AN4286
Application note

SPI protocol used in the STM32 bootloader

Introduction

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

For more information about the SPI hardware resources and requirements for your slave bootloader, refer to application note “STM32 microcontroller system memory boot mode” (AN2606).

This document applies to the products listed in *Table 1*.

Table 1. Applicable products

Product family	Product series
Microcontrollers	STM32L0 series: – STM32L051xx, STM32L052xx, STM32L053xx – STM32L062xx, STM32L063xx STM32F4 series: – STM32F401xx, STM32F411xx – STM32F405xx, STM32F407xx – STM32F415xx, STM32F417xx – STM32F429xx, STM32F439xx

Contents

1 SPI Bootloader code sequence 5

2 Bootloader command set 8

2.1 Safety of communication 9
2.2 Get command 9
2.3 Get Version command 12
2.4 *Get ID* command 13
2.5 *Read Memory* command 15
2.6 Go command 18
2.7 *Write Memory* command 20
2.8 *Erase Memory* command 24
2.9 *Write Protect* command 27
2.10 *Write Unprotect* command 30
2.11 *Readout Protect* command 31
2.12 *Readout Unprotect* command 33

3 Evolution of the bootloader protocol versions 36

4 Revision history 37

List of tables

Table 1. Applicable products 1
Table 2. SPI bootloader commands 8
Table 3. Bootloader protocol versions 36
Table 4. Document revision history 37

List of figures

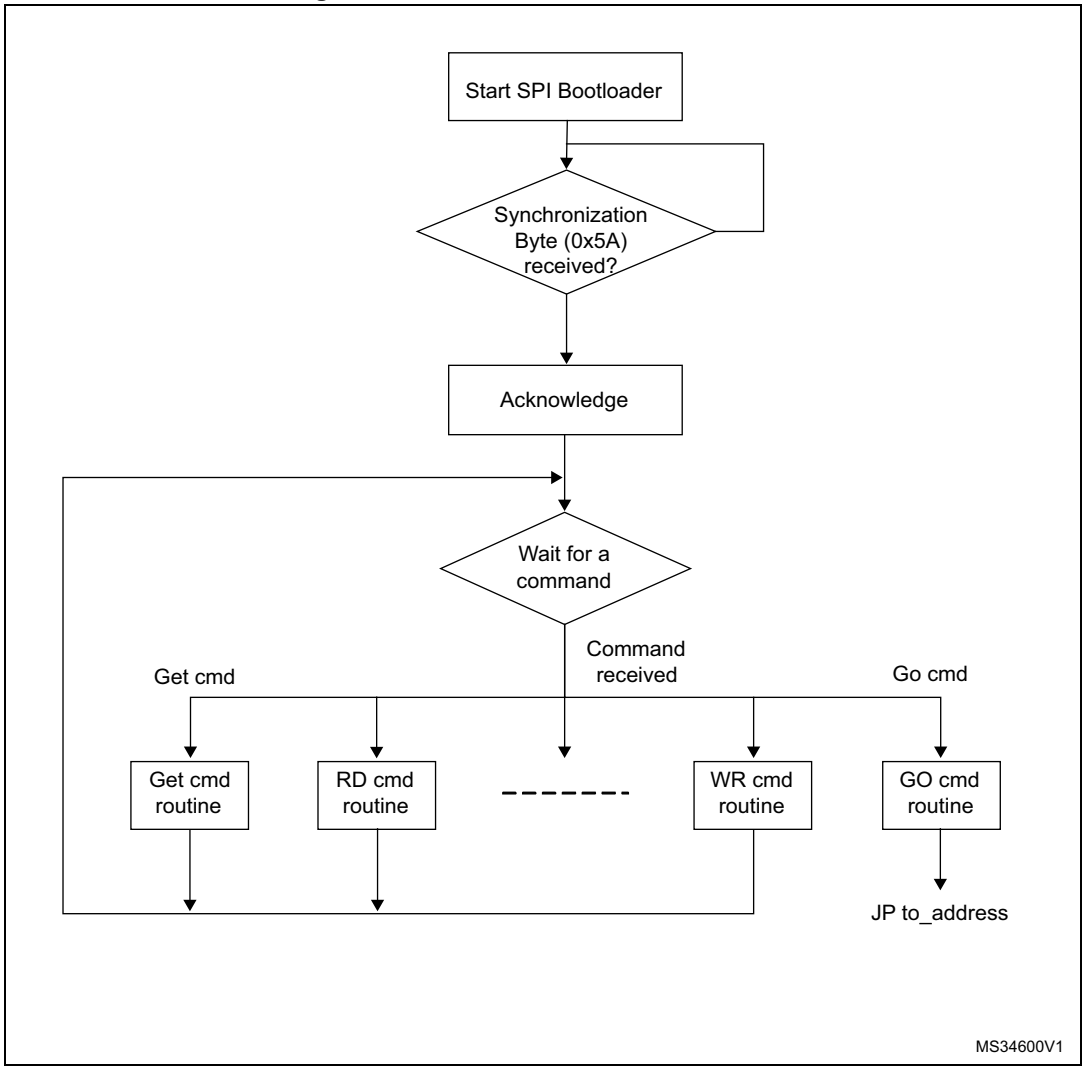
Figure 1. Bootloader for STM32 with SPI . . . 5
Figure 2. Get ACK procedure (Master side) . . . 6
Figure 3. Bootloader SPI synchronization frame . . . 6
Figure 4. SPI command frame . . . 7
Figure 5. Read data frame . . . 7
Figure 6. Get command: master side . . . 10
Figure 7. Get command: slave side . . . 11
Figure 8. Get Version: master side . . . 12
Figure 9. Get Version: slave side . . . 13
Figure 10. Get ID command: master side . . . 14
Figure 11. Get ID command: slave side . . . 15
Figure 12. Read Memory command: master side . . . 16
Figure 13. Read Memory command: slave side . . . 17
Figure 14. Go command: master side . . . 19
Figure 15. Go command: slave side . . . 20
Figure 16. Write Memory command: master side . . . 22
Figure 17. Write Memory command: slave side . . . 23
Figure 18. Erase Memory command: master side . . . 25
Figure 19. Erase Memory command: slave side . . . 26
Figure 20. Write Protect command: master side . . . 28
Figure 21. Write Protect command: slave side . . . 29
Figure 22. Write Unprotect command: master side . . . 30
Figure 23. Write Unprotect command: slave side . . . 31
Figure 24. Readout Protect command: master side . . . 32
Figure 25. Readout Protect command: slave side . . . 33
Figure 26. Readout Unprotect command: master side . . . 34
Figure 27. Readout Unprotect command: slave side . . . 35

1 SPI Bootloader code sequence

The STM32 bootloader is an SPI slave.

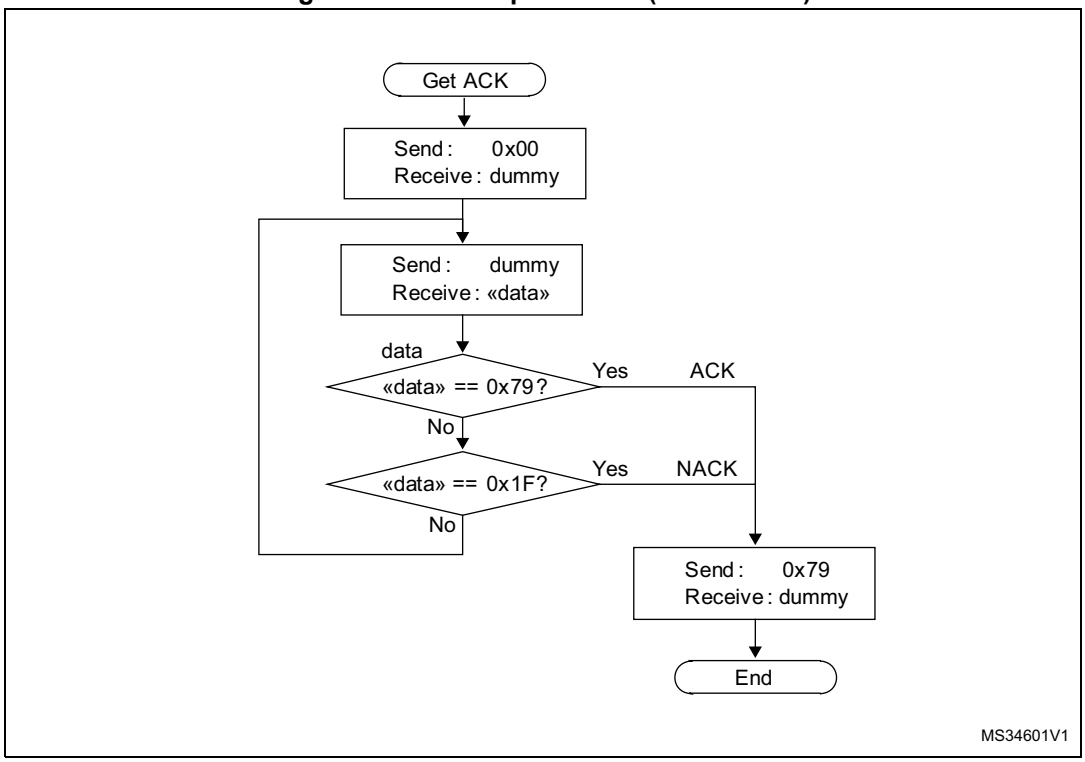
For all SPI bootloader operations, the NSS pin (chip select) must be tied low. If the NSS pin is tied high, the communication on the SPI bus will be ignored by the STM32 slave.

Figure 1. Bootloader for STM32 with SPI



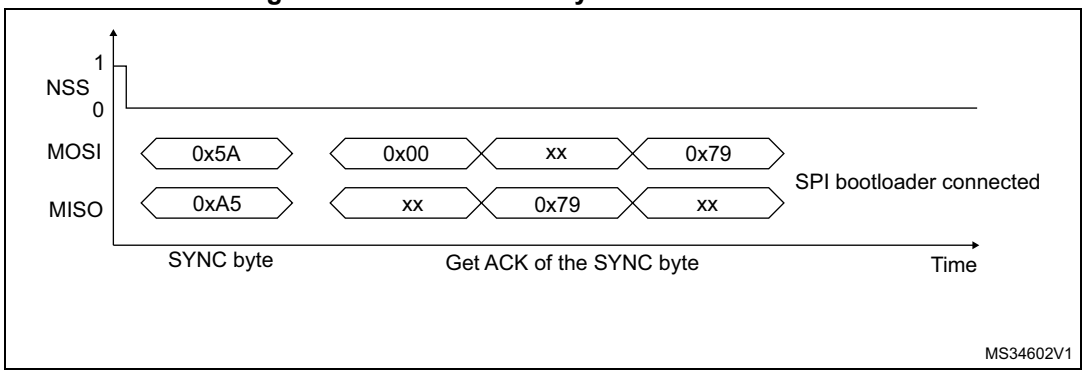
Once the system memory boot mode is entered and the STM32 microcontroller has been configured (for more details, refer to your STM32 system memory boot mode application note) the bootloader code begins to scan the SPI_MOSI line pin, waiting to detect a synchronization byte on the bus (0x5A). Once a detection 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)



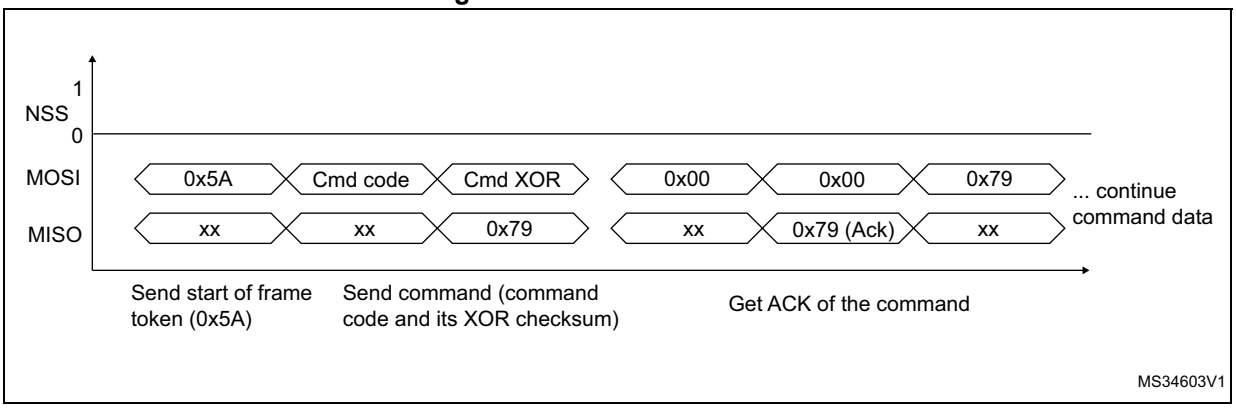
To start communication with the bootloader, the Master must first send a synchronization byte (0x5A) and wait to receive an acknowledge (ACK).

Figure 3. Bootloader SPI synchronization frame



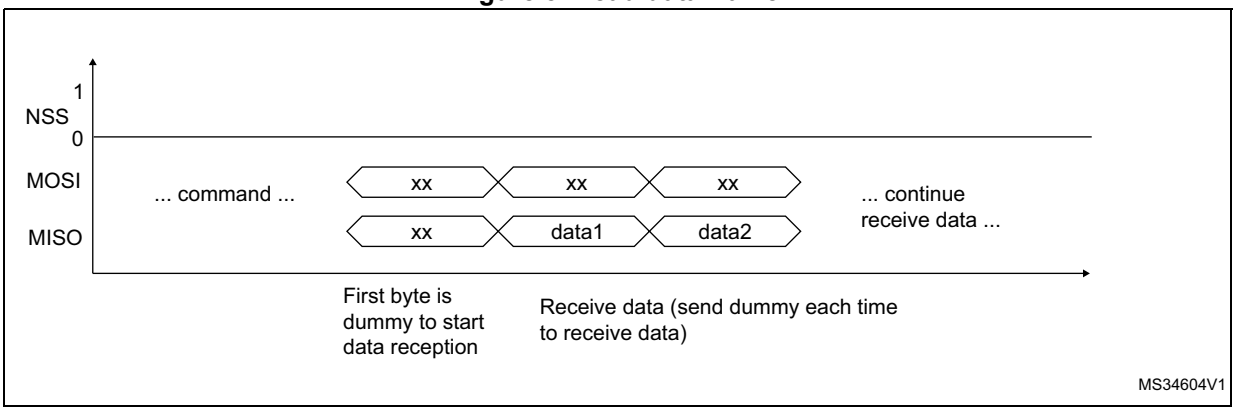
Where xx represents a dummy byte.

Figure 4. SPI command frame



To read any data, the Master should send a dummy byte before starting to read data sent by the slave. 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)	Command code	Command 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.

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 NACKed and have no effect on the slave. Once the RDP has been removed, the other commands become active.
3. Please refer to STM32 product datasheet and AN2606: STM32 microcontroller system memory boot mode to know which memory spaces are valid for this command.

Since 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's answer is not immediate, the received data is ignored (dummy) while the Master is transmitting (this data is 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 should always send 0x00 (this data is 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 XORed. A byte containing the computed XOR of all previous bytes is added to the end of each communication (checksum byte). By XORing all received bytes, data + 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 (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

Note: The master's 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 + 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 you 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

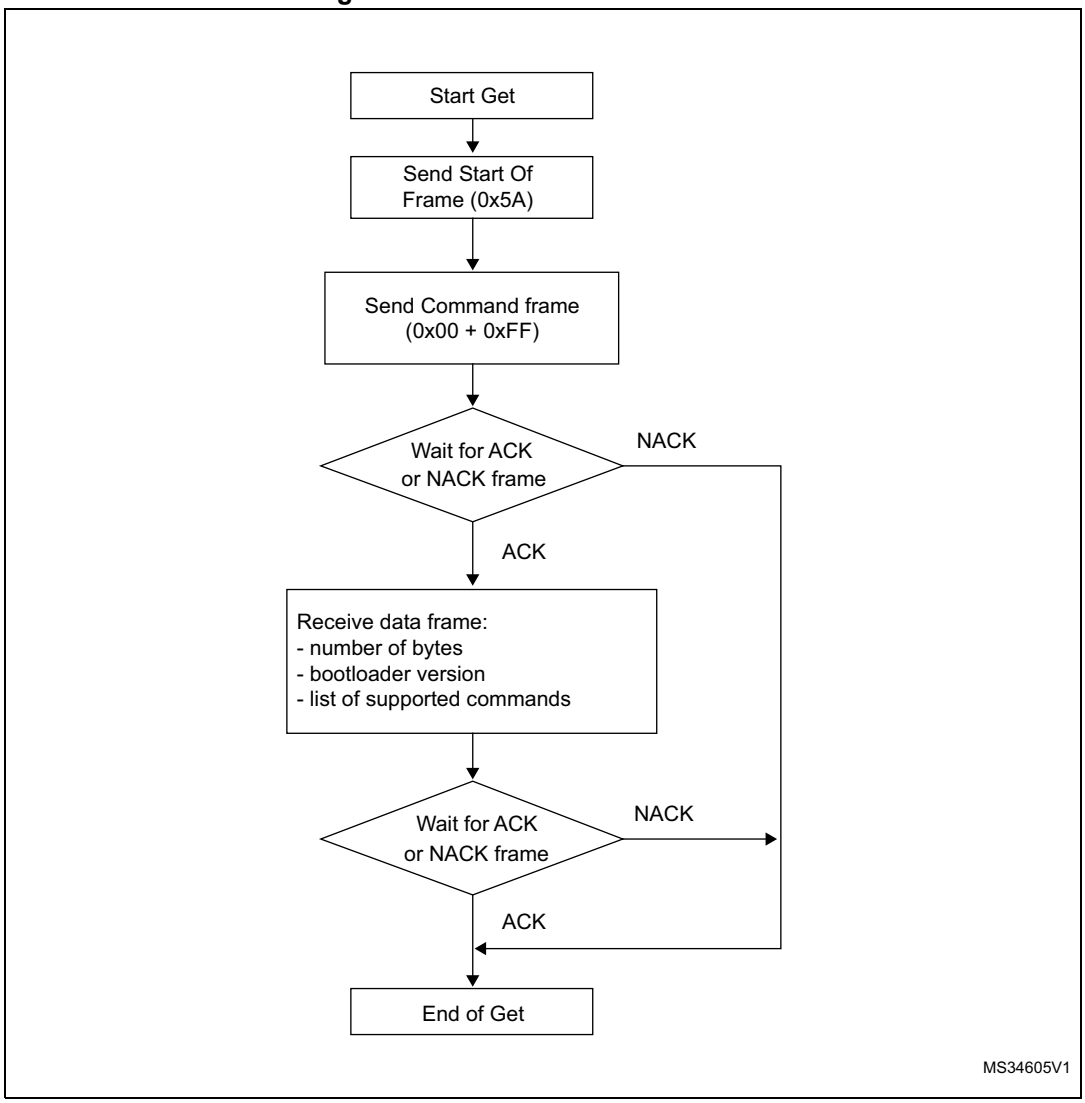
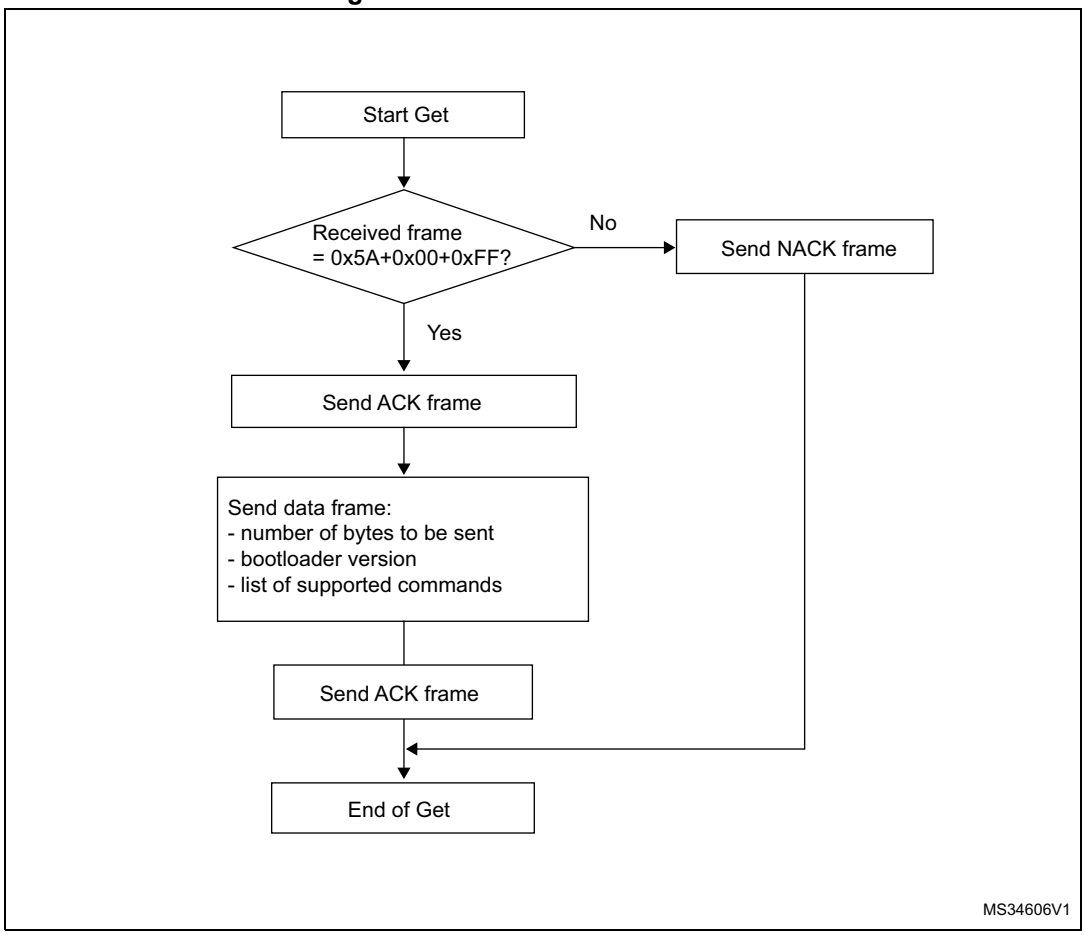


Figure 7. Get command: slave side



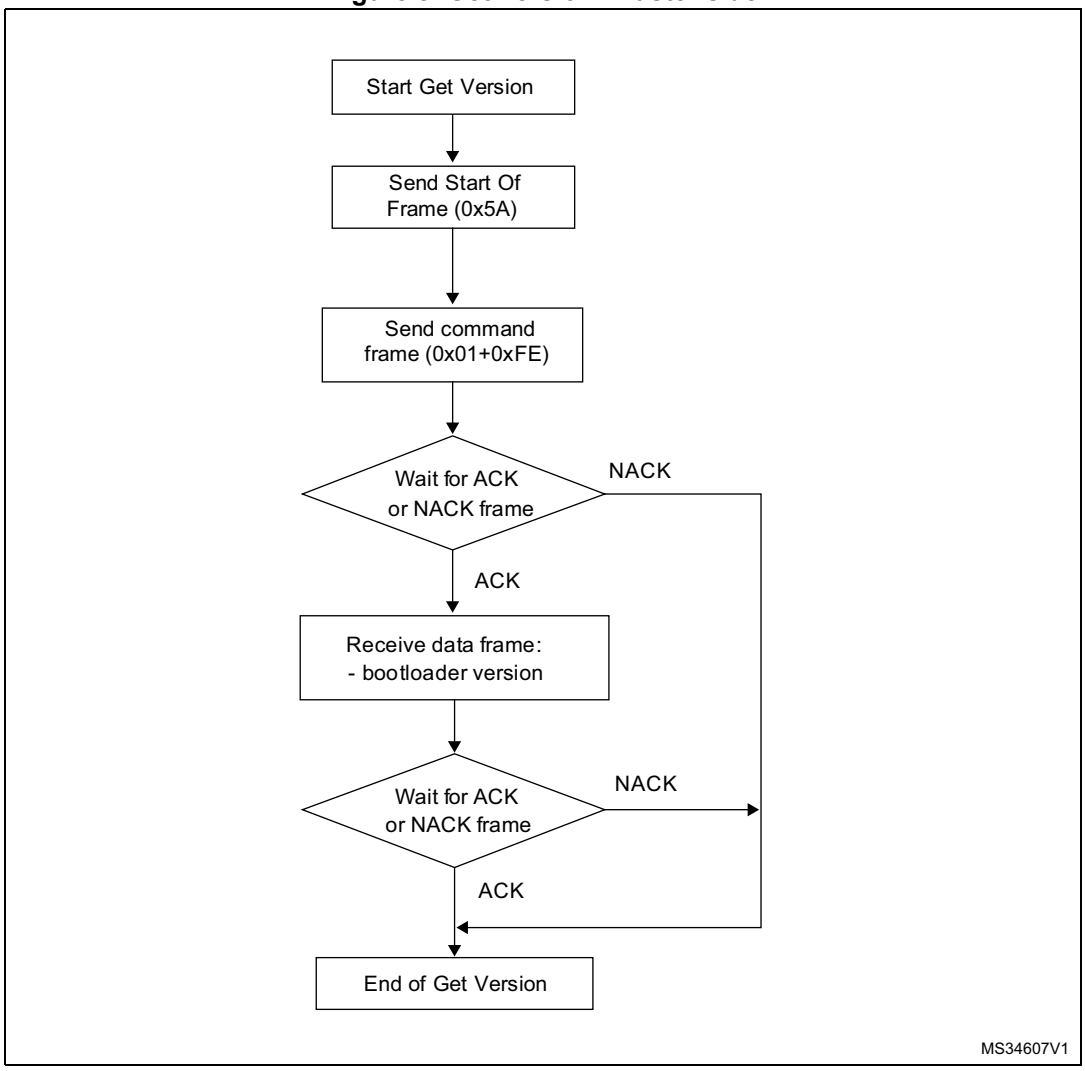
The STM32 sends the bytes as follows.

- 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)

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 information described below (bootloader version) to the master.

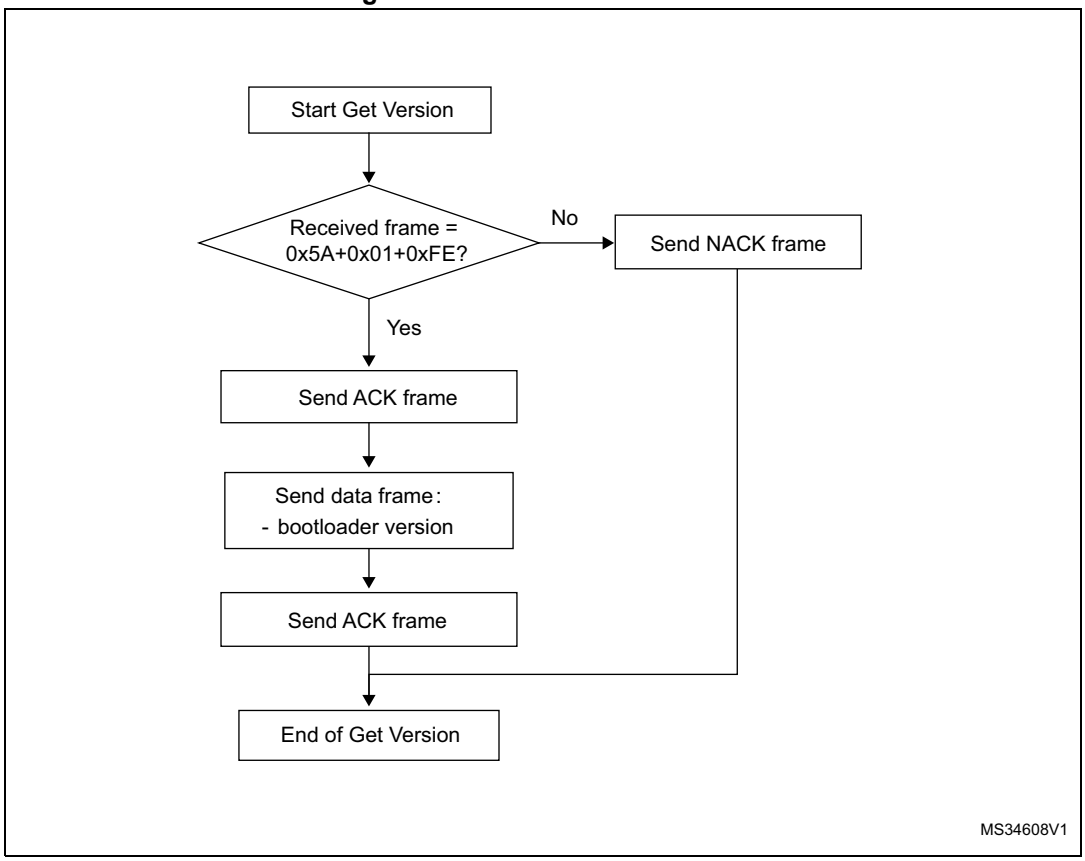
Figure 8. Get Version: master side



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: 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

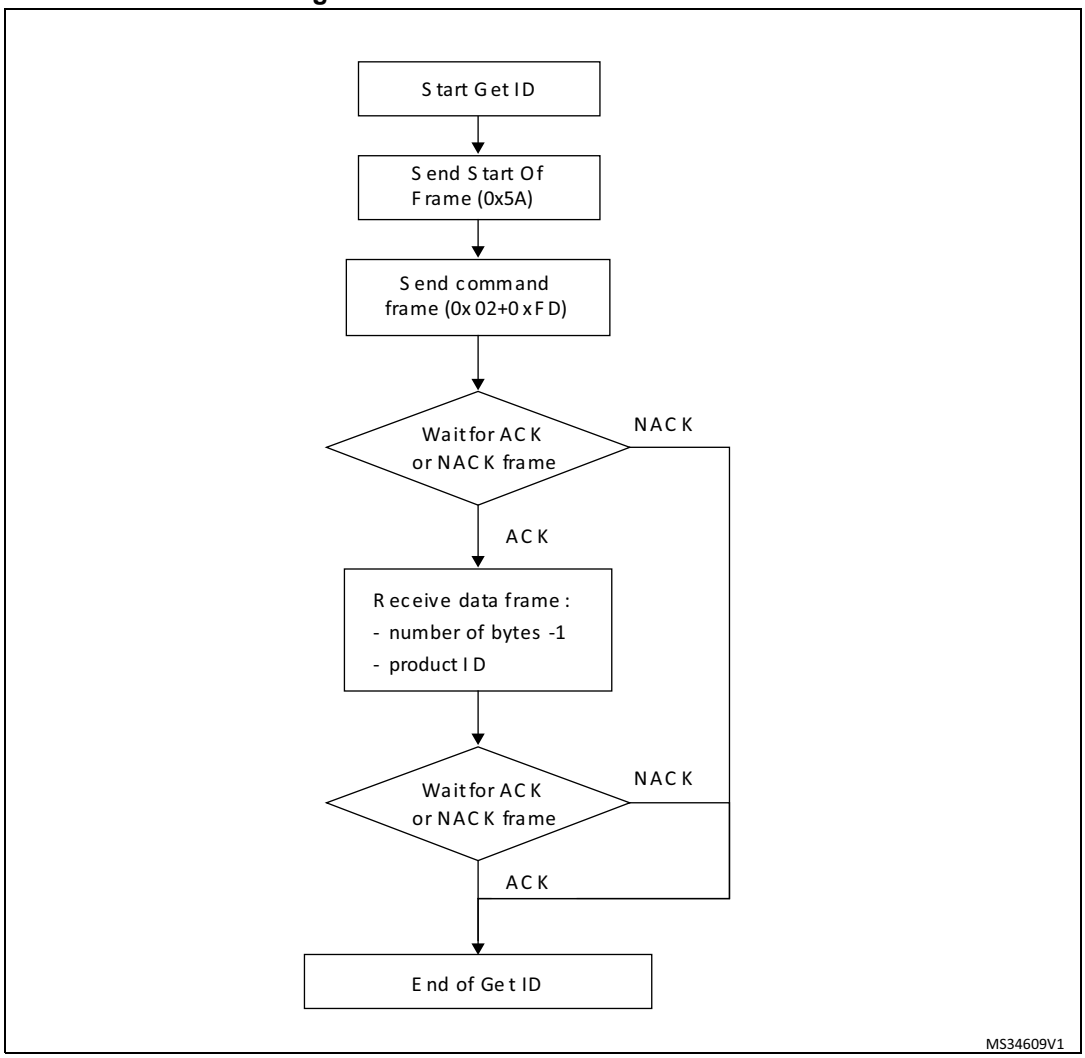
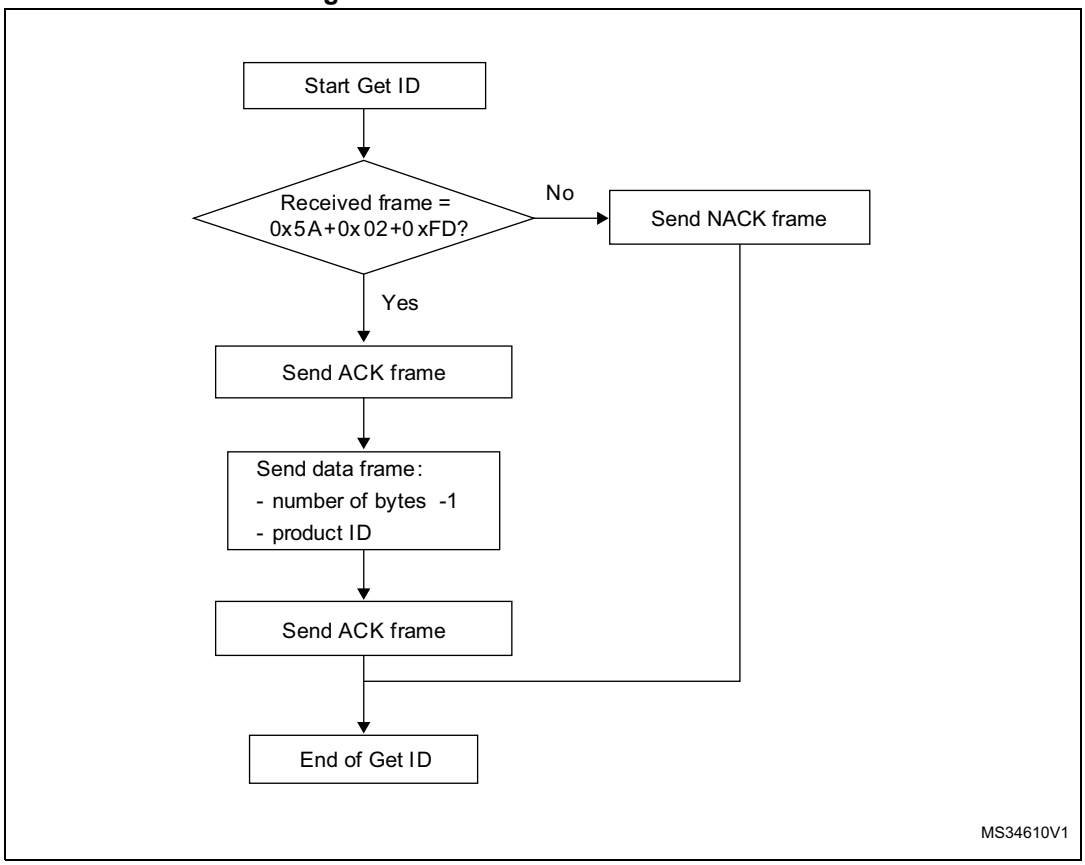


Figure 11. Get ID command: slave side



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 address 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)
- 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)
- 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

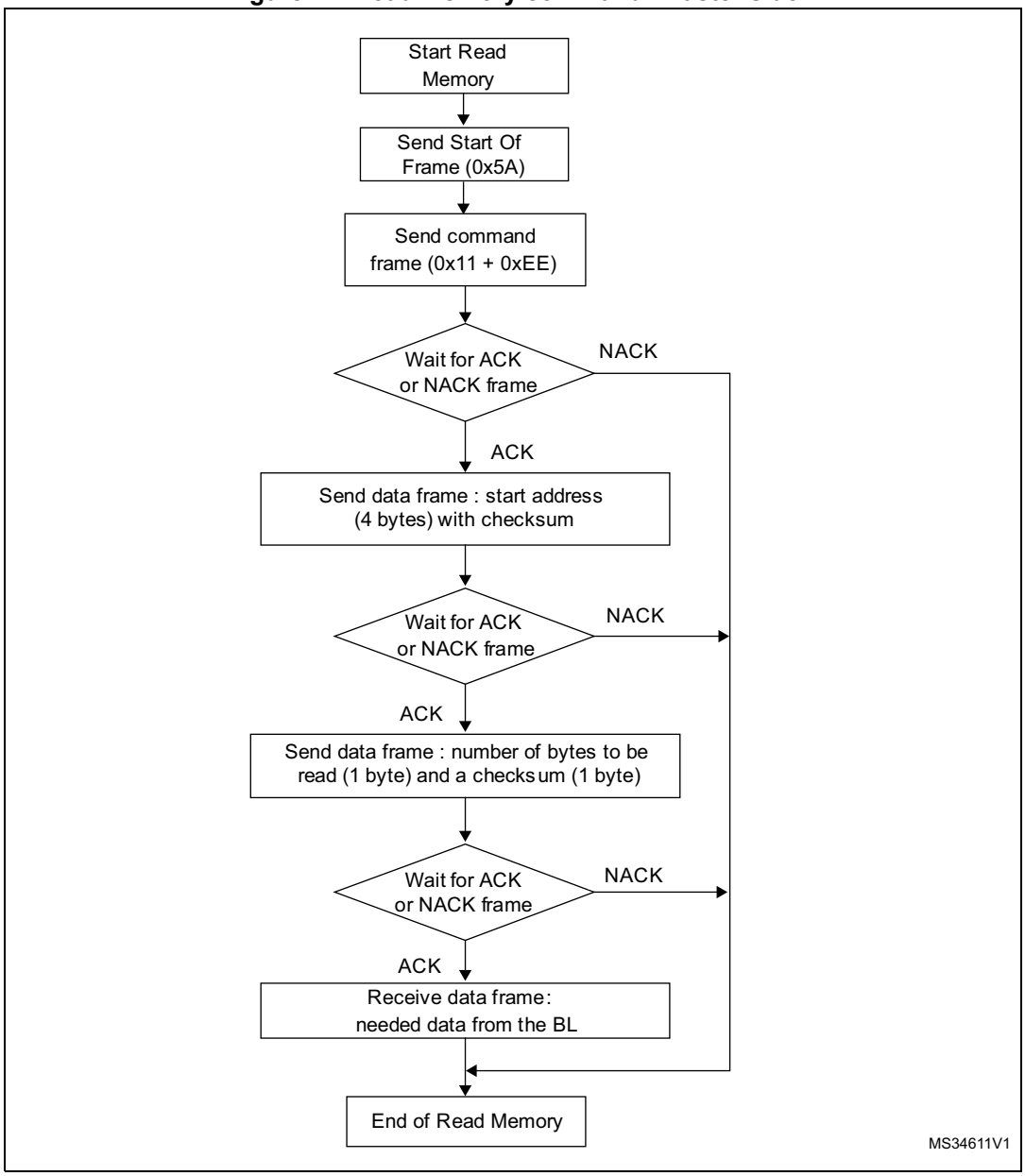
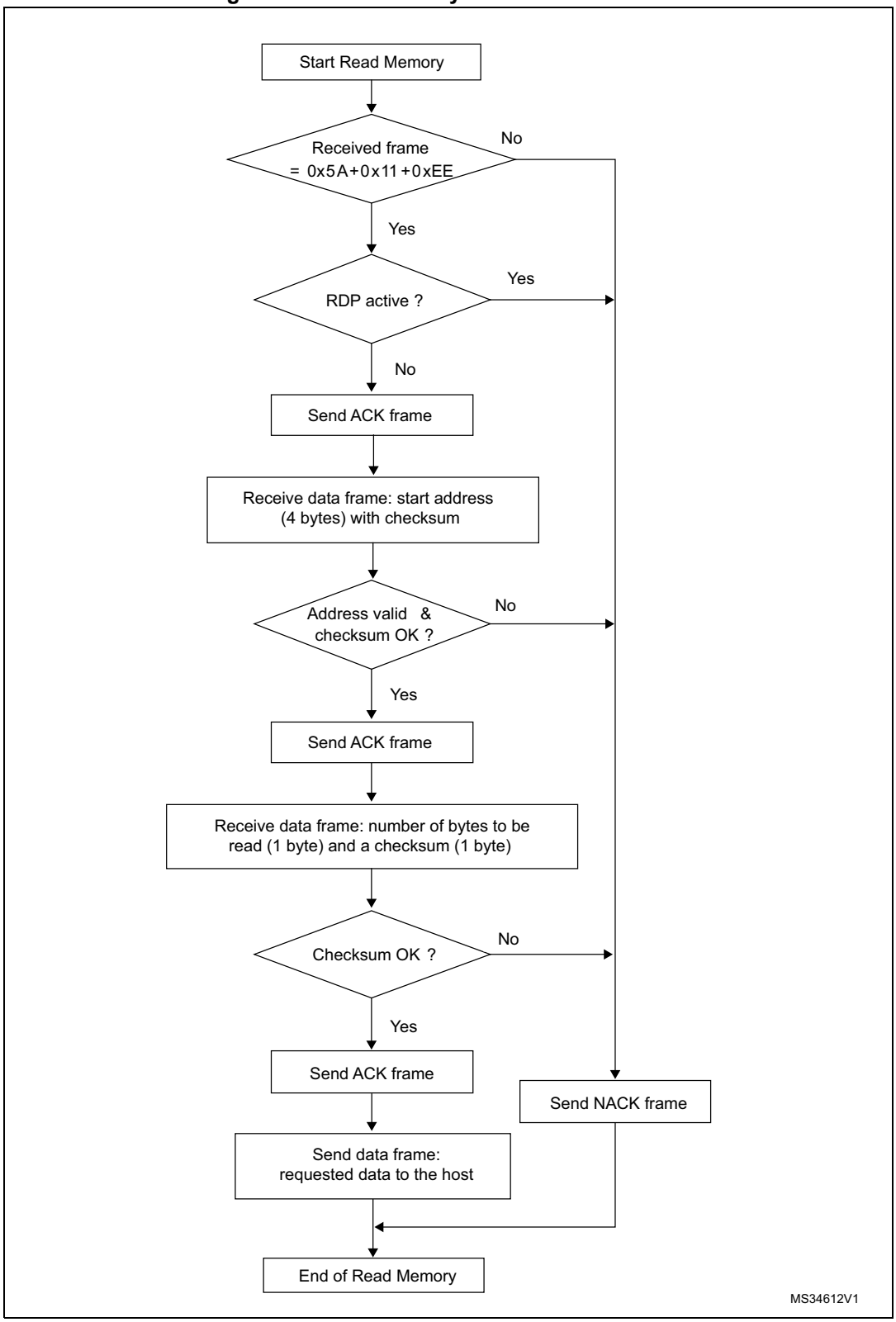


Figure 13. Read Memory command: slave side



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 address 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.

- Initializes the registers of the peripherals used by the bootloader to their default reset values.
- Initializes the user application's main stack pointer.
- Jumps to the memory location programmed in the received 'address + 4' (which corresponds to the address of the application's 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 should send 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 as follows to the STM32.

- Start Of Frame: 0x5A
- Byte 1: 0x21
- Byte 2: 0xDE
- Wait for ACK (as described in section 1)
- 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

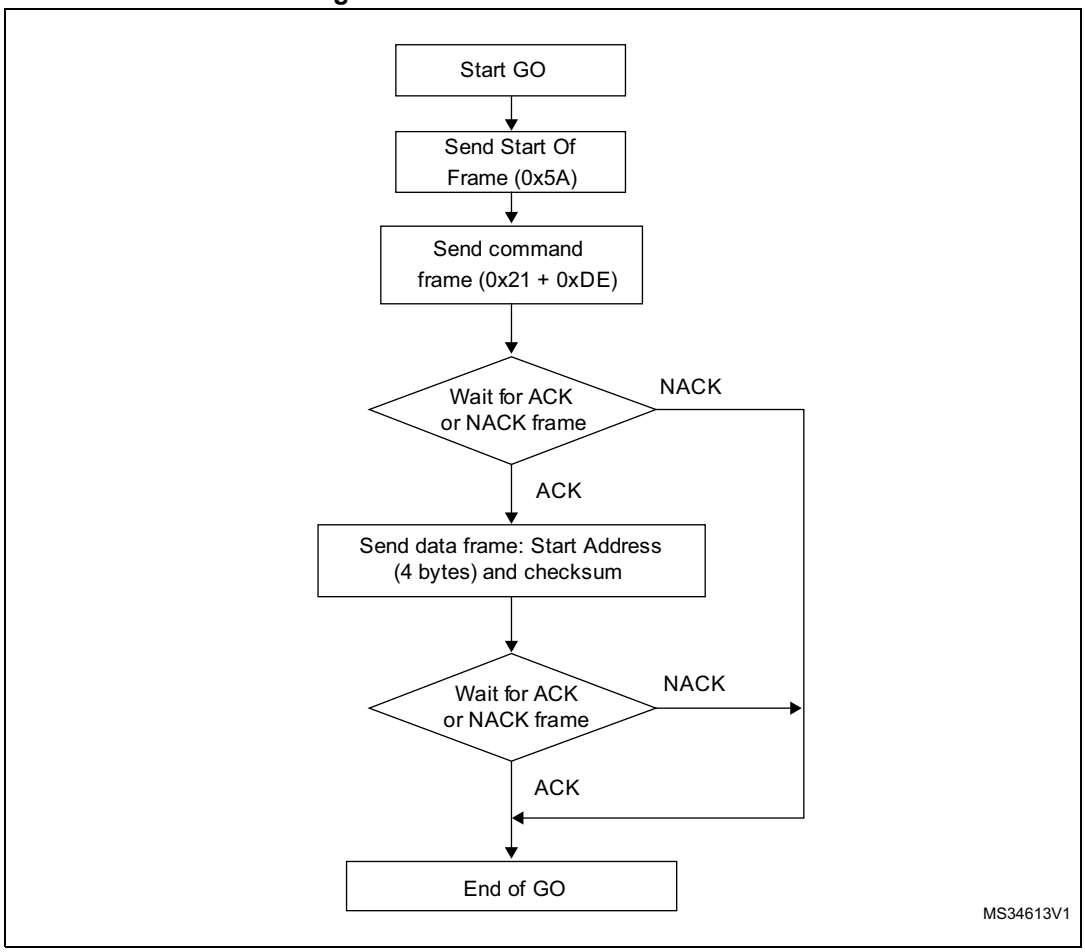
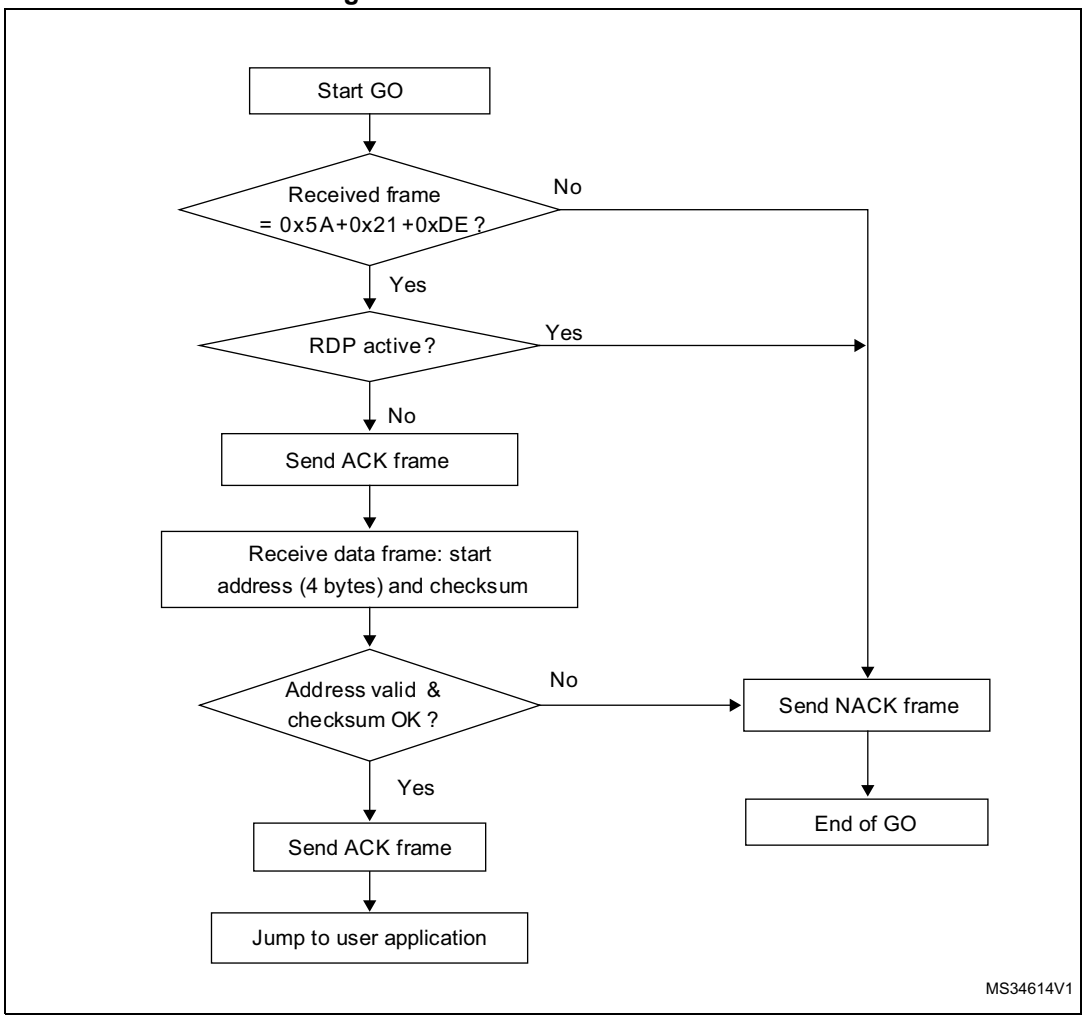


Figure 15. Go command: slave side



2.7 *Write Memory* command

The Write Memory command is used to write data to any valid memory 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 then does the following.

- 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's option bytes.

If the write memory 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 should be in multiples of two bytes. If less data is written, the remaining bytes should be filled by 0xFF.

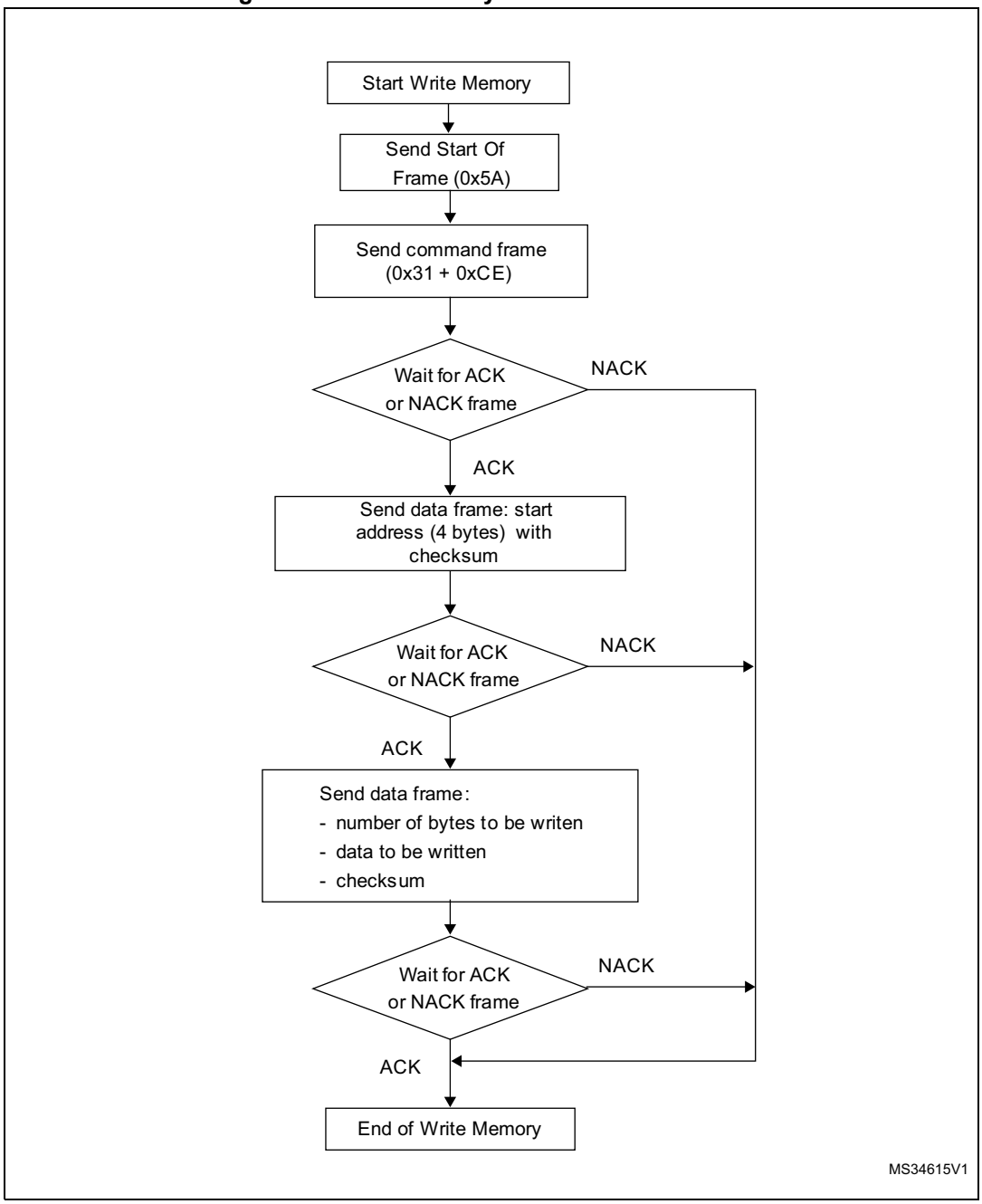
When writing to the RAM, you should take care not to overlap the first RAM memory used by the bootloader firmware.

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

The master sends the bytes to the STM32 as follows.

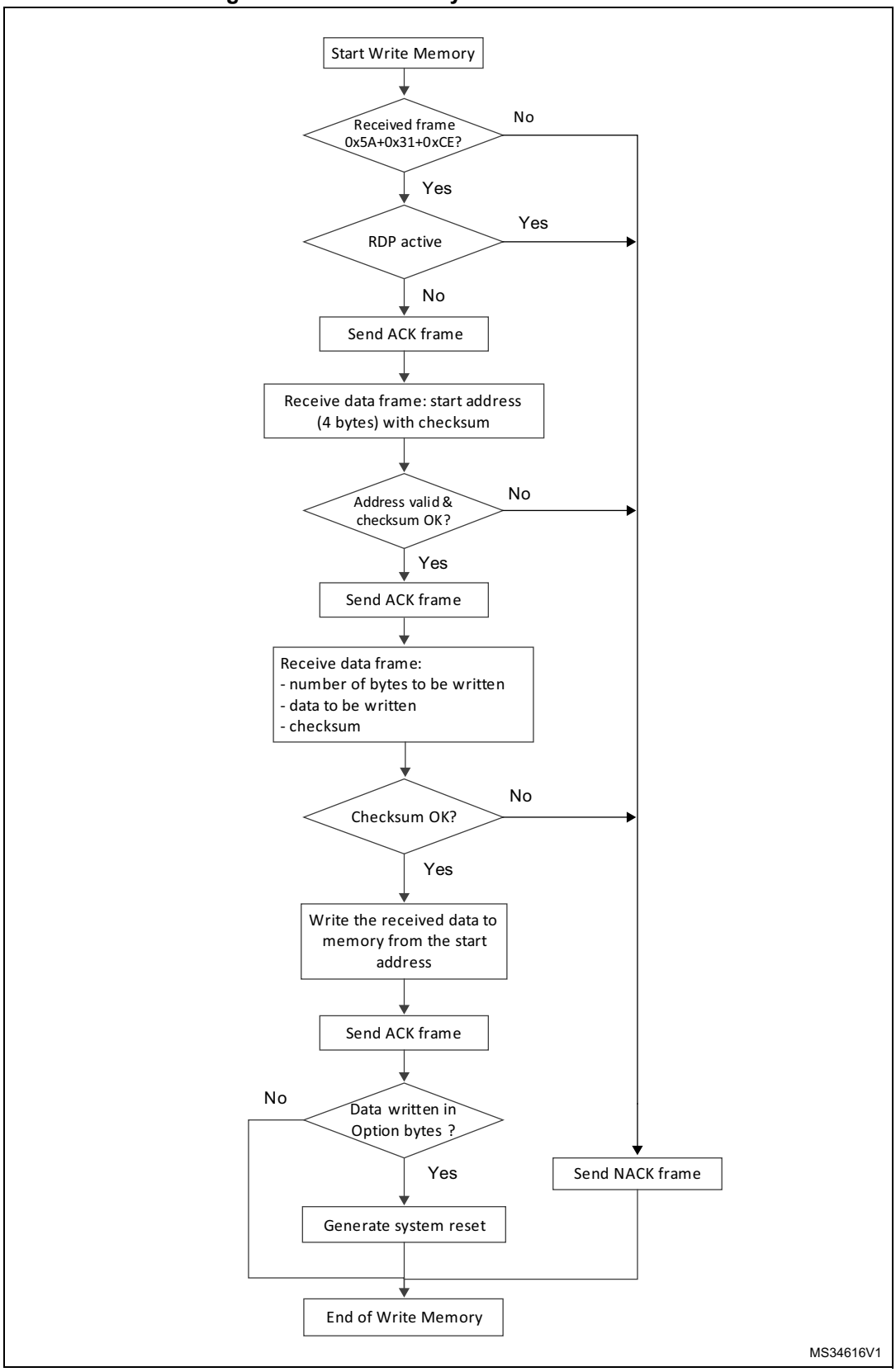
- Start Of Frame: 0x5A
- Byte 1: 0x31
- Byte 2: 0xCE
- Wait for ACK (as described in *Chapter 1*):
- Byte 3 to byte 6: start address
  - byte 3: MSB
  - byte 6: LSB
- Byte 7: checksum: XOR (Byte3, Byte4, Byte5, Byte6)
- Wait for ACK (as described in *Chapter 1*)
- Byte 8: number of bytes to be received (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 a Bank 1 mass erase (only for products supporting this feature).
  - 0xFFFD for a 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 relative to the product and thus should 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)
- 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

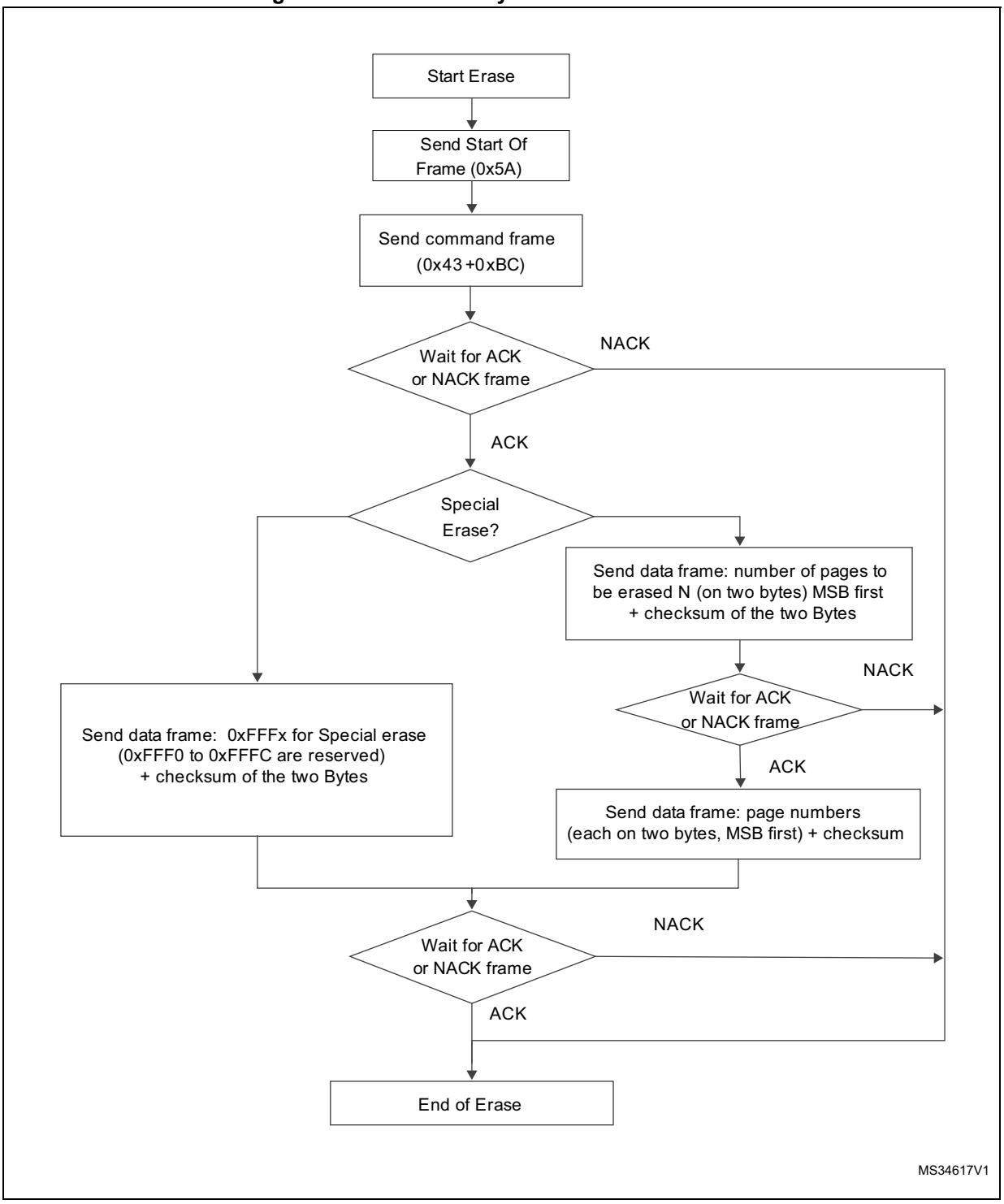
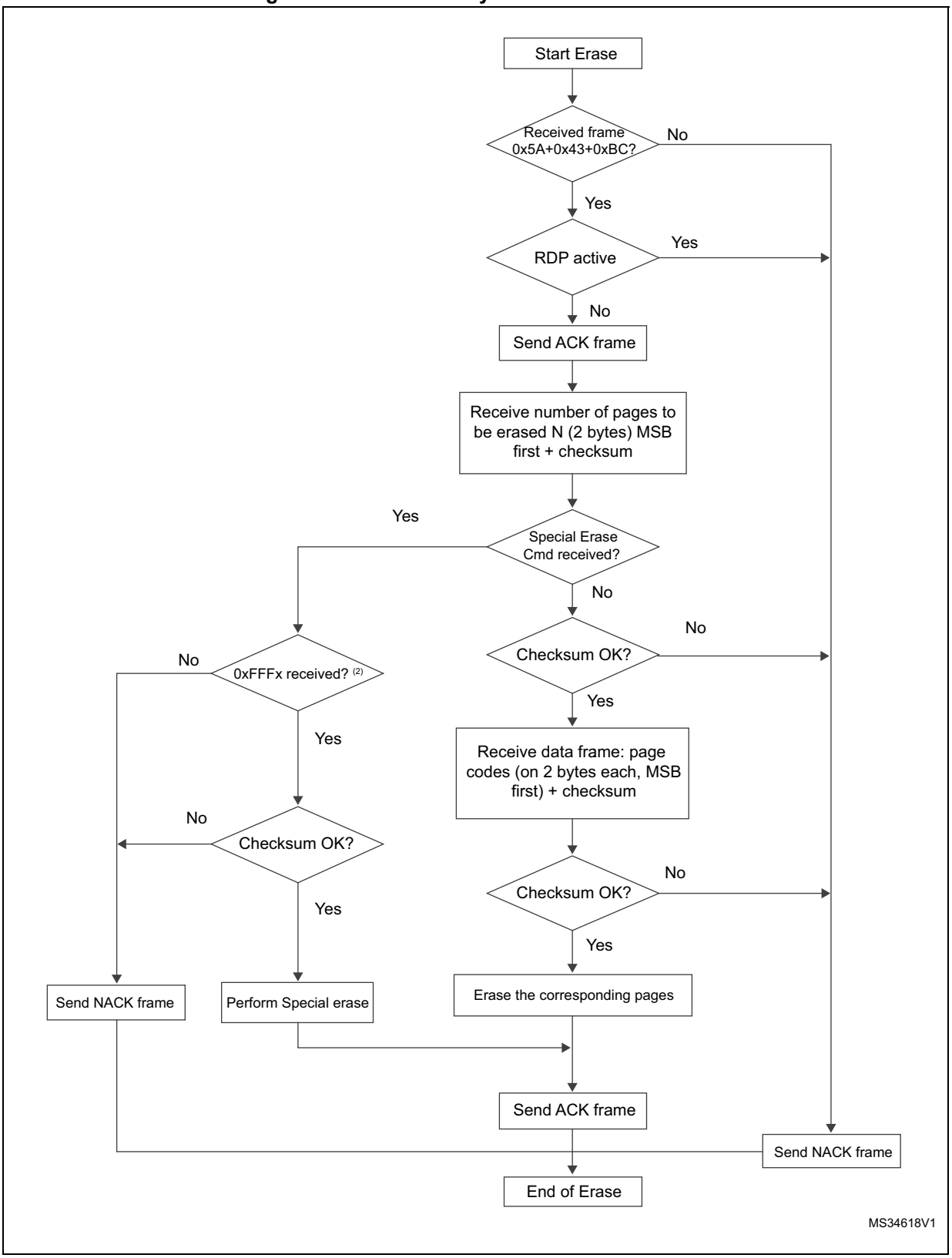


Figure 19. Erase Memory command: slave side



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 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, which 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 that had been 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

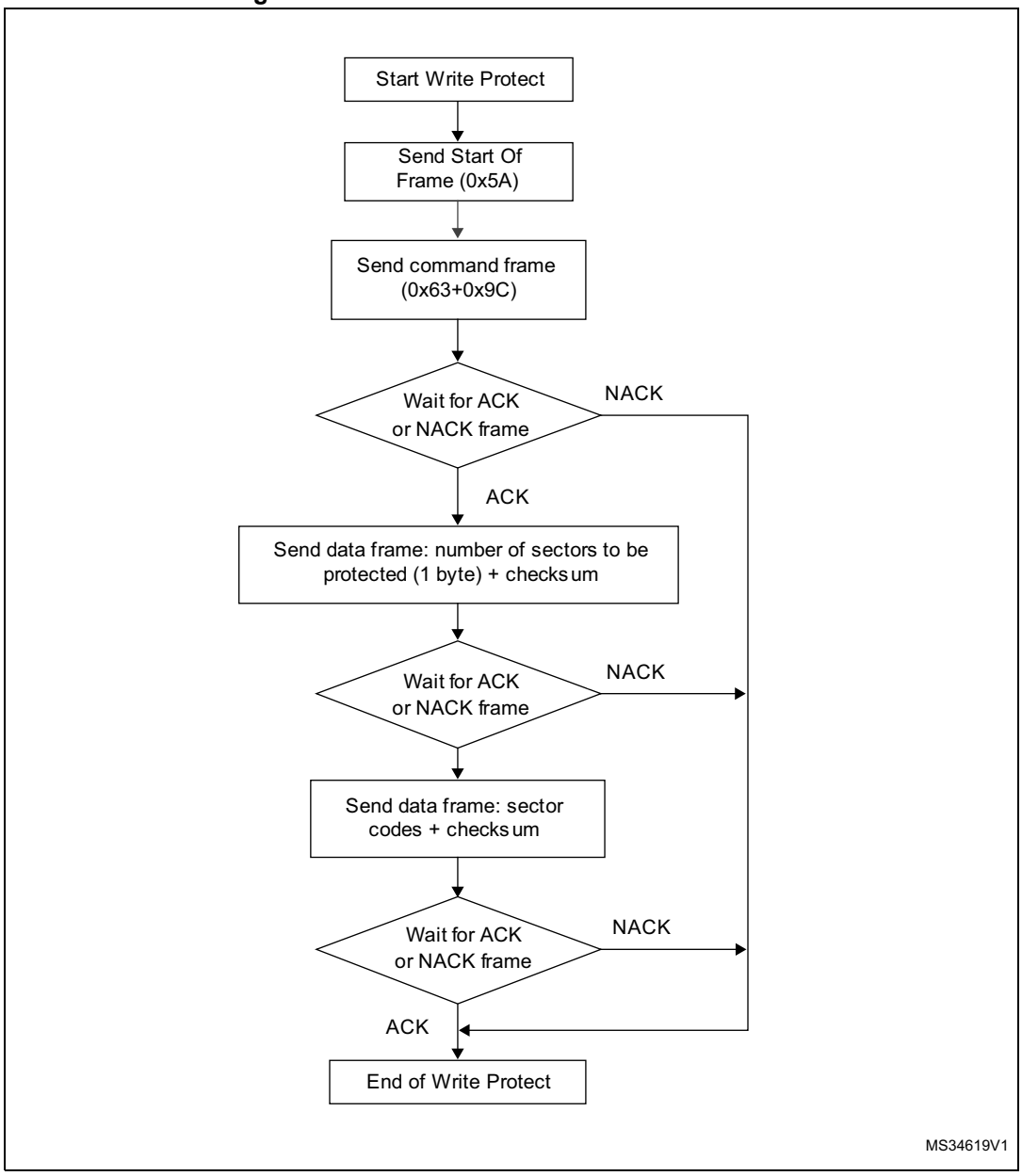
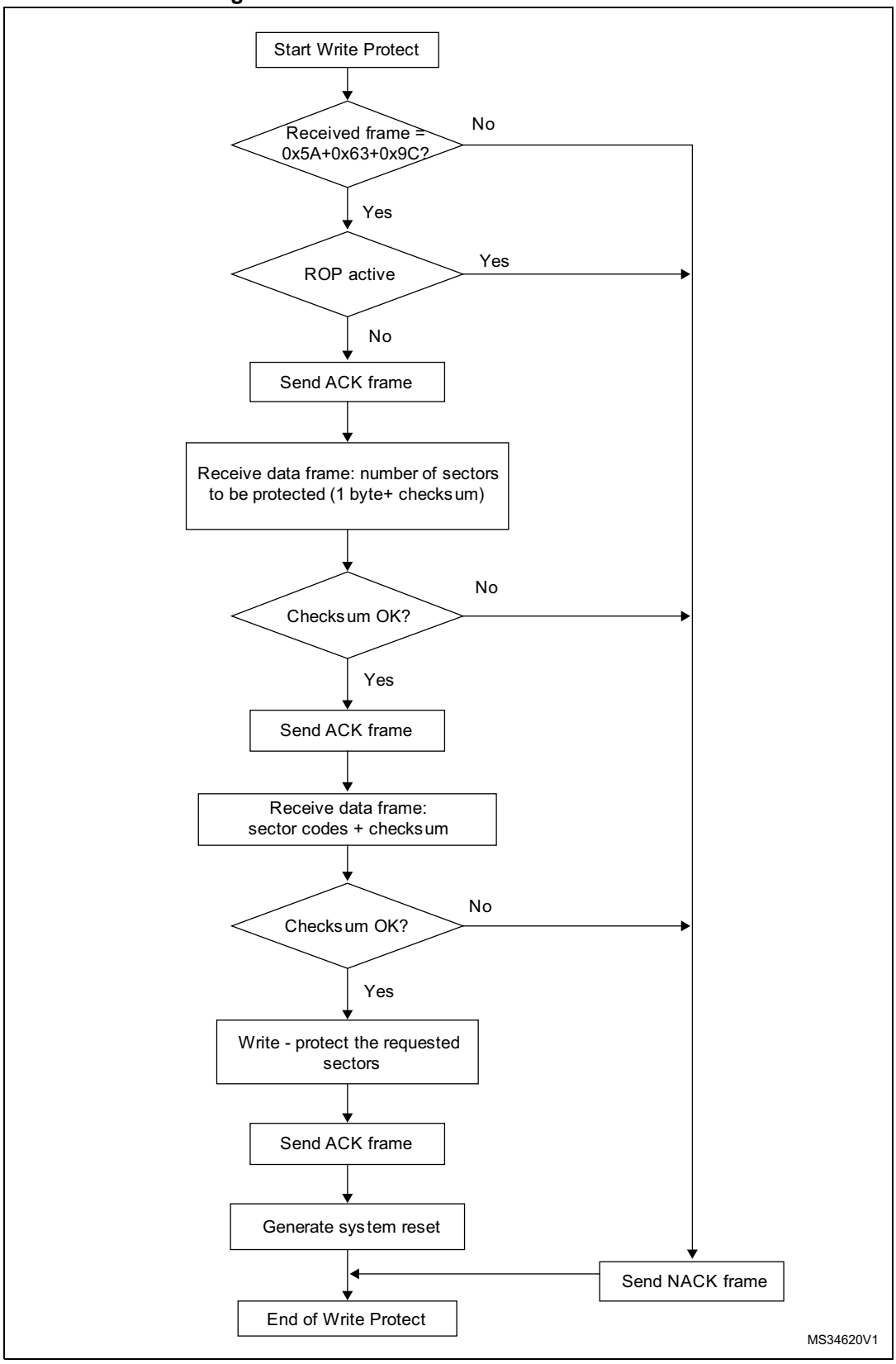


Figure 21. Write Protect command: slave side



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

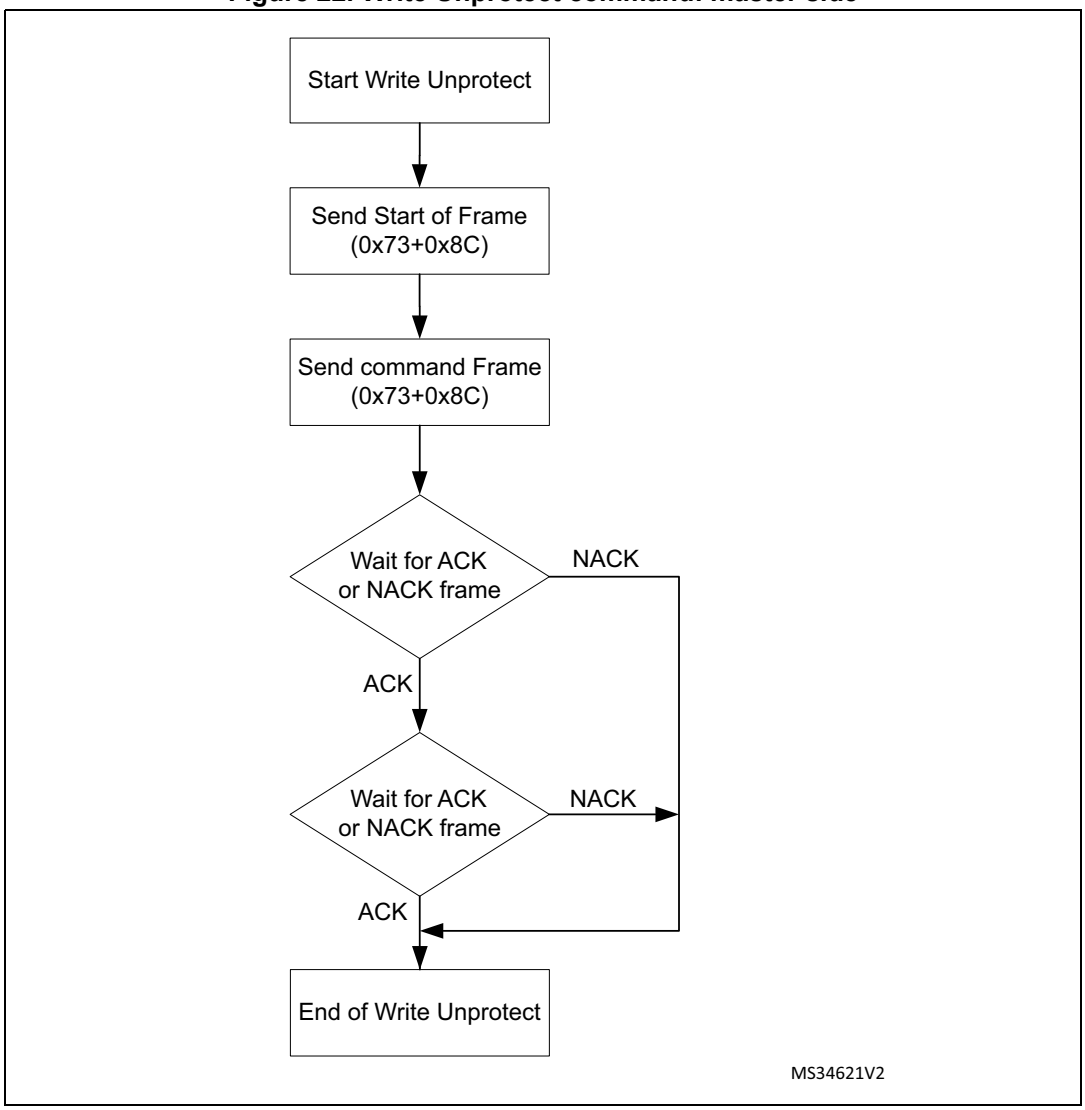
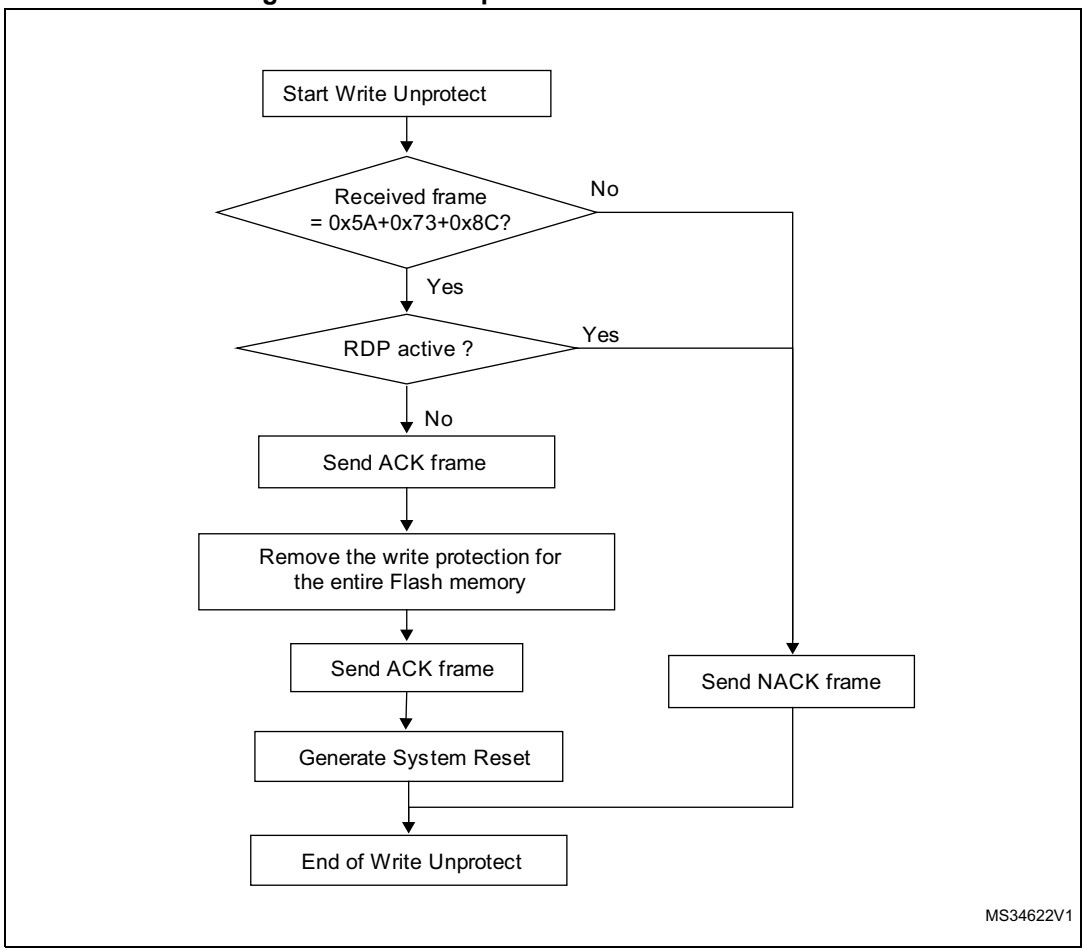


Figure 23. Write Unprotect command: slave side



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

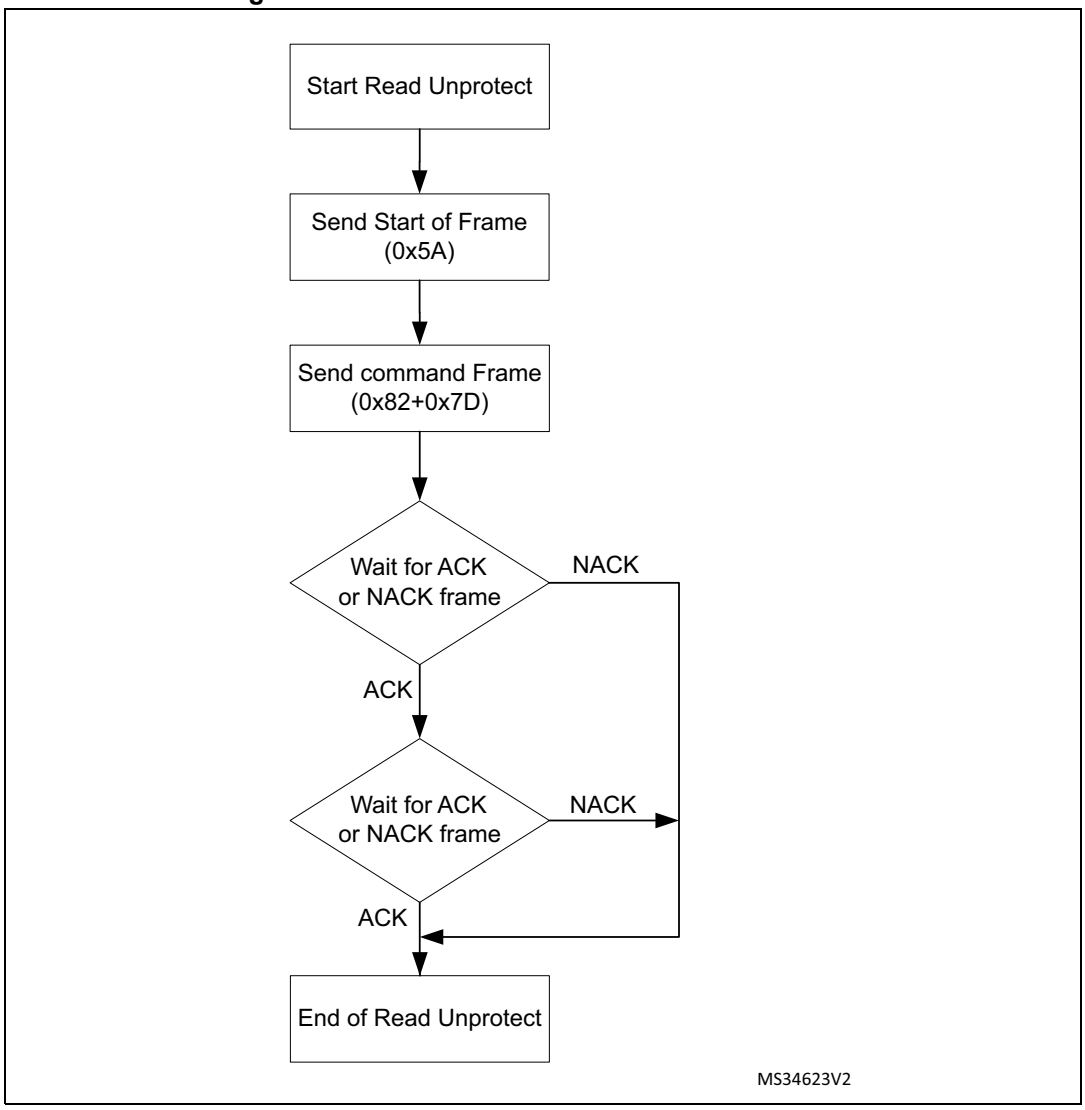
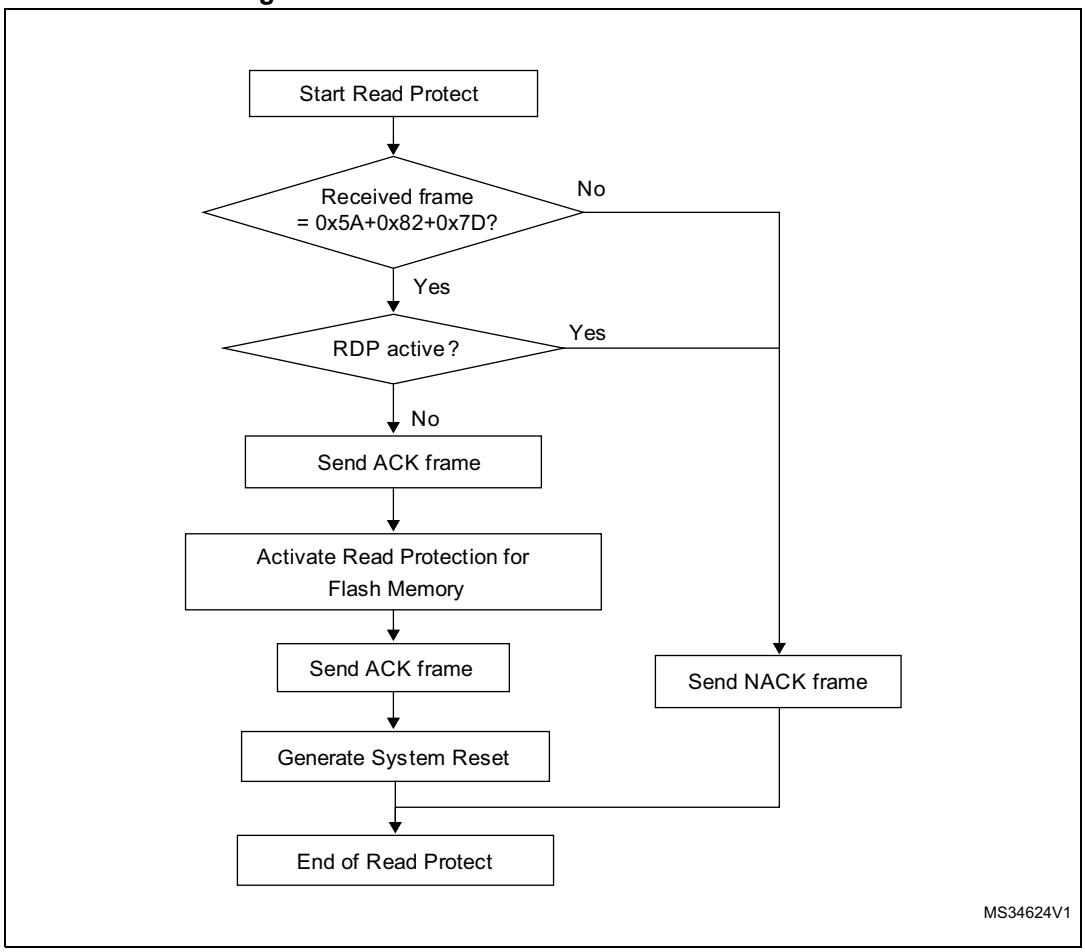


Figure 25. Readout Protect command: slave side



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 entire 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

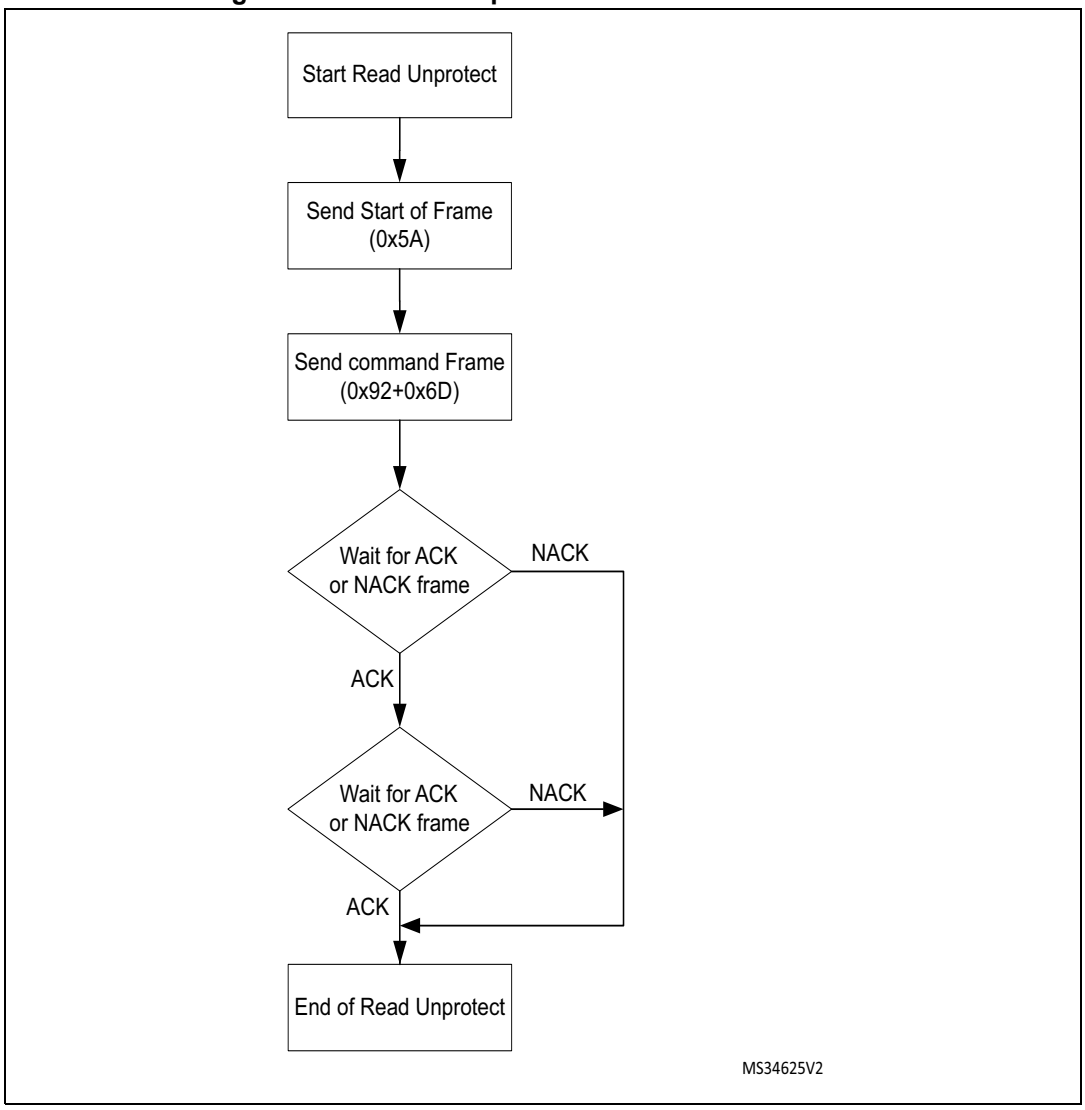
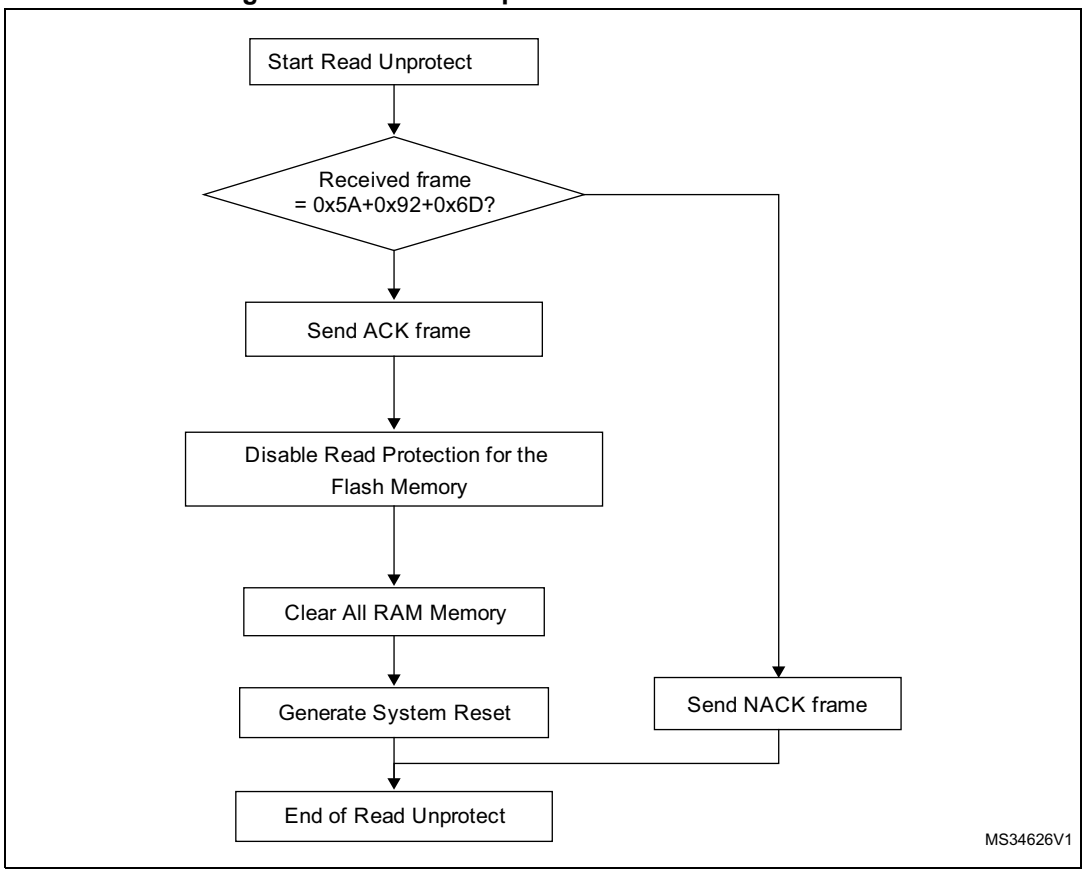


Figure 27. Readout Unprotect command: slave side



3 Evolution of the bootloader protocol versions

Table 3 lists the bootloader versions.

Table 3. Bootloader protocol versions

Version	Description
V1.1	Updated the Acknowledge mechanism. Updated the Get, GetID, GetVersion and Read commands.
V1.0	Initial bootloader version.

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*.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2014 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com