AN4221
Application note

How to use I2C protocol in bootloader on STM32 MCUs

Introduction

This application note describes the I2C protocol used in the STM32 microcontroller bootloader, detailing each supported command.

This document applies to the STM32 products embedding bootloader versions V5.x, V6.x,V7.x, V8.x, V9.x, V10.x, V11.x, V13.x, and V14.x, as specified in the application note AN2606 "*STM32 microcontroller system memory boot mode*", available on *www.st.com*. These products are listed in *Table 1*, and are referred to as STM32 throughout the document.

For more information about the I2C hardware resources and requirements for your device bootloader, refer to the already mentioned AN2606.

Table 1. Applicable products

Type	Part numbers or series
Microcontrollers	STM32C0 series STM32F0 series STM32F3 series STM32F4 series STM32F7 series STM32G0 series STM32G4 series STM32G5 series STM32H5 series STM32H7 series STM32L0 series STM32L4 series STM32L5 series STM32U0 series STM32U3 series STM32U5 series STM32WB series STM32WBA series

Contents

1 I2C bootloader code sequence 5

2 Bootloader command set 6

2.1 Get command 8
2.2 Get Version command 12
2.3 Get ID command 13
2.4 Read Memory command 15
2.5 Go command 18
2.6 Write Memory command 21
2.7 Erase Memory command 24
2.8 Write Protect command 27
2.9 Write Unprotect command 30
2.10 Readout Protect command 31
2.11 Readout Unprotect command 33
2.12 No-Stretch Write Memory command 35
2.13 No-Stretch Erase Memory command 38
2.14 Special command 41
2.15 Extended Special command 44
2.16 No-Stretch Write Protect command 46
2.17 No-Stretch Write Unprotect command 49
2.18 No-Stretch Readout Protect command 51
2.19 No-Stretch Readout Unprotect command 53
2.20 No-Stretch GetCheckSum command 55

3 Bootloader protocol version evolution 58

Revision history 59

List of tables

Table 1. Applicable products 1
Table 2. I2C bootloader commands 6
Table 3. Bootloader protocol versions 58
Table 4. Document revision history 59

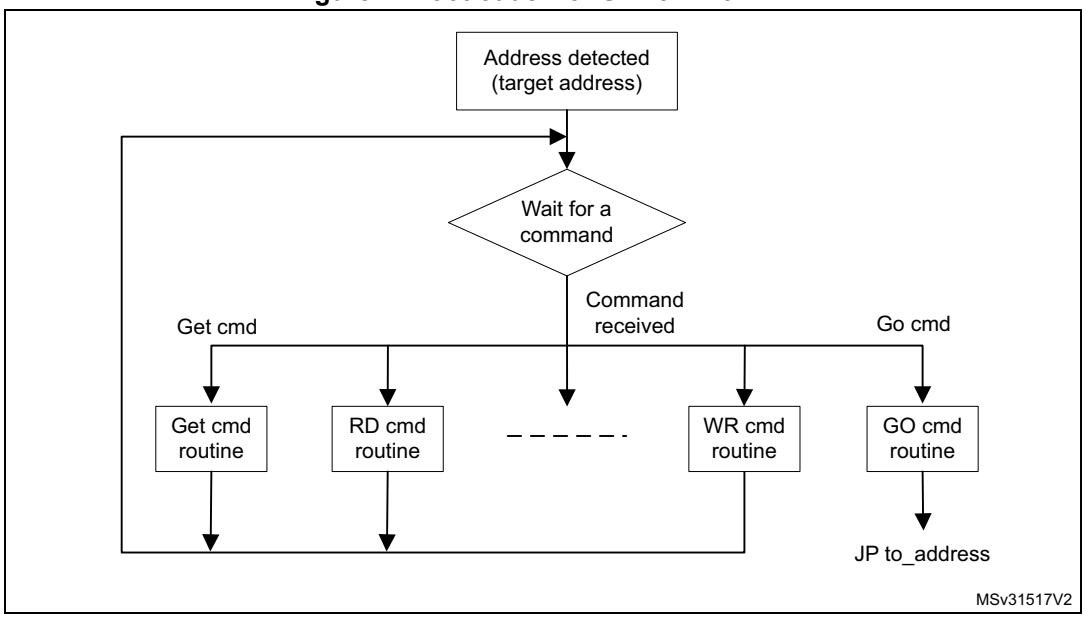
List of figures

Figure 1. Bootloader for STM32 with I2C . . . 5
Figure 2. Get command: host side . . . 9
Figure 3. Get command: device side . . . 9
Figure 4. Get Version: host side . . . 12
Figure 5. Get Version: device side . . . 13
Figure 6. Get ID command: host side . . . 14
Figure 7. Get ID command: device side . . . 14
Figure 8. Read Memory command: host side . . . 16
Figure 9. Read Memory command: device side . . . 17
Figure 10. Go command: host side . . . 19
Figure 11. Go command: device side . . . 20
Figure 12. Write Memory command: host side . . . 22
Figure 13. Write Memory command: device side . . . 23
Figure 14. Erase Memory command: host side . . . 25
Figure 15. Erase Memory command: device side . . . 26
Figure 16. Write Protect command: host side . . . 28
Figure 17. Write Protect command: device side . . . 29
Figure 18. Write Unprotect command: host side . . . 30
Figure 19. Write Unprotect command: device side . . . 31
Figure 20. Readout Protect command: host side . . . 32
Figure 21. Readout Protect command: device side . . . 32
Figure 22. Readout Unprotect command: host side . . . 33
Figure 23. Readout Unprotect command: device side . . . 34
Figure 24. No-Stretch Write Memory command: host side . . . 36
Figure 25. No-Stretch Write Memory command: device side . . . 37
Figure 26. No-Stretch Erase Memory command: host side . . . 39
Figure 27. No-Stretch Erase Memory command: device side . . . 40
Figure 28. Special command: host side . . . 41
Figure 29. Special command: device side . . . 42
Figure 30. Extended Special command: host side . . . 44
Figure 31. Extended Special command: device side . . . 45
Figure 32. No-Stretch Write Protect command: host side . . . 47
Figure 33. No-Stretch Write Protect command: device side . . . 48
Figure 34. No-Stretch Write Unprotect command: host side . . . 49
Figure 35. No-Stretch Write Unprotect command: device side . . . 50
Figure 36. No-Stretch Readout Protect command: host side . . . 51
Figure 37. No-Stretch Readout Protect command: device side . . . 52
Figure 38. No-Stretch Readout Unprotect command: host side . . . 53
Figure 39. No-Stretch Readout Unprotect command: device side . . . 54
Figure 40. No-Stretch GetCheckSum command: host side . . . 56
Figure 41. No-Stretch GetCheckSum command: device side . . . 57

1 I2C bootloader code sequence

The I2C bootloader code sequence for STM32 microcontrollers, based on Arm®(a) cores, is sketched in *Figure 1*.

Figure 1. Bootloader for STM32 with I2C



Note: The I2C target address for each product bootloader is specified in AN2606.

Once the system memory boot mode has been entered and the microcontroller has been configured (for more details, refer to the application note for the system memory boot mode of the device you are using), the bootloader code begins to scan the I2C_SDA line pin, waiting to detect its own address on the bus. Once detected, the I2C bootloader firmware begins receiving host commands.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) 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.

2 Bootloader command set

The supported commands are listed in *Table 2*.

“No-Stretch” commands are supported starting from the V1.1 protocol version: they enable a better management of commands when the host must wait for a significant time before an operation is completed by the bootloader. It is recommended to use these commands instead of equivalent regular commands whenever possible.

Table 2. I2C bootloader commands

Commands(1)	Command code	Command description
Get(2)	0x00	Gets the version and the allowed commands supported by the current version of the protocol.
Get Version(2)	0x01	Gets the protocol version.
Get ID(2)	0x02	Gets the chip ID.
Read Memory	0x11	Reads up to 256 bytes of memory, starting from an address specified by the application.
Go(3)	0x21	Jumps to user application code located in the internal flash memory.
Write Memory(3)	0x31	Writes up to 256 bytes to the memory, starting from an address specified by the application.
No-Stretch Write Memory(3)(4)	0x32	Writes up to 256 bytes to the memory, starting from an address specified by the application and returns busy state while operation is ongoing.
Erase	0x44	Erases from one to all flash memory pages or sectors using two-byte addressing mode.
No-Stretch Erase(3)(4)	0x45	Erases from one to all flash memory pages or sectors using two-byte addressing mode and returns busy state while operation is ongoing.
Special	0x50	Generic command to add new features depending on the product constraints, without adding a new command for every feature.
Extended Special	0x51	Generic command that allows the user to send more data compared to the Special command.
Write Protect	0x63	Enables write protection for some sectors.
No-Stretch Write Protect(4)	0x64	Enables write protection for some sectors and returns busy state while operation is ongoing.
Write Unprotect	0x73	Disables write protection for all flash memory sectors.
No-Stretch Write Unprotect(4)	0x74	Disables write protection for all flash memory sectors and returns busy state while operation is ongoing.
Readout Protect	0x82	Enables read protection.
No-Stretch Readout Protect(4)	0x83	Enables read protection and returns busy state while operation is ongoing.
Readout Unprotect(2)	0x92	Disables read protection.

Table 2. I2C bootloader commands (continued)

Commands[1]	Command code	Command description
No-Stretch Readout Unprotect[2][4]	0x93	Disables read protection and returns busy state while operation is ongoing.
No-Stretch Get Memory Checksum[2]	0xA1	Gets CRC checksum value for a memory portion, based on its offset and length.

1. If a denied command is received, or if an error occurs during the command execution, the bootloader sends an NACK byte and goes back to command checking.
2. Protection: when the protection is active, only this limited subset of commands is available. All other commands are NACK-ed, and have no effect on the device. The protection depends upon the product family:
   - for the STM32H5 series: TrustZone® (TZEN) = 0, Product state > Provisioning and Hide Protection Level (HDPL) = 3
   - for all the other products listed in *Table 1*: Read protection set.
3. Refer to the product datasheet and to AN2606 to know the memory spaces valid for these commands.
4. No-Stretch commands are available only with I2C protocol V1.1 and above.

No-Stretch commands

No-Stretch commands make it possible to execute Write, Erase, Write Protect, Write Unprotect, Read Protect, and Read Unprotect operations without stretching the I2C line while the bootloader is performing the operation. It is possible to communicate with other devices on the bus while the bootloader performs operations that require waiting time.

The difference between these commands and the standard ones is at the end of the command: when the host requests ACK/NACK at the end of the command, instead of stretching the I2C line, the bootloader answers with a third state, which is Busy (0x76). When the host receives the Busy state, it polls again on the state, and reads one byte until it receives an ACK or an NACK response.

Communication safety

All communications from the programming host to the device are verified by checksum. Received blocks of data bytes are XOR-ed. A byte containing the computed XOR of all previous bytes is added to the end of each communication (checksum byte). By XOR-ing all received bytes, data plus checksum, the result at the end of the packet must be 0x00.

For each command, the host sends a byte and its complement (XOR = 0x00).

Each packet is accepted (ACK answer) or discarded (NACK answer):

- ACK = 0x79
- NACK = 0x1F

With No-Stretch commands, Busy state is sent instead of ACK or NACK when an operation is ongoing:

- BUSY= 0x76

Note: The host frame can be one of the following:

- *Send Command frame: the host initiates communication as controller transmitter, and sends two bytes to the device: command code + XOR.*
- *Wait for ACK/NACK frame: the host initiates an I2C communication as controller receiver, and receives one byte from the device: ACK or NACK or BUSY.*
- *Receive Data frame: the host initiates an I2C communication as controller receiver, and receives the response from the device. The number of bytes received depends on the command.*
- *Send Data frame: the host initiates an I2C communication as controller transmitter, and sends the needed bytes to the device. The number of bytes transmitted depends on the command.*

Caution: For Write, Erase. and Read Unprotect commands, the host must respect the timings (such as page write, sector erase) specified in product datasheets. As an example, when launching an Erase command, the host must wait (before the last ACK of the command) for a duration equivalent to the maximum sector/page erase time specified in datasheet (or at least the typical sector/page erase time).

Caution: For I2C communication, a timeout mechanism is implemented, it must be respected for bootloader commands to be executed correctly. This timeout is implemented between two I2C frames in the same command. For example, for a Write memory command, a timeout is inserted between the command-sending frame and the address memory-sending frame. Also, the same timeout period is inserted between two successive instances of data reception or transmission in the same I2C frame. If the timeout period has elapsed, a system reset is generated to avoid a bootloader crash. Refer to the section dedicated to I2C connection timing of AN2606 to get the I2C timeout value for each STM32 product.

2.1 Get command

This command allows the user to get the version of the protocol and the supported commands. When the bootloader receives the command, it transmits the protocol version and the supported command codes to the host, as described in *Figure 2*.

Figure 2. Get command: host side

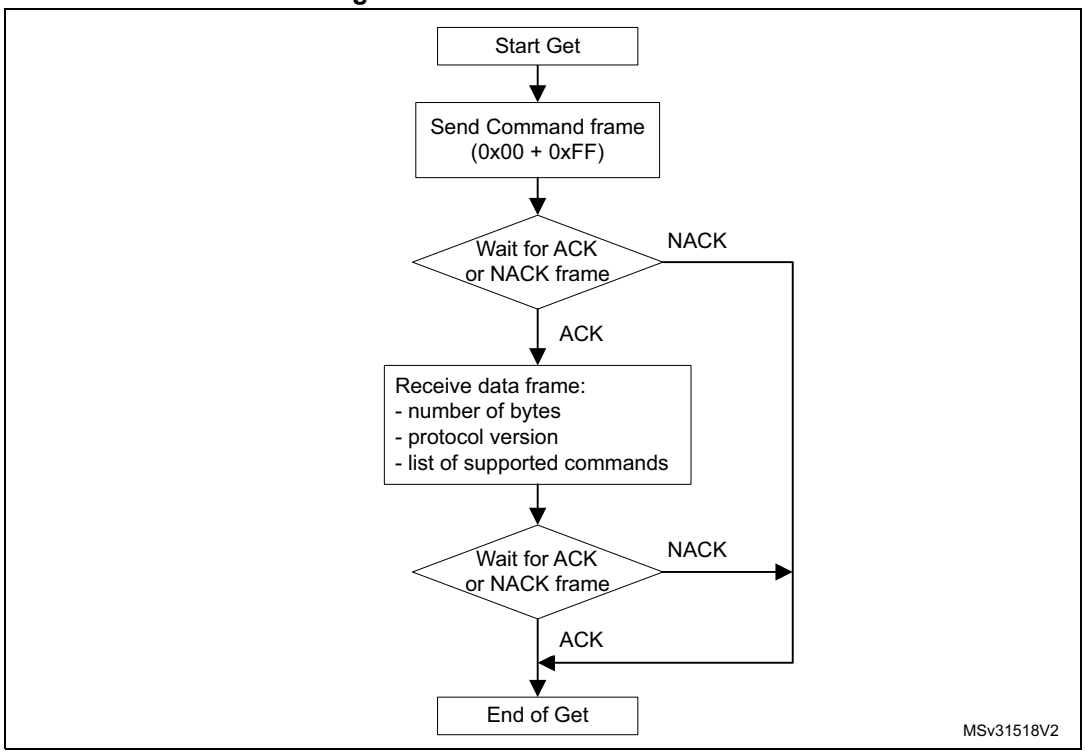
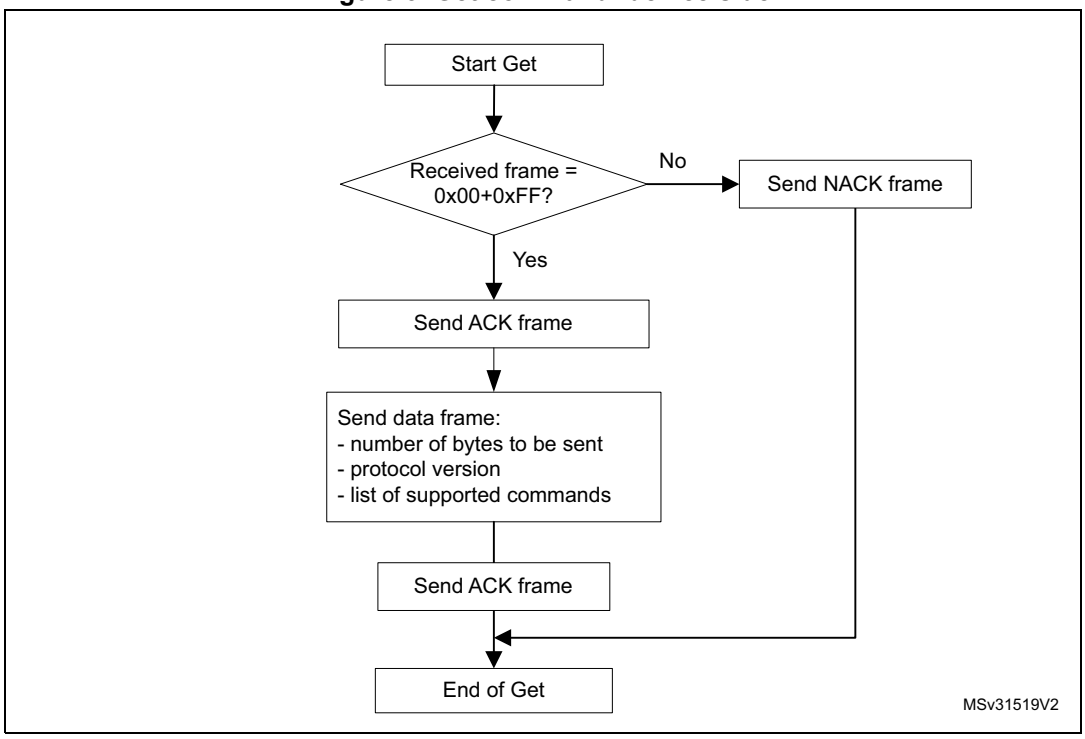


Figure 3. Get command: device side



The STM32 sends the bytes as follows:

- For I2C protocol V1.0:
  - Byte 1: ACK
  - Byte 2: N = 11 = Number of bytes to follow - 1, except current and ACKs
  - Byte 3: Bootloader version 0x10 = Version 1.0
  - Byte 4: 0x00 - Get command
  - Byte 5: 0x01 - Get Version command
  - Byte 6: 0x02 - Get ID command
  - Byte 7: 0x11 - Read Memory command
  - Byte 8: 0x21 - Go command
  - Byte 9: 0x31 - Write Memory command
  - Byte 10: 0x44 - Erase command
  - Byte 11: 0x63 - Write Protect command
  - Byte 12: 0x73 - Write Unprotect command
  - Byte 13: 0x82 - Readout Protect command
  - Byte 14: 0x92 - Readout Unprotect command
  - Byte 15: ACK
- For I2C protocol V1.1:
  - Byte 1: ACK
  - Byte 2: N = 17 = Number of bytes to follow - 1, except current and ACKs
  - Byte 3: Bootloader version 0x11 = Version 1.1
  - Byte 4: 0x00 - Get command
  - Byte 5: 0x01 - Get Version command
  - Byte 6: 0x02 - Get ID command
  - Byte 7: 0x11 - Read Memory command
  - Byte 8: 0x21 - Go command
  - Byte 9: 0x31 - Write Memory command
  - Byte 10: 0x44 - Erase command
  - Byte 11: 0x63 - Write Protect command
  - Byte 12: 0x73 - Write Unprotect command
  - Byte 13: 0x82 - Readout Protect command
  - Byte 14: 0x92 - Readout Unprotect command
  - Byte 15: 0x32 - No-Stretch Write Memory command
  - Byte 16: 0x45 - No-Stretch Erase command
  - Byte 17: 0x64 - No-Stretch Write Protect command
  - Byte 18: 0x74 - No-Stretch Write Unprotect command
  - Byte 19: 0x83 - No-Stretch Readout Protect command
  - Byte 20: 0x93 - No-Stretch Readout Unprotect command
  - Byte 21: ACK
- For I2C protocol V1.2:
  - Byte 1: ACK
  - Byte 2: N = 18 = Number of bytes to follow - 1, except current and ACKs

- Byte 3: Bootloader version 0x12 = Version 1.2
- Byte 4: 0x00 - Get command
- Byte 5: 0x01 - Get Version command
- Byte 6: 0x02 - Get ID command
- Byte 7: 0x11 - Read Memory command
- Byte 8: 0x21 - Go command
- Byte 9: 0x31 - Write Memory command
- Byte 10: 0x44 - Erase command
- Byte 11: 0x63 - Write Protect command
- Byte 12: 0x73 - Write Unprotect command
- Byte 13: 0x82 - Readout Protect command
- Byte 14: 0x92 - Readout Unprotect command
- Byte 15: 0x32 - No-Stretch Write Memory command
- Byte 16: 0x45 - No-Stretch Erase command
- Byte 17: 0x64 - No-Stretch Write Protect command
- Byte 18: 0x74 - No-Stretch Write Unprotect command
- Byte 19: 0x83 - No-Stretch Readout Protect command
- Byte 20: 0x93 - No-Stretch Readout Unprotect command
- Byte 21: 0xA1 - No-Stretch Get Memory Checksum command
- Byte 22: ACK

Some commands depend upon the HW features. Beginning from the SPI BL version V2.0, the number of commands is no more fixed, and can change from product to product.

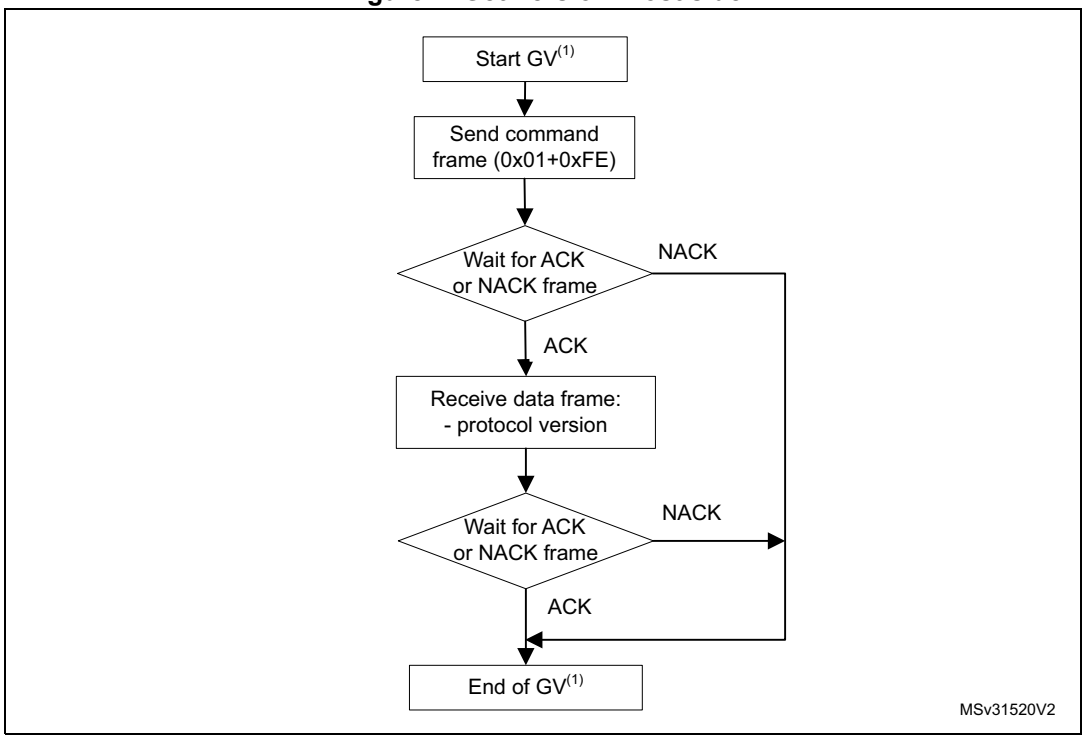
As an example, for the STM32H5 series there is no RDP HW feature. The Get command is:

- Byte 1: ACK
- Byte 2: N = 15 = the number of bytes to follow – 1, except current and ACKs
- Byte 3: protocol version (0x20 = version 2.0)
- Byte 4: 0x00 - Get command
- Byte 5: 0x01 - Get Version command)
- Byte 6: 0x02 - Get ID command
- Byte 7: 0x11 - Read Memory command
- Byte 8: 0x21 - Go command
- Byte 9: 0x31 - Write Memory command
- Byte 10: 0x44 - Erase command
- Byte 11: 0x50 - Special command
- Byte 12: 0x63 - Write Protect command
- Byte 13: 0x73 - Write Unprotect command
- Byte 14: 0x32 - No-Stretch Write Memory command
- Byte 15: 0x45 - No-Stretch Erase command
- Byte 16: 0x64 - No-Stretch Write Protect command
- Byte 17: 0x74 - No-Stretch Write Unprotect command
- Byte 18: 0xA1 - No-Stretch Get Memory Checksum command
- Byte 19: ACK

2.2 Get Version command

This command is used to get the I2C protocol version. When the bootloader receives the command, it transmits the protocol version to the host.

Figure 4. Get Version: host side

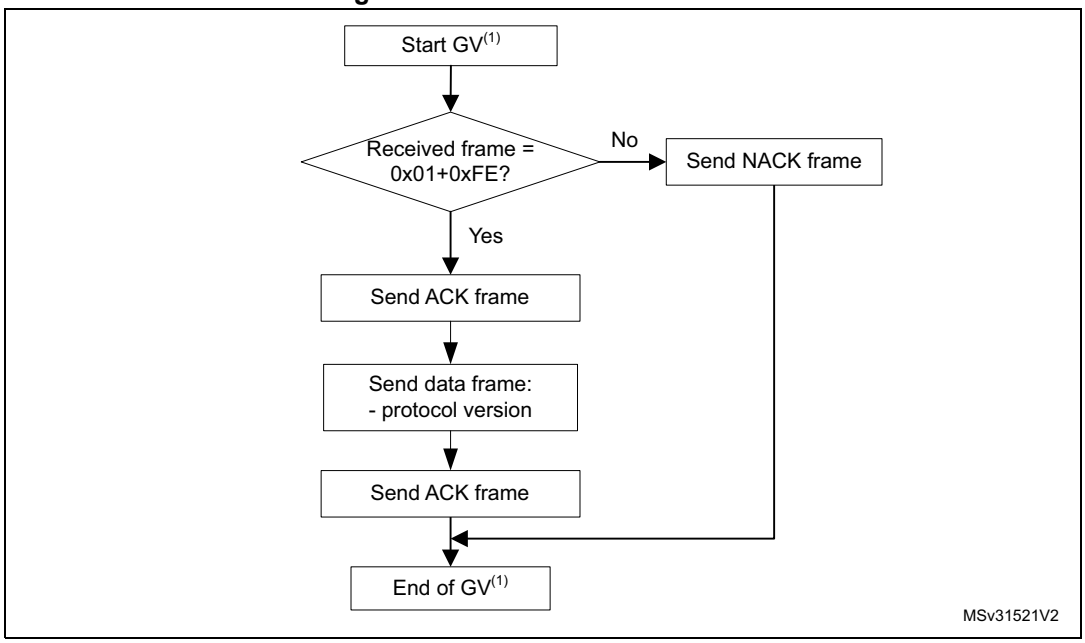


1. GV = Get Version.

The STM32 sends the bytes as follows:

- Byte 1: ACK
- Byte 2: protocol version (0 < Version ≤ 255), for example, 0x10 = Version 1.0
- Byte 3: ACK

Figure 5. Get Version: device side



1. GV = Get Version

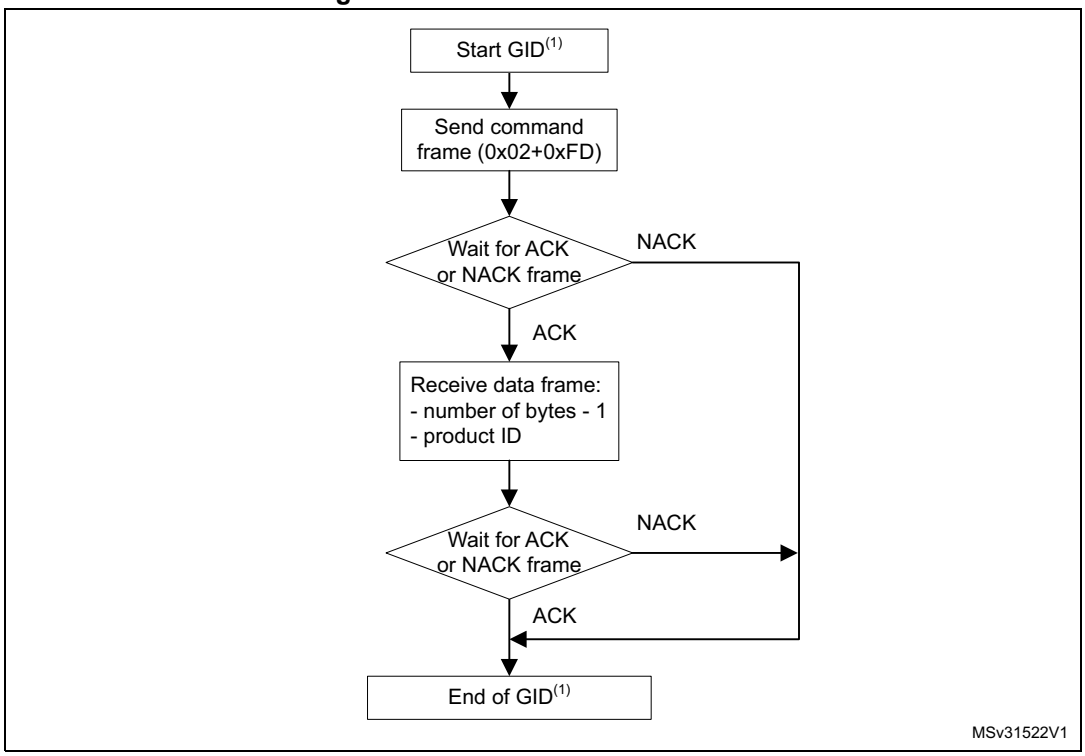
2.3 Get ID command

This command is used to get the version of the chip ID (identification). When the bootloader receives the command, it transmits the product ID to the host.

The STM32 device sends the bytes as follows:

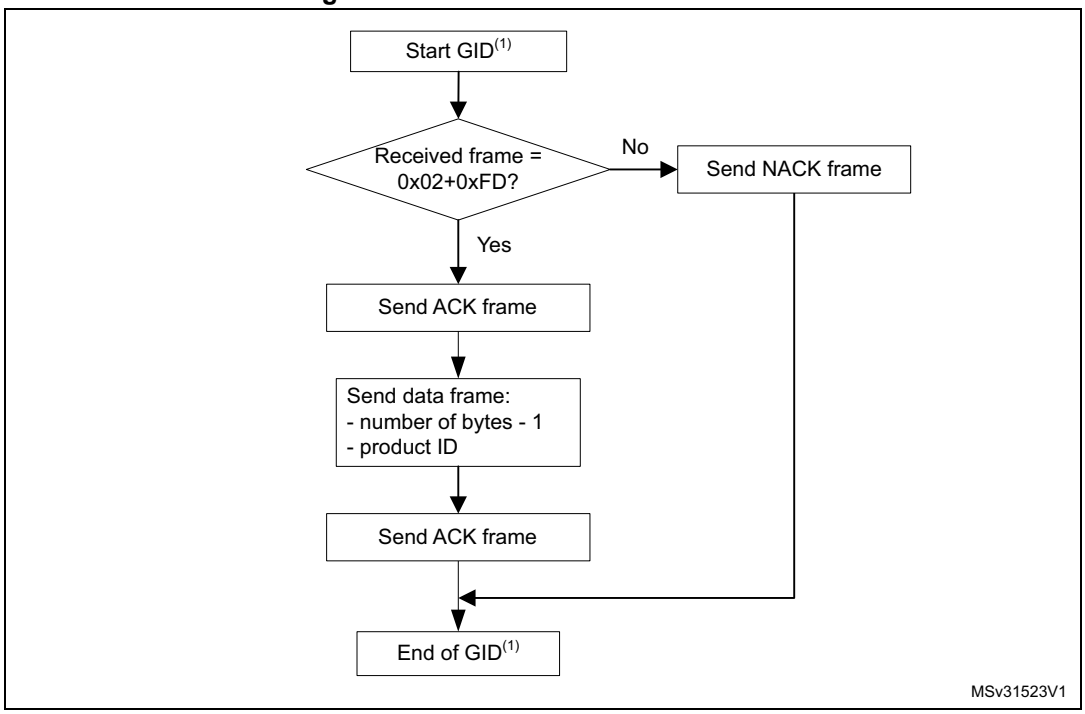
- Byte 1: ACK
- Byte 2: N = number of bytes - 1 (for STM32, N = 1), except for current byte and ACKs
- Bytes 3-4: PID (product ID)
  - Byte 3 = MSB
  - Byte 4 = LSB
- Byte 5: ACK

Figure 6. Get ID command: host side



1. GID = Get ID.

Figure 7. Get ID command: device side



1. GID = Get ID.

2.4 Read Memory command

This command is used to read data from any valid memory address.

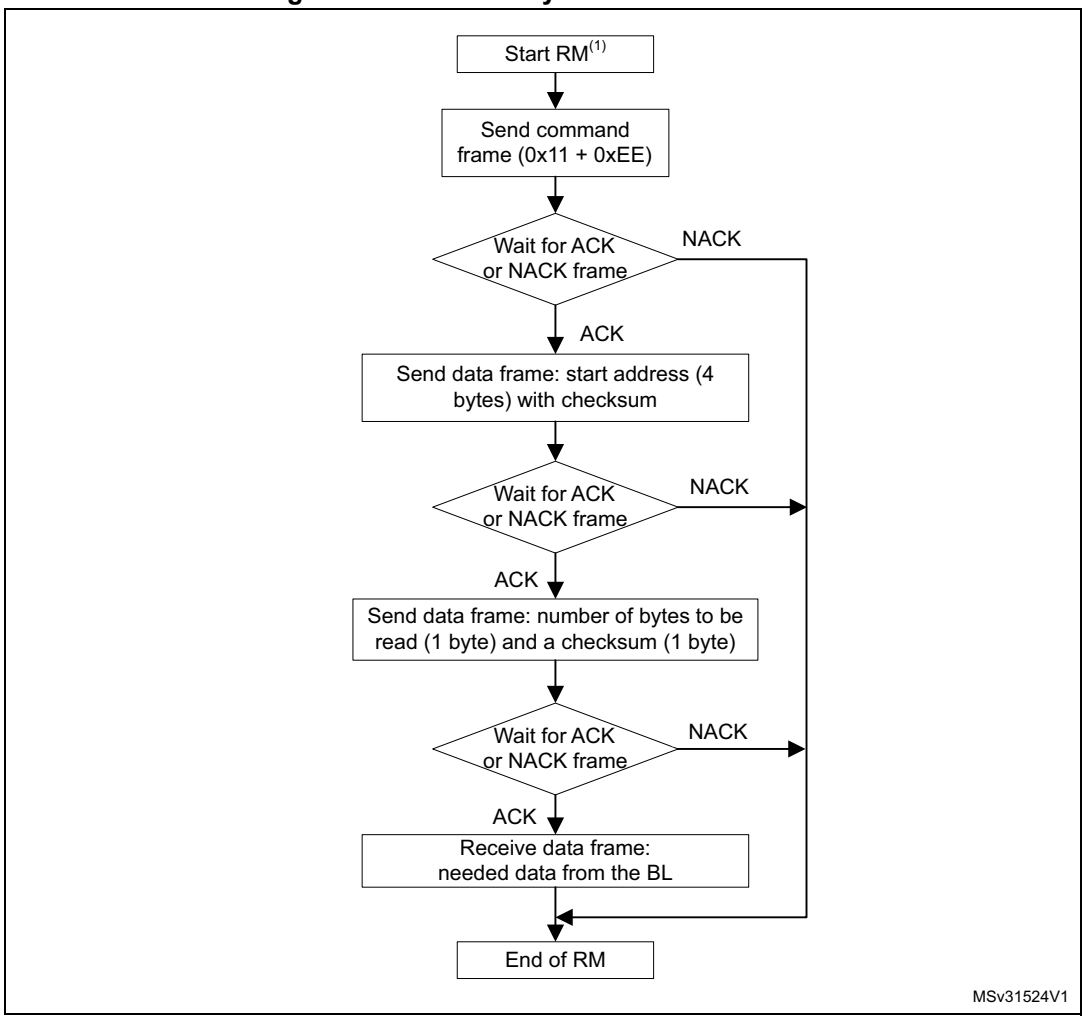
When the bootloader receives the command, it transmits the ACK byte to the application. The bootloader then waits for a 4-byte address (byte 1 is the address MSB, byte 4 is the LSB) and a checksum byte, then it checks the received address. If the address is valid and the checksum is correct, the bootloader transmits an ACK byte; otherwise, it transmits an NACK byte and aborts the command.

If the address is valid and the checksum is correct, the bootloader waits for the number of bytes to be transmitted (N bytes), and for its complemented byte (checksum). If the checksum is correct, the bootloader transmits the needed data to the application, starting from the received address. If the checksum is not correct, it sends an NACK before aborting the command.

The host sends bytes to the STM32 as follows:

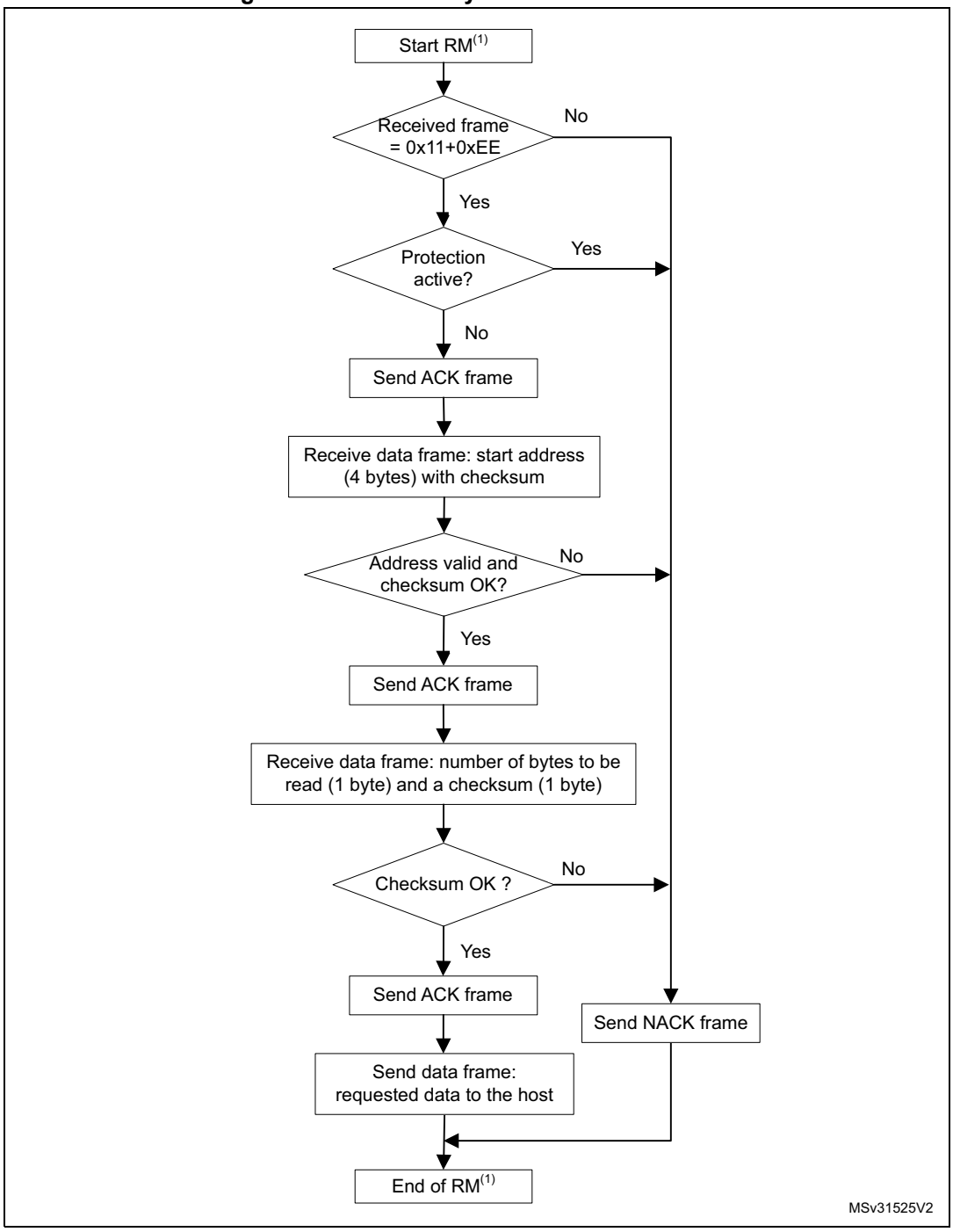
1. Bytes 1-2: 0x11 + 0xEE
2. Wait for ACK
3. Bytes 3-6: start address (byte 3: MSB, byte 6: LSB)
4. Byte 7: checksum: XOR (byte 3, byte 4, byte 5, byte 6)
5. Wait for ACK
6. Byte 8: number of bytes to be read - 1 ($0 < N \leq 255$)
7. Byte 9: checksum: XOR byte 8 (complement of byte 8)

Figure 8. Read Memory command: host side



1. RM = Read Memory.

Figure 9. Read Memory command: device side



1. RM = Read Memory.

2.5 Go command

The Go command is used to execute the downloaded code or any other code, by branching to an address specified by the application. When the bootloader receives the Go command, it transmits the ACK byte to the application. The bootloader then waits for a 4-byte address (byte 1 is the address MSB, byte 4 is the LSB) and a checksum byte, then checks the received address. If the address is valid and the checksum is correct, the bootloader transmits an ACK byte; otherwise, it transmits an NACK byte and aborts the command.

When the address is valid and the checksum is correct, the bootloader firmware performs the following operations:

1. Initializes the registers of the peripherals used by the bootloader to their default reset values
2. Initializes the user application main stack pointer
3. Jumps to the memory location programmed in the received address + 4 (corresponds to the address of the application reset handler). For example, if the received address is 0x08000000, the bootloader jumps to the memory location programmed at address 0x08000004.

In general, the host sends the base address where the application to jump to is programmed.

Note: Jumping to the application only works if the user application correctly sets the vector table to point to the application address.

The host sends bytes to the STM32 as follows:

1. Byte 1: 0x21
2. Byte 2: 0xDE
3. Wait for ACK
4. Byte 3 to byte 6: start address
   - Byte 3: MSB
   - Byte 6: LSB
5. Byte 7: checksum: XOR (byte 3, byte 4, byte 5, byte 6)
6. Wait for ACK

Figure 10. Go command: host side

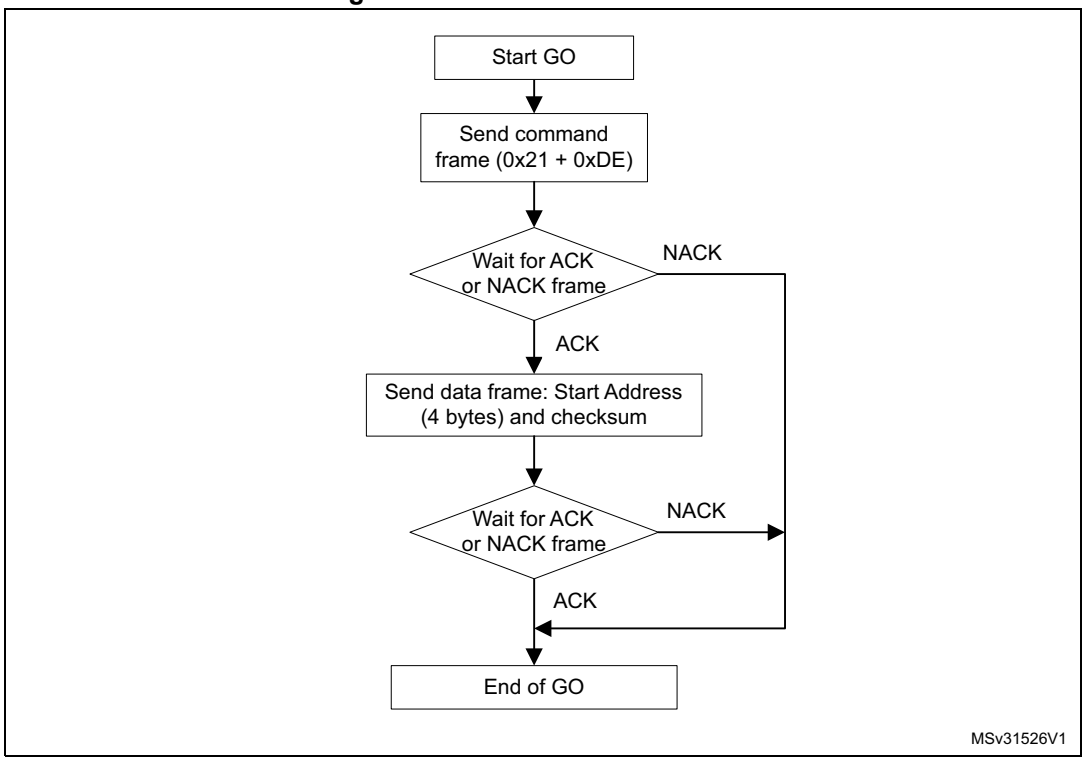
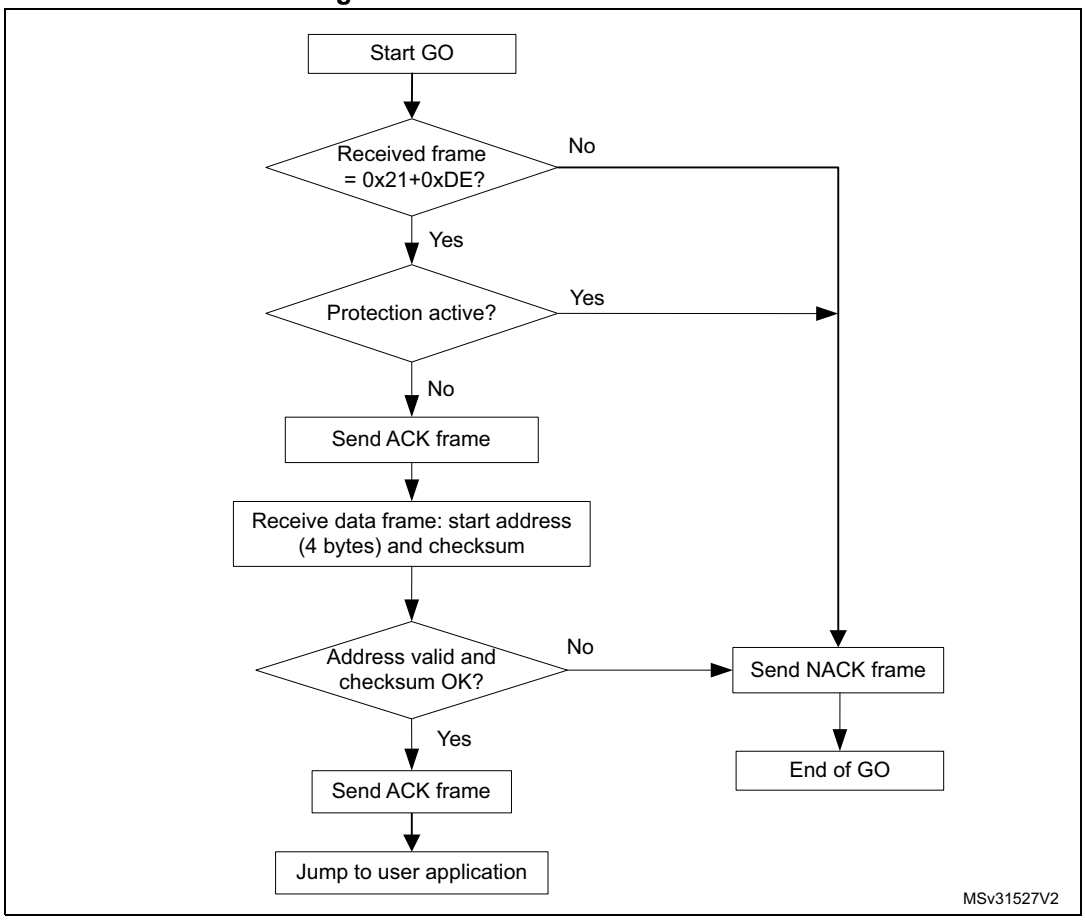


Figure 11. Go command: device side



2.6 Write Memory command

This command is used to write data to any valid memory address (see *Note:* below) of RAM, flash memory, or the option byte area.

When the bootloader receives the command, it transmits the ACK byte to the application. The bootloader then waits for a 4-byte address (byte 1 is the address MSB, and byte 4 is the LSB) and a checksum byte, and then checks the received address.

If the received address is valid and the checksum is correct, the bootloader transmits an ACK byte; otherwise, it transmits an NACK byte and aborts the command. When the address is valid and the checksum is correct, the bootloader:

1. Gets a byte, N, which contains the number of data bytes to be received
2. Receives the user data ((N + 1) bytes) and the checksum (XOR of N and of all data bytes)[a]
3. Programs the user data to memory, starting from the received address.

At the end of the command, if the write operation is successful, the bootloader transmits the ACK byte; otherwise, it transmits an NACK byte to the application and aborts the command.

If the Write Memory command is issued to the option byte area, all bytes are erased, new values are written, and the bytes are reloaded.

The maximum length of the block to be written depends upon the product, and the address received from the host must be the start address of the option byte area. For more information about option bytes, refer to the STM32 product reference manual.

Note: The maximum length of the block to be written to RAM or flash memory is 256 bytes.

When writing to the RAM, do not overlap the first RAM used by the bootloader firmware.

No error is returned when performing write operations to write-protected sectors.

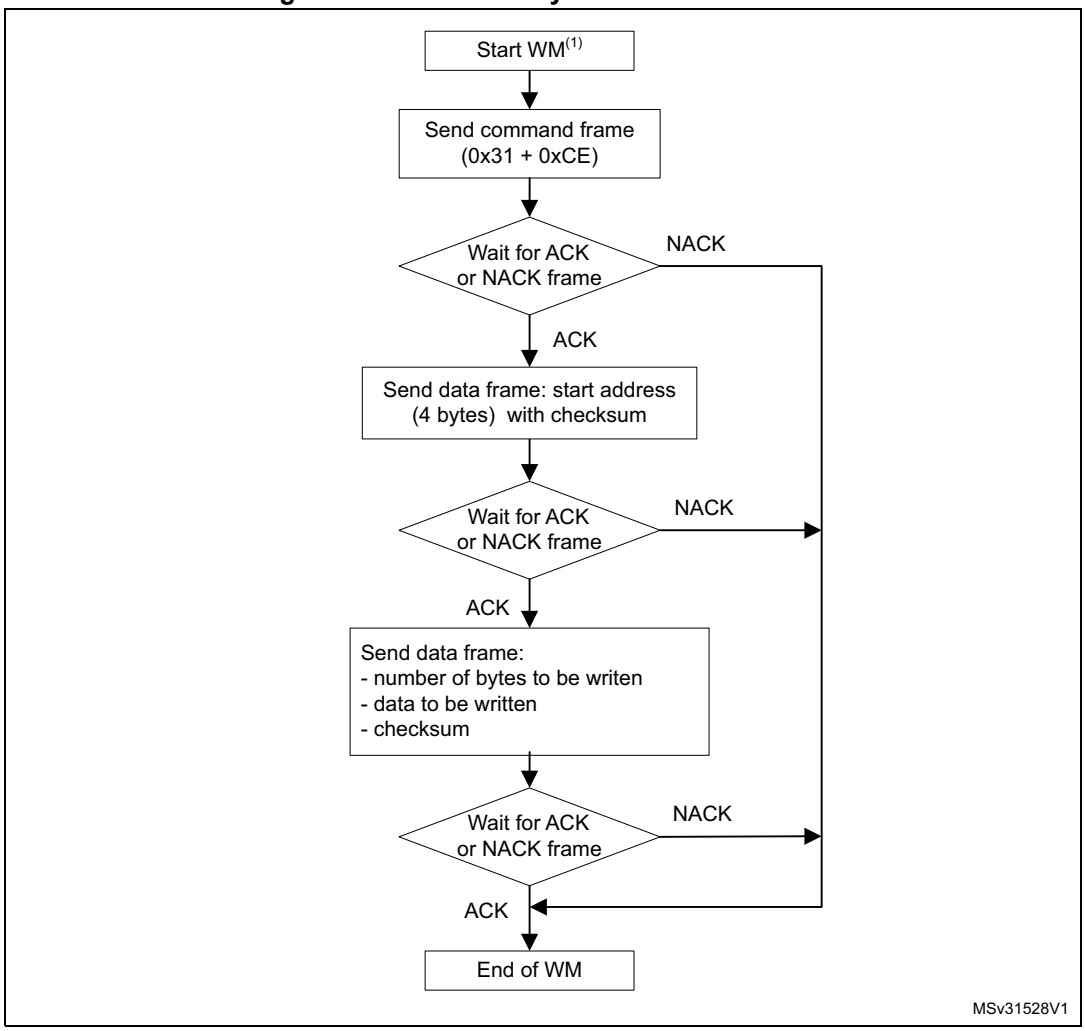
On some products the bootloader generates a system reset after the option bytes are reloaded, as indicated in footnote 2.

The host sends the bytes to the STM32 as follows:

1. Byte 1: 0x31
2. Byte 2: 0xCE
3. Wait for ACK
4. Byte 3 to byte 6: start address
   - Byte 3: MSB
   - Byte 6: LSB
5. Byte 7: checksum: XOR (byte 3, byte 4, byte 5, byte 6)
6. Wait for ACK
7. Byte 8: number of bytes to be received (0 < N ≤ 255)
8. N +1 data bytes: (max 256 bytes)[a]
9. checksum byte: XOR (N, N+1 data bytes)
10. Wait for ACK

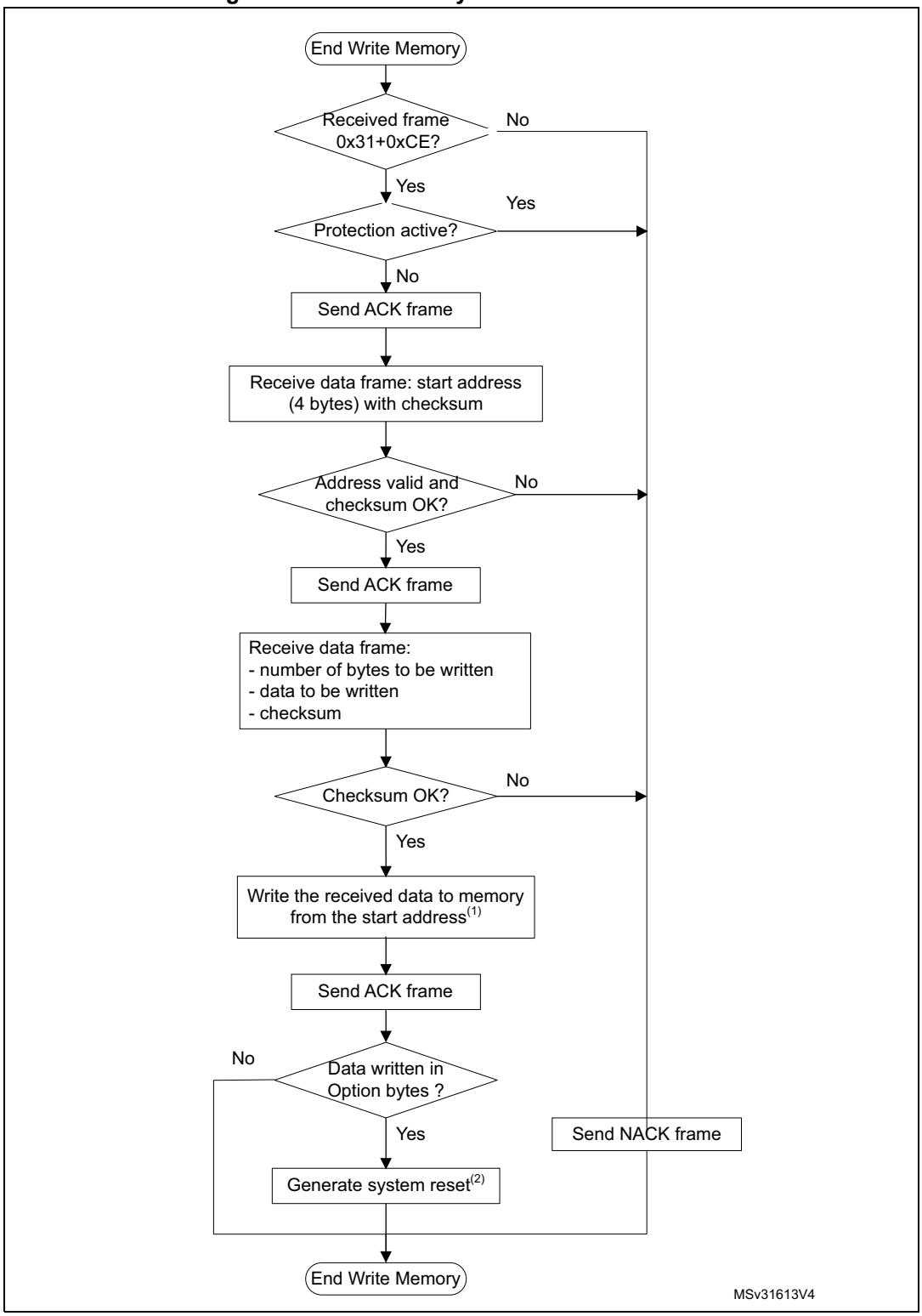
a. Every word (U32) must be transmitted in LSB to MSB order (byte 1 is the LSB, and byte 4 is the MSB).

Figure 12. Write Memory command: host side



1. WM = Write Memory.

Figure 13. Write Memory command: device side



1. If the address to write corresponds to an option byte address, an option byte reload is triggered after writing the data aligned with every MCU flash interface specification.
2. A system reset is called only for some STM32 BL (STM32F0/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

2.7 Erase Memory command

This command allows the host to erase flash memory pages or sectors using a two-byte addressing mode.

When the bootloader receives the command, it transmits the ACK byte to the host. The bootloader receives two bytes (number of pages or sectors to be erased), the flash memory page or sector codes (each one coded on two bytes, MSB first), and a checksum byte (XOR of the sent bytes). If the checksum is correct, the bootloader erases the memory and sends an ACK byte to the host; otherwise, it sends an NACK byte, and the command is aborted.

Erase Memory command specifications

The bootloader receives one half-word (two bytes) containing the number of pages or sectors to erase diminished by 1. If 0xFFFY is received (where Y is from 0 to F), a special erase is performed (0xFFFF for global mass erase, 0xFFFE and 0xFFFD, respectively, for bank1 and bank2 mass erase).

The bootloader receives:

- in the case of a special erase, one byte: the checksum of the previous bytes (for example, 0x00 for 0xFFFF)
- if N pages or sectors are erased, 2 x N bytes, each half-word of which contains a page or sector number coded on two bytes, with the MSB first. Then, all previous byte checksums are received in one byte.

Note: Some products do not support the mass erase feature, in this case use the Erase command to erase all pages or sectors.
The maximum number of pages or sectors is product-related, and must be respected. The maximum number of pages or sectors that can be erased in the same command is 512. Codes from 0xFFFC to 0xFFF0 are reserved.
No error is returned when performing erase operations on write-protected sectors.

For dual bank products, sectors 0 to (N / 2 - 1) correspond to bank1, sectors N / 2 to N -1 correspond to bank2. If option byte SWAP_BANK exists, sectors 0 to (N / 2 - 1) correspond to bank2, sectors N / 2 to N -1 correspond to bank1.

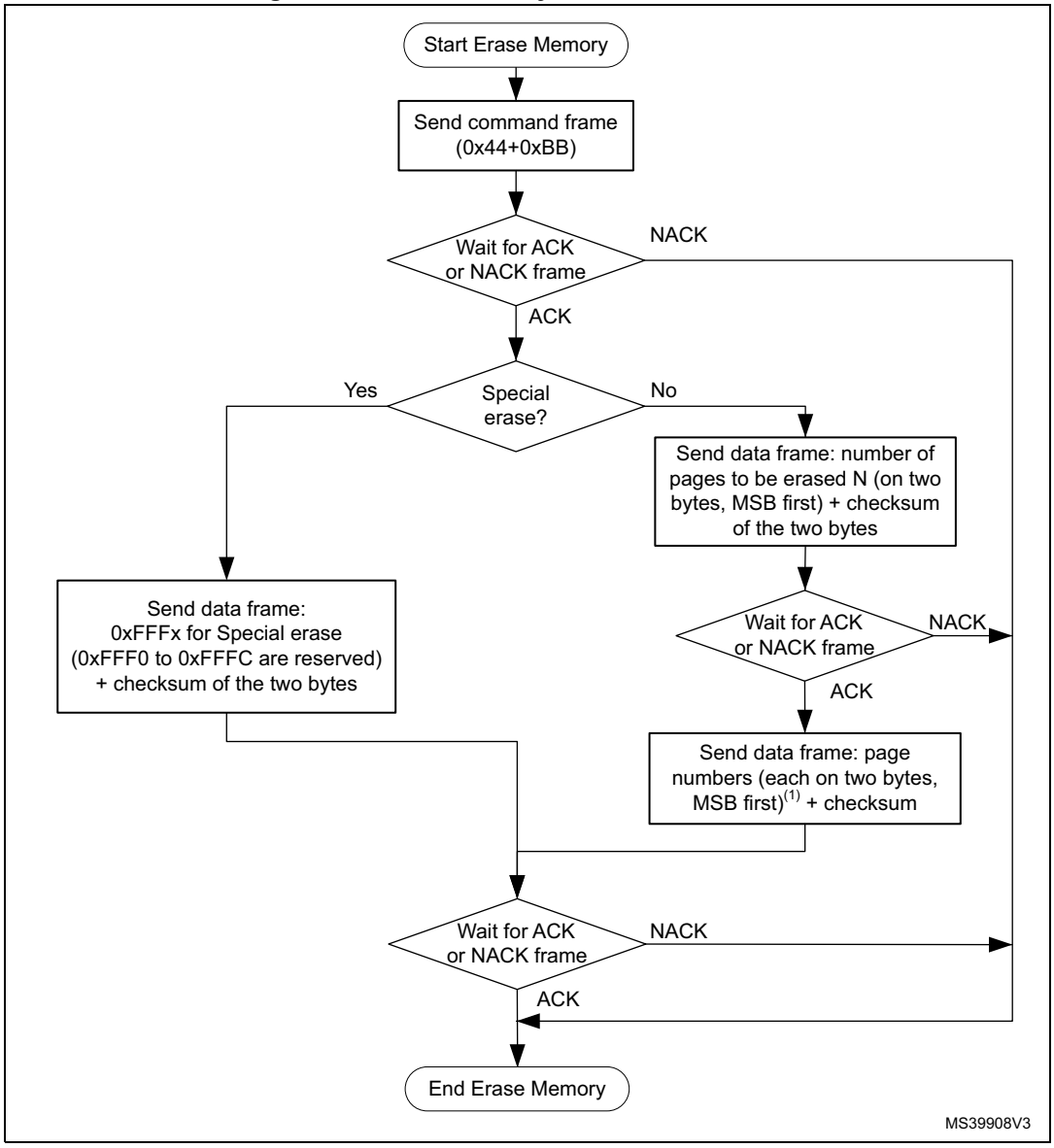
The host sends bytes to the STM32 as follows:

1. Byte 1: 0x44
2. Byte 2: 0xBB
3. Wait for ACK
4. Bytes 3-4: Special erase (0xFFFx) for Special erase, or Number of pages or sectors to be erased - 1 for Page erase
5. Byte 5: checksum of bytes 3-4
6. Wait for ACK
7. (2 x N) bytes (page numbers or sectors coded on two bytes, MSB first) and then the checksum for these bytes
8. Wait for ACK

- Example of I2C frame:
  - erase page 1:
    0x44 0xBB **Wait ACK** 0x00 0x00 0x00 **Wait ACK** 0x00 0x01 0x01 **Wait ACK**
  - erase page 1 and page 2:

0x44 0xBB **Wait ACK** 0x00 0x01 0x01 **Wait ACK** 0x00 0x01 0x00 0x02 0x03 **Wait ACK**

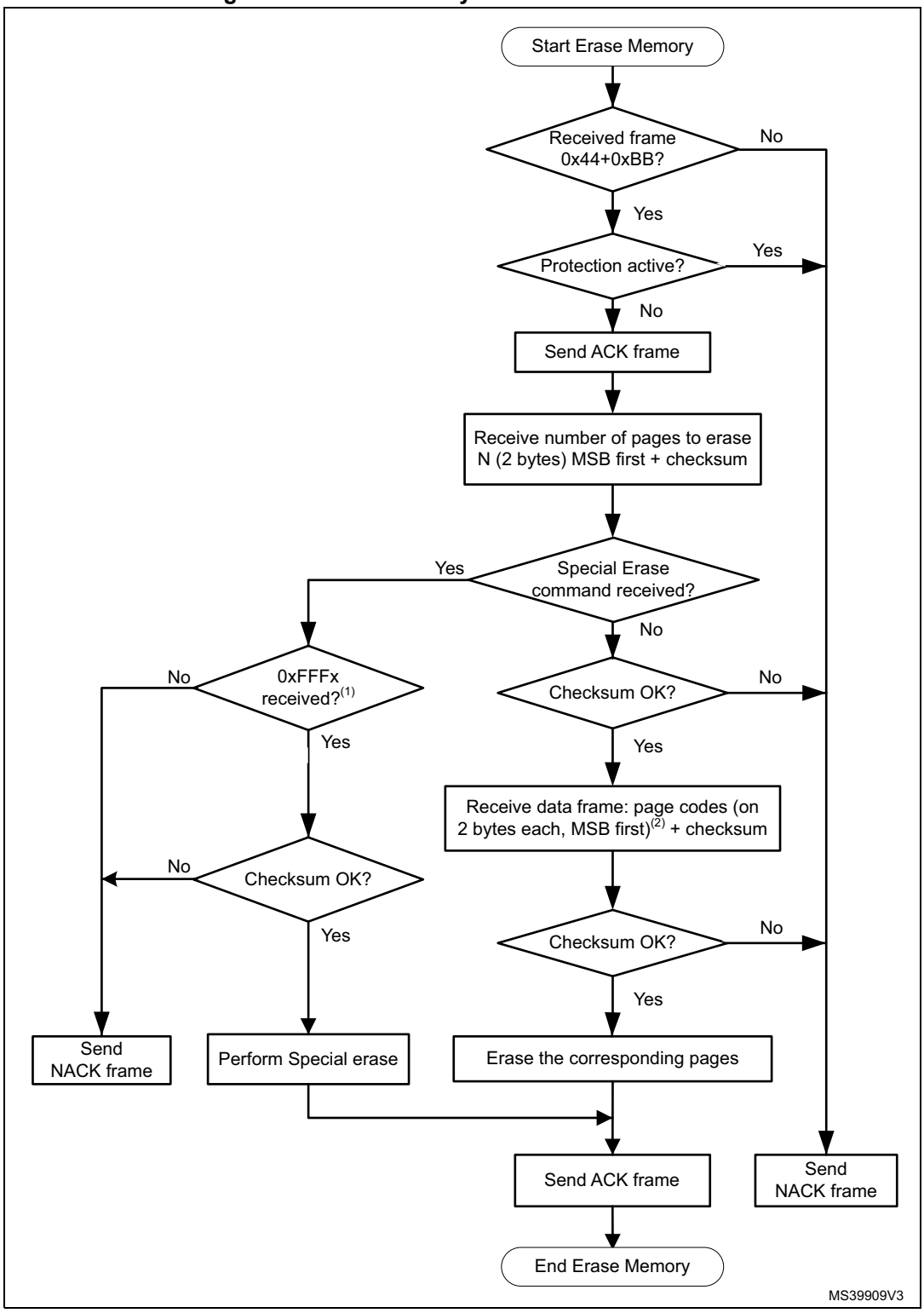
Figure 14. Erase Memory command: host side



1. For dual bank products, sectors 0 to (N / 2 - 1) correspond to bank1, sectors N / 2 to N -1 correspond to bank2. If option byte SWAP_BANK exists, sectors 0 to (N / 2 - 1) correspond to bank2, sectors N / 2 to N -1 correspond to bank1.

Note: Some products do not support the Special erase feature. For these products, this command is NACK-ed.

Figure 15. Erase Memory command: device side



1. Requested Special Erase command is NACK-ed if not supported by the used STM32 product.
2. For dual bank products, sectors 0 to (N / 2 - 1) correspond to bank1, sectors N / 2 to N -1 correspond to bank2. If option byte SWAP_BANK exists, sectors 0 to (N / 2 - 1) correspond to bank2, sectors N / 2 to N -1 correspond to bank1.

2.8 Write Protect command

This command is used to enable the write protection for some or all flash memory sectors. When the bootloader receives the command, it transmits the ACK byte to the host. The bootloader then waits for the number of bytes to be received (sectors to be protected), and then receives the flash memory sector codes from the application.

At the end of the Write Protect command, the bootloader transmits the ACK byte and generates a system reset to take the new configuration of the option byte into account.

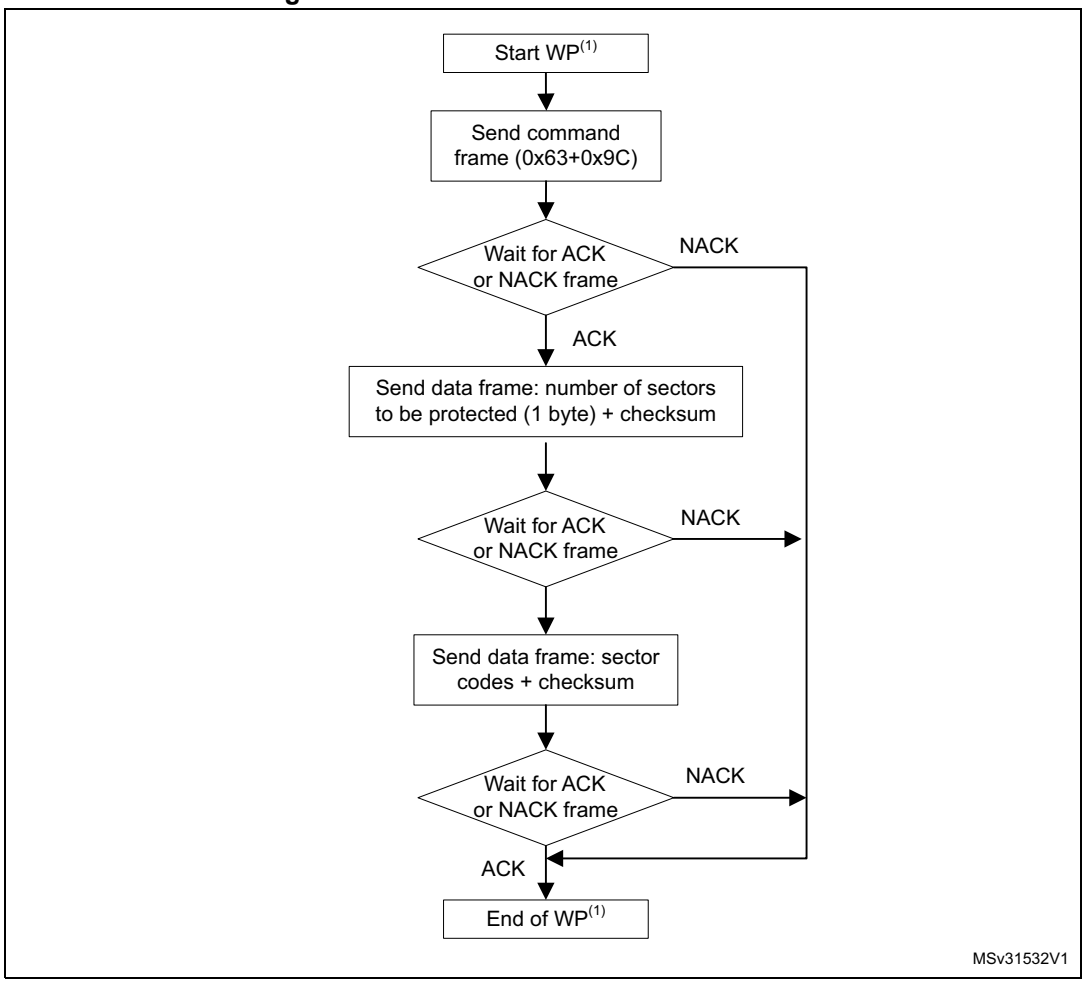
The Write Protect command sequence is as follows:

- The bootloader receives one byte that contains N, the number of sectors to be write-protected - 1 ($0 \leq N \leq 255$).
- The bootloader receives (N + 1) bytes, each of them containing a sector code.

Note: The total number of sectors and the number of the sector to be protected are not checked. This means that no error is returned when a command is passed with a wrong number of sectors to be protected, or with a wrong sector number.

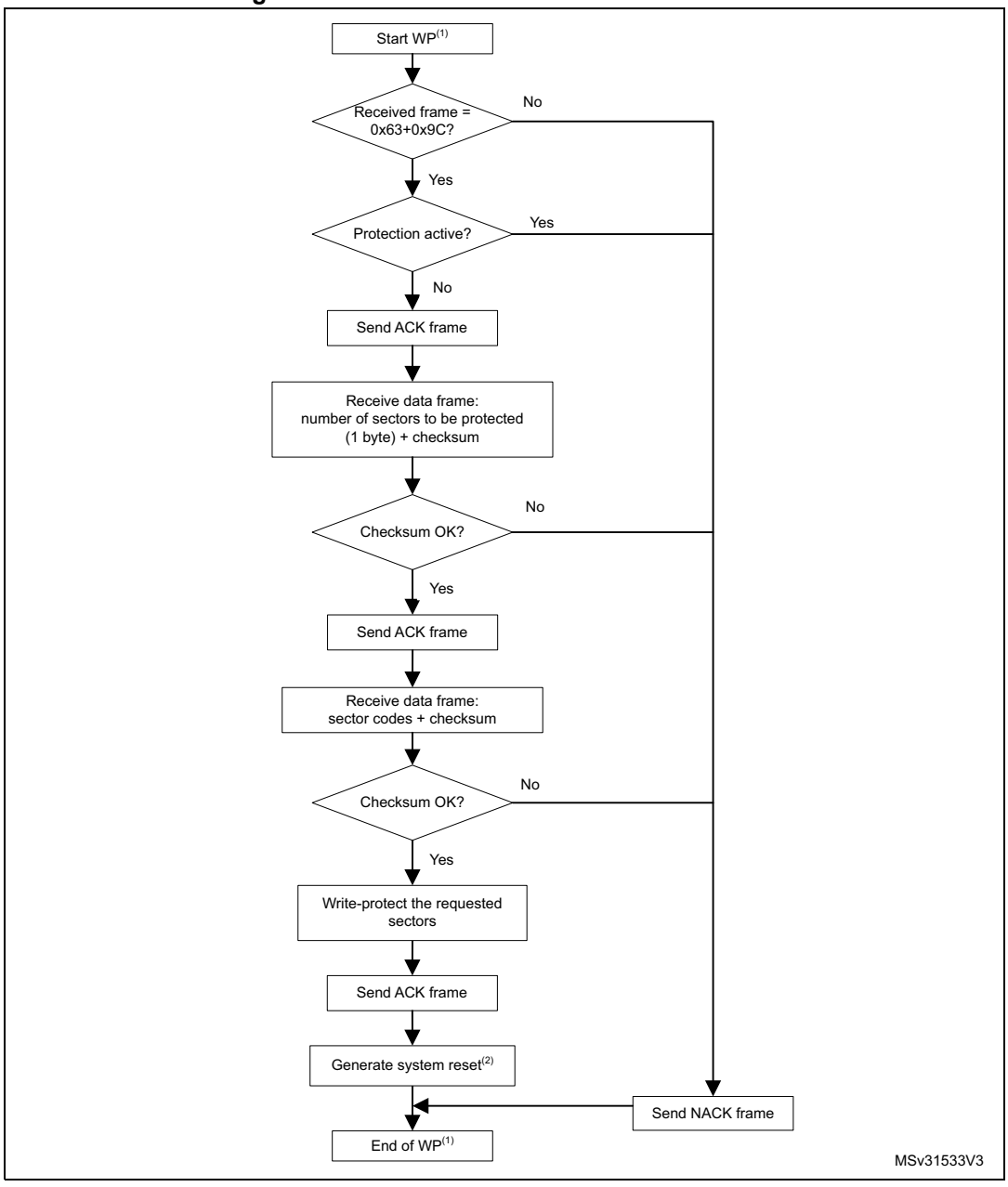
If a second Write Protect command is executed, the flash memory sectors protected by the first command become unprotected, and only the sectors passed with the second Write Protect command become protected.

Figure 16. Write Protect command: host side



1. WP = Write Protect.

Figure 17. Write Protect command: device side



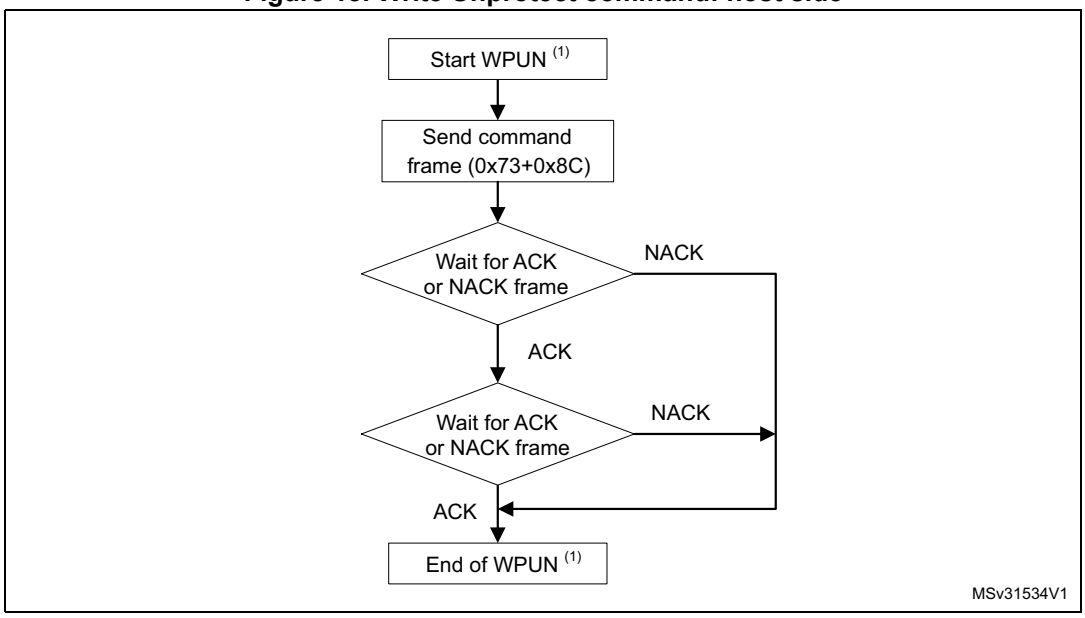
1. WP = Write Protect.
2. System reset is called only for some STM32 BL (STM32F0/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

2.9 Write Unprotect command

This command is used to disable the write protection of all flash memory sectors. When the bootloader receives the command, it transmits the ACK byte to the host, then disables the write protection of all flash memory sectors, and transmits the ACK byte.

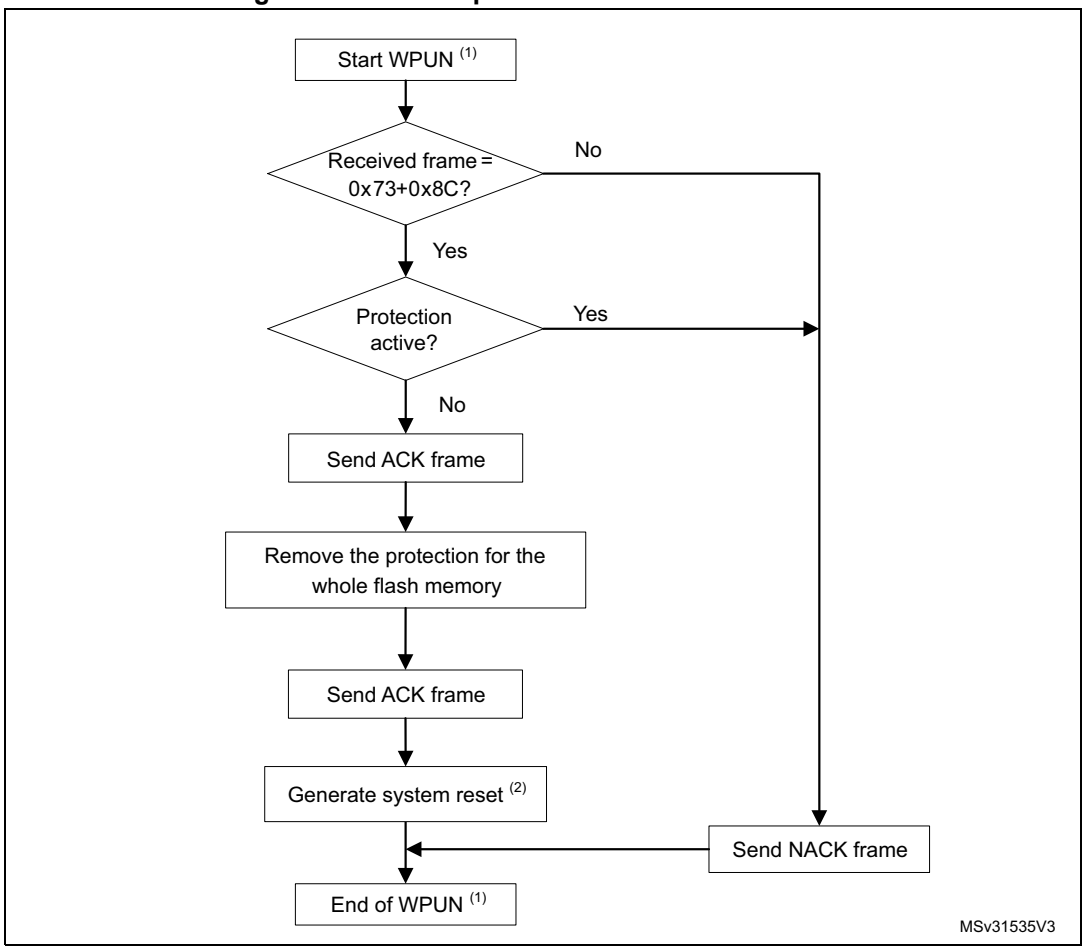
A system reset is generated to take the new configuration of the option byte into account.

Figure 18. Write Unprotect command: host side



1. WPUN = Write Unprotect.

Figure 19. Write Unprotect command: device side



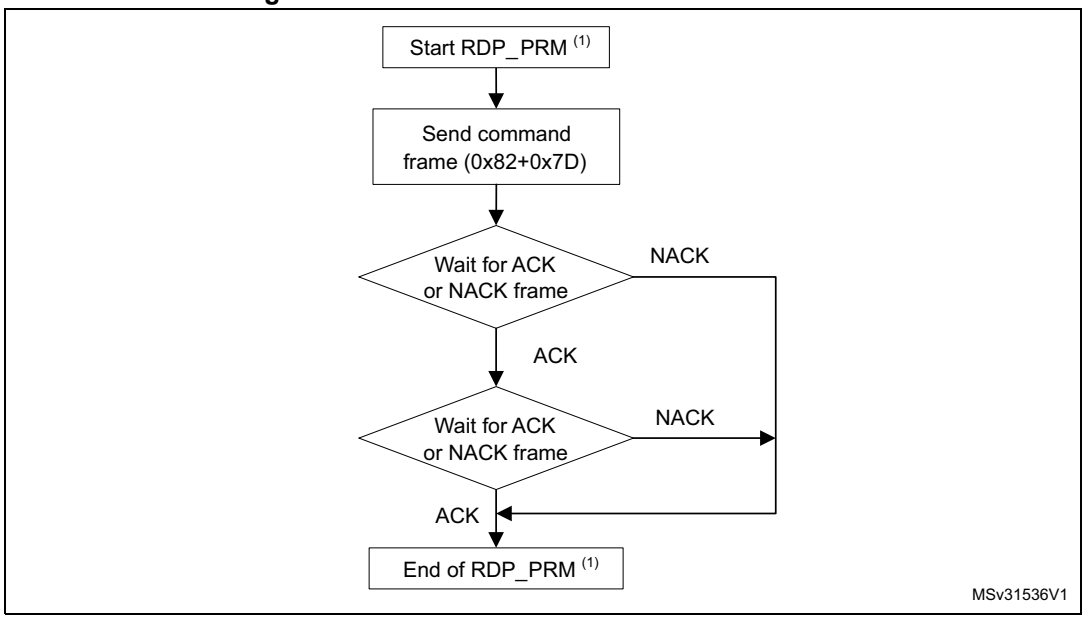
1. WPUN = Write Unprotect.
2. System reset is called only for some STM32 BL (STM32F0/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

2.10 Readout Protect command

This command is used to enable the flash memory read protection. When the bootloader receives the command, it transmits the ACK byte to the host, and enables the read protection for the memory.

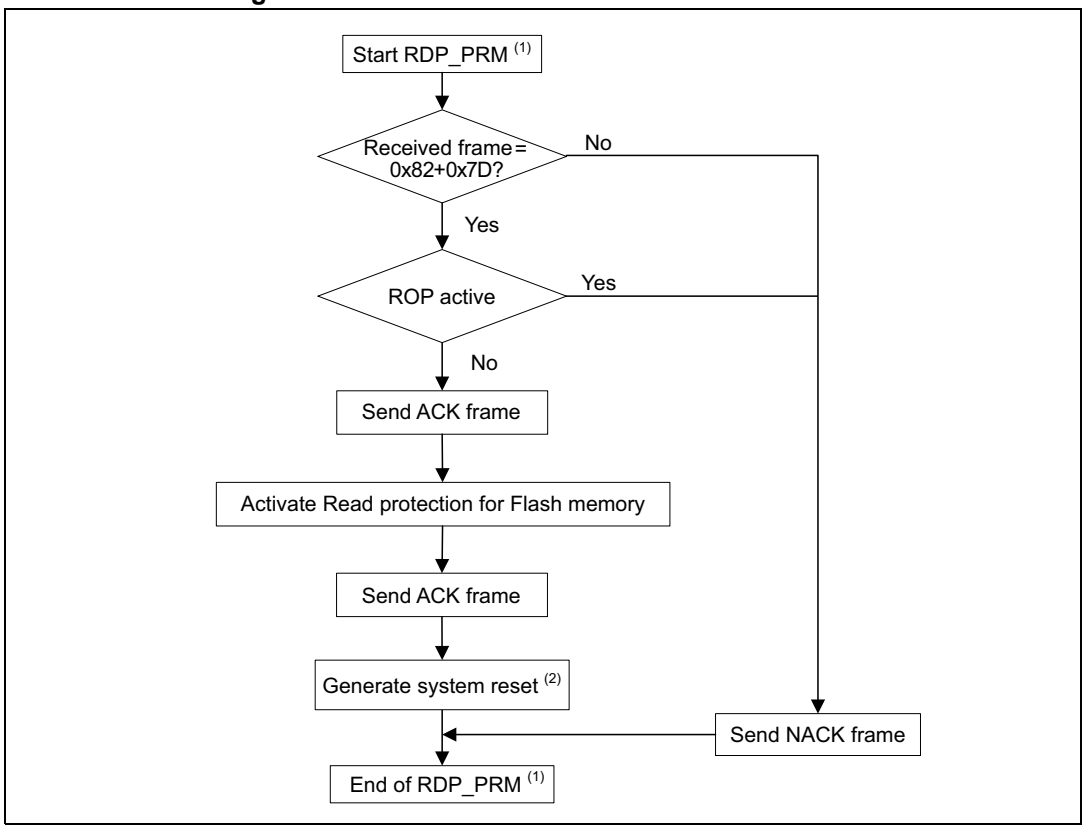
At the end of the Readout Protect command, the bootloader transmits the ACK byte and generates a system reset to take the new configuration of the option byte into account.

Figure 20. Readout Protect command: host side



1. RDP_PRM = Readout Protect.

Figure 21. Readout Protect command: device side



1. RDP_PRM = Readout Protect.
2. System reset is called only for some STM32 BL (STM32F0/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

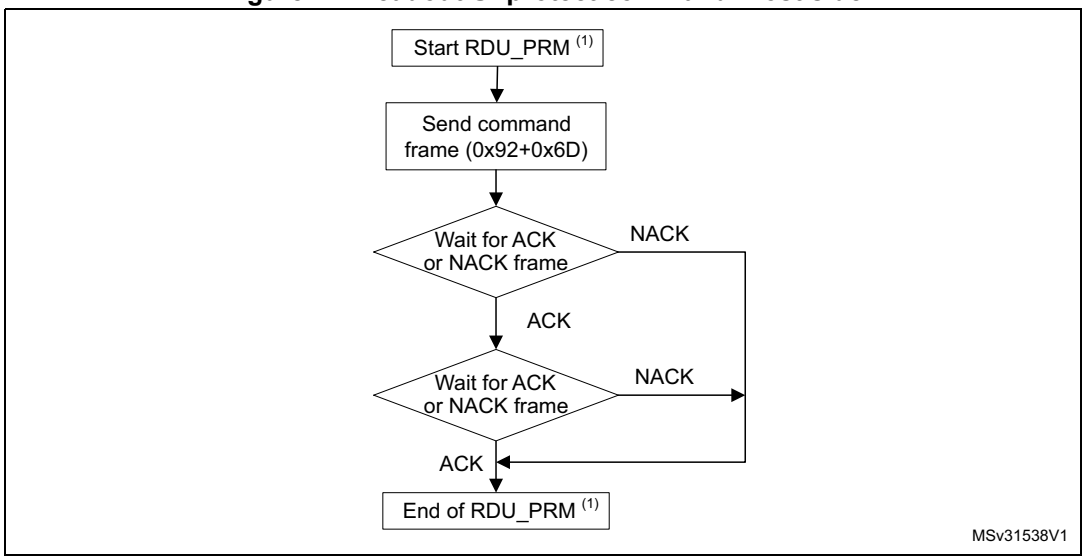
2.11 Readout Unprotect command

This command is used to disable flash memory read protection. When the bootloader receives the command, it transmits the ACK byte to the host, then disables the read protection for the whole flash memory, which results in a total erasure. If the operation is not successful, the bootloader transmits an NACK, and the read protection remains active.

Note: This operation takes the same time to erase all pages or sectors (or to perform a mass erase if supported by the product), so the host must wait until it ends. For the flash memory erase timings refer to the product datasheet.

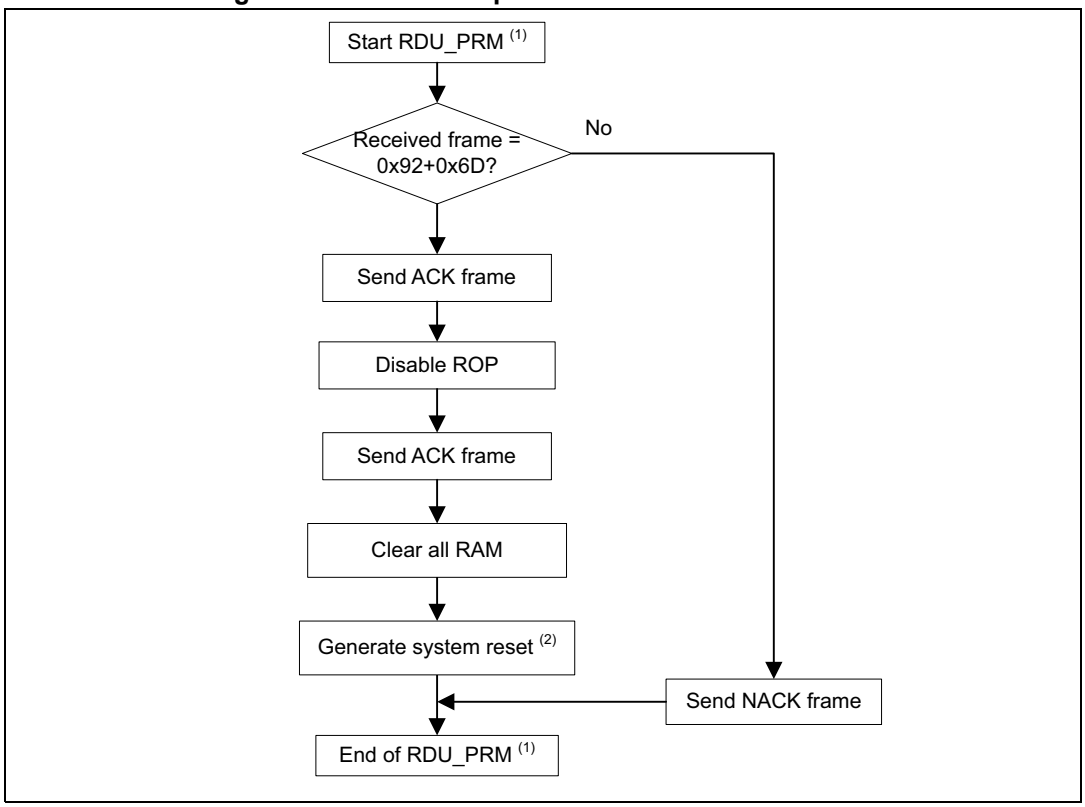
At the end of the Readout Unprotect command, the bootloader transmits an ACK and generates a system reset to take the new configuration of the option byte into account.

Figure 22. Readout Unprotect command: host side



1. RDU_PRM = Readout Unprotect.

Figure 23. Readout Unprotect command: device side



1. RDU_PRM = Readout Unprotect.
2. System reset is called only for some STM32 BL (STM32F0/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

2.12 No-Stretch Write Memory command

The No-Stretch Write Memory command is used to write data to any valid memory area.

When the bootloader receives the No-Stretch Write Memory command, it transmits the ACK byte to the application. The bootloader then waits for a 4-byte address (byte 1 is the address MSB, and byte 4 is the LSB) and a checksum byte, and then checks the received address.

If the received address is valid and the checksum is correct, the bootloader transmits an ACK byte; otherwise, it transmits an NACK byte and aborts the command. When the address is valid and the checksum is correct, the bootloader:

1. Gets a byte, N, which contains the number of data bytes to be received
2. Receives the user data ((N + 1) bytes) and the checksum (XOR of N and of all data bytes)[(a)]
3. Programs the user data to memory, starting from the received address
4. Returns a Busy state (0x76) while the operation is ongoing

At the end of the command, if the write operation is successful, the bootloader transmits the ACK byte; otherwise, it transmits an NACK byte to the application and aborts the command.

If the Write Memory command is issued to the option byte area, all bytes are erased, new values are written, and the bytes are reloaded.

Note: The maximum length of the block to write is 256 bytes. For the option bytes the maximum length depends upon the product, and the address received from the host must be the start address of the option byte area. For more information, refer to the reference manual.

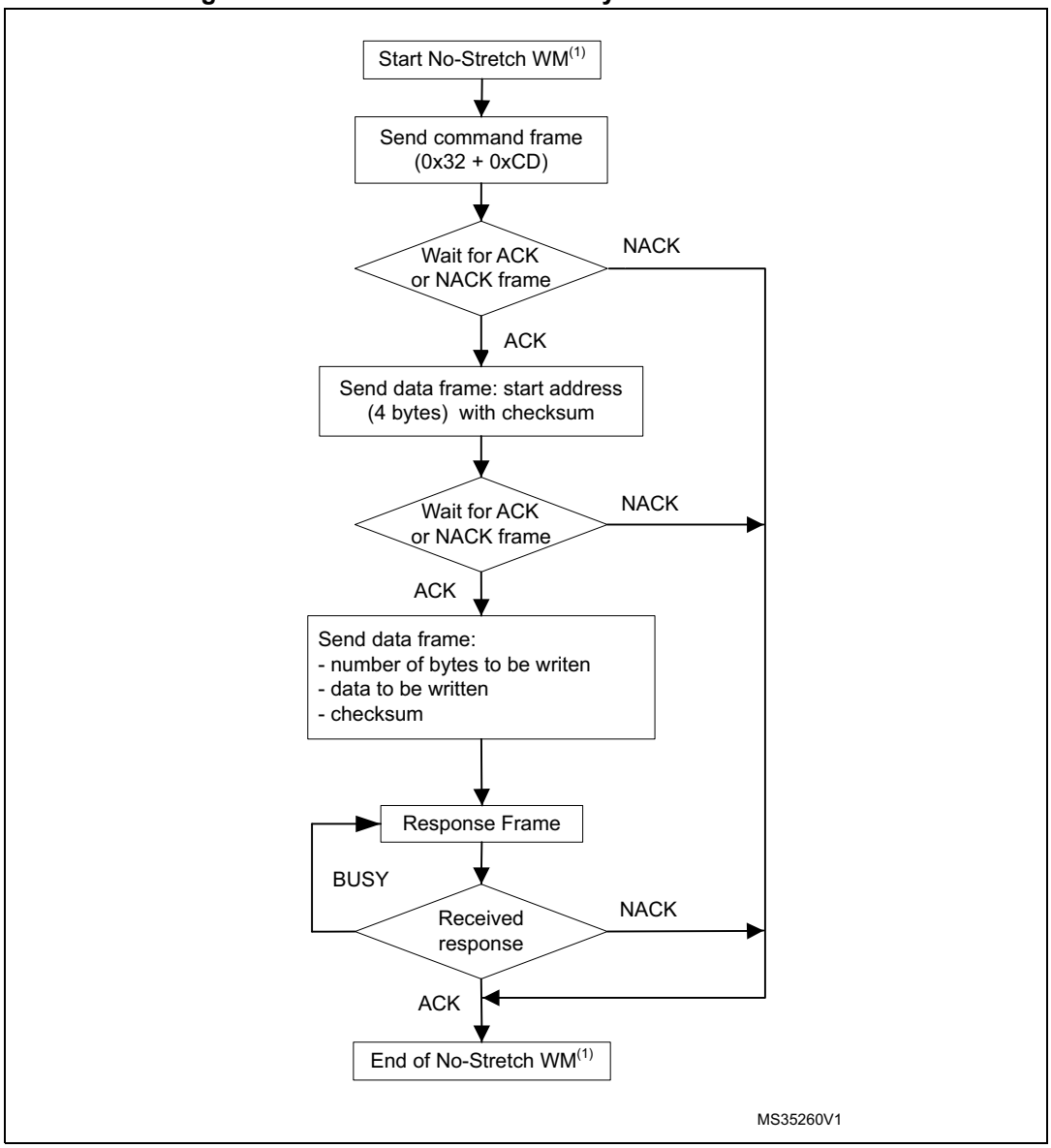
No error is returned when performing write operations to write-protected sectors.

The host sends the bytes to the STM32 as follows:

1. Byte 1: 0x32
2. Byte 2: 0xCD
3. Wait for ACK
4. Byte 3 to byte 6: start address
   - Byte 3: MSB
   - Byte 6: LSB
5. Byte 7: checksum: XOR (byte 3, byte 4, byte 5, byte 6)
6. Wait for ACK
7. Byte 8: number of bytes to be received (0 < N ≤ 255)
8. N +1 data bytes[(a)] (max 256 bytes)
9. Checksum byte: XOR (N, N+1 data bytes)
10. Wait for ACK (if Busy, keep polling on ACK/NACK)

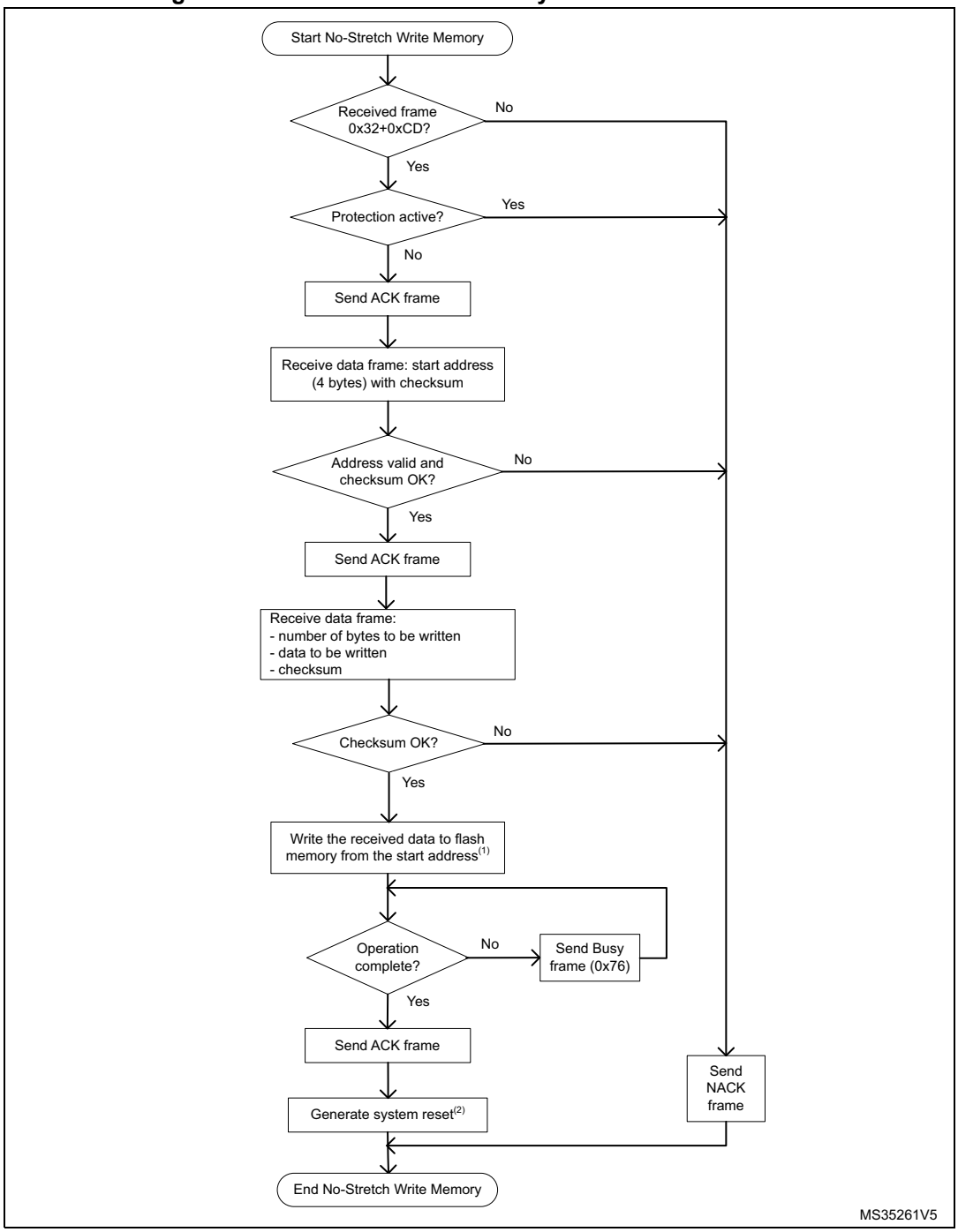
a. Every word (U32) must be transmitted in LSB to MSB order (byte 1 is the LSB, and byte 4 is the MSB).

Figure 24. No-Stretch Write Memory command: host side



1. WM = Write Memory.

Figure 25. No-Stretch Write Memory command: device side



1. If the address to write correspond to the option byte address, an option byte reload is triggered after writing the data aligned with every MCU flash interface specification.
2. A system reset is called only for some STM32 BL (STM32F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

2.13 No-Stretch Erase Memory command

This command allows the host to erase flash memory pages or sectors using a two-byte addressing mode.

When the bootloader receives the command, it transmits the ACK byte to the host. The bootloader then receives two bytes (number of pages or sectors to be erased), the flash memory page or sector codes (each one coded on two bytes, MSB first) and a checksum byte (XOR of the sent bytes). If the checksum is correct, the bootloader erases the memory (returns the Busy state 0x76 while operation is ongoing), then sends an ACK byte to the host; otherwise, it sends an NACK byte to the host, and the command is aborted.

No-Stretch Erase Memory command specifications

The bootloader receives one half-word (two bytes) containing the number of pages or sectors to erase diminished by 1. If 0xFFFY is received (where Y is from 0 to F), a special erase is performed (0xFFFF for global mass erase, 0xFFFE and 0xFFFD, respectively, for bank1 and bank2 mass erase).

The bootloader receives:

- In the case of a special erase, one byte: the checksum of the previous bytes (for example, 0x00 for 0xFFFF)
- If N pages or sectors are erased, 2 x N bytes, each half-word of which contains a page or sector number coded on two bytes, with the MSB first. All previous byte checksums are received in one byte.

Note: Some products do not support the mass erase feature, in this case use the Erase command to erase all pages or sectors.
The maximum number of pages or sectors depends upon the product, and must be respected. The maximum number of pages or sectors that can be erased in the same command is 512.
Codes from 0xFFFC to 0xFFF0 are reserved.
No error is returned when performing erase operations on write-protected sectors.

For dual bank products, sectors 0 to (N / 2 - 1) correspond to bank1, sectors N / 2 to N -1 correspond to bank2. If option byte SWAP_BANK exists, sectors 0 to (N / 2 - 1) correspond to bank2, sectors N / 2 to N -1 correspond to bank1.

The host sends bytes to the STM32 as follows:

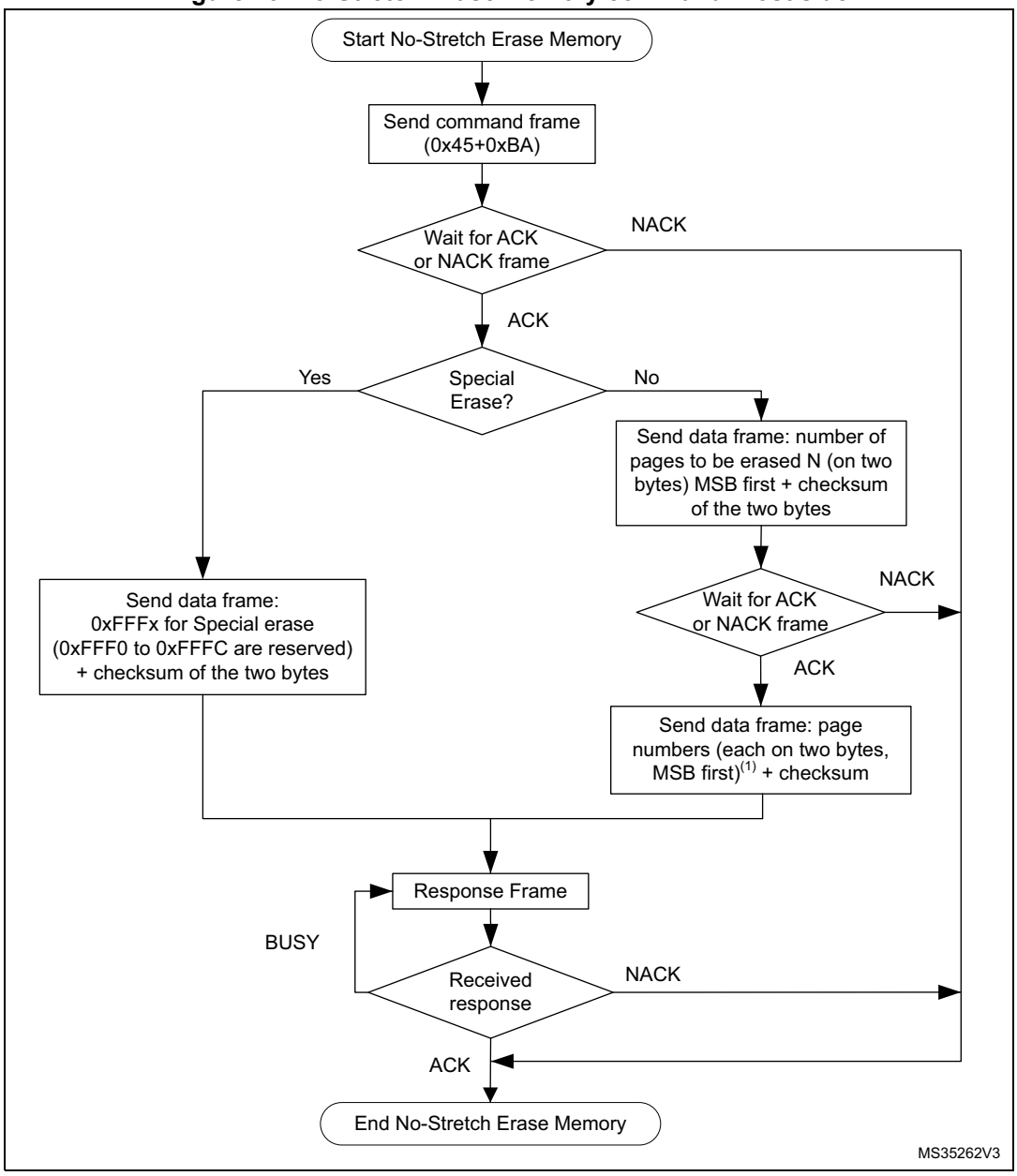
1. Byte 1: 0x45
2. Byte 2: 0xBA
3. Wait for ACK
4. Bytes 3-4: Special erase (0xFFFx) for Special erase, Number of pages or sectors to erase - 1 for Page erase
5. Byte 5: checksum of bytes 3-4
6. Wait for ACK (if Busy, keep polling on ACK/NACK)
7. (2 x N) bytes (page numbers or sectors coded on two bytes, MSB first), and then the checksum for these bytes
8. Wait for ACK (if Busy, keep polling on ACK/NACK)

- Example of I2C frame:
  - erase page 1:
    0x45 0xBA **Wait ACK** 0x00 0x00 0x00 **Wait ACK** 0x00 0x01 0x01 **Wait ACK**

– erase page 1 and page 2:

0x45 0xBA **Wait ACK** 0x00 0x01 0x01 **Wait ACK** 0x00 0x01 0x00 0x02 0x03 **Wait ACK**

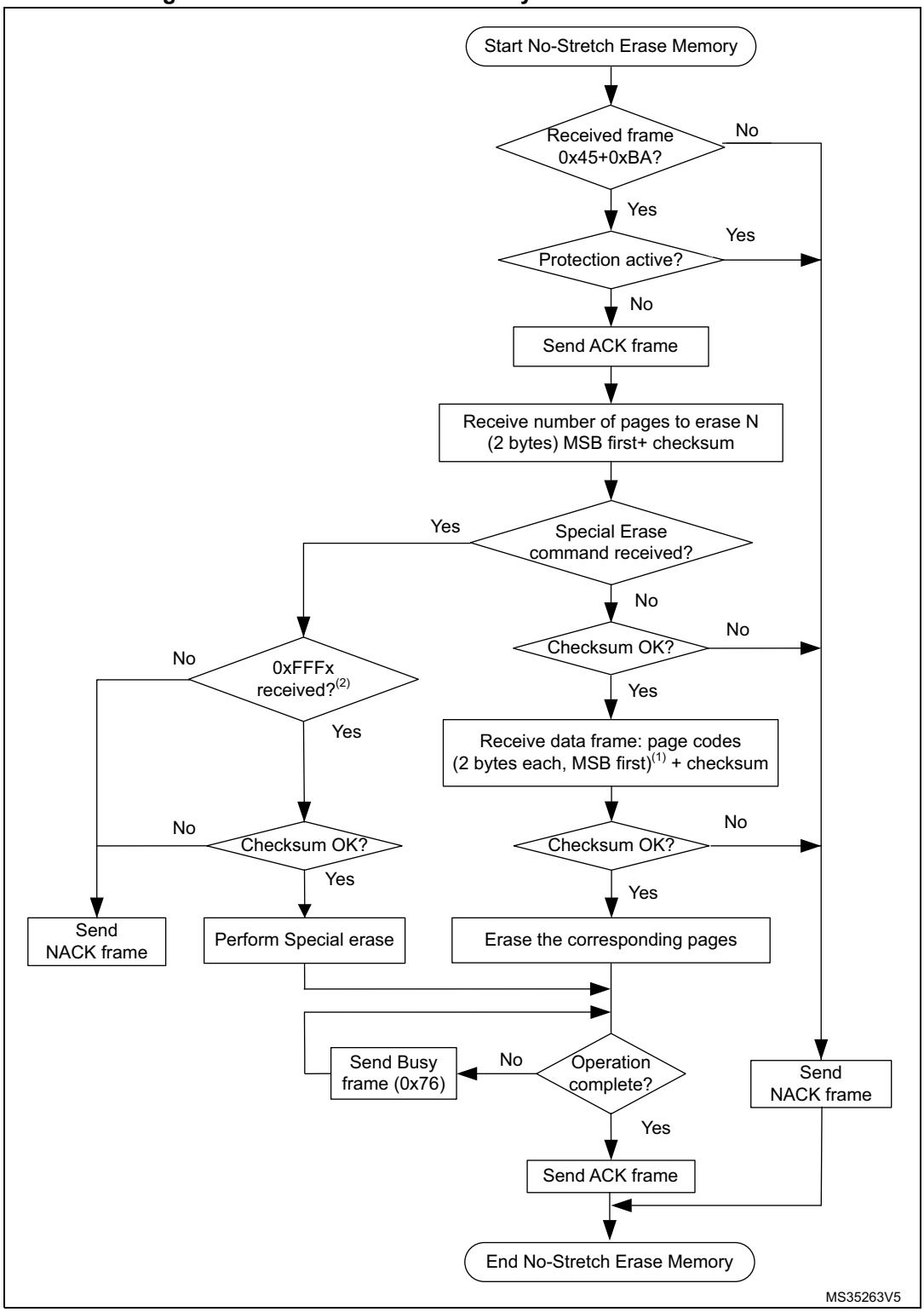
Figure 26. No-Stretch Erase Memory command: host side



1. For dual bank products, sectors 0 to (N / 2 - 1) correspond to bank1, sectors N / 2 to N -1 correspond to bank2. If option byte SWAP_BANK exists, sectors 0 to (N / 2 - 1) correspond to bank2, sectors N / 2 to N -1 correspond to bank1.

Note: Some products do not support the Special erase feature. For these products, this command is NACK-ed.

Figure 27. No-Stretch Erase Memory command: device side



1. For dual bank products, sectors 0 to (N / 2 - 1) correspond to bank1, sectors N / 2 to N -1 correspond to bank2. If option byte SWAP_BANK exists, sectors 0 to (N / 2 - 1) correspond to bank2, sectors N / 2 to N -1 correspond to bank1.
2. Requested Special erase command is NACK-ed if not supported by the used STM32 product.

2.14 Special command

New bootloader commands are needed to support new STM32 features and to fulfill customers needs. To avoid specific commands for a single project, the Special command has been created, to be as generic as possible.

Figure 28. Special command: host side

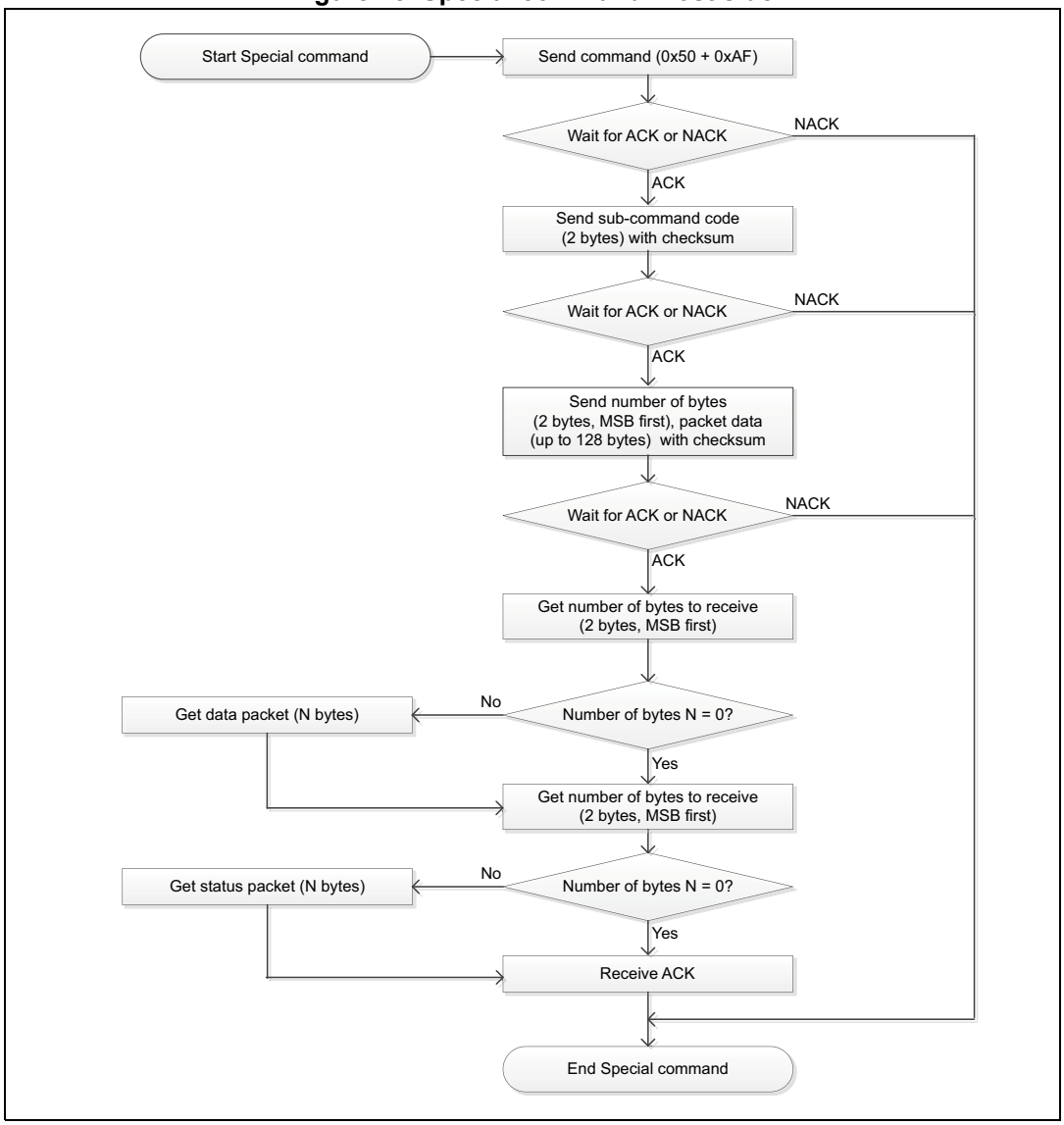
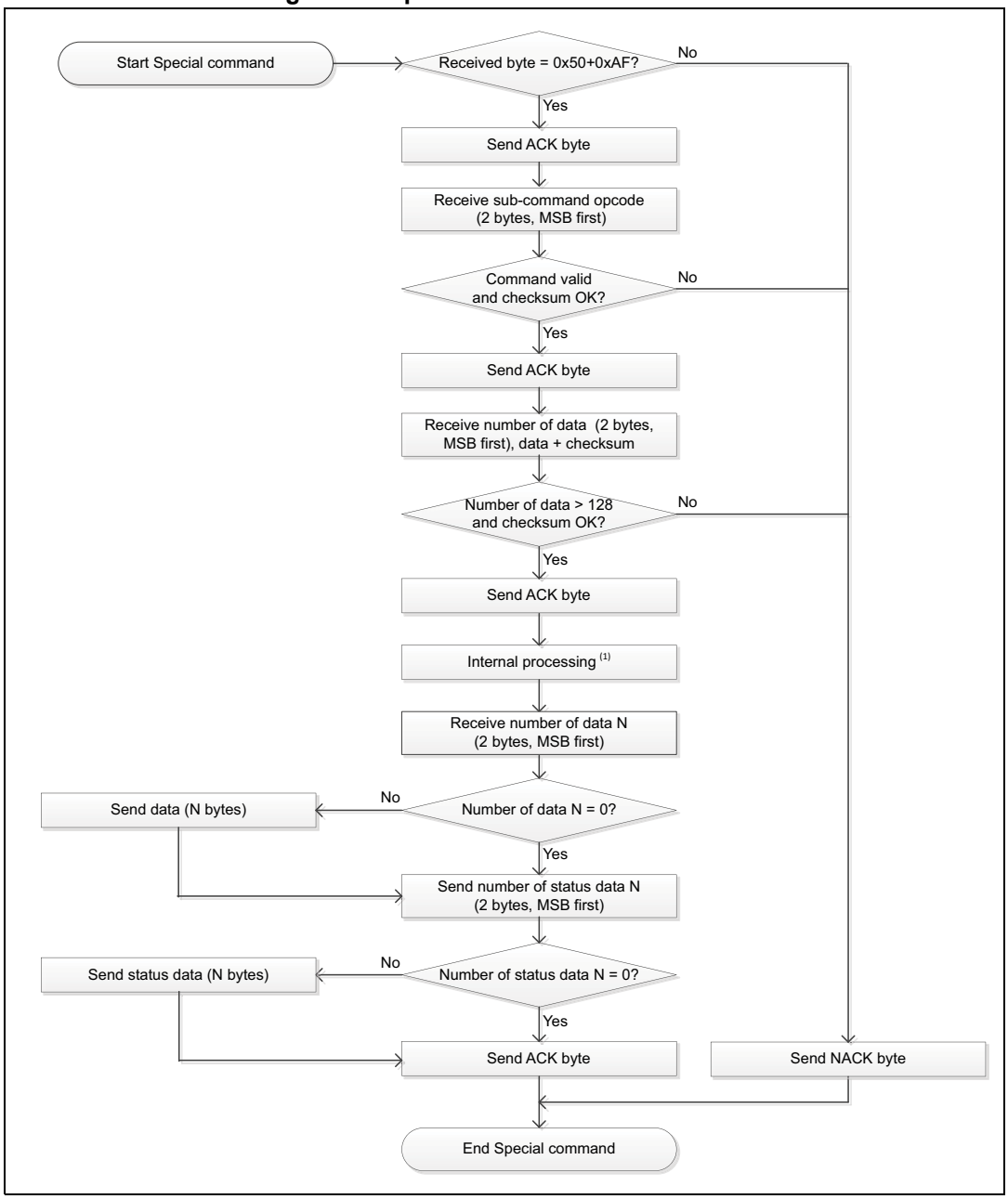


Figure 29. Special command: device side



1. The internal processing depends on the project needs.

When the bootloader receives the Special command, it transmits the ACK byte to the host. Once the ACK is transmitted, the bootloader waits for a subcommand opcode (two bytes, MSB first) and a checksum byte. If the subcommand is supported by the STM32 bootloader and its checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits an NACK byte and aborts the command.

To keep the command generic the data packet received by the bootloader can have different sizes depending on the subcommand needs.

Therefore, the packet is split in two parts:

- Size of the data (2 bytes, MSB first)
- N bytes of data
  - If N = 0, no data are transmitted
  - N must be lower than 128.

If all conditions are satisfied (N ≤ 128 and the checksum is correct), the bootloader transmits an ACK. Otherwise, it transmits an NACK byte and aborts the command.

Once the subcommand is executed using the received data, the bootloader sends a response consisting of two consecutive packets:

- Data packet
  - Size of the data (2 bytes, MSB first)
  - N bytes of data
  - If N = 0, no data are transmitted
- Status packet
  - Size of the status data (2 bytes, MSB first)
  - N bytes of data
  - If N = 0, no status data are transmitted

An ACK byte closes the Special command interaction between the bootloader and the host.

2.15 Extended Special command

This command is slightly different the Special command. It allows the user to send more data with the addition of a new buffer of 1024 bytes and as a response it only returns the commands status.

Figure 30. Extended Special command: host side

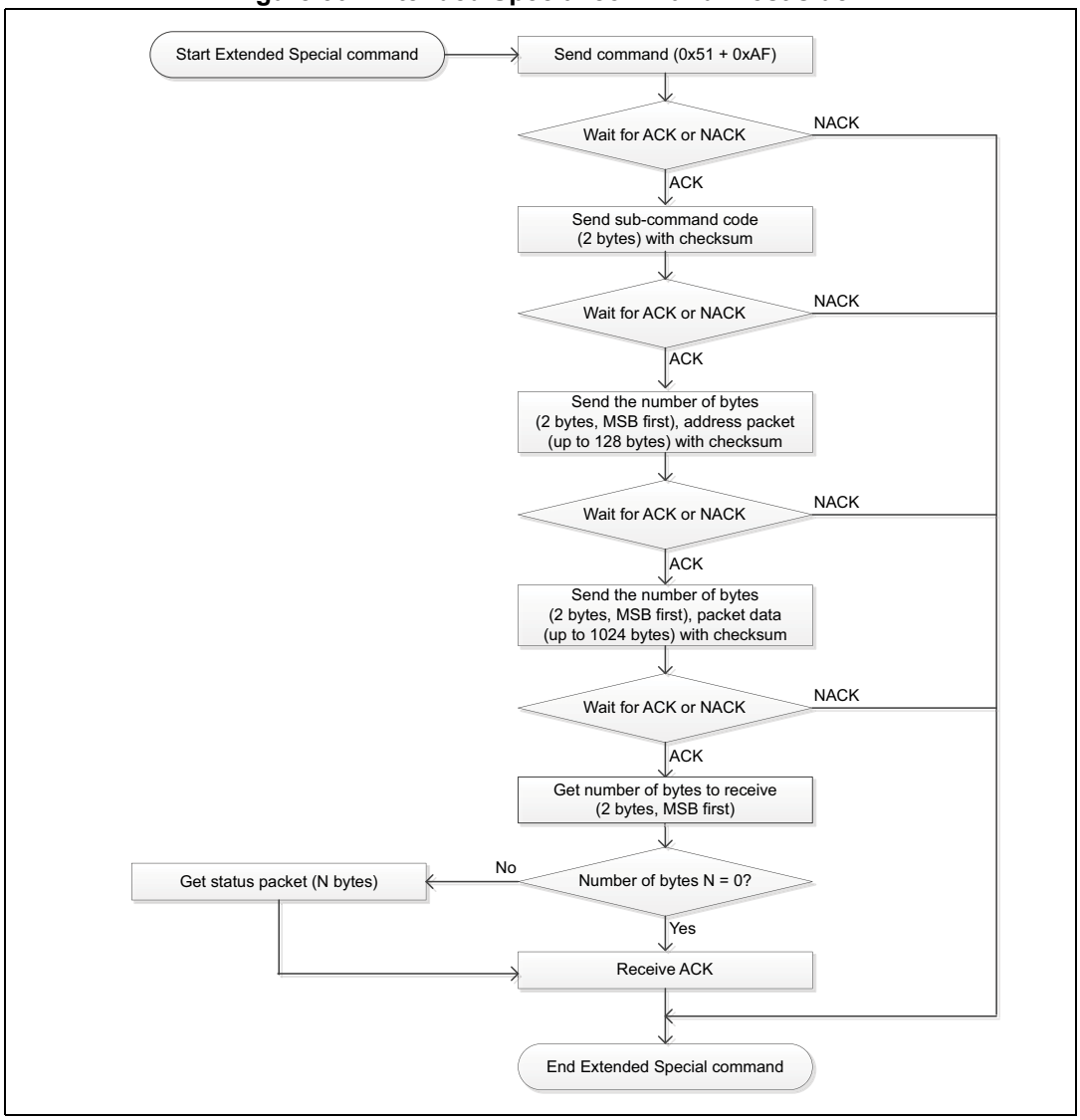
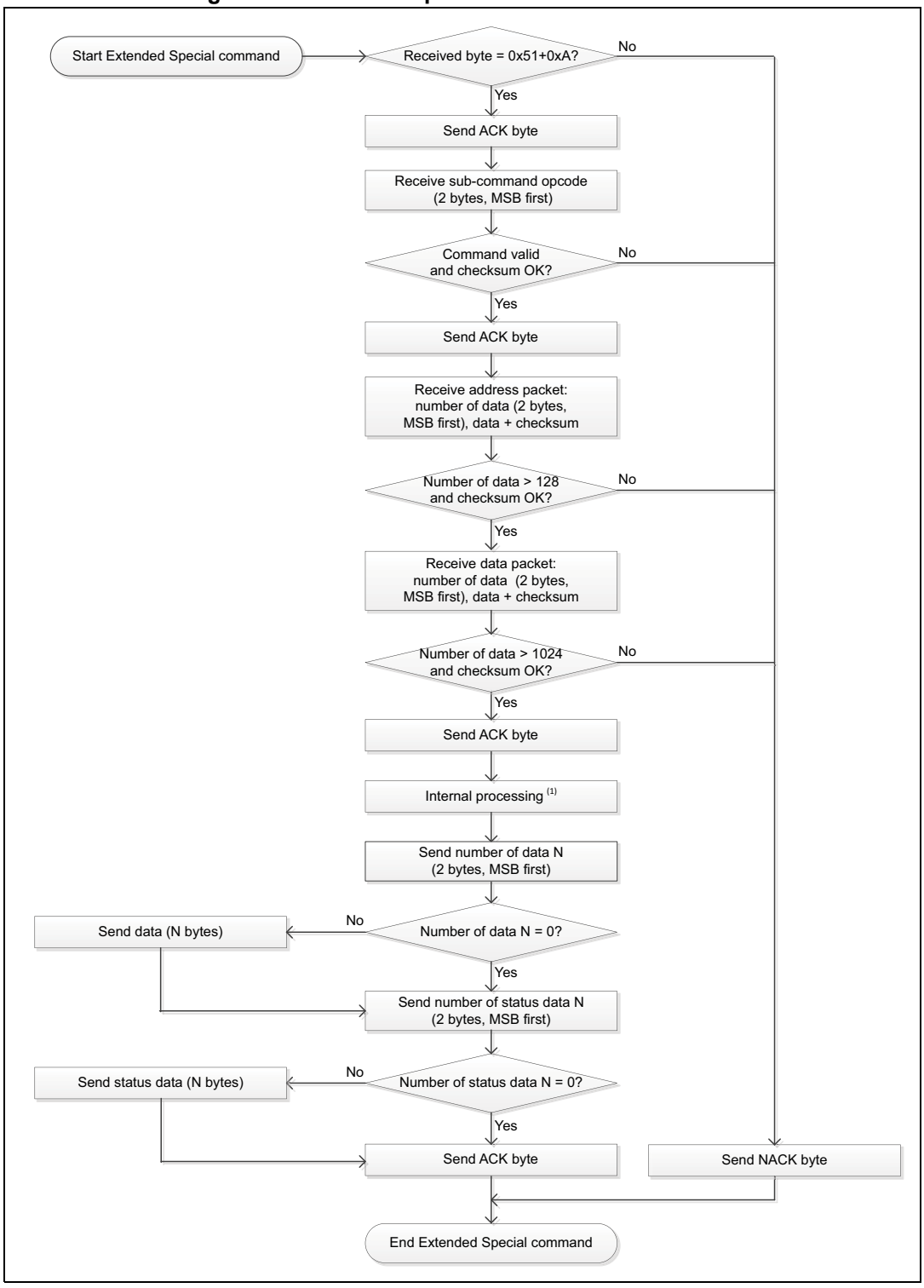


Figure 31. Extended Special command: device side



1. The internal processing depends on the project needs.

When the bootloader receives the extended special command, it transmits the ACK byte to the host. Once the ACK is transmitted, the bootloader waits for a subcommand opcode (two bytes, MSB first) and a checksum byte. If the subcommand is supported and its checksum

is correct, the bootloader transmits an ACK byte, otherwise it transmits an NACK byte and aborts the command.

The two packets then can be received depending on the subcommand needs:

- Packet1: Data1 packet, where the number of bytes is limited to 128 bytes.
- Packet2: Data2 packet, where the number of bytes is limited to 1024 bytes.

If all conditions are satisfied (Packet1: N ≤ 128 and checksum is correct, Packet2: N ≤ 1024 and checksum is correct), the bootloader transmits an ACK, otherwise it transmits an NACK byte and aborts the command.

Once the subcommand is executed using the received data, the bootloader sends a response consisting of one packet:

- Size of the data (2 bytes, MSB first)
- N bytes of data
  - If N = 0, no data are transmitted

An ACK byte closes the Extended Special command interaction between the bootloader and the host.

2.16 No-Stretch Write Protect command

This command is used to enable the write protection for some or all flash memory sectors.

When the bootloader receives the command, it transmits the ACK byte to the host. The bootloader then waits for the number of bytes to be received (sectors to be protected), then receives the flash memory sector codes from the application and returns Busy state (0x76) while operation is ongoing.

At the end of the No-Stretch Write Protect command, the bootloader transmits the ACK byte and generates a system reset to take the new configuration of the option byte into account.

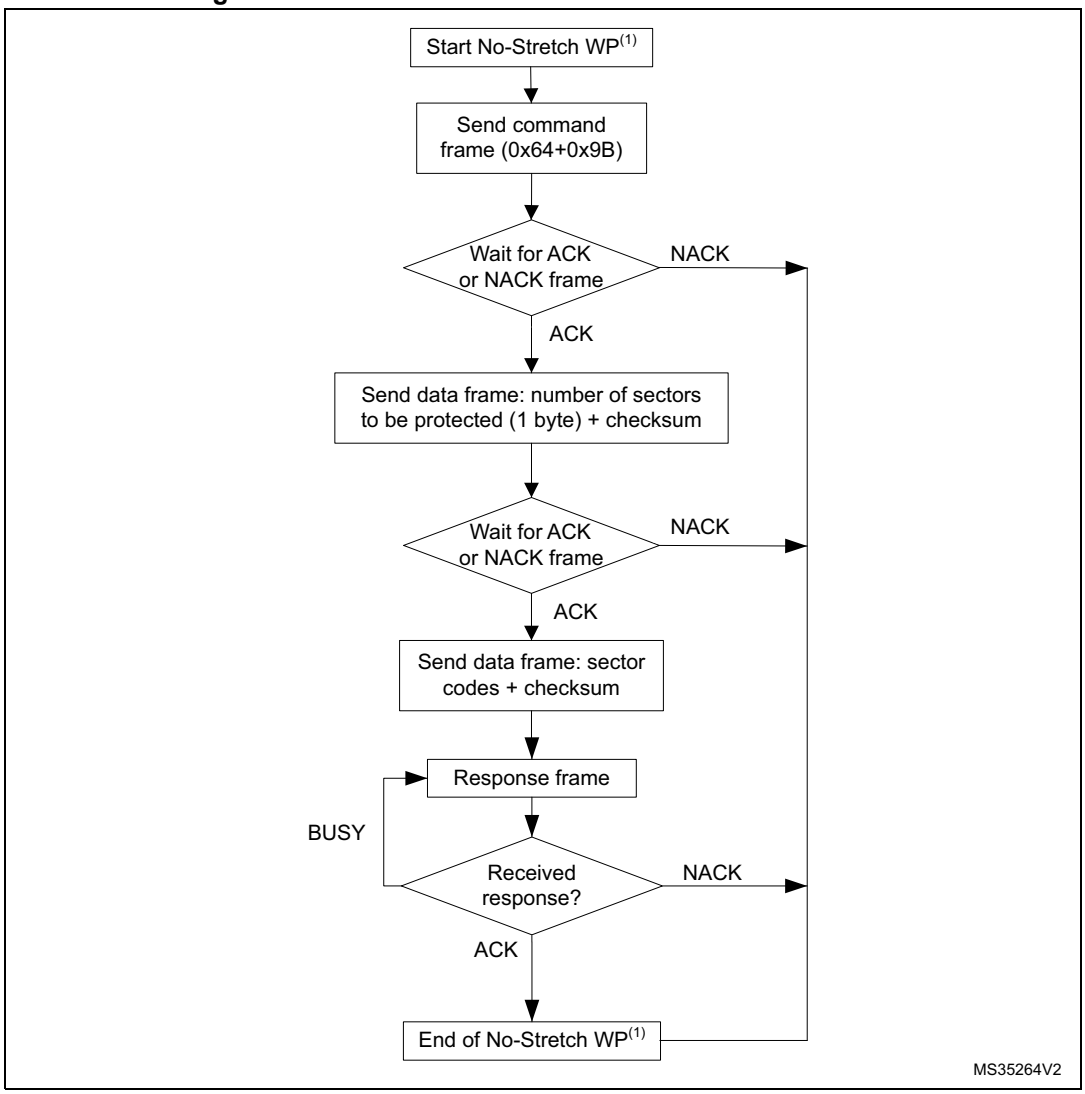
The Write Protect command sequence is as follows:

- The bootloader receives one byte that contains N, the number of sectors to be write-protected - 1 (0 ≤ N ≤ 255).
- The bootloader receives (N + 1) bytes, each byte of which contains a sector code.

Note: The total number of sectors and the sector number to be protected are not checked. This means that no error is returned when a command is passed with either a wrong number of sectors to be protected, or a wrong sector number.

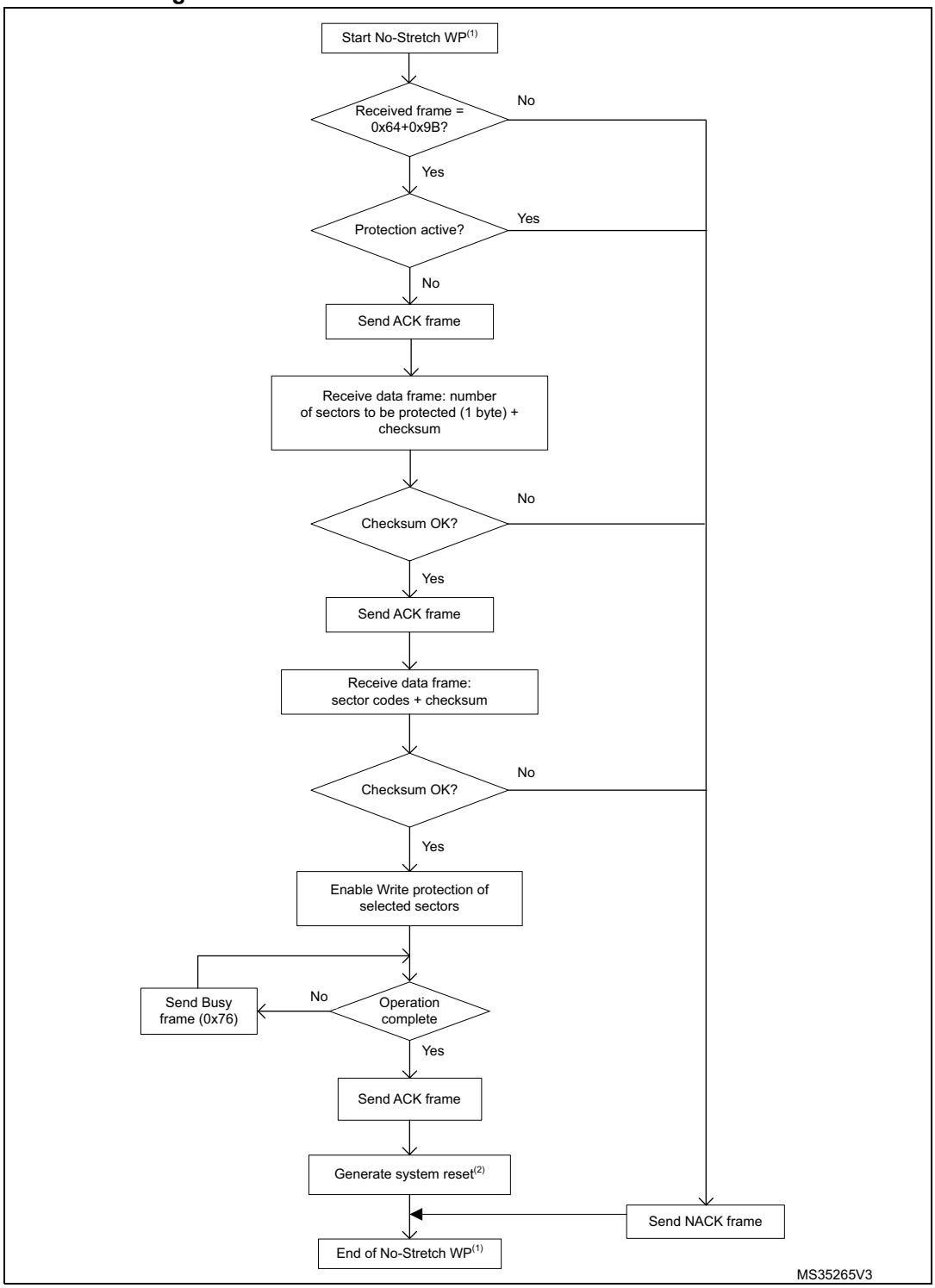
If a second Write Protect command is executed, the flash memory sectors that had been protected by the first command become unprotected, and only the sectors passed within the second Write Protect command become protected.

Figure 32. No-Stretch Write Protect command: host side



1. WP = Write Protect.

Figure 33. No-Stretch Write Protect command: device side



1. WP = Write Protect.
2. System reset is called only for some STM32 BL (STM32F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

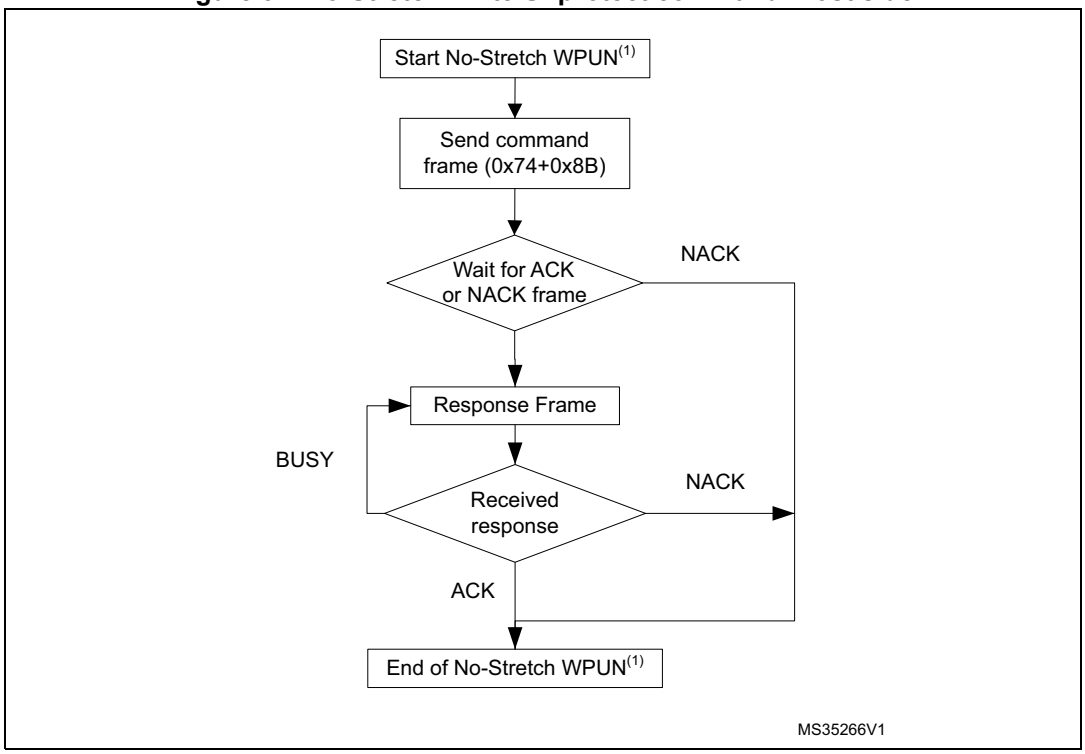
2.17 No-Stretch Write Unprotect command

This command is used to disable the write protection of all flash memory sectors.

When the bootloader receives the command, it transmits the ACK byte to the host, then disables the write protection of all flash memory sectors, and returns Busy state (0x76) while the operation is ongoing. At the end, it transmits the ACK byte.

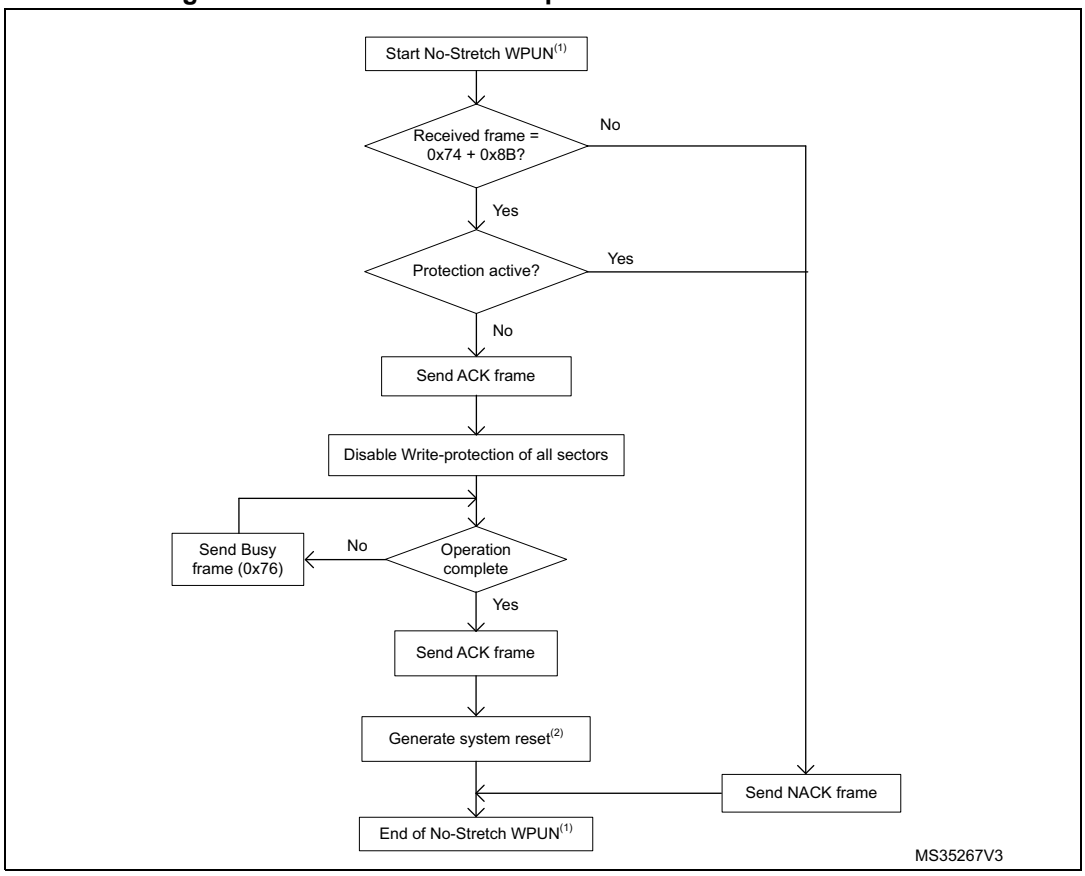
A system reset is generated to take the new configuration of the option byte into account.

Figure 34. No-Stretch Write Unprotect command: host side



1. WPUN = Write Unprotect.

Figure 35. No-Stretch Write Unprotect command: device side



1. WPUN = Write Unprotect.
2. System reset is called only for some STM32 BL (STM32F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

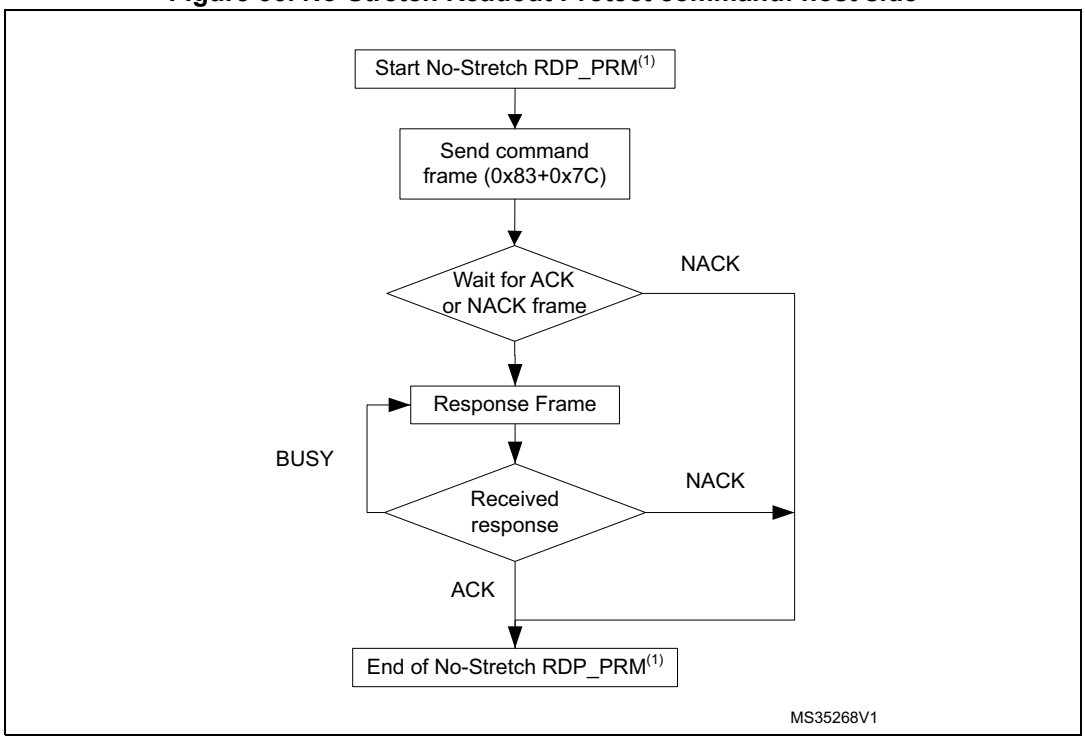
2.18 No-Stretch Readout Protect command

This command is used to enable the flash memory read protection.

When the bootloader receives the command, it transmits the ACK byte to the host, enables the read protection for the flash memory, and returns Busy state (0x76) while the operation is ongoing.

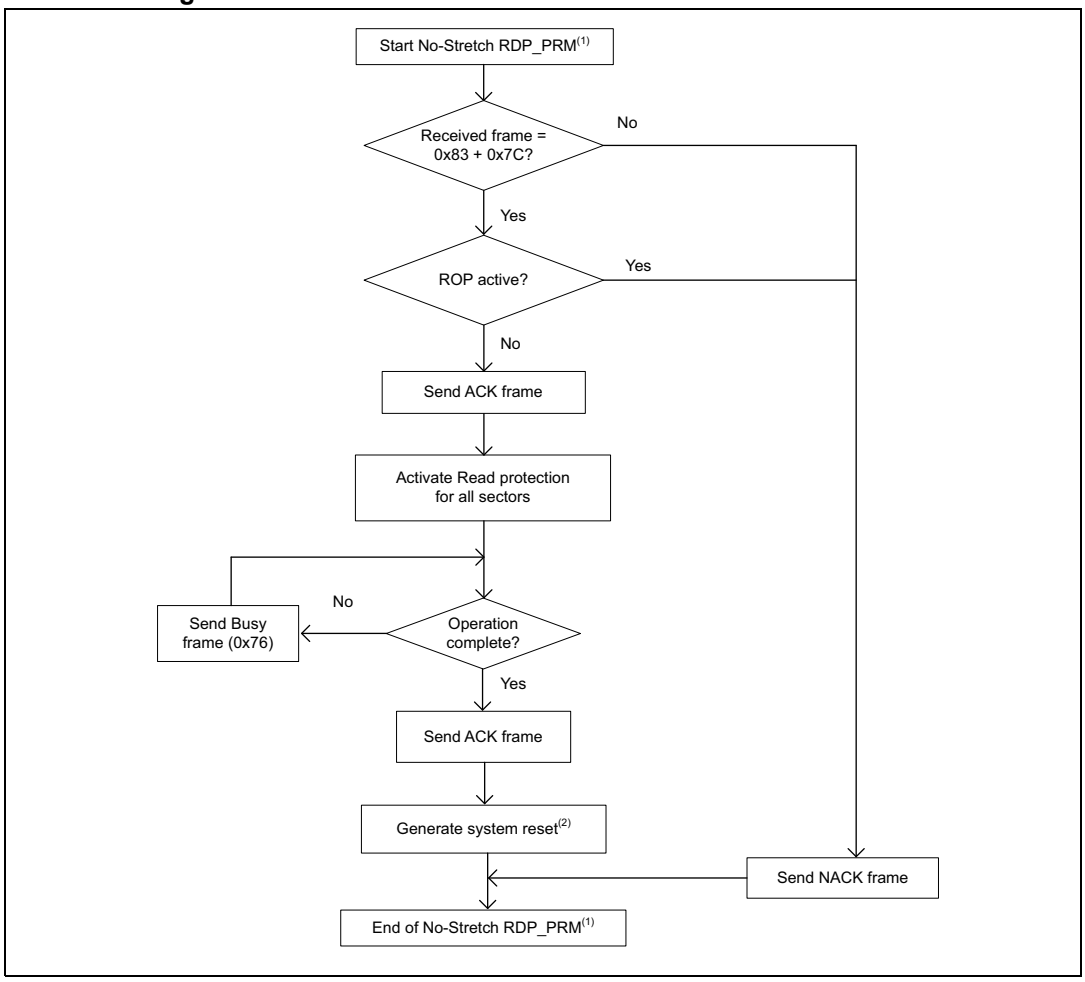
At the end of the No-Stretch Readout Protect command, the bootloader transmits the ACK byte, and generates a system reset to take the new configuration of the option byte into account.

Figure 36. No-Stretch Readout Protect command: host side



1. RDP_PRM = Readout Protect.

Figure 37. No-Stretch Readout Protect command: device side



1. RDP_PRM = Readout Protect.
2. System reset is called only for some STM32 BL (STM32F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

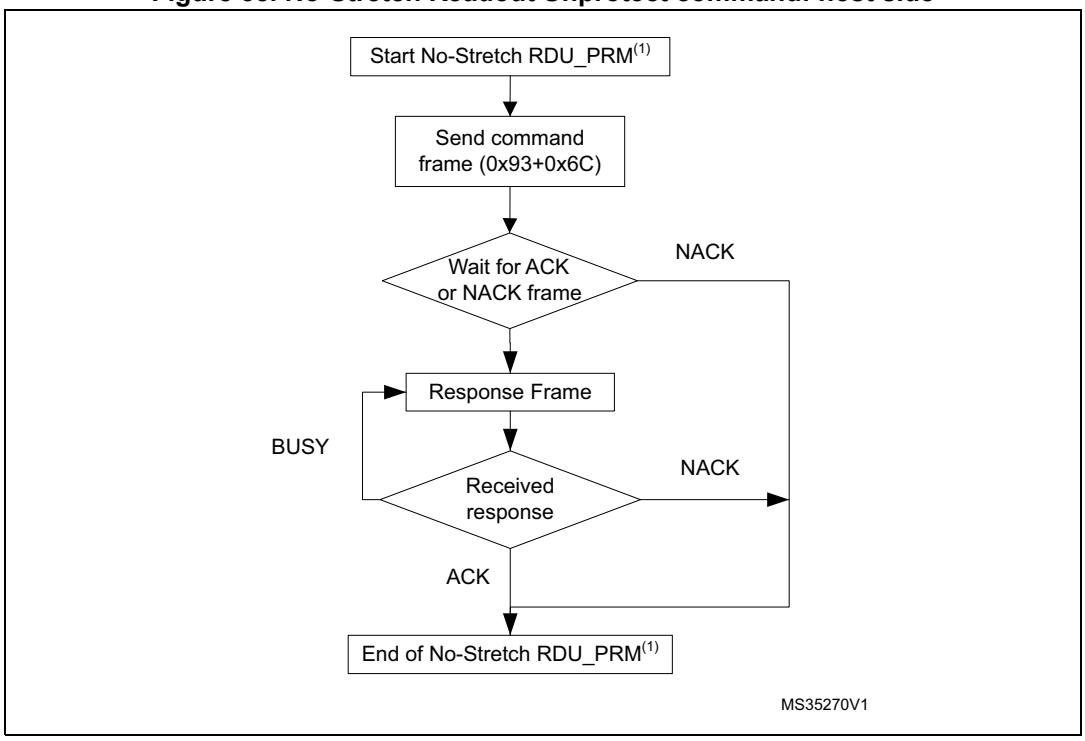
2.19 No-Stretch Readout Unprotect command

This command is used to disable flash memory read protection. When the bootloader receives the command, it transmits the ACK byte to the host.

The bootloader then disables the read protection for the entire flash memory, which results in an erasure of the entire flash memory, and returns Busy state (0x76) while the operation is ongoing. If the operation is unsuccessful, the bootloader transmits an NACK, and the read protection remains active.

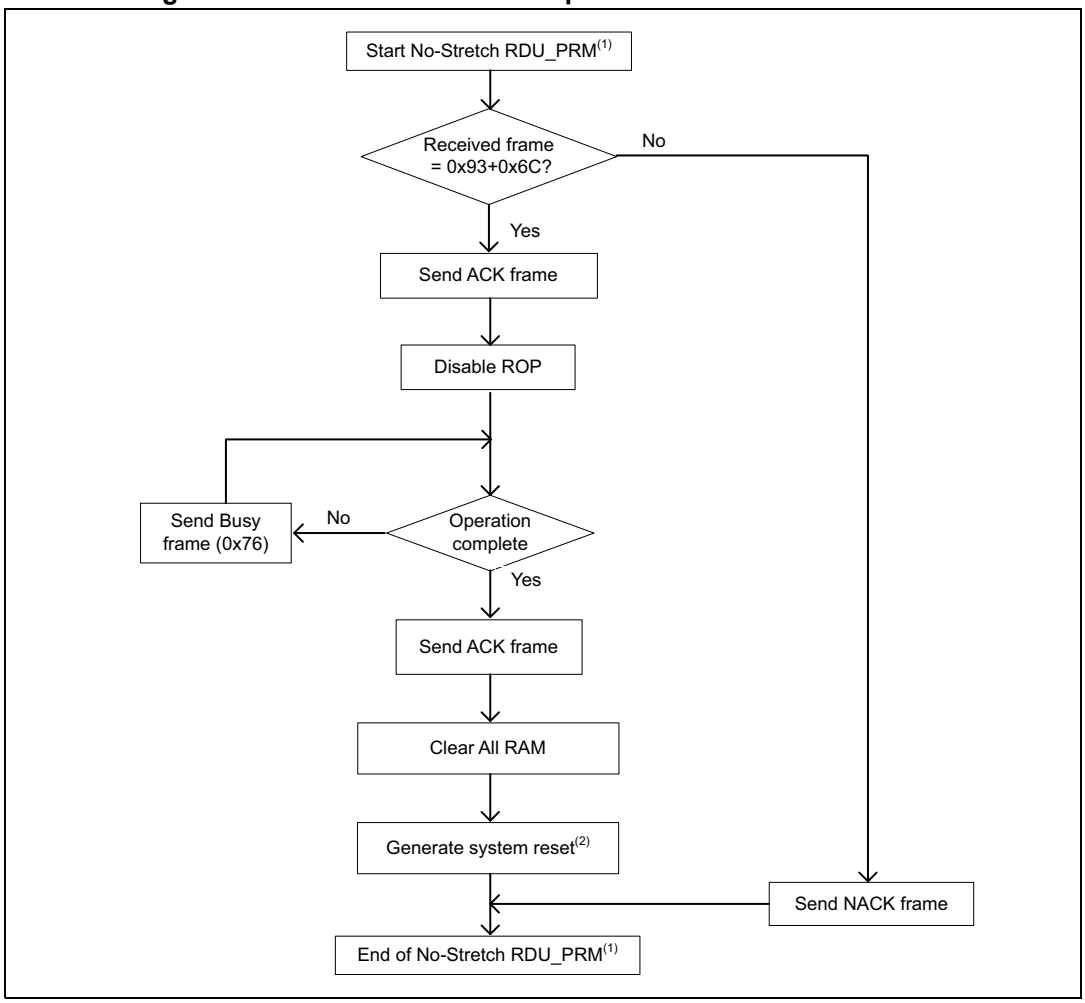
At the end of the No-Stretch Readout Unprotect command, the bootloader transmits an ACK and generates a system reset to take the new configuration of the option byte into account.

Figure 38. No-Stretch Readout Unprotect command: host side



1. RDU_PRM = Readout Unprotect.

Figure 39. No-Stretch Readout Unprotect command: device side



1. RDU_PRM = Readout Unprotect.
2. System reset is called only for some STM32 BL (STM32F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products.

2.20 No-Stretch GetCheckSum command

This command is used to compute the CRC value (based on the CRC IP) of a given flash memory range, defined by the memory offset and size.

When the bootloader receives the command, it transmits the ACK byte to the application, waits for a 4-byte address (byte 1 is the MSB, byte 4 is the LSB) and a checksum byte, and then it checks the received address. If the address is valid, within the flash memory, and the checksum is correct, the bootloader transmits an ACK byte; otherwise, it transmits an NACK byte and aborts the command.

If the address is valid, within the flash memory, and the checksum is correct, the bootloader waits for the size of the memory range (4 bytes, byte 1 is the MSB, byte 4 is the LSB) for the checksum calculations, and for a checksum byte. If the size is different from 0, a multiple of four, resulting in an address in the flash when added to the address, and the checksum is correct, the bootloader transmits an ACK to the application; otherwise, it transmits an NACK byte and the aborts the command.

If the address and the size are valid, the application waits for the CRC computation. A Busy state (0x76) is sent while the operation is ongoing.

At the end of the command, if the GetCheckSum operation is successful, the bootloader transmits the checksum result (4-byte, MSB first) and its checksum.

The bootloader uses the CRC IP with the default HW configuration, as described in the reference manual. The CRC internal calculation is done word (U32) by word. Every word (byte1 is the LSB, byte 4 the MSB) is read by the CPU and written to the CRC_DR register for the calculation. Respect the right endianess when simulating the CRC behavior.

Figure 40. No-Stretch GetCheckSum command: host side

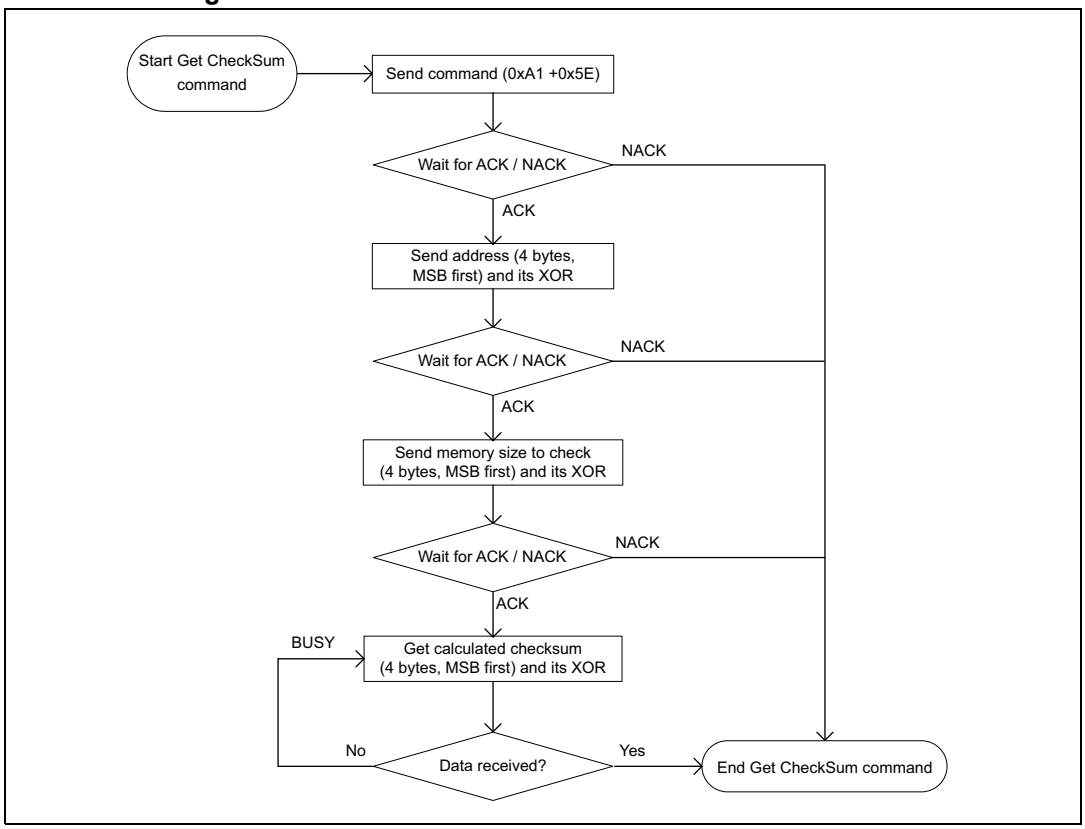
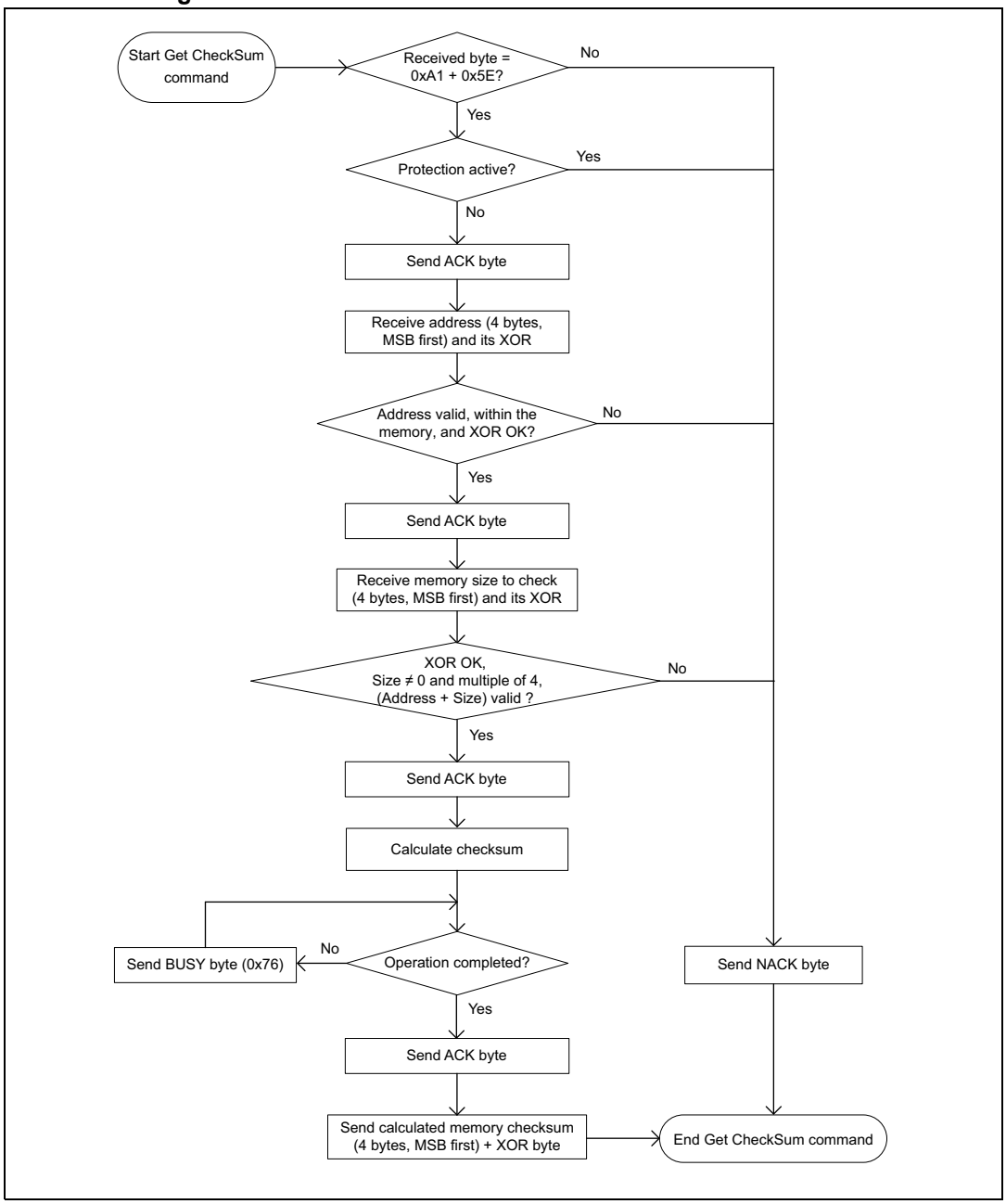


Figure 41. No-Stretch GetCheckSum command: device side



3 Bootloader protocol version evolution

Table 3 lists the bootloader versions.

Table 3. Bootloader protocol versions

Version	Description
V1.0	Initial protocol version.
V1.1	This version implements new I2C commands: – No-Stretch Write Memory – No-Stretch Erase Memory – No-Stretch Write Protect – No-Stretch Write Unprotect – No-Stretch ReadOut Protect – No-Stretch ReadOut Unprotect
V1.2	This version implements the new I2C command No-Stretch Get Memory CheckSum.
V2.0	The number of commands can vary on STM32 devices with the same protocol version v2.0. To know the supported commands, use Get command.

Revision history

Table 4. Document revision history

Date	Revision	Changes
18-Jan-2013	1	Initial release.
02-May-2014	2	Updated list of Applicable products in *Table 1*. Updated set of commands in *Table 2*. Updated *Section 2: Bootloader command set*. Added *Section 2.12*, *Section 2.13*, *Section 2.16*, *Section 2.17*, *Section 2.18* and *Section 2.19*. Added new Protocol version in *Table 3*.
08-Oct-2015	3	Updated *Introduction*, *Section 2: Bootloader command set*, *Section 2.7: Erase Memory command* and *Section 2.13: No-Stretch Erase Memory command*. Updated *Table 1: Applicable products* and *Table 2: I2C bootloader commands*. Updated *Figure 14: Erase Memory command: host side*, *Figure 15: Erase Memory command: device side*, *Figure 26: No-Stretch Erase Memory command: host side* and *Figure 27: No-Stretch Erase Memory command: device side*.
19-Oct-2016	4	Updated *Introduction* and *Table 1: Applicable products*.
15-Mar-2017	5	Updated *Table 1: Applicable products*.
15-Jan-2019	6	Updated *Table 1: Applicable products*. Updated *Section 1: I2C bootloader code sequence*. Minor text edits across the whole document.
05-Apr-2019	7	Updated *Table 1: Applicable products*.
23-Sep-2019	8	Updated *Table 1: Applicable products*.
27-Nov-2019	9	Updated *Table 1: Applicable products*. Updated *Figure 25: No-Stretch Write Memory command: device side*, *Figure 27: No-Stretch Erase Memory command: device side* and *Figure 32: No-Stretch Write Protect command: host side*.
11-Jun-2021	10	Updated *Table 2: I2C bootloader commands*. Added *Section 2.14: Special command* and *Section 2.15: Extended Special command*. Updated *Figure 13: Write Memory command: device side*, *Figure 17: Write Protect command: device side*, *Figure 19: Write Unprotect command: device side*, *Figure 21: Readout Protect command: device side*,, *Figure 23: Readout Unprotect command: device side*, *Figure 25: No-Stretch Write Memory command: device side*, *Figure 33: No-Stretch Write Protect command: device side Figure 35: No-Stretch Write Unprotect command: device side*, *Figure 37: No-Stretch Readout Protect command: device side* and *Figure 39: No-Stretch Readout Unprotect command: device side* and their footnotes. Minor text edits across the whole document.

Table 4. Document revision history (continued)

Date	Revision	Changes
09-Feb-2022	11	Updated *Table 1: Applicable products*, *Table 2: I2C bootloader commands* and *Table 3: Bootloader protocol versions*. Updated *Section 2.1: Get command*. Added *Section 2.20: No-Stretch GetCheckSum command*. Minor text edits across the whole document.
14-Feb-2023	12	Updated *Introduction*, *Section 2.1: Get command*, and *Section 2.2: Get Version command*. Updated *Table 1: Applicable products*, *Table 2: I2C bootloader commands* and its footnotes, and *Table 3: Bootloader protocol versions*. Updated *Figure 2: Get command: host side*, *Figure 3: Get command: device side*, *Figure 4: Get Version: host side*, *Figure 5: Get Version: device side*, *Figure 9: Read Memory command: device side*, *Figure 11: Go command: device side*, *Figure 13: Write Memory command: device side*, *Figure 14: Erase Memory command: host side*, *Figure 15: Erase Memory command: device side*, *Figure 17: Write Protect command: device side*, *Figure 19: Write Unprotect command: device side*, *Figure 25: No-Stretch Write Memory command: device side*, *Figure 27: No-Stretch Erase Memory command: device side*, *Figure 33: No-Stretch Write Protect command: device side*, and *Figure 35: No-Stretch Write Unprotect command: device side*. Minor text edits across the whole document.
06-Mar-2024	13	Added STM32U0 and STM32WBA series, hence updated *Table 1: Applicable products*. Updated footnote *2* of *Table 2: I2C bootloader commands*. Minor text edits across the whole document.
18-Feb-2025	14	Added STM32U3 series, hence updated *Table 1: Applicable products*. Replaced master/slave with controller/target. Updated *Section 2.6: Write Memory command*, *Section 2.7: Erase Memory command*, *Section 2.12: No-Stretch Write Memory command*, *Section 2.13: No-Stretch Erase Memory command*, and *Section 2.20: No-Stretch GetCheckSum command*. Updated figures *13* to *15*, *25* to *27*, and their footnotes. Minor text edits across the whole document.
28-Jan-2026	15	Added *Section 1: I2C bootloader code sequence*. Updated document title.
23-Sep-2026	16	Added STM32G5 to *Table 1: Applicable products*. Removed section General information as it is repeated in *Section 1: I2C bootloader code sequence*.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved