AN3156
应用笔记

STM32 自举程序中使用的 USB DFU 协议

前言

本应用笔记说明了 STM32 微控制器自举程序中使用的 USB DFU 协议。它详细说明了每个支持的指令。若需器件自举程序 USB 硬件资源和要求的更多信息，请参考 “STM32 系统存储器自举模式 ” 应用笔记 （AN2606）。

表 1. 适用产品

类型	料号或产品系列
微控制器	STM32L1 系列： – STM32L1xxxC, STM32L1xxxD, STM32L1xxxE STM32F0 系列 STM32F1 系列： – STM32F105xx, STM32F107xx STM32F2 系列 STM32F3 系列： – STM32F373xx, STM32F302xx, STM32F303xB(C), STM32F301xx STM32F4 系列： – STM32F401xx、 STM32F411xx – STM32F405xx、 STM32F407xx – STM32F415xx、 STM32F417xx – STM32F427xx, STM32F429xx – STM32F437xx, STM32F439xx

目录

表格索引

图片索引

1 自举程序代码序列

不同版本的自举程序 DFU 在协议 （请求和指令）方面没有区别。要查看具体的区别列表，请参见第 6 节。

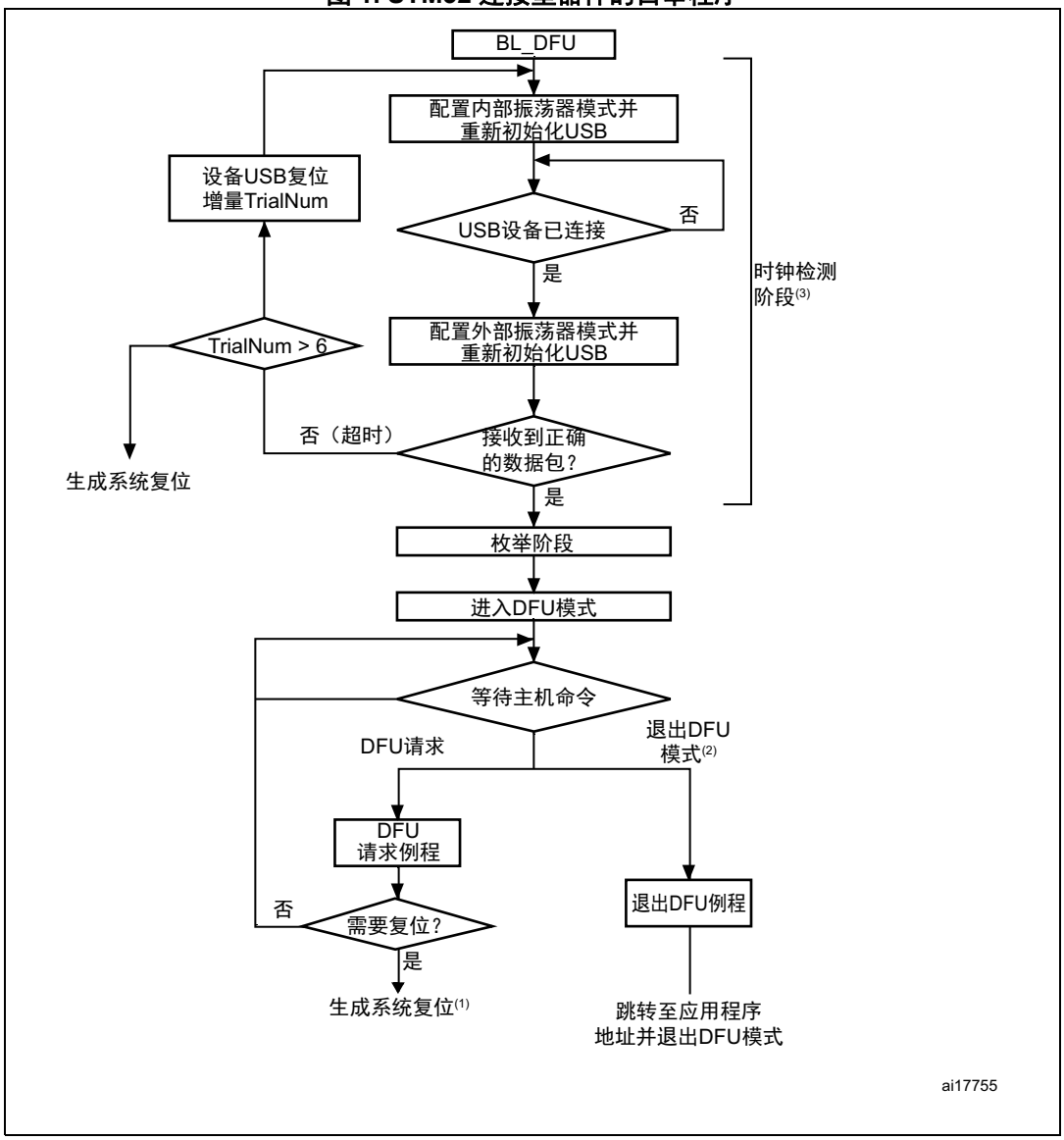
进入系统存储器自举模式并且 STM32 器件已配置完毕后 （更多详细信息，请参见 AN2606），自举程序代码会配置 USB 及其中断，并会等待 “ 枚举完成 ” 中断。

插入 USB 线缆之后，会立即执行 USB 枚举 （如果 USB 线缆已插入，则会立即执行 USB 枚举）。如果不希望 STM32 进入 USB DFU 自举程序，必须在复位前先拔出 USB 线缆。

自举程序版本会返回到 bcd 器件字段 MSB 中的器件描述符 （示例：0x2000 = 版本 2.0）。

对于互连型 USB DFU 自举程序，器件会先尝试使用 25 MHz 配置，如果失败，随后会尝试使用 14.7456 MHz，如果失败，最后会尝试使用 8 MHz 配置。如果仍失败，会使用较大的超时值重复执行上述操作 （会再次测试三种配置）。如果第二次尝试也失败的话，会生成系统复位。

图 1. STM32 连接型器件的自举程序

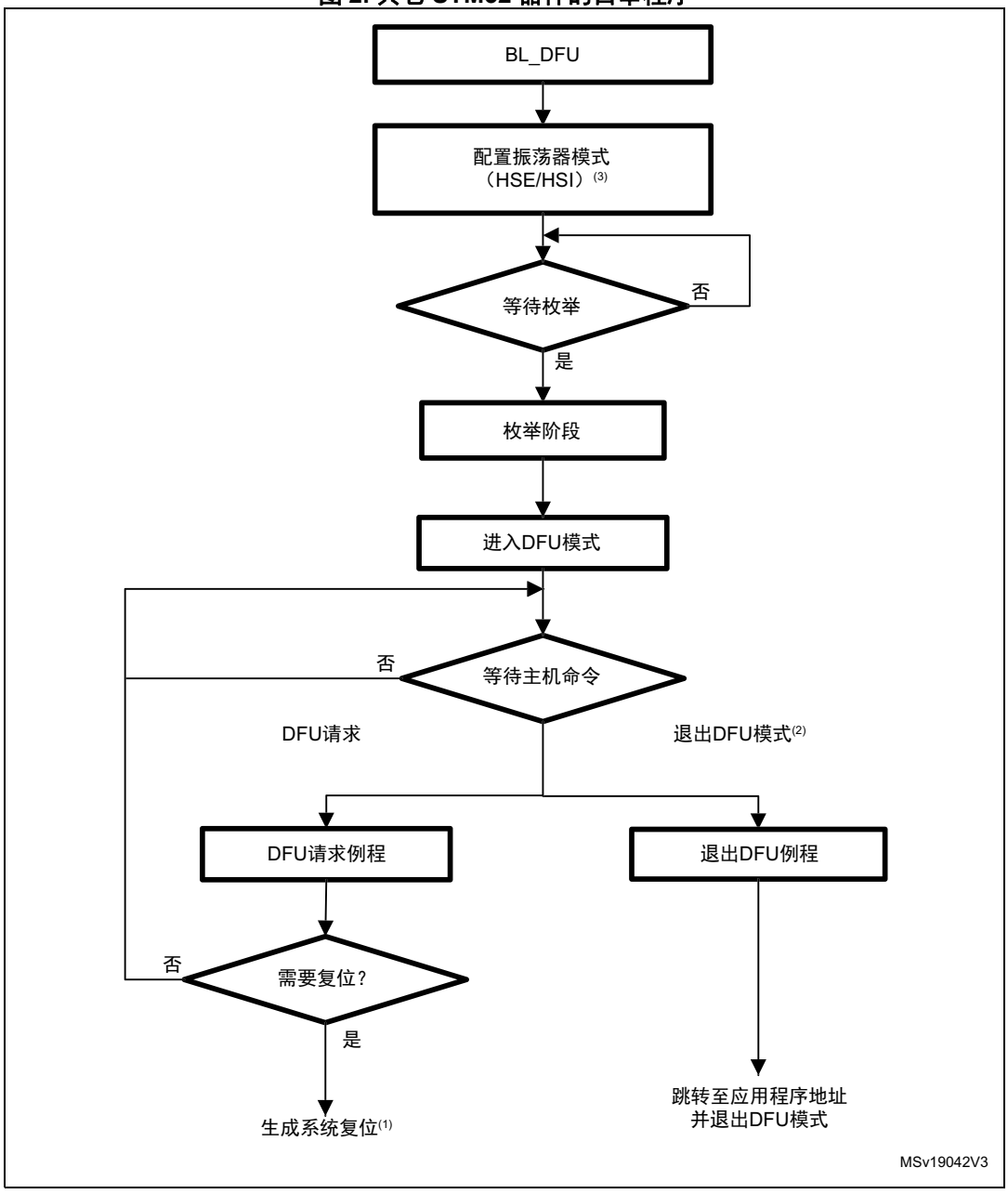


1. 系统复位后，器件可能会返回到 BL_DFU 循环，也可能执行 Flash 存储器中的代码，具体视连接状态和自举引脚的状态而定。
2. 先发出 0 Data Download 请求，然后再发出 GetStatus 请求和 Device Reset 请求，即可退出 DFU 模式。
3. 六次尝试后 （三种时钟配置均测试两次），会生成系统复位。

- 如果产品使用 HSE 进行 USB 操作 （连接型产品除外）：
  - 启动时，会测量 HSE （若存在），如果支持，则会配置 USB。如果未检测到 HSE，自举程序将执行系统复位。如果测得的 HSE 时钟值属于不受支持的值，USB 协议将无法正常执行。
- 如果产品使用 HSI 进行 USB 操作。
  - 启动时，会使用 HSI 时钟配置 USB。

关于产品配置的更多详细信息，请参见 AN2606。

图 2. 其它 STM32 器件的自举程序



1. 系统复位后，器件可能会返回到 BL_DFU 循环，也可能执行 Flash 存储器中的代码，具体视连接状态和自举引脚的状态而定。
2. 先发出 0 Data Download 请求，然后再发出 GetStatus 请求和 Device Reset 请求，即可退出 DFU 模式。
3. 对于某些产品，不会为 USB 自举程序操作使用外部振荡器 HSE，而只会使用内部振荡器 HSI。请查阅 AN2606 产品的相关章节，了解需要为每种产品使用哪种振荡器。

注： *自举程序启动时，内部振荡器 (HSI) 会用作 USB 接口的时钟源。检测到 USB 事件后，外部振荡器会配置为 USB 时钟源。*

2 USB DFU 自举程序请求

USB DFU 自举程序支持 2004 年 8 月 5 日发布的 1.1 版 “ 器件固件升级通用串行总线器件升级规范 ” 规定的 DFU 协议和请求。关于这类请求的更多详细信息，请参见规范。

表 2 和*表 3* 列举了 DFU 类特有的请求及其参数。

表 2. DFU 类请求

请求	请求代码	请求说明
DFU_DETACH	0x00	请求器件退出 DFU 模式并进入应用程序。
DFU_DNLOAD	0x01	请求将数据从主机传输到器件，以便将数据加载到器件的内部 Flash 存储器中。还包括擦写指令。
DFU_UPLOAD	0x02	请求将数据从器件传输到主机，以便将器件内部 Flash 存储器的内容加载到主机文件中。
DFU_GETSTATUS	0x03	请求器件向主机发送状态报告 （包括执行上一请求后得出的状态以及执行该请求后器件将立即进入的状态）。
DFU_CLRSTATUS	0x04	请求器件清除错误状态并转至下一步。
DFU_GETSTATE	0x05	请求器件仅发送将在该请求后立即进入到的状态。
DFU_ABORT	0x06	请求器件退出当前状态 / 操作并立即进入空闲状态。

注： *对于自举程序，DETACH 请求没有任何意义。自举程序是通过系统复位启动的，具体视自举模式配置设置而定，也就是说，此时不会运行其它应用程序。*

表 3. DFU 类特有的请求汇总

bmRequest	bRequest	wValue	wIndex	wLength	Data
00100001b	DFU_DETACH	wTimeout	Interface	Zero	None
00100001b	DFU_DNLOAD	wBlockNum	Interface	Length	Firmware
10100001b	DFU_UPLOAD	Zero	Interface	Length	Firmware
00100001b	DFU_GETSTATUS	Zero	Interface	6	Status
00100001b	DFU_CLRSTATUS	Zero	Interface	Zero	None
00100001b	DFU_GETSTATE	Zero	Interface	1	Status
00100001b	DFU_ABORT	Zero	Interface	Zero	None

通信安全

主机与器件之间的通信安全是通过嵌入的 USB 保护机制 （CRC 校验、确认等）来保障的。不会对已传输的数据或自举程序特有的指令 / 数据进行进一步保护。

3 DFU 自举程序指令

DFU_DNLOAD 和 DFU_UPLOAD 请求主要用于执行简单的存储器读写操作。这两个请求也用于发出集成自举程序指令 （write、 read unprotect、 erase、 set address 等）。DFU_GETSTATUS 指令会触发这些指令被真正执行。

在 DFU 下载请求中，指令是通过 USB 请求结构中的 **wValue** 参数选择的。如果 **wValue** = 0，主机在发送请求之后发出的数据就是自举程序指令代码。第一个字节是指令代码，其它字节 （如果存在）是与该指令相关的数据。

在 DFU 上传请求中，指令是通过 USB 请求结构中的 **wValue** 参数选择的。如果 **wValue** = 0，则会选择 Get 指令并执行。

表 4. DFU 自举程序指令

DFU 请求	自举程序指令	写保护禁用 读保护禁用	写保护启用 读保护禁用	读保护启用
DFU_UPLOAD	Read Memory	允许	允许	不允许
	Get	允许	允许	允许
DFU_DNLOAD	Write Memory	允许	允许(1)	不允许
	Erase	允许	允许 (1)	不允许
	Read Unprotect	NA(2)	NA(2)	允许(3)
	Set Address Pointer	允许	允许	允许
	Leave DFU mode	允许	允许	允许

1. 允许执行此操作，但操作无效：自举程序不会返回错误，但操作不会执行，因为扇区受到写保护。这一点仅适用于 Flash 存储器，不适用于 RAM 存储器或选项字节区域。
2. 运行执行此操作，但此操作没有任何意义，因为存储器未受保护。
3. 在这种情况下，会同时擦除 Flash 存储器 （从 0x0800 0000 开始）和 RAM。选项字节区域会复位为默认值。

如果执行 Read Unprotect 操作，同时存储器未受保护，那么整个 RAM 存储器会被自举程序固件清空，而 Flash 存储器不会被擦除 （由于 Flash 存储器之前未受读保护）。

没有针对 Write Protect、 Write Unprotect 和 Read Protect 操作的指令。这些操作应通过用于选项字节区域的 Write Memory 和 Read Memory 指令来执行。

4 DFU_UPLOAD 请求指令

上传请求允许执行不同指令。指令是通过 USB 请求结构中 **wValue** 的参数值来选择的。在*第 4.1 节*到*第 5.5 节*中所描述的操作是被支持的。

4.1 Read Memory

当 **wValue** > 1 时，会选择 Read memory 操作。

主机会请求器件从内部 Flash 存储器、嵌入式 RAM、系统存储器的有效存储器地址 （见说明）、或者从选项字节发送指定数目的数据字节 (**wLength**)。

注： *更多关于您所使用的器件的有效存储器地址信息，请参见第 4 节：DFU_UPLOAD 请求指令。*

允许读取的字节数取决于存储器目标：

- 对于内部 Flash 存储器、嵌入式 RAM 和系统存储器：读取字节的大小为 2 到 2048 字节
- 对于选项字节：读取字节的大小应等于选项字节块的大小
- 对于其它存储器位置，请参见 AN2606 中相应产品的 “ 重要考量因素 ” 章节。

主机请求从哪一地址开始读取数据是使用 wBlockNumber (**wValue**) 的值以及地址指针在下列公式中计算得出的：

地址 = ((wBlockNum – 2) × wTransferSize) + Address_Pointer，其中：

– wTransferSize 是请求的数据缓冲区的长度。

地址指针应事先通过 Set Address Pointer 指令 （使用 DFU_DNLOAD 请求）指定。否则（如果未事先指定地址），器件会假定起始地址为内部 Flash 起始地址 (0x08000000)。

如果启用 Flash Read Protection，无论读取目标是内部 Flash 存储器、嵌入式 RAM、系统存储器还是选项字节，都不会执行 Read 操作，返回的器件状态为 Status = dfuERROR、State = errVENDOR。

4.2 Get 指令

wValue = 0 时，会选择该指令。

主机会请求读取自举程序支持的指令。收到该指令后，器件会返回 N 个代表指令代码的字节。

STM32 会发送以下字节 (N = 4)：

字节 1： 0x00 - Get 指令

字节 2： 0x21 - Set Address Pointer

字节 3： 0x41 - Erase

字节 4： 0x92 - Read Unprotect

DFU_UPLOAD 指令的处理过程如*图 3* 和*图 4* 所示。

图 3. DFU_UPLOAD 请求：器件端

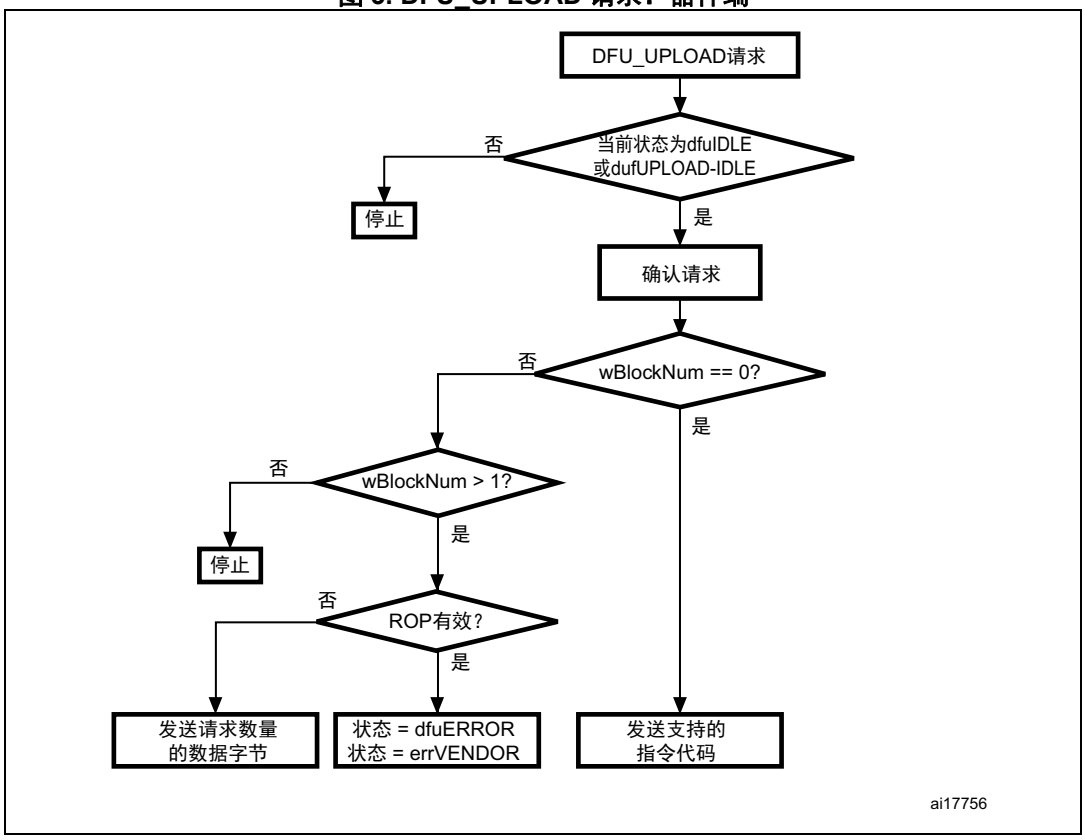
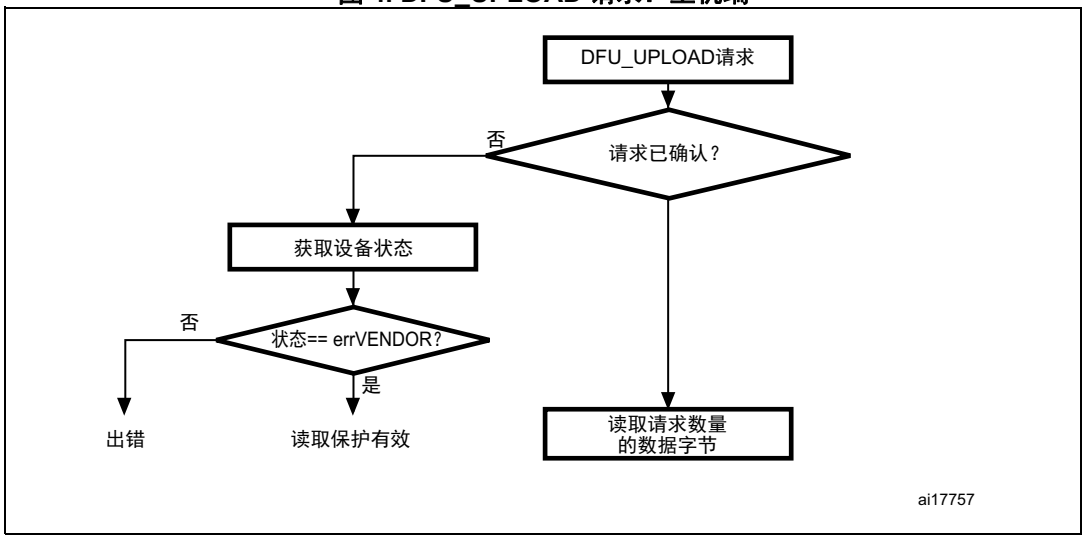


图 4. DFU_UPLOAD 请求：主机端



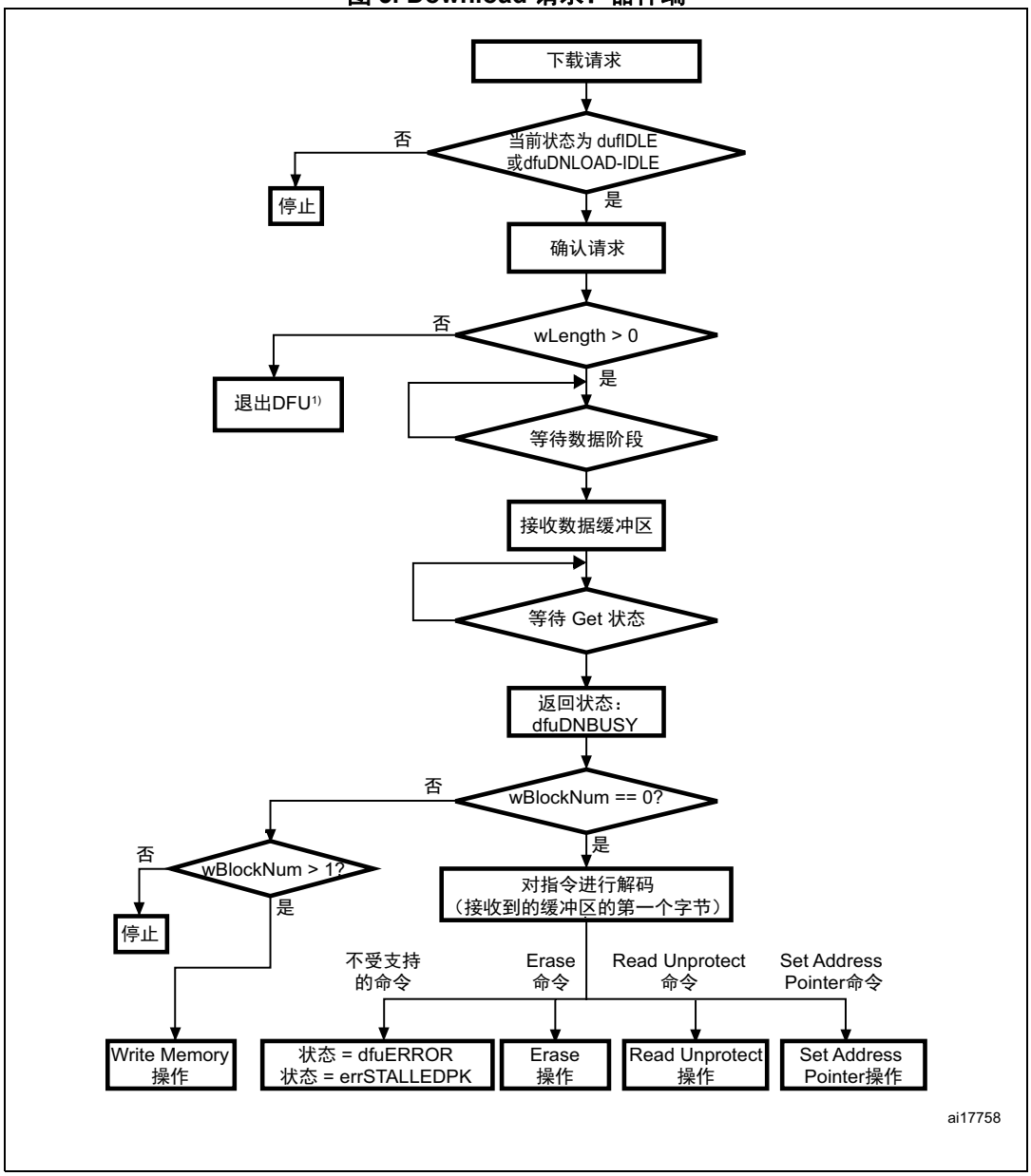
注： *发出 Upload 请求之前，主机必须检查器件是否处于正确状态（dfuIDLE 或 dfuUPLOAD-IDLE 状态），并且要检测状态中是否报错。如果器件并未处于要求的状态，主机需要清除错误（DFU_CLRSTATUS 请求）并获取新状态，直至器件恢复到 dfuIDLE 状态。*

5 DFU_DNLOAD 请求指令

下载请求用于执行不同的指令。指令是通过 USB 请求结构中 **wValue** 的参数值来选择的。支持的操作如下：

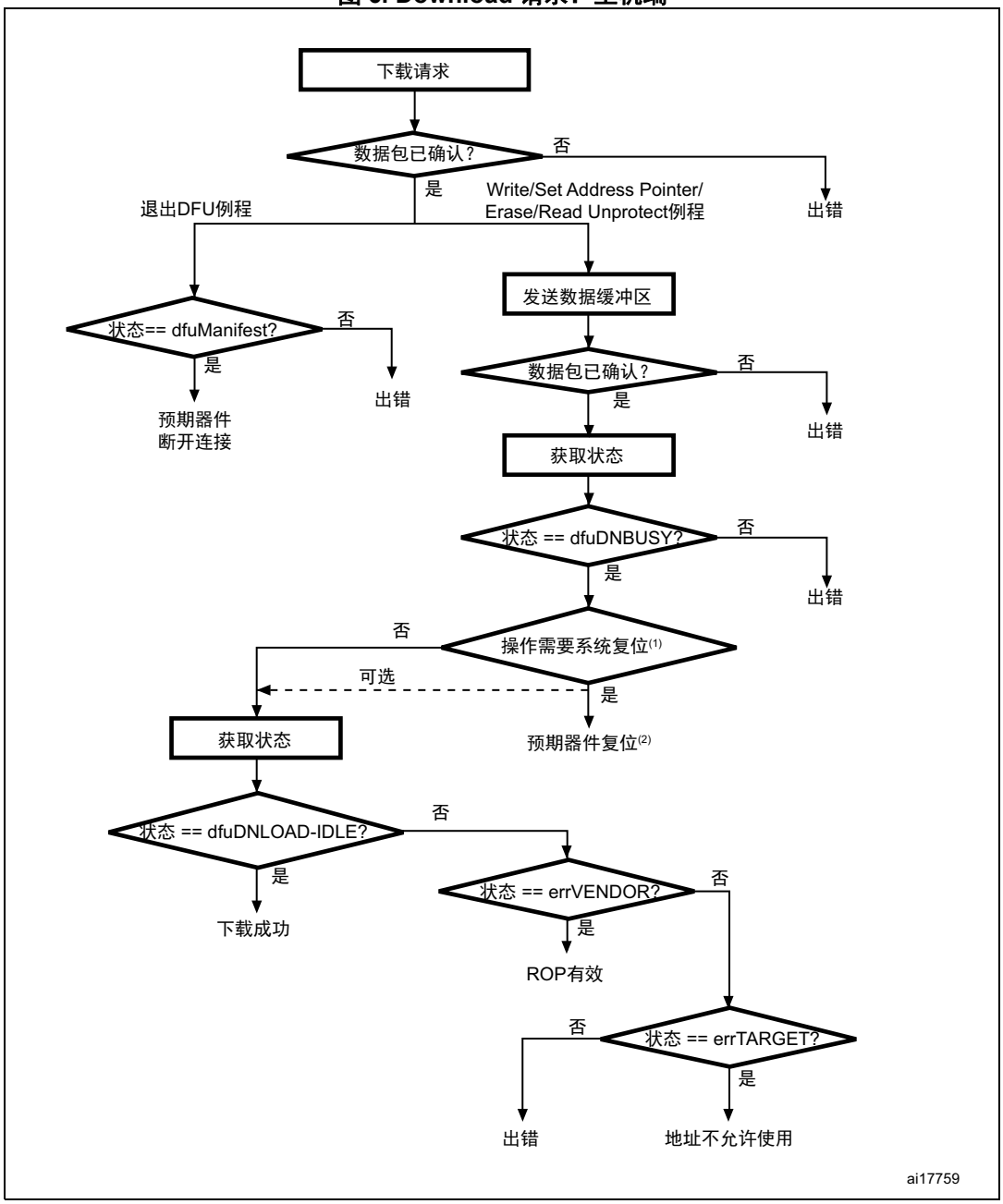
- Write Memory (**wValue** > 1)
- Set Address Pointer （**wValue** = 0，并且第一个字节 = 0x21）
- Erase （**wValue** = 0，并且第一个字节 = 0x41）
- Read Unprotect （**wValue** = 0，并且第一个字节 = 0x92）
- Leave DFU （退出 DFU 模式并跳转至应用程序）

图 5. Download 请求：器件端



1. 该命令可用于复位器件或跳转到应用程序。

图 6. Download 请求：主机端



1. 需要系统复位的操作包括：对选项字节执行的 Read Unprotect 指令和 Write 操作。
2. 恢复为 dfuDNBUSY 状态后，器件会执行请求的操作并执行系统复位。主机可能会等待下一次枚举，也可能再次执行 Get status，但器件将无法响应，除非执行请求的操作失败。

注： *发出 Download 请求之前，主机必须检查器件是否处于正确状态 （dfuIDLE 或 dfuDNLOD-IDLE 状态），并且要检测状态中是否报错。如果器件并未处于要求的状态，主机需要清除其错误 （DFU_CLRSTATUS 请求）并再次获取状态，直至器件恢复到 dfuIDLE 状态。*

5.1 Write memory

当 **wValue** > 1 时，会选择 Write memory 操作。

主机会请求器件接收指定数目 (**wLength**) 的数据字节，并将这些字节加载到内部 Flash 存储器、嵌入式 RAM 中的有效存储器地址 （见说明）或选项字节中。

注： *更多关于您所使用的器件的有效存储器地址信息，请参见第 4 节：DFU_UPLOAD 请求指令。*

允许写入的字节数取决于存储器目标：

- 对于内部 Flash 存储器和嵌入式 RAM：写入字节的大小为 2 到 2048 字节
- 对于选项字节：写入字节的大小应等于选项字节块的大小
- 对于其它存储器位置，请参见 AN2606 中相应产品的 “ 重要考量因素 ” 章节。

注： *对于选项字节，可以写入不同于块大小的字节，但建议一次性写入整个块，以确保数据完整性。如果目标为选项字节区域，地址指针必须始终是选项字节的起始地址，否则将不会执行请求。*

仅当 DFU_GETSTATUS 请求是由主机发出的情况下， Write memory 操作才能有效执行。如果器件返回的状态不是 dfuDNBUSY，说明发生了错误。

需要再发一次 DFU_GETSTATUS 请求，检查指令是否正确执行，但目标位置是选项字节区域的情况除外 （在这种情况下，器件会在写入操作完成后立即复位）。如果接收到的地址不正确或不受支持，器件状态会变为 Status = dfuERROR、 State = errTARGET。

主机请求从哪一地址开始写入数据是使用 wBlockNumber (**wValue**) 的值以及地址指针在与上传请求相同的公式中计算得出的：

地址 = ((wBlockNum – 2) × wTransferSize) + Addres_Pointer，其中：

– wTransferSize：主机发送的数据缓冲区的长度

– wBlockNumber：**wValue** 参数的值

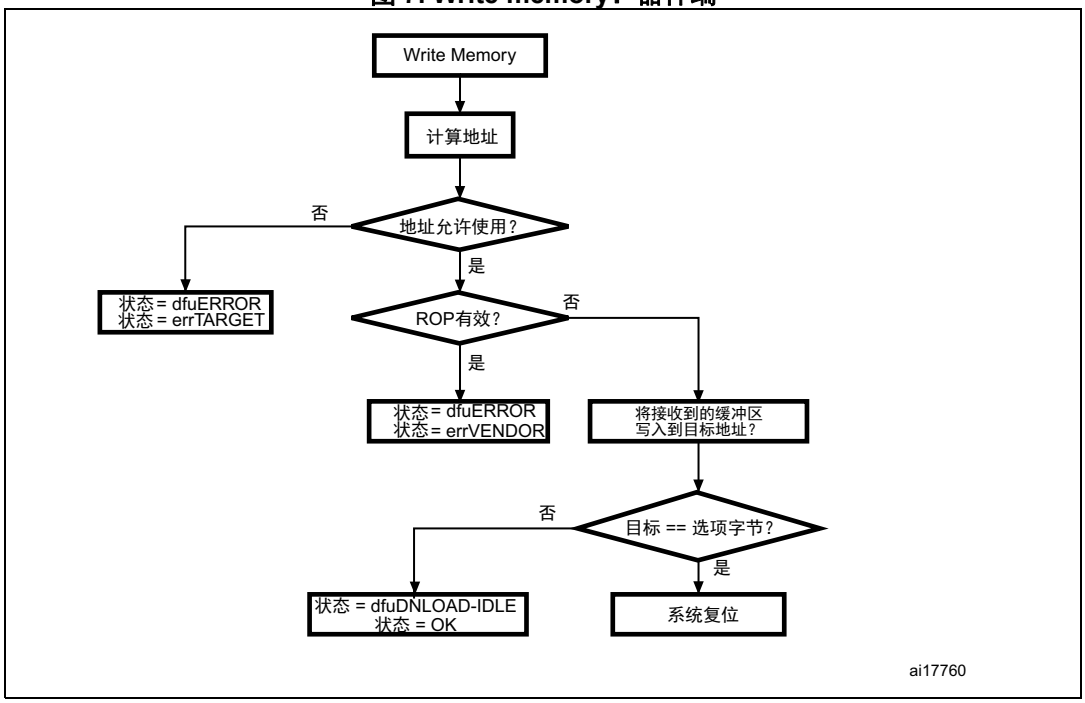
如果启用 Flash Read Protection，无论读取目标是内部 Flash 存储器、嵌入式 RAM、系统存储器还是选项字节，都不会执行 Write memory 操作，返回的器件状态为 Status = dfuERROR、 State = errVENDOR。

若 Write Memory 指令用于选项字节区域，则在写入新值之前会擦除所有选项。在指令末尾，自举程序会生成系统复位，以使选项字节的新配置生效。

注： 1 *当写入 RAM 时，您应注意不要与自举程序固件使用的第一个 RAM 存储器重叠。*

2 *当向写保护的扇区执行写操作时，不会返回错误。*

图 7. Write memory：器件端



5.2 Set Address Pointer 指令

如果 **wValue** = 0，并且主机发送的缓冲区的第一个字节是 0x21，则会选择 Set Address Pointer 指令。缓冲区长度应为 5 （其余四个字节是地址字节， LSB 优先 （32 位地址格式））。

主机会发送包含上述参数的 DFU_DNLOAD 请求，以设置计算 Read memory 和 Write memory 操作的起始地址所使用的地址指针值。

STM32 器件接收的字节如下：

字节 1： 0x21 - Set Address Pointer 指令

字节 2： A[7:0] - 地址指针的 LSB

字节 3： A[15:8] - 地址指针的第二个字节

字节 4： A[22:16] - 地址指针的第三个字节

字节 4： A[31:23] - 地址指针的 MSB

发送 Set Address Pointer 指令后，主机需要发送 DFU_GETSTATUS 请求。

仅当 DFU_GETSTATUS 请求是由主机发出的情况下， Set AddressPointer 指令才能有效执行。如果器件返回的状态不是 dfuDNBUSY，说明发生了错误。

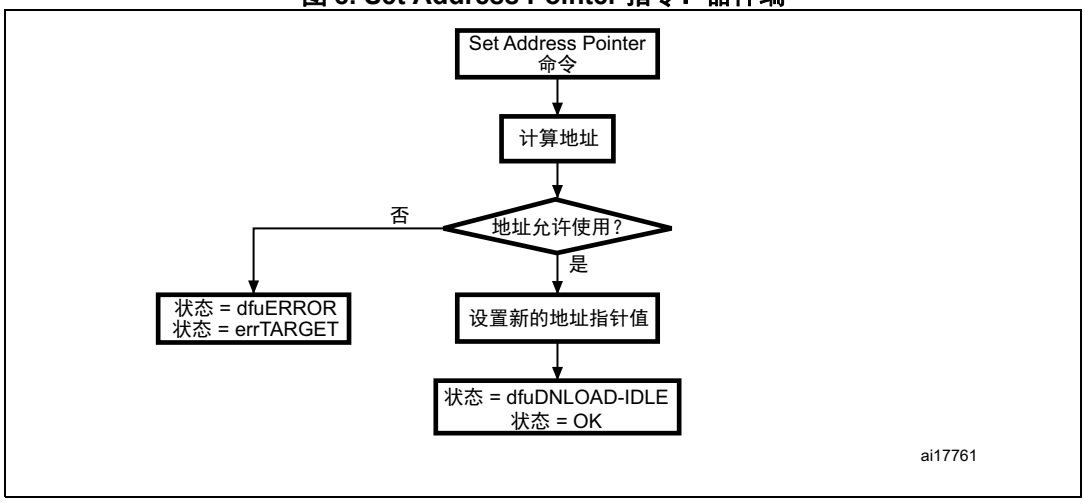
需要再发一次 DFU_GETSTATUS 请求，检查指令是否正确执行。如果接收到的地址不正确或不受支持，器件状态会变为 Status = dfuERROR、 State = errTARGET。

允许存储地址指针值的位置包括 Flash 存储器、嵌入式 RAM、系统存储器中的有效存储器地址 （见说明）以及选项字节。

注： 1 更多关于您所使用的器件的有效存储器地址信息，请参见第 4 节：DFU_UPLOAD 请求指令。

2 Flash Read Protection 启用或禁用时，允许执行 Set Address Pointer 指令。

图 8. Set Address Pointer 指令：器件端



5.3 Erase 指令

如果 **wValue** = 0，并且主机发送的缓冲区的第一个字节是 0x41，则会选择 Erase 指令。对于页擦除操作，缓冲区长度是 5 个字节 （其余四个字节是地址字节， LSB 优先），对于批量擦除操作，缓冲区也可以只有 1 个字节 （仅包含指令字节）。

主机会发送包含上述参数的 DFU_DNLOAD 指令，以擦除一页内部 Flash 存储器，或对该 Flash 存储器执行批量擦除。

device 接收到的字节如下 （页擦除）：

字节 1： 0x41 - Erase 指令

字节 2： A[7:0] - 页地址的 LSB

字节 3： A[15:8] - 页地址的第二个字节

字节 4： A[22:16] - 页地址的第三个字节

字节 4： A[31:23] - 页地址的 MSB

或者，如果接收到 1 个字节的指令：

STM32 接收到的字节如下 （批量擦除）：

字节 1： 0x41 - Erase 指令

发送 Erase 指令后，主机需要发送 DFU_GETSTATUS 请求。

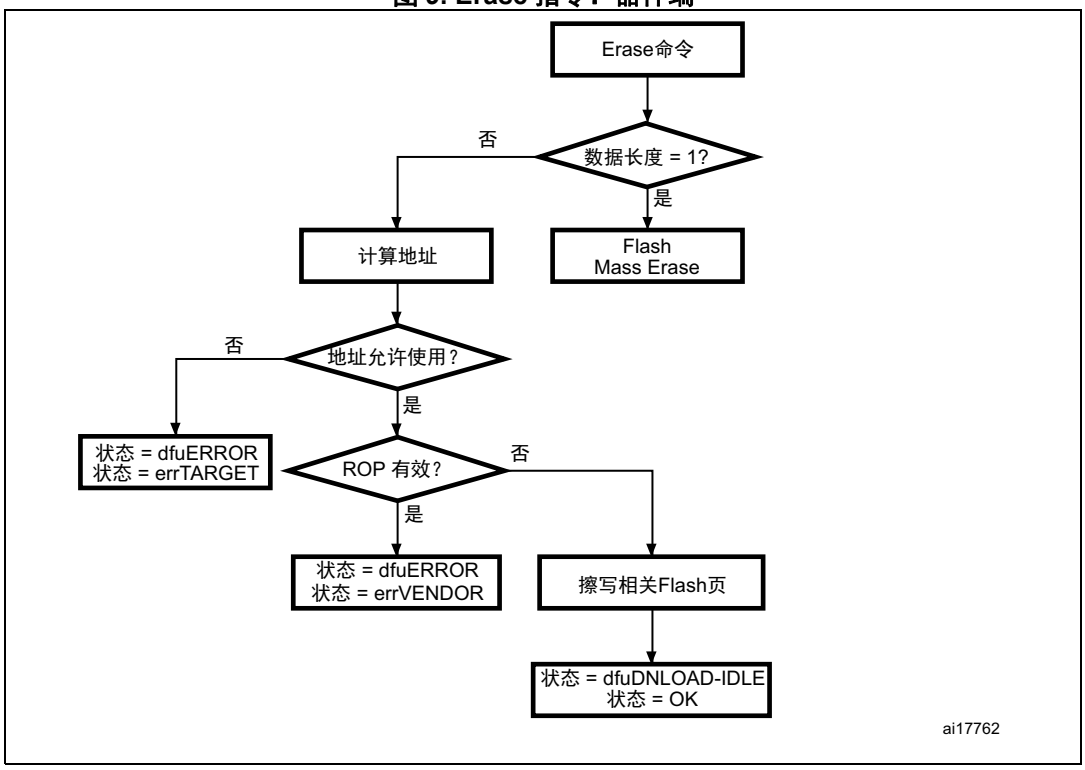
仅当 DFU_GETSTATUS 请求是由主机发出的情况下， Erase 指令才能有效执行。如果器件返回的状态不是 dfuDNBUSY，说明发生了错误。

需要再发一次 DFU_GETSTATUS 请求，检查指令是否正确执行。如果接收到的页地址不正确或不受支持，器件状态会变为 Status = dfuERROR， State = errTARGET。如果激活了 Flash Read Protection，器件会恢复为状态 Status = dfuERROR， State = errVENDOR，并且器件会忽略擦写操作。

允许的 Erase 页地址为内部 Flash 存储器地址。

注： 当向写保护的扇区执行Erase 操作时，不会返回错误。

图 9. Erase 指令：器件端



5.4 Read Unprotect 指令

如果 **wValue** = 0，并且主机发送的缓冲区的第一个字节是 0x92，则会选择 Read Unprotect 指令。缓冲区长度应仅为 1 个字节 （仅包含指令字节）。

主机会发送包含上述参数的 DFU_DNLOAD 请求，以取消对内部 Flash 存储器的读保护。

device 接收到的字节如下：

字节 1： 0x92 - Read Unprotect 指令

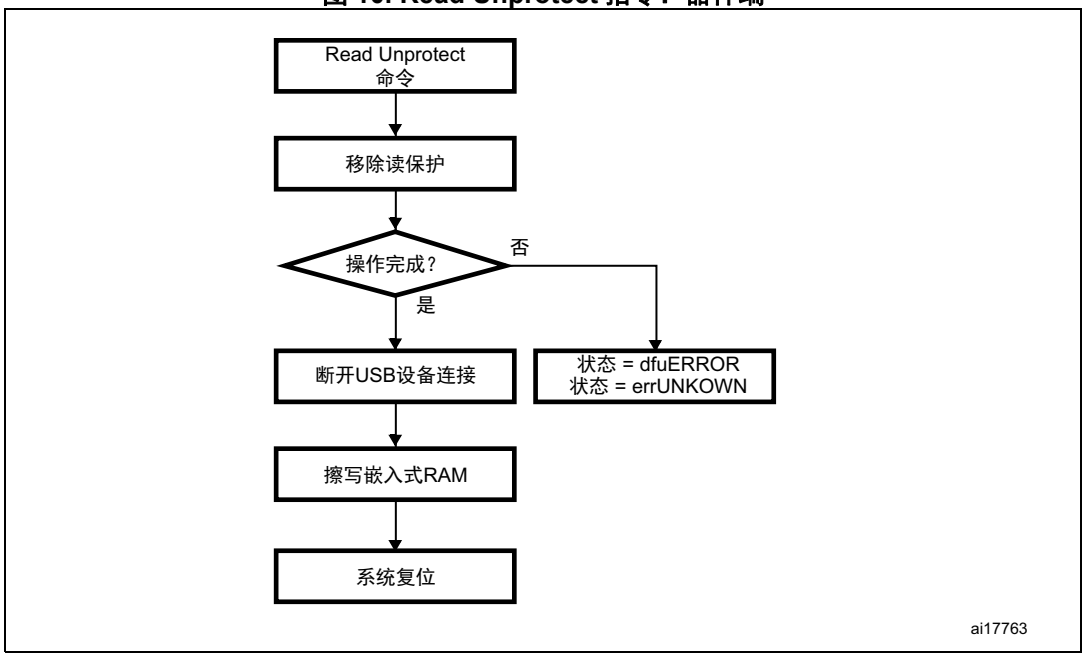
发送 Read Unprotect 指令后，主机需要发送 DFU_GETSTATUS 请求。

仅当 DFU_GETSTATUS 请求是由主机发出的情况下， Read Unprotect 指令才能有效执行。如果器件返回的状态不是 dfuDNBUSY，说明发生了错误。执行此操作后，器件会取消 Read Protection，进而会彻底擦除内部 Flash 存储器和嵌入式 RAM。

因此，执行完该指令之后，器件会立即断开自身连接，并会执行系统复位。在这种情况下，器件将无法响应下一个 Get Status 请求。主机必须等待器件再次被枚举。

还可以再发一次 DFU_GETSTATUS 请求 （器件仍保持连接的情况下），检查指令是否正确执行。如果器件无法执行指令，则会返回错误状态 （具体视错误类型而定）。

图 10. Read Unprotect 指令：器件端



5.5 Leave DFU mode

可以使用 DFU 下载请求退出 DFU 模式 （和自举程序）并跳转到已加载的应用程序 （内部 Flash 存储器或嵌入式 RAM 中）。

主机会发送数据长度为 0 的 DFU_DNLOAD 请求 （请求后没有数据阶段），以通知器件主机需要退出 DFU 模式。如果器件当前状态为 dfuDNLOAD-IDLE 或 dfuIDLE，则会确认该请求。

仅当 DFU_GETSTATUS 请求是由主机发出的情况下， DFU Leave 操作才能有效执行。如果器件返回的状态不是 dfuMANIFEST，说明发生了错误。执行该操作后，器件会执行以下操作：

- 断开自身连接
- 将自举程序所用外设的寄存器初始化至其默认复位值
- 初始化用户应用的主堆栈指针
- 跳转至收到的 ' 地址指针 + 4' 所编程的存储器位置，对应于应用复位处理程序的地址。例如，若收到的地址为 0x0800 0000，则自举程序跳转至编程为 0x0800 0004 地址的存储器位置。
  总之，主机发送基址，应用编程跳转。

需要在启动 Leave DFU 例程之前设定地址指针 （使用 Set Address Pointer 指令），否则，自举程序将跳转到默认地址 （内部 Flash 存储器起始地址：0x08000000）。

还可以通过上一 Write Memory 操作设置地址指针：如果执行的是下载操作，则将存储为此次下载使用的地址指针，并在后续跳转时使用该地址指针）。

注： *如果地址指针指向的地址不包含可执行代码，那么器件会复位，并可能再次进入自举程序模式 （具体视自举引脚的状态而定）。*

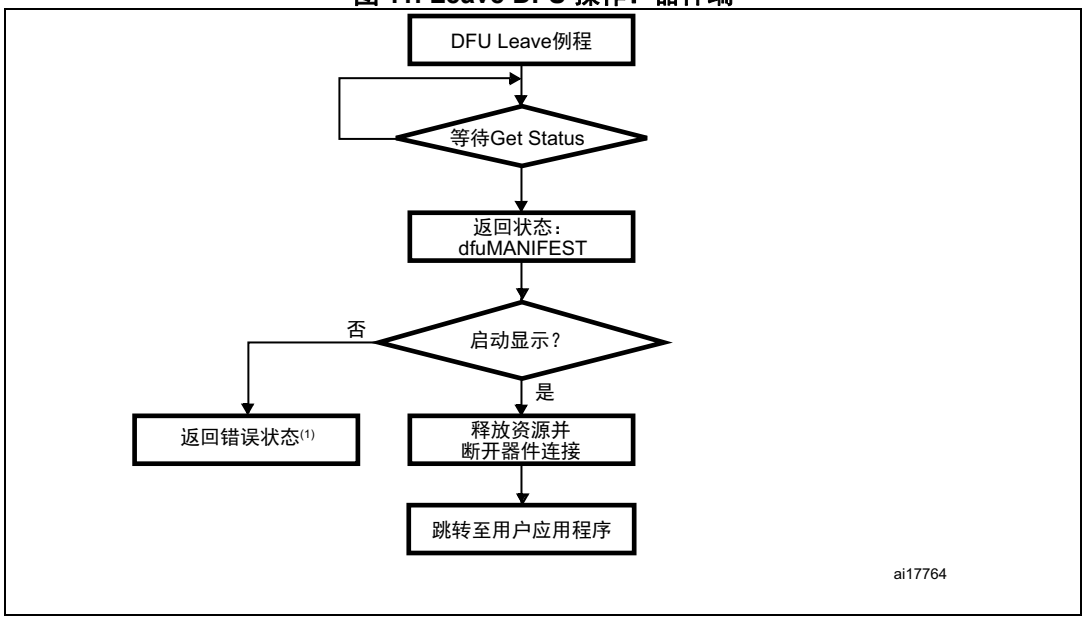
由于自举程序 DFU 应用不允许进行表示，表示阶段完成后，器件将无法响应主机请求。

还可以再发一次 DFU_GETSTATUS 请求 （器件仍保持连接的情况下），检查指令是否正确执行。如果器件无法执行指令，则会返回错误状态 （具体视错误类型而定）。

注： 1 *仅当用户应用正确设置了指向应用地址的向量表时，跳转到应用才能工作。*

2 *从自举程序跳转到使用 USB IP 的已加载应用程序代码时，用户应用程序需要先禁用所有待处理的 USB 中断并复位内核，然后再启用中断。否则，待处理中断 （通过自举程序代码发出）可能会干扰用户代码并导致函数错误。退出系统存储器自举模式后，不需要执行此步骤。*

图 11. Leave DFU 操作：器件端



1. 此状态取决于错误来源以及当前状态。

6 自举程序协议版本演进

表 5 列出了自举程序的版本。

表 5. 自举程序协议版本

版本	说明
V2.0	初始自举程序版本。
V2.1	DFU 自举程序版本 V2.1。该版本与版本 V2.0 的不同之处在于延长了接口描述符，包括 OTP 存储器接口和器件功能接口。 V2.0 和 V2.1 在不同器件上实施。请参见 AN2606 了解您的器件上实施的是哪种版本。 利用适当的超时解决了写入数据存储器时的时间问题。
V2.2	将选项字节、 OTP 和器件功能描述符更新为仅支持 Read/Write 操作，而不是 Read/Write/Erase 操作

7 版本历史

表 6. 文档版本历史

日期	版本	变更
2010 年 3 月 9 日	1	初始版本。
2011 年 4 月 15 日	2	在*第 1 节*中介绍了自举程序版本 V2.0 和 V2.1，并更新了自举程序序列的描述。增加了*图 2：其它 STM32 器件的自举程序*。 在*第 4.1 节：Read Memory* 中更新了从选项字节区域读取数据时允许的字节数，并添加了其它存储器位置。 在*第 5.1 节：Write memory* 中更新了写入选项字节区域时允许的字节数，并添加了其它存储器位置。 在*第 6 节*中添加了自举程序 V2.1。
2013 年 12 月 12 日	3	更改了*图 1：STM32 连接型器件的自举程序*的标题。 更新了*图 2：其它 STM32 器件的自举程序*，包括标题。 增加了*图 2* 下的*注：*。 增加了*表 1：适用产品*。
2014 年 4 月 30 日	4	更新了*表 1：适用产品*和*表 5：自举程序协议版本*。 更新了*第 1 节：自举程序代码序列*。 删除了专门介绍器件相关自举程序参数的章节。 更新了*图 2：其它 STM32 器件的自举程序*并添加了脚注 *3.*

表 7. 中文文档版本历史

日期	版本	变更
2017 年 6 月 24 日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。