参考資料

*Application Note*

# *C29x 安全およびセキュリティ ユニットによるランタイム安全およびセキュリティの実装*

*Ibukun Olumuyiwa, Marlyn Rosales Castaneda, and Aishwarya Rajesh*

**概要**

安全/セキュリティ ユニット (SSU) は、アプリケーション コードのランタイム機能安全とサイバーセキュリティ保護を実現する F29x デバイスの統合モジュールです。SSU の機能を活用すると、堅牢な Freedom From Interference (FFI)、セキュアなタスク分離、デバッグ セキュリティ、ハードウェアでのファームウェア更新保護を実現できるほか、リアルタイム制御システムで必要とされる低レイテンシ性能を維持できます。F29 SDK の一部として提供される SysConfig ツールは、SSU を構成し、ユーザー アプリケーションで安全およびセキュリティ保護を実現するための使いやすいグラフィカル ユーザー インターフェイス (GUI) を提供します。このアプリケーション ノートでは、SSU のさまざまな機能について検討し、組込みシステムの開発者が SysConfig 内の SSU ツールを活用してリアルタイム アプリケーションにおける実行時の安全性とセキュリティを設計および実装する方法を解説します。

## 目次



## 図の一覧

## 商標

# 1 はじめに

テキサス・インスツルメンツの C29 CPU は、リアルタイム制御アプリケーション向けに、業界をリードする性能を実現します。C29 は、128 ビットの超長命令ワード (VLIW) アーキテクチャ、64 ビットの固定小数点演算と浮動小数点演算、超低レイテンシの処理、ハードウェア割り込み優先度設定を特徴とし、最も要求の厳しい車載用および産業用制御アプリケーションを実行するための適切な機能を備えています。SSU は、C29 CPU と協調して動作し、システム設計者がリアルタイム性能を犠牲にせずに、リアルタイム制御アプリケーションにおける安全とセキュリティに関する最も厳格な最新規格を満たすのに役立ちます。SSU により、ユーザーは、真の FFI、セキュアなタスク分離、高度なデバッグおよびファームウェア更新セキュリティを実現しながら、最も厳しい条件が要求されるリアルタイム制御システムに必要とされる高速および低レイテンシの処理を維持できます。

このアプリケーション ノートでは、C29x CPU と SSU を使用して、ランタイム アプリケーションの安全性とセキュリティをリアルタイム制御システムに実装する方法について説明します。C29x、SSU のアーキテクチャは、動的なコンテキスト依存のメモリ保護、複数の専用 CPU STACK ポインタによるセキュアなタスク分離、セキュリティのためのマルチユーザー デバッグ ゾーンを備えています。

# 2 補足オンライン情報

特定のデバイスにおける C29x CPU および SSU の詳細な説明については、デバイス固有のデータ シートと、対応するテクニカル リファレンス マニュアル (TRM) を参照してください。

このアプリケーション レポートは、F29H85x ファミリを使用して作成されています。F29 SDK と SysConfig ツールは、すべての F29x プラットフォーム デバイスをサポートしています。

- *C29x CPU および命令セット ユーザー ガイド*
- *F29H85x および F29P58x リアルタイム マイクロコントローラ データシート*
- *F29H85x および F29P58x リアルタイム マイクロコントローラ テクニカル リファレンス マニュアル*
- *C29x Academy : 安全およびセキュリティ ユニット (SSU) の章*

TI E2E™ コミュニティにより、追加サポートが提供されています。

# 3 SSU の概要

SSU は、C29 CPU サブシステムには不可欠であり、状況に制約のあるメモリの保護、ランタイムコード分離、デバッグ セキュリティ、セキュア ファームウェア アップデートを提供します。SSU は、C29 CPU とシステムの他の部分との間のファイアウォールとして動作し、ユーザー アクセス保護ポリシーを適用して、デバッグ アクセスとフラッシュ コントローラの動作を管理します。図 3-1 に、C29 サブシステムに統合された SSU の概要を示します。SSU で利用できる機能は、ソフトウェアのオーバーヘッドを増やさずにリアルタイム制御システムで真の FFI を実現するために使用でき、そうしないとリアルタイム性能に悪影響を及ぼす可能性があります。SSU を使うと、システム設計者は、同じ CPU コア上で複数の制御および通信機能を組み合わせながら、各機能を他の機能から分離することができます。これにより、システムの目標を達成するために必要なコア数を削減したり、システムの安全整合性レベルを向上させたりできます。

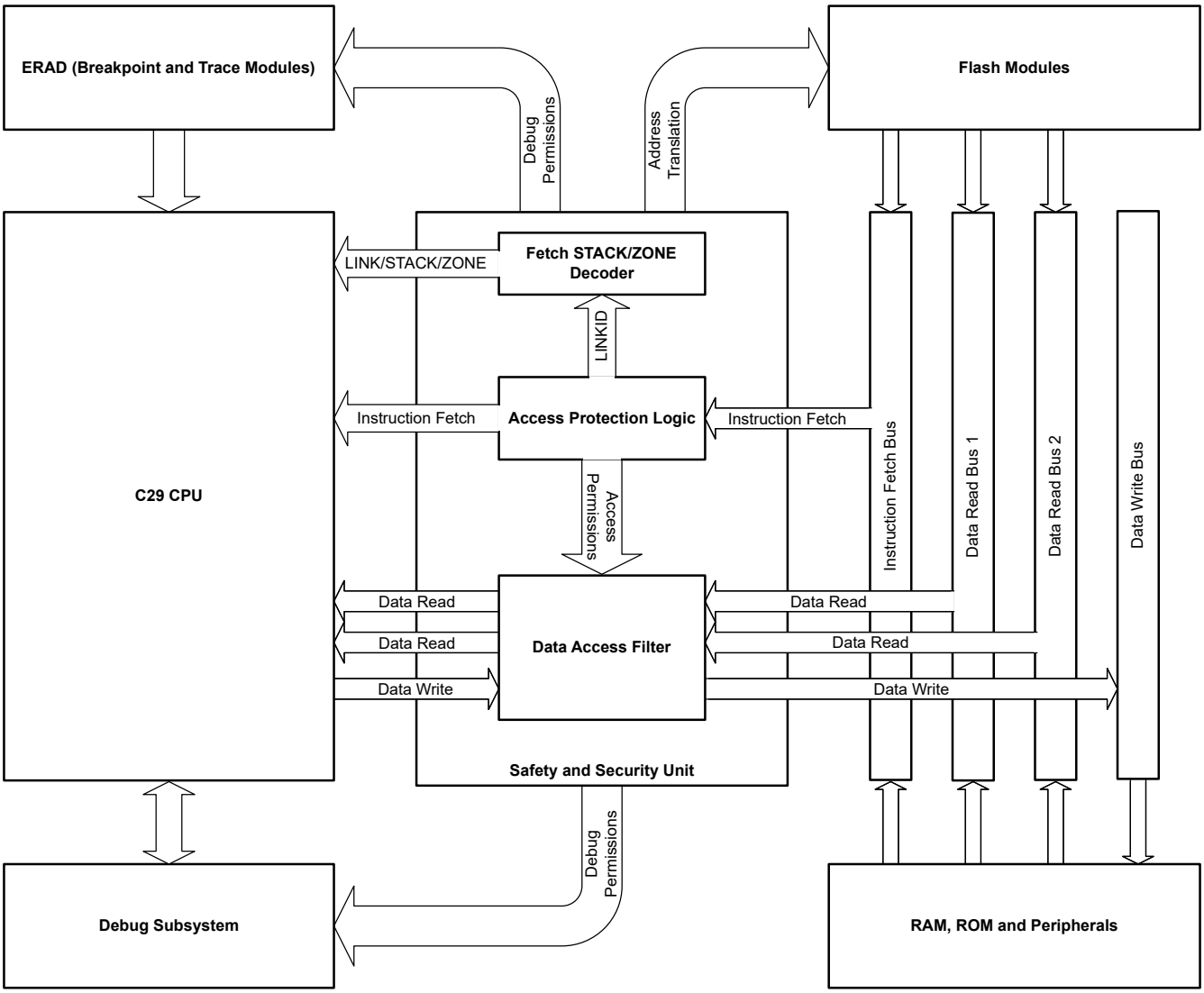


**図 3-1. SSU システム ブロック図 (概略図)**

# 4 主要概念の定義

ここでは、主要な概念の定義について説明します。

**アクセス保護領域 (APR)** これは SSU のメモリ保護の基本単位です。アクセス保護範囲は、フラッシュ メモリ、SRAM、ペリフェラルの特定の領域を対象としています。各 APR は、各 LINK の読み取りおよび書き込みアクセス許可を定義します。APR は、コード領域として構成することもできます。コード領域は、そのメモリ領域からの CPU 命令のフェッチを可能にします。

**LINK** C29 CPU サブシステムでは、LINK が状況に応じたメモリ保護の基盤を形成します。各 LINK は、実行可能コードの 1 つまたは複数の領域を表すことができます。関連 LINK 識別子は、そのコードでアクセスできるデータ メモリ領域 (APR) を決定するために使用されます。

**STACK** STACK は、コード実行コンテキストを互いに分離します。各 STACK には、C29 CPU に専用の STACK ポインタがあり、他の STACK からハードウェア安全性とセキュリティを分離できます。すべての LINK は 1 つの STACK のみに属しますが、STACK には複数の LINK を含めることができます。

**ゾーン** ゾーンによって、デバッグおよびファームウェアの更新権限が決まります。APR、LINK、および STACK は CPU ごとに個別に定義されますが、ゾーンはデバイス全体にまたがっており、ハードウェア セキュリティ モジュール (HSM)(SSU によって制御されません) は除外されています。

**HSM** ハードウェア セキュリティ マネージャは、デバイス内の自己完結型サブシステムであり、セキュア ブート、セキュア ストレージ、デバッグおよびファームウェア更新セキュリティ、ランタイム暗号化サービスなど、主要なセキュリティ サービスを提供します。HSM は、アプリケーション C29 CPU サブシステムに不可欠な部分である SSU とは異なります。HSM と SSU は、デバッグ許可を除き、デバイスで補完的な役割と直交的な役割を実行します。HSM と SSU の両方が、そのリソースへのデバッグ アクセスを有効にする前に、リソースへのアクセスを許可する必要があります。

**部分的デバッグ** ゾーンが部分的デバッグ用にイネーブルになっている場合、ユーザーは CPU 実行のデバッグ (HALT、RESUME、および CPU レジスタの表示) が許可されますが、そのゾーン内の LINK によってアクセスできるメモリへのデバッグ読み取りおよび書き込みアクセスはブロックされます。

**完全デバッグ** ゾーンで完全デバッグがイネーブルになっている場合、ユーザーは CPU をデバッグし、ゾーン内の任意の LINK で許可されているすべてのメモリ アクセスを実行できます。

**SECCFG** これは、SSU 構成設定の保存用に指定された特別なフラッシュ領域です。SECCFG 領域に保存された値は、デバイスのブート中に SSU レジスタにロードされます。これらの設定のほとんどは、実行時に変更できません。変更できるのは、新しい値を SECCFG にプログラミングし、デバイスをリセットすることのみです。

**UPP** ユーザー保護ポリシー。これは、SECCFG 領域にプログラムされる SSU 構成設定の集合です。

**メモリ領域** SysConfig で構成されているメモリ領域。これは、アクセス保護範囲 (APR) に相当します。

**モジュール** SysConfig では、モジュールは、LINK、その LINK に関連付けられているコード メモリ領域 (実行可能 APR)、モジュールに*属する*データ メモリ領域 (データ APR) とペリフェラル、およびモジュールに関連するペリフェラル割り込みで構成されます。実際には、モジュールにより、ユーザーはアプリケーションを別々のタスクまたはパーティションに編成し、機能安全とセキュリティのために相互に分離できるようになります。

**共有メモリ** SysConfig では、共有メモリは、複数のモジュールでアクセス可能な 1 つ以上の APR で構成されています。共有メモリを使用して、異なるメモリ範囲のモジュール間でデータを共有でき、それらのモジュールに属する他のメモリ領域の安全保護を維持できます。

**サンドボックス** SysConfig では、サンドボックスは 1 つの STACK で構成され、1 つまたは複数のモジュールを含めることができます。

**RTOS** FreeRTOS® や AUTOSAR® のようなリアルタイム オペレーティング システム。

# 5 安全およびセキュリティ目標

システム設計者の皆様は、安全/セキュリティ ユニットを使用して、リアルタイム組込みシステムの設計において、安全およびセキュリティに関する重要な目標を達成することができます。これらの目的は次のとおりです。

1. **メモリ保護:** 機能安全目標をサポートする組込みマイクロコントローラの重要な要素は、メモリ保護ユニット (MPU) です。MPU は、システム内のメモリを介してアクセス制御ルールを適用し、不正な読み取り、偶発的な上書き、またはコードやデータに対する不正な変更を防止します。メモリ保護は、システムの安定性、信頼性、セキュリティを維持する上で重要な役割を果たします。SSU は高度な MPU 機能を提供します。これにより、ソフトウェアの介入なしで、状況に応じたリアルタイムのスイッチング保護が可能になります。
2. **Freedom from Interference:** 車載用電子機器の機能安全規格を定義した ISO 26262 規格では、Freedom From Interference (FFI) とは、「*安全要件の違反につながる可能性のある複数の素子の間にカスケード故障が発生していない*」と定義されています。カスケード接続の障害は、システム内の 1 つのコンポーネントで障害が発生した場合に発生し、そのコンポーネントの故障が原因で、システム内の別のコンポーネントに障害が発生します。これらの障害が発生すると、正のフィードバック ループが徐々に増加する可能性があります。SSU は、複数の異なるシステム ソフトウェア コンポーネントを互いに完全に分離するためのメカニズムを備えているため、1 つの部品の安全性障害が発生してもアプリケーションの残りの部分が損なわれることはありません。
3. **セキュリティの分離:** 干渉からの安全の自由に加えて、SSU はセキュリティ分離の目標をサポートしており、各アプリケーション コンポーネントに、実行中のコード資産とデータ資産の機密性と整合性を保護するセキュアな実行環境を提供します。
4. **リアルタイム性能:** SSU の重要な目標は、リアルタイム性能に影響を及ぼさずに、安全性とセキュリティの保護機能を提供することです。メモリ保護、セキュリティ分離、その他の SSU 機能はすべて、ソフトウェアの介在なしでリアルタイムに実行されるため、スーパーバイザ ソフトウェアのオーバーヘッドによる余分なレイテンシが不要になります。C29 CPU の業界をリードする性能と組み合わせることで、システム設計者は性能、安全性、セキュリティの目標を犠牲にせずに、同じ CPU 上で複数の制御機能を組み合わせ、システム全体のコストを低減できます。
5. **セキュア デバッグとファームウェア アップデート:** SSU には、システム ソフトウェアを複数のユーザー デバッグ ゾーンに分割できる機能があり、複数のチームが同じチップ上で異なるソフトウェア コンポーネントを安全に維持およびデバッグできます。SSU はまた、フラッシュ ファームウェアを管理し、ファームウェア アップデートの実行が許可されるユーザーやコードを制御し、Firmware-Over-The-Air (FOTA) や、A、B のスワッピングやロールバック保護をハードウェアで行う Live Firmware Update (LFU) などのメカニズムを可能にします。

# 6 システム設計

アプリケーション用に SSU を設定する最初の手順は、必要なシステム パーティションを決定することです。SSU には、アプリケーション サブシステムを分割するための 3 つのレベルの階層があります。

1. ゾーン：各ゾーンによって、チップ上のすべての C29 CPU に対するデバッグ アクセスが決まります。ゾーンは、複数のコードの所有者またはエンティティが、同じチップ上にあるアプリケーションの異なるパーティションを開発して維持できるように設計されています。たとえば、組み込みアプリケーションの特定の側面がサードパーティ ベンダーによって所有および保守されている場合、システムは次の 2 つのゾーンに分割できます。
   a. ZONE1：プライマリ ユーザー ゾーン (プライマリ システム開発者が所有)
   b. ZONE2：サードパーティ開発者が所有するセカンダリ ユーザー ゾーン。

   このパーティション分割により、サードパーティー開発者は、プライマリ ユーザーのコードやデータ資産へのアクセスを必要とせずに、同じチップ上でアプリケーション機能の開発、デバッグ、保守を行うことができます。さらに、各ユーザー ゾーンには、次の 2 つのレベルのデバッグ許可が用意されています。

   a. 部分的デバッグ – HALT、STEP、BREAKPOINT などの CPU デバッグ コマンドを使用できますが、メモリ アクセスはできません
   b. 完全デバッグ – ゾーン内に含まれるすべての LINK に対して、メモリ ロケーションへのアクセスが許可されています。

   たとえば、サードパーティ開発者のようなセカンダリ ユーザーは、ZONE2 でアプリケーション モジュールをデバッグし、ZONE1 への部分的なデバッグ アクセス権を付与することができます。これにより、セカンダリ ユーザーはプライマリ ユーザーの資産にアクセスすることなく、コンテキスト内でアプリケーションを効果的にデバッグできます。

   各デバイスには、次の 3 つのユーザー ゾーンがあります。ZONE1、ZONE2 および ZONE3。ZONE1 はプライマリ ユーザー ゾーン、ZONE2 と ZONE3 はセカンダリ ユーザー ゾーンです。
2. サンドボックス (STACK)：サンドボックスは、CPU 内でセキュリティと安全を分離します。各サンドボックスは、SSU 内の STACK に関連付けられています。各サンドボックスには、他のサンドボックスではアクセスできない専用の物理 STACK ポインタが CPU 内にあり、読み取り/書き込み権限がそのサンドボックスに属するコードのみに制限されている専用の STACK メモリ AP 領域があります。

   ある STACK から別の STACK に交差する場合、特別な C29 CPU ゲート命令が必要です。これらの命令は、各関数の開始時と終了時、および関数コールまたは分岐時にコンパイラによって挿入される必要があります。これらのメカニズムは、コード実行のリダイレクトや STACK の操作を試みるマルウェア攻撃に対するセキュリティ保護を提供します。

   サンドボックスは、SSU STACK と、STACK メモリ AP 領域を含む STACK に関連するすべてのもので構成されます。各 STACK は 1 つのゾーンに属しますが、1 つのゾーンに複数の STACK を含めることができます

**表 6-1. CPU ごとに事前定義済みの**

| STACK # | 説明 |
|---|---|
| STACK0 | この STACK は TI の内部用に予約されており、ユーザーが構成することはできません。 |
| STACK1 | この STACK は主にブートローダで使用されますが、オプションとして他のユーザーアプリケーションコードに関連付けることもできます。STACK1 は常に ZONE1 に関連付けられており、LINK (LINK1) は 1 つだけ含まれています。 |
| STACK2 | これはプライマリ ユーザー STACK です。STACK2 は常に ZONE1 に関連付けられています。STACK2 には常に LINK2 が含まれているが、他の LINK を含めることもできます。 |

3. アプリケーション モジュール (LINK)：アプリケーション モジュールは、システム アプリケーションの基本的なパーティションです。各モジュールは、単一の SSU LINK、LINK のコード、LINK に関連付けられたすべてのデータ メモリ AP 領域、およびモジュールに関連付けられたすべてのペリフェラルと割り込みを含む 1 つ以上のコード メモリ AP 領域で構成されます。

通常、コード AP 領域には .text およびコードを含む他のリンカ出力セクションが含まれ、データ AP 領域には .bss、.const およびデータと変数を含むその他のリンカ出力セクションが含まれます。

各 LINK により SSU メモリ保護が可能になり、CPU 内のその他の LINK からの安全保護を提供します。各 AP 領域は、各 LINK のアクセス権限を定義します。これらのアクセス許可は、LINK ID 命令に応じて、メモリ アクセスを実行するすべての命令に対してリアルタイムで適用されます。互いに安全に絶縁する必要がある機能は、個別のモジュールに配置できます。セキュリティ分離が必要な場合は、これらのモジュールは別々のサンドボックスに配置されます。そうでない場合は、モジュールを同じサンドボックスに配置できます。

**表 6-2. CPU ごとに事前定義済みの LINK**

| LINK # | 説明 |
|---|---|
| LINK0 | この LINK は TI の内部用に予約されており、ユーザーが構成することはできません。 |
| LINK1 | この LINK は主にブートローダで使用されますが、オプションとして他のユーザーアプリケーションコードに関連付けることもできます。CPU1.LINK1 には、AP で定義された保護に加えて、特定のシステム構成レジスタへのアクセスを可能にする特別な固定権限があります。 |
| LINK2 | これはプライマリ ユーザー LINK です。CPU1.LINK2 はシステム セキュリティ ルート LINK (SROOT) であり、システム構成レジスタへのアクセスと制御のオーバーライドを可能にする特別な固定アクセス許可を持っています。この LINK は通常、特権ホスト機能を RTOS レベルで実行します。 |

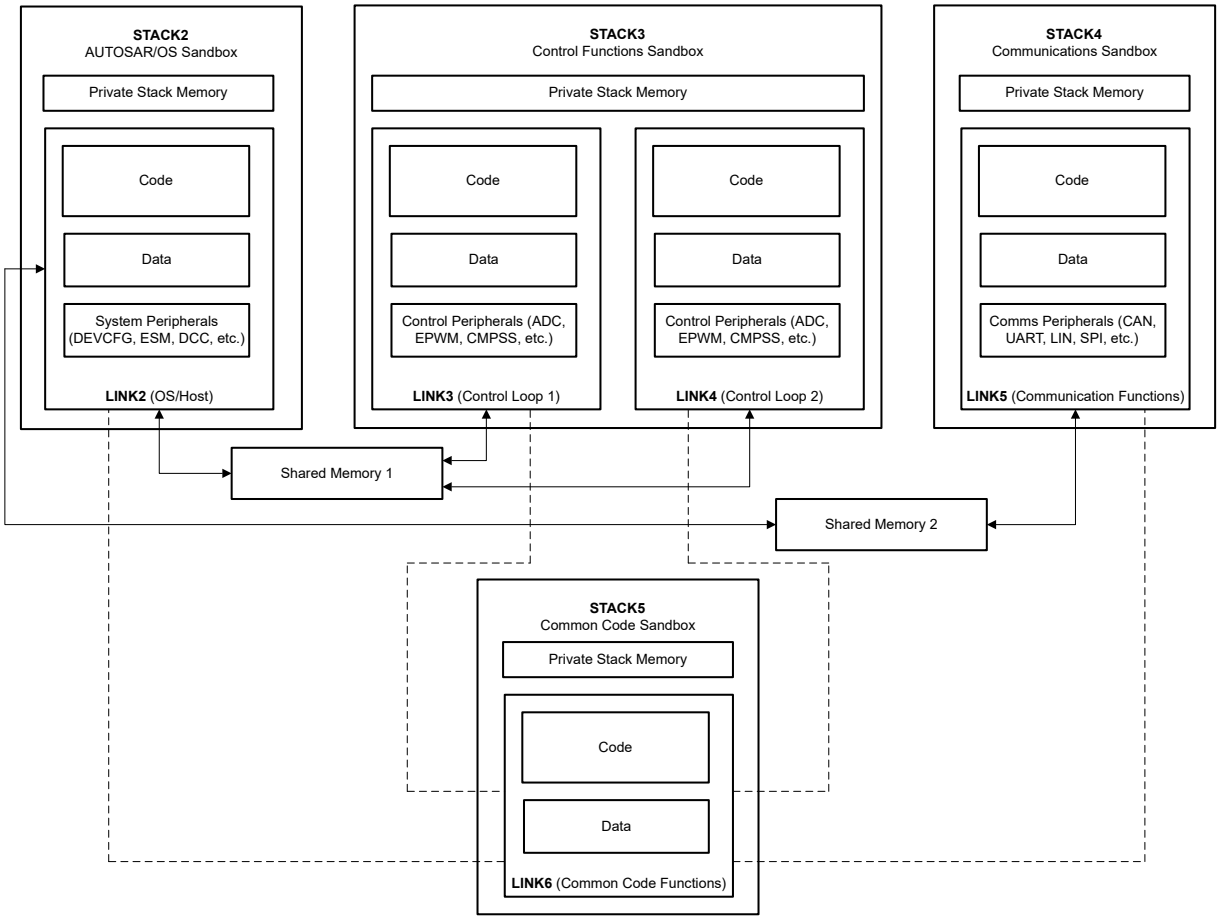


**図 6-1. SSU を使用したソフトウェア パーティションの例**

図 6-1 に、シングル CPU、シングル ゾーン システムでの SSU システム パーティションの例を示し、表 6-3 でこの構成の詳細について説明します。

**表 6-3. SSU パーティショニングの例に関する説明**

| STACK | LINK | 詳細 |
|---|---|---|
| STACK 2 | LINK2 | RTOS は STACK2.LINK2 で動作し、システム構成の初期化、ペリフェラルと割り込みの設定、メイン実行 LOP の開始を行います。さまざまなタスクおよび個別のタスク STACK を他のリンクに配置することができますが、すべての RTOS タスク/STACK は SSU STACK2 内に配置する必要があります。RTINT を使用する場合、RTINT に関連するコードとデータは、SSU STACK2 以外の別の SSU STACK に配置する必要があります。 |
| STACK 3 | LINK3 と LINK 4 | *制御ループ 1* と *制御ループ 2* の 2 つの制御機能があります。これらの制御機能はそれぞれ別々のアプリケーション モジュール (LINK) に配置され、両方のアプリケーション モジュールは同じサンドボックスに配置されます。このシステムでは、2 つの制御機能の間に安全絶縁が必要ですが、これら 2 つの機能の間でセキュリティ絶縁を行う必要はありません。 |
| STACK 4 | LINK 5 | LINK5 には、UART や CAN-FD コードなどのホスト通信コードが含まれています。外部インターフェイスからのデータは、システム内の他の機能に対してセキュリティ上の脅威をもたらす可能性があるため、このモジュールは個別のサンドボックスに配置されます。 |
| STACK 5 | LINK 6 | システム内の他のすべてのモジュール間で共有される共通のコード関数が含まれています。LINK 6 は、他の LINK のアクセス保護継承 LINK (APILINK) として定義されています。共通コード モジュールは、システムの他の部分からセキュリティを分離するために、別のサンドボックスに配置されます (継承された権限を維持しながら)。共通コード LINK は独自の STACK に配置することを推奨します。 |

SysConfig には、マルチコア アプリケーションを包括的にサポートしています。内蔵のメモリ アロケータ ツールは、複数の CPU にわたってアプリケーション モジュールに関連付けられたメモリ領域を自動的に管理し、デバイス全体にわたるペリフェラルの割り当てを管理します。SysConfig ツールには共有メモリ機能も含まれており、同じ CPU 上のモジュール間または複数の CPU 上のモジュール間で共有できるメモリ領域を定義できます。

# 7 SSU の設定

## 7.1 フラッシュ SECCFG 領域

フラッシュ SECCFG 領域は、ユーザー保護ポリシー (UPP) の保存に使用されます。これは、SSU の構成とブート設定専用の、C29 アプリケーション フラッシュ バンクの特別な NONMAIN 領域です。SECCFG にプログラムされた設定は、デバイス起動時に SSU メモリ マップ レジスタにロードされます。ほとんどの場合、次のデバイス リセットまでロックされます。デバイスの各プライマリ CPU (つまり、奇数番号の C29 CPU。例：CPU1、CPU3) には、ベース セクタと予約セクタの 2 つの SECCFG セクタがあります。これらの設計は、1 つのセクタを消去して新しい構成値をプログラムし、もう 1 つがアクティブなときに実行できるようになっています。

**注**

現在アクティブな SECCFG セクタを消去して再プログラムすることはしないでください。消去とプログラミングのプロセス中にデバイス リセットが発生すると、デバイスは起動に失敗して動作不能になります。常に、代替の SECCFG セクタ アドレスに新しい設定をプログラムします。フラッシュ アドレス変換ロジックは、書き込みおよび消去動作中に、このアドレスを現在の非アクティブな SECCFG セクタに自動的に転送します。SysConfig ツールは、生成された .out ファイルの中で SECCFG イメージを代替セクタに自動的に割り当て、正しい更新手順を可能にします。

**注**

F29x デバイスには、4 種類のフラッシュ バンク モードがあります。さまざまなバンク モードの詳細については、F29x テクニカル リファレンス マニュアルを参照してください。関心のあるアプリケーションのシナリオに合わせてバンク モードを適切に構成するには、「接続」で CPU1 を右クリックし、「CCS デバッグ」ビューで「プロパティ」オプションを選択します。プロパティ ウィンドウ内で、「フラッシュ設定」ドロップダウンを選択します。「バンク モード」に移動して、目的のバンク モードを選択します。「バンク モードをプログラム」をクリックします。

SSU ユーザー保護ポリシーの整合性を保護するため、SECCFG セクタにはブート時にチェックされる CRC 値が含まれています。この CRC は、アクセス保護設定、LINK と STACK 構成、フラッシュ書き込みおよびフラッシュ消去保護、フラッシュの更新権限、デバッグ設定、ブート設定、SSU 動作モードについて説明します。デバッグ パスワードは、この CRC 計算から除外されます。

SECCFG セクタの包括的なマップは、デバイスのテクニカル リファレンス マニュアルに記載されています。

## 7.2 SSU 開発ライフサイクル

SSU は、3 つのモードのいずれかで動作するように構成できます。SSUMODE1、SSUMODE2 および SSUMODE3。これらの動作モードは、ユーザーがシステム設計に安全機能とセキュリティ機能を実装する際に、開発プロセスを容易にすることを目的としています。ユーザーが SECCFG セクタを更新するために必要な権限を持っている限り、SSU を再構成して、任意の動作モードから他の動作モードに変更できます。

テキサス・インスツルメンツから出荷するユニットは、SSUMODE1 を使用して最初に出荷を開始します。このモードでは、メモリ マップの範囲全体が LINK2 (セキュリティ ルート LINK) にマップされ、すべての LINK に対して、AP で設定可能なすべてのメモリ領域に対して完全な読み取り/書き込みアクセスが可能です。SSUMODE1 でもハード コーディングされた保護は有効ですが、すべてのコードは LINK2 として実行されるため、ユーザー コードに制限はありません。

SSUMODE2 では、AP 領域保護が強制されますが、デバッグおよびフラッシュ更新保護はディセーブルのままです。最良の結果を得るには、まず SSUMODE1 のアプリケーション機能を完全に検証し、次に SSU 設定を実装して SSUMODE2 でテストします。ランタイム安全およびセキュリティ設定の検証が完了したら、デバッグ パスワードを構成し、デバイスを SSUMODE3 に配置できます。このモードでは、デバッグ ポートがデフォルトで閉じ、認証が強制され、フラッシュ更新保護がアクティブになります。

**注**

フラッシュ書き込みおよびフラッシュ消去の保護は永続的であり、SSUMODE の設定に関係なく反転することはできません。この機能は、セキュリティ アルゴリズムを実装する目的などで、ユーザがフラッシュ コードの一部を変更不可能にする必要がある使用例を対象としています。デバイスのフラッシュの内容を最終化する前に、フラッシュ書き込みおよびフラッシュ消去保護の設定を行わないでください。

## 7.3 SysConfig ツールを使用

SysConfig は、F29x リアルタイム制御マイコンを含む TI のマイコン製品を構成するための使いやすい方法を提供するグラフィカル ユーザー インターフェイス (GUI) ツールです。SysConfig は、ペリフェラル、割り込み、ピン多重化の初期化などを含む、デバイスの初期化コードを自動的に生成します。また、SysConfig は、デバイス設定エラーを自動的に識別し、構成の問題を修正するための役立つガイダンスを提供します。また、グラフィカルな可視化機能を使用して、異なるデバイス間でアプリケーションを簡単に移植することができます。

C29 SysConfig はデバイス F29 SDK の一部として利用可能で、Code Composer Studio™ (CCS) 統合開発環境 (IDE) に組み込まれている SysConfig ツールが必要です。また、他の開発環境で使用するためのスタンドアロン ツールとしても利用可能です。

C29 SysConfig 内の SSU ツールは、F29x アプリケーション内での SSU 保護と絶縁を定義および実装するための包括的なツールです。このツールは、アプリケーション パーティションの定義、コードおよびデータ セクションの割り当て、メモリおよびペリフェラルの保護の定義、デバッグ セキュリティ オプションの実装のための使いやすいフローを提供します。SSU ツールは、これらの保護機能を実装するために必要なすべてのコード ファイルと出力ファイルを自動的に生成します。これには、SECCFG セクタ イメージ、アプリケーション リンカ コマンド ファイル、ヘッダー ファイル、必要な初期化コードが含まれます。

### *7.3.1 システム セキュリティ設定の有効化*

アプリケーションで SSU 機能をイネーブルにする最初のステップは、SysConfig に「*システム セキュリティ*」オプションを追加することです。左側のバーにあるモジュール名の右側にある (+) ボタンをクリックします。図 7-1 を参照してください。

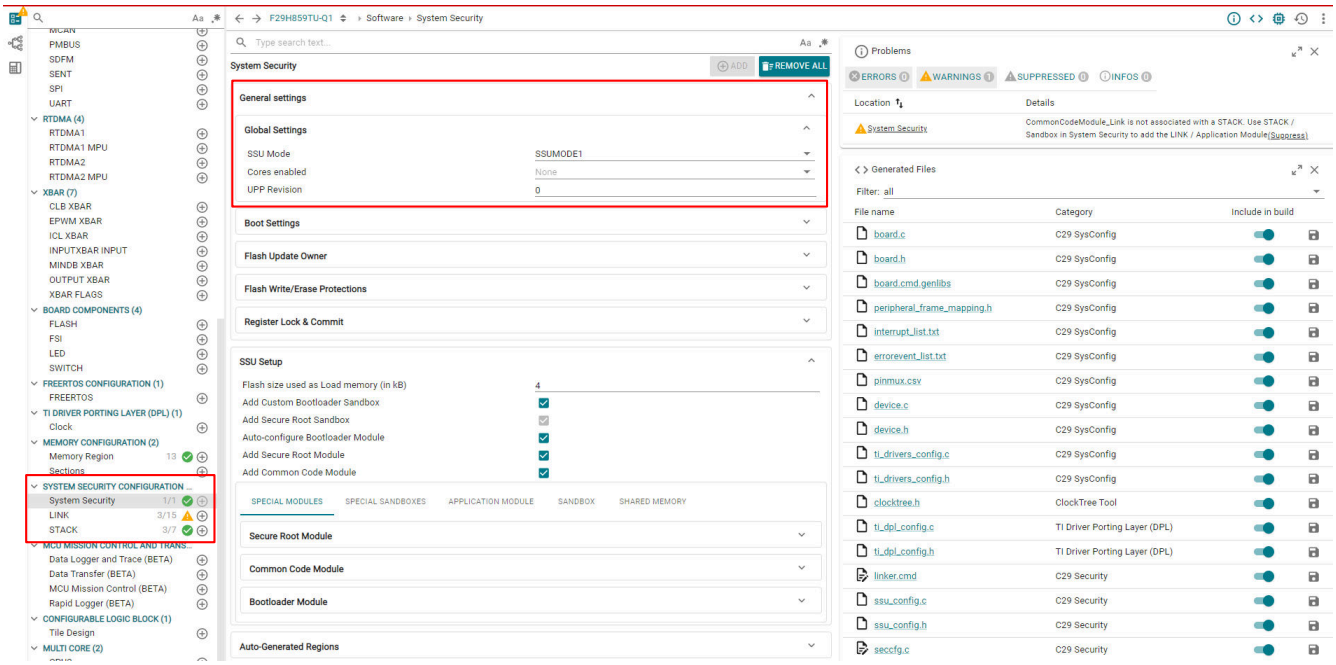


**図 7-1. システム セキュリティ設定ページ**

「*システム セキュリティ*」モジュールには、いくつかの設定オプション グループがあります。最初の設定は「*一般設定*」で、SSU 動作モード、UPP リビジョン番号、および有効化されたコアの設定が含まれています。

「*システム セキュリティ*」モジュールには、デバイスのブート モードの選択、フラッシュ更新保護の設定、デバッグ パスワード (SSU モードとして SSUMODE 3 が選択されている場合のみ表示可能)、SSU レジスタのロックに関する構成オプションも含まれています。デバッグ パスワードや Flash Update オーナー設定など、これらの設定の一部では、SSUMODE3 操作を有効にする必要があります。

SysConfig に「*システム セキュリティ*」モジュールが追加されると、LINK、STACK、メモリ領域モジュールなど、他のモジュールが自動的に追加されます。LINK モジュールと STACK モジュールを変更する必要はありません。すべての設定は、「*システム セキュリティ*」モジュールを使用して行うことができます。LINK モジュールと STACK モジュールは、「*システム セキュリティ*」モジュールで選択された LINK 構成と STACK 構成の概要を提供するだけです。「*システム セキュリティ*」モジュールが追加されると、リンカ コマンド ファイル (.cmd) も SysConfig から自動的に生成されます。既存のプロジェクトを使用している場合は、競合を防ぐために、プロジェクト ビルドから .cmd ファイルを除外します。

**注**

リンカ コマンド ファイルに追加の変更や追加を必要とする場合は、SysConfig のメモリ構成セクションにある「*メモリ領域*」と「*セクション*」のモジュールを使用します。

**注**

SysConfig で「*システム セキュリティ*」モジュールを最初に追加すると、CommonCodeModule_Link が「*システム セキュリティ*」モジュールの STACK /サンドボックスに関連付けられていないという警告が表示されます。この警告を解決するには、「*システム セキュリティ*」モジュールの SSU セットアップ内にあるサンドボックスタブに移動し、CommonCodeModule_LINK がモジュールの 1 つとして選択されたサンドボックスを追加します。

### *7.3.2 アプリケーション モジュールの設定*

SysConfig を使用すると、AP 範囲を作成し、オブジェクト ファイル、ライブラリ、入力セクションに基づいて LINK 権限を構成することができます。新しいアプリケーション モジュールを作成すると、SysConfig によって自動的に LINK が作成され、AP 領域の標準セットが作成されます。

- フラッシュから実行されるコード領域 (ModuleName_codeAPR_Flash)
- [オプション] RAM から実行されるコード領域 (ModuleName_codeAPR_RAM)
- RAM 内の可変データ領域 (ModuleName_dataAPR_RW)
- オプションで RAM (ModuleName_dataAPR_RO) に配置できる読み取り専用データ領域

ユーザーは、標準領域に加えて、「*カスタム セクションを使用*」チェックボックスをオンにして、追加するカスタム セクションを指定することで、アプリケーション モジュールに関連付けるカスタム セクション名を構成できます。SysConfig は、定義されたすべての AP 領域を SSU 設定に追加し、関連 LINK を各領域に適切な権限を持つように構成します。さらに、各 AP 領域のリンカ コマンド ファイルに出力セクションが作成され、設定どおりに入力セクションをそのメモリ領域に配置するようリンカに指示します。

コード関数とデータをアプリケーション モジュールに関連付けるには、ファイル拡張子を差し引いたファイル名を「*含めるファイル*」入力フィールドに追加するだけです。ライブラリは、対応する入力フィールド (ライブラリ ファイル拡張子を含む) を編集してモジュールに追加することもできます。ライブラリから特定のオブジェクトを選択するには、myLibrary.lib<myFuncs1.o> などのリンカ コマンド ファイル構文を使用します。以下を実行する必要があります。SysConfig は、各オブジェクトの .text、.bss、.data、.rodata、.const の各入力セクションを、リンカ コマンド ファイル内の対応する出力セクションに自動的に割り当てます。

モジュールにメモリを割り当てるには、APR タイプごとに必要なメモリ量を指定するだけです (フラッシュ コード、RAM コード、RW データ、RO データ)。SysConfig は、メモリ内の AP 領域を自動的に配列し、最小のウェイト状態に必要な最適なメモリ タイプを選択します。アプリケーション モジュールがパフォーマンス要件を満たすためにフラッシュではなく RAM から実行する必要がある場合は、「*RAM に place.text セクションを配置*」チェックボックスを選択します。このチェックボックスを選択すると、SysConfig によって新しい RAM コード領域が作成され、ブート時にフラッシュから関連するコードをロードして RAM から実行するようにリンカ コマンド ファイルを設定します。必要に応じて、ゼロウェイト状態アクセスのためにルックアップ テーブルなどの読み取り専用データまたは定数データを RAM に配置することもできます。

コードおよびデータ メモリ領域に加えて、SysConfig で構成されている既存のペリフェラルも、各アプリケーション モジュールに自動割り当てできます。指定されたペリフェラルへの読み取り/書き込みアクセスと読み取り専用アクセスのどちらかを有効にするための 2 つのドロップダウン選択フィールドが用意されています。

ペリフェラル割り込みは、含まれるフィールドを使用して簡単に追加できます。このオプションは、選択されたペリフェラル割り込みに正しい実行 LINK を割り当てるように PIPE モジュールを設定します。

**注**

INT は、INTSP レジスタで設定された指定された STACK を使用します。これはすべての INT に適用可能な設定です。したがって、1 つ以上の INT を使用する場合、ISR は、INTSP レジスタで選択された STACK と一致する同じ SSU STACK に配置する必要があります。RTINT を使用する場合に制限はありません。

SysConfig の「メモリ構成」セクションにある「*メモリ領域*」モジュールには、現在のアプリケーション モジュール用に作成された各 AP 領域の詳細が表示されます。このフレームには、いくつかの追加の設定オプションもあります。

- **0 WS メモリのみを使用**：RAM コードをゼロウェイト状態 RAM に制限します。
- **等価 RTDMA MPU 領域の作成**：DMA 転送の開始アドレスと終了アドレスが同じ MPU 領域を作成します。
- **他のコアと共有**：複数の CPU で 1 つのメモリ領域を使用できるようにします。

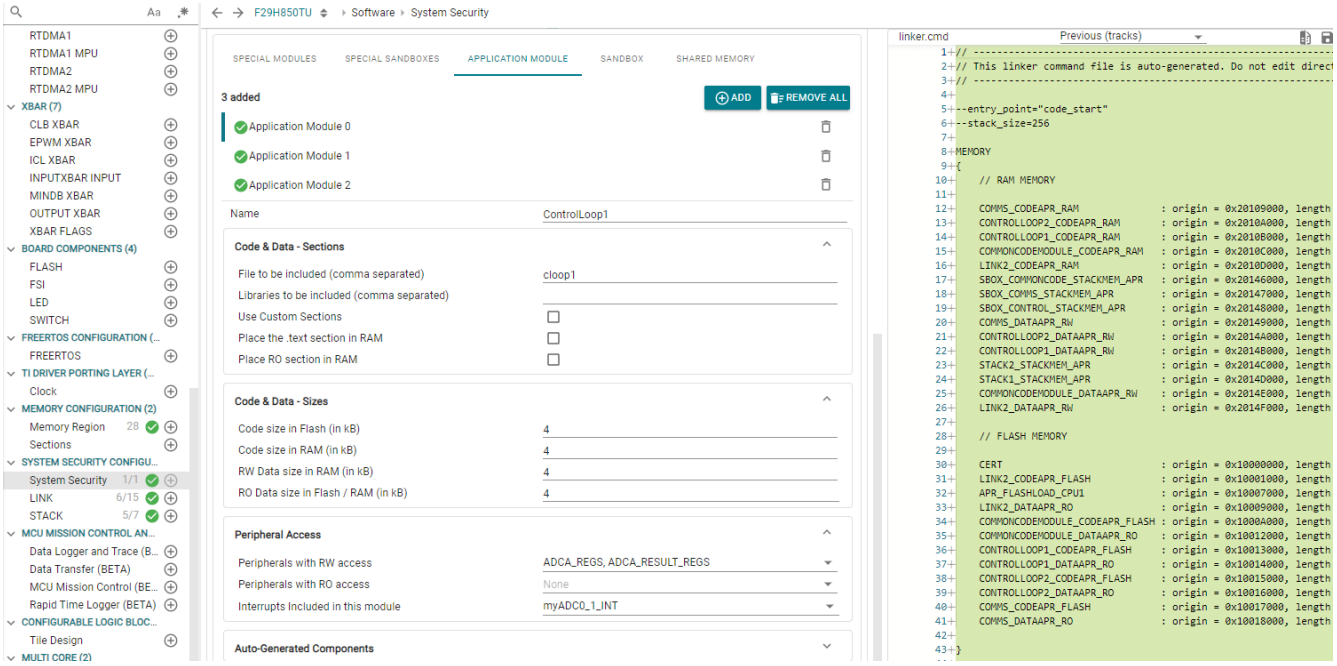


**図 7-2. アプリケーション モジュールの設定例**

### *7.3.3 特殊モジュールの設定*

SysConfig では、システム内で事前定義された機能を備えたモジュールを直接構成するオプションを提供します。

- LINK2 – システムのセキュリティ ルート LINK、
- LINK1 – ブートローダー LINK、
- アクセス保護の継承のための共通コード LINK

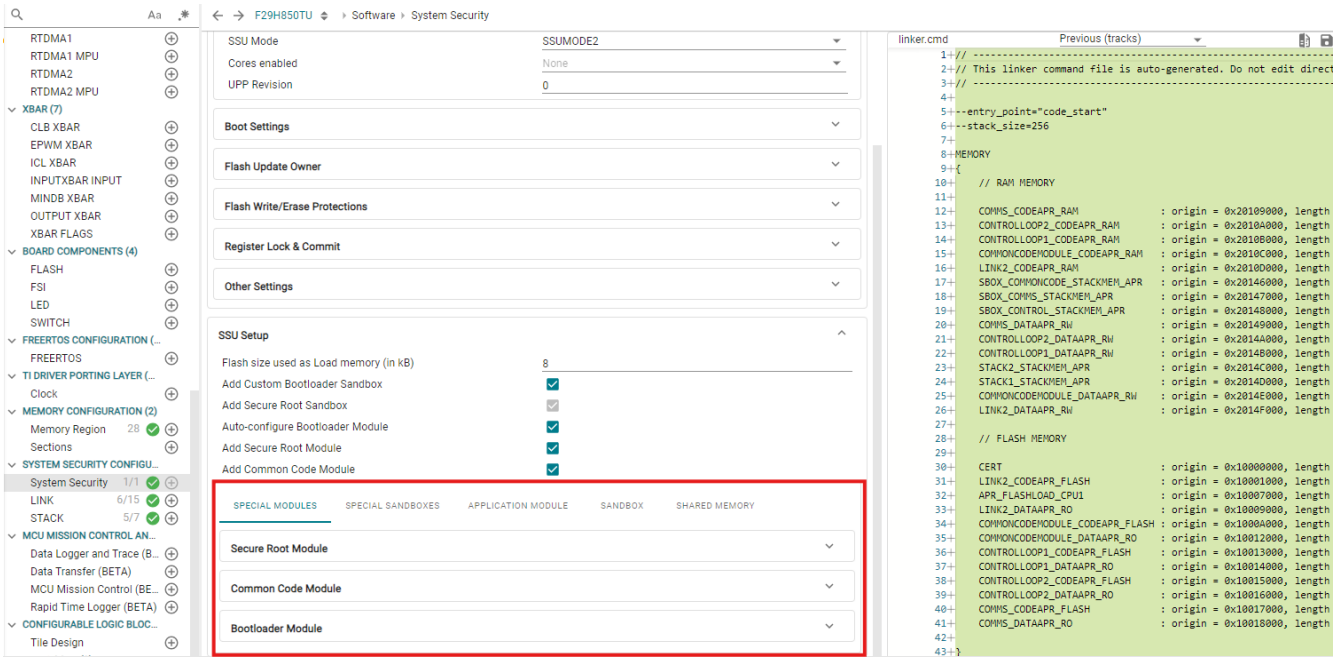


**図 7-3. 特別なモジュールの設定例**

特殊なモジュールを有効にした後、ドロップダウン ボックスを展開して各モジュールを設定できます。以降のセクションでは、特殊モジュールについて詳しく説明します。

#### 7.3.3.1 LINK2 構成

LINK2 は、各 CPU 上で最もセキュアな LINK です。LINK2 は、セキュアな CPU レジスタにアクセスしたり、監督タスクを実行したりする権限を持っています。ほとんどの場合、RTOS レイヤ機能は LINK2 に配置されています。特に CPU1.LINK2 には、次のようなデバイス全体の特別な昇格特権があります。

- デバイス構成レジスタに書き込む機能
- 特定の SSU レジスタに書き込む機能
- MPU 構成を含め RTDMA を構成する機能

LINK2 に、システムの初期設定とボードの設定、およびオペレーティング システム機能の実行を担当するすべてのコード セクションとデータ セクションを配置します。表 7-1 に、LINK2 で必須のペリフェラル レジスタと、それらのレジスタが必須である理由を示します。

**表 7-1. LINK2 の必須ペリフェラル アクセス保護**

| ペリフェラル レジスタ ベース | 理由 |
|---|---|
| CPU_SYS_REGS | CPU システム レジスタ |
| PER_CFG_REGS_WD_REGS | ウォッチドッグ レジスタの設定 |
| CPUTIMER2_REGS | ドライバ移植レイヤ (DPL) 用の CPU タイマ 2 レジスタ |
| DCC1_REGS | クロック モニタ |
| ERR_AGG_REGS | エラー/NMI 処理 |
| ESM_CPU1_REGS | エラー/NMI 処理 |
| ESM_CPU2_REGS | エラー/NMI 処理 |
| ESM_CPU3_REGS | エラー/NMI 処理 |
| ESM_SYS_REGS | エラー/NMI 処理 |
| ESM_SAFETY_AGG_REGS | エラー/NMI 処理 |

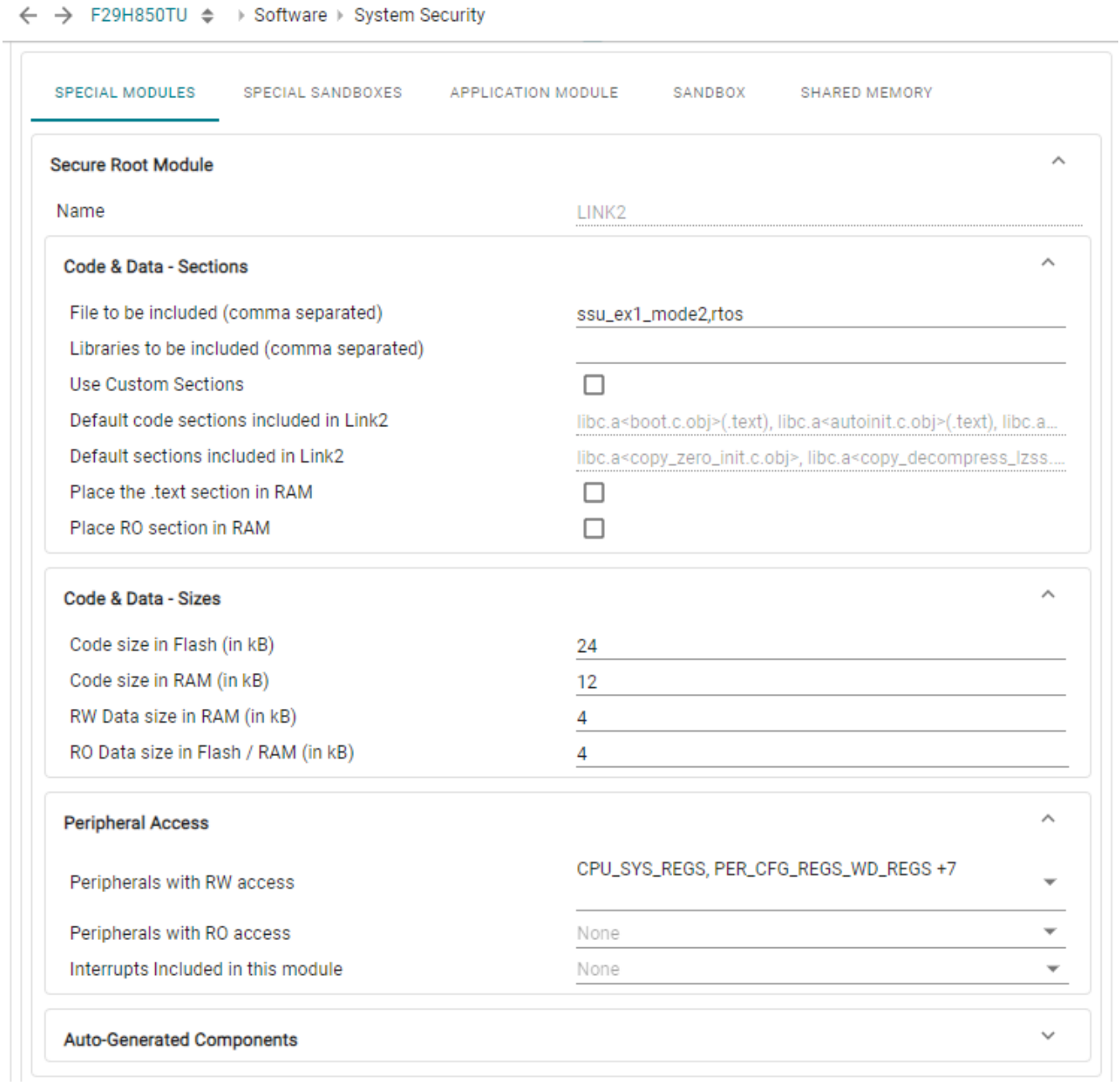


**図 7-4. LINK2 の構成例**

#### 7.3.3.2 LINK1 構成

CPU1.LINK1 CPU1.LINK1 は主にブートローダーで使用され、特定のシステム構成レジスタに書き込む機能など、ブートローダー機能をサポートするための追加のハードコードされた権限があります。LINK1 は、従来のユーザー LINK としても使用できますが、ブートローダーに関連しないコードやデータを LINK1 に追加することは、他のすべての LINK がすでに使用されている場合の最後の手段として推奨されません。

いずれかのペリフェラル ブート モードを「*ブート設定*」グループで構成すると、SysConfig は、そのブートモードが機能するのに必要な各ペリフェラル (CAN、UART、SPI など) にアクセスできるように LINK1 を自動的に構成します。これらのペリフェラルは、IPC や HSM メールボックスなど、デバイスのブートに常に必要な特定のペリフェラルに加えて、SysConfig によって LINK1 モジュールに自動的に追加されます。表 7-2 に、デバイス ブートの一部として LINK1 に必要とされる必須ペリフェラルの完全なリストを示します。

**表 7-2. LINK1 の必須ペリフェラル アクセス保護**

| ペリフェラル レジスタ ベース | 理由 |
|---|---|
| PER_CFG_REGS_WD_REGS | ウォッチドッグ レジスタの設定 |
| IPC_CPU1_SEND_REGS_HSM_CH0 | デバイス認証中の HSM とのハンドシェイク |
| IPC_CPU1_SEND_REGS_HSM_CH1 | デバイス認証中の HSM とのハンドシェイク |
| HSM_MAILBOX | IPC 用の HSM メールボックス |
| GPIO_DATA_REGS | ブート モードの選択およびエラー ステータス ピン構成 |
| MCANA_MSGRAM | ペリフェラル ブート モード |
| MCANA_REGS | ペリフェラル ブート モード |
| UARTA_REGS | ペリフェラル ブート モード |
| DCC1_REGS | クロック モニタ |
| ERR_AGG_REGS | エラー/NMI 処理 |
| ESM_CPU1_REGS | エラー/NMI 処理 |

#### 7.3.3.3 共通コード リンク設定

### *7.3.4 サンドボックスの定義*

SysConfig 内のサンドボックスを使用して、アプリケーションの他の部分からセキュリティを分離する必要があるアプリケーション モジュールのグループを定義します。各サンドボックスは SSU STACK に関連付けられており、少なくとも 1 つのアプリケーション モジュールと STACK メモリ AP 範囲が含まれています。サンドボックス内のアプリケーション モジュールに関連付けられているすべての LINK には、サンドボックス STACK メモリへの読み取り/書き込みアクセス権があります。その他のすべての LINK にはアクセス権がありません。各サンドボックスは 1 つのデバッグ ゾーンに関連付けられています。

SysConfig は、各サンドボックスのリンカ コマンド ファイル内で SECURE_GROUP を定義します。この設定により、リンカは他の STACK からサンドボックス STACK へのすべての関数コールに対して、保護されたコールを必要とします。デフォルトでは、SECURE_GROUP への保護されていないコールは、リンカにエラーを発生させます。SysConfig には、保護されたコール要件を満たすために、トランポリンとランディング コールを自動生成するオプションがあります。このオプションを有効にするときは、信頼できないコードに対して不要なクロス STACK トランポリンが生成されていないことを確認するために、出力リンカ マップ ファイルを確認してください。

**注**

クロス STACK トランポリンは、STACK メモリとの間で CPU レジスタを保存および復元する必要があるため、レイテンシーを増加させる可能性があり、アプリケーションのパフォーマンスに影響を与える可能性があります。最高の性能を得るためには、関数定義 __attribute__((c29_protected_call)) を追加して、保護された関数コールをアプリケーション コードに直接実装します。

**注**

STACK1 設定には、「*特殊モジュール*」タブからアクセスできます。

### *7.3.5 共有メモリの追加*

共有メモリ領域は、複数のアプリケーション モジュールからアクセスできる特別なアクセス保護領域 (APR) です。SysConfig では、「*システム セキュリティ*」ページの「*共有メモリ*」タブを選択し、「*追加*」ボタンをクリックすると、共有メモリを追加できます。複数の共有メモリを追加できますが、CPU で使用可能な APR の総数に制限されます（各種アプリケーションモジュール用に定義された APR を含む）。各共有メモリにソース ファイルを含めることができます。SysConfig は、構成どおりに、これらのファイルから .bss、.data、.const、.rodata セクションを追加します。カスタム セクション名を含めるように定義することもできます。

共有メモリごとに、読み取り専用権限を必要とするアプリケーション モジュールと、書き込み権限を必要とするモジュールを選択できます。SysConfig は、各モジュールの LINK に対して定義されている APR 権限を自動的に設定します。

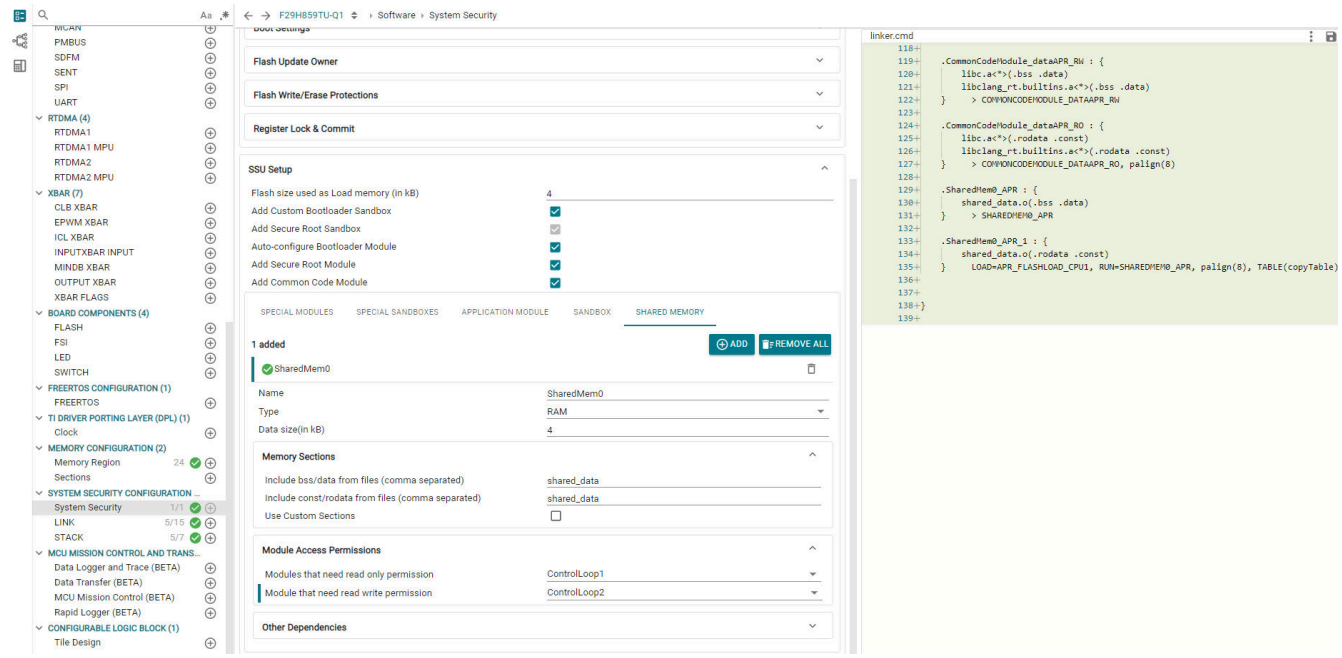


**図 7-5. 共有メモリの設定例**

# 8 デバッグ許可

## 8.1 パスワードベースのロック解除

「*システム セキュリティ*」モジュール内で、SSUMODE を選択できます。図 8-1 に示すように、SSUMODE 3 ではセキュアデバッグとファームウェアの更新がサポートされています。そのため、パスワード入力と一致に基づいてデバッグ許可をイネーブルにするには、最初に SSUMODE3 を選択します。SSUMODE3 を選択すると、「デバッグ アクセス (SSU モード 3 のみ)」という名前のセクションが表示されます。このセクションでは、部分デバッグ用と完全デバッグ用のパスワードを各ゾーンに定義できます。C29 デバッグにはパスワードが 1 つしかないため、エントリは 1 つだけです。これらのパスワードはブート時に SECCFG にロードされます。

**注**

SSUMODE3 を使用する場合は、正しいアクセス許可が設定されていることを確認してください。SSUMODE2 を使用しているときに、ユーザー テスト権限を最初に設定することをお勧めします。適切な APR 構成なしでデバイスが SSUMODE3 に移行した場合、保護が適用されているため、コードは正しく実行されません。

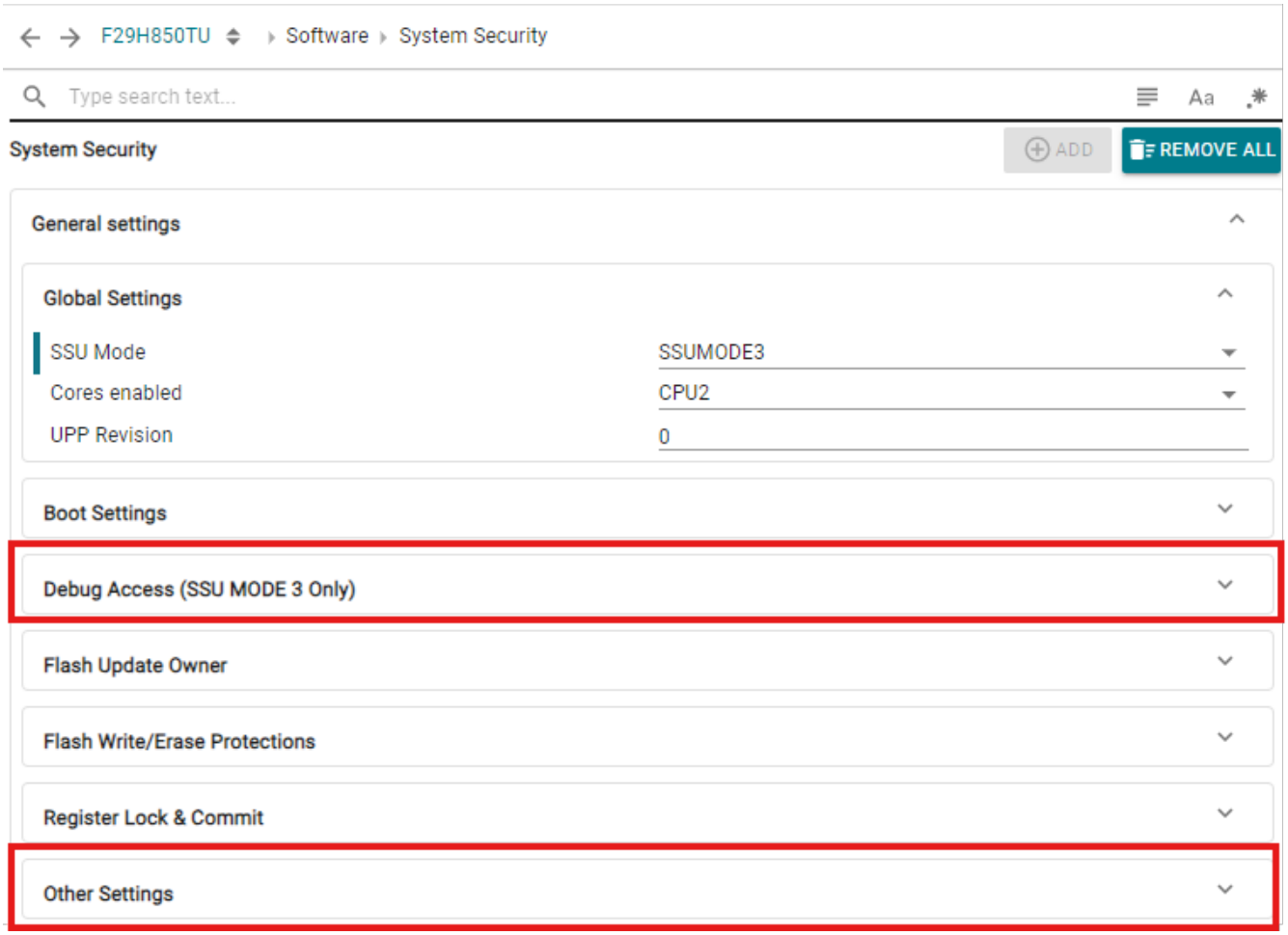


**図 8-1. SSU モード 3 の追加設定**

デバイスが SSUMODE3 になると、すべてのデバッグ アクセスは終了します。さまざまなゾーンの適切なデバッグを開くには、SysConfig で設定されたパスワードを SEC-AP インターフェイス経由でデバイスにスキャンする必要があります。そのためには、SysConfig 内で使用されたパスワードを使用して SEC-AP gel ファイルを変更します。CCS で SEC-AP gel ファイルにアクセスするには、以下の手順を使用します。

- 「接続」ペインの「CCS デバッグ」ビュー内で表示可能なコアのいずれかを右クリックし、「すべてのコアを表示」を選択します。

- C29 SEC-AP コアに接続します
- CCS ツールバー メニューの「表示」に移動し、「GEL ファイル」を選択します。
- 提供された SEC-AP gel ファイルを開きます
- SysConfig で説明したように、さまざまなゾーンのパスワードを変更します。128 ビット パスワードの書き込み順序は次のとおりです。[31:0] ビット、[63:32] ビット、[95:64] ビット、[127:96] ビット。
- SECAP インタフェースから切断し、再接続します。接続時にパスワードがスキャンされます。したがって、SECAP インターフェイスを介してスキャンされたパスワードが、SysConfig を使用して構成されたパスワードに一致する場合、C29 とゾーンが開きます。

デバッグが正常にオープンになると、該当する C29 コアへの接続が可能です。

# 9 SSU のデバッグ

## 9.1 ビルド エラーのデバッグ

**表 9-1. 可能なビルド エラー**

| エラーの例 | 説明 | 分解能 |
|---|---|---|
| 「syscfg/linker.cmd」、73 行目：エラー #10099-D：プログラムが利用可能なメモリに収まらないか、セクションにこのセクションで生成できないトランポリンを必要とするコール サイトが含まれているか、セクションにパディングされた関数が含まれています。アライメントを伴う配置は、セクション「.LINK2_codeAPR_Flash」サイズ 0x5052 ページ 0 に対して失敗します。利用可能なメモリ範囲：LINK2_CODEAPR_FLASH サイズ：0x2000 不使用：0x1fe0 最大ホール：0x1fe0 | 定義された APR サイズが十分ではありません。 | いずれかの APR タイプ (フラッシュ、RAM、RW、または RO データ) に割り当てられたメモリが十分でない場合、プロジェクトのビルド時にビルド エラーが発生します。APR の正しいサイズを決定するには、指定されたサイズ (0x5052) を 10 進数 (20,562) に変換します。これは、APR の合計サイズが約 20K であることを示しています。APR は 4KB にバインドされているため、この特殊なケースでは、APR サイズを 24KB に設定する必要があります。 |
| エラー #10483-D：出力セクション「.comms_Module_codeAPR_Flash」から保護されていないシンボル「UART_writeCharArray」への保護されていないコールは許可されていません：SECURE_GROUP の不一致。コールは SECURE_GROUP「STACK2_STACK」にあり、着信側は SECURE_GROUP「sbox_CommonCode_STACK_COMMONCODE」にあります。 | クロス STACK コールには、適切なエントリおよび終了命令が含まれていません。 | クロス STACK コールには、常に適切なエントリおよび終了命令が含まれている必要があります。SSU ツールには、保護されていないコールの処理方法を指定するオプションがあります。次の 3 つのモードがあります。<br>• リンカ エラーを生成：適切な入力および終了命令がない場合はエラーがスローされます (ユーザーが設定する必要があります)。<br>• リンカ トランポリンとランディング パッドを追加します (安全)。コンパイラは、選択した STACK へのすべてのコールに適切なエントリ命令と終了命令を挿入します。レジスタは、クロス STACK コールを通じて保持されます。<br>• リンカ トランポリンとランディング パッドを追加します (安全ではないが高速)。コンパイラは、選択した STACK へのすべてのコールに適切なエントリ命令と終了命令を挿入します。レジスタは、クロス STACK コール間で保持されません。<br>適切なオプションを選択して、クロス STACK のエントリおよび終了命令を処理します。 |

## 9.2 ランタイム エラーのデバッグ

SSU や他のペリフェラルからのさまざまなエラーをキャプチャし、F29x のエラー アグリゲータ モジュール (EAM) とエラー通知モジュール (ESM) により記録します。CCS の EAM および ESM から発生するエラーを表示する方法の詳細については、『F29x エラー処理およびデバッグ ガイド』アプリケーション レポートを参照してください。

**表 9-2. 可能な SSU ランタイム エラー**

| エラーの例 | 説明 | 分解能 |
|---|---|---|
| C29xx_CPU1：フラッシュ プログラミング中のエラー。アドレス 0x10D85000、FMSTAT (一部のデバイスでは STATCMD) 0x00000000、値 0x00000101<br>C29xx_CPU1：Filer Loader：メモリの書き込みに失敗：不明なエラー<br>C29xx_CPU1：GEL：ファイル： C:\Users…..ssu_ex1_mode2.out: ロードに失敗しました。 | SECCFG をプログラムするためにフラッシュ設定が正しく設定されていない場合、以下のエラーが見られます。 | SECCFG メモリを消去できるようにするには、「フラッシュ設定」の「*メイン以外の消去設定*」セクション内の「フラッシュ メモリにデータをロードする前に、メイン以外のフラッシュ消去を許可」チェックボックスにチェックが入っていることを確認します。その後、プログラムを再フラッシュします<br>**注**<br>SECCFG プログラミングを操作しない場合、この設定が**選択されていない**ことを確認します |

**表 9-2. 可能な SSU ランタイム エラー (続き)**

| エラーの例 | 説明 | 分解能 |
|---|---|---|
| CPU1_DW Errors (HP Error Addr = 0x60070018, LP Error Addr = 0x00000000, PC = 0x10010646)<br>セキュリティ違反 | 以下のエラーは、プログラム カウンタ (PC) アドレス 0x10010646 にある命令がアドレス 0x60070018 にデータを書き込もうとしたが、それを行うための十分な権限がないためにデータを書き込めなかったことを示しています。 | この問題をデバッグするには、CCS Disassembly ビュー (*View -> Disassembly*) で PC アドレスを検索します。これは、プログラム カウンタの位置を示しています。これにより、書き込みアクセスが禁止されていたコードの実行 (つまり、その関数と行) に関する洞察が得られます。この例では、プログラム カウンタは LINK4 内の「update_PSFB()」機能に配置されていました。<br>次に、CCS (View -> Memory) でメモリ ブラウザを開き、アドレス 0x60070018 を入力します。これにより、命令が書き込もうとしていた内容がメモリに表示されます。この場合、0x60070018 は UARTA レジスタに対応します。F29x DS のメモリ マップを使用して、提供されたエラー アドレスをレジスタ セットと関連付けることもできます。この場合、ユーザーはメモリ マップで 0x6007_0000 を調べることができます。<br>したがって、「update_PSFB()」関数内のコードを UARTA レジスタに書き込むために、LINK4 は UARTA ペリフェラルへの R/W アクセスを行う必要があります。<br>**注**<br>CPU1 エラーには、PR、DR1、DR2 など多くの種類があります。これらの個別のエラーの意味の詳細については、TRM の「エラー アグリゲータ」の章または『F29x エラー処理およびデバッグ ガイド』アプリケーション レポートを参照してください。 |
| SSU Errors (HP Error Addr = 0x3008000C, LP Error Addr = 0x00000000, PC = 0x00000000)<br>CPU1_SSU_MMR_ACCESS_ERROR | このエラーは、SSU が SSU メモリ マップ レジスタ (MMR) のいずれかにアクセスできなかったことを示しています。 | この問題をデバッグするには、最初は CCS メモリブラウザ (View -> Memory) で提供されているアドレスを検索します。アドレス 0x3008000C は LINK2_AP_OVERRIDE に対応します。CCS 内の「検索」機能を使用して、プロジェクト内で「SSU_enableLink2APOverride()」が呼び出されたすべてのインスタンスを検索しました。この例では、関数「SSU_enableLink2APOverride()」が LINK2 の外部で使用されており、このエラーが発生しています。他の SSU MMR アクセス エラーが発生した場合は、同様の方法を使用します。 |

**表 9-2. 可能な SSU ランタイム エラー (続き)**

| エラーの例 | 説明 | 分解能 |
|---|---|---|
| フラッシュ中 -「警告:ターゲット CPU が持続的なフォルト状態で固着している可能性があります。」 | デバイスは SSUMODE2 にあり、メモリ保護機能を実装しています。 | SSU が安全およびセキュリティ保護を有効にした状態で使用する想定ではない場合、SSU モードを SSU モード 1 に変更する必要があります。これを行うには、SECCFG を再プログラムしてデバイスを SSUMODE1 にし、BANKMGMT の BANKMODE0 に再プログラムします。<br>• 検出されたデバイスにデフォルトの seccfg プログラムをロードします。デフォルトの seccfg プログラムは、次のパスにあります。C:\ti\<f29h85x-sdk_version_x>\source\defseccfgbin<br>• EVM または HW をリセット<br>• ターゲットに接続し、SSUGENREGS レジスタで SSUMODE = 0x30 であることを確認します。<br>• フラッシュ設定で、フラッシュ BANKMODE を BANKMODE 0 にプログラムするか、または目的の BANKMODE が設定されていることを確認します。<br>• バンクモードを変更した場合は、CCS でターゲットから切断し、EVM で XRSN を発行します (controlSOM には XRSN ボタンがあります)。<br>• ターゲットに接続し、SSUGENREGS で BANKMODE = 0x3 または目的の BANKMODE が正しく設定されていることを確認します。 |

# 10 SSU に関するよくある質問 (FAQ)

**表 10-1. SSU に関する質問と回答**

| # | 質問 | 回答 |
|---|---|---|
| 1 | 実行時に APR を変更する場合、APR を変更する前に APR を無効にする必要がありますか。 | いいえ、有効になっている間は変更できます。 |
| 2 | LINK2 AP オーバーライド機能の目的は何ですか。 | LINK2 は、LINK2 にこれらの権限が与えられていない場合でも、有効または設定されているすべての APR に対する読み取り/書き込み (R/W) アクセスを取得します。 |
| 3 | コードとデータの両方の APR の範囲を変更する場合、LINKID は必須フィールドですか。 | いいえ。LINKID は XE ビットが設定されている場合のみ必要です (コード APR)。 |
| 4 | SSU APR レジスタ内の START ADDRESS、END ADDRESS、および LINKID フィールドをランタイム中に動的に変更できますか。 | はい、実行時に更新できます。これらのレジスタを変更するためのアクセスは、LINK2 のみです。実行時に、LINKx_CFG と STACKy_CFG を変更することはできません。 |
| 5 | モジュール設定の一部としてファイルが含まれている場合、ファイル全体が APR を使用してキャプチャされますか。 | はい。SysConfig の SSU ツールを使用して、その LINK の APR の一部として、ファイル内のコードと変数が設定されます。 |

## 11 まとめ

C29x SSU はリアルタイム制御アプリケーション用の高度な安全性およびセキュリティ機能を可能にします。たとえば、コンテキストに応じたメモリとペリフェラルの保護機能 (タスク間または割り込みを処理する際にソフトウェアのオーバーヘッドを排除) し、高度なデバッグ セキュリティ オプションなどです。使いやすい SysConfig ツールを使用することで、システム設計者はアプリケーションを複数のパーティションに分割し、安全とセキュリティを分離して、オブジェクト ファイル、ライブラリ、ペリフェラルをこれらの各パーティションに自動的に割り当てることができます。共有メモリ リソースは、複数のアプリケーションモジュールで使用するように定義することもできます。SysConfig は、デバイス全体のメモリのさまざまなアプリケーション パーティションへの割り当てを自動的に処理し、マルチコア構成を完全にサポートしています。このツールは、目的の保護ポリシーを実装するために必要なすべての出力ファイルを生成します。この出力ファイルは、アプリケーション .out イメージに組み込むことができます。

## 12 参考資料

1. テキサス・インスツルメンツ、『*C2000™ SysConfig アプリケーション ノート*』
2. テキサス・インスツルメンツ、『*TI Resource Explorer : 『C2000™ リアルタイム マイコン*
3. テキサス インスツルメンツ、『Code Composer Studio (CCS) IDE』
   - TI のマイコンと組込みプロセッサ ポートフォリオをサポートする統合開発環境（IDE）
   - SysConfig ツールは CCS に統合済み (組込み SysConfig サポート)
4. テキサス・インスツルメンツ、『SysConfig スタンドアロン バージョン』
   - SysConfig スタンドアロン バージョンは、SysConfig ツールを内蔵していない他の IDE と組み合わせて使用することができます

## 13 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, 2024 to OCTOBER 31, 2025 (from Revision * (November 2024) to Revision A (October 2025))** **Page**

# 重要なお知らせと免責事項

テキサス・インスツルメンツは、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、テキサス・インスツルメンツ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 テキサス・インスツルメンツ製品の選定、(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 テキサス・インスツルメンツ製品を使用するアプリケーションの開発の目的でのみ、テキサス・インスツルメンツ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。テキサス・インスツルメンツ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、テキサス・インスツルメンツおよびその代理人を完全に補償するものとし、テキサス・インスツルメンツは一切の責任を拒否します。

テキサス・インスツルメンツの製品は、テキサス・インスツルメンツの販売条件、または ti.com やかかる テキサス・インスツルメンツ製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。テキサス・インスツルメンツがこれらのリソースを提供することは、適用されるテキサス・インスツルメンツ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。

お客様がいかなる追加条項または代替条項を提案した場合でも、テキサス・インスツルメンツはそれらに異議を唱え、拒否します。

郵送先住所:Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

# 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