STM32WB 系列的 ST 固件升级服务

引言

本文档介绍了 STM32WB 系列微控制器可用的固件升级服务（FUS）。这些服务由位于嵌入式 Flash 存储器安全部分中的意法半导体代码提供，可供通过用户 Flash 存储器在 Cortex®-M4 上运行的任何代码使用，也可以通过嵌入式自举程序指令（也在 Cortex®-M4 上运行）使用。

1 概述

本文档适用于基于 ARM®的 STM32WB 系列器件。

注意： *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。*

1.1 固件升级服务定义

FUS（固件升级服务）是在 STM32WB Cortex®-M0+上运行的固件，并提供以下功能：

1. 安装、升级或删除 STM32WB Cortex®-M0+无线协议栈：
   - 仅由意法半导体加密和签名
   - 或者，如果需要，客户还可以进行额外的双重签名
2. FUS 自我升级：
   - 仅由意法半导体加密和签名
   - 或者，如果需要，客户还可以进行额外的双重签名
3. 客户验证密钥管理：
   - 用于对映像进行双重签名
   - 安装、更新和锁定客户验证密钥
4. 用户密钥管理：
   - 存储客户密钥
     - 主密钥
     - 简单的明文密钥
     - 加密密钥（通过主密钥）
     - 在安全区域中，只能通过 Cortex®-M0+代码来访问。
   - 在安全模式下将存储的密钥（简单或加密）写入到 AES1（高级加密标准）（Cortex®-M4 无法访问此密钥）
   - 锁定存储的密钥，以防其在下一次系统复位之前被使用
   - 从 AES 中卸载之前加载的密钥，以防被其他应用程序使用
   - 密钥宽度：128 或者 256 位
   - 最多 100 个用户密钥（通过主密钥或明文密钥加密）和 1 个用户主密钥
5. 与 Cortex®-M4（用户代码或自举程序）通信：
   - 通过 IPCC 指令和响应模式（与无线协议栈模式相同）
   - STM32WB 自举程序已支持的指令（在 ROM 中）

1.2 FUS 版本管理和识别

用户需要根据第 1.6 节“共享表存储器使用情况”和第 6.1 节“共享表使用情况”所述读取 SRAM2a 中的共享表存储器，以识别 FUS 版本。

IPCCDBA 选项字节指定的 SRAM2a 中的第一个字是“器件信息表”地址。该表（如表 7 所描述）在偏移 0xC 处包含 FUS 版本，版本信息以四个字节编码。通常情况下，如果 IPCCDBA=0x0000，且@0x20030000 包含 0x20030024，则 FUS 版本为@0x20030030。

FUS 映像的安装必须遵循映像（二进制版本）说明中规定的条件。

注意： *当 SWD 接口结合 V2.7.0 以前版本的 STM32CubeProgrammer（STM32CubeProg）使用时，设备信息表的地址位于 0x20030890。对于 STM32CubeProgrammer V2.7.0 及更高版本，设备信息表则位于 0x20030024。*

表 1. FUS 版

版本	说明
V0.5.3	在生产时为所有 STM32WB5xx 设备编程的默认版本。 必须升级到 V1.0.1（STM32WB5xG 器件）或 V1.0.2（STM32WB5xE/5xC 器件）。 www.st.com 上不提供此版本下载，用户无法安装。
V1.0.1	www.st.com 上提供的首个官方版本，专用于 STM32WB5xG 器件（Flash 存储器大小为 1 MB） 此版本不能安装在 STM32WB5xE/5xC 器件上，否则器件将进入锁定状态，无法进行进一步更新。
V1.0.2	www.st.com 上提供的首个官方版本，专用于 STM32WB5xE/5xC 器件（Flash 存储器大小为 512 KB 和 256 KB） 如果器件采用 FUS V0.5.3，则在 STM32WB5xG 器件上使用 V1.0.2 版本。 如果 STM32WB5xG 器件的 FUS 版本为 V1.0.1，则不需要升级到 V1.0.2；因为与 V1.0.1 相比，该版本无任何新特性/变化。 如果用户在 STM32WB5xG 器件（FUS 版本为 V1.0.1）上启动 FUS V1.0.2 的安装，则 FUS 会返回 FUS_STATE_IMG_NOT_AUTHENTIC 错误并放弃升级。
V1.1.0	FUS 更新以支持以下功能： • 增加 FUS_ACTIVATE_ANTIROLLBACK 指令，允许用户激活无线协议栈的防回滚功能。 • 用户可以激活此功能，以防安装旧的无线协议栈。 • 将安全启动替换为 V1.1.0 版本（以恢复出厂设置取代完全芯片锁定） 如果出现 Flash ECC、损坏或选项字节损坏错误，则添加恢复出厂设置。 恢复出厂设置意味着删除无线协议栈（如果有），通过 FUS 重新启动，完全删除其他用户扇区。 FUS V1.1.0 只能安装在 FUS 版本为 V1.0.1 或 V1.0.2 的器件上。 如果器件的 FUS 版本为 V0.5.3，用户应先安装 V1.0.2，然后再安装 V1.1.0。 当在 FUS V0.5.3 基础上安装 FUS V1.1.0 时，会导致 FUS_STATE_IMAGE_NOT_AUTHENTIC 错误并放弃升级。
V1.1.1	FUS 更新以支持 STM32WB5xx 640 KB 产品。 www.st.com 上不提供此版本下载，无法用于升级。 此版本完全兼容 V1.1.0，除了新的 640 KB 产品的管理，无任何区别。
V1.1.2	FUS 更新旨在： • 优化 Flash 使用：可以安装协议栈，从而在之前安装的协议栈下方保持一个扇区分离（而不是第 2 节“无线协议栈映像操作”中所述的协议栈大小空间限制） • 增强安全性 要从 FUS V1.1.0 升级到 FUS V1.1.2，必须首先激活防回滚功能。激活防回滚功能前，必须已安装无线协议栈。 可无限制地从 V1.1.0 升级到 V1.2.0，也无需用户执行附加操作。

版本	说明
V1.2.0	FUS 更新旨在： • 将 V1.1.2 FUS 更新纳入到生产中 • 允许直接从 FUS V1.1.0 更新到 FUS V1.2.0，无需激活防回滚功能。 • 允许直接从 FUS V0.5.3 更新到 FUS V1.2.0（无需安装中间的 FUS 版本） • 安全更新 可无限制地从 FUS V1.1.0 或任何其他 FUS 版本升级到 FUS V1.2.0，且无需用户交互。

下表详细说明了 FUS 版本兼容性选项（当可以从一个版本升级到另一版本时）。FUS V1.2.0 是允许从任何以前版本进行升级的版本。它采用两种二进制形式发布：

- stm32wb5x_fus_fw_V1.2.0.bin：用于从任何 FUS 版本 V1.x.y 的升级
- stm32wb5x_fus_fw_V1.2.0_for_V0.5.3.bin：从 FUS 版本 V0.5.3 升级

注意： *STM32WB10xx 和 STM32WB15xx 仅采用 FUS V1.2.0，它完全兼容 STM32WB5xxx FUS V1.2.0，但是不提供用户密钥服务。*

表 2. FUS 版本兼容性

升级		目标版本					
		V0.5.3	V1.0.2	V1.1.0	V1.1.1	V1.1.2	V1.2.0
原始版本	V0.5.3	X	√	X	X	X	√
	V1.0.2	X	X	√	X	√	√
	V1.1.0	X	X	**X**	X	(1)	√
	V1.1.1	X	X	X	X	√	√
	V1.1.2	X	X	X	X	√	√
	V1.2.0	X	X	X	X	X	√

图例：
- X：无法升级
- √：可升级
- **X**：不得升级，否则加密密钥会丢失
- (1)：可升级，但需要首先安装 BLE 协议栈并启用防回滚功能

FUS 版本可来自两个不同来源：

- 意法半导体在生产阶段在 STM32WB 系列器件中直接编程。
- 可从 www.st.com 获取。这种方式主要用于 FUS 版本升级过程。

下表详细说明了每个版本在生产时以及 www.st.com 上的可用性。

表 3. FUS 版本可用性

版本	生产	www.st.com 上的二进制
V0.5.3	√	X
V1.0.2	√	√
V1.1.0	√	√
V1.1.1	√	X
V1.1.2	X	√
V1.2.0	√	√
图例： • X：不可用 • √：可提供		

1.2.1 已知限制

本节详细说明了最新版 FUS（V1.2.0）的已知限制。

- 升级错误
  发生外部断电或复位事件时可能会发生升级错误。这是因为在固件升级操作期间发生外部断电或者强制复位。当前操作可能会受到破坏。这种情况下，FUS 会中止操作并返回错误消息。解决方案：当返回错误消息时，从头开始重复固件升级操作。OTA（无线升级）时必须特别谨慎，因为再次下载映像可能需要无线协议栈。
- 无线协议栈
  如果已安装无线协议栈“A”，则必须安装刚好比无线协议栈“A”大一个扇区的另一无线协议栈“B”。接着，除非将无线协议栈“B”加载到地址“add” < 0x080F4000 - 3x SizeOf（“B”）中，否则 FUS 会拒绝操作。解决方案是在无线协议栈中添加填充，以避免大小相差一个扇区的情况。当使用 STM32CubeWB V1.14.0 之前采用的一些无线协议栈时，就会遇到这种限制。有关更多详细信息，请参见 STM32CubeWB 版本说明。从 STM32CubeWB v1.14.1 版本起，将控制无线固件二进制的大小，以保证所有生成的二进制之间至少相差两个扇区大小，从而解决此限制。

1.3 如何激活 FUS

FUS 在 Cortex®-M0+上以及专用于 FUS 和无线协议栈的受保护 Flash 存储区上运行。有两种可能的情况：

表 4. FUS 激活案例

情况	如何激活 FUS
无线协议栈未运行（例如，STM32WB 系列器件首次运行或者已移除无线协议栈）	确保已通过置位 PWR_CR4 寄存器中的 C2BOOT 位来激活 Cortex®-M0+ 确保 IPCCDBA（选项字节）指向 SRAM2a 中的有效共享表信息结构（填入指向器件信息表和系统表的正确指针） *注意：　两种情况均由系统自举程序自动执行。因此，如果在系统存储器上配置了器件启动，则必须激活 FUS，而无需进一步的用户操作。* 否则，必须通过在 Cortex®-M4 CPU 上运行的用户代码来执行这些操作。
无线协议栈已安装并正在运行	执行与上面相同的步骤 请求无线协议栈通过发送两条连续的 FUS_GET_STATE 指令来启动 FUS。第一条指令必须返回 FUS_STATE_NOT_RUNNING 状态，第二条指令会使 FUS 启动。

为检查 FUS 是否正在运行，可使用以下选项：

- 发送一条 FUS_GET_STATE 指令，并检查返回的状态。如果为 FUS_STATE_NOT_RUNNING，则 FUS 未运行。
- 检查 SBRV 选项字节值：
  - 如果是 0x3D800（用于 FUS V0.5.3）或 0x3D000（用于 FUS V1.x.z），则 FUS 必须运行
  - 如果与 0x3D800（用于 FUS V0.5.3）或 0x3D000（用于 FUS V1.x.z）不同，则 FUS 不运行
- 发送无线协议栈指令：
  - 如果得到确认，则 FUS 未运行
  - 如果未得到确认，则 FUS 正在运行
- 读取共享表信息：
  - 读取 IPCCDBA（在选项字节中），以获取 SRAM2a 中的共享表起始地址
  - 获取器件信息表地址
  - 读取“最后的 FUS 活动状态”字段
    - 0x04 意味着无线协议栈一定正在运行
    - 其他值意味着 FUS 一定正在运行
  - 启动时读取 FUS 发送的“Async Ready”事件。有关此事件和内容的更多信息，请参考第 6.3.2 节“事件数据包”。

1.4 存储器映射

FUS 在 Flash 存储器中有一个专用空间，该空间取决于 FUS 大小。它还使用 SRAM2a 和 SRAM2b 中的专用空间以及 SRAM2a（共享表）中的共享空间。由选项字节定义 Flash 存储器 SRAM2a 和 SRAM2b 中的专用空间大小。若需更多信息，请参见产品参考手册。

与无线协议栈（如果安装）共享专用 Flash 存储器和 SRAM 区域。但在给定时间，只能在 Cortex®-M0+上运行 FUS 或无线协议栈。

图 1. Flash 存储器映射

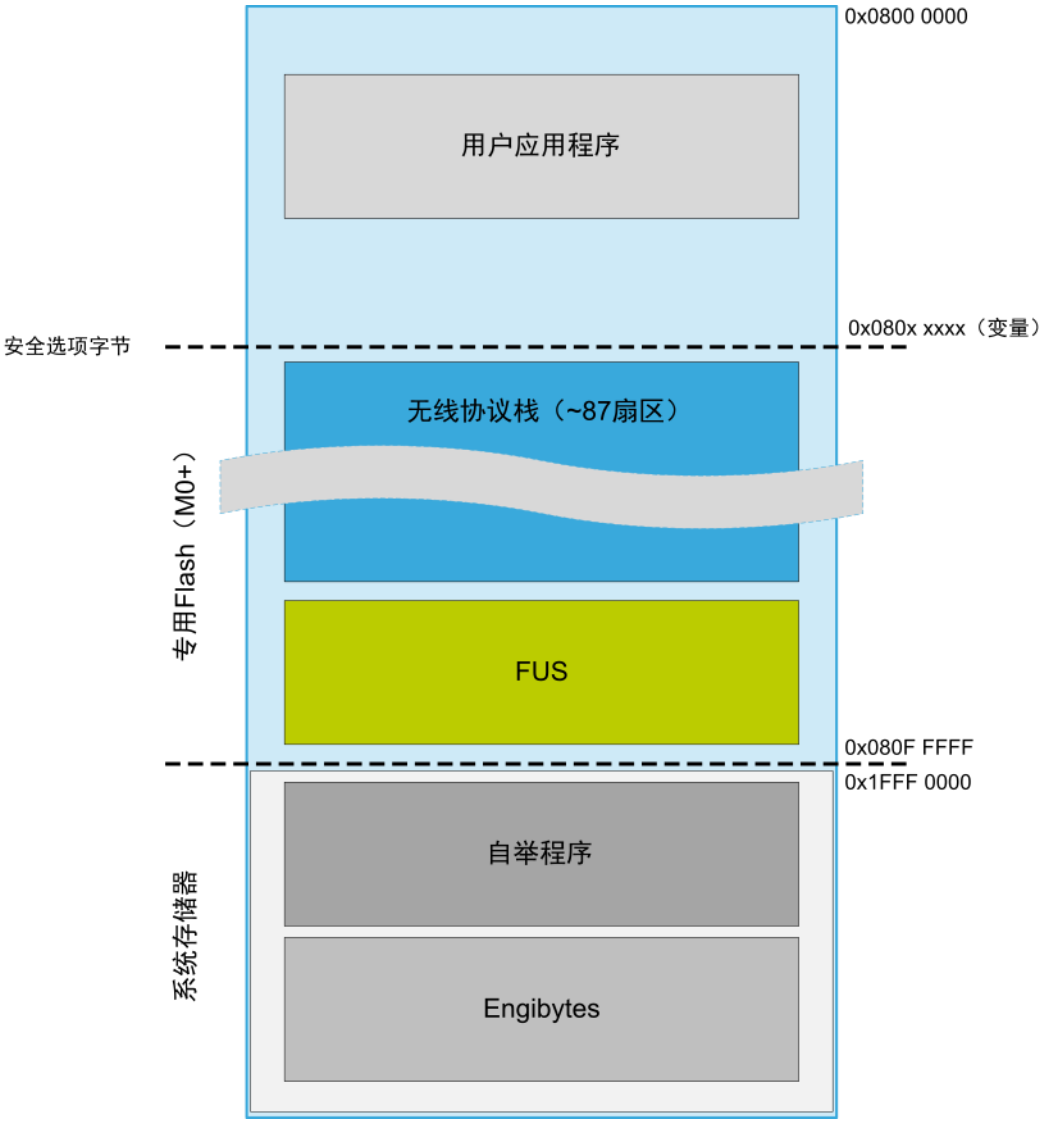
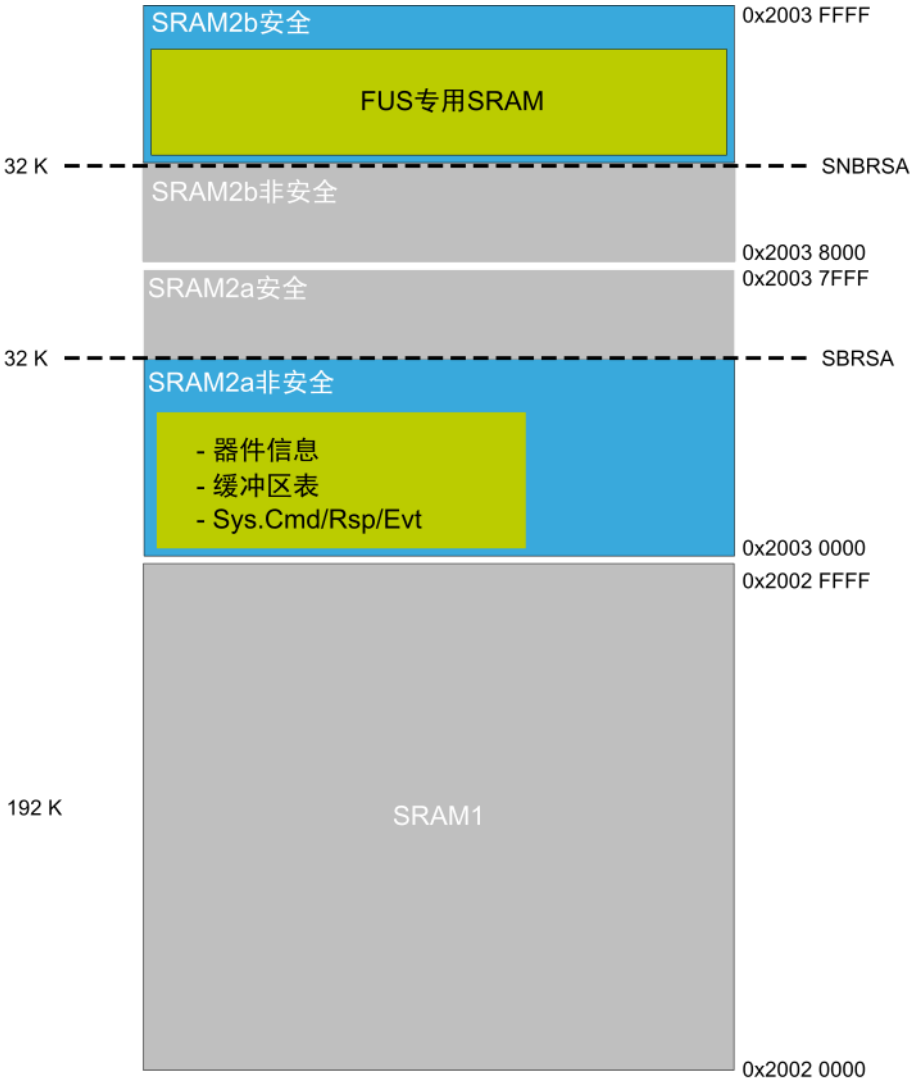


图 2. SRAM 存储器映射



1.5 FUS 资源使用情况

FUS 仅配置/使用表 5 中列出的资源。

在使能 Cortex®-M0+之前，必须通过 Cortex®-M4 应用程序来配置 RCC（复位和时钟控制）、Flash 存储器、PWR（电源控制）以及 STM32WB 系列微控制器正常操作所需的所有必要组件（这在启动时由系统自举程序自动完成）。

表 5. FUS 资源使用情况

资源	用例	配置
Flash	始终	FUS 根据其大小、当前无线协议栈的大小以及请求安装的映像使用专用 Flash 存储器。 在 FUS 操作期间，可以写入和/或擦除部分专用 Flash 存储器。 **小心：** 在 FUS 运行时，注意在 Flash 存储器上执行写/擦除周期的操作。
SRAM2b	始终	FUS 根据其版本使用 SRAM2b 安全区域。
SRAM2a	始终	FUS 根据其版本使用 SRAM2a 安全区域。 FUS 使用 SRAM2a 公共区域写入信息表和指令表的共享表。
IPCC	始终	FUS 将 IPCC 用于 Cortex®-M0+和 Cortex®-M4 用户应用程序或自举程序或 JTAG 之间的邮箱服务。 使用两个通道：P1CH2（指令/响应通道）和 P1CH4（跟踪通道）。
PKA	需要安装时	启用、配置 PKA 并将其用于签名验证。
AES1	需要密钥服务时	将 AES1 配置为安全模式（仅 Cortex®-M0+可访问密钥寄存器） FUS 将用户请求的密钥写入到 AES1 密钥寄存器。 将 AES1 配置为安全模式后，将始终保持在安全模式下，直到下次系统复位或执行 FUS_UNLOAD_USR_KEY 指令。
选项字节	需要安装/删除时	FUS 使用 Cortex®-M0+寄存器对选项字节进行编程：仅修改 SFR 和 SBRR 寄存器。
CRC	需要安装时	CRC 用于身份验证，FUS 不会对其进行初始化。如果 Cortex®-M4 用户应用程序使用 CRC，则必须在启动 FUS 或无线协议栈安装操作前复位此 CRC。
系统复位	需要安装/删除时	当加载选项字节或检测到严重错误后，FUS 会强制系统复位。
NVIC	始终	使用以下处理程序： • NMI • SysTick • IPCC_C2_RX_C2_TX_HSEM

重要： *在 FUS 或无线协议栈升级/删除操作期间，Cortex®-M4 和 SWD 不得：*

- *在 Flash 上执行任何写入/擦除操作*
- *对选项字节执行任何写入操作*
- *更改 PWR 和 RCC 配置。*

如果在 FUS 或无线协议栈升级/删除期间执行任何上述操作，则会带来损坏 Flash 和丢失数据的风险。

重要： *如果在FUS 操作（安装/删除）期间发生电源故障，则可能会出现以下3 种情况之一：*

- *无影响的电源故障：如果Flash 内容未损坏，FUS 将恢复故障并继续操作，无需任何用户干预。*
- *Flash 损坏的电源故障：Flash 内容损坏，FUS 未安装映像（作为非整数被拒绝）。FUS 擦除映像并生成错误（FUS_ERR_IMG_CORRUPT）。用户必须重新加载二进制，以重新开始整个操作，并将升级指令发送到FUS。*
- *选项字节损坏的电源故障：由硬件启动安全模式，且所有 Flash 被硬件锁定。这种情况下，如果 FUS V1.1.0 或更高版本正在运行，则会触发恢复出厂设置（用户应将值 0x00008000 写入到地址@0x5800040C，从而激活CM0）。如果低于V1.1.0 的FUS 版本正在运行，则此时无法进行恢复。*

注意： *如果FUS 存储了用户密钥，当升级FUS 时，将擦除这些密钥。*

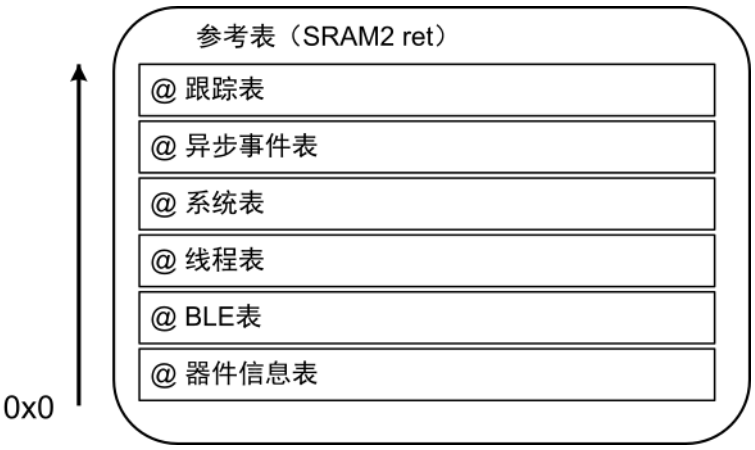
1.6 共享表存储器使用情况

通信数据缓冲区通过查找表指定，该查找表的地址由选项字节：IPCCDBA（IPCC 邮箱数据缓冲区基址）确定。该地址提供了缓冲区表指针的基址：如构建无线应用（AN5289）中所述。

如果 IPCCDBA 指向并非适合所有表指针的地址，如（SRAM2a_END_ADDRESS - SharedTable_BaseAddress）< SizeOf（SharedTable），则 FUS 必须完全放弃使用共享表，因此无法与 FUS 之间进行通信或发布指令。

用户应用程序必须正确设置共享表基址，否则，它必须停止 FUS 服务初始化。

图 3. 共享表架构



FUS 仅使用两个表：

- 器件信息表：该表提供在启动时从 FUS 发送到 Cortex®-M4 用户应用程序（或 JTAG）的有用信息（启动时由 FUS 写入的内容）。
- 系统表：该表允许在 FUS 和 Cortex®-M4 用户应用程序之间交换指令和响应。

2 无线协议栈映像操作

FUS 允许用户安装、升级和删除无线协议栈。

无线协议栈必须由意法半导体提供（加密并签名），以便由 FUS 安装。用户可以使用 ST 工具根据第 4 节“用户身份验证”所述，将自定义签名添加到无线协议栈映像二进制中（如果 FUS 已加载用户身份验证密钥）。

无线协议栈的安装、升级和删除操作通过自举程序、JTAG 或用户应用程序来执行。STM32CubeProgrammer 提供了通过自举程序接口执行该操作的工具：USART 和 USB-DFU，也可直接通过 SWD 接口执行。

2.1 无线协议栈安装和升级

以下需要切记的两个有用定义：

- 无线协议栈安装意味着先安装在没有安装无线协议栈的芯片上。
- 无线协议栈升级：意味着安装到已装有无线协议栈（可能正在运行，也可能未运行）的芯片上。

操作说明

为了执行无线协议栈安装或升级，请遵循以下步骤：

1. 从 www.st.com 或从 STM32CubeMX 存储库下载无线协议栈映像。
2. 将无线协议栈映像写入用户 Flash 存储器的以下地址中：DownloadAddress = 0x08000000 +（SFSA x SectorSize）- ImageSize - 1 x SectorSize，其中：
   - DownloadAddress 是与扇区大小一致、将加载新无线协议栈的地址
   - SFSA 是指示 Flash 存储器安全区当前边界的安全选项寄存器值
   - 对于 STM32WB5xxx，SectorSize 为 4 KB，对于 STM32WB1xxx，SectorSize 为 2 KB
   - ImageSize 是将安装的二进制大小（字节）

   使用 STM32CubeProgrammer 时，会基于 SFSA 和二进制大小建议最佳位置。

   开始安装新的无线协议栈前，最好执行 FUS_FW_DELETE 操作，但并非必须执行此操作。在 STM32CubeProgrammer 上选择“-firstinstall=0”选项或取消选中“First Install”（首次安装）框，将强制删除。

   如果新的无线协议栈大小大于已经安装的协议栈，请阅读第 2.2 节“无线协议栈删除”中的“存储器说明”。
3. 确保 FUS 正在运行（请参考第 1.3 节“如何激活 FUS”中的步骤）。
4. 通过 IPCC 机制发送 FUS_FW_UPGRADE 指令（在以下章节中解释）。
5. 发送 FUS_GET_STATE，直至获得 FUS_STATE_NOT_RUNNING 的状态（这意味着无线协议栈已安装并正在运行）。

在安装过程中，预计会执行多次系统复位。这些系统复位由 FUS 执行，为修改专用存储器参数和使 Cortex®-M0+运行已安装的无线协议栈所必须。系统复位次数取决于配置和新旧映像的位置。

表 6 说明了请求安装/升级操作时可能出现的错误及其各自的结果。

表 6. FUS 升级返回错误

误差	原因	结果
空间不足	当前安装的无线协议栈与所加载映像的地址之间的空间过小。	安装请求被拒绝。FUS 返回错误状态并回到空闲状态。
未发现映像签名	签名的头文件或主体不正确或损坏。	FUS 返回验证错误，然后回到空闲状态。未安装映像，Flash 存储器/SRAM 没有变化。
未发现映像客户签名	签名的头文件或主体不正确或损坏。	FUS 返回验证错误，然后回到空闲状态。未安装映像，Flash 存储器/SRAM 没有变化。

误差	原因	结果
映像已损坏	映像头文件不正确或映像损坏。	FUS 返回映像损坏错误，然后回到空闲状态。映像未安装和被 FUS 擦除。
FUS 未返回任何状态	在收到指令响应之前，FUS 已执行复位。	重发指令必须收到 FUS 响应。
其他故障	在 FUS 操作期间，外部电源中断或外部复位	FUS 一定能够恢复和删除损坏的映像，并返回默认状态。它可能执行多次系统复位，以便完成恢复操作。

存储器方面的考虑

首次安装或未安装无线协议栈时，FUS 不会对安装无线协议栈的地址进行任何优化。无线协议栈映像必须安装在用户加载它的那个地址中。

在无线协议栈升级时（已安装无线协议栈），FUS 可能会在升级无线协议栈之后和运行无线协议栈之前移动升级后的无线协议栈。

在这种情况下，剩余的空间可供 Cortex®-M4 用户应用程序使用。

在安装/升级操作成功完成后，SRAM2a、SRAM2b、Flash 存储器安全边界和 SBRV 值将根据已安装的无线协议栈的要求进行更改。

2.2 无线协议栈删除

无线协议栈删除意味着删除已安装在芯片上的无线协议栈（不论其是否正在运行）。

操作说明

要执行无线协议栈删除，请执行以下步骤：

1. 确保 FUS 正在运行（请参考第 1.3 节“如何激活 FUS”中的步骤）
2. 通过 IPCC 机制发送 FUS_FW_DELETE 指令（在以下章节中解释）
3. 发送 FUS_GET_STATE，直至达到 FUS_STATE_IDLE 并收到错误代码 FUS_STATE_NO_ERROR。在删除过程中，预计会执行多次系统复位。这些系统复位由 FUS 执行，为修改专用存储器参数所必须。系统复位次数取决于无线协议栈的配置和位置。

如果在未安装无线协议栈的情况下发送一个删除请求，则 FUS 将返回错误状态，以通知未找到无线协议栈（FUS_STATE_IMG_NOT_FOUND）。

存储器方面的考虑

在成功完成删除操作后，无线协议栈使用的所有空间可供 Cortex®-M4 用户应用程序使用，或用于进一步的无线协议栈安装操作。

映像起始地址必须与扇区起始地址一致（这是 4 k 字节的倍数），映像大小必须是 4 字节的倍数，否则 FUS 将拒绝安装过程。

当已安装协议栈“A”时，如果必须安装新的无线协议栈“B”，并且 B 的大小大于 A 的大小，则对于可以加载“B”的地址，有两种可能的选项：

- 条件 1（背对背选项）：（必须满足以下 C1 和 C2 两个条件）
  - C1.AddressOf(B) >（FUS_ADD -（2 x SizeOf(B)）（与扇区大小一致）
  - C2.AddressOf(B) < AddressOf(A) - SizeOf(B) （与扇区大小一致）
- 条件 2（非背对背选项）：（仅满足条件 C3）
  - C3.AddressOf(B) <（FUS_ADD -（3 x SizeOf(B)））（地址必须与扇区大小一致）

其中 FUS_ADD 是 Flash 中的 FUS 地址（对于 STM32WB5xxx，此地址为 0x080F4000，对于 STM32WB1xxx，地址为 0x08046000）

所有情况下，最优化的下载地址为：

DownloadAddress = 0x08000000 +（SFSA x SectorSize）- SizeOf(B) - 1xSectorSize

2.3 无线协议栈启动

可能存在无线协议栈已安装但当前未运行的情况。结果是：安装已完成，无线协议栈正在运行，然后用户应用程序连续发送两次 FUS_GET_STATE，使 FUS 重新启动。

在这种情况下，可通过发送 FUS_START_WS 指令来启动无线协议栈执行。该指令可将 Cortex®-M0+执行切换到无线协议栈，并导致至少一次系统复位。

当 FUS_GET_STATE 返回 FUS_STATE_NOT_RUNNING 值时，该指令完成。在收到该值后，不得发出其他 FUS_GET_STATE，否则将再次执行 FUS。

2.4 防回滚功能激活

如果 FUS 支持防回滚，可以向 FUS 发送指令来激活该功能。

一旦 FUS 执行此指令，没有方法可以禁用此功能。

该功能通过 FUS_ACTIVATE_ANTIROLLBACK 指令激活。

发送该指令后，可以发送 FUS_GET_STATE 指令查看其状态。FUS 随后应返回状态 FUS_STATE_IDLE。

该指令不可逆。该指令不适用于 FUS，因为 FUS 无法回滚。

重要： *激活防回滚功能前，请确保无线协议栈已正确安装，且未被删除。如果在未安装任何无线协议栈的情况下激活此功能，则 FUS 会将 0xFFFFFFFF 注册为新版本，并且无法安装无线协议栈。*

当“防回滚”功能激活时，它会锁定可安装的无线协议栈版本。

此时将无法安装版本低于当前版本的任何无线协议栈。例如，如果安装了无线协议栈 V1.9.0，当“防回滚”功能激活时，则只能安装 V1.9.0 及更高版本的无线协议栈（但是无法再安装 V1.8.0 版本）。

3 FUS 升级

FUS 能够以与无线协议栈升级相同的方式进行自我升级。FUS 无法删除。

3.1 操作说明

为了执行 FUS 升级，请执行以下步骤：

1. 从 www.st.com 或从 STM32CubeMx 存储库下载 FUS 映像。
2. 将 FUS 映像写入 FUS 映像目录 Release_Notes.html 文件所指示的用户 Flash 存储器地址中。
3. 确保 FUS 正在运行（请参考第 1.3 节“如何激活 FUS”中的步骤）。
4. 通过 IPCC 机制发送 FUS_FW_UPGRADE 指令（在以下章节中解释）。
5. 发送 FUS_GET_STATE，直至获得 FUS_STATE_NOT_RUNNING 的状态（这意味着无线协议栈已安装并正在运行）。

在安装过程中，为了修改专用存储器参数和使 Cortex®-M0+运行已安装的无线协议栈，预计需要由 FUS 执行多次系统复位。系统复位次数取决于配置和新旧映像的位置。

FUS 将映像识别为 FUS 升级映像，并相应地启动 FUS 升级。如果已安装固件协议栈且新 FUS 的大小大于当前 FUS 的大小，则此操作会导致重新安置固件协议栈。此信息以及任何相关限制详见 FUS 映射版本说明。

3.2 存储器方面的考虑

FUS 升级不需要特定存储器条件。但是，如果新的 FUS 映像大于现有的 FUS 映像，升级可能导致向 Flash 存储器向下移动无线协议栈，以便为 FUS 升级提供足够的空间。

这意味着：

- 可用于 Cortex®-M4 用户应用程序的 Flash 存储器空间更少。
- 无线协议栈从其当前地址移动到 FUS 定义的另一个地址。
- 如果在邻近无线协议栈起始扇区的扇区中写入用户代码，则在该操作期间用户代码可能会被擦除。

FUS 的大小及其升级结果在其 Release_Notes.html 文件中有详细说明。

映像起始地址必须与扇区起始地址一致（4 k 字节的倍数），映像大小必须是 4 字节的倍数，否则 FUS 将拒绝安装过程。

4 用户验证

FUS 服务使用户可以将自定义签名添加到意法半导体（由意法半导体加密和签名）提供的任何映像（无线协议栈或 FUS 映像）中。

STM32CubeProgrammer 用户手册中提供了使用用户验证密钥对二进制文件进行签名的说明。

FUS 仅在已安装用户验证密钥的情况下才会检查用户签名。

签名是基于 RSA ECC Prime256v1 (NIST P-256) 和 HASH-256 的 64 字节数据缓冲区。它是由 STM32CubeProgrammer 工具生成的。

4.1 安装用户验证密钥

FUS 允许通过以下步骤存储用户验证密钥：

1. 确保 FUS 正在运行（请参考第 1.3 节“如何激活 FUS”中的步骤）。
2. 通过 IPCC 机制发送 FUS_UPDATE_AUTH_KEY 指令（在以下章节中解释）
3. 发送 FUS_GET_STATE，直至获得相当于 FUS_STATE_IDLE 的状态。该操作不会产生任何系统复位。

安装用户验证密钥后，可使用与上面相同的流程进行更改（除非已完成锁定用户验证密钥操作）。但是不能删除。

安装该密钥后，FUS 必须在执行安装或升级前系统地检查二进制用户签名。如果签名不存在或者不可信，则拒绝安装或升级，其错误相当于 FUS_STATE_IMG_NOT_AUTHENTIC。

4.2 锁定用户验证密钥

FUS 允许锁定用户验证密钥。这意味着在整个产品生命周期中无法再更改此密钥。该操作在执行后无法撤销。

要锁定用户验证密钥：

1. 确保 FUS 正在运行（请参考第 1.3 节“如何激活 FUS”中的步骤）。
2. 通过 IPCC 机制发送 FUS_LOCK_AUTH_KEY 指令（在以下章节中进行解释）。
3. 发送 FUS_GET_STATE，直至获得相当于 FUS_STATE_IDLE 的状态。该操作不会产生任何系统复位。

完成该操作后，将锁定用户验证密钥。

5 客户密钥存储

FUS 允许将客户密钥存储在专用的 FUS Flash 存储区中，然后以安全模式将存储的密钥加载到 AES1 中（AES1 密钥寄存器只能由 Cortex®-M0+访问，数据寄存器只能由 Cortex®-M4 用户应用程序访问）。

5.1 密钥类型和结构

FUS 支持存储 101 个密钥（1 个主密钥和 100 个明文/加密密钥）

密钥大小可能为 128 或 256 位。所有类型的密钥都具有相同的密钥大小和结构。无法更改或删除任何存储的密钥。

FUS 支持三种类型的密钥：

- **明文密钥：**发送至 FUS 的未加密密钥。
- **主密钥：**该密钥以不加密方式发送给 FUS 并用于解密稍后将发送给 FUS 的其他密钥。必须在可信的环境（无法在通信路径上提取密钥的环境）中存储该密钥。主密钥使用户能够在不受信任的环境中共享加密密钥，而不会暴露其内容。主密钥无法写入 AES1 密钥寄存器。它专门用于解密，不能更改和删除。主密钥仅写入一次，之后绝不再更新。一旦主密钥写入完成，任何再次写入主密钥的请求都会被拒绝并显示错误消息。如果写入超过 100 个密钥，则会导致该指令被拒绝。
- **加密密钥：**以加密格式发送给 FUS 的密钥。然后，在使用该密钥之前，FUS 使用主密钥对其进行解密。此密钥必须附带一个 IV（初始化向量），以允许 FUS 对其解密。16 位的 IV 会与密钥本身通过相同的指令包发送。

用户密钥加密必须基于 AES-128 GCM 模式。FUS 在不使用硬件 AES 的情况下解密密钥。

必须通过用于发送密钥的指令数据包将密钥类型传达给 FUS（指令说明中有更多详细信息）。

密钥通过其索引来管理。

当密钥被发送到 FUS 时，FUS 确认收到密钥，并以密钥已分配的索引进行响应。该索引由 FUS 分配且无法通过用户应用程序进行更改。

要存储密钥，用户应用程序必须将 FUS_STORE_USR_KEY 发送到 FUS（具有密钥类型和相关 IV（如果存在）），然后接收密钥索引。

要使用存储的密钥，用户应用程序必须：

- 配置 AES1 初始化寄存器和 IV 寄存器。
- 将 FUS_LOAD_USR_KEY 发送到 FUS，并等待收到响应，这意味着密钥已被写入 AES1 密钥寄存器。
- 写入 AES1 数据寄存器，以使用存储的密钥解密/加密数据（密钥寄存器仍处于受保护状态，Cortex®-M4 用户应用程序无法访问）。如果写入的密钥数量超过 100，则该指令将被拒绝。

FUS 还提供两项额外服务，用于用户密钥管理。这两项服务仅供 Cortex®-M4 用户应用程序在安全应用程序上下文中使用，不会被自举程序或 STM32CubeProgrammer 影响。

用户密钥锁定

此服务将确保任何应用程序在下次器件复位前都无法再使用密钥（无法加载到 AES 中）。可以通过发送 FUS_LOCK_USR_KEY 指令来使用该服务，该指令包含待锁定密钥的索引（主密钥索引始终为 0，无法被锁定或加载）。

FUS_LOCK_USR_KEY 指令发送以后，FUS 将请求的密钥的状态存储为已锁定，为该密钥索引发出任何 FUS_LOAD_USR_KEY 指令将导致操作失败（指令响应返回 0x01）。

用户密钥卸载

该服务用于卸载 AES 中当前加载的密钥（如果已经使用 FUS_LOAD_USR_KEY）并防止用户应用程序使用加载的密钥进行任何进一步操作。

可以通过发送 FUS_UNLOAD_USR_KEY 指令来使用该服务，该指令包含待卸载密钥的索引（主密钥索引始终为 0，无法被加载或卸载）。

发送 FUS_UNLOAD_USR_KEY 指令之后，FUS 将 0 写入 AES 的密钥寄存器中，确保加载的密钥不再被使用。

6 与 FUS 通信

与 FUS 的通信通过 IPCC 通道和 Cortex®-M4 用户应用程序或自举程序或 JTAG 完成。在所有情况下，通信原则完全相同。

使用 STM32 系统自举程序与 FUS 进行通信时，通过直接使用 Bootloader 接口（USART 或 USB-DFU）可提供所有底层的抽象。

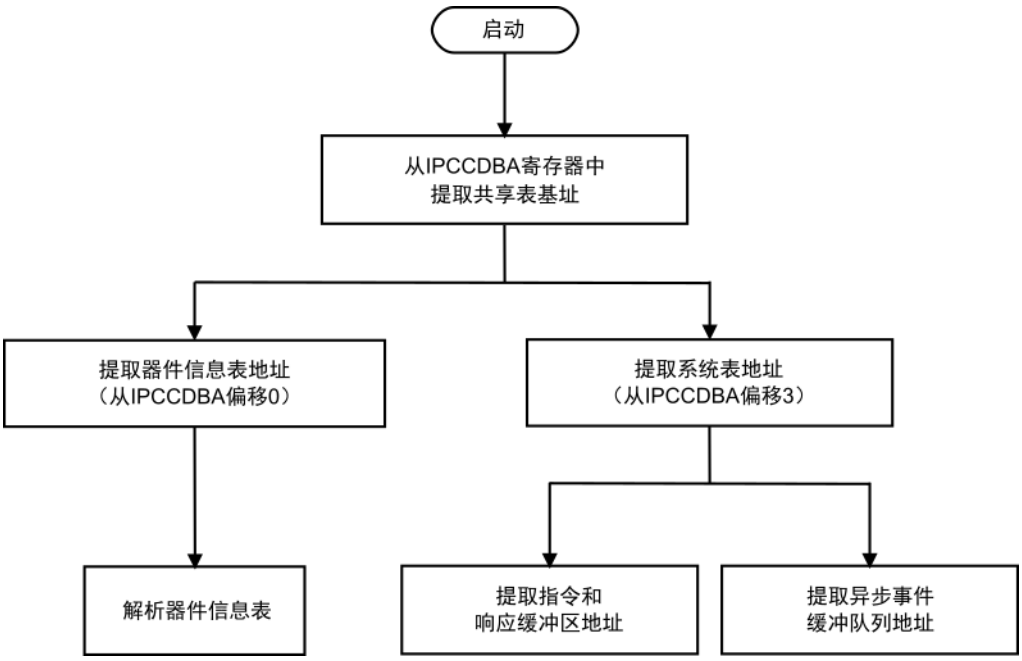
要与 FUS 通信，将要使用两个要素：

- 共享表：用于存储 FUS 的信息和获取指令/响应数据包。
- IPCC：用于交换消息通知（消息内容位于共享表中）。

6.1 共享表的使用

共享表是位于 SRAM2a 公共区域中的信息结构，有关此结构的说明，请参见后续小节。FUS 使用两个共享表，分别称为器件信息表和系统表。必须通过 Cortex®-M4 用户应用程序（或 JTAG 应用程序）来解析它们，以便与 FUS 正确通信。

图 4. 共享表使用过程



6.1.1 器件信息表

此表是一个 42 字节缓冲区，用于更新器件的当前状态。在启动时或已编程的系统复位之前可通过 FUS 代码或无线协议栈代码来更新该表。

表 7. 器件信息表

位域	大小（字节）	值
器件信息表状态	4	0xA94656B9：器件信息表有效 任何其他值：器件信息表无效
保留	1	保留
最后的 FUS 活动状态	1	• 0x00：FUS 空闲 • 0x01：无线协议栈固件升级 • 0x02：FUS 固件升级 • 0x03：FUS 服务 • 0x04：无线协议栈运行中 • 0x05-0xFE：未使用 • 0xFF：错误
最后的无线协议栈状态	1	0x00：未启动 0x01：运行中 0x08-0xFE：未使用 0xFF：错误
当前的无线协议栈类型	1	0x00：无 0x01：BLE 0x02：Thread 类型 1 0x03：Thread 类型 2 无线协议栈文档中提供了更多详细信息。
安全启动版本	4	固件版本： [31:24]：主版本（在向后兼容性被破坏时更新） [23:16]：次版本（添加主要功能时更新） [15:8]：子版本（针对细微更改的更新） [7:4]：分支（特定版本） [3:0]：构建（构建版本）
FUS 版	4	固件版本 [31:24]：主版本（在向后兼容性被破坏时更新） [23:16]：次版本（添加主要功能时更新） [15:8]：子版本（针对细微更改的更新） [7:4]：分支（特定版本） [3:0]：构建（构建版本）
FUS 存储器大小	4	当前的 FUS 栈存储器使用情况： [31:24]：SRAM2b：使用 n 个 1K 的扇区 [23:16]：SRAM2a：使用 n 个 1K 的扇区 [15:8]：保留 [14:0]：Flash 存储器：使用 n 个 4K 的扇区
无线协议栈版本	4	固件版本： [31:24]：主版本（在向后兼容性被破坏时更新） [23:16]：次版本（添加主要功能时更新） [15:8]：子版本（针对细微更改的更新） [7:4]：分支（特定版本） [3:0]：构建（构建版本） 当不存在无线协议栈时，所有数据均为 0xFFFF FFFF
无线协议栈存储器大小	4	当前的无线协议栈存储器使用情况： [32:24]：SRAM2b：使用 n 个 1K 的扇区 [23:16]：SRAM2a：使用 n 个 1K 的扇区 [15:8]：保留 [14:0]：Flash 存储器：使用 n 个 4K 的扇区 当不存在无线协议栈时，所有数据均为 0xFFFF FFFF

位域	大小（字节）	值
无线 FW-BLE 信息	4	[31:0]：为无线协议栈使用预留 当不存在无线协议栈时，所有数据均为 0xFFFF FFFF
无线 FW-thread 信息	4	[31:0]：为无线协议栈使用预留 当不存在无线协议栈时，所有数据均为 0xFFFF FFFF
保留	4	0x00000000
UID64	8	STM32 器件唯一的 64 位 ID
器件 ID	2	STM32 通用器件 ID

6.1.2 系统表

系统表为包含两个缓冲区指针的 8 字节表，如下表所示。

表 8. 系统表内容

地址	大小（字节）	目录	说明
0x00	4	系统指令/响应缓冲区的地址	在任何给定时间使用单个缓冲区，只需写入一个指令或其响应。响应将覆盖该指令。新指令会覆盖先前指令的任何响应。
0x04	4	系统事件队列缓冲区的地址（第一个事件的地址）	必要时必须解析 FUS 代码并将其装入队列。事件消息作为链表来管理，一旦 Cortex®-M4 已读取它们（通过 IPCC 来通知），就会将其释放。 仅通过其大小来解析事件。（不是链表结构），

为获得用于与 FUS 通信的有用信息，Cortex®-M4 代码（应用程序或自举程序）执行图 4 中所述的解析。

6.2 IPCC 的用法

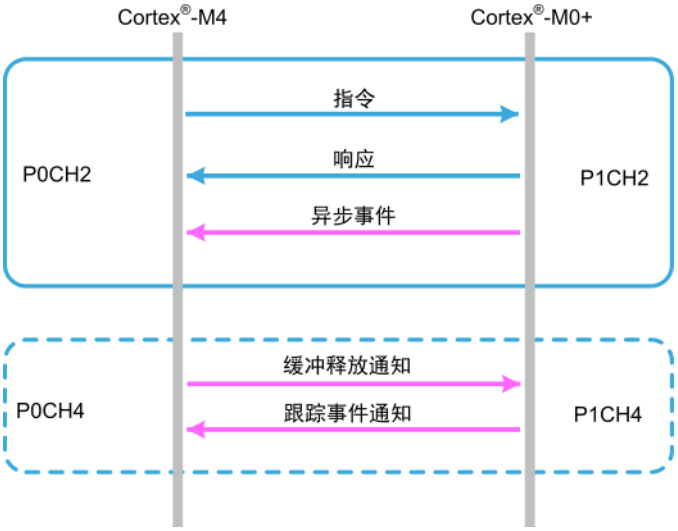
FUS 使用系统 IPCC 分配的通道：P0CH2（位于 Cortex®-M4 端）和 P1CH2（位于 Cortex®-M0+端）。这些通道提供三种通信方式：

- Cmd：从 Cortex®-M4 发送至 Cortex®-M0+的指令请求。此路径用于向 Cortex®-M0+发送指令。
- Rsp：响应从 Cortex®-M0+发送至 Cortex®-M4 的指令。此路径仅用于回应 Cortex®-M4 请求的指令。
- Asynch Evt:从 Cortex®-M0+发送至 Cortex®-M4 的异步事件。此路径用于将异步事件通知 Cortex®-M4，而无需 Cortex®-M4 回应此事件。

一些可选通道可供 FUS 使用：

- FUS（Cortex®-M0+）可使用 P1CH4 来输出跟踪事件
- Cortex®-M4 可使用 P0CH4 向 Cortex®-M0+通知缓冲区释放事件。

图 5. FUS 使用的 IPCC 通道

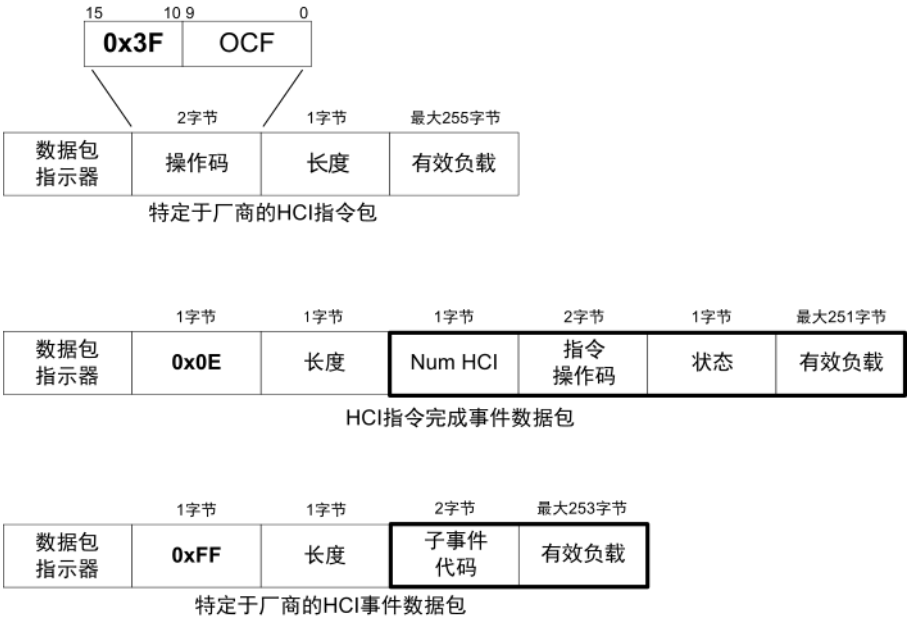


6.3 FUS 指令

FUS 使用与无线协议栈相同的指令/响应结构并基于 HCI 模型。FUS 使用 HCI 指令的子集，即：

- 厂商特定 HCI 指令数据包：用于将指令从 Cortex®-M4 发送到 Cortex®-M0+。
- HCI 指令完成事件数据包：用于将响应从 Cortex®-M0+发送到 Cortex®-M4
- 厂商特定 HCI 事件数据包：用于将异步事件从 Cortex®-M0+发送到 Cortex®-M4。

图 6. FUS HCI 子集



6.3.1 数据包指示器

数据包指示器具有一个字节，其值取决于数据包类型。

表 9. 数据包指示器的值

数据包类型	数据包指示器的值
指令数据包	0x10
响应数据包	0x11
事件数据包	0x12

6.3.2 事件数据包

FUS 仅发送一个异步事件。此事件仅在启动 FUS 时发送。长度字段表示 SubEvtCode+有效负载的长度。

表 10. FUS 异步事件（供应商特定的 HCI 事件）

长度	SubEvtCode	有效负载	意义
3	0x9200	错误代码： • 0x00：无线协议栈运行中 • 0x01：FUS 正在运行 • 0x02：SW 错误 • 0x03 至 0xFF：未使用	FUS 初始化阶段完成，错误代码显示在有效负载字节中。

6.3.3 指令数据包

下表详细列出了 FUS 支持的所有指令及其 HCI 格式值。

表 11. FUS 指令（厂商特定 HCI 指令包）

指令	操作码	长度（字节）	有效负载
保留	0xFC00	N/A	N/A
FUS_GET_STATE	0xFC52	0	无
保留	0xFC53	N/A	N/A
FUS_FW_UPGRADE	0xFC54	0 / 4[1] / 8 [2]	无 （可选 4 字节）固件映像位置地址 （可选 8 字节）固件目标地址
FUS_FW_DELETE	0xFC55	0	无
FUS_UPDATE_AUTH_KEY	0xFC56	最多 65 个	字节 0：验证密钥字节数 N 字节 1-字节 N-1：验证密钥数据
FUS_LOCK_AUTH_KEY	0xFC57	0	无
FUS_STORE_USR_KEY	0xFC58	N + 2	字节 0：密钥类型： • 0x00：无 • 0x01：简单密钥 • 0x02：主密钥 • 0x03：加密密钥 字节 1：密钥字节数 N 字节 2-字节 N-1：密钥数据（密钥值 + IV（如果存在））
FUS_LOAD_USR_KEY	0xFC59	1	字节 0：密钥索引（从 0 到 124）
FUS_START_WS	0xFC5A	0	无

指令	操作码	长度（字节）	有效负载
FUS_LOCK_USR_KEY	0xFC5D	1	1 字节，待锁定的密钥索引
FUS_UNLOAD_USR_KEY	0xFC5E	1	1 字节，待卸载的密钥索引
FUS_ACTIVATE_ANTIROLLBACK	0xFC5F	0	无
保留	0xFC60 至 0xFCFF	N/A	N/A

1. *4 字节，未用于当前版本。*
2. *8 字节，未用于当前版本。*

6.3.4 响应数据包

对于每个指令数据包，FUS 发送的响应数据包包含下表中详细列出的信息。NumHCI 字段值始终被设为 0xFF。长度字段表示 NumHCI+CmdOpcode+状态+有效负载的长度。因此，如果没有有效负载，则长度值为 4。

表 12. FUS 响应（HCI 指令完成数据包）

状态	长度	Cmd 操作码值	状态值	有效负载
FUS_STATE	5	0xFC52	表 FUS 状态值中的值	表 FUS 状态错误值中的值。
FW_UPGRADE_STATE	4	0xFC54	• 0x00：开始操作 • 0x01：失败 • 0x02-0xFF：未使用	无
FW_DELETE_STATE	4	0xFC55		无
UPDATE_AUTH_KEY_STATE	4	0xFC56	• 0x00：操作完成 • 0x01：失败 • 0x02-0xFF：未使用	无
LOCK_AUTH_KEY_STATE	4	0xFC57		无
STORE_USR_KEY_STATE	5	0xFC58		1 字节：密钥索引（从 0 到 100）
LOAD_USR_KEY_STATE	4	0xFC59		无
FUS_START_WS_START	4	0xFC5A	• 0x00：开始操作 • 0x01：失败 • 0x02-0xFF：未使用	无
FUS_LOCK_USR_KEY	4	0xFC5D	• 0x00：操作完成 • 0x01：失败 • 0x02-0xFF：未使用	无
FUS_UNLOAD_USR_KEY	4	0xFC5E	• 0x00：操作完成 • 0x01：失败 • 0x02-0xFF：未使用	无
FUS_ACTIVATE_ANTIROLLBACK	4	0xFC5F	• 0x00：操作完成 • 0x01：失败 • 0x02-0xFF：未使用	无

下表中详细列出了 FUS 响应状态值。有些值以范围形式表示（例如 0x10 到 0x1F），这意味着该范围内的所有值具有相同的状态含义（例如，0x12 或 0x1E 都表示 FUS_STATE_FW_UPGRD_ONGOING）。此范围内的值为将来的协议扩展预留。

表 13. FUS 状态值

值	名称	意义
0x00	FUS_STATE_IDLE	FUS 处于空闲状态。上次操作成功完成并返回其状态。没有正在进行的操作。
0x01..0x0F	未使用	保留这些值，以供将来使用。
0x10..0x1F	FUS_STATE_FW_UPGRD_ONGOING	固件升级操作正在进行中。
0x20..0x2F	FUS_STATE_FUS_UPGRD_ONGOING	FUS 升级操作正在进行中。
0x30..0x3F	FUS_STATE_SERVICE_ONGOING	服务正在运行：验证密钥服务（更新/锁定）或用户密钥服务（存储/载入）。
0x40..0xFE	未使用	保留这些值，以供将来使用。
0xFF	FUS_STATE_ERROR	发生错误。有关错误来源的更多详情，请参见响应有效负载。

表 14. FUS 状态错误值

值	名称	意义
0x00	FUS_STATE_NO_ERROR	未发生错误。
0x01	FUS_STATE_IMG_NOT_FOUND	已请求固件/FUS 升级，但未找到映像。（如映像头文件已损坏或 Flash 存储器已损坏）
0x02	FUS_SATE_IMC_CORRUPT	已请求固件/FUS 升级，找到映像，真实但不完整（数据损坏）
0x03	FUS_STATE_IMG_NOT_AUTHENTIC	已请求固件/FUS 升级，找到映像，但其签名无效（签名错误，签名头文件错误）
0x04	FUS_SATE_NO_ENOUGH_SPACE	已请求固件/FUS 升级，找到映像且映像真实，但由于已安装的映像，没有足够的空间来安装映像。将协议栈安装在较低位置，然后重试。
0x05	FUS_IMAGE_USRABORT	用户中止操作，或发生关断
0x06	FUS_IMAGE_ERSERROR	Flash 擦除错误
0x07	FUS_IMAGE_WRTERROR	Flash 写入错误
0x08	FUS_AUTH_TAG_ST_NOTFOUND	在映像中未发现意法半导体签名
0x09	FUS_AUTH_TAG_CUST_NOTFOUND	在映像中未发现客户签名
0x0A	FUS_AUTH_KEY_LOCKED	用户试图加载的密钥当前已被锁定
0x11	FUS_FW_ROLLBACK_ERROR	检测到向旧固件版本回滚，且不允许回滚
0x12..0xFD	N/A	保留供将来使用。
0xFE	FUS_STATE_NOT_RUNNING	FUS 当前未运行，无线协议栈正在运行并返回此状态。
0xFF	FUS_STATE_ERR_UNKOWN	未知错误

6.4 映像包尾

每个映像升级元素都有自己的包尾：

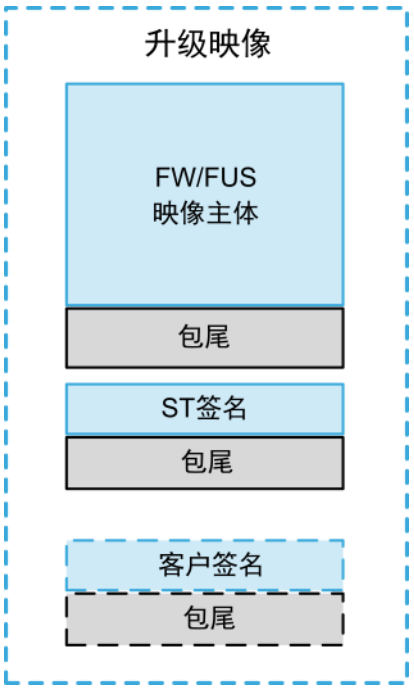
- 映像主体
- 意法半导体签名（强制性元素）
- 客户签名（可选元素）

尾文件必须直接位于作为尾文件的相对元素的末尾（例如，映像主体的头文件地址必须邻接映像主体地址）

身份验证标签没有这种邻接性要求，因此无需置于映像旁边。它们位于 Flash 存储器中的任何位置。FUS 独立于映像位置来寻找它们。

所有映像和包尾地址及大小均必须为 4 字节倍数和 4 字节对齐，否则 FUS 无法识别它们。

图 7. 映像包尾布置



每个包尾均包含一个标识值，以便 FUS 能够识别它。

图 8. FW/FUS 升级映像包尾结构

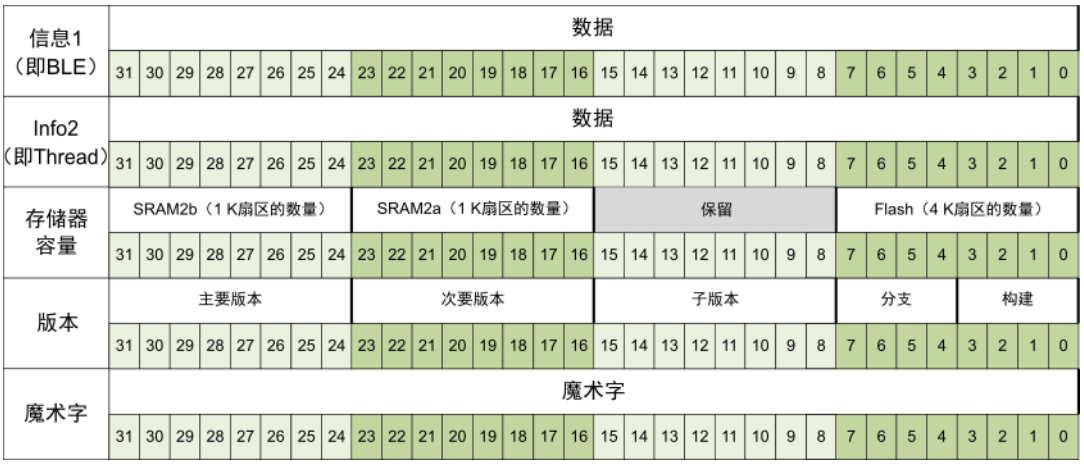


表 15. 映像包尾结构解析

位域	意义
Info1	特定于无线协议栈/FUS 映像
Info2	特定于无线协议栈/FUS 映像
Flash 存储器	映像总大小以 4 KB 的倍数表示
SRAM2a	映像所需的 SRAM2a 安全区总空间
SRAM2b	映像所需的 SRAM2b 安全区总空间
构建	版本构建号
分支	版本分支号
子版本	版本控制号
次版本	次版本号
主版本	主版本号
魔术字	用于识别映像性质的特定值。

注意： *FUS V1.2.0 版本在二进制中写为 0xFFFFFFFF，以便能够从所有版本 FUS 进行升级。*

图 9. 签名（标签）包尾结构

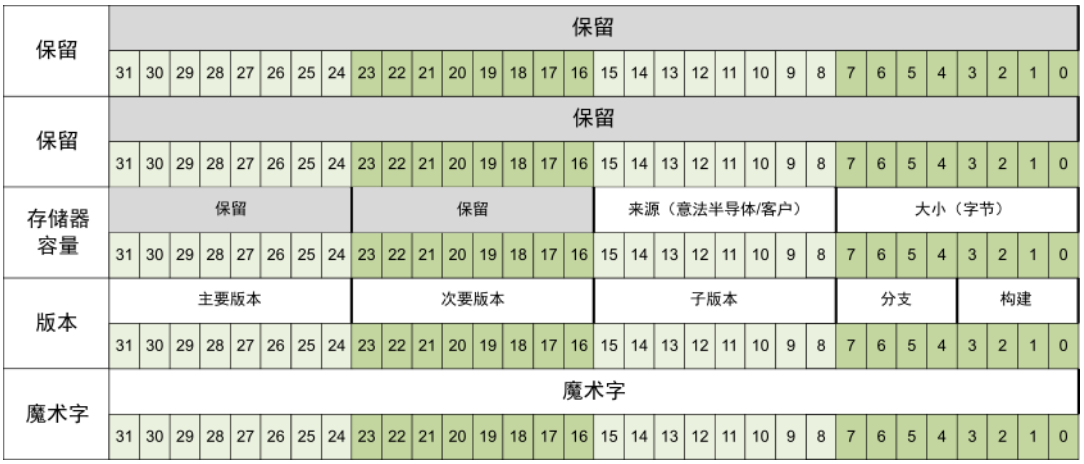


表 16. 签名包尾解析

位域	意义
保留	此版本中未使用
大小	签名总字节数（不含包尾）
源	签名性质： 0x00：ST 签名 0x01：客户签名
构建	版本构建号
分支	版本分支号
子版本	版本控制号
次版本	次版本号
主版本	主版本号
魔术字	用于识别映像性质的特定值。

魔术字值允许识别下表中详细列出的映像性质：

表 17. 魔术字值

值	性质
0x23372991	无线协议栈映像
0x32279221	FUS 映像
0xD3A12C5E	意法半导体签名
0xE2B51D4A	客户签名
0x42769811	其他固件映像

7 FUS 的 STM32 系统自举程序扩展

为支持 FUS 操作，指令集扩展已被添加到 STM32WB 系统自举程序中。这些指令在 USART 和 USB-DFU 接口上实现，并遵循与现有的标准自举程序相同的规则。

为帮助理解本节，需要事先阅读 STM32 微控制器系统存储器启动模式（AN2606）以及用于 STM32 自举程序（AN3155）的 USART 协议和用于 STM32 自举程序（AN3156）的 USB DFU 协议文档。

7.1 USART 扩展

两个指令已添加到自举程序 USART 标准协议中，以支持 FUS 扩展。所有 FUS 均通过这两个特殊指令来传递：一个用于写入（用于从主机到 FUS 的所有 FUS 指令），另一个用于读取（用于从 FUS 到主机的所有 FUS 指令）。

表 18. 自举程序 USART 指令扩展

指令	操作码	使用
特殊读取指令	0x50（补码 0xAF）	从 FUS 获取数据
特殊写入指令	0x51（补码 0xAE）	向 FUS 发送数据

注意： *对于自举程序，FUS 中添加的以下指令不受支持（无论是在 UART 上，或者在 USB DFU 上）*

- FUS_LOCK_USR_KEY
- FUS_UNLOAD_USR_KEY
- FUS_ACTIVATE_ANTIROLLBACK

锁定和卸载用户密钥是两个仅供 Cortex®-M4 用户应用程序使用的指令。

激活防回滚：可以在 Cortex®-M4 用户应用程序代码中实现，或者使用 STM32CubeProgrammer 功能，或使用 STM32 开源的自举程序示例代码。

7.1.1 USART 特殊读取

特殊读取指令用于执行从器件发送请求数据的 FUS 指令。它分为五个独立数据包：

- 特殊读取指令数据包：
  - 主机发送特殊读取指令代码和补码（0x50, 0xAF）并等待 ACK/NACK 字节。如果收到 NACK，则意味着不支持该指令。
- 指令操作码数据包
  - 主机发送包含以下内容的指令数据包：
    - FUS 指令操作码（2 个字节）
    - FUS 指令操作码的 XOR（2 个字节）
  - 如果支持操作码，则器件发送 ACK。否则发送 NACK。
- 地址数据包：
  - 主机用两个字节（MSB 在前）发送地址数据包有效负载大小。
  - 主机发送地址有效负载字节（MSB 在前）。
  - 主机发送数据包 XOR 值（当前数据包中所有先前字节的校验和，1 个字节）。
  - 如果数据正确且受支持，器件将发送 ACK。否则发送 NACK。
- 响应数据包：（可选）
  - 器件用 2 个字节（MSB 在前）发送以字节为单位的包数据有效负载大小。
  - 器件发送数据有效负载字节（MSB 在前）。某些指令不需要数据有效负载。

- 响应状态数据包：
  - 器件用 2 个字节（MSB 在前）发送以字节为单位的数据包有效负载大小。
  - 器件用一个字节发送指令状态（主机所请求的当前指令的状态）。
  - 器件用 1 个字节来发送当前器件状态（有效负载大小 > 3，则可选）。
  - 器件用 1 个字节发送当前指令错误代码（或密钥索引）。
  - 器件通过发送 ACK 来指示响应数据包结束。

图 10. USART 特殊读取指令

7.1.2 USART 特殊写入

特殊写入指令用于执行从器件发送请求数据的 FUS 指令。它分为四个独立数据包：

- 特殊写入指令数据包：主机发送特殊写入指令代码和补码（0x51, 0xAE）并等待 ACK/NACK 字节。如果收到 NACK，则意味着不支持该指令。
- 指令操作码数据包：
  - 主机发送包含以下内容的指令数据包：
    - FUS 指令操作码（2 个字节）
    - FUS 指令操作码的 XOR（2 个字节）
  - 如果支持操作码，则器件发送 ACK。否则发送 NACK。
- 地址数据包：
  - 主机用两个字节（MSB 在前）发送地址数据包有效负载大小。
  - 主机发送地址有效负载字节（MSB 在前）。
  - 主机发送数据包 XOR 值（当前数据包中所有先前字节的校验和，1 个字节）。
  - 如果数据正确且受支持，器件将发送 ACK。否则发送 NACK。
- 数据包：
  - 主机用 2 个字节（MSB 在前）发送以字节为单位的包数据有效负载大小。当指令不需要数据时，该数字可能为零。
  - 主机发送数据有效负载字节（MSB 在前）。如果有效负载大小为零，则不发送数据。
  - 主机发送数据包 XOR 值（当前数据包中所有先前字节的校验和，1 个字节）。
  - 如果数据正确且受支持，器件将发送 ACK。否则发送 NACK。
- 响应数据包：
  - 器件用 2 个字节（MSB 在前）发送以字节为单位的数据包有效负载大小。
  - 器件用一个字节发送指令状态（主机所请求的当前指令的状态）。
  - 器件可能用 1 个字节来发送当前器件状态（有效负载大小 > 1，则可选）。
  - 器件用 1 个字节来发送当前的指令错误代码（有效负载大小 > 1，则可选）。
  - 器件通过发送 ACK 来指示响应数据包结束。

图 11. USART 特殊写入指令

7.1.3 USART FUS 指令映射

只有一个 FUS 指令映射在特殊读取指令上。

表 19. 映射在读取指令上的 USART FUS 指令

指令	操作码	地址数据包	数据包	Cmd 状态数据包
FUS_GET_STATE	0x54	大小 = 0x0000 数据 = 无	大小 = 0x0003 数据 = [0x00、FUS_STATE、错误代码]	大小 = 0x0001 或 0x0003 如果 OK，则数据 = [0x00]，如果 KO，则数据 = [0x01、状态、错误]

有七个 FUS 指令映射在特殊写入指令上。

表 20. 映射在写入指令上的 USART FUS 指令

指令	操作码	地址数据包	数据包	Cmd 状态数据包
FUS_FW_DELETE	0x52	大小 = 0x0000 数据 = 无	大小 = 0x0000 数据 = 无	大小 = 0x0001 或 0x0003 如果 OK，则数据 = [0x00]，如果 KO，则数据 = [0x01、状态、错误]
FUS_FW_UPGRADE	0x53	大小 = 0x0000 数据 = 无	大小 = 0x0000 数据 = 无	
FUS_UPDATE_AUTH_KEY	0x56	大小 = 0x0000 数据 = 无	大小 = 最多 65 个字节 数据 = 密钥（1 个字节的密钥大小 + 64 个字节的密钥数据）	
FUS_LOCK_AUTH_KEY	0x57	大小 = 0x0000 数据 = 无	大小 = 0x0000 数据 = 无	
FUS_STORE_USR_KEY	0x58	大小 = 0x0000 数据 = 无	大小 = 最多 34 个字节 数据 = [密钥类型（1 个字节）、密钥大小（1 个字节）、密钥数据（16/32 个字节）]	大小 = 0x0003 数据 = [0x00、状态、密钥索引]
FUS_LOAD_USR_KEY	0x59	大小 = 0x0000 数据 = 无	大小 = 0x0001 数据 = [密钥索引]	大小 = 0x0001 或 0x0003 如果 OK，则数据 = [0x00]，如果 KO，则数据 = [0x01、状态、错误]
FUS_START_WS	0x5A	大小 = 0x0000 数据 = 无	大小 = 0x0000 数据 = 无	

7.2 USB-DFU 扩展

通过自举程序 USB-DFU 标准下载和上传指令来处理 FUS 指令。

7.2.1 USB-DFU 下载 FUS 扩展

自举程序 USB-DFU 下载 FUS 扩展的管理方式与 SET_ADDRESS_POINTER 和 ERASE 标准指令相同：值 = 0，后续字节是 MSB 优先的指令数据。

FUS_STORE_USR_KEY 是一个例外，它分为两步：

1. 下载指令，只允许发送密钥数据（最大 34 个字节）
2. 上传指令，必须在下载步骤之后完成，并允许获取密钥索引（1 个字节）

表 21. USB-DFU 下载扩展

指令	操作码	数据
FUS_FW_DELETE	0x52	无
FUS_FW_UPGRADE	0x53	无
FUS_UPDATE_AUTH_KEY	0x56	密钥缓冲区 = [密钥大小（1 个字节），密钥数据（64 个字节，MSB 在前）]
FUS_LOCK_AUTH_KEY	0x57	无
FUS_STORE_USR_KEY	0x58	密钥缓冲区 = [密钥类型（1 个字节），密钥大小（1 个字节），密钥数据（16/32 个字节）]
FUS_LOAD_USR_KEY	0x59	密钥索引（1 个字节）
FUS_START_WS	0x5A	无

7.2.2 USB-DFU 上传 FUS 扩展

自举程序 USB-DFU 上传 FUS 扩展与用于读取物理地址（wBlockNum > 1）的常规上传指令采用相同的管理方式。但在这种情况下使用了虚拟存储器地址掩码：0xFFFF0000.因此，通过读取虚拟地址 0xFFFF00YY 来管理 FUS 读取指令，其中 YY 为 FUS 指令操作码。

上传指令可用于执行 FUS_STORE_USR_KEY 的第二步，即获取密钥索引。

表 22. USB-DFU 上传扩展

指令	地址	返回的数据
FUS_GET_STATE	0xFFFF0054	状态缓冲区 = [FUS 状态（1 个字节），FUS 错误代码（1 个字节）]
FUS_STORE_USR_KEY	0xFFFF0058	密钥索引（1 个字节）

8 常见问题与故障排除

表 23. 常见问答

问题/故障排除	回答
接收的意法半导体原装 STM32WB 器件具体包含哪些内容？	意法半导体提供的所有 STM32WB 器件默认包含 FUS 和自举程序。 但是不包含预安装的无线协议栈。
我无法读取 FUS 版本	在满足以下条件时，可以访问器件信息表： 1. 器件信息表地址写在 IPCCDBA 选项字节指定的位置。 2. Cortex®-M0+已启用 3. FUS 运行在 Cortex®-M0+（而不是无线协议栈）上（如果无线协议栈已运行，可以发送 2 个 FUS_GET_STATE 指令以强制 FUS 运行）。所以当通过 SWD 访问器件时，由于器件信息表还未写入或 Cortex®-M0+还未启用，所以此时器件信息表无效是正常现象。在自举程序运行之后会执行上述操作（1）和（2），故而此时读取器件信息表更方便。 *注意：可以通过 SWD 进行连接并禁用硬件复位选项（热插拔），并保持通过自举程序启动，从而允许用户读取器件信息表。*
我想升级 FUS 映像，并且我已经安装了无线协议栈。升级 FUS 前是否需要删除无线协议栈？	一般情况下（特别是从 FUS V0.5.3 升级时），建议在 FUS 升级之前删除无线协议栈。 如果现有的 FUS 版本高于 V0.5.3，则无需删除无线协议栈。
如何快速了解我的器件运行的是 FUS 还是无线协议栈？	有多种方法可以查看： • 读取选项字节并检查 SBRV 的值。如果 FUS 正在运行，则为 0x3D000（如果 FUS V0.5.3 正在运行，则为 0x3D800） • 读取器件信息表@0x20030030，如果与 FUS 版本不一致，则表示运行的是无线协议栈，或 Cortex®-CM0+未启用。 • 发送 FUS_GET_STATE 指令，如果收到 FUS_STATE_NOT_RUNNING，则表示运行的是无线协议栈，或 Cortex®-CM0+未启用。
IPCCDBA 选项字节的作用是什么？	PCCDBA 用于修改器件信息表的读写偏移量。
升级完成后，我无法访问 Flash 存储器，也不能与 FUS 通信。	首先检查 SFSA 是否等于 0x00。如果是，则意味着安全模式已经触发。 安全模式会在选项字节损坏时触发。 这可能发生在 FUS 升级操作期间或任何处理选项字节的用户应用程序操作期间。 安全模式触发后，会设置 SFSA=0x00（所有 Flash 存储器安全）以锁定器件，防止用户应用程序/调试器访问用户 Flash 存储器。 该操作不可逆。 从 FUS V1.1.0 开始，安全模式被修改为恢复出厂设置，而不是锁定器件。
是否可以降级 FUS 版本（例如，当前运行的 FUS 版本为 V1.0.2，是否可以安装 FUS V1.0.1？）	任何情况下都不能将 FUS 降级。只能安装高级别版本。 如果尝试降级，FUS 将拒绝升级并返回错误消息。

问题/故障排除	回答
使用 FUS 执行升级的常见 STM32CubeProgrammer 指令是什么？	1. 首先通过发送 FUS_GET_STATE 指令来查看 FUS 是否运行，直至收到 FUS_STATE_IDLE 状态响应： – STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate – STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate – STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate 发送 3 次 FUS_GET_STATE 指令将确保 FUS 运行并在大多数情况下处于空闲状态。 2. 删除现有无线协议栈，安装新的无线协议栈（升级无线协议栈时）： – STM32_Programmer_CLI.exe -c port=usb1 -fwupgrade stm32wb5x_BLE_Stack_fw.bin 0x080CB000 firstinstall=0 – STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate – STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate – …（保持发送 -fusgetstate，直至收到的状态为 FUS_STATE_NOT_RUNNING） 设置“firstinstall=0”确保将在安装新协议栈前删除以前的协议栈。 即使之前没有安装协议栈，设置“firstinstall=0”也不会引起任何问题。 交替进行 FUS 映像安装（升级 FUS 时）： • STM32_Programmer_CLI.exe -c port=usb1 -fwupgrade stm32wb5x_FUS_fw.bin 0x080EC000 firstinstall=0 • STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate • STM32_Programmer_CLI.exe -c port=usb1 -fusgetstate • …（保持发送 -fusgetstate，直至收到的状态为 FUS_STATE_IDLE） “firstinstall=0”意味着将在升级 FUS 之前删除现有无线协议栈。 如果从 FUS V0.5.3 以外的 FUS 版本升级，可以使用“firstinstall=1”。
什么是安全模式，如何使用？	安全模式是 FUS 的一个独立部分，专门适用于一种情况：选项字节损坏。 如果选项字节损坏，则 STM32WB 硬件会无视运行的固件，强制以安全模式启动。 之后，安全模式会： • 将器件锁定在完全安全模式（当 FUS 版本低于 V1.1.0 时），这意味着所有的器件 Flash 存储器都无法访问，且该操作不可逆（没有办法取消，器件无法再次使用）。 • 或者恢复出厂设置（如果 FUS 版本为 V1.1.0 或更高版本），这意味着删除无线协议栈（如已安装），擦除 Cortex®-M4 代码，并将启动重置为 FUS（最初状态）。该操作同样不可逆。为了激活安全模式，用户必须使用 SWD 接口将值 0x00008000 写入到地址 0x5800040C，从而激活 Cortex®-M0+。
FUS 是否会在安装后擦除加密固件的阴影？	是的，FUS 确实会在安装并移动到上层地址后清除加密固件的残留阴影扇区。
可以安装的固件映像大小是否存在限制？ 安装新映像前是否需要删除已安装的固件映像？	使用 V1.2.0 以前的 FUS 版本时： • 如果在另一无线固件映像 A 已安装/正在运行时安装无线固件映像 B；如果 B 的大小大于 A 的大小，并且 B 的加载地址过于接近 A 的地址（A 的起始地址和 B 的结束地址之间没有足够的空闲空间，如第 2 节“无线协议栈映像操作”所述），那么器件可能会被指向固件映像 A（然后受到损坏）而不是指向固件 B 的 SBRV 值阻止，这种情况下，则无法恢复。 • 因此，建议在安装固件映像 B 之前删除固件映像 A（对于 FOTA，这可能不可行），或者在执行安装前确保具有足够的可用空间。此已知限制已在 FUS V1.2.0 中得以解决。

问题/故障排除	回答
升级 FUS V1.2.0 后，STM32CubeProgrammer（或其他编程接口）上收到一条错误消息，错误代码：“FUS_UFB_CORRUPT. 这是什么意思？这种情况下应采取什么措施？	当从一些旧版本的 FUS 升级到 FUS V1.2.0 时，出现 FUS_UFB_Corrupt 错误消息属于正常现象。 这意味着 UFB 区域（用于存储 FUS 配置信息）已被擦除，需要配置为复位值。 FUS 随后会写入复位值并触发系统复位，然后清除错误。FUS 返回 FUS_IDLE 状态且无错误。 用户端无需执行任何操作。

版本历史

表 24. 文档版本历史

日期	版本	变更
2019 年 3 月 21 日	1	初始版本。
2019 年 6 月 17 日	2	增加了第 1.2 节“FUS 版本管理和识别” 更新了： • 第 2 节“无线协议栈映像操作”、第 5.1 节“密钥类型和结构”、第 5.1 节“密钥类型和结构” • 表 20. 映射在写入指令上的 USART FUS 指令，表 22。USB-DFU 上传扩展
2019 年 7 月 10 日	3	更新了表 7. 器件信息表
2020 年 3 月 31 日	4	更新了： • 第 1.1 节“固件升级服务定义”、第 2 节“无线协议栈映像操作”、第 2.1 节“无线协议栈安装和升级”、第 2.2 节“无线协议栈删除”、第 5.1 节“密钥类型和结构”、第 7.1 节“USART 扩展” • 表 1. FUS 版本，表 11. FUS 指令（厂商特定 HCI 指令包），表 12. FUS 响应（HCI 指令完成数据包），表 14. FUS 状态错误值 增加了：第 2.4 节“防回滚功能激活”和第 8 节“常见问题与故障排除”
2021 年 5 月 6 日	5	更新了： • 第 1.2 节“FUS 版本管理和识别” • 表 1.FUS 版 • 第 1.3 节“如何激活 FUS” • 图 1.Flash 存储器映射 • 第 1.5 节“FUS 资源使用情况” • 表 5.FUS 资源使用情况 • 第 2.4 节“防回滚功能激活” • 第 8 节“常见问题和故障排除” 增加了： • 表 1. FUS 版 • 表 2. FUS 版本兼容性
2021 年 10 月 22 日	6	更新了： • 第 1.5 节“FUS 资源使用情况” • 图 8.FW/FUS 升级映像包尾结构 • 图 9.签名（标签）包尾结构 • 第 8 节“常见问题和故障排除以及新限制”
2022 年 8 月 2 日	7	增加了第 1.2.1 节“已知限制” 更新了： • 第 2.1 节“无线协议栈安装和升级” • 第 2.2 节“无线协议栈删除” • 第 8 节“常见问题和故障排除”
2023 年 1 月 16 日	8	更新了表 5. FUS 资源使用情况。 更新了第 2.2 节“无线协议栈删除”。 对整个文档进行少量文字修订。

目录

表格索引

图片目录