参考資料

Errata

MSPM0C1103、*MSPM0C1104*、*MSPM0C1103-Q1*、*MSPM0C1104-Q1*、*MSPS003F3*、*MSPS003F4* 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、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 B、C、D
ADC_ERR_05	✓
ADC_ERR_09	✓
CLK_ERR_02	✓
CPU_ERR_01	✓
CPU_ERR_02	✓
CPU_ERR_03	✓
CPU_ERR_04	✓
CRC/CRCP_ERR_01	✓
FCC_ERR_01	✓
FLASH_ERR_03	✓
FLASH_ERR_05	✓
FLASH_ERR_06	✓
FLASH_ERR_08	✓
GPIO_ERR_03	✓
GPIO_ERR_04	✓
GPIO_ERR_09	✓
I2C_ERR_03	✓
I2C_ERR_04	✓

エラッタ番号	Rev B、C、D
I2C_ERR_05	✓
I2C_ERR_06	✓
I2C_ERR_07	✓
I2C_ERR_08	✓
I2C_ERR_09	✓
I2C_ERR_10	✓
I2C_ERR_13	✓
I2C_ERR_15	✓
RST_ERR_01	✓
SPI_ERR_03	✓
SPI_ERR_04	✓
SPI_ERR_05	✓
SPI_ERR_06	✓
SPI_ERR_07	✓
SPI_ERR_10	✓
SYSCTL_ERR_03	✓
SYSCTL_ERR_05	✓
SYSCTL_ERR_06	✓
SYSCTL_ERR_07	✓
SYSOSC_ERR_02	✓
SYSOSC_ERR_04	✓
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	✓

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、C、D
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。」

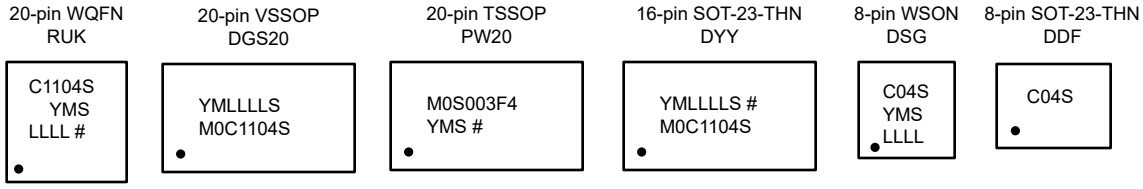
MSP デバイスの特性は完全に明確化されており、デバイスの品質と信頼性が十分に示されています。テキサス・インスツルメンツの標準保証が適用されます。

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。



YM = Year, month date code
S = Assembly site
# = Die revision
LLLL = Assembly lot code

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 (パッケージマーキング)	バージョン (デバイスの工場出荷時定数メモリ内)
B	0x1
C	0x2
D	0x3

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_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 サンプリ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 サンプリングを開始します。

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_09

ADC モジュール

カテゴリ

機能

機能

ADC のオフセット誤差は、アプリケーション コードで補正する必要があります

説明

ADC オフセット誤差の補正データが正しく適用されないため、アプリケーション コードで実装する必要があります。

回避方法

補正データは、ファクトリ領域のアドレス 0x41C40040 に格納されます。この操作を容易にするため、2 つの DriverLib API (***DL_ADC12_getADCOffsetCalibration*** と ***DL_FactoryRegion_getADCOffset***) が SDK に実装されています。

```
__STATIC_INLINE int16_t DL_ADC12_getADCOffsetCalibration(float userRef)
{
float adcBuff = DL_FactoryRegion_getADCOffset() * (3.3 / userRef);
return (int16_t)(round(adcBuff));
}

__STATIC_INLINE float DL_FactoryRegion_getADCOffset(void)
{
return ((float) (*(int16_t *) (0x41C40040)));
}
```

補正データを変数に保存し、その後、ADC 変換結果に適用できます。
以下のサンプル コードは、SDK に付属の ADC サンプルに含まれている補正データの適用方法

ADC_ERR_09 (続き) *ADC モジュール*

を示しています。

```
volatile uint16_t gAdcResult;
volatile int16_t gADCOffset;
gADCOffset =
DL_ADC12_getADCOffsetCalibration(ADC12_0_ADCMEM_0_REF_VOLTAGE_V);
gAdcResult = DL_ADC12_getMemResult(ADC12_0_INST, DL_ADC12_MEM_IDX_0);
int16_t adcRaw = (int16_t) gAdcResult + gADCOffset;
if (adcRaw < 0)
{
adcRaw = 0;
}
if (adcRaw > 4095)
{
adcRaw = 4095;
}
gAdcResult = (uint16_t) adcRaw;
```

ADC オフセット補正データは、以下の使用事例に適用できます。
ADC のオフセット誤差は、アプリケーション コードで補正する必要があります

- DMA なしで ADC を動作させる場合 - MEMRES レジスタから取得した ADC の結果にオフセットを加算する必要があります。
- DMA を使用して ADC を動作させる場合 — メモリ アドレスに格納されている ADC の結果にオフセットを加算する必要があります。
- ADC ウィンドウ コンパレータを使用する場合 — コンパレータ機能の設定時に使用したウィンドウ スレッショルドからオフセットを減算する必要があります。

CLK_ERR_02 *CLK モジュール*

カテゴリ

機能

機能

HFCLK_IN が HFCLK および HSCLK のソースである場合、MFCLK が誤って BUSCLK から供給される

説明

HSCLK が HFCLK からのみ供給されており、HFCLK_IN が HFCLK のソースとして構成されている場合、MFCLK は 4MHz ではなく BUSCLK に誤って設定されます。

回避方法

回避方法はありません。

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

CPU_ERR_01 (続き) *CPU モジュール*

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:
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

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。

CPU_ERR_03 (続き) *CPU モジュール*

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 &= 0x6; // プリフェッチャーを無効化、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; // プリフェッチャーを有効化

CPU_ERR_04 *CPU モジュール*

カテゴリ

機能

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と、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、別のウェークアップ機能 (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_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_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

FLASH_ERR_06
(続き)

フラッシュ モジュール

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

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

GPIO_ERR_04 (続き)

GPIO モジュール

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。

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。

GPIO_ERR_09

GPIO モジュール

カテゴリ

機能

機能

GPIO DOUT レジスタへの DMA 書き込みは、GPIO DMAMASK レジスタの構成に関係なく行われます

説明

DMA を使用して GPIO DOUT レジスタに書き込む場合、GPIO DMAMASK の設定は DMA 経由でのレジスタ書き込みに影響を与えません。そのため、GPIO DMAMASK 機能は使用できません。

回避方法

DMA を使用して GPIO DOUT レジスタに書き込まれた内容が、更新されるピンと一致していることを確認します。GPIO DMAMASK レジスタは、マスキングに利用しないでください。

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

I2C_ERR_04 (続き) ***I2C モジュール***

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

説明

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、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。状況によっては、SDA が 0 でコントローラが IDLE 状態になることがありますが、ここでは、コントローラが IDLE 状態から移行したり、SDA 値を更新したりできません。ターゲットの BUSBUSY がセットされ (ACTIVE ビットのトグルにより、ライン上で開始が検出される)、コントローラが停止を駆動してクリアできないため、BUSBUSY はクリアされません。

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。I2C START 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。これにより、タイムアウト イベントをトリガし、転送開始時にトランザクションを

I2C_ERR_06 (続き) ***I2C モジュール***

完了することなく I2C コントローラを 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 I2C START ビットの書き込みが開始します。その後、START 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 (30µs) 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 High 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

I2C_ERR_09 (続き) *I2C モジュール*

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

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 x CPU CLK / I2C 機能クロック = 3 x CPU CLK / (CLKSEL / 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 x 32 MHz / (4 MHz / 8) = 192 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 4µs 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。

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_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。

SPI_ERR_03 (続き) ***SPI モジュール***

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

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

SPI_ERR_06 (続き) ***SPI モジュール***

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_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に 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ョルド 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。

SYSCTL_ERR_03 ***SYSCTL モジュール***

カテゴリ

機能

SYSCTL_ERR_03 (続き) ***SYSCTL 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。

SYSCTL_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

SYSCTL_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。

SYSCTL_ERR_07 ***SYSCTL モジュール***

カテゴリ

機能

SYSCTL_ERR_07
(続き)

SYSCTL モジュール

機能

FCL が有効な状態でデバイスが RUN2/SLEEP2 モードになると、予期しない BOR リセット サイクルが発生します

説明

FCL が有効な状態で、デバイスが RUN2/SLEEP2 モードで動作している場合、BOR1/2/3 で予期しない BOR リセット サイクル (5ms) 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 が有効な状態でデバイスを RUN2 または SLEEP2 モードにしないでください。

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、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。

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 ±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。

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

TIMER_ERR_01 (続き)

TIMx モジュール

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 *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。

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。

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。

UART_ERR_09 (続き)

UART モジュール

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

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 NOVEMBER 30, 2025 to JULY 30, 2026 (from Revision C (November 2025) to Revision D (July 2026)) **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