

Errata

MSPM33C32xx、MSPM33C32xx-Q1 マイコン



概要

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

目次

1 機能アドバイザリ.....	1
2 プログラム済みのソフトウェア アドバイザリ.....	2
3 デバッグ専用のアドバイザリ.....	2
4 コンパイラ アドバイザリによって修正.....	2
5 デバイスの命名規則.....	2
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.....	4
7 商標.....	18
8 改訂履歴.....	18

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	リビジョン A
AES_ERR_01	✓
CPU_ERR_05	✓
FLASH_ERR_02	✓
GPIO_ERR_05	✓
GSC_ERR_01	✓
IOMUX_ERR_01	✓
I2S_ERR_01	✓
KEYSTORE_ERR_01	✓
PMCU_ERR_15	✓
RST_ERR_01	✓
RST_ERR_02	✓
SYSCTL_ERR_01	✓
SYSPLL_ERR_01	✓
TIMER_ERR_04	✓
TIMER_ERR_06	✓
TIMER_ERR_07	✓
TIMER_ERR_08	✓
UNICOMMI2CC_ERR_01	✓
UNICOMMI2CC_ERR_02	✓
UNICOMMI2CC_ERR_03	✓
UNICOMMSPI_ERR_02	✓

エラー番号	リビジョン A
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_10	✓
UNICOMMUART_ERR_12	✓
UNICOMMUART_ERR_13	✓
UNICOMMUART_ERR_14	✓
UNICOMMUART_ERR_15	✓

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。

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、製品開発の進捗段階を表します。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同時に表 5-1、デバイスリビジョンからバージョン ID へのマッピングを定義しています。

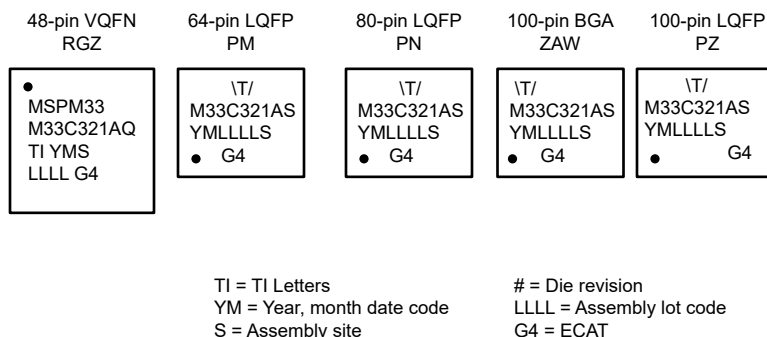


図 5-1. パッケージの記号表記

表 5-1. ダイ リビジョン

リビジョンレター	バージョン(デバイスの工場出荷時定数メモリ内)
A	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ーには、リビジョン文字に基づいて、特定のバースに該当するか否かがマークされています。この文字は、デバイスのメモリに保存された整数にマップされ、アプリケーションソフトウェア または接続されたデバッグプローブによるリビジョンの検索に使用できます。

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。
CPU_ERR_05	CPU モジュール
カテゴリ	機能
機能	MTB_BASE レジスタ値が正しく読み取られません
説明	MTB_BASE レジスタを読み取る際に、MTB_SRAM の基点が 1 ビット右にシフトされます。たとえば、MTB_SRAM が 0x80806000 の場合、実際の MTB_SRAM の基点は 0x40403000 です。
回避方法	MTB_BASE レジスタを読み取り、MTB_SRAM の基点を確認する場合は、読み取り値を 1 ビット右にシフトします。この問題を回避するには、次のコードを使用してください。 <pre>uint32_t mtb_base = (*(uint32_t*)(0xE004300C) >> 1);</pre>
FLASH_ERR_02	FLASH モジュール
カテゴリ	機能
機能	NONMAIN でのデバッグの無効は、パスワードを使用して再度有効にできる
説明	デバッグが NONMAIN 設定 (DEBUGACCESS = 0x5566) により無効化されている場合でも、プログラムされたパスワードを使用してデバイスにアクセスできる可能性があります (パスワードが明示的に設定されていない場合はデフォルト値が使用される)。
回避方法	回避方法 1: DEBUGACCESS を Debug Enabled with Password オプション (DEBUGACCESS = 0xCCDD) に設定し、PWDDEBUGLOCK フィールドに一意のパスワードを入力します。より高度のセキュリティを確保するために、暗号化されたランダムなデバイス固有のパスワードを使用することをお勧めします。これにより、適切な 128 ビットのパスワードでデバッグアクセスが可能になりますが、一部のデバッグコマンドで、CFG-AP と SEC-AP にアクセスすることもできます。

FLASH_ERR_02

(続き)

FLASH モジュール

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。

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 DMAMASK を 0xFFFFFFFFF 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)。

GSC_ERR_01

GSC モジュール

カテゴリ

機能

機能

一部のタイマが同じセキュリティ属性になっています

説明

PPC_SECATTRIB_TIMER レジスタでは、SEC_TIMA0_0 レジスタと SEC_TIMA0_1 レジスタが連結されているため、TIMERA0_0 と TIMERA0_1 のセキュリティ属性を同じにする必要があります。TIMERG4_2 と TIMERG4_1 も同じ構成ですが、これは、DMA がタイマにアクセスしようとしたときに限定されます。

回避方法

TIMERA0_0 と TIMERA0_1 の両方を同じセキュリティレベルに維持してください。これが不可能な場合は、別のタイマ インスタンスのペアを使用します。特に DMA を使用して TIMERG4_2 または TIMERG4_1 にアクセスする場合は、TIMERG4_2 と TIMERG4_1 を同じセキュリティレベルに維持してください。

IOMUX_ERR_01 IOMUX モジュール

カテゴリ

機能

機能

リセット後に TDO ピンが Low 信号を出力します。

説明

リセット後の TDO ピンが Low 信号として駆動されます。その他のピンはリセット後にハイ インピーダンス状態になります。

回避方法

このピンをハイ インピーダンス状態にする必要がある場合は、PC ビットと PF ビットの両方を 0 に設定します。それ以外の場合は、この IO で使用する予定のペリフェラルを接続します。

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_15 PMCU モジュール

カテゴリ

機能

機能

SYSPLL がイネーブルになる前の GENCLKEN クロック関連の構成が無視されます

PMCU_ERR_15 (続き)

PMCU モジュール

説明

SYSPLL を構成する場合、HSCLKEN レジスタの SYSPLLEN ビットを設定した後に GENCLKEN レジスタを構成します。SYSPLLEN ビットがセットされる前に、GENCLKEN レジスタの EXTDIVCAN、MCLKEXTDIVEN、I2SPLLCLKDIVEN、I2SPLLCLKDIVCFG ビットがプログラムされると、GENCLKEN レジスタの構成が適切に設定されず、EXTDIVCAN と I2SPLLCLKDIVCFG の構成は無視されます。

回避方法

SYSPLLEN ビットがセットされた後で、GENCLKEN レジスタの EXTDIVCAN、MCLKEXTDIVEN、I2SPLLCLKDIVEN、I2SPLLCLKDIVCFG ビットを構成します。

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。

RST_ERR_02

RST モジュール

カテゴリ

機能

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

RST_ERR_02 (続き) RST モジュール

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。

SYSCTL_ERR_01 SYSCTL モジュール

カテゴリ

機能

機能

SW-POR 機能は、HW_POR と組み合わせて使用できます

説明

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

回避方法

番号

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。

SYSPLL_ERR_01

(続き)

SYSPLL モジュール

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 = 160MHz; LFCLK = 32.768kHz
- 測定 FCC カウント = $160,000,000 / 32,768 = 4,882$

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 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 = $4,882 * 1.05 = 5,126$
- FCC カウント下限 = $4,882 * 0.95 = 4,638$

タイミング考慮事項:

- クロック同期時間: 5 ~ 6 LFCLK サイクル
- FCC トリガ時間: 1 ~ 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 = 160MHz; HFCLK = 40MHz; LFCLK = 32.768kHz
- 予想クロック比 = $160\text{MHz} / 40\text{MHz} = 4.0000$
- 測定 FCC カウント (SYSPLL) = $160,000,000 / 32,768 = 4,882$
- 測定 FCC カウント (HFCLK) = $40,000,000 / 32,768 = 1,220$
- 測定クロック比 = $4,882 / 1,220 = 4.0016$

SYSPLL_ERR_01

(続き)

SYSPLL モジュール

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 ~ 6 LFCLK サイクル
- FCC トリガ時間: 1 ~ 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:

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 100 μ s かかります。

回避方法

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

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

TIMER_ERR_06

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

TIMER_ERR_06 (続 き)

TIMER モジュール

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。

TIMER_ERR_08

TIMx モジュール

カテゴリ

機能

機能

高度タイマ GEN_EVENT1 出力のシーケンシャル イベント生成が無視されます

説明

高度タイマ (TIMA) では、GEN_EVENT1 は 1 つのイベントに対してのみ機能し、最初のイベントがトリガされた後、イベント ハンドラを介して適切に伝搬されません。

回避方法

高度タイマでの繰り返しイベント出力に GEN_EVENT1 を使用しないでください。代わりに GEN_EVENT0 を使用するか、汎用タイマ (TIMG) インスタンスを使用してください。

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

UNICOMMI2CC_E RR_01 (続き)

UNICOMMI2CC モジュール

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 の値が大きいの(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 = $3 \times 2 \times (80\text{MHz} / 4\text{MHz}) = 120$ 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

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

UNICOMMI2CC_E RR_03 (続き)

UNICOMMI2CC モジュール

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大きくないため、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。

UNICOMMSPI_ER R_02

UNICOMMSPI モジュール

カテゴリ

機能

機能

コントローラー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。

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

UNICOMMSPI_ER

R_04 (続き)

UNICOMMSPI モジュール

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

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、ブレイク 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

ブレイク フィールドのシナリオ (ハイパルス グリッチ) :ブレイク 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、ブレイク 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

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

UNICOMMUART_E

RR_05 (続き)

UNICOMMUART モジュール

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_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_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

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

UNICOMMUART_E

RR_14 (続き) *UNICOMMUART モジュール*

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。

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 June 30, 2026 to September 16, 2026 (from Revision B (June 2026) to Revision C (September 2026))

Page

• I2S_ERR_01 アドバイザリを追加しました.....	6
• RST_ERR_01 回避策を更新しました.....	7
• RST_ERR_02 アドバイザリを追加しました.....	7
• UNICOMMI2CC_ERR_02 アドバイザリを追加しました.....	12
• UNICOMMI2CC_ERR_03 アドバイザリを追加しました.....	12
• UNICOMMSPI_ERR_02 アドバイザリを追加しました.....	13
• UNICOMMSPI_ERR_03 アドバイザリを追加しました.....	13
• UNICOMMSPI_ERR_04 アドバイザリを追加しました.....	13
• UNICOMMUART_ERR_12 アドバイザリを追加しました.....	17
• UNICOMMUART_ERR_13 アドバイザリを追加しました.....	17
• UNICOMMUART_ERR_14 アドバイザリを追加しました.....	17

-
- UNICOMMUART_ERR_15 アドバイザリを追加しました..... [18](#)
-

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](#) 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