应用报告
ZHCA999D–October 2016–Revised March 2020

MSP430FRBoot - 适用于 MSP430™ FRAM 大型存储器型号器件的主存储器引导加载程序和无线更新

Ryan Brown, Katie Pier, and Gary Gao *MSP430 Apps*

摘要

本应用报告是 *MSPBoot – 适用于 MSP430 微控制器的主存储器引导加载程序*的扩展，它介绍了为 MSP430™FRAM 微控制器实现存储在主存储器中的引导加载程序的方法，此引导加载程序能够使用通用异步接收器/发送器 (UART) 通信或串行外设接口 (SPI) 总线和 CC110x 射频收发器来完成无线下载 (OAD)。此引导加载程序不仅实现了高度灵活性和模块化，还保持了很小的占用空间，因此是一种非常具有成本效益的解决方案，并支持大型存储器型号（存储器空间大于 16KB 的器件）。

适用于主从设备的软件包（包含示例和源代码）可从 http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP430FRBoot/latest/index_FDS.html 获取。5 节提供了如何运行示例的分步过程。

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 MSP430 FRAM 微控制器的受保护存储器 (ROM) 中。有关 BSL 的更多详细信息，请参阅*《MSP430™ FRAM 器件引导加载程序 (BSL) 用户指南》*。本应用报告中描述的配套资料可从下面的地址下载：http://www.ti.com/lit/zip/SLAA721。

内容

附图目录

附表目录

商标

1 简介

本文是对 *MSPBoot – 适用于 MSP430™ 微控制器的主存储器引导加载程序*背后原始理论的拓展研究。许多 FRAM 应用 都需要一种可轻松进行现场升级的解决方案。MSP430FRBoot 旨在通过用户定义的任何定制通信外设和进入顺序来完成此任务。本文通过两个不同的示例进一步演示这些功能。一个示例使用 UART 协议在设备之间创建简单的两线通信链接，而另一个示例使用 SPI 总线和两个 CC110x 器件来完成无线下载。最重要的是，这些解决方案可以在一个具有成本效益的设计中保持高性能、高集成度和超低功耗等特性。

MSP430 FRAM 器件配备了非常有用的 UART 引导加载程序 (BSL)，通过该 BSL 可以轻松进行现场升级。大多数 MSP430 FRAM 器件都有一个存储在 ROM 中的 BSL，该 BSL 支持 UART，且不能修改为支持 I^2C 或其他接口。MSP430FRxxxx1 器件实现了一个 I^2C BSL 解决方案，而不是 UART 解决方案。此外，BSL 不能包含应用程序可能需要的自定义进入顺序。有关 BSL 的更多详细信息，请参阅*《MSP430™ FRAM 器件引导加载程序 (BSL) 用户指南》*。

鉴于这些局限性，可能有必要创建一个存储在主存储器中并仍然能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具有以下特征的 MSP430FRBoot 引导加载程序：

- 占用空间小（所需大小小于 4KB）
- 支持20-bit地址，用于大型存储器型号
- 支持 FRAM 器件上提供的 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。
- 不同的校验选项支持可定制的稳健性级别
- 支持双映像，以避免通信异常造成的升级问题
- 允许在应用程序中使用所有中断

- 应用程序可以重复使用引导加载程序中的低级驱动程序，也可以实现自己的驱动程序。
- 可配置的进入引导程序方式
- 使用 CRC-CCITT 对应用程序进行可选的验证
- 提供源代码，允许进行其它定制

该引导加载程序随附源代码，其中包括不同工程配置、应用程序示例和主机程序示例，以便简化测试、定制和实现。本应用报告假定您已了解 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
MCU	微控制器
MI	MSPBoot 存储器接口
MSPBoot	*MSPBoot – 适用于 MSP430™ 闪存微控制器的主存储器引导加载程序*介绍的引导加载程序
MSP430FRBoot	本应用报告介绍的引导加载程序
OSI	开放系统互连
OAD	无线下载
SPI	串行外设接口
ROM	只读存储器
UART	通用异步收发器

1.2 约定

本文档包含的一些 UART 传输示例使用以下格式：

主机→目标
目标→主机
St：启动
Sp：停止

SPI 传输示例使用以下格式：

主设备→从设备
从设备→主设备
X：不用考虑

2 实现

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 1 显示了软件层。

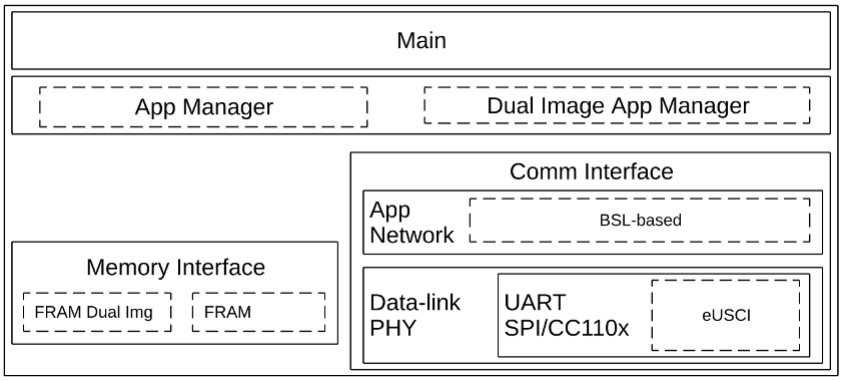

图 1. MSPBoot 软件架构

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 MSP430FRBoot 层
- 实现用于轮询通信接口和处理命令的主循环

图 2 显示了主例程的状态图

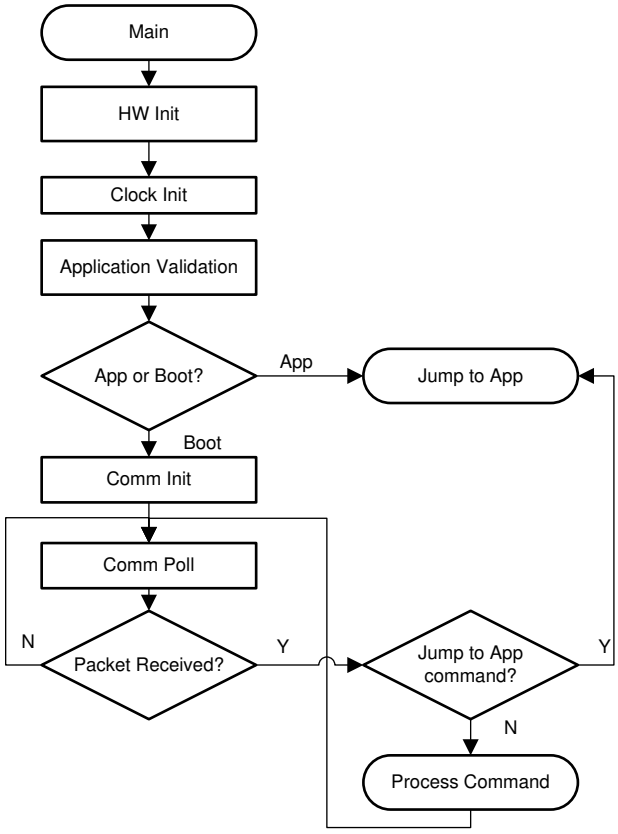

图 2. 主例程的流程图

2.2 应用程序管理器

应用程序管理器的主要功能是：

- 检测器件何时应处于引导加载程序模式与应用程序模式
- 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 引导加载程序和应用程序检测

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 3 显示了此决策过程的实现。

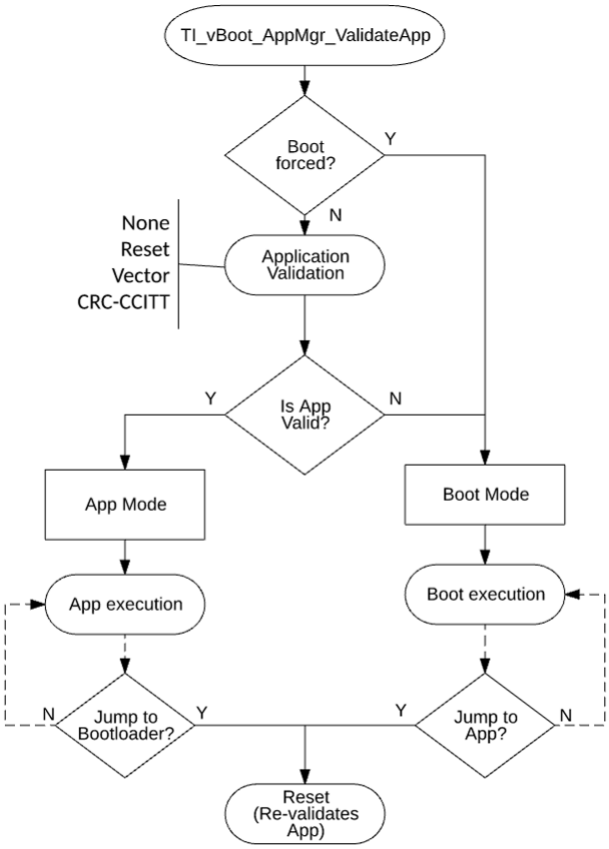

图 3. 应用管理器执行的应用程序验证

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：

  – MSP430FR5969 中的 P1.1（MSP-EXP430FR5969 中的 S2 按钮）
  – MSP430FR5994 中的 P5.5（MSP-EXP430FR5994 中的 S2 按钮）
  – MSP430FR2433 中的 P2.7（MSP-EXP430FR2433 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。

- 单映像模式（新应用程序将下载到应用程序区域）
  – 在此模式下可以使用两个选项（在 TI_MSPBoot_Config.h 中定义）：
    - Level_1：检查复位向量是否为空 (0xFFFF)
    - Level_2：在应用程序区域中执行 CRC 校验，并使用 CRC 结果与位于应用程序存储器起始处的 CRC 签名进行比较
- 双映像模式（新应用程序将下载到专用区域中，并在 CRC 校验后复制到应用程序区域中）

图 4 显示了双映像模式下验证过程的实现

图 **4.** 双映像应用程序验证

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。

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。
    1. 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。
    2. 有效则继续。
2. 将应用区域程序替换为下载区域程序。
3. 验证应用程序区域中的映像。
    1. 如果有效，则擦除下载区域。复位将执行应用程序，因为应用程序区域中的映像有效。
    2. 否则请退出。这是意外状态，但复位将再次验证两个映像。

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。

FRAM 器件使用软件 BOR 强制进行复位，这是将 MSP430 MCU 恢复到默认状态的有效方法。默认情况下，声明 HW_RESET_BOR 处于启用状态。

2.2.2 存储器分配

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 MSP430 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 MSP430 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 FRAM 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 5）。

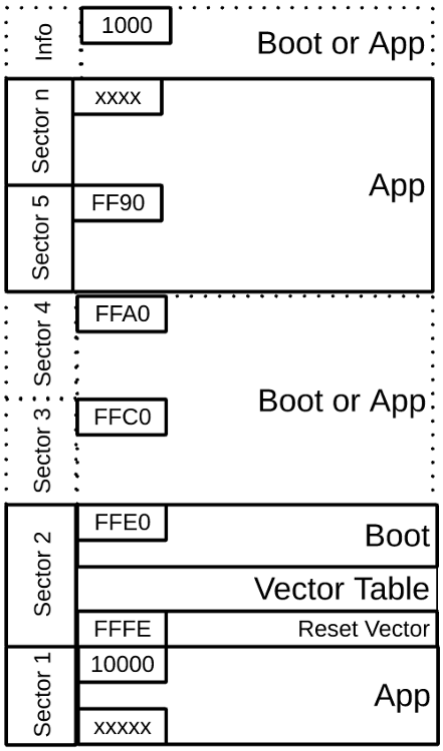

图 5. 存储器分配

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。应用程序可以使用额外的 20 位空间（0x10000 及更高）。

2.2.3 FRAM 器件中的中断矢量

FRAM 没有最小擦除大小的限制，因此可以在 FRAM 器件中对所有中断进行重新编程，而不会有擦除复位矢量的风险。默认情况下，MSP430FRBoot 使用 MPU 来启用引导加载程序区域的保护功能，但在重新编程中断矢量时会禁用此功能。

某些 MSP430 MCU 支持将矢量重定向到硬件中的 RAM (SYSRIVECT)，这可能是一个很好的选择，对具有足够 RAM 的器件来说，尤其如此。这也允许使用 MPU 模块对引导加载程序进行全面保护。

2.3 存储器接口 (MI)

为了保护引导加载程序区域，MSP430 MCU 在逻辑上分为以下几个区段：

- 应用程序区域：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：这是包含用户应用程序和已重定向矢量表的不可写区段
- 下载区域：这是包含用户应用程序和已重定向矢量表的可写区段（仅双映像模式）

每个区段的大小在项目链接器文件中定义。在 Code Composer Studio™IDE (CCS) 的示例项目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 FRAM 器件，这种存储器保护的实现方式如下：

- FRAM 不需要擦除，但是当执行擦除以计算有效的 CRC 时，应用程序存储器将写入 0xFF。
- 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。
- MPU 保护引导加载程序区域。用户可以根据应用程序修改 MPU 设置，但 TI 建议始终保护引导加载程序区域。
- 仅在更新节 2.2.3中所述的中断矢量时，才禁用 MPU 保护。

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。引导加载程序区域使用 MPU 进行硬件保护。

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区段，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。
  - 应用程序区域：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 2.2.1.2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区段。

每个扇区的大小在项目链接器文件中定义。CCS 的示例项目中提供了显示不同存储器大小的示例。

2.4 通信接口 (CI)

CI 的用途包括：

- 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是使用 UART 或 SPI 实现的，并且支持 eUSCI，但是如果需要，可以包括其他选项。通过使用表 1 中的回调函数提供指向结构的指针来初始化 PHY-DL 层。

表 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（例如，超时）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 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 6）。

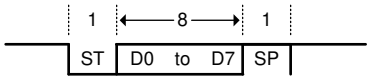

图 6. UART 8-N-1 格式

默认波特率定义为 CONFIG_CI_PHYDL_UART_BAUDRATE = 57600。

2.4.1.2 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 7）：

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

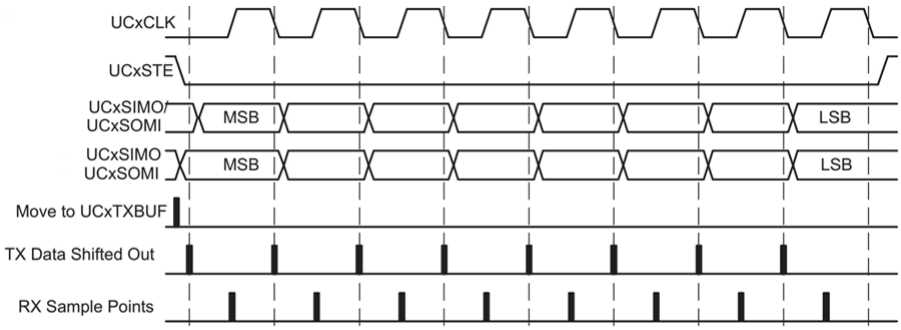

图 7. SPI 格式

2.4.1.3 CC110x

CC110x 器件使用表 2 中所示的数据包结构发送数据。

表 2. CC110x 数据包结构

帧头	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），这些区段不包括在数据包中。

MSP430FRBoot 中 CC110x 的配置如下：

- 数据传输速度为 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项目。请参阅 *CC1101 低功耗低于 1GHz 射频收发器*数据表，了解有关常见 CC110x 命令的更多信息以及其他通信详细信息。

此数据包结构与基于 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也提到了这一点。

2.4.1.4 通信共享

用户应用程序可以根据需要使用通信接口（UART、GPIO 或其他接口），因为当微控制器跳转到应用程序时，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 3 中的函数指针。

表 3. Boot2App_Vector_Table 定义

函数指针	说明
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	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	用于写入 TX 缓冲区的函数

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。软件包中包含了演示如何共享 CI PHY-DL 的应用程序示例。

2.4.2 NWK-APP

利用 CI Network-Application 层，可实现通信协议，解读来自 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，MSP430FRBoot 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 FRAM 器件引导加载程序 (BSL) 用户指南》对 BSL 进行了详细说明。

在 MSP430FRBoot 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 4 中所示的格式。

表 4. 基于 BSL 的协议命令格式

帧头	长度	有效载荷	Checksum[L]	Checksum[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）。

帧头：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、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 5 中的命令会作为有效载荷来实现。

表 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（FRAM 中为 512B）。

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）。

来自目标的每个响应始终是一个字节。表 6 列出了有效值。

表 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 中列出的传入 CC110x 数据包与表 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 UART 或 CC110x 的示例

将 UART 与基于 BSL 的协议一起使用时，适用以下注意事项：

- 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 2.4.1.1 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：主机擦除微控制器应用程序区域。

0x80	0x01	0x15	0x64	0xA3
帧头	长度	ERASE_APP	Checksum_L	Checksum_H

目标器件处理命令，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 MSP430FRBoot

MSPBoot 的设计重点是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，或者增加新的功能。根据应用要求，已将不同级别的定制添加到 MSP430FRBoot 代码中，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或禁用预处理程序定义来选择这些选项。表 7 列出了可以在 TI_MSPBoot_Config.h 中配置的选项。

表 7. 可选配置

值	说明	
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。	
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。	
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	通过检查其复位矢量来验证应用程序。	
2	通过检查 CRC_CCITT 来验证应用程序。	
CONFIG_CI_PHYDL_COMM_SHARED		
已定义	通信接口 PHY-DL 层与应用程序共享。	
未定义	CI PHY-DL 不与应用程序共享。	
CONFIG_CI_PHYDL_TIMEOUT		
已定义	在 CI PHY-DL 中检测超时。	
未定义	CI PHY-DL 不检测超时。	
CONFIG_CI_PHYDL_ERROR_CALLBACK		
已定义	在检测到超时错误时，将调用回调函数。	
未定义	在检测到超时时，不调用回调函数。	

如果要选择其他定制方案，可通过在项目中添加和使用适当的文件来实现。表 8 列出了项目中可互换的文件。

表 **8.** 定制文件

文件	说明
CI PHY-DL	
TI_MSPBoot_CI_PHYDL_USCI_UART.c	使用 eUSCI 作为 UART
TI_MSPBoot_CI_PHYDL_CC1101.c	使用 CC110x
MI	
TI_MSPBoot_MI_FRAM.c	用于对应用程序 FRAM 进行编程的 API
TI_MSPBoot_MI_FRAMDualImg.c	在 FRAM 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.c	可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 Code Composer Studio IDE，它包含三个器件（MSP430FR5969、MSP430FR5994 和 MSP430FR2433），这三个器件具有两个通信接口（UART 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项目分开，并可以在 Project → Build Configurations → Set Active 下选择预定义的配置。

4 构建 MSPBoot

此部分提供了分步指南，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。5 节介绍如何构建和使用主机 应用程序 来运行演示。

4.1 *LaunchPad™ 开发套件硬件*

该软件包在其 LaunchPad™开发套件上分别提供了 MSP430FR5969、MSP430FR5994 和 MSP430FR2433 的示例（分别为 MSP-EXP430FR5969、MSPEXP430FR5994 和 MSP-EXP430FR2433）（参见图 8）。

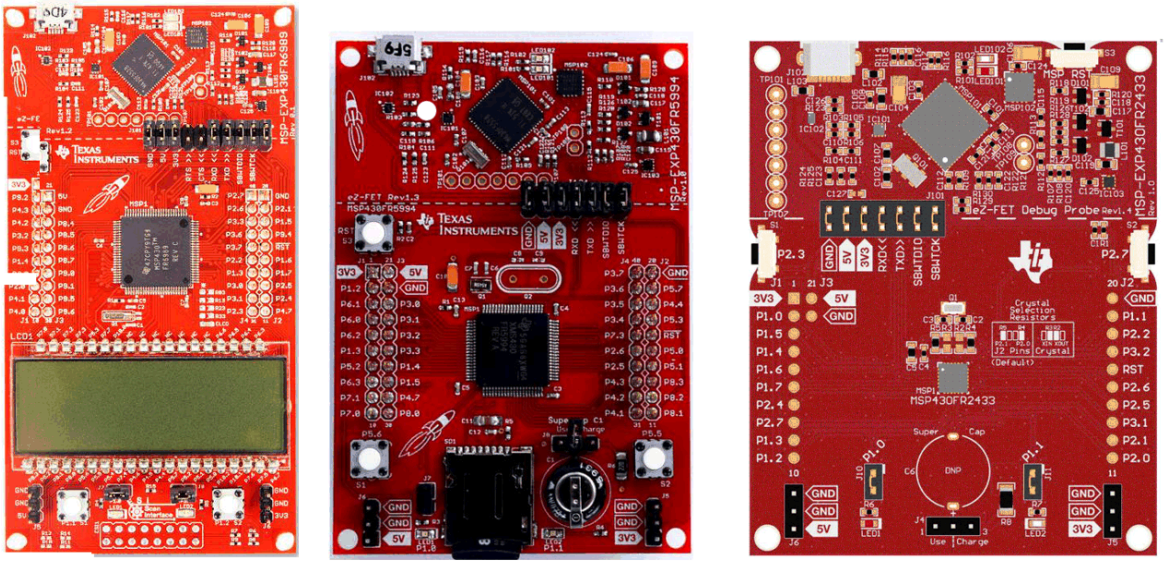

图 8. MSP-EXP430FR5969、MSP-EXP430FR5994 和 MSP-EXP430FR2433

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）和按钮（S1 和 S2）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。为便于使用，示例的设计都旨在使主机和目标 LaunchPad 开发套件应为相同的派生品，尽管可以根据需要针对不同的配置进行修改。

4.2 *CC110x 硬件*

可通过两个硬件选项将 CC110x 通信用于 MSP430FRBoot 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 9 显示了这两个选项。

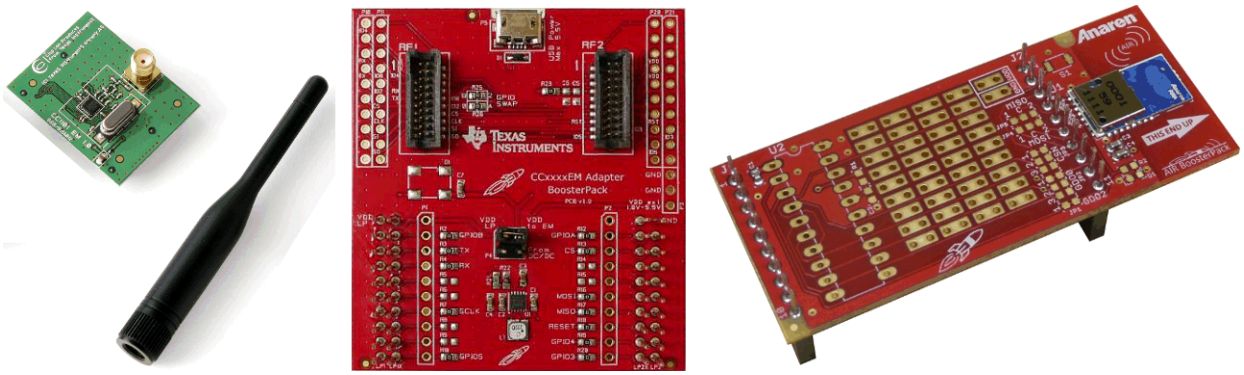

图 9. CC1101EMK868-915、BOOST-CCEMADAPTER 和 430BOOST-CC110L

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 工具页面。

4.3 软件

软件包中包含以下内容文件夹：

- 目标：目标引导加载程序和演示 应用程序。
  - **FR5969_UART、FR5969_CC1101、FR5994_UART、FR5994_CC1101、FR2433_UART、FR2433_CC1101**：支持具有指定通信的适当 FRAM 派生品的项目。
    - **CCS**：CCS 项目文件。
      - **MSPBoot**：引导加载程序的 CCS 项目文件。
        - Config：引导加载程序的 CCS 链接器文件。
      - **App1_MSPBoot**：应用程序示例 1 的 CCS 项目文件。
        - Config：App1 的 CCS 链接器文件。
      - **App2_MSPBoot**：应用程序示例 2 的 CCS 项目文件。
        - Config：App2 的 CCS 链接器文件。
    - **Src**：源代码。
      - **MSPBoot**：引导加载程序的源代码。
        - **AppMgr**：应用管理器源代码文件。
        - **Comm**：CI 源代码文件。
        - **MI**：MI 源代码文件。
      - **App1**：应用程序示例 1 的源代码。
      - **App2**：应用程序示例 2 的源代码。
- 主机：主机演示应用程序。
  - **MSP-EXP430FR5969、MSP-EXP430FR5994、MSP-EXP430FR2433**：支持相应的 LaunchPad 开发套件的主机项目（请参见节 4.3.2）。
    - CCS：CCS 项目文件。
    - Src：源代码。
      - TargetApps：转换后的目标应用程序示例。

- **430txt_converter**：用于将 CCS 输出文件转换为主机 TargetApps 的脚本和 应用程序 。有关详细信息，请参阅节 4.3.2。
- **linkerGen**：为特定器件派生品创建自定义命令链接器文件。有关详细信息，请参阅节 4.3.3。

4.3.1 构建目标软件

1. 选择一个目标处理器：MSP430FR5969、MSP430FR5994 或 MSP430FR2433。
2. 打开 CCS 并选择或创建工作区。
3. 将 MSPBoot CCS 项目导入工作区中。这些项目位于 MSP430FRBoot_<version>\Target_CCS\<target>\CCS\

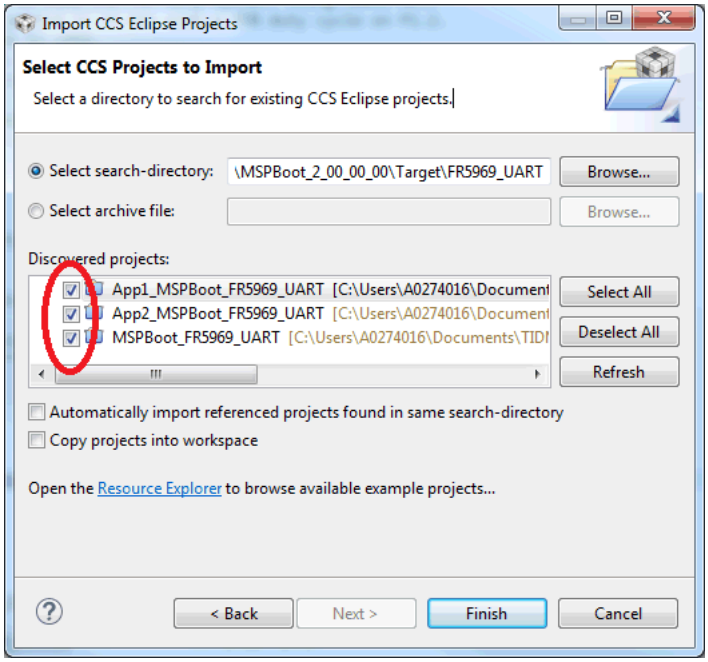

图 **10.** 导入 **MSPBoot CCS** 项目

4. 构建引导加载程序。
   a. 选择 MSPBoot 项目。
   b. 根据3.1 节，选择正确的目标配置。

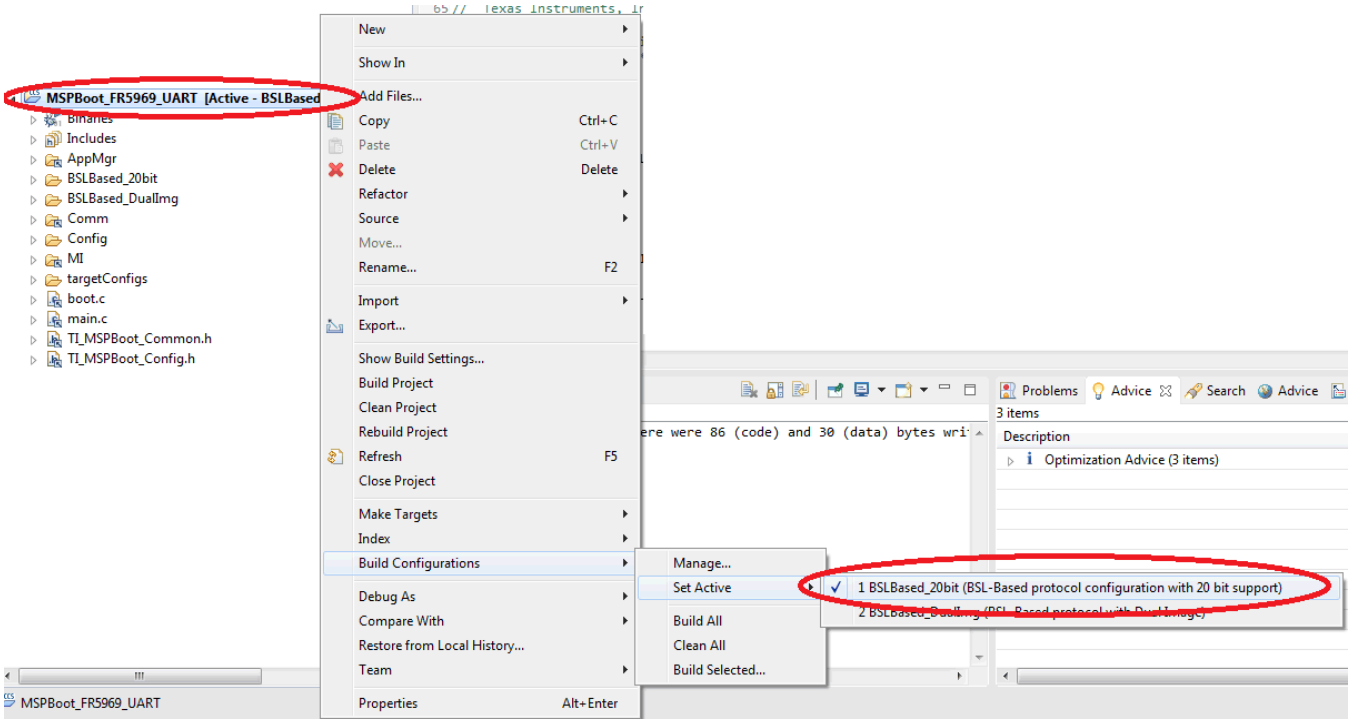

图 **11.** 选择目标配置

c. 构建  并下载  。仅应将目标 LaunchPad 开发套件连接到 PC。

5. 构建这两个 应用程序。

a. 选择 App1_MSPBoot 项目，然后选择与引导加载程序相同的配置：

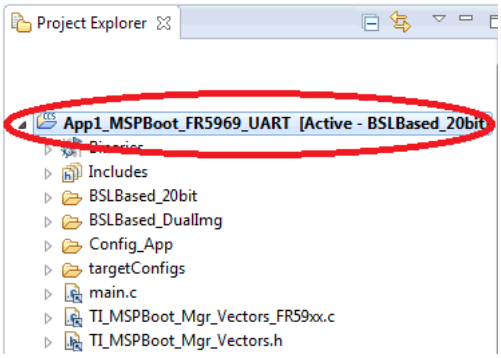

图 **12.** 选择 **App1_MSPBoot** 项目

b. 单击 Build  项目。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 4.3.2 说明了如何转换映像，而5 节说明了如何使用主机演示来下载映像。

c. 对 App2_MSPBoot 重复步骤 5。

4.3.2 转换应用程序输出映像

在 App1 或 App2 项目中找到 output.txt 映像。

该 .txt 文件不包含 CRC，需要将其转换为主机项目可以使用的格式。为了简化这一过程，软件包中包含 430txt2C，这是一个 Python 脚本，用于将 MSP430 .txt 文件转换为 C 数组：

- 在下面的文件夹中找到转换工具：“MSP430FRBoot_<version>\Utilities\430txt_converter”。
- Readme.txt - 显示如何使用转换工具
- demo_test.bat - 需要根据您的用例进行修改，并调用 430txt2c.py
- 430txt2c.py - 根据 demo_test.bat 中的命令将 .txt 文件转换为 .c 文件

注: 使用此工具之前，请先安装 Python 3。

4.3.3 生成链接器文件

CCS 和 IAR 的链接器文件包含在所有目标配置中，它们可以用作其他器件或自定义项目的起点。MSP430FRBoot 包含一个链接器生成器脚本，该脚本也可以帮助完成此过程。

- 在下面的文件夹中找到转换工具：“MSP430FRBoot_<version>\Utilities\linkerGen”。
- Readme.txt - 显示如何使用转换工具
- demo_link.bat - 需要根据您的用例进行修改，并调用“MSPFRBOOT_Linkergen_v2.py”
- MSPFRBOOT_Linkergen_v2.py - 基于 demo-link.bat 中的命令生成链接器文件

注: 使用此工具之前，请先安装 Python 3。

5 将 FRAM LaunchPad 开发套件用作主机的演示

该软件包包括在 MSP-EXP430FR5969、MSP-EXP430FR5994 和 MSP-EXP430FR2433 LaunchPad 开发套件上运行的主机设备的项目和源代码。每个套件都支持自己的 MSPBoot 协议和所有目标派生品。例如，MSP-EXP430FR5969 支持 MSP430FR5969 MSPBoot 目标以单映像或双映像模式等进行 UART 或 CC110x 通信。可以对其进行定制，以便可以使用任何 LaunchPad 开发套件主机对带有适当 MSP430FRBoot 固件的任何目标 MSP 派生品进行编程。

5.1 硬件

从前面介绍的选项来看，这个演示使用 FRAM LaunchPad 开发包连接到相同派生品的目标。对于 UART 通信，除了接地以外，还需要连接 eUSCI TXD 和 RXD 线。确保主机 TXD 线连接到目标 RXD 线，主机 RXD 线连接到目标 TXD 线。

使用 CC110x 通信时，目标设备和主机设备之间不需要布线。只需将带有 BOOST-CCEMADAPTER 或 430BOOST-CC110L 的 CC1101EMK868-915 连接到相应的 LaunchPad 开发套件或器件引脚即可完成设置。确保 BoosterPack 插件模块在 LaunchPad 开发工具包板上的安装方向正确。

表 9 列出了用于每个 MSP 器件和通信类型的特定 eUSCI 外设。

表 9. eUSCI 外设连接

CI	引脚	MSP 派生品			
		MSP430FR5969	MSP430FR5994	MSP430FR2433	
UART	RXD	P2.6/UCA1RXD	P6.1/UCA3RXD	P1.5/UCA0RXD	
	TXD	P2.5/UCA1TXD	P6.0/UCA3TXD	P1.4/UCA0TXD	
	GND				
SPI (CC110x)	MISO	P1.7/UCB0SOMI	P5.1/UCB1SOMI	P2.5/UCA1SOMI	
	MOSI	P1.6/UCB0SIMO	P5.0/UCB1SIMO	P2.6/UCA1SIMO	
	CLK	P2.2/UCB0CLK	P5.2/UCB1CLK	P2.4/UCA1CLK	
	SS	P3.0	P4.4	P2.2	

5.2 构建主机项目

可以按照以下步骤构建主机项目：

1. 将项目导入 CCS。项目文件位于 MSPBoot\Host\ <host>\CCS 中，其中 <host> 是 LaunchPad 开发工具包（MSP-EXP430FR5969、MSP-EXP430FR5994 或 MSP-EXP430FR2433）的变体。
2. 选择目标派生品。这可以使用 CCS 中的不同目标配置进行选择（请参见图 13）

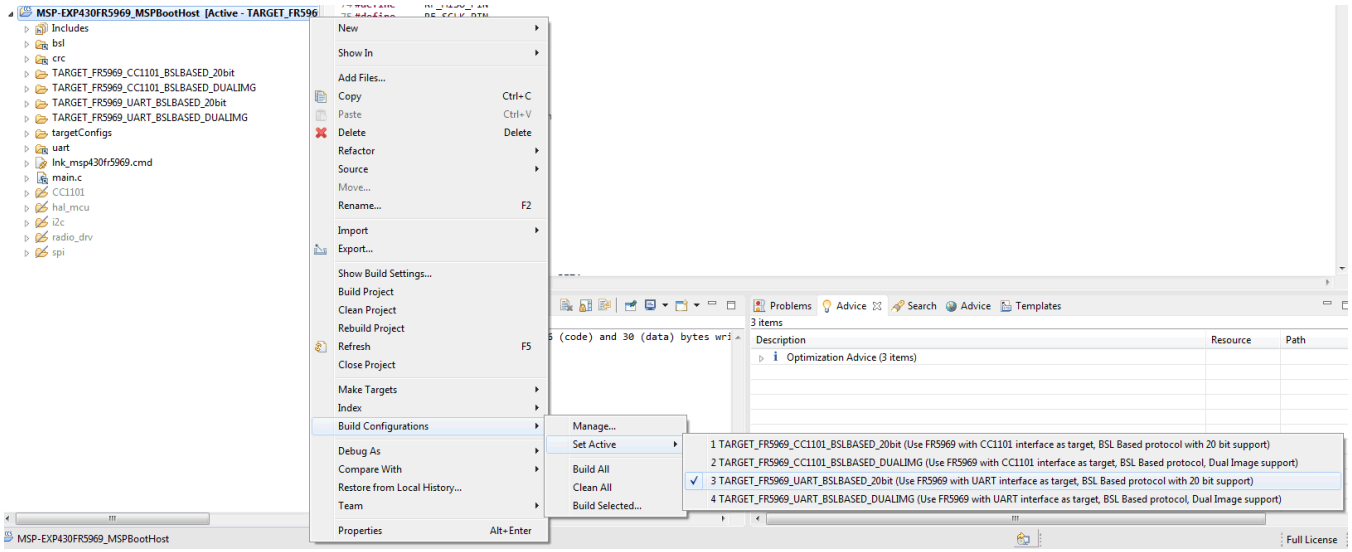

图 **13.** 在 **CCS** 中选择主机项目的目标

3. 构建并下载应用程序。此时应将主机 LaunchPad 开发套件连接到 PC。

预构建的映像包含 在内， 但是可以按照节 4.3.1和节 4.3.2中所述的步骤替换或更新目标应用程序。

5.3 运行演示

主机 LaunchPad 开发套件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）。由于主机和目标 LaunchPad 开发工具包都是相同的硬件，因此对每个板进行相应的标记可能有助于避免混淆。无论使用哪种通信类型或映像模式，都将按照以下步骤运行演示：

1. 如节 4.3.1所述构建并下载 MSPBoot，并构建 App1 和 App2。
2. 根据节 4.3.2，转换 App1 和 App2。

注： 批处理文件 PrepareCCSOutput_[FR derivative].bat 显示了如何转换为 C 数组并复制输出文件。在此主机实现中，MSP430 MCU 保留了不带 CRC 的目标映像，因此它是在假设未编程使用的位置为 0xFF 时计算 CRC 值的。

---

3. 如5.2 节所述，构建并下载主机应用程序。
4. 根据所需通信类型（5.1 节所述的 UART 或4.2 节所述的 CC110x 解决方案之一）连接板。
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 S1 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 0xA0（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 S1 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 检查验证应用程序区域后擦除下载区域。

注： 由于 MSP-EXP430FR2433 上的引脚冲突，请通过移除跳线，从 P1.1 断开 LED2 的连接，然后物理连接至 P1.2。

6 将目标端示例项目移植到其他 MSP430FR 器件

本节列出了将目标端示例项目移植到其他 MSP430FR 器件的步骤示例。在 IAR 中使用 MSP430FR2355，并使用 UART 单映像模式。举例而言：

1. 选择 FR2433_UART_Single 目标演示项目作为模板。这些演示项目位于“MSP430FRBoot_<version>\Target_IAR\MSP430FR2433_TARGET\FR2433_UART_Single”。
2. 将此项目和名为“FR2433_Linker_files”的链接器文件夹复制到另一个文件夹，如图 14 所示。对于 MSP430fr2xx4xx 系列器件，请选择 MSP430FR2433 目标演示项目作为模板。对于 MSP430FR5xx6xx 系列器件，请选择 MSP430FR5969 目标演示项目作为模板。

图 **14.** 将 **UART** 单个项目和链接器文件夹复制到其他文件夹

3. 将工作区名称从“MSP430FR2433_UART_Single”更改为“MSP430FR2355_UART_Single”（基于您的器件）。

图 **15.** 更改工作区名称

a. 该工作区包括以下三个项目：
   i. “App1_UART_Single”使用按钮从应用程序跳转到引导代码。
   ii. “App2_UART_Single”使用 UART 接收跳转命令，然后从应用程序跳转到引导代码。
      A. 根据您的应用程序需求选择一种解决方案。
   iii. “Boot_UART_Single”是引导加载程序代码。

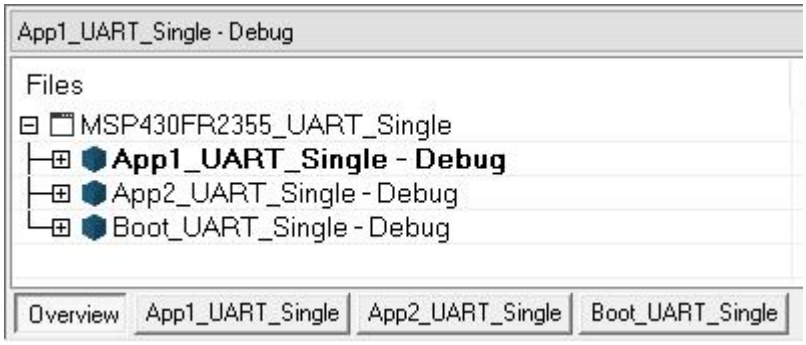

图 **16.** 工作区中的项目

注: 不同的应用程序案例可能使用不同的外设，例如 PGIO 或 UART。

4. 打开工作区。
5. 选择引导项目“Boot_UART_Single”。
6. 在整个项目中搜索“//Need to change”以查找需要修改的文件，如下所示。

图 **17.** 在引导项目中查找结果

7. 打开需要更改的文件。
8. 根据您的应用程序，在每个文件中搜索“//Simple change”，以查找需要修改的区域。图 18 显示了使用“main.c”进行的搜索。您需要根据应用程序更改寄存器和 GPIO。（有关如何进行更改的更多详细信息，可以比较 MSP430FR2433、MSP430FR5969 和 MSP430FR5994 之间的项目差异）

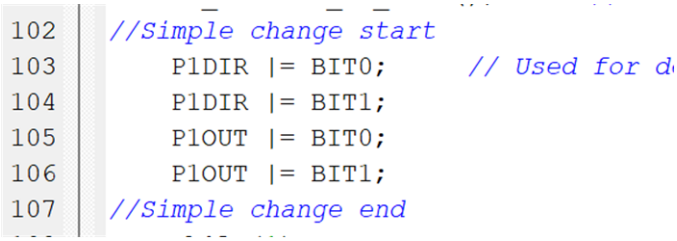

图 **18.** 查找简单更改的结果以更改 **GPIO**

9. 在“App1_UART_Single”或“App2_UART_Single”项目中使用相同的方法进行修改。
10. 打开文件夹“Utilities\linkerGen”。
11. 阅读文件“Readme.txt”以了解如何生成链接器文件以及如何基于您的器件生成链接器文件。

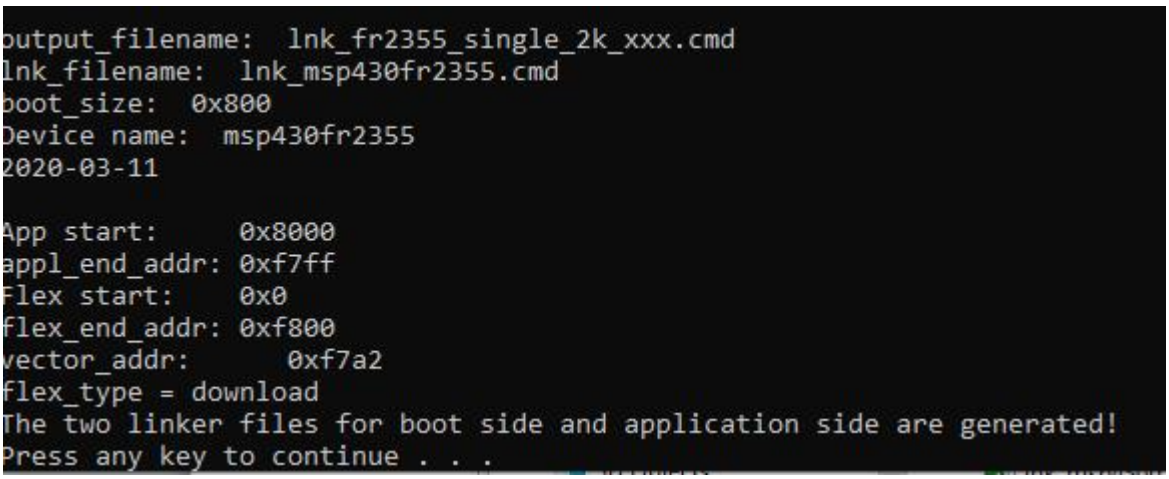

图 **19.** 成功生成链接器文件

12. 删除 MSP430FR2433 的链接器文件，并将新生成的链接器文件复制到名为“FR2433 链接器文件”的文件夹中

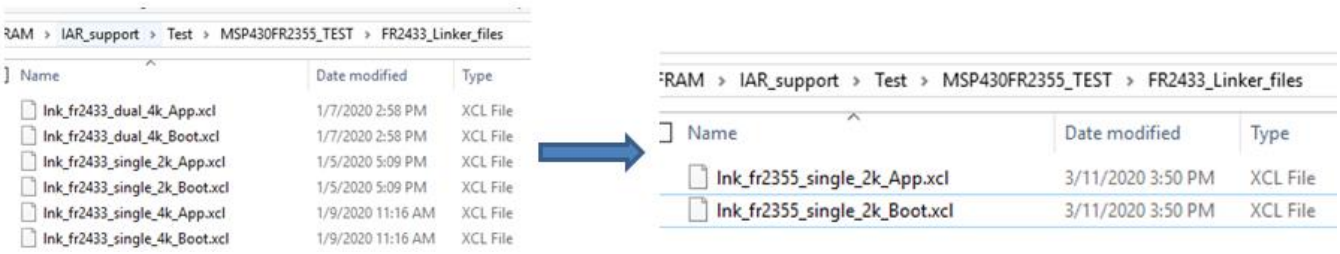

图 **20.** 将链接器文件移到链接器文件夹

13. 打开引导项目“Boot_UART_Single”选项。
14. 将器件更改为 MSP430fr2355（基于您的设备）。
15. 将“__MSP430FR2433__”的定义更改为“__MSP430FR2355__”。
16. 更改新建链接器文件名。
17. 对应用程序的项目执行步骤 14 至 16。

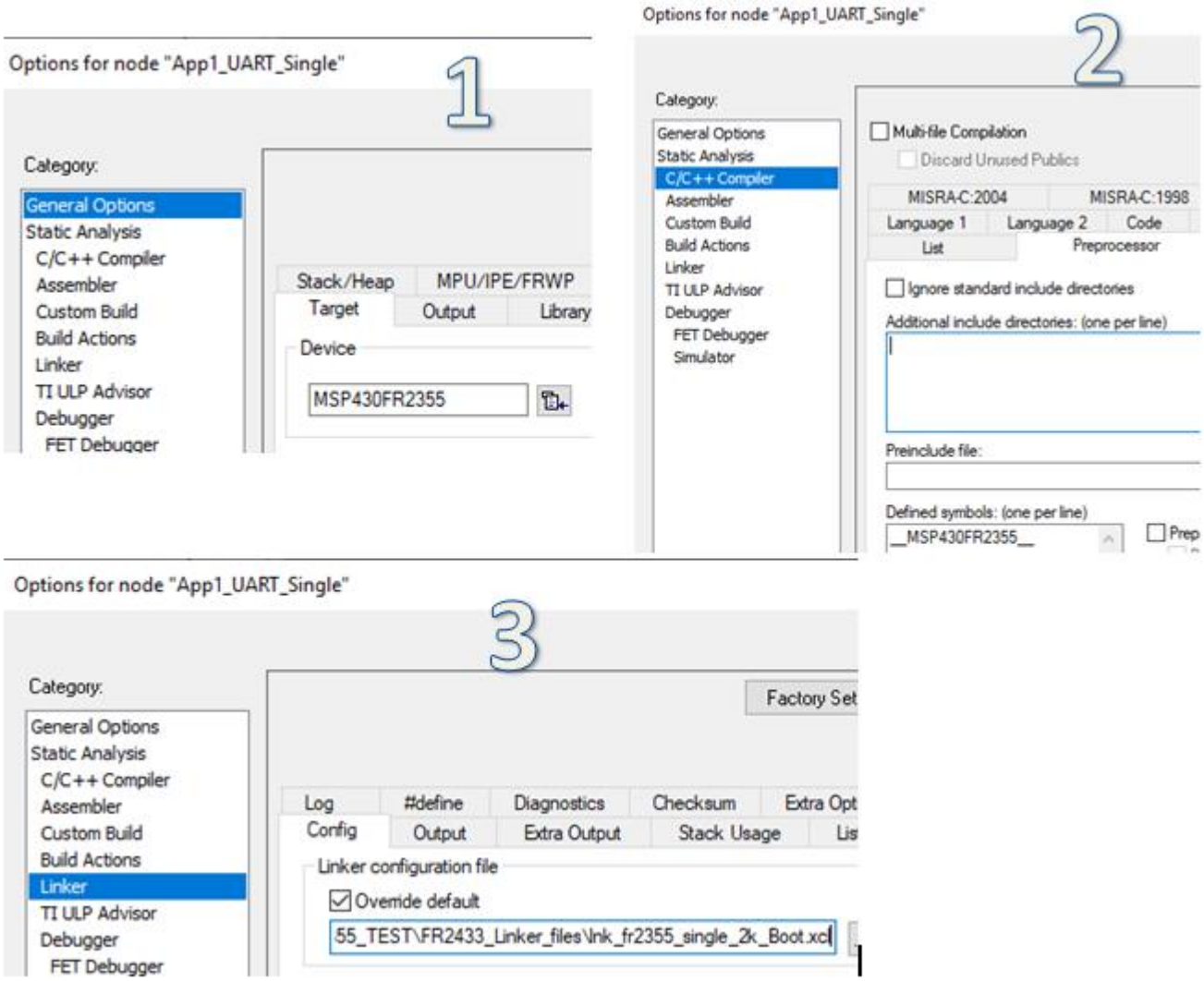

图 21. 配置目标选项

18. 生成应用程序项目以生成 .txt 文件。
19. 使用文件夹“Utilities\430txt_converter”中的工具将 .txt 文件转换为 .c 文件。这将使映像文件易于主机项目使用。

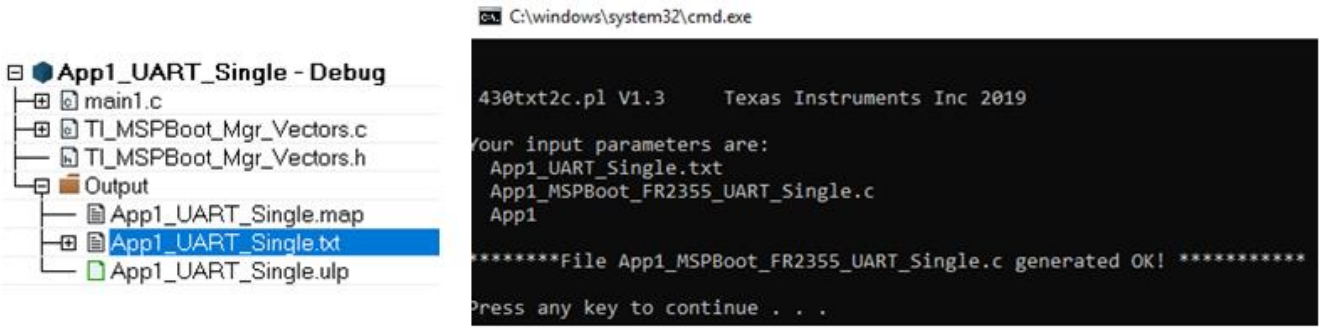

图 **22.** 生成 **.txt** 文件并将其转换为 **.c** 文件

20. 将转换器生成的 .c 文件复制到主机项目文件夹“MSP430FR2433_Host\TargetApps”中
21. 打开名为“UART_Single”的主机项目，然后打开“Main_uart.c”以更改您在步骤 20 中使用的 .c 文件名，如图 23 所示。

```
45
46  #if (defined(TARGET_FR5969_UART_BSLBASED_20bit))
47  #include "MSP430FR5969_App1_UART_Single.c"
48  #include "MSP430FR5969_App2_UART_Single.c"
49  #elif (defined(TARGET_FR5969_UART_BSLBASED_DUALIMG))
50  #include "MSP430FR5969_App1_UART_Dual.c"
51  #include "MSP430FR5969_App2_UART_Dual.c"
52  #else
```

图 **23.** 更改主机中的映像名称

22. 还需要将主机工程中 Main_uart.c 中 CRC_Addr 改为FR2355的起始地址0x8000, 同理App_StartAddress 改为 0x8002.(CCS中为0x8003). 如果只需要下载App1，不用下载App2还需要注释掉代码‘sentBSLFlipFlop = !sentBSLFlipFlop;’修改完成后就可以下载到板子上进行测试了。

7 参考文献

1. *《MSPBoot – 适用于 MSP430 微控制器的主存储器引导加载程序》*
2. *《MSP430™ FRAM 器件引导加载程序 (BSL) 用户指南》*
3. *《MSP430FR58xx、MSP430FR59xx、MSP430FR68xx 和 MSP430FR69xx 系列用户手册》*
4. *《MSP430FR4xx 和 MSP430FR2xx 系列器件用户指南》*
5. *《CC1101 低功耗、低于 1GHz 射频收发器》*

修订历史记录

注：之前版本的页码可能与当前版本有所不同。

Changes from November 7, 2019 to March 30, 2020 **Page**

- 更新了应用程序验证部分 7
- 添加了将目标端示例项目移植到其他 **MSP430FR** 器件部分 24

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