AN3508

利用 MPLAB® Harmony v3 在 SAM E54 单片机（MCU）上实现双存储区自举程序

简介

自举程序是一段代码，用于将应用程序代码（固件）编程或重新编程到单片机的内部闪存，而无需使用外部编程器或调试器。

双存储区自举程序的主要特性如下：

- 上电复位（Power-on-Reset，POR）后运行的第一个程序，负责将固件装入特定的存储单元。
- 可以通过 USB、以太网、CAN、UART、I^2C 和 SPI 等通信接口与主机程序通信以接收固件。
- 使用常规的传统编程方法（例如使用外部编程器或调试器（SWD 和 JTAG））编程到单片机中。
- 负责确定用户打算更新固件还是运行现有固件。单片机的同一存储空间中可以有两个代码映像共存（自举程序和用户应用程序（固件））。

SAM E54 MCU 在内部闪存上提供双存储区支持。使用双存储区闪存时，可在不影响活动存储区上现有应用程序的情况下向非活动存储区中编程新版本固件。

MPLAB Harmony v3 提供用于 32 位单片机的自举程序框架，可用于升级目标器件上的固件，而无需使用外部编程器或调试器。本文档将介绍 MPLAB Harmony v3 提供的双存储区自举程序。双存储区自举程序可利用内部闪存的双存储区功能更加安全地升级应用程序。

目录

1. 硬件和软件要求

1.1 SAM E54 Xplained Pro 评估工具包

SAM E54 Xplained Pro 评估工具包是用于评估 SAM E54 单片机（MCU）的开发工具包。SAM E54 基于 Arm® Cortex® -M4，能够在 120 MHz 频率下运行。该 Pro 评估工具包包括板上嵌入式调试器，无需借助外部工具即可对 SAM E54 进行编程或调试。该评估工具包还提供外部连接器以扩展电路板的功能，并简化定制设计的开发过程。

SAM E54 Xplained Pro 评估工具包可从 Microchip 直销网站购买。

1.2 MPLAB® X 集成开发环境（Integrated Development Environment，IDE）和 XC 编译器

MPLAB X IDE 是一款可扩展且高度可配置的软件程序，包含多种功能强大的工具，可帮助用户发现、配置、开发、调试和验证大多数 Microchip 单片机的嵌入式设计。

MPLAB X IDE 可从 Microchip 网站获取。本文档使用 MPLAB X IDE 版本 5.35。

MPLAB XC 编译器可从 Microchip 网站获取。本文档使用 MPLAB XC32 版本 2.40。

1.3 MPLAB Harmony v3

MPLAB Harmony v3 是一个完全集成的嵌入式软件开发框架，能够提供灵活且可互操作的软件模块，可使用户将资源专注于 32 位 PIC®和 SAM 单片机应用程序的开发，而无需处理器件详细信息、复杂协议和库集成等挑战。

MPLAB Harmony v3 包括 MPLAB Harmony 配置器（MPLAB Harmony Configurator，MHC），这是一种易于使用的开发工具，带有图形用户界面（Graphical User Interface，GUI），可简化器件设置、库选择、配置和应用程序开发。MHC 可作为插件与 MPLAB X IDE 集成，它具有单独的 Java™可执行文件，可独立用于其他开发环境。

本文档中的示例使用以下资源库，可以从 GitHuB 下载：

- CSP（芯片支持包）
- DEV_PACKS（Harmony v3 产品数据库）
- MHC（MPLAB Harmony v3 配置器）
- 自举程序（自举程序）

MPLAB Harmony v3 框架下载器也可用于下载资源库。

1.4 Python

本文档将介绍如何使用 Python 脚本将二进制输出转换为包含十六进制输出的“C”样式数组。Python 还用于合并自举程序二进制文件和应用程序二进制文件。

本文档中介绍的转换和合并是使用 Python v3.6 来执行。

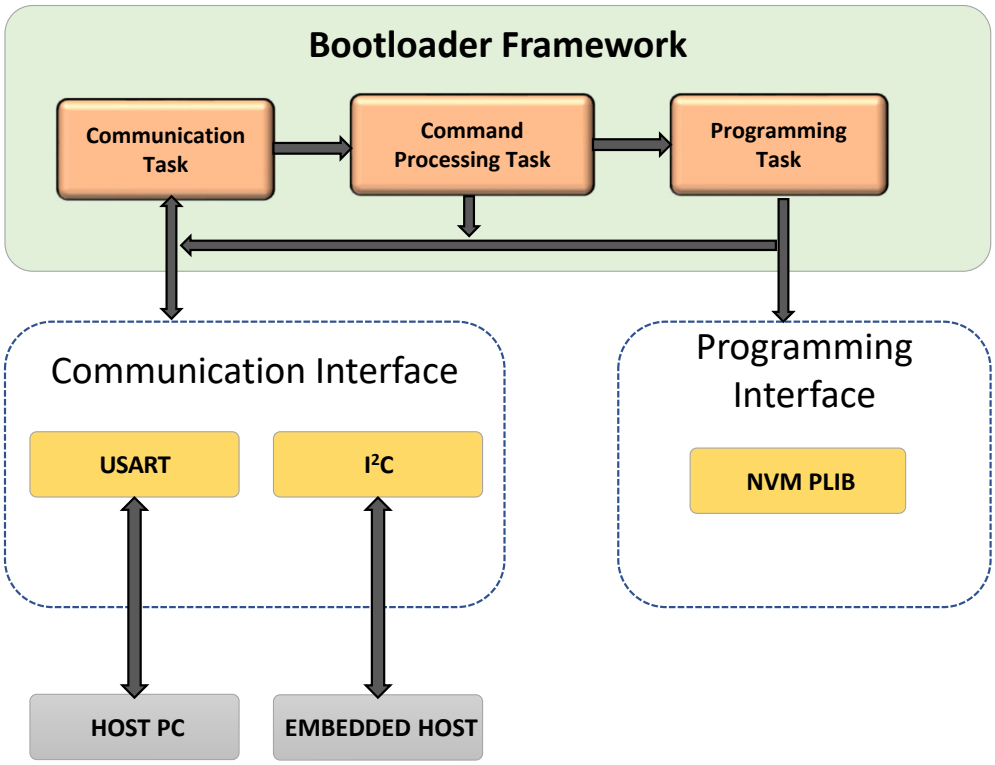
2. 说明

2.1 自举程序框架

MPLAB Harmony v3 自举程序框架分为以下子任务：

- 通信任务。
- 命令处理任务。
- 编程任务。

图 2-1. Harmony v3 自举程序框架



通信任务

该任务负责以轮询模式通过选定的通信接口从主机 PC 或嵌入式主机接收数据。在将主机传入的数据包传送到命令处理任务之前，通信任务会验证该数据包是否包含预期的报头信息。

命令处理任务

该任务处理从通信任务接收的命令并作出响应，然后相应地将响应返回给主机 PC。如果接收到的命令是编程命令，则该任务会将控制权交给编程任务。

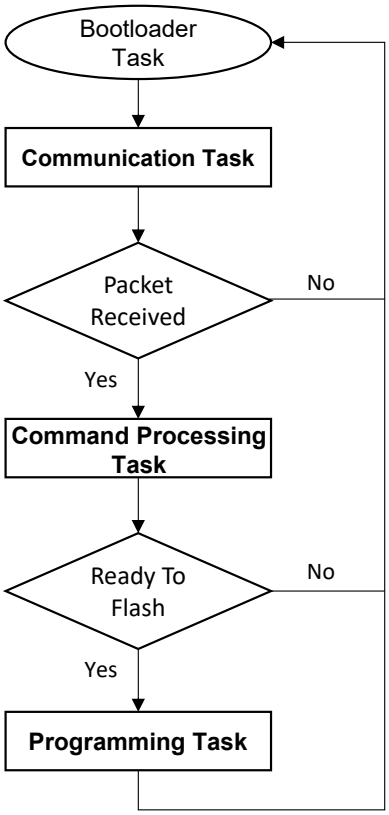
编程任务

该任务负责使用接收到的数据包对内部闪存进行编程。它使用非易失性存储器（Non-Volatile Memory，NVM）外设库执行解锁、擦除或写入操作。在等待闪存操作完成的同时并行调用通信任务以接收下一个数据包。

流程图

固件升级执行流程图如下所示。

图 2-2. 自举程序框架执行流程图



2.2 工作模式

自举程序通过预定义的通信协议与个人计算机主机应用程序进行通信（有关更多信息，请参见 UART 自举程序协议）。

自举程序框架具有以下两种工作模式：

- 基本模式（单存储区自举程序）。
- 故障保护更新模式（双存储区自举程序）。

2.2.1 基本模式（单存储区自举程序）

基本模式自举程序位于闪存的起始单元。它会基于主机发送的二进制文件执行闪存擦除、编程或验证操作。当完成固件升级和验证后，它将跳转到应用程序的起始地址。

有关基本模式自举程序的详细说明，请参见参考资料部分中指定的文档。

2.2.2 故障保护更新模式（双存储区自举程序）

使用基本模式自举程序时面临的挑战之一是引导过程失败。在固件升级阶段，引导过程可能会失败。由于多种原因（例如，接口断开连接和断电等），自举程序可能无法完成正在进行的固件升级。当固件升级过程意外中止时，嵌入式器件将进入不稳定状态，并且可能无法按预期工作。

故障保护自举程序克服了基本自举程序的限制。故障保护自举程序的设计目标是即使在引导过程中出现固件升级失败，系统也仍然可以运行稳定的应用程序映像。

具有双存储区闪存的器件支持故障保护更新。通常，单片机中的存储器由一个或多个存储区构成。尽管大多数单片机都采用单存储区存储器，但仍有一些高端单片机采用双存储区。在使用双存储区闪存的情况下，用户在对一个存储区进行编程时将不会影响另一个存储区的应用程序代码。

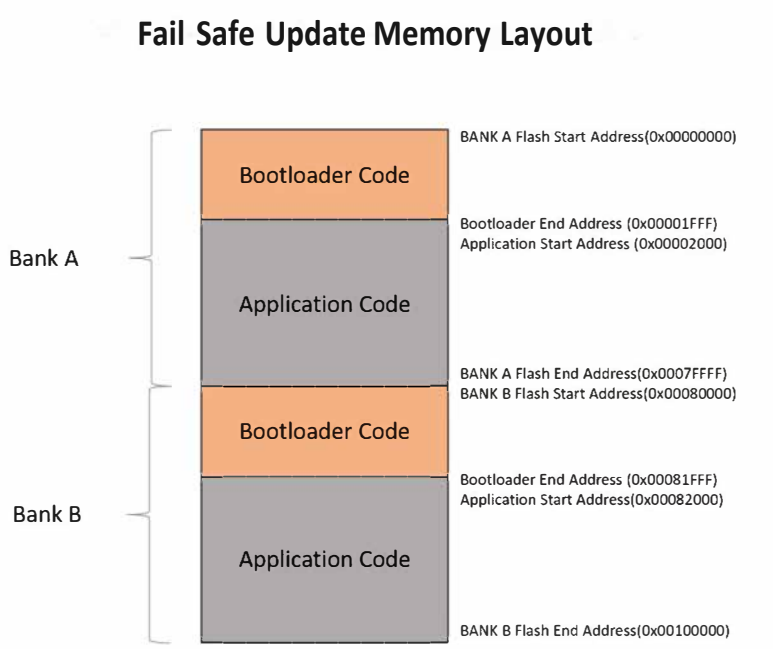
引导故障情况由双存储区自举程序解决（故障保护更新模式）。如果使用双存储区自举程序，当器件在一个存储区中运行时，用户可以为另一个存储区中的固件升级新功能并在升级完成后交换固件。如果升级过程失败，器件可以借助在第一个存储区中运行的工作固件副本继续正常工作。

在双存储区自举程序中，存储器分为两个存储区。每个存储区都包含自举程序代码（位于各存储区的起始单元）和固件（应用程序代码），如下图所示。

从一个存储区引导时，另一个存储区便用作升级缓冲区以接受新固件。在接收并验证新固件后，将切换引导存储区。因此，存储器中有两个可用的固件版本。自举程序可以根据主机应用程序发送的地址在任一存储区中执行闪存操作。完成验证后，它将执行存储区交换并复位系统以运行在另一个存储区中编程的应用程序。

下图给出了双存储区自举程序的存储器布局：

图 2-3. 双存储区自举程序存储器



SAM E54 闪存配置为两个存储区：存储区 A 和存储区 B。两个存储区的开头均为自举程序，随后是应用程序映像，如上图所示。

默认情况下，存储区 A 映射到地址 0x00000000，存储区 B 映射到地址 0x00080000。映射到地址 0x00000000 的存储区称为活动存储区（默认为存储区 A），而映射到地址 0x00080000 的另一个存储区称为非活动存储区。

注： 映射到地址 0x00000000 的存储区称为活动存储区，因为 Cortex-M CPU 架构设计为从地址 0x00000000 运行起始指令。因此，复位时需要运行的代码需要映射到 0x00000000。

活动存储区中的自举程序可以接收以下升级请求：

- 升级地址 0x00002000 处的活动存储区应用程序映像。
- 升级地址 0x00082000 处的非活动存储区应用程序映像。
- 升级地址 0x00080000 处的非活动存储区合并映像（自举程序 + 应用程序）。

升级活动存储区应用程序映像

自举程序从主机接收应用程序映像，在成功执行升级后会通知主机应用程序。主机随后会发送一个复位命令以运行升级后的应用程序，示例如下：

- 通常在器件装入自举程序（出厂时）而非应用程序时，向自举程序发出该请求。当用户为器件的第一个应用程序映像升级请求选择该选项时，将现场发出该应用程序升级请求。
- 该选项可用于升级闪存中的某些元数据。要添加或升级的元数据只是应用程序映像的一小部分，因此更新元数据不需要对应用程序的整个存储区域进行升级。

升级非活动存储区应用程序映像或合并映像

从活动存储区运行的自举程序从主机接收应用程序映像或合并映像，在成功执行升级后会通知主机应用程序。主机随后会发送存储区交换和系统复位（BKSWRST）命令。BKSWRST 执行以下操作：

- 交换存储区以使非活动存储区处于活动状态，活动存储区处于非活动状态。存储区 A 处于非活动状态，而存储区 B 处于活动状态。
- 发出复位命令以运行升级后的应用程序。

闪存的特殊熔丝位中自带映射到闪存地址 0x00000000 的存储区的信息。这些熔丝位可以单独擦除或编程。当自举程序从主机接收到 BKSWRST 命令时，它会将闪存（NVM）控制寄存器中的 BKSWRST 位置 1。BKSWRST 置 1 时，闪存（NVM）控制器将交换存储区并根据熔丝位（STATUS.AFIRST）的最新状态设置熔丝位（STATUS.AFIRST），如下所示：

- STATUS.AFIRST = 0；存储区 B 的起始地址映射到 0x00000000
- STATUS.AFIRST = 1；存储区 A 的起始地址映射到 0x00000000

复位时，闪存（NVM）控制器将检查熔丝位（STATUS.AFIRST）的状态，并跳转到活动存储区以运行代码。

2.3 UART 自举程序协议

自举程序固件通过使用预定义的通信协议与个人计算机主机应用程序进行通信，以在 Harmony v3 自举程序框架和主机之间交换数据。

UART 自举程序协议包括保护、数据大小、命令和数据字节，如下图所示。

图 2-4. 自举程序协议

协议的详细信息如下：

- **GUARD**
  - 保护字节是一个常量值：**0x5048434D**
  - 该值可防止虚假命令
  - 自举程序始终在数据包接收开始时检查保护值，并相应地继续后续操作
- **数据大小**
  - 该字段指示要接收的数据字节数
  - 该值因命令而异
- **命令**
  - 指示要处理的命令。每个命令的宽度为一个字节
  - 支持下列命令：
    - Unlock（0xA0）
    - Data（0xA1）
    - Verify（0xA2）
    - Reset（0xA3）
    - Bank Swap and reset（0xA4）
- **数据**
  - 包含根据命令要处理的实际数据
  - 数据大小字段指示要接收的数据长度
  - 自举程序接收的数据大小以字（4 个字节）为单位
  - 所有数据字必须以小尾数法（LSB 在前）格式发送

响应代码

自举程序将发送一个单字符响应代码来响应每条命令。只能在接收到前一条命令的响应代码后或在 100 ms 超时（没有响应）后发送后续命令。

有效的响应代码如下：

- OK（0x50）——成功接收和处理命令
- Error（0x51）——处理命令期间出现错误
- Invalid（0x52）——接收到无效命令
- CRC OK（0x53）——CRC校验成功
- CRC Fail（0x54）——CRC校验失败

2.4 自举程序触发方法

可以使用以下方法调用自举程序：

- 在每次系统复位时或在器件中没有有效固件时运行自举程序。自举程序将在循环中持续等待从主机接收固件以进行升级。如果应用程序起始地址的第一个字是存储区 A（0x00002000），而存储区 B（0x00082000）不是 0xFFFFFFFF，则认为固件有效。通常，该字包含一个初始堆栈指针值，因此除非器件被擦除，否则第一个字始终不是 0xFFFFFFFF。系统复位时，自举程序会检查是否存在固件升级触发信号。如果没有有效的固件升级触发信号，自举程序将尝试运行现有固件。如果没有有效的固件，自举程序将跳转到一个循环中，等待从主机接收固件。
- 自举程序会提供 `bootloader_Trigger()` 函数，以便用户升级现有应用程序。`bootloader_Trigger()` 函数将检查开关按下事件或 SRAM 中的模式，以了解是否存在升级现有应用程序的请求。**`bootloader_Trigger()`** 函数的代码如下所示。该触发函数从自举程序系统初始化函数调用。

```
#define BTL_TRIGGER_PATTERN  0x5048434D

static uint32_t *ramStart = (uint32_t *)BTL_TRIGGER_RAM_START;

bool bootloader_Trigger(void)
{
    /* 检查 RAM 前 16 个字节中的自举程序触发模式以进入自举程序 */
    if (BTL_TRIGGER_PATTERN == ramStart[0] && BTL_TRIGGER_PATTERN == ramStart[1] &&
    BTL_TRIGGER_PATTERN == ramStart[2] && BTL_TRIGGER_PATTERN == ramStart[3])
    {
        ramStart[0] = 0;
        return true;
    }

    /* 检查开关按下事件以进入自举程序 */
    if (SWITCH_Get() == 0)
    {
        return true;
    }

    return false;
}
```

在应用程序运行时，可以使用以下方法来升级固件：

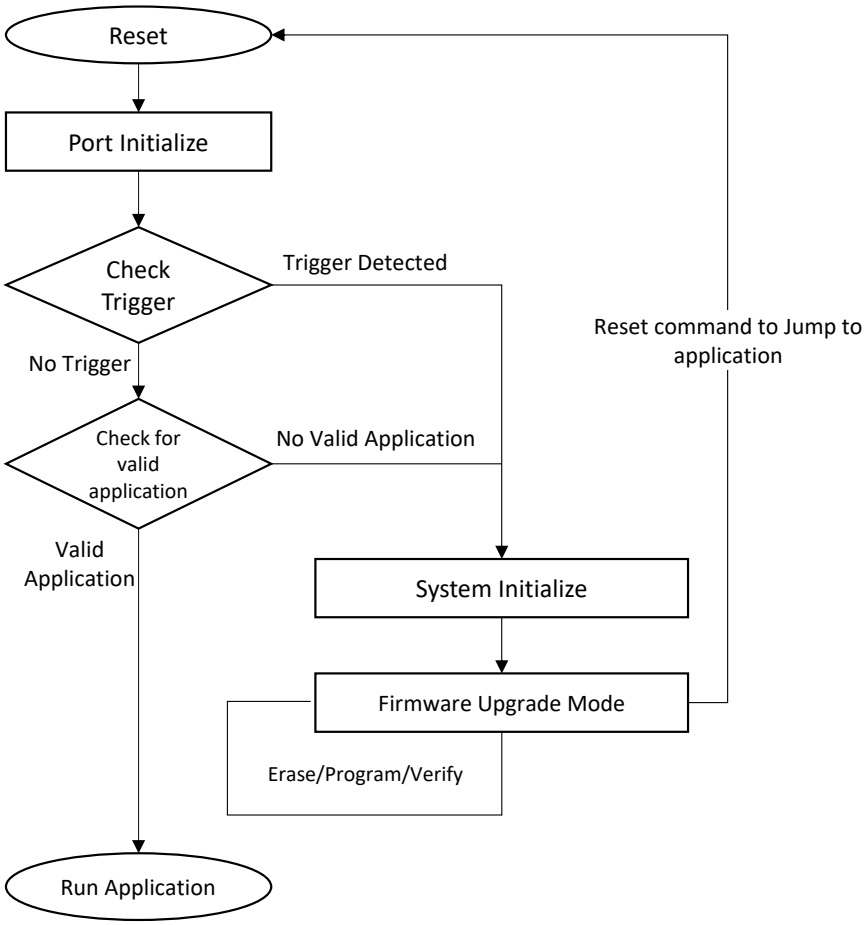
- 外部触发：在应用程序运行时，用户将同时按下外部系统复位开关和用户开关。器件将复位并开始运行自举程序。由于按下了用户开关，`bootloader_Trigger()` 函数将通过 `SWITCH_Get()` 函数检测到开关按下事件并返回 true，表示请求固件升级。自举程序负责从主机接收数据并升级器件。
- 软件应用程序触发：如果应用程序无法通过外部触发或应用程序要求根据特定命令升级固件，则可以使用软件触发方法来运行自举程序以进行固件升级。
- 应用程序实现 `invoke_bootloader()` 函数。如果应用程序想要在运行时自行升级，它将调用 `invoke_bootloader()` 函数并用已知模式（0x5048434D）填充 SRAM 中的专用区域，然后发出软件复位。这一预填充的 SRAM 模式将在 `bootloader_trigger()` 函数中进行比较，如果发生匹配，`bootloader_trigger()` 函数将返回 true，表示请求固件升级。自举程序负责从主机接收数据并升级器件。

应用程序可使用以下代码请求执行自举程序：

```
void invoke_bootloader(void)
{
    uint32_t *sram = (uint32_t *)BTL_TRIGGER_RAM_START;

    sram[0] = 0x5048434D;
    sram[1] = 0x5048434D;
    sram[2] = 0x5048434D;
    sram[3] = 0x5048434D;

    NVIC_SystemReset();
}
```

2.5 自举程序系统级执行流程

图 2-5. 自举程序系统级执行流程图



- 器件复位后，自举程序将初始化系统和端口并开始执行。
- 如果未从用户方接收到用于升级固件的有效触发信号，则自举程序将开始执行用户应用程序（如果已存在用户应用程序）。
- 如果触发信号有效，则自举程序将初始化系统，升级固件并发出复位 BKSWRST 命令以运行升级后的应用程序。

3. 配置双存储区自举程序

在 MPLAB Harmony v3 中，双存储区自举程序称为 UART 故障保护自举程序。它包含两个应用程序：

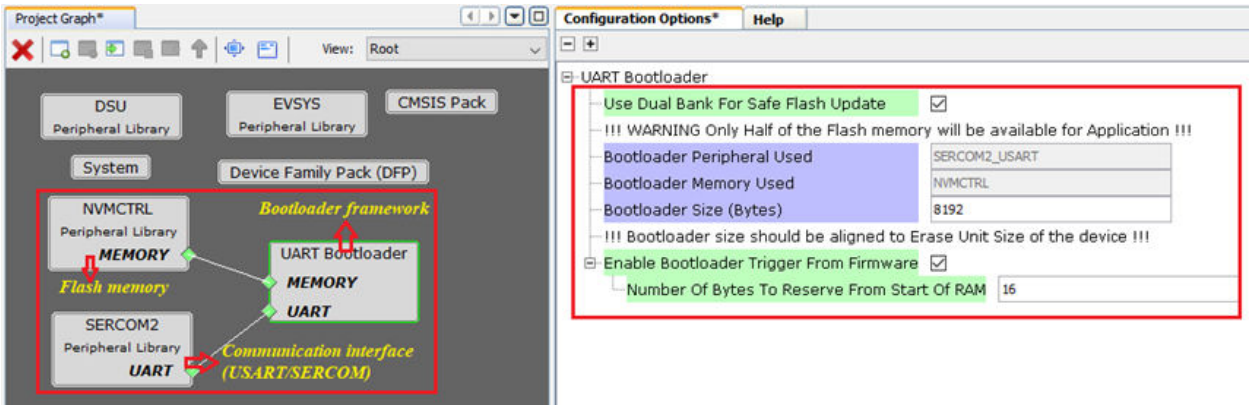
- 自举程序：uart_fail_safe_bootloader_sam_e54_xpro 是自举程序代码，用于执行固件升级。
- 测试应用程序：uart_fail_safe_bootloader_test_app_sam_e54_xpro 是用户应用程序代码。

注： 可以使用以下路径在自举程序 MPLAB Harmony v3 资源库中找到项目：*<Harmony framework>\bootloader\apps\uart_fail_safe_bootloader*

在 MPLAB Harmony v3 中配置自举程序库

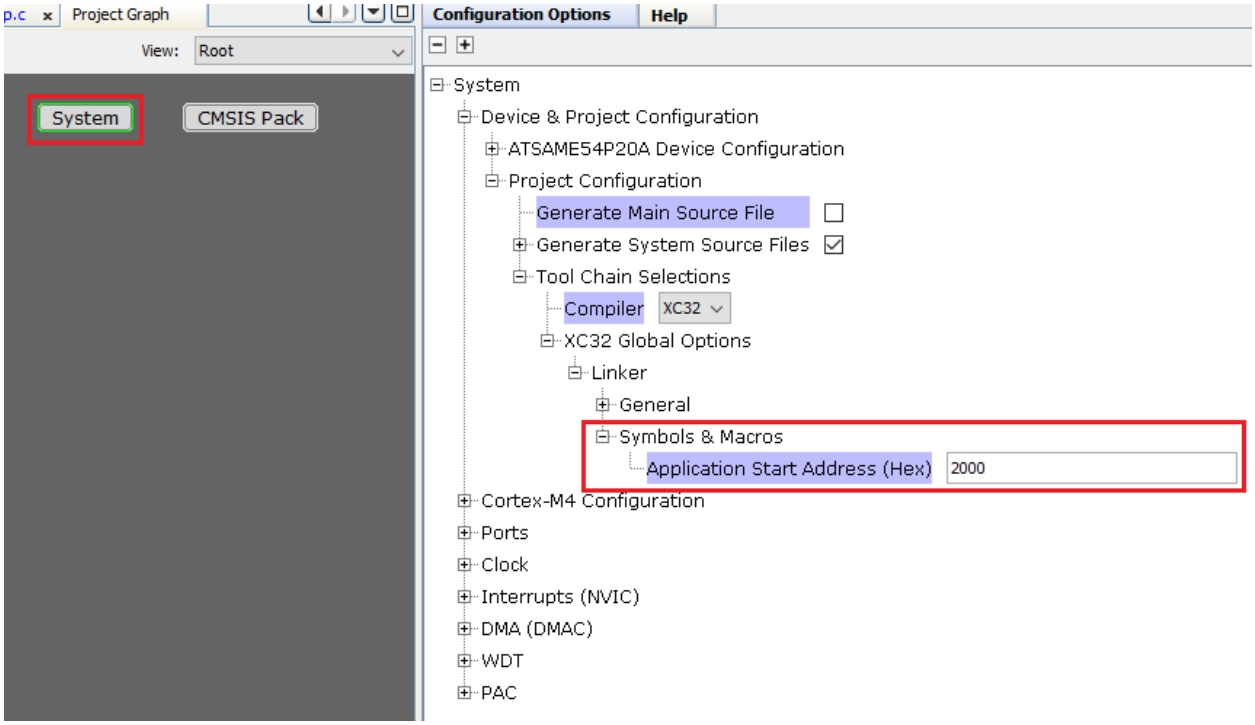
- **Use Dual Bank for Safe Flash Update（使用双存储区进行安全闪存更新）：**
  - 用于将自举程序配置为使用器件的双存储区上传应用程序
- **Bootloader Peripheral Used（使用的自举程序外设）：**
  - 指定自举程序在这种情况下用来接收应用程序的通信外设，串行通信（SERCOM）或 USART。
- **Bootloader Memory Used（使用的自举程序存储器）：**
  - 指定自举程序用来执行闪存操作的存储器外设
- **Bootloader Size (Bytes)（自举程序大小（字节））：**
  - 指定自举程序所需的最大闪存大小。
  - 该大小基于自举程序类型和使用的存储器计算。
  - 因器件而异，并且必须始终与器件擦除单元大小对齐。
- **Enable Bootloader Trigger from Firmware（使能从固件触发自举程序）：**
  - 该选项可用于在软复位后从应用程序固件强制触发自举程序
- **Number of Bytes to Reserve from Start of RAM（从 RAM 开头保留的字节数）：**
  - 该选项将提供的偏移量添加到自举程序链接描述文件中的 RAM 起始地址中
  - 应用程序固件可以将模式存储在 RAM 开头的保留字节区域，以供自举程序在复位时在 `bootloader_Trigger()` 函数中进行检查。

图 3-1. 自举程序配置



- **Application Start Address (Hex)（应用程序起始地址（十六进制））：**
  - 由自举程序编程的应用程序的起始地址。
  - 当用户配置自举程序大小时，MHC 将自动填充应用程序的起始地址，如自举程序配置所示。该值将等于自举程序大小（自举程序大小 = 8K（0x00002000））。
  - 器件复位时，自举程序将使用该值跳转到应用程序。

图 3-2. 应用程序起始地址配置



3.1 自举程序链接描述文件

自举程序库使用通过 MHC 生成的自定义链接描述文件（btl.ld）。MHC 生成指定的自举程序大小、ROM（只读存储器）和 RAM（随机访问存储器）地址，如下图中的红框内容所示。

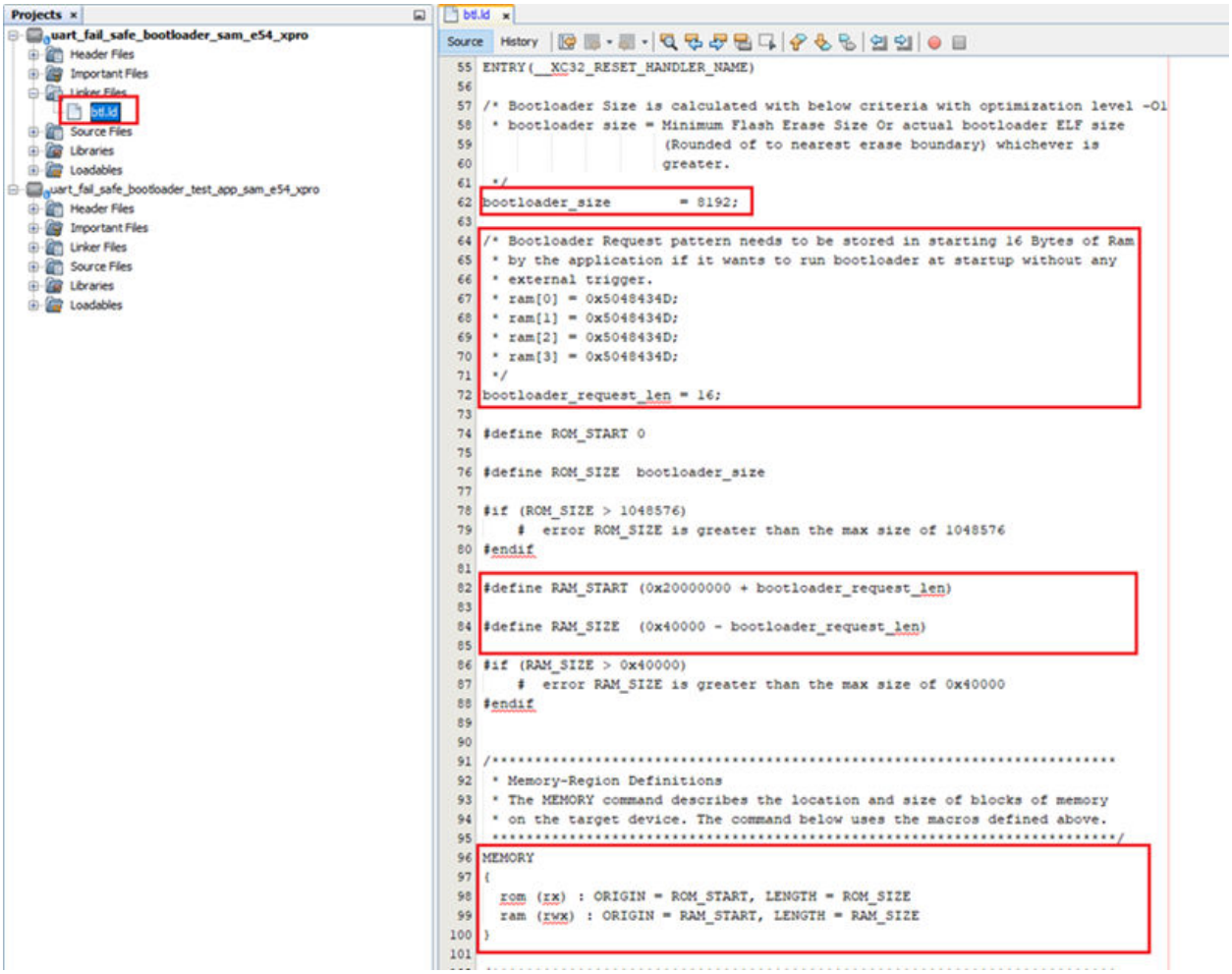
链接描述文件中填充的值基于 MHC 配置（自举程序配置）的自举程序组件。

配置链接描述文件，使自举程序从 RAM 运行，以便闪存写入操作和下一个数据块的接收操作同时进行。

如果应用程序想要在启动时不借助任何外部触发来运行自举程序，必须将自举程序请求模式存储在 RAM 开头的 16 个字节中，如下图所示。

即使在-O1 优化中自举程序的大小为 1672 字节，SAM E54 的自举程序大小也会四舍五入为最接近的擦除单元大小（8192 字节）。这有助于在自举程序上进一步添加其他功能，并避免应用程序与自举程序重叠。

图 3-3. 自举程序链接描述文件



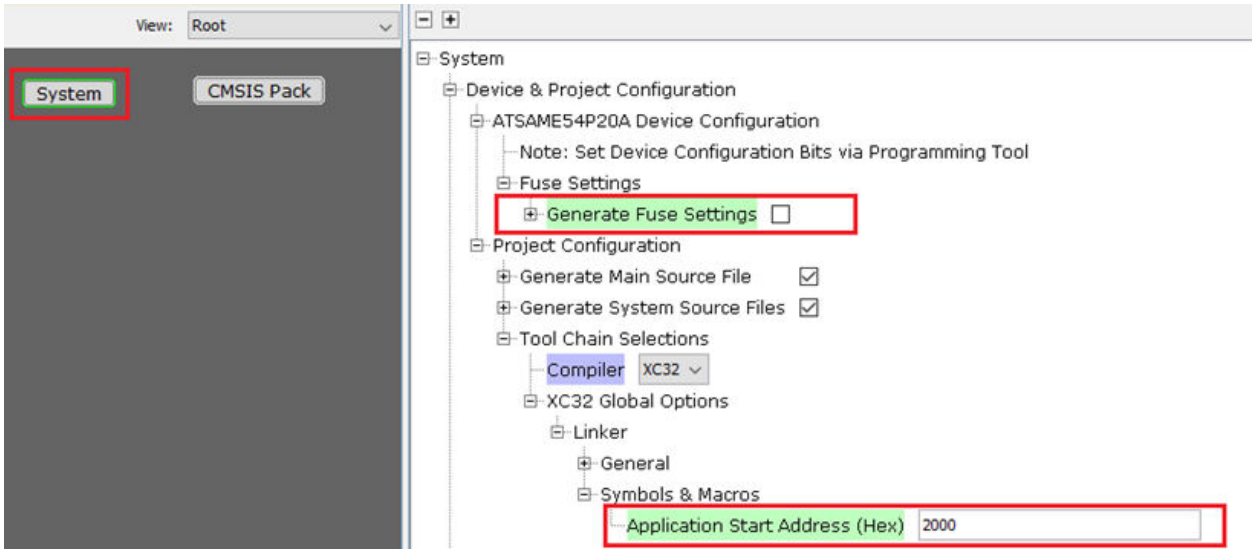
注： 用户必须确保用户应用程序的存储区不会与预留给自举程序的存储区重叠。

3.2 测试应用程序配置

- **禁止 Generate Fuse Settings（生成熔丝设置）：**
  通常，熔丝配置设置通过编程工具进行编程。在本文档中讨论的参考应用程序中，由于应用程序通过自举程序编程，因此禁止了熔丝设置。

  注： 熔丝设置不能通过固件编程。
  通过十六进制文件生成时，使能熔丝设置会增加二进制文件的大小。
- **Application Start Address (Hex)：**
  - 应用程序的起始地址。
  - 应用程序起始地址值必须等于或大于闪存基址 + 自举程序大小。
  - 器件复位时，自举程序将使用应用程序起始地址值跳转到应用程序。该值必须与生成过程中提供给自举程序代码的值相匹配，如应用程序起始地址配置所示。
  - 应用程序起始地址将用于生成 XC32 编译器设置，以将代码放置在预期地址，如下图和测试应用程序项目设置所示。

图 3-4. 测试应用程序配置

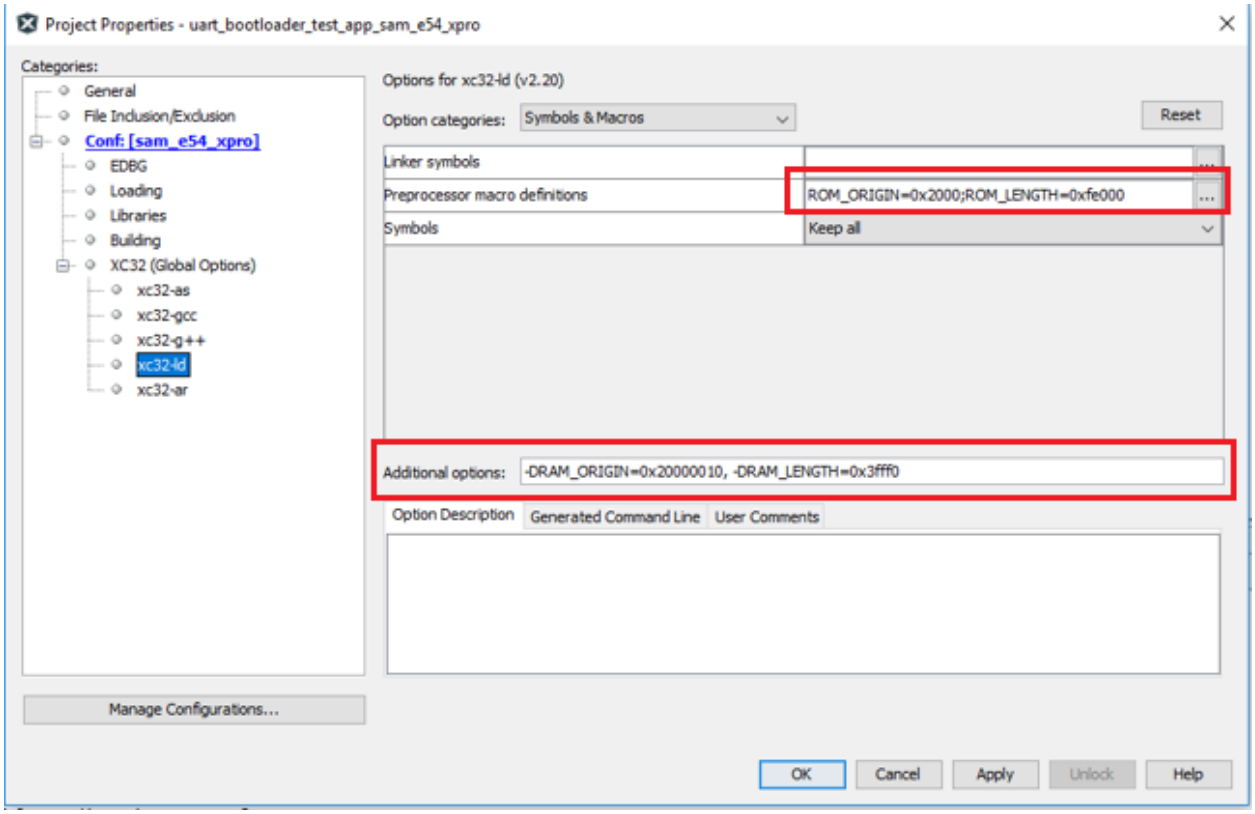


3.3 测试应用程序项目设置

- **Preprocessor Macro Definitions（预处理器宏定义）：**
  - ROM-ORIGIN 和 ROM_LENGTH 为 XC32 链接器变量，将由此处提供的值改写。
  - 重新生成后，将使用 MHC（自举程序链接描述文件）中提供的应用程序起始地址的值自动填充链接描述文件中的应用程序起始地址值。
- **Additional Options（其他选项）：**
  - 必须通过保留 RAM 开头 16 个字节来提供 RAM_ORIGIN 和 RAM_LENGTH 值，以从固件触发自举程序。
  - 这是可选的，如果不需要软触发自举程序，则可以忽略。

```
定制链接器选项：-DRAM_ORIGIN=0x20000010, -DRAM_LENGTH=0x3fff0
```

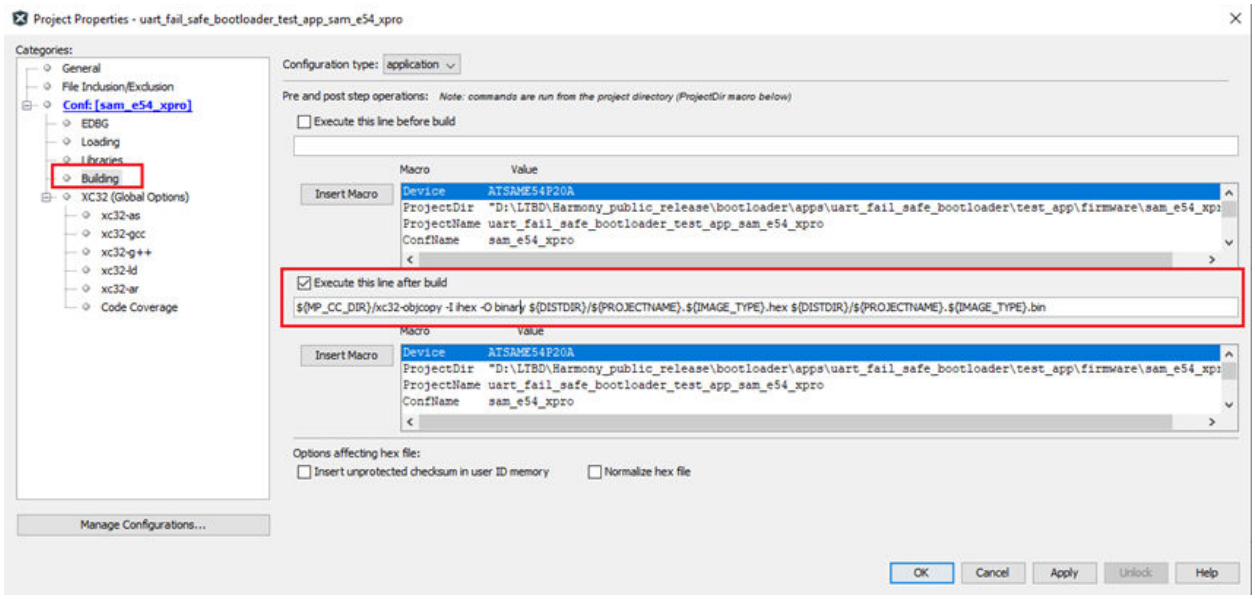
图 3-5. 测试应用程序项目设置



- **Execute this line after build（在编译后执行该代码行）：**
  - 该选项可用于在编译完成后自动从十六进制文件生成二进制文件

```
定制链接器选项：“${MP_CC_DIR}/xc32-objcopy -I ihex -O binary ${DISTDIR}/${PROJECTNAME}.${IMAGE_TYPE}.hex ${DISTDIR}/${PROJECTNAME}.${IMAGE_TYPE}.bin”
```

图 3-6. 测试应用程序二进制文件生成设置



4. 运行演示

4.1 运行自举程序应用程序

1. 将 micro USB 线缆连接到 SAM E54 Xplained Pro 板的 DEBUG 端口。
2. 使用 MPLAB X IDE 编译并编程 UART 故障保护自举程序（双存储区自举程序）。
3. 为 UART 故障保护自举程序应用程序启动 MHC。
   - 按照测试应用程序配置部分所示禁止 Fuse Settings（熔丝设置）。
   - 按照测试应用程序配置部分所示使能 MPLAB X 项目属性中的 Execute this line After Build（编译后执行该代码行）选项。
   - 重新生成代码。
4. 使用 MPLAB X IDE 再次编译自举程序应用程序（uart_fail_safe_bootloader_sam_e54_xpro）。
   - 生成自举程序应用程序的二进制文件需要该步骤。
5. 使用 MPLAB X IDE 编译测试应用程序（uart_fail_safe_bootloader_test_app_sam_e54_xpro），但不进行编程。
6. 从命令提示符处运行 `btl_app_merge_bin.py` 以合并生成的自举程序二进制文件和应用程序二进制文件。命令提示符处必定会显示以下输出。

```
Command: python <python script> -o <Offset> -b bootloader image -a application image

<Offset>: Application start address (E.g. - 0x00002000)
<python script>: btl_app_merge_bin.py

Example: python <harmony3_path>\bootloader\tools\btl_app_merge_bin.py -o 0x00002000 -b
<harmony3_path>\bootloader\apps\uart_fail_safe_bootloader\bootloader\firmware
\sam_e54_xpro.X\dist\sam_e54_xpro\production
\sam_e54_xpro.X.production.bin -a
<harmony3_path>\bootloader\apps\uart_fail_safe_bootloader\test_app\firmware
\sam_e54_xpro.X\dist\sam_e54_xpro\production
\sam_e54_xpro.X.production.bin
```

图 4-1. 自举程序和应用程序二进制文件合并输出

```
##### Merged Bootloader and Application binaries to  btl_app_merged.bin #####
```

注：运行帮助命令时会提供关于可用选项的简要概述，如下所示。

```
Command: python <python script> --help

<python script>: btl_app_merge_bin.py

Example: python <harmony3_path>\bootloader\tools\btl_app_merge_bin.py --help
```

图 4-2. 应用程序二进制文件合并帮助窗口

```
Usage: btl_app_merge_bin.py [options]

Options:
  -h, --help            show this help message and exit
  -v, --verbose         enable verbose output
  -b BTL_FILE, --btl_file=BTL_FILE
                        bootloader binary file to program
  -a APP_FILE, --app_file=APP_FILE
                        application binary file to program
  -o OFFS, --offset=OFFS
                        application start offset (default 0x2000)
  -d DEV, --device=DEV  target device (same5x/samd5x)
```

7. 从命令提示符处运行 btl_host.py 以将合并的二进制文件编程到另一个区域。合并的二进制文件 btl_app_merged.bin 将在执行 btl_app_merge_bin.py 的路径中生成。

```
Command: python <python script> -v -s -i <COM PORT> -d <Device Name> -a <Address> -f
<bootloader_application_merged_image>

<python script>: btl_host.py
<COM PORT>: Serial communication port
<Device Name>: SAME54
<Address>: Application start address (Bank A: 0x00002000 / Bank B: 0x00080000)

Example: python <harmony3_path>\bootloader\tools\btl_host.py -v -s -i COM18 -d same5x -a
0x00080000 -f btl_app_merged.bin
```

注：运行帮助命令时会提供关于可用选项的简要概述，如下所示。

```
Command: python <python script> --help

<python script>: btl_host.py

Command: python <harmony3_path>\bootloader\tools\btl_host.py --help
```

图 4-3. 应用程序自举程序主机帮助窗口

```
Usage: btl_host.py [options]

Options:
  -h, --help            show this help message and exit
  -v, --verbose         enable verbose output
  -r BAUD, --baud=BAUD  UART baudrate
  -t, --tune            auto-tune UART baudrate
  -i PATH, --interface=PATH
                        communication interface
  -f FILE, --file=FILE  binary file to program
  -a ADDR, --address=ADDR
                        destination address
  -p SectSize, --sectorSize=SectSize
                        Device Sector Size in Bytes
  -b, --boot            enable write to the bootloader area
  -s, --swap            swap banks after programming
  -d DEV, --device=DEV  target device (samc2x/samd1x/samd2x/samd5x/samda1/same
                        7x/same5x/samg5x/saml2x/samha1/pic32mk/pic32mx/pic32mz
                        )
```

8. 下图给出了固件编程的示例输出。

图 4-4. 固件升级输出

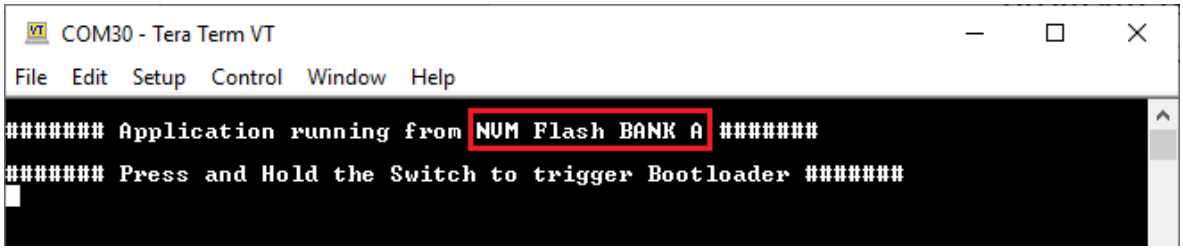
```
Unlocking

Uploading 4 blocks at address 524288 (0x80000)

Programming: |||||||||||||||||||||||||||||||||||||||||||||||||| 100.0% Complete

Verification

... success

Swapping Bank And Rebooting

Reboot Done
```

4.2 运行测试应用程序

1. 为 UART 故障保护自举程序应用程序执行运行自举程序应用程序步骤（如果尚未完成）。
2. 如果以上步骤成功，则 SAM E54 Xplained Pro 板上的 LED0 必定会开始闪烁。
3. 打开计算机上的终端应用程序（例如 Tera Term）。

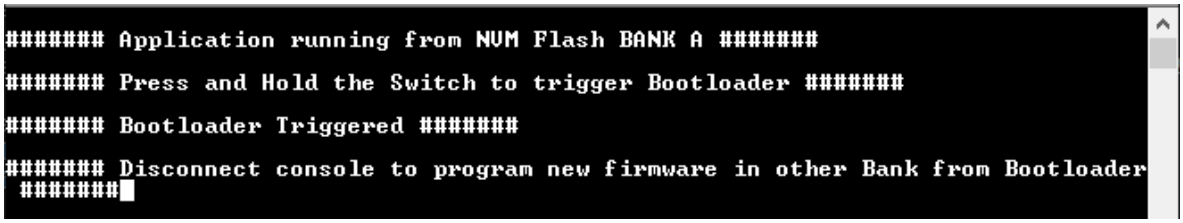
4. 配置串行端口设置，具体如下：
   - 波特率：115200
   - 数据：8 位
   - 奇偶校验：无
   - 停止：1 位
   - 流控制：无
5. 对器件进行复位或掉电再上电。
6. LED 必定会开始闪烁，并且控制台上将显示以下输出：
   - NVM 闪存存储区既可以是存储区 A 也可以是存储区 B，具体取决于程序的运行位置。

图 4-5. 在存储区 A 上运行的应用程序



7. 按住开关 SW0 触发自举程序对另一个存储区中的固件进行编程，控制台上将显示以下输出。

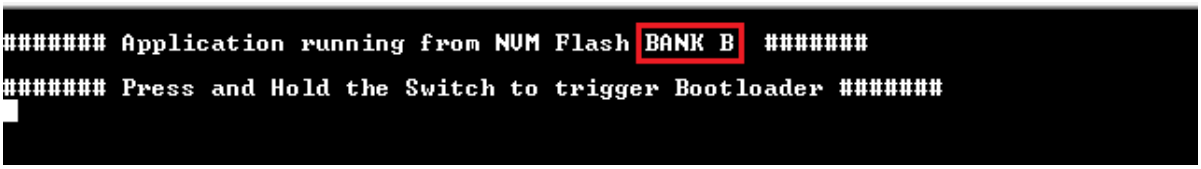
图 4-6. 被触发进入自举程序的应用程序



重复运行自举程序应用程序中的步骤 6 至 8 以切换到存储区。

- 该步骤用于验证从*测试应用程序*触发自举程序之后自举程序是否处于运行状态，以及在另一个存储区中编程新固件。
- 观察所显示*测试应用程序*控制台中的存储区与第一次运行相比的变化，如下图所示。

图 4-7. 在存储区 B 上运行的应用程序



5. 参考资料

- 有关自举程序的详细说明，请参见：
  <Harmony path>\bootloader\doc\help_bootloader.chm
- MPLAB Harmony GitHub：
  github.com/Microchip-MPLAB-Harmony
- 如何设置 MPLAB Harmony v3 软件开发框架：
  https://www.microchip.com/mymicrochip/filehandler.aspx?ddocname=en1000821
- SAM D5x/E5x MCU 的 Harmony v3 外设库入门：
  microchipdeveloper.com/harmony3:same54-getting-started-training-module
- Harmony v3 登录页面：https://www.microchip.com/mplab/mplab-harmony
- SAM E5x（Cortex® M4）器件上的时钟系统配置和使用：
  http://www.microchip.com.cn/newcommunity//Uploads/202002/5e4284b98ba04.pdf
- Harmony 自举程序资源库：
  https://github.com/Microchip-MPLAB-Harmony/bootloader

Microchip 网站

Microchip 网站（www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

商标

ISBN： 978-1-5224-7348-0

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820