Application Note

F29H85x™ 的串行闪存编程

Skyler Baumer, Sen Wang, Aakash Kedia, Arush Pant

摘要

在无法使用 JTAG 对目标器件进行编程的情况下，通常需要对嵌入式处理器进行编程。在此类情况下，工程师需要依赖外设编程设计。通过在 ROM 中添加多个程序加载实用程序，C2000™ 器件可帮助实现串行编程。这些实用程序很有用，但只能解决一半的编程问题，因为它们只允许将应用程序代码加载到 RAM 中。本应用手册从闪存内核的角度介绍了这些 ROM 加载程序。使用 ROM 加载程序将闪存内核加载到 RAM，然后执行 ROM 加载程序，用于在最终应用中对目标器件的片上闪存进行编程。在 F29H85x 上使用闪存内核时的一个关键考虑因素是添加硬件安全模块 (HSM)。本文档详细介绍了 C2000 器件的一种可能的实施方式，并提供了用于评估设计的 PC 实用程序。

内容

插图清单

表格清单

商标

1 编程基础知识

在对器件进行编程之前，有必要了解 C2000 器件的非易失性存储器的工作原理。闪存是一种非易失性存储器，允许用户将其中的内容轻松擦除并重新编程。擦除操作将扇区中的所有位设置为“1”，而编程操作则有选择地将位设置为“0”。

所有 C2000 器件上的闪存操作均使用 CPU 执行。算法被加载到 RAM 中并由 CPU 控制以执行任何闪存操作。例如，若要使用 Code Composer Studio™ 擦除或编程 C2000 器件的闪存，需要将闪存算法加载到 RAM 中并让处理器执行它们。没有使用特殊的 JTAG 命令。所有闪存操作均通过闪存应用程序编程接口 (API)来执行。所有闪存操作都是使用 CPU 完成的，因此器件编程有很多可能性。无论内核和应用程序如何引入器件中，闪存都是使用 CPU 进行编程的。

2 引言

ROM 引导加载程序（也被称为主引导加载程序或简称为加载程序）是一小段代码，位于目标器件的引导 ROM 存储器中，允许从外部主机加载和执行代码。大多数情况下，使用通用异步接收器/发送器 (UART) 或控制器局域网 (CAN) 等通信外设将代码加载到器件中，而不是使用 JTAG，后者需要用到昂贵的专用工具，不建议在商业环境中使用。

通过引导引脚，可以使用确定调用哪个 ROM 加载程序的各种外设来配置不同的引导模式。在本应用手册中，使用的外设是 UART。与引导引脚关联的引导模式指的是外设的第一个实例。对于 UART，引导模式与 UART**A** 关联。

C2000 器件通过在 ROM 中添加一些基本的加载实用程序来部分解决固件更新问题。根据器件和存在的通信外设，可以使用 UART、串行外设接口 (SPI)、内部集成电路 (I2C) 和 CAN 将代码加载到片上 RAM 中。这些加载程序的一部分存在于每个 C2000 器件中，但加载程序只能将代码加载到 RAM 中。如何弥合这一差距并将应用程序代码编程到非易失性存储器中？

概括来说，对闪存等非易失性存储器的应用程序编程需要两个步骤：

1. 使用 ROM 引导加载程序将辅助引导加载程序下载到 RAM。
2. 在 RAM 中运行辅助引导加载程序以将应用程序下载到闪存。

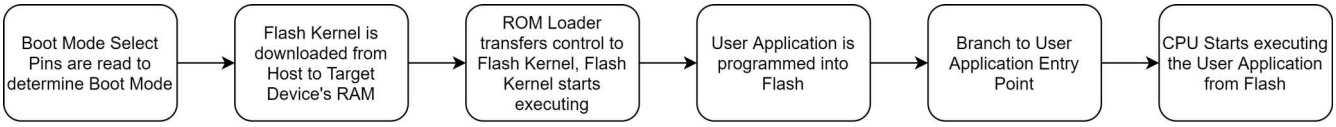

图 2-1. 闪存内核流程

尽管基于 C28 器件实现的闪存内核与本文中说明的内核在总体概念上相似，但请注意，它们之间存在一些主要区别。

2.1 硬件安全模块

基于 C28 的器件的闪存内核设计与 F29H85x 闪存内核设计之间的主要区别在于硬件安全模块 (HSM) 的集成。HSM 是一个子系统，可提供安全和加密功能。C29 CPU 与 HSM 连接，以执行代码身份验证、安全启动、安全固件升级和加密运行时通信所需的加密操作。

在 UART 启动序列期间，HSM 负责对传入的映像进行身份验证。要使身份验证成功，传入的映像必须包括 X.509 证书。要使用闪存内核正确生成 X.509 证书，请参阅节 2.2。

HSM 引入了不同器件*安全状态*的概念。器件状态为“高安全性 - 现场安全 (HS-FS)”、“高安全性 - 密钥预置 (HS-KP)”和“高安全性 - 安全启用 (HS-SE)”。默认情况下，F29H85x 器件状态为 HS-FS。表 2-1 介绍了这三种状态之间的差异。

表 2-1. 器件安全状态

	HS-FS	HS-KP	HS-SE
C29 引导映像（闪存内核）	未强制执行安全启动	使用由 Keywriter 编程的客户密钥强制执行安全启动	使用由 Keywriter 编程的客户密钥强制执行安全启动

表 2-1. 器件安全状态（续）

	HS-FS	HS-KP	HS-SE
HSM 引导映像	强制执行安全启动（使用 TI 提供的默认密钥）	使用由 Keywriter 编程的客户密钥强制执行安全启动	使用由 Keywriter 编程的客户密钥强制执行安全启动
C29 JTAG	默认打开	默认打开	默认关闭
SoC 防火墙	默认打开	对 HSM 禁用，对 C29 启用	对 HSM 禁用，对 C29 启用
C29 CPU 对 C29 闪存存储体的访问	启用	禁用	启用

TI 提供了 OTP（一次性可编程）Keywriter，可用于将 HS-FS 器件转换为 HS-KP 或 HS-SE。OTP Keywriter 是 TI 提供的一组 HSM 运行时固件和工具（证书生成），在器件上执行时可以结合起来实现以下功能：

- 预置最多两组客户密钥（两个公钥用于引导映像的信任根，加密密钥用于引导映像的解密）。
- 对扩展 OTP 字段进行编程，这些字段是可用于客户特定用途的附加 OTP 字段。
- 对 OTP 字段 KEY_COUNT、KEY_REVISION 进行编程，可实现器件从 HS-FS（现场安全）到 HS-KP（密钥预置）的转换。
- 对 SBL、HSM、APP、SECCFG 映像的 SWREV 等 OTP 字段进行编程，通过 HS-KP 和 HS-SE 器件上的安全启动强制执行防回滚检查。

对这些字段进行编程后，器件状态转换为 HS-KP，引导 ROM 根据器件中预置的密钥通过映像身份验证和解密来强制执行安全启动。安全启动需要使用客户密钥对映像进行加密（可选）和签名。然后，SoC 使用活动 MPK 哈希（用于验证签名）和 MEK（用于解密）对此映像进行验证。

小心

- 在各个存储区域中进行的这种烧录密钥操作是不可逆的，因此需要谨慎操作，确保提供正确格式的密钥值和正确的密钥预置。
- 对字段进行编程的操作是不可逆的，如果提供了错误的值或配置，则会对器件造成永久损坏。

有关 OTP Keywriter 的更多信息，请从 F29H85x MCU SDK 下载页面申请受限制的安全软件包。

HS-FS、HS-KP 和 HS-SE 器件的固件升级过程有所不同。

- 有关 HS-FS 器件上 C29 CPU1 固件升级的详细信息，请参阅 CPU1 固件升级 (HS-FS)。
- 有关 HS-KP/HS-SE 器件上 C29 CPU1 固件升级的详细信息，请参阅 CPU1 固件升级（HS-KP 或 HS-SE）。
- 有关 HS-KP/HS-SE 器件上 HSM 固件升级的详细信息，请参阅 HSM 固件升级（HS-KP 或 HS-SE）。

2.2 ROM 引导加载程序

说明了闪存内核的基本理念以及如何进行固件升级，下一节将详细介绍该过程的第一步：通过 bootROM 将内核加载到 RAM。

开始时，器件启动，并根据引导模式决定器件是执行已编程到闪存中的代码还是使用某个 ROM 加载程序加载代码。本应用手册重点介绍未连接仿真器时的引导执行路径。

备注

本节基于 F29H85x 器件。特定器件的具体信息可以在器件特定技术参考手册 (TRM) 的 *ROM 代码和外设启动* 部分中找到。

表 2-2. F29H85x 器件的默认引导模式

引导模式	GPIO72（默认引导模式选择引脚 1）	GPIO84（默认引导模式选择引脚 0）
并行 I/O	0	0
UART	0	1
CAN	1	0
闪存	1	1

在启动 ROM 准备好使用的器件后，器件决定从哪里开始执行。如果是独立启动，器件通过检查两个 GPIO（如表 2-2 中所示，默认选择是 GPIO 72 和 84）的状态来实现此目的。在某些情况下，可以检查编程到一次性可编程 (OTP) 存储器中的两个值。在本应用手册中所述的实现方案中，使用了 UART 加载程序，因此在上电时 GPIO 84 必须强制为高电平，GPIO 72 必须强制为低电平。如果器件启动时出现这种情况，则 ROM 中的 UART 加载程序以 115200 的波特率开始执行并运行。此时，器件已准备好接收来自主机的代码。

之前基于 C28 的器件和 F29H85x 之间的主要区别在于是否包含硬件安全模块 (HSM)。所有启动流程都需要 HSM 对传入映像进行身份验证，然后才能执行启动流程。

有关启动流程的详细信息，请参阅技术参考手册 (TRM) 的 *ROM 代码和外设启动* 部分。有关 HSM 和 C29 CPU 在启动序列期间如何进行通信，请参阅 *器件启动流程图* 部分。

2.3 带有 X.509 证书的合并映像

随着 HSM 的加入，BootROM 要求所有传入的映像采用二进制格式，并且附带 X.509 证书。二进制文件的前 0x1000 个字节必须包含密钥证书。

下面列出了由 UART 闪存内核和其他 SDK 示例提供的默认编译后步骤，用于生成带有 X.509 证书的合并二进制映像。

对于新的 Code Composer Studio 工程，将以下脚本粘贴到工程属性的“编译”类别下的*编译后步骤*部分。要独立运行脚本，请在器件 SDK 的根目录中找到 import.mak 文件，以获取该变量的默认别名。

RAM 编译后步骤（对于闪存内核）：

```
${CG_TOOL_OBJCOPY} --strip-all -O binary ${ProjName}.out ${ProjName}.bin
$(PYTHON)  ${COM_TI_MCU_SDK_INSTALL_DIR}/tools/boot/signing/mcu_rom_image_gen.py --image-bin $
{ProjName}.bin --core C29 --swrv 1 --loadaddr 0x200E1000 --sign-key ${COM_TI_MCU_SDK_INSTALL_DIR}/
tools/boot/signing/mcu_custMpk.pem --out-image ${ProjName}.cert.bin --boot RAM --device f29h85x --
debug DBG_SOC_DEFAULT
```

- 生成基于 RAM 的二进制映像： ${ProjName}.cert.bin
- mcu_rom_image_gen.py ram 参数：--loadaddr 0x200E100 (LDAx RAM)，--boot RAM

闪存编译后步骤（对于闪存应用程序映像）：

```
${CG_TOOL_OBJCOPY} --remove-section=cert -O binary ${ProjName}.out ${ProjName}.bin
$(PYTHON) ${COM_TI_MCU_SDK_INSTALL_DIR}/tools/boot/signing/mcu_rom_image_gen.py --image-bin $
{ProjName}.bin --core C29 --swrv 1 --loadaddr 0x10001000 --sign-key ${COM_TI_MCU_SDK_INSTALL_DIR}/
tools/boot/signing/mcu_gpkey.pem --out-image ${ProjName}_cert.bin --device f29h85x --boot FLASH --
img_integ no
${CG_TOOL_OBJCOPY} --update-section cert=C29-cert-pad.bin ${ProjName}.out ${ProjName}_cert.out
$(DELETE) ${ProjName}.out C29-cert-pad.bin;
$(RENAME) ${ProjName}_cert.out ${ProjName}.out
```

- 生成基于闪存的二进制映像： ${ProjName}_cert.bin
- mcu_rom_image_gen.py 闪存参数：--loadaddr 0x10001000（闪存入口地址），--boot FLASH

上面显示的两个编译后步骤都会为应用程序生成证书，将应用程序 .out 文件转换为二进制文件，并创建带有 X.509 证书的合并二进制映像。

此外，编译后步骤利用 SDK 中提供的两个不同密钥：

- mcu_gpkey.pem：TI 提供的通用密钥，用于生成 TI 密钥证书。
- mcu_custMpk.pem：一个虚拟密钥，这个虚拟密钥模拟了一个自定义密钥，可用作示例自定义密钥，用于测试密钥预置和代码预置。

备注

要使用自定义密钥证书更改生成的密钥证书，请提供备用密钥并通过参数 *--sign-key* 传递到 python 脚本中。对于所有涉及 HS-KP 和 HS-SE 以及 HS-SE 的 HSMRt 的闪存映像，这都是必需的。

3 闪存内核实现

与基于 C28 的设计相比，F29H85x 的 UART 闪存内核由于集成了 HSM 而有很大不同。下面简要说明了本示例中实现的功能。

- CPU1 DFU（仅限 HS-FS）
  1. 器件需要配置为存储体模式 0。
  2. CPU1 中的闪存内核将应用程序编程到闪存存储体。
  3. 在 0x10000000 - 0x10000FFF 中对证书编程，证书不用于身份验证，而是用于推断内核二进制文件的大小。
  4. 可在其余闪存地址中对应用程序进行编程，TI 建议将 codestart 设置为 0x10001000。因为 0x10001000 是闪存引导模式中的闪存入口地址。

> **小心**
> 对于不关心 HSM 提供的安全功能的用户，只需使用 CPU1 DFU 流来升级 CPU1 固件。

- 加载 HSMRt（密钥预置和代码预置的先决条件）
  1. CPU1 中的闪存内核接收密钥预置 HSM 运行时 (HSMRt) 并将其放置在 LDAx RAM 中。
  2. HSM 验证 HSMRt 映像的身份并开始在 LDAx RAM 中执行运行时。
- CPU1 密钥预置 (HS-FS -> HS-KP)
  1. CPU1 中的闪存内核加载 HSMRt
  2. 闪存内核接收密钥证书并放置在 LDAx RAM 中
  3. HSM 验证密钥证书并将其编程到 OTP 中
- CPU1 代码预置（仅限 HS-SE）
  1. CPU1 中的闪存内核加载到 HSMRt
  2. 闪存内核接收 CPU1 应用程序映像证书并与 HSMRt 共享证书
  3. 成功对映像进行身份验证后，HSMRt 通过确认消息进行响应，之后闪存内核开始通过 UART 将数据块导入到 LDAx 存储器中。
  4. 每次接收到 16KB（LDAx 存储器的大小）数据后，闪存内核都会发送 HSM 请求来对数据进行编程，以供进一步处理
  5. 接收到所有数据块并进行编程后，请求 HSMRt 验证在 HSM 活动和休眠存储体中进行编程的代码。当 HSMRt 固件根据证书对编程的映像进行身份验证时，还会对证书进行进一步编程，以便确保在后续的下电上电期间能够成功引导。
  6. 成功完成身份验证后，HSM 会将固件编程到 CPU1 闪存
- HSM 代码预置 (HS-KP/HS-SE -> HS-SE)
  1. CPU1 中的闪存内核加载到 HSMRt
  2. 闪存内核接收 HSM 应用程序映像证书并与 HSMRt 共享证书
  3. 成功对映像进行身份验证后，HSMRt 通过确认消息进行响应，之后闪存内核开始通过 UART 将数据块导入到 LDAx 存储器中。
  4. 每次接收到 16KB（LDAx 存储器的大小）数据后，闪存内核都会发送 HSM 请求来对数据进行编程，以供进一步处理
  5. 接收到所有数据块并进行编程后，请求 HSMRt 验证在 HSM 活动和休眠存储体中进行编程的代码。当 HSMRt 固件根据证书对编程的映像进行身份验证时，还会对证书进行进一步编程，以便确保在后续的下电上电期间能够成功引导。
- SECCFG 代码预置 (HS-KP -> HS-SE)
  1. CPU1 中的闪存内核加载到 HSMRt
  2. 闪存内核接收映像证书并与 HSMR 共享证书
  3. 成功对映像进行身份验证后，HSMRt 通过确认消息进行响应，之后闪存内核开始通过 UART 将 SecCfg 数据导入 LDAx 存储器
  4. 在接收所有 SecCfg 数据并对其进行编程后，请求 HSMRt 验证在休眠存储体中编程的 SecCfg，确保计数器值有效。当 HSMRt 根据证书对编程的映像进行身份验证时，还会对证书进行进一步编程，以便确保在后续的下电上电期间能够成功引导。
  5. 注意，对于 HS-SE 器件，根据 SSU 寄存器的交换值作出证书的编程决策。

小心

密钥预置和代码预置都需要在 HSM 上运行基于 RAM 的 HSMRt 映像。但是，该映像在预置的每个阶段都需要不同的密钥证书。

在密钥预置中，用户必须为 HSMRt 映像提供默认 TI 密钥证书，以便在 HS-FS 器件上完成身份验证，而代码预置则需要包含用户密钥证书的映像，以便在 HS-KP/HS-SE 器件上完成身份验证。

硬件安全模块部分介绍的 OTP Keywriter 固件包含可执行这些任务的二进制映像。

在存储体模式 0 中，UART 闪存内核专为 CPU1 而设计。这意味着所有闪存存储体都分配给 CPU1，并且存储体交换功能不可用。有关存储体模式的更多详细信息，请参阅 TRM 的*存储体模式和交换*部分。闪存内核与 F29H85x MCU SDK 中提供的主机 PC 应用程序 (MCU_SDK_F28H85x > *utilities* > *flash_programmers* > *serial_flash_programmer*) 进行通信，并就接收数据包和给定命令的完成情况向主机提供反馈。

将内核加载到 RAM 并通过 UART ROM 引导加载程序执行后，内核首先初始化 PLL、GPIO、UARTA 模块和闪存模块。UART 通信的波特率配置为 115200，与 UART 引导流程类似。在此之后，内核开始一个 while 循环，等待来自主机的命令，执行命令，并将状态包发送回主机。当发送 Run（运行）或 Reset（重置）命令时，此 while 循环中断。

命令在表 4-2 中描述的数据包中发送，每个数据包要么被确认，要么不被确认。除 Run（运行）和 Reset（重置）外，所有命令都在完成后发送带有操作状态的数据包。状态数据包会发送 16 位状态代码和 32 位地址。如果出现错误，数据中的地址会指定第一个错误的地址。发生 NO_COMMAND_ERROR 时，地址为 0x1000。

3.1 CPU1 固件升级 (HS-FS)

要对 HS-FS 器件的 CPU1 执行固件升级，请编译工程的 CPU1_APP 构建配置。可以通过右键点击工程，将鼠标悬停在 Build Configurations 上并选择 CPU1_APP 来完成此操作。

对于器件固件升级 (DFU) 命令，将执行以下步骤：

1. CPU1 中的内核接收执行 DFU 的命令数据包。
2. 为准备新的应用程序，在编程前擦除整个闪存。
3. 内核准备从主机接收 X.509 证书。
4. 内核确认传入证书具有适当的大小和格式，并得出传入映像的大小。证书暂时存储在 RAM 中。
   a. 而在此过程中，不会对证书或传入映像进行身份验证。由于内核根据证书推导映像大小并在独立的闪存启动过程中进行身份验证，因此仍需要正确生成证书。
5. 如果接受证书大小和格式，则内核准备接收新应用程序并将其编程到闪存中
6. 内核以 1024 位的块对传入的应用程序进行编程。在对每个块编程之后，内核会验证数据和 ECC 是否正确编程到闪存中。
7. 成功对整个映像进行编程后，内核会将关联的证书编程到 FLC1 B0/B1 的前 0x1000 个地址。将证书编程到闪存的这个区域后，器件能够在独立闪存启动模式下启动至现有应用程序。

要生成可由内核加载的应用程序映像，请参阅节 2.3 的编译后步骤。有关用法的更多详细信息，请参阅节 5.1。

3.2 密钥预置（HS-FS 至 HS-KP）

要在 HS-FS 器件的 CPU1 上执行密钥预置，请编译工程的 KP_APP 构建配置。可以通过右键点击工程，将鼠标悬停在 Build Configurations 上并选择 KP_APP 来完成此操作。

密钥预置将 HS-FS 器件转换为 HS-KP 器件，并且发生以下事件：

1. UART 启动模式下的 BootROM 接收 UART 闪存内核并启动内核。
2. CPU1 中的内核接收一个命令数据包，从而接收 HSMRt 映像。
3. 内核准备作为合并映像的一部分从主机接收 X.509 证书。
4. 内核确认传入证书具有适当的大小和格式，并得出传入映像的大小。证书暂时存储在 RAM 中。
5. 内核将 HSMRt 映像存储在共享 LDAx RAM 中，并请求 HSM 进行身份验证。
6. 身份验证成功后，HSM 开始在共享 LDAx RAM 中执行 HSMRt。
7. 然后，内核接收一个命令数据包，从而接收 HSM 密钥。

8. 内核接收密钥证书并放置在共享 LDAx RAM 中。
9. 内核会通知 HSM 已收到密钥。
10. 成功验证密钥证书的身份后，HSMRt 会将密钥编程到 OTP 中。
11. 执行上电复位以将器件转换为 HS-KP。

> **小心**
> 需要将节 2.1 中所述的 Keywriter 二进制映像用作 HSMRt。

请参阅节 5.5，了解在主机应用程序中执行这些操作的步骤。

如节 2.1 中所述，Keywriter 固件用于对客户密钥进行编程并从 HS-FS 转换到 HS-KP。Keywriter 固件支持对以下密钥类型和字段进行编程：

密钥	说明	KeyWriter 使用说明	对 HS-SE 器件的影响
SMPKH	次级制造商公钥哈希 SMPKH 长度：512 位 BCH 长度：64 位	• 客户主密钥集：公钥哈希 • SMPK 是 4096 位客户 RSA 公钥，用于验证引导映像中包含的 RSA 签名。 • SMPKH 值：通过 SHA512 哈希算法生成 SMPK 的 512 位哈希，分为 256 位的两个部分。即 SMPKH_P1/SMPKH_P2 • BCH 值：对于 SMPKH_P1 和 SMPKH_P2 的每个部分，使用算法 (288, 261, 7) 计算 32 位 BCH。此算法支持 3 位纠错 • KeyWriter 强制执行的检查：不适用	当密钥配置字段 KEYREV=1 时，安全启动的活动密钥用于验证启动映像 x.509 证书的信任根
SMEK	次级制造商加密密钥 SMEK 长度：256 位 BCH 长度：32 位	• 客户主密钥集：加密密钥 • SMEK 是用于引导映像加密的 256 位客户加密密钥 • SMEK 值：用于引导映像 AES-CBC 加密的初始 256 位对称密钥 • BCH 值：对于 256 位 SMEK，使用算法 (288, 261, 7) 计算 32 位 BCH。此算法支持 3 位纠错 • KeyWriter 强制执行的检查：不适用	当密钥配置字段 KEYREV=1 时，如果通过 x509 证书加密和启用，则安全启动的活动密钥用于为引导映像解密

密钥	说明	KeyWriter 使用说明	对 HS-SE 器件的影响
BMPKH	备份制造商公钥哈希 BMPKH 长度：512 位 BCH 长度：64 位	• 客户备份密钥集：公钥哈希 • **备注** 器件支持备份密钥对，该密钥对必须由 Keywriter 预置，并且在现场部署产品后，就可以通过在现场对 KEVREV 字段进行增量编程来激活。请注意，如果 Keywriter 未在 HS-FS 器件上预置备份密钥对，则在器件转换到 HS-SE 后，无法对备份密钥对进行编程。 • BMPK 是 4096 位客户 RSA 公钥，用于验证引导映像中包含的 RSA 签名。 • BMPKH 值：通过 SHA512 哈希算法生成 BMPK 的 512 位哈希，分为 256 位的两个部分。即 BMPKH_P1/BMPKH_P2 • BCH 值：对于 BMPKH_P1 和 BMPKH_P2 的每个部分，使用算法 (288, 261, 7) 计算 32 位 BCH。此算法支持 3 位纠错 • KeyWriter 强制执行的检查：不适用	当密钥配置字段 KEYREV=2 时，安全启动的活动密钥用于验证启动映像 x.509 证书的信任根
BMEK	备份制造商加密密钥 BMEK 长度：256 位 BCH 长度：32 位	• 客户备份密钥集：加密密钥 • BMEK 是用于引导映像加密的 256 位客户加密密钥 • BMEK 值：用于引导映像 AES-CBC 加密的初始 256 位对称密钥 • BCH 值：对于 256 位 BMEK，使用算法 (288, 261, 7) 计算 32 位 BCH。此算法支持 3 位纠错 • KeyWriter 强制执行的检查：不适用	当密钥配置字段 KEYREV=2 时，如果通过 x.509 证书加密和启用，则安全启动的活动密钥用于为引导映像解密
KEYCNT	密钥计数配置字段 长度：16 位	• 器件中预置的活动密钥集，支持以下值 • 如果只有主密钥集是器件的记录计划 (SMPKH/SMEK) • 如果主密钥集和备份密钥集均为器件的记录计划（SMPKH/SMEK 和 BMPKH/BMEK） • **备注** 该字段控制器件中的活动密钥集。如果用户希望启用备份密钥集，则强制性要求是预置备份密钥对并通过 Keywriter 将 KEYCNT 设置为 2。如果此字段设置为 1，则器件只能支持将主密钥集用于安全启动，并且以后无法更新该字段。 • KeyWriter 强制执行的检查：不适用	器件中预置的活动密钥集可供密钥管理器用于解码和设置密钥。
KEYREV	密钥版本配置字段 长度：16 位	• 器件中用于信任根的当前活动密钥版本，支持以下值 • 用于安全启动的主密钥集 (SMPK/SMEK)（KeyWriter 的建议配置） • 用于安全启动的备份密钥集 (BMPK/BMEK) • KeyWriter 强制执行的检查：不适用	安全启动的当前活动密钥版本

密钥	说明	KeyWriter 使用说明	对 HS-SE 器件的影响
MSV	特定于型号的值 长度：24 位 BCH：8 位	• 24 位特定于型号的值，客户可以对 24 位值进行编程，以便在器件的生产流程或启动流程中区分使用相同 SoC 的产品型号。 • BCH 值：对于 24 位 MSV，使用算法 (32, 24, 8) 计算 8 位 BCH。此算法支持 2 位纠错	对启动 ROM 没有影响，SW 需要理解此字段的使用方法
SWREV-SBL	SBL 软件版本 长度：64 位	• 支持 32 个值（1 至 32 个状态，具有双重冗余） • **建议将初始值设置为 0x1**	通过 SWRV 扩展为 SBL 映像 x.509 证书启用防回滚功能
SWREV-HSM	TIFS-MCU 软件版本 长度：64 位	• 支持 32 个值（1 至 32 个状态，具有双重冗余） • **建议将初始值设置为 0x1**	通过 SWRV 扩展为 TIFS-MCU 映像 x.509 证书启用防回滚功能
SWREV-APP	应用程序映像软件版本 长度：192 位	• 支持 64 个值（1 至 64 个状态，具有双重冗余） • 建议将初始值设置为 0x1	对启动 ROM 没有影响，TIFS-MCU 需要理解此字段的使用方法
扩展 OTP	扩展 OTP 阵列长度：1664 位	供客户使用的 1664 位扩展 OTP 阵列	对启动 ROM 没有影响

3.3 CPU1 安全固件升级（HS-KP/SE 至 HS-SE）

要在 HS-KP 或 HS-SE 器件的 CPU1 上执行安全固件升级，请编译工程的 CP_APP 构建配置。可以通过右键点击工程，将鼠标悬停在 Build Configurations 上并选择 CP_APP 来完成此操作。

要在 HS-KP 或 HS-SE 器件上执行 CPU1 安全固件升级，会发生以下事件：

1. UART 启动模式下的 BootROM 接收 UART 闪存内核并启动内核。
2. CPU1 中的内核接收一个命令数据包，从而接收 HSMRt 映像。
3. 内核准备作为合并映像的一部分从主机接收 X.509 证书。
4. 内核确认传入证书具有适当的大小和格式，并得出传入映像的大小。证书暂时存储在 RAM 中。
5. 内核将 HSMRt 映像存储在共享 LDAx RAM 中，并请求 HSM 进行身份验证。
6. 身份验证成功后，HSM 开始在共享 LDAx RAM 中执行 HSMRt。
7. 内核接收一个命令数据包，从而接收 CPU1 闪存应用程序映像。
8. 内核接收 X.509 映像证书并将其与 HSMRt 分享。
9. 成功对映像进行身份验证后，HSMRt 通过确认消息进行响应，之后闪存内核开始通过 UART 将数据块导入到 LDAx 存储器中。
10. 每次接收到 16KB（LDAx 存储器的大小）数据后，闪存内核都会发送 HSM 请求来对数据进行编程以供进一步处理。
11. 接收到所有数据块并进行编程后，请求 HSMRt 验证在 HSM 活动和休眠存储体中进行编程的代码。当 HSMRt 固件根据证书对编程的映像进行身份验证时，还会对证书进行进一步编程，以便确保在后续的下电上电期间能够成功引导。
12. 成功完成身份验证后，HSM 会将固件编程到 CPU1 闪存。
    a. 如果器件之前处于 HS-KP，则器件将转换为 HS-SE。

请参阅节 5.6，了解在主机应用程序中执行这些操作的步骤。

小心

对于 HS-KP 或 HS-SE 器件，DPL 中断 LINK 和 STACK 指针需要分别设置为 LINK2 和 STACK2。要调整该设置，请在 CCS 中打开 syscfg 文件，选择“TI Driver Porting Layer (DPL)”部分下面的 *Clock*。

需要将节 2.1 中所述的 Keywriter 二进制映像用作 HSMRt。

3.4 HSM 固件升级（HS-KP/SE 至 HS-SE）

要在 HS-KP 或 HS-SE 器件的 HSM 上执行固件升级，请编译工程的 CP_APP 构建配置。可以通过右键点击工程，将鼠标悬停在 Build Configurations 上并选择 CP_APP 来完成此操作。

要在 HS-KP 或 HS-SE 器件上执行 CPU1 固件升级，会发生以下事件：

1. UART 启动模式下的 BootROM 接收 UART 闪存内核并启动内核。
2. CPU1 中的内核接收一个命令数据包，从而接收 HSMRt 映像。
3. 内核准备作为合并映像的一部分从主机接收 X.509 证书。
4. 内核确认传入证书具有适当的大小和格式，并得出传入映像的大小。证书暂时存储在 RAM 中。
5. 内核将 HSMRt 映像存储在共享 LDAx RAM 中，并请求 HSM 进行身份验证。
6. 内核接收一个命令数据包，从而接收 HSM 闪存应用程序映像。
7. 内核接收 X.509 映像证书并将其与 HSMRt 分享。
8. 成功对映像进行身份验证后，HSMRt 通过确认消息进行响应，之后闪存内核开始通过 UART 将数据块导入到 LDAx 存储器中。
9. 每次接收到 16KB（LDAx 存储器的大小）数据后，闪存内核都会发送 HSM 请求来对数据进行编程以供进一步处理。
10. 接收到所有数据块并进行编程后，请求 HSMRt 验证在 HSM 活动和休眠存储体中进行编程的代码。当 HSMRt 固件根据证书对编程的映像进行身份验证时，还会对证书进行进一步编程，以便确保在后续的下电上电期间能够成功启动。
11. 成功完成身份验证后，HSM 会将固件编程到 CPU1 闪存。
    a. 如果器件之前处于 HS-KP，则器件将转换为 HS-SE。

请参阅节 5.6，了解在主机应用程序中执行这些操作的步骤。

小心

对于 HS-KP 或 HS-SE 器件，DPL 中断 LINK 和 STACK 指针需要分别设置为 LINK2 和 STACK2。要调整该设置，请在 CCS 中打开 syscfg 文件，选择“TI Driver Porting Layer (DPL)”部分下面的 *Clock*。需要将硬件安全模块部分所述的 Keywriter 二进制映像需要用作 HSMRt。

3.5 SECCFG 代码预置（HS-KP/SE 至 HS-SE）

要对 HS-KP 或 HS-SE 器件执行 SECCFG 编程，请编译工程的 CP_APP 构建配置。可以通过右键点击工程，将鼠标悬停在 Build Configurations 上并选择 CP_APP 来完成此操作。

要将新映像编程到 HS-KP 或 HS-SE 器件上的 SECCFG，会发生以下事件：

1. UART 引导模式下的 BootROM 接收 UART 闪存内核并引导内核。
2. CPU1 中的内核接收一个命令数据包，从而接收 HSMRt 映像。
3. 内核准备作为合并映像的一部分从主机接收 X.509 证书。
4. 内核确认传入证书具有适当的大小和格式，并得出传入映像的大小。证书暂时存储在 RAM 中。
5. 内核将 HSMRt 映像存储在共享 LDAx RAM 中，并请求 HSM 进行身份验证。
6. 身份验证成功后，HSM 开始在共享 LDAx RAM 中执行 HSMRt。
7. 内核接收一个命令数据包，从而接收 SEC CFG 映像。
8. 内核接收 X.509 映像证书并将其与 HSMRt 分享。
9. 成功对映像进行身份验证后，HSMRt 通过确认消息进行响应，之后闪存内核开始通过 UART 将 SEC CFG 数据导入 LDAx 存储器

10. 在接收所有 SecCfg 数据并对其进行编程后，请求 HSMRt 验证在休眠存储体中编程的 SecCfg，确保计数器值有效。当 HSMRt 根据证书对编程的映像进行身份验证时，还会对证书进行进一步编程，以便确保在后续的下电上电期间能够成功引导。
    a. 注意，对于 HS-SE 器件，根据 SSU 寄存器的交换值作出证书的编程决策。

请参阅节 5.6，了解在主机应用程序中执行这些操作的步骤。

小心

对于 HS-KP 或 HS-SE 器件，DPL 中断 LINK 和 STACK 指针需要分别设置为 LINK2 和 STACK2。要调整该设置，请在 CCS 中打开 syscfg 文件，选择“TI Driver Porting Layer (DPL)”部分下面的 *Clock*。需要将节 2.1 中所述的 Keywriter 二进制映像用作 HSMRt。

4 主机应用程序：UART 闪存编程器

4.1 概述

UART 闪存编程器是一个命令行界面程序，在主机 PC 上运行，并与目标器件上的 bootROM 或 UART 闪存内核连接。该程序可以轻松地集成到脚本环境中，用于诸如生产线编程之类的应用。

UART 闪存编程器用 C++ 编写，可在 Windows® 或 Linux® 中作为 Microsoft Visual Studio® 或 CMake 工程进行构建。工程和源代码可以在 SDK 的工具目录中找到 (*f29h85x-sdk_x_xx_xx_xx* > *tools*> *flash_programmers* > *uart_flash_programmer*)。

提供了两个预编译的 Windows 可执行文件：

- uart_flash_programmer.exe (x86_64)：该程序首先在 UART 引导模式下将 SBL (辅助引导加载程序，在本例中为 UART 闪存内核) 映像发送到 BootROM，随后 BootROM 将器件控制移交给 SBL。然后，系统会提示用户选择在内核中通信和执行命令的选项。
  - **备注**
    BootROM 遵循严格的状态机序列，要求在任何 HSM 服务之前进行 SBL 引导，因此，请使用这个可执行文件进行有关 HSM 的操作。例如，使用这个可执行文件进行密钥预置和代码预置。
  - 有关 BootROM 的详细信息，请参阅节 2.2。
- uart_flash_programmer_appIn.exe (x86_64)：该程序绕过将内核发送到 BootROM，并直接提示用户使用命令选项。这对于调试自定义内核很有用。
  - 用户可以通过 CCS 或其他映像加载方法将内核直接加载到器件。
  - 用户可以取消定义*内核*宏，以便包括 Common.h 文件，并重新编译工程以生成使用此行为的可执行文件。

对于 Linux 用户，提供了 shell 脚本 *build_cmake.sh*，以便自动执行 CMake 构建。默认源代码生成一个与 uart_flash_programmer.exe 相同的可执行文件。

若要使用此工具对 F29H85x 器件进行编程，请确保目标器件已复位且处于 UART 引导模式，并且 UART 引脚通过 UART 收发器连接至主机 PC 串行端口。有关设置详细信息，请参阅节 5.1。

通过提供 -h 或 --help 作为参数，可以显示支持的参数。

```
Syntax:
uart_flash_programmer.exe -d f29h85x -p <COM/tty Port> -k <uart kernel image>.bin -a <CPU1
application image>.bin -e <F29x alternate entry address>-r <HSM runtime image>.bin -f <user HSM
keys>.bin -t <CPU1 application image>.bin -g <HSM application image>.bin -c <sec cfg program
image>.bin -q -w
```

表 4-1. 支持的参数

-d，--device <device>	要连接和加载的器件的名称。 目前，F29H85x 是唯一支持的器件。
-k，--kernel <file>	CPU1 闪存内核的文件名
-a，--appcpu1 <file>	要通过 DFU 为 HS-FS 器件下载的 CPU1 应用程序映像的文件名。
-r，--hsmrt <file>	基于 RAM 的 HSM 运行时映像的文件名。在密钥预置和代码预置之前加载运行时映像需要此文件名。
-f， --hsmkeys <file>	用于将器件转换为 HS-KP 的 HSM 证书密钥映像的文件名。
-t，--cpappcpu1 <file>	对于 HS-KP/HS-SE 器件的代码预置，这是基于闪存的 CPU1 应用程序映像的文件名。
-g，--cpapphsm <file>	对于 HS-KP/HS-SE 器件的代码预置，这是基于闪存的 HSM 应用程序映像的文件名。
-s， --cpseccfg <file>	用于通过代码预置对非主闪存中的 SEC CFG 区段进行编程的映像的文件名。
-e，--entry <hex_num>	用于覆盖 C29 CPU1 应用程序的默认入口地址的可选参数。例如，对于十六进制地址 0x10001000，传递 10001000。TI 建议使用默认的 10001000 作为入口地址，因为这是 bootROM 闪存入口点。
-h，--help	显示帮助对话框。
-q， --quiet	安静模式。禁止所有非必要的打印输出。

表 4-1. 支持的参数（续）

-l， --log <file>	日志模式。重定向 指定文件的所有非必要打印输出。如果已指定，则覆盖静默模式。
-w	退出前等待按键。

-d、-p、-k 是必需的参数。

备注

编程到 F29H85x 中的所有文件都必须采用二进制格式并与 X.509 证书结合使用。二进制文件的前 0x1000 个字节必须包含证书。有关编译后步骤的说明，请参阅#none#。

4.2 使用 Visual Studio 构建 UART 闪存编程器

UART 闪存编程器可以作为 Visual Studio 工程进行编译和运行。

1. 导航至 uart_flash_programmer 目录。
2. 双击 uart_flash_programmer.sln 打开 Visual Studio 工程。
3. Visual Studio 打开后，依次选择“Build”→“Build Solution”。
4. Visual Studio 编译完毕后，依次选择“Debug”→“uart_flash_programmer properties”。
5. 依次选择“Configuration Properties”→“Debugging”。
6. 选中“Command Arguments”旁的输入框。
7. 键入参数。节 4.1 描述了这些参数。
   - 示例：

```
-d f29h85x -k flash_kernel.bin -a application.bin -p COM34
```

8. 依次点击“Apply”和“OK”。
9. 依次选择“Debug”→“Start Debugging”以开始运行工程。
10. Visual Studio 命令提示符的输出需要与图 5-1 中显示的内容相匹配。

4.3 使用 CMake 构建 UART 闪存编程器

提供了 shell 脚本 build_cmake.sh 来自动进行 CMake 工程构建。该脚本包含以下构建 CMake 工程的步骤：

- 使用 CMake 进行构建的说明：
  - 在终端中，通过以下命令继续创建 build 文件夹并进入文件夹
    - **mkdir build && cd build**
  - 通过以下命令生成 CMake 构件
    - **cmake -S .. -DCMAKE_BUILD_TYPE={Debug/Release}**
  - 通过以下命令构建 CMake 工程
    - **cmake --build**

在 build 文件夹中生成的可执行文件名为 *uart_flash_programmer*。

由 Visual Studio 和 CMake 生成的可执行文件具有完全相同的功能，并且使用方式相同。接下来我们来看看 uart_flash_programmer。

4.4 数据包格式

数据包以标准格式在主机和器件之间发送。数据包允许发送可变数量的数据，同时确保数据包的正确传输与接收。报头、报尾和校验和字段有助于确保数据在传输过程中不被破坏。校验和是命令和数据字段中字节的总和。

请注意，UART 闪存内核和主机编程器都使用共享 *f29h85x_kernel_commands_cpu1.h* 来同步数据包宏，例如报头、报尾、NAK、ACK、命令和状态错误值。用户可以创建新的宏或对现有宏进行修改。

表 4-2. 数据包格式

接头	数据长度	命令	数据	校验和	报尾
2 字节	2 字节	2 字节	长度字节	2 字节	2 字节
0x1BE4	数据长度（以字节表示）	命令	数据	命令和数据校验和	0xE41B

主机和器件都用 ACK 或 NAK 响应数据包。

表 4-3. ACK 或 NAK 值

ACK	NAK
0x2D	0xA5

4.5 内核命令

表 4-4 中提供了命令及相关内核行为的简要说明。

表 4-4. CPU1 内核命令流

内核命令	命令代码	说明
DFU CPU1	0x01	1.接收没有任何数据的命令数据包 2.逐字节接收闪存应用程序 3.编程并验证应用程序 4.发送闪存状态数据包 5.发送关于最终状态的消息
加载 HSMRt 映像	0x0B	1.接收没有任何数据的命令数据包 2.逐字节接收 HSMRt 3.将 HSMRt 放置在共享 LDAx RAM 中 4.发送闪存状态数据包 5.等待接收 HSM 客户端的状态 6.发送关于最终状态的消息
加载 HSM 密钥	0x0E	1.接收没有任何数据的命令数据包 2.逐字节接收 HSMRt 3.将密钥证书放置在共享 LDAx RAM 中，以便 HSM 可以对其进行编程 4.等待 IPC 消息形式的 HSM 身份验证状态 5.发送状态数据包 6.转发来自 HSM 的状态日志消息
负载 HSM 代码预置映像（适用于 HSM 的固件升级）	0x0D	1.接收没有任何数据的命令数据包 2.逐字节接收 HSM 固件，将包括的密钥证书发送给 HSM 进行验证 3.继续将固件存储在共享 LDAx RAM 中，以便 HSM 按块进行验证（如果固件超过 RAM 的大小） 4.等待 HSM 对编程的固件执行完整性检查 5.发送闪存状态数据包 6.发送状态日志消息
加载 C29 代码预置映像（适用于 HS-SE C29 的固件升级） 对 SECCFG 区段进行编程 (HS-SE)	0x10 0x0C	与 HSM 代码预置映像相同 1.接收没有任何数据的命令数据包 2.逐字节接收 SECCFG 映像并存储在共享 RAM 中，以便 HSM 进行验证 3.发送状态数据包 4.发送状态日志消息
Run CPU1	0x09	1.接收没有任何数据的数据包 2.分支到应用程序入口点
Reset CPU1	0x0A	1.接收没有任何数据的数据包 2.启用看门狗并允许其引发复位

5 示例用法

F29H85x MCU SDK 中提供了上述内核：*mcu_sdk_f29h85x/examples/driverlib/single_core/flash/uart_flash_kernel*。主机应用程序也可以在同一 SDK 中找到 (*mcu_sdk_f29h85x /tools/ flash_programmers / uart_flash_programmer*)。本节详细介绍了如何结合使用内核和主机编程器在 F29H85x 器件上执行固件升级。

5.1 将闪存内核加载到器件中

在 F29H85x 上执行固件升级的第一步是通过 ROM 引导加载程序将闪存内核加载到 RAM。确保为所需的行为加载适当的构建配置。例如，对于 HS-FS 器件的 CPU1 DFU，请确保在编译内核时选择 CPU1_APP 构建配置。实现此目的需要遵循以下步骤：

1. 配置引导模式选择引脚以将器件置于 UART 引导模式。
2. 复位器件。
3. 通过 UART 主机编程器将内核发送到器件。

下面将进一步详细介绍这些步骤。

5.1.1 硬件设置

正确设置器件，使其能够与运行 UART 闪存编程器的主机 PC 进行通信。

1. 首先要做的是确保正确配置引导模式选择引脚，以将器件引导至 UART 引导模式。
2. 接下来，将相应的 UART 引导加载程序 GPIO 引脚连接到与主机 PC 串行端口相连的 Rx 和 Tx 引脚。通常需要收发器来将虚拟串行端口从 PC 转换为可以连接到器件的 GPIO 引脚。在某些系统（例如 controlSOM）上，使用 FTDI 芯片将用于 UART 通信的 GPIO 引脚连接到 USB 虚拟串行端口。请参阅 controlSOM 的器件特定用户指南，获取有关启用 UART 通信所需的开关配置的信息。
3. 正确设置硬件以与主机进行通信后，复位器件。该操作是为了将器件启动至 UART 引导模式。

5.1.2 运行 UART 闪存编程器

备注

TI 建议在运行 UART 闪存编程器之前复位器件，以便 bootROM 在等待 UART 数据时不会超时。

1. 导航到包含已编译的 uart_flash_programmer 可执行文件的文件夹。
2. 使用以下命令结构运行可执行文件 uart_flash_programmer.exe：

```
:> uart_flash_programmer.exe -d f29h85x -p COM41 -k ex3_uart_flash_kernel.bin --appcpu1
c29_cpu1_application.bin --hsmrt HSM_runtimeImage.bin --hsmkeys HSM_customKeyCert.bin --
cpseccfg sec_cfg_cert.bin --cpappcpu1 c29_cpu1_application.bin --cpapphsm hsm_application.bin
```

   程序会自动连接到器件并将 CPU1 内核（-k 选项）下载到 RAM 中。如果证书有效且身份验证成功，则器件开始在 LPAx RAM 中执行内核。现在，CPU1 内核正在运行并等待来自主机的数据包。
3. uart_flash_programmer 将选项输出到屏幕上以供选择，而这些选项会发送到器件内核（请参阅图 5-1）。
4. 在前一个操作完成后，重新输出相同的选项。

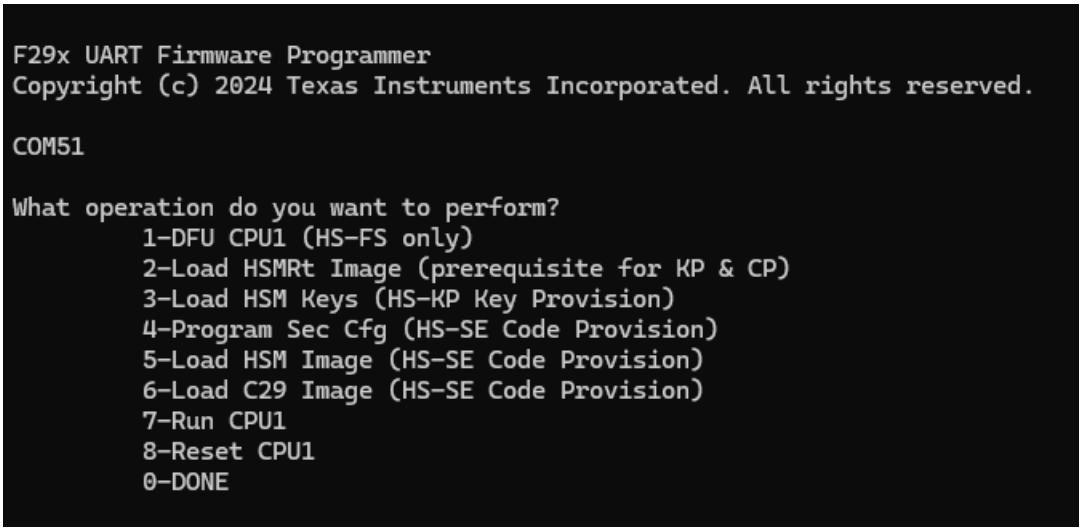

图 5-1. UART 闪存编程器将闪存内核下载到 RAM 后提示输入下一个命令

5.2 CPU1 器件固件升级（仅限 HS-FS）

将 UART 闪存内核加载到器件 RAM 后，可以执行器件固件升级。本节讨论如何在 HS-FS 器件上执行此操作。

图 6-1 展示了在 RAM 中运行闪存内核时可用的选项。要在 HS-FS 器件上进行固件升级，请选择 DFU CPU1（选项 1）。成功后，选择 Run CPU1（选项 7）以从内核分支到新加载的应用程序。或者，如果器件设置为闪存引导模式，则 reset CPU1（选项 8）也有效。

5.3 将 HS-FS 转换为 HS-SE

接下来的章节以及之后的章节讨论了如何使用闪存编程器命令将 HS-FS 器件转换为 HS-SE 器件。

总的来说，要转换默认的 HS-FS 器件，用户必须首先：

- 执行密钥预置，该预置将转换为 HS-KP 的中间状态。（密钥已预置，但到目前为止尚未刷写映像）。
- 执行代码预置，在成功刷写到闪存存储体之后，代码预置会将 HS-KP 器件转换为 HS-SE。
  - 作为代码预置的一部分，闪存编程器提供了三个选项来对闪存进行编程：
    - CPU1 闪存
    - HSM 闪存
    - SEC CFG 闪存（非主闪存存储体的一部分）
- 在第一个代码预置且器件处于 HS-SE 状态之后，仍允许通过任何后续代码预置对闪存进行编程。并且器件保持 HS-SE 状态。

尽管提供的三个代码预置选项中的任何一个都会使器件进入 HS-SE 状态，但 TI 建议在 CPU1 闪存或 HSM 闪存编程之前，先对 SEC CFG 进行编程。

在后续的章节中，按时间顺序讨论每个预置流程。

> **小心**
> BootROM 遵循状态机序列，要求在任何 HSM 服务之前进行 SBL 引导，因此，请使用常规的*非 appIn* 版本闪存编程器。请参阅节 4.1，了解两者之间的差异。

5.4 加载基于 RAM 的 HSMRt 映像

对于与密钥预置和代码预置相关的所有功能，都需要加载基于 RAM 的 HSM 运行时 (HSMRt) 映像。

HSMRt 为来自闪存内核的 HSM 请求提供服务，经过身份验证后，对于将器件转换为不同状态所需的一组字段进行编程。有关要编程的字段的详细信息，请参阅节 2.1 中的 *OTP Keywriter* 部分。

在 UART 引导模式下通过 BootROM 将闪存内核引导到器件后，选择 Load HSMRt Image（选项 2）。主机将基于 RAM 的 HSMRt 映像发送到 C29 CPU1，随后由 HSM 进行验证。成功验证后，这个基于 RAM 的 HSMRt 开始在共享 RAM 中执行。

请注意，密钥预置和代码预置需要两个单独的 HSMRt 二进制文件，每个二进制文件具有不同的密钥证书。在 HS-FS 的密钥预置中，密钥证书需要是 TI 提供的默认密钥，而代码预置使用用户的自定义密钥证书。

5.5 密钥预置（HS-FS 至 HS-KP）

如前所述，密钥预置可通过提供用户密钥证书来替代 TI 提供的默认密钥证书，从而将 HS-FS 器件永久转换为 HS-KP。

闪存编程器上的过程如下：

- 启动闪存编程器，等待闪存内核下载到器件上，并使用图 6-1 中列出的选项提示用户。
- 选择 Load HSMRt Image（选项 2），等到完成。
- 选择 Load HSM keys（选项 3），等到完成。
- 执行上电复位。

在此过程中，因为器件仍处于 HS-FS 状态，HSMRt 可以使用默认的 TI 签名密钥证书。在 HS-FS 状态下，闪存内核不会进行身份验证，因此任何证书都适用，并且需要通过自定义密钥对 HSM 密钥证书进行签名。

所需的参数包括 -d (--device)、-p (--port)、-k (--kernel)、-r (--hsmrt)、-f (--hsmkeys)。

例如：

```
uart_flash_programmer.exe -d f29h85x -p COM41 --kernel ex3_uart_flash_kernel.bin --hsmrt
HSM_runtimeImage.bin --hsmkeys HSM_customKeyCert.bin
```

有关构建闪存内核的过程和重要说明，请参阅节 3.2。

5.6 代码预置（HS-KP/SE 至 HS-SE）

在密钥预置将器件转换为 HS-KP 后，可以对闪存 CPU1/HSM 应用或 SEC CFG 区段执行代码预置以进入相应的闪存存储体。

闪存编程器上的过程如下：

- 启动闪存编程器，等待闪存内核下载到器件上，并使用图 6-1 中列出的选项提示用户。
- 选择 Load HSMRt Image（选项 2），等到完成。
- 选择三个选项中的任何一个，然后等到完成。
  - Program Sec Cfg（选项 4）
  - Load HSM Image（选项 5）
  - Load C29 CPU1 Image（选项 6）
- 如果需要，继续执行另外两个代码预置选项。

请注意，所有映像（闪存内核、HSMRt、应用程序/安全配置）都需要用户密钥证书作为二进制映像的一部分。请参阅节 2.3，了解有关使用自定义密钥生成密钥证书的说明。

所需的参数包括 -d (--device)、-p (--port)、-k (--kernel) 以及 -t (--cpappcpu1)、-g (--cpapphsm)、-s (--cpseccfg)，具体取决于代码预置选项。

例如：

```
uart_flash_programmer.exe -d f29h85x -p COM41 --kernel ex3_uart_flash_kernel.bin --hsmrt
HSM_runtimeImage.bin --cpseccfg sec_cfg_cert.bin --cpappcpu1 c29_cpu1_application.bin --cpapphsm
hsm_application.bin
```

有关构建闪存内核以进行代码预置的过程和重要说明，请参阅节 3.3。

6 故障排除

下面提供了用户在使用 UART 闪存内核时遇到的一些常见问题的回答。

6.1 一般信息

问题：我找不到 UART 闪存内核工程，它们在哪里？

回答：

器件	编译配置	位置
F29H85x	CPU1_RAM	mcu_sdk_f29h85x\examples\driverlib\single_core\flash\uart_flash_kernel

6.2 UART 引导

问题：我无法在 UART 引导模式下将 UART 内核下载到 RAM，我应该怎么做？

回答：

- 用户遇到的常见问题是未对 UART 引导模式使用正确的引导引脚。例如，在 F29H85x 器件上，UART 引导模式提供五个选项供 GPIO 引脚使用。确保默认选项的引脚没有用于其他用途。如果引脚已使用，则请确保使用另一个 UART 引导选项，以便器件能够连接到另一组引脚。确保 UART 内核工程也使用此 UART 引导 GPIO 选项作为 UART_GetFunction() 的参数。
- 确保二进制文件的前 0x1000 个字节中有关联的 X.509 证书。有关具体说明，请参阅节 2.2。
- 确保用户使用高质量 UART 收发器，以便更大限度地减少波特率方面的问题。
- 对于波特率和连接问题，请尝试为器件运行 UART 环回和回显示例（用户可以在相关器件的 MCU SDK driverlib 文件夹中找到这些示例）。

6.3 应用程序加载

问题：我可以成功将内核下载到器件，但无法成功地将应用程序加载到闪存。我需要检查什么？

回答：

- 将链接器命令文件中所有分配给闪存的区段都对齐到 512 位边界。可以通过在适当的区段中添加 palign(32) 来完成此操作，如下所示。

```
.text : {} > FLASH_RP0, palign(32)
```

- 确保二进制文件的前 0x1000 个字节中有适当的 X.509 证书。有关具体说明，请参阅节 3.1。

7 总结

随着应用程序复杂性的增加，修复错误、添加特性和修改嵌入式固件的需求在最终应用程序中变得越来越重要。通过使用引导加载程序，可以轻松且经济地实现此类功能。

本应用手册旨在通过引入辅助引导加载程序 (SBL) (在本例中为 UART 闪存内核) 来解决这个问题。本文档讨论了 F29H85x MCU SDK 中的内核和主机应用工具的具体细节。

8 参考资料

1. 德州仪器 (TI) : *F29H85x 和 F29P58x 实时微控制器* 技术参考手册
2. 德州仪器 (TI) , F29H85x 和 F29P58x 实时微控制器 数据表

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、某特定用途方面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功能安全、信息安全、监管或其他要求。

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。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/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。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265