

Errata

MSPM0G511x、MSPM0G5187 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.....	3
3 デバッグ専用のアドバイザリ.....	3
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 デバイスの記号表記とリビジョンの識別.....	3
6 アドバイザリの説明.....	5
7 商標.....	33
8 改訂履歴.....	33

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 (プロトタイプ X マーク付き製品)	Rev C
AES_ERR_01	✓	✓
BSL_ERR_01	✓	✓
BSL_ERR_02	✓	
COMP_ERR_06	✓	✓
CPU_ERR_02	✓	✓
CPU_ERR_03	✓	✓
FCC_ERR_01	✓	✓
FLASH_ERR_03	✓	✓
FLASH_ERR_04	✓	✓
FLASH_ERR_05	✓	✓
FLASH_ERR_08	✓	✓
FLASH_ERR_09	✓	✓
GPIO_ERR_05	✓	✓
GPIO_ERR_06	✓	✓
I2S_ERR_01	✓	✓
KEYSTORE_ERR_01	✓	✓
PMCU_ERR_14	✓	✓
PMCU_ERR_16	✓	✓
RST_ERR_01	✓	✓

エラッタ番号	Rev B (プロトタイプ X マーク付き製品)	Rev C
RST_ERR_02	✓	✓
SPI_ERR_10	✓	✓
SYSCTL_ERR_01	✓	✓
SYSCTL_ERR_02	✓	✓
SYSCTL_ERR_03	✓	✓
SYSCTL_ERR_04	✓	✓
SYSCTL_ERR_05	✓	✓
SYSCTL_ERR_06	✓	✓
SYSCTL_ERR_11	✓	✓
SYSCTL_ERR_12	✓	✓
SYSOSC_ERR_01	✓	✓
SYSOSC_ERR_02	✓	✓
SYSOSC_ERR_04	✓	✓
SYSOSC_ERR_05	✓	✓
SYSPLL_ERR_01	✓	
TIMER_ERR_04	✓	✓
TIMER_ERR_06	✓	✓
TIMER_ERR_07	✓	✓
UNICOMMI2CC_ERR_01	✓	✓
UNICOMMI2CC_ERR_02	✓	✓
UNICOMMI2CC_ERR_03	✓	✓
UNICOMMI2CT_ERR_01	✓	✓
UNICOMMI2CT_ERR_02	✓	✓
UNICOMMI2CT_ERR_03	✓	✓
UNICOMMSPI_ERR_01	✓	✓
UNICOMMSPI_ERR_02	✓	✓
UNICOMMSPI_ERR_03	✓	✓
UNICOMMSPI_ERR_04	✓	✓
UNICOMMUART_ERR_01	✓	✓
UNICOMMUART_ERR_02	✓	✓
UNICOMMUART_ERR_03	✓	✓
UNICOMMUART_ERR_04	✓	✓
UNICOMMUART_ERR_05	✓	✓
UNICOMMUART_ERR_06	✓	✓
UNICOMMUART_ERR_07	✓	✓
UNICOMMUART_ERR_08	✓	✓
UNICOMMUART_ERR_09	✓	✓
UNICOMMUART_ERR_10	✓	✓
UNICOMMUART_ERR_11	✓	✓
UNICOMMUART_ERR_12	✓	✓
UNICOMMUART_ERR_13	✓	✓
UNICOMMUART_ERR_14	✓	✓
UNICOMMUART_ERR_15	✓	✓
USB_ERR_01	✓	✓
USB_ERR_02	✓	✓

エラッタ番号	Rev B (プロトタイプ X マーク付き製品)	Rev C
USB_ERR_03	✓	
USB_ERR_04	✓	✓
VREF_ERR_04	✓	✓

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、指定されたり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、指定されたりビジョンに問題が存在することを示します。

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、指定されたり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。TI 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 デバイスの命名規則には、デバイス ファミリ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の記号表記の方法を示し、表 5-1 はデバイス 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。

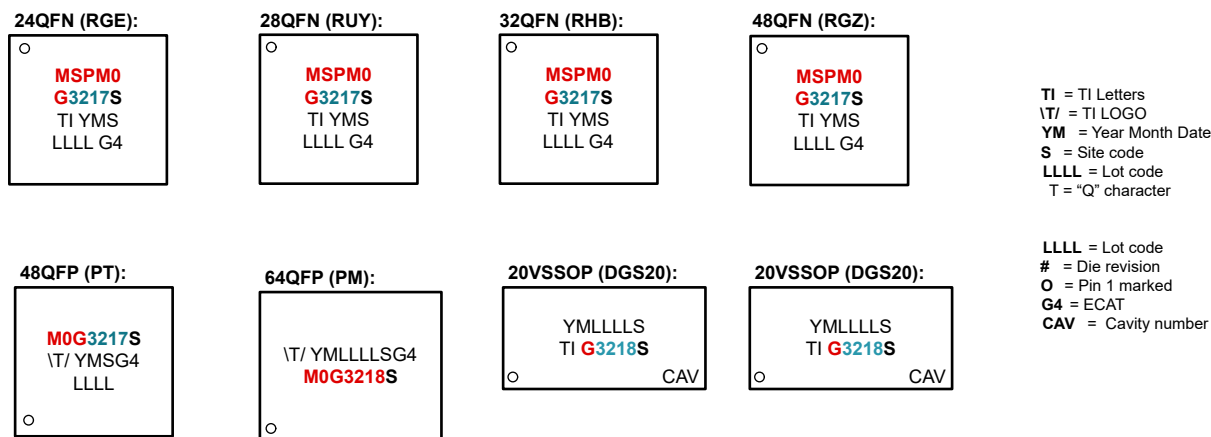


図 5-1. パッケージの記号表記

表 5-1. ダイリビジョン

リビジョンレター	製品開発ステータス	DEVICEID.VERSION
A	プロトタイプ (X マーク付き)	0
B	RTM	0

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、デバイスのファクトリー定数メモリ (DEVICEID.VERSION フィールド) に格納されている整数値に対応しており、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を使用して、この値からリビジョンを確認できます。

注

DEVICEID.VERSION フィールドは、RTM (Release to Market、市場投入) 後の製品ハードウェアが改版されるたびに値が増加します。同じ VERSION 値内では、シルクスクリーン上の「X」の文字によって、特定のバリエーションを識別できます。

6 アドバイザリの説明

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ステディのステータスビットを確認します。
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。
BSL_ERR_02	BSL モジュール
カテゴリ	機能
機能	DFU モードでは、メモリの読み戻しが常に可能になってしまう
説明	DFU (USB BSL) モードが起動されると、BSL が正しいパスワードでロック解除されていない場合や、NONMAIN 構成で読み出し機能が無効になっていても、メモリの読み戻しコマンドは常に通過します。メモリ範囲チェックもバイパスされ、ペリフェラル、MAIN FLASH、NONMAIN FLASH、SRAM メモリなどのすべてのメモリを読み取ることができます。

BSL_ERR_02 (続き) **BSL** モジュール

回避方法

セカンダリ BSL を使用してセキュリティ機能を上書きします。

COMP_ERR_06 **COMP** モジュール

カテゴリ

機能

機能

出力の準備ができる前に **SYSOSC** がディセーブルになっている場合、**COMP** 出力の準備はできないか、割り込みが生成されます

説明

COMP をイネーブルにするか、その入力構成 (**IPSEL/IMSEL**) を変更すると、**COMP** 出力の準備ができた時点で **OUTRDYIFG** ビットがセットされるようになります。**OUTRDYIFG** は、出力の準備ができるまで、すべての **COMP** 割り込みの起動を阻止します。

COMP OUTRDYIFG ビットでは、**SYSOSC** が 4MHz 以上で設定される必要があります。**SYSOSC** がディセーブルの場合、**OUTRDYIFG** はセットされず、すべての **COMP** 割り込みの起動を永久にブロックします。

回避方法

COMP を想定どおりに動作させるには、**OUTRDYIFG** ビットがセットされるまで **SYSOSC** をイネーブルにする必要があります。**COMP** をイネーブルにするか、**IPSEL/IMSEL** を構成するとき、**SYSOSC** をディセーブルする **LPM** に入力する前に、またはクロックを **LFCLK** に変更する前に、**OUTRDYIFG** ビットが設定されるのを待つ必要があります。

1. コンパレータをイネーブルにする
2. 必要に応じて **IPSEL/IMSEL** を設定する
3. **OUTRDYIFG** 割り込みを待つ
4. 必要に応じて、低消費電力モード (**LPM**) に移行するか、クロックを **LFCLK** に変更します

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ャを無効化します
```

CPU_ERR_02 (続き) CPU モジュール

DL_SYSCCTL_setShutdownStorageByte(DL_SYSCCTL_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。

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; // プリフェッチャを無効化、その他の設定は維持
SYSCCTL.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
```

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

FLASH_ERR_03

(続き)

FLASH モジュール

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

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。

SYSCTL_DEDERRADDR の復帰アドレスが **0x00Dxxxxx** 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_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

FLASH_ERR_09 *FLASH* モジュール

カテゴリ

機能

機能

特定のフラッシュ スワップ ポリシー下では、bank1 の消去時に、BANK1 の静的書き込み保護は機能しません

説明

デバイスのフラッシュ メモリが最大容量のデバイスでない場合 (例:MSPM0L111x ファミリーには、MSPM0L1117 と MSPM0L1116 として 2 つの異なるデバイスがありますが、MSPM0L1116 にのみこの問題があります)、以下の条件で bank1 が予期せず消去されるという問題があります: 失敗ケース 1: BANK1 の静的書き込み保護が有効、フラッシュ スワップ ポリシーが無効、SECCFG_Reg.FLBANKSWP レジスタの USEUPPER ビットが 0 または 1 の場合、バンク消去コマンドで BANK1 が予期せず消去されます。失敗ケース 2: BANK1 の静的書き込み保護が有効、フラッシュ スワップ ポリシーが有効、SECCFG_Reg.FLBANKSWP レジスタの USEUPPER ビットが 0 の場合、バンク消去コマンドで BANK1 が予期せず消去されます。

回避方法

1.Bank1 にはバンク消去の代わりにセクタ消去を使用します。または 2.同じデバイス ファミリー内で最大容量のフラッシュ デバイスを使用します。

GPIO_ERR_05 *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 DOUTTGL 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

同時 DMA 転送が進行中の間に、DOUTTGL レジスタへの CPU 書き込みに GPIO DMAMASK レジスタ情報が誤って適用されます。

回避方法

アプリケーション コードで、DOUTTGL レジスタへの CPU 書き込みアクセスが発行される前に、DOUTTGL レジスタの対応するビットに対して GPIO DMAMASK ビットが 1 に設定されていることを確認します。GPIO レジスタへの DMA 転送が不要な場合は、IO 初期化ステップ中に GPIO

GPIO_ERR_05 (続き)

GPIO モジュール

DMAMASK を 0xFFFFFFFF として構成できます。これにより、このエラッタの競合が解決されます。アプリケーションで GPIO レジスタへの DMA 書き込み転送も必要な場合は、アプリケーションで DMA と CPU の両方を使用して、デバイス内の同じ GPIO モジュールの DOUTTGL レジスタに書き込むことを推奨します。デバイスに複数の GPIO モジュールがある場合、DMA と CPU は、異なる GPIO モジュールの DOUTTGL レジスタに同時に書き込むことができます (一方で、CPU が書き込み先の GPIO モジュール用に GPIO DMAMASK を構成する必要もあります)。

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;
```

GPIO_ERR_06 (続き)

GPIO モジュール

```
doutVal &= ~pinsMask;
doutVal |= (pinsVal & pinsMask);
gpio->DOU31GL31_0 = ~(gpio->DOU31_0) & doutVal;
gpio->DOU31GL31_0 = gpio->DOU31_0 & (~doutVal);
}
```

I2S_ERR_01

I2S モジュール

カテゴリ

機能

機能

I2S/TDM モジュールの有効化後、最初の TXIFG 割り込みが発生しない

説明

I2S/TDM モジュールが有効化された際、最初の TXIFG 割り込みは発生しません。

回避方法

INTCTL.INTERVAL ビットに 1 を書き込むことで、最初の TXIFG 割り込みを生成してください。

KEYSTORE_ERR_01

キーストアモジュール

カテゴリ

機能

機能

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

説明

STATUS.STAT 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、レジスタウィンドウを介したデバッガアクセスは 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.STAT を読み取ると、0x00 が返されます。

回避方法

STATUS.STAT=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。

PMCU_ERR_14

PMCU モジュール

カテゴリ

機能

機能

GPIO によってトリガされた DMA 転送が、FCL をイネーブルにした低消費電力モードでウェイクアップするまで保留される

説明

FCL がイネーブルで低消費電力モード (STOP2/STANDBY0/STANDBY1) の場合、GPIO からのイベント チャネルによって DMA がトリガされます。イベントまたは割り込みによってデバイスが

PMCU_ERR_14 (続き)

PMCU モジュール

ウェークアップされるまで、この DMA 送信は開始されません。ウェークアップする前に追加の消費電力が確認されます。

回避方法

GPIO イベントによってトリガされる DMA を使用する代わりに、ソフトウェアで DMA 転送を開始するには GPIO 割り込みを使用します。割り込み終了後、再度低消費電力モードに移行します。

PMCU_ERR_16

PMCU モジュール

カテゴリ

機能

機能

電源電圧が 3.6V の時に、シャットダウン電流が仕様値を超えてしまう

説明

VDD 電源電圧が 3.6V の場合、特定のコーナー ケースでは、シャットダウン時の電流がデータシートの仕様値を超えてしまうことがあります。特に、 -40°C では、シャットダウン電流 I_{VDD} が 200nA を超える場合があります、USB シャットダウン電流 I_{VUSB} が 2 μA を超える場合があります。

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

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 250 μs (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。

RST_ERR_02

RST モジュール

カテゴリ

機能

RST_ERR_02 (続き) RST モジュール

機能

NRST デアサートが最初の LFOSCGOOD アサートと一致した場合、デバイスは起動に失敗する可能性があります

説明

起動イベント中、内部コア電圧 (VCORE) が安定したレベルに達すると、内部低周波数発振器 (LFOSC) が有効になります。LFOSCGOOD 信号は、VCORE が安定状態に達してから 250µs (最小) から 1.5ms (最大) の範囲内で High にアサートされます。VDD の VBOR+ スレッシュホールドは、VCORE が安定したときのリファレンス ポイントとして使用できます。NRST のデアサート (Low から High への遷移) が 200ns ウィンドウ内に LFOSCGOOD の立ち上がりエッジと一致した場合、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、スタートアップシーケンスのブートフェーズを終了できません。NRST デアサートタイミングは、VDD の立ち上がり速度と、NRST 回路上の外部抵抗およびコンデンサ (RC) ネットワークの両方によって制御されるため、実際にはこの状態をトリガできます。この現象が発生しやすいことが知られている構成例として、NRST 回路上に 47k の抵抗と 10nF のコンデンサが含まれている場合が挙げられます。

回避方法

回避方法 1: RC フィルタには十分小さなプルアップ抵抗を使用して、LFOSCGOOD の最小アサート時間 (250µs) よりも少なくとも 50µs 前に、NRST 信号が VDD の 90% まで確実に立ち上がるようにします。これにより、NRST デアサートエッジが LFOSCGOOD のアサートウィンドウから十分に離れるようにします。必要なプルアップの最小抵抗値は、外部リセットコントローラ (使用する場合) の IO 駆動強度によって決まることに注意してください。準拠する構成として確認されているのは、NRST 回路上に 4.7kΩ の抵抗と 10nF のコンデンサを配置した構成です。
Workaround2: 外部リセットコントローラを採用し、VDD 立ち上がりの開始から少なくとも 2.0ms 間、NRST をロジック Low レベルに保持します。これにより、LFOSCGOOD の最大アサート時間 (1.5ms) 後に NRST デアサートが発生することが保証されます。

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 チャン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 チャン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_01 SYSCCTL モジュール

カテゴリ

機能

SYSCTL_ERR_01

(続き)

SYSCTL モジュール

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

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 はクリアされません

SYSCTL_ERR_04

(続き)

SYSCTL モジュール

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

SYSRESET から復帰後、SYSSTATUSCLR = 0xCE000001 を用いて FLASHSEC ステータスを手動でクリアしてください。

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 をディセーブルにすると、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セ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_11 SYSCTL モジュール

カテゴリ

機能

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

SYSCTL_ERR_11

(続き)

SYSCTL モジュール**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。

SYSCTL_ERR_12**SYSCTL モジュール****カテゴリ**

機能

機能

FCL が有効な状態でデバイスが RUN2/SLEEP2 モードの場合、BOR1/2/3 発生後も BOR イベントがマスクされたままとなる

説明

RUN2 または SLEEP2 電源モードで FCL が有効になっている場合、BOR マスクロジックが想定どおりに動作せず、最初の BOR 発生後に続いて発生するすべての BOR イベントを検出できなくなる。ソフトウェアによって SYSOSC が再度有効化されるまで、BOR MASK はクリアされる。

回避方法

回避方法 1: BOR1/2/3 検出が必要な場合は、FCL を有効にせず、デバイスを RUN2/SLEEP2 モードにします。回避策 2: 初期マスキング期間が経過した後、BOR/NMI ハンドラで BOR マスクフラグを手動でクリアする。回避策 3: Sysosc の有効/無効を定期的に切り替えて BOR マスクフラグをクリアすることで、ソフトウェアベースの更新メカニズムを実装しま

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 モードでは、SYSOSC の精度が低下します

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、システム内のノイズとの間の同期に起因します。

回避方法

SYSPLL FCL ON モードで使用する場合は、次のように SYSPLL 周波数に 4MHz の倍数以外の値を使用します。例：78MHz または 79MHz

SYSPLL を 16、32、48、40、64、80MHz などにししないでください。

78MHz の場合：

SYSPLLCFG1.PDIV = 0x3 と SYSPLLCFG1.QDIV を 38 に設定します。

4MHz の倍数でない SYSPLL 動作周波数であっても、FCL サンプリング クロックに対して少なくとも 1 マイクロ秒に 1 回は同期します。2 つのクロックが共通係数を共有する場合は、さらに高い頻度で同期します。

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イクアップ中、SYSOSC はベース周波数より低く動作する場合があります

SYSOSC_ERR_05

(続き)

SYSOSC モジュール**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。

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**周波数確認プロセス**

SYSPLLEN ビットを 1 に設定した場合は常に、周波数クロック カウンタ (FCC) を使用して SYSPLL の出力周波数を監視します。正しい周波数が確立されると、SYSPLL がディスエーブルになり、再度イネーブルになるまで、その周波数は安定した状態を維持します (SYSPLLEN ビットを 0 から 1 にトグル)。誤った周波数が検出された場合は、SYSPLL をディスエーブルにしてから再度イネーブルにして、別の検証を実行します。

回避方法 1:FCC カウントチェック

SYSPLL 出力クロック周波数をカウントするには、FCC トリガ ソースとして LFCLK を使用します。FCC を実行し、LFCLK をリファレンスとして使用して、設定された SYSPLL 周波数に対して測定値を検証します。

計算例:

- SYSPLLCLK0 = 80MHz ; LFCLK = 32.768kHz

- 測定 FCC カウント = $80,000,000 / 32,768 = 2,441$

FCC カウント許容誤差:

実際の FCC 数は、合計クロック精度 (SYSPLLCLK0 および LFCLK) によって異なります。FCC

SYSPLL_ERR_01

(続き)

SYSPLL モジュール

チェック範囲を許可するには、 $\pm 5 \sim 10\%$ を追加することを推奨します。

- FCC カウント上限 = $2,441 \times 1.05 = 2,563$

- FCC カウント下限 = $2,441 \times 0.95 = 2,318$

タイミング考慮事項:

- クロック同期時間: $5 \sim 6$ LFCLK サイクル

- FCCトリガ時間: $1 \sim 32$ LFCLK サイクル (ユーザー構成可能)

レジスタ構成:

- FCC 設定: `SYSCTL.GENCLKCFG.FCCTRIGSRC = 1;`

- `SYSCTL.GENCLKCFG.FCCLVLTRIG = 0;`

- `SYSCTL.GENCLKCFG.FCCTRIGCNT = 0;`

- `SYSCTL.GENCLKCFG.FCCSELCLK = 4;`

- FCC を開始: `SYSCTL.FCCCMD = 0x0E000001U`

- FCC 完了ステータスのチェック: `SYSCTL.CLKSTATUS.FCCDONE`

- FCC カウントの読み取り: `SYSCTL.FCC`

タイムアウト保護:

FCC DONE ステータス監視中にソフトウェアベースのタイムアウトを実装し、ロックされていない SYSPLL 周波数が FCC トリガ クロック周波数 (LFCLK) よりも低い状態での待ち時間を長くしないようにします。

```
fccTimeoutCounter = 0;
```

```
while (DL_SYSCTL_isFCCDone() == 0) {
```

```
    delay_cycles(977); /* 1x LFCLK cycle = 32MHz/32.768kHz */
```

```
    fccTimeoutCounter++;
```

```
    if(fccTimeoutCounter > 65) break;
```

```
    /* Timeout set to approximately 2ms (ユーザーによるカスタマイズ可能) */
```

```
}
```

FCC チェック再起動:

FCC 測定値が予想される範囲を超えている場合は、SYSPLL をディスエーブルにしてから再度イネーブルにし (SYSPLLEN を 0 に設定してから 1 に設定)、FCC 検証を繰り返します。

回避方法 2: FCC 比率チェック FCC

トリガ ソースとして LFCLK を使用し、SYSPLL 出力と入力クロック周波数の両方をカウントします。FCC を実行し、出力クロックと入力クロックとの間の FCC チェック値の測定された比率が予想される比率であることを確認します。

計算例:

- SYSPLL = 80MHz ; HFCLK = 40MHz ; LFCLK = 32.768kHz

- Expect clock ratio = $80\text{MHz}/40\text{MHz} = 2.0000$

- Measured FCC count (SYSPLL) = $80,000,000/32,768 = 2,441$

- Measured FCC count (HFCLK) = $40,000,000/32,768 = 1,220$

- 測定クロック比 = $2,441/1,220 = 2.0008$

FCC 比率許容誤差:

FCC 比率方法を使用すると、結合クロック精度誤差を除去し、FCC の不確定性 (2 カウントクロック サイクル) および計算の丸め誤差のみに依存します。これにより、FCC カウント チェック方式に比べて、 $\pm 0.3\%$ などの許容範囲をはるかに狭くすることができます。

タイミングに関する考慮事項:

- クロック同期時間: $5 \sim 6$ LFCLK サイクル

- FCCトリガ時間: $1 \sim 32$ LFCLK サイクル (ユーザー構成可能)

- 完全な FCC 比チェックあたりの合計時間: $2 \times (\text{同期時間} + \text{トリガ時間})$

FCC 比チェックフロー:

SYSPLL_ERR_01

(続き)

SYSPLL モジュール

- 1.SYSPLL 出力クロック用に FCC を構成します (SYSPLL0 または SYSPLL2X)
- 2.FCC を開始し、FCC DONE を待機します (タイムアウト保護を追加。「FCC カウント チェック」を参照)
- 3.FCC チェック カウントを読み戻します
- 4.SYSPLL 入力クロックの FCC を構成します (SYSOSC または HFCLK)
- 5.FCC を開始し、FCC DONE を待機します (タイムアウト保護を追加。「FCC カウント チェック」を参照)
- 6.FCC チェック カウントを読み戻します
- 7.FCC チェック比率を計算し、予測比率範囲と比較します
- 8.FCC 比が予想される範囲を超えている場合は、SYSPLL をディスエーブルにしてから再度イネーブルにし (SYSPLLEN を 0 に設定してから 1 に設定)、FCC 比検証を繰り返します。

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タ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。

UNICOMMI2CC_E RR_01

UNICOMMI2CC モジュール

カテゴリ

機能

機能

I2C コントローラの BUSY ステータス ポーリングの問題

説明

BUSRTRUN/FRAME_START ビットを設定して I2C コントローラ転送を開始するとき、BUSY ステータス フラグがアサートするのに約 2 ~ 3 I2C 機能クロック サイクルが必要です。BUSRTRUN/FRAME_START を設定した直後に BUSY ビットをポーリングする場合、アプリケーションが適切に設定する前にステータスをチェックすることがあります。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より顕著になります。

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。遅延の推奨値は次のとおりです。ソフトウェア遅延 = $3 \times \text{I2C 機能クロック} = 3 \times \text{clock_divider} \times (\text{CPU_CLK} / \text{選択したクロックソース周波数})$ 。たとえば、clock_divider が 2、クロックソースが 4MHz(MFCLK)、CPU_CLK が 80MHz の場合: ソフトウェア遅延 = $3 \times 2 \times (80\text{MHz} / 4\text{MHz}) = 120 \text{ CPU サイクル}$

UNICOMMI2CC_E RR_02

UNICOMMI2CC モジュール

カテゴリ

機能

機能

I2C コントローラの ACKOEN ビットが有効になっている場合、RXDONE 割り込みが 2 回発生します

UNICOMMI2CC_E RR_02 (続き)

UNICOMMI2CC モジュール

説明

I2C コントローラの CTR.ACKONE ビットが有効になっている場合、ソフトウェアは CTR.ACK ビットに 0 または 1 を書き込むことで、ACK/NACK の送信を制御します。これは、ストップ状態になる前の RXDONE 割り込み内で実行できます。CTR.ACK ビットを書き込んだ後、I2C バス上に有効な ACK/NACK 信号が出現し、その後の I2C ストップ状態で追加の RXDONE 割り込みが発生します。

回避方法

CTR.ACK ビットを書き込む前に RXDONE 割り込みを無効にし、次の送信の前に手動で有効にしてください。

UNICOMMI2CC_E RR_03

UNICOMMI2CC モジュール

カテゴリ

機能

機能

I2C コントローラーのクロックストレッチ機能を有効にすると、SCL 周波数が低下する

説明

— 回避策として ETW 機能の使用を推奨しないでください。— 本 ERRATA の提案によって、本デバイスの当該リビジョンまたは以前のリビジョンにおける既存の他の ERRATA に影響が生じないことを確認してください。クロックストレッチ機能を有効 (CR.CLKSTRETCH = 1) にすると、SCL 周波数は設定値よりも低くなります。たとえば、I2C コントローラーのバス速度を 400kHz に設定した場合、クロックストレッチ機能を有効にすると、I2C コントローラーのバスクロックは 400kHz よりも低くなり、360kHz や 390kHz などになります。同様の現象は、100kHz や 1MHz など、他のバス速度でも発生します。

回避方法

I2C 通信機能への影響は小さいため、クロックストレッチ機能を無効にすることは推奨しません。クロックストレッチ機能を無効にすると、I2C バスクロックは期待される速度になります。

UNICOMMI2CT_ER R_01

UNICOMMI2CT モジュール

カテゴリ

機能

機能

CTR が別々に書き込まれると、I2C ターゲットはアイドル状態を保持します

説明

I2C ターゲットは、次のシナリオでアイドル状態を維持します: 1. I2C は MFCLK をクロックソースとして使用します。2. CLKDIV = 0。3. 最初のレジスタ書き込みで、CTR.START、CTR.STOP、または CTR.BLEN ビットを設定します。4. 2 回目のレジスタ書き込みで、CTR.FRM_START を設定します。

回避方法

I2C がクロックソースとして MFCLK を使用し、かつ CLKDIV = 0 の場合は、CTR.START、CTR.STOP、CTR.BLEN、または CTR.FRM_START の各ビットを 1 回のレジスタ書き込みで設定することを推奨します。

UNICOMMI2CT_ER R_02

UNICOMMI2CT モジュール

カテゴリ

機能

機能

RXFIFO データ整合性の問題

説明

UNICOMMI2CT の RX モードでは、受信フレームの最後のビットが受信された直後に RXDONE 割り込みフラグがアサートされます。ただし、RXFIFO バッファの内容更新は、RXDONE のアサートに対して、さらに 1 機能クロック (I2C 機能クロック) のレイテンシを伴います。CPU がこの期間より前に RXFIFO の読み取り動作を開始すると、古い FIFO データにアクセスし、データ整合性の問題を引き起こす可能性があります。この問題は、UNICOMMI2CT 機能クロックが CPU クロックよりもはるかに遅い場合に特に問題となります。

回避方法

データの可用性を確保するため、RXFIFO を読み取る前に固定の遅延を追加します。遅延の推奨値は次のとおりです。ソフトウェア遅延 = 1 I2C 機能クロック = 1 clock_divider (CPU_CLK / 選択したクロック ソース周波数) たとえば、clock_divider = 2、クロック ソース = 4MHz (MFCLK)、CPU_CLK = 80MHz の場合:ソフトウェア遅延 = 1 2 (80MHz/4MHz) = 40 CPU サイクル

UNICOMMI2CT_ER R_03

UNICOMMI2CT モジュール

カテゴリ

機能

機能

TSTART フラグが SR.ADDRMATCH の更新より前に更新されてしまう

説明

UNICOMM_I2CT では、ターゲット START ステータス ビットの割り込みステータス ビット (RIS.TSTART) が、アドレス一致ステータス (SR.ADDRMATCH) ビットよりも 2 ~ 3 I2C 機能クロック サイクル前に設定されています。この間に CPU がアクセスすると、古い値が読み取られてしまう可能性があります。この問題は、UNICOMMI2CT 機能クロックが CPU クロックよりもはるかに遅い場合に特に問題となります。

回避方法

ADDRMATCH を読み取る前に、固定の遅延を追加します。読み取り前の推奨ソフトウェア遅延:ソフトウェア遅延 = 3 I2C 機能クロック = 3 clock_divider (CPU_CLK / 選択したクロック ソース周波数) たとえば、clock_divider = 2、クロック ソース = 4MHz (MFCLK)、CPU_CLK = 80MHz の場合:ソフトウェア遅延 = 3 2 (80MHz/4MHz) = 120 CPU サイクル

UNICOMMSPI_ER R_01

UNICOMMSPI モジュール

カテゴリ

機能

機能

TXFIFO への読み書きが同時に行われた場合、SPI のアンダーフロー イベントが生成されない可能性がある

UNICOMMSPI_ER R_01 (続き)

UNICOMMSPI モジュール

説明

SPI ペリフェラル モードで動作している場合、ステータス フラグが不正確になる可能性のある競合状態が発生することがあります。この問題は、以下の特定の条件下で発生します。1.送信 FIFO が空になる 2.SPI ペリフェラルがデータ送信要求を受信する (通常は外部コントローラから) 3.CPU/DMA が送信バッファに新しいデータを書き込もうとする。4.この CPU/DMA による書き込み動作が、コントローラからの新しい送信要求と同時に発生する。このようなタイミングに依存する状況では、ステータス フラグ (FIFO が空 / Underflow フラグなど) が実際の状態と異なる値を示す可能性があります。これは、データ送信エラーや割り込みの取りこぼしにつながる可能性があります。この問題により、以下が発生する可能性があります: 不適切なステータス フラグ読み出し データ送信エラーの可能性割り込み生成の不安定化

回避方法

この問題を軽減するには: 1.CPU/DMA と SPI ペリフェラルの間のデータ レートを注意深く管理します 2.重要なアプリケーションでは、SPI 送信バッファへの DMA 転送に高い優先度を設定することを検討します。

UNICOMMSPI_ER R_02

UNICOMMSPI モジュール

カテゴリ

機能

機能

コントローラ側のビジー状態が正しく読み取られない

説明

SPI のビジー状態は TXFIFO のデータ要素に依存しないため、基本的に SPI TXFIFO にデータをロードしても、CLKSEL が低速クロック (LFCLK など) または CLKDIV > 1 の場合、ビジービットはすぐにハイになりません。SPI ソースクロックが低速であるため、TXFIFO へのデータロード後から実際に通信が開始されるまでの間、ステータスが誤って読み取られることになります。

回避方法

アプリケーションコードを作成する際に、TXFIFO の STATUS ビットが空かどうかをチェックする機能を追加できます。

UNICOMMSPI_ER R_03

UNICOMMSPI モジュール

カテゴリ

機能

機能

CPU を使用して RXFIFO をクリアすると、2 回目に RTOUT がセットされない

説明

rxtimeout を有効にして SPI 通信を完了し、その後 RTOUT がセットされた状態で、CPU を使用して RXFIFO のデータを読み出し、IIDX を読み出して割り込みをクリアした場合、IP をリセットまたは無効化せずに、同じ送信を 2 回目に行うと、rxtimeout カウンタが「0」であり、RXFIFO が空でないにもかかわらず、RTOUT 割り込みがセットされない。

UNICOMMSPI_ER

R_03 (続き)

UNICOMMSPI モジュール

回避方法

CPU で直接 RXFIFO のデータを読み出す代わりに、DMA を使用して RXFIFO のデータを読み出してください。または、CPU で直接 RXFIFO のデータを読み出した後、UNICOMM SPI をリセットしてください。

UNICOMMSPI_ER

R_04

UNICOMMSPI モジュール

カテゴリ

機能

機能

DSS の不一致が発生した状態で前回転送後に再転送を行うと、RX に保存されたデータが正しくありません

説明

コントローラ側とペリフェラル側でデータパケットサイズ設定 (DSS) が異なる状態でデータを送信した後、RXCLR を実行して FIFO をクリアし、その後、正しいデータパケットサイズで再度データを送信しても、最初の SCLK パルスが発生した際に、以前のデータが RXFIFO に格納されてしまう

回避方法

コントローラ側とペリフェラル側のデータパケットサイズ設定 (DSS) が同じであることを確認してください。

UNICOMMUART_E

RR_01

UNICOMMUART モジュール

カテゴリ

機能

機能

IrDA モードでグリッチが発生した場合のデータ完全性の問題

説明

UNICOMM-UART は IrDA をサポートしていますが、ノイズの多い環境では制限があります。グリッチ フィルタがないため、データラインにグリッチが発生すると、IrDA モードではデータ整合性の問題が発生する可能性があります。IrDA はエッジ検出ロジックに依存しているため、これらのグリッチが有効なパルスとして誤認識され、その結果 FIFO に予期しないデータが入り、LTOUT の計算に問題が生じます。

回避方法

なし

UNICOMMUART_E

RR_02

UNICOMMUART モジュール

カテゴリ

機能

機能

IDLELINE モードにおけるグリッチによる IDLE 期間検出の問題

UNICOMMUART_E**RR_02 (続き)****UNICOMMUART モジュール****説明**

アイドル期間中にグリッチが発生すると、受信側のアイドル検出が乱され、その結果、後続のアドレス バイトを Rx 側が破棄してしまう可能性があります。重要なのは、この故障モードがタイミング依存である点です。アイドル期間に対して特定のタイミングでグリッチが挿入された場合にのみ、この問題が発生します。それで、アイドル期間内のすべてのグリッチが破損を引き起こすわけではありません。

回避方法

なし

UNICOMMUART_E**RR_03****UNICOMMUART モジュール****カテゴリ**

機能

機能

マンチェスター モードでのグリッチによるデータ完全性の問題

説明

マンチェスター符号化されたデータ ストリーム中のグリッチにより、信号の正確なエッジ タイミングが乱され、誤った受信データやデータ整合性の問題が発生します。

回避方法

なし

UNICOMMUART_E**RR_04****UNICOMMUART モジュール****カテゴリ**

機能

機能

LIN モードでは、ブレーク フィールドおよび / または同期フィールド中にグリッチが発生すると、データが欠落します

説明

ブレーク フィールドのシナリオ (ハイパルス グリッチ) : ブレーク フィールド期間中に負のエッジが発生すると、カウンタがゼロにリセットされ、ブレーク フィールドの検出に失敗し、その結果データが失われます。同期フィールドのシナリオ: 同期フィールド中にグリッチが発生すると、誤った LINC0/LINC1 割り込みがトリガされ、Rx ラインの負のエッジでカウンタがリセットされ、同期フィールドの検証に失敗し、その結果データが失われます。

回避方法

なし

UNICOMMUART_E**RR_05****UNICOMMUART モジュール****カテゴリ**

機能

機能

UART グリッチ フィルタが存在しないことに起因する問題

UNICOMMUART_E

RR_05 (続き)

UNICOMMUART モジュール

説明

1. IRDA モードのデータ整合性: グリッチ フィルタがないため、データラインにグリッチが発生すると、IrDA モードではデータ整合性の問題が発生する可能性があります。IrDA はエッジ検出ロジックに依存しているため、これらのグリッチが有効なパルスとして誤認識されてしまいます。

2. Ideline 検出: IDLELINE 検出中、信号のグリッチが発生すると、内部のアイドルライン カウンタがゼロにリセットされ、検出プロセスが妨げられる可能性があります。この干渉により、IDLELINE ステータス フラグが正しく設定されなくなり、モジュールが通信ライン上の有効なアイドル状態を認識できなくなります。ただし、この妨害後も追加のグリッチが発生せずにアイドルライン状態が持続する場合、検出メカニズムは最終的に回復し、アイドル状態を正しく検出します。

3. Manchester モードのデータ整合性: Manchester 符号化された伝送のデータ サンプリング期間中に信号グリッチが発生すると、受信機は誤ったデータを取得し、ビットの誤解釈につながります。これらのサンプリング エラーにより、破損したデータ値が受信バッファに格納され、受信メッセージのデータ整合性に障害が生じます。サンプリング ウィンドウ外で発生するグリッチは、通常、データの解釈に影響を与えません。

4. BREAK/SYNC フィールド周辺における LIN モードの感度: a. BREAK フィールド: LIN 通信中、特に BREAK FIELD のクロック サンプリング期間中に信号のグリッチが発生した場合、検出メカニズムが早期に終了する可能性があります。しかし、システムには回復力があり、障害発生後も追加のグリッチが発生せずに BREAK フィールドの状態が継続する場合、検出ロジックは回復し、BREAK フィールドを正常に識別します。b. SYNC フィールド: SYNC フィールドのキャリブレーション中、システムは同様の脆弱性を示し、グリッチが負エッジ (RXNE) および正エッジ (RXPE) の検出の両方に干渉する可能性があります。

5. グリッチに伴う LTOUT/RTOUT 検出の問題: アイドルライン検出中に信号グリッチが発生すると、LTOUT/RTOUT のタイミング計算が一時的に中断され、タイムアウト カウンタが 0 にリセットされます。中断後もラインに追加のグリッチが発生しなければ、設定された時間が経過した後に LTOUT/RTOUT 状態が検出されます。

回避方法

回避策はありません。

UNICOMMUART_E

RR_06

UNICOMMUART モジュール

カテゴリ

機能

機能

ストップ ビット処理が原因で、RTOUT/LTOUT 計算に問題が発生

説明

レシーバ側では、機能ステート マシンがストップ ビットの途中でストップ ビットから IDLE に遷移します。これにより、受信タイムアウト (RTOUT) およびライン タイムアウト (LTOUT) カウンタは、終了ではなく、ストップ ビットの中央でカウントを開始します。RTOUT/LTOUT は、ボー周期の半分前にトリガされることになります。これは、UART 機能クロック周波数が高いボーレートの場合に特に顕著です。

回避方法

RTOUT カウンタに、ストップ ビット半周期分の補正を追加します。

UNICOMMUART_E**RR_07****UNICOMMUART** モジュール**カテゴリ**

機能

機能

RS-232 モードでは、UART が無効化されている場合、RTS ラインは HIGH になりません

説明

UART が無効化されている場合、RTS ラインはアイドル状態 (HIGH) に戻らず、LOW のまま固定されます。

回避方法

ソフトウェアを使用して内部プルアップ抵抗を有効にし、RTS ラインの IO を Hi-Z モードに設定します。

UNICOMMUART_E**RR_08****UNICOMMUART** モジュール**カテゴリ**

機能

機能

LTOUT 割り込みが継続的に発生する

説明

ライン タイムアウト (LTOUT) 割り込みが発生すると、カウンタが再起動してカウントを続行し、タイムアウト時間が経過するたびに LTOUT 割り込みが繰り返し発生します。この問題は、RTOUT イベントについても同様に発生します。

回避方法

LTOUT ISR で LTOUT 検出を無効化します。新しい文字を受信した時点で、再度有効化します。

UNICOMMUART_E**RR_09****UNICOMMUART** モジュール**カテゴリ**

機能

機能

ISO-7816 スマートカード モードのボーレート制限

説明

ISO-7816 スマートカード モードで 9600 ボーレートを実現するには、以下の制約があるので、57MHz を超える UARTCLK 周波数が必要です。1.ISO-7816 規格では、ビット 2 あたり 372 クロック サイクルが必要です。MSPM0 ISO-7816 モードでは、オーバーサンプリング レート (OVS) は UART ペリフェラルの最大 UARTCLK 計算で 16x に固定されています。必要な $UARTCLK = 9600 * 372 * 16 = 57.139MHz$

回避方法

入力周波数が低い UART は、スマートカード モードをサポートできません。

UNICOMMUART_E

RR_10

UNICOMMUART モジュール

カテゴリ

機能

機能

LIN レジスタの CLKDIV 制限

説明

CLKDIV 値が 0 以外の場合、LINC0/1 に書き込んでも効果はありません。

回避方法

LIN レジスタを適切に構成するには: 1. まず、CLKDIV を '0' に設定します。 2. 目的の値を使用して、LINC0/1 レジスタ構成を更新します 3. CLKDIV を目的の動作値に復元します

UNICOMMUART_E

RR_11

UNICOMMUART モジュール

カテゴリ

機能

機能

UART の無効化後も BUSY ビットが設定されたままになる

説明

UNICOMMUART が無効になっていても、FIFO にデータが残っている場合、STAT.BUSY ビットは High (アサート状態) のままになります。

回避方法

UNICOMMUART を無効化する前に、IFLS.TXCLR ビットを設定して TXFIFO の内容をクリアします。

UNICOMMUART_E

RR_12

UNICOMMUART モジュール

カテゴリ

機能

機能

フレームの途中で CTL0.TXE ビットを 0 に設定すると、現在の送信でエラーが発生する

説明

パケットの送信を開始し、転送の途中で CTL0.TXE ビットを 0 に設定すると、現在のパケット送信が完了せず、TX ラインは Low にプルされます。これにより、BUSY (ビジー状態) が示されます。TX ラインは、CTL0.TXE = 1 になった場合にのみ解放され、それまでは BUSY が 1 のままになります。

回避方法

この問題を回避するには、BUSY ステータスが Low になり、かつ CTL0.TXE = 0 になるまでポーリングする必要があります。

UNICOMMUART_E

RR_13

UNICOMMUART モジュール

カテゴリ

機能

UNICOMMUART_E**RR_13 (続き) UNICOMMUART モジュール**

機能

RTOUT 割り込みが 1fclk サイクル早く発生する

説明

RTOUT (受信タイムアウト) 割り込みが、本来発生すべきタイミングより 1fclk サイクル早く発生します。

回避方法

RTOUT 割り込みが 1fclk サイクル早く発生すると想定することで、この問題に対処できます。

UNICOMMUART_E**RR_14 UNICOMMUART モジュール**

カテゴリ

機能

機能

CTS ラインの立ち下がりエッジと立ち上がりエッジの両方で CTS 割り込みが切り替わる

説明

CTS ラインの立ち下がりエッジと立ち上がりエッジの両方で CTS 割り込みが切り替わります。本来は、CTS ラインの立ち下がりエッジのみで切り替わるべきです。

回避方法

RIS ビットの代わりに、STAT.CTS ビットが 1 に切り替わることを監視してください。この STAT.CTS = 1 の切り替わりが、CTS ラインの立ち下がりが発生したことを示すためです。

UNICOMMUART_E**RR_15 UNICOMMUART モジュール**

カテゴリ

機能

機能

fclk 間での LCRH.SENDIDLE アクセスの取りこぼしおよび、新しい IDLELINE フレームが生成されない

説明

2 つの fclk エッジの間で、LCRH.SENDIDLE を 0 から 1 に切り替えてから 1 から 0 に戻す、または 1 から 0 に切り替えてから 0 から 1 に戻す場合、その切り替えが取りこぼされ、新しい IDLELINE フレームは送信されません。

回避方法

LCRH.SENDIDLE を切り替えた後、すぐには元に戻さず、1fclk より長くその状態を維持してください。LCRH.SENDIDLE は fclk でサンプリングされるためです。

USB_ERR_01**USB モジュール**

カテゴリ

機能

機能

CLKEN ビットを FLEN ビットより先にクリアすると、USBFLI が無効にならない

USB_ERR_01 (続き) **USB モジュール**

説明

CLKEN ビットをクリアした際に FLLEN ビットが設定されたままになっていると、USBFLL クロックはロック状態を維持し、想定どおりには無効化されません。

回避方法

USBFLL クロックを適切に無効化するには、CLKEN ビットをクリアする前に FLLEN ビットをクリアします。

USB_ERR_02 **USB モジュール**

カテゴリ

機能

機能

IMASK、RIS、MIS レジスタで、DMADONE ビットと DMAPREIRQ ビットが入れ替わっている

説明

IMASK、RIS、MIS レジスタにはビット割り当ての誤りがあり、DMADONE と DMAPRERD のビット位置が誤って逆になっています。 - DMAPREIRQ ビット (19:12) は、対応する DMADONE 割り込みステータスを示します (本来の機能とは異なります) - DMADONE ビット (11:4) は、対応する DMA プリ割り込みステータスを示します (本来の機能とは異なります) このビットの入れ替わりは、IIDX レジスタの読み出し値にも影響します。例:- BIT4 は本来 DMADONEARX を示すためのビットですが、実際には DMAPREARX (BIT12) の状態を示します - BIT5 は本来 DMADONEATX を示すためのビットですが、実際には DMAPREATX (BIT13) の状態を示します - BIT11 は本来 DMADONEDTX を示すためのビットですが、実際には DMAPREDTX (BIT19) の状態を示します

回避方法

回避方法はありません。

USB_ERR_03 **USB モジュール**

カテゴリ

機能

機能

SOF を FLL 基準クロックとして設定すると、USB FLL 周波数ロック時間が長くなる

説明

USBFLL の基準クロックとして SOF (Start of Frame) を使用する場合、USBFLL が正しい周波数にロックするまでに通常より長い時間を要するケースが存在します。テスト結果によると、ロック時間が最大約 30ms になる可能性があります。

回避方法

1.USB FLL の基準クロック ソースとして、SOF の代わりに LXFT を使用します (USBFLLCTL.REFSEL を 1h に設定) 2.USB USB クロック ソースとして SYSPLL を選択します (GENCLKCFG.USBCLKSRC を 1h に設定)

USB_ERR_04 **USB モジュール**

カテゴリ

機能

USB_ERR_04 (続き) USB モジュール

機能

複数のエンドポイントが構成されている場合、エンドポイント割り込みが取りこぼされる可能性がある

説明

構成されたエンドポイントでデータが受信または送信されると、対応する EP 割り込みが INTRTX または INTRRX レジスタに記録されます。これらのレジスタで新しいイベントが発生すると、CPU に対して割り込みイベントがトリガされます。ISR ハンドラは、INTRTX と INTRRX を読み出して割り込みの発生元を特定し、割り込みレジスタをクリアする必要があります。割り込みレジスタの自動クリア動作には、CPU クロック サイクルが必要です。複数のエンドポイントが構成されたシステムでは、自動クリア プロセスがまだ進行中の間に、新しい EP 割り込みが発生することがあります。その結果、INTRTX または INTRRX レジスタがアサートされた状態のままとなり、新たな割り込みイベントが発生したことが CPU に認識されない可能性があります。

回避方法

INTRTX または INTRRX レジスタを読み出すときは、INTRTX または INTRTX を数回読み出して、割り込みレジスタが実際にクリアされていることを確認してください。これにより、新たに発生した割り込みが新しい割り込みイベントとして確実に認識され、CPU に適切に通知されるようになります。

以下はコード実装例です。

```
uint_fast8_t intr = 0; /* 割り込みステータス */
uint_fast8_t intr_reg = 0; /* 一時的な割り込みレジスタ */
intr_reg = musb_regs->intr_tx; /* intr_tx の読み出しでレジスタはクリアされるが、その間に新たな割り込みが発生する可能性がある */
(intr_reg) { /* intr_tx がクリアされるまで複数回読み出す */
    intr |= intr_reg; intr_reg = musb_regs->intr_tx;
}
intr &= musb_regs->intr_txen; /* 無効化されている割り込みを除外 */
/* EP0 を処理 */
if (intr & TU_BIT(0)) {
    process_ep0(rhport); intr &= ~TU_BIT(0);
}
/* EPn を処理 */
while (intr) {
    unsigned const num = __builtin_ctz(intr);
    process_edpt_n(rhport, tu_edpt_addr(num, TUSB_DIR_IN));
    intr &= ~TU_BIT(num);
}
```

VREF_ERR_04

VREF モジュール

カテゴリ

機能

機能

コンパレータがサンプリング (超低消費電力) モードの場合、ADC では VREF を使用できない

説明

以下の 2 つの使用事例では、コンパレータ (COMP) がサンプリング (超低消費電力) モードの場合、ADC は VREF を使用できません。

VREF_ERR_04 (続き)

VREF モジュール

ケース 1:

構成: ADC は内部 VREF を基準電圧として使用するよう構成されており、一方、COMP は VDDA または外部基準電圧を使用し、サンプル / ホールド モードが有効になっています
問題: COMP から VDDA / 外部基準電圧の要求がトリガされると、内部 VREF モジュールがサンプル / ホールド モードに入ってしまう、VREF を ADC で使用できなくなります。

ケース 2:

構成: ADC と COMP はどちらも、内部 VREF をリファレンスとして使用するよう構成されており、COMP はサンプリング モードで動作します。
問題: VREF モジュールは、COMP のサンプル / ホールド要求によってサンプル / ホールド モードに移行し、VREF への ADC アクセスを防止します。

回避方法

ADC サンプリングを有効化する前に、COMP がサンプリング (超低消費電力) モードに構成されていないことを確認してください。

例:

COMP.CTL2.REFMODE = 0;

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 November 30, 2025 to September 31, 2026 (from Revision * (November 2025) to Revision A (September 2026))

	Page
• 初版リリース.....	1
• BSL_ERR_01 アドバイザリを追加しました.....	5
• BSL_ERR_02 アドバイザリを追加しました.....	5
• COMP_ERR_06 アドバイザリを追加しました.....	6
• CPU_ERR_02 機能を更新しました.....	6
• CPU_ERR_02 の説明を更新しました.....	6
• CPU_ERR_02 回避策を更新しました.....	6
• CPU_ERR_03 回避策を更新しました.....	7
• FCC_ERR_01 アドバイザリを追加しました.....	7
• FLASH_ERR_09 アドバイザリを追加しました.....	9
• GPIO_ERR_06 の説明を更新しました.....	10
• GPIO_ERR_06 回避策を更新しました.....	10
• I2S_ERR_01 アドバイザリを追加しました.....	11
• PMCU_ERR_14 アドバイザリを追加しました.....	11
• PMCU_ERR_16 アドバイザリを追加しました.....	12
• RST_ERR_01 回避策を更新しました.....	12
• RST_ERR_02 アドバイザリを追加しました.....	12
• SPI_ERR_10 アドバイザリを追加しました.....	13
• SYSCTL_ERR_04 回避策を更新しました.....	14
• SYSCTL_ERR_05 アドバイザリを追加しました.....	15
• SYSCTL_ERR_06 アドバイザリを追加しました.....	15

• SYSCTL_ERR_11 アドバイザリを追加しました.....	15
• SYSCTL_ERR_12 アドバイザリを追加しました.....	16
• SYSOSC_ERR_04 アドバイザリを追加しました.....	17
• SYSOSC_ERR_05 アドバイザリを追加しました.....	17
• SYSPLL_ERR_01 回避策を更新しました.....	18
• TIMER_ERR_06 モジュールを更新しました.....	20
• UNICOMMI2CC_ERR_01 機能を更新しました.....	21
• UNICOMMI2CC_ERR_01 の説明を更新しました.....	21
• UNICOMMI2CC_ERR_01 回避策を更新しました.....	21
• UNICOMMI2CC_ERR_02 アドバイザリを追加しました.....	21
• UNICOMMI2CC_ERR_03 アドバイザリを追加しました.....	22
• UNICOMMI2CT_ERR_02 機能を更新しました.....	23
• UNICOMMI2CT_ERR_02 の説明を更新しました.....	23
• UNICOMMI2CT_ERR_02 回避策を更新しました.....	23
• UNICOMMI2CT_ERR_03 機能を更新しました.....	23
• UNICOMMI2CT_ERR_03 の説明を更新しました.....	23
• UNICOMMI2CT_ERR_03 回避策を更新しました.....	23
• UNICOMMSPI_ERR_01 の説明を更新しました.....	23
• UNICOMMSPI_ERR_01 回避策を更新しました.....	23
• UNICOMMSPI_ERR_02 アドバイザリを追加しました.....	24
• UNICOMMSPI_ERR_03 アドバイザリを追加しました.....	24
• UNICOMMSPI_ERR_04 アドバイザリを追加しました.....	25
• UNICOMMUART_ERR_05 機能を更新しました.....	26
• UNICOMMUART_ERR_05 回避策を更新しました.....	26
• UNICOMMUART_ERR_05 の説明を更新しました.....	26
• UNICOMMUART_ERR_06 機能を更新しました.....	27
• UNICOMMUART_ERR_06 の説明を更新しました.....	27
• UNICOMMUART_ERR_06 回避策を更新しました.....	27
• UNICOMMUART_ERR_08 機能を更新しました.....	28
• UNICOMMUART_ERR_08 の説明を更新しました.....	28
• UNICOMMUART_ERR_08 回避策を更新しました.....	28
• UNICOMMUART_ERR_09 機能を更新しました.....	28
• UNICOMMUART_ERR_09 の説明を更新しました.....	28
• UNICOMMUART_ERR_09 回避策を更新しました.....	28
• UNICOMMUART_ERR_10 機能を更新しました.....	29
• UNICOMMUART_ERR_10 の説明を更新しました.....	29
• UNICOMMUART_ERR_10 回避策を更新しました.....	29
• UNICOMMUART_ERR_11 機能を更新しました.....	29
• UNICOMMUART_ERR_11 の説明を更新しました.....	29
• UNICOMMUART_ERR_11 回避策を更新しました.....	29
• UNICOMMUART_ERR_12 アドバイザリを追加しました.....	29
• UNICOMMUART_ERR_13 アドバイザリを追加しました.....	29
• UNICOMMUART_ERR_14 アドバイザリを追加しました.....	30
• UNICOMMUART_ERR_15 アドバイザリを追加しました.....	30
• USB_ERR_01 アドバイザリを追加しました.....	30
• USB_ERR_02 アドバイザリを追加しました.....	31
• USB_ERR_03 アドバイザリを追加しました.....	31
• USB_ERR_04 アドバイザリを追加しました.....	31
• VREF_ERR_04 アドバイザリを追加しました.....	32

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