# ModusToolbox™用 PSOC™ 4 IEC 60730 クラス B および IEC 61508 SIL セーフティソフトウェアライブラリ

## 本書について

### 適用範囲と目的

このアプリケーション ノートは、PSOC™ 4 MCU IEC 60730 クラス B および IEC 61508 の安全完全性レベル (SIL) に対応したセーフティソフトウェアライブラリを、ModusToolbox™およびペリフェラルドライバライブラリ (PDL) とともに使用する方法について説明します。AN236847 には、PSOC™ 4100S/4100S Max/4100S Plus 256KB/4500S デバイスファミリのセーフティライブラリソースコードおよびサンプルプロジェクトが含まれており、信頼性と安全な動作を確保するためのセルフチェックルーチンを備えています。サンプルプロジェクトに含まれているライブラリルーチンとサンプルを、ご使用のアプリケーションに組み込むことができます。このアプリケーションノートでは、ライブラリで使用可能な API 関数についても説明します。

### 対象者

このドキュメントの対象者は、電子システムの設計エンジニア、技術者および、開発者です。

本書は、ModusToolbox™ソフトウェアで PSOC™ 4 ソフトウェアライブラリを使用するすべてのユーザーを対象とします。

# 目次

**本書について ..... 1**
**目次 ..... 2**
**1 はじめに ..... 4**
**2 IEC 60730-1 Annex H の概要 ..... 5**
**3 IEC 61508-2 Annex A の概要 ..... 7**
**4 IEC 60730 クラス B および IEC 61508 の要件 ..... 8**
**5 セーフティライブラリの使用法 ..... 10**
**6 PSOC™ 4 の API 関数 ..... 11**
6.1 CPU レジスタテスト ..... 14
6.2 プログラムカウンターテスト ..... 14
6.3 プログラムフローテスト ..... 16
6.4 割込み処理と実行テスト ..... 17
6.5 クロックテスト ..... 19
6.6 フラッシュ (不変メモリ) テスト ..... 20
6.6.1 Fletcher チェックサム法 ..... 21
6.6.2 CRC-32 法 ..... 21
6.6.3 プログラミングステップ ..... 21
6.7 SRAM (可変メモリ) テスト ..... 22
6.7.1 マーチ C テスト ..... 22
6.7.2 マーチ C アルゴリズム ..... 23
6.7.3 テスト時間 ..... 25
6.8 スタックオーバーフローテスト ..... 25
6.9 デジタル I/O テスト ..... 26
6.10 アナログテスト ..... 27
6.10.1 リファレンス電圧の生成 ..... 27
6.10.2 ADC テスト ..... 28
6.10.3 コンパレータテスト ..... 29
6.10.4 オペアンプテスト ..... 30
6.11 UART 通信テスト ..... 32
6.12 通信 SPI テスト ..... 33
6.13 I2C 通信テスト ..... 34
6.14 スタートアップ コンフィギュレーションレジスタ テスト ..... 35
6.15 ウォッチドッグテスト ..... 37
6.16 TCPWM テスト ..... 37
6.16.1 タイマー/カウンター テスト ..... 38
6.16.2 PWM テスト ..... 38
6.16.3 PWM Gate Kill テスト ..... 39
6.17 CAPSENSE™ テスト ..... 39
6.18 通信 UART データ転送プロトコルの例 ..... 40
6.18.1 データ配信制御 ..... 41
6.18.2 PSOC™の実装 ..... 41
**7 テストファイルの一覧 ..... 46**
**8 まとめ ..... 48**
**9 Appendix A: フラッシュ (不変メモリ) テストのチェックサムの設定 ..... 49**
**10 Appendix B: IEC 60730-1 準拠証明書 ..... 51**
**11 Appendix C: サポートされている製品番号 ..... 61**

目次

12 Appendix D. MISRA 準拠 ..... 62
参考資料 ..... 64
改訂履歴 ..... 65
免責事項 ..... 66

# 1 はじめに

現在、家電製品や工業製品の自動電子制御の大部分は、シングルチップのマイコンユニット (MCU) を使用しています。メーカーは、MCU で実行され、家電製品や産業機械を制御するための隠れたインテリジェンスを提供するリアルタイムの組み込みファームウェアを開発します。過熱, 静電放電, 過電圧, またはその他の要因による MCU の損傷により、最終製品が制御不能または安全でない状態になる可能性があります。

International Electrotechnical Commission (IEC) 60730-1 安全規格は、家電製品の機械的, 電気的, 電子的, 環境的耐久性, EMC, および異常な動作について説明しています。IEC 61508 は、工業設計における電気/電子/プログラマブル電子 (E/E/PE) 安全関連システムの要件を詳述しています。両方の仕様のテスト要件は類似しており、このドキュメントとセーフティソフトウェアライブラリで説明されています。

このアプリケーションノートでは、IEC 60730-1 の Annex H「電子制御の要件」と IEC 61508-2 の Annex A「E/E/PE 安全関連システムの技術と対策:運転中の故障の制御」に焦点を当てています。これらのセクションでは、家電製品や産業機械用の組込み制御ハードウェアおよびソフトウェアの安全な動作を促進するテストと診断方法について詳しく説明します。

# 2 IEC 60730-1 Annex H の概要

IEC 60730-1 規格の Annex H は、家電製品ソフトウェアを次のカテゴリに分類しています。

- クラス A 制御機能は、機器の安全性に依拠することを意図したものではありません。例としては、湿度制御, 照明制御, タイマー, およびスイッチがあります。
- クラス B 制御機能は、制御された機器の安全でない操作を防ぐことを目的とします。例としては、洗濯機のサーマルカットオフやドアロックがあります。
- クラス C 制御機能は、特別な危険 (制御された機器によって引き起こされる爆発など) を防ぐことを目的とします。例としては、自動バーナー制御や、密閉された通気孔のない給湯器システム用のサーマルカットアウトがあります。

洗濯機, 食器洗い機, 乾燥機, 冷蔵庫, 冷凍庫, 調理用加熱機器などの大型家電製品は、クラス B に分類される傾向があります。例外は、ガスを使用した乾燥機など、爆発を引き起こす可能性のある器具で、クラス C に分類されます。

このアプリケーションノートで紹介するクラス B セーフティソフトウェアライブラリとサンプルプロジェクトは、クラス B カテゴリで規定されている自己テストおよび自己診断方法を実装しています。これらは、ソフトウェア関連の障害やエラーを検出し、それらに対応するためのさまざまな手段を提供します。IEC 60730-1 規格によると、自動電子制御のメーカーは、次のいずれかの構造を使用してクラス B ソフトウェアを設計する必要があります。

- 機能テストを備えたシングルチャネル
- 定期的なセルフテストを備えたシングルチャネル
- 比較のないデュアルチャネル (Figure 1 を参照)

機能テストによるシングルチャネル構造では、ソフトウェアは単一の CPU を使用して必要な機能を実行するように設計されます。機能テストは、アプリケーションの起動後に実行され、すべての重要な機能が確実に機能していることを確認します。

定期的なセルフテストによるシングルチャネル構造では、ソフトウェアは単一の CPU を使用して必要な機能を実行するように設計されます。定期的なテストはソフトウェアに組み込まれており、セルフテストはソフトウェアが実行モードのときに定期的に行われます。CPU は、電子制御の重要な機能を定期的にチェックすることが期待されており、エンドアプリケーションの動作と競合することはありません。

比較構造のないデュアルチャネル構造では、ソフトウェアは 2 つの CPU を使用して重要な機能を実行するように設計されています。重要な機能を実行する前に、両方の CPU が対応するタスクを完了したことを共有する必要があります。例えば、ランドリーのドアロックが解除されると、Figure 1 に示すように、一方の CPU がドラムを回転させるモーターを停止し、もう一方の CPU がドラムの速度をチェックしてドラムが停止したことを確認します。

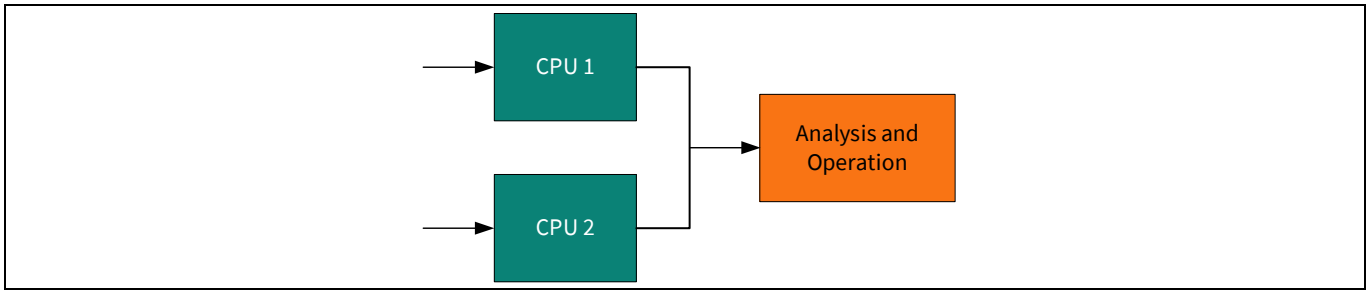


**Figure 1 比較構造のないデュアルチャネル**

### IEC 60730-1 Annex H の概要

デュアルチャネル構造の実装は、2 つの CPU (または 2 つの MCU) が必要なため、コストが高くなります。さらに、2 つのデバイスが定期的に相互に通信する必要があるため、より複雑になります。定期的なセルフテストを備えたシングルチャネル構造が最も一般的な実装です。

# 3 IEC 61508-2 Annex A の概要

IEC 61508-2 Annex A は、産業用設計の関連技術および対策に対して対応できる最大診断範囲を定義します。このライブラリに記載されていない追加要件は、鉄道, プロセス制御, 自動車, 原子力, 機械などの特定の業界に適用される場合があります。各安全度水準 (SIL) について、付録では、ランダムなハードウェア, システム, 環境, および運用上の障害を制御するための手法と対策を推奨します。アーキテクチャと対策の詳細については、IEC 61508-6 の Annex B および IEC 61508-7 の Annex A を参照してください。

このような障害が発生したときに回避または制御するには、通常、いくつかの対策が必要です。IEC 61508 Annex A および B の要件は、E/E/PE システムの安全ライフサイクルのさまざまなフェーズで障害を回避するための対策 (Annex B) と、運用中の障害を制御するための対策 (Annex A) に分かれます。故障を制御するための対策は、E/E/PE 安全関連システムに組み込まれる機能です。

このプロセスは、制御された機器の各危険なイベントのリスクを評価することから始まります。通常、診断範囲と安全故障率は、各故障が発生する可能性と故障の結果に基づいて決定されます。これは例えば、リモートの壊滅的な障害が、頻繁に発生する無視できる障害と同様のリスクを持つように、リスクを重み付けします。リスク評価の結果は、エンドシステムの要件となるターゲット SIL です。

SIL レベルの意味は、デバイスの操作頻度によって異なります。ほとんどのデバイスは、年に 1 回以上使用されるため、「需要が高い」に分類されます。高い需要レベルの SIL レベルで、使用時間あたりの危険な故障が発生する確率は次のとおりです。

- **SIL 1:** $\geq 10^{-6} \sim < 10^{-5}$ (11 年間で 1 回の失敗)
- **SIL 2:** $\geq 10^{-7} \sim < 10^{-6}$ (114 年間で 1 回の失敗)
- **SIL 3:** $\geq 10^{-8} \sim < 10^{-7}$ (1,144 年間で 1 回の失敗)
- **SIL 4:** $\geq 10^{-9} \sim < 10^{-8}$ (11,446 年間で 1 回の失敗)

# 4 IEC 60730 クラス B および IEC 61508 の要件

IEC 60730-1 Class B Annex H Table H.11.12.7 および IEC 61508-2 Annex A Tables A.1～A.14 によると、ソフトウェアの分類に応じて、特定のコンポーネントをテストする必要があります。一般に、各コンポーネントは、対応するコンポーネントを検証またはテストするためのオプションの手段を提供し、メーカーに柔軟性を提供します。

シングルチャネル構造のクラス B IEC 60730 および IEC 61508 に準拠するために、電子制御のメーカーは Table 1 にリストされているコンポーネントをテストする必要があります。

**Table 1 シングルチャネル構造の試験が必要なコンポーネント**

| 電子制御でテストする必要があるクラス B IEC 60730 コンポーネント (Annex H の表 H.11.12.7) | テストが必要な IEC61508 コンポーネント (Annex A の表 A.1～A14) | フォールト/エラー |
|---|---|---|
| 1.1 CPU レジスタ | A.4, A.10 CPU レジスタ | スタック |
| 1.3 CPU プログラムカウンター | A.4, A.10 プログラムカウンター | スタック |
| 2. 割込み処理と実行 | A.4 割込み処理 | 割込みなし、または割込みが多すぎる |
| 3. クロック | A.11 クロック | 不正な周波数 |
| 4.1 不変メモリ | A.5 不変メモリ | すべてのシングルビットフォールト |
| 4.2 可変メモリ | A.6 可変メモリ | DC フォールト |
| 4.3 アドレス指定 (可変/不変メモリに関連) | A.4, A.10 アドレス計算 | スタック |
| 5.1 内部データパスデータ | A.8 データパス (内部通信) | スタック |
| 5.2 内部データパスアドレス指定 (拡張メモリ MCU システムのみ) | – | 不正なアドレス |
| 6.1 外部通信データ | A.7 I/O ユニットとインターフェース | ハミング距離 3 |
| 6.2 外部通信のアドレス指定 | A.7 I/O ユニットとインターフェース | ハミング距離 3 |
| 6.3 タイミング | – | 時間/シーケンスでの不正なポイント |
| 7.1 I/O 周辺機器 | A.7 I/O ユニットとインターフェース | Appendix B「IEC 60730-1、H.27」で指定されているフォールト状態 |
| 7.2.1 アナログ A/D および D/A コンバーター | A.3 アナログ信号監視 | Appendix B「IEC 60730-1、H.27」で指定されているフォールト状態 |
| 7.2.2 アナログ マルチプレクサ | – | 不正なアドレス指定 |

ユーザーアプリケーションは、クラス B 安全ソフトウェアライブラリの実行中に割込みを有効または無効にする必要があるかを決定する必要があります。例えば、CPU セルフテスト ルーチンの実行中に割込みが発生した場合、どのレジスタでも予期しない変更が発生する可能性があります。そのため、割込みサービスルーチン (ISR) を実行すると、レジスタの内容が期待値と一致しなくなります。

## IEC 60730 クラス B および IEC 61508 の要件

クラス B 安全ソフトウェアライブラリのサンプルプロジェクトは、正しいセルフテストのために割込みを無効にして有効にする必要がある場所を示します。

# 5 セーフティライブラリの使用法

このアプリケーションノートで説明するセーフティソフトウェアライブラリは、PSOC™ 4 MCU デバイスで使用できます。API も含まれて、フォールト検出機能によりアプリケーションの信頼性を最大化します。インフィニオンは、クラス B 標準規格に準拠して開発された mtb-stl ミドルウェア、およびサンプルコードをインフィニオン GitHub リポジトリで提供しています。

*Note:* *mtb-stl ライブラリの現在のバージョン(v3.0.0) と上記のサンプルコードは、まだクラス B の認証がされていません。クラス B の認証は、2025 年半ばまでに行われます。クラス B および mtb-stl 認証がされたバージョンのライブラリを使用する場合は、AN236847 に掲載されているライブラリおよびサンプル プロジェクトを利用してください。*

対応デバイスの STL 認証とクラス B 認証により、お客様は安全に関わるシステムの認証を取得できます。一部のセルフテストは、クラス B 安全ソフトウェア ライブラリの*.c* ファイルと*.h* ファイルに適切な API 関数を追加するだけで適用できます。

このアプリケーションノートでは、次の 2 種類のセルフテスト機能について説明し、実装します。

- IEC 60730-1 Class B および IEC 61508-2 規格への適合に役立つセルフテスト機能。
  - **CPU レジスタ:** スタックビットのテスト
  - **プログラムカウンター:** 正しいアドレスへのジャンプテスト
  - **プログラムフロー:** 正しいファームウェアプログラムフローを確認するためのテスト
  - **割込みの処理と実行:** 適切な割込み呼び出しと周期性のテスト
  - **クロック:** 不正な周波数をテスト
  - **フラッシュ (不変メモリ):** メモリ破損のテスト
  - **SRAM (可変メモリ):** スタックビットと適切なメモリアドレス指定のテスト
  - **スタックオーバーフロー:** プログラム実行中にプログラムデータメモリのスタックオーバーフローを確認するためのテスト
  - **デジタル I/O:** ピン間短絡テスト
  - **ADC と DAC:** 適切な機能テスト
  - **コンパレータ:** 適切な機能テスト
  - **通信 (UART, SPI, I2C):** データ受信テスト
- PSOC™ 4 MCU がプログラム可能な相互接続によりサポートできる追加のセルフテスト機能。多くの場合、エンドアプリケーションには、IEC 60730-1 または IEC 61508-2 の Annex B に規定されていない場合でも、これらのセルフテストも必要です。
  - オペアンプテスト
  - スタートアップ時のコンフィギュレーションレジスタのテスト
  - チップリセット用ウォッチドッグタイマのテスト
  - 適切な機能と外部/内部ハードウェアの損傷を確認する CAPSENSE™テスト

すべてのセルフテストは、デバイスの起動直後に一度実行でき、デバイス動作中に継続的に実行できます。起動時にセルフテストを実行すると、アプリケーションのファームウェアを実行する前に、チップが動作に適しているかどうかを判断する機会が得られます。通常の動作中に実行されるセルフテストにより、継続的な損傷検出とユーザー定義の是正措置が可能になります。

次のセクションでは、各テストの実装の詳細について説明し、対応するテストの実行に必要な API をリストします。

# 6 PSOC™ 4 の API 関数

次の表に、積分器が処理する必要があるテスト関数と戻り値を示します。

*Note:* *このアプリケーションノートは、まだクラス B 認証を受けていない mtb-stl クラス B ライブラリの最新バージョンをサポートするために更新しました。以下の API は最新の mtb-stl ライブラリ (v3.0.0) のものです。クラス B 認定バージョンのライブラリとサンプルを使用したい場合は、AN236847 で利用可能なライブラリとサンプルプロジェクトを使用してください。*

| テストファイル名 | テスト関数 | 戻り値 |
|---|---|---|
| SelfTest_Analog.c | `SelfTest_ADC` | 0: テストは合格しました<br>1: テストに失敗しました |
| | `SelfTests_Opamp` | 0: テストは合格しました<br>1: テストに失敗しました |
| | `SelfTests_Comparator` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_Clock.c | `SelfTest_Clock` | 1: テストに失敗しました<br>2: テスト中<br>3: テストは完了しました<br>4: 誤った使い方 |
| SelfTest_ConfigRegisters.c | `SelfTests_StartUp_ConfigReg` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_CPU.c | `SelfTest_CPU_Registers` | 0: テストは合格しました<br>1: テストに失敗しました |
| | `SelfTest_PC` | 0: テストは合格しました<br>1: テストに失敗しました |
| | `SelfTest_PROGRAM_FLOW` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_Flash.c | `SelfTest_FlashCheckSum` | 1: テストに失敗しました<br>2: テスト中<br>3: テストは完了しました |
| SelfTest_Interrupt.c | `SelfTest_Interrupt` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_IO.c | `SelfTest_IO` | 0: テストは合格しました<br>1: テストに失敗しました (VCC への短絡)<br>1: テストに失敗しました (GND への短絡) |

| テストファイル名 | テスト関数 | 戻り値 |
|---|---|---|
| SelfTest_RAM.c | `SelfTests_SRAM_March` | 1: テストに失敗しました<br>2: テスト中<br>3: テストは完了しました |
| | `SelfTests_Stack_March` | 1: テストに失敗しました<br>2: テスト中<br>3: テストは完了しました |
| SelfTest_Stack.c | `SelfTests_Stack_Check_Range` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_FPU_Regs.c | `SelfTest_FPU_Registers` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_IPC.c | `SelfTest_IPC` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_WDT.c | `SelfTest_WDT` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_DMAC.c | `SelfTest_DMAC` | 0: テストは合格しました<br>1: テストに失敗しました (VCC への短絡)<br>1: テストに失敗しました (GND への短絡) |
| SelfTest_DMA_DW.c | `SelfTest_DMA_DW` | 0: テストは合格しました<br>1: テストに失敗しました (VCC への短絡)<br>1: テストに失敗しました (GND への短絡) |
| SelfTest_Timer_Counter.c | `SelfTest_Counter_Timer` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_PWM.c | `SelfTest_PWM` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_PWM_GateKill.c | `SelfTest_PWM_GateKill` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_CANFD.c | `SelfTest_CANFD` | 0: テストは合格しました<br>1: テストに失敗しました |
| SelfTest_UART_SCB.c | `SelfTest_UART_SCB` | 1: エラーが検出されました<br>2: テスト中<br>3: 合格、テストは完了しました<br>4: TX は空ではありません<br>5: RX は空ではありません<br>6: TX が有効になっていません<br>7: RX が有効になっていません |

| テストファイル名 | テスト関数 | 戻り値 |
|---|---|---|
| SelfTest_UART_master_message.c | `UartMesMaster_Init` | 無 |
| | `UartMesMaster_DataProc` | 0: テストは合格しました<br>1: テストに失敗しました |
| | `UartMesMaster_State` | '0': 最後のトランザクションプロセスが正常に終了し、受信したバッファに応答が含まれています。ユニットは新しいプロセスを開始する準備ができています。<br>'1': 最後のトランザクションプロセスがエラーで終了し、ユニットは新しいプロセスを開始する準備ができています。<br>'2': ユニットはアクティブなトランザクション操作でビジー状態です。 |
| | `UartMesMaster_GetDataSize` | バッファ内の受信データ サイズ |
| SelfTest_UART_slave_message.c | `UartMesSlave_Init` | 無 |
| | `UartMesSlave_Respond` | '0': テストは合格しました<br>'1': テストに失敗しました |
| | `UartMesSlave_State` | '0': 最後のトランザクション プロセスが終了しました。<br>'1': ユニットはマーカーを受け取り、バッファに受信したデータがあります。マスターは応答を待ちます。<br>'2': ユニットは応答の送信でビジー状態です。 |
| | `UartMesSlave_GetDataPtr` | 受信したデータ バッファへのポインターを返します。 |
| SelfTest_SPI_SCB.c | `SelfTest_SPI_SCB` | '1': テストに失敗しました<br>'2': テスト中<br>'3': テストは完了しました<br>'4': TX は空ではありません<br>'5': RX は空ではありません |
| SelfTest_I2C_SCB.c | `SelfTest_I2C_SCB` | '1': テストに失敗しました<br>'2': テスト中<br>'3': テストは完了しました |

## 6.1 CPU レジスタテスト

Arm®Cortex®-M0 または Cortex® -M0+ CPU を搭載した PSOC™ 4 には、16 ビットと 32 ビットのレジスタがあります。

- **R0 ～ R12:** 汎用レジスタ
- **R13 – スタックポインター (SP):** スタックポインターは 2 つあり、一度に 1 つだけ使用できます。SP は常に 32 ビットワードにアラインされます。ビット [1:0] は常に無視され、'0' と見なされます。
- **R14 –リンクレジスタ:** このレジスタは、関数呼び出し中のリターンプログラムカウンターを格納します。
- **R15 –プログラムカウンター:** このレジスタは、プログラムフローを制御するために書き込むことができます。

CPU レジスタテストでは、チェッカーボードテストを使用して、CPU レジスタのスタックアットフォールトを検出します。このテストでは、レジスタのビットが値 '0' または '1' でスタックしていないことを確認します。これは、次の主要なタスクを実行する非破壊テストです。

1. テスト対象の CPU レジスタの内容は、ルーチンを実行する前にスタックに保存されます。
2. レジスタは、バイナリシーケンスを連続して書き込むことによってテストされます 01010101 その後に 10101010 をレジスタに書き込み、これらのレジスタから値を読み出して検証します。
3. 戻り値が一致しない場合、テストはエラーコードを返します。

チェッカーボード方式は、プログラムカウンターを除くすべての CPU レジスタに実装されます。

**関数**

```
uint8 SelfTest_CPU_Registers(void)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

ロケーション:

- *SelfTest_CPU.c*
- *SelfTest_CPU.h*

関数 `SelfTest_CPU_Registers` は、CPU テストを行うために呼び出されます。

エラーが検出された場合、PSOC™ デバイスの動作は予測不可能であり、安全でない可能性があるため、動作し続けるべきではありません。

## 6.2 プログラムカウンターテスト

PSOC™ 4 CPU プログラムカウンター R15 レジスタは、CPU レジスタセットの一部です。これらのレジスタをテストするには、チェッカーボードテストが一般的に使用されます。アドレス 0x555...および 0xAAA...をこのテストに割り当てる必要があります。0x555...および 0xAAA...はチェッカーボードのビットパターンを表します。使用されるアドレスは、デバイスのフラッシュサイズによって異なり、チェッカーボードパターンに続くフラッシュ内の最大アドレスで、 4 バイトアラインである必要があります。

プログラムカウンター (PC) テストは、IEC 60730 規格のセクション H.2.16.5 で定義されている機能テストを実装します。PC は、次に実行される命令のアドレスを保持します。テストでは、次の主要なタスクが実行されます。

1. フラッシュメモリ内の異なるアドレスにある関数が呼び出されます。PSOC™ 4 の場合、これは GCC では **.ld* ファイルのリンカースクリプトを使用し、IAR では **.icf* ファイルのリンカースクリプトを使用して実行できます。GCC コンパイラの場合、これらは次のように提供されるカスタムリンカーファイルにあります。

```
NV_CONFIG2 0x15554 :
{
    . = 0x00;
    KEEP (*(PC5555))
} >flash =0

NV_CONFIG3 0x2AAA8 :
{
    . = 0x00;
    KEEP (*(PCAAAA))
} >flash =0
```

この GCC リンカースクリプトの例は、フラッシュサイズが 0x40000 の場合、4 バイトアラインの場合の最大チェッカーボードパターンを示します。サンプルプロジェクトでは、GCC *linker.ld* ファイルに既に追加されています。リンカーファイルを追加するには、リンカースクリプトへのパスを `LINKER_SCRIPT` Makefile 変数に指定します。

2. これらの関数はユニークな値を返します。
3. 戻り値は、PC テスト機能を使用して検証されます。
4. 値が一致すると、PC は正しい場所に分岐するか、プログラムの実行が範囲外であるために WDT によってリセットがトリガーされます。

### 関数

```
uint8  SelfTest_PC(void)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

ロケーション:

- *SelfTest_CPU.c*
- *SelfTest_CPU.h*

関数 `SelfTest_PC()` が呼び出されて PC テストが実行されます。

*Note:* *PSOC™ 4 の場合は、Table 2 にリストされている対応するファイルをリンカーとして追加します。*

Table 2 に、さまざまなサイズのデバイス用のリンカースクリプトの例をリストします。これらのリンカースクリプトは、次の形式でクラス B ライブラリにパッケージ化されています。 *psoc4-class-b-library/Linker/{TOOLCHAIN}/{BSP}*

**Table 2　　カスタム リンカーファイル**

| PSOC™ファミリ | フラッシュサイズ (KB) | カスタム リンカーファイル |
|---|---|---|
| PSOC™ 4100S | 64 | *.../TOOLCHAIN_GCC_ARM/TARGET_CUSTOM_BSP_4100S/linker.ld*<br>*.../TOOLCHAIN_IAR/TARGET_CUSTOM_BSP_4100S/linker.icf* |
| PSOC™ 4100S Max | 384 | *.../TOOLCHAIN_GCC_ARM/TARGET_CY8CKIT-041S-MAX/linker.ld*<br>*.../TOOLCHAIN_IAR/ TARGET_CY8CKIT-041S-MAX /linker.icf* |
| PSOC™ 4100S Plus 256KB | 256 | *.../TOOLCHAIN_GCC_ARM/ TARGET_CUSTOM_BSP_4100S-PLUS-256K/linker.ld*<br>*.../TOOLCHAIN_IAR/ TARGET_CUSTOM_BSP_4100S-PLUS-256K /linker.icf* |
| PSOC™ 4500S | 256 | *.../TOOLCHAIN_GCC_ARM/TARGET_CY8CKIT-045S/linker.ld*<br>*.../TOOLCHAIN_IAR/ TARGET_CY8CKIT-045S /linker.icf* |

*Note:*　　*各デバイスファミリのリンカーファイルは、対応するファミリのデフォルトデバイスに基づいて作成されます。フラッシュサイズがデフォルトデバイスと異なる場合は、リンカーファイルを更新します。*

## 6.3　　プログラムフローテスト

プログラムの実行フローを確認するために使用できる特定の方法を示します。すべての重要な実行コードブロックについて、ブロックの実行前と実行直後に、ユニークな番号が補完的なカウンターに加算または減算されます。これらの手順により、コードブロックがメインプログラムフローから正しく呼び出されているかどうかを確認し、ブロックが正しく実行されているかどうかを確認できます。これはスタンドアロンテストではないため、テストライブラリには含まれません。これは、最終アプリケーションに統合するのに最適であり、ライブラリにスタンドアロンテストとしては含まれません。

常に同じ数の出口と入口のポイントがある限り、カウンターペアはテストされた各ブロックの後に常に補完的になります。Figure 2 を参照してください。

予期しない値は、プログラムフロー実行エラーとして扱う必要があります。

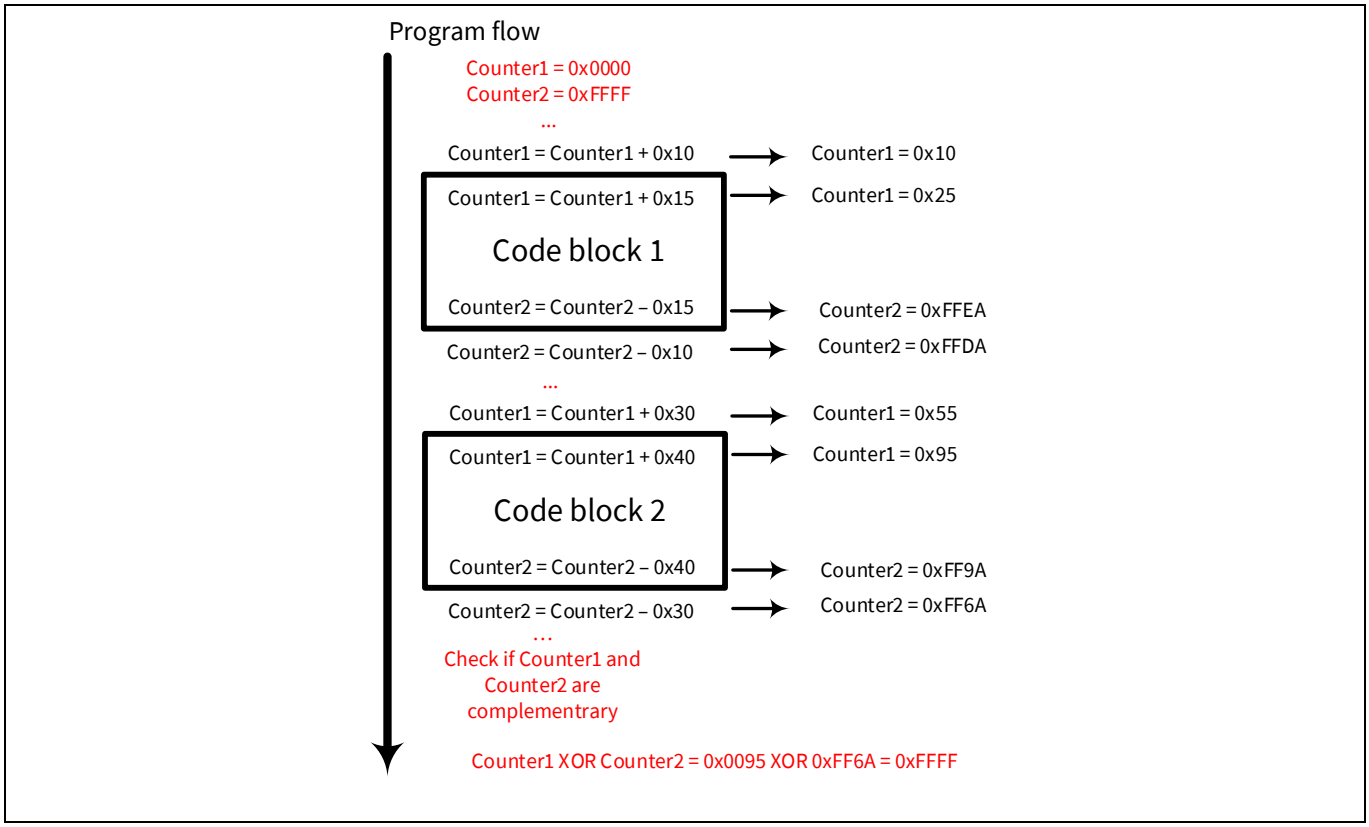


**Figure 2** プログラムフローテスト

## 6.4 割込み処理と実行テスト

PSOC™ 4 割込みコントローラは、ハードウェアリソースが、メインコードによって実行中の現在のタスクと無関係に、プログラムの実行を新たなアドレスに変更するためのメカニズムを提供します。また、ISR の完了後に中断されたコードの継続も処理します。

割込みテストは、IEC 60730 規格のセクション H.2.18.10.4 で定義されている独立したタイムスロット監視を実装します。発生した割込みの数が事前定義された範囲内にあるかどうかを確認します。このテストは、ユーザーアプリケーションの ISR とタイミングが使える様に調整する必要があります。

割込みテストの目的は、割込みが定期的に発生することを確認することです。このテストでは、タイマー UM によって駆動される割込みソースを使用して、割込みコントローラをチェックします。

**関数**

```
uint8 SelfTest_Interrupt(TCPWM_Type *base, uint32_t cntNum)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

パラメータ:

- `base` – TCPWM インスタンスへのポインター
- `cntNum` – 選択した TCPWM のカウンター インスタンス番号

ロケーション:

- *SelfTest_Interrupt.c*
- *SelfTest_Interrupt.h*

*Note:* *このテストでは全体の割込みは許可する必要がありますが、isr_1 以外はディセーブルにしなければいけません。*

`SelfTest_Interrupt()` 関数を呼び出して、割込みコントローラの動作を確認します。Selftest は、任意の TCPWM インスタンスを使用できるようにパラメータ化されています。関数を呼び出すと、タイマーが開始されます。

CYBSP_TIMER_HW は、1 ミリ秒あたり 12 個の割込みを生成するように設定されています。SelfTest_Interrupt_ISR_TIMER は、発生した割込みの数をカウントします。SelfTest_Interrupt_ISR_TIMER のカウント値が ≥9 かつ ≤15 の場合、テストは合格です。このテストに合格するための割込みの具体的な数はアプリケーションによって異なり、必要に応じて変更できます。

Figure 3 に、割込みセルフテストのフローチャートを示します。

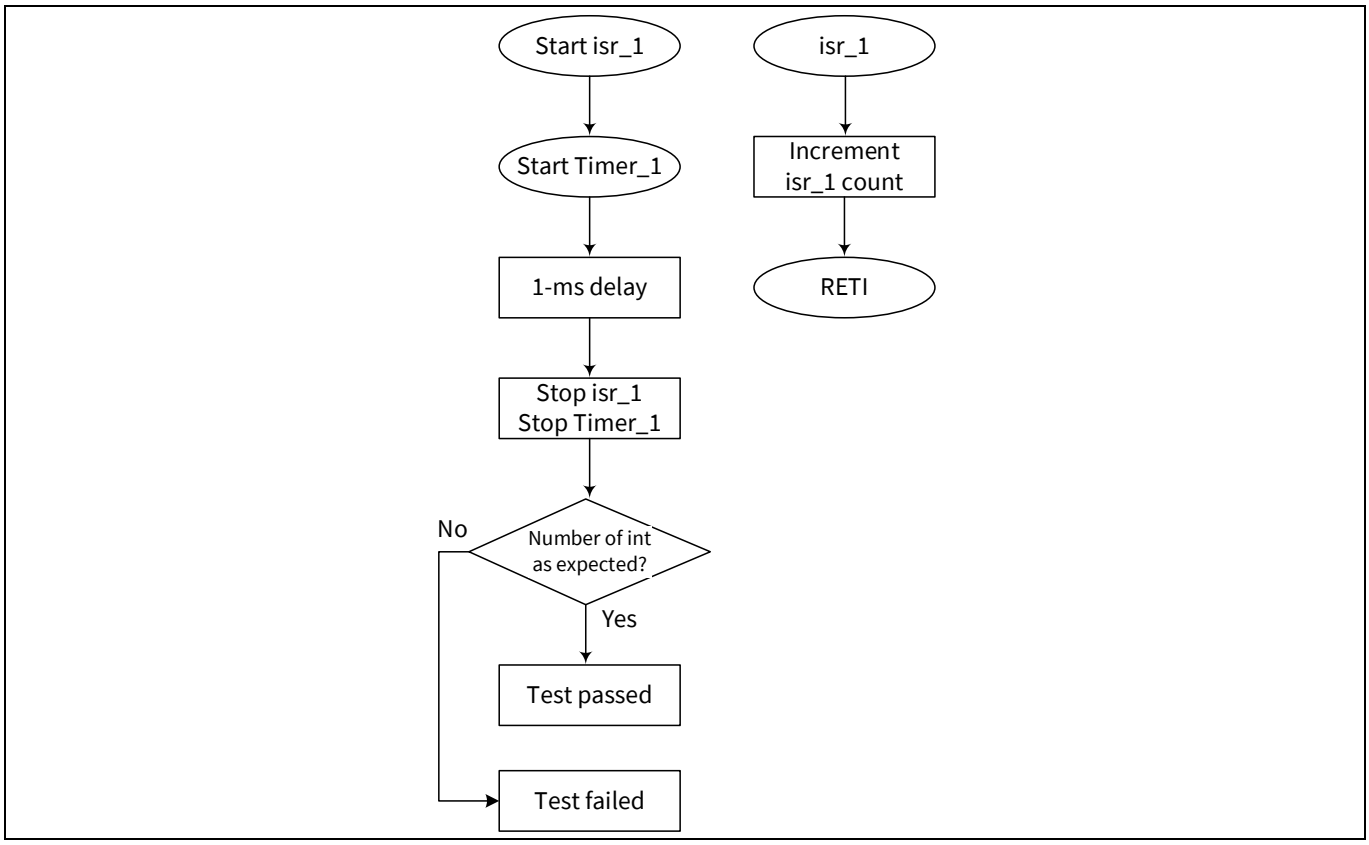


**Figure 3　　割込みセルフテストのフローチャート**

*Note:* *上の図では、"SelfTest_Interrupt_ISR_TIMER" は "isr_1" に名前が変更されています。*

## 6.5 クロックテスト

クロックテストは、IEC 60730 規格のセクション H.2.18.10.4 で定義されている独立したタイムスロット監視を実装します。これは、内部メイン発振器 (IMO) システムクロックの信頼性、特にシステムクロックが内部低速発振器 (ILO) の許容範囲内で速すぎず遅すぎないことを検証します。ILO クロックの精度は-50/+100%です。これを超える確度が必要な場合は、高精度のシステムレベルの信号または製造テストを使用して、ILO をより正確にトリミングできます。ILO トリミングが必要な場合は、CLK_ILO_TRIM レジスタを使用してトリミングします。

**関数**

```
uint8 SelfTest_Clock(TCPWM_Type * base, uint32_t cntNum)
```

戻り値:

- '1' – エラーが検出されました
- '2' – テスト中
- '3' – テストが終了し、正常に完了しました
- '4' – 誤った使い方

パラメータ:

- `base` – TCPWM インスタンスへのポインター
- `cntNum` – 選択した TCPWM のカウンター インスタンス番号

ロケーション:

- *SelfTest_Clock.h*
- *SelfTest_Clock.c*

*Note:* *テストされたクロック精度は、SelfTest_Clock.h file で定義されています。クロック精度は、エンドシステム要件に基づいて変更される場合があります。*

クロックテストでは、WDT に統合された 40 kHz ILO によってクロックされる 16 ビットタイマー0 を使用します。WDT タイマー0 は、オーバーフローを伴う連続アップカウント 16 ビットタイマーです。テストは、タイマーの現在のカウントを読み出すことから開始し、次に IMO によってクロックされる TCPWM タイマー遅延を使用して 1 ミリ秒待機し、最後にタイマーカウントを 2 回目に読み出します。次に、2 つのカウント値が減算され、必要に応じて、テスト中のタイマーオーバーフローの特殊なケースに対して修正されます。次に、測定された期間 (ノミナル 33 カウント) がテストされます。事前定義された範囲内にある場合、テストは合格です。Figure 4 に、クロックセルフテストのフローチャートを示します。

セルフテストは、複数回呼び出されるように設計されます。`Selftest_Clock()`への最初の呼び出しは、TCPWM タイマーを設定し、初期 WDT カウントを取得します。後続の呼び出しでは、TCPWM タイマーの ISR がトリガーされたかどうかがチェックされます。ISR が 1 回呼び出されたことが検出されると、最初のセルフテスト関数呼び出しの WDT 値と ISR の WDT 値を比較します。このテスト方法により、必要に応じてセルフテストをバックグラウンドで実行できます。

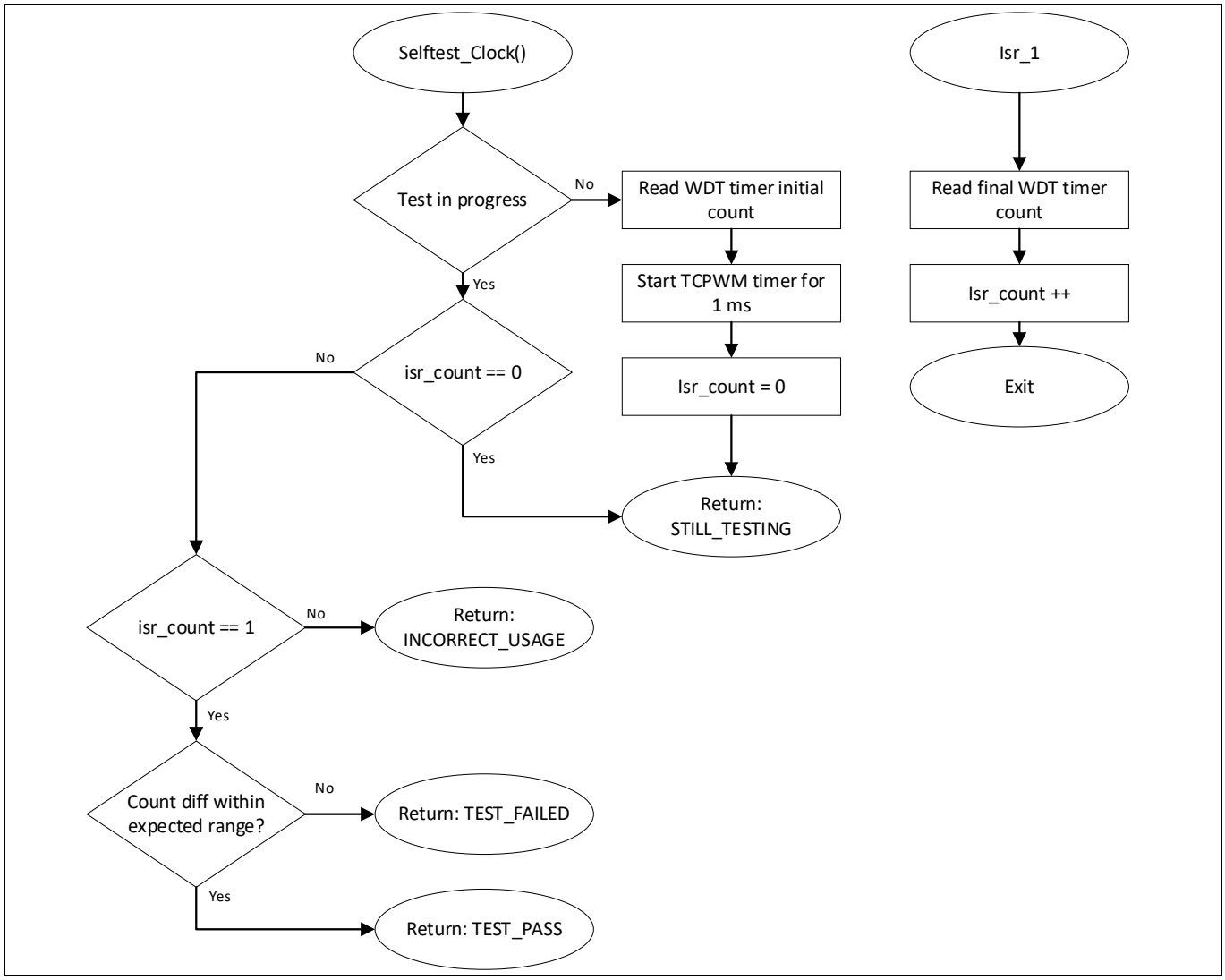


**Figure 4　　クロックセルフテストフローチャート**

## 6.6　　フラッシュ (不変メモリ) テスト

PSOC™ 4 デバイスには、最大 384 KB のフラッシュメモリが内蔵されます。フラッシュメモリは行で編成されており、各行には 128 データバイトが含まれています。

フラッシュメモリの完全な診断を完了するには、使用されているすべてのフラッシュのチェックサムを計算する必要があります。

現在のライブラリでは、Fletcher の 64 ビットチェックサムまたは 32 ビット CRC32 が使用されます。Fletcher の 64 ビット方式がデフォルトとして選択されたのは、十分なビットエラー検出を維持し、CRC-32 よりも大幅に高速で小さいためです。Fletcher の 64 ビットアルゴリズムは、完全な 32 ビットワードを読み出し、CRC32 と比較して ROM ルックアップテーブルを必要とせず、通常は読み出しごとに 2 つの追加のみを実行します。

チェックサムは、*SelfTest_Flash.h* の FLASH_TEST_MODE マクロを書き換えることで変更できます。

*SelfTest_Flash.h* ファイル内の PSOC_FLASH_END_ADDR は、監視する必要があるフラッシュアドレスの終端を定義します。

### 6.6.1 Fletcher チェックサム法

チェックサムフラッシュテストは、各 ROM またはフラッシュを読み出し、値を 64 ビット変数に累積して、フラッシュメモリ全体の実行中のチェックサムを計算します。フラッシュの実際の 64 ビットチェックサムは、フラッシュ自体の最後の 8 バイトに格納されます。提供されているカスタムリンカーファイルには、セクションが含まれます。flash_checksum GCC コンパイラ用です。IAR コンパイラでは、アドレスはインラインで定義されます。テストがフラッシュの終わりから 8 バイト (384 KB デバイスでは 0x05FFF8) に達すると、テストは停止します。カスタムリンカーファイルを使用して、チェックサムを目的の場所に配置します (Table 2 を参照)。次に、計算されたチェックサム値が、フラッシュの最後の 8 バイトに保存されている実際の値と比較されます。不一致はフラッシュエラーを示し、無効なコードを実行しようとしないようにコードの実行がフリーズされます。

*Note:* *チェックサムは、SFLASH, EEPROM, またはその他の外部フラッシュにも保存できます。*

### 6.6.2 CRC-32 法

CRC-32 方式では、ルックアップテーブルを使用して実行を高速化しますが、バイト境界でメモリを読み出し、読み出しごとに複数のビット演算を実行する必要があるため、Fletchers 64 ビット方式よりも低速です。選択された CRC32 多項式は、ISO/IEC/IEEE 802-3 で定義されている最も一般的な多項式バージョン 0x04C11DB7 (反転=0xEDB88320) です。Fletcher 64 ビットストレージサイズとの互換性を維持するために、CRC32 値は 8 バイトの予約済みフラッシュ領域の最後の 4 バイトに格納されます。

### 6.6.3 プログラミングステップ

テストを開始する前に、Appendix A: フラッシュ (不変メモリ) テストのチェックサムの設定で説明されているように、正しい事前計算済みチェックサムを設定する必要があります。

**関数**

```
uint8 SelfTest_FlashCheckSum(uint32_t DoubleWordsToTest)
```

戻り値:

- '1' – エラーが検出されました
- '2' – 1 つのブロックのチェックサムが計算されましたが、Flash の終わりに達しませんでした
- '3' – パス、すべてのフラッシュのチェックサムが計算され、チェックされます

パラメータ:

- `DoubleWordsToTest` – 関数呼び出しごとに計算されるフラッシュの 32 ビット Double ワードの数

ロケーション:

- *SelfTest_Flash.c*
- *SelfTest_Flash.h*

`SelfTest_FlashCheckSum()`関数は、チェックサムメソッドを使用してフラッシュメモリ破損テストを実行するために呼び出されます。呼び出し中に、この関数は 1 つのフラッシュブロックのチェックサムを計算します。ブロックのサイズは、*SelfTest_Flash.h* ファイルのパラメータを使用して設定できます。

```
/*Set size of one block in Flash test*/
#define FLASH_DOUBLE_WORDS_TO_TEST (512u)
```

この関数は、フラッシュ領域全体がテストされるまで複数回呼び出す必要があります。関数を呼び出すたびに、自動的に次のテストブロックにインクリメントされます。ブロックのチェックサムが計算され、テストされたフラッシュの終了アドレスに達すると、テストは 0x03 を返します。ブロックのチェックサムが計算されても、フラッシュの終了アドレスに達していない場合、テストは 0x02 を返します。エラーが検出された場合、テストは 0x01 を返します。

*Note:* *実行時にフラッシュに変更があった場合、このチェックは機能しません。チェックサムは、テストを呼び出す前に更新する必要があります。フラッシュの内容を変更する可能性のある他のテストは、フラッシュテストの前に呼び出す必要があります。*

## 6.7 SRAM (可変メモリ) テスト

*Note:* *PSOC™ 4 デバイスには、最大 32 KB の SRAM が内蔵されています。この SRAM の一部には、メモリの最後にあるスタックが含まれます。*

可変メモリテストは、IEC 60730 規格のセクション H.2.19.6 で定義されている定期的な静的メモリテストを実装します。変数メモリ内のシングルビットフォールトを検出します。可変メモリテストは、破壊的または非破壊的である場合があります。破壊テストではテスト中にメモリの内容が破壊されますが、非破壊テストではメモリの内容が保持されます。このライブラリで使用されているテストアルゴリズムは破壊的ですが、テストの前にメモリの内容を最初に保存し、完了後に内容を復元するようコードにカプセル化されます。

変数メモリには、プログラムの実行中に変化するデータが含まれます。RAM メモリテストは、RAM メモリのいずれかのビットが「1」または「0」でスタックしているかどうかを判断するために使用されます。マーチメモリテストとチェッカーボードテストは、DC 障害をチェックするために最も広く使用されている静的メモリアルゴリズムの 1 つです。

マーチテストは、アルゴリズムがわずかに異なる類似のテストのファミリで構成されています。マーチテストバリエーションは大文字で示され、特定のアーキテクチャのテスト要件に合わせてテストを調整できます。マーチ C テストは、チェッカーボードアルゴリズムよりも優れたテストカバレッジを提供し、このデバイスに最適なマーチ法であるため、PSOC™ 4 セーフティソフトウェアライブラリに実装されています。「可変 SRAM」領域と「スタック SRAM」領域には別々の機能が実装されており、テスト中にデータが破損しないようにしています。

### 6.7.1 マーチ C テスト

マーチテストでは、メモリ配列内のすべてのメモリセルに対して有限の演算セットを実行します。マーチ C テストは、変数メモリ内の次のタイプの故障を検出するために使用されます。

- スタックアットフォールト
- アドレッシングフォールト
- トランジションフォールト
- カップリングフォールト

各ロケーションのテストには 11 回の操作が必要なため、テストの複雑さは 11n です。"n" はメモリ内のビット数を示します。このテストは通常、破壊的なものですが、インフィニオンでは、各テストで小さなメモリブロックのみをテストすることで、データ破損のないマーチ C テストを提供し、ブロックの内容を保存して復元できるようにします。

### 6.7.2 マーチ C アルゴリズム

**Table 3 マーチテスト表記**

| テスト表記 | 説明 |
|---|---|
| > | アドレスシーケンスを昇順で並べ替える |
| < | アドレスシーケンスを降順で並べ替える |
| <> | アドレスの順序を昇順または降順で並べ替える |
| r0 | 読み出し操作を示す (メモリセルから「0」を読み出す) |
| r1 | 読み出し操作を示します (メモリセルから「1」を読み出す) |
| w0 | 書き込み操作を示す (メモリセルから「0」を書き込む) |
| w1 | 書き込み操作を示す (メモリセルから「1」を書き込む) |

```
MarchC
{
    <> (w0)
    > (r0 then w1)
    > (r1 then w0)
    <> (r0)
    < (r0 then w1)
    < (r1 then w0)
    > (r0)
}
```

**関数**

```
uint8 SelfTests_SRAM_March(uint8_t *startAddr, uint32_t size, uint8_t
*buffAddr, uint32_t buffSize)
```

戻り値:

- '1' – エラーが検出されました
- '2' – テスト中
- '3' – 合格、すべての RAM がテストされました。

ロケーション:

- *SelfTest_RAM.c*
- *SelfTest_RAM.h*

この関数は、データを破損せずに実行時にマーチ SRAM テストを実行するために呼び出されます。

この関数は、開始アドレスと終了アドレスのポインターを使用して、テスト対象の SRAM の領域を SRAM の別の予約部分にバックアップし、データを復元することで、実行時のマーチ C テストを実行します。

SRAM の予約部分も、データをコピーする前にマーチテストを使用してテストされます。メモリのこの領域は破損します。したがって、その中に変数を格納または配置することは禁止されます。このメモリの制御は、ユーザーに割り当てられます。使用されていない配列に割り当て、この配列の最適化を禁止

するコンパイラディレクティブを設定することを推奨します。これは、mtb-example-psoc4-sram-test プロジェクトで示されます。

SRAM の予約領域は、SRAM の末尾のスタック領域の直前にあり、*SelfTest_SRAM_March_GCC.s* ファイルと *SelfTest_SRAM_March_IAR.s* ファイルで次のパラメータを使用して設定します。

```
/* Choose maximum block size for reserved area */
.if (TEST_BLOCK_SRAM_SIZE > TEST_BLOCK_STACK_SIZE)
    .equ TEST_BLOCK_SIZE,    TEST_BLOCK_SRAM_SIZE
.else
    .equ TEST_BLOCK_SIZE,    TEST_BLOCK_STACK_SIZE
.endif


.equ MARCH_SRAM_BASE,           (CY_SRAM_BASE)
.equ MARCH_STACK_BASE,          (CY_SRAM_BASE + CY_SRAM_SIZE -
                                 CY_STACK_SIZE)
.equ MARCH_STACK_END,           (CY_SRAM_BASE + CY_SRAM_SIZE)
.equ MARCH_BUFF_ADDR_START,     (MARCH_STACK_BASE - TEST_BLOCK_SIZE)
.equ MARCH_BUFF_ADDR_END,       (MARCH_BUFF_ADDR_START +
                                 TEST_BLOCK_SRAM_SIZE)
.equ MARCH_BUFF_ADDR_STACK_END, (MARCH_BUFF_ADDR_START +
                                 TEST_BLOCK_STACK_SIZE)
```

SRAM のサイズは、*SelfTest_RAM_Config_GCC.inc* ファイルと *SelfTest_Ram_Config_IAR.inc* ファイルにあります。

```
.set CY_SRAM_BASE,              (0x20000000UL)
.set CY_STACK_SIZE,       (0x00000400u)
.set CY_SRAM_SIZE,              (0x00008000UL)
```

**Table 4　　SRAM のメモリマップ例**

| メモリ領域 | 説明 |
|---|---|
| [0x20000000;<br>(0x20008000 - CY_STACK_SIZE - TEST_BLOCK_SIZE)] | 可変 SRAM |
| [(0x20008000 - CY_STACK_SIZE - TEST_BLOCK_SIZE);<br>(0x20008000 - CY_STACK_SIZE)] | マーチ C テスト用バッファ<br>(SRAM の予約部分) |
| [(0x20008000 - CY_STACK_SIZE);<br>0x20008000] | スタック SRAM |

呼び出し中、この関数は SRAM の 1 ブロックをテストします。ブロックのサイズは、SRAM の予約バッファ領域のサイズと同じです。これは、*SelfTest_SRAM_March.s* ファイル内の次のパラメータを使用して設定できます。

```
.equ TEST_BLOCK_SRAM_SIZE,      0x00000100
```

このテストは、可変 SRAM 領域のみを対象とし、スタック領域は対象としません。ブロックが正常にテストされ、予約領域の開始アドレスに到達した場合、テストは 0x03 を返します。ブロックが正常にテストされたが、予約領域の開始アドレスに達していない場合、テストは 0x02 を返します。エラーが検出された場合、テストは 0x01 を返します。

### 6.7.3 テスト時間

デフォルトの SRAM テスト設定は、指定された SRAM サイズのアドレス空間をテストするまで、呼び出されるたびに 256 バイトをテストします。テストが呼び出されるブロックサイズと頻度は変更できます。このテストは、デフォルト設定では 1 MB あたり 1 秒かかります。

デフォルトのスタックメモリテスト設定は、1 kb のスタックアドレス空間をテストするまで、呼び出されるたびに 21 バイトをテストします。テストが呼び出されるブロックサイズと頻度は変更できます。このテストは、デフォルト設定で 1 KB ごとに 4.2 ms かかります

## 6.8 スタックオーバーフローテスト

スタックは、情報を一時的に保存するために CPU が使用する RAM のセクションです。この情報はデータ又はアドレスである場合があります。CPU には、レジスタの数が限られているため、このストレージ領域が必要です。

PSOC™ 4 では、スタックは RAM の最後にあり、下向きに使って行きます。スタックポインターは 32 ビット幅で、PUSH 命令ごとにデクリメントされ、POP 命令でインクリメントされます。

スタックオーバーフローテストの目的は、プログラムの実行中にスタックがプログラムデータメモリとオーバーラップしないようにすることです。これは、例えば、再帰関数が使用されている場合に発生する可能性があります。

このテストを実行するために、スタックの最後に予約された固定メモリブロックに定義済みのパターンが格納され、テスト関数が定期的に呼び出されて検証されます。スタックオーバーフローが発生した場合、予約されたブロックは破損したデータバイトで上書きされ、オーバーフローエラーとして扱う必要があります。

**関数**

```
void SelfTests_Init_Stack_Test_Range(uint16_t* stack_address, uint16_t
stack_length)
```

戻り値: なし

ロケーション:

- *SelfTest_Stack.c*
- *SelfTest_Stack.h*

この関数は、スタックオーバーフローをテストするために実行時に定期的に呼び出されます。ブロックサイズは偶数の値である必要があり、*SelfTest_Stack.h* にあるマクロを使用して変更できます。

```
#define STACK_TEST_BLOCK_SIZE 0x08u
```

パターンは、*SelfTest_Stack.h* にあるマクロを使用して変更できます。

```
#define STACK_TEST_PATTERN 0x55AAu
```

## 6.9 デジタル I/O テスト

PSOC™ 4 は最大 98 本のプログラム可能な GPIO ピンを提供します。どの GPIO ピンも、CAPSENSE™, LCD, アナログ, またはデジタルにできます。駆動モード, 強度, およびスルーレートはプログラム可能です。

デジタル I/O はポートに配置され、ポートごとに最大 8 つのピンがあります。一部の I/O ピンは、特殊機能 (USB, デバッグポート, 水晶発振器) で多重化されています。特殊機能は、特定の機能に関連付けられた制御レジスタを使用して有効にします。

テストの目標は、I/O ピンが GND または Vcc に短絡していないことを確認することです。

通常の動作条件では、ピンとグランド間およびピンと VCC 間の抵抗は非常に高くなります。短絡を検出するために、抵抗値を PSOC™ 4 の内部プルアップ抵抗と比較します。

ピンとグランド間の短絡を検出するために、ピンはプルアップ駆動モードに設定されます。通常の状態では、CPU はプルアップ抵抗のために論理値"1"を読み出します。ピンが低抵抗でグランドに接続されている場合、入力レベルは論理値"0"として認識されます。

センサーと VCC の短絡を検出するために、センサーピンはプルダウンモードに設定されています。通常の状態では入力レベルはゼロです。

*Note:* *このテストはアプリケーションに依存しており、カスタマイズが必要な場合があります。デフォルトのテスト値では、ピンがエンドアプリケーションに対して一時的に正しくない状態に設定される可能性があります。*

### 関数

`uint8_t SelfTest_IO(void)`

戻り値:

- '0' – エラーはありません
- '1' – エラー検出 (VCC への短絡)
- '2' – エラー検出 (GND への短絡)

ロケーション:

- *SelfTest_IO.c*
- *SelfTest_IO.h*

関数 `SelfTests_IO()`は、I/O ピンの GND または Vcc への短絡をチェックするために呼び出されます。`SelfTests_IO()`関数の `PintToTest` 配列は、テストする必要があるピンを設定するために使用されます。

例:

```
static const uint8 PinToTest[] =
{
       0xCCu, /* PORT0 mask */
       0x1Fu, /* PORT1 mask */
       0xC2u, /* PORT2 mask */
       0xF0u, /* PORT3 mask */
```

```
        0x0Cu, /* PORT4 mask */
};
```

各ピンは、PinToTest テーブルポートマスクの対応するビットで表されます。ピン 0 は LSB で表され、ピン 7 は MSB で表されます。ピンをテストする必要がある場合は、対応するビットを 1 に設定する必要があります。

## 6.10 アナログテスト

アナログテストは、ADC, コンパレータ, およびオペアンプテストで構成されます。これらのテストは、2 つの固定リファレンス電圧に依存しています。特定の PSOC™デバイスに応じて、これらの電圧リファレンスは、外部ハードウェアを使用して内部または外部で生成できます。ユーザーによって電圧リファレンスを設定し、電圧をテスト対象のハードウェアにルーティングします。これにより、ユーザーは柔軟性を持ち、内部ルーティングを制御できます。

AMUX バスはセグメント化されているため、通常の使用で柔軟性が高まります。 mtb-example-psoc4-analog-test のサンプルコードでは、これらのセグメントが結合されているため、電圧リファレンスが ADC やその他のテスト対象のアナログリソースにルーティングできます。

### 6.10.1 リファレンス電圧の生成

デバイスが MSC CAPSENSE™ハードウェアを 2 つ持つ場合に、内部リファレンス電圧が生成できます。PSOC™ 4100S Max は、高性能ハードウェアを備えたその例の 1 つです。各 MSC ブロックは、固定電圧 Vref とレシオメトリック電圧 Vdda/2 を 1 つのリファレンスとして出力します。これらの電圧リファレンスは、2 つの AMUX バスを介して ADC などのアナログリソースにルーティングできます。2 つの MSC ブロックを使用するようにライブラリを設定するには、*次のように SelfTest_Analog.h* を変更します。

```
#define ANALOG_TEST_VREF                (ANALOG_TEST_VREF_DUAL_MSC)
```

デバイスに IDAC と GPIO ピンを介して AMUXBUS に接続された外部プルダウン抵抗がある場合、設定可能なリファレンス電圧を生成できます。CSDv2 ブロックを搭載したデバイスは、この動作モードをサポートします。CSDv2 ブロックには 2 つの IDAC が含まれており、IDAC とプルダウン抵抗の組み合わせごとに 1 つの電圧リファレンスを生成できます。IDAC から AMUXBUS を介してプルダウン抵抗に固定電流を印加すると、AMUXBUS に設定可能な電圧リファレンスが作成されます。4.7 kΩ が計算で使用されるデフォルトのプルダウン抵抗値であることに注意してください。CSDv2 ブロックを使用するようにライブラリを設定するには、*SelfTest_Analog.h* ファイルを次のように変更します。

```
#define ANALOG_TEST_VREF                (ANALOG_TEST_VREF_CSD_IDAC)
```

外部電圧リファレンスは、外部ハードウェアを使用してデバイスに提供できます。可能な方法は、VDDA と GND の間に 3 つの抵抗による分圧器を使用することです。これにより、2/3*VDDA と 1/3*VDDA の 2 つのリファレンス電圧が生成されます。これらのリファレンス電圧は、任意の GPIO ピンに接続でき、GPIO を介して AMUXBUS に接続できます。外部電圧値を使用するようにライブラリを設定するには、*次のように SelfTest_Analog.h* ファイルを変更します。

```
#define ANALOG_TEST_VREF                (ANALOG_TEST_VREF_EXTERNAL)
```

**Table 5 リファレンス電圧と PSOC™ 4 ファミリ**

| 電圧リファレンス | 対応する PSOC™ 4 デバイスファミリー |
|---|---|
| ANALOG_TEST_VREF_EXTERNAL | すべての PSOC™ 4 デバイスはこのライブラリによってサポートされます |
| ANALOG_TEST_VREF_CSD_IDAC | PSOC™ 4500S, 4100S, 4100S Plus 256K ファミリー |

| 電圧リファレンス | 対応する PSOC™ 4 デバイスファミリー |
|---|---|
| ANALOG_TEST_VREF_DUAL_MSC | PSOC™ 4100S Max |

電圧リファレンスを ANALOG_TEST_VREF_DUAL_MSC または ANALOG_TEST_VREF_CSD_IDAC に設定すると、リファレンス電圧を設定するための適切なヘルパー機能が自動的に含まれます。電圧リファレンスの設定の詳細については、*SelfTest_Analog.c/h* を参照してください。電圧リファレンスを設定し、適切なアナログブロックにルーティングする方法については、*mtb-example-psoc4-analog-test* のサンプルコードを参照してください。

### 6.10.2 ADC テスト

ADC テストは、IEC 60730 規格のセクション H.2.18.8 で定義されている独立した入力比較を実装します。これは、指定された許容範囲内になるように設計された入力/出力を比較するためのフォールト/エラー制御手法を提供します。

このテストの目的は、ADC の動作を確認することです。テストを実行する前に、ユーザーコードが適切な電圧リファレンスを AMUXBUSA と AMUXBUSB に接続し、ADC を構成できることを確認する必要があります。リファレンス電圧 (mV) と精度制限 (%) は、*SelfTest_Analog.h* ファイルで定義されています。リファレンス電圧の生成の詳細については、リファレンス電圧の生成セクションを参照してください。

リファレンス電圧 (mV) と精度制限 (%) は、*SelfTest_Analog.h* ファイルで定義されています。

```
#define ADC_TEST_ACC                    12u /* 12% accuracy */
#define ANALOG_TEST_VREF1_MV            (INTERNAL_BANDGAP_VREF_MV)
#define ANALOG_TEST_VREF2_MV            (CY_CFG_PWR_VDDA_MV/2u)
```

**関数**

```
uint8_t SelfTests_ADC(SAR_Type * base, uint32_t channel, int16_t
expected_res, int16_t accuracy, uint32_t vbg_channel, bool count_to_mV)
```

戻り値:

- '0' – エラーはありません
- 1 - エラーが検出されました

パラメータ:

- base–SAR インスタンスレジスタを記述する構造体へのポインター
- channel–電圧を読み出すチャネルの番号
- expected_res–期待される結果 (mV)
- accuracy – 許容できる期待値の上下の閾値

ロケーション:

- *SelfTest_Analog.h*
- *SelfTest_Analog.c*

デジタル化された入力電圧値が、定義された精度内で必要な基準電圧値と等しい場合、テストは成功です。テストが成功すると、関数は「0」を返します。それ以外の場合は、「1」を返します。

*Note:* *このテストは、2 つの異なる電圧をテストするために 2 回呼び出す必要があります。実際のハードウェア障害が発生した場合、テストベンチは、テストに失敗した関数呼び出しのインスタンスに応じて、"Error: "ADC_V1" または "Error: ADC_V2" を出力します。*

*Note:* *テスト機能では、ADC は設定されません。したがって、テストの終了後に設定して復元する必要があります。これを行う方法の例については、mtb-example-psoc4-analog-test サンプルコードを参照してください。Init 関数を呼び出すことは、設定を復元する最も一般的な方法です。他の参照リソースやルーティングリソースも、init 関数を再利用して復元する必要があるかもしれません。*

次の 2 つの例は、選択したリファレンス電圧源に応じて ADC テストを実行する方法を示します。

2 つの電圧リファレンス (外部リファレンス電圧または MSC リファレンス電圧) を使用して ADC をテストするには、次のようにします。

1. リファレンス電圧 1 と 2 を設定してください。
2. ADC をセルフテスト用に設定してください。
3. リファレンス電圧を ADC にルーティングしてください。
4. リファレンス電圧 1 のチャネルに対して `SelfTest_ADC()` を実行してください。
5. リファレンス電圧 2 のチャネルに対して `SelfTest_ADC()` を実行してください。
6. ADC と内部ルーティングを復元してください。

1 つの設定可能な電圧リファレンス (IDAC 法のリファレンス電圧) を使用して ADC をテストするには、次のようにします。

1. 最初のテスト電圧を出力するようにリファレンス電圧を設定してください。
2. ADC をセルフテスト用に設定してください。
3. リファレンス電圧を ADC にルーティングしてください。
4. 設定可能なリファレンス電圧のチャネルに対して `SelfTest_ADC()`を実行してください。
5. 2 番目のテスト電圧を出力するようにリファレンス電圧を設定してください。
6. 設定可能なリファレンス電圧のチャネルに対して `SelfTest_ADC()`を実行してください。
7. ADC と内部ルーティングを復元してください。

### 6.10.3 コンパレータテスト

コンパレータの機能テストは、2 つの異なるリファレンス電圧を同時に使用して実行されます。リファレンスは、リファレンス電圧の生成セクションに示すように、内部または外部で生成できます。これらのリファレンス電圧は、AMUX バスを介してアナログコンパレータの入力に送られ、入力がスイッチングされて、コンパレータの出力値が強制的に変化します。

**関数**

```
uint8_t SelfTest_Comparator(LPCOMP_Type const* lpcomp_base、
            cy_en_lpcomp_channel_t lpcomp_channel,uint8_t expected_res)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

パラメータ:

- `lpcomp_base` – 低電力コンパレータレジスタ構造体ポインター
- `lpcomp_channel` – 低電力コンパレータのチャネル番号
- `expected_res` – コンパレータ出力期待値正の入力が負の入力電圧より大きいと予想される場合、非ゼロ値。

ロケーション:

- *SelfTest_Analog.h*
- *SelfTest_Analog.c*

出力状態が読み出され、期待値と比較されます。テストが成功すると、関数は「0」を返します。それ以外の場合は、「1」を返します。

*Note:* *このテストは、コンパレータへの 2 つの入力電圧設定をテストするために 2 回呼び出す必要があります。実際のハードウェア障害が発生した場合、テストベンチは、テストに失敗した関数呼び出しのインスタンスに応じて、"Error: COMP_V1" または "Error: "COMP_V2" を出力します。*

*Note:* *テスト機能では、コンパレータは設定されません。したがって、テストの終了後に設定して復元する必要があります。これを行う方法の例については、mtb-example-psoc4-analog-test サンプルコードを参照してください。Init 関数を呼び出すことは、設定を復元する最も一般的な方法です。他の参照リソースやルーティングリソースも、init 関数を再利用して復元する必要があるかもしれません。*

2 つの電圧リファレンスを使用してコンパレータをテストするには:

1. リファレンス電圧 1 と 2 を設定してください。
2. コンパレータをセルフテスト用に設定してください。
3. リファレンス電圧 1 と 2 をコンパレータの正と負の端子にルーティングしてください。
4. `SelfTest_Comparator()`を実行してください。
5. コンパレータに入る 2 つの基準電圧を切り替えます。
6. `SelfTest_Comparator()`を実行してください。
7. コンパレータと内部ルーティングを復元してください。

### 6.10.4 オペアンプテスト

オペアンプテストは IEC 60730 規格では提供されていませんが、安全性を重要視するブロックでオペアンプを使用する場合に役立ちます。

オペアンプテストは、IEC 60730 規格のセクション H.2.18.8 で定義されている「独立した出力比較」テストの原則に準拠しています。これは、指定された許容範囲内になるように設計された入力/出力を比較するためのフォールト/エラー制御手法を提供します。

このテストの目的は、オペアンプのアナログ機能をテストすることです。テストの前に、「リファレンス電圧の生成」セクションで説明したように、リファレンス電圧を生成する必要があります。次に、リファレンス電圧をオペアンプに接続する必要がありますが、これは通常、AMUX バスを介して行うことができます。オペアンプは 1 倍のゲインで構成され、その出力は ADC に直接ルーティングされます。こ

のテストは、信号パスにオペアンプを追加するだけで ADC テストと同じように動作し、同じ許容誤差パーセントを使用します。

### 関数

```
uint8 SelfTests_Opamp (SAR_Type* sar_base, int16_t expected_res, int16_t accuracy, uint32_t opamp_in_channel, bool count_to_mV)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

パラメータ:

- `sar_base` – SAR インスタンスレジスタ構造体ポインター
- `expected_res` – 期待電値 (mV)
- `accuracy` – 許容できる期待値の上下の閾値

ロケーション:

- *SelfTest_Analog.h*
- *SelfTest_Analog.c*

入力オペアンプの電圧値が、定義された精度の範囲内で期待値と等しい場合、テストは成功です。テストが成功すると、関数は「0」を返します。それ以外の場合は、「1」を返します。

*Note:* *このテストは、2 つの異なる電圧をテストするために 2 回呼び出す必要があります。実際のハードウェア障害が発生した場合、テストベンチは、テストに失敗した関数呼び出しのインスタンスに応じて、"Error: OPAMP_V1" または "Error: OPAMP_V2 Task" を出力します。*

*Note:* *テスト機能では、ADC または CTB は設定されません。したがって、テストの終了後に設定して復元する必要があります。これを行う方法の例については、mtb-example-psoc4-analog-test サンプルコードを参照してください。Init 関数を呼び出すことは、設定を復元する最も一般的な方法です。他の参照リソースやルーティングリソースも、init 関数を再利用して復元する必要があるかもしれません。*

次の 2 つの例に、選択したリファレンス電圧源に応じてオペアンプテストを実行する方法を示します。

2 つの電圧リファレンス(外部または MSC リファレンス電圧)を使用してオペアンプをテストする:

1. リファレンス電圧 1 と 2 を設定してください。
2. ADC と CTB (オペアンプ) をセルフテスト用に設定してください。
3. リファレンス電圧 1 をオペアンプの正入力にルーティングしてください。
4. `SelfTest_Opamp()`を実行してください。
5. リファレンス電圧 2 をオペアンプの正入力にルーティングしてください。
6. `SelfTest_Opamp()`を実行してください。
7. ADC, CTB, および内部ルーティングを復元してください。

1 つの設定可能な電圧リファレンス (IDAC 法のリファレンス電圧) を使用してオペアンプをテストする:

1. 最初のテスト電圧を出力するようにリファレンス電圧を設定してください。

2. ADC と CTB (オペアンプ) をセルフテスト用に設定してください。
3. リファレンス電圧をオペアンプにルーティングしてください。
4. `SelfTest_Opamp()`を実行してください。
5. 2 番目のテスト電圧を出力するようにリファレンス電圧を設定してください。
6. `SelfTest_Opamp()`を実行してください。
7. ADC, CTB, および内部ルーティングを復元してください。

## 6.11 UART 通信テスト

このテストは、UART データループバックテストを実装します。このループバックは、Smart I/O を使用して内部で実行することも、物理的に外部で配線することもできます。送信されたデータが受信データと等しく、2 が返された場合、テストは成功です。各関数呼び出しは、テストデータをインクリメントします。256 回の関数呼び出しの後、テストで 256 個の値すべてのテストが終了し、それらがすべて成功すると、関数は 3 を返します。Smart I/O は、Smart I/O に接続されている SCB のループバックに使用できます。

すべてのデバイスが RX 信号と TX 信号を Smart I/O 経由でルーティングできるわけではないことに注意してください。SCB から Smart I/O への接続については、デバイスのデータシートを参照してください。ただし、UART RX ピンと TX ピンが外部で接続されている場合、このテストはすべての SCB で使用できます。

**関数**

```
uint8  SelfTest_UART_SCB(CySCB_Type* base)
```

戻り値:

- '1' – エラーが検出されました
- '2' – 現在の値でテストに合格、0x00 から 0xFF の範囲内のすべてのテストは完了していません
- '3' – 合格、0x00 から 0xFF の全範囲のテストを完了
- '4' – ERROR_TX_NOT_EMPTY
- '5' – ERROR_RX_NOT_EMPTY
- '6' – ERROR_TX_NOT_ENABLE
- '7' – ERROR_RX_NOT_ENABLE

パラメータ:

- `base` - 設定する SCB ハードウェアへのポインター

ロケーション:

- *SelfTest_UART_SCB.h*
- *SelfTest_UART_SCB.c*

入力端子と出力端子は、対応するピンとループを切り替えて相互にループし、UART マルチプレクサとデマルチプレクサを使用して内部ループバックテストを提供します。テスト前に受信バッファまたは送信バッファが空でない場合、テストは実行されず、ERROR_RX_NOT_EMPTY または ERROR_TX_NOT_EMPTY の状態を返します。

呼び出し中、関数は 1 バイトを送信します。送信された値は、各関数呼び出しの後にインクリメントされます。テスト値の範囲は 0x00～0xFF です。

*Note:* *内部ループバックに Smart I/O を使用する場合。ユーザーはテスト前に Smart I/O を設定し、テスト後に Smart I/O を復元してください。*

## 6.12 通信 SPI テスト

このテストは、SPI ループバックテストを実装します。このループバックは、Smart I/O を使用して内部で実行することも、物理的に外部で配線することもできます。送信されたデータが受信データと等しく、2 が返された場合、テストは成功です。各関数呼び出しは、テストデータをインクリメントします。256 回の関数呼び出しの後、テストで 256 個の値すべてのテストが終了し、それらがすべて成功すると、関数は 3 を返します。

すべてのデバイスが Smart I/O を介して MOSI 信号と MISO 信号をルーティングできるわけではないことに注意してください。SCB から Smart I/O への接続については、デバイスのデータシートを参照してください。ただし、SPI MOSI ピンと MISO ピンが外部で接続されている場合は、このテストをすべての SCB で使用できます。

**関数**

```
uint8  SelfTest_SPI_SCB(CySCB_Type* base)
```

戻り値:

- '1' – エラーが検出されました
- '2' – 現在の値でテストに合格、0x00 から 0xFF の範囲内のすべてのテストは完了していません
- '3' – 合格、0x00〜0xFF の全範囲のテストを完了
- '4' – ERROR_TX_NOT_EMPTY
- '5' – ERROR_RX_NOT_EMPTY

パラメータ:

- `base` - 設定する SCB ハードウェアへのポインター

ロケーション:

- *SelfTest_SPI_SCB.h*
- *SelfTest_SPI_SCB.c*

SPI の入力端子と出力端子は、対応するピンとループを相互に切り替えて、マルチプレクサとデマルチプレクサを使用した内部ループバックテストを提供します。テスト前に受信バッファまたは送信バッファが空でない場合、テストは実行されず、ERROR_RX_NOT_EMPTY または ERROR_TX_NOT_EMPTY の状態を返します。

呼び出し中、関数は 1 バイトを送信します。送信された値は、各関数呼び出しの後にインクリメントされます。テスト値の範囲は 0x00〜0xFF です。

*Note:* *内部ループバックに Smart I/O を使用する場合。ユーザーはテスト前に Smart I/O を設定し、テスト後に Smart I/O を復元してください。*

## 6.13 I2C 通信テスト

このテストは、I2C マスターとスレーブ間の外部接続が必要であり、2 つの SCB ブロックを使用します。

このプロジェクトでは、**Error! Reference source not found.**に示すように、テスト対象のコンポーネントとして I2C マスターを使用します。変更を加えなくても、プロジェクトは I2C スレーブのテストにも使用でき、両方が同時にテストされます。

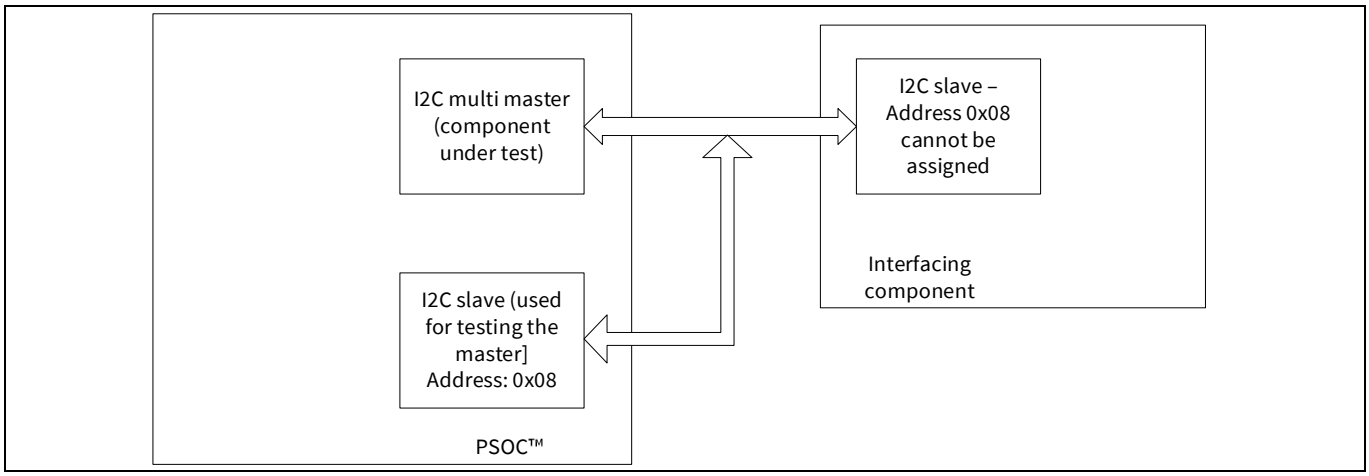


**Figure 5 テスト中の I2C マスター**

1. マスターは、スレーブアドレス[0x08]に I2C データを定期的に送信します。
2. スレーブは、データを読み出し、補完し、スレーブ読み出しバッファに書き込みます。
3. マスターはスレーブからデータを読み出し、最後のインタラクションで送信されたデータの補数と比較します。

### 関数

```
uint8_t SelfTest_I2C_SCB (CySCB_Type* master_base,
                          cy_stc_scb_i2c_context_t* master_context,
                          CySCB_Type* slave_base,
                          cy_stc_scb_i2c_context_t* slave_context,
                          uint8_t* slave_read_buf, uint8_t* slave_write_buf)
```

戻り値:

- '1' – エラーなし
- '2' – エラーを検出しました
- '3' – I2C マスタービジー

パラメータ:

- `master_base` – マスター I2C SCB へのポインター
- `master_context` – マスター I2C SCB コンテキストへのポインター
- `slave_base` – スレーブ I2C SCB へのポインター
- `slave_context` – スレーブ I2C SCB コンテキストへのポインター
- `slave_read_buf` – スレーブ読み出しバッファへのポインター

- `transmitData` – I2C スレーブから受信したデータと比較するデータ

ロケーション:

- *SelfTest_I2C_SCB.h*
- *SelfTest_I2C_SCB.c*

**I2C 書き込み API – 操作**

- I2C マスタートランザクションを開始して、スレーブにデータを書き込みます
- 書き込み転送の完了を待機します
- スレーブ書き込みバッファを読み出します
- 読み出されたデータに対して 1 の補数を実行します
- 補完されたデータをスレーブ読み出しバッファに書き込みます

**I2C 読み出し API – 操作**

- I2C マスタートランザクションを開始して、スレーブからデータを読み出します
- 読み出し転送の完了を待機します
- 読み出されたデータが、書き込まれたデータの補数と等しいかどうかを確認します

## 6.14 スタートアップ コンフィギュレーションレジスタ テスト

このテストでは、スタートアップコンフィギュレーションレジスタを確認する方法の例について説明します。

1. デジタルクロック設定レジスタをテストします。
2. アナログ設定レジスタをテストします (起動後にデフォルト値に設定されます)。
3. GPIO コンフィギュレーションと HSIOM レジスタをテストします。

これらのスタートアップコンフィギュレーションレジスタは通常静的で、「cycfg」のプレフィックスが付いたファイル (*cycfg_clocks.c* など) で設定されます。これらのファイルは、アプリケーションフォルダの下の *GeneratedSource* フォルダにあります。まれに、これらのレジスタの一部が動的に更新される場合があります。動的に更新されるレジスタは、このテストから除外する必要があります。代わりに、動的レジスタは、現在の正しい値に関する知識を使用してアプリケーションでテストされます。

関数には 2 つのテストモードが実装されています。

- デバイスの起動後に、スタートアップコンフィギュレーションレジスタの複製をフラッシュメモリに保存します。定期的に、コンフィギュレーションレジスタは保存されている複製と比較されます。破損したレジスタは、チェック後にフラッシュから復元できます。
- CRC ステータスセマフォが設定されている場合は、計算された CRC を以前に Flash に保存された CRC と比較します。ステータスセマフォが設定されていない場合は、CRC を計算してフラッシュに保存し、ステータスセマフォを設定する必要があります。

*Note:* *デバイスコンフィギュレータで変更を加えると、他の設定レジスタが cycfg ファイルに生成される場合があります。その後、設定によって変更されるレジスタに基づいて、必要なレジスタのリストを変更する必要があります。*

**関数**

```
cy_en_flashdrv_status_t SelfTests_Save_StartUp_ConfigReg(void)
```

戻り値:

- '0' – フラッシュへの書き込みは成功しました
- '1' – フラッシュ書き込み中にエラーが検出されました

ロケーション:

- *SelfTest_ConfigRegisters.c*
- *SelfTest_ConfigRegisters.h*

*Note:* *この機能は、PSOC™ 4 の初期電源投入し初期化後、メインプログラムに入る前に一度呼び出す必要があります。スタートアップコンフィギュレーションレジスタの値をフラッシュに書き込みます。この最初の書き込み後(通常は製造中) は、レジスタ値は既に格納されているため、この関数を再度呼び出す必要はありません。*

*Note:* *スタートアップコンフィギュレーションレジスタまたはCRC 値が、すでに使用されているフラッシュロケーションに保存されていないことを確認します。例えば、フラッシュテストでは、デフォルトでは最後の8 バイトを使用してフラッシュチェックサムが格納されます。*

### 関数

```
uint8_t SelfTests_StartUp_ConfigReg(void)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

ロケーション:

- *SelfTest_ConfigRegisters.c*
- *SelfTest_ConfigRegisters.h*

この関数はスタートアップコンフィギュレーションレジスタをチェックします。次の 2 つの別個の動作モードがあります。

- CRC-16 の計算と検証
- レジスタの複製コピーとの比較

モードは、 *SelfTest_ConfigRegisters.h ファイル内の* パラメータを使用して定義できます。

```
#define STARTUP_CFG_REGS_MODE    CFG_REGS_CRC_MODE or CFG_REGS_TO_FLASH_MODE
#define STARTUP_CFG_REGS_MODE     (CFG_REGS_TO_FLASH_MODE)
```

フラッシュモードでは、テストはレジスタの複製をフラッシュに格納し、レジスタを複製と比較します。値が異なる場合は、フェイル (1) を返します。このモードでは、レジスタを復元できます。`SelfTests_Save_StartUp_ConfigReg` は、複製をフラッシュに保存するために使用されます。`CONF_REG_FIRST_ROW` はフラッシュメモリ内のスタートアップコンフィギュレーションレジスタの位置を定義し、SelfTest_ConfigRegisters.h で自動的に計算されます。

```
#define STARTUP_CFG_REGS_MODE     (CFG_REGS_CRC_MODE)
```

CRC モードでは、関数はレジスタの CRC32 を計算し、CRC をフラッシュに保存します。その後、関数呼び出しは CRC32 を再計算し、保存された値と比較します。値が異なる場合は、フェイル (1) を返します。

デフォルトでは、CRC 値はフラッシュの最後尾から 8 バイト目 から 5 バイト目の位置に格納されます。フラッシュテストと併用する場合は、フラッシュの最後の 8 バイトを使用するため、CRC 値をフラッシュの末尾から 16 バイト目 から 13 バイト目の位置に移動することを推奨します。

CRC-2 値のカスタムロケーションを定義するためには *SelfTest_ConfigRegisters.h* ファイル内の定義 `CRC_STARTUP_SEMAPHORE_SHIFT` および `CRC_STARTUP_LO` を更新します。

## 6.15 ウォッチドッグテスト

この関数は、ウォッチドッグ機能テストを実装します。この関数は WDT を開始し、無限ループを実行します。WDT が機能する場合は、リセットが生成されます。リセットの後、ファームウェアはリセットソースを検出します。ウォッチドッグがリセットのソースである場合、関数は戻ります。それ以外の場合は、無限ループが実行されます。

**関数**

```
void  SelfTest_WDT(void)
```

戻り値: なし

ロケーション:

- *SelfTest_WDT.h*
- *SelfTest_WDT.c*

Figure 6 にテストフローチャートを示します。

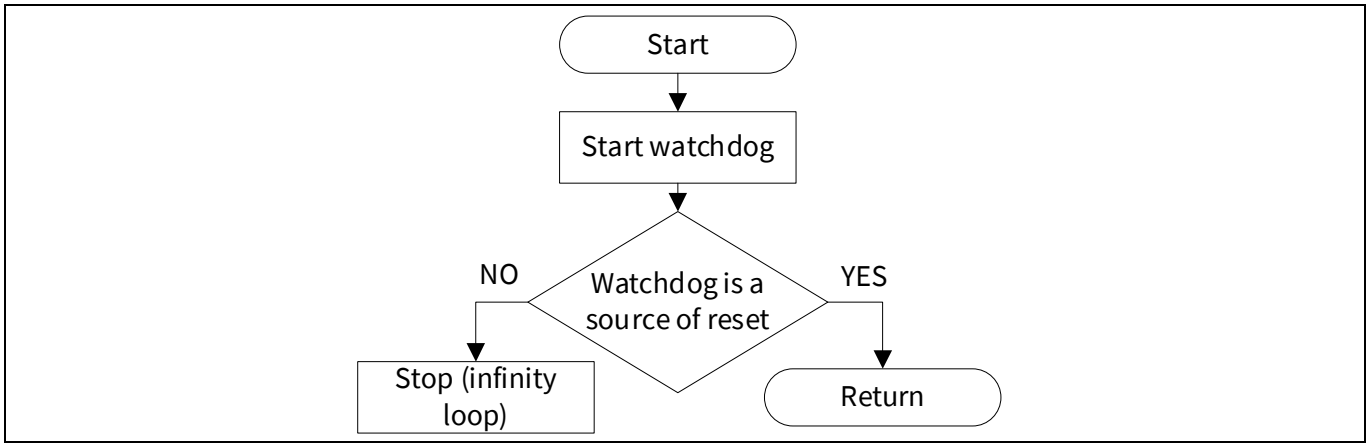


**Figure 6 WDT テストの PSOC™実装**

## 6.16 TCPWM テスト

このテストでは、クラス B Safety Test LibraryPWM を使用して、PSOC™6 MCU でタイマー/カウンター、PWM、および PWM 出力 Gate Kill として構成された TCPWM リソースをテストする方法を示します。IEC 60730 標準規格に準拠したこれらの周辺機器の適切な動作と精度を検証します。

### 6.16.1 タイマー/カウンター テスト

タイマー/カウンターは、クロック (タイマー) または外部イベント (カウンター) をカウントするために使用されます。テストではタイマー機能を使用します。テストは、すべての安全性を重要視するタイマー/カウンターで実行する必要があります。

#### 関数

```
uint8_t SelfTest_Counter_Timer(void)
```

戻り値:

‘0’ – エラーはありません

‘1’ – エラーが検出されました

ロケーション:

- *SelfTest_Timer_Counter.h*
- *SelfTest_Timer_Counter.c*

### 6.16.2 PWM テスト

PWM は、次の手順を使用してテストされます。

1. 1/3 デューティ ON、2/3 デューティ OFF、1 ミリ秒周期で動作するように 16 ビット PWM を設定してください。PWM を開始してください。
2. CPU を 5 ミリ秒間ループさせてください。ループ内で出力を連続的にポーリングしてください。0 と 1 の出力のインスタンスをカウントしてください。
3. OFF/ON 比が 1 7/8 から 2 1/8 の間であれば、テストは成功します。範囲が使用されるのは、CPU ポーリング ループが PWM タイミングに対して非同期であるためです。

#### 関数

```
uint8_t SelfTest_PWM(GPIO_PRT_Type *pinbase, uint32_t pinNum)
```

戻り値:

‘0’ – エラーはありません

‘1’ – エラーが検出されました

パラメータ:

`pinbase` – PWM ピンが接続されている GPIO ポート インスタンスへのポインター

`pinNum` – PWM ピンが接続されている GPIO ピン番号

ロケーション:

- *SelfTest_PWM.h*
- *SelfTest_PWM.c*

### 6.16.3 PWM Gate Kill テスト

「Gate Kill」機能は、モーターコントローラーやマルチレベルパワーコンバーターで使用されます。過電圧または過電流状態が検出されると、過電圧または過電流が発生してから 50 ナノ秒未満で Gate Kill が出力ドライバをシャットダウンします。

#### 関数

```
uint8_t SelfTest_PWM_GateKill(TCPWM_Type *base, uint32_t cntNum)
```

戻り値:

'0' – エラーはありません

'1' – エラーが検出されました

パラメータ:

`base` – TCPWM インスタンスへのポインター

`cntNum` – 選択した TCPWM のカウンター インスタンス番号

ロケーション:

- *SelfTest_PWM_GateKill.h*
- *SelfTest_PWM_GateKill.h*

PWM GateKill は、次の手順を使用してテストされます。

1. Kill モードを Stop on Kill に設定してください。
2. 低電力コンパレーター出力または SAR ADC の電圧範囲違反は、TCPWM の Kill 画像信号への経路です。このため、+ve 端子の電圧が-ve よりも大きい場合、過電圧/過電流状態を示します。
3. 過電圧または過電流状態が発生すると、PWM 出力は Kill されます。
4. TCPWM ベースと `CntNum` は、カウンターが停止しているかどうかを確認するために渡されます。
5. カウンターがインクリメント/デクリメントされていない場合、PWM 出力は非アクティブです。

## 6.17 CAPSENSE™ テスト

CAPSENSE™セーフティライブラリには、CAPSENSE™コンポーネントと関連周辺機器の適切な機能を検証するためのいくつかのテストが含まれています。このライブラリは、インフィニオンによる事前認証 (IEC 60730-1 による) を受けており、標準の CAPSENSE™コンポーネントの一部として含まれます。このライブラリには、次の検証のためのテストが含まれます。

- センサー短絡
- 外付けコンデンサ
- CAPSENSE™設定データの完全性
- センサー信号 (raw カウント) とセンサーベースラインの完全性
- VDDA

*Note:* *詳細、サンプルコード、および証明書 (これらは NDA の下で配布されています) については、インフィニオン テクニカルサポートにお問い合わせください。*

## 6.18 通信 UART データ転送プロトコルの例

システムコンポーネント間でデータを転送する際のシステムの安全性を高めるために、CRC およびパケット処理を備えた通信プロトコルを使用できます。安全通信の例を次に示します。

データは、定義された構造を持つパケットに配置され転送されます。すべてのパケットには、パケットの安全な転送を確保するために、パケットデータで計算された CRC16 があります。次の図に、パケットの形式を示します。

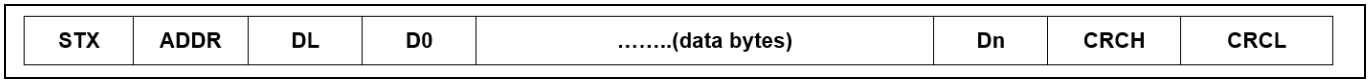


**Figure 7 パケット構造**

予約済みパケット開始マーカー (STX) 値の送信を許可するには、一般的なエスケープ方式を使用します。パケット内の任意のバイトが STX または ESC と等しい場合、 パケットは 2 バイトのシーケンスに変化します。パケットバイト = ESC の場合は、2 バイト (ESC, ESC + 1) に置き換えます。パケットバイト = STX の場合は、2 バイト (ESC, STX + 1) に置き換えます。この手順では、ユニークなパケット開始シンボルを提供します。ESC バイトは常に 0x1B に等しくなります。これはパケットの一部ではなく、常に (パケットバイト+ 1) または (ESC, STX + 1) の前に送信されます。 Table 6 に、パケットフィールドの説明を示します。

**Table 6 パケットフィールドの説明**

| 名称 | 長さ | 値 | 説明 |
|---|---|---|---|
| STX | 1 バイト | 0x02 | ユニークなパケットマーカーの開始 = 0x02。 |
| ADDR | 1 バイトまたは 2 バイト | 0x02 を除く 0x00…0xFF | スレーブアドレス。このバイトが STX と等しい場合、(ESC) + (STX + 1) の 2 バイトシーケンスに変化します。このバイトが ESC と等しい場合、(ESC) + (ESC +1) の 2 バイトシーケンスに変化します。 |
| DL | 1 バイトまたは 2 バイト | 0x02 を除く 0x00…0xFF | パケットのデータ長 (プロトコルバイトなし)。このバイトが STX と等しい場合、(ESC) + (STX + 1) の 2 バイトシーケンスに変化します。このバイトが ESC と等しい場合、(ESC) + (ESC + 1) の 2 バイトシーケンスに変化します。 |
| D0…Dn (data) | 1…510 バイト | 0x02 を除く 0x00…0xFF | パケットデータ。データ内のいずれかのバイトが STX と等しい場合、(ESC) + (STX + 1) の 2 バイトシーケンスに変化します。データ内のいずれかのバイトが ESC と等しい場合、そのバイトは (ESC) + (ESC + 1) の 2 バイトシーケンスに変化します。 |
| CRCH | 1 バイトまたは 2 バイト | 0x02 を除く 0x00…0xFF | パケット CRC の MSB。CRC-16 を使用。CRC は ADDR から最後のデータバイトまでのすべてのパケットバイトに対して計算されます。CRC は ESC 変更手順後に計算されます。このバイトが STX と等しい場合、(ESC) + (STX + 1) の 2 バイトシーケンスに変化します。このバイトが ESC と等しい場合、(ESC) + (ESC + 1) の 2 バイトシーケンスに変化します。 |

| 名称 | 長さ | 値 | 説明 |
|---|---|---|---|
| CRCL | 1 バイトまたは 2 バイト | 0x02 を除く 0x00…0xFF | パケット CRC の LSB。CRC-16 を使用。CRC は ADDR から最後のデータバイトまでのすべてのパケットバイトに対して計算されます。CRC は ESC 変更手順後に計算されます。このバイトが STX と等しい場合、(ESC) + (STX + 1) の 2 バイトシーケンスに変化します。このバイトが ESC と等しい場合、(ESC) + (ESC + 1) の 2 バイトシーケンスに変化します。 |

### 6.18.1 データ配信制御

通信手順は 3 つの部分に分けられます。

- リクエスト送信 (反対側がリクエスト受信)
- 応答待機 (反対側がリクエストを分析)
- 応答受信 (反対側が応答を送信)

「リクエスト送信」は、STX の送信, バイト変更手順によるデータ長とデータの送信, CRC の計算, および CRC の送信で構成されます。

「応答受信」は、STX を見つけて、CRC 計算を開始することから構成されます。受信したアドレスが無効な場合は、STX バイトの検索が繰り返されます。アドレスが有効な場合、データ長とデータバイトが受信されます。その後、CRC カウンターは停止し、2 つの CRC バイトが受信されます。これらのバイトは、計算された CRC 値と比較されます。

要求を送信した後、タイムアウト期間内に応答が受信されないかどうかを検出するために、ガードタイマーが開始されます。

### 6.18.2 PSOC™の実装

UART SCB コンポーネントは、信号を物理的に生成するために使用されます。ソフトウェア CRC-16 計算は、送受信されたすべてのバイトに適用されます (STX および CRC 自体を除く)。失敗したパケットトランザクションを検出するために、タイマーが使用されます。

このプロジェクトで実装されている 3 つの割込みは、完全な割込み駆動のバックグラウンドプロセスを提供します。

- UART の送信割込みは、FIFO not full イベントが RAM から新しいデータを取得して TX バッファに配置するように、また送信完了イベントが CRC 計算を開始または停止するように設定されます。
- UART の受信割込みは、FIFO not empty イベント用に設定され、受信データの解析, CRC の計算, および受信データの RAM への保存を行います。
- タイマー割込みは、失敗した送信の終了を検出するために使用されます。

このソフトウェアは、割込み駆動のドライバとして実装されます。つまり、ユーザーはプロセスを開始し、ユニットの状態を確認するだけです。すべてのセンサーをバックグラウンドでスキャンします。

### マスターのプロトコルユニットを操作するための 4 つの機能

#### 関数 1

```
void UartMesMaster_Init(CySCB_Type* uart_base, TCPWM_Type* counter_base,
                        uint32_t cntNum)
```

戻り値: なし

パラメータ:

- `uart_base` – マスター UART SCB インスタンスへのポインター
- `counter_base` – マスター TCPWM インスタンスへのポインター
- `cntNum` – 選択した TCPWM のカウンター インスタンス番号

ロケーション:

- *SelfTest _UART_master_message.h*
- *SelfTest _UART_master_message.c*

この関数は、UART メッセージユニットを初期化します。

#### 関数 2

```
uint8_t UartMesMaster_DataProc(uint8 address, char * txd, uint8 tlen, char *
rxd, uint8 rlen)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

パラメータ:

- `address` – データ転送用のスレーブアドレス
- `txd` – 送信データへのポインター
- `tlen` – 送信データのサイズ (バイト単位)
- `rxd` – 受信データが格納されるバッファへのポインター
- `rlen` – 受信データバッファのサイズ

ロケーション:

- *SelfTest _UART_master_message.h*
- *SelfTest _UART_master_message.c*

この関数は、メッセージの送受信プロセスを開始し、プロセス開始の結果 (0 = 成功、1 = エラー) を返します。エラーは、装置が既にメッセージの送信でビジー状態になっているか、またはヌル送信長が検出された場合に発生することがあります。

### 関数 3

```
uint8_t UartMesMaster_State(void)
```

戻り値:

- '0' (UM_ERROR) – 最後のトランザクションプロセスがエラーで終了し、ユニットは新しいプロセスを開始する準備ができています。
- '1' (UM_COMPLETE) – 最後のトランザクションプロセスが正常に終了し、受信したバッファに応答が含まれています。ユニットは新しいプロセスを開始する準備ができています。
- '2' (UM_BUSY) – ユニットはアクティブなトランザクション操作でビジー状態です。

ロケーション:

- *SelfTest _UART_master_message.h*
- *SelfTest _UART_master_message.c*

この関数は、UART メッセージユニットの現在の状態を返します。

考えられる結果は次のとおりです。

- **UM_ERROR:** 最後のトランザクションプロセスがエラーで終了し、ユニットは新しいプロセスを開始する準備ができています。
- **UM_COMPLETE:** 最後のトランザクションプロセスが正常に終了し、受信したバッファに応答が含まれています。ユニットは新しいプロセスを開始する準備ができています。
- **UM_BUSY:** ユニットはアクティブなトランザクション操作でビジー状態です。

### 関数 4

```
uint8_t UartMesMaster_GetDataSize(void)
```

戻り値: データサイズを返します

ロケーション:

- *SelfTest _UART_master_message.h*
- *SelfTest _UART_master_message.c*

この関数は、受信バッファに格納されている受信データサイズを返します。ユニットがビジー状態の場合、または最後のプロセスでエラーが発生した場合は、0 を返します。

## スレーブのプロトコルユニットを操作するための 5 つの機能

### 関数 1

```
void UartMesSlave_Init(CySCB_Type* uart_base, uint8_t address)
```

戻り値: なし

パラメータ:

- `uart_base` – スレーブ UART SCB インスタンスへのポインター
- `address` – スレーブバスアドレス

ロケーション:

- *SelfTest _UART_slave_message.h*

- *SelfTest _UART_slave_message.c*

UART メッセージユニットを初期化する関数です。入力パラメータは次のとおりです。

- Address: スレーブアドレス:

**関数 2**

```
uint8_t UartMesSlave_Respond(uint8_t * txd, uint8 tlen)
```

戻り値:

- '0' – エラーはありません
- '1' – エラーが検出されました

パラメータ:

- `txd` – 送信されたデータへのポインター
- `tlen` – 送信データのサイズ (バイト単位)

ロケーション:

- *SelfTest _UART_slave_message.h*
- *SelfTest _UART_slave_message.c*

この関数は応答を開始します。プロセス開始の結果を返します。成功は 0 で、エラーは 1 です (ユニットはマーカーを受け取っていません)。

入力パラメータは次のとおりです。

- `txd`: 送信されたデータ (リクエストデータ) へのポインター
- `tlen`: リクエストの長さ (バイト単位)

**関数 3**

```
uint8_t UartMesSlave_State(void)
```

戻り値:

- '0' (UM_IDLE) – 最後のトランザクションプロセスが終了しました
- '1' (UM_PACKREADY) –ユニットはマーカーを受け取り、バッファに受信したデータがあります。マスターは応答を待ちます。
- '2' (UM_RESPOND) – ユニットは応答の送信でビジーです。

ロケーション:

- *SelfTest _UART_slave_message.h*
- *SelfTest _UART_slave_message.c*

この関数は UART メッセージユニットの現在の状態を返します。考えられる結果は次のとおりです。

- **UM_IDLE:** 最後のトランザクションプロセスが終了しました。
- **UM_PACKREADY:** ユニットはマーカーを受け取り、バッファに受信したデータがあります。マスターは応答を待ちます。
- **UM_RESPOND:** ユニットは応答の送信でビジー状態です。

### 関数 4

`uint8_t UartMesSlave_GetDataSize(void)`

戻り値: データサイズを返します

ロケーション:

- *SelfTest _UART_slave_message.h*
- *SelfTest _UART_slave_message.c*

この関数は、受信バッファに格納された受信データサイズを取得します。ユニットの状態が UM_PACKREADY でない場合は、0 を返します。

### 関数 5

`uint8_t * UartMesSlave_GetDataPtr(void)`

戻り値: データへのポインターを返します

ロケーション:

- *SelfTest _UART_slave_message.h*
- *SelfTest _UART_slave_message.c*

この関数は、受信したデータへのポインターを取得します。

# 7 テストファイルの一覧

*Note:* *以下の表に示すファイルは、以前に認証されたバージョンの Class B ライブラリのファイルであり、AN236847 で入手できます。これらのファイルのバージョン/ MD5 ハッシュは、最新の mtb-stl ライブラリに対応していません。*

**Table 7 テストファイル**

| ファイル名 | バージョン | MD5 Hash HEX |
|---|---|---|
| *SelfTest_Analog.c* | 2.0 | 4DE4F05A48A83AB4EEA8F65AB6C52837 |
| *SelfTest_Analog.h* | 2.0 | 0CE91504AA03F1A1574951968F57A8EC |
| *SelfTest_Clock.c* | 2.0 | 19BD8BA39F135841DE0FDAF4371C665E |
| *SelfTest_Clock.h* | 2.0 | 635C7258BC068DA5737A9987B750CE76 |
| *SelfTest_ConfigRegisters.c* | 2.0 | 132610656813C5AF71C661DD869B2643 |
| *SelfTest_ConfigRegisters.h* | 2.0 | 102E012419ED441C6CB712DEC1795CDA |
| *SelfTest_CPU.c* | 2.0 | AA91A7068A395C88F03F333A2FEACFFF |
| *SelfTest_CPU.h* | 2.0 | 1DA7BDE5CB1C36C24201FD43131FCD24 |
| *SelfTest_CPU_Regs_GCC.s* | 2.0 | 3ACC8B37F9A9337E2DCE381EE8821305 |
| *SelfTest_CPU_Regs_IAR.s* | 2.0 | 09D5CB4B6F931FB1B32B0443D9860FD9 |
| *SelfTest_CPU_Regs.h* | 2.0 | 67E394266C180F03A9EB417C68C2D85B |
| *SelfTest_CRC_calc.c* | 2.0 | F1D39B0681B65B2C858194006C9CA323 |
| *SelfTest_CRC_calc.h* | 2.0 | 76CF6CB094006FD85CA0304EE3415A70 |
| *SelfTest_Flash.c* | 2.0 | B0EA5A9F14CD938341F6EAE17F389DF6 |
| *SelfTest_Flash.h* | 2.0 | D97934D4B6BD2F951ED9C7A71A524EC4 |
| *SelfTest_Interrupt.c* | 2.0 | 76A401EBAFB9B78C49F6160C1CA9B3FB |
| *SelfTest_Interrupt.h* | 2.0 | E6AF3F2ECAF346007BA01E69D2C61108 |
| *SelfTest_IO.c* | 2.0 | FC20FDB410949D724D254E94AD4DF501 |
| *SelfTest_IO.h* | 2.0 | 8407D43F8C1A7C8DDA2133AF7740131E |
| *SelfTest_RAM.c* | 2.0 | 0BC17EEA7D603EE04E7824535AB03B67 |
| *SelfTest_RAM.h* | 2.0 | 6D38C82A5E59DCF51B3FC5DDBDC595A8 |
| *SelfTest_SRAM_March_GCC.s* | 2.0 | 7D0BE45C70C18B30E42E5E93F2D24197 |
| *SelfTest_SRAM_March_IAR.s* | 2.0 | F14E5CA9C2CD5BEBFB79601FB65BE8EA |
| *SelfTest_RAM_Config_GCC.inc* | 2.0 | 5ED1FDCBBA4145ED73EF4D80302E1791 |
| *SelfTest_RAM_Config_IAR.inc* | 2.0 | 665FC0E6A58D63C70E52DCBE2971016E |
| *SelfTest_Stack.c* | 2.0 | E3036FFCAE03340DAB268CF979AFDE38 |
| *SelfTest_Stack.h* | 2.0 | 4C1871E98228FF4C93701C90339A1ACC |
| *SelfTest_UART_SCB.c* | 2.0 | 8835286EB479506326DE55F564856C78 |
| *SelfTest_UART_SCB.h* | 2.0 | A3DDE512B924F7EB852C52AE1660C57E |
| *SelfTest_I2C_SCB.c* | 2.0 | 1513172CBA33CC63772F39B3DCF7852D |
| *SelfTest_I2C_SCB.h* | 2.0 | 0D81FBBE1DEDB1927767266E340F1EDA |
| *SelfTest_SPI_SCB.c* | 2.0 | C219A9F3AB9E644578AABC335D5AF015 |

## テストファイルの一覧

| ファイル名 | バージョン | MD5 Hash HEX |
|---|---|---|
| *SelfTest_SPI_SCB.h* | 2.0 | 135C0D97BF5E6102E7202D7F6CC198B3 |
| *SelfTest_WDT.c* | 2.0 | 5702BD9DF5789F935E5C949045BD1052 |
| *SelfTest_WDT.h* | 2.0 | 2B71FB9B5011955CB8F9BF09377C8AE6 |
| *SelfTest_UART_slave_message.c* | 2.0 | 3CCA9666FF5A72D4B5CBC697E5D3295B |
| *SelfTest_UART_master_message.c* | 2.0 | 82EB11D06640F61148A8971BEAAE4FCF |
| *SelfTest_UART_Debug.c* | 2.0 | EB121120BB39A48B5FC9E568556BD0C9 |
| *SelfTest_UART_slave_message.h* | 2.0 | 5F0C214005CE0F5BC305D62C7750F071 |
| *SelfTest_UART_master_message.h* | 2.0 | 4961BC5D5E5CBAF7AC331A8BFAEEAAF2 |
| *SelfTest_UART_Debug.h* | 2.0 | FE20C7C33C6191270B23212F857A07CA |
| *SelfTest_TestConfiguration.h* | 2.0 | 80B0D9192726F41D7A163242DDBC180D |
| *Cy_capsense_selftest_v2.c* | 4.0 | 6D1B9AF86501F9AA57FC54F685E955E4 |
| *Cy_capsense_selftest_v2.h* | 4.0 | BB9FD9216117AFC3E7471725CE6725FB |
| *Cy_capsense_selftest_v3.c* | 4.0 | 7740A68DBDCA2E6172371790558F7452 |
| *Cy_capsense_selftest_v3.h* | 4.0 | D090D616E01D0A6656A766437C99E8D0 |
| *Cy_capsense_selftest.c* | 4.0 | D75F821B05BD0F0F2A872BCE759B7C05 |
| *Cy_capsense_selftest.h* | 4.0 | BBC6CAA165B01DA4BD494176F9315998 |

# 8 まとめ

このアプリケーションノートでは、IEC 60730 および IEC 61508 規格で定義された診断テストの実装方法について説明しました。これらの規格を白物家電やその他の電化製品の設計に組み込むことで、消費者の安全性が新たなレベルになります。

PSOC™ 4 が提供する独自のハードウェア設定機能を活用することで、設計者は電子システムのコストを維持または削減しながら、規制に準拠することができます。PSOC™とセーフティソフトウェアライブラリを使用することで、強力なシステムレベルの開発プラットフォームを作成し、優れた性能、市場投入までの時間の短縮、エネルギー効率を実現できます。

# 9 Appendix A: フラッシュ (不変メモリ) テストのチェックサムの設定

次の手順は、適切なフラッシュおよび ROM 診断テストのためにデバイスをプログラムするのに役立ちます。

1. *SelfTest_Flash.c* ファイルに保存されているチェックサム値を 0x0000 に設定して、ModusToolbox™ でプロジェクトをビルドしてください。

   GCC コンパイラの場合:

   ```
   static volatile const uint32_t flash_StoredCheckSum __attribute__((used, section(".flash_checksum")))= 0x00000000UL;
   ```

   IAR コンパイラの場合:

   ```
   #pragma location = ((uint32_t)(CY_FLASH_BASE + CY_FLASH_SIZE – 8u))
   static const uint32_t flash_StoredCheckSum = 0x00000000UL;
   ```

2. 計算されたフラッシュチェックサムを読み出してください。

a) デバッグモードでチェックサムを読み出してください。

b) プロジェクトファイル SelfTest_Flash.c を開いてください。そして、デバッグモードで、下の図に示す行にブレークポイントを設定してください。

c) [F10]を押してブレークポイントの位置を 1 ステップ送り変数「flash_CheckSum」の上にマウスを置きます。この変数に格納された値は、Figure 8 に示すように表示されます。

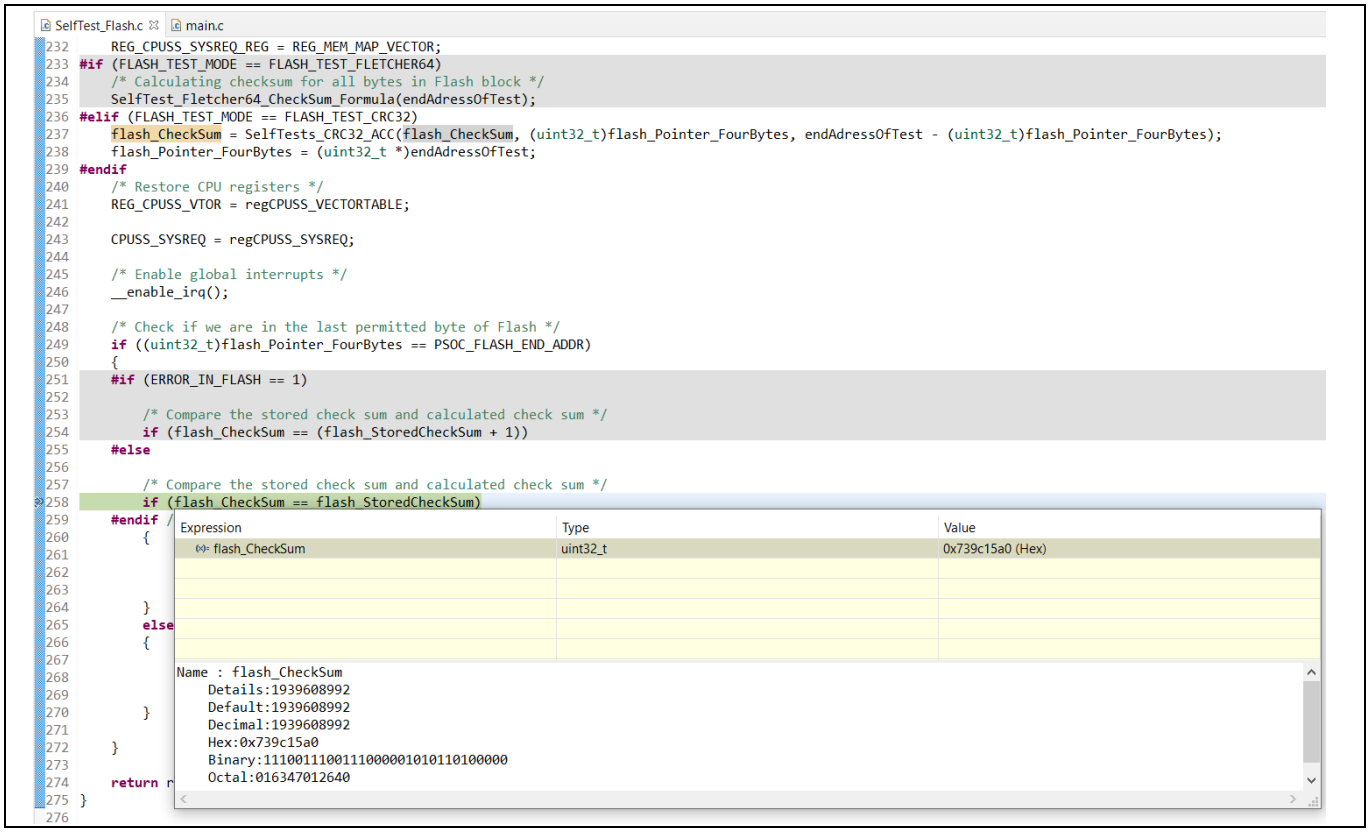


**Figure 8 デバッグモードでの格納済みチェックサム**

d) 通信プロトコルを使用してチェックサムを読み出してください。

- フラッシュチェックサム出力のテストプロセスを高速化するには、UART を使用します。この機能は、クラス B ファームウェアに実装されています。保存されたフラッシュチェックサムが計算されたフラッシュチェックサムと一致しない場合、計算されたチェックサム値を出力します。このプロジェクトを使用するには、Figure 8 に示す UART パラメータを設定してください。

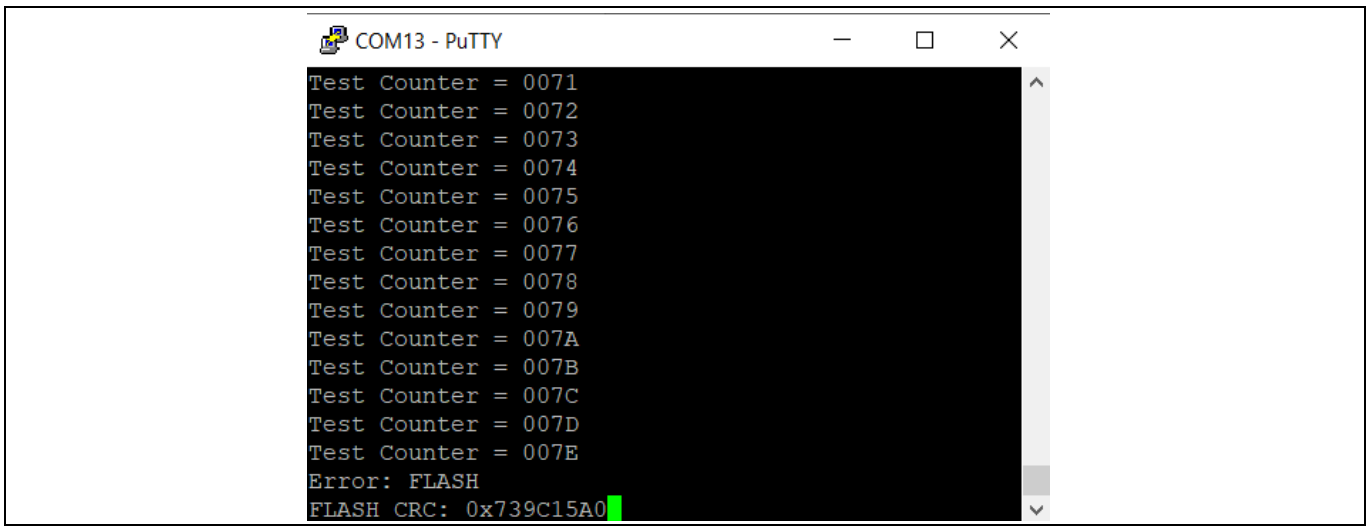


**Figure 9 UART を使用したチェックサム出力**

3. このチェックサム値をコピーして、*SelfTest_Flash.c* のチェックサムの場所に保存します。

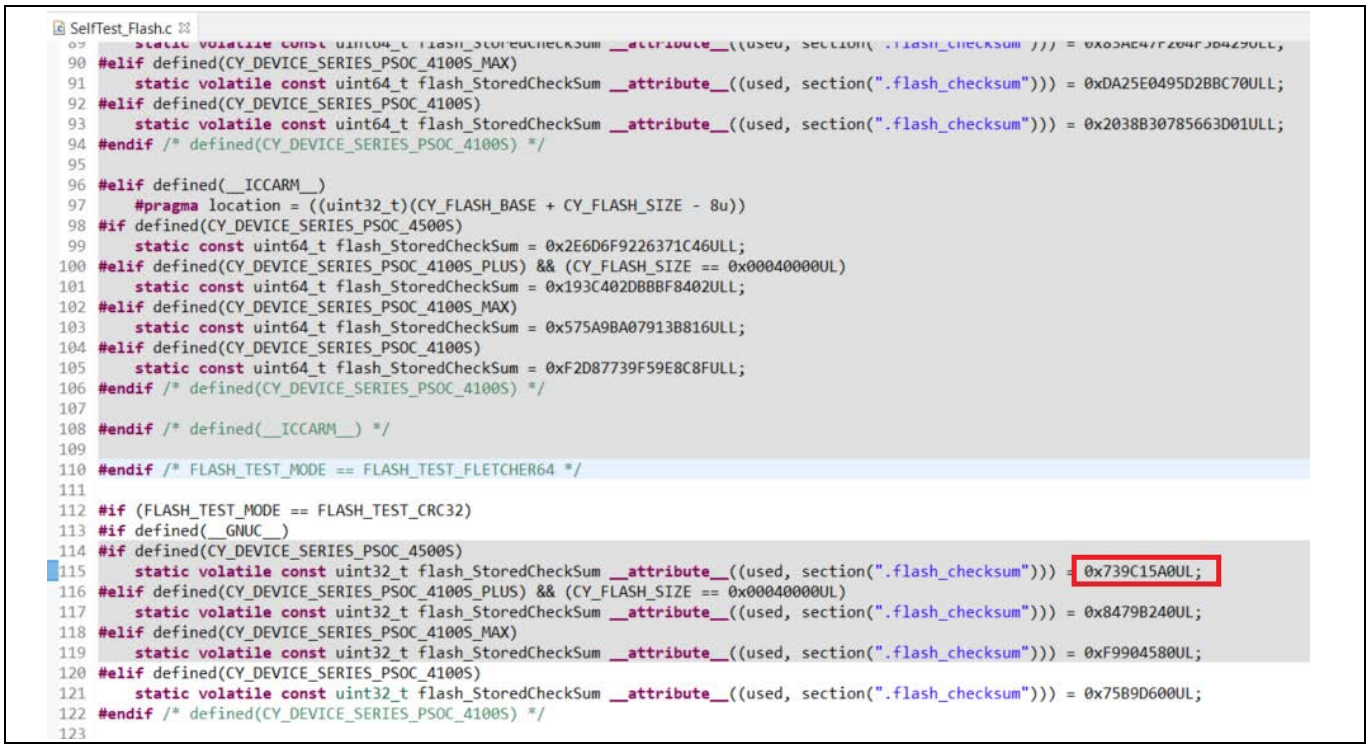


**Figure 10 チェックサム定数の実際のチェックサムへの再割り当て**

4. プロジェクトをコンパイルし、PSOC™ 4 デバイスをプログラムしてください。

# 10 Appendix B: IEC 60730-1 準拠証明書

*Note:* *このセクションに記載されている認定証は、AN236847 で入手できる古いバージョンの Class B ライブラリ用です。*



Ref. Certif. No.

US-42301-UL

IEC SYSTEM FOR MUTUAL RECOGNITION OF TEST CERTIFICATES FOR ELECTRICAL EQUIPMENT (IECEE) CB SCHEME

**CB TEST CERTIFICATE**

| | |
|---|---|
| Product | Software Libraries |
| Name and address of the applicant | CYPRESS SEMICONDUCTOR CORP<br>198 CHAMPION CT SAN JOSE, CA 95134-1709<br>United States |
| Name and address of the manufacturer | CYPRESS SEMICONDUCTOR CORP<br>198 CHAMPION CT SAN JOSE, CA 95134-1709<br>United States |
| Name and address of the factory | CYPRESS SEMICONDUCTOR CORP<br>198 CHAMPION CT SAN JOSE, CA 95134-1709<br>United States<br>☐ Additional Information on page 2 |
| Note: *When more than one factory, please report on page 2* | |
| Ratings and principal characteristics | Not Applicable |
| Trademark / Brand (if any) | Not Applicable |
| Customer's Testing Facility (CTF) Stage used | |
| Model / Type Ref. | Integrated, Protective Control with Type 2 action (Self-Test Software Library – Safety Control), PSoC 4 IEC 60730 Class B Safety Software Library (AN36847) |
| Additional information (if necessary may also be reported on page 2) | Only the requirements of IEC 60730-1, as documented in the accompanying test report, have been addressed.<br>☐ Additional Information on page 2 |
| A sample of the product was tested and found to be in conformity with | IEC 60730-1:2013, IEC 60730-1:2013/AMD1:2015, IEC 60730-1:2013/AMD2:2020 |
| As shown in the Test Report Ref. No. which forms part of this Certificate | 4790880026-20230619 issued on 2023-06-19 |

This CB Test Certificate is issued by the National Certification Body

UL Solutions

☑ UL Solutions (US), 333 Pfingsten Rd IL 60062, Northbrook, USA
☐ UL Solutions (Demko), Borupvang 5A DK-2750 Ballerup, DENMARK
☐ UL Solutions (JP), Marunouchi Trust Tower Main Building 6F, 1-8-3 Marunouchi, Chiyoda-ku, Tokyo 100-0005, JAPAN
☐ UL Solutions (CA), 7 Underwriters Road, Toronto, M1R 3B4 Ontario, CANADA

For full legal entity names see www.ul.com/ncbnames

Date: 2023-06-21

Signature: Jolanta M. Wroblewska

**Figure 11 コンプライアンス証明書**

## Appendix B: IEC 60730-1 準拠証明書

10 YEARS OF PRODUCT SAFETY

VDE INSTITUTE

VDE Prüf- und Zertifizierungsinstitut GmbH • Merianstr. 28 • 63069 Offenbach

Cypress Semiconductor Corporation
198 Champion Court
[redacted]
SAN JOSE CA 95134
USA

Offenbach, 2024-09-02

| Your ref. | Your letter | Our ref. - please indicate | Contact |
|---|---|---|---|
| [redacted] | 2023-10-20 | 5030935-4970-0001/313289<br>TL2/swa | [redacted] |

**Micro controller**
*Translation: In any case the German version shall prevail*

**P R Ü F B E R I C H T**
**zur Information des Auftraggebers**
***Test Report for the Information of the applicant***

**I Produkt / *Product*:**

**Typ / *Type*:** **PSoC™ 4 MCU familie/family**
**60730-1 Selbstdiagnose Bibliothek Klasse B und CPASENSE™ Test.**
**60730-1 Self diagnostic library class B and CAPSENSE™ test.**

Dear Sirs,

dieser Prüfbericht enthält das Ergebnis einer einmaligen Untersuchung an dem zur Prüfung vorgelegten Erzeugnis. Ein Muster dieses Erzeugnisses wurde geprüft, um die Übereinstimmung mit den nachfolgend aufgeführten Normen bzw. Abschnitten von Normen festzustellen.
Die Prüfung wurde durchgeführt vom 2024-05-05 to 2024-07-28.

*This test report contains the result of a singular investigation carried out on the product submitted. A sample of this product was tested to found the accordance with the thereafter listed standards or clauses of standards resp. The testing was carried out from* 2024-05-05 to 2024-07-28.

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618

Der Prüfbericht berechtigt Sie nicht zur Benutzung eines Zertifizierungszeichens des VDE und berücksichtigt ausschließlich die Anforderungen der unten genannten Regelwerke.

*The test report does not entitle for the use of a VDE Certification Mark and considers solely the requirements of the specifications mentioned below.*

Wenn gegenüber Dritten auf diesen Prüfbericht Bezug genommen wird, muss dieser Prüfbericht in voller Länge an gleicher Stelle verfügbar gemacht werden.

*Whenever reference is made to this test report towards third party, this test report shall be made available on the very spot in full length.*

**Beschreibung / Description**

**Bibliothek / Library**

Die Selbstdiagnose Methoden decken die Anforderungen der Tabelle H2 der 60730-1 Abschnitt 7.2 ab. Siehe Kapitel „Methoden“.

*The self-diagnose methods are covering the requirements of Table H2 of 60730-1 Clause 7.2. see chapter “methods”*

Die CAPSENSE™ Bibliothek der PSoC™4 Familie ist für die Erzeugung kapazitiver Sensorfelder vorgesehen.

The CAPSENSE library is *foreseen to be used to create capacitive sensor fields in conjunction with the micro controller family* PSoC™4

Das Paket bietet Methoden zur Erkennung und Behandlung typischer Fehler in kapazitiven Sensortastensteuerungen.

*The package provides methods to detect and handle typical failures in capacitive sensor button controls.*

**Methoden des Diagnosepakets**

- CPU Registertest
- Startup Configuration Register Test
- Program counter Test
- Programm Ablaufüberwachung
- Interrupt Überwachung
- Clock Test
- Flash Test
- SRAM Test
- March C Test
- Stack Test
- Digital I/O Test
- ADC Test
- Comparator Test
- Opamp Test
- UART Test

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618



2/9

- SPI Test
- Capsense™ tests*

**Methods of the diagnostic package**

- CPU register test
- Startup configuration register test
- Program counter Test
- Program flow test
- Interrupt test
- Clock test
- Flash Test
- SRAM Test
- March C Test
- Stack Test
- Digital I/O Test
- ADC Test
- Comparator Test
- Opamp Test
- UART Test
- SPI Test
- Capsense™ tests*

English only
*

**7.18 CAPSENSE™ test**

The CAPSENSE™ safety library includes several tests to verify the CAPSENSE™ components and proper functionality of related peripherals. This library is pre-certified by Infineon (per IEC 60730-1) and are available as included as part of the standard CAPSENSE™ component. This library includes tests for verifying:

- Sensor shorts
- External capacitor
- Integrity of CAPSENSE™ configuration data
- Integrity of sensor signal (raw count) and sensor baseline
- VDDA

| TABLE H.2 – MEASURES TO ADDRESS FAULT/ERRORS (Software Class B) | | | |
|---|---|---|---|
| Component | Fault/error | Declared measures | Verdict |
| 1. CPU | - | - | - |
| 1.1 Registers | Stuck at | CPU register test | P |
| 1.3 Program counter | Stuck at | Program counter test<br>Program flow test | P |
| 2. Interrupt handling and execution | No interrupt | Interrupt test | P |
| | Too frequent interrupt | Interrupt test | P |
| 3. Clock | Wrong frequency (for quartz synchronized clock: harmonics/ sub-harmonics only) | Clock test | P |
| 4. Memory | - | - | - |
| 4.1 Invariable memory | All single bit faults | FLASH Test | P |
| 4.2 Variable memory | DC fault | March C test | P |
| 4.3. Addressing (relevant to variable and invariable memory) | Stuck at | With 4.1 and 4.2 | P |
| 5. Internal data path | - | - | - |
| 5.1 Data | Stuck at | With 4.1 and 4.2 | P |
| 5.2 Addressing | Wrong address | With 4.1 and 4.2 | P |
| 6. External communication | - | - | - |
| 6.1 Data | Hamming distance 3 | Loopback tests for UART and SPI, further tests may be necessary in the final application. | P |
| 6.2 Addressing | Wrong address | N/A | N/A |
| 6.3 Timing | Wrong point in time | N/A | N/A |
| | Wrong sequence | N/A | N/A |
| 7. Input/output periphery | - | - | - |

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618



4/9

| | | | |
|---|---|---|---|
| 7.1 Digital I/O | Fault conditions specified in Cl.H.13 | Errors to Vcc and Gnd<br>Further failures may need to be considered in the final application | P |
| 7.2 Analog I/O | - | - | - |
| 7.2.1 A/D and D/A-convertor | Fault conditions specified in Cl. H.13 | ADC test | P |
| 7.2.2 Analog multiplexer | | | N/A |
| 9. Custom chips e.g. ASIC, GAL, gate array | Any output outside the static and dynamic functional specification | | N/A |

## Test File List

| File Name | Version | MD5 Hash HEX |
|---|---|---|
| SelfTest_Analog.c | 2.0 | 4DE4F05A48A83AB4EEA8F65AB6C52837 |
| SelfTest_Analog.h | 2.0 | 0CE91504AA03F1A1574951968F57A8EC |
| SelfTest_Clock.c | 2.0 | 19BD8BA39F135841DE0FDAF4371C665E |
| SelfTest_Clock.h | 2.0 | 635C7258BC068DA5737A9987B750CE76 |
| SelfTest_ConfigRegisters.c | 2.0 | 132610656813C5AF71C661DD869B2643 |
| SelfTest_ConfigRegisters.h | 2.0 | 102E012419ED441C6CB712DEC1795CDA |
| SelfTest_CPU.c | 2.0 | AA91A7068A395C88F03F333A2FEACFFF |
| SelfTest_CPU.h | 2.0 | 1DA7BDE5CB1C36C24201FD43131FCD24 |
| SelfTest_CPU_Regs_GCC.s | 2.0 | 3ACC8B37F9A9337E2DCE381EE8821305 |
| SelfTest_CPU_Regs_IAR.s | 2.0 | 09D5CB4B6F931FB1B32B0443D9860FD9 |
| SelfTest_CPU_Regs.h | 2.0 | 67E394266C180F03A9EB417C68C2D85B |
| SelfTest_CRC_calc.c | 2.0 | F1D39B0681B65B2C858194006C9CA323 |
| SelfTest_CRC_calc.h | 2.0 | 76CF6CB094006FD85CA0304EE3415A70 |
| SelfTest_Flash.c | 2.0 | B0EA5A9F14CD938341F6EAE17F389DF6 |
| SelfTest_Flash.h | 2.0 | D97934D4B6BD2F951ED9C7A71A524EC4 |
| SelfTest_Interrupt.c | 2.0 | 76A401EBAFB9B78C49F6160C1CA9B3FB |
| SelfTest_Interrupt.h | 2.0 | E6AF3F2ECAF346007BA01E69D2C61108 |
| SelfTest_IO.c | 2.0 | FC20FDB410949D724D254E94AD4DF501 |
| SelfTest_IO.h | 2.0 | 8407D43F8C1A7C8DDA2133AF7740131E |
| SelfTest_RAM.c | 2.0 | 0BC17EEA7D603EE04E7824535AB03B67 |
| SelfTest_RAM.h | 2.0 | 6D38C82A5E59DCF51B3FC5DDBDC595A8 |
| SelfTest_SRAM_March_GCC.s | 2.0 | 7D0BE45C70C18B30E42E5E93F2D24197 |
| SelfTest_SRAM_March_IAR.s | 2.0 | F14E5CA9C2CD5BEBFB79601FB65BE8EA |
| SelfTest_SRAM_Config_GCC.inc | 2.0 | 5ED1FDCBBA4145ED73EF4D80302E1791 |
| SelfTest_SRAM_Config_IAR.inc | 2.0 | 665FC0E6A58D63C70E52DCBE2971016E |
| SelfTest_Stack.c | 2.0 | E3036FFCAE03340DAB268CF979AFDE38 |
| SelfTest_Stack.h | 2.0 | 4C1871E98228FF4C93701C90339A1ACC |
| SelfTest_UART_SCB.c | 2.0 | 8835286EB479506326DE55F564856C78 |
| SelfTest_UART_SCB.h | 2.0 | A3DDE512B924F7EB852C52AE1660C57E |

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618

Page 7 - 02.09.2024 Our reference 5030935-4970-0001/313289
TL2/swa

| | | |
|---|---|---|
| SelfTest_I2C_SCB.c | 2.0 | 1513172CBA33CC63772F39B3DCF7852D |
| SelfTest_I2C_SCB.h | 2.0 | 0D81FBBE1DEDB1927767266E340F1EDA |
| SelfTest_SPI_SCB.c | 2.0 | C219A9F3AB9E644578AABC335D5AF015 |
| SelfTest_SPI_SCB.h | 2.0 | 135C0D97BF5E6102E7202D7F6CC198B3 |
| SelfTest_WDT.c | 2.0 | 5702BD9DF5789F935E5C949045BD1052 |
| SelfTest_WDT.h | 2.0 | 2B71FB9B5011955CB8F9BF09377C8AE6 |
| UART_slave_message.c | 2.0 | 3CCA9666FF5A72D4B5CBC697E5D3295B |
| UART_master_message.c | 2.0 | 82EB11D06640F61148A8971BEAAE4FCF |
| UART_Debug.c | 2.0 | EB121120BB39A48B5FC9E568556BD0C9 |
| UART_slave_message.h | 2.0 | 5F0C214005CE0F5BC305D62C7750F071 |
| UART_master_message.h | 2.0 | 4961BC5D5E5CBAF7AC331A8BFAEEAAF2 |
| UART_Debug.h | 2.0 | FE20C7C33C6191270B23212F857A07CA |
| TestConfiguration.h | 2.0 | 80B0D9192726F41D7A163242DDBC180D |
| Cy_capsense_selftest_v2.c | 4.0 | 6D1B9AF86501F9AA57FC54F685E955E4 |
| Cy_capsense_selftest_v2.h | 4.0 | BB9FD9216117AFC3E7471725CE6725FB |
| Cy_capsense_selftest_v3.c | 4.0 | 7740A68DBDCA2E6172371790558F7452 |
| Cy_capsense_selftest.c | 4.0 | D75F821B05BD0F0F2A872BCE759B7C05 |
| Cy_capsense_selftest.h | 4.0 | BBC6CAA165B01DA4BD494176F9315998 |

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618

Page 8 - 02.09.2024 Our reference 5030935-4970-0001/313289
TL2/swa

**II Angewandte Normen / *Relevant standards***

IEC 60730-1:2013
IEC 60730-1:2013/AMD1:2015 IEC 60730-1:2013/AMD2:2020
DIN EN 60730-1 (VDE 0631-1):2021-06; EN 60730-1:2016+A1:2019
EN 60730-1:2016/A2:2022
(Anhang H für Software Klasse B)
*(Annex H for software class B)*

IEC 60335-1:2010
IEC 60335-1:2010/AMD1:2013 IEC 60335-1:2010/AMD2:2016
DIN EN 60335-1 (VDE 0700-1):2020-08;
EN 60335-1:2012+AC+A11+A13+A1+A2+A14:2019
EN 60335-1:2012/A15:2021
(Anhang R für Software Klasse R1)
(Annex R for software class R1)

**III PRÜFUNG / TEST**

Die unter I genannten Methoden wurden unter Zuhilfenahme einer speziellen Testumgebung überprüft.
*The methods mentioned under I have been verified with the aid of a specifically designed test environment.*

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618

8/9

8/9

Page 9 - 02.09.2024 Our reference 5030935-4970-0001/313289
TL2/swa

**IV ERGEBNIS / *RESULT***

Die unter I benannten Selbst-Diagnose-Routinen erfüllen die Anforderungen der unter II benannten Normen. Die unter I benannten Selbst-Diagnose-Routinen können zum Aufbau einer Selbst-Test Bibliothek gemäß der unter II benannten Normen verwendet werden.

*The self-diagnostic routines mentioned under I fulfill the requirements of the standards mentioned under II. The self-diagnostic routines mentioned under I are suitable to be used to create a self-test library according the standards mentioned under II.*

Mit freundlichen Grüßen

.....
[redacted]

.....
[redacted]

VDE Prüf- und Zertifizierungsinstitut GmbH
Merianstr. 28
63069 Offenbach, Germany
Phone +49 69 8306-0
Fax +49 69 8306-555
www.vde.com/institute
vde-institut@vde.com
Commerzbank AG Frankfurt
IBAN DE91 5008 0000 0198 0270 00
BIC DRES DE FF XXX
VAT-ID DE 261 922 990
Tax number 04425092566
Managing Directors:
Ansgar Hinz
Dr. jur. Beate Mand
Location: Offenbach
Registered at the local court Offenbach am Main,
No. HRB 43618

9/9

9/9

**Figure 12 VDE コンプライアンスレポート**

# 11 Appendix C: サポートされている製品番号

**Table 8 認定ライブラリによりサポートされている PSOC™ 4 ファミリ**

| デバイス ファミリ | データシート | 版数 |
|---|---|---|
| PSOC™ 4100S | PSOC™ 4 MCU: PSOC™ 4100S | *Q |
| PSOC™ 4100S Max | PSOC™ 4 MCU: PSOC™ 4100S Max | *F |
| PSOC™ 4100S Plus 256 KB | PSOC™ 4 MCU: PSOC™ 4100S Plus 256 KB | *H |
| PSOC™ 4500S | PSOC™ 4 MCU: PSOC™ 4500S | *E |

サポートされている製品番号の詳細については、対応するデバイスデータシートの「注文情報」セクションを参照してください。

# 12 Appendix D. MISRA 準拠

この Appendix の Table 9 と Table 10 に、テストプロジェクトの MISRA-C:2004 準拠と逸脱の詳細を示します。

自動車業界ソフトウェア信頼性協会 (MISRA) 仕様は、ファームウェア設計に適用される 122 の必須ルールと 20 の推奨ルールを網羅しています。自動車業界は、車載デバイスに組み込まれたファームウェアコードの品質と堅牢性を向上させるために、この仕様をまとめました。

**Table 9 検証環境**

| コンポーネント | 名称 | バージョン |
|---|---|---|
| テスト仕様 | MISRAC:2012 Guidelines for use of C language in critical systems (重要なシステムでの C 言語の使用に関するガイドライン) | 第 3 版、2019 年 2 月 |
| 対象デバイス | PSOC™ 4100S、PSOC™ 4100S Max、PSOC™ 4100S Plus 256 KB、PSOC™ 4500S | Production |
| 対象コンパイラ | GCC | v10.3.1 |
| | IAR | v9.30.1 |
| 生成ツール | ModusToolbox™ | v3.0 |
| Peripheral Driver Library | PDL | mtb-pdl-cat2 V2.1.0 |
| MISRA チェックツール | Coverity 静的解析ツール | 2021.9.0 |

**Table 10 逸脱したルール**

| MISRA-C:2012 ルール | ルールクラス (R/A) | ルールの説明 | 逸脱の説明 |
|---|---|---|---|
| 1.2 | A | 言語拡張機能は使用しないでください。 | 低レベルコンパイラ固有のコマンドを使用する必要があります。サポートされている各コンパイラで正しいことを確認しました。 |
| 2.5 | A | プロジェクトには未使用のマクロ宣言を含めないでください | ライブラリのすべての機能を同時に有効にできるわけではないため、無効なテストに必要なマクロは未使用として表示されます。 |
| 3.1 | R | 文字シーケンス /* と // は、コメント内で使用しないでください。 | 「/」および「//」とともに使用されるハイパーリンク。ライブラリの文書化に使用するためのコメントにのみ含まれます。コードでは使用されません。 |
| 4.9 | A | 互換性がある関数のようなマクロよりも、関数を優先して使用する必要があります。 | プラットフォーム PDL の使用のため、マクロのような機能の使用は避けられません。 |
| 8.6 | R | 外部リンケージを持つ識別子は、外部定義を 1 つだけ持つ必要があります。 | C で宣言されていますが、定義されていません。アセンブリファイルで定義されています。 |

| MISRA-C:2012 ルール | ルールクラス (R/A) | ルールの説明 | 逸脱の説明 |
|---|---|---|---|
| 8.7 | A | 関数とオブジェクトは、1 つの翻訳単位でのみ参照される場合、外部リンケージで定義しないでください。 | グローバル変数は、安全ライブラリとユーザープロジェクトの統合に基づいて可能な限り削除します。 |
| 8.9 | A | オブジェクトの識別子が単一の関数にのみ出現する場合は、そのオブジェクトをブロックスコープで定義する必要があります。 | スコープは、ユーザープロジェクトでのライブラリ統合に依存します。 |
| 8.13 | A | ポインターは、可能な限り const 修飾された型を指す必要があります。 | 誤検知。 |
| 11.4 | A | オブジェクトへのポインターと整数型の間で変換は実行しないでください | ポインターを整数にキャストしたり、その逆を行ったりすることは、PDL の使用要件により避けられません。 |
| 13.2 | R | 式の値とその永続的な副作用は、許可されているすべての評価順序で同じになります。 | 評価で volatile 変数を複数回使用します。volatile 変数が評価ごとに 1 つ以上の演算を持つのは ISR にあるため、これは問題ありません。 |
| 13.5 | R | 論理 && または||演算子の右側のオペランドには、永続的な副作用が含まれてはいけません。 | 比較して volatile 変数を使用することは、ISR でのテストのために避けられません。 |
| 14.2 | R | for ループは適切に形成されなければいけません | 誤検知。 |
| 14.3 | R | 制御式は不変ではありません | ブール式は PDL の使用により常に true と評価されることは避けられません |

# 参考資料

[1] IEC 60730 Standard: *Automatic electrical controls, IEC 60730-1 Edition 6.0, 2022-09;* オンラインで購入可能

[2] IEC 61508 Standard: *Functional safety of electrical/electronic/programmable electronic safety-related systems, IEC 61508-2 Edition 2.0, 2010-04*; オンラインで購入可能

# 改訂履歴

| 版数 | 発行日 | 変更内容 |
|---|---|---|
| ** | 2025-03-05 | 本版は英語版 002-36847 Rev. *B について、インフィニオン DEVELOPER COMMUNITY の参画者によって日本語に翻訳されたドキュメントです。 |
| *A | 2025-04-11 | これは英語版 002-36847 Rev. *D を翻訳した日本語版 Rev. *A です。 |

**Trademarks**
All referenced product or service names and trademarks are the property of their respective owners.
PSOC™, formerly known as PSoC™, is a trademark of Infineon Technologies. Any references to PSoC™ in this document or others shall be deemed to refer to PSOC™.

**Edition 2025-04-11**
**Published by**
**Infineon Technologies AG**
**81726 Munich, Germany**

**© 2025 Infineon Technologies AG. All Rights Reserved.**

**Do you have a question about this document?**
**Email:** erratum@infineon.com

**Document reference**
**002-41282 Rev. *A**

**重要事項**

本手引書に記載された本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

**警告事項**

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。
インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。