参考資料

Application Note

F29x エラー処理およびデバッグ ガイド

Prarthan Bhatt

概要

このアプリケーション ノートでは、エラー処理アーキテクチャの概要と、エラーイベントを効率的にデバッグする方法に関するガイダンスを提供することに重点を置いています。F29x デバイスのアーキテクチャは、さまざまなエンド アプリケーションにおいて機能安全に重点を置いた体系的なエラー処理の管理を提供します。エラー アグリゲータ モジュール (EAM) およびエラー通知モジュール (ESM) は、デバイス全体のすべてのエラー イベントについてエラー集約、ログ記録、および構成可能な応答を提供します。このアプリケーション ノートでは、EAM および ESM が提供するツールとエラー ログを使用してエラー ソースをデバッグする方法について説明します。

目次

商標

English Document: SDAA104

1 はじめに

機能安全が重要となる開発においては、体系的故障とランダム故障の両方を管理する必要があります。F29x デバイスのアーキテクチャにはハードウェア安全機構が組み込まれており、デバイス全体にわたるすべてのエラー イベントを体系的に管理します。まず、高レベルでエラー処理アーキテクチャを理解し、その後、エラー ログの解釈方法や各エラー イベントに対する応答の設定方法を詳しく確認します。

このアプリケーション ノートは、まずエラー処理アーキテクチャの概要を示し、その後、EAM および ESM の機能やツールについて例を用いて詳しく説明し、最後によくある質問とエラー デバッグのヒントをまとめます。

表 1-1 に、このアプリケーション ノートで使用されている用語と略語を示します。

表 1-1. 使用する用語と略語、およびそれらの説明

用語または略語	説明
系統的誤差	決定論的原因故障は、設計、開発、製造プロセスの不備に起因し、通常は開発プロセスのギャップから生じます。詳細については、ISO 26262 などの安全規格を参照してください
ランダム故障	ランダム故障とは、部品の動作寿命中に予測不能なタイミングで発生するハードウェアの故障のことです。体系的故障が決定論的で設計上の欠陥に起因するのとは異なり、ランダム故障は統計的性質を持ち、確率的解析によって管理されます。
システム アドレス	各 MCU には固有のメモリ マップがあり、各部品のアドレス範囲を定義しています。システム アドレスは、マップ内にある特定のリソースに対応するアドレスです
EAM	エラー アグリゲータ モジュール
ESM	エラー通知モジュール
NMI	ノンマスカブル割り込み
CCS	コード コンポーザー スタジオ

2 エラー処理アーキテクチャの概要

図 2-1 に、エラー処理の概要を示します。エラーは、ペリフェラル、モジュール、メモリ、インターコネクト、あるいは処理ユニットといったソースで検出され、EAM に伝達されて集約された後、ESM に渡され、デバイス内でユーザーが構成可能なエラーレスポンスへとつながります。エラー集約とロギングが必要な重要なデバイス エラー イベントは EAM を通過しますが、他のすべてのデバイス エラー イベントはエラー ソースから ESM に直接渡され、*F29x テクニカル リファレンス マニュアル*の ESM の章の「エラー イベント」の表に一覧表示されています。

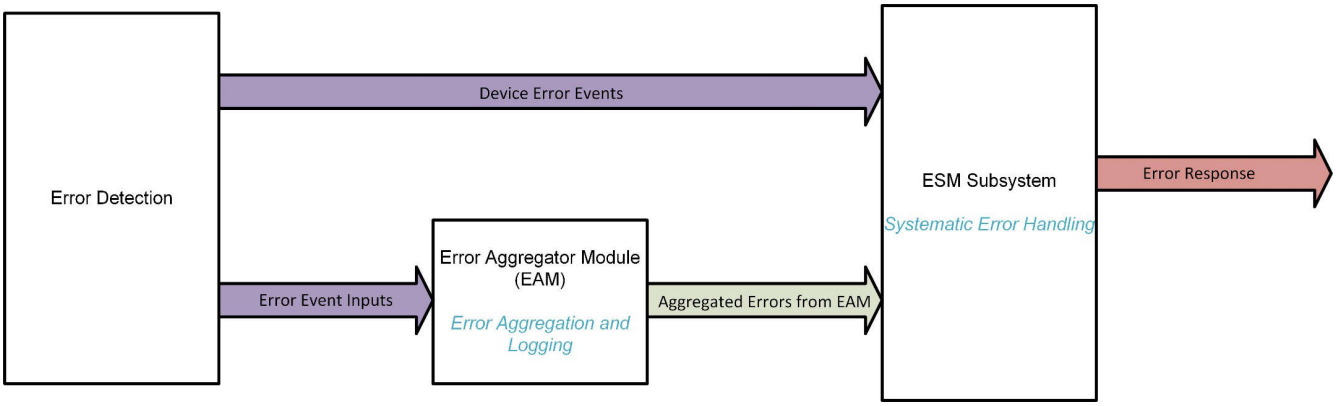

図 2-1. デバイス エラー処理アーキテクチャ

3 使用例の概要

EAM と ESM の機能については、以下のセクションで解説します。例を用い、それぞれの ESM 部品と EAM 部品を個別に取り上げて説明します。このアプリケーション ノートでは、マルチコアの例 (CPU1 および CPU3) である F29x SDK の *F29 SDK ESM マルチコアの例*を使用しています。CPU3 のアプリケーション コードが M0RAM の領域に書き込みを行ったことで、CPU3 DW バスのセキュリティ違反エラーが発生します。本例では、このエラーを EAM と ESM を用いてどのように処理するかを示します。

このアプリケーション ノートでは、この例を取り上げ、ツールを使用してエラーをデバッグする方法を示します。

4 エラー アグリゲータの概要

エラー アグリゲータ モジュール (EAM) は、**ESM** と**重要なモジュール**の間のインターフェイスであり、C29x CPU、PIPE、RTDMA、メモリ コントローラ、ペリフェラル ブリッジ、読み取りインターフェイスなどのエラーを生成します。EAM は、同種のエラーを集約してエラー数を削減し、ESM に渡されるエラーを減らすために必要なエラー ロギングと集約機能を提供します。

デバイスには以下の EAM モジュールが含まれています (x は 1 ～ 3、y は 1 ～ 2):

1. CPUx PR エラー アグリゲータ - CPU プログラムのフェッチ アクセス中に発生したエラーを集約します
2. CPUx DR1 エラー アグリゲータ - DR1 ポートでの CPU データ読み取りアクセス中に発生したエラーを集約します
3. CPUx DR2 エラー アグリゲータ - DR2 ポートでの CPU データ読み取りアクセス中に発生したエラーを集約します。
4. CPUx DW エラー アグリゲータ - CPU データ書き込みアクセス中に発生したエラーを集約します
5. CPUx INT エラー アグリゲータ - CPU および関連する PIPE モジュールからの割り込み関連エラーを集約します
6. RTDMAy DR エラー アグリゲータ - RTDMA データ読み取りアクセス中に発生したエラーを集約します
7. RTDMAy DW エラー アグリゲータ - RTDMA データ書き込みアクセス中に発生したエラーを集約します
8. SSU エラー アグリゲータ - SSU モジュールから送信されたエラーを集約します。

EAM モジュールの詳細については、*F29x テクニカル リファレンス マニュアル*の「エラー アグリゲータ」の章を参照してください。

以下のセクションでは、エラー集約、エラー ログ、およびエラー フラグ レジスタの解釈例を使用して示します。

4.1 エラー アグリゲーション

C29x CPU には、CPU DR1 (データ読み取りバス 1)、DR2 (データ読み取りバス 2)、DW (データ書き込みバス)、PR (プログラム フェッチ/ 読み取りバス) の 4 つのバスがあります。各バスから発生したエラーは、それぞれの EAM エラー フラグ レジスタで個別に検出し、記録され、エラー源の切り分けに使用されます。

エラー アグリゲーションに加えて、エラー イベントも**低優先度**エラーと**高優先度**エラーに分けられます。この例では、4 つのバスすべてのエラー イベントを最初に、エラーの**重大度**に基づく低優先度と高優先度の 2 つのカテゴリに分類してから、集約します。集約された出力 — 低優先度および高優先度のエラー イベント — は、その後 ESM に渡されます。エラー優先度は、重大度に応じてデバイス内で**事前定義**されています。EAM でキャプチャされたすべてのエラーのエラーの優先度の詳細については、*F29x テクニカル リファレンス マニュアル*の「エラー アグリゲータ」の章を参照してください。

各エラーにはエラー タイプの値と固定の事前定義済み優先度が割り当てられており、以下の表に CPU PR バスを例として示します。

表 4-1 に示す例は、CPU PR バス用です。シングル ビット (訂正可能エラー) および WARNPSP エラーは低優先度エラーに分類され、他のすべてのエラーは高優先度エラーに分類されます。CPU PR EAM 内のすべての高優先度エラー タイプには単一の集約出力があり、同様に CPU PR EAM 内のすべての低優先度エラーについても 1 つの集約出力があります。

表 4-1. EAM CPU PR エラー タイプ優先度

エラー タイプ値	CPUx PR エラー	RAM、ROM、FRI–PR エラー	優先順位
0x01	命令フェッチ セキュリティ違反。命令パケットが LINK、STACK、ZONE の境界を越えました。 リニア コードが LINK、STACK、ZONE の境界を越えました。 通常の分岐および呼び出しが STACK、ZONE の境界を越えました。	予約済み	**High**
0x02	セキュア エントリ エラー	予約済み	**High**
0x04	セキュア終了エラー	予約済み	**High**
0x08	MAX PSP エラー	予約済み	**High**
0x10	アクセス タイムアウト エラー	予約済み	**High**
0x20	アクセス ACK エラー	アクセス ACK エラー	**High**
0x40	修正不可能なエラー	修正不可能なエラー	**High**
0x80	修正可能なエラー	予約済み	Low
0x100	WARN PSP エラー	予約済み	Low
0x200	ソフトウェア ブレークポイント エラー	予約済み	**High**
0x400	不正な命令エラー	予約済み	**High**
0x800	命令のタイムアウトエラー	予約済み	**High**

すべての CPU バス である CPU PR、DR1、DR2、DW からの高優先度エラーは、CPU HPERR (高優先度エラー) として統合され、ESM に送られます。同様に、CPU PR、DR1、DR2、DW からのすべての低優先度エラーは CPU LPERR (低優先度エラー) として統合され、ESM に送られます。これは、図 4-1 に示すとおりです。

集約の利点は、ESM に渡されるエラー イベントの数と、それらのエラー イベント (上表に表示) に対する ESM 内でのエラー レスポンス構成を、すべての CPU バスにわたって高優先度と低優先度の 2 種類のエラー イベントにまで削減できる点です。特にデバイス内にこのようなエラー イベントが複数存在する場合、各 CPU バスから発生する各エラーを個別に設定するため、これは冗長です。そのため、すべての CPU バスでのエラーが集約され、ESM に供給されます。エラーの集約に加えて、分離も重要です。これにより、ユーザーは低優先度エラーと高優先度エラーに対して、それぞれ適切なアクションを生成するよう ESM を構成できるからです。

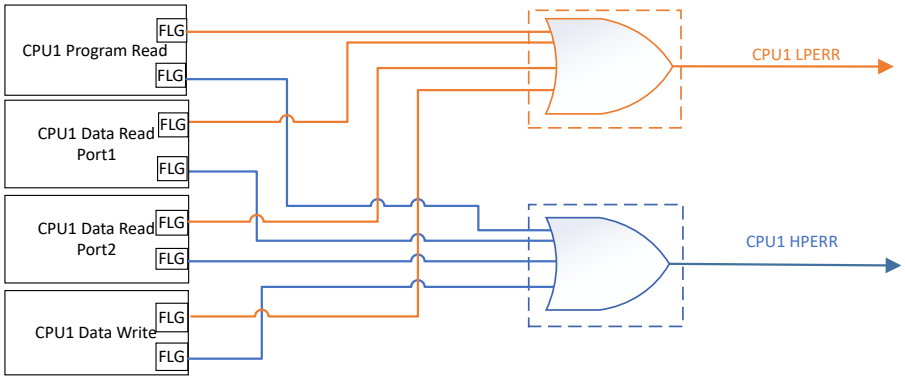

図 4-1. CPU1 EAM モジュール エラー集約

注

EAM エラー タイプ優先度は、ESM 出力優先度と混同しないでください。ESM 出力優先度の詳細については *F29x テクニカル リファレンス マニュアル*の ESM の章を参照してください。

4.2 エラー ログ

EAM モジュールは、ユーザーがデバイス内のすべての重大なエラーの原因をデバッグするために必要な包括的なエラー ロギング機能を提供します。

すべてのエラーは、次の情報とともに EAM レジスタに記録されます:

1. エラー タイプ – 特定のエラー タイプにマッピングされるマルチビット値 (例:アクセス アクノリッジ エラー、修正不能なエラーなど)。この値はエラー タイプごとにあらかじめ定義されており、例えば CPU PR エラーにおける Access ACK エラーの場合は 0x20 となります (上記「エラー集約」セクションの表を参照)。
2. エラー アドレス - エラーが発生したシステム アドレスであり、エラーの発生元をデバッグするために使用されます。独立した高優先度エラー アドレス レジスタと低優先度エラー アドレス レジスタがあります。
3. プログラム カウンタ (CPU EAM モジュールにのみ適用) — キャプチャされたプログラム カウンタ アドレスにより、エラーの原因を特定するのに役立ちます。PC アドレスは、エラーの原因となっている各 CPU が実行していたプログラム カウンタ アドレスに対応するコードを識別するのに特に役立ちます。

4.3 EAM モジュールを用いたエラー デバッグ

エラー デバッグを容易にするため、図 4-2 に示すように、エラー処理機能が Code Composer Studio (CCS) に統合されています。

CCS スクリプト メニューを使用すると、キャプチャされたエラー ステータスを確認できます。エラー デバッグのために以下に示すすべての手順は、画像 図 4-2 に示すように、スクリプト メニューから実行可能な GEL ファイルの Error_agg_Check_Status() ホット メニュー関数を使用して実行します。

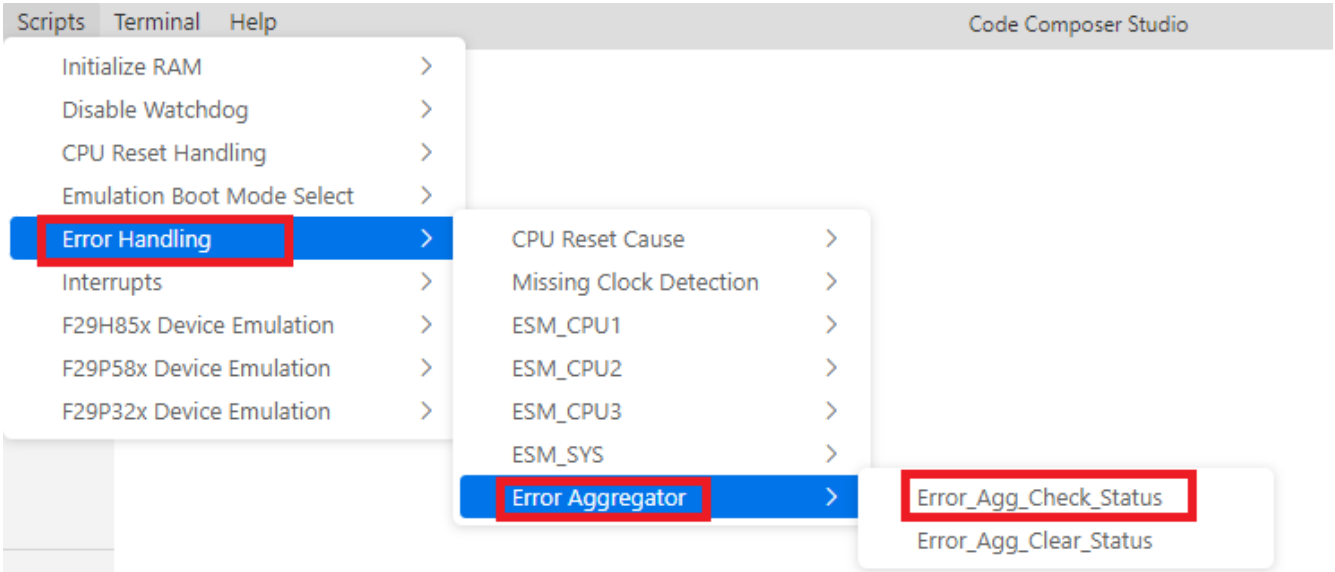

図 4-2. エラー アグリゲータ GEL ファイル関数

1. 各 EAM モジュール (CPU PR、DR1、DR2、DW、RTDMA、SSU、CPU INT、Ethercat) のエラー タイプ レジスタ値を確認し、その値が 0x0 以外であればエラーが発生しているかどうかを特定します。
2. 特定の EAM モジュールのエラー タイプ レジスタが 0x0 以外の値を示す場合、対応するエラー アグリゲータの低優先度エラーの場合は**低優先度エラー アドレス**を、高優先度エラーの場合は**高優先度エラー アドレス**を参照します。
3. 特定の EAM モジュール エラー タイプ レジスタに 0x0 以外の値がある場合、CPU EAM モジュールの場合のみ、エラー アグリゲータ**プログラム カウンタ** アドレスを見つけます。

4.3.1 EAM エラー デバッグ

1. Error_Agg_Check_Status() GEL ファイルのホット メニュー関数を実行してください。図 4-3 は、F29 SDK の ESM マルチコア例 (esm_ex1_cpu1_cpu3) に対応する CCS GEL 出力です。Error_Agg_Check_Status() 関数は、出力に対して以下を実行します:
   a. Error_Agg_Check_Status() は、エラー タイプ値を対応するエラーにマッピングします。この例では、GEL 出力に示されているように、エラーは CPU3 DW セキュリティ違反エラーです。これは、例の中で CPU3 のコードが許可されていない M0RAM の領域に書き込みを行っているためです。これは、エラー デバッグの例を示すエラー シナリオを作成するために意図的に行われています。
   b. 高優先度エラー アドレスは 0x20000000、プログラム カウンタは 0x10402C14 です。以下の GEL 出力に示されているとおり、対応する CPU3_DW 高優先度エラー アドレス レジスタおよびプログラム カウンタ レジスタから取得されます。

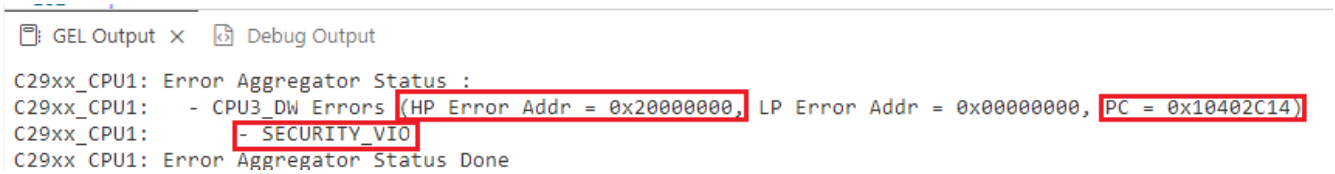

図 4-3. エラー アグリゲータ チェック ステータス GEL 出力ログ

2. Error_Agg_Check_Status() GEL 関数は、以下に示すように、CPU1 または CPU3 から ESM/EAM フラグをクリアする前に実行する必要があります。これを実現するために、NMI ISR – ESM/EAM フラグ クリア図に示すように、ESM/EAM フラグ クリア関数の実行前にブレークポイントを配置しました。これにより、EAM レジスタがクリアされる前に GEL 関数によって読み出され、デコードされるようにしています。

   Interrupt_clearEsmEaFlags() は、F29 SDK に含まれるリファレンス driverlib 関数であり、すべての EAM および ESM フラグをクリアします。本例の CPU1 および CPU3 の NMI ISR (NMI ISR – ESM/EAM フラグ クリア図を参照) で使用されているほか、driverlib のデフォルト NMI ハンドラにも含まれています。もし Interrupt_clearEsmEaFlags() 関数がすでに実行されている場合、ユーザーは nmiStatus 構造体 (ESM/EAM エラー フラグ レジスタ値を保存するためのメモリ領域) を参照することで、図 4-5 に示すように CCS の WATCH ウィンドウからエラー情報を確認できます。

注

EAM および ESM フラグのクリアは、NMI (マスク不可能割り込み) ISR における重要なステップです。これにより NMIWD (NMI ウォッチドッグ) のタイムアウトを回避し、システム リセット (XRSn) がトリガされるのを防ぎます。

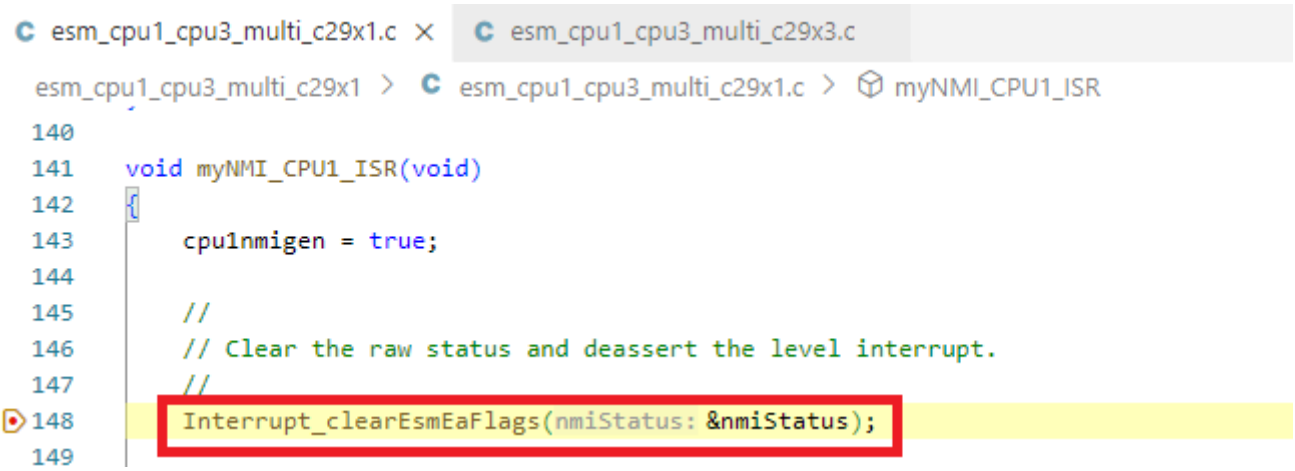

図 4-4. NMI ISR – ESM/EAM クリア フラグ

図 4-5. NMI ステータス キャプチャ ログ

3. 同じ例に対する CCS のレジスタ ビュー出力を以下に示します。
   a. レジスタ出力における高優先度エラー アドレス、プログラム カウンタ アドレス、およびエラー タイプ値は、「エラー アグリゲータ チェック ステータス GEL 出力ログ - 図 4-3」に示された GEL 関数ファイルの出力内容と一致しています。

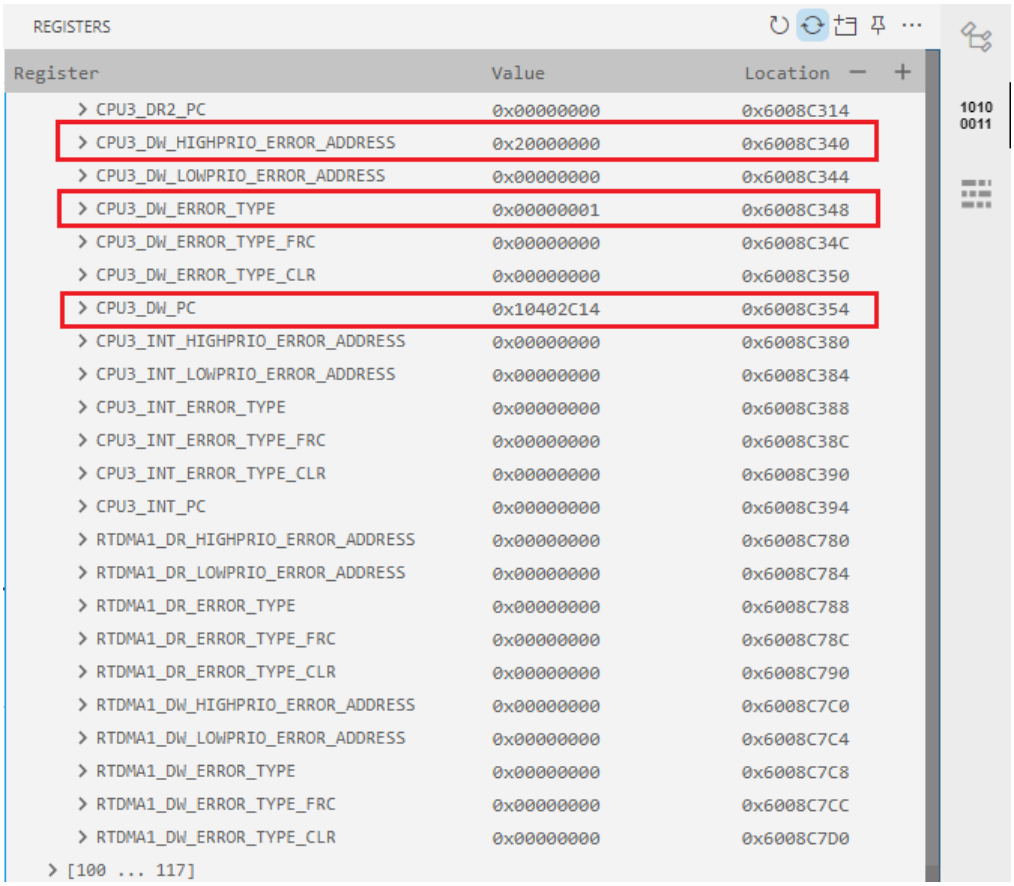

図 4-6. エラー アグリゲータ レジスタの CCS レジスタ表示

4.3.2 エラー アドレスとプログラム カウンタ値の解釈

セクション 3.2 で説明しているように、エラー アドレスとプログラム カウンタ アドレスを、エラーのソースのデバッグに使用できます。このセクションでは、F29 SDK から esm マルチコアの例 (esm_ex1_cpu1_cpu3) を取得したエラー アドレスとプログラム カウンタの解釈を示します。

プログラム カウンタ (PC) のアドレスを CCS 分解図にコピーして、エラーの発生したソース コードを見つけることができます。この例では、図 4-7 に示すように、EAM キャプチャされた PC アドレス (0x10402C14) に対する対応する PC アドレス CCS 分解表示を見ると、EAM の高優先度アドレス レジスタにも記録されている、位置 0x20000000 (M0 RAM) へのデータ書き込み動作を示しています。したがって、PC アドレスおよびエラー アドレス情報では、ユーザーは、エラーの原因となったメモリ位置への特定の CPU3 ソース コード書き込み動作をピンポイントで特定できます。

CPU3 DW (データ書き込み) バスで発生したエラーは、CPU3 アプリケーション コードでは CPU3 コードから許可されていない M0RAM_data から M0RAM への書き込み動作があるため、コードの観点で期待される動作と一致します。CPU3 には M0RAM に対する読み取りデータ許可のみがあるため、この場合の書き込み動作では CPU3 DW バスでセキュリティ違反エラーが発生しました。

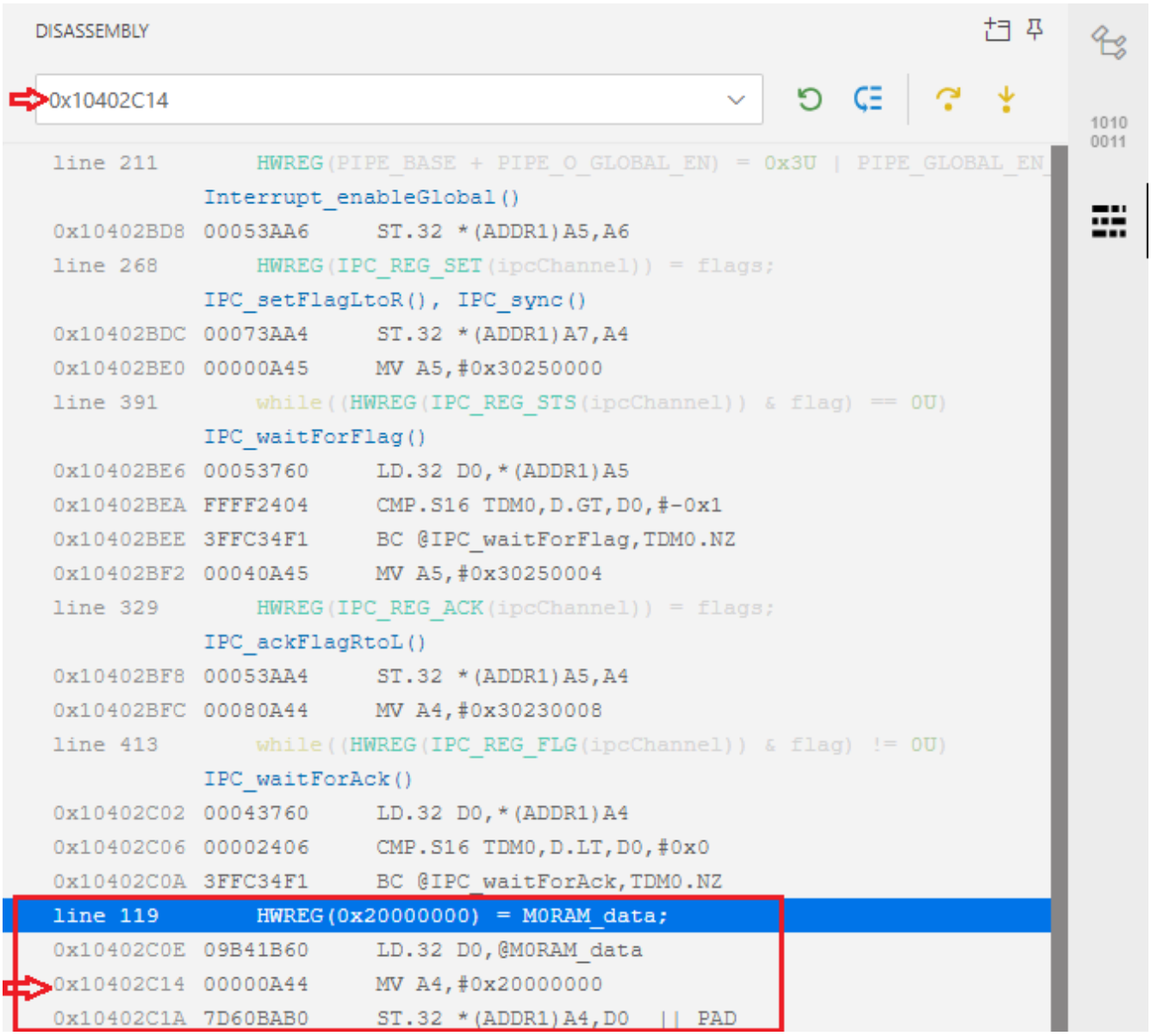

図 4-7. プログラム カウンタの分解図

表 4-2. CPU3 からの M0RAM へのアクセス

メモリ	インターリーブ型	CPU1	CPU2	CPU3	HSM	RTDMA1	RTDMA2
M0 RAM	あり	0WS データ (読み取りおよび書き込み)	0WS データ (読み取り専用)	3WS データ (読み取り専用)	-	-	-

5 エラー通知モジュールの概要

エラー通知モジュール (ESM) は、デバイス全体で発生するエラー イベントに対する応答を一元的に集約する仕組みを提供しており、これは複数の安全が必須なアプリケーションにとって極めて重要です。

ESM サブシステムには以下のモジュールが含まれています:

1. ESM CPU1 - CPU1 への出力専用 ESM モジュール
2. ESM CPU2 - CPU2 への出力専用 ESM モジュール
3. ESM CPU3 - CPU3 への出力専用 ESM モジュール

4. システム ESM — システム レベルの出力専用の ESM モジュール (主に ERRORSTS ピン出力、デバイス リセット、XBAR イベント出力を用いた他モジュールとの統合)

図 5-1 では、ESM サブシステムがデバイス レベルでどのように統合されるかを詳細に説明しています。詳細については、F29x TRM の ESM 章を参照してください。

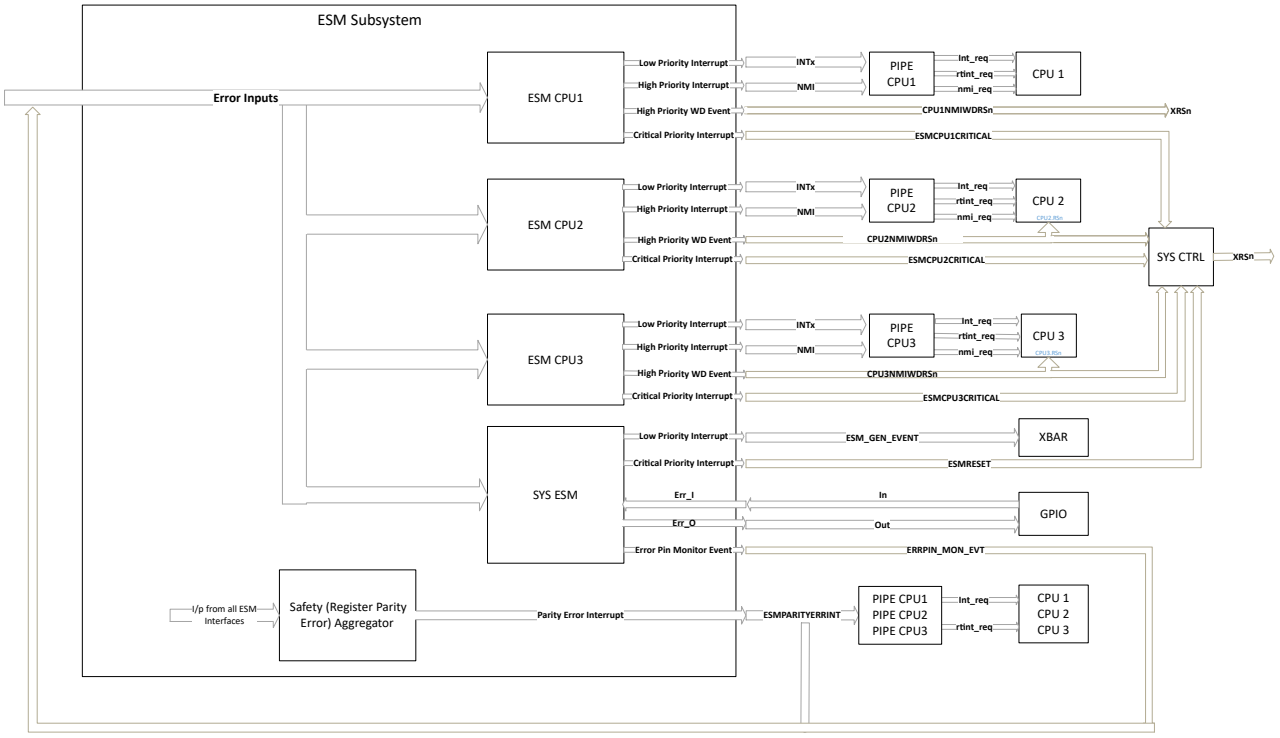

図 5-1. ESM サブシステム統合のブロック図

ESM は、エラーを重大度で分類し、プログラム可能なエラーレスポンスを提供する機能を備えています。エラー通知モジュールは、発生したエラーの重大度に応じて、エラーピン応答、選択可能な割り込み優先度応答、または CPU へのマスク不可能割り込み (NMI) を示す手段を提供します。ユーザーは、各エラー イベントに対してどのエラー レスポンスを取るかを決定する責任を負っており、それがシステムの安全コンセプトと整合するようにする必要があります。

1. 特定の CPU への割り込み:
   a. 割り込み (PIPE から CPU への INT または RTINT)(ESM の低優先度割り込み出力) — 一般的には、デバイスで発生した訂正可能エラーや重大度の低いエラーに対して選択されます。また、CPU 外部での診断用途として実装することもできます。割り込みにより、CPU 外部のイベントが割り込みハンドラへのプログラム シーケンスのコンテキスト転送を生成でき、そこでソフトウェアが故障を処理する機会を得られます。
   b. マスク不可能割り込み (NMI)(ESM の高優先度割り込み出力) — 一般的に、デバイスで発生した修正不可能なエラーや致命的なエラーに対して選択されます。この場合、エラー レスポンスとしてコンテキストを NMI ISR に転送し、ソフトウェアが故障を処理して動作を安全に中止する機会が提供されます。
2. エラー通知ピン:
   a. エラー ピン (ERRORSTS) の動作は、PMIC (パワー マネジメント集積回路) などの外部モニタ向けであり、外部エラーのレスポンスを生成する必要がある場合に使用されます。
3. リセット
   a. 対応 CPU リセット (CPURSn):ESM は、MCU 内でエラーを検出した際に、システム安全な状態に移行させるため、個々の CPU にリセットを生成することが可能です。

b. デバイス リセット (XRSn): エラーが検出されると、デバイス リセット (XRSn) をトリガして、MCU を安全な状態にします。

以下のセクションでは、ESM サブシステムの上記の出力についての ESM CPU およびシステム ESM モジュールの**構成**の概要を簡単に示します。詳細については、*F29x テクニカル リファレンス マニュアル*の ESM の章とデバイスの統合を参照してください。

ESM のエラー イベントの概要と重要なポイント:

1. エラー イベントは、すべての ESM モジュール (ESM CPU1/2/3 およびシステム ESM) に共通です
2. 各 ESM モジュールには個別の**構成**および**ステータス** レジスタがあるため、すべての ESM モジュールは互いに独立して動作できるため、さまざまな使用事例に柔軟に対応できます。例えば、あるエラー イベントが発生した場合、ESM CPU1 から CPU1 への割り込み出力を生成するように構成しつつ、ESM CPU3 モジュールから CPU3 への割り込み出力は構成**しない**、という設定が可能です。
3. エラー イベントは 32 のグループに分けられます。F29x デバイスには合計 256 のエラー イベントがあり、したがって全部で 8 グループ存在します。

注

すべての Group0 エラー イベントは、デフォルトで NMI をトリガするようにマッピングされています。Group0 エラー イベントは、EAM (エラー アグリゲータ モジュール) からの高優先度集約 CPU エラー出力です。

5.1 ESM エラー イベント出力の構成およびステータス情報

下図は、各 ESM ブロックをどのように構成することで、それぞれの ESM モジュールからの出力に影響を与えられるかを示しています。これらの出力をデバイス ペリフェラルに接続する方法については、*F29x テクニカル リファレンス マニュアル* ESM の章の「ESM サブシステム デバイスの統合図」を参照してください。

以下に示すステータスレジスタは、どのエラー イベントがアクティブで有効化され、ESM CPU やシステム ESM モジュールからの出力に影響を与えているかを特定するのに役立ちます:

1. RAW ステータス/ セット レジスタ (RAW_j) — エラー イベントがアクティブであるかを示すレジスタです。j はエラー イベント インデックスを表します (j = 0 ～ 255)。
2. 割り込みイネーブル ステータス/クリア レジスタ (STS_j) — このレジスタは、エラー イベントがアクティブであり、低優先度または高優先度の割り込みに影響を与えるよう有効化されているかどうかを示します。j はエラー イベント インデックスを表します (j = 0 ～ 255)。

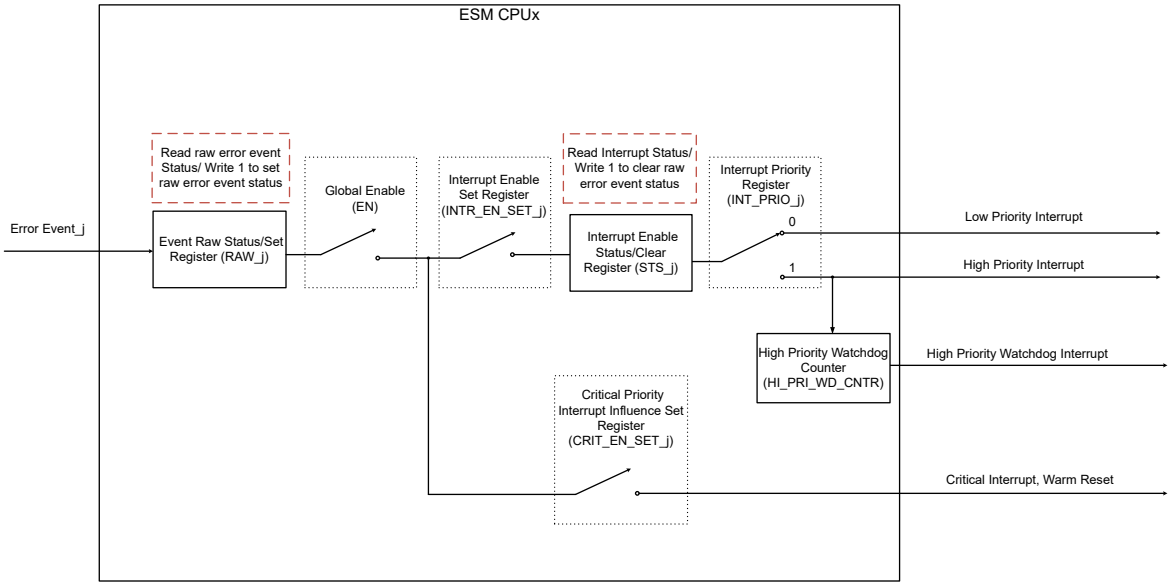

図 5-2. ESM CPU の詳細構成とステータス情報ビュー

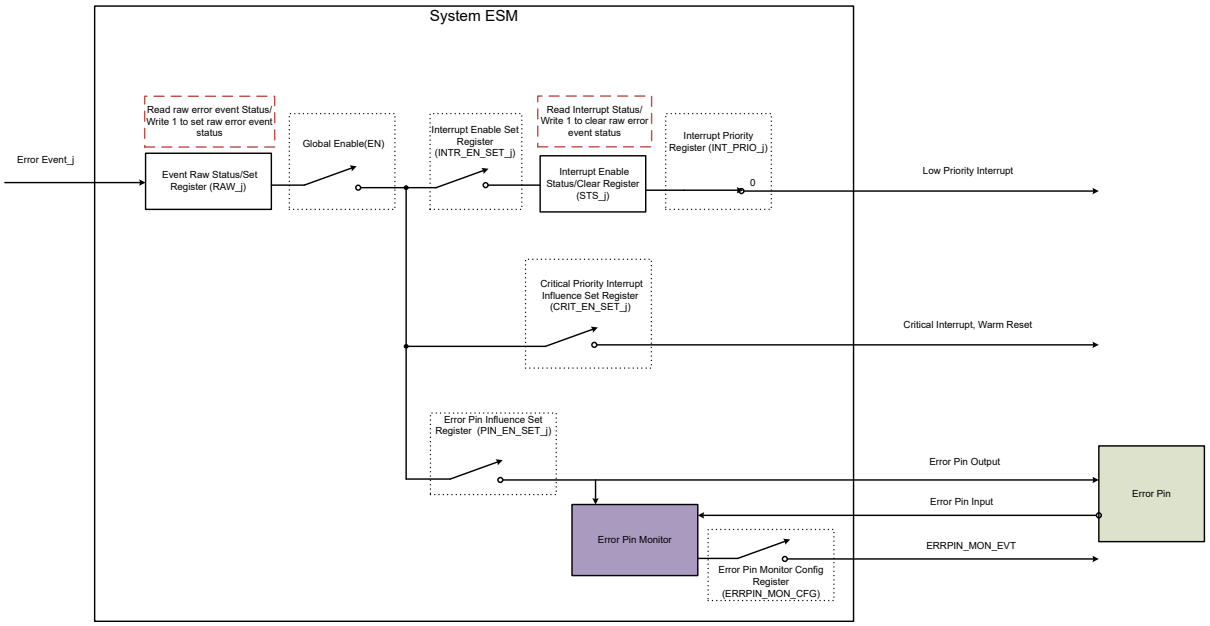

図 5-3. システム ESM 詳細構成およびステータス情報ビュー

5.1.1 Sysconfig ESM の構成

Sysconfig は、セクション 5.1 で説明されているように、各 ESM モジュールからの目的の出力になるように個別のエラーイベントを設定できます。

例えば 図 5-4 では、ESM CPU1 をエラー イベント「エラー アグリゲータ CPU1 HPERR」に対して構成し、高優先度 NMI 出力を生成する方法を示しています。各 ESM CPU の Sysconfig モジュール内にあるグローバル パラメータ設定

を使用することで、高優先度ウォッチドッグの有効化、ウォッチドッグ カウンタのプリロード値の設定に加えて、低優先度割り込みや NMI 出力の割り込みハンドラ構成を定義することができます。

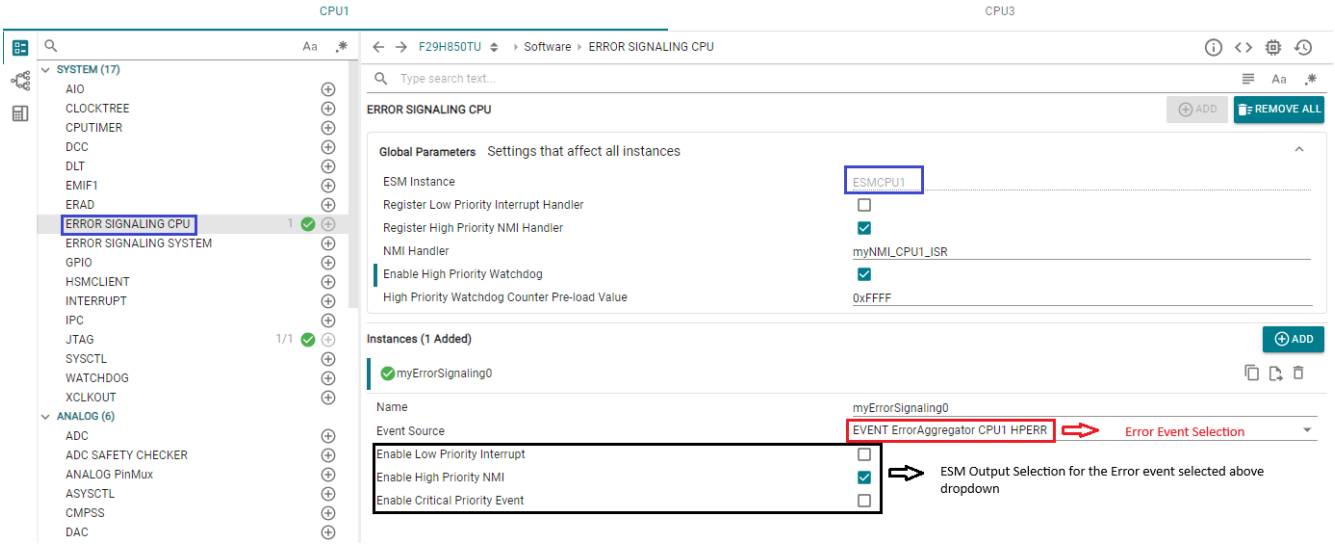

図 5-4. ESM CPU Sysconfig モジュール

例えば 図 5-5 では、システム ESM をエラー イベント「エラー アグリゲータ CPU1 HPERR」に対して構成し、エラー ピンに影響を与えるよう有効化する方法を示しています。エラー ステータス ピン (ERRORSTS) の構成 (極性や出力ピンモード構成など) は、システム ESM Sysconfig モジュールのグローバル パラメータ セクションを使用しても実行できます。

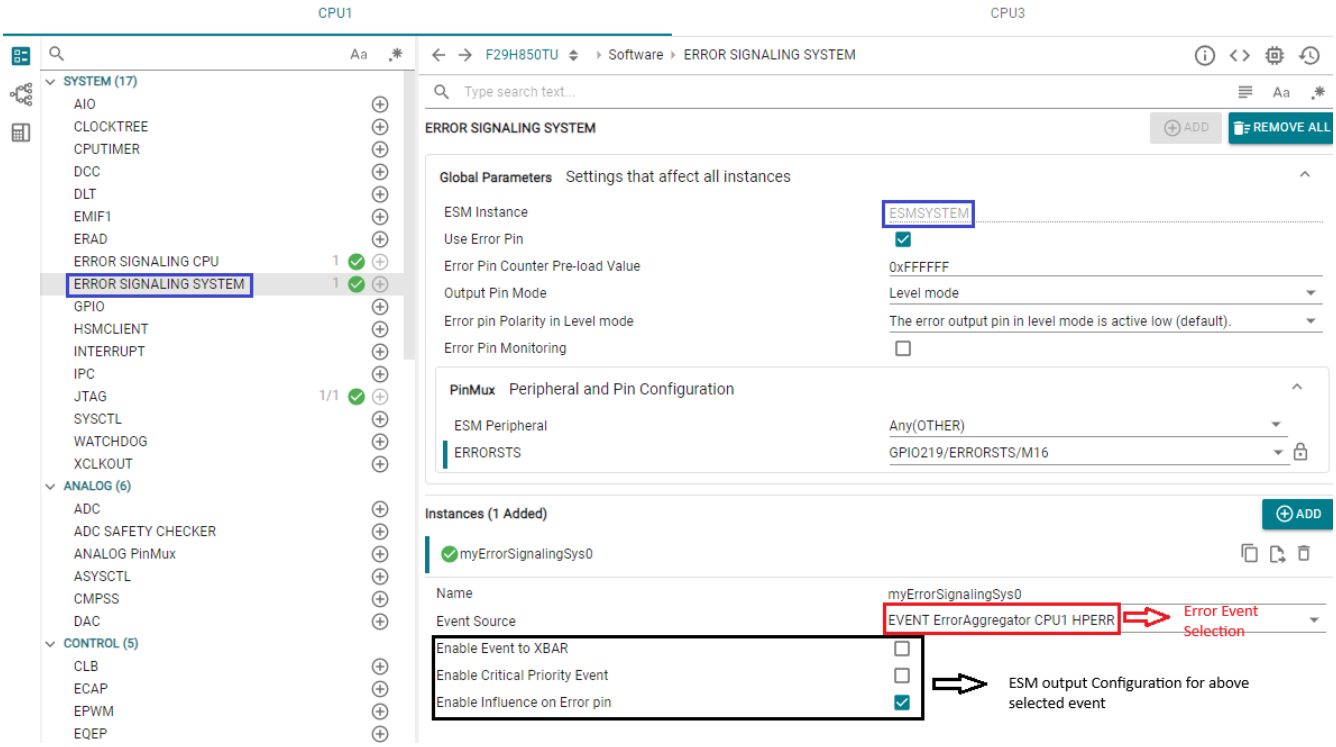

図 5-5. システム ESM Sysconfig モジュール

5.2 ESM エラー イベント デバッグ

エラー イベントをデバッグするには、以下の手順に従います:

1. 図 5-6 に示すように、次のスクリプト ESM_CPU_Check_Status () GEL ファイル ホット メニュー関数を CCS から実行して、エラー イベントのステータスを確認します。この機能出力は、2 つのカテゴリに分類されたエラー イベントを示します:
   a. **アクティブ/ 保留中**エラー イベント — アクティブ/保留中のエラーイベントを示します
      i. エラー イベントがアクティブであるとは、エラー イベントの Raw ステータスがセットされていることを意味します。このとき、機能は F29x TRM の ESM エラー イベント表に記載された各イベントについて、RAW ステータス レジスタ (RAW_j) をチェックします。
   b. **アクティブ**、**保留中**、および**有効**エラー イベント –アクティブ / 保留中で、イネーブルされているエラー イベントを示します。
      i. エラーイベントがアクティブ、保留中、かつイネーブルであるとは、エラーイベントの RAW ステータスがセットされており、さらにユーザーによって割り込みイネーブル セット レジスタも設定されていて、対応する ESM モジュールから割り込み出力をトリガできる状態を意味します。

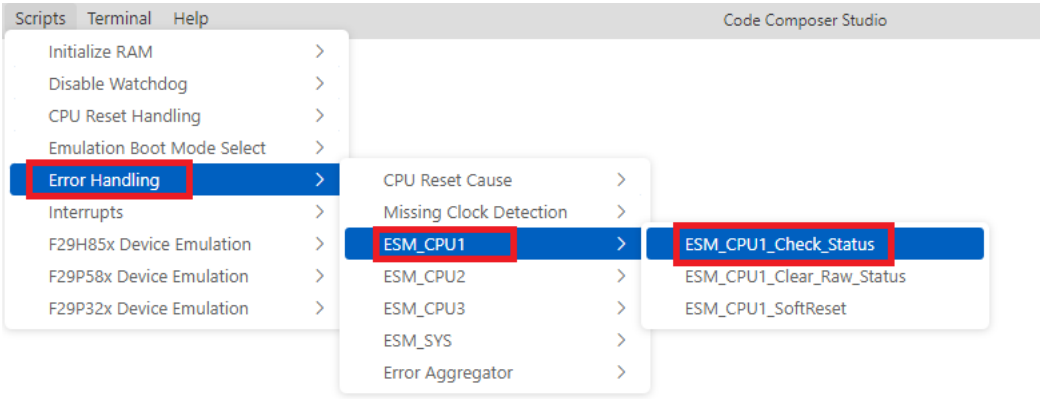

図 5-6. ESM エラー ステータス GEL 関数

2. GEL 出力の例と、それに対応する ESM レジスタとの関連を、下図に示します。これは、F29 SDK から引用した同じ ESM マルチコア例の続きです。
   a. 図 5-7 は、CPU1_ERAD_NMI エラー イベントがアクティブであり、イネーブルであることを示しています。ESM CPU1 - CPU1_ERAD_NMI エラー イベントに対しても割り込み優先度レジスタ (INT_PRIO) が設定されており、これにより CPU1 (ESM CPU1 内) および CPU3 (ESM CPU3 内) の両方で NMI がトリガーされます。
   b. これに加えて、ErrorAggregator_CPU3_HPERR (上記のセクションで説明しているように、CPU3 DW バスのセキュリティ違反エラーによるエラーアグリゲータ CPU3 高優先度エラー)、EPWMXBAR1、CPU1 高優先度割り込み、および CPU3 高優先度割り込み出力もアクティブで、ESM RAW ステータスレジスタ (RAW_j) 値を使用してデコードされます。CPU1 および CPU3 の高優先度割り込み出力は、それぞれ CPU1 および CPU3 の NMI 出力フラグに対応します。また、EPWM XBAR および ERAD NMI イベントは NMI エラッタ回避策の実装で使用されるため、想定どおりアクティブ状態となります。詳細は F29x デバイスのエラッタを参照してください。

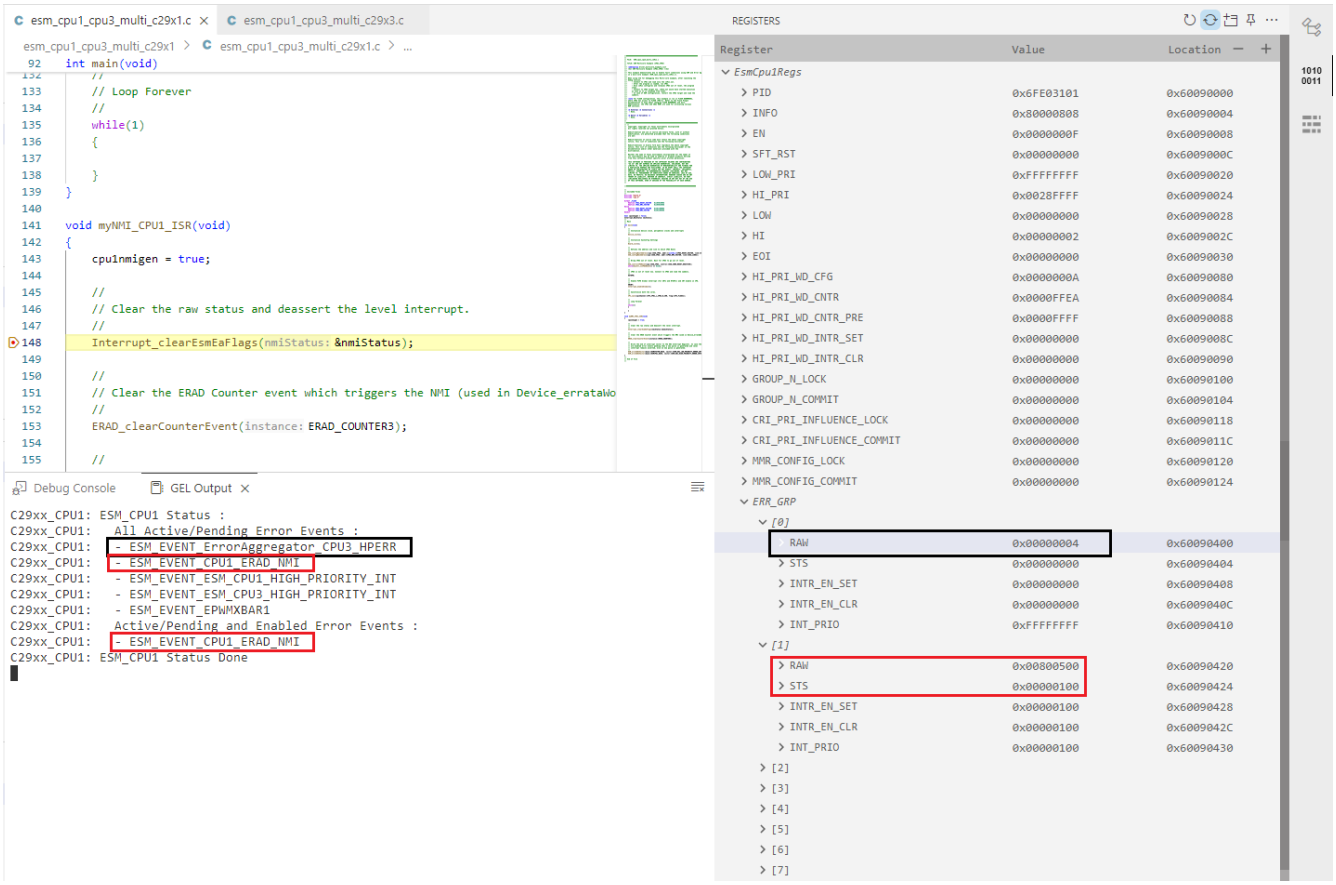

図 5-7. ESM エラー イベント ステータス GEL 出力

3. EAM レジスタ フラグと同様に、RAW ステータス レジスタを NMI ISR 内でクリアする前に GEL 出力を確認してください。または、前述のように nmiStatus 構造体を確認し、後でデバッグ用に保存されている内容を参照してください。特定のイベントがアクティブ時に NMI をトリガするよう構成されている場合、NMIWD タイムアウトを回避するためには、ESM RAW ステータス (RAW_j) レジスタ フラグをクリアすることが必要です。

5.3 ESM に関するその他のデバッグのヒント

1. RESC (リセット要因) レジスタを確認し、ESM 高優先度ウォッチドッグ割り込み (NMIWD) が原因で NMIWDRSn が発生したかどうかを検証します。システム ESM がエラー イベントのクリティカル優先度割り込み出力に設定されている場合、RESC レジスタの ESMRESET ビットがチェックされます。
2. ESM HI_PRI レジスタを確認します。このレジスタは、最も優先度の高い未処理の高優先度割り込みを示します。最小のイベント番号の優先度が最も高く、値 0xFFFF はアクティブ / 保留中の高優先度の割り込みが存在しないことを示します。
3. ESM LOW_PRI レジスタを確認します。このレジスタは、最も優先度の高い未処理の低優先度割り込みを示します。最も小さいイベント番号が最高優先度を持ちます。値が 0xFFFF の場合、アクティブまたはペンディング状態の低優先度割り込みが存在しないことを示します。

6 BootROM における EAM/ESM エラー ステータス

アプリケーションが NMIWD (高優先度ウォッチドッグ) のタイムアウト前にエラーをクリアできない場合、ESM CPU1 インスタンスからリセットがトリガされます。この場合、デバイスリセット (XRSn) 後に実行される bootROM がエラーをクリアし、連続する NMIWD リセット ループを回避します。また、エラー情報とステータスを M0 RAM に保存し (下表参照)、後続のデバッグに利用できるようにします。

BootROM は次のステータスをクリアします:

1. ESM グループ 0 の RAW ステータスは、ESM CPU1 およびシステム ESM のインスタンスに対する ESM サブシステムのものです
2. すべての CPUx エラー アグリゲータ タイプのレジスタ

さらに、エラー原因のデバッグ用として、以下の情報が M0 RAM に保存されます:

1. Group0 専用の ESM RAW ステータス
2. エラー アグリゲータ CPU1 - PR、DR1/2、DW、INT インスタンスのエラー情報 (高優先度エラー アドレス、低優先度エラー アドレス、エラー タイプ、プログラム カウンタ レジスタを含む)

表 6-1. BootROM エラー ステータス情報

説明	アドレス
ESM RAW ステータス	0x2000_0868
CPU1 PR エラー アグリゲータ高優先度エラー アドレス	0x2000_086C
CPU1 PR エラー アグリゲータ低優先度エラー アドレス	0x2000_0870
CPU1 PR エラー アグリゲータ エラー タイプ	0x2000_0874
CPU1 PR エラー アグリゲータ PC 値	0x2000_0878
CPU1 DR1 エラー アグリゲータ高優先度エラー アドレス	0x2000_087C
CPU1 DR1 エラー アグリゲータ Low 優先度エラー アドレス	0x2000_0880
CPU1 DR1 エラー アグリゲータ エラー タイプ	0x2000_0884
CPU1 DR1 エラー アグリゲータ PC 値	0x2000_0888
CPU1 DR2 エラー アグリゲータ高優先度エラー アドレス	0x2000_088C
CPU1 DR2 エラー アグリゲータ Low 優先度エラー アドレス	0x2000_0890
CPU1 DR2 エラー アグリゲータ エラー タイプ	0x2000_0894
CPU1 DR2 エラー アグリゲータ PC 値	0x2000_0898
CPU1 DW エラー アグリゲータ高優先度エラー アドレス	0x2000_089C
CPU1 DW エラー アグリゲータ低優先度エラー アドレス	0x2000_08A0
CPU1 DW エラー アグリゲータ エラー タイプ	0x2000_08A4
CPU1 DW エラー アグリゲータ PC 値	0x2000_08A8
CPU1 INT エラー アグリゲータ高優先度エラー アドレス	0x2000_08AC
CPU1 INT エラー アグリゲータ低優先度エラー アドレス	0x2000_08B0
CPU1 INT エラー アグリゲータ エラー タイプ	0x2000_08B4
CPU1 INT エラー アグリゲータ PC 値	0x2000_08B8

7 FAQ:

1. ユーザーが NMI ISR を設定して構成しますか？

   はい。推奨される方法として、ユーザーはデバイス初期化の際に NMI ISR をセットアップしておく必要があります。これにより、高優先度のエラー (例えば Group0 のエラー イベント) が発生して NMI が各 CPU にトリガされた場合でも、F29x TRM および ESM マルチコア F29 SDK の例に従って、EAM および ESM 内のエラー フラグを適切にクリアできるようになります。ESM の raw ステータス フラグをクリアしないと、NMIWD のタイムアウトが発生し、XRSn (デバイスリセット) がトリガされます。ユーザーは RESC (リセット要因) レジスタ内の NMIWD ビットを確認することで、それを確認できます。
2. もしアプリケーション側で NMI ISR がセットアップされておらず、Group 0 の ESM エラー イベントのデフォルト設定に基づいてエラー イベントが NMI を発生させた場合はどうなりますか？

   回答 — ユーザー / アプリケーション側で NMI ISR が設定されていない場合、CPU は BootROM 内のデフォルト NMI ハンドラに移行し、そこで CPU がエラーステータスとフラグをクリアして保存し、デバッグのために M0 RAM アドレスに格納します。詳細については、BootROM TRM の該当する章を参照してください。
3. NMI ISR が設定されているものの、エラー フラグがクリアされなかった場合はどうなりますか？

   回答 — ESM の NMIWD タイムアウトが発生し、対応する CPU から高優先度のウォッチドッグ割り込み出力が生成されます。ESM CPU1 の高優先度ウォッチドッグ割り込み出力は XRSn を発生させるように接続されており、ESM CPU2/ CPU3 はそれぞれの CPURSn を発生させます。これは ESM サブシステム統合ビュー図に示されています。
4. 高優先度の CPU EAM エラー (Group 0 エラー イベントとして ESM に渡される) について、ESM が各 CPU に NMI を生成するように構成されていない場合はどうなりますか？

   回答 – ESM Group0 エラー イベントがアクティブで、EAM モジュールを通じて渡されているものの、NMI を生成するように構成されていない場合、CPU は故障状態に入ります。CPU EAM モジュールの高優先度エラーがアクティブになった場合は、必ず該当 CPU に NMI をトリガーするよう構成しておく必要があります。
5. ESM/EAM のフラグは、XRSn (デバイス リセット) または CPURSn (CPU リセット) の後にクリアされますか？

   回答 – いいえ。本書で「EAM エラー フラグ」と呼ばれているエラー タイプ レジスタや、「ESM エラー フラグ」と呼ばれている ESM RAW ステータス レジスタ (RAW_j) は、PORESETn によってのみリセットされます。これらのフラグは、XRSn や CPURSn の後でも値を保持します。これは、アプリケーション ソフトウェアで処理されなかったエラー イベントに応答して ESM によってリセットがトリガされた場合、デバッグのためにこのエラー フラグ情報が必要となるためです。
6. NMI を引き起こしたすべてのイベントについて ESM の Raw ステータス レジスタをクリアした後でも、なぜ CPU は NMI ISR に入り続けますか？

   回答 – ESM RAW ステータス レジスタ (RAW_j) のフラグをクリアした後は、対応するエラーの ESM 割り込み出力に対して、適切なキーを書き込んで EOI レジスタも更新する必要があります。EOI への書き込みステップは、基本的にユーザーが ESM 割り込み出力を確認したことを示すものであり、その結果、ESM 出力がデアサートされます。EOI レジスタに書き込みを行わない場合、ESM エラー イベント フラグをクリアしても、ESM 割り込み出力はアサートされたままになります。
7. 特定のエラー イベントの発生に基づいて、CPU1 にのみ NMI を生成し、CPU3 には生成しない、またはその逆を行うことは可能ですか？

   回答 – はい。ESM は非常に柔軟に設定可能であり、各 CPU 専用の ESM タイルが割り当てられています。そのため、初期化時に設定を行うことで、各 CPU 専用の ESM タイルから任意の CPU に対して NMI または他のエラーレスポンスを生成することや、エラーレスポンスを無効化すること、または別の ESM CPU タイルから特定のエラーイベントに対して異なるレスポンスを設定することが可能です。

8 まとめ

ESM と EAM は、デバイスでエラーが発生した際に、それを処理する体系的な方法をユーザーに提供します。アプリケーション ノートでは、エラー イベントが検出されたソースから、ESM からデバイスにどのようにエラー応答が供給されるかを説明しています。最後に、このドキュメントでは、エラーの原因を特定するために使用できるデバッグ ツールの詳細を示します。

9 参考資料

1. テキサス インスツルメンツ、*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
2. テキサス インスツルメンツ、*F29H85x-SDK：CPU1 と CPU3 の間の ESM の例*
3. テキサス インスツルメンツ、*CCSTUDIO：Code Composer Studio™ 統合開発環境 (IDE)*
4. テキサス インスツルメンツ、C2000™ SysConfig

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