AN4286
Application note

SPI protocol used in the STM32 bootloader

Introduction

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

This document applies to the STM32 products embedding bootloader versions V8.x, V9.x, V11.x, V12.x and V13.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
Microcontrollers	STM32F4 Series STM32F7 Series STM32G0 Series STM32G4 Series STM32H7 Series STM32L0 Series STM32L4 Series STM32L5 Series STM32WB Series STM32WL Series

Contents

List of tables

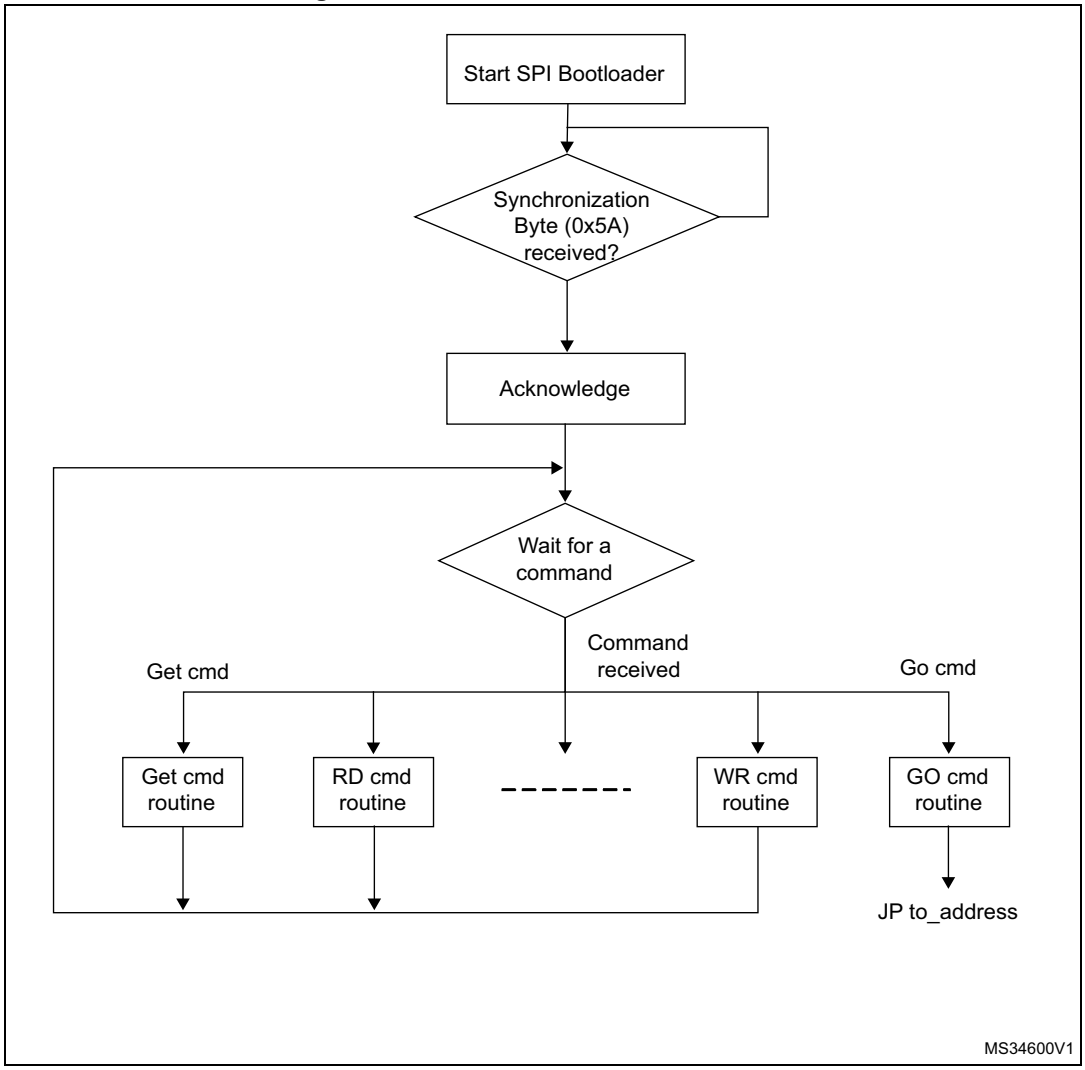
List of figures

1 SPI bootloader code sequence

The bootloader for STM32 microcontrollers, based on Arm®(a) core(s), is an SPI slave.

For all SPI bootloader operations, the NSS pin (chip select) must be low. If the NSS pin is high the communication on the SPI bus is ignored by the STM32 slave.

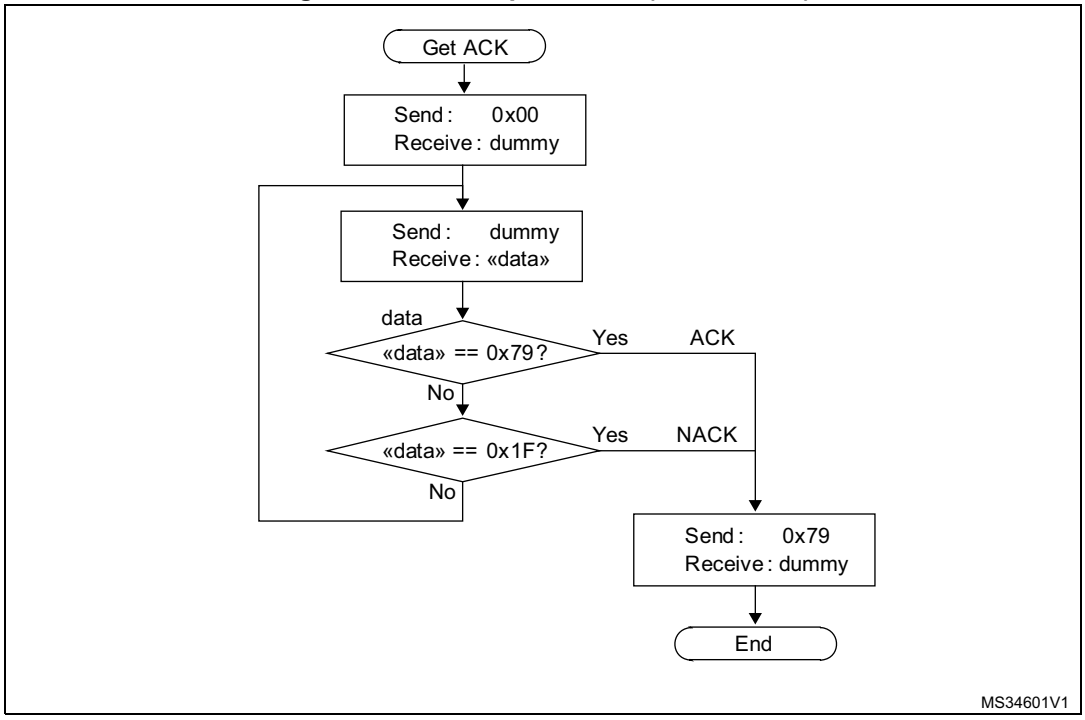
Figure 1. Bootloader for STM32 with SPI



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

Once the system memory boot mode is entered and the STM32 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 on the bus (0x5A). 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 to receive an acknowledge (ACK).

Figure 3. Bootloader SPI synchronization frame

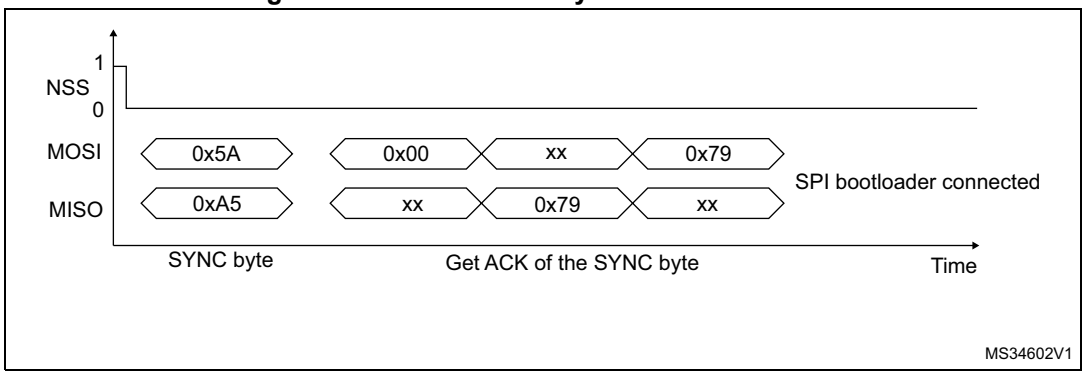
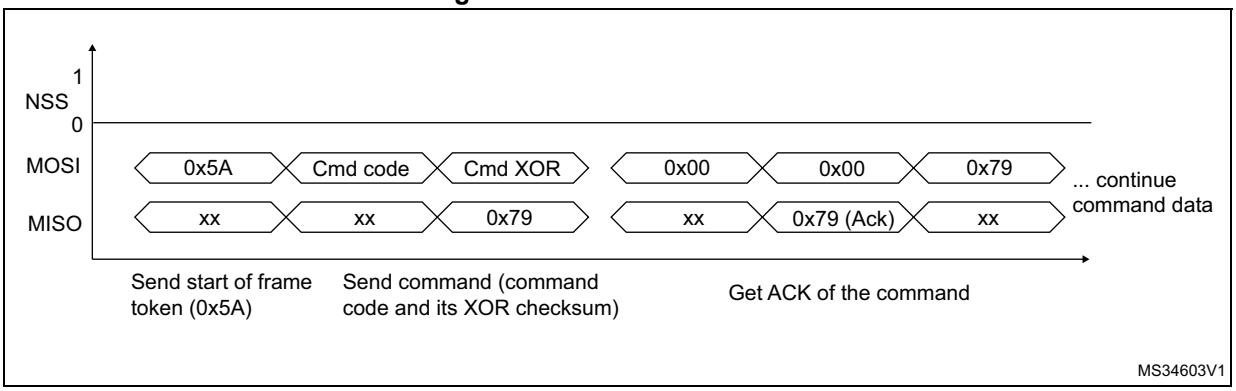
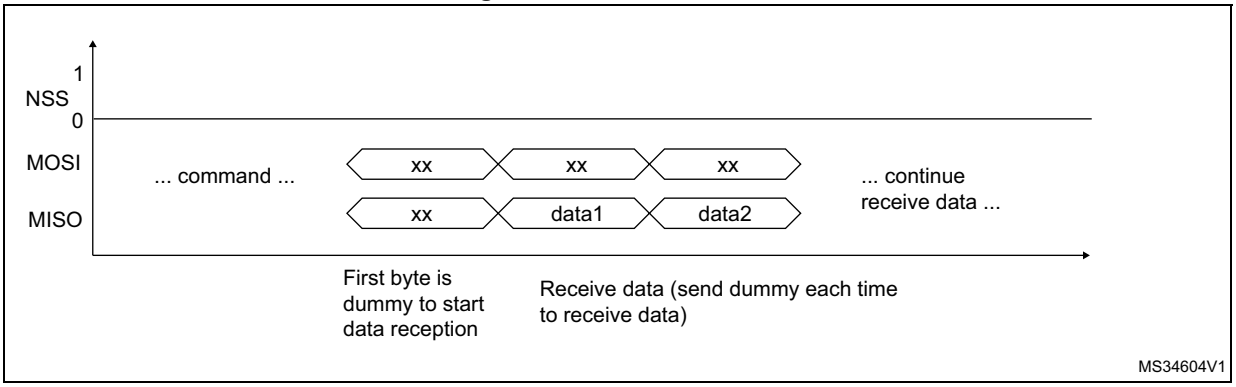


Figure 4. SPI command frame



To read any data sent by the slave the master must first send a dummy byte. This applies to all commands where a read is required.

Figure 5. Read data frame



2 Bootloader command set

Table 2 lists the supported commands. Each command is further 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 bootloader 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.
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 a 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 slave. Once the RDP has been removed, the other commands become active.
3. Refer to 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. Since 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 has to transmit data, the master sends its clock, so it has to transmit data on the MOSI line to be able to receive slave 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 plus checksum, the result at the end of the packet must be 0x00.
- If the received data is one byte, then 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 received bytes 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 transmitted bytes depends on the command.

2.2 Get command

The Get command enables 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 master, as described in *Figure 6*.

Figure 6. Get command: master side

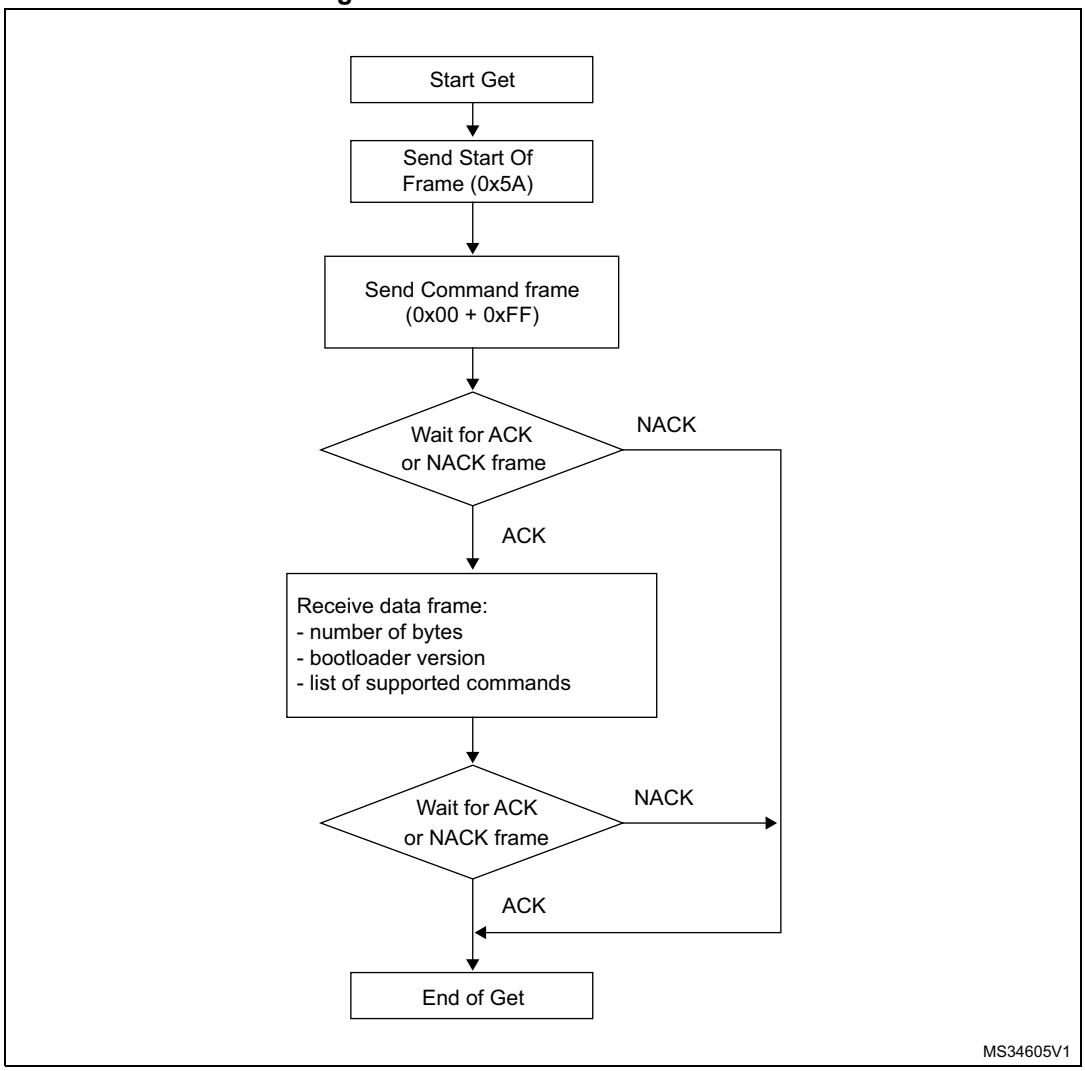
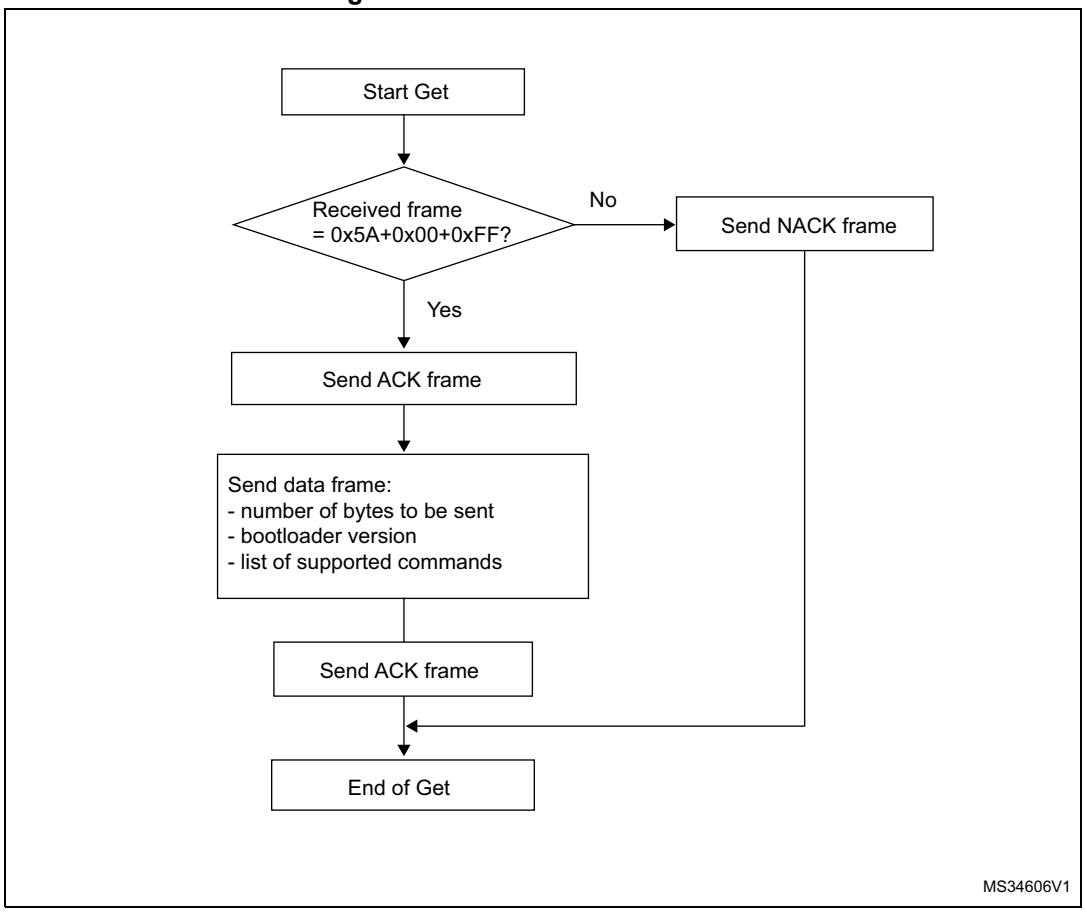


Figure 7. Get command: slave side



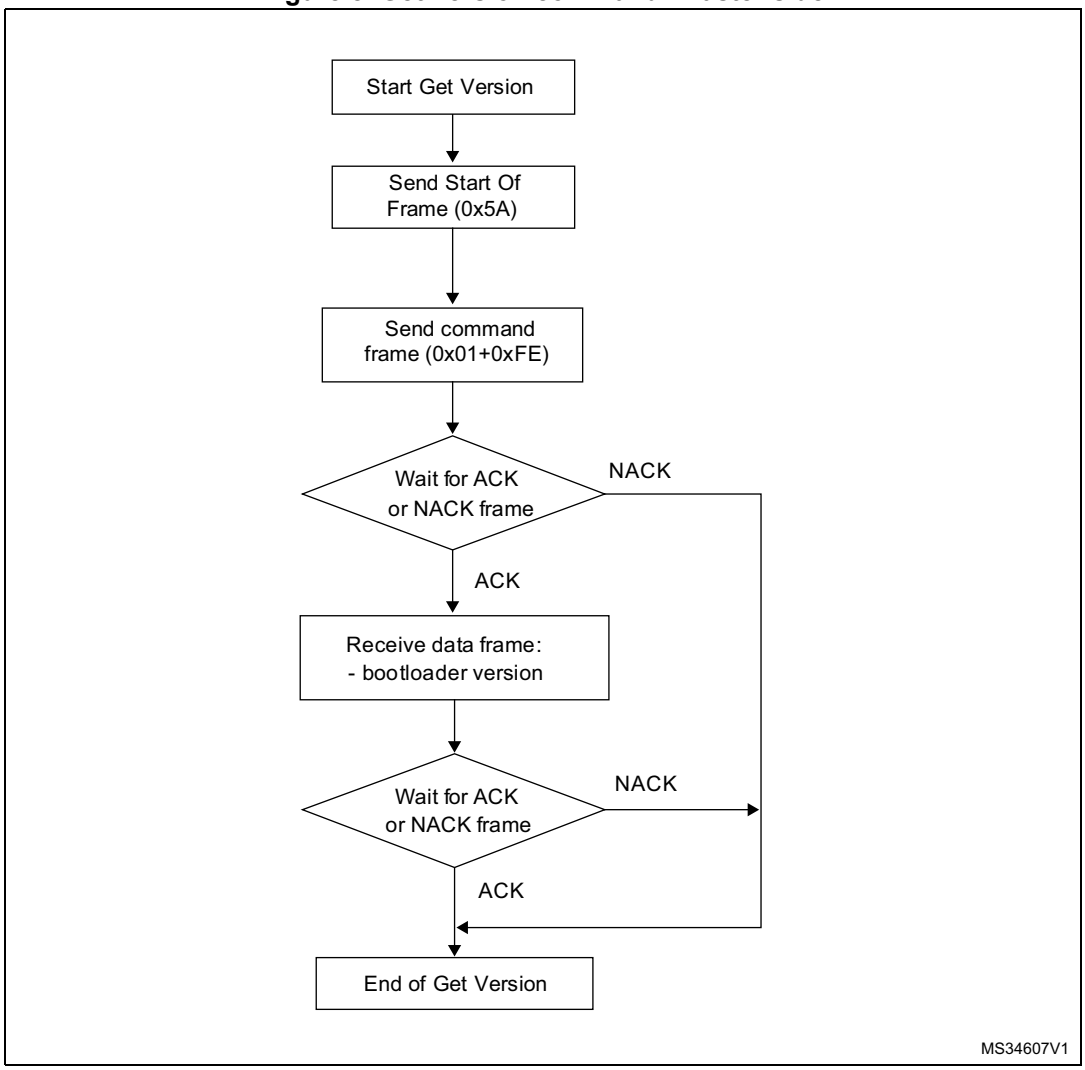
The STM32 sends the bytes as follows.

- Byte 1: ACK
- Byte 2: N = 11 = the number of bytes to follow – 1 except current and ACKs
- Byte 3: bootloader version (0 < version < 255), example: 0x10 = version 1.0.
- Byte 4: 0x00 (Get command)
- Byte 5: 0x01 (Get Version)
- Byte 6: 0x02 (Get ID)
- 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), available only for Version 1.3

2.3 Get Version command

The Get Version 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.

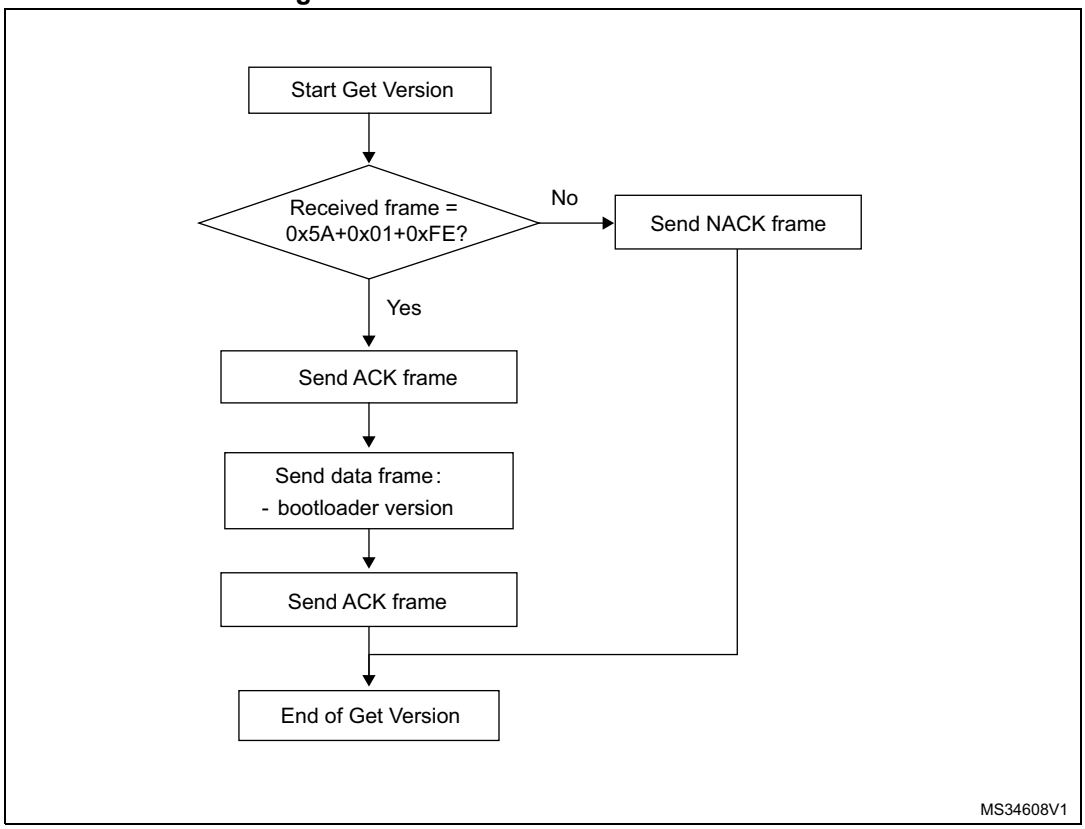
Figure 8. Get Version command: master side



The STM32 sends the bytes as follows:

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

Figure 9. Get Version command: slave side



2.4 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 master.

The STM32 slave sends the bytes as follows.

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

Figure 10. Get ID command: master side

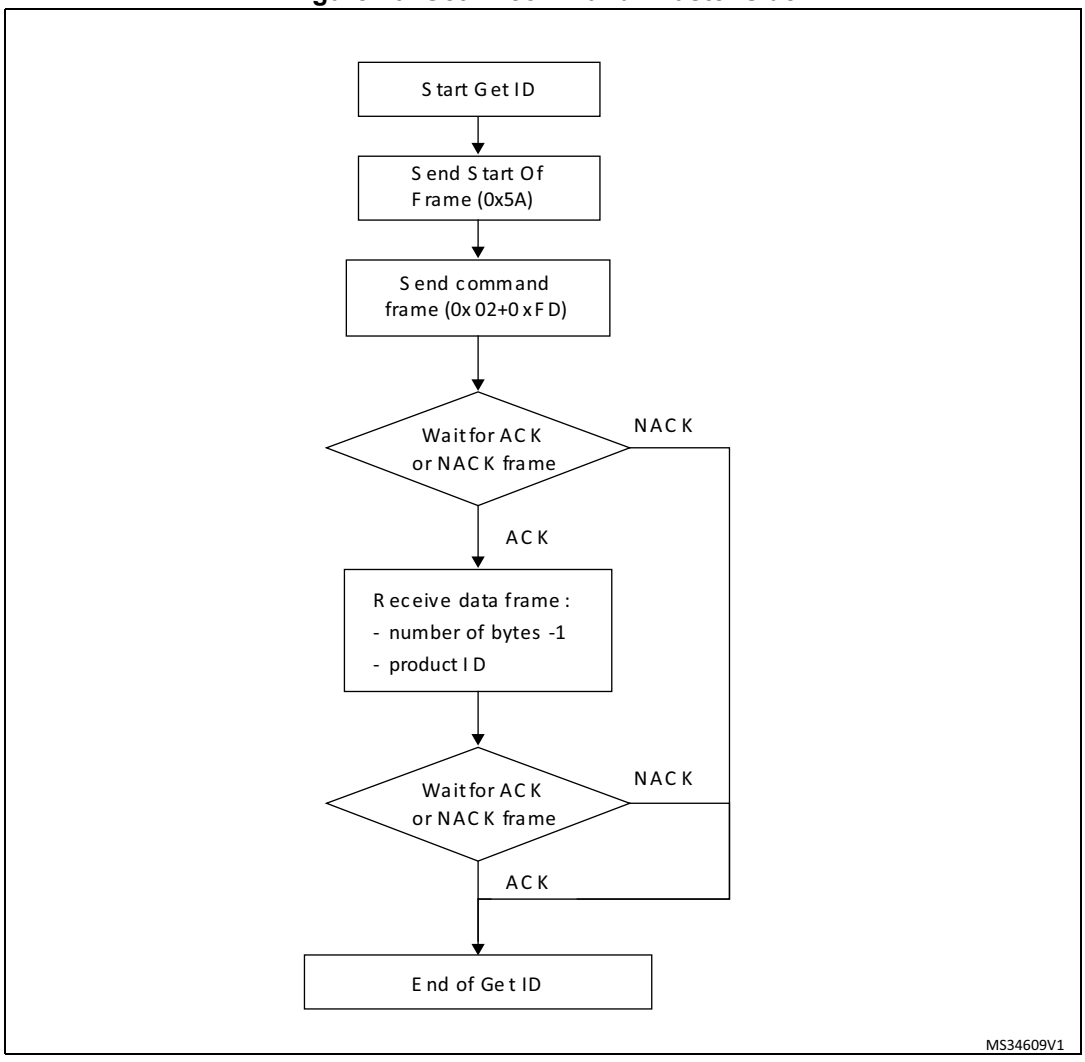
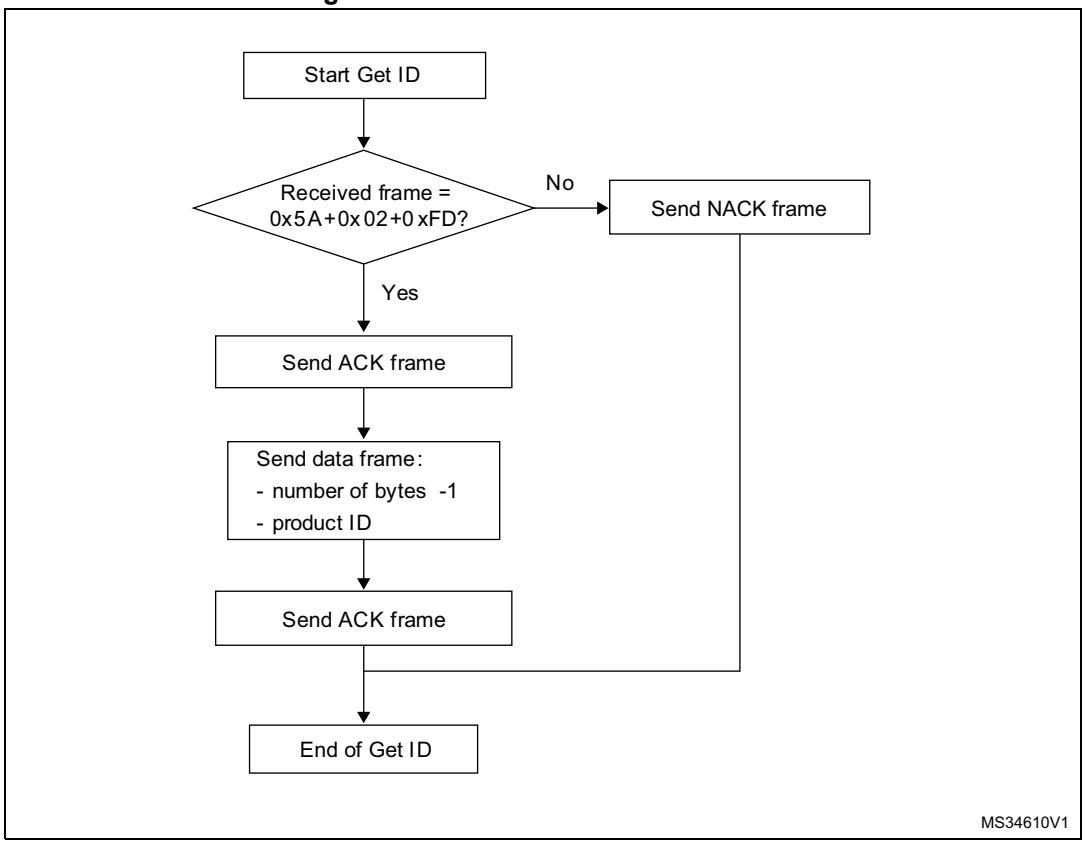


Figure 11. Get ID command: slave side



2.5 Read Memory command

The Read Memory 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 Read Memory 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 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 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 (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 a 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 to 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

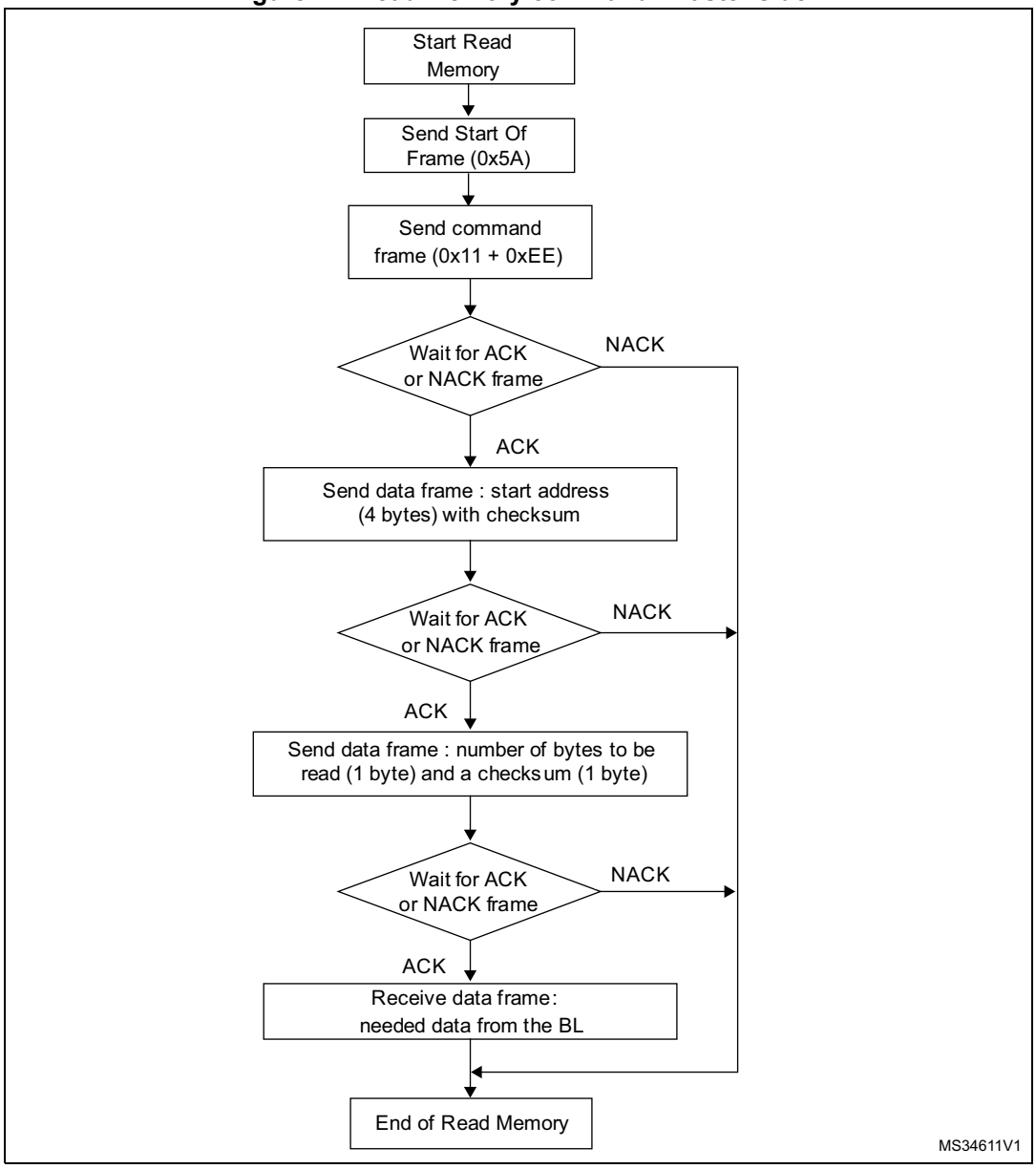
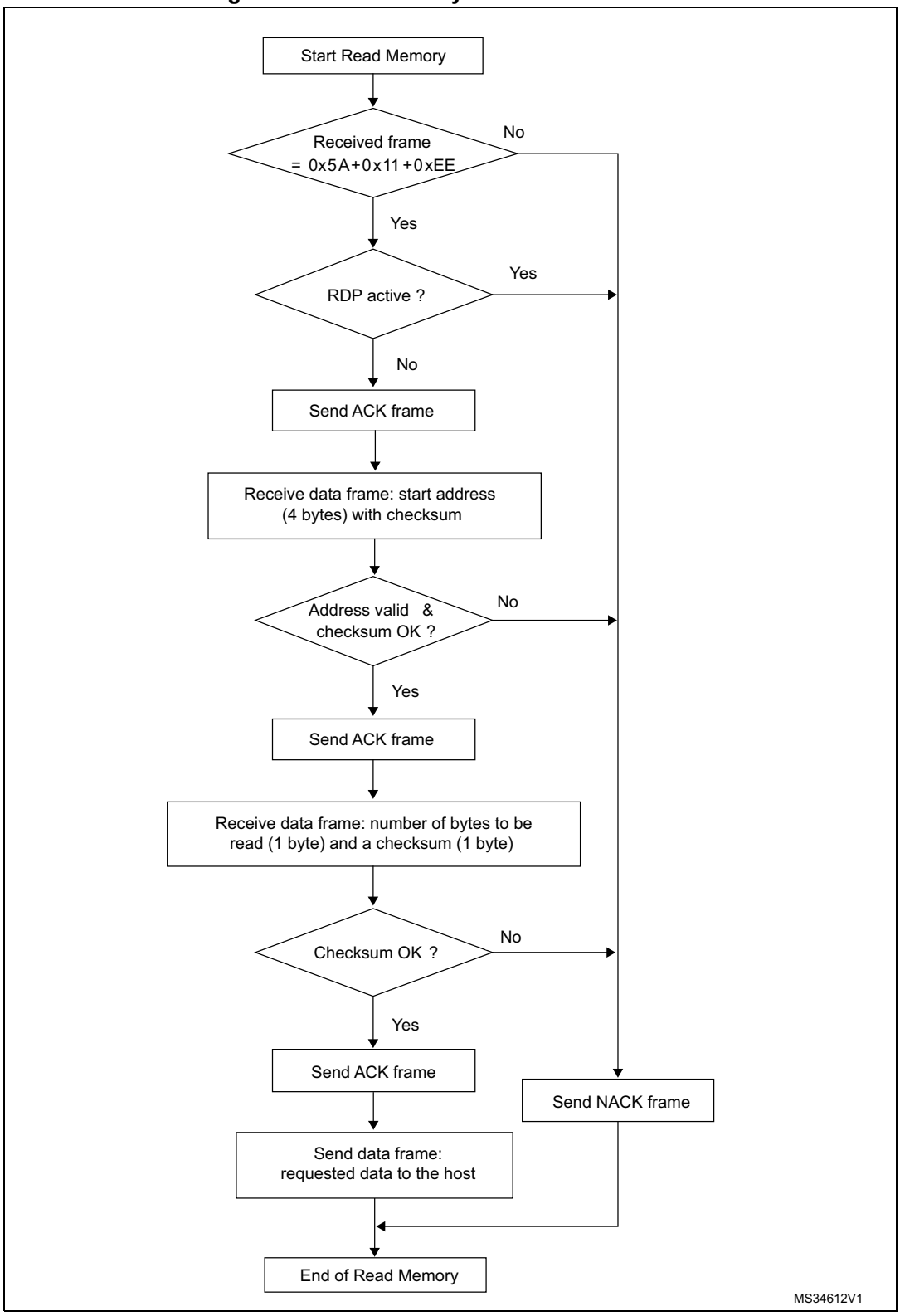


Figure 13. Read Memory command: slave side



2.6 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 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 a 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

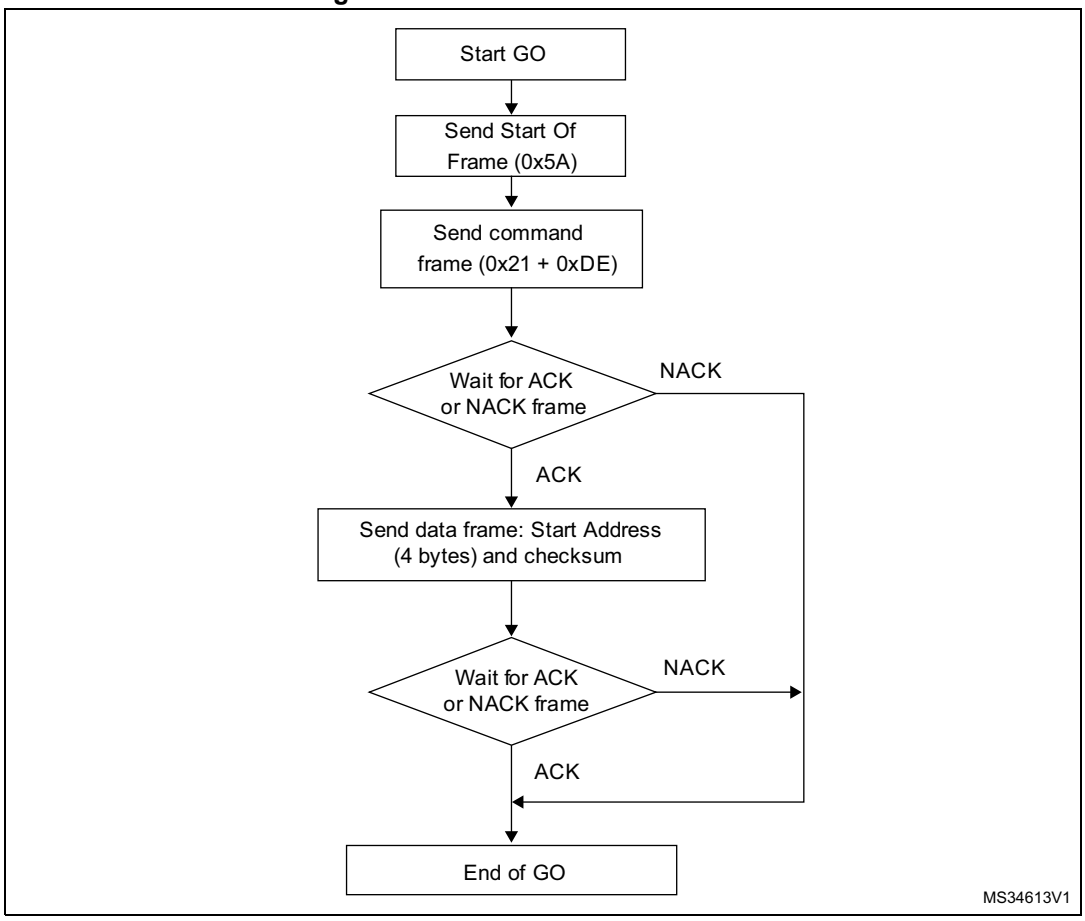
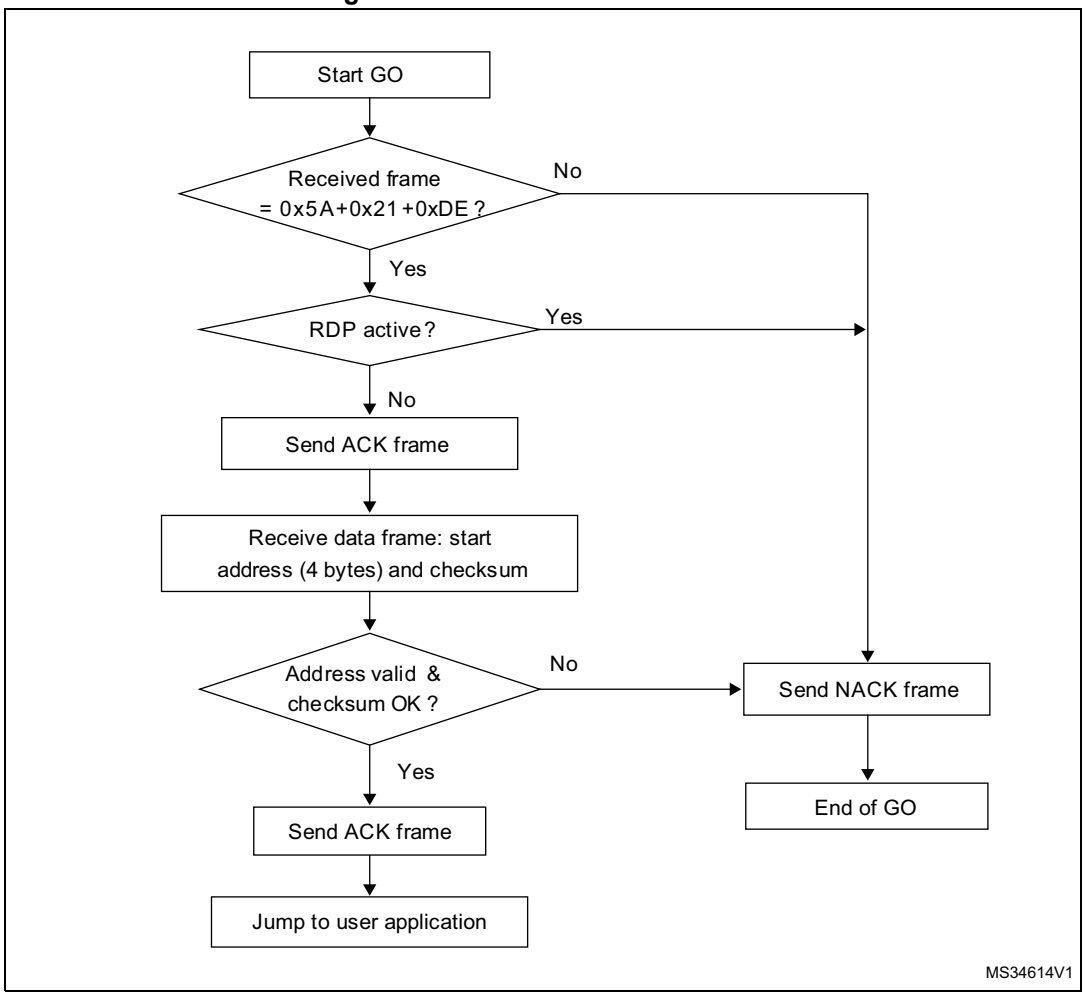


Figure 15. Go command: slave side



2.7 Write Memory command

The Write Memory 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 Write Memory 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 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 a 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 a NACK byte to the application and aborts the command.

If the Write Memory command is issued to the Option byte area, all options are erased before writing the new values, and at the end of the command the bootloader generates a system reset to take into account the new configuration of the option byte. The start address and the maximum length of the block to be written in the Option byte area has to respect the address and size of the product option bytes.

If the write destination is the Flash memory then the master has to 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.

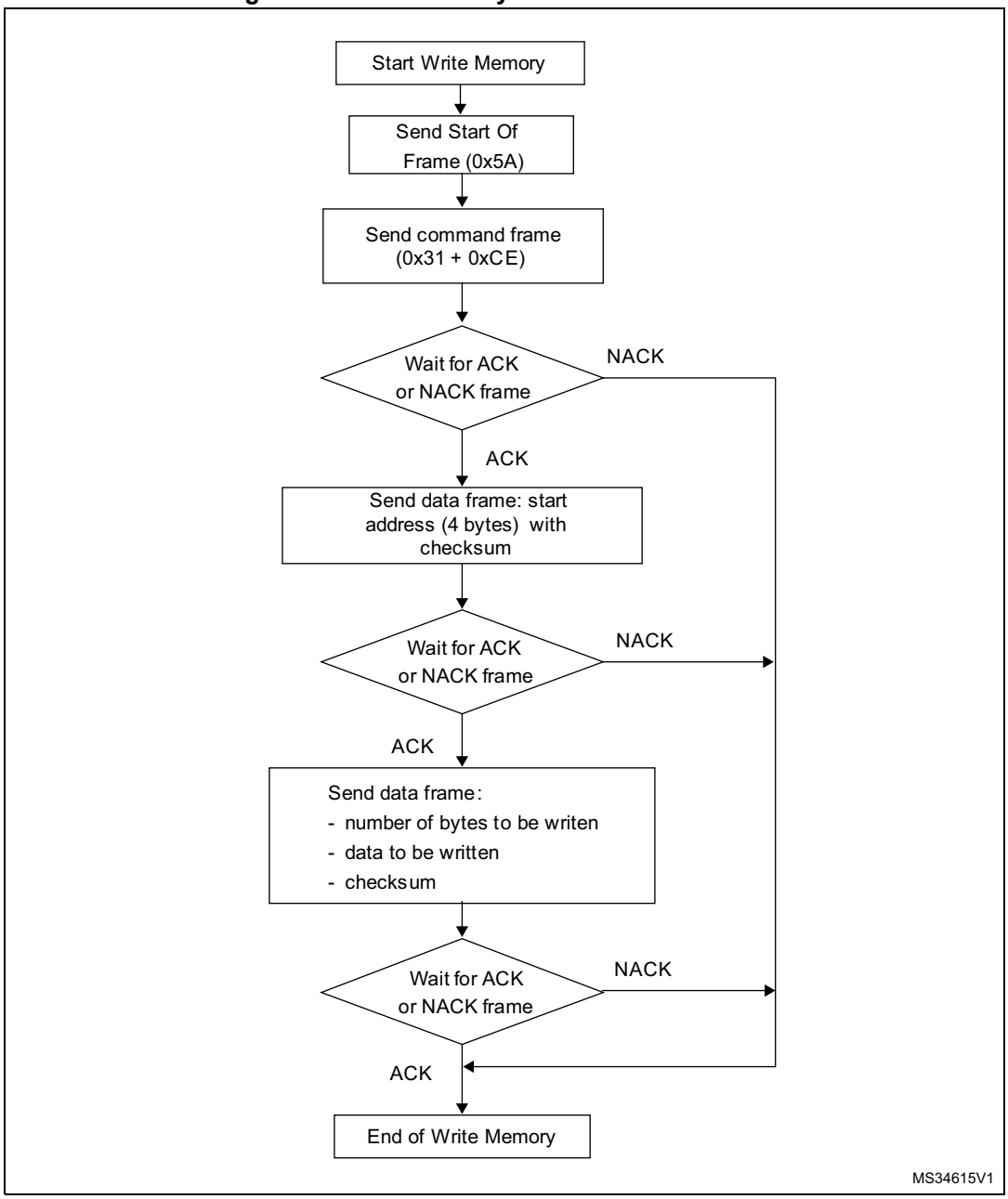
When writing to the RAM, user must not overlap the first RAM used by the bootloader firmware.

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

The master sends the bytes to the STM32 as follows.

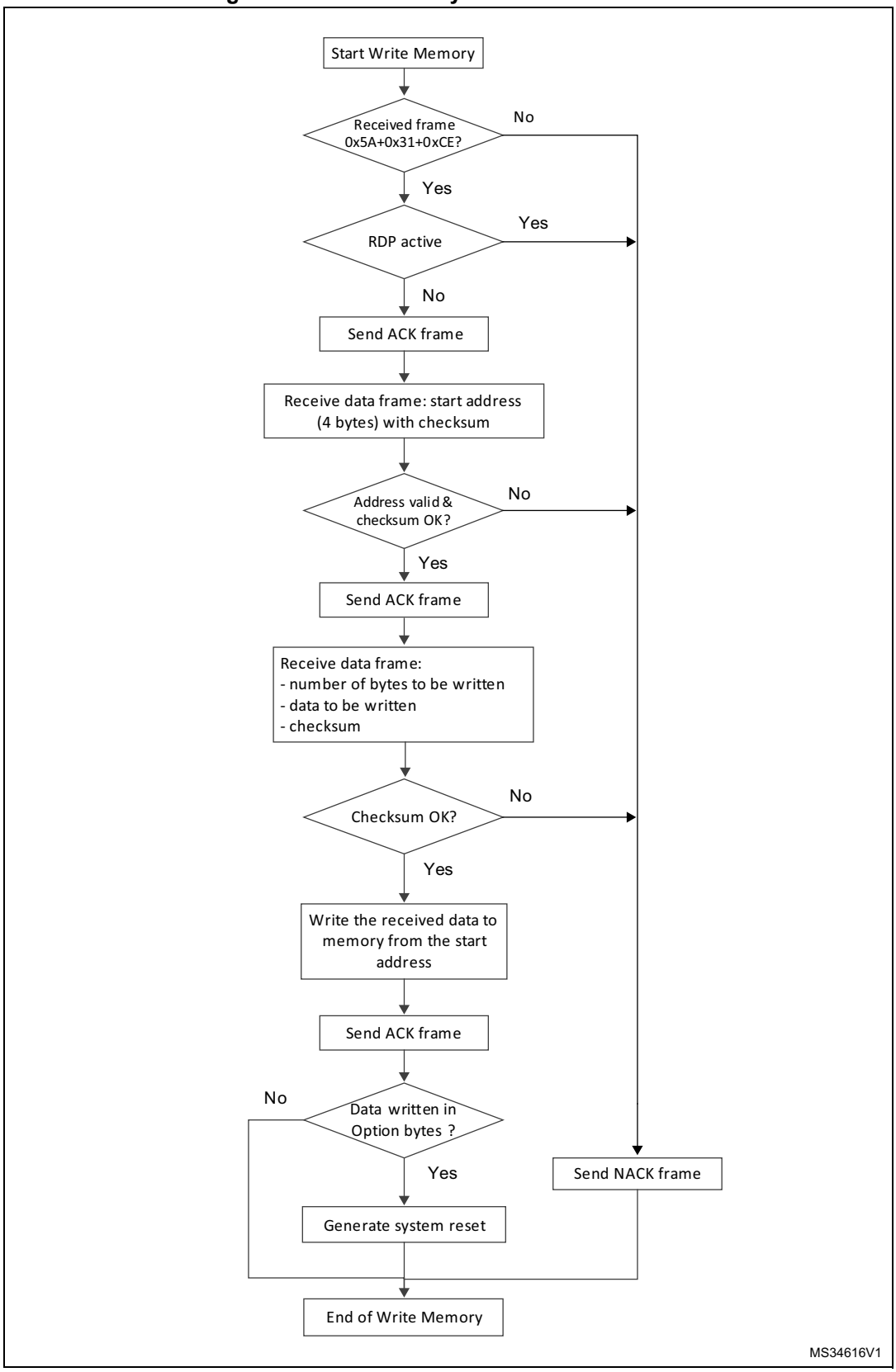
- Start of frame: 0x5A
- Byte 1: 0x31
- Byte 2: 0xCE
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Byte 3 to byte 6: start address
  - Byte 3: MSB
  - Byte 6: LSB
- Byte 7: checksum: XOR (byte3, byte4, byte5, byte6)
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Byte 8: number of bytes to be received (0 < N ≤ 255)
- N +1 data bytes: (max 256 bytes)
- Checksum byte: XOR (N, N+1 data bytes)

Figure 16. Write Memory command: master side



Note: In some operating conditions, the master has to wait for a delay of 1 ms after receiving the Acknowledge and before sending the data frame (number of bytes to be written, data to be written and checksum).

Figure 17. Write Memory command: slave side



2.8 Erase Memory command

The Erase Memory command allows the master to erase the Flash memory pages or sectors using two-byte addressing mode. When the bootloader receives the Erase Memory 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 a 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 be erased.

- 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 other values where 0 ≤ N < maximum number of pages or sectors: N + 1 pages or sectors are erased.

The bootloader then receives the following.

- In the case of a special erase, one byte: checksum of the previous bytes: (that is, 0x00 for 0xFFFF).
- In the case of N+1 pages or sector 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 or sectors is product-related, and must be respected.

The master sends bytes to the STM32 as follows.

- Start of frame: 0x5A
- Byte 1: 0x44
- Byte 2: 0xBB
- Wait for ACK (as described in *Section 1: SPI bootloader code sequence*)
- Bytes 3-4:
  - Special erase: global erase (0xFFFY where Y={F, E, D})

OR

  - Number of pages or sectors to be erased (N+1 where: 0 ≤ N < maximum number of pages or sectors).
- Remaining bytes:
  - Checksum of bytes 3-4 in the case of a special erase (0x00, 0x01 or 0x02).

OR

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

Figure 18. Erase Memory command: master side

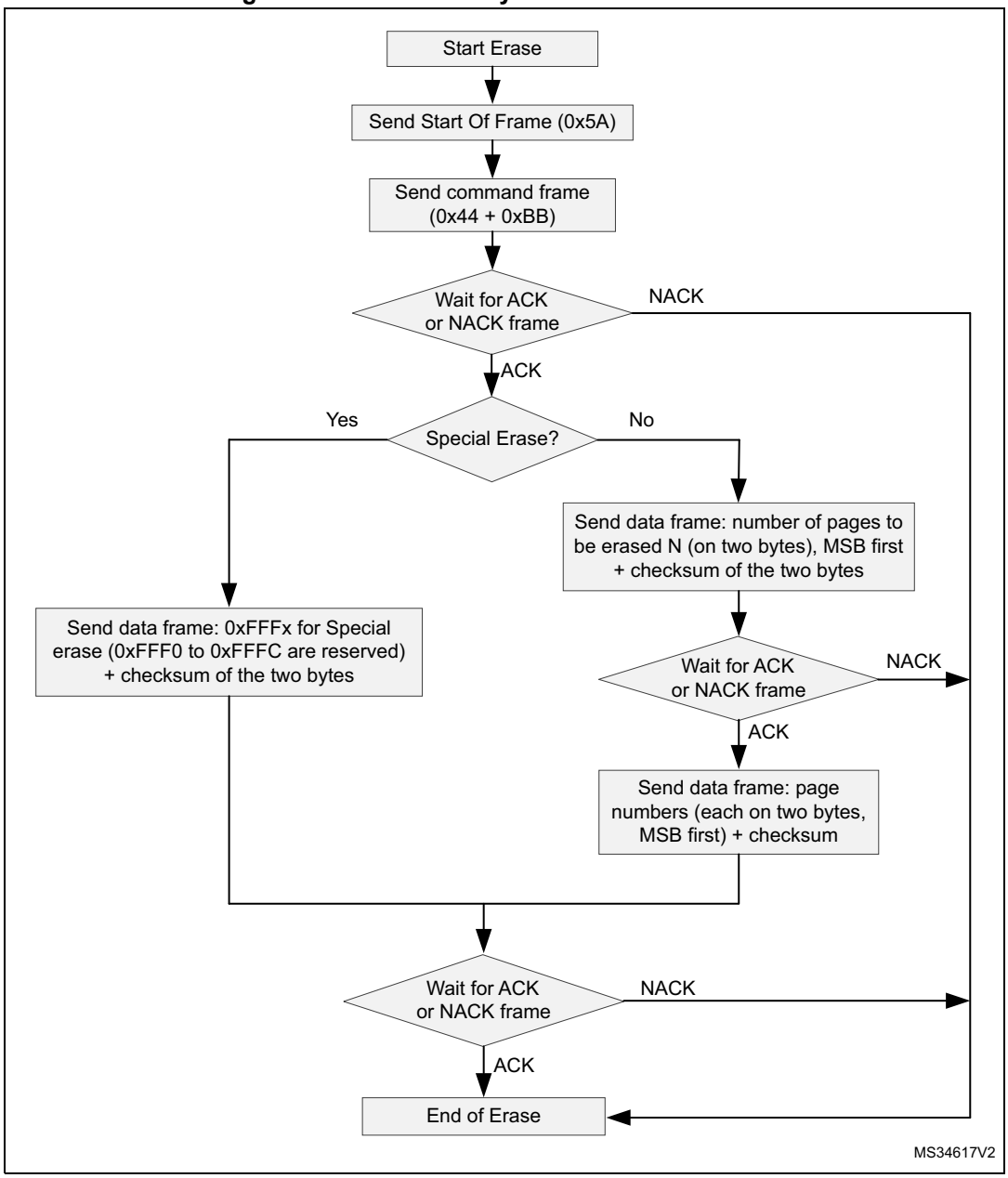
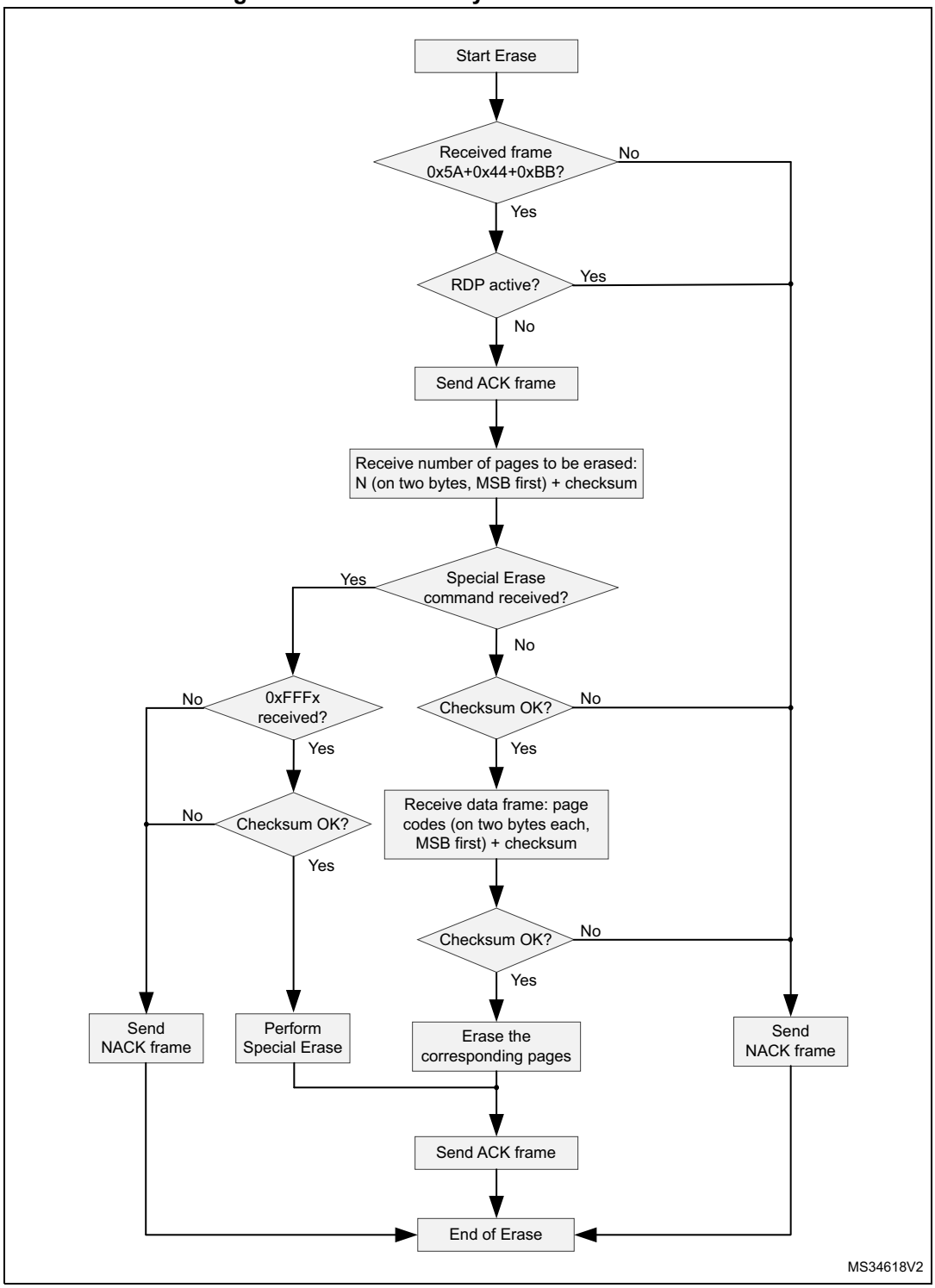


Figure 19. Erase Memory command: slave side



2.9 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 master. After 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.

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

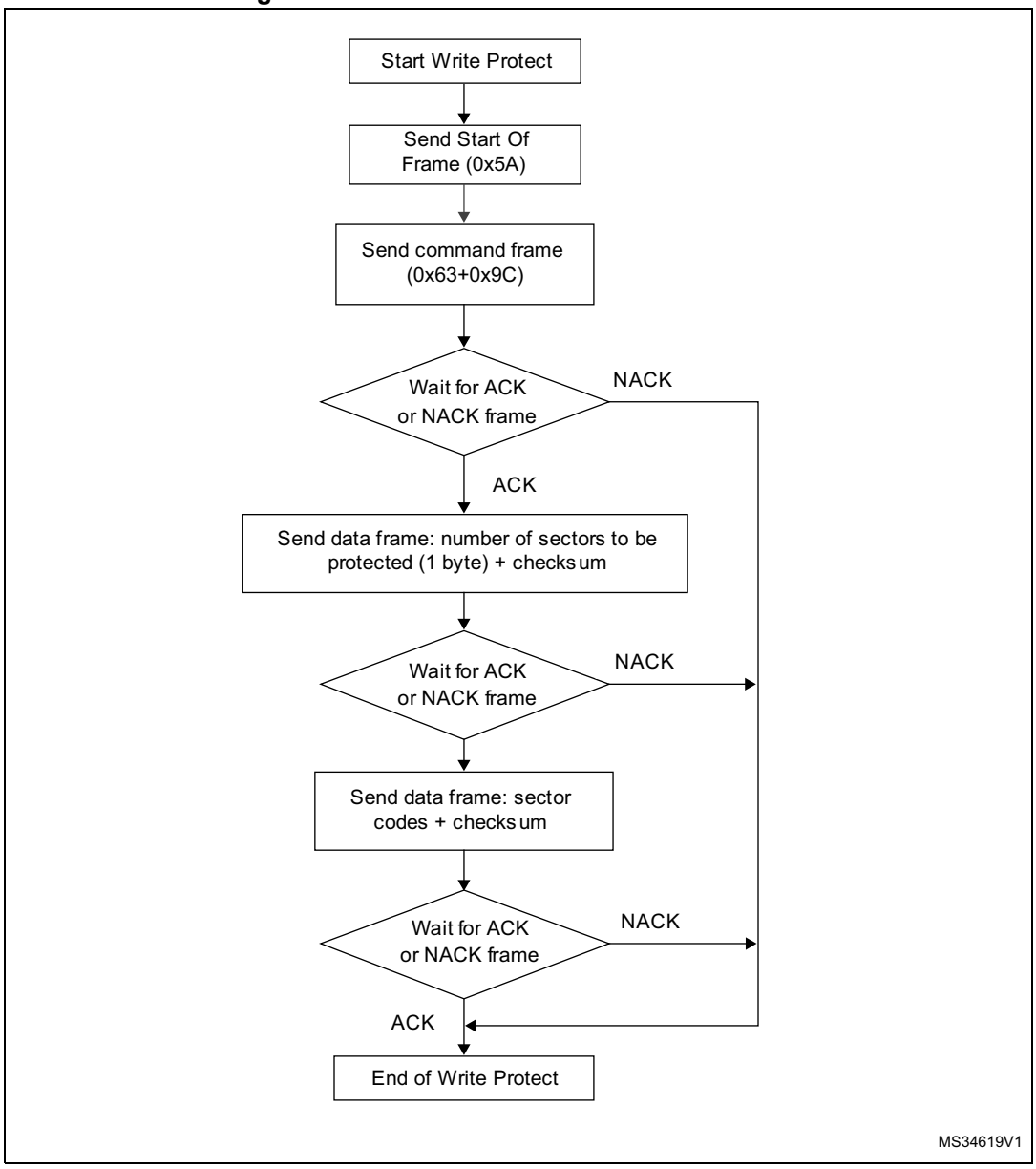
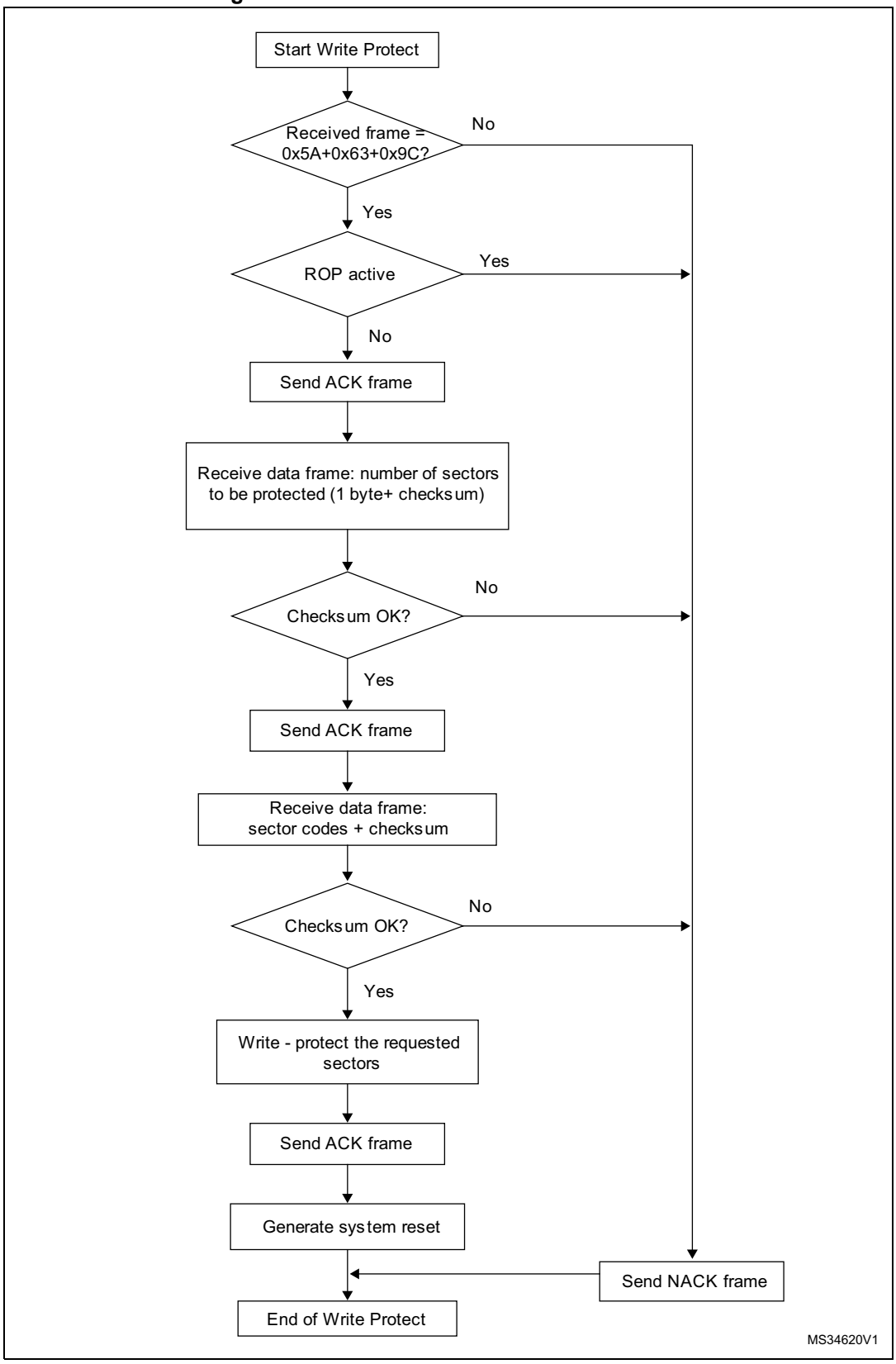


Figure 21. Write Protect command: slave side



2.10 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 master. After 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 22. Write Unprotect command: master side

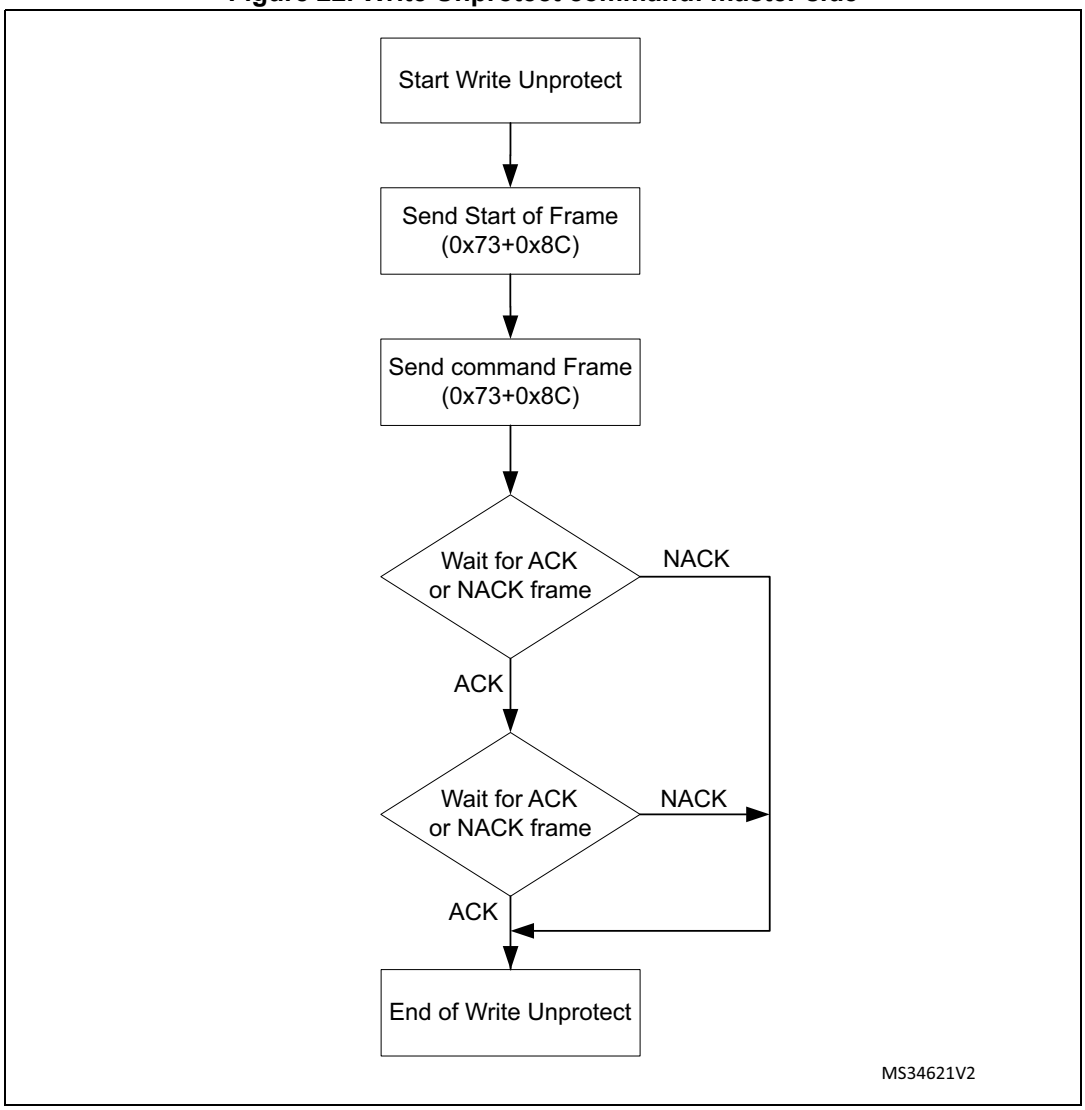
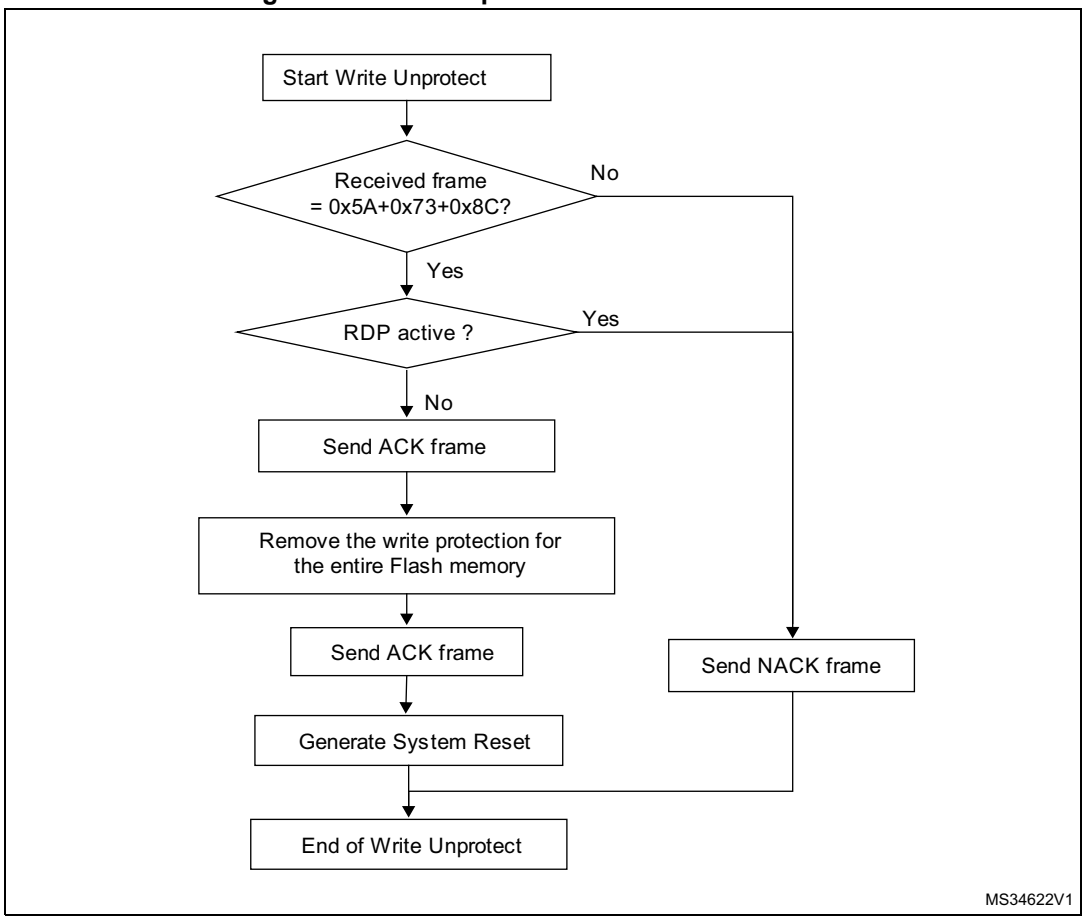


Figure 23. Write Unprotect command: slave side



2.11 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 master. After 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 24. Readout Protect command: master side

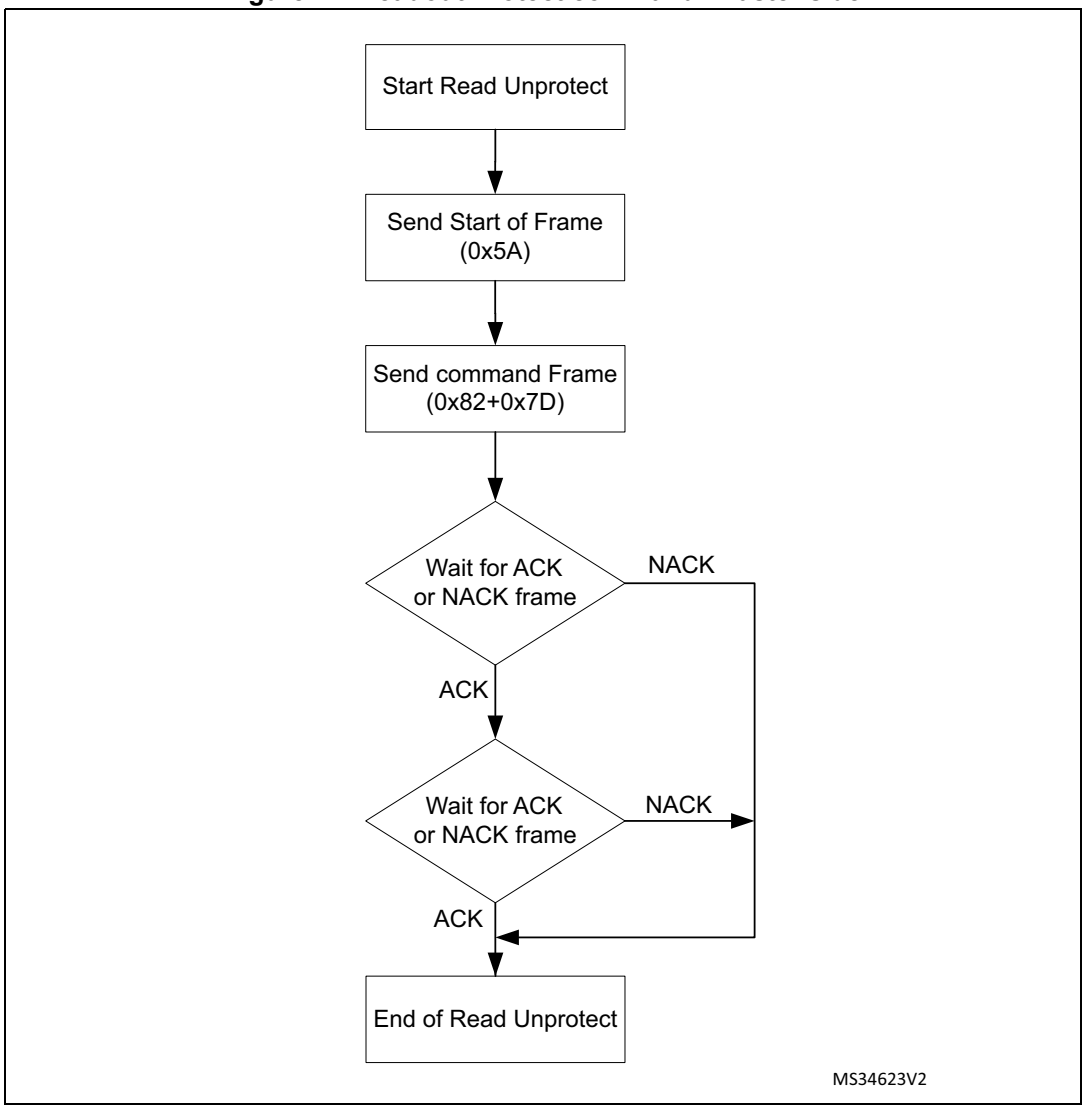
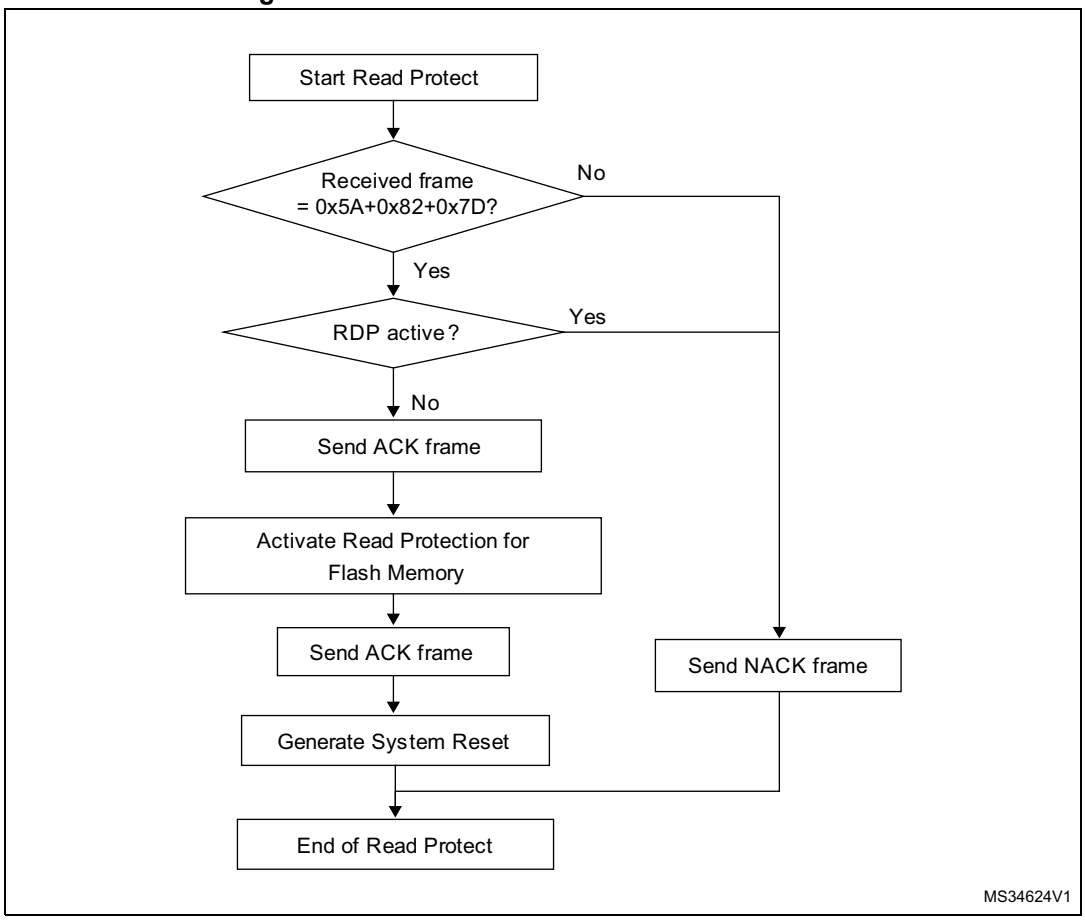


Figure 25. Readout Protect command: slave side



2.12 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 master. After transmission of the ACK byte, the bootloader erases all the Flash memory sectors and it 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.

The master has to wait enough time for the read protection disable (which is equivalent to the Mass Erase time on most products - refer to 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

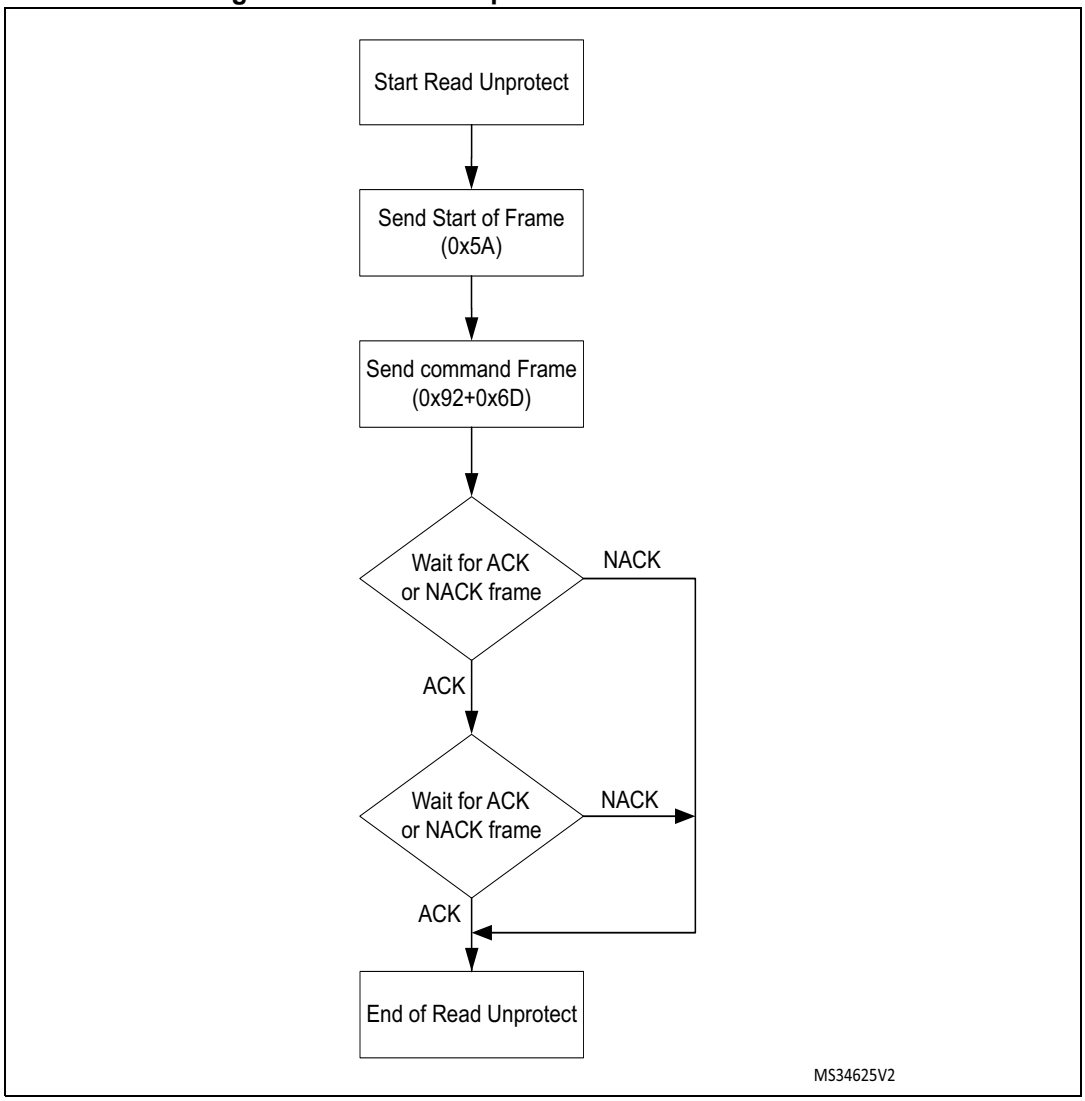
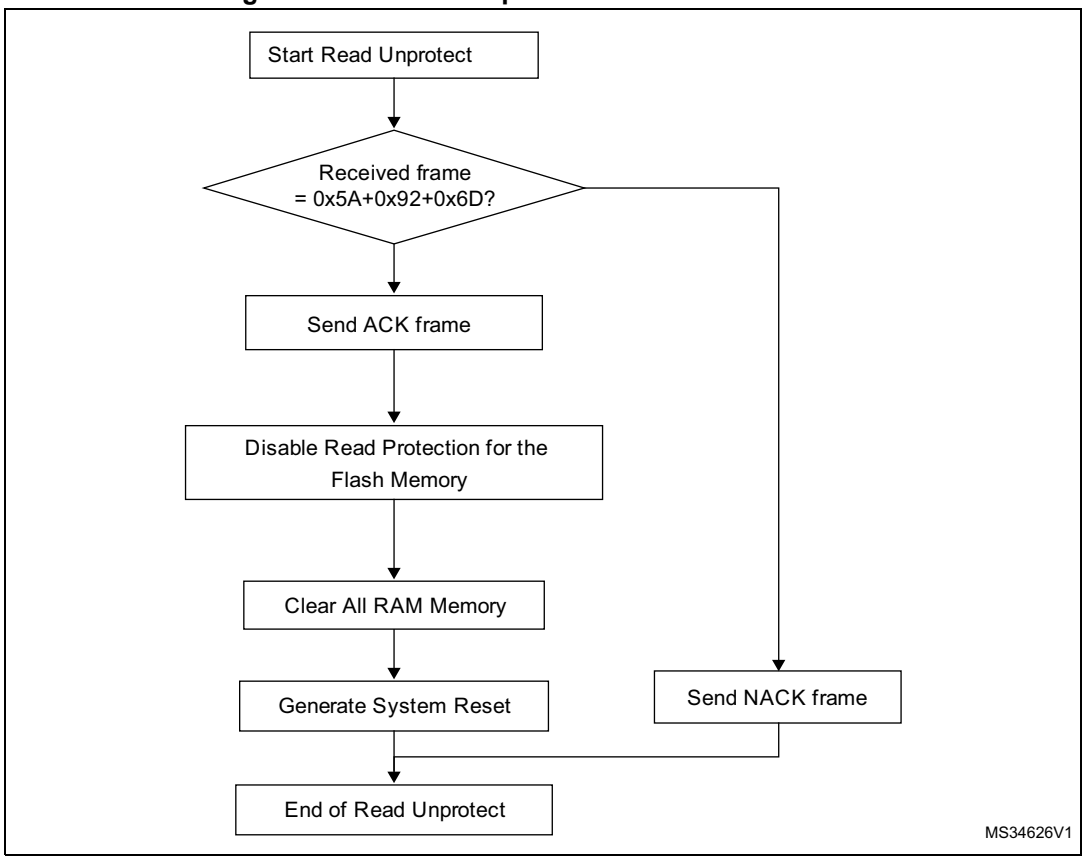


Figure 27. Readout Unprotect command: slave side



2.13 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 from 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.

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 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)

Bytes 8 to 11 Start address: 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: Memory area size (number of 32-bits words): byte 13: MSB, byte 16: LSB

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

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 polynomial: byte 18: MSB, byte 21: LSB

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

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

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

Bytes 23 to 26: CRC initialization value: byte 26: MSB, byte 26: LSB

Byte 27: Checksum: XOR (byte 23, byte 24, byte 25, byte 26)

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

Figure 28. Get Checksum command: host side

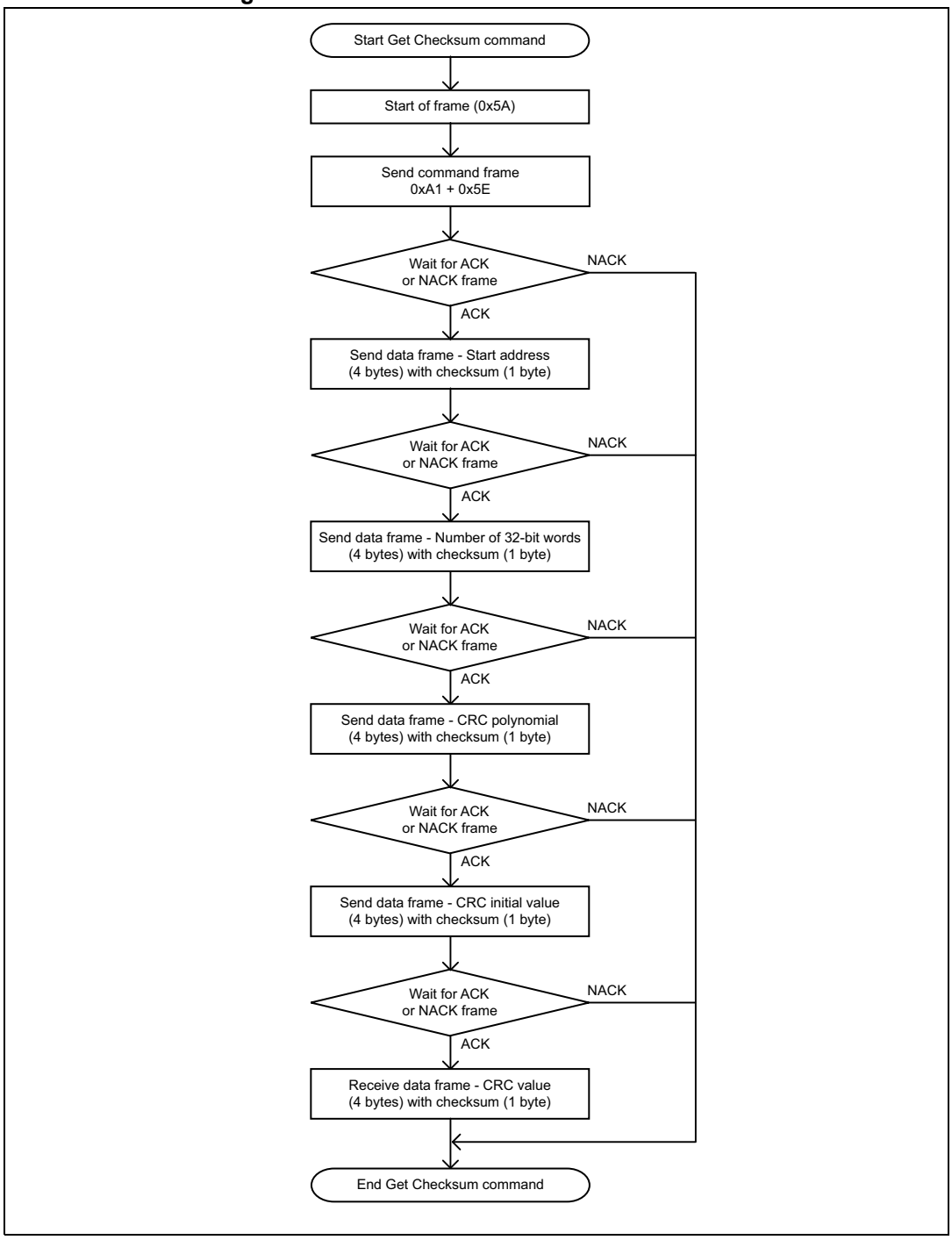
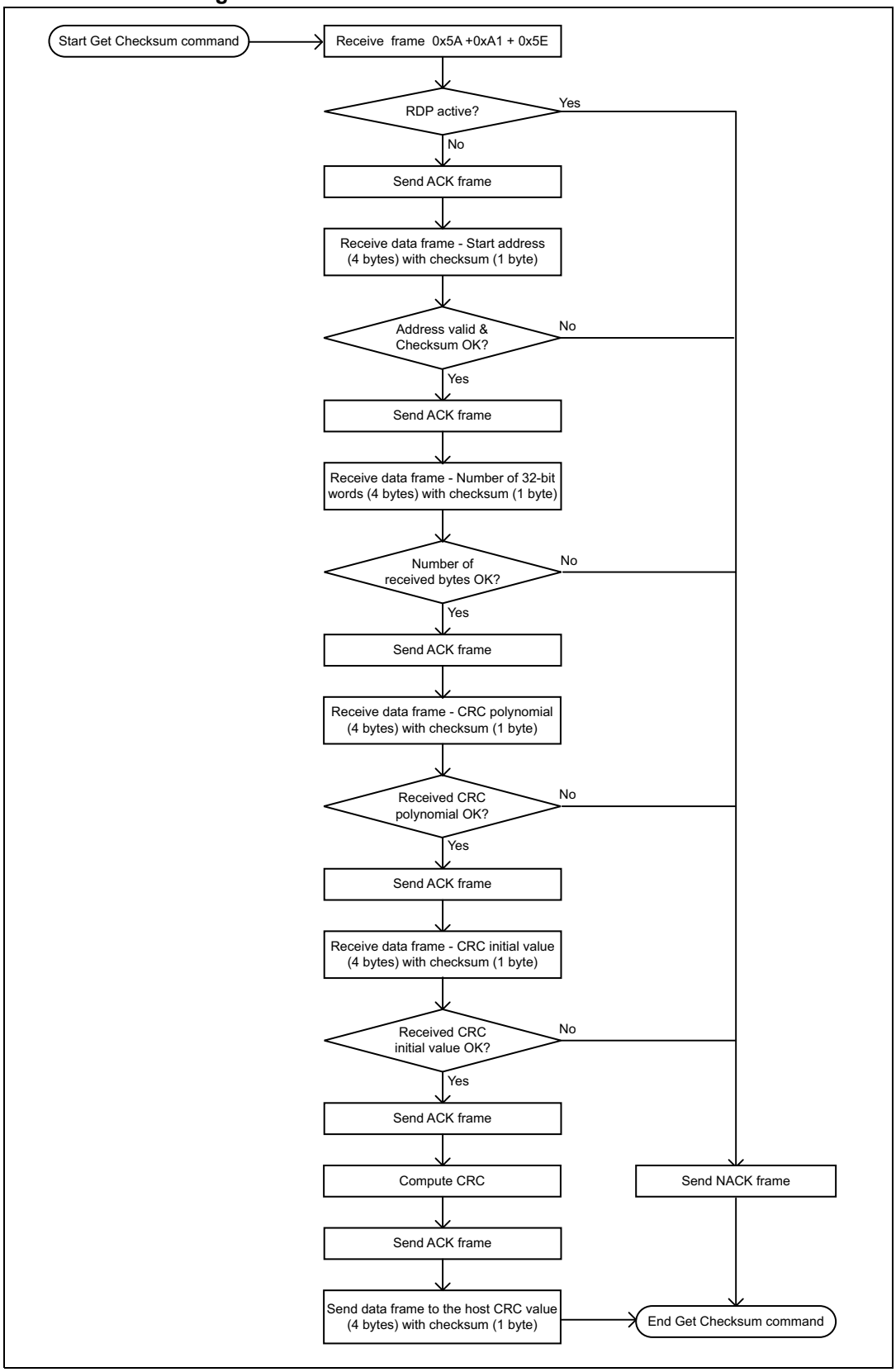


Figure 29. Get Checksum command: device side



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.

4 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.