

Errata

MSPM0L110x、MSPM0L13xx マイコン



概要

この文書では、機能仕様に対する既知の例外 (アドバイザリ) について説明します。

目次

1 エラッタに関する注記.....	1
2 機能アドバイザリ.....	1
3 プログラム済みのソフトウェア アドバイザリ.....	3
4 デバッグ専用のアドバイザリ.....	3
5 コンパイラ アドバイザリによって修正.....	3
6 デバイスの命名規則.....	3
6.1 デバイスの記号表記とリビジョンの識別.....	4
7 アドバイザリの説明.....	5
8 改訂履歴.....	36

1 エラッタに関する注記

エラッタのリストおよび各項目を読む前に、以下の注意事項を確認してください:

- ADC_ERR_04 はこのデバイスに影響を与えなくなりました
- FLASH_ERR_04 はこのデバイスに影響を与えなくなりました

2 機能アドバイザリ

デバイスの動作、機能、またはパラメータに影響するアドバイザリ。

✓ チェック マークは、指定されたリビジョンに問題が存在することを示します。

エラッタ番号	Rev C	Rev D
ADC_ERR_01	✓	✓
ADC_ERR_02	✓	✓
ADC_ERR_03	✓	✓
ADC_ERR_04	✓	
ADC_ERR_05	✓	✓
ADC_ERR_06	✓	✓
ADC_ERR_11	✓	✓
COMP_ERR_01	✓	✓
COMP_ERR_02	✓	✓
COMP_ERR_03	✓	✓
COMP_ERR_05	✓	✓
CPU_ERR_01	✓	✓
CPU_ERR_02	✓	✓
CPU_ERR_03	✓	✓
CPU_ERR_04	✓	✓
CRC/CRCP_ERR_01	✓	✓
FCC_ERR_01	✓	✓

エラッタ番号	Rev C	Rev D
FLASH_ERR_02	✓	✓
FLASH_ERR_03	✓	✓
FLASH_ERR_04	✓	
FLASH_ERR_05	✓	✓
FLASH_ERR_06	✓	✓
FLASH_ERR_08	✓	✓
GPIO_ERR_01	✓	✓
GPIO_ERR_02	✓	✓
GPIO_ERR_03	✓	✓
GPIO_ERR_04	✓	✓
GPIO_ERR_06	✓	✓
I2C_ERR_01	✓	✓
I2C_ERR_02	✓	✓
I2C_ERR_03	✓	✓
I2C_ERR_04	✓	✓
I2C_ERR_05	✓	✓
I2C_ERR_06	✓	✓
I2C_ERR_07	✓	✓
I2C_ERR_08	✓	✓
I2C_ERR_09	✓	✓
I2C_ERR_10	✓	✓
I2C_ERR_13	✓	✓
I2C_ERR_15	✓	✓
PMCU_ERR_01	✓	✓
PMCU_ERR_02	✓	✓
PMCU_ERR_03	✓	✓
PMCU_ERR_10	✓	✓
PMCU_ERR_13	✓	✓
PWREN_ERR_01	✓	✓
RST_ERR_01	✓	✓
SPI_ERR_01	✓	✓
SPI_ERR_03	✓	✓
SPI_ERR_04	✓	✓
SPI_ERR_05	✓	✓
SPI_ERR_06	✓	✓
SPI_ERR_07	✓	✓
SPI_ERR_08	✓	✓
SPI_ERR_09	✓	✓
SPI_ERR_10	✓	✓
SYSCTL_ERR_03	✓	✓
SYSCTL_ERR_05	✓	✓
SYSCTL_ERR_06	✓	✓
SYSCTL_ERR_11	✓	✓
SYSOSC_ERR_01	✓	✓
SYSOSC_ERR_02	✓	✓
SYSOSC_ERR_04	✓	✓

エラー番号	Rev C	Rev D
SYSOSC_ERR_05	✓	✓
SYSOSC_ERR_06	✓	✓
TIMER_ERR_01	✓	✓
TIMER_ERR_04	✓	✓
TIMER_ERR_06	✓	✓
TIMER_ERR_07	✓	✓
UART_ERR_01	✓	✓
UART_ERR_02	✓	✓
UART_ERR_04	✓	✓
UART_ERR_05	✓	✓
UART_ERR_06	✓	✓
UART_ERR_07	✓	✓
UART_ERR_08	✓	✓
UART_ERR_09	✓	✓
UART_ERR_10	✓	✓
UART_ERR_11	✓	✓
VREF_ERR_01	✓	✓

3 プログラム済みのソフトウェア アドバイザリ

工場出荷時にプログラムされたソフトウェアに影響を及ぼすアドバイザリ。

✓ チェック マークは、指定されたリビジョンに問題が存在することを示します。

4 デバッグ専用のアドバイザリ

デバッグ動作のみに影響するアドバイザリ。

✓ チェック マークは、指定されたリビジョンに問題が存在することを示します。

5 コンパイラ アドバイザリによって修正

コンパイラの回避方法により解決されるアドバイザリ各アドバイザリについては、回避策が適用されている IDE およびコンパイラのバージョンを参照してください。

✓ チェック マークは、指定されたリビジョンに問題が存在することを示します。

6 デバイスの命名規則

製品開発サイクルの段階を示すため、TI はすべての MSP MCU デバイスの型番に接頭辞を割り当てています。MSP MCU 商用ファミリの各番号には、MSP、X のいずれかの接頭辞があります。MSP または XMS。これらの接頭辞は、製品開発の進展段階を表します。段階には、エンジニアリング プロトタイプ(XMS)から、完全認定済みの量産デバイス(MSP)までがあります。

XMS - 実験段階のデバイスであり、必ずしも最終製品の電気的特性を表しているとは限りません

MSP - 完全に認定済みの量産版デバイス

サポートツールの名前付けプレフィックス:

X: 開発サポート製品。テキサス・インスツルメンツの社内認定試験はまだ完了していません。

null: 完全に認定済みの開発サポート製品です。

XMS デバイスと **MSPX** 開発サポート ツールは、以下の免責事項に基づいて出荷されます:

「開発中の製品は、社内での評価用です。」

MSP デバイスの特性は完全に明確化されており、デバイスの品質と信頼性が十分に示されています。テキサス・インスツルメンツの標準保証が適用されます。

プロトタイプ デバイス (XMS) は、標準の量産デバイスよりも故障率が高いことが予想されます。これらのデバイスは、予測される最終使用時の故障率が未定義であるため、テキサス・インスツルメンツはそれらのデバイスを量産システムで使用しないよう推奨しています。認定済みの量産デバイスのみを使用する必要があります。

TI デバイスの項目表記には、デバイス ファミリ名の接尾辞も含まれます。この接尾辞は、温度範囲、パッケージ タイプ、配布形式を示しています。

6.1 デバイスの記号表記とリビジョンの識別

次のパッケージ図はパッケージ記号化スキームを示すとともに表 6-1、デバイスリビジョンからバージョン ID へのマッピングを定義しています。

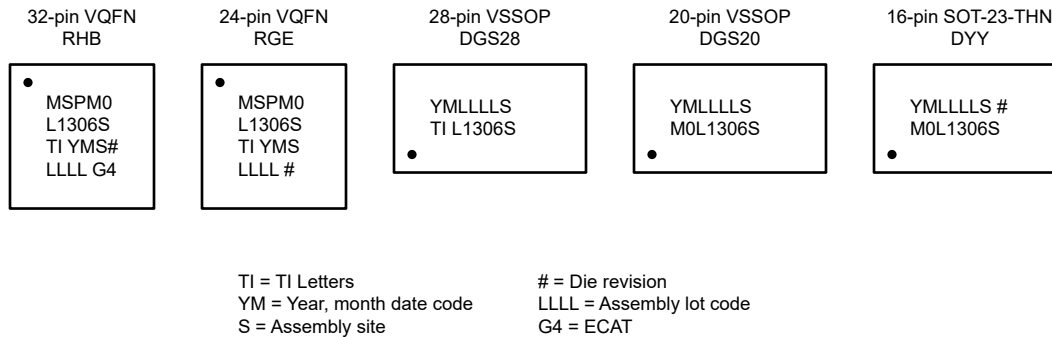


図 6-1. パッケージの記号表記

表 6-1. ダイ リビジョン

リビジョンレター	バージョン (デバイスの工場出荷時定数メモリ内)
C	1
D	1

リビジョン文字は、製品のハードウェアの改訂版を示します。このドキュメントのアドバイザーには、リビジョン文字に基づいて、特定のバースに該当するか否かがマークされています。この文字は、デバイスのメモリに保存された整数にマップされ、アプリケーションソフトウェア または接続されたデバッグプローブによるリビジョンの検索に使用できます。

7 アドバイザリの説明

ADC_ERR_01 ADC モジュール

カテゴリ

機能

機能

ADC は STANDBY1 モードでは高速クロックをトリガできません

説明

デバイスが STANDBY1 モードで動作している場合、ADC モジュールがイベント システム経由でトリガされた際 (たとえば、タイマなどのイベント パブリッシャが ADC のイベント サブスクライバ ポートを通じてイベントを生成した場合)、非同期の高速クロック要求が正しくアサートされないことがあります。

回避方法

イベント ファブリック経由で ADC 変換をトリガする場合は、STANDBY0 以上の電力モードを使用してください。

ADC_ERR_02 ADC モジュール

カテゴリ

機能

機能

ADC が反復モードのときの低消費電力モードでの電流の増加

説明

ADC が低電力モード (LPM) で EVENT トリガを使用し、リピート シーケンス モードまたはリピート シングル モードに設定されている場合、ADC モジュールは無効化されるまで追加の電流を消費し続けます。表示される電流の値は、データシートの SLEEP0 電流特性と同等です。

回避方法

回避方法 1: ADC をリピートなしのシングル モードまたはシーケンス モードに設定し、変換後に割り込みを使用して次のトリガに備えて ADC を再有効化します。回避方法 2: LPM モードに移行するときに、ADC をリセットまたは ADC の構成を変更します。LPM に移行するときは反復モードを使用しないことをお勧めします。

ADC_ERR_03 ADC モジュール

カテゴリ

機能

機能

ADC グランド基準に VSSA を使用する場合、ADC の SNR と DNL 性能が低下します。

説明

ADC がグランド基準として VSS を使用している場合、VSS グランドのノイズ源は ADC 結果に影響を及ぼし、SNR と DNL の性能が低下します。

回避方法

回避方法 1: ADC リファレンスには VREF+ ピンと VREF- ピンを使用します。VREF ピンを使用すると、ADC はアナログ グランド基準電圧を備えているため、外部ノイズ源の影響は減少します。回避方法 2: VSS のノイズ源を低減します。一般的な原因は CPU またはデジタル スイッチング ノイズです

ADC_ERR_04	ADC モジュール
カテゴリ	機能
機能	ADC が反復モードのときの低消費電力モードでの電流の増加
説明	ADC が低電力モード (LPM) で EVENT トリガを使用し、リピート シーケンス モードまたはリピート シングル モードに設定されている場合、ADC モジュールは無効化されるまで追加の電流を消費し続けます。
回避方法	ADC をリピートなしのシングル モードまたはシーケンス モードに設定し、変換後に割り込みを使用して次のトリガに備えて ADC を再有効化します。
ADC_ERR_05	ADC モジュール
カテゴリ	機能
機能	IP(周辺モジュール) が有効化される前にハードウェアイベントが生成された場合、その ADC トリガはキューに保持されたままになります
説明	ADC が HW イベントトリガを受信すると、CTL0.ENC = 0x0 であっても、保留中のイベントは ADC トリガキューに入ります。ADC が CTL1.TRIGSRC = 0x1 (HW トリガ) かつ CTL0.ENC = 1 に設定されると、キューに設定された HW トリガは、ADC サンプルングおよび変換プロセスを開始します。保留中の HW 要求は、CTL1.TRIGSRC = 0x0 (SW トリガ) の場合でもキューに入ることができますが、CTL1.TRIGSRC = 0x1 かつ CTL0.ENC = 0x1 の場合のみ ADC サンプルングを開始します。
回避方法	HW トリガを使用することを予期した場合にのみ ADC F_SUB を構成し、そうでない場合、保留中の要求がキューに登録されます。SW トリガモードと HW トリガモードを切り替える場合、ADC (RSTCTL) をリセットすると保留中のキューがクリアされますが、ADC の再構成が必要となります。
ADC_ERR_06	ADC モジュール
カテゴリ	機能
機能	ADC 出力コードが DNL/INL の仕様の劣化を招きます
説明	変換エラーが発生すると、ADC の入力電圧に対応する変化がないにもかかわらず、デジタル出力コード上に $\pm 64\text{LSB}$ の固定的なジャンプとしてそのエラーが現れます。 最悪条件下 (-40°C) においては、12 ビット モードで変換された 12M 回のサンプルにつき 1 回、エラーが発生します。

ADC_ERR_06 (続き) ADC モジュール

0°C時のエラーレートは 24M 回に 1 回、55°C時は 60M 回に 1 回です (使用された VDD 電圧と基準電圧はエラー レートに影響しない)。

回避方法

アプリケーションのニーズに応じて、最適な回避策は異なりますが、本ソフトウェアでは、以下の回避策を提案します。最適な回避方法の選択は、システム設計者の判断に委ねられます。

回避方法 1: ADC の結果がアプリケーションで設定されたスレッシュホールドを外れた場合 (ADC ウィンドウ コンパレータやソフトウェアによるスレッシュホールド判定を使用)、重要なシステム判断を行う前に、もう一度 ADC の結果を取得するか、次の変換結果を待つようにします

回避方法 2: 後処理時に、ADC 値の中央値または予測値からかけ離れた ADC 値を破棄します。期待値は、システム内で取得された実際のサンプルの平均に基づいて設定し、除外のためのスレッシュホールドは、測定されたシステム ノイズの大きさに基づいて決定する必要があります。

回避方法 3: 単一の誤変換結果の影響を最小限に抑えるために、ADC サンプルの平均化を使用します。

ADC_ERR_11

ADC モジュール

カテゴリ

機能

機能

MEMCTL.TRIG = 0x1 の反復モードでは、ADC ビジーは High のままで、ADC ENC はクリアされる

説明

MEMCTL.TRIG = 0x1 でシングルまたはリピート シーケンス モードの場合 (次の変換にはトリガが必要)。ADC は新しいトリガを待機しているため、ADC はビジー状態のままです。これにより、明示的に 0 を書き込んでも ENC ビットはクリアされません。

回避方法

回避方法 1: ADC ペリフェラル (ADCx.RSTCTL = 0xB1000003) をリセットして、ビジー ステータスと ENC ビットをクリアします。以下を再設定する必要があります:

回避策 2: ADC をシングルモード ADCx.CTL1.CONSEQ = 0x0 または 0x1 に設定します。次に、最後のメモリ制御レジスタ MEMCTL[Y].TRIG を 0x0 に設定します (次の変換は自動です)。ADC は次のチャンネルをサンプリングしますが、その後は繰り返されません。これにより、ビジー ステータスと ENC がクリアされます。

COMP_ERR_01

COMP モジュール

カテゴリ

機能

機能

コンパレータ出力は、STANDBY0 モードでトグルを維持します

COMP_ERR_01 (続き)

COMP モジュール

説明

コンパレータの反転入力マルチプレクサ (IMSEL) が 0 (COMP0_IN0-) に設定されており、かつデバイスの動作モードが STANDBY0 に設定されている場合、コンパレータの出力が入力電圧に関係なく予期せずトグルする可能性があります。

回避方法

STANDBY0 モードでコンパレータを使用し、負入力端子に外部電圧を印加する場合は、コンパレータのチャンネル 1 (IMSEL = 1) を使用してください。

STANDBY0 モードでコンパレータを使用し、内部リファレンス電圧(DAC8、VREF など) を負の入力に適用する場合は、IMSEL を 0 以外の値に設定してください (IMSEL 0)。

COMP_ERR_02

COMP モジュール

カテゴリ

機能

機能

DACCODEx をヒステリシスに使用すると、COMP 出力がトグルします

説明

COMP モジュール内の 8 ビット DAC を COMP の入力として使用し、DAC の出力が DACCODEx に設定された値間で切り替わる場合、COMP の出力は、新しい基準値を直ちに上回ったかのようにトグル動作します。

これは、COMPx.CTL2.DACCTL ビットの設定に関係なく発生します。

この現象は、COMP モジュールにカスタムまたは非対称のヒステリシスを実装する目的で、2 つの DACCODEx コードを使用しているアプリケーションで最も一般的に見られます。

回避方法

COMPx.CTL1.HYST レジスタ ビットに設定されているヒステリシス値を利用します。

COMP_ERR_03

COMP モジュール

カテゴリ

機能

機能

入力交換機能を使用する場合、COMP ヒステリシス機能は機能しません

説明

COMP モジュールのヒステリシス機能を使用している状態で、COMP の入力を入れ替える (COMPx.CTL1.EXCH = 1) と、COMP モジュールが不安定になる可能性があります。

回避方法

入力交換機能で COMP モジュールを利用する場合は、内部ヒステリシスの方法を適用しないでください。

COMP_ERR_05

COMP モジュール

カテゴリ

機能

機能

コンパレータが有効になると、出力により立ち上がりおよび立ち下がり割り込みが設定されます

概要

コンパレータが有効なとき、コンパレータによって立ち上がりと立ち下がりが設定されます。

回避方法

1. ICLR ビットを使用して CPU 割り込みをクリアします。
ICLR は一般的イベントのクリアには機能しません。COMP の汎用イベントをクリアするには、以下の手順に従ってください(以下は DriverLib 関数です。当社の MSPM0 SDK の機能の内容を確認して、ビット操作を確認できます)
a. COMP を有効にする前に、COMP パブリッシャを何らかのダミー ID で構成します。
DL_COMP_setPublisherChanID(COMP_0_INST, 0); // 実際のパブリッシャを削除します
b. DL_COMP_enableEvent(COMP_0_INST, (DL_COMP_EVENT_OUTPUT_EDGE)); // IMASK で COMP イベントを有効にします
c. DL_COMP_enable(COMP_0_INST); // COMP モジュールを有効にします。この手順で RIS のイベントがクリアされます。
d. DL_COMP_disableEvent(COMP_0_INST, (DL_COMP_EVENT_OUTPUT_EDGE)); // IMASK におけるクリアで COMP イベントが無効になります。
DL_COMP_setPublisherChanID(COMP_0_INST, COMP_0_INST_PUB_CH) // 実際のパブリッシャ f を構成。DL_COMP_enableEvent(COMP_0_INST, (DL_COMP_EVENT_OUTPUT_EDGE)); // IMASK で COMP イベントを再度有効化

または
コンパレータを有効後、割り込みを読み取り、コンパレータが有効になったために最初の割り込みが発生したことを検出します。

CPU_ERR_01

CPU モジュール

カテゴリ

機能

機能

メイン フラッシュと他のフラッシュ領域を切り替えると、CPU キャッシュの内容が破損する可能性がある

説明

メイン フラッシュと、NONMAIN や Factory 領域など他の不揮発性メモリ領域との間でアクセスを切り替える際に、キャッシュの破損が発生する可能性があります。

回避方法

メイン メモリ以外の領域に安全にアクセスするには、次の手順に従います:
1. CPUSS.CTL.ICACHE = 0x0 に設定して、キャッシュを無効にします。
2. SHUTDOWN Memory SYSCTL.SOCLOCK.SHUTDNSTORE0 から読み取ります。
3. NONMAIN または Factory Region メモリへの必要なアクセスを実行します。
4. CPUSS.CTL.ICACHE = 0x1 に設定して、キャッシュを再び有効にします。

CPU_ERR_02	CPU モジュール
カテゴリ	機能
機能	CPUSS のプリフェッチ / キャッシュ無効化に関する制限
説明	保留中のフラッシュ メモリへのアクセスがある場合、CPU プリフェッチ / キャッシュの無効化は反映されません。
回避方法	キャッシュとプリフェッチャを無効にして (順序は問いません)、SYSTCL のシャットダウン メモリ (SHUTDNSTORE) へのメモリ アクセスを発行します (詳細については、デバイス TRM の SHUTDNSTORE レジスタを確認してください)。 メモリ アクセスが完了すると、プリフェッチャ / キャッシュは無効になります。 例: <pre>CPUSS->CTL = (CPUSS_CTL_PREFETCH_DISABLE CPUSS_CTL_ICACHE_DISABLE); // 命令キャッシュとプリフェッチを無効化します DL_SYSCTL_setShutdownStorageByte(DL_SYSCTL_SHUTDOWN_STORAGE_BYTE_X , data) // SHUTDOWN メモリにバイトを保存します</pre>
CPU_ERR_03	CPU モジュール
カテゴリ	機能
機能	低電力モードへの遷移時に、プリフェッチャが誤った命令を読み取る可能性がある
説明	低電力モードへ遷移する際に保留中のプリフェッチがある場合、プリフェッチャが誤って正しくないデータ (すべて 0) をフェッチする可能性があります。デバイスがウェイクアップした際、もしプリフェッチャおよびキャッシュが ISR コードによって上書きされない場合、フラッシュから実行されるメイン コードが破損する可能性があります。たとえば、ISR が SRAM 内にある場合、フラッシュからプリフェッチされた誤ったデータは上書きされません。ISR から復帰する際に、プリフェッチャ内の破損したデータが CPU によってフェッチされ、誤った命令が実行されるおそれがあります。ハードウェア イベント ウェイクアップは、デバイスをウェイクアップするがプリフェッチャをフラッシュしないプロセスのもう 1 つの例です。
回避方法	低電力モードに入る前にプリフェッチャを無効にします。 例: <pre>CPUSS->CTL &= 0x6; // プリフェッチャを無効化、その他の設定は維持 SYSCTL.SOCLOCK.SHUTDNSTORE0 // SHUTDOWN メモリから読み出し __WFI(); // または __WFE(); この関数は低電力モードへの遷移を呼び出す CPUSS->CTL = 0x1; // プリフェッチャを有効化</pre>
CPU_ERR_04	CPU モジュール
カテゴリ	機能

CPU_ERR_04 (続き) CPU モジュール

機能

キャッシュが、ハード フォルトの原因となった FLASH 内のアドレスから正しいデータを返しません

説明

デバイスが FLASH アドレスからハード フォルトを引き起こした後、デバイスがハードフォルトから復旧し、(その FLASH アドレスのデータを格納している) キャッシュから情報を取得しようとする、戻り値がゼロになります。
さらに、同じ FLASH アドレスにアクセスすると、新しいハード フォルトはトリガされません。

回避方法

CPUSS.CTL.ICACHE ビットを変更して、キャッシュを無効化してから再度有効化します。

例:

CPUSS->CTL &= ~(CPUSS_CTL_ICACHE_ENABLE); // 命令キャッシュを無効化します

CPUSS->CTL |= CPUSS_CTL_ICACHE_ENABLE; // 命令キャッシュを有効化します

CRC/ CRCP_ERR_01

CRC/CRCP モジュール

カテゴリ

機能

機能

LPM が中断している状態では、DMA をトリガして CRC/CRCP モジュールにアクセスすることはできません

説明

RUN0/1/2 および SLEEP0/1/2 モードでは、DMA は通常 CRC/CRCP モジュールにアクセスできます。STOP/STANDBY モードでは、DMA がトリガされると、デバイスは中断状態の低消費電力モードに移行します。ただし、一部のデバイスでは、このモードになると、DMA は CRC/CRCP モジュールにアクセスできなくなります。

回避方法

1.CRC/CRCP への DMA アクセスが必要な場合は、STOP/STANDBY モードにしないでください。2.STOP/STANDBY モードでは、別のウェイクアップ機能 (割り込みなど) を使用して、デバイスを STOP/STANDBY モードから復帰させ、CRC/CRCP への DMA アクセスをトリガするか、CRC/CRCP へのアクセスに DMA ではなく CPU を使用します。

FCC_ERR_01

FCC モジュール

カテゴリ

機能

機能

BUSCLK が LFCLK から供給されている場合、FCC が誤って動作する

説明

BUSCLK が LFCLK から供給される場合、FCC は期待どおりに動作しません。FCC 完了信号がアサートされないか、または誤った FCC 値が報告されます。

回避方法

回避方法はありません。

FLASH_ERR_02 *FLASH* モジュール

カテゴリ

機能

機能

NONMAIN でのデバッグの無効は、パスワードを使用して再度有効にできる

説明

デバッグが NONMAIN 設定 (DEBUGACCESS = 0x5566) により無効化されている場合でも、プログラムされたパスワードを使用してデバイスにアクセスできる可能性があります (パスワードが明示的に設定されていない場合はデフォルト値が使用される)。

回避方法

回避方法 1:

DEBUGACCESS を Debug Enabled with Password オプション (DEBUGACCESS = 0xCCDD) に設定し、PWDDEBUGLOCK フィールドに一意のパスワードを入力します。より高度のセキュリティを確保するために、暗号化されたランダムなデバイス固有のパスワードを使用することをお勧めします。これにより、適切な 128 ビットのパスワードでデバッグアクセスが可能になりますが、一部のデバッグコマンドで、CFG-AP と SEC-AP にアクセスすることもできます。

回避方法 2:

SWDP_MODE を無効にして、物理的な SW デバッグポートを完全に無効にします。これにより、デバイスへのデバッグアクセスや要求は完全に防止されますが、Failure Analysis やリターンフローに影響が出るおそれがあります。

FLASH_ERR_03 *FLASH* モジュール

カテゴリ

機能

機能

2 待機状態のフラッシュ アクセスの直後に無効なブート コード領域へのアクセスが行われると、次のフラッシュ アクセスでも違反が発生する可能性があります

説明

2 待機状態が設定されている状態で、フラッシュ アクセスの直後に BOOTCODE 領域へのアクセスを行うと、その次のフラッシュ アクセスでも違反が発生する可能性があります。

回避方法

ブート フェーズ終了後は、ブートコード領域へのアクセスを行わないでください。そうしない場合、ブートコード違反の後に正しいフラッシュ アクセスを行うまでに、少なくとも 4 クロック サイクルの間隔を空ける必要があります。

FLASH_ERR_04 *FLASH* モジュール

カテゴリ

機能

機能

エラーがメイン フラッシュ領域以外で発生した場合、SYSCTL_DEDERRADDR には誤ったアドレスが報告されます。

説明

FLASH_ERR_04

(続き)

FLASH モジュール

回避方法

SysCtl_DEDERRADDR の戻りアドレスで 0x00Cxxxx が返る場合は、0x41000000 で OR 演算を実行して、NONMAIN または工場出荷時領域の復帰アドレスに適切なアドレスを取得します。たとえば、SYSCTL_DEDERRADDR = 0x00C4013C の場合、実際のアドレスは 0x41C4013C となります。

SYSCTL_DEDERRADDR の復帰アドレスが 0x00Dxxxx を返した場合、Databank の正しい復帰アドレスを得るために、0x41000000 との OR 演算を行います。たとえば、SYSCTL_DEDERRADDR = 0x00D0012A の場合、実際のアドレスは 0x00D0012A となります。

FLASH_ERR_05

FLASH モジュール

カテゴリ

機能

機能

DEDERRADDR に誤ったリセット値が設定される可能性があります

説明

SYSCTL -> DEDERRADDR のリセット値では、正しい 0x00000000 のかわりに 0x00C4013C が返されることがあります。エラーが発生している場所はファクトリトリム領域であり、故障を示すものではありません。そのため、この値は無視して問題ありません。デバイスに NONMAIN をプログラムされると、リセット値が変化する傾向があります。

回避方法

0x00C4013C を別のリセット値として受け入れ、ブートからのデフォルト値を 0x00000000 または 0x00C4013C にすることができます。戻り値はデバイス上の MAIN フラッシュの範囲外であるため、実際のフラッシュ DED ステータスから返された可能性はありません。

FLASH_ERR_06

フラッシュ モジュール

カテゴリ

機能

機能

CPU と DMA は、同時にフラッシュにアクセスすることはできません

詳細

CPU と DMA はフラッシュに同時にアクセスすることができません。これらが同時にアクセスすると、フラッシュから誤ったデータが読み出される可能性があります。

回避方法

CPU と DMA 経由で同時にフラッシュにアクセスすることはできません。通常のフラッシュ操作 (プログラム/ 消去/ 読み取り検証/ ブランク検証動作など) や DMA によるフラッシュからの読み出しを行う場合、ソフトウェアは、フラッシュがビジー状態の間に CPU がフラッシュにアクセスしないようにする必要があります。これを回避する方法としては、フラッシュ操作の実行中にコードを SRAM 上に配置するか、DMA が読み出す必要のあるデータをフラッシュ メモリから SRAM に移しておく方法があります。

FLASH_ERR_08 *FLASH モジュール*

カテゴリ

機能

機能

通常の無効なメモリ領域に対してハード フォルトは生成されません

説明

不正なメモリ アドレス空間へのアクセス中は、以下に示すようにハード フォルトは生成されません。1. 0x010053FF ~ 0x20000000 2. 0x40BFFFFFF ~ 0x41C00000 3. 0x41C007FF ~ 0x41C40000

回避方法

番号

GPIO_ERR_01 *GPIO モジュール*

カテゴリ

機能

機能

STANDBY モードでは、GPIO のウェイクアップ エッジが失われる可能性があります

説明

一度 GPIO のエッジでウェイクアップした後、STANDBY/STOP/SLEEP モードでは、その後の GPIO ウェイクアップ エッジが見逃される可能性があります。

ケース 1:

STANDBY0 ウェイクアップ: MCU が STANDBY/STOP/SLEEP モードに入り、ある IO が「ウェイク」状態にあるときに、IO を「非ウェイク」状態に 3 LFCLK サイクル未満の間だけ戻し、再度アサートした場合、次のウェイクアップ エッジは検出されない可能性があります。

ケース 2:

STANDBY1 ウェイクアップ: GPIO エッジを使ってウェイクアップした場合に、デバイスが STANDBY1 に戻る時点で GPIO パルスがまだアクティブであると、その後のウェイクアップ エッジは検出されません。

回避方法

ケース 1:

デバイスがアクティブ モードの間に GPIO をディアサートしておく

または GPIO のウェイクアップ パルスを 3 LFCLK サイクルより長くなるようにします

ケース 2:

GPIO ウェイクアップ エッジを立ち下がりエッジと立ち上がりエッジの両方に設定する、または、STANDBY1 に入力する前に GPIO ウェイクアップ パルスがアクティブでないことを確認します

GPIO_ERR_02 *GPIO モジュール*

カテゴリ

機能

機能

PA2 ピンに負の電流が注入された後の大電流

GPIO_ERR_02 (続き)

GPIO モジュール

説明

PA2 ピンに負の電流が注入されると、デバイスに予期しない持続的な大電流消費が発生する可能性があります。

この正誤表は、PA2 が ROSC モードで使用されている場合 (ROSC と VSS の間に 100kΩ の抵抗が実装されている場合) は有効ではありません。

回避方法

PA2 ピンの近くに抵抗を配置し、PA2 ピンと接続されている回路との間に、標準値 100Ω の直列抵抗を配置します。この抵抗は、PA2 ピンで観測される高速過渡を制限することを目的としており、この状況が発生するのを防止するのに十分です。

OR

PA2 ピンの使用は避け、代わりに代替ピンを使用してください。

GPIO_ERR_03

GPIO モジュール

カテゴリ

機能

機能

デバッガで GPIO EVENT0 IIDX を読み取ると、割り込みがクリアされます。

説明

GPIO の EVENT0 の IIDX をデバッガで読み取ると、CPU による読み取りと見なされ、割り込みがクリアされます。

回避方法

デバッグ中、event0 の IIDX は、ソフトウェアで RIS を読み取ることで確認できます。

GPIO_ERR_04

GPIO モジュール

カテゴリ

機能

機能

グローバルの高速ウェイクアップを設定すると、GPIO ピンから DIN レジスタへのデータ転送が行われなくなる

説明

CTL レジスタの「高速ウェイクのみ」(fastwake-only) ビットを設定し、実行モード中に GPIO ピンにデータを強制的に出力した場合、デバイスはウェイクアップしますが、GPIO ピン上のデータは DIN レジスタに反映されません。これは、CTL レジスタ構成により GPIO ピンから DIN レジスタへのデータフローがブロックされるためです。

回避方法

GPIO ピンが DIN レジスタに入ることを想定している場合、GPIO の「高速ウェイクのみ」機能は使用しないでください。

GPIO_ERR_06**GPIO モジュール****カテゴリ**

機能

機能

DMA 転送が進行中の間に、GPIO DOUT、DOUTSET、DOUTCLR レジスタへの書き込みが失われる可能性があります

説明

GPIO DOUT、DOUTSET、DOUTCLR レジスタには DMA からアクセスできません。実装の誤りにより、同時 DMA 転送が進行中の間に、GPIO DOUT、DOUTSET、DOUTCLR への CPU アクセスもブロックされます。

回避方法

アプリケーションコードでは、DOUT、DOUTSET、DOUTCLR レジスタに書き込む代わりに、ソフトウェアは DOUTTGL レジスタに対して等価な書き込みを実行する必要があります (DOUTTGL レジスタへの CPU 書き込みの制限については、「回避方法 GPIO_ERR_05」を参照)。

以下の疑似コードでは、「ピン」は、構成する GPIO モジュールのピンのビットベクトルを示しています。

```
DL_GPIO_setPins(GPIO_Regs* gpio, uint32_t pins)
{
    gpio->DOUTTGL31_0 = ~(gpio->DOUT31_0) & pins;
}
```

```
DL_GPIO_clearPins(GPIO_Regs* gpio, uint32_t pins)
{
    gpio->DOUTTGL31_0 = gpio->DOUT31_0 & pins;
}
```

```
DL_GPIO_writePins(GPIO_Regs* gpio, uint32_t pins)
{
    gpio->DOUTTGL31_0 = ~(gpio->DOUT31_0) & pins;
    gpio->DOUTTGL31_0 = gpio->DOUT31_0 & (~pins);
}
```

```
DL_GPIO_writePinsVal(GPIO_Regs* gpio, uint32_t pinsMask, uint32_t pinsVal)
{
    uint32_t doutVal = gpio->DOUT31_0;
    doutVal &= ~pinsMask;
    doutVal |= (pinsVal & pinsMask);
    gpio->DOUTTGL31_0 = ~(gpio->DOUT31_0) & doutVal;
    gpio->DOUTTGL31_0 = gpio->DOUT31_0 & (~doutVal);
}
```

I2C_ERR_01**I2C モジュール****カテゴリ**

機能

I2C_ERR_01 (続き) I2C モジュール

機能

SMBUS クイック コマンドが発行されると、I2C モジュールが SBMUS モードで SDA ラインをホールドし続ける場合があります

説明

I2C モジュールがターゲット モードで SBMUS に設定されている場合、バス コントローラがデバイス宛に SMBUS クイック コマンド (I2C START コンディション → 7 ビット アドレス → 1 ビットの R/W ビット → 1 ビットの ACK → I2C STOP 条件) を発行すると、I2C モジュールが SDA ラインを Low に引き下げようとする場合があります。これがバス コントローラによる I2C STOP 条件の生成と同時に起こると、STOP 条件が正しく完了せず、通信が妨げられる可能性があります。

回避方法

I2C モジュールが SDA ラインを Low に駆動するのを防ぐために、アドレス ACK が完了する前に、MSB を 1 に設定したデータを I2C モジュールの送信 FIFO にロードします。これにより、バス コントローラは STOP 条件を正常に発行し、SMBUS クイック コマンドを完了できます。

I2C_ERR_02 I2C モジュール

カテゴリ

機能

機能

I2C クイック コマンド読み取りモードは、特定の条件でのみ機能します

説明

読み取りモードでの I2C のクイック コマンドは、TXFIFO に MSB が 1 に設定されたデータがあり、かつクロック ストレッチが無効の場合のみ動作します。

回避方法

MSB が 1 に設定されたダミー データを TXFIFO に入れ、クロック ストレッチ (CLKSTRETCH = 0) を無効にします。

I2C_ERR_03 I2C モジュール

カテゴリ

機能

機能

ソースに MFCLK を使用している場合、I2C ペリフェラル モードを起動できません

説明

I2C モジュールがペリフェラル モードに設定されており、かつ I2C が MFCLK (中周波クロック) をソースとして使用していて、さらにデバイスが STOP2 または STANDBY0/STANDBY1 の電力モードにある場合、データを受信しても I2C はデバイスをウェイクアップできません。

回避方法

I2C をペリフェラル モードで使用し、データ受信時に低電力モードからのウェイクアップを必要とする場合は、I2C のクロック ソースを MFCLK ではなく BUSCLK に設定します。

I2C_ERR_04

I2C モジュール

カテゴリ

機能

機能

SCL が Low で SDA が High の状態では、ターゲット I2C はストレッチを解除できません。

概要

- 1: SCL ラインを接地して解放し、デバイスは無制限に SCL を Low にプルします。
- 2: ポストクロックストレッチ、タイムアウト、解放。ライン上に別のクロック Low がある場合、本デバイスは無期限に SCL を Low にプルします。

回避方法

I2C ターゲットアプリケーションで、非同期高速クロック要求を使用した低電力モードでのデータ受信が不要な場合は、**SWUEN** をデフォルトで無効にすることを推奨します (リセット時や電源サイクル時を含む)。この場合、バグの説明 1 と 2 は発生しません。

I2C ターゲットアプリケーションで、非同期高速クロック要求を使用した低電力モードでのデータ受信が必要な場合は、低電力モードへ移行する直前に **SWUEN** を有効にし、復帰後に **SWUEN** をクリアします。このシナリオでも、I2C ターゲットが低消費電力のときにバグ説明 1 および 2 が発生するおそれがあります。バス上の他のデバイスによって連続的なクロックストレッチングまたはタイムアウトが発生すると、SCL ラインが無期限にストレッチされます。この状況から回復するには、I2C ターゲットデバイスで Low タイムアウト割り込みを有効にし、低タイムアウト ISR 内で I2C モジュールをリセットして再初期化します。

I2C_ERR_05

I2C モジュール

カテゴリ

機能

機能

進行中のトランザクション中に **ACTIVE** ビットをトグルすると、I2C SDA が 0 に固定化されるおそれがあります。

概要

進行中の転送中に **ACTIVE** ビットがトグルされると、ステート マシンはリセットされます。ただし、I2C コントローラによって駆動される SDA と SCL 出力はリセットされません。SDA が 0 の状態で I2C コントローラが IDLE 状態に遷移すると、I2C コントローラは IDLE 状態から先へ進めず、SDA の値も更新できなくなります。I2C ターゲットの **BUSBUSY** が設定され (**ACTIVE** ビットのトグルによってライン上で開始が検出されます)、**BUSBUSY** はクリアされません。これは、I2C コントローラが **STOP** 条件を生成せず、停止を駆動してクリアできないためです。

回避方法

進行中のトランザクション中は、**ACTIVE** ビットをトグルしないでください。

I2C_ERR_06

I2C モジュール

カテゴリ

機能

機能

SMBus の High タイムアウト機能は、I2C クロックが 24 kHz 未満になると動作しません

I2C_ERR_06 (続き) I2C モジュール

説明

SMBus の High タイムアウト機能は、I2C クロックレートが 24 kHz 未満 (20 kHz、10 kHz など) では正常に動作しません。SMBus 仕様から、アクティブトランザクション中の SCL High 時間の上限は 50µs です。開始 MMR ビットの書き込みから SCL Low までに要する合計時間は 60µs で、50µs 以上です。タイムアウトイベントのトリガがかかりされ、転送開始時にトランザクションを完了することなく I2C コントローラが IDLE に移行できます。以下は詳細な説明です。SCL が 20 kHz に構成されている場合、SCL の Low 期間と High 期間はそれぞれ 30 µs および 20 µs です。まず、High タイムアウトカウンタでデクリメントが開始し、同時に MMR ビットの書き込みが開始します。その後、START MMR ビットの書き込みから SDA が Low (スタート条件) になるまでに、1 SCL Low 期間 (30 µs) かかります。次に、SDA が Low (スタート条件) になってから SCL が Low になる (データ転送が開始) までにさらに別の SCL Low 期間 (30µs) がかかり、この時点で High タイムアウトカウンタが停止します。合計で、カウンターの開始から終了まで 60µs かかります。ただし、高タイムアウトカウンタには上限 (50µs) により、I2C トランザクションは問題なく正常に動作しますが、タイムアウトイベントがトリガされます。

回避方法

I2C クロックが 24KHz 未満の場合は、SMBus 高タイムアウト機能を使用しないでください。

I2C_ERR_07 I2C モジュール

カテゴリ

機能

機能

コントローラの制御レジスタへの連続書き込みを行うと、I2C 通信が開始されない可能性があります。

説明

CTR レジスタへの連続書き込みを行うと、次の CTR.START によって正しく開始条件が発生しない可能性があります。

回避方法

CTR.START を含むすべての CTR ビットは、1 回の書き込みでまとめて設定するか、CTR ビットの書き込み後に十分な待機時間を挟んでから CTR.START を書き込む必要があります。

I2C_ERR_08 I2C モジュール

カテゴリ

機能

機能

RXDONE 割り込みの直後に FIFO を読み出すと、誤ったデータが取得されます。

説明

RXDONE 割り込みが発生したとき、FIFO は最新のデータに対して更新されない場合があります。

回避方法

最新のデータが FIFO に確実に反映されるように、2 つの I2C クロックサイクル分待機してください。I2C CLK は、I2C レジスタの CLKSEL レジスタに基づいています。

I2C_ERR_09	I2C モジュール
カテゴリ	機能
機能	I2C を低速で動作させている場合、割り込みサービスルーチン (ISR) 内での読み取り時に、開始アドレス一致ステータスがタイミング的に更新されていない可能性があります。
説明	I2C 速度が 100kHz 未満で動作している場合、ADDRMATCH ビット (TSR レジスタのアドレス一致) が割り込みによる読み取りに間に合うように設定されない可能性があります。
回避方法	I2C で 100kHz 未満で実行している場合は、ADDRMATCH ビットを読み取る前に少なくとも 1 つの I2C CLK サイクルを待機します。
I2C_ERR_10	I2C モジュール
カテゴリ	機能
機能	低消費電力に移行しないよう、I2C ビジー ステータスは有効になっています。
説明	I2C ターゲットモードでは、STOP ビットがない場合、トランザクションの後、I2C ビジーステータスは High のままです。
回避方法	STOP ビットを送信するように I2C コントローラをプログラムします。最後のバイトに対して NACK を送信しないでください。任意の I2C 転送を必ず STOP 条件で終了し、適切な BUSY ステータスと非同期クロック要求の動作にしてください (低消費電力モードへの再移行に備えるため)。
I2C_ERR_13	I2C モジュール
カテゴリ	機能
機能	I2C BUSY ビットをポーリングしても、コントローラの転送が完了したことが保証されない場合があります。
説明	I2C コントローラ転送を開始するために CCTR.BURSTRUN ビットを設定した後、BUSY ステータスがアサートされるまでに約 3 回の I2C 機能クロックサイクルかかります。CCTR.BURSTRUN を設定した後すぐに転送完了を待つために BUSY ビットのポーリングを使用すると、BUSY ステータスが設定される前にチェックされる可能性があります。この問題は、CLKDIV 値が高い場合 (I2C 機能クロックが遅くなる)、またはコンパイラの最適化レベルが高い場合に発生する可能性が高くなります。
回避方法	BUSY ステータスをポーリングする前にソフトウェア遅延を追加してください。ソフトウェア遅延 = $3 \times \text{CPU CLK} / \text{I2C 機能クロック} = 3 \times \text{CPU CLK} / (\text{CLKSEL} / \text{CLKDIV})$ 。例えば、クロック分周器 (CLKDIV) が 8、クロックソース (MFCLK) が 4 MHz、CPU CLK が 32 MHz の場合、ソフトウェア遅延 = $3 \times 32 \text{ MHz} / (4 \text{ MHz} / 8) = 192 \text{ CPU サイクル}$

I2C_ERR_15	I2C モジュール
カテゴリ	機能
機能	I2C SDA のグリッチにより、低消費電力モードで I2C ペリフェラルがアクティブ状態になる
説明	I2C SDA 上の 4 μ s 以内のグリッチにより、低消費電力モード (STOP/STANDBY) で I2C ペリフェラルがアクティブ ステータスに切り替わり、低消費電力モードに戻らなくなります。
回避方法	1.I2C バスの容量を増やします。ただし、合計が I2C 規格で許容される値を超えないようにしてください。2.I2C バスの電圧を上昇させます。3.I2C ラインをノイズ発生源から離します。4.I2C ペリフェラルのステータスを確認するために、デバイスを定期的にウェークアップし、I2C ペリフェラルのステータスが IDLE ステータスでない場合は、I2C をリセットして再度初期化します。
PMCU_ERR_01	GPIO モジュール
カテゴリ	機能
機能	PA2/ROSC ピンの電流リーク
説明	ROSC ではない機能構成でピン PA2/ROSC を使用する場合、ピンが high に駆動されていると、大きなリーク電流が観測されます。これは、約 17k Ω のインピーダンスを経由するグラウンドに意図しない接続が原因です。 このエラッタは、ROSC 機能には影響しません。
回避方法	ROSC の代替機能として PA2/ROSC を使用する場合、ピンが外部または内部のいずれかで high に駆動されているとき、エネルギー バジェットにおける追加のリーク電流を考慮する必要があります。 抵抗を使用してピンを外部でプルアップすると、このエラッタにより分圧回路が形成されるため、外部電圧は想定よりも低くなります。
PMCU_ERR_02	GPIO モジュール
カテゴリ	機能
機能	デバイスのスタートアップ時の過渡電圧出力
説明	ROSC ではない機能構成でピン PA2/ROSC を使用する場合、起動時に PA2/ROSC ピンに意図しない過渡電圧パルスが観測されることがあります。 このエラッタは、ROSC 機能には影響しません。

PMCU_ERR_02 (続 き)

GPIO モジュール

回避方法

回避策はありません。

PMCU_ERR_03

BOR モジュール

カテゴリ

機能

機能

BOR1、BOR2、BOR3 は、スタンバイ モードでは動作しません

説明

代替 BOR スレッショルド (BOR1、BOR2、BOR3) は、デバイスがスタンバイ モードで動作している場合、ユーザーが選択可能では機能しないため、これらのスレッショルドを超えるとデバイスは正しくリセットされません。

回避方法

スタンバイ モードで動作する場合は、BOR1、BOR2、BOR3 を使用しないでください。スタンバイ モードに移行する前に、デフォルトの BOR0 レベルを使用するようデバイスを構成します。スタンバイ モードを終了すると、代替 BOR レベルを再度イネーブルにすることができます。

PMCU_ERR_10

PMCU モジュール

カテゴリ

機能

機能

特定の動作条件下では、VBOOST により大きな遅延が発生する可能性があります

説明

アナログマルチプレクサの VBOOST は、VDD < 1.8V で大きな遅延が発生しました。このため、HFXT、COMP、SYSOSC (FCL-EXTERNAL R)、OPA、GPAMP などのその他のモジュールのセトリグタイムが遅延します。

回避方法

VDD を 1.8V 以上に維持し、GENCLKCFG[23:22] = 0x2 を設定して、VBOOST を ONALWAYS モードで使用します。

PMCU_ERR_13

PMCU モジュール

カテゴリ

機能

機能

STOP2 または STANDBY0 からのウェークアップ時に MCU がスタックする可能性があります

説明

デバイスが STOP2 または STANDBY に移行するときにプリフェッチ アクセスが保留されている場合、デバイスがウェークアップしたときに、保留中のプリフェッチにより、デバイスが通常実行を再開できない可能性があります。エラッタは、WFI 命令がワードアライメントされておらず、フラッシュの待機ウェイト状態が 2 の場合に発生します。このような場合、DMA 転送も保留中の割り込みも処理されません。

PMCU_ERR_13 (続 き)

PMCU モジュール

回避方法

ユーザーはプリフェッチを無効化し、シャットダウン ストア メモリ読み取りを発行する必要があります。これにより、新しいプリフェッチが発行されなくなり、保留中のプリフェッチを完了させることができます。

PWREN_ERR_01

GPIO モジュール

カテゴリ

機能

機能

PWREN レジスタを無効にした後でも、ペリフェラル レジスタには引き続きアクセス可能な状態となります

説明

PWREN レジスタを 0 に設定してペリフェラルの電源を無効にした場合でも、ペリフェラルのレジスタは読み出すとデータ値を保持しているように見ることがあります。PWREN が 0 の場合にレジスタを読み書きしても、ペリフェラルは動作していないため、影響はありません。

影響を受けるペリフェラル: コンパレータ (COMP)、オペアンプ (OPA)、タイマ A、タイマ G、汎用入出力 (GPIO)、ウィンドウ付きウォッチドッグ タイマ (WWDG)。

回避方法

ペリフェラルの PWREN レジスタが 0 に設定されている場合、ペリフェラルに関連するレジスタの値は無視するか、無効なものとして扱う必要があります。

RST_ERR_01

RST モジュール

カテゴリ

機能

機能

LFXT または LFCLK_IN が失われた後に、NRST 解除エッジが LFOSCGOOD の立ち上がりエッジと一致すると、デバイスはリセット状態のままとなります

説明

特定のコーナー ケースで、NRST 信号の解除エッジ (Low から High への遷移) が LFOSCGOOD の立ち上がりエッジと一致した場合、デバイスは NRST のデアサートを検出できず、無制限にリセット状態にとどまります。問題が発生する可能性のある条件: LFCLK ソースが LFCLK_IN または LFXT から LFOSC に切り替わると、LFOSC は再度有効になり、再有効イベントから 250µs (最小) から 1.5ms (最大) 以内に LFOSCGOOD のステータスが High にアサートされます。NRST Low パルスが、その解除エッジ (Low から High への遷移) が 50ns のウィンドウ内で LFOSCGOOD の立ち上がりエッジと一致するように印加された場合、NRST デアサートは検出されず、デバイスはリセット状態のままとなります。

回避方法

この問題を回避するため、NRST Low パルス幅を 2ms より長く維持してください。

SPI_ERR_01

SPI モジュール

カテゴリ

機能

SPI_ERR_01 (続き) SPI モジュール

機能

SPI パリティ ビットは機能しません

説明

SPI ハードウェア パリティ モードは機能しません。

回避方法

SPI モジュールの CTL1 レジスタ内にある PTEN ビットまたは PREN ビットによってハードウェア パリティを有効にしないでください。パリティの計算とチェックは、アプリケーション ソフトウェアを使用して実装できます。

SPI_ERR_03

SPI モジュール

カテゴリ

機能

機能

ペリフェラルとして構成した場合、CSCLR を有効にすると、受信データは SPH = 0 モードで 1 ビット右シフトされる

説明

ペリフェラル モードで CSCLR を有効にした場合、CS 信号がアクティブまたは非アクティブのときに SCK ラインにグリッチが発生すると、次の最初のフレームで受信データが 1 ビット右方向にシフトします。この問題は、SPH = 0 の Motorola SPI フレーム フォーマットで発生し、CS が非アクティブの時に SCK がトグルするマルチ ペリフェラル モードに影響します。

回避方法

1. CSCLR = 0h に設定します。
2. SPH = 0 モードで CSCLR = 1h に設定すると、常に最初のフレームをドロップします。

SPI_ERR_04

SPI モジュール

カテゴリ

機能

機能

SPI ペリフェラルが受信モードのみの場合、各フレーム受信後の IDLE/BUSY ステータストグル。

概要

SPI ペリフェラルが受信モードのみの場合、SPI がデータを連続的に受信している間に、各フレーム受信の後で、IDLE 割り込みおよび BUSY ステータスがトグルされます (SPI_PHASE = 1)。ここでは、ペリフェラルの TXFIFO にロードされるデータはなく、TXFIFO は空です。

回避方法

SPI ペリフェラルのみの受信モードを使用しないでください。SPI ペリフェラルを送受信モードに設定します。TX FIFO のデータを SPI 用に設定する必要はありません。

SPI_ERR_05

SPI モジュール

カテゴリ

機能

SPI_ERR_05 (続き) SPI モジュール

機能

SPI ペリフェラルの受信タイムアウト割り込みは、RXFIFO のデータの有無にかかわらず発生します

概要

SPI タイムアウト割り込みを使用すると、最終的な SPI CLK を受信した後も RXTIMEOUT でデクリメントが継続するため、誤った RXTIMEOUT が発生するおそれがあります。

回避方法

最後のパケットを受信した後は、RXTIMEOUT を無効にします (これは ISR 内で実行可能です)。その後、SPI 通信が再開されるときに、RXTIMEOUT を再度有効にしてください。

SPI_ERR_06

SPI モジュール

カテゴリ

機能

機能

デバッグ HALT がアサートされている場合、IDLE/BUSY ステータスは SPI IP の正しい状態を反映しません

概要

IDLE/BUSY は HALT とは無関係で、RXFIFO/TXFIFO の書き込み/読み取りストロブのみをゲーティングします。つまり、コントローラがデータ送信中であっても、そのデータが FIFO にラッチされていない状態で BUSY ステータスが設定されてしまいます。POCI 回線は、停止中に以前に送信されたデータを回線上で送信します

回避方法

SPI IP が停止しているときは、IDLE/BUSY ステータスを使用しないでください。

SPI_ERR_07

SPI モジュール

カテゴリ

機能

機能

SPI ペリフェラルで TXFIFO への読み取り / 書き込みが同時に発生した場合、SPI アンダーフロー イベントは生成しない場合があります。

説明

SPI.CTL0.SPH = 0 であり、本デバイスが SPI ペリフェラルとして構成されている場合。

SPI コントローラからの読み取り要求がある間に TXFIFO への書き込みが発生した場合、読み取り / 書き込み要求が同時に発生するため、アンダー フロー イベントが生成されない可能性があります。

回避方法

SPI コントローラによるデバイスのアドレス指定中、TXFIFO が確実に空でないようにします。これは、同じ TXFIFO アドレスへの書き込みと読み取りを避けるために、データを事前ロードすることで実現できます。あるいは、CRC のようなデータチェック戦略を使用してパケットが確実に正しく送信されるようにし、CRC が一致しない場合にデータを再送信することもできます。

SPI_ERR_08	SPI モジュール
カテゴリ	機能
機能	DMA がイネーブルのとき、SPI_DMA_DONE_RX 割り込みは RX 割り込みに影響を及ぼす
説明	SPI_DMA_DONE_RX 割り込みフラグがアサートされも割り込みがイネーブルでない場合、RX 割り込みは次のタイミングでトリガされません。
回避方法	DMA_DONE_RX 割り込みを有効にします。または、SPI_DMA_DONE_RX 割り込みフラグを手動でクリアします。
SPI_ERR_09	SPI TX FIFO 割り込みは、モジュールで 1 回だけトリガできます
カテゴリ	機能
機能	
説明	SPI Tx FIFO のトリガは 1 回のみ設定されます。その後、Tx FIFO がスレッシュホールドを超えずに、割り込みサービス ルーチンから復帰した場合、FIFO がスレッシュホールド レベルを下回っていても RIS は再トリガされません。
回避方法	「TX FIFO スレッシュホールドレベル」を「TX FIFO は空」に構成してください。
SPI_ERR_10	SPI モジュール
カテゴリ	機能
機能	DMA は、最新の SPI トリガ カウントをサンプリングしていません
説明	受信タイムアウト (RTOUT) イベントで DMA 転送をトリガするように SPI が構成されている場合、トリガ要求の時点で DMA チャネルが別の転送の処理をビジー状態にしている場合、DMA が要求をアクノリッジする前に SPI が受信した追加のバイトは転送に含まれません。DMA は、トリガ要求が発行されたときに受信されたバイト数のみを転送し、その後受信したバイトは RX FIFO で未読のままになります。
回避方法	RX トリガ条件を RTOUT と組み合わせて使用するよう DMA_TRIG_RX を構成します。RX トリガは、RX FIFO が設定されたスレッシュホールド レベルに達すると起動し、DMA 処理が開始される前に、予測されるバイト数がすべて FIFO に存在することを保証します。これにより、DMA チャネルによる要求のアクノリッジが遅延するかどうかに関係なく、DMA 転送カウントが正確であることが保証されます。
SYSCCTL_ERR_03	SYSCCTL モジュール
カテゴリ	機能

SYSTL_ERR_03

(続き)

SYSTL モジュール

機能

DEDERRADDR は、**SYSRESET** または **SYSSTATUSCLR** への書き込みの後にも持続します

詳細

SYSRESET または **SYSSTATUSCLR** レジスタへの書き込みの後も、**DEDERRADDR** は持続します。この値は、新しい **FLASHDED** エラーが発生した場合にのみ上書きされます。この挙動は、初期リセット値をゼロに規定されているテクニカル リファレンス マニュアル (TRM) に矛盾します。

回避方法

回避方法はありません。

SYSTL_ERR_05 LFCLK モジュール

カテゴリ

機能

機能

シャットダウンからの終了時に **LFCLK** が動作しない

説明

LFCLK_IN ピンがプルアップ付きの汎用入力 (または) **LFCLK_IN** 機能として構成されている場合、この構成では、シャットダウン モードを終了すると、**LFCLK** がスタックします。

回避方法

次のいずれかを選択: 1. この **LFCLK_IN** I/O では、プルアップの代わりにプルダウンをイネーブルにする、2. 入力として構成するのを避ける

SYSTL_ERR_06 CLK_OUT モジュール

カテゴリ

機能

機能

非同期クロックが **CLK_OUT** ソースとして選択されているときに、ユーザーが **CLK_OUT** をディスエーブルにすると、外部クロック出力 (**CLK_OUT**) でグリッチが発生する可能性があります

説明

CLK_OUT のクロック ソースが現在のバス クロックと非同期である場合 (たとえば、バス クロックが **SYSOSC** で、**CLK_OUT** のクロック ソースが **LFCLK** に選択されている場合)、この場合、**CLK_OUT** ピンが一定時間イネーブルになってからディスエーブルになるときにグリッチが発生することがあります

回避方法

外部ピンへのグリッチ出力を回避するには、以下のシーケンスに従ってください: **CLK_OUT** を有効化するケース: **IOMUX** の有効化設定は、**CLK_OUT** の有効化設定の後に行う必要があります。 **CLK_OUT** を無効化するケース: **IOMUX** の無効化設定は、**CLK_OUT** の無効化設定より前に行う必要があります。

SYSTL_ERR_11 SYSTL モジュール

カテゴリ

機能

SYSCCTL_ERR_11

(続き)

SYSCCTL モジュール**機能**

FCL がイネーブルでデバイスが RUN2/SLEEP2 モードの場合、予期しない BOR リセット サイクルが発生する

説明

FCL がイネーブルでデバイスが RUN2/SLEEP2 モードで動作する場合、予期しない BOR が発生し、BOR1/2/3 に NMI が発生します

回避方法

BOR1/2/3 モニタを使用する場合は、FCL を有効にしてデバイスを RUN2 または SLEEP2 モードで動作させることは避けてください。

SYSOSC_ERR_01 SYSOSC モジュール**カテゴリ**

機能

機能

STOP1 モードと SYSOSC の FCL を併用すると、MFCLK にドリフトが発生する可能性があります

概要

MFCLK が有効で、SYSOSC が周波数補正ループ (FCL) モードを使用しており、STOP1 の低消費電力動作モードを使用している場合、SYSOSC が 4MHz から 32MHz に戻るとき (STOP1 から RUN モードへの終了時、または SYSOSC を 32MHz に強制的に強制的に印加する非同期高速クロック要求時のいずれか)、MFCLK が 2 サイクルドリフトすることがあります。

回避方法

STOP1 モードの代わりに STOP0 モードを使用してください。STOP0 モードでは MFCLK のドリフトは発生しません。

または

STOP1 を使用する場合は、FCL モードで SYSOSC を使用しないでください (FCL を無効のままにしておきます)。

SYSOSC_ERR_02 SYSOSC モジュール**カテゴリ**

機能

機能

SYSOSC が FCL モードで無効化されている LPM 中に非同期クロック要求を受信しても、MFCLK は動作しません

説明

以下のシナリオでは、MFCLK はトグルを開始しません:

- 1.FCL モードを有効にした後、MFCLK を有効にします
- 2.SYSOSC が無効になる低消費電力モードに移行します (SLEEP2/STOP2/STANDBY0/STANDBY1)。
- 3.MFCLK を機能クロックとして使用する一部のペリフェラルから非同期要求を受信されます。ASYNC 要求を受信すると、SYSOSC は有効になり、ulpclock は 32MHz になります。ただし、デ

SYSOSC_ERR_02

(続き)

SYSOSC モジュール

バイスが依然として LPM に設定されているため、MFCLK はゲートオフの状態となり、一切トグルしません。

回避方法

SYSOSC が FCL モードを使用している場合は、通常 SYSOSC がオフになる LPM モードへ移行する際に、ペリフェラル用の MFCLK を有効にしないでください。

SYSOSC_ERR_04 SYSOSC モジュール

カテゴリ

機能

機能

FCL ON モードで SYSOSC の精度が低下する

説明

内部発振器 SYSOSC の FCL ON モードを使用する場合、精度が最大 $\pm 3\%$ 低下する可能性があります。精度の低下は、4MHz FCL サンプリング クロックと、4MHz の高調波で動作する他のペリフェラルによって発生する、システム内のノイズとの同期に起因します。

回避方法

システムからノイズを完全に除去する回避方法はありません。ただし、FCL ON モードを使用する場合、ペリフェラルが 4MHz の倍数ではない周波数で動作しているときに、劣化を最小限に抑えることができます。

SYSOSC_ERR_05 SYSOSC モジュール

カテゴリ

機能

機能

FCL が有効な場合、LPM からの非同期高速ウェークアップ中、SYSOSC はベース周波数より低く動作する場合があります

説明

FCL = 有効な場合、低消費電力モード時およびアクティブな非同期高速クロック要求がある場合、SYSOSC はベース周波数より最大 10% 低い値で動作する可能性があります。これは、デバイスが低消費電力モードにある間に発生します。なぜなら、FCL = 有効であっても、FCL = 無効トリムが常に適用されてしまい、FCL = 有効トリムが使用されないためです。デバイスがウェークアップし、LPM を終了して要求を処理すると、SYSOSC は正しいトリムを適用して、目的のターゲット周波数で通常の FCL = 有効動作を再開します。

ほとんどの使用事例では大きな影響はなく、必要に応じて FCL および非同期の高速クロック要求の両方を引き続き使用できます。このエラッタが大きな影響を与える主なアプリケーションは、UART が非同期高速クロック要求のソースであるアプリケーションです。この一時的なシフトにより、デバイスが完全にウェークアップするまで実効ボーレートに影響を及ぼす可能性があるためです。

SYSOSC_ERR_05

(続き)

SYSOSC モジュール**回避方法**

1. FCL = 無効のままにします
2. FCL = 有効にする必要がある場合、非同期高速クロック要求を無効化します。

SYSOSC_ERR_06 SYSOSC モジュール**カテゴリ**

機能

機能

SYSOSC が FCL モードで無効化されている LPM 中に非同期クロック要求を受信した場合、MFCLK は 4MHz ではなく 32MHz で動作します

説明

以下のシナリオでは、MFCLK が誤って 32MHz に構成されています:

1. FCL モードを有効にした後、MFCLK を有効にします
2. SYSOSC が無効になる低消費電力モードに移行します (SLEEP2/STOP2/STANDBY0/STANDBY1)。
3. MFCLK を機能クロックとして使用する一部のペリフェラルから非同期リクエストを受信します。非同期リクエストを受信すると、SYSOSC が有効になり、ULPCLK は 32MHz になります。その場合、MFCLK が 32MHz に誤って構成されています。

回避方法

SYSOSC が FCL モードを使用している場合は、通常 SYSOSC がオフになる LPM モードへ移行する際に、ペリフェラル用の MFCLK を有効にしないでください。

TIMER_ERR_01 TIMx モジュール**カテゴリ**

機能

機能

ハードウェア イベントでタイマを開始する場合、キャプチャ モードが誤った値を取得することがあります

説明

いずれかのタイマ インスタンスをキャプチャ モードで使用している場合、ゼロ条件 (ZCOND) やロード条件 (LCOND) によってタイマを開始すると、本来キャプチャすべき値ではなく、ゼロ値やロード値が該当する TIMx.CC レジスタにキャプチャされてしまうことがあります。この問題は、周期やパルス幅のキャプチャといった周期的な使用ケースに影響を及ぼします。

回避方法

以下のソフトウェア フローを使用して、周期またはパルス幅を計算します。回避方法の例については、MSPM0-SDK の `timx_timer_mode_capture_duty_and_period` を参照してください。

1. 0h に設定して、ZCOND または LCOND を無効化します。
2. キャプチャが発生した場合、キャプチャ値は正しく TIMx.CC に格納されます
3. TIMx.CTR をリロード値 (load または 0) に設定してタイマを再起動します。

TIMER_ERR_04 *TIMER* モジュール

カテゴリ

機能

機能

TIMER をゼロ イベントの直前に再有効化すると、再有効化が失われる可能性があります

説明

タイマーをワンショット モードで使用している場合、ゼロ イベント付近で再有効化を行うと再有効化が失われる可能性があります。タイマー有効ビットのハードウェア更新には、1 機能クロック サイクルが必要です。たとえば、タイマーのクロック ソースが 32.768kHz で、クロック分周比が 3 の場合、有効ビットが正しく 0 に設定されるまでに約 100μs かかります。

回避方法

タイマーを再有効化する前に 1 機能クロック サイクル分待機するか、一度タイマーを無効化してから再度有効化してください。

CTRCTL.EN = 0 でカウンタを無効化してから、CTRCTL.EN = 1 で再度有効化します

TIMER_ERR_06 *TIMER* モジュール

カテゴリ

機能

機能

CLKEN ビットに 0 を書き込んでも、カウンタは無効化されません

説明

カウンタ クロック制御レジスタ (CCLKCTL) のクロック イネーブル ビット (CLKEN) に 0 を書き込んでも、タイマは停止しません。

回避方法

カウンタ制御 (CTRCTL) イネーブル (EN) ビットに 0 を書き込むことで、タイマを停止します。

TIMER_ERR_07 *TIMG* モジュール

カテゴリ

機能

機能

初期リピート カウンタの周期は、次のリピートより 1 回だけ少なくなる

説明

タイマ リピート カウンタ モードを使用する場合、以下のリピート カウンタには 0 とロード値の間の遷移が含まれるため、最初のリピートのカウントは後続のリピートより 1 回少なくなります。たとえば、TIMx.RCLD = 0x3 の場合、観測可能な 3 つのゼロ イベントが最初のリピート カウンタに現れ、観測可能な 4 つのゼロ イベントが後続するリピート カウンタ シーケンスに現れます。

回避方法

初期 RCLD 値を想定される RCLD より 1 だけ大きく設定し、リピート カウンタ ゼロ イベント (REPC) の ISR 内で、RCLD を目的の値に設定します。

たとえば、4 回の繰り返しを行う場合は、初期 RCLD 値を RCLD = 0x5 に設定し、REPC 割り込み用のタイマ ISR 内で、RCLD = 0x4 に設定します。これで、すべてのタイマーの繰り返しで、ゼロ / ロード イベントの数が同一になります。

UART_ERR_01	UART モジュール
カテゴリ	機能
機能	STANDBY1 モードへの遷移時に、UART のスタート条件が検出されないことがあります
概要	デバイスが STANDBY1 モードのときに、UART 送信によって開始された非同期高速クロック要求を処理した後、デバイスは STANDBY1 モードに戻ります。STANDBY1 モードへの復帰中に別の UART 送信が開始されると、デバイスはそのデータを正しく検出および受信できません。
回避方法	UART のスタート条件が繰り返し発生することが想定される場合は、STANDBY0 モードまたはそれ以上の低消費電力モードを使用してください。
UART_ERR_02	UART モジュール
カテゴリ	機能
機能	TXE のみが有効な場合、UART 送信終了の割り込みは設定されません
概要	デバイスを送信のみに設定すると (CTL0.TXE = 1、CTL0.RXE = 0)、UART 送信終了 (EOT) 割り込みのトリガはかかりません。デバイスが送受信に設定されている場合 (CTL0.TXE = 1、CTL0.RXE = 1)、EOT は正常にトリガされます
回避方法	UART 送信終了割り込みを使用するときは、CTL0.TXE ビットおよび CTL0.RXE ビットの両方を設定します。ピンを UART 受信として割り当てる必要はないので注意してください。
UART_ERR_04	UART モジュール
カテゴリ	機能
機能	クロックが SYSOSC から LFOSC に遷移する際、高速クロック要求が無効になっていると、UART データが誤って受信される可能性があります
概要	シナリオ: 1. UART の機能クロックとして LFCLK が選択されます 2.3 倍オーバーサンプリングで構成された 9600 のボーレート 3. UART 高速クロック要求が無効になっている状態で、UART 受信転送中に ULPCCLK が SYSOSC から LFOSC に切り替わると、1 ビットが誤って読み取られることがあります
回避方法	LPM モードで UART を使用する場合は、UART 高速クロック要求を有効にしてください。

UART_ERR_05	UART モジュール
カテゴリ	機能
機能	UART モジュールのデバッグ停止機能の制限
概要	本来は既存のフレームを完了して停止することが期待されますが、すべての Tx FIFO 要素が送信されてから通信が停止します。
回避方法	デバッグ停止がアサートされた後は、データが TX FIFO に書き込まれないようにしてください。
UART_ERR_06	UART モジュール
カテゴリ	機能
機能	UART 9 ビットモードでの予期しない RTOUT/Busy/Async の動作
説明	UART 受信タイムアウト (RTOUT) は、マルチノード構成では正しく動作しません。この構成では、1 つの UART がコントローラとして動作し、他の UART ノードはペリフェラルとして機能し、各ペリフェラルは 9 ビット UART モードで異なるアドレスに設定されます。 最初の UART コントローラが UART ペリフェラル 1 と通信し、ペリフェラル 1 のアドレスを最初のバイトとして送信してからデータを送信することで、ペリフェラル 1 がアドレスの一致を確認してデータを受信しました。コントローラがペリフェラル 1 との通信を終了した後、バス上で異なるアドレスに構成された別の UART ペリフェラル (ペリフェラル 2) との通信を直ちに開始すると、ペリフェラル 1 は設定されたタイムアウト期間が経過しても RTOUT を設定しません。ペリフェラル 2 との通信中もペリフェラル 1 の RTOUT カウンタはリセットされ続け、RTOUT が設定されるのは、コントローラがペリフェラル 2 との通信を完了した後になります。 BUSY 要求と Async 要求で同様の動作が確認観察されました。コントローラがバス上の別のペリフェラルと通信中で、アドレスが一致しない場合でも、Busy および Async 要求が設定されます。
回避方法	1 つのコントローラが複数のペリフェラルに接続されたマルチノード UART 通信では、RTOUT / BUSY / 非同期クロック要求の動作は使用しないでください。
UART_ERR_07	UART モジュール
カテゴリ	機能
機能	IDLE LINE モードにおいて、RTOUT カウンタが期待どおりにカウントされません
概要	UART のアイドルラインモードでは、ラインがアイドル状態で、FIFO に何らかの要素がある場合でも、RTOUT カウンタはスタックします。つまり、IDLE LINE モードでは RTOUT 割り込みは動作しません。 アドレスが一致しない場合、Rx ラインでトグルの発生を検出すると RTOUT カウンタがリロードされます。

UART_ERR_07 (続き)

UART モジュール

マルチレスポンド構成の場合、コマンドと他のレスポンド間で通信が行われていると、RTOUT イベントの取得に不定の遅延が発生するおそれがあります。

回避方法

UART モジュールを IDLLINE モード/マルチノード UART アプリケーションのいずれかで使用する場合、RTOUT 機能を有効にしないでください。

UART_ERR_08

UART モジュール

カテゴリ

機能

機能

STAT BUSY は、UART モジュールの正しいステータスを表していません

概要

UART モジュールが無効で TXFIFO でデータが利用可能である場合でも、STAT BUSY は High のままです。

回避方法

TXFIFO ステータスと CTL0.ENABLE レジスタビットをポーリングして、ビジーステータスを識別します。

UART_ERR_09

UART モジュール

カテゴリ

機能

機能

UART を低速で動作させている場合、UART ADDR_MATCH ビットが読み出し時点までに設定されないことがあります。

説明

アドレス一致割り込み中に、コードが ISR にジャンプして FIFO を読み取ります。アドレス一致割り込みがストップ ビットの前に発生するため、UART は RX ラインで送信されたアドレスとしてのデータを正しく受信できないことがあります。

回避方法

ADDR_MATCH ビットがセットされるのを確実にするために、データを読み出す前に 1 UART CLK サイクル分待機します。

UART_ERR_10

UART モジュール

カテゴリ

機能

機能

UART IrDA モードの BUSY ビットの設定が遅延する

説明

IrDA モードでは、UART.STAT.BUSY ビットは IrDA スタートパルスの 2 番目のエッジで設定されます。そのため、BUSY ステータスが正しくセットされる前に、1 ビット分の送信が完了してしまう

UART_ERR_10 (続き)

UART モジュール

可能性があります。この間にソフトウェアが **BUSY** ビットをポーリングすると、IrDA スタートパルス送信中にもかかわらず **UART** がビジーでないと誤って認識されることがあります。この **BUSY** ステータスの動作は **UART** のボーレートに依存します。**UART** 送信が遅い (ボーレートが低い) ほど、**BUSY** が正しく設定されるまでの遅延時間が長くなります。

回避方法

BUSY ステータスをチェックする前に、1 ビット送信の時間分の遅延を挿入します。別の方法としては、**UART.STAT.BUSY == 0x0** の後に **UART.STAT.BUSY == 0x1** をチェックすることで、ボーレートや他の **ISR** に依存しない動的遅延を実現できます。

UART_ERR_11

UART モジュール

カテゴリ

機能

機能

UART 受信タイムアウトが、**STOP** ビット転送中に、予期したタイミングよりも早くカウントを開始する

説明

STOP ビット転送時に、受信タイムアウトが **STOP** ビット転送の途中でカウントを開始する場合があります。その結果、**RXTOSSEL** の設定値が小さすぎる場合、意図しない **RTOUT** 割り込みが発生する可能性があります。たとえば、ボーレートが **1Mbps** で、**RXTOSSEL** が **1** に設定されている場合、想定される **RTOUT** は **STOP** ビット転送の **1μs** 後に発生するはずですが、実際には **RTOUT** 割り込みが **0.5μs** で設定されます。

回避方法

UART.IFLS.RXTOSSEL レジスタは、受信タイムアウト (**RTOUT**) 割り込みが発生するまでのビット時間を選択します。早期割り込みを防止するには、**RXTOSSEL** の値を **1** より大きくする必要があります。受信タイムアウト時間は次のように計算できます。受信タイムアウト = $(\text{RXTOSSEL} - 0.5) / \text{ボーレート}$

VREF_ERR_01

VREF モジュール

カテゴリ

機能

機能

VREF を無効化した後、**VREF READY** ビットはクリアされません

説明

SYSRST 後に **VREF** モジュールを初めて有効にする際は、**VREF READY** ビットを本来の機能として使用することができます。アプリケーション内で **VREF** モジュールを無効にした場合でも、**VREF READY** ビットはクリアされません。このエラーの結果として、**VREF** モジュールを再度有効にした際には、**VREF** モジュールの安定性を示すために **VREF READY** ビットを使用することができません。

回避方法

アプリケーション内で **VREF** モジュールを再度有効にする場合は、**VREF** モジュールを使用する前に、**TIMER** モジュールを用いて最大の **VREF** 立ち上がり時間待機してください。**VREF** の立ち上がり時間については、デバイスのデータシートを参照してください。

8 改訂履歴

資料番号末尾の英字は改訂を表しています。その改訂履歴は英語版に準じています。

Changes from AUGUST 30, 2025 to JULY 30, 2026 (from Revision D (August 2025) to Revision E (July 2026))

	Page
• ADC_ERR_02 の説明を更新しました.....	5
• ADC_ERR_02 回避策を更新しました.....	5
• ADC_ERR_05 回避策を更新しました.....	6
• ADC_ERR_05 の説明を更新しました.....	6
• ADC_ERR_06 回避策を更新しました.....	6
• ADC_ERR_06 の説明を更新しました.....	6
• ADC_ERR_11 モジュールを更新しました.....	7
• ADC_ERR_11 機能を更新しました.....	7
• ADC_ERR_11 の説明を更新しました.....	7
• ADC_ERR_11 回避策を更新しました.....	7
• CPU_ERR_01 機能を更新しました.....	9
• CPU_ERR_01 の説明を更新しました.....	9
• CPU_ERR_01 回避策を更新しました.....	9
• CPU_ERR_02 機能を更新しました.....	10
• CPU_ERR_02 の説明を更新しました.....	10
• CPU_ERR_02 回避策を更新しました.....	10
• CPU_ERR_03 機能を更新しました.....	10
• CPU_ERR_03 の説明を更新しました.....	10
• CPU_ERR_03 回避策を更新しました.....	10
• CPU_ERR_04 カテゴリを更新しました.....	10
• CPU_ERR_04 モジュールを更新しました.....	10
• CPU_ERR_04 機能を更新しました.....	10
• CPU_ERR_04 の説明を更新しました.....	10
• CPU_ERR_04 回避策を更新しました.....	10
• CRC/CRCP_ERR_01 カテゴリを更新しました.....	11
• CRC/CRCP_ERR_01 モジュールを更新しました.....	11
• CRC/CRCP_ERR_01 機能を更新しました.....	11
• CRC/CRCP_ERR_01 説明を更新しました.....	11
• CRC/CRCP_ERR_01 回避策を更新しました.....	11
• FCC_ERR_01 カテゴリを更新しました.....	11
• FCC_ERR_01 回避策を更新しました.....	11
• FCC_ERR_01 機能を更新しました.....	11
• FCC_ERR_01 の説明を更新しました.....	11
• FCC_ERR_01 モジュールを更新しました.....	11
• FLASH_ERR_02 機能を更新しました.....	12
• FLASH_ERR_02 の説明を更新しました.....	12
• FLASH_ERR_02 回避策を更新しました.....	12
• FLASH_ERR_03 を追加.....	12
• FLASH_ERR_08 モジュールを更新しました.....	14
• FLASH_ERR_08 機能を更新しました.....	14
• FLASH_ERR_08 の説明を更新しました.....	14
• FLASH_ERR_08 回避策を更新しました.....	14
• GPIO_ERR_04 モジュールを更新しました.....	15
• GPIO_ERR_04 カテゴリを更新しました.....	15
• GPIO_ERR_04 機能を更新しました.....	15

• GPIO_ERR_04 回避策を更新しました.....	15
• GPIO_ERR_04 の説明を更新しました.....	15
• GPIO_ERR_06 カテゴリを更新しました.....	16
• GPIO_ERR_06 モジュールを更新しました.....	16
• GPIO_ERR_06 機能を更新しました.....	16
• GPIO_ERR_06 の説明を更新しました.....	16
• GPIO_ERR_06 回避策を更新しました.....	16
• I2C_ERR_09 の説明を更新しました.....	20
• I2C_ERR_09 回避策を更新しました.....	20
• I2C_ERR_13 モジュールを更新しました.....	20
• I2C_ERR_13 機能を更新しました.....	20
• I2C_ERR_13 回避策を更新しました.....	20
• I2C_ERR_13 の説明を更新しました.....	20
• I2C_ERR_15 カテゴリを更新しました.....	21
• I2C_ERR_15 モジュールを更新しました.....	21
• I2C_ERR_15 機能を更新しました.....	21
• I2C_ERR_15 回避策を更新しました.....	21
• I2C_ERR_15 の説明を更新しました.....	21
• PMCU_ERR_10 を追加.....	22
• PMCU_ERR_13 機能を更新しました.....	22
• PMCU_ERR_13 の説明を更新しました.....	22
• PMCU_ERR_13 回避策を更新しました.....	22
• RST_ERR_01 機能を更新しました.....	23
• RST_ERR_01 の説明を更新しました.....	23
• RST_ERR_01 回避策を更新しました.....	23
• SPI_ERR_03 回避策を更新しました.....	24
• SPI_ERR_07 の説明を更新しました.....	25
• SPI_ERR_07 回避策を更新しました.....	25
• SPI_ERR_08 モジュールを更新しました.....	26
• SPI_ERR_08 機能を更新しました.....	26
• SPI_ERR_08 の説明を更新しました.....	26
• SPI_ERR_08 回避策を更新しました.....	26
• SPI_ERR_09 モジュールを更新しました.....	26
• SPI_ERR_09 の説明を更新しました.....	26
• SPI_ERR_09 回避策を更新しました.....	26
• SPI_ERR_10 カテゴリを更新しました.....	26
• SPI_ERR_10 モジュールを更新しました.....	26
• SPI_ERR_10 機能を更新しました.....	26
• SPI_ERR_10 の説明を更新しました.....	26
• SPI_ERR_10 回避策を更新しました.....	26
• SYSCTL_ERR_03 を追加.....	26
• SYSCTL_ERR_05 カテゴリを更新しました.....	27
• SYSCTL_ERR_05 モジュールを更新しました.....	27
• SYSCTL_ERR_05 機能を更新しました.....	27
• SYSCTL_ERR_05 の説明を更新しました.....	27
• SYSCTL_ERR_05 回避策を更新しました.....	27
• SYSCTL_ERR_06 カテゴリを更新しました.....	27
• SYSCTL_ERR_06 モジュールを更新しました.....	27
• SYSCTL_ERR_06 機能を更新しました.....	27
• SYSCTL_ERR_06 の説明を更新しました.....	27
• SYSCTL_ERR_06 回避策を更新しました.....	27

• SYSCTL_ERR_11 カテゴリを更新しました.....	27
• SYSCTL_ERR_11 モジュールを更新しました.....	27
• SYSCTL_ERR_11 機能を更新しました.....	27
• SYSCTL_ERR_11 回避策を更新しました.....	27
• SYSCTL_ERR_11 の説明を更新しました.....	27
• SYSOSC_ERR_04 カテゴリを更新しました.....	29
• SYSOSC_ERR_04 モジュールを更新しました.....	29
• SYSOSC_ERR_04 回避策を更新しました.....	29
• SYSOSC_ERR_04 機能を更新しました.....	29
• SYSOSC_ERR_04 の説明を更新しました.....	29
• SYSOSC_ERR_05 カテゴリを更新しました.....	29
• SYSOSC_ERR_05 モジュールを更新しました.....	29
• SYSOSC_ERR_05 機能を更新しました.....	29
• SYSOSC_ERR_05 回避策を更新しました.....	29
• SYSOSC_ERR_05 の説明を更新しました.....	29
• SYSOSC_ERR_06 カテゴリを更新しました.....	30
• SYSOSC_ERR_06 モジュールを更新しました.....	30
• SYSOSC_ERR_06 機能を更新しました.....	30
• SYSOSC_ERR_06 の説明を更新しました.....	30
• SYSOSC_ERR_06 回避策を更新しました.....	30
• TIMER_ERR_04 の説明を更新しました.....	31
• TIMER_ERR_04 回避策を更新しました.....	31
• TIMER_ERR_06 モジュールを更新しました.....	31
• TIMER_ERR_07 を追加.....	31
• UART_ERR_10 モジュールを更新しました.....	34
• UART_ERR_10 機能を更新しました.....	34
• UART_ERR_10 の説明を更新しました.....	34
• UART_ERR_10 回避策を更新しました.....	34
• UART_ERR_11 モジュールを更新しました.....	35
• UART_ERR_11 機能を更新しました.....	35
• UART_ERR_11 の説明を更新しました.....	35
• UART_ERR_11 回避策を更新しました.....	35

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含みいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとし、TI は一切の責任を拒否します。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、[TI の販売条件](#)、[TI の総合的な品質ガイドライン](#)、[ti.com](https://www.ti.com) または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

Copyright © 2026, Texas Instruments Incorporated

最終更新日：2025 年 10 月