PIC32CM LS60 MCU 上的安全固件更新

AN5999

产品页链接

简介

现代嵌入式系统常常需要进行固件升级，以修复问题或增添功能。与此同时，保护知识产权也至关重要。尽管单片机配备了强大的固件安全功能，但固件在从外部源传输的过程中，依然存在被截获的风险。解决这一问题的方法是实现安全的固件升级流程，并确保仅对外发布加密后的固件版本。

本文档概述了使用 PIC32CM LS60 Curiosity Pro 评估工具包安全升级固件的过程。安全固件升级机制在主机端实现，而安全自举程序则利用板上 ATECC608B 加密协处理器在客户端实现。主机评估板会为应用程序生成摘要和签名，使用对称密钥对应用程序进行加密，并通过 UART 接口将其传输至客户端。客户端收到加密映像后先进行解密，然后使用签名和主机的公钥对应用程序进行验证。如果验证成功，则系统将执行新固件。

目录

1. 硬件和软件要求

1.1. PIC32CM LS60 Curiosity Pro 评估工具包

PIC32CM LS60 Curiosity Pro 评估工具包非常适合用于对安全、超低功耗的 PIC32CM LS60 Arm® Cortex®-M23 单片机进行评估和原型设计。该 MCU 将 TrustFLEX ATECC608B 安全子系统和 Arm TrustZone®技术集成在单一封装中。此外，还具有增强型外设触摸控制器（Peripheral Touch Controller，PTC）以及智能模拟功能，例如运放、ADC、DAC 和模拟比较器。

该评估工具包配有 Arduino® Uno 连接器、microBUS™插座和扩展插座，便于进行扩展开发。其板上内置嵌入式调试器，无需借助外部工具即可进行编程或调试。此外，还支持嵌入式电流测量（XAM），并可通过数据可视化器进行实时功率测量。

PIC32CM LS60 MCU 的主要特性如下：

- 48 MHz Arm Cortex-M23 内核
- 512 KB 闪存和 64 KB SRAM
- 安全引导和加密加速器

PIC32CM LS60 Curiosity Pro 评估工具包可从 Microchip 直销网站获取。

1.2. MPLAB® X 集成开发环境和 MPLAB XC32 编译器

MPLAB X 集成开发环境（Integrated Development Environment，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，可用于发现、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。本文档介绍了 MPLAB X IDE 版本 6.25 或更高版本。
- MPLAB XC32 编译器可从 Microchip 网站获取。本文档介绍了 MPLAB XC32 版本 4.60 或更高版本。

1.3. MPLAB Harmony v3

MPLAB Harmony v3 是一款完全集成的软件开发框架，能够提供灵活且可互操作的软件模块，可将资源专注于创建 32 位 PIC®和 SAM 器件应用程序的开发，而无需处理器件详细信息、复杂协议和库集成等挑战。

它包括 MPLAB 代码配置器（MPLAB Code Configurator，MCC），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。MCC 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。

本文档中讨论的应用程序使用以下 MPLAB Harmony v3 资源库。这些资源库可从 GitHub 下载：

- csp v3.22.2 或更高版本（MPLAB Harmony v3 芯片支持包）
- bsp v3.20.1 或更高版本（MPLAB Harmony v3 板级支持包）
- bootloader v3.7.0 或更高版本
- cryptoauthlib v20250217 或更高版本（MPLAB Harmony v3 加密身份验证库）
  或
- 使用 MCC 内容管理器下载上述资源库

1.4. Python

该应用程序使用 Python v3.10 脚本将目标固件从 PC 传输至主机板。

2. 安全

增强系统安全性需要实现以下几个关键特性：

- 机密性
- 完整性
- 身份验证

机密性确保敏感数据对未经授权的个人或器件保持不可访问状态。固件开发人员的主要目标是保护应用程序免遭竞争对手未经授权的访问或复制，从而维护代码的机密性。其目的是将代码的访问权限限定在目标器件上，这些器件是惟一被授权使用该代码的实体。

单片机通常配备安全功能，以防止恶意实体访问存储在器件内的程序代码。但是，当需要现场更新固件时，制造商必须将更新后的固件分发给客户，使其能够自行更新器件。该过程可能会让固件被逆向工程，从而暴露原始代码。

身份验证用于验证固件的来源，确认其是由授权制造商发布，而非来自未授权实体。器件固件的可更新性带来了安装未授权软件的风险，这些软件可能由怀有恶意目的的第三方制作，旨在绕过安全协议或利用器件功能。

此外，还存在合法固件被滥用于非目标器件的风险，例如未经授权的硬件克隆品或为攻击而设计的器件。这构成了身份验证方面的另一个关注点，此次是针对目标器件本身。

完整性对于检测数据的任何更改至关重要。例如，一个经授权的固件可能被以某种方式篡改，使其看起来仍然真实可信，但它却可能促成类似于前面提到的攻击。因此，确保数据完整性对于维护系统的整体安全性至关重要。

缺乏安全措施的固件容易受到攻击，导致其机密性、完整性和真实性受到破坏。因此，必须实施策略来加强这三个关键领域。可通过多种方法来确保固件免遭未授权用户蓄意更改。这些方法将在后续章节中概述。

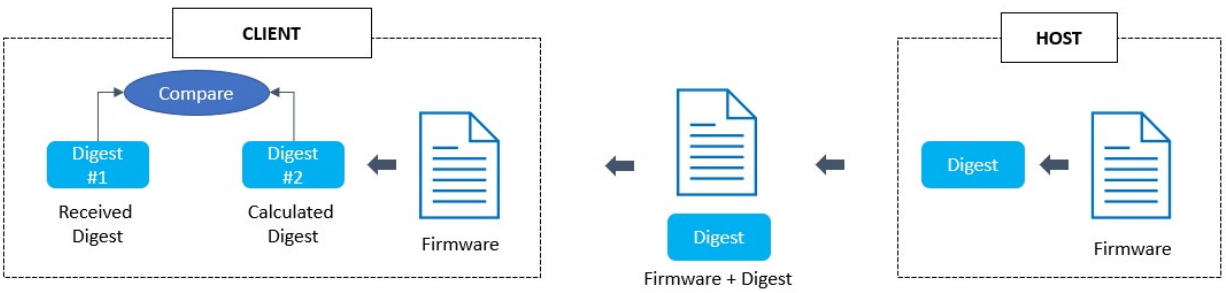
2.1. 哈希函数

哈希函数的主要目标是为给定的数据集生成惟一的数字标识符，类似于“指纹”。与错误检测码不同，每个数据集都必须与惟一的标识符相关联。

为确保固件的完整性，需计算其惟一标识符并附加到文件中。当收到固件及其相关标识符后，自举程序会重新计算标识符并将其与原始标识符进行比较，以确认固件未被修改。

在操作上，哈希函数接受可变长度的输入并生成预定大小的输出（称为报文摘要）。报文摘要具有几个关键属性，例如优秀的扩散性，可确保即使输入发生最微小的变化，输出也会产生显著差异。

图 2-1. 固件哈希



尽管哈希函数的输出大小固定，这在理论上限制了为每个可能的数据生成惟一摘要的可能性，但其设计使得找到两个能产生相同摘要的不同报文变得极其困难，从而在实际应用中有效模拟了惟一性。

仅依赖哈希函数来保护固件的一个局限在于，任何人都可以执行哈希计算，潜在的攻击者可能更改文件并重新计算哈希值，从而使自举程序无法检测到修改。

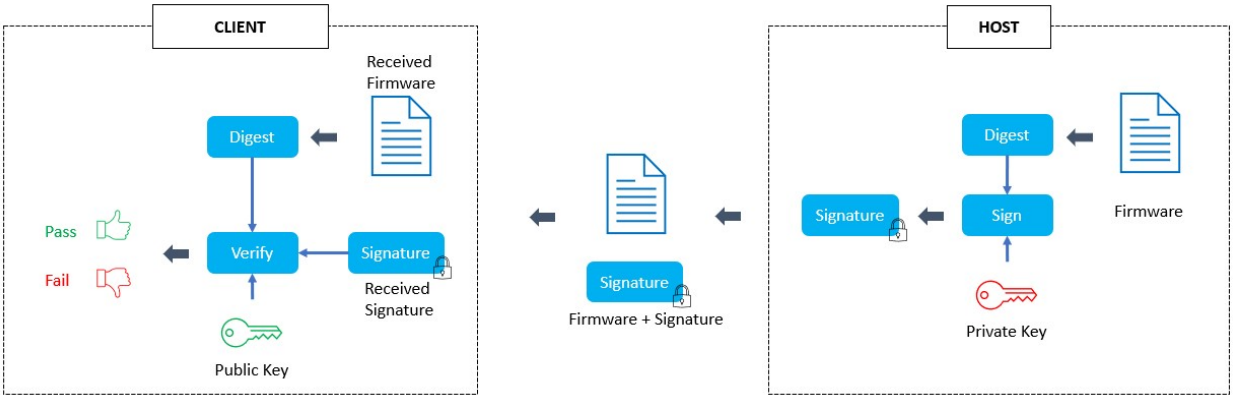
但是，使用哈希函数仍然可以有效地在运行时验证固件完整性并防止执行受损应用程序。

2.2. 数字签名

为通过可验证的真实性增强安全性，建议对哈希值进行加密，这构成了数字签名的基础。在此过程中，首先通过哈希函数生成固件摘要，随后使用公钥密码技术对其进行加密，最终生成一个数字签名（类似于日常交易中使用的传统签名）。

公钥加密或非对称加密使用一对密钥来提升安全性。主机使用私钥（该密钥始终保密）对签名进行加密，而客户端则使用对应的公钥对其进行解密。

图 2-2. 数字签名的创建与验证



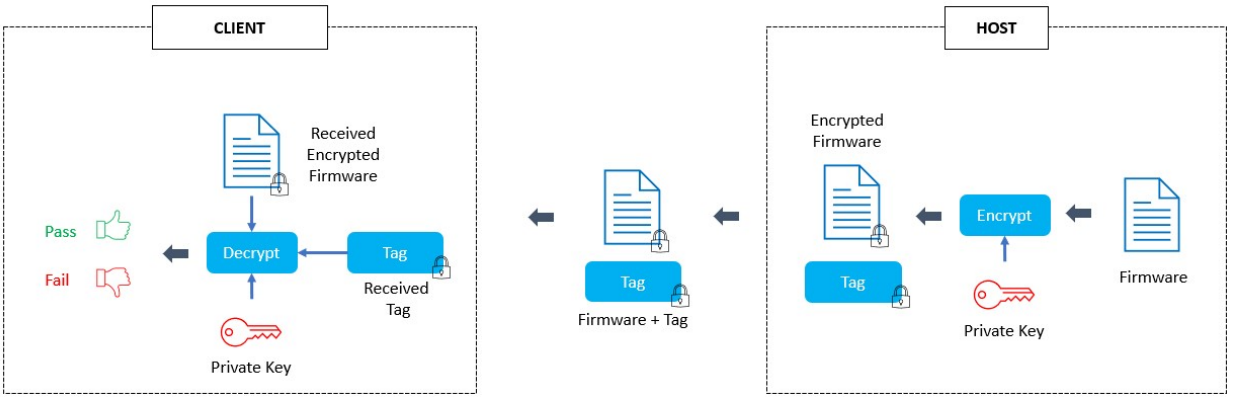
私钥加密数据的独占性确保了只有主机能够创建签名，从而有效防止未授权用户重现前文所述的攻击。但是，任何人都可以使用制造商的公钥确认签名的真实性。

2.3. 加密和解密

数据隐私通过加密技术来保障。数据通过加密算法和加密密钥进行处理，生成与（明文）原始数据不同的密文。没有对应的解密密钥，数据将呈现为无意义的乱码，从而防止任何未授权者读取。

实践中，通常使用私钥算法来生成加密的固件。不能使用公钥系统，否则任何人都可以解密固件。因此，加密和解密使用相同的密钥，并由自举程序与主机共享。

图 2-3. 固件加密和解密



仅靠代码加密本身并不能解决所有安全问题。例如，固件仍可能被修改（尽管难度很大）。攻击者可能设法定位代码中重要变量的位置并进行调整，直到获得期望的结果。

身份验证标记（也称为 MAC 标记）是附加到报文的一小段数据，用于验证报文的完整性和真实性。它确保报文未被篡改，且确实来自声称的源。

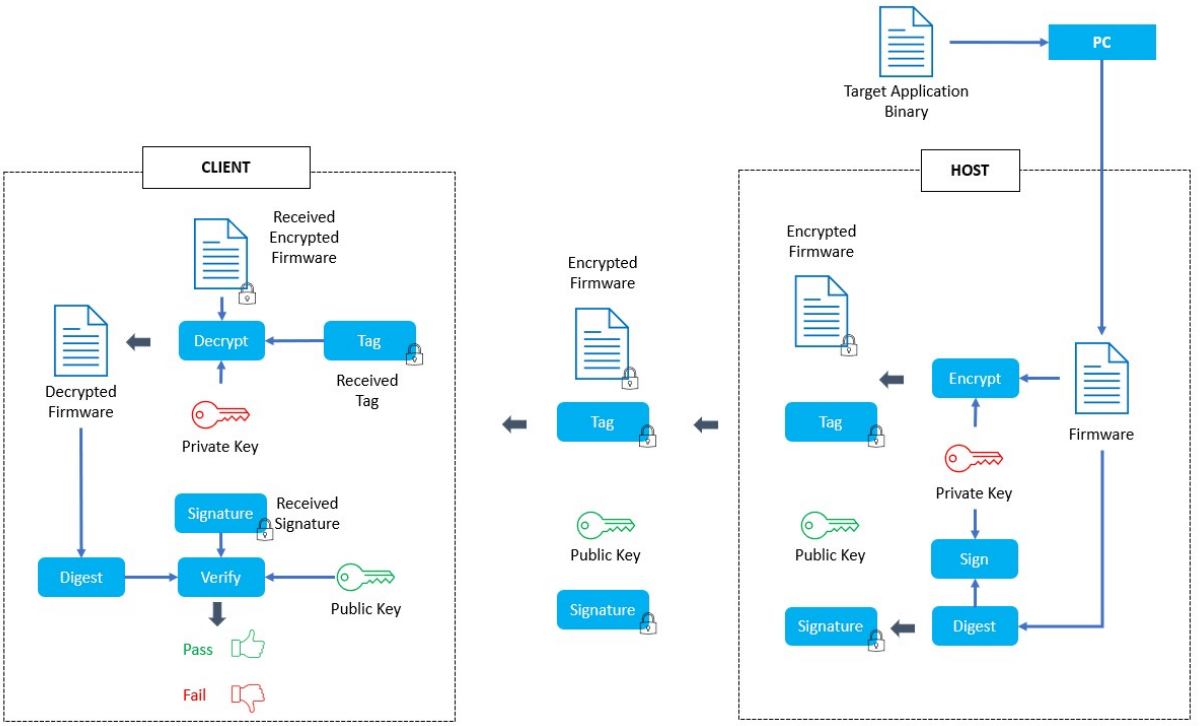
身份验证标记与加密能自然结合，尤其是在使用对称加密算法的系统中。由于加密和身份验证可以依赖同一底层算法，这不仅能缩短代码长度，还能简化实现。安全加密模式（如 AES-GCM（Galois/计数器模式））有效地结合了这两个过程，使用单一密钥和密码在一个统一的操作中同时提供机密性和完整性。

2.4. PIC32CM LS60 MCU 上的安全固件升级实现

固件更新的安全性依赖于多种加密要素，如哈希函数、数字签名算法和加密协议。利用 PIC32CM LS60 MCU 中内置的安全元件（ATECC608B），可以有效地集成这些安全功能。

下图展示了 PIC32CM LS60 MCU 与主机 PC 之间进行的安全固件升级。

图 2-4. PIC32CM LS60 MCU 上的安全固件升级



对于主机，按照以下步骤启动安全固件升级过程：

- 使用私钥对目标固件进行加密。
- 使用同一私钥对身份验证标记进行加密。
- 为了在客户端对目标固件进行身份验证，应用哈希函数来生成固件摘要。
- 然后使用私钥对固件摘要进行签名，为固件生成数字签名。
- 基于私钥派生出公钥，以便对固件进行身份验证。

注：加密后的固件会根据客户端的页大小要求被分割成更小的片段。

对于客户端，按照以下步骤执行安全固件升级：

- 使用私钥对加密后的固件进行解密，这通过解密后的身份验证标记进行验证。
- 将解密的映像编程到目标闪存中。
- 为收到的固件计算摘要。

- 结合使用公钥、收到的签名以及计算出的摘要来验证固件的完整性和真实性。

3. 设计

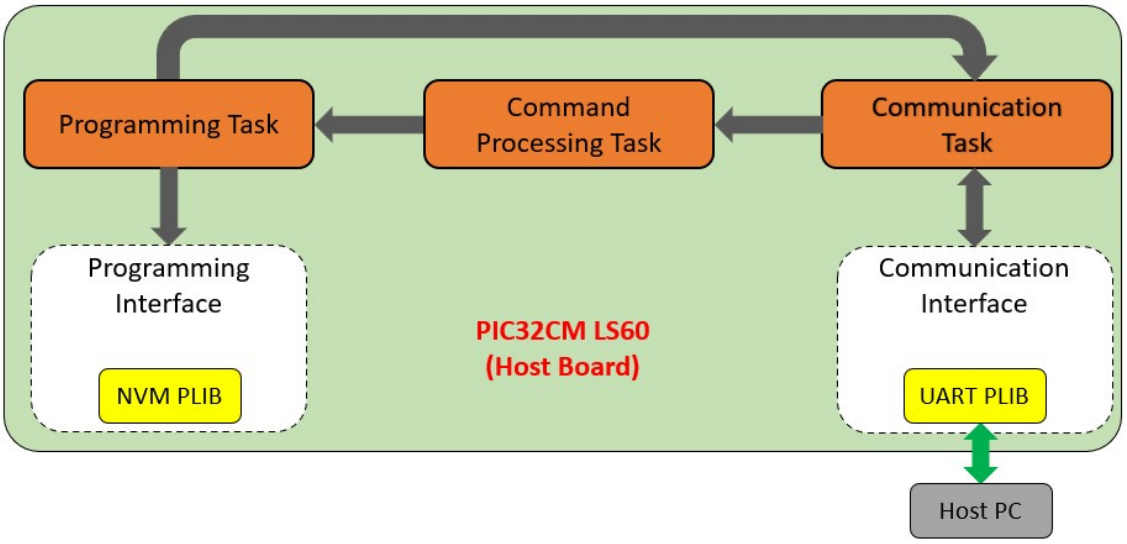
PIC32CM LS60 MCU 的安全固件升级应用程序利用其内置的安全元件 ATECC608B 来确保应用程序映像安全传输至目标器件。主机和客户端均利用安全元件的功能对应用程序映像进行加密、解密和身份验证，从而保障固件升级过程的完整性和安全性。

安全固件升级应用程序设计包括以下两种模式：

- 通过 PC 在主机上进行客户端固件升级：MCU 主机使用预定义的通信协议（具体为 UART 固件升级协议）与个人计算机应用程序交互，以便将客户端固件传输至 MCU 主机。
  客户端固件升级任务包括以下子任务：
  - 通信
  - 命令处理
  - 编程

  下图展示了通过 PC 在主机上进行客户端固件升级的应用程序设计。

  图 3-1. 通过 PC 在主机上进行客户端固件升级的应用程序设计



- 通过主机在客户端进行安全固件升级：将指定的固件映像安全存储在其安全闪存中。使用预定义的通信协议（详见安全固件升级协议）向客户端传输加密数据及相关标记。然后使用私钥对固件映像进行数字签名，该签名可供客户端验证传入固件的真实性。通过身份验证后，主机发送复位命令以启动新固件运行。
  主机应用程序包括以下子任务：
  - 命令发起
  - 映像读取
  - 映像加密
  - 通信

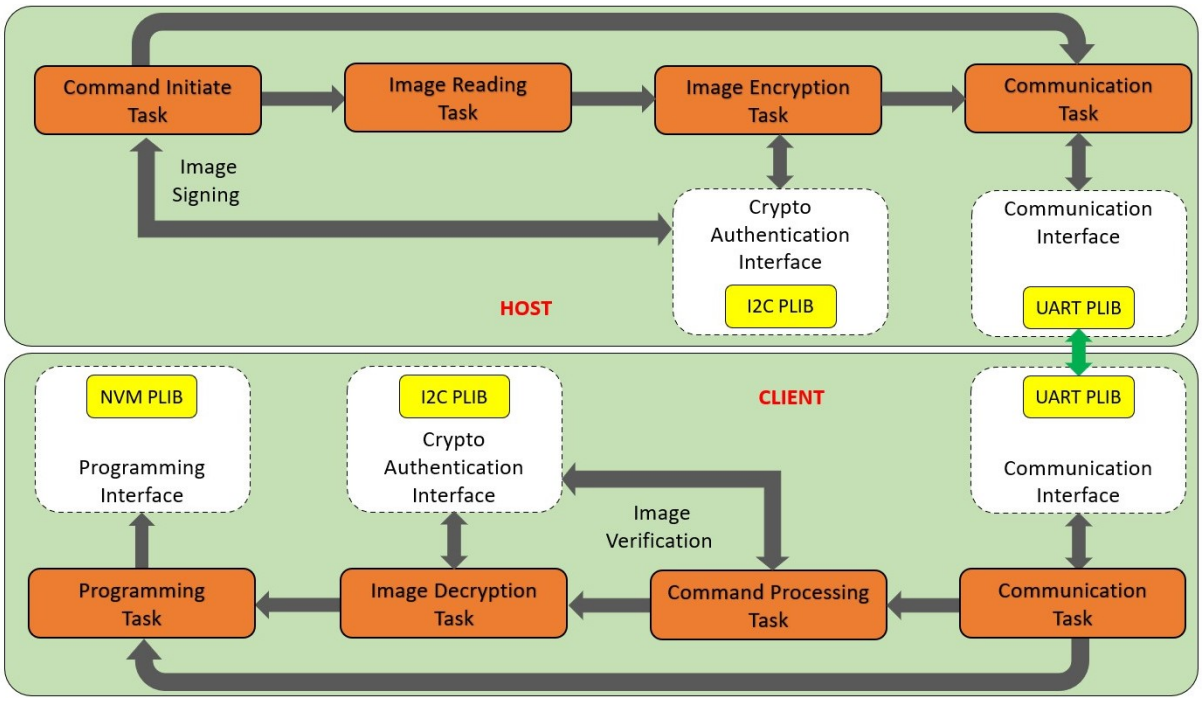
  客户端：使用预定义的通信协议（在安全固件升级协议中定义）从主机获取加密的固件映像。接收后，客户端对数据进行解密，并在成功验证标记后将其编程到闪存中。在完整固件完成编程后，客户端计算签名并执行验证过程。

客户端应用程序包括以下子任务：

- 通信
- 命令处理
- 映像解密
- 编程

下图展示了安全固件升级的应用程序设计。

图 3-2. 通过主机在客户端进行安全固件升级的应用程序设计

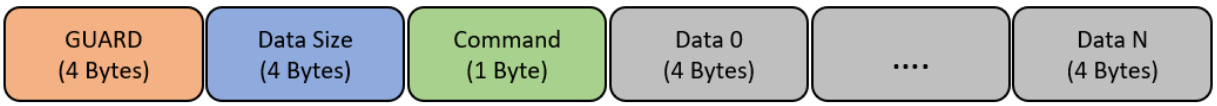


3.1. UART 固件升级协议

UART 固件升级应用程序使用预定义的通信协议在 PC 与主机 MCU 之间交换数据。

该协议包括保护、数据大小、命令和数据字节，如下图所示。

图 3-3. UART 固件升级命令



- 保护
  - 保护字节是一个常量值 0x5048434D。
  - 该值可防止虚假命令。
  - 主机 MCU 固件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。每条命令的宽度为 1 字节。
  - 支持的命令如下：
    - Unlock (0xA0)
    - Data (0xA1)
    - Verify (0xA2)
    - Reset (0xA3)
    - Device Configuration (0xA5)
    - Firmware Version (0xA6)
- 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。
  - 主机 MCU 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

发送命令后，主机 MCU 将以单字符或双字符代码进行响应。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)
- Error (0x51)
- Invalid (0x52)
- CRC OK (0x53)
- CRC Failed (0x54)

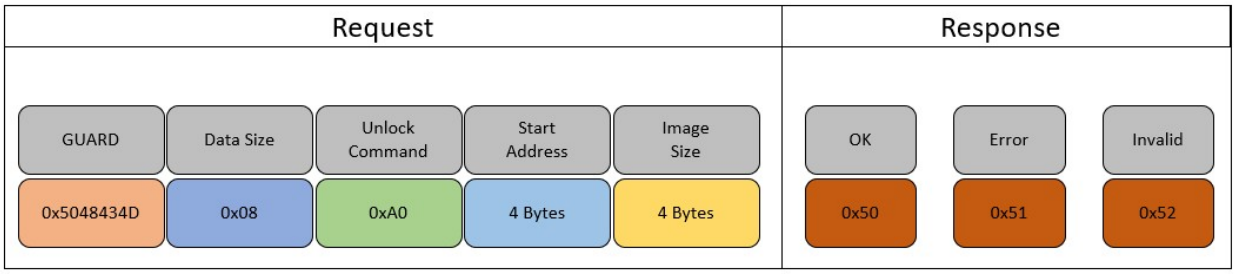
3.1.1. 命令说明

本节给出了 UART 固件升级协议中使用的请求和响应代码的详细说明。

Unlock 命令

Unlock 命令序列如下图所示，其中包含相应的响应。

图 3-4. UART 固件升级 Unlock 命令

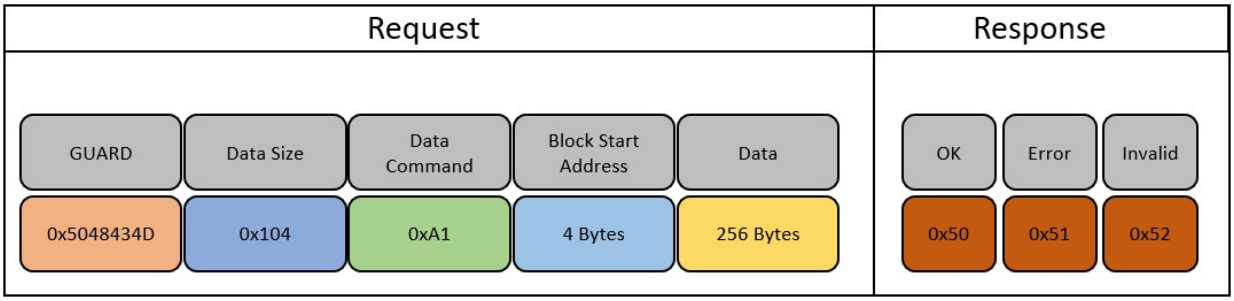


- Unlock 命令必须在第一个 Data 命令发出之前发出
  - 该命令用于计算应用程序的起始地址和结束地址。
  - 该信息将用于验证发送的地址是否处于闪存范围内。
  - 该命令将用于验证要编程的数据包随附的地址是否处于调用 Unlock 命令的区域内。
- 要接收的数据字节数为 8 个字节（起始地址 + 映像大小）。
- 起始地址：
  - 闪存的应用程序起始地址。
  - 该地址取决于器件，应始终大于或等于自举程序的结束地址。
  - 必须在擦除单元大小边界处对齐，该边界取决于器件。
  - 要自行升级自举程序，该值必须设置为 0。
- 映像大小必须以擦除单元字节为增量，具体取决于器件。

Data 命令

Data 命令序列如下图所示，其中包含相应的响应。

图 3-5. UART 固件升级 Data 命令

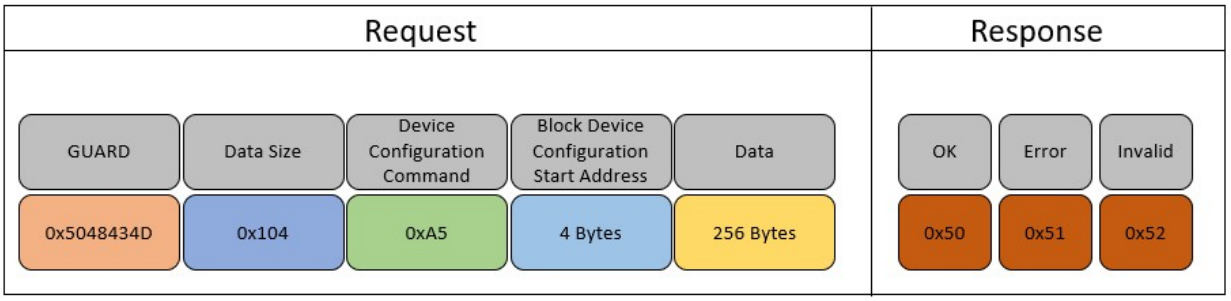


- Data 命令用于传输映像数据。
- 总数据大小包括块起始地址（4 字节）和数据（256 字节）。
- 块起始地址必须位于使用 Unlock 命令解锁的区域内。
- 如果尝试在解锁区域的边界以外执行写操作，则将导致错误，提供的数据将被丢弃。

Device Configuration 命令

带相应响应的 Device Configuration 命令序列如下图所示。

图 3-6. UART 固件升级 Device Configuration 命令

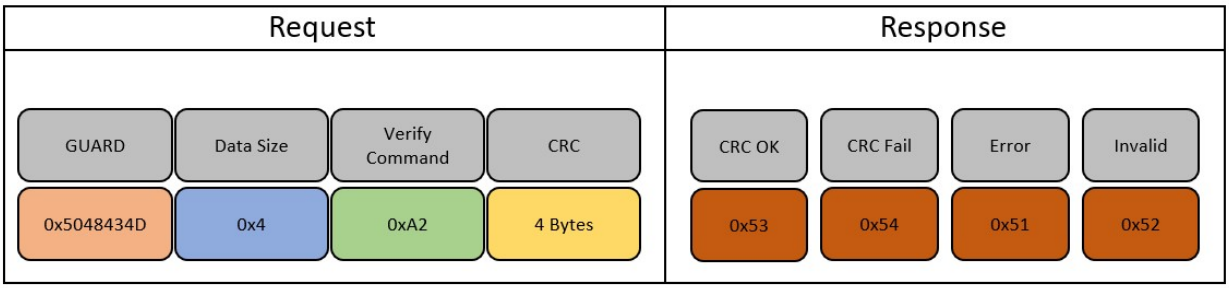


- 该命令仅在为客户端固件选择使能熔丝编程功能时才使能。
- Device Configuration 命令用于发送器件配置位（熔丝设置），数据大小等于器件配置区域起始地址（4 字节）和器件配置页大小（256 字节）的总和，具体取决于器件。
- 器件配置起始地址应为器件配置区域的起始地址。
- 器件配置数据应包含适用于器件的所有熔丝设置。不支持部分熔丝位编程。
- 如果尝试请求在器件配置区域以外执行写操作，则将导致错误，提供的数据将被丢弃。

Verify 命令

带相应响应的 Verify 命令序列如下图所示。

图 3-7. UART 固件升级 Verify 命令

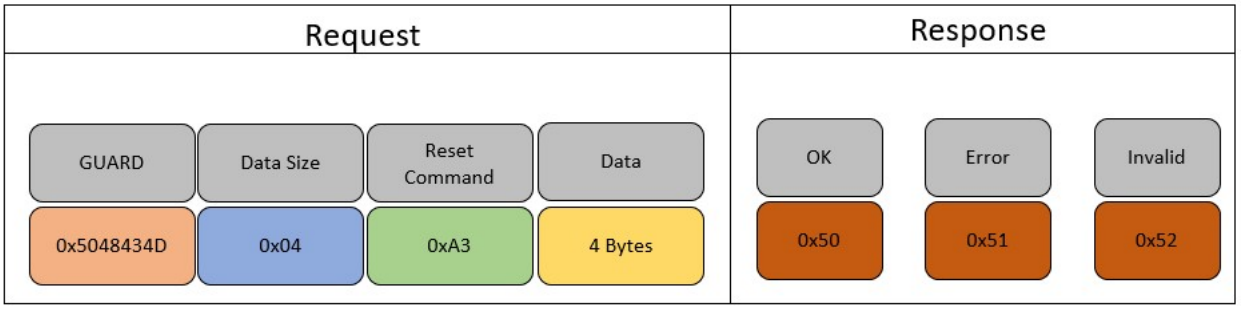


- Verify 命令用于确认传输和编程的映像数据的完整性。
- 映像 CRC 是标准 IEEE® CRC32，其多项式为 0xEDB88320。
- 编程后，会根据从闪存中读取的值计算内部 CRC，因此该命令会进行全面验证。
- 映像 CRC 计算会在之前解锁的区域上进行。

Reset 命令

带相应响应的 Reset 命令序列如下图所示。

图 3-8. UART 固件升级 Reset 命令



- Reset 命令用于从自举程序切换到应用程序的执行。
- 需补充额外的字节以满足协议结构的要求。
- 该功能在主机无法直接控制复位引脚的情况下至关重要。此外，即使在主机能够控制复位引脚时，该命令仍具实用价值。

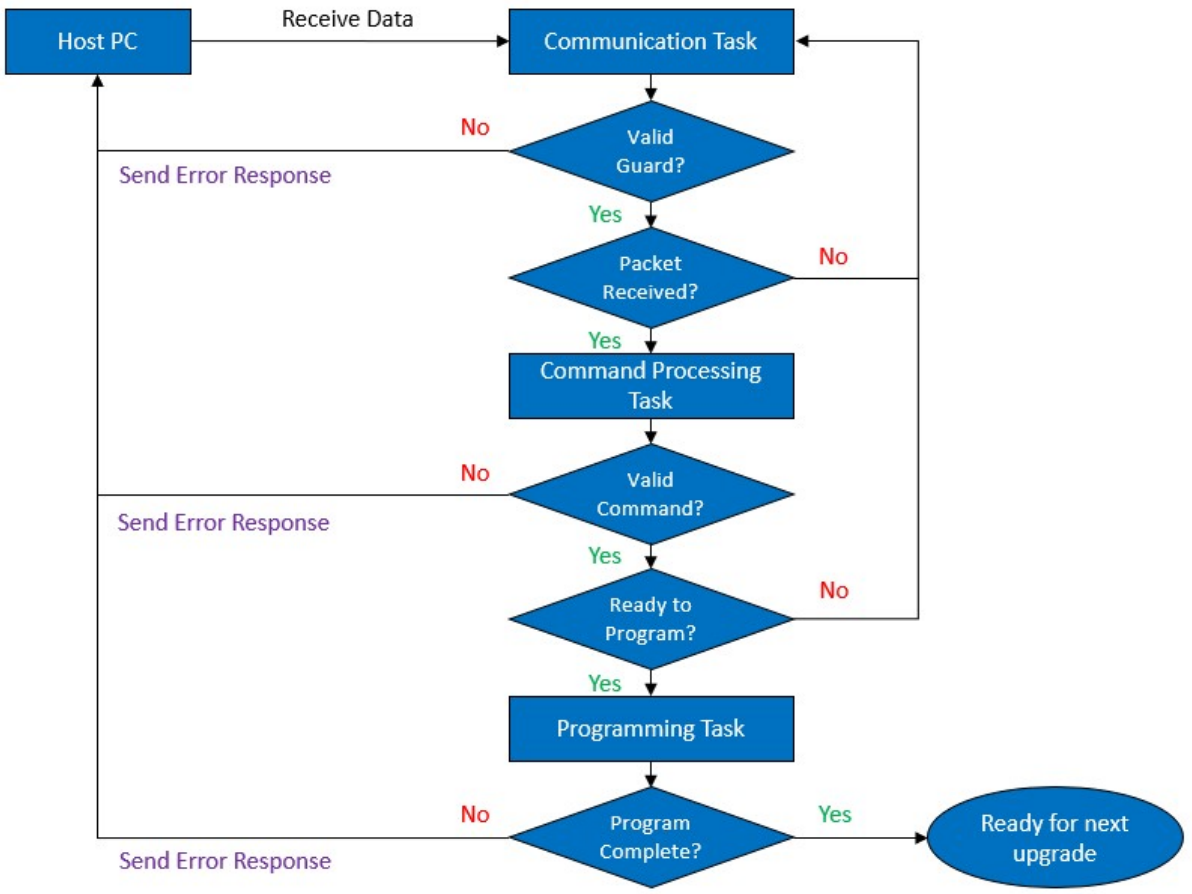
3.1.2. 任务说明

本节给出了 UART 固件升级模式中使用的任务的详细说明。

- 通信任务——负责在中断模式下通过选定的通信接口从主机 PC 接收数据。在将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——处理从通信任务接收的命令并作出响应。这会相应地对主机 PC 作出响应。如果接收到的命令与编程相关，则该任务会将控制权委托给编程任务。
- 编程任务——负责使用接收到的数据包对内部闪存进行编程。该任务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，该任务并行调用通信任务以接收下一个数据包。

下图给出了固件升级执行流程图。

图 3-9. 主机上的客户端固件升级执行流程图

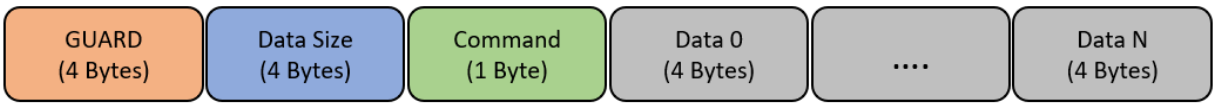


3.2. 安全固件升级协议

安全固件升级应用程序使用预定义的通信协议在客户端与主机之间交换数据。

基于 UART 的安全固件升级协议包括保护、数据大小、命令和数据字节，如下图所示。

图 3-10. 安全固件升级命令



- 保护
  - 保护字节是一个常量值 0x5048434D。
  - 该值可防止虚假命令。
  - 自举程序固件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。每条命令的宽度为 1 字节。
  - 支持的命令如下：
    - Unlock (0xA0)
    - Firmware Data (0xA1)
    - Verify (0xA2)
    - Reset (0xA3)
    - Flash Status (0xA4)
    - Device Configuration (0xA5)
- 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

发送命令后，自举程序将以单字符或双字符代码进行响应。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)
- Error (0x51)
- Invalid (0x52)
- Signature OK (0x53)
- Signature Failed (0x54)
- Flash Write OK (0x55)
- Flash Write Failed (0x56)

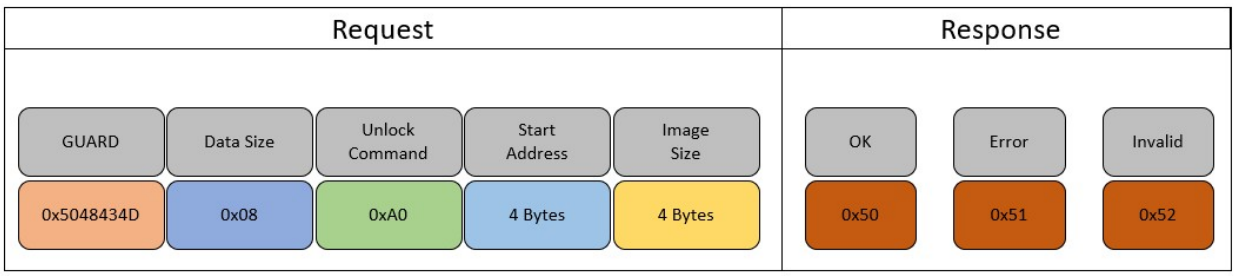
3.2.1. 命令说明

本节给出了安全固件升级协议中使用的请求和响应代码的详细说明。

Unlock 命令

Unlock 命令序列如下图所示，其中包含相应的响应。

图 3-11. 安全固件升级 Unlock 命令



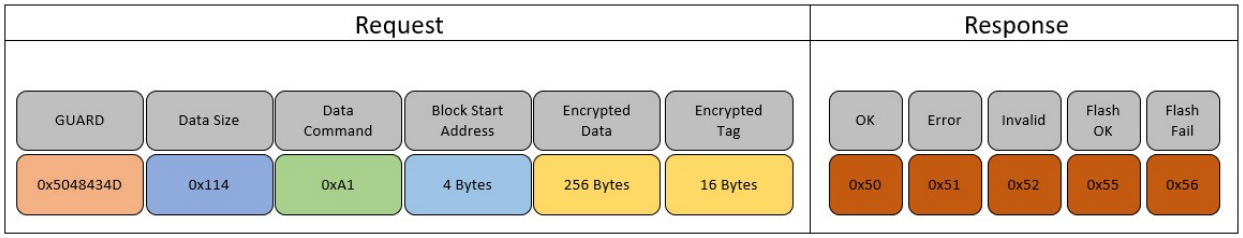
- Unlock 命令必须在第一个 Data 命令发出之前发出

  - 该命令用于计算应用程序的起始地址和结束地址。
  - 该信息将用于验证发送的地址是否处于闪存范围内。
  - 该命令将用于验证要编程的数据包随附的地址是否处于调用 Unlock 命令的区域内。
- 要接收的数据字节数为 8 个字节（起始地址 + 映像大小）。
- 起始地址：
  - 闪存的应用程序起始地址。
  - 该地址取决于器件，应始终大于或等于自举程序的结束地址。
  - 必须在擦除单元大小边界处对齐，该边界取决于器件。
  - 要自行升级自举程序，该值必须设置为 0。
- 映像大小必须以擦除单元字节为增量，具体取决于器件。

Data 命令

Data 命令序列如下图所示，其中包含相应的响应。

图 3-12. 安全固件升级 Data 命令

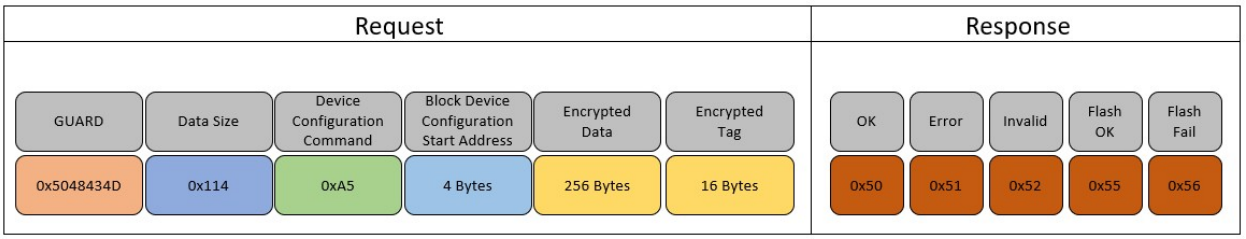


- Data 命令用于传输映像数据。
- 总数据大小包括块起始地址（4 字节）、加密数据（256 字节）和标记（16 字节）。
- 块起始地址必须位于使用 Unlock 命令解锁的区域内。
- 如果尝试在解锁区域的边界以外执行写操作，则将导致错误，提供的数据将被丢弃。如果向闪存写入数据的操作不成功，系统将生成闪存失败代码。

Device Configuration 命令

带相应响应的 Device Configuration 命令序列如下图所示。

图 3-13. 安全固件升级 Device Configuration 命令



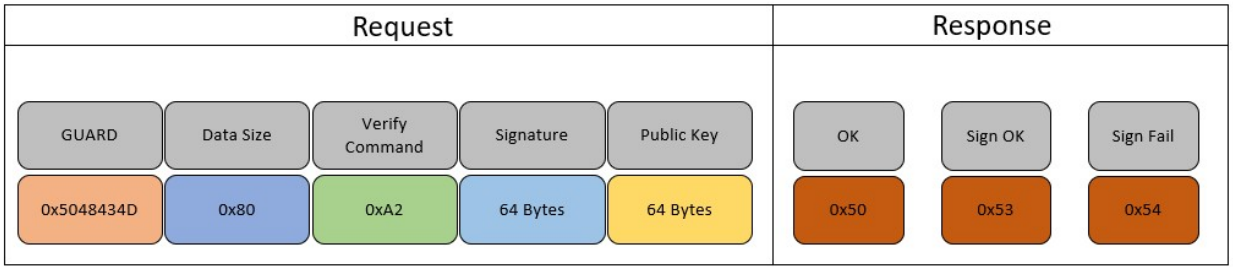
- 该命令仅在为自举程序选择使能熔丝编程功能时才使能。
- Device Configuration 命令用于发送器件配置位（熔丝设置）。数据大小等于器件配置区域起始地址（4 字节）和器件配置页大小（256 字节）的总和，具体取决于器件。
- 器件配置起始地址应为器件配置区域的起始地址。

- 器件配置数据应包含适用于器件的所有熔丝设置。不支持部分熔丝位编程。
- 如果尝试请求在器件配置区域以外执行写操作，则将导致错误，提供的数据将被丢弃。

Verify 命令

带相应响应的 Verify 命令序列如下图所示。

图 3-14. 安全固件升级 Verify 命令

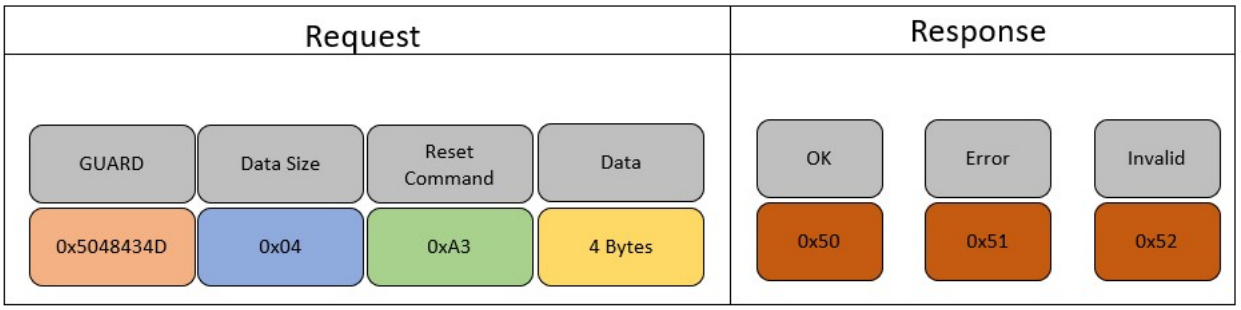


- Verify 命令用于确认传输和编程的映像的完整性。
- 主机端基于应用程序映像使用私钥生成签名。
- 客户端根据编程完成后从闪存中获取的数据计算出签名。将使用公钥验证接收到的应用程序映像的签名。

Reset 命令

带相应响应的 Reset 命令序列如下图所示。

图 3-15. 安全固件升级 Reset 命令



- Reset 命令用于从自举程序切换到应用程序的执行。
- 需补充额外的字节以满足协议结构的要求。
- 该功能在主机无法直接控制复位引脚的情况下至关重要。此外，即使在主机能够控制复位引脚时，该命令仍具实用价值。

3.2.2. 任务说明

本节给出了安全固件升级模式中使用的任务的详细说明。

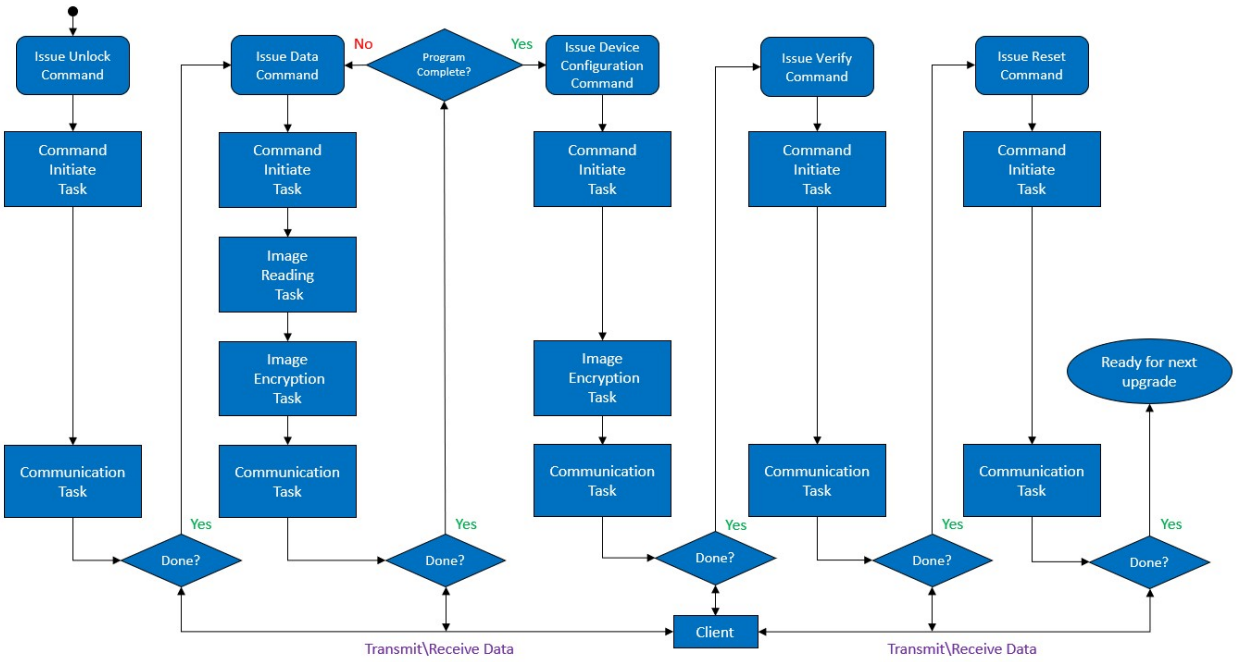
3.2.2.1. 主机任务

- 命令发起任务——该任务负责发起各种命令，以与客户端或目标系统建立连接。该任务根据安全固件升级协议规范构建数据包。固件映像的传输由映像读取任务管理。
- 映像读取任务——该任务负责将目标固件分割成易于管理的数据包，以便将固件完整传输到客户端的器件。分割后的数据随后转发给映像加密任务进行后续加密。

- 映像加密任务——该任务负责使用内部安全元件（ATECC608B）对分割后的固件数据进行加密，确保将加密后的数据连同标记和必要报头信息一起提供给通信任务。
- 通信任务——通信任务负责数据传输，并在任何传输失败时向客户端提供响应。该任务由所选通信接口在中断模式下执行。

下图给出了主机上的固件升级执行流程图。

图 3-16. 主机上的安全固件升级执行流程图

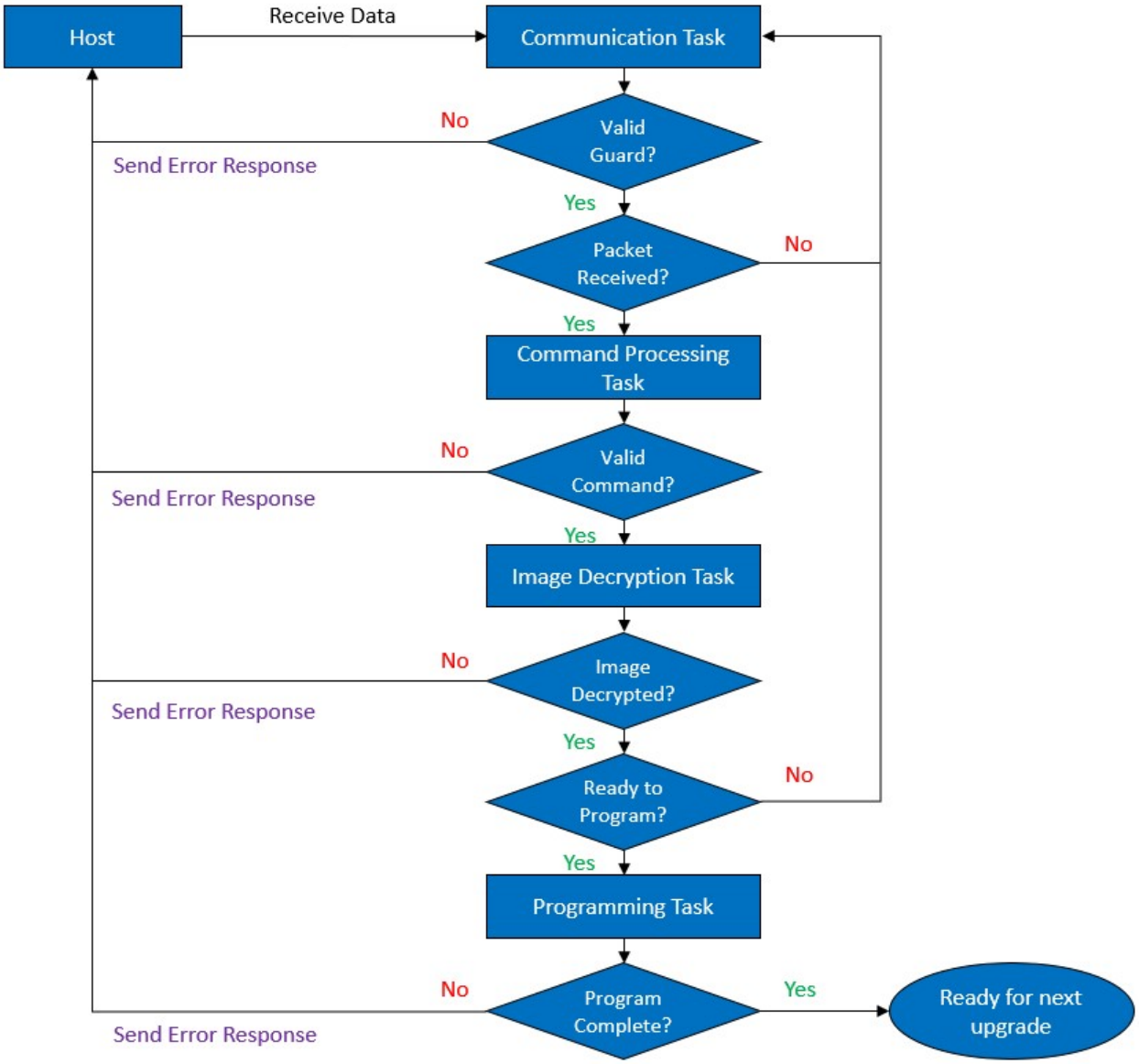


3.2.2.2. 客户端任务

- 通信任务——负责通过所选通信接口在中断模式下从嵌入式主机接收数据。该任务通过验证传入数据包的报头信息是否符合预期参数来确保其完整性，之后将数据转发给命令处理任务进行后续操作。
- 命令处理任务——负责解析和执行由通信任务转发的命令。该任务根据需要向主机返回响应。当命令涉及编程时，该任务会将控制权交由编程任务处理。
- 映像解密任务——负责使用内部安全元件（ATECC608B）对从主机接收的固件数据及相关标记进行解密。该任务会通过比较解密后的标记来确保数据的真实性，之后再将数据传送给编程任务进行操作。
- 编程任务——负责使用接收到的数据包更新内部闪存。为此，该任务利用 NVM 外设库来执行必要的解锁、擦除和写入操作，以确保成功对存储器进行编程。

下图给出了客户端上的固件升级执行流程图。

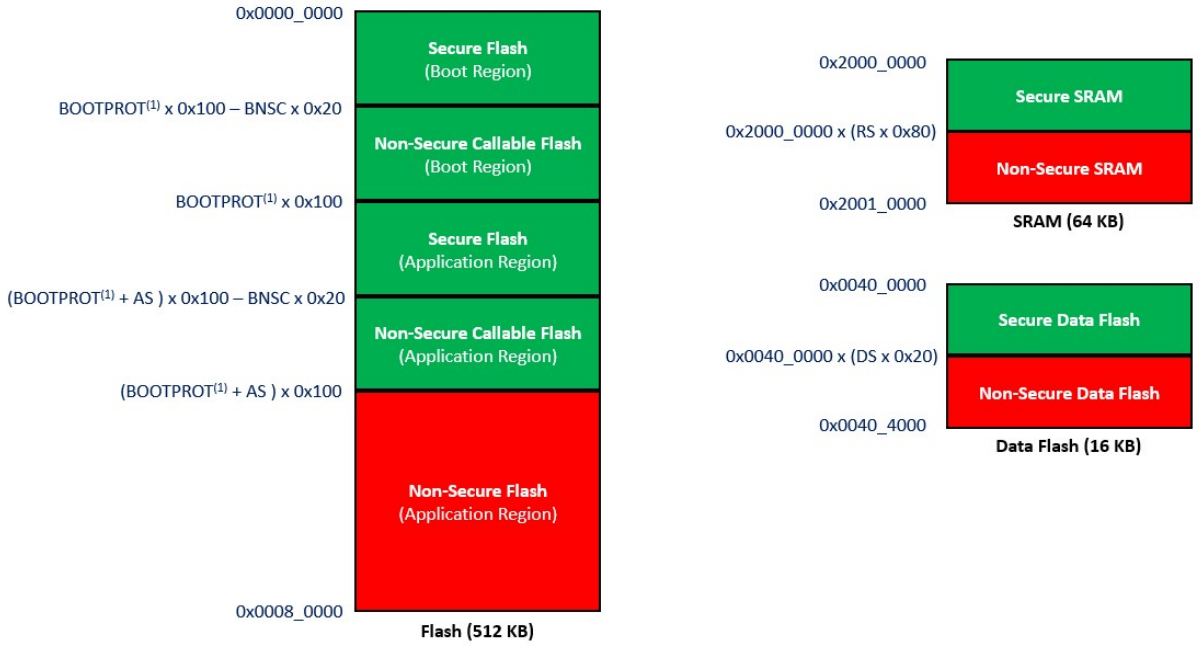
图 3-17. 客户端上的安全固件升级执行流程图



3.3. 存储器布局

PIC32CM LS60 MCU 系列最多划分为 9 个存储区，如下图所示。每个区域的大小可使用特定的 NVM 配置位域（如 BNSC、BOOTPROT、AS、ANSC、DS 和 RS）进行配置。

图 3-18. PIC32CM LSx MCU 存储器布局



注：

1. BOOTPROT = BS。
2. 有关上图中所示的所有 NVM 配置位域缩写的定义，请参见术语表。

每个存储区会在硬件中预配置为下列其中一种属性：

- 安全（S）：用于存储器和外设，但只有安全软件可以访问。
- 非安全可调用（NSC）：一种特殊的安全存储单元，支持软件从非安全状态切换为安全状态。
- 非安全（NS）：用于存储器和外设，器件上运行的所有软件都可以访问。

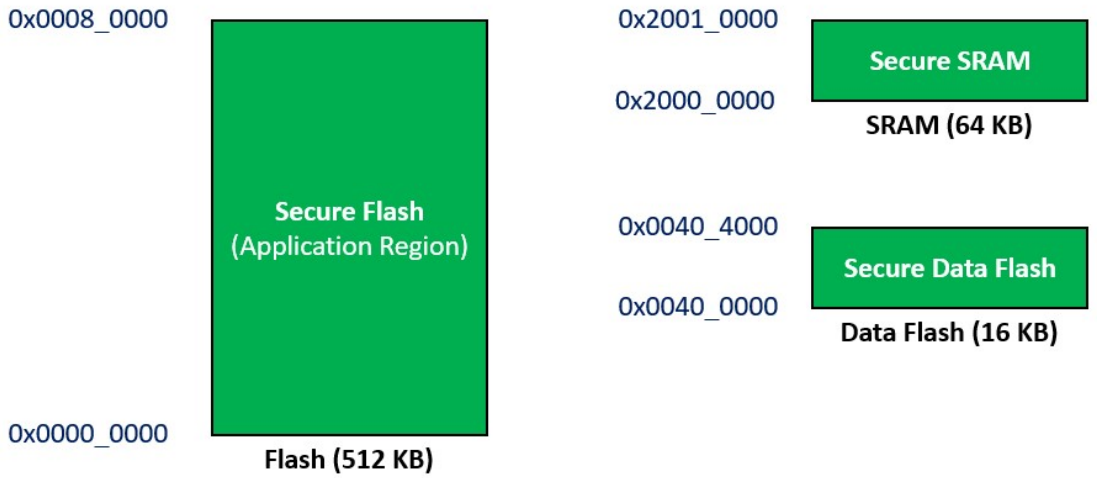
每个区域的安全属性将定义该区域中存储的代码的安全状态。

该安全固件升级应用程序的目标是安全地将客户端固件从主机传输到目标器件。为了增强固件安全性，主机和客户端固件都专门使用安全闪存区域。

下图显示了 PIC32CM LS60 MCU 的安全固件升级应用程序的存储器布局。

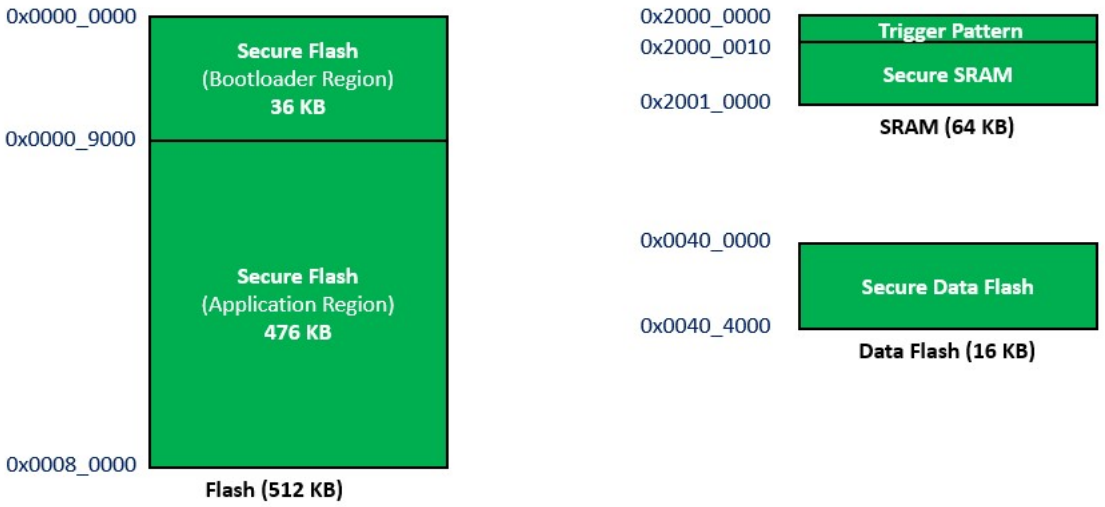
主机固件将整个闪存和 SRAM 用作安全区域。由于客户端固件内嵌于主机固件中，因此也存放在安全位置。

图 3-19. 主机上的存储器布局



客户端的闪存分为两部分：自举程序区域（包含安全固件自举程序代码）和应用程序区域（用于存储接收到的固件）。自举程序触发模式存储在 SRAM 的 0x20001000 地址处，其余存储空间将用于存储自举程序。所有存储区均配置为安全区域，以防范外部威胁。

图 3-20. 客户端上的存储器布局



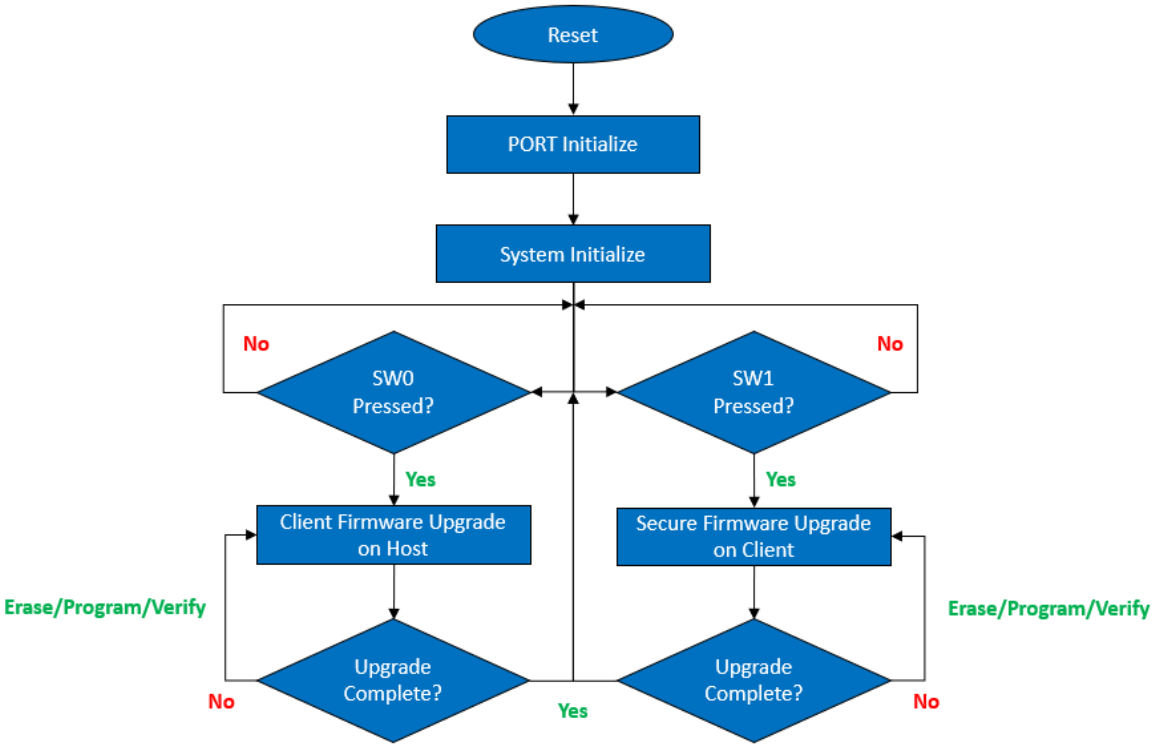
3.4. 执行流程

首次编程安全固件升级应用程序时，使用主机上的主机应用程序和客户端上的自举应用程序。以下主题将讨论 MPLAB Harmony v3 主机应用程序和自举应用程序系统级执行流程。

主机应用程序系统级执行流程

下图展示了首次编程目标应用程序的主机应用程序系统级执行流程。

图 3-21. 主机固件流程



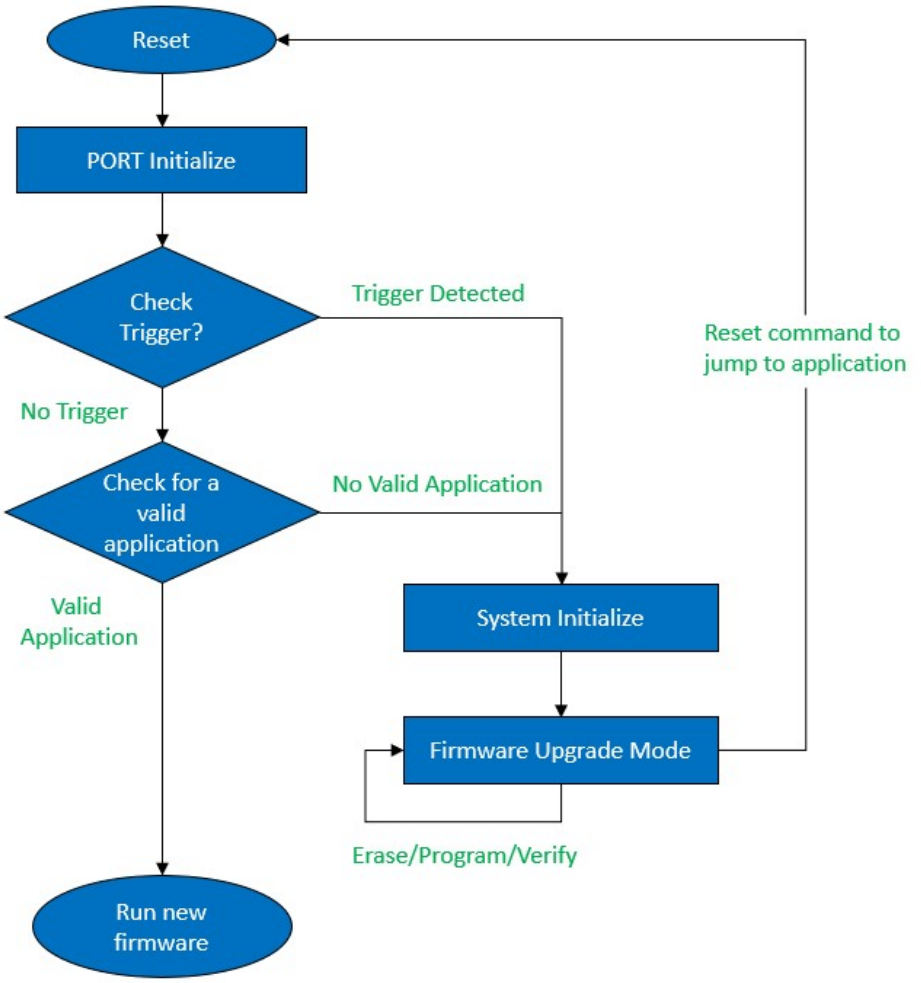
主机应用程序遵循以下执行顺序：

1. 主机 MCU 设计为等待按钮按下，以启动主机上的客户端固件升级或客户端上的安全固件升级。
2. 按下 SW0 按钮时，主机 MCU 进入客户端固件升级模式。在该模式下，它通过 UART 从主机 PC 接收客户端固件。
3. 按下 SW1 按钮时，主机 MCU 进入安全固件升级模式。在该模式下，它通过 UART 将客户端固件传输至客户端。
4. 成功接收或传输固件后，主机 MCU 返回其初始状态。

自举应用程序系统级执行流程

下图给出了首次编程目标应用程序的自举程序系统级执行流程。

图 3-22. 编程目标应用程序的自举程序流程



自举应用程序遵循以下执行顺序：

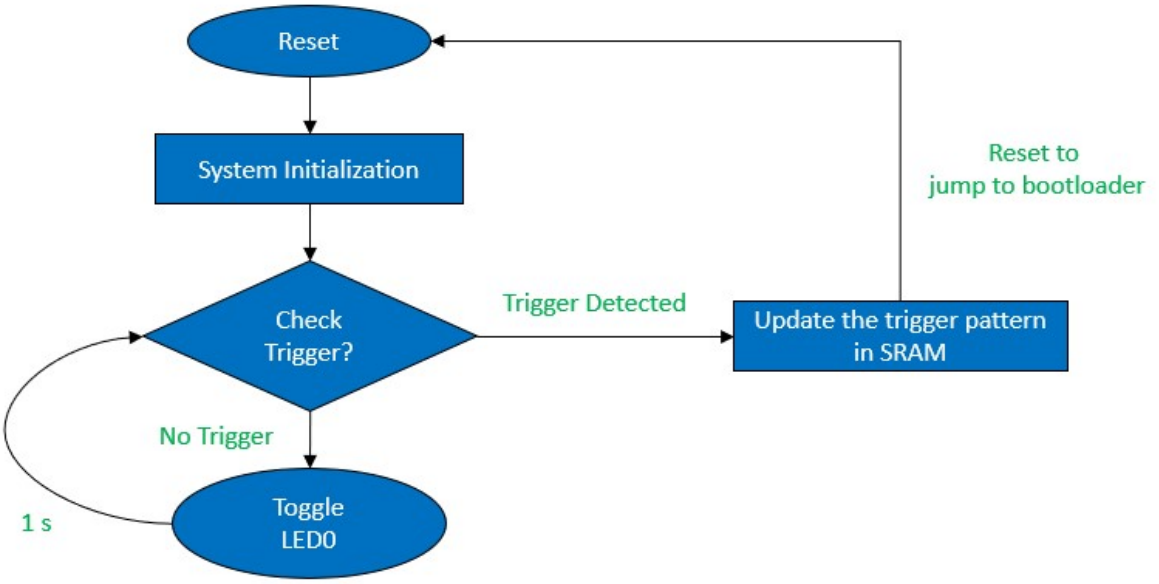
1. 当器件复位时，自举程序代码开始运行。如果没有进入固件升级模式的条件，则自举程序将在初始化端口后运行用户应用程序。
2. 当检测到以下触发条件时，自举程序进入固件升级模式：
   - 使用板上开关作为自举程序触发引脚，在器件复位时强制进入自举程序。
   - 检查来自 SRAM 前 16 字节的特定自举程序请求模式（0x5048434D），以在器件复位时强制进入自举程序。
3. 自举程序在固件升级模式下执行擦除、编程和验证操作。
4. 接收到整个固件后，自举程序通过初始化软件复位来装载新固件。

目标应用程序系统级执行流程

通过自举程序成功编程目标应用程序后，自举程序将装载目标应用程序，CPU 随即开始执行它。

下面的流程图展示了目标应用程序的执行流程。

图 3-23. 目标应用程序执行流程



目标应用程序的执行顺序如下：

1. 当器件复位时，自举程序从安全闪存的应用程序区域装载目标应用程序。
2. 目标应用程序初始化系统并在串行控制台上打印启动消息。
3. 板载 LED0 每秒切换一次状态，直至按下 SW0 开关进入固件升级模式，此时通过修改 SRAM 中的触发模式来触发系统复位，从而运行自举应用程序。

4. 编译并运行应用程序

4.1. 主机板和客户端板上的密钥配置

按照“Key Provisioning with 32-bit MCU Secure Devices using the TPDS Technical Brief”（*DS90003374*）中提供的说明，为主机和客户端板编程机密信息密钥和 I/O 保护密钥。这两个密钥对于在两块板上运行安全固件升级演示至关重要。

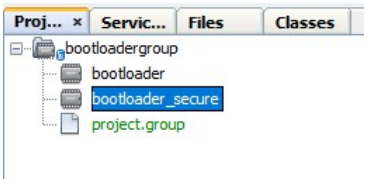
注：确保在可信平台设计套件中添加了 MPLAB X IDE 路径（*File* > *Preferences* > *MPLAB X IDE path*（文件 > 首选项 > MPLAB X IDE 路径））。

4.2. 运行客户端自举应用程序

按照以下步骤在其中一个 PIC32CM LS60 Curiosity Pro 评估工具包上编程客户端自举应用程序。

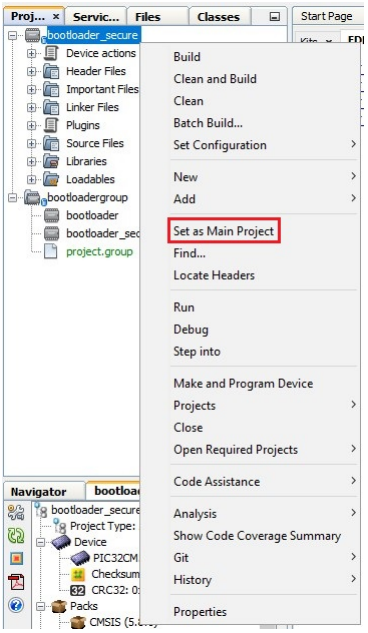
1. 下载客户端自举应用程序（单击此处）。
2. 使用 MPLAB X IDE 打开客户端自举应用程序项目组（路径：*<已下载路径>/pic32cm_ls60_secure_firmware_upgrade/firmware/bootloader/bootloadergroup*）。
3. 要启动项目，请单击并展开 **bootloadergroup**，然后双击 **bootloader_secure**。

图 4-1. 加载客户端自举程序项目



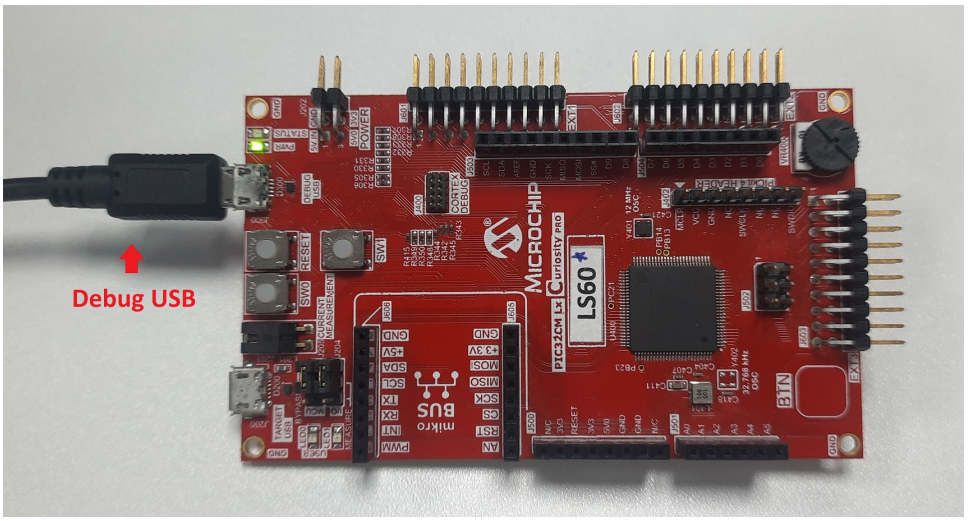
4. 要将 bootloader_secure 项目设置为主项目，请右键单击该项目，然后选择 **Set as Main Project**（设置为主项目）。

图 4-2. 选择客户端自举程序项目作为主项目



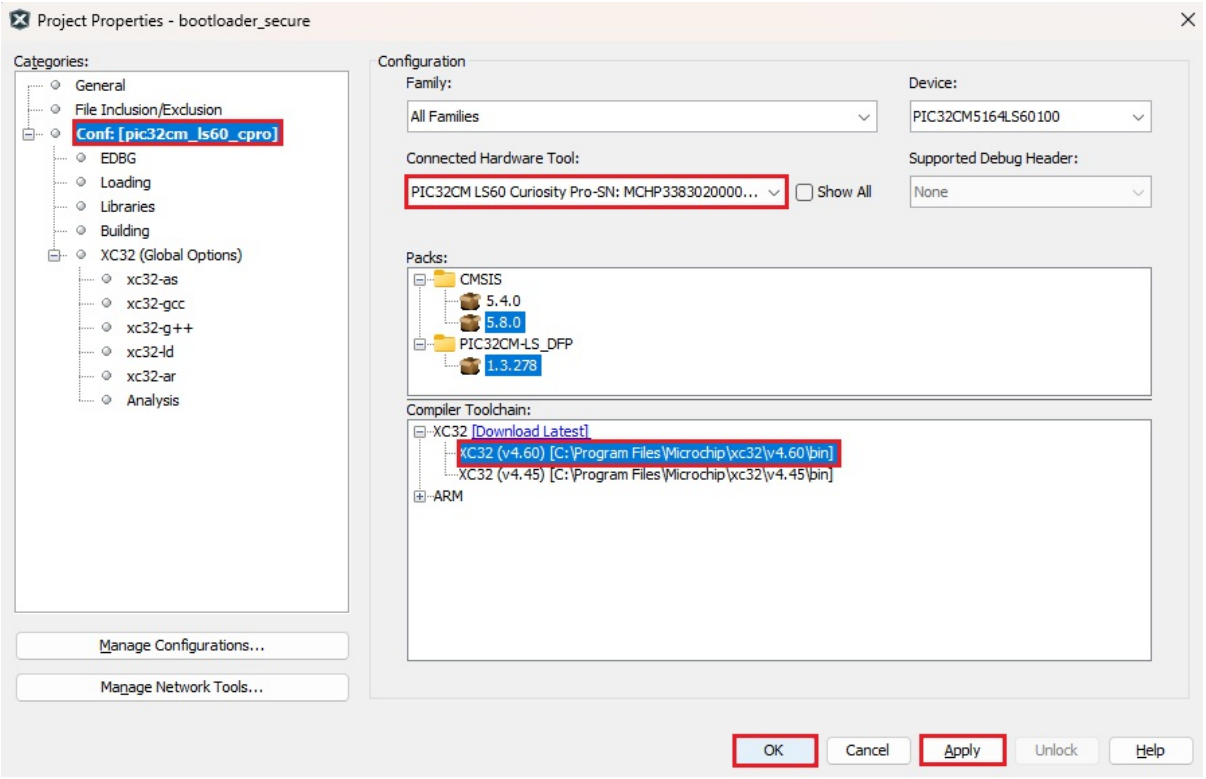
5. PIC32CM LS60 Curiosity Pro 评估工具包支持使用嵌入式调试器（EDBG）进行调试。将“Type-A 公头转 micro-B”USB 线缆连接到 PIC32CM LS60 Curiosity Pro 评估工具包上的 micro-B 调试 USB 端口，以便对 PIC32CM LS60 Curiosity Pro 评估工具包进行供电和调试。

图 4-3. PIC32CM LS60 Curiosity Pro 板硬件设置



6. 在客户端项目属性窗口中，选择硬件工具和编译器，然后在 MPLAB X IDE 项目属性窗口中执行这些任务
   a. 在左侧的 *Categories*（类别）部分下，选择 **Conf: [pic32cm_ls60_cpro]**。
   b. 在右侧 Configuration（配置）属性页面中选择 Connected Hardware Tool（连接的硬件工具）和 Compiler Toolchain（编译器工具链），如下所示。

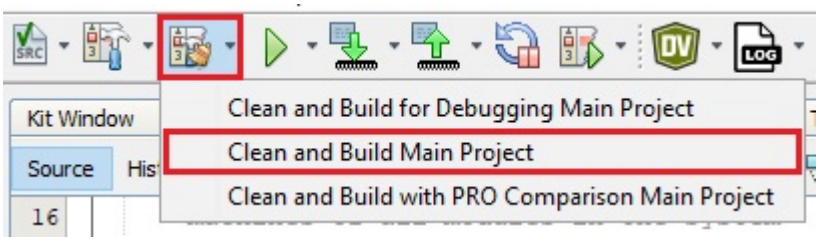
图 4-4. 客户端项目属性——PIC32CM LS60 Curiosity Pro 评估工具包



注：记下客户端自举程序板的序列号，避免与主机板混淆。

7. 单击 **Apply**（应用），然后单击 **OK**（确定）。
8. 通过单击 **Clean and Build**（清除并编译）图标或从下拉列表中选择 **Clean and Build Main Project**（清除并编译主项目）编译项目，然后验证项目是否成功编译。

图 4-5. 清除并编译项目



9. 单击下图中红框部分的图标以编程应用程序。

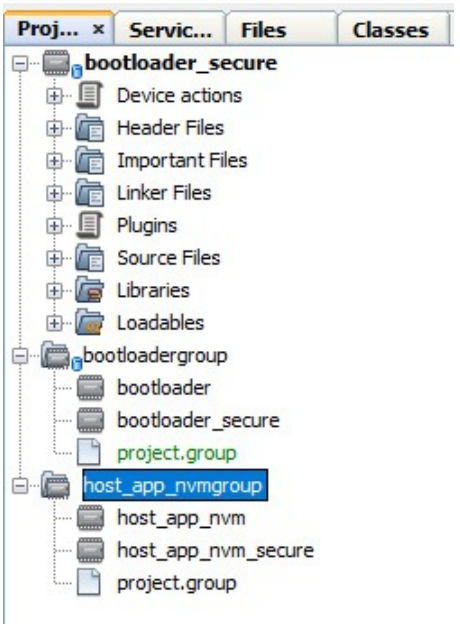
图 4-6. 编程器件

4.3. 运行主机应用程序

按照以下步骤在其中一个 PIC32CM LS60 Curiosity Pro 评估工具包上编程主机应用程序。

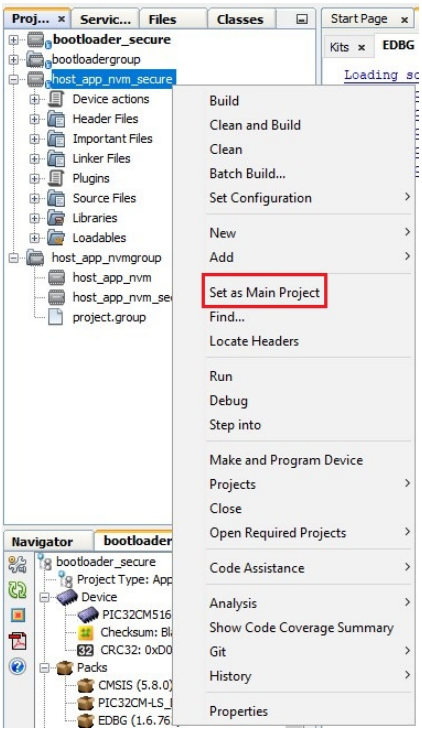
1. 下载主机应用程序（单击此处）。
2. 使用 MPLAB X IDE 打开主机应用程序项目组（路径：*<已下载路径>/pic32cm_ls60_secure_firmware_upgrade/firmware/host_app_nvm/host_app_nvmgroup*）。
3. 要启动项目，请单击并展开 **bootloader_secure**，然后双击 **host_app_nvm_secure**。

图 4-7. 加载主机项目



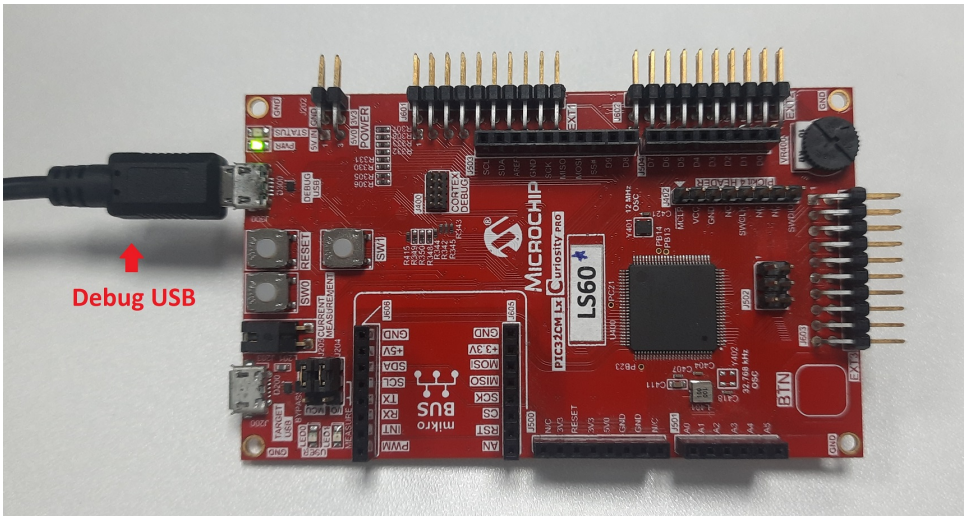
4. 通过右键单击项目并选择 **Set as Main Project**，将 **host_app_nvm_secure** 项目设置为主项目。

图 4-8. 将主机项目设置为主项目



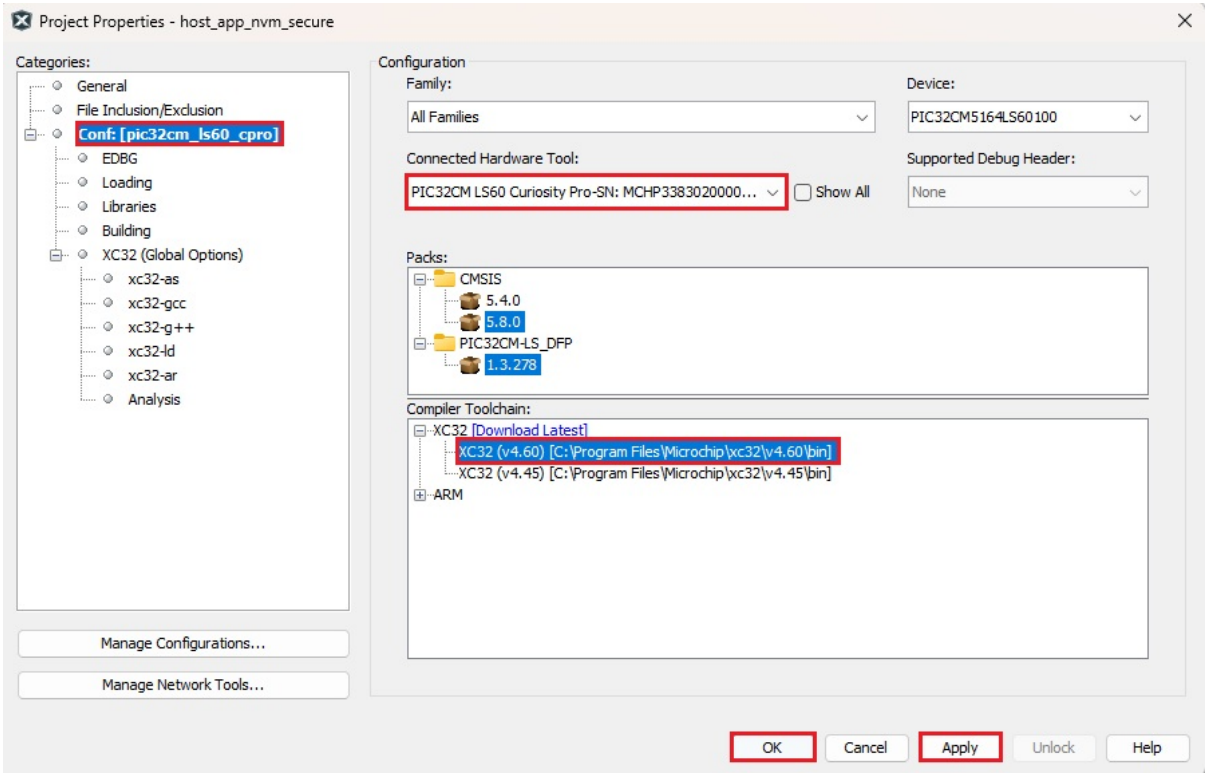
5. PIC32CM LS60 Curiosity Pro 评估工具包支持使用嵌入式调试器（EDBG）进行调试。将“Type-A 公头转 micro-B”USB 线缆连接到 PIC32CM LS60 Curiosity Pro 评估工具包上的 micro-B 调试 USB 端口，以便对 PIC32CM LS60 Curiosity Pro 评估工具包进行供电和调试。

图 4-9. PIC32CM LS60 Curiosity Pro 板硬件设置



6. 转到主机项目属性并选择硬件工具和编译器，然后在 MPLAB X IDE 项目属性窗口中执行以下操作：
   a. 在左侧的 *Categories* 部分下，选择 **Conf: [pic32cm_ls60_cpro]**。
   b. 在右侧 Configuration 属性页面中选择 Connected Hardware Tool 和 Compiler Toolchain，如下所示。

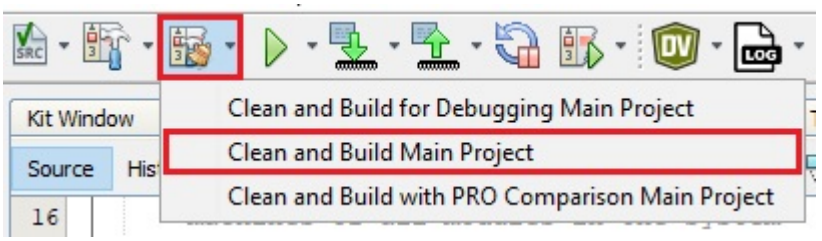
图 4-10. 主机项目属性——PIC32CM LS60 Curiosity Pro 评估工具包



注：确保为主机应用程序选择不同的 PIC32CM LS60 Curiosity Pro 评估工具包。

7. 单击 **Apply**，然后单击 **OK**。
8. 通过单击 **Clean and Build** 图标或从下拉列表中选择 **Clean and Build Main Project** 编译项目，然后验证项目是否成功编译。

图 4-11. 清除并编译项目



9. 单击下图中红框部分的图标以编程应用程序。

图 4-12. 编程器件

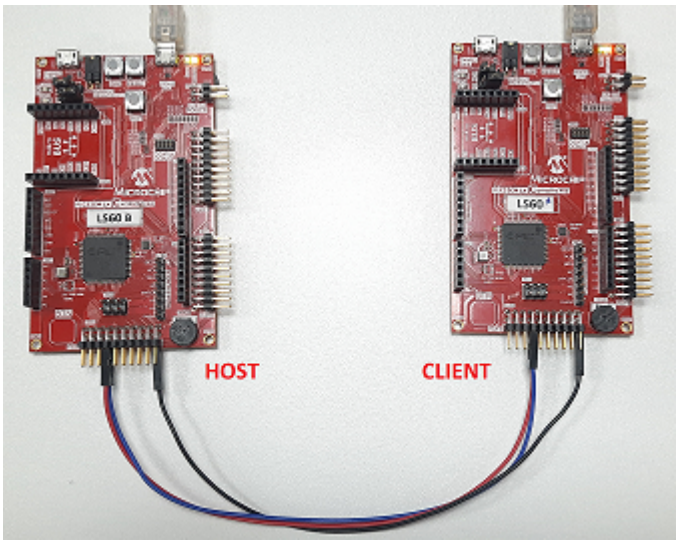
5. 观察串行控制台上的输出

1. 根据下表使用跳线连接主机与客户端之间的 UART 线（TX 和 RX）和接地（GND）。

表 5-1. 主机与客户端之间的硬件连接

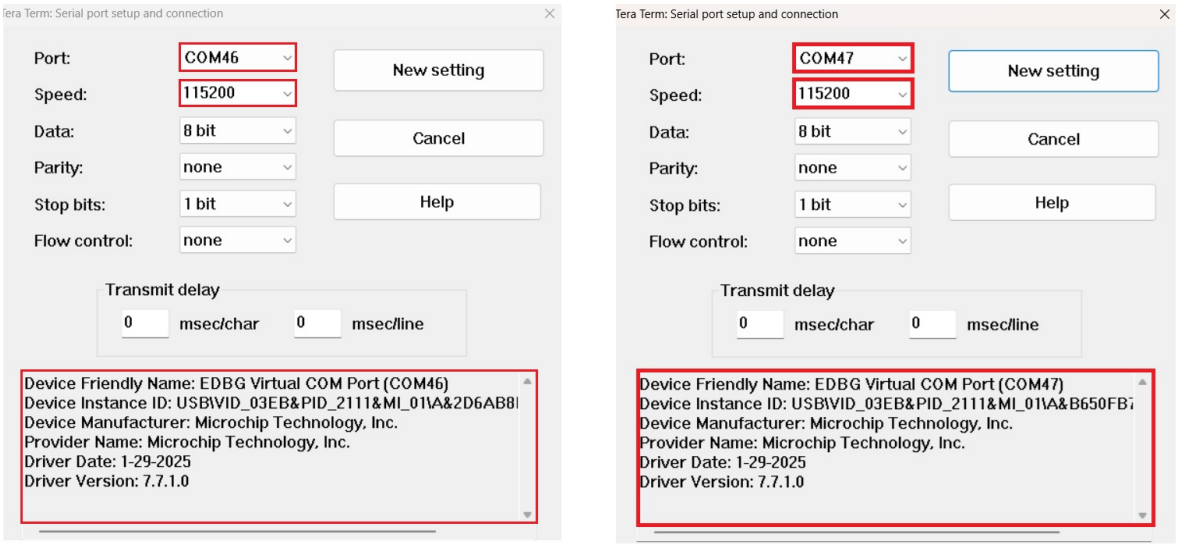
PIC32CM LS60 Curiosity Pro	UART	UART	GND
主机	PA12 （EXT 3，PIN 14，TX）	PA13 （EXT 3，PIN 13，RX）	GND （EXT 3，PIN 2，GND）
客户端 （自举程序）	PA13 （EXT 3，PIN 13，RX）	PA12 （EXT 3，PIN 14，TX）	GND （EXT 3，PIN 2，GND）

图 5-1. 主机与客户端之间的硬件连接



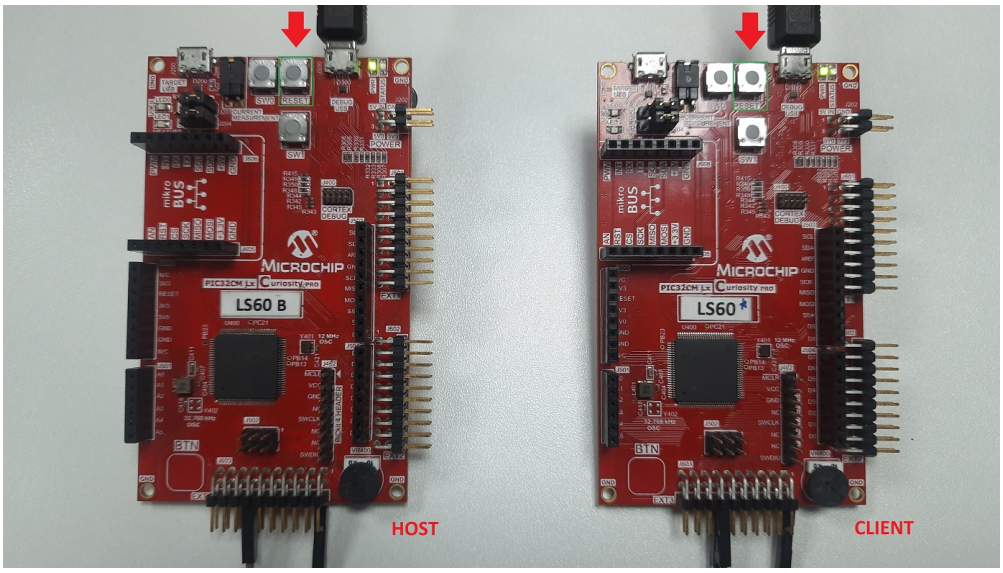
2. 为主机和客户端打开两个串行控制台应用程序（例如，主机 PC 上的 Tera Term），然后对串行端口进行如下设置。

图 5-2. 主机和客户端的串行端口配置



3. 按下两个 PIC32CM LS60 Curiosity Pro 评估工具包上的**复位**按钮，以复位主机和客户端器件。

图 5-3. 复位主机板和客户端板



4. 观察主机和客户端串行控制台上的控制台消息。

图 5-4. 主机控制台消息

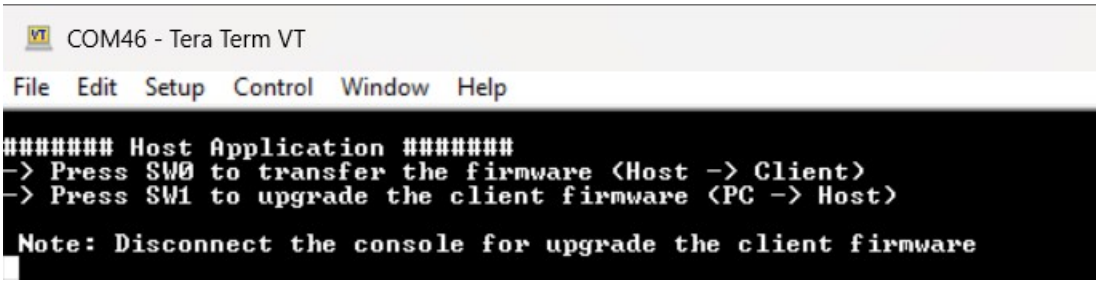
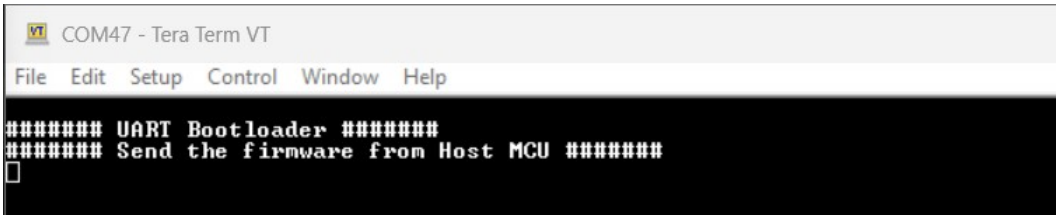
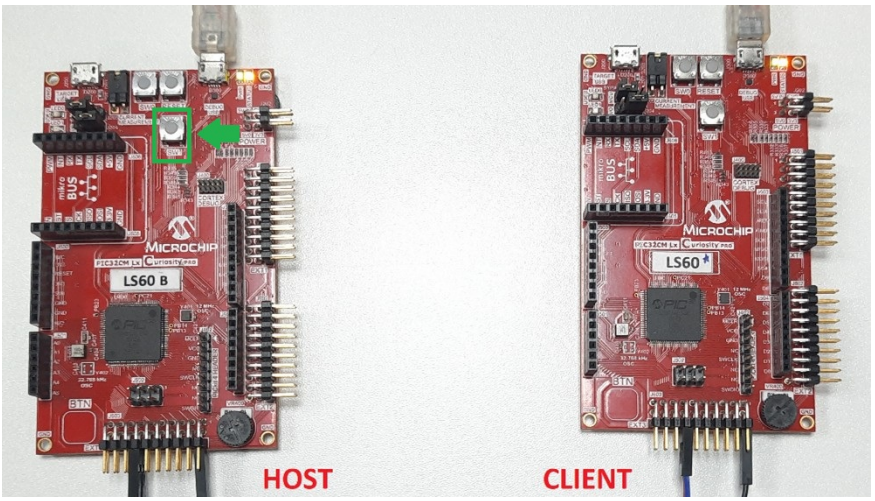


图 5-5. 客户端控制台消息



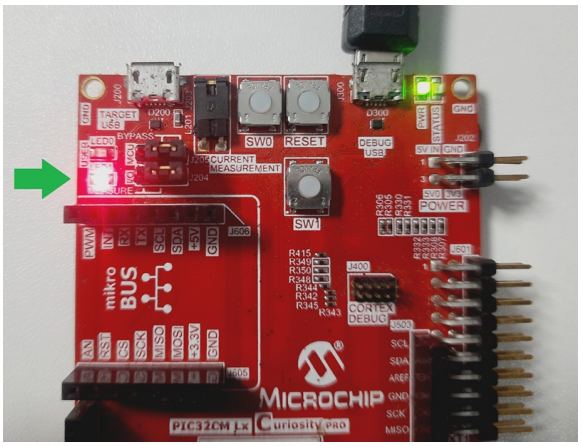
5. 断开或关闭主机 MCU 的串行控制台，以使主机 PC 能够访问并传输客户端固件。
6. 按下主机评估板上的 **SW1** 按钮，以从 PC 接收客户端固件。

图 5-6. 将客户端固件传输到主机板上



注：主机评估板上的 **LED1** 将亮起，以指示主机 MCU 已准备好接收客户端固件。

图 5-7. 客户端固件接收过程指示



7. 打开命令提示符并运行 Python 脚本（`btl_host.py`），用于将目标应用程序二进制文件和配置位传输至主机 MCU 板。

```
python <harmony3_path>/bootloader/tools/btl_host.py -v -i <COM PORT> -d PIC32CM -a 0x9000
-f <downloaded path>/pic32cm_ls60_secure_firmware_upgrade/hex/test_app/
test_app_secure.X.production.bin -c <downloaded path>/pic32cm_ls60_secure_firmware_upgrade/
scripts/application_user_configurations_out.txt
```

注：用户需要在运行 Python 脚本之前安装 pyserial 包。

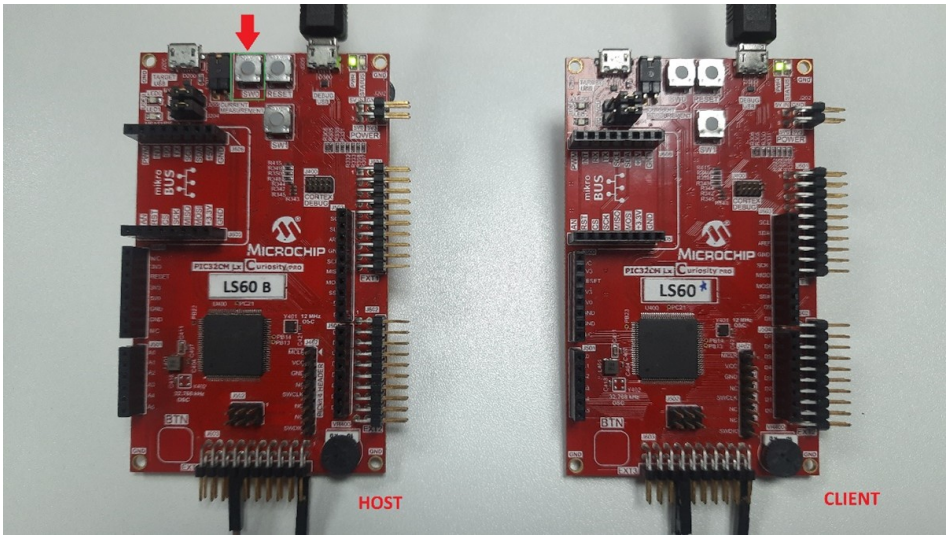
图 5-8. 运行 Python 脚本

图 5-9. 完成固件传输

```
C:\WINDOWS\system32\cmd.
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>python D:\Harmony_3\bootloader\tools\btl_host.py -v -i COM46 -d PIC32CM -a 0x9000 -f D:\pic32cm_ls60_secure_firmware_upgrade\hex\test_app\test_app_secure.X.production.bin -c D:\pic32cm_ls60_secure_firmware_upgrade\scripts\application_user_configurations_out.txt

Reading Bootloader Version

Bootloader version : v3.7

Unlocking

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

Verification

... success

Sending Device Configuration Bits

Rebooting

Reboot Done
```

8. 按照步骤 2 中所述重新打开串行控制台。
9. 按下主机评估板上的 SW0 按钮以升级客户端固件。固件升级的进度将显示在终端控制台中。

图 5-10. 在客户端上启动固件升级过程



注：主机评估板上的 LED0 将亮起，以指示主机 MCU 已在客户端上启动安全固件升级。

图 5-11. 客户端上的安全固件升级过程指示

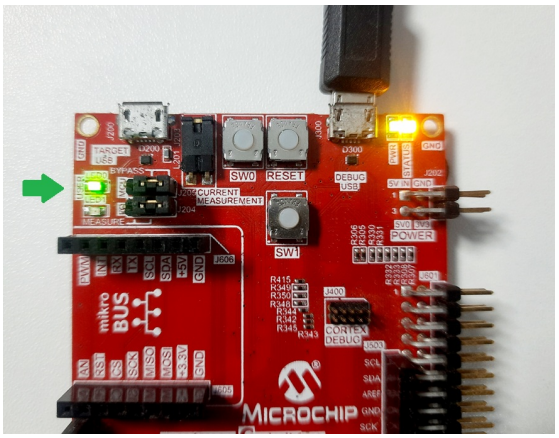
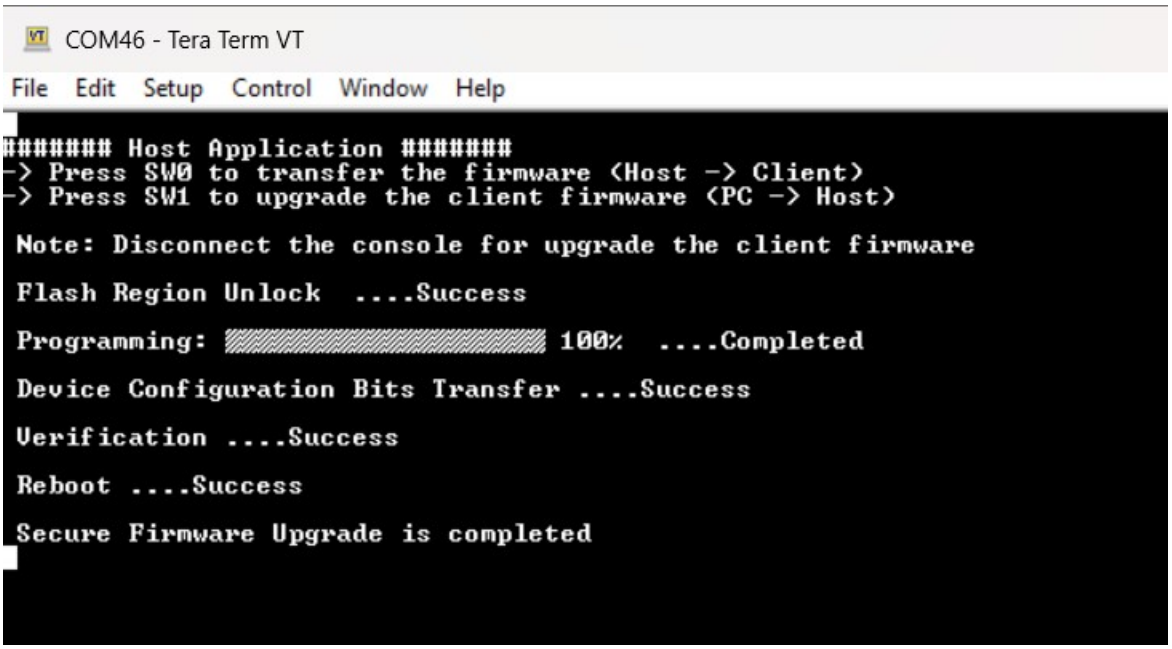
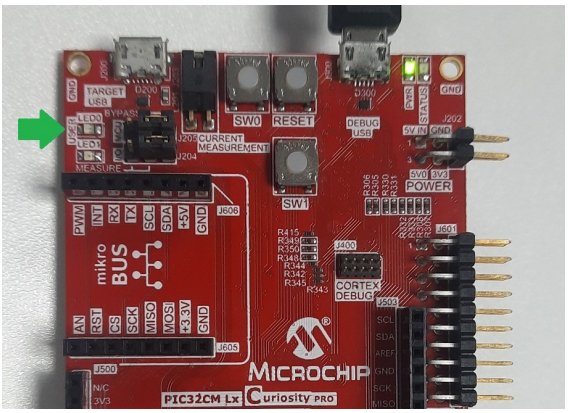


图 5-12. 主机固件升级进度



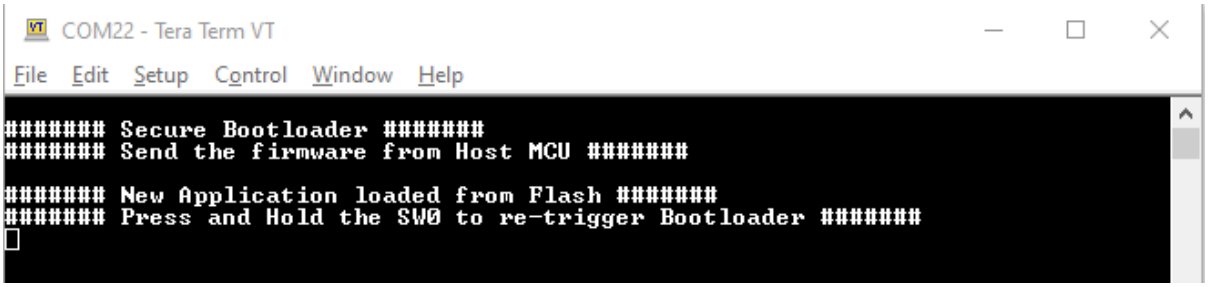
10. 当固件更新完成后，主机评估板上的 LED0 将熄灭。

图 5-13. 主机固件升级完成



11. 在客户端评估板上，LED0 将开始闪烁并在终端控制台上显示启动消息。

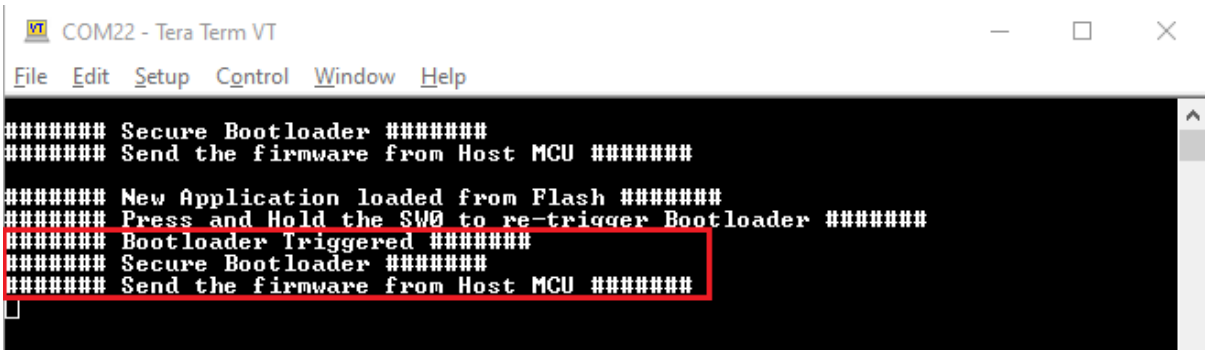
图 5-14. 客户端固件升级完成



5.1. 客户端上的自举程序触发机制

1. 按住客户端评估板上的 SW0 按钮，以从测试应用程序触发自举程序。

图 5-15. 触发自举程序



2. 按下主机评估板上的复位按钮，然后按下 SW0 以重新编程应用程序。

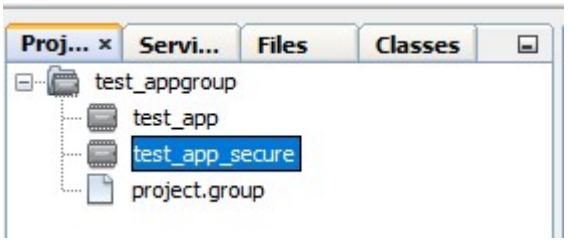
6. 在主机应用程序中更新目标固件

6.1. 更新目标固件

要使用 MPLAB Harmony v3 更新目标固件，请按照以下步骤操作。

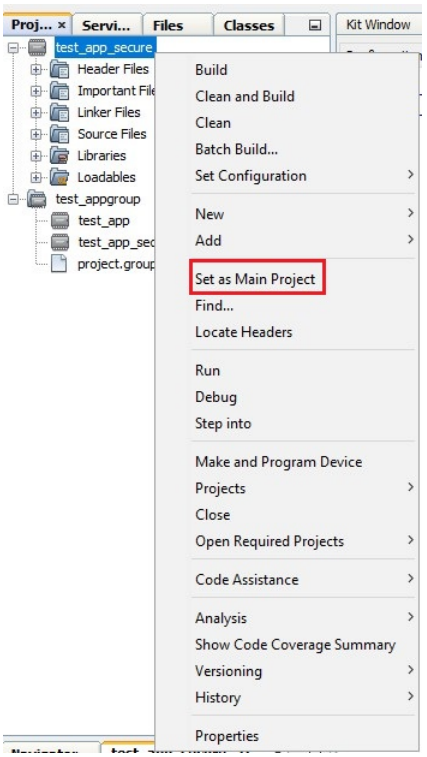
1. 下载目标固件（单击此处）。
2. 使用 MPLAB X IDE 打开目标应用程序项目组（路径：*<已下载路径>/pic32cm_ls60_secure_firmware_upgrade/firmware/test_app/test_appgroup*）。
3. 要启动项目，请单击并展开 test_appgroup，然后选择 **test_app_secure** 项目。

图 6-1. 加载目标应用程序项目



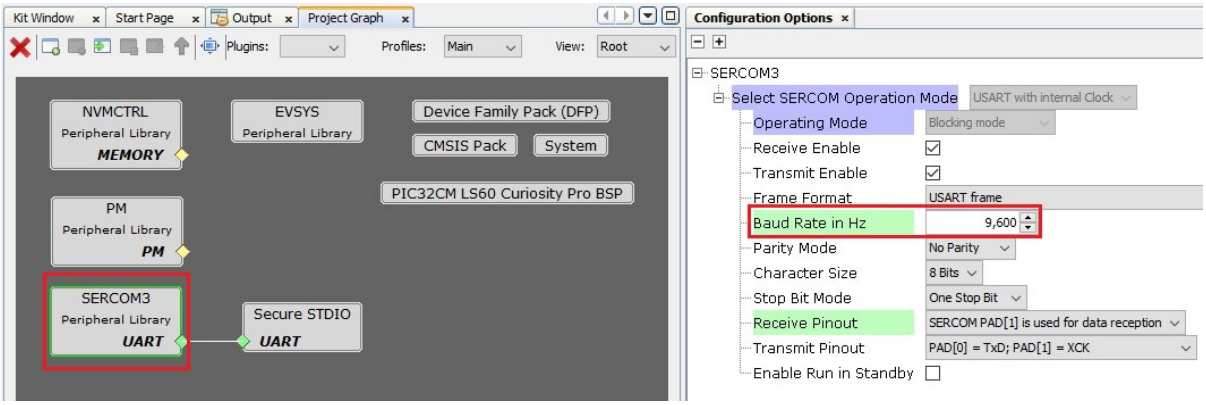
4. 要将 test_app_secure 项目设置为主项目，请右键单击项目，然后选择 **Set as Main Project**。

图 6-2. 将目标应用程序自举程序项目设置为主项目



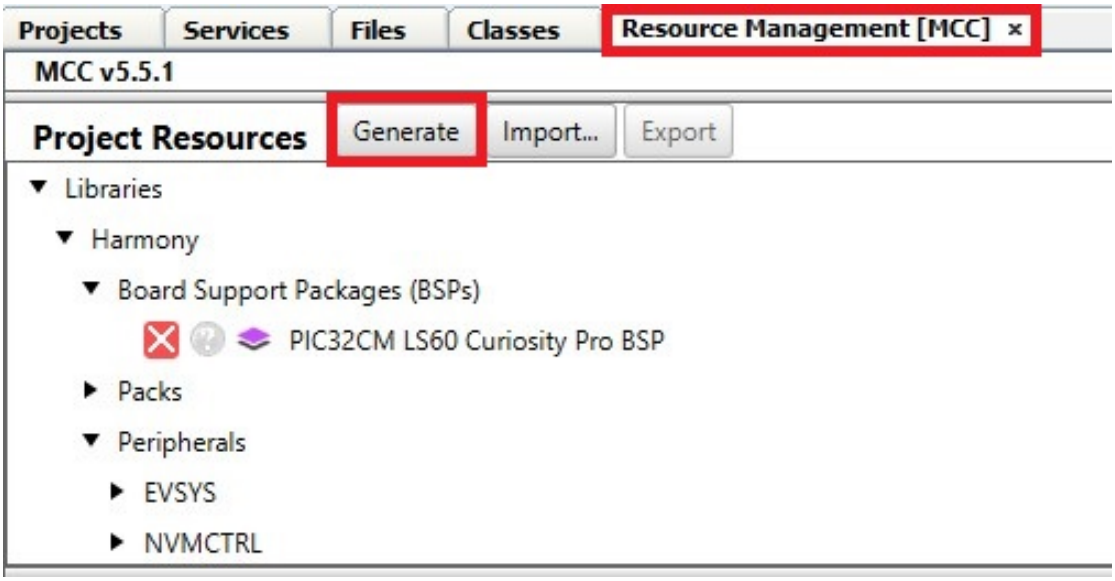
5. 启动 MCC，然后从左侧窗格选择 **SERCOM3**。在右侧 **Configuration Options**（配置选项）属性页面中，对于 Baud Rate in Hz（波特率（Hz））选择 **9600**。

图 6-3. 目标应用程序——SERCOM 波特率重新配置



6. 配置外设后，单击 **Resource Management [MCC]**（资源管理[MCC]），然后单击 **Generate**（生成）选项卡。

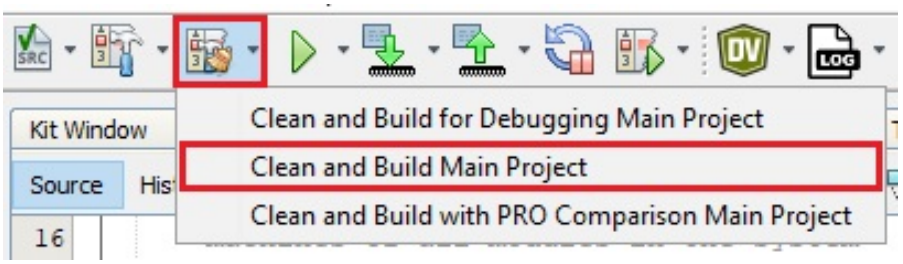
图 6-4. 生成代码



注：在编译项目之前重新生成 test_app 项目的代码。

7. 通过单击 **Clean and Build** 图标或从下拉列表中选择 **Clean and Build Main Project** 编译项目，然后验证项目是否成功编译。

图 6-5. 清除并编译项目



6.2. 将新的目标固件加载到主机应用程序中

当 test_app_secure 编译完成后，重复执行观察串行控制台上的输出中提及的所有步骤。

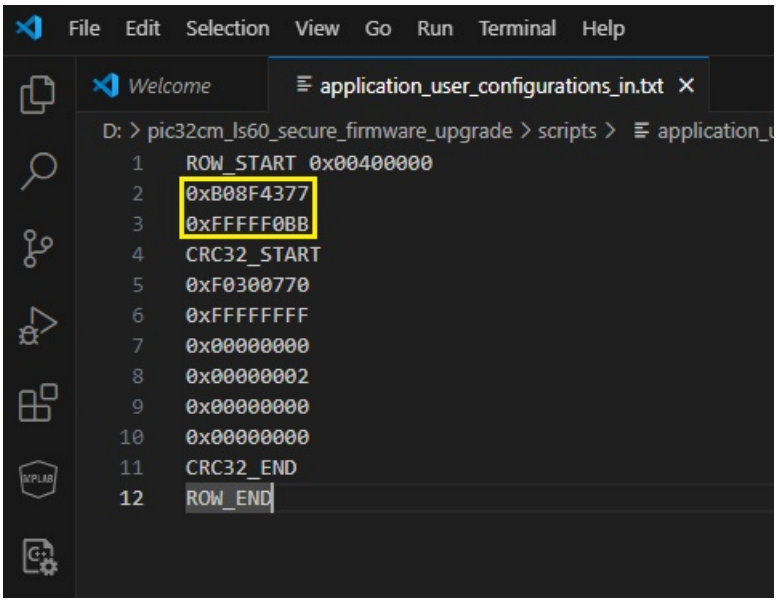
注：

- 在客户端固件升级过程中，会自动从 hex/test_app 文件夹中检索更新后的目标应用程序（.bin 文件）。
- 此外，任何新的外设都会集成到目标固件中，并且目标或客户端应用程序的用户行配置（配置位）会相应地在主机 MCU 内进行更新。

按照以下步骤更新目标固件配置位。

1. 在文件编辑器（如 Notepad 或 Visual Studio Code）中打开 `application_user_configurations_in.txt`。
2. 根据 test_app_secure 项目中的外设使用情况编辑配置位。

图 6-6. 配置位修改



注：

- 主机 MCU 将客户端的配置位值存储在安全数据存储区（0x0040_0000）中。
- 更多信息，请参见器件数据手册中*存储器*一章的 *NVM 配置行*一节。

3. 打开命令提示符并运行 Python 脚本（`btl_dev_cfg_gen.py`），以创建 `application_user_configurations_out.txt` 配置文件。`btl_host.py` 脚本将使用该文件在对目标或客户端应用程序进行编程时更新配置位。

```
python <harmony3_path>/bootloader/tools/btl_dev_cfg_gen.py -v -f  <downloaded path>/
pic32cm_ls60_secure_firmware_upgrade/scripts/application_user_configurations_in.txt -o
<downloaded path>/ pic32cm_ls60_secure_firmware_upgrade/scripts/
application_user_configurations_out.txt
```

图 6-7. 生成用户行配置文件

7. 结论

本文档指导如何通过 UART 接口安全地更新固件，如何利用内部安全元件，以及如何通过 MPLAB Harmony v3 框架重新配置应用程序。本文档为开发需要实现安全固件升级功能的实时应用提供了实用的参考资料。

注：PIC32CM LS60 中的引导 ROM 通过 SHA-256、HMAC 和 ATECC608B 支持安全引导。使能安全引导后，可对闪存的可配置部分（称为引导区域）中的客户端自举程序映像进行验证。如果身份验证过程失败，可复位器件并重新启动身份验证过程。客户端自举程序可利用这一安全引导功能来提高安全级别。

8. 术语表

下表列出了本文档中使用的 NVM 配置位域缩写及其定义。

缩写	NVM 配置行	定义
BOOTPROT（引导保护）	BOCOR	定义引导区域的（S）+（NSC）子区域的大小
BOOTOPT（引导选项）	BOCOR	定义应用于 BOOTPROT 子区域的安全引导检查
BNSC（引导非安全可调用）	BOCOR	定义引导区域的（NSC）子区域的大小
AS（应用程序安全）	UROW	定义应用程序区域的（S）+（NSC）子区域的大小
ANSC（应用程序非安全可调用）	UROW	定义应用程序区域的（NSC）子区域的大小
RS（SRAM 安全）	UROW	定义 SRAM 区域的（S）区域的大小
DS（数据闪存安全）	UROW	定义数据闪存区域的（S）区域的大小

9. 参考资料

- 可从 Microchip 网站（www.microchip.com）下载以下文档：
  - PIC32CM LE00/LS00/LS60 Family Data Sheet（*DS60001615*）
  - PIC32CM LE00/LS00/LS60 Curiosity Pro User Guide（*DS70005443*）
  - Getting Started with the PIC32CM LE00/LS60/LS60 Curiosity Pro Board（*DS00004511*）
  - PIC32CM LS00/LS60 Security Reference Guide（*DS00003992*）
  - Key Provisioning with 32-bit MCU Secure Devices using TPDS（*DS90003374*）
- PIC32CM LS60 Curiosity Pro 评估工具包
- 使用 MPLAB® Harmony v3 软件框架在 PIC32CM LS60 Curiosity Pro 评估工具包上实现安全引导
- 有关 MPLAB Harmony v3 的更多信息，请访问 Microchip 网站： developerhelp.microchip.com/xwiki/bin/view/software-tools/harmony/
- 有关各种应用程序的更多信息，请参见 https://github.com/Microchip-MPLAB-Harmony/reference_apps
- 有关 32 位单片机资料和解决方案的更多信息，请参见 《32 位单片机相关资料和解决方案参考指南》（*DS70005534A_CN*）

10. 版本历史

10.1. 版本 A（2025 年 6 月）

这是本文档的初始版本。

Microchip 信息

商标

“Microchip”的名称和徽标组合、“M”徽标及其他名称、徽标和品牌均为 Microchip Technology Incorporated 或其关联公司和/或子公司在美国和/或其他国家或地区的注册商标或商标（“Microchip 商标”）。有关 Microchip 商标的信息，可访问 https://www.microchip.com/en-us/about/legal-information/microchip-trademarks。

ISBN： 979-8-3371-2765-1

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。

本出版物及其提供的信息仅适用于 Microchip 产品，包括设计、测试以及将 Microchip 产品集成到您的应用中。以其他任何方式使用这些信息都将被视为违反条款。本出版物中的器件应用信息仅为您提供便利，将来可能会发生更新。您须自行确保应用符合您的规范。如需额外的支持，请联系当地的 Microchip 销售办事处，或访问 www.microchip.com/en-us/support/design-help/client-support-services。

Microchip“按原样”提供这些信息。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或附带的损失、损害或任何类型的开销，Microchip 概不承担任何责任，即使 Microchip 已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔，Microchip 在任何情况下所承担的全部责任均不超出您为获得这些信息向 Microchip 直接支付的金额（如有）。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切损害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

Microchip 器件代码保护功能

请注意以下有关 Microchip 产品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：在正常使用且符合工作规范的情况下，Microchip 系列产品非常安全。
- Microchip 注重并积极保护其知识产权。严禁任何试图破坏 Microchip 产品代码保护功能的行为，这种行为可能会违反《数字千年版权法案》（Digital Millennium Copyright Act）。
- 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。

产品页链接

PIC32CM5164LS60048、PIC32CM5164LS60064 和 PIC32CM5164LS60100