

サイプレスはインフィニオン テクノロジーズになりました

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

文書の内容の継続性

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

注文時の部品番号の継続性

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



PSoC 4100S および PSoC 4100S Plus

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)

Document No. 002-16642 Rev. *A

February 25, 2021

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

Copyrights

© Cypress Semiconductor Corporation, 2016-2021. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社（以下「Cypress」という。）に帰属する財産である。本書面（本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア（以下「本ソフトウェア」という。）を含む）は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、本段落で特に記載されているものを除き、その特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず、かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合、Cypress は、(1) 本ソフトウェアの著作権に基づき、(a) ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、かつ組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ、（直接又は再販売者及び販売代理店を介して間接のいずれかで）本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること、並びに (2) 本ソフトウェア (Cypress により提供され、修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス（サブライセンスの権利を除く）を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関して、明示又は黙示をとわず、いかなる保証（商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない）も行わない。いかなるコンピューティングデバイスも絶対に安全ということはない。従って、Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず、Cypress は、Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて、本書面に記載された製品には、エラーと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり、公表された仕様とは異なる動作をする場合がある。適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のある、いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報（あらゆるサンプルデザイン情報又はプログラムコードを含む）は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計、プログラム、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用、又は装置若しくはシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせるようなその他の使用（以下「本目的外使用」という。）のためには設計、意図又は承認されていない。重要な構成部分とは、その不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部をとわず一切の責任を負わず、かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本来目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任（人身傷害又は死亡に基づく請求を含む）から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, Capsense, EZ-USB, F-RAM, 及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは、cypress.com を参照すること。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。

Content Overview



Section A: 概要	3
1. はじめに	17
2. 利用の手引き	22
3. 文書の構成	23
Section B: CPUシステム	27
4. Cortex-M0+ CPU	28
5. DMAコントローラ モード	34
6. 割り込み	51
7. デバイスセキュリティ	63
Section C: システムリソースサブシステム (SRSS)	65
8. I/Oシステム	66
9. クロック供給システム	89
10. 電源と監視	101
11. チップ動作モード	105
12. 電力モード	107
13. ウォッチドッグタイマ	111
14. トリガーマルチプレクサブロック	116
15. リセット システム	120
Section D: デジタル システム	123
16. シリアル通信ブロック(SCB)	124

17. コントローラエリアネットワーク (CAN)	171
18. タイマ、カウンタ、およびPWM	192
19. 真の乱数生成器	218
20. LCD直接駆動	223
Section E: アナログシステム	237
21. SAR ADC	238
22. 低消費電力コンパレータ	266
23. 連続時間ミニブロック (CTBm)	273
24. CapSense	283
25. 温度センサー	284
Section F: プログラムとデバッグ	287
26. デバッグ インターフェース	288
27. 不揮発性メモリ プログラム	296
用語集	312

Contents



Section A: 概要	3
1. はじめに	17
1.1 トップレベルアーキテクチャ	17
1.2 特長	18
1.3 CPUシステム	19
1.3.1 プロセッサ	19
1.3.2 割込みコントローラ	19
1.3.3 Direct Memory Access、ダイレクトメモリアクセス	19
1.3.4 メモリ	19
1.3.4.1 フラッシュメモリ	19
1.3.4.2 SRAM	19
1.4 システムワイドリソース	19
1.4.1 クロック供給システム	19
1.4.2 電力システム	19
1.4.3 GPIO	20
1.4.4 ウォッチドッグタイマ	20
1.5 固定機能デジタルブロック	20
1.5.1 タイマ/カウンタ/PWMブロック	20
1.5.2 シリアル通信ブロック	20
1.5.3 コントローラエリアネットワーク(CAN)	20
1.5.4 真性乱数発生器 (TRNG)	20
1.5.5 LCDセグメント駆動	20
1.6 アナログシステム	20
1.6.1 SAR ADC	20
1.6.2 連続時間ブロック ミニ	20
1.6.3 低消費電力コンパレータ	21
1.7 特殊周辺機能	21
1.7.1 CapSense	21
1.7.1.1 IDACとコンパレータ	21
1.8 プログラムとデバッグ	21
1.9 デバイス機能の要約	21
2. 利用の手引き	22
2.1 対応	22
2.2 製品アップグレード	22
2.3 開発キット	22
2.4 アプリケーションノート	22
3. 文書の構成	23
3.1 主なセクション	23
3.2 本書の表記法	23

3.2.1	レジスタの表記法	23
3.2.2	数値の表記法	23
3.2.3	測定単位	24
3.2.4	略語	24
Section B: CPUシステム		27
4. Cortex-M0+ CPU		28
4.1	特長	28
4.2	ブロック図	29
4.3	動作原理	29
4.4	アドレスマップ	29
4.5	レジスタ	30
4.6	動作モード	31
4.7	命令セット	31
4.7.1	アドレス アライメント	32
4.7.2	メモリ エンディアン	32
4.8	Systickタイマ	32
4.9	デバッグ	33
5. DMAコントローラ モード		34
5.1	ブロック図の解説	34
5.1.1	トリガーソースと多重化	35
5.1.1.1	トリガーマルチプレクサ	35
5.1.1.2	ソフトウェアトリガーの作成	38
5.1.2	保留中のトリガー	38
5.1.3	出力トリガー	38
5.1.4	チャネルの優先順位付け	38
5.1.5	データ転送エンジン	39
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	42
5.2.3	記述子の連鎖	42
5.2.4	転送モード	43
5.2.4.1	トリガーごとに単一のデータ要素 (OPCODE 0)	43
5.2.4.2	トリガーごとの記述子全体 (OPCODE 1)	45
5.2.4.3	トリガーごとの記述子チェーン全体 (OPCODE 2)	45
5.3	動作とタイミング	47
5.4	アービトレーション	48
5.5	レジスタ一覧	50
6. 割り込み		51
6.1	特長	51
6.2	動作原理	51
6.3	割り込みと例外の動作	52
6.3.1	割り込み/例外の処理	52
6.3.2	レベルとパルスの割り込み	52
6.3.3	例外ベクタ テーブル	53
6.4	例外ソース	54
6.4.1	リセット例外	54
6.4.2	マスク不可能割り込み(NMI)例外	54
6.4.3	HardFault例外	54

6.4.4	監視呼び出し(SVCall)例外	55
6.4.5	PendSV例外	55
6.4.6	SysTick例外	55
6.5	割込みソース	56
6.6	例外優先度	59
6.7	割込みのイネーブルとディセーブル	59
6.8	例外状態	60
6.8.1	保留中の例外	60
6.9	例外のスタック使用量	60
6.10	割込みと低消費電力モード	61
6.11	例外-初期化と構成	61
6.12	レジスタ	62
6.13	関連文書	62
7.	デバイスセキュリティ	63
7.1	特長	63
7.2	動作原理	63
7.2.1	デバイスセキュリティ	63
7.2.2	フラッシュのセキュリティ	64
Section C:	システムリソースサブシステム (SRSS)	65
8.	I/Oシステム	66
8.1	特長	66
8.2	GPIOインターフェースの概要	66
8.3	I/Oセルのアーキテクチャ	68
8.3.1	デジタル入力バッファ	69
8.3.2	デジタル出力ドライバ	69
8.3.2.1	駆動モード	69
8.3.2.2	スルーレート制御	71
8.4	高速I/Oマトリックス	71
8.5	スマートI/O	73
8.5.1	概要	73
8.5.2	ブロック コンポーネント	74
8.5.2.1	クロックとリセット	74
8.5.2.2	シンクロナイザー	75
8.5.2.3	LUT3	75
8.5.2.4	データ ユニット	79
8.5.3	配線	83
8.5.4	動作	84
8.6	電源投入時のI/O状態	85
8.7	省電力モードでの動作	85
8.8	割込み	85
8.9	ペリフェラルの接続	86
8.9.1	ファームウェアで制御されるGPIO	86
8.9.2	アナログIO	87
8.9.3	LCD駆動	87
8.9.4	CapSense	87
8.9.5	シリアル通信ブロック(SCB)	87
8.9.6	タイマ/カウンタ/パルス幅変調器 (TCPWM)ブロック	88
8.10	レジスタ	88

9. クロック供給システム	89
9.1 ブロック図	89
9.2 クロック ソース	90
9.2.1 内部主発振器 (IMO)	90
9.2.1.1 スタートアップ動作	91
9.2.1.2 プログラミング クロック (36MHz)	92
9.2.2 内部低速発振器 (ILO)	92
9.2.3 外部クロック (EXTCLK)	92
9.2.4 外部水晶発振器 (ECO)	92
9.2.4.1 ECOトリミング	92
9.2.5 PLL	93
9.2.6 時計用水晶発振器 (WCO)	94
9.3 クロック分配	94
9.3.1 HFCLKおよびPLL入力の選択	94
9.3.2 HFCLK入力の選択	95
9.3.3 LFCLK入力の選択	95
9.3.4 SYSCLKプリスケアラのコンフィギュレーション	96
9.3.5 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	96
9.4 低消費電力動作	99
9.5 レジスター一覧	100
10. 電源と監視	101
10.1 ブロック図	101
10.2 電源供給のシナリオ	102
10.2.1 1.8V~5.5Vの単一の非安定化電源	102
10.2.2 直接1.71V~1.89 Vの安定化電源	102
10.3 動作原理	103
10.3.1 レギュレータの概要	103
10.3.1.1 アクティブ デジタル レギュレータ	103
10.3.1.2 ディープスリープ レギュレータ	104
10.4 電圧監視	104
10.4.1 パワーオンリセット(POR)	104
10.4.1.1 電圧低下検出(BOD)	104
10.5 レジスター一覧	104
11. チップ動作モード	105
11.1 ブート	105
11.2 ユーザ	105
11.3 特権	105
11.4 デバッグ	105
12. 電力モード	107
12.1 アクティブモード	108
12.2 スリープモード	108
12.3 ディープスリープモード	108
12.4 電力モードの概要	109
12.5 低消費電力モードへの移行および復帰	110
12.6 レジスター一覧	110
13. ウォッチドッグタイマ	111
13.1 特長	111

13.2	ブロック図	111
13.3	動作原理	112
13.3.1	WDTの有効化と無効化	112
13.3.2	WDT割込みと低消費電力モード	113
13.3.3	WDTリセット モード	113
13.4	追加タイマ	113
13.4.1	WDT0およびWDT1	114
13.4.2	WDT2	114
13.4.3	カスケード	114
13.5	レジスター一覧	115
14.	トリガーマルチプレクサブロック	116
14.1	特長	116
14.2	アーキテクチャ	116
14.2.1	マルチプレクサグループのトリガー	117
14.2.2	ソフトウェアトリガー	119
14.3	レジスター一覧	119
15.	リセット システム	120
15.1	リセットソース	120
15.1.1	パワーオン リセット	120
15.1.2	ブラウンアウトリセット	120
15.1.3	ウォッチドッグリセット	120
15.1.4	ソフトウェア開始リセット	121
15.1.5	外部リセット	121
15.1.6	フォールトリセットプロテクション	121
15.2	リセットソースの識別	121
15.3	レジスター一覧	122
Section D:	デジタル システム	123
16.	シリアル通信ブロック(SCB)	124
16.1	特長	124
16.2	シリアル ペリフェラル インターフェース(SPI)	124
16.2.1	特長	124
16.2.2	概要説明	125
16.2.3	SPI動作モード	126
16.2.3.1	Motorola社SPI	126
16.2.3.2	Texas Instruments社SPI	128
16.2.3.3	National Semiconductor社SPI	130
16.2.4	スレーブへのクロック供給用のSPIマスターの使用	131
16.2.5	イーザーSPIプロトコル	131
16.2.5.1	EZアドレス書き込み	131
16.2.5.2	メモリ アレイ書き込み	131
16.2.5.3	メモリ アレイの読み出し	131
16.2.5.4	EZSPIモード用のSCBの設定	132
16.2.6	SPIレジスタ	133
16.2.7	SPI割込み	134
16.2.8	SPIの有効化と初期化	134
16.2.9	内部と外部クロック供給SPI動作	136
16.2.9.1	非EZの動作モード	137

16.2.9.2	EZ動作モード	138
16.3	UART	139
16.3.1	特長	139
16.3.2	概要説明	139
16.3.3	UARTの動作モード	139
16.3.3.1	標準プロトコル	139
16.3.3.2	スマートカード (ISO7816)	147
16.3.3.3	IrDA	148
16.3.4	UARTレジスタ	149
16.3.5	UART割込み	149
16.3.6	UARTの有効化と初期化	150
16.4	インター インテグレートッド サークット (I2C)	151
16.4.1	特長	151
16.4.2	概要説明	152
16.4.3	用語および定義	153
16.4.3.1	クロック ストレッチ	153
16.4.3.2	バス アービトレーション	153
16.4.4	I2C動作モード	153
16.4.4.1	書き込み転送	154
16.4.4.2	読み出し転送	154
16.4.5	イージーI2C (EZI2C)プロトコル	155
16.4.5.1	メモリ アレイ書き込み	155
16.4.5.2	メモリ アレイの読み出し	156
16.4.6	I2Cレジスタ	157
16.4.7	I2C割込み	158
16.4.8	I2Cの有効化および初期化	158
16.4.8.1	I2C標準 (非EZ) モード構成	158
16.4.8.2	EZI2Cモード構成	159
16.4.9	I2Cにおける内部および外部クロック動作	160
16.4.9.1	I2Cの非EZ動作モード	160
16.4.9.2	EZモードでのI2C動作	161
16.4.10	スリープから復帰	161
16.4.11	マスター モード転送の例	163
16.4.11.1	マスター送信	163
16.4.11.2	マスター受信	164
16.4.12	スレーブ モード転送の例	165
16.4.12.1	スレーブ送信	165
16.4.12.2	スレーブ受信	166
16.4.13	EZスレーブ モード転送の例	167
16.4.13.1	EZスレーブ送信	167
16.4.13.2	EZスレーブ受信	168
16.4.14	マルチマスター モード転送の例	169
16.4.14.1	マルチマスター-スレーブが無効	169
16.4.14.2	マルチマスター-スレーブが有効	170
17.	コントローラエリアネットワーク (CAN)	171
17.1	特長	171
17.2	ブロック図	172
17.3	CANメッセージ フレーム	173
17.3.1	データフレーム	173
17.3.1.1	標準データ フレーム	173
17.3.1.2	拡張データ フレーム	174

17.3.2	リモート フレーム	174
17.3.3	エラー フレーム	174
17.3.4	オーバーロード フレーム	174
17.4	CANにおけるメッセージ送信	175
17.4.1	メッセージ アービトレーション	175
17.4.2	メッセージ送信プロセス	175
17.4.3	メッセージ中止	176
17.4.4	シングル ショット送信	176
17.4.5	拡張データ フレームの送信	177
17.5	CANにおけるメッセージ受信	177
17.5.1	メッセージ受信プロセス	177
17.5.2	受け入れフィルタ	178
17.5.2.1	例	179
17.5.3	DeviceNetフィルタ処理	179
17.5.4	拡張データ フレームのフィルタ処理	180
17.5.5	受信メッセージ バッファの連結	180
17.6	リモート フレーム	181
17.6.1	要求ノードによるリモート フレームの送信	181
17.6.2	リモート フレームの受信	182
17.6.3	RTR自動応答	182
17.6.4	拡張フォーマットでのリモート フレーム	182
17.7	時間起動CAN	182
17.7.1	TTCANタイマ	182
17.8	ビット タイム コンフィギュレーション	184
17.8.1	取り得るビット レートとシステム クロック(SYSCLK)	184
17.8.2	ビット レートTSEG1とTSEG2の設定	186
17.8.2.1	例	186
17.9	CANのエラーと割り込み	187
17.9.1	エラーの種類	187
17.9.1.1	BITエラー	187
17.9.1.2	FORMエラー	187
17.9.1.3	ACKNOWLEDGEエラー	187
17.9.1.4	CRCエラー	187
17.9.1.5	STUFFエラー	187
17.9.2	エラー キャプチャ レジスタ	187
17.9.3	CANにおけるエラー状態	188
17.9.4	CANにおける割り込みソース	188
17.9.4.1	CANからのコア割り込み	188
17.9.4.2	TT_ENABLE = 1による割り込みルーティング	189
17.10	CANにおける動作モード	191
17.10.1	実行/停止モード	191
17.10.2	リッスンオンリー モード	191
17.10.3	ループバック テスト モード	191
17.10.3.1	外部ループバック モード	191
17.10.3.2	内部ループバック モード	191
18.	タイマ、カウンタ、およびPWM	192
18.1	特長	192
18.2	ブロック図	193
18.2.1	TCPWMブロックにおけるカウンタの有効化/無効化	193
18.2.2	クロック	193
18.2.3	トリガー入力に基づいたイベント	194

18.2.4	出力信号	197
18.2.4.1	トリガー条件発生時の信号	197
18.2.4.2	割り込み	198
18.2.4.3	出力	198
18.2.5	電力モード	199
18.3	動作モード	199
18.3.1	タイマモード	200
18.3.1.1	ブロック図	200
18.3.1.2	動作原理	200
18.3.1.3	タイマモードの設定方法	202
18.3.2	キャプチャモード	203
18.3.2.1	ブロック図	203
18.3.2.2	動作方法	203
18.3.2.3	キャプチャモードの設定方法	204
18.3.3	直交デコーダモード	205
18.3.3.1	ブロック図	205
18.3.3.2	動作原理	206
18.3.3.3	直交モードの設定方法	208
18.3.4	パルス幅変調モード	209
18.3.4.1	ブロック図	209
18.3.4.2	動作原理	209
18.3.4.3	その他のコンフィギュレーション	211
18.3.4.4	キル機能	212
18.3.4.5	PWMモードの設定方法	212
18.3.5	デッドタイム付きパルス幅変調モード	213
18.3.5.1	ブロック図	213
18.3.5.2	動作原理	213
18.3.5.3	デッドタイム付きPWMモードの設定方法	214
18.3.6	疑似乱数パルス幅変調モード	215
18.3.6.1	ブロック図	215
18.3.6.2	動作原理	215
18.3.6.3	疑似乱数PWMモードの設定方法	216
18.4	TCPWMレジスタ	217
19.	真の乱数生成器	218
19.1	特長	218
19.2	ブロック図	219
19.3	動作モード	219
19.3.1	固定リング発振器	220
19.3.2	ガロアベースのリング発振器	221
19.3.3	フィボナッチベースのリング発振器	222
20.	LCD直接駆動	223
20.1	特長	223
20.2	LCDセグメント駆動の概要	223
20.2.1	駆動モード	224
20.2.1.1	PWM駆動	224
20.2.1.2	デジタル相関	230
20.2.2	駆動モードの推奨使用方法	233
20.2.3	デジタルコントラストコントロール	233
20.3	ブロック図	234
20.3.1	動作方法	235

20.3.2	高速および低速のマスター信号発生器	235
20.3.3	マルチプレクサおよびLCDピン回路	235
20.3.4	ディスプレイ データ レジスタ	235
20.4	レジスター一覧	236
Section E: アナログシステム		237
21. SAR ADC		238
21.1	特長	238
21.2	ブロック図	239
21.3	動作方法	240
21.3.1	SAR ADCコア	240
21.3.1.1	シングルエンドと差動モード	240
21.3.1.2	入力範囲	240
21.3.1.3	結果データのフォーマット	240
21.3.1.4	反転入力選択	241
21.3.1.5	分解能	242
21.3.1.6	取得時間	242
21.3.1.7	SAR ADCクロック	242
21.3.1.8	SAR ADCタイミング	243
21.3.2	SARMUX	243
21.3.2.1	アナログ配線	244
21.3.2.2	アナログ相互接続	244
21.3.3	SARREF	250
21.3.3.1	リファレンス電圧のオプション	250
21.3.3.2	バイパスコンデンサ	250
21.3.3.3	入力範囲とリファレンス電圧	251
21.3.4	SARSEQ	251
21.3.4.1	平均化	252
21.3.4.2	範囲検出	252
21.3.4.3	二重バッファ	253
21.3.4.4	注入チャンネル	253
21.3.5	割込み	255
21.3.5.1	スキャン終了割込み(EOS_INTR)	255
21.3.5.2	オーバーフロー割込み	255
21.3.5.3	衝突割込み	255
21.3.5.4	注入変換の終了の割り込み(INJ_EOC_INTR)	255
21.3.5.5	範囲検出の割込み	256
21.3.5.6	飽和検出の割込み	256
21.3.5.7	割込み要因のあらまし	256
21.3.6	トリガー	256
21.3.6.1	外部トリガー コンフィギュレーション	258
21.3.7	SAR ADCステータス	258
21.3.8	低消費電力モード	259
21.3.9	システム動作	259
21.3.9.1	SARMUXアナログ配線	260
21.3.9.2	グローバルSARSEQコンフィギュレーション	261
21.3.9.3	チャンネル コンフィギュレーション	261
21.3.9.4	チャンネル イネーブル	262
21.3.9.5	割込みマスク	262
21.3.9.6	トリガー	262
21.3.9.7	各割込みの終了後、データを取り込みます	262
21.3.9.8	注入変換	263

21.3.10 温度センサー コンフィギュレーション	263
21.4 レジスタ	264
22. 低消費電力コンパレータ	266
22.1 特長	266
22.2 ブロック図	266
22.3 動作原理	268
22.3.1 入力コンフィギュレーション	268
22.3.2 出力および割込みコンフィギュレーション	268
22.3.3 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	269
22.3.4 ヒステリシス	270
22.3.5 低消費電力モードからの復帰	271
22.3.6 コンパレータ クロック	271
22.3.7 オフセット調整	271
22.4 レジスタの概要	272
23. 連続時間ミニブロック(CTBm)	273
23.1 特長	273
23.2 ブロック図	274
23.3 動作原理	274
23.3.1 電力モードのコンフィギュレーション	275
23.3.2 出力駆動強度コンフィギュレーション	275
23.3.3 位相補正	276
23.3.4 スイッチ制御	277
23.3.4.1 入力コンフィギュレーション	277
23.3.4.2 出力コンフィギュレーション	278
23.3.4.3 コンパレータ モード	279
23.3.4.4 コンパレータ コンフィギュレーション	279
23.3.4.5 コンパレータ 割込み	280
23.3.4.6 ディープスリープ モード動作	280
23.4 レジスタ概要	282
24. CapSense	283
25. 温度センサー	284
25.1 特長	284
25.2 動作方法	284
25.3 温度センサー コンフィギュレーション	286
25.4 アルゴリズム	286
25.5 レジスタ	286
Section F: プログラムとデバッグ	287
26. デバッグ インターフェース	288
26.1 特長	288
26.2 機能説明	288
26.3 シリアル ワイヤ デバッグ(SWD)インターフェース	289
26.3.1 SWDタイミングの詳細	290
26.3.2 ACK応答の詳細	290
26.3.3 ターンアラウンド(Trn) 期間の詳細	291
26.4 Cortex-M0+デバッグおよびアクセスポート(DAP)	292

26.4.1	デバッグ ポート(DP)レジスタ	292
26.4.2	アクセスポート(AP)レジスタ	292
26.5	PSoC 4デバイスのプログラム	293
26.5.1	SWDポートの開通	293
26.5.1.1	SWDポートの開通シーケンス	293
26.5.2	SWDプログラム モードへの移行	293
26.5.3	SWDプログラム ルーチンの実施	293
26.6	PSoC4 SWDデバッグインターフェース	294
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	294
26.6.2	ブレークポイント ユニット(BPU)	294
26.6.3	データ ウォッチポイント(DWT)	294
26.6.4	PSoC 4デバイスのデバッグ	295
26.7	レジスタ	295
27.	不揮発性メモリ プログラム	296
27.1	特長	296
27.2	機能説明	296
27.3	システム コールの搭載	297
27.4	ブロッキングと非ブロッキングのシステム コール	297
27.4.1	システム コールの実行	297
27.5	システム コール	298
27.5.1	シリコンID	299
27.5.2	クロック設定	299
27.5.3	ロード フラッシュ バイト	300
27.5.4	列書き込み	301
27.5.5	列プログラム	302
27.5.6	全消去	303
27.5.7	チェックサム	304
27.5.8	書き込み保護	305
27.5.9	非ブロッキング列書き込み	306
27.5.10	非ブロッキング列プログラム	307
27.5.11	レジューム非ブロッキング	308
27.6	システム コール ステータス	309
27.7	非ブロッキング システム コール疑似コード	310
用語集		312

Section A: 概要



本セクションは次の章を含みます。

- はじめに (17 ページ)
- 利用の手引き (22 ページ)
- 文書の構成 (23 ページ)

改訂履歴

版数	発行日	変更内容
**	02/23/2017	これは英語版 002-10621 Rev. *A を翻訳した日本語版 002-16642 Rev. ** です。
*A	02/25/2021	これは英語版 002-10621 Rev. *J を翻訳した日本語版 002-16642 Rev. *A です。

1. はじめに



PSoC[®]4 は、Arm[®]Cortex[®]-M0+ CPU を備えたプログラム可能な組み込みシステムコントローラです。PSoC 4100S および PSoC 4100S Plus デバイスは、PSoC 4000 ファミリの拡張版であり、PSoC 4 のより大きなメンバーと上位互換性があります。PSoC 4100S Plus デバイスは、PSoC 4100S デバイスと比較して、より大きなフラッシュメモリ、より多くのペリフェラル、より多くの I/O を提供します。

PSoC 4 デバイスには以下の特長があります：

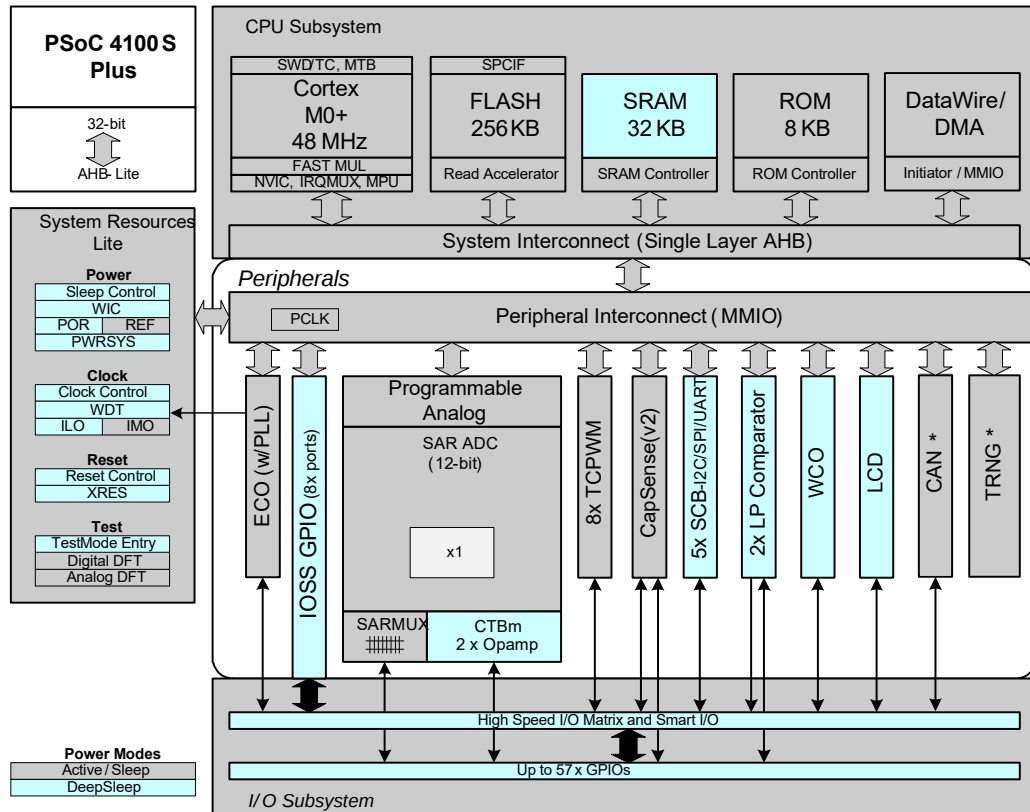
- 高性能、32 ビットのシングル サイクル Cortex-M0+ CPU コア
- 高性能アナログ システム
- 自己および相互静電容量式タッチ センシング (CapSense[®])
- コンフィギュレーション可能なタイマ / カウンタ / PWM ブロック
- アナログ信号調整用のコンフィギュレーション可能アナログ ブロック
- コンフィギュレーション可能な通信ブロック：I²C、SPI および UART 動作モード
- 低消費電力動作モード：スリープ およびディープスリープ

このドキュメントでは、PSoC4100S および PSoC 4100S Plus デバイスの各機能ブロックについて詳しく説明します。この情報は設計者がシステムレベル設計を作成するために役に立ちます。

1.1 トップ レベル アーキテクチャ

Figure 1-1 は、PSoC 4100S Plus アーキテクチャの主要コンポーネントを示します。

Figure 1-1. PSoC 4100S Plus ブロック図



* Available only in PSoC 4100S Plus 128KB

1.2 特長

PSoC 4100S Plus デバイスには、次の主要コンポーネントがあります。

- 最大0.9DMIPS/MHzを供給するシングルサイクル乗算器を備えた 32 ビット Cortex-M0+ CPU
- 最大 256 KB のフラッシュと 32 KB の SRAM
- ダイレクト メモリ アクセス (DMA)
- 暗号化アプリケーションの安全な鍵生成のための真の乱数ジェネレータ
- 最大 8 つの中央揃えのパルス幅変調器 (PWM)、補完的なデッドバンドプログラマブル出力
- 1つのウォッチドッグタイマと3つの汎用タイマ (割り込み機能付き)
- 複数チャンネル用にハードウェア シーケンサを備えた 12 ビット SAR ADC (サンプリング レートは 1Mps)
- アナログ信号調整用およびコンパレータとして使用可能な最大 2 個のオペアンプ
- 2 個の低消費電力コンパレータ
- SPI、UART、I²C、およびローカル相互接続ネットワーク (LIN) スレーブシリアル通信チャネルとして機能できる最大 5 つのシリアル通信ブロック (SCB)
- 1つのコントローラエリアネットワーク (CAN) ブロック
- 1 つの真の乱数ジェネレータ (TRNG) ブロック
- スマート I/O ブロック、I/O 信号パスにブール関数を実行する能力を提供
- CapSense
- セグメント LCD 直接駆動
- 低消費電力動作モード: スリープ およびディープスリープ
- シリアルワイヤ デバッグ (SWD) を介してプログラミングとデバッグ システム
- PSoC Creator™ IDE ツールによる完全サポート

1.3 CPU システム

1.3.1 プロセッサ

PSoCの心臓部は、最大48 MHzで動作する32ビットCortex-M0 + CPU コアです。このプロセッサは広範なクロックゲートを備え、低消費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。この命令セットにより、バイナリコードは完全な上位互換性があり、Cortex M3 や M4 等の上位高性能プロセッサとして使用できます。

CPU は 1 サイクルで 32 ビット結果を出すハードウェア乗算器を備えます。

1.3.2 割り込みコントローラ

CPU サブシステムには、32 個の割り込み入力を備えたネストされたベクタ割り込みコントローラ (NVIC) と、ディープスリープモードからプロセッサをウェイクアップできるウェイクアップ割り込みコントローラ (WIC) があります。

1.3.3 Direct Memory Access、ダイレクトメモリアクセス

DMA エンジンには、プログラム可能な記述子チェーンを使用して、メモリマップ内の任意の場所 (ペリフェラル→ペリフェラルおよびペリフェラル←→メモリ) で独立したデータ転送を行うことができます。

1.3.4 メモリ

1.3.4.1 フラッシュメモリ

PSoC 4 メモリサブシステムには、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 に密結合されたフラッシュアクセラレータを備えたフラッシュモジュールがあります。フラッシュアクセラレータは、シングルサイクルSRAMアクセスパフォーマンスの平均85%を提供します。

1.3.4.2 SRAM

PSoC 4 メモリサブシステムは、デバイスのすべての電力モードで保持されるSRAMを提供します。

1.4 システムワイドリソース

1.4.1 クロック供給システム

クロックシステムは、内部クロックとして内部メイン発振器 (IMO) と内部低速発振器 (ILO) で構成され、外部クロック、外部水晶発振器 (ECO)、および時計用水晶発振器 (WCO) を備えています。

IMO は $\pm 2\%$ の精度を持ち、デバイスの内部クロックの主要供給源です。デフォルトの IMO 周波数は 24MHz で、24MHz から 48MHz の間で 4MHz のステップで調整できます。アプリケーションの要件を満たすために、複数のクロック周波数はメインクロック周波数から生成されます。

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

1 MHz ~ 48 MHz の範囲の外部クロックソースを使用して、IMO の代わりに機能ブロックのクロック派生物を生成できます。

ECO は、外部水晶を使用して、周波数が最大 33 MHz の非常に正確なクロックを生成します。最大 48 MHz の周波数を生成するために使用できるフェーズロックループ (PLL) があります。

WCO は 32kHz 時計用水晶発振器です。これは高精度タイミングのアプリケーション用に、 $\pm 1\%$ の精度に IMO の動的調整に使用されます。

1.4.2 電力システム

このデバイスは、1.71V ~ 5.5V の範囲の単一の外部電源で動作します。複数の電源ドメインを提供します - デジタルセクションに電力を供給する V_{DD} 、およびアナログセクションのノイズ分離用の V_{DDA} 。 V_{DD} および V_{DDA} は外部で短絡する必要があります。

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

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

1.4.3 GPIO

すべての GPIO は以下の特性を持ちます：

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

さらに、デバイスには最大3つのスマート I/O ブロックがあり、ポート I/O でブール関数を実行する機能を提供します。スマート I/O ブロックは、低消費電力モードを含むすべてのデバイス消費電力モードで使用可能です。

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

1.4.4 ウォッチドッグタイマ

PSoC デバイスは1個の16ビットウォッチドッグタイマを備えています。これは、予期しないファームウェア実行経路や、CPU 機能を損なう電圧低下のイベントでデバイスを自動的にリセットすることができます。

これに加えて、2個の16ビットおよび1個の32ビットアップカウンタタイマは汎用目的に使用できます。

1.5 固定機能デジタルブロック

1.5.1 タイマ / カウンタ / PWM ブロック

タイマ / カウンタ / PWM ブロックは、ユーザがプログラム可能な周期長を持つ最大8つの16ビットカウンタで構成されます。TCPWM ブロックはキャプチャレジスタ、周期レジスタおよび比較レジスタを持ちます。ブロックはコンプリメンタリ、デッドバンドプログラマブル出力をサポートします。これは所定の状態にする強制的な出力を行うキル入力も有ります。ブロックの他の機能には、中央揃え PWM、クロックプリスケール、疑似ランダム PWM および直交デコーダがあります。

1.5.2 シリアル通信ブロック

デバイスは3個のSCBを持ちます。各SCBはI²C、UART、ローカルインターコネクトネットワーク (LIN) スレーブ、または SPI となるシリアル通信インターフェースを実装します。

各SCBの機能を以下に示します：

- 標準的な I²C マルチマスターとスレーブ機能
- Motorola、Texas Instruments、National Semiconductor (MicroWire) のプロトコルと互換性のある標準的な SPI マスターとスレーブの機能

- SmartCard リーダー (ISO7816)、IrDA および LIN プロトコルと互換性のある標準的な UART 送受信機能
- 標準 LIN スレーブは LIN V1.3 および LIN V2.1/2.2 仕様規格に準拠
- 32バイトバッファを利用するEZ機能モードのサポート

1.5.3 コントローラエリアネットワーク (CAN)

CAN 2.0A および 2.0B をサポートする PSoC 4100S Plus デバイスには1つのCANブロックが提供されています。CAN ブロックには、それぞれ独自のメッセージフィルタを備えた16個の受信バッファと、8個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。

1.5.4 真性乱数発生器 (TRNG)

TRNG ブロックは、統計的にランダムな数値を生成します。これは、プロセスを再度実行しても複製できない物理的なランダムバリエーションに基づいています。TRNG ブロックによって生成された乱数は、暗号化アプリケーションでのセキュアキーの生成に使用できます。

1.5.5 LCD セグメント駆動

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

1.6 アナログシステム

1.6.1 SAR ADC

PSoC 4 デバイスは、構成可能な12ビット1-Msps SAR ADC を備えています。ADCは3つの内部リファレンス電圧 (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 ハードウェアシーケンサは用意されており、CPU の介入なしに複数チャンネルをスキャンします。

1.6.2 連続時間ブロック ミニ

連続時間ブロック ミニ (CTBm) 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。PSoC4 デバイスには、そのようなCTBmブロックが1つあります。

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、ト

ランスインピーダンス アンプは外部コンポーネントで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機能します。

1.6.3 低消費電力コンパレータ

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

1.7 特殊周辺機能

1.7.1 CapSense

CapSense の機能は、CapSense シグマ デルタ (CSD) ブロックにより自己容量および相互容量のモードで、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 デバイス機能の要約

Table 1-1 に、PSoC 4100S および PSoC 4100S Plus デバイスの概要を示します。

Table 1-1. PSoC 4100S および PSoC 4100S Plus/256 KB デバイスの概要

特長	PSoC 4100S	PSoC 4100S Plus
最大 CPU 周波数	48MHz	48MHz
フラッシュ メモリ	64 KB	128KB, *256KB
SRAM	8 KB	16KB, *32KB
DMA	—	—
GPIO (最大)	36	57, *54
スマート I/O	2 ポート	3 ポート, *2 ポート
CapSense	利用可能	利用可能
LCD ドライバ	利用可能	利用可能
タイマ、カウンタ、PWM (TCPWM)	5	8
16 ビットタイマ	2	2
32 ビットタイマ	1	1
シリアル通信ブロック (SCB)	3	5
IDAC (CapSense 用)	2	2
オペアンプ (CTBm)	2	2
低消費電力コンパレータ (LPCOMP)	2	2
SAR ADC	12 ビット、1Msps	12 ビット、1Msps
時計用水晶発振器 (WCO)	利用可能	利用可能
外部水晶発振器 (ECO)	—	—
電力モード	アクティブ、スリープおよびディープスリープ	アクティブ、スリープおよびディープスリープ

注 *: Table 1-1 の PSoC 4100S Plus 256KB でのみ使用できます。

1.7.1.1 IDAC とコンパレータ

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

1.8 プログラムとデバッグ

PSoC 4 デバイスはオンチップ SWD インターフェース経由で、デバイスのプログラミングとデバッグの機能をサポートします。PSoC Creator IDE は、完全に統合されたプログラミングとデバッグのサポートを提供します。SWD インターフェース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. 利用の手引き



2.1 対応

PSoC[®]4 製品の無料サポートは、www.cypress.com/psoc4 でオンラインで利用できます。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

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

2.3 開発キット

サイプレス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 オンラインストア ウェブサイト www.cypress.com/cypress-store を参照してください。製品ごとに利用可能なアイテムをご覧いただくため、**Programmable System-on-Chip** をクリックしてください。開発キットは Digi-Key、Avnet、Arrow、Future の各社より入手することも可能です。

2.4 アプリケーションノート

アプリケーションノート「[AN79953 - Getting Started with PSoC 4](#)」を参照してください。PSoC 4 デバイスの機能および迅速に PSoC Creator と PSoC4 開発キットを使用した簡単な PSoC アプリケーションを作成する詳細情報があります。

3. 文書の構成



この文書は次の節を含みます：

- Section B: CPU システム (27 ページ)
- Section C: システムリソースサブシステム (SRSS) (65 ページ)
- Section D: デジタル システム (123 ページ)
- Section E: アナログシステム (237 ページ)
- Section F: プログラムとデバッグ (287 ページ)

3.1 主なセクション

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。

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

3.2 本書の表記法

本書では見出しのフォントの他に、4 つの特徴的なフォントを使用しています。

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

レジスタの規則は、*PSoC 4100S: PSoC 4 Registers TRM* および *PSoC 4100S Plus: PSoC 4 Registers TRM* で詳しく説明されています。

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。

Table 3-1. 測定単位

略語	単位
bps	ビット毎秒
°C	摂氏温度
dB	デシベル
fF	フェムトファラド
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 (K ビット)	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	マイクロファラド
μs	マイクロ秒
μV	マイクロボルト
μVrms	マイクロボルト自乗平均 (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	ピコファラド
pp	ピークツーピーク
ppm	100 万分の 1
SPS	サンプル毎秒
σ	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。

Table 3-2. 略語

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

Table 3-2. 略語 (continued)

略語	定義
EEPROM	電氣的消去書き込み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)
FSR	フルスケール範囲
GPIO	general purpose I/O (汎用 I/O)
HCI	ホストコントローラインターフェース
HFCLK	high-frequency clock (高周波クロック)
HSIOM	high-speed I/O matrix (高速 I/O マトリックス)
I ² C	I ² C(規格)
IDE	統合開発環境
ILO	内部低速発振器
ITO	indium tin oxide (インジウム錫酸化物)
IMO	internal main oscillator (内部主発振器)
INL	integral nonlinearity (積分非直線性)
I/O	入出力
IOR	I/O read (I/O 読み出し)
IOW	I/O write (I/O 書き込み)
IRES	initial power on reset (初期パワーオン リセット)
IRA	interrupt request acknowledge (割り込み要求認識)
IRQ	interrupt request (割り込み要求)
ISR	interrupt service routine (割り込みサービス ルーチン)
IVR	interrupt vector read (割り込みベクタ読み出し)
LCD	液晶ディスプレイ
LFCLK	low-frequency clock (低周波クロック)
LPCOMP	low-power comparator (低消費電力コンパレータ)
LRb	last received bit (最後に受信したビット)
LRB	last received byte (最後に受信したバイト)
LSb	最下位ビット
LSB	least significant byte (最下位バイト)
LUT	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)
MPU	memory protection unit (メモリ保護ユニット)
MSb	最上位ビット
MSB	most significant byte (最上位バイト)
MSP	main stack pointer (メインスタック ポインタ)
NMI	non-maskable interrupt (マスク不可割り込み)
NVIC	ネスト可能なベクタ割り込みコントローラ

Table 3-2. 略語 (continued)

略語	定義
PC	プログラム カウンタ
PCB	プリント基板
PCH	program counter high (プログラム カウンタ上位バイト)
PCL	program counter low (プログラム カウンタ下位バイト)
PD	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	疑似乱数列
PSoC [®]	プログラマブル システムオンチップ
PSP	process stack pointer (プロセススタック ポインタ)
PSRR	power supply rejection ratio (電源電圧変動除去比)
PSSDC	power system sleep duty cycle (電源システム スリープ デューティ サイクル)
PWM	pulse width modulator (パルス幅変調器)
RAM	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	読み取り専用メモリ
RMS	ルート平均平方
RW	読み出し / 書き込み
SAR	逐次比較レジスタ
SEG	LCD segment signal (LCD セグメント信号)
SC	スイッチド キャパシタ
SCB	serial communication block (シリアル通信ブロック)
SIE	serial interface engine (シリアル インターフェース エンジン)
SIO	special I/O (特殊 I/O)
SE0	single-ended zero (シングルエンド ゼロ)
SNR	signal-to-noise ratio (信号対雑音比)
SOF	フレームの開始
SOI	start of instruction (命令の開始)
SP	stack pointer (スタック ポインタ)
SPD	sequential phase detector (順次位相検出器)
SPI	serial peripheral interconnect (シリアル ペリフェラル インターコネクト)

Table 3-2. 略語 (continued)

略語	定義
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	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)
SYSCCLK	system clock (システム クロック)
SWD	single wire debug (シングル ワイヤ デバッグ)
TC	terminal count (ターミナル カウント)
TCPWM	timer、counter、PWM (タイマ / カウンタ / PWM)
TD	transaction descriptors (トランザクション記述子)
TIA	trans-impedance amplifier (トランスインピーダンス アンプ)
UART	汎用非同期レシーバ / トランスミッタ
UDB	Universal Digital Block (汎用デジタル ブロック)
USB	ユニバーサル シリアル バス
USBIO	USB I/O (USB 入出力)
VTOR	vector table offset register (ベクタ テーブル オフセット レジスタ)
WCO	watch crystal oscillator
WDT	WatchDog Timer (ウォッチドッグ タイマ)
WDR	ウォッチドッグ リセット
XRES	外部リセット
XRES_N	external reset、active low (外部リセット、アクティブ LOW)

Section B: CPU システム

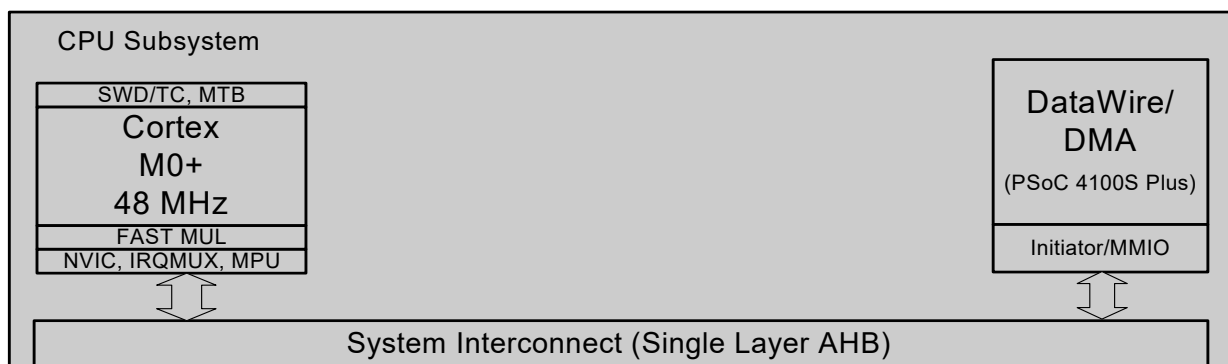


本セクションは次の章を含みます：

- [Cortex-M0+ CPU \(28 ページ\)](#)
- [DMA コントローラ モード \(34 ページ\)](#)
- [割り込み \(51 ページ\)](#)
- [デバイスセキュリティ \(63 ページ\)](#)

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、低電力動作用に最適化された 32 ビット CPU です。高効率の 2 段パイプラインおよび固定 4GB メモリ マップがあり、ARMv6-M Thumb 命令セットに対応します。Cortex-M0+ は 32 ビットのシングル サイクル乗算命令およびレイテンシの短い割り込み処理にも対応します。CPU コアに密接にリンクされるその他のサブシステムはネスト型ベクタ割り込みコントローラ (NVIC)、SYSTICK タイマ、およびデバッグ機能を含みます。

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

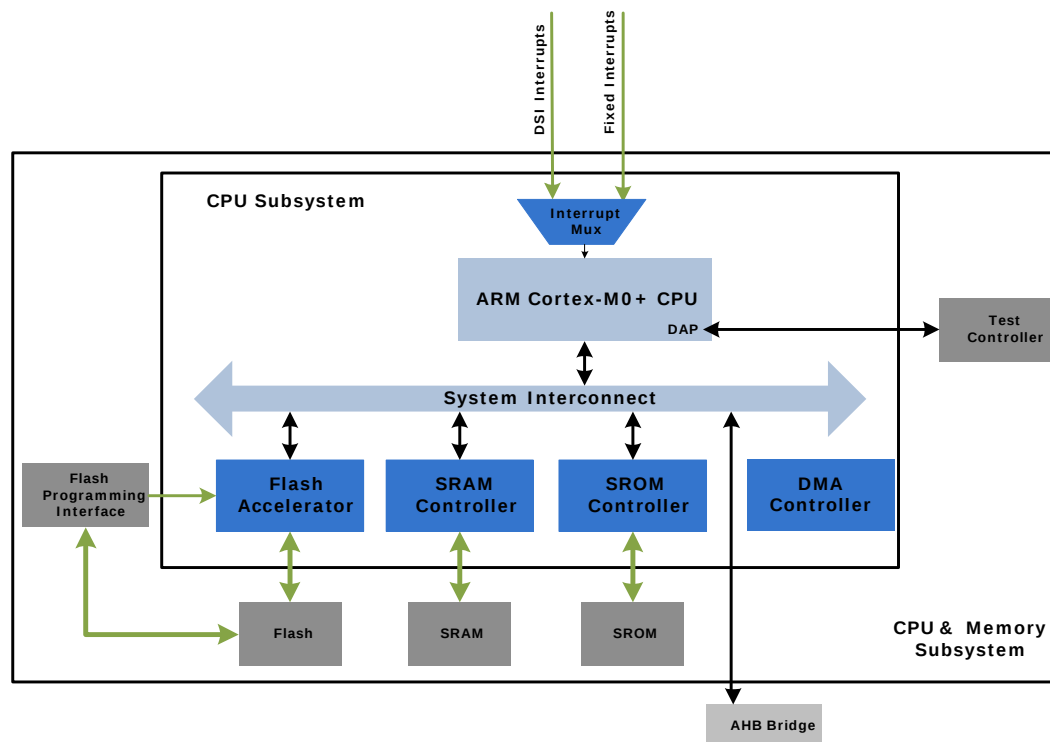
4.1 特長

PSoC 4 Cortex-M0+ は以下の特徴があります：

- プログラムとデバッグが容易であり、8 ビット /16 ビットプロセッサからの移行がより簡単
- 最大 0.9DMIPS/MHz で動作でき、実行速度を向上させ、消費電力を削減
- 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
- 迅速で確定的な割り込み応答を実現するために割り込みと例外をサポートする NVIC ユニット
- 設計時に設定可能なメモリ保護ユニット (MPU) を実装
- 非特権モードおよび特権モードの実行をサポート
- 任意のベクタ テーブル オフセット レジスタ (VTOR) をサポート
- 機能強化されたデバッグ機能：
 - SWD ポート
 - ブレークポイント
 - ウォッチポイント

4.2 ブロック図

Figure 4-1. CPU サブシステム ブロック図



4.3 動作原理

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

プロセッサは 2 つの動作モードをサポートします (“動作モード” (31 ページ) を参照)。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 (4 GB) のアドレス空間は、Table 4-1 に示す領域に分割されています。コードはコード領域および SRAM 領域から実行できることに注意してください。

Table 4-1. Cortex-M0+ アドレスマップ

アドレス範囲	名称	用途
0x00000000 ~ 0x1FFFFFFF	Code (コード)	プログラムコード領域。データも配置可能。アドレス 0 から始まる例外ベクタテーブルを保持。
0x20000000 ~ 0x3FFFFFFF	SRAM	データ領域。この領域からのコードを実行することもできる

Table 4-1. Cortex-M0 + アドレス マップ

アドレス範囲	名称	用途
0x40000000 ~ 0x5FFFFFFF	Peripheral (ペリフェラル)	すべてのペリフェラル レジスタ。この領域からのコードを実行することができない。
0x60000000 ~ 0xDFFFFFFF		未使用。
0xE0000000 ~ 0xE00FFFFF	PPB	CPU コア内のペリフェラル レジスタ
0xE0100000 ~ 0xFFFFFFFF	Device (デバイス)	PSoC 4 固有実装領域。

4.5 レジスタ

Table 4-2 に示すように、Cortex-M0 + には 16 個の 32 ビットレジスタがあります。

- R0 ~ R12: 汎用レジスタ。すべての命令で R0 ~ R7 にアクセスでき、サブセットの命令で他のレジスタにアクセスできます。
- R13: スタック ポインタ (SP)。スタック ポインタは 2 つあり、1 度に 1 つのみ使用可能です。スレッド モードでは、CONTROL レジスタはメイン スタック ポインタ (MSP) またはプロセス スタック ポインタ (PSP)、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 カウンタを保存します。
- R15: プログラム カウンタ。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。

Table 4-2. Cortex-M0+ レジスタ

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

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

Table 4-3 に、PSR ビットの割り当て方法を示します。

Table 4-3. Cortex-M0+ PSR ビット割り当て

ビット	PSR レジスタ	名称	説明
31	APSR	N	負のフラグ
30	APSR	Z	ゼロ フラグ
29	APSR	C	キャリーまたはボロー フラグ
28	APSR	V	オーバーフロー フラグ
27 ~ 25	—	—	予約済み
24	EPSR	T	Thumb 状態のビット。常に 1 でなければなりません。T ビットが 0 のときに命令を実行しようとする、HardFault 例外が発生します。
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 = ハードフォールト 4 ~ 10 = 予約済み 11 = SVCcall 12、13 = 予約済み 14 = PendSV 15 = SysTick 16 = IRQ0 ... 35 = IRQ19

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、設定可能な優先順位の例外、つまり HardFault、NMI、リセットを除くすべての例外は無効になります。例外のリストについては、[割り込み \(51 ページ\)](#) を参照してください。

4.6 動作モード

Cortex-M0+ プロセッサは以下の 2 つの動作モードに対応します：

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

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

ハンドラ モードでは MSP が常に使用されるため、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

Table 4-4 に示すように、Cortex-M0+ は、Thumb 命令セットのバージョンを実装しています。詳細については Cortex-M0+ の全般的なユーザ 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、ARM レジスタを取ります。命令はオペランドに作用し、結果を転送先レジスタに保存します。PC または SP をオペランド、転送先レジスタとして使用することは多くの命令でできないか制限があります。

Table 4-4. Thumb 命令セット

ニーモニック	簡単な説明
ADCS	キャリー付き加算
ADD{S} ^a	Add
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、割込みを無効化
CPSIE	プロセッサ状態を変更、割込みを有効化
DMB	データメモリバリア
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	複数のレジスタをロードし、後にインクリメント
LDR	PC 相対アドレスからレジスタをロード
LDRB	ワード単位でレジスタをロード
LDRH	ハーフワード単位でレジスタをロード
LDRSB	符号付きバイト単位でレジスタをロード
LDRSH	符号付きハーフワード単位でレジスタをロード
LSLS	論理左シフト
LSRS	論理右シフト
MOV{S} ^a	移動
MRS	特殊レジスタから汎用レジスタに移動
MSR	汎用レジスタから特殊レジスタに移動
MULS	乗算、32 ビット出力
MVNS	ビット単位 NOT
NOP	演算なし
ORRS	論理 OR
POP	スタックからレジスタをポップ
PUSH	スタックにレジスタのプッシュ
REV	バイト反転ワード
REV16	各ハーフワードバイト反転
REVSH	バイト反転符号付きハーフワード

Table 4-4. Thumb 命令セット

ニーモニック	簡単な説明
RORS	右ローテート
RSBS	逆減算
SBCS	キャリー付き減算
SEV	イベント送信
STM	複数のレジスタを保存し、後にインクリメント
STR	ワード単位レジスタ保存
STRB	レジスタをバイトとして保存
STRH	レジスタをハーフワードとして保存
SUB{S} ^a	減算
SVC	監視プログラム呼び出し
SXTB	符号拡張バイト
SXTH	符号拡張ハーフワード
TST	論理 AND でテスト
UXTB	1 バイトのゼロ拡張
UXTH	1 ハーフワードのゼロ拡張
WFE	イベントの待機
WFI	割込みの待機

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

4.7.1 アドレス アライメント

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

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

4.7.2 メモリ エンディアン

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

4.8 SysTick タイマ

Systick タイマは NVIC と統合され、SYSTICK 割込みを生成します。この割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 タイマは Cortex-M0+ 内部クロックまたは低周波数クロック (LF_CLK) のいずれかをソースとして使用します。

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)コンパレータおよび2つのウォッチポイント(データ)コンパレータを実装しています。

5. DMA コントローラ モード



PSoC 4100S Plus デバイスでのみ利用可能な DMA コントローラは、DataWire (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 コントローラは以下の特長があります：

- 8 つの DMA チャンネルをサポート
- チャンネルごとに 4 段階の優先度
- バイト、ハーフワード (2 バイト) およびワード (4 バイト) の転送
- チャンネルごとに 3 種類の動作モード
- コンフィギュレーション可能な割り込みの生成
- 転送完了時の出力トリガー
- 最大 65,536 データ要素の転送サイズ

DMA コントローラは 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。これらの動作モードにより、DMA のさまざまな動作シナリオを実装できます。動作モードを下記に示します。

- モード 0: トリガーごとに単一のデータ要素
- モード 1: トリガーごとのすべてのデータ要素
- モード 2: トリガーごとのすべてのデータ要素、および自動的にチェーン記述子をトリガー

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。各チャンネルは独立したディスクリプタ構造体を持ちます。

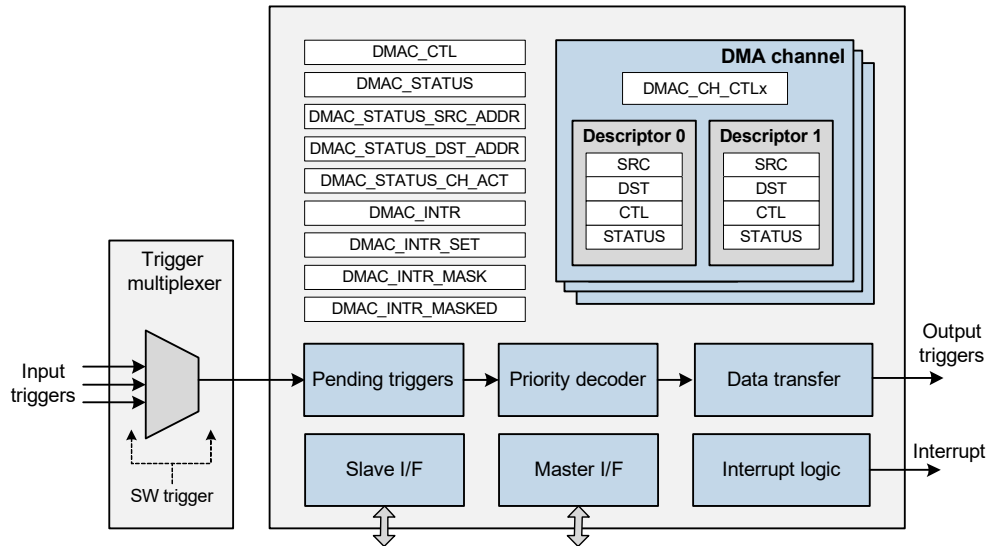
DMA コントローラはアクティブ/スリープ機能を提供し、ディープスリープ電源モードでは使用できません。

5.1 ブロック図の解説

DMA はメモリ、ペリフェラルおよびレジスタとの間でデータを転送します。これらの転送は CPU と独立して行われます。DMA は、1 回の転送で最大 65,536 のデータ要素を転送できます。これらのデータ要素は、8 ビット、16 ビット、または 32 ビット幅にすることができます。DMA は、DMA チャンネル (それ自体を含む)、別の DMA チャンネル、ペリフェラル、または CPU からの外部トリガーを介して各トランザクションを開始します。DMA は、データ転送タスクを CPU からオフロードするために最適に使用されます。

Figure 5-1 に、ブロックレベルでの DMA コントローラの概要を示します。

Figure 5-1. DMA コントローラのブロック図



すべての DMA チャンネルには 2 つの記述子があり、送信元アドレス、宛先アドレス、データ幅など、転送に固有のパラメータを構成します。DMA チャンネルでの転送開始はトリガーイベントです。トリガー信号は、DMA 自体を含め、デバイス内のさまざまな周辺機器から送信されます。

DMA コントローラには、マスターインターフェースとスレーブインターフェースの 2 つのバスインターフェースがあります。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。DMA は、マスターインターフェースのバスマスターです。これは、すべての DMA 転送が実行されるインターフェースです。

DMA 構成レジスタと記述子は、スレーブインターフェースを介してアクセスおよび再構成されます。スレーブ I/F は AHB-Lite バススレーブであり、PSoC メイン CPU が DMA controller's の制御 / ステータスレジスタにアクセスし、記述子構造にアクセスできるようにします。CPU は通常、このバスのマスターです。

トリガーを受信すると、DMA コントローラのステートマシンがアクティブになり、トリガーの優先順位付けと処理が行われ、記述子の設定に従ってデータ転送が開始されます。転送が完了すると、出力トリガーが生成され、トリガー条件または別の関数を開始するためのイベントとして使用できます。

DMA コントローラには、割り込みロジックブロックもあります。CPU に割り込むために DMA コントローラから使用できる割り込み線は 1 つだけです。転送の完了時にこの割り込みラインをアクティブにするように、個々の DMA 記述子を構成できます。

5.1.1 トリガーソースと多重化

すべての DMA チャンネルには、それに関連付けられた入力お

よび出力トリガーがあります。入力トリガーは、任意のペリフェラル、CPU、または DMA チャンネル自体から来てことができます。5.2.4 転送モードで定義されているように、入力トリガーは DMA 転送をトリガーするために使用されます。トリガー入力の 'logic high' は、DMA チャンネルをトリガーします。この 'logic high' の最小幅は 2 システムクロックサイクルです。非アクティブ化設定は、トリガー非アクティブ化の性質を構成します。

出力トリガーは、転送の完了を通知します。この信号は、DMA チャンネルへのトリガーとして、またはデジタル相互接続へのデジタル信号として使用できます。トリガー入力はさまざまなソースから取得でき、5.1.1.1 トリガーマルチプレクサを介してルーティングされます。

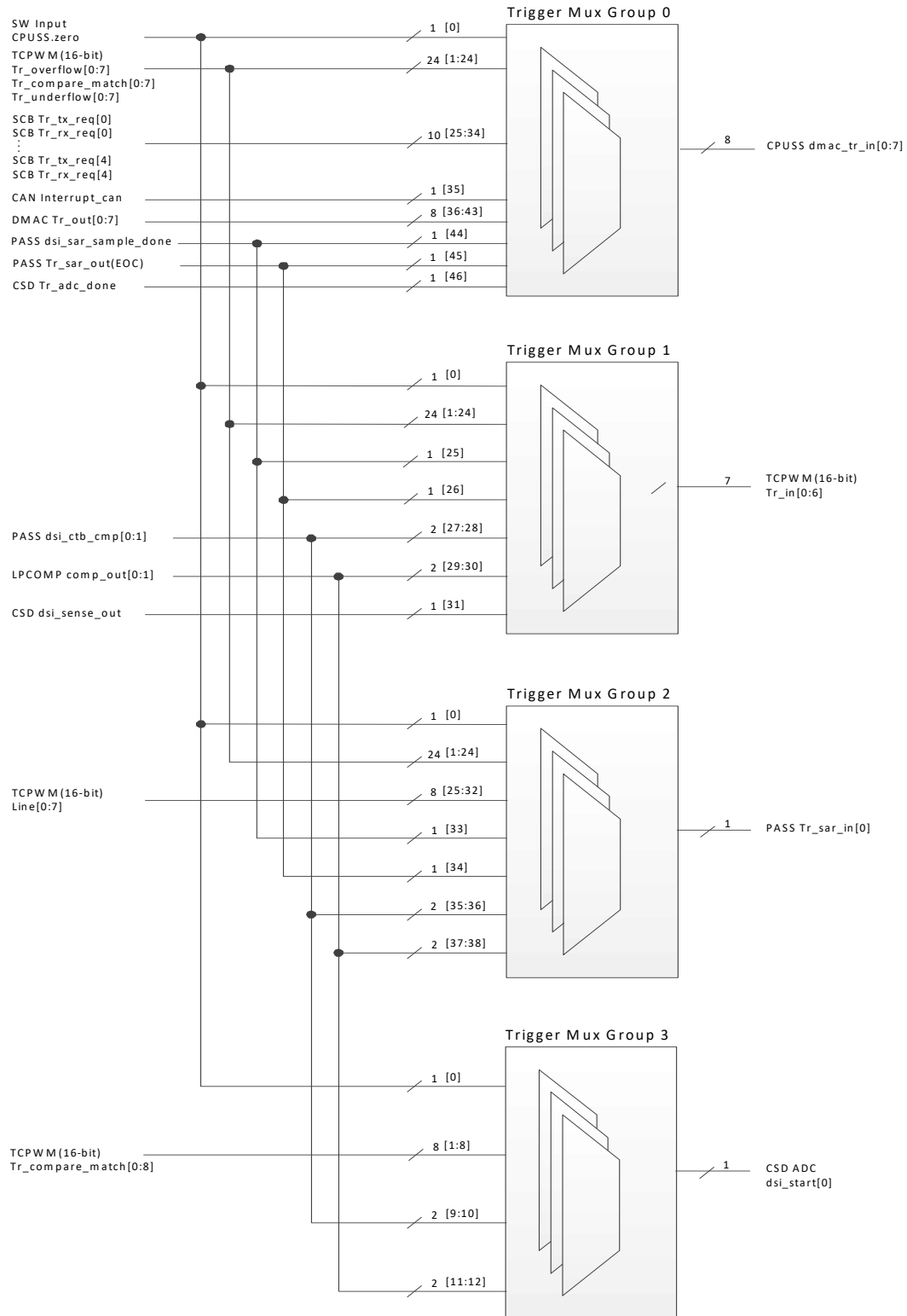
5.1.1.1 トリガーマルチプレクサ

DMA チャンネルは、PSoC のさまざまなペリフェラルソースからのトリガー入力を持つことができます。これは、トリガーマルチプレクサを通じて個々の DMA チャンネルトリガー入力にルーティングされます。

DMA トリガーでは、マルチプレクサはトリガーグループに編成されます。各トリガーグループは、個々の DMA チャンネルトリガー入力に供給する複数のマルチプレクサで構成されています。

PSoC 4100S Plus デバイスは、DMA にトリガー入力を提供する単一のトリガーグループ (トリガーグループ 0) を実装します。PSoC 4100S Plus デバイストリガー入力オプションは、TCPWM、SAR ADC、SCB、CAN、CapSense ADC、および DMA 出力トリガーから取得できます。PSoC 4100S Plus 256 KB デバイストリガー入力オプションは、TCPWM、SCB、EXCO、DMA 出力トリガー、SAR ADC、CapSense ADC、CTBm トリガー、LPComp トリガーから取得できます。Figure 5-2 に、PSoC 4100S Plus 256 KB トリガーマルチプレクサの実装を示します。

Figure 5-2. PSoC 4100S Plus 256 KB トリガーマルチプレクサの実装



個々の DMA チャンネルのトリガーソースは、PERI_TR_GROUP0_TR_OUT_CTLx[5:0] レジスタで選択されます。Table 5-1 に、PSoC 4100S Plus デバイストリガーマルチプレクサを示します。Table 5-2 は、PSoC 4100S Plus 256 KB デバイストリガーマルチプレクサを提供します。

Table 5-1. PSoC 4100S Plus デバイストリガーソース

PERI_TR_GROUP_TR_OUT_CTL x[5:0]	トリガーソース
0	ソフトウェアトリガー
1	TCPWM 0 オーバーフロー
2	TCPWM 1 オーバーフロー
3	TCPWM 2 オーバーフロー
4	TCPWM 3 オーバーフロー
5	TCPWM 4 オーバーフロー
6	TCPWM 5 オーバーフロー
7	TCPWM 6 オーバーフロー
8	TCPWM 7 オーバーフロー
9	TCPWM 0 比較一致
10	TCPWM 1 比較一致
11	TCPWM 2 比較一致
12	TCPWM 3 比較一致
13	TCPWM 4 比較一致
14	TCPWM 5 比較一致
15	TCPWM 6 比較一致
16	TCPWM 7 比較一致
17	TCPWM 0 アンダーフロー
18	TCPWM 1 アンダーフロー
19	TCPWM 2 アンダーフロー
20	TCPWM 3 アンダーフロー
21	TCPWM 4 アンダーフロー
22	TCPWM 5 アンダーフロー
23	TCPWM 6 アンダーフロー
24	TCPWM 7 アンダーフロー
25	SCB 0 TX リクエスト
26	SCB 0 RX リクエスト
27	SCB 1 TX リクエスト
28	SCB 1 RX リクエスト
29	SCB 2 TX リクエスト
30	SCB 2 RX リクエスト
31	SCB 3 TX リクエスト
32	SCB 3 RX リクエスト
33	SCB 4 TX リクエスト
34	SCB 4 RX リクエスト

Table 5-1. PSoC 4100S Plus デバイストリガーソース

PERI_TR_GROUP_TR_OUT_CTL x[5:0]	トリガーソース
35	CAN 割り込み
36	DMA チャンネル 0 トリガーアウト
37	DMA チャンネル 1 トリガーアウト
38	DMA チャンネル 2 トリガーアウト
39	DMA チャンネル 3 トリガーアウト
40	DMA チャンネル 4 トリガーアウト
41	DMA チャンネル 5 トリガーアウト
42	DMA チャンネル 6 トリガーアウト
43	DMA チャンネル 7 トリガーアウト
44	完了した SAR ADC サンプル
45	SAR 変換終了
46	CSD ADC サンプルの完了

Table 5-2. PSoC 4100S Plus 256 KB デバイストリガーソース

PERI_TR_GROUP_TR_OUT_CTL x[5:0]	トリガーソース
0	ソフトウェアトリガー
1	TCPWM 0 オーバーフロー
2	TCPWM 1 オーバーフロー
3	TCPWM 2 オーバーフロー
4	TCPWM 3 オーバーフロー
5	TCPWM 4 オーバーフロー
6	TCPWM 5 オーバーフロー
7	TCPWM 6 オーバーフロー
8	TCPWM 7 オーバーフロー
9	TCPWM 0 比較一致
10	TCPWM 1 比較一致
11	TCPWM 2 比較一致
12	TCPWM 3 比較一致
13	TCPWM 4 比較一致
14	TCPWM 5 比較一致
15	TCPWM 6 比較一致
16	TCPWM 7 比較一致
17	TCPWM 0 アンダーフロー

Table 5-2. PSoC 4100S Plus 256 KB デバイストリガースース

PERI_TR_GROUP_TR_OUT_CTL x[5:0]	トリガースース
18	TCPWM 1 アンダーフロー
19	TCPWM 2 アンダーフロー
20	TCPWM 3 アンダーフロー
21	TCPWM 4 アンダーフロー
22	TCPWM 5 アンダーフロー
23	TCPWM 6 アンダーフロー
24	TCPWM 7 アンダーフロー
25	SCB 0 TX リクエスト
26	SCB 0 RX リクエスト
27	SCB 1 TX リクエスト
28	SCB 1 RX リクエスト
29	SCB 2 TX リクエスト
30	SCB 2 RX リクエスト
31	SCB 3 TX リクエスト
32	SCB 3 RX リクエスト
33	SCB 4 TX リクエスト
34	SCB 4 RX リクエスト
35	EXCO trigger
36	DMA チャンネル 0 トリガーアウト
37	DMA チャンネル 1 トリガーアウト
38	DMA チャンネル 2 トリガーアウト
39	DMA チャンネル 3 トリガーアウト
40	DMA チャンネル 4 トリガーアウト
41	DMA チャンネル 5 トリガーアウト
42	DMA チャンネル 6 トリガーアウト
43	DMA チャンネル 7 トリガーアウト
44	SAR ADC0 サンプル完了
45	SAR ADC0 変換終了
46	CSD ADC サンプルの完了
47	CTBm [0] cmp0
48	CTBm [0] comp1
49	SAR ADC1 サンプル完了
50	SAR ADC1 変換終了

Table 5-2. PSoC 4100S Plus 256 KB デバイストリガースース

PERI_TR_GROUP_TR_OUT_CTL x[5:0]	トリガースース
51	CTBm [1] cmp0
52	CTBm [1] comp1
53	LPCOMP[0] 出力
54	LPCOMP[1] 出力

5.1.1.2 ソフトウェアトリガーの作成

すべての DMA チャンネルには、トリガー入力と出力トリガーが関連付けられています。このトリガー入力は、“[トリガーマルチプレクサ](#)” (35 ページ) で説明されているように、どのトリガーグループからでも入力できます。DMA チャンネルのソフトウェアトリガーは、トリガーマルチプレクサ設定のトリガー入力オプション 0 を使用して実装されます。PERI_TR_GROUP_TR_OUT_CTLx [5:0] がゼロの場合、DMA トリガーはソフトウェアトリガー用に構成されます。DMA チャンネルは、PERI_TR_CTL レジスタを使用してトリガーされます。

5.1.2 保留中のトリガー

DMA チャンネルがすでに動作していて、トリガーイベントが発生すると、トリガーに対応する DMA チャンネルは保留状態になります。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。複数のチャンネルトリガーを同時にアクティブ化できるため、これは不可欠です。一方、データ転送エンジンが 1 度に処理できるチャンネルは 1 つだけです。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。

5.1.3 出力トリガー

各チャンネルには出力トリガーがあります。このトリガーは、2 つのシステムクロックサイクルの間ハイです。トリガーは、データ転送の完了時に生成されます。システムレベルでは、これらの出力トリガーをトリガーマルチプレクサコンポーネントに接続できます。この接続により、DMA コントローラの出力トリガーを DMA コントローラの入力トリガーに接続できます。つまり、あるチャンネルの転送完了が、別のチャンネルをアクティブにしたり、再び同じチャンネルをアクティブにする場合があります。

5.1.4 チャンネルの優先順位付け

アクティブなトリガーを持つチャンネルが複数ある場合、チャンネルの優先度を使用して、データ転送エンジンにアクセスするチャンネルを決定します。優先度は、チャンネル制御レジスタ (DMAC_CH_CTL) の PRIO フィールドを使用して各チャネ

ルに設定されます。「0」は最高の優先度を表し、「3」は最低の優先度を表します。優先度のデコードでは、チャンネルの優先度を使用して、最も優先度の高いアクティブ化されたチャンネルを決定します。複数のアクティブ化されたチャンネルが同じ最高の優先順位を持っている場合、最小のインデックス「i」を持つチャンネルは、最高の優先順位のアクティブ化されたチャンネルと見なされます。

5.1.5 データ転送エンジン

データ転送エンジンは、ソースの場所から宛先の場所へのデータ転送を担当します。アイドルのとき、データ転送エンジンは最高の優先度で起動されたチャン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。データ転送エンジンは、次の状態を持つステートマシンを実装します。

- 状態0-デフォルト状態:これはDMAコントローラのアイドル状態であり、トリガー条件が転送を開始するのを待ちます。
- 状態1-ロード記述子:トリガー条件が発生し、優先順位が解決されると、データ転送エンジンはロード記述子状態に入ります。この状態では、アクティブな記述子(SRC、DST、およびCTL)がDMAコントローラにロードされ、転送が開始されます。DMAC_STATUS、DMAC_STATUS_SRC_ADDR、DMAC_STATUS_DST_ADDR、およびSTATUS_CH_ACTにも、現在アクティブなステータスが反映されます。
- 状態2-ソースからのデータの読み込み:データ転送エンジンはマスター I/F を使用して、ソースの場所からデータを読み込みます。
- 状態3-宛先でのデータの格納:データ転送エンジンはマスター I/F を使用して、データを宛先の場所に格納します。
転送モードによっては、状態2および3が複数回実行される場合があります。
- 状態4-記述子の格納:データ転送エンジンは、channel's 記述子構造を更新してデータ転送を反映し、記述子に格納します。
- 状態5-トリガー非アクティブ化の待機:トリガー非アクティブ化条件が2サイクルとして指定されている場合、この条件は、トリガーアクティブ化の2サイクル後に満たされます。「wait indefinitely (無期限に待機)」に設定されている場合、DMAコントローラは、トリガー信号がLowになるまでこの状態を維持します。
- 状態6-記述子応答の格納:このフェーズでは、記述子に基づくデータ転送が完了し、そのように構成されている場合は割り込みが生成される可能性があります。DMAC_DESCR_PING_STATUS または DMAC_DESCR_PONG_STATUS の Response フィー

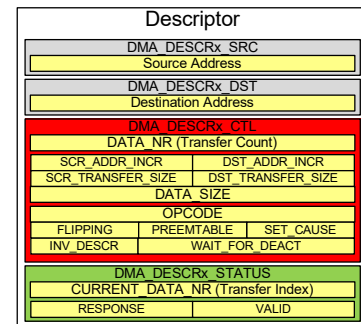
ルドにもデータが入力され、状態は状態0に遷移します。

5.2 ディスクリプタ

チャンネル内のソースと宛先間のデータ転送は、記述子を使用して構成されます。DMA の各チャンネルには、PING および PONG 記述子 (このドキュメントでは記述子0 および記述子1とも呼ばれます) という名前の2つの記述子があります。ディスクリプタは関連チャン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。

Figure 5-3 は、記述子の構造を示します。

Figure 5-3. ディスクリプタ構造体



- Status Registers
- Control Registers
- Address Registers

5.2.1 アドレス コンフィギュレーション

Figure 5-4 は、転送のアドレス構成に対する記述子設定の使用を示します。

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、ソースアドレスレジスタはベ

スアドレスのみを設定し、その後の転送はこれで増分されます。アドレス増分のサイズは、[5.2.2 転送サイズ \(42 ページ\)](#) で説明されている SCR_TRANSFER_SIZE 設定に基づいて決定されます。

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。アドレス増分のサイズは、[5.2.2 転送サイズ \(42 ページ\)](#) で説明されている DST_TRANSFER_SIZE 設定に基づいて決定されます。

記述子の無効化 (INV_DESCR): このビットが設定されている場合、記述子はすべてのデータ要素を転送し、descriptor's VALID ビットをクリアして無効にします。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。この設定は、転送の完了後に記述子が無効になることをユーザが求めている場合に使用されます。記述子は、記述子の STATUS レジスタの VALID ビットを設定することにより、ファームウェアで再度有効にすることができます。

プリエンタブル (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 チャンネルにより先取り、割り込みをされます。このチャン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 チャンネル間に共用されます。このビットを設定すると、対応するチャンネルをこの割り込みのソースにすることができます。

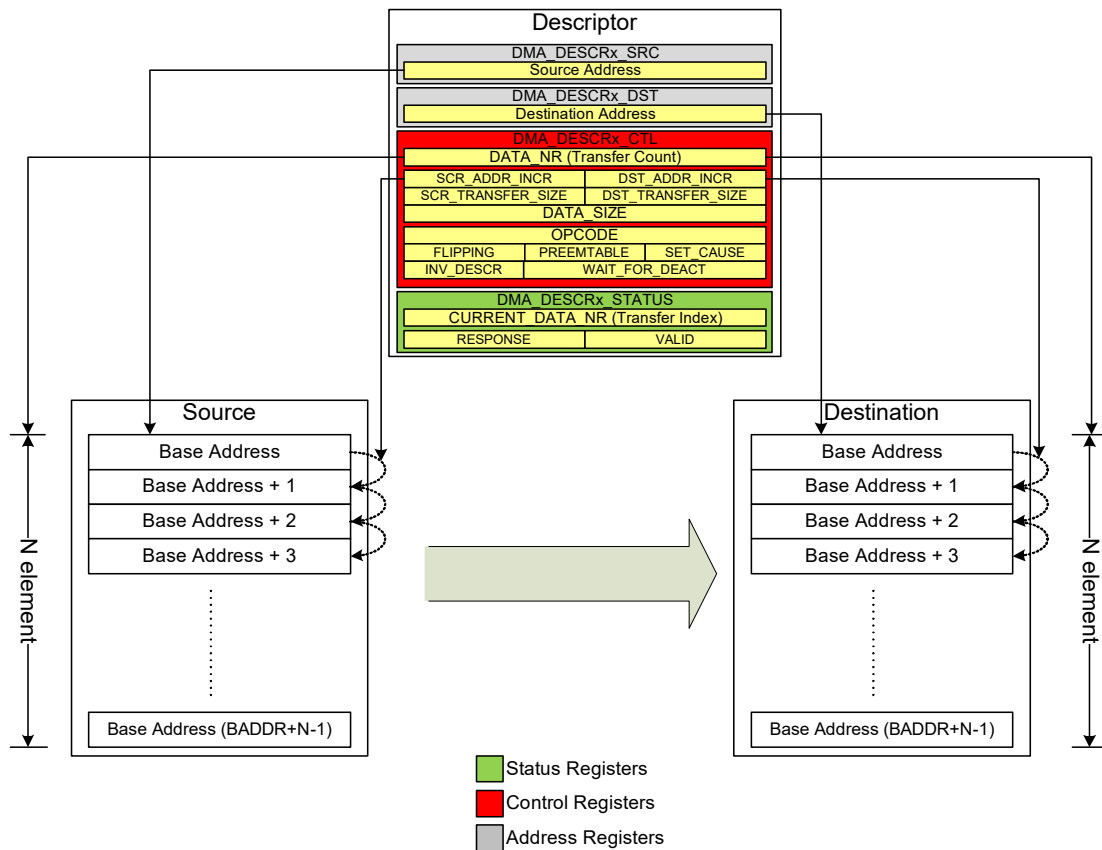
トリガータイプ (WAIT_FOR_DEACT): ディスクリプタに基づく DMA 転送が完了すると、データ転送エンジンはトリガーの非アクティブ化の状態をチェックします。これは、データ転送エンジンの状態 5 に対応しています。[5.2.2 転送サイズ \(42 ページ\)](#) を参照してください。DMA 入力トリガーのタイプにより、トリガー信号が非アクティブと見なされるタイミングが決まります。DMA 転送は、トリガーがアクティブになるとアクティブになりますが、トリガー状態が非アクティブになるまで、転送は完了したと見なされません。このフィールドは、controller's のデータ転送を、トリガーを生成したエージェントと同期するために使用されます。

このフィールドは、記述子の実行の完了時にのみ使用され、4 つの設定があります。

- 0- パルストリガー：非アクティブ化を待機しません。
- 1- レベル依存は 4 SYSCLK サイクル待機します：レベルトリガー信号が 4 サイクル検出された後、DMA トリガーは非アクティブ化されます。

- 2- レベル依存は 8 SYSCLK サイクル待機します。DMA 転送は、レベルトリガー信号が 8 サイクル検出された後に開始されます。
- 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。

Figure 5-4. 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。データ サイズ パラメータ (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 によって設定されるソースおよび宛先の転送サイズは、DATA_SIZE または 32 ビットのいずれかの値を持てます。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。

ほとんどの PSoC 4 周辺レジスタのデータ幅は 4 バイト (32 ビット) です。したがって、DMA がソースまたは宛先としてペリフェラルを使用している場合、SCR_TRANSFER_SIZE または DST_TRANSFER_SIZE は通常 32 ビットに設定する必要があります。DMA コンポーネントのソースと宛先の転送サイズは、移動する必要がある

データの幅に関係なく、ソースと宛先のアドレス可能な幅と一致する必要があります。DATA_SIZE パラメータは、実際のデータの幅に対応します。たとえば、16 ビット PWM が DMA データの宛先として使用される場合、TCPWM ブロック (およびほとんどの PSoC 4 ペリフェラルの周辺レジスタの幅) のため、DST_TRANSFER_SIZE は PWM レジスタの幅と一致するように 32 ビットに設定する必要があります) は常に 32 ビットです。ただし、この例では、16 ビット PWM は 2 バイトのデータしか使用しないため、宛先の DATA_SIZE は 16 ビットに設定されている可能性があります。SRAM とフラッシュは 8 ビット、16 ビット、または 32 ビットのアドレス指定が可能で、アプリケーションのニーズに合わせて任意のソースおよび宛先転送サイズを使用できます。

Table 5-3 に、転送サイズ設定の可能な組み合わせとその説明をまとめました。

Table 5-3. 転送サイズの設定

DATA_SIZE	SCR_TRANSFER_SIZE	DST_TRANSFER_SIZE	典型的な使用法	説明
8 ビット	8 ビット	8 ビット	メモリ→メモリ	データの操作なし
8 ビット	32 ビット	8 ビット	周辺機器→メモリ	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	メモリ→周辺機器	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	周辺機器→周辺機器	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	メモリ→メモリ	データの操作なし
16 ビット	32 ビット	16 ビット	周辺機器→メモリ	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	メモリ→周辺機器	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	周辺機器→周辺機器	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	周辺機器→周辺機器	データの操作なし

5.2.3 記述子の連鎖

全てのチャンネルに、転送設定を有する PING と PONG 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、個別のチャンネル制御レジスタ (DMAC_CH_CTL) の PING_PONG ビットに示されます。PING と PONG 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 PONG レジスタがアクティブのときに PING 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 FLIPPING ビットが有効になると、次に PING/PONG カウンタ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、OPCODE 2 転送モードと組み合わせて使用されます。従って、FLIPPING ビットが PING ディ

スクリプタ内で有効にされ OPCODE 2 用に設定された場合、チャンネルは PING ディスクリプタの次に PONG ディスクリプタを自動的に実行します。OPCODE 0 または OPCODE 1 のコンフィギュレーションの場合、PONG 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。

PING PONG の使用は、転送モードのコンテキストでより関連性が高くなります。

5.2.4 転送モード

ディスクリプタ実行中のチャンネルの動作はOPCODEの設定により定義されます。DMA コントローラの各チャンネルで使用可能な OPCODE は 3 つあります。

5.2.4.1 トリガーごとに単一のデータ要素 (OPCODE 0)

このモードは、OPCODE が 0 に設定されている場合に実現されます。DMA は、各トリガー信号で単一のデータ要素をソース位置から宛先位置に転送します。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。

Figure 5-5 に、この転送の一般的な使用例を示します。ここでは、UART 受信 (RX) レジスタがソースで、宛先は SPI 送信 (TX) レジスタなどのペリフェラルレジスタです。トリガーは、UART の DMA 要求信号からのものです。トリガーが受信されると、転送エンジンは UART RX レジスタからデータをロードし、下位 8 ビットを SPI TX レジスタに格納します。記述子が再実行されるため、トリガーが連続しても同じ動作です。

ソースとデスティネーションのデータ幅が 32 ビットとして割り当てられていることに注意してください。これは、PSoC のペリフェラルレジスタへのすべてのアクセスが 32 ビットでなければならないためです。有効なデータ幅は 8 ビットのみであるため、DATA_SIZE は 8 ビットとして維持されます。

Figure 5-5. OPCODE 0: ペリフェラルからペリフェラルへの簡単な DMA 転送

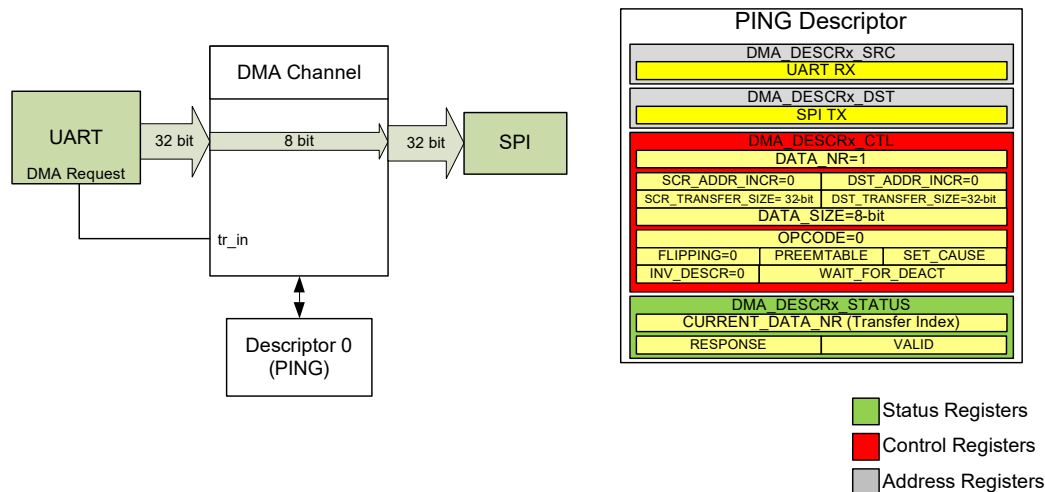
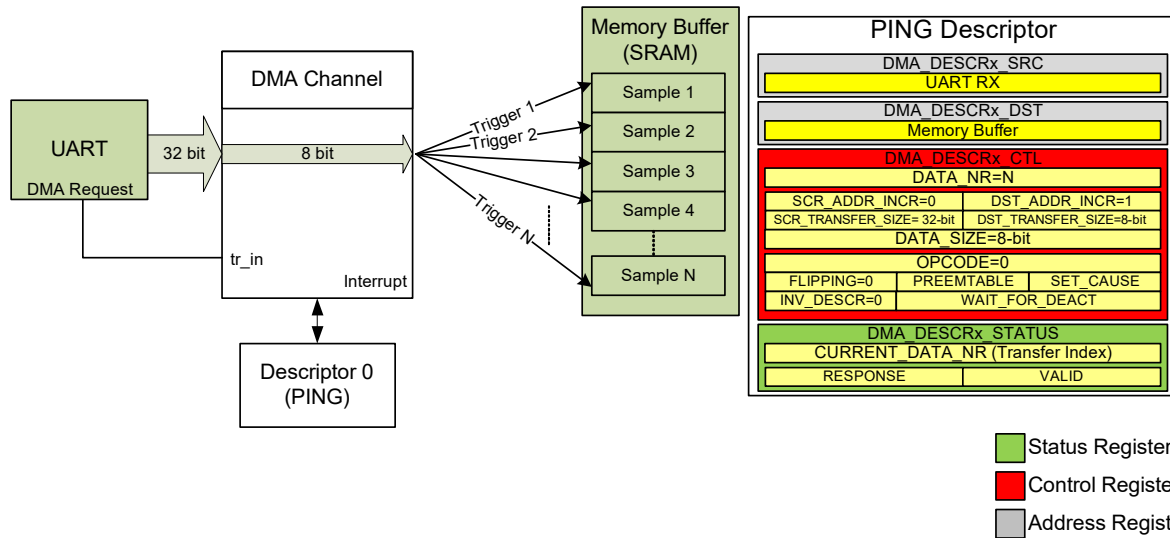


Figure 5-6 は、データ転送が UART RX レジスタとバッファの間で行われる別の使用例を示します。ユースケースは PING 記述子を示します。PING 記述子は、UART であるソースの場所からデータを取得しながら宛先をインクリメントするように構成されています。トリガーが受信されると、転送エンジンは UART RX レジスタからデータをロードし、メモリバッファ、サンプル 1 のメモリ位置に格納します。後続のトリガーは、PING 記述子バッファサイズ (DATA_NR フィールド) がいっぱいになるまで、サンプル 1 からの連続した場所に UART データを格納し続けます。

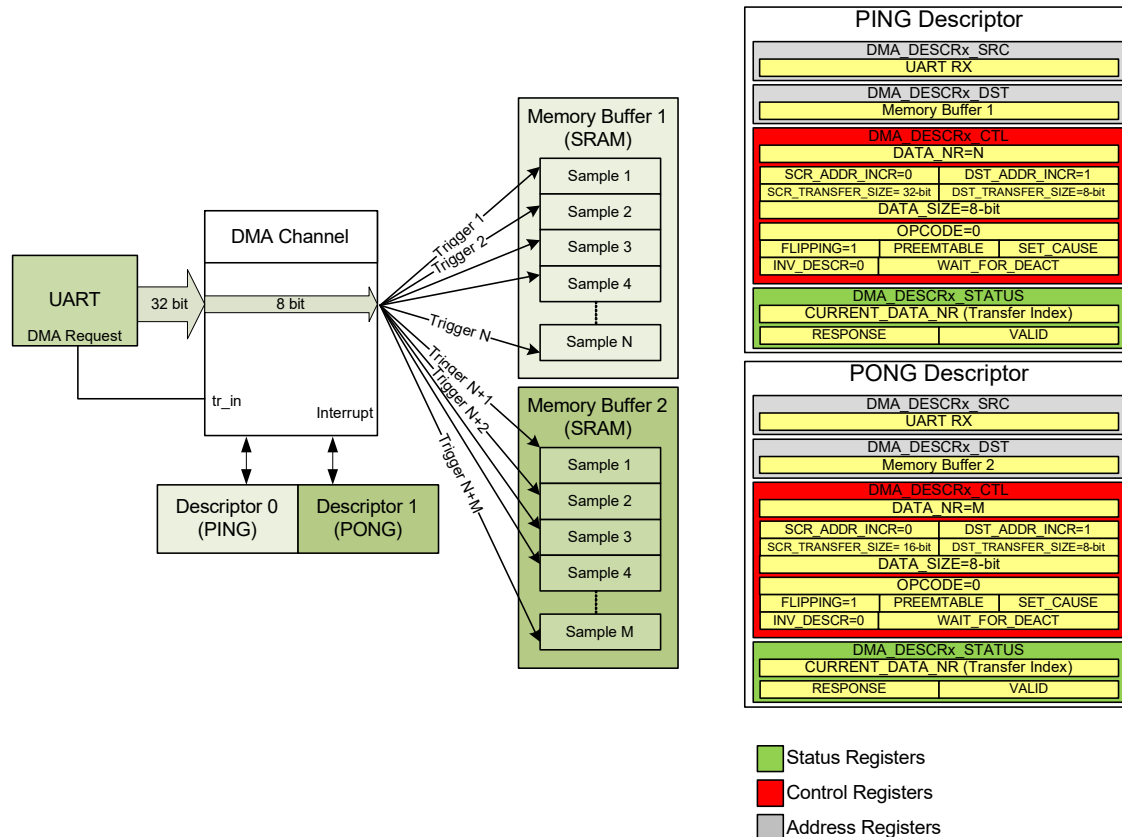
Figure 5-6. OPCODE 0: 宛先アドレス増分機能を使用した転送



同様の使用例を Figure 5-7 に示します。これは PING と PONG のディスクリプタの使用例です。PING 記述子が完了すると、コントローラはフリップして PONG 記述子を実行

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。

Figure 5-7. フリップping機能を使用した DMA 転送

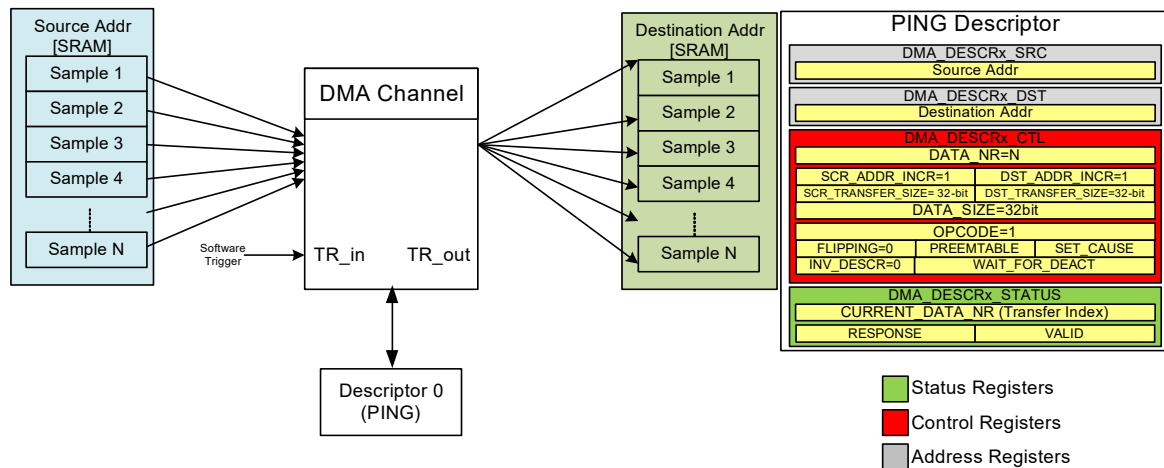


5.2.4.2 トリガーごとの記述子全体 (OPCODE 1)

この動作モードでは、DMA は 1 つのトリガーで複数のデータ要素をソースの場所から宛先の場所に転送します。OPCODE 1 の場合、コントローラ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パッファ転送に役に立ちます。トリガー条件が発生すると、記述子が完了するまで転送が続行されます。

Figure 5-8 は、ソースバッファの内容全体を宛先バッファに転送する OPCODE 1 転送を示します。全部の転送は単一の PING 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。

Figure 5-8. OPCODE 1 の DMA 転送の例

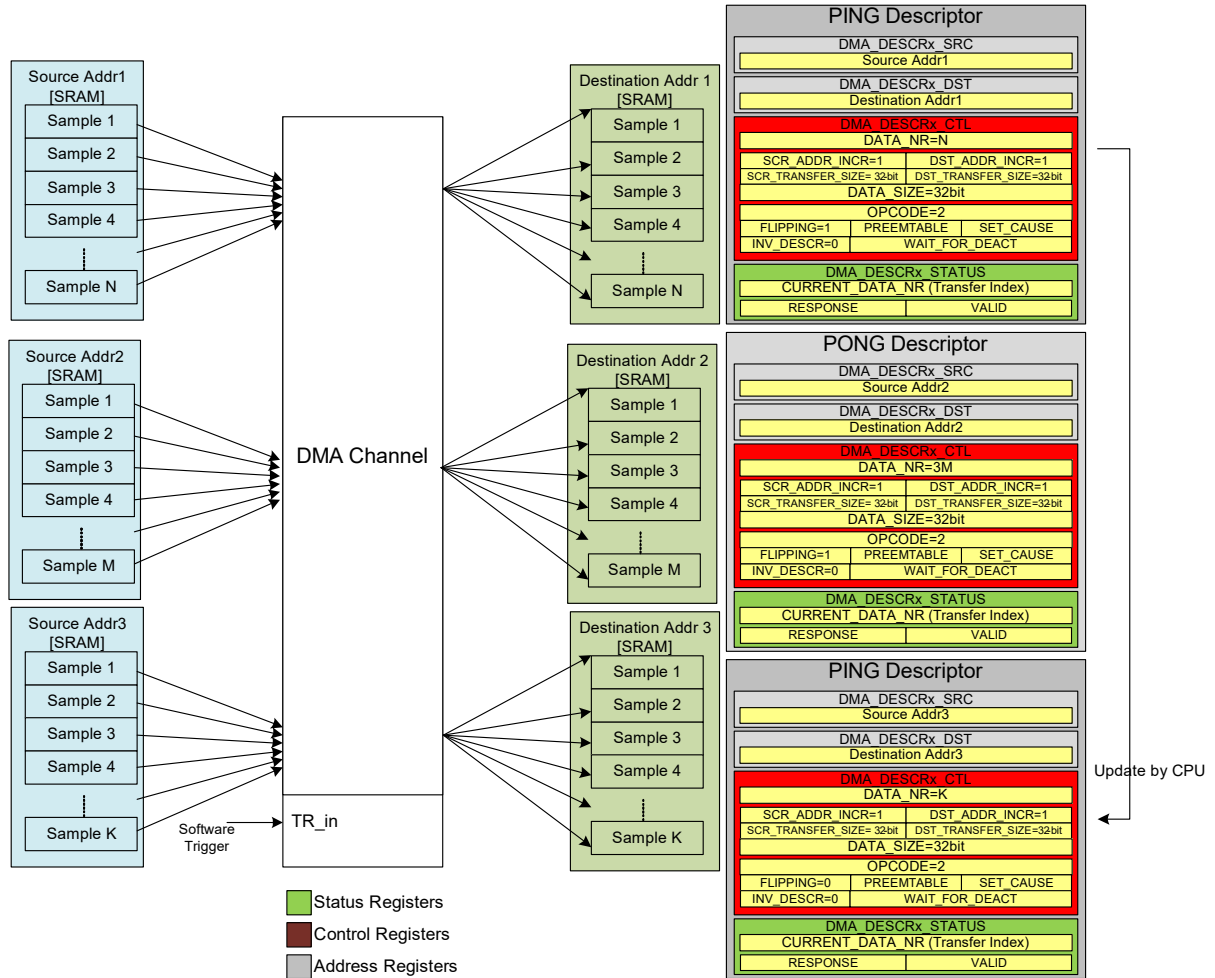


5.2.4.3 トリガーごとの記述子チェーン全体 (OPCODE 2)

OPCODE 2 は常に FLIPPING フィールドと組み合わせて使用されます。PING 記述子で FLIPPING を有効にして OPCODE 2 を使用すると、単一のトリガーで PING 記述子を実行し、自動的に PONG 記述子に切り替えて実行することもできます。PONG 記述子にも OPCODE 2 が提供されている場合、PING と PONG の間の循環は、記述子の 1 つが無効になるか、CPU によって変更されるまで続きます。

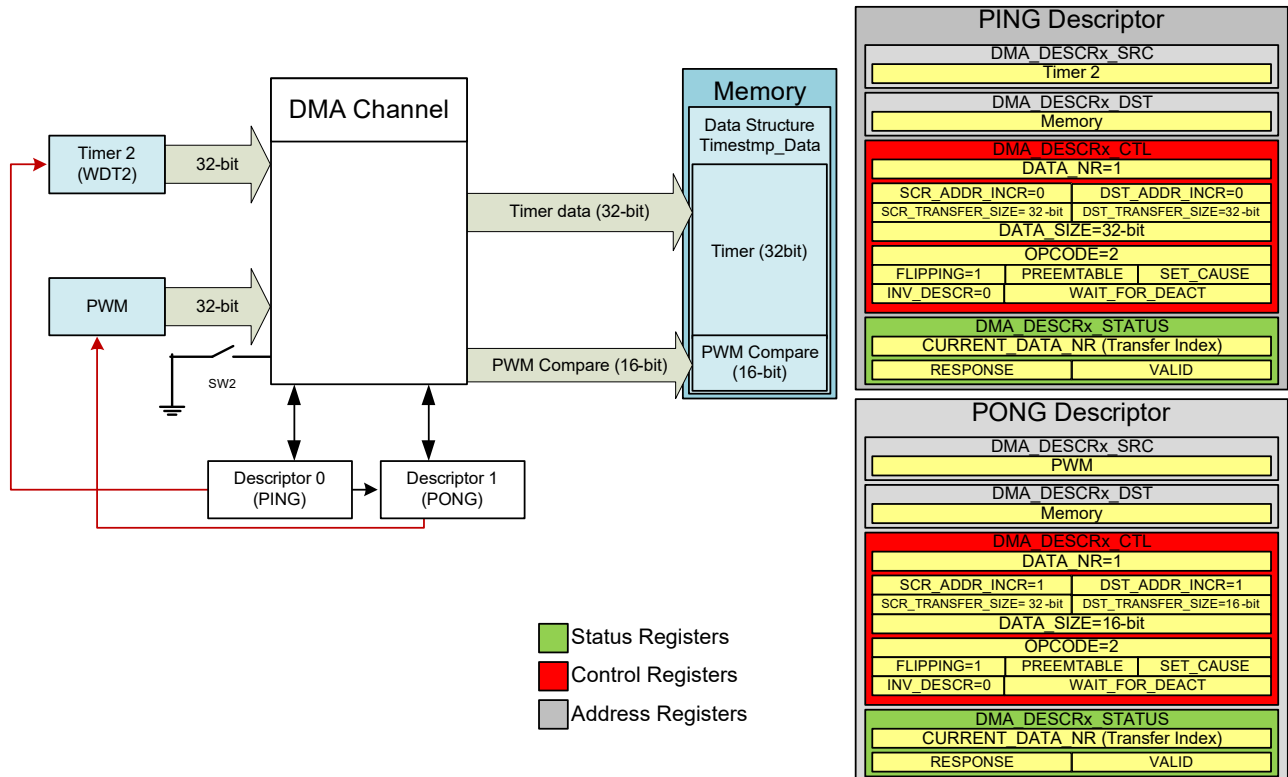
Figure 5-9 は、PING および PONG 記述子が OPCODE 2 動作作用に構成されており、PING レジスタの 2 回目の反復で、CPU によって FLIPPING が無効にされている場合を示します。

Figure 5-9. OPCODE 2 の DMA 転送の例



OPCODE 2 転送モードは、個別のユースケースを実装するようにカスタマイズできます。Figure 5-10 は、このような使用例の 1 つを示します。ここで、ソースデータは、連続したメモリではない 2 つの異なる場所から取得できます。宛先は、連続したメモリ位置にあるデータ構造です。1 つのソースはタイミングデータを保持する Timer 2 で、もう 1 つのソースは PWM コンペアレジスタです。両方のデータは、メモリ内の連続した場所に保存されます。

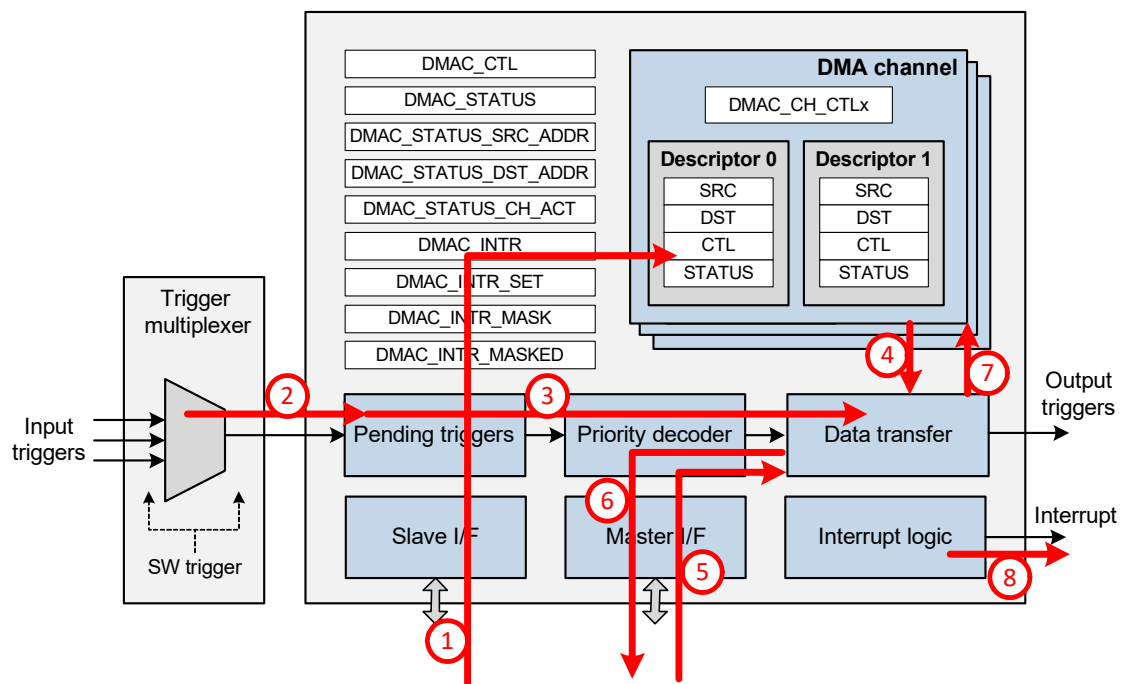
Figure 5-10. OPCODE 2: メモリへの複数のソース



5.3 動作とタイミング

Figure 5-11 は、トリガー、データ、または割り込みフローが重畳された DMA コントローラの設計を示します。

Figure 5-11. 動作フロー



このフローは、DMA コントローラのデータ転送に関するステップを例示します。

1. メイン CPU は、特定のチャンネルの記述子構造をプログラムします。また、チャンネルの特定のシステムトリガーを選択する DMA レジスタをプログラムします。
2. channel's のシステムトリガーがアクティブ化されます。
3. 優先度のデコードは、最も優先度の高いアクティブ化されたチャンネルを決定します。
4. データ転送エンジンは、アクティブ化されたチャンネルを受け入れ、チャンネル識別子を使用して、チャンネル記述子構造をロードします。記述子構造は、チャンネルのデータ転送を指定します。
5. データ転送エンジンは、マスター I/F を使用してソースの場所からデータをロードします。
6. データ転送エンジンは、マスター I/F を使用してデータを宛先の場所に格納します。単一要素 (オペコード 0) の転送では、ステップ 5 と 6 が 1 回実行されます。複数要素記述子 (オペコード 1 または 2) の転送では、複数のデータ要素転送を実装するために、手順 5 および 6 を順番に複数回実行できます。
7. データ転送エンジンは、channel's 記述子構造を更新してデータ転送を反映し、記述子 SRAM に格納します。
8. 記述子チャンネル構造で指定されたすべてのデータ転送が完了すると、割り込みが生成される場合があります (これはプログラム可能なオプションです)。

DMA コントローラのデータ転送ステップは、初期化ステップ、並行ステップ、または順次ステップのいずれかに分類できます。

- 初期化: これには、記述子構造をプログラムするステップ 1 が含まれます。このステップは、記述子構造ごとに行われます。これはメイン CPU によって実行され、アクティブ化されたチャンネルトリガーによって開始されることはありません。
- 同時: これにはステップ 2 と 3 が含まれます。これらのステップは、チャンネルごとに並行して実行されます。
- 順次: これには、ステップ 4 から 8 が含まれます。これらのステップは、アクティブ化されたチャンネルごとに順次実行されます。その結果、DMA コントローラのスループットは、これらの手順を実行するのにかかる時間によって決まります。この時間は 2 つの部分で構成されます。コントローラが (記述子をロードして保管するために) 費やした時間とバスインフラストラクチャで費やした時間です。後者の時間は、バスのレイテンシ (アービタおよびブリッジコンポーネントによって決定されます) およびターゲットメモリ / ペリフェラルに依存します。

単一のデータ要素を転送する場合、AHB-Lite バスに待機状態がないという想定の下で、1 つの完全な転送を完了するのに 12 クロックサイクルかかります。このモードで転送を完了するためのサイクル数の式を次に示します。

$$\text{サイクル数} = 12 + \text{LOAD 待機状態} + \text{STORE 待機状態}$$

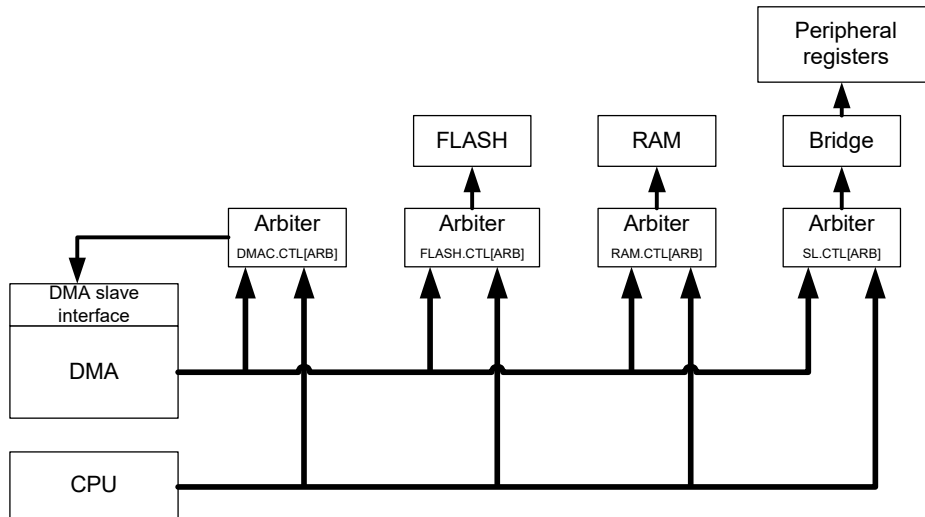
ディスクリプタ全体またはチェーンディスクリプタチェーンを転送する場合、最初のデータ要素には 12 クロックサイクルが必要です。後続の要素には 3 サイクルが必要です。これは、AHB-Lite バスで待機状態がないという前提でもあります。「N」要素を転送するサイクル数の式を次に示します。

$$\text{サイクル数} = (12 + \text{LOAD 待機状態} + \text{STORE 待機状態}) + (N-1) * (3 + \text{LOAD 待機状態} + \text{STORE 待機状態})$$

5.4 アービトレーション

デバイスの AHB バスには、CPU と DMA コントローラの 2 つのマスターがあります。すべての周辺機器とメモリは、スレーブインターフェースを介してバスに接続します。独自のアービタを備えたフラッシュメモリおよび RAM 用の専用スレーブインターフェースがあります。周辺レジスタはすべて、ブリッジを介して専用のアービタへの単一のスレーブインターフェースに接続します。DMA controller's の制御レジスタにアクセスするために使用される DMA コントローラのスレーブインターフェースは、すべて別のスレーブインターフェースを介して接続します。Figure 5-12 は、このアーキテクチャを示します。

Figure 5-12. PSoC 4 バスアーキテクチャ

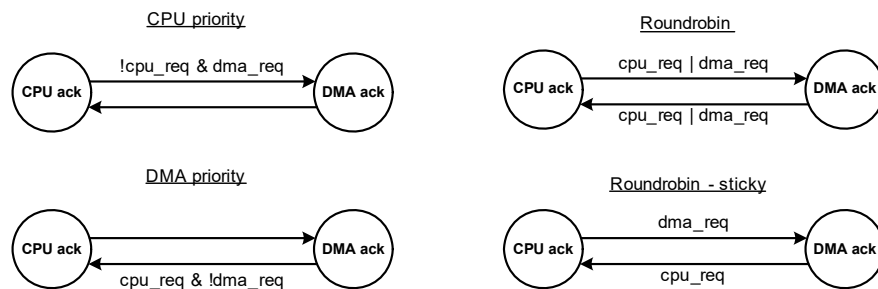


各スレーブの調停ポリシーは、次のいずれかです。

- CPU 優先度 :CPU は常に調停を優先します。DMA アクセスは、CPU 要求がない場合にのみ許可されます。
- DMA 優先度 :DMA は常に調停を優先します。CPU へのアクセスは、DMA 要求がない場合にのみ許可されます。
- ラウンドロビン : 調停の優先度により、すべてのリクエストで DMA と CPU の間の切り替えが継続されます。調停の優先度は、CPU または DMA の要求ごとに切り替わります。
- ラウンドロビンスティッキー : このモードはラウンドロビンに似ていますが、優先度は、優先度の低いマスターからのリクエストがあった場合にのみ切り替わります。たとえば、現在の優先度が CPU であり、DMA による要求があった場合、次の要求のために優先度が DMA に切り替わります。DMA からの要求がない場合、CPU は現在の優先度を保持します。

調停モデルは、次の図を使用して説明されています。

Figure 5-13. 調停モデル



5.5 レジスタ一覧

レジスタ名	備考	特長
DMAC_CTL	ブロック制御	DMA コントローラのイネーブル 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	現在の宛先アドレス	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	チャン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取って、すべてのアクティブな保留中のチャンネル (保留中またはデータ転送エンジン内) の情報を取得します。
DMAC_CH_CTLx	チャンネル制御レジスタ	チャンネル x についてチャンネル イネーブル、PING/PONG および優先度設定の機能を提供
DMAC_DESCRx_PING_SRC	PING ディスクリプタの転送元アドレス	チャンネル x 用の転送元の位置のベース アドレス
DMAC_DESCRx_PING_DST	PING ディスクリプタの転送先アドレス	チャンネル x 用の転送先の位置のベース アドレス
DMAC_DESCRx_PING_CTL	PING ディスクリプタの制御ワード	PING ディスクリプタのすべての制御設定
DMAC_DESCRx_PING_STATUS	PING 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の Data_NR インデックスステータス。
DMAC_DESCRx_PONG_SRC	PONG ディスクリプタの転送元アドレス	チャンネル x 用の転送元の位置のベース アドレス
DMAC_DESCRx_PONG_DST	PONG ディスクリプタの転送先アドレス	チャンネル x 用の転送先の位置のベース アドレス
DMAC_DESCRx_PONG_CTL	PONG ディスクリプタの制御ワード	PONG ディスクリプタのすべての制御設定
DMAC_DESCRx_PONG_STATUS	PONG 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の Data_NR 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	読み出し時、このレジスタはビット単位で、割り込み要求とマスクレジスタの間を反映します。このレジスタにより、ソフトウェアは、2 つのロード操作ではなく 1 つのロード操作で、マスクが有効なすべての割り込み原因のステータスを読み取ることができます。1 つは割り込み原因用、もう 1 つはマスク用です。ファームウェア開発が容易になる。

6. 割り込み



PSoC[®]4 の Arm Cortex-M0 + (CM0 +) CPU は、割り込みと例外をサポートしています。割り込みとは、タイマ、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ールトと内部システムタイマイベントなどの CPU によって生成されるイベントのことです。割り込みや例外が発生すると、いずれも現時点のプログラム実行が停止され、例外ハンドラまたは割り込みサービスルーチン (ISR) が CPU によって実行されます。デバイスは割り込みハンドラ /ISR と例外ハンドラの両方が統合された例外ベクタテーブルを持っています。

6.1 特長

PSoC4 が対応する割り込み機能を次に示します。

- 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 (NVIC) が CPU コアに統合され、低い割り込みレイテンシの実現
- ベクタテーブルは、フラッシュまたは SRAM のいずれかに配置することが可能
- 各割り込みに 0 から 3 までの優先度を設定可能
- レベルトリガーおよびパルストリガー割り込み信号

6.2 動作原理

Figure 6-1. PSoC 4 割り込みブロック図

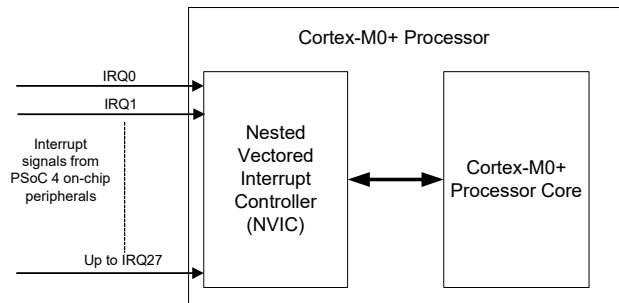


Figure 6-1 は、割り込み信号と Cortex-M0 + CPU 間の相互作用を示します。PSoC 4 には最大 32 個の割り込みがあります。これらの割り込み信号は、NVIC によって処理されます。NVIC は個々の割り込みの有効制御、優先度管理、CPU コアとの通信を担当します。例外は、CPU の外部のペリフェラルによって生成される割り込みとは異なり、CM0 + コアによって生成されるイベントの一部であるため、Figure 6-1 には示されていません。

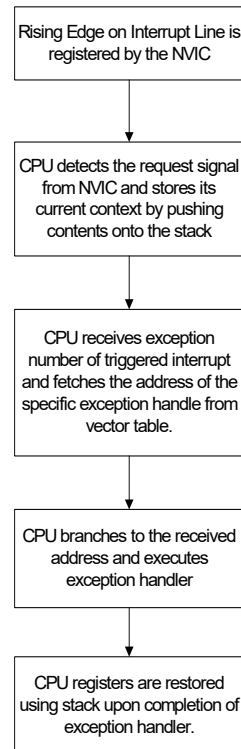
6.3 割り込みと例外の動作

6.3.1 割り込み / 例外の処理

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。Table 6-1 に示すように、すべての割り込みと例外には一意の例外番号があります。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を実行します。
5. 例外ハンドラが終了すると、CPU レジスタはスタックからポップし、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。

Figure 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優先度を評価し、CPU に最も高い優先度の割り込みの例外番号を送信します。従って、優先度の高い割り込みは優先度の低い ISR の実行をいつでもブロックすることができます。

例外は割り込みが処理される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を実行するために CPU に使用されます。

6.3.2 レベルとパルスの割り込み

NVIC は、割り込みライン (IRQ0 ~ IRQ31) でレベル信号とパルス信号の両方をサポートします。割り込みソースに応じてレベルおよびパルスのどちらかに分類されます。

Figure 6-3. レベル割り込み

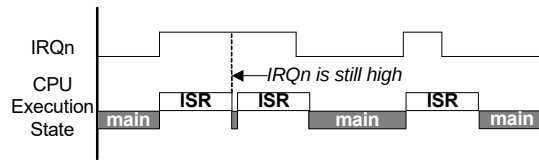


Figure 6-4. パルス割り込み

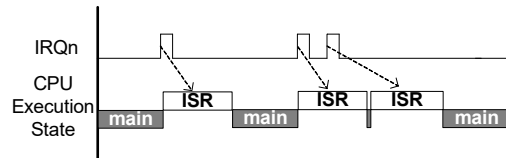


Figure 6-3 と Figure 6-4 は、それぞれレベル割り込みとパルス割り込みの動作を示します。以下のイベントシーケンスは、割り込みラインが最初に非アクティブ（論理 LOW）であるとして、レベルとパルス割り込みの処理を示します。

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です。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 実行が完了すると、保留中の割り込みが再び処理されます（パルス割り込みについては、Figure 6-4 を参照）。
4. ISR の完了後も割り込み信号がまだ高い場合は、保留状態になり、ISR が再度実行されます。Figure 6-3 は、レベルトリガー割り込みの場合を示します。割り込み信号がハイである限り、ISR が実行されます。

6.3.3 例外ベクタ テーブル

例外ベクタテーブル (Table 6-1) には、すべての例外ハンドラのエントリポイントアドレスが格納されています。CPU は例外番号に基づき、適切なアドレスをフェッチします。

Table 6-1. 例外ベクタ テーブル

例外番号	例外	例外優先度	ベクタ アドレス
–	初期スタック ポインタの値	該当なし (NA)	ベース アドレス : 0x00000000 (フラッシュ メモリの開始) または 0x20000000 (SRAM の開始)
1	リセット	–3、最優先	ベース アドレス + 0x04
2	マスク不可能割り込み (NMI)	–2	ベース アドレス + 0x08
3	ハードフォールト	–1	ベース アドレス + 0x0C
4 ~ 10	予約済み	なし	ベース アドレス + 0x10 ~ ベース アドレス + 0x28
11	監視呼び出し (SVCall)	設定可能 (0 ~ 3)	ベース アドレス + 0x2C
12 ~ 13	予約済み	なし	ベース アドレス + 0x30 ~ ベース アドレス + 0x34
14	PendSupervisory (PendSV)	設定可能 (0 ~ 3)	ベース アドレス + 0x38
15	システム タイマ (SysTick)	設定可能 (0 ~ 3)	ベース アドレス + 0x3C
16	外部割り込み (IRQ0)	設定可能 (0 ~ 3)	ベース アドレス + 0x40
...	...	設定可能 (0 ~ 3)	...
43	外部 割り込み (IRQ31)	設定可能 (0 ~ 3)	Base_Address + 0xBC

Table 6-1 では、最初のワード (4 バイト) は例外番号 0 としてマークされていません。例外テーブルの最初のワードは、メインスタック ポインタ (MSP) の値をデバイスリセットで初期化するために使用されるためです。例外とみなされません。ベクタテーブルは、ベクタテーブル オフセットレジ

スタ (VTOR) を変更することにより、メモリマップ (フラッシュまたは SRAM) の任意の場所に配置できます。このレジスタは、0xE000ED08 にある CM0+ のシステム制御空間の一部です。このレジスタは、ベクタテーブル アドレスのビット 31:8 を取ります。ビット 7:0 は予約済みです。したがっ

て、ベクタテーブルアドレスは 256 バイトにアラインする必要があります。ベクタテーブルを SRAM に移動する利点は、SRAM ベクタテーブルの内容を変更することで、例外ハンドラのアドレスを動的に変更できることです。しかし不揮発性フラッシュメモリのベクタテーブルをフラッシュメモリ書き込みによって変更する必要があります。

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

また、CPUSS_SYSREQ レジスタの SYSCALL_REQ ビットが設定されている場合、フラッシュアドレス 0x00000008 の読み取りは SROM にリダイレクトされ、フラッシュからではなく NMI ベクタアドレスをフェッチします。CPUSS_SYSREQ をリセットして、アドレス 0x00000008 のフラッシュを読み取ります。

例外ソース (例外番号 1 ~ 15) については、6.4 例外ソースで説明しています。Table 6-1 で予約済みとマークされている例外は使用されませんが、ベクタテーブルではアドレスが予約されています。割り込み要因 (例外番号 16 ~ 35) については、6.5 割り込みソースで説明しています。

6.4 例外ソース

このセクションでは、Table 6-1 にリストされているさまざまな例外ソースについて説明します (例外番号 1 から 15)。

6.4.1 リセット例外

デバイスリセットは PSoc 4 で例外として扱われます。最も優先度の高い例外である -3 の優先度に常に有効にされています。デバイスリセットは、パワーオンリセット (POR)、XRES 端子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、デバイスコンフィギュ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によってプログラムされ、外部ユーザ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 (Table 6-1 の例外 #1) は、スタートアップコードの場所をフラッシュメモリに格納します。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。デバイスで NMI 例外をトリガーするものは 2 つあります：

- **NMIPENDSET ビット (ユーザ NMI 例外) をセットすることによる NMI 例外：**NMI 例外は割り込み制御状態レジスタ (CM0P_ICSR レジスタ) に NMIPENDSET ビットを立てることにより、ソフトウェアでトリガーすることができます。このビットを立てることでアクティブなベクタテーブル (フラッシュまたは SRAM ベクタテーブル) に示された NMI ハンドラが実行されます。
- **システムコール NMI 例外：**フラッシュ書き込み動作とフラッシュチェックサム動作などの不揮発プログラム動作のために使用されます。これは、CPUSS_SYSREQ レジスタで SYSCALL_REQ ビットをセットすることでトリガーします。SYSCALL_REQ ビットによってトリガーされる NMI 例外は、常に SROM にある NMI 例外ハンドラコードを実行します。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 例外には使用されません。SROM の NMI ハンドラコードは、ユーザ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あるため、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ールト条件のためのキャッチオール例外です。CM0+ CPU は、HardFault 例外ハンドラにフォー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ールト状態から回復する能力を有している場合、ハンドラ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求するスーパーバイザコールを発行することを可能にします。PSoC 4のCM0+がシステム呼び出しNMI例外に対してSVCall例外に関連しない特権モードを使用することに注意してください。(特権モードの詳細については、[チップ動作モード \(105 ページ\)](#)を参照してください。)デバイスのアーキテクチャレベルでは、SVCallに対する他の特権モードサポートはありません。アプリケーション開発者は、エンドアプリケーションの要件に応じてSVCall例外ハンドラ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により処理を待ちます。システムハンドラ制御および状態レジスタ(SHCSR)のSVCALLPENDEッドビットは、SVCall例外の保留状態を確認、変更を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CM0P_ICSRのPENDSVSETビットをセットすることでトリガーされます。このビットを立てると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CM0P_ICSRのPENDSVCLRビットをセットすることでクリアすることができます。PendSV例外の優先度は、システムハンドラ優先度レジスタ3 (CM0P_SHPR3)の2ビットフィールドPRI_14[23:22]に値を書き込むことによって、0と3の間に設定できます。詳細については、「[Armrv6-M Architecture Reference Manual](#)」を参照してください。

6.4.6 SysTick 例外

PSoC 4のCM0+ CPUは、SysTickと呼ばれるシステムタイマをその内部アーキテクチャの一部として搭載しています。SysTickは、RTOSティックタイマ、高速アラームタイマ、単純なカウンタなどの様々な時間管理の目的のために簡単な24ビットデクリメントのカウンタを提供しています。SysTickタイマはそのカウントがゼロに達した時に割り込みを生成するよう設定することができ、その割り込みをSysTick例外と呼びます。この例外は、SysTick制御およびステータスレジスタ(CM0P_SYST_CSR)のTICKINTビット

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、システムハンドラ優先度レジスタ3 (CM0_SHPR3)の2ビットフィールドPRI_15[31:30]に値を書き込むことによって、0と3の間に設定できます。SysTick例外は割り込み制御状態レジスタCM0P_ICSRのPENDSTSETbビットに1を書き込むことによって、どの瞬間においても常にソフトウェアで生成することができます。同様に、SysTick例外の保留状態は、割り込み制御状態レジスタであるCM0P_ICSRのPENDSTCLRビットをセットすることによってクリアすることができます。

6.5 割り込みソース

PSoC 4 は、ペリフェラルからの最大 32 の割り込み (IRQ0 から IRQ31 または例外番号 16 – 47) をサポートします。各割り込みのソースは、Table 6-2、Table 6-3、および Table 6-4 にリストされています。PSoC 4 は割り込みラインそれぞれのための柔軟な選択機構を提供しています。割り込みは、TCPWM およびシリアル通信ブロックなどのオンチップ ペリフェラルからの標準の割り込みを含みます。生成された割り込みは、通常異なるペリフェラル ステートとの論理 OR です。ペリフェラル ステータス レジスタ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。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 ISR でペリフェラルステータスレジスタを読み取る必要があります。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。

GPIO 割り込みの詳細については、I/O システム (66 ページ) を参照してください。

Table 6-2. PSoC 4100S 割り込みソースのリスト

割り込み	Cortex-M0+ 例外番号	割り込みソース
NMI	2	SYSCALL_REQ
IRQ0	16	GPIO 割り込み - ポート 0
IRQ1	17	GPIO 割り込み - ポート 1
IRQ2	18	GPIO 割り込み - ポート 2
IRQ3	19	GPIO 割り込み - ポート 3
IRQ4	20	GPIO 割り込み - すべてのポート
IRQ5	21	LPCOMP (低電力コンパレータ)
IRQ6	22	WDT (ウォッチドッグ タイマ)
IRQ7	23	SCB0 (シリアル通信ブロック 0)
IRQ8	24	SCB1 (シリアル通信ブロック 1)
IRQ9	25	SCB2 (シリアル通信ブロック 2)
IRQ10	26	CTBm (連続時間ブロックミニ) - すべての CTBm
IRQ11	27	WCO WDT 割り込み
IRQ12	28	SPCIF 割り込み
IRQ13	29	CSD (CapSense)
IRQ14	30	TCPWM0 (タイマ / カウンタ / PWM0)
IRQ15	31	TCPWM1 (タイマ / カウンタ / PWM1)
IRQ16	32	TCPWM2 (タイマ / カウンタ / PWM2)
IRQ17	33	TCPWM3 (タイマ / カウンタ / PWM3)
IRQ18	34	TCPWM4 (タイマ / カウンタ / PWM4)
IRQ19	35	SAR ADC

Table 6-3. PSoC 4100S Plus 割り込みソース

割り込み	Cortex-M0+ 例外番号	割り込みソース
NMI	2	SYSCALL_REQ
IRQ0	16	GPIO 割り込み - ポート 0
IRQ1	17	GPIO 割り込み - ポート 1
IRQ2	18	GPIO 割り込み - ポート 2
IRQ3	19	GPIO 割り込み - ポート 3
IRQ4	20	GPIO 割り込み - すべてのポート
IRQ5	21	LPCOMP (低電力コンパレータ)
IRQ6	22	WDT (ウォッチドッグ タイマ)
IRQ7	23	SCB0 (シリアル通信ブロック 0)
IRQ8	24	SCB1 (シリアル通信ブロック 1)
IRQ9	25	SCB2 (シリアル通信ブロック 2)
IRQ10	26	SCB3 (シリアル通信ブロック 3)
IRQ11	27	SCB4 (シリアル通信ブロック 4)
IRQ12	28	CTBm (連続時間ブロックミニ) - すべての CTBm
IRQ13	29	WCO WDT 割り込み
IRQ14	30	DMA 割り込み
IRQ15	31	SPCIF 割り込み
IRQ16	32	CSD (CapSense)
IRQ17	33	TCPWM0 (タイマ / カウンタ /PWM0)
IRQ18	34	TCPWM1 (タイマ / カウンタ /PWM1)
IRQ19	35	TCPWM2 (タイマ / カウンタ /PWM2)
IRQ20	36	TCPWM3 (タイマ / カウンタ /PWM3)
IRQ21	37	TCPWM4 (タイマ / カウンタ /PWM4)
IRQ22	38	TCPWM5 (タイマ / カウンタ /PWM5)
IRQ23	39	TCPWM6 (タイマ / カウンタ /PWM6)
IRQ24	40	TCPWM7 (タイマ / カウンタ /PWM7)
IRQ25	41	SAR ADC
IRQ26	42	CAN 割り込み
IRQ27	43	暗号割り込み

Table 6-4. PSoC 4100S Plus 256 KB 割り込みソース

割り込み	Cortex-M0+ 例外番号	割り込みソース
NMI	2	SYSCALL_REQ
IRQ0	16	GPIO 割り込み - ポート 0
IRQ1	17	GPIO 割り込み - ポート 1
IRQ2	18	GPIO 割り込み - ポート 2
IRQ3	19	GPIO 割り込み - ポート 3
IRQ4	20	GPIO 割り込み - すべてのポート
IRQ5	21	LPCOMP (低電力コンパレータ)
IRQ6	22	WDT (ウォッチドッグタイマ)
IRQ7	23	SCB0 (シリアル通信ブロック 0)
IRQ8	24	SCB1 (シリアル通信ブロック 1)
IRQ9	25	SCB2 (シリアル通信ブロック 2)
IRQ10	26	SCB3 (シリアル通信ブロック 3)
IRQ11	27	SCB4 (シリアル通信ブロック 4)
IRQ12	28	CTBm0 (Continuous Time Block mini) - すべての CTBm0
IRQ13	29	CTBm1 (Continuous Time Block mini) - すべての CTBm1
IRQ14	30	WCO WDT 割り込み
IRQ15	31	DMA 割り込み
IRQ16	32	SPCIF 割り込み
IRQ17	33	CSD (CapSense)
IRQ18	34	TCPWM0 (タイマ / カウンタ / PWM0)
IRQ19	35	TCPWM1 (タイマ / カウンタ / PWM1)
IRQ20	36	TCPWM2 (タイマ / カウンタ / PWM2)
IRQ21	37	TCPWM3 (タイマ / カウンタ / PWM3)
IRQ22	38	TCPWM4 (タイマ / カウンタ / PWM4)
IRQ23	39	TCPWM5 (タイマ / カウンタ / PWM5)
IRQ24	40	TCPWM6 (タイマ / カウンタ / PWM6)
IRQ25	41	TCPWM7 (タイマ / カウンタ / PWM7)
IRQ26	42	SAR ADC 0
IRQ27	43	SAR ADC 1
IRQ28	44	EXCO 割り込み
IRQ29	45	予約済み
IRQ30	46	予約済み
IRQ31	47	予約済み

6.6 例外優先度

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調停に有用です。PSoC 4 は異なる例外に優先度を設定することを柔軟に行えます。リセット、NMI、および HardFault 以外のすべての例外には、構成可能な優先度レベルを割り当てられます。リセット、NMI および HardFault 例外はそれぞれ -3、-2、-1 の固定された優先度を持っています。PSoC 4 では優先度番号が小さい方がより高い優先度を表します。よってリセット、NMI、HardFault の例外は最上位優先度です。他の例外は優先度 0 ~ 3 を割り当てることができます。

PSoC 4 は優先度の高い例外が現在アクティブな例外ハンドラを妨害（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の実行を再開します。PSoC 4 の CM0+ CPU は最大 4 つの例外ネストを可能にします。CPU は同じ優先度の複数の例外要求を受信する場合、最小の例外番号のものが最初に処理されます。

例外番号 1 ~ 15 の優先度を設定するレジスタについては、“[例外ソース](#)” (54 ページ) を参照してください。

28 個の割り込みの優先度 (IRQ0 ~ IRQ27) は、割り込み優先度レジスタ (CM0P_IPR) に書き込むことで構成できます。これは 32 ビットレジスタのグループで、[Table 6-5](#) に示すように、各レジスタには 4 つの割り込みの優先度の値が格納されています。レジスタの他のビットフィールドは使用されません。

Table 6-5. 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
23:22	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 割り込みのイネーブルとディセーブル

NVIC は、ソフトウェアで 28 の割り込みを個別に有効および無効にするレジスタを提供します。割り込みが有効にされない場合、NVIC はその割り込みラインの割り込み要求を処理しません。割り込み有効セットレジスタ (CM0P_ISER) と割り込み有効クリアレジスタ (CM0P_ICER) はそれぞれの割り込みを有効 / 無効にするために使用されます。これらは 32 ビット幅のレジスタで、各ビットは同じ番号の割り込みに対応しています。これらのレジスタは、割り込みの有効ステータスを取得するためにソフトウェアで読み取ることもできます。[Table 6-6](#) に、これら 2 つのレジスタ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効果がないことに注意してください。

Table 6-6. 割り込みイネーブル / ディセーブル レジスタ

レジスタ	動作	ビット値	備考
割り込みイネーブルセットレジスタ (CM0P_ISER)	書込み	1	割り込みを有効にする
		0	何もしない
	読出し	1	割り込みが有効
		0	割り込みが無効
割り込みイネーブルクリアレジスタ (CM0P_ICER)	書込み	1	割り込みを無効にする
		0	何もしない
	読出し	1	割り込みが有効
		0	割り込みが無効

CM0P_ISER および CM0P_ICER レジスタは、割り込み IRQ0 ~ IRQ31 にのみ適用されます。これらのレジスタを使用して、例外番号 1 ~ 15 を有効または無効にすることはできません。15 の例外は、“[例外ソース](#)” (54 ページ) で説明されているように、有効化と無効化を独自にサポートしています。

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。構成可能な優先度例外には、[Table 6-1](#) にリストされているリセット、NMI、および HardFault を除くすべての例外が含まれます。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) が設定されている場合、構成可能な優先度例外は CPU によって処理されませんが、PM ビットがクリアされた後、CPU による処理を待つ保留状態になる場合があります。

6.8 例外状態

各例外は次のいずれかの状態にあります。

Table 6-7. 例外状態

例外状態	意味
Inactive (非アクティブ)	例外はアクティブでも保留中でもない。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
Pending (保留中)	例外要求が CPU/NVIC によって受信され、例外が CPU による処理を待機中
Active (アクティブ)	CPU で処理されているが、例外ハンドラ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可能。この場合両方の例外がアクティブ状態になる
アクティブおよび保留中	例外はプロセッサによって処理され、例外ハンドラの実行中に同じソースからの保留中の要求がある。

割り込み制御状態レジスタ (CM0P_ICSR) はさまざまな例外状態を説明するステータスビットがあります。

- CM0P_ICSR の VECTACTIVE ビット ([8:0]) は現時点実行中の例外番号を格納します。CPU が例外ハンドラを (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を注意してください。
- CM0P_ICSR の VECTPENDING ビット ([20:12]) は最上位優先度の保留中の例外の番号を格納します。保留中の例外が存在しない場合、この値はゼロです。
- CM0P_ICSR の ISRPENDING ビット (ビット 22) は、NVIC によって生成された割り込み (IRQ0 ~ IRQ27) が保留状態にあるかどうかを示します。

6.8.1 保留中の例外

周辺装置が NVIC に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。

NVIC は、割り込みの保留状態を設定およびクリアするための個別のレジスタビットを提供することにより、29 の割り込みラインのソフトウェア保留を可能にします。割り込み保留セットレジスタ (CM0P_ISPR) と割り込み保留クリアレジスタ (CM0P_ICPR) は割り込みラインの保留状態をセットまたはクリアするために使用されます。これらは 32 ビット幅のレジスタで、各ビットは同じ番号の割り込みに対応しています。

Table 6-8 は、これら 2 つのレジスタ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効果がないことに注意してください。

Table 6-8. 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	備考
割り込み保留セットレジスタ (CM0P_ISPR)	書込み	1	割り込みを保留状態にする
		0	何もしない
	読出し	1	割り込みが保留中
		0	割り込みが保留中でない
割り込み保留クリアレジスタ (CM0P_ICPR)	書込み	1	保留の割り込みをクリア
		0	何もしない
	読出し	1	割り込みが保留中
		0	割り込みが保留中でない

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

CM0P_ISPR および CM0P_ICPR レジスタは、32 個の周辺割り込み (例外番号 16 ~ 47) にのみ使用されることに注意してください。これらのレジスタは、例外番号 1 から 15 を保留するためには使用できません。これらの 15 の例外は、“[例外ソース](#)” (54 ページ) で説明されているように、保留に対する独自のサポートがあります。

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) 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の実行 (ハンドラモード) を開始します。CPU は 8 個の 32 ビットの内部レジスタをスタックにプッシュします。これらのレジスタはプログラムステータスレジスタ (PSR)、戻りアドレス、リンクレジスタ (LR または R14)、R12、R3、R2、R1 および R0 です。Cortex-M0+ には、MSP と PSP の 2 つのスタックポインタがあります。1 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

プッシュの動作後、例外ハンドラの実行をするために、CPU はハンドラ 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 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。詳細については、[Cortex-M0+ CPU \(28 ページ\)](#) を参照してください。

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には見えず、内部プロセッサアーキテクチャの一部です。テールチェーンと遅延到着メカニズムの詳細については、[Arm Infocenter](#) にアクセスしてください。

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 PSoC 4 は、低消費電力モードからのデバイス復帰を可能にします。ウェイクアップ割り込みコントローラ (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 およびディープ スリープ モードへの遷移をトリガーします。さまざまな低電力モードに入る順序については、[電力モード \(107 ページ\)](#) で詳しく説明しています。チップ低消費電力モードは固定機能割り込みソースの 2 つの分類があります：

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

6.11 例外 – 初期化と構成

本節はPSoC 4での例外の初期化とコンフィギュ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の場所の構成：例外を使用する最初の手順は、必要に応じて、フラッシュメモリまたはSRAMのいずれかでベクタテーブルの場所を構成することです。この構成は、VTOR レジスタのビット 31:28 に、ベクタテーブルが存在するフラッシュまたはSRAM アドレスの値を書き込むことによって行われます。このレジスタの書き込みは、デバイス初期化コードの一部として行われます。
アプリケーションがベクタ アドレスを動的に変更する必要がある場合、ベクタ テーブルがSRAMで利用できることを推奨します。テーブルがフラッシュに配置している場合には、フラッシュ書き込み動作でベクタ 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ギュレーション：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を設定してください。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. 例外ハンドラ関数を定義し、関数のアドレスを例外ベクタテーブルに書き込んでください。[Table 6-1](#) は、例外ベクタテーブル形式を示します。例外ハンドラのアドレスは、テーブルの適切な例外番号エントリに書き込む必要があります。
 - c. “例外優先度” ([59 ページ](#)) の説明に従って、例外の優先度を設定してください。
 - d. “割り込みのイネーブルとディセーブル” ([59 ページ](#)) で説明されているように、例外を有効にしてください。

6.12 レジスタ

Table 6-9. レジスタ一覧

レジスタ名	説明
CM0P_ISER	割り込みイネーブルセット レジスタ
CM0P_ICER	割り込みイネーブル クリア レジスタ
CM0P_ISPR	割り込み保留セット レジスタ
CM0P_ICPR	割り込み保留クリア レジスタ
CM0P_IPR	割り込み優先度レジスタ
CM0P_ICSR	割り込み制御状態レジスタ
CM0P_AIRCR	アプリケーション割り込みおよびリセット制御レジスタ
CM0P_SCR	システム制御レジスタ
CM0P_CCR	構成および制御レジスタ
CM0P_SHPR2	システムハンドラ優先度レジスタ 2
CM0P_SHPR3	システム ハンドラ優先度レジスタ 3
CM0P_SHCSR	システムハンドラ制御および状態レジスタ
CM0P_SYST_CSR	Systick コントロールおよびステータスレジスタ
CPUSS_CONFIG	CPU サブシステム構成レジスタ
CPUSS_SYSREQ	システム要求レジスタ

6.13 関連文書

- [Armv6-M Architecture Reference Manual](#)– このドキュメントでは、命令セット、NVIC アーキテクチャ、CPU レジスタの説明など、Arm Cortex-M0 + アーキテクチャについて説明します。

7. 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設計を保護する多数の選択肢を提供します。デバッグ機能を無効にするとともにフラッシュ保護を有効にすることにより高レベルのセキュリティを提供します。

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

注：最大のデバイスセキュリティが有効になっている場合、すべてのプログラミング、デバッグ、およびテストインターフェースが無効になるため、完全なデバイスセキュリティが有効になっている PSoC 4 デバイスは、障害分析に返されない場合があります。

7.1 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。

さらに、デバイスは個別のフラッシュ列データの保護も提供します。

7.2 動作原理

7.2.1 デバイスセキュリティ

CPU は通常のユーザ 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 モードまたは特権モードで動作します。ユーザ モードではフラッシュはプログラム可能で、デバッグ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：**ユーザは OPEN モードから PROTECTED モードに変更できます。このモードはユーザ コードまたはメモリへのすべてのデバッグ アクセスを無効にします。PROTECTED モードでは、少数の限られたレジスタにのみ

アクセスできます。フラッシュを再プログラムするためのレジスタへのデバッグ アクセスは不可能です。フラッシュを完全に消去した後にのみ、デバイスを OPEN モードに戻すことができます。

- **KILL モード**: ユーザは OPEN モードから KILL モードに変更できます。このモードでは、ユーザ 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ません。ほとんどのレジスタへのアクセスはまだ可能です。フラッシュを再プログラムするためのレジスタへのデバッグ アクセスは不可能です。デバイスは KILL モードの終了はできません。KILL モードにあるデバイスは不具合解析ができない場合があります。

7.2.2 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。この機能はユーザ独自のコードを保護するために利用されますが、フラッシュのブートローダ部分への予期しない上書きから保護するためにも使用されます。

フラッシュ メモリは行で構成されます。各行に 2 つの保護レベルの 1 つを割り当てられます。Table 7-1 を参照してください。フラッシュの保護レベルは、フラッシュの完全消去を実行することによってのみ変更できます。

詳細については、[不揮発性メモリ プログラム \(296 ページ\)](#) を参照してください。

Table 7-1. フラッシュ保護レベル

保護設定	許可	不可
Unprotected (未保護)	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	—
Full Protection (完全な保護)	外部読み出し ^a 内部読み出し	外部書き込み、 内部書き込み

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

Section C: システムリソースサブシステム (SRSS)

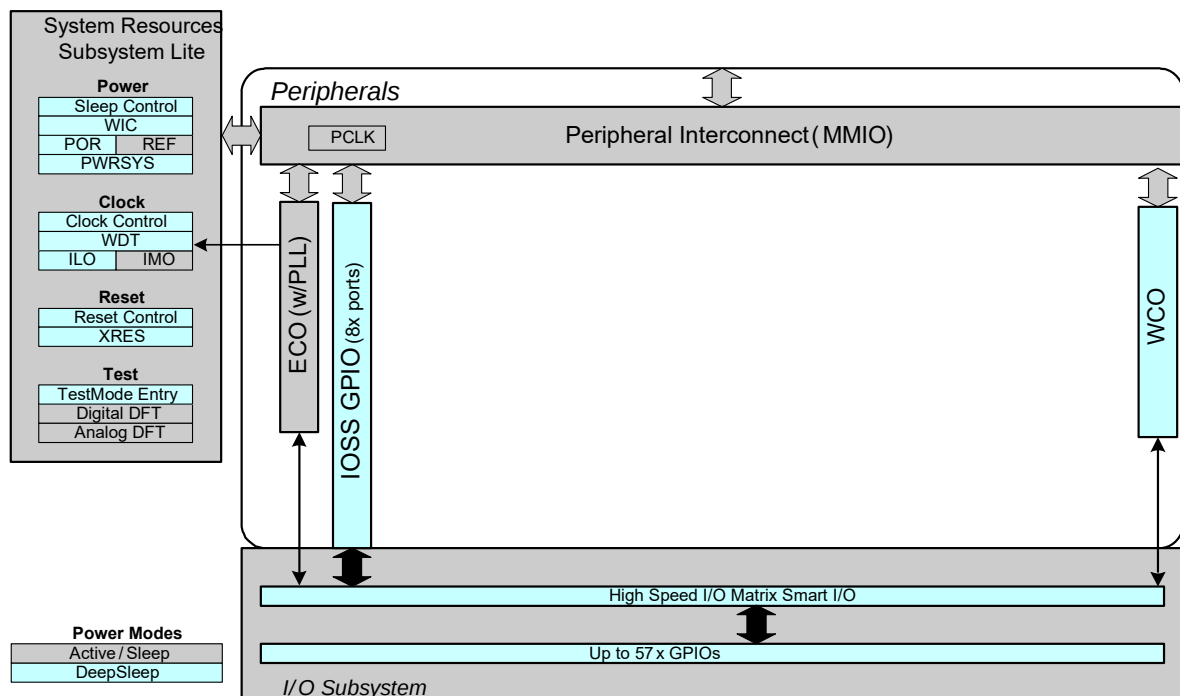


本セクションは次の章を含みます：

- I/O システム (66 ページ)
- クロック供給システム (89 ページ)
- 電源と監視 (101 ページ)
- チップ動作モード (105 ページ)
- 電力モード (107 ページ)
- ウォッチドッグタイマ (111 ページ)
- トリガーマルチプレクサブロック (116 ページ)
- リセット システム (120 ページ)

トップ レベル アーキテクチャ

システムリソース サブシステムのブロック図



8. I/O システム



本章では、PSoC[®] 4 の I/O システムの特長、アーキテクチャ、動作モードおよび割り込みについて説明します。PSoC 4 の GPIO ピンはポートにグループ分けされ、それぞれのポートは最大 8 本の GPIO を持っています。PSoC 4100S デバイスは 5 つのポートに配置された最大 36 個の GPIO を備え、PSoC 4100S Plus デバイスは 8 つのポートに最大 54 個の GPIO を備えています。

8.1 特長

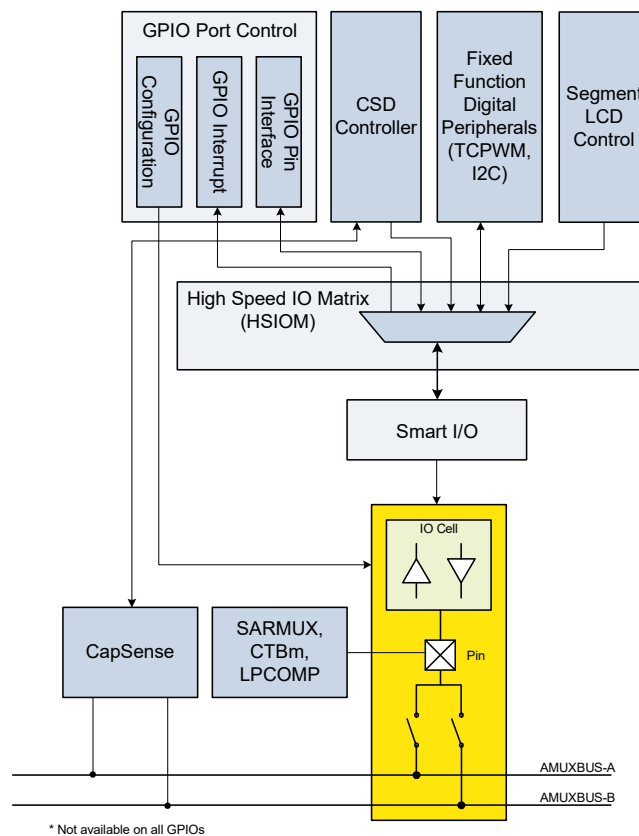
PSoC 4 の GPIO は以下の特長を持っています。

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

8.2 GPIO インターフェースの概要

PSoC 4 はアナログおよびデジタル周辺機器を搭載しています。Figure 8-1 は、ペリフェラルとピン間のルーティングの概要を示します。

Figure 8-1. GPIO インターフェースの概要

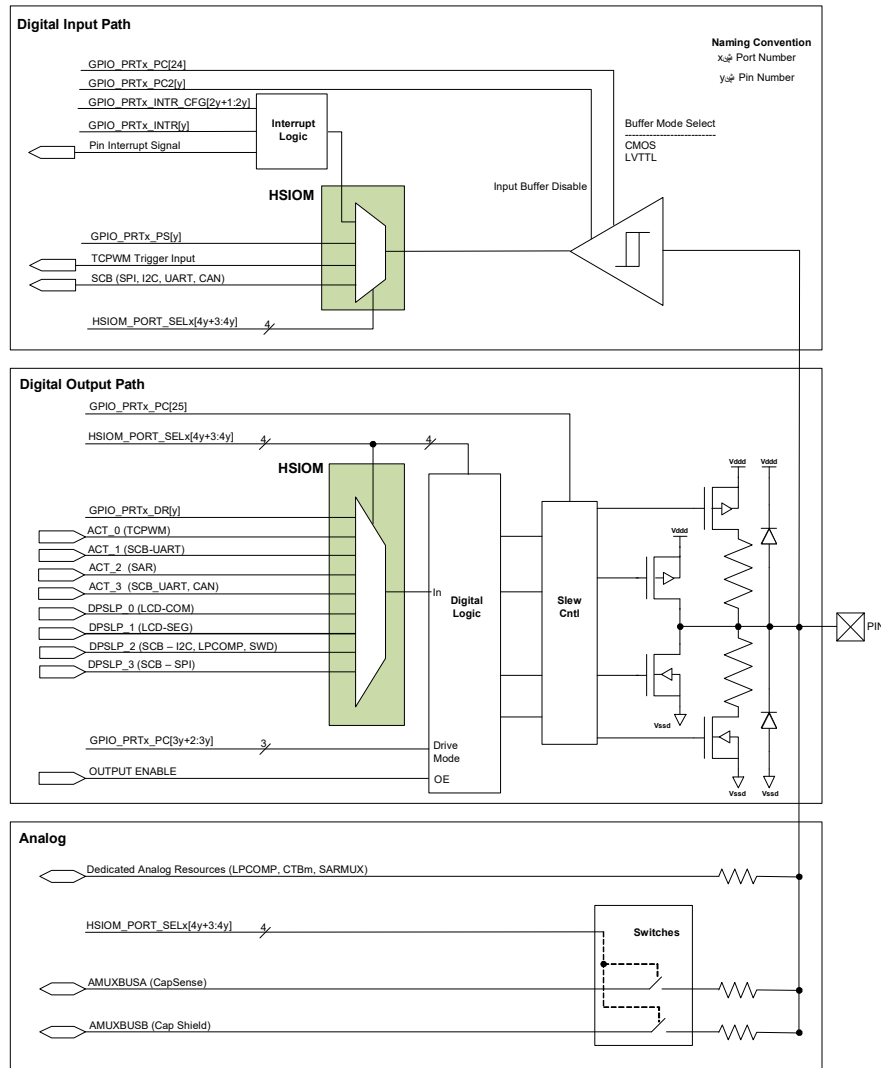


GPIO ピンは I/O セルに接続されます。これらのセルには、より高い入力インピーダンスを実現するためのデジタル入力用入力バッファ、およびデジタル出力信号用ドライバが搭載されています。デジタル ペリフェラルは高速 I/O マトリックス (HSIOM) を介して I/O セルに接続します。HSIOM はユーザにより選択されるペリフェラルをピンに接続するためのマルチプレクサを含んでいます。幾つかのポート ピンは、HSIOM とピン間にスマート I/O ブロックを搭載しています。スマート I/O ブロックはピン信号上の論理演算を可能にします。アナログ ペリフェラルおよびアナログ マルチプレクサ バ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

8.3 I/O セルのアーキテクチャ

Figure 8-2 に、I/O セルのアーキテクチャを示します。これは入力バッファと出力ドライバから構成されています。このアーキテクチャはあらゆる GPIO セルで共通です。デジタル入出力信号用に HSIOM マルチプレクサ / スマート I/O ブロックに接続します。

Figure 8-2. GPIO のブロック図



8.3.1 デジタル入力バッファ

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

- CMOS
- LVTTTL

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

Table 8-1. 入力バッファ モード

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

各モードのしきい値は、[PSoC 4100S datasheet](#)およびPSoC 4100S Plus 256 KB データシートから取得できます。入力バッファの出力は選択されたペリフェラルに配線するために HSIOM に接続します。HSIOM ポート選択レジスタ (HSIOM_PORT_SELx) に書き込むことでペリフェラルを選択します。[Figure 8-2](#) に示す HSIOM のデジタル入力周辺機器は、ピンに依存しています。各ピンで使用可能な機能については、[PSoC 4100S datasheet](#) および PSoC 4100S Plus 256 KB データシートを参照してください。

Table 8-2. 駆動モード設定

GPIO_PRTx_PC ('x' はポート番号を示し、'y' はピン番号を示します)				
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' のドライブモードを選択 (0 ≤ y ≤ 7)		
	High-Impedance Analog	0	High Z	High Z
	High-impedance Digital	1	High Z	High Z
	Resistive Pull Up (抵抗プルアップ)	2	Weak 1	Strong 0
	Resistive Pull Down (抵抗プルダウン)	3	Strong 1	Weak 0
	Open Drain, Drives Low (オープンドレイン、Low 駆動)	4	High Z	Strong 0
	Open Drain, Drives High (オープンドレイン、High 駆動)	5	Strong 1	High Z
	Strong Drive (ストロングドライブ)	6	Strong 1	Strong 0
	Resistive Pull Up and Down (抵抗プルアップとプルダウン)	7	Weak 1	Weak 0

8.3.2 デジタル出力ドライバ

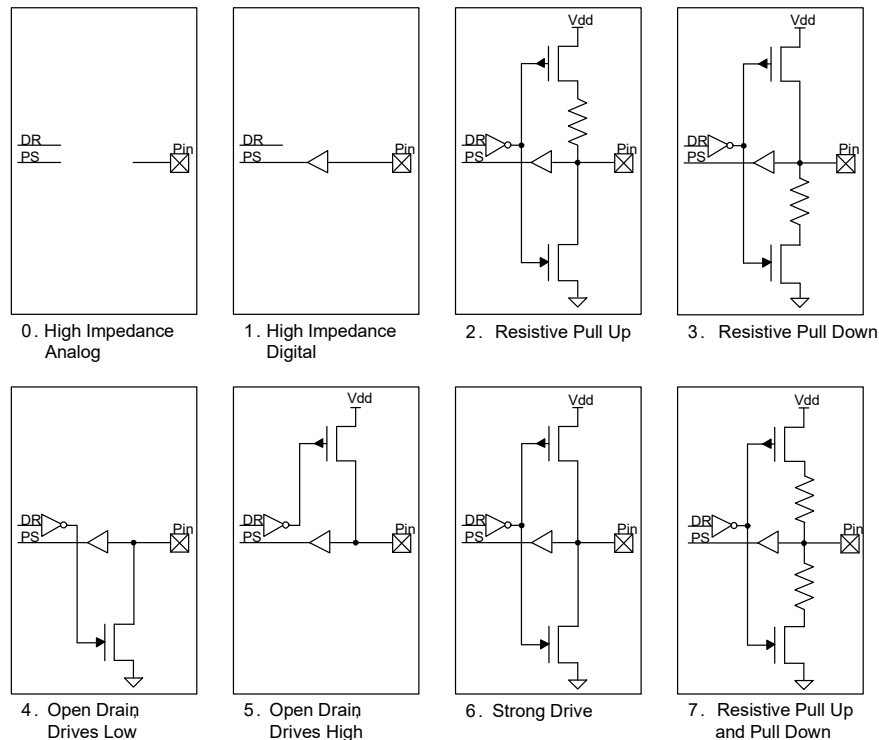
ピンはデジタル出力ドライバにより駆動されます。このドライバは異なる駆動モードを実装するための回路、およびデジタル出力信号用のスルーレート制御を含んでいます。ペリフェラルはHSIOMを介してデジタル出力ドライバに接続します。HSIOM ポート選択レジスタ (HSIOM_PORT_SELx) に書き込むことで特定のペリフェラルを選択できます。

PSoC4100S では、I/O は V_{DDD} 電源で駆動されます。各 GPIO ピンは、ピンの電圧が V_{DDD} 電源電圧を上回らないようにするための ESD ダイオードを備えています。ピンの電圧が I/O 電源電圧 V_{DDD} を上回らず、 V_{SSD} を下回らないことを確認してください。絶対最大および最小 GPIO 電圧については、[PSoC 4100S datasheet](#) および PSoC 4100S Plus 256 KB データシートを参照してください。デジタル出力ドライバは、ペリフェラルからの DSI 信号、または出力ピンに紐付いているデータ レジスタ(GPIO_PRTx_DR)を使用して有効化・無効化されます。データのペリフェラルソース選択についてや、コントロールソース選択を有効または無効にする方法については、[8.4 高速 I/O マトリックス](#)を参照してください。

8.3.2.1 駆動モード

各 I/O は、ポート構成レジスタ GPIO_PRTx_PC を使用して、8 つのドライブモードのいずれかに個別に構成できます。[Table 8-2](#) にドライブモードを示します。[Figure 8-2](#) は、8 つの駆動モードのそれぞれに基づくピンビューを示す簡略化された出力ドライバ図です。

Figure 8-3. I/O 駆動モードのブロック図



■ 高インピーダンス アナログ

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

■ 高インピーダンス デジタル

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

■ 抵抗プルアップまたは抵抗プルダウン

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

■ オープン ドレイン - HIGH に駆動および LOW に駆動

オープン ドレイン モードは、一方のデータ状態では高インピーダンスを、残り一方の状態ではストロング駆動を提供します。ピンはこれらのモードにおいて、デジタル入力またはデジタル出力として使用できます。従ってこのモードは双方向デジタル通信で広く使われています。信号が外部でプルダウンされた場合はオープン ドレイン HIGH 駆動モード、外部でプルアップされた場合はオープン ドレイン LOW 駆動モードを使用します。オープン ドレイン LOW 駆動モードの一般的な用途は、I²C バス信号ラインを駆動することです。

■ ストロングドライブ

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。通常の場合は、ストロングドライブモードピンを入力として使用しないでください。このモードは多くの場合、デジタル出力信号または外部トランジスタを駆動するために使用されます。

■ 抵抗プルアップおよび抵抗プルダウン

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

8.3.2.2 スルーレート制御

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

8.4 高速 I/O マトリックス

高速 I/O マトリックス (HSIOM) はデバイス内のペリフェラルへ GPIO を配線する高速スイッチのグループです。GPIO は複数の機能で共有されるため、HSIOM はピンを多重化し、ユーザにより選択された特定のペリフェラルに接続します。PSoC 4100S では、スマート I/O ブロックがポート 2 およびポート 3 ピンを HSIOM にブリッジします。PSoC 4100S Plus では、ポート 2 とポート 3 に加えてポート 1 がスマート I/O 機能をサポートします。他のポートは HSIOM に直接接続します。HSIOM_PORT_SELx レジスタは、ペリフェラルを選択するために提供されています。それは 1 本のピンが 4 ビットを持つ

各ポートで利用可能な 32 ビット幅レジスタです。このレジスタは、Table 8-3 示すように、ピンに対して最大 16 の異なるオプションを提供します。[Switching a GPIO Pin Connection between Analog and Digital Sources](#) を参照してください。

Table 8-3. PSoC 4100S HSIOM ポート設定

HSIOM_PORT_SELx ('x' はポート番号を示し、'y' はピン番号を示します)			
ビット	Name (SEL'y')	値	説明 (ピン'y' ソースを選択 ($0 \leq y \leq 7$))
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
	GPIO_DSI	1	出力はファームウェア制御ですが、OE は DSI から制御されます。
	DSI_DSI	2	出力と OE は両方とも DSI から制御
	DSI_GPIO	3	出力は DSI から制御されますが、OE はファームウェアで制御されます。
	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 ドライバは csd_charge 信号に接続 (ピンは依然として AMUXBUS-B に接続)
	ACTIVE_0	8	ピン特有のアクティブ ソース #0 (TCPWM 出力)
	ACTIVE_1	9	特定ピンのアクティブ ソース #1 (SCB-UART)
	ACTIVE_2	10	ピン特有のアクティブ ソース #2 (SAR ADC)
	ACTIVE_3	11	ピン固有のアクティブソース #3 (TCPWM 入力、SCB-UART、CAN)。
	DEEP_SLEEP_0	12	特定ピンのディープスリープ ソース #0 (LCD - COM)
	DEEP_SLEEP_1	13	特定ピンのディープスリープ ソース #1 (LCD - SEG)
	DEEP_SLEEP_2	14	ピン特有のディープスリープ ソース #2 (SCB-I ² C、SWD、LPCOMP)
	DEEP_SLEEP_3	15	特定ピンのディープスリープ ソース #3 (SCB-SPI)

注: アクティブソースとディープスリープソースはピンに依存します。各ピンでサポートされる機能の詳細については、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートの「ピン配置」セクションを参照してください。

8.5 スマート I/O

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

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

PSoC 4100S デバイスは、ポート 2 とポート 3 の 2 つのポートでスマート I/O をサポートし、PSoC 4100S Plus デバイスは、ポート 1、ポート 2、ポート 3 の 3 つのポートでスマート I/O をサポートします。レジスタの名称「PRGIO_PRT0」はポート 2 スマート I/O レジスタを示し、「PRGIO_PRT1」はポート 3 スマート I/O レジスタを示し、PRGIO_PRT2 はポート 1 スマート I/O レジスタを示します。一般的なスマート I/O レジスタを説明する際は「PRGIO_PRTx」命名法を使用します。

8.5.1 概要

スマート I/O ブロックは HSIOM と I/O ポート間にある信号経路に位置します。HSIOM は固定機能のペリフェラル および CPU から特定のポート ピンへと（逆の場合も同様）出力信号を多重化します。Figure 8-4 に示すように、スマート I/O ブロックはこの信号パスに配置され、ポートピンと HSIOM からの信号を処理できるブリッジとして機能します。

Figure 8-4. スマート I/O のインターフェース

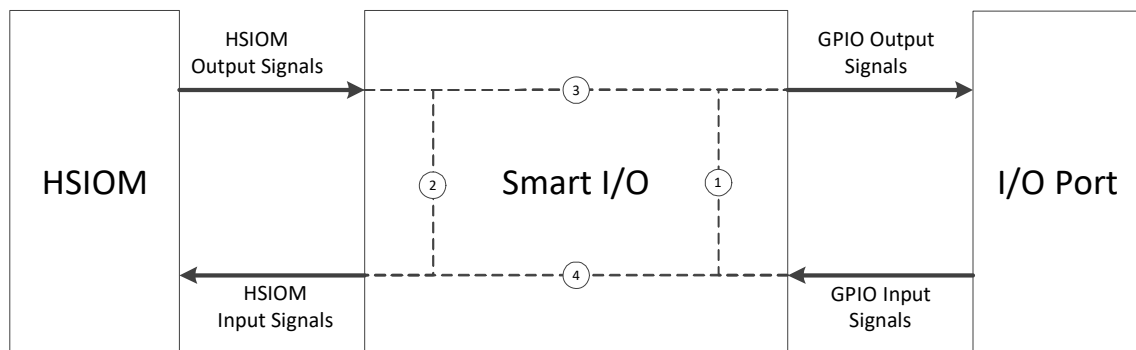


Figure 8-4 に示されたスマート I/O ブロックを介してサポートされる信号パスを次に示します。

1. I/O ポート信号で直接動作する自己完結型の論理関数を搭載
2. HSIOM で動作する自己完結型ロジック機能を実装する
3. HSIOM 出力信号で動作し、その出力信号を変更したり、変更された信号を I/O ポートの信号に配線
4. I/O ポートの信号で動作し、その信号を変更したり、変更された信号を HSIOM 入力信号に配線

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

8.5.2 ブロック コンポーネント

スマート I/O の内部論理には以下のコンポーネントが含まれます：

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

8.5.2.1 クロックとリセット

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

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

以下のクロック ソースが利用可能です：

- GPIO 入力信号「io_data_in[7:0]」：これらのクロックに関連するリセットはありません。
- HSIOM 出力信号「chip_data[7:0]」：これらのクロックに関連するリセットはありません。
- スマート I/O クロック (clk_prgio) は、ペリフェラルクロック分周器を使用してシステムクロック (clk_sys) から派生します。周辺機器のクロック分周器の詳細については、[クロック供給システム \(89 ページ\)](#) を参照してください。このクロックはアクティブ モードとスリープモードでのみ利用可能です。このクロックに関連するリセットは rst_sys_act_n か rst_sys_dpslp_n のいずれかです。これらのリセットはブロック同期状態がどのシステム電源モードでリセットされるかを判定します。例えば、rst_sys_act_n リセットはアクティブ モードにあるスマート I/O の同期機能向けであり、ディープスリープモードで有効化されます。
- 低周波数 (40kHz) のシステム クロック (clk_lf)：このクロックはディープスリープモードで利用可能です。このクロックに関連するリセットは rst_lf_dpslp_n です。

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

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

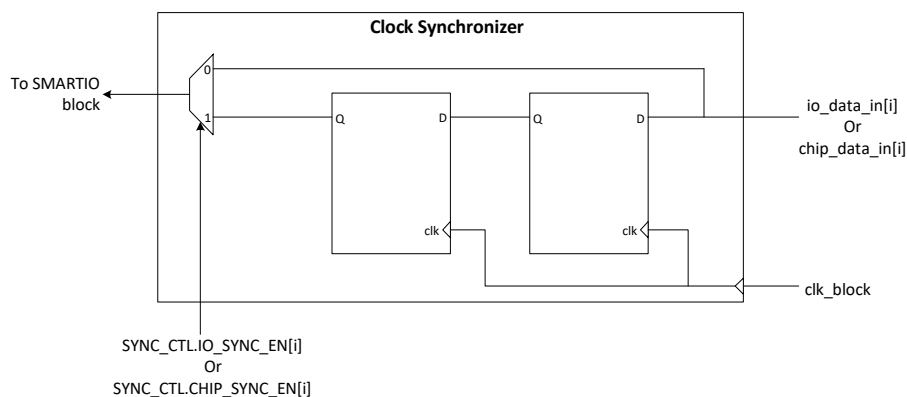
Table 8-4. クロックとリセット レジスタ制御

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

8.5.2.2 シンクロナイザー

各 GPIO 入力信号とデバイス入力信号 (HSIOM 入力) は非同期または同期で使用できます。信号を同期して使用するためには、Figure 8-5 に示すように、ダブルフリップフロップシンクロナイザーを両方の信号パスに配置して、信号をスマート I/O クロック (clk_block) に同期させます。各ピン / 入力の同期化は、PRGIO_PRT0_SYNC_CTL レジスタにある IO_SYNC_EN[i] ビット フィールド (GPIO 入力信号の場合) および CHIP_SYNC_EN[i] (HSIOM 信号の場合) をセットまたはクリアすることで有効化または無効化されます (「i」がピン番号を示す)。

Figure 8-5. スマート I/O クロック シンクロナイザー



8.5.2.3 LUT3

1 つのスマート I/O ブロックには 8 個のルックアップ テーブル (LUT3) コンポーネントが含まれます。LUT3 コンポーネントは 1 個の 3 入力 LUT と 1 個のフリップフロップから構成されます。各 LUT3 ブロックは 3 つの入力信号を持っており、PRGIO_PRTx_LUT_CTLy レジスタで設定されたコンフィギュレーションに基づいて出力を生成します (y は LUT3 番号を示

す)。各 LUT3 のコンフィギュレーションは、PRGIO_PRTx_LUT_CTLy レジスタにある 8 ビットのルックアップ ベクタ LUT[7:0] および 2 ビットのおペコード OPC[1:0] により定められます。8 ビットのベクタは 3 つの入力信号に対するルックアップ テーブルとして使用されます。2 ビットのおペコードはフリップフロップの使用を決めます。さまざまなおペコードの LUT3 構成を Figure 8-6 に示します。

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

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

PRGIO_PRTx_LUT_SELy レジスタの LUT_TR0_SEL[3:0] ビットは y 番目の LUT3 の tr0_in 信号を選択します。同様に、LUT_TR1_SEL[3:0] ビットおよび LUT_TR2_SEL[3:0] ビットはそれぞれ tr1_in 信号と r2_in 信号を選択します。詳細については、Table 8-5 を参照してください。

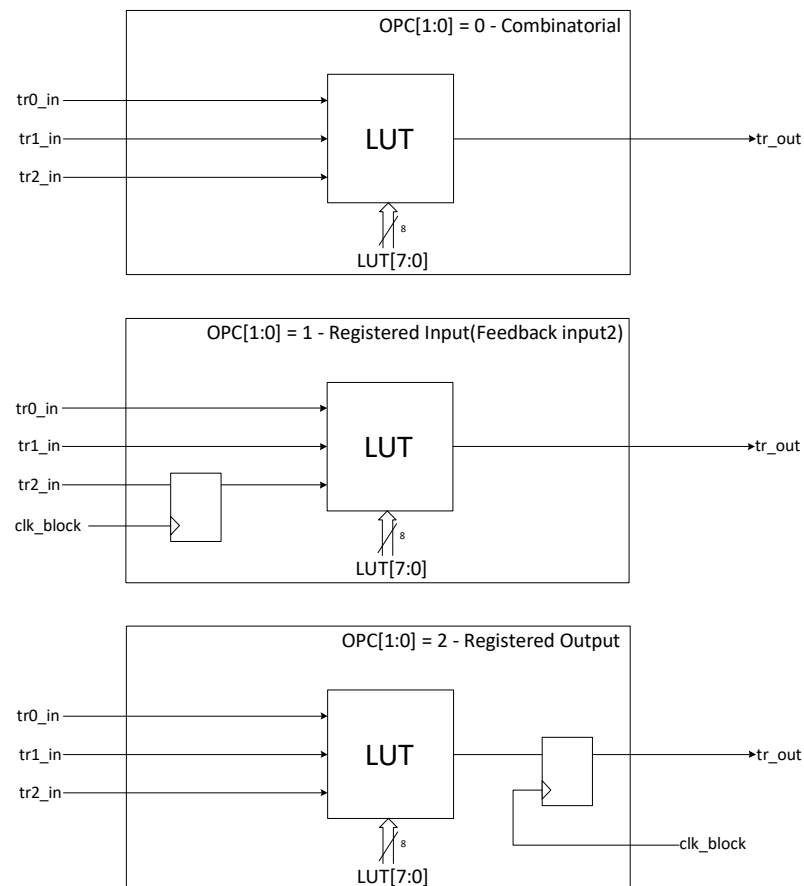
Table 8-5. LUT3 レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_LUT_CTLy [7:0]	LUT[7:0]	LUT のコンフィギュレーションである。LUT オペコード (LUT_OPC)、内部状態、および LUT 入力信号の tr0_in、tr1_in、tr2_in に応じて、LUT コンフィギュレーションは LUT 出力信号と次の連続状態を判定するために使用される。
PRGIO_PRTx_LUT_CTLy [9:8]	LUT_OPC[1:0]	Figure 8-6 に示すように、LUT オペコードは LUT の動作を指定します。
PRGIO_PRTx_LUT_SELy [3:0]	LUT_TR0_SEL [3:0]	LUT 入力信号「tr0_in」ソース選択： 「0」：データ単位出力 「1」：LUT 1 出力 「2」：LUT 2 出力 「3」：LUT 3 出力 「4」：LUT 4 出力 「5」：LUT 5 出力 「6」：LUT 6 出力 「7」：LUT 7 出力 「8」：chip_data[0] (LUT 0、1、2、3 の場合)；chip_data[4] (LUT 4、5、6、7 の場合) 「9」：chip_data[1] (LUT 0、1、2、3 の場合)；chip_data[5] (LUT 4、5、6、7 の場合) 「10」：chip_data[2] (LUT 0、1、2、3 の場合)；chip_data[6] (LUT 4、5、6、7 の場合) 「11」：chip_data[3] (LUT 0、1、2、3 の場合)；chip_data[7] (LUT 4、5、6、7 の場合) 「12」：io_data_in[0] (LUT 0、1、2、3 の場合)；io_data_in[4] (LUT 4、5、6、7 の場合) 「13」：io_data_in[1] (LUT 0、1、2、3 の場合)；io_data_in[5] (LUT 4、5、6、7 の場合) 「14」：io_data_in[2] (LUT 0、1、2、3 の場合)；io_data_in[6] (LUT 4、5、6、7 の場合) 「15」：io_data_in[3] (LUT 0、1、2、3 の場合)；io_data_in[7] (LUT 4、5、6、7 の場合)

Table 8-5. LUT3 レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_LUT_SELy [11:8]	LUT_TR1_SEL [3:0]	LUT 入力信号「tr1_in」ソース選択 : 「0」:LUT 0 出力 「1」:LUT 1 出力 「2」:LUT 2 出力 「3」:LUT 3 出力 「4」:LUT 4 出力 「5」:LUT 5 出力 「6」:LUT 6 出力 「7」:LUT 7 出力 「8」:chip_data[0] (LUT 0、1、2、3 の場合) ; chip_data[4] (LUT 4、5、6、7 の場合) 「9」:chip_data[1] (LUT 0、1、2、3 の場合) ; chip_data[5] (LUT 4、5、6、7 の場合) 「10」:chip_data[2] (LUT 0、1、2、3 の場合) ; chip_data[6] (LUT 4、5、6、7 の場合) 「11」:chip_data[3] (LUT 0、1、2、3 の場合) ; chip_data[7] (LUT 4、5、6、7 の場合) 「12」:io_data_in[0] (LUT 0、1、2、3 の場合) ; io_data_in[4] (LUT 4、5、6、7 の場合) 「13」:io_data_in[1] (LUT 0、1、2、3 の場合) ; io_data_in[5] (LUT 4、5、6、7 の場合) 「14」:io_data_in[2] (LUT 0、1、2、3 の場合) ; io_data_in[6] (LUT 4、5、6、7 の場合) 「15」:io_data_in[3] (LUT 0、1、2、3 の場合) ; io_data_in[7] (LUT 4、5、6、7 の場合)
PRGIO_PRTx_LUT_SELy [19:16]	LUT_TR2_SEL [3:0]	LUT 入力信号「tr2_in」ソース選択。エンコーディングは LUT_TR1_SEL と同様

Figure 8-6. スマート I/O LUT3 コンフィギュレーション



8.5.2.4 データ ユニット

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

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

- 定数「0x00」
- io_data_in[7:0]
- chip_data_in[7:0]
- PRGIO_PRTx_DATA レジスタの DATA[7:0] ビット フィールド

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

Table 8-6. データ ユニット レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_DU_CTL[2:0]	DU_SIZE[2:0]	データ ユニットのサイズ / 幅 (ビット単位) は DU_SIZE+1 である。例えば、DU_SIZE が 7 の場合、データ ユニットの幅が 8 ビットである
PRGIO_PRTx_DU_CTL[11:8]	DU_OPC[3:0]	データ ユニット オペコードはデータ ユニットの動作を特定する： 「0」：カウントアップ 「1」：カウントダウン 「2」：カウントアップ ラップ 「3」：カウントダウン ラップ 「4」：カウントアップ / ダウン 「5」：カウントアップ / ダウン ラップ 「6」：右回転 「7」：右にシフト 「8」：DATA0 & DATA1 「9」：Majority3 「10」：DATA1 と一致 その他：未定義
PRGIO_PRTx_DU_SEL[3:0]	DU_TR0_SEL[3:0]	データユニット入力信号「tr0_in」ソース選択： 「0」：定数 '0' 「1」：定数 '1' 「2」：データユニット出力 「10-3」：LUT 7-0 出力 その他：未定義
PRGIO_PRTx_DU_SEL[11:8]	DU_TR1_SEL[3:0]	データユニット入力信号「tr1_in」ソース選択。エンコーディングは DU_TR0_SEL と同様

Table 8-6. データ ユニット レジスタ制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_DU_SEL[19:16]	DU_TR2_SEL[3:0]	データユニット入力信号「tr2_in」ソース選択。エンコーディングはDU_TR0_SELと同様
PRGIO_PRTx_DU_SEL[25:24]	DU_DATA0_SEL[1:0]	データユニット入力データ「data0_in」ソース選択: 「0」: 定数「0」 「1」: data[7:0] 「2」: gpio[7:0] 「3」: DU Reg
PRGIO_PRTx_DU_SEL[29:28]	DU_DATA1_SEL[1:0]	データユニット入力データ「data1_in」ソース選択。エンコーディングはDU_DATA0_SELと同様
PRGIO_PRTx_DATA[7:0]	DATA[7:0]	データ ユニットの入力データ ソース

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

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

```
// The following is shared by all operations.
mask = (2 ^ (DU_SIZE+1) - 1)
data_eql_data1_in = (data & mask) == (data1_in & mask));
data_eql_0        = (data & mask) == 0);
data_incr         = (data + 1) & mask;
data_decr         = (data - 1) & mask;
data0_masked      = data_in0 & mask;

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

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

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

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

```

    else if (tr1_in) data <= data_eql_0      ? data0_masked: data_decr;

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

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

// ROR operation: rotates data right and LSB is sent out. The data for rotation is taken from
// data0.
Comb:tr_out = data[0];
Reg:  data <= data;
    if (tr0_in)      data <= data0_masked;
    else if (tr1_in) {
        data <= {0, data[7:1]} & mask; //Shift right operation
        data[du_size] <= data[0]; //Move the data[0] (LSB) to MSB
    }

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

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

// SHR_EQL operation: performs the same operation as SHR. Instead of shift-out, the output is
// a comparison result (data0 == data1).
Comb:tr_out = data_eql_data1_in;
Reg:  data <= data;

```

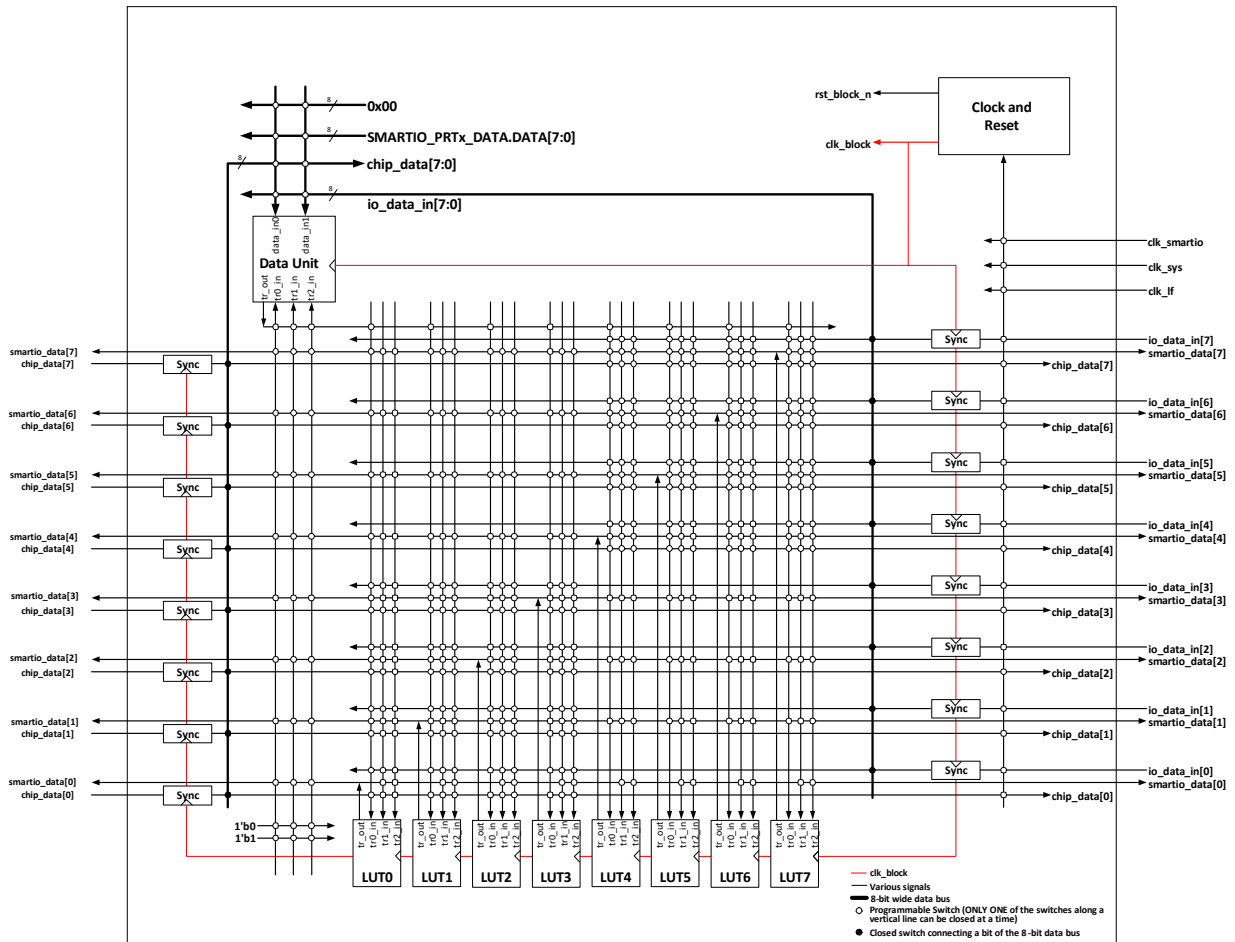
```
if      (tr0_in) data      <= data0_masked;
else if (tr1_in) {
    data      <= {0, data[7:1]} & mask;
    data[du_size] <= tr2_in;
}

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

8.5.3 配線

スマート I/O ブロックは、ブロックに / から信号を配線したり、ブロック内の様々なコンポーネント間に配線したりするために使用される数多くのスイッチを含みます。その配線スイッチは PRTGIO_PRTx_LUT_SELy と PRGIO_PRTx_DU_SEL レジスタを通じて処理されます。詳細については、*PSoC 4100S: PSoc 4 Registers TRM* を参照してください。スマート I/O 内部ルーティングを Figure 8-7 に示します。この図では、LUT7 ~ LUT4 は io_data_in/chip_data[7] ~ io_data_in/chip_data[4] で動作し、LUT3 ~ LUT0 は io_data_in/chip_data[3] ~ io_data_in/chip_data[0] で動作することに注意してください。

Figure 8-7. スマート I/O の配線



8.5.4 動作

スマート I/O ブロックは、次のように構成および操作する必要があります。レジスタ制御の詳細については、Table 8-7 を参照してください。

1. “ブロック コンポーネント” (74 ページ) で説明するように、ブロックを有効にする前に、すべてのコンポーネントを構成し、ルーティングを選択する必要があります。
2. コンポーネントのコンフィギュレーションや配線の他に、所望の動作を実現できるように幾つかのブロック レベルの設定を適切にコンフィギュレーションすることが必要です。
 - a. バイパス制御：スマート I/O 経路は、PRGIO_PRTx_CTL レジスタの BYPASS[i] ビット フィールドを設定することにより特定の GPIO 信号に対してバイパスされます。BYPASS[7:0] ビットフィールドでビット 'i' が設定されている場合、i 番目の GPIO 信号は HSIOM 信号パスに直接バイパスされます。その信号パスにはスマート I/O ロジックが存在しません。これは、スマート I/O 機能が選択 I/O でのみ必要とされる場合には特に有用です。
 - b. パイプライン トリガー モード：LUT3 入力マルチプレクサと LUT3 コンポーネント自体はいかなる組み合わせループも含んでいません。同様に、データ ユニットも組み合わせループを含みません。ただし、LUT3 が他の LUT3 またはデータ ユニットとやり取りをする場合は、意図しない組み合わせループが発生することは有り得ます。この制限を受けないようにするために、PRGIO_PRTx_CTL レジスタの PIPELINE_EN ビット フィールドが使用されます。このビット フィールドが設定される場合、すべての出力 (LUT3 とデータ ユニット) は他のコンポーネントに分岐する前に登録 (フロップ) されます。PIPELINE_EN ビットがクリアされると、出力はフロップされなくなります。
3. スマート I/O ブロックは希望の機能に応じてコンフィギュレーションされた後、PRGIO_PRTx_CTL レジスタの ENABLED ビット フィールドを設定することで有効化されます。ENABLED ビット フィールドがクリアされると、スマート I/O ブロックはバイパス モードに入ります。そのモードにおいては、GPIO 信号は直接 HSIOM 信号により制御されます (逆の場合も同様)。スマート I/O ブロックをコンフィギュレーションしなければなりません。即ち、レジスタの更新中にグリッチを防止するために、ブロックを有効化する前にすべてのレジスタ設定を更新しなければなりません。

Table 8-7. スマート I/O ブロック制御

レジスタ [BIT_POS]	ビット名	説明
PRGIO_PRTx_CTL[25]	PIPELINE_EN	パイプライン レジスタの有効化： '0': 無効 (レジスタはバイパスされます) '1': 有効
PRGIO_PRTx_CTL[31]	ENABLED	スマート I/O を有効にします。スマート I/O が完全に構成されている場合のみ、'1' に設定する必要があります。 '0': 無効 (信号はバイパスされます; BYPASS[7:0] が 0xFF であるかのように動作します)。無効の場合、ブロック (データ ユニットと LUT) リセットがアクティブ化される。 ブロックが無効の場合： - 低消費電力を確保するには、PIPELINE_EN レジスタフィールドを '1' に設定する必要があります。 - 低消費電力を保証するために、CLOCK_SRC レジスタフィールドを 20730 (クロックは一定の '0') に設定する必要があります。 '1': 有効。有効にすると、ブロックのリセットが無効になり、ブロックが完全に機能するようになるまで、3 つの「clk_block」クロックサイクルが必要です。このアクションにより、ブロックが完全に機能しているときに、I/O pins' の入力シンクロナイザーの状態が確実にフラッシュされます。
PRGIO_PRTx_CTL[7:0]	BYPASS[7:0]	スマート I/O のバイパスであり、1 ビットは 1 つの I/O ピンに対応する：BYPASS[i] は I/O ピン 'i' に対応する。ENABLED が '1' の場合、このフィールドが使用されます。ENABLED が '0' の場合、このフィールドは使用されず、スマート I/O は常にバイパスされます。 '0': バイパスなし (スマート I/O が信号パスに存在) '1': バイパス (スマート I/O は信号パスにありません)

8.6 電源投入時の I/O 状態

電源投入時に、すべての GPIO は高インピーダンス アナログ状態になり、入力バッファは無効にされます。実行中、GPIO は関連レジスタに書き込むことで設定することができます。デバッグ アクセスポート (DAP) の接続 (SWD ライン) をサポートするピンは電源投入時に常に SWD ラインとして有効化されることに注意してください。ただし、DSI 接続を無効にするか、HSIOM を介して汎用目的で再構成できます。ただし、この再構成は、デバイスが起動してコードの実行を開始した後にのみ行われます。

8.7 省電力モードでの動作

Table 8-8 は、低電力モードでの GPIO のステータスを示します。

Table 8-8. 低消費電力モードの GPIO

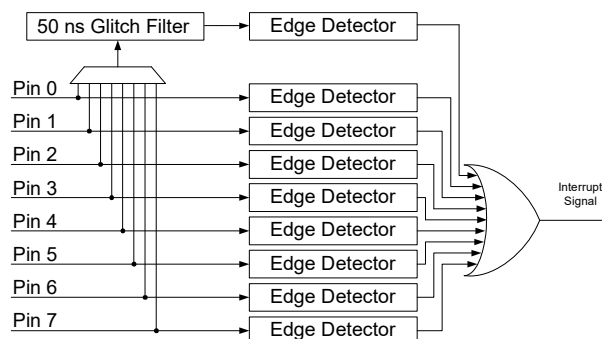
低消費電力モード	状態
Sleep (スリープ)	- GPIO が有効であり、CapSense、CTBm、SAR ADC、TCPWM、SCB などのペリフェラル、およびスリープモードで動作できる低消費電力コンパレータにより駆動される。 - 入力バッファが有効であるため、あらゆる I/O 上の割り込みを CPU をウェイクアップするために使用できる - AMUXBUS 接続は利用可能である
Deep-Sleep (ディープスリープ)	- I²C と SPI ピンを除いて、GPIO 出力ピンの状態はラッチされ、凍結状態のままである。SCB (I²C と SPI) ブロックはディープスリープモードで動作し、アドレス一致時または SPI スレーブ選択イベント時に CPU をウェイクアップすることができる。低電力コンパレータはその専用ピンから信号を受信し、CPU をウェイクアップすることができる。CTBm も専用ピンを持っており、このモードで機能する。 - 入力バッファはこのモードでも有効であり、ピン割り込みが機能する - AMUXBUS 接続は利用不可である

8.8 割り込み

PSoC 4 デバイスにおいては、すべてのポート ピンが割り込みを生成する能力を有します。Figure 8-2 に示すように、ピン信号は GPIO エッジ検出ブロックを介して割り込みコントローラにルーティングされます。

Figure 8-8 は、GPIO エッジ検出ブロックのアーキテクチャを示します。

Figure 8-8. GPIO エッジ検出ブロックのアーキテクチャ



エッジ検出器は各ピンに搭載されます。この検出器により、リコンフィギュレーションすることなく立ち上がりエッジ、立ち下がりエッジ、そしてその両方のエッジを検出することが可能です。Table 8-9 に示すように、エッジ検出器は、ポート割り込み構成レジスタ GPIO_PRTx_INTR_CFG の EDGE_SEL ビットに書き込むことによって構成されます。

Table 8-9. エッジ検出器のコンフィギュレーション

EDGE_SEL	設定
00	割り込みが無効
01	立ち上がりエッジの割り込み
10	立ち下がりエッジの割り込み
11	両エッジの割り込み

ピン以外に、エッジ検出器はグリッチフィルタ出力にも存在します。このフィルタはポートのいずれかのピンで使用されます。[Table 8-10](#) に示すように、ピンは GPIO_PRTx_INTR_CFG レジスタの FLT_SEL フィールドに書き込むことによって選択されます。

Table 8-10. グリッチフィルタの入力選択

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

ポートのエッジ検出器の出力は OR され、割り込みコントローラ (CPU サブシステムの NVIC) に配線されます。従って、1 つのポートに対して 1 つの割り込みベクタしかありません。ピンの割り込みについては、どのピンが割り込みを発生したかを調べる必要があります。ポート割り込みステータス レジスタの GPIO_PRTx_INTR を読み出すことで情報が得られます。このレジスタには、どのピンが割り込みをトリガーしたかという情報だけでなく、ピンのステータス情報も含まれており、CPU が単一の読み出し動作で両方の情報を読み出すことを可能にします。また、このレジスタはより重要な役割があります。それは割り込みをクリアすることです。該当するステータス ビットに「1」を書き込みことで割り込みをクリアできます。割り込みステータス ビットをクリアすることが重要です。そうしないと、割り込みは単一のトリガーに対して繰り返して発生するか、または複数のトリガーに対して 1 回だけ応答します。それらの関連事項は本章の後半で説明されます。さらに、該当するポートで割り込みが発生した時にポート割り込み制御ステータス レジスタが読み出されると、割り込みが適切に検出されない原因にもなることに注意してください。そのため GPIO 割り込みを使用する場合、コードの他の部分ではなく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。[Table 8-11](#) に、ポート割り込みステータス レジスタのビットフィールドを示します。

Table 8-11. ポート割り込みステータス レジスタ

GPIO_PRTx_INTR	説明
0000b ~ 0111b	ピン 0 からピン 7 への割り込みステータス。対応するビットに「1」を書き込むと、割り込みがクリアされます
1000b	グリッチフィルタからの割り込みステータス
10000b ~ 10111	ピン 0 ~ ピン 7 のステータス
11000b	グリッチフィルタの出力ステータス

エッジ検出器ブロックの出力は、[Figure 6-3 \(53 ページ\)](#) に示す割り込みソースマルチプレクサにルーティングされ、レベルおよび立ち上がりエッジ検出のオプションを提供します。レベル オプションが選択される場合、ポート割り込みステータス レジスタ ビットが設定されている限り、割り込みは繰り返してトリガーされます。立ち上がりエッジ検出オプションが選択される場合、ポート割り込みステータス レジスタがクリアされていなければ、割り込みは 1 回だけトリガーされます。従って、エッジ検出ブロックが使用される場合、割り込みステータス ビットをクリアすることが重要です。

8.9 ペリフェラルの接続

8.9.1 ファームウェアで制御される GPIO

ファームウェア制御の GPIO の HSIOM 設定については、[Table 8-3](#) 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 レジスタ以外に、GPIO_PRTx_DR_SET、GPIO_PRTx_DR_CLR および GPIO_PRTx_INV という 3 個のレジスタも提供されており、それぞれのレジスタは他のピンに影響を与えることなくポートの特定のピンの出力データをセット、クリア、そして反転します。これらのレジスタに「1」を書き込むことでデータをセット、クリア、反転します。「0」を書き込む場合はピンの状態に対して何の影響も与えません。

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

8.9.2 アナログ IO

LPCOMP、SARMUX や CTBm 等の低インピーダンス配線が必要なアナログ 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。専用ピンは性能の向上に役立っており、これらのアナログ リソースを使用する際は他のピンよりも優先されます。これらの専用ピンの詳細については、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および [PSoC 4100S Plus 256 KB 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 として構成するには、高インピーダンスアナログモードで構成し ([Table 8-2](#) を参照)、特定のアナログリソースでそれぞれの接続を有効にする必要があ

ります。これは該当のアナログ リソースに関連付けられるレジスタを介して実行されます。

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

8.9.3 LCD 駆動

あらゆる GPIO は、LCD コモンまたはセグメントを駆動する能力を有しています。HSIOM_PORT_SELx レジスタは、LCD 駆動ピンを選択するために使用されます。詳細については、[LCD 直接駆動 \(223 ページ\)](#) を参照してください。

8.9.4 CapSense

CSD をサポートするピンは CapSense ウィジェット (ボタン、スライダー要素、タッチパッドまたは近接センサー等) としてコンフィギュレーションすることができます。CapSense には、外部タンクコンデンサとシールドラインも必要です。[Table 8-12](#) は、CapSense に必要な GPIO および HSIOM 設定を示します。詳細については、[CapSense \(283 ページ\)](#) を参照してください。

Table 8-12. CapSense 設定

CapSense ピン	GPIO 駆動モード (GPIO_PRTx_PC)	デジタル入力バッファの設定 (GPIO_PRTx_PC2)	HSIOM 設定
Sensor (センサー)	High-Impedance Analog (高インピーダンス アナログ)	Disable Buffer (バッファを無効化)	CSD_SENSE
Shield (シールド基板)	High-Impedance Analog (高インピーダンス アナログ)	Disable Buffer (バッファを無効化)	CSD_SHIELD
CMOD (normal operation) (CMOD (通常動作))	High-Impedance Analog (高インピーダンス アナログ)	Disable Buffer (バッファを無効化)	AMUXBUS A または CSD_COMP
CMOD (GPIO precharge, only available in select GPIO) (CMOD (GPIO プリチャージ、選択 GPIO のみ利用可能))	High-Impedance Analog (高インピーダンス アナログ)	Disable Buffer (バッファを無効化)	AMUXBUS B または CSD_COMP
CSH TANK (GPIO precharge, only available in select GPIO) (CSH TANK (GPIO プリチャージ、選択 GPIO のみ利用可能))	High-Impedance Analog (高インピーダンス アナログ)	Disable Buffer (バッファを無効化)	AMUXBUS B または CSD_COMP

8.9.5 シリアル通信ブロック (SCB)

UART、I²C および SPI としてコンフィギュレーション可能な SCB はピンへの専用接続を持っています。[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および [PSoC 4100S Plus 256 KB データシート](#)、またはこれらの専用ピンの詳細をご覧ください。UART と SPI モードが使用される場合、入力ピンを高インピーダンス状態に維持するために、SCB はそのピンに対してデジタル出力バッファ駆動モードを制御します。つまり、SCB ブロックは、SPI マスターとしてコンフィギュレーションされる場合は UART Rx ピンと MISO ピンの出力バッファを無効化し、SPI スレーブとしてコンフィギュレーションされる場合は MOSI と選択ラインの出力バッファを無効化します。この機能は GPIO_PRTx_PC レジスタを用いて行われる駆動モードの設定を上書きします。

8.9.6 タイマ / カウンタ / パルス幅変調器 (TCPWM) ブロック

TCPWM はピンへの専用接続を持っています。これらの専用ピンの詳細については、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートを参照してください。開始や停止などの TCPWM ブロック入力がピンから取得される場合、TCPWM ブロックはその入力ピンの出力バッファを無効にするため、駆動モードは High-Z デジタルのみであることに注意してください。

8.10 レジスタ

Table 8-13. I/O レジスタ

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

注 : GPIO レジスタ名の 'x' はポート番号を示します。例えば、GPIO_PTR1_DR はポート 1 の出力データ レジスタを示します。スマート I/O レジスタ名の「x」はスマート I/O ポート番号を示します。スマート I/O のポート番号と実際のポート番号が異なる場合があります。詳細については、[8.5 スマート I/O \(73 ページ\)](#) を参照してください。

9. 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 リソースがあります。

- 2 つの内部クロック 源 :
 - 24 ~ 48MHz 内部主発振器 (IMO) $\pm 2\%$ の精度 (全周波数範囲に渡り、調整あり)
 - 40kHz 内部低速発振器 (ILO) $\pm 60\%$ の精度 (トリム可能、IMO を使って校正可能)
- 3 つの外部クロック ソース :
 - I/O 端子からの信号を使用し生成する外部クロック (EXTCLK)
 - 外部 4 ~ 33 MHz 水晶発振器 (ECO)
 - 外部 32 kHz 時計水晶発振器 (WCO)
- 最大 48 MHz の高周波クロック (HFCLK)、IMO、ECO、外部クロック、または PLL から選択
- 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 によって供給される最大 48MHz のシステム クロック (SYSCLK) の専用プリスケアラ
- 12 個の 16 ビット周辺クロック分周器
- 正確なクロック生成のための 6 つの分数分周器 (5 つの 16.5 分数分周器と 1 つの 24.5 分数分周器)
- 19 のデジタルおよびアナログ周辺クロック

9.1 ブロック図

Figure 9-1 は、PSoC 4 デバイスのクロックシステムの一般的なビューを示します。

Figure 9-1. クロック供給システムのブロック図

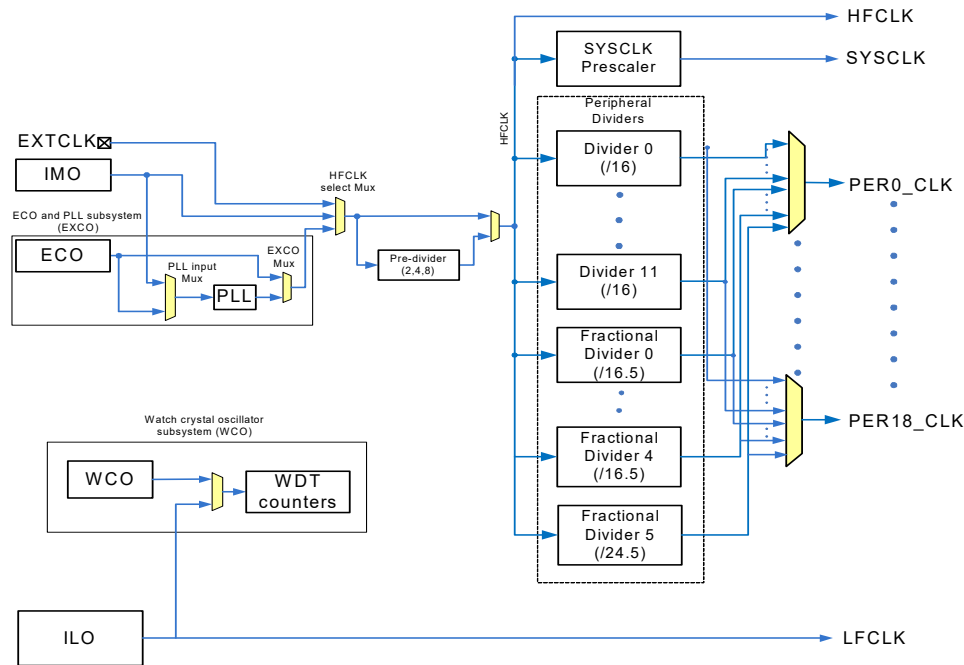


Figure 9-1 に示すように、デバイスの 6 つのクロックソースは IMO、ECO、EXTCLK、WCO、ILO です。IMO または ECO からクロックソースを取得するように構成できる PLL があります。PLL の出力周波数は 104MHz にもなることがあります。PLL の出力周波数は 104MHz まで高くなる可能性があります、すべてのブロックが 48MHz より高い周波数を受け入れるわけではありません。詳細については、[9.2.5 PLL \(93 ページ\)](#) を参照してください。HFCLK マルチプレクサは、EXTCLK、ECO、PLL (EXCO マルチプレクサを介して) または IMO から HFCLK ソースを選択します。HFCLK の周波数は最大 48MHz です。

9.2 クロック ソース

9.2.1 内部主発振器 (IMO)

内部主発振器 (IMO) は、アクティブおよびスリープ モードの間で、主クロック ソースとして利用可能な正確で高速な内部 (水晶なし) 発振器です。これはデバイスでデフォルトのクロック ソースです。その周波数は、 ± 2 パーセントの精度で、24MHz から 48MHz の間で 4MHz ステップで変更できます。

IMO 周波数は、[Table 9-1](#) に示すように、CLK_IMO_SELECT レジスタを使用して変更されます。デフォルトの周波数は 24MHz です。

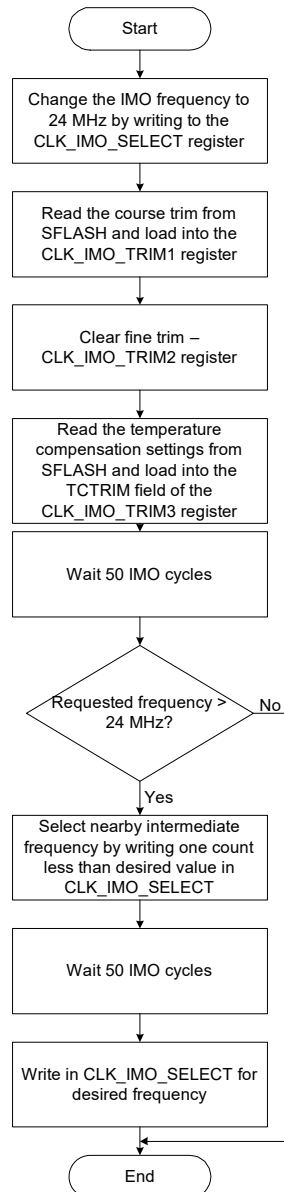
Table 9-1. IMO 周波数

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

正確な IMO 周波数を取得するために、トリムレジスタが用意されています。CLK_IMO_TRIM1 は 120 kHz のステップサイズの粗調整を提供し、CLK_IMO_TRIM2 は 15 kHz のステップサイズの微調整を提供し、CLK_IMO_TRIM3 の TCTRIM フィールドは温度補償を提供します。調整設定は CLK_IMO_SELECT によって選択することができるすべての周波数用に、製造中に行われます。これらの調整設定は SFLASH に格納されます。

調整設定はデバイス スタートアップ時にロードされます；しかし、ファームウェアは新しいトリム値のロードおよび実行時に周波数を変更することができます。Figure 9-2 のアルゴリズムに従って、IMO 周波数を変更します。

Figure 9-2. IMO 周波数の変更



9.2.1.1 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。

9.2.1.2 プログラミング クロック (36MHz)

フラッシュをプログラムするには、IMO を 48MHz に設定する必要があります。これはフラッシュのプログラム / 消去のタイミング目的に、チャージ ポンプを駆動するために使用されます。

9.2.2 内部低速発振器 (ILO)

内部低速発振器は、外部部品が必要なく動作し、公称 40kHz の安定したクロックを出力します。ILO の消費電力は比較的小さく、精度も良くありません。より高い精度の高周波クロックを使用して定期的に較正し、精度を向上させることができます。ILO はすべての電源モードで使用できます。デバイスで、ILO はシステムの低周波数クロック LFCLK として使用されます。ILO は低周波数のクロックを生成するための比較的不正確 (過電圧と過温度のときに $\pm 60\%$) な発振器です。動作中に IMO を使用して校正する場合、ILO は安定した温度および電圧で $\pm 10\%$ の精度です。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。

9.2.3 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 の指定した端子の信号から生成されます。このクロックは IMO のかわりにシステム高周波数クロック HFCLK の供給源として使用される場合があります。外部クロックの許容される周波数範囲は 1 ~ 48MHz です。デバイスは常に IMO を使用して起動しますが、外部クロックはユーザ モードで有効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。

特定の端子を EXTCLK の入力として設定する場合、デジタル入力バッファを有効にするために、端子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。詳細については、[I/O システム \(66 ページ\)](#) を参照してください。

9.2.4 外部水晶発振器 (ECO)

PSoC 4 デバイスは、外部の 4 MHz から 33.33 MHz の水晶を駆動する発振器を含んでいます。このクロック ソースは、PSoC の発振回路を使用して構成されます。この回路は、PSoC の外部クリスタルピンに実装する必要がある外部クリスタルを採用しています。

ECO は、ECO_CONFIG.CLK_EN (ビット 0) および ECO_CONFIG.ENABLE (ビット 31) レジスタのビットフィールドを使用して有効にできます。

9.2.4.1 ECO トリミング

ECO は、 $f = 4 \text{ MHz} - 33.333 \text{ MHz}$ の公称周波数範囲仕様で、さまざまな水晶およびセラミック共振器をサポートします。水晶メーカーは通常、パラメータの数値、つまり最大駆動レベル (DL)、等価直列抵抗 (ESR)、および並列負荷容量 (C_L) を提供します。これらのパラメータを使用して、トランスコンダクタンス (g_m) と最大ピーク間値 (V_{pp}) を計算できます。

最大ピークツーピーク値：

$$V_{PP} = 2 \times \frac{\sqrt{\frac{2 \times DL}{ESR}}}{4\pi \times f \times C_L}$$

相互コンダクタンス：

$$g_m = 4 \times 5 \times (2\pi \times f \times C_L)^2 \times ESR$$

$V_{pp} < 0.4 \text{ V}$ は ECO ではサポートされていません。同様に、 g_m を 18 mA/V 以上にすることはできません。

ATRIM および WDTRIM 設定は、発振器出力の振幅のトリムを制御します。振幅トリム (ATRIM) は、AGC が有効 (ECO_CONFIG.AGC_EN = 1) の場合に水晶駆動レベルを設定します。V_{PP} < 2 V に対して AGC を有効にし、他のすべてのケースでは無効にします。V_{PP} 値に基づいて、ATRIM 値と WDTRIM 値は、Table 9-2 に示すように設定されます。

Table 9-2. ATRIM および WDTRIM 設定

VPP	ATRIM	WDTRIM
0.4 V ≤ V _{PP} < 0.5 V	0x00	0x00
V _{PP} < 0.6 V	0x01	0x00
V _{PP} < 0.7 V	0x02	0x01
V _{PP} < 0.8 V	0x03	0x01
V _{PP} < 0.9 V	0x04	0x02
V _{PP} < 1.025 V	0x05	0x02
V _{PP} < 1.15 V	0x06	0x03
V _{PP} < 1.275 V	0x07	0x03

GTRIM は、アンプゲインのトリムを設定し、Table 9-3 に示すように計算された g_m に基づいて設定されます。

Table 9-3. GTRIM 設定

G _m	GTRIM
0 mA/V < g _m ≤ 4.5 mA/V	0x00
4.5 mA/V < g _m ≤ 9 mA/V	0x01
9 mA/V < g _m ≤ 13.5 mA/V	0x02
13.5 mA/V < g _m ≤ 18 mA/V	0x03

FTRIM と RTRIM は、フィルタ特性のトリムを設定し、Table 9-4 に示すように計算された g_m に基づいて設定されます。

Table 9-4. FTRIM および RTRIM 設定

公称周波数 f (MHz)	RTRIM	FTRIM
f > 30 MHz	0x00	0x00
30 MHz ≥ f > 24 MHz	0x01	0x01
24 MHz ≥ f > 17 MHz	0x02	0x02
17 MHz ≥ f	0x03	0x03

9.2.5 PLL

PSoC デバイスは、PLL を実装します。PLL は、高度に構成可能な周波数合成を提供します。PLL は主に 2 つの設定 P と Q を使用して構成されます。P は PLL のフィードバック分周器設定で、Q は基準クロック分周器です。したがって、PLL の出力周波数は基準クロックの P/Q 倍です。

PLL の P および Q は、PLL_CONFIG によって構成されます。このレジスタでは、PLL_CONFIG[7:0] が P 値で、PLL_CONFIG[13:8] が Q 値です。フィードバックカウンタ (P は 8 ビットの分解能) と基準カウンタ (Q は 6 ビットの分解能) のこの整数値を変更することにより、PLL は入力基準クロック周波数 (clk_ref) から多数の出力周波数を合成できます。

9.2.6 時計用水晶発振器 (WCO)

PSoC デバイスは、32.768kHz のとき計用水晶を駆動するための発振器が搭載されています。ILO と同様、WCO はすべての消費電力モードで使用可能です。このクロックは低消費電力を持っています。これにより、ディープスリープ モードなどの低消費電力モード動作に最適です。WCO は、WCO_CONFIG レジスタの ENABLE ビットにより有効 / 無効化されます。

WCO は、WCO_CONFIG[0] ビットをセットすることで低消費電力モードに移行します。一方、ブロックはデバイスがディープスリープ モードに移行する時にのみ、低消費電力モード移行が発生する自動モードに移行できます。このモードは WCO_CONFIG[1] をセットすることで有効にします。ブロックは WCO_CONFIG[0] をセットすることで低消費電力モードに移行される場合、自動モードは無視されることに注意してください。切り替え時には、WCO 出力周波数に若干の変動が生じることがあります。したがって、自動モードは、RTC などの高精度アプリケーションには推奨されません。

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

IMO は WCO にロックする機能をサポートします。WCO には IMO クロックの測定 / 比較および IMO トリミング回路があります。WCO は $\pm 1\%$ 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 の IMO トリミングロジックは、WCO_CONFIG の DPLL_ENABLE ビットを使用して有効にできます。この機能を使用する場合、ユーザファームウェアは、WCO の有効化と DPLL_ENABLE イベントの間に最小 500ms の時間があることを確認する必要があります。

9.3 クロック分配

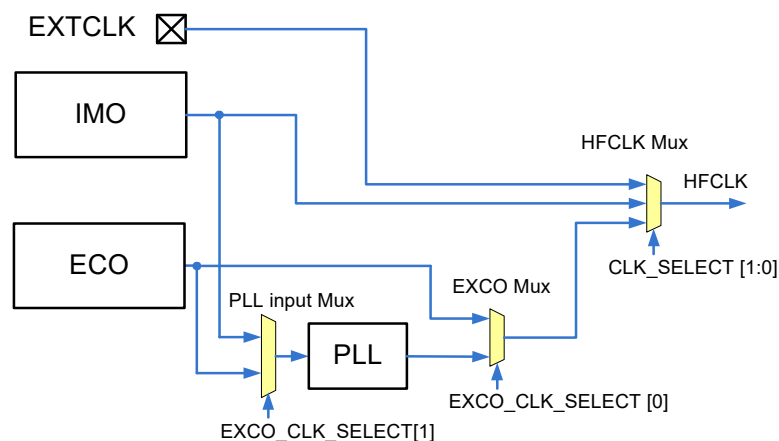
Figure 9-1 に示すように、PSoC 4 クロックが開発され、デバイス全体に分散されます。この分配コンフィギュレーションオプションを次に示します。

- HFCLK 入力選択
- PLL 入力選択
- LFCLK 入力選択
- SYSCLK プリスケアラ構成
- ペリフェラル分周器のコンフィギュレーション

9.3.1 HFCLK および PLL 入力の選択

Figure 9-3 に、HFCLK および PLL の選択オプションを示します。

Figure 9-3. HFCLK および PLL 選択オプション



PLL 入力には、IMO および ECO 出力信号を選択するマルチプレクサがあります。PLL の入力選択は、EXCO_CLK_SELECT[1] レジスタにあります。

HFCLK の場合、これは 2 段階の選択です。2 つのソース ECO と PLL は、EXCO_CLK_SELECT[0] で選択が構成されている EXCO マルチプレクサを使用して最初に多重化されます。HFCLK マルチプレクサと呼ばれる 2 番目のマルチプレクサは、EXCO 出力、IMO、および EXTCLK から選択します。この選択は、CLK_SELECT[1:0] レジスタで構成されます。

PSoC 4100S Plus デバイスでは、IMO が PLL への参照として使用され、ECO が無効になっている場合、または ECO クリスタルがボードにマウントされていない場合、ファームウェアは EXCO_PGM_CLK レジスタの CLK_ECO ビットをトグルして PLL を開始する必要があります。ビットを切り替える手順は、cy_boot コンポーネントの CySysClkPllSetSource または CySysClkWriteHfclkDirect API に示されています。独自のドライバを実装する場合は、システムドライバのクロック API にトグルプロシージャを実装する必要があります。EXCO_PGM_CLK レジスタについては、[PSoC 4100S Plus: PSoC 4 Registers TRM](#) で説明されています。

EXCO_PGM_CLK.ENABLE ビットを設定した後、0 と 1 を交互に書き込んで、EXCO_PGM_CLK.CLK_ECO ビットを 5 回切り替えます。トグル後、0 を書き込んで PLL を開始することにより、EXCO_PGM_CLK.ENABLE ビットをクリアします。

低電力動作：この手順は、デバイスがディープスリープ電力モードを終了するときを使用する必要があることに注意してください。デバイスがディープスリープモードに入る前に、システムクロック (HFCLK/SYSCLK) ソースを PLL から IMO に変更する必要があります。ディープスリープモードからウェイクアップした後、システムクロックソースを IMO から PLL に戻すことができます。

9.3.2 HFCLK 入力の選択

PSoC 4 の HFCLK には、IMO、ECO、PLL、および EXTCLK の 4 つの入力オプションがあります。ECO と PLL がステージで最初に選択され、ECO と PLL サブシステムのクロックを取得します。この入力、[Table 9-6](#) で説明されているように、ECO.CLK_SELECT register's の CLK_SELECT ビットを使用して選択されます。HFCLK 入力は、[Table 9-5](#) で説明されているように、CLK_SELECT レジスタの HFCLK_SEL ビットを使用して選択されます。

Table 9-5. HFCLK 入力選択ビット HFCLK_SEL

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

Table 9-6. CLK_SELECT 入力選択ビット

名称	説明
CLK_SELECT	ECO および PLL サブシステムのクロック選択 0: ECO。ECO 出力を HFCLK のソースとして使用 1: PLL。HFCLK のソースとして PLL 出力を使用

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

9.3.3 LFCLK 入力の選択

PSoC 4100S および PSoC 4100S Plus 256 KB デバイスで LFCLK のソースになることができるのは ILO のみです。

9.3.4 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 にすることができます。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 $2^7 = 128$ までの範囲内の 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、Table 9-7 で説明されているように、レジスタ CLK_SELECT ビット SYSCLK_DIV を使用して設定されます。プリスケーラは、最初は 1 で除算するように構成されています。

Table 9-7. SYSCLK プリスケーラ分周値ビット SYSCLK_DIV

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

9.3.5 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 には 18 個のクロック分周器があり、これには 12 個の 16 ビットクロック分周器、5 つの 16.5 ビット分数クロック分周器、および 1 つの 24.5 ビット分数クロック分周器が含まれます。分数クロック分周器により、クロック除数には 0 ~ 31/32 の分数があり得ます。分数分周器の出力周波数は次の式で計算されます: $F_{out} = F_{in} / (INT16_DIV + (FRAC5_DIV/32))$ 。たとえば、整数の除算値が 2 の 16.5 分周器 (INT16_DIV = 3、FRAC5_DIV = 0) は、48 MHz HFCLK から 16 MHz クロックを生成する信号を生成します。整数の除算値が 3 の 16.5 分周器 (INT16_DIV = 3、FRAC5_DIV = 0) は、48 MHz HFCLK から 12 MHz クロックを生成する信号を生成します。2 の整数分周値 (INT16_DIV = 3) と 16 の分数分周器 (FRAC5_DIV = 16) を持つ 16.5 分周器は、48 MHz HFCLK から 13.7 MHz クロックを生成する信号を生成します。13.7MHz クロックの周期はすべて等しい訳ではありません。その半数が 3HFCLK サイクルで、残りの半数 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。

12 の整数クロック分周器のそれぞれの分周値は PERI_DIV_16_CTLx レジスタで構成され、5 つの 16.5 ビットの分数クロック分周器は PERI_DIV_16_5_CTLx レジスタで構成されます。Table 9-8 と Table 9-9 は、これらのレジスタの構成を説明しています。24.5 ビットの分数分周器は、PERI_DIV_24_5_CTL0 レジスタを使用して構成されます。Table 9-10 に、これらのレジスタの構成を示します。

Table 9-8. 非分数ペリフェラル クロック分周器コンフィギュレーション レジスタ PERI_DIV_16_CTLx

ビット	名称	説明
0	ENABLE_x	分周器が有効。HW は、ENABLE コマンドの結果として、このフィールドを '1' に設定します。HW は、DISABLE コマンドの結果として、このフィールドを '0' に設定します。
23:8	INT16_DIV_x	(1+INT16_DIV) の値で分周。範囲 [1、65536] の整数除算が可能です。

Table 9-9. 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。HW は、ENABLE コマンドの結果として、このフィールドを '1' に設定します。HW は、DISABLE コマンドの結果として、このフィールドを '0' に設定します。
7:3	FRAC5_DIV_x	(FRAC5_DIV/32) 分数比分周。[0、31/32] 範囲内の分数分周が可能。 クロック周期の一部が他のクロック周期に比べて 1「clk_hf」サイクル長くなる場合があるため、分数分周によりクロック ジッタが生じることに注意してください
23:8	INT16_DIV_x	(1+INT16_DIV) の値で分周。[1、65536] 範囲内の分周が実行可能

Table 9-10. フラクショナルペリフェラルクロック分周器構成レジスタ PERI_DIV_24_5_CTL0

ビット	名称	説明
0	ENABLE_0	分周器が有効。ハードウェアは、ENABLE コマンドの結果としてこのフィールドを「1」に設定し、DISABLE コマンドの結果として「0」に設定します。
7:3	FRAC5_DIV_0	(FRAC5_DIV/32) 分数比分周。[0、31/32] 範囲内の分数分周が可能。一部のクロック周期が他のクロック周期よりも 1 clk_hf サイクル長くなる可能性があるため、分数除算の結果としてクロックジッタが発生することに注意してください。
31:8	INT24_DIV_0	(1 + INT24_DIV) による整数除算。範囲 [1、16,777,216] の整数除算が可能です。

各分周器は PERI_DIV_CMD レジスタによって有効にできます。このレジスタは、すべての 16 個の整数分周器と 4 個の分数分周器用のコマンド 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を次に示します。

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	Disabse															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ます。このフィールドは、16 ビット整数除算器の場合は '1'、16.5 ビット分数除算器の場合は '2'、24.5 ビット分数除算器の場合は '3' です。'

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 です。分数分周器の場合、このフィールドの値は 0 ~ 3 です。SEL_TYPE = 63 と SEL_TYPE = 3 である場合、いずれの分周器も指定しません。

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

48 MHz の HFCLK と、12 MHz の分周クロック A と 8 MHz の分周クロック B の必要性を検討してください。クロック A は、16 ビットの整数分周器 0 を使用し、HF_CLK ((PA_SEL_TYPE、PA_SEL_DIV) に揃えることによって作成されます。は (3、63)) で、DIV_16_CTL0.INT16_DIV は 3 です。クロック B は整数分周器 1 を使用し、クロック A に合わせて作成され ((PA_SEL_TYPE、PA_SEL_DIV) は (1、0))、DIV_16_CTL1.INT16_DIV は 5 です。2 つのクロック周期の最小公倍数は 12 HFCLK サイクルであるため、これによりクロック B とクロック A の位相が揃うことが保証され、クロック A と B は 12 HFCLK サイクルごとに整列します。クロック B はクロック A と位相を揃えますが、それ自体の分周値のために、依然として HFCLK を基準クロックとして使用することに注意してください。

PSoC の各ペリフェラルブロックには、固有のペリフェラルクロック (PERI#_CLK) が関連付けられています。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。

Table 9-11 と Table 9-12 は、マルチプレクサ出力の対応するペリフェラルブロックへのマッピングを示します (Figure 9-1 を参照)。すべてのペリフェラルクロック分周器は、それぞれの PERI_PCLK_CTLx レジスタを使用して特定のペリフェラルにマップできます。

Table 9-11. PSoC 4100S および PSoC 4100S Plus デバイスのペリフェラルクロックマルチプレクサの出力マッピング

PERI#_CLK	ペリフェラル
0	SCB0
1	SCB1
2	SCB2
3	SCB3
4	SCB4
5	CSD
6	TCPWM0
7	TCPWM1
8	TCPWM2
9	TCPWM3
10	TCPWM4
11	TCPWM5
12	TCPWM6
13	TCPWM7
14	SmartIO
15	SmartIO
16	SmartIO
17	LCD
18	SAR ADC

Table 9-12. PSoC 4100S Plus 256 KB デバイスペリフェラルクロックマルチプレクサ出力マッピング

PERI#_CLK	ペリフェラル
0	SCB0
1	SCB1
2	SCB2
3	SCB3
4	SCB4
5	CSD
6	TCPWM0
7	TCPWM1
8	TCPWM2
9	TCPWM3
10	TCPWM4
11	TCPWM5
12	TCPWM6
13	TCPWM7
14	SmartIO Port2

Table 9-12. PSoC 4100S Plus 256 KB デバイスペリフェラルクロックマルチプレクサ出力マッピング (continued)

PERI#_CLK	ペリフェラル
15	SmartIO ポート 3
16	SAR ADC0
17	LCD
18	SAR ADC1

Table 9-13. 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ペリフェラル クロック clock_x のマルチプレクサ出力に接続される。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: 24.5 (分数) クロック分周器

9.4 低消費電力動作

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

9.5 レジスタ一覧

Table 9-14. クロック供給システムのレジスタ一覧

レジスタ名	説明
CLK_IMO_TRIM1	IMO トリム レジスタ - このレジスタは粗調整用の IMO 調整値を含む
CLK_IMO_TRIM2	IMO トリム レジスタ - このレジスタは微調整用の IMO 調整値を含む
CLK_IMO_TRIM3	IMO 調整レジスタ - このレジスタは IMO 用の温度補正および IMO 周波数の粗調整と微調整のステップ サイズを調整するためのトリム設定を含む
PWR_BG_TRIM1	バンドギャップ調整レジスタ - これらのレジスタはバンドギャップ基準の調整を制御し、デバイスの基準電圧調整を可能にする
PWR_BG_TRIM2	
CLK_ILO_CONFIG	ILO コンフィギュレーション レジスタ - このレジスタは ILO コンフィギュレーションを制御
CLK_IMO_CONFIG	IMO コンフィギュレーション レジスタ - このレジスタは IMO コンフィギュレーションを制御
CLK_SELECT	クロック選択 - このレジスタはクロック ツリー コンフィギュレーションを制御し、異なる信号源からシステムクロックを選択
EXCO_CLK_SELECT	ECO および PLL サブシステムのクロック選択 - このレジスタは、ECO および PLL サブシステムの PLL 入力および出力の選択を制御します。
ECO_CONFIG	このレジスタは、ECO 構成を制御します。
PLL_CONFIG	このレジスタは、PLL 構成を制御します。
WCO_CONFIG	WCO イネーブル。このレジスタは外部のとき計用水晶発振器を有効 / 無効にする
PERI_DIV_16_CTLx	周辺クロック分周器制御レジスタ - これらのレジスタは、周辺クロック分周器を構成し、整数分周値を設定し、分周器を有効または無効にします。
PERI_DIV_16_5_CTLx	周辺クロック分数分周器制御レジスタ - これらのレジスタは、周辺クロック分周器を構成し、分数分周値を設定し、分周器を有効または無効にします。
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
PERI_DIV_24_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

10. 電源と監視



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

- 内部レギュレータへの 1.80V から 5.50 V の電源入力
- 1.71V ~ 1.89V¹ の直接供給

各種電力モードをサポートするために 2 つの内部レギュレータが用意されています :- アクティブ デジタル レギュレータ、ディープスリープ レギュレータ。

10.1 ブロック図

Figure 10-1. 電力システムのブロック図

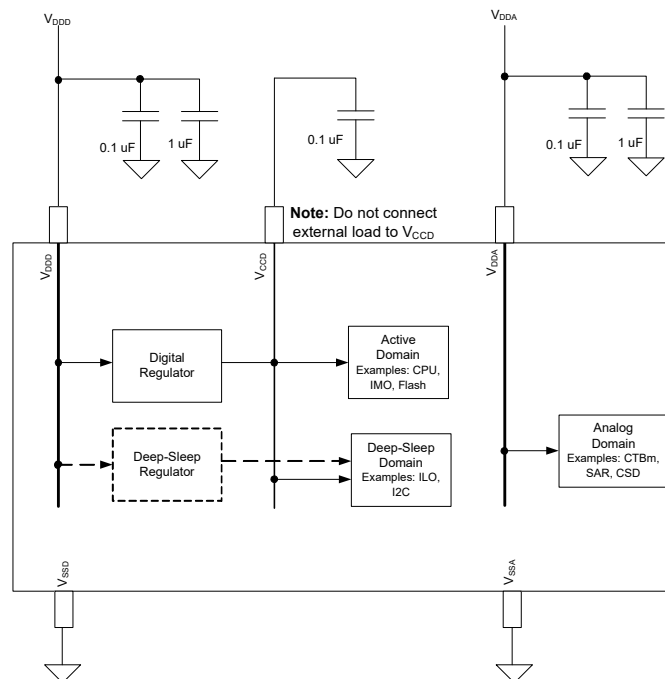


Figure 10-1 に、電源システム図とすべての電源ピンを示します。システムは、アクティブ モードで動作するデジタル回路用レギュレータがあります。アナログ レギュレータはありません。アナログ回路は V_{DDA} 入力で直接動作します。ディープスリープ モード用に個別レギュレータがあります。

電源電圧範囲は 1.71V ~ 5.5 V で、すべての機能と回路がその範囲で動作します。デバイスには非安定化外部電源供給と安定化外部電源供給の 2 つの異なる電源供給モードがあります。

1. システム電源が 1.80V?1.89 V の範囲にある場合、直接電源と内部レギュレータオプションの両方を使用できます。ユーザのシステム機能に応じて選択できます。直接供給オプションの場合、デバイスが損傷するため、供給電圧が 1.89V を超えることはできないことに注意してください。レギュレータがオフになるため、内部レギュレータオプションの場合は 1.80V を下回ってはなりません。

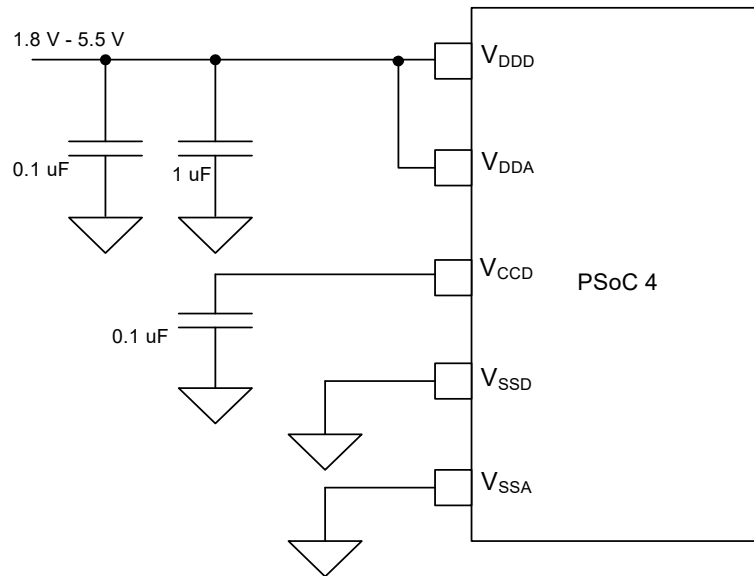
10.2 電源供給のシナリオ

下図はデバイスに電源供給される様々な方法を示します。

10.2.1 1.8V ~ 5.5V の単一の非安定化電源

1.8 V ~ 5.5 V の電源を非安定化電源入力として使用する場合は、[Figure 10-2](#) に示すように接続する必要があります。

Figure 10-2. 単一の安定化された V_{DD} 電源

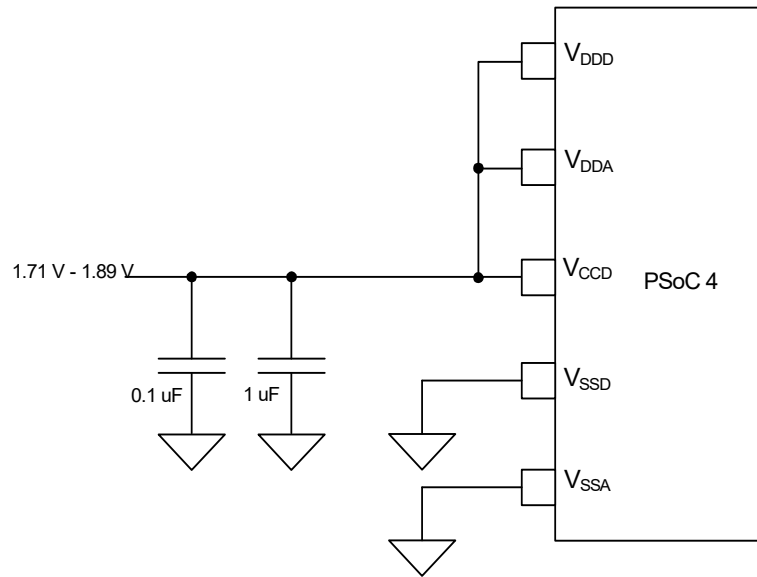


このモードでは、デバイスは、1.8V から 5.5V の範囲内のどこにでもあることができる外部電源によって給電されます。この範囲は、バッテリー駆動の動作向けにも設計されています。たとえば、チップは 3.5V から始まり 1.8V まで動作するバッテリーシステムから電力を供給できます。このモードでは、内部レギュレータが内部ロジックを提供します。 V_{CCD} 出力は 0.1 μ F 外部セラミック コンデンサを経由し、グラウンドにバイパスされる必要があります。

V_{DD} からグラウンドへのバイパスコンデンサも必要です。この周波数範囲のシステムの一般的な方法は、1 μ F から 10 μ F の範囲のバルクコンデンサを小さなセラミックコンデンサ (たとえば、0.1 μ F) と並列に使用することです。これらは単に経験則であり、重要なアプリケーションに対しては、設計のためと最適なバイパスを得るために、PCB レイアウト、リード インダクタンス、バイパス コンデンサ寄生容量をシミュレートする必要があることに留意してください。

10.2.2 直接 1.71V ~ 1.89 V の安定化電源

直接電源の構成で、 V_{CCD} および V_{DD} は互いに短絡させ、1.71V ~ 1.89V の電源に接続されます。[Figure 10-3](#) に示すように、この安定化電源はデバイスに接続する必要があります。

Figure 10-3. 単一の安定化されていない V_{DD} 電源


このモードでは、 V_{CCD} および V_{DD} ピンは互いに短絡し、バイパスします。内部レギュレータはファームウェアで無効にする必要があります。詳細は、[10.3.1.1 アクティブ デジタル レギュレータ \(103 ページ\)](#) を参照してください。

10.3 動作原理

[Figure 10-1](#) のレギュレータは、デバイスのさまざまなドメインに電力を供給します。すべてのコア レギュレータは、 V_{DD} 電源端子から電力を取り込みます。アナログ回路は V_{DDA} 入力から直接電源供給されます。

10.3.1 レギュレータの概要

10.3.1.1 アクティブ デジタル レギュレータ

Table 10-1. 電力モード別レギュレータの状態

モード	アクティブ デジタル レギュレータ	ディープスリープ レギュレータ
Deep-Sleep (ディープスリープ)	オフ	オン
Sleep (スリープ)	オン	オン
Active (アクティブ)	オン	オン

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ピン (V_{CCD}) に接続されており、外部デカップリングコンデンサ (1 μ F X5R) が必要です。

1.8V 以下の電源の場合、 V_{CCD} を直接供給する必要があります。この場合、[Figure 10-3](#) に示すように、 V_{CCD} と V_{DD} を互いに短絡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。この操作は直接電源モードでの消費電力を削減します。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のモードで使用できます

10.3.1.2 ディープスリープレギュレータ

このレギュレータはディープスリープモードで電源供給される ILO、WCO および SCB (I²C/SPI) の回路および低消費電力コンパレータに電源供給します。ディープスリープレギュレータはすべての消費電力モードで使用できます。アクティブおよびスリープのモードで、このレギュレータのメイン出力はアクティブデジタルレギュレータ (V_{CCD}) に接続されます。

10.4 電圧監視

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

10.4.1 パワーオンリセット (POR)

POR 回路は電源を投入してから電源電圧が上昇する間にリセットパルスが発生します。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、以後無効です。

10.4.1.1 電圧低下検出 (BOD)

BOD 回路は、デバイスをリセットすることで、動作中またはデータ保持中の回路を安全ではない電源供給条件から保護します。BOD 回路は V_{CCD} 電圧を監視します。BOD 回路は、電圧変動が安全な動作に必要な最小 V_{CCD} 電圧を下回るとリセットを生成します (詳細については、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートを参照してください)。システムは電源電圧が有効な範囲に回復するまでリセット状態を継続します。

信頼性の高いデバイス動作を保証するために、ウォッチドッグタイマはすべてのデザインで使用する必要があります。ウォッチドッグタイマは、CPU 機能を損なう可能性がある異常な電圧低下状態に対して保護します。詳細については、[ウォッチドッグタイマ \(111 ページ\)](#) を参照してください。

10.5 レジスタ一覧

Table 10-2. 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 - このレジスタはデバイスの電力モードおよびレギュレータの動作を設定

11. チップ動作モード



PSoC[®] 4 は 4 つの異なるモードでファームウェアを実行することができます。これらのモードは異なるハードウェア特権レベルでフラッシュおよび ROM での異なる場所からの実行を決定します。これらの内、3 つのモードだけは、エンド 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をデバッグするために他のモードと排他的に使用されます。

PSoC 4 の動作モードを次に示します。

- Boot (ブート)
- User (ユーザ)
- Privileged (特権)
- Debug (デバッグ)

11.1 ブート

ブートモードは、デバイスの SROM にハードコードされた命令によってデバイスが構成される動作モードです。デバイスがデバッグ取得シーケンスを受信していない場合、このモードはリセットの終了後に開始されます。ブートモードは特権モードです。このモードでは割込みが無効にされるため、ブートファームウェアは割り込まれずにデバイスをセットアップすることができます。ブート モード中、電源投入時の適切な動作を確保するために、ハードウェア トリム設定をフラッシュからロードします。ブート処理終了後、デバイスはユーザ 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。

11.2 ユーザ

ユーザ モードはフラッシュから通常のユーザ ファームウェアが実行される動作モードです。ユーザモードでは SROM からコードを実行できません。このモードでのファームウェアの実行には、PSoC Creator IDE によって自動的に生成されたファームウェアとユーザが作成したファームウェアが含まれます。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。

11.3 特権

特権モードは、デバイス ROM に格納されている特別なサブルーチンの実行を可能にする操作モードです。これらのサブルーチンはユーザが変更することはできず、中断または監視することを意図していない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。システムコールの実行方法の詳細については、“[システム コールの実行](#)” (297 ページ) 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 モードに戻ります。

11.4 デバッグ

デバッグ モードは PSoC 4 の操作パラメータの観察を可能にする動作モードです。このモードは開発中にファームウェアをデバッグするために使用されます。規定の接続時間内に SWD デバッガ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。デバッグモードでは、PSoC Creator や Arm MDK などの IDE がファームウェアをデバッグできます。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、[デバッグ インターフェース \(288 ページ\)](#) を参照してください。

保護モードの詳細については、[デバイスセキュリティ \(63 ページ\)](#) を参照してください。

12. 電力モード



PSoC[®] 4 はアプリケーションでの平均消費電力を最小にするため 3 つの電力モードを提供します。消費電力の大きい順にした電力モードを以下に示します。

- Active (アクティブ)
- Sleep (スリープ)
- Deep-Sleep (ディープスリープ)

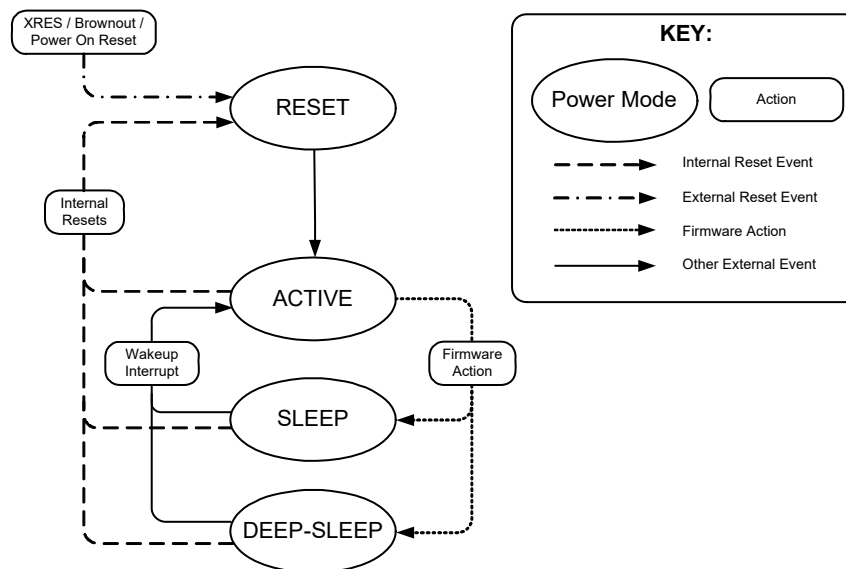
Active、Sleep、Deep-Sleep は、Arm CPU がサポートする標準の Arm 定義の電力モードです。

各種電力モードの消費電力は以下の方法で制御されます：

- ペリフェラルを有効 / 無効
- 内部レギュレータの電源供給をオン / オフ
- クロック ソースの電源供給をオン / オフ
- PSoC4 の他の部分の電源オン / オフ

Figure 12-1 は、さまざまな電力モードとそれらの間の可能な遷移を示します。

Figure 12-1. 電力モードの状態遷移図



注：ディープスリープパワーモードのアームの用語は 'SLEEPDEEP' です。

Table 12-1 は、PSoC 4 が提供する電力モードを示します。

Table 12-1. PSoC 4 電力モード

電力モード	説明	遷移条件	復帰ソース	アクティブなクロック	復帰動作	使用可能なレギュレータ
Active (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)	他の電力モードから復帰、内部と外部がリセット、電圧低下、パワー オン リセット	適用できません	すべて (プログラム可能)	該当なし	すべてのレギュレータが使用可能。外部のレギュレータ機能を使用している場合は、アクティブ デジタル レギュレータを無効にすることが可能
Sleep (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	任意の有効な割込み	すべて (プログラム可能) CPU クロックを除く	割込み	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
Deep-Sleep (ディープスリープ)	すべての内部電源はディープスリープ レギュレータにより駆動される。IMO と高速ペリフェラルはオフになる。低速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	ILO (40 kHz), WCO (32 kHz)	割込み	ディープスリープレギュレータ

Table 12-1 に記載されているウェイクアップソースに加えて、外部リセット (XRES) と電圧低下リセットにより、デバイスは任意の電力モードからアクティブモードになります。

12.1 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。

12.2 スリープモード

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

12.3 ディープスリープモード

ディープスリープ モードでは、CPU、SRAM および高速な回路がデータ保持状態に入ります。高速クロック (HFCLK と SYSCLK を含む) は無効です。オプションで、内部低周波数 (40kHz) 発振器と時計用水晶発振器 (WCO) はオンのままで、低周波数周辺機器は動作し続けます。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。消費電力と CTBm 帯域幅の詳細については、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートを参照してください。

使用可能なウェイクアップソースは、[Table 12-3](#) にリストされています。

12.4 電力モードの概要

Table 12-3 は、各低電力モードで使用可能な周辺機器を示します。Table 12-3 は、各電力モードで使用可能なウェイクアップソースを示します。

Table 12-2. 利用可能なペリフェラル

ペリフェラル	Active (アクティブ)	Sleep (スリープ)	Deep-Sleep (ディープスリープ)
CPU	利用可能	保持 ^a	保持
SRAM	利用可能	保持	保持
High-speed peripherals (高速ペリフェラル)	利用可能	利用可能	保持
Low-speed peripherals (低速ペリフェラル)	利用可能	利用可能	利用可能 (任意)
Internal main oscillator (IMO) (内部主発振器 (IMO))	利用可能	利用可能	利用不可
Internal low-speed oscillator (ILO, 40 kHz) (内部低速発振器 (ILO、40kHz))	利用可能	利用可能	利用可能 (任意)
Asynchronous peripherals (非同期ペリフェラル, 内部クロックで動作しないペリフェラル)	利用可能	利用可能	利用可能
Power-on-reset, Brownout detection (パワーオン リセット、電圧低下検出)	利用可能	利用可能	利用可能
Analog mux bus connection (アナログ マルチプレクサ バス接続)	利用可能	利用可能	利用可能
GPIO output state (GPIO 出力ステート)	利用可能	利用可能	利用可能

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

Table 12-3. 復帰ソース

電力モード	復帰ソース	復帰動作
Sleep (スリープ)	任意の有効な割込みソース	割込み
	任意のリセットソース	リセット
Deep Sleep (ディープスリープ)	GPIO 割込み	割込み
	I2C アドレス一致	割込み
	ウォッチドッグタイマ	割込み / リセット
	低電力コンパレータ	割込み
	CTBm	割込み

注 :Table 12-3 に記載されているウェイクアップソースに加えて、外部リセット (XRES) と電圧低下リセットにより、デバイスは任意の電力モードからアクティブモードになります。XRES と電圧低下は完全なシステムの再起動をトリガーします。凍結された GPIO を含むすべての状態を失います。この場合、復帰の原因をデバイスが再起動した後で読み込むことはできません。

12.5 低消費電力モードへの移行および復帰

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

スリープおよびディープスリープモードへの移行は、CM0P システムコントロールレジスタ (CM0P_SCR) のフラグ SLEEPDEEP によって制御されます。

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

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

スリープおよびディープスリープモードでは、選択可能なペリフェラルが利用可能で (Table 12-3 を参照)、ファームウェアは関連する割り込みを有効または無効にできます。有効にされた割り込みにより、低消費電力モードからアクティブ モードに復帰させることができます。さらに、任意の RESET はアクティブ モードにシステムを復帰させます。詳細については、[割り込み \(51 ページ\)](#) および [リセット システム \(120 ページ\)](#) を参照してください。

12.6 レジスター一覧

Table 12-4. 電力モードのレジスター一覧

レジスタ名	説明
CM0P_SCR	システム制御—システム制御データを設定または回復
PWR_CONTROL	電力モード制御—デバイスの電力モード オプションを設定し、現時点の状態を監視

13. ウォッチドッグタイマ



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

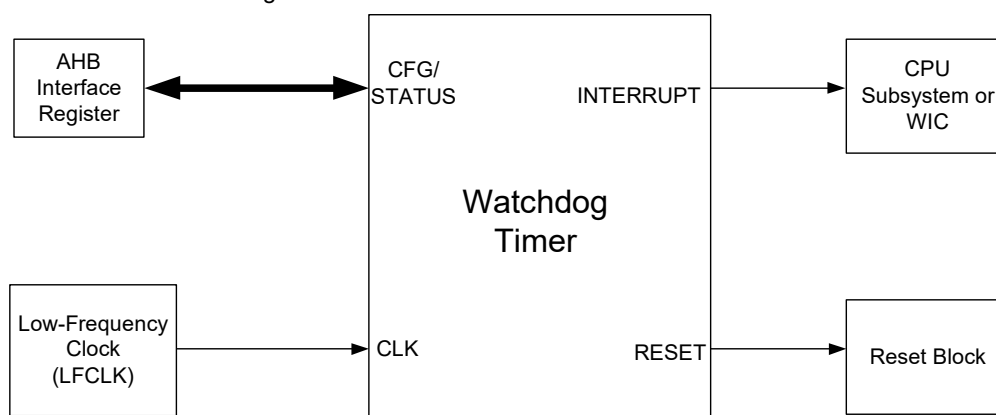
13.1 特長

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込み / 復帰トリガーを生成可能
- 16 ビット フリーランニング カウンタ機能

13.2 ブロック図

Figure 13-1. ウォッチドッグ タイマのブロック図



13.3 動作原理

ファームウェアで定期的にクリアされない限り、WDT は 3 番目の WDT 一致イベントでデバイスのハードウェア リセットをアサートします。WDT 割込みは、最大 2048ms のプログラマブルな周期を持ちます。WDT は、最大 16 ビットの分解能を有するフリーランニングラップアラウンドアップカウンタです。この分解能は設定可能であり、本節の後半で説明します。

WDT_COUNTER レジスタは WDT のカウント値を提供します。WDT は、WDT_COUNTER のカウント値が WDT_MATCH レジスタに格納された一致値と等しい場合に割り込みを生成しますが、カウントを '0' にリセットしません。代わりに、WDT はオーバーフローするまで (解像度が 16 ビットに設定されている場合は 0xFFFF の後) カウントを続け、0 にロールバックします。カウント値が再び一致値に達すると、別の割り込みが生成されます。カウンタが動作している時、一致カウントは変更可能なことに注意してください。

WDT 割込みが発生するたびに、SRSS_INTR レジスタの WDT_MATCH という名前のビットが設定されます。ウォッチドッグをリセットするには、SRSS_INTR の WDT_MATCH ビットに '1' を書き込んでこの割り込みをクリアする必要があります。ファームウェアは、2 回連続した割り込みで WDT をリセットしない場合、3 番目のイベントは、ハードウェア リセットを生成します。

WDT_MATCH レジスタの IGNORE_BITS は、WDT カウンタ全体の周期を減少させるために使用することができます。無視するビットは、破棄する必要がある MSB の数を指定することができます。例えば、IGNORE_BITS 値が 3 の場合、WDT カウンタは 13 ビット カウンタです。詳細については、*PSoC 4100S: PSoC 4 Registers TRM* および *PSoC 4100S Plus: PSoC 4 Registers TRM* の WDT_COUNTER、WDT_MATCH、および SRSS_INTR レジスタを参照してください。

システム クラッシュからの保護に WDT が使用される時、ウォッチドッグをリセットするために WDT 割込みと直接関係がないコードから WDT 割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 番安全な方法を以下に示します。

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

WDT を周期的割込みジェネレータとして使用するためには以下の手順に従ってください：

1. カウンタ分解能を設定するために、WDT_MATCH レジスタに希望する IGNORE_BITS を書き込みます。
2. WDT_MATCH レジスタに希望する一致値を書き込みます。
3. 保留中の WDT 割込みをクリアするために SRSS_INTR の WDT_MATCH ビットをクリアします。
4. SRSS_INTR_MASK の WDT_MATCH ビットをセットすることで、WDT 割込みを有効にします。
5. CM0_ISER レジスタでグローバル WDT 割込みを有効にします (詳細については、[割り込み \(51 ページ\)](#) を参照してください)。
6. ISR では、WDT 割込みをクリアし、既存のマッチ値に希望するマッチ値を追加します。これにより、カウンタは新しい一致値に達した時、他の定期的な割り込みが生成されます。

割り込みの詳細については、[割り込み \(51 ページ\)](#) を参照してください。

13.3.1 WDT の有効化と無効化

ウォッチドッグ カウンタは無効にできないフリーランニング カウンタです。ただし、キー '0xACED8865' を WDT_DISABLE_KEY レジスタに書き込むことにより、ウォッチドッグリセットを無効にすることができます。このレジスタへの他の値の書き込みはウォッチドッグリセットを有効にします。ウォッチドッグシステム リセットを無効にする場合、ファームウェアはシステム リセットを回避するウォッチドッグの定期的なリセットが必要ありません。ウォッチドッグ カウンタはまだ、割込みソースまたはウェイクアップ ソースとして使用することができます。このカウンタを停止する唯一の方法は、CLK_ILO_CONFIG レジスタの ENABLE ビットをクリアし、ILO を無効にすることです。ウォッチドッグ リセット

は ILO を無効にする前に無効にする必要があります。それ以外の場合、ILO を無効にする任意のレジスタへの書き込みは無視されます。ウォッチドッグ リセットの有効化は、自動的に ILO が有効になります。

注: 次の場合、WDT リセットを無効にすることはお勧めしません。

- 保護はファームウェア クラッシュに対して必要
- 電源供給は CPU 機能を損なう可能性がある突然の電圧低下イベントを生成する

13.3.2 WDT 割込みと低消費電力モード

ウォッチドッグ カウンタはアクティブ 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 (WIC) へ割り込み要求を送信することができます。これらは以下のとおりに動作します:

- **アクティブ モード:** このモードでは WDT は CPU へ割り込みを送信することができます。CPU は割り込み要求をアクノリッジし、ISR を実行します。ファームウェア内の ISR に入った後に、割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 は割り込み要求をアクノリッジし、ISR を実行します。ファームウェア内の ISR に入った後に、割込みをクリアする必要があります

デバイスの電源モードの詳細については、[電力モード \(107 ページ\)](#) を参照してください。

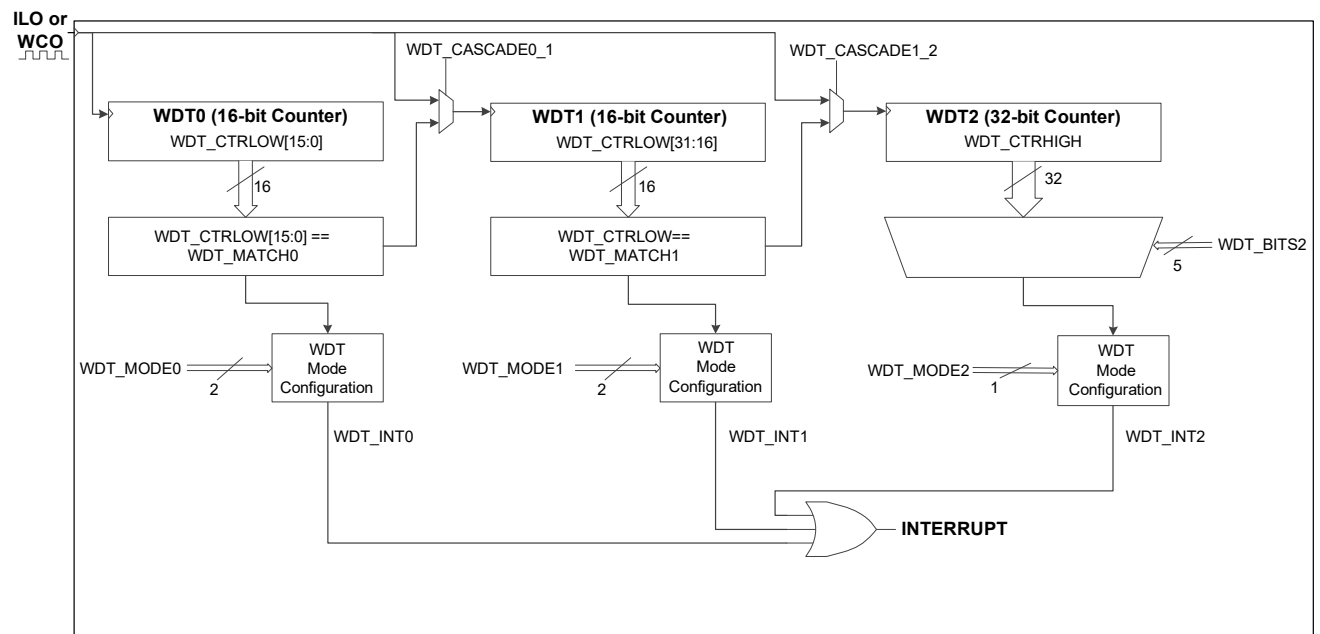
13.3.3 WDT リセット モード

RES_CAUSE レジスタの RESET_WDT ビットは WDT が生成するリセットを示します。このビットはクリアされるか、パワーオン リセット (POR)、電圧低下リセット (BOD) もしくは外部リセット (XRES) が発生するまでセットされたままです。他のすべてのリセットはこのビットに影響を与えません。詳細については、[リセット システム \(120 ページ\)](#) を参照してください。

13.4 追加タイマ

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

Figure 13-2. WDT 追加タイマ ブロック図



13.4.1 WDT0 および WDT1

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

- フリーラン
- クリア オン マッチ (設定可能な周期)

フリーランニング モードでは、タイマは 16 ビット範囲でカウントします。65535 ($2^{16}-1$) に到達すると、タイマは 0 にリセットし、再びカウントを開始します。クリア オン マッチ モードでは、WCO_WDT_MATCH レジスタの WDT_MATCH0 と WDT_MATCH1 に書き込まれたマッチ カウントは WDT0 と WDT1 のそれぞれの周期を決定します。タイマ カウントがマッチ値に到達すると、タイマは 0 にリセットし、再びカウントを開始します。これらの 2 つのコンフィギュレーションのいずれかが、WCO_WDT_CONFIG レジスタの WDT_CLEAR0 および WDT_CLEAR1 ビットを使用して選択します。WDT_CLEARx に「1」を書き込むと、一致時クリアモードが選択されます。このビットに「0」を書き込むと、マッチ カウントでタイマのクリアが無効にされ、フリーランニング モードが設定されます。マッチ カウントの変更は有効となるまで 3 入カクロック サイクルが必要なことに注意してください。デバイスをディープ スリープにする前に、マッチ カウントの更新後の少なくとも 1 入カクロック サイクルの遅延を確保します。

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

タイマは WCO_WDT_CONTROL レジスタの WDT_ENABLEx ビットへの「1」の書き込みによって有効化します。これは有効となるまで 3 クロック サイクルかかることに注意してください。この期間中にこのビットを複数回トグルすることは推奨しません。タイマが有効になった後、コンフィギュレーション レジスタ (WCO_WDT_CONFIG) への書き込みは推奨しません。タイマの現在値は WDT_CTRL0W レジスタから読み出すことができます。これは WCO_WDT_CONTROL レジスタの WDT_RESETx ビットへの「1」の書き込みによってリセットすることができます。

13.4.2 WDT2

これは以下の違いを持って WDT0 と WDT1 と同様です：

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

13.4.3 カスケード

カスケード オプションを以下に示します。

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

13.5 レジスタ一覧

Table 13-1 に、レジスタ制御の詳細を示します。

Table 13-1. WDT レジスタ

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

14. トリガーマルチプレクサブロック



PSoC 4 MCU の一部の周辺機器は、トリガー信号を使用して相互接続されます。トリガー信号は、周辺機器がイベントまたは状態の発生を示す手段です。これらのトリガーは、他のペリフェラルの一部のアクションに影響を与える、または開始する手段として使用されます。トリガーマルチプレクサブロックは、トリガーをソースペリフェラルブロックから宛先にルーティングするのに役立ちます。

14.1 特長

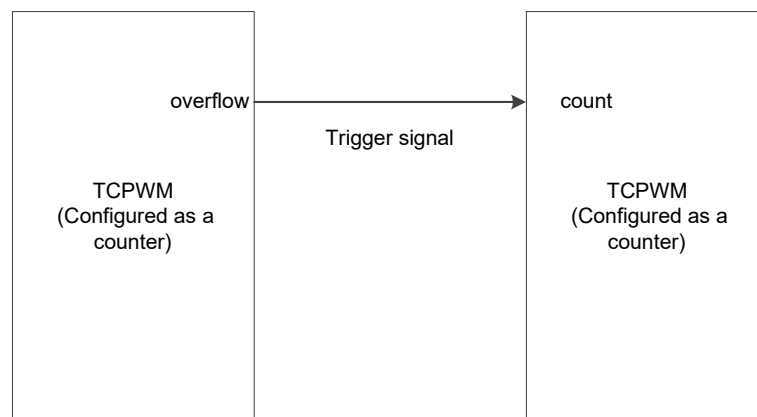
トリガーマルチプレクサブロックには次の機能があります。

- あるペリフェラルから別のペリフェラルにトリガー信号を接続する機能
- ブロック内の信号をトリガーできるソフトウェアトリガーをサポート
- 周辺機器間のトリガーの多重化をサポート

14.2 アーキテクチャ

PSoC 4 MCU のトリガー信号は、TCPWM オーバーフローなどの状態、またはアクションの完了などのイベントを示すために、ペリフェラルブロックによって生成されるデジタル信号です。これらのトリガー信号は、通常、他の周辺ブロックの他のアクションのイニシエーターとして機能します。例として、16 ビットカウンタの代わりに 32 ビットカウンタを作成するために 2 つの TCPWM を一緒にチェーン化します。これは、カウンタオーバーフロートリガーを使用して、2 番目の TCPWM のカウント入力をトリガーすることで実行できます。これは、[Figure 14-1](#) で確認できます。

Figure 14-1. トリガー信号の例



トリガールーティングをサポートするために、PSoC 4 MCU にはハードウェアがあり、これはトリガー信号を潜在的なソースから宛先にルーティングするために使用される一連のマルチプレクサです。このハードウェアは、トリガーマルチプレクサブロックと呼ばれます。トリガーマルチプレクサは、PSoC 4 MCU のペリフェラルブロックから発生するトリガー信号に接続し、それを別のペリフェラルにルーティングして、宛先ペリフェラルブロックでの動作を開始または影響を与えることができます。トリガーには、レベルセンシティブトリガーと立ち上がりエッジトリガーの 2 種類があります。立ち上がりエッジトリガーは、少なくとも 2 つの「clk_sys」サイクルの間「1」のままである必要があります。

14.2.1 マルチプレクサグループのトリガー

トリガーマルチプレクサブロックは、いくつかのトリガーマルチプレクサを使用して実装されます。トリガーマルチプレクサは、異なる周辺ブロックからのトリガー出力信号のセットから信号を選択して、別の周辺ブロックの特定のトリガー入力にルーティングします。これは Figure 14-3 で確認できます。マルチプレクサは、トリガーグループにグループ化されます。トリガーグループ内のすべてのトリガーマルチプレクサには、同様の入力オプションがあり、同様の宛先信号を供給するように設計されています。したがって、トリガーグループは、複数の入力を複数の出力に多重化するブロックと見なすことができます。この概念を Figure 14-2 に示します。

注：トリガーはさまざまな周辺機器に出力されるため、トリガーのルーティング図に示されているよりも多くのルーティングが存在する可能性があります。このルーティングの詳細については、トリガー先のペリフェラルブロックに移動してください。

Figure 14-2. マルチプレクサグループのトリガー

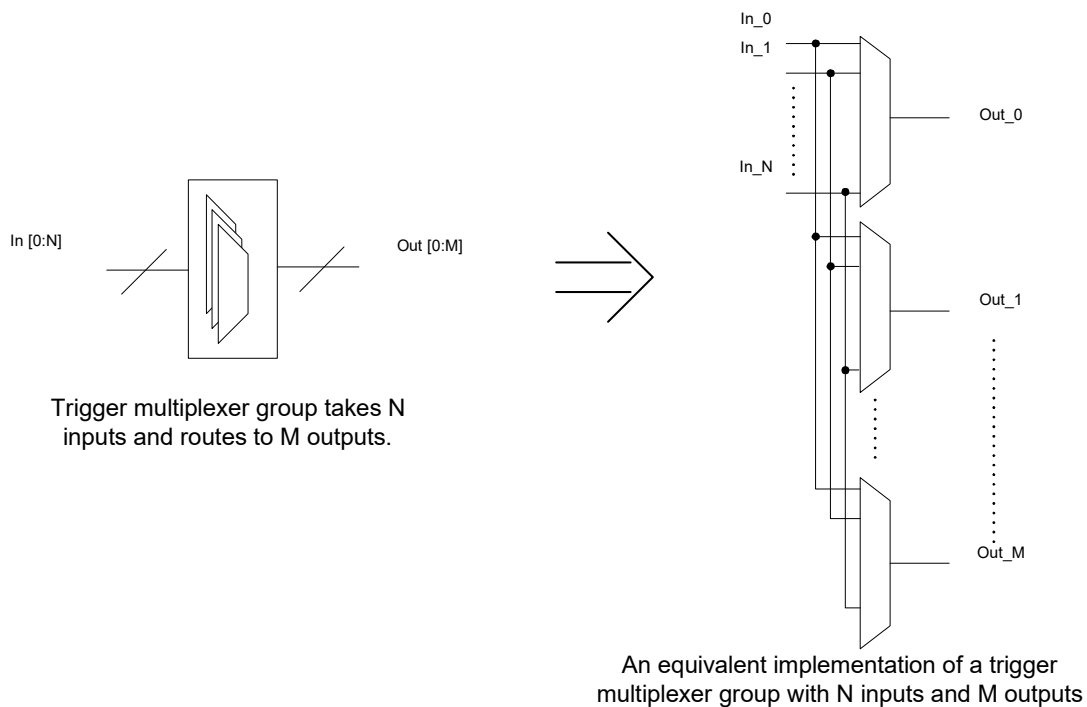
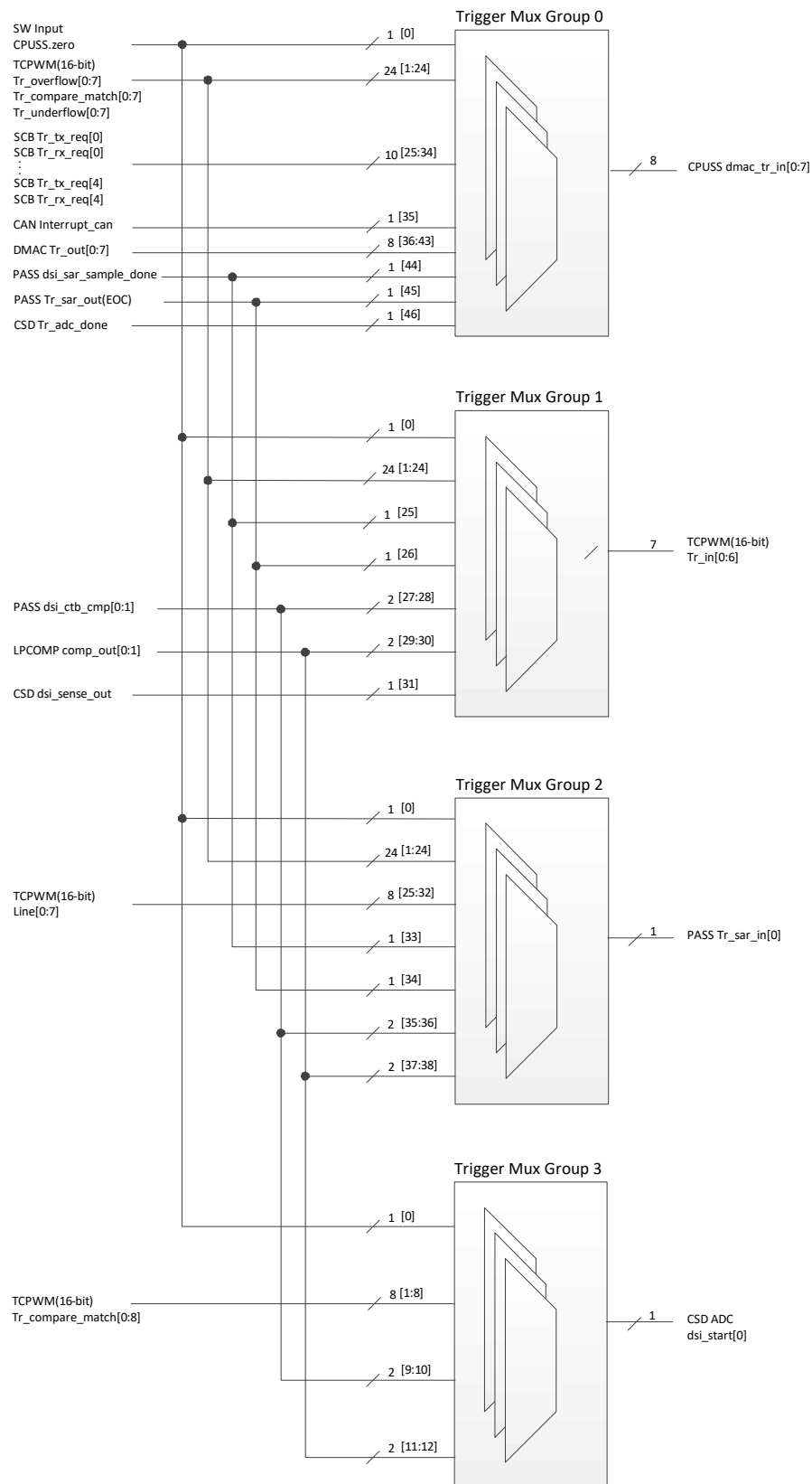


Figure 14-3. トリガーマルチプレクサブロックアーキテクチャ



14.2.2 ソフトウェアトリガー

トリガーマルチプレクサへのすべての入力信号と出力信号は、ソフトウェアからトリガーできます。これは、PERI_TR_CTLレジスタに書き込むことによって行われます。このレジスタを使用すると、多数のペリフェラルクロックサイクルで対応する信号をトリガーできます。PERI_TR_CTL[TR_GROUP] ビットフィールドは、アクティブ化される信号のトリガーグループを選択します。PERI_TR_CTL[TR_OUT] ビットフィールドは、トリガー信号がマルチプレクサの出力または入力のどちらにあるかを決定します。PERI_TR_CTL[TR_SEL] は、トリガーグループ内の特定の行を選択します。PERI_TR_CTL[TR_COUNT] ビットフィールドは、トリガーがアクティブになる周辺クロックの数を設定します。PERI_TR_CTL[TR_ACT] ビットフィールドは「1」に設定され、指定されたトリガーラインをアクティブにします。ハードウェアは、PERI_TR_CTL[TR_COUNT] によって設定されたサイクル数の後にトリガーが非アクティブ化された後、このビットをリセットします。

14.3 レジスター一覧

Table 14-1. レジスター一覧

レジスタ名	説明
PERT_TR_CTL	トリガーコマンドレジスタ。この制御により、トリガーマルチプレクサ構造の特定の入力トリガーまたは出力トリガーをソフトウェアでアクティブ化できます。
PERI_TR_GROUP[X]_TR_OUT_CTL[Y]	このレジスタは、トリガーグループ内の特定の出力トリガーの入力トリガーを指定します。すべてのトリガーマルチプレクサグループにはレジスタのグループがあり、レジスタの数はそのマルチプレクサグループからの出力バスサイズと等しくなります。レジスタ形式では、Xはトリガーグループ、Yはマルチプレクサからの出力トリガーライン番号です。

15. 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、ユーザが供給する外部ハードウェアまたは内部ソフトウェ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。また、PSoC 4 は特定のリセット検出を可能にするためのハードウェアも内蔵しています。

リセット システムには以下のリセットソースがあります：

- パワー オン リセット (POR): 電源電圧の立ち上がり時、デバイスをリセット状態に維持
- 電圧低下リセット (BOD): デバイスの動作中に電源電圧が動作仕様を下回る時にデバイスをリセット
- ウォッチドッグ リセット (WRES): ファームウェアの実行がウォッチドッグ タイマ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): 外部電気信号を使用してデバイスをリセット
- 保護フォールト 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

15.1 リセットソース

次の節は PSoC 4 で利用可能なリセット ソースについて説明します。

15.1.1 パワーオン リセット

パワー オン リセットは、電源投入時にシステム リセットのために提供されます。POR は、電源電圧 V_{DD} 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。

15.1.2 ブラウンアウトリセット

ブラウンアウトリセットは、チップのデジタル電圧供給 V_{CCD} を監視し、 V_{CCD} が [PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートで指定された最小ロジック動作電圧を下回ると、リセットを生成します。BOD はすべての電力モードで使用可能です。

15.1.3 ウォッチドッグリセット

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

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

詳細については、[ウォッチドッグタイマ \(111 ページ\)](#) を参照してください。

15.1.4 ソフトウェア開始リセット

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。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 (CM0P_AIRCR) は、SYSRESETREQ ビットに「1」が書き込まれると、デバイスを強制的にリセットします。CM0P_AIRCR は、書き込みのために上位 2 バイトに書き込まれる 05FA の値を必要とします。したがって、リセットには 05FA0004 を書き込みます。

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

15.1.5 外部リセット

外部リセット (XRES) はユーザ供給リセットで、アサートされると直ちにシステム リセットを発生させます。XRES ピンは **active low (アクティブLOW)** です。ピンの高電圧は効果がなく、低電圧はリセットを引き起こします。この端子はデバイス内で HIGH にプル アップされます。XRES はほとんどのデバイスで、専用ピンとして使用可能です。ピン配置の詳細については、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートのピン配置セクションを参照してください。

XRES 端子がアクティブの間、デバイスをリセット状態に保持します。端子が解放されると、デバイスは通常のブートシーケンスに従います。XRES およびその他の電気的特性の論理しきい値は、[PSoC 4100S datasheet](#)、[PSoC 4100S Plus datasheet](#)、および PSoC 4100S Plus 256 KB データシートの電気的仕様のセクションに記載されています。

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

15.1.6 フォールトリセットプロテクション

保護フォールト 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。保護フォー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、“[特権](#)” (105 ページ) を参照してください。

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

15.2 リセットソースの識別

デバイスがリセットを完了する時点で、直近またはそれ以前のリセット原因が分かるとその後の処理にしばしば都合の良い場合があります。これは主に RES_CAUSE レジスタを通じて可能です。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ールト リセットの検出をサポートします。これは POR、BOD または XRES の発生を記録しません。ビットは対応したリセットが発生した時にセットされ、その後クリアされるか、または POR リセットや外部リセット、電圧低下検出でデータが失われるまで、セットされたままです。

もし、RES_CAUSE レジスタ でリセットの原因が検出できない場合、それは非記録の 1 つであり、非記録リセットです：BOD、POR、XRES。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。

15.3 レジスタ一覧

Table 15-1. リセット システム レジスタ一覧

レジスタ名	説明
WDT_DISABLE_KEY	値 0XACED8865 を書き込むと、WDT は無効です。他の値では WDT は普通に動作
CM0P_AIRCR	Cortex-M0+ アプリケーション割込みおよびリセット制御レジスタ：このレジスタはソフトウェア リセットを開始することに加え、他の Cortex-M0+ 機能を実装
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

Section D: デジタル システム

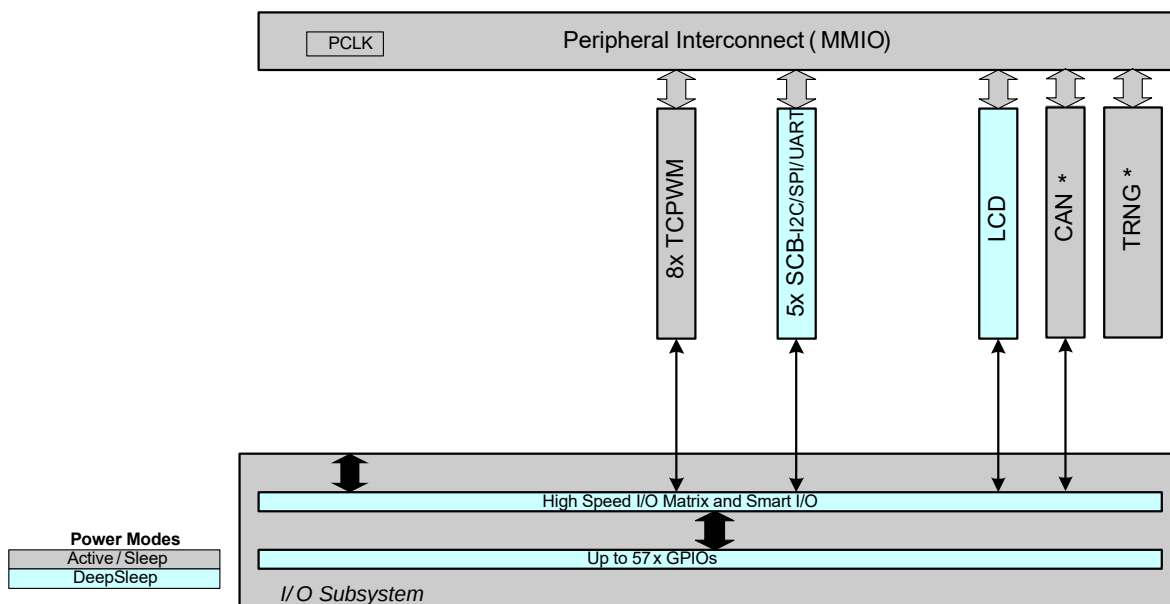


本セクションは次の章を含みます：

- シリアル通信ブロック (SCB) (124 ページ)
- コントローラエリアネットワーク (CAN) (171 ページ)
- タイマ、カウンタ、および PWM (192 ページ)
- 真の乱数生成器 (218 ページ)
- LCD 直接駆動 (223 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 in PSoC 4100S Plus 128KB

16. シリアル通信ブロック (SCB)



PSoC® 4 のシリアル通信ブロック (SCB) は SPI、UART、I²C の 3 つのシリアル インターフェース プロトコルに対応します。SCB は任意の時点で 1 つのプロトコルのみに対応します。PSoC 4100S Plus デバイスには最大 5 つの SCB があります。

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

3 つのプロトコルはそれぞれ以下の節で説明します。

16.2 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。

16.2.1 特長

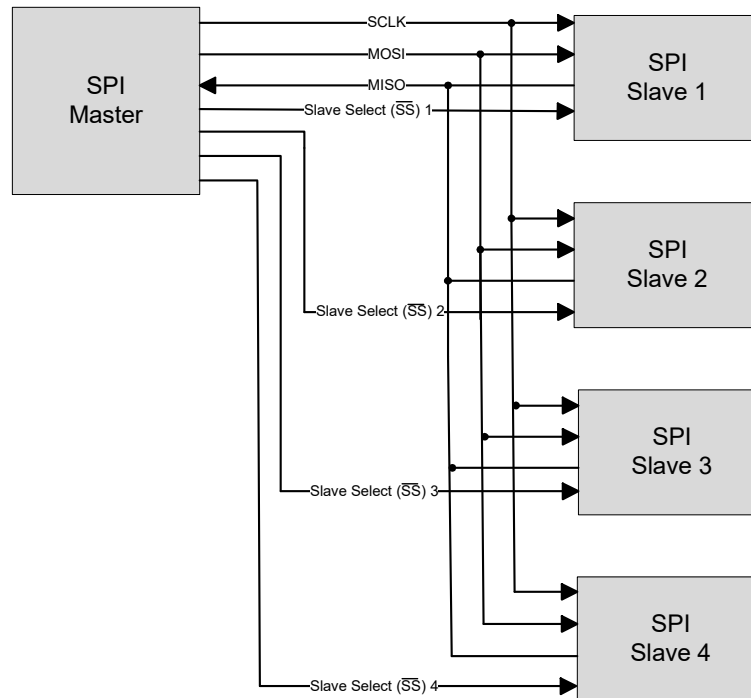
- マスターとスレーブ機能をサポート
- 3 つの種類の SPI プロトコルに対応：
 - Motorola 社 SPI: モード 0、1、2、3
 - Texas Instruments 社 SPI: モード 1 用のデータ フレームの同時インジケータと先行インジケータ機能付き
 - National Semiconductor (MicroWire) 社 SPI: モード 0
- 最大 4 本のスレーブ選択ラインをサポート
- データ フレーム サイズは 4 ビット ~ 16 ビットにプログラム可能
- 割込みまたはポーリングによる CPU インターフェース
- プログラマブルなオーバーサンプリング
- EZ 操作モード ([イージー SPI プロトコル](#)) をサポート
 - 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

16.2.2 概要説明

Figure 16-1 に、4 つのスレーブを持つ SPI マスターの例を示します。

Figure 16-1. SPI の例



標準 SPI インターフェースは以下の 4 つの信号を含んでいます。

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

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

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

SPI プロトコルの 3 つの変種が 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。

16.2.3 SPI 動作モード

16.2.3.1 Motorola 社 SPI

オリジナルの SPI プロトコルは Motorola 社によって定義されました。これは全二重プロトコルです。SS 回線が '0' に保持されていると、複数のデータ転送が発生する可能性があります。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場合、SS ラインは '1' に保持され、SCLK は通常 Low にプルされます。

Motorola 社の SPI モード

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

クロック極性はデータの送信中でない時の SCLK ラインの値を決定します。CPOL = '0' は、データを送信していないときの SCLK が '0' であることを示します。CPOL = '1' は、データを送信していないときの 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 は '0'、CPHA は '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 は '0'、CPHA は '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 は '1'、CPHA は '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 は '1'、CPHA は '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。

Figure 16-2 は、CPOL および CPHA の関数としての MOSI/MISO データの駆動とキャプチャを示します。

Figure 16-2. Motorola 社 SPI、4 モード

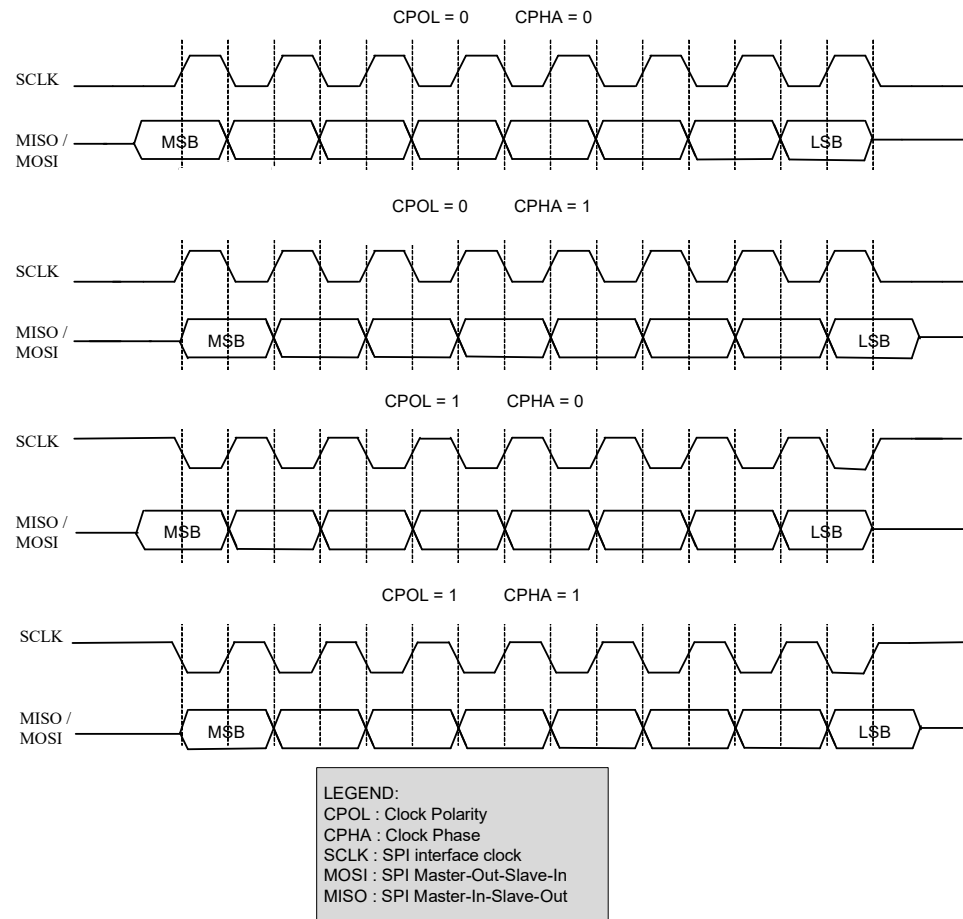
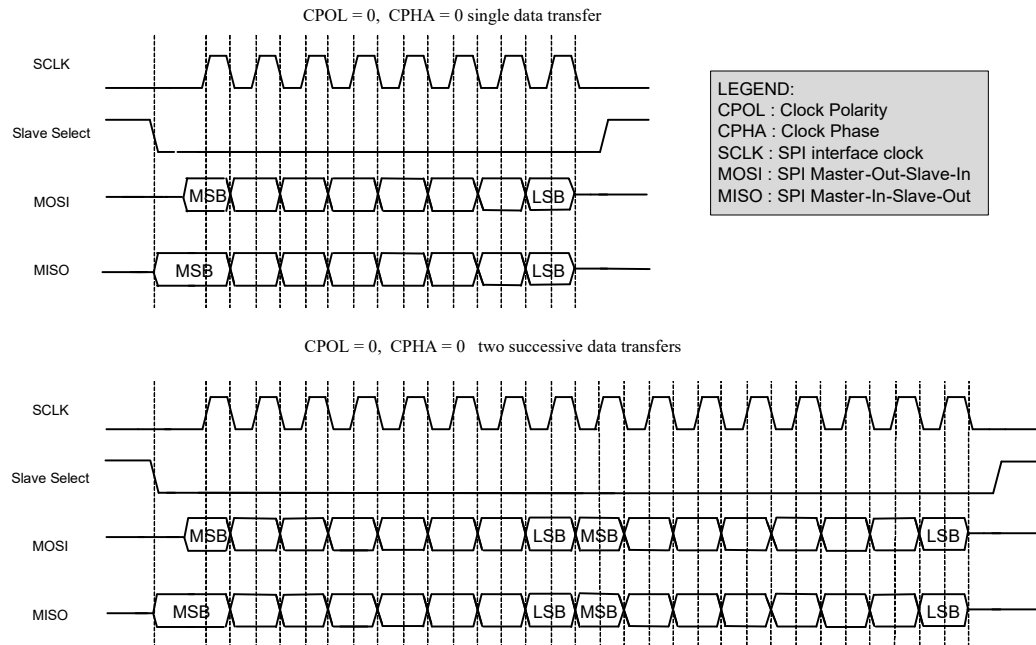


Figure 16-3 は、モード 0 での単一の 8 ビットデータ転送と 2 つの連続する 8 ビットデータ転送を示します (CPOL は '0'、CPHA は '0')。

Figure 16-3. Motorola 社 SPI のデータ転送例



SPI Motorola 社モード用の SCB の構成

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

1. SCB_CTRL レジスタの MODE (ビット [25:24]) に '01' を書き込んで SPI を選択します。
2. SCB_SPI_CTRL レジスタの MODE (ビット [25:24]) に '00' を書き込んで、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 社の動作モードを選択
4. “SPI の有効化と初期化” (134 ページ) に記載されている手順 2 ~ 4 に従います。

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 4100S: PSoc 4 Registers TRM](#) および [PSoC 4100S Plus: PSoc 4 Registers TRM](#) を参照してください。

16.2.3.2 Texas Instruments 社 SPI

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

Figure 16-4 は、単一の 8 ビットデータ転送と 2 つの連続する 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。

Figure 16-4. TI 社 SPI のデータ転送例

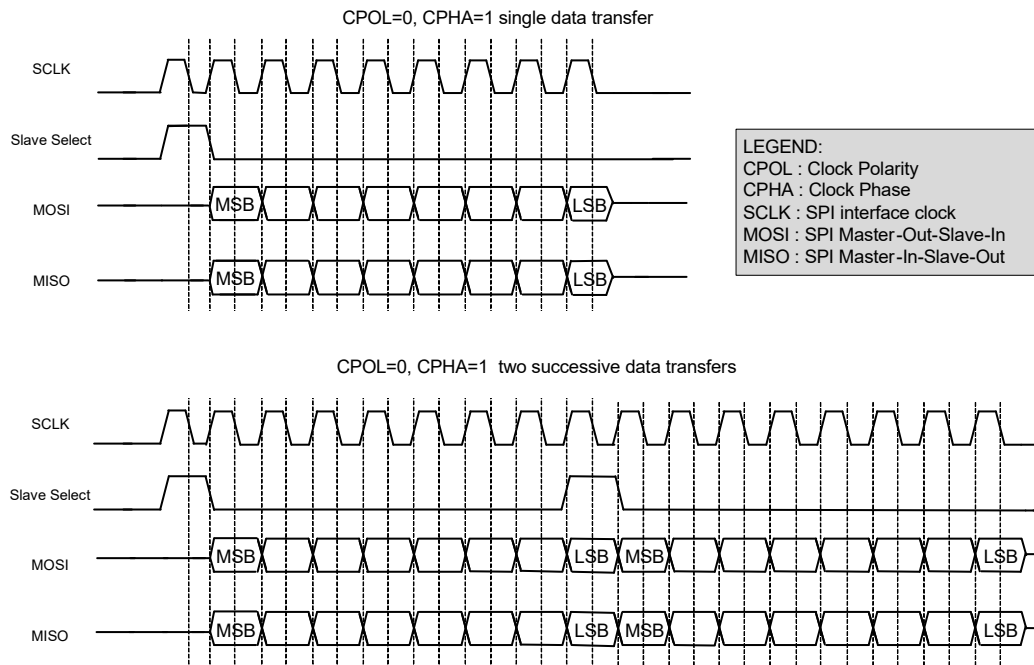
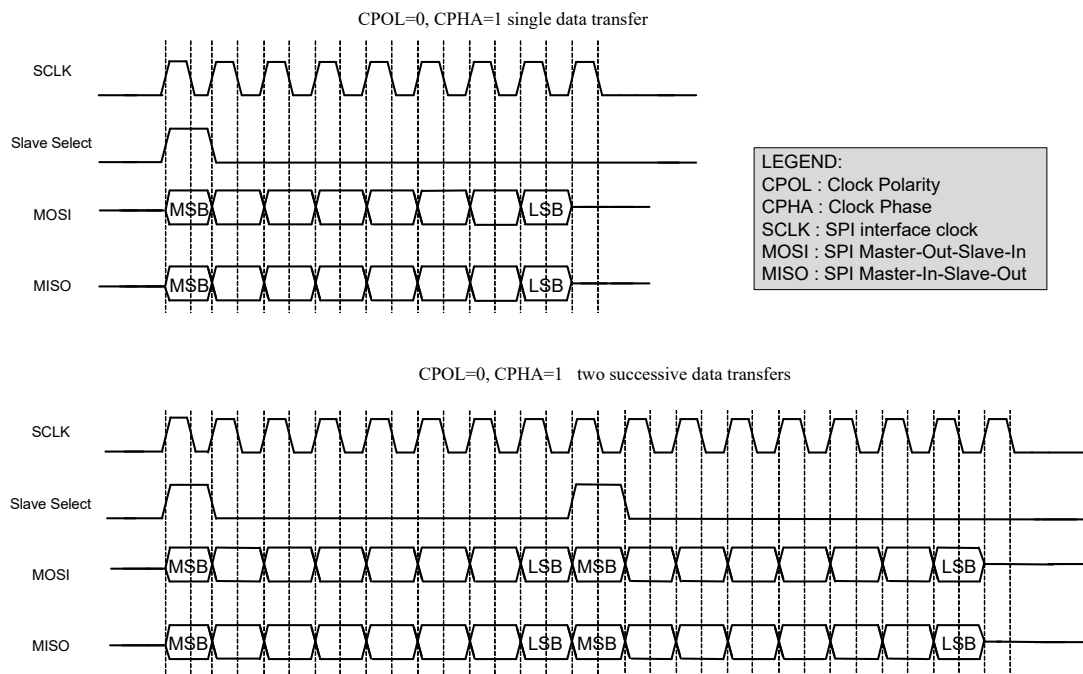


Figure 16-5 は、単一の 8 ビットデータ転送と 2 つの連続する 8 ビットデータ転送を示します。SELECT パルスはフレームの最初のデータ ビットと同時に発生します。

Figure 16-5. TI 社 SPI のデータ転送例



SPI TI モード用の SCB の設定

SPI TI モード用に SCB を設定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 [25:24]) に '01' を書き込んで SPI を選択します。
2. SCB_SPI_CTRL レジスタの MODE (ビット [25:24]) に '01' を書き込んで、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) に書き込むことにより、TI の動作モードを選択します ('1' は次のフレームの最初のビットに先行するように SELECT パルスを構成し、それ以外の場合は '0' を構成します)。
4. “SPI の有効化と初期化” (134 ページ) に記載されている手順 2 ～ 5 に従います。

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 4100S: PSoc 4 Registers TRM* および *PSoC 4100S Plus: PSoc 4 Registers TRM* を参照してください。

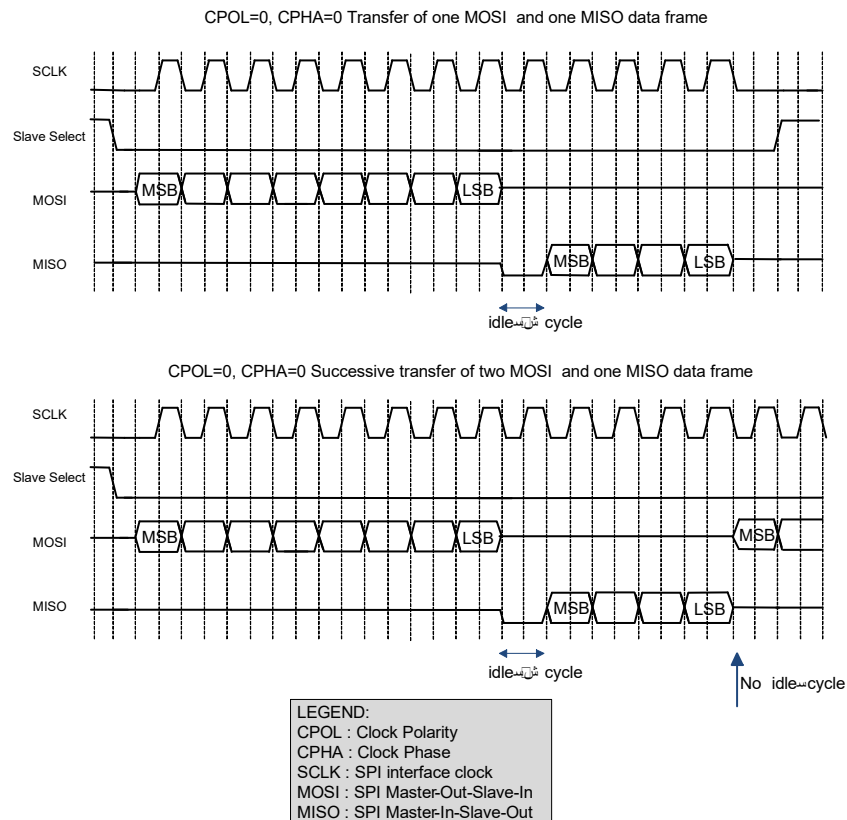
16.2.3.3 National Semiconductor 社 SPI

National Semiconductors' 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」ビット転送期間により、送信と受信が分離されます。ただし、連続するデータ転送は、アイドルビット転送期間によって分離されません。

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。

Figure 16-6 は、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。

Figure 16-6. NS 社 SPI のデータ転送例



SPI NS モード用の SCB の設定

SPI NS モード用に SCB を設定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 [25:24]) に '01' を書き込んで SPI を選択します。
2. SCB_SPI_CTRL レジスタの MODE (ビット [25:24]) に '10' を書き込んで、SPI NS モードを選択します。
3. “SPI の有効化と初期化” (134 ページ) に記載されている手順 2 ~ 5 に従います。

PSoC Creator がコンポーネント カスタマイザを利用してこれらのすべてを自動的に行うことに注意してください。これらのレジスタの詳細については、*PSoC 4100S: PSoc 4 Registers TRM* および *PSoC 4100S Plus: PSoc 4 Registers TRM* を参照してください。

16.2.4 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とき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。これを有効にするには、SCB_SPI_CTRL レジスタの SCLK_CONTINUOUS (ビット 5) に '1' を書き込みます。

16.2.5 イーザー SPI プロトコル

イーザー 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 に基づきます。このプロトコルは個別のフレームのレベルで CPU の介入なしにマスターとスレーブ間の通信を可能にします。

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

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

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

16.2.5.1 EZ アドレス書き込み

EZ アドレスの書き込みは、MOSI ライン上のコマンドバイト (0x00) で始まり、EZ アドレスを書き込む master's を示します。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。

16.2.5.2 メモリ アレイ書き込み

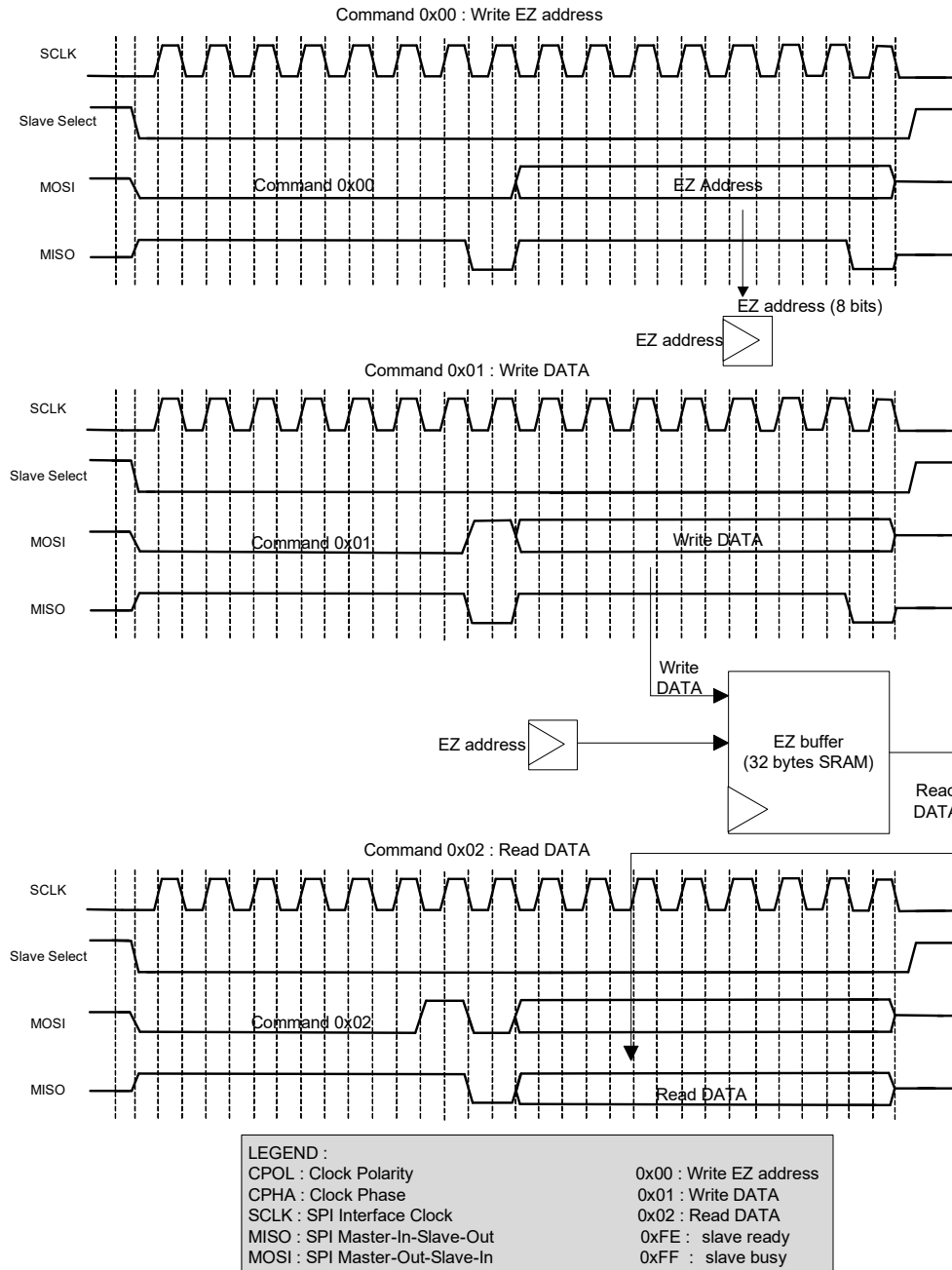
メモリアレイインデックスへの書き込みは、MOSI ラインのコマンドバイト (0x01) で始まり、メモリアレイに書き込む master's を示します。その後、スレーブはコマンドが登録された (0xFE) または非登録の (0xFF) であることを示すために MISO ライン上に返信バイトを駆動します。MOSI 上の追加書き込みのデータ バイトは通信した EZ アドレスによって示される位置でメモリアレイに書き込まれます。バイトがメモリアレイに書き込まれると EZ アドレスはスレーブによって自動的にインクリメントされます。EZ アドレスがメモリアレイのエントリの最大数 (32) を超えるとその値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

16.2.5.3 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取りは、MOSI ラインのコマンドバイト (0x02) で始まり、メモリアレイから読み取る master's を示します。その後、スレーブはコマンドが登録された (0xFE) または非登録の (0xFF) であることを示すために MISO ライン上に返信バイトを駆動します。MISO 上の追加読み出しのデータ バイトは通信した EZ アドレスによって示される位置でメモリアレイから読み出されます。バイトがメモリアレイから読み出されると、EZ アドレスはスレーブによって自動的にインクリメントされます。EZ アドレスがメモリアレイのエントリの最大数 (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。

Figure 16-7 は、EZ アドレスの書き込み、メモリアレイへの書き込み、および EZSPI プロトコルでのメモリアレイ操作からの読み取りを示します。

Figure 16-7. EZSPI の例



16.2.5.4 EZSPI モード用の SCB の設定

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

1. SCB_CTRL レジスタの EZ_MODE ビット (ビット 10) に '1' を書き込んで、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' を書き込むことにより、トランスミッタの連続送信モードを使用します。
3. “SPI の有効化と初期化” (134 ページ) に記載されている手順 2 ～ 5 に従います。

PSoC Creator がコンポーネント カスタマイザー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 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

16.2.6 SPI レジスタ

SPI インターフェースは、[Table 16-1](#) にリストされている 32 ビットの制御およびステータスレジスタのセットを使用して制御されます。これらのレジスタの詳細については、[PSoC 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

Table 16-1. SPI レジスタ

レジスタ名	動作
SCB_CTRL	SCB を有効にし、シリアル インターフェースのタイプ (SPI、UART、I ² C) を選択し、内部と外部クロック供給動作、EZ と非 EZ のモード動作を選択
SCB_STATUS	EZ モードではこのレジスタは外部からクロック供給されるロジックが EZ メモリを使用している可能性があるかを示す
SCB_SPI_CTRL	SPI をマスターまたはスレーブとして構成し、SPI プロトコル (Motorola、TI、National) を選択し、Motorola SPI で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 SPI で SELECT 信号のタイプを選択します。SPI がスレーブモードとして機能する場合、最初のチップ選択ピン SPI_SELECT [0] のみがスレーブモードで使用できます。
SCB_SPI_STATUS	SPI バスがビジーであること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。メディア 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 FIFO とシフト レジスタをクリアしトランスミッタ 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
SCB_TX_FIFO_WR	トランスミッタ FIFO に書き込まれるデータ フレームを格納。動作はブッシュ動作と同じ
SCB_RX_FIFO_RD	レシーバ FIFO から読み出されるデータ フレームを格納。データ フレームを読み出すとそのデータ フレームが FIFO から削除される。POP 動作と同じ。このレジスタはソフトウェアによって読み出される時にデータ フレームが FIFO から除去されるという副作用がある
SCB_RX_FIFO_RD_SILENT	レシーバ 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 FIFO に格納されているバイト数、データ フレームをハードウェアで読み出しするロケーション (読み出しポインタ)、新しいデータ フレームを書き込むロケーション (書き込みポインタ) を指定し、トランスミッタ 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
SCB_EZ_DATA	EZ メモリ位置内のデータを格納

16.2.7 SPI 割込み

SPI は内部割込み要求にも外部の割込み要求にも対応しています。内部割込みイベントはここで示されています。PSoC Creator はバッファ管理割込みの処理用に必要な割込みサービス ルーチン (ISR) を生成します。外部割込みコンポーネントを SPI コンポーネント (外部割込みが有効) の割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 の事前定義の割込みは TX 割込みと RX 割込みに分類できます。TX 割込みの出力はすべての可能な TX 割込みソースのグループの論理和です。この信号は任意の有効な TX 割込みソースが真の場合、HIGH になります。RX 割込みの出力はすべての可能な RX 割込みソースのグループの論理和です。この信号は任意の有効な Rx 割込みソースが真の場合、HIGH になります。様々な割込みレジスタは割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込みに対応します：

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

注：SPI 割り込み信号は、Cortex-M0 NVIC にハードワイヤードされており、外部ピンにルーティングできません。

16.2.8 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. [Table 16-3](#) 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. [Table 16-4](#) に示すように、SCB_TX_CTRL および SCB_RX_CTRL レジスタを使用して、一般的なトランスミッタおよびレシーバ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. [Table 16-5](#) に示すように、SCB_TX_FIFO_CTRL および SCB_RX_FIFO_CTRL レジスタをそれぞれ使用して、トランスミッタおよびレシーバ 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. トランスミッタとレシーバ 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。これらのレジスタビットを Table 16-2 に示します。
 5. ブロックを有効にします ('1' を SCB_CTRL レジスタの ENABLED ビットに書き込みます)。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。

Table 16-2. SCB_CTRL レジスタ

ビット	名称	値	説明
[25:24]	MODE	00	I ² C 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

Table 16-3. SCB_SPI_CTRL レジスタ

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

Table 16-4. SCB_TX_CTRL/SCB_RX_CTRL レジスタ

ビット	名称	説明
[3:0]	DATA_WIDTH	'DATA_WIDTH + 1' は、送信または受信されたデータフレームのビット数です。有効な範囲は [3, 15]。スタート、ストップとパリティのビットを含まない EZSPI の場合、この値は '0b0111' である必要があります
8	MSB_FIRST	1 = MSB ファースト 0 = LSB ファースト。EZSPI 用には、この値は 1 でなければならない
9	MEDIAN	これは SCB_RX_CTRL 専用。 3 タップのデジタルメディアン フィルタが入カインターフェース 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

Table 16-5. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL レジスタ

ビット	名称	説明
[7:0]	TRIGGER_LEVEL	トリガー レベル: このフィールドの値に比べてトランスミッタ FIFO のエントリ数が少ないまたはレシーバ FIFO のエントリ数が多い場合、それぞれでトランスミッタまたはレシーバトリガー イベントが生成される
16	CLEAR	'1' になると、トランスミッタまたはレシーバ FIFO およびシフトレジスタがクリアされます。
17	FREEZE	'1' の場合、送信機または受信機の FIFO に対するハードウェアの読み取り / 書き込みは影響しません。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

16.2.9 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。システムクロックの詳細については、[クロック供給システム \(89 ページ\)](#) を参照してください。オーバーサンプリングもサポートされています。オーバーサンプリングは高周波数クロックに対して行われます。SCB_CTRL レジスタの OVS ビット (ビット [3:0]) はオーバーサンプリングを指定します。

SPI マスター モードではオーバーサンプリングの有効な範囲は 4 ~ 16 です。したがってクロック速度が 48MHz の場合、最大ビット レートは 12Mbps です。しかし、I/O セルと配線遅延を考慮すれば、オーバーサンプリングは正常の動作のために 6 ~ 16 の範囲内で設定する必要があります。そのため最大ビットレートは 8Mbps です。**注:** 可能な最大ビットレートを実現するには、SPI マスターモードで LATE_MISO_SAMPLE を '1' に設定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。外部 SPI マスターがレイト MISO サンプリングをサポートし、中央値ビットが「0」に設定されている場合、達成可能な最大データレートは 16Mbps です。外部 SPI マスターが遅い MISO サンプリングをサポートしていない場合、最大データレートは 8Mbps に制限されます (中央ビットが「0」に設定されています)。これらのビットに基づいて、最大ビットレートを [Table 16-6](#) に示します。

Table 16-6. 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 メディアン	SCB_CTRL の LATE_MISO_SAMPLE
8 Mbps	≥6	0	1
6 Mbps	≥8	1	1
4 Mbps	≥12	0	0
3 Mbps	≥16	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、block's SRAM をメモリ構造として使用します。非EZ機能は、TXおよびRX FIFOとしてblock's SRAMを使用します。FIFOのサポートは、外部クロック動作では利用できません。
- Motorola 社モード 0、1、2、3

外部クロック供給のEZ動作モードでは (インターフェース クロックが48MHz のとき) 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 スレーブ選択が内部 ('0') または外部 ('1') のどちらでクロックされるかを示します。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**: プロトコル操作の残りの部分 (SPI スレーブ選択以外) が内部 ('0') または外部 ('1') のどちらでクロックされるかを示します。前述のように外部クロック供給動作は非 EZ 機能をサポートしません。

この 2 つのレジスタ フィールドは SPI の機能動作を決定します。これらのレジスタ フィールドはアクティブ、スリープ、ディープスリープのシステム消費電力モードでの必要な動作に基づいて設定する必要があります。不適切な設定を行うと、特定のシステム電源モードで正常に動作しない場合があります。Table 16-7 および Table 16-8 は、(非 EZ および EZ モードでの) SPI の設定について説明しています。

16.2.9.1 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。外部クロック動作は非 EZ 機能ではサポートされていない (FIFO サポートなし) ため、EC_OP_MODE は常に '0' に設定する必要があります。ただし、EC_AM_MODE は '0' または '1' に設定できます。Table 16-7 に、可能性の概要を示します。

Table 16-7. 非 EZ モードでの SPI 動作

SPI (非 EZ) モード				
システム消費電力 モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
Active (アクティブ) と Sleep (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 込みソースは無効 (MASK = 0)。 スリープ モードで MASK ビットはユーザによって設定 可能。	サポートされていま せん	サポートされていま せん
Deep-Sleep (ディー プスリープ)	サポートされていま せん	外部クロックを使用する選択 : 復帰割込みソースは有効 (MASK = 1)。 0xFF を送信		

EC_OP_MODE は '0' で、EC_AM_MODE は '0' です。この設定は、アクティブおよびスリープシステムの電源モードでのみ機能します。block's 全体の機能は、内部クロックドメインで提供されます。

EC_OP_MODE は '0' で、EC_AM_MODE は '1' です。この設定はアクティブおよびスリープシステム電源モードで機能し、ディープスリープシステム電源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、割込みを生成し、CPU を復帰させるために使用できる復帰割込みソース ビットをセットします。

- アクティブシステム電源モードでは、CPU とブロックの内部クロック動作がアクティブになり、ウェイクアップ割り込みの原因が無効になります (関連付けられた MASK ビットは '0' です)。ただし、スリープモードでは、アプリケーションに応じてウェイクアップ割り込みの原因を有効または無効にできます (MASK ビットは '1' または '0' のいずれかです)。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。
- ディープスリープシステム電源モードでは、CPU をウェイクアップする必要があります、ウェイクアップ割り込みの要因が有効になります (MASK ビットは '1')。ウェイクアップには時間がかかるため、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 ラインに送信されます)、ウェイクアップされたときに内部でクロックされる動作が次の SPI 転送を処理します。

16.2.9.2 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' の場合、EC_AM_MODE は '0' または '1' に設定でき、EC_OP_MODE が '1' の場合、EC_AM_MODE は '1' に設定する必要があります。Table 16-8 に、可能性の概要を示します。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。

Table 16-8. EZ モードでの SPI 動作

SPI、EZ モード				
システム消費電力 モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
Active (アクティブ) と Sleep (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
Deep-Sleep (ディープスリープ)	サポートされていません	外部クロックを使用する選択： 復帰割込みソースは有効 (MASK = 1)。 0xFF を送信		外部クロックを使用する選択。 外部クロックを使用する動作

EC_OP_MODE は '0' で、EC_AM_MODE は '0' です。この設定は、アクティブおよびスリープシステムの電源モードでのみ機能します。block's 全体の機能は、内部クロックドメインで提供されます。

EC_OP_MODE は '0' で、EC_AM_MODE は '1' です。この設定は、アクティブおよびスリープシステム電源モードで機能し、ディープスリープシステム電源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、割込みを生成し、CPU を復帰させるために使用できる復帰割込みソース ビットをセットします。

- アクティブシステム電源モードでは、CPU とブロックの内部クロック動作がアクティブになり、ウェイクアップ割り込みの原因が無効になります (関連付けられた MASK ビットは '0' です)。ただし、スリープモードでは、アプリケーションに基づいてウェイクアップ割り込みの原因を有効または無効にできます (MASK ビットは '1' または '0' のいずれかです)。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。
- ディープスリープシステム電源モードでは、CPU をウェイクアップする必要があるため、ウェイクアップ割り込みの原因が有効になります (MASK ビットは '1')。ウェイクアップには時間がかかるため、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 ラインに送信されます)、ウェイクアップされたときに内部クロック動作が次の SPI 転送を処理します。

EC_OP_MODE は '1' で、EC_AM_MODE は '1' です。この設定は、アクティブ、スリープ、およびディープスリープシステムの電源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。この設定により、block's SRAM への外部クロックアクセスが発生することに注意してください。これらのアクセスは、デバイスからの内部的にクロックされるアクセスと競合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可能性があります。SCB_CTRL レジスタのフィールド FIFO_BLOCK は、待機状態 ('1') またはバスエラー ('0') が生成されるかどうかを決定します。

16.3 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: トランスミッタ出力
- RX: レシーバ入力

16.3.1 特長

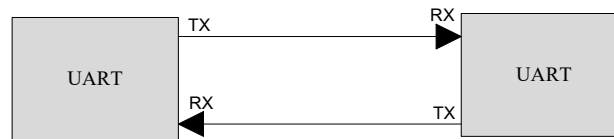
- 非同期トランスミッタとレシーバ機能
- 最大 3Mbps までのデータ レートをサポート
- UART プロトコルをサポート
 - 標準 UART
 - スマートカード (ISO7816) リーダー。
 - IrDA
- ローカル インターコネクト ネットワーク (LIN) をサポート

- 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

16.3.2 概要説明

Figure 16-8 に、標準の UART TX および RX を示します。

Figure 16-8. UART 例



典型的な UART 転送は、「スタートビット」、それに続く複数の「データビット」、オプションで「パリティビット」、最後に 1 つ以上の「ストップビット」で構成されます。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、トランスミッタとレシーバは自身のクロックを使用し 1 ビット転送周期について同調する必要があります。

3 つの異なるシリアル インターフェース プロトコルがサポートされます：

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

UART はデフォルトで 8 ビットのデータ フレーム幅をサポートします。ただし、これは 4 ~ 9 の範囲の任意の値に設定できます。スタート、ストップとパリティのビットを含まないストップビットの数は 1 ~ 4 の範囲で指定できます。

パリティビットは有効または無効にできます。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 モードの場合、パリティビットは自動的に無効です。Figure 16-9 は、SCB の UART インターフェースのデフォルト構成を示します。

注: UART インターフェースは外部クロック動作をサポートしていません。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。

16.3.3 UART の動作モード

16.3.3.1 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。スタートビット値は常に '0' で、データビット値は転送されたデータに依存し、パリティビット値はデータビット上で偶数または奇数のパリティを保証する値に設定され、ストップビット値は '1' です。パリティ ビットはトランスミッタ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によって使用可能です。データを送信しない場合、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、トランスミッタとレシーバはビット転送の周期について合意する必要があります。トランスミッタとレシーバはそれ自身の内部クロックを持ちます。レシーバ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。

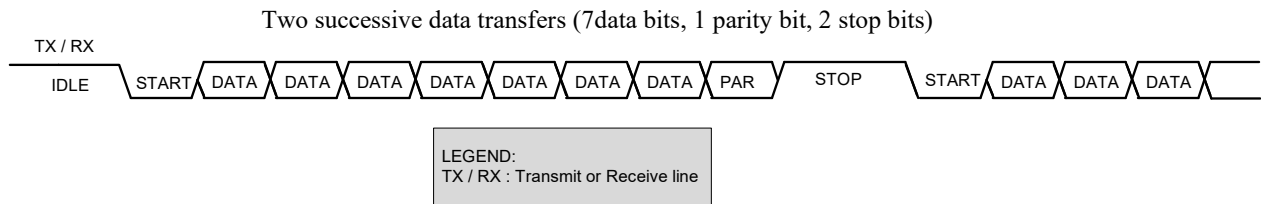
ストップビットからスタートビットへの遷移は、TX ラインの '1' から '0' への変化で表されます。レシーバはこの移行を使用してトランスミッタ クロックと同期化することが可能

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とレシーバ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とレシーバ間で合意され 1 ~ 3 ビットの転送周期の範囲内にあります。

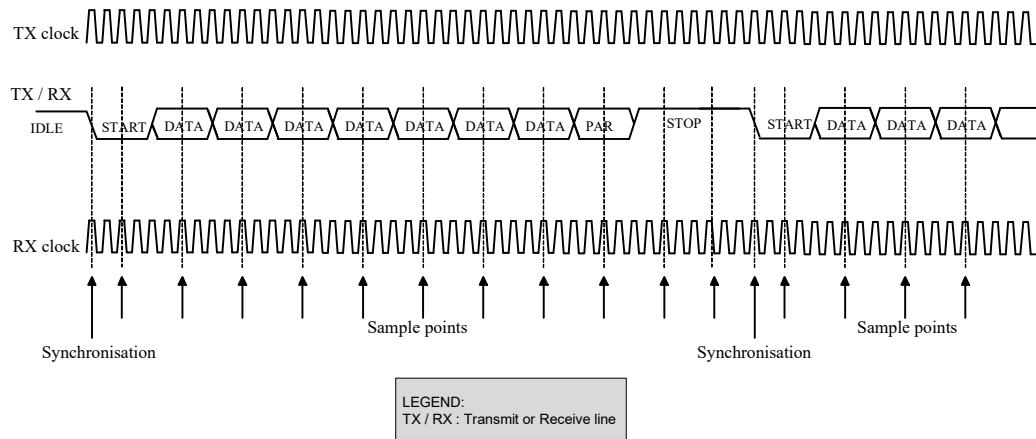
Figure 16-9 は、UART プロトコルを示します。

Figure 16-9. 標準プロトコルの例である UART



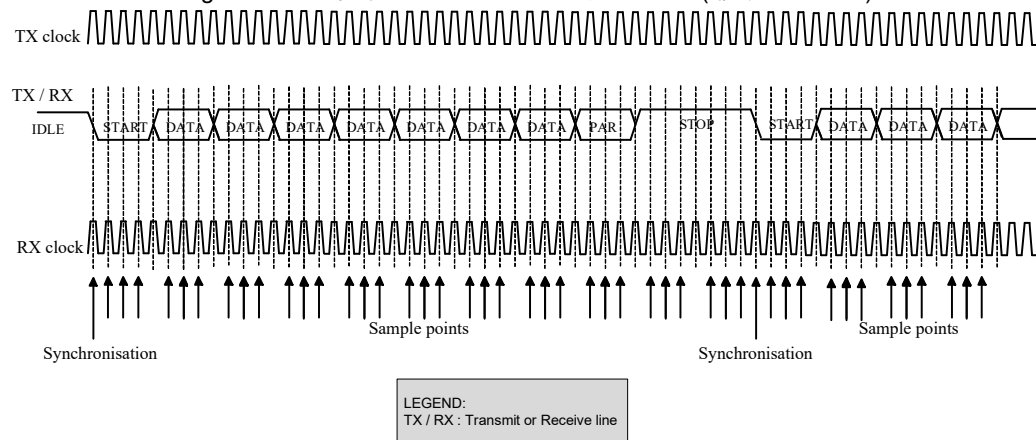
受信機は入力信号をオーバーサンプリングします。(receiver'sのクロック上の) ビット転送期間の中央にあるサンプルポイントの値が使用されます。Figure 16-10 はこれを示します。

Figure 16-10. 標準プロトコルの例である UART (シングル サンプル)



または、ビット転送期間の中央付近 (receiver's のクロック上) の 3 つのサンプルを使用して、精度を高めるために多数決を行います。Figure 16-11 はこれを示します。

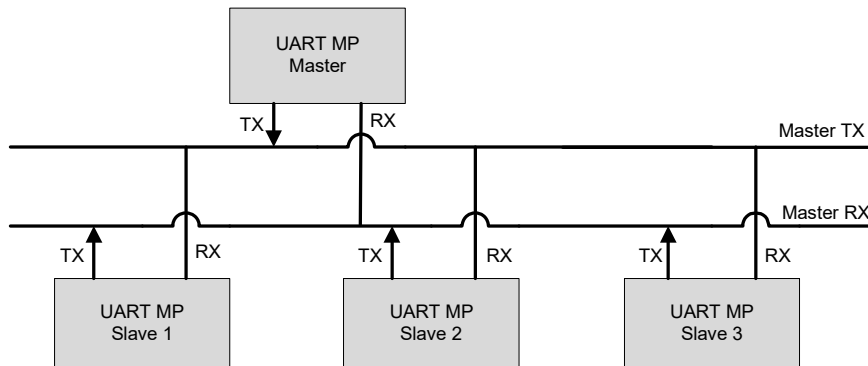
Figure 16-11. 標準プロトコルの例である UART (複数サンプル)



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、Figure 16-12 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。

Figure 16-12. UART MP モードのバス接続



UART_MP モードの主なプロパティを次に示します。

- 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。データフレームを Figure 16-13 に示します。
- パリティ ビットは無効

Figure 16-13. UART MP データ フレーム

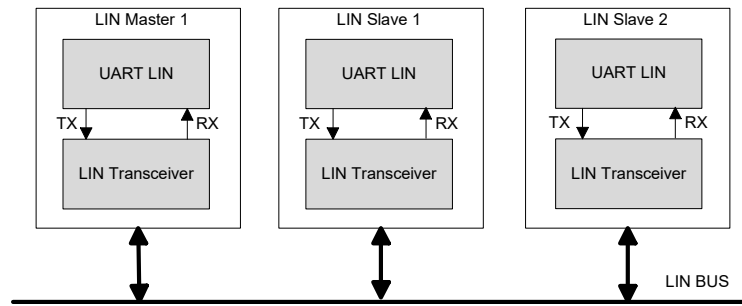


SCB は UART_MP モードでマスターまたはスレーブ デバイスとして使用可能です。SCB_TX_CTRL と SCB_RX_CTRL レジスタは両方とも9ビットのデータ フレーム サイズに設定できます。SCB が UART_MP マスター デバイスとして動作する時にファームウェアはすべてのアドレスまたはデータ フレーム用の MP フラグを変更します。UART_MP スレーブデバイスとして機能する場合、SCB_UART_RX_CTRL レジスタの MP_MODE フィールドを '1' に設定する必要があります。SCB_RX_MATCH レジスタはスレーブ アドレスとアドレス マスク用にセッ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

UART ローカル インターコネクト ネットワーク (LIN) モード

LIN プロトコルは SCB によって標準 UART の一部としてサポートされます。LIN はシングルマスターマルチスレーブ トポロジで設計されています。LIN バス上に1つのマスター ノードと複数のスレーブ ノードがあります。SCB UART は LIN マスターとスレーブ機能を両方ともサポートします。LIN 仕様では、物理層 (レイヤー 1) とデータリンク層 (レイヤー 2) の両方が定義されています。Figure 16-14 は、UART_LIN および LIN トランシーバを示します。

Figure 16-14. UART_LIN と LIN トランシーバ

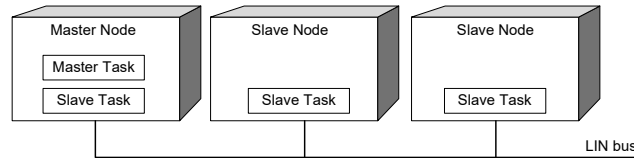


LIN プロトコルは以下の 2 タスクを定義します：

- マスター タスク：このタスクはヘッダ 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。

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ています。Figure 16-15 に示すように、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。

Figure 16-15. LIN バスのノードとタスク

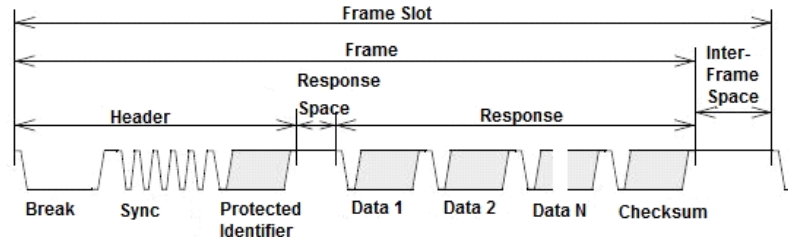


LIN フレームの構造

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。Figure 16-16 に示すように、フレームはヘッダーフィールドと応答フィールドに分割されます。

- ヘッダ フィールドは以下のものから設定されます：
 - ブレークフィールド (値が '0' の 13 ビット周期以上)。
 - 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。

Figure 16-16. 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。

Break Field (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は、Figure 16-17 に示すとおりです。

Figure 16-17. LIN ブレーク フィールド



Sync Field (同期フィールド)

このフィールドはヘッダ 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。同期フィールドを使用して、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させ、自動ポーレートを検出できます。Figure 16-18 は、LIN 同期フィールドの構造を示します。

Figure 16-18. LIN 同期フィールド



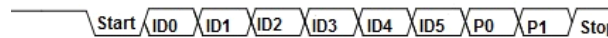
Protected identifier (PID) Field (保護識別子 (PID) フィールド)

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設定されます。PID フィールドの構造を Figure 16-19 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。

Figure 16-19 は、PID フィールド構造を示します。

Figure 16-19. PID フィールド



Data (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。

Checksum (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 フレームの 2 つのタイプのチェックサムを次に示します。

- 標準チェックサム: 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。

Unconditional Frame (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への応答を提供します。

Event-Triggered Frame (イベントトリガー フレーム)

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

- 同じ長さ
- 同じチェックサム モデル (標準または拡張) を使用
- 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

Sporadic Frame (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。

Diagnostic Frame (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を以下に示します。

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

マスター要求フレームを送信する前にマスター タスクはその診断モジュールへの問い合わせを行って、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。

Reserved Frame (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。

ウェイクアップは、バスを 250 μ s ~ 5 ms の間ドミナントにすることで、LIN バスに接続された任意のノード (LIN マスターまたは LIN スレーブのいずれか) によって開始できます。各スレーブは 100ms 以内に復帰要求を検出してヘッダ処理の準備ができています。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 バスの供給電圧範囲は 7V ~ 18V です。通常 LIN ライン ドライバは 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 を設定

標準 UART インターフェースとして SCB を設定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' を書き込むことにより、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' を書き込んで、UART プロトコルを標準プロトコルとして動作するように構成します。
3. UART MP モードまたは UART LIN モードを有効にするには、SCB_UART_RX_CTRL レジスタの MP_MODE (ビット 10) または LIN_MODE (ビット 12) にそれぞれ '1' を書き込みます。
4. “UART の有効化と初期化” (150 ページ) で説明されている手順 2 ~ 5 に従ってください。

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 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

16.3.3.2 スマートカード (ISO7816)

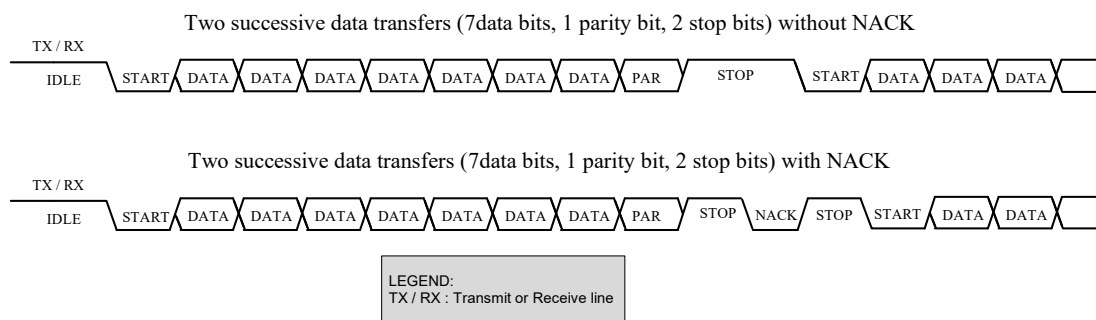
ISO7816 はシングル マスター シングル スレーブ トポロジで定義される非同期シリアルインターフェースです。ISO7816 はリーダー (マスター) とカード (スレーブ) 機能の両方を定義します。詳細については [ISO7816 Specification](#) を参照してください。マスター (リーダー) 機能のみは SCB によってサポートされます。このブロックは非同期文字送信を使用して基本物理層のサポートを提供します。UART_TX と UART_RX コントロール モジュール間の内部多重化で UART_TX ラインが SmartCard I/O ラインに接続されます。

SmartCard 転送は UART 転送と同じですがレシーバからトランスミッタへの否定応答 (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。解放すると、行は '1' (ストップビットの値) になります。1 ビット転送期間から停止期間に入ると、レシーバは 1 ビット転送期間の間、回線上で NACK ('0' の値) を駆動できます。この NACK がトランスミッタによって観察され、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。通常、実装ではプルアップ抵抗付きのトライステートドライバを使用するため、ラインがデータを送信していないか、ストップビットを送信していない場合、その値は '1' です。

Figure 16-20 は、スマートカードプロトコルを示します。

Figure 16-20. SmartCard 例



ISO7816 の通信ボーレートは以下のように計算されます：

$$\text{ボーレート} = f_{7816} \times (D/F)$$

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。

デフォルトでは、F = 372、D = f1、最大クロック周波数は 5MHz です。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 を設定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 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

1. SCB_CTRL レジスタの MODE (ビット [25:24]) に '10' を書き込んで、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 (ビット [25:24]) に '01' を書き込むことにより、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. [“UART の有効化と初期化” \(150 ページ\)](#) で説明されている手順 2 ~ 5 に従ってください。

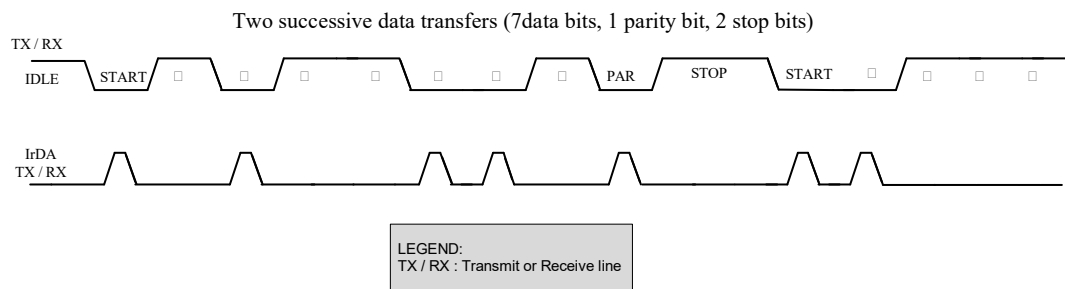
16.3.3.3 IrDA

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、このブロックを簡略化したシステムではその他の可能なシステム リソースを使って完全な IrDA 通信システムを搭載する方法を考慮する必要があります。

IrDA プロトコルは UART 信号に変調スキームを追加したものです。トランスミッタでビットが変調されます。レシーバ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' のビット値はライン上の短い '1' パルスによって通知され、'1' のビット値はラインを '0' に保持することによって通知されます。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 フィールドを設定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設定することで 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、9.6 Kbps で動作します。Figure 16-21 は、UART 転送がどのように IrDA 変調されるかを示します。

Figure 16-21. 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 を設定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 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

1. SCB_CTRL レジスタの MODE (ビット [25:24]) に '10' を書き込んで、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 (ビット [25:24]) に '10' を書き込んで、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 フィルタを有効にします。
4. “UART の有効化と初期化” (150 ページ) の説明に従って、SCB を構成します。

16.3.4 UART レジスタ

UART インターフェースは、Table 16-9 にリストされている一連の 32 ビットレジスタを使用して制御されます。これらのレジスタの詳細については、*PSoC 4100S: PSoc 4 Registers TRM* および *PSoC 4100S Plus: PSoc 4 Registers TRM* を参照してください。

Table 16-9. UART レジスタ

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

16.3.5 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です。この信号は、有効な TX 割り込みソースのいずれかが真の場合に 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
- ☐ アービトラリション ロスト (LIN または SmartCard モードのとき)

■ 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

16.3.6 UART の有効化と初期化

UART を以下の順序でプログラムしてください：

1. Table 16-10 に従って、SCB_UART_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. Table 16-11 に示すように、SCB_TX_CTRL および SCB_RX_CTRL レジスタを使用して、一般的な送信機および受信機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. Table 16-12 に示すように、SCB_TX_FIFO_CTRL および SCB_RX_FIFO_CTRL レジスタをそれぞれ使用して、トランスミッタおよびレシーバ 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. トランスミッタとレシーバ 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。動作モードも選択します (Table 16-13)。
5. ブロックを有効にします ('1' を SCB_CTRL レジスタの ENABLED ビットに書き込みます)。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。ブロックを再度有効にすると、再初期化が行われ、関連する状態 (FIFO コンテンツなど) が失われることに注意してください。

Table 16-10. SCB_UART_CTRL レジスタ

ビット	名称	値	説明
[25:24]	MODE	00	標準 UART
		01	スマートカード
		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 トランスミッタがその相手となるレシーバと通信することを許可	

Table 16-11. SCB_TX_CTRL/SCB_RX_CTRL レジスタ

ビット	名称	説明
[3:0]	DATA_WIDTH	'DATA_WIDTH + 1' はノーです。送信または受信したデータフレームのビット数。有効な範囲は [3, 15]。スタート、ストップとパリティのビットを含まない
8	MSB_FIRST	1 = MSB ファースト 0 = LSB ファースト
9	MEDIAN	これは SCB_RX_CTRL 専用。 3 タップのデジタルメディアン フィルタが入カインターフェース 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が必要とする UART IrDA モードの場合、これは常に '1' である必要があります。 1 = 有効 0 = 無効

Table 16-12. 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 FIFO のエントリ数が少ないまたはレシーバ FIFO のエントリ数が多い場合、それぞれでトランスミッタまたはレシーバトリガー イベントが生成される
16	CLEAR	'1' の場合、トランスミッタまたはレシーバ FIFO およびシフトレジスタはクリア / 無効化されます。
17	FREEZE	'1' の場合、送信機または受信機の FIFO に対するハードウェアの読み取り / 書き込みは影響しません。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

Table 16-13. SCB_CTRL レジスタ

ビット	名称	値	説明
[25:24]	MODE	00	I ² C 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

16.4 インター インテグレートッド サーキット (I²C)

本節は PSoC での I²C 実装を説明します。I²C プロトコル仕様の詳細については、[NXP website](#) に掲載される I²C バス仕様を参照してください。

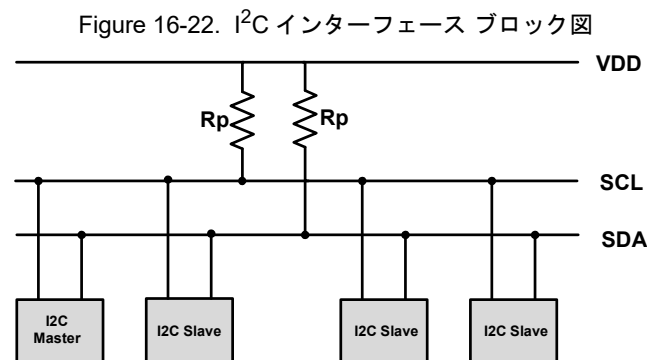
16.4.1 特長

このブロックは以下の機能をサポートします:

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

16.4.2 概要説明

Figure 16-22 に、I²C 通信ネットワークの例を示します。



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) が取り付けられています。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マITTERかレシーバのどちらでも動作できます。バスに接続する各スレーブ デバイスは、他と異なる 7 ビット アドレスをソフトウェアによりアドレス指定します。また、PSoC はファームウェアにより、I²C 向けの 10 ビット アドレス マッチングをサポートします。

16.4.3 用語および定義

Table 16-14 は、I²C 通信ネットワークで一般的に使用される用語について説明しています。

Table 16-14. I²C バス用語の定義

用語	説明
Transmitter (トランスミッター)	データをバスに送信するデバイス
Receiver (レシーバ)	データをバスから受信するデバイス
Master (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
Slave (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
Multi-master (マルチマスター)	メッセージを破損せずにバスを同時に制御できる 2 つ以上のマスター
Arbitration (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
Synchronization (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

16.4.3.1 クロックストレッチ

スレーブ デバイスはデータ処理の準備ができていない場合には SCL ラインを拘束するため「0」に駆動することがあります。I/O 信号インターフェースの実装により、SCL ライン値は「0」になり、他のマスターまたはスレーブが SCL ラインで駆動している値とは無関係です。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、SCL ラインで正のクロックパルス ('1') を生成できないときにそれを検出します。その後マスターデバイスは SCL ラインでのポジティブ エッジの生成を延期し、クロックをストレッチしているスレーブ デバイスと実質的に同期します。

16.4.3.2 バス アービトレーション

I²C プロトコルはマルチマスター、マルチスレーブのインターフェースです。バス調停は 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。たとえば、マスター 1 が SDA ラインの値 '1' を駆動しており、マスター 2 が SDA ラインの値 '0' を駆動している場合、実際のライン値は '0' です。入出力信号インターフェース。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。

16.4.4 I²C 動作モード

I²C は同期のシングル マスター、マルチマスター、マルチスレーブのシリアル インターフェースです。デバイスはマスター モード、スレーブ モード、マスター/スレーブ モードのいずれかで動作します。マスター/スレーブ モードでは、デバイスは指定されるとマスターからスレーブ モードに切り替えます。データ転送中はアクティブになるのは 1 個のシングル マスターのみです。アクティブなマスターは SCL ラインでのクロック駆動を担当します。Table 16-15 に、I²C 動作モードを示します。

Table 16-15. I²C モード

モード	説明
Slave (スレーブ)	スレーブ専用の動作 (初期設定)
Master (マスター)	マスター専用の動作
Multi-master (マルチマスター)	バス上で 2 つ以上のマスターをサポート
Multi-master-slave (マルチマスタースレーブ)	スレーブおよびマルチマスターの同時動作

I²C バスを介したデータ転送は、特定のフォーマットに従います。Table 16-16 に、I²C データ転送の一部であるいくつかの一般的なバスイベントを示します。書き込み転送と読み出し転送のセクションでは、データ転送中の I²C バスビットフォーマットについて説明します。

Table 16-16. 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。
Repeated START (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) の間には変えない

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、マスターは現在の操作が完了するまで待機してから、START 信号を発行

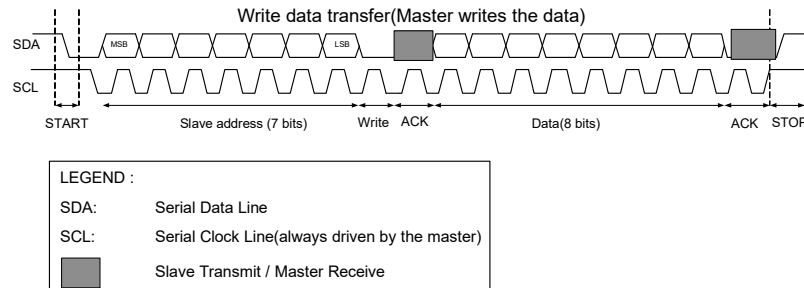
する必要があります (Table 16-16、Figure 16-23、および Figure 16-24 を参照)。マスターは、データ転送開始の合図として STOP 信号を探します。

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調停を失う場合、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。書き込みおよび読み取り転送の例は、1“マスターモード転送の例”(163 ページ)、“スレーブモード転送の例”(165 ページ)、および“マルチマスターモード転送の例”(169 ページ)にあります。

16.4.4.1 書き込み転送

Figure 16-23. マスター書き込みデータ転送

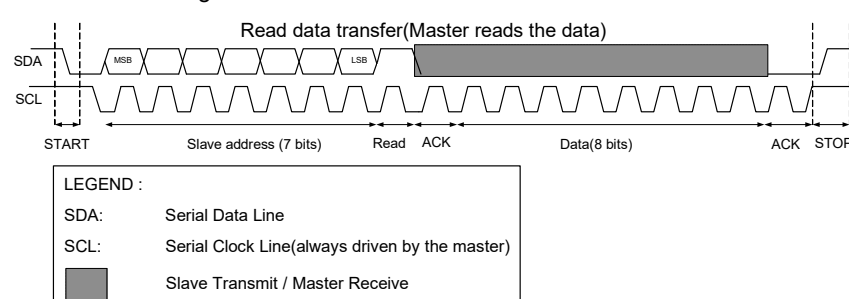


- 通常の書き込み転送はI²Cバス上でSTART条件を生成するマスターから始まります。次に、マスターは、START 条件の後に 7 ビットの I²C スレーブアドレスと書き込みインジケータ ('0') を書き込みます。アドレス指定されたスレーブは、9 番目のビット時間中にデータラインをローにプルすることにより、確認応答バイトを送信します。
- スレーブ アドレスがスレーブ デバイスのいずれにも一致しない場合、またはアドレス指定されたデバイスが要求に確認応答しない場合、そのデバイスは送信 SDA ラインを LOW に引き下げないことにより、確認応答なし (NACK) を送信します。確認応答がない場合、プルアップ抵抗の実装により、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。

16.4.4.2 読み出し転送

Figure 16-24. 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 条件の後に 7 ビット I²C スレーブアドレスと読み取りインジケータ ('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めない場合、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、プルアップ抵抗の実装により、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。

16.4.5 イーザー I2C (EZI2C) プロトコル

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

注: I²C ブロックには、ハードウェア FIFO メモリがあり、16 ビット幅で 16 箇所の深さで、バイト書き込みが可能です。EZ と非 EZ 機能へのアクセス方法は異なります。非 EZ モードで FIFO は TXFIFO と RXFIFO に分けられます。それぞれは 16 ビット幅 8 個の配置を有します。EZ モードでは、FIFO は 8 ビット幅 32 個の配置のシングル メモリ ユニットとして使用されます。

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

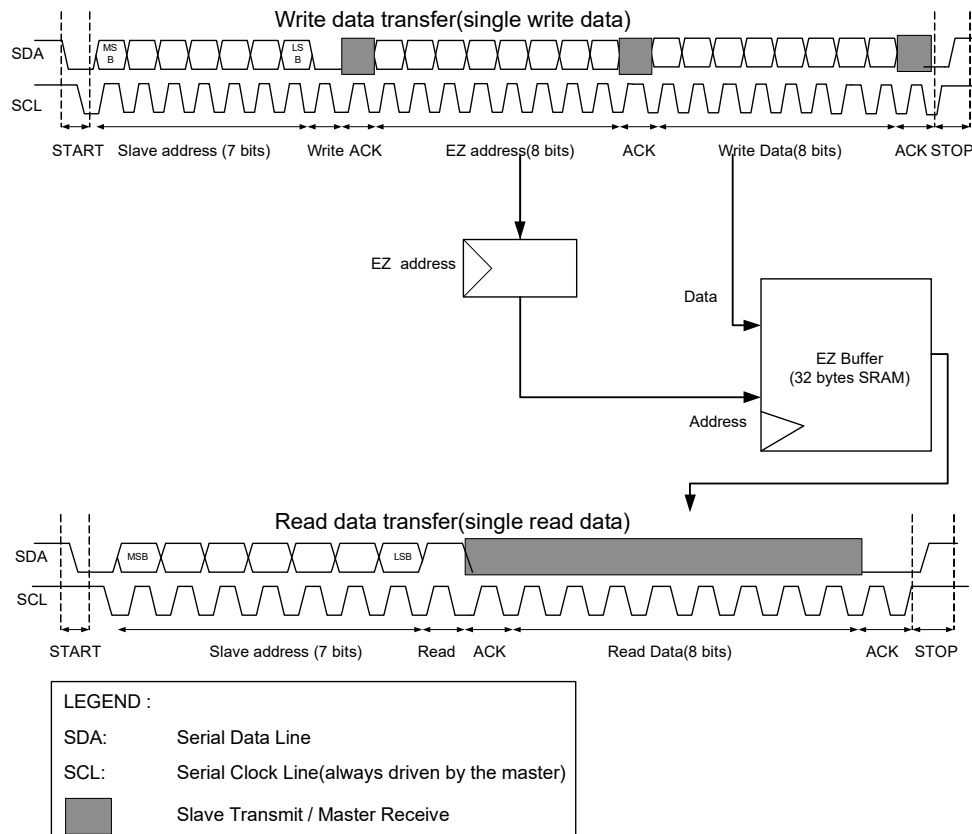
16.4.5.1 メモリ アレイ書き込み

メモリ アレイ インデックスへの EZ 書き込みは I²C 書き込み転送により行われます。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 バッファ量を超える場合には、次のバイトは最後の位置に上書きされます。

16.4.5.2 メモリ アレイの読み出し

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

Figure 16-25. EZI2C 書き込みおよび読み出しデータ転送



16.4.6 I2C レジスタ

I²C インターフェースは、Table 16-17 に示すように、構成、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によって制御されます。

Table 16-17. I2C レジスタ

レジスタ	機能
SCB_CTRL	I2C ブロックを有効にし、シリアル インターフェースのタイプ (SPI、UART、I2C) を選択。内部または外部からのクロック供給動作と EZ モードまたは非 EZ モードを選択するためにも使用
SCB_I2C_CTRL	モード (マスター、スレーブ) を選択しレシーバの 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設定
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 FIFO とシフト レジスタのクリア、トランスミッタ FIFO のフリーズ動作を指定
SCB_TX_FIFO_STATUS	トランスミッタ FIFO に格納されているバイト数、データ フレームをハードウェアで読み出しするロケーション (読み出しポインタ)、新しいデータ フレームを書き込むロケーション (書き込みポインタ) を指定し、トランスミッタ 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 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。メディア フィルタが入カインターフェース ライン上に使用されるかどうかも決定
SCB_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
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
SCB_RX_FIFO_RD	レシーバ 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 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な説明は、*PSoC 4100S: PSoC 4 Registers TRM* および *PSoC 4100S Plus: PSoC 4 Registers TRM* にあります。

16.4.7 I2C 割込み

固定機能 I2C ブロックは、以下の条件のために割込みを生成します。

- I2C マスター
 - I2C マスターが調停を喪失した
 - 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 スレーブが調停を喪失した
 - 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。

割込みの出力はすべての可能な割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込み条件が満たされた時に割込みがトリガーされます。割込みステータス レジスタは割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 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

16.4.8 I2C の有効化および初期化

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

16.4.8.1 I2C 標準 (非 EZ) モード構成

I2C は次の順でプログラムしなければなりません。

1. [Table 16-18](#) に従って、SCB_I2C_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。これはマスター スレーブ機能の選択動作を含んでいます。
2. [Table 16-19](#) に示すように、SCB_TX_CTRL および SCB_RX_CTRL レジスタを使用して、トランスミッタとレシーバの一般的な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. [Table 16-20](#) に示すように、SCB_TX_FIFO_CTRL および SCB_RX_FIFO_CTRL レジスタをそれぞれ使用して、トランスミッタおよびレシーバ 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. トランスミッタとレシーバ 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を [Table 16-21](#) に示します。I2C レジスタの詳細については、[PSoC 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

Table 16-18. SCB_I2C_CTRL レジスタ

ビット	名称	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

Table 16-19. 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、送信または受信されたデータフレームのビット数です。 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ディア 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

Table 16-20. 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 FIFO のエントリ数がより少ないまたはレシーバ 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またはレシーバ トリガー イベントが生成される
16	CLEAR	'1' になると、トランスミッタまたはレシーバ FIFO およびシフトレジスタがクリアされます。
17	FREEZE	'1' の場合、送信機または受信機の FIFO に対するハードウェアの読み取り / 書き込みは影響しません。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

Table 16-21. SCB_CTRL レジスタ

ビット	名称	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

16.4.8.2 EZI2C モード構成

EZI2C モードのために I2C ブロックを設定するには、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' を書き込んで、EZI2C モードを選択します。
2. [I2C 標準 \(非 EZ\) モード構成](#) で説明されている手順 2 ~ 4 に従います。
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

16.4.9 I2C における内部および外部クロック動作

I2C ブロックはデータ レート生成のために内部からクロックを供給される動作も外部からクロックを供給される動作もサポートします。内部クロック動作は PSoC システム バス クロックに引き出したクロック信号を使用します。外部クロック動作はユーザ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です。システムクロックの詳細については、[クロック供給システム \(89 ページ\)](#) 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：

- スレープ機能
- 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が内部 ('0') または外部 ('1') クロックのどちらであるかを示します。
- **EC_OP_MODE (外部クロック動作モード)** : (I2C アドレスの一致以外の) プロトコル動作の残りが内部 ('0') または外部 ('1') クロックのどちらであるかを示します。前述のように、外部クロック動作は非 EZ 機能をサポートしていません。

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。不適切な設定は、特定の電源モードでの誤動作を引き起こす可能性があります。[Table 16-22](#) および [Table 16-23](#) は、EZ モードおよび非 EZ モードでの I2C の設定を説明しています。

16.4.9.1 I2C の非EZ 動作モード

このモードでは FIFO サポートがないため、外部クロック動作は非 EZ 機能に対してサポートされません。したがって、非 EZ モードの場合、EC_OP_MODE は常に '0' に設定する必要があります。ただし、EC_AM_MODE は '0' または '1' に設定できます。[Table 16-22](#) は、可能性の概要を示します。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 は '0' で、EC_OP_MODE は '0' です。

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 は '1' で、EC_OP_MODE は '0' です。

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、割込みを生成し、CPU を復帰させるために使用できる復帰割込みソース ビットをセットします。

Table 16-22. 非 EZ モードでの I2C 動作

I2C (非EZ) モード				
システム消費電力 モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
Active (アクティブ) と Sleep (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。	サポートされていません	
Deep-Sleep (ディープスリープ)	サポートされていません	外部クロックを使用するアドレス一致。 内部クロックを使用する動作。		

- アクティブシステム電源モードでは、CPU はアクティブであり、ウェイクアップ割り込みの原因は無効です (関連付けられた MASK ビットは '0' です)。外部クロック供給回路はアドレス マッチングを担当し、内部クロック供給回路は、残りの I2C 転送を担当します。
- しかしスリープ モードではアプリケーションによって復帰割り込みソースが有効の場合も無効の場合もあります。残りの動作はアクティブ モードに似ています。
- ディープスリープ モードでは、CPU がシャットダウンされ、復帰割り込み要因が有効になっている場合は I2C のアクティビティの際に復帰します。CPU の復帰には時間がかかり、進行中の I2C 転送が確認応答なし (NACK) とされるかクロックがストレッチされます。NACK の場合、内部クロック供給回路は復帰後の最初の I2C 転送を担当します。クロックストレッチの場合、内部クロック供給ロジックは復帰時の進行中 / ストレッチされた転送を担当します。SCB_I2C_CTRL レジスタのレジスタビット S_NOT_READY_ADDR_NACK (ビット 14) は、外部クロックロジックが負の確認応答 ('1') またはクロックストレッチ ('0') のどちらを実行するかを決定します。

16.4.9.2 EZ モードでの I2C 動作

EZ モードでは 3 つの可能な設定があります。EC_OP_MODE が '0' の場合、EC_AM_MODE は '0' または '1' に設定でき、EC_OP_MODE が '1' の場合、EC_AM_MODE は '1' に設定する必要があります。Table 16-23 に、可能性の概要を示します。灰色のセルは、外部クロックロジック (スレーブ選択) から内部クロックロジック (残りの動作) への切り替えを伴うため、可能ですが推奨されない設定を示します。EC_AM_MODE = 0 と EC_OP_MODE = 1 の組み合わせオプシオンは無効でブロックが応答しません。

Table 16-23. EZ モードでの I2C 動作

I2C、EZ モード				
システム消費電力 モード	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
Active (アクティブ) と Sleep (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
Deep-Sleep (ディープスリープ)	サポートされていません	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 は '0' で、EC_OP_MODE は '0' です。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 は '1' で、EC_OP_MODE は '0' です。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 は '1' で、EC_OP_MODE は '1' です。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。この設定により、block's SRAM への外部クロックアクセスが発生することに注意してください。これらのアクセスは、デバイスからの内部的にクロックされるアクセスと競合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可能性があります。SCB_CTRL レジスタのフィールド FIFO_BLOCK (ビット 17) は、待機状態 ('1') またはバスエラー ('0') が生成されるかどうかを決定します。

16.4.10 スリープから復帰

I2C アドレスの一致が発生すると、システムはスリープまたはディープスリープ システム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。このオプシオ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 :I2C がスリープモードへの切り替え中にスレーブアドレスの一致でデバイスをウェイクアップするには、SCB_INTR_I2C_EC レジスタの割り込みビット WAKE_UP (ビット 0) を有効にする必要があります。

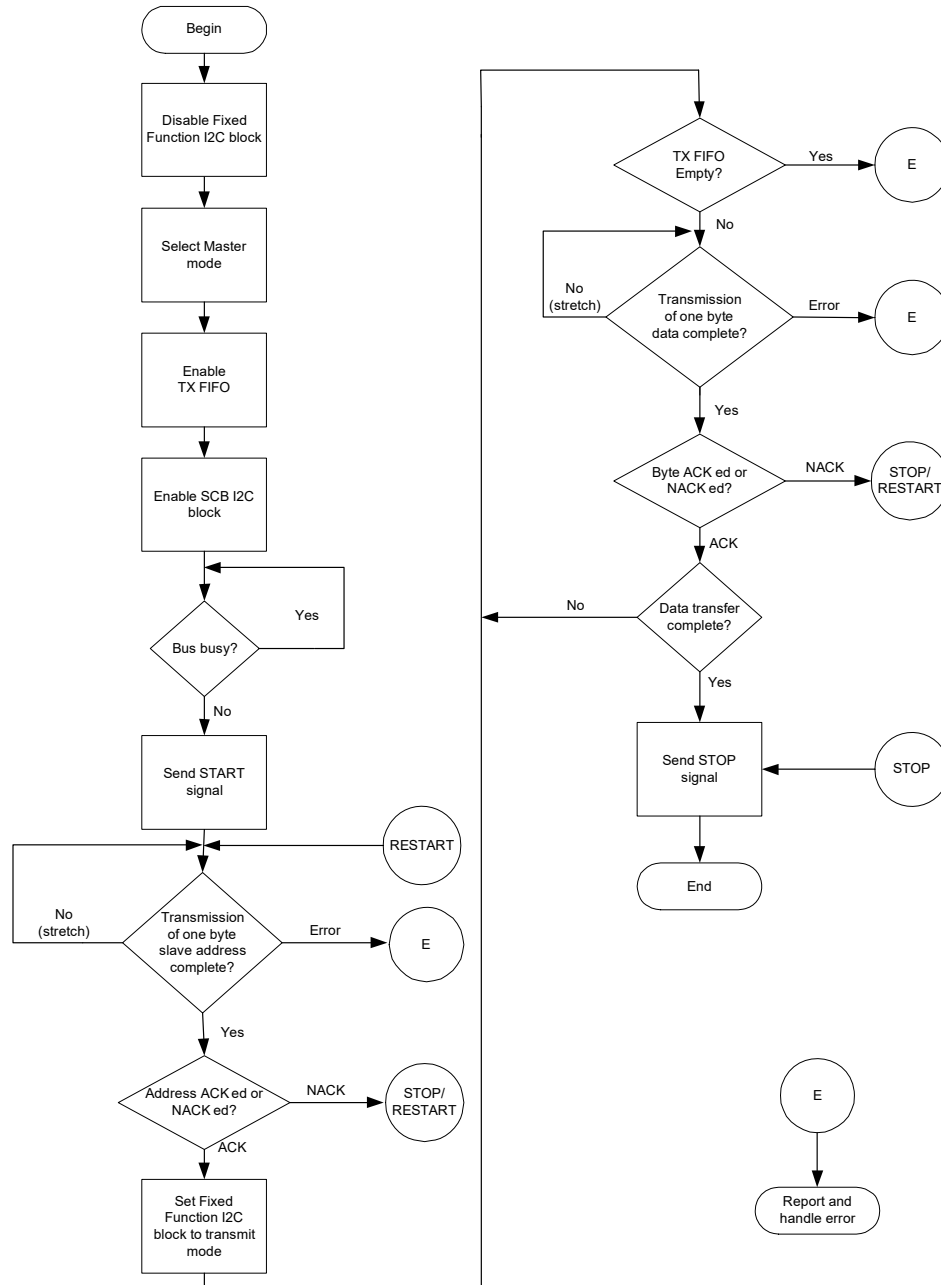
注 : デバイスが I2C スレーブモードで構成されている場合、ディープスリープ電源モードに入るとき、SCB へのクロックを無効にする必要があります。ディープスリープモードからの復帰時にクロックを有効にします。

16.4.11 マスター モード転送の例

マスターがデータを送信または受信します。

16.4.11.1 マスター送信

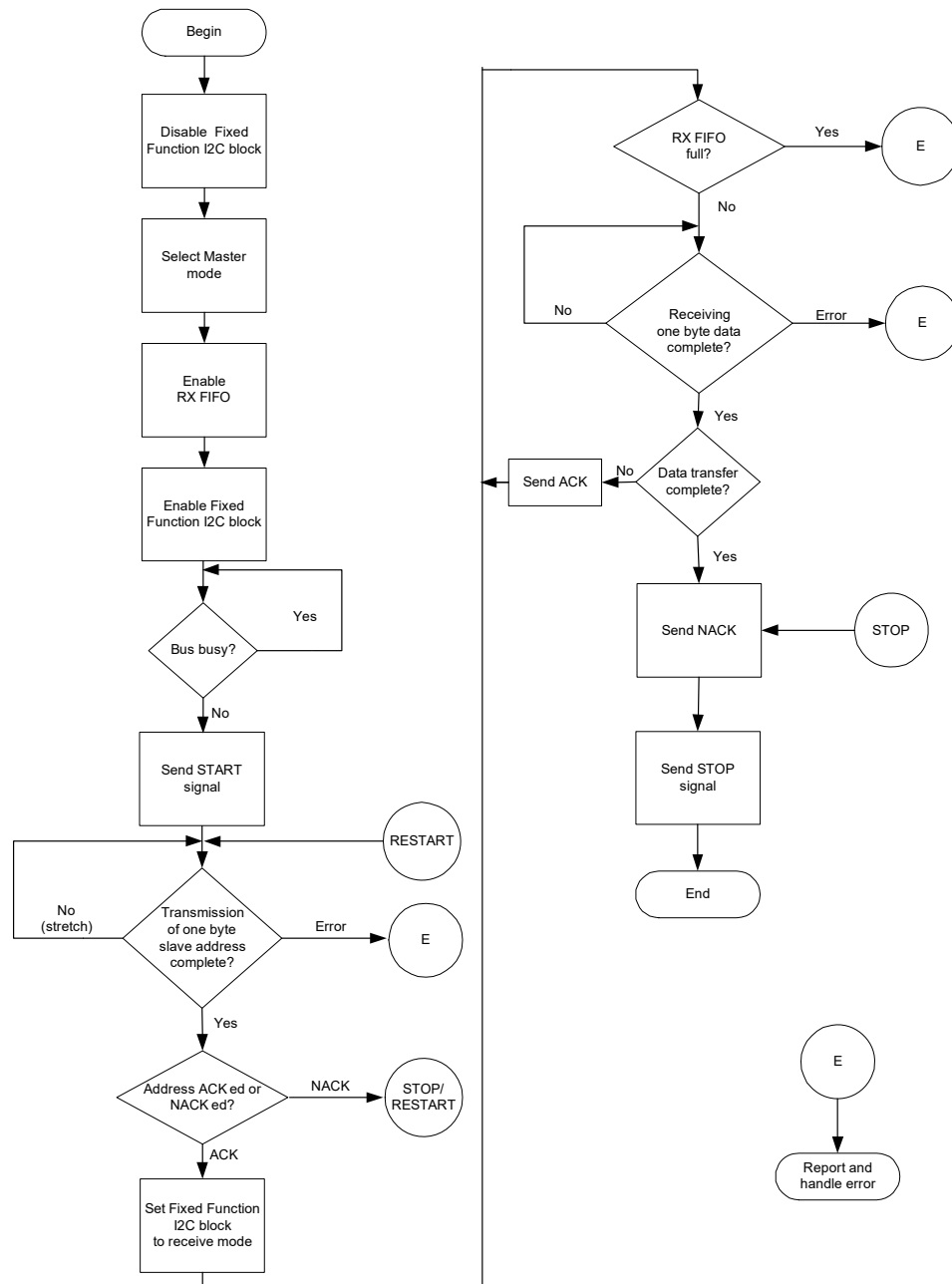
Figure 16-26. シングル マスター モードでの書き込み動作のフローチャート



注： 保留中の転送がない場合、SCB ハードウェアは開始条件を生成して戻ります (ハードウェアが開始条件を生成するまで待機しません)。I2C バスがビジーの場合、バスが解放されるまで、ハードウェアは開始条件を生成しません。SCB ハードウェアは、開始検出後にビジステータスを設定し、停止検出時にそれをクリアします。ESD または他のイベントによって引き起こされるノイズは、バス上で誤ったスタート条件を引き起こす可能性があります。次に、ハードウェアはバスがビジーであると想定するため、マスターは開始条件を生成しません。これが発生した場合、SCB ブロックをリセットする必要があります。

16.4.11.2 マスター受信

Figure 16-27. 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



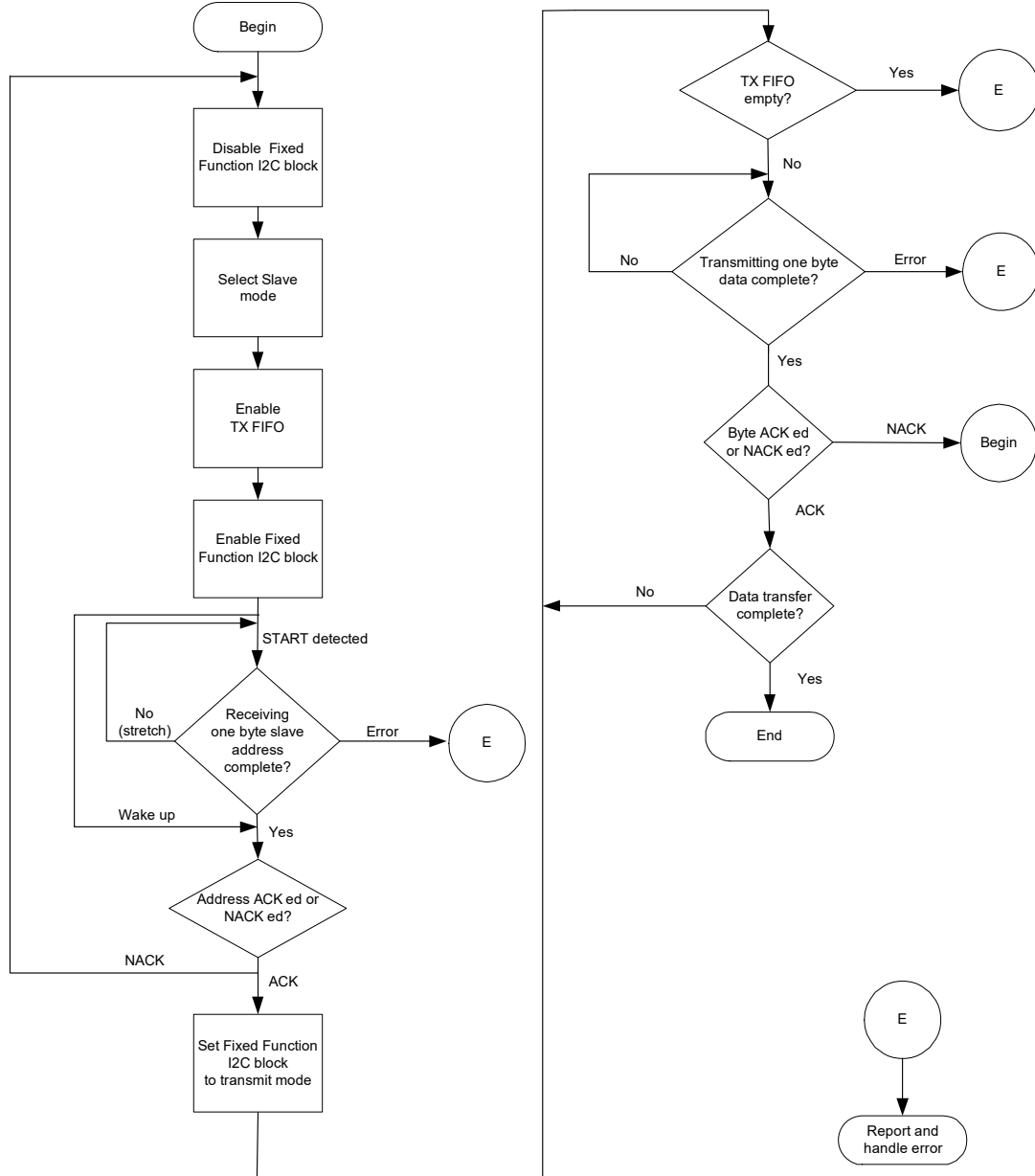
注：保留中の転送がない場合、SCB ハードウェアは開始条件を生成して戻ります（ハードウェアが開始条件を生成するまで待機しません）。I2C バスがビジーの場合、バスが解放されるまで、ハードウェアは開始条件を生成しません。SCB ハードウェアは、開始検出後にビジーステータスを設定し、停止検出時にそれをクリアします。ESD または他のイベントによって引き起こされるノイズは、バス上で誤ったスタート条件を引き起こす可能性があります。次に、ハードウェアはバスがビジーであると想定するため、マスターは開始条件を生成しません。これが発生した場合、SCB ブロックをリセットする必要があります。

16.4.12 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。

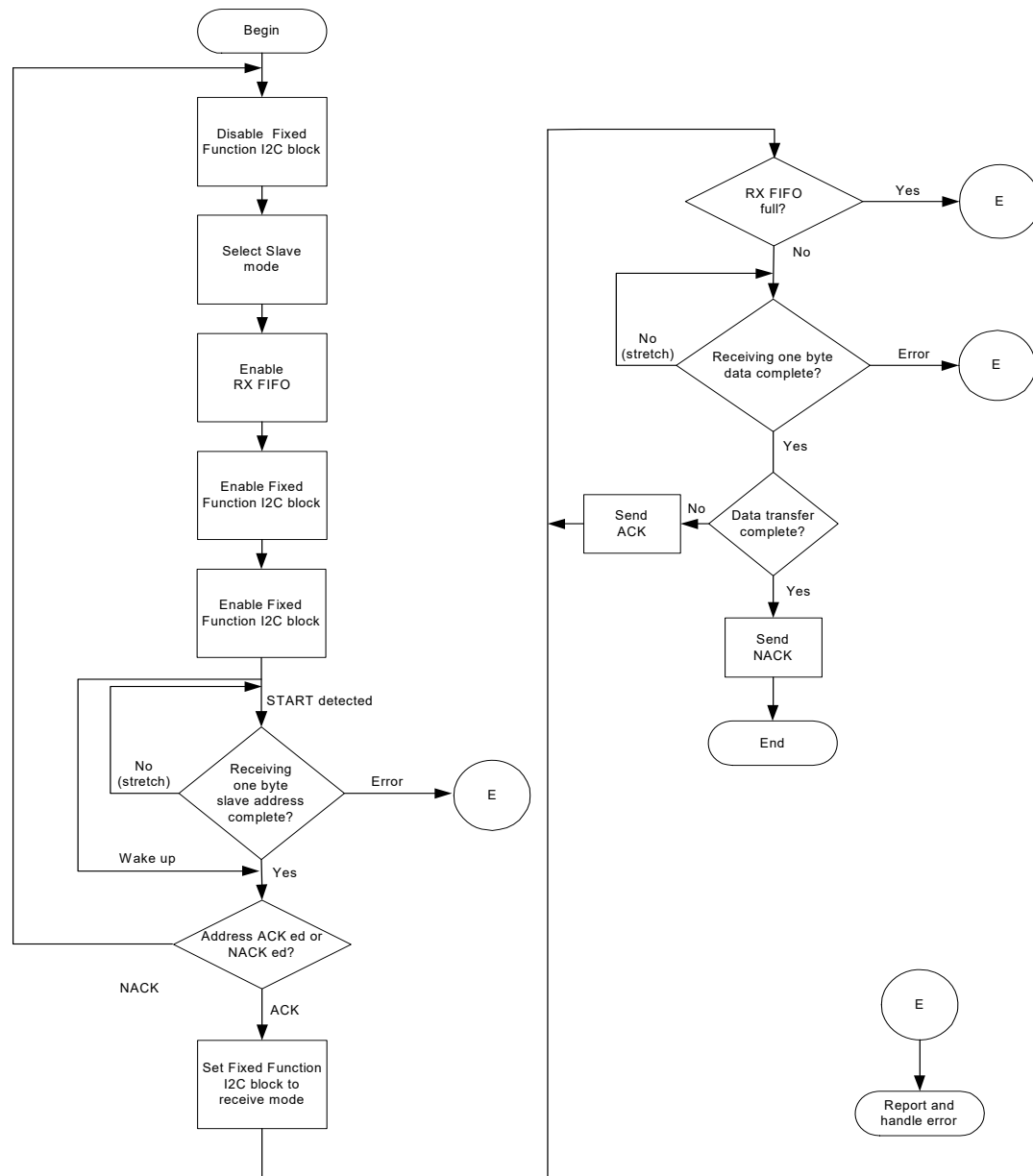
16.4.12.1 スレーブ送信

Figure 16-28. スレーブ モードでの書き込み動作のフローチャート



16.4.12.2 スレーブ受信

Figure 16-29. スレーブ モードでの読み出し動作のフローチャート

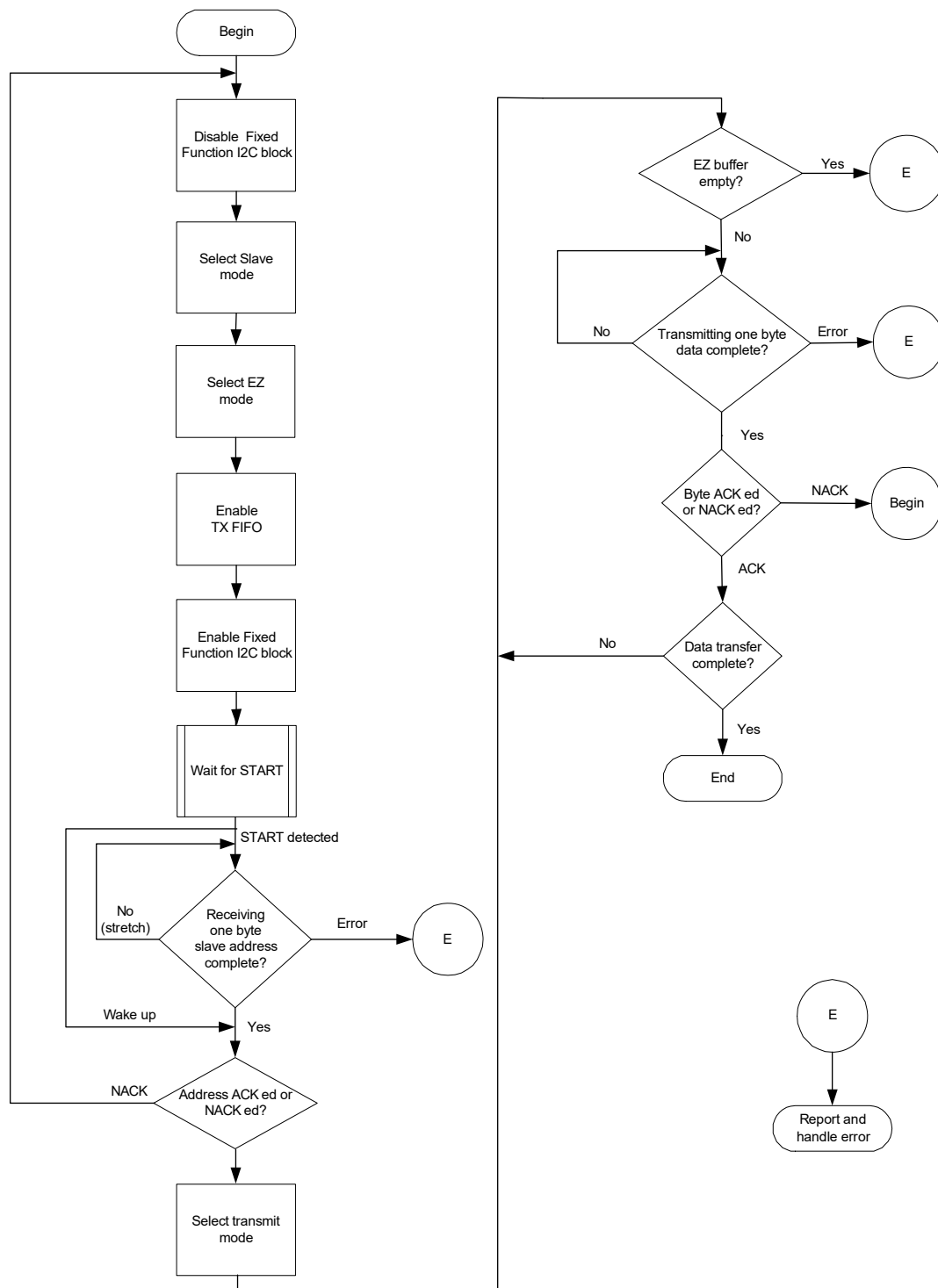


16.4.13 EZ スレーブ モード転送の例

EZ スレーブがデータを送信または受信します。

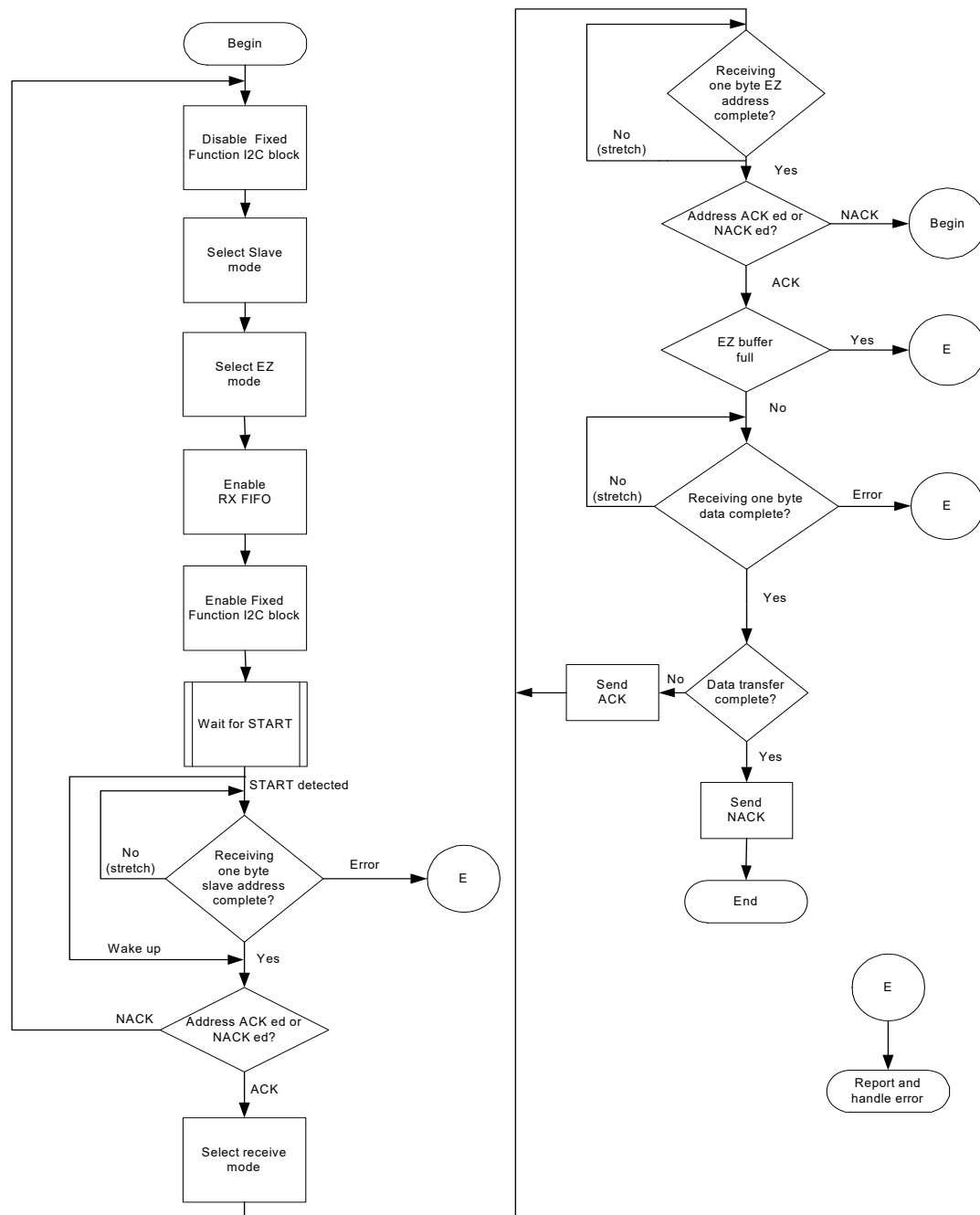
16.4.13.1 EZ スレーブ送信

Figure 16-30. EZI2C スレーブ モードでの書き込み動作のフローチャート



16.4.13.2 EZ スレーブ受信

Figure 16-31. EZI2C スレーブ モードでの読み出し動作のフローチャート

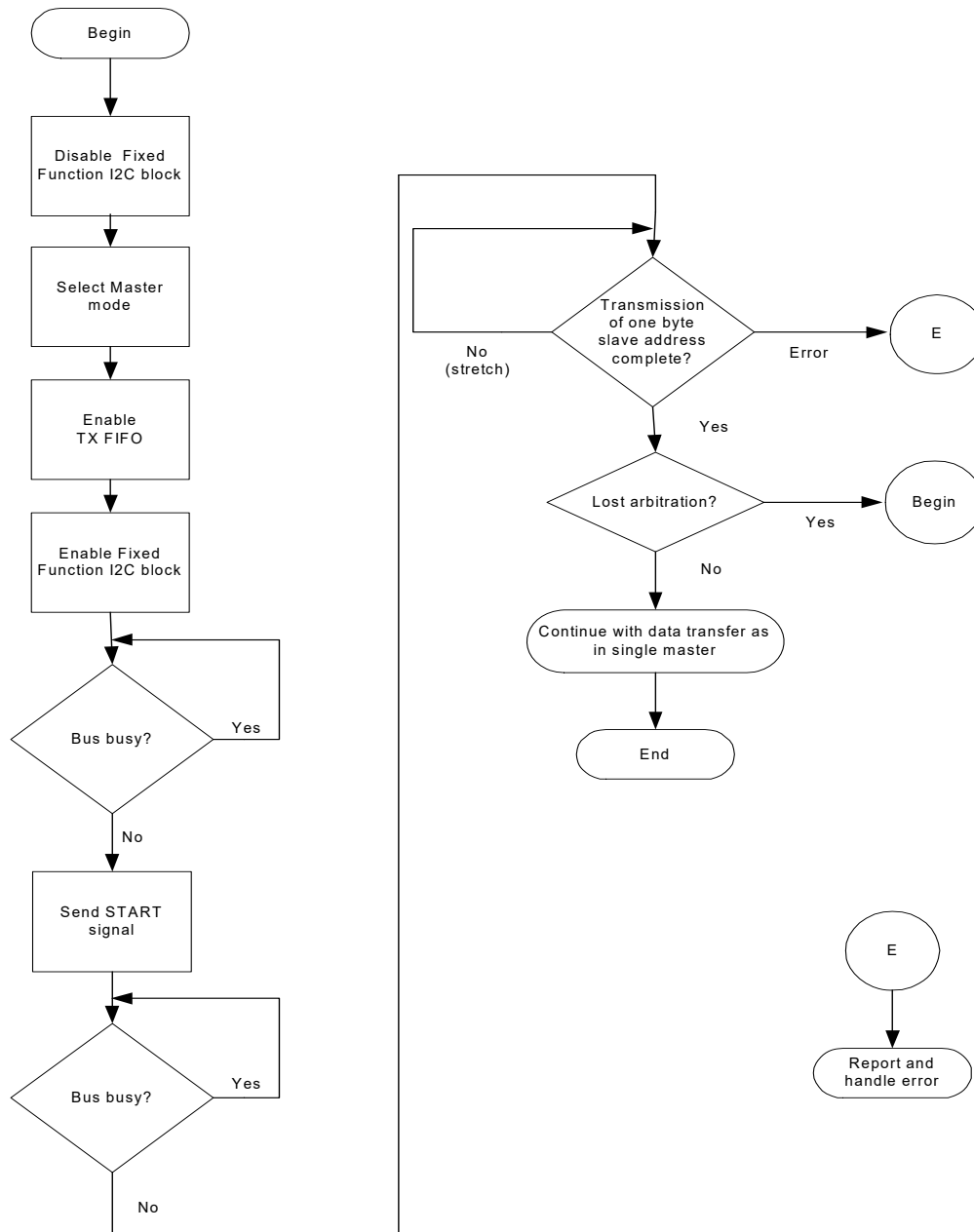


16.4.14 マルチマスター モード転送の例

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

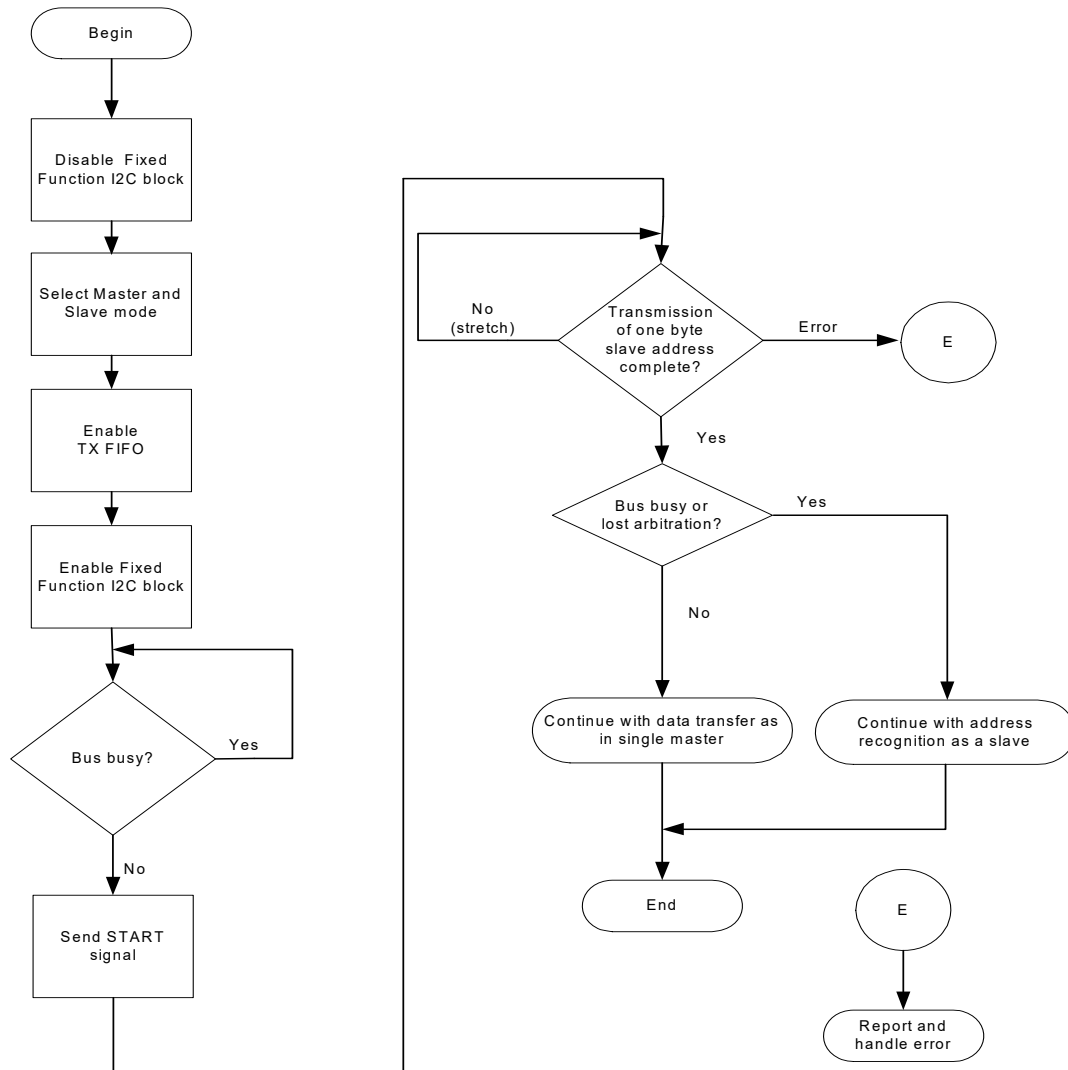
16.4.14.1 マルチマスター - スレーブが無効

Figure 16-32. 「マルチマスター - スレーブが無効」のフローチャート



16.4.14.2 マルチマスター-スレーブが有効

Figure 16-33. 「マルチマスター-スレーブが有効」のフローチャート

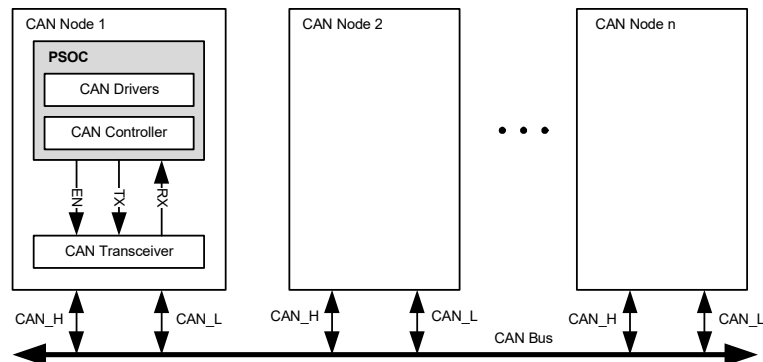


17. コントローラエリアネットワーク (CAN)



CAN ペリフェラルは、最大 1Mbps の通信ポーレートに対応できるフル機能のコントローラエリア ネットワーク (CAN) です。PSoC4100S Plus デバイスには 1 つの CAN コントローラブロックがあり、異なるピンセットにルーティングできます。CAN コントローラブロックは、PSoC 4100S デバイスでは利用できません。これらの CAN コントローラは 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。

Figure 17-1. CAN バスシステムの実装



17.1 特長

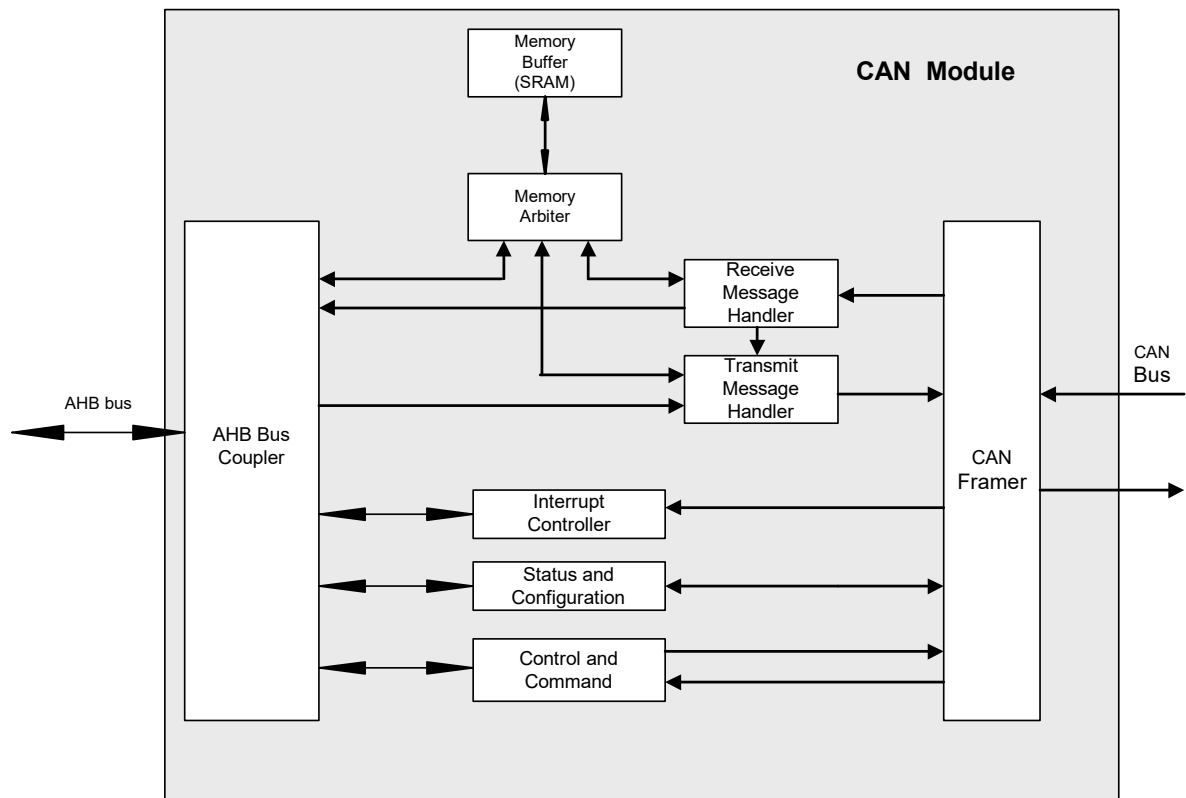
- CAN2.0A/B プロトコル仕様に準拠：
 - 標準と拡張フレーム
 - リモート送信要求 (RTR) をサポート
 - 最大 1Mbps のプログラム可能なビット レート
- 受信パス：
 - 16 個の受信メッセージ バッファ
 - 16 個のアクセプタンス フィルタとアクセプタンス マスク
 - DeviceNet のアドレス指定をサポート
 - ハードウェア FIFO を形成するために複数の受信バッファを連結するオプション
- 送信パス：
 - 8 個の送信メッセージ バッファ
 - 各送信メッセージ バッファ用のプログラム可能な優先順位
 - メッセージのシングル ショット送信をサポート
- 自動ポ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

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ユニットは、CAN モジュールのさまざまな割り込みを処理します。Figure 17-2 は、このプロセスを示しています。

Figure 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
- 過負荷フレーム

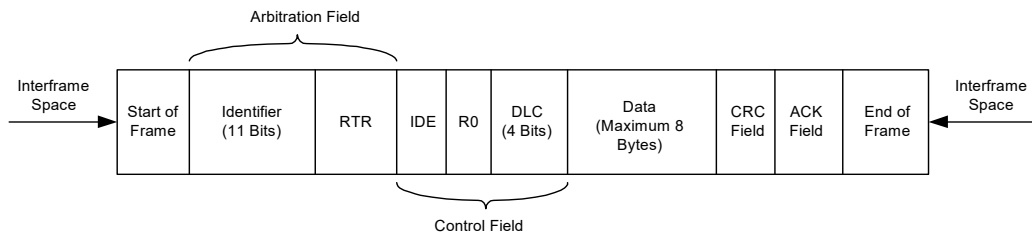
17.3.1 データ フレーム

データ フレームは主にトランスミッタとレシーバ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 フレームの場合、'0' はドミナントビット、'1' はリセツシブビットと呼ばれます。

17.3.1.1 標準データ フレーム

Figure 17-3 は、CAN の標準データフレームを示しています。

Figure 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するために用いられます。

リモート伝送要求ビット (RTR): データフレームの RTR ビット '0' (ドミナント) を設定し、リモートフレームの '1' (劣性) に設定します。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 '0' (標準データフレームの場合は優勢)、拡張 CAN データフレームの場合は '1' (劣性) でなければなりません。

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 デリミタから成ります。

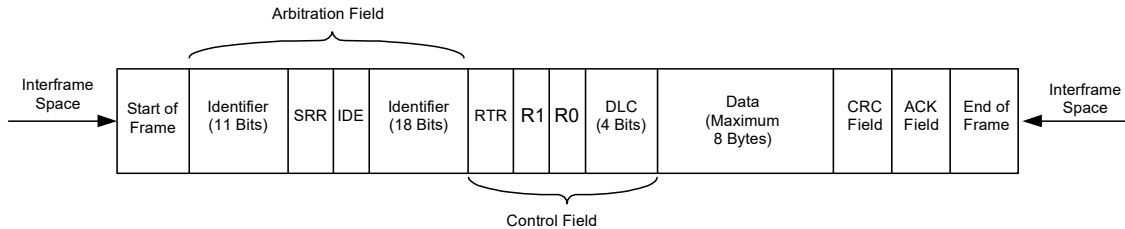
アクリッジ フィールド (ACK): ACK フィールドは 2 ビット長で、初期設定ではリセツシブです。レシーバはメッセージを正しく受信すると、ドミナントビットで ACK フィールドを上書きします。

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 個のリセツ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 フレームフォーマットを Figure 17-4 に示します。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、リセツシブです。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には、標準データフレームと比較して、追加の予約ビット 'R1' があります。

Figure 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 ビットがリセツ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 ビットもリセツシブ 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 個のリセツシブ 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。

エラー区切り文字：エラー デリミタは 8 個のリセツシブ ビットから成ります。ERROR FLAG の送信後、各ステーションは 'recessive' ビットを送信し、'recessive' ビットを検出するまでバスを監視します。その後、さらに 7 つの 'recessive' ビットを送信します。

17.3.4 オーバーロード フレーム

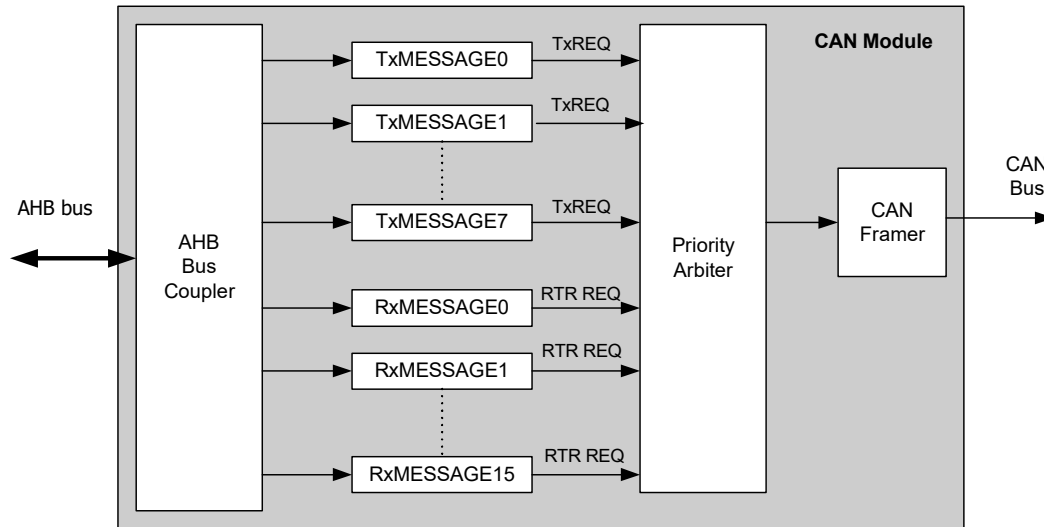
過負荷フレーム (OF) は、過負荷フラグと過負荷区切り文字で構成されます。PSoC 4100S Plus CAN コントローラは、次の条件が発生したときにアクティブになる反応過負荷フレームをサポートします：

- 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 デリミタまたはオーバーロード デリミタ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は、選んだ調停方式に応じてメッセージを選択します。調停方式はラウンド 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 調停ロストがある時、優先順位アービタ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。

Figure 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。調停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。

Fixed Priority: 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。優先順位調停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

Figure 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。

Figure 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 = 送信

標準データ フレームの送信手順を以下に示します。

- 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。
 - 標準データフレームの場合、RTR および IDE ビットに '0' (ドミナント) を書き込みます。
 - 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。
 - 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。
- 適切な優先順位調停方式を選択します。内部優先順位アービタは選んだ調停方式に応じてメッセージを選択します。
- それぞれの TX_REQ ビットを '1' に設定して送信を要求します。
- 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' に設定されている場合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。

注意事項

- 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' の場合、コマンドレジスタのビット [21:16] は保護されており、リードバック時に未定義の値を提供するため、変更できません。
- 書き込み保護ビット wpn1 (CAN_TX [n].CONTROL [3]) が '0' の場合、コマンドレジスタのビット [2] は変更できません。このビットは、リードバック時に '0' を提供します。
- 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、調停ロストやバス 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が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。

送信中に調停ロストやバス 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。

- 拡張日付フレームの場合、IDE ビットに '1' (劣性) を書き込みます。
- 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。

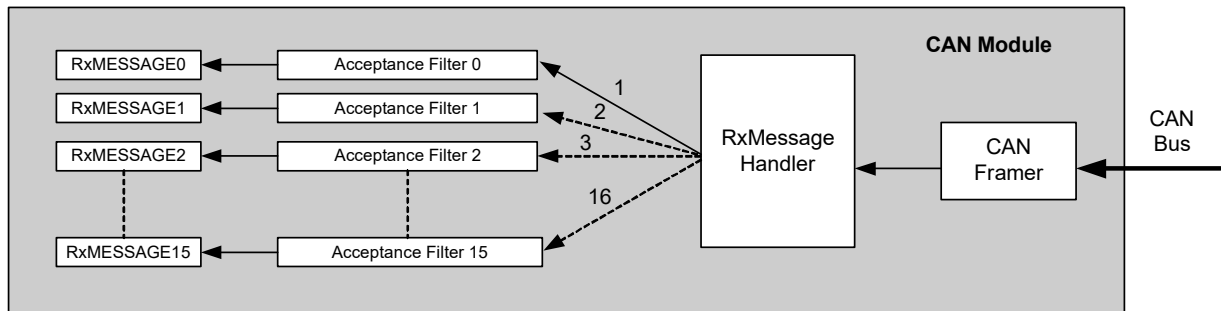
には 1 個の専用アクセプタンス フィルタがあります。CAN メッセージは CAN フレームで受信されます。受信されたメッセージは同時にすべてのアクセプタンス フィルタと比較され、受容されたメッセージは対応する受信メッセージ バッファに格納されます。メッセージ バッファのメッセージ アベイラビリティ (MSG_AV) ビットはセットされて新しいメッセージの存在を示します。別のメッセージを受信するために、MSG_AV ビットをクリアして受信されたメッセージをアクノリッジする必要があります。

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。

17.5 CAN におけるメッセージ受信

CAN モジュールには、Figure 17-7 に示すように、16 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

Figure 17-7. 受信 (Rx) ブロック図



17.5.1 メッセージ受信プロセス

Figure 17-8 は、受信メッセージに関連するレジスタを示しています。

Figure 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 ~ 15

メッセージの受信手順を以下に示します。

1. 新しいメッセージの受信後、RxMessageHandler ハードウェア (Figure 17-7 に示す) は、有効なバッファが見つかるまで、RxMessage0 から始まる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. 受信バッファが有効になっていることは、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 フィルタは着信するメッセージと一致します。
2. RxMessageHandler が空の有効なバッファを見つけると、メッセージが保存され、このバッファの MSG AV ビットが '1' に設定されます。
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 RX_MSG フラグ (CAN_INT_STATUS[12]) はアサートされます。

4. 受信バッファに MSG_AV = '1' で示されるメッセージが既に含まれており、LINK_FLAG ビットが設定されていない場合、RX_MSG_LOSS 割り込みフラグ (CAN_INT_STATUS[10]) がアサートされます。新しく受信したメッセージは破棄されます。

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 受け入れフィルタ

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処理に使用する独自のアクセプタンス フィルタが備えられています。アクセプタンス フィルタはアクセプタンス 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処理されます。

- Identifier (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 のとき、11 ビット識別子は AMR と ACR のビット [31:21] となります。

1. Useful for DeviceNet filtering as given in "DeviceNet フィルタ処理" (179 ページ).

17.5.2.1 例

メッセージとそのメッセージを受け入れるための受け入れフィルタ設定を Figure 17-9 に示します。

Figure 17-9. 受け入れフィルタ

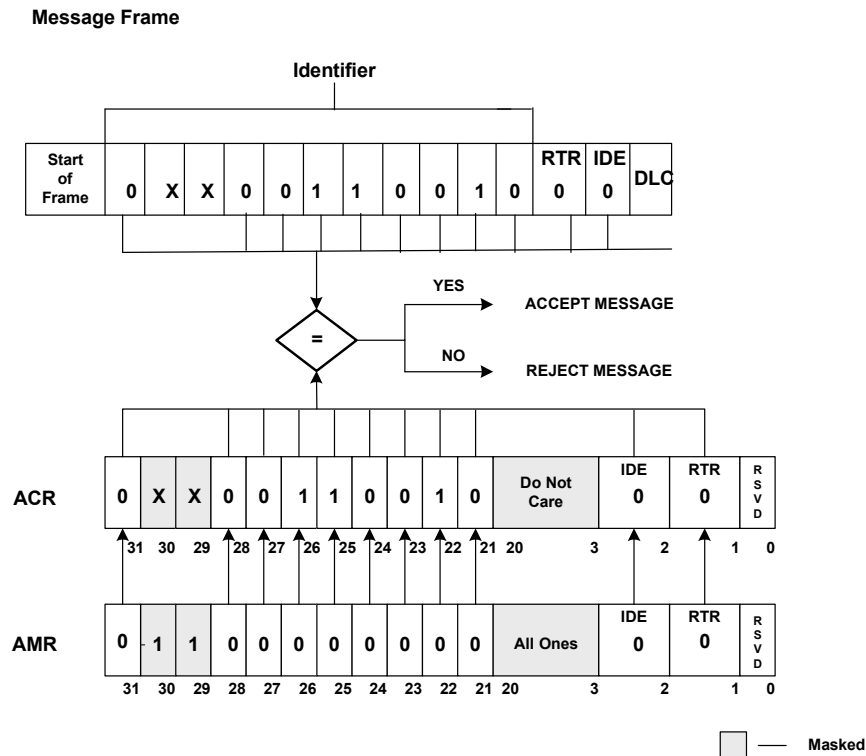


Figure 17-9 に示すように、影付きの領域はマスクされたビットです。AMR レジスタのビットが '1' に設定されている場合、ACR レジスタの対応するビットは受信したメッセージフレームに対してチェックされません。この例では、ビット 30、29、および 3 ~ 20 のビットが '1' に設定され、マスクされています。AMR レジスタの他のビットは '0' として書き込まれるため、Figure 17-9 に示すように、ACR レジスタのそれぞれのビットがメッセージビットと比較されます。ACR のビットがメッセージでの対応するビットと一致する場合、メッセージは受信メッセージバッファに格納されます。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] = All Ones

IDE = 0

RTR = 0

ACR の設定 :

ID[31:21] = 182h

ID[20:3] = Do Not Care

IDE = 0

RTR = 0

17.5.3 DeviceNet フィルタ処理

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 フィルタはこれらの 2 バイトに追加のフィルタ処理を行い、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。“例”(179 ページ) を使用して、DeviceNet フィルタリングを Figure 17-10 に示します。

Figure 17-10. DeviceNet フィルタ

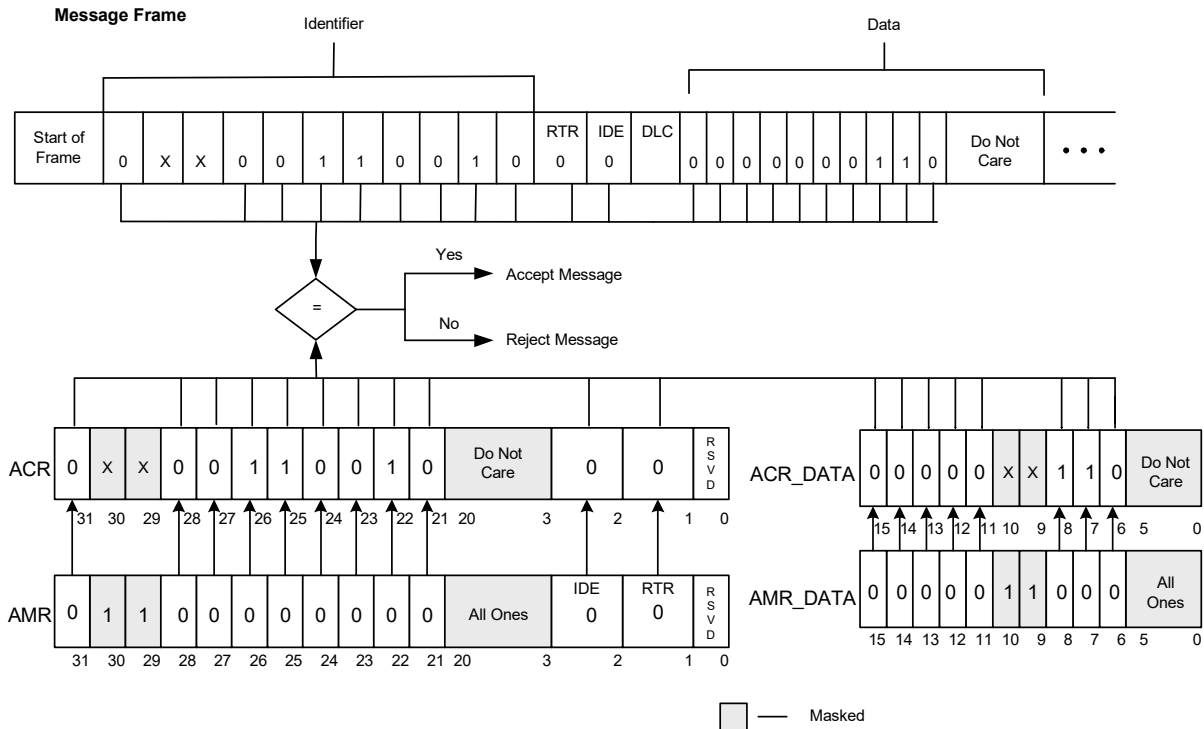


Figure 17-10 では、メッセージフレームのデータフィールドが、AMR_DATA レジスタによってマスクされていない ACR_DATA レジスタのビットと比較されます。

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] = All Ones

IDE = 0

RTR = 0

AMR_DATA[15:11],
AMR_DATA[8:6] = 0

AMR_DATA[10:9],
AMR_DATA[5:0] = All ones

ACR の設定 :

ID[31:21] = 182h

ID[20:3] = Do Not Care

IDE = 0

RTR = 0

ACR_DATA[15:6] = 06h

ACR_DATA[5:0] = Do Not
Care

Figure 17-10 の例は、10 データビットを使用したフィルタリングを示しています。AMR_DATA を使うと、最大 16 個のデータ ビットをフィルタ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処理

拡張データ フレームのフィルタ処理は標準データ 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リンクして、受信 FIFO と同様に機能する受信バッファアレイを形成できます。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。

受信バッファには既にメッセージ (MSG_AV = '1') があり、このバッファに新しいメッセージが到着すると、このメッセージは破棄されます (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。すでにいっぱいになっているものが見つかり (MSG_AV = '1')、

'LINK_FLAG' が設定されている場合、有効な受信バッファの検索が続行されます。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。

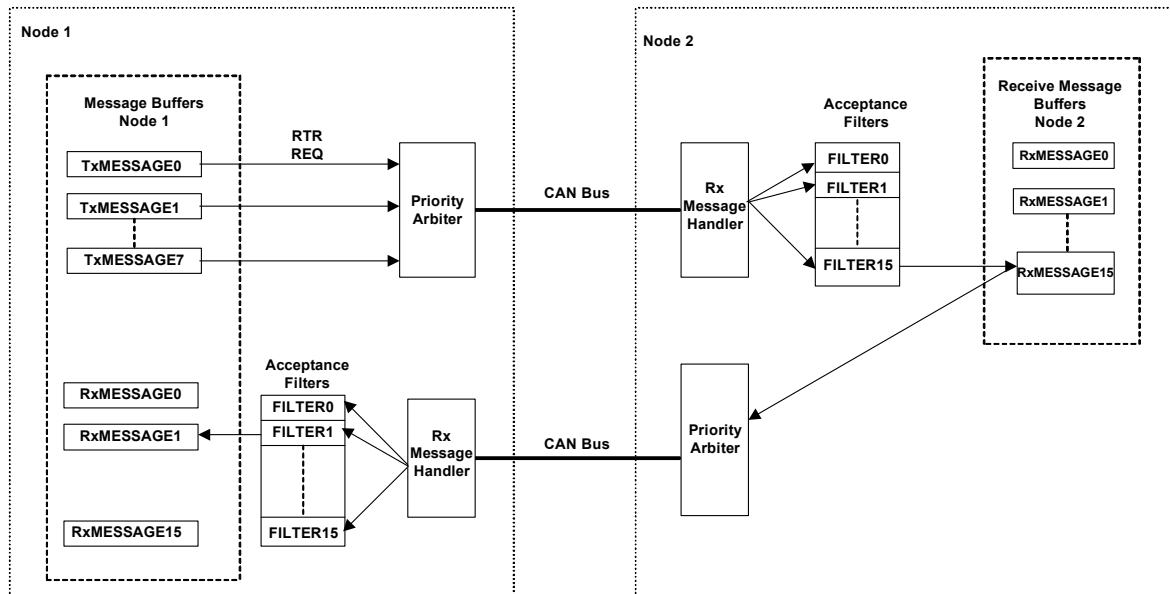
17.6 リモート フレーム

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、RTC ビットが常に '1' に等しく、DLC フィールドの値に関係なくデータフィールドが存在しないという点で、データフレームとは異なります。リモート送信要求のフローを Figure 17-11 に示します。

Figure 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 フィルタ設定はメッセージのアクセプタンス フィルタ設定と一致すれば、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。
- node1 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は、送信されたメッセージノード 1 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。

Figure 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

要求元ノード (Figure 17-11 に示すノード 1) がリモートフレームを送信するプロセスを次に示します。

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 (CAN_TX[n]_CONTROL[21]) を '1' に設定します。
3. 適切な優先順位調停方式を選択します。

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を以下に示します。

1. アクセプタンス フィルタ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. ビット 'RTR REPLY' を '1' に設定して、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 REPLY PNDG FLAG' は、RTR メッセージ要求が受信されたときに設定されます。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。

- IDE ビット (CAN_TX[n].CONTROL [20]) は '1' に設定され、拡張データフレームになります。
- 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。

- 17.4.4 シングル ショット送信で説明されているシングルショット転送の機能。
- TTCAN メッセージのタイミング用の内部タイマ

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

CAN ブロックにある 16 ビット タイマ サブブロックは TTCAN システムのタイマ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 カウントを取り込みます。以下は TTCAN タイマ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 カウンタ 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。詳細については、[17.9.4.2 TT_ENABLE = 1 による割り込みルーティング](#)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。詳細については、[17.9.4.2 TT_ENABLE = 1 による割り込みルーティング](#)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

ギュー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。

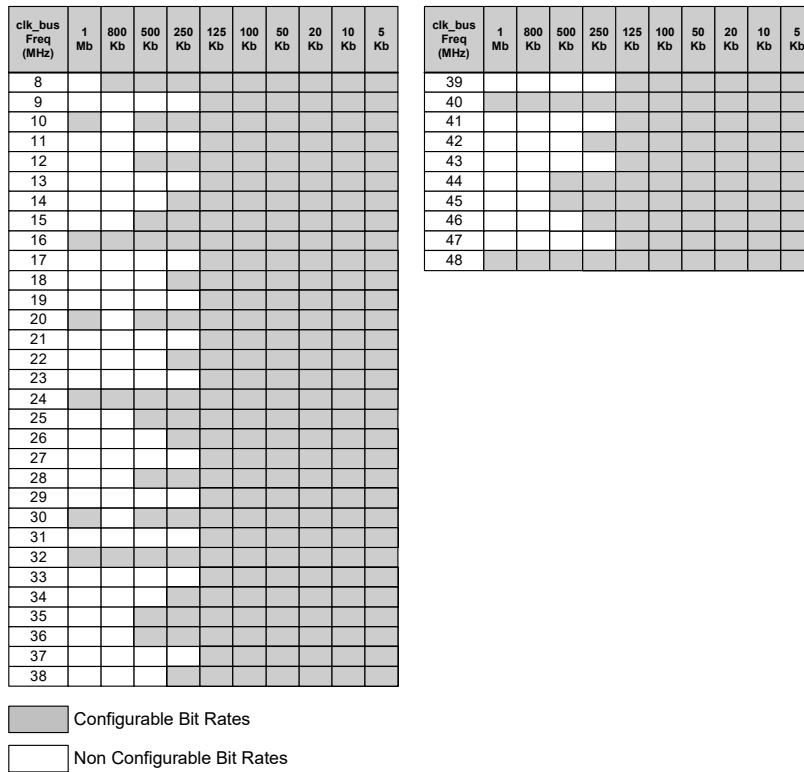
17.8.1 取り得るビット レートとシステム クロック (SYSCLK)

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。

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

SYSCLK が 8 MHz の倍数である場合、これらのビットレートはすべて構成可能です。最大 1 MHz のビットレートをサポートするには、最低 10 MHz の SYSCLK が必要です。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。いくつかの例外はありますが、SYSCLK が 1,000,000 Hz で割り切れない場合、10 ビットレートすべてを使用することはできません。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 ブロックの正確なクロック要件を満たすには、SYSCLK のクロックソースとして外部水晶発振器 (ECO) を使用するか、正確な外部クロックをデバイスにルーティングして SYSCLK ソースとして使用します。Figure 17-12 は、48 MHz から 100 MHz までの任意の fclk 周波数でサポートされる 10 ビットレートの表を示しています。PSoC 4100S Plus の最大可能周波数は 48 MHz です。

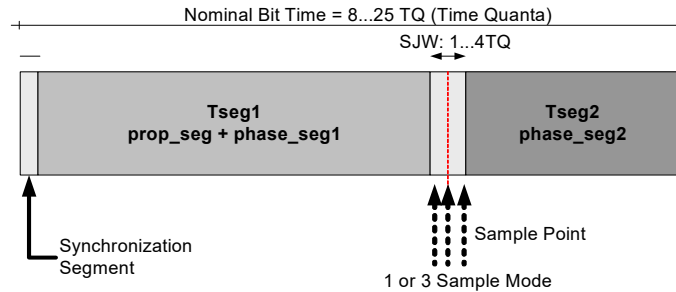
Figure 17-12. ビット レートと SYSCLK



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。[Figure 17-13](#) に示すように、ビット時間は3つのセグメントに分割されます。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。

Figure 17-13. ビット タイム



$$BitTime = (1 + tseg1 + 1 + tseg2 + 1) TQ \quad \text{Equation 17-1}$$

$$TQ = \frac{BRP+1}{SYSCLK} \quad \text{Equation 17-2}$$

注：ビット レート プリスケアラは 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。[Figure 17-14](#) を参照。

同期セグメント：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。

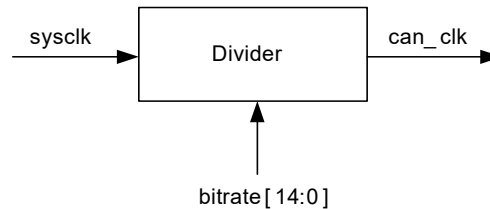
Tseg1, Tseg2: これらのセグメントはエッジ位相誤差を補正します。tseg1 はネットワークにおける遅延時間を含む伝播時間を含みます。セグメントの長さはエッジの位相誤差を補正するために増減されます。これは再同期化と呼ばれています。

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くしてください。

コンフィギュレーション レジスタ CAN_CONFIG はビット レート プリスケアラ (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。システムクロックを生成するために使用できるオプションの詳細については、[クロック供給システム \(89 ページ\)](#) 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。

Figure 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。

$$BRP = ((\text{タイム クォンタム} * \text{SYSCLK}) - 1) = 4$$

したがって、'4' の値を構成レジスタの 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の同期ジャンプ幅を修正するには、ビット CFG_SJW = '1' に書き込むことにより、値 '1' を使用します。

ビット cfg_tseg2 に '2' の値を書き込んで、tseg2 の値を 2TQ に設定します。

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

ここに $tseg1 = 3TQ$ 。

したがって、'3' の値を構成レジスタのビット cfg_tseg1 に書き込みます。

この手順は、Figure 17-12 で指定されているクロック周波数を使用し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 のエラーと割り込み

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。例外は、調停フィールドの詰め込まれたビットストリーム中または ACK スロット中に 'recessive' ビットを送信することです。トランスミッタはパッシブエラーフラグを送信し、'dominant' ビットを検出しても、これを BIT エラーとして解釈しません。

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シブ ビットを送信するトランスミッタは、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 4100S Plus CAN コントローラには、CAN バス診断に使用できる専用のエラーキャプチャレジスタ (ECR) があります。エラーキャプチャレジスタは、Figure 17-15 に示すとおりです。

Figure 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。連続するエラーキャプチャの場合、エラーキャプチャレジスタに '1' を書き込むことにより、ECR を再び準備する必要があります。

注: IDE ビットは ECR.FIELD の調停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。

- **Error Active:** 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。
- **Error Passive:** 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。
- **Bus Off:** 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 つのカウンタの値によって判定されます。

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

エラー カウンタは 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と受信エラーカウンタが 10 進数で 127 以下の場合、ノードは 'error active' 状態です。送信または受信エラーカウンタの値が 10 進数で 128 以上の場合、ノードは 'error passive' 状態です。送信エラーカウンタが 10 進数の 256 の値以上の場合、ノードは 'Bus Off' 状態です。

送信エラー数と受信エラー数の両方が 127 以下の場合、'error passive' ノードは再び 'error active' になります。

'Bus Off' 状態のノードは、11 の連続する 'recessive' ビットの 128 回の発生がバスで監視された後、エラーカウンタが両方とも '0' に設定された 'error active' になります。

エラーステータスレジスタには 2 つのビットがあります: 'txgte96' (CAN_ERROR_STATUS[18]) および 'rxgte96' (CAN_ERROR_STATUS[19])。これらのビットは送信エラー カウンタと受信エラー カウンタそれぞれが 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

コア割り込みは、Figure 17-16 に示すように、割り込みステータス (CAN_INT_STATUS) および割り込みイネーブルレジスタ (CAN_INT_EBL) によって制御されます。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。ビットが設定されると、'1' を書き込んでクリアされるまで、ビットは設定されたままです。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。

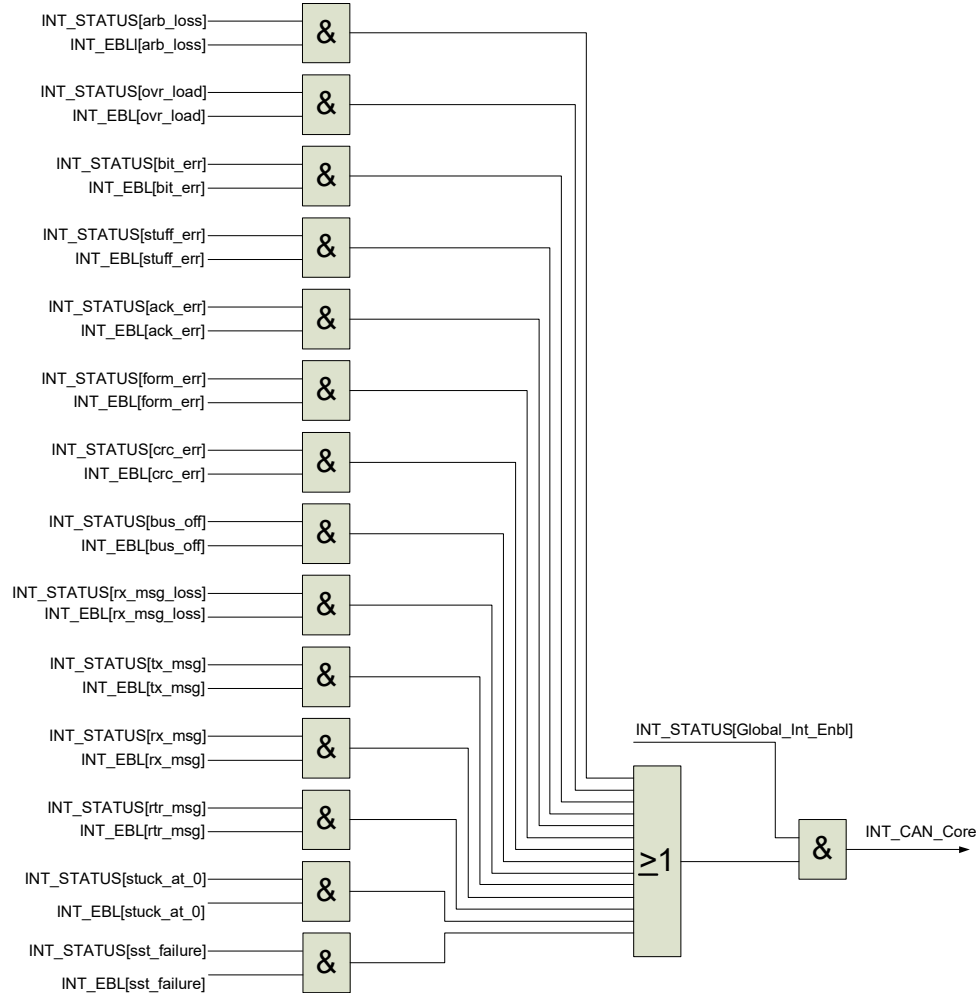
割り込みイネーブル レジスタ (INT_EBL) は割り込みステータス レジスタのどのビットを割り込み出力 (INT_CAN_Core) をアサートするために使用するかを制御します。INT_CAN_Core は、特定の割り込みステータス ビットと対応するイネーブル ビットがセットされるとアサートされます。INT_CAN_Core の送信は CAN_CNTL.TT_ENABLE ビットで制御されます。TT_ENABLE ビットが「0」であれば INT_CAN_Core 信号は CAN ブロック割り込み出力に直接接続されます。TT_ENABLE ビットが設定 (1) されている場合、Figure 17-11 に示すように、INTR_CAN_SET レジスタ設定に基づいて割り込みがルーティングされます。

CAN におけるさまざまなコア割り込みソースを以下に示します。

- 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到着したときに設定されますが、RxMessage フラグ MSG_AV が設定され、LINK_FLAG ビットは設定されません。保存するバッファがないため、新しいメッセージは破棄されます。
- 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調停に負けた。
- 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。

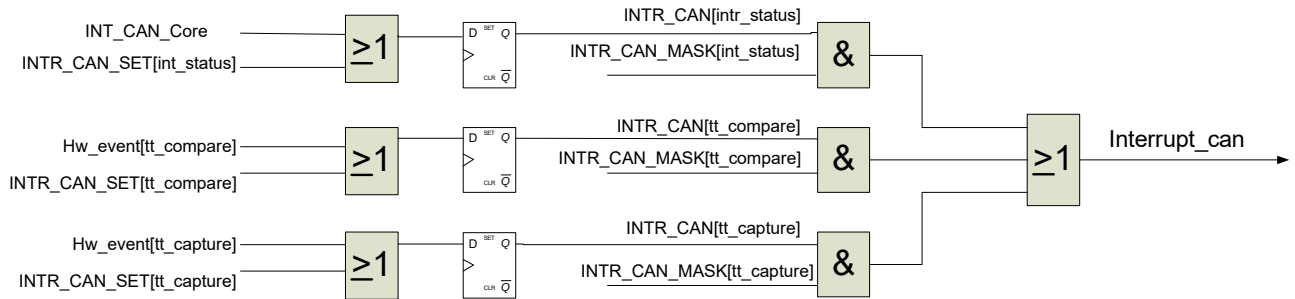
Figure 17-16. CAN コアからの割り込み



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' に設定されている場合、CAN 割り込み信号は、Figure 17-17 に示すように、INT_CAN_SET および INT_CAN_MASK レジスタ設定に基づいて、CAN コア割り込みまたは追加の TTCAN タイマハードウェア割り込みから生成されます。

Figure 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 つの動作モードを以下に示します。

- 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 コントローラは正常に動作します。CAN コントローラは 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 コントローラはバス上で受信されたメッセージをアキュリッジせずに 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を以下に示します。

1. CAN コントローラは、すべてのメッセージを受け入れるように初期化されています (グローバル / ローカルマスクは '0' に設定されています)。
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。コマンドレジスタの LOOPBACK ビット (CAN_COMMAND[2]) と LISTEN ビット (CAN_COMMAND[1]) の設定に基づいて、2 種類のループバックモードが PSoC 4100S Plus CAN コントローラブロックでサポートされています。

17.10.3.1 外部ループバック モード

COMMAND_LOOPBACK = 1 および COMMAND_LISTEN = 0 の場合、外部ループバックモードが有効になります。このモードでは、CAN_TX 出力ピンを外部で CAN_RX 入力ピンに接続できます。この IP は、送信されたトランザクションを処理して受信できます。

17.10.3.2 内部ループバック モード

COMMAND_LOOPBACK = 1 および COMMAND_LISTEN = 1 の場合、内部ループバックモードが有効になります。このモードでは、送信されたトランザクションは内部で受信機ロジックにルーティングされます。

18. タイマ、カウンタ、および PWM



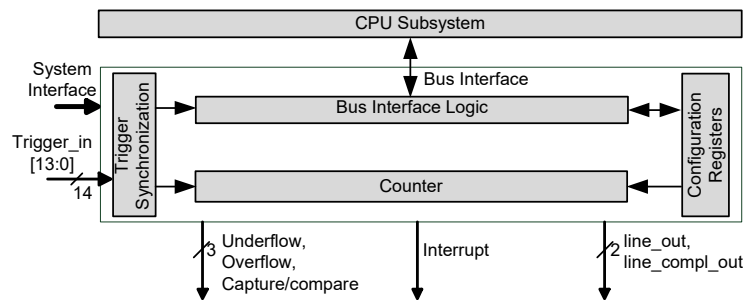
PSoC®4 内のタイマ、カウンタ、パルス幅変調器 (TCPWM) ブロックは 16 ビット タイマ、カウンタ、パルス幅変調器 (PWM)、直交デコーダの機能を搭載しています。ブロックは入力信号の周期とパルス幅の測定 (タイマ) や特定のイベントが発生する回数のカウント (カウンタ)、PWM 信号の生成、直交信号の復号に使用します。本章は TCPWM ブロックの機能、実装、動作モードを説明します。PSoC 4 MCU の TCPWM の最大数は部品番号によって異なります。TCPWM の数と TCPWM ピンの位置を確認するには、デバイスのデータシートを参照してください。

18.1 特長

- 最大 8 個の 16 ビットタイマ、カウンタ、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：
 - Timer (タイマ)
 - Capture (取り込み)
 - Quadrature decoding (直交復号)
 - Pulse width modulation (パルス幅変調)
 - Pseudo-random PWM (疑似乱数 PWM)
 - PWM with dead time (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込みに対応：
 - ターミナル カウント – カウンタ レジスタの最終値に到達する
 - 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
- アンダーフロー、オーバーフローおよびキャプチャ/比較の出力信号、これらは SAR ADC をトリガーするために配線可能
- PWM のコンプリメンタリ出力
- 他の TCPWM のアンダーフロー、比較一致またはオーバーフローの信号、SAR ADC の変換終了 (EOC) または Sample_Done 信号、CTBm コンパレータ出力または低消費電力コンパレータ (LPCOMP) 出力信号から、および専用 GPIO の立ち上がりエッジ、立ち下りエッジ、両エッジおよびトリガー レベル オプションから選択可能なスタート、リロード、ストップ、カウントおよび TCPWM によるキャプチャ イベント信号

18.2 ブロック図

Figure 18-1. TCPWM ブロック図



ブロック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。
- I/O 信号インターフェース：入力トリガー（リロード、スタート、ストップ、カウントおよびキャプチャ）を専用 GPIO に接続します。
- 割込み：ターミナル カウント (TC) または CC 条件をベースとするカウンタからの割込み要求信号を提供します。
- 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。このブロックに必要なすべてのレジスタの詳細については、“TCPWM レジスタ” (217 ページ) 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の有効化 / 無効化

カウンタは制御レジスタ 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：カウンタは、有効にする前に構成する必要があります。設定後、カウンタを有効にすると、レジスタは新しい設定で更新されます。カウンタが無効になると、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。

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。カウンタが有効になると生成されたカウンタ イネーブル信号 (counter_en) はカウンタ固有のクロック (counter_clock) を供給するために HFCLK をゲート制御します。本章の後半で説明する出力トリガーも HFCLK と同期化されます。

クロック分周：counter_clock は 1、2、4... 64、128 の分周比で分周できます。この事前スケーリングは、Table 18-1 に示すように、カウンタ制御 (TCPWM_CNT_CTRL) レジスタの GENERIC フィールドを変更することによって行われます。

Table 18-1. カウンタ 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 分周)
1	2 分周
2	4 分周
3	8 分周
4	16 分周
5	32 分周
6	64 分周
7	128 分周

注：クロックプリスケールは、直交モードおよび PWM-DT モードでは実行できません。

18.2.3 トリガー入力に基づいたイベント

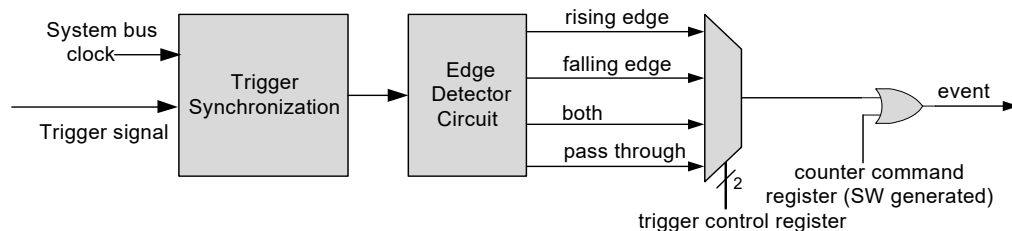
以下はハードウェアまたはソフトウェアによってトリガーされるイベントです。

- Reload (リロード)
- Start (開始)
- Stop (停止)
- Count (カウント)
- Capture/switch (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。Figure 18-2 は、任意のイベントトリガー信号のエッジ検出タイプの選択を示しています。

トリガー制御レジスタ 1 (TCPWM_CNT_TR_CTRL1) を構成することにより、イベントの発生に対して任意のエッジ (立ち上がり、立ち下がり、またはその両方) またはレベル (高) を選択できます。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、Figure 18-2 に示すように、ファームウェアはカウンタコマンドレジスタ (TCPWM_CMD) に書き込むことでイベントを生成できます。

Figure 18-2. トリガー信号エッジ検出



イベントを生成するトリガー信号は、GPIO 信号、TCPWM's アンダーフロー、比較一致またはオーバーフロー信号、SAR ADC の比較終了 (EOC) または Sample_Done 信号、CTBm コンパレータ出力、または低電力コンパレータ (LPCOMP) です。) 出力信号。Figure 18-3 は、すべてのイベントのトリガー信号の選択を示しています。

Figure 18-3. PSoC 4100S の Mux のトリガー

To use GPIOs for trigger, HSIOM_PORT_SELx register should be written

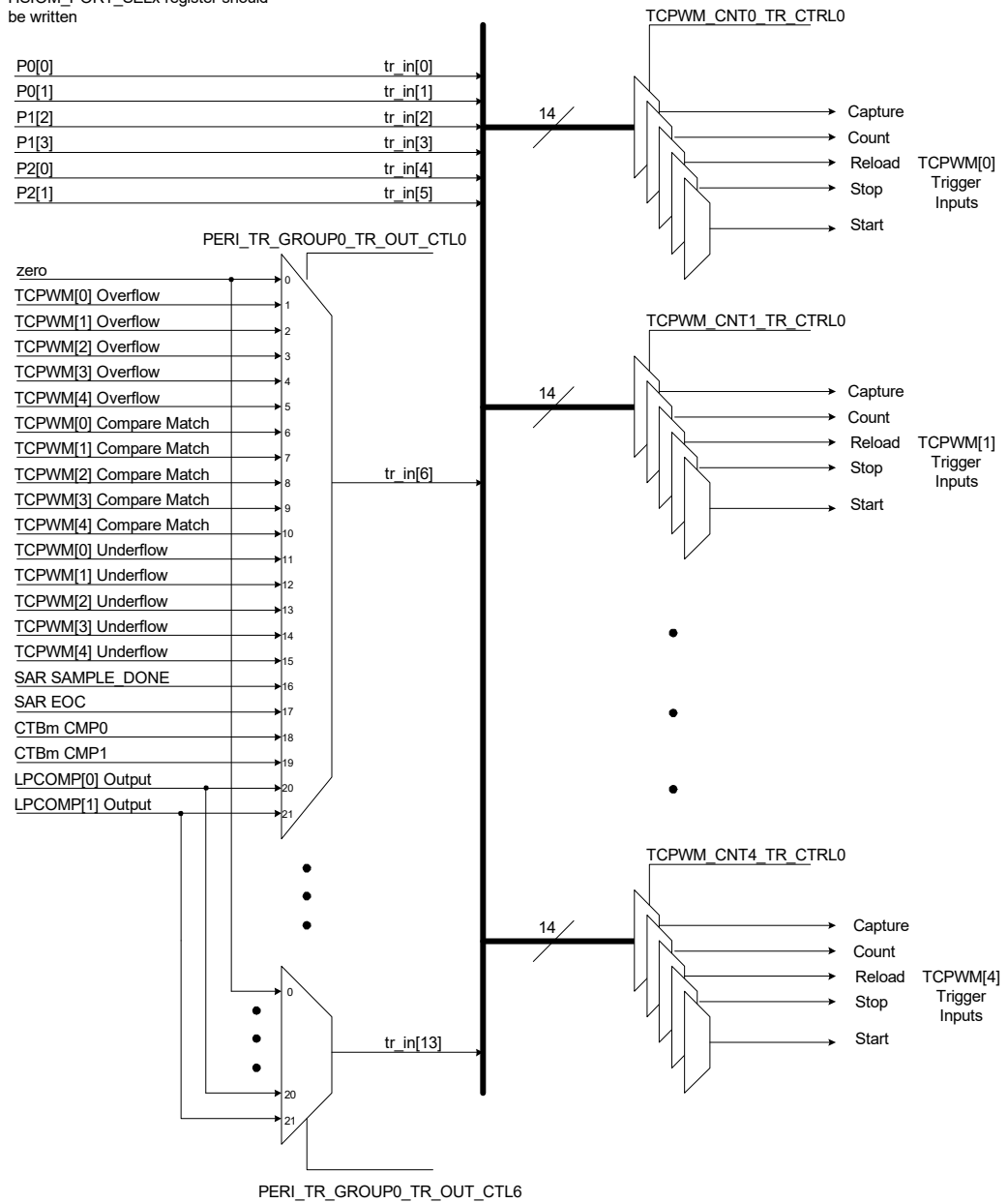
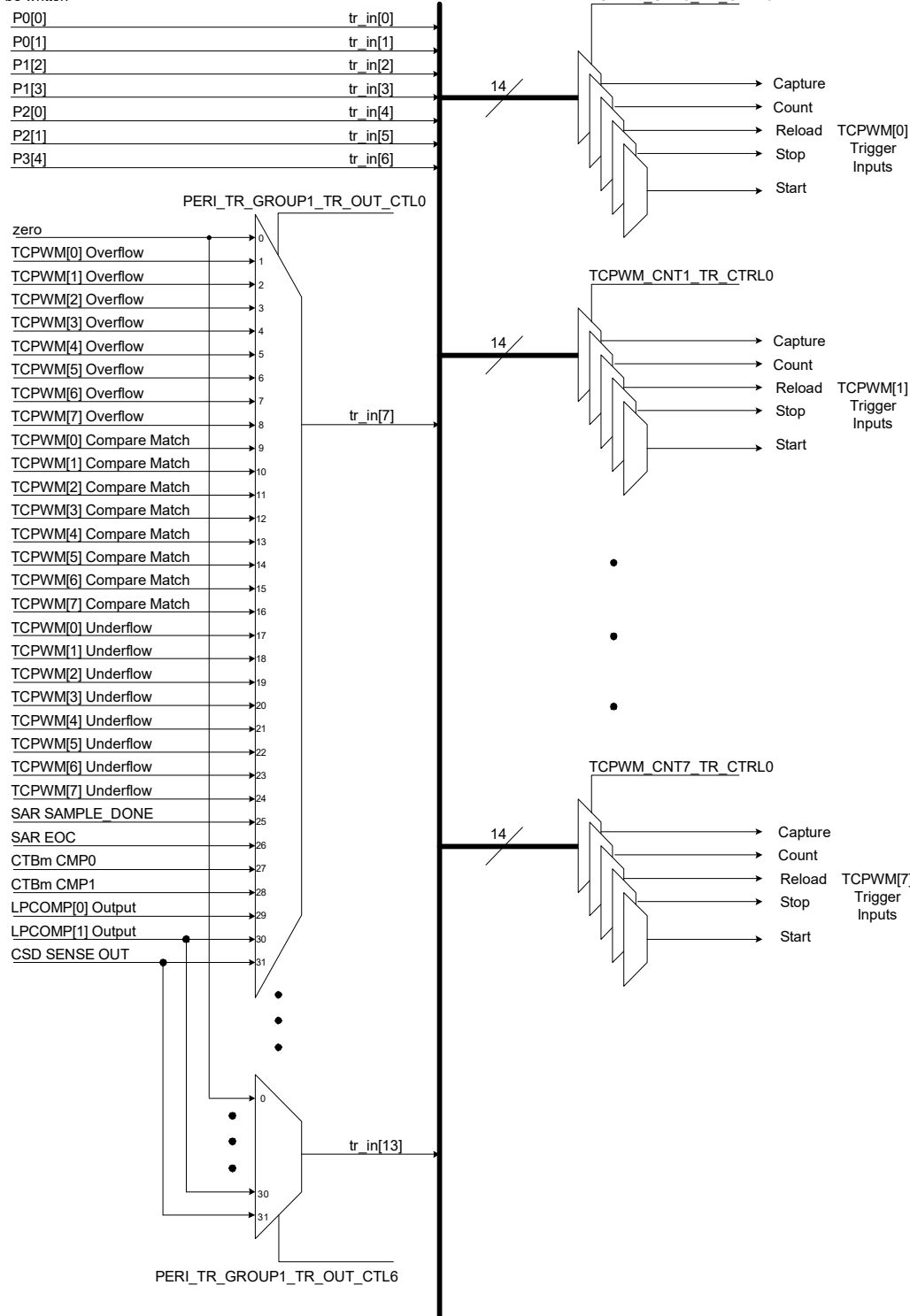


Figure 18-4. PSoC 4100S Plus での Mux のトリガー

To use GPIOs for trigger, HSIOM_PORT_SELx register should be written



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を初期化し、開始します。
 - カウントアップ モードおよびカウントダウン モードでは、カウンタ レジスタ (TCPWM_CNT_COUNTER) が「0」で初期化されます。
 - カウント アップ / ダウン モードでは、カウンタ 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可能
- **開始:** 開始イベントはカウンタ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 レジスタのある値への再初期化の後に使用できます。カウンタ 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、開始イベントは直交位相入力 phiB として機能します。これについては、“[直交デコーダ モード](#)” (205 ページ) で詳しく説明しています。
- **カウント:** カウンタ イベントではカウンタ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タ イベントは直交位相入力 phiA として使用できます。
- **停止:** 停止イベントはカウンタのインクリメント / デクリメントを停止します。開始イベントでカウンタは再び開始します。
 - 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 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。

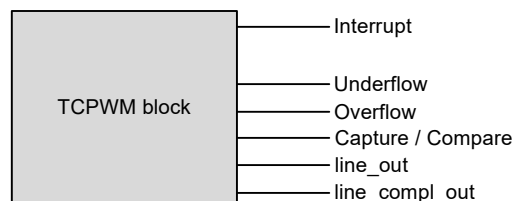
注意事項

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

18.2.4 出力信号

Figure 18-5 に示すように、TCPWM ブロックはいくつかの出力信号を生成します。

Figure 18-5. TCPWM 出力信号



18.2.4.1 トリガー条件発生時の信号

- カウントアップ モードでカウンタ レジスタが周期値に達するとカウンタ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 モードでカウンタ レジスタが 0 に達するとカウンタは内部アンダーフロー (UN) 条件を生成します。
- カウンタが実行中で、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:
 - カウンタが比較値と等しい
 - 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る

注：これらの信号が発生すると、HFCLK の 2 サイクルの間、ロジックハイのままです。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から専用割り込み出力信号を提供します。割り込みは TC 条件または CC 条件により生成できます。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。Table 18-2 に示すように、このブロックの割り込み処理には 4 つのレジスタが使用されます。

Table 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' に設定されます。このビットをクリアするには、'1' を書き込みます。
	1	CC_MATCH	カウント値がキャプチャ / 比較のレジスタ値と一致時、このビットは「1」にセット。このビットをクリアするには、'1' を書き込みます。
TCPWM_CNT_INTR_SET (割り込みセット要求レジスタ)	0	TC	'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設定します。読み出し時、このレジスタは割り込み要求レジスタの状態を示す
	1	CC_MATCH	'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設定します。読み出し時、このレジスタは割り込み要求レジスタの状態を示す
TCPWM_CNT_INTR_MASK (割り込みマスク レジスタ)	0	TC	割り込み要求レジスタの対応する TC ビットのマスク ビット
	1	CC_MATCH	割り込み要求レジスタの対応する CC_MATCH ビットのマスク ビット
TCPWM_CNT_INTR_MASKED (割り込みマスク要求レジスタ)	0	TC	対応する TC 要求とマスク ビットの論理積
	1	CC_MATCH	対応する CC_MATCH 要求とマスク ビットの論理積

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。TCPOV_CNT_TR_CTRL2 レジスタを構成することにより、OV、UN、および CC 条件を使用して、必要に応じて line_out および line_compl_out を駆動できます (Table 18-3 を参照)。

Table 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 Default Value = 3	1:0	0	line_out を '1' に設定	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 Default Value = 3	3:2	0	line_out を '1' に設定	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 Default Value	5:4	0	line_out を '1' に設定	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。詳細については、Table 18-4 を参照してください。

Table 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
Active (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
Sleep (スリープ)	すべてのカウンタ 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
Deep-Sleep (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

18.3 動作モード

カウンタブロックは、Table 18-5 に示すように、6 つの動作モードで機能できます。カウンタ制御レジスタ (TCPWM_CNTx_CTRL) の MODE[26:24] フィールドはカウンタを特定の動作モードに設定します。

Table 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
Timer (タイマ)	000	タイマまたはカウンタを搭載。カウンタは、カウントイベントが検出されたすべてのカウンタクロックサイクルで '1' ずつ増加または減少します。
Capture (取り込み)	010	キャプチャ入力を備えるタイマまたはカウンタを搭載。カウンタは、カウントイベントが検出されたすべてのカウンタクロックサイクルで '1' ずつ増加または減少します。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
Quadrature Decoder (直交デコーダ)	011	選択した (X1、X2 または X4) 符号化スキームに応じて 2 相入力に基づいてカウンタがデクリメント / インクリメントする直交デコーダを実装
PWM	100	8 ビット クロック プリスケール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 を搭載
PWM-PR	110	16 ビット線形帰還シフトレジスタ (LFSR) を用いる疑似乱数 PWM を搭載

Table 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により、カウンタをカウントアップ、カウントダウン、およびカウントアップするように構成できます。

Table 18-6. カウント モード コンフィギュレーション

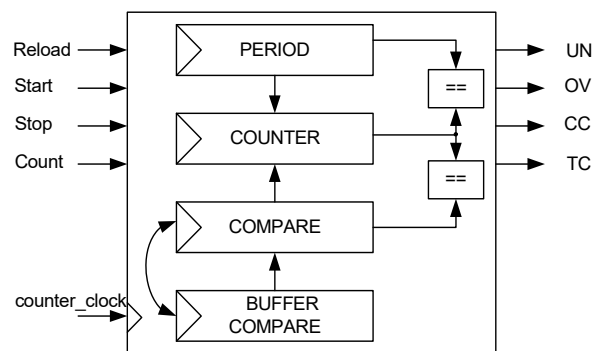
カウント モード	UP_DOWN_MODE[17:16]	説明
UP Counting Mode (アップ カウント モード)	00	周期値に達するまでカウンタをインクリメント。カウンタが周期値に達すると、ターミナル カウント (TC) 条件が生成される
DOWN Counting Mode (ダ ウンカウント モード)	01	「0」に達するまで周期値からカウンタをデクリメント。カウンタが「0」に達すると TC 条件が生成される
UP/DOWN Counting Mode 0 (アップ / ダウンカウント モード 0)	10	周期値に達するまでカウンタをインクリメントしてから、「0」に達するまでカウンタをデクリメント。「0」に達する時にのみ TC 条件が生成される
UP/DOWN Counting Mode 1 (アップ / ダウンカウント 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、カウンタが「0」に達する時にもカウンタが周期値に達する時にも TC 条件が生成される

18.3.1 タイマ モード

一般的にタイマ 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

Figure 18-6. タイマ モード ブロック図



18.3.1.2 動作原理

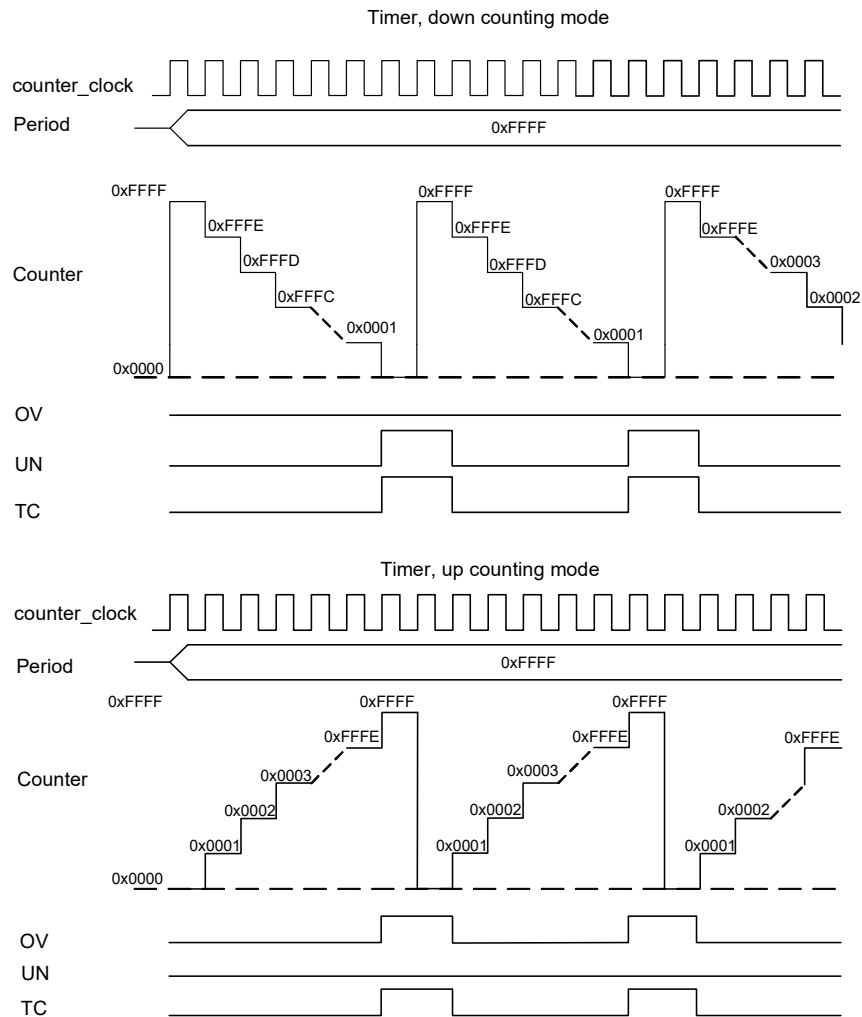
タイマはカウントアップ、ダウン、アップ / ダウン モードに設定できます。さらに連続モードまたはワンショット モードに設定することもできます。以下にタイマの動作原理を説明します：

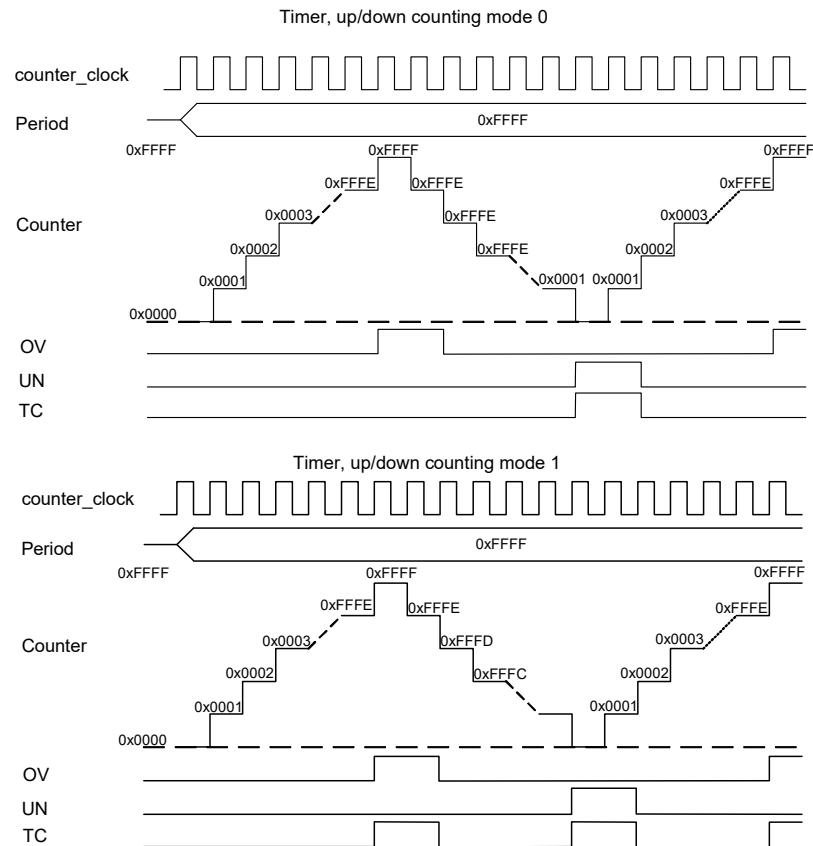
- タイマはアップ、ダウン、アップ / ダウンのカウンタです。
 - 現時点のカウント値はカウンタ レジスタ (TCPWM_CNTx_COUNTER) に格納されます。**注：**カウンタの実行中にこのレジスタに値を書き込むことはお勧めしません。
 - タイマの周期は周期レジスタに格納されます。
- カウンタは以下の異なるモードで再初期化します：
 - カウントアップ モードではカウンタ レジスタは周期に達すると自動的に 0 にリロードされます。
 - カウントダウン モードではカウンタ レジスタが 0 に達すると周期レジスタの値にリロードされます。
 - カウントアップ / ダウン モードではカウンタ 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タ値が 0 または周期値に達するとカウンタ方向が変化します
- カウンタ レジスタの値が比較レジスタの値と等しいと CC 条件を生成します。この条件の発生時に、カウンタ制御レジスタ (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。

Figure 18-7 は、4 つの異なるカウントモードでのカウンタのタイマ動作モードを示しています。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 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 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 サイクル(0からAまで、そして0に戻る)となります。

Figure 18-7. 各カウントモードにおけるタイマのタイミング図





注：OV および UN 信号は、“トリガー条件発生時の信号”(197 ページ)で説明するように、HFCLK の 2 サイクルの間、論理 High のままです。本章の図は HFCLK とカウンタ クロックが同じと想定しています。

18.3.1.3 タイマ モードの設定方法

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

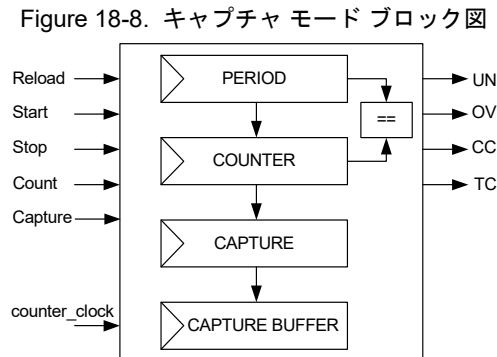
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」を書き込んで、カウンタを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに '000' を書き込んで、タイマモードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期値を設定します。
4. TCPWM_CNT_CC レジスタに 16 ビットの比較値を設定し、TCPWM_CNT_CC_BUFF レジスタにバッファの比較値を設定します。
5. CC 条件の発生で値を切り替える必要がある場合、TCPWM_CNT_CTRL レジスタの AUTO_RELOAD_CC フィールドをセットします。
6. Table 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むことにより、クロックプリスケールを設定します。
7. Table 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むことにより、カウントの方向を設定します。
8. TCPWM_CNT_CTRL の ONE_SHOT[18] フィールドに 0 か 1 を書き込んでタイマ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、“[割り込み](#)” (198 ページ) に示すように、TC または CC 条件で割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「1」を書き込んで、カウンタを有効にします。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。

18.3.2 キャプチャ モード

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

18.3.2.1 ブロック図



18.3.2.2 動作方法

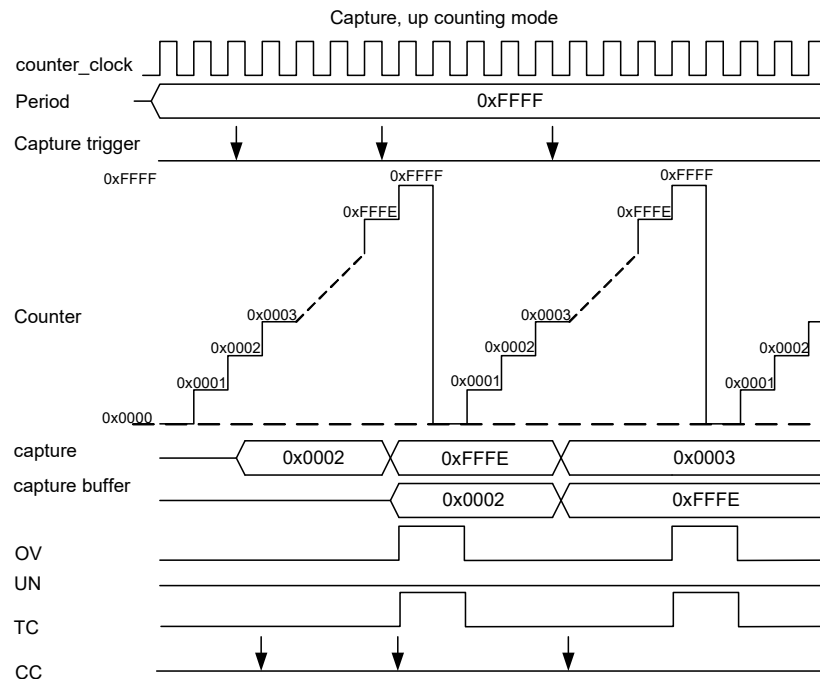
カウンタ制御レジスタ (TCPWM_CNT_CTRL) の UP_DOWN_MODE[17:16] ビット フィールドを設定することでカウンタをアップ、ダウン、アップ/ダウンのモードに設定することができます。

キャプチャ モードにおける動作を以下に示します。

- 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。

Figure 18-9 は、アップカウントモードでのキャプチャ動作を示しています。

Figure 18-9. キャプチャ モード - カウントアップ モードのカウンタのタイミング図



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。
- カウンタが周期値に達すると、内部オーバーフロー (OV) と TC 条件が生成されます。
- キャプチャ イベント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：イベントは観察されません。
 - 奇数のキャプチャ イベントがある場合：1 つのイベントが観察されます。

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

18.3.2.3 キャプチャ モードの設定方法

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

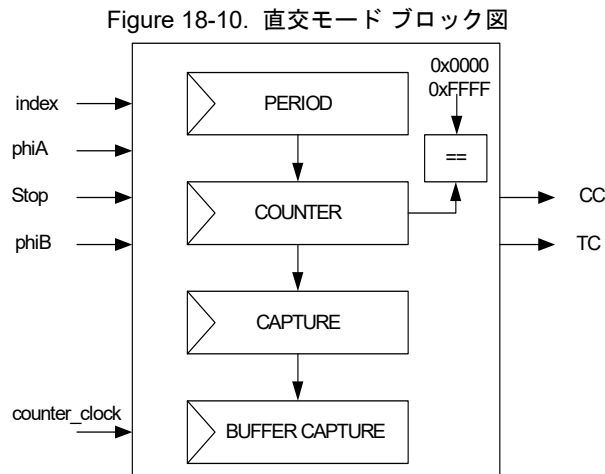
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「0」を書き込んで、カウンタを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに「010」を書き込んで、キャプチャモードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期値を設定します。
4. Table 18-1 に示すように、TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込むことにより、クロックプリスケールを設定します。
5. Table 18-6 に示すように、TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE[17:16] フィールドに書き込むことにより、カウントの方向を設定します。
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、およびカウント) の原因となるエッジを選択します。

9. 必要に応じて、“[割り込み](#)” (198 ページ) に示すように、TC または CC 条件で割り込みを設定します。
10. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに「1」を書き込んで、カウンタを有効にします。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。

18.3.3 直交デコーダ モード

直交デコーダは回転機器 (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。直交デコーダ信号はデコーダの phiA と phiB 入力を使用します。

18.3.3.1 ブロック図



18.3.3.2 動作原理

直交デコーダは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。

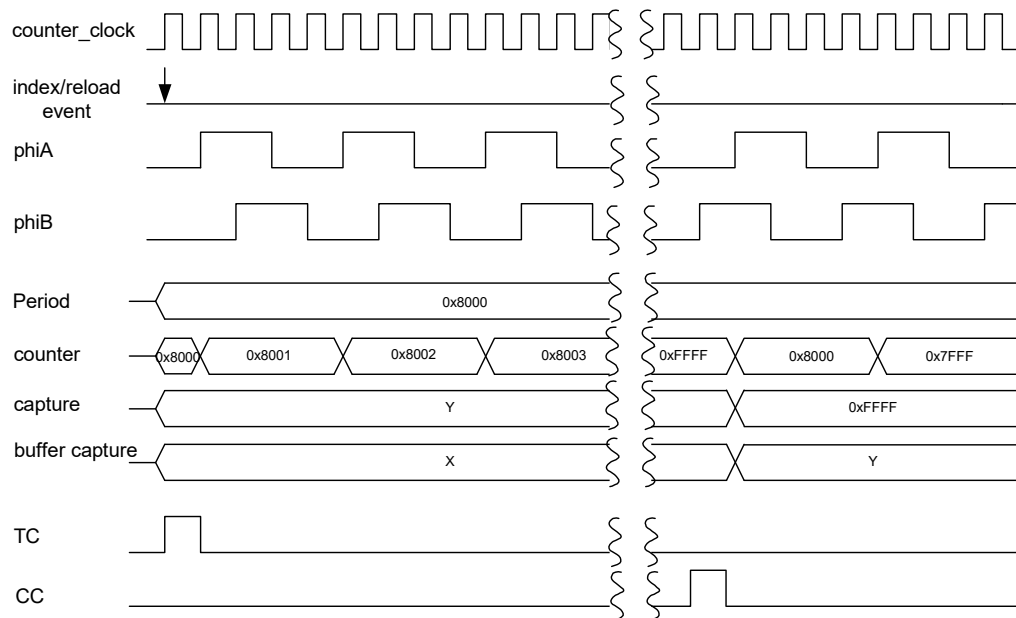
直交モードにおける動作を以下に示します。

- 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のハードウェア入力のカウンタ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、[Figure 18-11](#)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は 0x0000 (カウントアップモードの場合) または周期値 (カウントダウン モード) に設定されます。
注: ダウンカウントモードは、期間値 0x8000 (中間値) で使用することをお勧めします。
- カウント レジスタが 0x0000 または 0xFFFF に達すると CC 出力信号にパルスが生成されます。CC 条件の発生時、カウント レジスタは周期値 (この場合 0x8000) に設定します。
- 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 レジスタにコピーします。
 - キャプチャ レジスタの値はバッファ キャプチャ レジスタにコピーします。
 - この条件は割り込み要求を生成するために使用できません。
- 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を判定します :
 - カウンタ アンダーフローが発生したか (値 0)
 - カウンタ 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 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。値 '0' は以前のインクリメント操作を示し、値 '1' は以前のデクリメント操作を示します。[Figure 18-11](#) は、X1 エンコードモードでの直交動作を示しています。
 - phiA の正のエッジは、phiB が '0' の場合にカウンタをインクリメントし、phiB が '1' の場合にカウンタをデクリメントします。
 - カウント レジスタはインデックス / リロード イベントの発生時に周期値に初期化されます。

- カウンタがインデックス イベントで初期化されるとターミナル カウントを生成します。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、カウントレジスタ値がキャプチャレジスタにコピーされ、カウントレジスタは周期値 (この場合は 0x8000) で初期化されます。

Figure 18-11. 直交モード-X1 符号化のタイミング図

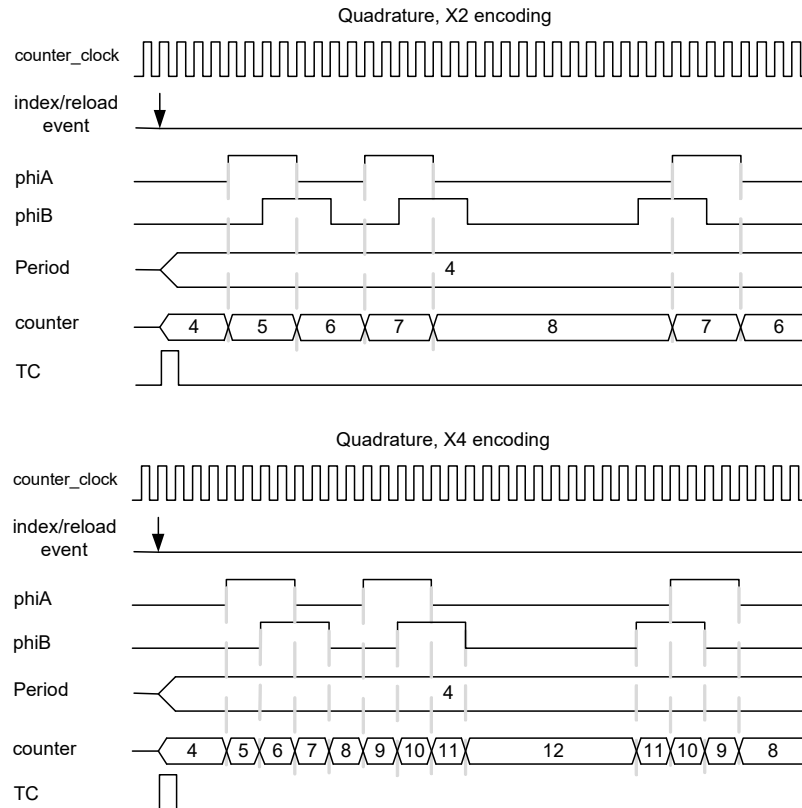
Quadrature, X1 encoding



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。

Figure 18-12 は、X2 および X4 エンコーディングモードでの直交モードの動作を示しています。

Figure 18-12. 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

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

1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '0' を書き込んで、カウンタを無効にします。
2. '011' を TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに書き込んで、直交モードを選択します。
3. TCPWM_CNT_PERIOD レジスタに必要な 16 ビット周期値を設定します。
4. TCPWM_CNT_CTRL レジスタの QUADRATURE_MODE[21:20] フィールドに書き込んで必要な符号化モードを設定します。
5. TCPWM_CNT_TR_CTRL0 レジスタを設定してイベント (インデックス、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、“[割り込み](#)” (198 ページ) に示すように、TC または CC 条件で割り込みを設定します。
8. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '1' を書き込んで、カウンタを有効にします。

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 = 周期値 / カウンタ クロック周波数

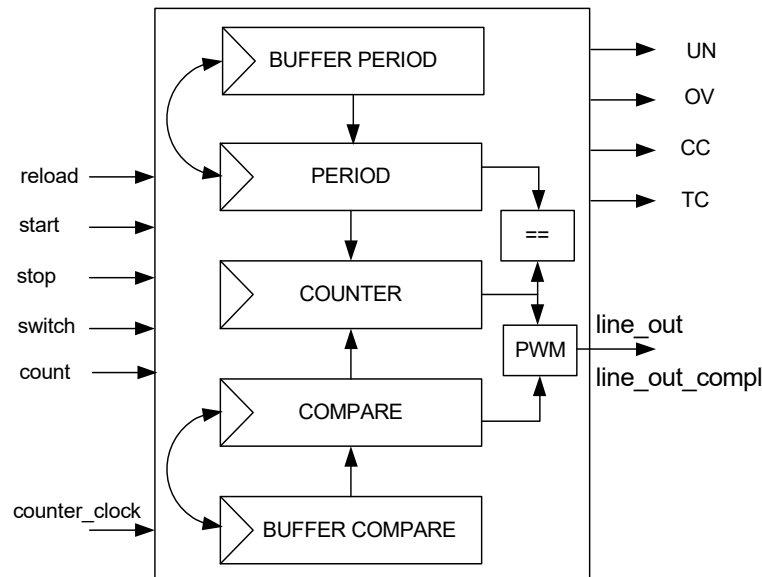
中央揃えモードの場合：PWM 周期 = $2 \times (\text{周期値} / \text{カウンタ クロック周波数})$

左および右揃えモードの場合：デューティ比 = 比較値 / 周期値

中央揃えモードの場合：デューティ比 = $(\text{周期} - \text{比較値}) / \text{周期}$

18.3.4.1 ブロック図

Figure 18-13. 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。Table 18-6 に示すように、TCPWM_CNT_CTRL レジスタの UP_DOWN_MODE [17:16] ビットを使用して選択した counter's のアップ、ダウン、およびアップ/ダウンカウントモードを使用することにより、目的の出力アライメントを実現できます。

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。信号は、TCPWM_CNT_TR_CTRL2 レジスタを構成することにより、出力ラインを切り替えたり、論理 '0' または '1' に設定したりできます。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を以下に示します。

- 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します。カウンタ制御レジスタの

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。

- バッファ期間レジスタとバッファ比較レジスタへの更新は、アクティブな切り替えイベントを持つ次の TC の前に完了する必要があります。それ以外の場合、Figure 18-15 に示すように、切り替えはレジスタの更新を反映しません。

中央揃えモードでは、出力ラインはターミナルカウントで '0' に設定され、CC 条件で切り替えられます

リロード イベントの発生時カウンタ 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。カウントのたびにカウンタ レジスタは比較レジスタ値と比較し、一致すると CC 信号を生成します。

Figure 18-14 は、バッファされた周期と比較レジスタ (アップ/ダウンカウントモード 0) を備えた中央揃え PWM を示しています。

Figure 18-14. センターアライン PWM のタイミング図

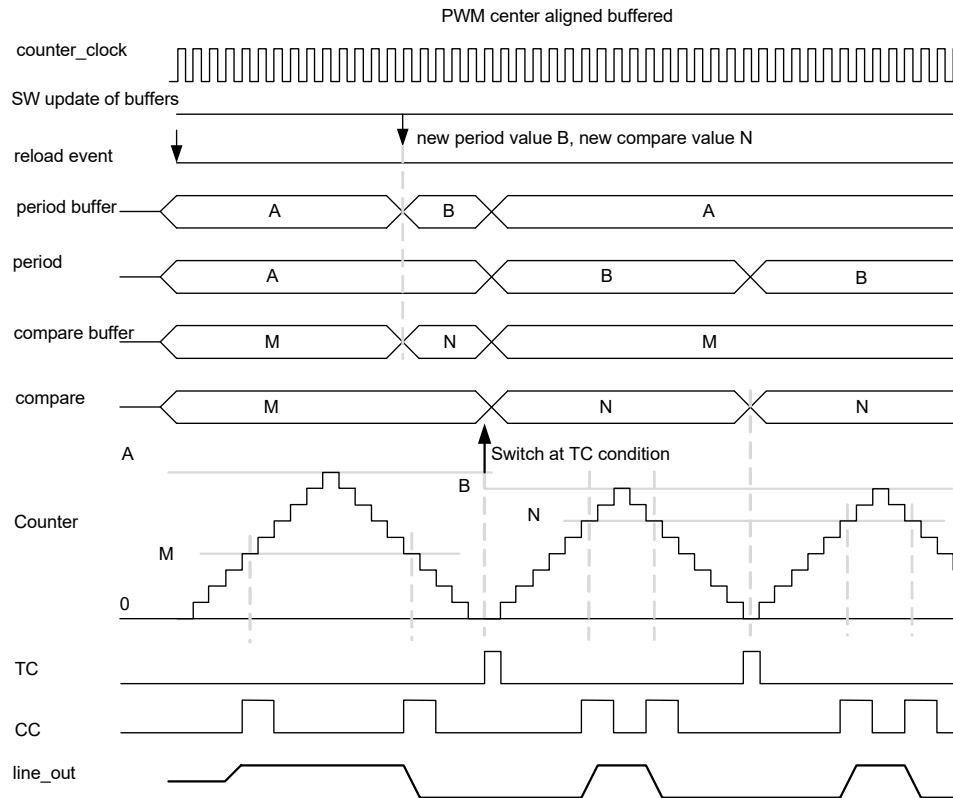
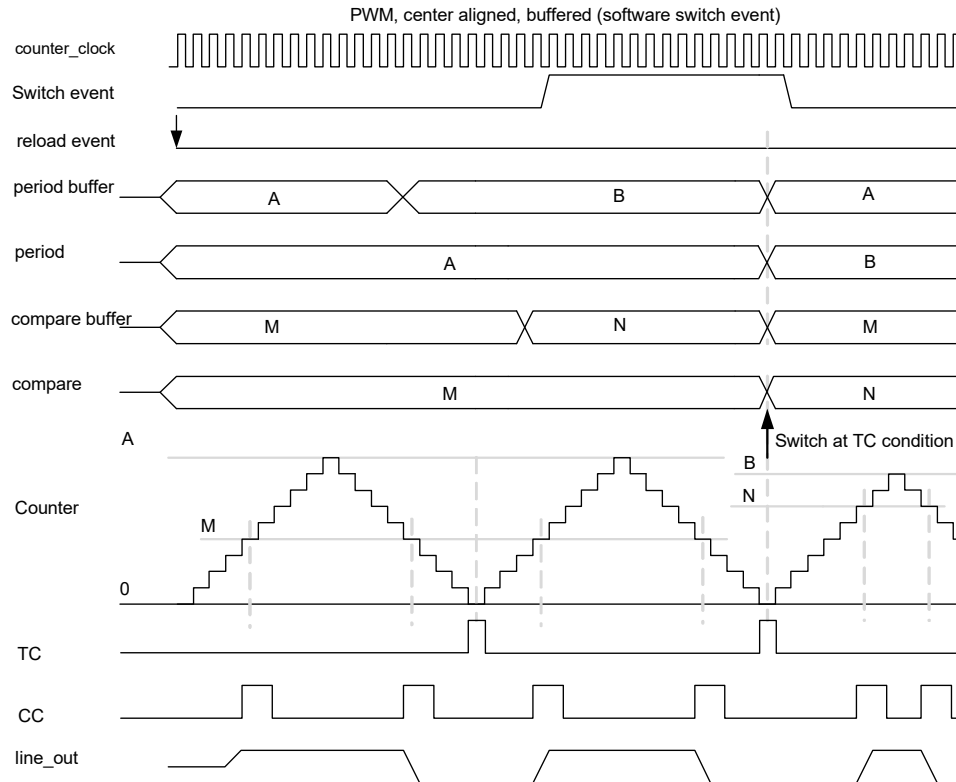


Figure 18-15 は、ソフトウェアで生成されたスイッチイベントを使用した中央揃え PWM を示しています。

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

Figure 18-15. 中央揃え PWM のタイミング図 (ソフトウェアスイッチイベント)



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が「0」または周期値に達すると TC が発生します。非対称的 PWM を作成するために比較レジスタを TC 条件ごとに (カウンタが「0」または周期値に達する時) 変更し、周期レジスタを 1 つの TC 条件おきに (カウンタが「0」に達する時のみ) 変更します。
- 左揃え PWM の場合は、アップカウントモードを使用します。出力線を '1' に設定するように OV 条件を構成し、出力線を '0' にリセットするように CC 条件を構成します。Table 18-3 を参照してください。
- 右揃え PWM の場合は、ダウンカウントモードを使用します。UN 条件を設定して出力行を '0' にリセットし、CC 条件を設定して出力行を '1' に設定します。Table 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。Table 18-7 に示すように、このイベントは、カウンタ制御レジスタの PWM_STOP_ON_KILL および PWM_SYNC_KILL フィールドを変更することにより、カウンタを停止するようにプログラムできます。

Table 18-7. ストップ オン キル機能のフィールド設定

PWM_STOP_ON_KILL フィールド	備考
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、カウンタは停止される

キルイベントは、Table 18-8 に示すように、非同期または同期になるようにプログラムできます。

Table 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	備考
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。kill 入力削除された直後に PWM を再起動するには、kill イベントを非同期にする必要があります (Table 18-8 を参照)。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

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

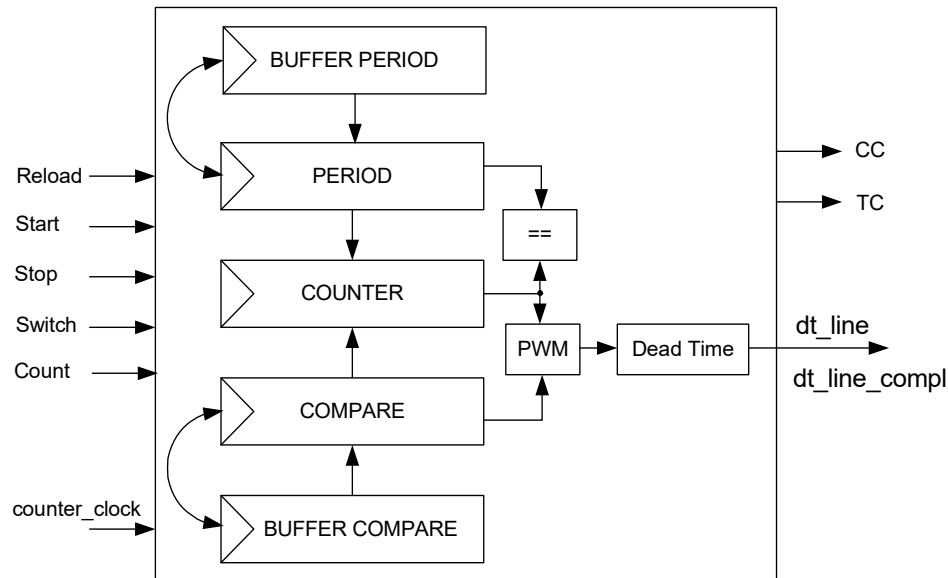
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '0' を書き込んで、カウンタを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに '100' を書き込んで、PWM モードを選択します。
3. Table 18-1 に示すように、TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込むことにより、クロックプリスケールを設定します。
4. TCPWM_CNT_PERIOD レジスタに 16 ビット周期を設定し、必要に応じて TCPWM_CNT_PERIOD_BUFF レジスタに切り替え用の周期値を設定します。
5. TCPWM_CNT_CC レジスタに 16 ビット比較値を設定し、必要に応じて TCPWM_CNT_CC_BUFF レジスタに切り替え用の比較値を設定します。
6. Table 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、左揃え、右揃え、または中央揃えの PWM を構成して、カウントの方向を設定します。
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 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、“[割り込み](#)” (198 ページ) に示すように、TC または CC 条件で割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '1' を書き込んで、カウンタを有効にします。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します。2 信号の遷移に指定の時間間隔を挿入します。2 つの補完的な出力ライン 'dt_line' および 'dt_line_compl' は、これらの 2 つのラインから派生しています。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

Figure 18-16. 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 の動作を以下に示します。

- UN、OV、および CC の条件に応じて、PWM line_out の立ち上がりエッジで、デッドタイムブロックは dt_line および dt_line_compl を「0」に設定します。
- 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。
- 不感帯期間が完了すると、dt_line は「1」に設定されます。
- UN、OV、および CC 条件に応じた PWM line_out の立ち下がりエッジで、デッドタイムブロックは dt_line および dt_line_compl を「0」に設定します。
- 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。
- 不感帯期間が完了すると、dt_line_compl が「1」に設定されます。
- 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。

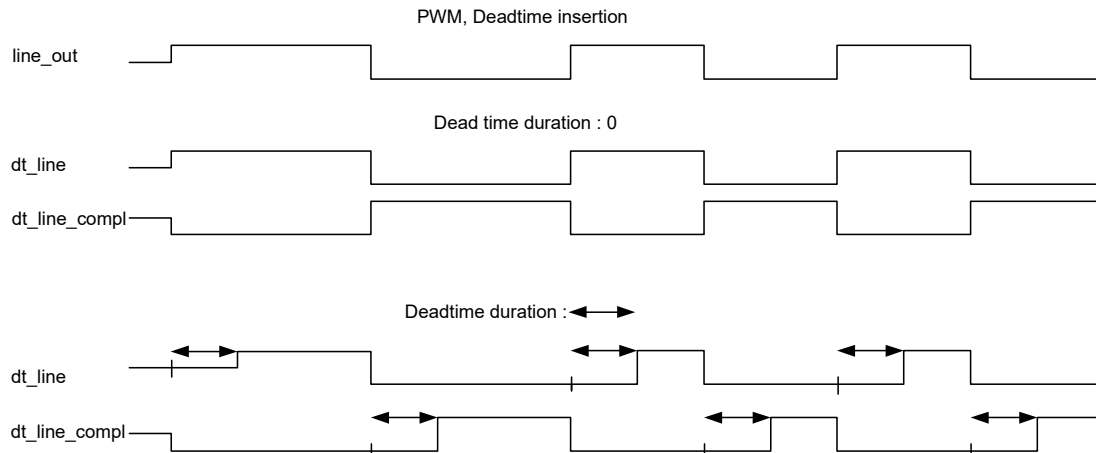
このモードは PWM モードに従い、以下の機能をサポートします：

- さまざまな出力アライメントモード
- PWM 「line_out」 および 「line_out_compl」 からそれぞれ派生した 2 つの相補出力ライン、dt_line および dt_line_compl
 - 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。

Figure 18-17は、相補出力ラインdt_lineおよびdt_line_complがPWM出力ラインline_outから生成される方法を示しています。

Figure 18-17. 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

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

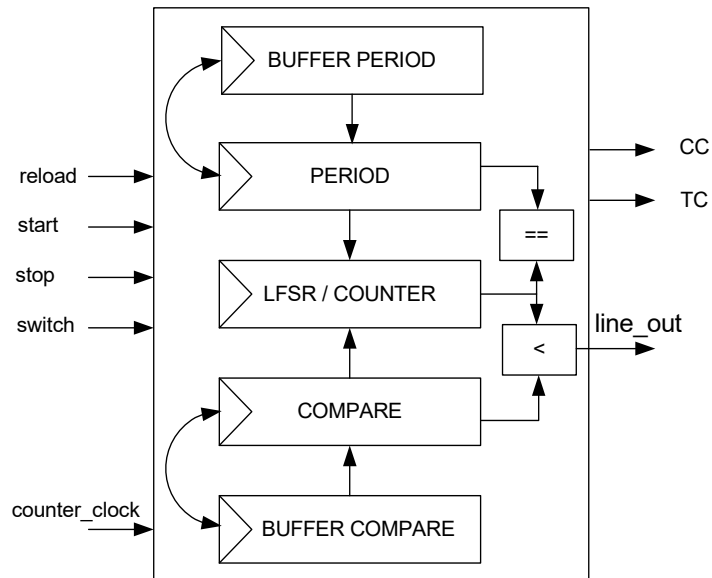
1. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '0' を書き込んで、カウンタを無効にします。
2. TCPWM_CNT_CTRL レジスタの MODE[26:24] フィールドに '101' を書き込んで、デッドタイムモードの PWM を選択します。
3. Table 18-1 に示すように、TCPWM_CNT_CTRL レジスタの GENERIC[15:8] フィールドに書き込むことにより、必要なデッドタイムを設定します。
4. TCPWM_CNT_PERIOD レジスタに 16 ビット周期を設定し、必要に応じて TCPWM_CNT_PERIOD_BUFF レジスタに切り替え用の周期値を設定します。
5. 必要に応じて、TCPWM_CNT_CC レジスタに 16 ビットの比較値を設定し、TCPWM_CNT_CC_BUFF レジスタにバッファの比較値を設定して値を切り替えます。
6. Table 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、左揃え、右揃え、または中央揃えの PWM を構成して、カウントの方向を設定します。
7. “パルス幅変調モード” (209 ページ) に示すように、TCPWM_CNT_CTRL レジスタの PWM_STOP_ON_KILL および PWM_SYNC_KILL フィールドを必要に応じて設定します。
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 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、“割り込み” (198 ページ) に示すように、TC または CC 条件で割り込みを設定します。
12. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '1' を書き込んで、カウンタを有効にします。ハードウェア スタート信号が有効になっていない場合カウンタを開始するためにファームウェア (TCPWM_CMD レジスタ) でスタートトリガーを与える必要があります。

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

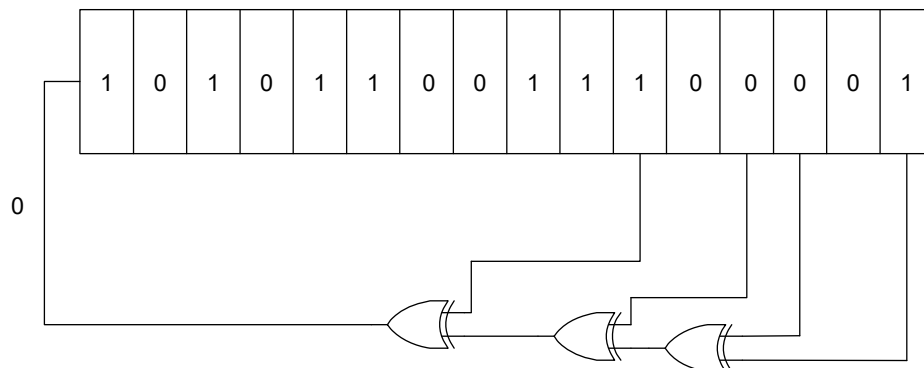
Figure 18-18. PWM-PR モード ブロック図



18.3.6.2 動作原理

Figure 18-19 に示すように、カウンタレジスタは多項式 $x^{16} + x^{14} + x^{13} + x^{11} + 1$ で LFSR を実装するために使用されます。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 レジスタを 0 以外の値に初期化する必要があります。ことに注意してください。

Figure 18-19. カウンタ レジスタを用いた疑似乱数系列の生成



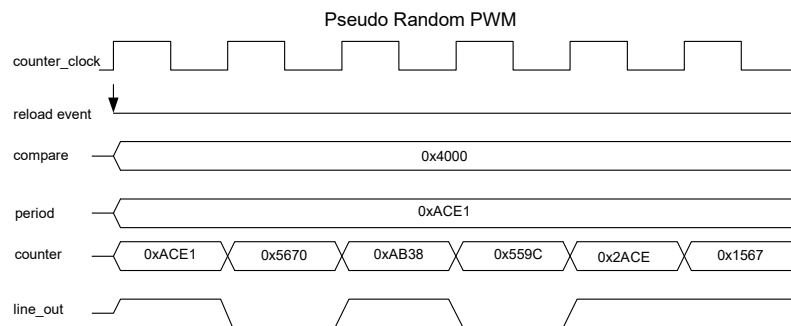
このプロセスの手順を以下に示します。

- PWM 出力ライン line_out は、カウンタレジスタの下位 15 ビット値がコンペアレジスタの値よりも小さい場合、'1' で駆動されます (counter[14:0]<compare[15:0])。「0x8000」以上の比較値は、常に PWM 出力ラインに '1' をもたらしめます。比較値「0」は、常に PWM 出力ラインに '0' をもたらしめます。'
- 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、カウンタを初期化しません。
- カウンタが周期値に等しくなるとターミナル カウントが生成されます。特定の初期値の場合 LFSR は予測可能なカウンタ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」後にカウンタを計算できます。計算したカウンタは周期値として使用され、TC は「n」の反復回数後に生成されます。
- TC では、切り替え / キャプチャイベントが条件付きで比較および期間レジスタのペアを切り替えます (カウンタ

制御レジスタの AUTO_RELOAD_CC および AUTO_RELOAD_PERIOD フィールドに基づく)。

- 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を停止することができます。
- カウンタ制御レジスタの 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はハードウェアにより停止されます。
- このモードではアンダーフロー、オーバーフロー、トリガー条件イベントは発生しません。
- CC 条件は、カウンタが実行中で、その値が比較値と等しい場合に発生します。Figure 18-20 は、疑似ランダムノイズの動作を示しています。
- 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)。

Figure 18-20. 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 / スイッチ入力信号は、エッジ (立ち上がり、立ち下がり、またはその両方) によって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

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

1. TCPWM_CTRL レジスタの COUNTER_ENABLED に '0' を書き込んで、カウンタを無効にします。
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、疑似ランダム 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 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、“[割り込み](#)”(198 ページ) に示すように、TC または CC 条件で割り込みを設定します。
10. TCPWM_CTRL レジスタの COUNTER_ENABLED フィールドに '1' を書き込んで、カウンタを有効にします。

18.4 TCPWM レジスタ

Table 18-9. TCPWM レジスタの一覧

レジスタ	備考	特長
TCPWM_CTRL	TCPWM 制御レジスタ	カウンタ 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 カウンタ割り込み原因レジスタ	集約された割り込み信号のソースを判定
TCPWM_CNT_CTRL	カウンタ制御レジスタ	カウンタ モード、符号化モード、ワンショット モード、スイッチング、キル機能、デッドタイム、クロック分周、カウント方向を設定
TCPWM_CNT_STATUS	カウンタ ステータス レジスタ	カウントの方向、デッドタイム期間、およびクロックの事前スケーリングを読み取ります。カウンタが実行されているかどうかを確認します
TCPWM_CNT_COUNTER	カウント レジスタ	16 ビット カウント
TCPWM_CNT_CC	カウンタ比較 / キャプチャ レジスタ	カウントを取り込むまたはカウンタ値と比較
TCPWM_CNT_CC_BUFF	カウンタ バッファ比較 / キャプチャ レジスタ	カウンタ CC レジスタのバッファ レジスタであり、周期を切り替える
TCPWM_CNT_PERIOD	カウンタ周期レジスタ	カウンタの上限値を格納
TCPWM_CNT_PERIOD_BUFF	カウンタ バッファ周期レジスタ	カウンタ周期レジスタのバッファ レジスタであり、比較値を切り替える
TCPWM_CNT_TR_CTRL0	カウンタ トリガー制御レジスタ 0	特定のカウンタ イベントのトリガーを選択
TCPWM_CNT_TR_CTRL1	カウンタ トリガー制御レジスタ 1	特定のカウンタ入力信号のエッジ検出
TCPWM_CNT_TR_CTRL2	カウンタ 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出力ラインを制御
TCPWM_CNT_INTR	割り込み要求レジスタ	TC または CC 条件が検出されるとレジスタ ビットをセット
TCPWM_CNT_INTR_SET	割り込み要求セット レジスタ	割り込み要求レジスタの対応するビットをセット
TCPWM_CNT_INTR_MASK	割り込みマスク レジスタ	割り込み要求レジスタのマスク
TCPWM_CNT_INTR_MASKED	マスクされた割り込み要求レジスタ	割り込み要求とマスク レジスタのビット論理積

19. 真の乱数生成器



PSoC 4100S Plus デバイスの Arm Cortex-M0 + CPU は、真の乱数を生成する真の乱数ジェネレータ (TRNG) コンポーネントをサポートします。生成された数値のビットサイズは、[0、32] の範囲でプログラム可能です。

19.1 特長

TRNG は、最大 6 つのリング発振器を使用して、物理的なノイズソースを提供します。リング発振器は、フィードバックループで接続された一連のインバーターで構成され、リングを形成します。インバーター遅延の温度感度のため、ジッタは ring's 発振信号に導入されます。ジッタされた発振信号がサンプリングされて、「デジタル化アナログ信号」(DAS) が生成されます。これは、すべての複数のリング発振器に対して行われます。

エントロピーを増やし、DAS ビットのバイアスを減らすために、DAS ビットはさらに後処理されます。後処理には 2 つのステップがあります。

- エントロピーを増加させるオプションの削減ステップ (最大 6 つの DAS ビットと 1 つまたは複数の DAS ビット周期)。
- '0'/'1' バイアスを減らすオプションのフォンノイマン補正ステップ。

この修正ステップでは、前のステップで生成された削減ビットのペアを処理します。2 つの削減されたビット r0 および r1 (r0 は r1 の前に生成される) が与えられると、補正ステップは次のように定義されます。

- $\{r0, r1\} = \{0, 0\}$: ビットは生成されません
- $\{r0, r1\} = \{0, 1\}$: '0' ビットが生成されます (ビット r0)
- $\{r0, r1\} = \{1, 0\}$: '1' ビットが生成されます (ビット r0)
- $\{r0, r1\} = \{1, 1\}$: ビットは生成されません

言い換えると、修正ステップは、'0' から '1' または '1' から '0' への遷移でのみビットを生成します。ランダムな入力ビットシーケンスの場合、補正ステップでは、入力ビットシーケンスの周波数の約 4 分の 1 の出力ビットシーケンスが生成されます (入力削減ビットは重複しないペアで処理され、ペアエンコーディングの半分だけが出力ビット)。

後処理により、真のランダムビットサンプルと見なされるビットサンプルが生成されます。真のランダムビットサンプルがレジスタにシフトされ、最大 32 ビットのランダム値が提供されます。

高いスイッチングアクティビティの結果として、リング発振器はかなりの量の電力を消費します。したがって、TRNG 機能を無効にすると、リングが「切断」され、切り替えが防止されます。TRNG 機能が有効になっている場合、リング発振器は最初は予測可能な動作をします。ただし、時間の経過とともに、微小な環境 (温度) の変化により、この予測可能な動作からの偏差が増加します。

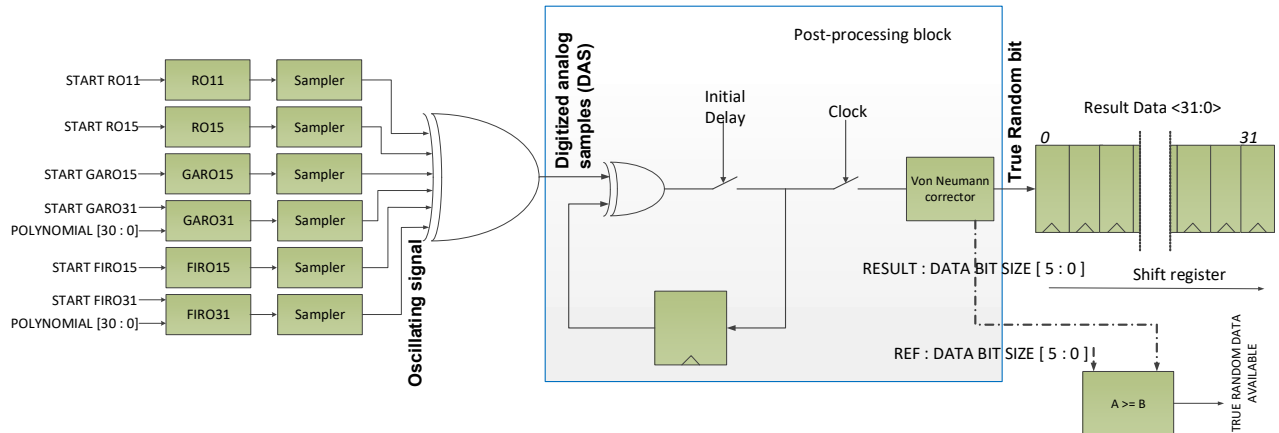
- 初期遅延の間、リング発振器は信頼できる物理的ノイズ源ではありません。
- 最初の遅延の後、同じリング発振器は異なる発振動作を示し、信頼できる物理的ノイズソースを提供します。

したがって、DAS ビットは初期化期間中に削除できます。

19.2 ブロック図

Figure 19-1 は、TRNG コンポーネントの概要を示しています。

Figure 19-1. TRNG の概要



リング発振器が停止すると、同期ロジックがリセットされることに注意してください。したがって、リング発振器は、後処理のリダクションステップに一定の '0' を提供します。

19.3 動作モード

TRNG は最大 6 つのリング発振器に依存しています。

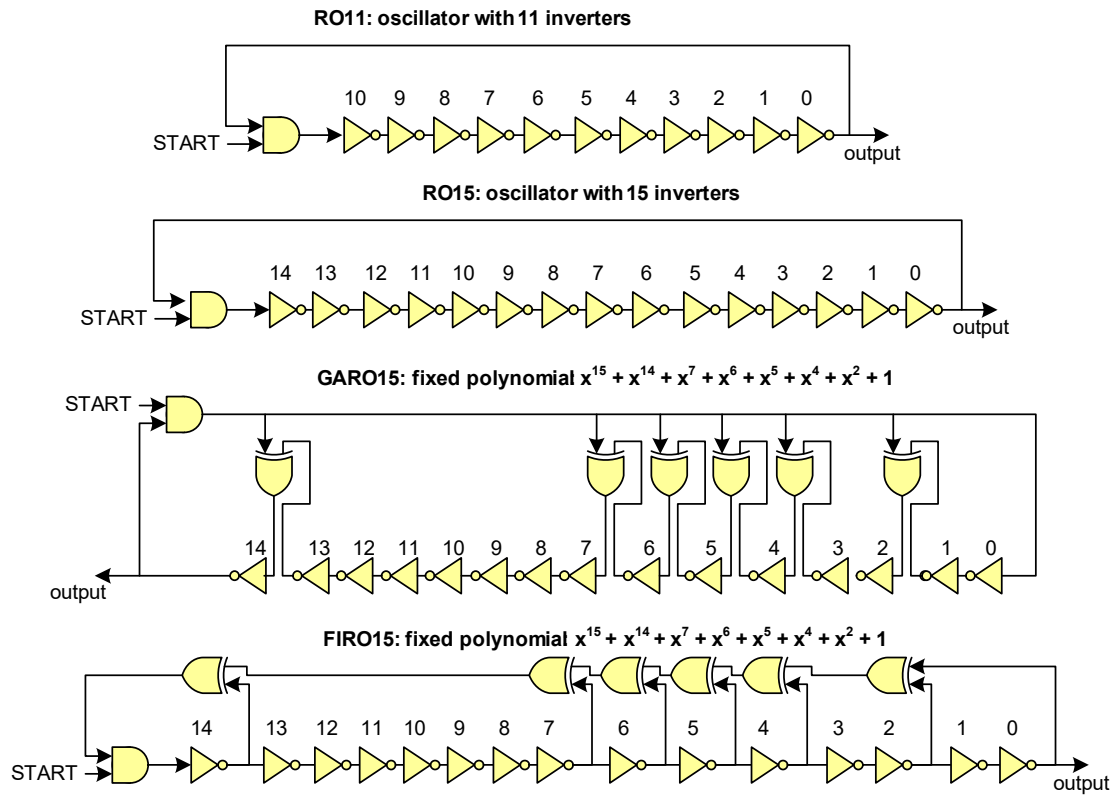
- RO11: 11 個のインバーターで構成される固定リング発振器。
- RO15: 15 個のインバーターで構成される固定リング発振器。
- GARO15: 15 インバーターの固定ガロアベースのリング発振器。
- GARO31: 最大 31 個のインバーターからなる柔軟なガロアベースのリング発振器。31 次までのプログラム可能な多項式は、発振器フィードバックに柔軟性を提供します。
- FIRO15: 15 インバーターの固定ガロアベースリング発振器。
- FIRO31: 最大 31 個のインバーターの柔軟なガロアベースのリング発振器。31 次までのプログラム可能な多項式は、発振器フィードバックに柔軟性を提供します。

各リング発振器は、開始または停止できます。停止すると、リングは「切断」され、切り替えが防止されます。

19.3.1 固定リング発振器

Figure 19-2 に、固定リング発振器の回路実装を示します。

Figure 19-2. 4 つの固定リング発振器 :RO11、RO15、GARO15、FIRO15

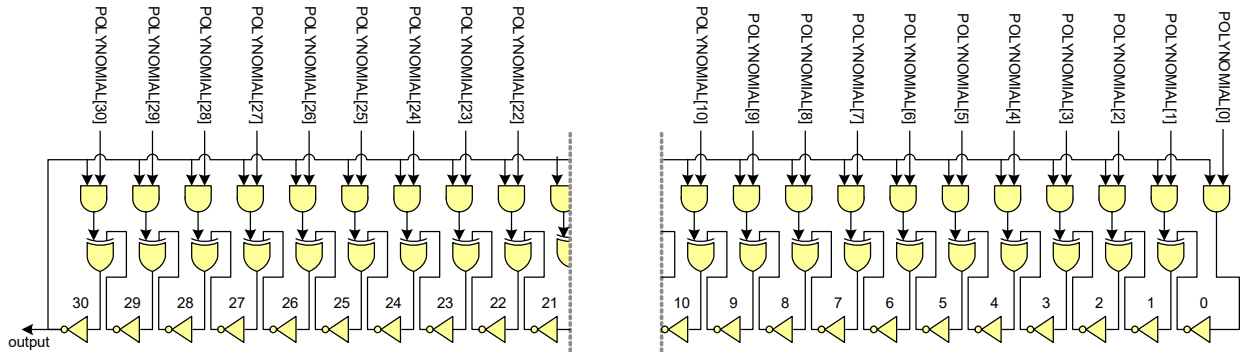


START 信号は MMIO レジスタフィールドから発信されます。

19.3.2 ガロアベースのリング発振器

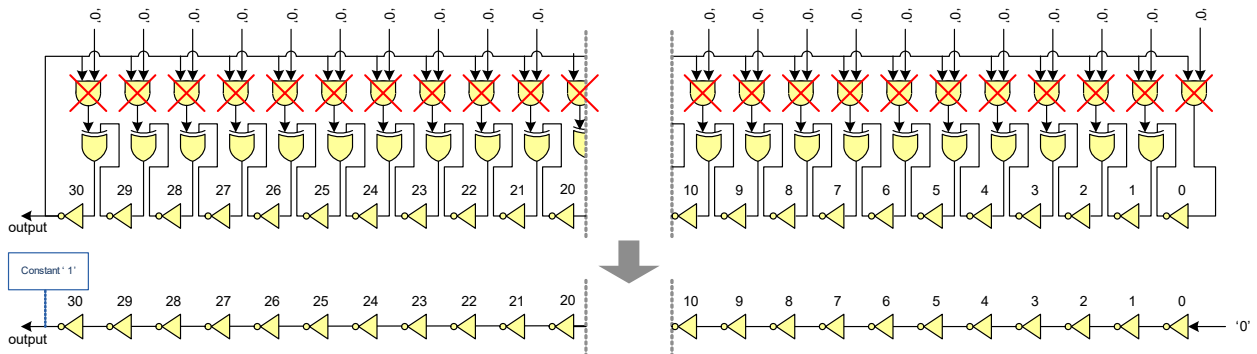
柔軟なガロアベースのリング発振器は、プログラム可能な多項式に依存して発振器フィードバックを指定します。これにより、1、3、5、...、31 インバーターのリングが可能です (発振信号を生成するには奇数が必要です)。Figure 19-3 に、ガロアベースのリング発振器の概要を示します。

Figure 19-3. 柔軟なガロアベースのリング発振器 :GAR031



リング発振器が停止すると、多項式は強制的に「0」になり、リングは Figure 19-4 に示すように切断されます。

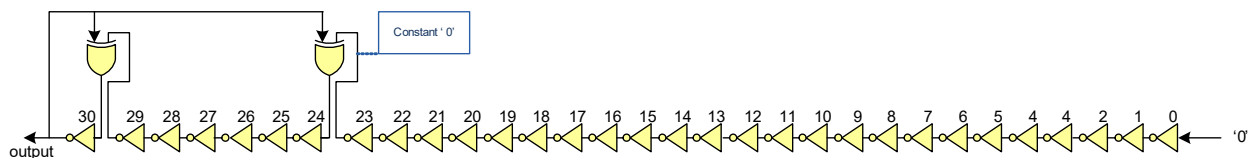
Figure 19-4. GAR031「停止」



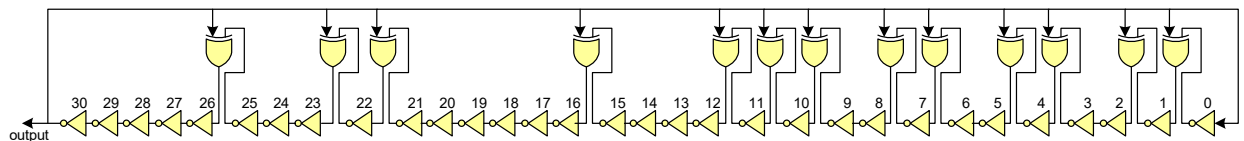
プログラム可能な多項式は、発振器のフィードバックを指定します。Figure 19-5 は、発振器フィードバックの 2 つの例を示しています。

Figure 19-5. GAR031 – 2 つの例

7-bit polynomial $x^7 + x^6 + 1$ (POLYNOMIAL = 0x0000:0041 << 24)



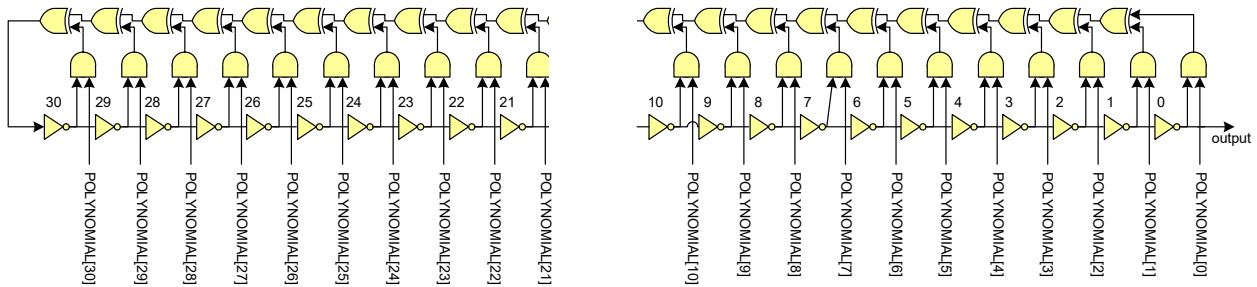
31 bit polynomial $x^{31} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$ (POLYNOMIAL = 0x04c1:1db7)



19.3.3 フィボナッチベースのリング発振器

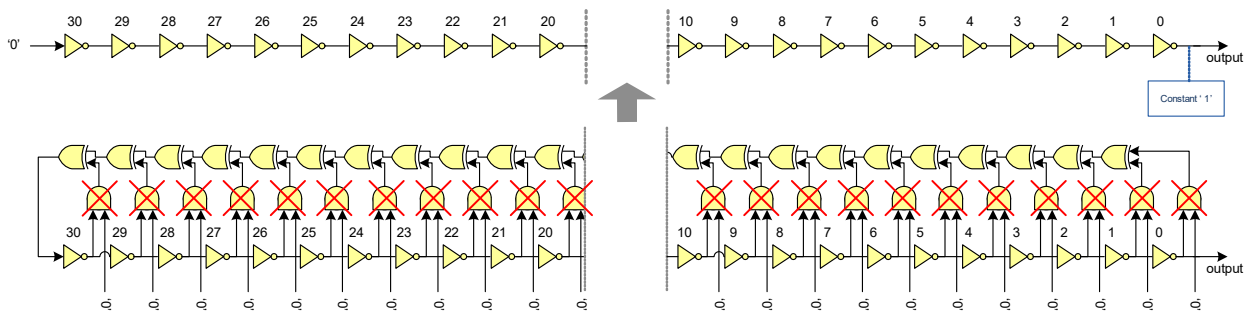
柔軟なフィボナッチベースのリング発振器も、プログラム可能な多項式に依存して発振器フィードバックを指定します。
Figure 19-6 に、フィボナッチベースのリング発振器の概要を示します。

Figure 19-6. 柔軟なフィボナッチベースのリング発振器 :FIRO31



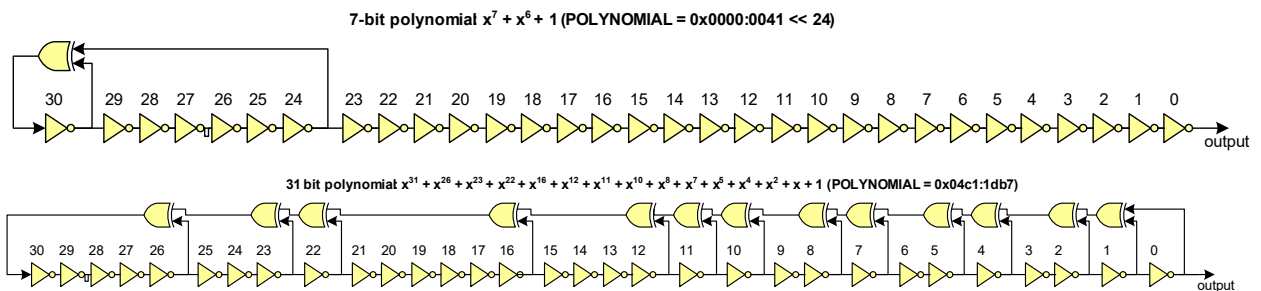
リング発振器が停止すると、多項式は強制的に「0」になり、リングは Figure 19-7 に示すように切断されます。

Figure 19-7. FIRO31「停止」。



プログラム可能な多項式は、発振器のフィードバックを指定します。Figure 19-8 に 2 つの例を示します。

Figure 19-8. FIRO31-2 つの例



20. LCD 直接駆動



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。

20.1 特長

PSoC 4 LCD セグメント駆動ブロックは以下の特長を持っています：

- PSoC 4100S で最大 28 セグメントと 8 コモン、PSoC 4100S Plus で 49 セグメントと 8 コモンをサポート
- タイプ A (標準) およびタイプ B (低消費電力) の駆動波形に対応
- コモンとセグメントを任意の GPIO ピンに配線可能
- 5 つの駆動方法に対応：
 - デジタル相関
 - 1/2 バイアス PWM
 - 1/3 バイアス PWM
 - 1/4 バイアスの PWM
 - 1/5 バイアスの PWM
- デジタル相関モードで 1.8V V_{DD} から 3V ディスプレイを駆動する機能
- 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

20.2 LCD セグメント駆動の概要

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 つをコモン電極 (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。電氣的に 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。識別比率が高いほど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。以下の用語はこれらの波形を定義するのに使用されます：

- **デューティ:'M'** 個の COM 電極を駆動するとき、ドライバは 1/M デューティで動作するとされています。各 COM 電極は実効的に 1/M の時間ドライブされます。
- **バイアス**: ドライブ波形が $(1/B) \times V_{DRV}$ 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されます。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 の PWM 駆動モードでは 1/2、1/3、1/4 および 1/5 バイアスに対応しています。

- **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時、すべてのセグメントで 0V DC (実効値は 0 ではない) になります。

PSoC 4 は全ての駆動モードで 2 種類の駆動波形をサポートします。内容：

- **タイプ A 波形**：このタイプの波形では、ドライバはフレームを M 個のサブフレームに構成します。'M' は COM 電極の数です。各 COM はフレームの間 1 回のみ駆動されます。例えば COM[i] は i サブフレームで駆動されます。
- **タイプ B 波形**：ドライバは 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相互で逆です。各 COM はフレームの間 2 回駆動されます。例えば COM[i] は i サブフレームと M+i フレームで駆動されます。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。

20.2.1 駆動モード

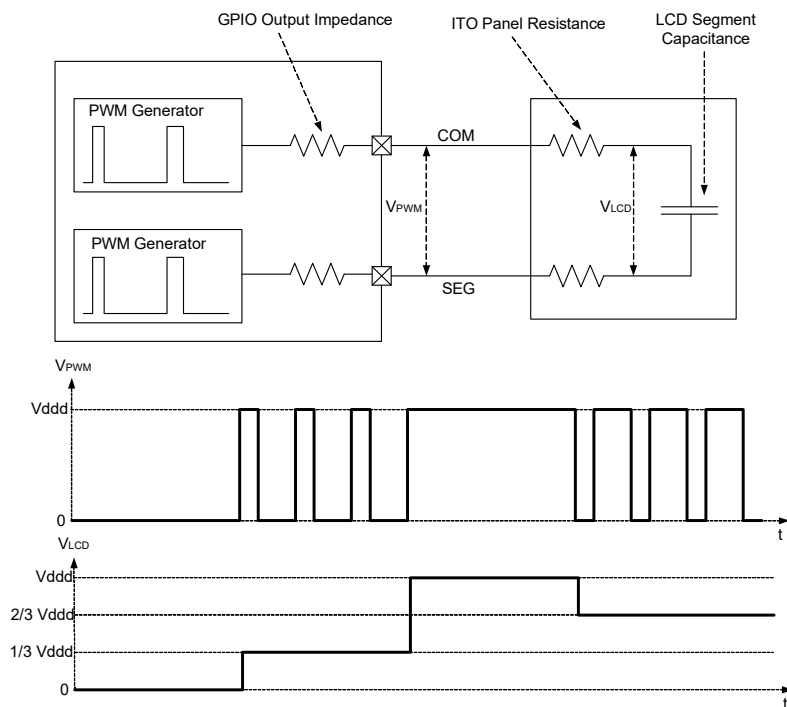
PSoC 4 は以下の駆動モードに対応します。

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

20.2.1.1 PWM 駆動

PWM 駆動モードでは、多電圧駆動信号は、LCD の固有抵抗と静電容量とともに PWM 出力信号を使用して生成されます。[Figure 20-1](#) はこれを示しています。

Figure 20-1. PWM 駆動 (1/3 バイアス)



駆動回路の出力波形は PWM 波形です。[Figure 20-1](#) に示すように、インジウムスズ酸化物 (ITO) パネルの抵抗と PWM をフィルタ処理するためのセグメント容量により、LCD セグメントの両端の電圧は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 (32 kHz、低速動作) または IMO (高速動作) から派生します。生成されたアナログ電圧は、通常、セグメント LCD 駆動用の非常に低い周波数 (約 50Hz) で動作します。

Figure 20-2 および Figure 20-3 は、1/2 バイアスおよび 1/4 デューティの COM および SEG 電極のタイプ A およびタイプ B の波形を示しています。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、Figure 20-4 と Figure 20-5 は、COM 電極と SEG 電極の 1/3 バイアスと 1/4 デューティのタイプ A とタイプ B の波形を示しています。

Figure 20-2. PWM1/2 タイプ A 波形の例

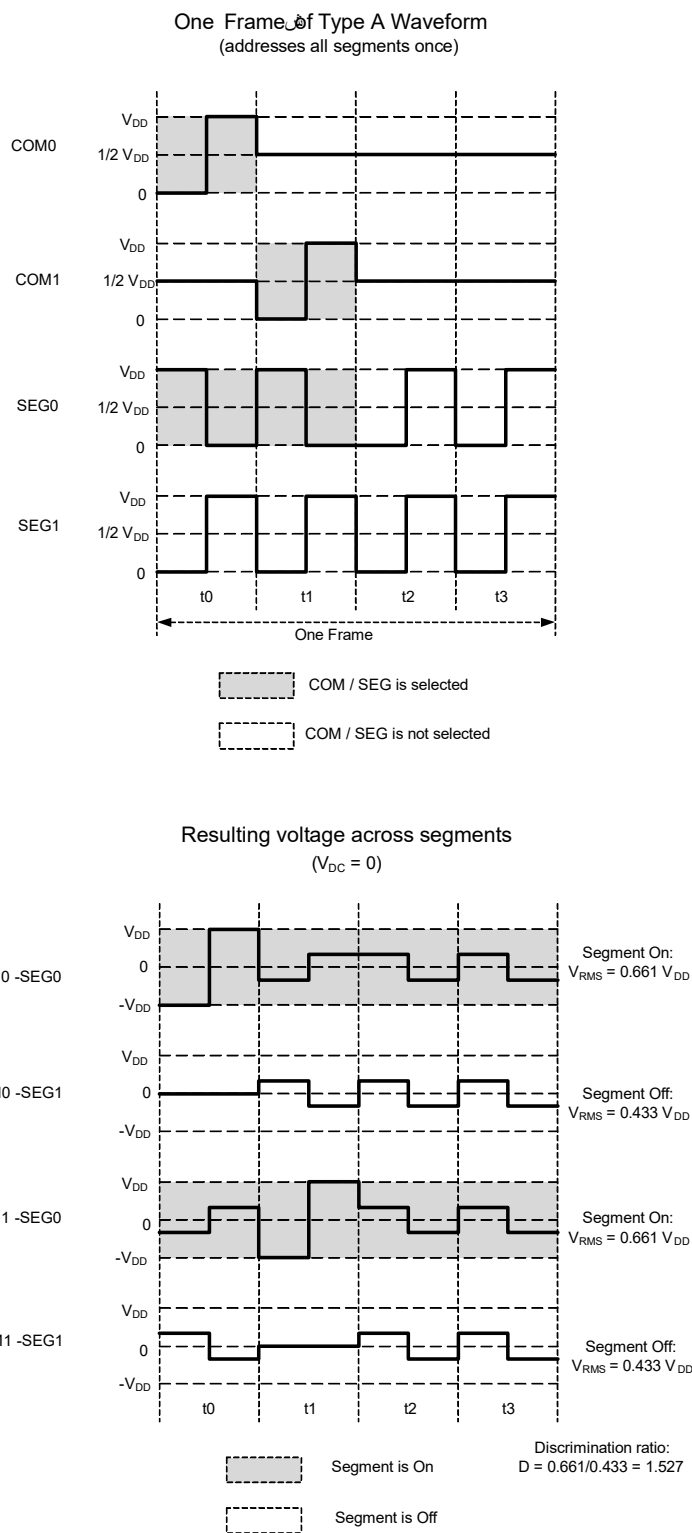


Figure 20-3. PWM1/2 タイプ B 波形の例

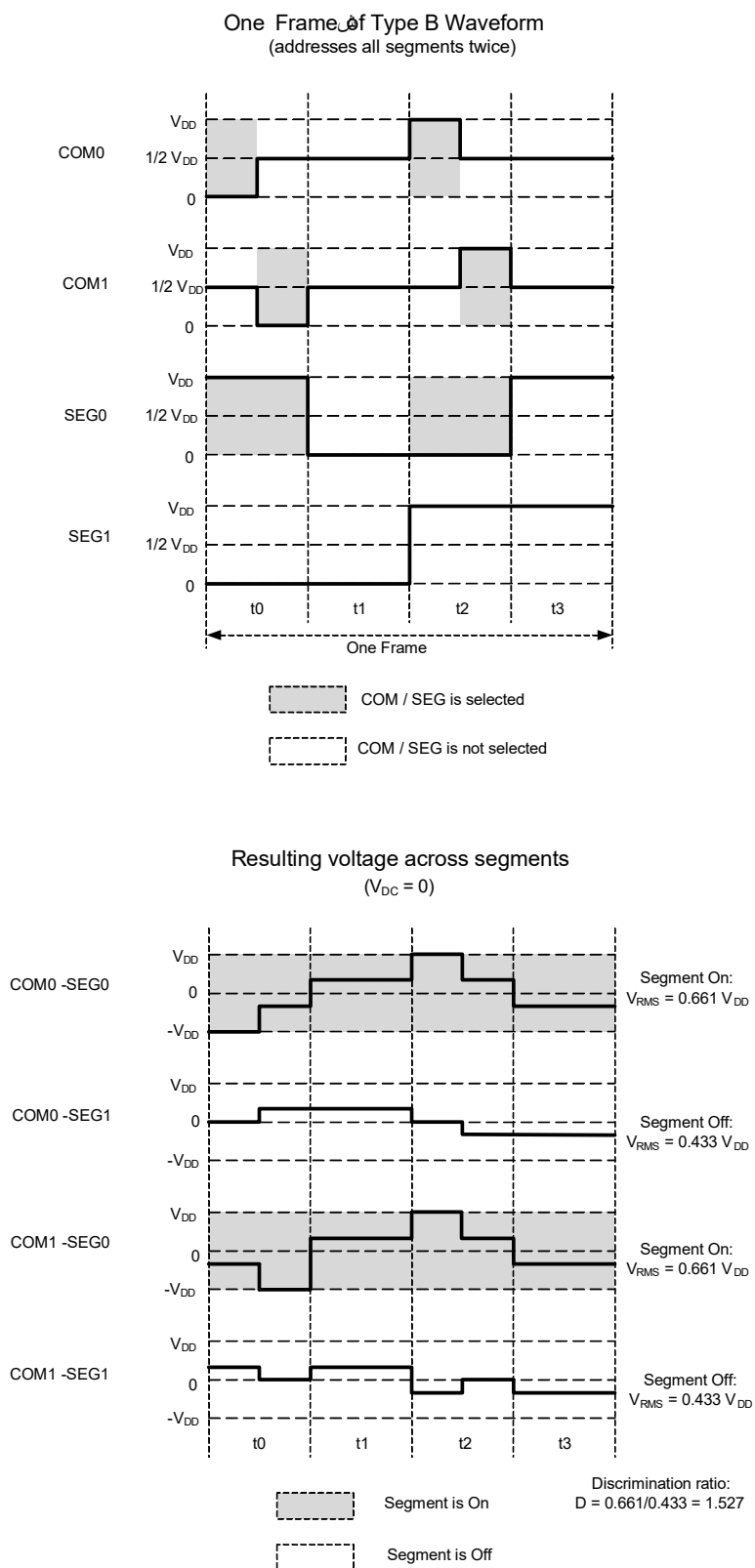


Figure 20-4. PWM1/3 タイプ A 波形の例

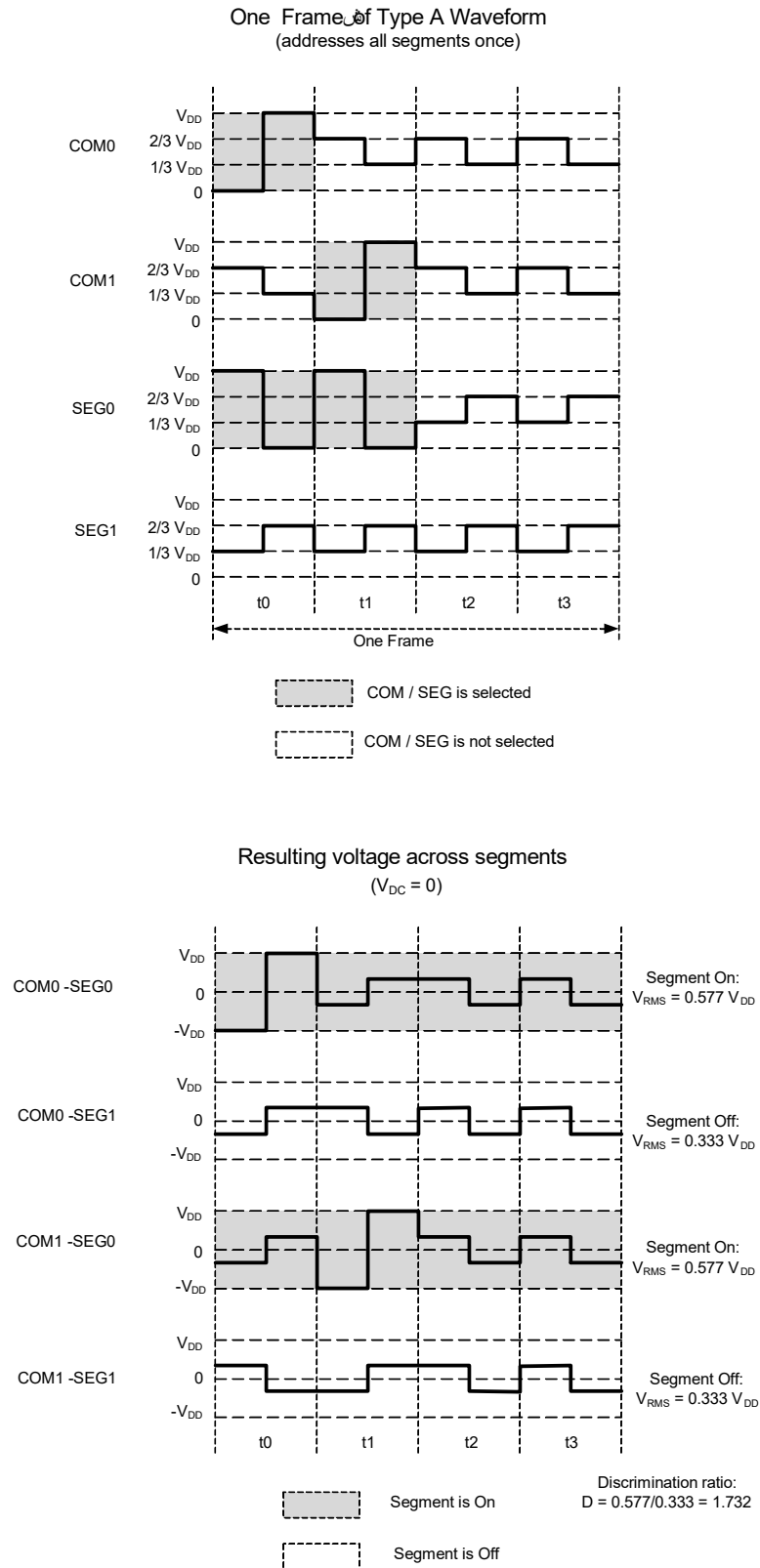
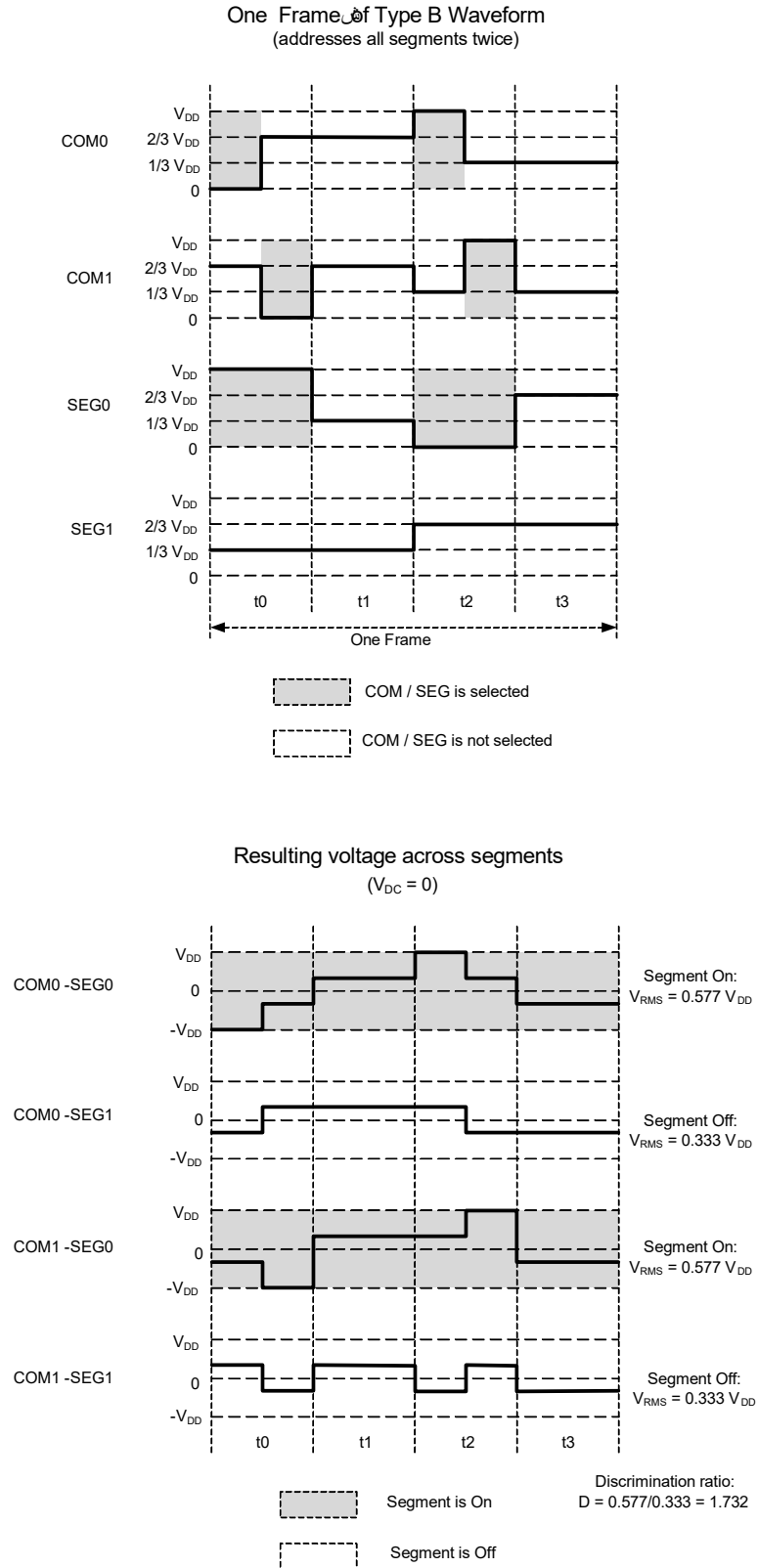


Figure 20-5. PWM1/3 タイプ B 波形の例



オンおよびオフの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：

$$V_{RMS(Off)} = \sqrt{\frac{2(B-2)^2 + 2(M-1)}{2M}} \times \left(\frac{V_{DRF}}{B}\right)$$

Equation 20-1

$$V_{RMS(On)} = \sqrt{\frac{2B^2 + 2(M-1)}{2M}} \times \left(\frac{V_{DRF}}{B}\right)$$

Equation 20-2

ここに B はバイアスであり、M はデューティ (COM の数) です。

たとえば、COM の数が 4 の場合、1/2 バイアスと 1/3 バイアスの結果の判別比 (D) は、それぞれ 1.528 と 1.732 です。1/3 バイアスは、2 つおよび 3 つの COM ドライブでも識別率を向上させます。従って 1/3 バイアスは 1/2 バイアスより良いコントラストを提供し、ほとんどのアプリケーションに推奨されます。1/4 および 1/5 バイアスは、LCD の高速動作でのみ使用できます。これは、COM の多い設計 (4 つ以上) で使用される場合、特に良い識別比率を提供します。

LCD を低速動作させる時、PWM 信号は ILO から生成されます。32 kHz 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容量ディスプレイを駆動するには、100k-1MΩ の追加の外部直列抵抗を使用する必要があります。外部抵抗は 1MHz 程度より大きい PWM 周波数の場合、必要ありません。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 バイアスモードには、PWM が COM 信号でのみ必要であるという利点があります。Figure 20-2 および Figure 20-3 に示すように、SEG 信号は論理レベルのみを使用します。

20.2.1.2 デジタル相関

デジタル相関モードは、バイアス電圧の生成ではなく、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 セグメントの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 信号のベース駆動周波数を 2 倍にします。それにより COM と SEG 駆動信号の位相関係を、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 ドライブアプローチのように信号の DC レベルを変化させることとは異なります。Figure 20-8 および Figure 20-9 は、動作原理を示す波形例です。

Figure 20-6. デジタル相関タイプ A 波形

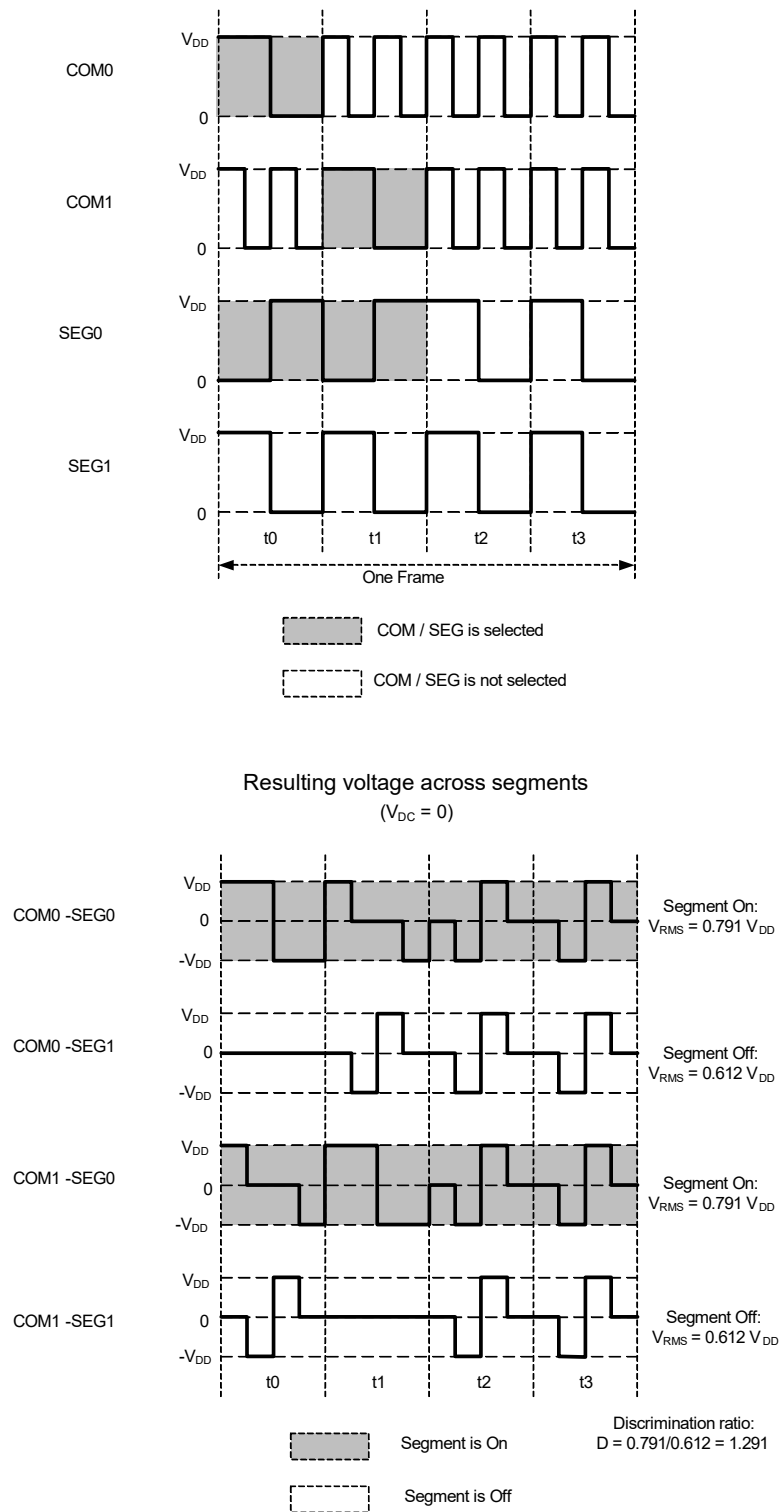
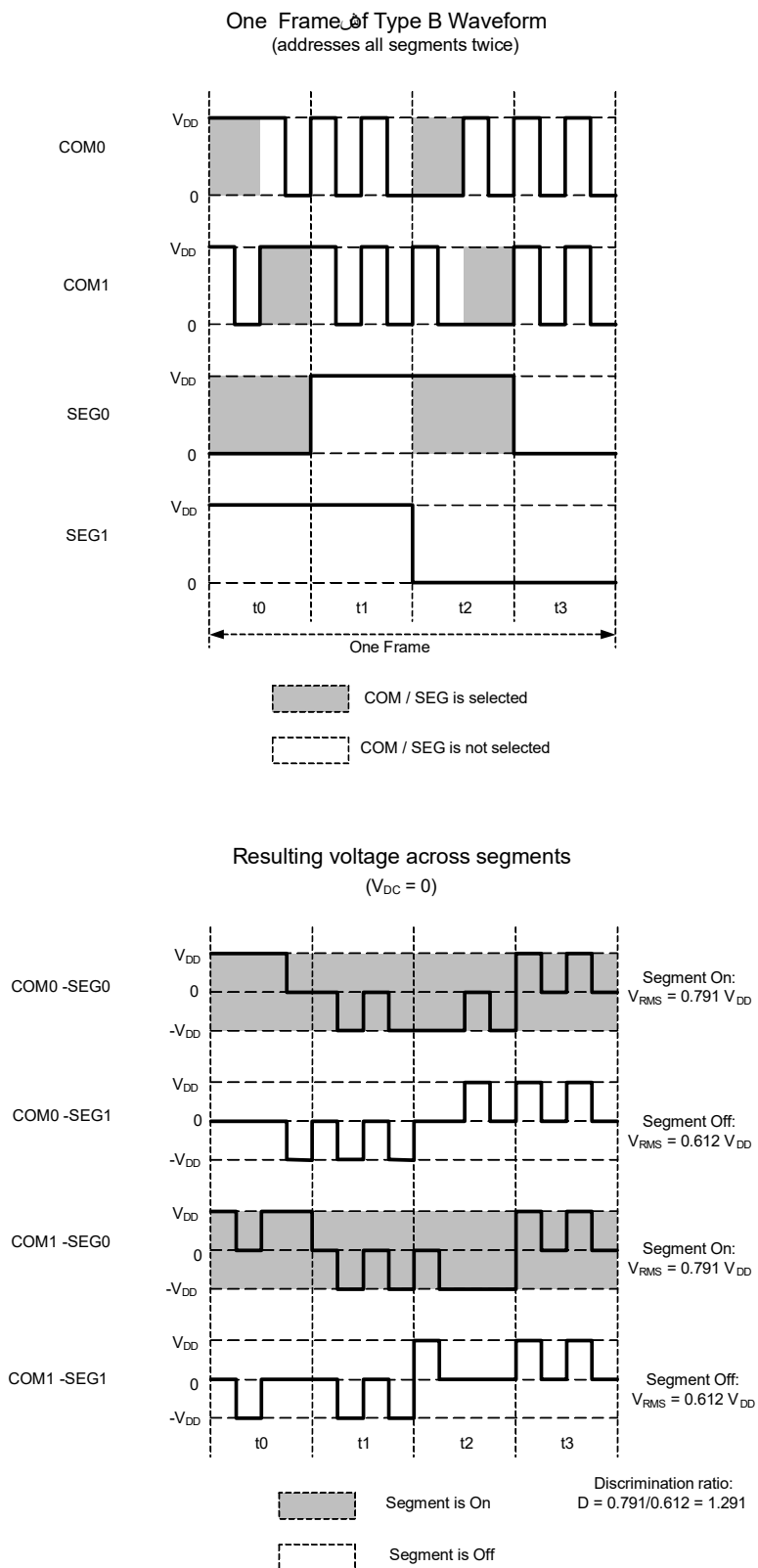
 One 'Frame' of Type A Waveform
 (addresses all segments once)


Figure 20-7. デジタル相関タイプ B の波形



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できます：

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

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

ここに B はバイアスであり、M はデューティ (COM の数) です。これにより、4 つの COM の識別率 (D) が 1.291 になります。デジタル相関モードには、1.8 V V_{DD} から 3 V ディスプレイを駆動する機能もあります。

20.2.2 駆動モードの推奨使用方法

[20.2.1.1 PWM 駆動](#)と [20.2.1.2 デジタル相関](#)で説明されているように、PWM ドライブモードはデジタル相関モードと比較して識別率が高くなっています。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を以下に示します。

Table 20-1. 駆動モードの推奨使用方法

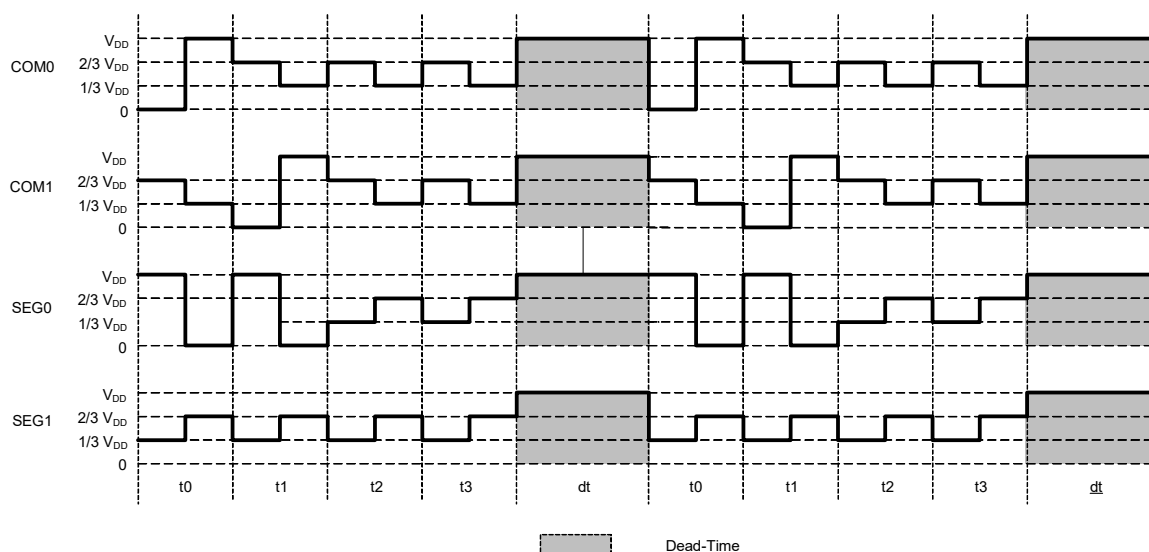
ディスプレイのタイプ	ディープスリープモード	スリープ / アクティブモード	注意事項
4 つの COM がある TN 液晶	デジタル相関	1/3 バイアス PWM	ファームウェアはディープスリープモードへの移行またはアクティブモードへの復帰前に、LCD 駆動モードを切り替える必要がある
4 つの COM がある STN 液晶	デジタル相関		STN 液晶ではコントラストに関する PWM 駆動の利点がない
8 COM、STN	サポートされていません	PWM 1/4 バイアスおよび 1/5 バイアス	高速 LCD モードでのみサポート。低速クロックは PWM を高いマルチプレクス比で動作するために十分に速くない

20.2.3 デジタルコントラストコントロール

すべての駆動モードで、デジタルコントラスト制御はセグメントのコントラストを変更するのに使用されます。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。デッドタイムは細かい分解能で制御できます。[Figure 20-8](#) は、1/3 バイアスおよび 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示しています。

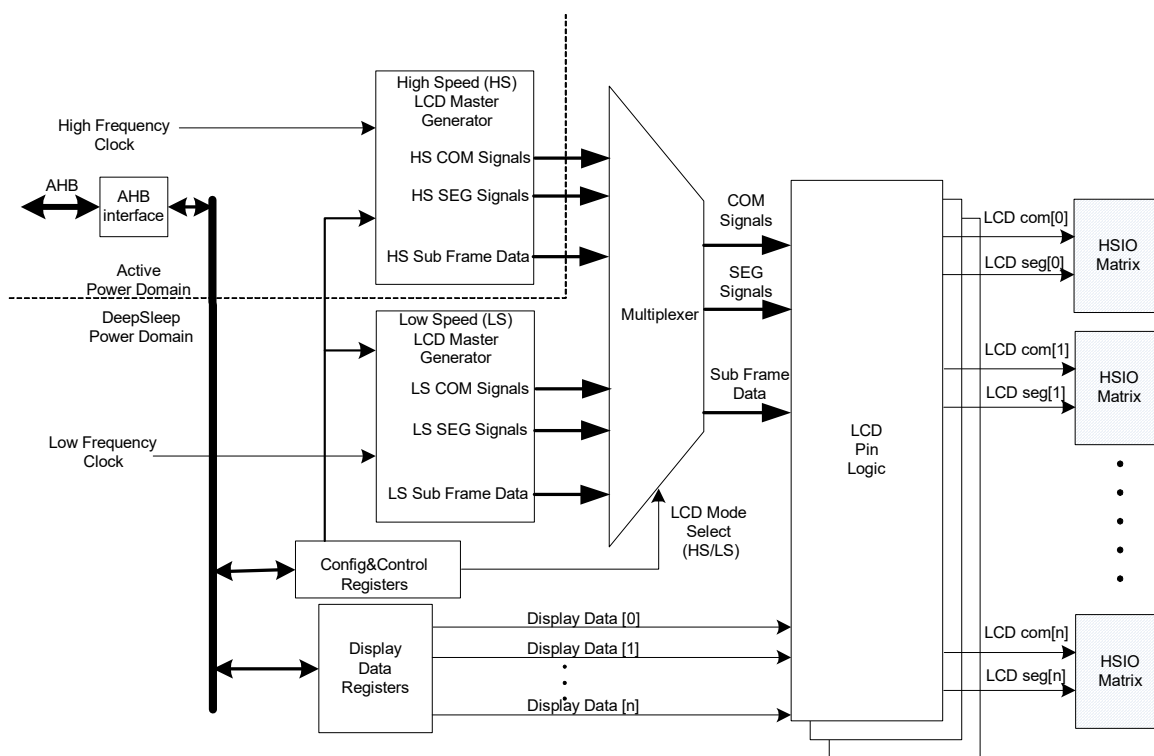
Figure 20-8. デッドタイム コントラスト制御

Two Frames of of Type A Waveform with Dead-time
(Example for $1/4^{\text{th}}$ Duty and $1/3^{\text{rd}}$ bias)



20.3 ブロック図

Figure 20-9. LCD ダイレクト駆動システムのブロック図



20.3.1 動作方法

LCDコントローラ ブロックは2つの信号発生器を持ちます。1つの信号発生器は高速クロック ソース HFCLK を持ち、もう1つは ILO から生成された低速クロック ソースを持ちます。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關駆動モードをサポートします。低速ジェネレータを使用した PWM ドライブモードでは、“PWM 駆動” (224 ページ) で説明されているように、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 のこの構成可能なピン割り当ては、GPIO および I/O マトリックスに実装されています。“高速 I/O マトリックス” (71 ページ) を参照してください。これらの2つの信号発生器は同じコンフィギュレーション レジスタを共有します。I/O レジスタをマッピングするこれらのメモリは、AHB インターフェースを使うシステム バス (AHB) に接続されます。

LCD コントローラはアクティブ、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。

20.3.2 高速および低速のマスター信号発生器

高速および低速のマスター信号発生器は相互に似ています。1つの違いは、高速バージョンがフレームとサブフレーム周期を発生させるために、分周比の大きな分周器を利用することです。これは、高速ブロック (HFCLK) のクロックが、低速ブロックに送られる ILO クロックの周波数に比べて一般的に 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 の駆動波形用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：
 - デジタル相關
 - 1/2 バイアス PWM
 - 1/3 バイアス PWM
 - 1/4 バイアスでの PWM (低速信号発生器でサポートしない)
 - 1/5 バイアスでの PWM (低速信号発生器でサポートしない)
 - オフ / ディセーブル。2つある信号発生器の内1つがオフになるように通常設定される

LCD_CONTROL レジスタにある OP_MODE および BIAS フィールドで駆動モードを選択します。

- サブフレームのタイミングを生成するためのカウンタ。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 $4 \times (C+1)$ です。低速信号発生器は 8 ビットカウンタを持っている。a-This counter generates a maximum half sub-frame period of 8ms from the ILO clock. 高速信号発生器は 16 ビットカウンタを持っている。
- デッドタイム周期を生成するためのカウンタ。これらのカウンタはサブフレーム周期カウンタ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

20.3.3 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。

20.3.4 ディスプレイ データ レジスタ

各 LCD セグメントピンは、独自の表示データレジスタ LCD_DATA_nx を備えた 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、セグメントを共通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_DATA_nx 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 の組合せのすべてのオン / オフ データを保持します。LCD_DATA0x は COM0 ~ COM3 用の SEG-COM データを、LCD_DATA1x は COM4 ~ COM7 用の SEG-COM データを保持します。各

LCD_DATA0x レジスタのビット [4i + 3:4i] ('i' はピン番号) は、Port [x] および COM [3,2,1,0] の Pin [i] の ON/OFF データを表します。Table 20-2 に示すような組み合わせ。LCD_DATAx レジスタは、各フレームの表示データに従っ

てプログラムする必要があります。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。

Table 20-2. LCD_DATA0x レジスタの SEG-COM マッピング (各 SEG は LCD ポートのピン)

ビット [31:28] = PIN_7[3:0]				ビット [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
ビット [23:20] = PIN_5[3:0]				ビット [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
ビット [15:12] = PIN_3[3:0]				ビット [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
ビット [7:3] = PIN_1[3:0]				ビット [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

20.4 レジスタ一覧

Table 20-3. LCD 直接駆動レジスタ一覧

レジスタ名	説明
LCD_ID	このレジスタには、LCD controller の情報とリビジョン番号があります。
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 ~ COM3 の LCD ポート ピン データ レジスタ (x = ポート番号、8 ポートを使用可能)
LCD_DATA1x	COM4 ~ COM7 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)

Section E: アナログシステム

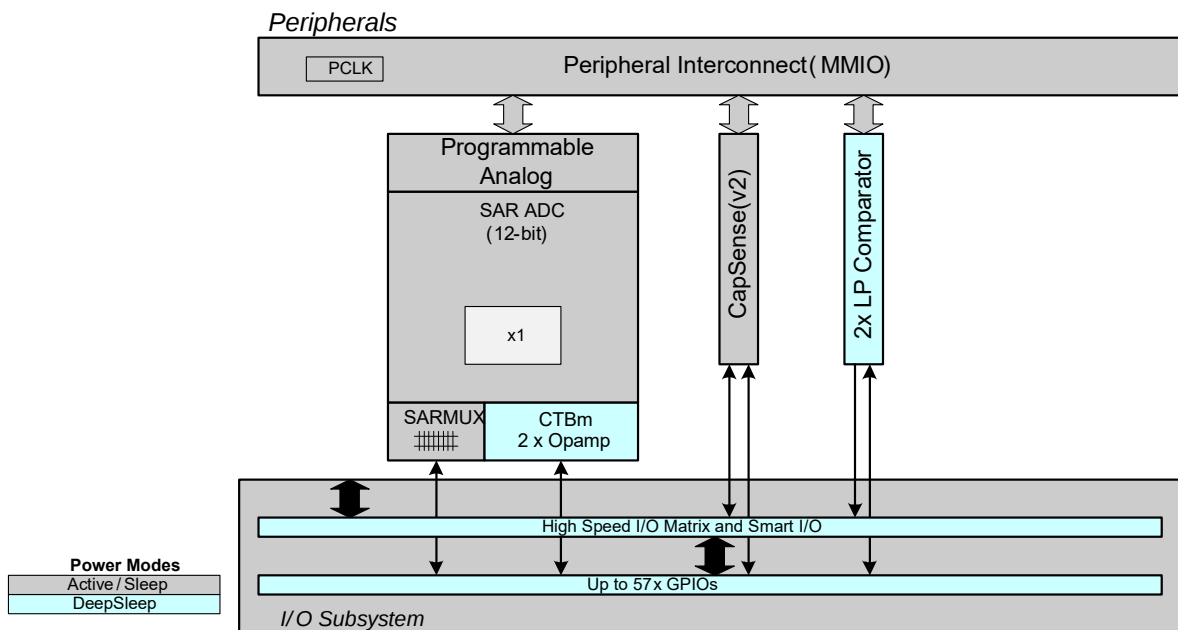


このセクションは次の章を含みます。

- SAR ADC (238 ページ)
- 低消費電力コンパレータ (266 ページ)
- 連続時間ミニブロック (CTBm) (273 ページ)
- CapSense (283 ページ)
- 温度センサー (284 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



21. 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。次のブロックで構成されています (Figure 21-1 を参照)。

- SARMUX
- SAR ADC コア
- SARREF
- SARSEQ

SAR ADC コアは サンプリング レートが 1Msps の高速 12 ビット ADC 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 16 本の内部チャン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 コントローラ SARSEQ は SARMUX と SAR ADC を制御し、CPU の介入なく有効になったチャン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。

各チャンネルの結果は二重バッファされ、スキャンの完了で割込みを生成することもできます。シーケンサは、割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。

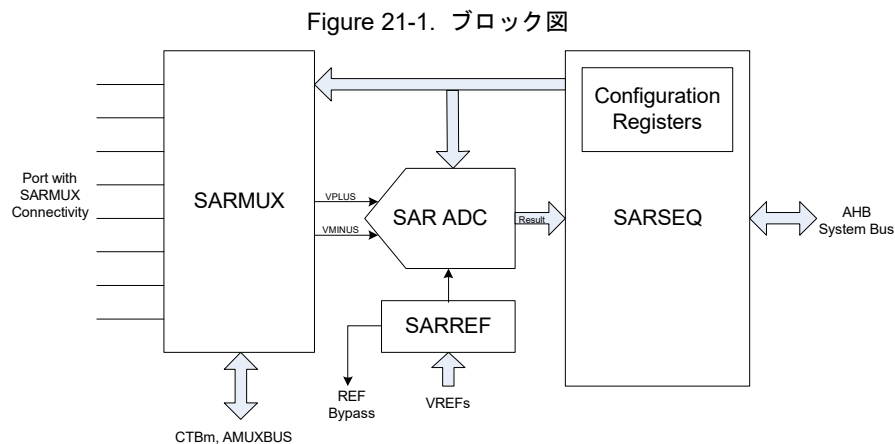
柔軟性をもっと高めるために、ファームウェアにより SARMUX 内のアナログ スイッチを含むほとんどのアナログ スイッチを制御することが可能です。これによりファームウェアで代替シーケンサを実装することができます。

21.1 特長

- デバイスの全電源電圧範囲で動作可能
- 最大 1Msps のサンプリング速度
- 個別に設定可能なチャンネル 16 本とインジェクションチャンネル 1 本
- 各チャンネルの持つ特長：
 - 外部ピンからの入力 (シングルエンド モードの 8 つのチャンネルおよび差動モードでの 4 つのチャンネルのみ) または内部信号 (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、CTBm コンパレータ、低消費電力コンパレータおよび変換信号の SAR エンドによりトリガーされる
 - ハードウェア / ファームウェアトリガー (ワンショット)、およびフリーランニング (連続変換) モード
- ハードウェアによる平均化に対応
 - 1 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
 - 1.2V 内部リファレンス電圧 (バッファあり)
 - 外部リファレンス
- 割込みの生成
 - 完了したスキャン変換
 - 各チャン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
- 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専用低消費電力モードがある

21.2 ブロック図



21.3 動作方法

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：割込み、低消費電力モード、SAR ADC の状態
 - システム運用
- コンフィギュレーション例

21.3.1 SAR ADC コア

PSoC 4 SAR ADC コアは 12 ビット SAR ADC です。この ADC の最大サンプル速度は 1Msps です。SAR ADC コアの特長を以下に示します。

- 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

21.3.1.1 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。 $-V_{REF} \sim +V_{REF}$ の差動入力の全範囲出力 (0 ~ 4095) を提供します。SAR ADC は反転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 モードはチャンネル 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。

反転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、[PSoC 4100S datasheet](#) および [PSoC 4100S Plus datasheet](#) 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。 V_{minus} 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ウ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) がグラウ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 の反転入力に接続する必要があり、入力範囲は $0 \sim 2 \times V_{REF}$ です。

温度センサーはシングルエンドモードでのみ使用できます。SAR_CTRL [11:9] を 0 に書き込みます。差動変換は温度センサーでは使用できません。結果は未定義です。

21.3.1.2 入力範囲

すべての入力は $V_{SSA} \sim V_{DDA}$ 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。反転入力の電圧が V_n で、ADC リファレンス電圧が V_{REF} である場合、非反転入力の電圧は $V_n \pm V_{REF}$ 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。シングルエンド モードでは、 V_n は V_{SSA} 、 V_{REF} または外部入力に接続します。

$V_n \pm V_{REF}$ は $V_{SSA} \sim V_{DDA}$ の範囲内にある必要があることに注意してください。たとえば一入力が V_{SSA} に接続した場合、+入力の範囲は $-V_{REF} \sim V_{REF}$ ではなく $0 \sim V_{REF}$ 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。非反転入力信号が 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。

21.3.1.3 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。

Table 21-1. 結果データのフォーマット

アライメント	署名済み / 未署名	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
Right (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
Right (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
Left (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	–	–	–	–	–	–	–	–

21.3.1.4 反転入力選択

負の入力接続の選択は、電圧範囲、SNR、および有効分解能に影響します (Table 21-2)。シングルエンド モードでは SAR ADC の反転入力は 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。

Table 21-2. 反転入力選択の比較表

シングルエンド / 差動	署名済み / 未署名	SARMUX Vminus	SARMUX Vplus Range	結果レジスタ	最大 SNR
Single-ended (シングルエンド)	N/A ^a	V_{SSA}	$+V_{REF}$ $V_{SSA} = 0$	0x7FF 0x000	より良い
Single-ended (シングルエンド)	符号無し	V_{REF}	$+2 \times V_{REF}$ V_{REF} $V_{SSA} = 0$	0xFFF 0x800 0	良い
Single-ended (シングルエンド)	符号付き	V_{REF}	$+2 \times V_{REF}$ V_{REF} $V_{SSA} = 0$	0x7FF 0x000 0x800	良い
Single-ended (シングルエンド)	符号無し	V_x	$V_x + V_{REF}$ V_x $V_x - V_{REF}$	0xFFF 0x800 0	最も良い
Single-ended (シングルエンド)	符号付き	V_x	$V_x + V_{REF}$ V_x $V_x - V_{REF}$	0x7FF 0x000 0x800	最も良い
Differential	符号無し	V_x	$V_x + V_{REF}$ V_x $V_x - V_{REF}$	0xFFF 0x800 0	最も良い
Differential	符号付き	V_x	$V_x + V_{REF}$ V_x $V_x - V_{REF}$	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 の反転入力に接続する必要があり、入力範囲は $0 \sim 2 \times V_{REF}$ 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氣的に差動モードに相当することに注意してください。しかし各差動ペアの奇数ピンが共通の代替グラウ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) がグラウンド電圧を下回ることができないからです。

21.3.1.5 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: チャンルごとに 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:

$$\text{変換時間}(\text{sar_clk}) = \text{分解能}(\text{ビット}) + 2$$

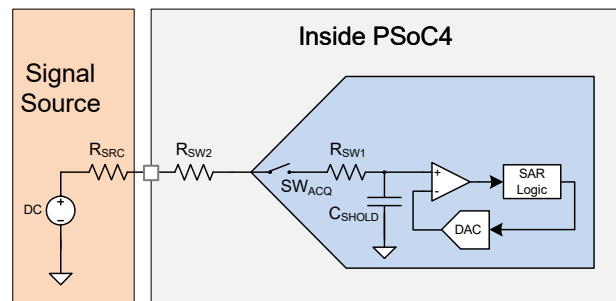
$$\text{取得時間と変換時間の和}(\text{sar_clk}) = \text{取得時間} + \text{分解能}(\text{ビット}) + 2$$

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 18 MHz の場合、変換には 18 の sar_clk が必要で、変換速度は 1Msps になります。分解能が低いほど変換速度は高くなります。

21.3.1.6 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。各チャンネルはグローバル 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。

Figure 21-2. 取得時間



取得時間は、Figure 21-2 に示すように、ルーティングパスの抵抗を介して ADC の内部ホールドコンデンサを充電するのに十分でなければなりません。取得時間の推奨値は:

$$t_{ACQ} \geq 9 \times (R_{SRC} + R_{SW2} + R_{SW1}) \times C_{SHOLD}$$

以下の場合:

$$C_{SHOLD} \approx 10 \text{ pF}$$

$R_{SW2} + R_{SW1}$ = ルーティングパスに応じて、~ 500 ~ 1000 オーム (詳細については、“アナログ配線” (244 ページ) を参照)。

R_{SRC} = 信号ソースの直列抵抗

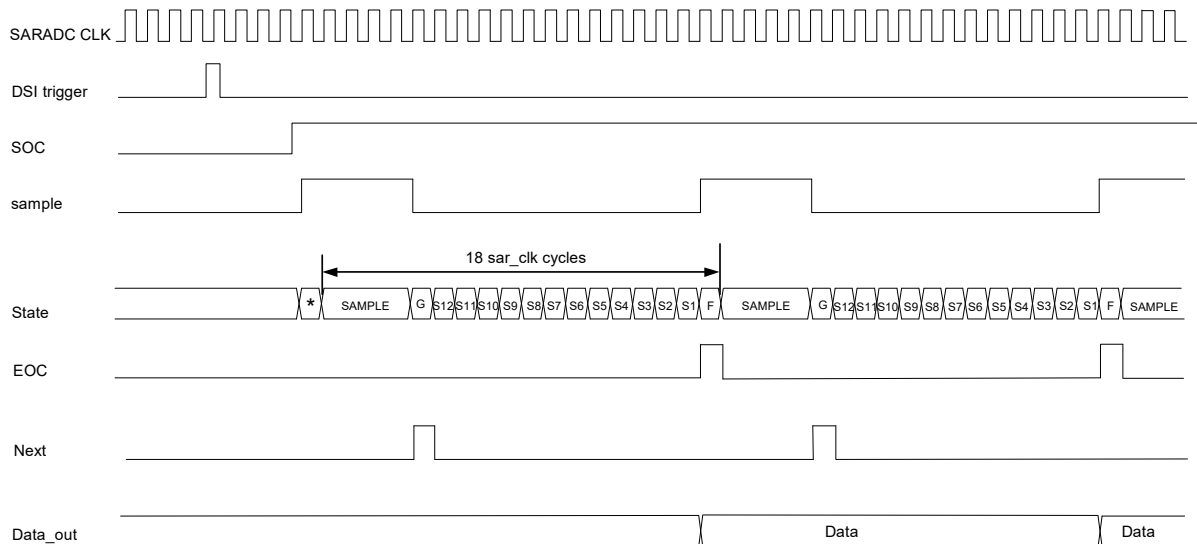
21.3.1.7 SAR ADC クロック

SAR ADC クロック周波数は 1MHz ~ 18MHz でなければなりません。これは、HFCLK からクロック分周器を介して供給されます。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。1Msps のサンプル速度を達成するために 18MHz SAR ADC クロックが必要です。このためにはシステム クロック (HFCLK) を 48MHz に代わり 36MHz に設定する必要があります。4 クロック (18MHz) の最小取得時間で 12 ビット ADC 変換には、完了するまでに 18 クロックが必要です。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。18 MHz での 4 クロックサイクルの最小取得時間は、SAR ブロック (Figure 21-2 の R_{SW1} および C_{SHOLD}) でサポートされている最小取得時間 (194 ns) に基づいていることに注意してください。

21.3.1.8 SAR ADC タイミング

Figure 21-3 は、SAR ADC のタイミング図を示しています。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。サンプリング (取得) の後、次のパルスを出力します。SARMUX は別のピンと信号にルーティングできます。これは、シーケンサ制御で自動的に行われます (詳細については、“SARSEQ” (251 ページ) を参照してください)。

Figure 21-3. SAR ADC タイミング



21.3.2 SARMUX

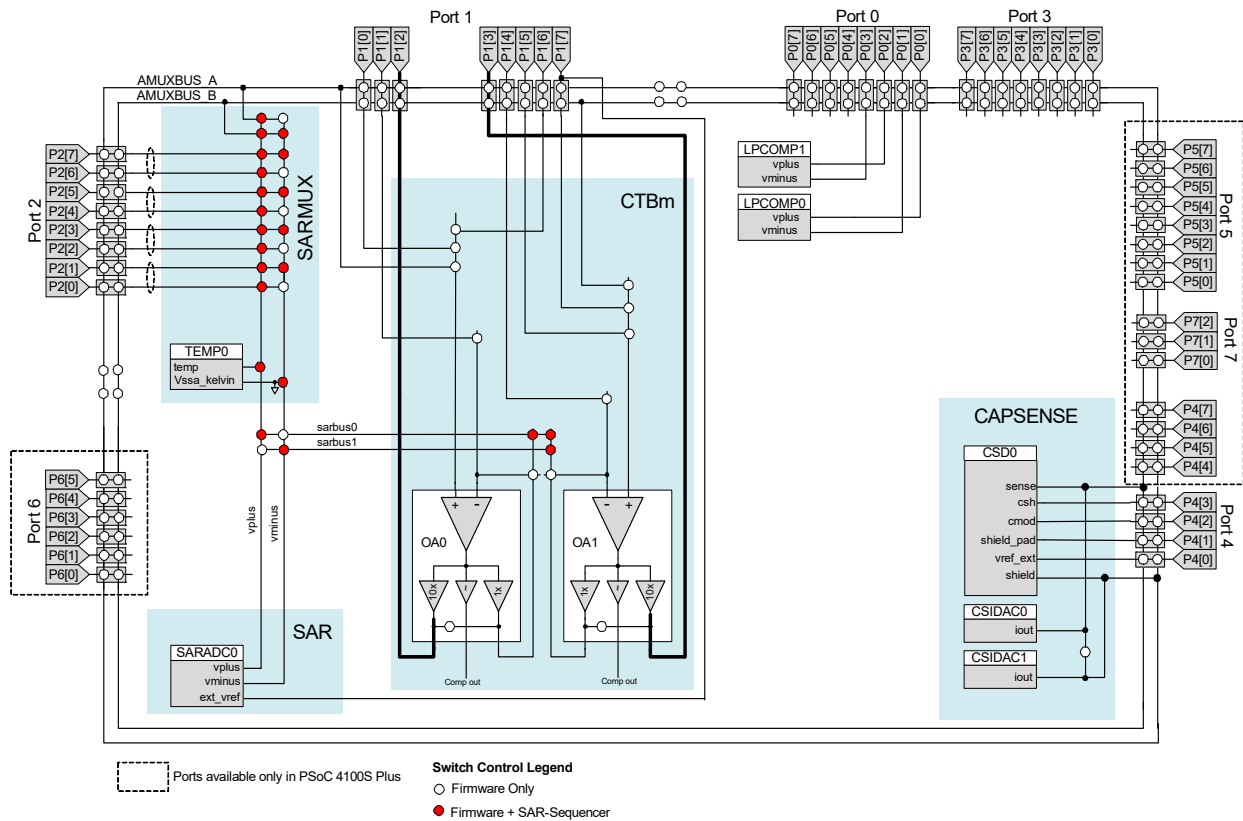
SARMUX はプログラム可能な専用アナログ マルチプレクサです。SARMUX の特長を以下に示します。

- 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 コントローラ ブロック (SARSEQ) またはファームウェアで制御
- 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
 - $V_{DDA} \geq 4.0V$ の場合、チャージポンプはオフになり、出力として V_{DDA} を供給します
- 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 でサンプリングするには十分な速度ではありません)
 - AMUXBUS_A/B (1Msps でサンプリングするほど高速ではない)

21.3.2.1 アナログ配線

SARMUX には、SARSEQ ブロック (シーケンサコントローラ) またはファームウェアによって制御できる多くのスイッチがあります。シーケンサ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。Figure 21-4 を参照してください。

Figure 21-4. SARMUX スイッチおよび制御機能



シーケンサ制御: スイッチは SARSEQ ブロック内のシーケンサで制御されます。各 channel's のアナログルーティングを設定した後、CPU の介入なしに、ラウンドロビン方式でマルチチャンネル自動スキャンを有効にします。すべてのスイッチがシーケンサによって制御できるわけではありません。Figure 21-4 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。

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 のすべてのスイッチを制御できます。Figure 21-4 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。ただしマルチチャンネルの取得には CPU の介入が必要となります。対応するレジスタは、SAR_MUX_SWITCH0、SAR_MUX_SWITCH_HW_CTRL です。および SAR_CTRL。

21.3.2.2 アナログ相互接続

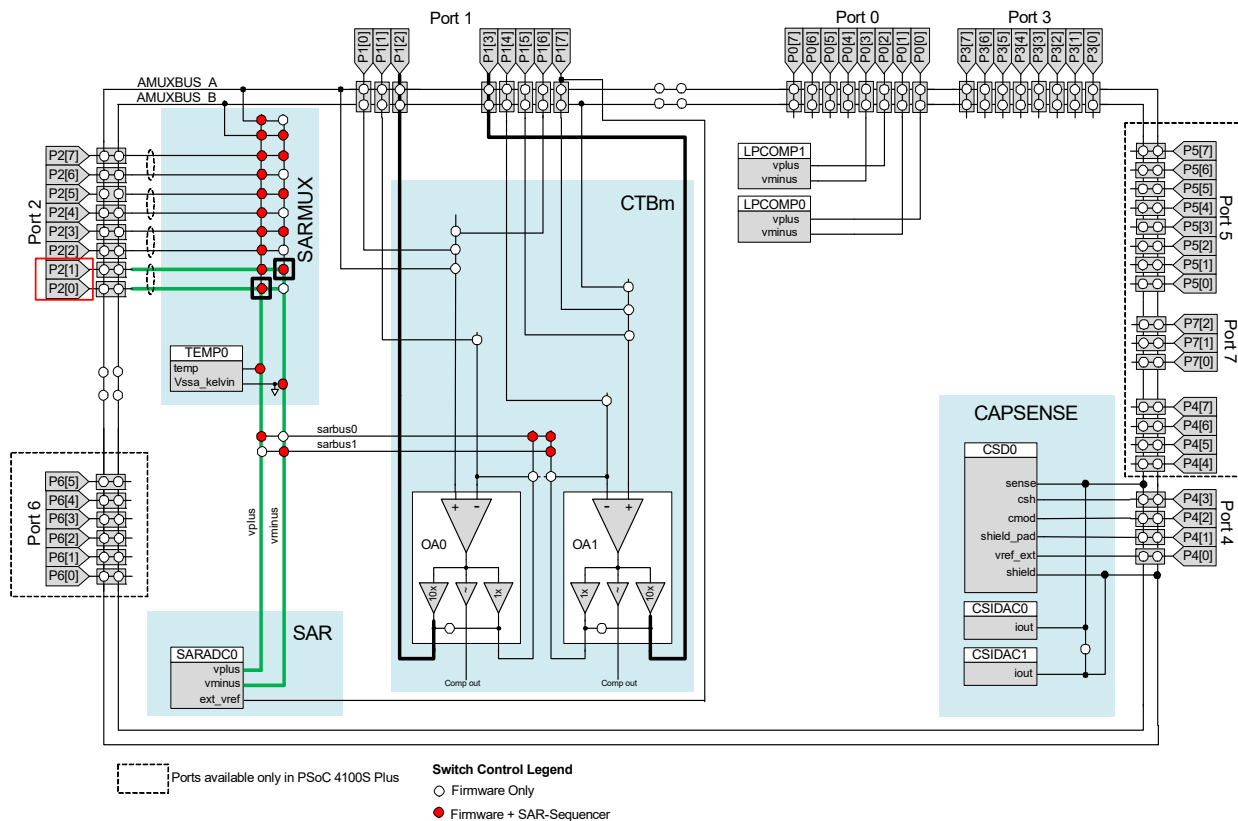
PSoC 4 のアナログ相互接続は非常に柔軟です。SAR ADC は SARMUX を介して外部ピンと内部信号を含む複数の入力に接続できます。たとえば CTBm など隣接したブロックに接続できます。また AMUXBUS_A/B を介してポート 2 以外のピンに接続できますが、スキャン性能が低下します (寄生カップリングが高くなり、セトリングされるまでの RC 時間が長くなります)。

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

Figure 21-5 は、SARMUX をサポートする 2 つの GPIO が、スイッチを介して差動ペア (Vpuls/Vminus) として SAR ADC に接続される方法を示しています。これらの 2 個のスイッチはシーケンサまたはファームウェアで制御できます。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] や P2[2] など、差動ペアとしてペアになっていないピンを使用する必要がある場合、シーケンサは機能しません。ファームウェアを使用します。

Figure 21-5. 外部ピンからの入力

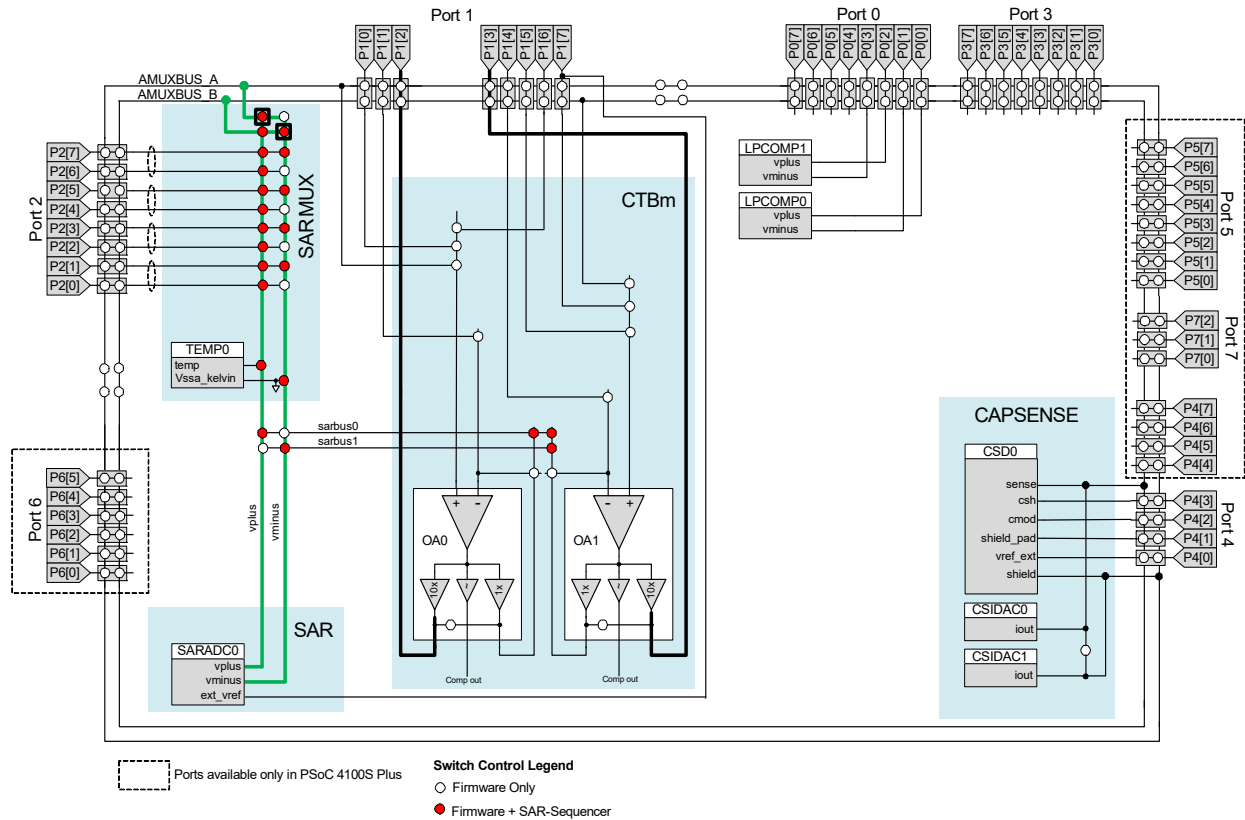


アナログバスからの入力 (AMUXBUS_A/B)

Figure 21-6 は、SARMUX 接続をサポートしない 2 つのピンが差動ペアとして ADC に接続される方法を示しています。追加のスイッチは、これらの 2 つのピンを AMUXBUS_A および AMUXBUS_B に接続し、次に AMUXBUS_A および AMUXBUS_B を ADC に接続する必要があります。

スイッチを追加すると、スキャンパフォーマンスが低下します (寄生結合が増加し、安定するまでの RC 時間が長くなります)。1Msps でサンプリングするには十分な速度ではありません。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)。

Figure 21-6. 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。Figure 21-7 は、オペアンプ (フォロワーとして構成) 出力をシングルエンド SAR ADC に接続する方法を示しています。負端子は V_{REF} に接続されています。Figure 21-8 は、2 つ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ています。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。追加のスイッチもあるため、1Msps でサンプリングするには十分な速度ではありません。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。

Figure 21-7. sarbus を介した CTBm 出力からの入力

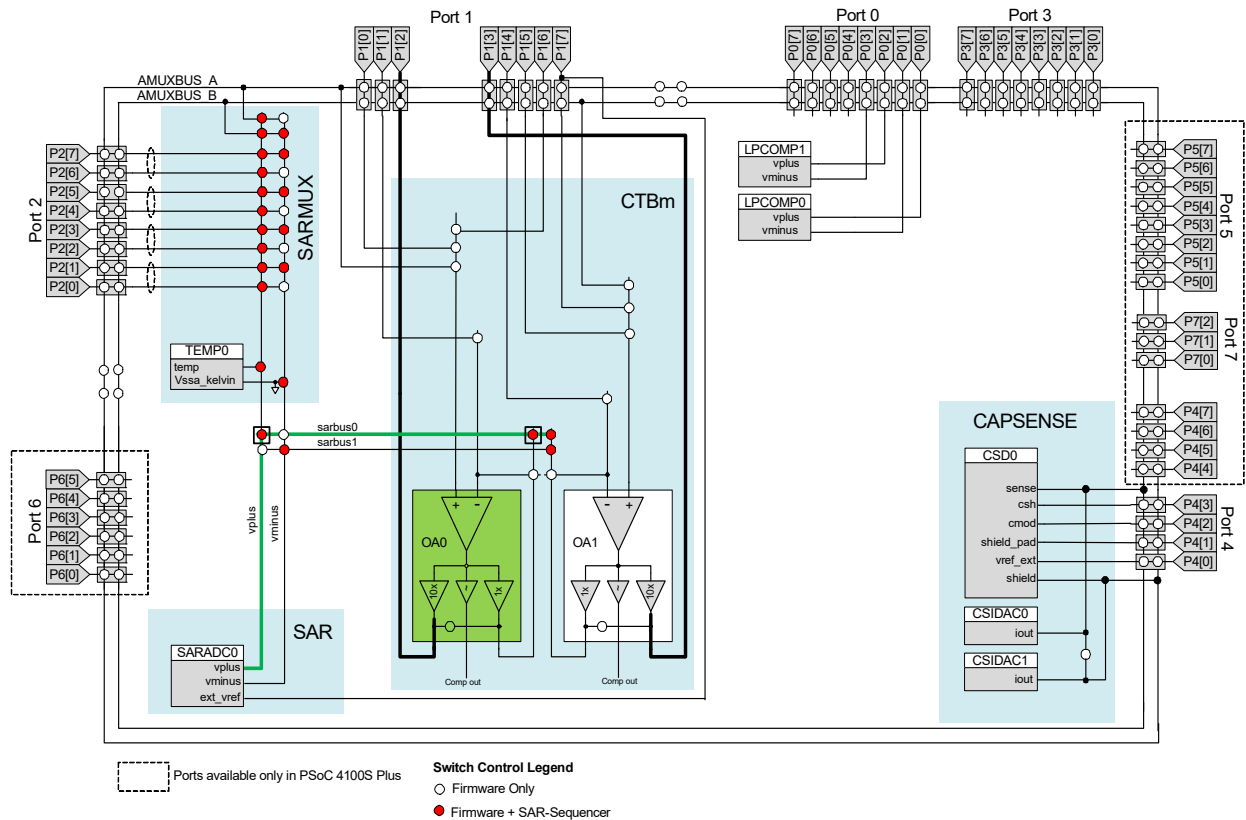
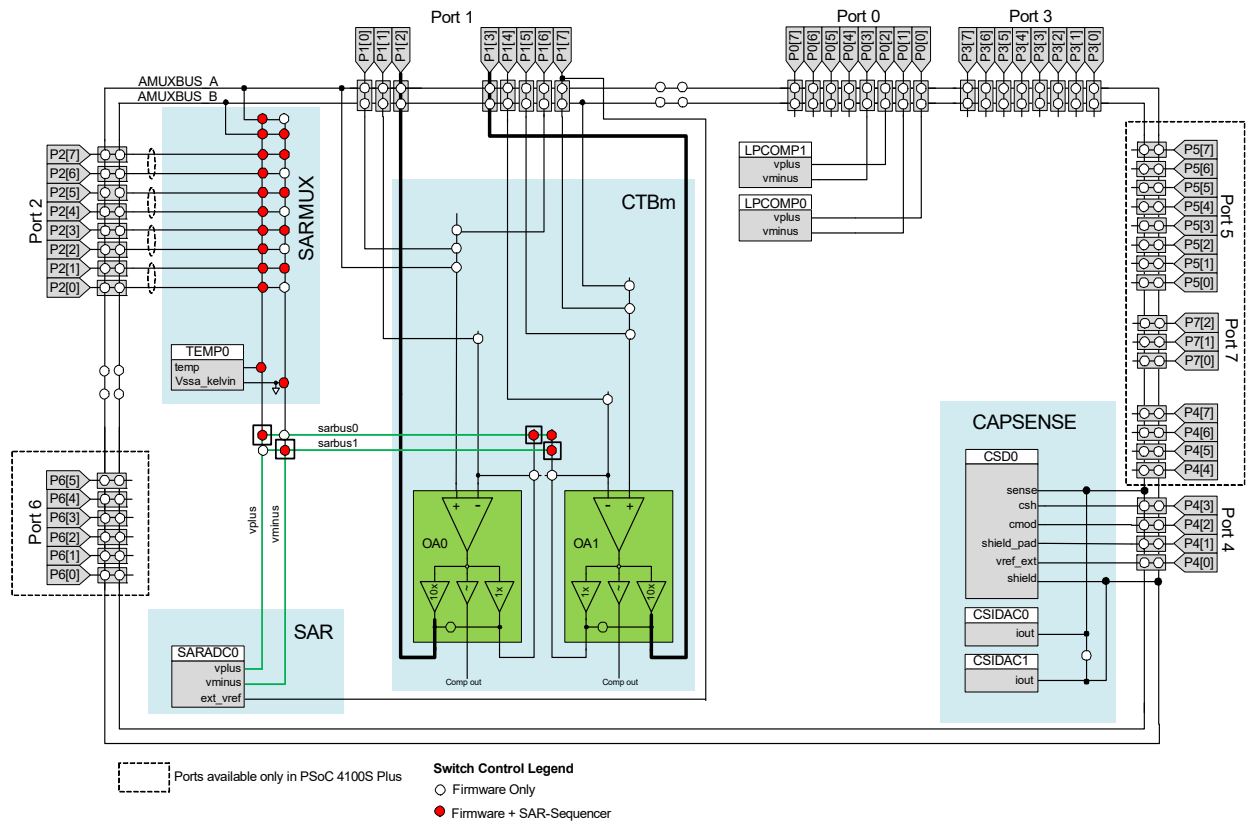


Figure 21-8. sarbus0 と sarbus1 を介した CTBm 出力からの入力

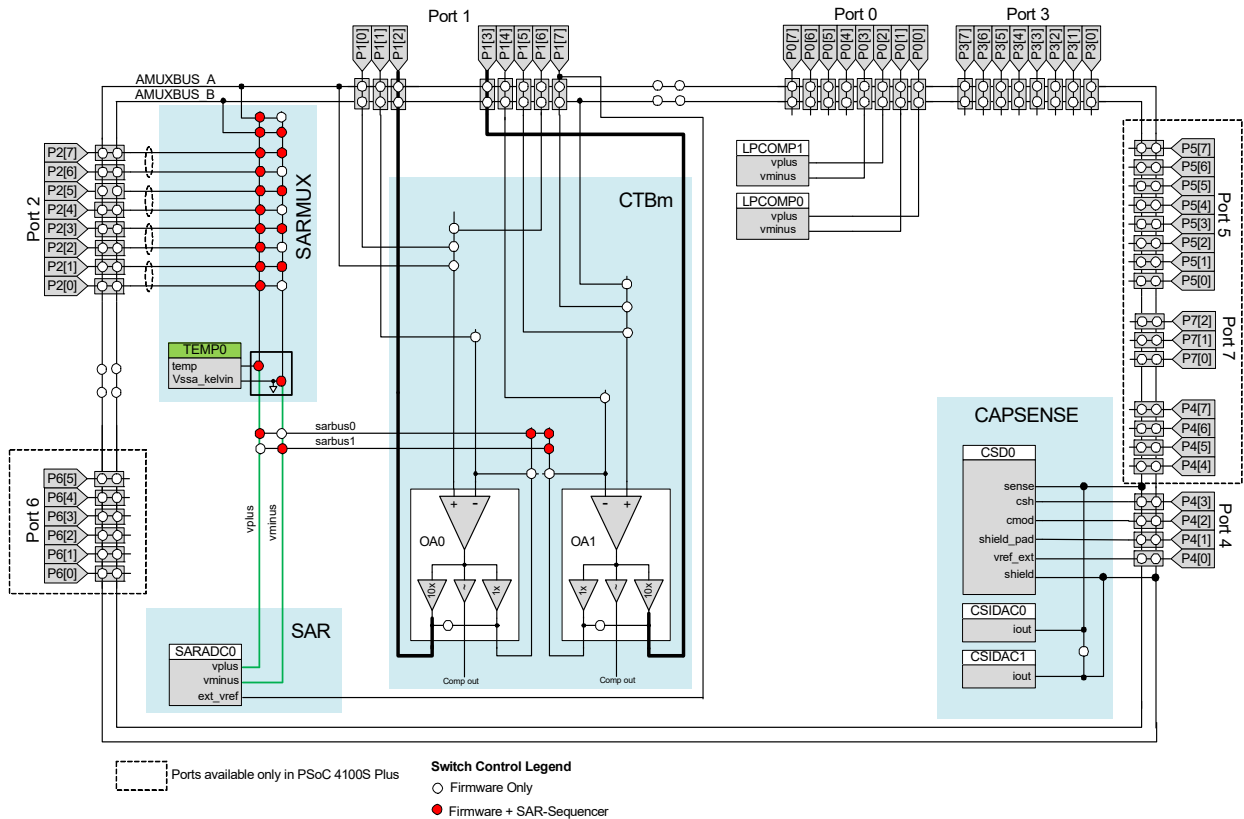


温度センサーからの入力

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ることに注意してください。

Figure 21-9 に示すように、温度センサーはスイッチを介して SAR ADC の正入力にルーティングでき、シーケンサ、ファームウェアで制御できます。MUX_FW_TEMP_VPLUS ビット (SAR_MUX_SWITCH0[17]) を設定すると、温度センサーを有効にして、その出力を SAR ADC の VPLUS に接続できます。このビットをクリアすると、バイアス電流がカットされて温度センサーが無効になります。

Figure 21-9. 温度センサーからの入力

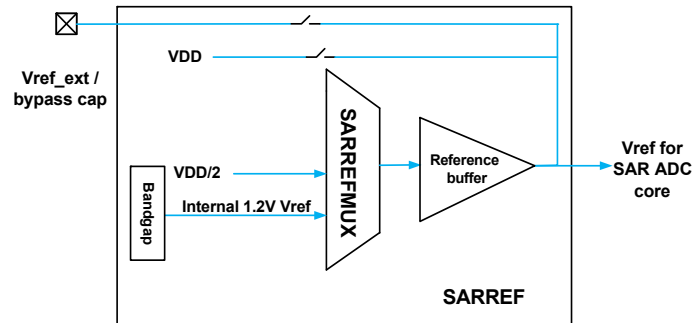


21.3.3 SARREF

SARREF の特長を以下に示します。

- リファレンス電圧オプション: V_{DDA} 、 $V_{DDA}/2$ 、1.2V バンドギャップ ($\pm 1\%$)、外部リファレンス電圧
- 内部リファレンス電圧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

Figure 21-10. SARREF ブロック図



21.3.3.1 リファレンス電圧のオプション

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。この選択により、バンドギャップまたは Ext V_{REF} /SAR バイパスピンに接続された外部 V_{REF} から V_{DDA} 、 $V_{DDA}/2$ 、および 1.2 V 内部リファレンスを接続できます (詳細については、[PSoC 4100S datasheet](#) および [PSoC 4100S Plus datasheet](#) 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。

21.3.3.2 バイパスコンデンサ

内部リファレンス電圧 1.2V 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リングされます。このリファレンス電圧は Ext V_{REF} /SAR バイパス ピンに接続できます。このピンでは外付け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 (12 ビット時) に制限されます。たとえば、バイパスコンデンサがなく、1.2 V の内部 V_{REF} がある場合、最大 SAR ADC 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。Table 21-3 に、12 ビット連続モード動作のさまざまなリファレンスモードとその最大周波数 / サンプルレートを示します。

Table 21-3. 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	Max Sample Rate (12 ビット)
バイパスキャップなしの 1.2V 内部 V_{REF}	4	0	有り	1.6MHz	100 ksps
バイパスキャップ付き 1.2V 内部 V_{REF}	4	1	有り	18MHz	1 Msps
外部 V_{REF} (低インピーダンス パス)	5	X	無し	18MHz	1 Msps
バイパス コンデンサを使わない場合の $V_{DDA}/2$	6	0	有り	1.6MHz	100 ksps
バイパス コンデンサを使う場合の $V_{DDA}/2$	6	1	有り	18MHz	1 Msps
V_{DDA}	7	X	無し	9 MHz	500 ksps

1.2 V の内部 V_{REF} 起動時間は、バイパスコンデンサのサイズによって異なります。Table 21-4 に、バイパスコンデンサの一般的な 2 つの値とその起動時間の仕様を示します。スキンの間にリファレンス電圧選択の変更、またはスリープ / ディープスリープ後のスキンの場合、SAR ADC がサンプリングを開始する時に、1.2V 内部 V_{REF} が安定することを確認してください。(V_{REF} が完全に放電される時) ワースト ケースのセトリング時間は起動時と同じです。

Table 21-4. バイパス コンデンサ静電容量値

内部 V_{REF} 立ち上がり時間	最大仕様
1 μ F 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	2 ms
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	200 μ s

21.3.3.3 入力範囲とリファレンス電圧

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 入力範囲は V_{REF} の選択で制限されます。反転入力 V_n で、ADC リファレンス電圧が V_{REF} である場合、非反転入力の電圧は $V_n \pm V_{REF}$ となります。この基準は反転入力と非反転入力 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。

21.3.4 SARSEQ

SARSEQ は専用シーケンサ コントローラであり、自動的に入力マルチプレクサをチャンネルからチャンネルまでスキャンし、結果をチャンネルごとのレジスタ アレイに格納します。

- CPU の介入なしに SARMUX アナログルーティングを自動制御
- SAR ADC コアを制御します (分解能、取得時間、リファレンスなど)
- SAR ADC からデータを受信し、前処理 (平均、範囲検出) を実行します
- ダブルバッファを使用して結果を保存するため、CPU は次のスキャンの進行中に最後のスキャンの結果を安全に取り出すことができます。

SARSEQ の特長を以下に示します。

- 16 本のチャンネルは CPU の介入なく個別に自動スキャンとして設定可能
- 不規則な信号を自動スキャンに挿入するための追加チャンネル (インジェクションチャンネル)
- 各チャンネルの持つ特長 :
 - シングルエンド入力または差動入力
 - 外部ピンからの入力 (シングルエンド モードの 8 つのチャンネルおよび差動モードでの 4 つのチャンネルのみ) または内部信号 (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
 - デフォルトの 12 ビット分解能、選択可能な予備分解能 : 8 ビットまたは 10 ビット
 - 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

- TCPWM ブロック、CTBm コンパレータ、低電力コンパレータ、SAR ADC 変換終了信号、およびファームウェアによってトリガーされます

- ハードウェアによる平均化に対応
 - 1 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ンネル飽和検出
 - 各チャン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
- 注入チャンネルの設定
 - 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

SAR_RANGE_COND ビットはチャンネル マスク可能範囲検出割込み (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)

詳細については、“範囲検出の割込み”(256 ページ)を参照してください。

21.3.4.3 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 の結果は、スキャンが完了するまで一連の作業レジスタに書き込まれます。スキャンが完了すると、データが 2 番目のレジスタセットにコピーされ、そこで user's アプリケーションでデータを読み取ることができます。この動作により、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。注入チャンネルを除き、すべての入力チャンネルは 16 のレジスタで二重バッファされます。注入チャンネルは通常のチャンネル 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。

21.3.4.4 注入チャンネル

注入チャンネルは通常のスキャンの一部ではないという点を除き、他のチャンネルに似ています。注入チャン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ンネルを有効にするとサンプル速度が変更されます。

注入チャン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。インジェクションチャンネルは連続またはトリガーに対応していません。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込みは不要です。

注入チャンネルの変換は SAR_INJ_CHAN_CONFIG レジスタをセットすることで通常チャン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

注入チャンネルはオーバーフロー割込みを除いて通常チャンネルと同じ割込みに対応しています。

- マスク可能変換の終了割込み INJ_EOC_INTR
- マスク可能範囲検出割込み INJ_RANGE_INTR
- マスク可能飽和検出割込み INJ_SATURATE_INTR
- マスク可能衝突割込み 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。

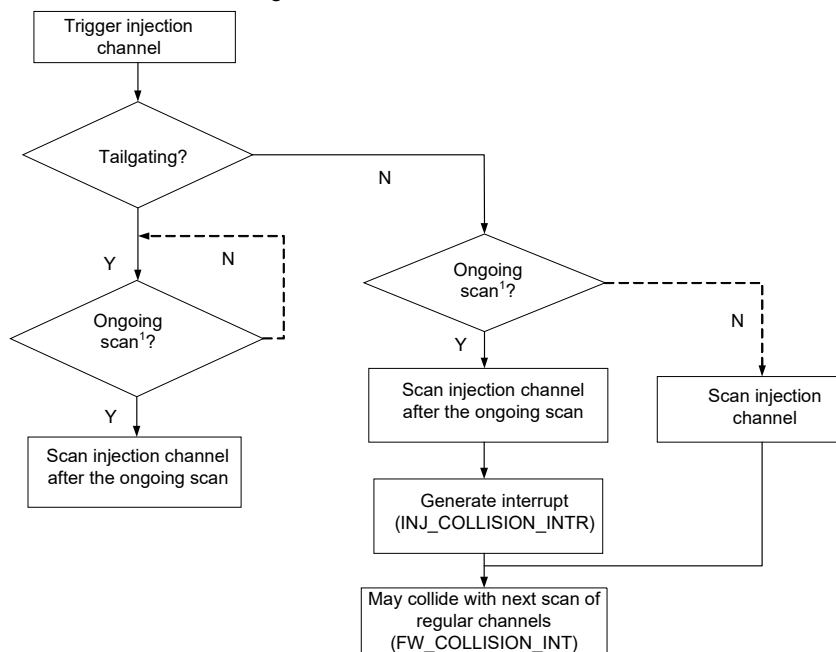
これらの機能については、“グローバル SARSEQ コンフィギュレーション”(261 ページ)、“チャンネルコンフィギュレーション”(261 ページ)、および“割込み”(255 ページ)で詳しく説明しています。

テールゲート

注入チャン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。INJ_TAILGATING = 1 (SAR_INJ_CHAN_CONFIG [30]) のセットによりテールゲートを選択することが推奨されます。注入チャンネルは進行中の通常チャン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 がアイドルのとき、テールゲートを選択すると、INJ_START_ENは次の通常チャンネル スキャンの終わりに注入チャンネルのスキャンを行います。

テールゲートを選択しない場合、実行中のスキャンがないか、または SAR ADC がアイドルであれば、INJ_START_EN ビットをセットするとインジェクションチャンネルの変換はすぐに開始されます。通常チャンネルのスキャンが実行中の場合、インジェクションチャンネルはその進行中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 1 つの問題は、次の通常チャンネルのスキャンがインジェクションチャンネルの変換と衝突する可能性があることです (FW_COLLISION_INTRがHIGHにセットされます)。その結果、次の通常チャンネルのスキャンはインジェクションスキャンが終わるまで延期されたため、通常のスキャンにジッタが発生します。

Figure 21-12. 注入チャンネルのフロー図



¹ scan here means scan of ALL the regular channels

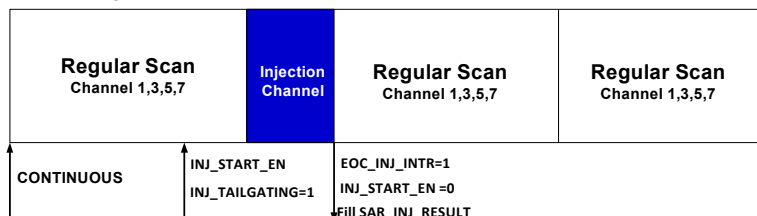
テールゲートの欠点は次のトリガーが発生するまで長時間かかる可能性があることです。通常チャンネルに衝突やジッタが発生する可能性がなければ、注入チャンネルはテールゲートを使わずに安全に使用できます。

注入チャン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。注入チャン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_COLLISION_INTR) のミラービットがあります。

Figure 21-13 は、連続スキャン中に注入チャンネルが有効で (チャンネル 1、3、5、および 7 が有効)、テールゲートが有効である場合の例です。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。

Figure 21-13. テールゲートが有効な注入チャンネル



21.3.5 割込み

割込みは以下の異なるイベントで生成することができます：

- スキャンの終了：すべての有効なチャンネルのスキャンが完了した時
- オーバーフロー：結果レジスタは前の結果が読み出される前に更新された時
- 衝突：SAR ADC が前のトリガーの処理中に新しいトリガーを受信した時
- インジェクション変換の終了：インジェクションチャンネルが変換された時
- 範囲検出：チャンネル結果が閾値と一致した時
- 飽和検出：チャンネル結果が設定した分解能の最小値または最大値と等しい時

この節は詳細に各割込みを説明します。これらの割込みは SAR_INTR_MASK レジスタ内に割込みマスクを持っています。割込みマスクを LOW に設定すると、対応する割込みソースが無視されます。割込みマスクのビットが HIGH であり、対応する割込みソースが保留中の場合では、SAR 割込みは生成されます。

割込みのサービスのとき、割込みサービスルーチン (ISR) はデータを読み出してから、割込みビットに「1」を書き込むことにより、割込みソースをクリアします。

SAR_INTR_MASKED レジスタは、割込みソースと割込みマスクの論理 AND です。このレジスタはファームウェアが割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_INTR_SET レジスタの EOS_SET など) は各割込みのトリガーに使用されます。この動作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込みを生成できます。

21.3.5.1 スキャン終了割込み (EOS_INTR)

スキャンの終了後、スキャン終了の割込み (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、GPIO に送信することもできます。EOS_INTR 信号は、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ンネル（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、割込みフラグおよび割込みマスクの論理 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。

21.3.5.2 オーバーフロー割込み

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、割込みフラグおよび割込みマスクの論理 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。

21.3.5.3 衝突割込み

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込みにより行われます。

衝突割込みは 3 つがあり、ファームウェア トリガー (FW_COLLISION_INTR) 用、外部トリガー (DSI_COLLISION_INTR) 用およびインジェクションチャンネル (INJ_COLLISION_INTR) 用です。これらの割込みは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。

外部トリガーがレベル 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、割込みフラグおよび割込みマスクの論理 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込みを設定することが可能です。

21.3.5.4 注入変換の終了の割り込み (INJ_EOC_INTR)

注入チャン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、この割込みをクリアします。

注入チャン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ンネルの変換も開始します。注入チャン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。

21.3.5.5 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設定します。読み出し時、このレジスタは割り込み要求レジスタを反映しています。

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)。

21.3.5.6 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。この機能は、サンプル値が特定の分解能の最大最小値に等しいことを検出し、対応するチャネルのマスク可能な割り込みフラグをセットします。この動作により 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_SATURATE_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設定します。読み出し時、このレジスタは割り込み要求レジスタを反映しています。

21.3.5.7 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。割り込みの要因の判定は 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)。

21.3.6 トリガー

スキャンのトリガーの 3 つの方法を以下に示します。

- 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。
- 外部トリガーは、TCPWM 出力、CTBm コンパレータ出力、低電力コンパレータ出力、および SAR ADC のサンプリング終了信号と変換終了信号です。ハードウェアトリガーを有効にするには、SAR_SAMPLE_CTRL レジスタの DSI_TRIGGER_EN ビットに '1' を書き込みます。トリガーの信号は、PSoC 4100S の PERI_TR_GROUP1_TR_OUT_CTL0 レジスタと PSoC 4100S Plus の PERI_TR_GROUP2_TR_OUT_CTL0 レジスタを使用して選択されます。

Table 21-5. PSoC 4100S のハードウェアトリガーソース選択

PERI_TR_GROUP1_TR_OUT_CTL0[6:0]	トリガーソース
0	0 にハードワイヤード (ファームウェアトリガー)
1	TCPWM 0 オーバーフロー
2	TCPWM 1 オーバーフロー
3	TCPWM 2 オーバーフロー
4	TCPWM 3 オーバーフロー
5	TCPWM 4 オーバーフロー
6	TCPWM 0 一致の比較
7	TCPWM 1 比較一致
8	TCPWM 2 比較一致
9	TCPWM 3 比較一致
10	TCPWM 4 比較一致
11	TCPWM 0 アンダーフロー

Table 21-5. PSoC 4100S のハードウェアトリガースソース選択

PERI_TR_GROUP1_ TR_OUT_CTL0[6:0]	トリガースソース
12	TCPWM 1 アンダーフロー
13	TCPWM 2 アンダーフロー
14	TCPWM 3 アンダーフロー
15	TCPWM 4 アンダーフロー
16	SAR ADC サンプル完了 (sdone) 信号
17	SAR ADC 変換終了 (eoc) 信号
18	CTBm コンパレータ 0 出力
19	CTBm コンパレータ 1 出力
20	LPCOMP 0 出力
21	LPCOMP 1 出力

Table 21-6. PSoC 4100S Plus のハードウェアトリガースソース選択

PERI_TR_GROUP2_ TR_OUT_CTL0[6:0]	トリガースソース
0	0 にハードワイヤード (ファームウェアトリガー)
1	TCPWM 0 オーバーフロー
2	TCPWM 1 オーバーフロー
3	TCPWM 2 オーバーフロー
4	TCPWM 3 オーバーフロー
5	TCPWM 4 オーバーフロー
6	TCPWM 5 オーバーフロー
7	TCPWM 6 オーバーフロー
8	TCPWM 7 オーバーフロー
9	TCPWM 0 一致の比較
10	TCPWM 1 比較一致
11	TCPWM 2 比較一致
12	TCPWM 3 比較一致
13	TCPWM 4 比較一致
14	TCPWM 5 比較一致
15	TCPWM 6 比較一致
16	TCPWM 7 比較一致
17	TCPWM 0 アンダーフロー
18	TCPWM 1 アンダーフロー
19	TCPWM 2 アンダーフロー
20	TCPWM 3 アンダーフロー
21	TCPWM 4 アンダーフロー
22	TCPWM 5 アンダーフロー
23	TCPWM 6 アンダーフロー
24	TCPWM 7 アンダーフロー
25	TCPWM 0 ライン出力

Table 21-6. PSoC 4100S Plus のハードウェアトリガースソース選択

PERI_TR_GROUP2_ TR_OUT_CTL0[6:0]	トリガースソース
26	TCPWM 1 ライン出力
27	TCPWM 2 ライン出力
28	TCPWM 3 ライン出力
29	TCPWM 4 ライン出力
30	TCPWM 5 ライン出力
31	TCPWM 6 ライン出力
32	TCPWM 7 ライン出力
33	SAR ADC サンプル完了 (sdone) 信号
34	SAR ADC 変換終了 (eoc) 信号
35	CTBm コンパレータ 0 出力
36	CTBm コンパレータ 1 出力
37	LPCOMP 0 出力
38	LPCOMP 1 出力

- 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ため、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。外部トリガーがファームウェアトリガーと同時に実行される場合、外部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 (衝突割込みがセットされます)。外部トリガーが連続トリガーと同時に実行される場合、両方は実質的に同時に処理されます (衝突トリガーが外部トリガーによりセットされることがあります)。

ファームウェア連続トリガーの場合では、シーケンサが 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 (シーケンサがアイドルの場合)。外部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です。

21.3.6.1 外部トリガー コンフィギュレーション

■ 同期化

着信外部トリガー信号は、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 (デフォルト) を用いてその信号を同期化する必要があります。ただしトリガー信号が AHB クロックと既に同期化された場合、この 2 つの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、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。

■ トリガー レベル

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。トリガーがレベル入力である場合、SAR はトリガー信号が HIGH である限り、新規のスキャンを開始します。トリガー信号がパルス入力である場合、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' が発生した後、SAR ADC がサンプリングを開始するように指示されるまでに送信時間がかかります。DSI_SYNC_TRIGGER と DSI_TRIGGER_LEVEL の構成が異なると、送信時間も異なります。Table 21-7 に最大時間を示します。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)、トリガーは無視されます。

Table 21-7. 外部トリガー最大時間

最大 External_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)
Pulse trigger: DSI_TRIGGER_LEVEL=0 (デフォルト)	1 clk_sys+2 clk_sar	3 clk_sys+2 clk_sar
Level Trigger: DSI_TRIGGER_LEVEL=1	2 clk_sar	2 clk_sys+2 clk_sar

Table 21-8. トリガー信号の要件

トリガー仕様	要件
Trigger Width (TW) (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。
Trigger interval (TI) (トリガー間隔 (TI))	トリガー間隔は、送信時間より長くする必要があります (Table 21-7 で指定)。それ以外の場合、2 番目のトリガーパルスは無視されます。

21.3.7 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 がインジェクションチャネルを示します)。SW_VREF_NEG ビットは、NEG を V_{REF} 入力と短絡させる現時点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 レジスタの現時点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。

21.3.8 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 ADC コア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。他の方法では、高精度を要しないチャネルで低い分解能を使用します。この動作で変換時間を 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)。さらに、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。

Table 21-9. 低消費電力用 ICONT_LV

ICONT_LV[1:0]	SAR ADC コアの相対消費電 [%]	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) [ksps]
0	100	18	4	1000
1	50	9	3	529
2	133	18	4	1000
3	25	4.5	2	281

SAR ADC コアの消費電力を制御することに加えて、VREF バッファ (使用する場合) によって消費される電力量も設定することができます。外部バイパスコンデンサを使用せずに VDDA の全範囲 (1.7V ~ 5.5V) で動作させるには、VREF バッファを 2 倍電力モードで動作させる必要があります。しかし、外付けバイパス コンデンサなしでサポートされる最大サンプル速度は 100ksps のままです。1Msps サンプル速度のために、外付けバイパス コンデンサおよび 18MHz クロックが必要です。詳細については、Table 21-10 を参照してください。

Table 21-10. SAR VREF 電源オプション

PWR_CTRL_VREF [1:0]	外付けバイパス コンデンサ要求	相対 VREF 電源 [%]	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]	VDDA 範囲
0	有り	100	18	4	1000	1.7 V - 5.5 V
0	無し	100	1.6	2	100	2.7 V - 5.5 V
2	無し	200	1.6	2	100	1.7 V - 5.5 V
1 または 3	無効な設定 - 使用すべきでない					

外部 VREF を使用することにより、VREF バッファとバイパス コンデンサは不要になり、SAR ADC ブロック全体で消費電力が削減されます。

21.3.9 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. シーケンサ / ファームウェアを用いて SARMUX アナログ配線 (ピン / 信号の選択) を設定
2. グローバルな SARSEQ 変換コンフィギュレーションを設定
3. 各チャネル ソース (ピン アドレスなど) を設定
4. チャネルを有効化
5. トリガー タイプを設定
6. 割込みマスクを設定
7. トリガー ソースを開始
8. 各変換割込みの終了後、データを取り込む
9. 必要に応じてインジェクション変換を実行

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。詳細なレジスタビット定義は、[PSoC 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) で利用可能です。

21.3.9.1 SARMUX アナログ配線

SARMUX アナログ配線を制御する方法は 2 つあり、シーケンサ制御およびファームウェア制御です。

シーケンサ制御

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。シーケンサ制御が無効になるように SWITCH_DISABLE を設定します。

シーケンサの制御では、チャン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、PIN_ADDR ビットは SAR_CHANx_CONFIG [2:0] です。Table 21-11 は、PORT_ADDR および PIN_ADDR のセットアップと、対応する SARMUX の選択を示しています。未使用のポート / ピンは、PSoC4 シリーズの他の製品用に予約されています。

Table 21-11. PORT_ADDR および 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。詳細については、[連続時間ミニブロック \(CTBm\) \(273 ページ\)](#) の章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 の非反転端子に接続します。差動モードが有効で、PORT_ADDR = 1 の場合 sarbus1 は SAR ADC の反転端子にのみ接続できます。

差動変換のために、反転端子の接続は、PORT_ADDR と PIN_ADDR により定義される非反転端子の接続に依存します。DIFFERENTIAL_EN の設定により、チャンネルは無視された 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。P.0/P.1、P.2/P.3、P.4/P.5、P.6/P.7 はシーケンサ制御のために有効な差動ペアです。ファームウェアに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]) はどちらの信号を反転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。反転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。詳細については、[“反転入力選択” \(241 ページ\)](#) 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、反転入力を 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 (正) および VMINUS (負) 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_CTRL[n] = 0)。またはハードウェア制御方式(シーケンサ)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 の一端子 (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 は反転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。反転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。詳細については、“反転入力選択”(241 ページ)を参照してください。

21.3.9.2 グローバル SARSEQ コンフィギュレーション

すべてのチャン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、チャンネル コンフィギュレーションにはグローバル 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。

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、結果は未定義です。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。

Table 21-12. グローバル コンフィギュレーション レジスタ

設定	制御レジスタ	参考文献
Reference selection (リファレンスの選択)	SAR_CTRL[6:4]	21.3.3.1 リファレンス電圧のオプション
Signed/unsigned selection (符号付き / 符号無し の選択)	SAR_SAMPLE_CTRL [3:2]	21.3.1.3 結果データのフォーマット
Data left/right alignment (データの左 / 右アライ メント)	SAR_SAMPLE_CTRL [1]	21.3.1.3 結果データのフォーマット
シングルエンド モードでの反転入力の選択	SAR_CTRL[11:9]	21.3.1.4 反転入力選択
Resolution (分解能)	SAR_SAMPLE_CTRL[0] ^a	21.3.1.5 分解能
Acquisition time (取得時間)	SAR_SAMPLE_TIME01 [25:0] SAR_SAMPLE_TIME32 [25:0]	21.3.1.6 取得時間
Averaging count (平均化回数)	SAR_SAMPLE_CTRL[7:4]	21.3.4.1 平均化
Range detection (範囲検出)	SAR_RANGE_THRES [31:0] SAR_RANGE_COND [31:30]	21.3.4.2 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。

21.3.9.3 チャンネル コンフィギュレーション

チャンネル コンフィギュレーションは以下のものを含まれます：

- 差動またはシングルエンドのモード選択
- 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、チャンネル 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)。ただしチャンネルが進行中のスキャン用に有効にされない場合、そのチャン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ンネルはこのルールのたった 1 つの例外です。有効にされたチャン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。有効にされたチャンネルを変更すると、サンプル レートが変更されることがあります。

Table 21-13. チャンネル コンフィギュレーション レジスタ

設定	レジスタ	参考文献
Single-ended/differential (シングルエンド / 差動)	SAR_CHANx_CONFIG [8]	21.3.1.1 シングルエンドと差動モード
Acquisition time selection (取得時間選択)	SAR_CHANx_CONFIG [13:12]	21.3.1.6 取得時間
Resolution selection (分解能選択)	SAR_CHANx_CONFIG [9]	21.3.1.5 分解能
Average enable (平均化イネーブル)	SAR_CHANx_CONFIG [10]	21.3.4.1 平均化

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。

Table 21-14. 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]	チャンネル分解能
OFF	0	1	8 ビット
OFF	1	1	10 ビット
OFF	0	0	12 ビット
OFF	1	0	12 ビット
ON	X	X	12 ビット

21.3.9.4 チャンネル イネーブル

CHAN_EN レジスタは各チャン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ンネルはスキャンされます。トリガーの後、チャンネル 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。この動作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ンネル)はスキャンが進行中に更新しないようにします。

21.3.9.5 割込みマスク

割込みソースは 6 つあります。すべての割込みソースには割込みマスクがあります：

- スキャン終了の割込み
- オーバーフロー割り込み
- 衝突割り込み
- インジェクション変換終了の割込み
- 範囲検出の割込み
- 飽和検出の割込み

各割込みには、割込み要求レジスタ (INTR、SATURATE_INTR、RANGE_INTR)、ソフトウェア割込みセット レジスタ (INTR_SET、SATURATE_INTR_SET、RANGE_INTR_SET)、割込みマスク レジスタ (INTR_MASK、SATURATE_INTR_MASK、RANGE_INTR_MASK)、および割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。割込み要因レジスタは、現在保留中の SAR 割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込み要因を判定します。

詳細については、[21.3.5 割込み](#)を参照してください。

21.3.9.6 トリガー

A/D 変換を開始する 3 つの方法を以下に示します。

- ファームウェア トリガー：SAR_START_CTRL [0]

- 外部トリガー：ds_i_trigger
- 連続トリガー：SAR_SAMPLE_CTRL [16]

詳細については、[21.3.6 トリガー](#)を参照してください。

21.3.9.7 各割込みの終了後、データを取り込みます

各スキャン後に必ず結果レジスタからデータを読み取ってください。そうしないと、次の scan's 構成のためにデータが変更される可能性があります。

16 ビットのデータ レジスタは、最大 8 チャンネルまで二重バッファの実装に使用されます(注入チャンネルには二重バッファがありません)。二重バッファは、各チャンネルごとに 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。データはこのチャン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ン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込みビットは、raw (マスクされない) 割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。

21.3.9.8 注入変換

INJ_START_EN (INJ_CHAN_CONFIG [31]) の開始ビットの設定により、注入チャン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、定期的なチャンネルの次のスキンの最後にスキャンされるインジェクションチャンネルを有効化します。詳細については、[21.3.4.4 注入チャンネル](#)を参照してください。

21.3.10 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。リファレンスは内部 1.2V からのもので、

ピンまたは信号は、3 つの方法で SAR ADC にルーティングできます。[Table 21-15](#) に、温度センサーを SAR ADC にルーティング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。

Table 21-15. 温度センサーと SAR ADC の接続

制御方法	設定
Sequencer (シーケンサ)	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
Firmware (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

a. 温度センサーでは NEL_SEG (SAR_CTRL [11:9]) を「0」で上書きします。

21.4 レジスタ

名称	オフセット	数量	Width	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	チャンネルのイネーブル 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	チャンネル コンフィギュレーション レジスタ
SAR_CHAN_WORK	0x0100	8	32	チャンネル ワーク データ レジスタ
SAR_CHAN_RESULT	0x0180	8	32	チャンネル結果データ レジスタ
SAR_CHAN_WORK_VALID	0x0200	1	32	チャンネル ワークデータ レジスタの有効ビット
SAR_CHAN_RESULT_VALID	0x0204	1	32	チャンネル結果データ 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	割込み要求レジスタ
SAR_INTR_SET	0x0214	1	32	割込み要求セット レジスタ
SAR_INTR_MASK	0x0218	1	32	割込みマスク レジスタ
SAR_INTR_MASKED	0x021C	1	32	割込み要求マスク レジスタ: 値が 0 以外の場合 NVIC への SAR 割込み信号は HIGH。読み出すとレジスタは割込み要求とマスク レジスタの間のビット単位の AND を反映
SAR_SATURATE_INTR	0x0220	1	32	飽和割込み要求レジスタ
SAR_SATURATE_INTR_SET	0x0224	1	32	飽和割込み要求セット レジスタ
SAR_SATURATE_INTR_MASK	0x0228	1	32	飽和割込みマスク レジスタ
SAR_SATURATE_INTR_MASKED	0x022C	1	32	飽和割込み要求マスク レジスタ
SAR_RANGE_INTR	0x0230	1	32	範囲検出割込み要求レジスタ
SAR_RANGE_INTR_SET	0x0234	1	32	範囲検出割込み要求セット レジスタ
SAR_RANGE_INTR_MASK	0x0238	1	32	範囲検出割込みマスク レジスタ
SAR_RANGE_INTR_MASKED	0x023C	1	32	範囲割込み要求マスク レジスタ
SASR_INTR_CAUSE	0x0240	1	32	割込み原因レジスタ
SAR_INJ_CHAN_CONFIG	0x0280	1	32	注入チャンネル コンフィギュレーション レジスタ
SAR_INJ_RESULT	0x0290	1	32	注入チャンネル結果レジスタ
SAR_MUX_SWITCH0	0x0300	1	32	SARMUX ファームウェア スイッチ制御

名称	オフセット	数量	Width	説明
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

22. 低消費電力コンパレータ



PSoC[®] 4 デバイスは 2 個の低消費電力コンパレータがあります。これらのコンパレータはすべての消費電力モードで高速アナログ信号の比較を可能にします。さまざまなデバイスの電源モードの詳細については、[電力モード \(107 ページ\)](#) を参照してください。正と負の入力は専用の GPIO 端子または AMUXBUS-A/AMUXBUS-B に接続することができます。コンパレータ出力は、ステータス レジスタにより CPU によって読み出され、割込みまたは復帰ソースとして使用、または GPIO に配線されます。

22.1 特長

PSoC 4 コンパレータは次の特長があります：

- コンフィギュレーション可能な正と負の入力
- プログラム可能な消費電力および動作速度
- 超低電力モードのサポート (< 4 μ A)
- オブションの 10mV 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)
- ディープスリープ モードでの復帰ソース

22.2 ブロック図

[Figure 22-1](#) に、低電力コンパレータのブロック図を示します。

The diagram illustrates the internal structure of the Low Power Comparator (LPC) and its connections to various system components. It is divided into three main power domains: Active Power Domain, DeepSleep Power Domain, and another Active Power Domain at the bottom.

- Top Active Power Domain:** Contains the **AHB IF** (Arbitrated Host Bus Interface) and **MMIO Registers**. The AHB IF is connected to the AHB bus. The MMIO Registers are connected to the AHB IF and receive **Sync** signals from the comparators.
- DeepSleep Power Domain:** Contains the core comparator logic. It includes two comparators, **Comparator 0** and **Comparator 1**, each with an **Edge Detector**. The comparators are connected to I/O pads **P0.0**, **P0.1**, **P0.2**, and **P0.3**. The edge detectors output **Falling, rising, both** and **Intr_clr** signals. The **Intr_clr** signals are connected to the **Interrupt Generation** block. The **Interrupt Generation** block outputs **comp_intr** to the **MMIO Registers** and **lpcomp_comp[0]** to the **HSIOM** (Host System Interface Over Memory).
- Bottom Active Power Domain:** Contains the **Sync** block, which receives **Sync** signals from the comparators and outputs **<To MMIO Registers>**.
- External Connections:** The LPC is connected to the **AMUXBUS_A** and **AMUXBUS_B** multiplexers. A note indicates that the low power comparator is not part of the low power block and is in the GPIO block, with each GPIO connecting to **AMUXBUS_A/_B**.

Legend:

- Red line: MMIO interface signals
- Black line: Comparator related signals
- Green line: Signal connection to HSIOM

22.3 動作原理

以下のセクションでは、入力構成、電力と速度モード、出力と割り込みの構成、ヒステリシス、低電力モードからのウェイクアップ、コンパレータクロック、オフセットトリムなど、PSoC4 低電力コンパレータの動作について説明します。

22.3.1 入力コンフィギュレーション

コンパレータへの入力を以下に示します。

- 専用入力端子から 2 つの正と負の入力
- AMUXBUS を介し、任意の端子から 2 つの正と負の入力 (ディープスリープモードでは使用不可)
- 外部端子からの入力と、内部で生成された信号を利用する入力。外部信号と内部信号は、コンパレータの正入力と負入力、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、コンパレータの入力に接続される。
- 内部で生成された信号を正と負に入力。内部で生成された信号は AMUXBUS-A/AMUXBUS-B を介して、コンパレータの入力に接続される。

Figure 22-1 から、P0.0 と P0.1 がコンパレータ 0 の正と負の入力に接続されていることに注意してください。P0.2 と P0.3 はコンパレータ 1 の入力に接続します。また、AMUXBUS ネットはコンパレータ入力に直接接続されていないことに注意してください。このようにしてコンパレータの接続は、対応する入力ピンを介して AMUXBUS ネットに配線されます。コンパレータの接続に AMUXBUS を使用する場合は、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。コンパレータ入力の接続に AMUXBUS を使用する設計では、それらをオープンの状態にすべきです。AMUXBUS 接続はディープスリープモードで使用できないことに注意してください。もし、ディープスリープ動作で必要な場合、低消費電力コンパレータは専用端子で接続する必要があります。この制限事項は、接続に AMUXBUS を使用し、内部で生成されたいずれの信号配線も含みます。GPIO を AMUXBUS A/B に接続する方法、またはコンパレータ入力用に GPIO を設定する方法の詳細については、[I/O システム \(66 ページ\)](#) を参照してください。

22.3.2 出力および割り込みコンフィギュレーション

Comparator0 および Comparator1 の出力は、LPCOMP_CONFIG レジスタ (Table 22-1) の OUT1 ビット [6] および OUT2 ビット [14] でそれぞれ利用できます。コンパレータ出力は、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。各コンパレータの入力是对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。表 221 に示

すように、INTTYPEx ビットを使用して、割り込みタイプを無効、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に選択できます。

各 comparator's の出力は、HSIOM を介して GPIO ピンに直接ルーティングできます。コンパレータ出力は、HSIOM のディープスリープソース2接続として使用できます。HSIOM の詳細については、[“高速 I/O マトリックス” \(71 ページ\)](#) を参照してください。低電力コンパレータ出力をサポートするピンの詳細については、[PSoC 4100S datasheet](#) および [PSoC 4100S Plus datasheet](#) を参照してください。これらの端子での出力はコンパレータから直接に出力され、同期化されません。これらは端子でディープスリープソースとして動作するためであり、コンパレータ出力をディープスリープ電力モードで利用するためです。

エッジイベント中に、コンパレータが割り込みをトリガーします (Figure 22-1 の intr_comp1/intr_comp2 信号)。割り込み要求は、Comparator0 と Comparator1 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。Comparator0 と Comparator1 は両方とも、2 つの割り込みの論理 OR である共通の割り込み (Figure 22-1 の comp_intr 信号) を共有し、CPU NVIC の低電力コンパレータ block's 割り込みとしてマッピングされます。詳細については、[割り込み \(51 ページ\)](#) を参照してください。両方のコンパレータが使用される場合、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 Comparator0 と Comparator1 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、ファームウェアの LPCOMP_INTR レジスタの COMP1 および COMP2 ビットに '1' を書き込むことにより、割り込みをクリアする必要があります。割り込みがクリアされない次の比較イベントは割り込みをトリガーせず、CPU はイベントを処理することができなくなります。

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

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

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

Table 22-1. LPCOMP_CONFIG レジスタの出力と割込みコンフィギュレーション

レジスタ [ビット位置]	ビット名 (Bit_Name)	説明
LPCOMP_CONFIG[6]	OUT1	コンパレータ 0 の現在の出力値
LPCOMP_CONFIG[14]	OUT2	コンパレータ 1 の現在の出力値
LPCOMP_CONFIG[5:4]	INTTYPE1	コンパレータ 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 1 が IRQ をトリガーするエッジを設定 00: 無効 01: 立ち上がりエッジ 10: 立ち下がりエッジ 11: 立ち上がりと立下りエッジ
LPCOMP_INTR[0]	COMP1	コンパレータ 0 の割り込み: コンパレータ 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。割り込みをクリアするために '1' を書き込みます
LPCOMP_INTR[1]	COMP2	コンパレータ 2 の割り込み: コンパレータ 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。割り込みをクリアするために '1' を書き込みます
LPCOMP_INTR_SET[0]	COMP1	'1' を書き込んで、Comparator0 のソフトウェア割り込みをトリガーします
LPCOMP_INTR_SET[1]	COMP2	「1」の書き込みにより、コンパレータ 1 のソフトウェア割り込みをトリガー

22.3.3 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
- 超低電力

コンパレータ 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 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。さまざまな電力設定の応答時間と消費電力の仕様については、[PSoC 4100S datasheet](#) および [PSoC 4100S Plus datasheet](#) を参照してください。

Table 22-2 で説明されているように、LPCOMP_CONFIG レジスタの ENABLE1 ビット [7] および ENABLE2 ビット [15] を使用して、コンパレータを有効 / 無効にします。

注：コンパレータが有効になっているときに電力モードが変更されると、コンパレータの出力にグリッチが発生する場合があります。それを防止するため、電力モードを変更する前に、コンパレータを無効にします。

Table 22-2. コンパレータ動作モードの選択ビット

レジスタ [ビット位置]	ビット名 (Bit_Name)	説明
LPCOMP_CONFIG[1:0]	MODE1	コンパレータ 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 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 0 の有効ビット 0: コンパレータ 0 が無効 1: コンパレータ 0 が有効
LPCOMP_CONFIG[15]	ENABLE2	コンパレータ 1 の有効ビット 0: コンパレータ 1 が無効 1: コンパレータ 1 が有効

22.3.4 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 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。

Table に示すように、10 mV のヒステリシスレベルは、LPCOMP_CONFIG レジスタの HYST1 ビット [2] および HYST2 ビット [10] を使用して有効 / 無効にします。

Table 22-3. ヒステリシス制御ビット HYST1 と HYST2

レジスタ [ビット位置]	ビット名 (Bit_Name)	説明
LPCOMP_CONFIG[2]	HYST1	コンパレータ 0 への 10mV のヒステリシスを有効 / 無効にします -0: ヒステリシスを有効にする -1: ヒステリシスを無効にする
LPCOMP_CONFIG[10]	HYST2	コンパレータ 1 への 10mV のヒステリシスを有効 / 無効にします -0: ヒステリシスを有効にする -1: ヒステリシスを無効にする

22.3.5 低消費電力モードからの復帰

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

ディープスリープ電力モードでは、コンパレータ 0 またはコンパレータ 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 INTTYPE_x ビットを、必要に応じて低消費電力モードからデバイスを復帰させコンパレータについて設定します。LPCOMP_INTR_MASK レジスタのマスクビットは、comparator's の割り込みの 1 つまたは両方を CPU が処理するかどうかを選択するために使用されます。

22.3.6 コンパレータ クロック

コンパレータはシステムの主クロックである SYSCLK を割り込み同期化用のクロックとして使用します。

22.3.7 オフセット調整

コンパレータ オフセットは工場出荷時に 4.0mV 以内に調整されます。トリミングは 2 ステップのプロセスで、最初に 0.1V に等しいコモンモード電圧でトリミングされ、次に $V_{DD}-0.1V$ に等しいコモンモード電圧でトリミングされます。オフセット電圧は、0.1V ~ $V_{DD}-0.1V$ 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高精度な調整が必要とされる場合、その入力同相モード電圧で調整を行ってください。コンパレータ オフセット調整は、LPCOMP_TRIM1/2/3/4 のレジスタを使用して行います。LPCOMP_TRIM1 と LPCOMP_TRIM2 は、コンパレータ 0 をトリミングするために使用されます。LPCOMP_TRIM3 および LPCOMP_TRIM4 は、コンパレータ 1 をトリミングするために使用されます。トリム値を変更するビットフィールドは、LPCOMP_TRIM1 および LPCOMP_TRIM3 の TRIMA ビット [4:0] と、LPCOMP_TRIM2 および 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 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お

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. コンパレータを設定し、ヒステリシスをオフにし、出力を確認します。
3. 出力が High の場合、オフセットは正です。出力が HIGH ではないと、オフセットは負です。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] はオフセットの量を制御し、TRIMA ビット [4] はオフセットの極性を制御します ('1' は正のオフセットを示し、'0' は負のオフセットを示します)。
 - b. TRIMA ビット調整の完了後、引き続き、TRIMB ビット [3:0] を出力が反転するまで調整します。TRIMB ビットの調整はコンパレータ動作が低速モードの場合にのみ有効です。TRIMB ビット [3] はオフセットの極性を制御します。TRIMB ビット [2:0] はオフセット量です。
 - c. 3-b のステップが完了した後、TRIMA と TRIMB ビットの値は、その特定の V_{REF} での最良の調整値になります。

22.4 レジスタの概要

Table 22-4. 低消費電力コンパレータ レジスタ要約

レジスタ	機能
LPCOMP_ID	LPCOMP コントローラ 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 割込みレジスタ
LPCOMP_INTR_SET	LPCOMP 割込みセット レジスタ
LPCOMP_INTR_MASK	LPCOMP 割込み要求マスク レジスタ
LPCOMP_INTR_MASKED	LPCOMP マスクされた割込み出力レジスタ
LPCOMP_TRIM1	コンパレータ 0 の調整領域
LPCOMP_TRIM2	コンパレータ 0 の調整領域
LPCOMP_TRIM3	コンパレータ 1 の調整領域
LPCOMP_TRIM4	コンパレータ 1 の調整領域

23. 連続時間ミニブロック (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、コンパレータに設定可能な 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 インターフェース、スイッチ制御および割込みが含まれます。PSoC 4100S および PSoC 4100S Plus デバイスには、ディープスリープ電源モードで動作可能な 1 つの CTBm ブロックがあります。

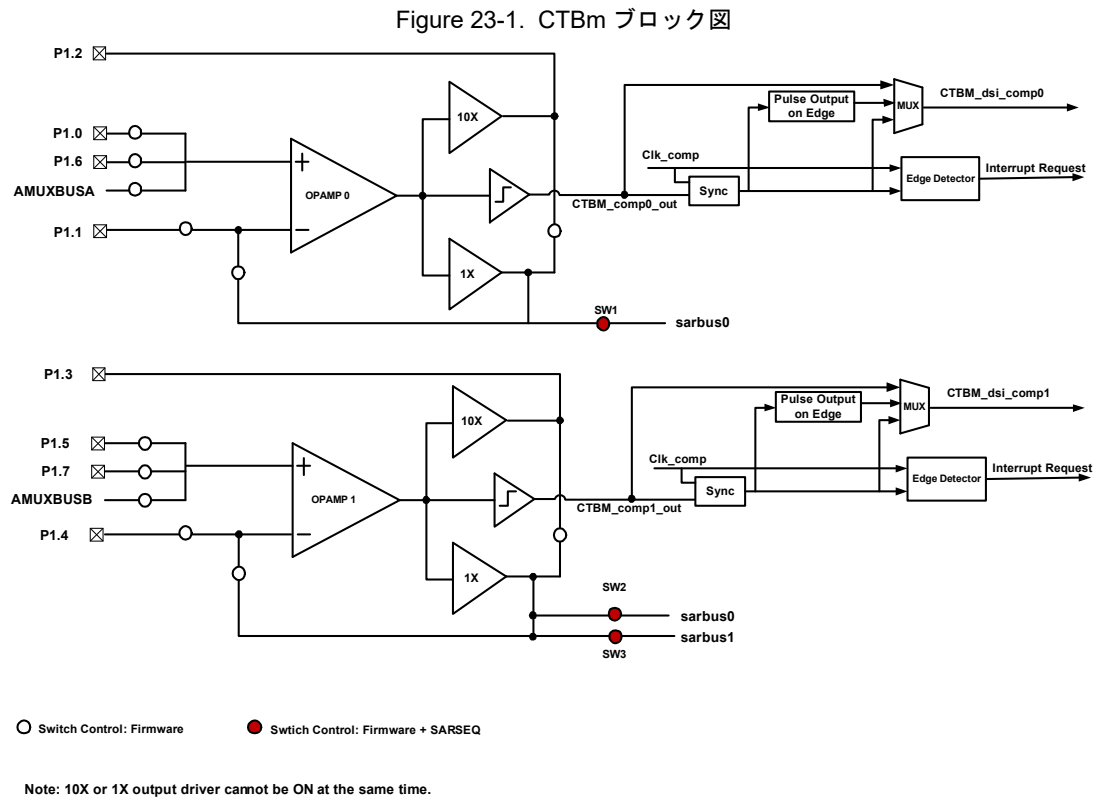
23.1 特長

PSoC4 CTBm ブロック内のオペアンプは以下の特長があります。

- 高性能かつ高度に設定可能な独立したオンチップ アンプ
- プログラム可能な電力モード、帯域幅、位相補正、出力駆動強度
- 1mA または 10mA を選択可能な出力電流駆動能力
- 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ワー 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
- ディープスリープ デバイス電力モードでも動作する

23.2 ブロック図

Figure 23-1 は、PSoC 4 デバイスで利用可能な CTBm ブロックのブロック図を示しています。



23.3 動作原理

ブロック図が示すように、CTBm には 2 つの同一のオペアンプがあります。Figure 23-1 に示すように、各オペアンプには 1 つの入力ステージと 3 つの出力ステージがあり、すべてが共通の入力ステージを共有しています。1 度を選択できるのはそのうちの 1 つだけです。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。他の設定可能な機能は電力モードと速度、位相補正およびスイッチ 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、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。Opamp0 に CTBM_OA_RES0_CTRL [11] ビットをセットし、Opamp1 に CTBM_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設定してください：

1. 電力モードを設定
2. 出力駆動強度を設定
3. 位相補正を設定
4. 入力スイッチを設定
5. 出力スイッチを設定 (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. 電力モードを設定
2. 入力スイッチを設定
3. 必要に応じて、コンパレータの出力回路を構成します - 割り込みの生成、出力など

4. ヒステリシスを設定し、コンパレータを有効にする

23.3.1 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 CTBM_OA_RESx_CTRL 内の PWR_MODE ビット [1:0] を使用して設定されます。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。詳細については、[Table 23-1](#) を参照してください。さまざまな電力モードでのゲイン帯域幅、スルーレート、 I_{OUT} 仕様については、[PSoC 4100S datasheet](#) および [PSoC 4100S Plus datasheet](#) を参照してください。

23.3.2 出力駆動強度コンフィギュレーション

各オペアンプの出力ドライバは、内部ドライバ (クラス A/1X ドライバ) または外部ドライバ (クラス AB/10X ドライバ) に構成できます。1X ドライバと 10X ドライバは相互に排他的です。これらを同時にアクティブにすることはできません。1X 出力ドライバは、小型のオンチップ容量性および抵抗性負荷を高速で駆動するのに適しています。10X の出力ドライバ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 ドライバ出力は外部ピンに接続されます。[Table 23-1](#) に示すように、各ドライバモードには低、中、または高電力モードがあります。

Table 23-1. 出力ドライバと電力モード

電力モード / I_{OUT} 駆動能力	CTBM_OA_RESx_CTRL[1:0]			
	00 (無効)	01 (低)	10 (中)	11 (大)
External Driver (10X) (外部ドライバ (10X))	オフ	10 mA	10 mA	10 mA
Internal Driver (1X) (内部ドライバ (1X))	オフ	100 μ A	400 μ A	1 mA

10X と 1X の出力駆動能力 (0: 1X、1: 10X) を選択するために CTBM_OA_RESx_CTRL[2] ビットが使用されます。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 出力ドライバを選択します。特殊な場合では出力を 1X 出力ドライバで外部ピンに、または 10X 出力ドライバで内部負荷 (例えば、SAR ADC) に接続するために、CTBM_OAx_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。

[Table 23-2](#) に、オペアンプ出力のドライブ強度と電力モードの構成に使用されるビットをまとめました。

Table 23-2. CTBM レジスタにおける出力駆動強度と電力モード設定

レジスタ [ビット位置]	ビット名 (Bit_Name)	説明
CTBM_CTB_CTRL[31]	ENABLE	CTBM の電力モードの選択 0: CTBM が無効 1: CTBM が有効
CTBM_OA_RES0_CTRL [11]	OA0_PUMP_EN	Opamp0 ポンプイネーブルビット 0: Opamp0 ポンプは無効です 1: Opamp0 ポンプが有効
CTBM_OA_RES1_CTRL [11]	OA1_PUMP_EN	Opamp1 ポンプイネーブルビット 0: Opamp1 ポンプが無効 1: Opamp0 ポンプが有効

Table 23-2. CTBM レジスタにおける出力駆動強度と電力モード設定 <Italic> (continued)

レジスタ [ビット位置]	ビット名 (Bit_Name)	説明
CTBM_OA_RES0_CTRL [1:0]	OA0_PWR_MODE	Opamp0 電力モード選択ビット 00: Opamp0 はオフ 01: Opamp0 は低消費電力モードにある 10: Opamp0 は中消費電力モードにある 11: Opamp0 は高消費電力モードにある
CTBM_OA_RES1_CTRL [1:0]	OA1_PWR_MODE	Opamp1 電力モード選択ビット 00: Opamp1 はオフ 01: Opamp1 は低消費電力モードにある 10: Opamp1 は中消費電力モードにある 11: Opamp1 は高消費電力モードにある
CTBM_OA_RES0_CTRL [2]	OA0_DRIVE_STR_SEL	Opamp0 出力駆動強度選択ビット 0: Opamp0 出力駆動強度は 1X 1: Opamp0 出力駆動強度は 10X
CTBM_OA_RES1_CTRL [2]	OA1_DRIVE_STR_SEL	Opamp1 出力駆動強度選択ビット 0: Opamp1 出力駆動強度は 1X 1: Opamp1 出力駆動強度は 10X

23.3.3 位相補正

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正コンデンサブロックを持ちます。各オペアンプの補償は、Table 23-3 で説明されているように、それぞれの CTBM_OAx_COMP_TRIM レジスタによって制御されます。PSoC 4100S datasheet および PSoC 4100S Plus datasheet のすべての GBW スルーレート仕様は、すべての補償トリムに適用されることに注意してください。

Table 23-3. CTBm 内の Opampx (Opamp0 または Opamp1) 位相補正ビット

レジスタ [ビット位置]	ビット名 (Bit_Name)	説明
CTBM_OAx_COMP_TRIM[1:0]	OAx_COMP_TRIM	Opampx 位相補正トリム ビット 00: 位相補正なし 01: 最小補償、高速、低安定性 10: 位相補正中、バランス速度、バランス安定性 11: 位相補正大、低速、高い安定性

23.3.4 スイッチ制御

CTBm はオペアンプ入出力を設定するための多くのスイッチを持っています。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 (CTBM_OA0_SW、CTBM_OA1_SW) を設定することで制御されます。その 3 個のスイッチは SAR ADC のレジスタおよび CTBm レジスタ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 CTBM_OAx_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。スイッチを開くには、CTBM_OAx_SW_CLEAR に「1」を書き込んで、CTBM_OAx_SW の対応するビットをクリアします。スイッチとスイッチが有効にする接続の詳細については、[PSoC 4100S: PSoC 4 Registers TRM](#) および [PSoC 4100S Plus: PSoC 4 Registers TRM](#) を参照してください。

23.3.4.1 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 AMUXBUS-A に接続し、Opamp1 は AMUXBUS-B に接続します。

注： 正と負の入力バスでは、1 つのスイッチのみを閉じる必要があります。そうしないと、異なる入力ソースが互いに短絡する可能性があります。

- 非反転入力: 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 本の AMUXBUS ラインから入力を選択できます。詳細については、[Table 23-4](#) を参照してください。

Table 23-4. 非反転入力選択

	正の入力	スイッチ制御ビット	説明
Opamp0	AMUXBUS A	CTBM_OA0_SW [0]	0: スイッチを開く ; 1: スイッチを閉じる
	P1.0	CTBM_OA0_SW [2]	0: スイッチを開く ; 1: スイッチを閉じる
	P1.6	CTBM_OA0_SW [3]	0: スイッチを開く ; 1: スイッチを閉じる
Opamp1	AMUXBUS B	CTBM_OA1_SW [0]	0: スイッチを開く ; 1: スイッチを閉じる
	P1.5	CTBM_OA1_SW [1]	0: スイッチを開く ; 1: スイッチを閉じる
	P1.7	CTBM_OA1_SW [4]	0: スイッチを開く ; 1: スイッチを閉じる

- 負の入力 : opamp0 と opamp1 の両方には、アナログスイッチを介した 2 つの負の入力オプションがあります。1 つは外部ピンまたは CTBM_OAx_SW レジスタによって制御される出力フィードバックです。[Table 23-5](#) に制御ビットを示します。

Table 23-5. 反転入力選択

	反転入力	スイッチ制御ビット	説明
Opamp0	P1.1	CTBM_OA0_SW [8]	0: スイッチを開く ; 1: スイッチを閉じる
	1X 出力ドライバによる Opamp0 出力フィードバック	CTBM_OA0_SW [14]	0: スイッチを開く ; 1: スイッチを閉じる
Opamp1	P1.4	CTBM_OA1_SW [8]	0: スイッチを開く ; 1: スイッチを閉じる
	1X 出力ドライバによる Opamp1 出力フィードバック	CTBM_OA1_SW [14]	0: スイッチを開く ; 1: スイッチを閉じる

23.3.4.2 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。opamp0 出力は sarbus0 に接続でき、opamp1 は sarbus0 または sarbus1 に接続できます。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ためのものです。sarbus への3つの出力ルーティングスイッチは SAR ADC レジスタおよび CTBm レジスタ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。

次の真理値表 (Table 23-6、Table 23-7、および Table 23-8) は、3つのスイッチの制御ロジックを示しています。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 レジスタを使用する際、CTBM_SW_HW_CTRL のビット [2] または [3] を設定する必要があります。CTBM_OAx_SW[18]/[19] は他の制御ビットをマスクすることができます：CTBM_OAx_SW[18]/[19] が 0 に設定されると、SW[n] が「0」に設定されます。

CTBM_SW_STATUS [30:28] レジスタは、スイッチ SW1/2/3 の現時点のステータスを示します。

Table 23-6. SW1 制御論理の真理値表

PORT_ADDR	PIN_ADDR	CTBM_SW_HW_CTRL[2]	CTBM_OA0_SW[18]	SW1
X	X	X	0	0
X	0	1	1	0
0	X	1	1	0
X	X	X	1	1
X	X	0	1	1
1	2	X	1	1

Table 23-7. SW2 制御論理の真理値表

DIFFERENTIAL_EN	PORT_ADDR	PIN_ADDR	CTBM_SW_HW_CTRL[3]	CTBM_OA0_SW[18]	SW2
X	X	X	X	0	0
X	X	0	1	1	0
X	0	X	1	1	0
1	X	X	X	1	0
X	X	X	0	1	1
X	X	X	X	1	1
0	1	3	X	1	1

Table 23-8. SW3 制御論理の真理値表

DIFFERENTIAL_EN	PORT_ADDR	PIN_ADDR	CTBM_SW_HW_CTRL[3]	CTBM_OA0_SW[18]	SW3
X	X	X	X	0	0
X	X	0	1	1	0
X	0	X	1	1	0
0	X	X	X	1	0
X	X	X	0	1	1
X	X	X	X	1	1
1	1	2	X	1	1

23.3.4.3 コンパレータ モード

各オペアンプは、その CTBM_OA_RESx_CTRL[4] ビットをセットすることで、コンパレータに設定することができます。コンパレータを有効にすると、補正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特長を次に示します。

- オプションの 10mV 入力ヒステリシス
- 設定可能な電力モード / 速度
- オプションのコンパレータ出力同期化
- 1mV 以下に調整されたオフセット
- 設定可能なエッジ検出 (立ち上がり / 立ち下がり / 両方 / ディセーブル)

23.3.4.4 コンパレータ 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、一方向 (低から高) で有効にできます。入力ヒステリシスは CTBM_OA_RESx_CTRL[5] を設定することで有効にすることができます。2 個のコンパレータは、CTBM_OA_RESx_CTRL[1:0] を設定することで制御される低、中および高の 3 つの電力モードもあります。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。

システム AHB クロックでコンパレータ出力の同期化は CTBM_OA_RESx_CTRL[6] で設定することができます。

Comparator0 と Comparator1 の出力状態は、それぞれ CTBM_COMP_STAT[0] と CTBM_COMP_STAT[16] に格納されます。

Table 23-9 に、CTBM ブロックでコンパレータモードを構成するために使用されるさまざまなビットをまとめました。

Table 23-9. 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 [ビット位置]	ビット名 (Bit_Name)	説明
CTBM_OA_RESx_CTRL[4]	OAx_COMP_EN	Opampx コンパレータ イネーブル ビット 0: コンパレータ モードは Opampx で無効 1: コンパレータ モードは Opampx で有効
CTBM_OA_RESx_CTRL[5]	OAx_HYST_EN	Opampx コンパレータ イネーブル ビット 0: ヒステリシスは Opampx で無効 1: ヒステリシスは Opampx で無効
CTBM_OA_RESx_CTRL[6]	OAx_BYPASS_DSI_SYNC	DSI (トリガー) 出力の Opampx 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
CTBM_OA_RESx_CTRL[7]	OAx_DSI_LEVEL	Opampx 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

23.3.4.5 コンパレータ割込み

コンパレータ出力は、割込みを発生するエッジ (ディセーブル / 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。これは CTBM_OA_RESx_CTRL[9:8] ビットで設定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。CTBM_INTR[0] は Comparator0 IRQ に使用され、CTBM_INTR[1] は Comparator1 IRQ に使用されます。各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。詳細については、[割り込み \(51 ページ\)](#) を参照してください。CTBMx_INTR ビットをポーリングすることで、どのコンパレータが ISR をトリガーしたかを確認することができます。

CTBM_INTR_MASK レジスタには、各割込みに該当する割込みマスク ビットがあります。割込みマスクを LOW に設定すると、対応する割込みソースが無視されます。CTBM_INTR レジスタ内の割込みフラグの論理 AND と CTBM_INTR_MASK レジスタ内の対応する割込みマスクが「1」の場合、NVIC への CTBm コンパレータ割込みがアクティブになります。

CTBM_INTR ビット [1:0] に「1」を書き込むと、対応する割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、割込みフラグと割込みマスクの交点 (論理 AND) も CTBM_INTR_MASKED レジスタ内で可能です。

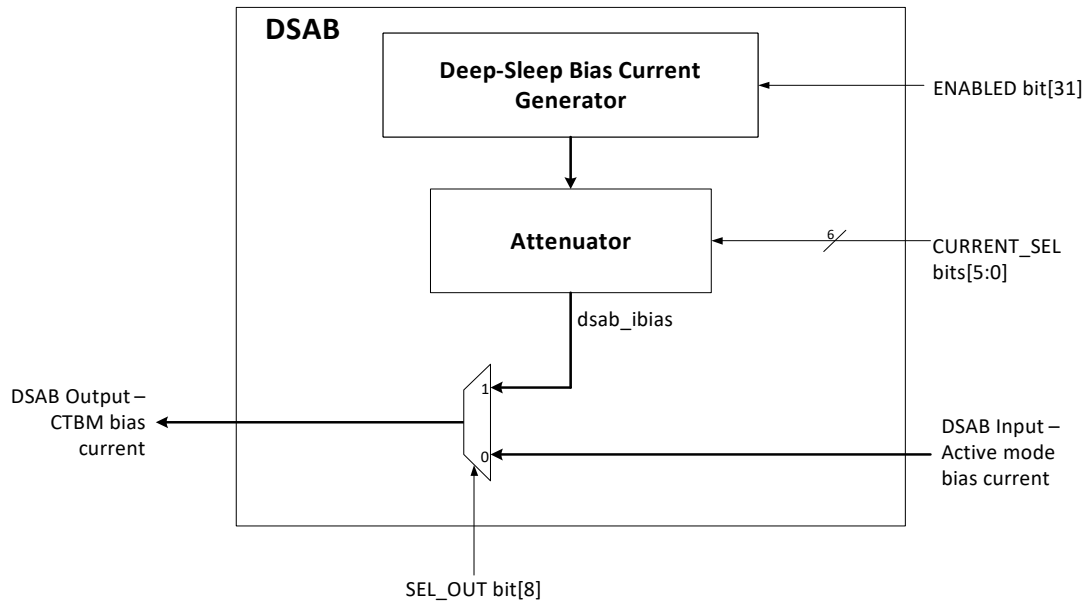
検証およびデバッグ目的のために CTBM_INTR_SET レジスタ内の各割込みにセット ビットが用意されています。これによりファームウェアが実際のコンパレータ切り替えイベントなしで割込みを生成することができます。

23.3.4.6 ディープスリープモード動作

ディープスリープモードでは、バイアス電流、リ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。さまざまな電力モードおよび各モードで使用可能なブロックの詳細については、[電力モード \(107 ページ\)](#) 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。

[Figure 23-2](#) に、DSAB ブロックのアーキテクチャを示します。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 $CURRENT_SEL \times 0.075\mu A (\pm 5\%)$ です。SEL_OUT ビット [8] は、CTBm バイアスにルーティングできる 2 つのバイアス電流間の選択を制御するために使用されます。[Table 23-10](#) に、PASS_DSAB_DSAB_CTRL レジスタのビット構成設定をまとめました。

Figure 23-2. 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。その結果、オペアンプの最大入力コモンモード電圧は約 $V_{DDA} - 1.3V$ に制限されます。さらに、スイッチポンプ (3.3V 未満で動作するときアナログスイッチに必要) が利用できないため、アナログのオン抵抗は電源電圧が 3.3V を下回ると、スイッチは通常の仕様を超えて増加します。信号速度が低い限り、アナログスイッチのオン抵抗を高くすることは正当化できます。したがって、 V_{DDA} は、アナログスイッチの抵抗が大きくなる前に、最大 2.8V になる可能性があります。その結果、最も低い可能な電源電圧を設定します。ただし、ディープスリープモードでオペアンプを使用する場合は、3.3V 以上の V_{DDA} を使用することをお勧めします。ディープスリープモード中のオペアンプの仕様については、[PSoC 4100S datasheet](#) および [PSoC 4100S Plus datasheet](#) をご覧ください。

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、CTBM_CTLB_CTRL レジスタの DEEPSLEEP_ON ビット [30] を設定します。このビットはディープスリープ中に、CTBM の 2 つのオペアンプが有効にされます。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。

Table 23-10. 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 [ビット位置]	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 × 0.075μA (±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

23.4 レジスタ概要

Table 23-11. レジスタ概要

名称	説明
CTBM_CTRL	グローバル CTBm ブロック イネーブル
CTBM_OA_RES0_CTRL	Opamp0 制御レジスタ
CTBM_OA_RES1_CTRL	Opamp1 制御レジスタ
CTBM_COMP_STAT	コンパレータ ステータス
CTBM_INTR	割り込み要求レジスタ
CTBM_INTR_SET	割り込み要求セット レジスタ
CTBM_INTR_MASK	割り込み要求マスク
CTBM_INTR_MASKED	マスクされた割り込み要求
CTBM_OA0_SW	Opamp0 スイッチ制御
CTBM_OA0_SW_CLEAR	Opamp0 スイッチ制御クリア
CTBM_OA1_SW	Opamp1 スイッチ制御
CTBM_OA1_SW_CLEAR	Opamp1 スイッチ制御クリア
CTBM_SW_HW_CTRL	CTBm ハードウェア制御イネーブル
CTBM_SW_STATUS	CTBm パス スイッチ制御ステータス
CTBM_OA0_OFFSET_TRIM	Opamp0 調整制御
CTBM_OA0_SLOPE_OFFSET_TRIM	Opamp0 調整制御
CTBM_OA0_COMP_TRIM	Opamp0 調整制御
CTBM_OA1_OFFSET_TRIM	Opamp1 調整制御
CTBM_OA1_SLOPE_OFFSET_TRIM	Opamp1 調整制御
CTBM_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

24. CapSense



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

自己容量を検知する PSoC 4 の CapSense タッチ検知方法は、CapSense Sigma Delta (CSD) として知られています。同様に、相互容量センシング方式は CapSense クロスポイント (CSX) として知られています。CSD および CSX タッチセンシング方式は、業界最高クラスの信号対雑音比 (SNR)、高いタッチ感度、低電力動作、および優れた EMI 性能を提供します。

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

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

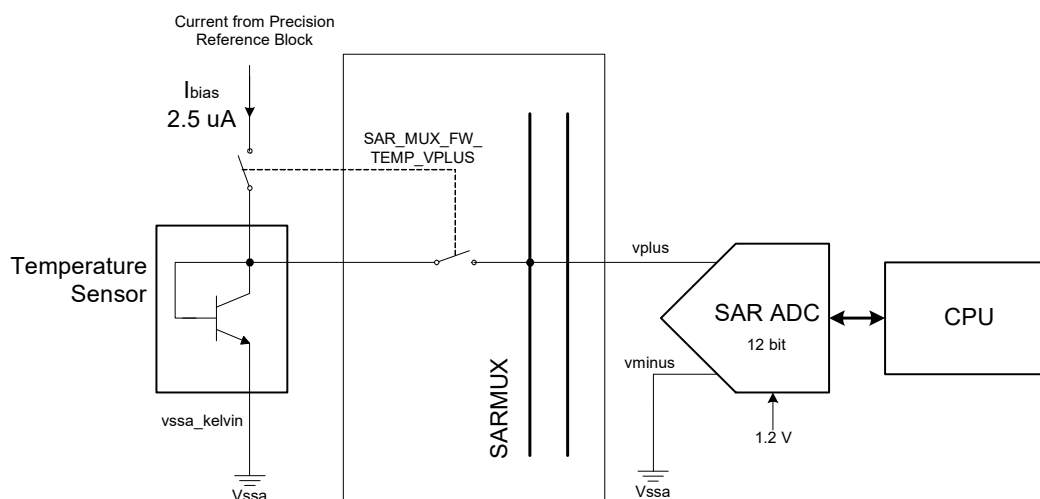
温度センサーは以下の特長があります：

- 温度範囲 $\pm 40^{\circ}\text{C} \sim +85^{\circ}\text{C}$ で摂氏 $\pm 5^{\circ}$ の精度
- 1.2V リファレンスの 12 ビット SAR ADC を使用する場合、摂氏 0.5 度 /LSB 分解能 (増幅なし)
- 10 μs のセトリング時間

25.2 動作方法

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-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。このプロパティは、[Figure 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。

Figure 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：

$$\text{Temp} = (A \times \text{SAR}_{\text{out}} + 2^{10} \times B) + T_{\text{adjust}} \quad \text{Equation 25-1}$$

- 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。

$$A = (\text{signed int}) \left(2^{16} \left(\frac{100^{\circ}\text{C} - (-40^{\circ}\text{C})}{\text{SAR}_{100^{\circ}\text{C}} - \text{SAR}_{-40^{\circ}\text{C}}} \right) \right)$$

Equation 25-2

ここで、

$\text{SAR}_{100^{\circ}\text{C}}$ = 100 °Cでの ADC カウント

$\text{SAR}_{-40^{\circ}\text{C}}$ = -40 °Cでの ADC カウント

定数 'A' は、レジスタ SFLASH_SAR_TEMP_MULTIPLIER に格納されています。

- 「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。

$$B = (\text{unsigned int}) \left(2^6 \times 100^{\circ}\text{C} - \left(\frac{A \times \text{SAR}_{100^{\circ}\text{C}}}{2^{10}} \right) \right)$$

Equation 25-3

ここで、

$\text{SAR}_{100^{\circ}\text{C}}$ = 100 °Cでの ADC カウント

定数 'B' は、レジスタ SFLASH_SAR_TEMP_OFFSET に格納されます。

- T_{adjust} は°C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 $T_{\text{initial}} = (A \times \text{SAR}_{\text{out}} + 2^{10} \times B)$ 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。

$$T_{\text{adjust}} = \left(\frac{0.5^{\circ}\text{C}}{100^{\circ}\text{C} - 15^{\circ}\text{C}} \times (100^{\circ}\text{C} \times 2^{16} - T_{\text{initial}}) \right)$$

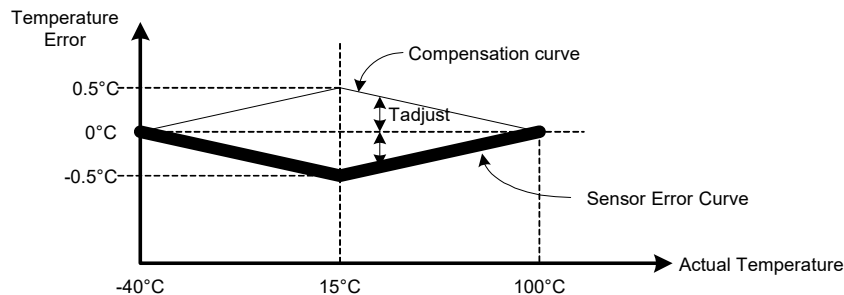
Equation 25-4

中心値より低い場合、 T_{adjust} は以下の式で計算されます。

$$T_{\text{adjust}} = \left(\frac{0.5^{\circ}\text{C}}{40^{\circ}\text{C} + 15^{\circ}\text{C}} \times (40^{\circ}\text{C} \times 2^{16} - T_{\text{initial}}) \right)$$

Equation 25-5

Figure 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 1.2V 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

温度センサー出力はシーケンサまたはファームウェアによって制御される専用のスイッチを介して SAR ADC の非反転入力に接続されます。ADC を使用して温度センサー出力を読み取る方法の詳細については、[SAR ADC \(238 ページ\)](#) を参照してください。

25.4 アルゴリズム

1. SARMUX と SAR ADC を有効にします。
2. $V_{NEG} = V_{SS}$ 、 $V_{REF} = 1.2V$ 、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. 線形方程式 ([Equation 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： $0xBC4B \times 0x595 = (-17333)_{10} \times (1429)_{10} = (-24768857)_{10}$
- b. B に 1024 を掛けます： $0x65B4 \times 0x400 = (26036)_{10} \times (1024)_{10} = (26660864)_{10}$
- 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 = $29.19219^{\circ}\text{C} \sim 29.2^{\circ}\text{C}$ となります。

25.5 レジスタ

名称	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	Equation 25-1 で定義されている乗数定数 'A'。
SFLASH_SAR_TEMP_OFFSET	Equation 25-1 で定義されている定数 'B'。

Section F: プログラムとデバッ

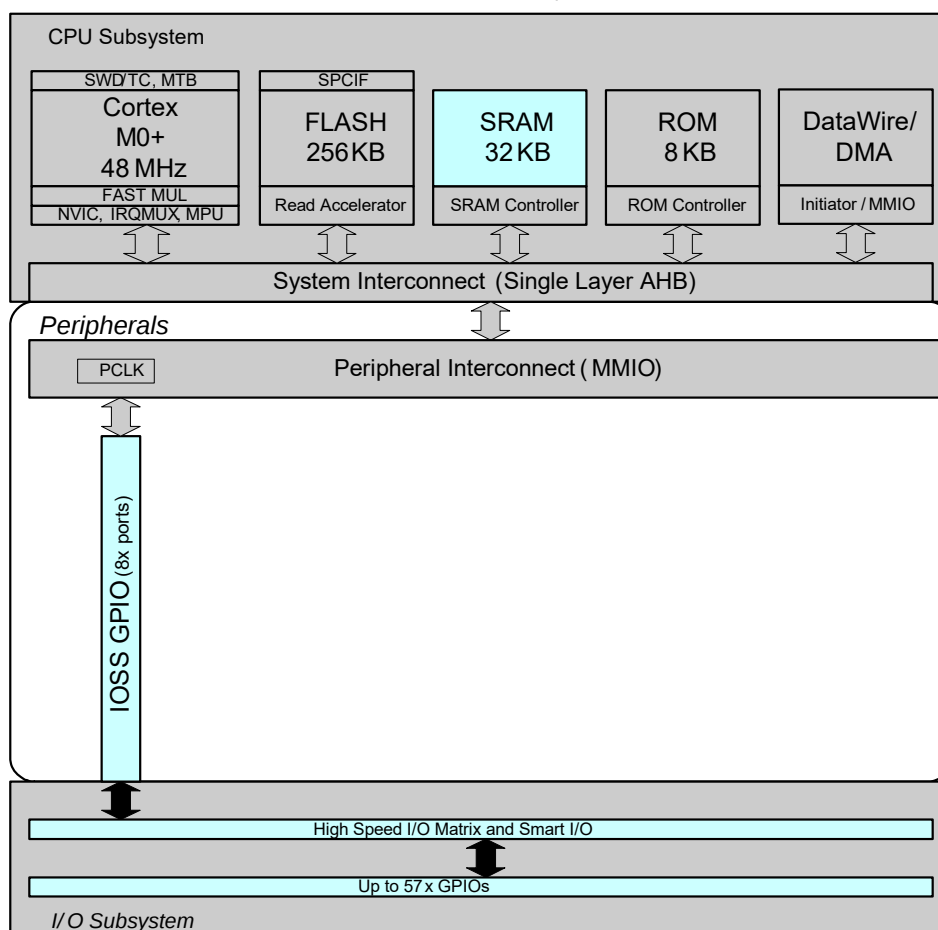


本セクションは次の章を含みます：

- デバッグ インターフェース (288 ページ)
- 不揮発性メモリ プログラム (296 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。シリアルワイヤーデバッグ (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。

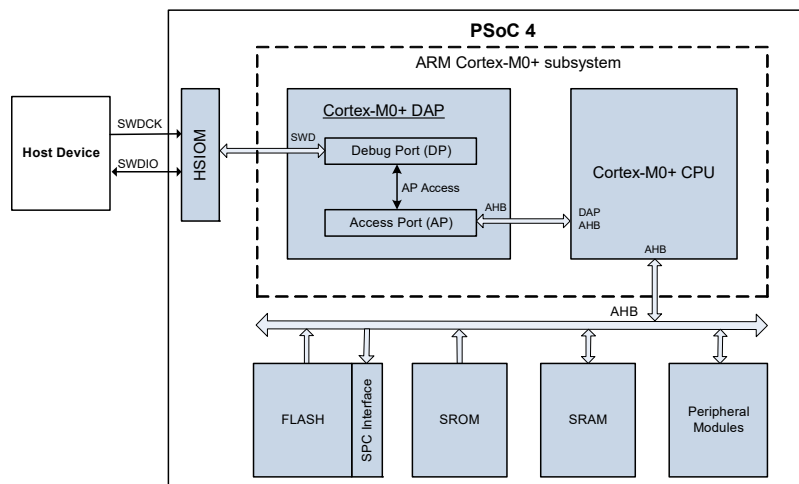
26.1 特長

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

26.2 機能説明

Figure 26-1 に、PSoC 4 のプログラムおよびデバッグインターフェースのブロック図を示します。Cortex-M0+ のデバッグとアクセスポート (DAP) は、プログラムおよびデバッグ インターフェースとして機能します。外部プログラマまたはデバッガ (別名「ホスト」) は、SWD インターフェースの 2 つのピン - 双方向データピン (SWDIO) とホスト駆動クロックピン (SWDCK) を使用して PSoC4 「ターゲット」の DAP と通信します。部品上の SWD ポート端子 (SWDIO と SWDCK) は高速 I/O マトリックス (HSIOM) を通じて DAP と通信します。HSIOM の詳細については、[I/O システム \(66 ページ\)](#) を参照してください。

Figure 26-1. デバッグ インターフェース



DAP は、アーム指定の高度な高性能バス (AHB) インターフェースを使用して Cortex-M0 + CPU と通信します。AHB はデバイス内部で使用するシステム相互接続プロトコルであり、AHB マスターによるメモリとペリフェラル レジスタへのアク

セスを円滑にします。デバイスには、Arm CM0 CPU コアと DAP の 2 つの AHB マスターがあります。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

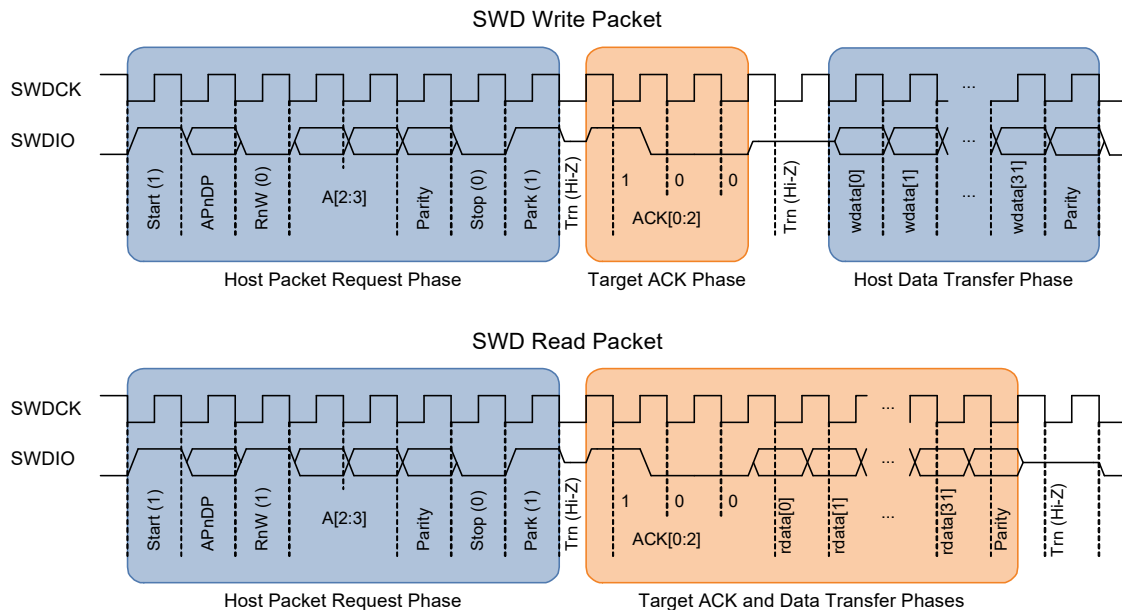
PSoC4 の Cortex-M0+ は、SWD インターフェースを介したプログラミングとデバッグをサポートします。SWD プロトコルは、パケットベースのシリアル トランザクション プロトコルです。端子レベルでは双方向データ信号 (SWDIO) と一方向クロック信号 (SWDCK) を使用します。クロックラインは常にホスト プログラマが駆動し、データラインはホストまたはターゲットのいずれかが駆動します。完全なデータ転送 (1 つの SWD パケット) は 46 クロックを必要とし、次の 3 つの段階から成り立っています：

- **ホストパケットリクエストフェーズ** – ホストが PSoC4 ターゲットにリクエストを発行します。
- **ターゲット確認応答フェーズ** – PSoC 4 ターゲットは確認をホストに送信します。
- **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。

Figure 26-2 に、読み取りおよび書き込み SWD パケットのタイミング図を示します。

Figure 26-2. SWD 読み書きパケットのタイミング図



SWD 読み書きパケットを送信するシーケンスを下記に示します。

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」 ビット (RnW) は、データ転送の方向を制御します。1b1 はターゲットの「読み取り」を表し、1b0 はターゲットの「書き込み」を表します。

- 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、Table 26-3 と Table 26-4 を参照してください。注：アドレスビットは、LSB ファーストで送信されます。
- 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あります。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場合、ヘッダーは PSoC 4 によって無視されます。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、Table 26-1 を参照してください。注: ACK ビットは LSB ファーストで送信されます。
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 ラインをクロック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 1 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。

Table 26-1 と Figure 26-2 は、SWDIO ビットの書き込みと読み取りのタイミングを示しています。

Table 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
Host Packet Request (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
Host Data Transfer (ホスト データ転送)		
Target Ack Response (ターゲット ACK 応答)	ホスト読み出し	ターゲット書き込み
Target Data Transfer (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 応答はデータ転送段階を必要とします。FAULT 状態の場合、プログラミング動作を直ちに中断させる必要があります。Table 26-2 は、ACK ビット フィールドのデコードの詳細を示しています。

Table 26-2. SWD 転送 ACK 応答の復号化

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、ホストはデータ転送段階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、データ転送段階は 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

Figure 26-2 に示すように、パケットリクエストと ACK フェーズの間、およびホストライト転送の ACK とデータフェーズの間には、ターンアラウンド期間があります。SWD プロトコルに従って、Trn 期間は、ホストとターゲットの両方によってそれぞれの SWDIO ラインで駆動モードを変更するために使用されます。パケット要求後の最初の Trn 期間中に、ターゲットは SWDCK の立ち上がりエッジで SWDIO ライン上の ACK データを駆動し始めます。この動作によりホストが次の立ち下がりエッジで ACK データを読み出すことを確保します。従って最初の Trn の持続時間はサイクルの半分しかありません。SWD パケットの 2 番目の Trn 期間は 1 サイクル半です。ホストも PSoC 4 も、Trn 期間中に SWDIO ラインを駆動するべきではありません。

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 識別コード等も含まれています。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可能に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 の完全なアーキテクチャの詳細は、[Arm® Debug Interface v5 Architecture Specification](#) に記載されています。

26.4.1 デバッグ ポート (DP) レジスタ

Table 26-3 は、プログラミングとデバッグに使用される Cortex-M0 + DP レジスタと、対応する SWD アドレスビットの選択を示しています。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 レジスタに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® Debug Interface v5 Architecture Specification](#) を参照してください。

Table 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	完全名	レジスタの機能
ABORT	0 (DP)	2b00	0 (W)	AP アボート レジスタ	このレジスタは DAP を強制中断させ、エラーと曖昧な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、Cortex-M0 + CPU の SWD ID である 0x0BC11477 を保持します。
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する

26.4.2 アクセスポート (AP) レジスタ

Table 26-4 は、プログラミングとデバッグに使用される主要な Cortex-M0 + AP レジスタと、対応する SWD アドレスビットの選択を示しています。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。

Table 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	完全名	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	このレジスタは、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 は以下のシーケンスを使用してプログラムされます。プログラミングアルゴリズム、タイミング仕様、およびプログラミングに必要なハードウェア構成の詳細については、[PSoC 4100M, PSoC 4200M, PSoC 4200D, PSoC 4400, PSoC 4000S, PSoC 4700S Device Programming Specifications](#) を参照してください。

1. PSoC 4 の SWD ポートを取得します。
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 SWD ポートの開通シーケンス

デバイスプログラミングの最初のステップは、ホストが target's の SWD ポートを取得することです。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 ポートが疑似コードを取得
ToggleXRES(); // Toggle XRES pin to reset device

//Execute Arm's connection sequence to acquire SWD-port
do
{
    SWD_LineReset(); //perform a line reset (50+ SWDCK clocks with SWDIO high)
    ack = Read_DAP ( IDCODE, out ID); //Read the IDCODE DP register
}while ((ack != OK) && time_elapsed < ms); // retry connection until OK ACK or timeout

if (time_elapsed >= ms) return FAIL; //check for acquire time out

if (ID != CMOP_ID) return FAIL; //confirm SWD ID of Cortex-M0+ CPU. (0x0BC11477)
```

この疑似コードでは、SWD_LineReset() は、デバッグアクセスポートをリセットする標準の 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。

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。ラインリセットと IDCODE 読み取りのシーケンスは、IDCODE 読み取りに対して OK ACK が受信されるか、タイムアウト (ms) が発生するまで繰り返す必要があります。タイムウィンドウ内に OK ACK が受信され、読み取られた IDCODE が Cortex-M0 + DAP の IDCODE と一致する場合、SWD ポートは取得済み状態にあると言えます。

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。プログラミングモードに移行するためのタイミング仕様と疑似コードは、[PSoC 4100M, PSoC 4200M, PSoC 4200D, PSoC 4400, PSoC 4000S, PSoC 4700S Device Programming Specifications](#) ドキュメントに詳しく記載されています。ポート開通の手段とこの手段を実行するために最低限必要なクロック周波数は 1.5MHz です。

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 モードのとき、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。プログラミングルーチンについては、[不揮発性メモリ プログラム \(296 ページ\)](#) を参照してください。プログラミングルーチンを呼び出す正確なシーケンスは、[PSoC 4100M, PSoC 4200M, PSoC 4200D, PSoC 4400, PSoC 4000S, PSoC 4700S Device Programming Specifications](#) に記載されています。

26.6 PSoC4 SWD デバッグインターフェース

Cortex-M0+ DAP デバッグ機能は、浸入デバッグおよび非浸入デバッグの 2 種類に分けられています。浸入デバッグには、プログラムのホールドとステップ実行、ブレークポイントおよびデータ ウォッチポイントがあります。非浸入デバッグには、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構成レジスタ
- 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+ デバッグで未サポートです。

デバッグアーキテクチャの詳細については、「[Armv6-M Architecture Reference Manual](#)」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。これらのレジスタの完全なビットレベルの定義については、「[Armv6-M Architecture Reference Manual](#)」を参照してください。

- デバッグ 停止 制御 および ステータス レジスタ (CM0P_DHCSR) – このレジスタには、デバッグを有効にし、CPU を停止し、シングルステップ操作を実行するための制御ビットがあります。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 ビットも含まれます。
- デバッグ フォールト ステータス レジスタ (CM0P_DFSR) – このレジスタはデバッグ イベントが発生した理由を説明します。これには、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 イベントがあります。
- デバッグ コア レジスタ セレクト レジスタ (CM0P_DCRSR) – このレジスタは、Cortex-M0 + CPU での汎用レジスタを選択するために使用されます。この汎用レジスタへの読み書き動作は外部デバッグによって実行される必要があります。
- デバッグ コア レジスタ データ レジスタ (CM0P_DCRDR) – このレジスタは、CM0P_DCRSR レジスタで選択したレジスタに転送するデータを格納するために使用されます。
- デバッグ 例外 および モニター 制御 レジスタ (CM0P_DEMCR) – このレジスタは、グローバル デバ

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できます。BPU には 2 種類のレジスタがあります。

- ブレークポイント制御レジスタ (CM0P_BP_CTRL) は、BPU を有効にし、デバッグシステムでサポートされているハードウェアブレークポイントの数を格納するために使用されます (PSoC4 の CM0 DAP には 4 つ)。
- ハードウェア ブレークポイントごとに 1 つのブレークポイント比較レジスタ (CM0P_BP_COMPx) があります。ブレークポイント比較レジスタはブレークポイントのイネーブル ビット、比較アドレス値およびブレークポイント デバッグ イベントのトリガー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 (PC) 命令アドレスでウォッチポイント サポートを提供しています。DWT は 2 ウォッチポイントをサポートします。また PC サンプル レジスタを使用して外部プログラム カウンタ 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 カウンタの非浸入のおおまかなプロファイリングに使用されます。DWT の最も重要なレジスタを次に示します。

- ウォッチポイント比較 (CM0P_DWT_COMPx) レジスタは、ウォッチポイントコンパレータがウォッチポイント イベントの生成に使用する比較値を格納します。各ウォッチポイントに対して関連する DWT_COMPx レジスタを持っています。
- ウォッチポイントマスク (CM0P_DWT_MASKx) レジスタは、関連するウォッチポイントで一致するアドレス範囲に適用される無視マスクを格納します。
- ウォッチポイント機能 (CM0P_DWT_FUNCTIONx) レジスタは、ウォッチポイント イベントのトリガー条件を格納します。イベントはプログラム カウンタ ウォッチポイント イベントまたはデータ アドレス読み書きアクセス ウォッチポイント イベントです。また関連するウォッチポイント イベントが発生するとステータス ビットを設定します。

- ウォッチポイントコンパレータ PC サンプルレジスタ (CM0P_DWT_PCSR) は、プログラムカウンタの現在の値を格納します。このレジスタはプログラム カウンタレジスタの非侵入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構成レジスタ、BPU 内のレジスタ、および DWT 内のレジスタにアクセスして、ターゲット PSoC4 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。

ターゲット PSoC 4 をデバッグする最初のステップは、SWD ポートを取得することです。開通シーケンスは、SWD リ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、対応する端子を他の目的に使用しないでください。SWD ポートピンを設定して SWD インター

フェースまたは LCD や GPIO などの他の機能にのみ使用できるようにする方法を理解するには、[I/O システム \(66 ページ\)](#) を参照してください。デバッグが必要とされた場合は、SWD ポート端子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 端子を他の目的にも使用できます。

SWD ポートが開通した時、外部デバッグ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、ホールド、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、[デバイスセキュリティ \(63 ページ\)](#) で説明されている全体的なデバイス保護設定の影響も受けます。OPEN 保護モードのみがデバイス デバッグに対応します。デバイスがアクティブ モードからディープスリープ モードまたはスリープ モードのいずれかに移行する場合、外部デバッグとターゲット 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、デバッグ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

Table 26-5. レジスタ一覧

レジスタ名	説明
CM0P_DHCSR	デバッグ ホールド制御およびステータス レジスタ
CM0P_DFSR	デバッグ フォールト ステータス レジスタ
CM0P_DCRSR	デバッグ コア レジスタ選択レジスタ
CM0P_DCRDR	デバッグ コア レジスタ データ レジスタ
CM0P_DEMCR	デバッグ例外およびモニター制御レジスタ
CM0P_BP_CTRL	ブレークポイント制御レジスタ
CM0P_BP_COMPx	ブレークポイント比較レジスタ
CM0P_DWT_COMPx	ウォッチポイント比較レジスタ
CM0P_DWT_MASKx	ウォッチポイント マスク レジスタ
CM0P_DWT_FUNCTIONx	ウォッチポイント機能レジスタ
CM0P_DWT_PCSR	ウォッチポイント コンパレータ PC サンプル レジスタ

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他のサードパーティプログラマは、これらの関数を使用して、アプリケーション hex ファイルのデータで PSoC4 デバイスをプログラムできます。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 機能説明

フラッシュ プログラム動作はシステム コールとして搭載されています。システムコールは特権モードで SROM から実行されます。ユーザは 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、システム性能コントローラ インターフェース (SPCIF) の入力レジスタに関数オペコードと関数パラメータを書き込んで、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 (SPC) は SROM から対応するシステム コールを実行して、SPCIF ステータス レジスタを更新します。DAP または CPU は関数実行の成否を取得するためにこのステータス レジスタを読み出す必要があります。関数実行の一部である SROM のコードは、SPCIF とやり取りすることにより、実際のフラッシュ プログラム動作を実行します。

PSoC 4 フラッシュはプログラム消去のプログラム (PEP) シーケンスによりプログラムされます。すべてのフラッシュ セルは既知の状態にプログラムされ、消去されてから、選択したビットがプログラムされます。このシーケンスは蓄積電荷のバランスを取ることで、フラッシュの寿命を長期化します。フラッシュに書き込む時には、データは最初にページラッチバッファにコピーされます。フラッシュ書き込み関数は後程このデータをフラッシュに転送するために使用されます。

外部のプログラマは、コマンドをデバッグ アクセスポート (DAP) に送信することで、SWD プロトコルを使用して PSoC 4 のフラッシュ メモリをプログラムします。外部プログラマを使用した PSoC 4 デバイスのプログラミングシーケンスは、[PSoC 4100M](#), [PSoC 4200M](#), [PSoC 4200D](#), [PSoC 4400](#), [PSoC 4000S](#), [PSoC 4100S](#), [PSoC 4700S Programming Specifications](#) に記載されています。また、CM0+ CPU は AHB インターフェースを介して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：フラッシュへの書き込みには、20 ミリ秒もかかる場合があります。この期間中のデバイスのリセットは禁止です。予期しない変更がフラッシュに発生する可能性があります。リセットソース ([リセット システム \(120 ページ\)](#) を参照) には、XRES ピン、ソフトウェアリセット、およびウォッチドッグが含まれます。これらが誤ってアクティブにならないようにしてください。また低電圧検出回路はリセットではなく割込みが発生するように設定される必要があります。

注：PSoC 4 は、アプリケーション固有の情報を保存するために使用できるユーザ監視フラッシュ (SFlash) を実装しています。これらの行は hex ファイルの一部ではありません。そのプログラミングはオプションです。

27.3 システム コールの搭載

システム コールは以下のものから構成されます：

- オペコード：一意の 8 ビットオペコード
- パラメータ：2つの8ビットのパラメータはすべてのシステム コールに必須です。これらのパラメータはキー 1 とキー 2 と呼ばれ、以下のように定義されます：
 $\text{key1} = 0\text{x}B6$
 $\text{key2} = 0\text{x}D3 + \text{Opcode}$
 2つのキーを渡すことでユーザ 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。キー 1 とキー 2 のパラメータが正しくない場合、SRAM 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を必要とすることがあります。
- 戻り値：いくつかのシステム コールは実行完了時に、シリコン ID あるいはチェックサムなどの値を戻します。
- 完了ステータス：各システム コールは32ビットのステータスを返し、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と、CPU は SRAM で対応するコードにジャンプします。実行が完了すると、元のスレッドの実行が再開されます。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だけにシステム コールを使用します。そのプロセスでは CPU が停止します。

非ブロッキング システム コールは非ブロッキング列書き込み、非ブロッキング列プログラムとレジューム 非ブロッキングの 3 つがあります。その他のシステム コールはすべてがブロッキング システム コールです。

フラッシュ上で消去またはプログラム操作を実行している間、CPU はフラッシュからコードを実行できないため、ノンブロッキングシステムコールは、SRAM から実行されているコードからのみ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た場合、結果は未定義であり、フラッシュフェッチ操作の実行中にバスエラー

を返し、ハードフォールトをトリガーする可能性があります。

システム性能コントローラ (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。システム コールの後続ステップが完了することを保証するために、SPC 割り込みサービスルーチン (ISR) からノンブロッキング再開関数を呼び出します。CPU は非ブロッキング書き込みまたはプログラム操作が行われているときに SRAM からのみコードを実行できるため、SPC ISR も SRAM に配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関数の場合は 1 回、非ブロッキング書き込み操作では 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 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 コールの開始手順を以下に示します。

1. 関数パラメータの設定：関数パラメータ (キー 1、キー 2、追加のパラメータ) を準備する 2 つの方法について、詳細を以下に示します。
 - a. CPUSS_SYSARG レジスタへの関数パラメータの書き込み：この方法は、CPUSS_SYSARG レジスタからのパラメータ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で書き込む必要があります。
 - b. 関数パラメータを SRAM に書き込む：このメソッドは、SRAM からパラメータを取得する関数に使用されます。最初に、パラメータを指定された順序で連続した SRAM の場所へ書き込む必要があります。それから、最初のパラメータのアドレスである 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_SYSREQ 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が参照する SRAM 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_SYSREQ 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完全なリストとその説明については、[Table 27-1](#) を参照してください。

5. 戻り値の回収: シリコン ID やチェックサムなどの値を戻すシステム コールの場合、CPU または DAP は CPUSS_SYSREQ と CPUSS_SYSARG のレジスタを読み出して戻り値をフェッチします。

27.5 システム コール

[Table 27-1](#) は、PSoC 4 でサポートされるすべてのシステム コールを、機能の説明とデバイス保護モードでの可用性とともに示します。デバイス保護設定の詳細については、[デバイスセキュリティ \(63 ページ\)](#) 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。

Table 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
		開放	保護	キル	
Silicon ID (シリコン ID)	デバイス シリコン ID、ファミリ ID およびリビジョン ID を戻します	✓	✓	–	✓
Load Flash Bytes (ロードフラッシュ バイト)	後でサイズが 128 バイトのフラッシュ行にプログラムされるデータを、1 バイト単位でページ ラッチ バッファにロード	✓	–	–	✓
Write Row (列書き込み)	フラッシュ行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	✓	–	–	✓
Program Row (列プログラム)	ページ ラッチ バッファ内のデータでフラッシュ行をプログラム	✓	–	–	✓
Erase All (全消去)	フラッシュ アレイのすべてのユーザ コードおよび監視フラッシュ領域にあるフラッシュ行レベル保護データを消去	✓	–	–	
Checksum (チェックサム)	フラッシュ メモリ全体 (ユーザおよび監視領域) 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	✓	✓	–	✓
Write Protection (書き込み保護)	フラッシュ行レベルの保護設定およびチップ レベルの保護設定を監視フラッシュ (行 0) にプログラム	✓	✓	–	
Non-Blocking Write Row (非ブロッキング列書き込み)	フラッシュ行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 / 消去パルスの間、ユーザは SRAM からコードを実行できます。この機能は CPU アクセス専用です	–	–	–	✓
Non-Blocking Program Row (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フラッシュ行をプログラム。プログラム / 消去パルスの間、ユーザは SRAM からコードを実行できます。この機能は CPU アクセス専用です	–	–	–	✓
Resume Non-Blocking (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戻します。これらの値は CPUSS_SYSARG と CPUSS_SYSREQ レジスタに戻されます。パラメータは CPUSS_SYSARG と CPUSS_SYSREQ レジスタに渡されます。

パラメータ

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 (キー 1)
ビット [15:8]	0xD3	Key2 (キー 2)
ビット [31:16]	0x0000	未使用
CPUSS_SYSREQ レジスタ		
ビット [15:0]	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

アドレス戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [7:0]	シリコン ID 下位	
ビット [15:8]	シリコン ID 上位	2500-25FF
ビット [19:16]	マイナー リビジョン Id	これらの値については、 PSoC 4100M, PSoC 4200M, PSoC 4200D, PSoC 4400, PSoC 4000S, PSoC 4100S, PSoC 4700S Programming Specifications を参照してください
ビット [23:20]	メジャー リビジョン Id	
ビット [27:24]	0xXX	未使用 (ドントケア)
ビット [31:28]	0xA	成功ステータス コード
CPUSS_SYSREQ レジスタ		
ビット [11:0]	ファミリ ID	PSoC 4100S Plus のファミリ ID は 0xAB です
ビット [15:12]	チップ保護	デバイスセキュリティ (63 ページ) を参照
ビット [31:16]	0XXXXX	未使用

27.5.2 クロック設定

この関数はフラッシュ プログラミングおよび消去動作に必要なクロックを初期化します。この API は、フラッシュ書き込みとフラッシュ消去 API の呼び出しの前に、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz で IMO に設定されていることを確認するために使用されます。IMO がチャージポンプクロックのソースであり、48 MHz でない場合、フラッシュの書き込みおよび消去 API はフラッシュに作用せずに終了し、「無効なポンプクロック周波数」ステータスを返します。

27.5.3 ロード フラッシュ バイト

この関数は、フラッシュ行にプログラムされるデータをページ ラッチ バッファにロードします。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「バイト アドレス」の入力パラメータ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 ラッチの内容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 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

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xD7	Key2 (キー 2)
ビット [23:16]	バイト アドレス	データ書き込み用のページ ラッチ バッファの開始アドレス 0x00 – ラッチ バッファのバイト 0 0x7F – ラッチ バッファのバイト 127
ビット [31:24]	フラッシュ マクロ選択	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数については、 Cortex-M0+ CPU (28 ページ) 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	Don't care パラメータ
ビット [23:16]	0XXX	Don't care パラメータ
ビット [31:24]	0XXX	Don't care パラメータ
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
.	.	.
.	.	.
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 バイト
CPUSS_SYSARG レジスタ		
ビット [31:0]	32'hYY	最初の関数パラメータ (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0004	フラッシュ バイト ロード オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xFFFFFFFF	未使用 (ドントケア)

27.5.4 列書き込み

この関数は、ページ ラッチ バッファ内のデータで 1 列のフラッシュを消去してから、プログラムします。ページ ラッチ バッファのすべてのデータが 0 になると、プログラムはスキップされます。この機能のパラメータは SRAM に保存されます。保存されたパラメータの開始アドレスは、CPUSS_SYSARG レジスタに書き込まれます。この関数は、列がプログラムされた後で、ページ ラッチ コンテンツをクリアします。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前にロード フラッシュ バイト関数を呼び出します。この関数は対応するフラッシュ列が書き込み保護されていない場合のみに書き込み動作を行うことができます。

詳細については、*PSoC 4100S: PSoC 4 Registers TRM* および *PSoC 4100S Plus: PSoC 4 Registers TRM* の CLK_IMO_CONFIG レジスタを参照してください。

パラメータ

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xD8	Key2 (キー 2)
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 (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0005	列書き込みオPCODE
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0XXXXXXXX	未使用 (ドントケア)

27.5.5 列プログラム

この関数は、ページ ラッチ バッファ内のデータで、フラッシュのアドレス付けされた列をプログラムします。ページ ラッチ バッファのすべてのデータが 0 になると、プログラムはスキップされます。この関数を呼び出す前に、列が消去状態である必要があります。列がプログラムされた後でページ ラッチバッファの内容をクリアします。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前にロード フラッシュ バイト関数を呼び出します。この関数を呼び出す前に、列が消去状態である必要があります。この関数は対応するフラッシュ列が書き込み保護されていない場合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

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xD9	Key2 (キー 2)
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 (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0006	列プログラムオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0xFFFFFFFF	未使用 (ドントケア)

27.5.6 全消去

この関数は、フラッシュの主アレイのすべてのユーザ コード、各フラッシュ マクロの監視フラッシュ行 0 の行レベル保護データを消去します。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。チップの保護モードが OPEN であり DAP からのプログラム モードの場合のみに、この API は呼び出すことができます。チップの保護モードが PROTECTED である場合、保護の設定を OPEN に変更するために、DAP は書き込み保護 API を使用する必要があります。保護設定を PROTECTED から OPEN に変更すると自動的に全消去が行われます。

パラメータ

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xDD	Key2 (キー 2)
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 (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x000A	すべてのオペコードを消去
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0XXXXXXXX	未使用 (ドントケア)

27.5.7 チェックサム:

この関数は、フラッシュメモリの全体または 1 フラッシュ行を読み出し、そのフラッシュ領域で読み出した各バイトの 24 ビット チェックサムを返します。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として渡されます。パラメータのバイト 2 とバイト 3 は、チェックサムをフラッシュメモリの全体またはユーザコードフラッシュ行で実行するかを選択します。

パラメータ

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 (キー 1)
ビット [15:8]	0xDE	Key2 (キー 2)
ビット [31:16]	列 ID	チェックサム演算を実行するフラッシュ行の番号を選択。 列の番号 - 16 ビットのフラッシュ列の番号 または 0x8000 - フラッシュメモリ全体のチェックサムが計算されます
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x000B	チェックサム オペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:24]	0xX	未使用 (ドントケア)
ビット [23:0]	チェックサム:	選択したフラッシュ領域の 24 ビットチェックサム値

27.5.8 書き込み保護

この関数は、フラッシュ列レベル保護の設定とデバイスの保護の設定の両方を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、スーパバイザ フラッシュの列 0 に位置する最後のバイトのフラッシュ マクロ ゼロに保存されます。監視フラッシュ列のサイズはユーザ コード フラッシュ列のサイズと同じです。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージ ポンプ クロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。ロード フラッシュ バイト関数は、フラッシュ マクロのフラッシュ保護バイトをマクロに対応したページ ラッチ バッファにロードするために使用されます。ロード関数用の開始アドレス パラメータは 0 にする必要があります。フラッシュ マクロ番号はプログラムの対象となるマクロでなければならず、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として渡されます。

パラメータ

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 (キー 1)
ビット [15:8]	0xE0	Key2 (キー 2)
ビット [23:16]	デバイス保護バイト	Flash Macro 0 にのみ適用可能なパラメータ 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PCODE
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:24]	0xX	未使用 (ドントケア)
ビット [23:0]	0x000000	

27.5.9 非ブロッキング列書き込み

この関数はフラッシュ行がノンブロッキング方式で CM0+ CPU によって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。ノンブロッキングシステムコールの説明は、“[ブロッキングと非ブロッキングのシステム コール](#)” (297 ページ) で説明されています。

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。

各フェーズが実行されている間、CPU は SRAM からコードを実行できます。非ブロッキング書き込み行システムコールが開始されると、非ブロッキング書き込み操作の完了に必要な Resume Non-Blocking 関数以外のシステムコール関数を呼び出すことはできません。各段階が完了した後、SPC は自らの割込みをトリガーします。この割込みでは、ノンブロッキング再開のシステム コールを呼び出します。

注： デバイスファームウェアは、非ブロッキング書き込み行中にデバイスをスリープ状態にしようとししないでください。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。

注意事項： この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージポンプクロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前に、列プログラムするデータバイトをロードするためロードフラッシュバイト関数を呼び出します。また、ノンブロッキング書き込み行関数は、SRAM からのみ呼び出すことができます。これは、CM0 + CPU がフラッシュ消去プログラム操作を実行している間、フラッシュからコードを実行できないためです。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、ハードフォールトをトリガーします。

パラメータ

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xDA	Key2 (キー 2)
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 (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0007	非ブロッキング列書き込みオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータスコード
ビット [27:0]	0XXXXXXXX	未使用 (ドントケア)

27.5.10 非ブロッキング列プログラム

この関数はフラッシュ列が非ブロッキング方式で CM0+ CPU にプログラムされる必要時に使用され、CPU はプログラム動作が継続中でも SRAM からのコードを実行することができます。ノンブロッキングシステムコールの説明は、“[ブロッキングと非ブロッキングのシステムコール](#)” (297 ページ) で説明されています。プログラム操作の実行中、CPU は SRAM からコードを実行できます。非ブロッキングプログラム行システムコールが呼び出されると、ユーザは、非ブロッキング書き込み操作の完了に必要な Resume Non-Blocking 関数以外のシステムコール関数を呼び出すことができません。

非ブロッキング列書き込みのシステムコールと異なり、プログラムシステムコールには単一の段階しかありません。従ってレジューム非ブロッキング関数は、非ブロッキングプログラムシステムコールの使用時に、SPC 割り込みから 1 回だけ呼び出す必要があります。

注意事項：この関数を呼び出す前に設定クロック API 関数を呼び出します。設定クロック API は、チャージポンプクロック (clk_pump) と HF クロック (clk_hf) が 48MHz の IMO に設定されていることを保証します。この関数を呼び出す前に、列プログラムするデータバイトをロードするためロードフラッシュバイト関数を呼び出します。また、ノンブロッキングプログラム行関数は、SRAM からのみ呼び出すことができます。これは、CM0 + CPU がフラッシュプログラム操作を実行している間、フラッシュからコードを実行できないためです。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、ハードフォールトをトリガーします。

パラメータ

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xDB	Key2 (キー 2)
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 (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0008	非ブロッキング列プログラムオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータスコード
ビット [27:0]	0XXXXXXXX	未使用 (ドントケア)

27.5.11 レジューム非ブロッキング

この関数は非ブロッキング列書き込みと非ブロッキング列プログラムのシステム コールにより開始される消去とプログラムの追加段階を完了します。この関数は非ブロッキング列書き込みの呼び出しの後に 3 回、非ブロッキング 列プログラムの呼び出しの後に 1 回呼び出す必要があります。プログラム動作または消去動作のすべての段階が完了する前に、その他のシステム コールは実行できません。非ブロッキング関数の使用手順の詳細については、“[ブロッキングと非ブロッキングのシステム コール](#)” (297 ページ) を参照してください。

パラメータ

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、ワード整列の SRAM アドレス)		
ビット [7:0]	0xB6	Key1 (キー 1)
ビット [15:8]	0xDC	Key2 (キー 2)
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 (キー 1) を格納する SRAM の 32 ビットのワード整列アドレス
CPUSS_SYSREQ レジスタ		
ビット [15:0]	0x0009	レジューム非ブロッキングオペコード
ビット [31:16]	0x8000	SYSCALL_REQ ビットをセット

戻り値

アドレス	戻り値	説明
CPUSS_SYSARG レジスタ		
ビット [31:28]	0xA	成功ステータス コード
ビット [27:0]	0XXXXXXXX	未使用 (ドントケア)

27.6 システム コール ステータス

それぞれのシステム コールを実行すると、ステータス コードが CPUSS_SYSARG レジスタの引数に上書きされます。成功のステータスは 0xAXXXXXXX であり、その中の X はシステム コールが値を戻す場合に「ドントケア」値または戻り値を示します。失敗のステータスは 0xF00000XX と示され、その中の XX が失敗コードです。

Table 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 の 32 ビット値)	説明
AXXXXXXXh	成功:「X」は、API が CPUSS_SYSARG レジスタに直接パラメータを戻さない限り、SRAM によって戻された「0」の値をもつ「ドントケア」値を示します。
F0000001h	無効なチップ保護モード: この API は現在のチップ保護モード中は使用不可
F0000003h	無効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、または提供したサイズがページアドレスより大きすぎます。
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	再開完了: すべてのノンブロッキング API は完了。再開 API は次のノンブロッキング API が実行されるまで呼び出すことが不可
F0000008h	保留中再開: ノンブロッキング API は開始された後、他の API が呼び出される前に再開 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 は次の再開動作を試みる前に発生する必要がある
F000000Ah	チェックサムゼロ失敗: 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 オペコードではない
F000000Ch	キーオペコード不一致: 提供したオペコードは key1 と key2 が一致しません。
F000000Eh	無効な開始アドレス: 開始アドレスが指定した終了アドレスよりも大きい
F0000012h	無効なポンプクロック周波数: フラッシュ書き込みまたは消去が実行される前に IMO を 48MHz、HF クロック ソースを IMO クロック ソースに設定することが必要です

27.7 非ブロッキング システム コール 疑似コード

本節は、ノンブロッキング システム コールを設定し、フラッシュ プログラム動作中に 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_REG  REG( 0x40100000 )
#define CPUSS_SYSREQ_REG  REG( 0x40100004 )
#define CPUSS_SYSARG_REG  REG( 0x40100008 )

#define ROW_SIZE_        ()
#define ROW_SIZE         (ROW_SIZE_)

/*Variable to keep track of how many times SPC ISR is triggered */
__ram int iStatusInt = 0x00;

__flash int main(void)
{
    DoUserStuff();

    /*CM0+ interrupt enable bit for spc interrupt enable */
    CM0_ISER_REG |= 0x00000040;

    /*Set CPUSS_CONFIG.VECS_IN_RAM because SPC ISR should be in SRAM */
    CPUSS_CONFIG_REG |= 0x00000001;

    /*Call non-blocking write row API */
    NonBlockingWriteRow();

    /*End Program */
    while(1);
}

__sram void SpcIntHandler(void)
{
    /* Write key1, key2 parameters to SRAM */
    REG( 0x20000000 ) = 0x0000DCB6;

    /*Write the address of key1 to the CPUSS_SYSARG reg */
    CPUSS_SYSARG_REG = 0x20000000;

    /*Write the API opcode = 0x09 to the CPUSS_SYSREQ.COMMAND
    * register and assert the sysreq bit
    */
    CPUSS_SYSREQ_REG = 0x80000009;

    /* Number of times the ISR has triggered */
    iStatusInt ++;
}

__sram void NonBlockingWriteRow(void)
{
    int iter;

    /*Load the Flash page latch with data to write*/
    /* Write key1, key2, byte address, and macro sel parameters to SRAM
    */
    REG( 0x20000000 ) = 0x0000D7B6;
```

```
//Write load size param (128 bytes) to SRAM
REG( 0x20000004 ) = 0x0000007F;

for(i = 0; i < ROW_SIZE/4; i += 1)
{
    REG( 0x20000008 + i*4 ) = 0xDADADADA;
}

/*Write the address of the key1 param to CPUSS_SYSARG reg*/
CPUSS_SYSARG_REG = 0x20000000;

/*Write the API opcode = 0x04 to CPUSS_SYSREQ.COMMAND
 * register and assert the sysreq bit
 */
CPUSS_SYSREQ_REG = 0x80000004;

/*Perform Non-Blocking Write Row on Row 200 as an example.
 * Write key1, key2, row id to SRAM row id = 0xC8 -> which is row 200
 */
REG( 0x20000000 ) = 0x00C8DAB6;

/*Write the address of the key1 param to CPUSS_SYSARG reg */
CPUSS_SYSARG_REG = 0x20000000;

/*Write the API opcode = 0x07 to CPUSS_SYSREQ.COMMAND
 * register and assert the sysreq bit
 */
CPUSS_SYSREQ_REG = 0x80000007;

/*Execute user code until iStatusInt equals 3 to signify
 * 3 SPC interrupts have happened. This should be 1 in case
 * of non-blocking program System Call
 */
while( iStatusInt != 0x03 )
{
    DoOtherUserStuff();
}

/* Get the success or failure status of System Call*/
syscall_status = CPUSS_SYSARG_REG;
}
```

このコードでは、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 例外テーブルには、SPC にあるように定義されている *SpclntHandler()* 関数のアドレスとして、SPC 割り込みのベクタアドレスが必要です。CM0+ 例外テーブルを SRAM に配置する構成の詳細については、[割り込み \(51 ページ\)](#) を参照してください。非ブロッキングプログラムシステムコールの疑似コードも同様ですが、関数のオペコードとパラメータが異なり、iStatusInt 変数を 3 ではなく 1 にポーリングする必要がある点が異なります。これは、非ブロッキングプログラムシステムコールに対して SPC ISR が 1 回だけトリガーされるためです。

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' とするロジック信号。 2. 2 つの状態のうち、高い電圧側を論理値「1」状態とするロジック信号
active low (アクティブ LOW)	1. アサート状態を論理値「0」状態とするロジック信号 2. 2 つの状態のうち、低い電圧側を論理値「1」状態とするロジック信号：反転論理
address (アドレス)	情報ユニットを格納するメモリ位置 (RAM、ROM またはレジスタ) を識別するためのラベルや番号
algorithm (アルゴリズム)	操作の繰り返しをしばしば含む、有限個の手順で数学の問題を解くための手順
ambient temperature (周囲温度)	指定領域、特に PSoC デバイスの周囲領域の温度
analog (アナログ)	analog signals (アナログ信号) を参照してください。
analog blocks (アナログブロック)	基本的なプログラム可能なオペアンプ回路。SC (スイッチド キャパシタ) および CT (連続時間) ブロック。これらのブロックは相互接続して、ADC、DAC、多極フィルタ、ゲインステージなどを提供することができます。
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 (ADC))	アナログ信号を対応する振幅のデジタル信号に変更するデバイス。一般的に ADC は電圧をデジタル数値に変換。デジタル - アナログ変換器 (DAC) は ADC の逆の動作を行う
AND (論理積)	ブール代数を参照してください
API (Application Programming Interface) (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 アプリケーションと低レベルのサービスと関数 (ユーザ 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)	ベクタまたはリストともいわれる配列は、コンピュータ プログラミングにおける最も単純なデータ構造の 1 つ。配列は通常サイズが等しく、データ タイプも同じデータ要素の固定数を保持。連想配列とは対照的に、個別要素は連続した整数の範囲を使用してインデックスによってアクセスされる。殆どの高位プログラミング言語は配列をデータ タイプとして内蔵。一部の配列は多次元であり、即ち、固定された数の整数 (例えば 2 つの整数のグループ) によってインデックスされる。1 次元と 2 次元配列が最も一般的。また、いくつかの一般的な形で接続したコンデンサや抵抗のグループをアレイということもある。
assembly (アセンブリ)	特定プロセッサの機械語を記号化した言語。アセンブラによってアセンブリ言語を機械コードに変換。通常、アセンブリ コードの各行は 1 つの機械命令を生成するが、マクロの使用が一般的である。アセンブリ言語は低位言語と見なされるのに対して、C 言語は高位言語と見なされている
asynchronous (非同期)	どのクロック信号にも関係なく動作し、認識されるデータの信号。
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)	- 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。 - 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
バイアス	- 基準値からの意図的な偏差の値。 - 基準値と一連の値の平均値の間の誤差。 - デバイスを動作させる基準レベルを確立するために、デバイスに適用される電氣的、機械的、磁氣的、その他の力 (場) です。
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's M8CP は 8 ビットマイクロコントローラであるため、PSoC デバイスのネイティブデータチャンクサイズは 1 バイトです。
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. 発振器などの単一機能を実行する機能単位。 2. デジタル PSoC ブロックやアナログ PSoC ブロックのように、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することができる機能単位
Boolean Algebra (ブール代数)	数学やコンピュータ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: $\sim A$, $A_{\sim}$, $!A$)。
break-before-make (ブレイクビフォアメーク)	スイッチブレードが、新しい接続状態 (メーク) に入る前に、いったん遮断状態 (ブレイク) 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を通して配線され、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用のストレージ領域。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. 時にそれが外部デバイスに送信される前、または外部デバイスから受信される前に、メモリ部分がデータ格納のためにとっておかれる。 3. システムの出カインピーダンスを下げるために使用される増幅器
bus (バス)	1. 複数ラインの名前付き接続。回路どうしをバスにバンドルすることにより、類似したルーティングパターンを持つネットの配線は容易になる 2. 共通機能を実行し、同様のデータを運ぶ信号一式。一般的にベクトル表記で表される。例えば、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。容量はファラドの単位で測定される
capture (キャプチャ)	コンピュータ ファイルにデータを手入力するのに対して、ソフトウェアまたはハードウェアを使用して自動的に情報を抽出すること
chaining (連結)	2 つ以上の 8 ビット デジタル ブロックを繋いで、16 ビット、24 ビット、さらに 32 ビットの関数を形成。連結により、Compare (コンペア)、Carry (キャリー)、Enable (イネーブル)、Capture (キャプチャ) および Gate (ゲート) 等のような特定信号をあるブロックから別のブロックに転送することが可能
checksum(チェックサム)	データの個々のワードの値を総計 (sum) に加算することで、データ セットのチェックサムを生成。実際のチェックサムは、単に総計結果であるか、所定値を生成するために総計に加算しなければならない値である
clear (クリア)	ビット / レジスタの値を論理「0」にする
clock (クロック)	一定の周波数およびデューティ比で周期信号を生成するデバイス。クロックは時々、異なる論理ブロックを同期化するために使用される。
clock generator (クロックジェネレータ)	クロック信号を生成するための回路
CMOS	相補的に接続された MOS トランジスタを用いて構成された論理ゲート。CMOS は相補型金属酸化膜半導体の頭字語
comparator (コンパレータ)	2 つの入力レベルが同時に所定の振幅要件を満たすたびに、出力電圧または電流を生成する電子回路
compiler (コンパイラ)	C のような高位言語を機械言語に変換するプログラム
configuration (コンフィギュレーション)	コンピュータ 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 (CRC)) 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 1 組みの双方向信号。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) チャンネル上の信号によってデータのある場所から別の場所へ転送すること

debugger (デバグガ) ユーザ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、開発者はデバグガにより、ファームウェアを 1 段階ずつ手順を追って実行したり、ブレークポイントを設定したりメモリを分析可能

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はない期間です。

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 ー 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) カウンタやタイマ、シリアル レシーバ、シリアル トランスミッタ、CRC ジェネレータ、疑似乱数生成器、SPI として機能できる 8 ビットの論理 ブロックです。

digital logic (デジタルロジック) 回路またはシステム操作を表す 2 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 (DAC)) 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 (立ち下がリエッジ) ロジック 1 からロジック 0 への遷移。ネガティブエッジとも呼ばれます。

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) 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스에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 メモリ) 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に提供する、電氣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) 1 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つの出力チャンネルと1つ以上の入力チャンネルを持っている。そのように、出力チャンネルの状態は完全に入力チャンネルの状態によって決定される(状態切り替えの過渡期以外)。
2. 複数の組み合わせ論理要素のいずれかは、少なくとも2つの入力を持つ(例えば AND、OR、NAND および NOR (ブール代数も参照してください))

grand (接地)

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. 電氣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、コンピュータや組み込みシステムのすべての物理的な部分と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

|

I²C	Philips Semiconductors 社 (現 NXP Semiconductors 社) の 2 線式シリアル コンピュータ バス。I ² C はインター インテグレートド サーキット (内部集積回路)。組み込みシステムの低速周辺機器を接続するために使用される。オリジナル システムはバッテリー制御インターフェースとして 1980 年代初頭に作成されました。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。I ² C は、クロックとデータの 2 つの双方向ピンのみを使用します。どちらも + 5V で動作し、抵抗で High にプルされます。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 (I/O))	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どうしが接続し、互いに影響し合うための手段
発生用	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 (ISR))	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は、それぞれ独自の優先順位を持っており、個別の ISR コードブロックと共存する場合がある。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停止したポイントにデバイスを戻す

J

- jitter (ジッタ)**
1. 理想的な位置からの移行のタイミング誤配置。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
- ロード** 電力 (ワット)、電流 (アンペア) 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 (LVD))** V_{DD} を感知し、 V_{DD} 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、レジスタ空間ヘ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)	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、コントローラの寸法を低減し、コストを削減。マイクロコントローラは通常、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のだけでなく、覚えやすい構造とデータ一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 モードまたはタイマ モードのいずれかである
modulation (変調)	キャリア信号、特に正弦波信号の情報をエンコードするための技術範囲。変調を行うデバイスはいわゆる変調器と呼ばれている
Modulator (変調器)	キャリア上の信号を変換するデバイス
MOS	金属酸化膜半導体の頭字語
most significant bit (MSb) (最上位ビット)	最上位値 (一般的には、左側のビット) を表す 2 進数の桁またはビットのこと。ビットとバイトを区別するために、ビットの場合は小文字の「b」(MSb) を使用
most significant byte (MSB) (最上位バイト)	最上位値 (一般的には、左側のバイト) を表すマルチバイト文字内のバイト。ビットとバイトを区別するために、バイトの場合は大文字の「B」(MSB) を使用
multiplexer (mux) (マルチプレクサ)	1. 複数の入力から選び出して、データを選択した入力から出力へ伝送するために、2 進数の値またはアドレスを使用する論理関数。 2. 外部信号によって制御され、異なる入力 (または出力) 信号が異なる時間で同じラインを使用できるようにする技術。この多重化技術は、配線と I/O ポートを節約するために使用される

N

NAND (否定論理積)	ブール代数を参照してください
negative edge (ネガティブエッジ)	ロジック 1 からロジック 0 への遷移。立ち下がりエッジとも呼ばれます。
net (ネット)	デバイス間の配線
nibble (ニブル)	4 ビットのグループ、即ち 1 バイトの半分に相当
noise (ノイズ、雑音)	1. 信号に影響を与える、およびその信号によって運ばれた情報を歪める可能性がある妨害。 2. 電圧や電流、データなど実体の 1 つ以上の特性のランダムなばらつき
NOR (否定論理和)	ブール代数を参照してください
NOT (論理否定)	ブール代数を参照してください

O

OR (論理和)	ブール代数を参照してください
oscillator (発振器)	クロック周波数を生成するために使用される回路。水晶制御のものもある。
output (出力)	アナログ ブロックまたはデジタル ブロックによって生成された電気信号

P

parallel (パラレル)	1 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)	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 ブロック)	SSC 命令のパラメータ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です。通常、この 2 進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) になるために 2 進桁をデータに追加
path (パス)	1. コンピュータ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)	ロジック 0 からロジック 1 への遷移。立ち上がりエッジとも呼ばれます。
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)	命令ポインタ (プログラム カウンタとも呼ばれている) 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 プロセッサのレジスタ。特定マシンの詳細に応じて、実行されている命令のアドレスまたは次に実行すべき命令のアドレスを保存
protocol (プロトコル)	規約の集合。特にネットワーク上の通信を監視するルール
PSoC®	サイプレスのプログラマブル システムオンチップ (PSoC®) デバイス
PSoC blocks (PSoC ブロック)	アナログ ブロックおよびデジタル ブロックを参照してください
PSoC Creator™	サイプレスの次世代のプログラマブル システムオンチップ技術のソフトウェア
pulse (パルス)	信号のいくつかの特性 (例えば位相や周波数) が、ベースライン値から急峻に変化し、その後ベースライン値に急峻に復帰すること
pulse width modulator (PWM) (パルス幅変調器)	関数としてデューティ比を変化させる出力

R

RAM	ランダム アクセスメモリ (random access memory) の頭字語。データを読み出ししたり、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。クロック信号はカウンタの第 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 進数カウンタは、2 進数で 0 から $2^n - 1$ までカウントすることが可能
rising edge (立ち上がりエッジ)	ポジティブ エッジを参照してください
ROM	読み出し専用メモリ (read only memory) の頭字語。データを読み出すことはできるが、新しいデータを書き込むことはできないデータ ストレージ デバイス
routine (ルーチン)	共有または頻繁に使用されている別のコード ブロックによって呼び出されるコード ブロック
routing (配線)	参照ライブラリで設定された設計ルールに従って、設計におけるオブジェクトを物理的に接続すること
runt pulses (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)	- すべてのイベントが相次いで発生するプロセスを示します。 - 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 (整定時間)	入力に変化した後に、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 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 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) 実際の搭載。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)	カウンタが 0 までカウントする状態
threshold	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	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トランスミッタは、データの平行ビットとシリアルビット間での変換を行う
user (ユーザ)	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 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 モジュールは周辺機能に高レベルの 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} 「電圧ドレイン」を意味するパワーネットの名前。最も正の電源信号。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} 「電圧源」を意味する電力網の名前。最も負の電源信号。

W

WatchDog Timer (ウォッチドッグ タイマ) 定期的に点検整備される必要があるタイマ。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 信号を振幅対時間のプロットとして表現したもの

X

XOR (排他的論理和) ブール代数を参照してください