Application Note

通过 C29x 功能安全和信息安全单元实现运行时功能安全和信息安全

Ibukun Olumuyiwa, Marlyn Rosales Castaneda, and Aishwarya Rajesh

摘要

功能安全和信息安全单元 (SSU) 是 F29x 器件中的集成模块，可为应用代码启用运行时功能安全和网络安全保护。借助 SSU 的功能，能够在硬件中实现强大的无干扰 (FFI) 功能、安全任务隔离、调试安全和固件更新保护，并保持实时控制系统所需的低延迟性能。SysConfig 工具是 F29 SDK 的一部分，提供了一个易于使用的图形用户界面 (GUI)，用于配置 SSU 和在用户应用中启用功能安全和信息安全保护。本应用手册探讨了 SSU 的各种功能，以及嵌入式系统开发人员如何使用 SysConfig 中的 SSU 工具在实时应用中设计和实施运行时功能安全和信息安全保护。

内容

插图清单

商标

AUTOSAR® is a registered trademark of AUTOSAR Development Partnership.
所有商标均为其各自所有者的财产。

1 简介

德州仪器 (TI) C29 CPU 为实时控制应用提供出色的性能。C29 采用 128 位超大指令字 (VLIW) 架构、64 位定点和浮点运算、超低延迟处理和硬件中断优先级，可以很好地运行要求严苛的汽车和工业控制应用。SSU 与 C29 CPU 相结合，可在不影响实时性能的情况下帮助系统设计人员满足实时控制应用中严格的现代功能安全和信息安全标准。利用 SSU，用户可以实现真正的 FFI、安全任务隔离以及高级调试和固件更新安全性，同时保持要求严苛的实时控制系统所需的高速和低延迟处理。

本应用手册介绍了如何使用 C29x CPU 和 SSU 在实时控制系统中实现运行时应用程序的功能安全和信息安全。C29x SSU 架构提供了动态上下文相关内存保护、多个专用 CPU 栈指针的安全任务隔离以及多用户调试 ZONE 以实现信息安全。

2 补充性在线信息

有关特定器件上 C29x CPU 和 SSU 的详细说明，请参阅器件特定数据表和相应的技术参考手册 (TRM)。

本应用报告是使用 F29H85x 系列器件撰写的。F29 SDK 和 SysConfig 工具支持所有 F29x 平台器件。

- *C29x CPU 和指令集用户指南*
- *F29H85x 和 F29P58x 实时微控制器数据表*
- *F29H85x 和 F29P58x 实时微控制器技术参考手册*
- *C29x Academy：功能安全和信息安全单元 (SSU) 一章*

TI E2E™ 社区还提供了其他支持。

3 SSU 概述

SSU 是 C29 CPU 子系统不可分割的一部分，提供上下文关联内存保护和运行时代码隔离、调试安全性和安全固件更新。SSU 充当 C29 CPU 与系统其他部分之间的防火墙，执行用户访问保护策略，管理调试访问和闪存控制器操作。图 3-1 展示了集成在 C29 子系统中的 SSU 的简化概览。SSU 的可用功能可用于在实时控制系统中实现真正的 FFI，而不会增加软件开销，否则会对实时性能产生负面影响。利用 SSU，系统设计人员可以在同一个 CPU 内核上组合多个控制和通信功能，同时保持这些功能相互隔离。这可以减少实现系统目标所需的内核数量，或有助于实现更高的系统安全完整性级别。

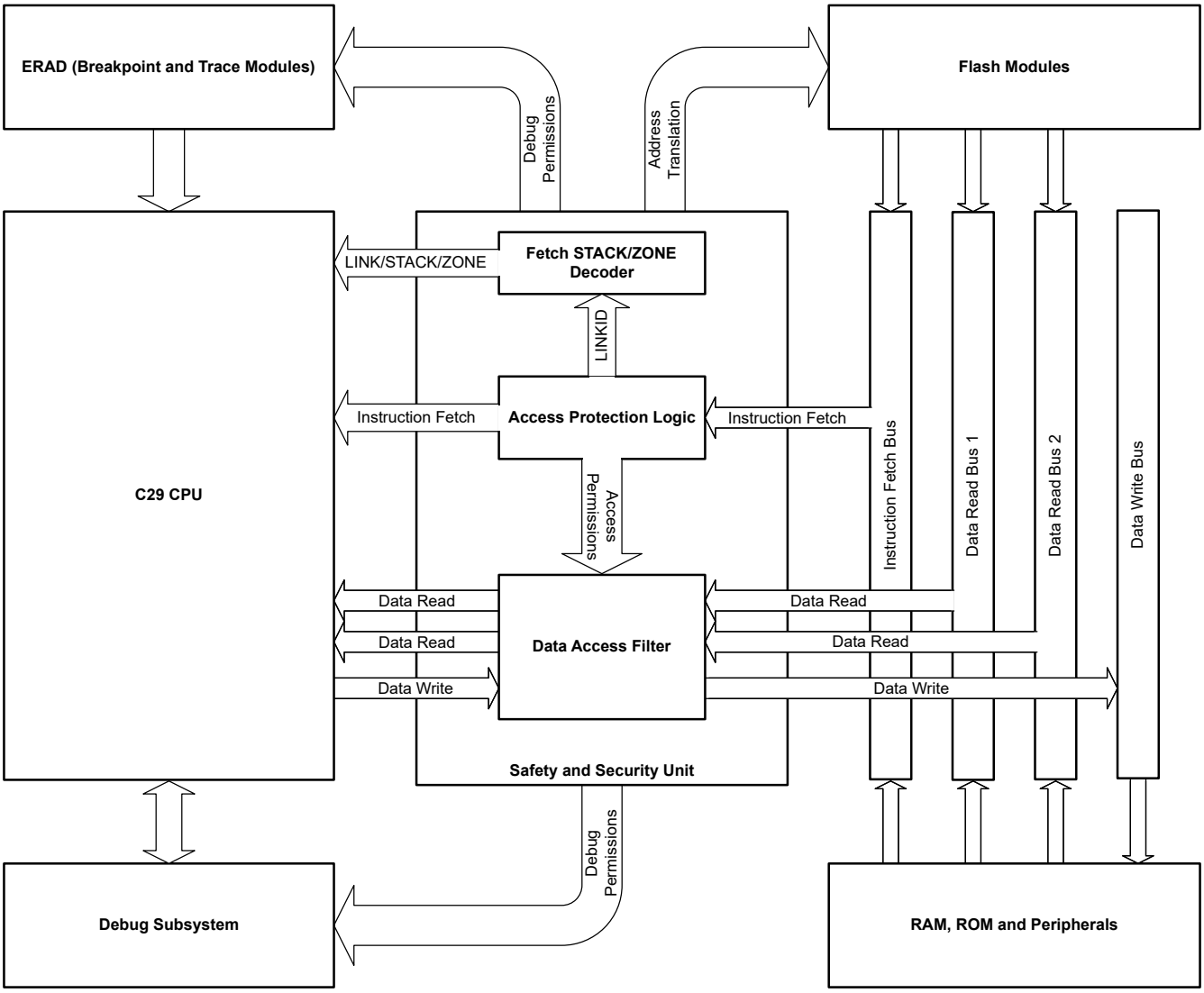

图 3-1. SSU 系统方框图（简化视图）

4 关键概念定义

本节包含关键概念的定义。

访问保护范围 (APR)	这是 SSU 中内存保护的基本单位。访问保护范围涵盖闪存、SRAM 或外设的特定区域。每个 APR 定义每个 LINK 的读取和写入访问权限。APR 也可以配置为代码区域，这样可以从该内存区域获取 CPU 指令。
LINK	在 C29 CPU 子系统中，LINK 构成了上下文相关内存保护的基础。每个 LINK 可以表示一个或多个可执行代码区域。相关 LINK 标识符用于确定该代码可以访问哪些数据内存区域 (APR)。
STACK	STACK 使代码执行上下文相互隔离。每个 STACK 在 C29 CPU 中都有一个专用栈指针，并可从硬件层面与其他 STACK 中的代码实现功能安全和信息安全隔离。每个 LINK 都属于一个且仅属于一个 STACK，但一个 STACK 可以包含多个 LINK。
区域	ZONE 决定调试和固件更新权限。每个 CPU 的 APR、LINK 和 STACK 都是独立定义的，而 ZONE 则跨越整个器件，不包括硬件安全模块 (HSM)（该模块不受 SSU 管理）。
HSM	硬件安全管理器是器件内的自包含子系统，提供关键安全服务，包括安全启动、安全存储、调试和固件更新安全性以及运行时加密服务。HSM 与 SSU 不同，SSU 是应用程序 C29 CPU 子系统的组成部分。HSM 和 SSU 在器件上扮演互补和正交的角色，但调试授权除外：HSM 和 SSU 必须先授权访问资源，然后才能对该资源进行调试访问。
部分调试	当某个 ZONE 启用部分调试时，允许用户调试 CPU 的执行（停止、恢复和查看 CPU 寄存器），但阻止用户对该 ZONE 中的 LINK 可以访问的内存进行调试读取和写入访问。
完全调试	当某个 ZONE 启用完全调试时，用户可以调试 CPU 并执行该 ZONE 内的任何 LINK 允许的所有内存访问。
SECCFG	这是一个特殊的闪存区域，指定用于存储 SSU 配置设置。在器件启动期间，存储在 SECCFG 区域的值将加载到 SSU 寄存器中。这些设置大多无法在运行期间更改，只能通过将新值编程到 SECCFG 中并重置器件来修改。
UPP	用户保护策略。这是编程到 SECCFG 区域的 SSU 配置设置集合。
内存区域	SysConfig 中配置的内存区域。这相当于访问保护范围 (APR)。
模块	在 SysConfig 中，模块由一个 LINK、与该 LINK 相关联的代码内存区域（可执行 APR）、*属于* 该模块的数据内存区域（数据 APR）和外设，以及与模块相关联的外设中断组成。在实践中，模块允许用户将应用程序组织为不同的任务或分区，这些任务或分区可以相互隔离，以保证功能安全和信息安全。
共享内存	在 SysConfig 中，共享内存包含一个或多个可由多个模块访问的 APR。共享内存可用于在不同内存范围的模块之间共享数据，同时为属于这些模块的其他内存区域提供安全保护。
沙盒	在 SysConfig 中，沙盒由一个 STACK 组成，并且可以包含一个或多个模块。
RTOS	实时操作系统，如 FreeRTOS® 或 AUTOSAR®。

5 功能安全和信息安全目标

功能安全和信息安全单元使系统设计人员能够在实时嵌入式系统设计中实现与功能安全和信息安全相关的重要目标。这些目标包括：

1. **内存保护：**嵌入式微控制器中支持功能安全目标的基本要素是内存保护单元，即 MPU。MPU 对系统中的内存强制执行访问控制规则，以防止未经授权的读取、意外覆盖或对代码和数据进行未经授权的修改。内存保护在维持系统稳定性、可靠性和安全性方面发挥着重要作用。SSU 提供上下文关联的高级 MPU 功能，无需软件干预即可实时提供切换保护。
2. **无干扰：**ISO 26262 标准定义了汽车电子产品的功能安全标准，其中“无干扰”(FFI) 被定义为 *“两个或多个元件之间不存在可能导致违反安全要求的级联故障”*。级联故障是指系统中一个元件的故障会导致系统中的另一个元件发生故障；这些故障会导致一个逐渐扩大的正反馈回路。SSU 提供了将多个不同的系统软件组件完全相互隔离的机制，从而使一个组件的安全故障不会危及应用程序的其他部分。
3. **安全隔离：**除了功能安全不受干扰外，SSU 还支持信息安全隔离目标，为每个应用组件提供安全的执行环境，在运行期间保护代码和数据资产的机密性和完整性。
4. **实时性能：**SSU 的一个重要目标是在不影响实时性能的情况下提供功能安全和信息安全保护。内存保护、信息安全隔离和其他 SSU 功能都是在没有软件干预的情况下实时执行，从而消除了由于监控软件开销造成的额外延迟。结合 C29 CPU 出色的性能，系统设计人员可以在不牺牲性能、功能安全或信息安全目标的情况下，在同一个 CPU 上实现多种控制功能，从而降低整体系统成本。
5. **安全调试和固件更新：**SSU 提供了将系统软件划分为多个用户调试 ZONE 的功能，使多个团队能够在同一芯片上安全地维护和调试不同的软件组件。SSU 还管理闪存固件，控制允许执行固件更新的用户和代码，并启用固件无线更新 (FOTA) 和实时固件更新 (LFU) 等机制，同时在硬件中提供 A、B 交换和回滚保护。

6 系统设计

为应用程序配置 SSU 的第一步是确定所需的系统分区。SSU 为应用程序子系统的分区提供了三个层次：

1. ZONE ：每个 ZONE 决定芯片上所有 C29 CPU 的调试访问权限。ZONE 的设计目的是使多个代码所有者或实体能够开发和维护驻留在同一芯片上的应用程序的不同分区。例如，如果嵌入式应用的某个方面由第三方供应商拥有和维护，那么系统就可以分为两个 ZONE：
   a. ZONE1：主用户 ZONE，由主系统开发商拥有；
   b. ZONE2：辅助用户 ZONE，由第三方开发商拥有。

   这种分区使第三方开发商能够在同一芯片上开发、调试和维护应用程序功能，而无需访问主用户的代码和数据资产。此外，每个用户 ZONE 还提供两级调试授权：

   a. 部分调试 - 允许使用 CPU 调试命令，如停止、步进和断点，但不能访问内存
   b. 完全调试 - 允许访问 ZONE 内所有 LINK 的内存位置。

   例如，第三方开发人员等辅助用户可在 ZONE2 中调试应用模块，同时也可获得 ZONE1 的部分调试访问权限，这样辅助用户就能在上下文中有效调试应用，而无需访问主用户的资产。

   每个器件都有 3 个可用的用户 ZONE：ZONE1、ZONE2 和 ZONE3。ZONE1 是主用户 ZONE；ZONE2 和 ZONE3 是辅助用户 ZONE。
2. 沙盒 (STACK)：沙盒用于在 CPU 内提供安全隔离。每个沙盒都与 SSU 中的一个 STACK 相关联。每个沙盒在 CPU 中都有一个专用的物理栈指针，其他沙盒无法访问该指针，同时还有一个专用的栈内存 AP 区域，只有属于该沙盒的代码对该区域具有读取/写入权限。

   从一个 STACK 跨到另一个 STACK 时，需要使用特殊的 C29 CPU 门指令。编译器必须在每个函数的入口和出口以及函数调用或分支处插入这些指令。这些机制可提供安全保护，防止试图重定向代码执行或操纵栈的恶意软件攻击。

   沙盒由 SSU STACK 以及与 STACK 相关的所有内容（包括栈内存 AP 区域）组成。每个 STACK 属于一个 ZONE，但一个 ZONE 可以包含多个 STACK

表 6-1. 每个 CPU 的预定义 STACK

STACK #	说明
STACK0	此 STACK 保留供 TI 内部使用，无法由用户配置。
STACK1	此 STACK 主要用于引导加载程序，但也可以与其他用户应用程序代码相关联。STACK1 始终与 ZONE1 关联，仅包含一个 LINK (LINK1)。
STACK2	这是主用户 STACK。STACK2 始终与 ZONE1 关联。STACK2 始终包含 LINK2，但也可以包含其他 LINK。

3. 应用模块 (LINK)：应用模块是系统应用程序的基本分区。每个模块均由单个 SSU LINK、一个或多个包含 LINK 代码的代码内存 AP 区域、与 LINK 相关的所有数据内存 AP 区域以及与模块相关的所有外设和中断组成。

   通常，代码 AP 区域包含 .text 和其他包含代码的链接器输出段，数据 AP 区域包含 .bss、.const 和其他包含数据和变量的链接器输出段。

   每个 LINK 都能启用 SSU 内存保护，提供 CPU 中其他 LINK 的安全保护。每个 AP 区域都为每个 LINK 定义了访问权限。这些权限会根据 LINK ID 指令，在每条执行内存访问的指令中实时执行。需要相互功能安全隔离的功能可以放在单独的模块中。如果需要信息安全隔离，则将这些模块置于单独的沙盒中；如果不需要，则可将这些模块置于同一沙盒中。

表 6-2. 每个 CPU 的预定义 LINK

LINK #	说明
LINK0	此 LINK 保留供 TI 内部使用，无法由用户配置。

表 6-2. 每个 CPU 的预定义 LINK（续）

LINK #	说明
LINK1	此 LINK 主要用于引导加载程序，但也可以与其他用户应用程序代码相关联。CPU1.LINK1 具有一些特殊的固定权限，除了 AP 定义的保护外，还能访问某些系统配置寄存器。
LINK2	这是主用户 LINK。CPU1.LINK2 是系统安全根 LINK (SROOT)，具有特殊的固定权限，可访问系统配置寄存器和覆盖控制。该 LINK 通常在 RTOS 级别执行特权主机功能。

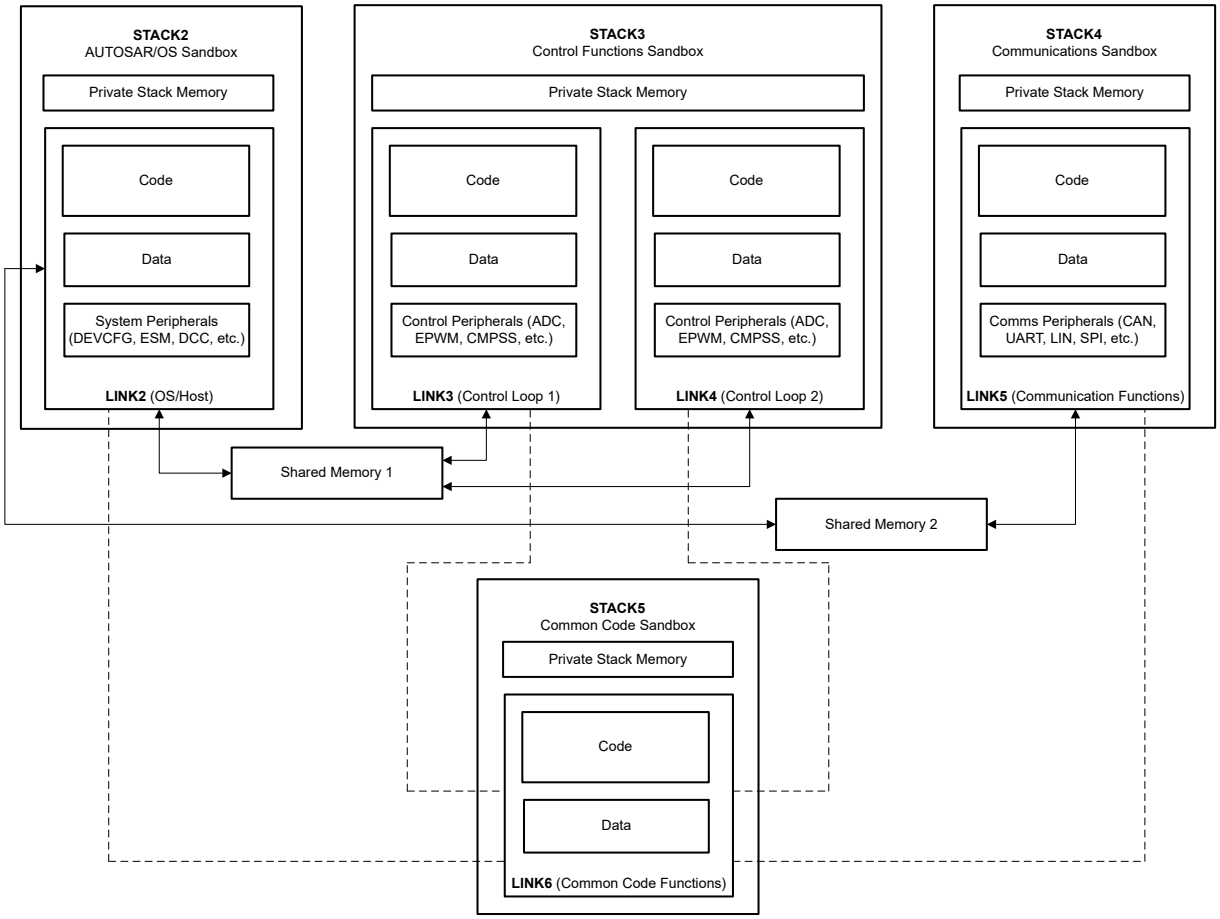

图 6-1. 使用 SSU 进行软件分区的示例

图 6-1 展示了单 CPU、单 ZONE 系统上的 SSU 系统分区示例，并且表 6-3 介绍了该配置背后的详细信息。

表 6-3. 所解释的 SSU 分区示例

STACK	LINK	详细信息
STACK 2	LINK2	RTOS 在 STACK2.LINK2 中运行，负责初始化系统配置、设置外设和中断，并启动主执行循环。虽然可以将各种任务和各自的任务栈放置在其他 LINK 中，但所有 RTOS 任务/栈都需要位于 SSU STACK2 中。如果使用了 RTINT，则与 RTINT 相关的代码和数据应置于 SSU STACK2 以外的单独 SSU STACK 中。
STACK 3	LINK3 和 LINK 4	有两个控制功能：*Control Loop1* 和 *Control Loop 2*。这些控制功能中的每一个都放在单独的应用模块 (LINK) 中，并且两个应用模块放在同一沙盒中。在此系统中，两个控制功能之间需要物理安全隔离，但不需要信息安全隔离。
STACK 4	LINK 5	LINK5 包含主机通信代码，例如 UART 或 CAN-FD 代码。由于来自外部接口的数据可能会对系统中的其他功能构成安全威胁，因此该模块被置于单独的沙盒中。

表 6-3. 所解释的 SSU 分区示例 （续）

STACK	LINK	详细信息
STACK 5	LINK 6	包含系统中所有其他模块之间共享的公共代码函数。LINK 6 被定义为其他 LINK 的访问保护继承 LINK (APILINK)。通用代码模块也被置于单独的沙盒中，以保持与系统其他部分的信息安全隔离（同时保持继承的权限）。建议将通用代码 LINK 放置在其自身的 STACK 中。

SysConfig 完全支持多核应用。内置内存分配器工具可自动管理多个 CPU 上与应用模块相关的内存区域，还可管理整个器件的外设分配。SysConfig 工具还包括共享内存功能，用于定义可在同一 CPU 或多个 CPU 上的模块之间共享的内存区域。

7 配置 SSU

7.1 闪存 SECCFG 区域

闪存 SECCFG 区域用于存储用户保护策略 (UPP)。这是 C29 应用闪存组的一个特殊 NONMAIN 区域，专门用于 SSU 配置和引导设置。编程到 SECCFG 中的设置会在器件启动时加载到 SSU 内存映射寄存器中，并且大多数情况下，在下一次器件重置之前始终保持锁定状态。器件中的每个主 CPU（即奇数 C29 CPU，例如 CPU1、CPU3）都有两个 SECCFG 扇区：基本扇区和备用扇区。这两个扇区被设计为可以在一个扇区处于活动状态时，擦除另一个扇区并用新的配置值对其进行编程。

备注

请勿尝试擦除当前处于活动状态的 SECCFG 扇区并对其重新编程。如果在擦除和编程过程中发生器件复位，则器件随后将无法启动并无法运行。始终将新配置编程到备用 SECCFG 扇区地址中。在编程和擦除操作期间，闪存地址转换逻辑会自动将该地址路由到当前处于非活动状态的 SECCFG 扇区。SysConfig 工具会自动将 SECCFG 映像分配给生成的 .out 文件中的备用扇区，以启用正确的更新过程。

备注

F29x 器件上有四种不同的闪存组模式。有关不同组模式的更多详细信息，请参阅 F29x 技术参考手册。要针对相关应用场景正确配置组模式，请在 CCS Debug 视图中右键单击 Connections 内的 CPU1，然后选择“Properties”选项。在属性窗口中选择“Flash Settings”下拉菜单。转至“Bank Mode”并选择所需的组模式。单击“Program Bank Mode”。

为保护 SSU 用户保护策略的完整性，SECCFG 扇区包含一个 CRC 值，在引导时会检查该值。该 CRC 包括访问保护设置、LINK 和 STACK 配置、闪存写入和闪存擦除保护、闪存更新权限、调试设置、启动设置和 SSU 运行模式。调试密码不包括在此 CRC 计算中。

有关 SECCFG 扇区的全面地图，请参阅器件技术参考手册。

7.2 SSU 开发生命周期

SSU 可配置为三种工作模式之一：SSUMODE1、SSUMODE2 和 SSUMODE3。这些操作模式旨在方便开发过程，协助用户在系统设计中实施功能安全和信息安全功能。只要用户拥有更新 SECCFG 扇区的必要权限，就可以重新配置 SSU，将其更改为任何其他工作模式。

德州仪器 (TI) 出厂的器件最初处于 SSUMODE1 模式。在这种模式下，整个内存映射范围都映射到 LINK2（安全根 LINK），所有 LINK 对所有 AP 可配置内存区域都具有完全读取和写入权限。即使在 SSUMODE1 中，硬编码保护也仍然有效；不过，由于所有代码都以 LINK2 运行，因此实际上对用户代码没有任何限制。

在 SSUMODE2 中，会强制执行 AP 区域保护，但调试和闪存更新保护仍被禁用。为获得更佳效果，应首先在 SSUMODE1 中全面验证应用程序功能，然后在 SSUMODE2 中实施 SSU 设置并进行测试。完成运行时功能安全和信息安全设置验证后，即可配置调试密码，并将器件置于 SSUMODE3 模式。在此模式下，调试端口默认处于关闭状态，强制执行身份验证，并激活闪存更新保护。

备注

无论 SSUMODE 设置如何，闪存写保护和闪存擦除保护都是永久性的，无法逆转。该功能适用于用户需要使闪存代码的某些部分不可更改的用例，例如，为了实现安全算法。在最终确定器件闪存内容之前，请勿尝试配置闪存写入和闪存擦除保护。

7.3 使用 SysConfig 工具

SysConfig 是一种图形用户界面 (GUI) 工具，它提供一种易于使用的方法来配置 TI 微控制器产品，包括 F29x 实时控制 MCU。SysConfig 会自动为器件生成初始化代码，包括外设、中断、引脚多路复用初始化等。SysConfig 还能自动识别器件设置错误，并提供纠正配置问题的有用指导，提供图形可视化，并帮助在不同器件之间轻松移植应用程序。

C29 SysConfig 作为器件 F29 SDK 的一部分提供，需要 SysConfig 工具，该工具与 Code Composer Studio™ (CCS) 集成开发环境 (IDE) 一起内置提供，也可作为独立工具与其他开发环境一起使用。

C29 SysConfig 内的 SSU 工具是一款综合工具，可在 F29x 应用中定义和实现 SSU 保护和隔离。该工具提供了一个易于使用的流程，用于定义应用程序分区、分配代码和数据段、定义内存和外设保护，以及实施调试安全选项。SSU 工具可自动生成实现这些保护所需的所有代码和输出文件，包括 SECCFG 扇区映像、应用程序链接器命令文件、头文件和任何所需的初始化代码。

7.3.1 启用系统安全配置

在应用程序中启用 SSU 功能的第一步是在 SysConfig 中添加 *System Security* 选项，方法是单击左侧栏中模块名称右侧的 (+) 按钮，请参阅图 7-1。

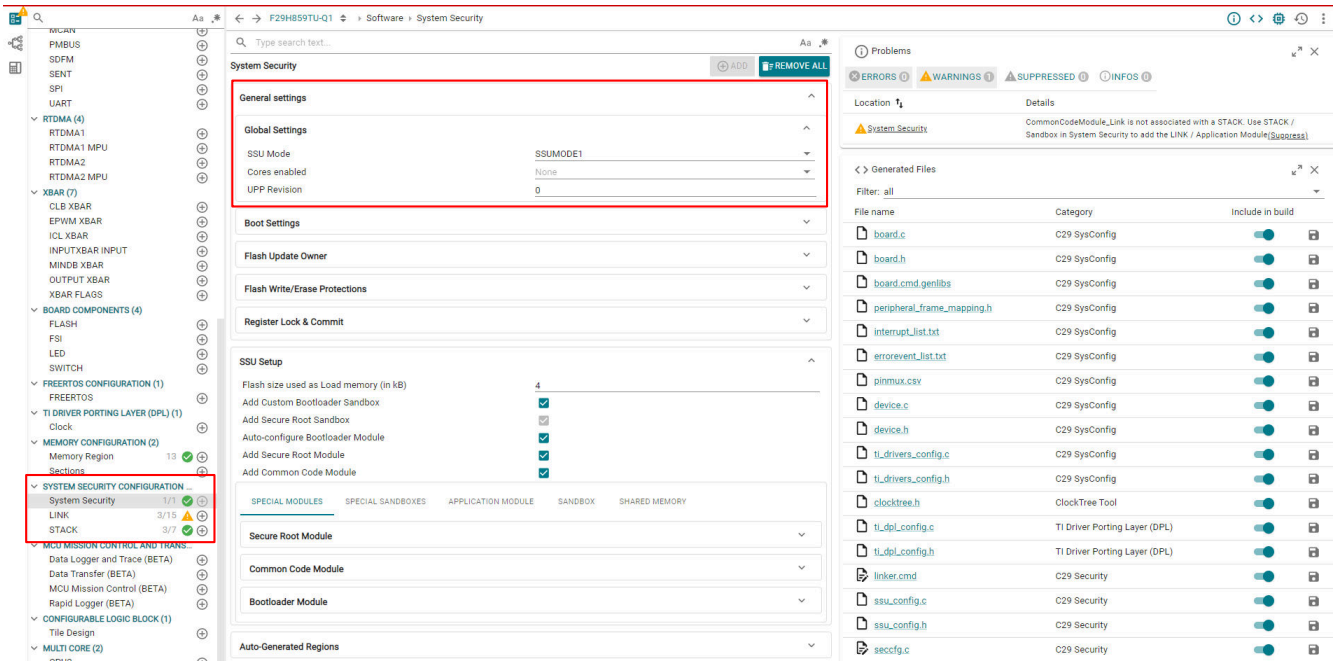

图 7-1. System Security 配置页面

在 *System Security* 模块中，有多个配置选项组。第一个是 *General Settings*，其包含 SSU 操作模式、UPP 版本号和启用的内核的设置。

System Security 模块还包括用于选择器件引导模式、配置闪存更新保护、调试密码（仅在选择 SSUMODE 3 作为 SSU 模式时才可见）和锁定 SSU 寄存器的配置选项。其中一些设置（例如调试密码和闪存更新所有者设置）需要 SSUMODE3 操作才能生效。

在 SysConfig 中添加 *System Security* 模块时，还会自动添加其他模块，例如 LINK、STACK 和内存区域模块。无需修改 LINK 和 STACK 模块。所有配置都可通过 *System Security* 模块完成。LINK 和 STACK 模块仅提供通过 *System Security* 模块选择的 LINK 和 STACK 配置的摘要。添加 *System Security* 模块后，也会自动通过 SysConfig 生成链接器命令文件 (.cmd)。如果使用现有工程，请从工程构建中排除.cmd 文件，防止冲突。

备注

如果需要对链接器命令文件进行额外的修改或添加，请使用 SysConfig 的内存配置部分中的 *Memory Region* 和 *Sections* 模块。

备注

当首次在 SysConfig 中添加 *System Security* 模块时，会出现一条警告，指示 CommonCodeModule_Link 与 *System Security* 模块中的 STACK/SANDBOX 未关联。要解决该警告，请转至 *System Security* 模块的 SSU 设置中的 SANDBOX 选项卡，然后在选择 CommonCodeModule_LINK 作为模块之一的情况下，添加一个沙盒。

7.3.2 配置应用模块

SysConfig 提供了一种基于目标文件、库和输入段创建 AP 范围和配置 LINK 权限的简单方法。创建新的应用模块后，SysConfig 会自动创建一个 LINK 以及一组标准的 AP 区域：

- 从闪存执行的代码区域 (ModuleName_codeAPR_Flash)
- [可选] 从 RAM 执行的代码区域 (ModuleName_codeAPR_RAM)
- RAM 中的变量数据区域 (ModuleName_dataAPR_RW)
- 可选择放置在 RAM 中的只读数据区域 (ModuleName_dataAPR_RO)

除标准区域外，用户还可以通过选择 *Use Custom Sections* 复选框并指定要添加的自定义段，来配置与应用模块相关联的自定义段名称。SysConfig 将所有定义的 AP 区域添加到 SSU 设置，并将关联的 LINK 配置为对每个区域拥有适当的权限。此外，每个 AP 区域都会在链接器命令文件中创建一个输出段，指示链接器按照配置的方式将输入段放入该内存区域。

要将代码函数和数据与应用模块关联起来，只需在 *Files to be included* 输入字段中添加文件名，并去掉文件扩展名即可。也可通过编辑相应的输入字段将库添加到模块中（包含库文件扩展名）。要从库中选择特定对象，可使用链接器命令文件语法，例如 myLibrary.lib<myFuncs1.o>。这就是需要完成的所有工作：SysConfig 会自动将每个对象的 .text、.bss、.data、.rodata 和 .const 输入段分配给链接器命令文件中相应的输出段。

要为模块分配内存，只需指定每种 APR 类型（闪存代码、RAM 代码、RW 数据、RO 数据）所需的内存量即可。SysConfig 会在内存中自动排列 AP 区域，根据最少等待状态的要求选择理想的内存类型。如果应用模块必须从 RAM 而不是闪存执行才能满足性能要求，请选中 *Place .text section in RAM* 复选框。选中该复选框后，SysConfig 将创建一个新的 RAM 代码区域，并配置链接器命令文件，以便在启动时从闪存加载相关代码并从 RAM 运行。如果需要，只读数据或常量数据（如查询表）也可以放在 RAM 中，以实现零等待状态访问。

除了代码和数据内存区域外，通过 SysConfig 配置的现有外设也可以自动分配给每个应用模块。提供了两个下拉选择字段，用于启用对指定外设的读取/写入访问或只读访问。

还可使用“Interrupts Included”字段轻松添加外设中断。该选项可配置 PIPE 模块，为所选外设中断分配正确的执行 LINK。

备注

INT 使用通过 INTSP 寄存器配置的指定 STACK。这是适合所有 INT 的配置。因此，如果使用一个或多个 INT，则必须将 ISR 放置在同一 SSU STACK 中，其与 INTSP 寄存器中选择的 STACK 保持一致。使用 RTINT 时不存在限制。

SysConfig 的内存配置部分中的*内存区域*模块显示了为当前应用程序模块创建的每个 AP 区域的详细信息。该框还提供了一些其他配置选项：

- **仅使用 0 WS 内存：**将 RAM 代码限制为零等待状态 RAM。
- **创建等效的 RTDMA MPU 区域：**创建一个具有相同起始地址和结束地址的 MPU 区，以用于 DMA 传输。
- **与其他内核共享：**启用供多个 CPU 使用的内存区域。

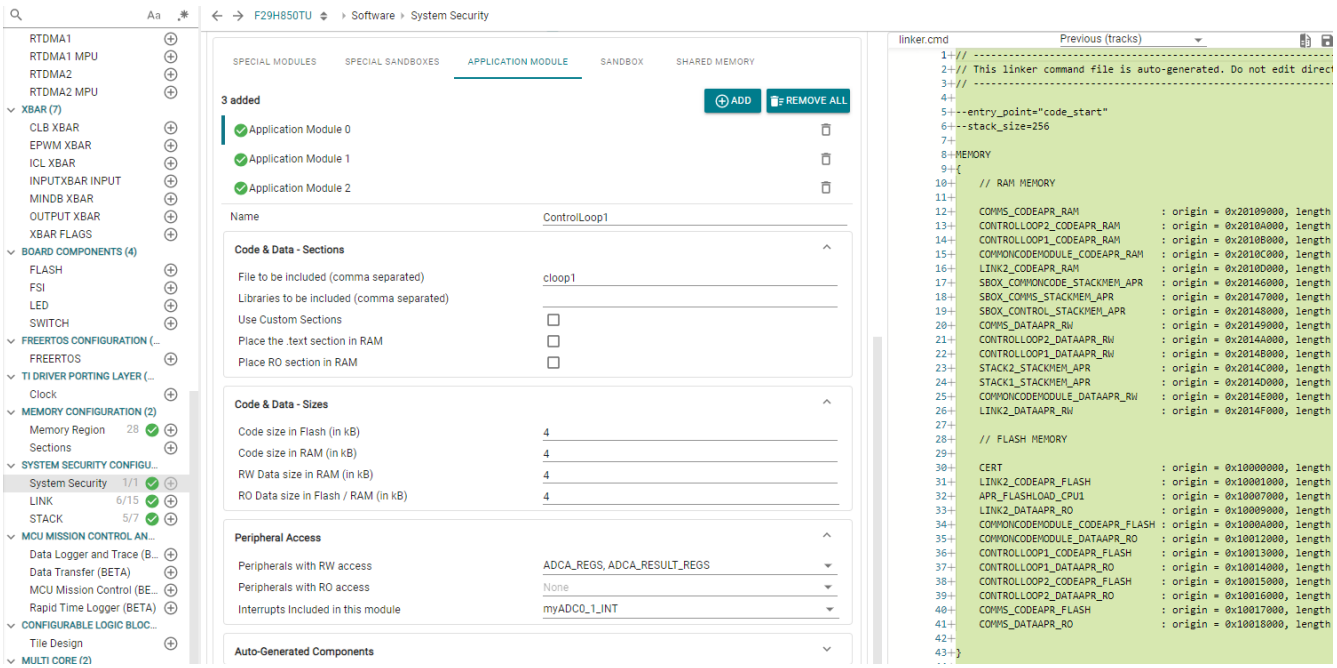

图 7-2. 应用模块配置示例

7.3.3 配置特殊模块

SysConfig 提供了直接配置系统中具有预定义功能的模块的选项：

- LINK2 – 系统安全根 LINK
- LINK1 – 引导加载程序 LINK
- 通用代码 LINK，用于访问保护继承

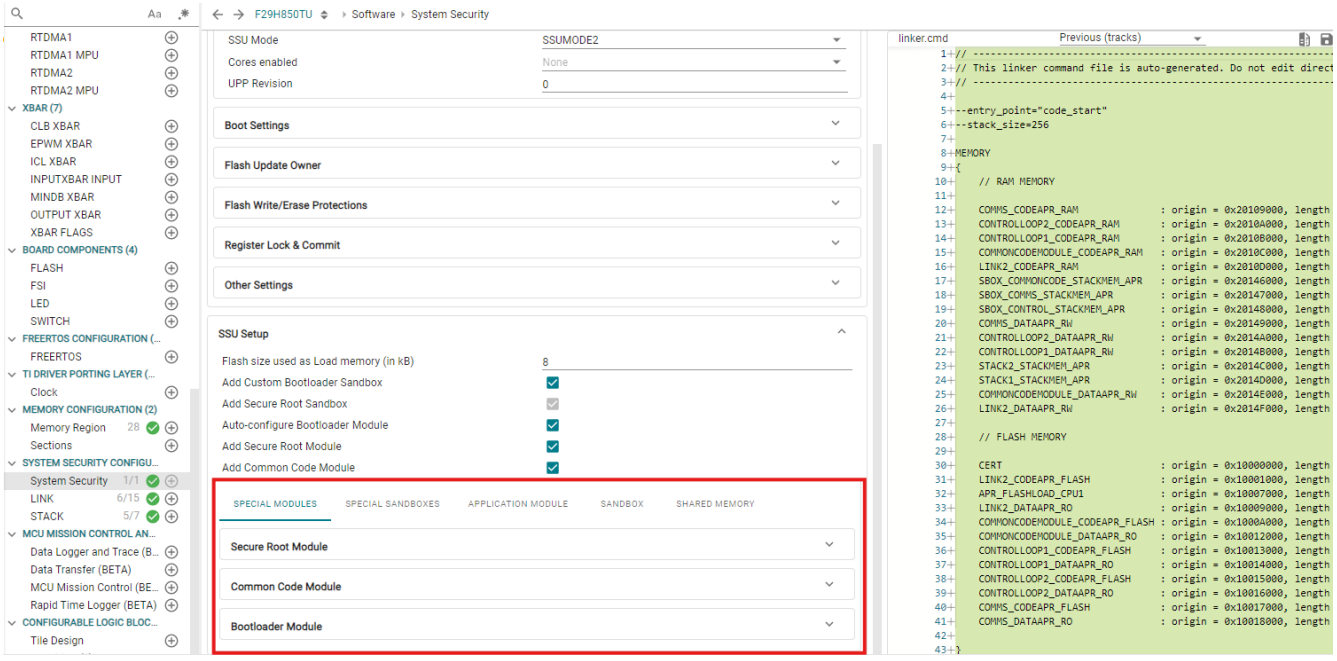

图 7-3. 特殊模块配置示例

启用特殊模块后，可通过展开下拉框来配置每个模块。以下各节将更详细地解释这些特殊模块。

7.3.3.1 LINK2 配置

LINK2 是每个 CPU 上最安全的 LINK。LINK2 具有提升权限，包括访问安全 CPU 寄存器和执行监督任务的能力。大多数情况下，RTOS 层函数置于 LINK2 中。特别是 CPU1.LINK2 具有器件范围的特殊提升权限，包括：

- 能够写入器件配置寄存器
- 能够写入某些 SSU 寄存器
- 能够配置 RTDMA，包括 MPU 配置

将负责执行初始系统配置和电路板配置以及运行操作系统功能的所有代码和数据部分放入 LINK2。表 7-1 介绍了 LINK2 必需的外设寄存器，以及必须有这些寄存器的原因。

表 7-1. LINK2 的强制性外设访问保护

外设寄存器基址	Reason (原因)
CPU_SYS_REGS	CPU 系统寄存器
PER_CFG_REGS_WD_REGS	设置看门狗寄存器
CPUTIMER2_REGS	驱动程序移植层 (DPL) 的 CPU 计时器 2 寄存器
DCC1_REGS	时钟监控器
ERR_AGG_REGS	错误/NMI 处理
ESM_CPU1_REGS	错误/NMI 处理
ESM_CPU2_REGS	错误/NMI 处理
ESM_CPU3_REGS	错误/NMI 处理
ESM_SYS_REGS	错误/NMI 处理
ESM_SAFETY_AGG_REGS	错误/NMI 处理

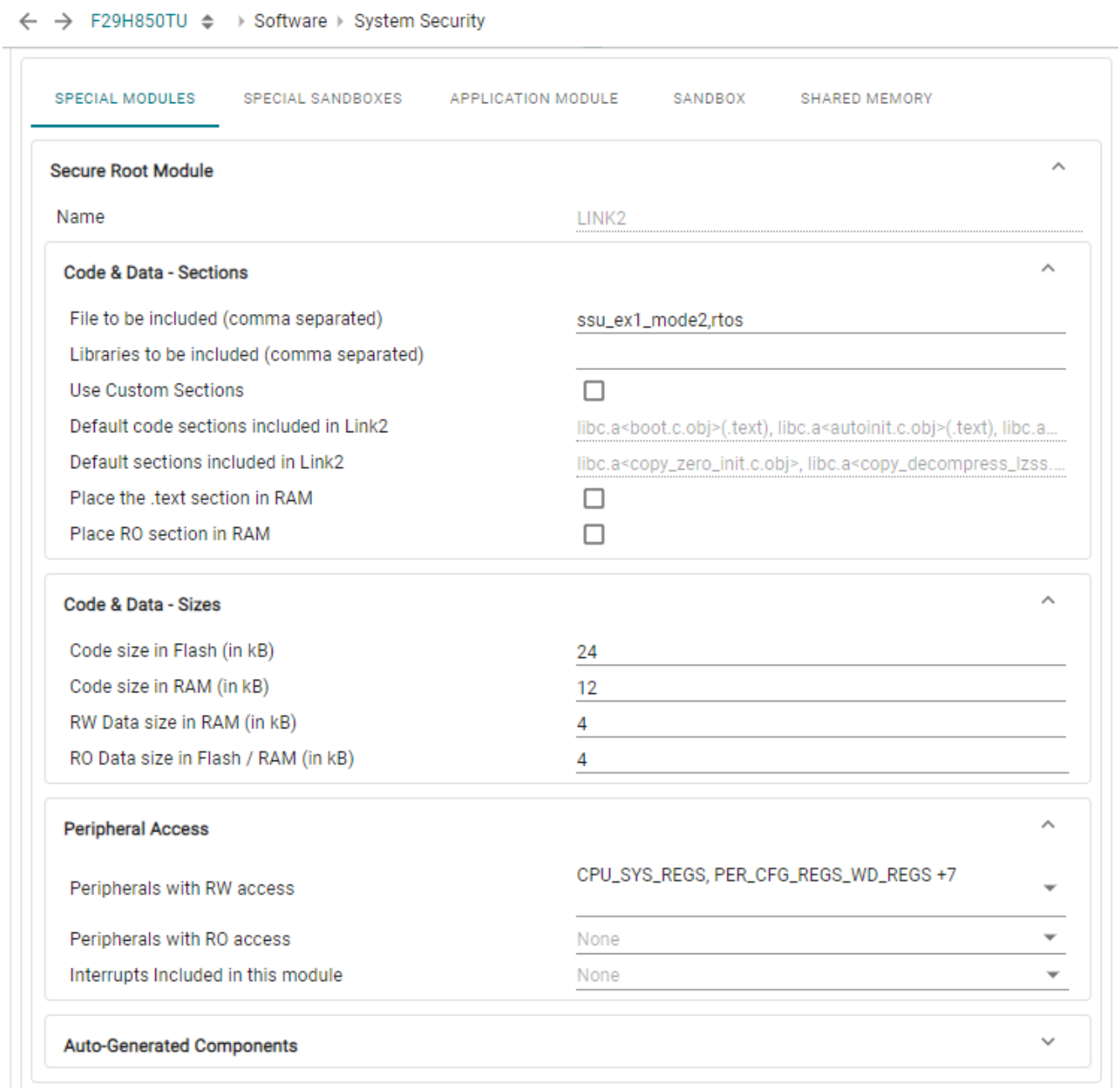

图 7-4. LINK2 配置示例

7.3.3.2 LINK1 配置

CPU1.LINK1 主要用于引导加载程序，具有支持引导加载程序功能的额外硬编码权限，如写入某些系统配置寄存器的能力。LINK1 也可用作传统的用户 LINK；但不建议向 LINK1 添加与引导加载程序无关的代码和数据，除非是所有其他 LINK 都已被使用，不得已而为之。

在 *Boot Settings* 组中配置任何外设引导模式时，SysConfig 都会自动配置 LINK1，使其能够访问该引导模式运行所需的相应外设，例如 CAN、UART 或 SPI。除了器件启动始终需要的某些外设（如 IPC 和 HSM 邮箱）外，SysConfig 还会自动将这些外设添加到 LINK1 模块中。表 7-2 介绍了器件启动过程中 LINK1 所需强制外设的完整列表。

表 7-2. LINK1 的强制性外设访问保护

外设寄存器基址	Reason（原因）
PER_CFG_REGS_WD_REGS	看门狗寄存器设置

表 7-2. LINK1 的强制性外设访问保护（续）

外设寄存器基址	Reason（原因）
IPC_CPU1_SEND_REGS_HSM_CH0	器件身份验证期间与 HSM 握手
IPC_CPU1_SEND_REGS_HSM_CH1	器件身份验证期间与 HSM 握手
HSM_MAILBOX	IPC 的 HSM 邮箱
GPIO_DATA_REGS	引导模式选择及错误状态引脚配置
MCANA_MSGRAM	外设引导模式
MCANA_REGS	外设引导模式
UARTA_REGS	外设引导模式
DCC1_REGS	时钟监控器
ERR_AGG_REGS	错误/NMI 处理
ESM_CPU1_REGS	错误/NMI 处理

7.3.3.3 通用代码链接配置

7.3.4 定义沙盒

使用 SysConfig 中的沙盒定义必须与应用程序其他部分信息安全隔离的应用模块组。每个沙盒都与一个 SSU STACK 关联，并且至少包含一个应用模块以及一个堆栈内存 AP 范围。与沙盒中的应用模块关联的所有 LINK 都具有对沙盒堆栈内存的读取/写入访问权限；所有其他 LINK 都没有访问权限。每个沙盒都与一个调试 ZONE 关联。

SysConfig 在链接器命令文件中为每个沙盒定义了一个 SECURE_GROUP。此设置使链接器要求从其他 STACK 进入沙盒 STACK 的所有函数调用都必须受到保护。默认情况下，任何进入 SECURE_GROUP 的未受保护调用都会导致链接器生成错误。SysConfig 提供了自动生成 trampoline 和登录调用的选项，以满足受保护调用要求。启用该选项时，请务必检查输出链接器映射文件，以确认没有生成到不可信代码的不良跨堆栈 trampoline。

备注

由于需要将 CPU 寄存器保存到堆栈内存或从堆栈内存恢复 CPU 寄存器，跨堆栈 trampoline 可能会增加延迟，从而可能影响应用程序性能。为了获得出色性能，可通过向函数定义中添加 __attribute__((c29_protected_call))，直接在应用程序代码中实现受保护的函数调用。

备注

可在 *Special Modules* 选项卡下访问 STACK1 配置。

7.3.5 添加共享内存

共享内存区域是可由多个应用模块访问的特殊访问保护区域 (APR)。在 SysConfig 中，可通过在 *System Security* 页面上选择 *Shared Memory* 选项卡并点击 *Add* 按钮来添加共享内存。可以添加多个共享内存，但受 CPU 上可用 APR 总数（包括为各种应用模块定义的 APR）的限制。每个共享内存都可以包含源文件；SysConfig 会按照配置的方式从这些文件中添加 .bss、.data、.const 和 .rodata 段。也可以定义要包括的定制段名。

对于每个共享内存，您都可以选择哪些应用模块需要只读权限、哪些模块需要写权限。SysConfig 会自动配置每个模块的 LINK 所定义的 APR 权限。

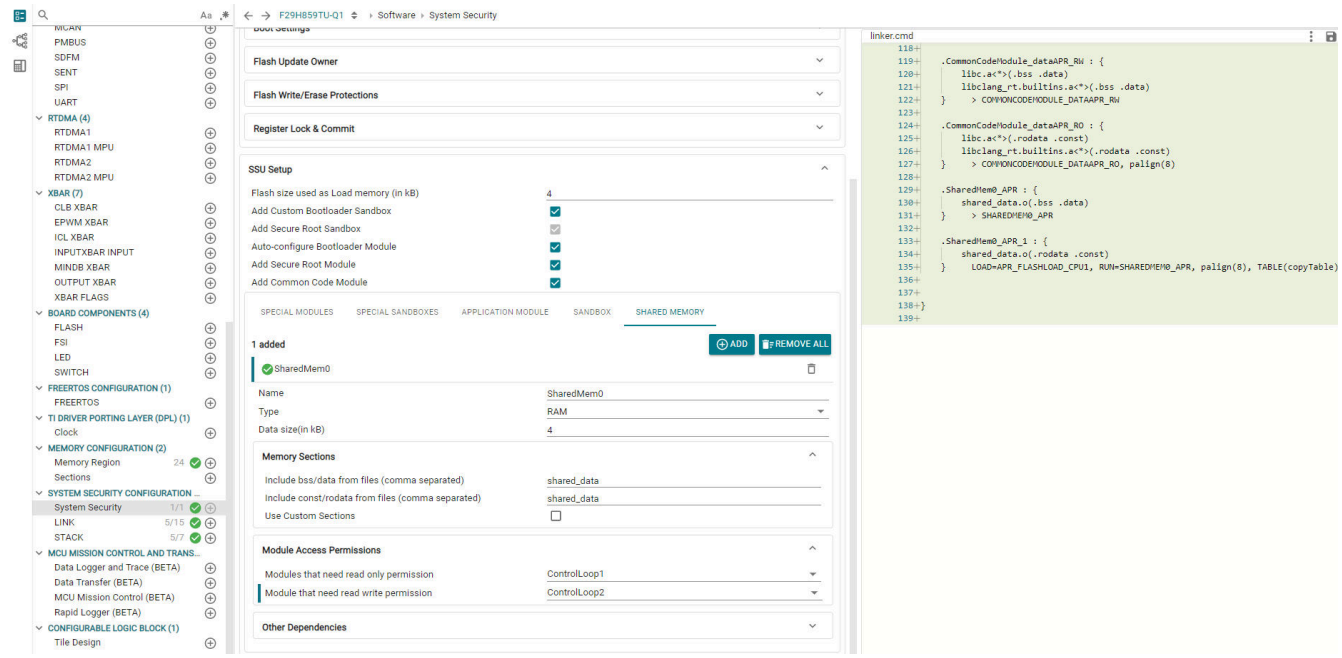

图 7-5. 共享内存配置示例

8 调试授权

8.1 基于密码的解锁

在 *System Security* 模块内，可以选择 SSUMODE。如图 8-1 中所示，支持 SSUMODE 3 安全调试和固件更新。因此，若要根据密码条目和匹配项启用调试授权，首先选择 SSUMODE3。选择 SSUMODE3 后，将出现一个名为“调试访问（仅 SSU 模式 3）”的部分。在该部分中，可以为每个 ZONE 定义部分调试和完全调试的密码。C29 调试只有一个密码，因此只有一个条目。这些密码将在启动时加载至 SECCFG。

备注

使用 SSUMODE3 时，请确认已设置正确的访问权限。建议先在 SSUMODE2 中设置用户测试权限。如果器件在没有正确的 APR 配置的情况下转换到 SSUMODE3，则由于强制执行保护，代码将无法正确执行。

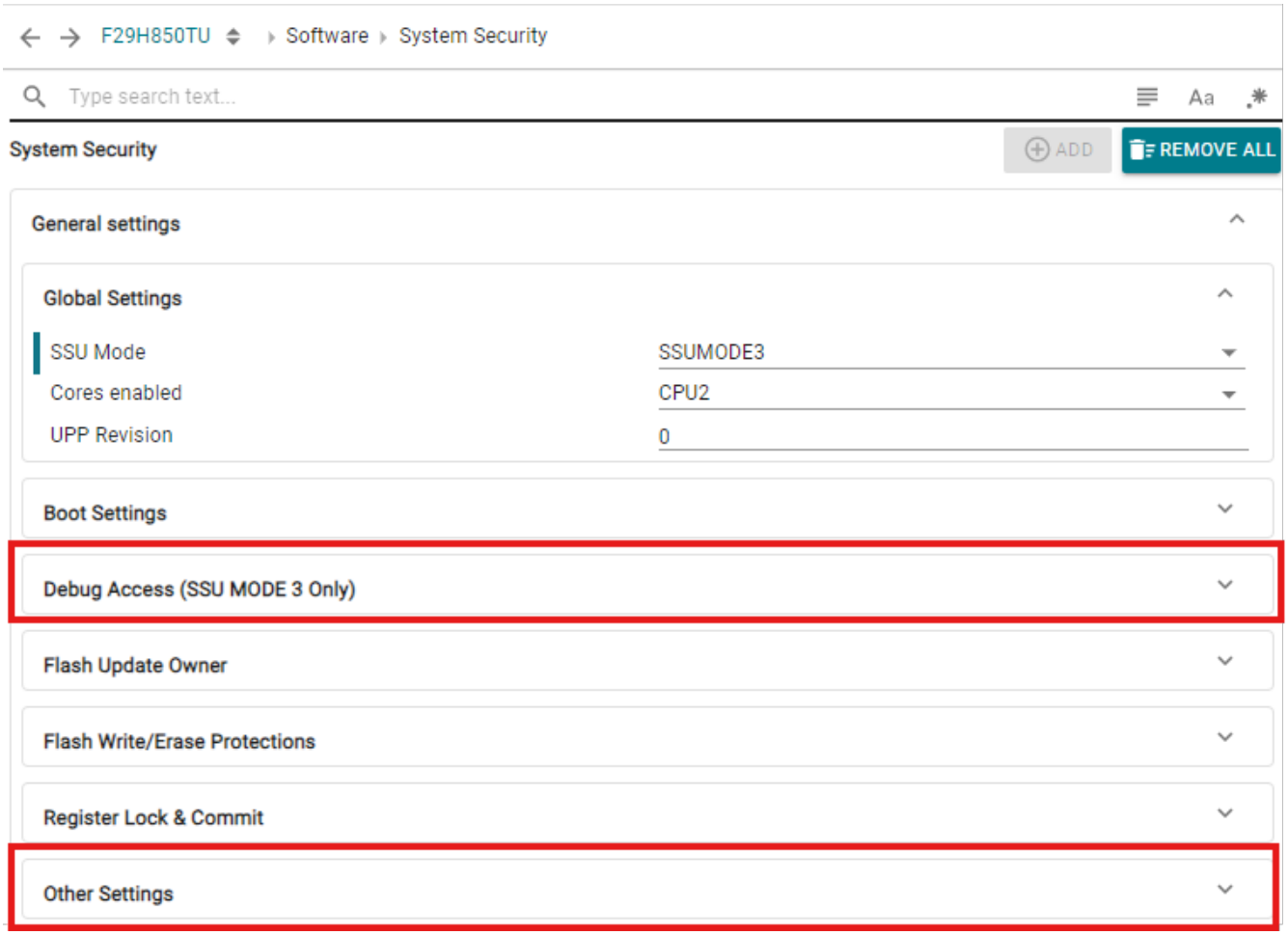

图 8-1. SSU 模式 3 附加设置

一旦器件处于 SSUMODE3 状态，所有调试访问都会关闭。需要通过 SEC-AP 接口将 SysConfig 中的密码设置扫描到器件中，以针对不同的区域进行适当调试。为此，请使用 SysConfig 中使用的密码修改 SEC-AP GEL 文件。要在 CCS 中访问 SEC-AP GEL 文件，请执行以下步骤：

- 右键单击连接窗格中 CCS Debug 视图中的任何可见内核，然后选择“Show all cores”
- 连接到 C29 SEC-AP 内核
- 转至 CCS 工具栏菜单中的“View”，然后选择“GEL Files”
- 打开提供的 SEC-AP gel 文件

- 修改 SysConfig 中提供的不同区域的密码。128 位密码的写入顺序如下所示：[31:0] 位、[63:32] 位、[95:64] 位、[127:96] 位。
- 断开与 SECAP 接口的连接，然后重新连接。会在连接时扫描到密码。因此，如果通过 SECAP 接口扫描的密码与通过 SysConfig 配置的密码匹配，则 C29 和区域会打开。

成功打开调试后，可以连接到相应的 C29 内核。

9 调试 SSU

9.1 调试构建错误

表 9-1. 可能的构建错误

错误示例	说明	分辨率
“syscfg/linker.cmd”，第 73 行：错误 #10099-D：程序将无法放入可用的内存中，或者该段包含一个需要蹦床函数的调用点，而该蹦床函数无法为该段生成，或该段包含填充的函数。对于第 0 页大小为 0x5052 的“.LINK2_codeAPR_Flash”部分，对齐放置失败。可用内存范围： LINK2_CODEAPR_FLASH 大小：0x2000 未使用：0x1fe0 最大孔：0x1fe0	定义的 APR 大小不足。	如果为任何 APR 类型（闪存、RAM、RW 或 RO 数据）分配的内存不足，则在工程构建时将出现构建错误。要确定 APR 的正确大小，请将提供的大小 (0x5052) 转换为十进制数 (20,562)。这表示 APR 的总大小约为 20k。APR 绑定到 4kB，因此对于此分区情况，APR 大小需要设置为 24kB。
错误#10483-D：不允许从输出段“.comms_Module_codeAPR_Flash”到未受保护符号“UART_writeCharArray”进行未受保护的调用：SECURE_GROUP 不匹配；调用位于 SECURE_GROUP "STACK2_STACK”，被调用者位于 SECURE_GROUP "sbox_CommonCode_STACK_COMMONCODE"	跨堆栈调用未附带相应的进入和退出指令。	跨 STACK 调用必须一直附带适当的进入和退出指令。SSU 工具有一个选项可以指定如何处理未受保护调用。共有三个选项： • 生成链接器错误：如果没有适当的进入和退出说明（由用户设置），则会引发错误。 • 添加链接器蹦床函数和焊盘（安全）：编译器为针对所选 STACK 的所有调用插入适当的进入和退出指令。寄存器会在跨 STACK 调用中保留。 • 添加链接器蹦床函数和焊盘（不安全，速度更快）：编译器为针对所选 STACK 的所有调用插入适当的进入和退出指令。寄存器不会在跨 STACK 调用中保留。 选择适当选项来处理跨 STACK 进入和退出指令。

9.2 调试运行时错误

通过 F29x 的错误聚合器模块 (EAM) 和错误信令模块 (ESM) 捕获和记录来自 SSU 及其他外设的各种错误。有关如何查看 CCS 中 EAM 和 ESM 错误的更多信息，请参阅 F29x 错误处理和调试指南应用报告。

表 9-2. 可能的 SSU 运行时错误

错误示例	说明	分辨率
C29xx_CPU1：闪存编程期间出错。地址 0x10D85000，FMSTAT（某些器件上为 STATCMD）0x00000000，值 0x00000101 C29xx_CPU1：文件管理器加载程序：内存写入失败：未知错误 C29xx_CPU1：GEL：File: C:\Users…..ssu_ex1_mode2.out：加载失败。	当未正确设置闪存设置来对 SECCFG 进行编程时，会看到以下错误。	要允许擦除 SECCFG 内存，请验证 Flash Settings 的 *NonMain Erase Settings* 部分中的“Allow NonMain Flash erase before loading data to Flash memory”复选框选中状况。之后，重新刷新程序 **备注** 验证在未用于 SECCFG 编程时**未选择**此设置

表 9-2. 可能的 SSU 运行时错误（续）

错误示例	说明	分辨率
CPU1_DW 错误（HP 错误地址 = 0x60070018、LP 错误地址 = 0x00000000、PC = 0x10010646） 安全违规	以下错误表示，位于程序计数器 (PC) 地址 0x10010646 的指令尝试将数据写入地址 0x60070018，但由于没有足够的权限而无法写入。	要调试该问题，请在 CCS Disassembly 视图 (*View -> Disassembly*) 中查找 PC 地址。这示出了程序计数器的位置。这可以深入了解禁止写入访问的代码执行（即哪个函数以及该函数的哪一行）。在本例中，程序计数器位于 LINK4 中的“update_PSFB()”函数中。 接下来，在 CCS (View -> Memory) 中打开存储器浏览器并输入地址 0x60070018。这提供了指令将要写入的内存内容。在这种情况下，0x60070018 对应于 UARTA 寄存器。F29x DS 中的内存映射也可用于将提供的错误地址与设置的寄存器相关联。在这种情况下，用户可以在内存映射中查找 0x6007_0000。 因此，LINK4 需要被赋予对 UARTA 外围设备的 R/W 访问权限，以便“update_PSFB()”函数中的代码写入 UARTA 寄存器。 **备注** 有多种 CPU1 错误，例如 PR、DR1、DR2 等。有关这些单个错误类型的含义的更多详细信息，请参阅 TRM 或 F29x 错误处理和调试指南应用报告中的错误聚合器一章。
SSU 错误（HP 错误地址 = 0x3008000C、LP 错误地址 = 0x00000000、PC = 0x00000000） CPU1_SSU_MMR_ACCESS_ERROR	该错误表明 SSU 不被允许访问 SSU 内存映射寄存器 (MMR) 之一。	要调试该问题，请首先在 CCS 内存浏览器中查找提供的地址 (View -> Memory)。地址 0x3008000C 对应于 LINK2_AP_OVERRIDE。CCS 中的“搜索”功能用于定位项目中调用“SSU_enableLink2APOverride()”的所有实例。在本例中，在 LINK2 之外使用了“SSU_enableLink2APOverride()”函数，导致了此错误。遇到其他 SSU MMR 访问错误时，请使用类似的方法。
在刷写期间-“警告：目标 CPU 可能滞留在持续故障状态”	器件处于 SSUMODE2 状态并实现内存保护。	如果 SSU 不适合在启用功能安全和信息安全保护的情况下使用，则需要将 SSU 模式更改为 SSU 模式 1。为此，对 SECCFG 重新编程以将器件置于 SSUMODE1，并在 BANKMGMT 中对 BANKMODE0 重新编程。 • 将默认的 seccfg 程序加载至找到的器件。默认的 seccfg 程序位于以下路径： C:\ti\<f29h85x-sdk_version_x>\source\defseccfgbin • 在 EVM 或 HW 上提供复位 • 连接到目标并且验证 SSUGENREGS 寄存器中的 SSUMODE = 0x30。 • 在 Flash Settings 中，将 FLASH BANKMODE 编程为 BANKMODE 0，或验证是否设置了所需的 BANKMODE。 • 如果组模式发生更改，请断开与 CCS 中目标的连接，并在 EVM（controlSOM 具有 XRSN 按钮）上发出 XRSN。 • 连接到目标并验证 SSUGENREGS 中的 BANKMODE = 0x3 或正确设置了所需的 BANKMODE。

10 SSU 常见问题解答 (FAQ)

表 10-1. SSU 问题及答案

#	问题	答案
1	如果在运行期间修改 APR，是否有必要在更改 APR 之前禁用 APR？	否，可以在启用它时更改它。
2	LINK2 AP 覆盖功能有什么作用？	LINK2 获取对已启用或配置的所有 APR 的读写 (R/W) 访问，即使未向 LINK2 提供这些权限也是如此。
3	在修改代码和数据 APR 的范围时，LINKID 是否为必填字段？	否，仅当设置了 XE 位时才需要 LINKID（代码 APR）。
4	是否可在运行时动态更改 SSU APR 寄存器中的起始地址、结束地址和 LINKID 字段？	是，它可以在运行时进行更新。只有 LINK2 能访问并修改这些寄存器。在运行时无法修改 LINKx_CFG 和 STACKy_CFG。
5	如果包含一个文件（作为模块配置的一部分），则整个文件是否通过 APR 捕获？	是，文件中的代码和变量通过 SysConfig 中的 SSU 工具设置为该 LINK 的 APR 的一部分。

11 总结

C29x SSU 可为实时控制应用提供先进的功能安全和信息安全功能，包括新颖的上下文关联内存和外设保护功能，可在切换任务或中断服务时消除软件开销，同时还包括先进的调试安全选项。利用简单易用的 SysConfig 工具，系统设计人员可以将应用程序划分为多个分区，以实现功能安全和信息安全隔离，并自动为每个分区分配目标文件、库和外设。还可以定义共享内存资源，供多个应用模块使用。SysConfig 可自动处理整个器件上各种应用分区的内存分配，并完全支持多核配置。该工具可生成实施所需保护策略所需的所有输出文件，然后将其构建到应用程序 .out 映像中。

12 参考资料

1. 德州仪器 (TI)，*C2000™ SysConfig 应用手册*
2. 德州仪器 (TI)，*TI Resource Explorer：C2000™ 实时微控制器*
3. 德州仪器 (TI)，Code Composer Studio (CCS) IDE
   - 集成开发环境 (IDE)，支持 TI 的微控制器和嵌入式处理器产品
   - SysConfig 工具以集成方式在 CCS 中提供（内置 SysConfig 支持）
4. 德州仪器 (TI)，SysConfig 独立版本
   - SysConfig 独立版本可与没有内置 SysConfig 工具的其他 IDE 一起使用

13 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from NOVEMBER 30, 2024 to OCTOBER 31, 2025 (from Revision * (November 2024) to Revision A (October 2025)) **Page**

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