

Errata

MSPM0G351x, MSPM0G151x, MSPM0G351x-Q1 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 機能アドバイザリ	1
2 プログラム済みのソフトウェアアドバイザリ	2
3 デバッグ専用のアドバイザリ	2
4 コンパイラアドバイザリによって修正	3
5 デバイスの命名規則	3
5.1 デバイスの記号体系およびリビジョン識別方法	3
6 アドバイザリの説明	5
7 商標	25
8 改訂履歴	25

1 機能アドバイザリ

デバイスの動作、機能、パラメータに影響を与えるアドバイザリ。

✓ 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 A(プロトタイプ X マーク付き製品)	Rev C
ADC_ERR_06	✓	✓
ADC_ERR_10	✓	✓
AES_ERR_01	✓	✓
CPU_ERR_02	✓	✓
CPU_ERR_03	✓	✓
FLASH_ERR_01	✓	✓
FLASH_ERR_03	✓	✓
FLASH_ERR_04	✓	✓
FLASH_ERR_05	✓	✓
FLASH_ERR_08	✓	✓
GPIO_ERR_04	✓	✓
I2C_ERR_04	✓	✓
I2C_ERR_05	✓	✓
I2C_ERR_06	✓	✓
I2C_ERR_07	✓	✓
I2C_ERR_08	✓	✓
I2C_ERR_09	✓	✓
I2C_ERR_10	✓	✓
I2C_ERR_13	✓	✓

エラッタ番号	Rev A(プロト タイプXマー ク付き製品)	Rev C
KEYSTORE_ERR_01	✓	✓
MATHACL_ERR_01	✓	✓
MATHACL_ERR_02	✓	✓
PMCU_ERR_09	✓	✓
PMCU_ERR_10	✓	
PMCU_ERR_11	✓	✓
RST_ERR_01	✓	✓
RTC_ERR_01	✓	✓
SPI_ERR_02	✓	✓
SPI_ERR_04	✓	✓
SPI_ERR_05	✓	✓
SPI_ERR_06	✓	✓
SPI_ERR_07	✓	✓
SRAM_ERR_03	✓	
SYSCTL_ERR_01	✓	✓
SYSCTL_ERR_02	✓	✓
SYSCTL_ERR_03	✓	✓
SYSCTL_ERR_04	✓	✓
SYSOSC_ERR_01	✓	✓
SYSOSC_ERR_02	✓	✓
SYSPLL_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_10	✓	✓
UART_ERR_11	✓	✓

2 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、指定したリビジョンに問題が存在することを示します。

3 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。

エラッタ番号	リビジョン A	Rev B
GPIO_ERR_03	✓	✓

4 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、指定したリビジョンに問題が存在することを示します。

5 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。

5.1 デバイスの記号体系およびリビジョン識別方法

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し **表 5-1**、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。

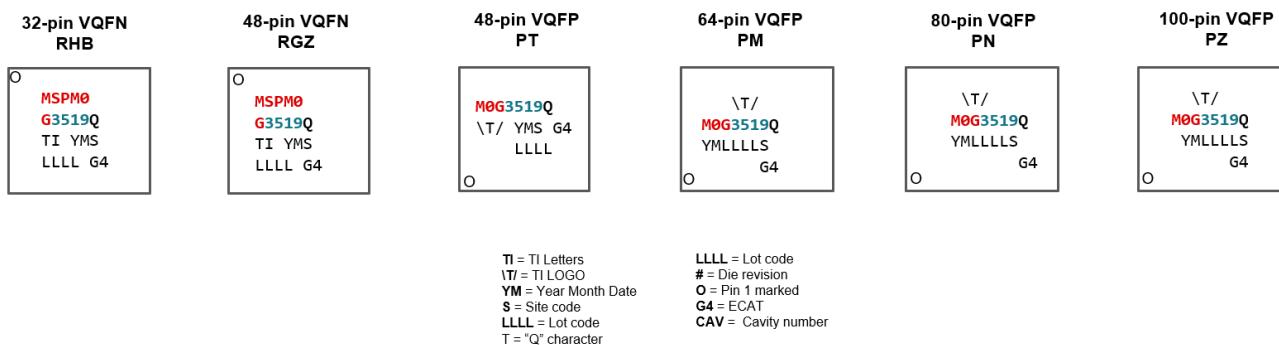


図 5-1. パッケージの記号表記

表 5-1. 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)
A	1
C	2

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リには、リビジョン文字に基づいて、特定のバイ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。

6 アドバイザリの説明

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 のデジタル出力コードに ± 64 LSB の固定ジャンプが発生しますが、ADC 入力電圧は変化しません。

最悪のシナリオ (-40°C) では、12 ビットモードでのエラー率は 24M の変換サンプルあたり 1 です。
(VDD 電圧と使用されるリファレンス電圧はエラー率に影響を与えません)

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ョルドを外れた場合 (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ョルド判定を使用)、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ョルドは、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_10

ADC モジュール

カテゴリ

機能

機能

PA15 がトグルしているときに ADCMEMRES スワップが発生します

説明

セットアップ条件:

1. ADC は反復シーケンスモードです。

2. ADC は任意のチャネルシーケンスを使用してデータを読み取ることができます。

3. PA15 は、外部デバイスまたは MCU 自体 (I2C や PWM など) によってトグルされています。

観察:

ソフトウェアが ADC 変換を開始すると、MEMRES データが入れ替わることがあります。つまり、MEMRES0 に格納されるべきデータは MEMRES1 に格納され、MEMRES1 のデータは MEMRES2 に格納され...というように続けます。反復モードでこのエラーが発生すると、最後の MEMRES が MEMRES0 に格納されます。

ADC_ERR_10 (続き)ADC モジュール

PA15 のトグル信号は ADC の変換クロックに影響を与える可能性があります。その結果、ADC は正しいチャンネルデータが入力される前に前回の結果を保存します。

回避方法

PA15 では高速スイッチング信号 (12kHz 以上) の使用を避けてください。

AES_ERR_01

AES モジュール

カテゴリ

機能

機能

AES Saved Context Ready 割り込みが予想どおりに生成されていません

説明

Saved Context Ready 割り込みが生成されていません。いずれかの AES レジスタに対してアクセス(読み取りまたは書き込み)が行われた場合に、割り込みが生成されます。

回避方法

ポーリングベースのメカニズムを使用して、割り込みをせず、CTRL レジスタの保存済みコンテキストレディのステータスビットを確認します。

CPU_ERR_02

CPU モジュール

カテゴリ

機能

機能

CPUSS のプリフェッチ機能を無効にする制限

説明

保留中のフラッシュメモリアクセスがある場合、CPU プリフェッチを無効にしても無効にはなりません。

回避方法

プリフェッチャーを無効にし、SYSCTL でシャットダウンメモリへのメモリアクセス (SHUTDNSTORE) を発行します。これは SYSCTL.SOCLOCK.SHUTDNSTORE0; で実行できます。

メモリアクセスが完了すると、プリフェッチャーは無効になります。

例:

CPUSS.CTL.PREFETCH = 0x0、プリフェッチャーを無効にします
 SYSCTL.SOCLOCK.SHUTDNSTORE0、シャットダウンメモリへのメモリアクセス

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

CPU_ERR_03 (続き) CPU モジュール

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:

```
CPUSS.CTL.PREFETCH = 0x0; // プリフェッチャを無効化  
SYSCLOCK.SHUTDNSTORE0; // シャットダウン 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.PREFETCH = 0x1; // プリフェッチャを再有効化
```

FLASH_ERR_01 FLASH モジュール

カテゴリ

機能

機能

FACTORY 領域にアクセスすると、フラッシュのウェイト状態が 2 のハードフォルトが発生します。

概要

フラッシュのウェイト状態が 2 に設定されているときに FACTORY 領域にアクセスすると、ハードフォルトがトリガれます。MCLK が 32MHz を超える値に設定されている場合、フラッシュのウェイト状態は 2 に設定する必要があります。

回避方法

FACTORY 領域にアクセスするには、MCLK を低い周波数に設定し、フラッシュのウェイト状態を 0 または 1 に設定してください。フラッシュのウェイト状態が 2 未満のときに、工場出荷領域にアクセスします(MCLK は 32MHz 以下とします)。実行時にアプリケーションがこれらの値にアクセスする必要がある場合は、SRAM、MAIN フラッシュ、または DATA フラッシュにデータをキャッシュしてください。標準値は、温度センサの較正值です。

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。

FLASH_ERR_03

(続き)

FLASH モジュール**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

NONMAIN または Factory 領域でエラーが発生した場合、SYSCTL_DEDERRADDR に誤ったアドレスが報告される

説明

FLASHDED エラーが発生すると、データの最上位バイト (MSB) が切り捨てられます。デバイスのメモリ制限では、最上位バイトは MAIN フラッシュの復帰アドレスに影響を与えません。NONMAIN フラッシュまたは Factory 領域の場合、MSB は 0x41xx.xxxx である必要があります。

回避方法

SysCtl_DEDERRADDR の戻りアドレスで 0x00Cxxxxx 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。

メインフラッシュ DED の場合、SYSCTL_DEDERRADDR をそのまま使用できます。

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 をプログラムすると、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_08**FLASH モジュール****カテゴリ**

機能

FLASH_ERR_08

(続き)

FLASH モジュール

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. 0x40BFFFFF ~ 0x41C00000 3. 0x41C007FF ~ 0x41C40000

回避方法

番号

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 ファストウェイクの設定により、PAD データが DIN レジスタに入力されません。

説明

CTL レジスタのファストウェイクのみのビットを設定し、実行モードでデータを PAD に強制的に入力しても、PAD のデータは DIN レジスタに反映されません。これは、CTL レジスタの設定により、PAD から DIN レジスタへのデータの流れが阻止されるためです。

回避方法

PAD のデータが DIN レジスタに入力されることが予想される場合は、GPIO ファストウェイクのみの機能を使用しないでください。

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。

I2C_ERR_04 (続き) I2C モジュール

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

進行中の転送中にアクティブ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、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 の状態でコントローラが IDLE 状態に遷移すると、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。ターゲットの BUSUSY がセットされ(アクティブ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)、BUSY はクリアされません。これは、コントローラが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

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 50 μ s 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 60 μ s で、50 μ s 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 μ s および 20 μ s 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

I2C_ERR_06 (続き) I2C モジュール

に、1 SCL Low 期間(30 μ s)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(30 μ s)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 60 μ s かかります。ただし、高タイムアウトカウンタには上限(50 μ s)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

連続 CTR レジスタへの書き込みでは、次の CTR .START によって正しく開始条件が発生しません。

回避方法

CTR.START を含むすべての CTR ビットを 1 回の書き込みで書き込むか、CTR 書き込みと CTR.START 書き込みの間に 1 クロック サイクル待機します。

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

概要

RXDONE 割り込みが発生したとき、FIFO は最新のデータに対して常に更新されるとは限りません。

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。

I2C_ERR_09 (続き) I2C モジュール

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

概要

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。すべての I2C 転送は 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を維持してください(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 = $3 \times 32 \text{ MHz} / (4 \text{ MHz} / 8) = 192 \text{ CPU サイクル}$

KEYSTORE_ERR_

01

キーストアモジュール

カテゴリ

機能

機能

STATUS.STATUS の値は、キーアクセスがない場合、0 または 1 になります。

KEYSTORE_ERR_

01 (続き)

キーストアモジュール

説明

STATUS.STATUS のリセット値は 1 で、以下の条件で 0 になります。1)リセット後、レジスタウンドウを介したデバッグアクセスは 0x00 を返します。2)リセット後、最初の CPU 読み取りは 0x01 を返し、その後の CPU 読み取りは 0x00 を返します。3)リセット後、最初に他のキーストアレジスタを読み取り、次に STATUS.STATUS を読み取ると、0x00 が返されます。

回避方法

STATUS.STATUS=0x0 は「エラーなし」を意味します。スロットが有効かどうか (キーが存在するかどうか) を確認するには、STATUS.VALID を確認してください。

MATHACL_ERR_0

1

MATHACL モジュール

カテゴリ

機能

機能

MATHACL ステータスエラービットはクリアされません

概要

mathacl によってステータスエラーが生成された場合 (例:0 で除算)、STATUS レジスタがはクリアされません。

回避方法

ペリフェラルをリセットして、STATUS ビットをクリアします。

MATHACL_ERR_0

2

MATHACL モジュール

カテゴリ

機能

機能

MATHACL で $\text{COS}(-180)$ を実行すると -1 ではなく 1 が返され、 $\text{SIN}(-90)$ でも -1 の代わりに 1 が返されます

概要

$\text{COS}(-180)$ または $\text{SIN}(-90)$ を実行すると、MATHACL は -1 ではなく 1 を返します

回避方法

回避策はありません。ソフトウェアで結果をネガティブに補正してください。

PMCU_ERR_09

PMCU モジュール

カテゴリ

機能

機能

POR (NRST > 1s) リセット後、RSTCAUSE が誤って 0xC に更新されます。

説明

NRST を使用して POR (NRST > 1s) リセットをトリガーすると、RSTCAUSE が誤って 0xC に更新されます。これは 0x2 に更新されるべきです。

PMCU_ERR_09 (続)

き)

PMCU モジュール**回避方法**

リセット原因を確認する必要がある場合は、RSTCAUSE ステータスを取得した後、利用可能な SHUTDOWN メモリバイト (SHUTDNSTOREx) の 1 つを使用して、ゼロでないデータを保存してください。RSTCAUSE が 0xC を返す場合、SHUTDNSTOREx データがクリアされると POR が発生し、SHUTDNSTOREx データが維持されると BOR が発生します。

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

概要

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 などの他のモジュールのセトリン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_11**PMCU モジュール****カテゴリ**

機能

機能

NRST < 1 のパルスにより、シャットダウンモードで誤った `rstcause` が発生

概要

次の条件下で `rstcause` の値が正しくありません。予想される `rstcause` は 0x05 です。

- (i) デバイスをシャットダウンモードに構成済み
- (ii) WFI()を呼び出し
- (iii) デバイスをシャットダウンモードから復帰させるために NRST < 1 秒のパルスを与えます

回避方法

回避方法はありません。

RST_ERR_01**RST モジュール****カテゴリ**

機能

機能

LFCLK_IN が LFCLK のソースとして選択されており、かつ LFCLK_IN が無効になっている場合、NRST リリースは検出されません

説明

LFCLK = LFCLK_IN で、LFCLK_IN を無効にすると、NRST パルスエッジ検出を見逃されし、デバイスがリセットから復帰しないコーナーシナリオが発生します。この問題は、NRST パルス幅

RST_ERR_01 (続き) RST モジュール

が $608\mu\text{s}$ 未満のときに見られます。NRST パルスが $608\mu\text{s}$ を超える場合は、リセットは通常どおり表示されます。

回避方法

この問題を回避するため、 $608\mu\text{s}$ よりも高い NRST パルス幅を維持します。

RTC_ERR_01

RTC モジュール

カテゴリ

機能

機能

一部の RTC 割り込みは、STANDBY1 では使用できません

概要

STANDBY1 のとき、RTCRDY 割り込みと RTC_PRESCALER1 割り込みではデバイスをウェークアップできません。

回避方法

RTC で STANDBY1 からデバイスをウェークアップするときは、RTC_ALARM や RTC_PRESCALER0 などの利用可能な他の割り込みを使用します。

SPI_ERR_02

SPI モジュール

カテゴリ

機能

機能

低消費電力モード (LPM) からウェークアップした後の、SPI クロックとデータバイトの欠落

概要

デバイスが低消費電力状態からウェークアップした後、SPI モジュールは、送信された最初のバイトの最初の数クロックサイクルおよびデータビットを適切に伝搬できません。

回避方法

ウェークアップ後の SPI データの整合性を維持するには、LPM を開始および終了するときに次のシーケンスを使用します：

1. SPI モジュールを無効にする
2. 割り込み (WFI) を待機する - LPM に入る
3. LPM からのウェイクアップ (任意のソース)。
4. SPI モジュールを有効にする。

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。

SPI_ERR_04 (続き) SPI モジュール

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ではなく、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

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 を受信した後でも 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/読み取りストロー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 にラッピング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。

SPI_ERR_07 (続き) SPI モジュール

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。

SRAM_ERR_03 SRAM モジュール

カテゴリ

機能

機能

SRAM パリティおよび ECC 機能は、Rev A デバイスではサポートされていません

説明

Rev A デバイスでは、SRAM パリティと ECC 機能はサポートされていません。Rev A デバイスでは、SRAM パリティおよび ECC 機能を使用しないでください。

回避方法

なし。

SYSCTL_ERR_01 SYSCTL モジュール

カテゴリ

機能

機能

SW-POR 機能は、HW_POR と組み合わせて使用できます

説明

ソフトウェア トリガ POR を生成するために正しいキーを使って LFSSRST レジスタに書き込むと、RSTCAUSE レジスタには、予測される 0x3 (ソフトウェア トリガ POR) ではなく 0x2 (NRST トリガ POR) が表示されます。これは、SW-POR 機能が HW-POR パスと組み合わされているためです。

回避方法

番号

SYSCTL_ERR_02 SYSCTL モジュール

カテゴリ

機能

機能

BOOTRST の後には、SYSSTATUS.FLASHSEC はゼロ以外になります

SYSCTL_ERR_02

(続き)

SYSCTL モジュール**説明**

BOOTRST/ブートコード完了後、SYSSTATUS.FLASHSEC はゼロ以外になります。これは、お客様がブートコードが完了した後に表示されます。

回避方法

番号

SYSCTL_ERR_03 SYSCTL モジュール

カテゴリ

機能

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。

SYSCTL_ERR_04 SYSCTL モジュール

カテゴリ

機能

機能

SYSRESET 後に *SYSSTATUS.FLASHSEC* がクリアされません。

説明

SYSSTATUS.FLASHSEC は、*SYSRESET* 後にクリアされず、*SYSSTATUSCLR* レジスタに書き込まれることでのみクリアされます。

回避方法

番号

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が受信されます。

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。

SYSPLL_ERR_01 SYSPLL モジュール

カテゴリ

機能

機能

SYSPLL 周波数が有効になっているとき、正しい周波数にロックされない場合がある。

説明

SYSCTL.HSCLKEN レジスタ内の SYSPLLLEN ビットを 1 に設定すると、SYSPLL は位相同期ループ のサーチを実行します。周波数が正しい値に設定されないと、サーチ動作が失敗することがあります。その場合は、得られる周波数が設定値と大きく異なってしまいます。

回避方法

SYSPLLLEN ビットが 1 に設定されている間は、周波数クロック カウンタ (FCC) を使用して SYSPLL の出力周波数を確認してください。正しい周波数に一度修正すれば、その後は無効化 (SYSPLLLEN = 0) および再有効化 (SYSPLLLEN = 1) されるまで維持されます。再有効化後は、ロック サーチが再実行されるため、SYSPLL 出力周波数も再確認する必要があります。

回避方法 1: SYSPLLCLK0 を FCC の CLK 入力として、LFCLK をトリガ ソースとしてそれぞれ設定します。FCC を実行し、設定した SYSPLL 周波数に対する測定値を LFCLK を基準として確認します。たとえば、SYSPLL = 80MHz、LFCLK = 32kHz の場合、FCC カウントは $80,000,000 / 32,768 = 2441$ になります。実際のカウント値はクロック精度に依存するため、許容範囲として $\pm 5\%$ を見込むことが推奨されます。FCC の推定実行時間は 30μs です。

FCC の設定: SYSCTL.GENCLKCFG.FCCTRIGCNT = 0,
 SYSCTL.GENCLKCFG.FCCTRIGSRC = 1, SYSCTL.GENCLKCFG.FCCSELCLK = 4。
 FCC が異常値の場合は、SYSPLLLEN を一度 0 にしてから 1 に戻します (SYSPLL をディスエーブルしてから再度イネーブルにする)。再度 FCC チェックを実行します。

回避方法 2: SYSOSC/2 を CLK_OUT ピンから出力し、その信号を FCC_IN に配線します。
 SYSPLLCLK0 を FCC CLK として、FCC_IN をトリガソースとしてそれぞれ使用します。16 クロ

SYSPLL_ERR_01

(続き)

SYSPLL モジュール

ック サイクルにわたって FCC を実行し、SYSOSC を基準として、設定された SYSPLL 周波数の値を確認します。たとえば、SYSPLL = 80MHz および SYSOSC/2 = 16MHz の場合、得られる FCC カウントは $80,000,000/16,000,000 * 16 = 80$ になります。実際のカウント値はクロック精度に依存するため、許容範囲として $\pm 5\%$ を見込むことが推奨されます。FCC の推定実行時間は 1 μ s です。

FCC の設定: SYSCTL.GENCLKCFG.FCCTRIGCNT = 0x0F、
 SYSCTL.GENCLKCFG.FCCTRIGSRC = 0、SYSCTL.GENCLKCFG.FCCSELCLK = 4。

FCC が異常値の場合は、SYSPLEN を一度 0 にしてから 1 に戻します (SYSPLL をディスエーブルしてから再度イネーブルにする)。再度 FCC チェックを実行します。

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 100 μ s 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

TIMA と TIMG 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

概要

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

初期リピート カウンタの周期は、次のリピート モジュールより 1 回だけ少なくなる

カテゴリ

機能

TIMER_ERR_07 (続)

き) 初期リピートカウンタの周期は、次のリピートモジュールより1回だけ少なくなる

機能**TIMER****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回少くなります。たとえば、 $\text{TIMx.RCLD} = 0x3$ 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 $\text{RCLD} = 0x5$ に設定し、REPC割り込み用のタイマ ISR 内で、 $\text{RCLD} = 0x4$ 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 ($\text{CTL0.TXE} = 1$ 、 $\text{CTL0.RXE} = 0$)、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 ($\text{CTL0.TXE} = 1$ 、 $\text{CTL0.RXE} = 1$)、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つていると、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つている状態で、UART 受信転送中に ULPCLK 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。

UART_ERR_06 (続
き)

UART モジュール
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。

マルチレスポンダ構成の場合、コマンダと他のレスポンダ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、ビージーステータスを識別します。

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

UART_ERR_10 (続

き)

UART モジュール

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。別の方針としては、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、RXTOSEL 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、RXTOSEL 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 1 μ s 後に発生するはずですが、実際には RTOUT 割り込みが 0.5 μ s で設定されます。

回避方法

UART.IFLS.RXTOSEL 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、RXTOSEL 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 = (RXTOSEL - 0.5) / ボーレート

7 商標

すべての商標は、それぞれの所有者に帰属します。

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 JULY 30, 2025 to NOVEMBER 30, 2025 (from Revision B (July 2025) to Revision C (November 2025))
Page

• PMCU_ERR_12 を削除。	1
• ADC_ERR_06 モジュールを更新しました。	5
• ADC_ERR_06 機能を更新しました。	5
• ADC_ERR_06 回避策を更新しました。	5
• ADC_ERR_06 の説明を更新しました。	5
• ADC_ERR_10 の説明を更新しました。	5
• ADC_ERR_10 回避策を更新しました。	5
• CPU_ERR_02 機能を更新しました。	6
• CPU_ERR_02 回避策を更新しました。	6

• CPU_ERR_03 機能を更新しました.....	6
• CPU_ERR_03 の説明を更新しました.....	6
• CPU_ERR_03 回避策を更新しました.....	6
• FLASH_ERR_08 モジュールを更新しました.....	8
• FLASH_ERR_08 機能を更新しました.....	8
• FLASH_ERR_08 の説明を更新しました.....	8
• FLASH_ERR_08 回避策を更新しました.....	8
• I2C_ERR_07 回避策を更新しました.....	11
• I2C_ERR_09 の説明を更新しました.....	11
• I2C_ERR_09 回避策を更新しました.....	11
• I2C_ERR_13 カテゴリを更新しました.....	12
• I2C_ERR_13 モジュールを更新しました.....	12
• I2C_ERR_13 機能を更新しました.....	12
• I2C_ERR_13 回避策を更新しました.....	12
• I2C_ERR_13 の説明を更新しました.....	12
• KEYSTORE_ERR_01 モジュールを更新しました.....	12
• KEYSTORE_ERR_01 機能を更新しました.....	12
• KEYSTORE_ERR_01 の説明を更新しました.....	12
• KEYSTORE_ERR_01 回避策を更新しました.....	12
• SPI_ERR_07 の説明を更新しました.....	16
• SPI_ERR_07 回避策を更新しました.....	16
• SYSCtrl_ERR_01 モジュールを更新しました.....	17
• SYSCtrl_ERR_01 機能を更新しました.....	17
• SYSCtrl_ERR_01 の説明を更新しました.....	17
• SYSCtrl_ERR_01 回避策を更新しました.....	17
• SYSCtrl_ERR_02 モジュールを更新しました.....	17
• SYSCtrl_ERR_02 機能を更新しました.....	17
• SYSCtrl_ERR_02 の説明を更新しました.....	17
• SYSCtrl_ERR_02 回避策を更新しました.....	17
• SYSCtrl_ERR_04 回避策を更新しました.....	19
• SYSCtrl_ERR_04 モジュールを更新しました.....	19
• SYSCtrl_ERR_04 機能を更新しました.....	19
• SYSCtrl_ERR_04 の説明を更新しました.....	19
• SYSPPLL_ERR_01 カテゴリを更新しました.....	20
• SYSPPLL_ERR_01 モジュールを更新しました.....	20
• SYSPPLL_ERR_01 機能を更新しました.....	20
• SYSPPLL_ERR_01 の説明を更新しました.....	20
• SYSPPLL_ERR_01 回避策を更新しました.....	20
• TIMER_ERR_04 の説明を更新しました.....	21
• TIMER_ERR_04 回避策を更新しました.....	21
• TIMER_ERR_07 カテゴリを更新しました.....	21
• TIMER_ERR_07 モジュールを更新しました.....	21
• TIMER_ERR_07 の説明を更新しました.....	21
• TIMER_ERR_07 回避策を更新しました.....	21
• TIMER_ERR_07 機能を更新しました.....	21
• UART_ERR_10 モジュールを更新しました.....	24
• UART_ERR_10 機能を更新しました.....	24
• UART_ERR_10 の説明を更新しました.....	24
• UART_ERR_10 回避策を更新しました.....	24
• UART_ERR_11 モジュールを更新しました.....	25
• UART_ERR_11 機能を更新しました.....	25

- UART_ERR_11 の説明を更新しました..... 25
- UART_ERR_11 回避策を更新しました..... 25

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默示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または默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、(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、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](#) 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 © 2025, Texas Instruments Incorporated

最終更新日：2025 年 10 月