Application Note

MSPBoot – 适用于 MSP430™ 闪存微控制器的主存储器引导加载程序

Luis Reynoso, Caleb Overbay *MSP430 Apps*

摘要

本应用手册介绍了如何实现在 MSP430™ 基于闪存的微控制器 (MCU) 的主存储器中驻留的引导加载程序，此引导加载程序能够使用内部集成电路 (I^2C)、通用异步接收器/发送器 (UART) 或串行外设接口 (SPI) 总线以及 CC110x 射频收发器来完成无线下载 (OAD)。此引导加载程序虽然高度灵活且模块化，但占用空间小，因此是一种非常具有成本效益的解决方案，并支持大型存储器模型（存储器空间大于 64KB 的器件）。

可从以下 URL 下载一个软件包，其中包含用于主机器件和目标器件的示例和源代码：https://www.ti.com.cn/tool/cn/download/MSPBOOT

如需了解介绍如何运行示例的分步过程，请参阅节 4.2.5。

请勿将此引导加载程序与 MSP430 引导加载程序 (BSL) 混淆，后者存储在某些 MSP430 MCU 的受保护存储器（ROM 或闪存）中。有关 MSP430 BSL 的更多信息，请参阅 *MSP430™ 闪存器件引导加载程序 (BSL) 用户指南* 和 *创建基于闪存的定制引导加载程序 (BSL)*。

备注

MSP430FRBoot 是 MSPBoot 的扩展，其使用各种通信接口和无线下载 (OAD) 功能为 MSP430 FRAM MCU 实现驻留在主存储器中的引导加载程序。MSP430FRBoot 支持大型存储器模型（存储器空间大于 64KB 的器件）以及双映像和中断重定向选项，因此是替代 MSP430 FRAM MCU 内置 BSL 的出色可定制方案。

内容

商标

1 引言

MSP430 MCU 配备了引导加载程序 (BSL)，通过该 BSL 可以轻松进行现场升级。有关 MSP430 BSL 的更多信息，请参阅 *MSP430™ 闪存器件引导加载程序 (BSL) 用户指南* 和 *创建基于闪存的定制引导加载程序 (BSL)*。该 BSL 在 MSP430F5xx 和 MSP430F6xx 器件中是可定制的，因为它位于闪存中。

其它系列（例如 MSP430G2xx）具有存储在 ROM 中的 BSL，该 BSL 仅支持 UART，不能进行修改来支持 I^2C 或其他接口。鉴于这些限制，有必要创建一个存储在主存储器中并有助于轻松实现应用程序的引导加载程序。

本应用报告介绍如何实现名为 MSPBoot 并具有以下特征的引导加载程序：

- 占用空间小（所需大小小于 4KB）
- 整合了 20 位和 16 位，分别用于大型存储器模型器件和小型存储器模型器件
- 支持 USI、USCI 和 eUSCI 外设
- UART 通信使用较小的存储空间提供最简单的有线接口
- SPI 总线提供无线下载（使用 CC110x），占用空间稍大
- 不同的选项支持可定制的稳健性级别
- 可选的双映像支持可应对通信中断问题
- 允许在应用程序中使用所有中断
- 应用程序可以选择重复使用引导加载程序中的低级通信驱动程序，也可以实现自己的驱动程序
- 可配置的 Boot 进入方式
- 使用 CRC-CCITT 对应用程序进行可选的验证
- 提供源代码，允许进行其他定制

该引导加载程序随附源代码，其中包括不同示例配置、应用程序示例和主机示例，以便简化测试、定制和实现。本应用报告假定您已了解 I^2C、UART 和 SPI 规范以及低于 1GHz 射频通信协议。

1.1 术语表

BOR	欠压复位
BSL	MSP430 引导加载程序
CI	MSPBoot 通信接口
CRC	循环冗余校验
eUSCI	增强型通用串行通信接口
I^2C	内部集成电路
MCU	微控制器
MI	MSPBoot 存储器接口
MSPBoot	本应用报告介绍的引导加载程序
MSP430FRBoot	MSP430FRBoot - 适用于 MSP430™ FRAM 大型存储器模型器件的主存储器引导加载程序和无线更新介绍的引导加载程序
OAD	无线下载
OSI	开放系统互连
PUC	上电清除复位
ROM	只读存储器
SPI	串行外设接口
UART	通用异步收发器
USCI	通用串行通信接口
USI	通用串行接口

1.2 约定

本文档包含的 I²C 传输示例使用以下约定：

S：启动
P：停止
Sr：重复开始
主到从
从到主
A：响应（SDA 低电平）
/A：无应答（SDA 低电平）
R：R/W 位 =1
W: R/W 位 = 0

UART 传输示例使用以下格式：

主机到目标
目标到主机
St：启动
SP：停止

SPI 传输示例使用以下格式：

主到从
从到主
X: 不用考虑

2 执行

使用模块化方法可以在 MSP430 器件之间轻松迁移，并可以定制每一层。图 2-1 显示了软件层。

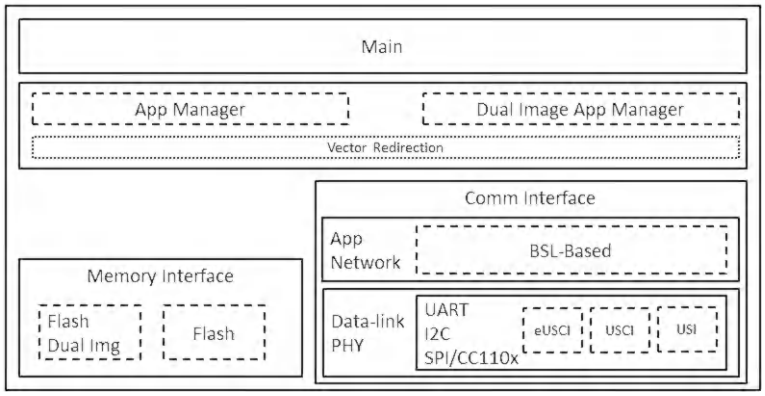


图 2-1. MSPBoot 软件架构

以下各节将更详细地介绍每个模块。

2.1 主例程

主例程具有以下用途：

- 初始化 MSP430 MCU 的基本功能。
- 初始化其他 MSPBoot 层。
- 实现用于轮询通信接口和处理命令的主循环。

图 2-2 显示了主例程的状态图：

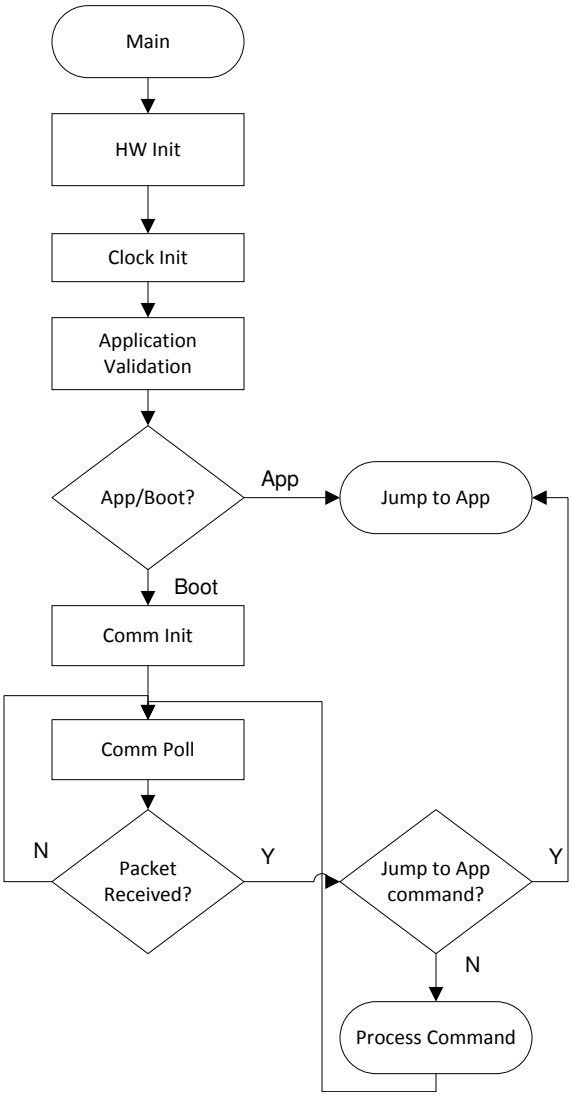


图 2-2. 主例程的流程图

2.2 应用程序管理器

应用程序管理器的主要目的是：

- 检测器件何时应处于引导加载程序模式或应用程序模式
- 验证应用程序
- 重新映射中断向量
- 从引导加载程序跳转到应用程序
- 在双映像模式下恢复有效映像

2.2.1 引导和应用程序检测

应用程序管理器通过运用以下规则来检测应该执行引导加载程序还是应用程序：

- 在以下情况下执行应用程序：
  - 应用程序有效（请参阅节 2.2.1.2）

    并且
  - 外部事件或应用程序未强制使用引导加载程序（请参阅节 2.2.1.1）
- 在以下情况下执行引导加载程序：
  - 外部事件或应用程序强制使用引导加载程序

    或者
  - 应用程序无效

图 2-3 显示了此决策过程。

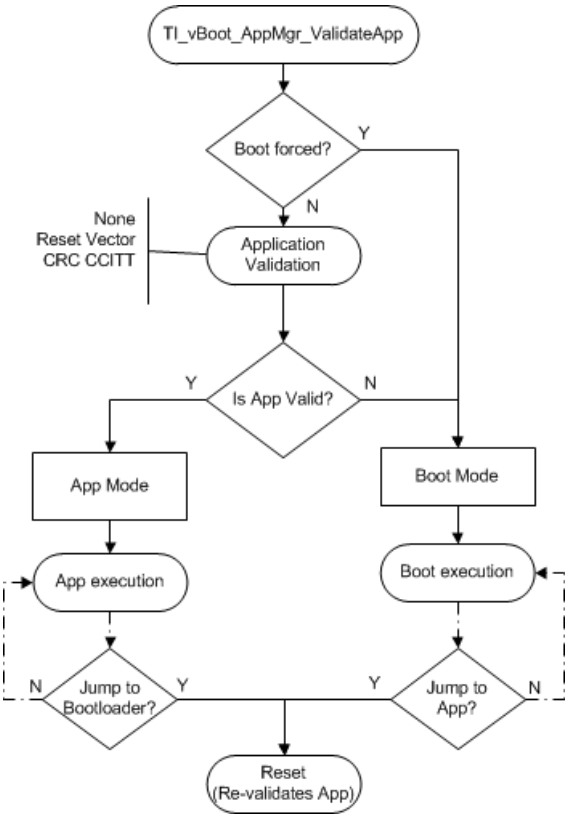


图 2-3. AppManager 执行的应用程序验证

2.2.1.1 强制使用引导加载程序模式

即使应用程序有效，也可以通过以下选项强制使用引导加载程序模式：

- 选项 1：外部事件，例如复位后的 GPIO 状态。

  默认情况下，软件会在复位后检查以下 GPIO 是否为低电平以强制使用引导加载程序模式：

  - MSP430G2xxx 中的 P1.3（MSP-EXP430G2 中的 S2 按钮）
  - MSP430F5529 中的 P1.1（MSP-EXP430F5529 中的 S2 按钮）

  可根据需要在 TI_MSPBoot_AppMgr_BootisForced() 中修改此事件。

- 选项 2：应用程序要求执行引导加载程序模式。

  变量 StatCtrl 和 PassWd 是保留变量并在应用程序和引导加载程序之间共享。为强制使用引导加载程序模式，应用程序会将这些变量设置为：

  PassWd = 0xC0DE

  StatCtrl.BIT0 = 1

2.2.1.2 应用程序验证

应用程序验证机制允许引导加载程序在执行应用程序之前对其进行验证。可采用三种方法支持不同级别的代码规模和安全性：

- None：不验证应用程序，认为应用程序始终有效。可通过外部事件强制使用引导加载程序模式。不推荐。
- 复位向量：如果复位向量不同于 0xFFFF（已擦除状态），则认为应用程序有效并执行应用程序。
- CRC_CCITT：计算整个应用程序映像的 CRC CCITT，并将结果与期望值进行比较。请注意，基于 BSL 的协议（请参阅节 2.4.2.1）使用 CRC CCITT，因此在使用基于 BSL 的协议时建议使用此验证方法。此外，这个方法不考虑修改闪存内容（例如数据记录）的应用程序。当执行这种类型的应用程序时，存储在闪存中的 CRC CCITT 会变得不正确，即使可能有效，引导加载程序也不会允许应用程序运行。开发可以修改闪存的应用程序时，建议使用复位矢量验证方法。

请注意，验证方法可以防止执行损坏的应用程序，但是不能确保应用程序的完整性和功能，这是用户的责任。如果应用程序不具有预期的功能，则仍可以使用硬件进入序列来恢复 MSP430 器件。

2.2.1.3 跳转到应用程序

当通信协议检测到下载完成并且器件应跳转到应用程序时，MSPBoot 会强制进行复位。

支持软件 BOR 的器件（例如 MSP430F5529）使用此方法强制进行复位，这是将 MSP430 恢复到默认状态的有效方法。定义了 HW_RESET_BOR 后会启用此方法。

对于不具有这种机制的器件（例如 MSP430G2xx），可使用 PUC 复位。这属于强制性复位，但不会清除所有寄存器。定义 HW_RESET_PUC 后，引导加载程序将清除相关的寄存器。

2.2.2 矢量重定向

MSPBoot 无法擦除或重新编程引导加载程序区域。此限制提供了更安全的实现方案，因为引导加载程序始终可访问，并可以通过强制使用引导加载程序模式来恢复 MCU。

复位矢量是引导加载程序必不可少的组成部分，因为其可以强制 MCU 始终跳转到引导加载程序的进入序列，所以不应将其擦除。由于复位矢量位于 16 位闪存空间 (0xFFFE) 的顶端，因此引导加载程序代码位于相邻位置（请参阅图 2-4）。

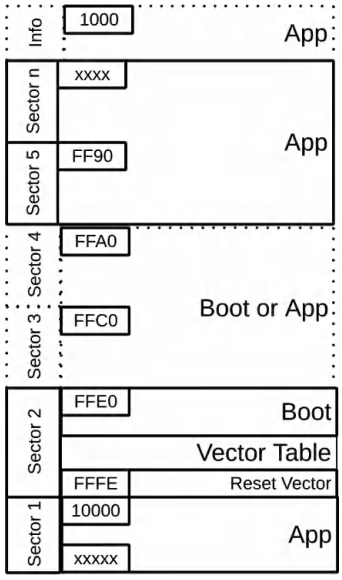


图 2-4. 存储器分配

中断矢量表也位于受保护的引导区域中。因为预计中断表的值将根据应用程序而变化，所以这意味着必须遵循一些特殊注意事项以允许应用程序中断。在大型存储器模型器件（0x10000 及更高）中，应用程序可以使用额外的 20 位空间。

2.2.3 闪存器件中的中断矢量

闪存擦除的最小大小为一个段，在 MSP430 MCU 中为 512 字节。考虑到这一点，应保护整个中断表不被擦除。为了在应用程序级别允许中断，我们实现了一种软件矢量重定向方法来修复默认矢量表的内容并指向存储在应用程序空间中的代理表。

图 2-5 显示了此实现背后的概念：

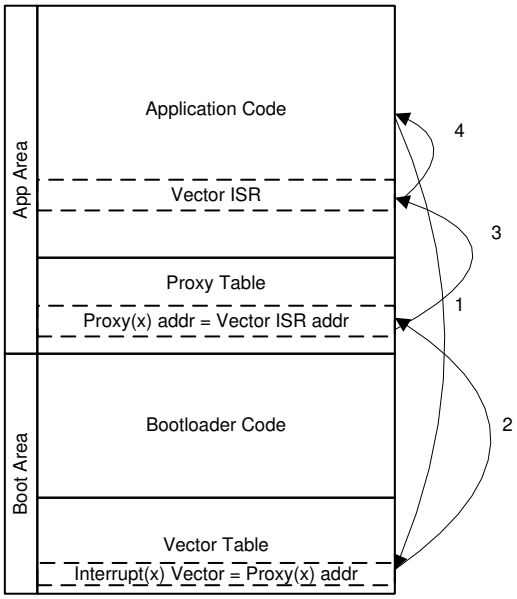


图 2-5. 矢量重定向实现

1. 应用程序收到一个中断请求，当前地址将压入堆栈，然后 CPU 从矢量表中获取地址。
2. 矢量表包含每个中断的代理位置的地址。CPU 获取代理表中相应条目的地址并跳转到该地址。
3. 代理表包含分支指令 (BRA)，后跟每个 ISR 的实际地址。CPU 执行 BRA 指令，并跳转到相应的应用程序矢量 ISR。
4. ISR 完成后将执行 RETI 指令，并从堆栈中弹出前一个地址。

此过程对于应用程序的实现几乎是透明的，但是必须注意，由于从代理表到应用程序 ISR 的额外跳转，延迟会增加。

示例代码中包含了演示如何实现中断的应用示例以演示此功能。

请注意，某些 MSP430 MCU 支持将矢量重定向到硬件中的 RAM (SYSRIVECT)，这对于具有足够 RAM 的器件而言可能是一个很好的选择。

2.2.4 双映像支持

应用程序管理器还可以支持双映像模式。在这种模式下，即使映像下载中断或新下载的映像损坏，也总是希望主存储器中存在有效的应用程序。此模式对于 OAD 至关重要，因为 OAD 可能会意外中断通信。

在双映像模式下，主存储器被分割成下载区域和应用程序区域，如节 2.3.1 所述。此模式下的应用程序验证过程不同于常规的过程，如图 2-6 所示。

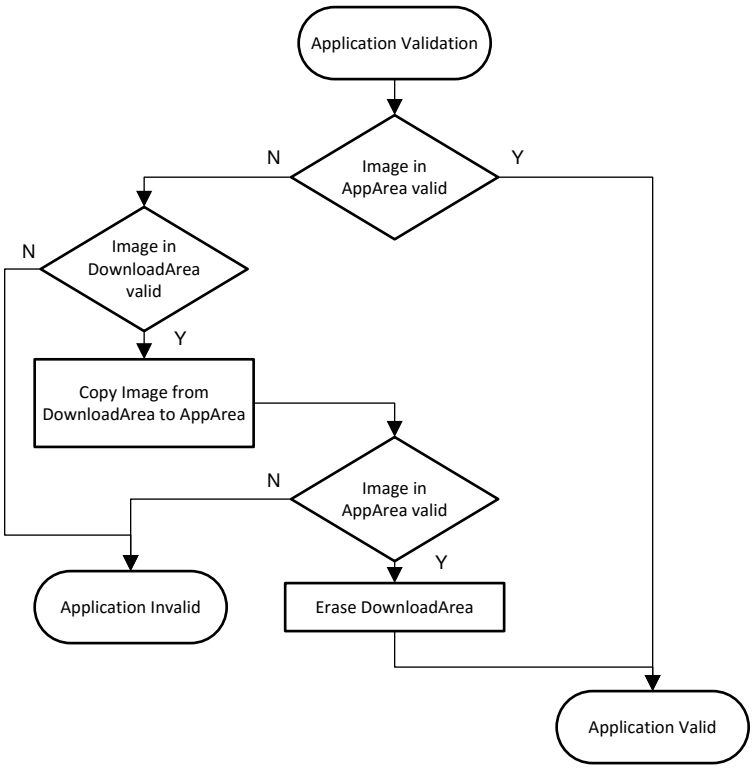


图 2-6. 双映像应用程序验证

2.2.4.1 在双映像模式下跳转到应用程序

应用程序下载过程完成后，MSPBoot 在跳转到新应用程序之前将执行以下步骤：

1. 验证下载区域中的新映像
   a. 如果无效，则退出（复位会再次强制使用引导加载程序，并仅在原始映像有效时才执行应用程序）
   b. 如果有效，则继续
2. 将应用程序区域替换为下载区域
3. 验证应用程序区域中的映像
   a. 如果有效，则擦除下载区域（复位应执行应用程序，因为应用程序区域中的映像有效）
   b. 如果无效，则退出（意外状态，但复位将再次重新验证两个映像）

2.3 存储器接口 (MI)

为了保护引导加载程序区域，存储器在逻辑上分为两部分：

- 应用领域：这是包含用户应用程序和已重定向矢量表的可写区段
- 引导加载程序区域：这是包含引导加载程序和矢量表的不可写区段

每个扇区的大小在工程链接器文件中定义。Code Composer Studio™ IDE (CCS) 的示例工程中提供了显示不同存储器大小的示例。存储器接口提供了一个 API 可用于对应用程序存储器区域进行编程和擦除以及保护引导加载程序区域。对于闪存器件，这种存储器保护的实现方式如下：

- 不执行批量擦除，而是使用段擦除来擦除应用程序。
- 验证要擦除或编程的地址，以免引导加载程序区域发生意外损坏

备注

MSPBoot 在执行更新时不允许对引导加载程序区域进行写入或擦除访问，但在执行应用程序时无法防止意外擦除。

2.3.1 双映像支持

启用双映像支持后，存储器接口模块会将 MSP430 应用程序区域划分为两个子部分，从而产生以下逻辑存储器映射：

- 非引导区域
  - 下载区域：此区段用作存储新应用程序映像的临时缓冲区。主机无法访问该区域中的物理地址，但是当主机尝试下载到应用程序区域中的逻辑地址时，将写入该区域。
  - 应用领域：此区段用于执行当前应用程序映像。主机可以使用该区域中的逻辑地址，但是主机无法写入物理地址。验证下载区域中的新映像后，引导加载程序会更新应用程序区域。节 2.2.4 对此过程进行了说明。
- 引导区域
  - 这是包含引导加载程序和矢量表的只读部分。

每个扇区的大小在工程链接器文件中定义。在 CCS 的示例工程中提供了显示不同存储器大小的示例。

2.4 通信接口 (CI)

通信接口的目的是：

- 从主机接收数据并将数据发送到主机
- 实现通信协议
- 解析数据、验证数据包并执行适当的命令
- 根据函数的输出，生成响应

根据开放系统互连 (OSI) 模型，CI 分为两个模块：

- Physical-DataLink (PHY-DL)
- Network-Application (NWK-APP)

2.4.1 Physical-DataLink (PHY-DL)

PHY-DL 层提供了一个硬件抽象层 (HAL) 来简化向不同 MSP430 衍生产品或外设迁移的过程。PHY-DL 层提供了一个稳定的通道向主机发送数据和从主机接收原始数据。当前的引导加载程序最初是使用 I²C、UART 或 SPI 实现的。该引导加载程序当前支持 USI、USCI 和 eUSCI 模块，但是如果需要，可以包括其他选项。

通过使用表 2-1 中的回调函数提供指向结构的指针来初始化 PHY-DL 层。

表 2-1. PHY-DL 回调结构

t_CI_Callback	结构类型定义
.RxCallback	收到新字节时调用
.TxCallback	需要传输字节时调用
.ErrorCallback(1)	在 PHY-DL 中检测到错误（超时）时调用
.StartCallback(1)	在检测到数据包开始时调用
.StopCallback(1)	在检测到数据包结束时调用

(1) 回调是可选的。协议或 CI 可能不需要回调。

更高级别的层 (NWK-APP) 使用回调函数来实现通信协议。根据协议的不同，某些回调不是必需的，因此可以在 PHY-DL 层中将它们禁用以减少占用空间。节 2.4.2 中介绍了 NWK-APP 层。

2.4.1.1 I²C

多年来，由于 I²C 总线实施成本低且具有稳健性和灵活性，因此一直是嵌入式应用中最常见的通信协议之一。对于需要简单通信链路的低成本应用来说，这是一种有用的接口。该接口能够与 MSP430 微控制器完美匹配，此系列的微控制器在一个具有成本效益的解决方案中融合了高性能、高集成度和超低功耗等特性。I²C 接口是使用具有 MSP430 MCU 预定义地址的 7 位寻址格式来实现的：

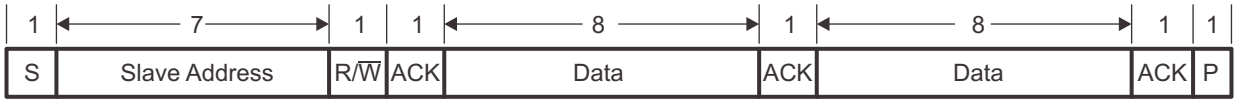


图 2-7. I²C 7 位寻址格式

预定义的 MSP430 MCU 地址定义为：

CONFIG_CI_PHYDL_I2C_SLAVE_ADDR = 0x40

为支持 USI、G2xx3 USCI、F5xx/6xx USCI 和 eUSCI，还包含相关文件。

2.4.1.1.1 超时检测

在 I²C 通信期间，当从设备需要更多时间来处理数据包时，将时钟线保持在低电平是有效的。这种机制称为时钟拉伸，尽管这非常有用，但也可能导致器件无限期保持总线，从而使总线停止。

PHY-DL 层可以选择性检测线路保持太长时间的情况，如果检测到这种情况，PHY-DL 层可以复位接口。

此功能是根据 CONFIG_CI_PHYDL_TIMEOUT 启用的。USCI 和 USI 实现方案使用 TA1 来实现此功能，而 eUSCI 包括此功能的硬件支持。

2.4.1.2 UART

UART 接口使用 8-N-1 格式（8 个数据位，无奇偶校验位，1 个停止位）实现（请参阅图 2-8）。

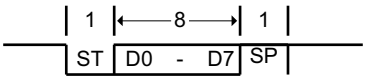


图 2-8. UART 8-N-1 格式

默认波特率定义为 CONFIG_CI_PHYDL_UART_BAUDRATE = 9600。

2.4.1.3 SPI

用于 CC110x 通信的 SPI 接口是使用以下配置实现的（另请参阅图 2-9）：

- 8 位数据
- MSB 在前
- 时钟极性 = 0（非活动状态为高电平）
- 时钟相位 = 1（数据在第一个时钟边沿改变，在随后的边沿被捕捉）
- 3 引脚配置，使用 GPIO 实现 SS

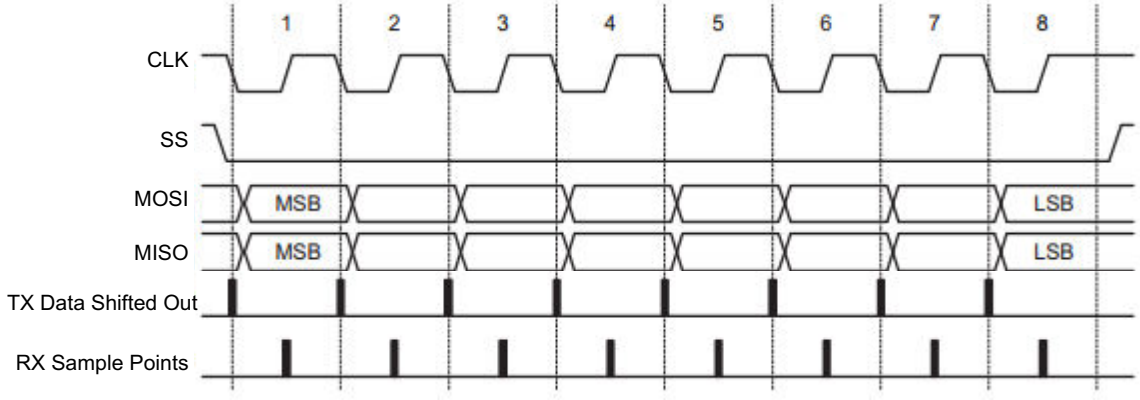


图 2-9. SPI 格式

2.4.1.4 CC110x

CC110x 器件使用表 2-2 中所示的数据包结构发送数据。

表 2-2. CC110x 数据包结构

标头	长度(1)	命令	地址(2)	数据(2)	校验码
0x80	N	1 字节	3 字节	N-6 字节	2 字节

(1) 数据包的最大长度为 24 字节；因此，每个数据包允许的最大数据字节数 (N) 为 16。
(2) 如果长度等于 1（仅命令），这些子部分不包括在数据包中。

MSPBoot 中 CC110x 的配置如下：

- 数据速度为 250kbps
- 载波频率为 902750Hz

通过 TI_MSPBoot_CI_PHYDL_CC1101.c 内部 radio_init 函数发送的变量，可以将数据速度设置为 1.2 或 38.4kbps。可在 TI_MSPBoot_Config.h 中更改无线电频率。必须同时更改目标和主机固件工程。请参阅 CC1101 低功耗 Sub-1GHz 射频收发器数据表，了解有关常见 CC1101x 命令的更多信息以及其他通信详细信息。

数据包结构与基于 BSL 的协议相同，因此可以使用预期的格式将其直接从 PHY-DL 层传输到 NWK-APP 层。节 2.4.2.1.2 也对此进行了介绍。

2.4.1.5 通信共享

用户应用程序可以根据需要使用通信接口（I^2C、UART、GPIO 或其他用途），因为当 MCU 跳转到应用程序时，资源将被释放。根据需要，CI PHY-DL 可与应用程序共享，从而允许其使用同一个通信接口并减少应用程序占用空间。启用此功能后，引导加载程序将共享表 2-3 中的函数指针。

表 2-3. Boot2App_Vector_Table 定义

Boot2App_Vector_Table	此表包含共享 CI PHY-DL 函数的地址
TI_MSPBoot_CI_PHYDL_Init	用于初始化 PHY-DL 的函数，该函数将一个指针传递给应用程序 t_CI_Callback。
TI_MSPBoot_CI_PHYDL_Poll	此函数检查所有相关标志并在需要时调用相应的回调
TI_MSPBoot_CI_PHYDL_TxByte (1)	用于写入 TX 缓冲区的函数

(1) 仅为 SPI 和 UART 实现回调，I^2C 不需要。

应用程序必须声明自己的回调，这些回调在 CI PHY-DL 初始化期间传递，并在检测到相应事件时调用。PHY-DL 层的设计将占用空间小作为最重要的一个考虑因素。如果 PHY-DL 的实现不够充分，应用程序始终可以实现自己的驱动程序。随附软件包中的应用程序 2 示例展示了如何共享 CI PHY-DL。

2.4.2 NWK-APP

利用 CI Network-Application 层，可实现通信协议，解释来自 PHY-DL 的原始数据，并在执行适当的命令之前验证此类数据。为简便起见，MSPBoot 仅使用基于 BSL 的协议。

2.4.2.1 基于 BSL 的协议

MSP430 BSL 是 MSP430 MCU 中包含的标准引导加载程序。*MSP430™ 闪存器件引导加载程序 (BSL) 用户指南* 对此进行了详细说明。

在 MSPBoot 中实现的基于 BSL 的协议可以保持稳健性，但是并不能实现所有命令，并且与 BSL 协议完全采用相同的格式来减少其占用空间。此协议*基于数据包*，具有表 2-4 中的格式。

表 2-4. 基于 BSL 的协议命令格式

标头	长度	有效载荷	校验码 [L]	校验码 [H]
0x80	1 到 PAYLOAD_MAX_SIZE(1)	1 到 PAYLOAD_MAX_SIZE 字节	1 字节	1 字节

(1) PAYLOAD_MAX_SIZE 默认设置为 20（1 个命令字节 + 3 个地址字节 + 16 个数据字节）

标头：固定为 0x80

长度：1 字节加有效载荷长度。有效值为 1 到 PAYLOAD_MAX_SIZE。

有效载荷：1 到 PAYLOAD_MAX_SIZE 字节，包含命令、可选地址和数据（可选，根据命令类型而定）。

校验和：有效载荷的 16 位 CRC CCITT

表 2-5 中的命令会作为有效载荷来实现。

表 2-5. 基于 BSL 的协议命令

命令	CMD	字节 $_1$	字节 $_2$	字节 $_3$	字节 $_4$	⋯	字节 $_{length-1}$
ERASE_SEGMENT	0x12	ADDR[L]	ADDR[M]	ADDR[H]	X	X	X
ERASE_APP	0x15	X	X	X	X	X	X
RX_DATA_BLOCK	0x10	ADDR[L]	ADDR[M]	ADDR[H]	DATA0	X	DATAn
TX_VERSION	0x19	X	X	X	X	X	X
JUMP2APP	0x1C	X	X	X	X	X	X

ERASE_SEGMENT

擦除由 ADDR 寻址的存储器段（闪存中为 512 字节）。

ERASE_APP

擦除应用程序区域。

RX_DATA_BLOCK

编程从地址 ADDR 开始的 *n* 字节数据。

TX_VERSION

向目标请求 MSPBoot 版本。

JUMP2APP

指示目标跳转到应用程序映像（在验证之后）。来自目标的响应始终为一个字节（表 2-6 列出了有效值）。

表 2-6. 基于 BSL 的协议从设备响应

响应	值	说明
OK	0x00	上一条命令正确执行
HEADER_ERROR	0x51	帧头不正确
CHECKSUM_ERROR	0x52	帧校验和不正确
PACKETZERO_ERROR	0x53	数据包长度 = 0
PACKETSIZE_ERROR	0x54	数据包长度 > MAX_LEN
UNKNOWN_ERROR	0x55	协议错误
INVALID_PARAMS	0xC5	命令收到的参数不正确
INCORRECT_COMMAND	0xC6	收到的命令无效
MSPBOOT_VERSION	0 到 0xFF	发送为 TX_VERSION 命令的响应（默认为 0xA0）

2.4.2.1.1 安全性

每个数据包的内容均通过 16 位 CRC 验证，从而为引导加载程序提供更高的稳健性。主机可以检查每条命令的结果，如果上一条命令未成功执行，则会重试。

ERASE_SEGMENT 和 RX_DATA_BLOCK 命令可以擦除和写入 16 位存储器映射中的任何区域，因此可能损坏引导加载程序。为避免这种可能性，TI 建议在编程或擦除操作之前包含 CONFIG_MI_MEMORY_RANGE_CHECK MI 定义以验证地址。如果该过程中断，应用程序区域可能会损坏，因此 TI 建议使用节 2.2.1.2 中所述的应用程序验证方法之一，或使用双映像方法。

2.4.2.1.2 使用 CC110x 的基于 BSL 的协议

该协议的 CC110x 实现方案遵循与 UART 相同的准则，但是由于信息是以完整的数据包而不是以字节接收的，因此包含一些细微的更改。由于表 2-2 中列出的传入 CC110x 数据包与表 2-4 中预期的基于 BSL 的协议相同，因此可以将来自 PHY-DL 层的数据直接传输到 NWK-APP 层，而无需进行转换。

2.4.2.1.3 采用 I²C 的示例

将 I²C 与基于 BSL 的协议一起使用时，需要了解以下注意事项：

- 主机始终启动传输（R 或 W）。
- 数据包必须包含从设备的地址。所有其他地址将被忽略。
- 如果从设备尚未准备好处理数据或发送响应，它将使时钟保持低电平（在 I²C 中称为“拉伸时钟”）
- 请注意，不同的命令具有不同的处理时间
- 示例：主机从 MCU 读取版本

S	0x40	W	A	0x80	A	0x01	A	0x19	A	0xE8	A	0x62	A	P
	Addr			Header		Length		TX_ VERSION		Checksum _L		Checksum _H		

S	0x40	R	A	0xA0	/A	P
	Addr			Version		

- 示例：主机将 16 个字节写入地址 0xC000。

S	0x40	W	A	0x80	A	0x14	A	0x10	A	0x00	A	0xC0	A	0x00	A	0x03	A	0xEE	A
	Addr			Header		Length		RX_DATA_ BLOCK		AddrL		AddrM		AddrH		Data0		Data1	

0x47	A	0xFF	A	0xB2	A	0x40	A	0x80	A	0x5A	A	0x20	A	0x01	A	0xD2	A
Data2		Data3		Data4		Data5		Data6		Data7		Data8		Data9		Data10	

0xD3	A	0x22	A	0x00	A	0xD2	A	0xD3	A	0x15	A	0xE4	A	P
Data11		Data12		Data13		Data14		Data15		Checksum _L		Checksum _H		

S	0x40	R	A	0x00	/A	P
	Addr			OK		

2.4.2.1.4 采用 UART 或 CC110x 的示例

将 UART 与基于 BSL 的协议一起使用时，需要了解以下注意事项：

- 不需要地址，因为预计通信将是点对点的通信。
- UART 中的所有字节均按照节 2.4.1.2 中所述采用 8-N-1 格式。
- 目标准备就绪时（而不是主机请求时）将以命令的结果响应
- 主机应在发送命令后等待目标的响应，最好是超时。
- 不同的命令具有不同的处理时间。
- 示例：主机擦除 MCU 应用程序区域

0x80	0x01	0x15	0x64	0xA3
Header	Length	ERASE_APP	Checksum _L	Checksum _H

目标器件将处理命令，并在准备好后提供响应结果。

0x00
OK

将 CC110x 与基于 BSL 的协议一起使用时，适用同样的注意事项。对于这些注意事项，例外情况是节 2.4.2.1.2，其中指出从 CC110x 接收的数据包中所有字节均采用基于 BSL 的协议的预期格式；因此，这些字节可以直接从 PHY-DL 传输到 NWK-APP。尽管处理时间是相同的，但是无线通信的预期速度可能比 UART 慢一些，并且主机等待时间应该会延长以进行补偿。

3 定制 MSPBoot

MSPBoot 设计为具有低成本和小占位空间；但是，某些应用需要具有更高级别的安全性和稳健性，或者可以从这些特性中受益。根据应用要求，MSPBoot 代码中提供了不同级别的定制，并且可以根据特定需求进行调整。可通过添加适当的文件或通过启用和禁用某些预处理程序定义来选择这些选项。表 3-1 列出了可以在 TI_MSPBoot_Config.h 中配置的选项。

表 3-1. 可选配置

值	说明	对代码大小的影响
NDEBUG		
已定义	忽略 ASSERT_H 函数。启用看门狗。	-
未定义	调试期间使用。选中 ASSERT_H 函数。禁用看门狗。	增加大约 20 字节
CONFIG_MI_MEMORY_RANGE_CHECK		
已定义	确认要擦除或编程的地址在应用程序区域内。	增加大约 44 字节
未定义	不验证要擦除或编程的地址。主机必须发送正确的地址。	-
CONFIG_APPMGR_APP_VALIDATE		
1	不验证应用程序	-
2	通过检查 CRC-CCITT 验证应用程序。	增加大约 6 字节
CONFIG_CI_PHYDL_COMM_SHARED		
已定义	通信接口 PHY-DL 层与应用程序共享。	增加大约 28 字节
未定义	CI PHY-DL 不与应用程序共享。	-
CONFIG_CI_PHYDL_I2C_TIMEOUT		
已定义	在 CI PHY-DL 中检测超时。	增加大约 48 到 62 字节
未定义	CI PHY-DL 不检测超时。	-

表 3-1. 可选配置（续）

值	说明	对代码大小的影响
CONFIG_CI_PHYDL_START_CALLBACK		
已定义	在检测到“启动”时，调用回调函数（仅某些协议或通信接口才需要）。	增加大约 12 字节
未定义	在检测到“启动”时，不调用回调函数。	-
CONFIG_CI_PHYDL_STOP_CALLBACK		
已定义	在检测到“停止”时，调用回调函数（仅某些协议或通信接口才需要）。	增加大约 38 到 54 字节
未定义	在检测到“停止”时，不调用回调函数。	-
CONFIG_CI_PHYDL_ERROR_CALLBACK		
已定义	在检测到超时错误时，调用回调函数（仅某些协议或通信接口才需要）。	增加大约 16 到 20 字节
未定义	在检测到超时错误时，不调用回调函数。	-
CONFIG_CI_PHYDL_CC1101_FREQUENCY		
已定义	定义 CC110x 通信的频率	-
未定义	-	-
CONFIG_CI_PHYDL_UART_BAUDRATE		
已定义	定义 UART 通信的波特率	-
未定义	-	-
CONFIG_CI_PHYDL_I2C_SLAVE_ADDR		
已定义	定义使用 I²C 通信时 MSP430 响应的地址	-
未定义	-	-

如果要选择其他定制方案，可通过在工程中添加和使用适当的文件来实现。表 3-2 列出了工程中可互换的文件。

表 3-2. 定制文件

CI PHY-DL	
TI_MSPBoot_CI_PHYDL_USI_I2C_Slave.c	使用 USI 作为 I²C 从设备
TI_MSPBoot_CI_PHYDL_USCI_I2C_Slave_x2xx.c	使用 USCI 作为 x2xx 器件上的 I²C 从设备
TI_MSPBoot_CI_PHYDL_USCI_I2C_slave.c	使用 USCI 作为 I²C 从设备
TI_MSPBoot_CI_PHYDL_USI_I2C_slave.c	使用 USI 作为 I²C 从设备
TI_MSPBoot_CI_PHYDL_eUSCI_I2C_slave.c	使用 eUSCI 作为 I²C 从设备
TI_MSPBoot_CI_PHYDL_USCI_UART_x2xx.c	使用 USCI 作为 x2xx 器件上的 UART
TI_MSPBoot_CI_PHYDL_USCI_UART.c	使用 USCI 作为 UART
TI_MSPBoot_CI_PHYDL_eUSCI_UART.c	使用 eUSCI 作为 UART
TI_MSPBoot_CI_PHYDL_CC1101.c	使用 CC110x
MI	
TI_MSPBoot_MI_Flash_20Bit.c	用于对大型存储器模型器件中的应用程序闪存进行编程的 API
TI_MSPBoot_MI_FlashDualImg_20Bit.c	用于在大型存储器模型器件的闪存中实现双映像的 API
TI_MSPBoot_MI_Flash_16Bit.c	用于对小型存储器模型器件中的应用程序闪存进行编程的 API
TI_MSPBoot_MI_FlashDualImg_16Bit.c	用于在小型存储器模型器件的闪存中实现双映像的 API
应用管理器	
TI_MSPBoot_AppMgr.c	标准应用管理器
TI_MSPBoot_AppMgrDualImg_20Bit.c	在大型存储器模型器件中支持双映像的应用管理器
TI_MSPBoot_AppMgrDualImg_16Bit.c	在小型存储器模型器件中支持双映像的应用管理器

3.1 预定义的定制

软件包中包含 CCS 工程，该工程支持两个器件（MSP430F5529 和 MSP430G2553），这两个器件具有三个通信接口（UART、I²C 或带有 CC110x 的 SPI）并且每个器件有两个预定义的配置（单映像、双映像）。在提供的 CCS 示例中，器件和通信接口通过选择的工程分开，并可以在 Project → Build Configurations → Set Active 下选择预定义的配置。

4 构建 MSPBoot

此部分作为分步指南，介绍了如何为目标器件构建引导加载程序和演示应用程序。

节 4.2 介绍了如何构建和使用示例应用程序来运行演示。

4.1 启动新工程

软件包中包含以下内容：

- **Host_Examples**：这些示例显示了将 MSP-EXP430F5529 或 MSP-EXP430G2 用作主机，该主机通过 UART、I²C 或带有 CC110x 的 SPI 通信接口对 MSP430 基于闪存的目标进行编程。
- **MSP430F5529_Examples：**这些示例显示了将 MSP430F5529 用作 MSPBoot 目标。这包括引导加载程序代码和每个通信接口（UART、I²C 或带有 CC110x 的 SPI）的两个示例应用程序。
- **MSP430G2553_Examples：**这些示例显示了将 MSP430G2553 用作 MSPBoot 目标。这包括引导加载程序代码和每个通信接口（UART 或 I²C）的两个示例应用程序。
- **实用程序：**
  - **430 TXT 转换器：**这是一个 Perl 脚本，用于将 CCS 输出文件转换为 Host TargetApps。请参阅节 4.1.2.1。
  - **Project_Creator：**这是一个脚本，用于自动生成 CCS 工程脚本文件以用于创建新的 MSPBoot 工程，请参阅节 4.1.1.1。

该软件包是使用 CCS 7.2.0 进行构建和测试的。其他 IDE 版本和编译器不能直接支持所提供的资源，可能需要进行一些修改。

4.1.1 创建新的 MSPBoot 工程

虽然随附软件包中提供的示例在引入到 MSPBoot 时是一个很好的参考起点，但是您可能希望使用其他基于闪存的 MSP430 器件型号。随附软件包中包含一个工程生成脚本，该脚本生成一个 CCS 工程脚本文件，使用此文件可创建新的 MSPBoot 工程。创建后，需要通过用户输入修改几个文件，以确保新创建的源代码可用于所使用的 MSP430 器件型号（请参阅节 4.1.1.3）。

4.1.1.1 MSPBootProjectCreator.pl

MSPBootProjectCreator：生成开始在 CCS 中创建 MSPBoot 工程所需的所有引导加载程序和应用程序源文件。

备注

需要 Perl 解释器才能运行此脚本。如果需要，请访问 https://www.perl.org/ 来下载解释器。

Location：Utilities/Project Creator/MSPBootProjectCreator.pl

语法：

```
[] denotes an optional field
All numbers should be in hexadecimal format
MSPBootProjectCreator.pl
                    [-help]
                    -hdr <header_file>
                    -lnk_file <lnk_msp430.cmd>
                    [-boot_size <size>]
                    [-shared_vectors <number>]
                    [-dual_image]
                    [-I2C]
                    [-UART]
```

其中（所有数字均为十六进制值）：

- -hdr <header_file> = 指定 MSP430 器件型号的头文件。需要与 MSPBootProjectCreator.pl 位于同一目录中。该文件的默认路径是 C:\ti\ccsvx\ccs_base\msp430\include，其中 ccsvx 是所使用的 ccs 版本。
- -lnk_file <lnk_msp430.cmd> = 指定所用器件的默认链接器文件。需要与 MSPBootProjectCreator.pl 位于同一目录中。该文件的默认路径是 C:\ti\ccsvx\ccs_base\msp430\include，其中 ccsvx 是所使用的 ccs 版本。
- -boot_size <size> = 可选参数。指定引导加载程序区域的大小。仅允许 0x400 的增量。如果省略，则会使用默认的引导加载程序大小。
- -shared_vectors <number> = 可选参数。指定共享矢量的数量（十六进制）。如果未指定，则默认值为 3 个矢量。
- -dual_image = 可选参数。指定所创建文件应启用双映像引导加载程序。如果未指定，则认定为单映像。
- -I2C, -UART = 指定引导加载程序的通信接口。必须从二者中择一。

备注

使用生成器脚本时，不支持带有 CC110x 的 SPI，因为它具有许多特定于器件的依赖性。如果尝试开发的是 OAD 应用程序，请参阅随附软件包中提供的示例。

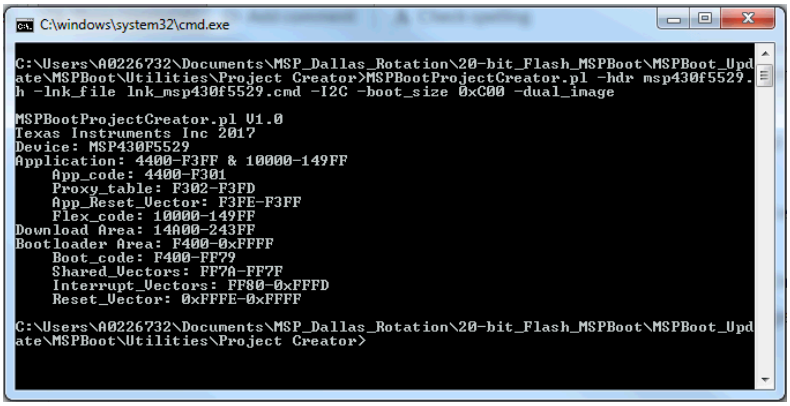


图 4-1. 示例命令

该脚本使用位于 Linker_Templates、Vector_Templates 和 ProjectSpec_Templates 文件夹中的模板。可以根据需要修改这些模板的内容，但在运行脚本时这些模板必不可少。该脚本还使用 Src 文件夹中的代码。同样，可以根据需要修改这些文件的内容，但是它们的名称和文件路径必须保持相同，才能确保脚本正常运行。

4.1.1.2 在 CCS 中导入工程规范文件

一旦执行了 MSPBootProjectCreator 并创建了工程规范文件，您将需要在 CCS 中使用该文件来创建工程。

1. 首先打开 CCS 并导航到 Project → Import CCS Projects...
   a. 浏览到由 MSPBootProjectCreator.pl 生成的工程规范文件
   b. 确定您将使用的应用程序模板并选择适当的工程
      i. App_Shared_Comm：显示如何与引导加载程序共享通信接口
      ii. App_Simple：不与引导加载程序共享通信接口
   c. 单击 Finish

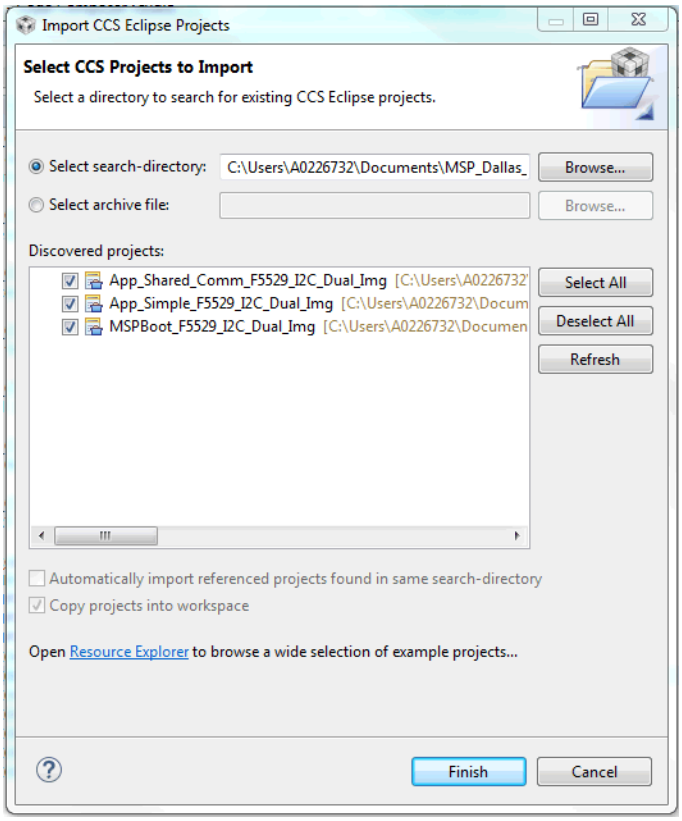


图 4-2. 在 CCS 中导入工程规范文件

2. 从每个工程中删除默认的链接器命令文件。该文件的标题为 lnk_<device variant>.cmd。
3. 在按照节 4.1.1.3 的要求修改了适当的源代码之后，您可以构建工程并在 MSP430 器件上对其进行编程。

4.1.1.3 修改生成的源代码

使用 MSPBootProjectCreator 时，有几个文件在创建后需要用户修改。这些文件包括：

- 在 MSPBoot 中：
  - main.c
  - TI_MSPBoot_Config.h
  - Comm/PHY_DataLink/TI_MSPBoot_CI_PHYDL_xxxx_xxx.c
  - AppMgr/TI_MSPBoot_AppMgr.c
- 在 Application(s) 中：
  - main.c
  - TI_MSPBoot_Mgr_Vectors_xxxx.c

这些文件中需要修改的部分用 //TODO: 标记加以标记，该标记易于搜索并在 CCS 中突出显示。每个 TODO 也会显示在 CCS 任务列表中，可通过 View → Other...→ General → Tasks 来访问任务该列表。每个任务都包含有关必须修改的内容以及在何处查找示例的说明。使用工程生成脚本时，必须进行此类修改才能确保 MSPBoot 和应用程序代码正确运行。

4.1.1.3.1 修改 MSPBoot Main.c

MSPBoot main.c 中有三个 TODO 项需要用户修改：

- 定义调试接口
  - 例如，调试接口可以是在引导加载程序中时点亮的 LED 或进入特定状态的 GPIO
- 初始化时钟系统
  - 默认情况下，MSPBoot 随附的示例将系统频率设置为 8MHz。MSPBoot 可以在 8MHz、4MHz 或 1MHz 下运行。有关更多详细信息，请参阅随附软件包和器件特定用户指南中提供的示例。
- 初始化硬件
  - 当器件进入引导加载程序代码时，未初始化为默认值的外设寄存器会影响其正常运行。例如，由应用程序设置为每秒中断一次的计时器将影响 MSPBoot 的运行。因此，TI 强烈建议将所有外设寄存器初始化为默认值。如果是在 BOR 之后进入的引导加载程序，则所有外设寄存器均已设置为默认值。

4.1.1.3.2 修改 TI_MSPBoot_Config.h

TI_MSPBoot_Config.h 中有四个 TODO 项需要用户修改：

- 定义 MCLK 频率
  - 修改 main.c 之后，代码的其他部分需要知道 MCLK 频率才能正常运行。
- 设置看门狗时间间隔
  - 在有些应用中，如果引导加载程序无响应，可能需要将器件复位。时间间隔长短的最大影响因素是所编程器件的闪存大小。闪存空间越多，时间间隔必须越长。例如，在包含 128KB 闪存的 MSP430F5529 中，时间间隔大于 6 秒。如果使用看门狗计时器，则 NDEBUG define 语句也必须取消注释。
- 设置硬件进入条件
  - 通常，引导加载程序提供了一种通过硬件条件进入代码的方法。在 MSPBoot 中，用户可以根据引脚的状态是高电平还是低电平来做决策。

- 取消配置选项的注释
  - 根据使用的通信接口，有许多不同的配置选项需要选择。您也可以在表 3-1 中找到这方面的更多信息。
  - 使用 I²C 时：
    - 取消 CONFIG_CI_PHYDL_START_CALLBACK 和 CONFIG_CI_PHYDL_I2C_SLAVE_ADDR 的注释
  - 使用 UART 时：
    - 取消 CONFIG_CI_PHYDL_UART_BAUDRATE 的注释
  - 使用 CC110x 时：
    - 取消 CONFIG_CI_PHYDL_CC1101_FREQUENCY 的注释

4.1.1.3.3 修改 TI_MSPBoot_CI_PHYDL_xxxx_xxx.c

TI_MSPBoot_CI_PHYDL_xxxx_xxx.c 中有三个 TODO 项需要用户修改：

- 定义通信模块
  - 根据器件和通信协议的不同，MSP430 中的多个外设可以实现相同的结果。例如，在 MSP430F5529 中，USCI_A0 和 USCI_A1 模块都可以执行 UART 通信。用户需要修改与此 TODO 关联的定义，以便根据应用要求使用适当的外设寄存器进行通信。
- 定义计时器模块（仅限 USCI 和 USI I²C）
  - 在 I²C 通信中，可以配置可选的超时，以利用各种 MSP430 MCU 中的计时器模块。用户需要修改与此 TODO 关联的定义，以便根据应用要求使用相应的计时器。
- 设置 GPIO 引脚
  - 根据通信接口和所使用的具体模块的不同，用户需要将 GPIO 设置为适当的外设功能。有关如何设置通信接口的引脚的更多信息，请参阅特定器件数据表。

4.1.1.3.4 修改 TI_MSPBoot_AppMgr.c

TI_MSPBoot_AppMgr.c 中有一个 TODO 项需要用户修改。在 TI_MSPBoot_Restore_DefaultClockSettings 函数中，建议用户将时钟设置设为默认值，就好像该器件刚刚复位一样。该函数在跳转到应用程序代码之前被调用，并将确保 MSPBoot 时钟设置不会被带到应用程序中。

4.1.1.3.5 修改 Application Main.c

根据使用的是简单应用程序模板还是共享通信应用程序模板，TODO 项的数量有所不同。但是，每个都包含 TODO 项，用于添加用户定义的值、函数及其原型以及中断服务例程。这些 TODO 项仅需要替换为相应的用户代码。每个模板还提供对函数 TI_MSPBoot_JumpToBoot 的访问，该函数允许应用程序代码跳转到引导加载程序。

4.1.1.3.6 修改 TI_MSPBoot_Mgr_Vectors_xxxx.c

TI_MSPBoot_VecRed_xxxx_App.c 中有两个 TODO 项需要用户修改：

- 添加 ISR 原型
  - 为了让矢量重定向能够正常工作，请为应用程序中使用的每个 ISR 添加外部原型。这些将在下一个 TODO 中使用。
- 更新代理矢量表
  - 要以应用程序中使用的 ISR 更新代理中断矢量表，请将“Dummy_ISR”文本替换为相应的 ISR 函数名称。每行旁边的注释将显示与 ISR 相关的外设。

4.1.2 使用 MSPBoot 加载应用程序代码

创建自定义应用程序以使用 MSPBoot 加载到 MSP430 时，请执行以下步骤以获得出色结果：

1. 在不使用 MSPBoot 的情况下开发应用程序。
   a. 此过程包括创建工程，使用默认的链接器文件，以及像不使用主存储器引导加载程序一样开发代码。
2. 开发应用程序后，将代码转移到应用程序模板之一。
   a. App_Simple：不与引导加载程序共享通信接口
   b. App_Shared_Comm：显示如何与引导加载程序共享通信接口
3. 如节 4.1.1.3.6 中所述修改矢量重定向文件。
4. 编辑工程属性以输出 TI-TXT 十六进制格式文件。
   a. Project Properties → MSP430 Hex Utility → Enable MSP430 Hex Utility
   b. Project Properties → MSP430 Hex Utility → Output Format Options → Output TI-TXT hex format (--ti_txt)
5. 构建工程。
6. 使用工程 Debug 文件夹中的 TI-TXT 文件生成可以从主机处理器加载的 C 文件（请参阅节 4.1.2.1）。
7. 将 MSPBoot 加载到目标器件上。
8. 如果目标器件尚未执行引导加载程序代码，必须强制目标器件进入引导加载程序。为此，可将应用程序代码设置为在收到特定命令时跳转到引导加载程序。请参阅提供的示例，以获取有关如何完成此操作的更多信息。
9. 将应用程序 C 文件加载到目标器件上。
   a. 如需更多信息，请参阅随附软件包中包含的示例主机工程。

4.1.2.1 转换应用程序输出映像

通过选择 Project Properties → MSP430 Hex Utility，可将 CCS 工程设置为生成 MSP430 .txt 格式或 Intel .hex 格式的输出，从而了解更多信息。这两种文件都不包含 CRC，但可以由主机处理器计算 CRC，或者需要手动将 CRC 添加到生成的文件中。无论如何计算 CRC，都必须将应用程序映像转换为可供主机处理器使用的格式。为了简化这一过程，软件包中包含 image2C，这是一个 Perl 脚本，用于将 MSP430 .txt 文件或 Intel .hex 文件转换为 C 数组。

Location：MSPBoot\Utilities\430 Image Converter\image2C.pl

语法：

```
[] denotes an optional field
image2C.pl
                [-help]
                -src <src_file>
                -dest <dest_file>
                -struct <array_name>
                [-20_bit]
```

- -src <src_file> = 指定 .txt 或 .hex 格式的源文件。
- -dst <dest_file> = 指定 .c 格式的目标文件。
- -struct <array_name> = C 文件中数组的名称。如果使用随附软件包中提供的主机示例，TI 建议将结构命名为 App1 或 App2。
- -20_bit = 可选参数。指定创建的文件与大型存储器模型（20 位）MSP430 器件兼容。如果使用大型存储器模型器件，则需要使用此命令才能正确生成文件。如果未指定，则认定为单映像。

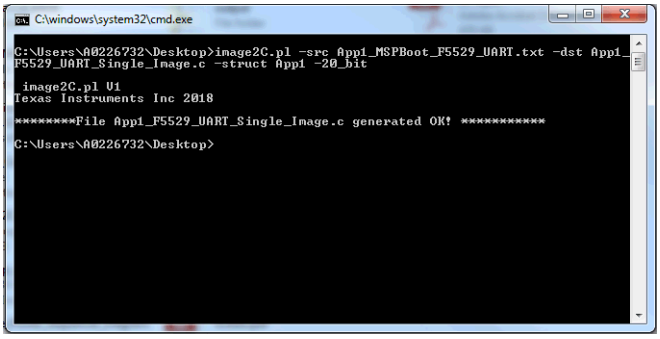


图 4-3. image2C 示例

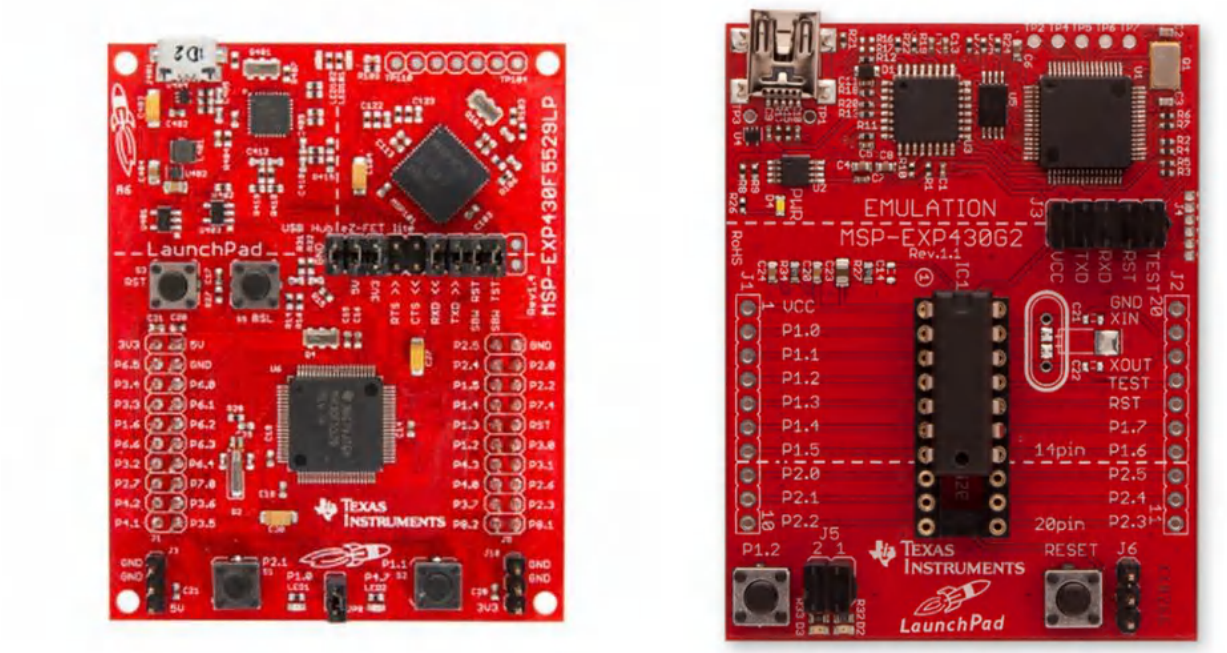
备注

需要 Perl 解释器才能运行此脚本。如果需要，请访问 https://www.perl.org/ 来下载解释器。

4.2 示例

该软件包中包含 MSP430G2553 和 MSP430F5529 的示例。可以使用任何 MSP430 板，但本应用报告中使用的示例是 LaunchPad™ 开发套件 MSP-EXP430G2 和 MSP-EXP430F5529。

4.2.1 LaunchPad 开发套件硬件



A. 在 MSP-EXP430G2 中，P2.0 默认未连接到 LED2。使用 I²C 通信时，可以添加外部连接以进行演示
B. 使用 I²C 通信时，必须在 MSP-EXP430G2 中拆除将 P1.6 连接到 LED2 的跳线 J5。

图 4-4. 目标板：MSP-EXP430F5529 和 MSP-EXP430G2

引导加载程序和演示应用程序在 LaunchPad 开发套件的所有变体中使用相同的 LED (LED1 和 LED2) 表示法。对于每种电路板衍生品，与这些 I/O 外设相对应的引脚分配都不同。对于每种用途，示例的设计都旨在使主机和目标 LaunchPad 开发套件可以是相同的派生品，尽管有示例，但可以根据需要针对不同的配置进行修改。

4.2.2 CC110x 硬件

可通过两个硬件选项将 CC110x 通信用于 MSPBoot 示例。第一个选项是 CC1101EMK868-915 和 BOOST-CCEMADAPTER 的组合，但是最简单的解决方案是使用 430BOOST-CC110L。图 4-5 显示了这两个选项。

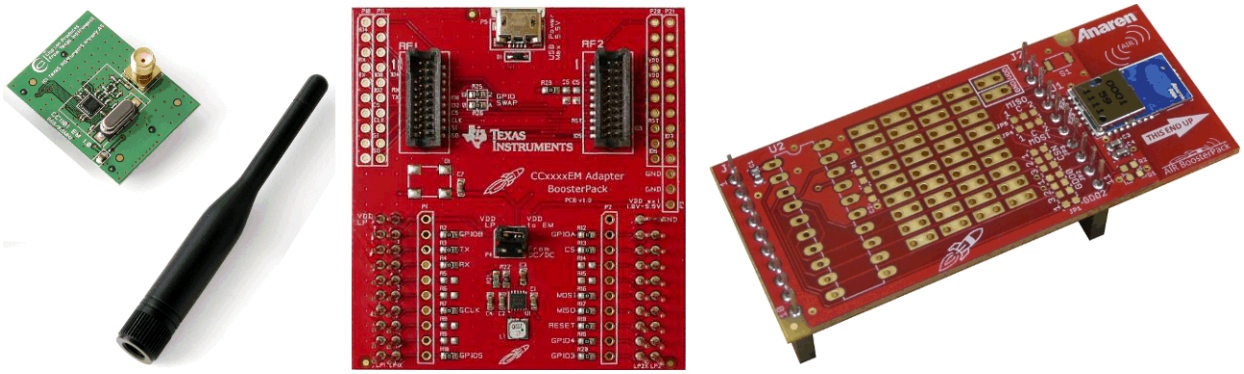


图 4-5. CC101EMK868-915、BOOST-CCEMADAPTER 和 430BOOST-CC10L

每个选项都需要两个单元，一个表示主机器件，另一个表示目标。两种解决方案在所有 LaunchPad 开发套件中都兼容，并采用直接连接方式，因此不需要其他硬件即可运行所提供的示例。有关 LaunchPad 开发套件生态系统以及 BoosterPack™ 插件模块的更多信息，请访问 TI LaunchPad 开发套件。

> **小心**
> 430BOOST-CC10L 和 CC101EMK868-915 使用不同的参考晶体来生成适当的频率。将 CC101EMK868-915 与 BOOST-CCEMADAPTER 一起使用时，必须将 hal_spi_ref_exp5529.h 中找到的参考晶体定义从 27000 修改为 26000，以适应从 430BOOST-CC10L 上 27MHz 晶体到 CC101EMK868-915 上 26MHz 晶体的变化。

4.2.3 构建目标工程

1. 选择一个目标处理器：MSP430F5529 或 MSP430G2553。
2. 选择通信接口：I^2C、UART 或带有 CC110x 的 SPI。
3. 打开 CCS 并选择或创建工作区。
4. 将 MSPBoot CCS 工程导入工作区中。这些工程位于 MSPBoot\<target>_Examples\<communication_interface>\
   a. 选中 Copy projects into workspace 复选框以确保您所处理的工程位于工作区中而不是 PC 上的其他地方。

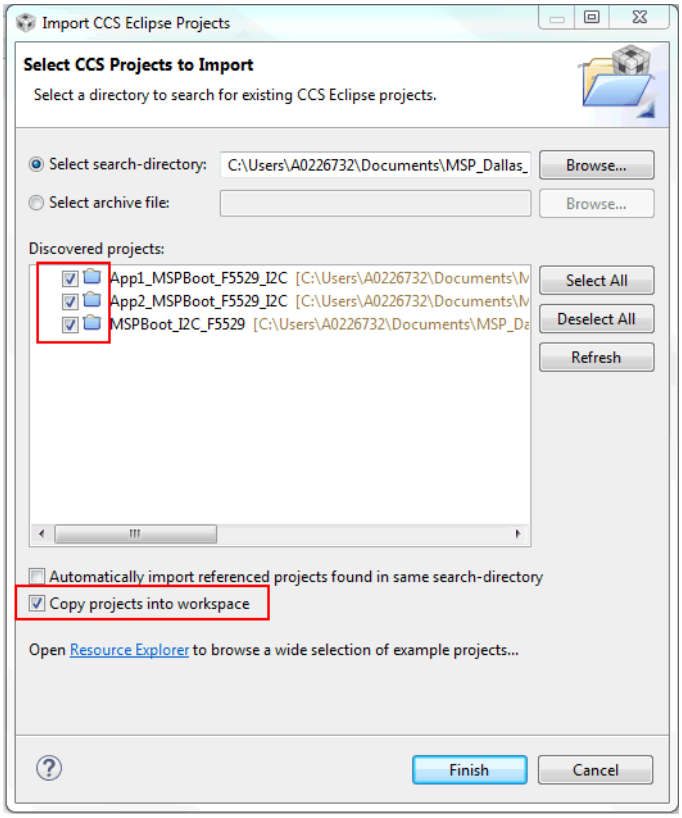


图 4-6. 导入 MSPBoot CCS 工程

5. 构建引导加载程序
   a. 选择 MSPBoot 工程
   b. 选择适当的目标配置（单映像或双映像）

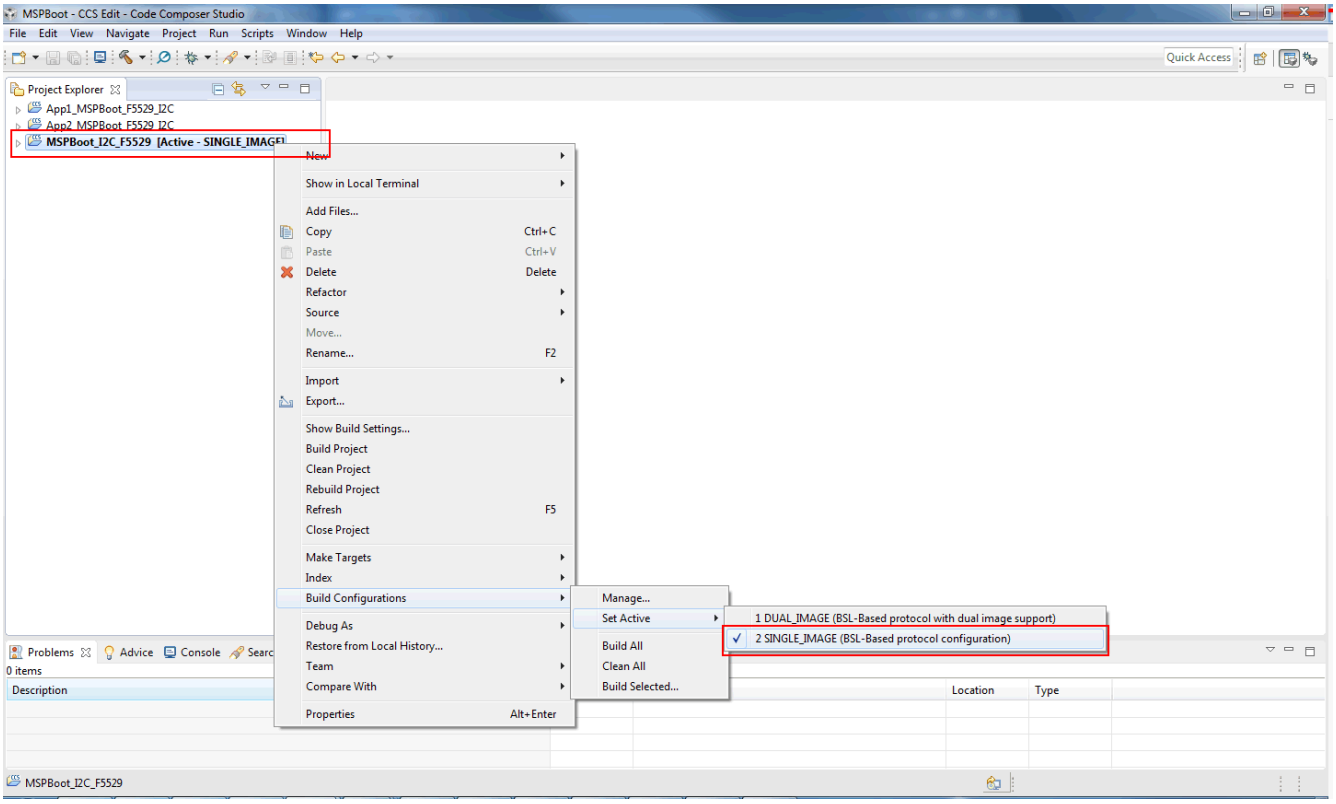


图 4-7. 选择目标配置

6. 构建  并下载  。仅应将目标 LaunchPad 开发套件连接到 PC
7. 构建两个应用程序。
   a. 选择 App1_MSPBoot 工程，然后选择与引导加载程序相同的配置。

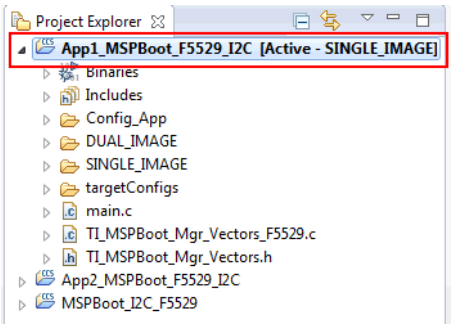


图 4-8. 选择 App1_MSPBoot 工程

8. 点击“Build”  工程。在此步骤之后会生成输出，但是将通过主机处理器转换并下载输出。节 4.1.2.1 说明了如何转换映像，而节 4.2.4 说明了如何使用主机演示来下载映像。
9. 对 App2_MSPBoot 重复步骤 6

4.2.4 构建主机工程

可按照以下步骤构建主机工程：

1. 将该工程导入 CCS。工程文件位于 MSPBoot\Host_Examples\Host_Examples_for_<target>_target\<communication interface>\。
2. 在 main.c 中找到位于几个地址（包括 CRC 地址）的定义上方的 TODO 项
   a. 更新相关的值以匹配目标器件链接器命令文件中定义的地址。
3. 将节 4.2.3 的步骤 8 生成的目标应用程序 C 文件添加到主机工程中排除的 TargetApps 文件夹
4. 在 main.c 中找到位于目标应用程序文件的定义上方的 TODO 项
   a. 更新这些文件的名称以匹配步骤 3 中引用的目标应用程序 C 文件。
5. 构建主机工程
   a. 选择主机工程
   b. 选择适当的目标配置（单映像或双映像）

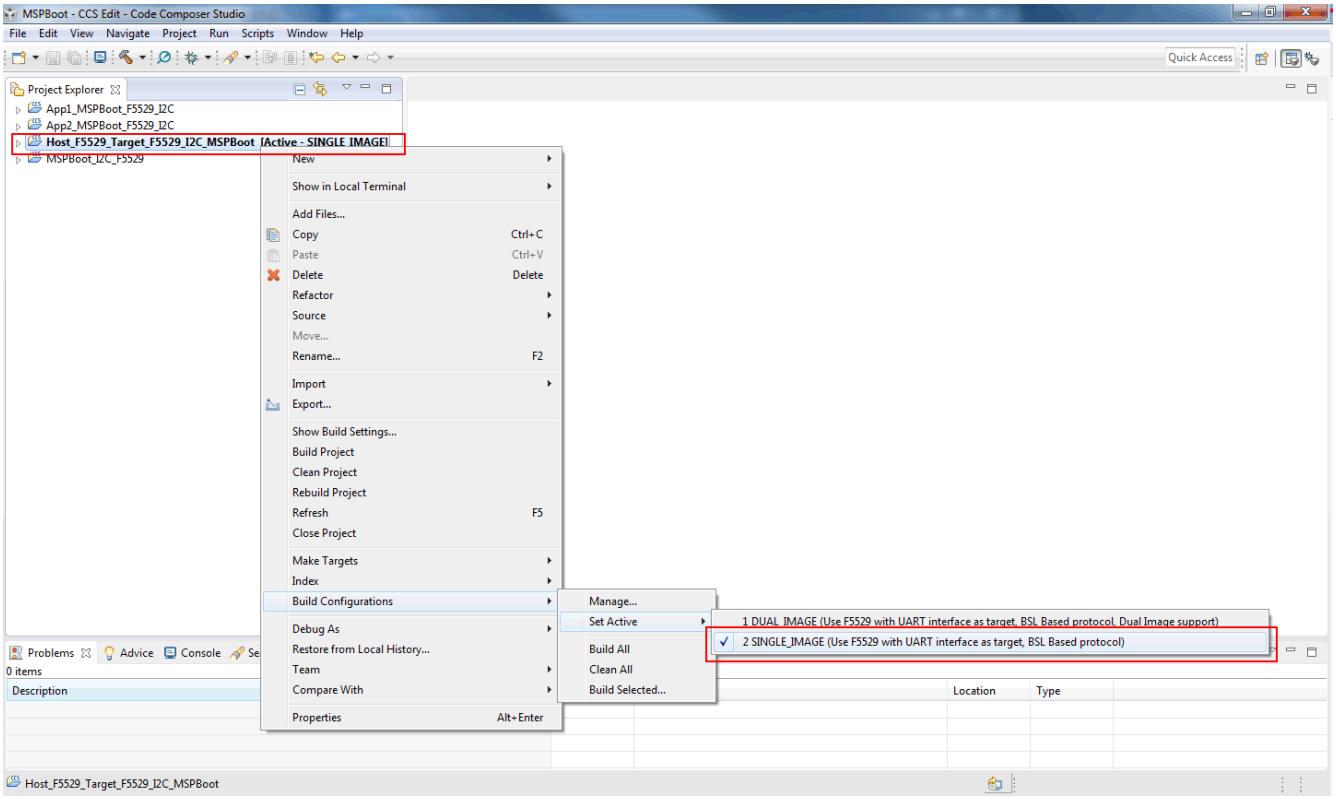


图 4-9. 在 CCS 中选择主机工程的目标

6. 构建  并下载  。仅应将主机 LaunchPad 开发套件连接到 PC。

该工程使用位于以下文件夹中的应用程序映像：

<Project_Dir>\Target_Apps

其中的 Project_Dir 是主机工程所在的目录。默认情况下，也应从主机构建中排除此文件夹。预构建的映像包含在内，但是可以按照节 4.2.3 和节 4.1.2.1 中所述的步骤替换或更新目标应用程序。

4.2.5 运行示例

主机项目将两个不同的映像发送到目标器件，并使用按钮进行用户交互。在两个 LaunchPad 开发套件上都不需要以 USB 连接到计算机即可运行演示；但是，每个套件都应通过 eZ-FET 由 USB 连接供电，或通过连接到 V_{CC} 和 GND 引脚的稳定 3.3V 外部电源供电（在这种情况下，请确保 eZ-FET 断开连接）。无论使用哪种通信类型或映像模型，都将按照以下步骤运行演示：

1. 如节 4.2.3 所述构建并下载 MSPBoot，并构建 App1 和 App2。
2. 根据节 4.1.2.1 转换 App1 和 App2。
3. 如节 4.2.4 所述构建并下载主机应用程序。
4. 根据所需的通信类型（I^2C、UART 或带有 CC110x 的 SPI）连接电路板。
   a. 每个主机项目都在代码的开头包含一个注释图表，其中描述了正确的连接。
5. 在两个器件中重置并执行代码。
6. 要进入目标引导加载程序模式（LED1 和 LED2 都保持亮起状态即表示处于该模式），请执行以下步骤：
   a. 如果目标没有有效的应用程序（默认），则目标将保持在引导加载程序模式下。
   b. 在硬件中，通过按住目标器件上的 S2 按钮，同时按住再释放复位按钮，即可强制进入引导加载程序模式。
   c. 如果是运行一个应用程序：
      i. 在目标器件上按 S2 按钮时，APP1 跳转到引导加载程序模式。
      ii. 当收到“强制引导”命令时（仅在共享 CI PHY-DL 时才受支持），APP2 会跳转到引导加载程序模式。
7. 按下主机板上的 S2 按钮。主机器件执行以下命令序列：
   a. 两次切换 LED1。
   b. 发送“强制引导”命令 (0xAA)。
      i. 如果目标器件已经处于引导加载程序模式，则会丢弃数据包，因为 CRC 不正确。
      ii. 如果目标正在运行 APP2，则目标器件将进入引导加载程序模式。
   c. 请求引导加载程序版本（发送 TX_VERSION 命令）。
      i. 如果目标响应为 0xA1（符合 BSL 协议预期），则主机继续。
      ii. 如果目标响应是任何其他值，则主机将中止事务。
   d. 擦除目标应用程序区域（发送 ERASE_APP 命令）。
   e. 发送 APP1（使用 RX_DATA_BLOCK 命令）。
   f. 设定 APP1 的 CRC（使用 RX_DATA_BLOCK 命令）。
   g. 强制目标应用程序运行（发送 JUMP2APP 命令）。
   h. 两次切换 LED1 表示传输成功，而保持 LED1 亮起表明主机已准备好发送 APP2。
8. 传输完成后，目标开始运行 APP1。
   a. 目标器件使 LED1 闪烁。
   b. LED1 会使用计时器定期闪烁。
   c. 按下目标板上的 S2 按钮进入引导加载程序模式。
9. 在目标处于引导加载程序模式下时，按下主机板上的 S2 按钮以发送 APP2。完成切换后，主机板的 LED1 保持熄灭，表明 APP1 已准备好发送。

10. 传输完成后，目标开始运行 APP2。
    a. 目标器件使 LED2 闪烁。
    b. 按下目标板上的 S2 按钮以切换 LED2。
    c. 由于 CI 已初始化，因此主机可以在新的传输序列开始时发送“强制引导”命令以在目标器件中强制进入引导加载程序模式。
11. 按下主机上的 S2 按钮以重新开始发送 APP1 的新序列。

双映像模式在传输完成后会在主机中暂停一小段时间，同时还会验证下载区域，将存储器转移到应用程序空间中，并在通过 CRC-CCITT 检查来验证了应用程序区域后擦除下载区域。

5 参考文献

1. *MSP430x2xx 系列用户指南*
2. *MSP430x5xx 和 MSP430x6xx 系列用户指南*
3. I^2C 规范 2.1
4. *MSP430™ 闪存器件引导加载程序 (BSL) 用户指南*
5. *创建基于闪存的自定义引导加载程序 (BSL)*

6 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 D (February 2018) to Revision E (January 2024)	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式	1
• 更改了用于下载软件包的链接	1

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