Application Note

MSPM0 MCU 中的网络安全机制

摘要

MSPM0 微控制器提供了多种信息安全机制，旨在帮助开发人员实施其安全措施来保护代码、数据和密钥等资产。此外，MSPM0 器件中还提供了用于安全启动和安全存储的硬件和软件组合解决方案。本文档介绍了这些器件中提供的安全机制及其功能和限制、运行方式，以及如何针对基本用例对它们进行配置。

内容

商标

1 引言

随着工业、汽车和个人电子产品应用的互联程度越来越高，并且攻击者可以使用的工具不断增加，嵌入式应用中的器件安全变得越来越重要。TI 的 MSPM0 微控制器包含各种硬件和软件安全支持技术，供工程师在开发注重安全性的应用时使用。

1.1 关键概念

表 1-1. 关键概念

术语	含义
NONMAIN	一个专用闪存区域，可配置器件引导相关参数。请参阅 MSPM0 NONMAIN 闪存操作指南以了解 NONMAIN 运行指南。
安全启动	作为执行的先决条件，确认和验证可更新固件和软件组件的完整性和真实性的过程。
INITDONE	INITDONE 是一些 MSPM0 器件中的一个寄存器，用于隔离特权状态和非特权状态。INITDONE 在特权状态结束时由 CSC 触发，而 CSC 中配置的所有非静态安全策略都将在 INITDONE 期间生效。
客户安全代码 (CSC)	MSPM0 SDK 中针对具有 INITDONE 机制的器件提供的安全启动解决方案。它是信任根的一部分，可在生产后保持不可更改，并实现应用程序映像完整性和真实性验证以及其他安全策略配置。CSC 还可以表示 MSPM0 硬件特性，这意味着 MSPM0 器件支持 INITDONE 机制。
启动映像管理器 (BIM)	MSPM0 SDK 中针对没有 INITDONE 机制的器件提供了安全启动解决方案。
信任根 (RoT)	特别是指不可更改的信任根，这是设备上最受信任的安全组件。它本质上是可信的，因为它不能在制造后进行修改。不存在任何更深层次的软件可以验证它是否真实且未经修改。在 CSC 解决方案中包括 ROM 引导代码及具有静态写保护的 CSC。
密钥库	AES 密钥的安全存储。只有 CSC 可以将密钥配置到密钥库中，主应用程序可以将加密引擎 (AES) 配置为使用存储的密钥之一，但永远无法访问任何存储的密钥。
防火墙	针对闪存存储器某些特定区域的动态保护机制，包括写保护、读取-执行保护和 IP 保护。
存储体交换	在 MSPM0 双组器件上配置闪存存储体地址映射的机制。它在 CSC 中进行配置，并在 INITDONE 之后生效。
静态写保护	由 NONMAIN 配置启用的静态写保护机制。除非更改了 NONMAIN 配置以再次启用写入，否则 ROM 引导代码完成后无法修改受保护区域。
SHA2-256	哈希算法采用整个消息并且将其压缩成一个固定长度（256 位）摘要。它用来验证消息完整性。仅在 MSPM0 器件中通过软件来支持。
ECDSA P256	用来验证消息真实性的非对称算法。仅在 MSPM0 器件中通过软件来支持。
AES	高级加密标准，一些 MSPM0 器件为 AES 提供硬件加速器。
TRNG	真随机数生成器，一些 MSPM0 器件为 TRNG 提供硬件加速器。

1.2 网络安全目标

一般而言，在嵌入式应用中，网络安全的主要目标是在如下方面保护关键资产：

- 机密性（对机密数据保密）
- 完整性（保护数据不被修改）
- 真实性（确保各方都是真实的）
- 可用性（确保数据和/或功能在需要时可用）
- 不可否认性（数据的来源和/或身份对其他各方都是可证明的）

这些关键目标通常适用于可能处于以下状态的资产：

- 静态（微控制器上未在使用的代码、数据或密钥）
- 使用中（微控制器上正在应用中使用的代码、数据或密钥）
- 传输中（微控制器上正在 MCU 和其他实体之间移动的代码、数据或密钥）

1.3 平台信息安全机制

MSPM0 器件中包含的安全启用程序如表 1-2 所示。调试安全功能和主闪存存储器完整性验证功能可以在器件系列技术参考手册的 NONMAIN 布局类型和 NONMAIN 寄存器部分找到。安全启动、安全存储和加密加速器特性可以在特定于器件的数据表中找到。

表 1-2. MSPM0 MCU 平台信息安全机制

信息安全机制	器件特性	M0C1103/4	M0C1105/6	M0L1x0x/M0L134x	M0G110x	M0G3x0x/M0G150x	M0H321x	M0L111x	M0Lx22x	M0Gx51x
调试安全性	密码验证的调试访问		哈希化	是	是	是	哈希化	哈希化	哈希化	哈希化
	密码验证的引导加载程序访问	无 ROM BSL	无 ROM BSL	是	是	是	无 ROM BSL	哈希化	哈希化	哈希化
	密码验证的主闪存批量擦除		哈希化	是	是	是	哈希化	哈希化	哈希化	哈希化
	密码验证的完全恢复出厂设置		哈希化	是	是	是	哈希化	哈希化	哈希化	哈希化
	TI 失效分析 (FA) 启用/禁用		是	是	是	是	是	是	是	是
	串行线调试 (SWD) 接口的完全硬件禁用	是	是	是	是	是	是	是	是	是
	可永久锁定的器件配置数据	是	是	是	是	是	是	是	是	是
	防错器件配置数据		是	是	是	是	是	是	是	是
	密码仅用哈希形式存储 (SHA2-256)		是				是	是	是	是
安全启动	CSC 存在		是				是	是	是	是
	可永久锁定的主闪存存储器	是	是	是	是	是	是	是	是	是
	CRC-32 验证的主闪存区域		是	是	是	是	是	是	是	是
	SHA2-256 验证的主闪存区域		是				是	是	是	是
	引导时主闪存应用程序的单点入口	是	是	是	是	是	是	是	是	是
	非对称固件映像身份验证例程（基于软件的 ECDSA、P-256、SHA2-256）		是	是	是	是	是	是	是	是
	对称固件映像身份验证例程（基于硬件的 AES-CMAC）							是	是	是
	用于 ECDSA 公共密钥撤销和回滚保护的可锁定闪存		是				是	是	是	是
	SRAM 写入执行互斥 (W^X) 边界		是	是	是	是	是	是	是	是

表 1-2. MSPM0 MCU 平台信息安全机制（续）

信息安全机制	器件特性	M0C1103/4	M0C1105/6	M0L1x0x/M0L134x	M0G110x	M0G3x0x/M0G150x	M0H321x	M0L111x	M0Lx22x	M0Gx51x
安全存储	闪存写保护防火墙		是				是	是	是	是
	闪存读取/执行 (RX) 保护防火墙		是				是	是	是	是
	闪存 IP 保护区域（仅执行，无读取访问权限）		是				是	是	是	是
	闪存存储体写入执行互斥 (W^X)							是	是	是
	数据存储体读写保护									是
	密钥存储库（最多四个 128 位密钥或两个 256 位密钥，加上一个会话密钥）							是	是	是
	硬件单调计数器							是		是
加密加速	具有自检功能的真随机数发生器 (TRNG)					是		是	是	是
	基本 AES 加速器（不支持 GCM/CMAC/GHASH）					是		是	是	是
	高级 AES 加速器（支持 GCM/CMAC/GHASH）							是	是	是
器件身份	唯一器件标识符（96 位）	是	是	是	是	是	是	是	是	是
认证	ARM PSA 电平						L1 计划	L1 计划	L1	L1 计划
	EVITA 功能					EVITA-Light		EVITA-Light	EVITA-Light	EVITA-Light
	符合 ISO 21434 流程							计划		计划
攻击抵抗分析	3P 固件漏洞分析						是	是	是	是
	引导配置例程故障注入攻击对策						是	是	是	是

2 器件安全模型

MSPM0 系列的安全功能具有两个广泛的阶段：

1. 静态、TI 编写的引导代码受用户定义的配置的影响，在进入闪存应用程序之前强制执行
2. 用户编写的客户安全代码 (CSC) 位于 MAIN 闪存中，受静态写保护，在 BCR 之后运行，并在跳转到应用程序之前强制执行附加策略和/或验证应用程序的真实性和完整性

并非所有器件都支持额外 CSC 步骤。虽然只有 CSC 可强制执行其他安全策略（例如防火墙），但不支持 CSC 的器件可能仍会验证应用程序。应用程序验证将在“安全启动”部分进一步讨论，因此我们将本章中的讨论限于强制执行其他策略，并将此阶段称为 CSC。

本节概述了两个类别中器件启动过程和用户指定的策略，用户可以设置这些策略来支持各种用例。

2.1 器件标识

所有 MSPM0 器件都包含一个特定于器件的 96 位识别码（器件 ID），该识别码可由应用软件读取。有关器件 ID 的更多信息，请参阅技术参考手册和器件数据表。

TI 设计的器件 ID 对于每个发运的器件都是唯一的，因此可用于识别特定器件或将其与任何其他器件区分开来。虽然器件 ID 是唯一的，但它不是加密随机值，因为一些位对应于器件特性，例如器件型号和产品版本。

2.2 启动时的初始条件

在冷上电 (POR) 期间，器件会复位至一个安全状态。数字 IO 引脚采用高阻抗配置，所有外设功能均断开连接，NRST 引脚处于 NRST 模式，并且串行线调试 (SWD) 接口引脚处于 SWD 模式。在欠压复位释放后，串行线调试端口 (SW-DP) 最初会启用，以允许一个调试探针建立到调试子系统的初始连接。

在启动过程的这一阶段，调试探针可访问的唯一调试访问端口 (DAP) 是配置访问点 (CFG-AP) 和安全访问点 (SEC-AP)。连接的调试探针可以使用 CFG-AP 来读取通用器件信息（例如器件的通用器件型号）。SEC-AP 可用于尝试向引导配置例程传递命令消息。对器件的应用调试访问（通过 AHB-AP、ET-AP 和 PWR-AP DAP）仍会被硬件防火墙阻止。因此，器件硬件不允许在器件加电期间对处理器、EnergyTrace 状态或电源配置进行任何调试访问。

在欠压复位 (BOR) 之后，始终会生成引导复位 (BOOTRST)，从而开始执行引导配置例程。

2.3 引导配置例程 (BCR)

MSPM0 器件在只读存储器 (ROM) 中包含一个不可变的信任根引导配置例程。引导配置例程 (BCR) 始终是紧跟器件的 BOOTRST 之后在 Cortex-M0+ 处理器上运行的第一个代码。BCR 还会在软件调用引导加载程序 (BSL) 时运行，因为这是授权 BSL 条目所需的。BCR 的核心职责是：

1. 将器件正常运行所需的 TI 工厂数据从 FACTORY 闪存存储器区域加载到逻辑中，并通过 CRC-32 验证工厂数据（包括器件修整数据）的完整性
2. 将用户指定的器件配置（包括安全策略）从 NONMAIN 闪存区域加载到逻辑中，并通过 CRC-32 验证用户配置数据的完整性
3. 检查是否通过串行线调试 (SWD) 接口发送任何引导命令，对这些命令进行授权（如果适用）并进行处理（如经授权）
4. 如果启用了 BSL，检查是否存在引导加载程序 (BSL) 调用条件，如果发生有效调用，则启动 BSL
5. 在使用 CRC-32 或 SHA-256 启动用户应用程序之前，检查 MAIN 闪存存储器区域中包含用户应用程序代码的用户定义部分的完整性
6. 确定在 CSC 发出 INITDONE 之后，还是在 BCR 结束时释放 AHP-AP、ET-AP 和 PWR-AP DAP（详见 FIX 部分）。
7. 将任何引导错误记录到 CFG-AP
8. 通过从 MAIN 闪存中的 0x0000.0000 获取堆栈指针和从地址 0x0000.0004 获取重置向量，触发硬件从 MAIN 闪存中的单个入口点开始执行

2.4 引导加载程序 (BSL)

MSPM0 器件还可能在只读存储器 (ROM) 中包含一个不可变的引导加载程序 (BSL)。与串行线调试 (SWD) 接口相反，BSL 提供了一种通过标准串行接口（UART 或 I2C）对器件存储器内容进行编程和验证的方法。

BSL 只能由 BCR 启动。BCR 会检查是否存在有效的 BSL 调用条件（软件调用、IO 引脚调用、空白器件调用），并验证在启动 BSL 之前是否启用了 BSL 以供使用。当 BSL 退出时，BCR 会再次运行以加载当前器件安全策略并启动用户应用程序。

BSL 始终受用户指定的 256 位密码保护，在启动 BSL 会话时，该密码必须通过 UART 或 I2C 接口传递给 BSL。如果不使用 BSL，则可以将其禁用（请参阅 BSL 启用/禁用策略）。

2.5 启动流程

在下图中显示了 MSPM0 器件的启动流程概要。

在 BOOTRST 时，TI 引导代码开始执行。成功引导之后，引导代码会发出 BOOTDONE。此时，SYSCTL 向器件发出 SYSRST 以触发从闪存执行。根据引导配置记录，这会触发启动主应用程序（如果此配置中不存在 CSC）或启动 CSC（如果配置了 CSC）。

CSC 负责确定执行存储体、存储器区域保护、安全密钥初始化到密钥库等。当客户安全代码发出 INITDONE（通过写入 SYSCTL.SECCFG.INITDONE MMR）时，SYSCTL 会发出第二个 SYSRST。器件再次从映射到闪存的 0x0 开始执行，而 CSC 会执行第二次。这一次，CSC 将发现先前已经发出 INITDONE（这是通过读取 SYSCTL.SECCFG.SECSTATUS.INITDONE 位来确定的），因此会直接调用主应用程序。

从 BOOTRST 到主应用程序的概括性引导流：

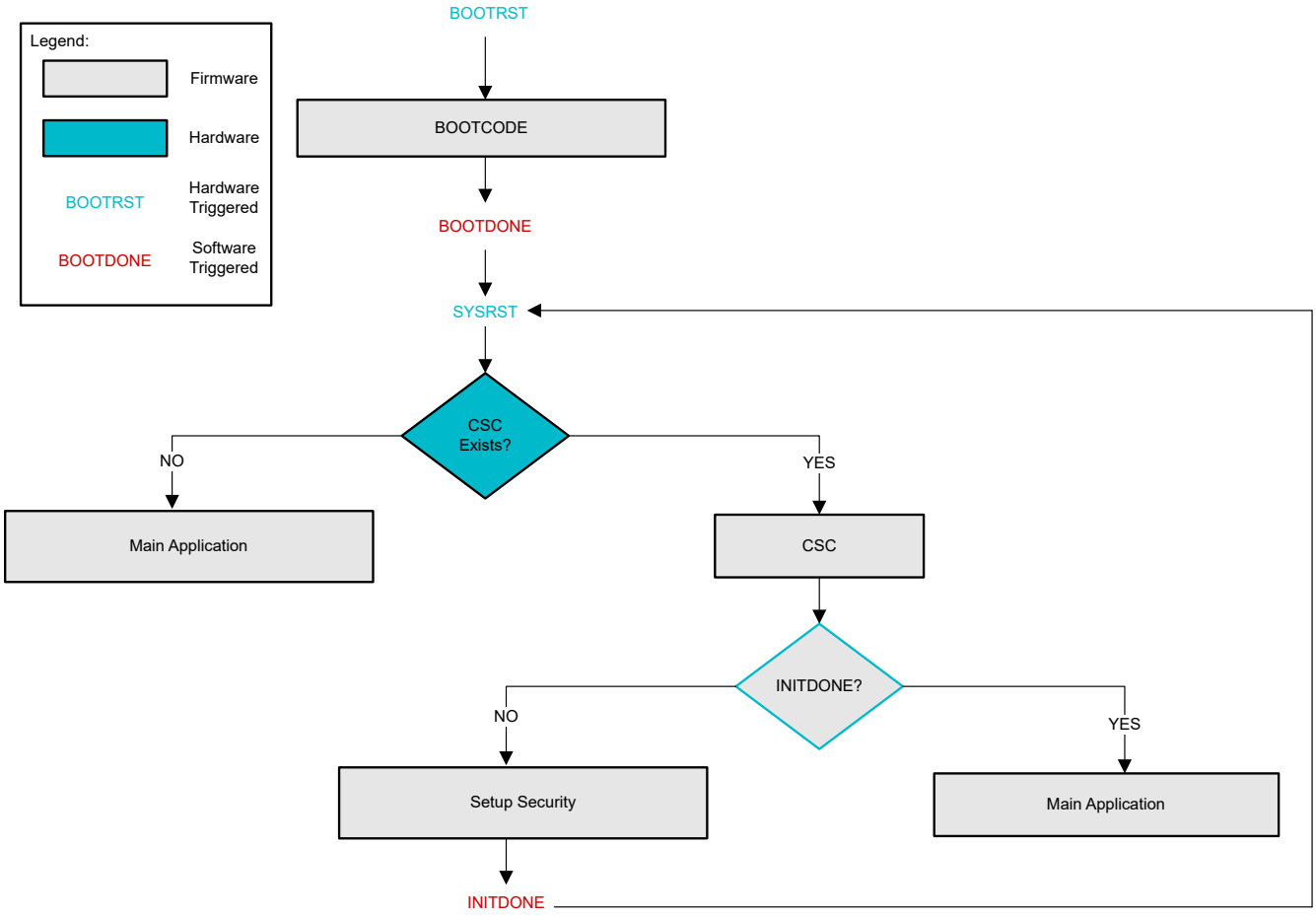

图 2-1. 启动流程概要

备注

为了使引导流程与启用了非安全功能的器件保持兼容，引导配置的默认设置设为“CSC 不存在”状态。

安全执行流程是 CSC_EXISTS = YES 情况下的路径。在这种情况下，可能会观察到在 BOOTRST 之后，有两个 SYSRST 将在主应用程序启动之前发出。在首个 SYSRST 之后，客户启动代码开始执行。它会配置安全性并发出 INITDONE。此时会锁定并强制执行安全配置。随即会发出第二个 SYSRST 以重新开始执行启动代码。在第二个 SYSRST 时，由于 INITDONE 为 YES，因此会启动主应用程序。

请注意，BCR 和 BSL 都在可锁定的 NONMAIN 闪存区域中包含用户指定的配置数据结构。节 2.6 中介绍了通过这些数据结构指定的这些安全策略。

2.6 用户指定的安全策略

MSPM0 器件包含一个专用的闪存区域，用于存储用户指定的安全和器件配置策略。该区域称为 NONMAIN 闪存区域。引导配置例程 (BCR) 和引导加载程序 (BSL) 会引用 NONMAIN 闪存区域中存储的用户指定数据以配置器件的运行。

在生产过程中，用户必须按照所需的策略配置器件的 NONMAIN 闪存区域。本节将介绍用户可通过 NONMAIN 配置存储器配置的安全策略。

NONMAIN 闪存区域分为两个不同的数据结构：

- BCR 配置（如 节 2.3 所述），用于设置引导配置安全策略
- BSL 配置（如 节 2.6.1.4 所述），用于设置引导加载程序安全策略

这两个数据结构都由自己的 32 位 CRC 摘要支持，这些摘要用作配置数据防错方案的一部分。

备注

BCR 和 BSL 配置结构中包含了本文档中所示之外的其他参数；本文档重点介绍了与安全性相关的参数。有关 NONMAIN 闪存区域中 BCR 和 BSL 配置结构的完整说明，请参阅相应技术参考手册中架构一章的引导配置部分。

2.6.1 引导配置例程 (BCR) 策略

下一节介绍了在闪存的 NONMAIN 区域中静态定义并通过引导配置例程 (BCR) 强制执行的策略。除非另有说明，否则所有 MSPM0 器件都能够强制执行这些策略。

2.6.1.1 串行线调试相关策略

串行线调试相关策略配置可通过器件的物理调试接口 (SWD) 获得的功能。默认情况下，TI 的 MSPM0 器件处于不受限制的状态。该状态支持轻松地进行生产编程、评估和开发。但是，不建议在大规模生产中使用这种不受限制的状态，因为它会留下很大的攻击面。为了在保持配置过程简单的同时满足各种需求，MSPM0 器件支持三个通用安全级别：无限制（级别 0）、自定义限制（级别 1）和完全限制（级别 2）。表 2-1 显示了三种通用安全级别，从限制性最低到限制性最高。

SWD 接口有 4 种主要用途需要考虑保护：

- 应用调试访问，其中包括：
  - 通过 AHB-AP 对处理器、存储器映射和外设进行完全访问
  - 通过 ET-AP 访问器件 EnergyTrace+ 状态信息
  - 通过 PWR-AP 访问器件电源状态控制以进行调试
- 批量擦除访问，其中包括：
  - 能够通过 SWD 发送一条命令来擦除 MAIN 内存区域，同时保持 NONMAIN 器件配置存储器不变
- 恢复出厂设置访问，其中包括：
  - 能够通过 SWD 发送命令来擦除 MAIN 存储器区域并将 NONMAIN 器件配置存储器恢复到 TI 出厂默认设置（0 级）
- TI 失效分析访问，其中包括：

- TI 能够通过 SWD 启动失效分析回流（请注意，在向 TI 提供 FA 访问权限之前，TI FA 流始终强制恢复出厂设置；这确保在失效分析流程启动时，TI 没有任何机制来读取存储在器件闪存中的专有客户信息）

表 2-1. 通用安全级别

级别	场景	SW-DP 策略	应用调试策略	批量擦除策略	恢复出厂设置策略	TI FA 策略
0	无限制	EN	EN	EN	EN	EN
1	自定义限制	EN	EN、DIS	DIS	EN、DIS	EN、DIS
2	完全限制	DIS	无关（禁用 SW-DP 时无法访问）(1)			

(1) 当 SW-DP 策略是**禁用 SW-DP** 时，从 SWD 接口的角度来看，批量擦除和恢复出厂设置策略是无关的。但是，如果启用了引导加载程序 (BSL)，则批量擦除和恢复出厂设置策略会影响通过 BSL 提供的功能。有关保护 BSL 的详细信息，请参阅 BSL 安全性部分。

2.6.1.1.1 SWD 安全级别 0

SWD 安全级别 0 是限制性最低的 SWD 安全状态。这是 TI 新器件的默认状态，也是器件在成功恢复出厂设置后的状态。此状态下对应用调试访问、整体擦除、恢复出厂设置和失效分析没有限制。

何时使用此状态

级别 0 非常适合原型设计和开发，因为它允许对器件存储器进行编程以及对处理器和外设进行调试。

何时不应使用此状态

大规模生产中不应使用级别 0。攻击者可以完全自由地读取器件存储器的内容、操纵器件的执行，并或许能更改闪存的内容（取决于闪存写保护方案）。

2.6.1.1.2 SWD 安全级别 1

SWD 安全级别 1 允许自定义安全配置。物理调试端口 (SW-DP) 保持启用状态，并且每个功能（应用调试、批量擦除命令、恢复出厂设置命令和 TI 失效分析）都可以单独启用、禁用或（在某些情况下）通过密码身份验证启用，从而提供相当大的灵活性来根据特定用例定制器件行为。

何时使用此状态

级别 1 非常适合受限的原型设计/开发场景以及大规模生产场景，在这些场景中，需要保留某些 SWD 功能（例如恢复出厂设置和 TI 失效分析），同时禁用其他功能（例如应用调试）。表 2-2 中给出了级别 1 自定义配置的常见示例。

表 2-2. 级别 1 配置示例

级别 1 场景	配置			
	应用调试	批量擦除	恢复出厂设置	TI FA
这种场景使用用户指定的密码限制了调试访问，但保留了恢复出厂设置和 TI 失效分析。此配置允许进行现场调试（使用密码），另外还允许通过恢复出厂设置来将器件恢复到默认的“级别 0”状态。	EN（具有密码）	DIS	EN	EN
此场景不允许进行调试。它允许恢复出厂设置，但只能通过用户指定的密码来使用。这提供了一种在现场访问器件的方法，方法是在密码已知时清除 MAIN 存储器内容并将器件恢复到“级别 0”状态。重要的是，即使恢复出厂设置密码被泄露，攻击者也无法读取 MAIN 闪存中的专有信息。	DIS	DIS	EN（具有密码）	EN
此场景不允许调试，也不允许 TI 失效分析。这可防止 TI 在器件上执行恢复出厂设置和进一步的 FA 活动，除非用户在将器件返回 TI 进行 FA 之前使用自己指定的密码执行恢复出厂设置。	DIS	DIS	EN（具有密码）	DIS

备注

对于大多数标准生产用例，建议使用级别 1 配置。对于不需要安全启动的应用，TI 建议在生产中使用级别 1，同时保持启用恢复出厂设置（使用密码）和 TI 失效分析。在此类配置中，用户（使用密码）或 TI（通过失效分析返回流程）或许能在器件配置后将器件恢复到限制性较低的状态。在需要最大安全启动保证的用例中，可以使用限制性较高的级别 1 或级别 2 进行生产，但需要权衡的是，器件在配置后可能无法恢复到限制性较低的状态。

何时不应使用此状态

如果需要对器件进行完全访问，则不应在原型设计期间使用级别 1；在这种情况下，应使用级别 0。

级别 1 也不应用于需要最高限制状态且不启用 SWD 功能的大规模生产场景；在这种情况下，应改为使用级别 2，因为它直接禁用整个 SWD 物理接口，并更大限度地减少误配置的可能性。

备注

如果器件配置为禁用应用调试和恢复出厂设置，则用户要恢复对器件的调试访问，唯一方法是用户应用代码提供了一种机制，可将 NONMAIN 配置更改为限制性较低的状态。如果 NONMAIN 通过静态写保护锁定，则状态不可逆，用户无法重新获得调试访问。

2.6.1.1.3 SWD 安全级别 2

SWD 安全级别 2 会将器件配置为最高限制状态。物理调试端口 (SW-DP) 被完全禁用，所有 SWD 可访问的功能（应用调试、批量擦除、恢复出厂设置和 TI 失效分析）都不能通过 SWD 访问，无论它们的配置如何。

当选择级别 2（SW-DP 禁用）时，应用调试配置和 TI 失效分析配置字段都是无关字段，不会影响器件配置。

如果 BSL 被禁用，则批量擦除和恢复出厂设置配置字段也是无关字段。但是，如果 BSL 被启用，那么 BSL 仍然使用批量擦除和恢复出厂设置配置字段来授权来自 BSL 接口的批量擦除或恢复出厂设置命令。

何时使用此状态

级别 2 只用于大规模生产，这种情况下无需进一步访问任何 SWD 功能且器件需要达到最大安全状态。

何时不应使用此状态

请勿在以下情况下使用级别 2：

- 未来需要通过 SWD 进行应用调试或重新编程
- 为了 TI 能够对器件执行失效分析
- 为了通过 SWD 发送批量擦除或恢复出厂设置命令来从闪存中删除专有信息

备注

器件配置为级别 2（SW-DP 禁用）后，就**无法**通过 SWD 进一步访问器件。要将器件恢复到级别 0 或级别 1 状态并恢复 SWD 访问，唯一方法是启用 BSL 和恢复出厂设置（允许发送 BSL 恢复出厂设置命令），或者用户应用程序代码中包含一种机制，可将 NONMAIN 配置更改为限制性较低的状态。在任何一种情况下，如果 NONMAIN 通过静态写保护锁定，则级别 2 状态不可逆，法重新获得 SWD 访问。

2.6.1.2 引导加载程序 (BSL) 启用/禁用策略

与串行线调试接口相反，引导加载程序 (BSL) 提供了一种通过标准串行接口（UART 或 I2C）对器件存储器进行编程和验证的方法。BSL 具有自己的配置策略，但由 BCR 确定是否启用了 BSL 以进行调用，还是要禁用 BSL（不可调用）。

由于 BSL 提供了一个额外的攻击面，如果它未在应用程序中使用，则可以在用户指定的启动安全策略中禁用它。如果在应用程序中使用 BSL，则在 BSL 配置策略中管理 BSL 安全设置（包括 BSL 访问密码）。

2.6.1.3 闪存保护和完整性相关策略

闪存保护和完整性策略规定闪存的哪些扇区被锁定而无法修改，以及启动过程中在启动用户应用程序之前要检查哪些扇区的完整性。

2.6.1.3.1 锁定应用 (MAIN) 闪存

MSPM0 MCU 实现了一个静态写保护方案，以将 MAIN 闪存区域中用户定义的扇区锁定，从而防止在运行时针对相应扇区执行任何编程或擦除操作。所需的静态写保护方案配置为 NONMAIN 闪存区域中启动安全策略的一部分。

用途

静态写保护方案支持在闪存中存储一个用户定义的、具有以下特性的固定应用程序：

- 在编程结束并被锁定后，该应用程序就无法由应用程序代码或 ROM 引导加载程序进行修改
- 如果置于闪存的开头，该应用程序始终是 ROM 引导配置例程转换到执行用户应用程序时执行的第一个代码

MSPM0 静态写保护支持这两个特性，要实现安全启动映像管理器，必须满足这些特性。

功能

当引导配置例程将转换到执行 MAIN 闪存中的引导加载程序或用户应用程序代码时，在 NONMAIN 中配置为写入锁定的任何扇区都在功能上不可更改。如果应用程序代码或引导加载程序尝试对受静态保护的扇区进行任何编程或擦除，都会导致硬件闪存操作错误，并且扇区不会被修改。

静态写保护可防止应用程序代码或引导加载程序进行任何修改，但通过 SWD 接口发送的批量擦除或恢复出厂设置命令将被接受。如果不需要这种行为，可以使用唯一的密码来保护批量擦除或恢复出厂设置 SWD 命令，也可以禁用这两个命令（请参阅 SWD 策略）。要完全消除任何修改受静态写保护的 MAIN 闪存扇区的方法，必须禁用批量擦除和恢复出厂设置命令（或 SW-DP），并且 NONMAIN 引导配置存储器也必须具有静态写保护，以防止应用程序代码通过修改 NONMAIN 区域内容来更改底层写保护方案。下一节会对此进行介绍。

2.6.1.3.2 锁定配置 (NONMAIN) 闪存

MSPM0 MCU 实现了一个静态写保护机制，以在运行时锁定 NONMAIN 闪存区域，从而防止对该区域进行任何编程/擦除操作。写保护方案配置为 NONMAIN 闪存区域中启动安全策略的一部分。

用途

默认情况下，NONMAIN 配置存储器（包含用户指定的启动安全策略和引导加载程序策略）不受写保护。这样一来，用户就可以在配置期间擦除 NONMAIN，并使用可用于大规模生产的用户指定策略重新编程。

在许多情况下，配置存储器最好在配置完毕后锁定。锁定配置存储器的好处是可以防止引导加载程序或应用程序代码对安全策略、引导加载程序策略和静态写保护策略进行任何未经授权的修改。在大多数应用中，大规模生产的器件无需修改配置存储器，即使在器件固件更新时也是如此。

功能

当配置为受保护时，整个 NONMAIN 区域都将被写锁定，并且在引导配置例程将执行传递给引导加载程序或 MAIN 闪存中的用户应用程序代码时在功能上不可更改。如果应用程序代码或引导加载程序尝试对 NONMAIN 进行任何编程或擦除，都会导致硬件闪存操作错误，并且扇区不会被修改。

静态写保护可防止应用程序代码或引导加载程序进行任何修改，但通过 SWD 接口发送的恢复出厂设置命令仍会被接受。如果不需要这种行为，可以使用唯一的密码来保护恢复出厂设置 SWD 命令，或者完全禁用该命令（请参阅 SWD 策略）。要完全消除任何修改 NONMAIN 配置存储器的方法，必须禁用恢复出厂设置命令和 TI FA（或 SW-DP）。

备注

当 NONMAIN 受到静态写保护并且禁用了恢复出厂设置命令和 TI FA（或 SW-DP）时，NONMAIN 相当于不可改变的只读存储器，并且不再能够通过任何方式更改器件配置。此外，如果任何 MAIN 存储器区域扇区配置了静态保护，则这些扇区也不能通过任何方式进行修改，可能会被视为不可更改。

2.6.1.3.3 验证应用 (MAIN) 闪存的完整性

BCR 支持在将执行从 BCR（位于 ROM 中）转移到用户应用程序（位于 MAIN 闪存中）之前，检查 MAIN 闪存中用户指定地址范围的数据完整性。

用途

完整性检查可用作额外的步骤，以确保在引导 ROM（通常是安全启动映像管理器）之后首先运行的代码具有与预期值匹配的 CRC/SHA256 摘要。此完整性检查降低了闪存中关键代码（可能负责验证其余用户应用软件映像）的任何意外损坏可能会造成安全漏洞的可能性。

功能

可以将起始地址、长度和 ISO-3309 CRC-32 或 SHA2-256 摘要配置到 NONMAIN 配置存储器中。在引导过程中，BCR 将计算 MAIN 闪存中指定范围的 CRC-32 摘要，并根据配置的（预期）摘要来验证计算的摘要。如果这些值匹配，则会启动用户应用程序。如果这些值不匹配，则不会启动用户应用程序，结果会导致灾难性的引导错误。

2.6.1.4 引导加载程序 (BSL) 安全策略

BSL 安全策略在调用时由引导加载程序解释，并包含以下参数：

- BSL 访问密码，如 节 2.6.1.4.1 所述
- BSL 读取策略，如 节 2.6.1.4.2 所述
- BSL 安全警报策略（篡改检测），如 节 2.6.1.4.3 所述

2.6.1.4.1 BSL 访问密码

对 BSL 的访问始终受到用户指定的 256 位密码保护。无法选择禁用密码。必须在调用后向 BSL 提供密码，才能获得权限来访问大多数的 BSL 功能。如果未提供密码，则允许的 BSL 命令只有*获取身份*和*启动应用程序*。

如果向 BSL 提供了错误的密码，BSL 会暂停 2 秒，随后可以再尝试发送正确的密码。在三次密码尝试失败后，安全警报功能将被激活（请参阅 节 2.6.1.4.3）。

2.6.1.4.2 BSL 读取策略

BSL 可选择支持出于调试和/或诊断目的读取器件存储器（在通过正确密码匹配获得 BSL 访问权限后）。默认情况下，为了安全起见，会禁用此功能，以防止他人从器件中提取敏感代码和/或数据。禁用 BSL 读取策略后，可通过 BSL 接口向主机提供的唯一信息是最小段长度为 1KB 的存储器段 CRC32 摘要。如果需要直接读取器件存储器，可在 BSL 配置中启用该功能。

2.6.1.4.3 BSL 安全警报策略

BSL 提供了一种警报机制，用于在怀疑发生篡改时采取措施。具体而言，如果在一个 BSL 会话期间 3 次将错误的密码传递给 BSL，则会激活安全警报，并且 BSL 可能会根据指定的安全警报策略以三种不同的方式之一进行响应：

1. 发出恢复出厂设置命令（擦除 MAIN 闪存并重置 NONMAIN 闪存区域）。
2. 禁用 BSL（保持 MAIN 闪存不变，但重新配置 NONMAIN 以阻止访问 BSL）。
3. 忽略（不修改配置并允许继续密码尝试）。

备注

选项 1 和 2 要求 NONMAIN 闪存区域未受静态写保护（请参阅 节 2.6.1.3.2）。

选择选项 1 时，配置为受静态写保护的任何 MAIN 存储器区域（请参阅 节 2.6.1.3.1）将不会在恢复出厂设置期间被擦除。

2.6.2 客户安全代码 (CSC) 安全策略

以下部分介绍由支持客户安全代码 (CSC) 在支持的器件上强制执行的策略。所有策略都在执行从 BOOTRST 发出的 CSC 期间进行配置，并且只能在器件上发出 INITDONE 信号之前进行修改。发出 INITDONE 并发生 SYSRST 后，策略将保持有效，并且在 BOOTRST 或 POR 之前无法修改。

2.6.2.1 CSC 强制存储体交换

在具有多个 MAIN 闪存存储体的器件上，可以支持系统中的两个版本的应用，每个存储体一个。在这种情况下，CSC 可以根据映像的版本和真实性选择运行其中一个或另一个。节 3 部分将进一步讨论决策过程。

多存储体 MSPM0 器件中的存储体交换功能指南可以在 MSPM0 系列的闪存多存储体功能中找到。

2.6.2.2 CSC 强制防火墙

支持 CSC 的器件包含可在器件上激活的多个不同防火墙：

- MAIN 闪存写保护防火墙 — INITDONE 后将不再是可写/可擦除的闪存的指定扇区。这以任何存储体写入/执行排除项为基础。
- MAIN 闪存读取-执行保护防火墙 - 应用程序无法读取或执行的指定闪存区域。读取此区域将返回全 0。
- MAIN 闪存 IP 保护防火墙 — 闪存或者数据总线无法读取但允许从 CPU 获取的指定闪存区域。使代码的特定可执行部分不可读，从而保护要读取的敏感算法。这以任何存储体写入/执行排除项为基础。
- DATA 闪存写保护防火墙 — 如果存在 DATA 存储体，则应用程序不会写入/擦除指定的扇区
- DATA 闪存读取保护防火墙 — 如果存在 DATA 存储体，则应用程序无法读取指定的扇区。读取此区域将返回全 0

“读保护”防火墙可用于隐藏应用程序的机密，“写保护”防火墙可用于将以后无法修改的信息传递给应用程序，因此可被视为可信。

在具有多个存储体的器件上，防火墙也会跨存储体进行镜像。这意味着，对于闪存大小为 0x4.0000 的双存储体器件（存储体从地址 0x0000000 和 0x2.0000 开始），0x5000-0x6000 的读取保护防火墙将从地址范围 0x5000 - 0x6000 和地址范围 0x2.5000-2.6000 返回全 0。

2.6.2.3 CSC 密钥写入 KEYSTORE

AES 密钥可安全地存储在 KEYSTORE 中。KEYSTORE 只能在发出 INITDONE 之前的 CSC 期间供访问（读取和写入）。在应用程序执行期间（在 INITDONE 之后），应用程序可以控制加载到 AES 引擎中的 KEYSTORE 密钥，但在整个加载过程中，应用程序不会看到密钥。有关更多详细信息，请参阅节 4.5。

2.6.3 配置数据错误抵抗

MSPM0 器件采用多种机制来降低因 NONMAIN 配置存储器中出现数据错误而导致安全性丧失的可能性。

2.6.3.1 由 CRC 支持的配置数据

NONMAIN 存储器中的 BCR 配置数据和 BSL 配置数据结构各自包含一个 CRC 值，该值对应于相应结构的 CRC 摘要。在器件启动过程中，BCR 将计算数据结构的 CRC 摘要，并将其与存储的 CRC 值进行比较，确认匹配后才会信任并使用这些结构中包含的数据。

BCR 配置 CRC 故障处理

如果 BCR 配置数据（包含 SWD 策略、BSL 启用/禁用策略以及闪存保护和完整性检查策略）在启动期间未能通过 CRC 检查，则会产生灾难性的启动错误并施加以下限制：

- 错误原因将作为引导诊断记录在 CFG-AP 中
- BSL 将不会被调用，即使它配置为要启用
- 用户应用程序不会启动
- 应用程序调试访问不会启用
- 如果启用了或使用密码启用了，则会执行待处理的 SWD 恢复出厂设置命令
- 如果已启用，则会执行待处理的 TI 失效分析流程条目
- 启动过程将最多重试 3 次
  - 如果第 2 次或第 3 次尝试通过，器件将正常启动
  - 如果第 3 次尝试仍未通过，则在下一次 BOR 或 POR 之前不会再进行启动尝试

此 CRC 校验的好处是，在启动过程中可以明确地检测配置数据中是否存在任何位翻转，例如静态写保护配置（安全启动的支柱）。故障处理程序会明确阻止 BSL 和用户应用程序运行，唯一受支持的选项（SWD 恢复出厂设置和 TI FA）受 16 位模式匹配字段保护。

BSL 配置 CRC 故障处理

如果 BSL 配置数据（包含 BSL 密码和 BSL 策略）在 BSL 调用期间未通过 CRC 检查，则会导致灾难性的启动错误并施加以下限制：

- 错误原因作为引导诊断记录在 CFG-AP 中
- BSL 不会被调用，即使它配置为要启用
- 用户应用程序不会启动
- 应用程序调试访问不会启用

- 启动过程最多会重试 3 次
  - 如果第 2 次或第 3 次尝试通过，器件将正常启动
  - 如果第 3 次尝试仍未通过，则在下一次 BOR 或 POR 之前不会再进行启动尝试

此 CRC 校验的好处是，在调用过程中可以明确地检测 BSL 配置数据中是否存在任何位翻转。故障处理程序会阻止 BSL 使用可能导致安全性丧失的无效数据启动。

TI 出厂修整数据 CRC 故障处理

除了用户指定的配置数据外，如果 TI 出厂修整在启动期间未能通过 CRC 检查，也会导致灾难性的启动错误并具有以下限制：

- 错误原因将作为引导诊断记录在 CFG-AP 中
- BSL 将不会被调用，即使它配置为要启用
- 用户应用程序不会启动
- 应用程序调试访问不会启用
- 如果已启用，则会执行待处理的 TI 失效分析流程条目
- 启动过程将最多重试 3 次
  - 如果第 2 次或第 3 次尝试通过，器件将正常启动
  - 如果第 3 次尝试仍未通过，则在下一次 BOR 或 POR 之前不会再进行启动尝试

2.6.3.2 16 位关键字段模式匹配

BCR 配置存储器中的关键策略（如 SWD 安全策略）会在 NONMAIN 存储器中实现为 16 位模式匹配字段，并具有以下特性：

- 需要精确的模式匹配，才能启用较低的安全状态
- 如果 16 位字段中的任何值与确切定义的模式不匹配，都会导致相应参数处于最高安全状态

这种行为可防止 single-bit 翻转导致器件进入比最初指定更低的安全状态。

3 安全启动

MSPM0 器件支持结合使用软硬件功能来对应用软件进行身份验证（安全启动）。虽然并非所有 MSPM0 器件都提供安全存储来保护对称密钥不受软件攻击，但这些器件都支持基于非对称和对称的身份验证方案。

MSPM0 架构包括实现安全启动所需的几个关键硬件特性：

- 可锁定闪存，用于存储固定的身份验证固件和身份验证密钥
- 引导期间的单点入口，可确保安全启动映像管理器始终是 BCR 之后运行的首个应用程序

MSPM0 软件开发套件 (SDK) 包括启动映像管理器 (BIM) 和客户安全代码 (CSC) 参考应用，用于在 MSPM0 MCU 上实现安全启动。此参考应用可轻松配置并预置到 MSPM0 器件中。

3.1 安全处理环境隔离

某些安全启动过程需要通过硬件机制将安全处理环境 (SPE) 与非安全处理环境 (NSPE) 隔离，并且应用程序固件上的任何更新均应由信任根 (RoT) 验证，以在执行之前立即检查完整性和真实性。

在 MSPM0 系列中，一些器件在硬件中提供此类隔离机制，以确保 CPU 在 INITDONE 之前在可信环境（特权状态）中执行，并在 INITDONE 之后在不可信环境（非特权状态）中执行。当程序在特权状态下（INITDONE 之前）执行时，CPU 具有以下权限，并且不允许在 INITDONE 之后更改这些配置。

- 在节 4.5 中设置 AES 密钥
- 设置存储体交换策略
- 针对数据写保护、读取执行保护或 IP 保护设置 节 4。
- 应用程序节 2.6.1.3.3。

备注

应用程序完整性和真实性验证与 INITDONE 硬件无关，但这是在 CSC 解决方案中的特权状态下实现的。

其他 MSPM0 器件不提供此类隔离机制，以便所有 MAIN 闪存程序都使用相同的权限执行。请参阅表 3-1，了解 MSPM0 器件的安全启动特性摘要（根据器件是否具有硬件隔离机制 (INITDONE)）。

表 3-1. MSPM0 安全启动特性比较

器件	MSPM0Gx10x、MSPM0Gx50x、MSPM0L130x	MSPM0L111x、MSPM0Lx22x、MSPM0Gx51x
INITDONE	否	是
安全启动解决方案	启动映像管理器 (BIM)	客户安全代码 (CSC)
密钥库	否	是
存储体交换	否	是
防火墙	否	是
CMAC	否	是
ECDSA+SHA256	受软件支持	受软件支持

3.2 客户安全代码 (CSC)

客户安全代码 (CSC) 是采用硬件隔离机制 (INITDONE) 的 MSPM0 器件的安全启动解决方案。图 3-1 展示了 CSC 引导和启动序列。在 BOOTRST 时，TI ROM 代码开始执行。成功引导后，引导代码会发出 BOOTDONE。此时，SYSCTL 向器件发出 SYSRST 以触发从 MAIN 闪存存储器执行。引导代码完成后，MAIN 闪存程序始终从物理地址 0x0004 向量（复位处理程序）开始。根据 NONMAIN 闪存 BCR 中的 CSCEXISTS 配置，BOOTDONE 之后有两个执行流程：

- CSCEXISTS 设置：CSC 引导序列启用，并且 MAIN 闪存程序以 INITDONE 在清除状态下启动。在这种情况下，用户需要将 CSC 固件（MSPM0 SDK CSC 示例）放入 MAIN 闪存 0x0000 地址。CSC 固件需要通过 NONMAIN BCR 配置进行静态写保护。
- CSCEXISTS 清除：不允许 CSC 引导序列，MAIN 闪存程序在置位状态下以 INITDONE 启动。任何与安全相关的策略都是不可配置的，在这种情况下，用户需要将应用程序固件放入 MAIN 闪存 0x0000 地址中。

备注

MAIN 闪存程序始终在 BOOTDONE 之后自物理地址 0x0004 开始。由于在 BOOTRST 期间会复位存储体交换策略，因此 MAIN 闪存程序始终在 BOOTDONE 之后启动而不会进行存储体交换。仅当在 NONMAIN 配置中同时启用 CSCEXISTS 和 FLASHBANKSWAPPOLICY 时，存储体交换才会在 INITDONE 之后生效。

对于 CSC 现有的情况，CSC 负责确定执行库、内存区域保护、将安全密钥初始化到 KEYSTORE 中、进行应用程序完整性和真实性验证等。该器件正在特权状态下工作，具有配置这些安全策略的权限。在 CSC 结束时会发出 INITDONE（通过写入 SYSCTL.SECCFG.INITDONE，请参阅特定于器件的技术参考手册以了解寄存器定义），然后 SYSCTL 会发出第二个 SYSRST，下面列出的所有安全策略在 INITDONE 期间生效，并且在下一个 BOOTRST 之前不修改：

- 防火墙保护策略
- 存储体交换策略
- 密钥库保护

INITDONE 之后，器件进入非特权状态，再次从 MAIN 闪存的地址 0x0004 开始执行，并且 CSC 会第二次执行。这一次，CSC 发现 INITDONE 之前已经发出过（这是通过读取 SYSCTL.SECCFG.SECSTATUS.INITDONE 位来确定的），因此会直接跳至主应用程序。请参阅图 3-2，了解特权状态 (INITDONE) 和非特权状态（INITDONE 之后）下的 CSC 执行流程。

有关引导和启动顺序的更多详细信息，请参阅 MSPM0 G 系列 80MHz 微控制器技术参考手册（修订版 C）的安全一章。

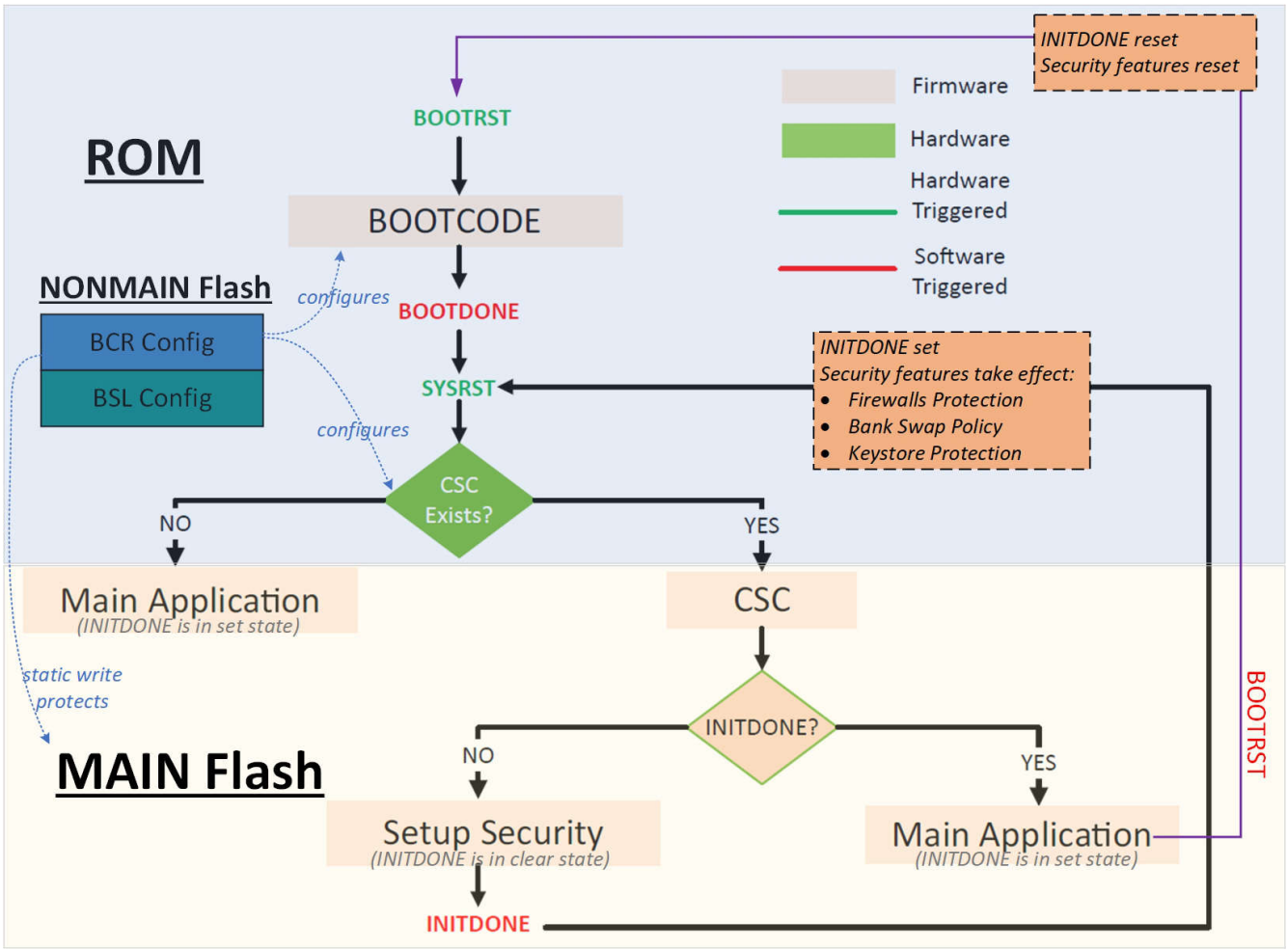

图 3-1. CSC 引导和启动序列

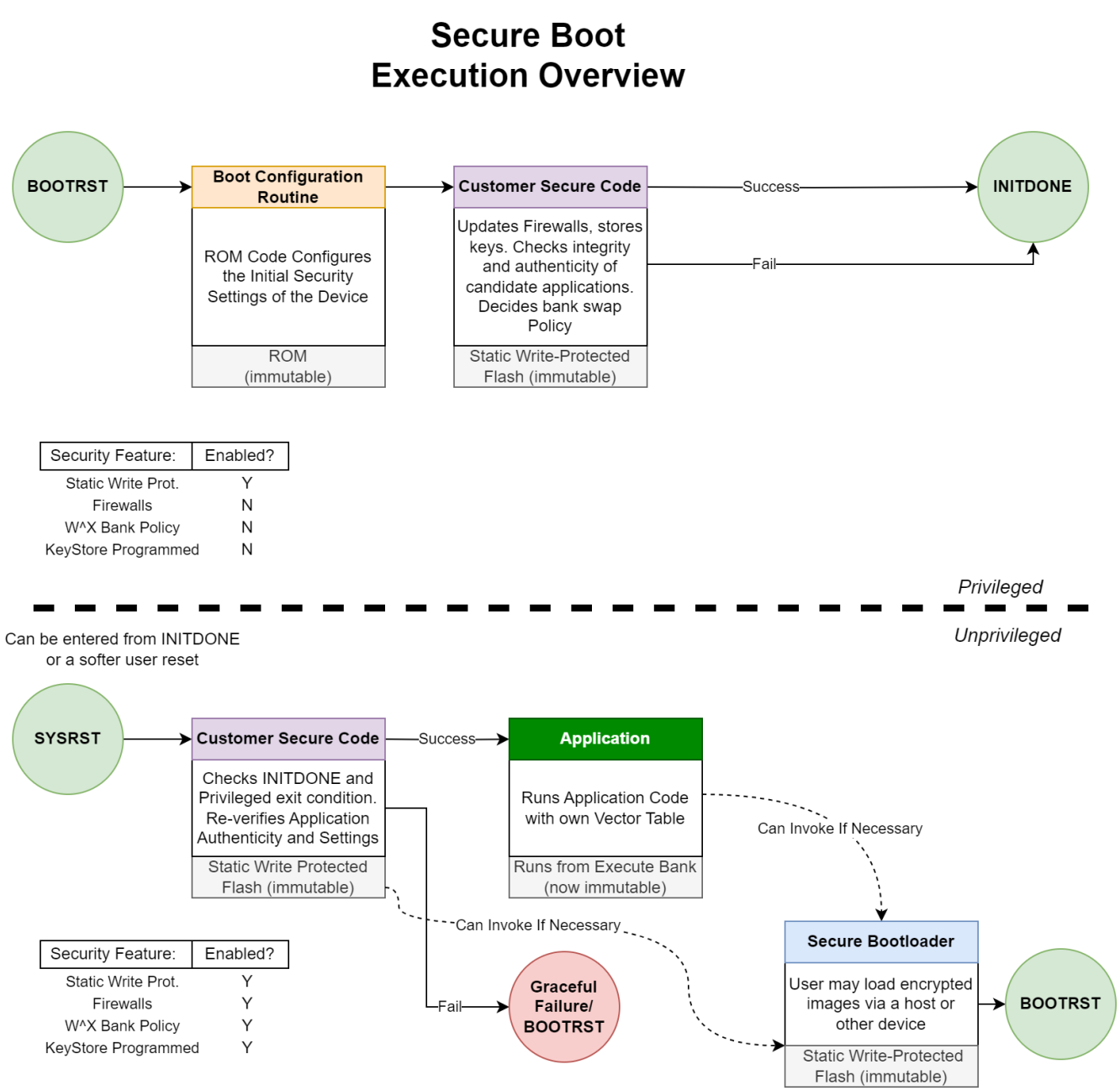

图 3-2. 客户安全代码 (CSC) 执行概述

3.2.1 安全启动流程

本节介绍基于 MSPM0 SDK CSC 示例 (MSPM0 SDK 2.08.00.03) 的 CSC 解决方案中的详细引导流程，如图 3-3 所示。整个执行流程与图 3-1 和图 3-2 中所示的流程图基本兼容。

完成 ROM 引导代码执行后，在程序首次进入 CSC 固件时，INITDONE (SYSCTL.SECCFG.SECSTATUS.INITDONE) 处于清除状态。CSC 首先在特权状态下工作。它从两个闪存存储体中搜索最高版本的映像，检查版本回滚，然后通过对称方法（硬件中的 AES-CMAC）或非对称方法（软件中的 SHA256+ECDSA）验证应用程序映像的权威和完整性。验证通过后，CSC 会更新回滚计数器、CMAC 标签、SECRET 密钥和 KEYSTORE。然后，它在 SECRET 闪存区域和可锁定闪存区域中配置防火墙，并确定存储体交换策略。CSC 发出 INITDONE 以触发 SYSRST，并且器件进入非特权状态。在置位状态下检查 INITDONE 后，器件会再次从 CSC 固件运行。检查前一个引导状态成功后，CSC 会跳转到应用程序映像以启动应用程序。

下面是一些与 CSC 示例执行流程相关的重要说明：

- 需要启用 NONMAIN 闪存中的 CSCEXISTS 和 FLASHBANKSWAPPOLICY 字段，才能启用整个 CSC 序列。
- PB0 表示物理 Bank0。由于存储体交换策略在特权状态 (pre-INITDONE) 下不会生效，因此在特权状态下使用的闪存地址 CSC 始终指物理地址。
- 如果 PB0 和 PB1 中的两个映像的版本相同，则会以更高的优先级验证并执行 PB0 映像。
- 如果最高版本的映像未通过 SHA256+ECDSA 验证，则会立即验证另一个存储体 (如果存在) 中的映像。
- 在非对称身份验证的情况下，首先在软件中计算应用程序代码的安全哈希 (SHA2-256) 摘要，然后软件 ECDSA 根据固件中的公钥验证映像签名。
- 对称 AES-CMAC 算法是一种省时的机制，用于在未检测到固件更新时验证应用程序映像。由于 AES-CMAC 是硬件加速的，因此简单地检查标签并确保其未经修改要快得多，因为它是不对称的验证。AES-CMAC 方法仅在发生 BOOTRST 时应用，并且自上次 BOOTRST 后，闪存中没有更高版本的映像放置。
- SECRET 闪存区域是用户指定的一个区域，用于存储机密信息并且由防火墙进行读取-执行保护。可锁定闪存区域是用户指定的区域，用于存储未修改的信息并由防火墙进行写保护。有关更多详细信息、请参阅闪存存储器映射。

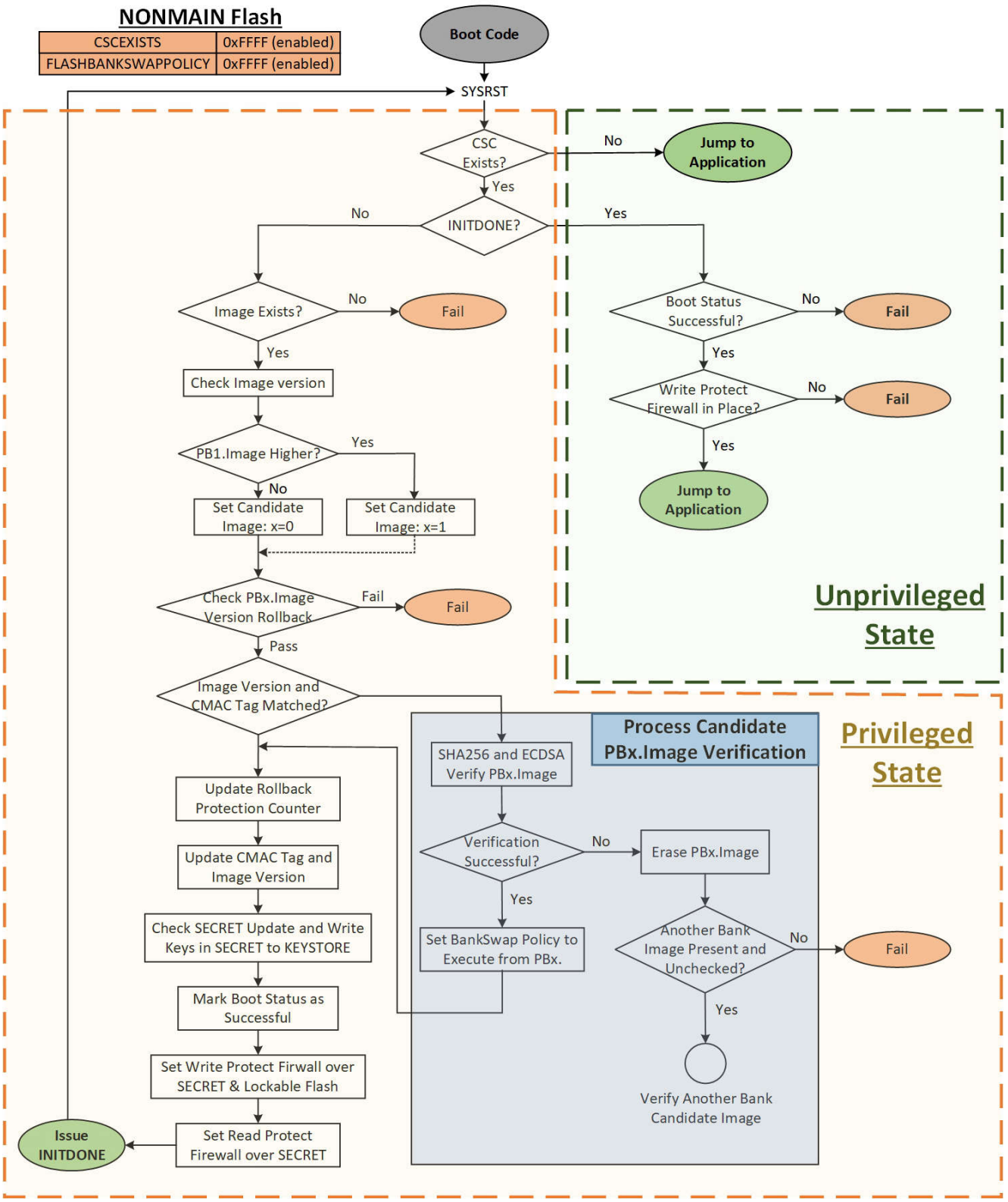

图 3-3. MSPM0 SDK CSC 执行流程

3.2.2 闪存映射

图 3-4 展示了 CSC 安全启动中的详细闪存存储器映射。以下是 CSC 中各个部分的说明：

- **SECRET**：SECRET 在特权执行流中可见，但将受到读取保护防火墙保护，从而使其对非特权流（CSC 和应用程序）的任何方面都不可见。SECRET 区域可用于存储应在运行时加载到 KEYSTORE 中的非易失性密钥。因此，非特权代码将能够使用这些密钥，但不具有读取访问权限。用户也可以自定义它以包含 CMAC 标签及密钥等其他信息。
- **可锁定闪存**：可锁定的闪存针对需要由特权状态写入并由非特权流读取但未修改的关键信息提供动态写保护。此区域中会到来的两个典型内容是安全计数器（回滚保护）、密钥库哈希表和映像哈希。请注意，可锁定内容将同时编程到闪存 bank0 和 bank1 中的 CSC 区域，以确保两个存储体应用程序能够以相同的方式访问该区域。
- **CSC 中断向量**：这些是客户安全代码的中断向量。如果发生 BOOTRST 或 SYSRST，该中断向量表将始终是从闪存运行的第一个内容。这是强制执行的，因为 VTOR 将在两个复位中被清除，这意味着将使用 0x0000（将指向逻辑组 0，其中存在不可更改 CSC 的副本）。
- **CSC 代码**：主代码和安全基元是客户安全代码的大量内容。这与中断向量一起在两个组之间重复。主存储体和辅助存储体上的映像应该相同，只是对代码的引用就像代码从 0x0000 运行一样。在存储体交换期间，在 INITDONE 触发 SYSRST 之后，程序将始终从逻辑存储体 0（逻辑 0x0000 地址）运行。FLASHCTL 将根据存储体交换策略配置将地址 0x0000 映射到 PB0 起始地址 0x0000 或者 PB1 起始地址 0x0000。

备注

在可交换存储体的配置中，防火墙保护会自动镜像到两个存储体。

下面是应用程序映像的各个部分：

- 映像标头、映像 TLV 和映像尾部信息：这些部分由 MCUBOOT 提供的签名工具 imgtool 生成（请参阅 *<mspm0_sdk_path>\source\third_party\mcuboot\scripts* 中的 python 脚步）。这些内容将在 CCS 编译后处理步骤中生成并合并至编译后的应用程序映像中。MSPM0 SDK 中有一个 customer_secure_sample_image 示例，展示了如何在 CCS 中构建已签名的映像。以下是这些图像部分的说明：
  - **图像标头**：应用程序映像的标头信息，包括标头魔数 (0x96F3B83D)、映像尺寸和映像版本。它位于应用程序中断向量之前的地址 0x100 字节（默认）。
  - **映像 TLV**：MCUBOOT 定义了包含映像元数据（放置在映像末尾之后）的类型长度值记录 (TLV)。MSPM0 CSC 中定义的 TLV 包括：TLV 魔数 (0x6907)、映像哈希、ECDSA 公钥哈希和 ECDSA 签名。要了解更多详细信息，请参阅 main · mcu-tools/mcuboot · GitHub 上的 mcuboot/docs/design.md。
  - **映像尾部信息**：一个位于图像闪存区域末尾的 16 字节魔数内容。

备注

只会对应用中断向量起始地址和图像 TLV 起始地址中的图像内容进行 SHA256 验证。

- **应用程序中断向量**：这些是应用程序使用的单独中断向量。在 CSC 跳转到应用程序期间，向量表偏移寄存器 (VTOR) 指向存储器中的该位置，因此未来所有中断都会发生，并且没有指向这组中断向量的复位链接。起始地址需要使用 32 字节对齐方式。

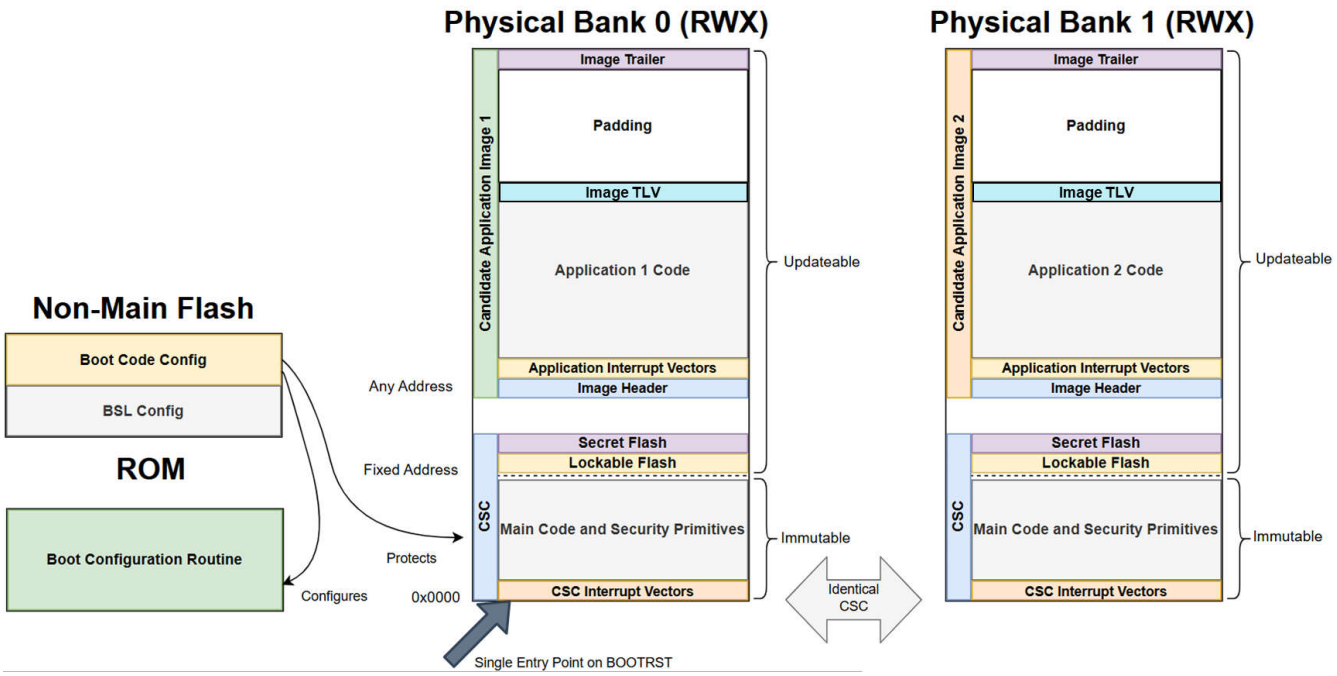

图 3-4. CSC 闪存映射

3.2.3 特性

3.2.3.1 CMAC 加速

CMAC（基于密码的消息身份验证代码）是一种用于验证数据完整性及真实性的加密算法。它的工作原理是使用 CMAC 密钥来生成 CMAC 标签，计算速度会因所处理映像的长度而异。

它利用 AES（高级加密标准）算法，当通过硬件加速实现时，CMAC 可提供高安全性和快速处理速度。这使得它特别适用于安全启动场景及需要高效消息身份验证的环境。

只有新映像需要完整且耗时的验证过程来验证其完整性及真实性。但是，如果验证的映像保持不变，我们可以利用在上次身份验证期间保存的状态信息。将 CMAC 与 AES 硬件加速结合使用，验证未更改映像的过程变得非常快速和高效，大大减少了安全启动所需的时间，并实现快速系统启动。

3.2.3.2 非对称验证

通过加密算法来验证新映像的完整性及真实性。只有已验证的映像才被视为安全映像并且可以执行。

SHA-256

SHA-256（安全哈希算法 256）是一种广泛使用的加密哈希函数，用于从任何输入消息生成固定长度的 256 位摘要。该算法旨在确保即使是输入消息的微小变化也会导致显著不同的输出哈希，从而确保对输入变化的高度敏感性。

其核心功能之一是防冲突，这意味着两个不同的消息极不可能产生相同的哈希值。此属性使 SHA-256 在验证数据完整性方面非常可靠，因为可以轻松检测到任何修改。

实际上，SHA-256 通常用于数字签名和数据完整性检查。通过将整个映像冷凝为独特的摘要，SHA-256 为下一个 ECDSA 算法奠定了坚实基础。

ECDSA

ECDSA（椭圆曲线数字签名算法）是一种基于椭圆曲线数学运算的数字签名加密算法。与 RSA 等传统算法相比，它提供了高安全性和更短的密钥长度，使其高效且适合资源受限的环境。

ECDSA 是目前唯一支持 MSPM0 安全引导的非对称身份验证方法。ECDSA 是非对称算法，意味着有独立的公钥和私钥。公钥将存储在器件闪存中，私钥将由开发人员安全保存。CSC 不提供安全私钥管理。

ECDSA 使用映像的哈希值和公钥来验证数字签名，从而确保数据的真实性。虽然此过程比对称加密选项慢得多，但它不会在器件上造成关键漏洞。

备注

为了保持最终部署的启动流程，务必要确保私钥的安全和管理，以便不能轻松地对映像进行签名。将密钥保留在本地共享驱动器上时，并非属于安全的位置！MSPM0 当前没有提供安全私钥管理。

表 3-2 给出了这两种替代方案之间的权衡。

表 3-2. 安全启动算法比较

参数	非对称 (SHA2 + ECDSA)	对称 (CMAC)
身份验证用时	较长，因为使用软件哈希计算和公钥算法	较短，因为算法简单并且能够在可用时利用硬件 AES 加速
代码大小	较大，因为使用 SHA 和 ECDSA 算法	较小，尤其是在目标器件上提供 AES 加速时
密钥完整性	公钥必须配置到器件中，并且必须是不可更改的	共享密钥必须配置到器件中，并且必须是不可更改的
密钥机密性	公钥没有保密要求，也不需要保护公钥免受应用程序代码漏洞的影响	共享密钥必须保密，并且在不使用时应打包，并应使用静态读取防火墙（如果目标器件支持）进行保护，以保护共享密钥免受应用程序代码漏洞的影响

在大多数情况下，TI 建议采用非对称实现方案。如果代码大小受限且/或身份验证用时必须保持最短，则可以使用对称实现方案，但必须谨慎管理共享密钥。并非所有器件都提供了安全存储 (KEYSTORE) 来保护共享对称密钥免受软件漏洞的影响。有关详细信息，请参阅平台信息安全机制。

3.2.3.3 KEYSTORE 及防火墙

KEYSTORE 是一个受保护的 SRAM 存储器，可以安全地存储 AES 密钥，在 INITDONE 之前在 CSC 中配置密钥，并且应用程序可以触发从 KEYSTORE 到 AES 引擎的密钥传输，但不会在 INITDONE 之后直接访问（读取或写入）这些密钥。

防火墙是一些闪存保护机制，包括闪存写入保护、闪存读取-执行保护和闪存 IP 保护，它们在 CSC 中进行配置并在 INITDONE 之后生效。

请参阅节 4 了解更多详细信息。

备注

在可交换存储体的配置中，防火墙保护会自动镜像到两个存储体。

3.2.3.4 CSC 性能

MSPM0 CSC 代码大小与编译器及优化级别相关。默认情况下，CSC 代码大小信息如下所列：

- CSC 区域大小：18KB
- CSC 主代码大小：13KB
- SECRET 大小：1KB
- 锁定存储大小：1KB

可以自定义 CSC 示例以修改主代码和 SECRET 或者 Lock Storage Size。如果要求更改 CSC 区域大小，则需要相应地更改应用程序固件的起始地址。如需了解详细信息，可参阅节 3.2.4.4。

可以在表 3-3 中检查不同算法的 CSC 计时性能。从该表中可以看出，基于硬件的 CMAC 对称方法比基于软件的 SHA256-+ECDSA 算法快得多，该算法可以在 MAIN 闪存中没有更新固件时提供 MCU 的高效启动。

表 3-3. CSC 时序性能

ECDSA 验证 (SW)	SHA256 (SW)	CMAC (加速器)
~1.9 秒 @32MHz	~ 5ms/千字节	~ 0.6ms/千字节

3.2.4 快速入门指南

本节根据本指南文档和无映像加密功能的 MSPM0 SDK customer_secure_sample_image 示例提供了简要分步指导。有关带映像加密功能的 customer_secure_image_with_bootloader 示例的指导，请参阅安全引导用户指南中 MSPM0 客户安全代码和引导加载程序 (CSC) 用户指南的“加载二进制映像”部分。

3.2.4.1 环境设置

要运行初始设置，请确保使用最新的 pip 包安装 Python 3.7 或更新版本，并运行下面的命令下载必要的要求。

1. 打开命令行窗口，并运行以下注释以检查您的环境中是否安装了 Python：

```
python --version
```

2. 进入 MSPM0 SDK 安装路径 (例如 *C:\ti\mspm0_sdk_2_08_00_03*) ，并在命令行中运行以下命令以满足必要要求：

```
python -m pip install --user -r source/third_party/mcuboot/scripts/requirements.txt
```

3. 在 *customer_secure_sample_image* 项目的后构建步骤中，这些 python 库由 *<mspm0_sdk_path>/source/third_party/mcuboot/scripts* 文件夹下的 python 脚本应用于对应用程序映像进行签名。

3.2.4.2 分步指导

如果设置得很好，请按照以下步骤在 MSPM0 SDK CSC 示例上进行实践 (以 MSPM0G351x 器件为例) ：

1. 从 SDK 路径 *<mspm0_sdk_path>\examples\nortos\<mspm0_device>\boot_manager* 将 customer_secure_code 示例和 customer_secure_sample_image 示例导入 CCS 工作区。
2. 编译 *customer_secure_code* 示例及 *customer_secure_sample_image* 示例。对于样本映像示例 (同时针对 EITHER_SLOT_BLUE 和 EITHER_SLOT_GREEN 配置进行构建) ，您可以看到工程中生成的相应调试文件夹。

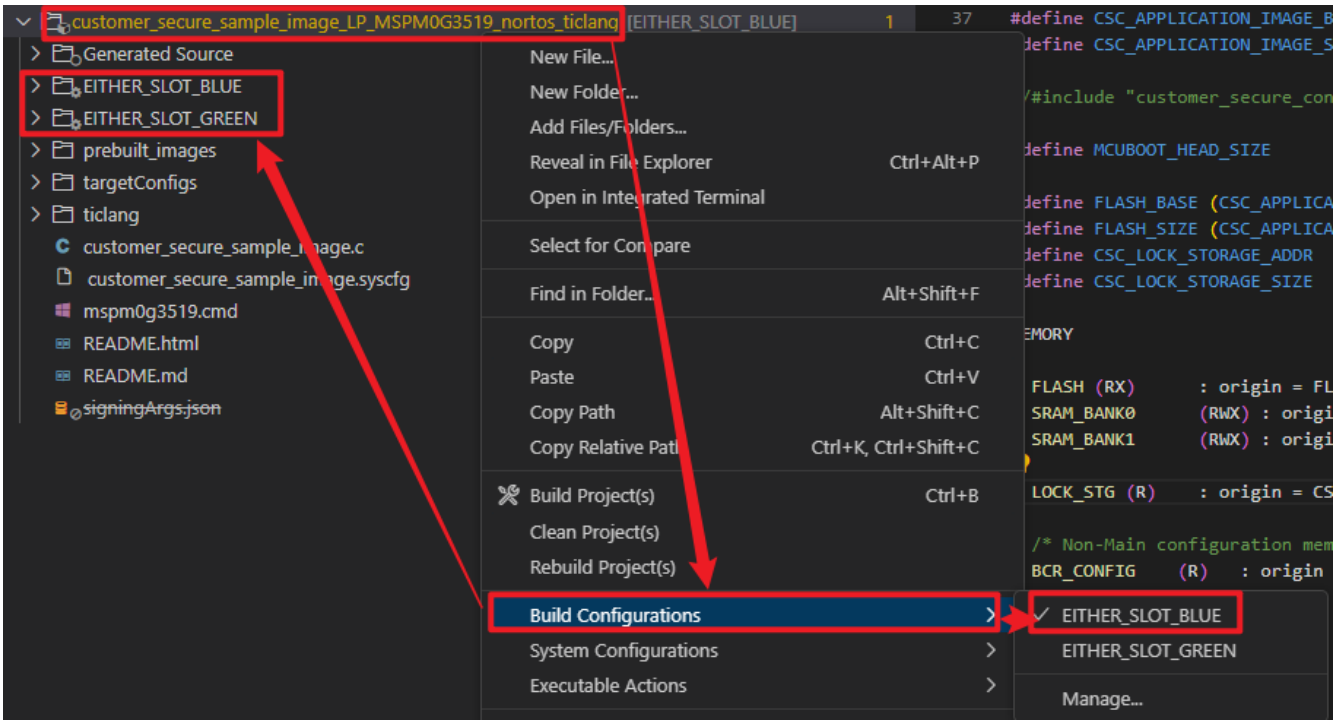

图 3-5. CSC 安全示例映像构建配置

3. 打开 UNIFLASH 软件编程工具 | TI.com 工具，将 PC 连接到 MSPM0 启动板并恢复出厂设置。

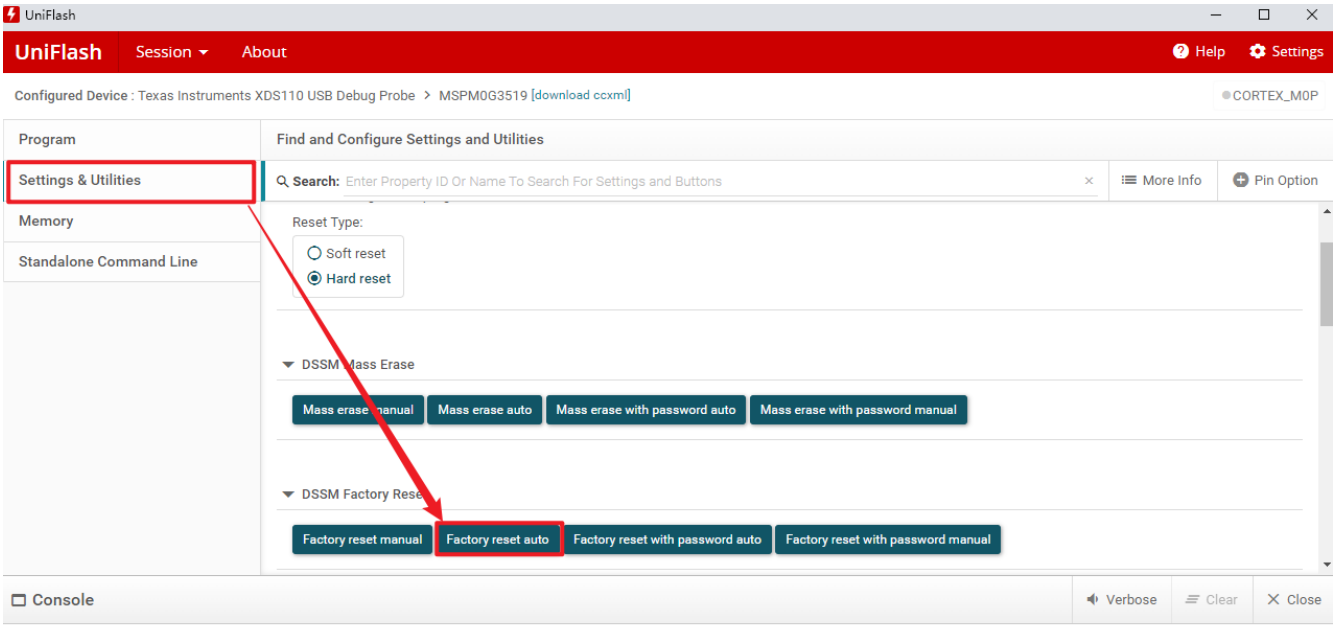

图 3-6. 通过 Uniflash 恢复出厂设置

4. 将闪存设置配置为**仅擦除 MAIN 和 NONMAIN 必要扇区**，该选项仅擦除映像输出文件中使用的闪存区域。

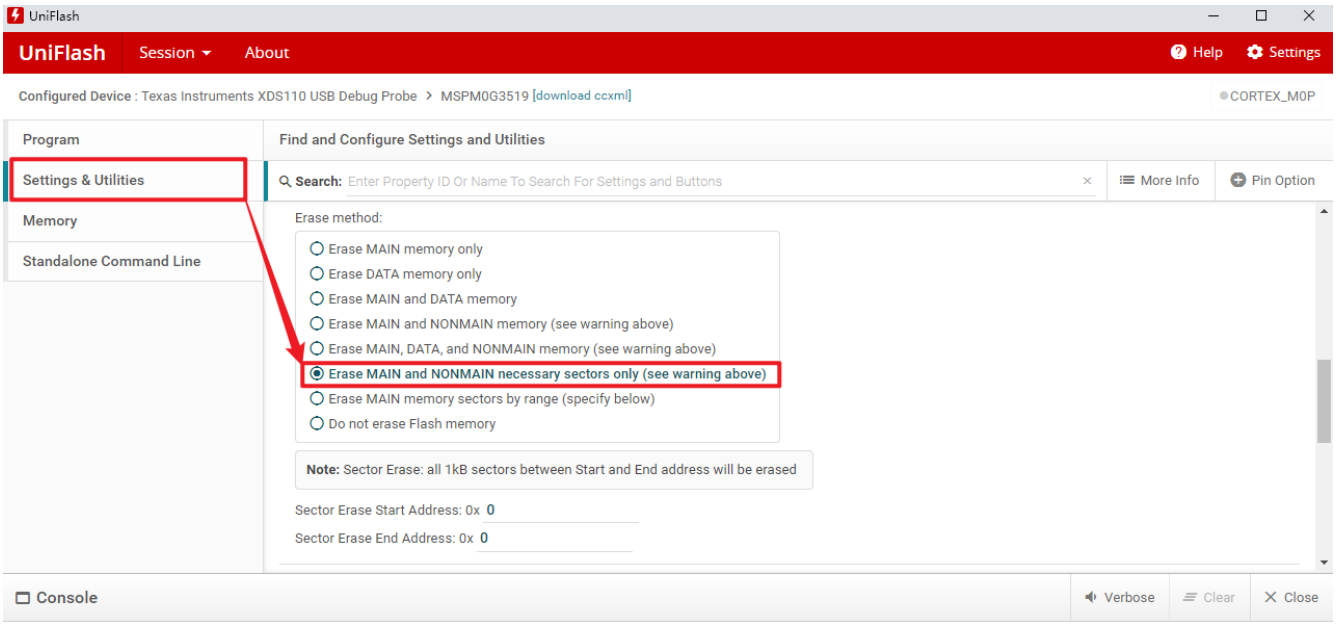

图 3-7. Uniflash 中的 CSC 闪存设置

5. 将以下文件加载到 MSPM0 器件中，然后对器件进行下电上电（或按启动板中的 NRST 按钮），您可以看到红色 LED 亮起 2s（CSC 执行以进行图像验证），然后蓝色 LED 闪烁（程序在物理 Bank1 中执行）。

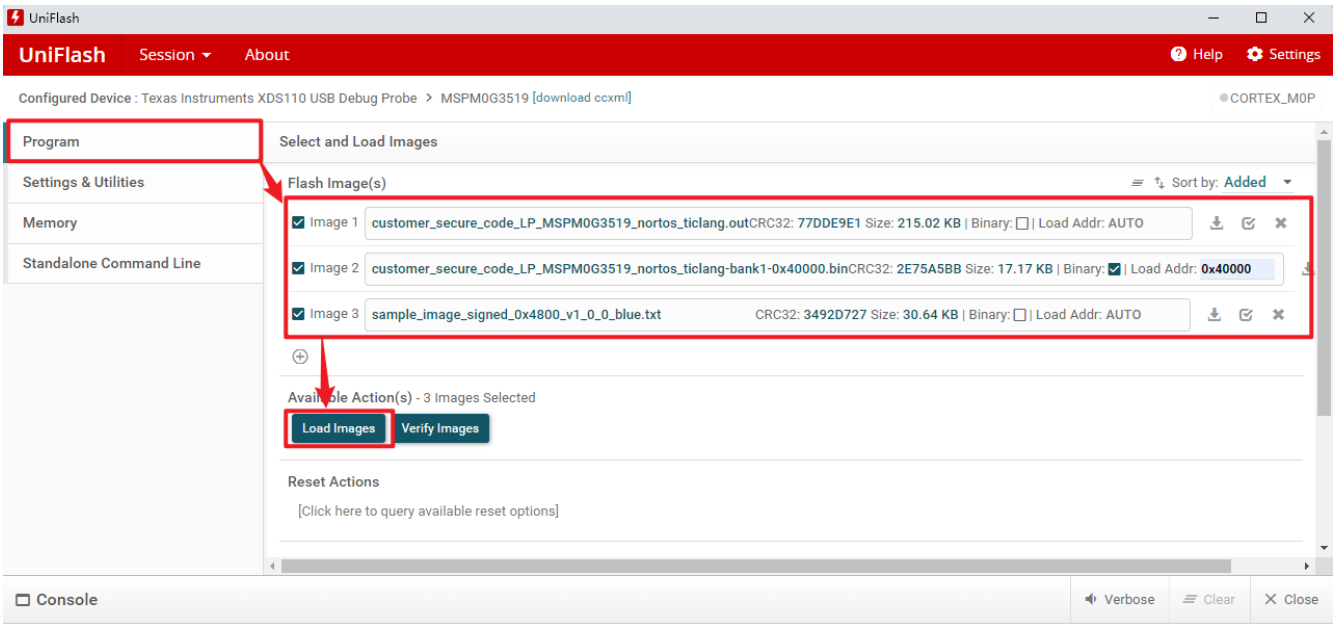

图 3-8. 通过 Uniflash 加载 CSC 映像

- *\customer_secure_code_LP_MSPM0G3519_nortos_ticlang\Debug\customer_secure_code_LP_MSPM0G3519_nortos_ticlang.out*：它包含用于物理 Bank0 和必要 NONMAIN 配置的 CSC 固件。
- *\customer_secure_code_LP_MSPM0G3519_nortos_ticlang\Debug\customer_secure_code_LP_MSPM0G3519_nortos_ticlang-bank1-0x40000.bin*：它是物理 Bank1 的 CSC 固件副本，应位于物理 Bank1 的起始地址（MSPM0G3519 器件为 0x40000）。
- *\customer_secure_sample_image_LP_MSPM0G3519_nortos_ticlang\EITHER_SLOT_BLUE\sample_image_signed_0x4800_v1_0_0_blue.txt*：这是应用程序映像，从物理 Bank1 0x44800 地址开始。

6. 使用 Uniflash 将更高版本的 GREED 应用程序映像更新到器件，然后按 NRST 按钮。您可以看到，在红色 LED 点亮 2s 后，绿色 LED 闪烁。CSC 不考虑映像如何加载到闪存中，此步骤仅展示了一种通过 Uniflash 将固件直接加载到 MCU 逻辑 Bank1 的方法。

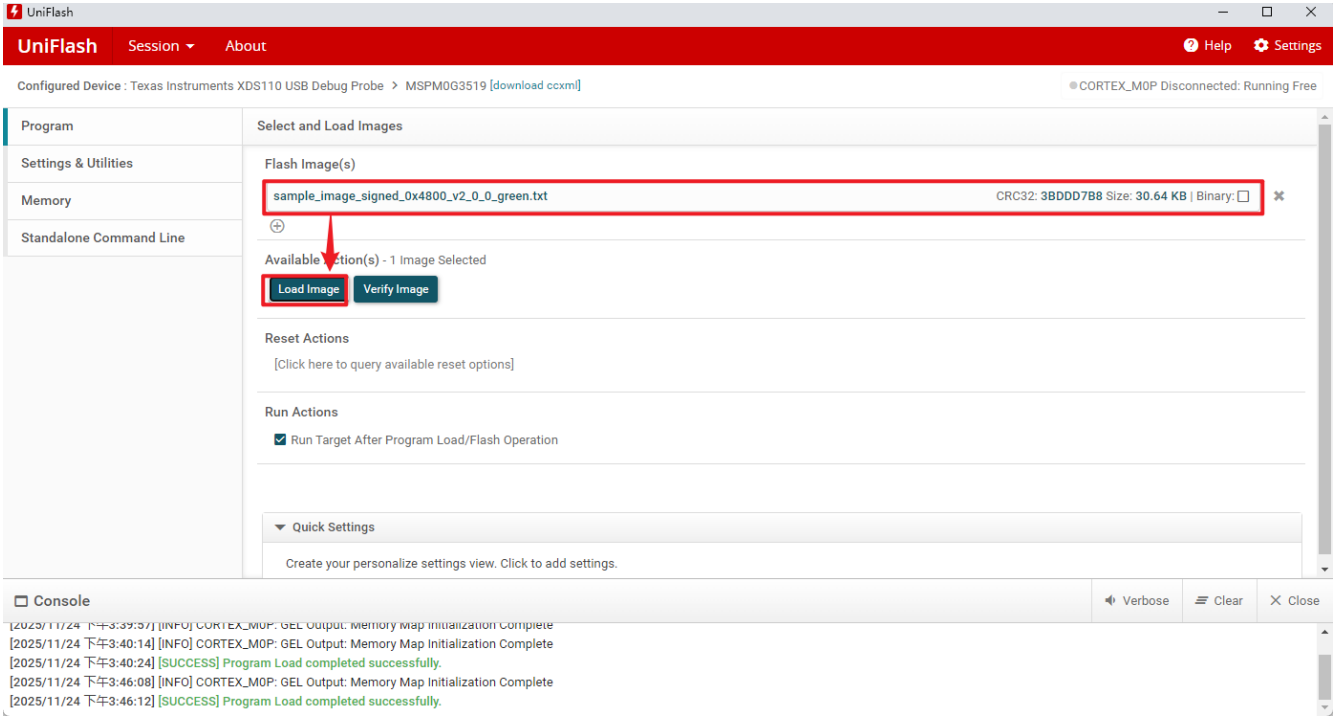

图 3-9. 通过 Uniflash 更新固件

3.2.4.3 CSC NONMAIN 配置

要启用 CSC 过程，需要一些 NONMAIN 配置。CSC 中有一些建议的 NONMAIN 配置：

1. **调试端口保护**：可在生产后选择完全通过密码禁用或启用 MSPM0 调试端口 (SWD)。提供了 256 位哈希密码来访问 CSC 器件（例如 MSPM0Gx51x、MSPM0Lx22x 及 MSPM0L111x）中的调试端口。有关详细信息，请参阅平台信息安全机制。
2. **CSC 静态写保护**：Bank0 和 Bank1 中的 CSC 固件在生产后需要进行静态写保护，以确保 CSC 区域不可更改和一次性可编程 (OTP)。
3. **NONMAIN 静态写保护**：由于 NONMAIN 包括静态写保护和调试访问的所有这些关键配置，因此 NONMAIN 内容本身也需要静态写保护，以防止应用程序进行擦除或写入操作。
4. **通过密码恢复出厂设置**：恢复出厂设置可以在访问 SWD 时将所有 NONMAIN 配置恢复为默认设置，并擦除 MAIN 闪存中的所有内容。如果用户不希望 MSPM0 恢复出厂设置，则可以使用密码启用恢复出厂设置。
5. **CSCEXISTS 及 FLASHBANKSWAPPOLICY**：CSCEXISTS 启用 CSC 引导序列，FLASHBANKSWAPPOLICY 启用存储体交换策略。在 SDK CSC 示例中启用了它们。

备注

配置 5 需要在 CSC 的开发阶段启用，只有在所有固件开发均已完成且器件将进入生产阶段时，才能启用配置 1-4。

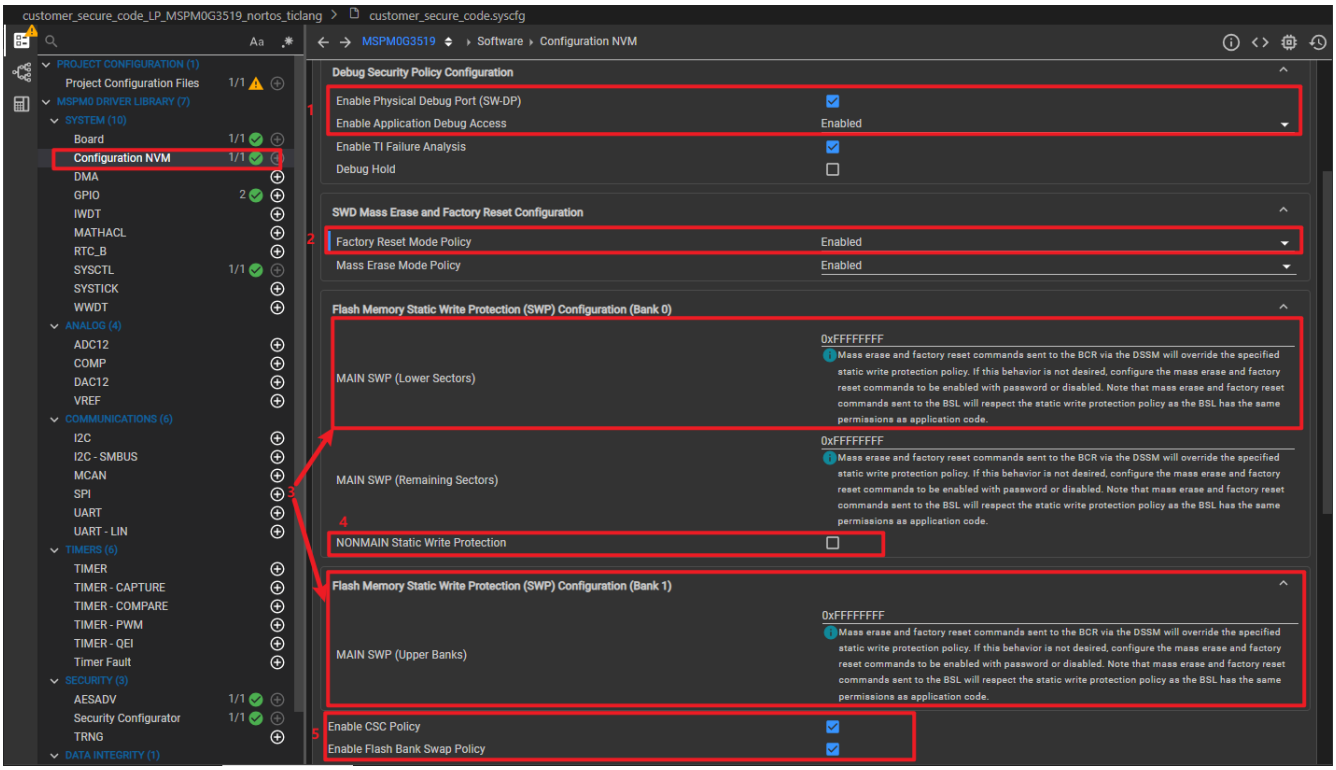

图 3-10. NONMAIN 安全配置

3.2.4.4 定制 CSC 示例上的更改

更改应用程序起始地址

本节介绍了 MSP0 SDK CSC 示例中使用的一些与闪存地址相关的参数，以帮助用户更好地了解如何更改应用程序起始地址。

请参阅图 3-11，图 3-12 左侧显示的参数是在 CSC 示例 Sysconfig 中定义的，所有这些参数都应该与 *customer_secure_code* 示例和 *customer_secure_sample_image* 示例链接器文件 (.cmd) 中的相应参数定义相同。

- CSC 锁定存储地址：此地址应定义成大于 CSC 代码大小。
- CSC 锁定存储大小：锁定存储区域的大小。
- CSC 机密地址：机密区域起始地址，就在锁定存储区域之后。
- CSC 机密大小：机密区域大小。
- CSC 应用程序映像基地址：映像标头的起始地址。应用中断向量将放置在此地址之后 0x100 字节处（图像标头大小）。
- CSC 应用程序映像大小：它应大于原始无符号应用程序代码大小 + 图像标头大小 + 图像 TLV 大小（CSC 示例中约 160 字节）。它确定映像尾部信息的放置位置，而未使用的区域将用 0xFF 填充。

备注

如果未在 CSC sysconfig 中启用“Security Configurator”，则将在 flash_mem_backend.c 文件中为不同的器件系列定义 CSC 地址和大小参数。用户需要更改此源文件以实现应用程序地址修改。需要对链接器文件和 signingArgs.json 文件进行相同的更改。

图 3-12 的右侧在 *customer_secure_sample_image* 示例中的 *signingArgs.json* 文件中定义：

- slotSize：它需要同 CSC 应用程序映像大小相同。
- 偏移量：它需要同 CSC 应用程序映像基地址相同。

如果用户想更改应用程序的起始地址（或任何其他地址，如机密或锁存储地址），他们需要同时修改 *customer_secure_code* 示例 sysconfig 文件和链接器文件、*customer_secure_sample_image* 示例链接器文件和 *signingArgs.json* 文件，以使修改有效。

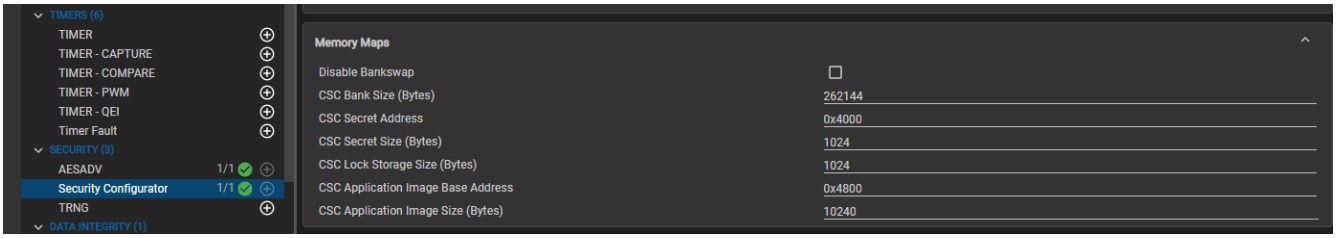

图 3-11. Sysconfig CSC 配置器

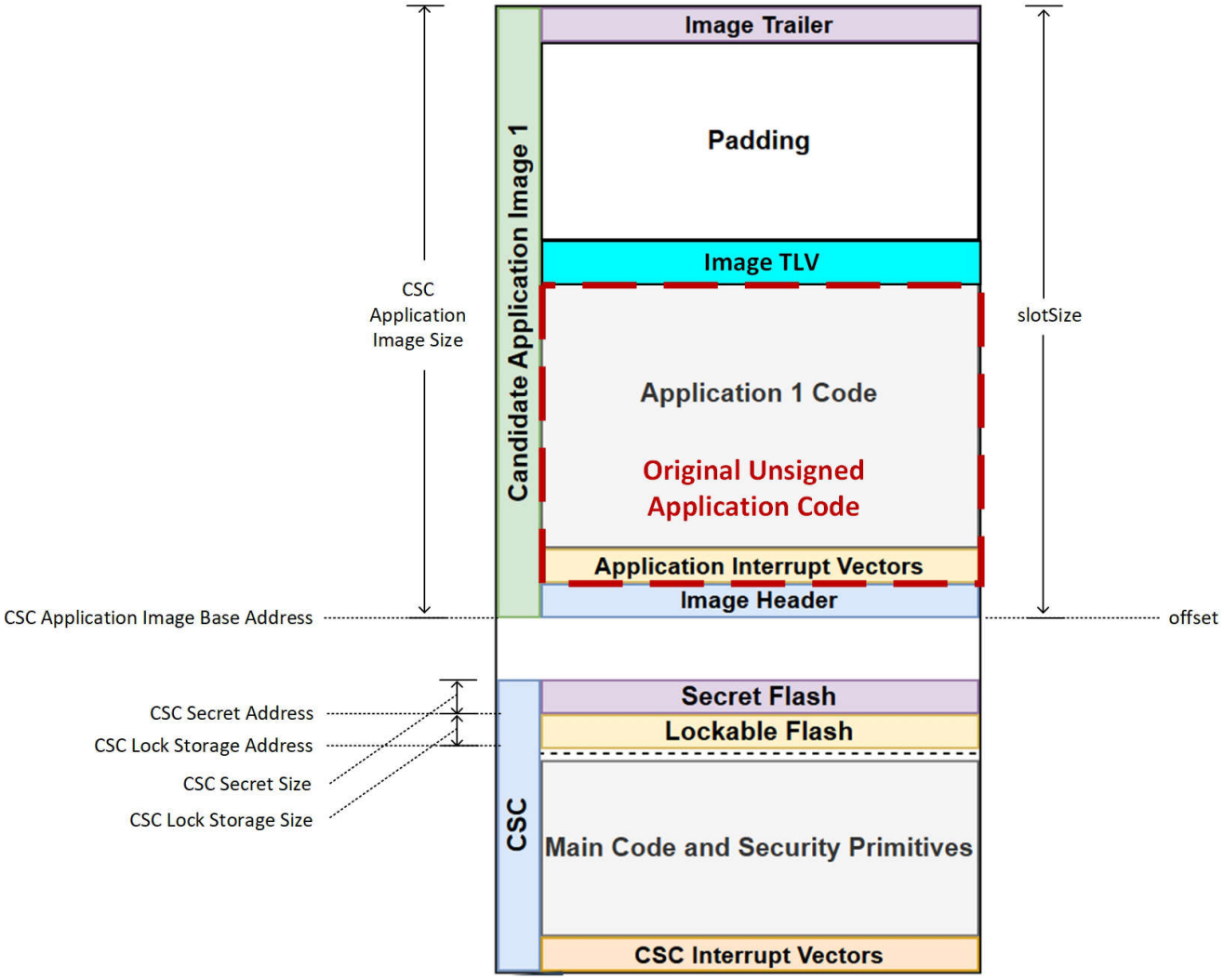

图 3-12. CSC 闪存映射参数

生成新的 ECDSA 密钥

请参阅安全引导用户指南中 MSPM0 客户安全代码和引导加载程序 (CSC) 用户指南的“使用客户安全代码进行开发”部分，了解如何通过 python 脚本创建新的 ECDSA 密钥并使用新密钥对应用程序映像进行签名。

3.3 启动映像管理器 (BIM)

启动映像管理器 (BIM) 是处理流程方面客户安全代码 (CSC) 的子集。由于该解决方案中没有将特权状态和非特权状态分开的硬件隔离机制，因此整个引导流程比 CSC 更简单，没有防火墙、KEYSTORE、存储体交换、CMAC 功能。有关 BIM 和 CSC 之间的比较，请参阅表 3-1。

3.3.1 安全启动流程

BIM 的安全启动流程可以在图 3-13 中看到。基于软件的非对称 SHA256 和 ECDSA 及 CSC 保持相同的功能和执行流程。并且在 BIM 内实现了以下安全启动配置步骤：

1. 必须配置启动映像管理器固件并将其编程到 MAIN 闪存中，其中复位矢量 0x0000.0004 指向启动映像管理器的开始
2. 启动映像管理器所需的任何身份验证密钥材料都必须编程到 MAIN 闪存中，与启动映像管理器相邻
3. 器件 NONMAIN 配置存储器必须使用以下特性进行编程：
   a. 包含启动映像管理器固件和密钥材料的 MAIN 闪存扇区必须配置为静态写保护以防止遭到修改。

b. NONMAIN 闪存扇区必须配置为静态写保护以防止遭到修改。
c. 建议使用密码保护或禁用批量擦除和恢复出厂设置命令（使用上述配置设置禁用恢复出厂设置将导致 NONMAIN 配置以及包含启动映像管理器和身份验证密钥的扇区永久锁定）。
d. 建议启用 MAIN 闪存完整性检查，并将地址范围设置为包括启动映像管理器和身份验证密钥。

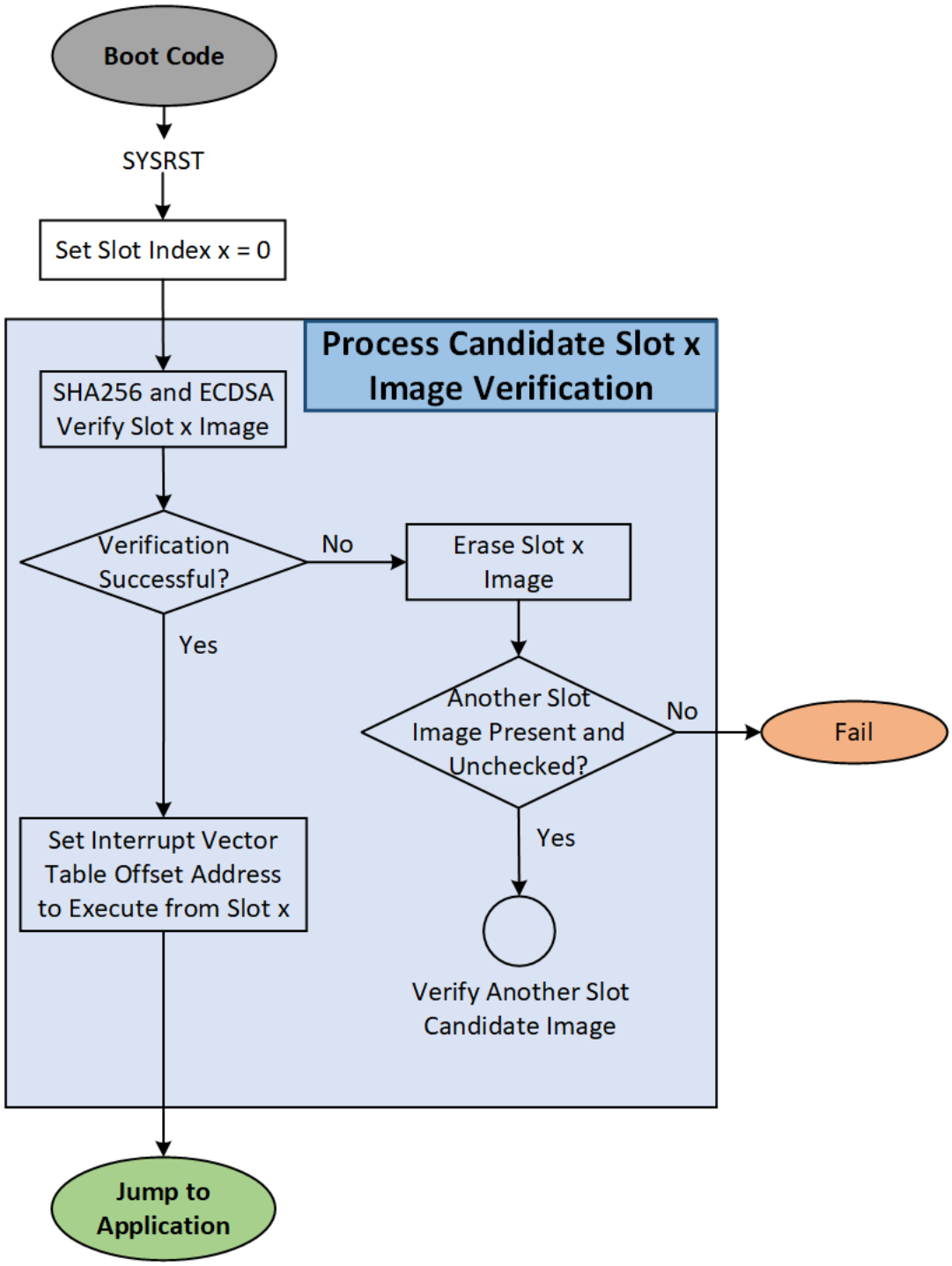

图 3-13. BIM 引导序列

3.3.2 闪存映射

可以在图 3-14 中看到 BIM 中的闪存存储器映射。由于此解决方案中没有存储体交换功能，因此应将两个应用程序映像映射到不同的闪存地址，这在工程链接器文件中指出。

如果需要修改应用程序映像地址，用户需要修改 flash_mem_backend.c 文件，以重新定义相应器件系列的存储器映射图中所示的地址和大小参数。

每个应用程序映像槽位包括映像标头、映像 TLV 及映像尾部信息，这些图片由 MCUBOOT 的签名工具 imgtool 生成。有关详细信息，请参阅闪存存储器映射。

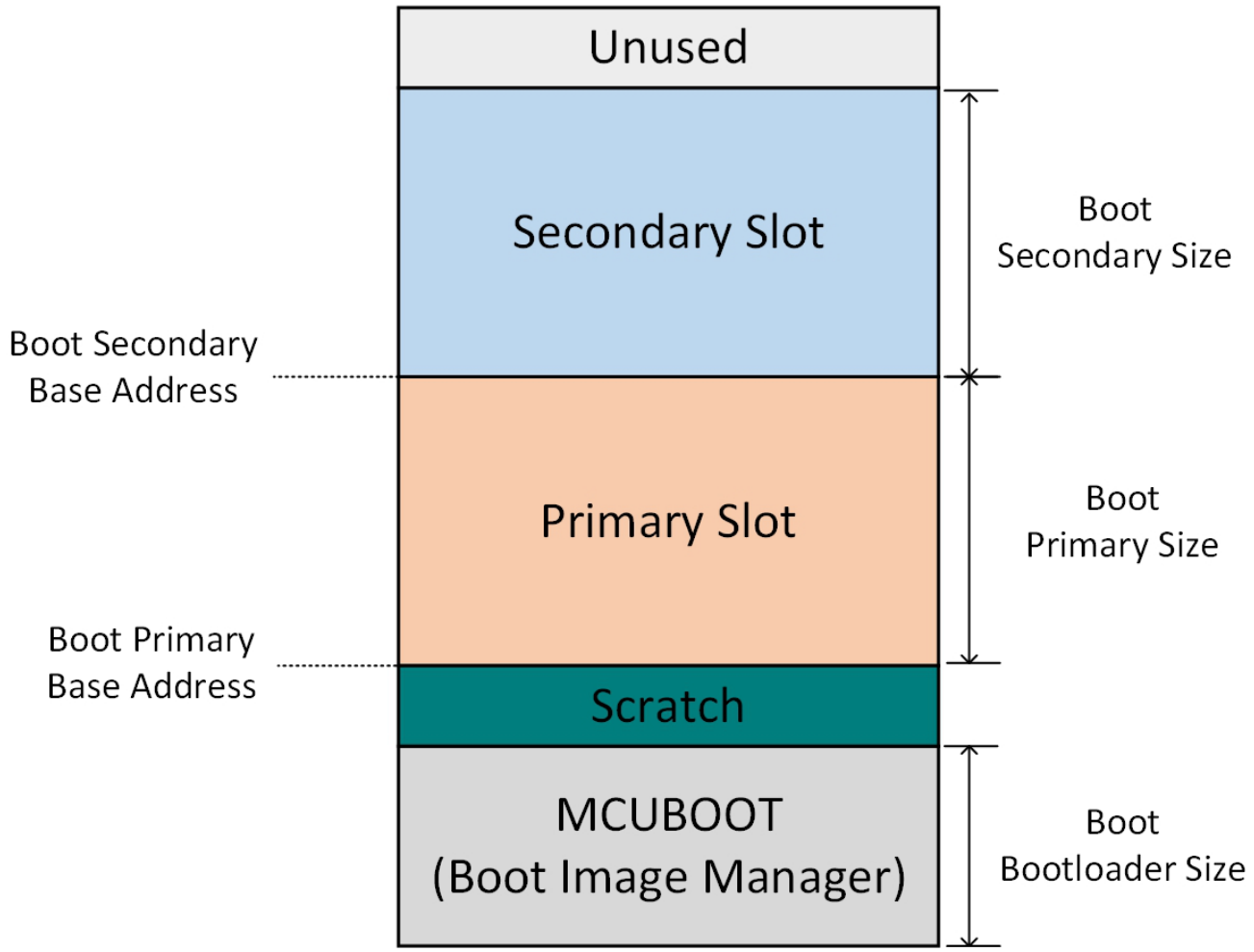

图 3-14. BIM 闪存映射

3.3.3 快速入门指南

用户可以通过参考 MSPM0 SDK 路径 *<mspm0_sdk_path>\examples\nortos\<mspm0_device>\boot_manager* 中的 *bim_sample_image* 示例和 *boot_application* 示例，并参考安全引导用户指南中的《MSPM0 引导映像管理器 (BIM) 用户指南》来开始评估 BIM 解决方案。

对映像签名需要相同的 python 环境，可以参考 CSC 指南在 BIM 示例上执行类似的操作步骤。有关详细信息、请参阅环境设置及分步指南。

4 安全存储

MSPM0 在闪存、AES 密钥和 SRAM 上提供各种类型的存储器保护机制，可满足多种安全要求。有关配置这些机制的详细信息，请参阅 MSPM0 G 系列 80MHz 微控制器技术参考手册（修订版 C）中的“安全”一章，并查看 MSPM0 器件系列有关这些特性的平台信息安全机制。

4.1 闪存写保护

提供了三个级别的写保护：

1. 由 TI 引导代码强制执行的静态写保护。它在 NONMAIN BCR 区域中配置，并在引导代码执行完成后生效。有关 CSC 引导序列，请参阅图 3-1。在需要将 CSC 视为信任根的扩展并且模式不可更改的情况下，此功能非常有用。
2. 由 CSC 强制执行的写保护，旨在进一步保护允许其更新但不应由应用程序修改的数据。此保护在 CSC 中配置，并在 INITDONE 之后生效。请注意，只有闪存的前 32KB 可配置为由 CSC 实施额外的写保护。在可交换存储体的配置中，这些保护会自动镜像到两个存储体。
3. 存储体交换上下文中的写保护。它也在 CSC 中配置，并在 INITDONE 之后生效。启用存储体交换后，逻辑下部存储体获得读取-执行权限并失去写入/擦除权限。另一个存储体（逻辑更高的存储体）可读以及可写但不可执行。

4.2 闪存读取-执行保护

闪存区域可配置为读取-执行保护，因此对该区域的读取和指令提取访问时将返回错误。CPU、DMA 和调试器访问均以相同的方式处理。

在可能需要此机制来防止重新执行 CSC 或防止从应用程序读取机密信息的情况下，此机制非常有用。

在可交换存储体的配置中，该保护会自动镜像到两个存储体。

4.3 闪存 IP 保护

可以将闪存存储器区域配置为读取保护：当允许指令提取访问时，对该区域的读取访问将返回错误。CPU、DMA 和调试器访问均以相同的方式处理。

如果需要阻止代码读取第三方供应商提供的软件 IP，此机制非常有用。请注意，需要进行 IP 保护的代码必须在编译时确保该区域不存在嵌入式数据访问（字面提取）。

在可交换存储体的配置中，该保护会自动镜像到两个存储体。

4.4 数据存储体保护

可将闪存 DATA 存储体的一个区域配置为读写保护，以阻止读取或写入，或同时阻止

这两种类型的访问。CPU、DMA 和调试器访问均以相同的方式处理。

只能以扇区 (1KB) 粒度保护 DATA 存储体的前 4KB。每个扇区可以采用以下保护方式：

- 读保护
- 写保护
- 两种
- 两者都不是

这种机制允许 CSC 在 DATA 存储体中保存机密数据或敏感数据，只有 CSC 可以在启动过程中读取/修改

这些数据。

4.5 安全密钥存储

需要保护密钥（非对称方案内的私钥）以确保机密性。只有受信任的代码才应配置一种机制来将密钥从闪存存储器安全地存入到只有加密引擎才能访问的位置（具体指 AES 加速器）。

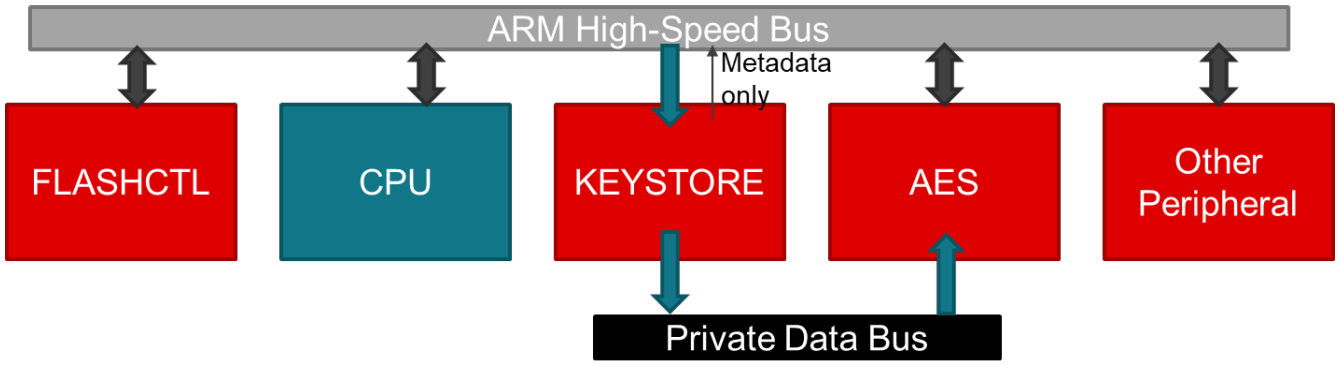

图 4-1. KEYSTORE 工作过程

只有 CSC 可在 INITDONE 之前将密钥配置为 KEYSTORE。随后，主应用程序可以配置加密引擎来使用其中一个存储的密钥，但其本身始终无法访问（读或写）任何存储的密钥。密钥从 KEYSTORE 到加密引擎的传输被安全地执行，对应用程序代码不可见。

KEYSTORE 内容将在每次 BOOTRST 时被擦除，并且存储将解锁以进行写入。KEYSTORE 内容不受 SYSRST 和低等级复位的影响。

4.6 SRAM 保护

缓冲区溢出是一种常见的漏洞来源，例如，损坏的返回地址可能导致执行过程跳转到恶意代码。为了缓解此类攻击，提供了 SRAM 代码保护功能，其中可以通过地址 A 将 SRAM 划分为两个区域：

- 区域 1（地址 < A）：读取-写入 (RW)，指令提取不可执行。
- 区域 2（地址>= A）：读取-执行 (RX)，不可写入。

A = 0 意味着整个 SRAM 为 RWX。这是 SRAM 的复位状态。

4.7 硬件单调计数器

一些器件提供了第二个 NONMAIN 扇区 (1KB)，访问时有以下限制：

- 始终不能被擦除（包括恢复出厂设置、批量擦除操作）
- 在 CSC 调用 INITDONE 之前只能被写入，之后只能被读取。

这种保护机制有助于实现基于硬件的单调计数器功能。由于该扇区只能被编程，因此它实际上可以用作非易失性递增计数器（或递减计数器，具体取决于软件对计数器值的解释）。CSC 可以在该扇区中维护修订版或回滚保护信息，确保低于计数器值的版本无法激活。请参阅平台信息安全机制，了解支持硬件单调计数器功能的器件。

5 加密加速

某些 MSPM0 MCU 提供针对高级加密标准 (AES) 的硬件加速功能，以及用于为加密生成真正随机数 (TRNG) 的硬件。请参阅特定于器件的数据表以确定器件是否具有 AES 加速器或 TRNG，或参阅*按子系列划分的安全推动因素*。

5.1 硬件 AES 加速

某些 MSPM0 器件包括针对高级加密标准 (AES) 的硬件加速。有两种类型的 AES 加速器，分别命名为 节 5.1.1 和 节 5.1.2，在 MSPM0 器件中定义，支持不同的功能集。有关 AES 和 AESADV 的比较，请参阅表 5-1。

请参阅特定于器件的数据表，以确定特定器件是否包含 AES 硬件加速。

表 5-1. AES 及 AESADV 的比较表

特性	基础 AES (AES)	高级 AES (AESADV)
ECB	√	√
CBC	√	√
OFB	√	√
CFB	√	√
CTR	√	√
CBC-MAC	√	√
CMAC		√
AES-CCM		√
AES-GCM		√

5.1.1 AES

AES 加速器模块根据高级加密标准 (AES) 在硬件中使用 128 位或 256 位密钥对 128 位数据块进行加密和解密。AES 是 FIPS PUB 197 中指定的对称密钥块加密算法。

AES 加速器的特性包括：

- AES 128 位块加密和解密
- DMA 触发器支持自动执行 NIST SP 800-38 中定义的 ECB、CBC、OFB 和 CFB 块加密模式
- 通过加密预先计算的（nonce || 计数器）块以及使用生成的密钥流加速纯文本异或，支持对 CTR 加密模式进行加速
- 支持对 CBC-MAC 标签计算（具有零初始化矢量的 CBC DMA 模式）进行加速
- 动态密钥扩展，用于加密和解密
- 用于解密的离线密钥生成
- 用于存储所有密钥长度的初始密钥的影子寄存器
- 8 位字节或 32 位字访问，以提供关键数据、输入数据和输出数据
- AES 就绪中断
- 在 RUN 和 SLEEP 模式下受支持（请参阅器件技术参考手册的工作模式部分）

AES 加速器硬件由 128 位状态存储器和相关的输入/输出寄存器、AES 加密/解密内核和控制逻辑、256 位 AES 密钥存储器和相关的输入寄存器组成。AES 加速器可对 128 位块进行快速加密和解密。表 5-2 中以块加密和块解密（使用预先生成的解密密钥）的周期和执行时间形式给出了 AES 加速器性能。

表 5-2. AES 硬件加速器关键性能指标

AES 密钥长度	加密			解密		
	周期	时间 (32MHz)	时间 (80MHz)	周期	时间 (32MHz)	时间 (80MHz)
128 位	168	5.25us	2.10us	168	5.25us	2.10us
256 位	234	7.31us	2.93us	234	7.31us	2.93us

5.1.2 AESADV

AESADV 加速器模块会加速硬件中基于 FIPS PUB 197 高级加密标准 (AES) 的加密和解密操作。

AESADV 加速器模块根据高级加密标准 (AES) 在硬件中使用 128 位或 256 位密钥对 128 位数据块进行加密和解密。AES 是 FIPS PUB 197 中指定的对称密钥块加密算法。AESADV 加速器的特性包括：

- AES 128 位块加密和解密
- 硬件中的密钥调度
- 仅加密/解密模式：CBC、CFB-1、CFB-8、CFB-128、OFB-128、CTR/ICM
- 仅身份验证模式：CBC-MAC、CMAC
- AES-CCM
- AES-GCM
- AES-CCM 和 AES-GCM 模式支持持续保持/恢复有效载荷数据
- 32 位字访问，提供关键数据、输入数据和输出数据
- AESADV 就绪中断
- 用于输入/输出数据的 DMA 触发器
- 在 RUN 和 SLEEP 模式下受支持（请参阅器件技术参考手册的工作模式部分）

AESADV 引擎包含一个执行加密/解密以及伽罗瓦域乘法的处理内核。该内核由软件通过存储器映射寄存器配置的配置和数据输入进行驱动。

AESADV 加速器可对 128 位块进行快速加密和解密。表 5-3 中以块加密和块解密（使用预先生成的解密密钥）的周期和执行时间形式给出了 AESADV 加速器性能。下表假设不存在导致引擎停转的系统开销（提供下一个输入或读取可用输出时出现延迟）。

表 5-3. AESADV 硬件加速器关键性能指标

AES 密钥长度	加密			解密		
	周期	时间 (32MHz)	时间 (80MHz)	周期	时间 (32MHz)	时间 (80MHz)
128 位	76	2.38us	0.95us	1.01us	2.38us	0.95us
256 位	81	2.53us	1.01us	81	2.53us	1.01us

5.2 硬件真随机数发生器 (TRNG)

某些 MSPM0 器件包含硬件真随机数生成器 (TRNG) 模块。TRNG 可用于轻松生成真随机种子值，这些值可用于播种确定性随机位发生器 (DRBG)。

TRNG 模块基于器件内部的模拟熵源来提供 32 位真随机输出。TRNG 本地提供了一个专用稳压器，用于防止电源操控攻击。

集成健康测试提供了 TRNG 模拟和数字组件的加电自检，并通过统计自检提供持续监控。

TRNG 适用于构建 TRNG + DRBG 系统，该系统可以通过用于加密随机数生成器的 NIST SP800-22 统计测试套件。图 5-1 中显示了 TRNG 的方框图。

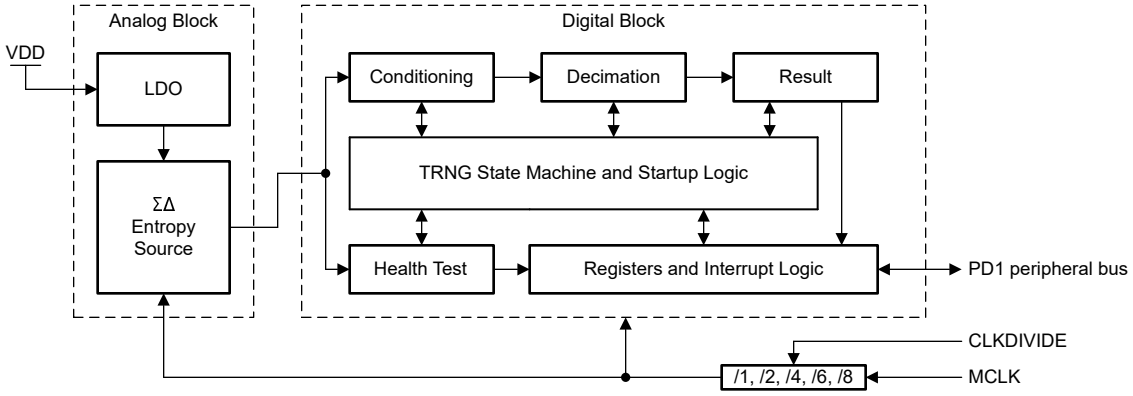

图 5-1. TRNG 方框图

有关 TRNG 操作的更多信息，请参考器件系列技术参考手册。

6 常见问题解答

1. **MSPM0 CSC 解决方案中的信任根是什么？**

   A：信任根包括不可变的 TI ROM 引导代码及静态写保护的 CSC 区域。它们在正确的 NONMAIN 配置后是不可更改的。有关详细信息，请参阅 CSC NONMAIN 配置。
2. **CSC 是否处理固件更新过程？**

   A：编号 CSC 仅验证已提前置于特定闪存地址的应用程序固件，但不处理固件更新过程（引导加载程序），也不考虑固件如何加载到闪存中。
3. **CSC 解决方案中不同算法的时序特性是什么？**

   A：有关详细信息，请参阅 CSC 性能。
4. **当我尝试将新固件下载到运行具有 CSC 或存储体交换的应用程序的器件时，为什么 CCS/Uniflash 报告擦除错误？**

   A：启用存储体交换后，逻辑低存储体将获得读取-执行权限并失去写入/擦除权限。另一个存储体（逻辑高存储体）可读以及可写但不可执行。当 CCS 或 Uniflash 尝试从地址 0x0000 下载固件时，由于逻辑低存储体地址不可擦除，它将报告擦除错误。您可以从高位存储体地址开始更新固件，或者只是在加载程序之前恢复出厂设置。
5. **下载 CSC 示例后，为什么需要下电上电或 NRST 复位？**

   A：CSC 示例包括用于启用 CSC 的 NONMAIN 配置。NONMAIN 配置在引导代码期间生效，只有 BOOTRST（或更高级别）复位可以使 MSPM0 返回到 ROM 引导代码。
6. **如何在 CSC 中更改应用程序起始地址？**

   A：有关详细信息，请参阅在 CSC 上自定义更改示例。
7. **对于 CSC、应用程序映像和 NONMAIN 区域输出，我应该选择哪种输出格式？**

   A：.txt/.bin/.hex 格式可用于固件更新。NONMAIN 配置应与 CSC 一起编程，不会与应用程序固件一起更新 NONMAIN 区域。有关指导，请参阅逐步指导。
8. **构建 customer_secure_sample_image 时，CCS 为什么报告编译后失败错误？**

   A：在构建 CSC 样本映像示例之前，请检查您是否已成功设置 Python 环境。并确保 CSC 示例与 CSC 样本映像示例位于同一工作区。
9. **应用程序中是否存在用于非对称加密/解密的密钥存储区域？**

   A：MSPM0 器件仅对 AES 引擎提供对称的密钥存储 (KEYSTORE)。非对称加密/解密算法密钥（如 ECDSA 公钥）存储在 SECRET 区域中。此 SECRET 区域只能在特权状态（INITDONE 预置）下访问、并且在运行应用程序时通过防火墙受读保护和写保护。
10. **我如何在链接器文件中提供应用程序地址？**

    A：在启用存储体交换的配置中，应始终使用给定的逻辑低存储体地址构建应用程序。以 MSPM0G3519 为例，链接器文件（CCS 中的.cmd）中定义的地址范围应为 0x00000~0x40000。但是，在更新应用固件时，需要将固件加载到逻辑高存储体地址，因为程序在逻辑低存储体中运行，只有逻辑高存储体具有读写访问权限。
11. **如果我要定义自己的应用程序映像格式该怎么办？**

    A：当前，SDK CSC 示例使用 MCUBOOT 提供的签名工具 imgtool 来生成包含接头和签名信息等的应用程序映像。用户可以定义自己的映像格式，但需要为自己定义的映像格式实现 CSC 中的解析程序。
12. **如果我只想将对称方法用于安全启动，该怎么办？**

    A：用户可以使用 AES-CMAC 在支持 AESADV 的 MSPM0 器件中进行对称映像验证，有关详细信息，请参阅平台信息安全机制。确保存储在 MCU 中的 AES 密钥已经事先通过安全的方式与映像供应商一致。

7 总结

MSPM0 MCU 中提供的信息安全机制为希望向新应用添加更多网络安全功能的 MCU 客户提供了独特的功能和价值组合。以同等价位提供的独特功能（例如密码验证的应用调试、批量擦除和恢复出厂设置）支持各种开发和生产用例，同时保持配置简单明了。

8 参考资料

- MSPM0 G 系列 80MHz 微控制器技术参考手册
- MSPM0 L 系列 32MHz 微控制器技术参考手册（修订版 E）
- MSPM0 C 系列 24MHz 微控制器技术参考手册（修订版 C）
- MSPM0 H 系列 32MHz 微控制器技术参考手册
- MSPM0 系列中的闪存多组特性
- MSPM0 客户安全代码及引导加载程序 (CSC) 用户指南
- GitHub - mcu-tools/mcuboot：32 位微控制器的安全启动！

9 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 JANUARY 31, 2023 to DECEMBER 31, 2025 (from Revision * (January 2023) to Revision A (December 2025))	Page
• 更新了 MSPM0 MCU 平台信息安全机制表格。	3
• 更新了 MSPM0 安全启动流程及实现。	17
• 更新了 MSPM0 安全存储特性。	32

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