AN3156
Application note

How to use USB DFU protocol in bootloader on STM32 MCUs

Introduction

This application note describes the USB DFU protocol used in STM32 microcontroller bootloader, detailing each supported command.

This document applies to the STM32 products embedding bootloader versions V3.x, V4.x, V7.x, V9.x, V10.x, V13.x, and V14x, as specified in AN2606 "*STM32 microcontroller system memory boot mode*" (available on *www.st.com)*, which also contains more information about the USB hardware resources and requirements for the device bootloader. These products are listed in *Table 1*, and are referred to as STM32 throughout the document.

Table 1. Applicable products

Type	Series
Microcontrollers	STM32C0 series
	STM32C5 series
	STM32F0 series
	STM32F1 series
	STM32F2 series
	STM32F3 series
	STM32F4 series
	STM32F7 series
	STM32G0 series
	STM32G4 series
	STM32H5 series
	STM32H7 series
	STM32L0 series
	STM32L1 series
	STM32L4 series
	STM32L5 series
	STM32U0 series
	STM32U3 series
	STM32U5 series
	STM32WB series
	STM32WBA series

Contents

List of tables

List of figures

1   Bootloader code sequence

There are no differences in terms of protocol (requests and commands) between different bootloader DFU versions. For the detailed difference list, refer to *Section 6*.

Once the system memory boot mode is entered and the STM32 microcontroller (based on Arm® Cortex®[a] cores) 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 the user does not want the microcontroller to enter the USB DFU bootloader application, the USB cable must 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. In case of fail, this operation is repeated with a higher timeout value (the three configurations are tested again). If the second trial fails, a system reset is generated.

arm

---

a.   Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

    The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

Figure 1. Bootloader for STM32 connectivity line devices

BL_DFU

Configure internal oscillator mode and re-initialize USB

Device USB reset Increment TrialNum

USB device connected — No

Yes

Clock detection phase[3]

TrialNum > 6

Configure external oscillator mode and re-initialize USB

Generate System Reset

No (Timeout) — Received correct packet?

Yes

Enumeration phase

Enter DFU mode

Wait for Host commands

DFU requests

Leave DFU Mode[2]

DFU request routines

No — Need reset?

Yes

Generate System Reset[1]

Leave DFU routine

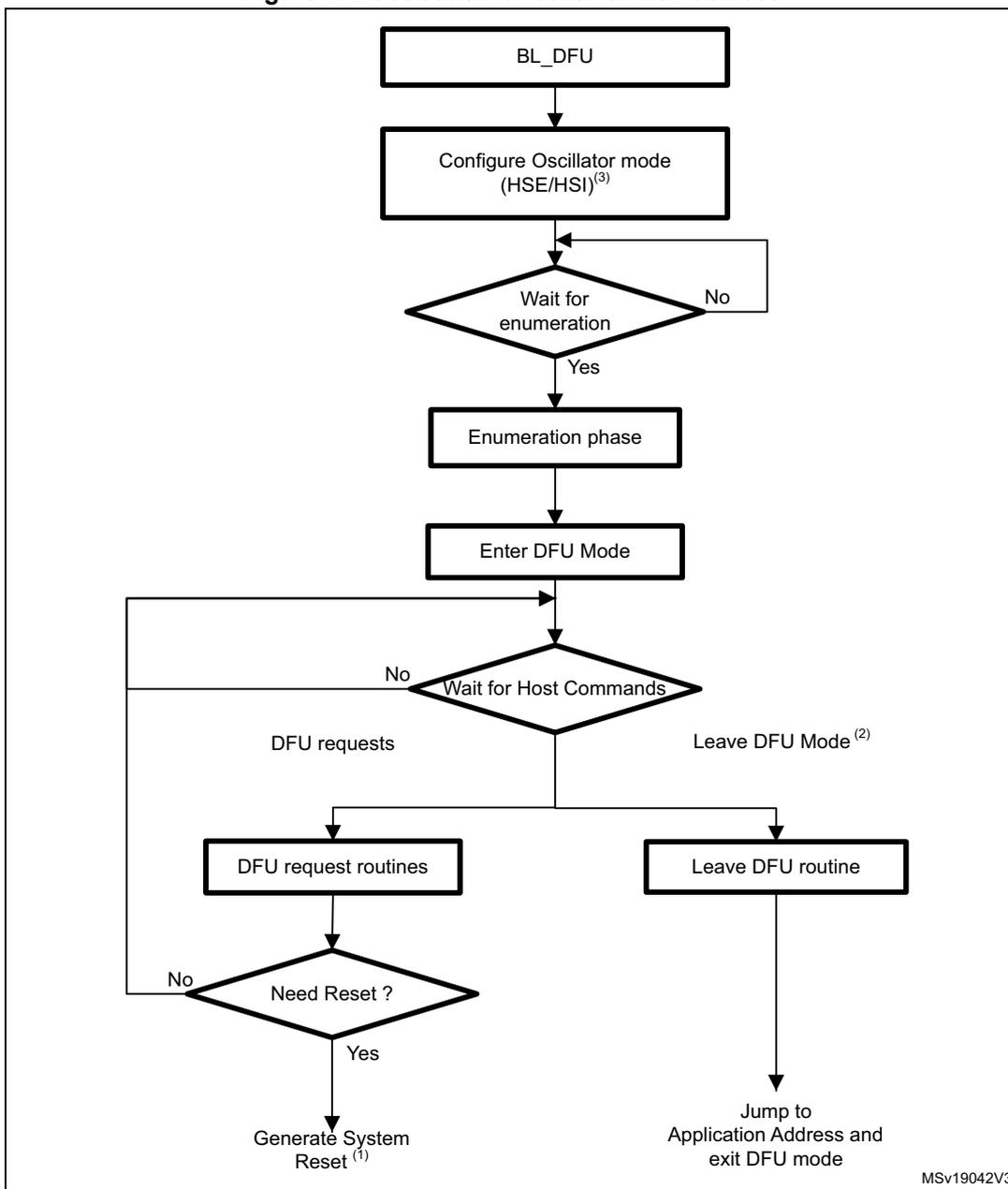Jump to Application Address and exit DFU mode

ai17755

1. After system reset, the device returns to the BL_DFU loop or executes code from flash memory/RAM, depending upon 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 supported the USB, is configured. If the HSE is not detected the Bootloader performs a system reset. If the measured value of the HSE clock is an unsupported value, the USB protocol does not work correctly.
- If the product uses the HSI for the USB operation:
  – At startup the USB is configured using the HSI clock.

Refer to AN2606 for more details about product configuration.

Figure 2. Bootloader for other STM32 devices

1. After system reset, the device returns to the BL_DFU loop or execute code from flash memory/RAM, depending upon 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, only the internal oscillator HSI is used. Check AN2606 to know which oscillator is required for each product.

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

2 USB DFU bootloader requests

The 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* detail the DFU class-specific requests and their parameters.

Table 2. DFU class requests

Request	Code	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 memory. Includes also erase commands.
DFU_UPLOAD	0x02	Requests data transfer from device to Host in order to load content of device internal flash memory 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 enters 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 enters 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 starts with a system reset depending on the boot mode configuration settings, which means that no other application is running at that time.*

Table 3. Summary of DFU class-specific requests

bmRequest	bRequest	wValue	wIndex	wLength	Data
00100001b	DFU_DETACH	wTimeout	Interface	0	None
00100001b	DFU_DNLOAD	wBlockNum	Interface	Length	Firmware
10100001b	DFU_UPLOAD	wBlockNum	Interface	Length	Firmware
10100001b	DFU_GETSTATUS	0	Interface	6	Status
00100001b	DFU_CLRSTATUS	0	Interface	0	None
0010001b	DFU_GETSTATE	0	Interface	1	State
00100001b	DFU_ABORT	0	Interface	0	None

Communication safety

The communication between host and device is secured by the embedded USB protection mechanisms (such as CRC checking, acknowledgments). 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 and read memory operations. They are also used to initiate the integrated bootloader commands (such as Write, Read Unprotect, Erase, Set Address). 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, 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 the command.

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

Table 4. DFU bootloader commands

DFU request	Bootloader command	Write protection disabled	Write protection enabled	Protection enabled
		Protection disabled		
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	N/A[2]	N/A[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 or to the option byte area.

2. Not applicable (this operation is allowed, but has no meaning as 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 the default values.

If the user performs a Read Unprotect operation while the memory is not protected, the whole RAM is cleared by the bootloader firmware, while the flash memory is not erased (since it was not read-protected).

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

4 DFU_UPLOAD request commands

The upload request enables the execution of different commands. Their selection is done through the value of **wValue** in the USB request structure. The supported operations are described in sections *4.1* to *5.5*.

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 *Section 4*) of the internal flash memory, embedded RAM, system memory, or from the option bytes.

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

- for the internal flash memory, the embedded RAM, and the system memory, read size can be from 2 to 2048 bytes
- for the option bytes read size must be equal to the option byte block size
- for other memory locations, refer to 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 must 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 memory start address (0x08000000).

If the flash memory 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

Beginning from the USB BL version V3.0, the number of commands is no more fixed. Some commands depend upon the HW features, hence they can be omitted.

As an example, for the STM32H5 series there is no RDP HW feature. The Get command is:

Byte 1:   0x00   Get command

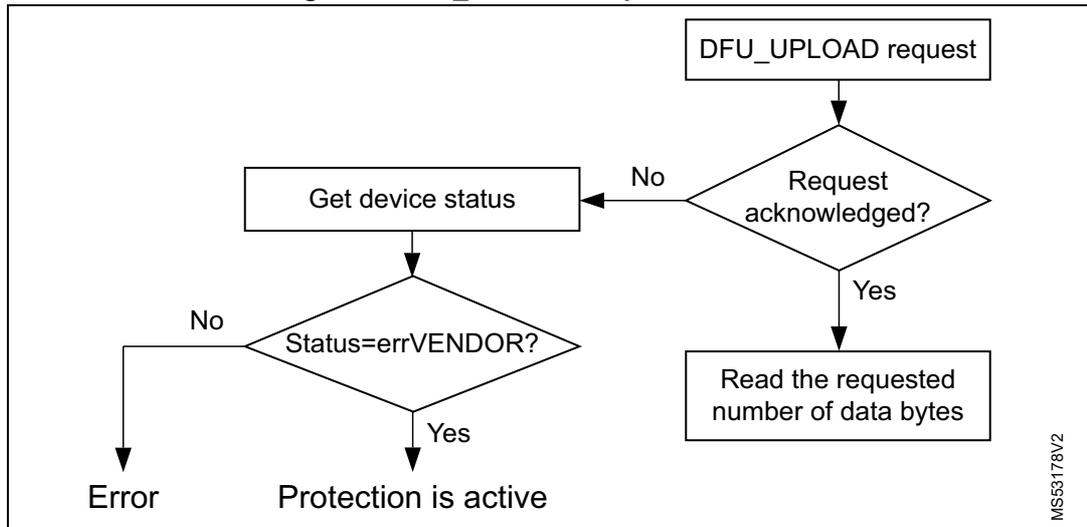Byte 2:   0x21   Set Address Pointer

Byte 3:   0x41   Erase

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

Figure 3. DFU_UPLOAD request: device side
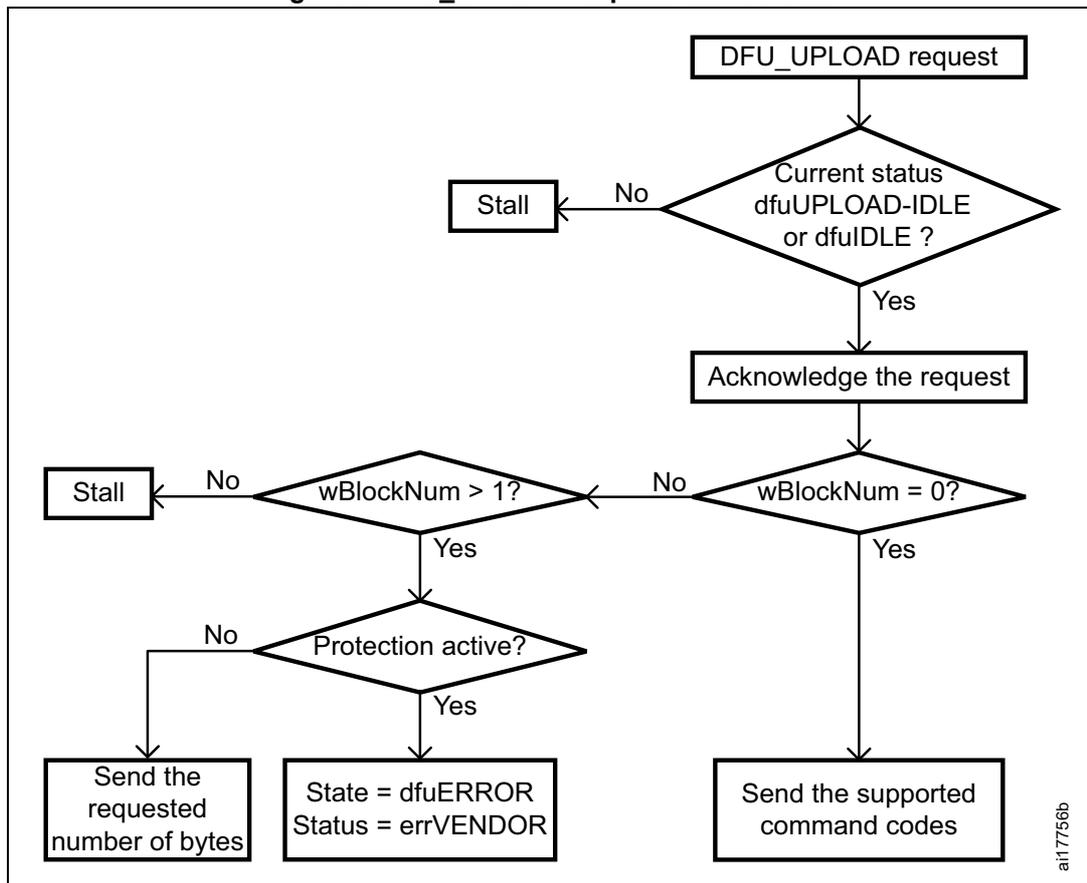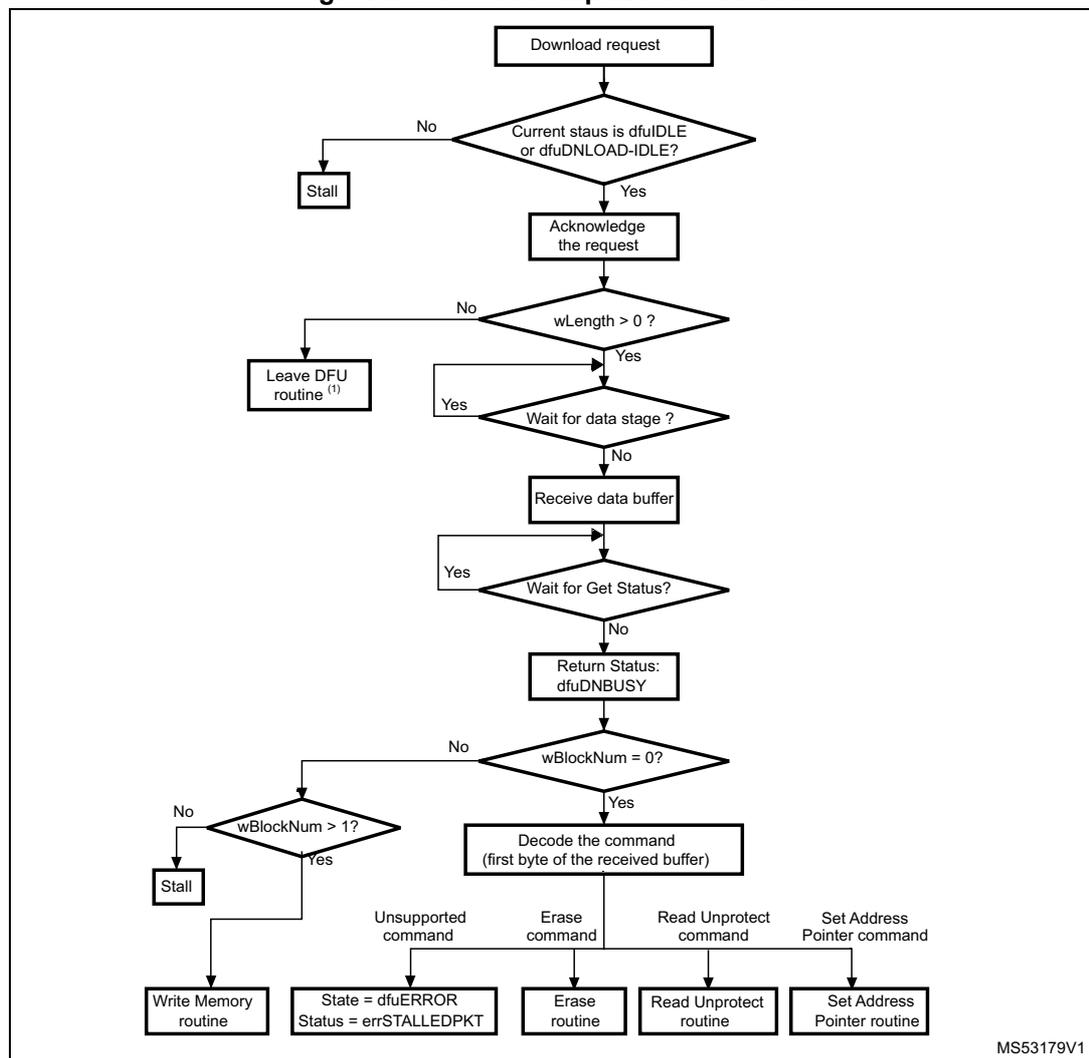
Figure 4. DFU_UPLOAD request: host side

Note: *Before issuing an Upload request, the host must 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, the host must 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)[a]
- Leave DFU (leave DFU mode and jump to application)
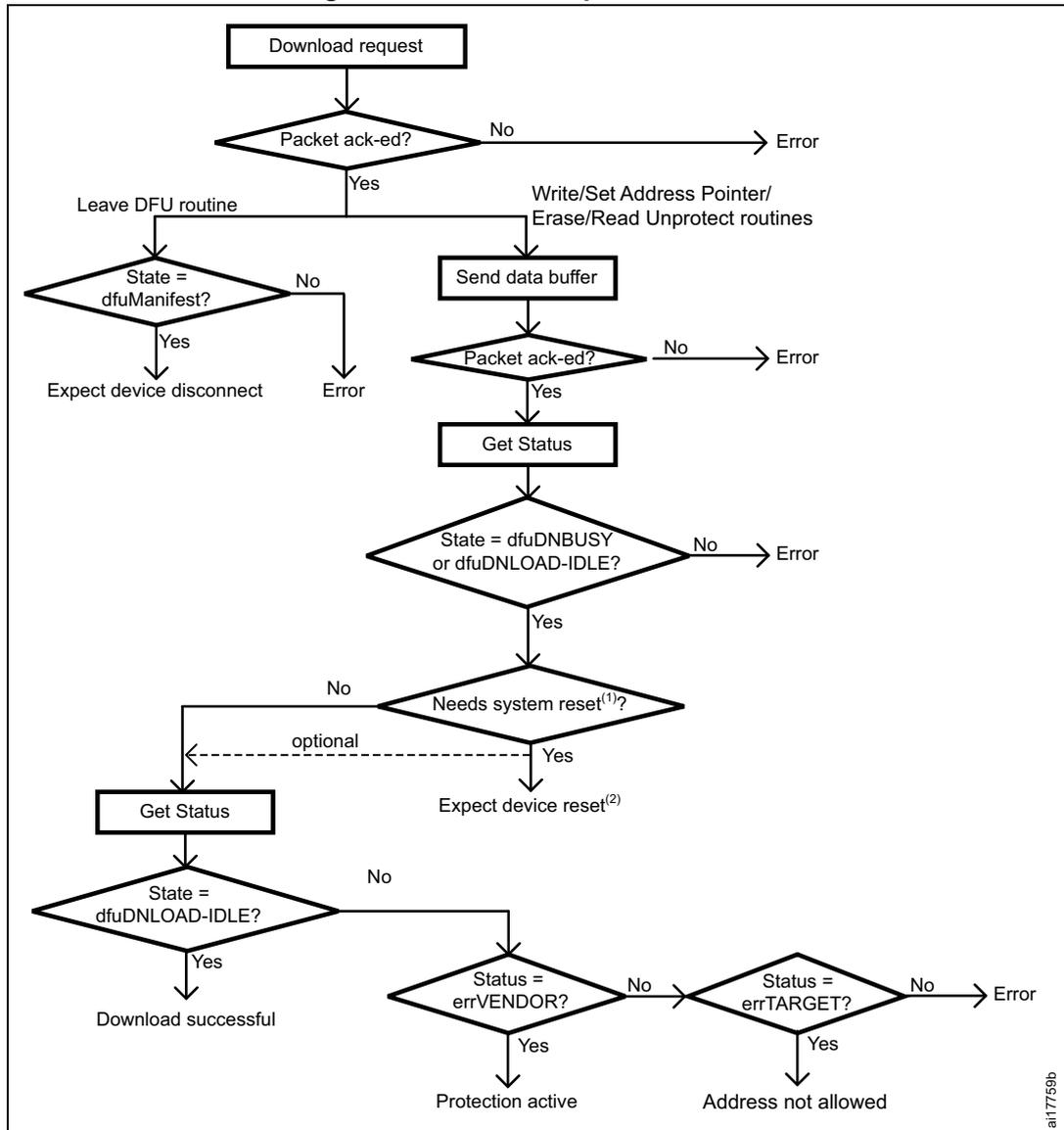
Figure 5. Download request: device side

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

a. Can be not supported from protocol version V3.0.

Figure 6. Download request: host side

1. Read Unprotect command and Write operations to the option bytes need system reset.

2. After returning to the dfuDNBUSY state, the device executes the requested operation and performs a system reset. The host can wait for the next enumeration or perform Get status again, but the device is unable to respond, unless it fails to execute the requested operation.

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

5.1 Write memory

This 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 *Section 4*) in the internal flash memory, RAM, or option bytes.

The allowed number of bytes to write depends upon the target memory:

- for the internal flash memory and embedded RAM the write size ranges from 2 to 2048 bytes
- for the option bytes the write size must be equal to the option byte block size
- for other memory locations refer to AN2606.

Note: *A different write size is possible for the option bytes, but it is recommended to write the entire block with a single operation, 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, an error 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 (Status = dfuERROR, State = errTARGET).

The address where 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 memory 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 memory, 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, take care not to overlap the first RAM used by the bootloader firmware.*

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

Figure 7. Write memory: device side
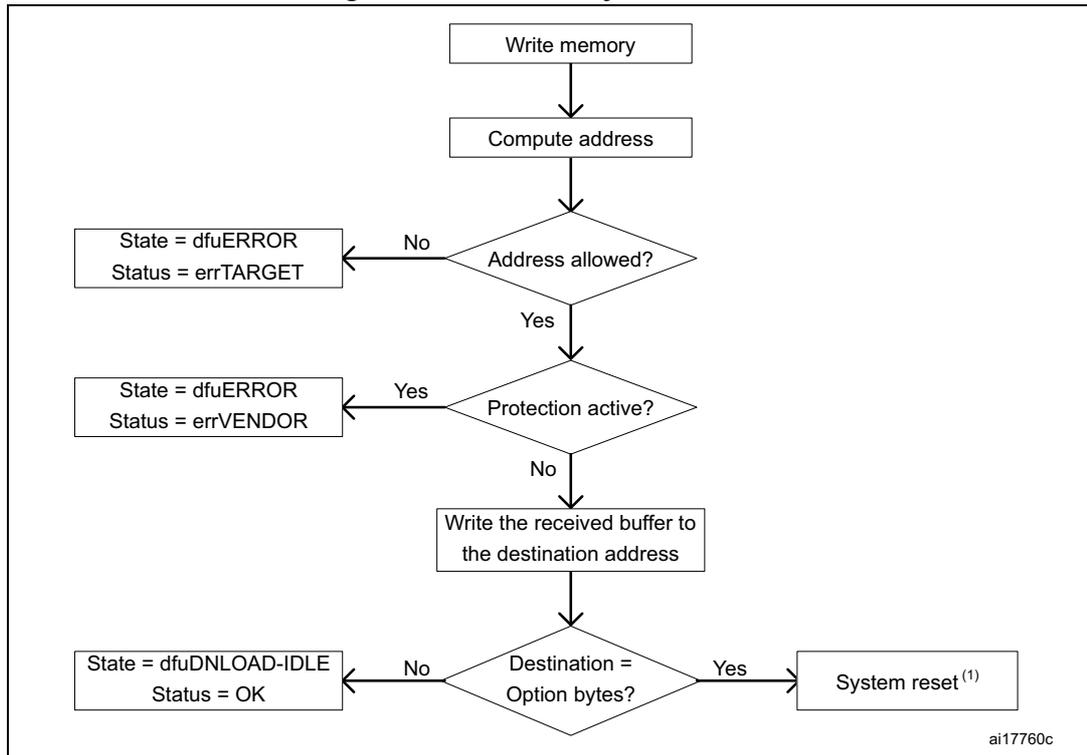
1. System reset is called only for some STM32 BL (STM32F0/F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

5.2 Set Address Pointer command

This command is selected when **wValue** = 0 and the first byte of the buffer sent by the host is 0x21. The buffer length is five bytes (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 must send the DFU_GETSTATUS request.

The Set Address Pointer command is executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, an error 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 (Status = dfuERROR, State = errTARGET).

The allowed locations for address pointer values are valid memory addresses in the internal flash memory, embedded RAM, system memory, and option bytes.

Note: 1 *Refer to Section 4 for more details about the valid memory addresses for the used device.*

2 *The Set Address Pointer command is allowed and executed when the flash memory read protection is enabled or disabled.*

Figure 8. Set Address Pointer command: device side
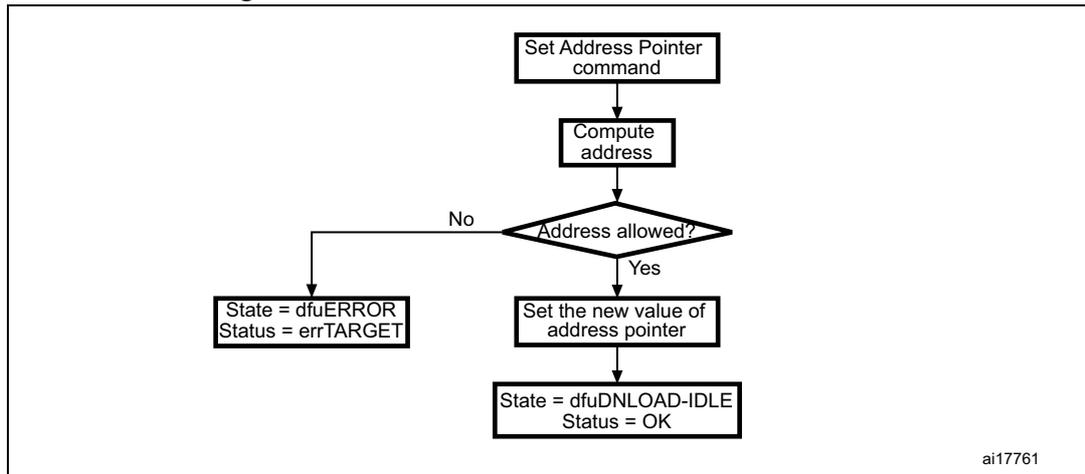
5.3 Erase command

This command is selected when **wValue** = 0 and the first byte of the buffer sent by the host is 0x41. The buffer length can be five bytes (the four remaining bytes are the address bytes, LSB first) for the page erase operation, or only one 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 mass erase it.

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 5:   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

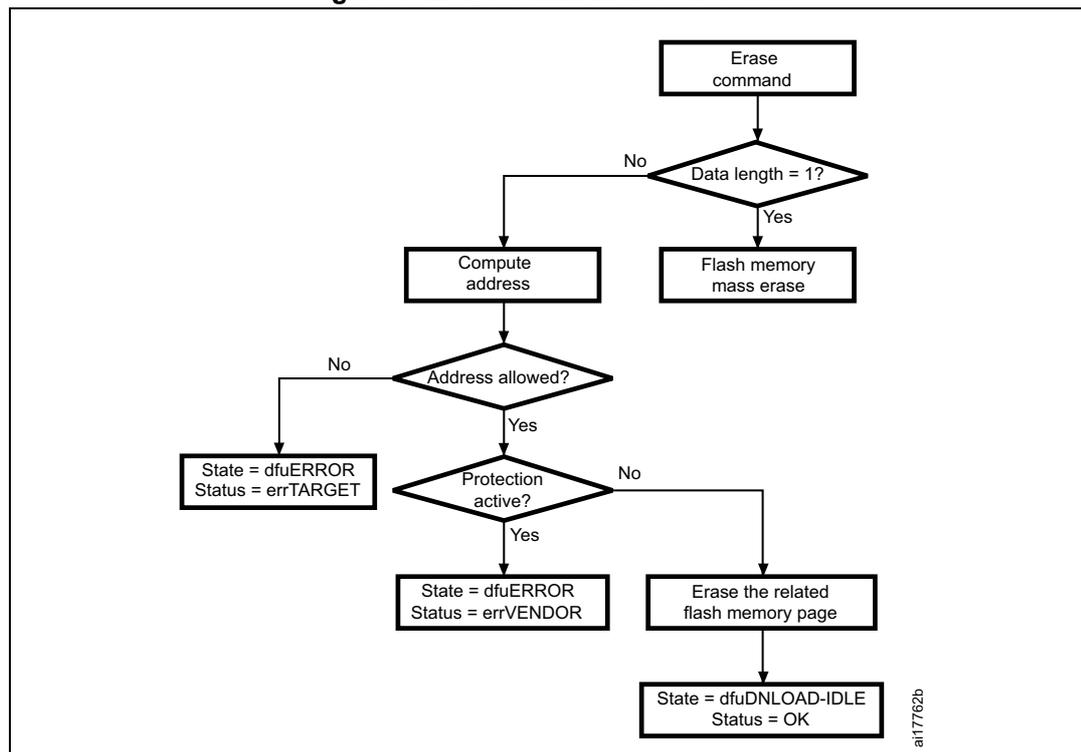After sending an Erase command, the host must send a DFU_GETSTATUS request.

The Erase command is executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, an error 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 (Status = dfuERROR, State = errTARGET). If protection is active, the device returns the status (Status = dfuERROR, State = errVENDOR), and the erase operation is ignored.

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

This command is selected when **wValue** = 0 and the first byte of the buffer sent by the host is 0x92. The buffer length is only one (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

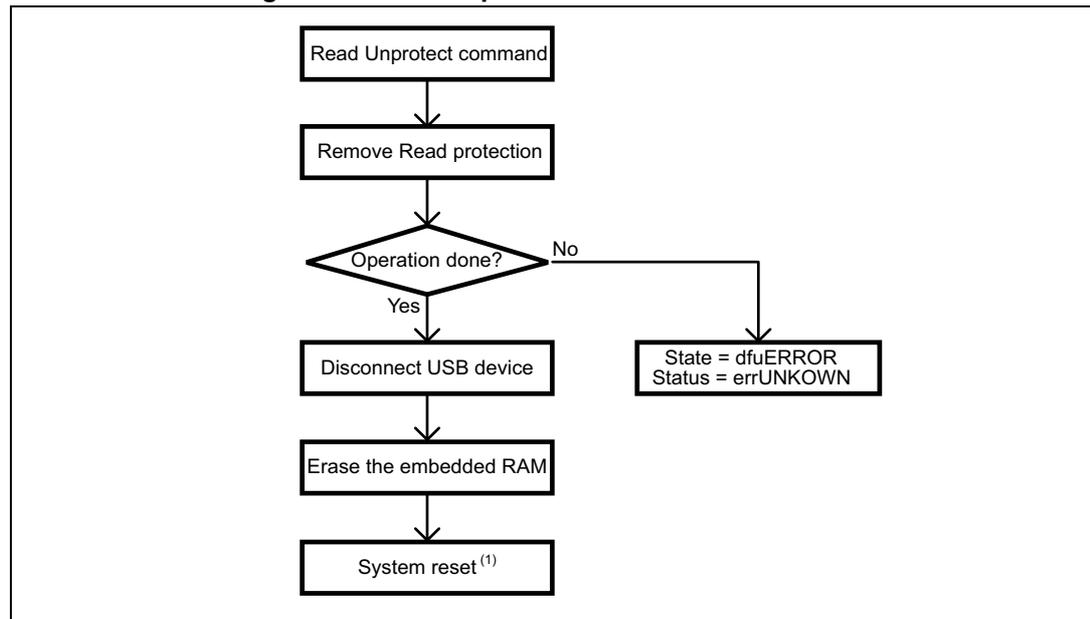After sending a Read Unprotect command, the host must send a DFU_GETSTATUS request.

The command is executed only when a DFU_GETSTATUS request is issued by the host. If the status returned by the device is not dfuDNBUSY, an error occurred. After this operation,

the device removes the read protection and, consequently, both the internal flash memory and the embedded RAM are fully erased.

Immediately after executing this command, the device disconnects itself and executes a system reset. In this case, the device is unable to respond to a second Get Status request, and the host must 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

1. System reset is called only in some STM32F0/F2/F4/F7, STM32L412xx/422xx, STM32L43xxx/44xxx, and STM32L45xxx/46xxx products.

5.5 Leave DFU mode

It is possible to exit DFU mode (and bootloader) and jump to a loaded application (in the internal flash memory 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) to inform the device that it must exit DFU mode. The device acknowledges this request if the current state is dfuDNLOAD-IDLE or dfuIDLE.

The 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, an error has occurred. After this operation, the device performs the following actions:

• Disconnects itself
• Initializes the registers of the peripherals used by the bootloader to their default reset values
• Initializes the user application main stack pointer
• Jumps to the memory location programmed in the received address pointer + 4, which corresponds to the address of the application reset handler (as an example, if the

received address is 0x0800 0000, the bootloader jumps to the memory location programmed at address 0x0800 0004). In general, the host must send the base address where the application to jump to is programmed.

The address pointer must be set (using the Set Address Pointer command) before launching the Leave DFU routine, otherwise the bootloader jumps 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 is stored and used later for the jump.

Note: *If the address pointer points to an address that does not contain executable code, 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 is unable 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 using the USB IP, the application must disable all pending USB interrupts and reset the core before enabling interrupts. Otherwise, a pending interrupt (issued from the bootloader code) can 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
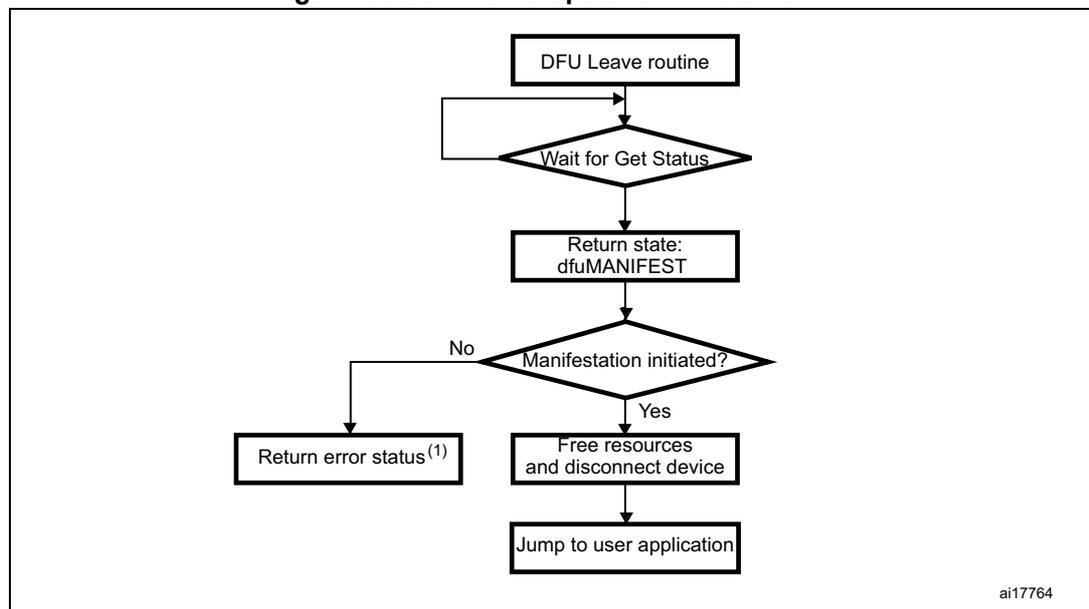
1. This status depends on the error origin and the current status.

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. V2.0 and V2.1 are implemented on different devices. Refer to AN2606 to know the version implemented on the used device. 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.
V3.0	The number of command can vary on devices with the same protocol version v3.0. To know supported commands, use Get command.

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 in *Section 1*. Added *Figure 2: Bootloader for other STM32 devices*. Updated allowed number of bytes when reading from option byte area, and added other memory locations in *Section 4.1: Read memory*. Updated allowed number of bytes when writing to option byte area, and added other memory locations in *Section 5.1: Write memory*. Added bootloader V2.1 in *Section 6*.
12-Feb-2013	3	Changed title for *Figure 1: Bootloader for STM32 connectivity line devices*. Updated *Figure 2: Bootloader for other STM32 devices* including title. Added *Note:* below *Figure 2*. Added *Table 1: Applicable products*.
30-Apr-2014	4	Updated *Table 1: Applicable products* and *Table 5: Bootloader protocol versions*. Updated *Section 1: Bootloader code sequence*. Removed section dedicated to Device-dependent bootloader parameters. Updated *Figure 2: Bootloader for other STM32 devices* and added footnote *3*.
21-Oct-2016	5	Updated *Introduction* and *Table 1: Applicable products*.
16-Mar-2017	6	Updated *Table 1: Applicable products*.
14-Feb-2019	7	Added STM32H7 series, hence updated *Table 1: Applicable products*. Updated *Section 1: Bootloader code sequence*. Minor text edits across the whole document.
21-Feb-2019	8	Added STM32WB series, hence updated *Table 1: Applicable products*.
09-Apr-2019	9	Added STM32G0 and STM32G4 series, hence updated *Table 1: Applicable products*.
23-Sep-2019	10	Added STM32L5 series, hence updated *Table 1: Applicable products*. Updated *Introduction* and *Section 5.3: Erase command*.
26-Nov-2019	11	Updated *Table 1: Applicable products*. Updated *Figure 4: DFU_UPLOAD request: host side* and *Figure 5: Download request: device side*.
02-Jun-2021	12	Updated *Figure 7: Write memory: device side* and *Figure 10: Read Unprotect command: device side* and added footnotes to them. Minor text edits across the whole document.
08-Feb-2022	13	Added STM32U5 series, hence updated *Table 1: Applicable products*.

Table 6. Document revision history (continued)

Date	Revision	Changes
07-Feb-2023	14	Updated *Introduction* and *Section 4.1: Read memory*. Updated *Table 1: Applicable products*, *Table 4: DFU bootloader commands*, and *Table 5: Bootloader protocol versions*. Added footnote in *Section 5: DFU_DNLOAD request commands*. Updated *Figure 3: DFU_UPLOAD request: device side*, *Figure 4: DFU_UPLOAD request: host side*, *Figure 6: Download request: host side*, *Figure 7: Write memory: device side*, and *Figure 9: Erase command: device side*. Minor text edits across the whole document.
23-Feb-2024	15	Added STM32U0 series, hence updated *Table 1: Applicable products*. Updated *Table 3: Summary of DFU class-specific requests*. Minor text edits across the whole document.
18-Sep-2024	16	Added STM32C0 series, hence updated *Table 1: Applicable products*.
07-Feb-2025	17	Added STM32U3 and STM32WBA series, hence updated *Table 1: Applicable products*.
16-Feb-2026	18	Added STM32C5 series. Updated *Table 1: Applicable products*. Updated document title.

IMPORTANT NOTICE – READ CAREFULLY