステータス レジスタはSAR動作のデバッグに役立ちます。

20.3.8 低消費電力モード

SAR ADC の電流消費量は、SAR ADC コアと SARREF の 2 つの部分に分けることが可能です。SAR 動作の消費電力を削減する方法はいくつかあります。最も簡単な方法はトリガー周波数を低減することです。それは 1 秒当たりの変換回数を低減することです。

SAR ADC は SAR ADC の全体の電力を制御する ICONT_LV[1:0] コンフィギュレーションビットを提供します。各電力設定の最大クロック速度を監視する必要があります。

表 20-7. 消費電力設定 ICONT_LV

ICONT_LV[1:0]	SAR ADC コアの相対消費電力 (%)	最大周波数 [MHz]	最小サンプル時間 [サイクル]	最大サンプル速度 (12 ビット分解能) [ksps]
0	100	18	4	1000
1	50	4.5	3	250
2	133	18	4	1000
3	25	4.5	2.25	125

最後に電力削減のために、高精度を要しないチャネルで低い分解能を使用します。これで変換時間を 18 サイクルから最大 4 サイクル短縮します (8 ビット分解能および最短のサンプル時間)。

20.3.9 システム動作

ENABLED ビット (SAR_CTRL [31]) の設定により SAR アナログが有効にされた後、SARSEQ で ADC 変換を開始するために、以下の手順に従ってください。

1. SAR ADC 制御モードを設定 : 20.3.10 レジスタ モード または 20.3.11 DSI モード
2. シーケンサー／ファームウェア／ DSI を用いて、SARMUX アナログ配線 (ピン／信号の選択) を設定
3. グローバルな SARSEQ 変換コンフィギュレーションを設定
4. 各チャネル ソース (ピン アドレスなど) を設定
5. チャネルを有効化
6. トリガー タイプを設定
7. 割り込みマスクを設定
8. トリガー ソースを開始
9. 各変換割り込みの終了後、データを取り込む
10. 必要に応じて注入変換を行う

レジスタ モードではレジスタを用いて、DSIモードではUDBからの DSI を使用して、SARMUX と SAR ADC の変換を制御します。この 2 つの制御モードの主要な相違点を表 20-8 に示します。DSI_MODE ビット (SAR_CTRL [29]) の設定で、DSI モードを有効にすることができます。

表 20-8. 制御モード間の違い

制御モード	レジスタ	DSI
DSI_MODE	0	1
SARMUX 制御	シーケンサー制御レジスタ SAR_CHANx_CONFIG、SAR_MUX_SWITCH0、SAR_MUX_HW_SWITCH_CTRL SAR_CTRL ファームウェア制御レジスタ SAR_MUX_SWITCH0、SAR_MUX_HW_SWITCH_CTRL、SAR_CTRL	DSI 信号制御信号 : dsi_out、dsi_oe,dsi_swctrl、dsi_sw_negvref ファームウェア制御レジスタ : SAR_MUX_SWITCH0、SAR_MUX_HW_SWITCH_CTRL、SAR_CTRL
グローバル コンフィギュレーション	グローバル コンフィギュレーション レジスタ : SAR_CTRL、SAR_SAMPLE_CTRL、SAR_SAMPLE01、SAR_SAMPLE23、SAR_RANGE_THES、SAR_RANGE_COND	グローバル コンフィギュレーション レジスタ : SAR_CTRL、SAR_SAMPLE_CTRL、SAR_SAMPLE01、SAR_SAMPLE23、SAR_RANGE_THES、SAR_RANGE_COND
チャネル コンフィギュレーション	チャネル コンフィギュレーション レジスタ : CHAN_CONFIG、CHAN_EN、INJ_CHAN_CONFIG	DSI 信号 : dsi_cfg_st_sel、dsi_cfg_average、dsi_cfg_resolution、dsi_cfg_differential (CHAN_CONFIG、CHAN_EN、INJ_CHAN_CONFIG が無視される)
トリガー	すべて適用可能 ファームウェア トリガー (SAR_START_CTRL[0]) DSI トリガー (dsi_trigger) 連続トリガー (SAR_SAMPLE_CTRL [0])	すべて適用可能 ファームウェア トリガー (SAR_START_CTRL[0]) DSI トリガー (dsi_trigger) 連続トリガー (SAR_SAMPLE_CTRL [0])
割り込み	すべて適用可能	すべて適用可能 (DSI 信号上の EOS_INTR、RANGE_INTR、SATURATE_INTR 出力のみ)
DSI 出力	対応	対応
結果データ	8 本のチャネルの結果レジスタと 1 本の注入チャネルの結果レジスタ	チャネル 0 の結果レジスタのみ
注入	対応	非対応
平均化	1 本のピン／信号で平均化に対応	異なるピン／信号で平均化に対応

20.3.10 レジスタ モード

レジスタを用いてSAR ADCを設定するのは最も一般的な用法です。レジスタ ビットの詳細な定義については PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM を参照してください。

20.3.10.1 SARMUX アナログ配線

レジスタ モードでは SARMUX アナログ配線を制御する方法は 2 つあり、シーケンサー制御およびファームウェア制御です。

シーケンサー制御

MUX_SWITCH_HW_CTRL レジスタの適切なハードウェア制御ビットとMUX_SWITCH0 レジスタのファームウェア制御ビットを 1 にセットする必要があります。SWITCH_DISABLE=0 であることを確認します。シーケンサー制御が無効になるように SWITCH_DISABLE を設定します。

シーケンサーの制御では、チャネルが変換するピンまたは内部信号はポートとピン アドレスの組み合わせにより指定されます。PORT_ADDR ビットは SAR_CHANx_CONFIG [6:4] であり、PIN_ADDR ビットは SAR_CHANx_CONFIG [2:0] です。表 20-9 は対応する SARMUX の選択による PORT_ADDR と PIN_ADDR の設定を示します。未使用のポート／ピンはPSoC 4シリーズの他の製品用に予約されます。

表 20-9. PORT_ADDR and PIN_ADDR

PORT_ADDR	PIN_ADDR	説明
0	0.7	SARMUX の 8 本の専用ピン
1	X	sarbus0[a]
1	X	sarbus1[a]
7	0	温度センサー
7	2	AMUXBUS-A
7	3	AMUXBUS-B

a. sarbus0 と sarbus1 は opamp0/1 を含む CTBm ブロックの出力に接続します。詳細は 261 ページの連続時間ブロックミニ (CTBm) を参照してください。PORT_ADDR=1 の場合 sarbus0 は PIN_ADDR の値にかかわらず SAR ADC の＋端子に接続します。差動モードが有効で、PORT_ADDR=1 の場合 sarbus1 は SAR ADC の－端子にのみ接続できます。

差動変換のために、－端子の接続は、PORT_ADDR と PIN_ADDR により定義される＋端子の接続に依存します。DIFFERENTIAL_EN の設定により、チャネルは無視された PIN_ADDR [0] とピン アドレスによって特定される偶数／奇数のピン ペアで差動変換を行います。P2.0/P2.1、P2.2/P2.3、P2.4/P2.5、P2.6/P2.7 はシーケンサー制御用の有効な差動ペアです。ファームウェアまたは DSI はより柔軟なアナログ回路を実装することができます。

シングルエンド変換のために、NEG_SEL (SAR_CTRL [11:9]) はどちらの信号を－入力に接続するかを決定するよう意図されます。差動モードではこれらのビットは無視されます。－入力の選択は入力電圧範囲および実効分解能に影響を与えます。詳細は 224 ページの－入力選択を参照してください。V_{SSA}、V_{REF} または SARMUX を経由する 8 本の外部入力から選択します。MUX_SWITCH_HW_CTRL レジスタがそのハードウェア制御ビットをもっていないため、－入力を V_{REF} に接続するために、追加ビットである SAR_HW_CTRL_NEGVREF (SAR_CTRL[13]) をセットしなければなりません。

ファームウェア制御

SARMUX はデフォルトでファームウェア制御で動作します。SAR ADC の VPLUS (＋) および VMINUS (－) の入力は、SAR_MUX_SWITCH0 [29:0] 内の適切なビットの設定により、別々に制御されます。ハードウェア スイッチ制御レジスタの適切なビットをクリアします (SAR_MUX_SWITCH_HW_CTR[n]=0)。またはハードウェア制御方式 (シーケンサー／ DSI) は SARMUX アナログ配線を制御します。

SAR_CTRL レジスタ ビット SWITCH_DISABLE は、配線スイッチの有効化から SAR シーケンサーの無効化に使用されます。ファームウェア制御モードがいつでもこのビットの値と無関係にスイッチを閉じることができることに注意してください。ただし 1 にセットすることを推奨します。

NEG_SEL (SAR_CTRL [11:9]) は、シングルエンド モードでどちらの信号が SAR ADC の－端子 (vminus) に接続されるかを決定します。差動モードではこれらのビットは無視されます。シングルエンド モードではシーケンサー制御を使用する時、これらのビットを設定しなければなりません。ファームウェア制御の使用時、NEG_SEL は無視され、SAR_MUX_SWITCH0 は－入力を制御するよう設定されます。特別な場合では、SAR_MUX_SWITCH0 は内部の V_{REF}vminus に接続せず、それから NEG_SEL を 7 に設定します。－入力の選択は入力電圧範囲、SNR および実効分解能に影響を与えます。詳細は 224 ページの－入力選択を参照してください。

20.3.10.2 グローバル SARSEQ コンフィギュレーション

すべてのチャネルに適用される変換選択の数はグローバルに設定されます。いくつかの場合では、チャネル コンフィギュレーションにはグロバール コンフィギュレーションのどちらを使用するかを選択するビットがあります。グローバル コンフィギュレーションは、レジスタ制御モードおよび DSI 制御モードの両方に適用されます。

SAR_CTRL、SAR_SAMPLE_CTRL、SAR_SAMPLE01、SAR_SAMPLE23、SAR_RANGE_THES、および SAR_RANGE_COND はすべてグローバル コンフィギュレーション レジスタです。

通常、スキャンが動作中にこれらの設定を変更しないようにします。使用中のコンフィギュレーション設定が変更されると、結果は未定義になります。現在未使用のコンフィギュレーション設定は進行中のスキャンに影響を与えず、変更することが可能です。

表 20-10. グローバル コンフィギュレーション レジスタ

コンフィギュレーション	制御レジスタ	参考文献
リファレンス電圧の選択	SAR_CTRL[6:4]	20.3.3.1 リファレンスのオプション
符号付き／符号無しの選択	SAR_SAMPLE_CTRL [3:2]	20.3.1.3 結果データのフォーマット
データの左／右アライメント	SAR_SAMPLE_CTRL [1]	20.3.1.3 結果データのフォーマット
シングルエンド モードでの－入力の選択	SAR_CTRL[11:9]	20.3.1.4 －入力選択
分解能	SAR_SAMPLE_CTRL[0][a]	20.3.1.5 分解能
取得時間	SAR_SAMPLE_TIME01 [25:0] SAR_SAMPLE_TIME32 [25:0]	20.3.1.6 取得時間
平均化回数	SAR_SAMPLE_CTRL[7:4]	20.3.4.1 平均化
範囲検出	SAR_RANGE_THRES [31:0] SAR_RANGE_COND [31:30]	20.3.4.2 範囲検出

a. 予備分解能はSAR_CHAN_CONFIGレジスタのSAR_RESOLUTIONビットで有効にします。予備分解能を有効にしない場合、ADCはSAR_SAMPLE_CTRL レジスタで設定された分解能にかかわらず 12 ビット分解能で動作します。

20.3.10.3 チャネル コンフィギュレーション

チャネル コンフィギュレーションは以下のものを含みます。

- シングルエンド入力または差動入力
- グローバル コンフィギュレーション選択 : サンプル時間、分解能、平均化イネーブル
- DSI 出力イネーブル

原則として、チャネル コンフィギュレーションは各スキャンの間のみで更新します (グローバル コンフィギュレーションと同様)。ただしチャネルが進行中のスキャン用に有効にされない場合、そのチャネル用のコンフィギュレーションは進行中のスキャンに影響を与えず、自由に変更することができます。このルールに違反すると結果は未定義です。自分を有効化するチャネルはこのルールのたった 1 つの例外です。有効にされたチャネルはスキャンが進行中に変更することができ、次のスキャンで有効です。有効にされたチャネルを変更すると、サンプル レートが変更されることがあります。

表 20-11. チャネル コンフィギュレーション レジスタ

コンフィギュレーション	レジスタ	参考文献
シングルエンド／差動	SAR_CHANx_CONFIG [8]	20.3.1.1 シングルエンドと差動モード
取得時間選択	SAR_CHANx_CONFIG [13:12]	20.3.1.6 取得時間
分解能選択	SAR_CHANx_CONFIG [9]	20.3.1.5 分解能
平均化イネーブル	SAR_CHANx_CONFIG [10]	20.3.4.1 平均化
DSI 出力イネーブル	SAR_CHANx_CONFIG [30]	20.3.11.8 DSI 出力イネーブル

SUB_RESOLUTION (SAR_SAMPLE_CTRL[0]) は、8 ビットと 10 ビットの予備分解能のどちらかを選択することが可能です。分解能 (SAR_CHANx_CONFIG [9]) は、12 ビットのデフォルト分解能と予備分解能のどちらかを決定することができます。平均化が有効にされる場合 SUB_RESOLUTION は無視され、分解能は最大 12 ビットまでに固定されます。

表 20-12. 分解能

平均化	SUB_RESOLUTION SAR_SAMPLE_CTRL[0]	レジスタ モード分解能 SAR_CHANx_CONFIG [9]	チャネル分解能
オフ	0	1	8 ビット
オフ	1	1	10 ビット
オフ	0	0	12 ビット
オフ	1	0	12 ビット
オン	X	X	12 ビット

20.3.10.4 チャネル イネーブル

CHAN_EN レジスタは各チャネルを個別に有効化することが可能です。次のトリガーが動作する場合、すべての有効化チャネルはスキャンされます。トリガーの後、チャネル イネーブルは次のスキャンの準備のために、直ちに更新されます。これは進行中のスキャンに影響を与えません。これはルールの例外であることに注意してください。その他のすべてのコンフィギュレーション (グローバルまたはチャネル) はスキャンが進行中に更新しないようにします。

20.3.10.5 割り込みマスク

割り込みソースは 6 つあります。すべての割り込みソースには割り込みマスクがあります。

- スキャンの終了の割り込み
- オーバーフロー割り込み
- 衝突割り込み
- 注入変換の終了の割り込み
- 範囲検出の割り込み
- 飽和検出の割り込み

各割り込みには、割り込み要求レジスタ (INTR、SATURATE_INTR、RANGE_INTR)、ソフトウェア割り込みセットレジスタ (INTR_SET、SATURATE_INTR_SET、RANGE_INTR_SET)、割り込みマスク レジスタ (INTR_MASK、SATURATE_INTR_MASK、RANGE_INTR_MASK)、および割り込み要求マスク結果レジスタ (INTR_MASKED、SATURATE_INTR_MASKED、RANGE_INTR_MASKED) があります。割り込み要因レジスタは、現在保留中の SAR 割り込みのあらましが分かるように追加され、ISR がこのレジスタを読み出すことで割り込み要因を判定します。

詳細は 20.3.5 割り込みを参照してください。

20.3.10.6 トリガー

A/D 変換を開始する方法は 3 つあり、以下の通りです。

- ファームウェア トリガー : SAR_START_CTRL [0]
- DSI トリガー : dsi_trigger
- 連続トリガー : SAR_SAMPLE_CTRL [16]

詳細は 20.3.6 トリガーを参照してください。

20.3.10.7 各割り込みの終了後、データを取り込みます

各スキャンの終了後の結果レジスタからデータを読み出すよう確認します。または次のスキャンのコンフィギュレーションによりデータは変更されることがあります。

16 ビットのデータ レジスタは、最大 8 チャネルまで二重バッファの実装に使用されます (注入チャネルには二重バッファがありません)。二重バッファは、各チャネルごとに 1 つのワーク レジスタと 1 つの結果レジスタを持ちます。データはこのチャネルをサンプルした直後、ワーク レジスタに書き込まれます。そのデータは、このスキャンのすべての有効にされたチャネルがサンプルされた後、ワーク レジスタから結果レジスタにコピーされます。

現在のスキャンでサンプルされた、対応するワーク データが有効になった後、CHAN_WORK_VALID ビットは設定されます。対応する CHAN_RESULT_VALID はスキャンの終了後、設定されます。CHAN_RESULT_VALID が設定されると、対応する CHAN_WORK_VALID ビットはクリアします。

ファームウェアを簡易にするため、SAR_CHAN_WORK レジスタのビット [31] は、SAR_CHAN_WORK_VALID レジスタ内の対応するビットのミラー ビットです。SAR_CHAN_RESULT のビット [29]、[30]、[31] は、SAR_SATURATE_INTR、SAR_RANGE_INTR および SAR_CHAN_RESULT_VALID レジスタの対応するビットのミラー ビットです。ここでミラーされる割り込みビットは、raw (マスクされない) 割り込みビットであることに注意してください。データ レジスタの読出しにより、ファームウェアがデータが有効であるかどうかを確認することに役立ちます。

DSI 出力が有効にされると、SARSEQ の結果データは UDB によって処理され、チャネル数により異なるチャネルのデータに対する、異なる処理の適用は可能にされます。詳細な説明は 20.3.11.8 DSI 出力イネーブルを参照してください。

20.3.10.8 注入変換

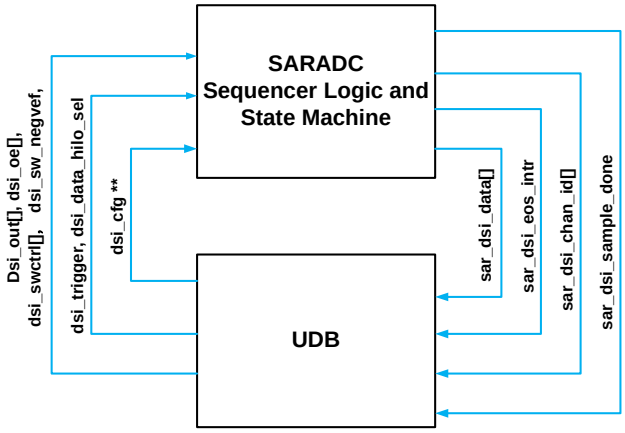
INJ_START_EN (INJ_CHAN_CONFIG [31]) の開始ビットの設定により、注入チャネルをトリガーすることができます。定期的で、自動的なスキャンの衝突を防止するために、INJ_CHAN_CONFIG [30] の設定により、テールゲートを有効化するように推奨されます。それが有効にされる場合、INJ_START_EN は、定期的なチャネルの次のスキャンの最後にスキャンされる注入チャネルを有効化します。

詳細は 20.3.4.4 注入チャネルを参照してください。

20.3.11 DSI モード

DSI 制御モードでは、割り込みマスク、範囲検出の設定、トリガーなどのグローバル コンフィギュレーションを除き、すべての SAR ADC コンフィギュレーションは UDB からの DSI 信号によって行われます。DSI モードとレジスタ モードの主な相違点は、DSI モードがハードウェアに ADC コンフィギュレーションをダイナミックに制御させることです。図 20-14 は DSI 入力および出力信号を指定する SAR ADC ブロック図 (図 20-1) のサブセットです。

図 20-14. DSI 制御モード ブロック図



DSI 制御モードは SAR_CTRL レジスタの DSI_MODE ビットの設定により選択されます。このモードでは、SARSEQ は CHAN_EN、CHAN_CONFIG および INJ_CHAN_CONFIG のすべてのチャネル コンフィギュレーションを無視します。その代わりに DSI 信号を経由してコンフィギュレーションを使用します。

次の DSI 信号は使用されています。

表 20-13. DSI 信号

信号	幅	説明
sar_dsi_sample_done	1	SAR ADC サンプリングが完了したことを示すパルス。スイッチは変換する必要のある次の信号 (SAR ADC の次の出力と同じ) に変更できる
sar_dsi_chan_id_valid	1	チャネル ID の有効信号
sar_dsi_chan_id	4	通常モード : チャネル ID。変換中のチャネルの ID DSI 制御モード : [0]= 飽和検出割り込み [1]= 範囲検出割り込み (データ出力と共に有効)
sar_dsi_data_valid	1	データの有効信号
sar_dsi_data	12	1 チャネルの変換結果 (平均化が有効な場合、平均化結果)。内部平均化結果は 16 ビット幅 dsi_data_hilo_sel=0 の場合 sar_dsi_data[11:0]= sar_data[11:0] dsi_data_hilo_sel=1 の場合 sar_dsi_data[7:0]= sar_data[15:8]、sar_dsi_data[11:8]=< 未定義 >
sar_dsi_eos_intr	1	SARSEQ がすべての有効なチャネルのスキャンを完了したことを示すスキャンの終了割り込み
dsi_out	8	dsi_out[0]=1 の場合 P2.0 は ADC に接続 dsi_out[1]=1 の場合 P2.1 は ADC に接続 … dsi_out[7]=1 の場合 P2.7 は ADC に接続 **注 :** ピンが vplus か vminus のどちらに接続されるかは MUX_SWITCH0 コンフィギュレーションに依存する
dsi_oe	4	dsi_oe[0]=1 の場合 AMUXBUSA は ADC に接続 dsi_oe[1]=1 の場合 AMUXBUSB は ADC に接続 dsi_oe[2]=1 の場合 opamp0 出力は ADC に接続 dsi_oe[3]=1 の場合 opamp1 出力は ADC に接続 **注 :** 信号が vplus か vminus のどちらに接続されるかは MUX_SWITCH0 コンフィギュレーションに依存する
dsi_swctrl[0]	1	SARMUX アナログ スイッチ制御、vssa_kelvin を vminus に接続
dsi_swctrl[1]	1	SARMUX アナログ スイッチ制御、temp_sens を vplus に接続

表 20-13. DSI 信号

信号	幅	説明
dsi_sw_negvref	1	SAR ADC 内部スイッチ制御、V_{REF} 入力を－入力 (NEG) に接続
dsi_cfg_st_sel	2	DSI 制御モードのコンフィギュレーション制御 : 4 つのグローバルなサンプル時間オプションから 1 つを選択
dsi_cfg_average	1	DSI 制御モードのコンフィギュレーション制御 : 平均化を有効にする
dsi_cfg_resolution	1	DSI 制御モードのコンフィギュレーション制御 : 0 = 12 ビット分解能 1 = グローバルに設定された予備分解能 (8 ビットか 10 ビット)
dsi_cfg_differential	1	DSI 制御モードのコンフィギュレーション制御 : 0 = シングルエンド、1 = 差動
dsi_trigger	1	SARSEQ による有効なチャネルのスキャンを開始するためのトリガー
dsi_data_hilo_sel	1	sar_dsi_data[7:0] に上位か下位バイト出力を選択。この信号は完全に非同期 (クロックと無関係に sar_dsi_data に影響を与える)

20.3.11.1 ファームウェア アナログ配線

DSI モードでは DSI 信号およびファームウェアはアナログ配線を実装することができます。レジスタ モードと同様で、ファームウェア制御はレジスタ コンフィギュレーションに関わらず、いつも有効です。ファームウェア制御の詳細については 20.3.2.1 アナログ配線を参照してください。

20.3.11.2 DSI アナログ配線

UDB ブロックからの DSI 信号は SARMUX スイッチの制御に使用されます。DSI 制御モードでは、SARSEQ はシーケンサーからのスイッチ イネーブルをいずれも出力しません。図 20-4 は DFT スイッチ (テスト用の設計) を除き、どのスイッチも制御することができることを示します。従って SAR ADC の－と＋入力は DSI モードでいずれのスイッチにも接続することが可能です。

DSI 信号の他に、レジスタに適切なハードウェアおよびファームウェアの制御ビットを設定する必要があります。これらのレジスタおよび信号は、SAR_MUX_SWITCH0 [n] = 1 および SAR_MUX_SWITCH_HW_CTRL[n] = 1 を含みます。V_{REF} は－入力に接続される時、DSI 信号を除き、SAR_CTRL [11:9] = 7 (ファームウェア制御領域) および SAR_CTRL [13] = 1 (ハードウェア制御ビット) を設定します。

DSI 信号は、シングルエンド モードでは dsi_swctrl[0] および dsi_sw_neg v_{REF} を通じて、SAR ADC の－端子を制御します。NEG_SEL (SAR_CTRL[11:9]) が設定される場合、NEG_SEL=7 は使用可能であり、他の値は無視されます。

表 20-14 は DSI 信号を示します。

表 20-14. DSI アナログ配線

信号	幅	説明
dsi_out	8	dsi_out[0]=1 の場合 P2.0 は ADC に接続 dsi_out[1]=1 の場合 P2.1 は ADC に接続 … dsi_out[7]=1 の場合 P2.7 は ADC に接続 **注 :** ピンが vplus か vminus のどちらに接続されるかは MUX_SWITCH0 コンフィギュレーションに依存する
dsi_oe	4	dsi_oe[0]=1 の場合 AMUXBUSA は ADC に接続 dsi_oe[1]=1 の場合 AMUXBUSB は ADC に接続 dsi_oe[2]=1 の場合 sarbus0 出力は ADC に接続 dsi_oe[3]=1 の場合 sarbus1 出力は ADC に接続 **注 :** 信号が vplus か vminus のどちらに接続されるかは MUX_SWITCH0 コンフィギュレーションに依存する
dsi_swctrl[0]	1	SARMUX アナログ スイッチ制御、V_{SSA} を vminus に接続
dsi_swctrl[1]	1	SARMUX アナログ スイッチ制御、温度センサーを vplus に接続
dsi_sw_negvref	1	SAR ADC 内部スイッチ制御、V_{REF} 入力を－入力 (NEG) に接続

20.3.11.3 グローバル SARSEQ コンフィギュレーション

グローバル コンフィギュレーションは、レジスタ制御モードおよび DSI 制御モードの両方に適用されます。詳細は 20.3.10.2 グローバル SARSEQ コンフィギュレーションを参照してください。

20.3.11.4 DSI チャネル コンフィギュレーション

DSI 制御モードではチャネル 0 のみが使用可能です。チャネル 0 のコンフィギュレーションは表 20-15 に示すように、DSI 信号によって行われます。CHAN_CONFIG および INJ_CHAN_CONFIG 内の CHAN_EN とチャネル コンフィギュレーションは無視されます。

dsi_cfg_* 信号は、DSI_SYNC_CONFIG の設定により、SAR クロック ドメイン (実は clk_hf) に任意に同期化されます。SAR を低周波数で動作させる場合、同期回路のバイパスが必要になることがあります。

表 20-15. DSI チャネル コンフィギュレーション

信号	幅	コンフィギュレーション	説明
dsi_cfg_st_sel	2	取得時間	DSI 制御モードのコンフィギュレーション制御 : 4 つのグローバルなサンプル時間オプションから 1 つを選択
dsi_cfg_average	1	平均化イネーブル	DSI 制御モードのコンフィギュレーション制御 : 平均化を有効にする
dsi_cfg_resolution	1	分解能	DSI 制御モードのコンフィギュレーション制御 : 0: 12 ビット分解能 1: グローバルに設定された分解能ビット SUB_RESOLUTION を使用 (8 ビットか 10 ビット)
dsi_cfg_differential	1	差動／シングルエンド	DSI 制御モードのコンフィギュレーション制御 : 0: シングルエンド 1: 差動

20.3.11.5 割り込み

SAR ADC 割り込みの概要については 244 ページの割り込みマスクを参照してください。すべての割り込みマスクはレジスタ制御モードで機能します。すべての割り込みが DSI で送信されるのではなく、SATURATE_INTR、RANGE_INTR および EOS_INTR は DSI 信号を経由して送信されます。

- DSIモードではチャネル0のみが有効ですので、データと共に、SATURATE_INTR は dsi_chan_id[0] の出力であり、SATURATE_INTR[0] は DSI 制御モードでは設定されます。
- DSIモードではチャネル0のみが有効ですので、データと共に、RANGE_INTR は dsi_chan_id[1] の出力であり、RANGE_INTR[0] は DSI 制御モードでは設定されます。
- チャネル イネーブルは無視されます。1 つのトリガー当たり 1 回の変換が行われるということです。EOS_INTR は変換ごとに生成されます。
- EOS_INTR は、DSI 信号である sar_dsi_eos_intr (dsi_-data_valid のコピー) の経由で常に送信されます。

表 20-16 は DSI 信号経由で送信される割り込みの一覧表です。

表 20-16. DSI 信号割り込み

信号	幅	説明
sar_dsi_chan_id	4	レジスタ モード : チャネル ID (変換中のチャネルの ID) DSI 制御モード : [0]= 飽和検出割り込み [1]= 範囲検出割り込み (データ出力と共に有効)
sar_dsi_eos_intr	1	SARSEQ がすべての有効なチャネルのスキャンを完了したことを示すスキャンの終了割り込み

20.3.11.6 トリガー

通常、DSI 制御モードは DSI トリガーと共に使用されます。ただしファームウェア トリガーと連続トリガーなどのその他のトリガー ソースも対応します。トリガー コンフィギュレーションは レジスタ制御モードのと同様です。詳細は239 ページのトリガーを参照してください。

DSI トリガーの場合、コンフィギュレーション設定 (dsi_cfg_*) およびスイッチ設定は、dsi_trigger が送信されるサイクルに先だって、安定な状態になる必要があります。また sar_dsi_sample_done の立ち上がりエッジが処理されるまで、安定な状態を保つ必要があります。

20.3.11.7 取り込みデータ

結果データおよびチャネル数は sar_dsi_data で送信されます。レジスタ制御モードの dsi_out_en high と同等です。詳

細は 20.3.11.8 DSI 出力イネーブルを参照してください。各変換の後、データはCHAN_WORK0およびCHAN_RESULT0レジスタの両方に書き込まれます。

20.3.11.8 DSI 出力イネーブル

DSI_OUT_ENビット (SAR_CHANx_CONFIG[31]) がセットされる場合、結果データとチャネル数は、結果レジスタに保存されてから、DSI バス (sar_dsi_data, sar_dsi_chan_id) で送信されます。これにより UDB は SARSEQ 結果データを処理し、チャネル数により、異なるチャネルのデータに対する異なる処理の適用が可能になります。

DSI バスで送信されるデータは、結果レジスタに保存されるのと同じフォーマットです。ただしデフォルトで 12LSB のみ送信されます。12 ビット以上が必要でない限り、左アライメントを使用しないようにします。上位の 8LSB を取得するためには、dsi_data_hilo_sel 入力を 1 にセットする必要があります。結果レジスタから 16 ビットのデータを取得するためには、まず dsi_data_hilo_sel = 0 を設定し、下位 12 ビットのデータを取得します。次いで dsi_data_hilo_sel = 1 を設定して、上位 8 ビットのデータを取得します。データの重複に対処するために、追加データの処理が必要です。

チャネル数 (sar_dsi_chan_id) は、SAR ADC がそのチャネルのサンプルを完了した後、いち早く送信されます。チャネル数自身は、いくつかの off-chip デバイスの電源を入れたり切ったりすることが可能な GPIO ピンを駆動するために、UDB をトリガーすることができます。これは、同一スキャンで次に続いてスキャンされるチャネルのアナログ入力ピンを駆動します (ここでは長いサンプル時間が役に立ちます)。

変換の終了後、データは 1 サイクルで送信されることに注意してください。チャネル数、データおよびそれらの有効信号は DSI バス上のシステム クロック サイクルで維持されます。

表 20-17. DSI 出力信号

信号	幅	説明
sar_dsi_sample_done	1	SAR ADC サンプリングが完了したことを示すパルス。スイッチは変換する必要のある次の信号 (SAR ADC の次の出力と同じ) に変更できる
sar_dsi_chan_id_valid	1	チャネル ID の有効信号
sar_dsi_chan_id	4	通常モード : チャネル ID。変換中のチャネルの ID DSI 制御モード : [0]= 飽和検出割り込み [1]= 範囲検出割り込み (データ出力と共に有効)
sar_dsi_data_valid	1	データの有効信号
sar_dsi_data	12	1 チャネルの変換結果 (平均化が有効な場合、平均化結果)。内部平均化結果は 16 ビット幅 dsi_data_hilo_sel=0 の場合 sar_dsi_data[11:0]= sar_data[11:0] dsi_data_hilo_sel=1 の場合 sar_dsi_data[7:0]= sar_data[15:8]、sar_dsi_data[11:8]=< 未定義 >
sar_dsi_eos_intr	1	SARSEQ がすべての有効なチャネルのスキャンを完了したことを示すスキャンの終了割り込み
dsi_data_hilo_sel	1	sar_dsi_data[7:0] に上位か下位バイト出力を選択。この信号は完全に非同期 (クロックと無関係に sar_dsi_data に影響を与える)

20.3.12 アナログ配線のコンフィギュレーション例

表 20-18 にシーケンサー制御、ファームウェア制御と DSI 制御のピン、信号選択のいくつかの例を示します。

表 20-18. アナログ配線のコンフィギュレーション例

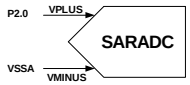
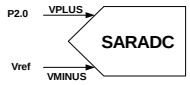
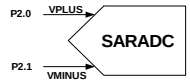
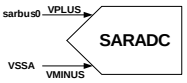
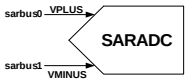
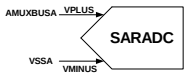
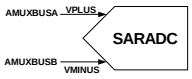
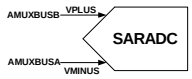
	シーケンサー制御	ファームウェア制御	DSI 制御
	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) SWITCH_DISABLE = 0 (CTRL[30]) PORT_ADDR = 0 (CHANx_CONFIG[6:4]) PIN_ADDR = 0 (CHANx_CONFIG[2:0]) NEG_SEL = 0 (CTRL [11:9]) MUX_SWITCH0[0] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH_HW_CTRL[16]= 1	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) SWITCH_DISABLE = 1 (CTRL[30]) MUX_SWITCH0[0] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[0] = 0 MUX_SWITCH_HW_CTRL[16] = 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 0 dsi_out [0] =1 dsi_swctrl[0]=1 MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH_HW_CTRL[16]=1 MUX_SWITCH0 [16] = 1
	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) SWITCH_DISABLE = 0 (CTRL[30]) PORT_ADDR = 0 (CHANx_CONFIG[6:4]) PIN_ADDR = 0 (CHANx_CONFIG[2:0]) NEG_SEL = 7 (CTRL [11:9]) MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0]=1 HW_CTRL_NEGVREF =1 (CTRL[13])	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) SWITCH_DISABLE = 1 (CTRL[30]) MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] =0 NEG_SEL = 7 (CTRL [11:9]) HW_CTRL_NEGVREF =0 (CTRL[13])	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 0 MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] = 1 dsi_out [0] =1 dsi_sw_negvref =1 HW_CTRL_NEGVREF =1 (CTRL[13])
	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) SWITCH_DISABLE = 0 (CTRL[30]) PORT_ADDR = 0 (CHANx_CONFIG[6:4]) PIN_ADDR = 0 or PIN_ADDR = 1 (CHANx_CONFIG[2:0]) MUX_SWITCH0[0] = 1 MUX_SWITCH0[9] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH_HW_CTRL[1] = 1	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) SWITCH_DISABLE = 1 (CTRL[30]) MUX_SWITCH0[0] = 1 MUX_SWITCH0[9] = 1 MUX_SWITCH_HW_CTRL[0] = 0 MUX_SWITCH_HW_CTRL[1] = 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 1 dsi_out [0] =1 dsi_out [1] =1 MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH0 [9] = 1 MUX_SWITCH_HW_CTRL[1]=1
	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) SWITCH_DISABLE = 0 (CTRL[30]) PORT_ADDR = 1 (CHANx_CONFIG[6:4]) NEG_SEL = 0 (CTRL [11:9]) MUX_SWITCH0[22] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[22] =1 MUX_SWITCH_HW_CTRL[16] =1 **注 :** sarbus1 と VPLUS の接続はポート／ピン制御で非対応	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) SWITCH_DISABLE = 1 (CTRL[30]) MUX_SWITCH0[22] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[22] = 0 MUX_SWITCH_HW_CTRL[16] = 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 0 dsi_oe [2] =1 dsi_swctrl[0]=1 MUX_SWITCH0 [16] = 1 MUX_SWITCH0[22] = 1 MUX_SWITCH_HW_CTRL[16]=1 MUX_SWITCH_HW_CTRL[22] =1

表 20-18. アナログ配線のコンフィギュレーション例 (続き)

	シーケンサー制御	ファームウェア制御	DSI 制御
	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) **SWITCH_DISABLE = 0** (CTRL[30]) PORT_ADDR = 1 **(**CHANx_CONFIG[6:4]) MUX_SWITCH0[22] = 1 MUX_SWITCH0[25] = 1 MUX_SWITCH_HW_CTRL[22]=1 MUX_SWITCH_HW_CTRL[23]=1	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) SWITCH_DISABLE = 1 (CTRL[30]) MUX_SWITCH0[22] = 1 MUX_SWITCH0[25] = 1 MUX_SWITCH_HW_CTRL[22] = 0 MUX_SWITCH_HW_CTRL[23] = 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 1 dsi_oe [2] = 1 dsi_oe [3] = 1 MUX_SWITCH0[22] = 1 MUX_SWITCH0[25] = 1 MUX_SWITCH_HW_CTRL[22]=1 MUX_SWITCH_HW_CTRL[23]=1
	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) **SWITCH_DISABLE = 0** (CTRL[30]) PORT_ADDR = 7 **(**CHANx_CONFIG[6:4]) PIN_ADDR = 2 **(**CHANx_CONFIG[2:0]**)** **NEG_SEL = 0** (CTRL [11:9]) MUX_SWITCH0[18] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[16]= 1	DIFFERENTIAL_EN = 0 (CHANx_CONFIG[8]) **SWITCH_DISABLE = 1** (CTRL[30]) MUX_SWITCH0[18] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[18]= 0 MUX_SWITCH_HW_CTRL[16]= 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 0 dsi_oe [0] = 1 dsi_swctrl[0]=1 MUX_SWITCH0[18] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[16]=1 MUX_SWITCH0 [16] = 1
	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) **SWITCH_DISABLE = 0** (CTRL[30]) PORT_ADDR = 7 **(**CHANx_CONFIG[6:4]) PIN_ADDR = 2 **(**CHANx_CONFIG[2:0]**)** MUX_SWITCH0[18] = 1 MUX_SWITCH0[21] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[19]= 1	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) **SWITCH_DISABLE = 1** (CTRL[30]) MUX_SWITCH0[18] = 1 MUX_SWITCH0[21] = 1 MUX_SWITCH_HW_CTRL[18]= 0 MUX_SWITCH_HW_CTRL[19]= 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 1 dsi_oe [0] = 1 dsi_oe [1] = 1 MUX_SWITCH0[18] = 1 MUX_SWITCH0[21] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[19]= 1
	非対応。 差動ペアはポート／ピン制御で固定されている	DIFFERENTIAL_EN = 1 (CHANx_CONFIG[8]) **SWITCH_DISABLE = 1** (CTRL[30]) MUX_SWITCH0[19] = 1 MUX_SWITCH0[20] = 1 MUX_SWITCH_HW_CTRL[18] =0 MUX_SWITCH_HW_CTRL[19] = 0	DSI_MODE = 1 (CTRL[29]) dsi_cfg_differential = 1 dsi_oe [0] = 1 dsi_oe [1] = 1 MUX_SWITCH0[19] = 1 MUX_SWITCH0[20] = 1 MUX_SWITCH_HW_CTRL[18] =1 MUX_SWITCH_HW_CTRL[19] = 1

20.3.13 温度センサー コンフィギュレーション

温度センシングおよび温度ベースの較正用に 1 個の内蔵温度センサーがあります。差動変換は温度センサー (変換結果が未定義) に使用されません。従って常にシング ルエンド モードで使用されます。リファレンス電圧は内部 1.024V から供給されます。

SAR ADC にピンまたは信号を配線する方法は 3 つあります。表 20-19 に温度センサーを SAR ADC にに配線する方法を示します。MUX_FW_TEMP_VPLUS ビット (SAR_MUX_SWITCH0[17]) をセットすると温度センサーが有効になり、出力が SAR ADC の VPLUS に接続されます。このビットをクリアすると温度センサーのバイアス電流が遮断され、温度センサーが無効になります。

表 20-19. 温度センサーと SAR ADC の接続

制御方法	設定
シーケンサー	DIFFERENTIAL_EN = 0 (SAR_CHANx_CONFIG[8]) VREF_SEL = 0 (SAR_CTRL[6:4]) PORT_ADDR = 7 (SAR_CHANx_CONFIG[6:4]) PIN_ADDR = 0 (SAR_CHANx_CONFIG[2:0]) SWITCH_DISABLE = 0 (SAR_CTRL[30]) SAR_MUX_SWITCH0[16] = 1 SAR_MUX_SWITCH0[17] = 1 SAR_MUX_SWITCH_HW_CTRL[16]= 1 SAR_MUX_SWITCH_HW_CTRL[17]= 1 NEG_SEL = 0 (SAR_CTRL [11:9]) 0 で上書き[a]
ファームウェア	DIFFERENTIAL_EN = 0 (SAR_CHANx_CONFIG[8]) VREF_SEL = 0 (SAR_CTRL[6:4]) SWITCH_DISABLE = 1 (SAR_CTRL[30]) SAR_MUX_SWITCH0[16] = 1 SAR_MUX_SWITCH0[17] = 1 SAR_MUX_SWITCH_HW_CTRL[16]= 0 SAR_MUX_SWITCH_HW_CTRL[17]= 0 NEG_SEL = 0 (SAR_CTRL [11:9]) 0 で上書き [a]
DSI	SWITCH_DISABLE = 1 (SAR_CTRL[30]) VREF_SEL = 0 (SAR_CTRL[6:4]) DSI 信号を設定 : dsi_cfg_differential=1 dsi_swctrl[1]=1 dsi_swctrl[0]=1 SAR_MUX_SWITCH0[16] = 1 SAR_MUX_SWITCH0[17] = 1 SAR_MUX_SWITCH_HW_CTRL[16]= 1 SAR_MUX_SWITCH_HW_CTRL[17]= 1 NEG_SEL = 0 (SAR_CTRL [11:9]) 0 で上書き [a]

a. 温度センサーでは NEL_SEG (SAR_CTRL [11:9]) を「0」で上書きします。

20.4 レジスタ

レジスタ名	オフセット	数量	幅	説明
SAR_CTRL	0x0000	1	32	グローバル コンフィギュレーション レジスタ アナログ制御レジスタ
SAR_SAMPLE_CTRL	0x0004	1	32	グローバル コンフィギュレーション レジスタ サンプル制御レジスタ
SAR_SAMPLE_TIME01	0x0010	1	32	グローバル コンフィギュレーション レジスタ サンプル時間仕様 ST0 と ST1
SAR_SAMPLE_TIME23	0x0014	1	32	グローバル コンフィギュレーション レジスタ サンプル時間仕様 ST2 と ST3
SAR_RANGE_THRES	0x0018	1	32	グローバル範囲検出閾値レジスタ
SAR_RANGE_COND	0x001C	1	32	グローバル範囲検出モード レジスタ
SAR_CHAN_EN	0x0020	1	32	チャネルのイネーブル ビット
SAR_START_CTRL	0x0024	1	32	開始制御レジスタ (ファームウェア トリガー)
SAR_CHAN_CONFIG	0x0080	8	32	チャネル コンフィギュレーション レジスタ
SAR_CHAN_WORK	0x0100	8	32	チャネル ワーク データ レジスタ
SAR_CHAN_RESULT	0x0180	8	32	チャネル結果データ レジスタ
SAR_CHAN_WORK_VALID	0x0200	1	32	チャネル ワーク データ レジスタの有効ビット
SAR_CHAN_RESULT_VALID	0x0204	1	32	チャネル結果データ レジスタの有効ビット
SAR_STATUS	0x0208	1	32	内部 SAR レジスタの状態 (デバッグ用)
SAR_AVG_STAT	0x020C	1	32	平均化の状態 (デバッグ用)
SAR_INTR	0x0210	1	32	割り込み要求レジスタ
SAR_INTR_SET	0x0214	1	32	割り込み要求セット レジスタ
SAR_INTR_MASK	0x0218	1	32	割り込みマスク レジスタ
SAR_INTR_MASKED	0x021C	1	32	割り込み要求マスク レジスタ : 値が 0 以外の場合 NVIC への SAR 割り込み信号は HIGH。読み出すとレジスタは割り込み要求とマスク レジスタの間のビット単位の AND を反映
SAR_SATURATE_INTR	0x0220	1	32	飽和割り込み要求レジスタ
SAR_SATURATE_INTR_SET	0x0224	1	32	飽和割り込み要求セット レジスタ
SAR_SATURATE_INTR_MASK	0x0228	1	32	飽和割り込みマスク レジスタ
SAR_SATURATE_INTR_MASKED	0x022C	1	32	飽和割り込み要求マスク レジスタ
SAR_RANGE_INTR	0x0230	1	32	範囲検出割り込み要求レジスタ
SAR_RANGE_INTR_SET	0x0234	1	32	範囲検出割り込み要求セット レジスタ
SAR_RANGE_INTR_MASK	0x0238	1	32	範囲検出割り込みマスク レジスタ
SAR_RANGE_INTR_MASKED	0x023C	1	32	範囲割り込み要求マスク レジスタ
SASR_INTR_CAUSE	0x0240	1	32	割り込み原因レジスタ
SAR_INJ_CHAN_CONFIG	0x0280	1	32	注入チャネル コンフィギュレーション レジスタ
SAR_INJ_RESULT	0x0290	1	32	注入チャネル結果レジスタ
SAR_MUX_SWITCH0	0x0300	1	32	SARMUX ファームウェア スイッチ制御
SAR_MUX_SWITCH_CLEAR0	0x0304	1	32	SARMUX ファームウェア スイッチ制御クリア
SAR_MUX_SWITCH_HW_CTRL	0x0340	1	32	SARMUX スイッチ ハードウェア制御
SAR_MUX_SWITCH_STATUS	0x0348	1	32	SARMUX スイッチ ステータス
SAR_PUMP_CTRL	0x0380	1	32	スイッチ ポンプ制御

21. 低消費電力コンパレーター

PSoC® 4 デバイスは 2 個の低消費電力コンパレーターがあります。これらのコンパレーターはストップ モードを除く電力モードで高速アナログ信号の比較を可能にします。電力モードの詳細については 81 ページの電力モードを参照してください。＋入力と－入力は専用の GPIO ピンまたは AMUXBUS-A/AMUXBUS-B に接続することが可能です。コンパレーター出力は、ステータス レジスタを介して、CPU に読み込まれたり、割り込みまたは復帰ソースとして使用されたり、GPIO の処理または配線のために DSI に入力されたりします。

21.1 特長

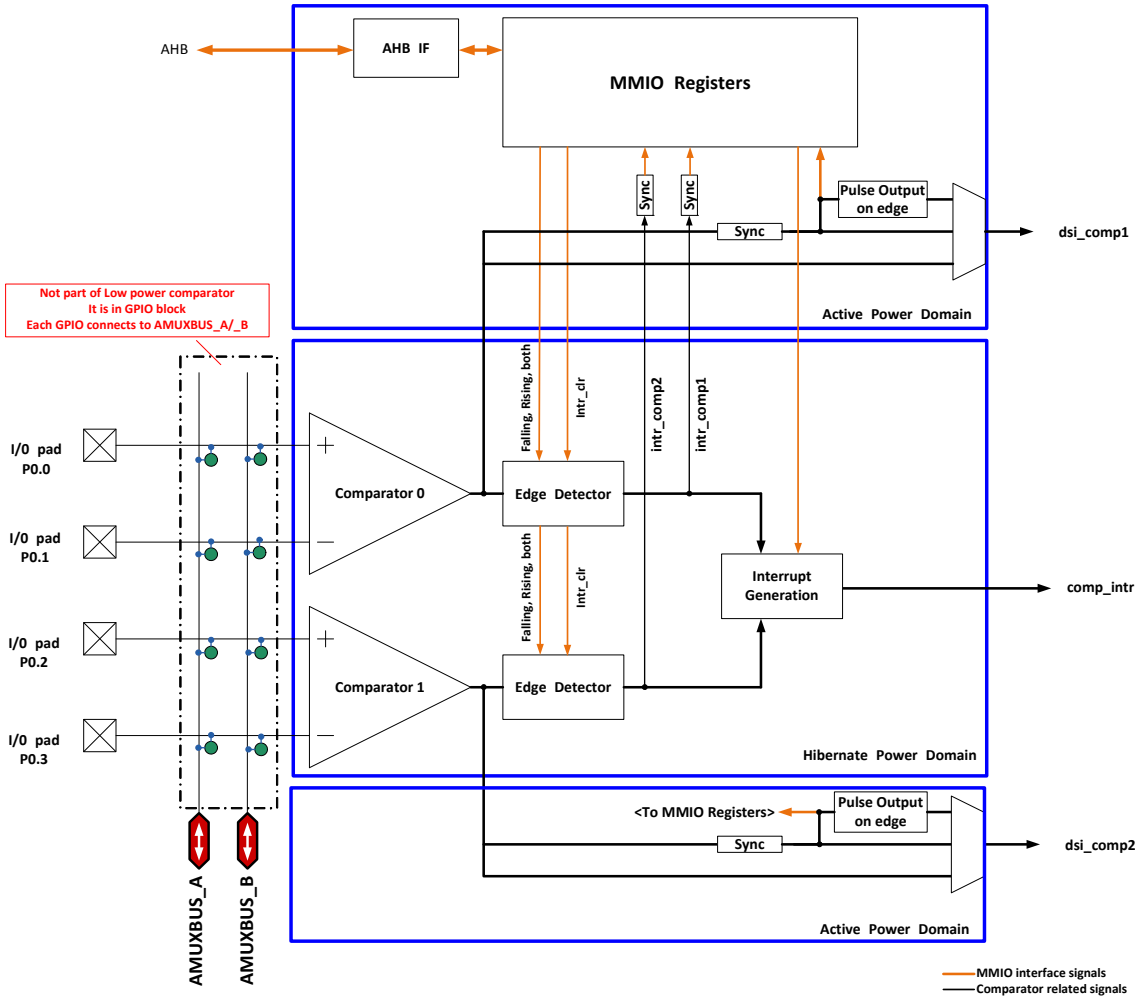
PSoC 4 コンパレーターの主な特長は次の通りです。

- プログラム可能な＋入力と－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 (4 µA 未満)
- 10mV の入力ヒステリシス
- 小さな入力オフセット電圧 (調整後 4mV 未満)
- コンパレーター出力によるディープスリープモード、ハイバネート モードからの復帰

21.2 ブロック図

図 21-1 低消費電力コンパレーターのブロック図を示します。

図 21-1. 低消費電力コンパレーター ブロック図



21.3 動作原理

これからPSoC 4の低消費電力コンパレーターの動作を説明します。入力コンフィギュレーション、消費電力と動作速度モード、出力と割り込みのコンフィギュレーション、ヒステリシス、ハイバネートからの復帰、コンパレーター クロックおよびオフセット調整の項目があります。

21.3.1 入力コンフィギュレーション

コンパレーターへの入力は以下の通りです。

- 専用の外部ピンから＋入力と－入力に配線。
- 外部ピンからの入力と、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＋入力と－入力、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＋入力と－入力に入力。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 21-1 で、P0.0 と P0.1 はコンパレーター 0 の＋入力と－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注意してください。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注意してください。このようにしてコンパレーターの接続は、対応する入力ピンを介して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、これらの入力ピン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。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 59 ページの I/O システムを参照してください。

21.3.2 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 21-1)。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入力は対

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 21-1 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方から選択します。

各コンパレーターの出力は、DSI を介してピンまたはその他のブロックに個別に配線することが可能です (図 21-1 の dsi_comp1/dsi_comp2 信号)。DSI へのコンパレーター出力は、SYSCLK の非同期、同期に、またはコンパレーター出力の立ち上がりエッジで同期されたパルス出力に設定することが可能です。LPCOMP_CONFIG レジスタの DSI_BYPASS1 ビット [16] と DSI_LEVEL1 ビット [17] は、DSI へのコンパレーター 0 の出力設定に使用されます。同様に、LPCOMP_CONFIG レジスタの DSI_BYPASS2 ビット [20] と DSI_LEVEL2 ビット [21] は、DSI へのコンパレーター 1 の出力設定に使用されます。詳細は表 21-1 を参照してください。

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 21-1 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 21-1 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 47 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。アクティブとスリープモードでは、各コンパレーターのトリガーを個別に処理するために、dsi_comp1/dsi_comp2 出力は UDB のマップされている割り込みに配線することが可能です。ただし UDB/DSI の配線はディープスリープとハイバネートのモードで使用不可です。

LPCOMP 割り込み (comp1_intr/comp2_intr) は SYSCLK に同期されます。dsi_comp1/dis_comp2 のクリアは comp1_intr/comp2_intr と同期します。

アクティブとスリープのモードでは、dsi_comp1/dsi_comp2 は、同期化、または非同期化の UDB の DSI 配線で GPIO またはその他のブロックに配線することができます。UDB DSI 出力には追加のシンクロナイザーがあります。DSI 信号の同期化の詳細については、137 ページの UDB を参照してください。ディープスリープとハイバネートのモードでは、UDB の電源がオフになりますので、この配線は使用できません。さらに dsi_comp1/dsi_comp2 をその他の処理のために UDB に配線する場合、タイミングはユーザーのアルゴリズムおよびシンクロナイザーの選択によります。

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。

ディープスリープとハイバネートのモードでは、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 は低消費電力のモードで指定の信号をモニターすることが可能です。

表 21-1. 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 [Bit_Pos]	ビット名 (Bit_Name)	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_CONFIG[16]	DSI_BYPASS1	DSI 出力用のコンパレーター 0 バイパス出力の同期化 0: SYSCLK に同期した出力 1: バイパス／非同期出力

表 21-1. 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 [Bit_Pos]	ビット名 (Bit_Name)	説明
LPCOMP_CONFIG[17]	DSI_LEVEL1	コンパレーター 0 の DSI 出力レベル 0: パルス出力 - 立ち上がりエッジで 2 つの SYSCLK サイクルに相当する幅のあるパルスを生成 1: レベル
LPCOMP_CONFIG[20]	DSI_BYPASS2	DSI 出力用のコンパレーター 1 バイパス出力の同期化 0: SYSCLK に同期した出力 1: バイパス／非同期出力
LPCOMP_CONFIG[21]	DSI_LEVEL2	コンパレーター 1 の DSI 出力レベル 0: パルス出力 - 立ち上がりエッジで 2 つの SYSCLK サイクルに相当する幅のあるパルスを生成 1: レベル
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します。「1」を書き込んで、割り込みをクリアする
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します。「1」を書き込んで、割り込みをクリアする
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する
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する

21.3.3 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 つのモードで動作することができます。

- 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高く、超低消費電力のモードで最も少ない。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。様々な電力の設定の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 21-2 のように、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、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。それを防止するため、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 21-2. コンパレーター動作モードの選択ビット

レジスタ [Bit_Pos]	ビット名 (Bit_Name)	説明
LPCOMP_CONFIG[1:0]	MODE1	コンパレーター 0 の電力モード選択 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

21.3.4 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が振動することを回避するのに役立ちます。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 21-3 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 21-3. ヒステリシス制御ビット HYST1 と HYST2

レジスタ [Bit_Pos]	ビット名 (Bit_Name)	説明
LPCOMP_CONFIG[2]	HYST1	コンパレーター 0 に 10 mV 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 10 mV 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

21.3.5 省電力モードからの復帰

コンパレーターは、スリープ、ディープスリープおよびハイバネートのモードを含むデバイスの省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、ディープスリープおよびハイバネートのモードからデバイスを復帰させることができます。省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。ディープスリープとハイバネートモードで使用できない機能は以下の通りです。

- AMUXBUS の接続を含む比較
- DSI を経由するコンパレーター出力配線

ディープスリープとハイバネートモードでは、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を選択するのに使用されます。

21.3.6 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。

21.3.7 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。調整は 2 箇所で行います。最初に同相電圧を 0.1 V にして調整し、次に同相電圧を V_{DD}–0.1 V にして調整します。オフセット電圧は、0.1 V ～ 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。通常の用途で再調整は推奨しません。

特定の入力同相モード電圧でより厳しい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実行してください。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実行します。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低速モード動作の場合だけです。

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 であると、オフセットは正になります。出力が High ではないと、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 ビットの調整が完了したら引き続き、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. 3b 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。

21.4 レジスタの概要

表 21-4. 低消費電力コンパレーター レジスタ

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

22. 連続時間ブロックミニ (CTBm)

連続時間ブロック ミニ (CTBm) は、連続時間信号処理系で使用する個別のオペアンプをチップ内に備えます。各 CTBm ブロックは入力／出力コンフィギュレーション用のスイッチ マトリックス、コンパレータとして構成可能な 2 つの同じオペアンプ、各オペアンプ内蔵のチャージ ポンプ、コンパレータ出力ルーティング用のデジタル インタフェース、スイッチ制御および割り込みが含まれます。The PSoC 4100M/4200M ファミリは 2 個の CTBm ブロックの中に 4 個のオペアンプを含みます。また CTBm ブロックはディープスリープ電力モードでも動作します。

22.1 特長

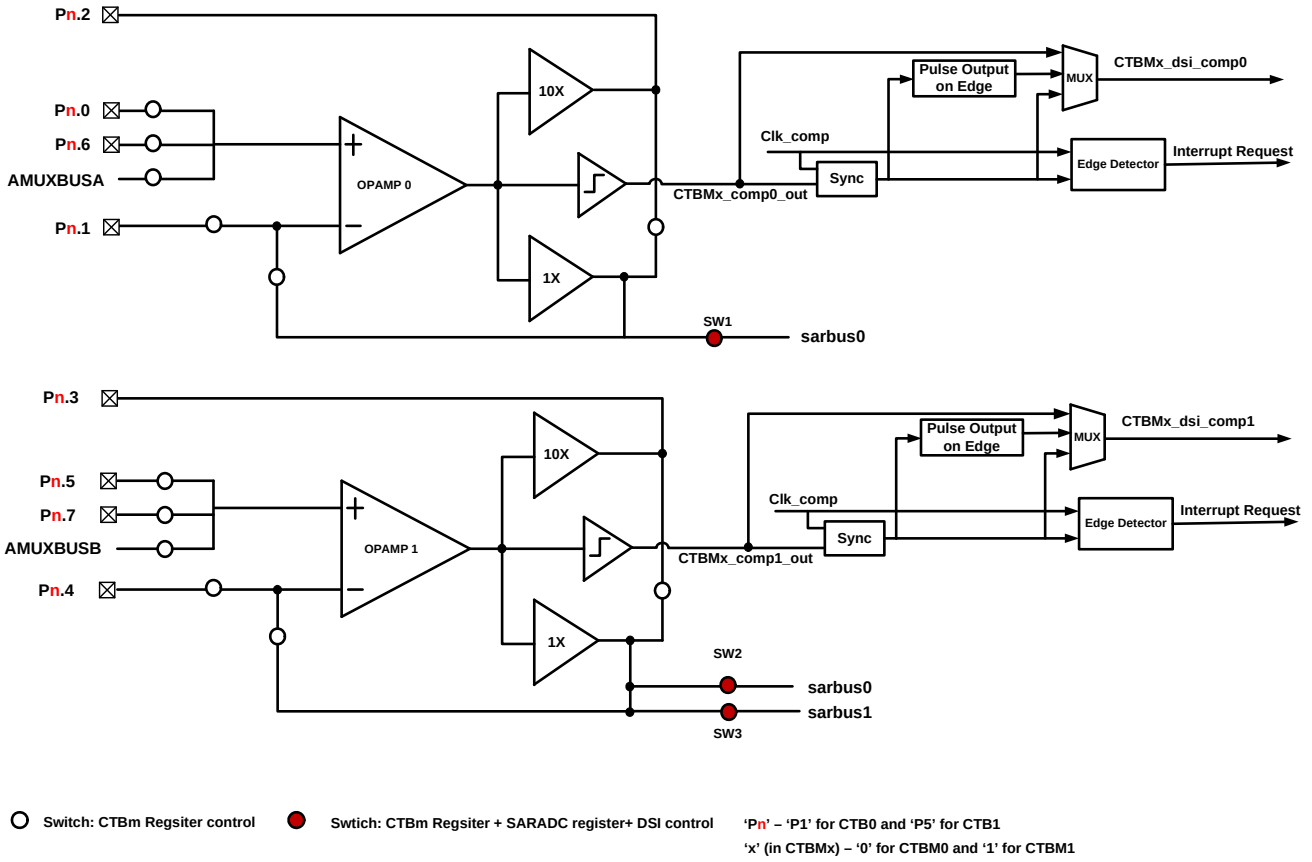
PSoC4 CTBm ブロック内のオペアンプは以下の特長があります。

- 個別、高性能かつ高度に構成可能なオンチップ アンプ
- プログラマブルな電力モード、帯域幅、位相補償、ドライブレベル
- 1mA または 10mA を選択可能な出力駆動能力
- 20pF 負荷で 6MHz のゲイン帯域幅
- 調整による 1mV 以下のオフセット
- オペアンプの電圧フォロア モードをサポートする
- オプションの 10mV ヒステリシスを持つコンパレータ モード
- SAR 入力用のバッファ／プリアンプ
- 1mA の負荷で Vss または Vdda 0.2V 以内にレール ツー レール
- 10mA 負荷用の Vss または Vdda 0.5V 以内でレール ツー レール
- 50pF の負荷で 4V/µs のスルー レート
- ディープスリープ デバイス電力モードでも動作する

22.2 ブロック図

図 22-1 に PSoC 4 デバイスに内蔵される CTBMx (CTMB0 と CTBM1) ブロックのブロック図を示します。

図 22-1. CTBm ブロック図



Note: 10X or 1X output driver cannot be on at the same time.

22.3 動作原理

ブロック図に示すように、各 CTBm は 2 個の同一のオペアンプとスイッチ マトリックスで構成されています。図 22-1 に示すように、それぞれのオペアンプは 1 つの入力ステージを共有する 1 つの入力と 3 つの出力ステージを含みます。出力ステージは Class-A(1X)、Class-AB(10X) またはコンパレータとして動作できます。他の構成可能な機能は電力モードと速度、位相補償およびスイッチ ルーティング制御です。

CTBm ブロックを使用するには必要に応じて、最初に外部コンポーネント (抵抗等) を接続します。次に CTBMx_CTB_CTRL [31] ビットを設定することでブロックを有効にします。ほぼレールツーレールの入力範囲と歪みの少ないコモン モード入力を実現するために、各オペアンプ内に 1 個のチャージ ポンプを備えます。CTBMx で opamp0 に CTBMx_OA_RES0_CTRL [11] ビットをセットし、opamp1 に CTBMx_OA_RES1_CTRL [11] ビットをセットすることで、チャージポンプを有効にすることができます。

オペアンプとチャージ ポンプを有効にし、以下の手順に従ってアンプを初期化してください。

1. 電力モードを構成する
2. ドライブレベルを構成する
3. 位相補償を構成する
4. 入力スイッチを構成する
5. 出力スイッチを構成する (オペアンプが SAR ADC に接続される必要がある場合に特に必要)

以下の手順に従ってコンパレータを初期設定してください。

1. 電力モードを構成する
2. 入力スイッチを構成する
3. 必要に応じてコンパレータ出力回路を構成する - 割り込み生成、DSI 出力等
4. ヒステリシスを構成し、コンパレータを有効にする

22.3.1 電力モードのコンフィギュレーション

オペアンプは低、中、大という 3 つの電力モードで動作することができます。CTBm はオペアンプに流れるリファレンス電流を調整することで消費電力を調整します。電力モードは CTBMx_OA_RESy_CTRL 内の PWR_MODE ビット [1:0] を使用して構成されます。スルーレートとゲイン帯域幅は大消費電力モードでは最大で、低消費電力モードでは最小です。電力モード コンフィギュレーションは 1X モードで

の最大出力駆動能力 (I_{OUT}) にも影響を与えます。詳細は 表 22-1 を参照してください。異なるモードでのゲイン帯域幅、スルー レートおよび I_{OUT} 仕様についていはデバイスデータシートを参照してください。

注：「y」は CTBMx 内の Opamp1/0 を示す (Opamp1 の場合は y = 1、Opamp0 の場合は y = 0)。

22.3.2 ドライブレベルコンフィギュレーション

各オペアンプの出力ドライバーは内部ドライバー (クラス A/1X ドライバー) または外部ドライバー (クラス AB/10X ドライバー) に構成することができます。1X と 10X ドライバーは相互に排他的で同時にアクティブにすることができません。1X 出力ドライバーは低いオンチップ容量や抵抗性負荷を高速に駆動するのに適しています。10X の出力ドライバーは大きな外部静電容量および抵抗負荷を駆動するのに使用します。1X ドライバ出力は sarbus 0/1 に接続され、10X ドライバー出力は外部ピンに接続されます。表 22-1 に示すように各ドライバーは低、中、大消費電力モードがあります。

表 22-1. 出力ドライバーと電力モード

電力モード／I_{OUT} ドライブ能力	CTBMx_OA_RESy_CTRL[1:0]			
	00 (無効)	01 (低)	10 (中)	11 (大)
外部ドライバー (10X)	オフ	10mA	10mA	10mA
内部ドライバー (1X)	オフ	100µA	400µA	1mA

10X と 1X ドライブ能力を選択するために CTBMx_OA_RESy_CTRL[2] ビットが使用されます (0: 1X、1: 10X)。オペアンプの出力が SAR ADC に接続される場合は 1X 出力ドライバーを選択することをお勧めします。オペアンプの出力が外部端子に接続される場合、10X 出力ドライバーを選択します。特殊な場合では出力を 1X 出力ドライバーで外部ピンに、または 10X 出力ドライバーで内部負荷 (例えば、SAR ADC) に接続するために、CTBMx_OAy_SW [21] を「1」に設定することもできます。しかしサイプレスはこの場合、性能を保証できません。

表 22-2 オペアンプのドライブレベルと電力モードを構成するために使用されるビットをまとめます。

表 22-2. CTBM レジスタにおけるドライブレベルと電力モード設定

レジスタ [Bit_Pos]	ビット名 (Bit_Name)	説明
CTBMx_CTB_CTRL[31]	ENABLE	CTBM の電力モードの選択 0: CTBM が無効 1: CTBM が有効
CTBMx_OA_RES0_CTRL [11]	OA0_PUMP_EN	Opamp0 ポンプ イネーブル ビット 0: Opamp0 ポンプが CTBMx で無効 1: Opamp0 ポンプが CTBMx で有効
CTBMx_OA_RES1_CTRL [11]	OA1_PUMP_EN	Opamp1 ポンプ イネーブル ビット 0: Opamp1 ポンプが CTBMx で無効 1: Opamp1 ポンプが CTBMx で有効
CTBMx_OA_RES0_CTRL [1:0]	OA0_PWR_MODE	Opamp0 電力モード選択ビット 00: Opamp0 が CTBMx でオフ 01: Opamp0 が CTBMx で低消費電力モードにある 10: Opamp0 が CTBMx で中消費電力モードにある 11: Opamp0 が CTBMx で大消費電力モードにある
CTBMx_OA_RES1_CTRL [1:0]	OA1_PWR_MODE	Opamp1 電力モード選択ビット 00: Opamp1 が CTBMx でオフ 01: Opamp1 が CTBMx で低消費電力モードにある 10: Opamp1 が CTBMx で中消費電力モードにある 11: Opamp1 が CTBMx で大消費電力モードにある
CTBMx_OA_RES0_CTRL [2]	OA0_DRIVE_STR_SEL	Opamp0 ドライブレベル選択ビット 0: Opamp0 ドライブレベルが CTBMx で 1X 1: Opamp0 ドライブレベルが CTBMx で 10X
CTBMx_OA_RES1_CTRL [2]	OA1_DRIVE_STR_SEL	Opamp1 ドライブレベル選択ビット 0: Opamp1 ドライブレベルが CTBMx で 1X 1: Opamp1 ドライブレベルが CTBMx で 10X

22.3.3 位相補償

各オペアンプは出力負荷に応じて、オペアンプ回路の安定性を最適化することを可能にするプログラマブルな補償コンデンサ ブロックを持ちます。表 22-3 で説明されるように、各オペアンプの補償はそれぞれの CTBMx_OAy_COMP_TRIM レジスタによって制御されます。デバイス データシート内のすべての GBW スルー レートの仕様はすべての補償調整に適用されることに注意してください。

表 22-3. CTMB 内の Opampy (Opamp0 または Opamp1) 位相補償ビット

レジスタ [Bit_Pos]	ビット名 (Bit_Name)	説明
CTBMx_OAy_COMP_TRIM[1:0]	OAy_COMP_TRIM	Opampy 位相補償設定ビット 00: CTBMx での位相補償少、高速、低い安定性 01: CTBMx での位相補償中、バランス速度、バランス安定性 11: CTBMx での位相補償大、低速、高い安定性

22.3.4 スイッチ制御

各 CTBm はオペアンプ入出力を構成するためのスイッチを多く持っています。sarbus0 と sarbus1 を介してオペアンプの出力を SAR ADC に接続するために使用される 3 個のスイッチを除き、ほとんどのスイッチは CTBm レジスタ (CTBMx_OA0_SW、CTBMx_OA1_SW) を構成することで制御されます。その 3 個のスイッチは SAR ADC のレジスタ、CTBm レジスタおよび DSI 信号によって制御されなければなりません。

スイッチはレジスタ CTBMx_OAy_SW 内の対応するビットをセットすることで閉じることができます。スイッチをクリアすると対応するスイッチが開きます。CTBMx_OAy_SW_CLEAR に「1」を書き込むと、CTBMx_OAy_SW 内の対応するビットをクリアすることがあります。スイッチと接続の詳細は PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ TRM を参照してください。

22.3.4.1 入力コンフィギュレーション

アナログ スイッチを介してオペアンプの入力端子をいくつかのオプションから選択することができます。これらのスイッチは外部ピンまたは AMUX バスからオペアンプ入力を接続する、または (バッファ機能のために) 帰還ループを形成するのに役立ちます。各オペアンプは 2 つの AMUXBUS ラインのいずれかに接続するスイッチを持っています。Opamp0 は各 CTBm の AMUXBUS-A に接続し、OPAMP1 は AMUXBUS-B に接続します。

注: 1 つのスイッチのみが入力端子パスのために閉じられる必要があります。そうでなければ、別の入力ソースと一緒に短絡されてしまうかもしれません。

- 非反転入力

  各 CTBm にある opamp0 と opamp1 の両方は、アナログ スイッチにより 2 本の外部ピンと 1 つの AMUXBUS ラインから入力を選択できます。詳細は表 22-4 を参照してください。

表 22-4. 非反転入力選択

	非反転入力	スイッチ制御ビット	説明
Opamp0	AMUXBUSA	CTBMx_OA0_SW [0]	0: スイッチを開く ; 1: スイッチを閉じる
	Pn.0[a]	CTBMx_OA0_SW [2]	0: スイッチを開く ; 1: スイッチを閉じる
	Pn.6	CTBMx_OA0_SW [3]	0: スイッチを開く ; 1: スイッチを閉じる
Opamp1	AMUXBUSB	CTBMx_OA1_SW [0]	0: スイッチを開く ; 1: スイッチを閉じる
	Pn. 5	CTBMx_OA1_SW [1]	0: スイッチを開く ; 1: スイッチを閉じる
	Pn.7	CTBMx_OA1_SW [4]	0: スイッチを開く ; 1: スイッチを閉じる

a. CTBm0 の場合は Pn = P1、CTBm1 の場合は Pn = P5

■ 反転入力
各 CTBm にある opamp0 と OPAMP1 の両方は、CTBMx_OAy_SW レジスタによって制御されている 1 本の外部ピンまたは出力フィードバックをアナログ スイッチにより入力として選択できます。表 22-5 に制御ビットの詳細を記載します。

表 22-5. 反転入力選択

	反転入力	スイッチ制御ビット	説明
Opamp0	Pn.1[a]	CTBMx_OA0_SW [8]	0: スイッチを開く ; 1: スイッチを閉じる
	1X 出力ドライバーによる Opamp0 出力フィードバック	CTBMx_OA0_SW [14]	0: スイッチを開く ; 1: スイッチを閉じる
Opamp1	Pn.4	CTBMx_OA1_SW [8]	0: スイッチを開く ; 1: スイッチを閉じる
	1X 出力ドライバーによる Opamp1 出力フィードバック	CTBMx_OA1_SW [14]	0: スイッチを開く ; 1: スイッチを閉じる

a. CTBm0 の場合は Pn = P1、CTBm1 の場合は Pn = P5

22.3.4.2 出力コンフィギュレーション

各オペアンプの出力は固定のピンに直接接続されています。追加の設定は必要ありません。必要に応じて、3 つのスイッチ (SW1/2/3) を介して sarbus0 または sarbus1 に接続することができます。各 CTBm の opamp0 出力は sarbus0 に接続し、opamp1 出力は sarbus0 か sarbus1 に接続することができます。sarbus0 と sarbus1 はオペアンプ出力を SAR ADC 入力マルチプレクサに接続するのに利用されます。sarbus への 3 つの出力ルーティング スイッチは CTBm、レジスタ SAR ADC レジスタ、DSI 信号によって制御されます。他のスイッチは CTBm レジスタのみによって制御することができます。

その 3 つのスイッチの制御論理は以下の真理値表 (表 22-6、表 22-7、表 22-8) に示されます。PORT_ADDR、PIN_ADDR および DIFFERENTIAL_EN はそれぞれ SAR_CHANx_CONFIG [6:4]、SAR_CHANx_CONFIG [2:0] および SAR_CHANx_CONFIG [2:0] から取得されます。PORT_ADDR が 0 に設定されるまたは PIN_ADDR が 0 に設定されると、SW[n] が「0」に設定されます。SAR レジスタまたは DSI 信号をスイッチを制御するのに使用する際、CTBMx_SW_HW_CTRL のビット [2] または [3] を設定する必要があります。CTBMx_OAy_SW[18]/[19] は他のビットをマスクすることができます – CTBMx_OAy_SW[18]/[19] = 0 の場合、SW[n] = 0。

レジスタ CTBMx__SW_STATUS [30:28] は、スイッチ SW1/2/3 の現在のステータスを示します。

表 22-6. SW1 制御論理の真理値表

PORT_ADDR	PIN_ADDR	CTBMx_SW_HW_CTRL[2]	dsi_out[2]	CTBMx_OA0_SW[18]	SW1
X	X	X	X	0	0
X	0	1	0	1	0
0	X	1	0	1	0
X	X	X	1	1	1
X	X	0	X	1	1
1	2	X	X	1	1

表 22-7. SW2 制御論理の真理値表

DIFFERENTIAL_EN	PORT_ADDR	PIN_ADDR	CTBMx_SW_HW_CTRL[3]	dsi_out[3]	CTBMx_OA0_SW[18]	SW2
X	X	X	X	X	0	0
X	X	0	1	0	1	0
X	0	X	1	0	1	0
1	X	X	X	0	1	0
X	X	X	0	X	1	1
X	X	X	X	X	1	1
0	1	3	X	X	1	1

表 22-8. SW3 制御論理の真理値表

DIFFERENTIAL_EN	PORT_ADDR	PIN_ADDR	CTBMx_SW_HW_CTRL[3]	dsi_out[3]	CTBMx_OA0_SW[18]	SW3
X	X	X	X	X	0	0
X	X	0	1	0	1	0
X	0	X	1	0	1	0
0	X	X	X	0	1	0
X	X	X	0	X	1	1
X	X	X	X	X	1	1
1	1	2	X	X	1	1

22.3.4.3 コンパレータ モード

各オペアンプは、その CTBMx_OA_RESy_CTRL[4] ビットを設定することにより、コンパレータとして構成することができます。コンパレータを有効にすると、補償コンデンサを完全に無効にし、クラス A (1X) とクラス AB (10X) 出力ドライバーをシャットダウンすることに注意してください。これらのコンパレータの主な特長は次の通りです。

- オプションの 10mV 入力ヒステリシス
- 構成可能な電力モード／速度
- オプションの DSI 出力同期化
- 1mV 以下に調整されたオフセット
- 構成可能なエッジ検出 (立ち上がり／立ち下がり／両方／ディスエーブル)

22.3.4.4 コンパレーター コンフィギュレーション

10mV ±5 パーセントのヒステリシスは 1 方向 (LOW から HIGH に) で有効にすることができます。入力ヒステリシスは CTBMx_OA_RESy_CTRL[5] を設定することで有効にすることができます。各 CTBm ブロック内の 2 個のコンパレータも、CTBMx_OA_RESy_CTRL [1:0] を設定することで制御される 3 つの電力モードがあります : 低、中および高。電力モードにより応答時間と消費電力が異なります。電力消費は高速モードでは最大で、超低消費電力モードでは最小です。消費電力と応答時間の正確な仕様は、データシートに記載されています。

コンパレータ出力は、同期化可能で DSI にルーティングされます。コンパレータ クロック (システム AHB クロック) との同期は CTBMx_OA_RESy_CTRL[6] で構成することができます。

comparator0 と comparator1 の出力状態はそれぞれ CTBMx_COMP_STAT[0] と CTBMx_COMP_STAT[16] に格納されます。

表 22-9 は CTBM ブロックのコンパレータ モードを構成するために使用される異なるビットをまとめています。

表 22-9. コンパレータ モードおよびコンフィギュレーション レジスタの設定

レジスタ [Bit_Pos]	ビット名 (Bit_Name)	説明
CTBMx_OA_RESyy_CTRL[4]	OAy_COMP_EN	Opampy コンパレータ イネーブル ビット 0: CTBMx の opampy でコンパレータ モードが無効 1: CTBMx の opampy でコンパレータ モードが有効
CTBMx_OA_RESy_CTRL[5]	OAy_HYST_EN	Opampy コンパレータ イネーブル ビット 0: CTBMx の opampy でヒステリシスが無効 1: CTBMx の opampy でヒステリシスが有効
CTBMx_OA_RESy_CTRL[6]	OAy_BY-PASS_DSI_SYNC	DSI (トリガー) 出力の Opampy バイパス コンパレータ出力同期化 0: 同期化 (レベルまたはパルス) 1: バイパス
CTBMx_OA_RESy_CTRL[7]	OAy_ DSI_LEVEL	Opampy コンパレータ DSI (トリガー) 出力同期レベル 0: パルス 1: レベル

22.3.4.5 コンパレータ割り込み

コンパレータ出力は、割り込みを発生するエッジ (ディスエーブル／立ち上がり／立ち下がり／両方) を検出するために使用されるエッジ検出器ブロックに接続されています。CTBMx_OA_RESy_CTRL[9:8] ビットで構成することができます。

各コンパレータは個別のIRQを持っています。CTBMx_INTR [0] は comparator0 IRQ に使用され、CTBMx_INTR [1] は comparator1 IRQ に使用されます。各 CTBM のコンパレータはそれぞれの個別の IRQ ビットを持っていますが、CPU NVIC では単一の CTBM ISR を共有します。詳細は 47 ページの割り込みをご覧ください。CTBMx_INTR ビットをポーリングすることで、どの CTBM のコンパレータが ISR をトリガーしたかを確認することができます。

割り込みは CTBMx_INTR_MASK レジスタ内にそれぞれ割り込みマスク ビットを持っています。割り込みマスクを LOW に設定すると、対応する割り込みソースが無視されます。CTBMx_INTR レジスタ内の割り込みフラグの論理 AND と CTBMx_INTR_MASK レジスタ内の対応する割り込みマスクが「1」の場合、NVIC への CTBm コンパレータ割り込みがアクティブになります。

CTBMx_INTR ビット [1:0] に「1」を書き込むと、対応する割り込みがクリアされます。

ファームウェアを容易にするため、割り込みフラグと割り込みマスクの交点 (論理 AND) も CTBMx_INTR_MASKED レジスタ内で可能です。

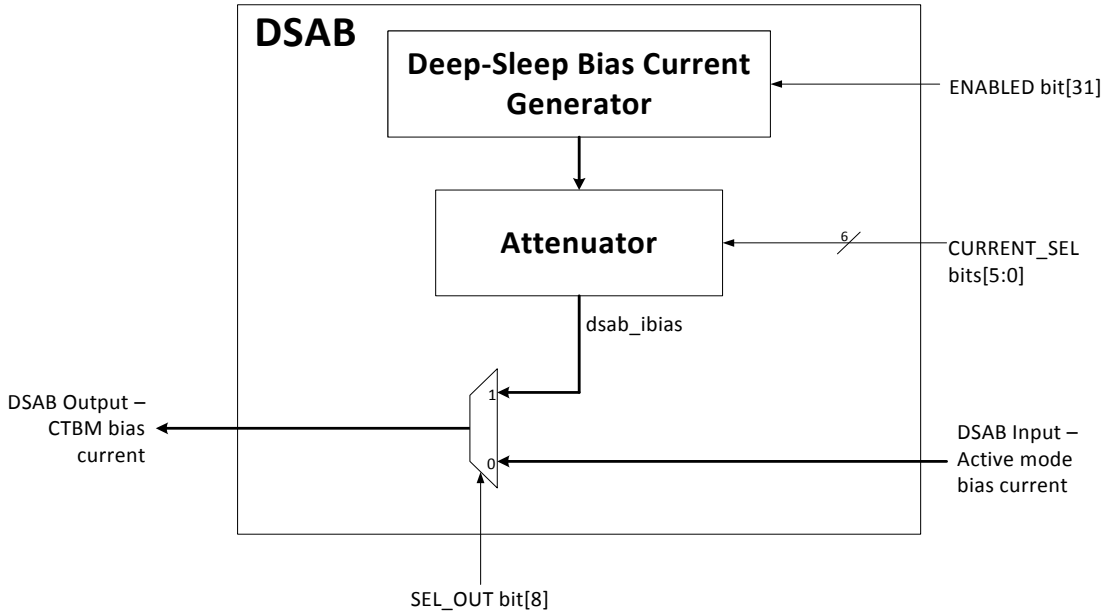
検証およびデバッグ目的のために CTBMx_INTR_SET レジスタ内の各割込みに設定ビットが用意されています。これによりファームウェアが実際のコンパレータ切り替えイベントなしで割り込みを生成することができます。

22.3.4.6 ディープスリープ モード動作

IMO ディープスリープ モードでは、バイアス電流、レファレンス電圧および IMO クロックを供給するブロックがオフにされます。その結果、動作がバイアス電流と IMO クロックに依存する CTBm 機能は利用できません。電力モードと利用できるブロックの詳細は 81 ページの電力モードを参照してください。ディープスリープ モード時に CTBm の機能を維持するために、代替のバイアス電流がディープスリープ アンプ バイアス (DSAB) ブロックと呼ばれる特別なブロックによって生成されます。これにより CTBm 内のオペアンプがディープスリープ モードで機能することができます。

図 22-2 DSAB ブロックのアーキテクチャを示しています。このブロックはアクティブ モード バイアス電流を入力として受給し、オペアンプのバイアス回路に供給されるバイアス電流を出力します。アクティブ モードでは、DSAB ブロックはパススルー ブロックのように動作し、入力から出力にバイアス電流をルーティングします。ディープスリープ モードでは、DSAB は有効の場合、代替バイアス電流を生成し、出力をユーザーが選択した値に減衰し、DSAB の出力にある CTBm にバイアス電流を提供します。DSAB ブロックは無効の場合、出力が常に入力バイアス電流に接続され、代替バイアス電流がディープスリープ時に生成されません。DSAB ブロックが無効の場合、オペアンプはディープスリープ モードでは機能しません。PASS_DSAB_DSAB_CTRL レジスタの ENABLED ビット [31] は、ブロックを有効または無効にします ; CURRENT_SEL ビット [5 : 0] は出力バイアス電流値を選択します。選択された値は CURRENT_SEL × 0.075uA (±5%) です。SEL_OUT ビット [8] は CTBm バイアスに 2 つのバイアス電流のどちらを接続するか制御します。表 22-10 は PASS_DSAB_DSAB_CTRL レジスタのビットコンフィギュレーション設定をまとめたものです。

図 22-2. ディープスリープ アンプのバイアス ブロック図



この機能は、節電のためディープスリープのような低消費電力モードでアクティブ状態を維持するオペアンプ ベースの回路を必要とするデザインに有用です。例えば、心拍数モニターなど常に動作を続けるオペアンプを必要とする電池式システムでは、チップの残りの部分がディープスリープ モードに入り、必要に応じて復帰すれば、消費電力を大幅に節約できます。DSAB ブロックによって供給されるバイアス電流は、バイアス電流の精度と安定性がアクティブ モードに劣ることに注意してください。また DSAB は代替クロックを生成しません。その結果、スイッチとオペアンプ関連チャージ ポンプのいずれもアクティブにされません。このためオペアンプの最大入力コモンモード電圧が約 VDDA - 1.3V に制限されます 。ディープスリープモード時のオペアンプ仕様についてはデバイス データシート を参照してください。

ディープスリープ モードでオペアンプを有効にするには、CTBMx_CTBM_CTB_CTRL レジスタの DEEPSLEEP_ON ビット [30] を設定します。これによりディープスリープ時に CTBMx のオペアンプの両方が有効にされます。CTBm のディープスリープ動作のために、DSAB ブロックを有効にする必要もあります。

表 22-10. DSAB と CTBM ディープスリープ コンフィギュレーション レジスタの設定

レジスタ [Bit_Pos]	ビット名 (Bit_Name)	説明
PASS_DSAB_DSAB_CTRL [5:0]	CURRENT_SEL	dsab_ibias の電流選択；dsab_ibias = CURRENT_SEL × 0.075uA (±5%)
PASS_DSAB_DSAB_CTRL [8]	SEL_OUT	CTBm バイアス電流選択 0: DSAB をバイパスしアクティブ モード バイアス電流を使用 1: dsab_ibias を CTBm バイアス電流として使用する
PASS_DSAB_DSAB_CTRL [31]	ENABLED	DSAB バイアス ジェネレータを有効にする 0: DSAB ブロックが無効にされ、CTBm バイアス電流がアクティブ モード バイアス電流に接続される 1: DSAB ブロックが有効にされ、CTBm バイアス電流が SEL_OUT 信号により制御される
CTBMx_CTBM_CTB_CTRL [30]	DEEPSLEEP_ON	ディープスリープ モードで CTBMx 機能を有効／無効にする 0: 有効 1: 無効

22.4 レジスタの概要

表 22-11. レジスタの概要

レジスタ名	説明
CTBMx_CTRL	グローバル CTBm ブロック イネーブル
CTBMx_OA_RES0_CTRL	Opamp0 制御レジスタ
CTBMx_OA_RES1_CTRL	Opamp1 制御レジスタ
CTBMx_COMP_STAT	コンパレータ ステータス
CTBMx_INTR	割り込み要求レジスタ
CTBMx_INTR_SET	割り込み要求セットレジスタ
CTBMx_INTR_MASK	割り込み要求マスク
CTBMx_INTR_MASKED	マスクされた割り込み要求
CTBMx_OA0_SW	Opamp0 スイッチ制御
CTBMx_OA0_SW_CLEAR	Opamp0 スイッチ制御クリア
CTBMx_OA1_SW	Opamp1 スイッチ制御
CTBMx_OA1_SW_CLEAR	Opamp1 スイッチ制御クリア
CTBMx_SW_HW_CTRL	CTBm ハードウェア制御イネーブル
CTBMx_SW_STATUS	CTBm バス スイッチ制御ステータス
CTBMx_OA0_OFFSET_TRIM	Opamp0 調整制御
CTBMx_OA0_SLOPE_OFFSET_TRIM	Opamp0 調整制御
CTBMx_OA0_COMP_TRIM	Opamp0 調整制御
CTBMx_OA1_OFFSET_TRIM	Opamp1 調整制御
CTBMx_OA1_SLOPE_OFFSET_TRIM	Opamp1 調整制御
CTBMx_OA1_COMP_TRIM	Opamp1 調整制御
PASS_DSAB_DSAB_CTRL	DSAB 制御レジスタ
PASS_DSAB_TRIM	IBIAS 調整レジスタ

23. 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。

23.1 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。

- 32 セグメント、4 コモンまで対応 (多重化比率 1:4)
- タイプ A (標準) およびタイプ B (省電力) の駆動波形に対応
- コモンとセグメントを任意の GPIO ピンに配線できる
- 3 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
- 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

23.2 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。PSoC 4 は 1/2、1/3 および 1/4 デューティに対応します。
- **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 はシステムの最高ドライブ電圧 (PSoC 4 での V_{DD} に等しい) です。PSoC 4 は PWM ドライブ モードで 1/2 および 1/3 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の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 (実効値は 0 ではない) となります。

PSoC 4 は全てのドライブ 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。

23.2.1 ドライブ モード

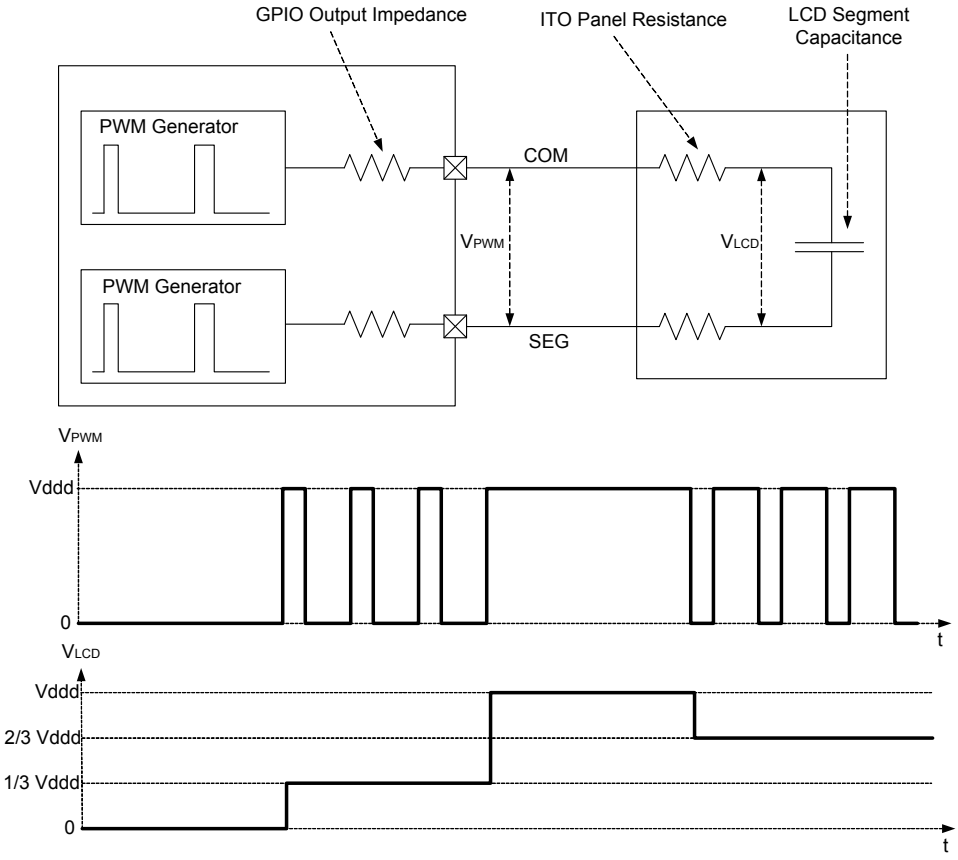
PSoC 4 は以下のドライブ モードに対応します。

- 1/2 バイアスでの PWM
- 1/3 バイアスでの PWM
- デジタル相関

23.2.1.1 PWM ドライブ

PWM ドライブ 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 23-1 はこれを説明する図です。

図 23-1. 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 波形を平滑化するのに、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 23-1 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) または IMO 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 (50 Hz 程度) で駆動します。

図 23-2 および図 23-3 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 23-4 および

図 23-5 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 23-2. PWM1/2 タイプ A 波形の例

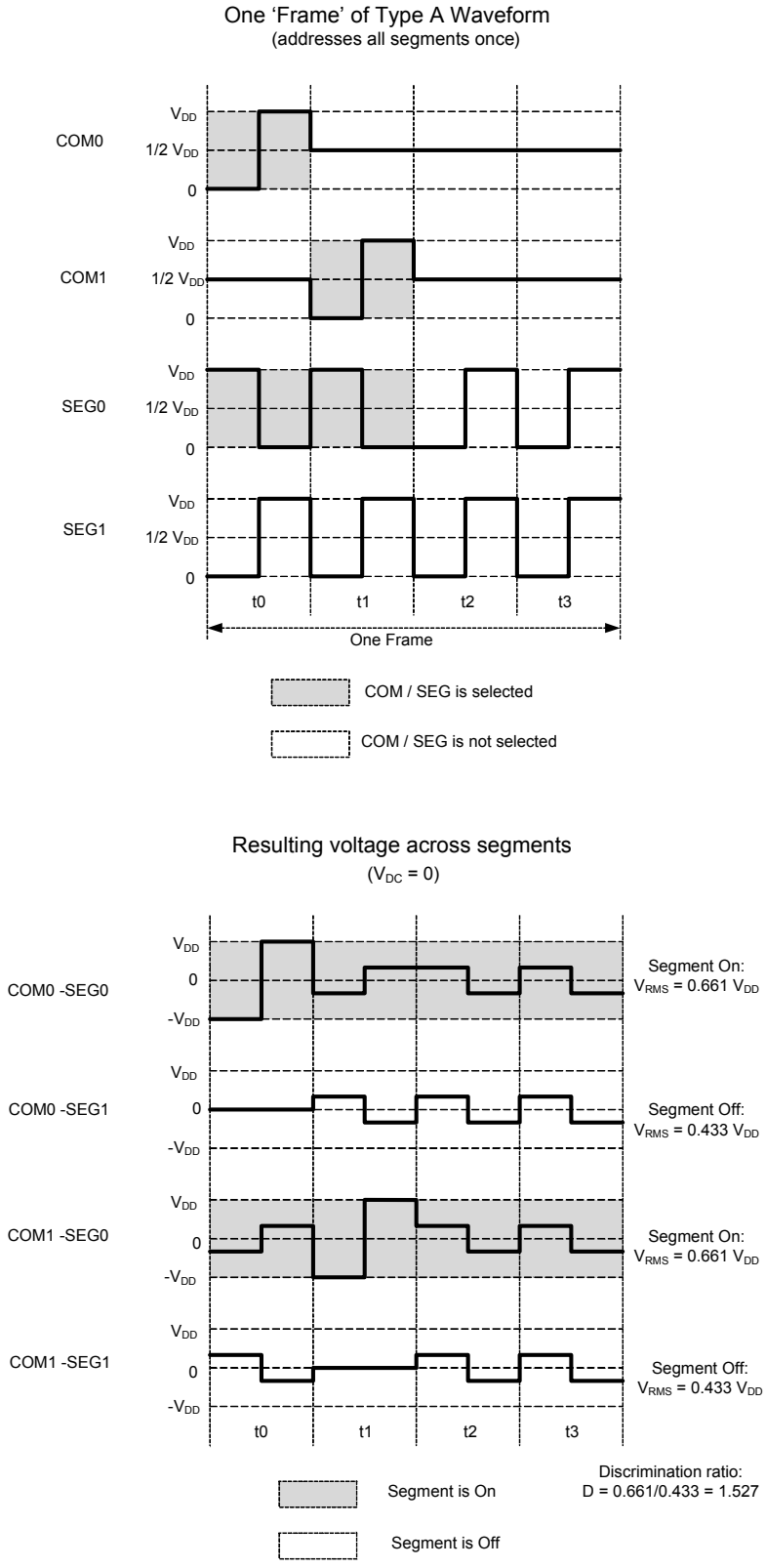


図 23-3. PWM1/2 タイプ B 波形の例

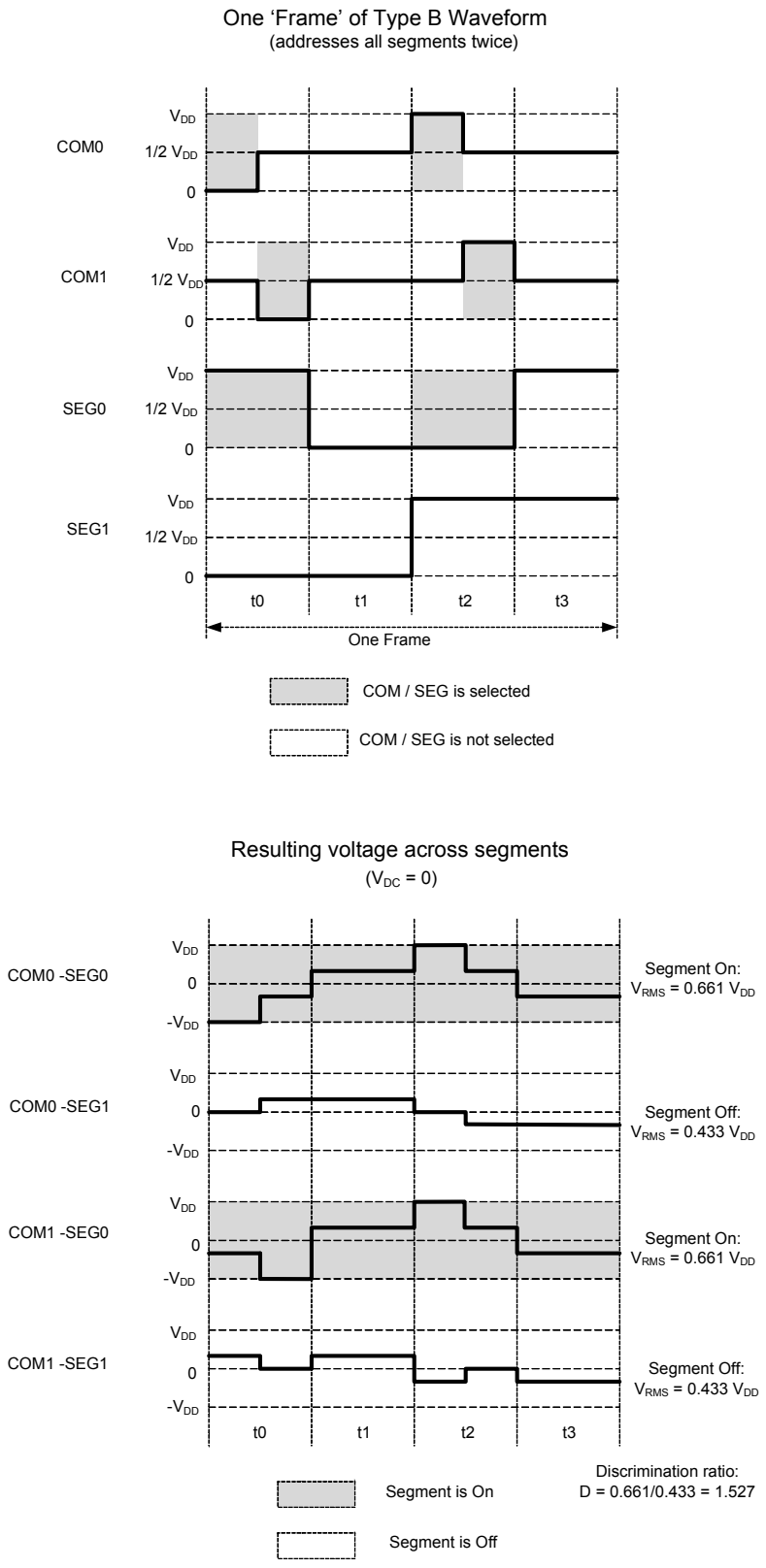


図 23-4. PWM1/3 タイプ A 波形の例

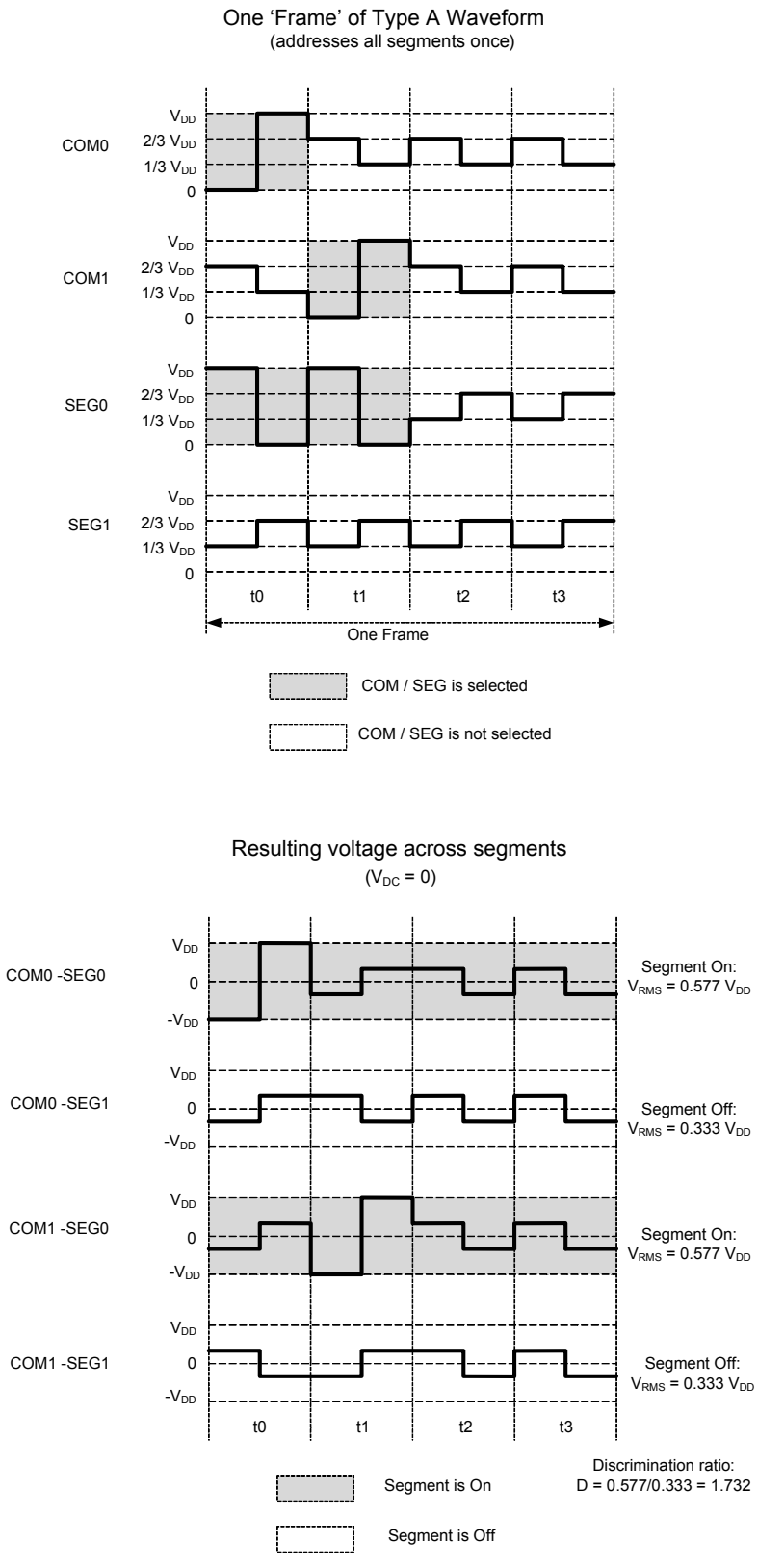
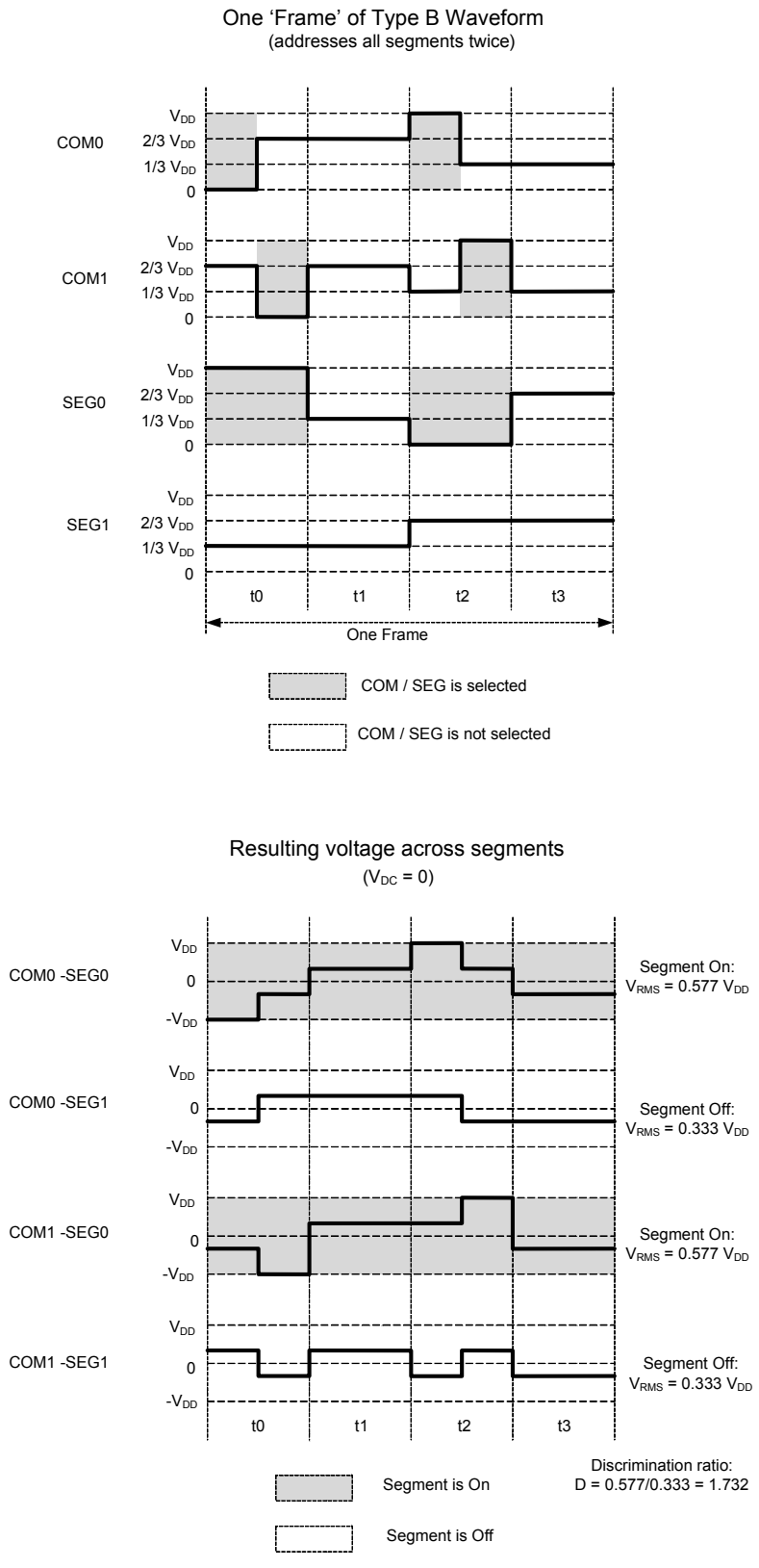


図 23-5. PWM1/3 タイプ B 波形の例



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。

$$V_{RMS(OFF)} = \sqrt{\frac{2(B-2)^2 + 2(M-1)}{2M}} \times \left(\frac{V_{DRV}}{B}\right)$$ **式 23-1**

$$V_{RMS(ON)} = \sqrt{\frac{2B^2 + 2(M-1)}{2M}} \times \left(\frac{V_{DRV}}{B}\right)$$ **式 23-2**

ここに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。

LCD を低速動作させるとき、PWM 信号は 32kHz 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、100 k から 1 MΩ 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 23-2 および図 23-3 に示すようにロジック レベルを使用することです。

23.2.1.2 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 23-8 および図 23-9 は動作原理を説明する波形例です。

図 23-6. デジタル相関 タイプ A 波形

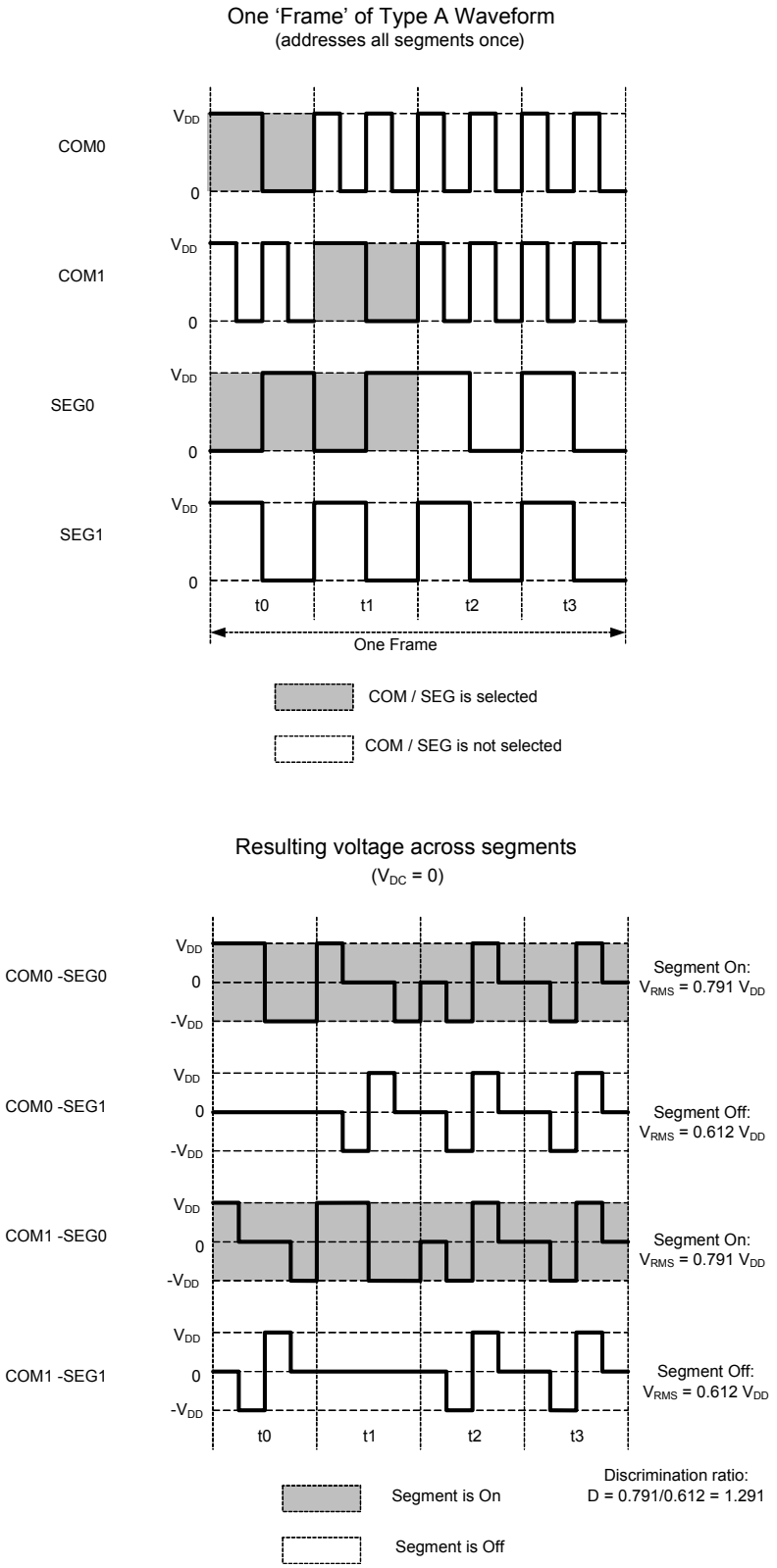
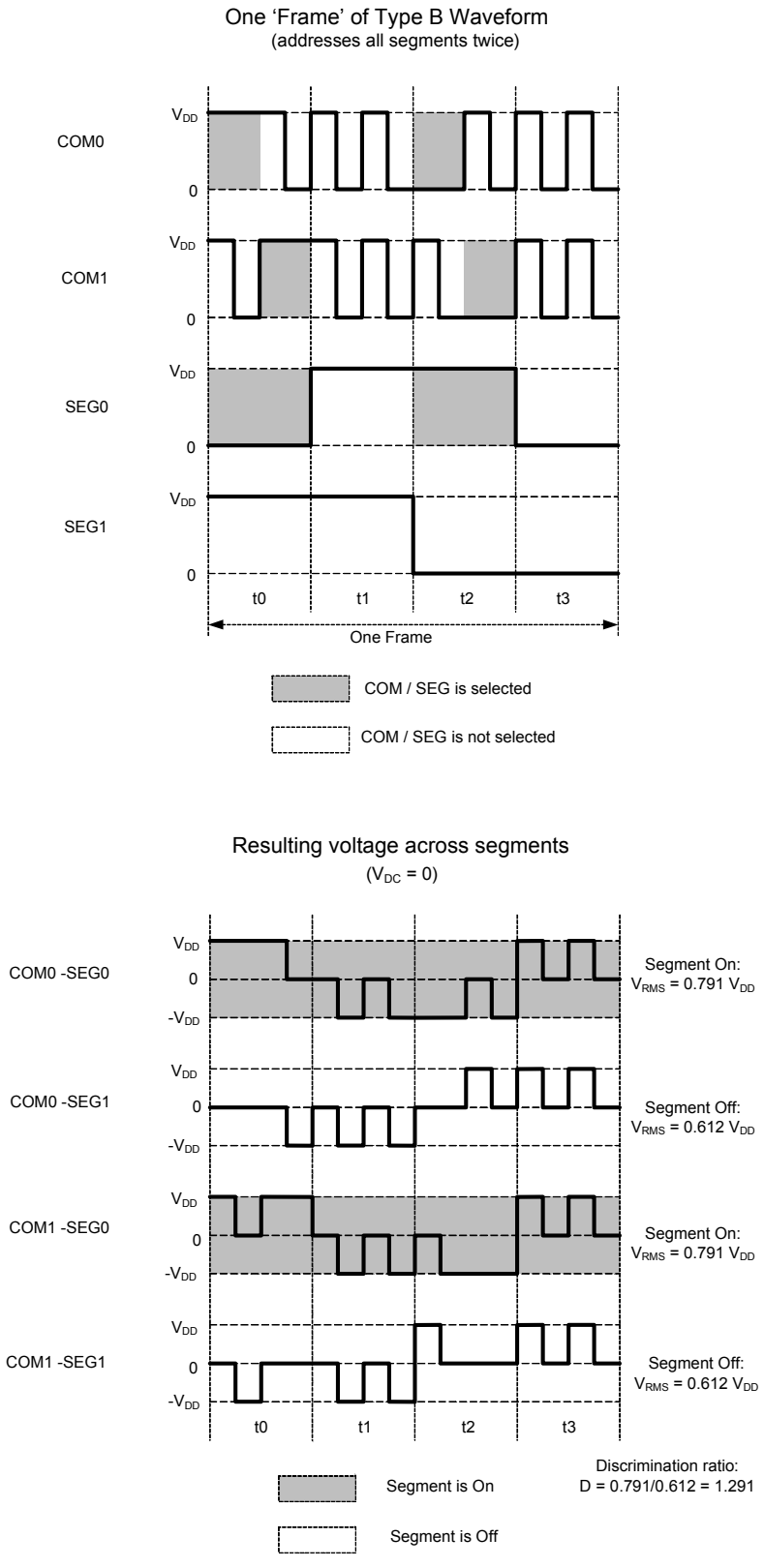


図 23-7. デジタル相関 タイプ B 波形



セグメントをオン、オフするために適用される RMS 電圧は下記のように計算されます。

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

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

ここに 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。

23.2.2 ドライブ モードの推奨使用方法

23.2.1.1 PWM ドライブおよび 23.2.1.2 デジタル相関に説明するように、PWM ドライブ 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。デジタル相関法のコントラストは PWM 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。

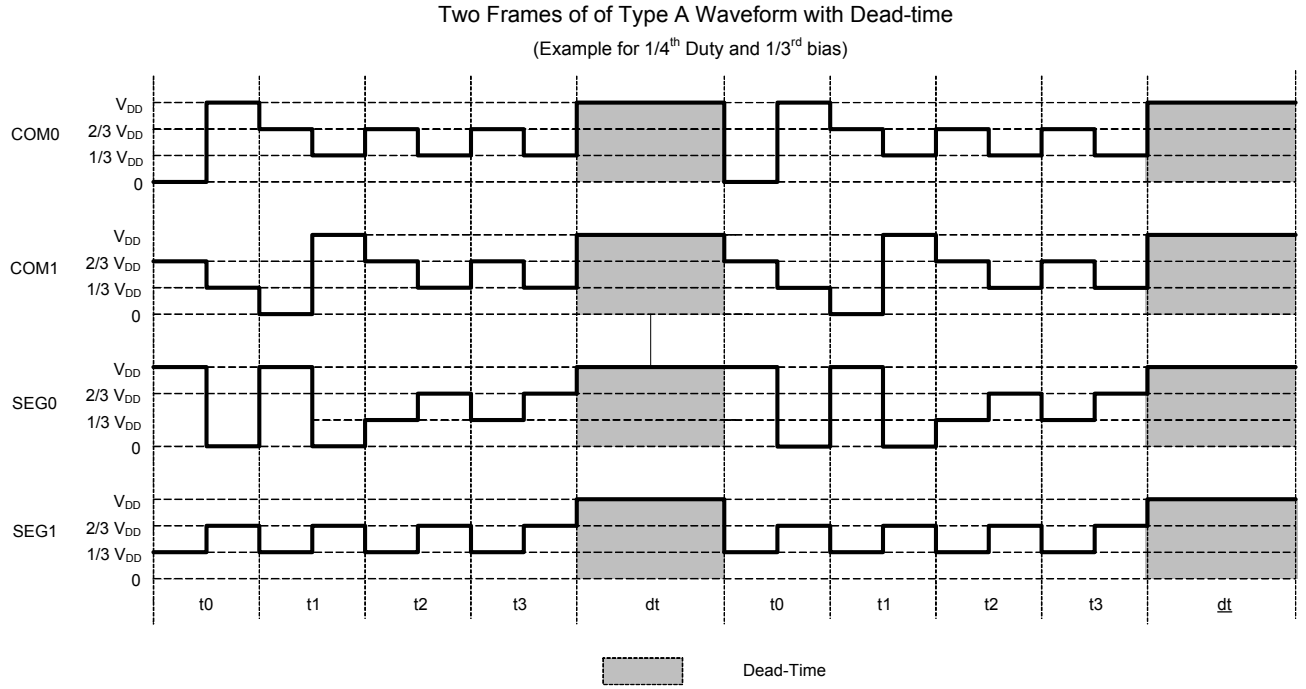
表 23-1. ドライブ モードの推奨使用方法

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
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 ドライブモードを切り替えなければならない
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

23.2.3 デジタル コントラスト制御

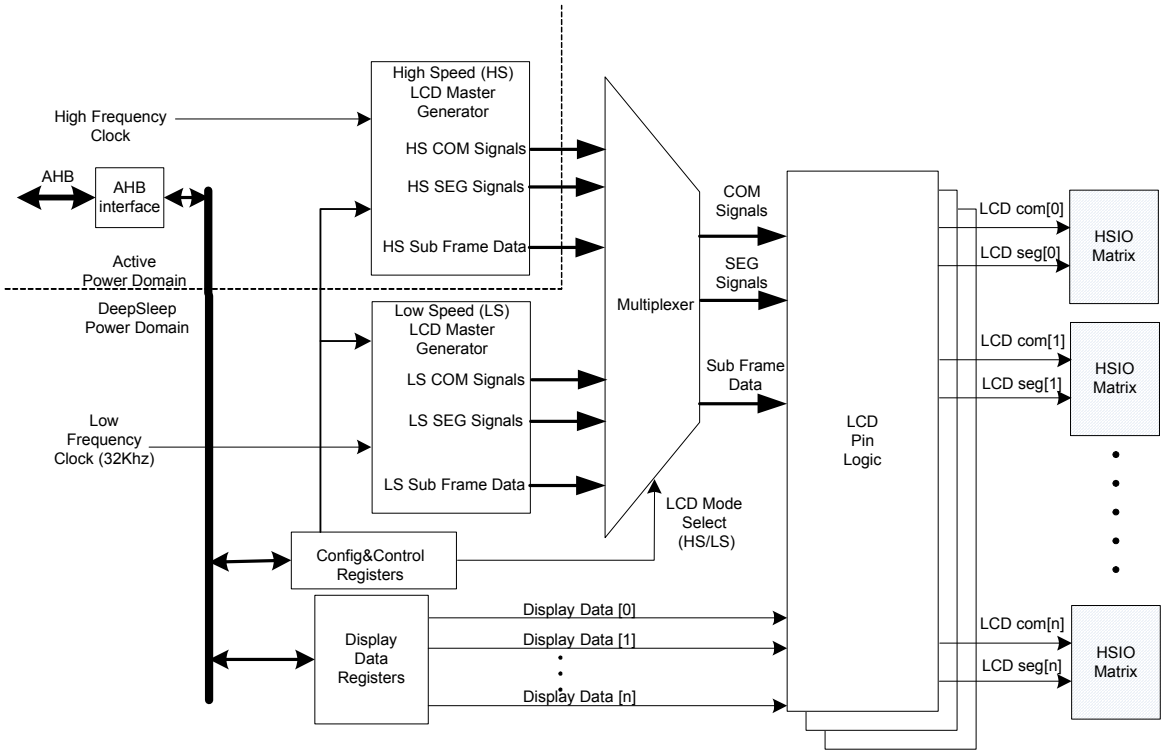
すべてのドライブ モードで、デジタル コントラスト制御はセグメントのコントラストを変更するのに使用されます。この方法は、セグメントの駆動時間を減少させることによってコントラストを減少させます。これは各フレーム後にデッドタイムを入れることによって実行されます。デッドタイム中にすべての COM および SEG 信号は論理 1 に駆動されます。デッドタイムは高分解能で制御されます。図 23-8 は 1/3 バイアスー 1/4 デューティ実装のデッドタイム コントラスト制御方法を説明します。

図 23-8. デッドタイム コントラスト制御



23.3 ブロック図

図 23-9. LCD ダイレクト ドライブ システムのブロック図



23.3.1 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 (32kHz) 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 272 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。63 ページの「高速 I/O マトリックス」を参照ください。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、スリープおよびディープ スリープ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。

23.3.2 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 (32kHZ) 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
- 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 フィールドでドライブ 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。これは 32kHz 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

23.3.3 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。

23.3.4 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_DATA0x 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注意してください。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_DATA0x 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 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 23-2 に示します。LCD_DATA0x 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 23-2. 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

23.4 レジスター覧

表 23-3. 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する
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

24. CapSense

PSoC®4 は CapSense® シグマ デルタ (CSD) 方式として知られる静電容量式のタッチ センシング方式を使用しています。CapSense のシグマ デルタ タッチ センシング方式は、業界最高の信号対雑音比 (SNR) を提供します。CSD はハードウェアとファームウェア技術を結合したものです。本章では CSD ハードウェアが PSoC 4 でどのように実装されるかについて説明します。

CSD の動作原理、CapSense 設計ツール、使いやすい PSoC Creator™ コンポーネント、チューナ GUI を使用する性能チューニング、プリント基板レイアウト設計の注意事項については PSoC 4 CapSense Design Guide を参照してください。

24.1 特長

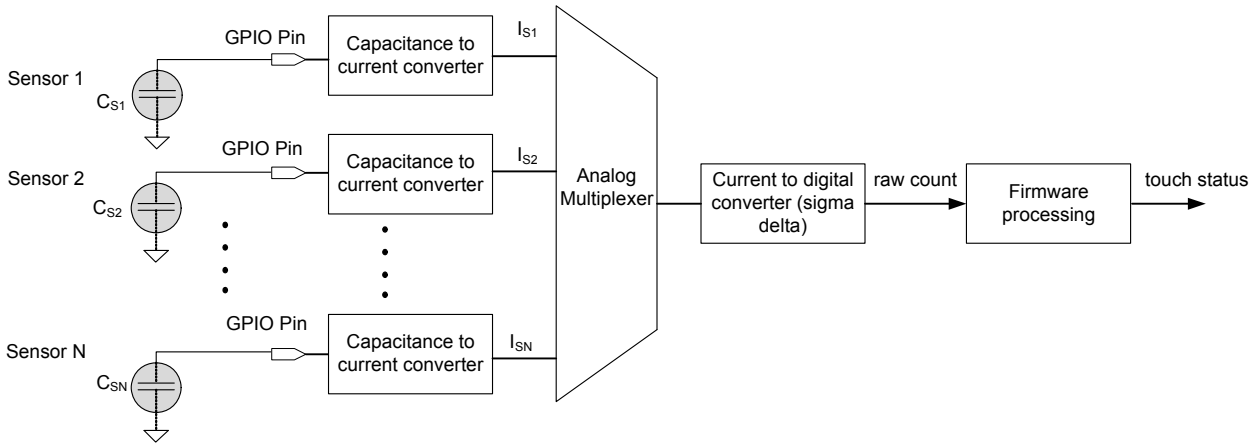
PSoC 4 CapSense は下記の特長があります。

- 堅牢なセンシング技術
- CSD 動作はクラス最高の SNR を提供
- 種々のオーバーレイ素材と厚さに対応した高性能検出
- SmartSense™ の自動チューニング技術
- 最大 55 個のセンサーをサポート
- 広範囲の近接感知
- シールド信号を用いた耐水性動作、すべての GPIO で対応
- 低消費電力
- スキャン速度と SNR を向上させる 2 つの IDAC 動作
- どの GPIO ピンでもセンシングまたはシールドに使用可能
- 電磁妨害 (EMI) を更に低減するための疑似乱数シーケンス (PRS) のクロック源
- シールド ラインに急速に電荷を移動させるために専用のチャージ タンク コンデンサを利用
- GPIO セル プリチャージが外部のタンクコンデンサを迅速に初期化
- 2 つの CSD ブロックに対応。その 1 つのブロックは CapSense アプリケーション、および汎用 IDAC とコンパレータ リソースの両方に使用可能。残りのブロックは汎用 IDAC とコンパレータ リソースのみに対応し、CapSense アプリケーションに使用不可。

24.2 ブロック図

図 24-1 に CSD システム ブロック図を示します。

図 24-1. CapSense モジュールのブロック 図



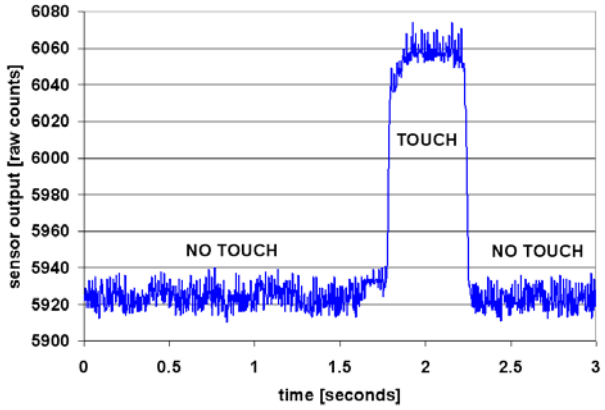
24.3 動作原理

CSD は各々の GPIO が、センサーの静電容量を等価な電流に変換するスイッチトキャパシタ回路を持っています。そしてアナログ マルチプレクサは、複数の電流の 1 つを選択して、それを電流 - デジタル コンバーターへ供給します。電流-デジタル コンバーターはシグマ デルタ ADC に似ています。

電流-デジタル コンバーターの出力カウント (Raw カウントとして知られる) は、センサーの静電容量に比例したデジタル値です。

図 24-2 は時間経過に伴う Raw カウントの変化を示します。指がセンサーに触れると、センサー静電容量が増加し、raw カウントがそれに比例して増加します。指定した閾値と Raw カウントの変化を比較することで、ファームウェアのロジックはセンサーがアクティブ (指がセンサーに触れる) であるかどうかを判定できます。

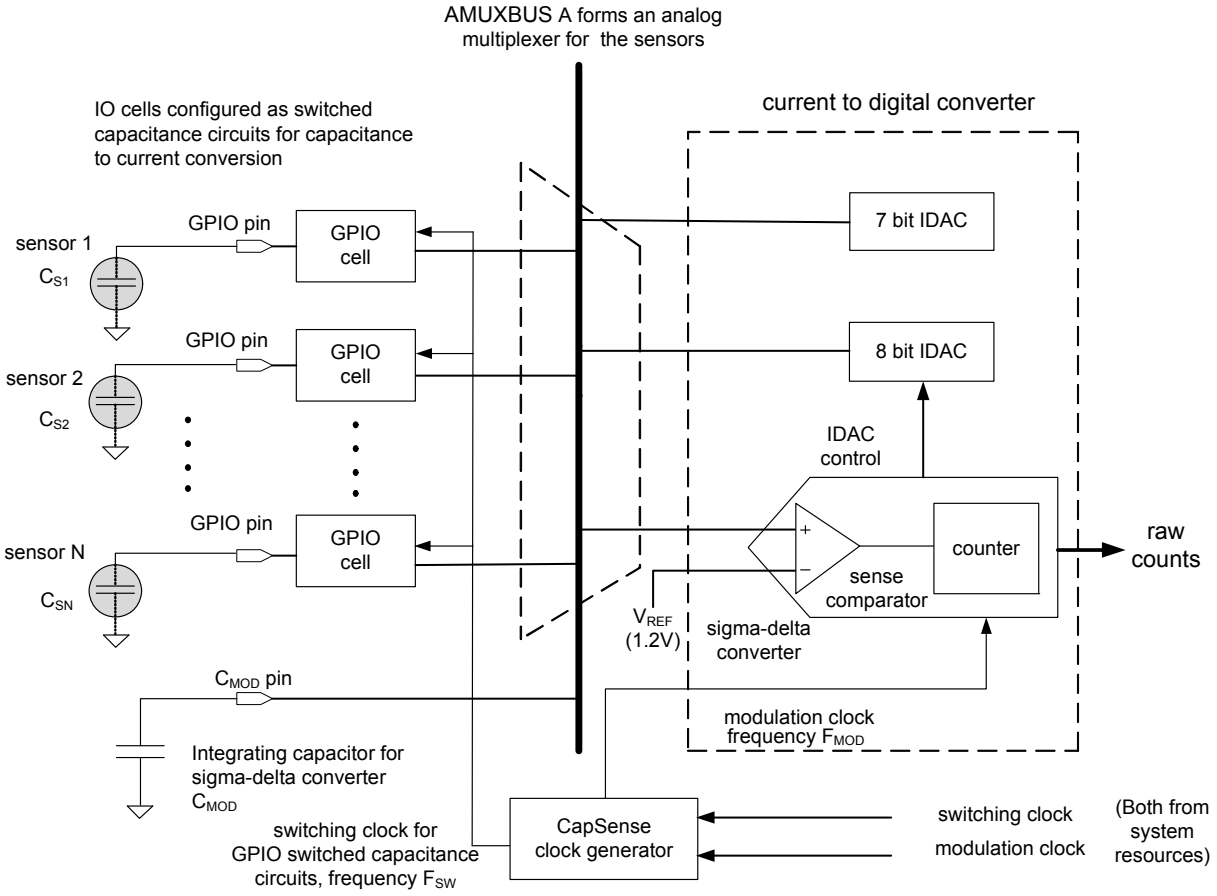
図 24-2. Raw カウントと時間



24.4 CapSense CSD のセンシング

図 24-3 に PSoC 4 CapSense ハードウェアのブロック図を示します。

図 24-3. PSoC 4 CapSense CSD センシング



24.4.1 GPIO セルの静電容量／電流コンバータ

CapSense CSD システムでは、GPIO セルがセンサーの静電容量を等価電流に変換するスイッチト キャパシタ回路として構成されています。図 24-4 は PSoC 4 GPIO セル構造の概略図を示します。

PSoC 4 は 2 つのアナログ マルチプレクサ バスを持っています。それらは CSD センシング用の AMUXBUS A、CSD シールド用の AMUXBUS B です。GPIO スイッチト キャパシタ回路は 2 つの構成があります。AMUXBUS A への電流吐き出しまたは AMUXBUS A からの電流吸いこみの構成です。図 24-5 は AMUXBUS A への電流吐き出し型スイッチト キャパシタ回路の構成を示します。

図 24-4. PSoC 4 の GPIO セル

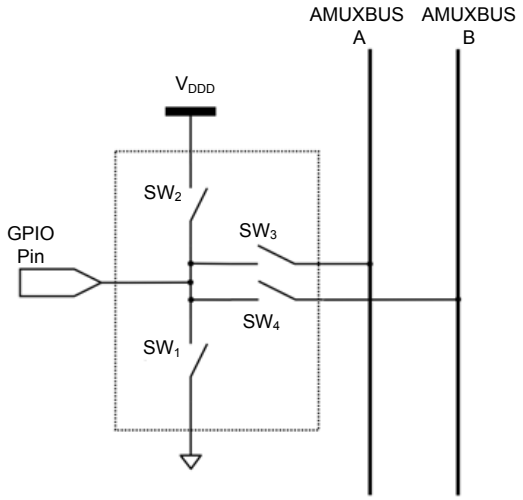
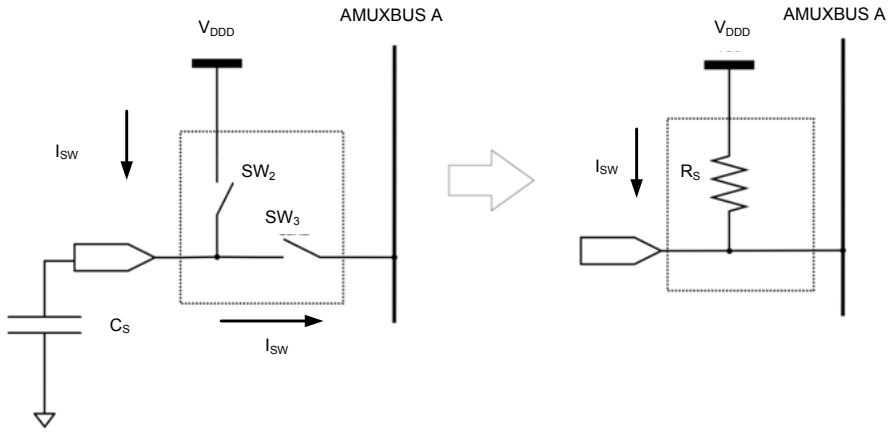


図 24-5. AMUXBUS A への電流吐き出し



周波数 F_{SW} の、位相が一致しない 2 つの重複しないクロック (図 24-3 を参照) がスイッチ SW_2 と SW_3 を制御します。図 24-5 に示すように SW_2 と SW_3 の連続切り替えにより、等価抵抗 R_S を形成します。等価抵抗 R_S の値は以下の通りです。

$$R_S = \frac{1}{C_S F_{SW}}$$

式 24-1

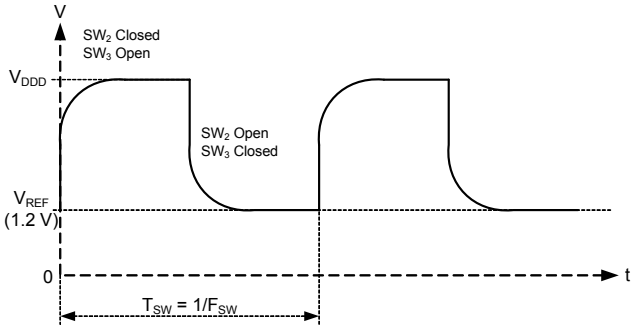
ここでは、

C_S= センサーの静電容量

F_{SW} = スイッチング クロックの周波数

シグマデルタコンバータは V_{REF} 定数で AMUXBUS A の電圧を維持します (この処理は 287 ページのシグマデルタ コンバータで説明されています)。図 24-6 はセンサー容量による電圧波形を示します。

図 24-6. センサー容量による電圧



式 23-3 により AMUXBUS A に供給される平均電流の値が得られます。

$$I_S = C_S F_{SW}(V_{DDD} - V_{REF})$$

式 24-2

図 24-7 は AMUXBUS A から電流吸い込み型スイッチ トキャパシタ回路の構成を示します。図 24-8 は C_S による電圧波形を示しています。

図 24-7. AMUXBUS A からの電流吸いこみ

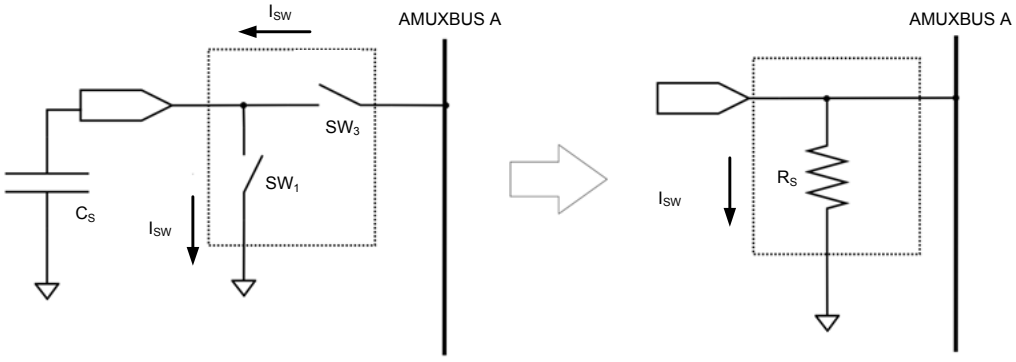
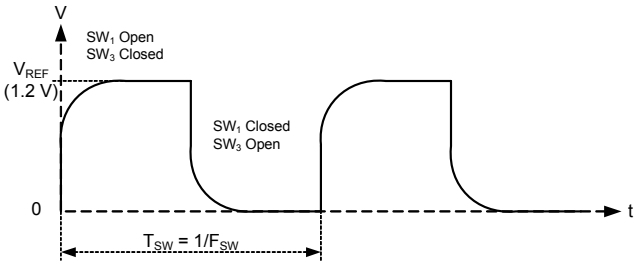


図 24-8. センサー容量による電圧



式 23-4 により、AMUXBUS A から供給される平均電流の値が得られます。

$$I_S = C_S F_{SW} V_{REF}$$

式 24-3

シグマデルタ コンバータは一度に 1 個のセンサーをスキャンします。図 24-3 に示すように AMUXBUS A は GPIO セルのいずれかを選択し、それをシグマデルタコンバータの入力に接続するために使用されます。AMUXBUS A と GPIO セル スイッチ (図 24-4 での SW_3 を参照) はこのアナログ マルチプレクサを形成します。AMUXBUS A は CSD に対応する PSoC 4 ピンに接続可能です。CSD に対応可能なピンの詳細については、デバイスのデータシート を参照してください。

検出、シールド、C_{MOD} の接続用に GPIO セルを設定する方法については59ページのI/Oシステムを参照してください。

24.4.2 CapSense クロック ジェネレータ

図 24-3 に示すように、このブロックはシステム リソースからのプログラマブルなクロック分周器と共にスイッチング クロック F_{SW} と変調クロック F_{MOD} を生成します。詳細については67 ページのクロック供給システムを参照してください。

スイッチング クロックは GPIO セルのスイッチト キャパシタ回路用に要求されます。シグマデルタコンバータはタイミング用に変調クロックを使用します。

システム リソースからの任意の 2 つのプログラマブルなクロック分周器は HFCLCK を分周して必要な周波数を生成するために使用できます。詳細は67 ページのクロック供給システムをご覧ください。通常は 2 個のカスケード接続されたクロック分周器が使用されます。1 個のクロック分周器は変調クロックを生成し、残りのクロック分周器はスイッチング クロックを生成します。

しかしながら最終のスイッチング クロック周波数は CapSense クロック ジェネレーターに左右されます。次の出力オプションがあります :

- 直接 : プログラマブルなクロック分周器の出力を直接使用します。このオプションを選択するには CSD_CONFIG レジスタ内の BYPASS_SEL ビットを「1」にセットします。
- 2 分周 : クロックを 2 分周します。このオプションを選択するには CSD_CONFIG レジスタ内の PRS_SELECT と BYPASS_SEL ビットをクリアします。
- 疑似乱数シーケンス (PRS): スイッチング周波数をより広い範囲に拡散することで CapSense システム内の EMI を削減します。このオプションを選択するには、CSD_CONFIG レジスタでは PRS_SELECT ビットをセットして、BYPASS_SEL ビットをクリアします。同じレジスタ内の PRS_12_8 ビットを使用することで 8 ビットまたは 12 ビットの疑似乱数シーケンスを選択できます。このビットをセットすることで 12 ビットのシーケンスを選択し、それをクリアすることで 8 ビットの PRS を選択します。

PRS が選択されれば最大のスイッチング周波数は次のとおりです。

$$F_{SW(maximum)} = \frac{F_{in}}{2}$$

式 24-4

ここで F_{in} はスイッチング分周器の周波数出力です。最小周波数は次のとおりです。

$$F_{SW(minimum)} = \frac{F_{in}}{PRS\ length-1}$$

式 24-5

ここで「PRS length」は 12 または 8 ビットです。平均スイッチング周波数は次のとおりです。

$$F_{SW(average)} = \frac{F_{in}}{4}$$

式 24-6

CSD_CONFIG 内の PRS_CLEAR ビットは PRS をクリアするために使用できます。セットされれば、このビットは疑似乱数ジェネレーターを初期状態にさせます。

24.4.3 シグマデルタ コンバータ

シグマデルタ コンバータは入力電流を対応するデジタル カウントに変換します。図 24-3 に示すようにこのコンバータは 1 個のコンパレータ、1 つの電圧リファレンス V_{REF}、1 個のカウンターと 2 個の電流デジタル - アナログ変換器 (IDAC) から構成されます。

シグマデルタ変調器はオン／オフ マナーで 8 ビット IDAC の電流を制御します。この IDAC は変調 IDAC として知られています。この 7 ビットの IDAC は補正 IDAC として知られて、常にオンまたは常にオフです。

シグマデルタ コンバータはシングル IDAC モードまたはデュアル IDAC モードのいずれかで作動できます。シングル IDAC モードでは補正 IDAC は常にオフです。デュアル IDAC モードでは、補正 IDAC は常にオンです。

図 24-1 に示すようにシグマデルタ コンバータは外部に積分コンデンサ C_{MOD} も必要とします。C_{MOD} の推奨値は 2.2nF です。PSoC 4 は 1 つの専用 C_{MOD} ピンを有します。詳細については デバイスのデータシートのピン配置を参照してください。

シグマデルタ変調器は C_{MOD} の両端電圧を V_{REF} に維持します。それは次のいずれかのモードで動作します。

- IDAC ソース モード : スイッチト キャパシタ回路が AMUXBUS A から電流を吸い込むと、IDAC はその電圧のバランスをとるために AMUXBUS A に電流を供給します。
- IDAC シンク モード : このモードでは IDAC は C_{MOD} から電流を吸い込み、スイッチト キャパシタ回路が C_{MOD} に電流を供給します。

両方の場合において C_{MOD} 電圧を V_{REF} に維持するために、変調 IDAC 電流は C_{MOD} 両端における少しの電圧変動に応じてオンとオフに切り替えます。

シグマデルタ コンバータは 8 ビット～16 ビットの分解能で動作することができます。シングル IDAC モードでは Raw カウントはセンサー静電容量に比例します。「N」がシグマデルタ コンバータの分解能であり、I_{MOD} が変調 IDAC 電流の値になると、IDAC ソース モードでの Raw カウントの近似値は式 16-7 から得られます。

$$Rawcount = 2^N \frac{V_{REF} F_{SW}}{I_{MOD}} C_S$$

式 24-7

同様に IDAC シンク モードでの Raw カウントの近似値は次の通りです。

$$Rawcount = 2^N \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S$$

式 24-8

両方の場合において raw カウントはセンサー静電容量 C_S に比例します。この Raw カウントはタッチを検出するためにファームウェアによって処理されます。CapSense 性能を向上するために両方の IDAC をデュアル IDAC モードで使用することができます。

このデュアル IDAC モードでは補正 IDAC は常にオンです。I_{COMP} が補正 IDAC 電流になると IDAC ソース モードでの Raw カウント用の式は以下の通りです。

$$Rawcount = 2^N \frac{V_{REF} F_{SW}}{I_{MOD}} C_S - 2^N \frac{I_{COMP}}{I_{MOD}}$$

式 24-9

IDAC シンク モードでの Raw カウントは式 16-10 で計算されます。

$$Rawcount = 2^N \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S - 2^N \frac{I_{COMP}}{I_{MOD}}$$

式 24-10

Rawカウントの値は常に正であることに注意してください。

I_{COMP}、I_{MOD}、F_{SW} などのハードウェア パラメーターは信頼性の高いタッチ検出のために最適値に調整する必要があります。調整プロセスの詳細な説明については「PSoC 4 CapSense Design Guide」を参照してください。

CSD_CONFIG、CSD_COUNTER、CSD_IDAC レジスタはシグマデルタコンバータの動作を制御します。CSD_CONFIG レジスタ内の重要なビットは以下のとおりです。

- CSD_CONFIG 内の ENABLE: CSD ブロックのマスター イネーブル ビットです。任意の CSD 動作が機能するためにこのビットを「1」にセットする必要があります。
- CSD_CONFIG 内の POLARITY: IDAC シンク モードと IDAC ソース モードから選択します。0: IDAC ソース モード、1: IDAC シンク モード .
- CSD_CONFIG 内の SENSE_COMP_BW : 検出コンパレータの帯域幅を選択します。このビットをセットすると高帯域幅が得られ、このビットをクリアすると低帯域幅が得られます。高帯域幅は CSD 動作に推奨されます。
- CSD_CONFIG 内の SENSE_COMP_EN: 検出コンパレータ回路をオンにするビットです。0: 検出コンパレータは電源オフです。1: 検出コンパレータは電源オンです。
- SENSE_EN: シグマ デルタ変調器出力を有効にするビットです。このビットも IDAC をオンにします。

IDAC は CSD 動作のために適切に設定する必要があります。詳細については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」での CSD_IDAC レジスタを参照してください。

CSD_COUNTER レジスタは現在選択されているセンサーのサンプリングを開始して結果を読み出すために使用されます。コンパレータが (変調クロック周波数で) サンプリングされ、サンプル値が 1 になるたびにこのレジスタの 16 ビット COUNTER フィールドはインクリメントします。通常新しい検出動作が開始すると、ファームウェアは「0」をこのフィールドに書き込みます。CSD_COUNTER レジスタ内の 16 ビットの PERIOD フィールドは静電容量 - デジタル変換を開始するために使用されます。このレジスタに非ゼロ値を書き込むと検出動作は開始します。ファームウェアによりこのフィールドに書き込まれる値は COUNTER フィールドがコンパレータ出力をサンプリングする期間を決定します。

CSD 動作を開始する前にクロック、GPIO、IDAC、シグマデルタ変調器を適切に設定する必要があります。PERIOD フィールドは各変調クロック サイクル後にデクリメントします。0 に達すると COUNTER フィールドはインクリメントを停止します。この時点のこのフィールドの値はセンサー静電容量の値に対応する Raw カウントです。

24.5 CapSense CSD シールド

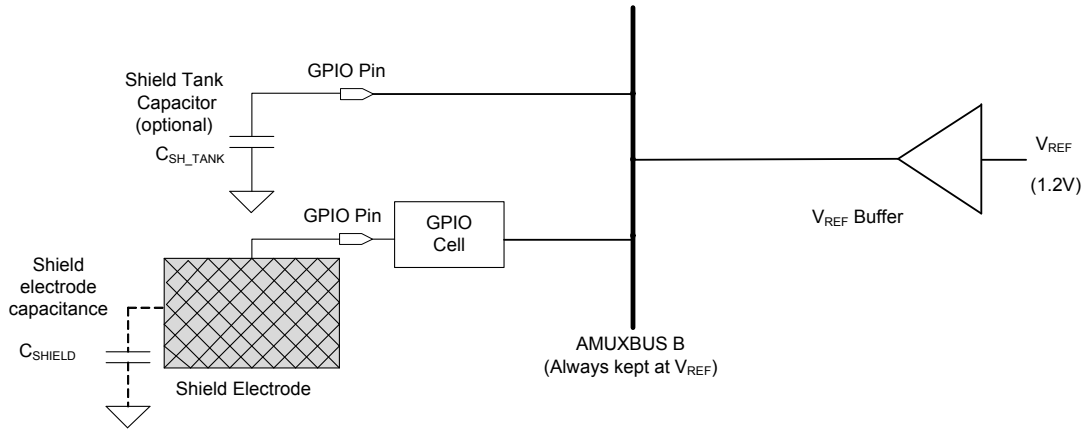
PSoC 4 CapSense は耐水性と近接センシング用のシールド電極をサポートします。耐水性についてはシールド電極は常にセンサーと同じ電位に保たれます。PSoC 4 CapSense はセンサーとシールド電極間の電位差を無効にするためにセンサー スイッチング信号のレプリカ信号でシールド電極を駆動するシールド回路を有します (285 ページの GPIO セルの静電容量／電流コンバータを参照)。シールドの基本については「PSoC 4 CapSense Design Guide」を参照してください。

センシング回路ではシグマデルタ コンバータは AMUXBUS A を V_{REF} に保ちます (287 ページのシグマデルタ コンバータを参照)。GPIO セルは AMUXBUS A と電源レール (設定によって、V_{DD} またはグランド) 間でセンサーを切り替えることでセンサー波形を生成します。シールド回路は同じような方法で動作します。AMUXBUS B は常に V_{REF} に保たれます。GPIO セルは AMUXBUS B と電源レール (センサーの設定と同じように、V_{DDD} またはグランド) 間でシールドを切り替えます。このプロセスはシールド電極上にセンサー スイッチング波形のレプリカを生成します。

AMUXBUS B を V_{REF} に保持する方法によって 2 つの異なる構成が可能です。

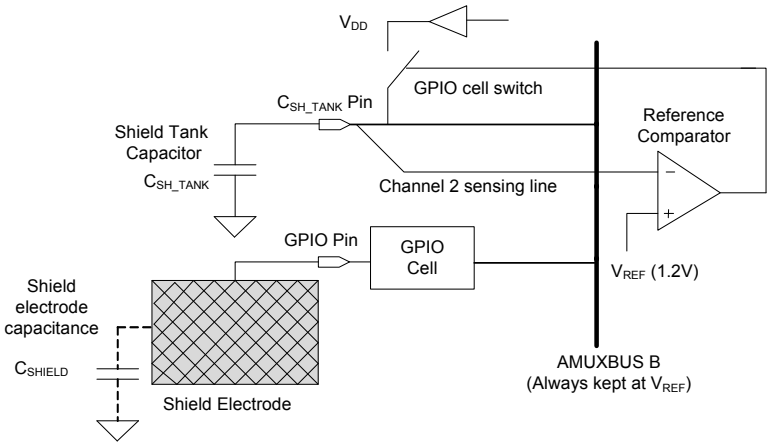
- V_{REF} バッファによるシールド駆動: この構成では図24-9に示すように電圧バッファは AMUXBUS B を V_{REF} に駆動するために使用されます。外部の C_{SH_TANK} コンデンサはスイッチング ノイズを減らすために推奨されます。CSD_CONFIG レジスタ内の REBUF_OUTSEL ビットをセットするとバッファ出力がAMUXBUS Bに接続されます。このレジスタ内の REFBUF_DRV ビット フィールドはバッファの駆動能力を設定するために使用できます。このフィールドに「0」を書き込むとバッファが無効になります。「1」「2」と「3」を書き込むとそれぞれ小電流、中電流、大電流の駆動モードが選択されます。

図 24-9. V_{REF} バッファを使用するシールド駆動



- GPIOセル プリチャージを使用するシールド駆動: この構成には図24-10に示すように外部のC_{SH_TANK}コンデンサが必要です。特定の GPIO セルとリファレンス コンパレータが C_{SH_TANK} コンデンサを充電するために使用されるので、AMUXBUS B を V_{REF} に駆動できます。電圧を V_{REF} に維持するためにリファレンス コンパレータは常に CSH_TANK コンデンサの電圧をモニターして、GPIO セル スイッチを制御します。図 24-10 に示すように、リファレンス コンパレータはチャネル 2 センシング ラインと呼ばれる専用の検出ラインを使用して C_{SH_TANK} コンデンサに接続します。

図 24-10. GPIO プリチャージを使用するシールド駆動



この GPIO セル プリチャージ機能は特定の C_{SH_TANK} ピンのみで使用できます。詳細については デバイスのデータシートでのデバイス ピン配置を参照してください。

CSD_CONFIG レジスタ内の COMP_MODE ビットでリファレンス バッファ プリチャージまたは GPIO プリチャージを選択します。0: リファレンス バッファ プリチャージ ; 1: GPIO プリチャージ。

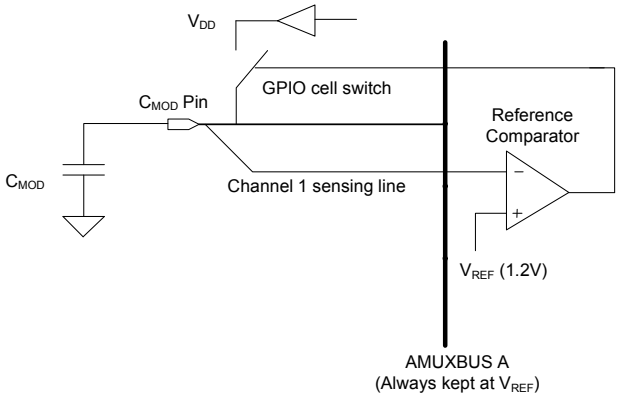
24.5.1 C_{MOD} プリチャージ

CapSense ハードウェアに初めて電源が投入される時、C_{MOD} の両端電圧は「0」から上昇します。その後シグマデルタ コンバータが C_{MOD} を V_{REF} までゆっくり充電します。充電電流は、IDAC がソース モードにある場合 IDAC によって供給され、IDAC がシンク モードにある場合センサーのスイッチト キャパシタ回路によって供給されます。C_{MOD} が比較的大きな容量のコンデンサなので充電は徐々に進みます。

C_{MOD} のプリチャージは C_{MOD} の両端電圧を V_{REF} に速く初期化するプロセスです。プリチャージはシグマデルタ コンバータが動作を開始するために必要な時間を減らします。C_{MOD} のプリチャージは 2 つのオプションがあります。

- V_{REF} バッファを使用するプリチャージ: シールドが有効になった時 V_{REF} バッファ出力は常に AMUXBUS B に接続されます (図 24-9)。V_{REF} バッファを使用してプリチャージするために、最初は C_{MOD} が AMUXBUS B に接続されます。プリチャージ プロセスの後に C_{MOD} は通常のシグマデルタ動作のために AMUXBUS A に接続されます。シールドが無効にされる時 V_{REF} バッファの出力はプリチャージ用に AMUXBUS A に常に接続されていて、その後切断されます。
- GPIO セルを使用するプリチャージ : この構成では特定の GPIO セルとリファレンス コンパレータが C_{MOD} コンデンサを V_{REF} に充電します。この GPIO セル プリチャージ機能は特定の C_{MOD} ピンのみで使用できます。詳細については デバイスのデータシートのピン配置を参照してください。この目的に使用されるコンパレータは CSH_TANK プリチャージに使用されるリファレンス コンパレータと同じです。CSD_CONFIG レジスタ内の COMP_PIN ビットはどのコンデンサがリファレンス コンパレータに接続されるかを選択するために使用されます。このビットが「0」の時、図 24-11 に示す「チャネル 1」の検出ラインが C_{MOD} をリファレンス コンパレータに接続するために使用されます。このビットが「1」の時、図 24-10 に示すチャネル 2 の検出ラインが CSH_TANK をリファレンス コンパレータに接続するために使用されます。GPIO セル プリチャージが機能するには GPIO セルを適切に設定する必要がある点に注意してください。

図 24-11. GPIO セル プリチャージ



GPIO セルを使用するプリチャージは V_{REF} バッファを使用するプリチャージより高速です。そのため GPIO プリチャージは推奨されるプリチャージ構成です。しかし CapSense の高速初期化が必要でなければ、V_{REF} バッファのプリチャージを使用できます。

またチャネル 1 検出ラインは C_{MOD} をシグマ デルタ変調器内の検出コンパレータに接続するためにも使用できます。CSD_CONFIG レジスタ内の SENSE_INSEL ビットを「1」にセットするとこのオプションが可能になります。このビットをクリアすると AMUXBUS A を使用して C_{MOD} を検出コンパレータに接続します。

24.6 汎用リソース : IDAC とコンパレータ

CapSense ブロックがタッチ センシングに使用されない場合、検出コンパレータと 2 個の IDAC は汎用アナログ ブロックとして使用できます。

AMUXBUS A を使用して、CSD に対応する GPIO を検出コンパレータの非反転入力に接続することが可能です。反転入力は 1.2V の V_{REF} に接続されます (図 24-3 を参照)。AMUXBUS A はコンパレータ入力でのアナログ マルチプレクサとして使用することもできます。287 ページのシグマデルタ コンバータで説明したとおり、CSD_CONFIG レジスタ内の SENSE_COMP_EN、SENSE_COMP_BW、ENABLE ビットは検出コンパレータを制御するために使用できます。

290 ページの CMOD プリチャージで説明したように AMUXBUS が他の用途に必要な場合、CSD_CONFIG レジスタ内の SENSE_INSEL ビットは検出コンパレータの非反転入力を固定の C_{MOD} ピンに接続するために使用可能です。コンパレータの出力を複数の GPIO に接続できます。詳細については 59 ページの I/O システムをご参照ください。

8 ビットの IDAC は 0 ～ 306µA (1.2µA/ ビット) または 0 ～ 612µA (2.4µA/ ビット) の範囲で動作します。7 ビットの IDAC は 0 ～ 152.4µA (1.2µA/ ビット) と 0 ～ 304.8µA (2.4µA/ ビット) の範囲に対応します。

8 ビットの IDAC と 7 ビットの IDAC は両方とも AMUXBUS A と AMUXBUS B を使用して GPIO に接続できます。両方の IDAC を 1 つの AMUXBUS に接続することもできます。IDAC は CSD 専用モード、汎用 (GP) モード、CSD と GP モードという 3 つの異なるモードで動作可能です。表 24-1 はこれらのモードのそれぞれで IDAC1 と IDAC2 を AMUXBUS A と AMUXBUS B に接続する方法について説明します。

表 24-1. IDAC モード

モード	AMUXBUS A	AMUXBUS B
CSD 専用	1.2V で両方の IDAC が電流の吸い込み、吐き出し動作	IDAC が未接続
汎用モード	8 ビット IDAC が電流の吸い込み、吐き出し動作	7 ビット IDAC が電流の吸い込み、吐き出し動作
CSD と GP モード	1.2V で 8 ビット IDAC が電流の吸い込み、吐き出し動作	7 ビット IDAC が電流の吸い込み、吐き出し動作

詳細については「 PSoC 4100M/4200M Family: PSoC 4 Registers TRM」での CSD_IDAC レジスタを参照してください。287 ページのシグマデルタ コンバータで述べたとおり、CSD_CONFIG レジスタは IDAC を有効にし、極性を設定するために使用できます。GPIO を AMUXBUS A と AMUXBUS B に接続する方法の詳細については 59 ページの I/O システムを参照してください。 予備の CSD ブロックは汎用モードのみに対応します。このブロックは CapSense アプリケーションに使用できません。

24.7 レジスター覧

表 24-2. CapSense レジスター覧

レジスタ名	説明
CSD_CONFIG	このレジスタは CSD ブロックとそのリソースを設定・制御するために使用
CSD_IDAC	このレジスタは IDAC 電流の設定を制御するために使用
CSD_COUNTER	このレジスタは選択した静電容量センサーのサンプリングを開始し、変換の結果を読み出すために使用
CSD_STATUS	このレジスタで CSD ブロック内の重要な信号を観測
CSD_INTR	これは CSD 割り込み要求レジスタ

25. 温度センサー

PSoC® 4 は内部ダイ温度を測定するために使用する温度センサーを内蔵します。センサーはダイオード接続されるトランジスタで構成されます。

25.1 特長

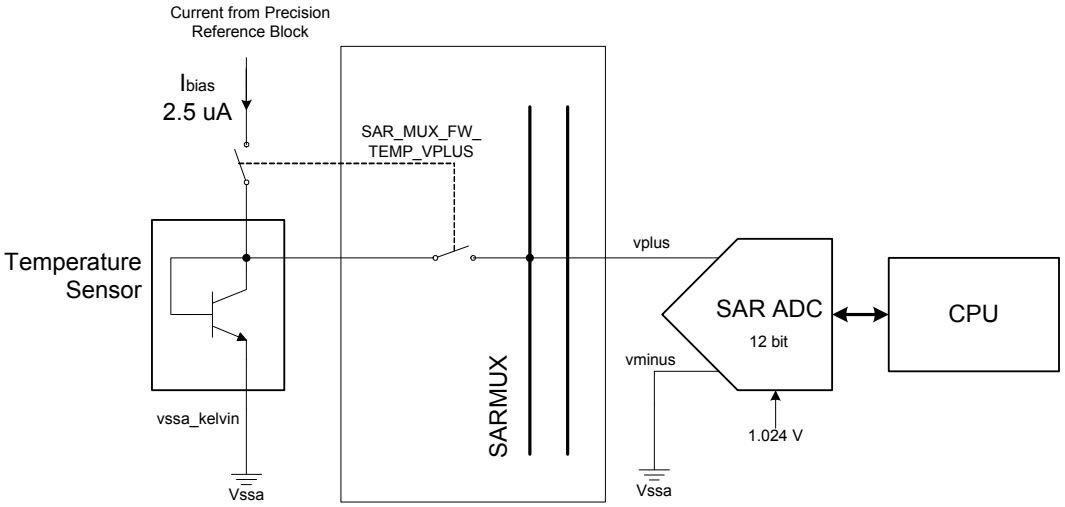
温度センサーは以下の特長があります。

- –40°C ～ +85°C の温度範囲で ±5 ℃の精度
- 1.024V のリファレンス電圧で 12 ビットの SAR ADC を使用する時に、温度センサーの分解能は 0.5°C/LSB (増幅なし)
- 10µs のセトリング時間

25.2 動作原理

温度センサーは、1 個のバイポーラ接合トランジスタ (BJT) をダイオードとした構成です。そのベースーエミッタ電圧 (V_{BE}) はコレクタ電流を一定としコレクターベース間電圧を 0 とするとき、温度に強く依存します。この性質は図 25-1 に示すように、SAR ADC を使用してトランジスタの V_{BE} を測定することによりダイ温度を計算するために使用されます。

図 25-1. 温度の検出原理



センサーからのアナログ出力 (V_{BE}) は SAR ADC を使用して測定します。摂氏単位のダイ温度は以下の式にしたがって、ADC の結果から計算されます。

$$Temp = (A \times SAR_{out} + 2^{10} xB) + T_{adjust}$$

式 25-1

- 温度は摂氏単位でスロープが補償された Q16.16 固定小数点形式です。
- 「A」は 16 ビットの乗数です。A の値は、PSoC 4 ファミリの特性データから 2 点のスロープを計算し決定します。この値は次の式に従って計算されます。

$$A = (\text{signedint})\left(2^{16}\left(\frac{100°C-(-40°C)}{SAR_{100°C}-SAR_{-40°C}}\right)\right)$$ **式 25-2**

ここで、

SAR_{100C} = 100°C での ADC カウント

SAR_{-40C} = –40°C での ADC カウント

定数「A」は SFLASH_SAR_TEMP_MULTIPLIER レジスタに保存されます。

- 「B」は 16 ビットのオフセット値です。B の値は、すべてのプロセスばらつきとチップ上の実際のバイアス電流 (I_{bias}) を考慮することによってダイ単位で決定されます。この値は次の式に従って計算されます。

$$B = (\text{unsignedint})\left(2^{6}x100°C-\left(\frac{A\times SAR_{100C}}{2^{10}}\right)\right)$$ **式 25-3**

ここで、

SAR_{100C} = 100°C での ADC カウント

定数「B」は SFLASH_SAR_TEMP_OFFSET レジスタに保存されます。

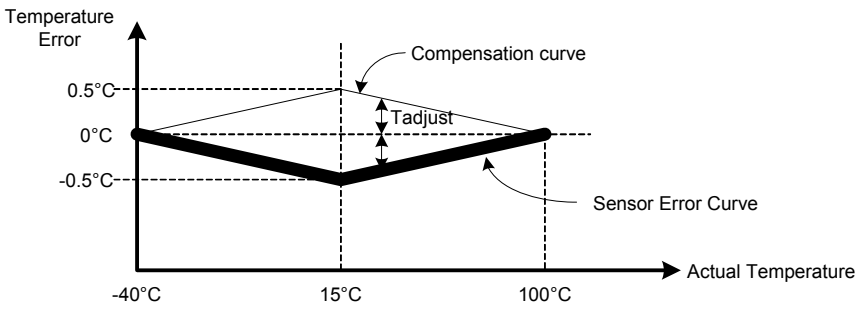
- T_{adjust} は℃単位でのスロープ補正係数です。温度センサーはスロープ補正係数を使用してデュアル スロープに補正されます。これはスロープ補正なしで得た結果に基づいて計算されます。すなわち $T_{initial} = (A\times SAR_{out}+ 2^{10}\times B)$ と評価します。中心値 (15°C) より高い場合、T_{adjust} は以下の式で計算されます。

$$T_{adjust} = \left(\frac{0.5°C}{100°C-15°C}\times(100°C\times 2^{16}-T_{initial})\right)$$ **式 25-4**

中心値より低い場合、Tadjust は以下の式で計算されます。

$$T_{adjust} = \left(\frac{0.5°C}{40°C+15°C}\times(40°C\times 2^{16}-T_{initial})\right)$$ **式 25-5**

図 25-2. 温度誤差補正



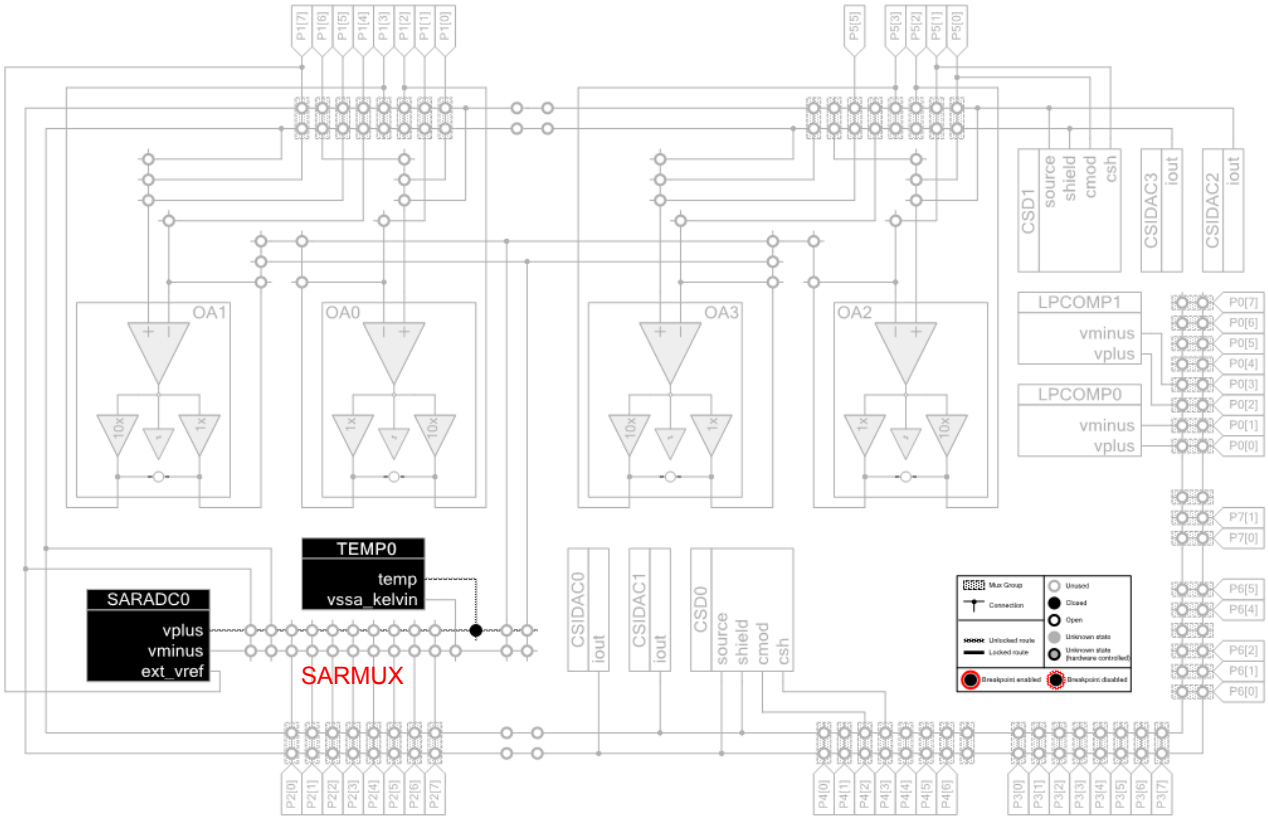
注 : A と B は工場で較正されフラッシュに保存される 16 ビットの定数です。これらの定数は SAR ADC が 1.024V のリファレンス電圧と 12 ビットの分解能で動作している場合にのみ有効になることに注意してください。

25.3 温度センサー コンフィギュレーション

図 25-3 に示すように温度センサーの出力はシーケンサー、ファームウェアまたはデジタル システム インターコネクト (DSI) に制御される専用のスイッチを介して SAR ADC の＋入力に接続されます。スイッチ (図 25-1 に示す SAR_MUX_FW_TEMP_VPLUS) の制御信号は、高精度リファレンスブロックからのバイアス電流を通し、センサーの出力を SAR ADC の＋入力に接続することにより温度センサーを有効にします。SAR_MUX_FW_TEMP_VPLUS の制御ビットは SAR_MUX-

_SWITCH0 レジスタの一部です。スイッチの状態は SAR_MUX_SWITCH_STATUS レジスタを使用して読み出すことができます。

図 25-3. SAR ADC への温度センサー出力の配線



25.4 アルゴリズム

1. SAR MUX と SAR ADC を有効にします。
2. V_{NEG} = V_{SS}、V_{REF} = 1.024V、12 ビット分解能および右アライメントの結果を得るように SAR ADC をシングルエンドモードに設定します。
3. 温度センサーを有効にします。
4. SAR ADC からデジタル出力を取得します。
5. 「A」と「B」をそれぞれ SFLASH_SAR_TEMP_MULTIPLIER と SFLASH_SAR_TEMP_OFFSET から取り出します。
6. 直線化式 (式 25-1) を使用してダイ温度を計算します。
   例えば A = 0xBC4B、B = 0x65B4 とします。所定の温度で SAR ADC の出力 (V_{BE}) が 0x595 であると仮定します。
   ファームウェアは以下の計算を行います。
   a. A に V_{BE} を掛けます : 0xBC4B × 0x595 = $(–17333)_{10}$ × $(1429)_{10}$ = $(–24768857)_{10}$
   b. B に 1024 を掛けます : 0x65B4 × 0x400 = $(26036)_{10}$ × $(1024)_{10}$ = $(26660864)_{10}$
   c. ステップ 1 とステップ 2 の結果を加算し $T_{initial}$ を得ます : $(–24768857)_{10}$ + $(26660864)_{10}$ = $(1892007)_{10}$ = 0x1CDEA7
   d. $T_{initial}$ 値を使用して T_{adjust} を計算します : $T_{initial}$ は上位 16 ビットに 2^{16} を掛けた結果です。つまり 0x1C00 = $(1835008)_{10}$ です。この値は 15°C (0x1C - 上位 16 ビット) より高いです。式 4 を使って T_{adjust} を計算します。その結果は 0x6C6C = $(27756)_{10}$ となります。

e. T_{adjust} と $T_{initial}$ を加算します : $(1892007)_{10} + (27756)_{10} = (1919763)_{10}$ = 0x1D4B13
f. 温度の整数部分は上位 16 ビット = 0x001D = $(29)_{10}$ です。
g. 温度の小数部は下位 16 ビット = 0x4B13 = $(0.19219)_{10}$ です。
h. ステップ f とステップ g の結果を組み合わせると、温度 = 29.19219°C ~ 29.2°C となります。

25.5 レジスタ

レジスタ名	説明
SAR_MUX_SWITCH0	このレジスタは、温度センサーを SAR マルチプレクサ端子に接続するための SAR_MUX_FW_-TEMP_VPLUS フィールドを持つ
SAR_MUX_SWITCH_STATUS	このレジスタは SAR マルチプレクサへの温度センサー スイッチ接続の状態を示す
SFLASH_SAR_TEMP_MULTIPLIER	式 25-1 に定義される乗数「A」
SFLASH_SAR_TEMP_OFFSET	式 25-1 に定義される定数「B」

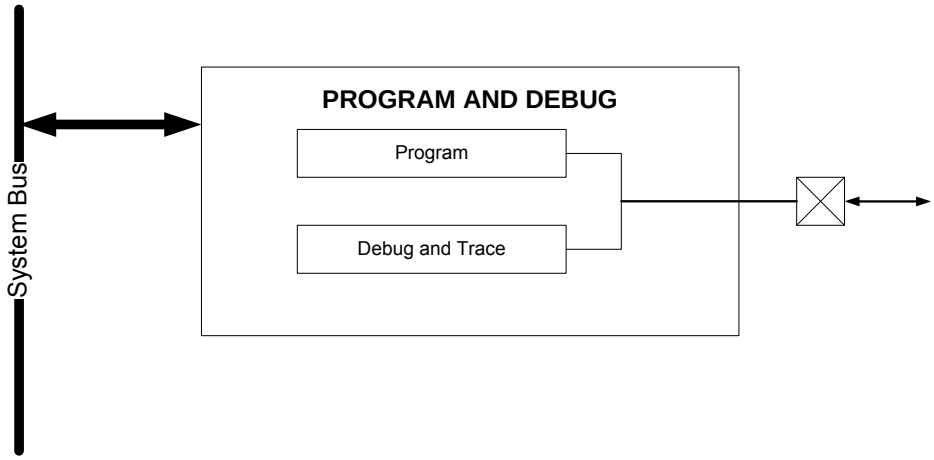
セクション F: プログラムおよびデバッグ

このセクションは次の章を含みます。

- デバッグ インターフェース (299 ページ)
- 不揮発性メモリ プログラム (307 ページ)

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部のブロック図



26. 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ています。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 PSoC 4 の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。

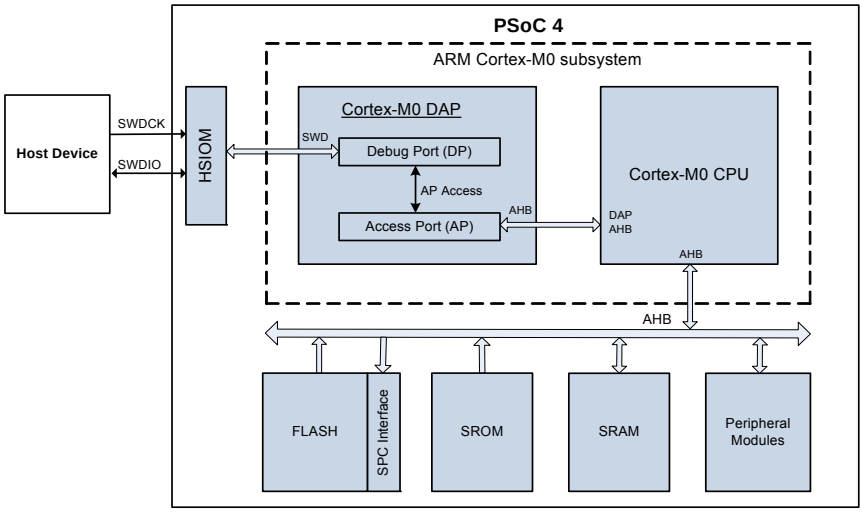
26.1 特長

- 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
- 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実装
- デバッグ期間中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

26.2 機能の詳細

図 26-1 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 デバッグ 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 の詳細については、59 ページの I/O システムを参照してください。

図 26-1. PSoC 4 のプログラムおよびデバッグ インターフェース



DAP は、ARM 特有の 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 PSoC 4 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。PSoC 4 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。

26.3 シリアル ワイヤ デバッグ (SWD) インターフェース

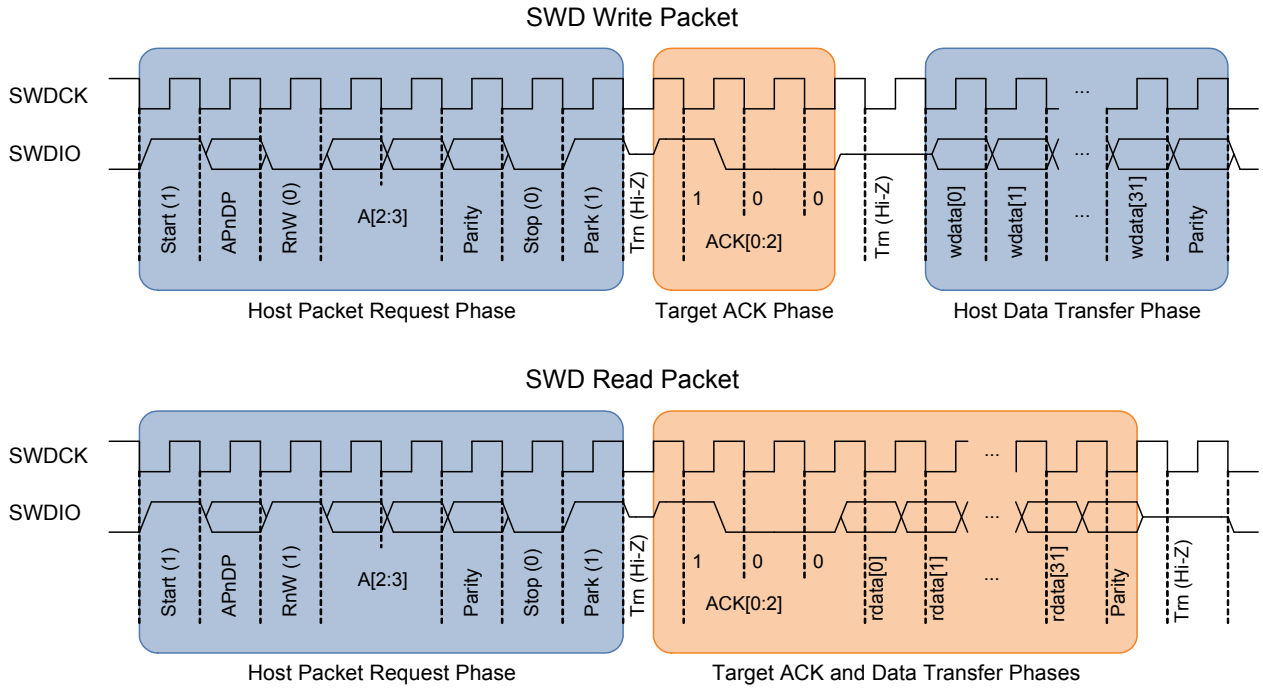
PSoC 4 の Cortex-M0 は、SWD インターフェースを介したプログラムおよびデバッグに対応しています。SWD プロトコルは、パケット ベースのシリアル トランザクション プロトコルです。ピン レベルでは双方向データ信号 (SWDIO) と一方向クロック信号 (SWDCK) を使用します。クロック ラインは常にホスト プログラマが駆動し、データ ラインはホストまたはターゲットのいずれかが駆動します。完全なデータ転送 (1 つの SWD パケット) は 46 クロックを必要とし、次の 3 つの段階から成り立っています。

- **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 26-2 に SWD パケットの読み書きタイミング図を示します。

図 26-2. PSoC 4 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 26-3 および表 26-4 をご覧ください。**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 26-1 をご覧ください。**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。

26.3.1 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 26-1 および図 26-2 に SWDIO ビットの読み書きタイミングを示します。

表 26-1. SWDIO ビット読み書きタイミング

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		

26.3.2 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 26-2 は ACK ビット フィールド コードの詳細を示します。

表 26-2. 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 応答の場合、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。

26.3.3 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 26-2 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。これ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。

26.4 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実装します。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® 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。

26.4.1 デバッグ ポート (DP) レジスタ

表 26-3 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注意してください。すべての DP レジスタの詳細については、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 26-3. 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	フルネーム	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 0x0BB11477 である Cortex-M0 CPU の SWD ID を保持
CTRL/ STAT	0 (DP)	2b01	X (R/W)	制御レジスタとステータス レジスタ	このレジスタは DP の制御を可能にし、DP のステータス情報を含む
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

26.4.2 アクセス ポート (AP) レジスタ

表 26-4 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 26-4. 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	フルネーム	レジスタの機能
CSW	1 (AP)	2b00	X (R/W)	制御レジスタとステータス 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

26.5 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 4100M/4200M デバイスのプログラム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。

26.5.1 SWD ポートの開通

26.5.1.1 1 次と 2 次の SWD ピンの組

デバイス プログラムにおける最初の手順は、PSoC 4 に SWD ポートを開通させることです。SWD ピンの情報については、デバイス データシート を参照してください。

SWD ピンの二組がデバイス内で利用可能な場合、監視フラッシュ領域の SWD_CONFIG レジスタは、プログラムおよびデバッグ動作に使用する SWD ピンの二組から一組を選択します。あらゆるプログラムおよびデバッグ セッションにおいて、SWD ピンの一組のみが使用されることに注意し

てください。工場出荷時のデバイスでは、一次の SWD ピンの組がデフォルトの選択です。2 次の SWD ピンの組を選択するには、その 2 次のピンの組コンフィギュレーションを有効にする HEX ファイルで 1 次のピンの組を使用してデバイスをプログラムする必要があります。そうすると 2 次の SWD ピンの組を使用することが可能になります。

26.5.1.2 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 < 1.5 ms); // retry connection until OK ACK or timeout

if (time_elapsed >= 1.5 ms) return FAIL; //check for acquire time out

if (ID != CM0_ID) return FAIL; //confirm SWD ID of Cortex-M0 CPU. (0x0BB11477)
```

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 (1.5ms) 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。

26.5.2 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/4200M デバイスのプログラム仕様の資料に詳しく記載しています。

26.5.3 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 307 ページの不揮発性メモリ プログラム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 4100M/4200M デバイスのプログラム仕様の資料で記載されています。

26.6 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 サポートを提供します。
- デバッグ ウォッチポイント (DWT) – ウォッチポイント サポートを提供します。トレース実行は Cortex-M0 デバッグでサポートされていません。

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。

26.6.1 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 をご覧ください。

- デバッグ ホールト制御およびステータス レジスタ (CM0_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 (CM0_DFSR) – このレジスタは、デバッグ イベントが発生した原因を説明します。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 (CM0_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 (CM0_DCRDR) – このレジスタは、CM0_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 (CM0_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 フォー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。

26.6.2 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 (CM0_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 (CM0_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。

26.6.3 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。トレース実行は PSoC 4 の Cortex-M0 によってサポートされていません。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 (CM0_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 (CM0_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 (CM0_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 (CM0_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。

26.6.4 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 デバイスをデバッグします。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 デバイス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か LCD と GPIO 等のような他の機能として使用するかのコンフィギュレーションについては 59 ページの I/O システム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 93 ページのデバイス セキュリティ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。またデバイスがハイバネート モードまたはストップ モードのいずれかの場合、外部デバッガはターゲット デバイスへの接続を失います。デバイスが再びアクティブ モードに入った後、接続を復旧しなければなりません。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。

26.7 レジスタ

表 26-5. レジスター覧

レジスタ名	説明
CM0_DHCSR	デバッグ ホールト制御およびステータス レジスタ
CM0_DFSR	デバッグ フォールト ステータス レジスタ
CM0_DCRSR	デバッグ コア レジスタ セレクタ レジスタ
CM0_DCRDR	デバッグ コア レジスタ データ レジスタ
CM0_DEMCR	デバッグ例外およびモニター制御レジスタ
CM0_BP_CTRL	ブレークポイント制御レジスタ
CM0_BP_COMPx	ブレークポイント比較レジスタ
CM0_DWT_COMPx	ウォッチポイント比較レジスタ
CM0_DWT_MASKx	ウォッチポイント マスク レジスタ
CM0_DWT_FUNCTIONx	ウォッチポイント機能レジスタ
CM0_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

27. 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。

27.1 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

27.2 機能の詳細

フラッシュ プログラム動作はシステム コールとして実装されます。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。これ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で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/4200M 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。この期間中にデバイスをリセットしてはいけません。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 (89 ページのリセット システムを参照) 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確認してください。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。

27.3 システム コールの実装

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 つのキーを渡すことでユーザー 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 コードを返します。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。
- 返し値 : 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返します。
- 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返して、CPU あるいは DAP はそれを読み込み成否を検証します。

27.4 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 非ブロッキングが 3 つあります。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 エラーを返し、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。

27.4.1 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て、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注意してください。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 27-1 を参照してください。
5. 返し値の回収: シリコン ID やチェックサムなどの値を返すシステム 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返し値をフェッチします。

27.5 システム コール

表 27-1 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 93 ページのデバイス セキュリティ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注意してください。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 27-1. 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
		Open	Protected	Kill	
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 を返する	✔	✔	–	✔
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 バッファにロードする 1 バイトの単位で最大列サイズ 128 バイトまで	✔	–	–	✔
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	–	–	–	✔

27.5.1 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返します。これらの値は CPUSS_SYSARG と CPUSS_SYSREQ レジスタに返されます。パラメーターは CPUSS_SYSARG と CPUSS_SYSREQ 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

返し値

アドレス	返し値	説明
CPUSS_SYSARG レジスタ		
ビット [7:0]	シリコン ID Lo	異なる部品番号のシリコン ID 値の詳細は デバイス データシートを参照してください
ビット [15:8]	シリコン ID Hi	
ビット [19:16]	マイナー リビジョン Id	これらの値の詳細は PSoC 4100M/4200M プログラム仕様 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 4200M と PSoC 4100M のファミリ ID は 0x0A1
ビット [15:12]	チップ保護	詳細については 93 ページのデバイス セキュリティ
ビット [31:16]	0xXXXX	未使用

27.5.2 ロード フラッシュ バイト

この関数 は、1 列のフラッシュでプログラムされるデータとページ 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 バイトまでです。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が実行する時まで残ります。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注意してください。

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]	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 0x80 – ラッチ バッファのバイト 128

アドレス	書き込まれる値	説明
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 31 ページの Cortex-M0 CPU 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 ビットのワードアライン 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

返し値

アドレス	返し値	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)

27.5.3 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のみに書き込み動作を行うことができます。

このシステム コールはフラッシュ 書き込み動作を実行する前に、36MHz IMO の出力を無効にすることに注意してください。36MHz IMO の出力は アナログ スイッチ ポンプまたは CTBm ポンプのソースに使用することが可能です。36MHz IMO の出力が使用される場合、システム コールが完了した後、手動で再度有効にする必要があります。特に CLK_IMO_CONFIG EN_CLK36 および FLASHPUMP_SEL はリセットする必要があります。

詳細情報については PSoC 4100M/4200M ファミリ : PSoC 4 レジスタ 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

返し値

アドレス	返し値	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)

27.5.4 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 1 列のフラッシュ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 ラッチバッファの中味をクリアします。

注意事項 : 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のみに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

返し値

アドレス	返し値	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)

27.5.5 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 マクロの監視フラッシュの列 0 の低レベル保護データを消去します。

注意事項 : 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

返し値

アドレス	返し値	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)

27.5.6 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 24 ビットを返します。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 コードおよび監視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 メモリの全体とユーザー 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

返し値

アドレス	返し値	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

27.5.7 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監視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、監視フラッシュの列 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。監視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 : 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のに使用されます。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。

それから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

返し値

アドレス	返し値	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	

27.5.8 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のある時に使用され、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 308 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。スリープモードにするとページ ラッチ バッファがリセットされフラッシュに 0 が書き込まれます。

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
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]	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

返し値

アドレス	返し値	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)

27.5.9 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のある時に使用され、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 308 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 コールの使用の時に、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、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]	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

返し値

アドレス	返し値	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)

27.5.10 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 308 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

返し値

アドレス	返し値	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)

27.6 システム コール ステータス

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 コールが値を返す場合に「ドントケア」値または返し値を示します。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 27-2. 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 : API が CPUSS_SYSARG レジスタにパラメーターを直接返さない限り、ドント ケア値「X」は SROM が返す値である 0
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る
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る

表 27-2. 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
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

27.7 非ブロッキング システム コール疑似コード

このセクションはフラッシュ 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例示します。

```
#define REG(addr)(*((volatile uint32 *) (addr)))
#define CM0_ISER_REG REG( 0xE000E100 )
#define CPUSS_CONFIG_REGREG( 0x40100000 )
#define CPUSS_SYSREQ_REG REG( 0x40100004 )
#define CPUSS_SYSARG_REG REG( 0x40100008 )
#define ROW_SIZE              128

//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)
{
   /* Call Resume API */
   // 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;
   iStatusInt ++; // Number of times the ISR has triggered
}
 __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 47 ページの割り込み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
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 (スイッチト 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

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位のサービスと関数 (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級言語と見なされるのに対して、C 言語は高級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. リファレンス値と値一式の平均値間の誤差
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、方程式を実装す 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アンプ

bus (バス)
1. 複数ラインの名前付き接続。バスとしてライン同士を束にすることにより、類似した配線パターンを持つラインの配線が容易になる
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 のような高級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))

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装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

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、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

NOT (論理否定) ブール代数を参照

O

OR (論理和) ブール代数を参照

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
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
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

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。またはデバイスがカスケード接続されている場合、スレーブ デバイスが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 他のデバイスに制御させるもの。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) 実際の実装。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レベルの 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

索引

A

アクティブ モード
PSoC 82
アナログ I/O 64

B

ブロック図
GPIO 60
ポート割り込み制御ユニット 65
プログラムおよびデバッグ インターフェース . . 299
ウォッチドッグ タイマー回路 85
ブラウンアウト リセット 89

C

クロック分配 70
クロック源
分配 70
クロック供給システム
はじめに 67
Cortex-M0
特長 31
命令セット 34
レジスタ 33

D

開発キット 23
ドキュメント
用語集 321
改訂履歴 15

E

例外
HardFault 50
NMI 49
PendSV 50
リセット 49
SVCall 50
SysTick 50
外部リセット 89

F

特長
I/O システム 59
ポート割り込み制御ユニット 65
ウォッチドッグ タイマー 85

G

用語 321
GPIO
ブロック図 60
GPIO ピンを使った CapSense のボタンとスライダの作成 64

H

ハイバネート モード 83
ハイバネート ウェイクアップ リセット 90
高インピーダンス アナログ駆動モード 62
高インピーダンス デジタル駆動モード 62
動作原理
ウォッチドッグ タイマー 86

I

I/O 駆動モード
高インピーダンス アナログ 62
高インピーダンス デジタル 62
オープン ドレイン 62
抵抗 62
strong (ストロング) 62
I/O システム
アナログ I/O 64
CapSense 64
特長 59
はじめに 59
LCD 駆動力 64
オープン ドレイン モード 62
ポート割り込み制御ユニットの
ピン コンフィギュレーション 66
レジスタの概要 66
抵抗 66
スルーレート制御 63
ストロング駆動モード 62
リセット ソースの識別 90
内部低速発振器 70

内部主発振器 67
内部レギュレータ 75
はじめに
　クロック ジェネレーター 67
　I/O システム 59
　リセット 89
　逐次近似レジスタアナログデジタルコンバータ . 221

L

LCD 駆動
　I/O システムの機能 64

O

発振器
　内部 PSoC 67
概要、ドキュメント
　改訂履歴 15

P

ポート割り込み制御ユニット
　特長 65
ポート割り込み制御ユニット
　ブロック図 65
　ピン コンフィギュレーション 66
パワーオンリセット 89
プログラムおよびデバッグ
　PSoC 22
保護フォルト リセット 90
PSoC
　アクティブ モード 82
　プログラムおよびデバッグ 22

R

レジスタの概要
　I/O システム 66
レジスタ
　Cortex-M0 33
レギュレータ
　内部 75
リセット
　リセット ソースの識別 89
　はじめに 89
リセット ソース
　説明 89
改訂履歴 15

S

SAR ADC
　はじめに 221
スリープ モード 82
I/O システムのスルーレート制御 63
ソフトウェア初期化リセット 90
ストップ モード復帰リセット 90
サポート 23
SWD インターフェース
　プログラムおよびデバッグ インターフェース . . 300
システム コール
　概要 308

U

アップグレード 23

W

ウォッチ ドッグ リセット 89
ウォッチドッグ タイマー
　無効化 87
　有効化 87
　特長 85
　動作原理 86
　割り込み 88
　動作モード 87