Application Note

F29x 错误处理和调试指南

Prarthan Bhatt

摘要

本应用手册重点概述了错误处理架构并就如何高效调试错误事件提供指导。F29x 器件架构提供系统性错误处理管理，专注于各种终端应用的功能安全。错误聚合器模块 (EAM) 和错误信令模块 (ESM) 为整个器件中的所有错误事件提供错误聚合、记录和可配置的响应。本应用手册就如何使用工具以及 EAM 和 ESM 提供的错误记录调试错误源提供了指导。

内容

商标

1 简介

对于功能安全关键型开发，必须对系统性故障和随机故障进行管理。F29x 器件架构具有内置硬件安全机制，可对整个器件的所有错误事件提供系统性管理。首先，从整体上理解错误处理架构，然后深入研究如何解读错误日志以及针对每个错误事件配置响应。

本应用手册首先概述错误处理架构，然后通过一个示例详细说明 EAM 和 ESM 功能和工具，最后介绍常见问题和错误调试技巧。

表 1-1 列出了本应用手册中使用的术语和缩写。

表 1-1. 使用的术语和缩写及说明

术语或缩写	说明
系统性错误	系统性故障是由设计、开发或制造流程中存在的某种不足引起的，并且通常源于开发流程中的缺陷。如需了解更多信息，请参阅 ISO 26262 等安全标准
随机错误	随机错误是指在元件使用寿命期间发生的、不可预测的硬件故障。与确定性的、由设计缺陷导致的系统性错误不同，随机错误是统计错误，可通过概率分析进行管理。
系统地址	每个 MCU 都有唯一的存储器映射，用于定义每个元件的地址范围。系统地址是位于此映射中并与特定资源相对应的地址
EAM	错误聚合器模块
ESM	错误信令模块
NMI	不可屏蔽中断
CCS	Code Composer Studio

2 错误处理架构概述

图 2-1 概括展示了如何实现错误处理。错误在错误源处检测到，错误源可以是外设、模块、存储器、互连或处理单元，错误传送到 EAM 进行错误聚合，然后传输到 ESM，在器件内进行用户可配置的错误响应。需要错误聚合和记录的关键器件错误事件通过 EAM，而所有其他器件错误事件则直接从错误源传输到 ESM，并在 *F29x 技术参考手册* 的 ESM 一章的“错误事件”表中列出。

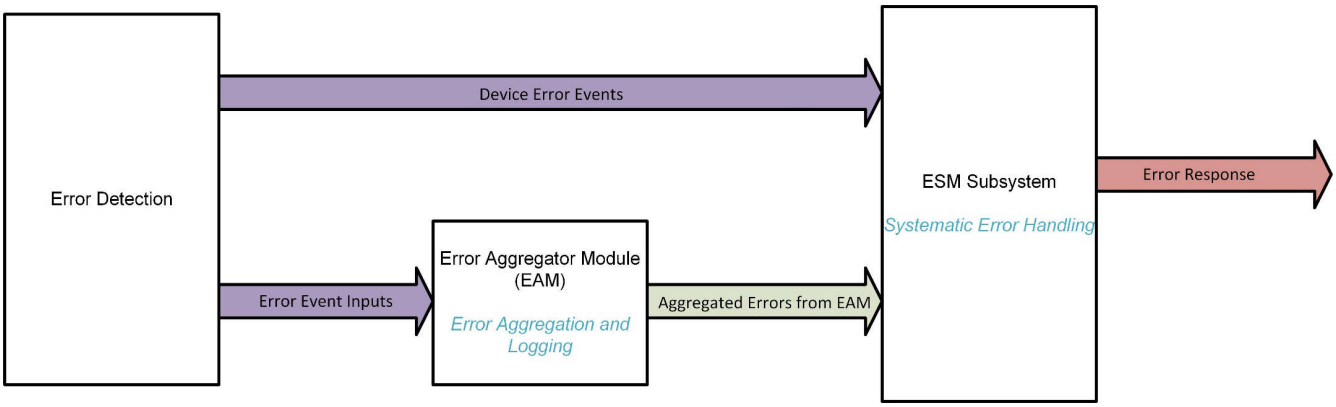

图 2-1. 器件错误处理架构

3 示例概述

以下各节将讲解 EAM 和 ESM 功能，并通过一个示例逐一介绍 ESM 和 EAM 元件。本应用手册使用了 F29x SDK 中的 *F29 SDK ESM 多核示例*（CPU1 和 CPU3）。CPU3 应用程序代码将写入 M0RAM 位置，这会导致 CPU3 DW 总线安全违例错误。示例展示了如何使用 EAM 和 ESM 来处理该错误。

本应用手册采用了该示例，展示了如何使用工具调试错误。

4 错误聚合器概述

错误聚合器模块 (EAM) 是 **ESM** 和可产生错误的**关键模块**（例如 C29x CPU、PIPE、RTDMA、存储器控制器、外设桥和读取接口）之间的接口。EAM 提供类似错误类型所必需的错误记录和聚合，以减少传递给 ESM 的错误数量。

该器件包含以下 EAM 模块（其中 x 从 1 到 3，y 从 1 到 2）：

1. CPUx PR 错误聚合器 - 聚合 CPU 程序获取访问期间发生的错误
2. CPUx DR1 错误聚合器 - 聚合在 DR1 端口上进行 CPU 数据读取访问期间发生的错误
3. CPUx DR2 错误聚合器 - 聚合在 DR2 端口上进行 CPU 数据读取访问期间发生的错误
4. CPUx DW 错误聚合器 - 聚合 CPU 数据写入访问期间发生的错误
5. CPUx INT 错误聚合器 - 聚合来自 CPU 和相关 PIPE 模块的中断相关错误
6. RTDMAy DR 错误聚合器 - 聚合 RTDMA 数据读取访问期间发生的错误
7. RTDMAy DW 错误聚合器 - 聚合 RTDMA 数据写入访问期间发生的错误
8. SSU 错误聚合器 - 聚合 SSU 模块发出的错误

有关 EAM 模块的详细视图，请参阅 *F29x 技术参考手册* 中的“错误聚合器”一章。

下面几节通过一个示例展示错误标志寄存器的错误聚合、错误记录和解读。

4.1 错误聚合

C29x CPU 有 4 条总线：CPU DR1（数据读取总线 1）、DR2（数据读取总线 2）、DW（数据写入总线）和 PR（程序获取/读取总线）。为了找出错误源，会分别检测并捕获/记录每个总线产生的错误并将其记录到相应的 EAM 错误标志寄存器中。

除了错误聚合之外，错误事件还被分成**低优先级**错误和**高优先级**错误。在此示例中，根据错误的**严重性**，所有四条总线中的错误事件首先分为两类：优先级和高优先级错误，然后进行聚合。聚合输出，即低优先级和高优先级错误事件随后传输到 ESM。根据严重性在器件中**预定义了**错误优先级，请参阅 *F29x 技术参考手册* 中的错误聚合器一章，深入了解有关 EAM 中捕获的所有错误的错误优先级。

每个错误都分配了错误类型值和固定的预定义优先级，如下面的 CPU PR 总线示例表所示。

表 4-1 中展示的示例针对 CPU PR 总线。单比特（可纠正错误）和 WARNPSP 错误归类为低优先级错误，所有其他错误归类为高优先级错误。CPU PR EAM 中的所有高优先级错误类型都有单个聚合输出，类似地，对于 CPU PR EAM 中的所有低优先级错误，也有一个聚合输出。

表 4-1. EAM CPU PR 错误类型优先级

错误类型值	CPUx PR 错误	RAM、ROM、FRI – PR 错误	优先级
0x01	指令获取安全违例。指令包跨过 LINK、STACK、ZONE 边界。 线性代码跨过 LINK、STACK、ZONE 边界。 常规分支和调用跨过 STACK、ZONE 边界。	保留	**高**
0x02	安全进入错误	保留	**高**
0x04	安全退出错误	保留	**高**
0x08	MAX PSP 错误	保留	**高**
0x10	访问超时错误	保留	**高**
0x20	访问 ACK 错误	访问 ACK 错误	**高**
0x40	不可纠正的错误	不可纠正的错误	**高**
0x80	可纠正错误	保留	低
0x100	WARN PSP 错误	保留	低
0x200	软件断点错误	保留	**高**
0x400	非法指令错误	保留	**高**
0x800	指令超时错误	保留	**高**

来自所有 CPU 总线的所有高优先级错误（CPU PR、DR1、DR2 和 DW）也会合并为 CPU HPERR（高优先级错误）并发送到 ESM。同样，来自 CPU PR、DR1、DR2 和 DW 的所有低优先级错误合并为 CPU LPERR（低优先级错误）并发送到 ESM。图 4-1 展示了这种情况。

聚合的优势在于，将传输到 ESM 的错误事件数量以及 ESM 中针对这些错误事件的相应错误响应配置（在上表详述获）在所有 CPU 总线上减少到两个错误事件（高优先级和低优先级）。尤其当器件中存在多个此类错误事件时，单独配置每个 CPU 总线产生的每个错误是没有必要的。因此，所有 CPU 总线上的错误被聚合在一起并提供给 ESM。除了聚合错误之外，分类也很重要，因为通过分类，用户可以将 ESM 配置为针对低优先级错误和高优先级错误分别生成适当的操作。

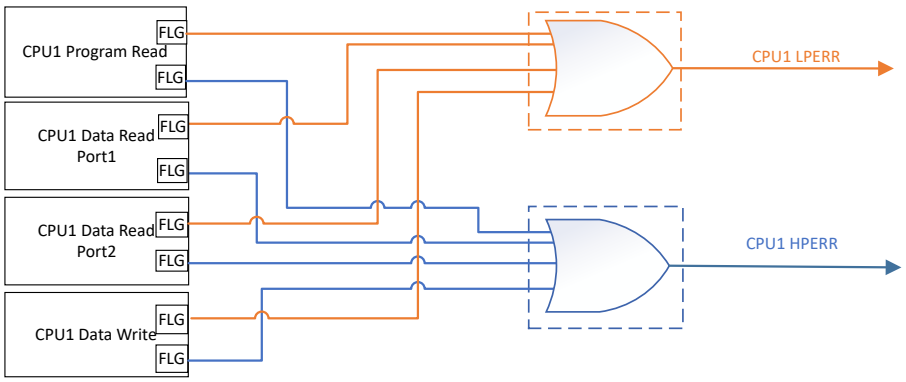

图 4-1. CPU1 EAM 模块错误聚合

备注

不要将 EAM 错误类型优先级与 ESM 输出优先级相混淆。要了解有关 ESM 输出优先级的更多信息，请参阅 *F29x 技术参考手册* 中的 ESM 一章。

4.2 错误记录

EAM 模块为用户调试器件中所有严重错误来源提供所需的全面错误记录功能。

所有错误都记录在 EAM 寄存器中并附带以下信息：

1. 错误类型 – 映射到特定错误类型（例如访问确认错误、不可纠正的错误等）的多位值。该值为每种类型预先定义，例如，0x20 表示发生 CPU PR 错误时的访问确认错误，如上面“错误聚合”一节中的表格所示。
2. 错误地址 – 发生错误的系统地址，用于调试错误来源。有单独的高优先级错误地址寄存器和低优先级错误地址寄存器。
3. 程序计数器（仅适用于 CPU EAM 模块） – 捕获的程序计数器地址有助于识别错误源。在识别与相应的 CPU 正在执行且导致错误的程序计数器地址相对应的代码时，PC 地址特别有用。

4.3 使用 EAM 模块进行错误调试

为了简化使错误调试，Code Composer Studio (CCS) 中集成了错误处理功能，如图 4-2 中所示。

用户可使用 CCS 的“脚本 (Scripts)”菜单查找捕获的错误状态。下面提到的所有错误调试步骤都是使用“脚本 (Scripts)”菜单中 GEL 文件可执行文件中的 Error_Agg_Check_Status() hotmenu 函数完成的，如图 4-2 中图所示。

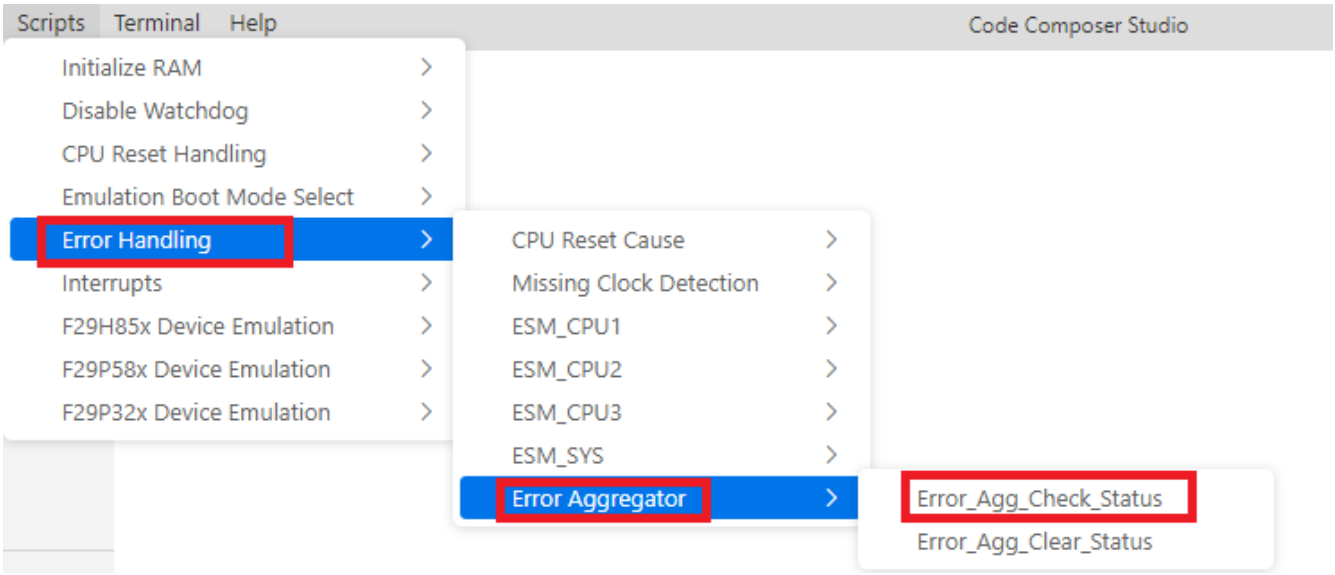

图 4-2. 错误聚合器 GEL 文件函数

1. 如果每个 EAM 模块（CPU PR/DR1/DR2/DW、RTDMA、SSU、CPU INT、Ethercat）的错误类型寄存器的值不是 0x0，请检查它们的值以确定是否发生了任何错误。
2. 如果特定的 EAM 模块错误类型寄存器的值不是 0x0，则查找相应的错误聚合器**低优先级错误地址**（低优先级错误）和**高优先级错误地址**（高优先级错误）。
3. 如果特定的 EAM 模块错误类型寄存器的值不是 0x0，则查找错误聚合器**程序计数器**地址（仅适用于 CPU EAM 模块）。

4.3.1 EAM 错误调试

1. 运行 Error_Agg_Check_Status() GEL 文件 hotmenu 函数，图 4-3 是 F29 SDK 中的 ESM 多核示例 (esm_ex1_cpu1_cpu3) 的 CCS GEL 输出。Error_Agg_Check_Status() 函数对输出执行以下操作：
   a. Error_Agg_Check_Status() 将错误类型值映射到错误。在本示例中（如 GEL 输出中所示），错误为 CPU3 DW 安全违例错误，因为在示例中，CPU3 代码写入 M0RAM 位置，这是不允许的。这是特意创建一个错误场景，以展示错误调试示例。
   b. 高优先级错误地址为 0x20000000，程序计数器为 0x10402C14，如下面的 GEL 输出所示，这是从相应的 CPU3_DW 高优先级错误地址和程序计数器寄存器中获取的。

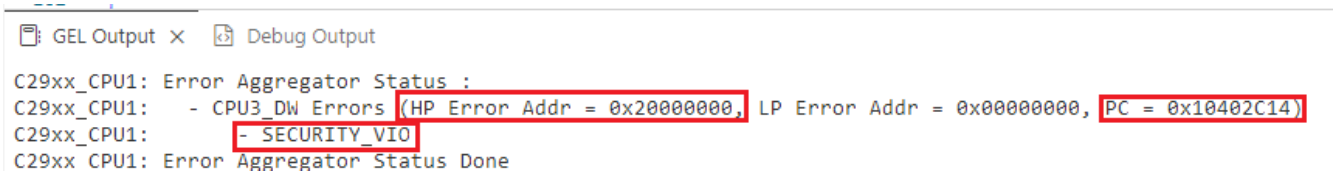

图 4-3. 错误聚合器检查状态 GEL 输出日志

2. 在从 CPU1 或 CPU3 清除 ESM/EAM 标志之前，需要运行 Error_Agg_Check_Status() GEL 函数，如下所示。为此，将断点放置在 ESM/EAM 标志清除函数执行之前，如下面的 NMI ISR – ESM/EAM 清除标志图中所示，以便在清除 EAM 寄存器之前由 GEL 函数读取和解码。

   Interrupt_clearEsmEaFlags() 是 F29 SDK 中提供的参考 driverlib 函数，它清除本示例的 CPU1 和 CPU3 NMI ISR（如 NMI ISR – ESM/EAM 清除标志中所示）中使用，并且也存在于 driverlib 默认 NMI 处理程序中的所有 EAM 和 ESM 标志。如果 Interrupt_clearEsmEaFlags() 函数已经执行，则用户可以查看 nmiStatus 结构（用于存储 ESM/EAM 错误标志寄存器值的存储器位置），以在 CCS 观察窗口中查找错误信息，如图 4-5 中所示。

备注

清除 EAM 和 ESM 标志是 NMI（不可屏蔽中断）ISR 中的一个重要步骤，因为这样可以避免 NMIWD（NMI 看门狗）超时并触发系统复位 (XRSn)。

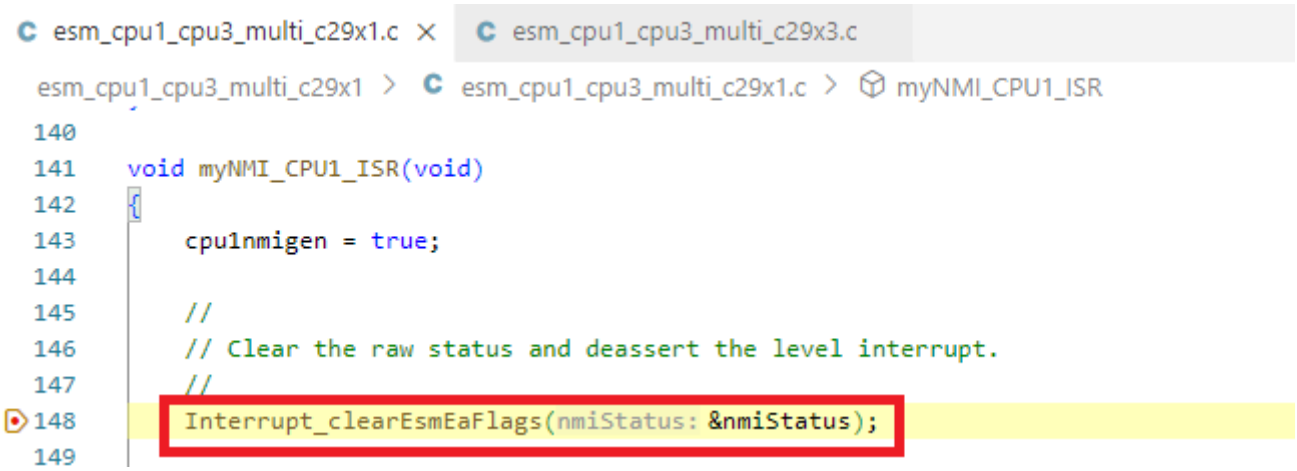

图 4-4. NMI ISR – ESM/EAM 清除标志

图 4-5. NMI 状态捕获日志

3. 以下是该示例中 CCS 的寄存器视图输出。
   a. 寄存器输出高优先级错误地址、程序计数器地址和错误类型值与错误聚合器检查状态 GEL 输出日志 — 图 4-3 中显示的 GEL 函数文件输出匹配。

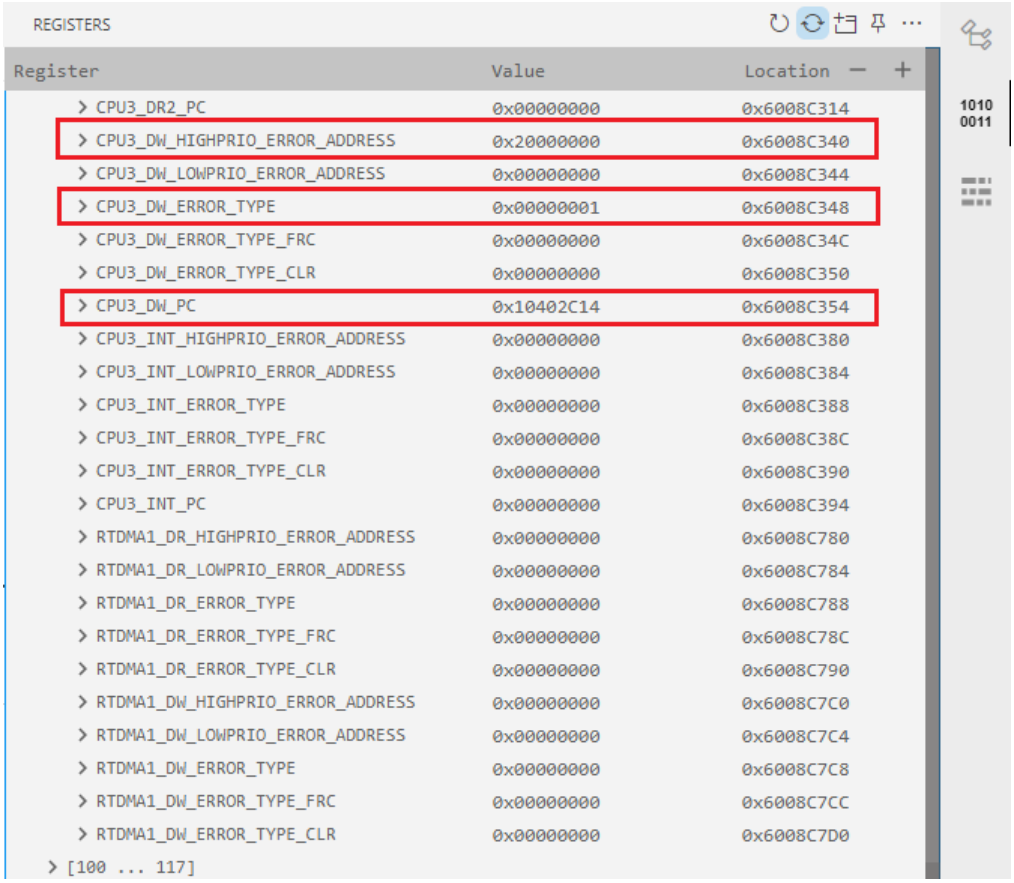

图 4-6. 错误聚合器寄存器的 CCS 寄存器视图

4.3.2 解读错误地址和程序计数器值

如第 3.2 节所述，错误地址和程序计数器地址可用于调试错误源。本节以 F29 SDK 中的 ESM 多核 (esm_ex1_cpu1_cpu3) 为例，展示如何解读错误地址和程序计数器。

可以将程序计数器 (PC) 地址复制到 CCS 反汇编视图，以查找发生错误的源代码。在此示例中，在查看 EAM 捕获的 PC 地址 (0x10402C14) 的相应 PC 地址 CCS 反汇编视图 (如图 4-7 中所示) 时，指向位置 0x20000000 (M0 RAM) 的数据写入操作，该位置也记录在 EAM 的高优先级地址寄存器内。因此，借助 PC 地址和错误地址信息，用户可以将问题精确指向导致错误的特定 CPU3 源代码写入操作。

CPU3 DW (数据写入) 总线上发生了错误，从代码角度来看，这也与预期行为相符，因为 CPU3 应用程序代码对 M0RAM 的 M0RAM_data 进行写入操作，这是 CPU3 代码不允许的。CPU3 仅具有 M0RAM 的读取数据权限，因此在这种情况下，写入操作导致了 CPU3 DW 总线上出现安全违例错误。

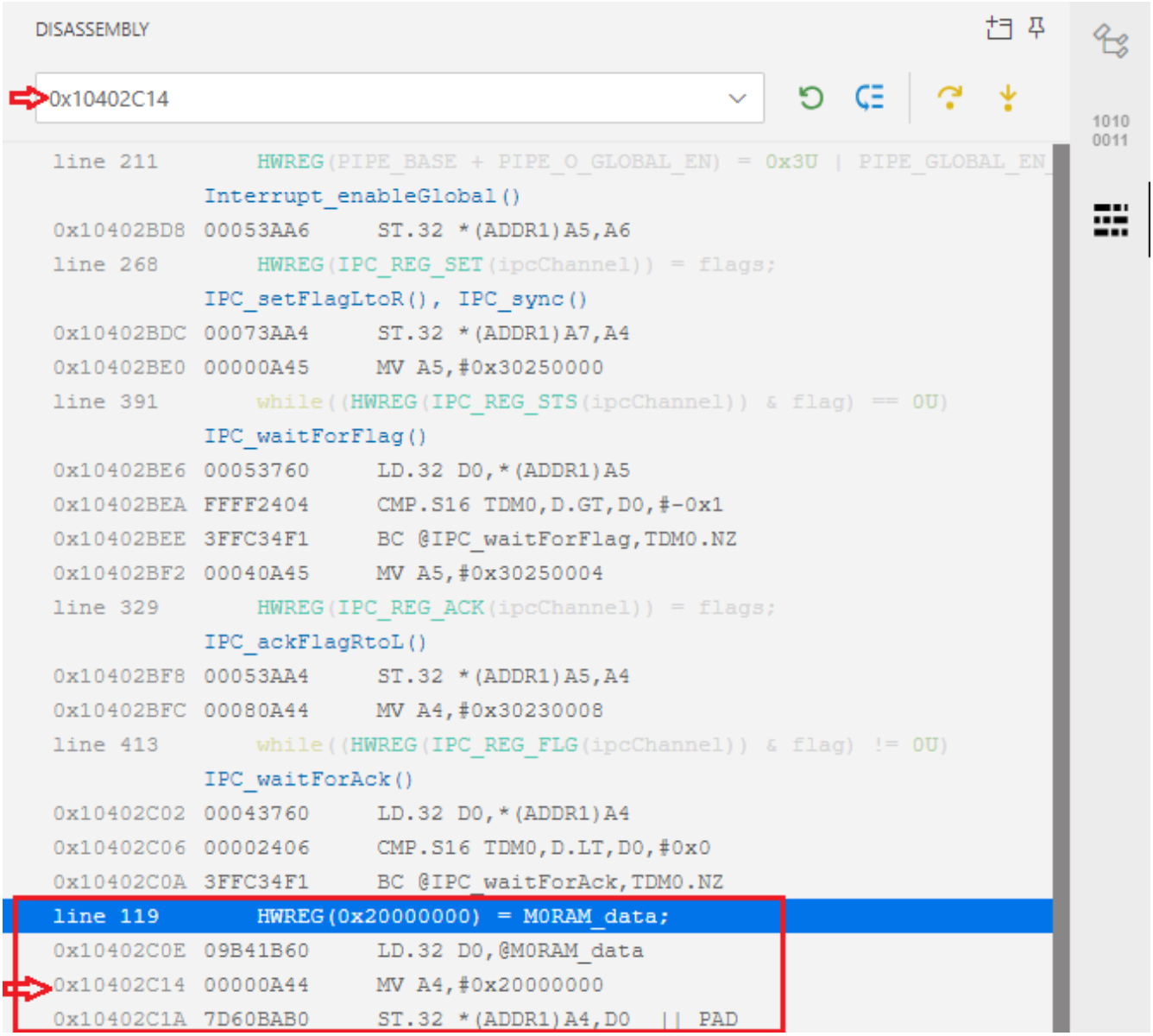

图 4-7. 程序计数器的反汇编视图

表 4-2. 从 CPU3 访问 M0RAM

存储器	交错式	CPU1	CPU2	CPU3	HSM	RTDMA1	RTDMA2
M0 RAM	是	0WS 数据（读写）	0WS 数据（只读）	3WS 数据（只读）	-	-	-

5 错误信令模块概述

错误信令模块 (ESM) 将对整个器件中错误事件的响应系统地整合到一个位置，这对于一些安全关键应用至关重要。

ESM 子系统包含以下模块：

1. ESM CPU1 - 输出到 CPU1 的专用 ESM 模块
2. ESM CPU2 - 输出到 CPU2 的专用 ESM 模块
3. ESM CPU3 - 输出到 CPU3 的专用 ESM 模块
4. 系统 ESM - 用于系统级输出的专用 ESM 模块（主要是 ERRORSTS 引脚输出、器件复位以及集成使用 XBAR 事件输出的其他模块）

图 5-1 详细介绍了 ESM 子系统如何在器件级集成，有关更多信息，请参阅 F29x TRM 中的 ESM 一章。

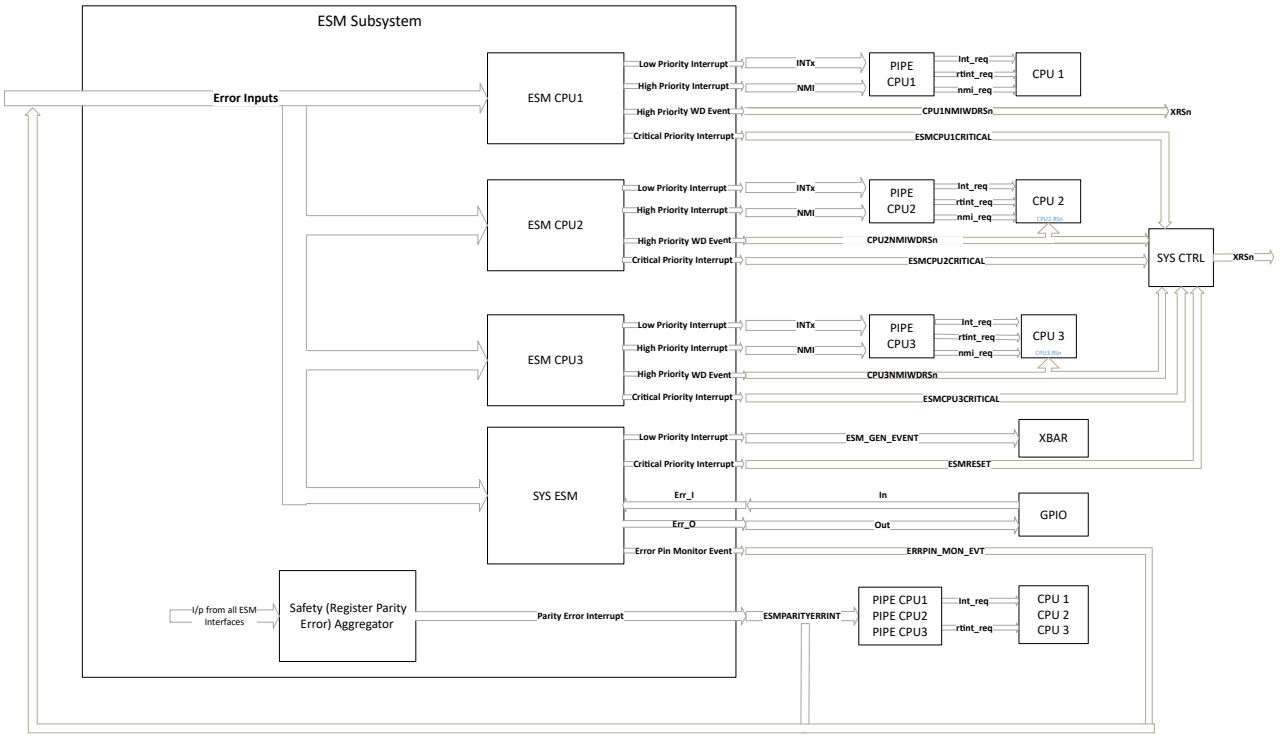

图 5-1. ESM 子系统集成方框图

ESM 提供了一些功能将错误按照严重性分类并提供可编程错误响应。错误信令模块提供了一种方法，根据所遇到的错误的严重性，向 CPU 指示错误引脚响应、可选的中断优先级响应或不可屏蔽中断 (NMI)。用户负责确定要为每个错误事件采取的错误响应，以便与系统安全概念保持一致。

1. 特定 CPU 中断：
   a. 中断（从 PIPE 到 CPU 的 INT 或 RTINT）（ESM 的低优先级中断输出）— 通常选择用于器件中遇到的可纠正或低严重性错误，或者可以实施以用于 CPU 外部的诊断。中断允许 CPU 外部的事件生成一个程序序列上下文，然后传输给中断处理程序，在那里软件有机会管理该故障。

b. 不可屏蔽中断 (NMI) (ESM 的高优先级中断输出) — 通常选择用于器件中遇到的不可纠正或严重错误，在这类情况下，需要进行错误响应以将上下文传输到 NMI ISR，并且软件有机会管理故障并安全地中止操作。

2. 错误信令引脚：
   a. PMIC (电源管理集成电路) 等外部监测器的错误引脚 (ERRORSTS) 操作，用于在所需响应为生成外部错误响应的情况下执行操作。
3. 复位
   a. 相应的 CPU 复位 (CPURSn)：ESM 能够为单个 CPU 生成复位，以便在检测到 MCU 中存在错误时使系统进入安全状态。
   b. 器件复位 (XRSn)：检测到错误时，触发器件复位 (XRSn) 以使 MCU 进入安全状态。

下一节简要概述了适用于 ESM 子系统中上述输出的 ESM CPU 和系统 ESM 模块的**配置**。有关更多详细信息，请参阅 *F29x 技术参考手册* 中的 ESM 一章和器件集成。

ESM 中错误事件的概述和须知要点：

1. 错误事件对所有 ESM 模块 (ESM CPU1/2/3 和系统 ESM) 通用
2. 每个 ESM 模块都有单独的**配置**和**状态**寄存器，因此所有 ESM 模块都可以彼此独立工作，从而针对不同的用例提供灵活性。例如，错误事件在发生时可配置为从 ESM CPU1 输出 CPU1 中断，而**不**配置为从 ESM CPU3 模块输出 CPU3 中断。
3. 错误事件进一步分成 32 组。F29x 器件总共有 256 个错误事件，因此总共有 8 个组。

备注

默认情况下，所有 Group0 错误事件都映射到触发器 NMI。Group0 错误事件是来自 EAM (错误聚合器模块) 的高优先级聚合 CPU 错误输出。

5.1 ESM 错误事件输出配置和状态信息

下图显示了如何配置相应的 ESM 块以影响相应 ESM 模块的可用输出。要进一步了解这些输出如何连接到器件外设，请参阅 *F29x 技术参考手册* ESM 一章中的 ESM 子系统器件集成图。

下面列出的状态寄存器有助于识别哪个错误事件处于活动状态，可以启用它们以影响 ESM CPU 和 Sys ESM 模块的输出：

1. 原始状态/设置寄存器 (RAW_j) - 指示错误事件是否处于活动状态，其中 j 表示错误事件索引 (j= 0 至 255)。
2. 中断使能状态/清除寄存器 (STS_j) – 指示错误事件是否处于活动状态，可以将其启用以影响低优先级中断或高优先级中断，其中 j 表示错误事件索引 (j= 0 至 255)。

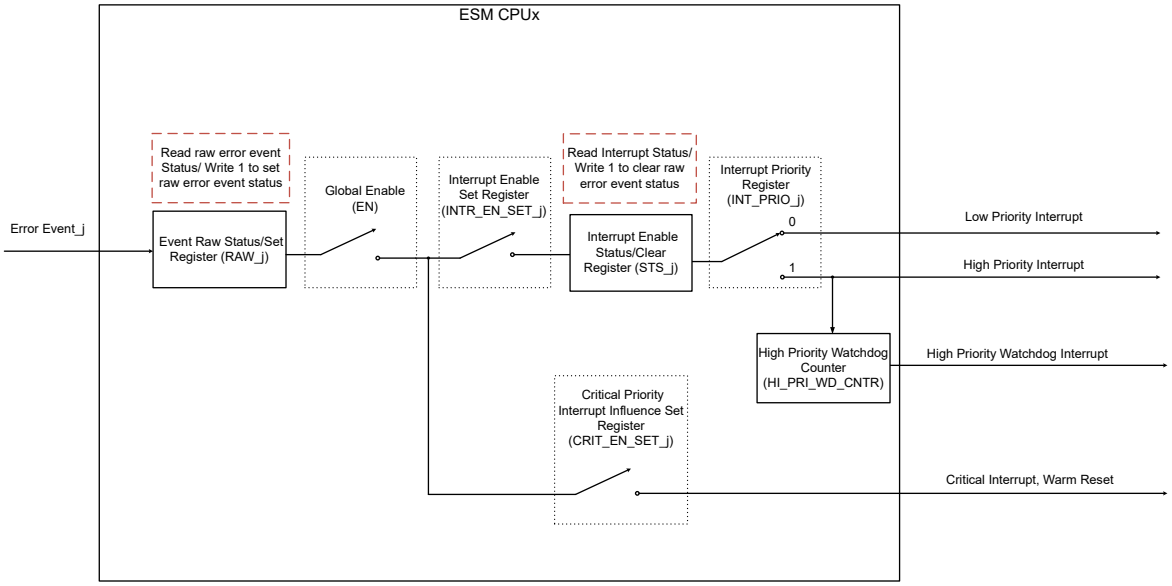

图 5-2. ESM CPU 详细配置和状态信息视图

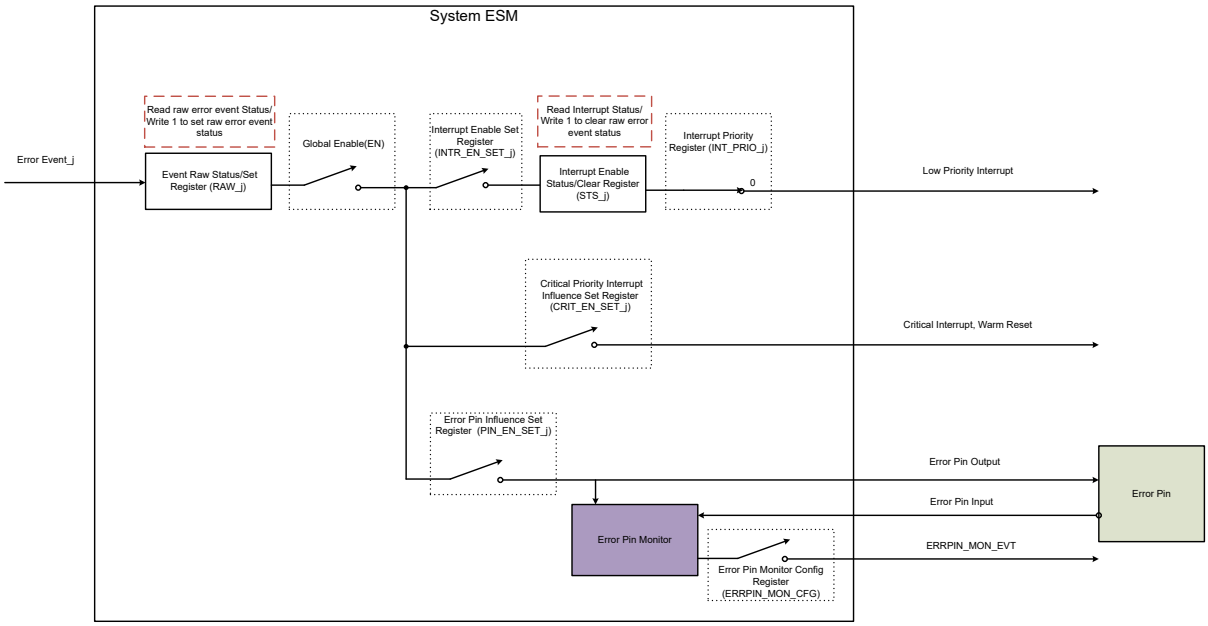

图 5-3. 系统 ESM 详细配置和状态信息视图

5.1.1 Sysconfig ESM 配置

Sysconfig 支持按照节 5.1 中的说明为各 ESM 模块的所需输出配置单独的错误事件。

例如，图 5-4 展示了如何针对错误事件 - Error Aggregator CPU1 HPERR 配置 ESM CPU1 以便获得高优先级 NMI 输出。相应 ESM CPU Sysconfig 模块中的全局参数设置可用于配置高优先级看门狗使能、看门狗计数器预加载值，并且可以定义低优先级中断和 NMI 输出的中断处理程序配置。

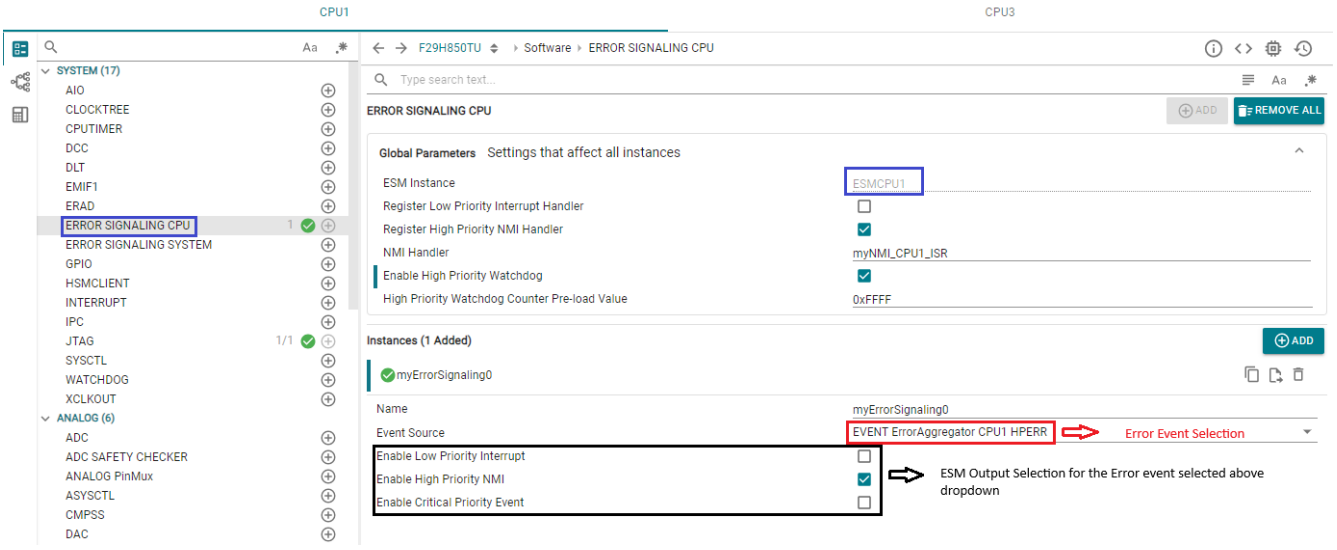

图 5-4. ESM CPU Sysconfig 模块

例如，图 5-5 展示了如何针对错误事件 - Error Aggregator CPU1 HPERR 配置系统 ESM，以实现对错误引脚的影响。也可以使用系统 ESM Sysconfig 模块中的全局参数部分来完成错误状态引脚 (ERRORSTS) 配置，例如极性、输出引脚模式配置等。

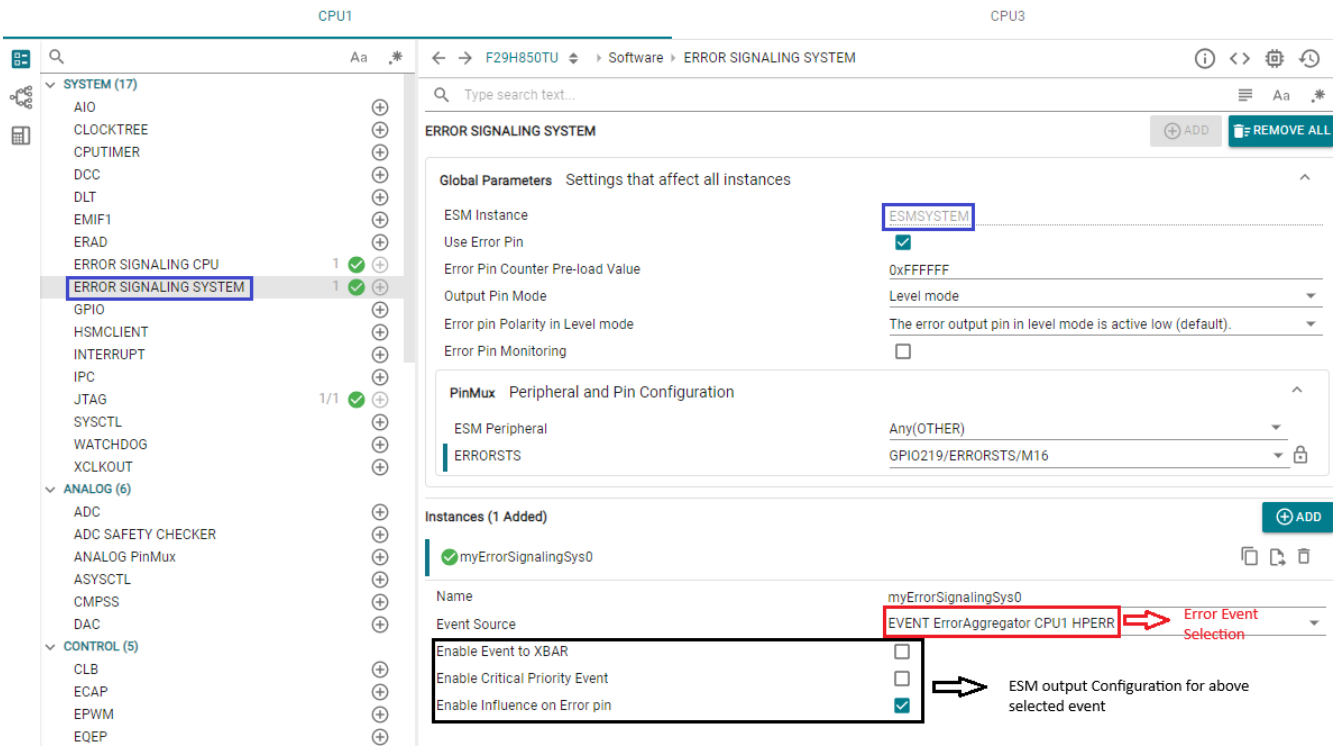

图 5-5. 系统 ESM Sysconfig 模块

5.2 ESM 错误事件调试

请按照以下步骤进行错误事件调试：

1. 在 CCS 中运行以下脚本 ESM_CPU_Check_Status() GEL 文件 hotmenu 函数，以检查错误事件的状态，如图 5-6 中所示。此函数输出指示的错误事件分为两类：
   a. **活动/待处理**错误事件 – 指示处于活动/待处理状态的错误事件
      i. 当错误事件处于活动状态意味着设置了错误事件的原始状态时，该函数会检查 F29x TRM ESM 错误事件表中所述每个事件的原始状态寄存器 (RAW_j)。
   b. **活动**、**待处理**和**已启用**错误事件 – 指示处于活动/待处理和已启用状态的错误事件。
      i. 当错误事件处于活动状态、待处理和启用状态意味着设置了错误事件的原始状态以及中断使能设置寄存器也由用户设置以触发来自相应 ESM 模块的中断输出。

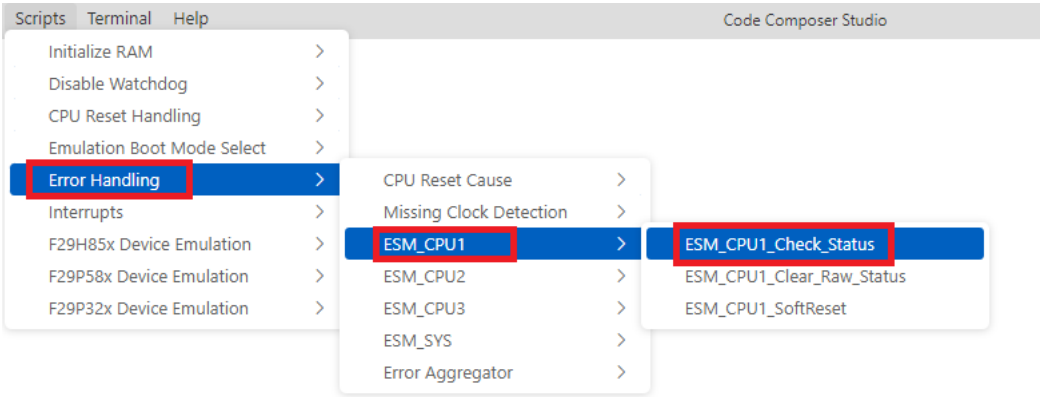

图 5-6. ESM 错误状态 GEL 函数

2. 下图显示了 GEL 输出的示例输出以及与 ESM 寄存器的相关性。这是 F29 SDK 中同一 ESM 多核示例的延续结。
   a. 图 5-7 显示 CPU1_ERAD_NMI 错误事件同时处于活动状态和启用状态。还会为 CPU1_ERAD_NMI 错误事件设置 ESM CPU1 - 中断优先级寄存器 (INT_PRIO)，以便为 CPU1（在 ESM CPU1 中）和 CPU3（在 ESM CPU3 中）触发 NMI。
   b. 除此之外，还有其他错误事件，例如 ErrorAggregator_CPU3_HPERR（CPU3 DW 总线上的安全违例错误导致的错误聚合器 CPU3 高优先级错误，如以上各节所述）、EPWMXBAR1、CPU1 高优先级中断和 CPU3 高优先级中断输出也处于活动状态，使用 ESM 原始状态寄存器 (RAW_j) 值对其进行解码。CPU1 和 CPU3 高优先级中断输出是 CPU1 和 CPU3 NMI 输出标志，而 EPWM XBAR 和 ERAD NMI 事件用于 NMI 勘误表权变措施实施，因此按预期激活，请查看 F29x 器件勘误表中的详细信息。

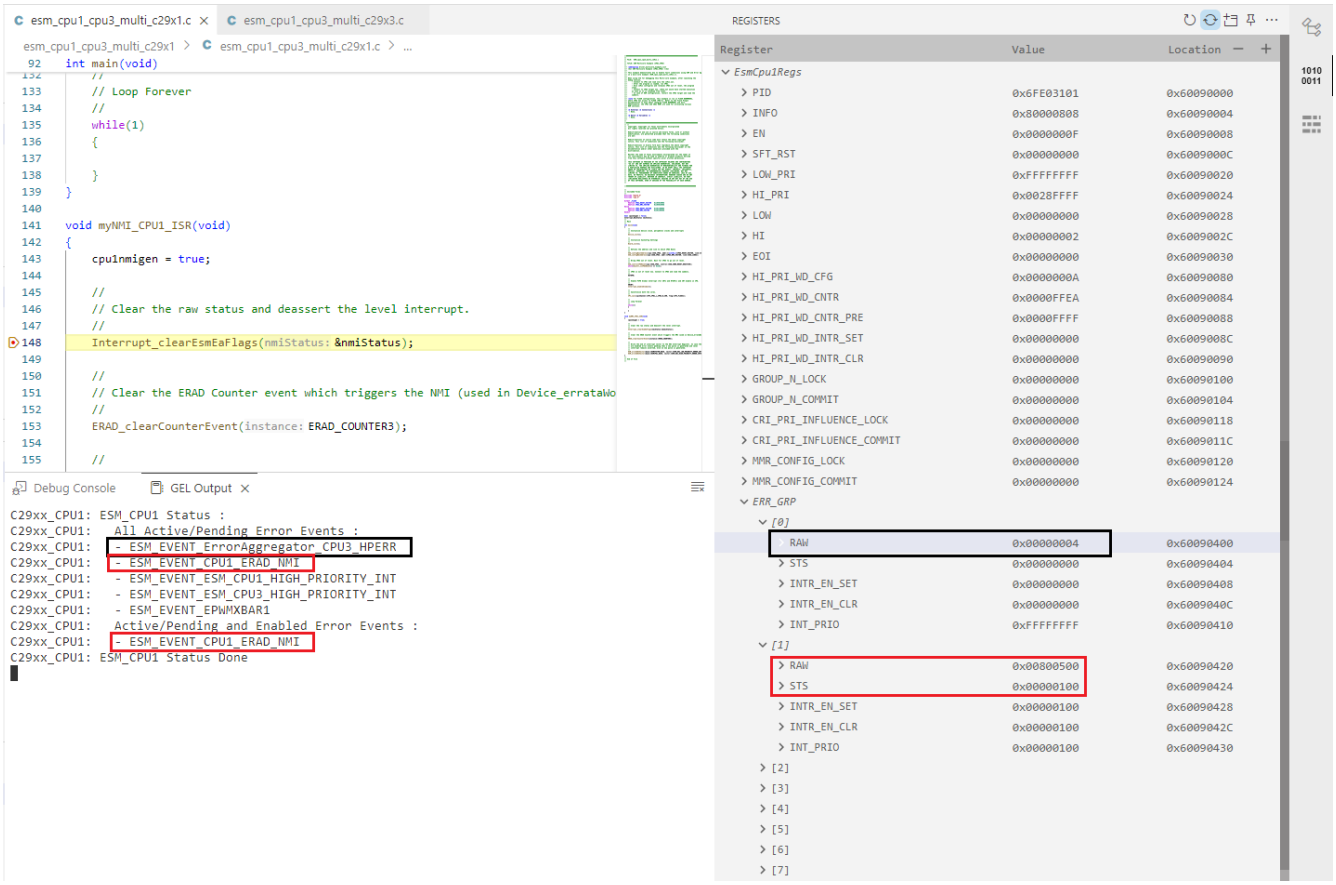

图 5-7. ESM 错误事件状态 GEL 输出

3. 与 EAM 寄存器标志类似，在 NMI ISR 中清除原始状态寄存器之前检查有无 GEL 输出，或者如前面所示检查相同的结构 (nmiStatus)，输出保存在该结构以便日后进行调试。如果特定事件配置为在活动状态时触发 NMI，则需要清除 ESM 原始状态 (RAW_j) 寄存器标志以避免 NMIWD 超时。

5.3 ESM 的其他调试技巧

1. 检查 RESC（复位原因）寄存器，以验证 ESM 高优先级看门狗中断 (NMIWD) 是否导致 NMIWDRSn 发生。如果针对任何错误事件的关键优先级中断输出配置系统 ESM，请检查 RESC 寄存器中的 ESMRESET 位。
2. 检查 ESM HI_PRI 寄存器，该寄存器显示最高优先级的未处理高优先级中断。最小的事件编号具有最高优先级，值 0xFFFF 表示没有高优先级中断处于活动/待处理状态。
3. 检查 ESM LOW_PRI 寄存器，该寄存器显示最高优先级的未处理低优先级中断。最小的事件编号具有最高优先级，值 0xFFFF 表示没有低优先级中断处于活动/待处理状态。

6 BootROM EAM 和 ESM 错误状态

如果应用程序无法在 NMIWD（高优先级看门狗）超时前清除错误，则 ESM CPU1 实例触发复位。在这种情况下，在器件复位 (XRSn) 后运行的 bootROM 清除错误以避免背对背 NMIWD REST 循环，并将错误信息和状态存储到 M0 RAM 中（请参阅下表），进行进一步调试。

BootROM 会清除以下状态：

1. ESM 子系统的 ESM CPU1 和系统 ESM 实例的 ESM Group0 原始状态
2. 所有 CPUx 错误聚合器类型寄存器

还会在 M0 RAM 中保存以下内容，以便用户调试错误源：

1. 仅 Group0 的 ESM 原始状态
2. 错误聚合器 CPU1 - PR、DR1/2、DW 和 INT 实例错误信息，包括高优先级错误地址、低优先级错误地址、错误类型和程序计数器寄存器

表 6-1. BootROM 错误状态信息

说明	地址
ESM 原始状态	0x2000_0868
CPU1 PR 错误聚合器高优先级错误地址	0x2000_086C
CPU1 PR 错误聚合器低优先级错误地址	0x2000_0870
CPU1 PR 错误聚合器错误类型	0x2000_0874
CPU1 PR 错误聚合器 PC 值	0x2000_0878
CPU1 DR1 错误聚合器高优先级错误地址	0x2000_087C
CPU1 DR1 错误聚合器低优先级错误地址	0x2000_0880
CPU1 DR1 错误聚合器错误类型	0x2000_0884
CPU1 DR1 错误聚合器 PC 值	0x2000_0888
CPU1 DR2 错误聚合器高优先级错误地址	0x2000_088C
CPU1 DR2 错误聚合器低优先级错误地址	0x2000_0890
CPU1 DR2 错误聚合器错误类型	0x2000_0894
CPU1 DR2 错误聚合器 PC 值	0x2000_0898
CPU1 DW 错误聚合器高优先级错误地址	0x2000_089C
CPU1 DW 错误聚合器低优先级错误地址	0x2000_08A0
CPU1 DW 错误聚合器错误类型	0x2000_08A4
CPU1 DW 错误聚合器 PC 值	0x2000_08A8
CPU1 INT 错误聚合器高优先级错误地址	0x2000_08AC
CPU1 INT 错误聚合器低优先级错误地址	0x2000_08B0
CPU1 INT 错误聚合器错误类型	0x2000_08B4
CPU1 INT 错误聚合器 PC 值	0x2000_08B8

7 常见问题解答：

1. 用户是否设置和配置 NMI ISR？

   解答 – 是的，建议用户在器件初始化期间设置 NMI ISR，这样，当发生高优先级错误（例如 Group0 错误事件）并触发相应 CPU 的 NMI 时，就可以按照 F29x TRM 中的指导和 ESM 多核 F29 SDK 示例，慢慢清除 EAM 和 ESM 中的错误标志。如果未能清除 ESM 原始状态标志，会导致 NMIWD 超时并触发 XRSn（器件复位）。用户可以检查 RESC（复位原因）寄存器中的 NMIWD 位以确认这一点。
2. 如果没有应用程序进行的 NMI ISR 设置并且没有基于组 0 ESM 错误事件的默认设置由错误事件导致的 NMI，会怎样？

   解答 — 如果没有用户/应用程序 NMI ISR 设置，则 CPU 转至 BootROM 中的默认 NMI 处理程序，这时，CPU 清除并将错误状态和标志保存在 M0 RAM 地址中以进行调试。有关更多信息，请参阅 BootROM TRM 一章。
3. 如果配置了 NMI ISR 但未清除错误标志，会怎样？

   解案 – ESM NMIWD 超时发生并导相应 CPU 产生高优先级看门狗中断输出。ESM CPU1 高优先级看门狗中断输出连接以导致 XRSn，而 ESM CPU2/CPU3 产生相应的 CPURSn，如 ESM 子系统集成视图中所示。
4. 如果没有将 ESM 配置为针对高优先级 CPU EAM 错误（作为组 0 错误事件传输给 ESM）产生相应 CPU 的 NMI，会怎样？

   解答 – 当 ESM Group0 错误事件处于活动状态并传输通过 EAM 模块，但未配置为产生 NMI 时，CPU 进入故障状态。处于活动状态的 CPU EAM 模块高优先级错误应配置为始终触发相应 CPU 的 NMI。
5. ESM/EAM 标志在 XRSn（设备重置）或 CPURSn（CPU 重置）之后是否清除？

   解答 – 不是，错误类型寄存器（本文档中称为 EAM 错误标志）和 ESM 原始状态寄存器 (RAW_j)（称为 ESM 错误标志）仅由 PORESETn 复位。即使在 XRSn 和 CPURSn 之后，这些标志也会保留值，因为应用程序需要这类错误标志信息，才能在由 ESM 触发复位并作为对应用程序软件未处理的错误事件响应的情况下进行调试。
6. 为什么即使在清除 ESM 原始状态寄存器后，对于导致 NMI 的所有事件，CPU 仍继续进入 NMI ISR？

   解答 – 在清除 ESM 原始状态寄存器 (RAW_j) 标志后，确保也使用相应的错误 ESM 中断输出适当键写入 EOI 寄存器。此写入 EOI 的步骤基本上是用户确认 ESM 中断输出，并将 ESM 输出置为无效。如果未对 EOI 寄存器进行写入，即使 ESM 错误事件标志被清除，ESM 中断输出也会保持有效。
7. 根据发生的特定错误事件，是否仅可以产生 CPU1 的 NMI 而不是 CPU3 的 NMI 并且反之亦然？

   解答 – 是的，ESM 是高度可配置的，并且由于每个 CPU 都有单独的专用 ESM 逻辑块，因此可以在初始化期间设置配置，以便从各自的 ESM CPU 逻辑块产生任一 CPU 的 NMI 或任何其他错误响应，同时禁用另一个 ESM CPU 逻辑块针对特定错误事件产生错误响应或配置不同的错误响应。

8 总结

ESM 和 EAM 为用户提供了系统性方法，以便在器件中发生错误时即时处理。应用手册重点介绍了错误事件如何从错误源（检测到错误之处）传输到 ESM 以及 ESM 如何向器件提供错误响应。最后，本文档详细介绍了可用于识别错误源的调试工具。

9 参考资料

1. 德州仪器 (TI)，*F29H85x 和 F29P58x 实时微控制器* 技术参考手册
2. 德州仪器 (TI)，*F29H85x-SDK：CPU1 和 CPU3 之间的 ESM 示例*
3. 德州仪器 (TI)，*CCSTUDIO：Code Composer Studio™ 集成式开发环境 (IDE)*
4. 德州仪器 (TI)，C2000™ SysConfig

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月