AN3155
Application note

USART protocol used in the STM32 bootloader

Introduction

This application note describes the USART protocol used in the STM32 microcontroller bootloader, providing details on each supported command.

This document applies to STM32 products embedding any bootloader version, as specified in application note AN2606 *STM32 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 USART hardware resources and requirements for your device bootloader, refer to the already mentioned AN2606.

Table 1. Applicable products

Type	Product series
Microcontrollers	STM32F0 Series STM32F1 Series STM32F2 Series STM32F3 Series STM32F4 Series STM32F7 Series STM32G0 Series STM32G4 Series STM32H7 Series STM32L0 Series STM32L1 Series STM32L4 Series STM32L5 Series STM32U5 Series STM32WB Series STM32WL Series

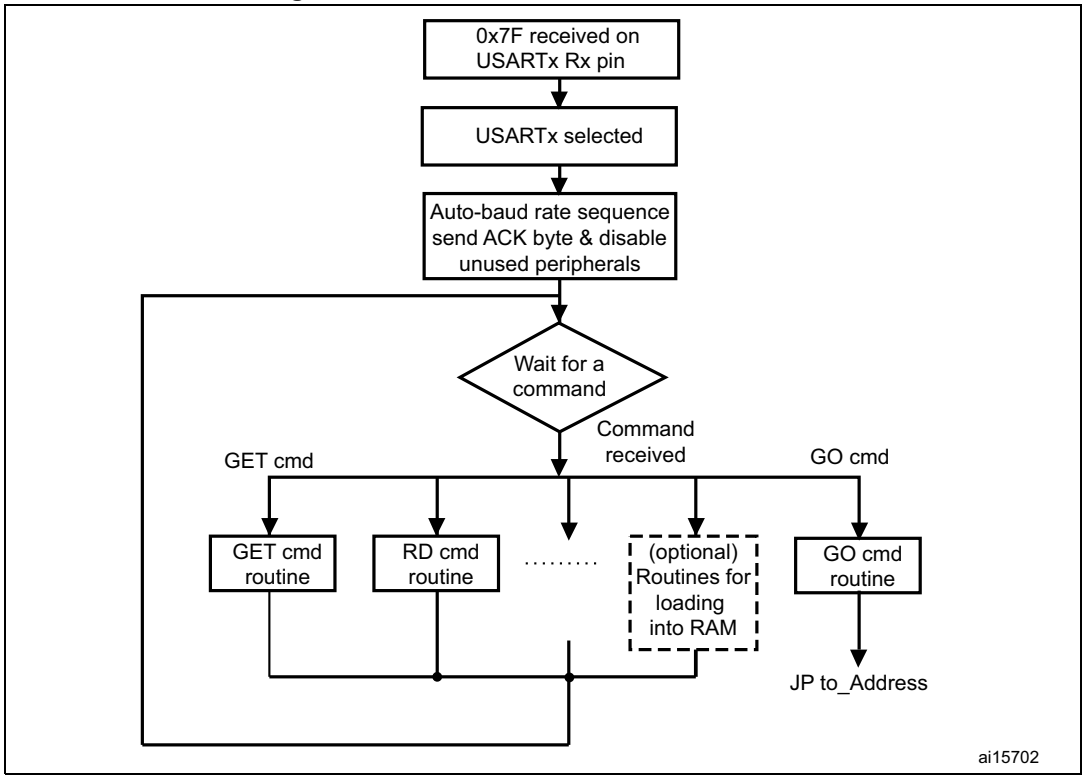
Contents

List of tables

List of figures

1 USART bootloader code sequence

Figure 1. Bootloader for STM32 with USART



Once the system memory boot mode is entered and the STM32 microcontroller (based on Arm®(a) cores) has been configured (for more details refer to AN2606) the bootloader code begins to scan the USARTx_RX line pin, waiting to receive the 0x7F data frame: a start bit, 0x7F data bits, even parity bit and a stop bit.

Depending on the USART IP, the baudrate detection is based on the HW (IP supporting auto baudrate) or on the SW. The following paragraphs explain the SW detection mode.

The duration of this data frame is measured using the Systick timer. The count value of the timer is then used to calculate the corresponding baud rate factor with respect to the current system clock.

Next, the code initializes the serial interface accordingly. Using this calculated baud rate, an acknowledge byte *(*0x79*)* is returned to the host, which signals that the STM32 is ready to receive commands.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Choosing the USARTx baud rate

The calculation of the serial baud rate for USARTx, from the length of the first received byte, is used to operate the bootloader within a wide range of baud rates. However, the upper and lower limits have to be kept, to ensure proper data transfer.

For a correct data transfer from the host to the microcontroller, the maximum deviation between the internal initialized baud rate for USARTx and the real baud rate of the host must be below 2.5%. The deviation (f_B, in percent) between the host baud rate and the microcontroller baud rate can be calculated using the formula below:

$$f_B = \left| \frac{\text{STM32 baud rate} - \text{Host baud rate}}{\text{STM32 baud rate}} \right| \times 100\% \text{ , where } f_B \leq 2.5\%.$$

This baud rate deviation is a nonlinear function, depending upon the CPU clock and the baud rate of the host. The maximum of the function (f_B) increases with the host baud rate. This is due to the smaller baud rate prescale factors, and the implied higher quantization error.

2.1 Minimum baud rate

The lowest tested baud rate (B_{Low}) is 1200. Baud rates below B_{Low} cause SysTick timer overflow. In this event, USARTx is not correctly initialized.

2.2 Maximum baud rate

B_{High} is the highest baud rate for which the deviation does not exceed the limit. All baud rates between B_{Low} and B_{High} are below the deviation limit.

The highest tested baud rate (B_{High}) is 115200.

3 Bootloader command set

The supported commands are listed in *Table 2*. Each command is described in this section.

Table 2. USART bootloader commands

Command(1)	Code	Description
Get(2)	0x00	Gets the version and the allowed commands supported by the current version of the bootloader.
Get Version & Read Protection Status(2)	0x01	Gets the bootloader version and the Read protection status of the Flash memory.
Get ID(2)	0x02	Gets the chip ID.
Read Memory(3)	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 or in the SRAM.
Write Memory(3)	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the application.
Erase(3)(4)	0x43	Erases from one to all the Flash memory pages.
Extended Erase(3)(4)	0x44	Erases from one to all the Flash memory pages using two byte addressing mode (available only for v3.0 USART bootloader versions and above).
Special	0x50	Generic command that allows 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 the write protection for some sectors.
Write Unprotect	0x73	Disables the write protection for all Flash memory sectors.
Readout Protect	0x82	Enables the read protection.
Readout Unprotect(2)	0x92	Disables the read protection.
Get Checksum	0xA1	Computes a CRC value on a given memory area with a size multiple of 4 bytes.

1. If a denied command is received or an error occurs during the command execution, the bootloader sends NACK byte and goes back to command checking.
2. Read protection. When the RDP (Read protection) option is active, only this limited subset of commands is available. All other commands are NACK-ed and have no effect on the device. Once the RDP has been removed, the other commands become active.
3. Refer to STM32 product datasheets and to AN2606 to know the valid memory areas for these commands.
4. Erase (x043) and Extended Erase (0x44) are exclusive. A device can support either the Erase command or the Extended Erase command, but not both.

Communication safety

All communication from the programming tool (PC) to the device is verified by:

1. 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.

2. For each command the host sends a byte and its complement (XOR = 0x00).
3. UART: parity check active (even parity).

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

- ACK = 0x79
- NACK = 0x1F

3.1 Get command

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

Figure 2. Get command: host side

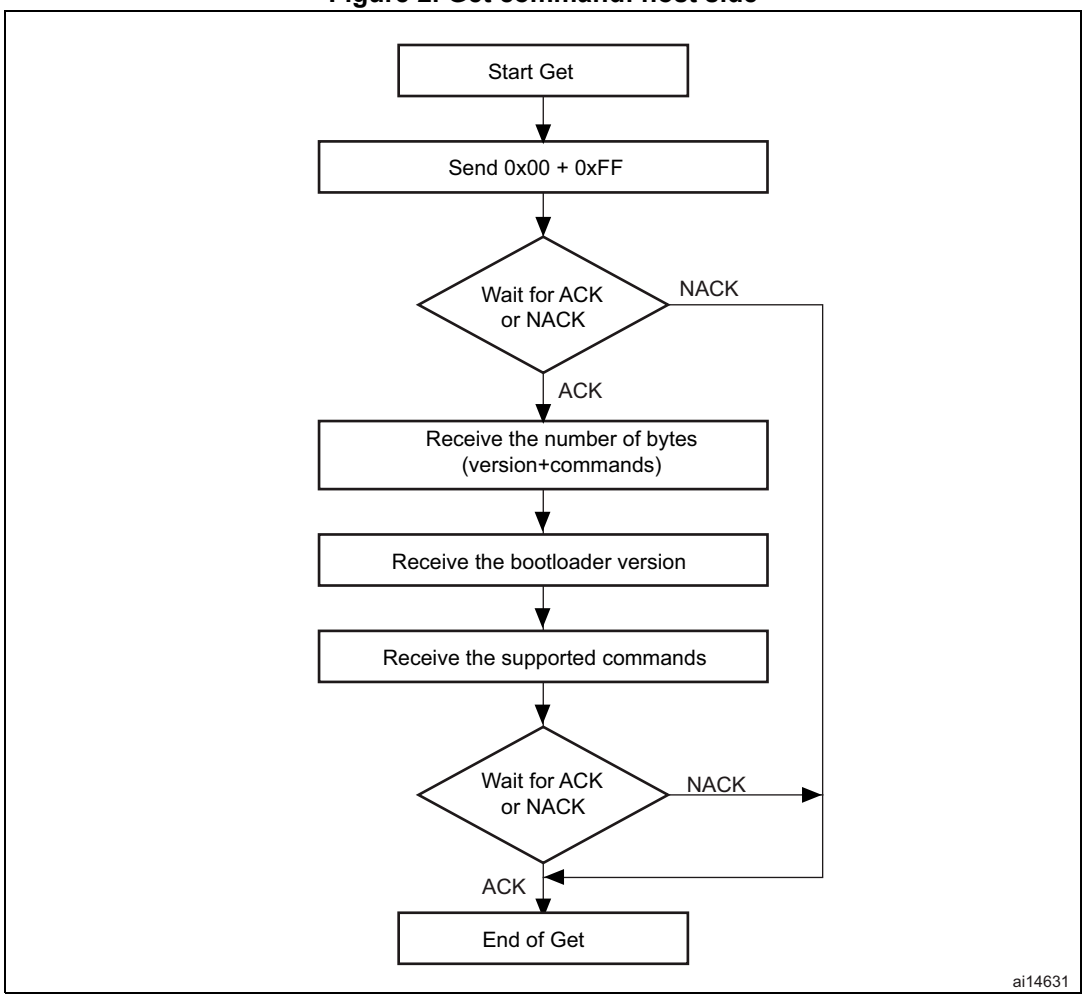
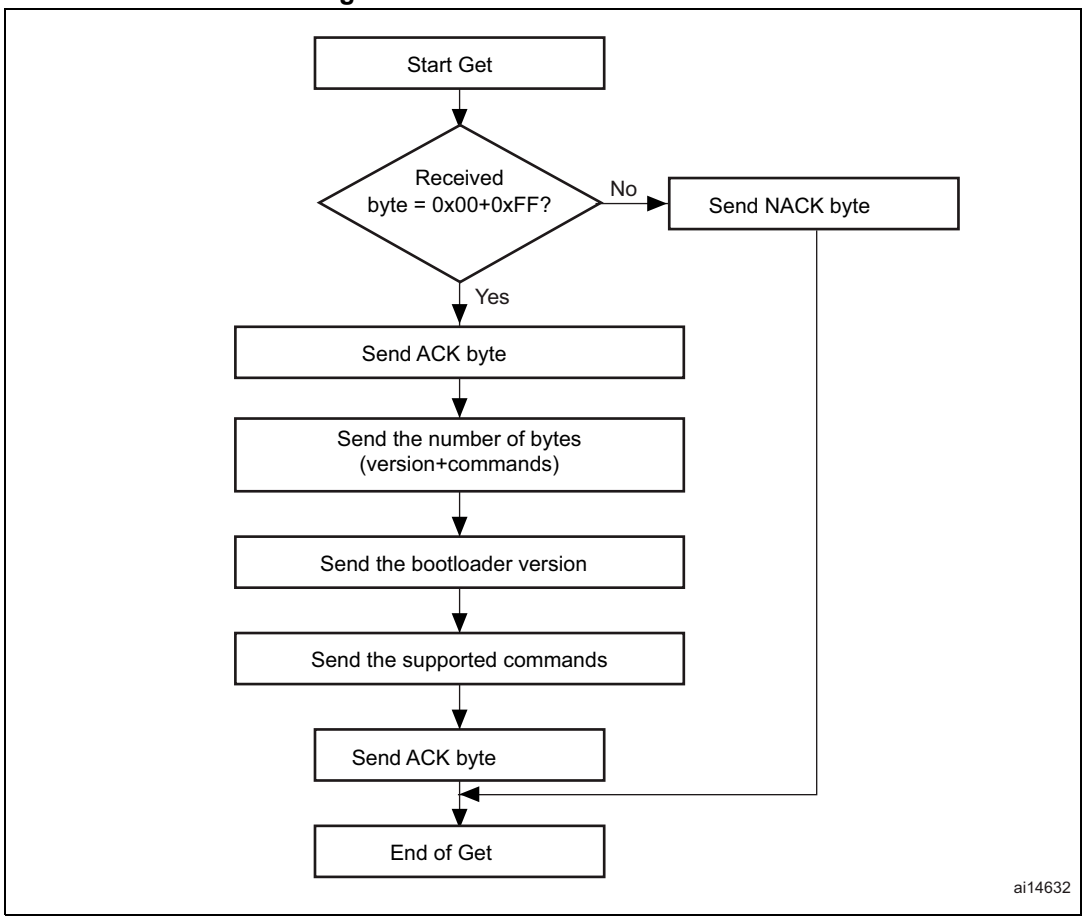


Figure 3. Get command: device side



The STM32 sends the bytes as follows:

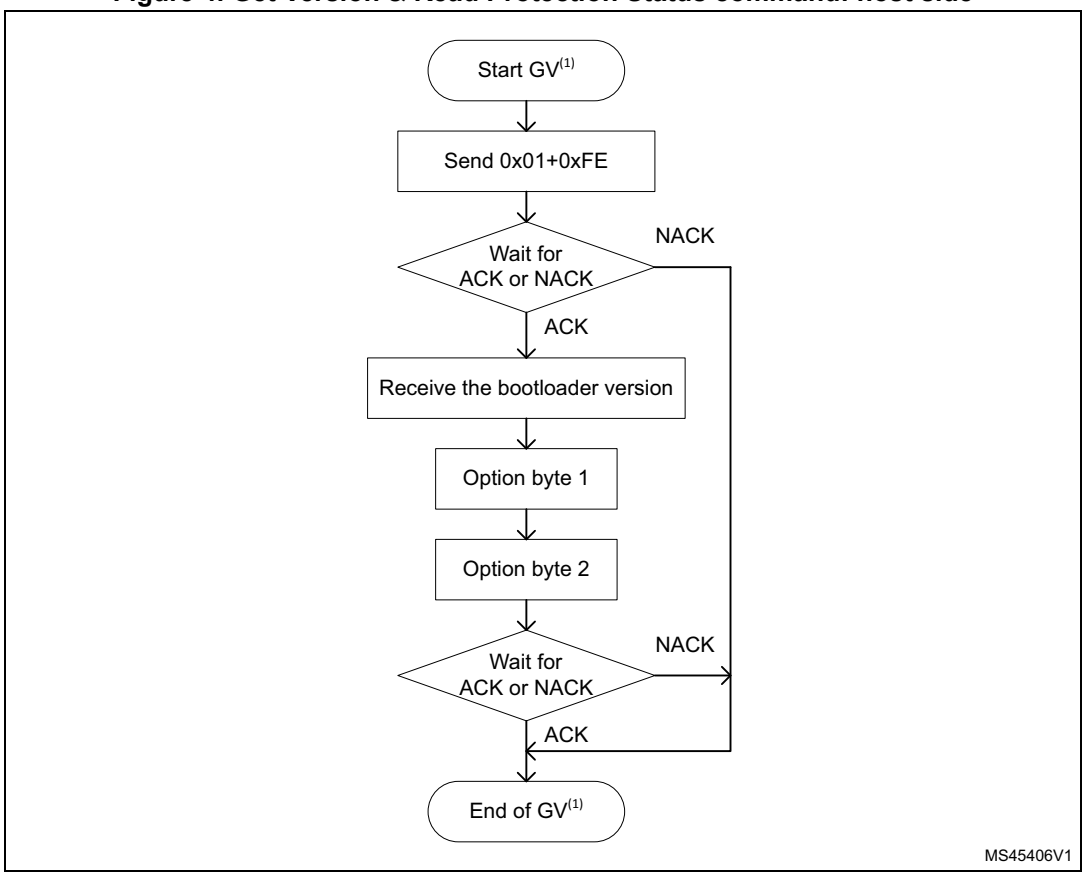
Byte 1:	ACK	
Byte 2:	N = 11 = the number of bytes to follow – 1 except current and ACKs.	
Byte 3:	Bootloader version (0 < version < 255), example: 0x10 = version 1.0	
Byte 4:	0x00	Get command
Byte 5:	0x01	Get Version and Read Protection Status
Byte 6:	0x02	Get ID
Byte 7:	0x11	Read Memory command
Byte 8:	0x21	Go command
Byte 9:	0x31	Write Memory command
Byte 10:	0x43 or 0x44	Erase command or Extended Erase command (exclusive commands)
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:	0xA1	Get Checksum command (only for version V3.3)

3.2 Get Version & Read Protection Status command

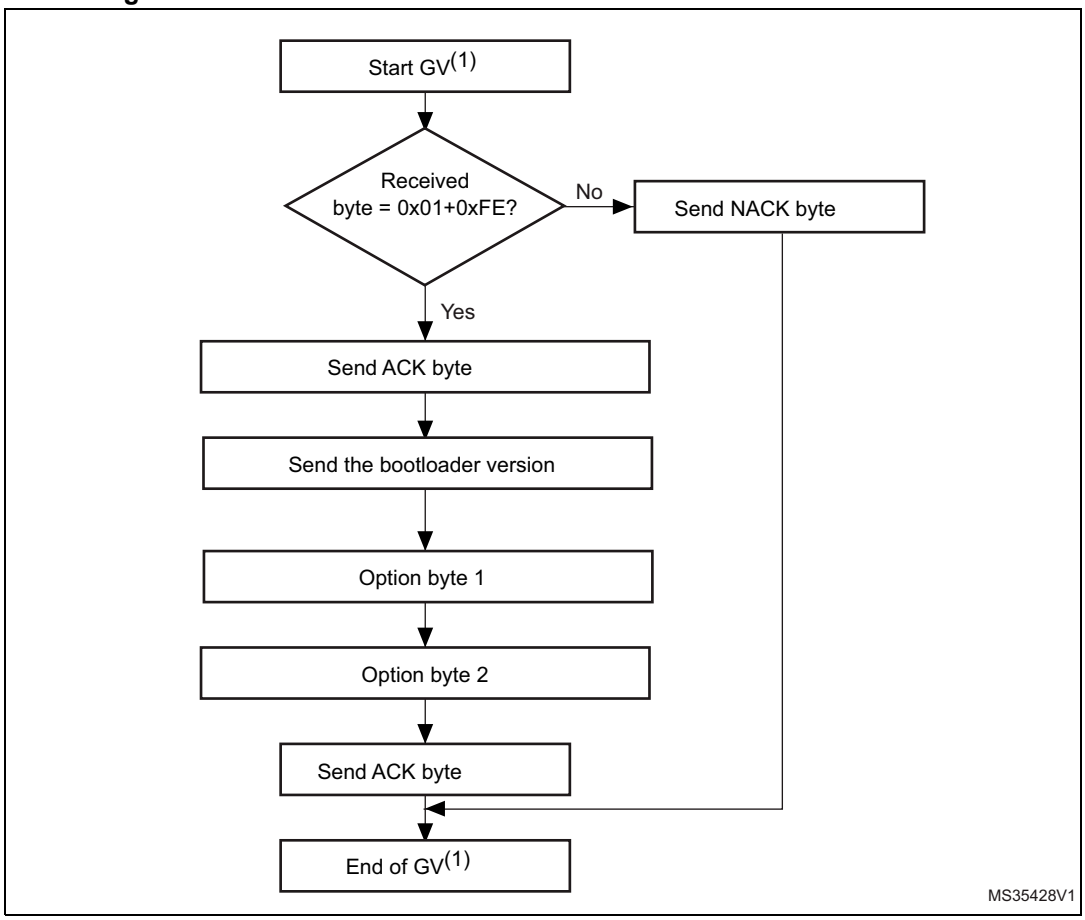
The Get Version & Read Protection Status command is used to get the bootloader version and the read protection status. After receiving the command the bootloader transmits the version, the read protection and number of times it has been enabled and disabled to the host.

Figure 4. Get Version & Read Protection Status command: host side



1. GV = Get Version & Read Protection Status.

Figure 5. Get Version & Read Protection Status command: device side



1. GV = Get Version & Read Protection Status.

The STM32 sends the bytes as follows:

Byte 1: ACK

Byte 2: Bootloader version (0 < version ≤ 255), example: 0x10 = version 1.0

Byte 3: Option byte 1: 0x00 to keep the compatibility with generic bootloader protocol

Byte 4: Option byte 2: 0x00 to keep the compatibility with generic bootloader protocol

Byte 5: ACK

3.3 Get ID command

The Get ID 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

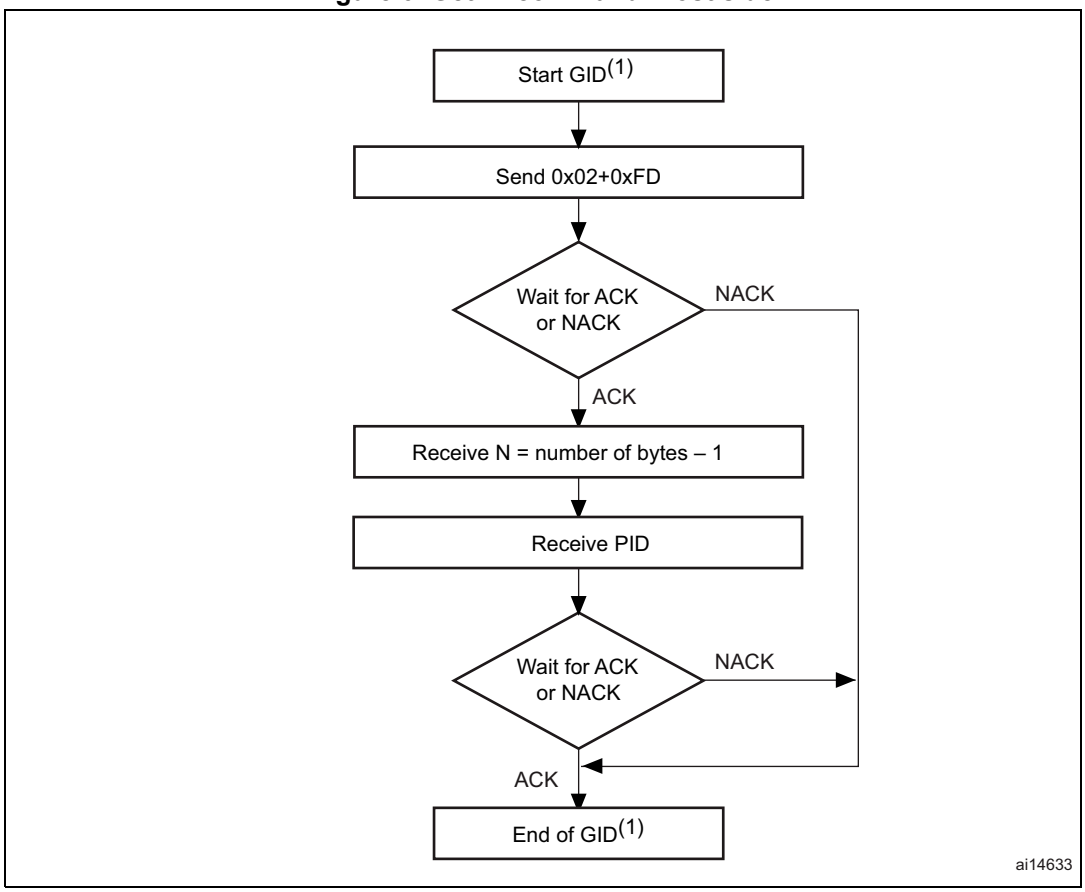
Byte 2: N = the number of bytes – 1 (N = 1 for STM32), except for current byte and ACKs.

Bytes 3-4: PID(1) byte 3 = 0x04, byte 4 = 0xXX

Byte 5: ACK

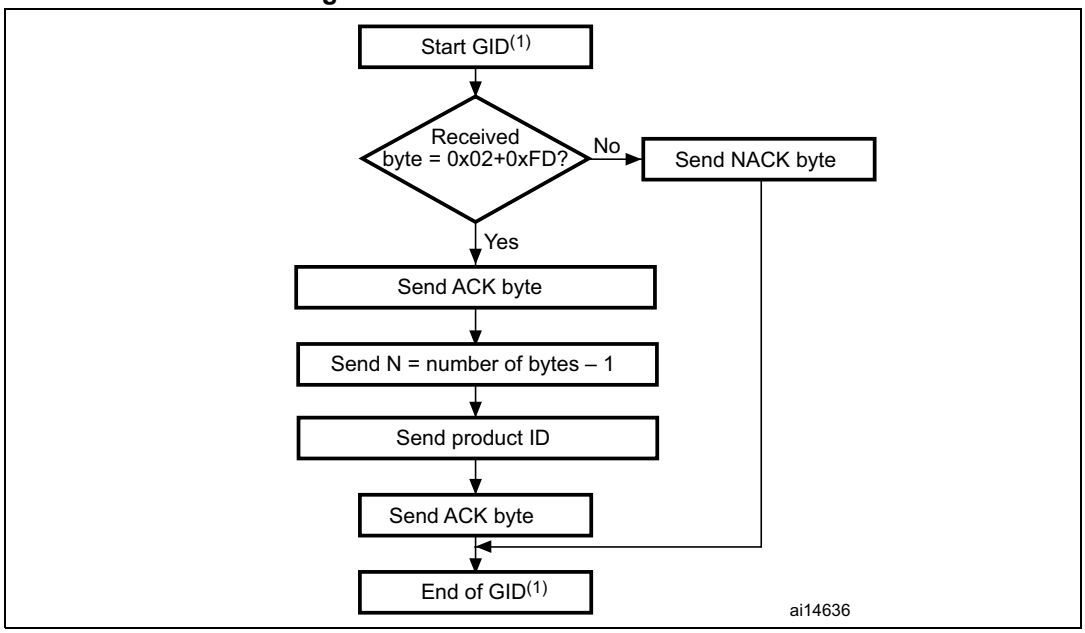
1. PID stands for product ID. Byte 1 is the MSB and byte 2 the LSB of the ID.

Figure 6. Get ID command: host side



1. GID = Get ID.

Figure 7. Get ID command: device side



1. GID = Get ID.

3.4 Read Memory command

The Read Memory command is used to read data from any valid memory address (refer to the product datasheets and to AN2606 for more details) in RAM, Flash memory and the information block (system memory or option byte areas).

When the bootloader receives the Read Memory command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (four bytes, byte 1 is the address MSB and 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 a NACK byte and aborts the command.

When the address is valid and the checksum is correct, the bootloader waits for the number of bytes to be transmitted – 1 (N bytes) and for its complemented byte (checksum). If the checksum is correct it then transmits the needed data ((N + 1) bytes) to the application, starting from the received address. If the checksum is not correct, it sends a NACK before aborting the command.

The host sends bytes to the STM32 as follows:

Bytes 1-2: 0x11 + 0xEE

Wait for ACK

Bytes 3 to 6 Start address byte 3: MSB, byte 6: LSB

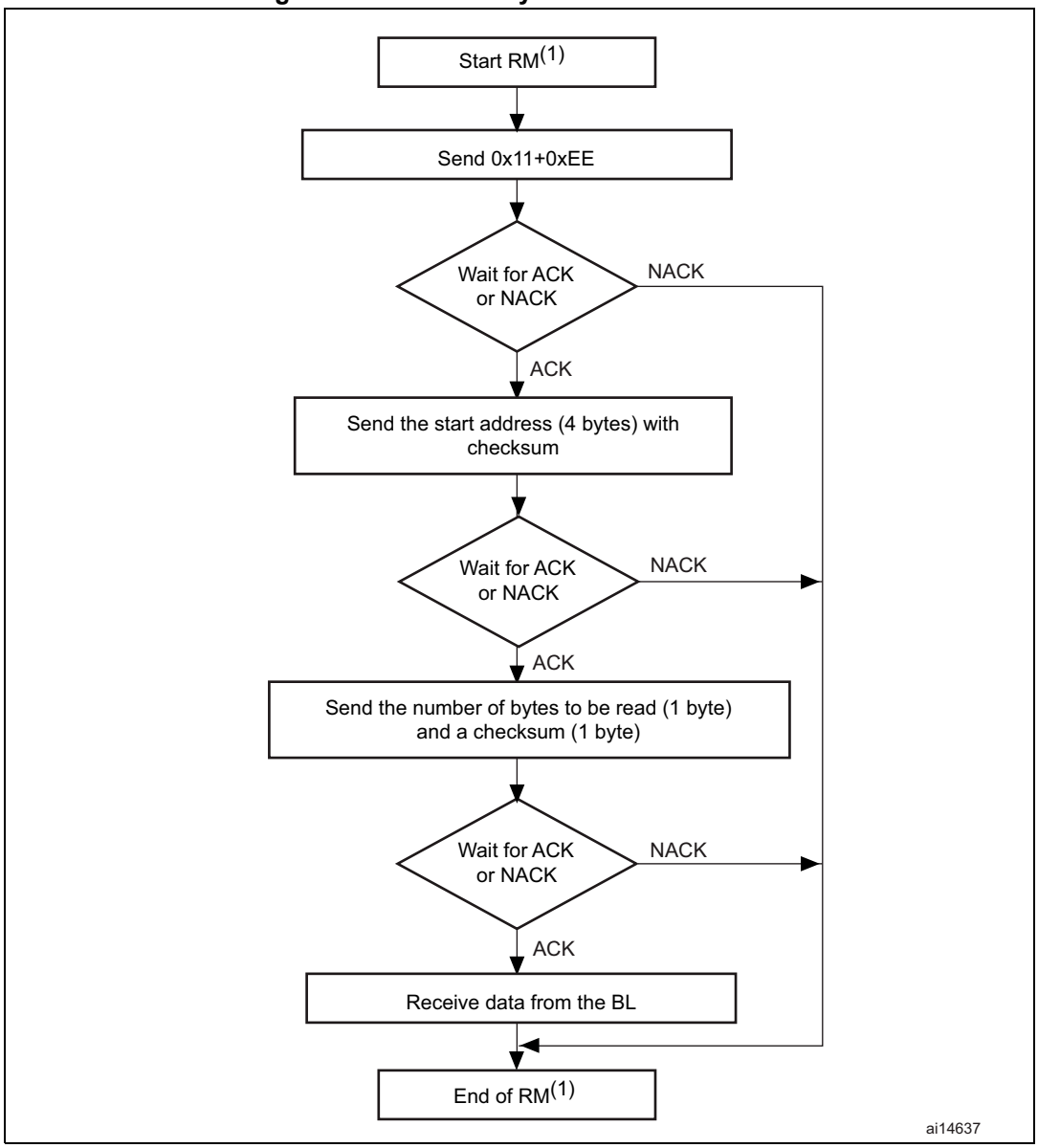
Byte 7: Checksum: XOR (byte 3, byte 4, byte 5, byte 6)

Wait for ACK

Byte 8: The number of bytes to be read – 1 ($0 < N \leq 255$);

Byte 9: Checksum: XOR byte 8 (complement of byte 8)

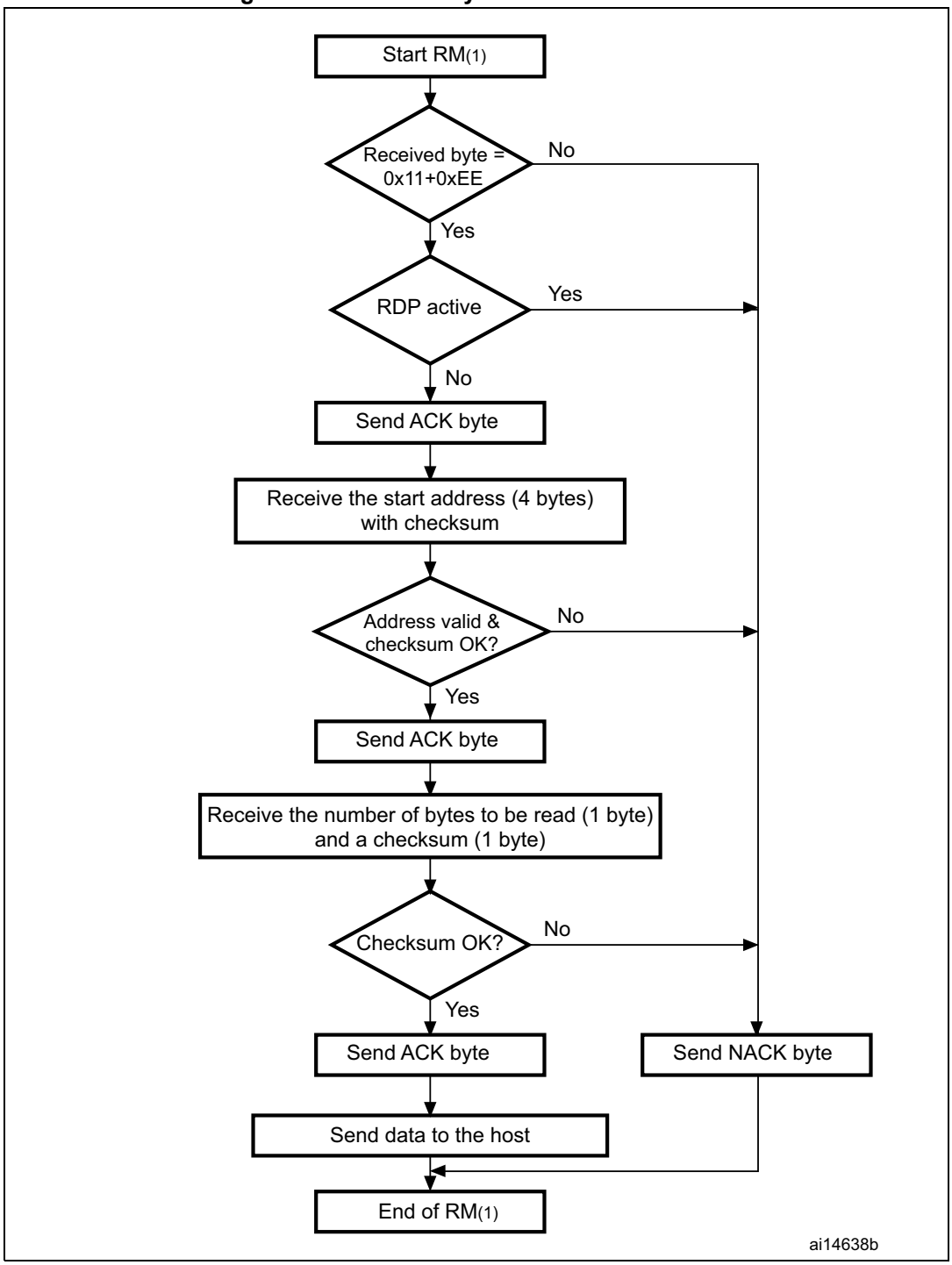
Figure 8. Read Memory command: host side



1. RM = Read Memory.

Note: Some products may return two NACKs instead of one when Read protection (RDP) is active (or Read protection level 1 is active). To know if a given product returns one or two NACKs in this situation, refer to the known limitations section relative to that product in AN2606.

Figure 9. Read Memory command: device side



1. RM = Read Memory.

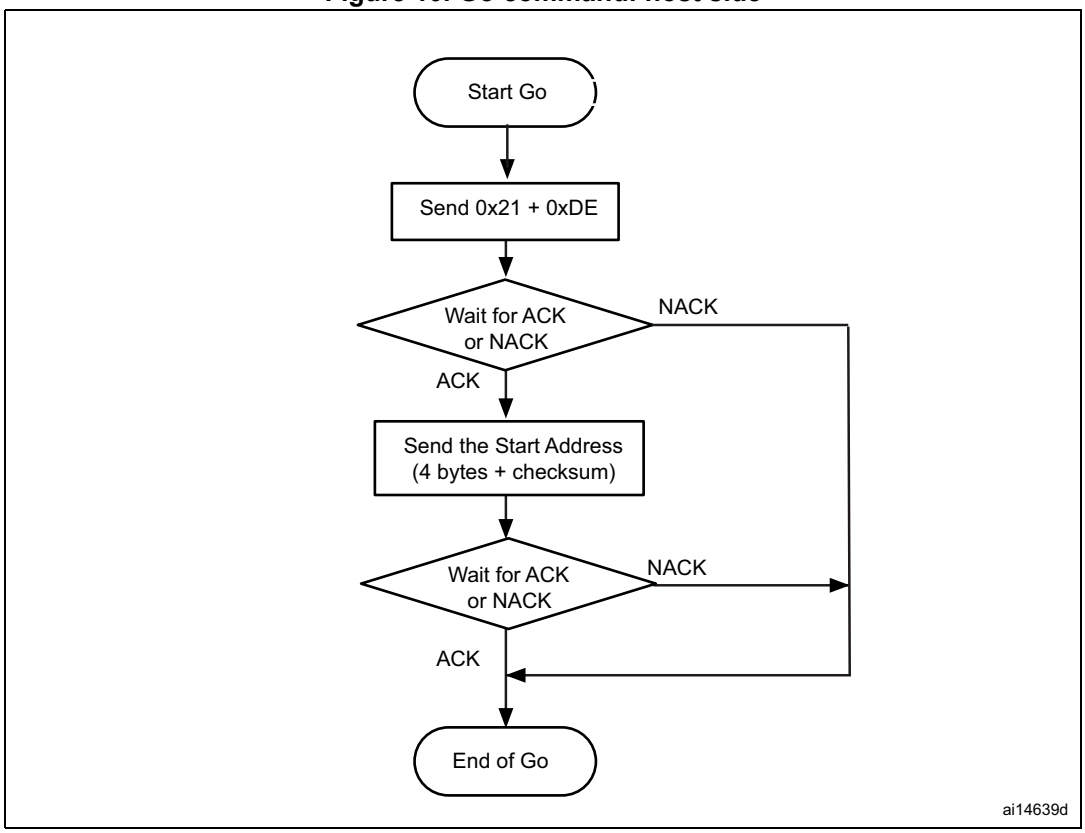
3.5 Go command

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. After the transmission of the ACK byte, the bootloader waits for an address (four bytes, byte 1 is the address MSB and byte 4 is 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 a NACK byte and aborts the command.

When the address is valid and the checksum is correct, the bootloader firmware

- initializes the registers of the peripherals used by the bootloader to their default reset values
- initializes the main stack pointer of the user application
- jumps to the memory location programmed in the received 'address + 4' (corresponding to the address of the application reset handler).
  For 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.

Figure 10. Go command: host side



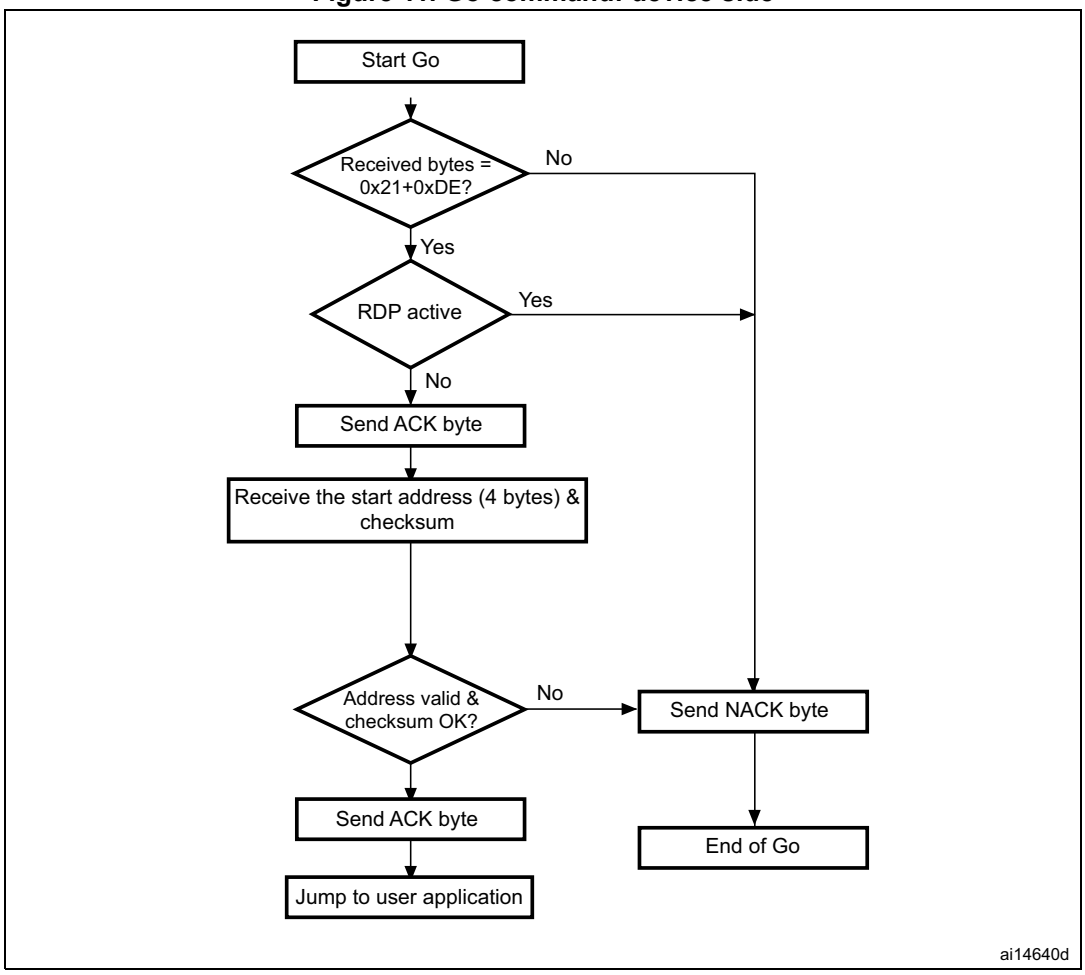
Note: Valid addresses for the Go command are in RAM or Flash memory (refer to STM32 product datasheets and to AN2606 for more details about the valid memory addresses for the used device). All other addresses are considered not valid and are NACK-ed by the device.

When an application is loaded into RAM and then a jump is made to it, the program must be configured to run with an offset to avoid overlapping with the first area used by the bootloader firmware (refer to STM32 product datasheets and to AN2606 for more details about the RAM offset for the used device).

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

Some products may return two NACKs instead of one when Read protection (RDP) is active (or Read protection level 1 is active). To know if a given product returns one or two NACKs in this situation, refer to the known limitations section relative to that product in AN2606.

Figure 11. Go command: device side



The host sends bytes to the STM32 as follows:

Byte 1:	0x21
Byte 2:	0xDE
Wait for ACK	
Bytes 3 to 6:	Start address (byte 3: MSB, byte 6: LSB)
Byte 7:	Checksum: XOR (byte 3, byte 4, byte 5, byte 6)

3.6 Write Memory command

The Write Memory command is used to write data to any valid memory address (see note below) i.e. RAM, Flash memory, or option byte area.

When the bootloader receives the Write Memory command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (four bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte, it then checks the received address. For the option byte area, the start address must be the base address of the option byte area (see note below) to avoid writing inopportunely in this area.

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

- gets a byte, N, which contains the number of data bytes to be received
- receives the user data ((N + 1) bytes) and the checksum (XOR of N and of all data bytes)
- programs the user data to memory starting from the received address
- at the end of the command, if the write operation was successful, the bootloader transmits the ACK byte; otherwise it transmits a NACK byte to the application and aborts the command.

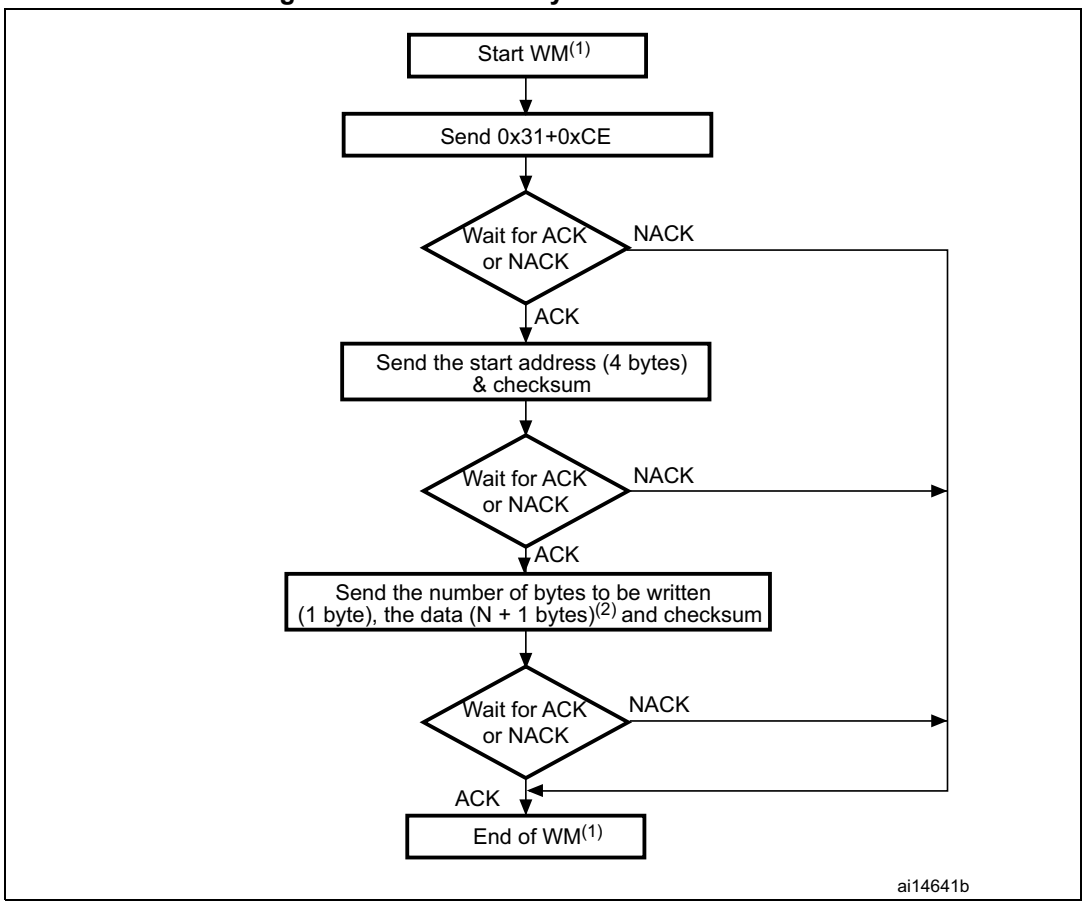
The maximum length of the block to be written for the STM32 is 256 bytes.

If the Write Memory command is issued to the option byte area, all bytes 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: When writing to the RAM, user must not overlap the first area used by the bootloader firmware.

No error is returned when performing write operations on write-protected sectors.No error is returned when the start address is invalid.

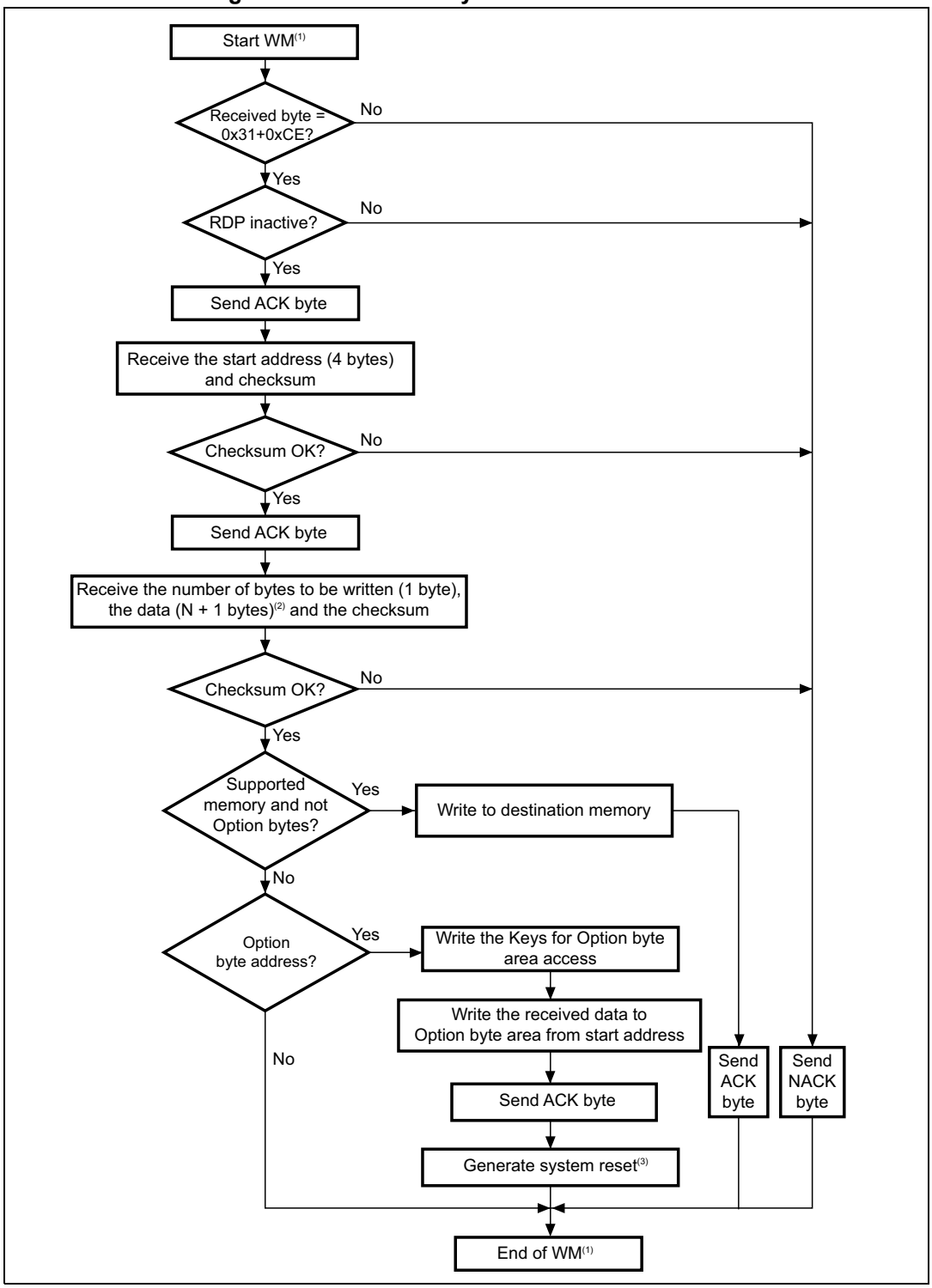
Figure 12. Write Memory command: host side



1. WM = Write Memory.
2. N+1 must be a multiple of 4.

Note: Some products may return two NACKs instead of one when Read protection (RDP) is active (or Read protection level 1 is active). To know if a given product returns one or two NACKs in this situation, refer to the known limitations section relative to that product in AN2606.

Figure 13. Write Memory command: device side



1. WM = Write Memory.
2. N+1 must be a multiple of 4.
3. System reset is called only for some STM32 BL (STM32F0/F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

The host sends the bytes to the STM32 as follows:

Byte 1:	0x31
Byte 2:	0xCE
Wait for ACK	
Bytes 3 to 6:	Start address (byte 3: MSB, byte 6: LSB)
Byte 7:	Checksum: XOR (byte3, byte4, byte5, byte6)
Wait for ACK	
Byte 8:	Number of bytes to be received ($0 < N \leq 255$)
N +1 data bytes:	Max 256 bytes
Checksum byte:	XOR (N, N+1 data bytes)

3.7 Erase Memory command

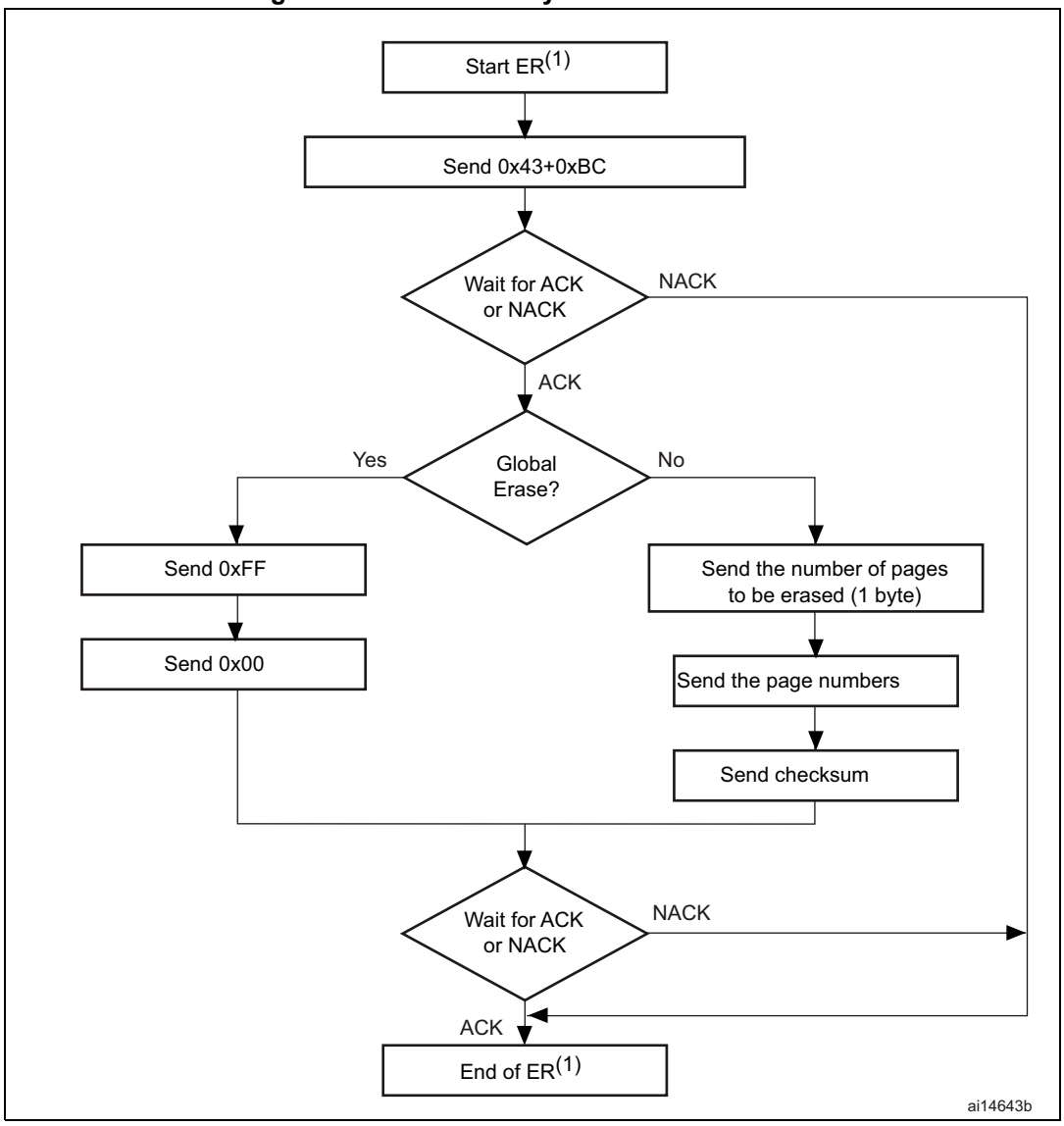
The Erase Memory command allows the host to erase Flash memory pages. When the bootloader receives the Erase Memory command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader receives one byte (number of pages to be erased), the Flash memory page codes and a checksum byte; if the checksum is correct the bootloader erases the memory and sends an ACK byte to the host, otherwise it sends a NACK byte to the host and the command is aborted.

Erase Memory command specifications:

1. The bootloader receives a byte that contains N, the number of pages to be erased – 1. N = 255 is reserved for global erase requests. For $0 \leq N \leq 254$, N + 1 pages are erased.
2. The bootloader receives (N + 1) bytes, each byte containing a page number.

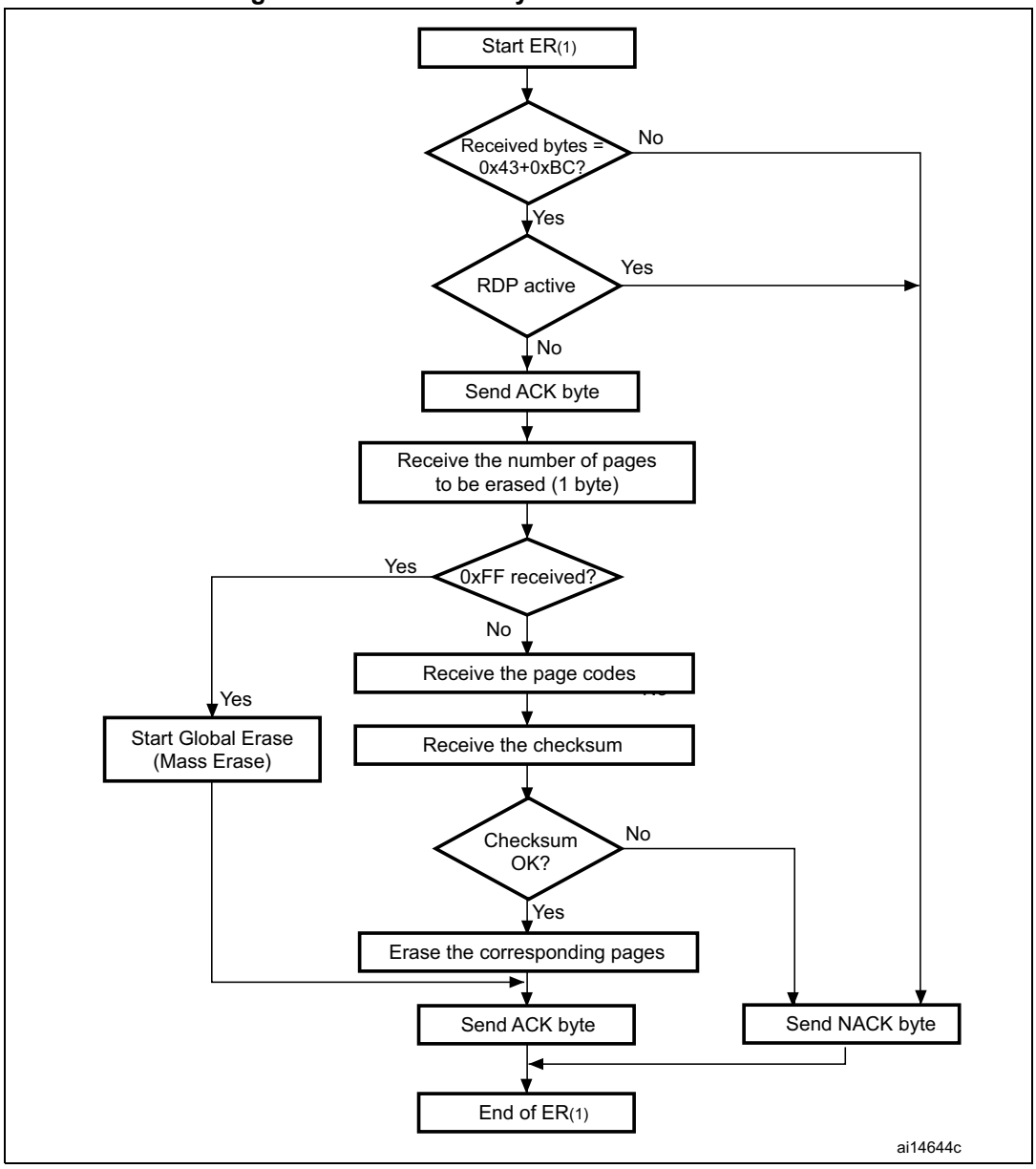
Note: No error is returned when performing erase operations on write protected sectors.

Figure 14. Erase Memory command: host side



1. ER = Erase Memory.

Figure 15. Erase Memory command: device side



1. ER = Erase Memory.

Note: After sending the erase memory command and its checksum, if the host sends 0xFF followed by data different from 0x00, the mass erase is not performed but an ACK is sent by the device.

The host sends bytes to the STM32 as follows:

Byte 1: 0x43

Byte 2: 0xBC

Wait for ACK

Byte 3: 0xFF or number of pages to be erased – 1 ($0 \leq N \leq$ maximum number of pages)

Byte 4: 0x00 (in case of global erase) or ((N + 1 bytes (page numbers) and then checksum XOR (N, N+1 bytes))

3.8 Extended Erase Memory command

The Extended Erase Memory command allows the host to erase Flash memory pages using two bytes addressing mode. When the bootloader receives the Extended Erase Memory command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader receives two bytes (number of pages to be erased), the Flash memory page 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 a NACK byte to the host and the command is aborted.

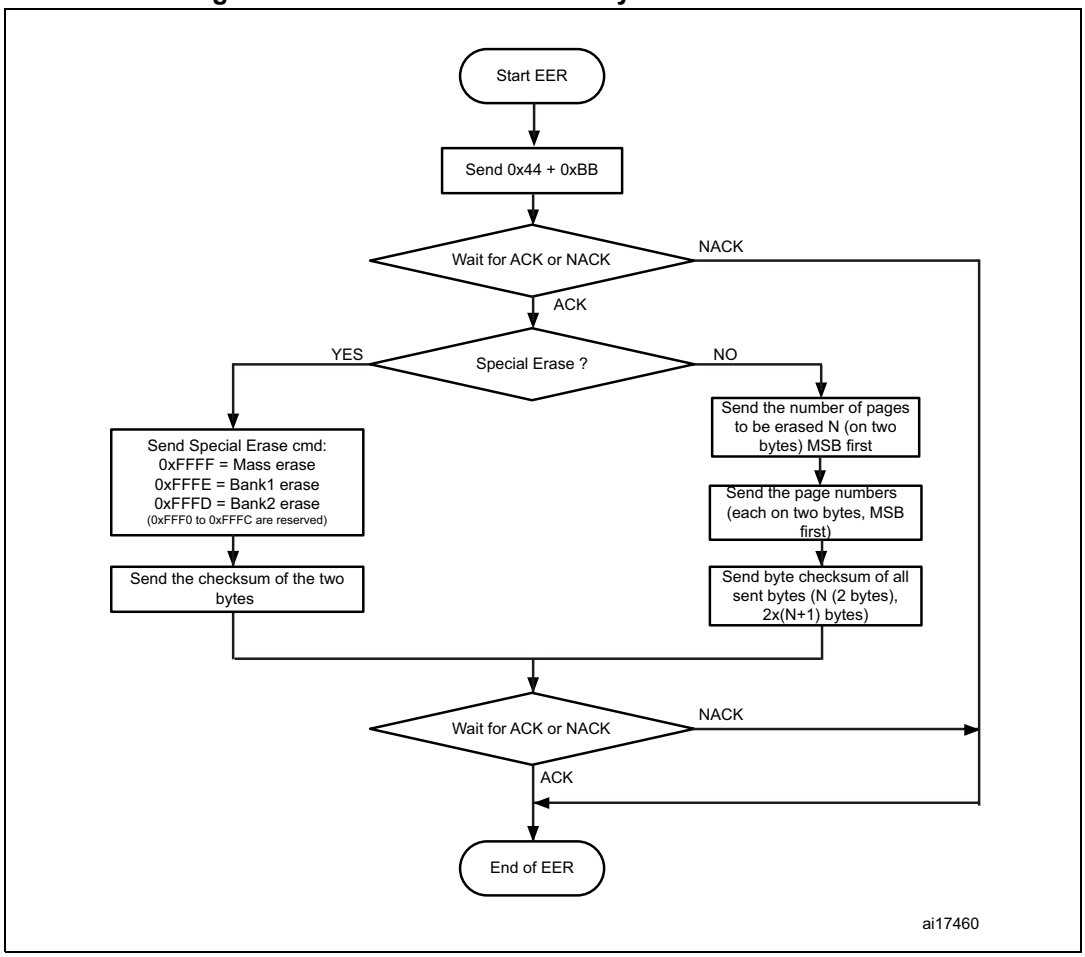
Extended Erase Memory command specifications:

1. The bootloader receives one half-word (two bytes) that contains N, the number of pages to be erased:
   a) For N = 0xFFFY (where Y is from 0 to F) special erase is performed:
      - 0xFFFF for global mass erase
      - 0xFFFE for bank 1 mass erase
      - 0xFFFD for bank 2 mass erase
      - Codes from 0xFFFC to 0xFFF0 are reserved
   b) For other values where $0 \leq N <$ maximum number of pages: N + 1 pages are erased.
2. The bootloader receives:
   a) In the case of a special erase, one byte: checksum of the previous bytes:
      - 0x00 for 0xFFFF
      - 0x01 for 0xFFFE
      - 0x02 for 0xFFFD
   a) In the case of N+1 page erase, the bootloader receives (2 x (N + 1)) bytes, each half-word containing a page number (coded on two bytes, MSB first). Then all previous byte checksums (in one byte).

Note: No error is returned when performing erase operations on write-protected sectors.

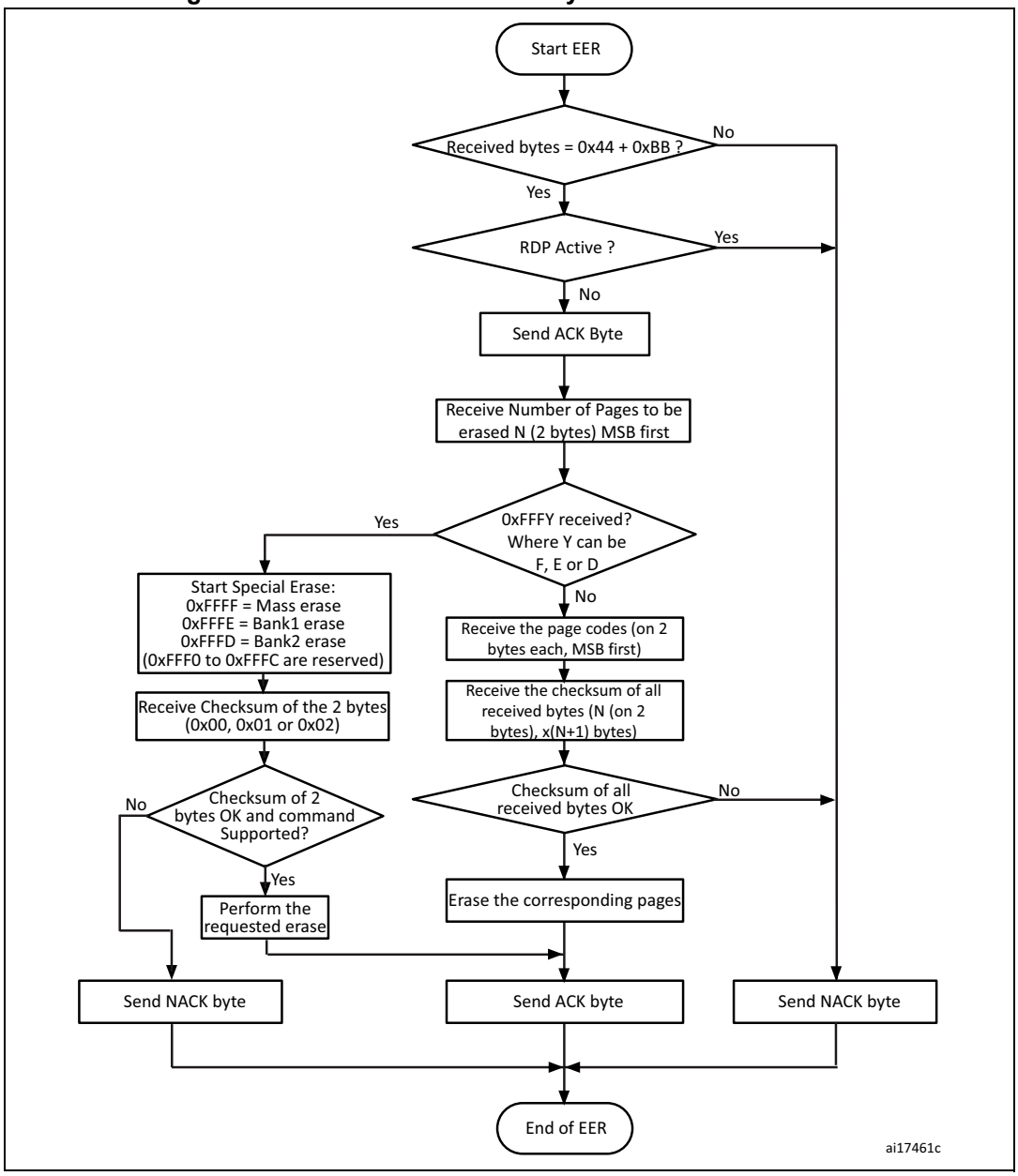
The maximum number of pages is relative to the product and must be respected.

Figure 16. Extended Erase Memory command: host side



1. EER = Extended Erase Memory

Figure 17. Extended Erase Memory command: device side



1. EER = Extended Erase Memory.

The host sends the bytes to the STM32 as follows:

Byte 1: 0x44

Byte 2: 0xBB

Wait for ACK

Bytes 3-4:
– Special erase (0xFFFF, 0xFFFE or 0xFFFD)

OR

– Number of pages to be erased (N+1 where: $0 \leq N <$ Maximum number of pages).

Remaining bytes
– Checksum of Bytes 3-4 in case of special erase (0x00 if 0xFFFFor 0x01 if 0xFFFE or 0x02 if 0xFFFD)

OR

– (2 x (N + 1)) bytes (page numbers coded on two bytes MSB first) and then the checksum for bytes 3-4 and all the following bytes)

3.9 Write Protect command

The Write Protect command is used to enable the write protection for some or all Flash memory sectors. When the bootloader receives the Write Protect command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader 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 into account the new configuration of the option byte.

Note: Refer to STM32 product datasheets and to AN2606 for more details about the sector size for the used device.

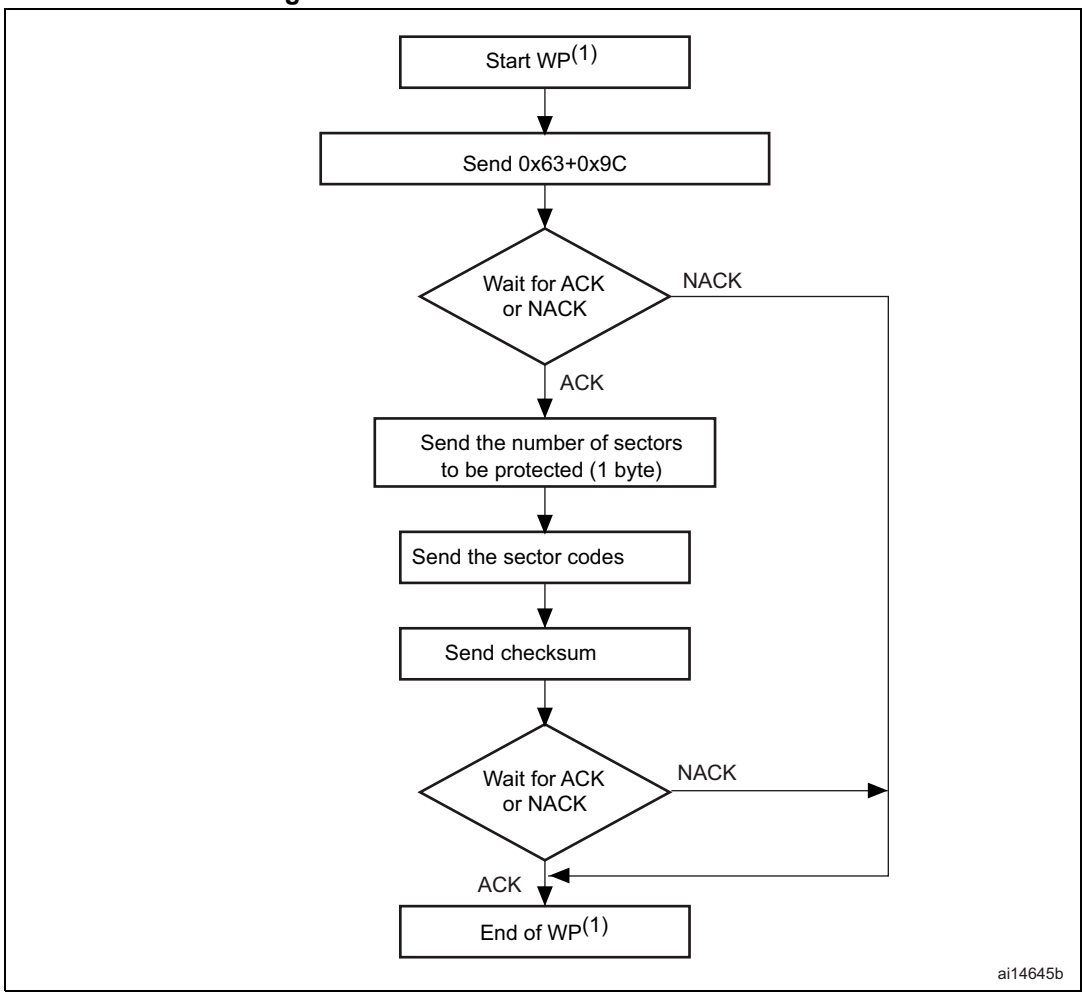
The Write Protect command sequence is as follows:

- 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 byte 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 a wrong number of sectors to be protected or a wrong sector number.

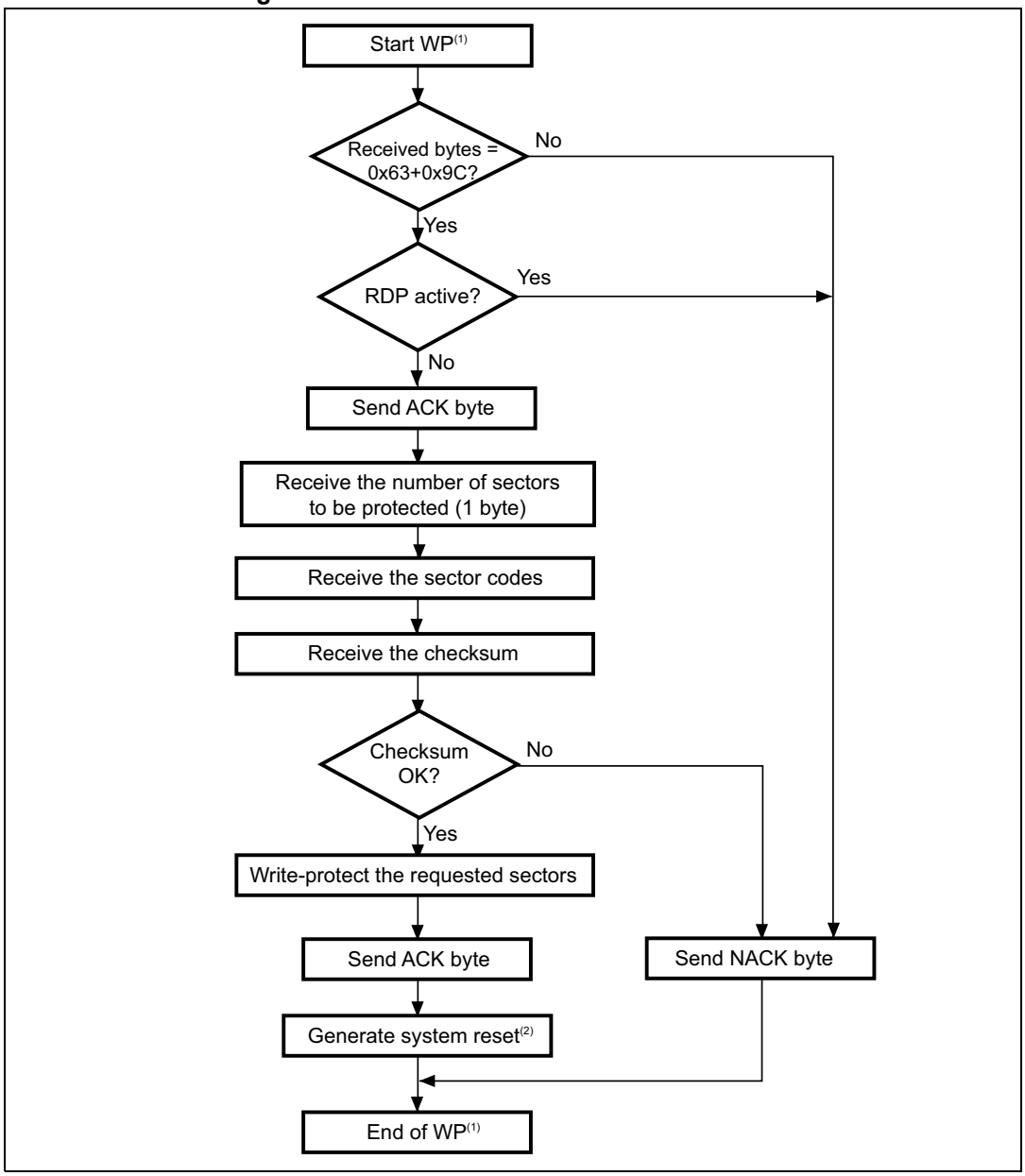
If a second Write Protect command is executed, the Flash memory sectors been protected by the first command become unprotected, and only the sectors passed within the second Write Protect command become protected.

Figure 18. Write Protect command: host side



1. WP = Write Protect.

Figure 19. Write Protect command: device side



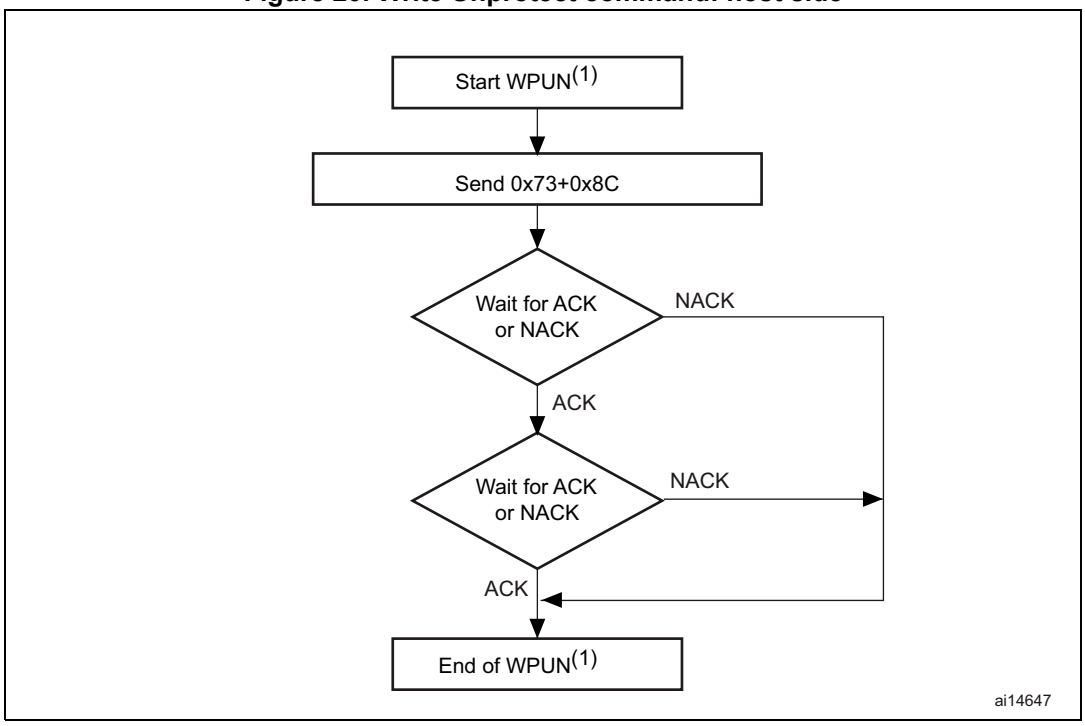
1. WP = Write Protect.
2. System reset is called only for some STM32 BL (STM32F0/F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

3.10 Write Unprotect command

The Write Unprotect command is used to disable the write protection of all the Flash memory sectors. When the bootloader receives the Write Unprotect command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader disables the write protection of all the Flash memory sectors. After the unprotection operation the bootloader transmits the ACK byte.

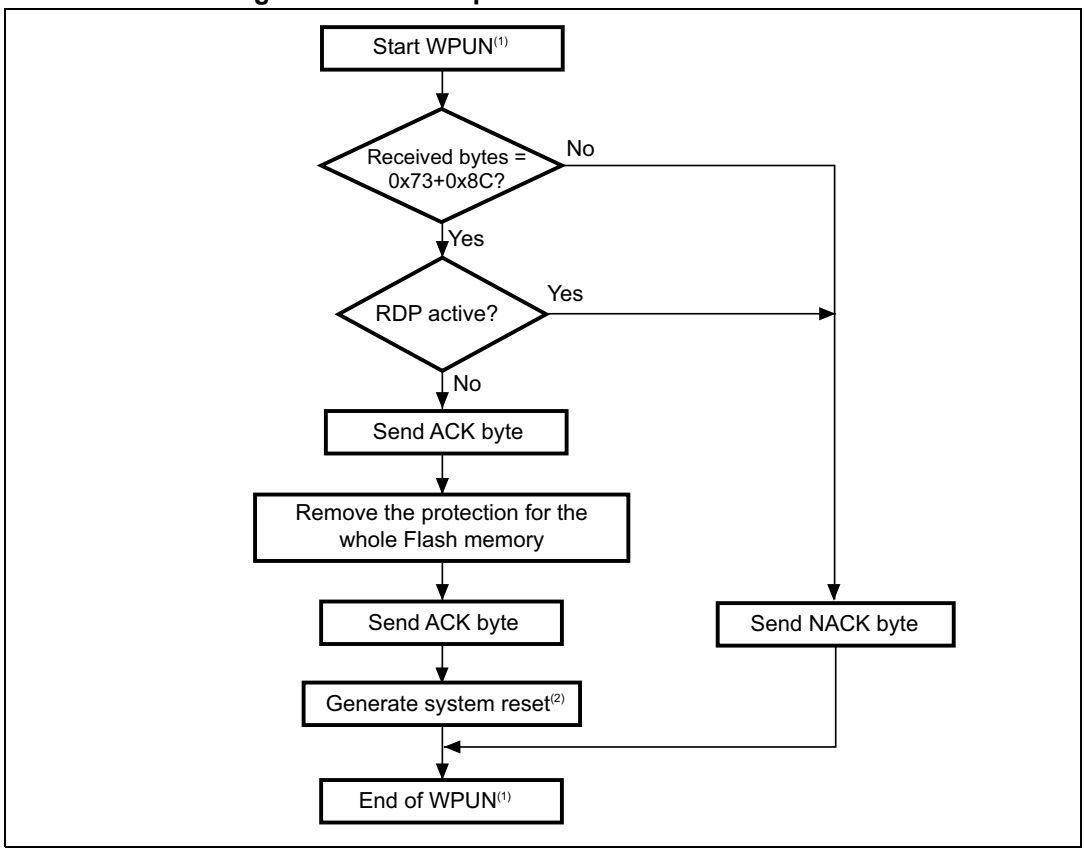
At the end of the Write Unprotect command, the bootloader transmits the ACK byte and generates a system reset to take into account the new configuration of the option byte.

Figure 20. Write Unprotect command: host side



1. WPUN = Write Unprotect.

Figure 21. Write Unprotect command: device side



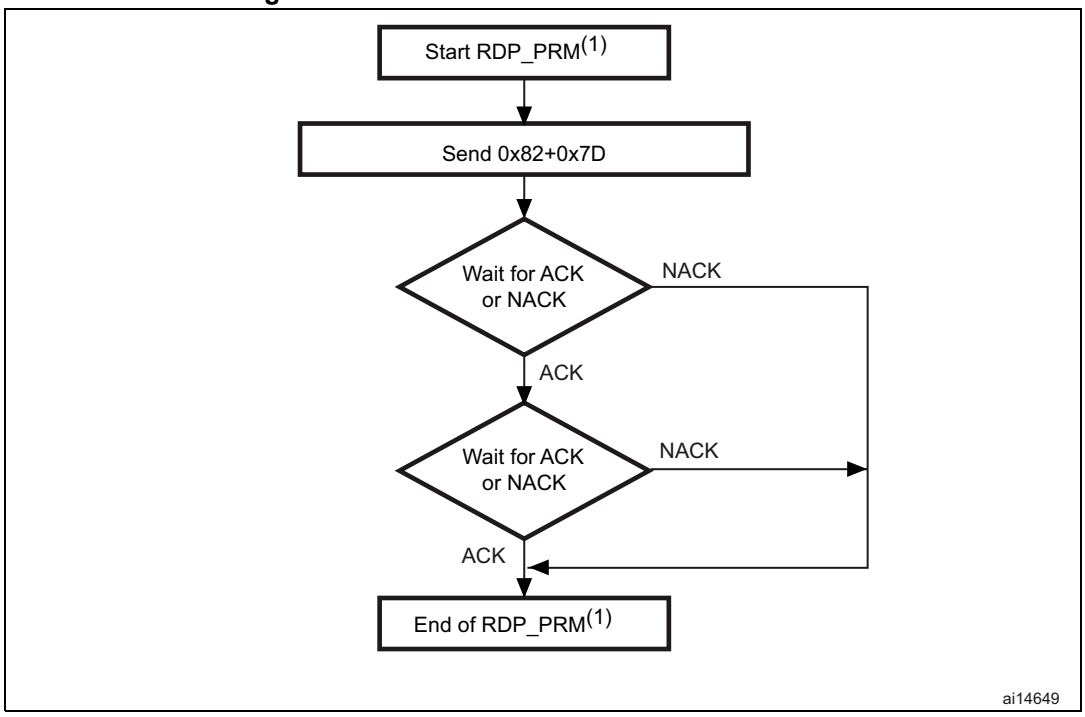
1. WPUN = Write Unprotect.
2. System reset is called only for some STM32 BL (STM32F0/F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

3.11 Readout Protect command

The Readout Protect command is used to enable the Flash memory read protection. When the bootloader receives the Readout Protect command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader enables the read protection for the Flash memory.

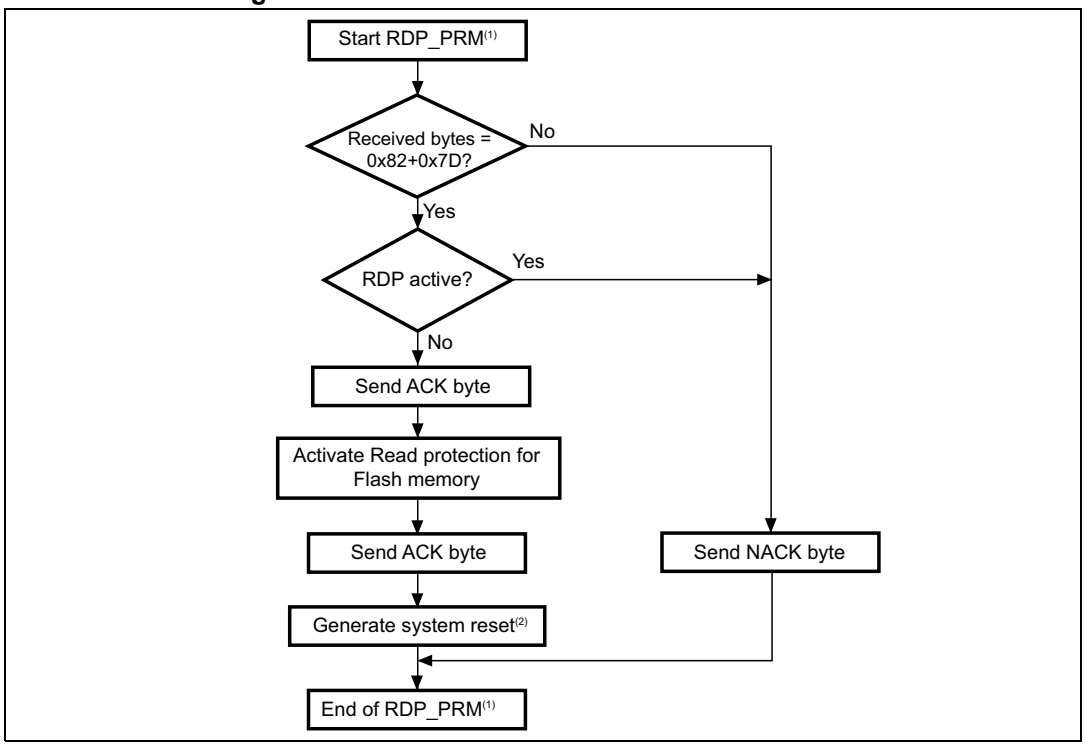
At the end of the Readout Protect command, the bootloader transmits the ACK byte and generates a system reset to take into account the new configuration of the option byte.

Figure 22. Readout Protect command: host side



1. RDP_PRM = Readout Protect.

Figure 23. Readout Protect command: device side



1. RDP_PRM = Readout Protect.
2. System reset is called only for some STM32 BL (STM32F0/F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

3.12 Readout Unprotect command

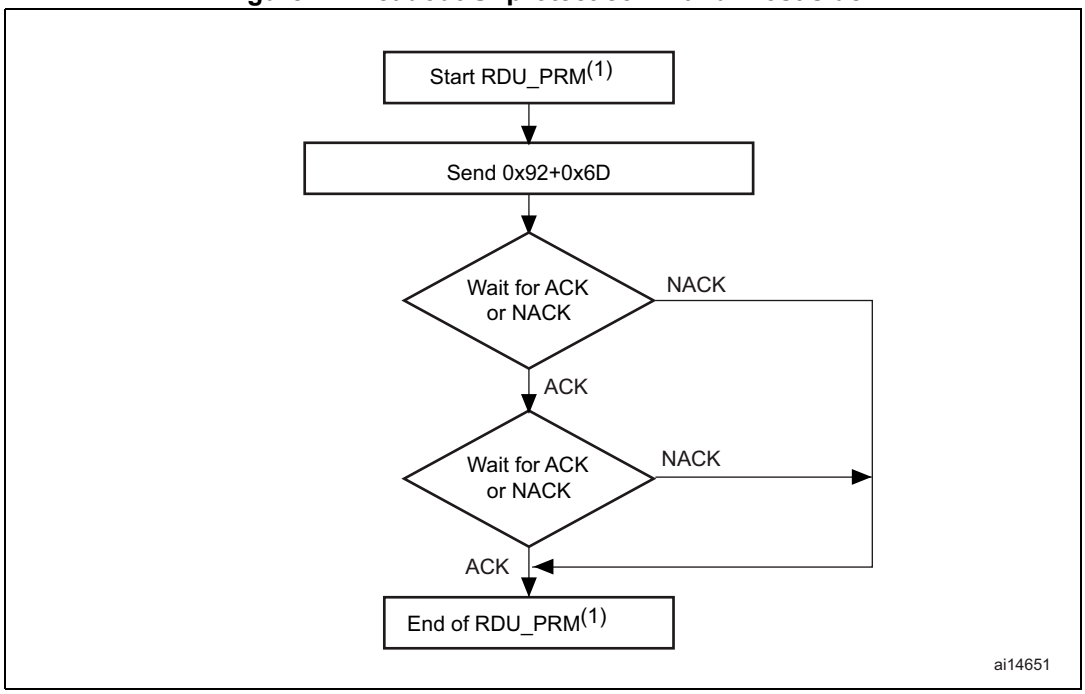
The Readout Unprotect command is used to disable the Flash memory read protection. When the bootloader receives the Readout Unprotect command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader erases all the Flash memory sectors and disables the read protection for the whole Flash memory. If the erase operation is successful, the bootloader deactivates the RDP.

If the erase operation is unsuccessful, the bootloader transmits a NACK and the read protection remains active.

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

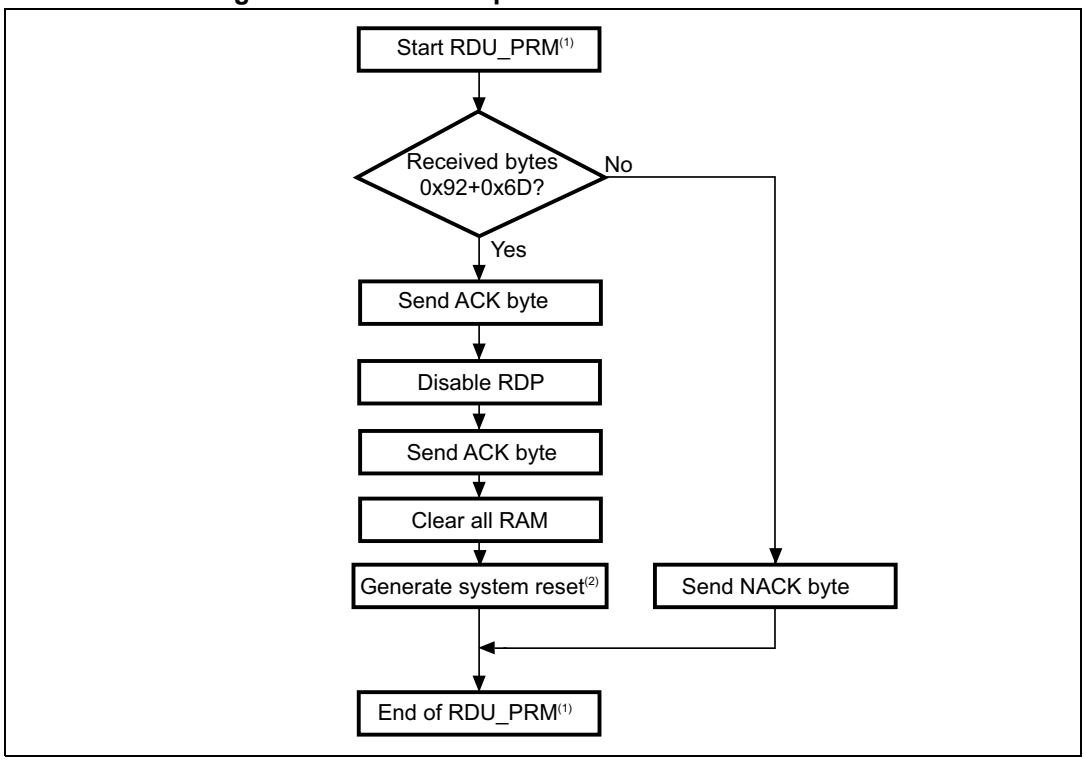
Note: For most STM32 products, the readout unprotect operation induces a mass erase of the Flash memory, so the Host has to wait sufficient time after the second ACK and before restarting connection. To know how much time this operation takes, refer to the mass erase time (when specified) in the product datasheet.

Figure 24. Readout Unprotect command: host side



1. RDU_PRM = Readout Unprotect.

Figure 25. Readout Unprotect command: device side



1. RDU_PRM = Readout Unprotect.
2. System reset is called only for some STM32 BL (STM32F0/F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

3.13 Get Checksum command

The Get Checksum command is used to compute a CRC value on a given memory area.

The memory area size must be a multiple of 32 bits (4 bytes).

When the bootloader receives the Get Checksum command, it transmits the ACK byte to the application.

After the transmission of the ACK byte the bootloader waits for an address (four bytes, byte 1 is the MSB and byte 4 is the LSB) with 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 a NACK byte and aborts the command.

When the address is valid and the checksum is correct, the bootloader waits for the size of the memory area that is expressed in 32-bit words (4 bytes) number and their complement byte (checksum).

If the checksum is not correct, the bootloader sends a NACK before aborting the command.

If the checksum is correct, the bootloader checks that the area size is different than 0 and that it does not exceed the size of the memory.

If the memory size is correct, the bootloader sends ACK byte to the application.

When the memory size is valid and the checksum is correct, the bootloader waits for the CRC polynomial value and its complement byte (checksum).

If the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command. If a product does not support polynomial value change, the value is ignored, but the device still sends ACK.

When the checksum is valid, the bootloader waits for the CRC initialization value and its complement byte (checksum).

If the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command. If a product does not support CRCR initialization value change, the value is ignored but the device still sends ACK.

If the checksum is correct, the bootloader computes the CRC of the given memory then sends an ACK to the application followed by the CRC value and is complement byte (checksum).

The host sends the bytes to the STM32 as follows:

Byte 1:	0xA1 + 0x5E
Wait for ACK	
Bytes 3 to 6:	Start address (byte 3: MSB, byte 6: LSB)
Byte 7:	Checksum: XOR (byte 3, byte 4, byte 5, byte 6)
Wait for ACK	
Bytes 8 to 11	Memory area size (number of 32-bit words) (byte 8: MSB, byte 11: LSB)
Byte 12:	Checksum: XOR (byte 8, byte 9, byte 10, byte 11)
Wait for ACK	
Byte 13 to 16:	CRC polynomial (byte 13: MSB, byte 16: LSB)
Wait for ACK	
Byte 17:	Checksum: XOR (byte 13, byte 14, byte 15, byte 16)
Wait for ACK	
Bytes 18 to 21	CRC initialization value (byte 18: MSB, byte 21: LSB)
Wait for ACK	
Byte 22:	Checksum: XOR (byte 18, byte 19, byte 20, byte 21)
Wait for ACK	

Figure 26. Get Checksum command: host side

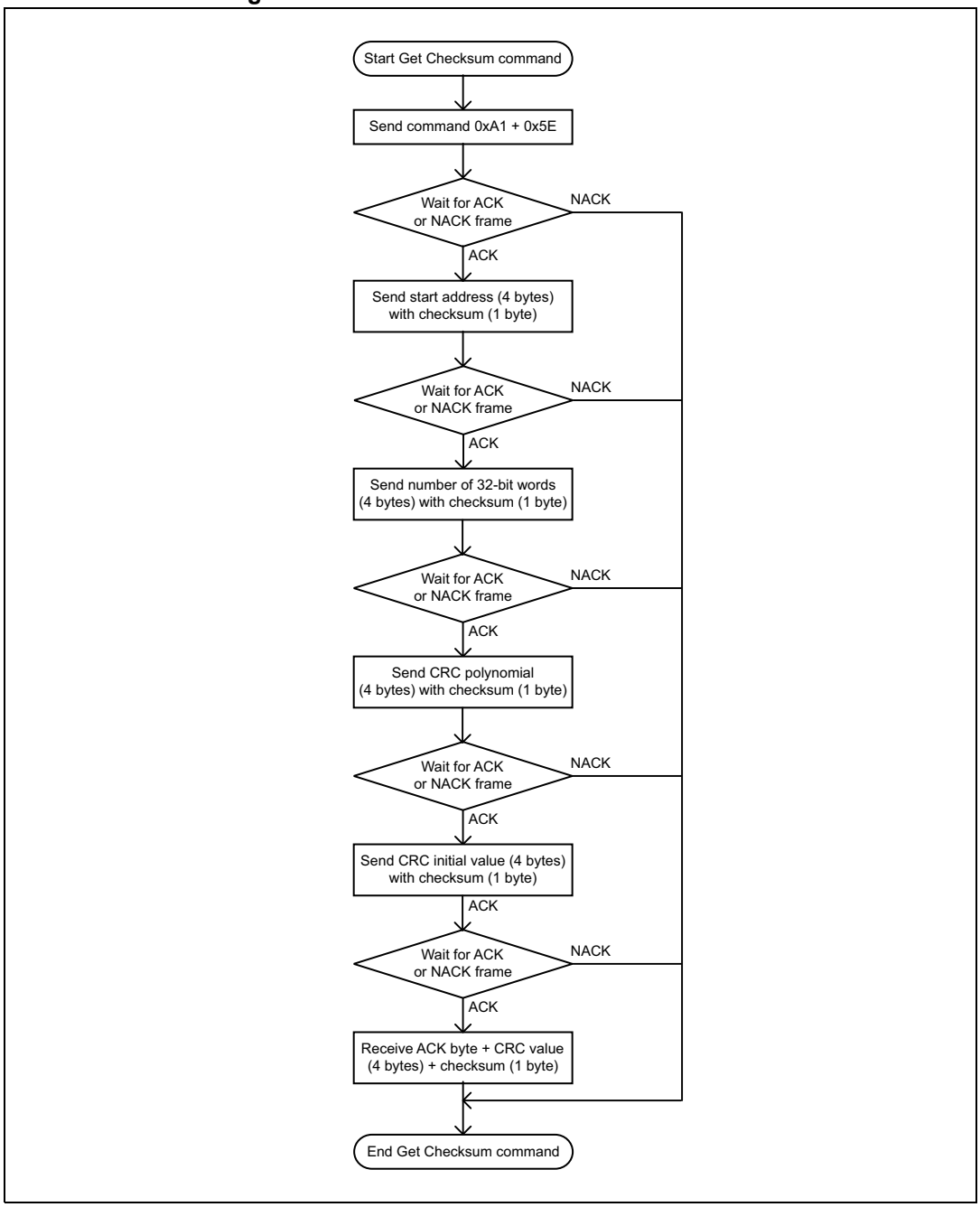
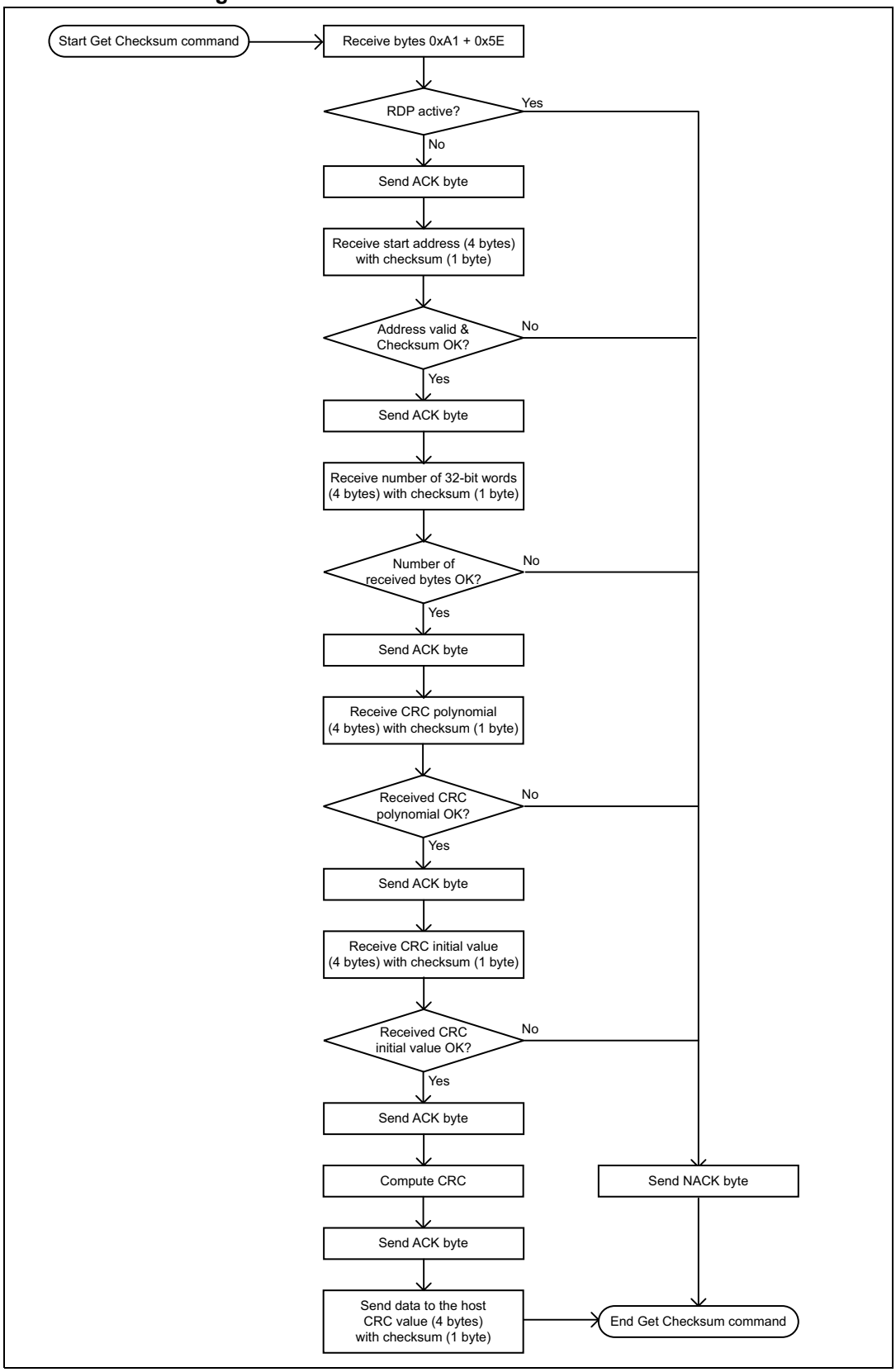


Figure 27. Get Checksum command: device side



3.14 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

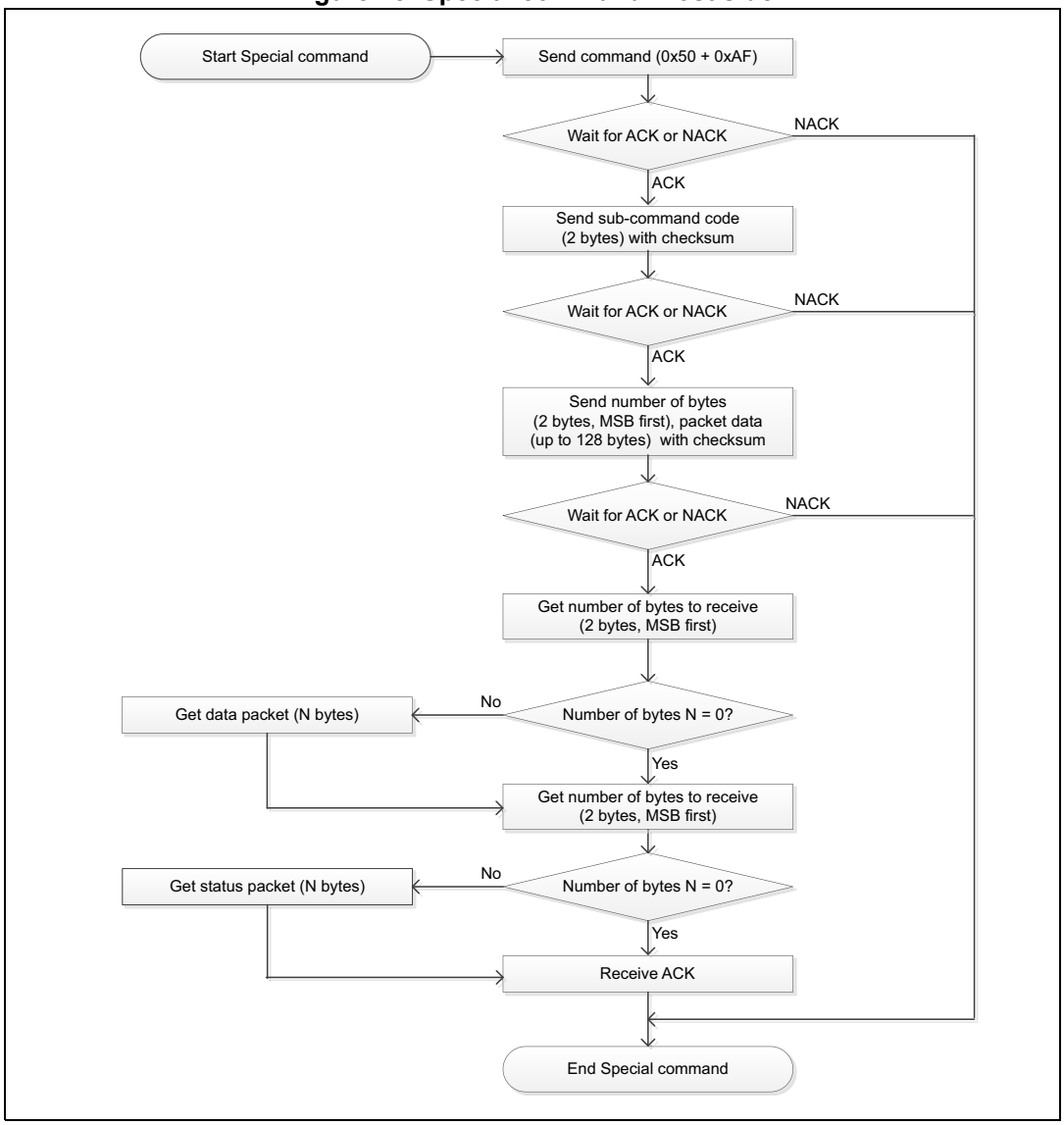
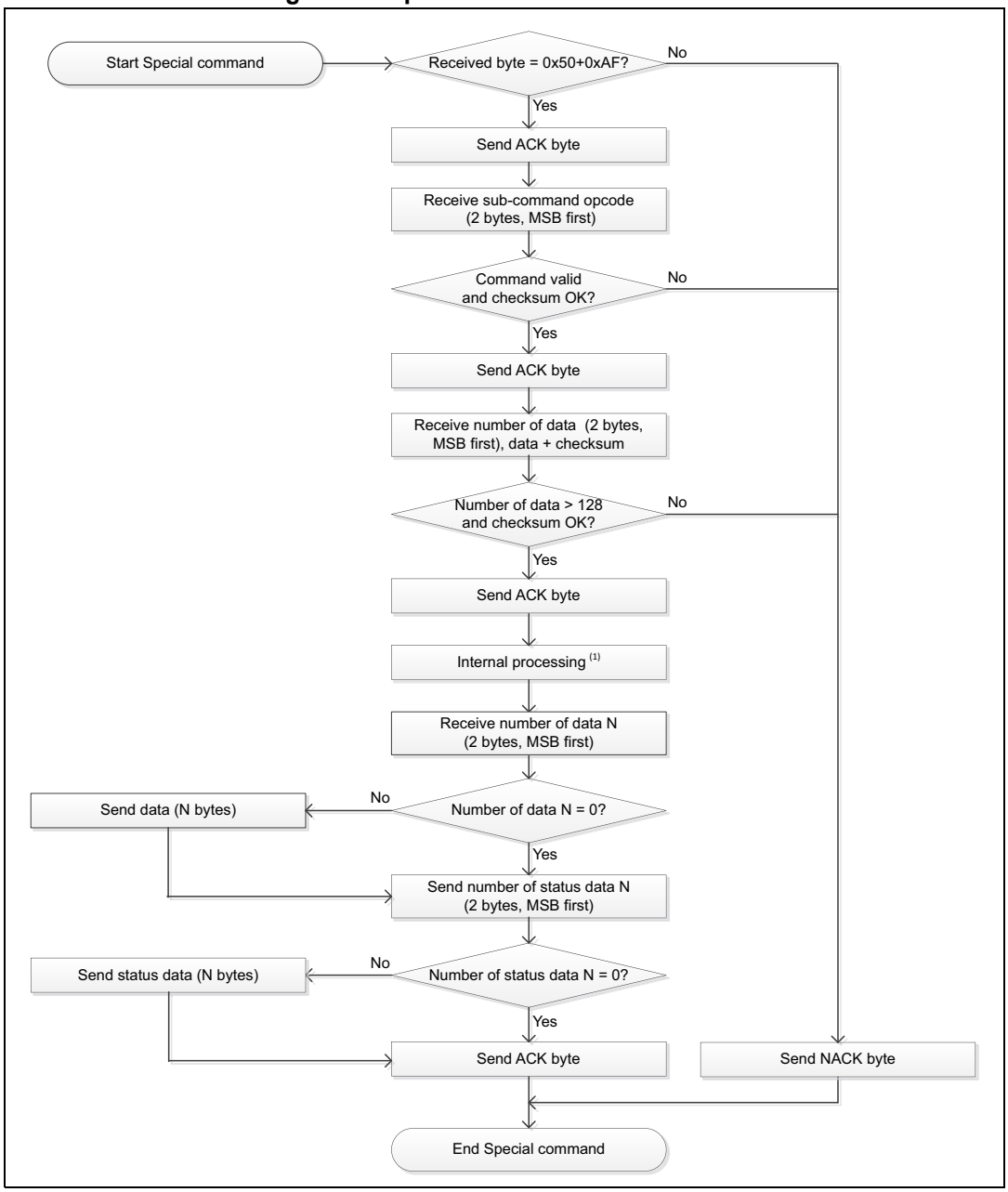


Figure 29. Special command: device side



1. The internal processing depends upon the project.

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 Sub-command opcode (two bytes, MSB first) and a checksum byte. If the Sub-command is supported by the STM32 bootloader and its checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command.

To keep the special command generic, the data packet received by the bootloader can have different sizes depending on the sub-command needs.

Therefore, the packet is split in two parts:

- Size of the data (two bytes, MSB first)
- N bytes of data
  - If N = 0, no data is transmitted
  - N must be inferior to 128.

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

Once the Sub-Command is executed using the received data, the bootloader sends a response that consist of two consecutive packets:

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

Finally, an ACK byte closes the special command interaction between bootloader and the host.

3.15 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 response it only returns the commands status.

Figure 30. Extended Special command: host side

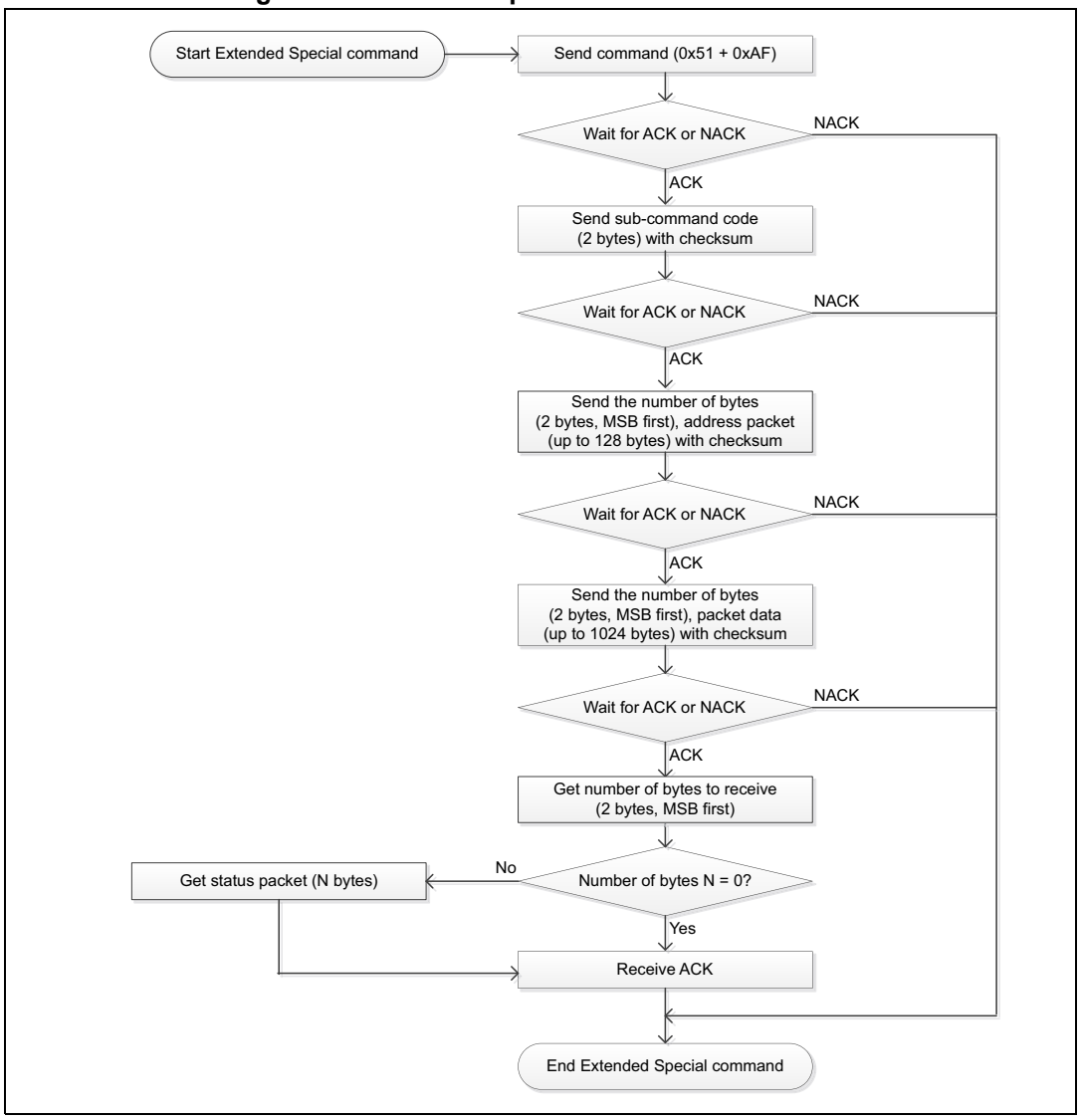
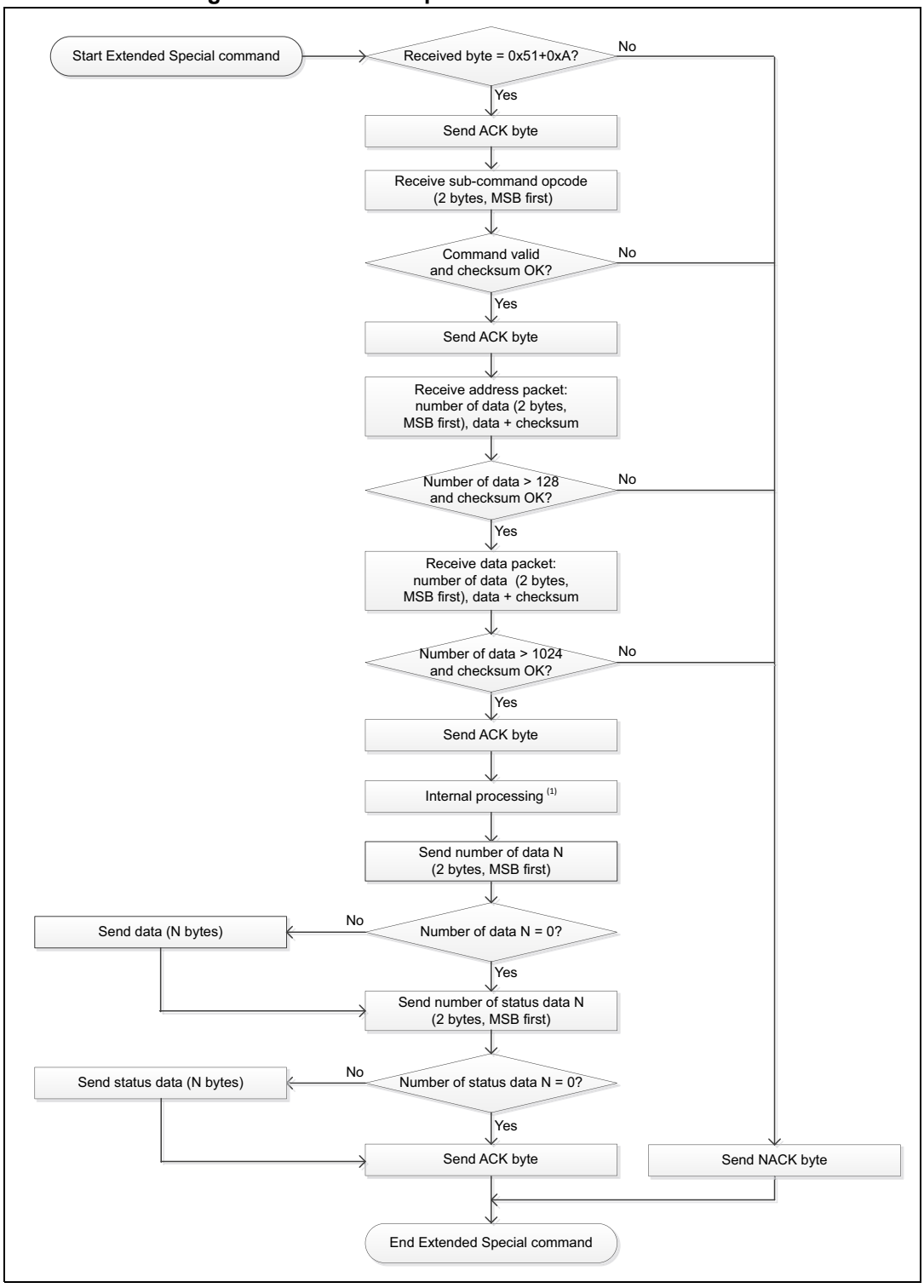


Figure 31. Extended Special command: device side



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 will wait for a Sub-command opcode (Two bytes, MSB first) and a checksum byte. If the Sub-command is supported by the

STM32 bootloader and its checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command.

The two packets then can be received depending on the sub-command needs:

- Packet1: Data 1 packet, where number of bytes is limited to 128 bytes.
- Packet2: Data2 packet, where number of bytes is limited to 1024 bytes.

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

Once the sub-Command is executed using received data, the bootloader sends a response consisting in one packet:

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

Finally, an ACK byte closes the extended special command interaction between bootloader and the host.

4 Bootloader protocol version evolution

Table 3 lists the bootloader versions.

Table 3. Bootloader protocol versions

Version	Description
V2.0	Initial bootloader version.
V2.1	– Update Go command to initialize the main stack pointer – Update Go command to return NACK when jump address is in the option byte area or system memory area – Update Get ID command to return the device ID on two bytes – Update the bootloader version to V2.1
V2.2	– Update Read Memory, Write Memory and Go commands to deny access, with a NACK response, to the first bytes of RAM used by the bootloader – Update Readout Unprotect command to initialize the whole RAM content to 0x0 before RDP disable operation
V3.0	– Extended Erase command added to support number of pages larger than 256 and separate bank mass erase – Erase command has not been modified in this version but, due to addition of the Extended Erase command it is no longer supported (Erase and Extended Erase commands are exclusive)
V3.1	– Limitation fix of: "When a Read Memory command or Write Memory command is issued with an unsupported memory address and a correct address checksum (i.e. address 0x6000 0000), the command is aborted by the bootloader device, but the NACK (0x1F) is not sent to the host. As a result, the next two bytes (that is, the number of bytes to be read/written and its checksum) are considered as a new command and its checksum" (1). – No changes in specification, the product implementation has been corrected
V3.3	– Added support for "Get Checksum" command. – Updated "Get Command" command to return the opcode of the "Get Checksum" command.

1. If the "number of data - 1" (N-1) to be read/written is not equal to a valid command code (0x00, 0x01, 0x02, 0x11, 0x21, 0x31, 0x43, 0x44, 0x63, 0x73, 0x82 or 0x92), then the limitation is not perceived from the host as the command is NACK-ed anyway (as an unsupported new command).

5 Revision history

Table 4. Document revision history

Date	Revision	Changes
09-Mar-2010	1	Initial release.
20-Apr-2010	2	*Table 2: USART bootloader commands*: added Extended Erase command; removed footnote 2 concerning read protection from the Readout Protect command. *Communication safety*: amended *Note 1*. *Section 3.1: Get command*: updated byte 10. Updated *Figure 10: Go command: host side* for missing ACK state. *Section 3.7: Write Memory command*: added *Note 1* and *Note 2*. *Figure 12*, and *Figure 13*: added notes regarding N+1. Added *Section 3.8: Extended Erase Memory command*. *Table 3: Bootloader protocol versions*: added v3.0.
12-Feb-2013	3	Added *Note:*, *Note* and *Note:*. Changed all occurrences of “ROP” by “RDP” including in figures: *Figure 9.*, *Figure 11.*, *Figure 13.*, *Figure 15.*, *Figure 17.*, *Figure 19.*, *Figure 21.*, *Figure 23.*, *Figure 25.*. Added *Table 1: Applicable products*.
26-Mar-2013	4	Added Version 3.1 in *Table 3: Bootloader protocol versions*. Updated “byte 4” value in *Section 3.3: Get ID command*. Replaced “address” by “ID” in this *Note 1*. Replaced “End of EER” by “End of Go” in *Figure 10: Go command: host side*. Updated first sentence in *Section 3.7: Write Memory command*. Removed “& address=0x1FFF F800” and replaced the two tests “Flash memory address?” and “RAM address?” by a single test in *Figure 13: Write Memory command: device side*. Precised missing “Y” values in the third test of *Figure 17: Extended Erase Memory command: device side*. Added *Note:* above *Figure 24: Readout Unprotect command: host side*.
22-May-2013	5	Replaced “STM32L151xx, STM32L152xx and STM32L162xx” by “STM32L1 Series” in *Table 1: Applicable products*.
20-Jun-2014	6	Updated *Table 1: Applicable products*. Removed footnote 4 and added footnote *3*.in *Table 2: USART bootloader commands*. Removed section 3.1 Device-dependent bootloader parameters. Updated *Figure 10: Go command: host side* and *Figure 11: Go command: device side*. Updated *Section 3.6: Write Memory command*.
21-Oct-2016	7	Introduced STM32L4 and STM32F7 Series, hence updated *Introduction* and *Table 1: Applicable products*.

Table 4. Document revision history (continued)

Date	Revision	Changes
14-Feb-2019	8	Added STM32H7 Series, hence updated *Table 1: Applicable products*. Updated *Section 1: USART bootloader code sequence*. Updated *Figure 4: Get Version & Read Protection Status command: host side*. Minor text edits across the whole document.
21-Feb-2019	9	Added STM32WB Series, hence updated *Table 1: Applicable products*.
09-Apr-2019	10	Added STM32G0 and STM32G4 Series, hence updated *Table 1: Applicable products*.
23-Sep-2019	11	Added STM32L5 Series, hence updated *Table 1: Applicable products*.
04-Dec-2019	12	Added STM32WL Series, hence updated *Table 1: Applicable products*.
30-Oct-2020	13	Updated *Section 3.1: Get command*. Added *Section 3.13: Get Checksum command*. Updated *Table 2: USART bootloader commands* and *Table 3: Bootloader protocol versions*. Minor text edits across the whole document.
08-Jun-2021	14	Updated *Section 1: USART bootloader code sequence* and *Section 3.13: Get Checksum command*. Updated *Table 2: USART bootloader commands*. Updated *Figure 13: Write Memory command: device side*, *Figure 19: Write Protect command: device side*, *Figure 21: Write Unprotect command: device side*, *Figure 23: Readout Protect command: device side* and *Figure 25: Readout Unprotect command: device side* and added footnotes to them. Updated *Figure 26: Get Checksum command: host side*. Added *Section 3.14: Special command* and *Section 3.15: Extended Special command*.
08-Feb-2022	15	Added STM32U5 Series, hence updated *Table 1: Applicable products*.