Application Note

C2000™ 微控制器的串行闪存编程

Charles Roberson, Sira Rao, Vamsikrishna Gudivada, and Skyler Baumer

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，工程师需要依赖串行编程解决方案。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。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 引言

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。

引导加载程序（也被称为 ROM 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。在大多数情况下，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。

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。本报告中使用的外设是串行通信接口（SCI，一般简称为 UART）。与引导引脚相关的引导模式是指外设的第一个实例 - 对于 SCI，引导模式将与 SCI**A** 相关联。

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)、以太网、CAN 和使用通用输入/输出 (GPIO) 的并行模式将代码加载到片上 RAM 中。这些加载程序的一部分存在于每个 C2000 器件中，它们非常易于使用，但只能将代码加载到 RAM 中。如何弥合这一差距并将其应用程序代码编程到非易失性存储器中？

本应用报告旨在使用闪存内核解决这一问题。闪存内核已经存在一段时间了，但本文档讨论了内核的细节以及 C2000Ware 中的主机应用程序工具。虽然此实施针对使用 SCI 外设的 C2000 器件，但相同的原则适用于 C2000 产品线中的所有器件以及 ROM 加载程序支持的所有通信选项。提供了一个命令行工具来解析应用程序映像并将其从主机 PC（仅限 Windows）传输到嵌入式器件。

总之，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 SCI ROM 引导加载程序将闪存内核下载到 RAM。
2. 在 RAM 中运行闪存内核以将应用程序下载到闪存。

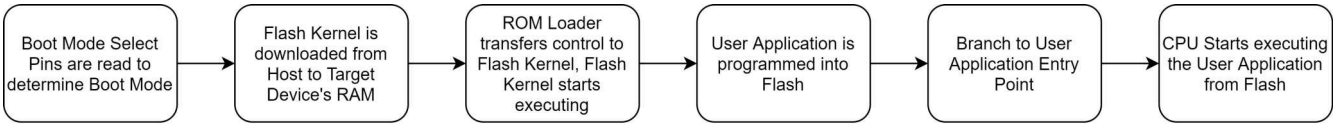

图 1-1. 闪存内核流程

2 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清除为“0”。某些器件上的闪存一次只能擦除一个扇区，而其他器件提供闪存组擦除选项。本报告中使用的术语串行闪存（串行闪存内核、串行闪存编程器等）仅指串行通信接口 (SCI)。

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。

3 ROM 引导加载程序

开始时，器件启动，并根据引导模式决定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。本应用报告重点介绍未连接仿真器时的引导执行路径。

备注

本节基于 TMS320F28004x 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 *引导 ROM* 部分中找到。

表 3-1. F28004x 器件的默认引导模式

引导模式	GPIO24（默认引导模式选择引脚 1）	GPIO32（默认引导模式选择引脚 0）
并行 I/O	0	0
SCI/等待引导	0	1
CAN	1	0
闪存	1	1

在引导 ROM 准备好使用的器件后，它决定应该从哪里开始执行。如果是独立启动，它通过检查两个 GPIO（如表 3-1 中所示，默认选择是 GPIO 24 和 32）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。在本应用报告中所述的实现方案中，使用了 SCI 加载程序，因此在上电时 GPIO 32 必须强制为高电平，GPIO 24 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 SCI 加载程序开始执行并等待通过主机（接收字符以确定以何种波特率进行通信）进行自动波特率锁定。此时，器件已准备好接收来自主机的代码。

ROM 加载程序要求以特定结构向其提供数据。该结构对所有 ROM 加载程序都是通用的，[1]中的 *引导加载程序数据流结构* 部分对此进行了详细介绍。您可以使用 TI C2000 编译器随附的 hex2000 实用程序，轻松生成这种格式的应用程序。通过添加具有以下选项的编译后处理步骤，甚至可以在 Code Composer Studio 编译过程中生成此文件格式：

```
"${CG_TOOL_HEX}" "${BuildArtifactFileName}" -boot -sci8 -a -o "${BuildArtifactFileBaseName}.txt"
```

对于 F2838x 中的 CM 内核，可以使用以下命令：

```
"${CG_TOOL_HEX}" "${BuildArtifactFileName}" -boot -gpio8 -a -o "${BuildArtifactFileBaseName}.txt"
```

或者，您可以使用 TI hex2000 实用程序将 COFF 和 EABI .out 文件转换为正确的十六进制格式引导文件。为此，您需要在“Project Properties”下启用 C2000 Hex Utility。具体命令如下：

```
hex2000.exe -boot -sci8 -a -o <file.txt> <file.out>
```

对于 F2838x 中的 CM 内核，可以使用以下命令：

```
armhex.exe -boot -gpio8 -a -o <file.txt> <file.out>
```

如前所述，ROM 加载程序只能将代码加载到 RAM 中，因此会将其加载到闪存内核中，这将在节 4 和节 5 中进行介绍。

4 闪存内核 A

闪存内核 A 在以下器件上运行：

- TMS320F2802x
- TMS320F2803x
- TMS320F2805x
- TMS320F2806x
- TMS320F2833x

若要查找这些器件的闪存内核工程的位置，请参阅节 7。

4.1 执行

闪存内核 A 基于 SCI ROM 加载程序源。为了使该代码能够擦除和编程闪存，必须合并闪存 API，这是通过链接闪存 API 来完成的。在接收到任何应用程序数据之前，闪存内核会擦除器件的闪存，以为编程做好准备。缓冲区用于保存接收到的连续应用程序代码块。当缓冲区已满或检测到新的非连续数据块时，将对缓冲区中的代码进行编程。此过程一直持续到收到整个应用程序为止。

用于传输应用程序数据的协议与 SCI ROM 加载器协议略有不同。这样做是为了提高编程速度，同时确保可靠通信。使用原有的 SCI ROM 加载器协议，大部分时间都不是用来传输数据，而是等待数据通过操作系统的不同层传播。SCI ROM 加载程序必须一次发送一个字节的数据（原因在于基于回声的流控制），因此每个字节都会导致操作系统传输延迟，这一事实使问题更加复杂。闪存内核使用相同的协议，但计算在每个数据块之后发送的校验和。这允许 PC 端应用程序通过操作系统的不同层一次发送多个字节，从而显著降低通信延迟。

此闪存内核可与 CSM 锁定器件一同使用。如果器件被锁定，串行闪存编程器仍然可以使用，方法是将闪存内核加载到不安全的 RAM 中，并在擦除和编程闪存之前修改内核以解锁器件。在这种情况下，应修改 *CsmUnlock()*（位于 <device_name>_SysCtrl.c 中）以将正确的 CSM 密码写入 CSM 寄存器。这一操作将解锁器件。如果用户在对器件进行编程时不想解锁 CSM，则闪存 API 将必须位于闪存（而非 ROM）中并被复制到安全 RAM 中，以便它们能够在不解锁器件的情况下编程/擦除安全闪存扇区。

4.1.1 应用加载

本节介绍了使用 SCI 引导模式将应用程序编程到闪存的整个流程。

若要确保器件已准备好进行 SCI 通信，需要重置器件同时确保引导模式引脚处于正确状态以选择 SCI 引导模式。随后的步骤如下：

1. 器件接收自动波特率字符，以确定加载操作的波特率。这在主机启动传输命令后不久进行。
2. 闪存内核被传输到器件，等待字符被回显，然后再发送下一个。确保闪存内核已编译并仅链接到 RAM。
3. ROM 转移控制权，闪存内核开始执行。从内核必须让器件准备好进行闪存编程到内核准备好开始通信的期间有些许延迟，在此期间内核会配置 PLL 和闪存等待状态。
4. 内核进入自动波特模式并等待接收自动波特字符。这可能允许内核的通信速度比 ROM 加载程序的速度更高，因为内核将 PLL 配置为更高的速度。SCI 具有自动波特锁，允许主机发送所需的任何波特率 - 双方不需要商定单个波特率。
5. 波特率一旦确定，就可以使用与闪存内核相同的格式下载应用程序。下载过程开始时，在实际应用程序代码之前传输一个密钥、一些保留字段和应用程序入口点。
6. 接收到入口点后，内核开始擦除闪存。擦除闪存可能需要几秒钟，因此请务必注意，虽然看起来应用程序加载可能已失败，但很可能只是闪存被擦除了。
7. 闪存被擦除后，应用程序加载将通过传输每个应用程序代码块并将其编程到闪存而继续。
8. 将数据块编程到闪存中后，校验和将发送回主机 PC，以确保嵌入式器件正确接收到所有数据。此过程一直持续到整个应用程序已被编程到闪存中为止。

应用程序编程到闪存中后，闪存内核会转到其在应用程序加载过程开始时的入口点来尝试运行应用程序。这需要一次器件复位。

5 闪存内核 B

闪存内核 B 在以下器件上运行：

- TMS320F2807x
- TMS320F2837xD
- TMS320F2837xS
- TMS320F28004x
- TMS320F2838x
- TMS320F28002x
- TMS320F28003x
- TMS320F280013x
- TMS320F280015x
- TMS320F28P65x
- TMS320F28P55x

若要查找这些器件的闪存内核工程的位置，请参阅节 7。

5.1 实施

闪存内核 B 比内核 A 更强大。它与 C2000Ware 中提供的主机 PC 应用程序 (*C2000Ware_x_xx_xx_xx* > *utilities* > *flash_programmers* > *serial_flash_programmer*) 进行通信，并就接收数据包和分配给它的命令的完成情况向主机提供反馈。

将内核加载到 RAM 并通过 SCI ROM 引导加载程序执行后，内核首先初始化器件的 PLL，初始化 SCIA，并在必要时控制闪存泵。然后，它等待来自主机的“a”或“A”，以便通过主机执行自动波特率锁定。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 5-1 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 0x12345678。

对于器件固件升级 (DFU) 命令，将执行以下步骤：

备注

这些步骤不适用于 F28P55x。有关 F28P55x 器件的详细信息，请参阅 *F28P55x SCI 闪存内核*。

1. 内核从 SCI 模块逐字节接收十六进制引导格式的文件，并对块大小计算校验和。收到数据块后，它发回校验和。
2. 接收到一个数据块并将其存储在缓冲器中后，内核会擦除之前未擦除过的扇区，并使用闪存 API 库将数据与 ECC 一同编程到闪存中。

备注

若要获取有关数据块以及如何确定块大小的更多信息，请参阅器件特定 TRM 的 *引导加载程序数据流结构* 部分。请注意，这里提到的块和块大小不要与前文提及的命令包相混淆。

3. 之后，内核会验证数据和 ECC 是否正确编程到闪存中。

备注

该内核仅擦除将应用程序和数据编程到闪存中所需的扇区。这与闪存内核 A 不同，它会在内核执行开始时擦除整个闪存。但是，闪存内核 B 还支持独立于 DFU 命令的擦除命令，这使用户能够擦除器件的特定扇区或整个闪存。

类似地，验证命令会接收一个十六进制引导格式的文件，内核只验证闪存的内容，而不是擦除和编程闪存。

5.1.1 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。

表 5-1. 数据包格式

报头	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

主机和器件按一次一个字（16 位）的方式发送数据包，LSB 后跟 MSB。主机和器件都用 ACK 或 NAK 响应数据包。

表 5-2. ACK/NAK 值

ACK	NAK
0x2D	0xA5

5.1.2 CPU1 内核命令

双核 F2837xD 的 CPU1 命令可用于 F2807x、F28004x 和 F2837xS 单核器件内核，不包括 *Run CPU1 Boot CPU2* 和 *Reset CPU1 Boot CPU2*。表 5-3 中提供了命令代码的简要说明。

表 5-3. CPU1 内核命令

内核命令	命令代码	说明
DFU CPU1	0x0100	1.接收没有任何数据的数据包 2.以引导十六进制格式逐字节接收闪存应用程序 3.选择性擦除、编程和验证 4.发送状态数据包 如果成功，则状态包中发送的地址为已编程闪存应用程序的入口点地址
Erase CPU1	0x0300	1.接收包含 32 位数据的数据包（如节 5.1.4 中所述） 2.擦除数据中指定的扇区 3.发送状态数据包
Verify CPU1	0x0500	1.接收没有任何数据的数据包 2.以引导十六进制格式逐字节接收闪存应用程序 3.验证闪存内容 4.发送状态数据包
Unlock CPU1 – Zone 1	0x000A	1.接收包含 128 位数据的数据包（如节 5.1.4 中所述） 2.将密码写入 DC SM 密钥寄存器 3.检查 Zone 1 是否已解锁 4.发送状态数据包
Unlock CPU1 – Zone 2	0x000B	1.接收包含 128 位数据的数据包（如节 5.1.4 中所述） 2.将密码写入 DC SM 密钥寄存器 3.检查 Zone 2 是否已解锁 4.发送状态数据包
Run CPU1	0x000E	1.接收包含 32 位地址的数据包（如节 5.1.4 中所述） 2.分支到 32 位地址
Reset CPU1	0x000F	1.接收没有任何数据的数据包 2.中断 while 循环并使看门狗计时器超时并复位
Run CPU1 Boot CPU2(1)	0x0004	1.接收包含 32 位地址的数据包 2.释放闪存泵，IPC 将 CPU2 引导至 SCI 引导模式，将 SCI 和共享 RAM 的控制权交给 CPU2，然后等待 CPU2 发出信号。 3.分支到地址
Reset CPU1 Boot CPU2 (1)	0x0007	1.接收没有任何数据的数据包 2.释放闪存泵，IPC 将 CPU2 引导至 SCI 引导模式，将 SCI 和共享 RAM 的控制权交给 CPU2，然后等待 CPU2 发出信号。 3.中断 while 循环并使看门狗计时器超时并复位。

(1) 此命令不适用于 F2807x、F2837xS 和 F28004x 单核器件内核。

5.1.3 CPU2 内核命令

表 5-4 显示了双核 F2837xD 器件内核上使用的 CPU2 命令的功能、命令代码和说明。

表 5-4. CPU2 内核命令

内核命令	命令代码	说明
DFU CPU2	0x0200	1.接收没有任何数据的数据包 2.以引导十六进制格式逐字节接收闪存应用程序 3.选择性擦除、编程和验证 4.发送状态数据包 如果成功，则状态包中发送的地址为已编程闪存应用程序的入口点地址
Erase CPU2	0x0400	1.接收包含 32 位数据的数据包（如节 5.1.4 中所述） 2.选择性擦除数据中指定的扇区 3.发送状态数据包
Verify CPU2	0x0600	1.接收没有任何数据的数据包 2.以引导十六进制格式逐字节接收闪存应用程序 3.验证闪存内容 4.发送状态数据包
Unlock CPU2 – Zone 1	0x000C	1.接收包含 128 位数据的数据包（如节 5.1.4 中所述） 2.将密码写入 DC SM 密钥寄存器 3.检查 Zone 1 是否已解锁 4.发送状态数据包
Unlock CPU2 – Zone 2	0x000D	1.接收包含 128 位数据的数据包（如节 5.1.4 中所述） 2.将密码写入 DC SM 密钥寄存器 3.检查 Zone 2 是否已解锁 4.发送状态数据包
Run CPU2	0x0010	1.接收包含 32 位地址的数据包（如节 5.1.4 中所述） 2.分支到 32 位地址
Reset CPU2	0x000F	1.接收没有任何数据的数据包 2.中断 while 循环并使看门狗计时器超时并复位

5.1.4 数据包数据

本节介绍了需要将数据发送到器件的命令的预期数据。

- ***擦除***
  与擦除命令一同发送的 32 位数据的每一位对应一个扇区。
  - 数据位 0 – 扇区 A
  - 数据位 1 – 扇区 B
  - 等等

表 5-5. 清除数据包

报头	数据长度	命令	数据	校验和	报尾
0x1BE4	6（字节）	0x0300 CPU1 0x0400 CPU2	32 位数据（1 表示擦除，0 表示不擦除）	命令和数据校验和	0xE41B

- ***解锁***
  - 第 1 个 32 位是密钥 1
  - 第 2 个 32 位是密钥 2
  - 第 3 个 32 位是密钥 3
  - 第 4 个 32 位是密钥 4

表 5-6. 解锁数据包

报头	数据长度	命令	数据	校验和	报尾
0x1BE4	4（字节）	0x000A CPU1 Z1 0x000B CPU1 Z2 0x000C CPU2 Z1 0x000D CPU2 Z2	128 位数据	命令和数据校验和	0xE41B

- *运行*
  - 32 位地址

表 5-7. 运行数据包

报头	数据长度	命令	数据	校验和	报尾
0x1BE4	4（字节）	0x000E CPU1 0x0020 CPU2	32 位数据	命令和数据校验和	0xE41B

5.1.5 状态代码

某个命令完成后，内核会向主机发送一个与主机到器件数据包格式相同的状态数据包。这可让主机知道是否发生错误、错误类型以及错误发生的位置。命令字段是上次完成的命令。数据字段包含一个 16 位状态代码，后跟一个发生错误的 32 位地址。如果没有错误，地址会是 0x12345678，除非它响应 DFU 命令（在这种情况下，地址是刚刚编程到闪存中的应用程序十六进制引导格式文件的入口点地址）。该地址随后可用于 RUN 命令，后者会告知 CPU 跳转到哪个地址并开始执行代码。

表 5-8 显示了状态代码。

表 5-8. 状态代码

状态代码	值	说明
NO_COMMAND_ERROR	0x1000	返回成功的结果
BLANK_ERROR	0x2000	返回擦除错误的结果
VERIFY_ERROR	0x3000	返回验证错误的结果
PROGRAM_ERROR	0x4000	返回编程错误的结果
COMMAND_ERROR	0x5000	返回命令无效的结果
UNLOCK_ERROR	0x6000	返回解锁失败的结果

对于 Program、Erase 和 Verify 命令，如果检测到闪存 API 错误，则会将其显示在控制台上。如果检测到 FMSTAT 错误，将显示 FMSTAT 寄存器内容。

状态代码	值
INCORRECT_DATA_BUFFER_LENGTH	0x7000
INCORRECT_ECC_BUFFER_LENGTH	0x8000
DATA_ECC_BUFFER_LENGTH_MISMATCH	0x9000
FLASH_REGS_NOT_WRITABLE	0xA000
FEATURE_NOT_AVAILABLE	0xB000
INVALID_ADDRESS	0xC000
INVALID_CPUID	0xD000
FAILURE	0xE000

有关这些错误的更多详细信息，请参阅器件特定闪存 API 指南。

下一节将概述 F2838x SCI 闪存内核及其使用方法。

5.2.1 CPU1-CPU2 内核

F2838x CPU1-CPU2 内核工程利用 CPU1 中的 SCI ROM 引导加载程序下载 CPU1 应用程序映像，并使用修改后的引导加载程序下载 CPU2 应用程序映像。CPU1 和 CPU2 内核和应用程序文件作为参数输入到 serial_flash_programmer 实用程序，一旦主机通过 CPU1 SCI ROM 引导加载程序完成了自动波特率锁定，就会下载 CPU1 内核。CPU1 内核下载完毕后，即可开始正常的内核操作，例如 DFU、擦除等。

CPU1 操作完成后，可以通过选择“Run CPU1 Load CPU2”或“Reset CPU1 Load CPU2”命令来下载 CPU2 内核。对于 F2838x，CPU2 没有 SCI ROM 引导加载程序，因此 CPU1 将 CPU2 内核写入 CPU2 的共享 RAM。还有一条指令用以跳转到 CPU1TOCPU2MSGRAM 中的 CPU2 入口点，该指令在 CPU2 引导序列期间复制到 M1RAM 中。在将分支指令写入 M1RAM 且 CPU2 引导序列完成后，CPU2 从 M1RAM 开始执行并跳转到内核入口点。CPu2 内核下载完毕后，即可开始正常的内核操作。CPU1 内核将等待 CPU2 完成其内核命令，然后再根据是否选择了“Run CPU1 Load CPU2”或“Reset CPU1 Load CPU2”进行应用程序寻址或复位。

CPU1 提供 9 个选项

1.DFU
2.Erase（擦除）
3.Verify（验证）
4.Unlock Zone 1（解锁 Zone 1）
5.Unlock Zone 2（解锁 Zone 2）
6.Run（运行）
7.Reset（复位）
8.Run CPU1 Load CPU2（运行 CPU1 加载 CPU2）
9.Run CPU1 Load CPU2（运行 CPU1 加载 CPU2）

CPU2 提供 7 个选项

1.DFU
2.Erase（擦除）
3.Verify（验证）
4.Unlock Zone 1（解锁 Zone 1）
5.Unlock Zone 2（解锁 Zone 2）
6.Run（运行）
7.Reset（复位）

5.2.1.1 内核命令

大多数命令与以前的器件相同，CPU1 有两个新命令：Run CPU1 Load CPU2 和 Reset CPU1 Load CPU2。

表 5-9 介绍了每条命令的工作原理。

表 5-9. 内核命令

内核命令	命令代码	说明
Run CPU1 Load CPU2	0x0004	1.接收包含 32 位地址的数据包。 2.将 CPU2 内核加载到 CPU2 内核链接器命令文件所指定的 GSRAM 段。 3.将 SCI 和共享 RAM 的控制权转移到 CPU2。 4.释放闪存泵并通过 IPC 为 CPU2 设置引导模式。 5.等待 CPU2 信号。 6.分支到地址。
Reset CPU1 Load CPU2	0x0007	1.接收没有任何地址的数据包。 2.将 CPU2 内核加载到 CPU2 内核链接器命令文件所指定的 GSRAM 段。 3.将 SCI 和共享 RAM 的控制权转移到 CPU2。 4.释放闪存泵并通过 IPC 为 CPU2 设置引导模式。 5.等待 CPU2 信号。 6.中断 while 循环并使看门狗计时器超时并复位。

5.2.2 CPU1-CM 内核

F2838x CPU1-CM 内核工程利用 SCI 引导加载程序下载 CPU1 应用程序映像，并使用修改后的引导加载程序下载 CM 应用程序映像。CPU1 和 CM 内核和应用程序文件作为参数输入到 serial_flash_programmer 实用程序，一旦主机通过 CPU1 SCI ROM 引导加载程序完成了自动波特率锁定，就会下载 CPU1 内核。CPU1 内核下载完毕后，即可开始正常的内核操作，例如 DFU、擦除等。

CPU1 操作完成后，可通过选择“Run CPU1 Load CM”或“Reset CPU1 Load CM”下载 CM 内核。CPU1 首先写入下载内核所需的 CM 引导模式。在为 CM 设置引导模式后，CPU1 跳转至一个复制函数，以便接收通过 SCI 从主机发送的 CM 内核，并将其复制到 CPU1-CM IPC 消息 RAM 中的缓冲区。一旦填满缓冲区，它就会向 CM 发送信号，然后 CM 会将缓冲区的内容复制到 CM RAM 中的预期地址。在 CM 引导序列期间，与 CPU1 复制函数进行通信的 CM 侧的复制函数被写入 S0RAM。此函数是使用 ARM 编译器编译的，并作为常量数组存储在 CPU1-CM 消息 RAM 中。CPU1 和 CM 复制函数一同使用以复制 CM 内核，直到所有内容都已写入。然后，CPU1 将 CM 内核入口地址发送给 CM，然后 CM 跳转到该地址以开始内核执行。然后，用户可以执行任何所需的 CM 命令。

CPU1 保持对 SCI 外设的控制，并在 CM 提出请求时发送其内核函数所需的 CM 数据。CM 内核可向 CPU1 内核指示它需要执行某个函数，例如 sciaGetWordData。CPU1 将获得执行该函数的结果并将其放入消息 RAM 中，以便 CM 复制并用于其所有命令。

CPU1 有 9 个选项	CM 有 7 个选项
1.DFU	1.DFU
2.Erase（擦除）	2.Erase（擦除）
3.Verify（验证）	3.Verify（验证）
4.Unlock Zone 1（解锁 Zone 1）	4.Unlock Zone 1（解锁 Zone 1）
5.Unlock Zone 2（解锁 Zone 2）	5.Unlock Zone 2（解锁 Zone 2）
6.Run（运行）	6.Run（运行）
7.Reset（复位）	7.Reset（复位）
8.Run CPU1 Load CM（运行 CPU1 加载 CM）	
9.Run CPU1 Load CM（运行 CPU1 加载 CM）	

5.2.2.1 内核命令

CPU1 有两个针对 F2838x SCI 闪存内核的新命令，表 5-10 中列出了这两个新命令。

表 5-10. CPU1 内核命令

内核命令	命令代码	说明
Run CPU1 Load CM	0x0030	1.接收包含 32 位地址的数据包。 2.释放闪存泵并通过 IPC 为 CM 设置引导模式。 3.通过 CPU1-CM IPC 消息 RAM 将 CM 内核写入 CM RAM。 4.等待 CM 信号并在请求时为 CM 执行 SCI 功能。 5.分支到地址。
Run CPU1 Load CM	0x0040	1.接收包含 32 位地址的数据包。 2.释放闪存泵并通过 IPC 为 CM 设置引导模式。 3.通过 CPU1-CM IPC 消息 RAM 将 CM 内核写入 CM RAM。 4.等待 CM 信号并在请求时为 CM 执行 SCI 功能。 5.中断 while 循环并使看门狗计时器超时并复位。

表 5-11 中列出了 CM 命令。

表 5-11. CM 内核命令

内核命令	命令代码	说明
DFU CM	0x0050	1.接收没有任何数据的数据包。 2.以引导十六进制格式逐字节接收闪存应用程序。 3.选择性擦除、编程和验证。 4.发送状态数据包。 如果成功，则状态包中发送的地址为已编程闪存应用程序的入口点地址。
Erase CM	0x0060	1.接收包含 32 位数据的数据包。 2.选择性擦除数据中指定的扇区。 3.发送状态数据包。
Verify CM	0x0070	1.接收没有任何数据的数据包。 2.以引导十六进制格式逐字节接收闪存应用程序。 3.以引导十六进制格式逐字节接收闪存应用程序。 4.发送状态数据包。

表 5-11. CM 内核命令（续）

内核命令	命令代码	说明
CM Unlock Zone 1	0x0080	1.接收包含 128 位数据的数据包。 2.将密码写入 DCSM 密钥寄存器。 3.检查 Zone 1 是否已解锁。 4.发送状态数据包。
CM Unlock Zone 2	0x0090	1.接收包含 128 位数据的数据包。 2.将密码写入 DCSM 密钥寄存器。 3.检查 Zone 2 是否已解锁。 4.发送状态数据包。
Run CM	0x00A0	1.接收包含 32 位地址的数据包。 2.分支到 32 位地址。
Reset CM	0x00B0	1.接收没有任何数据的数据包。 2.中断 while 循环并使看门狗计时器超时并复位。

5.2.3 使用 SCI 引导加载程序下载工程

本节详细介绍了在 controlCard 上运行 F2838x 内核所需的步骤。

5.2.3.1 CPU1-CPU2

1. 将 controlCard S1:A 位置 1 设为 OFF（左侧）。这会关闭 JTAG 通信。
2. 将 controlCard S1:A 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 将 S2 位置 1 设为 OFF（左侧），将位置 2 设为 ON（右侧）。这会将 controlCard 设置为 SCI 引导模式。
4. 打开命令窗口并导航至 serial_flash_programmer.exe 所在的位置。
5. 输入带有参数的命令，如下所述：

   示例：serial_flash_programmer.exe -d f2838x -k flash_kernel_c28x_dual_ex1_c28x1.txt -a led_ex1_c28x_dual_blinky_cpu1.txt -m flash_kernel_c28x_dual_ex1_c28x2.txt -n led_ex1_c28x_dual_blinky_cpu2.txt -b 9600 -p COM10 -v
6. 当内核下载完毕并弹出菜单时，执行所有 CPU1 操作，然后再继续执行 CPU2。重新启动 serial_flash_programmer 应用程序会将控制权交还给 CPU1，因此 CPU1 和 CPU2 操作可在运行 serial_flash_programmer 应用程序的同时执行。

5.2.3.2 CPU1-CM

1. 将 controlCard S1:A 位置 1 设为 OFF（左侧）。这会关闭 JTAG 通信。
2. 将 controlCard S1:A 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 将 S2 位置 1 设为 OFF（左侧），将位置 2 设为 ON（右侧）。这会将 controlCard 设置为 SCI 引导模式。
4. 打开命令窗口并导航至 serial_flash_programmer.exe 所在的位置。
5. 输入带有参数的命令，如下所述：

   示例：serial_flash_programmer.exe -d f2838x -k flash_kernel_c28x_cm_ex1_c28x1.txt -a led_ex1_c28x_cm_blinky_cpu1.txt -o flash_kernel_c28x_cm_ex1_cm.txt -r led_ex1_c28x_cm_blinky_cm.txt -b 9600 -p COM10 -v
6. 当内核下载完毕并弹出菜单时，执行所有 CPU1 操作，然后再继续执行 CM。执行 CM 操作后退出应用程序。重新启动 serial_flash_programmer 应用程序会将控制权交还给 CPU1，因此 CPU1 和 CM 操作可在运行 serial_flash_programmer 应用程序的同时执行。

5.2.4 使用 Code Composer Studio (CCS) 软件编译工程

5.2.4.1 CPU1-CPU2

1. 将 controlCard S1:A 位置 1 设为 ON（右侧）。这会开启 JTAG 通信。
2. 将 controlCard S1:A 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 在 Code Composer Studio™ IDE 中，导入并编译 CPU1 和 CPU2 内核工程。

4. 启动目标配置文件。
5. 连接至 CPU1。
6. 将工程文件夹中提供的 gel 文件加载到工程中。右键点击目标配置中的 CPU1，然后选择“Open GEL Files View”。
7. 在“GEL Files”选项卡中，点击“GEL Files”。在“Script”窗口中右键点击，然后选择“Load GEL...”。导航至工程文件夹并加载 gel 文件。
8. 连接至 CPU2。
9. 点击 CPU1 对应的“Reset”。CPU2 现处于等待引导中。
10. 点击 CPU2 对应的“Resume”。
11. 点击 CPU1 对应的“Load Program”。
12. 加载 CPU1 内核的 .out 文件。文件加载完毕后，点击“Resume”。
13. 打开命令窗口并导航至 serial_flash_programmer_appln.exe 所在的位置。
14. 输入带有参数的命令，如下所述：

    示例：serial_flash_programmer_appln.exe -d f2838x -d f2838x -k flash_kernel_c28x_dual_ex1_c28x1.txt -a led_ex1_c28x_dual_blinky_cpu1.txt -m flash_kernel_c28x_dual_ex1_c28x2.txt -n led_ex1_c28x_dual_blinky_cpu2.txt -b 9600 -p COM10 -v
15. 当弹出菜单时，执行所有 CPU1 操作，然后再继续执行 CPU2。

5.2.4.2 CPU1-CM

1. 将 controlCard S1:A 位置 1 设为 ON（右侧）。这会开启 JTAG 通信。
2. 将 controlCard S1:A 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 在 CCS 中，导入并编译 CPU1 和 CM 内核工程。
4. 启动目标配置文件。
5. 连接至 CPU1。
6. 将工程文件夹中提供的 gel 文件加载到工程中。右键点击目标配置中的 CPU1，然后选择“Open GEL Files View”（打开 GEL 文件视图）。
7. 在“GEL Files”（GEL 文件）选项卡中，点击“GEL Files”（GEL 文件）。在“Script”（脚本）窗口中右键点击，然后选择“Load GEL...”（加载 GEL...）。导航至工程文件夹并加载 gel 文件。
8. 连接至 CM。
9. 点击 CPU1 对应的“Reset”（重置）。CM 现处于等待引导中。
10. 点击 CM 对应的“Resume”（恢复）。
11. 点击 CPU1 对应的“Load Program”（加载程序）。
12. 加载 CPU1 内核的 .out 文件。文件加载完毕后，点击“Resume”（恢复）。
13. 打开命令窗口并导航至 serial_flash_programmer_appln.exe 所在的位置。
14. 输入带有参数的命令，如下所述。

    示例：serial_flash_programmer_appln.exe -d f2838x -k flash_kernel_c28x_cm_ex1_c28x1.txt -a led_ex1_c28x_cm_blinky_cpu1.txt -o flash_kernel_c28x_cm_ex1_cm.txt -r led_ex1_c28x_cm_blinky_cm.txt -b 9600 -p COM10 -v
15. 当弹出菜单时，执行所有 CPU1 操作，然后再继续执行 CM。执行 CM 操作后退出应用程序。重新启动 serial_flash_programmer 应用程序会将控制权交还给 CPU1，因此 CPU1 和 CM 操作应在运行 serial_flash_programmer 应用程序的同时执行。

下一节将概述 F28P65x SCI 闪存内核及其使用方法。

5.3.1 CPU1 内核

F28P65x CPU1 内核工程利用 CPU1 中的 SCI ROM 引导加载程序来下载 CPU1 应用程序映像和 CPU2 应用程序映像。CPU1 内核和应用程序文件作为参数输入到 serial_flash_programmer 实用程序，一旦主机通过 CPU1 SCI ROM 引导加载程序完成了自动波特率锁定，就会下载 CPU1 内核。CPU1 内核下载完毕后，即可开始正常的内核操作，例如 DFU、擦除等。

CPU1 操作完成后，可以通过选择“DFU CPU2”命令下载 CPU2 应用程序。CPU1 内核将等待 CPU2 应用程序下载完成，然后再根据是否选择了“Run CPU1”或“Reset CPU1”进行 CPU1 应用程序映像寻址或复位。对于

F28P65x，CPU2 没有 SCI ROM 引导加载程序，因此必须通过 CPU1 内核下载 CPU2 应用程序映像。CPU1 应用程序映像必须为 CPU2 设置 GSxRAM、闪存组和器件引导模式。CPU2 映像下载完毕后，即可继续正常的内核操作。

5.3.1.1 主机-内核通信：ControlCard

F28P65x 器件的主机-内核通信需要更改 SCI 的引导定义设置。默认情况下，该器件为 SCI 通信分配 GPIO12/13 引脚。用于 SCITX/SCIRX 的默认引导值与板上 FTDI 芯片使用的值不同，因此用户需要更改引导定义值，以便与 USB 转 UART SCI 线路所看到的值保持一致。USB 转 UART SCI 线路通过 ControlCard 的 GPIO 28/29 与 SCI 共享通信。因此，需要设置仿真引导模式才能使用 GPIO28/29 进行 SCI 通信。

5.3.1.2 主机-内核通信：LaunchPad™ 开发套件

F28P65x 器件的主机-内核通信需要更改 SCI 的引导定义设置。默认情况下，该器件为 SCI 通信分配 GPIO12/13 引脚。用于 SCITX/SCIRX 的默认引导值与板上 FTDI 芯片使用的值不同，因此用户需要更改引导定义值，以便与 USB 转 UART SCI 线路所看到的值保持一致。USB 转 UART SCI 线路通过 LaunchPad 的 GPIO 42/43 与 SCI 共享通信。因此，需要设置仿真引导模式才能使用 GPIO42/43 进行 SCI 通信。

5.3.1.3 内核命令

大多数命令与以前的器件相同。CPU1 的命令中有一些新功能更改：擦除 CPU1，擦除 CPU2。

表 5-12 介绍了每条命令的工作原理。

表 5-12. 内核命令

内核命令	命令代码	说明
Erase CPU1	0x0300	1.接收包含 32 位数据的数据包 2.擦除数据中指定的存储体 3.发送状态数据包
Erase CPU2	0x0400	1.接收包含 32 位数据的数据包 2.擦除数据中指定的存储体 3.发送状态数据包

5.3.2 使用 SCI 引导加载程序下载工程

本节详细介绍了在 controlCard 上运行 F28P65x 内核所需的步骤。

5.3.2.1 CPU1

1. 将 controlCard S1 位置 1 设为 OFF（左侧）。这会关闭 JTAG 通信。
2. 将 controlCard S1 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 将 S3 位置 1 设为 OFF（左侧），将位置 2 设为 ON（右侧）。这会将 controlCard 设置为 SCI 引导模式。
4. 打开命令窗口并导航至 serial_flash_programmer.exe 所在的位置。
5. 输入带有参数的命令，如下所述：
   - 示例：serial_flash_programmer.exe -d f28p65x -k flash_kernel_c28x_dual_ex1_c28x1.txt -a led_ex1_c28x_dual_blinky_cpu1.txt -n led_ex1_c28x_dual_blinky_cpu2.txt -b 9600 -p COM34 -v。
6. 当内核下载完毕并弹出菜单时，执行所有 CPU1 操作，然后再继续执行 CPU2。重新启动 serial_flash_programmer 应用程序会将控制权交还给 CPU1，因此 CPU1 和 CPU2 操作可在运行 serial_flash_programmer 应用程序的同时执行。

5.3.3 使用 CCS 编译工程

5.3.3.1 CPU1

1. 将 controlCard S1 位置 1 设为 ON（右侧）。这会开启 JTAG 通信。
2. 将 controlCard S1 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 在 CCS 中，导入并编译 CPU1 和 CPU2 内核工程。
4. 启动目标配置文件。

5. 连接至 CPU1。
6. 将工程文件夹中提供的 gel 文件加载到工程中。右键点击目标配置中的 CPU1，然后选择“Open GEL Files View”。
7. 在“GEL Files”选项卡中，点击“GEL Files”。在“Script”窗口中右键点击，然后选择“Load GEL...”。导航至工程文件夹并加载 gel 文件。
8. 连接至 CPU2。
9. 点击 CPU1 对应的“Reset”。CPU2 现处于等待引导中。
10. 点击 CPU2 对应的“Resume”。
11. 点击 CPU1 对应的“Load Program”。
12. 加载 CPU1 内核的 .out 文件。文件加载完毕后，点击“Resume”。
13. 打开命令窗口并导航至 serial_flash_programmer_appln.exe 所在的位置。
14. 输入带有参数的命令，如下所述：

    示例：serial_flash_programmer_appln.exe -d f28p65x -k flash_kernel_c28x_dual_ex1_c28x1.txt -a led_ex1_c28x_dual_blinky_cpu1.txt -n led_ex1_c28x_dual_blinky_cpu2.txt -b 9600 -p COM34 -v
15. 当弹出菜单时，执行所有 CPU1 操作，然后再继续执行 CPU2。

5.4 F28P55x SCI 闪存内核

本节概述了 F28P55x SCI 闪存内核并提供了使用说明。

5.4.1 实施

SCI 闪存内核 B 的 F28P55x 实现与上一个实现部分中描述的实现类似，但 DFU 命令期间闪存擦除的方式不同。

对于 F28P55x 上的器件固件升级 (DFU) 命令，将执行以下步骤：

1. 主机编程器会检查是否包含一个指定要擦除的闪存组和扇区的文本文件（请参阅指定应用的闪存组和扇区）
2. 如果包含，将擦除指定的闪存组和扇区。否则，会擦除所有闪存组和扇区，以便为编程做好准备
3. 内核从 SCI 模块逐字节接收十六进制引导格式的文件，并对块大小计算校验和。收到数据块后，内核会发回校验和。
4. 接收到一个数据块并将数据存储在缓冲器中后，内核会使用闪存 API 库将数据与 ECC 一同编程到闪存中。请记住，该内核使用闪存 API 的 512 位编程功能，因此应用必须与 512 位边界对齐。请参阅故障排除章节了解更多详情。

   备注

   若要获取有关数据块以及如何确定块大小的更多信息，请参阅器件特定 TRM 的 *引导加载程序数据流结构* 部分。请注意，这里提到的块和块大小不要与前文提及的命令包相混淆。
5. 之后，内核会验证数据和 ECC 是否正确编程到闪存中。

类似地，验证命令会接收一个十六进制引导格式的文件，内核只验证闪存的内容，而不是擦除闪存并对闪存进行编程。

5.4.1.1 指定应用的闪存组和扇区

要指定在 DFU 命令期间擦除哪些闪存组和扇区，在执行主机编程器时必须使用 [-t] 选项。在 [-t] 选项后输入格式如下的文本文件：

```
00 FFFFFFFF FFFFFFFF 01 FFFFFFFF FFFFFFFF 02 FFFFFFFF FFFFFFFF 03 FFFFFFFF FFFFFFFF 04 FFFFFFFF
00000000 05 0000000
```

前两个字符 00 指示随后的 64 位适用于闪存组 0。这被称为闪存组标识符。闪存组标识符 0xFFFFFFFF 之后的第一个 32 位指示扇区 0-31 被应用程序使用，应擦除。随后的 32 位 0xFFFFFFFF 表示扇区 32-127 被应用程序使用，应擦除。在第二个 32 位值中，每个位对应八个闪存扇区。例如，LSB 或位 0 对应于闪存扇区 32-39，而位 1 对应于扇区 40-47。在闪存组标识符后的 64 位中，1 标记要擦除的相应扇区，0 不标记要擦除的扇区。例如，如果只需要擦除闪存组 0 中的闪存扇区 0-31，则第二个 32 位值应更改为 0x00000000。

闪存组 1-4 仍采用这种格式。闪存组 4 标识符之后的第二个 32 位值需要始终保持为 0x00000000。与闪存组 0-3 不同，闪存组 4 具有 32 个扇区。此外，未使用最终标识符和 32 位值，即 05 和 0x00000000。

5.4.2 内核

F28P55x 内核工程利用 CPU1 中的 SCI ROM 引导加载程序来下载应用程序映像。CPU1 内核和应用程序文件作为参数输入到 serial_flash_programmer 实用程序，一旦主机通过 CPU1 SCI ROM 引导加载程序完成了自动波特率锁定，就会下载 CPU1 内核。CPU1 内核下载完毕后，即可开始正常的内核操作，例如 DFU、擦除等。节 5.1.2 展示了 CPU1 的内核命令。

5.4.3 使用 SCI 引导加载程序下载工程

1. 将 controlCard S4 位置 1 设为 OFF（左侧）。这会关闭 JTAG 通信。
2. 将 controlCard S4 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 将 S1 位置 1 设为 OFF（左侧），将位置 2 设为 ON（右侧）。这会将 controlCard 设置为 SCI 引导模式。
4. 打开命令窗口并导航至 serial_flash_programmer.exe 所在的位置。
5. 输入带有参数的命令，如下所述：
   - 示例：serial_flash_programmer.exe -d f28p55x -k flash_kernel_c28x_dual_ex1_c28x1.txt -a led_ex1_c28x_dual_blinky_cpu1.txt -n led_ex1_c28x_dual_blinky_cpu2.txt -b 9600 -p COM34 -v -t app_flash_banks_and_sectors.txt
6. 当内核下载完毕并弹出菜单时，执行所有所需的操作。

5.4.4 使用 CCS 编译工程

1. 将 controlCard S4 位置 1 设为 ON（右侧）。这会开启 JTAG 通信。
2. 将 controlCard S4 位置 2 设为 ON（右侧）。这会开启 UART 通信。
3. 在 CCS 中，导入并编译 CPU1 内核工程。
4. 启动目标配置文件。
5. 连接至 CPU1。
6. 点击“Load Program”。
7. 加载 CPU1 内核的 .out 文件。文件加载完毕后，点击“Resume”。
8. 打开命令窗口并导航至 serial_flash_programmer_appln.exe 所在的位置。
9. 输入带有参数的命令，如下所述：

   示例：serial_flash_programmer_appln.exe -d f28p55x -k flash_kernel_ex3_sci_flash_kernel.txt -a led_ex1_blinky.txt -b 9600 -p COM34 -v -t app_flash_banks_and_sectors.txt
10. 弹出菜单时，执行所有所需的 CPU1 操作。

6 实现示例

上述内核可在示例目录中特定器件的示例文件夹下的 C2000Ware 中找到。例如，F2837x 的闪存内核位于 *C2000Ware_x_x_xx_xx* > *device_support* > *f2837xd* > *examples* > *dual* > *F2837xD_sci_flash_kernels* > *cpu01*。主机应用程序位于 C2000Ware (*C2000Ware_x_xx_xx_xx* > *utilities* > *flash_programmers* > *serial_flash_programmer*)。源代码和可执行文件可在 *serial_flash_programmer* 文件夹中找到。本节详细介绍了 serial_flash_programmer：如何编译、运行并将其与闪存内核 A 和 B 一同使用。

备注

必须将相应器件的闪存内核提供给用于对闪存进行编程的主机应用程序工具。serial_flash_programmer 以相同的方式启动，与内核或器件无关。它首先通过 SCI ROM 引导加载程序将内核加载到器件。在此之后，该工具的功能会因所使用的器件和内核而异。

6.1 器件设置

6.1.1 闪存内核

Code Composer Studio (CCS) 的闪存内核源文件和工程文件在 C2000Ware 中提供，位于相应器件的示例目录中。将工程加载到 CCS 中进行编译。这些工程有一个编译后处理步骤，将编译和链接的 .out 文件转换为 SCI ROM 引导加载程序所需的正确的十六进制格式引导文件，并按扩展名为 .txt 的示例名称进行保存。

6.1.2 硬件

在 CCS 中编译内核后，务必正确设置器件，使其能与运行 serial_flash_programmer 的主机 PC 进行通信。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 SCI 引导模式。接下来，将相应的 SCI 引导加载程序 GPIO 引脚连接到与主机 PC COM 端口相连的 Rx 和 Tx 引脚。通常需要收发器来将虚拟 COM 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 controlCARD）上，使用 FTDI 芯片将用于 SCI 通信的 GPIO 引脚连接到 USB 虚拟 COM 端口。请参阅 controlCARD 的用户指南，获取有关启用 SCI 通信所需的开关配置的信息。在这种情况下，PC 必须连接到 controlCARD 上的 mini-USB，并使用 FTDI 芯片的通道 B 连接到器件上的 GPIO 引脚。正确设置硬件以与主机进行通信后，重置器件。该操作会将器件引导至 SCI 引导模式。

6.2 主机应用程序：serial_flash_programmer

6.2.1 概述

命令行 PC 实用程序是一种编程解决方案，可以轻松集成到脚本环境中，用于生产线编程等应用程序。它是使用 Microsoft Visual Studio® 用 C++ 编写的。工程及其源代码可在 C2000Ware (*C2000Ware_x_xx_xx_xx* > *utilities* > *flash_programmers* > *serial_flash_programmer*) 中找到。

若要使用此工具对 C2000 器件进行编程，请确保目标板已复位且当前处于 SCI 引导模式并连接至 PC COM 端口。该工具的命令行使用说明如下：

```
serial_flash_programmer.exe -d <device> -k <kernel file> -a <app file> -p COM <num>
[-m] <kernel2 name> [-n] <app2 name> [-o] <kernel3 name> [-r] <app3 name> [-b] <baudrate> [-q] [-w]
[-v] [-t]
```

-d <device>	要连接和加载到的器件名称： F2802x、F2803x、F2805x、F2806x、F2807x、F2833x、F2837xD、F2837xS、F28004x、F2838x、F28002x、F28003x、F280013x、F280015x、F28P65x 或 F28P55x
-k <file>	CPU1 闪存内核的文件名。 该文件必须采用 ASCII SCI 引导格式。
-a <file>	要下载 CPU1 或向其验证的应用程序文件名。该文件必须采用 ASCII SCI 引导格式。
-m <file>	CPU2 闪存内核的文件名。 该文件必须采用 ASCII SCI 引导格式。
-n <file>	要下载 CPU2 或向其验证的应用程序文件名。 该文件必须采用 ASCII SCI 引导格式。
-o <file>	CM 闪存内核的文件名。该文件必须采用 ASCII GPIO 引导格式。
-r <file>	要下载 CM 或向其验证的应用程序文件名。该文件必须采用 ASCII GPIO 引导格式。
-p COM<num>	设置要用于通信的 COM 端口
-b <num>	为 COM 端口设置波特率
-? 或 -h	显示帮助
-q	安静模式。禁止输出至 stdout
-w	退出前等待按键
-v	启用详细输出
-t	该选项仅适用于 F28P55x 器件。自定义闪存组和扇区擦除配置的文件名。按照*指定应用的闪存组和扇区* 中详述的格式创建该文件。

-d、-k、-a、-p 参数都是必需的。如果省略波特率，则默认设置为 9600。

备注

闪存内核和闪存应用程序都必须采用 SCI8 引导格式。对于 F2838x CM，闪存内核和闪存应用程序文件必须采用 GPIO8 引导格式。这已在前面的节 3 中讨论过，并且可以使用 hex2000 实用程序从 OUT 文件生成。

6.2.2 使用 Visual Studio 编译和运行 serial_flash_programmer

可以使用 Visual Studio 编译 Serial_flash_programmer.cpp。

1. 导航至 serial_flash_programmerdirectory。
2. 双击 serial_flash_programmer.sln 以打开 Visual Studio 工程。
3. 将 Visual Studio 打开后，依次选择“Build”→“BuildSolution”。
4. Visual Studio 编译完毕后，依次选择“Debug”→“serial_flash_programmerproperties”。
5. 依次选择“Configuration Properties”→“Debugging”。
6. 选中 CommandArguments 旁的输入框。
7. 按以下格式输入参数。节 6.2.1 描述了这些参数。

- 格式：

```
-d <device> -k <file> -a <file> -p COM<num> -b <baudrate>
```

- 示例：

```
-d f2807x -k C:\Documents\flash_kernel.txt -a C:\Documents\Test.txt -p COM7 -b 9600
```

8. 依次点击“Apply”和“OK”。
9. 依次选择“Debug”→“Start Debugging”以开始运行工程。

6.2.3 为 F2806x 运行 serial_flash_programmer（闪存内核 A）

备注

建议在运行 serial_flash_programmer 之前复位器件，以便自动波特率正确完成。

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

```
:> serial_flash_programmer.exe -d f2806x -k <~\f28069_flash_kernel.txt> -a <file> -p COM<num>
```

这首先使用引导加载程序将 f28069_flash_kernel 加载到器件的 RAM 中。然后，内核会执行并加载，而后使用由‘-a’命令行参数指定的文件对闪存进行编程。

6.2.4 为 F2837xD 运行 serial_flash_programmer（闪存内核 B）

备注

建议在运行 serial_flash_programmer 之前复位器件，以便自动波特率正确完成。

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

```
:> serial_flash_programmer.exe -d f2837xD -k <~\F2837xD_sci_flash_kernels_cpu01.txt>
-a <file> -m <~\F2837xD_sci_flash_kernels_cpu02.txt> -n <file> -p COM<num> -v
```

这将自动连接到器件，执行自动波特率锁定，并将 CPU1 内核下载到 RAM 中并执行它。现在，CPU1 内核正在运行并等待来自主机的数据包。

3. serial_flash_programmer 将选项输出到屏幕上以供选择，而这些选项将发送到器件内核（请参阅图 6-1）。选择适当的数字，然后在需要输入该命令时提供任何必要的信息（在节 5.1 中所述）。对于 DFU 或 Verify，原始命令已经指定了应用程序文件，因此此时不需要其他信息。图 6-2 显示了在执行命令 1-DFU CPU1 后窗口的外观。

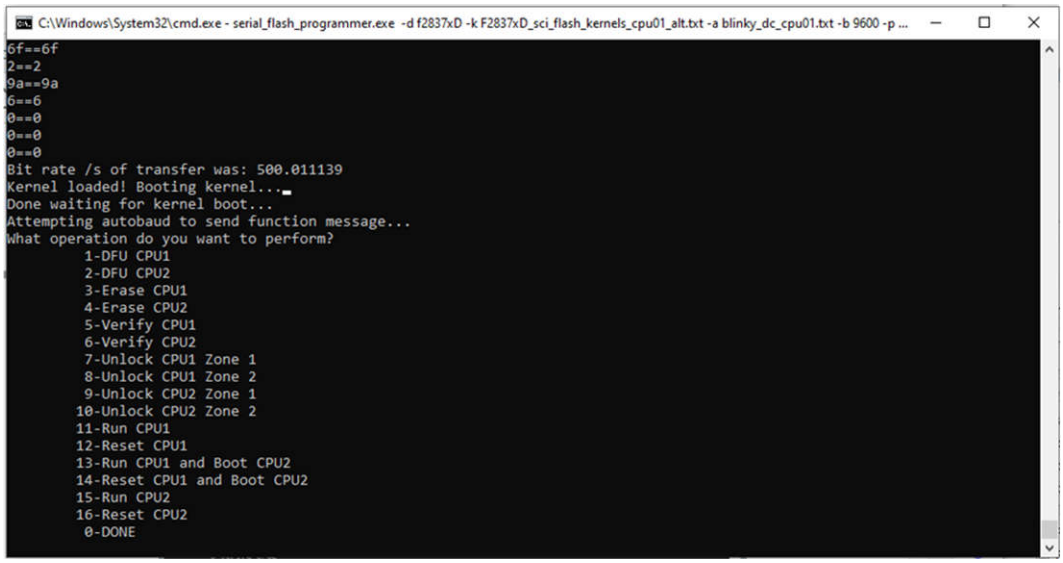

图 6-1. 串行闪存编程器将闪存内核下载到 RAM 后提示输入下一个命令

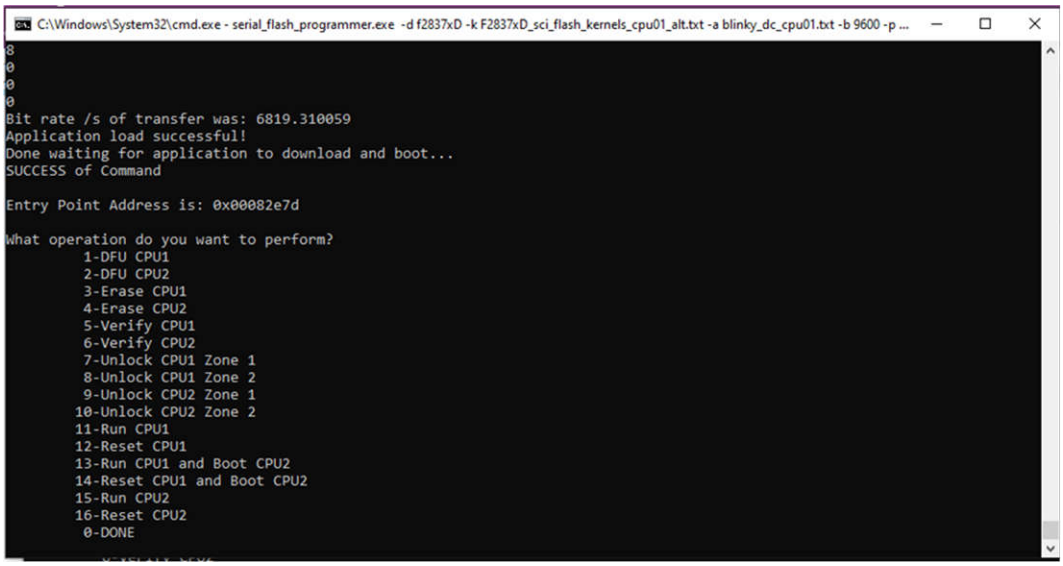

图 6-2. 将应用程序下载到闪存后的串行闪存编程器

6.3 主机应用程序：具有 SCI 闪存内核的 F28004x 上的固件更新

6.3.1 概述

如前所述，SCI 闪存内核支持固件更新。本节将介绍一种使用 SCI 引导和闪存引导模式执行此操作的可能方法。对于此场景，器件默认从闪存启动，并使用 SCI 引导程序进行固件更新。

值得注意的一件事是从闪存引导到底需要什么。如前所述，器件使用引导加载程序从外部源加载和运行代码。将 SCI 闪存内核加载到 RAM，然后将应用程序加载到闪存。如果从闪存引导，器件已经通过 Uniflash、CCS 或自定义引导加载程序配备了应用程序。

在此场景中，一旦器件从闪存启动，应用程序将会通过闪存入口点运行。对于固件更新，器件可以在 SCI 引导模式下启动，串行闪存编程器将 SCI 闪存内核下载到 RAM，这允许将新应用程序下载到闪存 - 因此固件升级无需将 SCI 闪存内核放入闪存中，从而节省闪存空间。

6.3.2 引导引脚配置

若要同时拥有闪存引导和 SCI 引导这两种引导模式，需要配置引导模式。引导控制 GPIO 引脚决定引导模式配置 - 对于 F28004x 器件，引导控制引脚是 GPIO 24 和 GPIO 32。为了从闪存引导，GPIO 24 和 GPIO 32 都必须上拉至 1，而对于 SCI 引导，GPIO 24 需要下拉至 0。

用户如何下拉 GPIO 引脚取决于实现方式 - 一个可能的例子是器件本身会在固件升级之前将 GPIO 引脚设为低电平 - 更多详细信息可在 [10] 中找到。

6.3.3 使用三种引导模式

假设用户想要在 3 种引导模式之间切换 - SCI、闪存和 I2C。需要使用三个引导模式引脚，因为它们并未包含在 4 种器件默认引导模式中。

为了在此处根据需要进一步自定义引导模式，用户需要使用用户可配置 DCSM OTP 的 BOOTPIN_CONFIG 位置。这允许用户选择最多 3 个 GPIO 引脚以拥有多达 8 种引导模式。

用户可配置 DCSM OTP 中的 BOOTDEF 寄存器还允许用户配置默认引导模式之外的引导模式。

若要在 SCI、闪存和 I2C 引导之间切换，需要执行以下步骤：

1. 适当地设置 BOOTPIN_CONFIG – 位 7-0、15-8、23-16 全部用于指示将使用哪些 GPIO 来表示引导模式。位 31-24 将具有密钥 0x5A，用以验证较早的位。
2. 将 BOOTDEF1 设置为 0x01 以使用引导模式 1 指示 SCI 引导（当引导模式选择 GPIO = 001 时），而 SCI GPIO 为 GPIO29 和 GPIO28。
3. 将 BOOTDEF3 设置为 0x03 以使用引导模式 3 指示闪存引导（引导模式选择 GPIO = 011 时）。闪存入口点是来自闪存组 0、扇区 0 的 0x80000。
4. 将 BOOTDEF7 设置为 0x07 以使用引导模式 7 指示 I2C 引导（当引导模式选择 GPIO = 111 时），而 I2C GPIO 为 GPIO32 和 GPIO33。

若要对引导模式选项进行试验，用户可在写入 OTP_BOOTPIN_CONFIG 之前使用 EMU_BOOTPIN_CONFIG 寄存器。OTP_BOOTDEF 还有一个用于试验的 EMU_BOOTDEF。

F2837x 上的 BOOTPIN_CONFIG 寄存器等效于 BOOTCTRL 寄存器。

6.3.4 执行实时固件更新

我们已经看到了如何将闪存内核下载到 RAM 中，并为将应用程序下载到闪存提供便利。通过将闪存内核存储在闪存中，用户无需将闪存内核下载到 RAM 中，从而能节省时间。可在闪存中直接将应用程序下载到闪存。另一方面，它会占用一些闪存空间。这是某些用户在决定使用时要权衡的方面。启用某些特性特别有用，例如实时固件更新 (LFU)。在 LFU 中，应用程序仅使用闪存引导模式，并且在应用程序运行时，用户可选择更新固件。应用程序会将控制权传递给闪存中的闪存内核，这将有助于更新闪存中的应用程序。在单组闪存上，这通常需要进行一次器件复位。但是，如果器件包含多个闪存组，则可以使用 LFU 避免器件重置，从而能够无缝过渡到新固件。

有关如何执行实时固件更新的更多详情，请参阅 [6] 和 [7]。

7 疑难解答

以下是针对用户在使用 SCI 闪存内核时遇到的一些常见问题的解决方案。

7.1 常见问题

问题：我找不到 SCI 闪存内核工程，它们在哪里？

回答：

器件	编译配置	位置
F2802x	RAM	C2000Ware_x_xx_xx_xx > device_support > f2802x > examples > structs > f28027_flash_kernel
F2803x	RAM	C2000Ware_x_xx_xx_xx > device_support > f2803x > examples > c28 > f2803x_flash_kernel
F2805x	RAM	C2000Ware_x_xx_xx_xx > device_support > f2805x > examples > c28 > f28055_flash_kernel
F2806x	RAM	C2000Ware_x_xx_xx_xx > device_support > f2806x > examples > c28 > f28069_sci_flash_kernel
F2807x	RAM	C2000Ware_x_xx_xx_xx > device_support> f2807x > examples > cpu1 > F2807x_sci_flash_kernel
F2833x	RAM	C2000Ware_x_xx_xx_xx > device_support > f2833x > examples > f28335_flash_kernel
F2837xS	RAM	C2000Ware_x_xx_xx_xx > device_support > f2837xs > examples > cpu1 > F2837xS_sci_flash_kernel > cpu01
F2837xD	RAM	C2000Ware_x_xx_xx_xx > device_support > f2837xd > examples > dual > F2837xD_sci_flash_kernels
F28004x	RAM，支持 LDFU 的闪存，不支持 LDFU 的闪存	C2000Ware_x_xx_xx_xx > driverlib > f28004x > examples > flash, select flashapi_ex2_sci_kernel
F2838x	RAM	CPU1-CPU2 C2000Ware_x_x_xx_xx > driverlib > f2838x>examples>c28x_dual>flash_kernel CPU1-CM C2000Ware_x_x_xx_xx > driverlib > f2838x>examples>c28x_cm>flash_kernel
F28002x	RAM，支持 LDFU 的闪存	C2000Ware_x_xx_xx_xx > driverlib > f28002x > examples > flash，选择 flash_kernel_ex3_sci_flash_kernel
F28003x	RAM，支持 LDFU 的闪存	C2000Ware_x_xx_xx_xx > driverlib > f28003x > examples > flash，选择 flash_kernel_ex3_sci_flash_kernel
F280013x	RAM	C2000Ware_x_xx_xx_xx > driverlib > f280013x > examples > flash，选择 flash_kernel_ex3_sci_flash_kernel
F280015x	RAM	C2000Ware_x_xx_xx_xx > driverlib > f280015x > examples > flash，选择 flash_kernel_ex3_sci_flash_kernel
F28P65x	RAM	C2000Ware_x_xx_xx_xx > driverlib > f28p65x > examples > c28x_dual > flash_kernel
F28P55x	RAM	C2000Ware_x_xx_xx_xx > driverlib > f28p55x > examples > flash，选择 f28p55x_flash_ex3_sci_flash_kernel

问题：闪存内核 A 和闪存内核 B 之间有何区别？

回答：A 类闪存内核在任何数据流入之前擦除整个闪存，而 B 类闪存内核仅在数据流入时擦除所需扇区。B 类内核还为用户提供以独立操作方式按需擦除所有或选定扇区的选项。此外，与仅支持应用程序加载命令的 A 类内核相反，B 类内核支持多种用户命令。

问题：如果 SCI 闪存内核没有下载，我首先应该检查什么？

回答：

- 要检查的程序的一个区域是链接器命令文件 - 确保所有闪存段都与 128 位边界对齐。在 SECTIONS 中，在将段分配给闪存的每一行之后添加一个逗号和“ALIGN(8)”。如果使用 F28P55x SCI 闪存内核，则将闪存段与 512 位边界对齐。将其修改为“ALIGN(32)”，而不是如上所述的“ALIGN(8)”。
- 用户遇到的另一个常见问题是未对 SCI 引导模式使用正确的引导引脚。例如，在 F28004x 器件上，SCI 引导模式提供 4 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，请确保使用另一个 SCI 引导选项，以便它能够连接到另一组引脚。确保 SCI 内核工程也使用此 SCI 引导 GPIO 选项作为 SCI_GetFunction() 的参数。
- 使用长电缆时，请使用较低的波特率来消除噪音。
- 对于波特率和连接问题，请尝试为器件运行 SCI 环回和回显示例（您可以在 C2000Ware 的 device_support 或器件的 driverlib 文件夹中找到这些示例）。

7.2 SCI 引导

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

回答：确保将正确的 GPIO 引脚用于 SCI 引导模式。相关详细说明，请参阅上一个问题。

7.3 F2837x

7.3.1 F2837xS

问题：SCI 内核下载到 RAM，应用程序下载到闪存，但没有出现命令窗口，完成后，闪存似乎没有写入？

回答：若要解决在将闪存内核下载到 RAM 后没有出现命令窗口的问题 - SCI 闪存内核工程包含 2 个可能的 SCIA 引导选项：

- SCIA 引导选项 2（GPIO 28、29）
- SCIA 引导选项 1（GPIO 84、85）

根据您的硬件配置使用正确的引脚。若要解决闪存未正确写入的问题 - 将所有闪存部分与链接器命令文件中的 128 位边界对齐。

7.3.2 F2837xD

问题：我想从闪存和 SCI 引导，如何在两者之间切换？闪存引导用于正常操作，SCI 引导用于固件更新。

回答：

这里提供一些选项。节 6.3 中提到了一些 - 器件可以在闪存引导模式下启动以正常运行，在 SCI 引导模式下启动以进行固件升级。用户需要设计一个方案，允许更改引导模式选择引脚设置。或者，用户可以决定只使用闪存引导模式，然后在软件中跳转到 SCI 引导功能。这可以通过 IO 触发器或主机命令来实现，作为调用 SCI_Boot() 函数的提示。这种方法的优点是用户不需要更改引导模式选择引脚设置。请注意，在这种情况下，不会调用 SCI ROM 引导加载程序。

问题：使用 SCI 引导模式升级固件是可行的，但是使用闪存引导和通过调用 SCI_Boot() 在软件中跳转到 SCI 引导不可行。我应该如何解决此问题？

回答：

- 确保正确的 GPIO 引脚用于 SCI 引导模式。
- 此外，有必要在程序中的 SCI_Boot() 调用之前禁用看门狗和中断，因为闪存正在更新。

问题：使用 ControlCARD，我无法在 SCI 引导中将 SCI 闪存内核下载到 RAM，我应该采取什么步骤？

回答：

1. controlCARD 要求使用 SCI 引导的选项 2 才能在器件上工作。如需了解更多详情，请参阅 [12] 的*警告/注意/勘误表*部分。为此，请确保使用 SCI_BOOT_ALTERNATE 作为 SCI_GetFunction 的参数编译 SCI 闪存内核工程。
2. 接下来，将 controlCARD 上的 SW1:A 位置 1 设置为 ON。这会允许连接仿真器。通过将 SW1 位置 1 设置为 0，将 SW1 位置 2 设置为 1，将引导模式设置为 SCI。
3. 之后，确保 SW1:A 位置 2 也为 ON。GPIO28（和 180-引脚 controlCARD 连接器的引脚 76）将耦合到 FTDI 的 USB 转串行适配器。这允许通过 FTDI 芯片与计算机进行 UART 通信。
4. 之后，通过 CCS 连接仿真器。启动 controlCard 的目标配置文件并连接到 CPU1。打开 Memory Window（“View”>“Memory Window”）并查找存储器中的位置 0xD00。将其更改为 0x815A。为了将备用 GPIO（28、29）用于 SCI 引导模式，我们需要设置相应 BOOTCTRL 寄存器（EMUBOOTCTRL (0xD00) 或 Z1-BOOTCTRL（0x7801E，位于用户可配置的 DCSM OTP 中）的 BMODE 字段（位 15:8）设为 0x81）。
5. 如果您也希望在 CPU2 上运行闪存内核，请在 CCS 中连接。点击“Reset CPU”，然后点击“Resume”。该程序将进入 ESTOP。再次点击“Resume”。
6. 在 CCS 中，对于 CPU1，点击“Reset CPU”，然后点击“Resume”(F8)。
7. 现在，使用适当的命令运行命令 serial_flash_programmer.exe。内核将下载至 RAM。当菜单弹出时，选择所需的操作以继续。若要查看此阶段命令窗口的外观，请参阅图 6-1。

7.3.3 F2837xD LaunchPad™

问题：我无法将 SCI 闪存内核加载到 RAM 中，原因何在？

回答：F2837xD LaunchPad 不支持 SCI 引导模式，因此该板不能与 SCI 闪存内核一起使用（请参阅[11 中的]*修订*部分）。

7.4 F28P65x

问题: 使用 ControlCARD，我无法在 SCI 引导中将 SCI 闪存内核下载到 RAM，我应该采取什么步骤？

答案：

1. 检查 controlCard 的修订版本。只有修订版 A 能够使用 SCI 闪存内核工程，因为 SCIRX/SCITX 通信线路正确连接到 GPIO28/29 以实现 SCI 自动波特率。低于版本 A 的版本不能用于该工程。如需了解更多详情，请参阅 [13] 的*警告/注意/勘误表*部分。
2. 默认情况下，器件的 FTDI 芯片接口使用 GPIO12/13 进行 SCI 通信。controlCARD 要求使用 SCI 引导的选项 5 才能在器件上工作。
3. 接下来，将 controlCARD 上的 S1 位置 1 设置为 ON。这会允许连接仿真器。通过将 S3 位置 1 设置为 0，将 SW1 位置 2 设置为 1，将引导模式设置为 SCI。
4. 之后，确保 S1 位置 2 也为 ON。GPIO28（和 180-引脚 controlCARD 连接器的引脚 76）将耦合到 FTDI 的 USB 转串行适配器。这允许通过 FTDI 芯片与计算机进行 UART 通信。
5. 之后，通过 CCS 连接仿真器。启动 controlCard 的目标配置文件并连接到 CPU1。打开 Memory Window（“View”>“Memory Window”）并查找存储器中的位置 0xD00。为了将备用 GPIO（28、29）用于 SCI 引导模式，我们需要将 EMU-BOOTDEF-LOW 字段（位 15:0）或 Z1-OTP-BOOTDEF-LOW（0x7800C，位于用户可配置的 DCSM OTP 中）设置为 0xA1。在各自的寄存器位置更改以下值：0xFFFF（在 0xD00）、0x5AFF（在 0xD01）、0x00A1（在 0xD04）。
6. 如果您还想运行 CPU2，请在 CCS 中连接。点击“Reset CPU”，然后点击“Resume”。该程序将进入 ESTOP。再次点击“Resume”。
7. 在 CCS 中，对于 CPU1，点击“Reset CPU”，然后点击“Resume”(F8)。
8. 现在，使用适当的命令运行命令 serial_flash_programmer.exe。内核将下载至 RAM。当菜单弹出时，选择所需的操作以继续。

问题: 使用 LaunchPad，我无法在 SCI 引导中将 SCI 闪存内核下载到 RAM，我应该采取什么步骤？

答案：

1. 默认情况下，器件的 FTDI 芯片接口使用 GPIO12/13 进行 SCI 通信。LaunchPad 要求使用 SCI 引导的选项 3 才能在器件上工作。在 SCI 闪存内核工程的器件头文件中，有一个用于使用 LaunchPad 的符号 (*_LAUNCHXL_F28P65X*) 必须添加到工程属性下的预定义符号列表中。此外，在工程中，请确保主源文件内为 *sciGetFunction* 提供的引导模式值设置为 *SCI_BOOT_ALT3*。重建工程以进行更改。
2. 接下来，在 LaunchPad 上将 S2 位置 1 设置为 OFF。这允许通过 FTDI 芯片与计算机进行 UART 通信。通过将 S3 位置 1 设置为 0，将 S3 位置 2 设置为 1，将引导模式设置为 SCI。
3. 之后，通过 CCS 连接仿真器。启动 controlCard 的目标配置文件并连接到 CPU1。打开 Memory Window（“View”>“Memory Window”）并查找存储器中的位置 0xD00。为了将备用 GPIO（42、43）用于 SCI 引导模式，我们需要将 EMU-BOOTDEF-LOW 字段（位 15:0）或 Z1-OTP-BOOTDEF-LOW（0x7800C，位于用户可配置的 DCSM OTP 中）设置为 0x61。在各自的寄存器位置更改以下值：0xFFFF（在 0xD00）、0x5AFF（在 0xD01）、0x0061（在 0xD04）。
4. 如果您还想运行 CPU2，请在 CCS 中连接。点击“Reset CPU”，然后点击“Resume”。该程序将进入 ESTOP。再次点击“Resume”。在 CCS 中，对于 CPU1，依次点击“Reset CPU”和“Resume”(F8)。现在，使用适当的命令运行命令 serial_flash_programmer.exe。内核将下载至 RAM。当菜单弹出时，选择所需的操作以继续。

8 参考资料

1. 德州仪器 (TI)：*TMS32028004x Piccolo 微控制器技术参考手册*
2. 德州仪器 (TI)：*TMS320F2837xD 双核 Delfino 微控制器技术参考手册* 的 *ROM 代码和外设启动* 部分
3. Piccolo 闪存 API 用户指南 - 位于 controlSUITE 中的以下位置：(/controlSUITE/libs/utilities/flash_api/DEVICE/VERSION/doc)
4. 德州仪器 (TI)：*TMS320F28M35x 和 TMS320F28M36x 闪存 API 参考指南*
5. 德州仪器 (TI)：*TMS320C28x 汇编语言工具用户指南*
6. 德州仪器 (TI)：*C2000 MCU 在有器件复位时的实时固件更新*
7. 德州仪器 (TI)：*C2000 MCU 在无器件复位时的实时固件更新*
8. 德州仪器 (TI)：*C2000 微控制器的 USB 闪存编程*
9. 德州仪器 (TI)：*TMS320F28004x 引导特性和配置*
10. 德州仪器 (TI)：*C2000 软件控制的固件更新流程*
11. 德州仪器 (TI)：*LAUNCHXL-F28379D 概述用户指南*
12. 德州仪器 (TI)：*TMS320F28379D controlCARD 用户指南*
13. 德州仪器 (TI)：*TMS320F28P65 controlCARD 用户指南*

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 Revision G (November 2023) to Revision H (April 2024) **Page**

- 更新了节 5 以提及新增了对 F28P55x 器件的支持 5
- 更新了节 5.1 以包含有关 F28P55x SCI 闪存内核的注释 5
- 添加了 *F28P55x SCI 闪存内核* 一节。 16
- 添加了章节，详细说明 F28P55x SCI 闪存内核实现与现有闪存内核 B 实现有何不同。 16
- 更新了节 6.2.1 以展示新添加的选项和 F28P55x 器件 19
- 更新了节 7.1 以展示新添加的选项和 F28P55x 器件 23

Changes from Revision F (July 2023) to Revision G (November 2023) **Page**

- 更新了节 5 5
- 添加了节 5.3 14
- 添加了节 5.3.1 14
- 添加了节 5.3.2 15
- 添加了节 5.3.3 15
- 更新了 节 6.2 19
- 更新了节 7.1 以展示 F28P55x 闪存内核工程在 C2000Ware 中的位置。此外，还添加了问题与解答以解决使用 F28P55x SCI 闪存内核时遇到的任何常见问题 23
- 添加了节 7.4 25

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