*Application Note*

# *C2000 微控制器的 CAN 闪存编程*

*Charles Roberson, Vamsikrishna Gudivada, and Skyler Baumer*

**摘要**

在无法使用 JTAG 调试探针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。在这些情况下，工程师需要依靠利用控制器局域网 (CAN) 或控制器局域网灵活数据速率（CAN-FD，也称为模块化控制器局域网 (MCAN)）等外设的编程解决方案。通过在 Boot-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，因为它们只能方便地将应用代码加载到 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 器件的可能的实施方式，并提供了用于评估解决方案的 PC 实用程序。

## 内容



## 插图清单



## 表格清单

## 商标



## 1 引言

CAN 闪存内核有助于对具有 CAN 外设的任何 C2000 MCU 进行固件更新。使用引导 ROM 中的 CAN 引导加载程序将其复制到器件的 RAM 中。然后，内核将使用 CAN 外设从主机传输固件并将其编程到闪存存储器中。闪存内核包含两部分软件：一个 PC 主机 CAN 编程器和一个用于 C2000 MCU 的 Code Composer Studio™ (CCS) 工程。F28003x 器件用作这些工程的基础，可进行修改以用于其他器件。如果要在任何器件上使用该工程，则可能需要修改与器件工作频率相关的参数。

总之，对闪存进行编程需要两个步骤：

1. 使用 CAN 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。

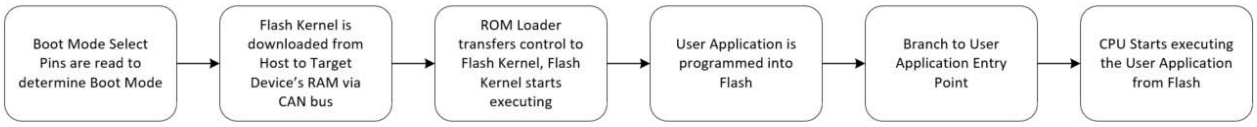


**图 1-1. CAN 引导流程**

## 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。

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。

**备注**

本报告中使用的术语 DCAN（DCAN 闪存内核、DCAN 闪存编程器等）指的是控制器局域网通信接口 (CAN) [8]。**本文档中描述的 DCAN 闪存编程器是指 CAN 模块。**

本报告中使用的术语 MCAN（MCAN 闪存内核、CAN 闪存编程器等）指的是模块化控制器局域网 (MCAN)。MCAN 可与控制器局域网灵活数据速率 (CAN-FD) [9] 互换使用的术语。**本文档中描述的 CAN 闪存编程器是指 MCAN 模块。**

在未来版本中，DCAN 和 MCAN 闪存工程的命名规则将保持一致，以方便使用。

## 3 ROM 引导加载程序和十六进制实用程序用法

器件启动之初，器件引导，并根据引导模式决定是执行已编程到闪存存储器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 (CCS) 时的引导执行路径。

**备注**

本节基于 TMS320F28003x 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* 一章中找到。

**表 3-1. F28003x 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（闪存或引导 ROM）。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。本应用手册所述的实现使用了 CAN 和 MCAN 加载程序，因此在上电时 GPIO 32 必须强制为低电平，GPIO 24 必须强制为高电平。如果器件引导时出现这种情况，ROM 中的 CAN 加载程序将开始执行并等待从主机接收数据。

对于 CAN 引导模式 (DCAN)，GPIO 分配的引导表由表 3-2 定义。表 3-3 展示了 MCAN 引导选项。请注意，DCAN 和 MCAN 的引导选项使用相同的 GPIO 分配来选择引导定义值。常见的 GPIO 对是 GPIO4 和 GPIO5，以及 GPIO13 和 GPIO12。如需更多信息，请查看特定于器件的 TRM 中的引导选项。

**表 3-2. CAN 引导选项**

| 选项 | BOOTDEF 值 | CANTXA GPIO | CANRXA GPIO |
|---|---|---|---|
| 0（默认值） | 0x02 | GPIO4 | GPIO5 |
| 1 | 0x22 | GPIO32 | GPIO33 |
| 2 | 0x42 | GPIO2 | GPIO3 |
| 3 | 0x62 | GPIO13 | GPIO12 |

**表 3-3. MCAN 引导选项**

| 选项 | BOOTDEFx 值 | CANTXA GPIO | CANRXA GPIO |
|---|---|---|---|
| 0 | 0x08 | GPIO4 | GPIO5 |
| 1 | 0x28 | GPIO1 | GPIO0 |
| 2 | 0x48 | GPIO13 | GPIO12 |

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，[6] 中的*引导加载程序数据流结构*一节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"
```

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>
```

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 DCAN 闪存内核和 MCAN 闪存内核小节介绍。

闪存内核期望固件映像采用相同的格式，因此上面的命令也可用于生成固件映像十六进制文件。

MCAN ROM 引导加载程序有三个引导选项，每个选项都映射到不同的 GPIO 引脚，用于 MCANRX 和 MCANTX 功能。为了将器件配置为 MCAN 引导模式，需要对器件的 OTP 进行编程以包括 MCAN 引导。有关如何对一次性

可编程 (OTP) 进行编程的更多详细信息，请参阅 *TMS320F28003x 实时微控制器技术参考手册* (TRM) 的 *引导 ROM* 一章。

# 4 DCAN 闪存内核

DCAN 闪存内核在以下器件上运行：

- TMS320F28003x
- TMS320F28P65x
- TMS320F280015x

若要查找这些器件的闪存内核工程的位置，请参阅疑难解答。

## 4.1 实施

DCAN 闪存内核基于 DCAN 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，F28P65x 和 F280015x（不包括 F28003x，此器件在稍后完成）DCAN 闪存内核会擦除器件的闪存以便为编程做好准备。F28P65x 和 F280015x DCAN 闪存内核工程允许用户指定在进行应用程序编程之前应擦除哪些闪存组和闪存扇区。自定义闪存组和扇区擦除中对此进行了更详细的论述。在闪存存储器中的相应位置进行擦除后，应用程序加载开始。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。

DCAN 模块在闪存内核中初始化后，该模块等待主机发送固件映像。闪存内核一次从主机接收 8 个字节，并将内容放入中间 RAM 缓冲区中。然后，将该缓冲区以 128 位或 512 位增量写入闪存。F28P65x 和 F280015x DCAN 闪存内核工程支持 512 位编程，而 F28003x 使用 128 位编程。如果需要，还有一个可用于 F28P65x 器件的 128 位编程工程。F280015x 闪存 API 支持 128 位编程，但闪存内核是用 512 位编程实现的。

在第一次写入扇区之前，F28003x（不包括 F28P65x 和 F280015x，因为闪存在此过程中的较早时间已被擦除）闪存内核会检查该扇区是否已被擦除，如果没有被擦除，F28003x 闪存内核会让闪存 API 执行擦除操作。此后，一个缓冲区将被写入闪存的内容填满，并从闪存 API 发送编程命令。一旦发生写入，闪存内核就会通过闪存 API 验证相应段是否已写入闪存的正确地址。一旦内核将所有内容复制到闪存，工程就会跳转到映像的入口地址。

存储在闪存中的固件映像的所有段都应根据一次性编程的位数对齐。如果一次编程 128 位（F28003x 和 F28P65x），这些段应与 128 位边界对齐。在固件镜像的链接器命令文件中，所有初始化段都需要映射到闪存扇区，并且在每次映射之后，需要添加 ALIGN(8) 指令以确保 128 位对齐。如果一次编程 512 位（F280015x 和 F28P65x），这些段应与 512 位边界对齐。在固件镜像的链接器命令文件中，所有初始化段都需要映射到闪存扇区，并且在每次映射之后，需要添加 ALIGN(32) 指令以确保 512 位对齐。

用于传输应用程序数据的协议与 DCAN ROM 加载程序协议略有不同。使用原始 DCAN ROM 加载程序协议，数据以每帧两个字节、100Kbps 的速度从主机传输到目标器件。由于数据每帧传输两个字节到 DCAN ROM 加载程序，这会增加下载内核或映像的总时间。将每帧数据长度代码 (DLC) 修改为 8 字节并将位速率增加至 1Mbps，就可以让 PC 侧应用程序一次通过主机发送多个字节，从而大大减少通信延迟。应用程序负载一节详细介绍了将比特率提高到 1Mbps 所需的更改。

### 4.1.1 自定义闪存组和扇区擦除

F280015x 和 F28P65x（包括 128 位和 512 位编程工程）DCAN 闪存内核允许用户指定在将应用程序加载到闪存之前应擦除哪些闪存组和闪存扇区。本节将讨论如何为 F28P65x 器件实现擦除，但 F280015x 的实现方式非常相似。

在 flash_kernel_ex5_can_flash_kernel.c 文件中，有四个 32 位无符号整数数组用于控制内核擦除的闪存组和扇区。

```
uint32_t Application_Flash_Banks[5] = {0,1,2,3,4};
uint32_t WE_Protection_A_Masks[5] = {0,0,0,0,0};
uint32_t WE_Protection_B_Masks[5] = {0,0,0,0,0};
uint32_t WE_Protection_OTP_Masks[5] = {0,0,0,0,0};
```

在 Application_Flash_Banks 内，应输入闪存组编号。在 WE_Protection_A_Masks 内，应输入与 Application_Flash_Banks 中每个闪存组的扇区 0-31 相对应的写入/擦除保护掩码。如上所示，每个闪存组的闪存扇区 0-31 将被擦除。在 WE_Protection_B_Masks 内，应输入与 Application_Flash_Banks 中每个闪存组的扇区 32-127 相对应的写入/擦除保护掩码。如上所示，每个闪存组的闪存扇区 32-127 将被擦除。更多有关这些掩码的信息，请参阅特定于器件的闪存 API 指南。同样，WE_Protection_OTP_Masks 中应填充每个闪存组的 OTP 所需的掩码。请注意，配置该数组不会导致 OTP 被擦除，而是使器件能够在应用程序加载期间对闪存组的 OTP 进行编程。

此外，将应用程序编程到闪存时使用 WE_Protection_A_Masks 和 WE_Protection_B_Masks。根据正在编程的数据的目标地址确定正确的掩码。

根据需要配置这些数组后，重新编译闪存内核并生成新的内核映像。

### 4.1.2 应用程序加载

本节介绍使用 DCAN 引导模式将应用程序编程到闪存的整个流程。

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

1. 器件进入 DCAN 引导加载程序，等待接收邮箱 1 中的消息帧。对于引导加载程序通信，可接受的消息具有 0x1 的消息标识符 (MSGID) 值。有关邮箱和 MSGID 的更多信息，请参阅器件特定 TRM 的 DCAN 一章 [6]。
2. 闪存内核以每帧 2 字节的数据传输到器件。主机编程器将帧传输到器件，检查数据字节 3 和 4 是否为非零值。文本文件的字节 3 和 4 必须替换为根据位时序寄存器值 (*CAN_CALC_BTRREG)* 的最终结果计算出的十六进制值，顺序为最低有效字节后跟最高有效字节。如果主机编程器识别出字节 3 和 4 的位时序更改，则主机编程器会将位时序更改发送到器件并重新初始化自身（跳过以下 7 个保留字）。器件会将比特率增加到所需的位时序，并继续接收帧，直到内核完成下载。

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 和闪存等待状态等。
   a. 此时，F28P65x 和 F280015x 器件将擦除用户指定的闪存组和扇区。
   b. F28003x 内核在稍后进行擦除。
4. 内核进入 DCAN 引导模式并等待接收邮箱 1 中的消息帧。*CAN_CALC_BTRREG* 值 (*bootloader_can_timing.h)* 在工程内调整为 1Mbps，并且 DCAN 消息缓冲器大小由内核从每帧 2 字节调整为每帧 8 字节，从而加快应用程序下载。
5. 主机编程器将延迟 5 秒，然后以 1Mbps 和每帧 8 字节的有效负载发送应用程序映像。
6. 下载过程开始时会读取一个密钥、一些保留字段和应用程序入口点。
   a. 此时，F28003x 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. 一旦闪存被擦除，应用程序加载将继续进行，将数据帧传输到应用程序代码块中，并将其编程到闪存中（每次 128 位或 512 位）。
   a. F280015x 和 F28P65x 闪存内核一次对 512 位进行编程
   b. F28003x 闪存内核和 128 位版本的 F28P65x 闪存内核一次对 128 位进行编程
8. 将一个数据块编程到闪存后，内核继续接收消息以编程下一个数据块。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 MCAN 闪存内核

MCAN 闪存内核在以下器件上运行：

- TMS320F28003x
- TMS320F28P65x
- TMS320F28P55x

若要查找这些器件的闪存内核工程的位置，请参阅疑难解答。

## 5.1 实施

闪存内核工程是根据 MCAN ROM 引导加载程序建模的。它直接进入已修改为写入闪存的 MCAN_Boot 函数。闪存内核的 MCAN 模块初始化与引导加载程序相同 - MCAN 模块的时钟源、标称和数据比特率、GPIO 引脚等由内核在初始化时根据引导模式进行设置。在接收到任何应用程序数据之前，F28P55x 和 F28P65x（不包括 F28P65x，此器件在稍后完成）MCAN 闪存内核会擦除器件的闪存以便为编程做好准备。此外，F28P55x MCAN 闪存内核工程允许用户指定在进行应用程序编程之前应擦除哪些闪存组和闪存扇区。自定义闪存组和扇区擦除中对此进行了更详细的论述。在闪存存储器中的相应位置进行擦除后，应用程序加载开始。

闪存内核一次从主机接收 64 个字节，并将内容放入中间 RAM 缓冲区中。然后，将该缓冲区以 128 位或 512 位增量写入闪存。F28003x 和 F28P65x 闪存内核一次写入 128 位，而 F28P55x 闪存内核一次写入 512 位。在第一次写入扇区之前，F28003x 会检查该扇区是否已被擦除，如果没有被擦除，闪存内核会让闪存 API 执行擦除操作（如上所述，F28P55x 和 F28P65x 闪存内核会事先擦除闪存）。此后，一个缓冲区将被写入闪存的内容填满，并从闪存 API 发送编程命令。一旦发生写入，闪存内核就会通过闪存 API 验证相应段是否已写入闪存的正确地址。一旦内核将所有内容复制到闪存，工程就会跳转到映像的入口地址。

存储在闪存中的固件映像的所有段都应根据一次性编程的位数对齐。如果一次编程 128 位（F28003x 和 F28P65x），这些段应与 128 位边界对齐。在固件镜像的链接器命令文件中，所有初始化段都需要映射到闪存扇区，并且在每次映射之后，需要添加 ALIGN(8) 指令以确保 128 位对齐。如果一次编程 512 位 (F28P55x)，这些段应与 512 位边界对齐。在固件镜像的链接器命令文件中，所有初始化段都需要映射到闪存扇区，并且在每次映射之后，需要添加 ALIGN(32) 指令以确保 512 位对齐。

用于传输应用程序数据的协议遵循 MCAN ROM 加载程序协议。使用原始 MCAN ROM 加载程序协议，使用的标称比特率为 1Mbps，并且每帧从主机向目标器件传输 64 个字节，以实现标称位时序。该协议使用的数据比特率是 2Mbps，以实现数据位时序。

### 5.1.1 自定义闪存组和扇区擦除

F28P55x MCAN 闪存内核允许用户指定在将应用程序加载到闪存之前应擦除哪些闪存组和闪存扇区。在 flash_kernel_ex4_can_flash_kernel.c 文件中，有四个 32 位无符号整数数组用于控制内核擦除的闪存组和扇区。

```
uint32_t Flash_Banks_To_Erase[5] = {0,1,2,3,4};
uint32_t CMD_WE_Protection_A_Masks[5] = {0,0,0,0,0};
uint32_t CMD_WE_Protection_B_Masks[5] = {0,0,0,0,0};
uint32_t CMD_WE_Protection_UO_Masks[5] = {0,0,0,0,0};
```

在 Flash_Banks_To_Erase 内，应输入闪存组编号。在 CMD_WE_Protection_A_Masks 内，应输入与 Flash_Banks_To_Erase 中每个闪存组的扇区 0-31 相对应的写入/擦除保护掩码。如上所示，每个闪存组的闪存扇区 0-31 将被擦除。在 CMD_WE_Protection_B_Masks 内，应输入与 Flash_Banks_To_Erase 中每个闪存组的扇区 32-127 相对应的写入/擦除保护掩码。如上所示，每个闪存组的闪存扇区 32-127 将被擦除。更多有关这些掩码的信息，请参阅特定于器件的闪存 API 指南。同样，CMD_WE_Protection_UO_Masks 中应填充每个闪存组的 OTP 所需的掩码。请注意，配置该数组不会导致 OTP 被擦除，而是使器件能够在应用程序加载期间对闪存组的 OTP 进行编程。

此外，将应用程序编程到闪存时使用 CMD_WE_Protection_A_Masks 和 CMD_WE_Protection_B_Masks。根据正在编程的数据的目标地址确定正确的掩码。

根据需要配置这些数组后，重新编译闪存内核并生成新的内核映像。

### 5.1.2 应用程序加载

本节介绍使用 MCAN 引导模式将应用程序编程到闪存的整个流程。

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

1. 器件进入 MCAN 引导加载程序并等待接收 RX 消息缓冲器中的消息帧。对于引导加载程序通信，可接受的消息具有 0x1 的 MSGID 值。
2. 闪存内核以每帧 64 字节数据和 1Mbps（标称比特率）的速度传输到器件。主机编程器将向器件传输帧，直到内核下载完成。
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 和闪存等待状态等。
   a. 此时，F28P55x 内核将擦除用户指定的闪存组/扇区。
   b. F28P65x 和 F28003x 内核会在稍后擦除闪存。
4. 内核进入 MCAN 引导模式并等待接收 RX 消息缓冲器中的消息帧。比特率切换 (BRS) 值在工程内进行调整，允许比特率增加到 2Mbps（数据比特率），从而加快应用程序下载。
5. 主机编程器延迟 5 秒，然后以 2Mbps 和每帧 64 字节的有效负载发送应用程序映像。
6. 下载过程开始时会读取一个密钥、一些保留字段和应用程序入口点。
   a. 此时，F28P65x 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. 一旦闪存被擦除，应用程序加载将继续进行，将数据帧传输到应用程序代码块中，并将其编程到闪存中（每次 128 位或 512 位）。
   a. F28P55x 闪存内核一次对 512 位进行编程
   b. F28003x 和 F28P65x 闪存内核一次对 128 位进行编程
      i. 在对数据进行编程之前，F28003x 内核会检查每个闪存扇区以查看其是否已被擦除。如果之前没有擦除该扇区，则会擦除该扇区并对应用程序数据进行编程。
8. 将一个数据块编程到闪存后，内核继续接收消息以编程下一个数据块。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。

# 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，F28003x 的 DCAN 闪存内核位于 *C2000Ware_x_x_xx_xx* > *driverlib* > *f28003x* > *examples* > *flash*。相关主机应用可在 C2000Ware (*C2000Ware_x_xx_xx_xx* > *utilities* > *flash_programmers* > *dcan_flash_programmer*) 中找到。源代码和可执行文件可在 *dcan_flash_programmer* 文件夹中找到。本节详细介绍了 dcan_flash_programmer：如何编译、运行并将其与 DCAN 闪存内核一同使用。同样，还将介绍 MCAN 闪存内核如何与 can_flash_programmer 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。主机编程器以相同的方式启动，与内核或器件无关。它首先通过 CAN/MCAN 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 CAN 或 MCAN 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

运行示例所需的硬件组件是连接到 CAN 收发器的 C2000 器件和 PEAK PCAN-USB Pro FD 分析仪。对于 ControlCard，需要使用定制设计的 CAN 收发器板以及 HSEC 180 引脚 ControlCard 扩展坞。定制设计的收发器板通过四种连接方式连接到 ControlCard：接地、3.3V、CANTX 和 CANRX。

LaunchPad™ 器件包含一个板载 CAN 收发器。PEAK PCAN-USB Pro FD 分析仪通过接地、CAN-Lo 和 CAN-Hi 连接来连接到 LaunchPad。板载 CAN 路由开关需要设置为低电平，以便收发器使用 GPIO 进行通信。

在 CCS 中编译内核后，务必正确设置器件，使其能与运行主机编程器的主机 PC 进行通信。首先要做的是确保正确配置引导模式值，以将器件引导至 CAN/MCAN 引导模式。接下来，使用收发器将 MCAN TX 和 RX 引脚连接到主机侧的 CAN 端口。

## 6.2 主机应用：*dcan_flash_programmer*

### 6.2.1 概述

主机负责将 DCAN 内核映像和闪存 ( 固件 ) 映像发送到 MCU。PEAK PCAN-USB Pro FD CAN 总线分析仪用作主机。闪存编程器工程在 *Visual Studio 2019* 上编译并运行。主机编程器使用 PEAK 的 *PCAN_Basic* API。*PCAN_Basic* API 可用于在 CAN 分析仪上发送和接收 CAN 帧。

**备注**

PEAK PCAN-USB Pro FD CAN 总线分析仪向后兼容，可以接收经典 CAN 帧以及 CAN-FD 帧。

在 F28003x 器件上，MCAN 模块的时钟由引导 ROM 切换到外部时钟源。LaunchPad 和 ControlCard 中的外部时钟为 20MHz。引导 ROM 将标称比特率配置为 100Kbps。主机 CAN 编程器将 PEAK CAN 分析仪配置为具有相同的时钟和标称比特率值。

主机初始化分析仪以使用 CAN，以 2 字节增量发送内核，然后以 8 字节增量发送映像，每帧之间延迟 10ms，以便闪存 API 有时间将接收到的数据编程到闪存中。一旦固件映像被写入，主机 CAN 编程器就会退出。

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* > *dcan_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 CAN 引导模式并连接至 PC COM 端口。主机编程器将把内核和应用程序文件作为命令行的输入。还有安静或详细输出的选项，以及在关闭 CAN 闪存编程器应用程序之前等待退出的选项。该工具的命令行使用说明如下：

```
dcan_flash_programmer.exe -d <device> -k <kernel file> -a <app file> [-q] [-w] [-v]
```

| | |
|---|---|
| -d <device> | - 要连接和加载的器件的名称：F28003x |
| -k <file> | - CPU1 闪存内核的文件名。该文件必须采用 ASCII 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。 |
| -? 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 verbose 输出。 |

**备注**

闪存内核和闪存应用程序都必须采用 SCI8 引导格式。CAN/MCAN ROM 加载程序遵循与 SCI ROM 加载程序相同的 8 位通道引导格式，使用 *-sci8* 格式选项。

### 6.2.2 使用 Visual Studio 编译和运行 *dcan_flash_programmer*

*dcan_flash_programmer.cpp* 可以使用 Visual Studio 进行编译。

1. 导航至 *dcan_flash_programmer* 目录。
2. 双击 *dcan_flash_programmer.sln* 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 → *dcan_flash_programmer* → properties。
5. 依次选择“Configuration Properties”→“Debugging”。
6. 选中 Command Arguments 旁的输入框。
7. 按以下格式输入参数。第 6.2.1 节介绍了这些参数
   a. 格式：`-d <device> -k <file> -a <file`
   b. `-d f28003x -k C:\Documents\flash_kernel.txt -a C:Documents\test.txt`
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 F28003x 运行 *dcan_flash_programmer*

1. 导航到包含已编译的 *can_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 *can_flash_programmer.exe*：

```
dcan_flash_programmer.exe -d f28003x -k <flash_kernel.txt> -a <file>
```

这首先使用引导加载程序将 *flash_kernel* 加载到器件的 RAM 中。然后，内核会执行并加载，而后用“-a”命令行参数指定的文件对闪存编程，如图 6-1 和图 6-2 所示。

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


**图 6-1. 将闪存内核下载到 RAM 后的 DCAN 闪存编程器提示**


**图 6-2. 下载闪存应用后的 DCAN 闪存编程器**

### 6.2.4 使用 DCAN 引导加载程序下载工程

本节详细介绍了在 ControlCard 上运行 DCAN 闪存内核所需的步骤：

1. 输入带有参数的命令，如下所述：
   a. 示例：*dcan_flash_programmer.exe -d f28003x -k flash_kernel_ex5_dcan_flash_kernel.txt -a led_ex1_blinky.txt -v*
2. 下载内核后，它会将应用文件移动到闪存并确认应用加载已完成。

对于 LaunchPad，必须执行以下步骤：

1. 将 LaunchPad SW4 位置设置为 OFF（关闭）。这样做是为了将 CANTX 和 CANRX 信号路由到接头而不是收发器。
2. 打开命令窗口并导航至 *dcan_flash_programmer.exe* 所在的位置。
3. 输入带有参数的命令，如下所述：
   a. 示例：*dcan_flash_programmer.exe -d f28003x -k flash_kernel_ex5_dcan_flash_kernel.txt -a led_ex1_blinky.txt -v*
4. 下载内核后，它会将固件传输到闪存并确认应用加载已完成。

### 6.2.5 使用 CCS 编译工程

1. 在 CCS 中，导入并编译 CPU1 内核工程。
2. 启动目标配置文件。
3. 连接至 CPU1。
4. 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”。
5. 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。

6. 在仿真模式下，需要设置以下存储器位置才能启用 CAN 引导模式：
   a. 带 0xFFFF 的位置 0xD00
   b. 带 0x5AFF 的位置 0xD01
   c. 带 0x00XX 的位置 0xD04，其中 XX 是 CAN 引导的引导模式：0x02、0x22、0x42 或 0x62。0x82、0xA2、0xC2 和 0xE2 的 SENDTEST CAN 引导模式分别使用与前四种配置相同的引脚，并且它们还发送两个 CAN 帧。在评估模式下，使用其中一种 SENDTEST 模式可确保 CAN 模块在主机开始发送闪存内核之前不会超时。要了解有关 SENDTEST 模式的更多信息，请参阅 C2000Ware 中的 DCAN 引导源文件（*C2000Ware_x_xx_xx_xx* > *driverlib* > *f28003x* > *examples* > *flash* > *DCAN_Boot.c*）。

图 6-3 展示了这些存储器位置的实现示例。对这些位置进行编程后，复位器件并点击“Resume”。现在，F28003x 器件应在 ROM 的 CAN 引导模式下等待。

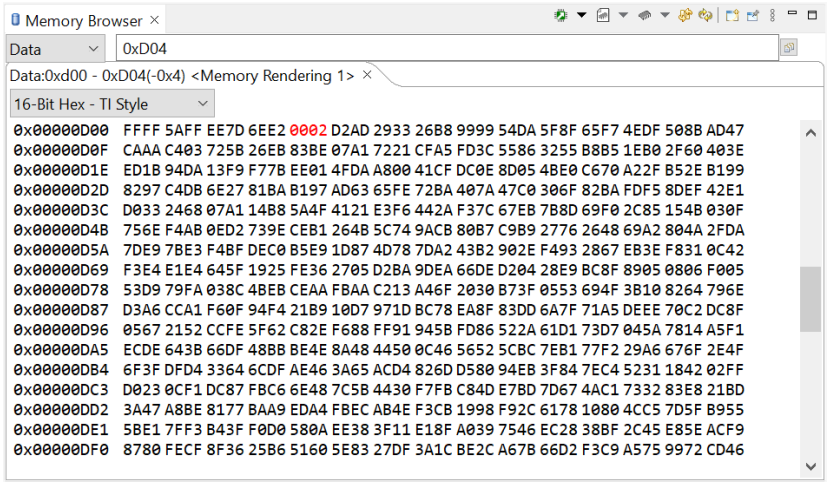


**图 6-3. CCS 中的存储器窗口示例（GPIO 引脚 4/5、CAN 引导模式）**

PEAK CAN 分析仪也需要连接至 PC。无需事先进行初始化。

## 6.3 主机应用：*can_flash_programmer* [MCAN]

### 6.3.1 概述

主机负责将 MCAN 内核映像和闪存（固件）映像发送到 MCU。PEAK PCAN-USB Pro FD CAN 总线分析仪用作主机。闪存编程器工程在 *Visual Studio 2019* 上编译并运行。主机编程器使用 PEAK 的 *PCAN_Basic* API。*PCAN_Basic* API 可用于在 CAN 分析仪上发送和接收 CAN-FD 帧。

在 F28003x 器件上，MCAN 模块的时钟由引导 ROM 切换到外部时钟源。LaunchPad 和 ControlCard 中的外部时钟为 20MHz。引导 ROM 将标称比特率配置为 1Mbps，将数据比特率配置为 2Mbps。主机 CAN 编程器将 PEAK CAN 分析仪配置为具有相同的时钟、标称和数据比特率值。

主机初始化分析仪以使用 CAN-FD，以 64 字节增量发送内核，然后以 64 字节增量发送映像，每帧之间延迟 100ms，以便闪存 API 有时间将接收到的数据编程到闪存中。一旦固件映像被写入，主机 CAN 编程器就会退出。

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* > *can_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 CAN 引导模式并连接至 PC COM 端口。主机编程器将把内核和应用程序文件作为命令行的输入。还有安静或详细输出的选项，以及在关闭 CAN 闪存编程器应用程序之前等待退出的选项。该工具的命令行使用说明如下：

```
can_flash_programmer.exe -d <device> -k <kernel file> -a <app file> [-q] [-w] [-v]
```

| | |
|---|---|
| -d <device> | - 要连接和加载的器件的名称：F28003x、F28P65x、F28P55x |

| | |
|---|---|
| -k <file> | - CPU1 闪存内核的文件名。该文件必须采用 ASCII 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。 |
| -? 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 verbose 输出。 |

**备注**

闪存内核和闪存应用程序都必须采用 SCI8 引导格式。CAN/MCAN ROM 加载程序遵循与 SCI ROM 加载程序相同的 8 位通道引导格式，使用 *-sci8* 格式选项。

### 6.3.2 使用 Visual Studio 编译和运行 *can_flash_programmer*

*can_flash_programmer.cpp* 可以使用 Visual Studio 进行编译。

1. 导航至 *can_flash_programmer* 目录。
2. 双击 *can_flash_programmer.sln* 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”→“*can_flash_programmer*”→“properties”。
5. 依次选择“Configuration Properties”→“Debugging”。
6. 选中“Command Arguments”旁的输入框。
7. 按以下格式输入参数。第 6.3.1 节介绍了这些参数。
   a. 格式：`-d <device> -k <file> -a <file`
   b. `-d f28003x -k C:\Documents\flash_kernel.txt -a C:Documents\test.txt`
8. 依次点击“Apply”和“OK”。
9. 依次选择“Debug”→“Start Debugging”以开始运行工程。

### 6.3.3 为 F28003x 运行 *can_flash_programmer*

1. 导航到包含已编译的 *can_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 *can_flash_programmer.exe*：

```
can_flash_programmer.exe -d f28003x -k <flash_kernel.txt> -a <file>
```

这首先使用引导加载程序将 *flash_kernel* 加载到器件的 RAM 中。然后，内核会执行并加载，而后用“-a”命令行参数指定的文件对闪存编程，如 和 所示。

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


图 6-4. 将闪存内核下载到 RAM 后的 CAN 闪存编程器提示


图 6-5. 下载闪存应用后的 CAN 闪存编程器

### 6.3.4 使用 MCAN 引导加载程序下载工程

本节详细介绍了在 ControlCard 上运行 MCAN 闪存内核所需的步骤：

1. 输入带有参数的命令，如下所述：
   a. 示例：*can_flash_programmer.exe -d f28003x -k flash_kernel_ex4_can_flash_kernel.txt -a led_ex1_blinky.txt -v*
2. 下载内核后，它会将应用文件移动到闪存并确认应用加载已完成。

对于 LaunchPad，必须执行以下步骤：

1. 将 LaunchPad SW4 位置设置为 OFF（关闭）。这样做是为了将 CANTX 和 CANRX 信号路由到接头而不是收发器。
2. 打开命令窗口并导航至 *can_flash_programmer.exe* 所在的位置。
3. 输入带有参数的命令，如下所述：
   a. 示例：*can_flash_programmer.exe -d f28003x -k flash_kernel_ex4_can_flash_kernel.txt -a led_ex1_blinky.txt -v*
4. 下载内核后，它会将固件传输到闪存并确认应用加载已完成。

### 6.3.5 使用 CCS 编译工程

1. 在 CCS 中，导入并编译 CPU1 内核工程。
2. 启动目标配置文件。
3. 连接至 CPU1。
4. 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”。
5. 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。
6. 在仿真模式下，需要设置以下存储器位置才能启用 MCAN 引导模式：
   a. 带 0xFFFF 的位置 0xD00
   b. 带 0x5AFF 的位置 0xD01
   c. 带 0x00XX 的位置 0xD04，其中 XX 是 MCAN 引导的引导模式：0x08、0x28 或 0x48。0x68、0x88 和 0xA8 的 SENDTEST MCAN 引导模式分别使用与前三种配置相同的引脚，并且它们还发送两个 CAN-FD 帧。在评估模式下，使用其中一种 SENDTEST 模式可确保 MCAN 模块在主机开始发送闪存内核之前不会超时。要了解有关 SENDTEST 模式的更多信息，请参阅 C2000Ware 中的 MCAN 引导源文件（*C2000Ware_x_xx_xx_xx > driverlib > f28003x > examples > flash > MCAN_Boot.c*）。

图 6-6 展示了这些存储器位置的实现示例。对这些位置进行编程后，复位器件并点击“Resume”。现在，F28003x 器件应在 ROM 的 MCAN 引导模式下等待。

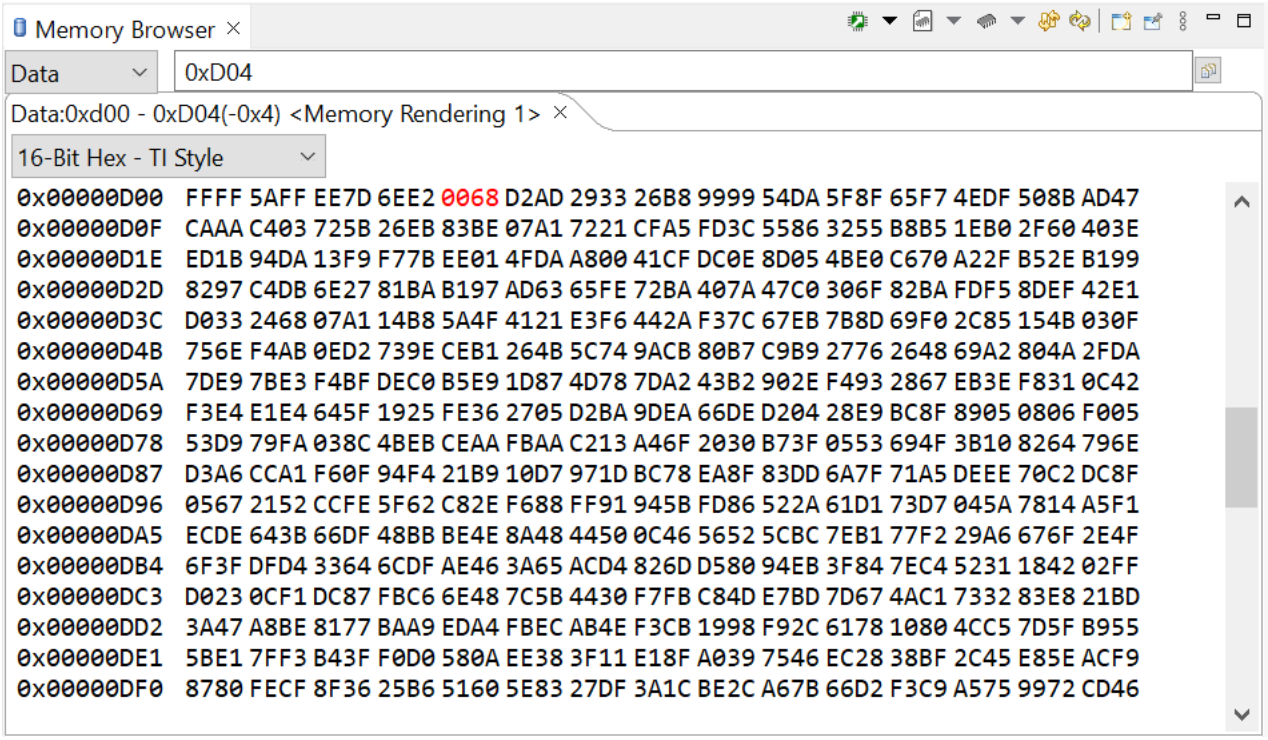


**图 6-6. CCS 中的存储器窗口示例（GPIO 引脚 4/5、SENDTEST MCAN 引导模式）**

PEAK CAN 分析仪也需要连接至 PC。无需事先进行初始化。

然后，在 *can_flash_programmer* 所在目录下的命令行中输入以下命令：

```
can_flash_programmer.exe -d f28003x -k -a
```

主机初始化 CAN 分析仪，然后打开闪存内核文件以传输到 MCAN 引导加载程序。整个文件发送完毕后，主机会延迟 5 秒钟再发送固件映像。

## 6.4 应用加载：CPU2 映像

对于 F28P65x 器件，有多个具有唯一起始地址的闪存组，因此可以通过组合 CPU1 应用映像和 CPU2 应用映像来对 CPU2 映像进行编程。

### 6.4.1 组合两个映像 (.txt)

生成的每个应用 .txt 文件其顶部都有一个标头，底部都有一个终止标头。由于存在多个起始和终止标头，这些文件无法使用文本编辑器进行组合。

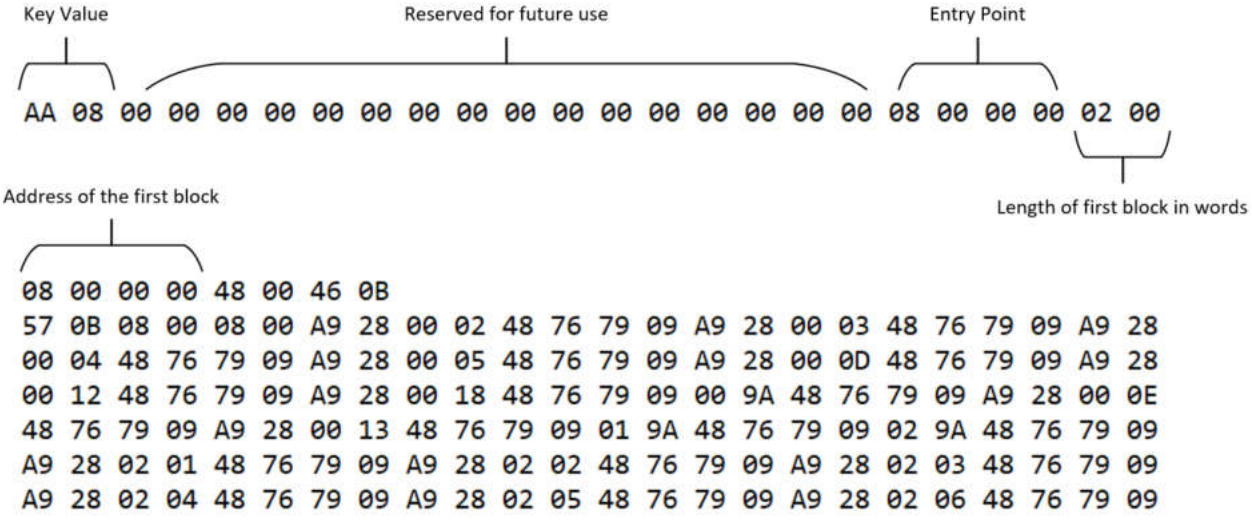


**图 6-7. 映像 A**

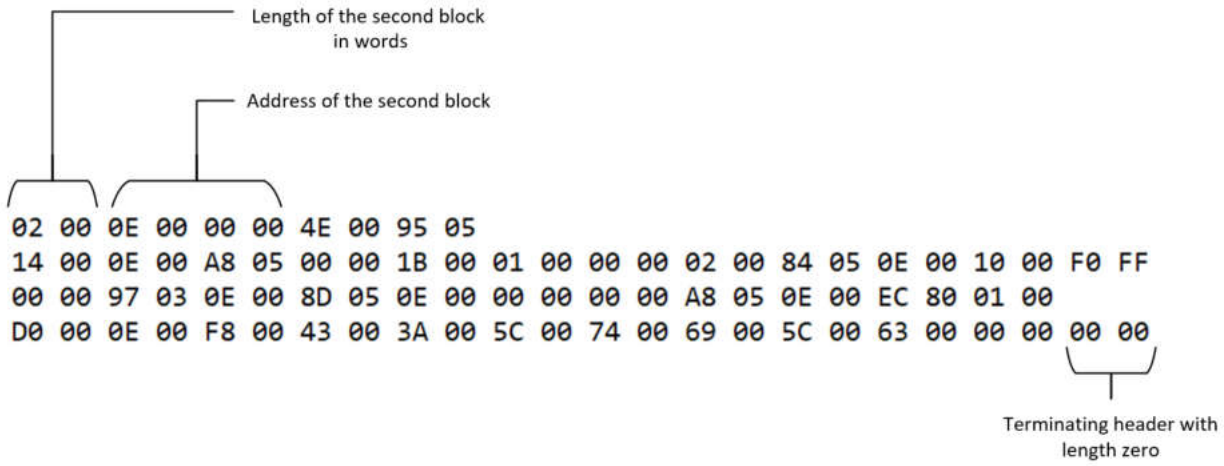


**图 6-8. 映像 B**

例如，一种解决方案是删除上面映像 B 中的终止标头和上面映像 A 中的起始标头（键值、保留字、入口点）。映像 A 和映像 B 在各自文件的顶部都有一个标头，在底部都有一个终止标头。映像 A 应附加到映像 B 的底部。删除后，我们就能成功地将两个映像合并到一个 .txt 文件中。必须注意每个映像的地址块，以确保不会丢失数据。

**备注**

合并映像 A 和映像 B 的顺序对 CAN/MCAN 引导加载程序识别 CPU1 的起点地址很重要。如果映像 A 应该是用于 CPU1 的应用映像，则应删除映像 A 的终止标头，还应删除映像 B 的顶部接头。然后，必须将映像 B 附加到映像 A 的 .txt 文件中。

# 7 疑难解答

以下是针对用户在使用 CAN 或 MCAN 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**：我找不到内核工程，它们在哪里？

**回答**：

| 器件 | 编译配置 | 位置 |
|---|---|---|
| F28003x | RAM | C2000Ware_x_xx_xx_xx > driverlib > f28003x > examples > flash |
| F28P65x | RAM | C2000Ware_x_xx_xx_xx > driverlib > f28p65x > examples > c28x_dual > flash_kernel |
| F280015x | RAM | C2000Ware_x_xx_xx_xx > driverlib > f280015x > examples > flash |

**问题:** 如果闪存内核没有下载，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正确边界对齐。F28P65x 和 F280015x DCAN 闪存内核显示其各自闪存 API 的 512 位编程能力。因此，由这两个内核加载的应用程序应采用 512 位对齐方式。同样，由 F28P55x MCAN 闪存内核加载的应用程序应采用 512 位对齐方式。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(32)”。

  对于本文档中提到的所有其他闪存内核，使用闪存 API 的 128 位编程函数。因此，由这些内核加载的应用程序应采用 128 位对齐方式。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)”。
- 用户遇到的另一个常见问题是他们没有将正确的引导引脚用于 CAN/MCAN 引导模式。例如，在 F28003x 器件上，MCAN 引导模式提供三个选项供 GPIO 引脚使用。确保默认选项的引脚没有用于其他用途。
- 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 CAN/MCAN 环回和传输示例（您应该能够在 *C2000Ware* 中或器件的 driverlib/examples 文件夹中找到这些示例）。
- PEAK PCAN USB-PRO FD 分析仪的运行 CAN-FD 时钟范围为 20MHz 至 80MHz。器件的 MCANxBIT 时钟必须与 PEAK 工具的 CAN-FD 时钟频率相匹配才能发送帧。
- *dcan_flash_programmer* 和 *can_flash_programmer* 都需要使用 PEAK PCAN USB-Pro FD 分析仪，因为该工具能够传输标准 CAN 帧和 CAN-FD 帧。PEAK PCAN USB FD 器件也是兼容的。

**问题:** 我已经组合了应用程序映像但仍然无法编程，如何解决此问题？

**回答：**确保组合 .txt 文件中的第一个映像包含 CPU1 的映像，后面是 CPU2 的映像。

## 7.2 DCAN 引导

**问题:** 下载 DCAN 内核后，我收到 DCAN 总线错误。可以采取哪些步骤来解决此问题？

**回答：**确保传输的 DCAN 内核文件具有用于 DCAN 位时序设置的正确值。文本文件的字节 3 和 4 必须替换为根据位时序寄存器值 (*CAN_CALC_BTRREG)* 的最终结果计算出的十六进制值，顺序为最低有效字节后跟最高有效字节。

**问题:** 对于 F28P65x 或 F280015x 器件，在 DCAN 引导模式下将 DCAN 闪存内核下载到 RAM 后，无法下载应用程序映像，我该怎么办？

**回答：**确保在生成 DCAN 闪存内核工程时使用的时钟使用内部晶体振荡器，即 INTOSC2 (*device.h)*。上电时，器件引导 ROM 由片上 10MHz 振荡器 (INTOSC2) 提供时钟。该值需要设置为主内部时钟源，并且是复位时的默认时钟。

**问题:** 对于 F28P65x 器件，我可以在 DCAN 引导模式下将 DCAN 闪存内核下载到 RAM 后下载应用程序映像，但 CPU2 无法执行其应用程序。如何解决此问题？

**回答：**CPU1 的应用程序映像必须设置 CPU2 的所选闪存组 GSxRAM，并将 GPIO 控制权交给 CPU2。然后，CPU1 必须在 CPU2 执行其应用程序映像之前执行自身的应用程序映像，否则，它可能会进入存储器的非法 ISR 段。有关这些项的更多详细信息，请参阅器件特定的 TRM [7]。

**问题**：F28003x、F28P65x 和 F280015x 存在示例工程，我如何调整这些工程以用于 F20013x？

**回答：**实施您自己的 F280013x DCAN 闪存内核版本相当简单。调整现有 F280015x DCAN 闪存内核以用于 F280013x 器件时，所需的更改极小。下面列出了移植工程所需的主要更改：

- 复制 C2000Ware 中的现有 .projectspec 文件 (C2000Ware_x_xx_xx_xx > driverlib > f280015x > examples > flash > CCS > f280015x_flash_eex5_dcan_flash_kernel.projectspec)
- 更改工程名称
- 将特定于 F280015x 的文件路径更改为 F280013x 的相应路径
  - 示例：<file action="copy" path="../../../../../libraries/flash_api/f280015x/lib/FAPI_F280015x_EABI_v2.00.10.lib" targetDirectory="" /> 现在应该变为 <file action="copy" path="../../../../../libraries/flash_api/f280013x/lib/FAPI_F280013x_EABI_v2.00.10.lib" targetDirectory="" />
- 将新的 .projectspec 导入 CCS
- 将对 F280015x 头文件的任何引用更改为对 F280013x 相应头文件的引用
  - 示例： FlashTech_F280015x_C28x.h > FlashTech_F280013x_C28x.h
- 验证内核是否配置了正确的 GPIO 和引导模式

### 7.3 MCAN 引导

**问题:** 对于 F28P65x 或 F28P55x 器件，在 MCAN 引导模式下将 MCAN 闪存内核下载到 RAM 后，无法下载应用程序映像，我该怎么办？

**回答：**确保在生成 MCAN 闪存内核工程时使用的时钟使用内部晶体振荡器，即 INTOSC2 (*device.h)*。上电时，器件引导 ROM 由片上 10MHz 振荡器 (INTOSC2) 提供时钟。该值需要设置为主内部时钟源，并且是复位时的默认时钟。

**问题:** 对于 F28P65x 器件，我可以在 MCAN 引导模式下将 MCAN 闪存内核下载到 RAM 后下载应用程序映像，但 CPU2 无法执行其应用程序。如何解决此问题？

**回答：**CPU1 的应用程序映像必须设置 CPU2 的所选闪存组 GSxRAM，并将 GPIO 控制权交给 CPU2。然后，CPU1 必须在 CPU2 执行其应用程序映像之前执行自身的应用程序映像，否则，它可能会进入存储器的非法 ISR 段。有关这些项的更多详细信息，请参阅器件特定的 TRM [7]。

## 8 参考资料

1. 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
2. 德州仪器 (TI)：*TMS320C28x 汇编语言工具用户指南*
3. 德州仪器 (TI)：*C2000 微控制器的 USB 闪存编程*
4. 德州仪器 (TI)：*TMS320F28003x 引导特性和配置*
5. 德州仪器 (TI)：*LAUNCHXL-F280039C 概述用户指南*
6. 德州仪器 (TI)：*TMS320F28003x 实时微控制器技术资源手册*
7. 德州仪器 (TI)：*TMS320F28P65x 实时微控制器技术资源手册*
8. 德州仪器 (TI)：*控制器局域网 (CAN) 简介*
9. 德州仪器 (TI)：*MCAN (CAN FD) 模块入门*

## 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 * (December 2023) to Revision A (April 2024)** **Page**

# 重要声明和免责声明

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 产品的应用。严禁对这些资源进行其他复制或展示。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