瑞萨 RA 产品家族

Arm TrustZone 的安全设计 - IP 保护

简介

适用于 ARMv8-M 的 Arm® TrustZone® 技术是一项可选的安全扩展，旨在为提高各种嵌入式应用中的系统级安全性奠定基础。本应用笔记解释了各种支持 RA MCU TrustZone 技术的硬件和软件功能，并提供了这些功能的使用指南。此外，本应用项目将引导用户逐步了解如何启动使用瑞萨 RA MCU 产品家族支持 TrustZone 技术的安全系统设计。

对于 Arm TrustZone 技术的基础知识，请阅读来自 Arm 的文档*《适用于 Armv8-M 架构的 Arm® TrustZone 技术》*。这篇文档的链接位于参考资料部分。本应用项目重点介绍 RA MCU 产品家族 TrustZone 技术的实现及特性，仅对 Arm TrustZone 技术进行简要介绍。

创建安全设计涉及硬件实施、软件开发（安全目的）以及工具支持。对于基于 TrustZone 的安全设计，工具在产品的开发、生产和部署中起着至关重要的作用。在参考本应用项目开始第一个支持 RA MCU TrustZone 技术的项目设计之前，建议用户阅读*《FSP 用户手册》的“入门：TrustZone 项目开发”部分*。这篇文档的链接位于参考资料部分。本文将在必要时引用该文档。

文中提供了一个基于 EK-RA6M4 的应用项目作为参考项目，供用户启动支持 RA MCU 产品家族 TrustZone 功能的应用，此项目实现了 TrustZone 技术的 IP 保护用例。

所需资源

软件和开发工具

- e² studio IDE v2021-04 或更高版本
- 瑞萨灵活配置软件包 (FSP) v3.0.0 或更高版本
- SEGGER J-Link® USB 驱动程序

  注：FSP、J-Link USB 驱动程序和 e² studio 这三个软件组件以捆绑包的形式通过一个可下载的平台安装程序提供，可从 FSP 网页上（网址为 renesas.com/ra/fsp）上下载
- 使用 www.renesas.com/us/en/products/software-tools/tools/programmer/flash-development-toolkit-programming-gui.html#downloads 下载并安装瑞萨闪存编程器 (RFP) V3.08 或更高版本

硬件

- EK-RA6M4，用于 RA6M4 MCU 系列的评估板 (renesas.com/ra/ek-ra6m4)
- 运行 Windows® 10 的工作站；Tera Term 控制台或类似的应用程序
- 一根 USB 线缆（Type-A 公头转 micro-B 公头）

前提条件和目标受众

本应用项目假设您在使用瑞萨 e² studio IDE 和 FSP 方面有一定的经验。在继续本应用项目中提供的动手练习之前，请阅读简介部分中提到的两个参考文档。此外，我们建议阅读应用笔记*《瑞萨 RA 产品家族 - 器件生命周期管理密钥安装》*的前两章了解支持 TrustZone 技术的 RA MCU 的器件生命周期状态。此外，用户需要知道如何使用 EK-RA6M4 进入 MCU 引导模式，并创建一个基本的 RFP 项目来与 MCU 进行通信。本应用项目仅提供所使用的特定功能的必要设置。有关 MCU 引导模式和 RFP 的更多信息，请参见*《瑞萨 RA6M4 系列用户手册：硬件》*和*《瑞萨闪存编程器用户手册》*。

目标受众是所有开始使用瑞萨 RA MCU 产品家族开发基于 Arm® TrustZone® 的应用的用户。

目录

1. Arm® TrustZone® 及其安全功能简介

1.1 TrustZone 技术概述

Arm® TrustZone® 技术是针对 MCU 功能的一种硬件强制隔离技术。Arm® TrustZone® 技术使系统和软件能够划分为安全区域和非安全区域。安全软件可以访问安全和非安全存储器和资源，而非安全软件只能访问非安全存储器和资源。这些安全状态与现有的线程和处理程序模式正交，可在安全和非安全状态下使能线程和处理程序模式。

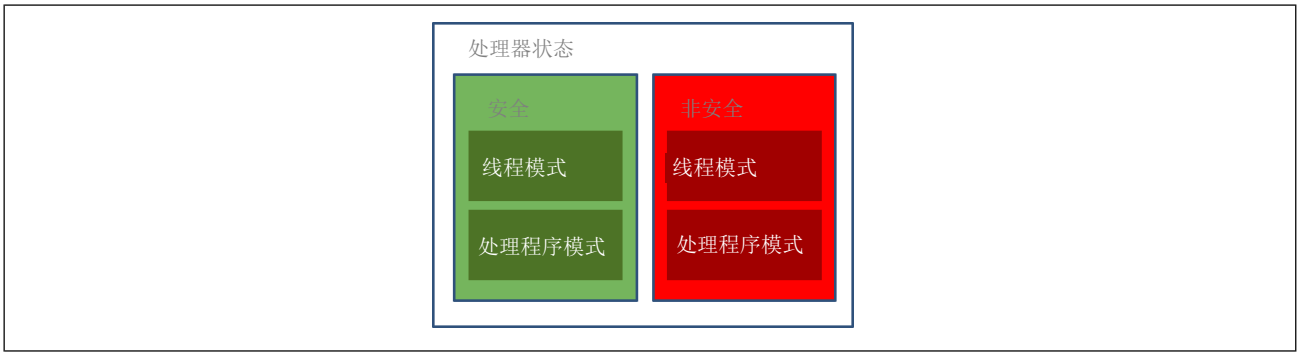

图 1. 处理器状态

带有安全扩展的 Armv8-M 架构是一个可选的架构扩展。如果实现了安全扩展，则系统默认以安全状态启动。如果未实现安全扩展，则系统始终处于非安全状态。Arm TrustZone 技术并未涵盖安全的所有方面。例如，它不包括加密。

因此，在 Armv8-M 架构具有安全扩展的设计中，对系统安全至关重要的组件可置于安全区域中。

这些关键组件包括：

- 安全自举程序
- 密钥
- 闪存编程支持
- 高价值资产

其余的应用程序置于非安全区域中。

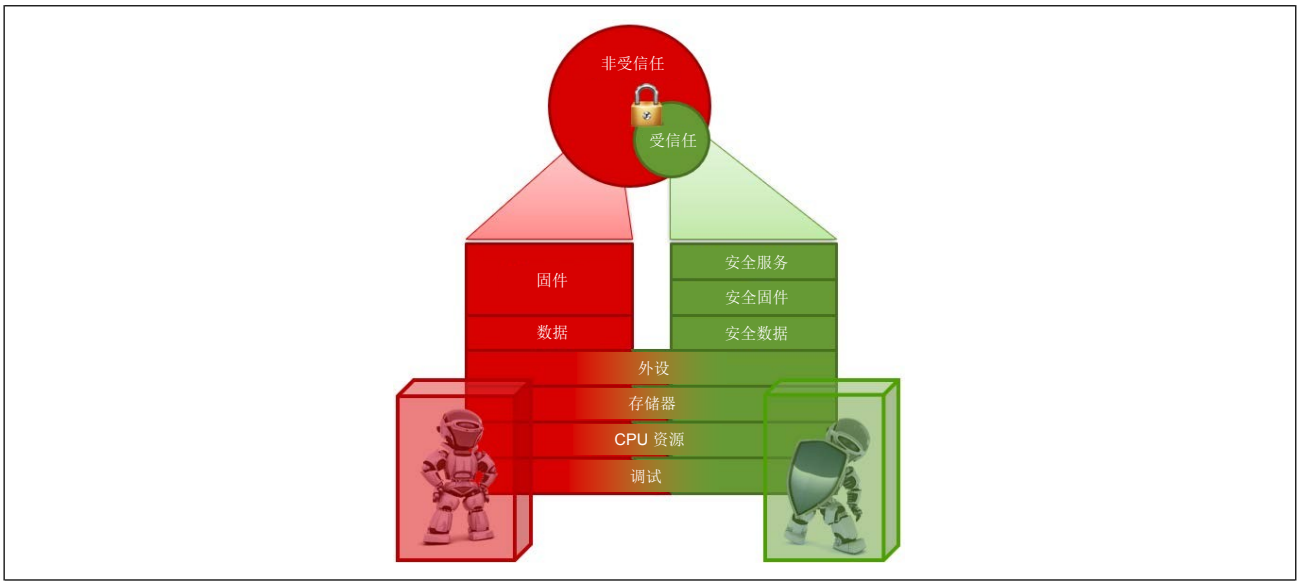

图 2. 安全和非安全区域

如简介部分中所述，有关 TrustZone 的定义和使用的更多详细信息，请参见 Arm 文档*《适用于 Armv8-M 架构的 Arm® TrustZone 技术》*。

1.2 采用 Arm® TrustZone® 的 RA MCU 硬件强制安全

要构建安全硬件平台，安全注意事项需要超越处理器级别。支持 Arm® TrustZone® 的瑞萨 RA MCU 将安全措施扩展到整个系统，包括：

- 存储器系统
- 总线系统
- 对安全和非安全外设的访问控制
- 调试系统

请注意，本节中使用 RA6M4 MCU 组作为参考。其他支持 TrustZone 技术的 MCU 在硬件功能的细节方面可能会有一些变化。

1.2.1 存储器分离

支持 RA TrustZone 技术的 RA MCU 上的代码闪存、数据闪存和 SRAM 通过 IDAU（实施定义属性单元）分为安全 (S)、非安全 (NS) 和非安全可调用 (NSC) 区域。当器件生命周期为安全软件开发 (SSD) 状态时，可通过使用串行编程命令将这些存储器安全属性编程到非易失性存储器中。有关器件生命周期状态定义和切换的信息，请参见*《RA6M4 用户手册：硬件》*的“安全功能”部分。

下图汇总了 8 个可用区域。

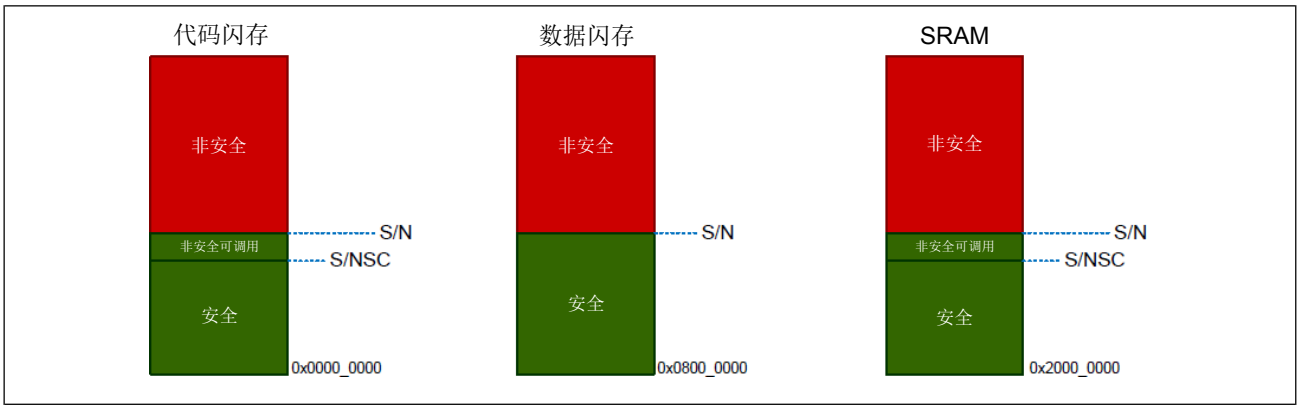

图 3. IDAU 区域

基于 TrustZone 的代码和数据闪存安全功能

从非安全区域读取的代码和数据闪存区域将生成 TrustZone 安全故障。根据不同的产品设计，代码和数据闪存编程和擦除 (P/E) 模式入口可以配置为仅通过安全软件实现，也可以配置为通过安全和非安全软件实现。

默认情况下，MCU 将代码和数据闪存 P/E 功能配置为仅通过安全软件实现。闪存驱动程序可以置于安全分区中，并且可以通过 FSP 配置为非安全可调用，以允许非安全应用程序执行闪存 P/E 操作。

表 1. 安全闪存区域读/写保护

访问违例	错误报告
闪存读取	TrustZone 安全故障：复位或 NMI
闪存 P/E 模式进入	闪存 P/E 错误标志：由 FSP 闪存驱动程序处理

RA MCU 产品家族支持安全区域和非安全区域的临时和永久闪存块保护。有关支持代码和数据闪存 TrustZone 技术的硬件功能的更多详细信息，请参见*《瑞萨 RA6M4 系列用户手册：硬件》*的“闪存”部分。

SRAM

包括 ECC 区域和奇偶校验的 SRAM 存储器（例如 SRAM0）可以通过存储器安全属性 (MSA) 分为安全/非安全可调用/非安全状态，并且可以防止非安全访问。当 MSA 指示 SRAM 存储区域处于安全或非安全可调用状态时，非安全访问无法覆写它们。

表 2. 安全 SRAM 区域读/写保护

访问违例	错误报告
SRAM 读取	Arm® TrustZone® 安全故障：复位或 NMI
SRAM 写入	Arm® TrustZone® 安全故障：复位或 NMI

1.2.2 总线系统分离

CPU、DMAC 和 DTC 的 IDAU 区域设置是一致的。DMAC 和 DTC 也使用主 TrustZone 过滤器。

1.2.2.1 DMA 控制器和数据传输控制器的主 TrustZone 过滤器

直接存储器访问控制器 (DMAC) 和数据传输控制器 (DTC) 由主 TrustZone 过滤器监控。在访问总线之前，会提前检测闪存和 SRAM 的 TrustZone 违例区域。DMAC 和 DTC 中的主 TrustZone 过滤器可以检测 IDAU 定义的闪存区域（代码闪存和数据闪存）和 SRAM 区域（ECC/奇偶校验 RAM）的安全区域。当非安全通道访问这些地址时，主 TrustZone 过滤器会检测到安全违例，不允许访问违例地址。对于 DMA 和 DTC，检测到的访问违例将作为“主 TrustZone 过滤器错误”处理，产生DMA_TRANSERR 中断以响应“主 TrustZone 过滤器错误”。

以下是有关 DMAC 安全属性的一些附加注释：

- 可以为每个通道单独配置安全属性。每个 DMA 通道可以设为安全或非安全属性。
- 只有安全代码可以配置 DMAC的启动方式，可以由安全代码启动，还是由非安全代码启动。
  - 如果在安全项目中使用 DMAC，FSP 将以安全模式启动 DMA，并通过设置相应的寄存器来防止非安全项目意外停止 DMAC。

1.2.2.2 以太网 DMA 控制器

RA6M4 要求将 EDMAC RAM 缓冲区置于 TrustZone 的非安全 RAM 中。EDMAC 被硬编码为 TrustZone 非安全总线主器件。这些硬件特性允许以下两个以太网程序分区选项：

- 在安全区域中运行以太网程序 ，在非安全区域中放置 EDMAC RAM 缓冲区
- 在非安全区域中运行以太网程序 EDMAC RAM 缓冲区

FSP 支持客户使用这两种选项实现。

1.2.2.3 总线主 MPU TrustZone 功能

总线主 MPU 可用于除 CPU 外的每个总线主器件的存储器保护功能。安全软件可以设置总线主 MPU 的安全属性。

有关总线系统安全属性控制的更多详细信息，请参见*《瑞萨 RA6M4 用户手册：硬件》*和*《FSP 用户手册》*。

1.2.3 IO 与外设分离

MCU 中的大多数外设都可以配置为安全或非安全，但有几个例外，如表 3 所示。

外设分为两种类型。

- 类型 1 外设具有一种安全属性，对所有寄存器的访问由一种安全属性控制。安全应用程序将类型 1 外设安全属性设置到外设安全属性寄存器（PSARx：x = B 至 E）。
  - e² studio 和 FSP 提供了简便的方式来分配 PSARx。
  - 外设的不同通道可以采用不同的安全属性。例如，UART 通道 0 和通道 1 可以具有不同的安全或非安全属性。
- 类型 2 外设具有每个寄存器或每个位的安全属性，根据这些安全属性控制对每个寄存器或位域的访问。安全应用程序将类型 2 外设安全属性设置到每个模块中的安全属性寄存器。对于安全属性寄存器，请参见每个外设的用户手册中的相关部分。
  - e² studio 和 FSP 可配置其中大多数外设，有几个例外的外设已采用合理的默认设置以提供更好的开发体验。
  - 有关每个外设的详细信息，请参见最新的*《FSP 用户手册》*。

表 3. 类型 1 和类型 2 外设列表

类型	外设
类型 1	SCI、SPI、USBFS、CAN、IIC、SCE9、DOC、SDHI、SSIE、CTSU、CRC、CAC、TSN、ADC12、DAC12、POEG、AGT、GPT、RTC、IWDT 和 WDT
类型 2	系统控制（复位、LVD、时钟发生电路、低功耗模式和电池备份功能）、闪存高速缓存、SRAM 控制器、CPU 高速缓存、DMAC、DTC、ICU、MPU、总线、安全设置、ELC 和 I/O 端口
始终非安全	CS 区域控制器、QSPI、OSPI、ETHERC 和 EDMAC

类型 2 外设的访问权限因外设而异。请参见每个外设的寄存器说明部分。

表 4. 基于 Arm® TrustZone® 的外设访问控制

权限	安全访问	非安全访问
外设配置为安全	允许	忽略写入/忽略读取，生成 TrustZone 访问错误
外设配置为非安全	允许	允许

时钟生成电路 (CGC) 注意事项

时钟生成电路针对每个时钟树控制都有单独的安全属性。当前版本的工具和 FSP 为提供以下的灵活时钟控制方案。

- 整个时钟树仅通过安全项目进行控制，在非安全项目中被锁定。
- 整个时钟树可通过非安全项目和安全项目进行控制。

有关操作细节，请参见时钟控制注意事项。

仅支持非安全分区操作的外设

如表 3 所示，以下三个外设在其安全属性方面有限制。

- **以太网：**有关以太网应用程序开发的限制，请参见第 1.2.2 节
- **CS 区域控制器、QSPI 和 OSPI：**这些外设仅为非安全外设。FSP 支持在全部三种项目类型中使用它们。有关基于项目配置器的项目类型定义，请参见第 3 章。

1.2.4 调试界面

对于支持 Arm® TrustZone® 技术的 RA MCU 产品家族，调试功能分为 DBG0、DBG1 和 DBG2 三个级别，以对支持 TrustZone 技术的调试功能提供支持，并为开发、生产和部署的产品提供安全性。

- DBG2：允许调试器连接，无限制访问存储器和外设。
- DBG1：允许调试器连接，但仅限访问非安全存储区域和外设
- DBG0：不允许调试器连接

调试级别根据产品的器件生命周期状态确定。有关更多详细信息，请参见*《瑞萨 RA6M4 系列用户手册：硬件》*一章中“安全功能”部分的“器件生命周期管理”。

通过器件生命周期管理系统可以实现调试级别的回退。有关相应的操作流程，请参见应用笔记*《瑞萨 RA 产品家族 - 器件生命周期管理密钥安装》*。

J-Link、E2 和 E2 Lite 调试器可以支持带 TrustZone 的瑞萨 RA MCU。

1.3 器件生命周期管理

支持 TrustZone 技术的 RA MCU 产品家族使用 TrustZone 技术功能和安全加密引擎 9 (SCE9) 实现增强型器件生命周期管理系统。器件生命周期管理在启用 TrustZone 技术的应用程序开发、生产和部署阶段非常重要。

有关器件生命周期状态定义和切换的信息，请参见*《RA6M4 硬件用户手册》*。有关在开发和生产阶段创建、安装和使用器件生命周期管理密钥的信息，请参见应用笔记*《瑞萨 RA 产品家族 - 器件生命周期管理密钥安装》*。

1.4 TrustZone 用例示例

此应用项目解释了 TrustZone 技术的两个特定用例，并为 IP 保护用例提供了一个示例软件项目。

有关攻击者可能尝试访问受保护信息的其他攻击场景以及 ARMv8-M 的 TrustZone 技术如何阻止这些攻击场景的信息，请参见*《适用于 Armv8-M 架构的 Arm® TrustZone 技术》*的第 2 章“安全性”。

1.4.1 知识产权 (IP) 保护

IP 保护是专有软件算法或数据保护的常见需求。TrustZone 技术为 IP 保护提供了良好的硬件隔离。TrustZone 技术在安全（“受信任”）和非安全（“非受信任”）代码/数据区域之间形成隔离。创建供他人集成的构建模块的客户可以利用 TrustZone 技术功能，将其软件 IP 存储在安全（“受信任”）区域中。

商业模式

并非所有软件开发人员都会创建最终产品。一些开发人员创建构建模块（例如算法），供其他人集成以创建最终产品。他们面临的一个困难是保护其软件 IP。他们的最终客户更愿意接收源代码，但源代码很容易被复制和重新分发。

即使是二进制库也不受完全保护，因为有些工具可以将二进制文件反汇编为汇编代码、甚至 C 源代码。

TrustZone 技术为这些开发人员提供了新的商业模式，开发人员可以将其算法编程到支持 TrustZone 的 MCU 的安全区域并销售增值 MCU，其 IP 受到 TrustZone 和 RA MCU 的器件生命周期管理 (DLM) 系统的保护。

用于 IP 保护的 RA MCU 器件生命周期管理功能

在开发过程中，DLM 状态回退将擦除受保护区域的闪存内容（除非永久锁定）。这样可以防止读取受保护区域的闪存，从而保护 IP 并在需要修改算法的情况下避免器件报废。

在生产期间，如果算法开发人员希望保留在应用程序已经烧写好的情况下调试其算法的能力，可以安装 DLM 密钥来实现 NSECSD 到 SSD 以及 DPL 到 NSECSD 的切换。有关器件生命周期状态定义和状态回归操作流程的信息，请参见应用笔记*《瑞萨 RA 产品家族 - 器件生命周期管理密钥安装》*。

用于 IP 保护的 RA MCU 闪存块锁定功能

RA MCU 支持临时和永久闪存块保护。这样便可保护客户 IP 和信任根免遭意外擦除和更改。

IP 保护开发、生产和部署流程

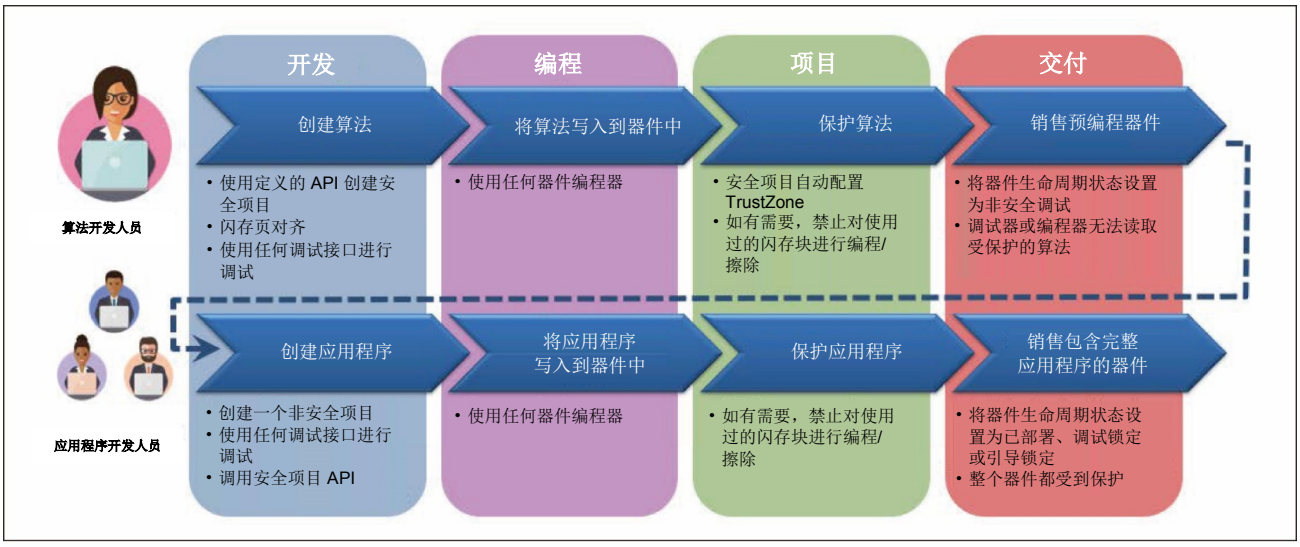

图 4. 使用 Arm® TrustZone® 的 IP 保护

IP 保护设计使用“分离式项目开发模型”。有关操作细节，请参见第 3.2 节。

1.4.2 信任根保护

信任根 (RoT) 是产品的安全基础。所有更高级别的安全都建立在 RoT 之上。RoT 还针对更高级别的安全漏洞实现了恢复功能。当信任根遭到破坏时，无法进行恢复，并可能导致严重的后果。对于物联网应用，信任根是经过身份验证的固件更新的基础，也可以实现安全的互联网通信。

为了减少攻击面，RoT 中包含的功能应该尽可能少。RoT 中的典型服务如下面的图 5 所示。

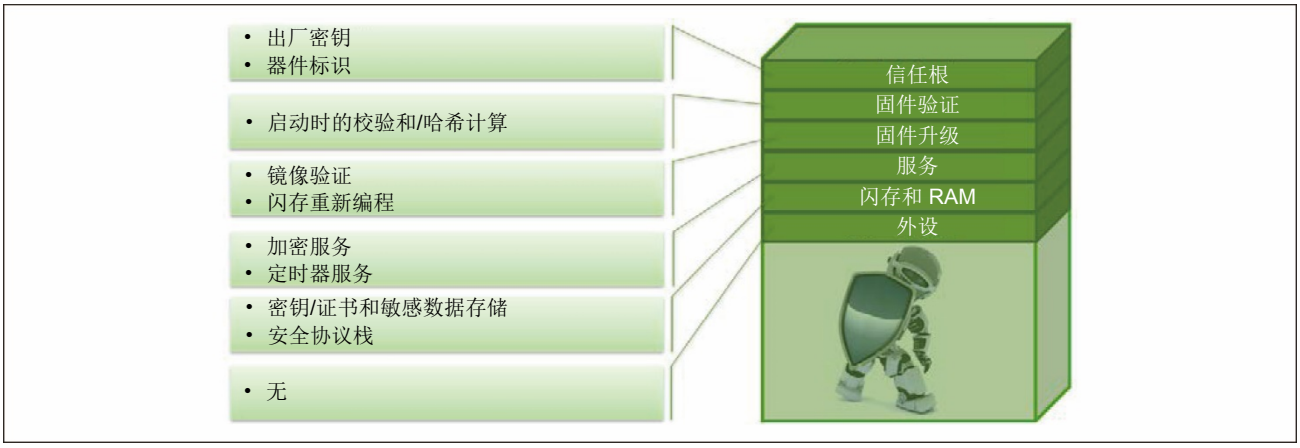

图 5. 信任根保护 - 尽可能减少安全区域中存储的内容

应考虑将所有其他应用程序代码和设备驱动程序分配给非安全区域。

2. 支持 Arm® TrustZone® 应用程序设计的软件功能

本章介绍了几个 IDE 功能，使用 TrustZone 硬件隔离时，这些功能和其他 MCU 硬件组件、FSP 软件或工具结合使用，简化软件开发流程。

2.1 IDAU 区域设置

出厂时，RA MCU 以 CM（芯片制造）生命周期状态交付给开发人员。在设置 IDAU 区域之前，MCU 必须切换到 SSD（安全软件开发）生命周期状态。

RA 产品家族的评估板和支持工具为处理从 CM 到 SSD 状态的切换提供了一种便捷的方法。

- 编译安全项目创建出要写入到安全区域的二进制代码之后，将计算出设置 TrustZone 存储器分区（IDAU 寄存器）所需的值。这些区域设置确保它们与代码和数据大小相匹配，并使攻击面尽可能小。
- 当第一个安全程序下载到 MCU 时，MCU 会自动从 CM 状态切换到 SSD 状态。
- IDAU 寄存器将在下载安全项目二进制文件时（MCU 复位之前）被设置。
- 用户可以使用 **“Renesas Device Partition Manager”**（瑞萨器件分区管理器）读出当前为 MCU 设置的 IDAU 区域，如图 7 所示。

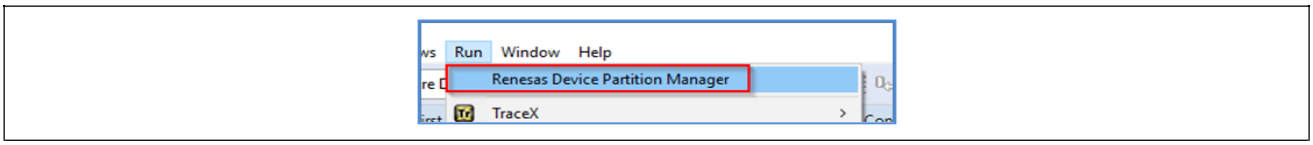

图 6. 打开“Renesas Device Partition Manager”（瑞萨器件分区管理器）

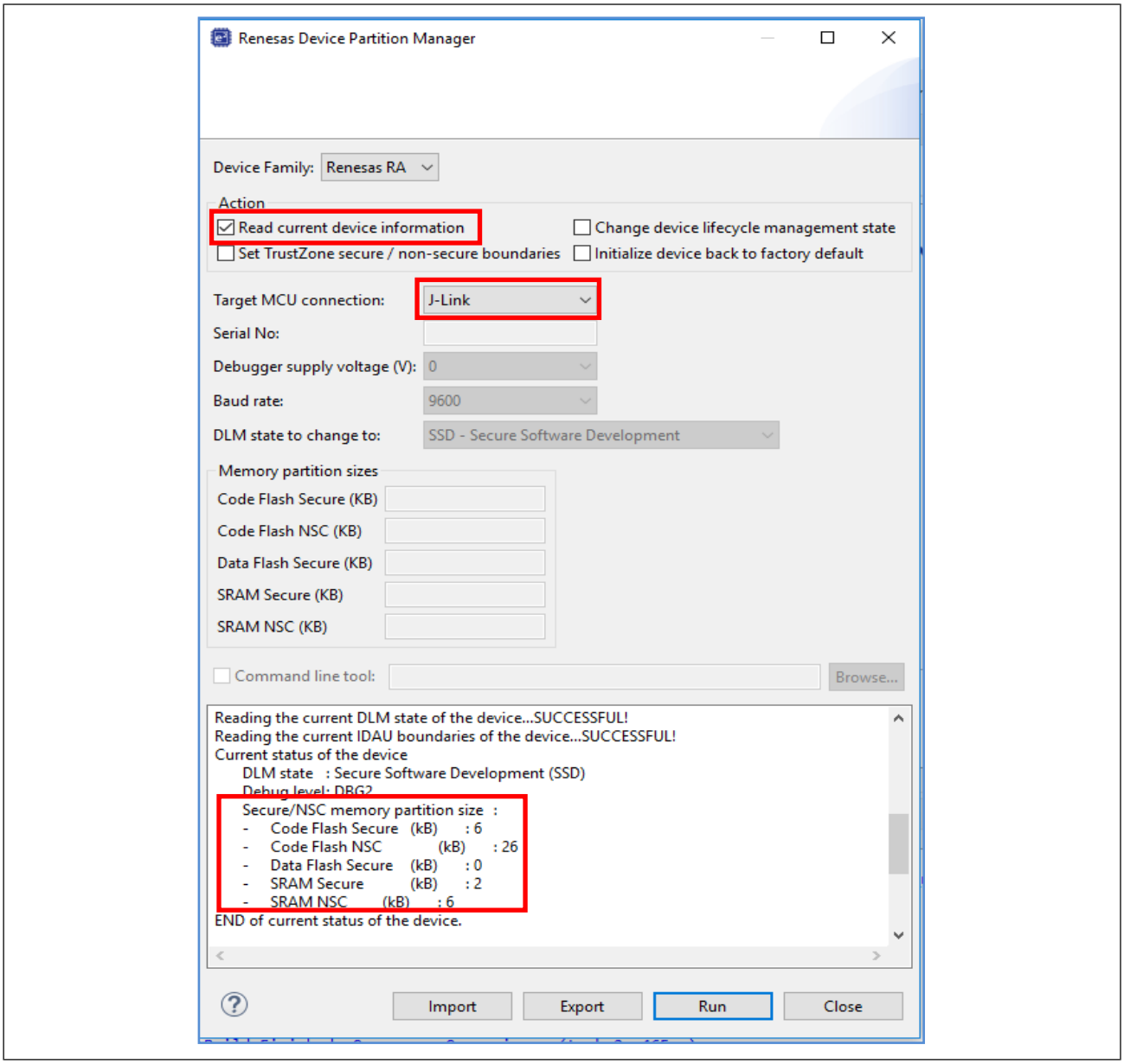

图 7. 使用“Renesas Device Partition Manager”（瑞萨器件分区管理器）读出 IDAU 区域设置

- 每次后续的新安全程序下载到 MCU 时，都会根据新设置更新 IDAU 区域设置。

需要注意的是，上述功能也取决于调试接口的硬件设计。使用 RA6M4 的硬件设计必须参考 EK-RA6M4 调试接口设计，以提供正确的连接来支持上述功能。

用户需要查看《*FSP 用户手册*》的“*入门：Arm® TrustZone® 项目开发*”部分的 IDAU 寄存器以及《*EK-RA6M4 硬件用户手册*》了解如何在使用支持 TrustZone 技术的 RA MCU 产品家族进行开发期间为 IDAU 区域设置准备硬件接口。

2.2 非安全可调用模块

非安全分区可能需要访问安全项目中的一些驱动程序和中间件堆叠。要使能生成 NSC 跳板，请从配置器中所选模块的右键单击上下文菜单中设置**“Non-Secure Callable”**（非安全可调用）。

注：只能将堆叠的顶层模块配置为 NSC。

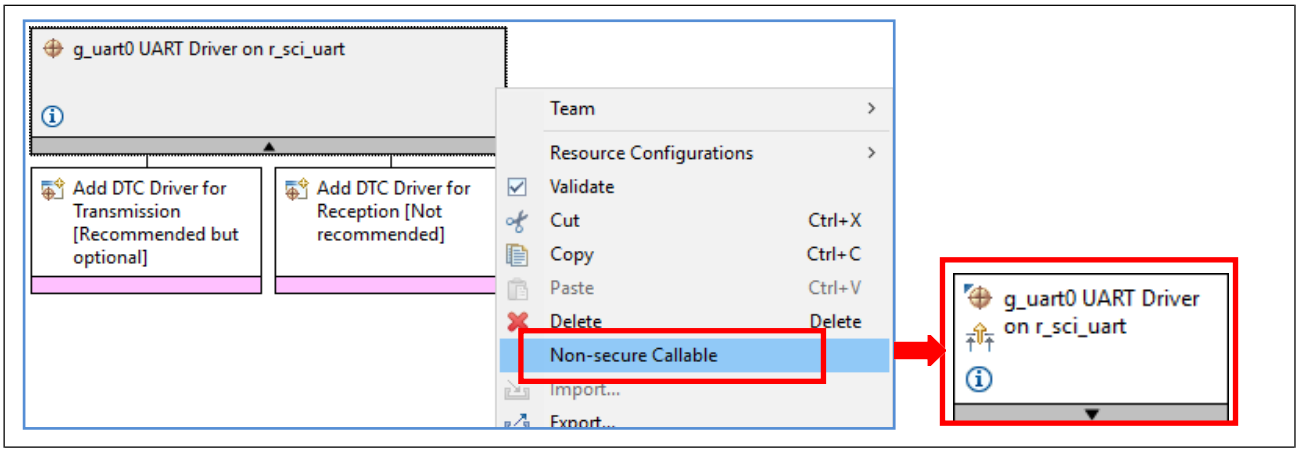

图 8. 生成 NSC 跳板

2.3 非安全可调用的保护函数

可以通过保护 API 从非安全项目访问 NSC 驱动程序。FSP 将自动为安全项目中配置为非安全可调用的所有堆叠顶部模块/驱动程序 API 生成保护函数。

用户在使用保护函数时应遵循的一些最佳实践和指南，如下所示。

2.3.1 限制对选定配置和控制的访问

生成的默认保护函数将忽略从 NS 端传入的 `p_ctrl` 和 `p_cfg` 参数。作为替代，保护函数将基于模块实例提供这些数据结构的静态安全区域实例。

```
BSP_CMSE_NONSECURE_ENTRY fsp_err_t g_uart0_open_guard(
            uart_ctrl_t *const p_api_ctrl, uart_cfg_t const *const p_cfg) {
      /* TODO: add your own security checks here */

      FSP_PARAMETER_NOT_USED(p_api_ctrl);
      FSP_PARAMETER_NOT_USED(p_cfg);

      return R_SCI_UART_Open(&g_uart0_ctrl, &g_uart0_cfg);
}
```

图 9. 保护函数示例

2.3.2 非安全缓冲区位置的测试

- 如果非安全区域提供输入（例如通过调用带数据缓冲区的 `write ()` 函数），则保护函数应检查数据缓冲区是否完全在 NS 区域内。
- 如果非安全区域提供用于存储输出的指针（例如通过调用带存储位置的指针的 `read ()` 函数），则保护函数应检查数据缓冲区是否完全在 NS 区域内。

有关使用 CMSE 库来处理此要求的示例，请参见第 2.7.1 节。

2.3.3 将非安全数据输入结构处理为易失性结构

如果非安全区域提供数据结构作为输入（例如，带 3 个成员的 `typedef'd` 结构），则保护函数应先在安全区域中复制数据结构，然后再传送给安全函数。这样做是因为非安全数据结构应视为易失性结构，非安全区域在调用 NSC 函数后可能会更改其内容。

有关如何处理此要求的示例，请参见第 2.7.2 节。

2.3.4 限制 NSC 函数中的参数数量

编译器使用寄存器 R0 到 R3 来传送参数和返回值。寄存器 R4 到 R12 在函数执行期间使用。被调用的函数将恢复寄存器 R4 到 R12。因此，如果 NSC API 用于具有 4 个以上参数的安全函数，则保护函数应定义一个具有不同原型的函数，该函数将成为处理所有参数的漏斗。新的函数原型应该采用一个数据结构，该结构的成员覆盖安全函数中的所有参数。这意味着非安全代码需要将函数参数放入结构中。然后，保护函数会将数据结构扩展为单独的参数并传送给安全函数。

下图给出了一个 FSP 示例，用于将 `R_SPI_WriteRead` 函数中的 5 个参数缩减为 NSC API 保护函数中的 4 个参数。

```
/** Non-secure arguments for write-read guard function */
typedef struct st_spi_write_read_guard_args
{
    void const          * p_src;
    void                * p_dest;
    uint32_t const        length;
    spi_bit_width_t const bit_width;
} spi_write_read_guard_args_t;

/* This function has been modified to reduce the number of arguments. */
BSP_CMSE_NONSECURE_ENTRY fsp_err_t g_spi0_write_read_guard_fanin(spi_ctrl_t *const p_api_ctrl,
        spi_write_read_guard_args_t *args)
{
    /* Verify all pointers are in non-secure memory. */
    spi_write_read_guard_args_t *args_checked = cmse_check_pointed_object (args, CMSE_AU_NONSECURE);
    FSP_ASSERT (args == args_checked);
    void const *p_src_checked = cmse_check_address_range ((void*) args_checked->p_src, args_checked->length,
                                                          CMSE_AU_NONSECURE);
    FSP_ASSERT (args_checked->p_src == p_src_checked);
    void *p_dest_checked = cmse_check_address_range (args_checked->p_dest, args_checked->length, CMSE_AU_NONSECURE);
    FSP_ASSERT (args_checked->p_dest == p_dest_checked);

    /* TODO: add your own security checks here */

    FSP_PARAMETER_NOT_USED (p_api_ctrl);

    return R_SPI_WriteRead (&g_spi0_ctrl, p_src_checked, p_dest_checked, args_checked->length, args_checked->bit_width);
}
```

图 10. 处理超过 4 个参数的安全函数

2.4 创建用户定义的非安全可调用函数

出于 IP 保护目的，用户可以在安全项目中创建自定义 NSC API，仅公开其算法的顶级控制并将 IP 存储在安全 Arm® TrustZone® 区域中。在创建用户定义的 NSC API 期间应采取上述预防措施。

创建自定义 NSC API 的步骤如下：

1. 通过声明带 `BSP_CMSE_NONSECURE_ENTRY` 的函数来创建非安全可调用自定义函数。
2. 创建一个包含所有自定义 NSC 函数原型的头文件，例如 `my_nsc_api.h`。
3. 使用“Build Variable”（构建变量）包含 NSC 头文件的路径，如下所示。
4. 编译安全项目以创建安全捆绑包。NSC 头文件会自动提取到非安全项目中使用。

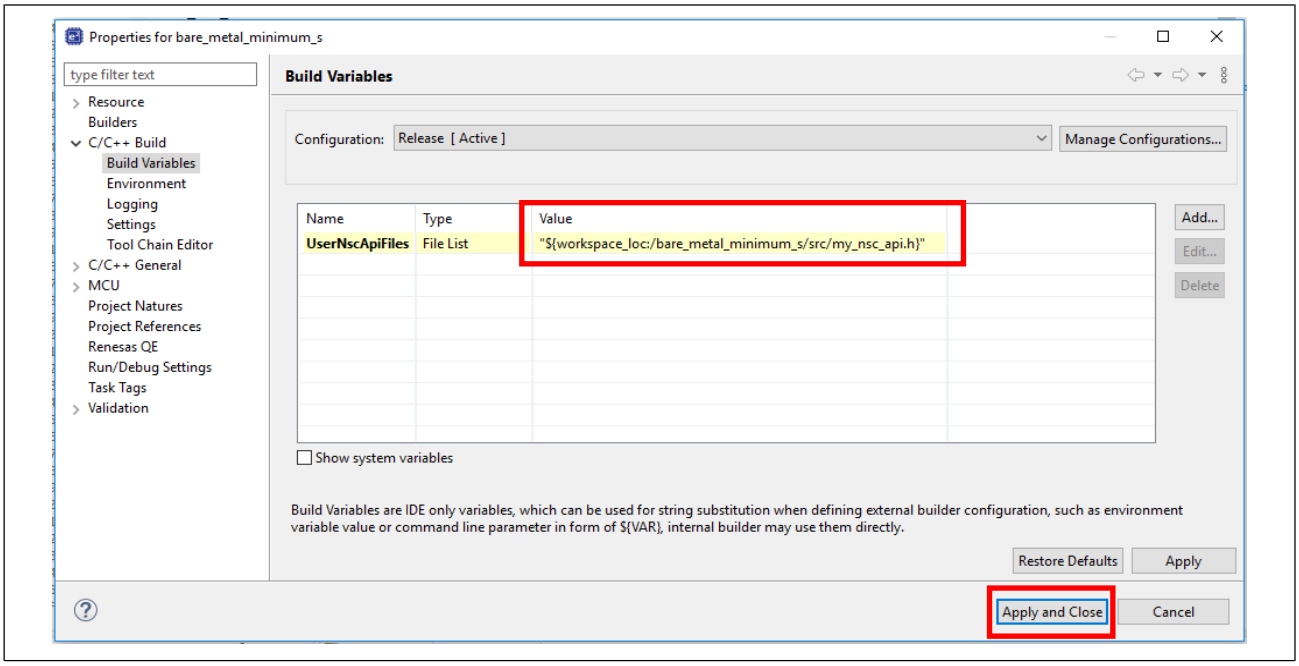

图 11. 链接用户定义的非安全可调用 API 头文件

2.5 RTOS 支持

瑞萨工具和 FSP 支持非安全分区 RTOS 集成，并通过非安全可调用 API 实现安全区域访问。安全项目可以使用 **“Secure TrustZone Support – Minimum”**（安全 TrustZone 支持 – 最小）项目类型添加 Arm® TrustZone® 上下文 RA 端口。有关操作详情，请参见第 3.1.1 节用于“安全项目处理”的步骤 3 和第 3.1.2 节用于“非安全项目处理”的步骤 5。

2.6 第三方 IDE 支持

RA 智能配置器 (RA SC) 支持 IAR Systems EWARM 和 Keil MDK (uVision) 等第三方 IDE。请参见 *《FSP 用户手册》* 中的“适用于 MDK 和 IAR 的 RA SC 用户指南”部分来了解使用这些第三方 IDE 进行开发所需的背景知识。

2.7 编写支持 TrustZone 技术的软件

使用 TrustZone 技术的安全设计面临一些特定的挑战，安全开发人员在编写安全应用软件时应该牢记这些挑战并采取相应的措施。

本节提供安全软件开发人员应考虑遵循的几个准则，以避免安全信息泄漏到非安全区域。

2.7.1 利用 CMSE 函数增强系统级安全性

本小节讨论如何利用 CMSE 库改进安全软件设计。CMSE 函数的一些示例是：

- `cmse_check_address_range`：例如，该函数可用于确认地址范围是否完全在非安全区域
- `cmse_check_pointed_object`：例如，该函数可用于确认指针指向的存储器是否完全在非安全区域

```
BSP_CMSE_NONSECURE_ENTRY fsp_err_t g_uart0_read_guard(uart_ctrl_t *const p_api_ctrl,
uint8_t *const p_dest,
        uint32_t const bytes)
{
    /* Verify all pointers are in non-secure memory.*/
    uint8_t *const p_dest_checked = cmse_check_address_range ((void*) p_dest, bytes,
CMSE_AU_NONSECURE);
    FSP_ASSERT (p_dest == p_dest_checked);

    /* TODO: add your own security checks here */

    FSP_PARAMETER_NOT_USED (p_api_ctrl);

    return R_SCI_UART_Read (&g_uart0_ctrl, p_dest_checked, bytes);
}
```

图 12. 非安全缓冲区地址范围检查

2.7.2 避免对当前处理的数据进行异步修改

处理示例如图 13 所示。当指针 p 指向非安全存储器时，在执行数组边界检查的存储器访问操作之后，但在对数组进行索引的存储器访问操作之前这段时间内，其值可能会更改。对非安全存储器的这种异步更改将导致此数组边界检查无效。

```
int array[N];
void foo(volatile int *p)
{
       int i = *p;
       if (i >= 0 && i < N) { array[i] = 0; }
}
```

图 13. 在安全代码中将非安全数据视为易失性数据

2.7.3 利用 Armv8-M 堆栈指针堆栈限制功能

Armv8-M 架构引入了堆栈限制寄存器，可在堆栈溢出时触发异常。采用 Arm® TrustZone® 技术的 CM23 有两个处于安全状态的堆栈限制寄存器：

- 主堆栈的堆栈限制寄存器：MSPLIM_S
- 进程堆栈的堆栈限制寄存器：PSPLIM_S

采用 TrustZone 技术的 CM33 有两个处于安全状态的堆栈限制寄存器和两个处于非安全状态的堆栈限制寄存器：

- 处于安全状态的主堆栈的堆栈限制寄存器：MSPLIM_S
- 处于安全状态的进程堆栈的堆栈限制寄存器：PSPLIM_S
- 处于非安全状态的主堆栈的堆栈限制寄存器：MSPLIM_NS
- 处于非安全状态的进程堆栈的堆栈限制寄存器：PSPLIM_NS

用户可以采用自定义的故障处理程序来捕捉堆栈限制溢出错误。

有关堆栈限制寄存器功能的更多信息，用户可以参见*《Arm® v8-M 架构参考手册》*的“Armv8-M 架构配置文件”部分。

3. 支持 TrustZone 技术的应用程序开发过程

瑞萨 e² studio IDE 专为基于 TrustZone 技术的应用程序而设计。从工具和 FSP 的角度来看，它基于以下实现功能提供了易用性：

- 强大的 RA 项目生成器可指导用户完成 TrustZone 项目创建过程。
- 安全程序下载期间，基于安全项目自动计算 TrustZone IDAU 的区域设置。更多细节，请参见第 2.1 节。
- 瑞萨灵活配置软件包 (FSP) 提供了一种快速的通用方法，可使用瑞萨 RA MCU 构建安全连接的物联网设备。

请注意，在本文档中，从屏幕截图中删除了 FSP 版本信息，因为这些说明适用于所有 FSP 版本 2.0.0 或更高版本。此声明适用于以下所有部分。

RA 项目生成器

RA 项目生成器提供了三种项目类型来创建使用支持 Arm® TrustZone® 技术的 MCU 进行开发的初始模板项目：

- **安全项目**和**非安全项目类型**对，分别与安全分区和非安全分区一起使用
- **扁平化项目**，可用于在没有 TrustZone 分区感知的情况下开发应用程序

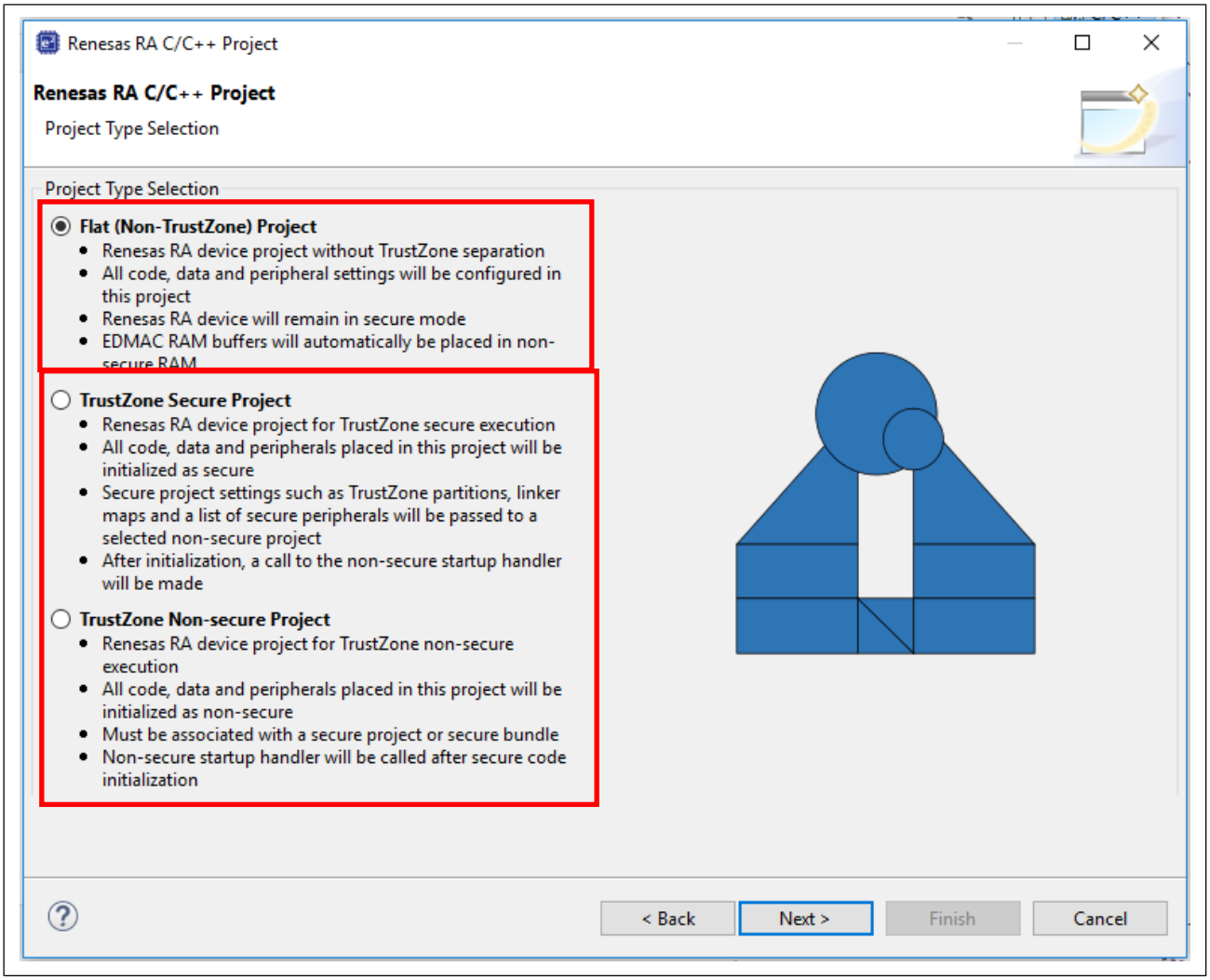

图 14. RA 项目生成器

对于支持 RA TrustZone 技术的 MCU，有两种开发模型。随后的小节将介绍基于这两种开发模型中的每一种的设计过程。第 4 章将介绍基于扁平化项目类型的设计过程。

- 联合式项目开发
  - 安全和非安全应用程序由一个受信任的团队开发
- 分离式项目开发
  - 安全和非安全应用程序由两个不同的团队开发
  - 非安全应用程序团队无法直接访问安全分区资产，只能通过非安全可调用 API 访问安全分区。

3.1 联合式项目开发

使用联合式项目开发模型时，安全和非安全项目由同一个受信任的团队开发。安全项目必须与非安全项目位于同一工作区中，并且通常在设计工程师有权访问安全和非安全项目源时使用。

此外，将引用一个安全的 `.elf` 文件并将其包含在调试编译的调试配置中，以便下载到目标器件。开发工程师将能够查看安全和非安全项目的源代码和配置。

3.1.1 开发安全项目

安全项目中定义的大多数外设和 IO 都配置为安全，但时钟、QSPI、OSPI 和 CS 区域除外。这些外设可用于安全项目，但配置为非安全。

以下步骤说明了开发安全项目的主要 IDE 操作步骤。

步骤 1：使用 RA 项目生成器模板创建新项目

瑞萨 RA MCU 工具提供了多个项目模板来帮助用户启动开发流程。

图 15 到图 19 给出了使用 e² studio 创建新项目时的一些常见步骤，无论是使用分离式项目开发模型还是联合式项目开发模型创建安全或非安全项目，这些步骤均适用。

- 此步骤将在联合式项目开发模型的非安全项目开发的上下文中引用
- 此步骤将在分离式项目开发模型的安全和非安全项目开发的上下文中引用

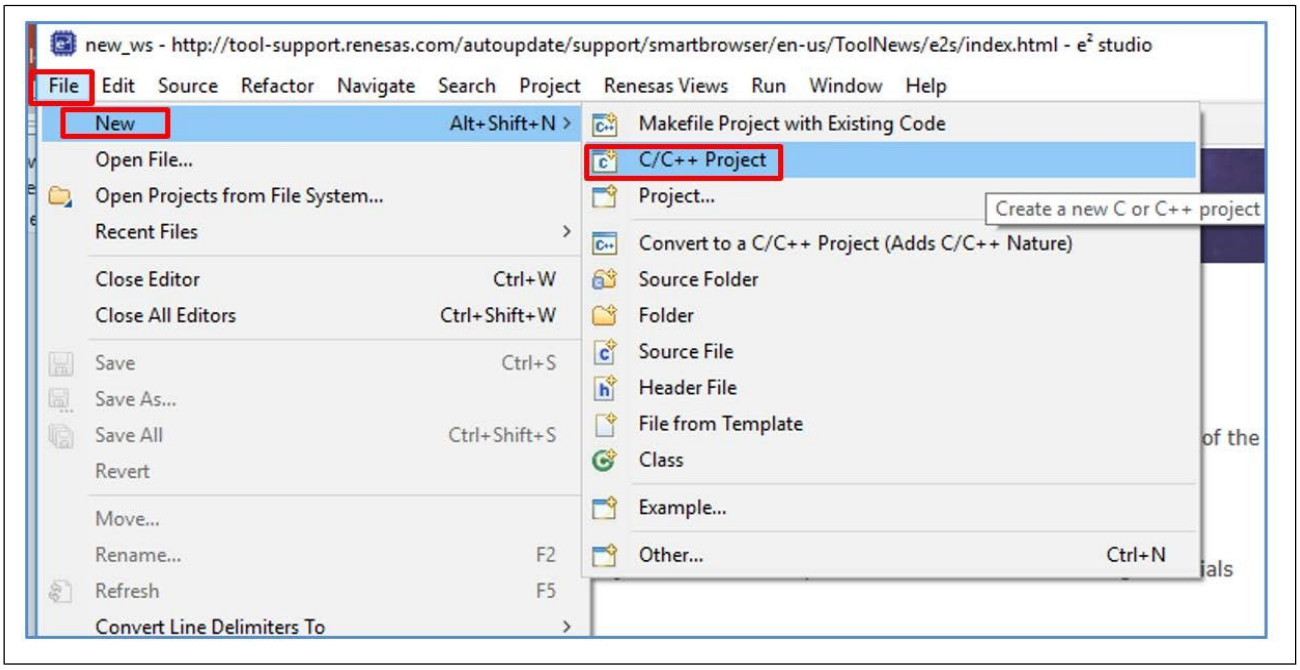

图 15. 创建新项目

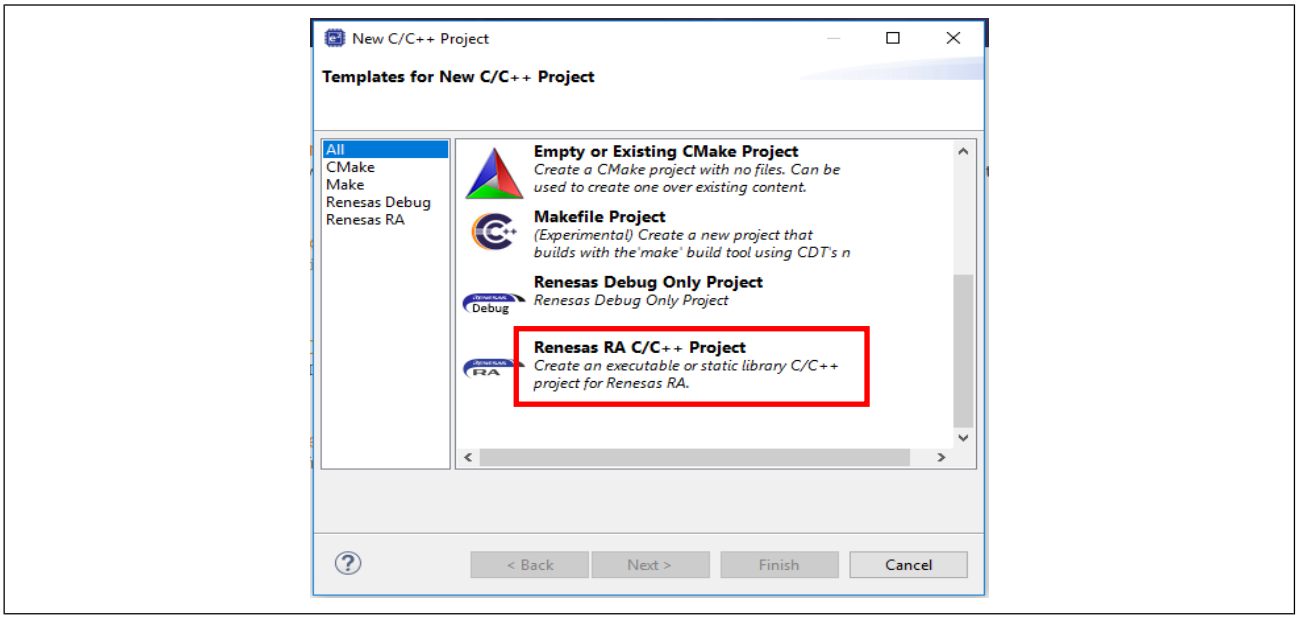

图 16. 选择“Renesas RA C/C++ Project”（瑞萨 RA C/C++ 项目）

单击**“Finish”**（完成），然后提供安全项目名称。请注意，最好在项目名称的末尾附上“_s”（用于安全项目）和“_ns”（用于非安全项目），以此提示该项目的安全属性。

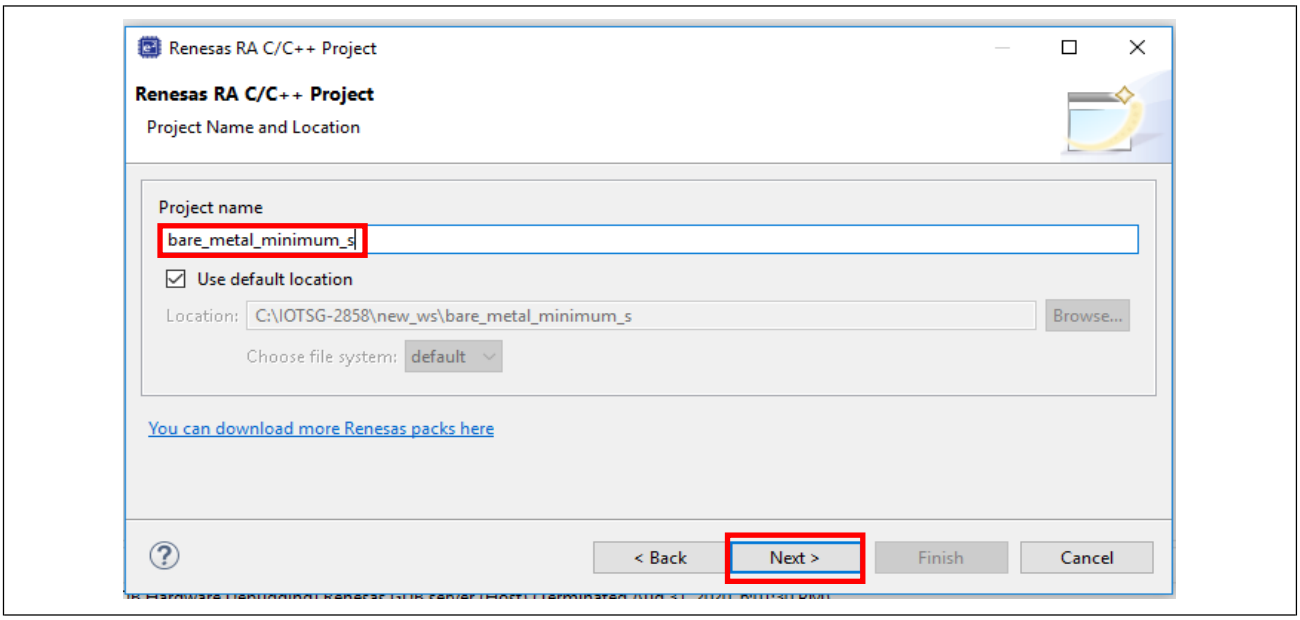

图 17. 定义安全项目的名称

单击“**Next**”（下一步），然后选择 EK-RA6M4 BSP。

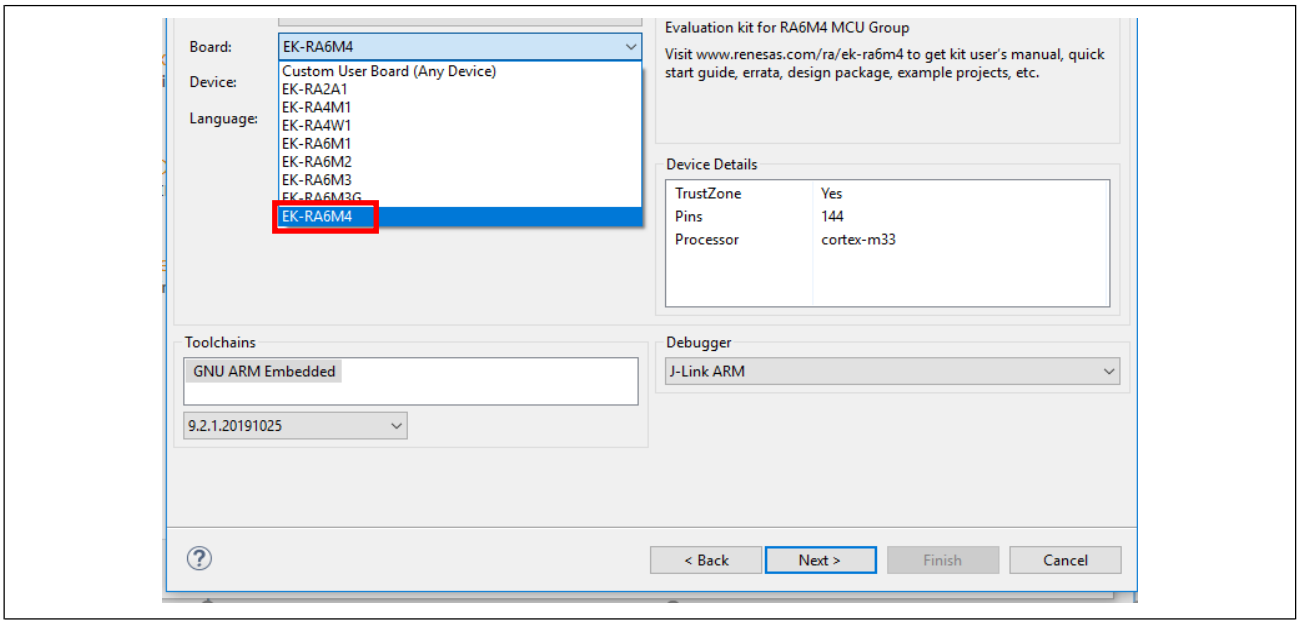

图 18. 选择 BSP

请注意，默认情况下，有关安全控制的 BSP 功能仅在安全项目中使能。

选择 BSP 后，单击“**Next**”（下一步），然后查看硬件设置页面的汇总。

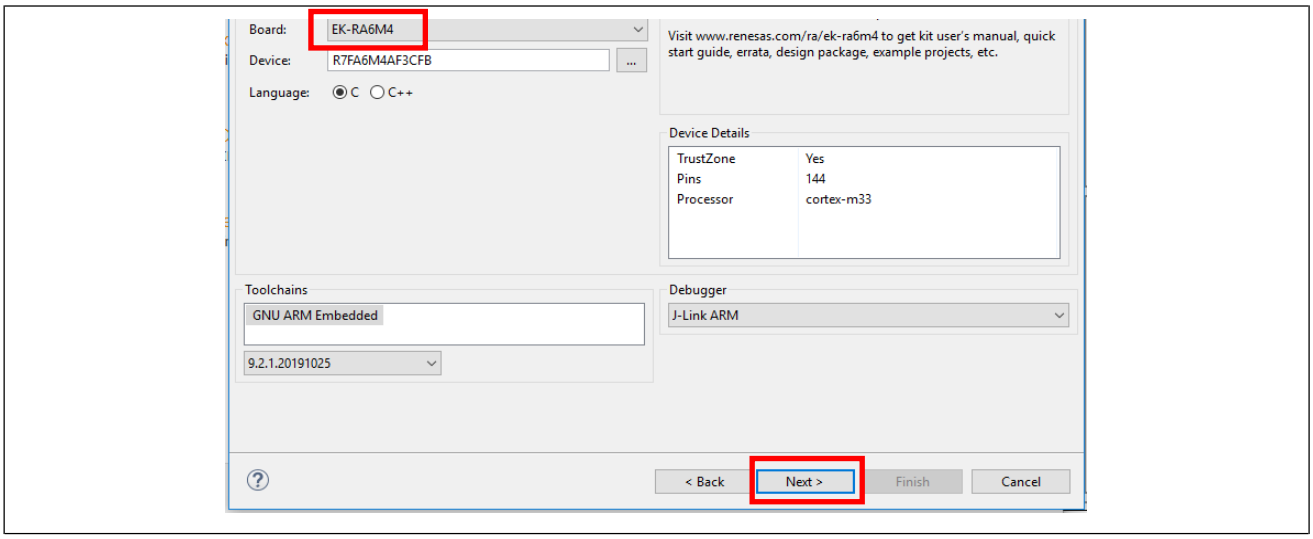

图 19. 在继续下一步之前检查配置

再次单击“**Next**”（下一步），继续执行接下来的步骤。

请注意下面的步骤 2 到步骤 7 是分离式项目开发模型和联合式项目开发模型的常见步骤。

这些步骤将在分离式项目开发模型的安全项目开发上下文中引用。

步骤 2：选择安全项目作为项目类型

选择“**Secure Project**”（安全项目）作为项目类型并花点时间阅读有关此项目类型的说明。在此项目中初始化的所有外设都将被假定为具有安全属性，但表 3 中列出的例外情况为**始终非安全**。置于此项目中的所有代码和数据将由 FSP BSP 初始化为安全，并且控制权将在安全项目执行结束时交由非安全项目复位处理程序。

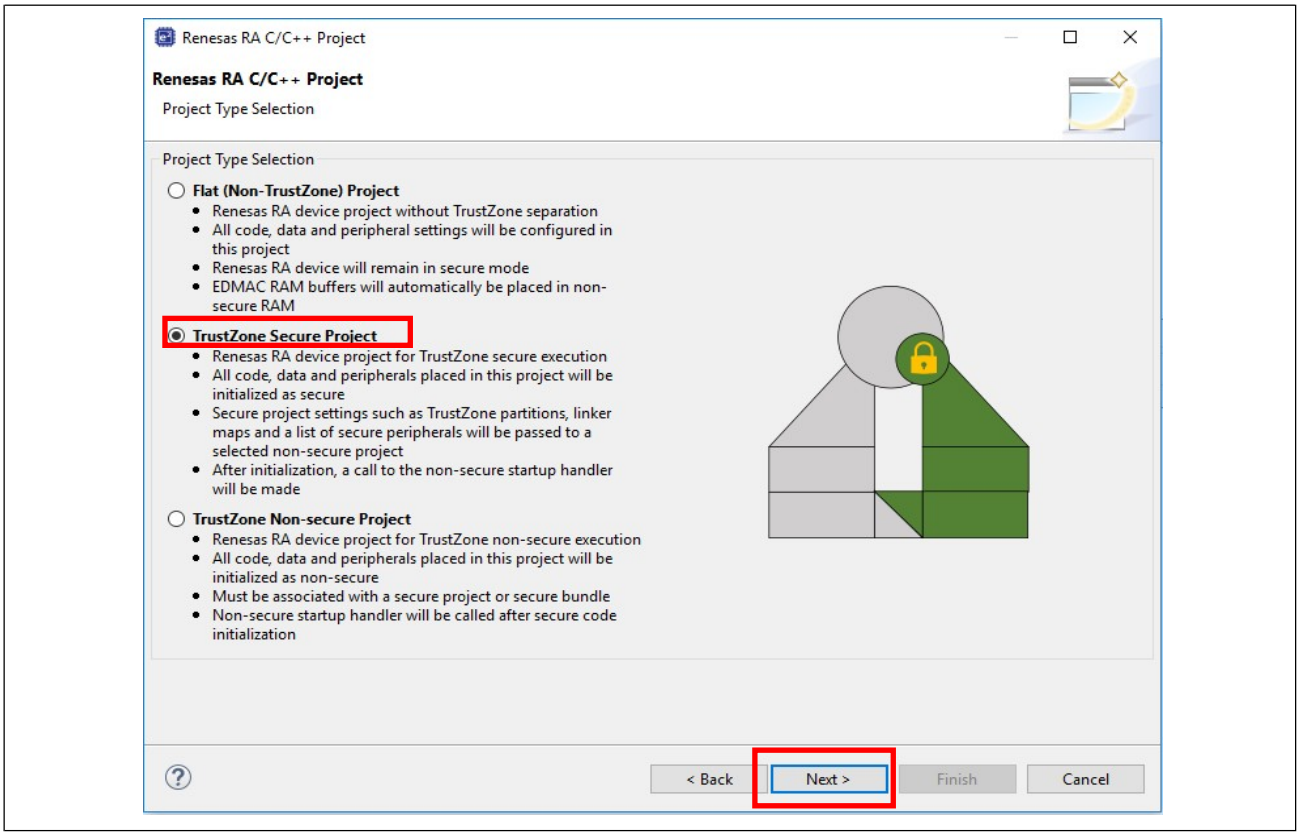

图 20. 选择安全项目类型

单击**“Next”**（下一步），选择项目模板。

步骤 3：选择项目模板

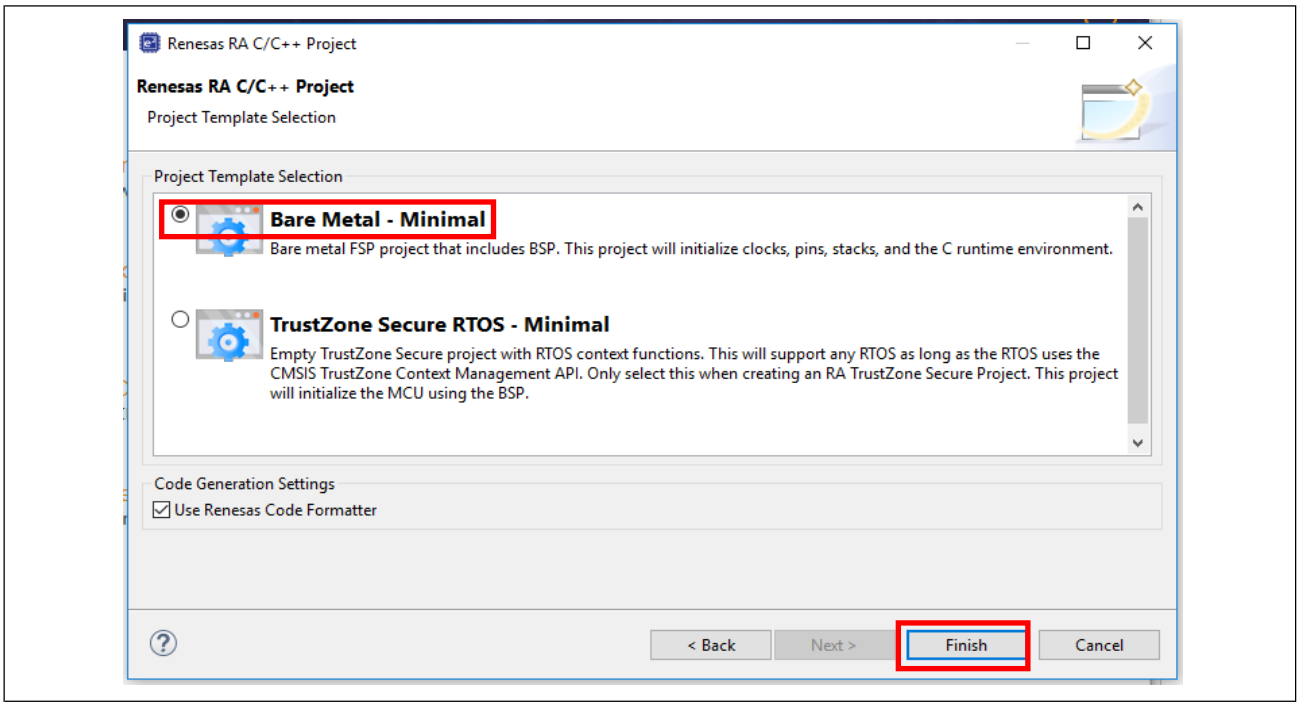

图 21. 选择项目模板

如图 21 所示，共有三个安全项目模板：

- **“Bare Metal – Minimal”（裸机 – 最小化）**

  具有 MCU 初始化功能的安全项目，支持切换到非安全分区。本文使用此项目模板作为示例来说明创建安全项目的一般步骤。

- **“TrustZone Secure RTOS - Minimal”（TrustZone 安全 RTOS - 最小化）**
  - 安全项目将在安全区域中为 RTOS 项目中需要访问NSC API 的线程添加相应的 RTOS 上下文。选择此项目类型后，将添加 Arm TrustZone 上下文 RA 端口，如图 22 所示。
  - RTOS 内核和用户任务将驻留在非安全分区中。

单击**“Finish”**（完成），允许项目生成器填充模板项目。

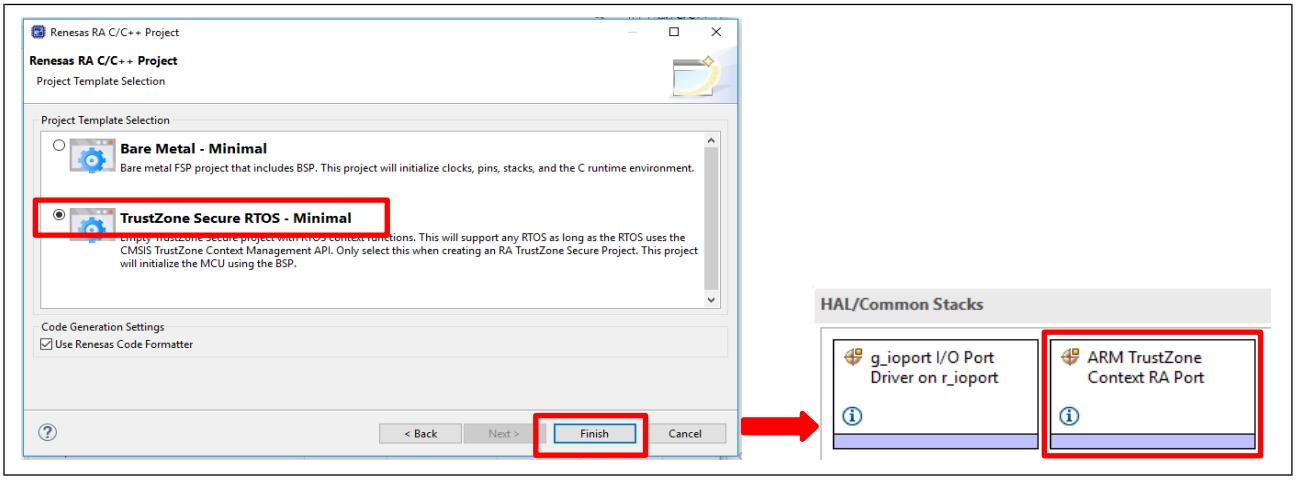

图 22. 添加 TrustZone 上下文 RA 端口

时钟控制注意事项

时钟将在安全项目中初始化以允许更快启动。默认情况下，FSP 将时钟生成电路 (CGC) 的所有安全属性设置为非安全，如图 23 所示。因此，安全和非安全项目都可以更改时钟设置。

用户可以选择将 CGC 的所有安全属性设置为安全，因此非安全项目开发人员无法覆写安全项目设置，如图 24 所示。

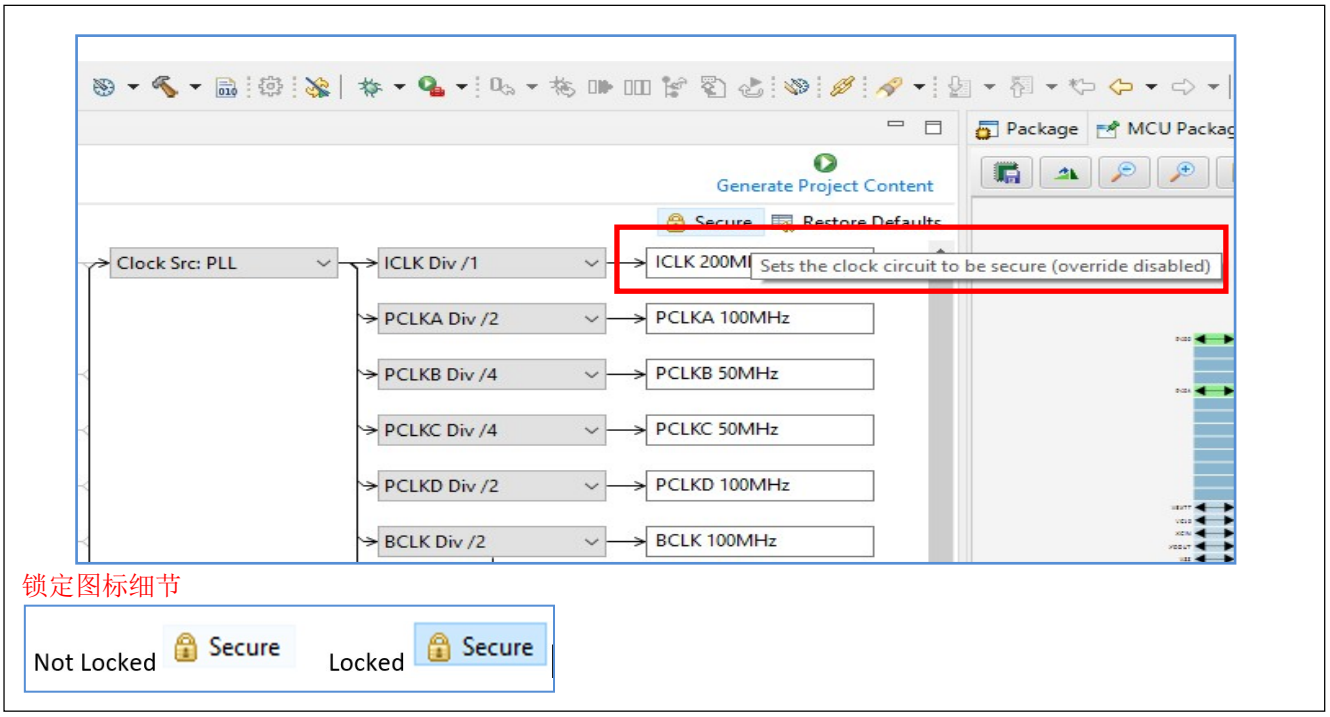

图 23. 安全项目将时钟设置为安全

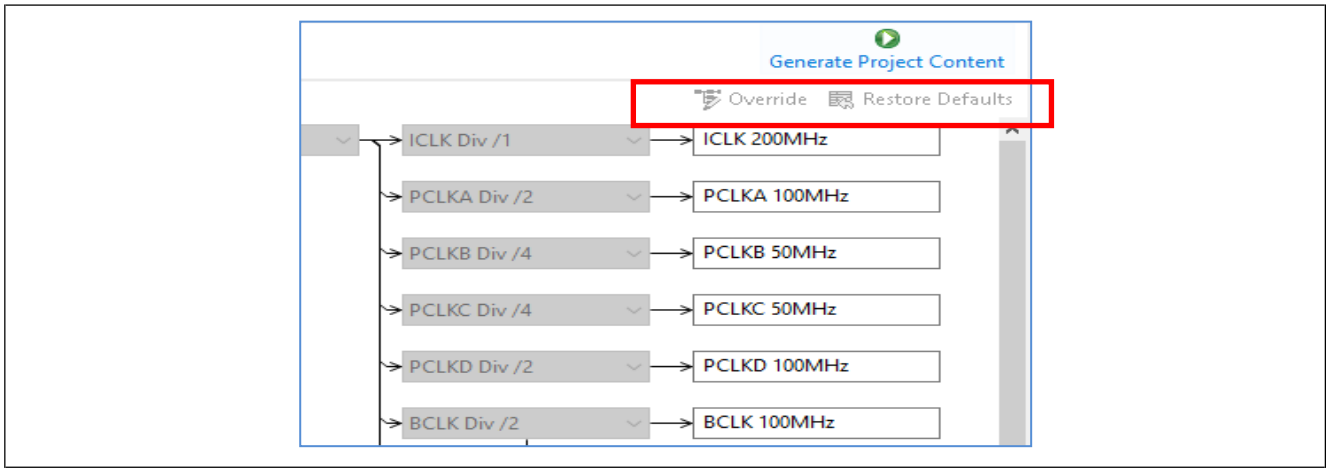

图 24. 禁止非安全项目时钟控制“覆写和恢复默认值”

步骤 4：生成项目内容并编译项目模板

双击 Configuration.xml 打开配置器。单击**“Generate Project Content”**（生成项目内容），如图 25 所示。

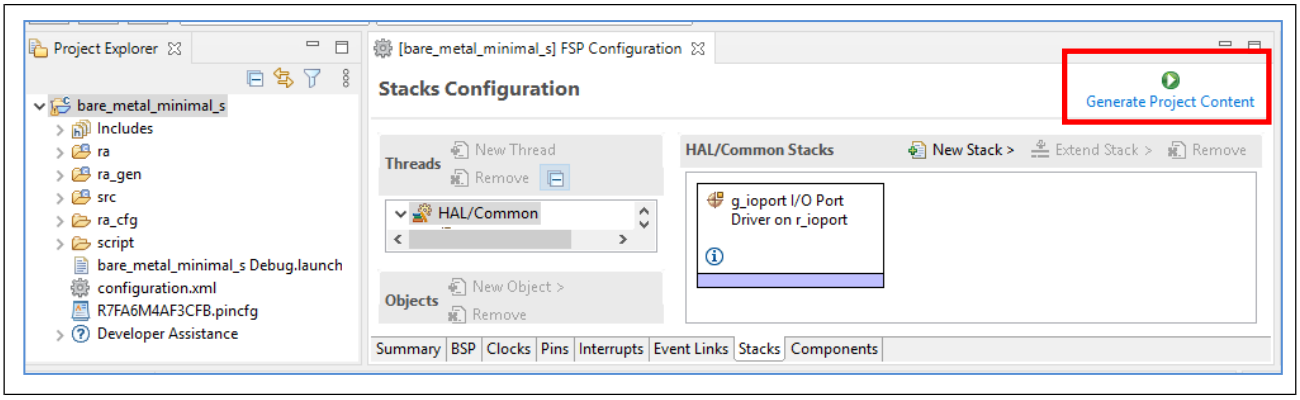

图 25. 生成项目内容

右键单击项目，选择**“Build Project”**（编译项目）。

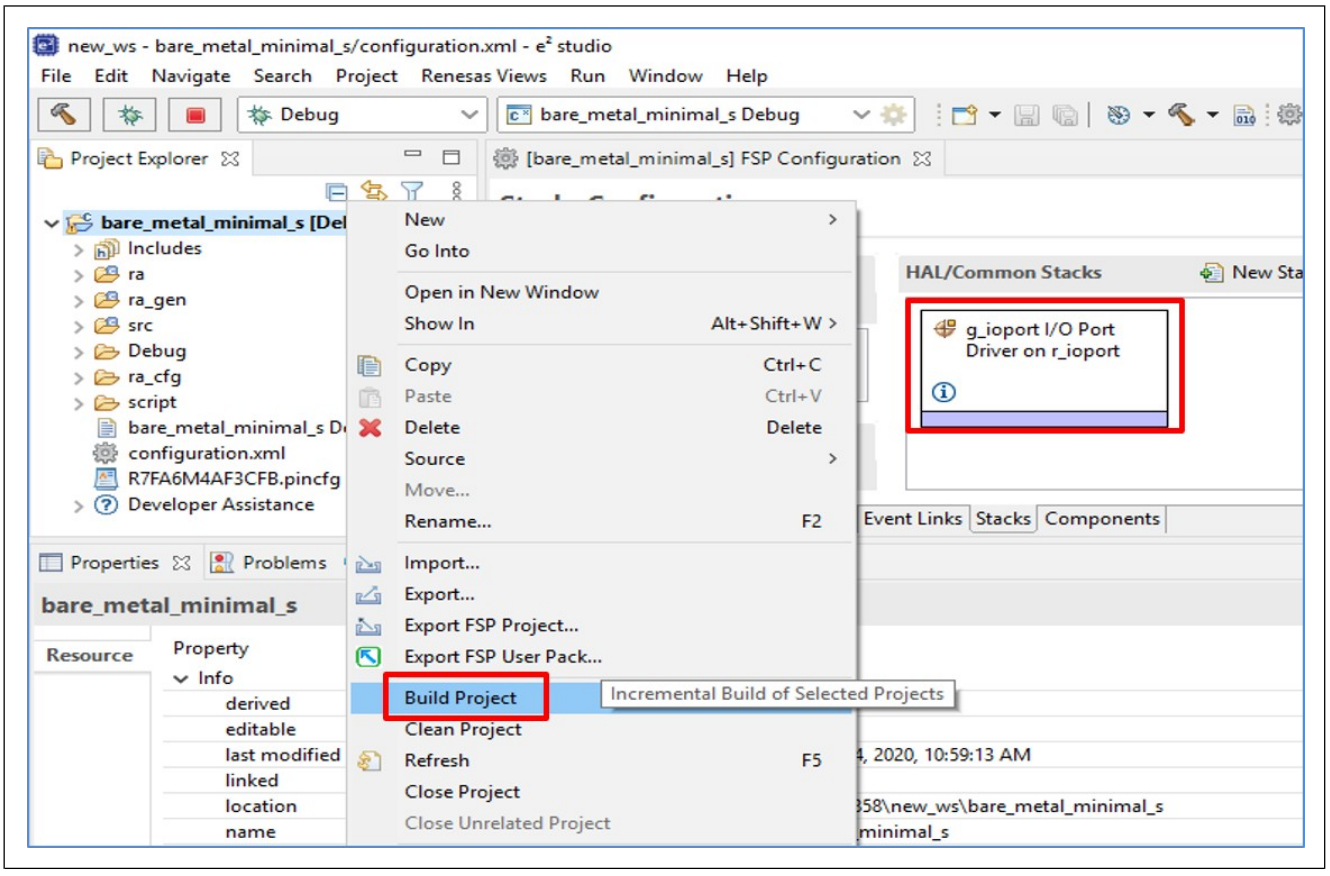

图 26. 编译模板项目

请注意，默认情况下，模板中包含用于控制安全 GPIO 引脚的 GPIO 驱动程序。如果不需要，用户可以将其删除，以减少项目占用空间。

下面是基于**裸机最小化**项目模板的编译结果示例。

```
CDT Build Console [bare_metal_minimal_s]
Extracting support files...
14:51:54 **** Incremental Build of configuration Debug for project bare_metal_minimal_s ****
make -j8 all
'Invoking: GNU ARM Cross Print Size'
arm-none-eabi-size --format=berkeley "bare_metal_minimal_s.elf"
   text    data     bss     dec     hex filename
   4572       8    1168    5748    1674 bare_metal_minimal_s.elf
'Finished building: bare_metal_minimal_s.siz'
' '

14:51:55 Build Finished. 0 errors, 0 warnings. (took 1s.501ms)
```

图 27. 裸机最小化安全模板项目编译结果

步骤 5：查看生成的初始安全捆绑包

编译成功后，将生成安全捆绑包 `<project_name>.sbd`，如图 28 所示。

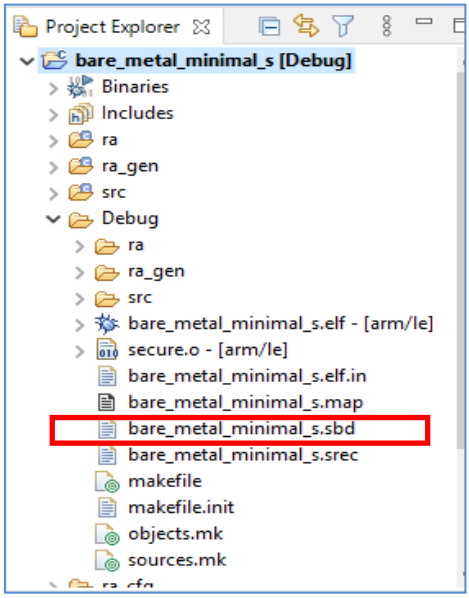

图 28. 生成的安全捆绑包

步骤 6：开发安全应用程序

在产品开发过程中，用户在完成开发之前，很可能会反复经历以下步骤：

- 添加所需的 FSP 模块
  - 如果需要，定义 NSC 模块。有关详细信息，请参见第 2.2 节。
  - 注：以太网不能用于安全项目，只能用于非安全项目。
- 如有需要，创建用户定义的非安全可调用函数。有关详细信息，请参见第 2.4 节。
- 开发安全应用程序
  - 设计代码流，在开始非安全项目执行之前运行不需要非安全可调用功能的安全应用程序：在调用函数 `R_BSP_NonSecureEnter();` 之前
- 重新编译并测试应用程序

步骤 7：隔离调试安全项目

采用联合式项目开发时，安全项目的调试通常不会与非安全项目的调试进行隔离。要单独调试安全项目，用户可以使用以下两个选项：

- 准备一个“虚拟/测试”非安全项目。这种方法的优势在于允许在测试非安全项目中调试非安全可调用 API。
- 在 `hal_entry.c` 中将 `R_BSP_NonSecureEnter();` 替换为 `while(1);`，然后单独调试安全项目。
  - 重要提示：用户需要在调试安全项目之后恢复 `R_BSP_NonSecureEnter();`，才能为 MCU 写入安全项目。

步骤 8：调试安全项目和非安全项目

对于联合式项目开发模型，可以在一个工作区中调试安全和非安全项目开发。对于联合式项目开发模型，安全项目的调试通常不会以隔离方式进行。有关操作细节，请参见第 3.1.2 节的步骤 **7**。

3.1.2 开发非安全项目

建立并编译安全模板项目后，用户便可在安全项目所在的同一工作区中开始创建非安全模板项目。

步骤 1：按照第 3.1.1 节中的步骤 1 启动新的非安全项目。

请注意，最好在项目名称的末尾附上“_ns”，以此提示该项目的安全配置。

步骤 2：选择非安全项目作为项目类型。

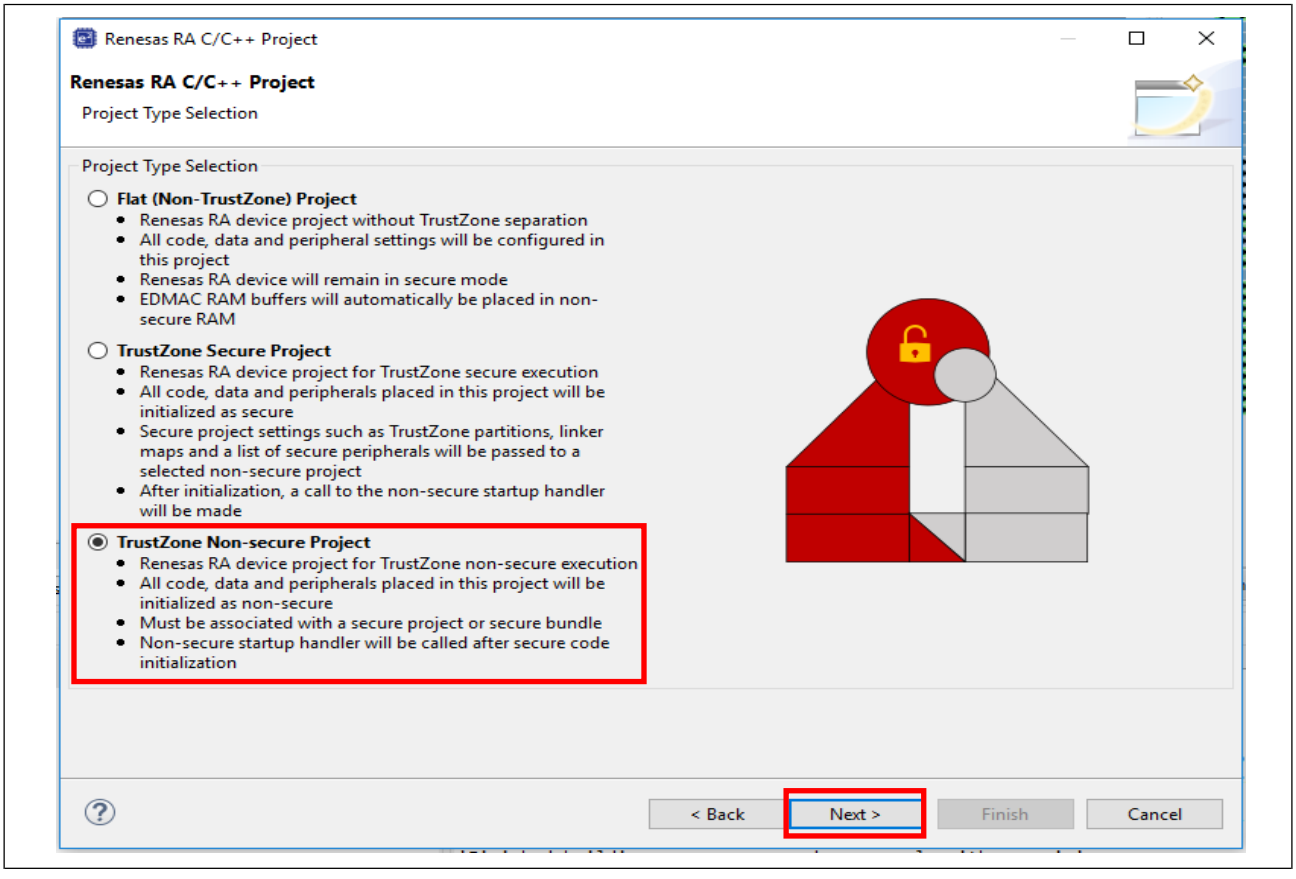

图 29. 选择非安全项目作为项目类型

步骤 3：建立到位于同一 e² studio 工作区中的安全项目的链接。

单击向下箭头，选择第 3.1.1 节中创建的安全项目 **bare_metal_minimum_s**。

请注意，安全项目必须位于同一工作区中，并且必须处于打开状态才能在选择框中引用。此外，还必须编译安全项目以创建用于设置非安全项目的信息。

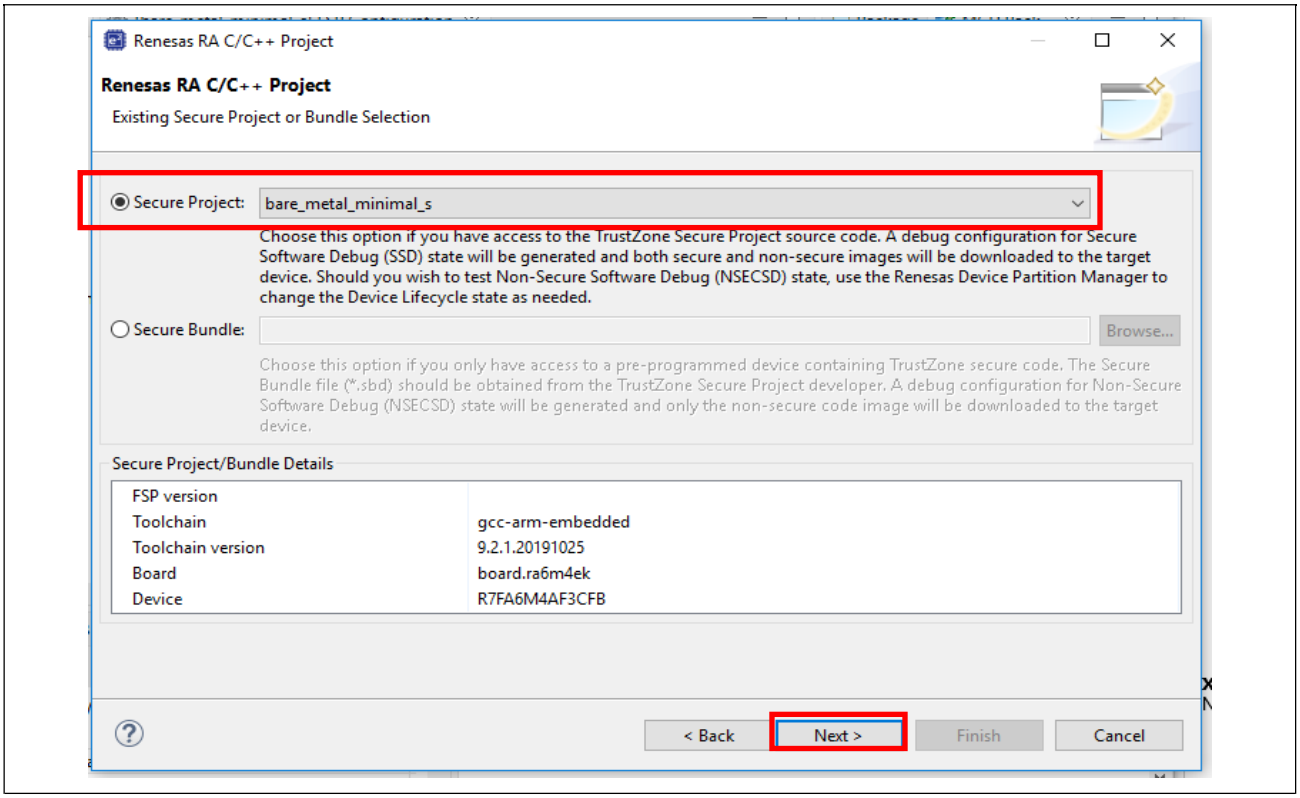

图 30. 建立到安全项目的链接

单击“**Next**”（下一步）继续。

步骤 4：按照下图提示选择非安全项目是否支持 RTOS。

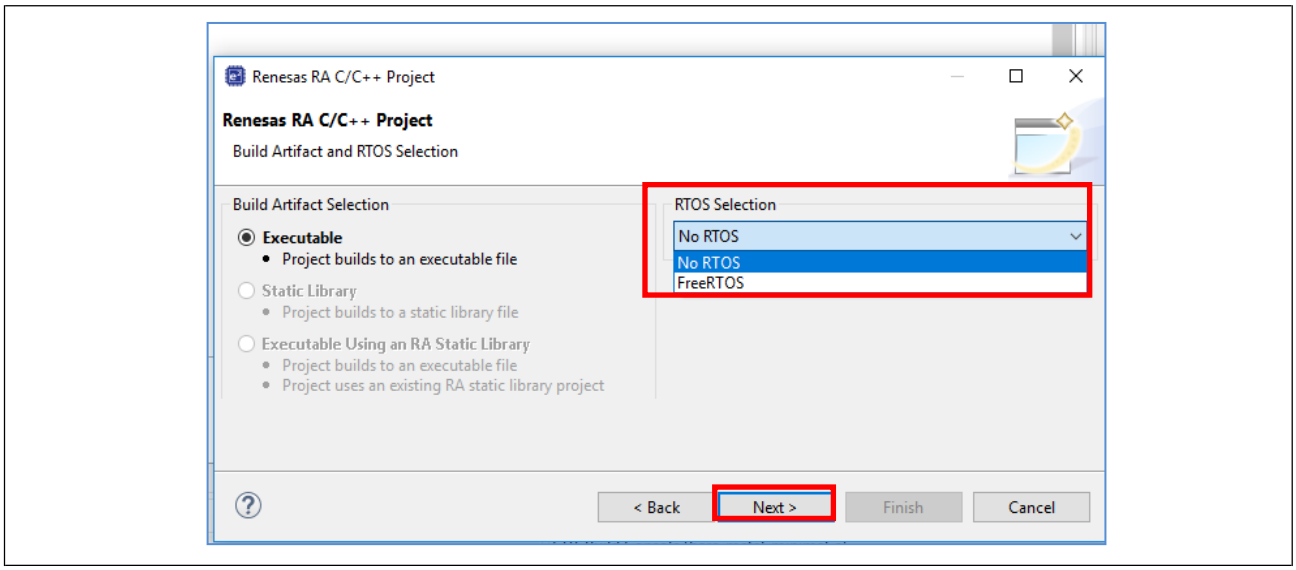

图 31. 选择是否在非安全项目中使用 FreeRTOS

单击“**Next**”（下一步）继续。

步骤 5：选择项目模板以完成创建非安全模板项目

- 如果选择 FreeRTOS，项目生成器提供以下两个项目模板。根据应用需求选择项目模板。

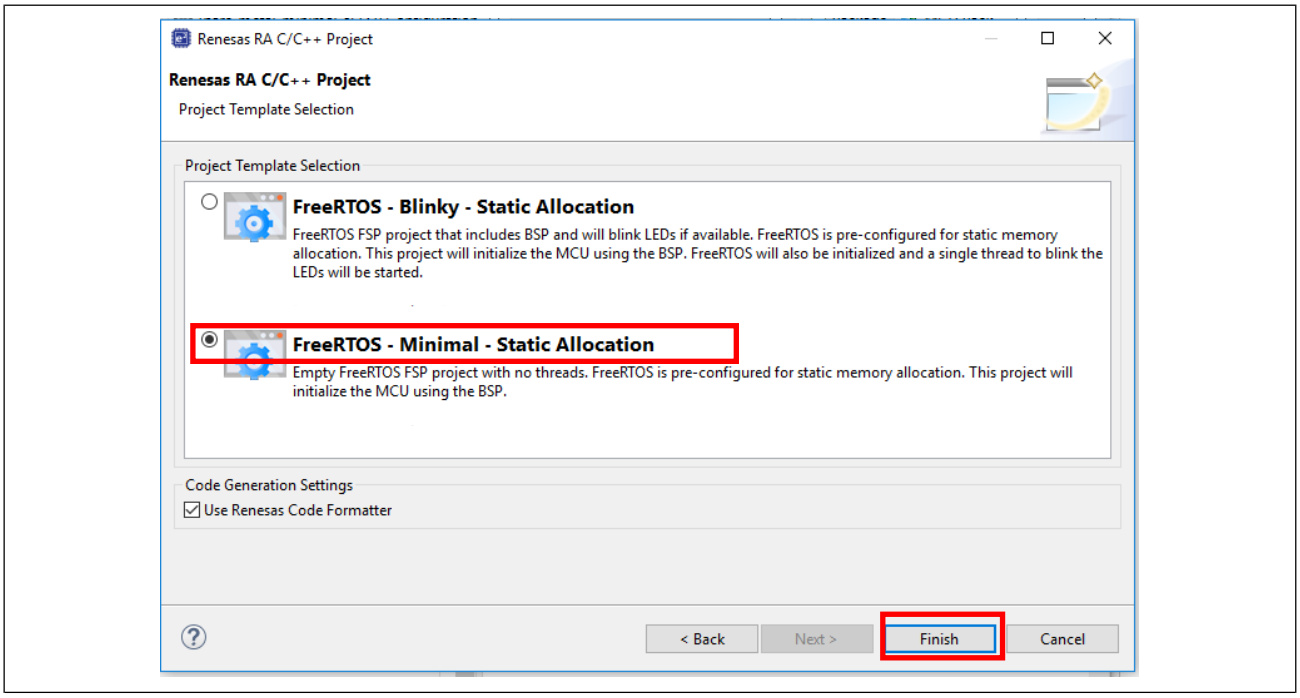

图 32. 支持 FreeRTOS 的项目的模板选项

请注意，如果选择了 FreeRTOS 并且从非安全项目中的线程访问 NSC 函数，则需要在配置器中针对该线程使能 **“Allocate secure context for this thread”**（为该线程分配安全上下文）。

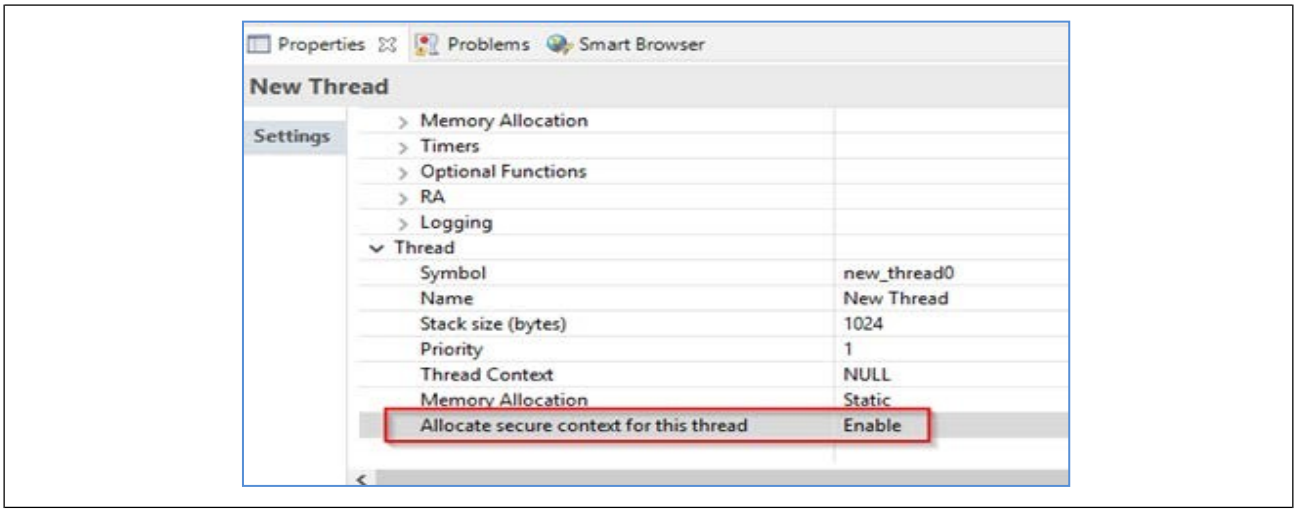

图 33. 使能安全上下文分配

- 如果选择“**No RTOS**”（无 RTOS），项目生成器会提供以下两个项目模板。
  如果要在非安全项目中集成除 FreeRTOS 之外的新 RTOS，则必须选择“**No RTOS**”（无 RTOS）选项。

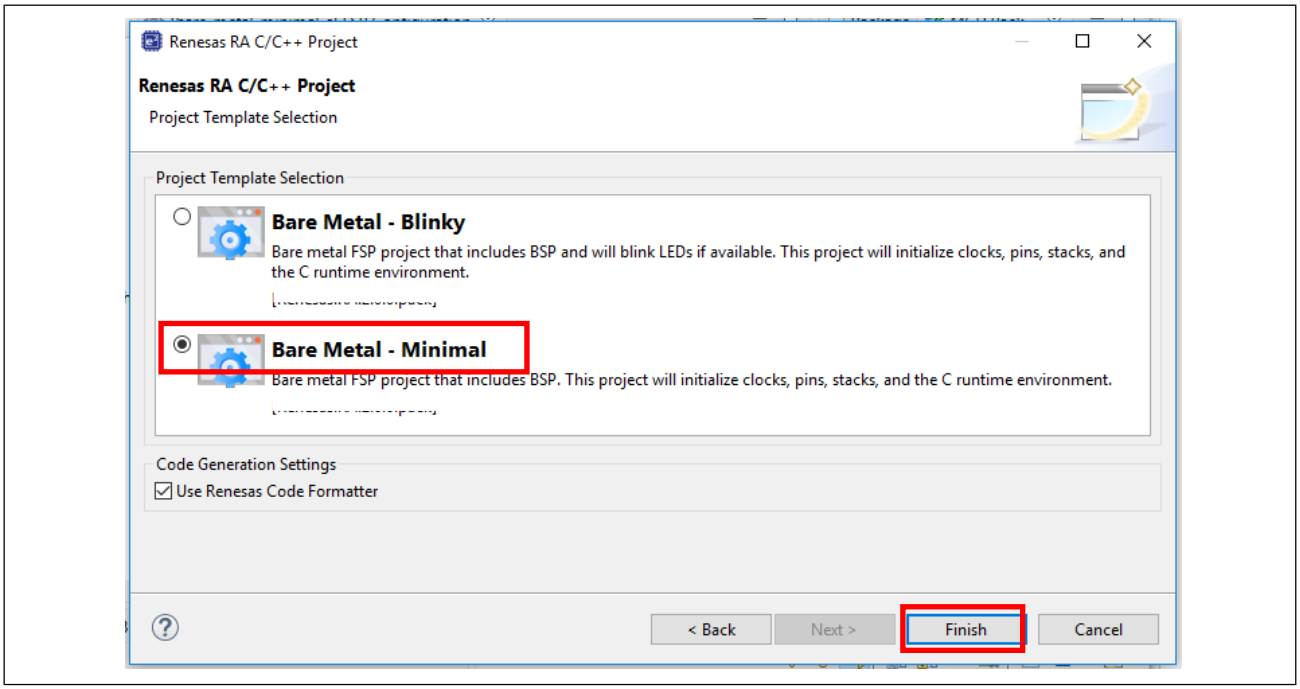

图 34. 非 FreeRTOS 应用的模板选项

- 单击“**Finish**”（完成）创建相应的模板项目。

  请注意，即使允许在 BSP“**Properties**”（属性）页面中配置某些安全属性，在当前的 IDE 支持下也不会将其使能。无法通过非安全项目配置以下属性：

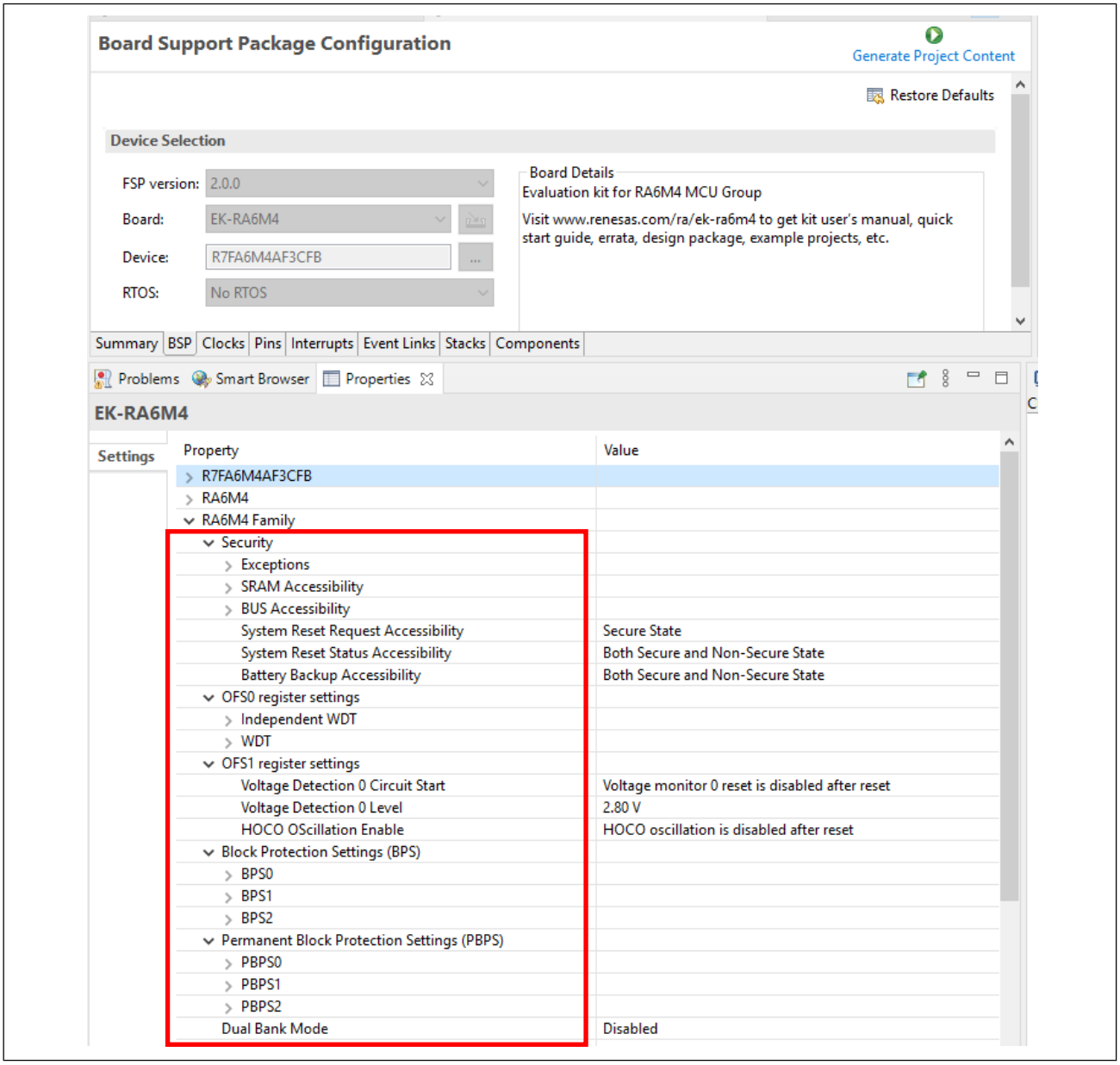

图 35. 不可通过非安全项目配置的属性

- 默认情况下，非安全项目 BSP 可以重新配置 MCU 时钟。请参见时钟控制注意事项。

步骤 6：按照第 3.1.1 节的步骤 1 的说明生成项目内容并编译非安全项目

请注意，安全项目 **bare_metal_minimun_s** 和 **bare_metal_minimum_ns** 位于同一个工作区。

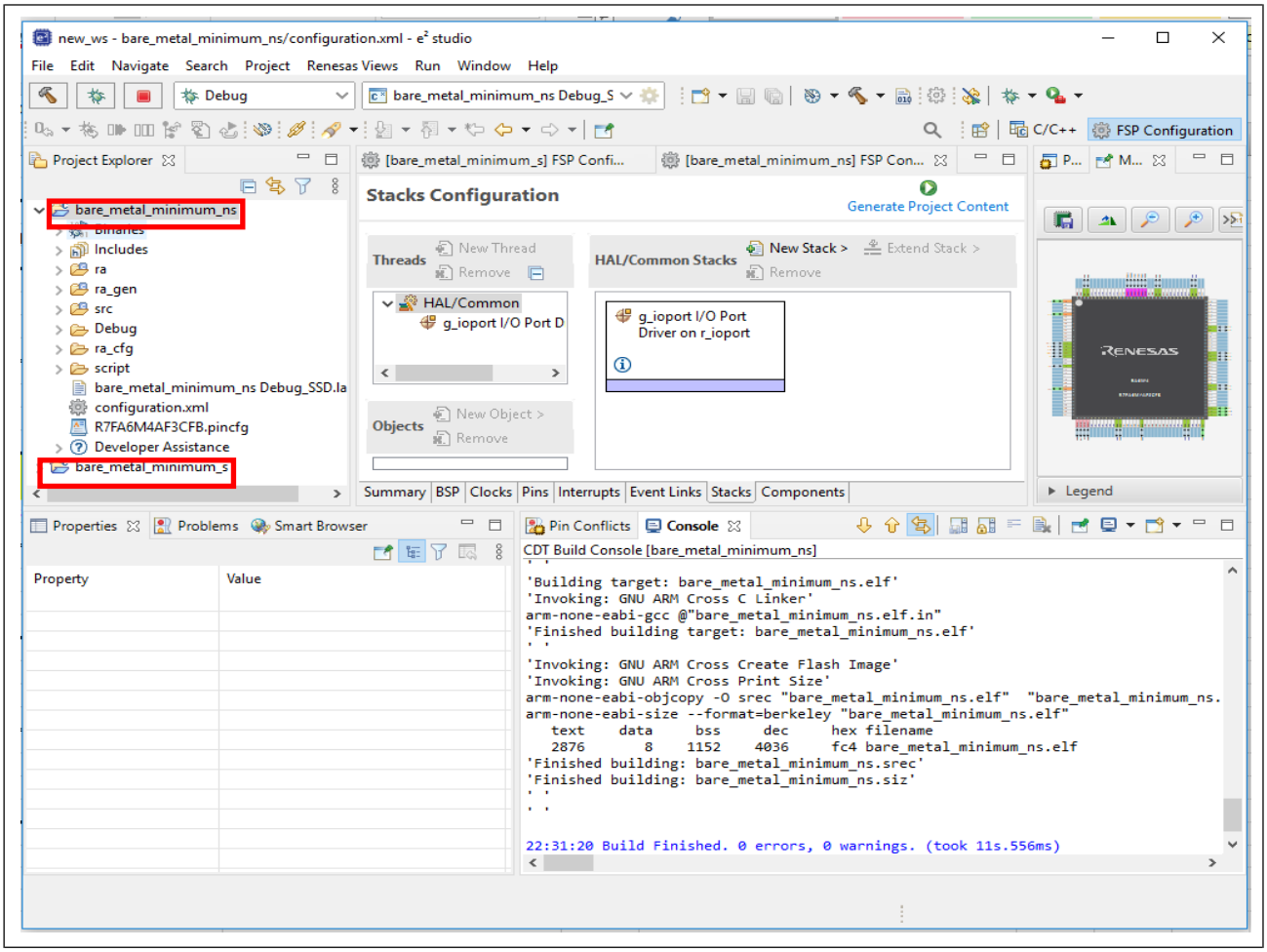

图 36. 编译非安全项目（无 RTOS，裸机最小化）

步骤 7：调试安全项目和非安全项目

如图 43 所示，非安全项目的调试配置默认将安全和非安全 `.elf` 文件写入到 MCU，以允许使用安全和非安全项目的统一调试会话。

注意，此时会生成 `<project_name> <build_configuration>_SSD.launch`，因为安全和非安全项目的调试均以SSD器件生命周期状态执行。

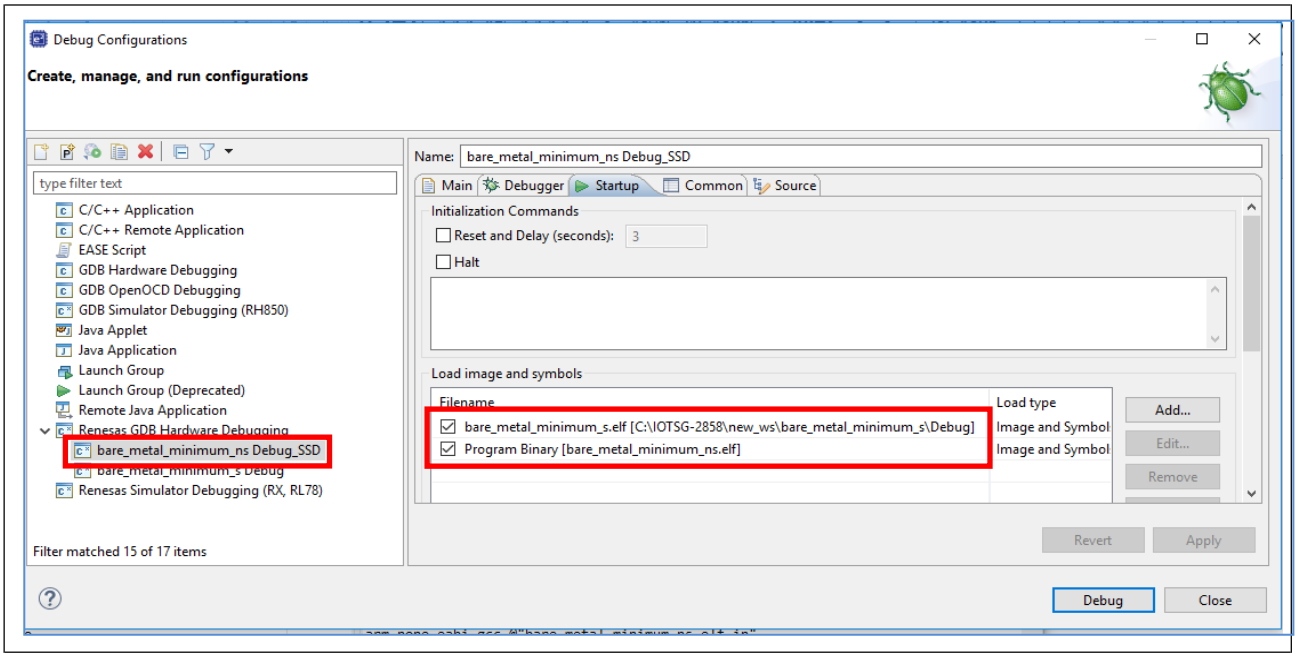

图 37. 调试安全项目和非安全项目

请注意，每次更改安全项目时都必须对其进行编译，以确保与非安全项目保持连接。当安全捆绑包更改时，将弹出一个窗口，要求用户获取最新的安全捆绑包。单击**“Yes”**（是），然后重新编译非安全项目，以便将使用更新后的 `<project_name>.sbd`。

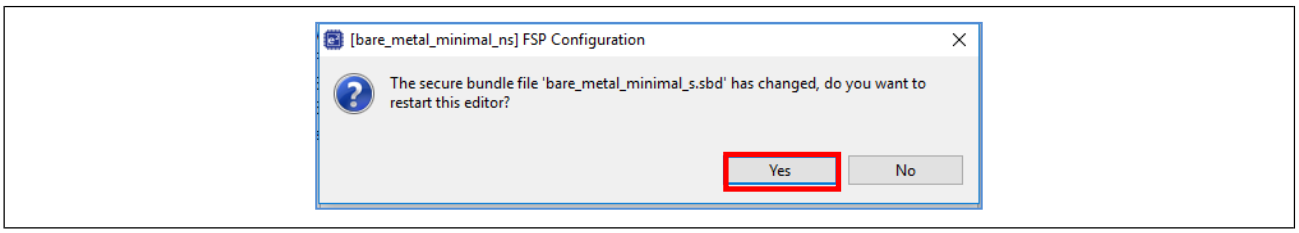

图 38. 安全捆绑包更新通知

确保安全项目和非安全项目之间同步的提示

为避免安全项目的意外更新被遗漏，用户还可以将安全项目定义为非安全项目的引用，这样在编译非安全项目时就会自动触发编译安全项目。

打开非安全项目的**“Properties”**（属性）页面，单击**“Project References”**（项目引用）并选择相应的安全项目作为引用项目。设置完成后，编译非安全项目时将始终触发重新编译安全项目。

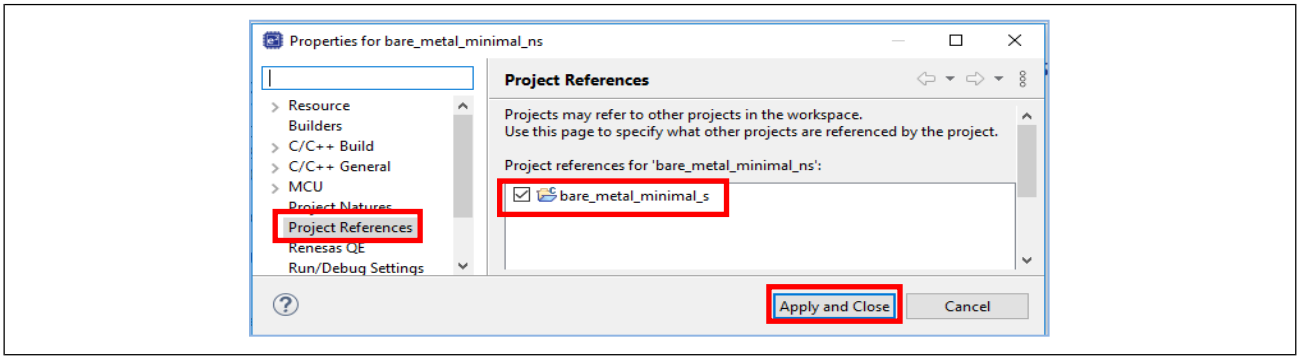

图 39. 创建项目引用

3.1.3 生产编程

此步骤用于生产流程；不是开发过程中需要的步骤。安全和非安全项目开发完成后，用户便可将以下信息发送到生产线，以便在销售前配置 MCU：

- 安全二进制文件
- 非安全二进制文件
- IDAU 区域配置

用户可参考附录 A 的第 6.2 节对安全二进制文件进行编程，参考第 6.3 节对非安全二进制文件进行编程，并将 MCU 状态切换到已部署状态。

3.2 分离式项目开发

以下是分离式项目开发模型的特点：

- 安全项目和非安全项目由两个不同的团队单独开发
- 安全项目将首先由 IP 提供商开发。IP 提供商创建一个安全捆绑包。
- 在非安全开发人员开始开发之前，安全捆绑包已预先烧录到 MCU 里。非安全开发人员只能看到非安全项目和非安全分区。

下一节将引导用户完成使用分离式项目开发模型的开发过程。

3.2.1 开发安全捆绑包并配置 MCU

使用分离式项目开发模型开发安全项目与使用联合式项目开发模型进行开发的过程非常相似。但是，本节将说明几个主要区别。

步骤 1：按照第 3.1.1 节的步骤 1 到步骤 6 建立安全模板项目并创建应用程序。

使用分离式项目开发模型调试安全项目时，不会与产品的非安全项目同时进行。如第 3.1.1 节的步骤 7 所述，用户可以创建一个虚拟非安全项目用于安全项目测试，例如测试非安全可调用 API。

步骤 2：为 MCU 配置安全项目并将器件生命周期状态切换到 NSECSD

分离式项目开发和联合式项目开发之间的主要区别在于，在分离式项目开发的非安全项目开发之前，需要为 MCU 配置与安全捆绑包相关联的安全二进制文件。安全捆绑包包含二进制格式的安全项目 IP 和安全项目的 NSC API 接口。此外，MCU 器件生命周期状态需要从 SSD 切换为 NSECSD 以保护安全内容。

3.2.2 在 NSECSD 状态下开发的限制和解决方法

当前版本的工具存在一个限制，即在使用分离式项目开发模型将 MCU 器件生命周期从 SSD 切换到 NSECSD 之前，除了安全二进制文件之外，还必须在器件上配置一个虚拟的非安全项目。这是允许在 NSEDSD 状态下恢复非安全开发所必需的。

- 在开发阶段，用户应遵循**联合式项目开发模型**准备一个与预期安全项目配对的虚拟非安全项目。首先对安全二进制文件和虚拟非安全二进制文件进行烧录，然后将器件生命周期状态切换到 NSECSD。
- 在生产阶段，用户应将以下两项发送给生产团队：
  - 安全二进制文件
  - IDAU 区域设置信息

  RFP 将用于对安全二进制文件进行编程并设置 IDAU 区域。有关操作细节，请参见附录 A 的第 6.2 节。
- 请注意，安全开发人员还需要向非安全开发人员提供安全捆绑包“<project_name>.sbd”，以允许非安全项目继续开发。
- 有关在 NSECSD 状态下支持非安全项目开发的一般流程的详细信息，请参见图 40。

3.2.3 在 NSECSD 状态下开发非安全项目

与联合式项目开发模型相比，使用分离式项目开发模型开发非安全项目有一些主要区别。

对于分离开发模型，非安全应用程序开发人员将接收处于 NSECSD 状态的 MCU。如上一节末尾中所述，需要特殊处理才能在 NSECSD 状态下进行开发。下面是在 NSECSD 状态下进行开发的一般流程汇总。

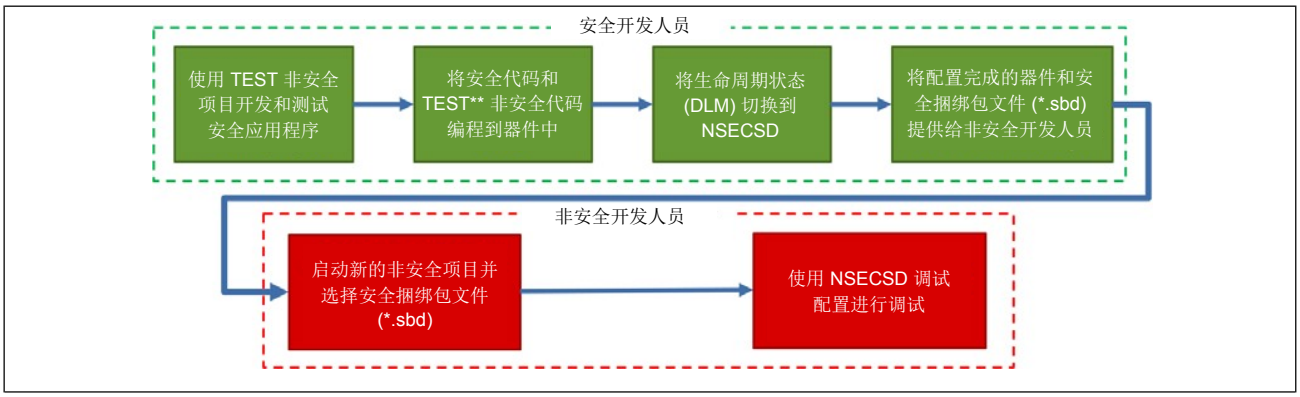

图 40. 在 NSECSD 状态下开发的开发流程

非安全开发人员在接收到已配置安全二进制文件、IDAU 区域和 NSECSD 状态非安全虚拟二进制文件的 MCU 后，便可使用以下步骤进行非安全项目开发。

1. **首先按照第 3.1.2 节中的步骤 1 和步骤 2 开始非安全项目开发。**
   通常，将在与安全项目不同的工作区中创建非安全项目，因为安全项目源文件和 `.elf` 文件不能用于非安全开发人员。
2. **当“Secure Bundle Selection”（安全捆绑包选择）窗口打开时，选择从安全开发人员处获得的安全捆绑包。**
   此步骤是联合式项目开发和分离式项目开发过程之间的主要区别。
   安全捆绑包包含以下信息以允许非安全项目开发：
   - MCU 启动代码
   - IDAU 区域设置
   - 已锁定安全外设配置设置的详细信息
   - 用户定义的非安全可调用 API 接口头文件（请参见第 2.4 节）

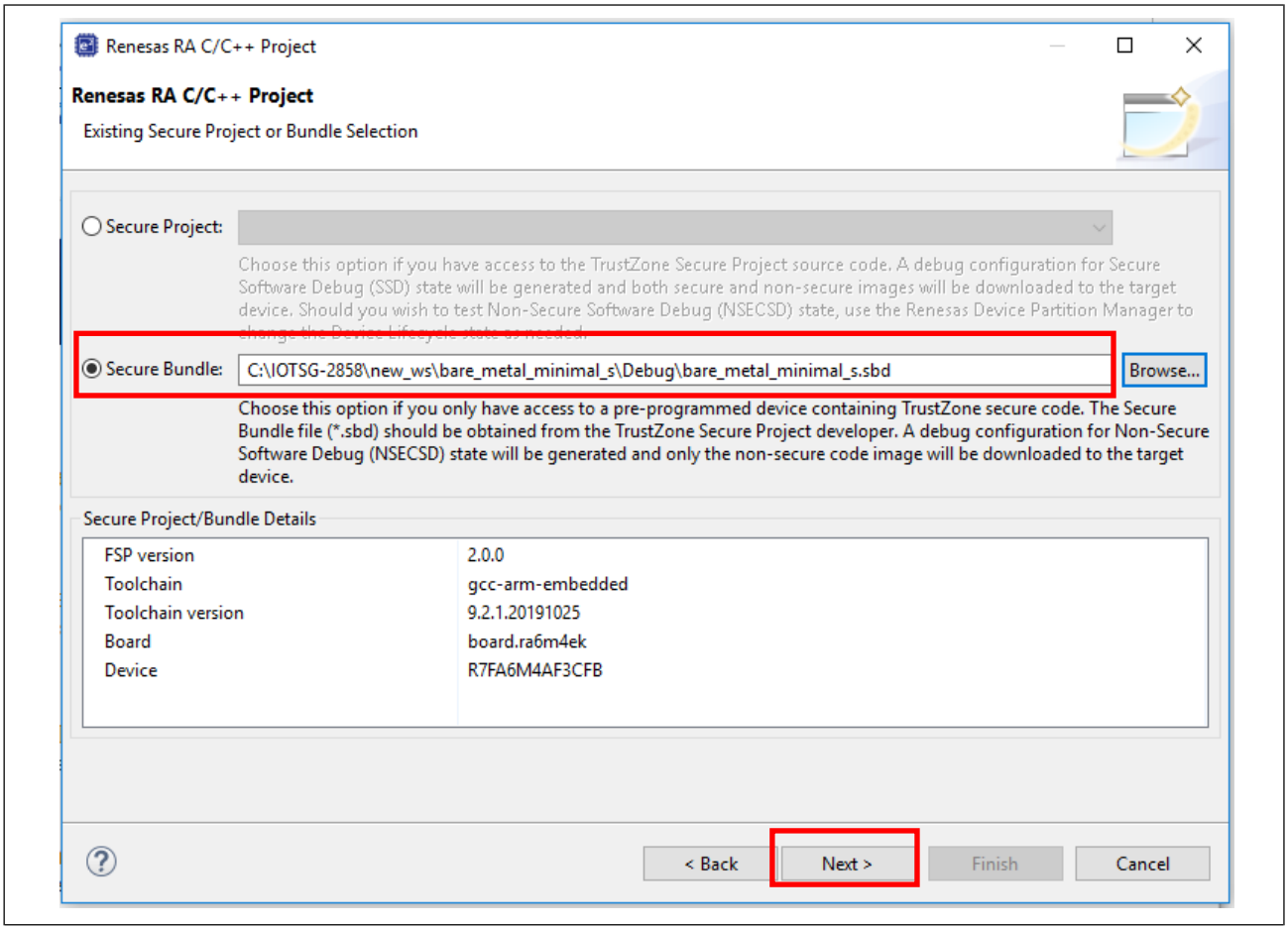

图 41. 创建到安全捆绑包的链接

请注意，安全捆绑包通过绝对路径链接。建议用户在“<project_name>.sbd”的文件夹位置变化时检查安全捆绑包的链接。

按照提示定义 RTOS 用法并选择模板项目。项目生成后，双击 configuration.xml 打开智能配置器。单击**“Generate Project Content”**（生成项目内容）并编译项目。

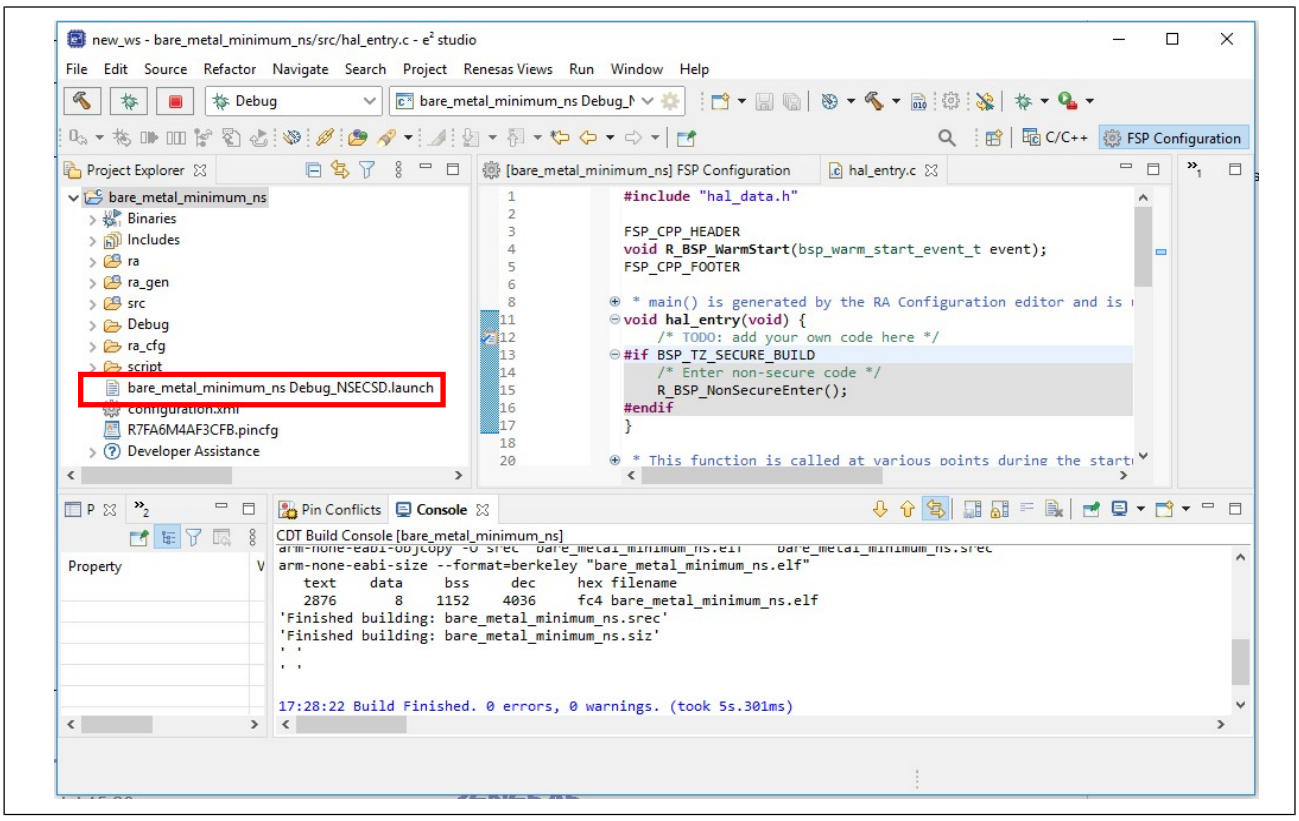

图 42. 非 RTOS 裸机最小化非安全项目模板的编译结果

请注意，在 NSECSD 状态下进行开发时将生成 `<project_name><build_configuration>_NSECSD.launch`。

3.2.3.1 调试非安全项目

在调试非安全项目之前，确保已在 MCU 上写入了安全二进制文件和虚拟非安全二进制文件。

在非安全项目调试期间，只会下载非安全 `.elf` 文件。对于非安全开发人员，在工作区中只有非安全项目可见，而采用联合式项目开发时，安全和非安全项目均可见。

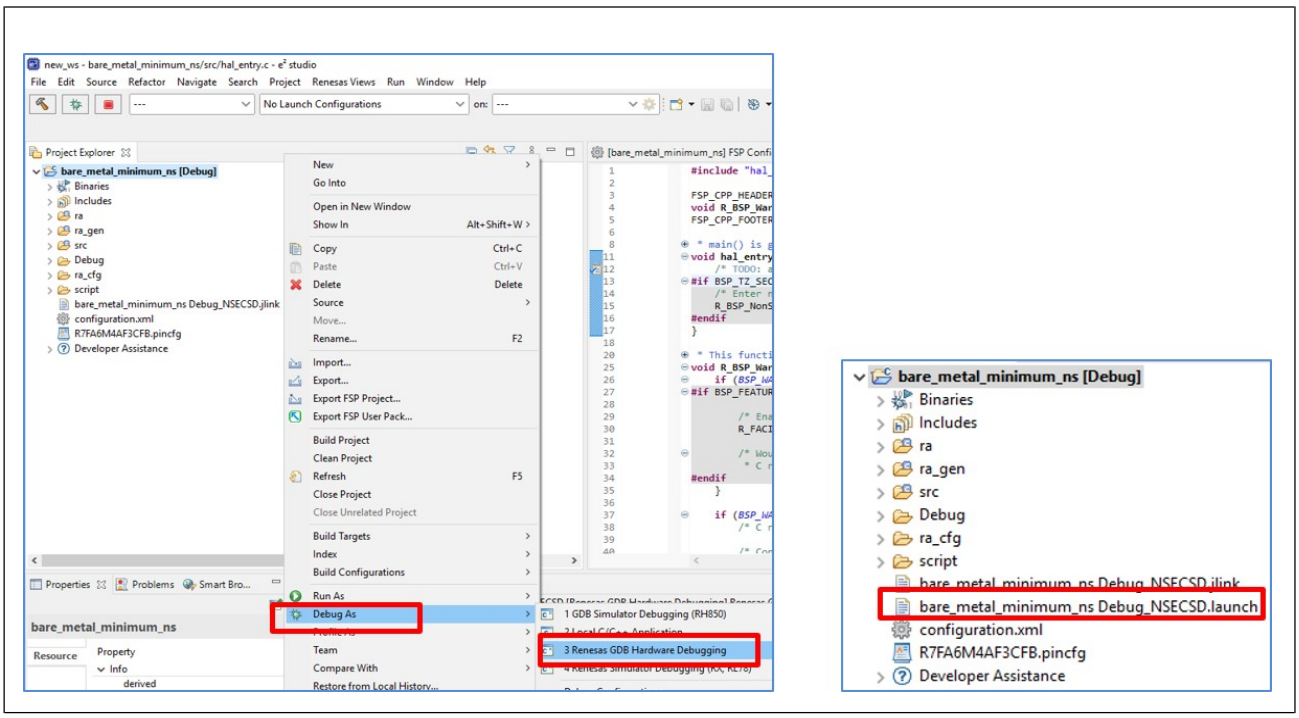

图 43. 调试非安全项目

更新安全捆绑包的注意事项

如果在非安全项目开发期间需要更新安全捆绑包，非安全开发人员需要将 MCU 返回给安全开发团队进行 MCU 更新。

请参见文档*《FSP 用户手册》的“入门：Arm® TrustZone® 项目开发”*部分的***非安全调试***了解在 NSECSD 器件生命周期状态下调试非安全项目时工具如何处理安全区域的保护。

3.2.3.2 烧录非安全项目并切换到 DPL 器件生命周期状态

此步骤适用于生产流程，在非安全项目开发期间通常不需要。

非安全项目经过完全调试后，非安全二进制文件可以发送到生产线以对 MCU 进行烧录并切换到 DPL 器件生命周期状态。有关操作细节，请参见附录 A“将 RFP 用于生产流程”中的第 6.3 节。

请参见应用笔记《*器件生命周期管理密钥安装*》了解其他可能的部署机制（LCK_DBG 和 LCK_BOOT）以及通过身份验证程序利用 DLM 密钥的状态回归方法。

4. 扁平化项目开发

RA 项目生成器中的扁平化项目类型是另一种开发模型，指在开发阶段，用户无需开发 带TrustZone 技术感知能力应用程序。

- 一个项目处理整个应用程序
- 开发流程与 Non-TrustZone 技术部分相同
- MCU 在 SSD 器件生命周期状态下运行
- 所有支持安全和非安全属性的外设都将在安全模式下运行
- 表 3 中标识为仅非安全的外设将在非安全模式下运行

4.1 扁平化项目开发

1. 按照第 3.2.1 节中的步骤 1 和步骤 2 开始创建扁平化项目模板项目。
2. 从项目生成器中选择**“Flat Project”**（扁平化项目）作为项目类型。请参见图 14（此处未复制图片）。
3. 在**“Build Artifact Selection”**（构建工件选择）和**“RTOS Selection”**（RTOS 选择）下做出选择。

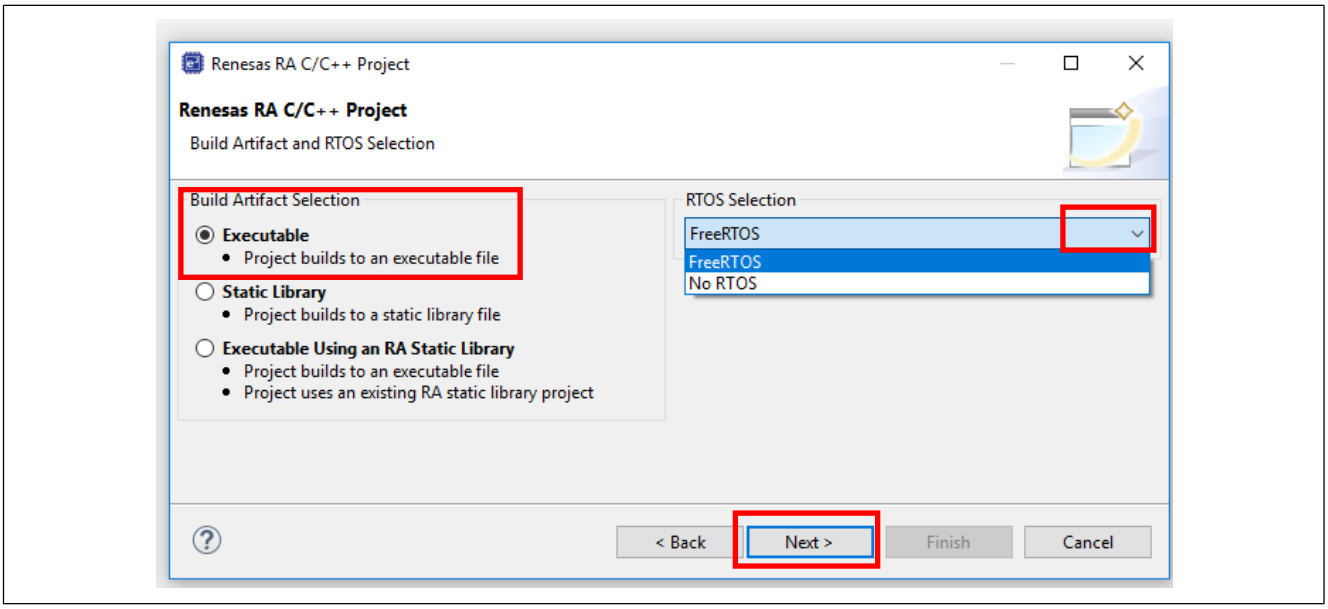

图 44. 选择构建工件和 RTOS 选项

其余的开发与支持 Non-TrustZone 技术的 MCU 的开发相同，超出了本应用项目的范围。

4. 调试扁平化项目

调试扁平化项目遵循 Non-TrustZone RA MCU 调试模型。启动文件名为：

`<program_name> <build_configuration>_Flat.launch`。

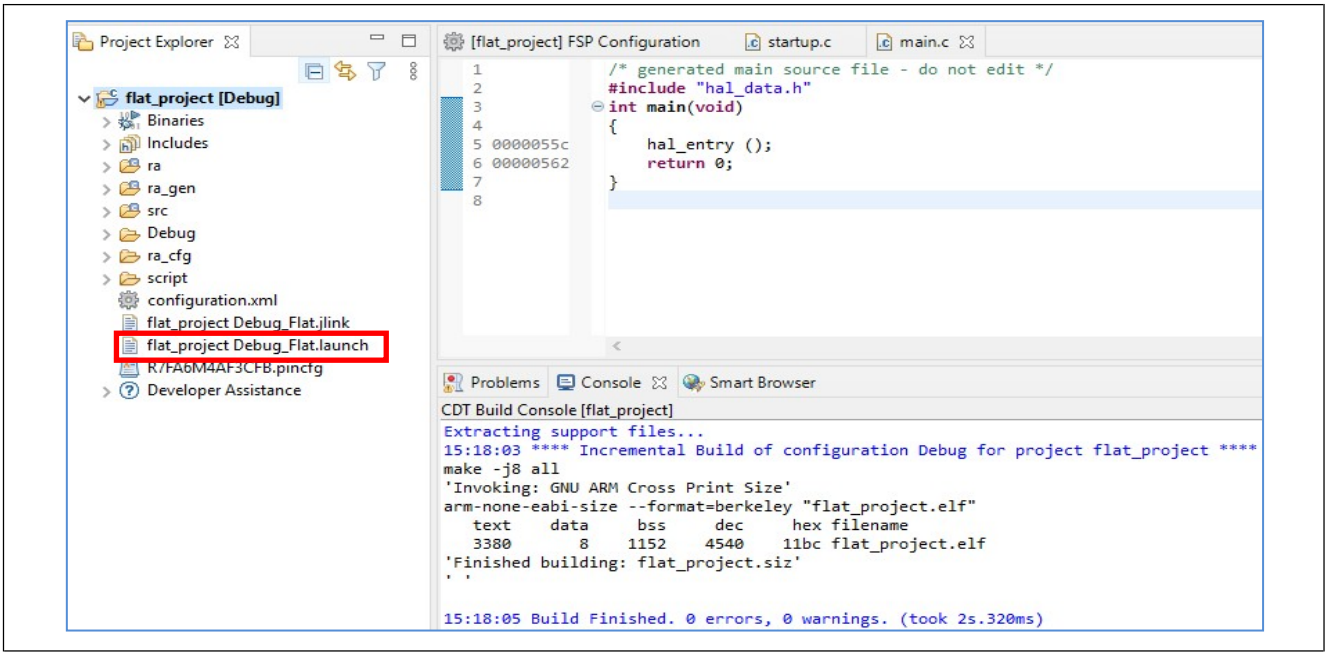

图 45. 选择构建工件和 RTOS 选项

以太网处理的特别注意事项

在以太网与扁平化项目一起使用的情况下，会将 32 KB 区域定义为非安全区域以支持以太网的 RAM 缓冲区应用。此过程由 IDE 和 FSP 自动处理，因此用户在开发过程中无需了解细节。

4.2 生产流程

扁平化项目开发模型的生产流程将引入 TrustZone 技术感知。扁平化项目开发是在 MCU 生命周期状态 SSD 下进行的。对于生产部署，用户具有与 TrustZone 技术感知开发模型相同的选项：分离开发模型或联合开发模型。

- 选项 1 是将 MCU 生命周期状态从 SSD 切换到 NSECSD，然后再切换到 DPL
  - 如果需要，MCU 生命周期状态可以进一步切换到 LCK_DBG 或 LCK_BOOT
- 选项 2 是将 MCU 状态从 SSD 直接切换到 LCK_DBG 或 LCK_BOOT

有关生产流程选项的详细信息，请参见应用笔记*《器件生命周期管理密钥安装》*。

5. 使用 e² studio 的 IP 保护示例项目

正如第 1.4.1 节中所述，IP 保护是 TrustZone 技术的强大用例。本文档随附的项目利用分离式项目开发模型提供使用 EK-RA6M4 的 IP 保护示例 TrustZone 用例。

5.1 概述

RA6M4 MCU 可以配置为使用 ADC 来监视片上温度传感器。此应用项目定义了一种算法，可根据从 ADC 读取的温度来控制 LED 闪烁模式。以下硬件组件由安全项目配置为安全：

- 用于片上温度传感器读数的 ADC 通道
- GPIO 400、404 和 415
- 通过 IDAU 设置的安全闪存和 SRAM

以下软件组件由安全项目配置为安全：

- FSP ADC HAL 驱动程序
- 相应 LED 驱动引脚的 FSP GPIO HAL 驱动程序
- 启动、扫描和停止 ADC 的应用程序代码
- 根据温度读数控制 LED 闪烁模式的应用程序代码
- 启动监视和反应算法的 API
  - 此 API 被定义为非安全可调用 API，其跳板暴露给非安全分区
- 停止监视和反应算法的 API
  - 此 API 被定义为非安全可调用 API，其跳板暴露给非安全分区

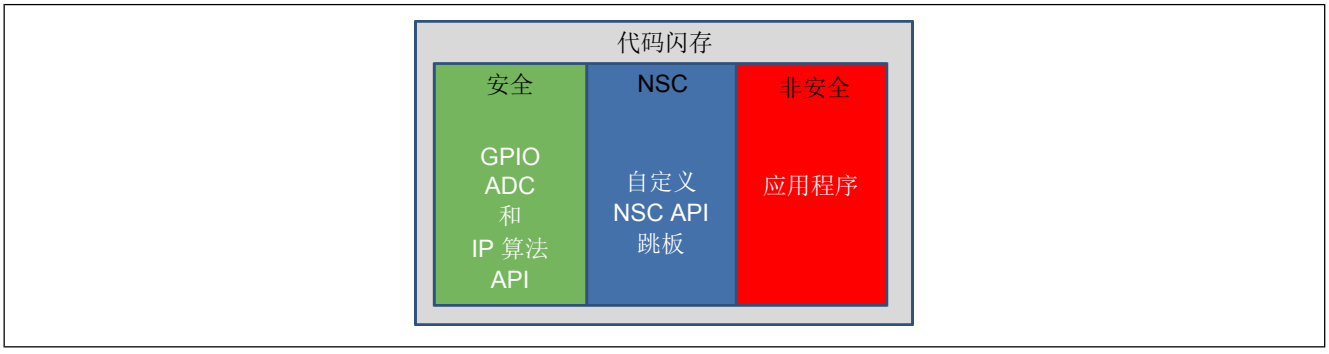

图 46. 传感器算法 IP 保护

5.2 系统架构

5.2.1 软件组件

图 47 所示为此示例项目中的安全、非安全和非安全可调用硬件和软件分区方案。

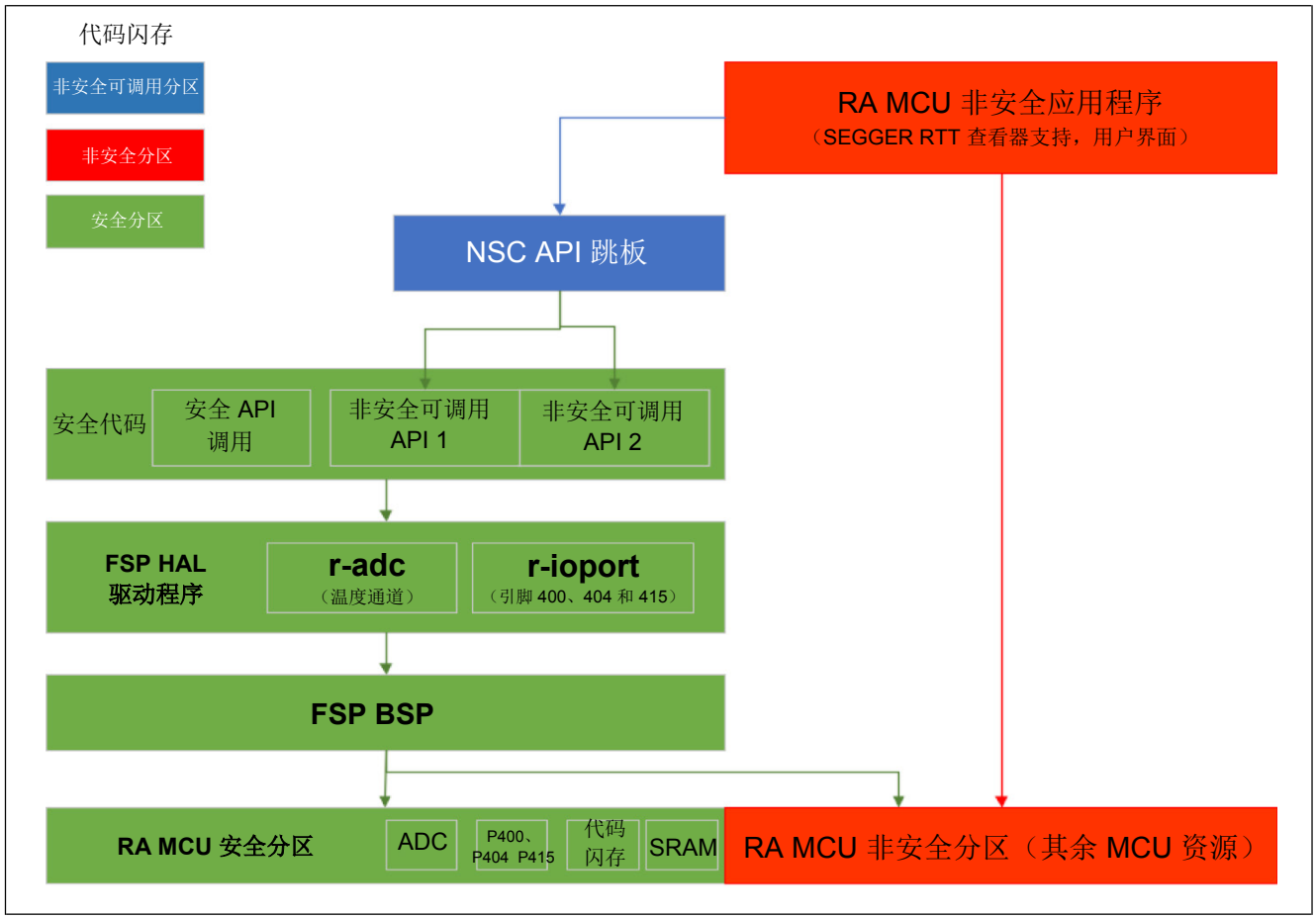

图 47. 软件架构框图

5.2.2 操作流程

图 48 所示为示例项目的系统级操作流程。

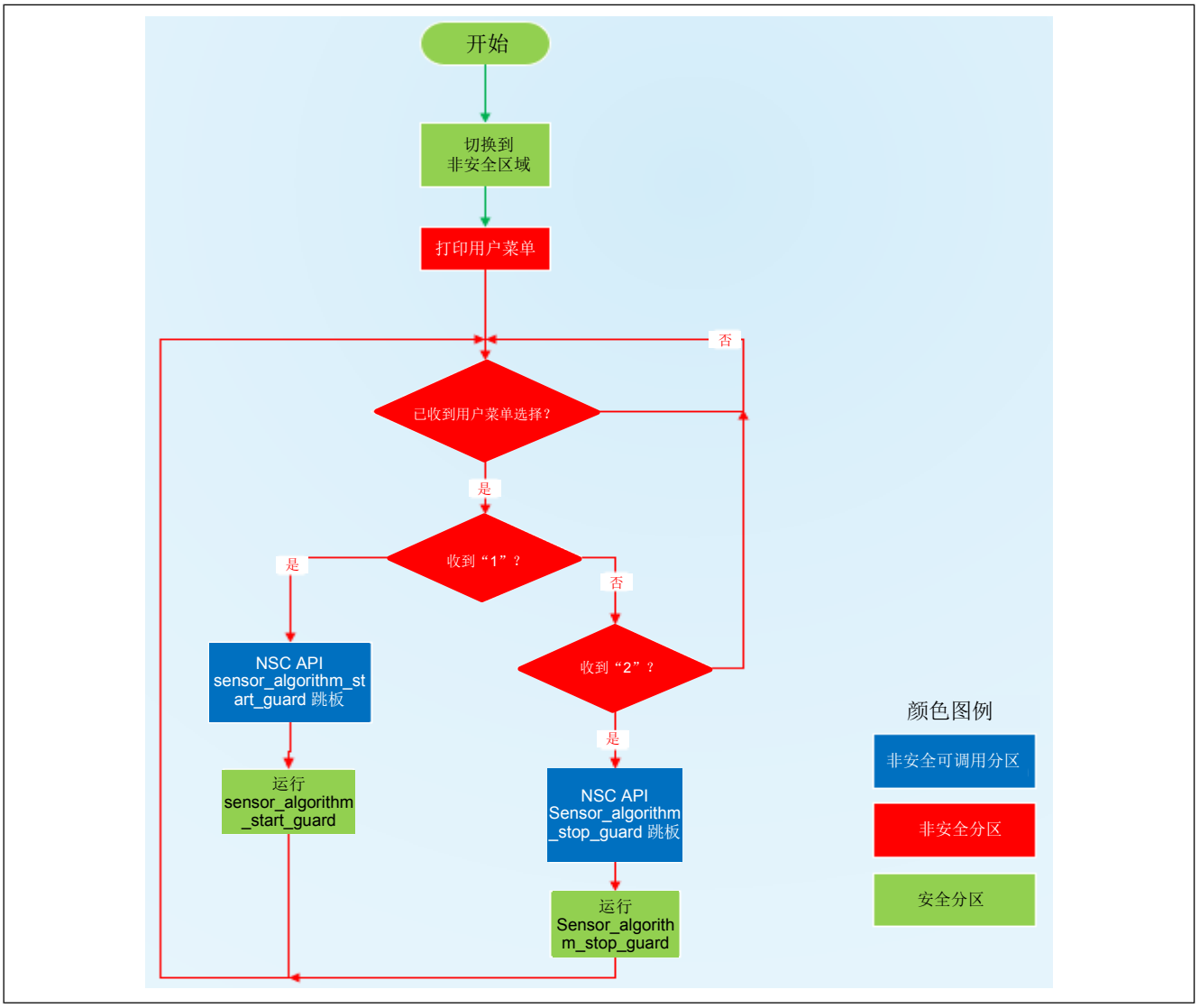

图 48. 操作流程

5.2.3 模拟“用户 IP 算法”

模拟**用户 IP 算法**如下面的图 49 所示。

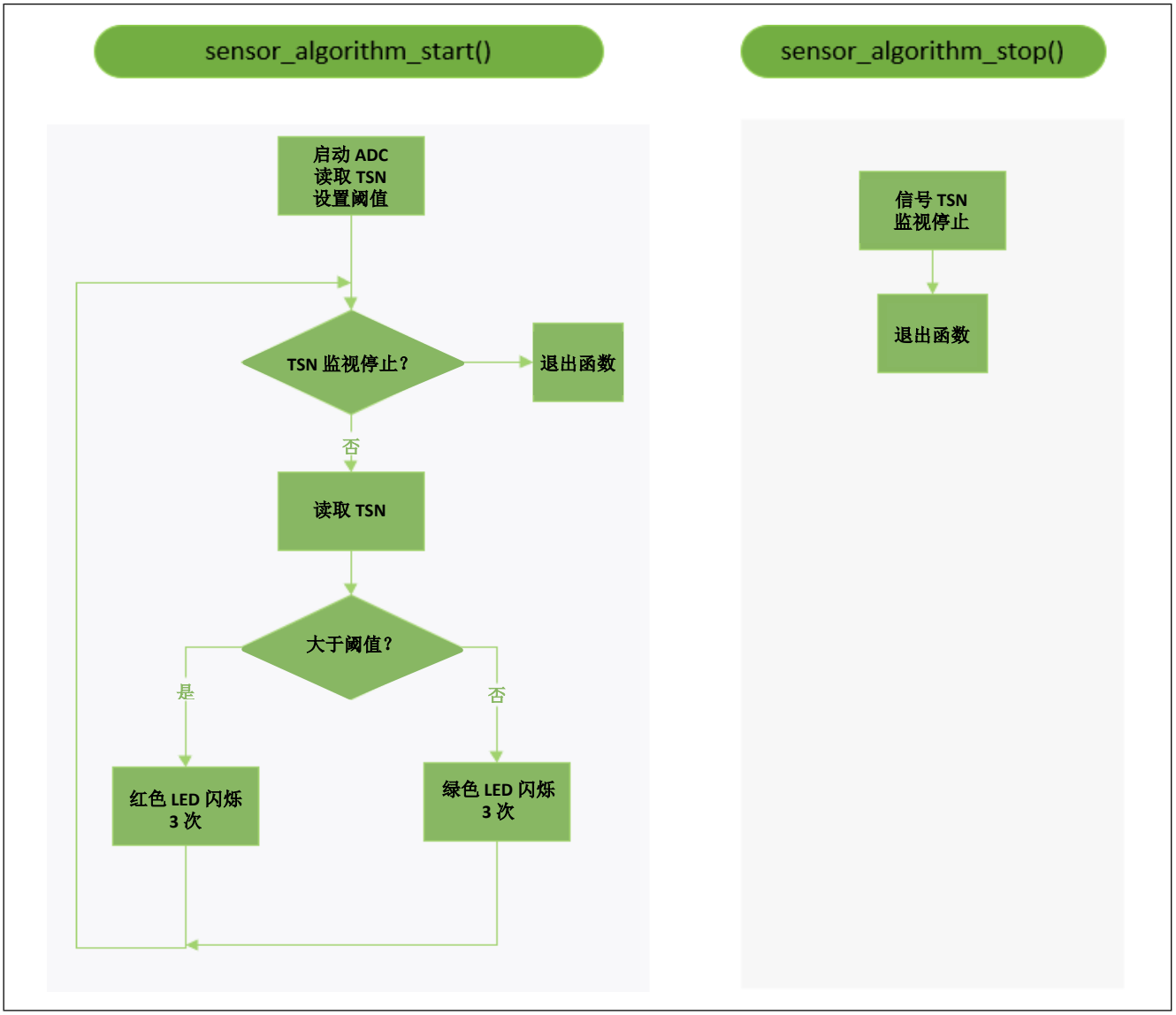

图 49. 模拟传感器 IP 算法（在安全分区中运行）

5.2.4 用户定义的非安全可调用 API

暴露给非安全分区的**非安全可调用**函数在安全项目的 `sensor_algorithm_nsc.h` 中定义。

```
/* File Name    : sensor_algorithm_nsc.h
 * DISCLAIMER
#ifndef SENSOR_ALGORITHM_NSC_H_
#define SENSOR_ALGORITHM_NSC_H_

#include <hal_data.h>

BSP_CMSE_NONSECURE_ENTRY void sensor_algorithm_start_guard(void);
BSP_CMSE_NONSECURE_ENTRY void sensor_algorithm_stop_guard(void);

#endif /* SENSOR_ALGORITHM_NSC_H_ */
```

图 50. 用户定义的 NSC API

此头文件的路径通过如下所示的构建变量“UserNscApiFiles”添加。

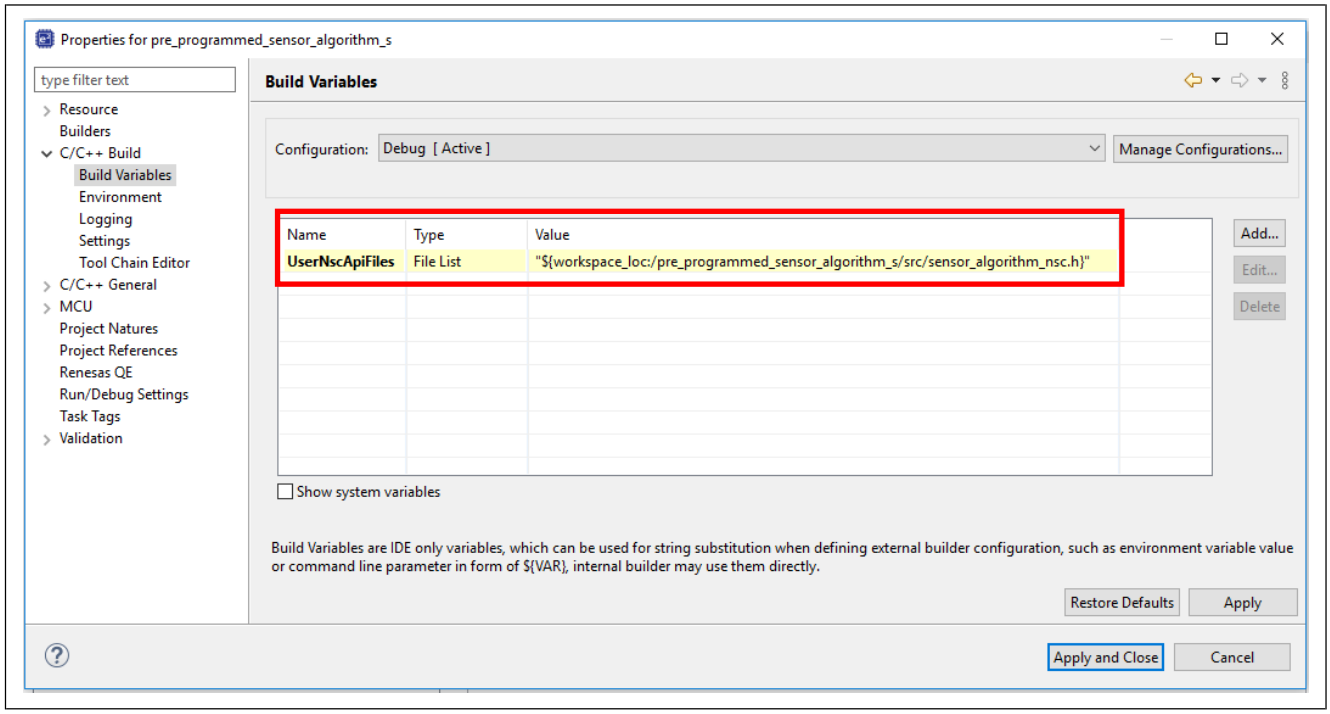

图 51. 用于链接用户 NSC 头文件的用户构建变量（安全项目设置）

5.3 运行示例应用程序

5.3.1 设置硬件

- 跳线设置 – 默认 EK-RA6M4 设置
  - 请参见*《EK-RA6M4 用户手册》*。
- 使用 USB 线缆将 EK-RA6M4 J10连接到开发 PC，使用板上调试器提供供电和调试功能。

5.3.2 导入、编译和写入安全二进制文件和虚拟非安全二进制文件

用户可以使用以下步骤为 MCU 配置安全二进制文件和虚拟非安全二进制文件。

1. 导入安全项目和虚拟非安全项目

解压包含在此应用项目中的 security_design_with_trustzone_ip_protection.zip 以显示以下文件夹：

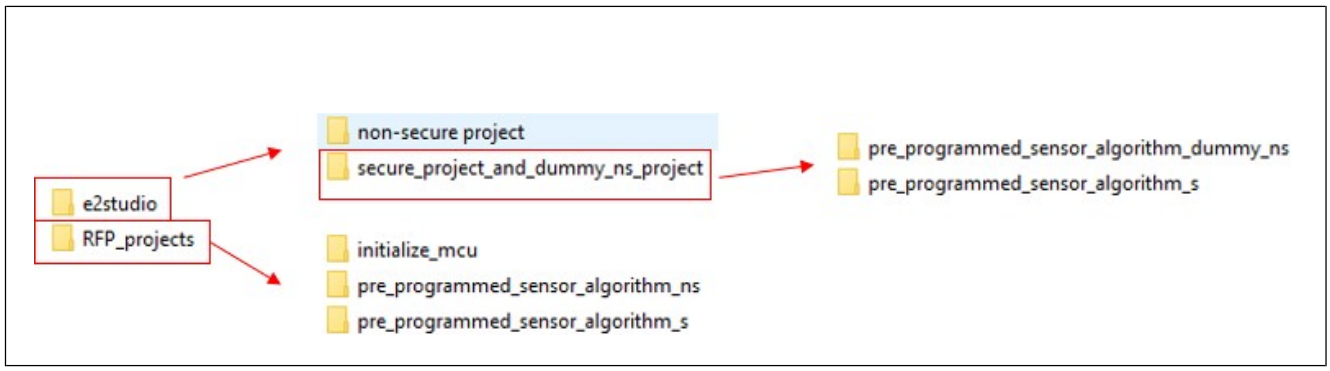

图 52. 软件项目内容

接下来，按照*《FSP 用户手册》*的“*将现有项目导入 e2 studio*”部分将安全项目和虚拟非安全项目导入到同一工作区中。

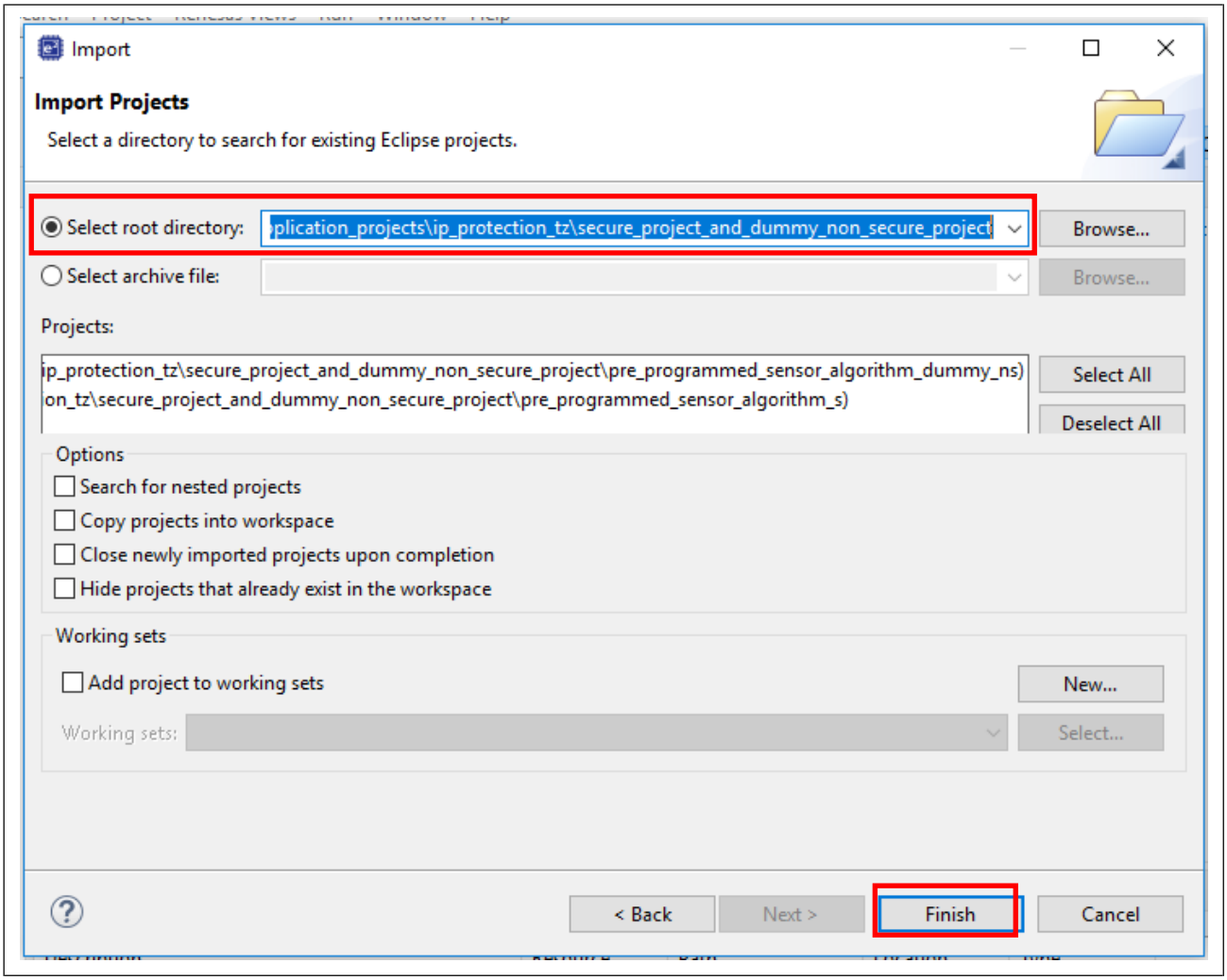

图 53. 导入安全项目和虚拟非安全项目

单击**“Finish”**（完成）。

5.3.2.1 使用 e² studio 编译安全二进制文件和虚拟非安全二进制文件

- 首先编译安全项目。双击打开安全项目中的 `Configuration.xml`。单击**“Generate Project Content”**（生成项目内容）。编译安全项目。
- 接下来，编译虚拟非安全项目。双击打开虚拟非安全项目中的 `Configuration.xml`。单击**“Generate Project Content”**（生成项目内容）。编译非安全项目。

5.3.2.2 初始化 MCU

此步骤是可选的，但建议执行此操作。在下载示例应用程序之前，建议用户将器件初始化为 SSD 状态。在此过程中，未锁定的闪存内容将被擦除。可以使用瑞萨器件分区管理器或 RFP 来实现此步骤。如果器件之前在 NSECSD 状态下使用或某个闪存块被临时锁定，这一步将尤其有用。

有关如何使用 RFP 执行此功能的说明，请参见附录 A 的第 6.1 节。使用瑞萨器件分区管理器和 J-Link 调试器初始化 MCU

在使用瑞萨器件分区管理器和板载 J-Link 调试器执行**“initialize device back to factory default”**（将器件初始化为出厂默认值）之前，建立以下连接。请注意，使用 RFP 时，**“initialize device back to factory default”**（将器件初始化为出厂默认值）执行与**“Initialize Device”**（初始化器件）相同的功能。

- EK-RA6M4 跳线设置：J6 闭合，J9 断开。其他跳线保持默认的设置。
- J10 和开发 PC 之间通过USB 线缆连接

请注意，如果使用 J-Link 作为连接接口，则在调试会话后，用户需要对电路板执行重新上电操作，才能使用瑞萨器件分区管理器。

打开瑞萨器件分区管理器

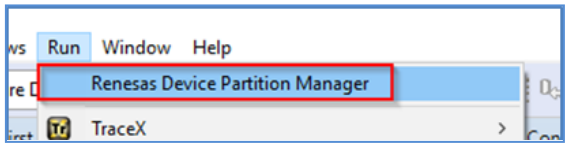

图 54. 打开“Renesas Device Partition Manager”（瑞萨器件分区管理器）

接下来，选中**“initialize device back to factory default”**（将器件初始化为出厂默认值），选择连接方式，然后单击**“Run”**（运行）。

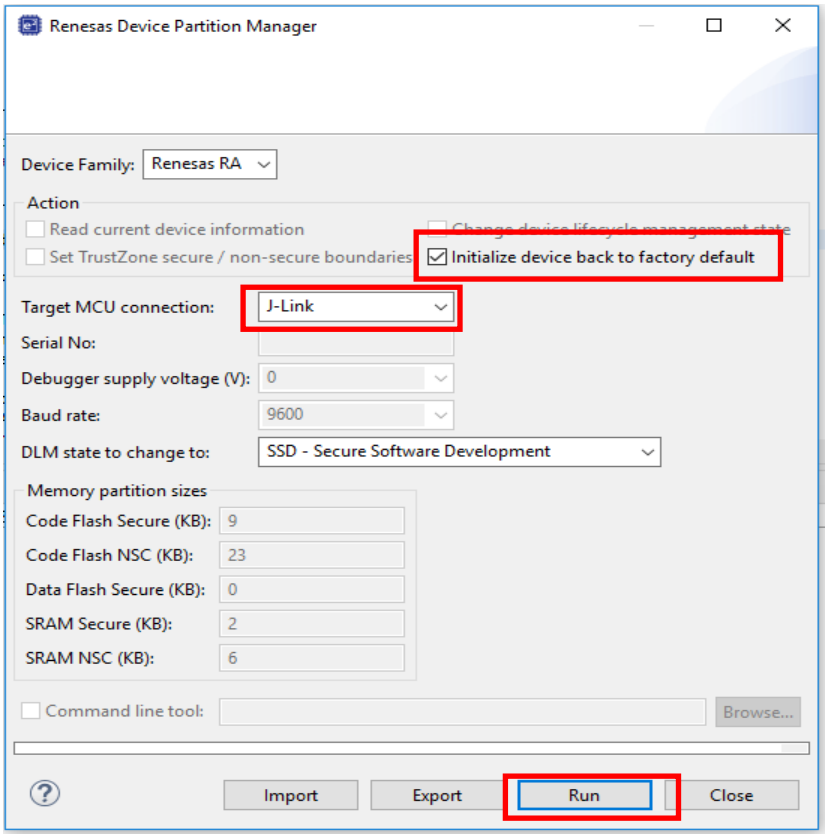

图 55. 使用“Renesas Device Partition Manager”（瑞萨器件分区管理器）初始化 RA6M4

MCU 初始化后，继续烧录安全二进制文件和虚拟非安全二进制文件。

5.3.2.3 使用 e2 studio 下载安全二进制文件和虚拟非安全二进制文件

右键单击 pre_programmed_sensor_algorithm_dummy_ns 项目并选择 **“Debug As > Renesas GDB Hardware Debug”**（调试方式 > 瑞萨 GDB 硬件调试）。

5.3.2.4 将 MCU 器件生命周期状态切换到 NSECSD

将安全二进制文件和虚拟非安全二进制文件下载到 MCU 后，用户可以使用 **“Renesas Device Partition Manager”**（瑞萨器件分区管理器）将 MCU 从 SSD 器件生命周期切换到 NSECSD 器件生命周期。

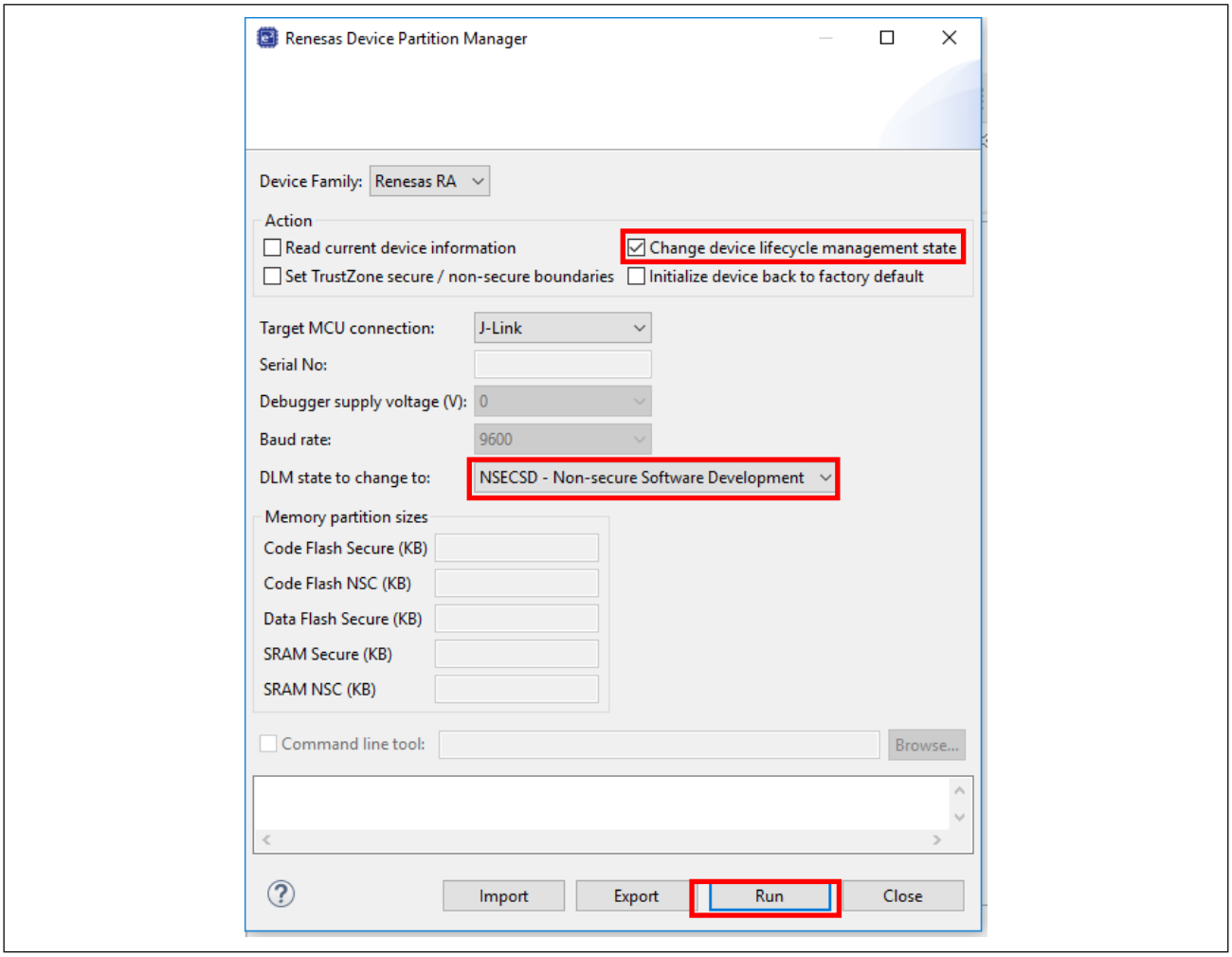

图 56. 使用“Renesas Device Partition Manager”（瑞萨器件分区管理器）从 SSD 切换到 NSECSD

MCU 现在已准备好使用真正的非安全二进制文件进行烧录。

用户可参见附录 A 的第 6.1 节和第 6.2 节了解下载安全二进制文件以及在生产阶段使用 RFP 设置 IDAU 区域的操作步骤。

5.3.3 导入、编译和烧录非安全项目

5.3.3.1 导入非安全项目

按照*《FSP 用户手册》*的“将现有项目导入 e2 studio”部分将非安全项目导入到工作区。可以导入到已导入安全项目的工作区中以验证示例项目。

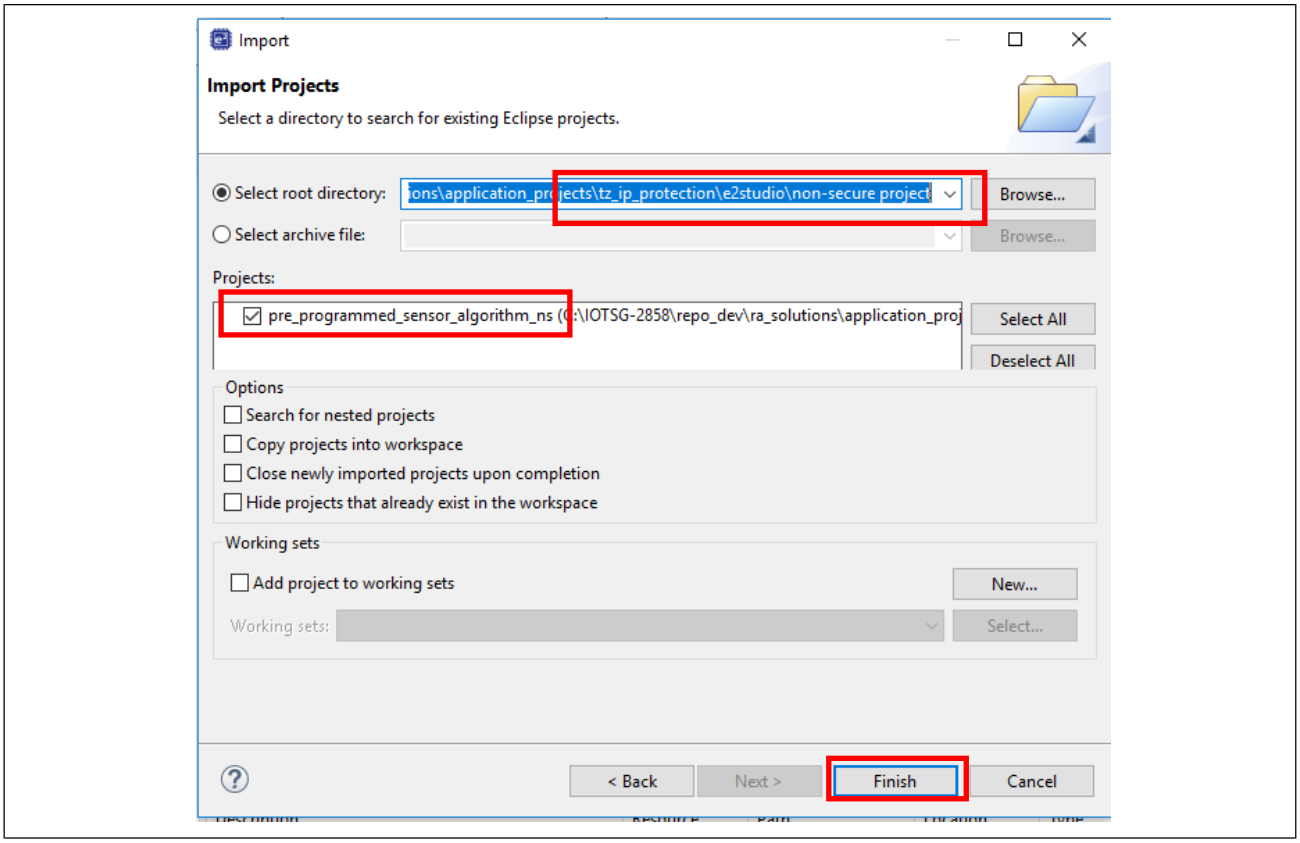

图 57. 导入非安全项目

请注意，用户需要更新构建变量 **SecureBundle**，方法是在进行其他步骤之前根据您的本地文件结构选择 `pre_programmed_sensor_algorithm_s.sbd`。这是当前工具的限制。

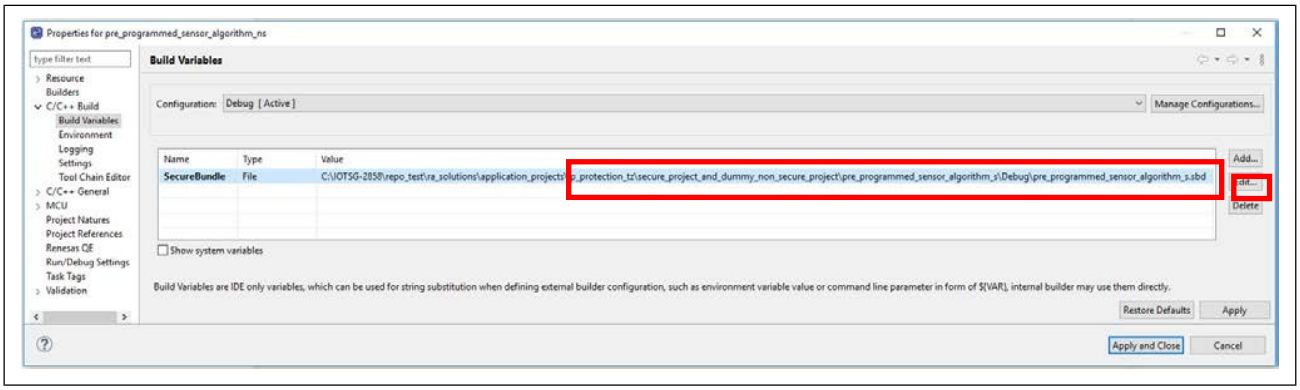

图 58. 引用安全捆绑包

设置完 **SecureBundle** 值后，继续编译非安全项目。

5.3.3.2 编译并下载非安全项目

- 双击打开非安全项目中的 `configuration.xml`。单击“**Generate Project Content**”（生成项目内容）。编译非安全项目。
- 下载并运行非安全项目。

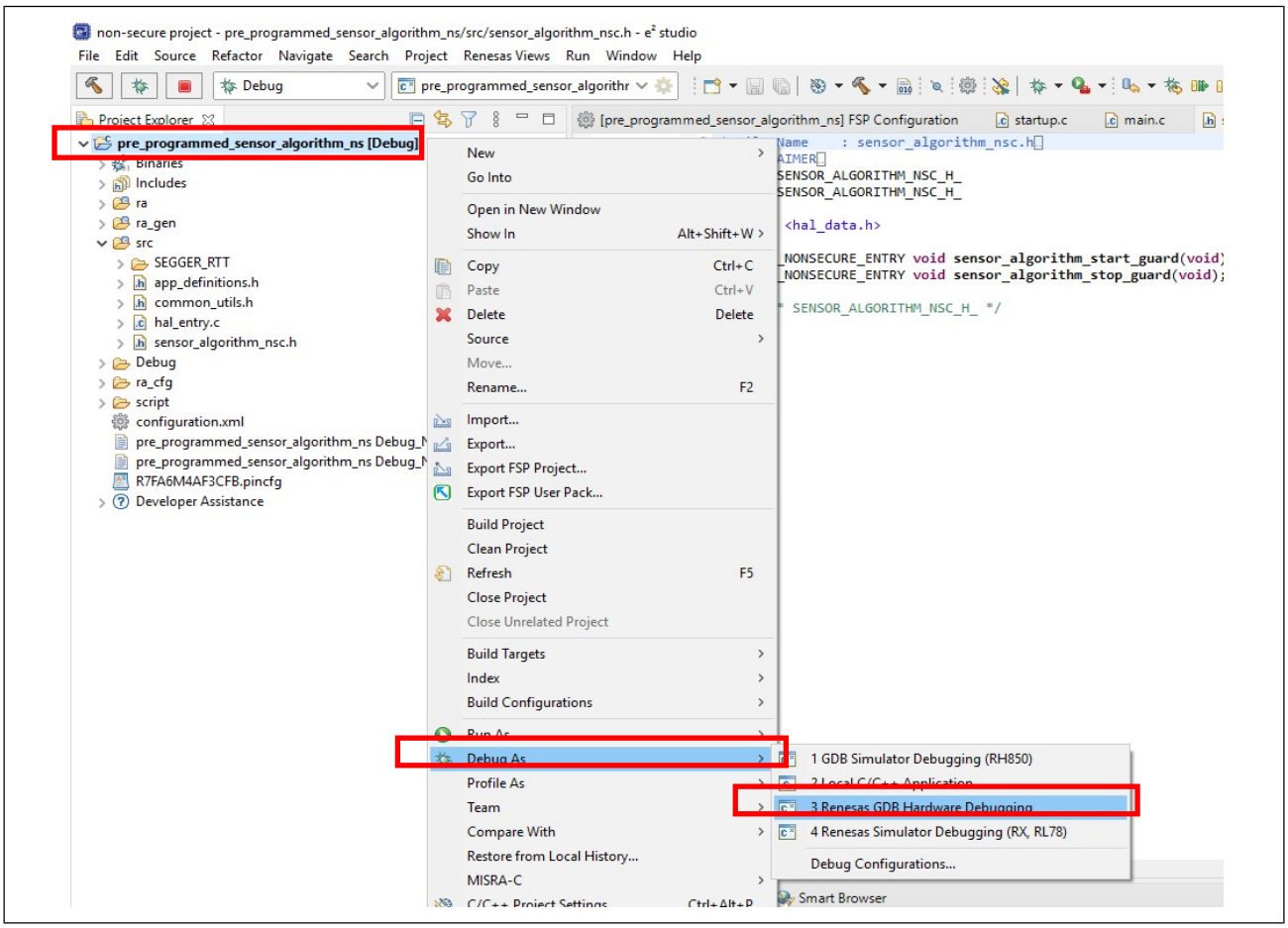

图 59. 下载并运行非安全项目

请注意，对于项目分离开发模型，通过引用安全捆绑包而不是安全项目（与虚拟非安全项目的情况一样）创建的非安全项目，调试会话仅下载非安全项目的 `.elf` 文件。

5.3.4 验证示例应用程序

项目现已加载，调试器应在非安全项目的 `Reset_Handler()` 中的 `SystemInit()` 调用处暂停。

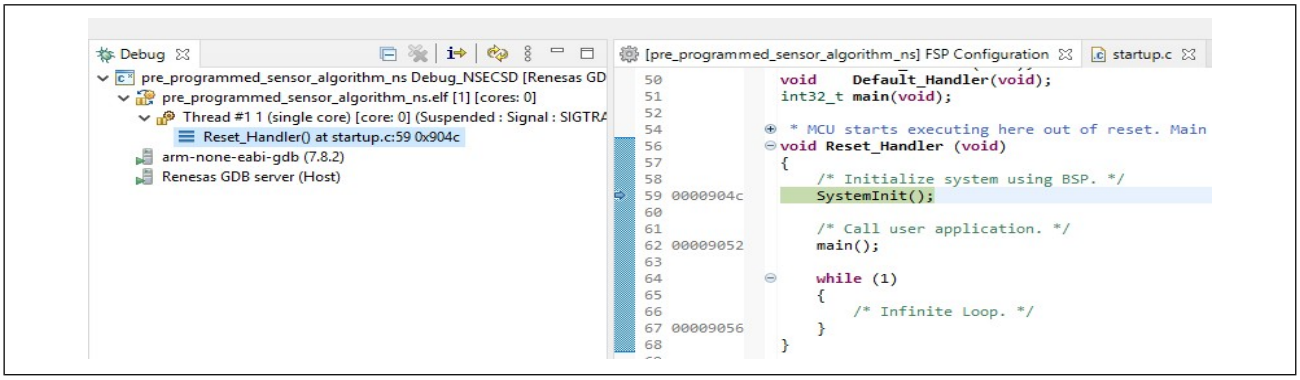

图 60. 运行非安全项目

打开 J-Link RTT 查看器 6.82d 或更高版本。首先单击“...”，选择瑞萨的 **R7FA6M4AF** 作为目标器件。接下来，将到 J-Link 的连接设置为 **“Existing Session”**（现有会话），将 **“RTT Control Block”**（RTT 控制模块）设置为 **“Search Range”**（搜索范围）。将搜索范围设置为 0x20000000 0x8000，然后单击 **“OK”**（确定），启动 RTT 查看器。

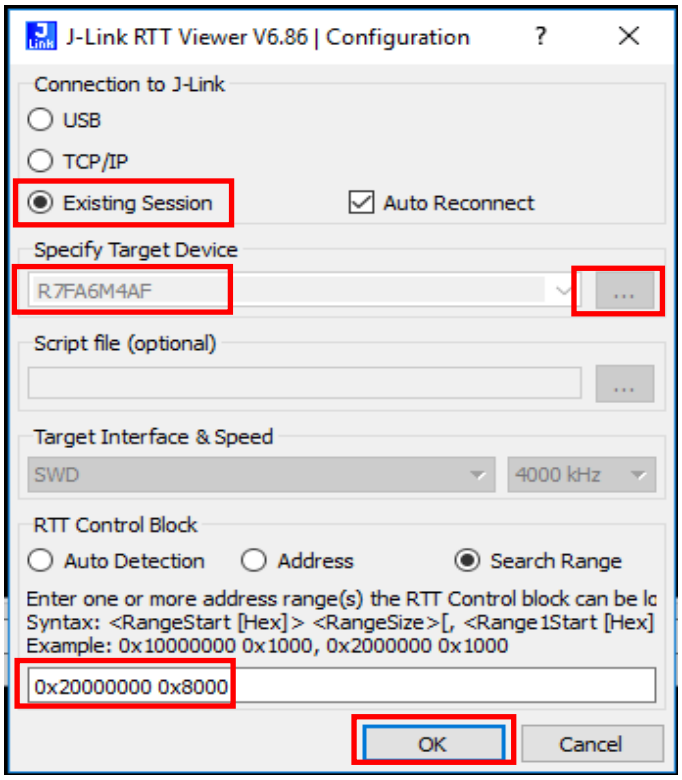

图 61. 启动 RTT 查看器

下一步，单击  两次，运行项目。

随后输出用户菜单，系统等待用户输入。

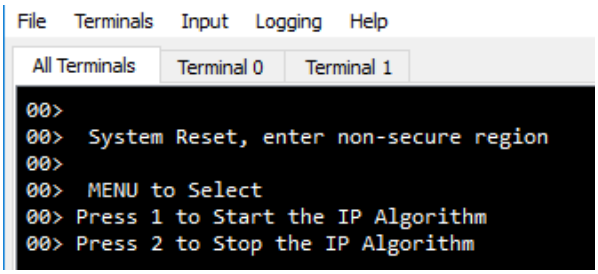

图 62. 用户菜单

输入 **1** 可启动 IP 算法。几秒钟后，用户将看到绿色 LED 开始闪烁。

用户可以 使MCU升温（例如，使用多个手指触摸 MCU），将看到绿色 LED 停止闪烁，大约 5-10 秒后红色 LED 开始闪烁。

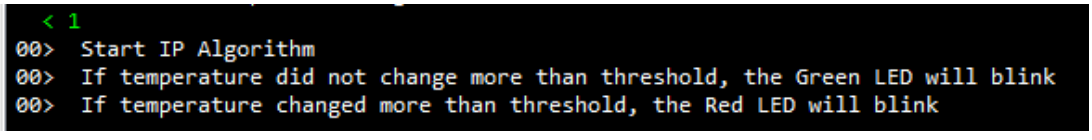

图 63. 用户输入 “1”

输入 **2** 可停止 IP 算法。绿色或红色 LED 将停止闪烁。蓝色 LED 将闪烁两次，之后也停止闪烁。

```
  < 2
00>  Stop_IP_Algorithm
00>  Blue LED will toggle twice
00>  Press 1 to restart the IP Algorithm
```

图 64. 用户输入“2”

用户可以重复输入 **1** 重新启动 IP 算法，输入 **2** 停止 IP 算法。

在独立模式下运行应用程序的注意事项

对 MCU 烧录应用程序代码后，用户可以在独立模式下运行应用程序（没有调试会话）。在这种情况下，将**“Connection to J-Link”**（到 J-Link 的连接）选择为 **USB**。

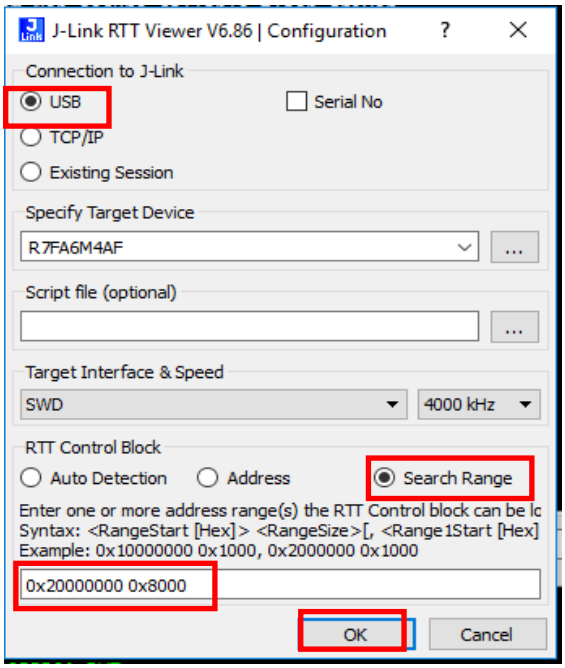

图 65. MCU 在独立模式下运行时的 Segger RTT 查看器连接设置

使用**瑞萨器件分区管理器**将 MCU 器件生命周期状态切换到 DPL 的注意事项

如前所述，DPL 状态下将禁止调试器连接，从而可保护安全和非安全代码和数据。一旦 MCU 切换到 DPL 状态，RTT 查看器输入/输出将不再可用。

RTT 查看器是用于调试应用程序或提供操作演示的便捷工具。在实际的客户应用中，在非安全项目中完成调试后，用户将移除 RTT 查看器的调试输入/输出并通过内部逻辑访问非安全可调用区域。对于实际应用中的状态或错误记录，UART 可以作为替代选项。

当原型设计将与客户共享而不使用 Segger RTT 查看器功能时，用户可以使用以下配置将 MCU 器件生命周期状态切换到 DPL。

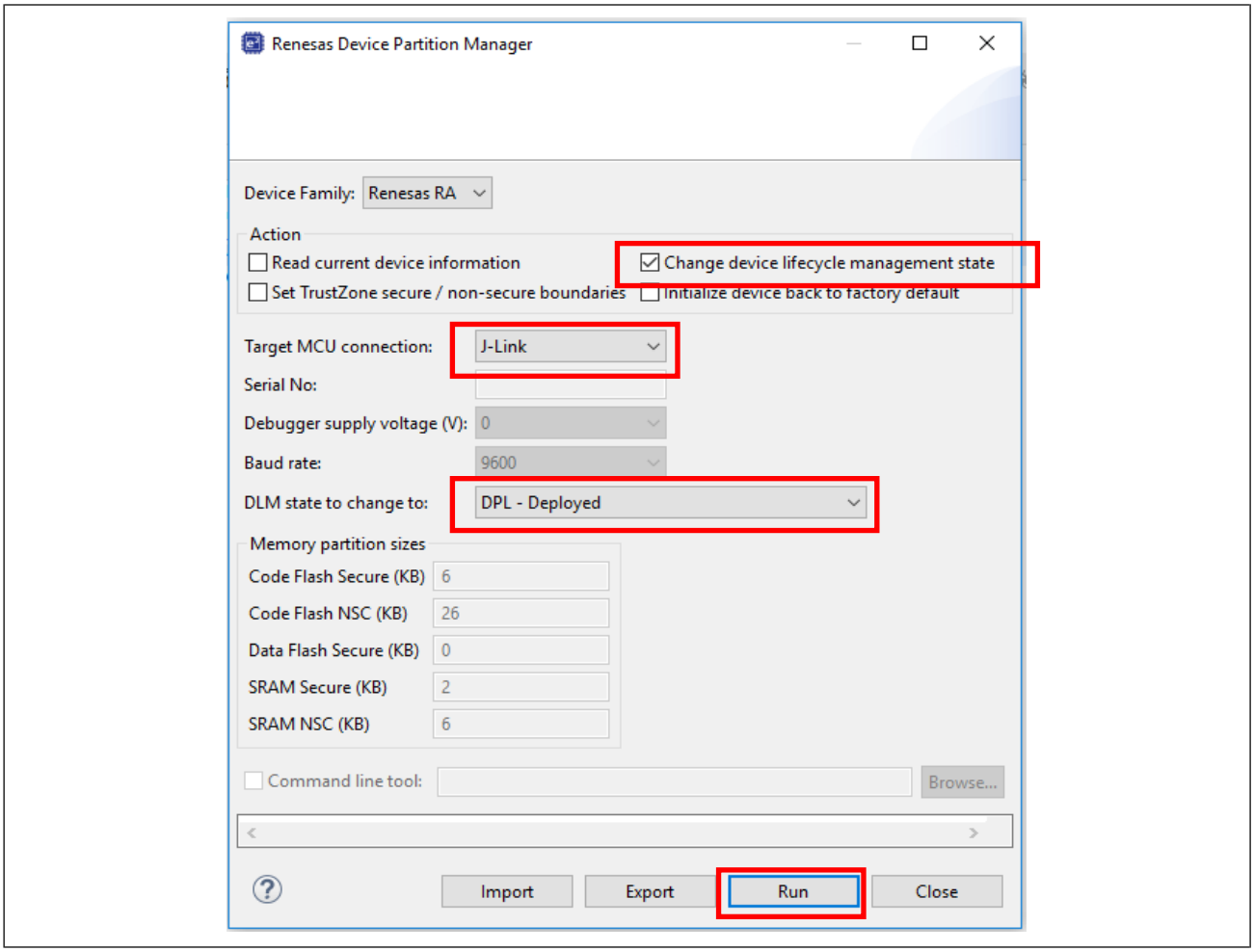

图 66. 将器件生命周期切换到 DPL

在 DPL 状态下，串行编程功能处于可用状态。用户可以使用瑞萨器件分区管理器将 MCU 初始化为 SSD，并擦除闪存内容（永久锁定的闪存块除外）。有关操作细节，请参见第 5.3.2 节的步骤 5.3.2.2。

对于使用 RFP 烧录非安全二进制文件的生产流程，请参见附录 A 的第 6.3 节了解操作细节。

这样便已完成本应用项目的指导过程。

6. 附录 A：将 RFP 用于生产流程

- 请注意，本章中的所有说明均基于RFP使用USB连接 J-Link 调试器的条件。对于其他连接，用户可参见《*RFP 用户手册*》了解相关说明。
- 本章中提供的所有说明都是为了支持第 5 章中所述的示例应用生产流程。每当联合项目开发模型和分离项目开发模型在生产操作上存在差异时，都会指出差异。但是，提供有关联合项目开发模型的生产流程的详细说明超出了本应用项目的范围。在将这些 RFP 项目用于其他应用之前，用户需要使用其应用特定的 IDAU 区域设置对这些 RFP 项目进行调整。

6.1 初始化 MCU

在使用 RFP 执行**“Initialize Device”**（初始化器件）之前，按照第 5.3.1 节建立以下连接。

接下来，启动 RFP，打开“`\RFP_projects\initialize_mcu\initialize_mcu.rpj`”，转到**“Device Information”**（器件信息）选项卡，选择**“Initialize Device”**（初始化器件）。

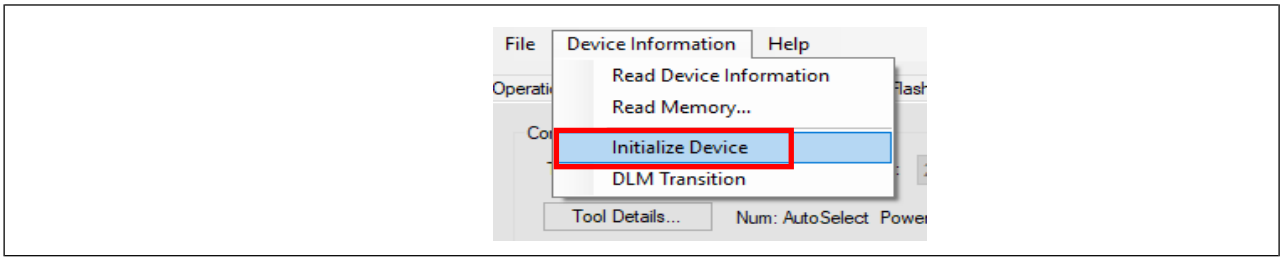

图 67. 使用 RFP 初始化

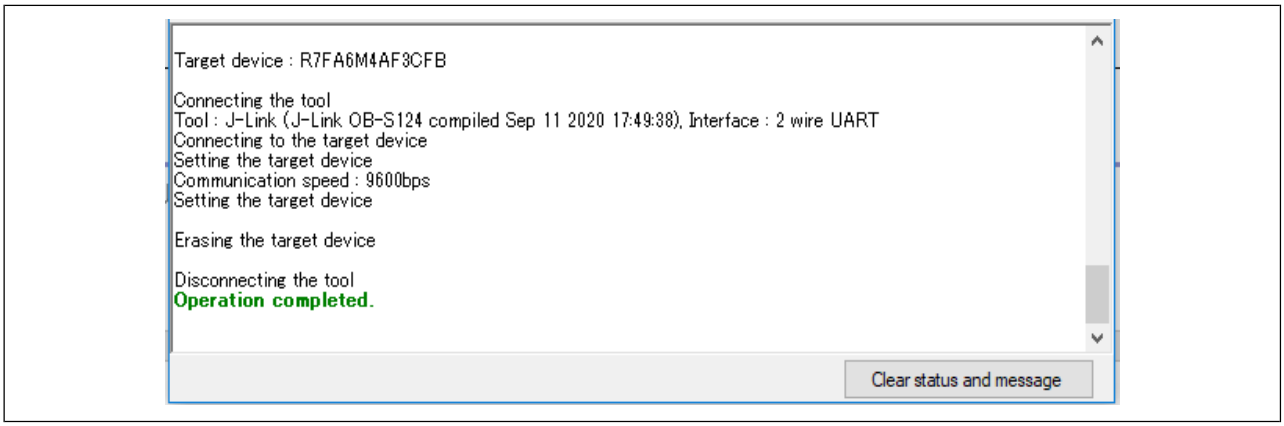

图 68. MCU 初始化成功

6.2 烧录安全二进制文件

打开附加的 RFP 项目 `\RFP_projects\pre_programmed_sensor_algorithm_s\pre_programmed_sensor_algorithm_s.rpj` 以执行以下功能：

- 烧录安全二进制文件进行
- 设置 IDAU 区域
- 切换到 NSECSD

图 69 所示为**“Operation Settings”**（操作设置）选项卡的设置。

- 选择**“Program”**（烧录）和**“Verify”**（验证），以便可以对安全二进制文件进行烧录和验证。
- 选择**“Program Flash Option”**（烧录闪存选项）和**“Verify Flash Option”**（验证闪存选项），以便可以设置和验证 IDAU 和器件生命周期状态。
- 未选择**“Erase”**（擦除）的原因是它已通过初始化命令处理，如第 6.1 节所示。

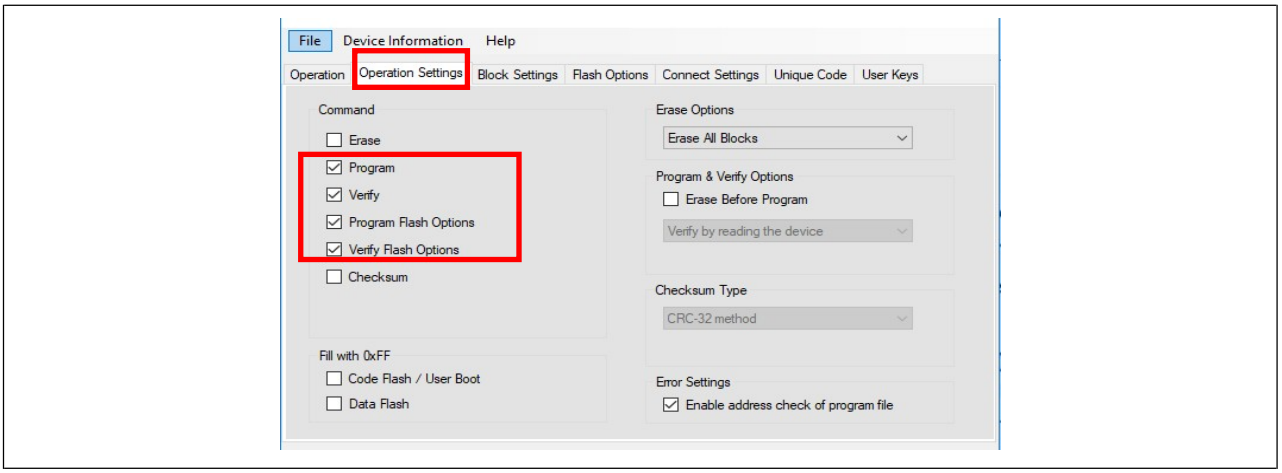

图 69. 设置操作设置 (RFP)

下面是本示例应用的 DLM 状态切换设置和 IDAU 区域设置，具体在**“Flash Options”**（闪存选项）选项卡下配置。

请注意，通过 RFP，可以直接将 MCU 器件生命周期状态从 SSD 切换到 NSECSD，而无需下载虚拟非安全二进制文件。虚拟非安全二进制文件仅用于启动非安全项目开发。

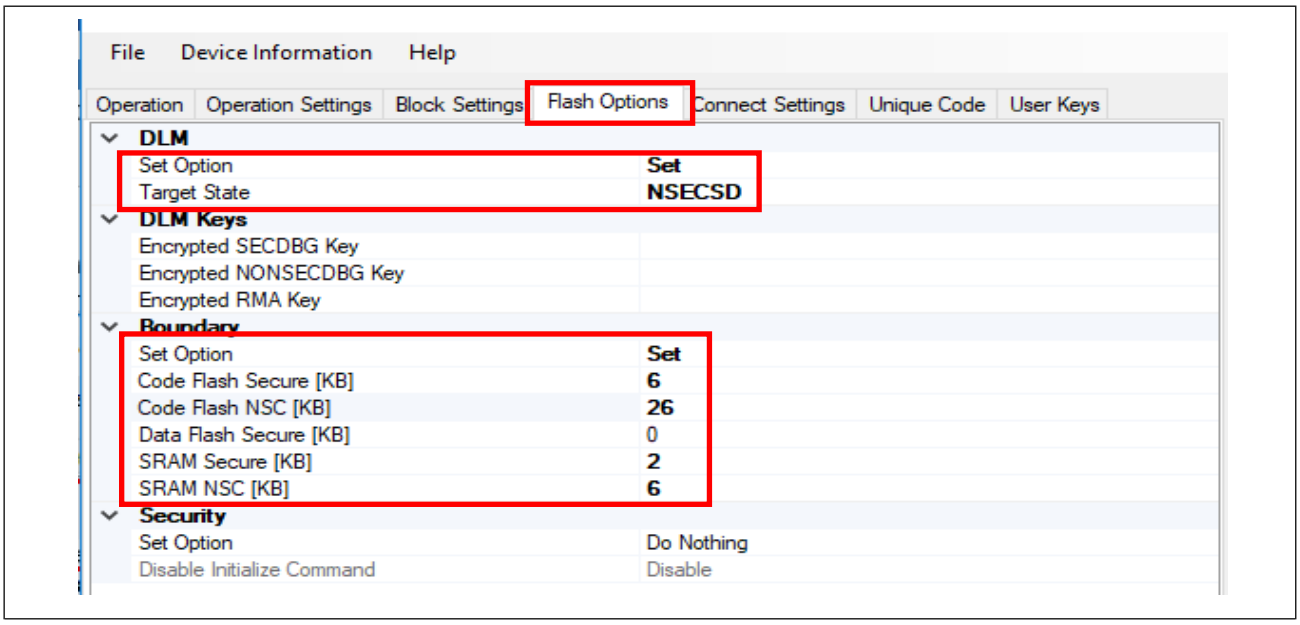

图 70. 设置 IDAU 区域

连接接口的设置如图 71 所示。

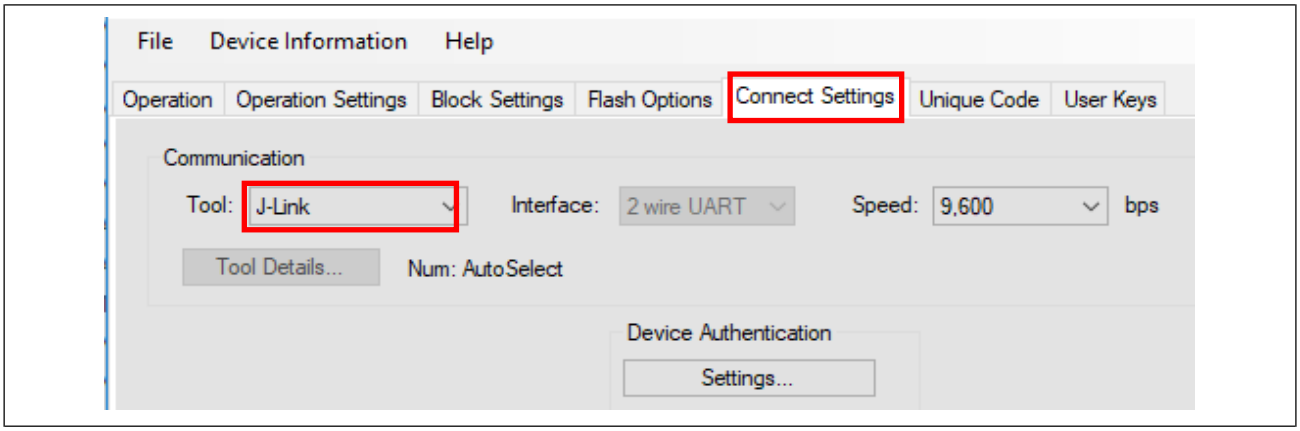

图 71. 设置连接

选择要烧录到 MCU 中的安全项目二进制文件（.srec 或 .hex）。

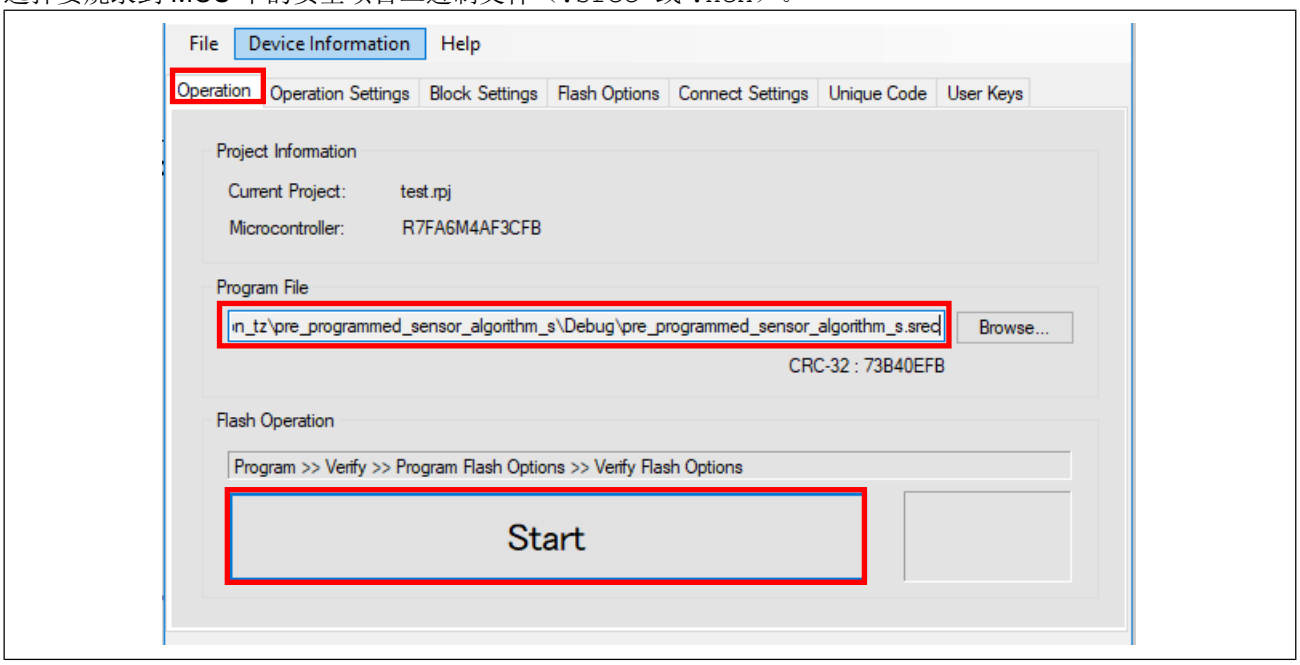

图 72. 选择要编程到 MCU 中的安全二进制文件

完成所有设置后，用户可以单击**“Start”**（开始）烧录安全二进制文件并设置 IDAU 区域。

6.3 烧录非安全二进制文件

使用 RFP 通过提供的 RFP 项目烧录非安全项目二进制文件：

\RFP_projects\pre_programmed_sensor_algorithm_ns\pre_programmed_sensor_algorithm_ns.rpj。

取消选中**“Operation Settings”**（操作设置）选项卡中的**“Program Flash Option”**（烧录闪存选项）和**“Verify Flash Option”**（验证闪存选项）。

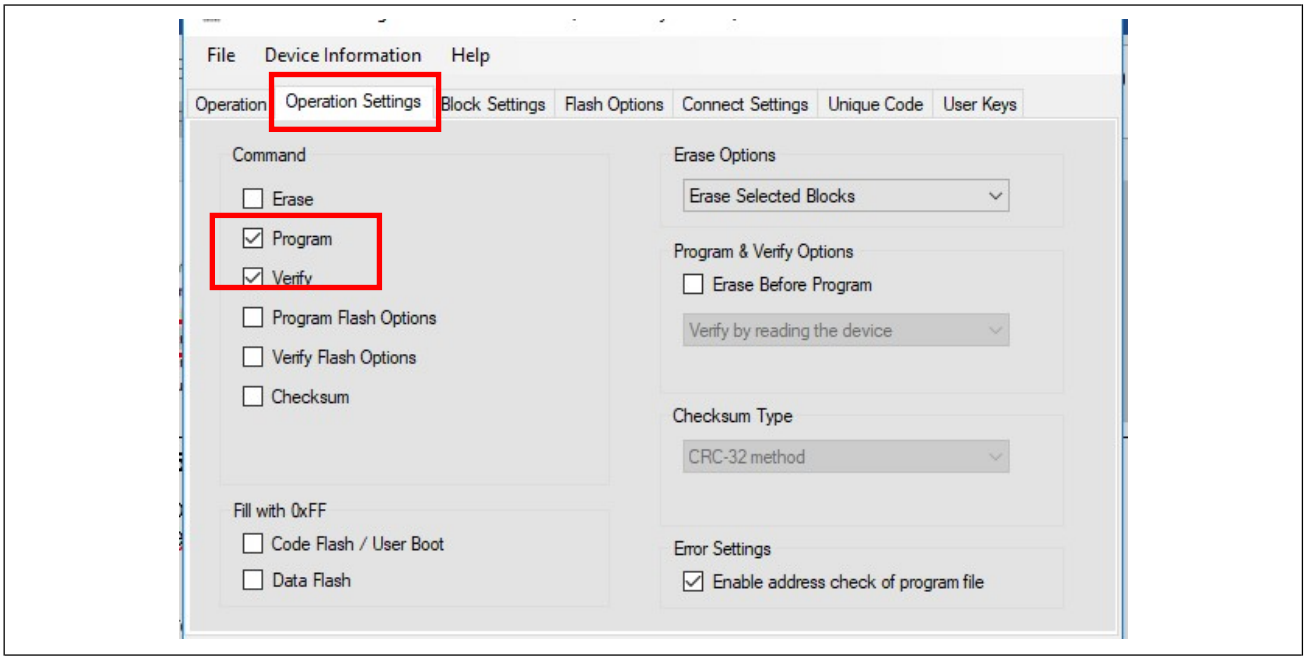

图 73. 非安全项目二进制文件烧录的操作设置

这里并未选择切换到 DPL。在生产流程中需要从**“Do Nothing”**（不执行任何操作）切换到**“Set”**（设置）。请注意，一旦切换到 DPL，便会禁用 JTAG 接口（无 Segger RTT 查看器输入/输出功能）。

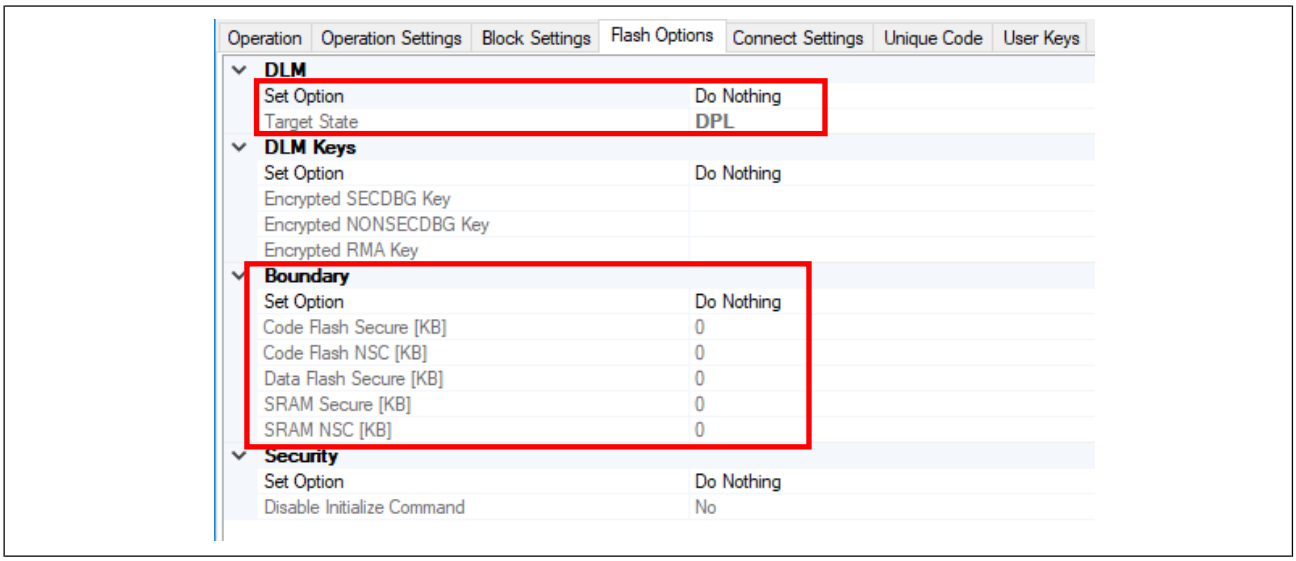

图 74. 非安全项目二进制文件下载的操作设置

“Connect Settings”（连接设置）应使用与图 71 所示相同的设置。

选择非安全二进制文件，如图 75 所示。

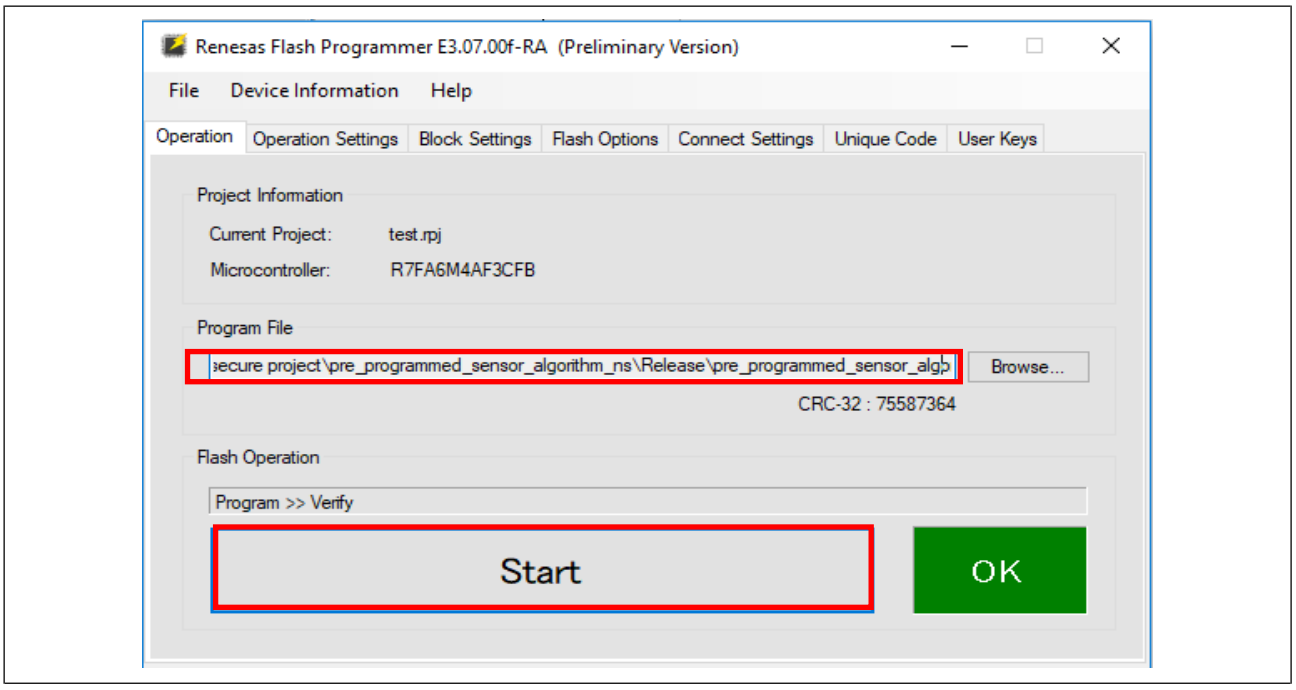

图 75. 选择非安全二进制文件

通过以上所有设置，用户可以单击**“Start”**（开始），烧录非安全二进制文件。

请注意，IP 保护用例的生产流程还需要将器件生命周期状态从 DPL 切换到 LCK_DBG 或 LCK_BOOT。但是，器件生命周期状态切换到 LCK_DBG 后，调试接口将被永久禁用。器件生命周期状态切换到 LCK_BOOT 后，串行编程接口将被永久禁用。为避免 MCU 调试和串行编程接口被意外禁用，除非这是用于生产用途，否则请勿将器件生命周期状态切换到 LCK_DBG 或 LCK_BOOT。

7. 附录 B：词汇表

术语	含义
SSD	器件生命周期状态：安全软件开发（**S**ecure **S**oftware **Development**）。调试级别为 DBG2。在这种状态下可以设置 IDAU 区域。
NSECSD	器件生命周期状态：非安全软件开发（**N**on-**SEC**ure **S**oftware **Development**）。调试级别为 DBG1。
DPL	器件生命周期状态：已部署（**DePL**oyed）。调试级别为 DBG0。
SCE9	安全加密引擎 9（**S**ecure **C**rypto **E**ngine **9**）：MCU 内孤立的子系统，受访问管理电路保护。执行加密操作。

8. 参考资料

1. *《瑞萨 RA6M4 系列用户手册：硬件》*
2. *《灵活配置软件包 (FSP) 用户手册》*
3. *《适用于 Armv8-M 架构的 Arm® TrustZone 技术》*
4. *《瑞萨 RA 产品家族 - 器件生命周期管理密钥安装 (R11AN0469CU)》*
5. *《瑞萨 RA 产品家族 - 使用 Arm TrustZone 保护静态数据 (R11AN0468EU)》*
6. *《Arm® v8-M 架构参考手册》*
7. *《Arm® Cortex®-M33 处理器技术参考手册》*
8. *《Arm® Cortex®-M33 器件通用用户指南》*

9. 网站和支持

如需了解 RA 单片机产品家族、下载工具和文档以及获得支持，请访问以下 URL。

EK-RA6M4 资源	renesas.com/ra/ek-ra6m4
RA 产品信息	renesas.com/ra
灵活配置软件包 (FSP)	renesas.com/ra/fsp
RA 产品支持论坛	renesas.com/ra/forum
瑞萨支持	renesas.com/support

版本历史记录

版本	日期	说明	
		页码	摘要
1.00	2020 年 10 月 1 日	—	初始版本
1.10	2021 年 6 月 2 日	—	更新到 FSP v3.0.0 并删除了 E2 的使用说明

注意

1. 本文档中电路、软件和其他相关信息的描述仅用于说明半导体产品的操作和应用示例。用户应对产品或系统设计中电路、软件和信息的纳入或任何其他用途承担全部责任。对于您或第三方因使用这些电路、软件或信息而引起的任何损失和损害，Renesas Electronics 不承担任何责任。
2. Renesas Electronics 特此声明，对于因使用本文档中所述的 Renesas Electronics 产品或技术信息（包括但不限于产品数据、图纸、图表、程序、算法和应用示例）而引起的侵权或与第三方有关的专利、版权或其他知识产权的任何其他索赔，概不承担任何责任和赔偿。
3. 对 Renesas Electronics 或其他公司的任何专利、版权或其他知识产权均不授予任何明示、暗示或其他形式的许可。
4. 您应负责确定需要从任何第三方获得哪些许可，并在需要时为合法进口、出口、制造、销售、使用、分销或以其他方式处置包含 Renesas Electronics 产品的任何产品获得此类许可。
5. 不得对 Renesas Electronics 产品的全部或部分进行更改、修改、复制或逆向工程。对于因更改、修改、复制或逆向工程而导致您或第三方蒙受的任何损失或损害，Renesas Electronics 不承担任何责任。
6. Renesas Electronics 产品根据以下两个质量等级进行分类：“标准”和“优质”。Renesas Electronics 每种产品的预期应用取决于产品的质量等级，具体如下所示。
   “标准”：计算机、办公设备、通信设备、测试和测量设备、视听设备、家用电器、机械工具、个人电子设备、工业机器人等
   “优质”：运输设备（汽车、火车、轮船等）；交通管制（交通信号灯）；大型通信设备；关键金融终端系统；安全控制设备等
   除非在 Renesas Electronics 数据手册或 Renesas Electronics 其他文档中明确指定为高可靠性产品或用于恶劣环境的产品，否则 Renesas Electronics 产品不适合或不授权用于可能对人类生命构成直接威胁或造成人身伤害（人造生命支持设备或系统；手术植入物等），或者可能造成严重的财产损失（空间系统、海底中继器、核动力控制系统、飞机控制系统、关键设备系统、军事装备等）的产品或系统。对于因使用任何与 Renesas Electronics 数据手册、用户手册或其他 Renesas Electronics 文档不一致的 Renesas Electronics 产品而引起的您或任何第三方所造成的任何损坏或损失，Renesas Electronics 不承担任何责任。
7. 没有任何半导体产品是绝对安全的。尽管 Renesas Electronics 的硬件或软件产品中可能实施了任何安全措施或功能， Renesas Electronics 对因任何漏洞或侵骇（包括但不限于以任何未经授权的方式访问或使用 Renesas Electronics 产品或使用 Renesas Electronics 产品的系统）而产生的任何后果概不负责。RENESAS ELECTRONICS 不担保或保证 RENESAS ELECTRONICS 产品或使用 RENESAS ELECTRONICS 产品创建的任何系统不会被破坏，或者可免于数据损坏、攻击、病毒、干扰、黑客攻击、数据丢失或失窃或其他安全入侵（“漏洞问题”）。RENESAS ELECTRONICS 不承担由任何漏洞问题引起的或与之相关的任何和所有责任或义务。此外，在适用法律允许的范围内，RENESAS ELECTRONICS 不对本文件和任何相关或附带的软件或硬件提供任何和所有明示或暗示的保证，包括但不限于对适销性或对特定用途的适用性的暗示保证。
8. 使用 Renesas Electronics 产品时，请参见最新的产品信息（数据手册、用户手册、应用笔记、可靠性手册中的“处理和使用半导体器件的一般说明”等），并确保使用条件符合 Renesas Electronics 在最大额定值、工作电源电压范围、散热特性和安装等方面的规定。对于因在超出上述规定范围的条件下使用 Renesas Electronics 产品而引起的任何失常、故障或事故，Renesas Electronics 不承担任何责任。
9. 尽管 Renesas Electronics 致力于提高 Renesas Electronics 产品的质量和可靠性，但半导体产品具有特定的特性，例如在特定速率下发生故障以及在某些使用条件下出现故障。除非在 Renesas Electronics 数据手册或 Renesas Electronics 其他文档中指定为高可靠性产品或用于恶劣环境的产品，否则 Renesas Electronics 的产品将不受抗辐射设计的约束。用户应负责采取安全措施，以防止人身伤害、火灾造成的伤害，和/或因 Renesas Electronics 产品发生故障或失常而对公众造成的危险，例如硬件和设备的安全设计，包括但不限于冗余、火控和故障预防、针对老化退化的适当处理或任何其他适当的措施。由于对微型计算机软件进行评估非常困难且无实操性，因此用户有责任评估自己生产的最终产品或系统的安全性。
10. 请联系 Renesas Electronics 销售办事处，以获取有关环境事宜的详细信息，例如每个 Renesas Electronics 产品的环境相容性。用户有责任认真、充分地研究有关纳入或使用受控物质的适用法律和法规（包括但不限于欧盟 RoHS 指令），并按照所有适用法律和法规使用 Renesas Electronics 产品。对于因您未遵守适用的法律和法规而造成的损坏或损失，Renesas Electronics 不承担任何责任。
11. Renesas Electronics 产品和技术不得被用于或纳入为任何适用的本国或外国法律、法规所禁止制造、使用或销售的产品或系统范围内。用户应遵守由对当事方或交易拥有管辖权的任何国家/地区的政府颁布和管理的任何可适用的出口控制法律和法规。
12. 应由Renesas Electronics 产品的购买方或分销商，或者对产品进行分发、处置或以其他方式出售或转让给第三方的任何其他当事方，负责将本文档中阐明的内容和条件提前通知前述第三方。
13. 未经 Renesas Electronics 事先书面同意，不得以任何形式全部或部分重印、再现或复制本文档。
14. 如果对本文档中包含的信息或 Renesas Electronics 产品有任何疑问，请联系 Renesas Electronics 销售办事处。

（注 1）本文档中的“Renesas Electronics”是指 Renesas Electronics Corporation，也包括其直接或间接控制的子公司。
（注 2）“Renesas Electronics 产品”是指 Renesas Electronics 开发或制造的任意产品。

（版本 5.0-1 2020 年 10 月）

公司总部

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

联系信息

有关产品、技术、文档最新版本或离您最近的销售办事处的更多信息，请访问：www.renesas.com/contact/。

商标

Renesas 和 Renesas 徽标是 Renesas Electronics Corporation 的商标。
所有商标和注册商标都是各自所有者的财产。