AN4286
Application note

How to use SPI protocol in bootloader on STM32 MCUs

Introduction

This application note describes the SPI protocol used in the STM32 microcontroller bootloader, detailing each supported command. The document applies to the STM32 products embedding bootloader versions V8.x, V9.x, V11.x, V12.x, V13.x, and V14.x, as specified in 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 SPI hardware resources and requirements for your device bootloader, refer to the already mentioned AN2606.

Table 1. Applicable products

Product family	Product series or line
Microcontrollers	STM32C0 series STM32C5 series STM32F4 series STM32F7 series STM32G0 series STM32G4 series STM32H5 series STM32H7 series STM32L0 series STM32L4 series STM32L5 series STM32U0 series STM32U3 series STM32U5 series STM32WB series STM32WBA series STM32WL5x/Ex line

Contents

List of tables

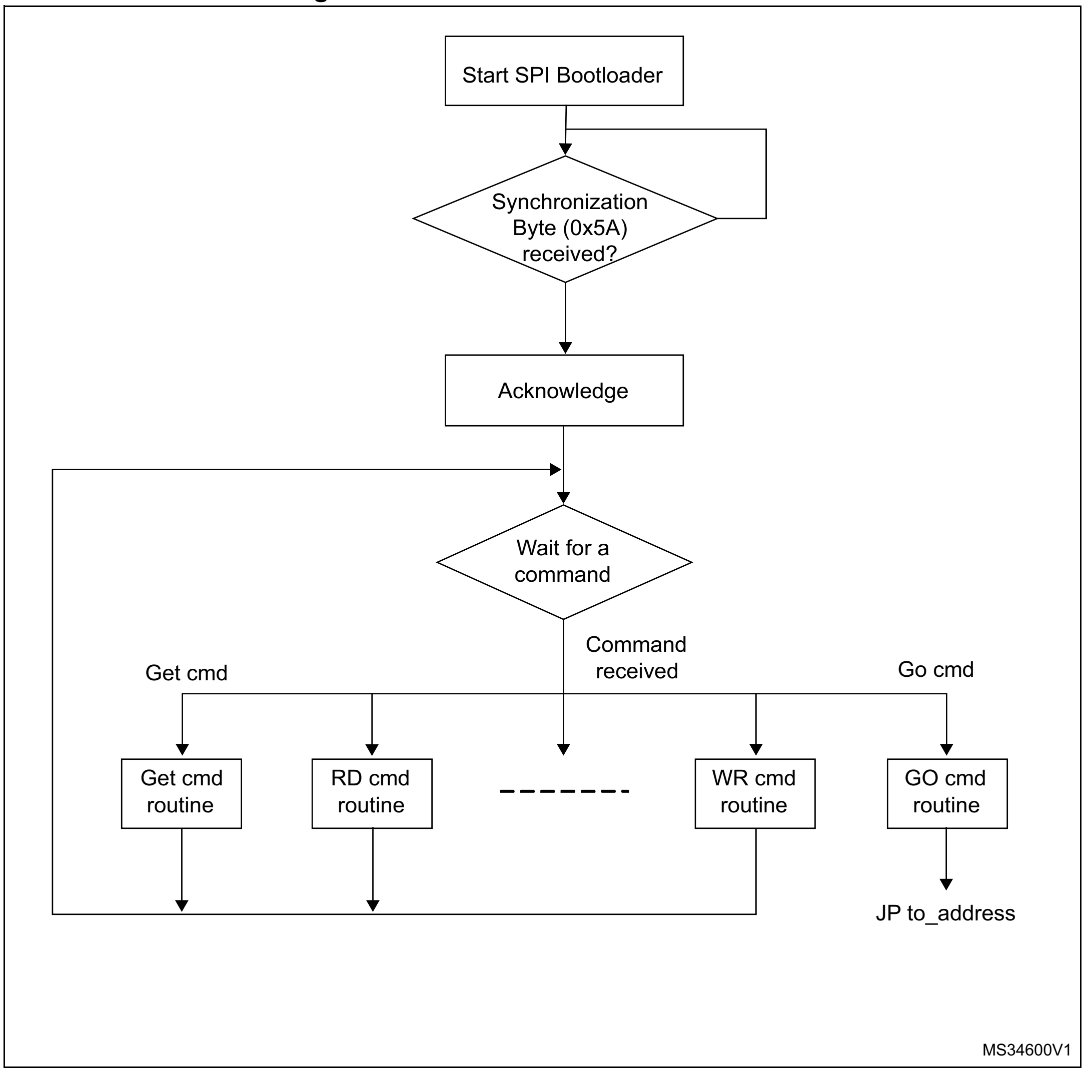
List of figures

1 SPI bootloader code sequence

The bootloader for STM32 microcontrollers, based on Arm® Cortex®[a] cores, is an SPI slave.

For all SPI bootloader operations, the NSS pin (chip select) must be low. If the NSS pin is high, the microcontroller ignores the communication on the SPI bus.

Figure 1. Bootloader for STM32 with SPI

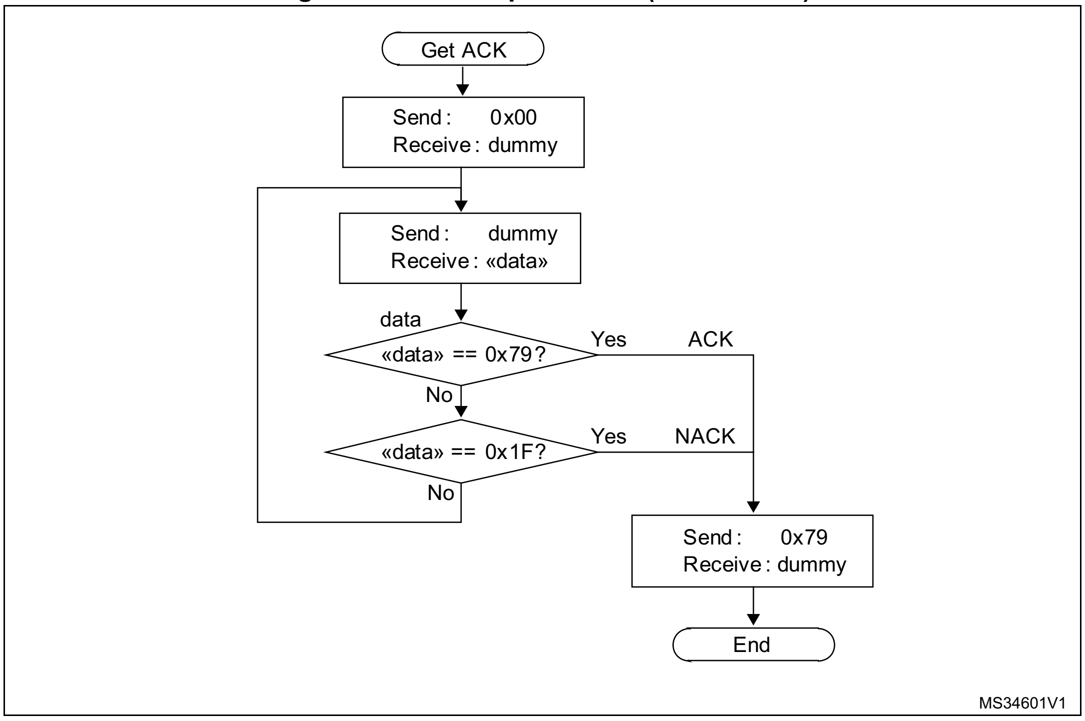


a. Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

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

Once the system memory boot mode is entered and the microcontroller has been configured (for more details, refer to STM32 system memory boot mode application notes), the bootloader code begins to scan the SPI_MOSI line pin, waiting to detect a synchronization byte (0x5A) on the bus. Once a detection occurs, the SPI bootloader firmware waits to receive the acknowledge procedure (refer to *Figure 2*), and then starts to receive master commands.

Figure 2. Get ACK procedure (master side)



As indicated in *Figure 3* (where xx represents a dummy byte), to start communication with the bootloader the master must first send a synchronization byte (0x5A), and then wait for an acknowledge (ACK).

Figure 3. Bootloader SPI synchronization frame

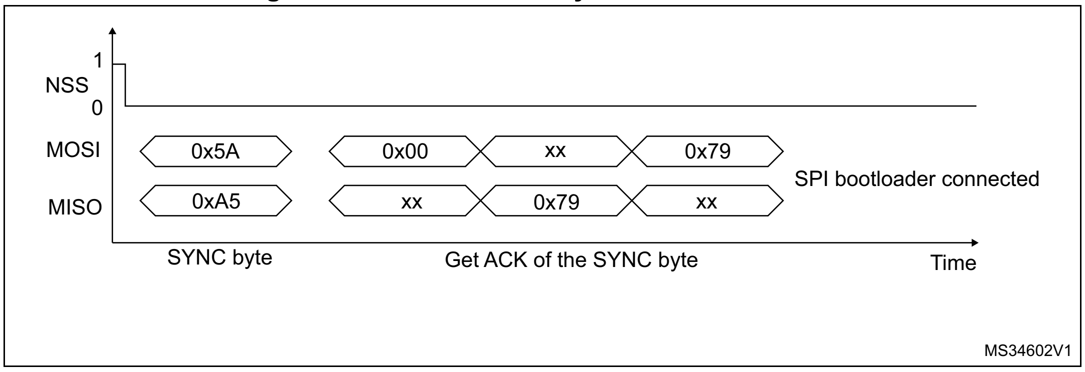
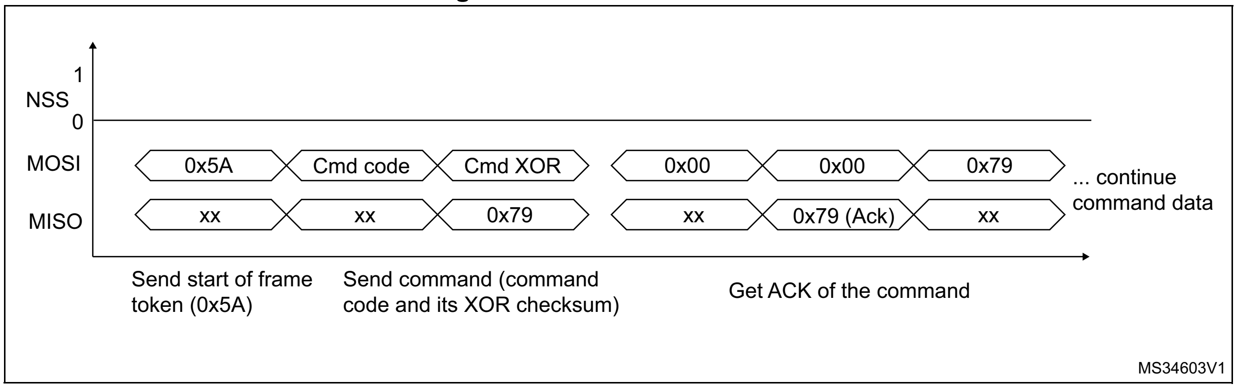
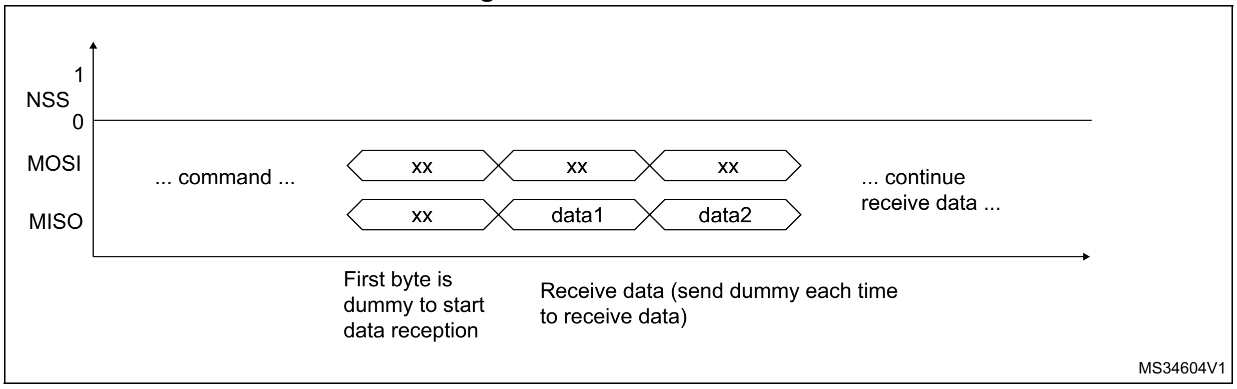


Figure 4. SPI command frame


1
NSS
0
MOSI
MISO
0x5A
Cmd code
Cmd XOR
0x00
0x00
0x79
xx
xx
0x79
xx
0x79 (Ack)
xx
... continue command data
Send start of frame token (0x5A)
Send command (command code and its XOR checksum)
Get ACK of the command
MS34603V1

To read data sent by the slave, the master must first send a dummy byte (called BUSY byte). This applies to all commands where a read is required.

Figure 5. Read data frame


1
NSS
0
MOSI
MISO
... command ...
xx
xx
xx
xx
data1
data2
... continue receive data ...
First byte is dummy to start data reception
Receive data (send dummy each time to receive data)
MS34604V1

Note: The dummy byte sent by the bootloader is 0xA5.

The minimum timing to observe between bytes sent over the SPI is 15 µs.

2 Bootloader command set

Table 2 lists the supported commands. Each command is described in this section.

Table 2. SPI bootloader commands

Command[1]	Code	Description
Get[2]	0x00	Gets the version and allowed commands supported by the current version of the bootloader.
Get Version[2]	0x01	Gets the protocol version.
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.
Write Memory[3]	0x31	Writes up to 256 bytes to the memory starting from an address specified by the application.
Erase[3]	0x44	Erases from one to all the flash memory pages or sectors using two-byte addressing mode.
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 write protection for some sectors.
Write Unprotect	0x73	Disables write protection for all flash memory sectors.
Readout Protect	0x82	Enables read protection.
Readout Unprotect[2]	0x92	Disables 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 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 STM32 product datasheet and to AN2606 to know which memory spaces are valid for this command.

As the SPI is configured in full duplex, each time the master transmits data on the MOSI line, it simultaneously receives data on the MISO line. As the slave answer is not immediate, the received data are ignored (dummy) while the master transmits (these data are not used by the master).

When the slave must transmit data, the master sends its clock. It must transmit data on the MOSI line to receive data on the MISO line. In this case, the master must always send 0x00 (datum not used by the slave).

2.1 Safety of communication

All communication from the programming master to the slave is verified in the following way.

- 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 and checksum), the result at the end of the packet must be 0x00.
- If the received data is one byte, its checksum is the bit negation of the value (as an example, the checksum of 0x02 is 0xFD).
- For each command, the master sends three bytes: a start of frame (SOF = 0x5A), a byte representing the command value, and its complement (XOR of the command and its complement = 0x00).
- Each packet is either accepted (ACK answer) or discarded (NACK answer).
  - ACK = 0x79
  - NACK = 0x1F

The master frame can be one of the following.

- Send command frame: the master initiates communication as master transmitter, and sends two bytes to the slave: command code plus XOR.
- Wait for ACK/NACK frame: the master initiates an SPI communication as master receiver and receives one byte from the slave: ACK or NACK.
- Receive data frame: the master initiates an SPI communication as master receiver and receives the response from the slave. The number of bytes received depends on the command.
- Send data frame: the master initiates an SPI communication as master transmitter, and sends the needed bytes to the slave. The number of bytes transmitted depends on the command.

2.2 Get command

This command enables the user to get the version of the protocol and the supported commands. When the bootloader receives the Get command, it transmits the protocol version and the supported command codes to the master, as described in *Figure 6*.

Figure 6. Get command: master side

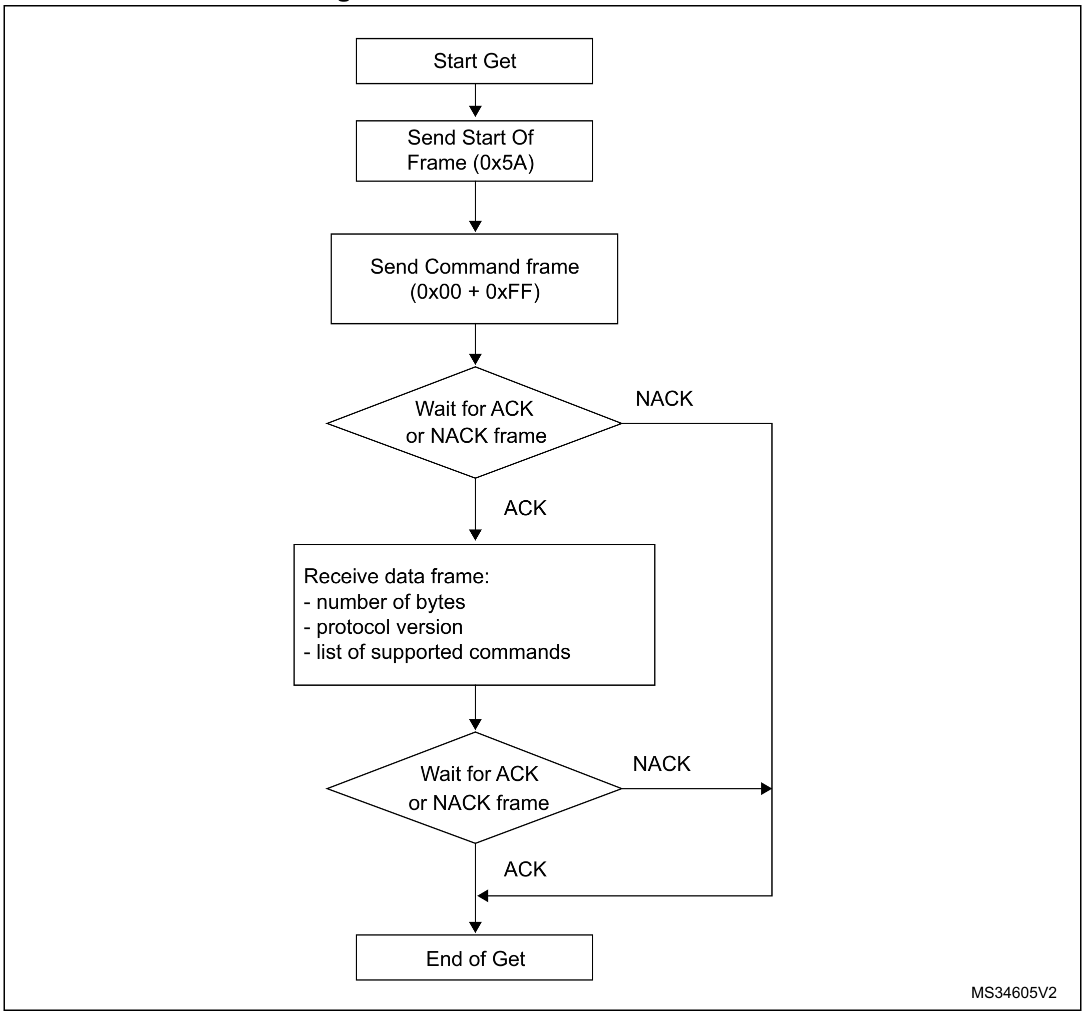
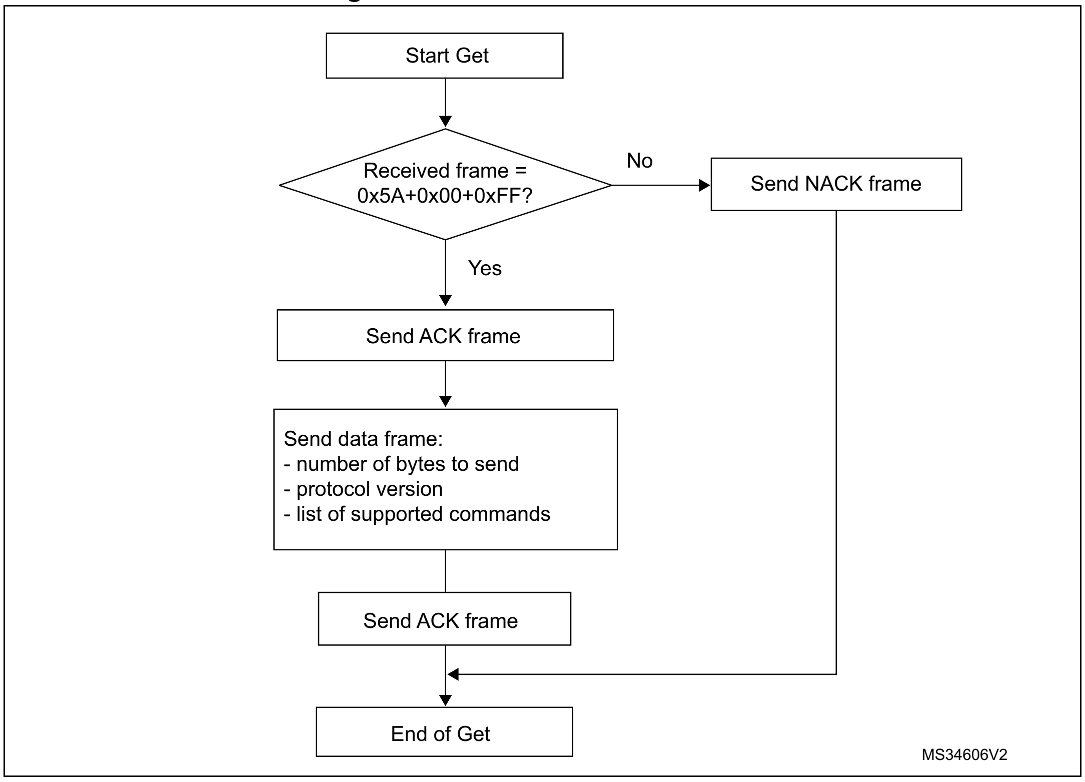

Start Get
Send Start Of Frame (0x5A)
Send Command frame (0x00 + 0xFF)
Wait for ACK or NACK frame
NACK
ACK
Receive data frame:
- number of bytes
- protocol version
- list of supported commands
Wait for ACK or NACK frame
NACK
ACK
End of Get
MS34605V2

Figure 7. Get command: slave 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: protocol version (0 < version < 255), example: 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: 0xA1 (Get Checksum command), available only for version 1.3

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.

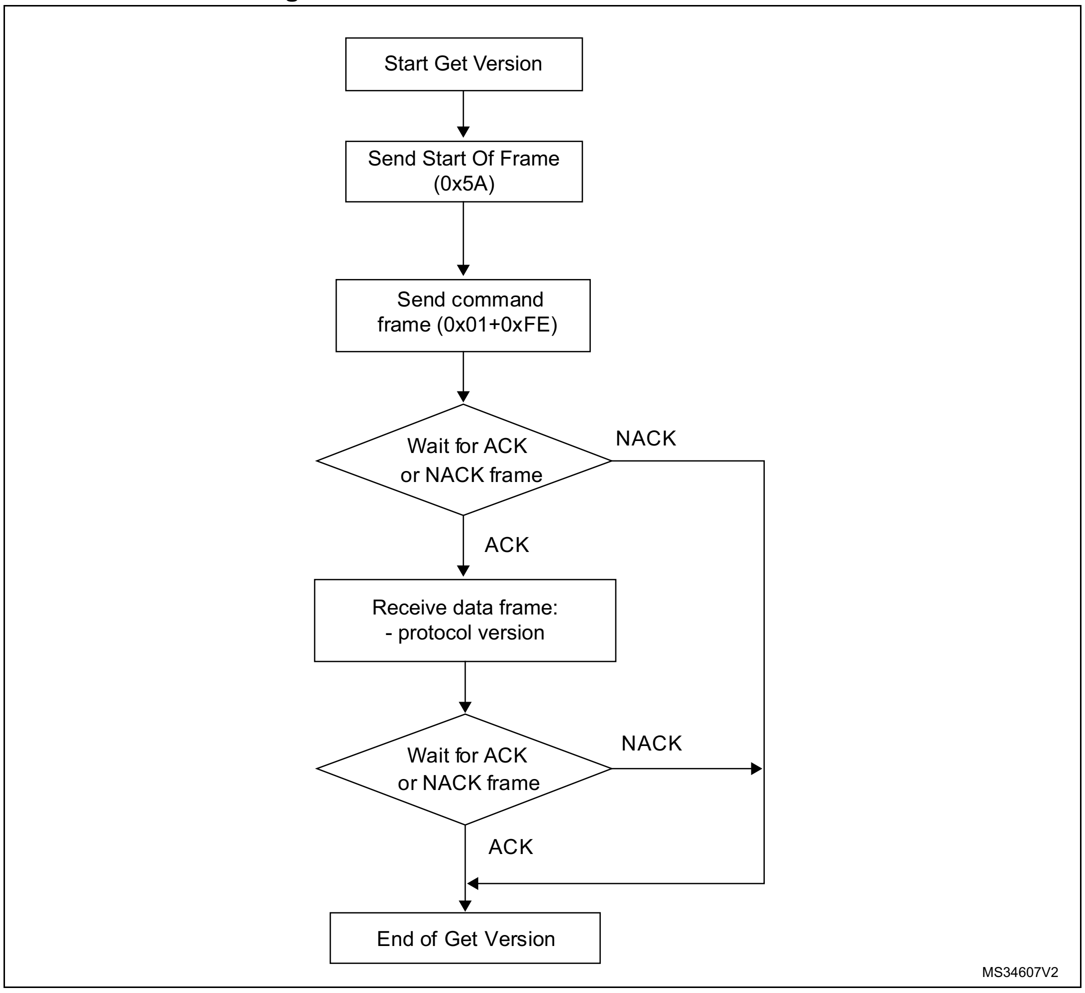
As an example, for the STM32H5 series there is no RDP HW feature. The Get command is:

- Byte 1: ACK
- Byte 2: N = 10 = 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)

2.3 Get Version command

This command is used to get the version of the SPI protocol. When the bootloader receives the command, it transmits the bootloader version to the master.

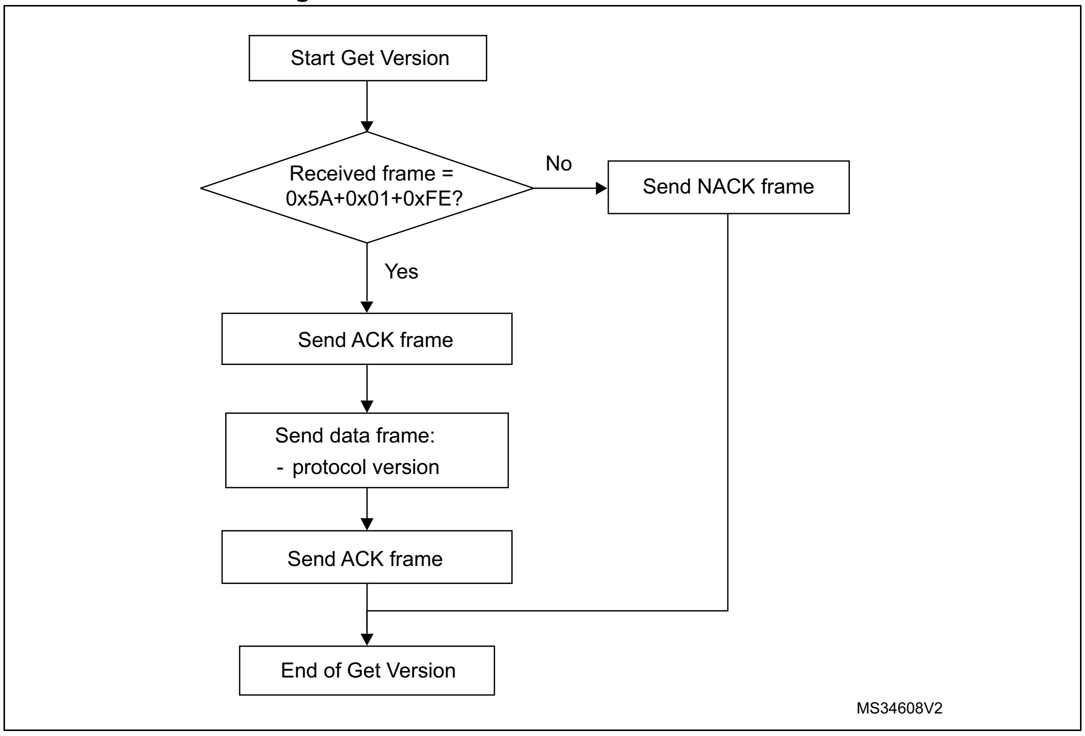
Figure 8. Get Version command: master side


Start Get Version
Send Start Of Frame (0x5A)
Send command frame (0x01+0xFE)
Wait for ACK or NACK frame
NACK
ACK
Receive data frame: - protocol version
Wait for ACK or NACK frame
NACK
ACK
End of Get Version
MS34607V2

The STM32 sends the bytes as follows:

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

Figure 9. Get Version command: slave side



2.4 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 master.

The STM32 slave sends the bytes as follows:

- Byte 1: ACK
- Byte 2: N = nmber of bytes – 1 (N = 1), except for the current byte and ACKs
- Bytes 3-4: PID
  - Byte 3 = MSB
  - Byte 4 = LSB
- Byte 5: ACK

Figure 10. Get ID command: master side

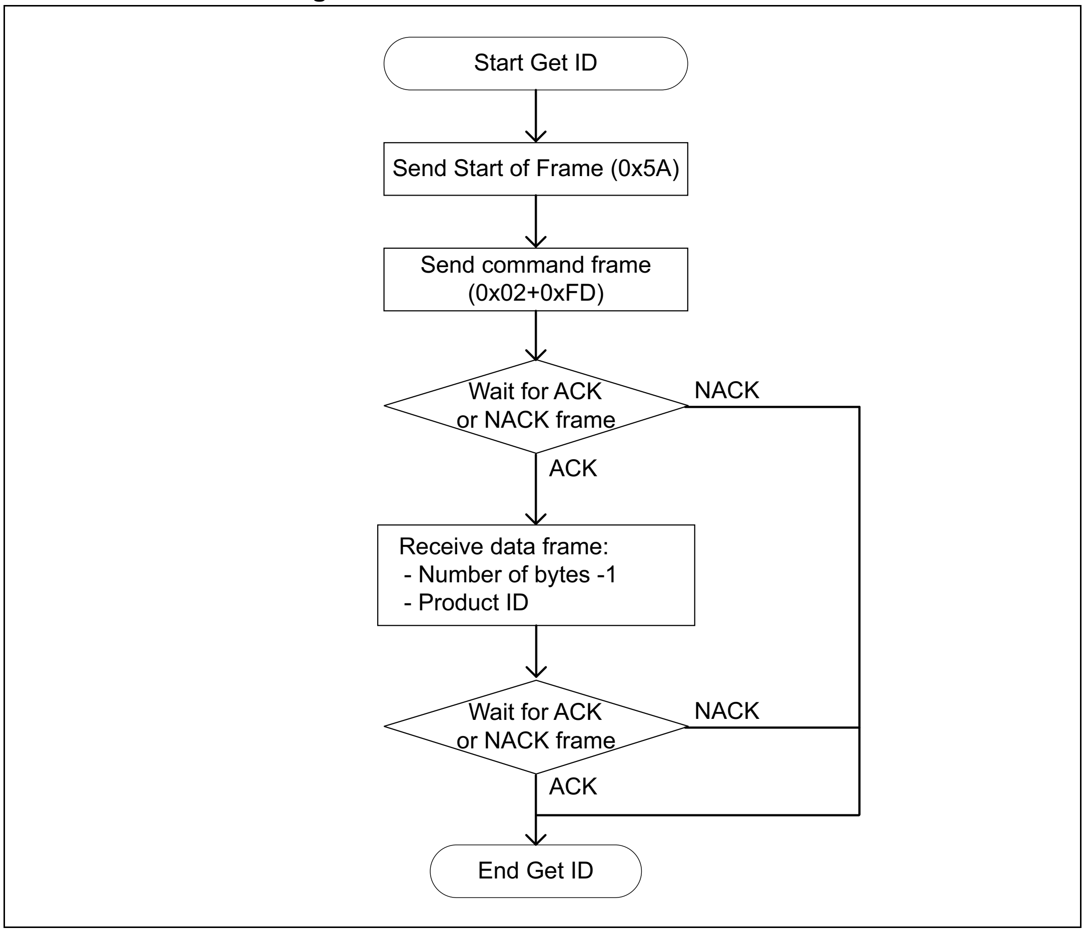
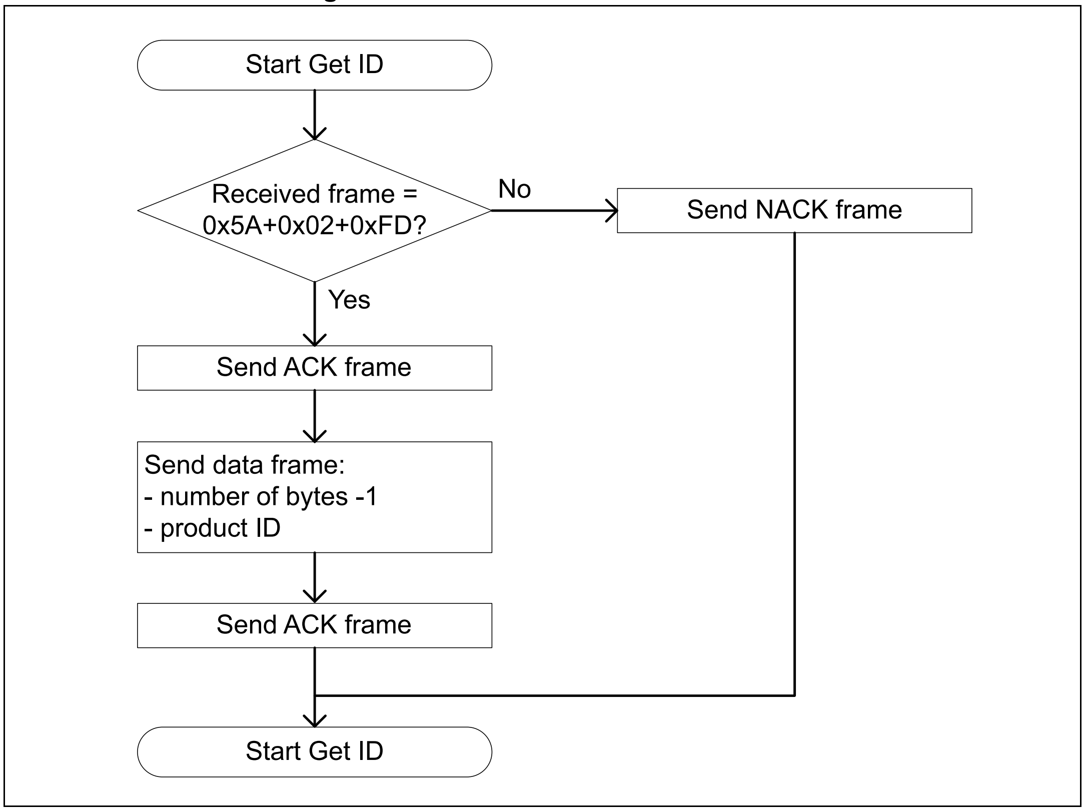

Start Get ID
Send Start of Frame (0x5A)
Send command frame (0x02+0xFD)
Wait for ACK or NACK frame
NACK
ACK
Receive data frame:
- Number of bytes -1
- Product ID
Wait for ACK or NACK frame
NACK
ACK
End Get ID

Figure 11. Get ID command: slave side



2.5 Read Memory command

This command is used to read data from any valid memory address in the RAM, flash memory, and information block (system memory or option byte areas).

When the bootloader receives the command, it transmits the ACK byte to the application, waits for an address (4 bytes, byte 1 being the MSB, and byte 4 being 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.

When 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, it 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 master sends bytes to the STM32 as follows:

- Start of frame: 0x5A
- Bytes 1-2: 0x11 + 0xEE
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Bytes 3-6: start address (byte 3: MSB, byte 6: LSB)
- Byte 7: checksum: XOR (byte 3, byte 4, byte 5, and byte 6)
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Byte 8: number of bytes to be read - 1 (0 < N ≤ 255)
- Byte 9: checksum: XOR byte 8 (complement of byte 8)

Figure 12. Read Memory command: master side

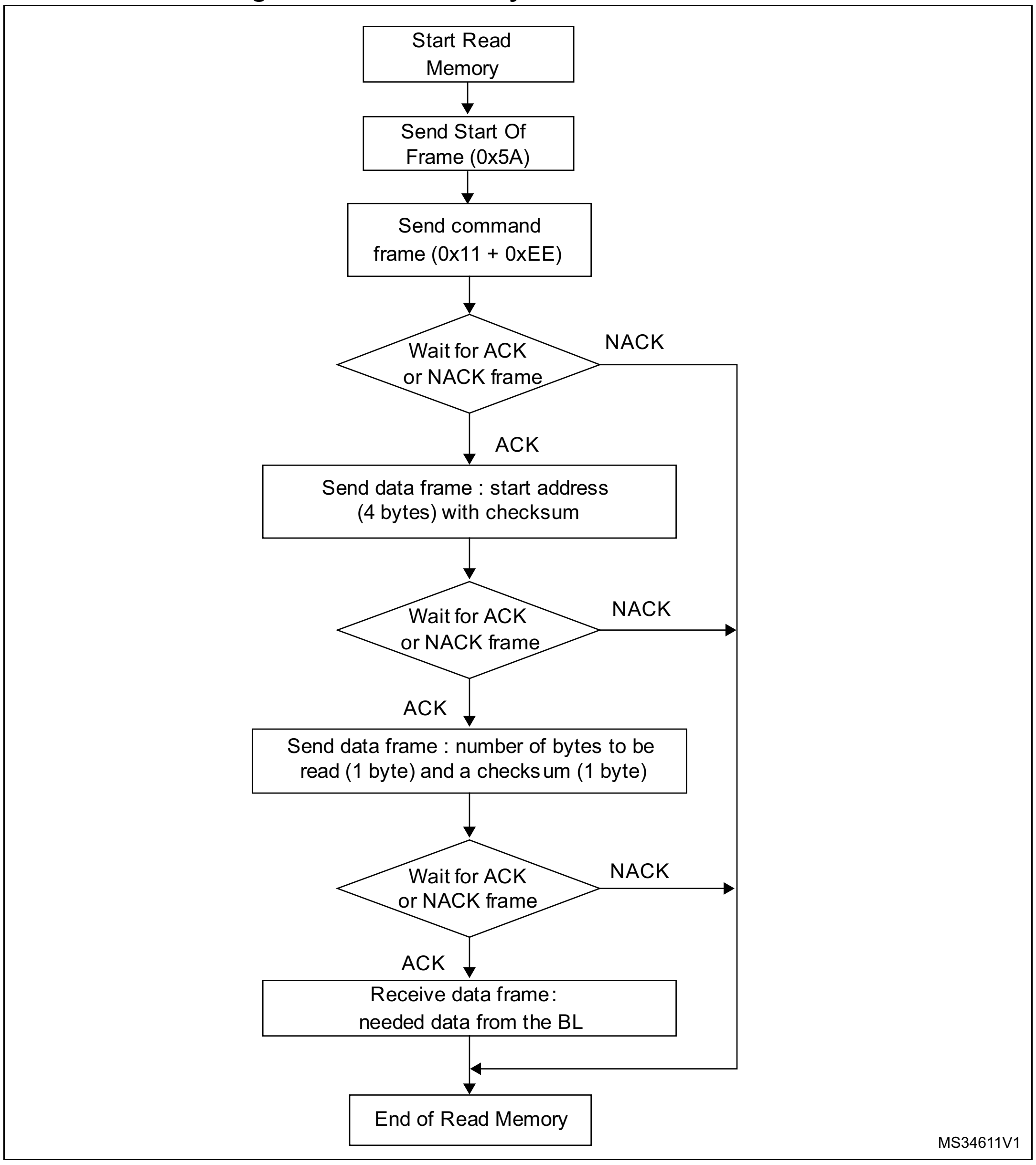
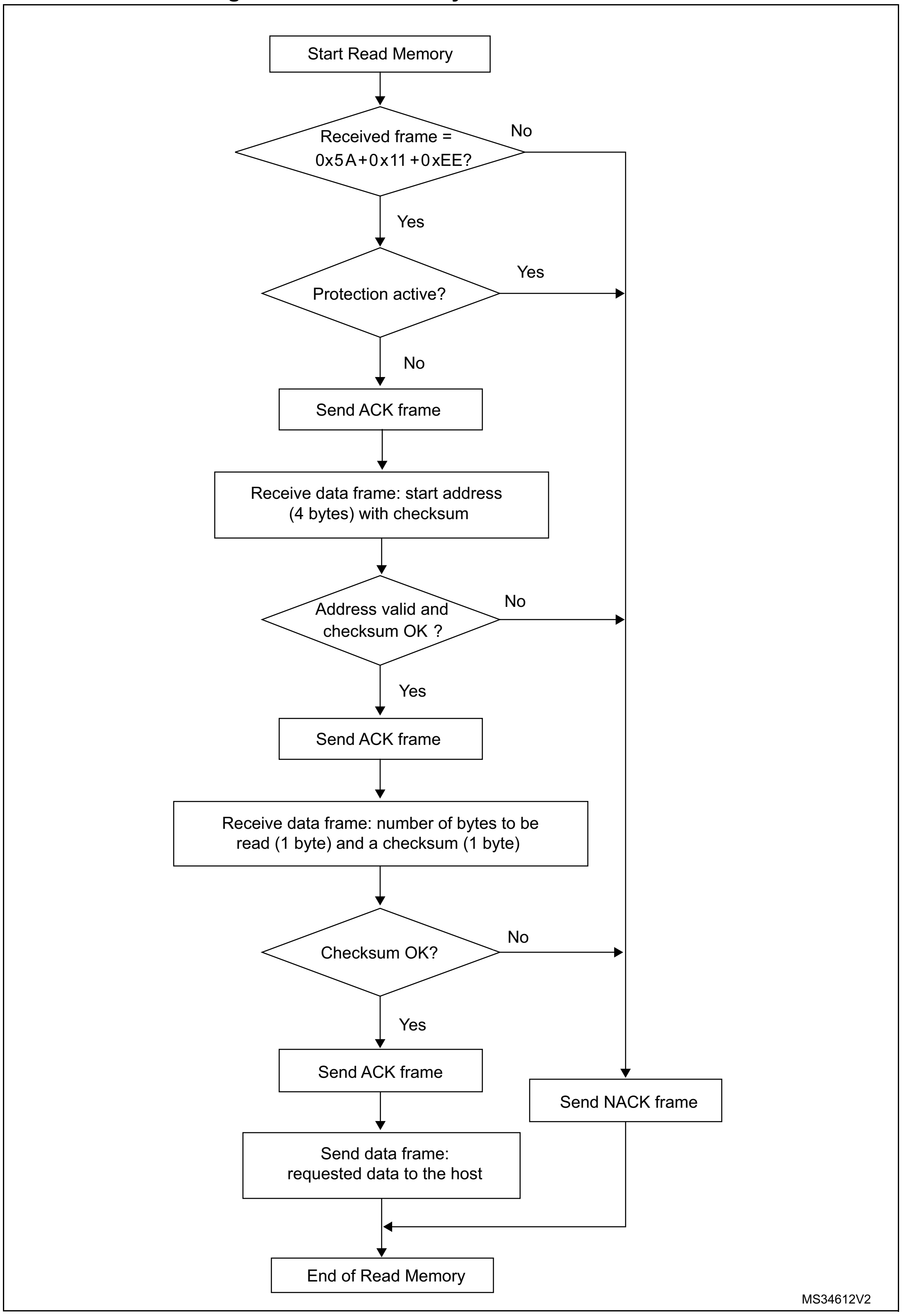

Start Read Memory
Send Start Of Frame (0x5A)
Send command frame (0x11 + 0xEE)
Wait for ACK or NACK frame
NACK
ACK
Send data frame : start address (4 bytes) with checksum
Wait for ACK or NACK frame
NACK
ACK
Send data frame : number of bytes to be read (1 byte) and a checksum (1 byte)
Wait for ACK or NACK frame
NACK
ACK
Receive data frame: needed data from the BL
End of Read Memory
MS34611V1

Figure 13. Read Memory command: slave side


Start Read Memory
Received frame = 0x5A+0x11+0xEE?
No
Yes
Protection active?
Yes
No
Send ACK frame
Receive data frame: start address (4 bytes) with checksum
Address valid and checksum OK ?
No
Yes
Send ACK frame
Receive data frame: number of bytes to be read (1 byte) and a checksum (1 byte)
Checksum OK?
No
Yes
Send ACK frame
Send NACK frame
Send data frame: requested data to the host
End of Read Memory
MS34612V2

2.6 Go command

This 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 command, it transmits the ACK byte to the application. After transmission of the ACK byte, the bootloader waits for an address (4 bytes, byte 1 being the MSB, and byte 4 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.

When the address is valid and the checksum is correct, the bootloader firmware performs the following actions.

- Initializes the registers of the peripherals used by the bootloader to their default reset values.
- Initializes the user application main stack pointer.
- Jumps to the memory location programmed in the received address + 4 (which 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 master sends the base address where the application to jump to is programmed.

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

The master sends bytes to the STM32 as follows:

- Start of frame: 0x5A
- Byte 1: 0x21
- Byte 2: 0xDE
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Byte 3 to byte 6: start address
  - Byte 3: MSB
  - Byte 6: LSB
- Byte 7: checksum: XOR (byte 3, byte 4, byte 5, and byte 6)

Figure 14. Go command: master side

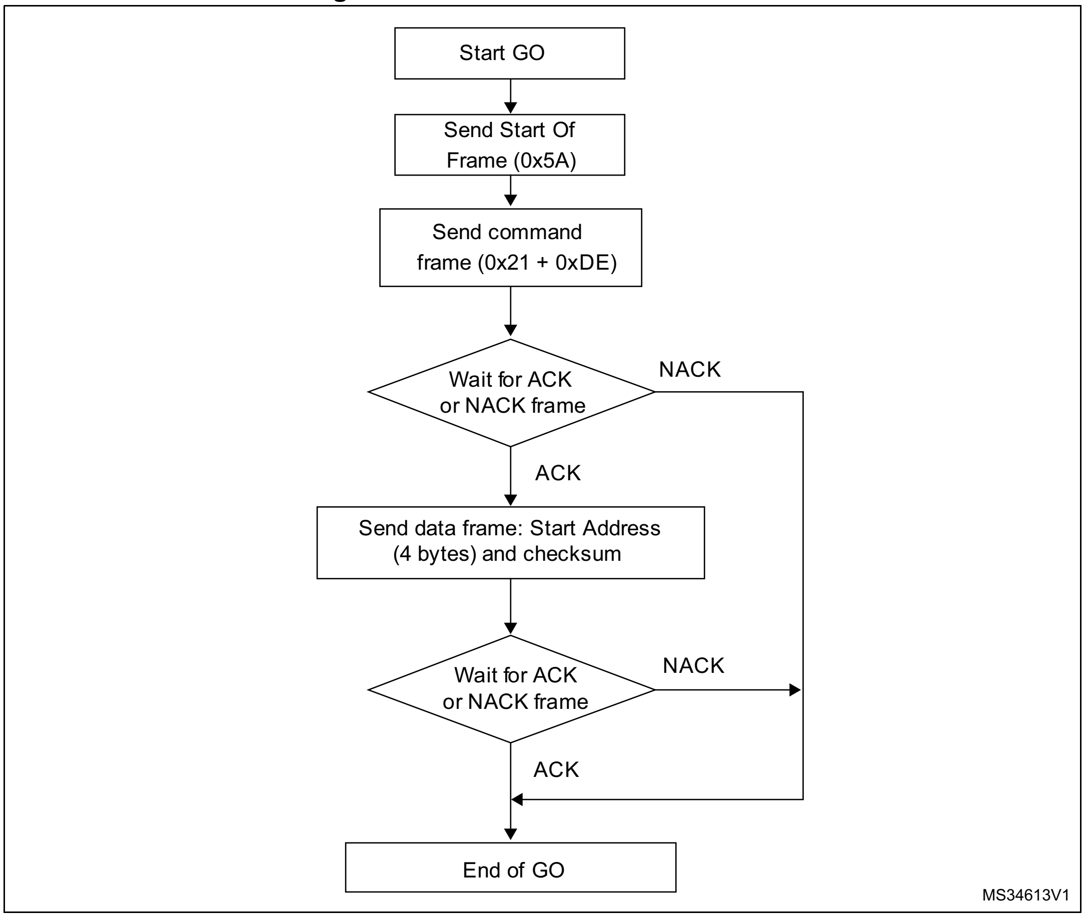
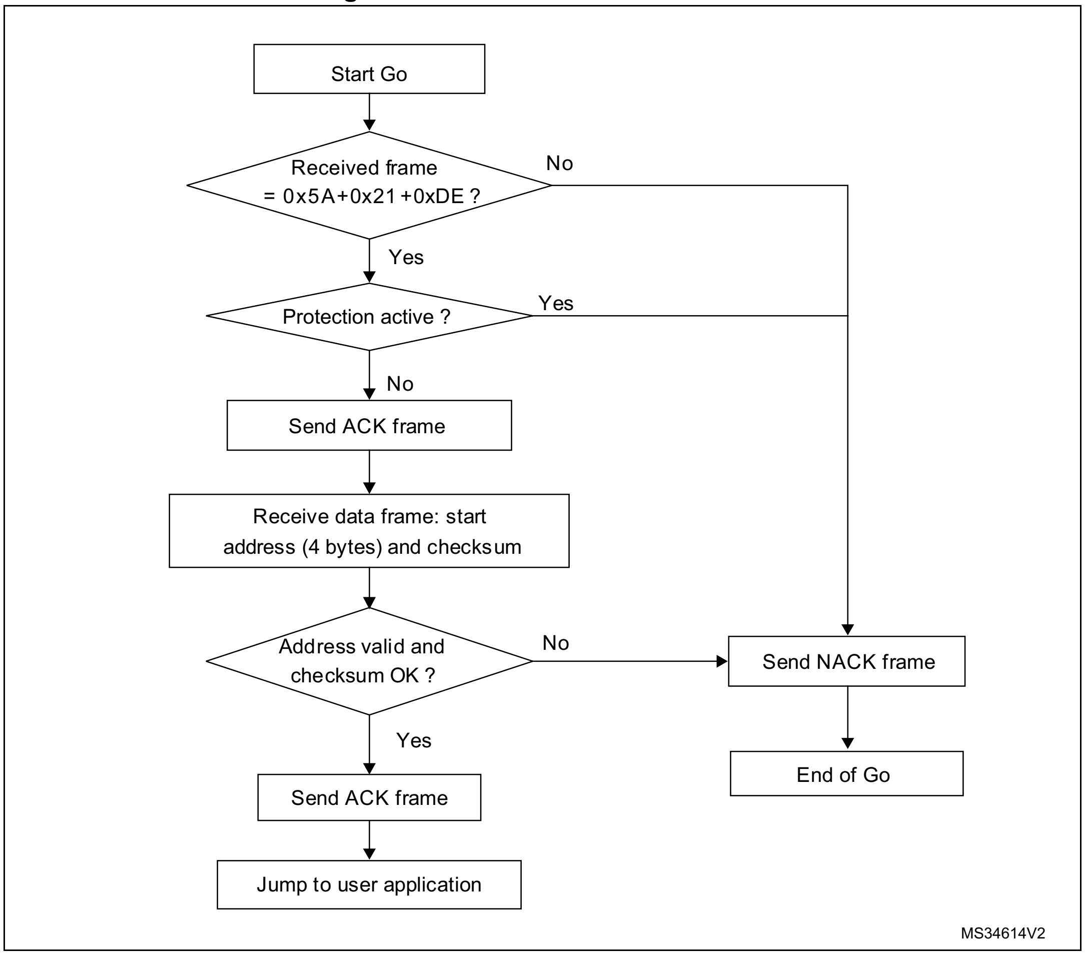

Start GO
Send Start Of Frame (0x5A)
Send command frame (0x21 + 0xDE)
Wait for ACK or NACK frame
NACK
ACK
Send data frame: Start Address (4 bytes) and checksum
Wait for ACK or NACK frame
NACK
ACK
End of GO
MS34613V1

Figure 15. Go command: slave side



2.7 Write Memory command

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

When the bootloader receives the command, it transmits the ACK byte to the application, waits for an address (4 bytes, byte 1 being the address MSB, and byte 4 being 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 performs the following actions:

- 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 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 option bytes are erased before writing the new values. At the end of the command the bootloader generates a system reset to take into account the new configuration. The start address and the maximum length of the block to write in the Option byte area must respect the address and size of the product option bytes.

If the write destination is the flash memory, the master must wait enough time for the sent buffer to be written (refer to product datasheet for timing values) before polling for a slave response.

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

Write operations to the flash memory must be word (16-bit) aligned and data must be in multiples of two bytes. If less data are written, the remaining bytes have to be filled by 0xFF.

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

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

The master sends the bytes to the STM32 as follows:

- Start of frame: 0x5A
- Byte 1: 0x31
- Byte 2: 0xCE
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Byte 3 to byte 6: start address
  - Byte 3: MSB
  - Byte 6: LSB
- Byte 7: checksum: XOR (byte3, byte4, byte5, byte6)
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- 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)

Figure 16. Write Memory command: master side

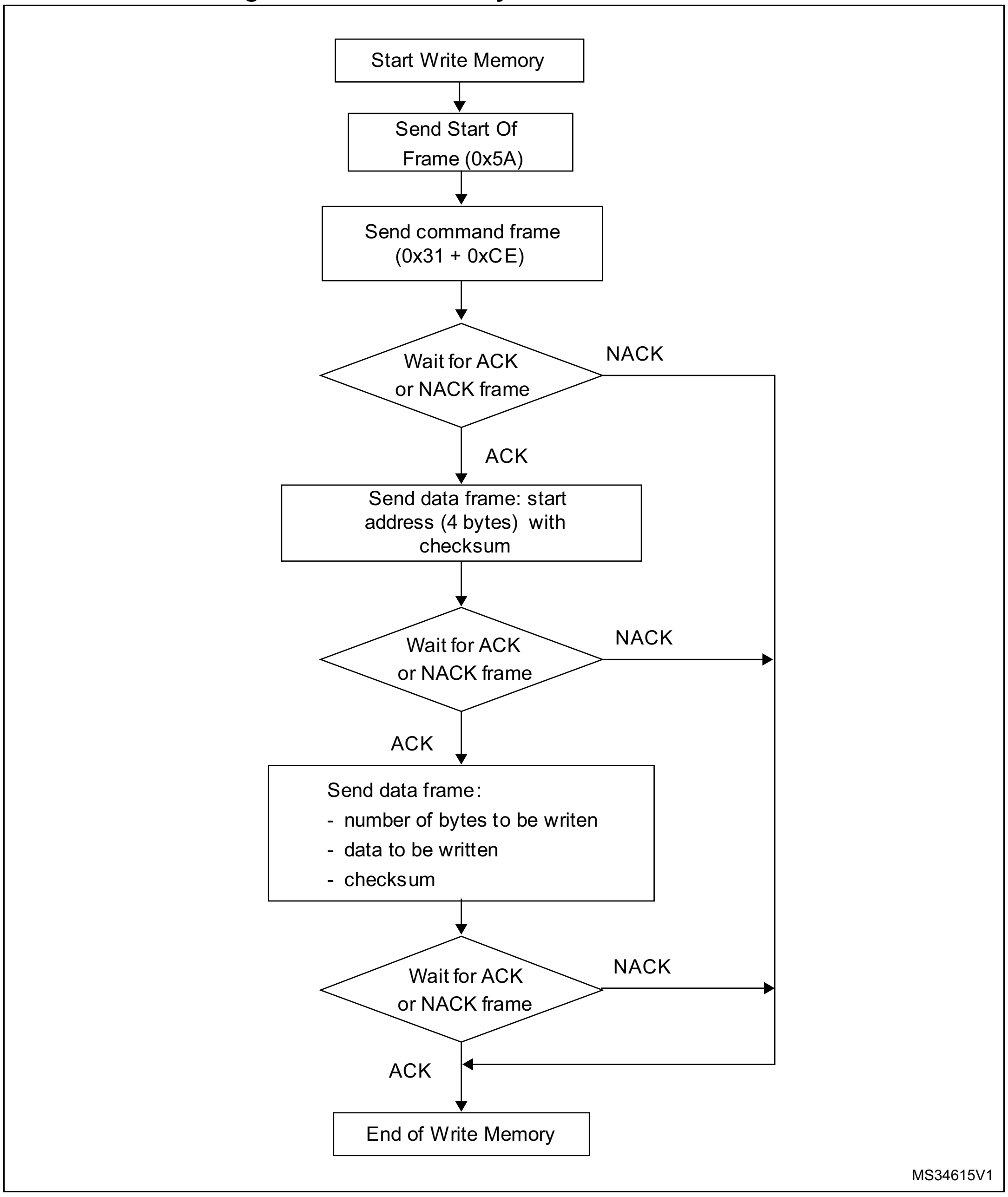
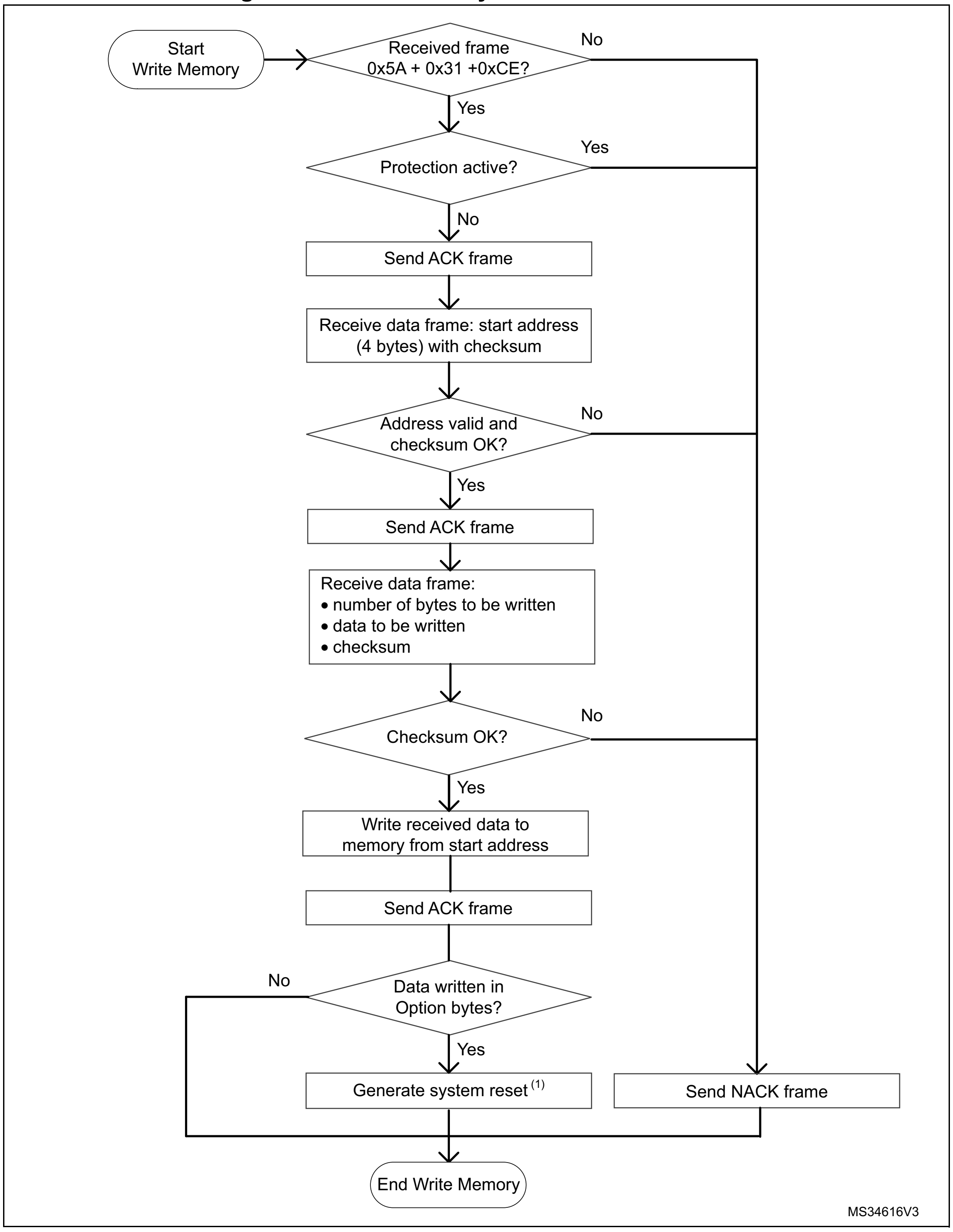

Start Write Memory
Send Start Of Frame (0x5A)
Send command frame (0x31 + 0xCE)
Wait for ACK or NACK frame
NACK
ACK
Send data frame: start address (4 bytes) with checksum
Wait for ACK or NACK frame
NACK
ACK
Send data frame:
- number of bytes to be writen
- data to be written
- checksum
Wait for ACK or NACK frame
NACK
ACK
End of Write Memory
MS34615V1

Figure 17. Write Memory command: slave side


Start Write Memory
Received frame 0x5A + 0x31 +0xCE?
No
Yes
Protection active?
Yes
No
Send ACK frame
Receive data frame: start address (4 bytes) with checksum
Address valid and checksum OK?
No
Yes
Send ACK frame
Receive data frame:
• number of bytes to be written
• data to be written
• checksum
Checksum OK?
No
Yes
Write received data to memory from start address
Send ACK frame
Data written in Option bytes?
No
Yes
Generate system reset (1)
Send NACK frame
End Write Memory
MS34616V3

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

2.8 Erase Memory command

This command allows the master to erase the flash memory pages or sectors using two-byte addressing mode. When the bootloader receives the command, it transmits the ACK byte to the master. After transmission of the ACK byte, the bootloader receives two bytes (number of pages or sectors 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 master. Otherwise, it sends an NACK byte to the master and the command is aborted.

Erase Memory command specifications

The bootloader receives two bytes that contain N, the number of pages or sectors to erase.

- For N = 0xFFFY (where Y is from 0 to F) a special erase is performed.
  - 0xFFFF for a global mass erase.
  - 0xFFFE for bank 1 mass erase (only for products supporting this feature).
  - 0xFFFD for bank 2 mass erase (only for products supporting this feature).
  - Values from 0xFFFC to 0xFFF0 are reserved.
- For values where $0 \leq N <$ maximum number of pages or sectors: N + 1 pages or sectors are erased.

The bootloader then receives the following:

- For a special erase, one byte: checksum of the previous bytes (such as 0x00 for 0xFFFF).
- For N+1 pages or sector erase, 2 x (N + 1) bytes, each half-word containing a page number (coded on two bytes, MSB first), followed by 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 or sectors is product-related, and must be respected.

The master sends bytes to the STM32 as follows:

- Mass/bank erase
  - Start of frame: 0x5A
  - Byte 1: 0x44
  - Byte 2: 0xBB
  - Wait for ACK (as described in *Section 1*)
  - Bytes 3-4: 0xFFFF for mass erase, 0xFFFE for bank1 erase, 0xFFFD for bank2 erase
  - Byte 5: checksum of bytes 3-4
  - Wait for ACK or NACK
- Page erase
  - Start of frame: 0x5A
  - Byte 1: 0x44
  - Byte 2: 0xBB
  - Wait for ACK (as described in *Section 1*)
  - Bytes 3-4: N (number of pages/sectors to be erased - 1, data sent MSB first)
  - Bytes 5: checksum of bytes 3-4
  - Wait for ACK or NACK
  - Remaining bytes: 2 x (N + 1) bytes + 1 byte checksum of all bytes (page numbers are coded on two bytes, MSB first)
  - Wait for ACK

Note: On some products (the older ones), the ACK byte is received before the end of erase operation. This is because the CPU stalls during the mass/bank/sectors erase operation, while SPI DMA on TX pin continues to run in circular mode. This induces sending unexpected latest data on the TX buffer (ACK).
Correction is to clear (fill DMA buffer with all BUSY bytes) before executing the erase.

Figure 18. Erase Memory command: master side

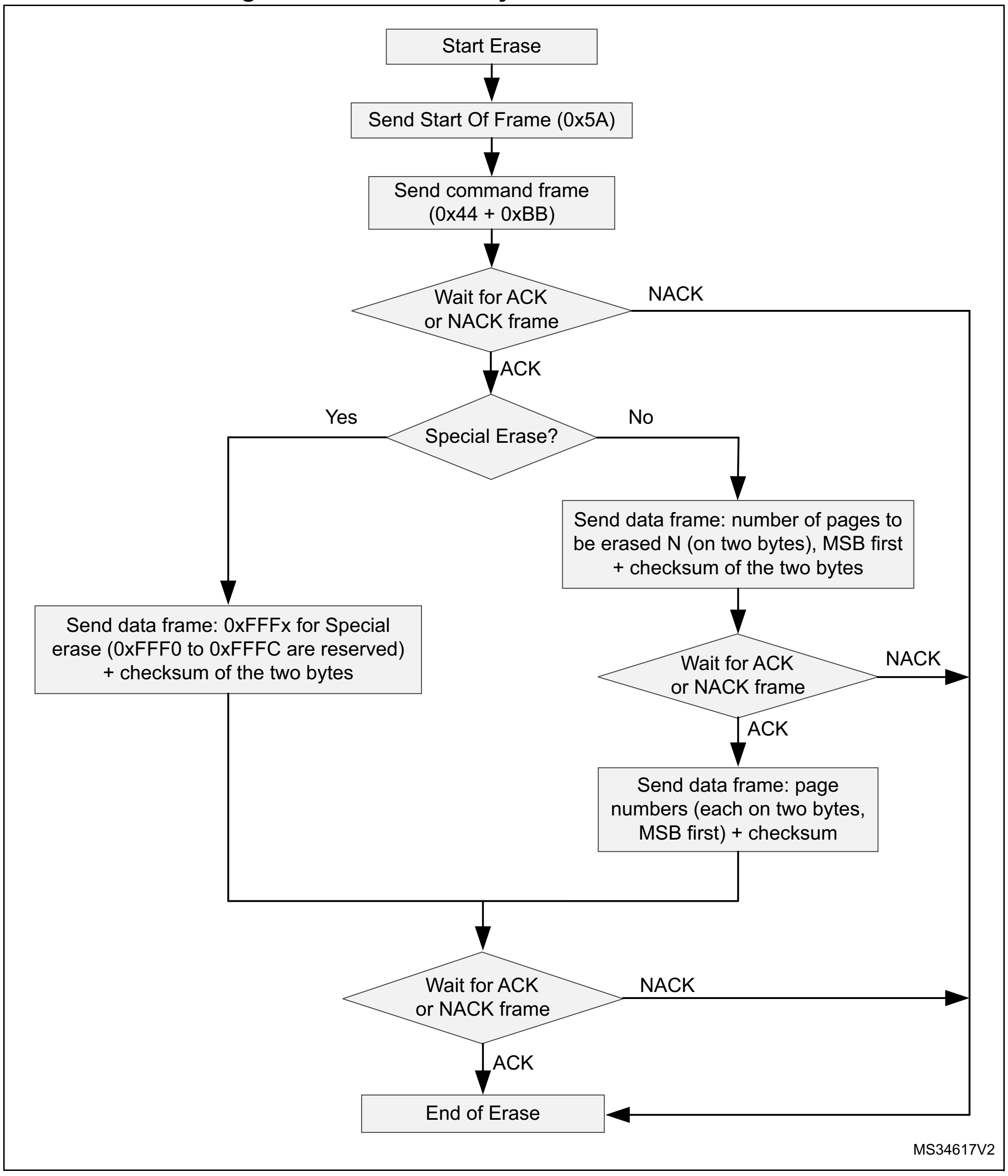
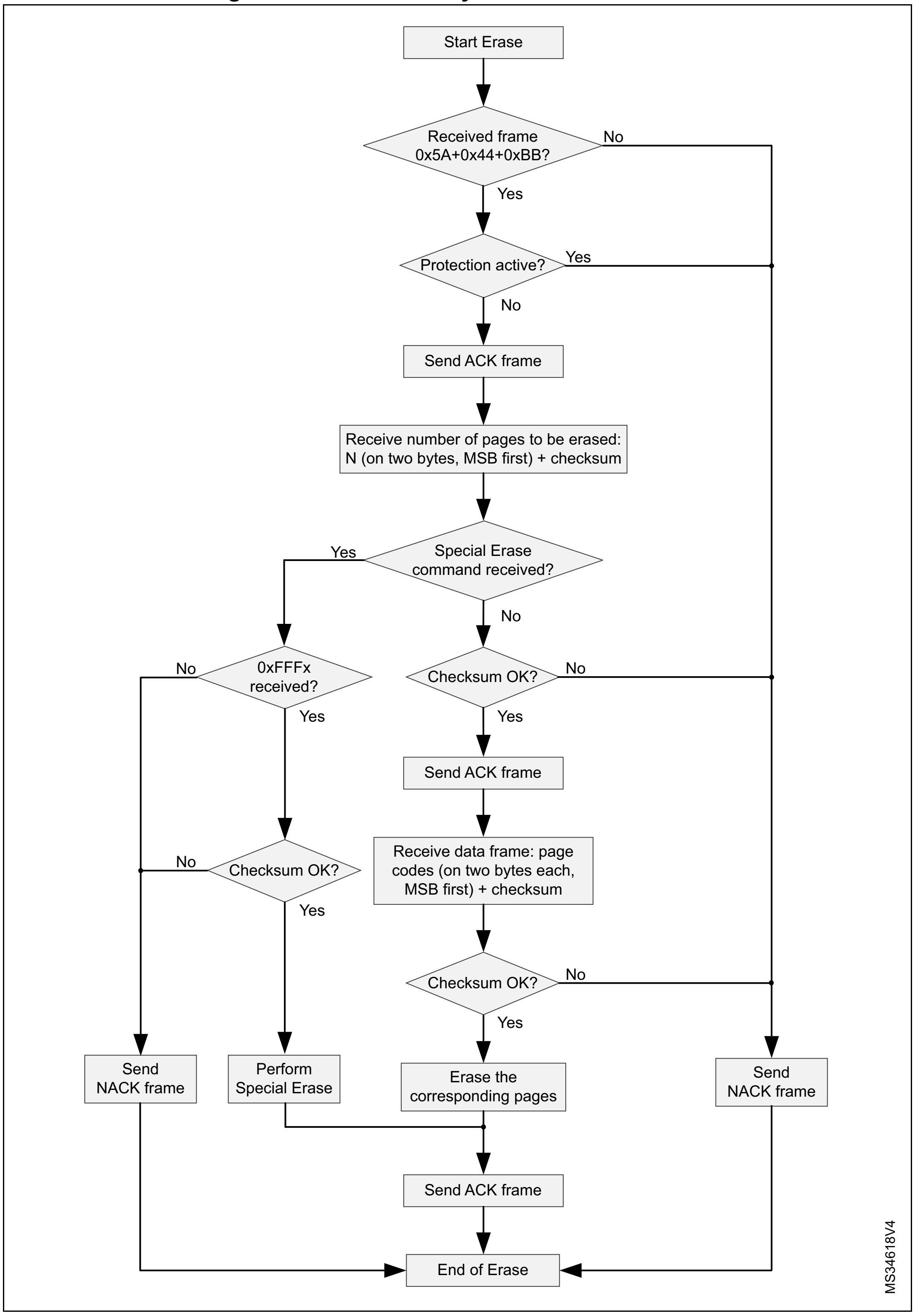

Start Erase
Send Start Of Frame (0x5A)
Send command frame (0x44 + 0xBB)
Wait for ACK or NACK frame
NACK
ACK
Special Erase?
Yes
No
Send data frame: number of pages to be erased N (on two bytes), MSB first + checksum of the two bytes
Send data frame: 0xFFFx for Special erase (0xFFF0 to 0xFFFC are reserved) + checksum of the two bytes
Wait for ACK or NACK frame
NACK
ACK
Send data frame: page numbers (each on two bytes, MSB first) + checksum
Wait for ACK or NACK frame
NACK
ACK
End of Erase
MS34617V2

Figure 19. Erase Memory command: slave side


Start Erase
Received frame 0x5A+0x44+0xBB?
No
Yes
Protection active?
Yes
No
Send ACK frame
Receive number of pages to be erased: N (on two bytes, MSB first) + checksum
Special Erase command received?
Yes
No
0xFFFx received?
No
Yes
Checksum OK?
No
Yes
Send ACK frame
Checksum OK?
No
Yes
Receive data frame: page codes (on two bytes each, MSB first) + checksum
Checksum OK?
No
Yes
Send NACK frame
Perform Special Erase
Erase the corresponding pages
Send NACK frame
Send ACK frame
End of Erase
MS34618V4

2.9 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 master, 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 command, the bootloader transmits the ACK byte, and generates a system reset to take into account the new configuration of the option byte.

The Write Protect command sequence is as follows:

- the bootloader receives one byte containing 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, consequently 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 protected by the first command become unprotected, and only the sectors passed within the second Write Protect command become protected.

Figure 20. Write Protect command: master side

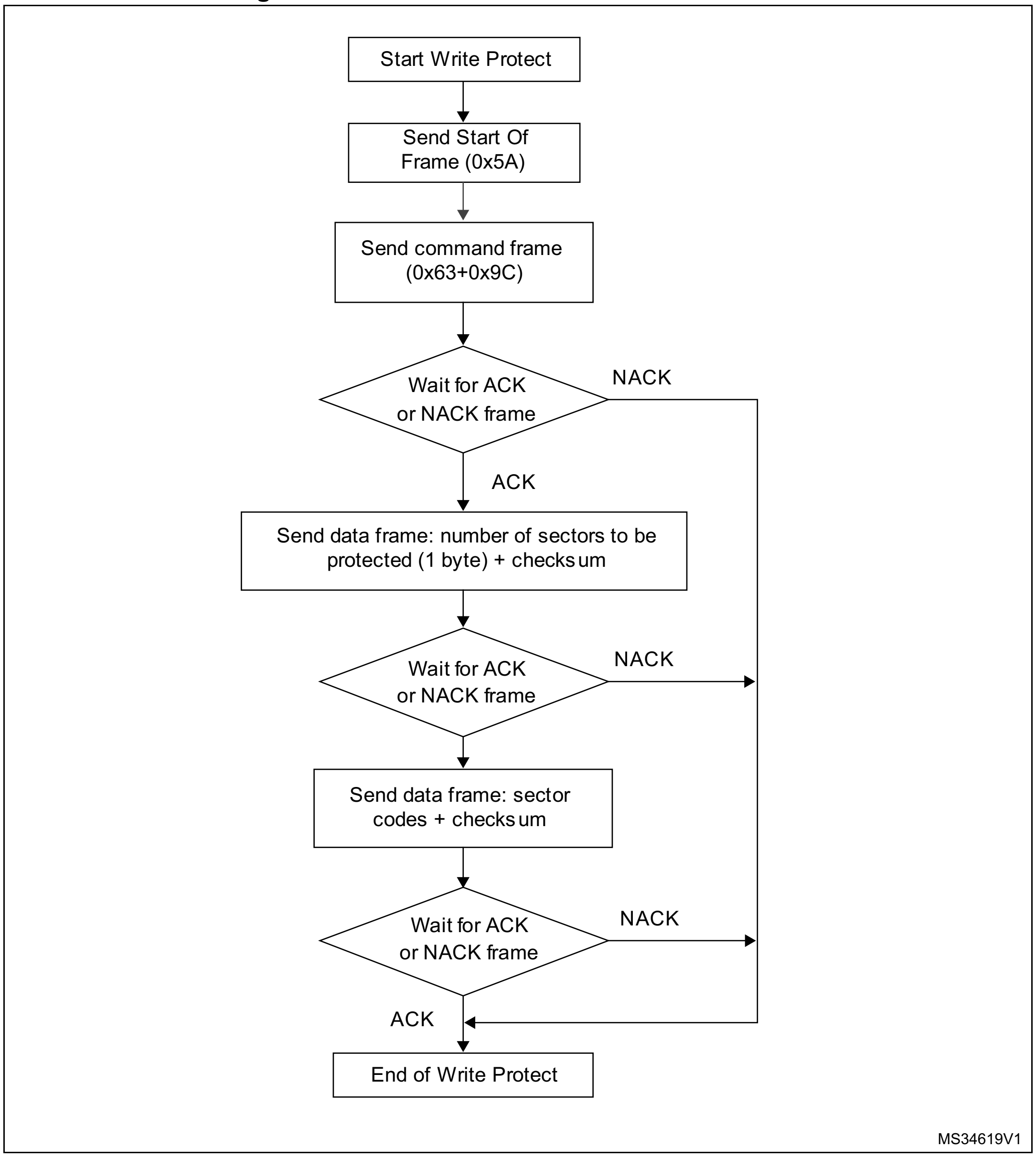
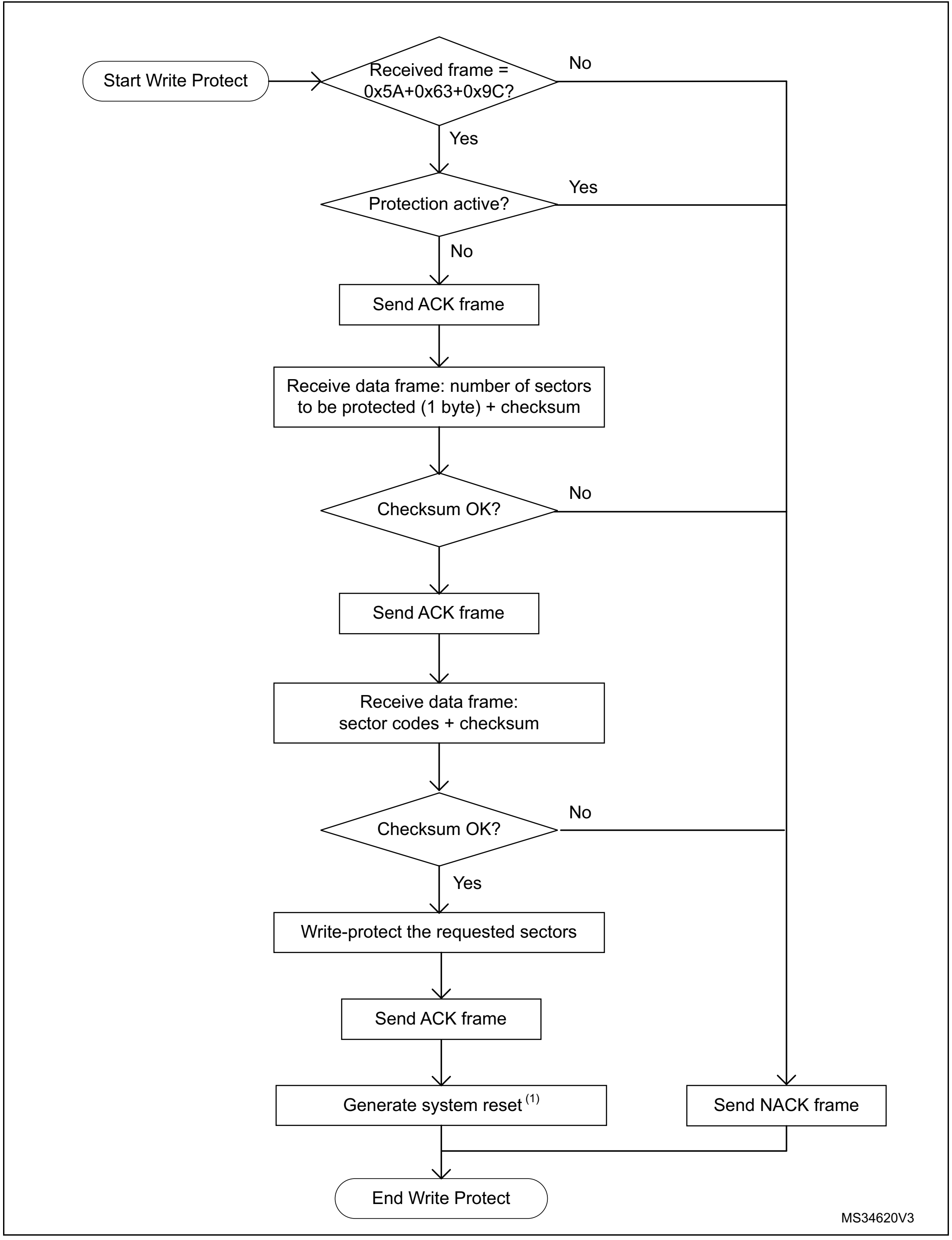

Start Write Protect
Send Start Of Frame (0x5A)
Send command frame (0x63+0x9C)
Wait for ACK or NACK frame
NACK
ACK
Send data frame: number of sectors to be protected (1 byte) + checksum
Wait for ACK or NACK frame
NACK
Send data frame: sector codes + checksum
Wait for ACK or NACK frame
NACK
ACK
End of Write Protect
MS34619V1

Figure 21. Write Protect command: slave side


Start Write Protect
Received frame = 0x5A+0x63+0x9C?
No
Yes
Protection active?
Yes
No
Send ACK frame
Receive data frame: number of sectors to be protected (1 byte) + checksum
Checksum OK?
No
Send ACK frame
Receive data frame: sector codes + checksum
Checksum OK?
No
Yes
Write-protect the requested sectors
Send ACK frame
Generate system reset (1)
Send NACK frame
End Write Protect
MS34620V3

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

2.10 Write Unprotect command

This command is used to disable the write protection of all the flash memory sectors. When the bootloader receives the command, it transmits the ACK byte to the master. After transmission of the ACK byte, the bootloader disables the write protection of all the flash memory sectors. After the operation, the bootloader transmits the ACK byte, and generates a system reset to take into account the new configuration of the option byte.

Figure 22. Write Unprotect command: master side

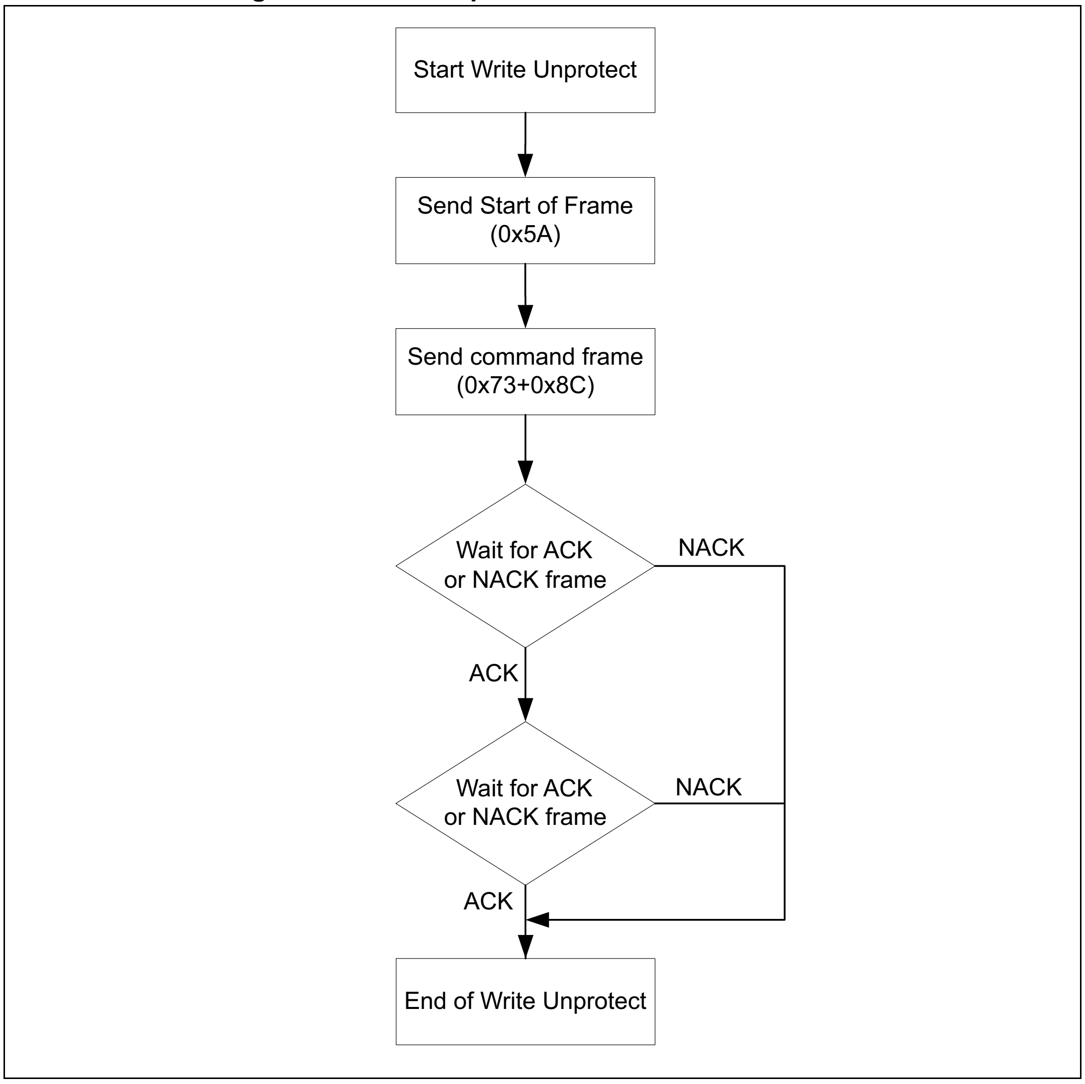
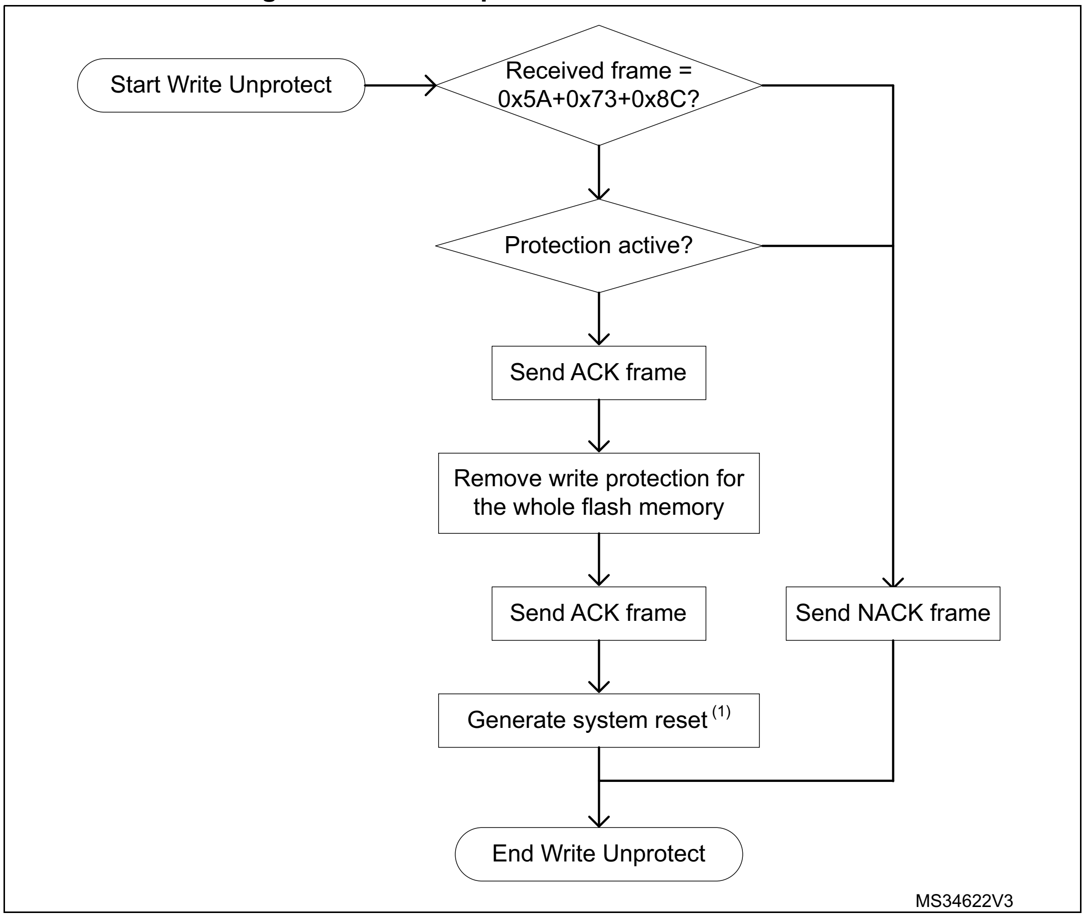

Start Write Unprotect
Send Start of Frame (0x5A)
Send command frame (0x73+0x8C)
Wait for ACK or NACK frame
NACK
ACK
Wait for ACK or NACK frame
NACK
ACK
End of Write Unprotect

Figure 23. Write Unprotect command: slave side



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

2.11 Readout Protect command

This 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 master. After transmission of the ACK byte, the bootloader enables the read protection.

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

Figure 24. Readout Protect command: master side

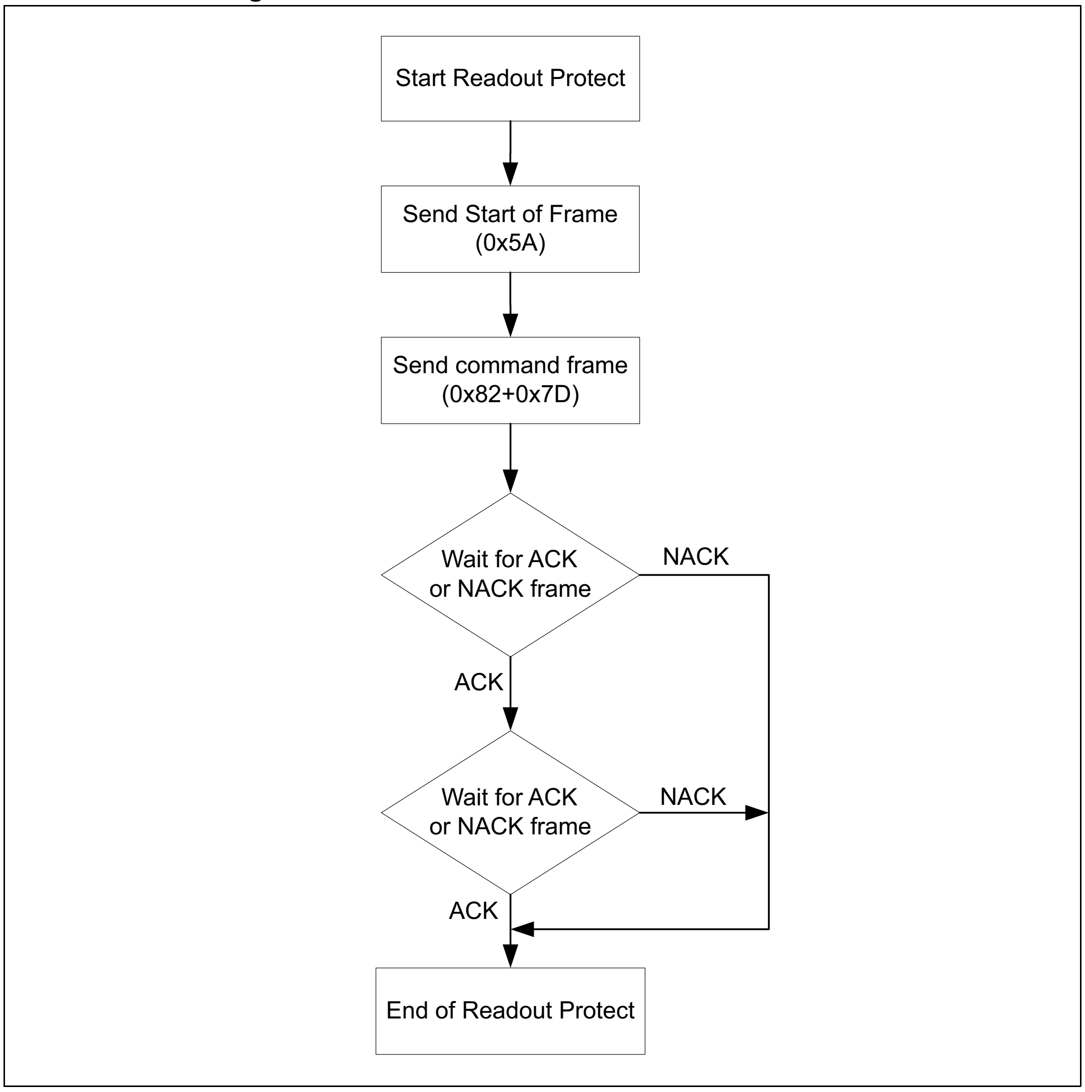
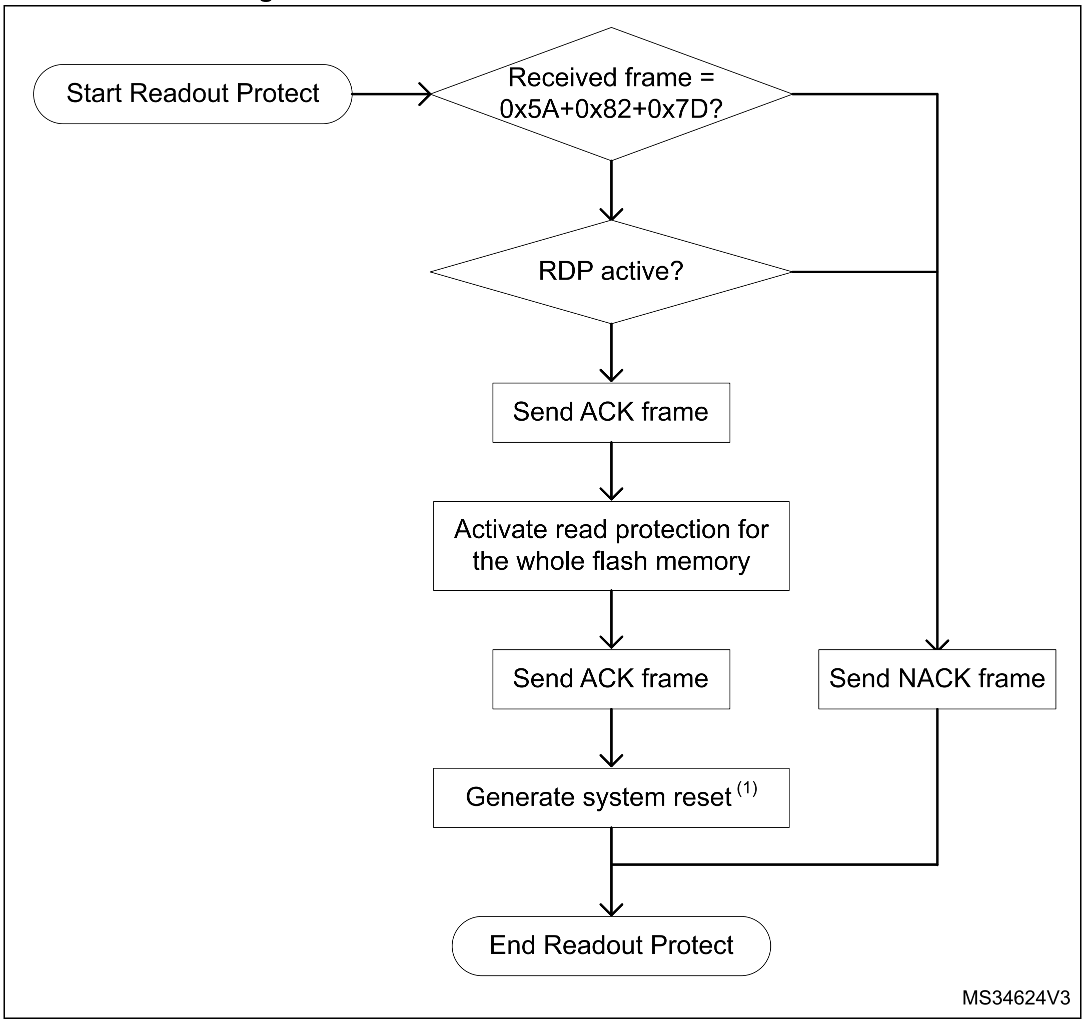

Start Readout Protect
Send Start of Frame (0x5A)
Send command frame (0x82+0x7D)
Wait for ACK or NACK frame
NACK
ACK
Wait for ACK or NACK frame
NACK
ACK
End of Readout Protect

Figure 25. Readout Protect command: slave side


Start Readout Protect
Received frame = 0x5A+0x82+0x7D?
RDP active?
Send ACK frame
Activate read protection for the whole flash memory
Send ACK frame
Send NACK frame
Generate system reset (1)
End Readout Protect
MS34624V3

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

2.12 Readout Unprotect command

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

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

The master must wait enough time for the read protection disable (equivalent to the Mass erase time on most products, see the product datasheet for more information) before polling for a slave response.

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.

Figure 26. Readout Unprotect command: master side

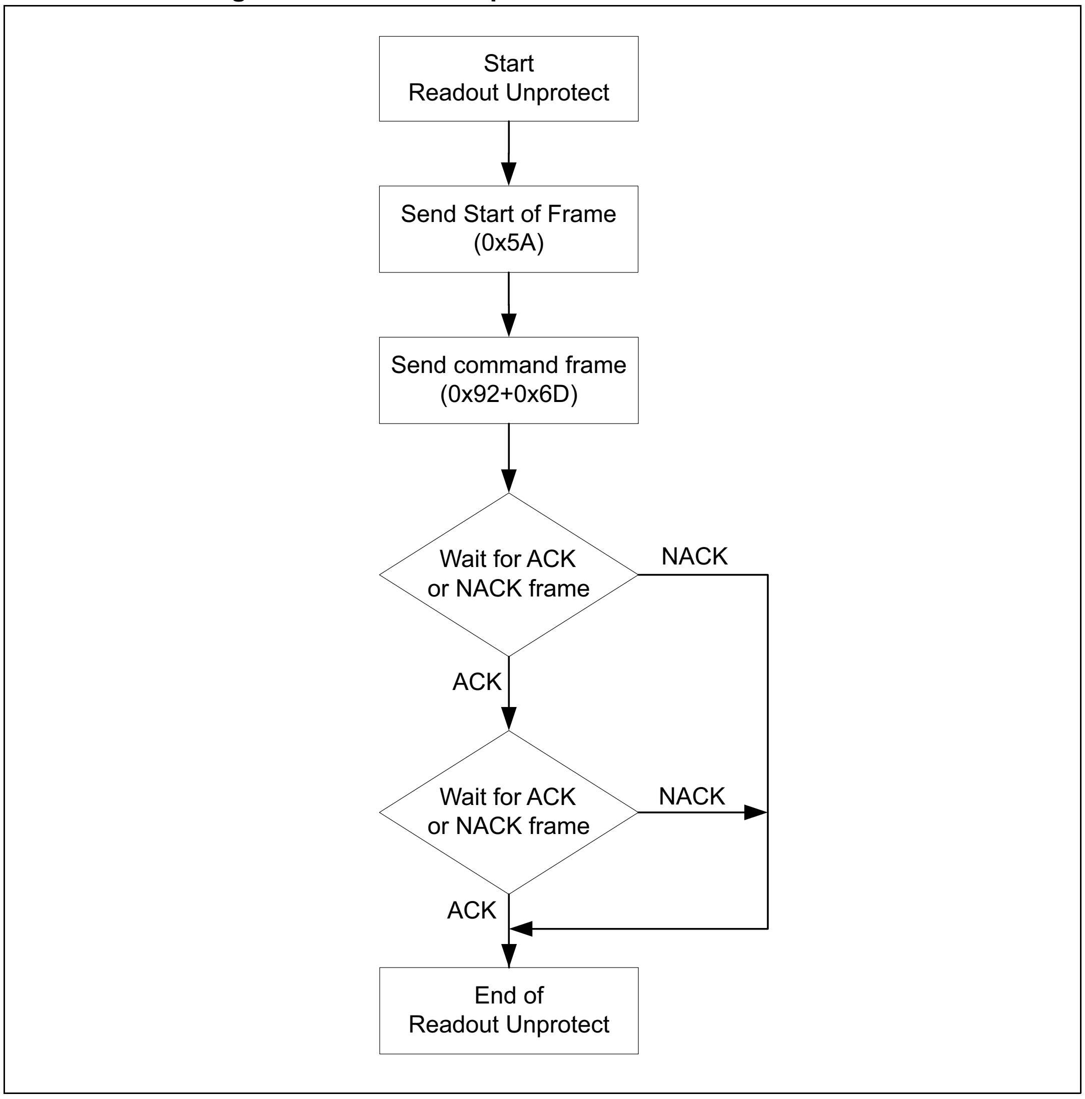
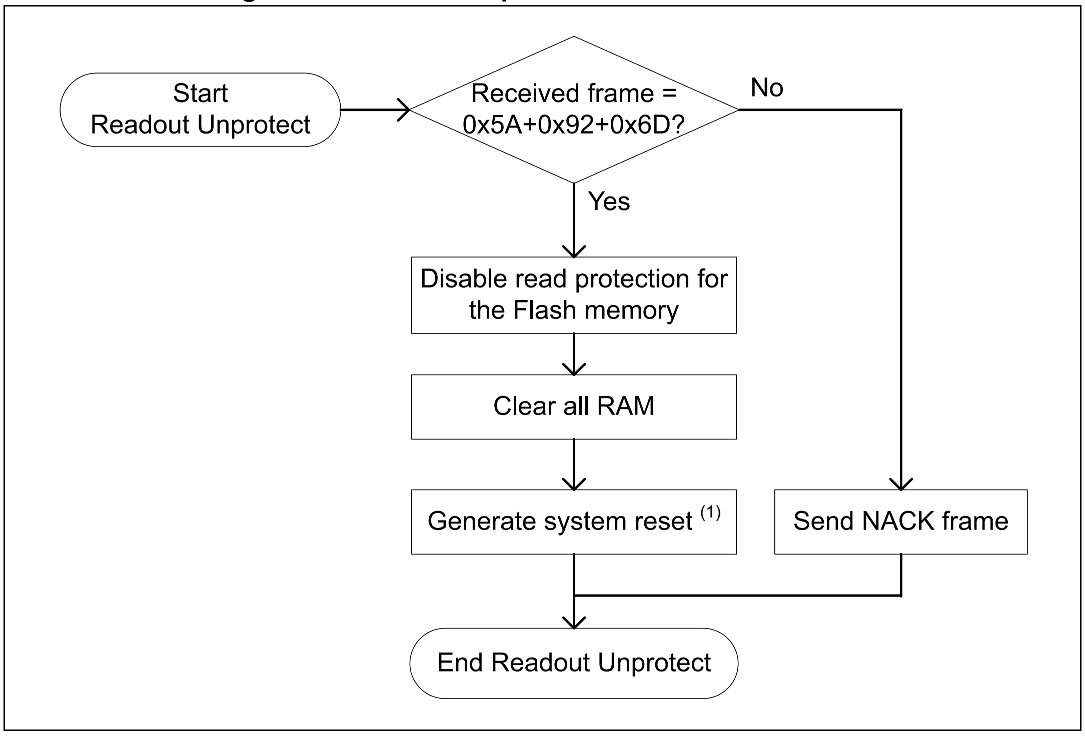


Figure 27. Readout Unprotect command: slave side


Start
Readout Unprotect
Received frame = 0x5A+0x92+0x6D?
No
Yes
Disable read protection for the Flash memory
Clear all RAM
Generate system reset (1)
Send NACK frame
End Readout Unprotect

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

2.13 Get Checksum command

This 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 command, it transmits the ACK byte to the application, 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 an 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, expressed in 32-bit words (4 bytes) number and their complement byte (checksum).

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

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

If the memory size is correct, the bootloader sends an 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 an 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 a product does not support CRC initialization value change the value is ignored, but the device still sends ACK.

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

If the checksum is correct, the bootloader computes the CRC of the given memory, and 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 (as described in *Section 1: SPI bootloader code sequence*)

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 (as described in *Section 1: SPI bootloader code sequence*)

Bytes 8 to 11 Memory area size (number of 32-bits words): byte 8: MSB, byte 11: LSB

Byte 12: Checksum: XOR (byte 8, byte 9, byte 10, byte 11)

Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)

Byte 13 to 16: CRC polynomial: byte 13: MSB, byte 16: LSB

Byte 17: Checksum: XOR (byte 13, byte 14, byte 15, byte 16)

Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)

Bytes 18 to 21 CRC initialization value: byte 18: MSB, byte 21: LSB

Byte 22: Checksum: XOR (byte 18, byte 19, byte 20, byte 21)

Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)

Figure 28. Get Checksum command: host side

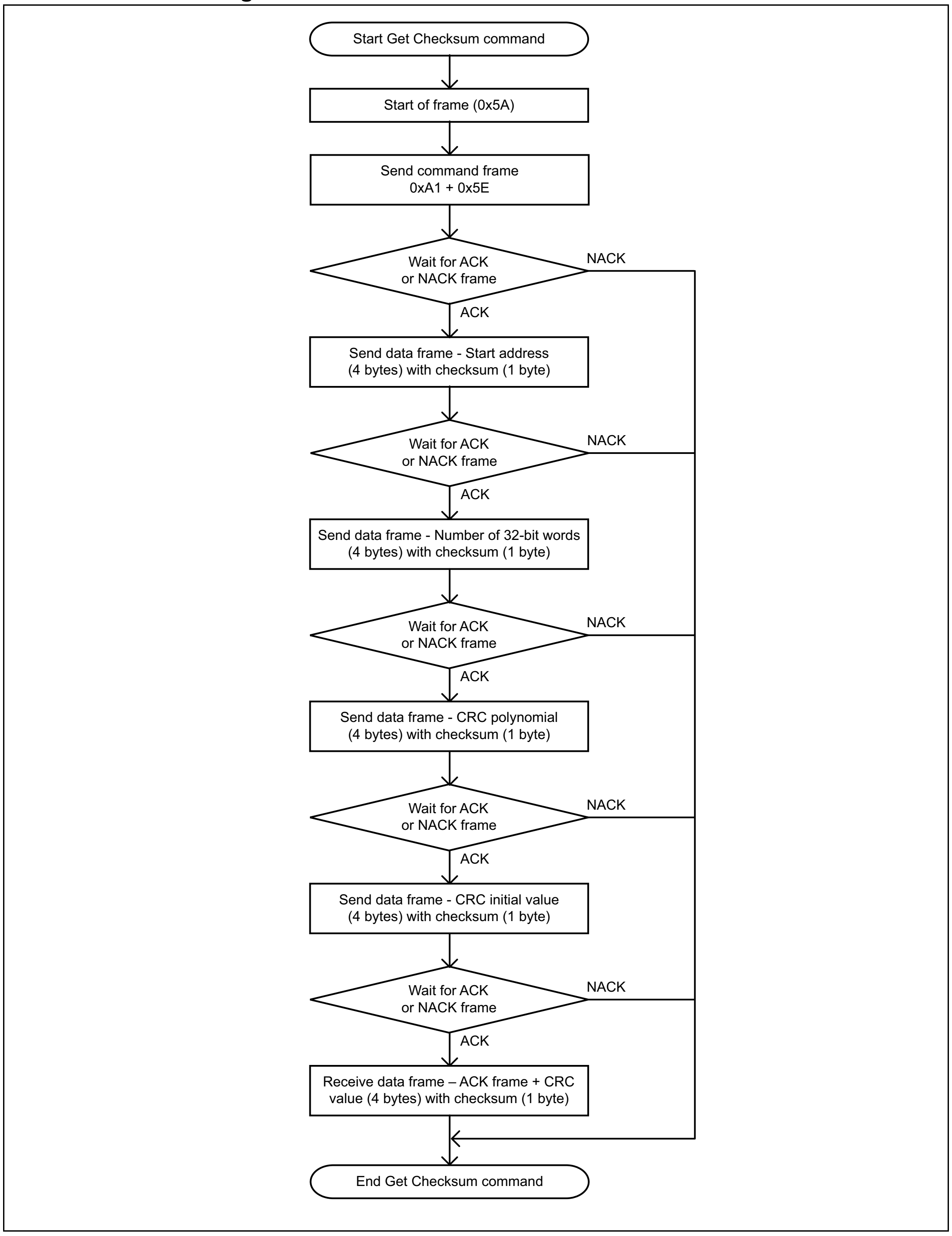
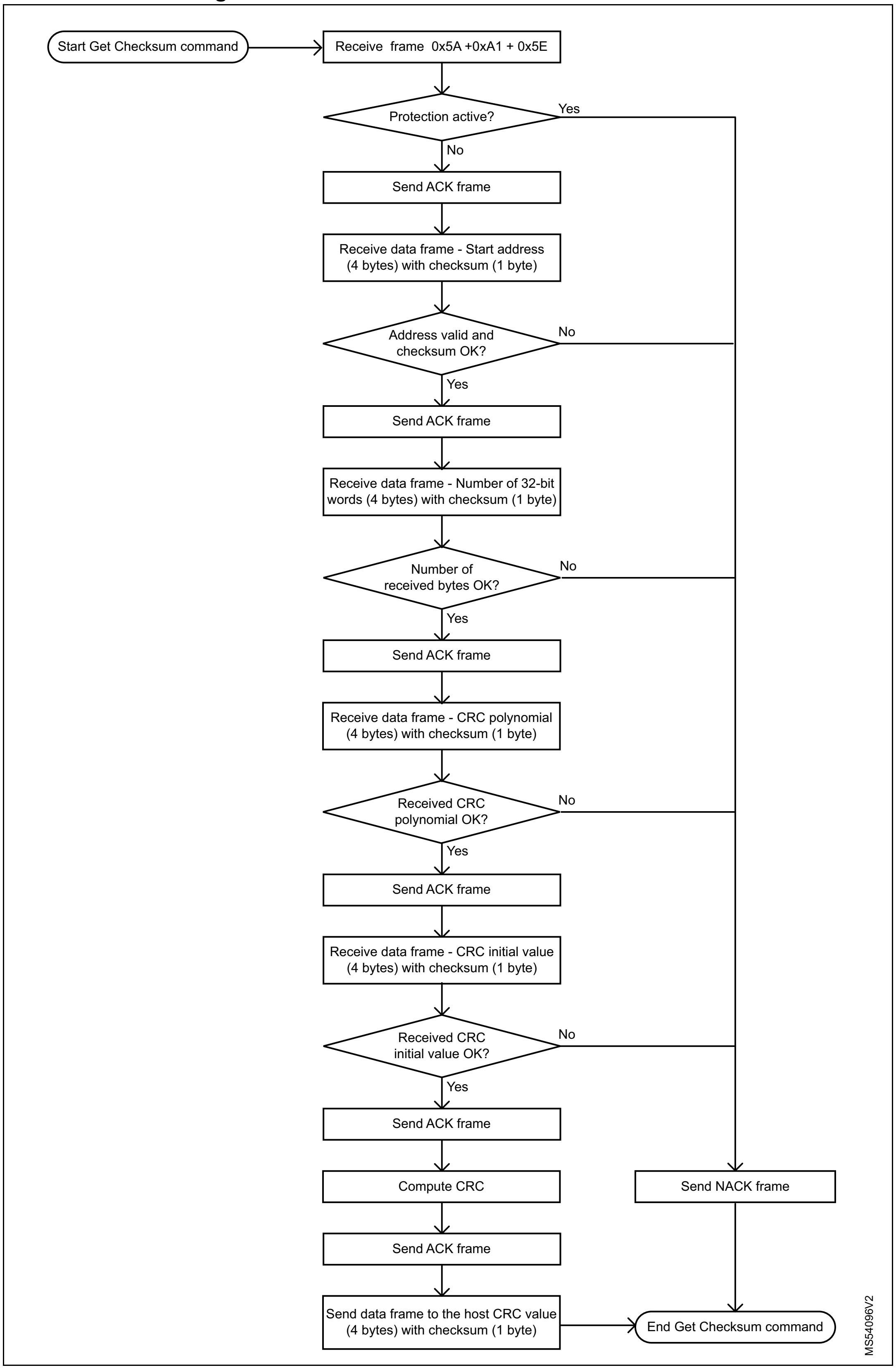

Start Get Checksum command
Start of frame (0x5A)
Send command frame
0xA1 + 0x5E
Wait for ACK
or NACK frame
NACK
ACK
Send data frame - Start address
(4 bytes) with checksum (1 byte)
Wait for ACK
or NACK frame
NACK
ACK
Send data frame - Number of 32-bit words
(4 bytes) with checksum (1 byte)
Wait for ACK
or NACK frame
NACK
ACK
Send data frame - CRC polynomial
(4 bytes) with checksum (1 byte)
Wait for ACK
or NACK frame
NACK
ACK
Send data frame - CRC initial value
(4 bytes) with checksum (1 byte)
Wait for ACK
or NACK frame
NACK
ACK
Receive data frame – ACK frame + CRC
value (4 bytes) with checksum (1 byte)
End Get Checksum command

Figure 29. Get Checksum command: device side


Start Get Checksum command
Receive frame 0x5A +0xA1 + 0x5E
Protection active?
Yes
No
Send ACK frame
Receive data frame - Start address (4 bytes) with checksum (1 byte)
Address valid and checksum OK?
No
Yes
Send ACK frame
Receive data frame - Number of 32-bit words (4 bytes) with checksum (1 byte)
Number of received bytes OK?
No
Yes
Send ACK frame
Receive data frame - CRC polynomial (4 bytes) with checksum (1 byte)
Received CRC polynomial OK?
No
Yes
Send ACK frame
Receive data frame - CRC initial value (4 bytes) with checksum (1 byte)
Received CRC initial value OK?
No
Yes
Send ACK frame
Compute CRC
Send NACK frame
Send ACK frame
Send data frame to the host CRC value (4 bytes) with checksum (1 byte)
End Get Checksum command
MS54096V2

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 30. Special command: host side

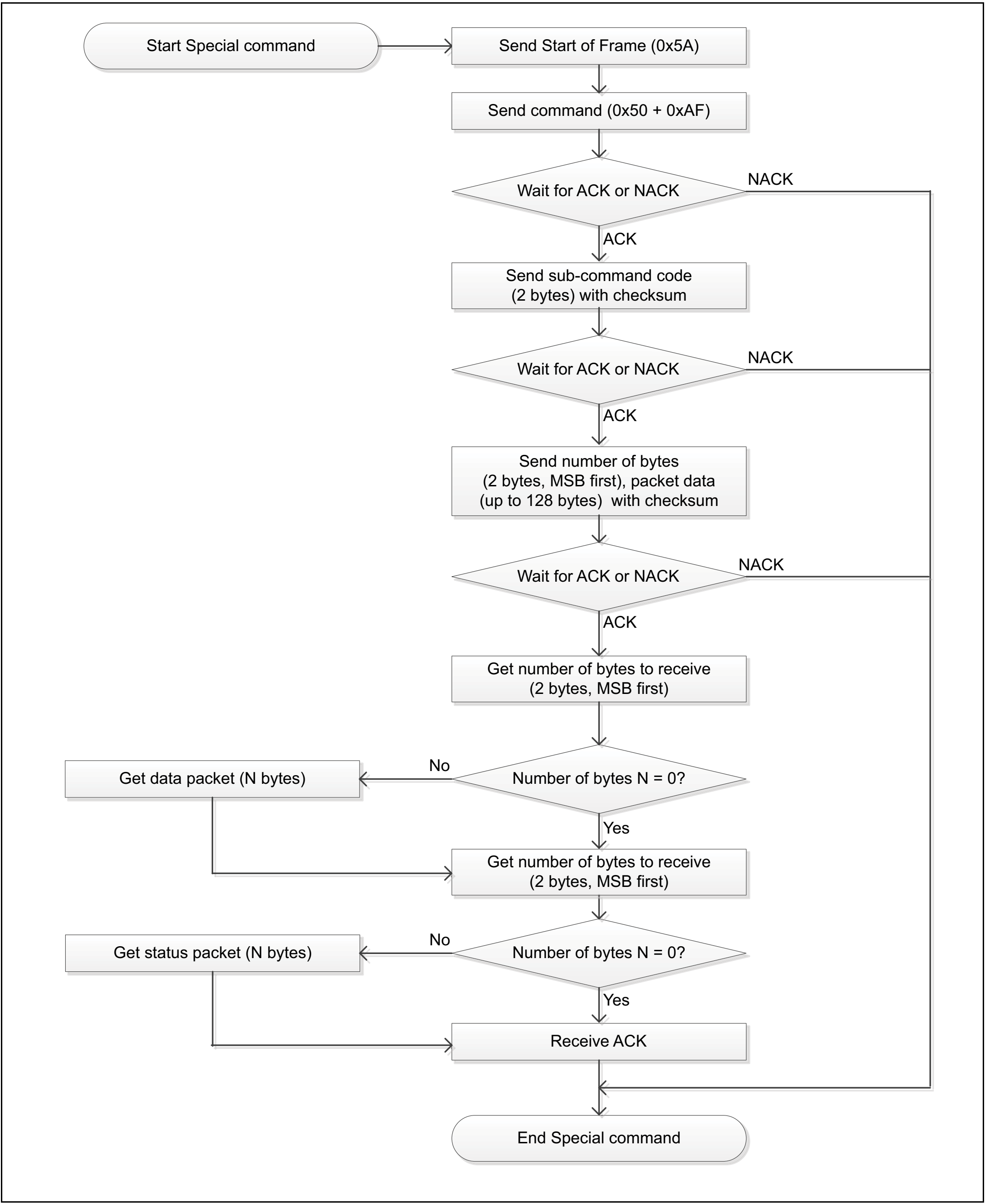
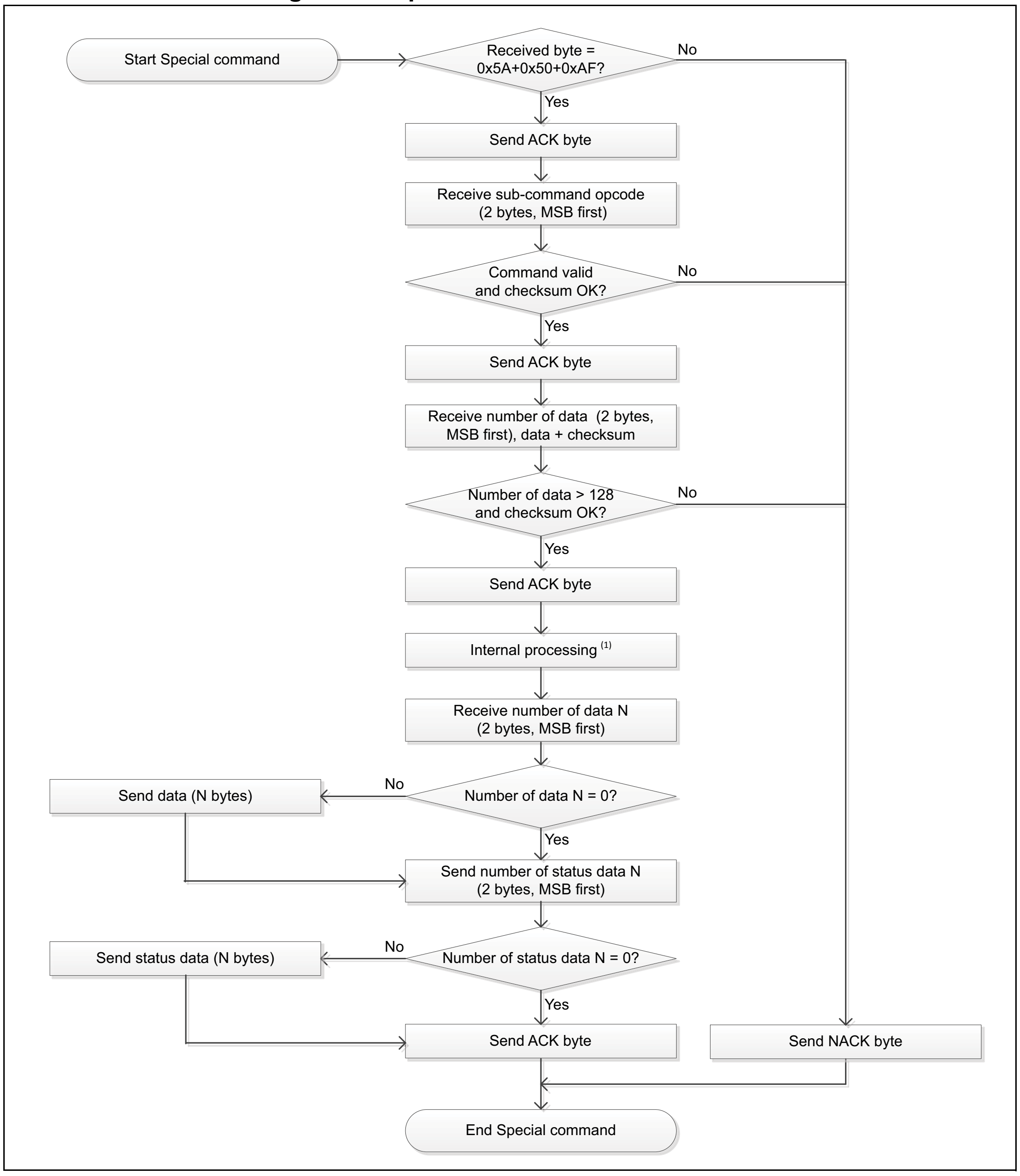

Start Special command
Send Start of Frame (0x5A)
Send command (0x50 + 0xAF)
Wait for ACK or NACK
NACK
ACK
Send sub-command code (2 bytes) with checksum
Wait for ACK or NACK
NACK
ACK
Send number of bytes (2 bytes, MSB first), packet data (up to 128 bytes) with checksum
Wait for ACK or NACK
NACK
ACK
Get number of bytes to receive (2 bytes, MSB first)
Get data packet (N bytes)
No
Number of bytes N = 0?
Yes
Get number of bytes to receive (2 bytes, MSB first)
Get status packet (N bytes)
No
Number of bytes N = 0?
Yes
Receive ACK
End Special command

Figure 31. Special command: device side


Start Special command
Received byte = 0x5A+0x50+0xAF?
No
Yes
Send ACK byte
Receive sub-command opcode (2 bytes, MSB first)
Command valid and checksum OK?
No
Yes
Send ACK byte
Receive number of data (2 bytes, MSB first), data + checksum
Number of data > 128 and checksum OK?
No
Yes
Send ACK byte
Internal processing (1)
Receive number of data N (2 bytes, MSB first)
Send data (N bytes)
No
Number of data N = 0?
Yes
Send number of status data N (2 bytes, MSB first)
Send status data (N bytes)
No
Number of status data N = 0?
Yes
Send ACK byte
Send NACK byte
End Special command

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 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 upon 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 from 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 returns only the commands status.

Figure 32. Extended Special command: host side

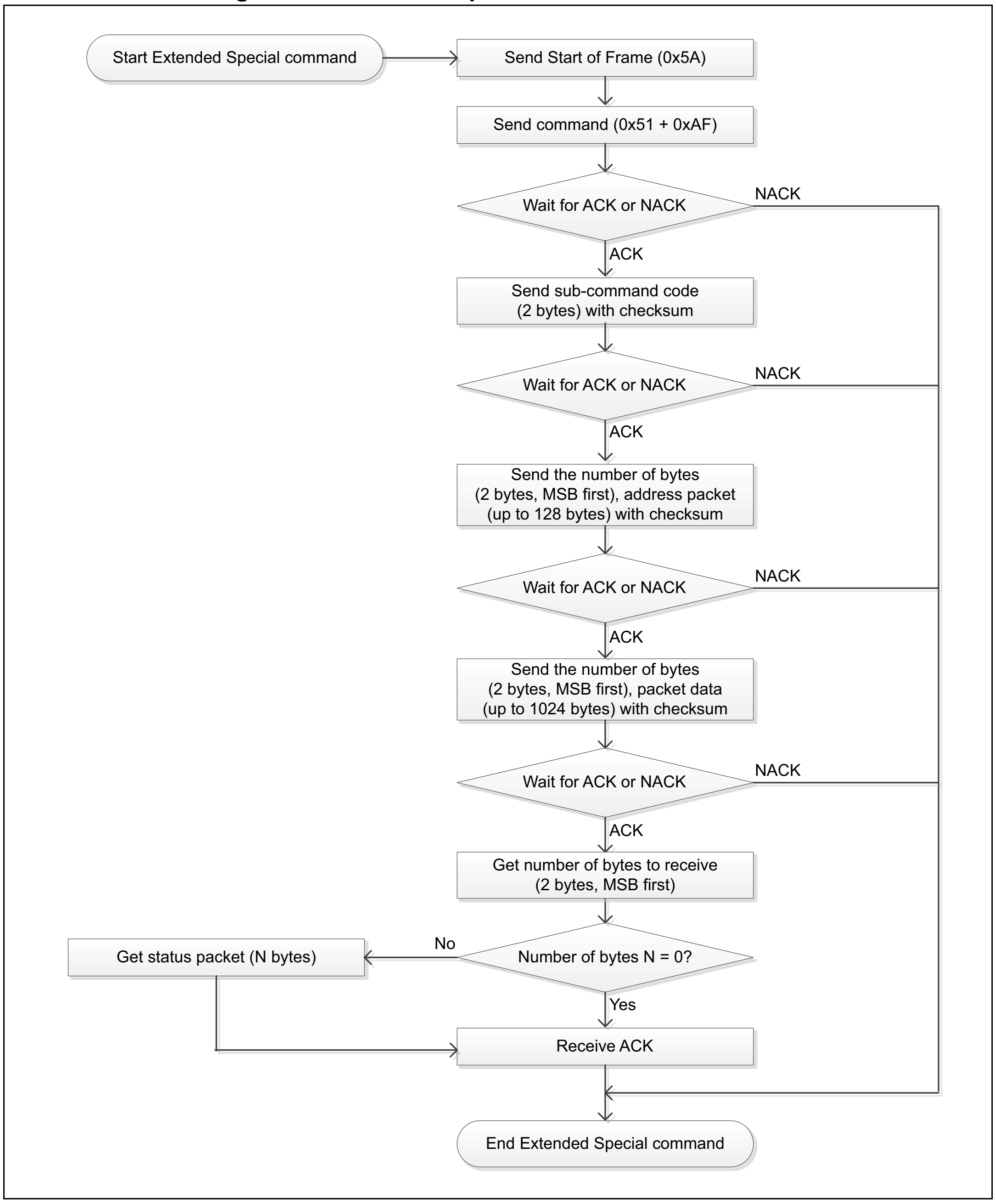
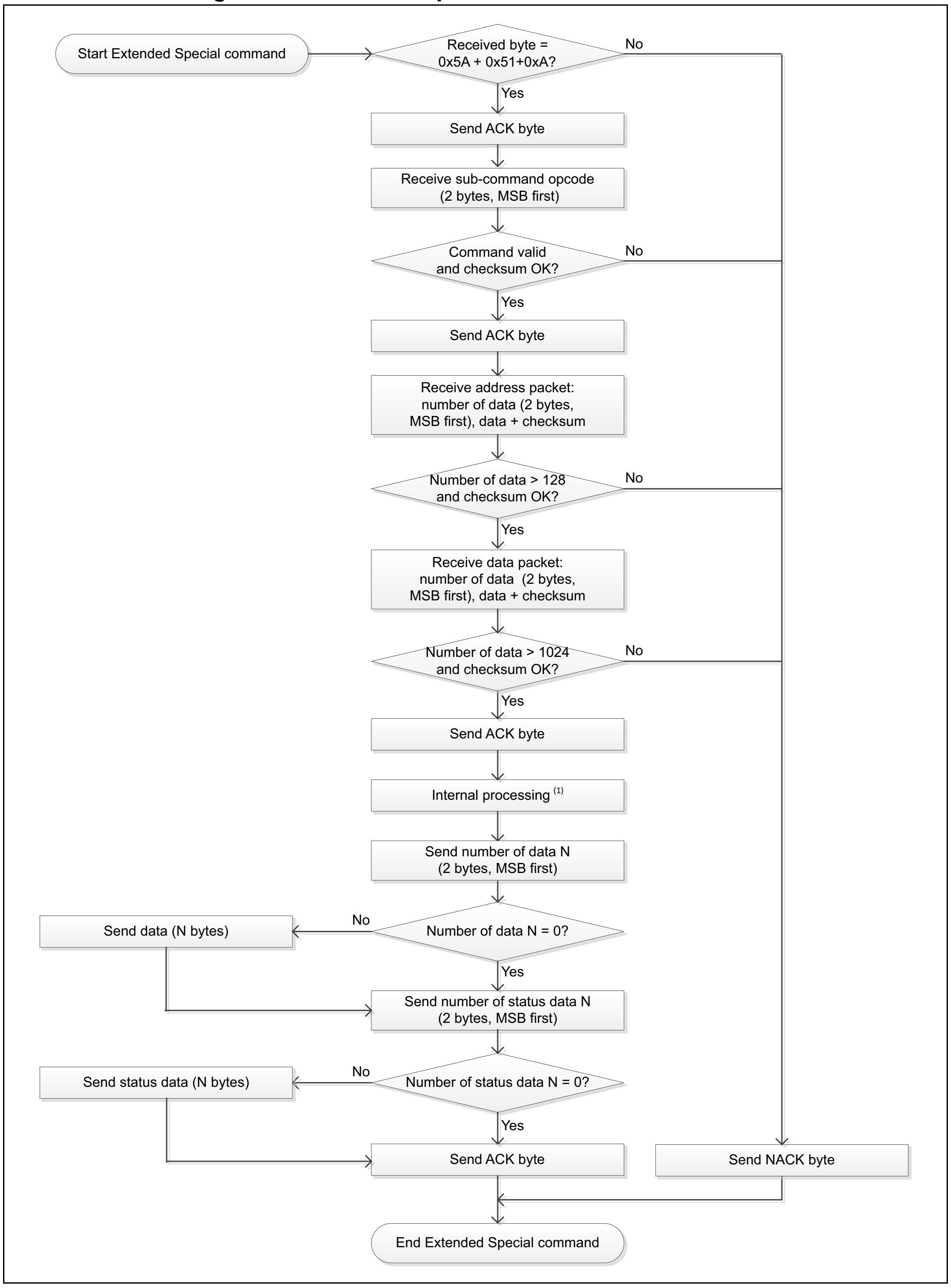

Start Extended Special command
Send Start of Frame (0x5A)
Send command (0x51 + 0xAF)
Wait for ACK or NACK
NACK
ACK
Send sub-command code (2 bytes) with checksum
Wait for ACK or NACK
NACK
ACK
Send the number of bytes (2 bytes, MSB first), address packet (up to 128 bytes) with checksum
Wait for ACK or NACK
NACK
ACK
Send the number of bytes (2 bytes, MSB first), packet data (up to 1024 bytes) with checksum
Wait for ACK or NACK
NACK
ACK
Get number of bytes to receive (2 bytes, MSB first)
Number of bytes N = 0?
No
Get status packet (N bytes)
Yes
Receive ACK
End Extended Special command

Figure 33. Extended Special command: device side



Start Extended Special command
Received byte = 0x5A + 0x51+0xA?
No
Yes
Send ACK byte
Receive sub-command opcode (2 bytes, MSB first)
Command valid and checksum OK?
No
Yes
Send ACK byte
Receive address packet: number of data (2 bytes, MSB first), data + checksum
Number of data > 128 and checksum OK?
No
Yes
Receive data packet: number of data (2 bytes, MSB first), data + checksum
Number of data > 1024 and checksum OK?
No
Yes
Send ACK byte
Internal processing (1)
Send number of data N (2 bytes, MSB first)
Send data (N bytes)
No
Number of data N = 0?
Yes
Send number of status data N (2 bytes, MSB first)
Send status data (N bytes)
No
Number of status data N = 0?
Yes
Send ACK byte
Send NACK byte
End Extended Special command

1. The internal processing depends on the project needs.

When the bootloader receives this 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: $N \le 128$ and checksum is correct, Packet2: $N \le 1024$ 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.

3 Evolution of the bootloader protocol versions

Table 3 lists the bootloader versions.

Table 3. Bootloader protocol versions

Version	Description
V1.0	– Initial bootloader version.
V1.1	– Updated the Acknowledge mechanism. – Updated the “Get”, “Get ID”, “Get Version” and “Read” commands.
V1.3	– Added support for “Get Checksum” command. – Updated “Get Command” command to return the opcode of the “Get Checksum” command.
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
27-Mar-2014	1	Initial release.
02-May-2014	2	Updated *Table 1: Applicable products*. Added footnote in *Table 2: SPI bootloader commands*. Updated *Section 2: Bootloader command set*. Updated *Figure 22*, *Figure 24* and *Figure 26*.
20-Oct-2016	3	Updated *Introduction* and *Table 1: Applicable products*. Updated *Figure 18: Erase Memory command: master side* and *Figure 19: Erase Memory command: slave side*.
10-Mar-2017	4	Updated *Table 1: Applicable products*.
15-Jan-2019	5	Updated *Table 1: Applicable products*. Updated *Section 1: SPI bootloader code sequence*. Minor text edits across the whole document.
05-Apr-2019	6	Updated *Table 1: Applicable products*.
24-Sep-2019	7	Updated *Table 1: Applicable products*.
27-Nov-2019	8	Updated *Table 1: Applicable products*.
03-Nov-2020	9	Updated *Section 2.2: Get command*. Updated *Table 2: SPI bootloader commands* and *Table 3: Bootloader protocol versions*. Added *Section 2.13: Get Checksum command*. Minor text edits across the whole document.
11-Jun-2021	10	Updated *Section 1: SPI bootloader code sequence* and *Section 2.13: Get Checksum command*. Added *Section 2.14: Special command* and *Section 2.15: Extended Special command*. Updated *Table 2: SPI bootloader commands*. Updated *Figure 17: Write Memory command: slave side*, *Figure 21: Write Protect command: slave side*, *Figure 23: Write Unprotect command: slave side*, *Figure 25: Readout Protect command: slave side*, *Figure 27: Readout Unprotect command: slave side*, and added footnotes to them. Updated *Figure 10: Get ID command: master side*, *Figure 11: Get ID command: slave side*, *Figure 22: Write Unprotect command: master side*, *Figure 24: Readout Protect command: master side*, *Figure 26: Readout Unprotect command: master side* and *Figure 28: Get Checksum command: host side*.
09-Feb-2022	11	Introduced STM32U5 series, hence updated *Table 1: Applicable products*. Updated *Section 2: Bootloader command set* and *Section 2.7: Write Memory command*.

Table 4. Document revision history

Date	Revision	Changes
14-Feb-2023	12	Updated *Introduction*, *Section 2.2: Get command*, and *Section 2.8: Erase Memory command*. Updated *Table 1: Applicable products*, *Table 2: SPI bootloader commands* and its footnotes, and *Table 3: Bootloader protocol versions*. Updated *Figure 6: Get command: master side*, *Figure 7: Get command: slave side*, *Figure 8: Get Version command: master side*, *Figure 9: Get Version command: slave side*, *Figure 13: Read Memory command: slave side*, *Figure 15: Go command: slave side*, *Figure 17: Write Memory command: slave side*, *Figure 19: Erase Memory command: slave side*, *Figure 21: Write Protect command: slave side*, *Figure 23: Write Unprotect command: slave side*, *Figure 25: Readout Protect command: slave side*, and *Figure 29: Get Checksum command: device side*. Minor text edits across the whole document.
24-Oct-2023	13	Updated *Section 2.2: Get command*. Updated *Figure 19: Erase Memory command: slave side*. Minor text edits across the whole document.
05-Mar-2024	14	Introduced STM32U0 and STM32WBA series, hence updated *Table 1: Applicable products*. Minor text edits across the whole document.
06-Feb-2025	15	Introduced STM32U3 series, hence updated *Table 1: Applicable products*. Minor text edits across the whole document.
16-Feb-2026	16	Added STM32C0 and STM32C5 series. Updated *Table 1: Applicable products*. Updated document title.