

## Application Note

**MSPM0 引导加载程序 (BSL) 实现**

Chao Gao

**摘要**

本应用手册提供了 MSPM0 器件的引导加载程序 (BSL) 的应用程序级描述。其中总结了与 MSPM0 有关的 BSL 相关资源，并提供了 SDK 中 BSL 示例或工具的分步使用。有关基于 ROM 的 BSL 的更多详细信息，请参阅 [MSPM0 引导加载程序用户指南](#)。

**内容**

<b>1 引言</b> .....	<b>3</b>
1.1 引导加载程序简介.....	3
1.2 BSL 主机实现摘要.....	10
<b>2 非主闪存中的 BSL 配置 (配置 NVM)</b> .....	<b>11</b>
2.1 非主闪存简介.....	11
2.2 示例 - 使用 SysConfig 禁用 PA18 BSL 调用引脚.....	12
<b>3 引导加载程序主机</b> .....	<b>13</b>
3.1 MCU 主机代码简介.....	13
3.2 PC 主机示例.....	19
<b>4 引导加载程序目标</b> .....	<b>21</b>
4.1 基于 ROM 的默认 BSL.....	21
4.2 基于闪存的插件接口演示.....	22
4.3 辅助 BSL 演示.....	24
<b>5 常见问题</b> .....	<b>27</b>
5.1 链接器文件修改.....	27
5.2 由 CCS 恢复出厂设置以恢复器件.....	28
<b>6 参考文献</b> .....	<b>29</b>
<b>修订历史记录</b> .....	<b>30</b>

**插图清单**

图 1-1. 通过 BSL 实现的固件更新结构.....	3
图 1-2. MSPM0 中的 BSL 结构.....	5
图 1-3. 基于 ROM 的 BSL 结构.....	5
图 1-4. 带有基于闪存的插件接口结构的基于 ROM 的 BSL.....	6
图 1-5. 基于闪存的辅助 BSL 结构.....	6
图 1-6. 辅助 BSL 解决方案.....	7
图 1-7. 辅助 BSL 执行流程.....	7
图 1-8. BSL 固件更新系统方框图.....	10
图 2-1. 禁用 PA18 BSL 调用引脚步骤一.....	12
图 2-2. 禁用 PA18 BSL 调用函数.....	12
图 2-3. 选择其他引脚作为 BSL 调用.....	13
图 2-4. 启用 NON-MAIN 闪存擦除.....	13
图 3-1. 主机工程的流程图.....	14
图 3-2. 将 TXT 文件转换为头文件的步骤.....	16
图 3-3. 硬件信号连接.....	17
图 3-4. 将主机工程导入 CCS 中.....	18
图 3-5. 在 CCS 中生成 TI-TXT 十六进制文件.....	18
图 3-6. BSL 默认密码文件 (BSL_Password32_Default.txt).....	19
图 3-7. LaunchPad 套件连接 (左: LP-MSPM0G3507, 右: LP-MSPM0L1306).....	20

图 3-8. 通过 GUI 使用 UART 下载映像的步骤.....	21
图 3-9. 更新 XDS110 固件.....	21
图 4-1. 在 CCS 中启动器件.....	23
图 4-2. 在 CCS 中连接器件.....	23
图 4-3. 在 CCS 中加载符号.....	23
图 4-4. Change Baudrate 命令中的数据段.....	24
图 4-5. 移至 0x4000 cmd 文件修改.....	25
图 4-6. 移至 0x4000 SysConfig 文件修改.....	25
图 5-1. 打开目标配置.....	28
图 5-2. 查找 ccxml 文件.....	28
图 5-3. 启动所选配置.....	29
图 5-4. 使用脚本执行恢复出厂设置.....	29
图 5-5. 控制台中的日志信息.....	29

## 表格清单

表 1-1. MSPM0L 和 MSPM0G BSL 解决方案摘要.....	4
表 1-2. MSPM0C 解决方案摘要.....	4
表 1-3. MSPM0 BSL 功能摘要.....	8
表 1-4. MSPM0 BSL 演示摘要.....	9
表 1-5. MSPM0 BSL 演示协同工作 - MCU 作为主机.....	9
表 1-6. MSPM0 BSL 演示协同工作 - PC 作为主机.....	9
表 2-1. 闪存区域.....	11
表 2-2. NON-MAIN 区域概述.....	11
表 2-3. NON-MAIN 闪存 BSL 配置主参数.....	11
表 3-1. 硬件信号连接.....	15
表 3-2. MSPM0C 的硬件信号连接.....	15
表 3-3. 跳线连接.....	17
表 3-4. 跳线连接.....	20
表 3-5. 独立信号连接.....	20
表 4-1. MSPM0 辅助 BSL 演示摘要.....	24

## 商标

Code Composer Studio™ is a trademark of Texas Instruments.

所有商标均为其各自所有者的财产。

## 1 引言

### 1.1 引导加载程序简介

#### 1.1.1 引导加载程序概念

借助微控制器引导加载程序，可以通过通用异步接收器/发送器 (UART) 或内部集成电路 (I2C) 等通用接口对 MCU 的内部存储器进行编程。借助引导加载程序，可在整个生命周期内快速轻松地对器件进行编程。通过 BSL 实现的固件更新结构显示为图 1-1。根据图 1-1，新固件可通过 BSL 主机下载到 MSPM0 器件中，该主机可以是具有 UART、I2C 等接口的 PC 或处理器。

在本应用手册中，所编程的 MCU 称为目标，而执行更新的器件或工具称为主机。

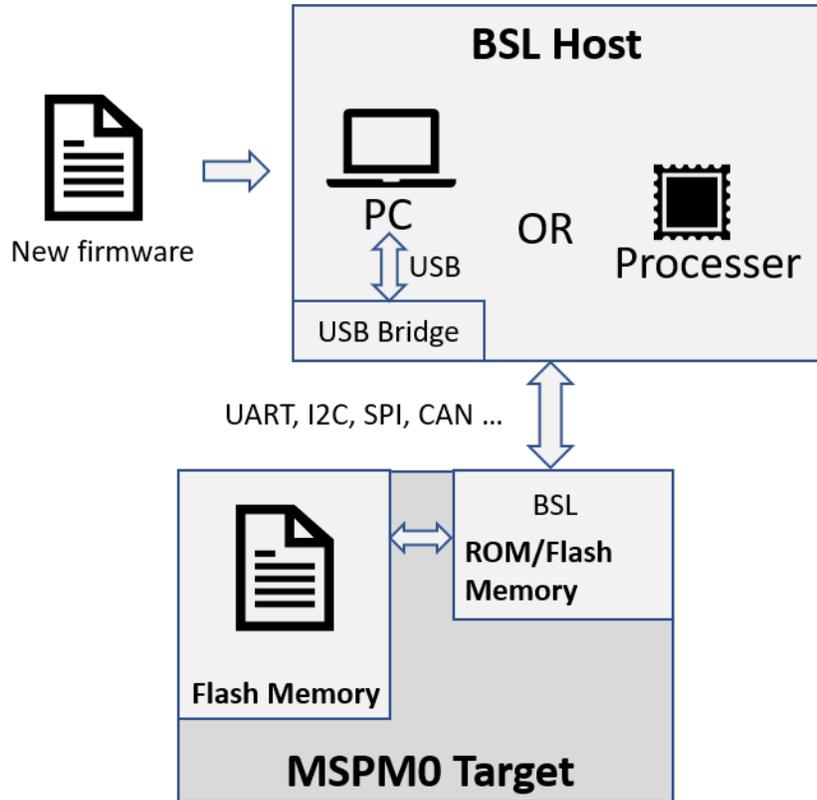


图 1-1. 通过 BSL 实现的固件更新结构

### 1.1.2 MSPM0 引导加载程序结构

MSPM0 器件提供了三种引导加载程序解决方案：基于 ROM 的 BSL、基于 ROM 的 BSL (带有基于闪存的插件接口) 和基于闪存的辅助 BSL。根据应用要求，从三种解决方案中选择一种即可。这三种解决方案使用同一种调用模式 (通用输入/输出 (GPIO) 调用、空白器件检测和软件调用)。有一些参数需要在 NON-MAIN 闪存中配置。如需了解更多详情，请参阅节 2。

**表 1-1. MSPM0L 和 MSPM0G BSL 解决方案摘要**

BSL 解决方案	ROM 成本	闪存成本 (默认)	接口	用于硬件调用的引脚	用于软件调用的引脚	用例
基于 ROM 的 BSL	5K	不适用	UART	4	2	需要遵循 TI 的协议和 UART/I2C 设置
			I2C	4	2	
基于 ROM 且具有插件接口的 BSL	5K (刚刚使用了 BSL 内核部分)	约 1.6K	UART	4	2	需要遵循 TI 的协议，因为接口电平完全是开源的。
		约 1.3K	I2C	4	2	
		约 1.6K	SPI	6	4	
		约 5.8K	CAN	4	2	
基于闪存的辅助 BSL	不适用	约 4.9K	UART	4	2	完全开源。
		约 4.7K	I2C	4	2	
		约 5K	SPI	6	4	
		约 9K	CAN	4	2	

**备注**

硬件调用需要的引脚比软件调用多两个，分别为复位引脚和 GPIO 调用引脚。

**表 1-2. MSPM0C 解决方案摘要**

BSL 解决方案	闪存开销	接口	用于硬件调用的引脚	用于软件调用的引脚	用例
基于闪存的 BSL	约 3.8K	UART	4	2	完全开源。
	约 3.5K	I2C	4	2	

**备注**

只有以下有限的功能存在闪存开销：Mass Erase、Get Device Identity 和 Program。其他功能可以在代码中启用。

图 1-2 展示了 MSPM0 中的 BSL 结构。

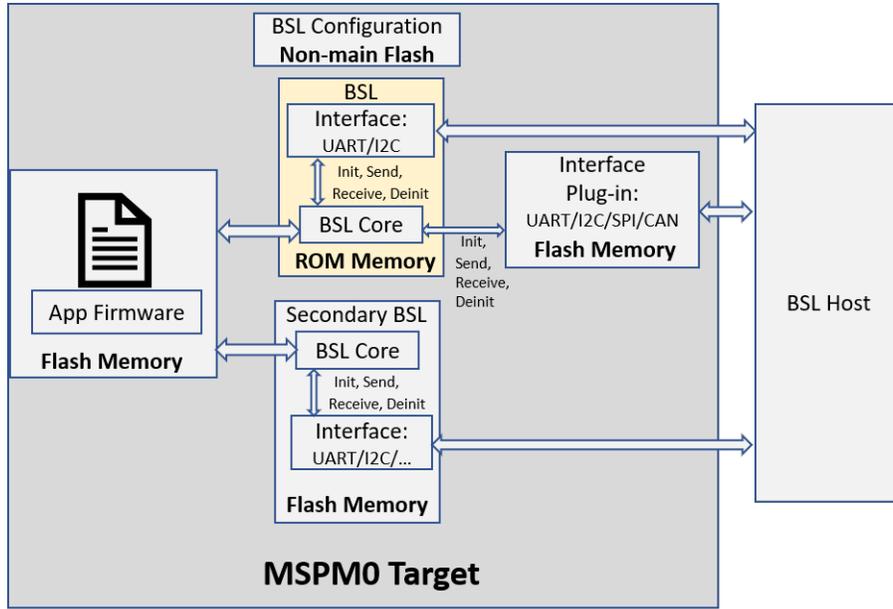


图 1-2. MSPM0 中的 BSL 结构

#### 1.1.2.1 基于 ROM 的 BSL

MSPM0 L&G 器件随附安装了基于 ROM 且高度可定制的引导加载程序，该引导加载程序支持 UART 和 I2C。

基于 ROM 的 BSL 包含 BSL 内核和接口。用于在主机和目标之间接收或发送数据包的接口。BSL 内核用于根据协议解释来自接口的数据包数据。其中一些参数可在非主闪存中配置，如 BSL 密码或 UART/I2C 引脚分配。

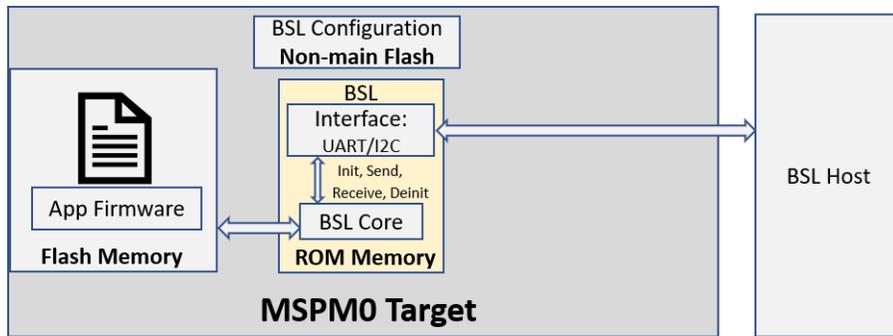


图 1-3. 基于 ROM 的 BSL 结构

### 1.1.2.2 带有基于闪存的插件接口的基于 ROM 的 BSL

如果基于 ROM 的通信接口 (UART/I2C) 无法满足应用的要求，则可以根据需要修改完全开源的基于闪存的接口插件演示，例如 UART、I2C、CAN 和串行外设接口 (SPI)。插件接口演示共享了基于 ROM 的 BSL 内核，用于解析能够节省闪存的数据包。

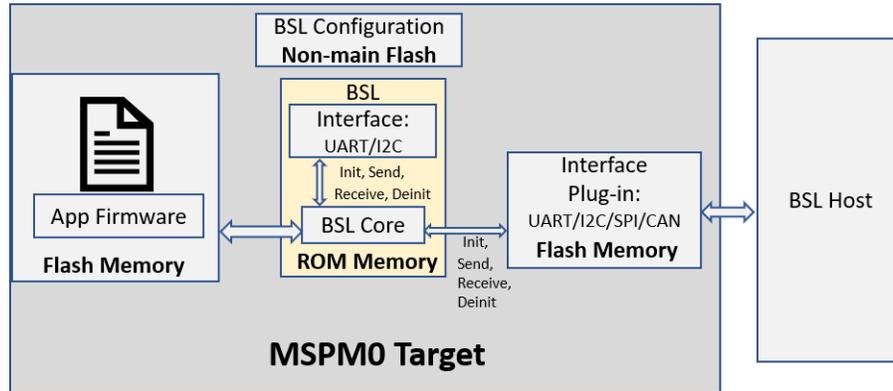


图 1-4. 带有基于闪存的插件接口结构的基于 ROM 的 BSL

### 1.1.2.3 基于闪存的辅助 BSL

如果需要专用协议，则无法再使用基于 ROM 的 BSL 内核，可以参考辅助 BSL 演示。SDK 中提供了完全开放源码的辅助 BSL 演示，您可以使用它轻松修改协议。辅助 BSL 演示的默认协议与基于 ROM 的 BSL 相同。

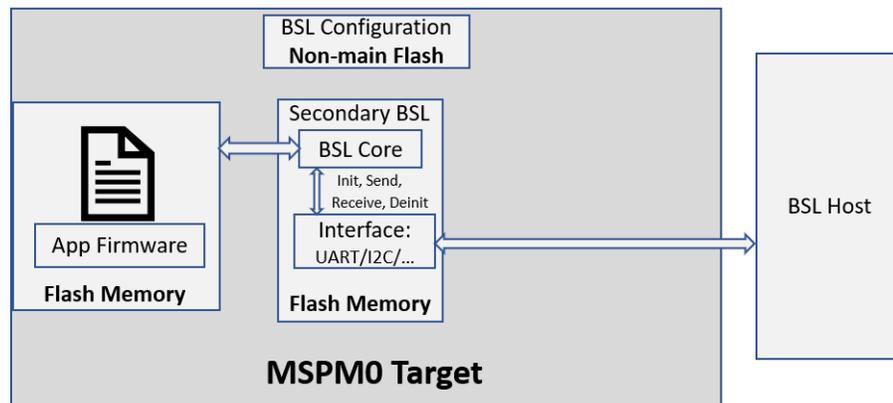


图 1-5. 基于闪存的辅助 BSL 结构

还提到了两种辅助 BSL 演示，如图 1-6 中所示。以 MSPM0G3507 为例：

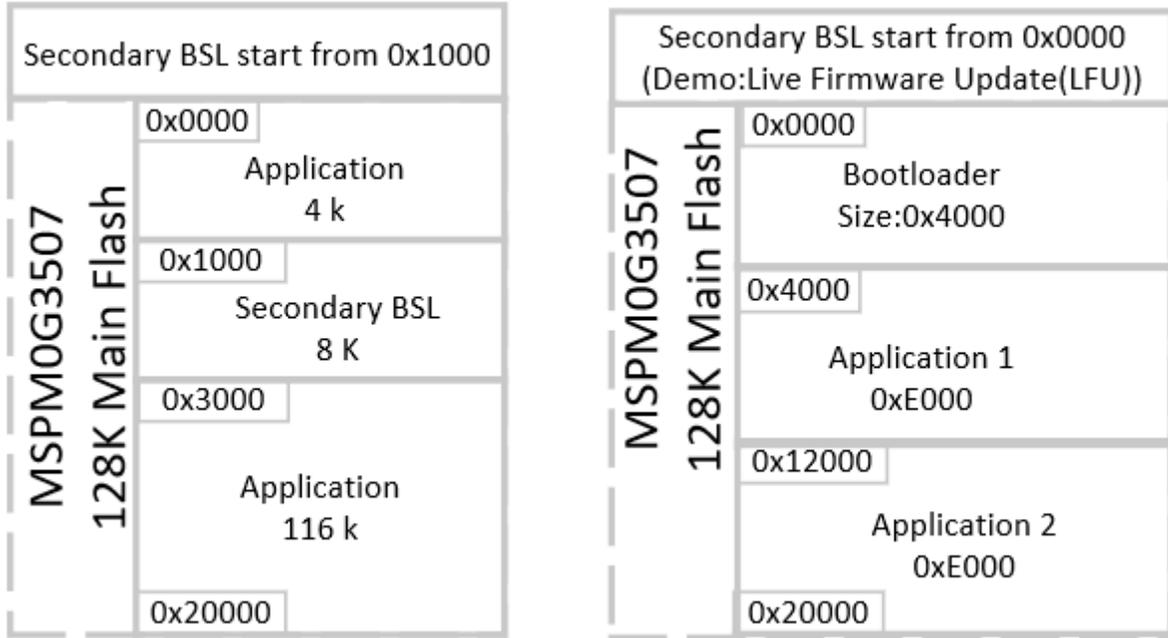


图 1-6. 辅助 BSL 解决方案

- 对于从 0x1000 开始的辅助 BSL，您可以将其放在闪存中除 0x0 之外的任何位置。因为应用程序代码必须从 0x0 地址开始。在这种情况下，器件上电或复位时，它会检查引导代码中的 BSL 调用条件，以决定运行应用程序代码还是 BSL 代码。此演示重复使用了基于 ROM 的 BSL 触发资源。（硬件、软件和空白器件检测，如需更多信息，请参阅 [MSPM0 引导加载程序用户指南](#) 的第 3.2 节。）图 1-7 展示了辅助 BSL 演示执行流程。

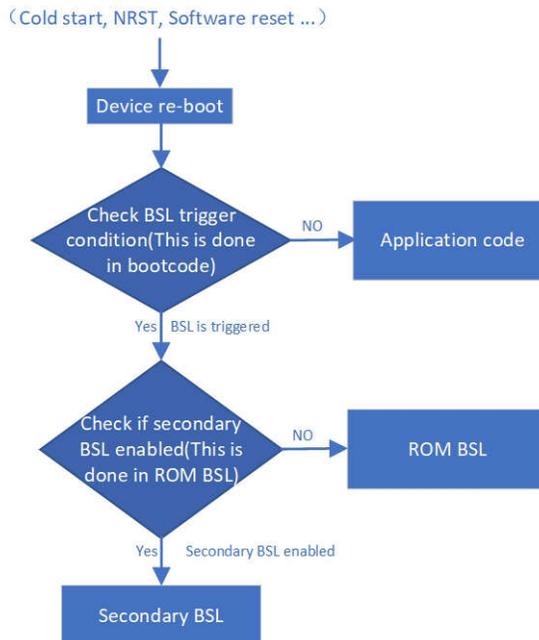


图 1-7. 辅助 BSL 执行流程

- 对于从 0 地址开始的辅助 BSL，每次上电或复位时，MCU 都会进入辅助 BSL 中。在辅助 BSL 中，使用自定义检查判断方式来决定是保持在 BSL 中进行固件更新还是进入应用程序。此解决方案的优势在于，客户可以使用不限于 GPIO、空白器件检测的特殊判断方式。例如，在跳转到应用程序代码之前需要检查应用程序的

CRC，以确保应用程序代码的完整性。另一个用例是对于某些不带 ROM BSL 的 MSPM0 器件，例如 MSPM0C，我们在 SDK 中提供了与其相关的演示代码。在此演示中，检查调用条件后，如果需要跳转到应用程序，它可以将 PC 设置为应用程序的起始地址。如需更多信息，请参阅节 4.3.2。还有一个演示在辅助 BSL 中使用 FreeRTOS，可实现实时固件更新。该演示意味着辅助 BSL 固件更新正在进行，而不会停止应用程序代码。有关更多信息，请参阅 [MSPM0 实时固件更新 \(LFU\) 引导加载程序实现](#)。

### 1.1.3 MSPM0 BSL 功能和演示摘要

表 1-3. MSPM0 BSL 功能摘要

	器件系列	MSPM0C	MSPM0L	MSPM0G
BSL 通用	BSL 存储器类型	闪存	ROM	ROM
	BSL 存储器大小	>3.5K	5K	5K
	非主闪存中的用户配置	✓	✓	✓
	UART	✓	✓	✓
	I2C	✓	✓	✓
插件接口演示	UART		✓	✓
	I2C		✓	✓
	SPI		✓	✓
	CAN			✓
BSL 调用	GPIO 调用	✓	✓	✓
	空白器件检测	✓	✓	✓
	软件调用	✓	✓	✓
硬件工具	带有 UART 的 XDS110	✓	✓	✓
软件工具	采用 SDK 封装的 MSPM0_BSL_GUI		✓	✓
	Uniflash		✓	✓
安全性	256 位密码保护	✓	✓	✓

SDK 中有一些 BSL 代码示例，可以对其进行总结，如表 1-4 所示。

表 1-4. MSPM0 BSL 演示摘要

	演示类型	演示名称	用例
目标端演示	插件接口演示	bsl_spi_flash_interface	当基于 ROM 的通信接口配置或类型不满足要求 ( 需要使用 UART1 模块作为接口或需要 SPI ) 时，可以使用 TI 的默认 BSL 协议
		bsl_uart_flash_interface	
		bsl_i2c_flash_interface	
bsl_can_flash_interface			
	辅助 BSL 演示	secondary_bsl (uart/i2c/spi/can) flash_bsl ( 适用于 MSPM0C )	当 TI 的默认 BSL 协议无法满足要求时，它重复使用基于 ROM 的 BSL 的相同触发条件，但 MSPM0C 的 flash_bsl 演示除外。
	应用演示	bsl_software_invoke_app_demo (uart/i2c/spi/can)	应用示例代码可与基于 ROM 的 BSL、基于闪存的辅助 BSL 演示或基于闪存的接口插件演示协同工作，它还包括软件触发功能。
主机端演示	MCU 或处理器作为主机	bsl_host_mcu_to_m0x_target (uart/i2c/spi/can)	将 MCU 或处理器用作主机并遵循 TI 的默认 BSL 协议时。它可与 ROM BSL 和默认辅助 BSL 演示一同使用。
	PC 作为主机	MSPM0_BSL_GUI/Uniflash	将 PC 用作带 UART 的主机并遵循 TI 的默认 BSL 协议时。这意味着，这可用于基于 ROM 的 UART BSL 或默认 UART 插件接口演示或默认辅助 BSL UART 演示。

表 1-5. MSPM0 BSL 演示协同工作 - MCU 作为主机

目标端			主机端	
	存储器位置	BSL 代码演示	应用代码演示	MCU/处理器主机
ROM BSL	ROM	/		
插件接口演示	主闪存 ( 需要与 ROM BSL 协同工作 )	bsl_spi_flash_interface	bsl_software_invoke_app_demo (uart/i2c/spi/can)	bsl_host_mcu_to_m0x_target (uart/i2c/spi/can)
		bsl_uart_flash_interface		
		bsl_i2c_flash_interface		
		bsl_can_flash_interface		
辅助 BSL 演示	主闪存	secondary_bsl (uart/i2c/spi/can)		

表 1-6. MSPM0 BSL 演示协同工作 - PC 作为主机

目标端			主机端	
	存储器位置	BSL 代码演示	应用代码演示	PC 主机
ROM BSL	ROM	/		MSPM0_BSL_GUI/Uniflash
插件接口演示	主闪存 ( 需要与 ROM BSL 协同工作 )	bsl_spi_flash_interface	bsl_software_invoke_app_demo (uart/i2c/spi/can)	不适用
		bsl_uart_flash_interface		MSPM0_BSL_GUI/Uniflash
		bsl_i2c_flash_interface		不适用
		bsl_can_flash_interface		不适用
辅助 BSL 演示	主闪存	secondary_bsl (uart/i2c/spi/can)		不适用

## 1.2 BSL 主机实现摘要

本应用手册介绍了两种主机的实现：一种是具有 XDS110 等接口桥接器的 PC，另一种是 MCU 或处理器。图 1-8 展示了信号连接图。

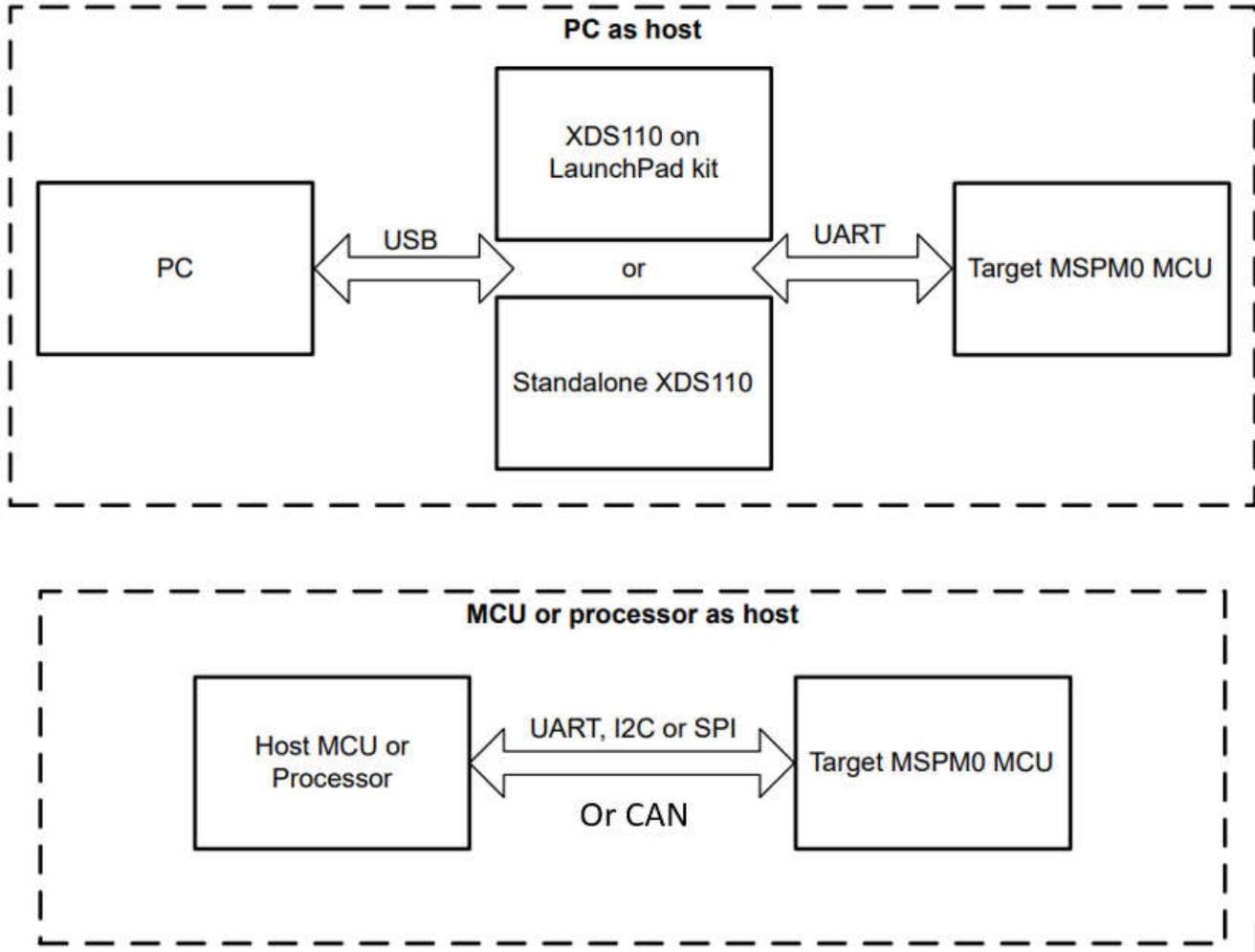


图 1-8. BSL 固件更新系统方框图

将 PC 用作主机时，可通过一个基于 python 3 开发的 GUI 来处理下载操作。其中包含一个预编译的 Windows 可执行文件（在 Win10 64 位上测试），GUI 的源代码也包含在 SDK 中。Uniflash 也可以在 PC 端使用。

当使用 MCU 或处理器作为主机时，有一些基于 MSPM0 的演示将充当主机 MCU，来为另一个 MSPM0 器件进行固件更新。

## 2 非主闪存中的 BSL 配置 (配置 NVM)

### 2.1 非主闪存简介

MSPM0 器件中有三种不同类型的闪存。

表 2-1. 闪存区域

闪存区域	区域内容	可执行	使用者	编程者
FACTORY	器件 ID 和其他参数	否	应用	仅限 TI (不可修改)
NON-MAIN	器件引导配置 (BCR 和 BSL)	否	引导 ROM	TI、用户
MAIN	应用代码和数据	是	应用	用户

NON-MAIN 是闪存的专用区域，可存储 BCR 和 BSL 引导器件所用的配置数据。该区域不用于任何其他目的。BCR 和 BSL 都具有配置策略，这些策略可以保留为默认值（在开发和评估期间是典型值），也可以通过更改编程到 NON-MAIN 闪存区域中的值来针对特定用途进行修改（在生产编程期间是典型值）。由于 MSPM0C 系列没有基于 ROM 的 BSL，因此 MSPM0C 系列器件的 NON-MAIN 中没有与 BSL 相关的配置部分。

表 2-2. NON-MAIN 区域概述

NON-MAIN 部分	起始地址	终止地址
BCR 配置	41C0.0000h	41C0.005Bh
BCR 配置 CRC	41C0.005Ch	41C0.005Fh
BSL 配置	41C0.0100h	41C0.0153h
BSL 配置 CRC	41C0.0154h	41C0.0157h

主 BSL 参数可在表 2-3 中配置。

表 2-3. NON-MAIN 闪存 BSL 配置主参数

参数用例	参数	说明
通用	BSLCONFIGID	BSL 配置 ID
	BSLPW	256 位 BSL 访问密码。(对于辅助 BSL，是可选的)
	BSLCONFIG0	BSL 调用引脚配置和存储器读出策略。(对于辅助 BSL，存储器读出策略是可选的)
	BSLAPPVER	应用版本字的地址。
	BSLCONFIG1	BSL 安全配置。(对于辅助 BSL，是可选的)
	BSLCRC	NON-MAIN 存储器 BSL_CONFIG 部分的 CRC 摘要 (CRC-32)。
基于 ROM 的 BSL	BSLPINCFG0	BSL UART 引脚配置
	BSLPINCFG1	BSL I2C 引脚配置
带有基于闪存的插件接口的基于 ROM 的 BSL	BSLPLUGINCFG	定义 MAIN 闪存中是否存在 BSL 插件及其类型。
	BSLPLUGINHOOK	用于插件初始化、接收、发送和取消初始化函数的函数指针
基于闪存的辅助 BSL	PATCHHOOKID	备用 BSL 配置
	SBLADDRESS	备用 BSL 的地址。

有关 NON-MAIN 闪存的更多详细信息，请参阅 [MSPM0 L 系列 32MHz 微控制器技术参考手册](#) 或 [MSPM0 G 系列 80MHz 微控制器技术参考手册](#)

## 2.2 示例 - 使用 SysConfig 禁用 PA18 BSL 调用引脚

NON-MAIN 配置可以使用 SysConfig 完成。以下示例展示了如何在 NON-MAIN 闪存中禁用 PA18 BSL 调用功能，因为 PA18 用于在 NON-MAIN 中由默认设置进行 BSL 调用。如果应用程序不使用 PA18 作为 BSL 调用，则必须下拉此引脚或在 NON-MAIN 中禁用其 BSL 调用功能，以避免器件在上电或复位时进入 BSL 模式。

1. 打开 SysConfig 并添加配置 NVM，执行此操作时会显示一个错误，以提醒您启用 NON-MAIN 闪存的风险。第 2 步中的接受配置风险可以消除错误。

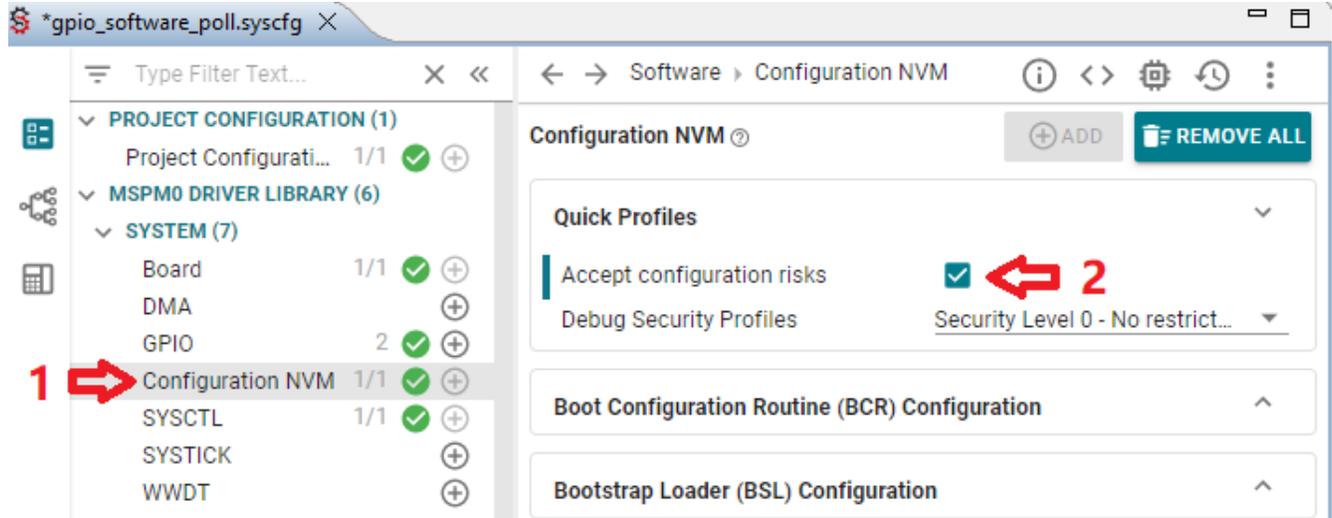


图 2-1. 禁用 PA18 BSL 调用引脚步骤一

2. 禁用图 2-2 中显示的 PA18 BSL 调用函数，或选择图 2-3 中显示的另一个 BSL 调用引脚。

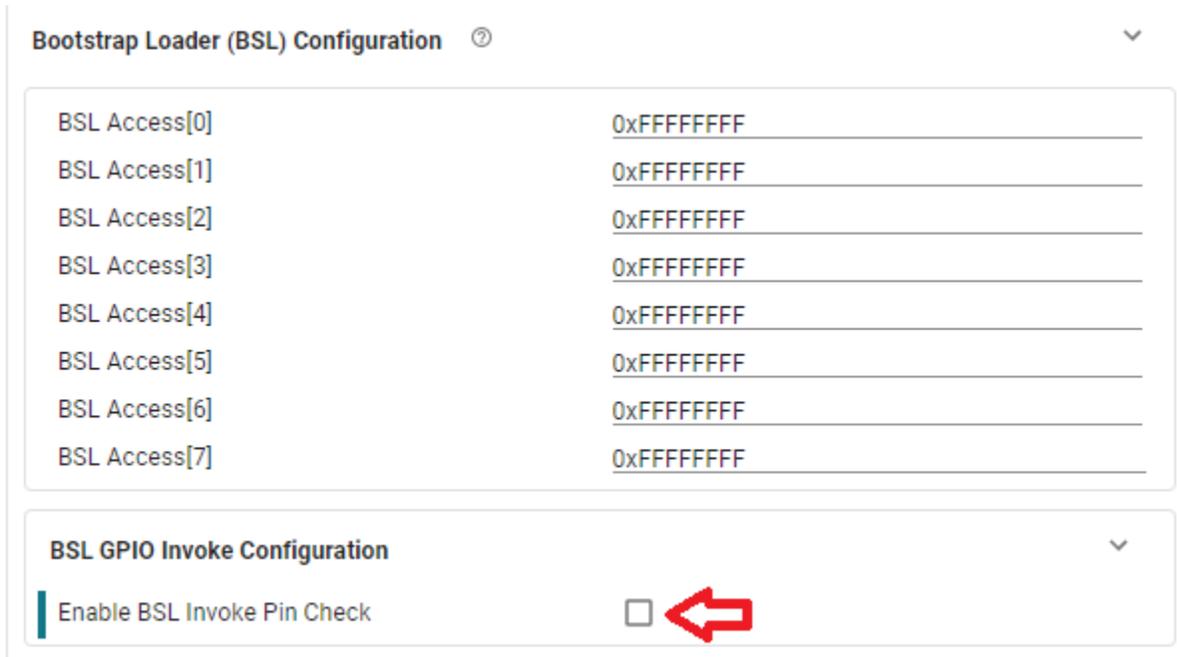


图 2-2. 禁用 PA18 BSL 调用函数



图 2-3. 选择其他引脚作为 BSL 调用

- 在 Code Composer Studio™ (CCS)、IAR 或 Keil 中构建工程，然后将代码下载到闪存中。下载映像的重要操作是启用 NON-MAIN 闪存擦除。例如，在 CCS 中按图 2-4 所示将其启用。

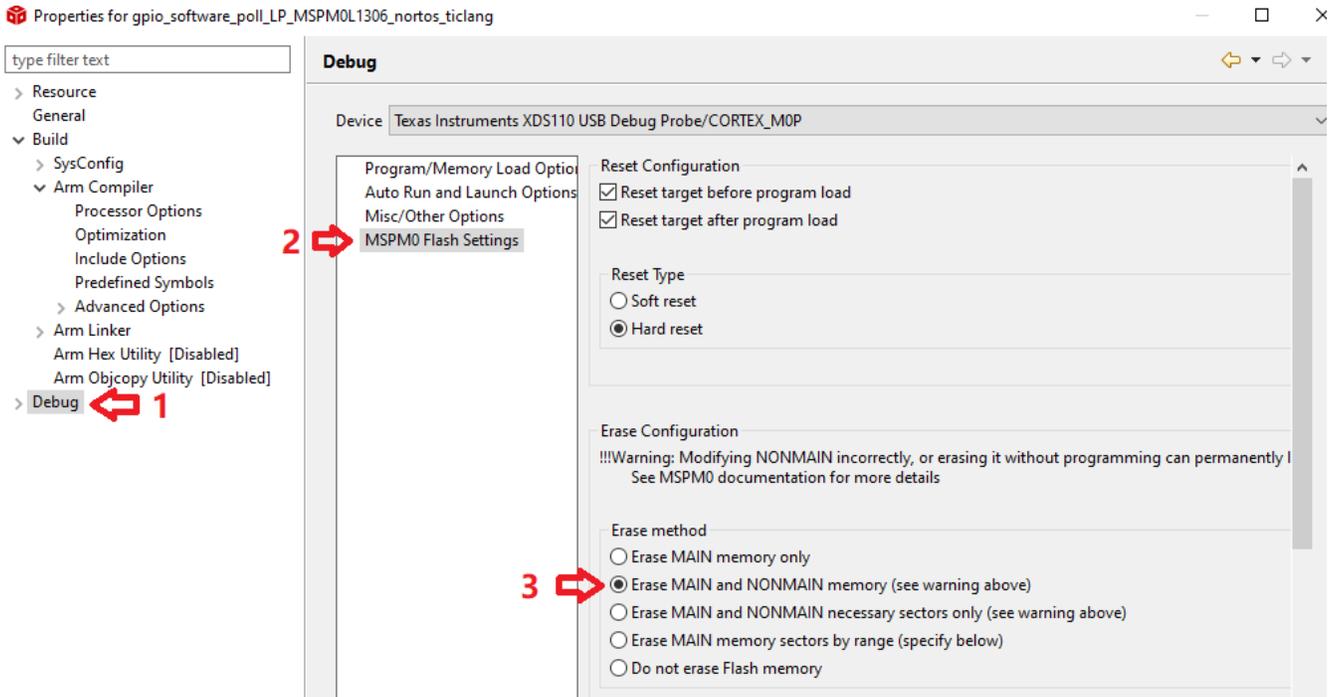


图 2-4. 启用 NON-MAIN 闪存擦除

### 3 引导加载程序主机

#### 3.1 MCU 主机代码简介

文件夹中提供了基于 Code Composer Studio™ (CCS) 的 MCU 主机演示

```
< ... \mspm0_sdk_xxxx\examples\nortos\LP_MSPM0xxx\bsl >
```

这些演示可以通过 UART、I2C、SPI 或 CAN 更新目标 MSPM0 器件。BSL 主机演示源代码包括 application\_image.h 文件中的目标器件固件，该文件由 SDK 中的 GUI 从 .txt 映像文件转换而来。如需了解更多详情，请参阅节 3.1.2。它还在名为 BSL\_PW\_RESET 数组的 main.c 文件中包含 BSL 密码。目标端密码在非主闪存 BSL 配置区域 BSLPW 中定义。图 3-1 展示了主机 BSL 工程的流程图。

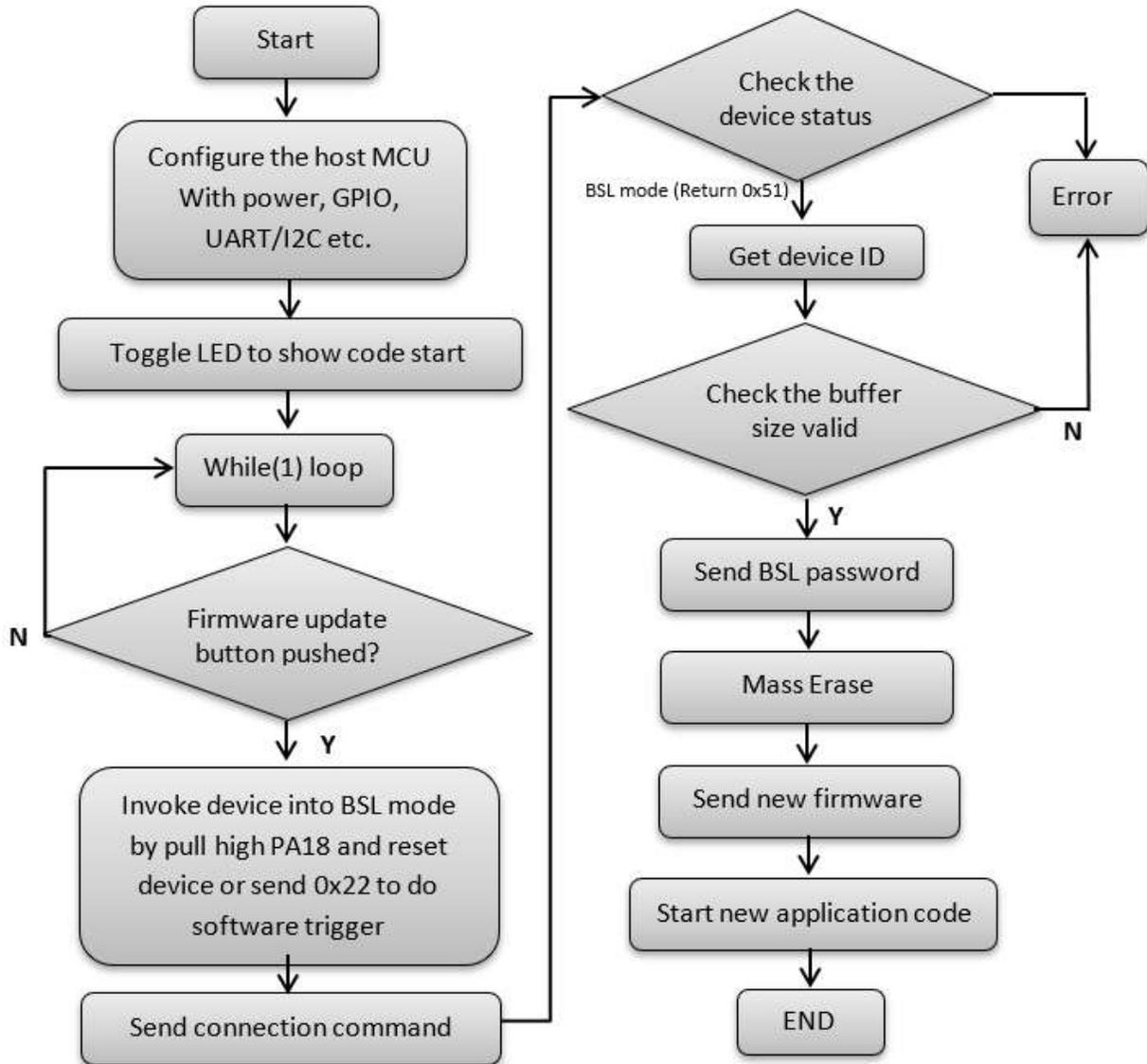


图 3-1. 主机工程的流程图

主机演示可支持将 PA18 引脚拉至高电平然后执行复位的硬件触发器。或者，演示也可以支持软件调用，只需发送 0x22 命令即可触发 BSL。

**备注**

使用软件触发器时，需要首先下载具有软件触发功能演示的应用程序。

### 3.1.1 硬件连接

主机演示代码还使用 MSPM0 作为主机 MCU。表 3-1 展示了主机和目标之间的硬件信号连接

表 3-1. 硬件信号连接

信号	LP-MSPM0G3507		LP-MSPM0L1306	
	主机器件	目标器件	主机器件	目标器件
复位	PB0	NRST 引脚	PA3	NRST 引脚
调用	PB16	PA18	PA7	PA18
UART	PB7/UART1_RX	PA10/UART0_TX	PA9/UART0_RX	PA23/UART0_TX
	PB6/UART1_TX	PA11/UART0_RX	PA8/UART0_TX	PA22/UART0_RX
I2C	PB2/I2C1_SCL	PA1/I2C0_SCL	PA11/I2C0_SCL	PA1/I2C0_SCL
	PB3/I2C1_SDA	PA0/I2C0_SDA	PA10/I2C0_SDA	PA0/I2C0_SDA
SPI	PB9/SPI1_SCLK	PB9/SPI1_SCLK	PA6/SPI0_SCLK	PA6/SPI0_SCLK
	PB8/SPI1_PICO	PB8/SPI1_PICO	PA5/SPI0_PICO	PA5/SPI0_PICO
	PB7/SPI1_POCI	PB7/SPI1_POCI	PA4/SPI0_POCI	PA4/SPI0_POCI
	PB6/SPI1_CS	PB6/SPI1_CS	PA8/SPI0_CS0	PA8/SPI0_CS
CANFD	PA12/CAN_TX	PA13/CAN_RX	\	\
	PA13/CAN_RX	PA12/CAN_TX	\	\

**备注**

只连接一个通信接口：UART、I2C 或 SPI。目标侧引脚是可在非主闪存中更改的默认配置引脚。

**备注**

使用软件调用时，不需要连接复位和调用信号。

**备注**

对于 CANFD，收发器需要与 MSPM0 主机端和目标端连接。

表 3-2. MSPM0C 的硬件信号连接

信号	LP-MSPM0C1104	
	主机器件	目标器件
复位	PA2	NRST 引脚
调用	PA4	PA18
UART	PA24/UART0_RX	PA27/UART0_TX
	PA27/UART0_TX	PA26/UART0_RX
I2C	PA11/I2C0_SCL	PA11/I2C0_SCL
	PA0/I2C0_SDA	PA0/I2C0_SDA

**备注**

使用软件调用时，不需要连接复位和调用信号。

**备注**

使用 UART 接口时，需要移除 J101 上连接到 XDS110 UART 反向通道的跳线。

### 3.1.2 TXT 到头文件的转换

MCU 主机固件包含一个目标应用程序映像作为头文件 (application\_image.h)。头文件是需要编程到 MSPM0 目标器件中的新应用程序固件。为了获取头文件，GUI MSPM0\_BSL\_GUI.exe 中包含一个转换实用工具，路径如下：

< ...\mspm0\_sdk\_xxxx\tools\bs\BSL\_GUI\_EXE >。

1. 在“MoreOption”菜单中选择“TXT\_TO\_H”。
2. 选择要转换的 TI-TXT 格式文件。可以使用输入文件夹中提供的一些简单的应用演示文件。
3. 选择输出文件所在的文件夹（例如，选择名为 Output 的文件夹）。
4. 点击“Convert”按钮以开始转换。

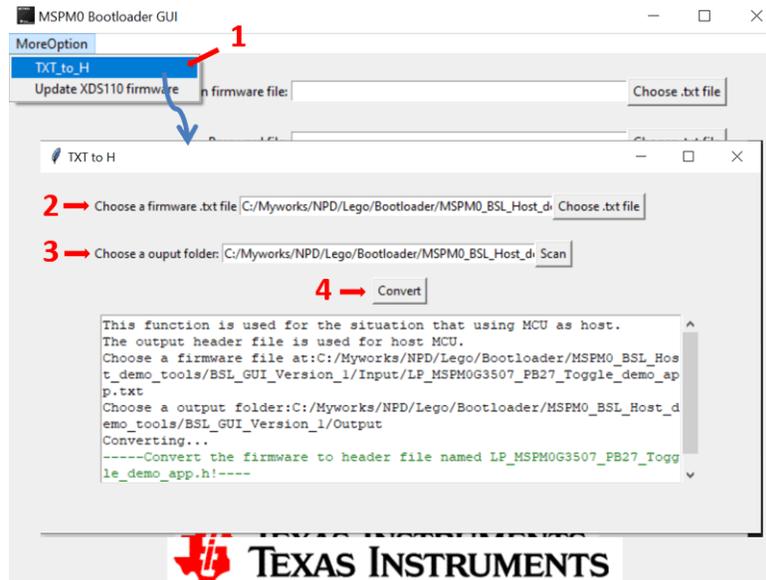


图 3-2. 将 TXT 文件转换为头文件的步骤

### 3.1.3 使用演示的分步操作

以下步骤介绍了如何使用 LP-MSPM0G3507 作为主机来对 MSPM0 MCU 进行编程。MSPM0G3507 用作目标器件，该演示中使用了硬件 BSL 调用和 UART 通信。通过使用适当的硬件连接，可以使用类似的过程通过 UART、I2C 或 SPI 对其他 MSPM0 器件进行编程（请参阅表 3-1）。

1. 按图 3-3 所示连接硬件信号。此示例使用 UART，因此无需连接 I2C 信号。

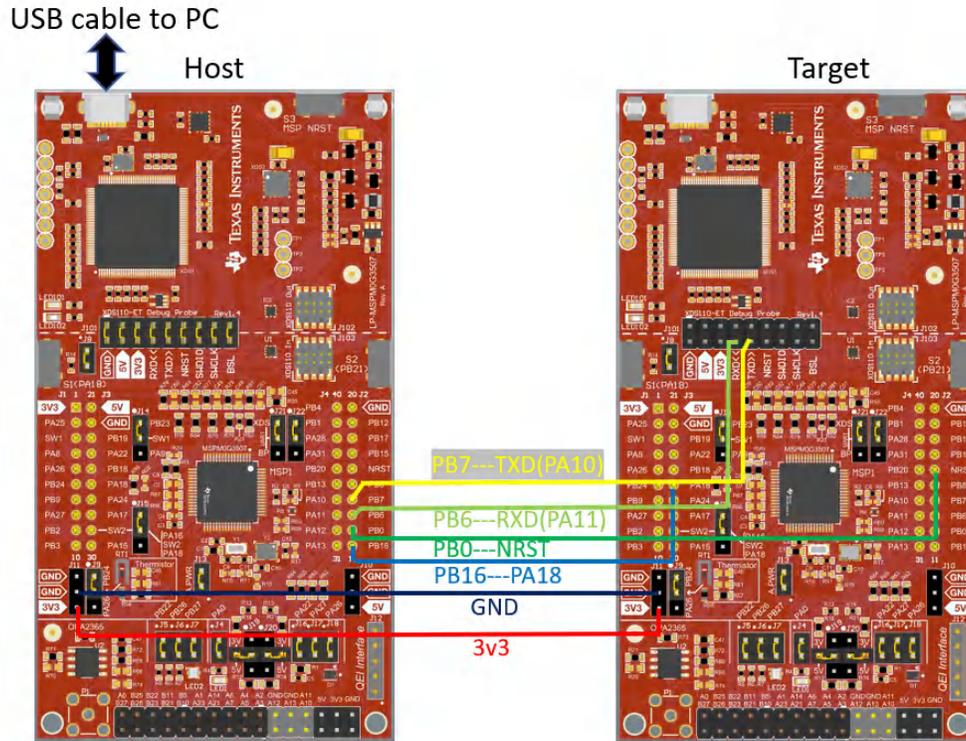


图 3-3. 硬件信号连接

2. 按表 3-3 所示连接跳线。

表 3-3. 跳线连接

电路板	模式	要连接的跳线	要断开的跳线
LP-MSPM0G3507	主机	J101 ( 电源和调试 )、J4、J7(LED)	无
LP-MSPM0G3507	目标	J7(LED) J21、J22 ( UART 至 J101 XDS110 )	全部在 J101 中

**备注**

如果使用 LP-MSPM0L1306 作为目标板，则必须移除 J6 上的跳线。

- 将文件夹 < ... \mspm0\_sdk\_xxxx\examples\nortos\LP\_MSPM0G3507\bsl\bsl\_host\_mcu\_to\_mspm0g1x0x\_g3x0x\_target\_uart> 中提供的带 UART 的 BSL 主机演示导入 CCS 中。

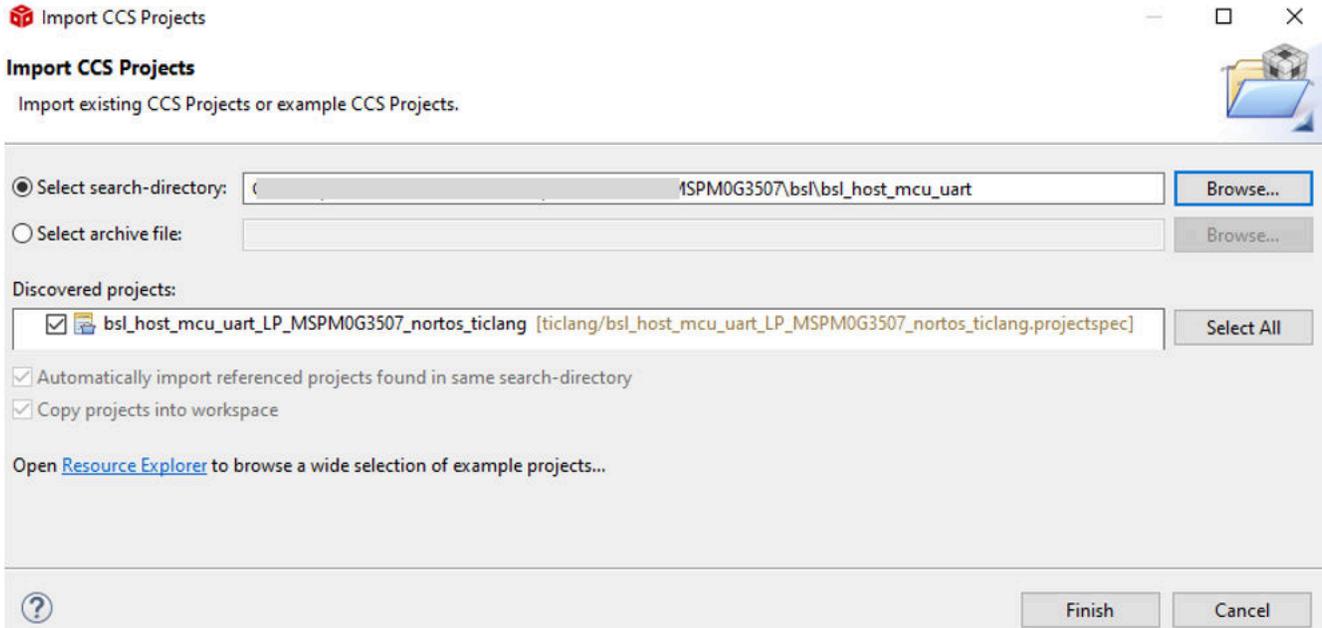


图 3-4. 将主机工程导入 CCS 中

- 必要时，修改 main.c 中 bsl\_password 数组中的密码。默认密码为 32 字节且全部为 0xFF。目标 BSL 密码在 NON-MAIN 存储器中定义。有关更多信息，请参阅技术参考手册 [1]、[2] 或引导加载程序用户指南 [3]。
- 如果只想运行演示，而无需对应用程序代码进行任何更改，BSL 主机演示包含从名为 bsl\_software\_invoke\_app\_demo\_uart 的演示生成的默认固件文件 application\_image\_uart.h，可以跳过第 6 步到第 8 步。
- 将应用程序代码（此处可使用演示 bsl\_software\_invoke\_app\_demo\_uart）导入 CCS 并以 TI-TXT 十六进制格式生成目标器件固件（请参阅图 3-5）。

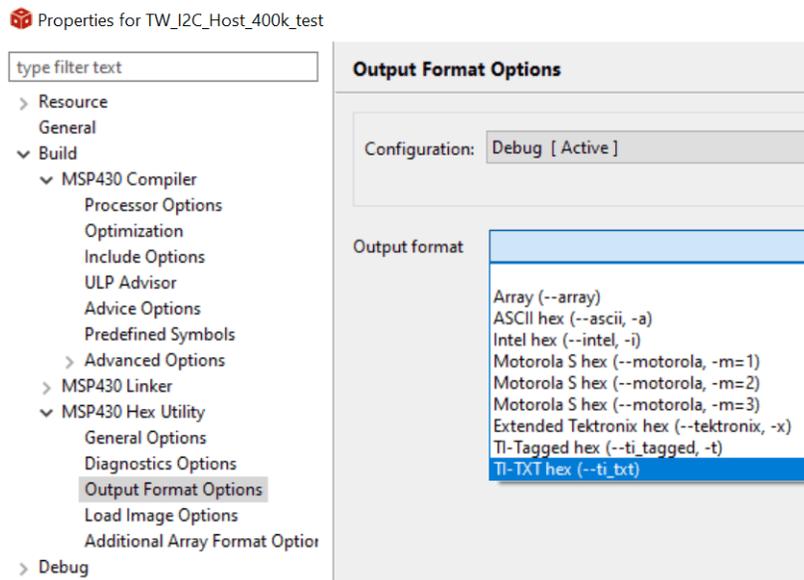


图 3-5. 在 CCS 中生成 TI-TXT 十六进制文件

- 运行 GUI MSPM0\_BSL\_GUI.exe，以将目标器件固件 .txt 格式文件转换为头文件。如需了解更多详情，请参阅节 3.1.2。
- 将 GUI 输出文件 xxx.h 的内容复制到主机工程文件 application\_image.h 中。

9. 构建主机工程并下载到 LP-MSPM0G3507。
10. 按下主机板上的按钮 S2 以启动固件更新。如果出现错误，LED1 会亮起。

### 3.2 PC 主机示例

PC 主机需要软件 GUI ( MSPM0\_BSL\_GUI.exe 或 Uniflash ) 和 USB 转 UART 桥接器。这里包含两个硬件桥接器 ( 可在 MSPM0\_BSL\_GUI.exe 中选择 ) : 一个是 MSPM0 LaunchPad 套件上的 XDS110, 另一个是**独立的 XDS110**。两个桥接器都支持反向通道 UART, 后者可用作 USB 转 UART 桥接器。LaunchPad 套件上的 XDS110 支持 NRST 引脚和 BSL 调用引脚控制, GUI 可以使用该引脚控制来在 MCU 上启动 BSL 模式。对于独立 XDS110, AUX 连接端口中的两个 GPIO 输出引脚 ( IOOUT0 和 IOOUT1 ) 可用于控制目标器件上的 NRST 引脚和 BSL 调用引脚并启动 BSL 模式。( 这是通过 MSPM0\_BSL\_GUI.exe 实现的。 )

---

#### 备注

由于 LP-MSPM0C1104 板载 XDS110 未布局 BSL 调用引脚, GUI 目前不支持该引脚。

---

#### 3.2.1 准备映像文件和密码文件

在通过 GUI 下载固件之前, 请准备两个文件: 应用固件文件和 BSL 密码文件。

GUI (MSPM0\_BSL\_GUI.exe) 仅支持 TI-TXT 格式。有关如何使用 CCS 生成此格式映像文件的详细信息, 请参阅节 3.1.3 中的 6。

密码文件的格式类似于 TI-TXT 格式, 如图 3-6 所示。BSL 密码在非主存储器中定义。有关更多信息, 请参阅技术参考手册 [1] [2] 或引导加载程序用户指南 [3]。如果 BSL 密码不是默认值 ( 全部为 0xFF ), 请修改密码文件。以下文件夹中提供了名为 BSL\_Password32\_Default.txt 的默认密码文件:

< ...\mspm0\_sdk\_xxxx\tools\bsl\BSL\_GUI\_EXE\Input >。

```

@00000000
FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
q
  
```

图 3-6. BSL 默认密码文件 (BSL\_Password32\_Default.txt)

### 3.2.2 使用 GUI 的步骤

1. 将目标器件和 XDS110 连接到 PC。使用 LaunchPad 套件中集成的 XDS110 时，请将 Micro USB 电缆连接到 PC，如图 3-7 所示。

MSPM0G3507 基于 ROM 的 BSL UART 引脚为 PA10 和 PA11，这些引脚都直接连接到 XDS110 反向通道 UART，因此需要 J101 中的所有跳线（请参阅表 3-5）。

在 LP-MSPM0L1306 上，XDS110 反向通道 UART 引脚与 BSL UART 引脚不同，因此请断开 J101 中的 TXD 和 RXD 并使用跳线连接 PA22 和 PA23（请参阅表 3-5）。

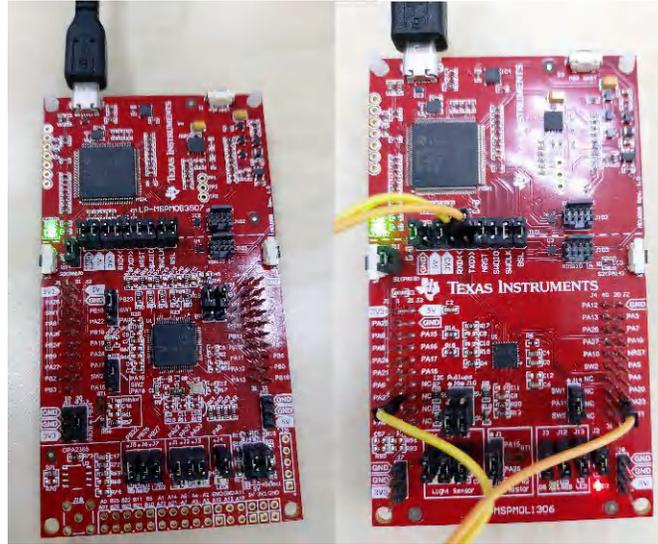


图 3-7. LaunchPad 套件连接 (左：LP-MSPM0G3507，右：LP-MSPM0L1306)

表 3-4. 跳线连接

电路板	模式	跳线需要组装	跳线不需组装
LP-MSPM0G3507	目标	J101 (电源、UART 引脚、复位和 BSL 调用引脚)、J4、J7(LED)、J21、J22 (UART 至 J101 XDS)	不适用
LP-MSPM0L1306	目标	J101 (GND、3V3、NRST、BSL)、J2、J3(LED)	J101 (TXD、RXD)

对于独立 XDS110，辅助接口 (AUX) 使用表 3-5 中的信号连接。

表 3-5. 独立信号连接

信号	独立的 XDS110		目标器件		
	信号	AUX 端口	信号	LP-MSPM0G3507	LP-MSPM0L1306
NRST	IO 输出	IOOUT0	NRST	NRST 引脚	NRST 引脚
调用	IO 输出	IOOUT1	默认值：调用引脚	PA18	PA18
UART	RXD	UARTRX	TXD	PA10/UART0_TX	PA23/UART0_TX
	TXD	UARTTX	RXD	PA11/UART0_RX	PA22/UART0_RX

2. 使用 GUI 将映像下载到目标。
  - a. 选择需要下载的 TI-TXT 格式映像文件。（名为 `input` 的文件夹中有两个演示映像）
  - b. 选择 TI-TXT 格式密码文（`input` 文件夹中有一个默认文件）。有关准备此文件的详细信息，请参阅节 3.2.1。
  - c. 选择硬件桥接器。
  - d. 点击“Download”按钮。

GUI 会自动调用 BSL，因此在此操作期间无需手动调用 BSL。

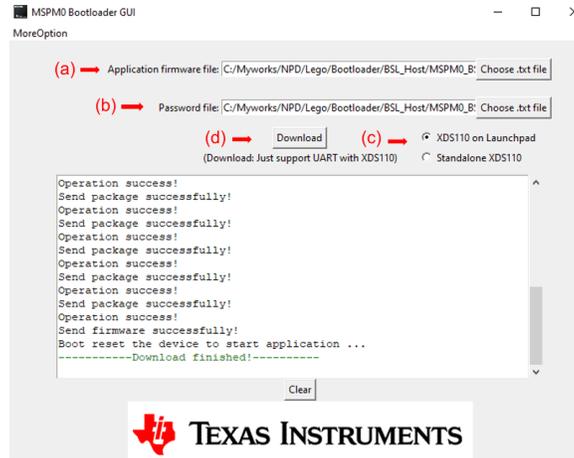


图 3-8. 通过 GUI 使用 UART 下载映像的步骤

3. 如果使用 XDS110，此 GUI 支持 XDS110 固件版本 firmware\_3.0.0.20 或更高版本。如果下载映像时出现错误，请更新 XDS110 固件。

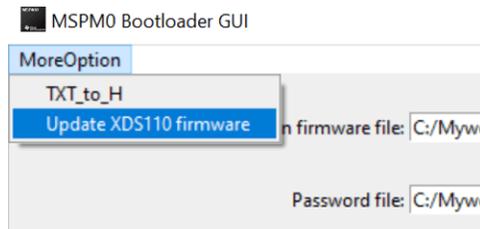


图 3-9. 更新 XDS110 固件

## 4 引导加载程序目标

### 4.1 基于 ROM 的默认 BSL

一些 MSPM0 器件包含基于 ROM 的 BSL。基于 ROM 的 BSL 可以仅支持 UART 和 I2C 接口。这无法更改，但可以针对 NON-MAIN 闪存中的某些特性进行配置。例如，UART/I2C 引脚分配或 I2C 地址等。有关更多详细信息，请参阅 [MSPM0 引导加载程序用户指南](#)。

#### 4.1.1 UART 接口

通过以下配置启用基于 MSPM0 ROM 的 BSL UART：

- 波特率：9600bps (可在 NON-MAIN [仅适用于某些器件]或 BSL 内核命令中更改)
- 数据宽度：8 位
- 1 个停止位
- 无奇偶校验位

UART 引脚分配和波特率 (只针对某些器件) 可在 NON-MAIN 闪存中进行配置。

#### 4.1.2 I2C 接口

通过以下配置启用基于 MSPM0 ROM 的 BSL I2C：

- 地址：0x48 (可以在 NON-MAIN 中更改)
- 地址宽度：7 位

I2C 引脚分配和从地址可在 NON-MAIN 闪存中配置。

## 4.2 基于闪存的插件接口演示

当基于 ROM 的 BSL 接口部分设置无法满足应用的要求时，可使用基于闪存的插件接口演示。由于通信部分的代码均为开源代码，因此客户可以更改代码中的所有设置。请记住，基于闪存的插件接口演示只能接收 BSL 数据包，不能解析数据包。所以，插件接口演示需要与位于 ROM BSL 中的 BSL 内核协同工作来解析命令。

### 4.2.1 UART 接口

默认情况下，通过以下配置启用基于 MSPM0 闪存的 UART：

- 波特率：9600bps
- 数据宽度：8 位
- 1 个停止位
- 无奇偶校验位

#### 4.2.1.1 使用演示的分步操作

以下是有关如何为 MSPM0G3507 使用基于闪存的 UART 插件接口演示的步骤：

1. 将基于闪存的 UART 插件接口演示从 SDK 导入到 CCS 中。

```
<···\mspm0_sdk_xxxx\examples\nortos\LP_MSPM0G3507\bsl\bsl_uart_flash_interface >
```

2. 进行必要的更改并构建工程。
3. 执行恢复出厂设置来清除 NONMAIN 中的任何静态闪存写保护设置。如果器件为空白，则可以跳过此步骤。如需详细了解执行此操作的步骤，请参阅节 5.2。
4. 将 UART 插件代码下载到闪存中。下载映像时的重要操作是启用 NONMAIN 闪存擦除，如图 2-4 所示。无法直接调试插件接口演示。如需了解更多详情，请参阅节 4.2.1.2。
5. 准备一个 LP-MSPM0G3507 LaunchPad 并使用 BSL 主机演示进行固件更新。如需了解更多详情，请参阅节 3.1.3 (MCU 作为主机) 或节 3.2.2 (PC 作为主机)。

#### 4.2.1.2 如何调试插件接口代码

当更改插件接口演示代码并进行调试时，请参照以下这些指南：

1. 进行所需的任何更改并构建插件接口工程。
2. 按照与节 2.2 中的步骤 C 相同的方式将其下载到 NON-MAIN 已擦除的器件中，然后进行下电上电。
3. 按以下图 4-1 所示启动器件。第 2 步是右键点击 ccxml 文件。

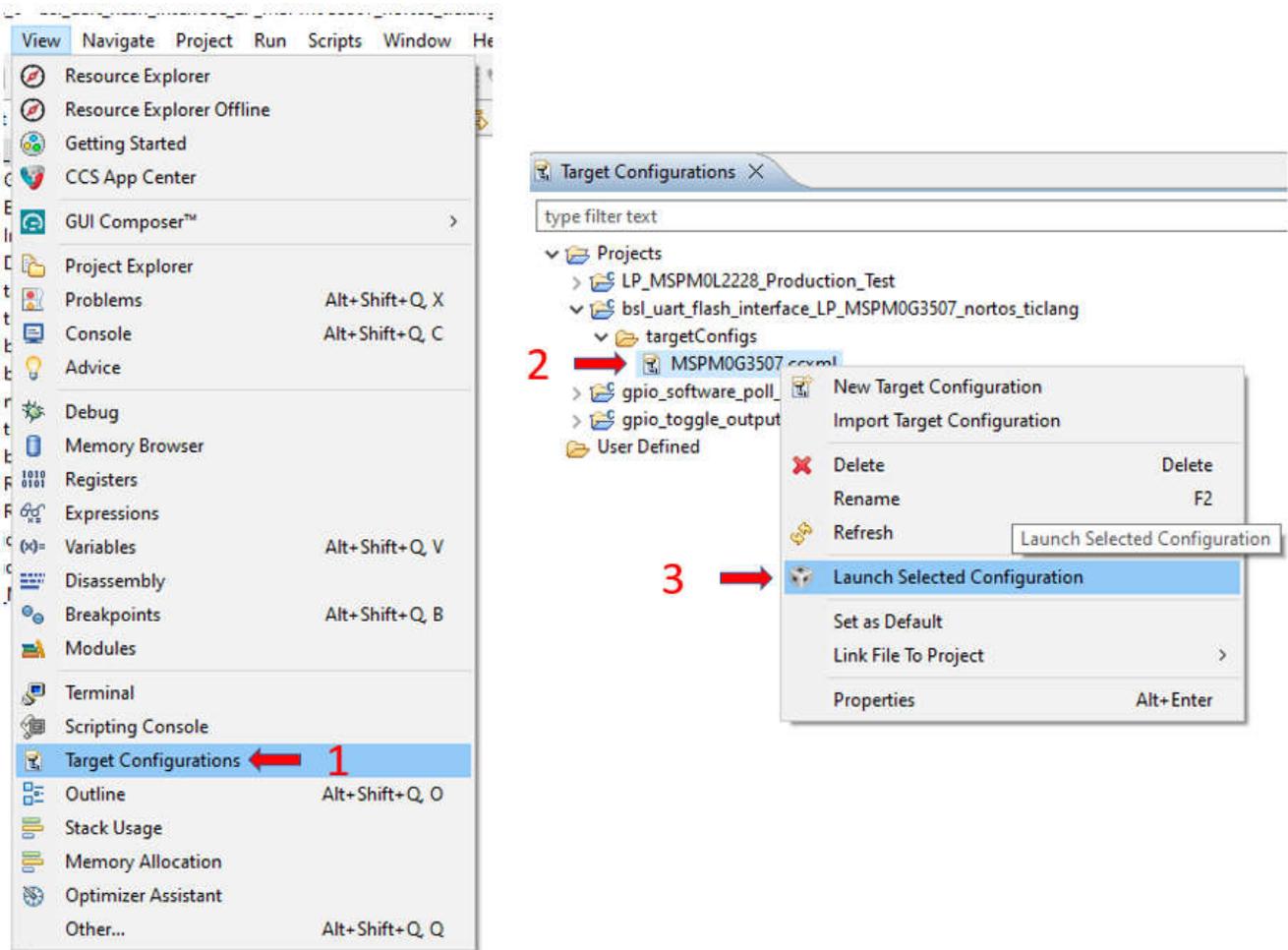


图 4-1. 在 CCS 中启动器件

4. 连接目标。



图 4-2. 在 CCS 中连接器件

5. 加载插件接口代码的符号并放置所需的断点。

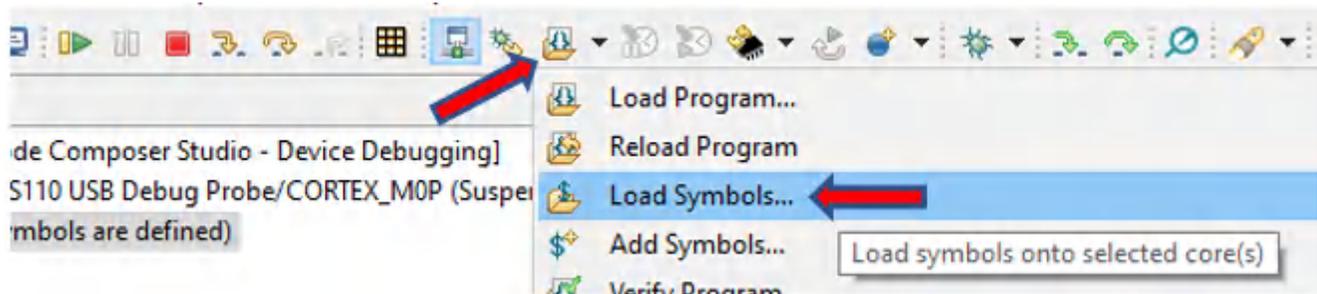


图 4-3. 在 CCS 中加载符号

6. 继续运行代码进行调试。由于应用程序区域为空，器件会自动进入 BSL 模式。

### 4.2.2 I2C 接口

BSL 中的 I2C 接口用作 I2C 目标或从器件。

- 默认情况下，I2C 目标地址为 0x48
- SCL 和 SDA 线路需要外部上拉电阻。

该演示的操作细节与 UART 插件接口类似。有关分步操作以及如何调试演示，请参阅节 4.2.1.1 和节 4.2.1.2

### 4.2.3 SPI 接口

SPI 插件演示在目标模式下配置了 SPI 以及其他默认设置，如下所示：

- Motorola 4 线制连接
- 数据在首个时钟沿捕捉
- 时钟极性低
- 位顺序 MSB

该演示的操作细节与 UART 插件接口类似。有关分步操作以及如何调试演示，请参阅节 4.2.1.1 和节 4.2.1.2

### 4.2.4 CAN 接口

默认情况下，CAN 插件演示按如下所示配置 CAN 模块：

- 该示例配置为最初以 1Mbps 的速率在 CAN 模式下工作。
- 根据从主机通过 `change baudrate` 命令获得的配置来更改通信的比特率。

`change baudrate` 命令中的数据段应该与图 4-4 中所示的格式相匹配。

Padding (5)	DRP (5)	DSJW (4)	DTSEG2 (4)	DTSEG1 (5)	NRP (9)	NSJW (7)	NSEG2 (7)	NTSEG1 (8)	BRS (1)	FD (1)
-------------	---------	----------	------------	------------	---------	----------	-----------	------------	---------	--------

图 4-4. Change Baudrate 命令中的数据段

- 在将 CAN 模式更改为 CAN FD 以校准传输延迟补偿属性值时，向 CAN 总线注入任意 CAN 帧。可以根据需要修改标识值。
- BSL 插件接受的消息标识符为 0x003
- BSL 插件发出的消息标识符为 0x004

该演示的操作细节与 UART 插件接口类似。有关分步操作以及如何调试演示，请参阅节 4.2.1.1 和节 4.2.1.2

## 4.3 辅助 BSL 演示

如果需要专用协议，则无法再使用基于 ROM 的 BSL 内核或不带 ROM BSL 的 MSPM0C，可以参考辅助 BSL 演示。SDK 中提供了完全开放源码的辅助 BSL 演示，您可以使用它轻松修改协议。辅助 BSL 演示的默认协议与基于 ROM 的 BSL 相同。图 1-6 中提到了几种辅助 BSL 演示。

表 4-1. MSPM0 辅助 BSL 演示摘要

演示	SDK 中的工程	用例
从 0x1000 开始的辅助 BSL	<...\mspm0_sdk_xx\examples\nortos\LP_MS PM0L1306 ( 或 LP_MSPM0G3507 ) \bsl\secondary_bsl_uart/i2c/spi/can>	只能用于在引导代码中进行 BSL 调用检测的器件 ( 通常是带有 ROM BSL 的器件 ) 并且需要专用协议。
适用于 MSPM0C 且从 0x0 开始的基于闪存的 BSL	<...\mspm0_sdk_xx\examples\nortos\LP_MS PM0C1104\bsl\flash_bsl>	MSPM0 没有基于 ROM 的 BSL，或者需要在跳转到应用程序之前更改判断条件，例如每次上电或复位时执行应用程序的区域 CRC。
实时固件更新 BSL	不适用	需要实时固件更新

传统的基于闪存的 BSL 解决方案与适用于 MSPM0C 且从 0x0 开始的基于闪存的 BSL 演示非常相似。对于此类解决方案，它将直接跳转到应用程序代码中，以便将 PC 设置为应用程序代码的起始地址。但在某些意外情况下，它可能会导致栈冲突问题。对于辅助 BSL 从 0x1000 开始的解决方案，它使用复位来跳转到应用程序代码，

在进入将更加稳定的应用程序代码之前复位所有寄存器或 SRAM。因此，如果 MSPM0 器件具有基于 ROM 的 BSL 并且还需要专用协议，强烈建议使用从 0x1000 开始的辅助 BSL 演示。

### 4.3.1 基于闪存的辅助 BSL 从 0x1000 开始

辅助 BSL 从 0x1000 开始，它可以放置在闪存区域中的任何位置（从 0x0 开始的位置除外）。因为应用程序代码必须从 0x0 开始。图 1-6 展示了辅助 BSL 演示执行流程。它可以支持 UART、I2C、SPI 或 CAN 接口（如果器件支持）。该演示的分步操作与节 4.2.1.1 中所示的操作相同。

当需要在修改后调试代码时，请执行节 4.2.1.2 中的步骤。

在辅助 BSL 中，中断向量表偏移地址已在位于名为 startup\_mspm0xxxx\_ticlang 文件的复位处理程序中移至从 0x1000 开始的位置（由于代码从 0x1000 开始）。

当尝试将辅助 BSL 移至另一个闪存区域时，可以在 cmd 文件中完成此操作。例如，将辅助 BSL 移至从 0x4000 开始的位置，修改 cmd 文件，如图 4-5 所示。

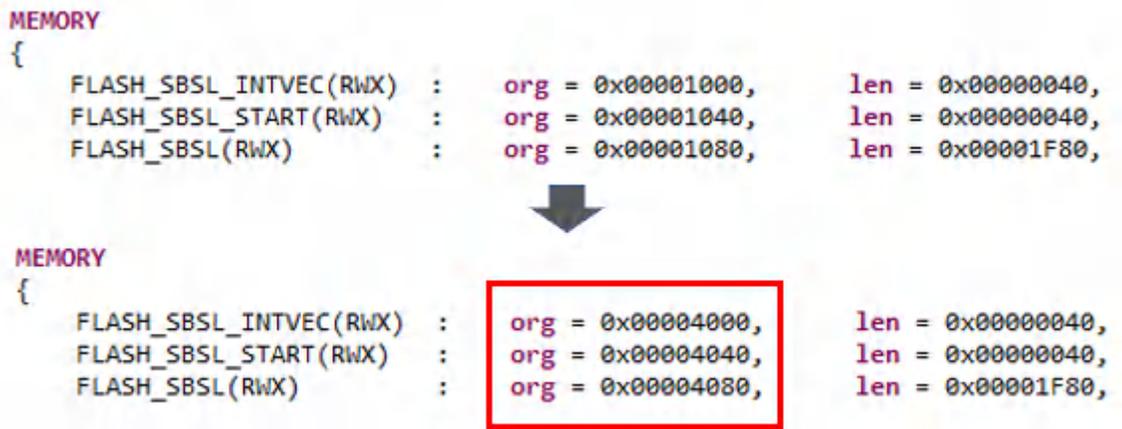


图 4-5. 移至 0x4000 cmd 文件修改

也应在 SysConfig 文件中修改闪存静态写保护参数和备用 BSL 的起始地址，如图 4-6 中所示。

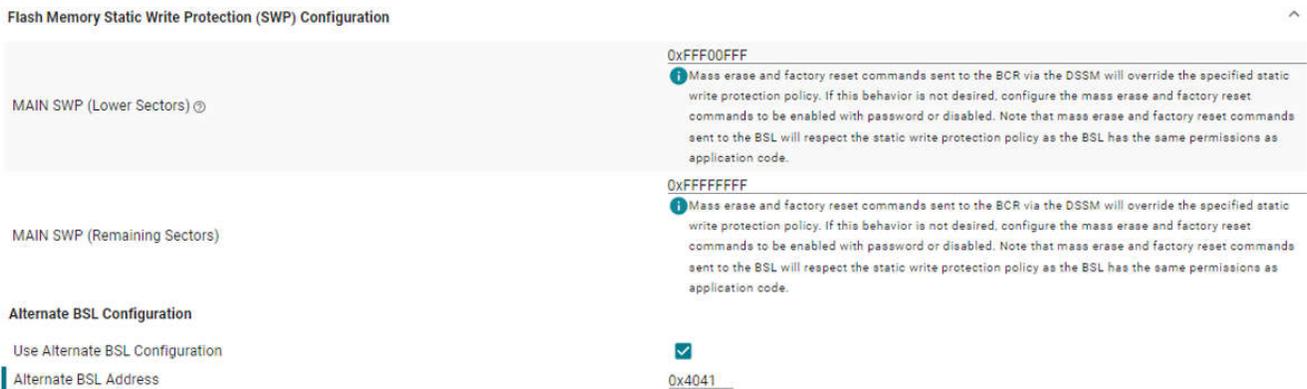


图 4-6. 移至 0x4000 SysConfig 文件修改

除了在辅助 BSL 中进行修改外，还需要修改应用程序的 cmd 文件，以避免重复使用辅助 BSL 所用的闪存区域。

### 4.3.2 基于闪存的辅助 BSL 从 0x0000 开始

对于从 0 地址开始的辅助 BSL，每次上电或复位时，MCU 都会进入辅助 BSL 中。在辅助 BSL 中，使用自定义检查判断方式来决定是保持在 BSL 中进行固件更新还是进入应用程序。此解决方案的优势在于，客户可以使用不限于 GPIO、空白器件检测的特殊判断方式。例如，在跳转到应用程序代码之前需要检查应用程序的 CRC，以确保应用程序代码的完整性。另一个用例是对于某些不带 ROM BSL 的 MSPM0 器件，例如 MSPM0C，我们在 SDK 中提供了与其相关的演示代码。跳转到应用程序时，可以将 PC 设置为应用程序的起始地址。

若要使用这种 BSL，最好创建两个工程：一个用于辅助 BSL，另一个用于应用程序。闪存区域需要分离。因此，一共将有两个中断表，每个工程有一个中断表。它需要配置矢量表偏移寄存器 (SCB->VTOR)，以便在从 BSL 跳转到应用程序代码时激活当前中断表（应用程序跳转到 BSL 将执行复位，进而将自动复位矢量表偏移寄存器）。

还有一个辅助 BSL 演示代码可支持实时固件更新。该演示意味着辅助 BSL 固件更新正在进行，而不会停止应用程序代码。有关更多信息，请参阅 [MSPM0 实时固件更新 \(LFU\) 引导加载程序实现](#)。

#### 4.3.2.1 MSPM0C 基于闪存的 0x0 地址 BSL 演示

由于 MSPMC 器件没有基于 ROM 的 BSL，因此它必须使用基于闪存的 BSL，并且每次上电或复位时它必须从 0x0 地址开始运行调用检测代码。

在此演示中，当器件上电或复位时，它将首先进入辅助 BSL 代码，在 BSL 复位处理程序中，它将检测 BSL 调用条件（空白检测、GPIO 调用或软件调用），以决定是需要保持在 BSL 代码中以进行固件更新，还是通过将 PC 设置为应用程序的起始地址来进入应用程序代码。此演示不包括应用代码 CRC 校验，您可以参考 [MSP430 应用手册](#)。

目前在此演示中使用以下代码在复位处理程序 ISR 中将 PC 设置为应用程序起始地址。

```
uint32_t *appResetHandler = (uint32_t *) (MAIN_APP_START_ADDR + VTOR_RESET_HANDLER_OFFSET);
appPointer FlashBSL_applicationStart = (appPointer) * (appResetHandler);
/* Before branch check if the address of reset handler is a valid Flash address */
if (((uint32_t *) MAIN_APP_RESET_VECTOR_ADDR) >= MAIN_APP_START_ADDR &&
    (((uint32_t *) MAIN_APP_RESET_VECTOR_ADDR) < (MAIN_APP_START_ADDR + DEVICE_FLASH_SIZE))) {
    FlashBSL_applicationStart();
}
```

还有另一种简单的方法可以执行如下跳转（APP\_AREA\_START\_ADDR 是保存中断向量表的应用程序区域起始地址，移位 4 个字节即可获取复位处理程序地址）

```
/*! Jumps to application using its reset vector address */
#define TI_MSPBoot_APPMGR_JUMPTOAPP()    (((void (*)()) (*(uint32_t *) (APP_AREA_START_ADDR + 4)))
();)
```

如果出现一些跳转问题，请尝试以下汇编代码；如果引导代码和应用程序代码共享某些 SRAM 区域，此汇编代码会先清除 RAM，然后跳转到应用程序的起始地址。

```

__asm(
#if defined(__GNUC__)
".syntax unified\n" /* Load SRAMFLASH register*/
#endif
"ldr r4, = 0x41C40018\n" /* Load SRAMFLASH register*/
"ldr r4, [r4]\n"
"ldr r1, = 0x03FF0000\n" /* SRAMFLASH.SRAM_SZ mask */
"ands r4, r1\n" /* Get SRAMFLASH.SRAM_SZ */
"lsrs r4, r4, #6\n" /* SRAMFLASH.SRAM_SZ to kB */
#if defined ECC
"ldr r1, = 0x20300000\n" /* Start of ECC-code */
"adds r2, r4, r1\n" /* End of ECC-code */
"movs r3, #0\n"
"init_ecc_loop: \n" /* Loop to clear ECC-code */
"str r3, [r1]\n"
"adds r1, r1, #4\n"
"cmp r1, r2\n"
"blo init_ecc_loop\n"
#endif
"ldr r1, = 0x20200000\n" /* Start of NON-ECC-data */
"adds r2, r4, r1\n" /* End of NON-ECC-data */
"movs r3, #0\n"
"init_data_loop:\n" /* Loop to clear ECC-data */
"str r3, [r1]\n"
"adds r1, r1, #4\n"
"cmp r1, r2\n"
"blo init_data_loop\n"
//Jump to Reset_Handler
"ldr r0, = 0x7004\n" //FLASH_SBSL_INTVEC in .cmd file+ 4
"ldr r0, [r0]\n"
"b!x r0\n"
);

```

如果器件支持 ECC SRAM，可添加 ECC 的定义。在此演示中，应用程序起始地址保存在地址 0x7004，根据您的应用程序起始地址对其进行更改。

如果在外设初始化之后放置跳转（在执行 main() 函数之前不调用跳转函数），这里有几点需要注意：

- 不要在 ISR 中跳转（复位处理程序 ISR 除外）。
- 禁用全局中断（可以使用此函数 `__disable_irq`；并需要通过调用 `__enable_irq`；在应用程序代码中启用它）→ 复位已使用的所有外设 → 清除所有待处理的 NVIC IRQ（调用此 API `NVIC_ClearPendingIRQ(IRQn_Type IRQn)`）→ 如果需要，清除 RAM → 跳转到应用程序代码起始地址。

#### 4.3.2.2 实时固件更新 (LFU) 解决方案

实时固件更新用于在固件更新期间运行应用程序代码。它使用 FreeRTOS 来处理固件更新以及同时运行的应用程序代码。有关更多信息，请参阅 [MSPM0 实时固件更新 \(LFU\) 引导加载程序实现](#)。

## 5 常见问题

### 5.1 链接器文件修改

当前提供的演示基于 CCS，大多数演示需要修改链接器文件以排列存储器。CCS 在 cmd 文件中用于处理这项工作。有关 cmd 链接器文件简介的更多信息，请参阅此网页 [TI 链接器命令文件入门](#)。

## 5.2 由 CCS 恢复出厂设置以恢复器件

如果无法访问器件，请尝试对 CCS 恢复出厂设置以恢复器件。具体步骤如下所示：

1. 硬件连接：带有 MSPM0 器件的 XDS110。

所需信号：GND、SWDIO、SWCLK、NRST

2. 打开目标配置。

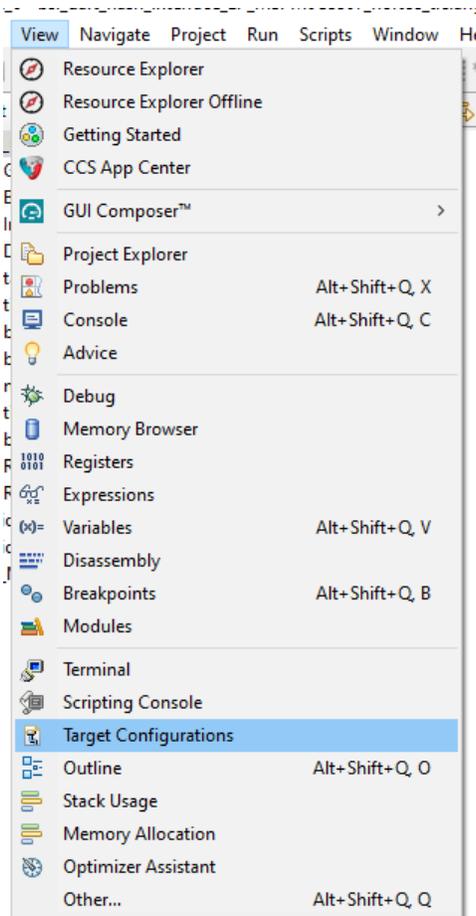


图 5-1. 打开目标配置

3. 在“Target Configurations”窗口中，找到当前 MSPM0 工程并展开文件夹以查找 .ccxml 文件：

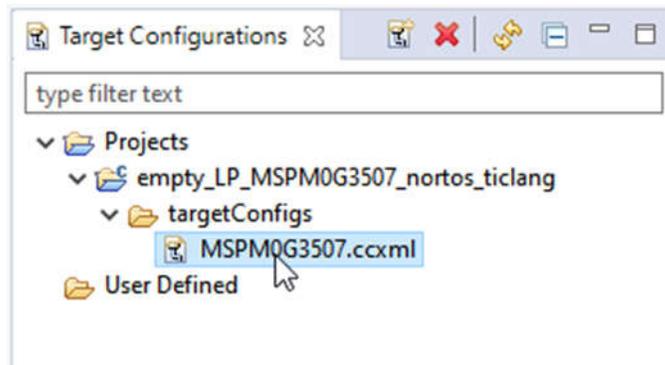


图 5-2. 查找 ccxml 文件

4. 右键点击 .ccxml 文件并点击“Launch Selected Configuration”。

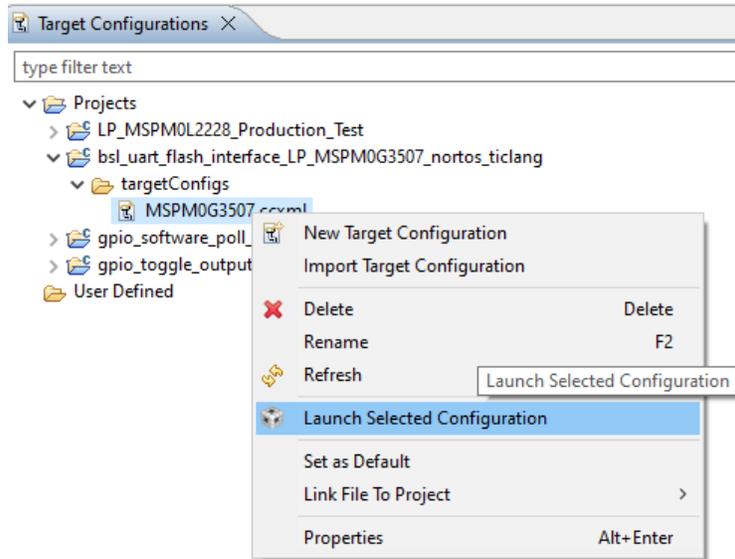


图 5-3. 启动所选配置

5. 依次点击“Scripts” → “MSPM0G3507 Commands” → “MSPM0\_Mailbox\_FactoryReset\_Auto”。

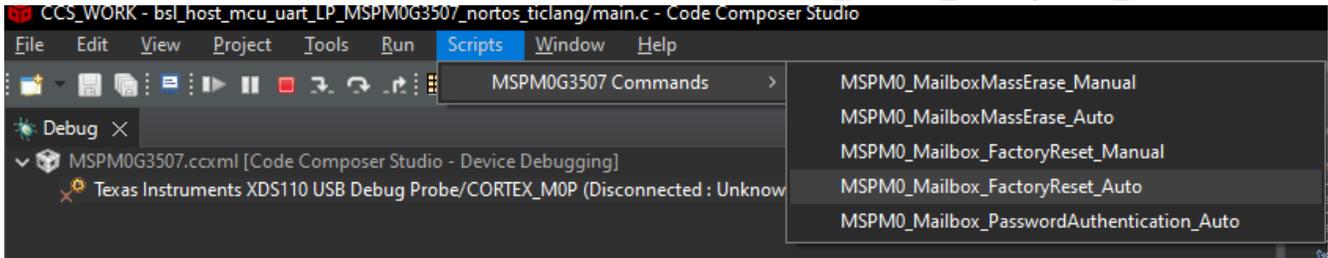


图 5-4. 使用脚本执行恢复出厂设置

6. 控制台将显示以下内容：

```

Console
MSPM0G3507.ccxml
CS_DAP_0: GEL Output: Attempting CS_DAP connection
CS_DAP_0: GEL Output: Attempting SEC_AP connection
CS_DAP_0: GEL Output: Initiating Remote Mass Erase
CS_DAP_0: GEL Output: Mass Erase Command Sent
CS_DAP_0: GEL Output: Press the reset button...
CS_DAP_0: GEL Output: Mass erase executed. Please terminate debug session, power-cycle and restart debug session.
    
```

图 5-5. 控制台中的日志信息

7. 如果这样不起作用，请尝试强制器件进入 BSL 并执行上面的步骤 b 至 e。为了强制器件进入 BSL 模式，如果您尚未在非主闪存中修改默认 BSL 调用引脚 PA18，可以在器件上电之前将 PA18 拉至高电平并使其保持高电平。如果您使用 LaunchPad，则只需在将该板连接到 PC 时按住与 PA18 连接的按钮。

## 6 参考文献

- 德州仪器 (TI) : [MSPM0 G 系列 80MHz 微控制器技术参考手册](#)
- 德州仪器 (TI) : [MSPM0 L 系列 32MHz 微控制器技术参考手册](#)
- 德州仪器 (TI) : [MSPM0 Bootloader 用户指南](#)

## 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

<b>Changes from Revision B (March 2024) to Revision C (September 2024)</b>	<b>Page</b>
• 更新了节 <a href="#">1.1.2</a> .....	4
• 更新了节 <a href="#">1.1</a> .....	6
• 更新了节 <a href="#">1.1.3</a> .....	8
• 更新了节 <a href="#">1.2</a> .....	15
• 更新了节 <a href="#">3</a> .....	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

Copyright © 2024，德州仪器 (TI) 公司