# AN3156
# Application note

## USB DFU protocol used in the STM32 bootloader

## Introduction

This application note describes the USB DFU protocol used in STM32 microcontroller bootloader. It details each supported command. For more information about the USB hardware resources and requirements for your device bootloader, please refer to the “STM32 system memory boot mode” application note (AN2606).

**Table 1. Applicable products**

| Type | Part number or product series |
|---|---|
| Microcontrollers | STM32L1 series:<br>– STM32L1xxxC, STM32L1xxxD, STM32L1xxxE<br>STM32F0 series<br>STM32F1 series:<br>– STM32F105xx, STM32F107xx<br>STM32F2 series<br>STM32F3 series:<br>– STM32F373xx, STM32F302xx, STM32F303xB(C), STM32F301xx<br>STM32F4 series:<br>– STM32F401xx, STM32F411xx<br>– STM32F405xx, STM32F407xx<br>– STM32F415xx, STM32F417xx<br>– STM32F427xx, STM32F429xx<br>– STM32F437xx, STM32F439xx |

# Contents

1 Bootloader code sequence . . . . . 5

2 USB DFU bootloader requests . . . . . 8

3 DFU bootloader commands . . . . . 10

4 DFU_UPLOAD request commands . . . . . 12

4.1 Read memory . . . . . 12

4.2 Get command . . . . . 12

5 DFU_DNLOAD request commands . . . . . 14

5.1 Write memory . . . . . 17

5.2 Set Address Pointer command . . . . . 18

5.3 Erase command . . . . . 19

5.4 Read Unprotect command . . . . . 20

5.5 Leave DFU mode . . . . . 21

6 Bootloader protocol version evolution . . . . . 24

7 Revision history . . . . . 25

# List of tables

Table 1. Applicable products . . . . . 1
Table 2. DFU class requests . . . . . 8
Table 3. Summary of DFU class-specific requests . . . . . 8
Table 4. DFU bootloader commands . . . . . 11
Table 5. Bootloader protocol versions . . . . . 24
Table 6. Document revision history . . . . . 25

# List of figures

Figure 1. Bootloader for STM32 connectivity line devices . . . 6
Figure 2. Bootloader for other STM32 devices . . . 7
Figure 3. DFU_UPLOAD request: Device side . . . 13
Figure 4. DFU_UPLOAD request: Host side . . . 13
Figure 5. Download request: Device side . . . 15
Figure 6. Download request: Host side . . . 16
Figure 7. Write Memory: Device side . . . 18
Figure 8. Set Address Pointer command: Device side . . . 19
Figure 9. Erase command: Device side . . . 20
Figure 10. Read Unprotect command: Device side . . . 21
Figure 11. Leave DFU operation: Device side . . . 23

# 1 Bootloader code sequence

There is no difference in terms of protocol (requests and commands) between different Bootloader DFU versions. For detailed difference list please refer to Section 6.

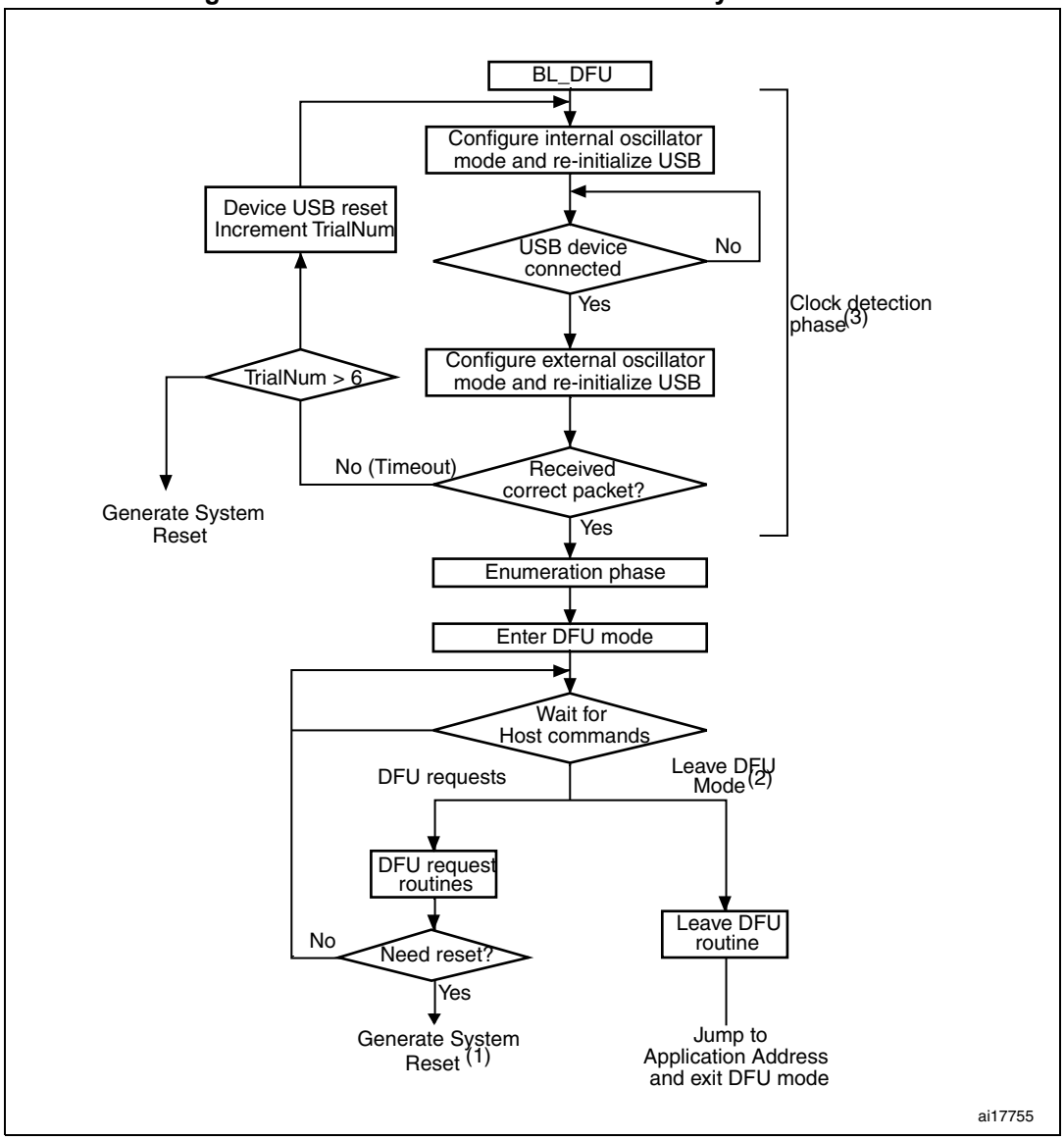
Once the system memory boot mode is entered and the STM32 device has been configured (for more details refer to AN2606), the bootloader code configures the USB and its interrupts, and waits for the "enumeration done" interrupt.

The USB enumeration is performed as soon as the USB cable is plugged (or immediately if the cable is already plugged). If you do not want the STM32 to enter the USB DFU bootloader application, the USB cable has to be unplugged before reset.

The bootloader version is returned in the device descriptor in the MSB of the bcd Device field (example: 0x2000 = Version 2.0).

For Connectivity line USB DFU bootloader, the device first tries the 25 MHz configuration, then, if it fails, the 14.7456 MHz configuration, and finally, if it fails, the 8 MHz configuration. If it still fails, this operation is repeated with a large timeout value (the three configurations are tested again). If the second trial also fails, a system reset is generated.

**Figure 1. Bootloader for STM32 connectivity line devices**

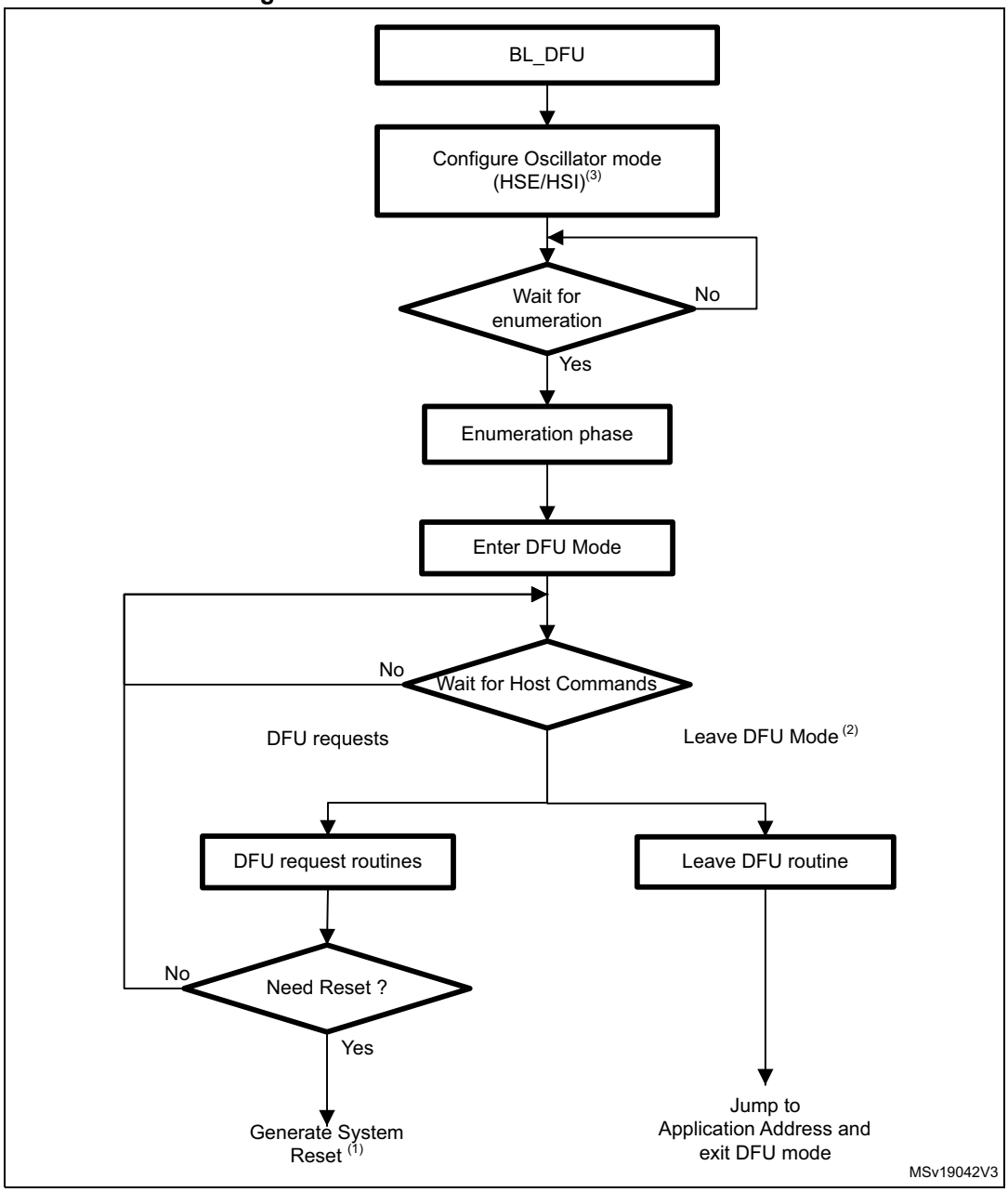



1. After system reset, the device may return to the BL_DFU loop or execute code from Flash memory/RAM depending on the connection states and the boot pin status.
2. Leave DFU is achieved by a 0 Data Download request followed by GetStatus request and Device Reset.
3. After six trials (the three clock configurations are tested twice), a System Reset is generated.

- If the product uses HSE for the USB operation (except Connectivity Line):
  - At startup the HSE is measured (if present) and if it is supported the USB is configured. If the HSE is not detected the Bootloader will perform a system reset. If the measured value of the HSE clock is unsupported value, the USB protocol will not function correctly.
- If the product uses the HSI for the USB operation.
  - At startup the USB is configured using HSI clock.

Refer to AN2606 for more details about product's configuration.

**Figure 2. Bootloader for other STM32 devices**




1. After system reset, the device may return to the BL_DFU loop or execute code from Flash memory/RAM depending on the connection states and the boot pin status.
2. Leave DFU is achieved by a 0 Data Download request followed by GetStatus request and Device Reset.
3. For some products, the external oscillator HSE is not used for USB Bootloader operations and only internal oscillator HSI is used. Check AN2606 product's section to know which oscillator is required for each product.

*Note: At Bootloader Startup, the internal oscillator (HSI) is used as clock source for the USB interface. When USB event is detected, the external oscillator is configured as USB clock source.*

# 2 USB DFU bootloader requests

USB DFU bootloader supports the DFU protocol and requests compliant with the “Universal Serial Bus Device Upgrade Specification for Device Firmware Upgrade” Version 1.1, Aug 5, 2004. For more details concerning these requests, refer to the specification.

*Table 2* and *Table 3* enumerate the DFU Class-Specific requests and their parameters.

**Table 2. DFU class requests**

| Request | Request code | Request description |
|---|---|---|
| DFU_DETACH | 0x00 | Requests the device to leave DFU mode and enter the application. |
| DFU_DNLOAD | 0x01 | Requests data transfer from Host to the device in order to load them into device internal Flash. Includes also erase commands. |
| DFU_UPLOAD | 0x02 | Requests data transfer from device to Host in order to load content of device internal Flash into a Host file. |
| DFU_GETSTATUS | 0x03 | Requests device to send status report to the Host (including status resulting from the last request execution and the state the device will enter immediately after this request). |
| DFU_CLRSTATUS | 0x04 | Requests device to clear error status and move to next step. |
| DFU_GETSTATE | 0x05 | Requests the device to send only the state it will enter immediately after this request. |
| DFU_ABORT | 0x06 | Requests device to exit the current state/operation and enter idle state immediately. |

*Note: The Detach request is not meaningful in the case of the bootloader. The bootloader is started by a system reset depending on the boot mode configuration settings, which means that no other application is running at this time.*

**Table 3. Summary of DFU class-specific requests**

| 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 | State |
| 00100001b | DFU_ABORT | Zero | Interface | Zero | None |

### Communication safety

The communication between host and device is secured by the embedded USB protection mechanisms (CRC checking, Acknowledgements, etc.). No further protection is performed for transferred data or for bootloader specific commands/data.

# 3 DFU bootloader commands

The DFU_DNLOAD and DFU_UPLOAD requests are mainly used to perform simple Write Memory and Read Memory operations. They are also used to initiate the integrated bootloader commands (write, read unprotect, erase, set address, etc.). The DFU_GETSTATUS command then triggers the command execution.

In the DFU download request the command is selected through the **wValue** parameter in the USB request structure. If **wValue** = 0 then the data sent by the host after the request is a bootloader command code. The first byte is the command code and the other bytes (if any) are the data related to this command.

In the DFU upload request the command is selected through the **wValue** parameter in the USB request structure. If **wValue** = 0 then Get Command is selected and performed.

**Table 4. DFU bootloader commands**

| DFU request | Bootloader command | Write protection disabled<br>Read protection disabled | Write protection enabled<br>Read protection disabled | Read protection enabled |
|---|---|---|---|---|
| DFU_UPLOAD | Read Memory | Allowed | Allowed | Not allowed |
| | Get | Allowed | Allowed | Allowed |
| DFU_DNLOAD | Write Memory | Allowed | Allowed(1) | Not allowed |
| | Erase | Allowed | Allowed**(1)** | Not allowed |
| | Read Unprotect | NA(2) | NA**(2)** | Allowed(3) |
| | Set Address Pointer | Allowed | Allowed | Allowed |
| | Leave DFU mode | Allowed | Allowed | Allowed |

1. This operation is allowed but not effective: the bootloader does not return an error but the operation is not executed since the sectors are write-protected. This applies only to the Flash memory. It does not apply to the RAM memory or the option byte area.
2. This operation is allowed but has no meaning since the memory is not protected.
3. In this case, both the Flash memory (from 0x0800 0000) and the RAM are erased. The option byte area is reset to default values.

If you perform a Read Unprotect operation while the memory is not protected, the entire RAM memory is cleared by the bootloader firmware and the Flash memory is not erased (since it was not previously read-protected).

There are no commands for the Write Protect, Write Unprotect and Read Protect operations. These operations should be performed through the Write Memory and Read Memory commands used for the option byte area.

# 4 DFU_UPLOAD request commands

The upload request allows different commands to be performed. The command selection is done through the value of parameter **wValue** in the USB request structure. The operations described in *Section 4.1* to *Section 5.5* are supported.

## 4.1 Read memory

The Read memory operation is selected when **wValue** > 1.

The host requests the device to send a specified number of data bytes (**wLength**) from valid memory address (see note) of the internal Flash memory, embedded RAM, system memory or from the option bytes.

*Note: Refer to Section 4: DFU_UPLOAD request commands for more details about the valid memory addresses for the device you are using.*

The allowed number of bytes to be read depends on the memory target:

- For the internal Flash memory, embedded RAM and system memory: read size can be from 2 to 2048 bytes
- For the option bytes: read size should be equal to the option byte block size
- For other memory locations, refer to the section "Important considerations" depending on the product in AN2606.

The address, from which the host requests to read data, is computed using the value of wBlockNumber (**wValue**) and the address pointer according to the following formula:

Address = ((wBlockNum – 2) × wTransferSize) + Address_Pointer, where:

– wTransferSize is the length of the requested data buffer.

The address pointer should be previously specified through a Set Address Pointer command (using a DFU_DNLOAD request). Otherwise (if no address was previously specified) the device assumes that it is the internal Flash start address (0x08000000).

If the Flash Read Protection is enabled, the Read operation is not performed and the returned device status is (Status = dfuERROR, State = errVENDOR) whatever the target (internal Flash memory, embedded RAM, system memory or option bytes).

## 4.2 Get command

This command is selected when **wValue** = 0.

The host requests to read the commands supported by the bootloader. After receiving this command, the device returns N bytes representing the command codes.

The STM32 sends bytes as follows (N = 4):

Byte 1: 0x00 - Get command

Byte 2: 0x21 - Set Address Pointer

Byte 3: 0x41 - Erase

Byte 4: 0x92 - Read Unprotect

The processing of the DFU_UPLOAD command is shown in *Figure 3* and *Figure 4*.

**Figure 3. DFU_UPLOAD request: Device side**

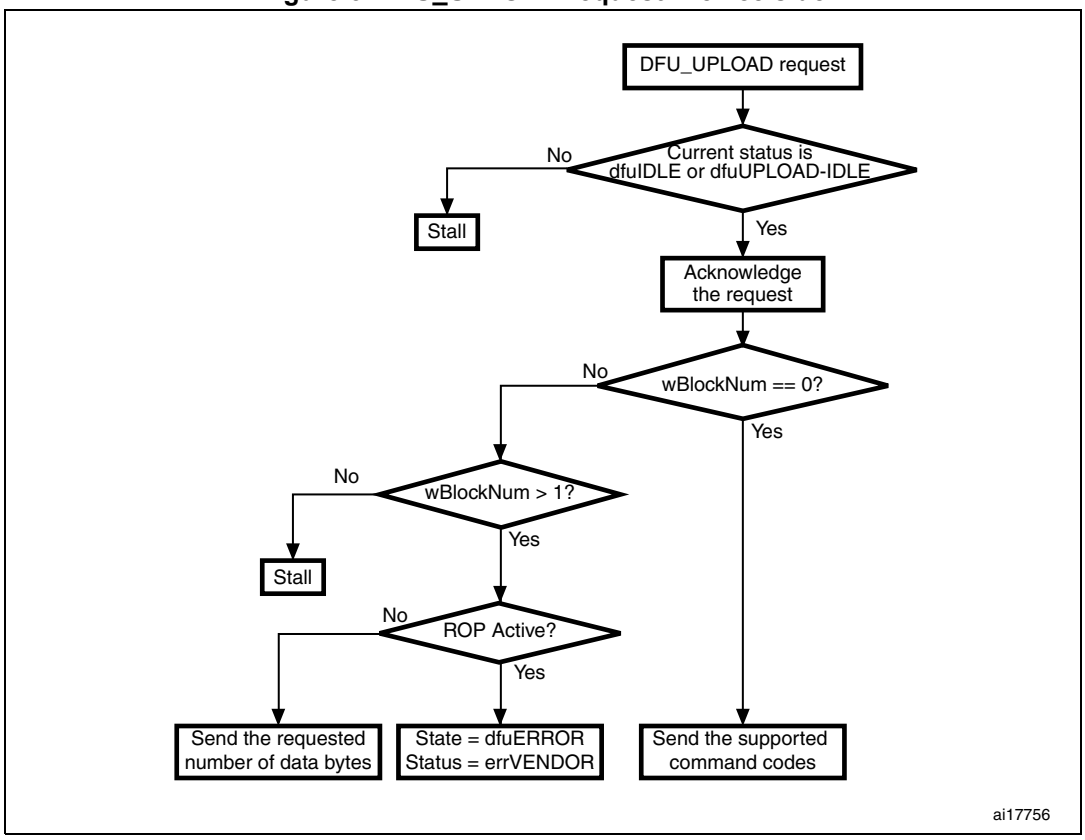
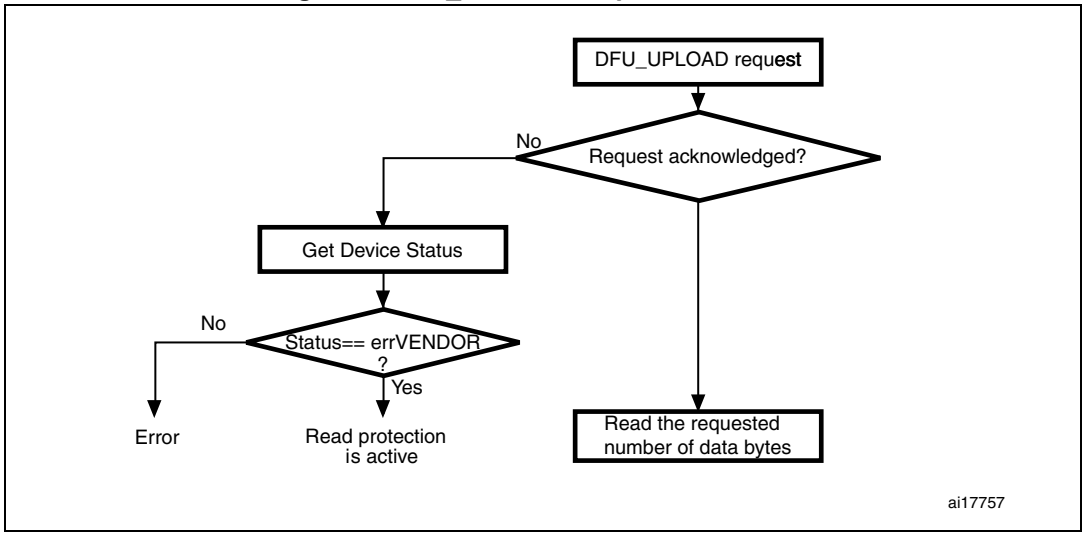



**Figure 4. DFU_UPLOAD request: Host side**



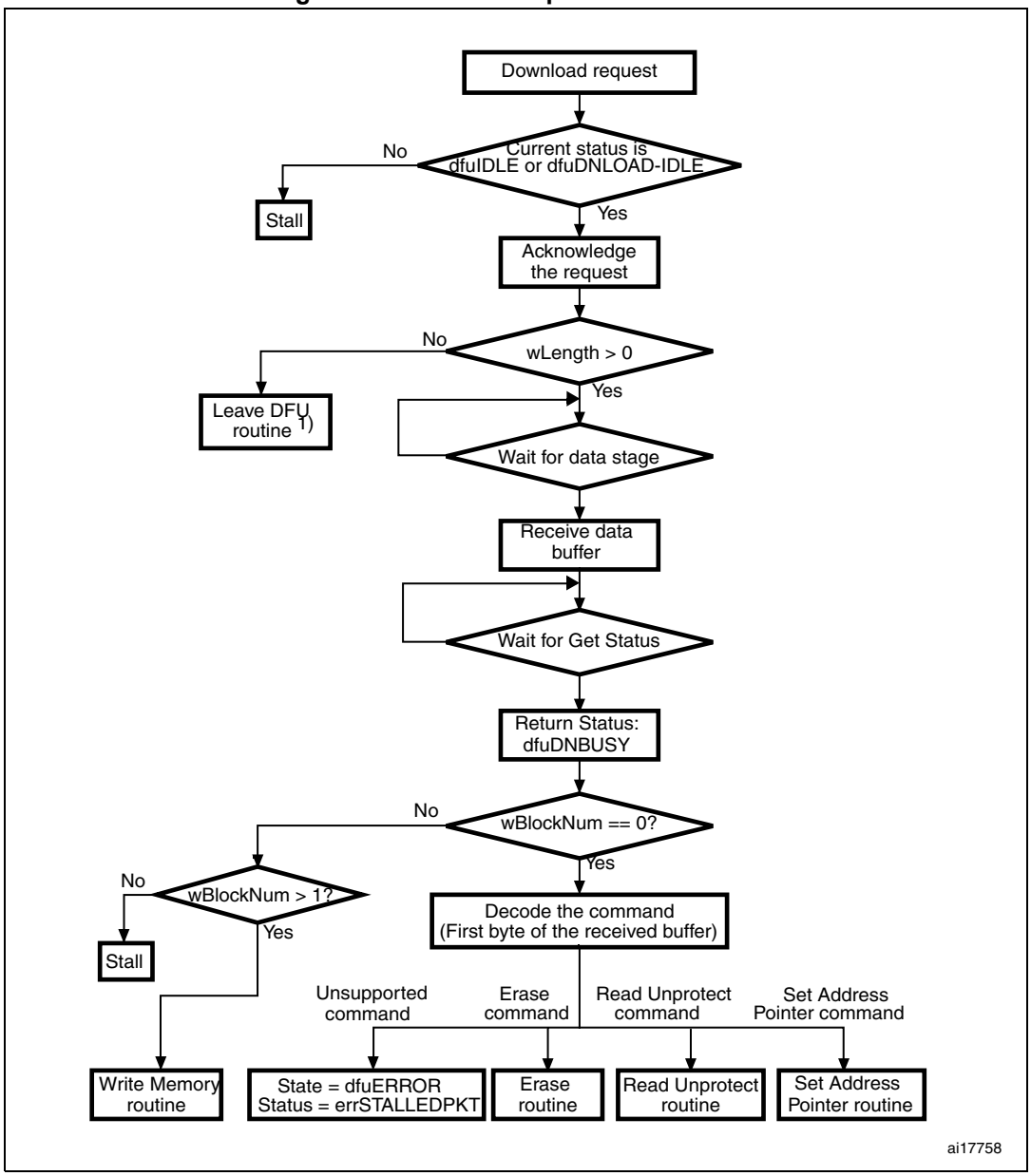

*Note: Before issuing an Upload request, the host has to check that the device is in a correct state (dfuIDLE or dfuUPLOAD-IDLE state) and that there is no error reported in the status. If the device is not in the required state/status, the host has to clear any error (DFU_CLRSTATUS request) and get the new status until the device returns to dfuIDLE state.*

# 5 DFU_DNLOAD request commands

The download request is used to perform different commands. The command selection is done through the value of parameter **wValue** in the USB request structure. The following operations are supported:

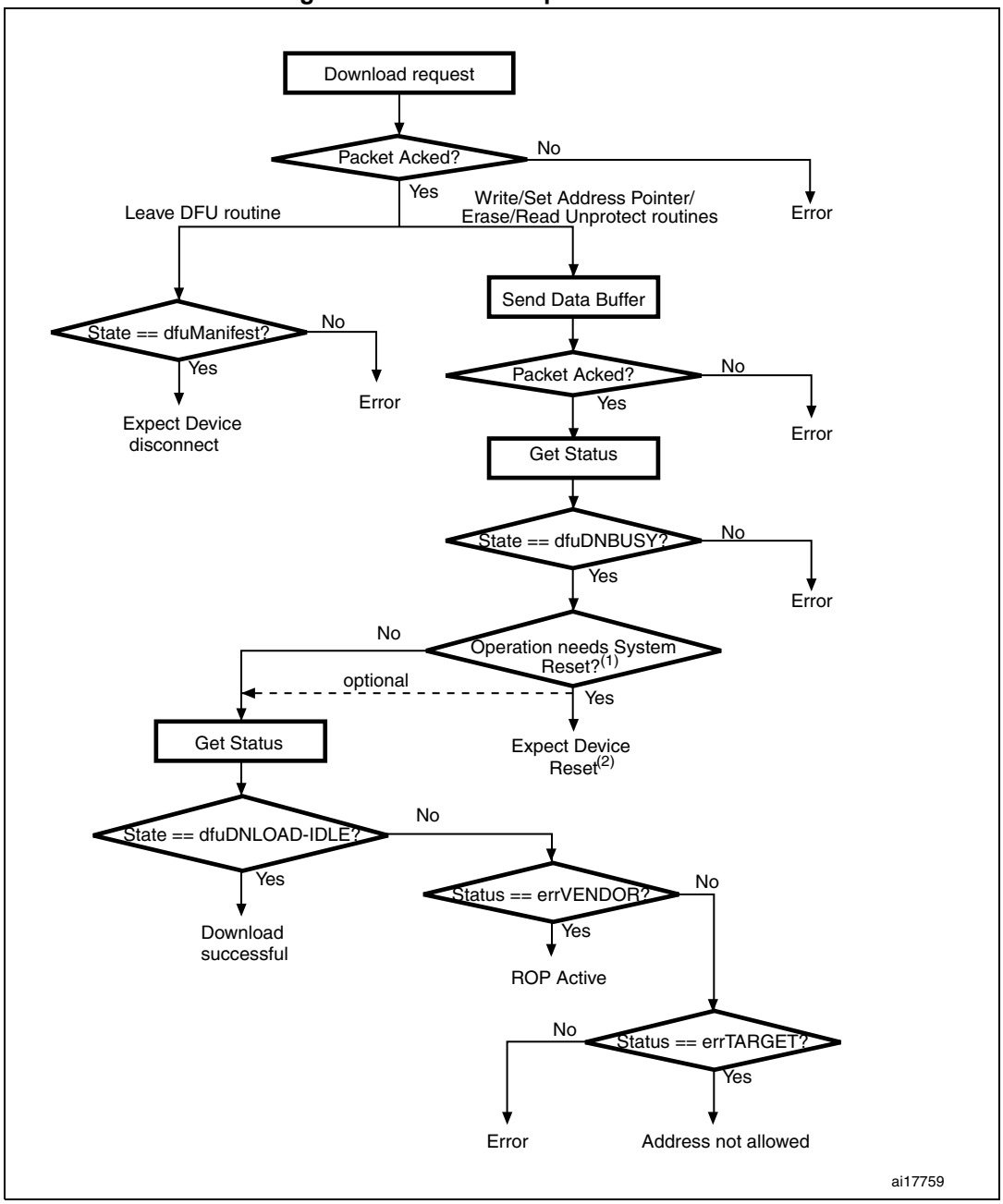
- Write Memory (**wValue** > 1)
- Set Address Pointer (**wValue** = 0 and first byte = 0x21)
- Erase (**wValue** = 0 and first byte = 0x41)
- Read Unprotect (**wValue** = 0 and first byte = 0x92)
- Leave DFU (leave DFU mode and Jump to application)

**Figure 5. Download request: Device side**




1. This routine can be used to reset the device or to jump to the application.

**Figure 6. Download request: Host side**




1. Operations needing System Reset are: Read Unprotect command and Write operations to the option bytes.
2. After returning to the dfuDNBUSY state, the device executes the requested operation and performs a system reset. The host may simply wait for the next enumeration or perform Get status again but the device will not be able to respond, unless it fails to execute the requested operation.

*Note: Before issuing a Download request, the host has to check that the device is in a correct state: dfuIDLE or dfuDNLOD-IDLE, and that there is no error reported in the status. If the device is not in the required state/status, the host has to clear any error (DFU_CLRSTATUS request) and get the status again until the device returns to the dfuIDLE state.*

## 5.1 Write memory

The Write memory operation is selected when **wValue** > 1.

The host requests the device to receive a specified number of data bytes (**wLength**) to load them into valid memory addresses (see note) in internal Flash memory, embedded RAM or option bytes.

*Note: Refer to Section 4: DFU_UPLOAD request commands for more details about the valid memory addresses for the device you are using.*

The allowed number of bytes to be written depends on the memory target:

- For the internal Flash memory and embedded RAM: write size can be from 2 to 2048 bytes
- For the option bytes: write size should be equal to the option byte block size
- For other memory locations, refer to the section "Important considerations" depending on the product in AN2606.

*Note: A different write size is possible for the option bytes but it is recommended to write the entire block at a time in order to ensure data integrity. When the target is the option byte area, the address pointer must always be the start address of the option bytes, otherwise, the request is not performed.*

The Write memory operation is effectively executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, then an error has occurred.

A second DFU_GETSTATUS request is needed to check if the command has been correctly executed, except when the destination is the option byte area (in this case the device is immediately reset after the write operation completion). If the received address is wrong or unsupported, the device status is then (Status = dfuERROR, State = errTARGET).

The address, to which the host requests to write data, is computed using the value of wBlockNumber (**wValue**) and the address pointer according to the same formula as for an upload request:

Address = ((wBlockNum – 2) × wTransferSize) + Addres_Pointer, where:

– wTransferSize: length of the data buffer sent by the host

– wBlockNumber: value of the **wValue** parameter

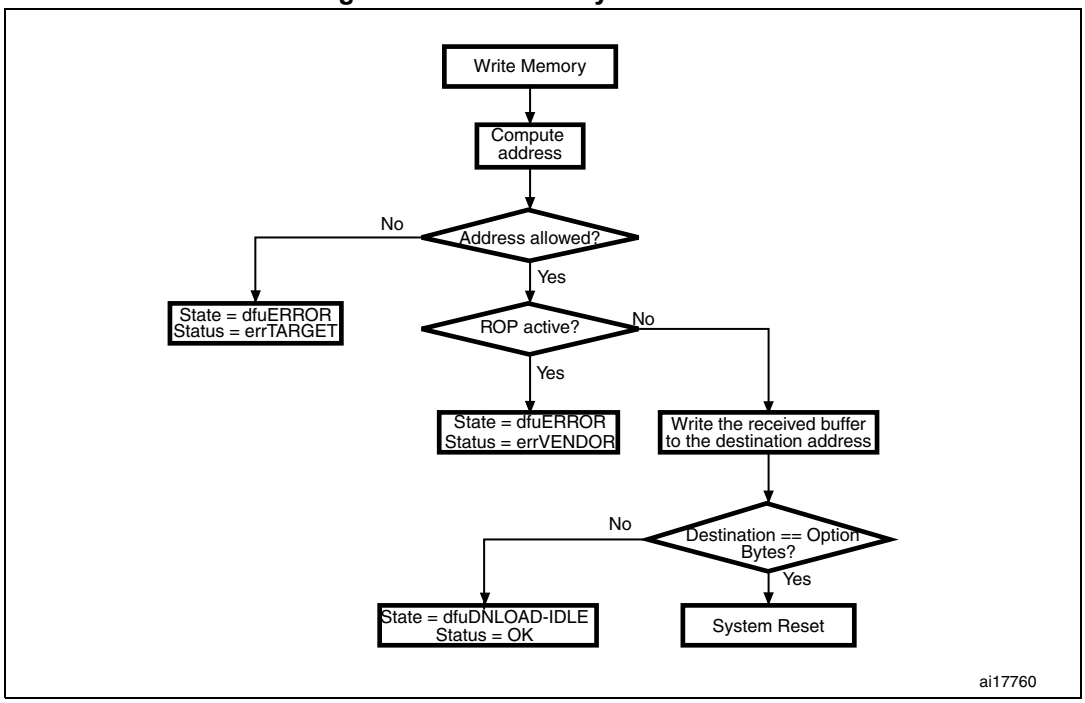
If the Flash Read Protection is enabled, the Write memory operation is not performed and the returned device status is (Status = dfuERROR, State = errVENDOR) whatever the target (internal Flash, embedded RAM or option bytes).

If the Write memory command is issued to the option byte area, all options are erased before writing the new values, and at the end of the command the bootloader generates a system reset to take into account the new configuration of the option bytes.

*Note: 1 When writing to the RAM, you should take care not to overlap the first RAM memory used by the bootloader firmware.*

*2 No error is returned when performing write operations on write protected sectors.*

Figure 7. Write Memory: Device side




## 5.2 Set Address Pointer command

The Set Address Pointer command is selected when **wValue** = 0 and the first byte of the buffer sent by the host is 0x21. The buffer length should be 5 (the four remaining bytes are the address bytes, LSB first (32-bit address format)).

The host sends a DFU_DNLOAD request with the above mentioned parameters to set the address pointer value used for computing the start address for Read and Write memory operations.

The STM32 receives bytes as follows:

Byte 1: 0x21 - Set Address Pointer command

Byte 2: A[7:0] - LSB of the address pointer

Byte 3: A[15:8] - Second byte of the address pointer

Byte 4: A[22:16] - Third byte of the address pointer

Byte 4: A[31:23] - MSB of the address pointer

After sending the Set Address Pointer command, the host has to send the DFU_GETSTATUS request.

The Set AddressPointer command is effectively executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, then an error has occurred.

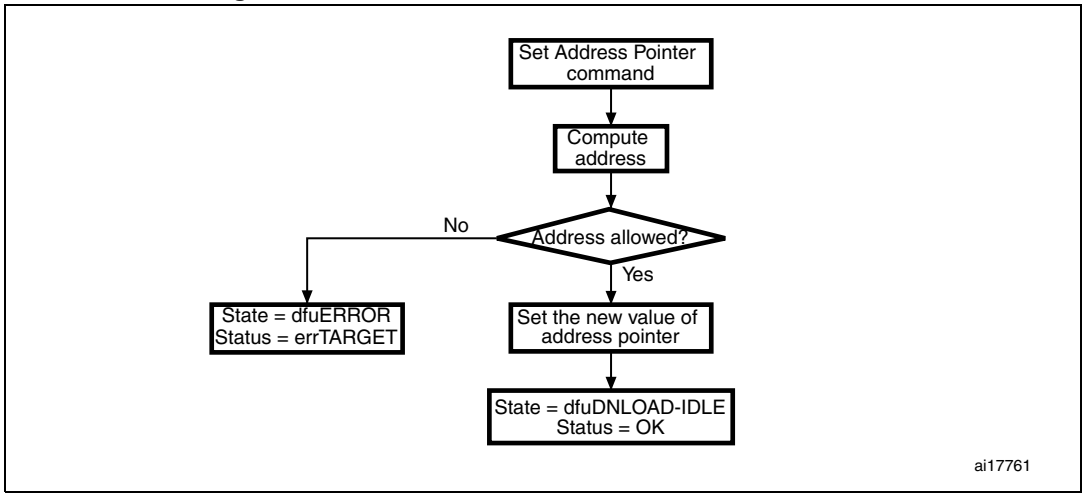
A second DFU_GETSTATUS request is needed to check if the command has been correctly executed. If the received address is wrong or unsupported, the device status is then (Status = dfuERROR, State = errTARGET).

The allowed locations for address pointer values are valid memory addresses (see note) in the internal Flash memory, embedded RAM, system memory and option bytes.

*Note: 1 Refer to Section 4: DFU_UPLOAD request commands for more details about the valid memory addresses for the device you are using.*

*2 The Set Address Pointer command is allowed and executed when the Flash Read Protection is enabled or disabled.*

**Figure 8. Set Address Pointer command: Device side**




## 5.3 Erase command

The Erase command is selected when **wValue** = 0 and the first byte of the buffer sent by the host is 0x41. The buffer length may be 5 bytes (the four remaining bytes are the address bytes, LSB first) for the page erase operation or only 1 byte (only the command byte) for the Mass erase operation.

The host sends a DFU_DNLOAD request with the above parameters to erase one page of the internal Flash memory or to perform a mass erase of this Flash memory.

The device receives the bytes as follows (page erase):

Byte 1: 0x41 - Erase command

Byte 2: A[7:0] - LSB of the page address

Byte 3: A[15:8] - Second byte of the page address

Byte 4: A[22:16] - Third byte of the page address

Byte 4: A[31:23] - MSB of the page address

Or, if a 1-byte command is received:

The STM32 receives the bytes as follows (Mass Erase):

Byte 1: 0x41 - Erase command

After sending an Erase command, the host has to send a DFU_GETSTATUS request.

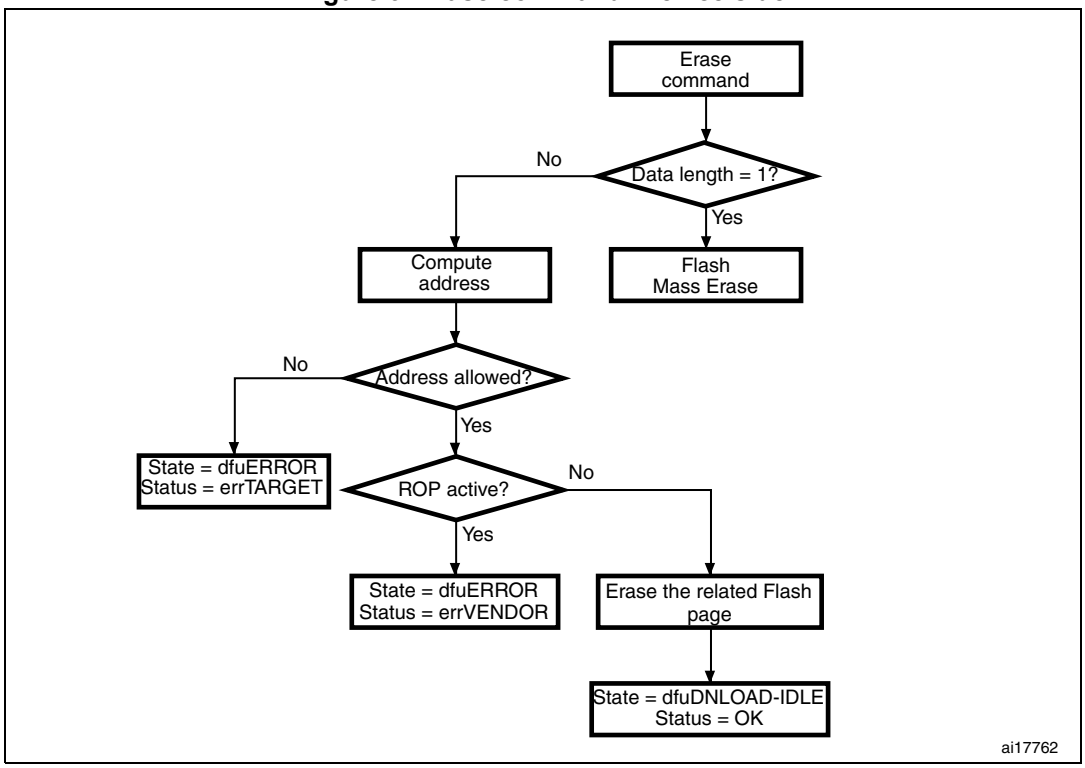
The Erase command is effectively executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, then an error has occurred.

A second DFU_GETSTATUS request is needed to check if the command has been correctly executed. If the received page address is wrong or unsupported, the device status is then (Status = dfuERROR, State = errTARGET). If the Flash Read Protection is active, then the device returns the status (Status = dfuERROR, State = errVENDOR) and the erase operation is ignored by the device.

The allowed Erase page addresses are internal Flash memory addresses.

*Note: No error is returned when performing Erase operations on write protected sectors.*

**Figure 9. Erase command: Device side**




## 5.4 Read Unprotect command

The Read Unprotect command is selected when **wValue** = 0 and the first byte of the buffer sent by the host is 0x92. The buffer length should be only 1 byte (only the command byte).

The host sends a DFU_DNLOAD request with the above parameters to remove the read protection of the internal Flash memory.

The device receives the byte as follows:

Byte 1: 0x92 - Read Unprotect command

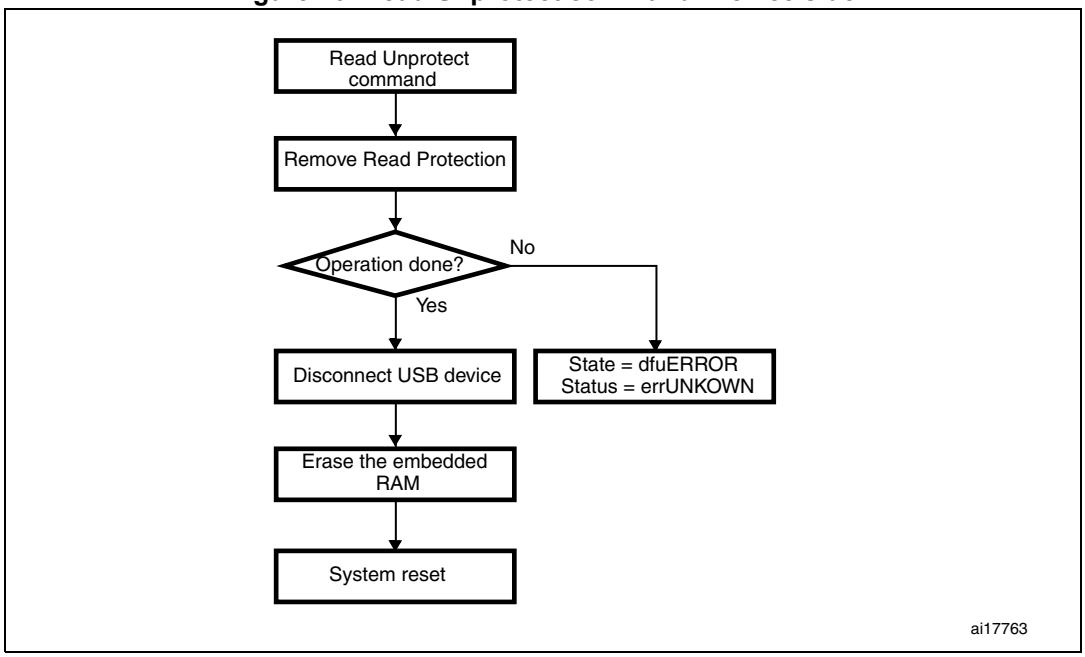
After sending a Read Unprotect command, the host has to send a DFU_GETSTATUS request.

The Read Unprotect command is effectively executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, then an error has occurred. After this operation, the device removes the Read Protection and, consequently, both the internal Flash and the embedded RAM are fully erased.

Hence, just after executing this command, the device disconnects itself and executes a system reset. In this case, the device is not able to respond to a second Get Status request. And the host has to wait until the device is enumerated again.

A second DFU_GETSTATUS request may also be issued (if the device is still connected) to check if the command has been correctly executed. If the device fails to execute the command it returns an error status (depending on the error type).

**Figure 10. Read Unprotect command: Device side**




## 5.5 Leave DFU mode

It is possible to exit DFU mode (and bootloader) and jump to a loaded application (in the internal Flash or in the embedded RAM) using the DFU download request.

The Host sends a DFU_DNLOAD request with 0 data length (no data stage after the request) in order to inform the device that it will have to exit DFU mode. The device acknowledges this request if the current state is dfuDNLOAD-IDLE or dfuIDLE.

The DFU Leave operation is effectively executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuMANIFEST, then an error has occurred. After this operation, the device performs the following:

- it disconnects itself
- it initializes the registers of the peripherals used by the bootloader to their default reset values
- it initializes the user application's main stack pointer
- it jumps to the memory location programmed in the received 'address pointer + 4', which corresponds to the address of the application's reset handler
  For example if the received address is 0x0800 0000, the bootloader will jump to the memory location programmed at address 0x0800 0004.
  In general, the host should send the base address where the application to jump to is programmed.

The address pointer has to be set (using the Set Address Pointer command) before launching the Leave DFU routine, otherwise, the bootloader will jump to the default address (internal Flash memory start address: 0x08000000).

The address pointer can also be set through the last Write Memory operation: if a download operation is performed, the address pointer used for this download will be stored and used later for the jump.

*Note: If the address pointer points to an address that does not contain executable code, then the device is reset and, depending on the state of the boot pins, may re-enter the bootloader mode.*

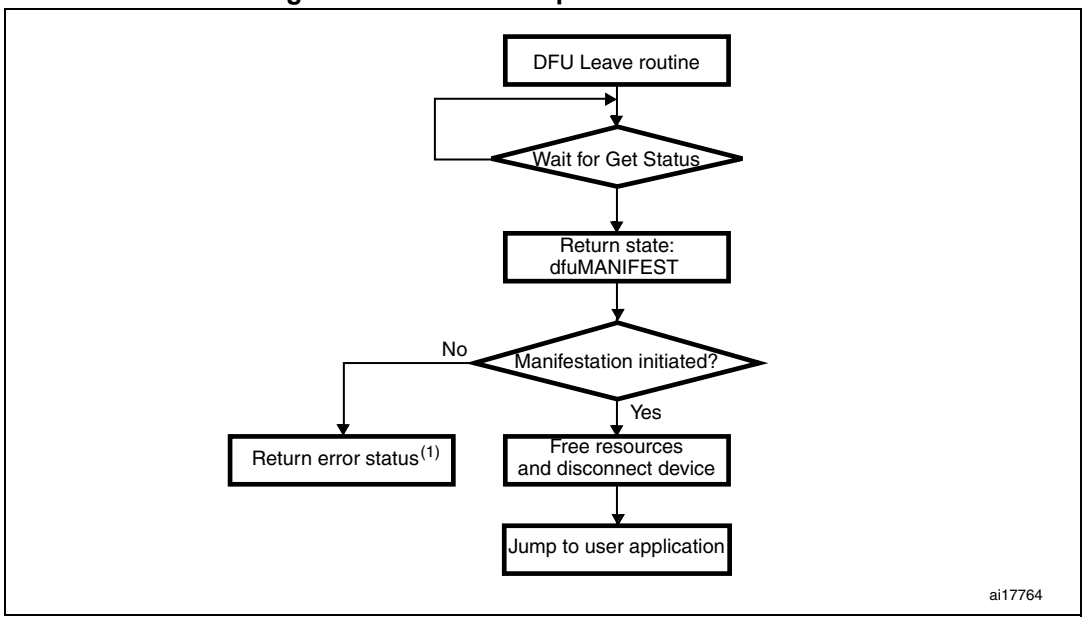
Since the bootloader DFU application is not manifestation-tolerant, the device will not be able to respond to host requests after a manifestation phase is completed.

A second DFU_GETSTATUS request may also be issued (if the device is still connected) to check if the command has been correctly executed. If the device fails to execute the command it returns an error status (depending on the error type).

*Note: 1 The Jump to application works only if the user application sets the vector table correctly to point to the application address.*

*2 When performing a jump from the bootloader to a loaded application code which uses the USB IP, the user application has to disable all pending USB interrupts and reset the core before enabling interrupts. Otherwise, a pending interrupt (issued from the bootloader code) may interfere with the user code and cause a functional failure. This procedure is not needed after exiting the system memory boot mode.*

**Figure 11. Leave DFU operation: Device side**




1. This status depends on the error origin and the current status.

# 6 Bootloader protocol version evolution

*Table 5* lists the bootloader versions.

**Table 5. Bootloader protocol versions**

| Version | Description |
|---|---|
| V2.0 | Initial bootloader version. |
| V2.1 | DFU Bootloader version V2.1. Unlike version V2.0, this version features an extended interface descriptor including OTP memory interface and Device Feature interface.<br>V2.0 and V2.1 are implemented on different devices. Refer to AN2606 to know which version is implemented on your device.<br>Fixed bug found when writing in data memory timing by using an adequate timeout. |
| V2.2 | Updated Option bytes, OTP and device feature descriptors to support only Read/Write operation instead of Read/Write/Erase |

# 7 Revision history

Table 6. Document revision history

| Date | Revision | Changes |
|---|---|---|
| 09-Mar-2010 | 1 | Initial release. |
| 15-Apr-2011 | 2 | Introduced bootloader version V2.0 and V2.1 and update description of bootloader sequence n *Section 1*. Added *Figure 2: Bootloader for other STM32 devices*.<br>Updated allowed number of bytes when reading from option byte area, and added other memory locations in *Section 4.1: Read memory*.<br>Updated allowed number of bytes when writing to option byte area, and added other memory locations in *Section 5.1: Write memory*.<br>Added bootloader V2.1 in *Section 6*. |
| 12-Feb-2013 | 3 | Changed title for *Figure 1: Bootloader for STM32 connectivity line devices*.<br>Updated *Figure 2: Bootloader for other STM32 devices* including title.<br>Added *Note:* below *Figure 2*.<br>Added *Table 1: Applicable products*. |
| 30-Apr-2014 | 4 | Updated *Table 1: Applicable products* and *Table 5: Bootloader protocol versions*.<br>Updated *Section 1: Bootloader code sequence*.<br>Removed section dedicated to Device-dependent bootloader parameters.<br>Updated *Figure 2: Bootloader for other STM32 devices* and added footnote *3.* |

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2014 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

**www.st.com**