参考資料

Application Note

異種 SOC でのメイン ドメインの復旧

Vinit Patel, Sudheer Kumar

概要

車載用と産業用の各システムで異種のマルチコア プロセッサを使用するには、障害の自動検出機能と復旧機能を搭載した高い信頼性が必要です。TI の TDA4x システム オン チップ (SoC) には、手動の介入なしで連続的に動作する必要がある ARM Cortex-A72、Cortex-R5F、C7x DSP コアなど複数のプロセッシング コアが搭載されています。このアプリケーション ノートでは、プロセッサ間通信 (IPC) ドライバと低消費電力管理 (LPM) ドライバを使用して、TDA4x プラットフォーム上でコア間のハートビート監視システムを実装する方法について説明します。このソリューションは、ピンポン プロトコルによるすべてのリモート コアのリアルタイム監視、構成可能な時間枠内の自動クラッシュ検出、マイコン専用モードのパワー サイクリングによる自律的な復旧を実現します。テストの結果、合計復帰時間は約 1.2 秒で、システム全体のパワー サイクルよりも高速です。この資料では、理論、実装の詳細、テスト結果を掲載しており、エンジニアが TDA4x ベースのシステムに同様のフォールト トレラント設計を実装するのに役立ちます。

目次

商標

1 概要

TDA4x は、車載および産業用アプリケーション向けに設計された高性能異種マルチコア システム オン チップ (SoC) です。これは、以下の複数の処理要素を統合するものです:

- ハイレベル オペレーティング システムを実行するための複数の ARM Cortex-A72 コア

- リアルタイム制御タスクに適した多数の ARM Cortex-R5F コア

- 計算負荷の高いアルゴリズム向けの複数の C7x デジタル信号プロセッサ

本番環境では、これらのコアは継続的に動作し、高可用性を維持する必要があります。ソフトウェアのバグ、メモリの破損、無限ループ、またはハードウェアの故障が原因で、個々のコアが応答しなくなる可能性があります。このような障害発生時に自動検出および復旧メカニズムがない場合、手動の介入が必要ですが、これは車載用や産業用の環境では許容されません。

このアプリケーション ノートでは、以下の機能を使用してメイン ドメイン システムのハートビート監視と自動復旧を実装することで、これらの課題に対処しています:

- すべてのリモート コア (RTOS および Linux) のリアルタイム監視

- IPC ピンポン プロトコルによる自動クラッシュ検出

- LPM ドライバを使用した自律的な復旧

- 設定可能な監視間隔と再試行メカニズム

- パフォーマンス分析のための詳細なタイミング プロファイリング

このリファレンス デザインは、SOC の常時オンのマイコンドメインを活用しています。このドメインは、マイコン専用モード遷移中も電力を供給し続けます。MCU1_0 コアはハートビート モニタとして機能し、他のすべてのコアの応答性を継続的にチェックします。コア障害が検出されると、システムはマイコンドメイン状態を維持しながら MAIN ドメインのパワー サイクルを自動的に実行することで、システム全体を再起動することなく迅速な復旧が可能になります。

このドキュメントでは、以下の内容を記載しています:

- 詳細なシステム アーキテクチャと設計の根拠

- タイミング測定によるテスト結果

対象者:このアプリケーション ノートは、プロセッサ SDK RTOS を使用して、Jacinto プラットフォーム上でフォールト トレラント システムを開発する組込みソフトウェア エンジニアを対象としています。

前提条件:

- TDA4x アーキテクチャに精通している

- IPC フレームワークと remoteproc フレームワークを理解している

- TI のプロセッサにおいて FreeRTOS と Linux の実装経験がある

2 システム アーキテクチャ

2.1 コア構成

TDA4x SoC には、クラスタに編成された複数のプロセッシング コアが搭載されています。表 2-1 に、この実装で使用されるコア構成を示します。

表 2-1. コア構成

コア	タイプ	オペレーティング システム	職種
MCU1_0	R5F	FreeRTOS	LPM ドライバ + ハートビート モニタ
MCU1_1	R5F	FreeRTOS	この構成では使用しません
MCU2_0	R5F	FreeRTOS	リモート コア (監視対象)
MCU2_1	R5F	FreeRTOS	リモート コア (監視対象)
MCU3_0	R5F	FreeRTOS	リモート コア (監視対象)
MCU3_1	R5F	FreeRTOS	リモート コア (監視対象)
MPU1_0	A72	Linux	Linux コア (監視対象)
C7x_1	DSP	FreeRTOS	DSP コア (監視対象)

MCU1_0 コアは、マイコンドメインに存在し、マイコン専用モード中も電力供給が維持されるため、ハートビート モニタとして指定されています。これにより、モニタは MAIN ドメインのパワー サイクルに耐え、復旧をオーケストレーションできます。

2.2 パワー ドメイン アーキテクチャ

TDA4x SOC は、サブシステムを 3 つのパワー ドメインに編成し、選択的な電力制御を実現します。表 2-2 に、これらのドメインを示します。

表 2-2. パワー ドメイン

ドメイン	コンテンツ
WKUP ドメイン	常時オン ロジック、ウェークアップ コントローラ、WKUP ペリフェラル
マイコンドメイン	MCU1_0、MCU1_1、マイコンペリフェラル、MCU SRAM
メイン ドメイン	A72 クラスタ、MCU2-4 R5F コア、C7x DSP、DDR、ほとんどの I/O

LPM ドライバは、次の 2 つの主要電力モードをサポートしています:

1. **アクティブ モード:** 3 つのドメインすべてに電力が供給されます。これは、すべてのコアがアプリケーションを実行する場合の通常の動作状態です。
2. **マイコン専用モード:** WKUP とマイコンドメインのみに電力が供給されます。MAIN ドメインは、すべての A72 コア、MAIN ドメイン R5F コア、C7x DSP、DDR コントローラを含め、完全に電源オフになっています。

この復旧メカニズムは、次の電力モードを利用します:

アクティブ -> マイコン専用 -> アクティブ

このシーケンスは、マイコンドメイン (したがって、私たちのメカニズム) が動作し続けている間、MAIN ドメインのパワー サイクルを実行します。TPS6594x PMIC は、MCU1_0 からの I2C 制御下で電圧レールを管理します。

2.3 IPC フレームワークの概要

プロセッサ間通信 (IPC) フレームワークは、コア間でのメッセージ パッシングを可能にします。この実装では RPMessage API を使用しており、以下の機能を提供します:

- 共有メモリを介した VirtIO ベースの転送

- 接続を確立するための名前付きサービス エンドポイント

- 非同期メッセージの送受信 (タイムアウト付き)

ハートビート監視には、次の 2 つの IPC サービスが使用されます:

1. **ti.ipc4.ping-pong (エンドポイント 13):** RTOS コア通信に使用されます。MCU1_0 モニタは ping メッセージを送信し、リモート RTOS コアからの pong 応答を待ち受けます。
2. **rpmsg_chrdev (エンドポイント 14):** Linux 通信に使用されます。Linux では、ユーザー空間アプリケーション向けに、カーネル RPMsg ドライバではなく、キャラクタ デバイス インターフェイス (rpmsg_char) が必要です。

デュアルサービス アプローチが必要な理由は次のとおりです:

- Linux はエンドポイントを動的に割り当てるため、最初のハンドシェイクが必要です

- RTOS コアは静的なエンドポイント割り当てを使用します

3 ハートビート監視設計

3.1 ピンポン プロトコル

ハートビート監視では、単純なピンポン プロトコルを使用します:

プロトコル動作:

1. モニタは「ping N」メッセージを送信します。N はシーケンス番号です
2. リモート コアは「pong N」で応答します
3. タイムアウト内に応答がない場合は、MAX_RETRIES まで再試行します
4. すべての再試行が失敗した場合は、コアをクラッシュ状態として宣言します
5. プロトコル パラメータ:

   間隔: 監視サイクルの間隔は 500ms です

   タイムアウト: 各 ping 応答を 2000ms 待機します

   最大再試行回数: 失敗を宣言する前に 3 回試行します

3.2 デュアルタスク アーキテクチャ

2 つの独立した FreeRTOS タスクが、さまざまなコア タイプの監視を処理します:

センダー タスク (ti.ipc4.ping-pong サービス):

- RTOS コアを監視: mcu2_0、mcu2_1、mcu3_0、mcu3_1、c7x_1
- エンドポイント 13 を使用します
- マイコンが通信を開始: ping を送信して、pong を受信します
- 各監視サイクルで、すべての RTOS コアを反復処理します

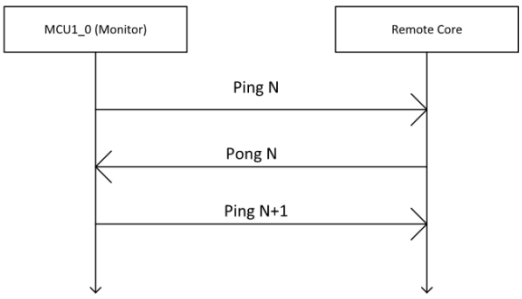

図 3-1. SenderTask メカニズム

レスポンダ タスク (rpmsg_chrdev サービス) :

- Linux コアを監視:mpu1_0
- エンドポイント 14 を使用します
- 動的エンドポイントをキャプチャするために、最初の「Linux ready」メッセージを待機します
- その後、マイコンが通信を開始して、ping を送信し、Linux は pong で応答します
- 最初のダミー メッセージを使用した理由は、Linux がエンドポイントを動的に割り当て、ダミー メッセージを使用して MCU1_0 (モニタ) が Linux のリモート エンドポイントを取得して通信するためです。

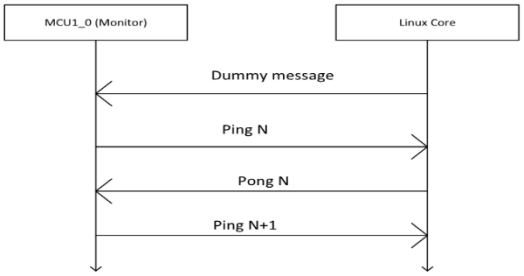

図 3-2. ResponderTask メカニズム

表 3-1 に、各コアの監視割り当てを示します。

表 3-1. 監視割り当て

コア	サービス	エンドポイント	監視元
mpu1_0 (Linux)	rpmsg_chrdev	14	レスポンダ タスク
mcu2_0	ti.ipc4.ping-pong	13	センダー タスク
mcu2_1	ti.ipc4.ping-pong	13	センダー タスク
mcu3_0	ti.ipc4.ping-pong	13	センダー タスク
mcu3_1	ti.ipc4.ping-pong	13	センダー タスク
c7x_1	ti.ipc4.ping-pong	13	センダー タスク

2 つのタスクに分離することで、次のような利点が得られます:

- 異なるタイミング要件を持つ独立した監視ループ

- Linux 固有のハンドシェイク ロジックの分離

- 並列監視機能

3.3 クラッシュ検出ロジック

クラッシュ検出ロジックは、監視対象の各コアの状態を維持します:

状態遷移:

- **NOT_RESPONDING -> ALIVE:** 初期化または復旧後に、最初の正常な pong 信号を受信した場合。ログには「復旧:コア X は今、生きています！」と記録
- **ALIVE -> CRASHED:** すべての再試行が終了しても ping が失敗した場合。ログには「警告:コア X が応答していません！」と記録これによりクラッシュ フラグがトリガされます。

メカニズムの概要:

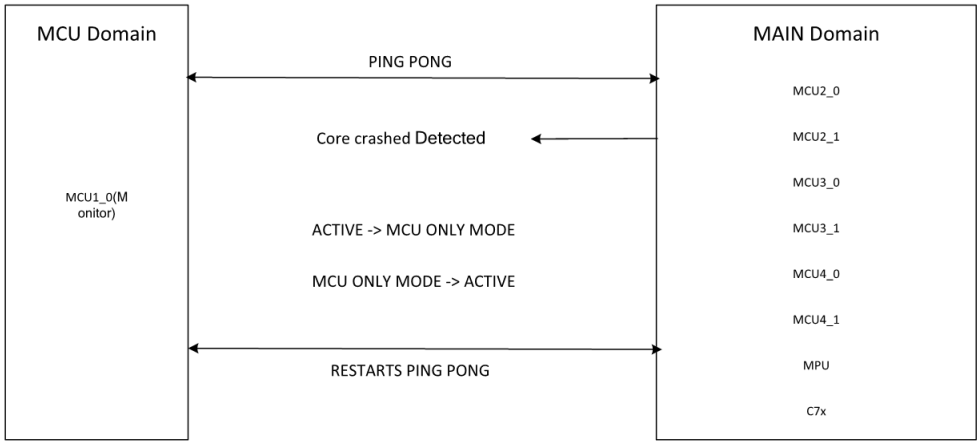

図 3-3. 簡単なワークフロー

4 復旧メカニズム

4.1 電力状態遷移

クラッシュが検出されると、復旧メカニズムは次のシーケンスを実行します:

1. IPC タスクを終了するように信号を送信します
2. セマフォを介してすべてのタスクが終了するまで待機します
3. IPC フレームワークを初期化解除します
4. リソース テーブルをシャドウの場所に保存します
5. アクティブ -> マイコン専用モードの遷移 (MAIN ドメインの電源オフ)
6. マイコン専用 -> アクティブ モードの遷移 (MAIN ドメインの電源オン)
7. シャドウの場所からリソース テーブルを復元します
8. IPC フレームワークを再初期化します
9. リモート コアを起動します
10. 監視メカニズムを再開します

電力の遷移は、次の LPM ドライバに実装されています:

Lpm_activeToMcuSwitch():

- MAIN ドメインにソフトウェア リセットを発行します
- MAIN ドメインのディープ スリープ分離を有効にします
- PMIC 状態の変更:I2C 経由でアクティブ -> マイコン専用

Lpm_mcuToActiveSwitch():

- PMIC 状態の変更:I2C 経由でマイコン専用 -> アクティブ
- MAIN ドメインのディープ スリープ分離を無効にします
- WKUPMCU2MAIN ブリッジと MAIN2WKUPMCU ブリッジを有効にします
- ボード構成を再初期化します
- DDR コントローラ、PLL、クロックを復元します

5 実装の詳細

5.1 構成パラメータ

ハートビート監視パラメータは、lpm_ipc.c で定義されています。

表 5-1. 構成パラメータ

パラメータ	値	説明
HEARTBEAT_INTERVAL_MS	500	監視サイクル間の時間 (ms)
HEARTBEAT_TIMEOUT_MS	2000	各 ping 応答のタイムアウト (ms)
HEARTBEAT_MAX_RETRIES	3	失敗を宣言する前に再試行します

5.2 パラメータ チューニング ガイダンス:

HEARTBEAT_INTERVAL_MS の減少:

+ コア障害の迅速な検出

- IPC オーバーヘッドと CPU 使用率の増加

HEARTBEAT_INTERVAL_MS の増加:

+ システム オーバーヘッドの低減

- クラッシュ検出速度の低下

HEARTBEAT_TIMEOUT_MS の減少:

+ 再試行ごとの障害検出時間の短縮

- コアに高負荷がかかっている場合、誤検出を引き起こす可能性

HEARTBEAT_MAX_RETRIES の増加:

+ 過渡的な通信遅延に対する耐性が向上

- 実際のクラッシュを検出する時間の増加

5.3 Linux 側の rpmsg_char 実装

ハートビート監視システムでは、MCU1_0 モニタとの通信を処理するための Linux ユーザー空間アプリケーションが必要です。この実装では、TI の rpmsg_char ライブラリとフレームワークを使用して、ユーザー空間アプリケーションとカーネルの RPMsg サブシステムをブリッジ接続します。

ti-rpmsg-char ライブラリは、Linux 上での RPMsg 通信用のユーザー空間 API を提供します。これは次の項目で構成されます:

- **コア ライブラリ:** libti_rpmsg_char.so.0.6.10 - エンドポイント管理とカーネル インタフェイスを処理します

- **シンプルなアプリケーション:** rpmsg_char_simple - Linux コアとのハートビート通信に使用されます

6 テスト結果と性能

6.1 タイミング プロファイル

タイミング プロファイリング システムは、復旧プロセス中の重要なポイントでタイムスタンプをキャプチャします。表 6-1 に、実際のテスト実行から測定値を示します。

表 6-1. 時間プロファイル

位相	時間 (µs)	時間 (ms)
クラッシュ検出 -> MCU ONLY の準備を開始	34 ～ 38	約 0.035
アクティブ -> マイコン専用モードへの遷移	7011 ～ 7044	約 7.0
マイコン専用 -> アクティブ モードへの遷移	150711 ～ 150752	約 150.7
Linux A72 コア ブート	1069435	約 1069.4
合計復帰時間		約 1227

約 1227ms の合計復帰時間は、フル パワー サイクルのブート時間約 1333ms と比較して良好であり、マイコンドメインのすべての状態を維持しながら約 105ms を短縮できます。

6.2 復旧検証

ハートビート監視システムは、次のような複数のクラッシュ シナリオでテストされました。

テスト ケース 1：RTOS コア クラッシュ (c7x_1)

- C7x DSP を停止することによるシミュレーション
- 検出時間：3 回の再試行サイクル
- 復旧：正常に完了し、すべてのコアが ALIVE 状態に戻りました

テスト ケース 2：R5F コア クラッシュ (mcu2_0)

- MCU2_0 コアの停止シミュレーション
- 検出時間：3 回の再試行サイクル
- 復旧：正常に完了し、すべてのコアが ALIVE 状態に戻りました

テスト ケース 3：Linux コア クラッシュ (mpu1_0)

- rpmsg_char_simple アプリケーションを強制終了するか、mpu1_0 コアを停止することによるシミュレーション
- 検出時間：3 回の再試行サイクル
- 復旧：正常に完了し、Linux が再起動して再接続され、すべてのコアが ALIVE 状態に戻りました

7 まとめ

本アプリケーション ノートでは、TDA4x プラットフォーム向けのコア間ハートビート監視と自動復旧の包括的な実装について説明しました。主な貢献は次のとおりです:

1. デュアルタスク監視アーキテクチャ:RTOS コアと Linux コア用に個別のタスクを設けることで、各環境に適したタイムアウト動作を用いた独立した監視が可能になります。
2. ピンポン プロトコル:シンプルな要求応答プロトコルは、構成可能なパラメータを使用して、信頼性の高いコア正常性検証を実現します。
3. セマフォ ベースの同期:適切なタスク同期により、競合状態のない信頼性の高い IPC シャットダウンが検証されます。
4. マイコン専用モードの復旧:SOC のパワー ドメイン アーキテクチャを活用すると、システム全体を再起動せずに自律的な復旧を実現できます。
5. タイミング プロファイル:詳細なタイミング測定により、約 1.2 秒の復旧性能が示されています。

この実装は、高可用性と自動復旧が不可欠な要件である、フォールトトレラントな車載および産業用システムの基盤を実現します。

今後の拡張には次のものがあります:

- 選択的なコア復旧 (MAIN ドメインの完全なパワー サイクルを行わずに、クラッシュしたコアのみの再起動)

8 参考資料

1. テキサス インスツルメンツ、『マイコン専用モードの概要』、ユーザー ガイド。
2. テキサス インスツルメンツ、『メイン ドメイン コアのハートビート監視と自立型復旧を可能にする方法』、FAQ (よくある質問)。
3. テキサス インスツルメンツ、『TDA4VM 上でマイコン専用モードを実行および検証する方法』、FAQ (よくある質問)。
4. テキサス インスツルメンツ、『TDAVH 上でマイコン専用モードを実行および検証する方法』、FAQ (よくある質問)。
5. テキサス インスツルメンツ、『*Linux rpmsg_char* ドライバ ガイド』、ユーザー ガイド。

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