参考資料

Errata

MSPM0G3x0x、MSPM0G1x0x、MSPM0G3x0x-Q1 マイコン

TEXAS INSTRUMENTS

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

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 B	Rev C
ADC_ERR_01	✓	✓
ADC_ERR_02	✓	✓
ADC_ERR_05	✓	✓
ADC_ERR_06	✓	✓
BSL_ERR_01	✓	✓
CLK_ERR_01	✓	✓
COMP_ERR_02	✓	✓
COMP_ERR_03	✓	✓
CPU_ERR_01	✓	✓
CPU_ERR_02	✓	✓
CPU_ERR_03	✓	✓
DMA_ERR_01	✓	✓
FLASH_ERR_02	✓	✓
FLASH_ERR_04	✓	✓
FLASH_ERR_05	✓	✓
FLASH_ERR_06	✓	✓
FLASH_ERR_08	✓	✓
GPIO_ERR_01	✓	✓
GPIO_ERR_04	✓	✓
I2C_ERR_01	✓	✓
I2C_ERR_02	✓	✓
I2C_ERR_03	✓	✓

エラッタ番号	Rev B	Rev C
I2C_ERR_04	✓	✓
I2C_ERR_05	✓	✓
I2C_ERR_06	✓	✓
I2C_ERR_07	✓	✓
I2C_ERR_08	✓	✓
I2C_ERR_09	✓	✓
I2C_ERR_10	✓	✓
I2C_ERR_13	✓	✓
MATHACL_ERR_01	✓	✓
MATHACL_ERR_02	✓	✓
PMCU_ERR_08	✓	✓
PMCU_ERR_10	✓	✓
PWREN_ERR_01	✓	✓
RST_ERR_01	✓	✓
RTC_ERR_01	✓	✓
SPI_ERR_01	✓	✓
SPI_ERR_02	✓	✓
SPI_ERR_03	✓	✓
SPI_ERR_04	✓	✓
SPI_ERR_05	✓	✓
SPI_ERR_06	✓	✓
SPI_ERR_07	✓	✓
SRAM_ERR_02	✓	✓
SYSCTL_ERR_02	✓	✓
SYSCTL_ERR_03	✓	✓
SYSCTL_ERR_04	✓	✓
SYSOSC_ERR_01	✓	✓
SYSOSC_ERR_02	✓	✓
SYSPLL_ERR_01	✓	✓
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	✓	✓

エラッタ番号	Rev B	Rev C
UART_ERR_11	✓	✓
VREF_ERR_01	✓	✓
VREF_ERR_02	✓	✓
WWDT_ERR_01	✓	✓
WWDT_ERR_02	✓	✓

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、指定され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	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、指定され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。

6 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 は、定期的なトリガの間で高速クロック要求を解放しません

説明

ADC が反復モードに設定され、イベント ファブリックを介して周期的にトリガされている場合、トリガ間で高速クロック要求を解除しないことがあります。その場合、ADC はクロック要求をホールドし、余分な電力を消費します。

回避方法

ADC をシングル モード (チャネル/シーケンスの終了時に ADC を無効化して処理を完了) で構成し、その後、ADC シーケンスの終了時にソフトウェア割り込みを使用して ADC を再度有効化し、次のトリガを待機させる方法を使用します。

ADC_ERR_05 *ADC モジュール*

カテゴリ

機能

機能

IP を有効にする前に生成された HW イベントは、キューに残ります

説明

ADC が HW イベント トリガ モードで設定されていて、有効化前にトリガが生成された場合、ADC 葉キューに残ります。ADC が有効化されると、変換がトリガされます。

回避方法

ADC を HW トリガ モードで設定した後、外部トリガを与える前に、まず ADC を有効にします。

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 ±64LSB 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℃) 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。
0℃時のエラーレートは 24M 回に 1 回、55℃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。

BSL_ERR_01 *BSL モジュール*

カテゴリ

機能

機能

アプリケーション ソフトウェアから BSL を呼び出すと、特定の条件下で失敗する可能性があります

説明

アプリケーション内からの呼び出し (BSL ソフトウェア呼び出し) によって BSL を起動しようとすると、SRAM エラー コードが原因で起動に失敗することがあります。リセット後、エラーの影響により BSL は起動されず、デバイスはアプリケーションに戻ってしまいます。

このエラッタは、ハードウェア起動メソッドによる BSL 起動には適用されません

回避方法

BSL をソフトウェアから呼び出す必要があるアプリケーションでは、デバイスをリセットする前に、アセンブリ コードを使用して SRAM 全体をクリアしてください。MSPM0 SDK バージョン 1.20.01.xx 以上に含まれる MSPM0 の「bsl_software_invoke」サンプルには、「bsl_software_invoke」サンプル内で修正例が含まれています。

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。

CLK_ERR_01 *CLK モジュール*

カテゴリ

機能

機能

デバッガ接続時に、HFXT 4MHz を MCLK として使用すると、ハードフォルトが発生することがあります

説明

MCLK が HFXT 4MHz に設定され、以下の構成が行われている場合: HFXTRSEL を 0 (4 ～ 8 MHz の水晶発振子に推奨される値) に設定している場合、プログラムがランダムにハードフォルトまたは NMI にジャンプすることがあります。

CLK_ERR_01 (続き) *CLK モジュール*

回避方法

HFXT 4MHz を MCLK として使用してコードをデバッグする場合は、HFXTRSEL の値を 1 以上 (8MHz ～ 48MHz の範囲) に設定します。

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 のプリフェッチ無効化に関する制限

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

プリフェッチを無効にした後、SYSCTL 内のシャットダウン メモリ (SHUTDNSTORE) に対してメモリ アクセスを実行してください。これは、SYSCTL.SOCLOCK.SHUTDNSTORE0 で行えます。メモリ アクセスが完了すると、プリフェッチャが無効化されます。
例：

```
CPUSS.CTL.PREFETCH = 0x0; // プリフェッチャを無効化
SYSCTL.SOCLOCK.SHUTDNSTORE0; // シャットダウン メモリへのアクセス
```

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
SYSCTL.SOCLOCK.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

DMA_ERR_01 *DMA モジュール*

カテゴリ

機能

機能

DMA または CPU が、クロック ドメインをまたいで周辺レジスタに同時アクセスした場合、動作が失われる可能性があります

説明

DMA および CPU は MCLK をソースとしています。MCLK が ULPCLK よりも高い周波数で動作している場合、DMA または CPU が、ULPCLK をソースとする周辺レジスタ (PD0 に属するすべてのペリフェラルを含む) に同時にアクセスした場合、動作が失われる可能性があります。アクセスされるペリフェラルまたはレジスタを同じにする必要はありません。注:ADC は PD0 に属するペリフェラルですが、DMA または CPU が ADC の SVT_MEMRES や SVT_FIFODATA にアクセスする場合には制限はありません。例:DMA が UART0 のレジスタにアクセスしていると仮定します。たとえば、DMA が TXDATA にデータを書き込む場合です。このとき、DMA 処理中に CPU が PD0 のペリフェラルにアクセスした場合 (例: CPU が TIMG0 の CTR からデータを読み込むなど)、DMA または CPU のどちらかがデータを失う可能性があります。

回避方法

MCLK が ULPCLK よりも高い周波数で動作している場合、CPU と DMA は PD0 ペリフェラルに同時にアクセスしないでください。

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

FLASH_ERR_02
(続き)

***FLASH* モジュール**

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_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

概要

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

FLASH_ERR_05
(続き)

***FLASH* モジュール**

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 はフラッシュメ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。0x40BFFFFF ～ 0x41C00000 3。0x41C007FF～ 0x41C40000

回避方法

なし

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

GPIO_ERR_01 (続き)

GPIO モジュール

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 ウェークアップ 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 ウェークアップ パルスがアクティブでないことを確認します

GPIO_ERR_03

GPIO および DEBUGSS 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「高速ウェークのみ」(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。

GPIO_ERR_04 (続き)

GPIO モジュール

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「高速ウェークのみ」機能は使用しないでください。

I2C_ERR_01

I2C モジュール

カテゴリ

機能

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ことがあります。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)

I2C_ERR_03 (続き) ***I2C モジュール***

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 になり、ターゲットのウェイクアップが有効になっている場合、デバイスが無期限にクロック ストレッチを行う可能性があります。
説明	デバイスがターゲットモードにあり、クロック ストレッチや外部によるグランド接続により SCL が Low に保たれている場合、コントローラがバスを解放しても、I2C ターゲットはクロック ストレッチを解除できない状態になることがあります。
回避方法	ターゲット ウェークアップ イネーブル ビット (SWUEN) をディセーブルにします。

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 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

I2C_ERR_06 (続き) ***I2C モジュール***

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。

概要

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

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。

概要

標準的な I2C 速度（100kHz 未満）で動作している場合、割り込みを通過する読み出しの時間内に ADDRMATCH ビット（TSR レジスタのアドレス一致）が設定されないおそれがあります。

I2C_ERR_09 (続き) ***I2C モジュール***

回避方法

非標準的な I2C 速度で動作する場合、ADDRMATCH ビットを読み取る前に、少なくとも 1 つの I2C クロックサイクル分の遅延を入れてください。

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 ビットのポーリングでは、コントローラ転送の完了を確実に保証できない場合がある

説明

CCTR.BURSTRUN ビットを設定して I2C コントローラ転送を開始した後、BUSY ステータスがアサートされるまでに、約 3 クロック分の I2C 機能クロックサイクルを要します。転送完了を待つように CCTR.BURSTRUN を設定した直後に BUSY ビットをポーリングすると、BUSY ステータスが設定される前にチェックされることがあります。この問題は、CLKDIV の値が大きい (その結果、I2C 機能クロックが遅くなる) 場合やコンパイラ最適化レベルが高い場合に、発生する可能性がより高くなります。

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ます。ソフトウェア遅延 = 3 x CPU CLK / I2C の機能クロック = 3 x CPU CLK / (CLKSEL / CLKDIV)。たとえば、クロック分周器 (CLKDIV) が 8MHz、クロックソース (MFCLK) が 4MHz、CPU CLK が 32MHz の場合、ソフトウェア遅延 = 3 x 32 MHz / (4 MHz/ 8) = 192 CPU サイクル。

MATHACL_ERR_01 ***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

MATHACL_ERR_01 (続き)

MATHACL モジュール

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_02

MATHACL モジュール

カテゴリ

機能

機能

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

概要

COS(-180)または 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_06

PMCU 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

説明

CPU と DMA はフラッシュメに同時にアクセスすることができません。たとえば、フラッシュの消去操作中に同時アクセスが発生すると、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。この問題は、DMA アクセスやプログラム/ 消去操作、読み出し検証/ ブランク検証など、CPU 以外の要因によって HREADY が CPU に対して Low に引き下げられている間に発生す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。プログラム/ 消去操作や、読み出し検証/ ブランク検証を行う場合、ソフトウェアは 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。これは、フラッシュ動作中にコードを SRAM に入れることで指定できます。

PMCU_ERR_08

PMCU モジュール

カテゴリ

機能

機能

デバイスが LPM へ移行中にトリガが与えられると、想定よりもウェイクアップ時間が長くなることがあります

PMCU_ERR_08 (続き)

PMCU モジュール

説明

デバイスが低電力モードへ移行中にウェイクアップ信号が与えられると、約 3us の追加ウェイクアップ時間が発生します。

回避方法

回避方法はありません。

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 などのそ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。

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えることがあります。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 (WWDT)、AES、TRNG。

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

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

RST_ERR_01 (続き) ***RST モジュール***

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 パルス幅が 608µs 未満のときに見られます。NRST パルスが 608µs 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、608µs 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 のとき合、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_01 ***SPI モジュール***

カテゴリ

機能

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_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

SPI_ERR_02 (続き) *SPI モジュール*

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_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。

SRAM_ERR_02 *SRAM モジュール*

カテゴリ

機能

機能

SRAM エラー アドレスは、最上位バイトが欠落した値を返します

説明

SRAM DEDERRADDR は誤った値を返し、最上位バイトは切り捨てられます。たとえば、0x20001234 のパリティ エラーは、パリティ エラー アドレス 0x00001234 を返します。

回避方法

SRAM DED または SRAM パリティ フォルトが発生した場合は、SRAM のオリジンと SYSCTRL->DEDERRADDR の戻り値に対してビットごとの OR 演算を行ってください。SRAM メモリ開始位置については、デバイス データシートを参照してください。リンカ ファイルも SRAM アドレスの送信元を保持しています。

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

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

なし

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 はクリアされません

説明

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

回避方法

なし

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

説明

MFCLK が有効になっており、SYSOSC が周波数補正ループ (FCL) モードを使用しており、STOP1 の低電力動作モードが使用されている場合、SYSOSC が 4MHz から 32MHz に切り替わる際 (STOP1 モードから RUN モードへの移行時、または SYSOSC を 32MHz に強制する非同期の高速クロック要求時)、MFCLK が 2 サイクル分ずれる可能性があります。

回避方法

Workaround1：STOP1 モードではなく STOP0 モードを使用します。STOP0 モードを使用する場合、MFCLK ドリフトは発生しません。Workaround2：STOP1 を使用する場合、SYSOSC を FCL モードで使用しないでください(FCL はディセーブルのままにします)。

SYSOSC_ERR_02 *SYSOSC モジュール*

カテゴリ

機能

SYSOSC_ERR_02 (続き)

SYSOSC モジュール

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

概要

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、ulpclk 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 SYSPLLEN 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

SYSPLLEN 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(SYSPLLEN = 0)および再有効化(SYSPLLEN = 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 ±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、SYSPLLEN 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

SYSPLL_ERR_01
(続き)

SYSPLL モジュール

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 ±5% を見込むことが推奨されます。FCC の推定実行時間は 1µs 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、SYSPLLEN 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_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。

TIMER_ERR_04 (続き)

***TIMER* モジュール**

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

***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

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

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

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。

UART_ERR_01 (続き)

UART モジュール

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 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。

UART_ERR_05 (続き)

UART モジュール

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。
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、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 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.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

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。このエラッタの結果として、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。

VREF_ERR_02 *VREF モジュール*

カテゴリ

機能

機能

VREF を 2.5V モードから 1.4V モードに切り替える際のスルーレートが非常に低くなります

説明

VREF の設定を 2.5V モードから 1.4V モードに変更する際、スルーレートが非常に遅くなります。

VREF_ERR_02 (続き)

VREF モジュール

回避方法

VREF モードを切り替えるには、次の手順を実行します。1. 構成を 1.4V モードに変更する前に、CTL0 レジスタ内の ENABLE ビットを使用して VREF を無効にします。2.ピン PA23 (VREF+) を GPIO 出力として構成し、このピンを論理 Low に 100µs 間駆動します。VREF+ ピンには 1µF の外付けコンデンサが接続されていることを前提としています。3. CTL0 レジスタの BUGCONFIG ビットを 1 に設定することで、VREF を 1.4V モードで再度有効にします。この手順では、2.5V モードから 1.4V モードに切り替えるために必要な時間は 200us です。

WWDT_ERR_01

WWDT モジュール

カテゴリ

機能

機能

ウォッチドッグ タイマ 1 (WWDT1) のイベントは、常に SYSRST を実行します。

説明

WWDT1 のイベントは、SYSTEMCFG.WWDTLP1RSTDIS ビットの設定に関係なく、常に SYSRST を実行します。したがって、WWDT1 は「NMI のみ」イベントをトリガできません。

回避方法

NMI の目的で WWDT0 を使用します。

WWDT_ERR_02

WWDT モジュール

カテゴリ

機能

機能

ウィンドウ ウォッチドッグ タイマ 1 (WWDT1) では、リセット要因が発生しません

説明

WWD1 によるリセットの後、SYSCTL.RSTCAUSE レジスタは、リセット原因が WWDT1 であることを正確に示しません。

回避方法

なし。

商標

すべての商標は、それぞれの所有者に帰属します。

7 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 18, 2025 to NOVEMBER 30, 2025 (from Revision C (July 2025) to Revision D (November 2025)) **Page**

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むいかなる保証も、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。

最終更新日：2025 年 10 月