User's Guide

MSPM33C 安全用户指南

TEXAS INSTRUMENTS

摘要

本文档可作为在应用开发期间使用 MSPM33C 器件系列中内置安全功能的指南。该功能包括 TrustZone® 及多个安全加速器。有关实现安全启动或安全调试的详细信息，请参阅 M33 TRM。

内容

商标

1 安全功能概述

MSPM33C 器件系列中包含的安全功能创建了一个框架，用于保证器件上部署的任何安全数据的完整性和真实性。

MSPM33C 器件系列内包含的主要安全特性包括：

- 安全执行环境
  - TrustZone®
    - 实施依赖归属单元 (IDAU)
    - 安全归属单元 (SAU)
  - 存储器保护单元 (MPU)
    - TrustZone® 及 MPU
  - 全局安全控制器 (GSC)
    - SRAM 保护控制器 (SPC)
    - 闪存保护控制器 (FPC)
    - 外设保护控制器 (PPC)
- 安全加速器
  - 高级加密标准 (AES)
  - 密钥库
  - 安全哈希算法 (SHA)
  - 公钥加速器 (PKA)
  - 后量子加密 (PQC)

1.1 术语

缩略词/术语	定义
GSC	全局安全控制器
PKA	公钥加速器
SRAM	同步随机存取存储器
MMR	存储器映射寄存器

2 安全执行环境

TrustZone® 架构通过 Arm® Cortex®-M33 启用。TrustZone® 允许编程器对存储器的某些部分进行安全的非安全分区，从而允许编程器限制未经授权的用户访问存储器的某些部分。此外，MSPM33C 器件还包含一个全局安全控制器 (GSC)，它可以在 CPU 子系统 (CPUSS) 之外提供额外的安全性。这可防止 DMA 等外部外设在不安全的处理器状态下访问存储器的安全区域。

2.1 TrustZone®

Cortex®-M33 TrustZone® 安全性可让编程人员将存储器划分为三种不同类型的区域：安全、非安全和非安全可调用。存储器的安全区域用户可以毫无问题地访问所有存储器区域。非安全存储器区域限制编程器，从而其仅能访问其他非安全或非安全可调用存储器区域。非安全可调用存储器区域是唯一的，在这里它们仅限于访问非安全存储器区域，除非用户调用允许编程器跳转到安全存储器区域的 SG 指令。有关更多详细说明，请参阅用于 Armv8-M 架构的 TrustZone® 技术文档。

为了定义哪些存储器区域是安全的，哪些是不安全的，使用了安全性归属单元 (SAU) 和实施定义归属单元 (IDAU)。IDAU 设置在 RTL 级别并且由 TI 配置。有关如何配置 IDAU 的信息，请参阅 MSPM33C3 系列 160MHz 微控制器技术参考手册的 CPU 部分。

CPU 子系统 (CPUSS) 使用 SAU 将存储器区域定义为安全和非安全。此外设允许编程器将他们的代码划分为安全和非安全。有关 SAU 和 IDAU 如何协同工作以将区域定义为安全或非安全的信息，请参阅用于 Armv8-M 架构的 TrustZone® 技术文档。

2.1.1 实施定义归属单元

实施定义归属单元 (IDAU) 由 TI 定义并在 MSPM33C3 系列 160MHz 微控制器技术参考手册的 CPU 部分中引用。了解 IDAU 的配置方式非常重要，因为 IDAU 的配置将与 SAU 一起用于配置区域的最终安全级别。例如，在 MSPM33C32 器件中，IDAU 配置为当地址的第 8 位为 1 时，存储器区域定义为非安全可调用，而当它为 0 时则定义为唯一非安全区域。例如，地址 0x1000.0000 定义为 NSC。

IDAU 及 SAU 均用于确定最终安全级别。两者中的最高安全级别设置为最终归属的安全级别。有关 MSPM33C321A 上的相关示例，请参阅图 2-1。

Memory Location	IDAU	SAU	Final Security Level
0x0000.0000 to 0x00FF.FFFF	Non-Secure	Non-Secure	Non-Secure
0x1000.0000 to 0x10FF.FFFF	Non-Secure Callable	Non-Secure Callable	Non-Secure Callable
0x2000.0000 to 0x203F.FFFF	Non-Secure	Secure	Secure
0x3000.0000 to 0x303F.FFFF	Non-Secure Callable	Non-Secure	Non-Secure Callable
0x4000.0000 to 0x4FFF.FFFF	Non-Secure	Non-Secure callable	Non-Secure callable
0x5000.0000 to 0x5FFF.FFFF	Non-Secure Callable	Secure	Secure

图 2-1. MSPM33C321A 配置示例上的 IDAU 及 SAU

2.1.2 安全性归属单元

编程器可以使用安全性归属单元 (SAU) 来定义存储器安全性的区域。最终安全级别同时使用 SAU 配置及 IDAU 配置。为了确定存储器安全级别的一部分，SAU 和 IDAU 之间具有更高的安全级别。例如，如果 SAU 将区域定义为安全，而 IDAU 将区域定义为非安全，则将区域定义为安全。

在 MSPM33 器件上，SAU 旨在支持将 8 个区域定义为安全、非安全或非安全可调用。

将区域定义成安全区域

SAU 的一个主要用例是将一个区域定义成安全或非安全区域。为了定义区域，通过写入 SAU 的寄存器执行以下步骤

1. 通过 SAU->RNR 寄存器选择要配置的区域
2. 使用 SAU->RBAR 寄存器选择区域基址
3. 设置 SAU->RLAR 以配置区域的大小
4. 对所有区域重复步骤 1 - 3
5. 配置 SAU->CTL 寄存器以传播全部已配置的区域

2.1.3 TrustZone® 软件开发

要在编程器代码开发中集成 SAU，我们建议使用 CMSE 库。该库由 Arm® 设计，可让用户轻松将函数定义为安全或非安全函数。如需更多有关此方面准则的详细信息，请参阅 ARMv8-M 安全软件指南。

ARMv8-M 指令使用 BXNS（Branch 和 Exchange Non-secure）和 SG（安全网关），在非安全和安全代码之间进行转换。BXNS 用于从安全代码转换到非安全代码，而 SG 用于从非安全代码转换到安全代码。SG 命令利用 NSC 区域进行此转换。为了轻松地将这些指令运用到编程环境中，CMSE 库将这些函数集成到函数调用中。

将函数定义为安全及非安全

CMSE 库的主要特性之一是它允许编程器将函数定义为安全或非安全。这决定了允许编程器在安全及非安全区域之间跳转的 CPU 状态。要将 CMSE 库添加到程序中，请使用以下标头。

```
#include <arm_cmse.h>
```

若要在代码中实现 SG 命令，可以为函数提供 cmse_nonsecure_entry 属性。该函数定义在代码开头调用安全网关 veneer。这对于定义函数非安全代码可以访问非常有用，其中使用安全存储器中的信息。

```
__attribute__((cmse_nonsecure_entry)) secure_fxn()
{

}
```

若要从安全代码中调用非安全函数，可以使用 BXNS 函数。为了使用 CMSE 库轻松将其集成到编程环境中，您可以为函数提供 cmse_nonsecure_call 属性。请参阅下面的代码块以获取示例。

```
__attribute__((cmse_nonsecure_call)) nonsecure_fxn()
{

}
```

2.2 内存保护单元

在 MSPM33C 器件中，CPUSS 中存在存储器保护单元 (MPU)。MPU 可让编程器将存储器区域定义为有权限和无权限。这类似于 SAU 如何将区域定义为安全及非安全。主要区别在于有权限模式会根据器件处于线程或者处理程序模式而更改。

复位结束后，器件处于线程模式，并且如果处理器或外设发出异常，则会进入处理程序模式。异常将始终处于有权限模式和线程模式，而 MPU 可用于确定权限级别。有关更多详细信息，请参阅 MSPM33C3 系列 160MHz 微控制器技术参考手册。

2.2.1 TrustZone® 及 MPU

在 MSPM33C CPUSS 中，MPU、SAU 及 IDAU 用于将存储器区域归属为不同权限和安全级别的组合。这可让编程器使用权限和安全限制来划分不同的访问级别。编程器可以使用的组合示例如图 2-2 所示。当器件处于处理程序模式时，处理器始终处于有权限状态。

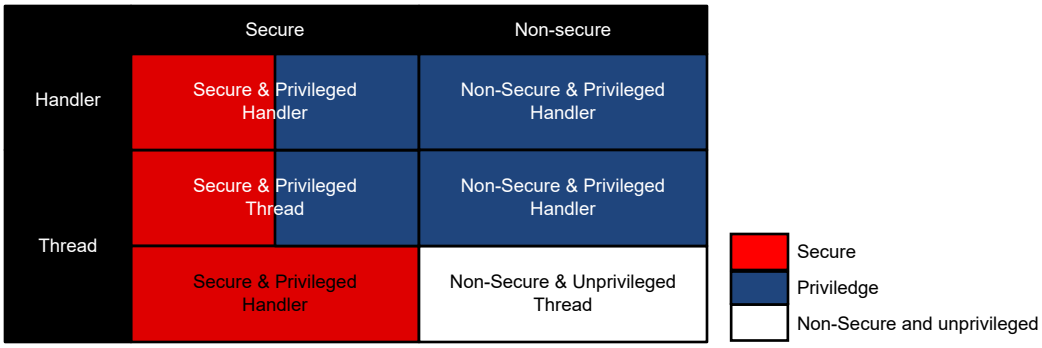

图 2-2. 安全性和有权限存储器属性

2.3 全局安全控制器

全局安全控制器 (GSC) 是 MSP 独特的外设，并在 TrustZone® 架构之上增加了额外的安全层。GSC 可配置外设、闪存和 SRAM 的安全及权限属性。这使编程器能够使用一个 IP 来保护其所有资源，并防止非安全外设访问安全存储器区域。例如，非安全 DMA 读取安全存储器。GSC 使编程器能够在非安全 DMA 尝试访问安全存储器区域时引发故障，从而防止这些攻击。有关 GSC 的更多详细信息、请参阅 MSPM33C3 系列 160MHz 微控制器技术参考手册的全局安全控制器一章。

2.3.1 GSC 存储器配置

GSC 利用 MPU 和 SAU 来配置具有安全性及权限属性的存储器。为了对此进行配置，GSC 有三个独立的控制器：外设保护控制器 (PPC)、SRAM 保护控制器 (SPC) 及闪存保护控制器 (FPC)。有关如何配置和使用寄存器的详细信息，请参阅 MSPM33C3 系列 160MHz 微控制器技术参考手册

GSC 具有与 SAU 和 MPU 类似的属性。这可能会导致不确定应使用哪个外设来保护存储器。SAU 及 MPU 在配置的区域数量或地址粒度方面受到限制。例如，M33 器件的 SAU 只能以非安全和安全属性归属 8 个存储器区域。未配置的存储器区域将保留安全属性。但是，GSC 将区域归属为 2kB 大小，从而在 1MB 器件上允许 255 个不同的安全区域。这 2kB 区域中的每个区域都可归属为安全访问或权限访问。

2.3.1.1 采用 GSC 和 SAU 时的安全例外

GSC 和 SAU 均具有以安全性归属存储器区域的功能。两者之间的一个主要区别是 GSC 将生成安全 NMI，而 SAU 生成安全故障。安全 NMI 通过错误聚合模块 (EAM) 处理，后者可让编程器查看有关错误的具体细节。例如发生的错误是哪个外设导致的，以及该外设的地址。GSC 还不限于 8 个区域，而是闪存、SRAM 和外设具有自己的安全属性粒度。有关更具体的信息，请参阅 MSPM33C3 系列 160MHz 微控制器技术参考手册。

要了解使用 GSC 和 SAU 将存储器区域归属为安全或非安全时发生的错误，请参见表 2-1。GSC 及 SAU 的安全例外中也提供了一个展示这种情况的图像

表 2-1. 基于 GSC 和 SAU 安全性归属的安全中断

处理器状态	SAU + IDAU 安全性归属	GSC 安全性归属	产生中断	访问违规
安全	安全	安全	无	无
不安全	安全	安全	安全故障	是，被 CPU 阻止
安全	安全	不安全	无	无
不安全	安全	不安全	安全故障	是，被 CPU 阻止
安全	不安全	安全	无	无
不安全	不安全	安全	安全 NMI	是，被 GSC 阻止
安全	不安全	不安全	无	无
不安全	不安全	不安全	无	无

GSC 及 SAU 的安全例外

2.3.1.2 GSC 和 MPU 的权限例外

GSC 和 MPU 可一起用于将存储器区域定义为私有边缘或非私有边缘，类似于 GSC 和 SAU 如何将存储器区域定义为安全或非安全。这使得编程器能够通过使用私有边缘访问在存储器区域上另外添加一层保护。由于私有边沿状态也发生变化，发生了异常，请确保在发生中断时预计可以进行私有边沿访问。

使用表 2-2 来确定在使用 GSC 和 MPU 将存储器区域归属为有权限或无权限时，首先发生哪一个故障。

表 2-2. 基于 GSC 及 MPU 权限归属的权限异常

处理器状态	MPU 权限归属	GSC 权限归属	产生中断	访问违规
有权限	有权限	有权限	无	无
无权限	有权限	有权限	MemMange 故障	是，被 CPU 阻止
有权限	有权限	无权限	无	无
无权限	有权限	无权限	MemMange 故障	是，被 CPU 阻止
有权限	无权限	有权限	无	无
无权限	无权限	有权限	有权限 NMI	是，被 GSC 阻止
有权限	无权限	无权限	无	无
无权限	无权限	无权限	无	无

3 信息安全模块

M33C 器件系列包括多个模块，可用于为客户实施安全的执行环境。这些安全模块包括：

- 高级加密标准 (AES)
- 密钥库
- 安全哈希算法 (SHA)
- 公钥加速器 (PKA)
- 后量子加密 (PQC)
  - ML-DSA

3.1 AES

AES 加速器模块会加速硬件中基于 FIPS PUB 197 AES 的加密和解密操作。

3.1.1 AES 概述

AES 加速器模块在硬件中使用 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
- 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
- AES 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）

AES 引擎的简要方框图如图 3-1 所示。AES 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。

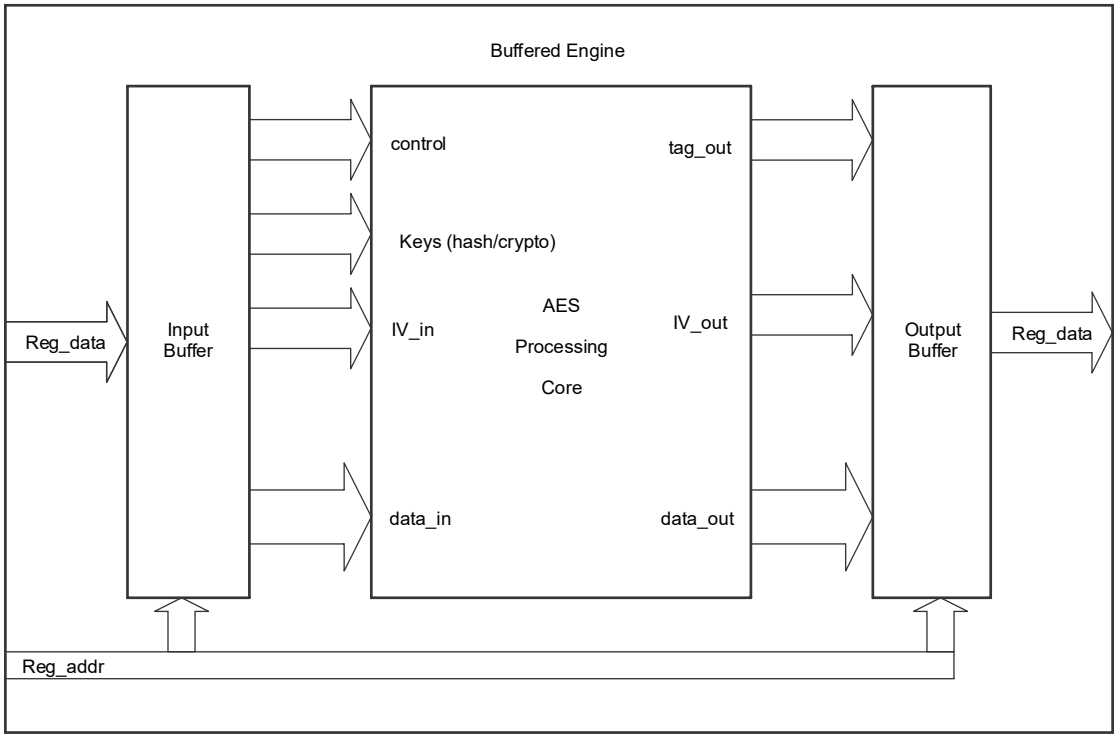

图 3-1. AES 方框图

3.1.2 AES 使用情况

AES 模块的使用方便地包装在德州仪器 (TI) MSPM33 驱动程序库 (DriverLib) 中。使用 AES DriverLib API 的示例可以在 EVM 特定的非 RTOS 示例子文件夹中找到，该子文件夹位于 MSPM33 SDK 的最新版本中。

3.1.2.1 配置

AES 模块的配置可以通过 SysConfig 完成。在 Basic Configuration 下，用户可以：

- 配置 aesadvhp 密钥长度（128/256 位密钥长度）
- 配置操作类型（加密/解密）
- 配置密码模式

有关更多详细信息，请查看所选器件的数据表。

3.1.2.2 设置

AES 模块的标准设置分成四个部分：

1. 设置 AES 状态变化监测
   a. 可以通过两种方法（触发的中断及寄存器轮询）将 AES 状态变化通知 CPU。如果使用 AES 中断来监测状态变化，则需要在初始化 AES 模块之前配置并启用这些中断。
2. AES 密钥设置
   a. 在初始化模块之前，必须在 AES 模块中设置 AES 密钥。
3. 设置 AES 密码模式
   a. 在 SysConfig 配置中选择的密码模式可以在步骤 1 和步骤 2 之后进行初始化
4. 等待 AES 模块准备就绪
   a. 可以通过在步骤 1 中选择的状态更改监测方法，向 CPU 通知，AES 模块已就绪

3.1.2.3 运行

配置和设置完成后，现在可以将数据加载到 AES 模块中，以便进行加密或解密。数据加载到模块中后，CPU 必须等到模块完成数据处理后，才能读取输出的加密/解密数据。这种等待可通过之前选择的状态监测方法来完成。如果在任何时候更改了操作类型，则必须重新初始化 AES 模块。

3.2 密钥库

3.2.1 概述

密钥库控制器提供对高级加密引擎 (AES) 密钥的安全管理。密钥库控制器的使用模式是在执行客户安全代码时将密钥安全地存入密钥库中，以及使 AES 引擎随后安全地访问密钥库，而不会向观察者泄漏任何密钥数据。128 位和 256 位密钥都可以存储在密钥库的密钥槽中。密钥库及其与 AES 引擎的交互用于安全操作，包括阻止部分密钥修改攻击。

3.2.2 密钥库使用情况

密钥库模块的使用方便地包装在德州仪器 (TI) MSPM33 驱动程序库 (DriverLib) 中。可以在特定于 EVM 的非 RTOS 示例子文件夹中找到使用 Keystore DriverLib API 的示例。

3.2.2.1 配置

密钥库模块的配置可以通过 SysConfig 来完成。在 Basic Configuration 下，用户可以：

- 配置密钥存储槽 (0-3) 或者槽位（用于 256 位密钥）
- 配置密钥长度（128/256 位密钥长度）
- 配置密钥加密接收器（AES 模块）
- 配置密钥加载位置

有关更多详细信息，请查看所选器件的数据表。

3.2.2.2 设置

密钥库模块的标准设置分为两个部分执行：

1. 设置 256 位密钥的数量
   a. 必须通知密钥库模块 256 位密钥的数量，以便可以调整可用的密钥槽。所有 256 位密钥都写入可用的最低槽位。
2. 密钥写入设置
   a. 在传输任何密钥之前，密钥库模块必须执行其密钥写入配置。

3.2.2.3 运行

配置和设置完成后，可通过提供 Keystore 模块指针和密钥传输配置来启动密钥传输。

3.3 SHA2

SHA2（安全哈希算法 2）提供一组符合 NIST 标准的加密哈希函数。本章介绍了 SHA 模块的运行情况。

3.3.1 SHA 简介

SHA 加密外设支持符合 FIPS 标准的安全哈希算法（SHA-224、SHA-256）和基于哈希的消息身份验证 (HMAC)，可用于消息身份验证应用。

3.3.1.1 SHA 特性

- 针对 FIPS PUB 180-2 和 FIPS PUB 180-3 标准的符合安全哈希标准的实施
- HMAC 支持所有符合 FIP PUB 198-1 的算法
- 高性能哈希，每个时钟周期执行一次哈希迭代（7.88 位）以实现更高的吞吐量
- 支持 SHA-224 和 SHA-256 的 HMAC 及基本哈希操作
- 哈希及 HMAC 上下文切换

- 支持 MAC 密钥 XOR 和消息填充
- 支持短于、等于或大于算法块大小的 MAC 密钥
- 支持自动消息数据调度
- 支持高达 2^{64} 位的消息大小，以 8 位为增量
- 自主支持基于消息长度生成 DMA 请求，从而减少 CPU 开销
- 支持用于输入消息解析的常量和增量地址

3.3.2 SHA 性能

表 3-1 提供了在 32MHz 下运行 MSPM33C321A 时多块图像的性能指标。

表 3-1. SHA 硬件加速器关键性能指标（多块）

图像尺寸	SHA 224 HASH	SHA 224 HMAC	SHA 256 HASH	SHA 256 HMAC
1kb	0.016ms	0.018ms	0.018ms	0.018ms
32kb	0.436ms	0.436ms	0.436ms	0.436ms
64kb	0.867ms	0.866ms	0.866ms	0.868ms
128kb	1.811ms	1.811ms	1.802ms	1.809ms

3.3.3 SHA 使用情况

SHA 模块的使用方便地包装在德州仪器 (TI) MSPM33 驱动程序库 (DriverLib) 中。使用 SHA DriverLib API 的示例可以在 EVM 特定的非 RTOS 示例子文件夹中找到，该子文件夹位于 MSPM33 SDK 的最新版本中。

3.3.3.1 配置

SHA 模块的配置可以通过 SysConfig 完成。在基本配置下，用户可以：

- 配置 SHA 模式 (HASH/HMAC)
- 配置 SHA 算法 (224/256)

有关更多详细信息，请查看所选器件的数据表。

3.3.3.2 设置

SHA 模块的标准设置分成四个部分：

1. SHA 状态监控的设置
   a. 可以通过两种方法（触发的中断及寄存器轮询）将 SHA 状态变化通知 CPU。如果使用 SHA 中断来监测状态变化，则需要在使用 SHA 模块之前配置并启用这些中断。
2. 设置 SHA 数据长度
   a. 在使用 SHA 模块之前，需要配置 SHA 数据长度
   b. 必须在 SHA 模式包括的情况下以字节为单位指定数据长度
   c. 然后可以通知 SHA 模块，模式和数据长度可用
3. MAC 密钥设置（仅适用于 HMAC 模式）
   a. 如果使用 SHA HMAC 模式，在使用 SHA 模块之前，必须为模块提供用于加密的 MAC 密钥
   b. 必须以字的形式将 MAC 密钥写入 SHA 模块，其中包括密钥总大小
   c. 然后，必须通知 SHA 模块，该密钥可用
   d. 必须通知 SHA 模块，可以开始处理密钥
   e. 最后，CPU 必须等到 SHA 模块完成处理
4. 直接存储器存取 (DMA) 的设置
   a. SHA 模块基于数据块运行，并且必须手动馈送数据才能使模块进行摘要。为了执行这一操作，可以通过 CPU 或 DMA 传输来馈送数据。如果使用 DMA 传输，则必须在使用 SHA 模块之前设置 DMA 通道。
   b. DMA 源地址必须指向要哈希处理的消息的开头
   c. DMA 目标地址必须指向 SHAW_DATA_FIXED 寄存器
   d. DMA 消息长度必须设置成以字为单位的消息长度
   e. DMA 源和目标宽度必须为一个字的大小

f. DMA 源必须配置为单递增模式，目标不得更改
g. 必须将 DMA 触发器设置为 SHAW_TRIGGER
h. 最后，可启用 DMA 通道

3.3.3.3 运行

配置和设置完成后，现在可以通过 DMA 或 CPU 将数据加载到 SHA 模块中。数据加载到模块中后，CPU 必须等到模块完成这些数据处理后，才能读取输出的哈希数据。这种等待可通过之前选择的状态监测方法来完成。从 SHA 模块读取哈希数据后，必须将其释放以用于下一个哈希请求。

3.4 PKA

3.4.1 PKA 简介

PKA 是用于公钥加速的集成模块，可减轻密集公共加密操作的计算负担。

3.4.1.1 PKA 特性

- PKA 引擎提供以下基本操作
  - 大矢量加法、减法以及组合加法及减法
  - 向左或向右移动大矢量
  - 大矢量乘法和除法（带或者不带商）
  - 大矢量比较和复制
- PKA 引擎提供以下复杂操作：
  - 大矢量无符号值模块化幂运算
  - 使用带预先计算的 Q 逆向量的 CRT 方法的大矢量无符号值模块化幂
  - 模块化反向
  - 在两种类型的曲线上进行 ECC 操作：Montgomery 曲线，如 Curve25519 和 Curve448，以及 $y^2=x^3+ax+b \pmod p$ 形式的任意曲线。
  - 椭圆曲线上的 ECC 点加/加倍，用仿射点或投影点作为输入/输出
  - 椭圆曲线上的 ECC 点乘
- 非法状态及时序攻击检测

3.4.2 PKA 使用情况

PKA 模块的使用方便地包装在德州仪器 (TI) MSPM33 驱动程序库 (DriverLib) 中。可在特定于 EVM 的非 RTOS 示例子文件夹中找到采用 PKA DriverLib API 的示例。

3.4.2.1 配置

只需将 PKA 模块添加到工程中，即可通过 SysConfig 配置 PKA 模块。

有关更多详细信息，请查看所选器件的数据表。

3.4.2.2 设置

PKA 模块的设置要求由 SysConfig 处理，因为初始化步骤是自动生成的。只需调用生成的 *SYSCFG_DL_init()* 函数即可。

3.4.2.3 运行

配置和设置完成后，现在可以使用 PKA 模块执行加密操作。最好先启动 PKA 操作，然后通过轮询 PKA 状态来等待操作完成。PKA 状态清除后，便可收集 PKA 操作的结果。

3.5 PQC

后量子加密 (PQC) 是一套加密标准，旨在抵御来自量子计算机的威胁，由美国国家标准与技术研究院 (NIST) 定义。其中一个标准是 Module-Lattice-Based Digital Signature Algorithm (ML-DSA)

3.5.1 ML-DSA

3.5.1.1 ML-DSA 简介

ML-DSA 是 NIST 定义的 PQC 算法之一，用于防止来自量子计算机的威胁。该标准可与客户安全代码 (CSC) 一同使用，以确保嵌入式应用的真实性。

3.5.1.2 ML-DSA 用法

ML-DSA 标准的用法方便地包装在德州仪器 (TI) MSPM33 加密库中。使用 ML-DSA 标准和 MCUBoot 的示例可以在 EVM 特定的非 RTOS 示例子文件夹中找到，该文件夹位于最新版本的 MSPM33 SDK 中。

3.5.1.2.1 配置

可通过 SysConfig 完成配置 MCUBoot 以利用 ML-DSA 的操作。在基本配置下，用户可以：

- 使用 ML-DSA 启用身份验证
- 使用 ECDSA 及 ML-DSA 启用身份验证

有关更多详细信息，请查看所选器件的数据表。

3.5.1.2.2 设置

然后、通过将 MLDSA 密钥和密钥长度存储在 MCUBoot 利用的全局 bootutil_keys 结构中，可以最终确定在 MCUBoot 中 ML-DSA 的使用设置。下方显示了这种情况的一个示例：

```
const struct bootutil_key bootutil_keys[] = {
    {
        .key = mldsa_pub_key,
        .len = &mldsa_pub_key_len,
    },
};
const int bootutil_key_cnt = 1;
```

3.5.1.2.3 运行

配置和设置完成后、MCUBoot 现在将利用 ML-DSA 和之前设置的公共 ML-DSA 密钥，在启动 *boot_go* 时验证应用程序映像。

4 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

日期	修订版本	注释
2025 年 12 月	*	初始发行版

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