AN2606
Application note

STM32F101xx and STM32F103xx system memory boot mode

Introduction

This application note describes the bootloader stored in the system memory of the STM32F101xx and STM32F103xx Flash-memory-based microcontrollers. All STM32F10xxx in production include the bootloader detailed in this application note.

The bootloader is used to program the application into the internal Flash memory. For more information about the Flash module organization, refer to the STM32F101xx and STM32F103xx Flash programming manual (PM0042).

The specifications cover the architectural model of the bootloader for the STM32F10xxx family, but the low-level communication software supports all the microcontroller families that implement the bootloader.

Contents

List of tables

List of figures

1 Bootloader description

1.1 Bootloader introduction

The bootloader is stored in the internal boot ROM memory (system memory), and its main task is to download the application program to the internal Flash memory through a USART1 communication interface.

The main features of the bootloader are the following:

- it uses an embedded communication peripheral to download the code (peripherals are handled in polling mode)
- it transfers and updates the Flash memory code, the data, and the vector table sections

1.2 Bootloader activation

The bootloader is automatically activated by configuring the BOOT0 and BOOT1 pins in the specific “System memory” configuration (see *Table 1*) and then by applying a reset.

Depending on the used pin configuration, the Flash memory, system memory or SRAM is selected as the boot space, as shown in *Table 1* below.

Table 1. Boot pin configuration

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
X	0	User Flash memory	User Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

Table 1 shows that the STM32F10xxx enters the System memory boot mode if the BOOT pins are configured as follows:

- BOOT0 = 1
- BOOT1 = 0

The values on the BOOT pins are latched on the fourth rising edge of SYSCLK after a reset.

Table 2. STM32F10xxx configuration in System memory boot mode

Feature/Peripheral	State	Comment
Clock source	HSI enabled	The system clock is equal to 24 MHz using the PLL
USART1_RX pin	Input	PA10 pin: USART1 receives
USART1_TX pin	Output	PA9 pin: USART1 transmits
SysTick timer	Enabled	Used to automatically detect the serial baud rate from the host.
USART1	Enabled	Once initialized the USART1 configuration is: 8-bits, even parity and 1 Stop bit
RAM	-	512 bytes starting from address 0x2000 0000 are used by the booltloader firmware
System memory	-	2 Kbytes starting from address 0x1FFF F000, contain the bootloader firmware

Note: The system clock is derived from the embedded internal high-speed RC, no external quartz is required for bootloader code.

1.3 Hardware requirements

The hardware required to put the STM32F10xxx into System memory boot mode consists of any circuitry, switch or jumper, capable of holding the BOOT0 pin high and the BOOT1 pin low during reset.

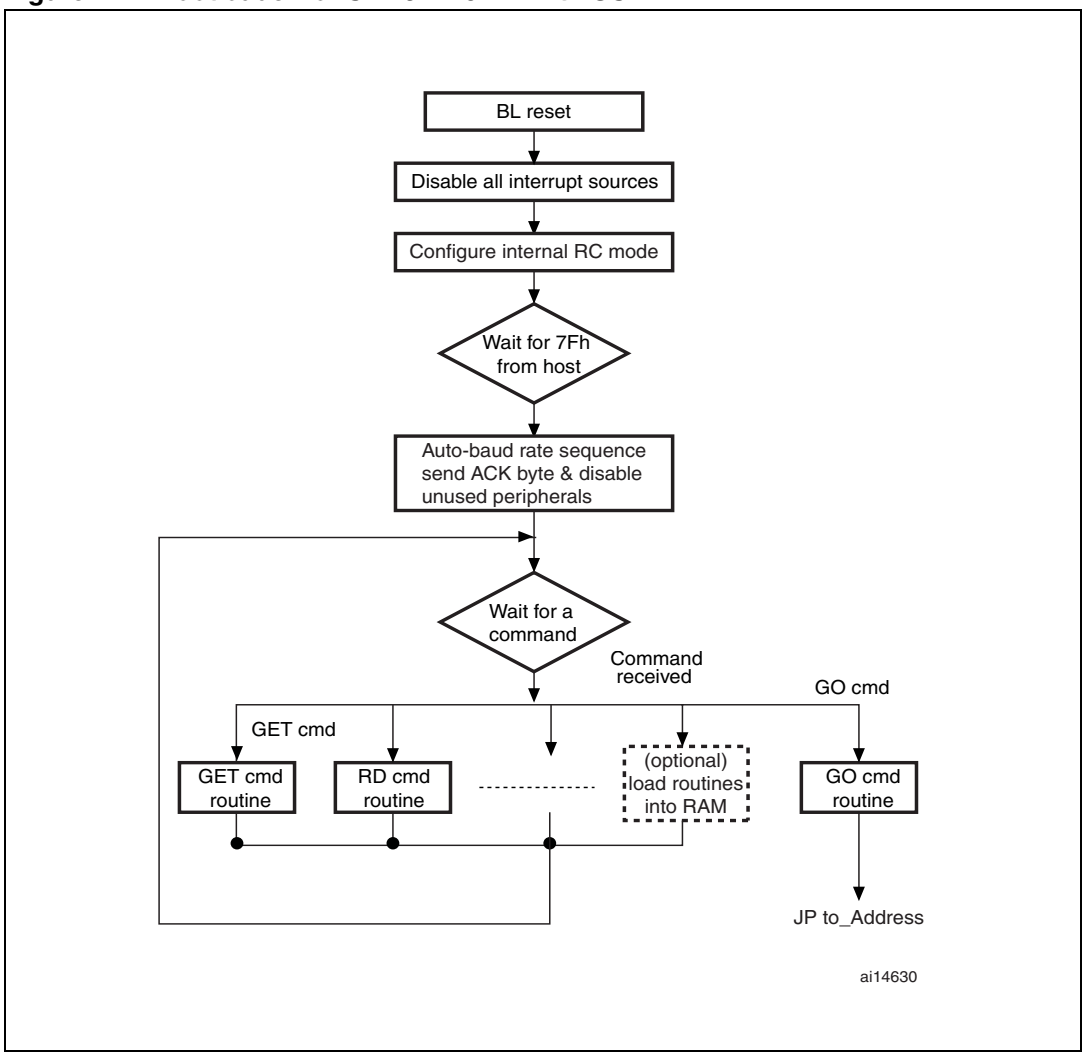
To connect to the STM32F10xxx during System memory boot mode, an RS232 serial interface (example, ST3232 RS232 transceiver) has to be directly linked to the USART1_RX (PA10) and USART1_TX (PA9) pins.

Note: USART1_CK, USART1_CTS and USART1_RTS pins are not used, therefore user can use these pins for other peripherals or GPIOs.

For more details about hardware recommendations, refer to application note AN2586: "*STM32F10xxx hardware development: getting started*", available from the STMicroelectronics website: *www.st.com*.

1.4 Bootloader code sequence

Figure 1. Bootloader for STM32F10xxx with USART1



Once the System memory boot mode is entered and the STM32F10xxx has been configured as described above, the bootloader code begins to scan the PA10 pin (USART1_RX line), waiting to receive 0x7F data: one start bit, 0x7F data bits, even parity bit and one stop bit.

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

Next, the code initializes the serial interface accordingly. Using this calculated baud rate, an acknowledge byte *(0x79)* is returned to the host, which signals that the STM32F10xxx is ready to receive user commands.

1.5 Choosing the USART baud rate

The calculation of the serial baud rate for USART1, from the length of the first byte that is received, is used to operate the bootloader within a wide range of baud rates. However, the upper and lower limits have to be kept, in order to ensure proper data transfer.

For a correct data transfer from the host to the STM32F10xxx, the maximum deviation between the internal initialized baud rate for USART1 and the real baud rate of the host should be below 2.5%. The deviation (f_B, in percent) between the host baud rate and the STM32F10xxx baud rate can be calculated using the formula below:

$$f_B = \left| \frac{\text{STM32Fxxx baud rate} - \text{Host baud rate}}{\text{STM32Fxxx baud rate}} \right| \times 100\% \text{, where } f_B \leq 2.5\%.$$

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

1.5.1 Minimum baud rate

The lowest tested baud rate (B_{Low}) is 1200. Baud rates below B_{Low} would cause the SysTick timer to overflow. In this event, USART1 would not be correctly initialized.

1.5.2 Maximum baud rate

B_{High} is the highest baud rate for which the deviation still does not exceed the limit. All baud rates between B_{Low} and B_{High} are below the deviation limit.
The highest tested baud rate (B_{High}) is 115 200.

1.6 Exiting System memory boot mode

System memory boot mode must be exited in order to execute a program in a normal User mode. The STM32F10xxx may exit this mode by applying a hardware reset. At the time of reset, the BOOT pins (BOOT0 and BOOT1) must be set at the proper levels to enter the desired user mode (see *Table 1*). Following the reset, the CPU starts code execution from the boot memory located at the bottom of the memory address space starting from 0x0000 0000.

2 Bootloader command set

The supported commands are listed in *Table 3* below. Each command is further described in this section.

Table 3. Bootloader commands

Command[1]	Command code	Command description
Get[2]	0x00	Gets the version and the allowed commands supported by the current version of the bootloader
Get Version & Read Protection Status[2]	0x01	Gets the bootloader version and the Read Protection status of the Flash memory
Get ID[2]	0x02	Gets the chip ID
Read Memory	0x11	Reads up to 256 bytes of memory starting from an address specified by the user
Go	0x21	Jumps to an address specified by the user to execute (a loaded) code
Write Memory	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the user
Erase	0x43	Erases from one to all the Flash memory pages
Write Protect[3]	0x63	Enables the write protection for some sectors
Write Unprotect[3]	0x73	Disables the write protection for all Flash memory sectors
Readout Protect[2]	0x82	Enables the read protection
Readout Unprotect[2]	0x92	Disables the read protection

1. If a denied command is received or an error occurs during the command execution, the bootloader sends NACK byte and goes back to command checking.
2. Read protection – When the RDP (read protection) option is active, only this limited subset of commands is available. All other commands are NACKed and have no effect on the device. Once the RDP has been removed, the other commands become active.
3. On the STM32F10xxx, the sector size is 4 KBytes so, depending on the product, this could be 4 pages (when the page is 1 Kbyte) or 2 pages (when the page is 2 KBytes) for the Write Protect and Write Unprotect commands.

Communication safety

All communications from the programming tool (PC) to the device are verified by:

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

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

- ACK = 0x79
- NACK = 0x1F

2.1 Get command

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

Figure 2. Get command: host side

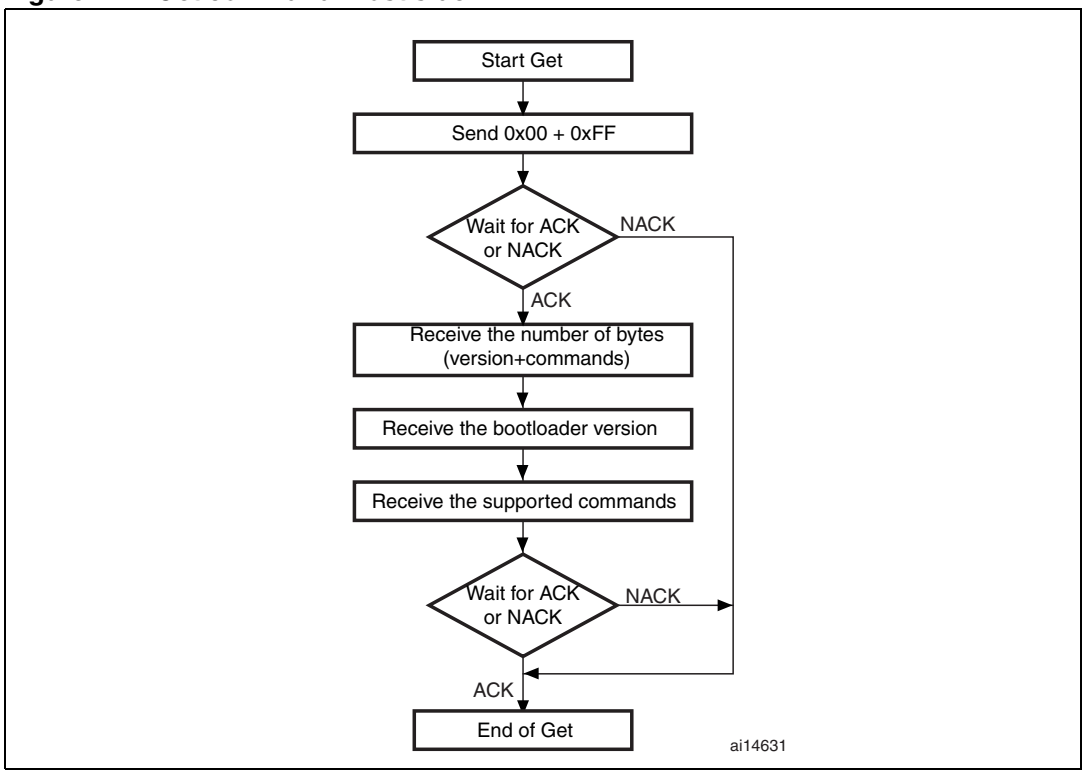
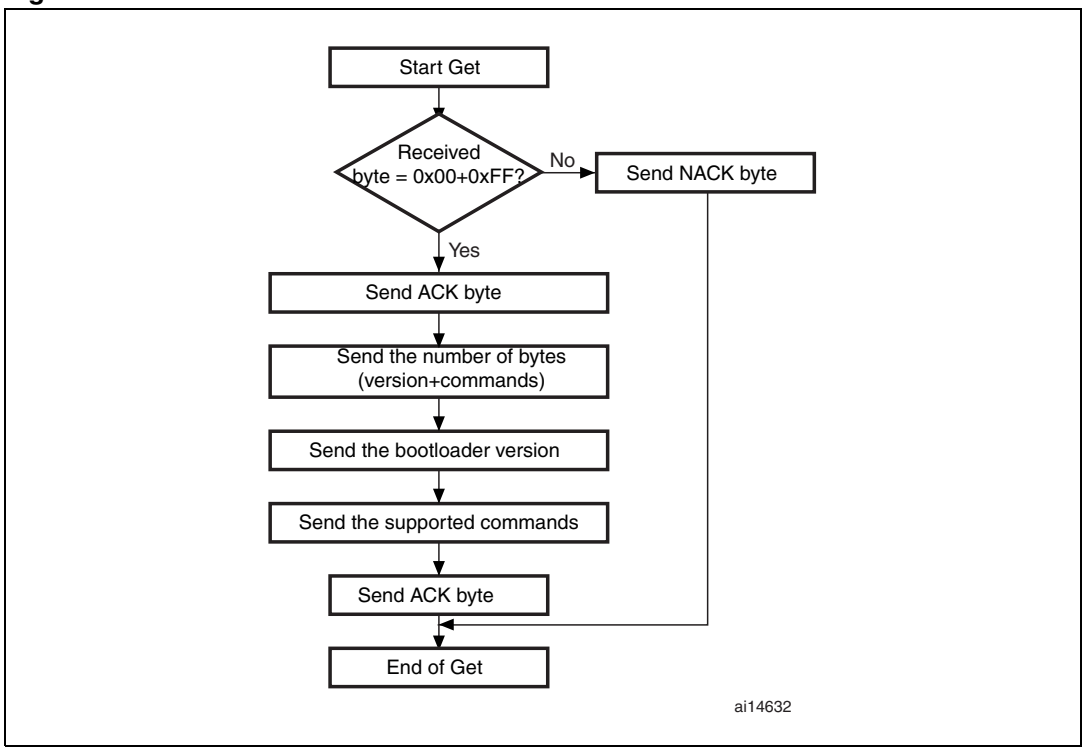


Figure 3. Get command: device side



The STM32F10xxx 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): 0x21 = Version 2.1

Byte 4: 0x00 – Get command

Byte 5: 0x01 – Get Version and Read Protection Status

Byte 6: 0x02 – Get ID

Byte 7: 0x11 – Read Memory command

Byte 8: 0x21 – Go command

Byte 9: 0x31 – Write Memory command

Byte 10: 0x43 – Erase command

Byte 11: 0x63 – Write Protect command

Byte 12: 0x73 – Write Unprotect command

Byte 13: 0x82 – Readout Protect command

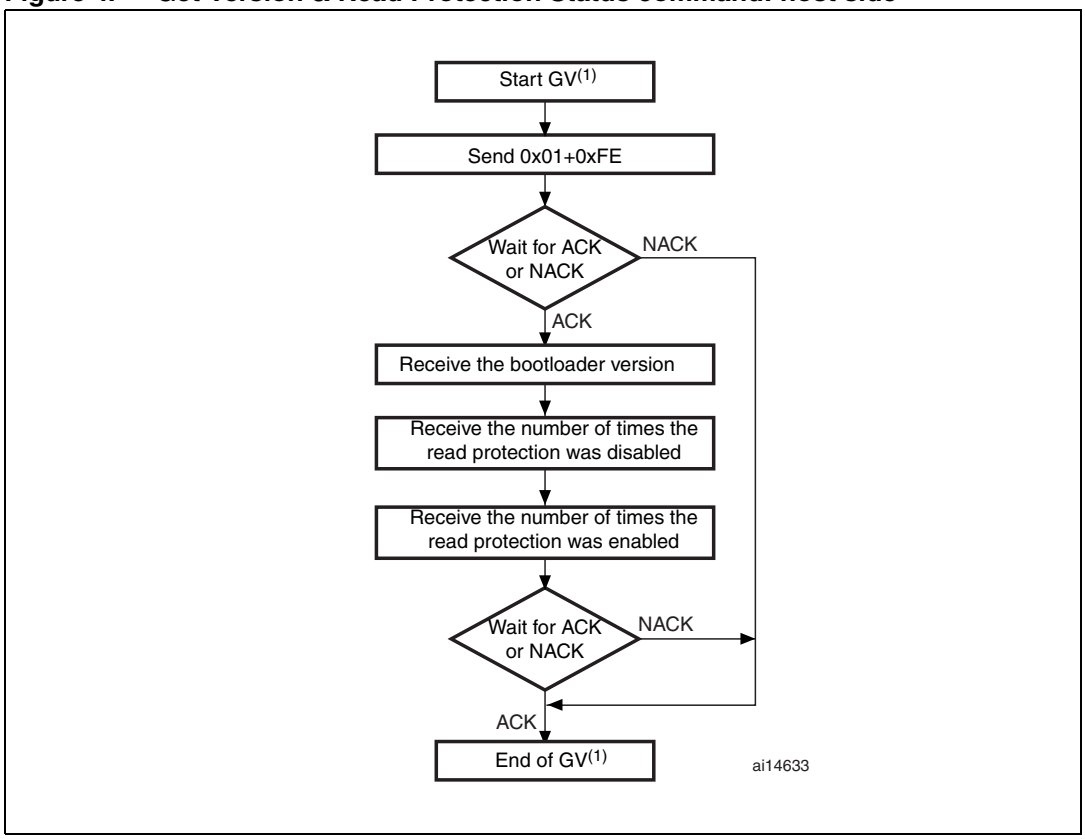
Byte 14: 0x92 – Readout Unprotect command

Last byte (15): ACK

2.2 Get Version & Read Protection Status command

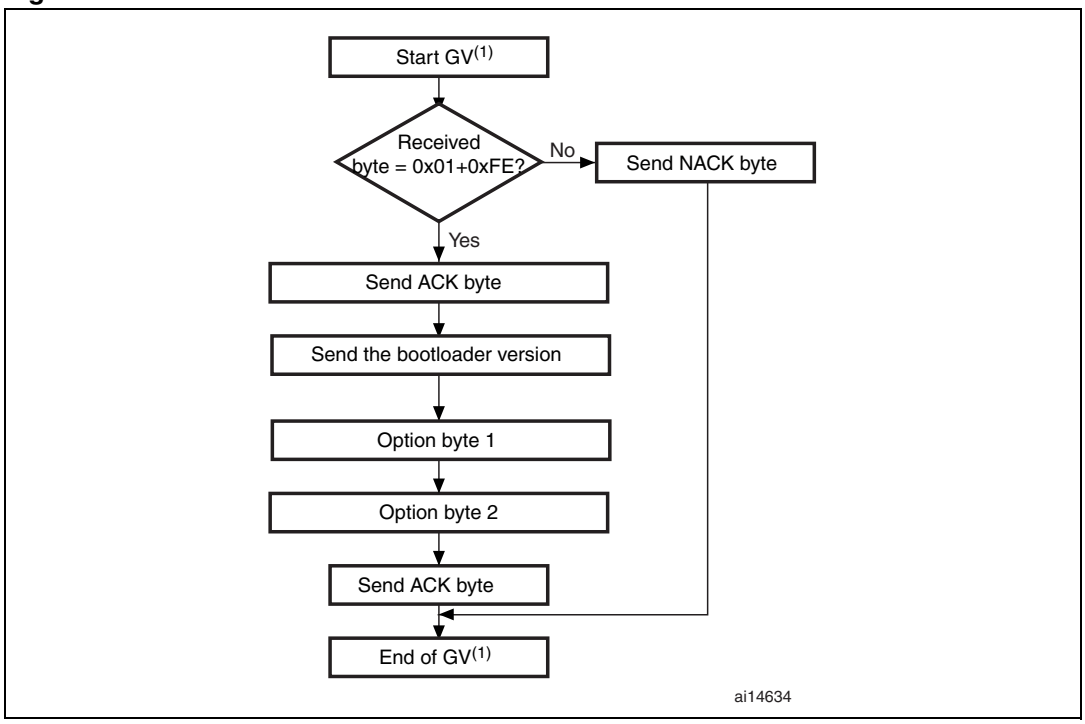
The Get Version & Read Protection Status command is used to get the bootloader version and the read protection status. When the bootloader receives the command, it transmits the information described below (version, read protection: number of times it was enabled and disabled) to the host.

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



1. GV = Get Version & Read Protection Status.

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



1. GV = Get Version & Read Protection Status.

The STM32F10xxx sends the bytes as follows:

Byte 1: ACK

Byte 2: The version of the bootloader (0 < Version ≤255)): 0x21 = Version 2.1

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

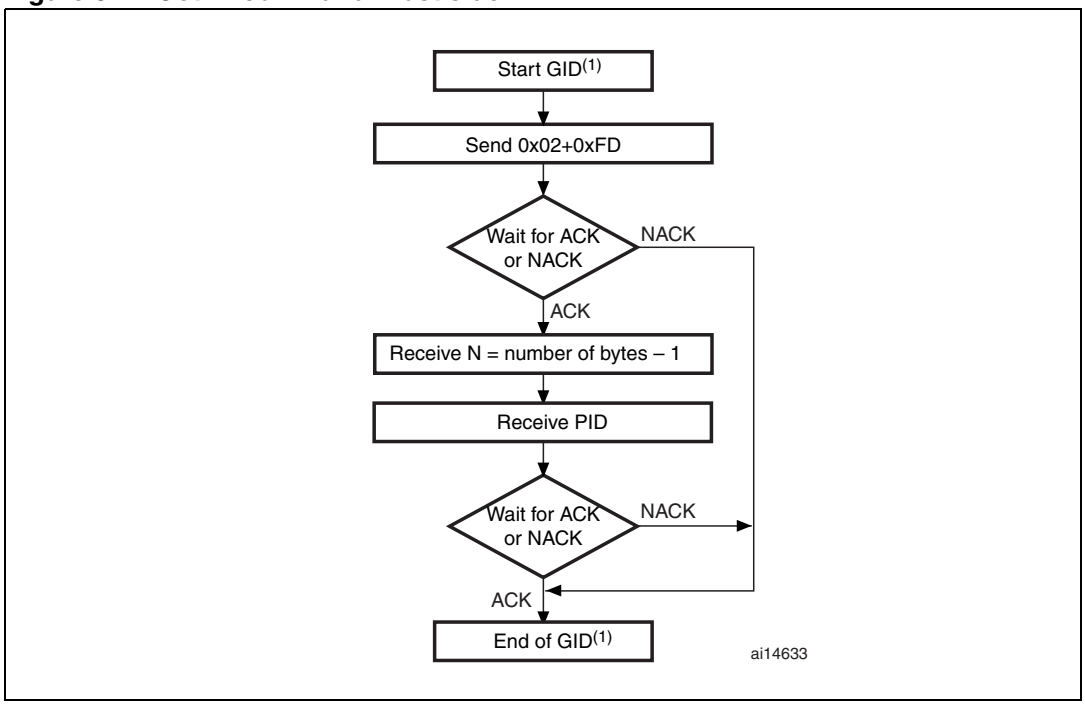
Byte 4: Option byte 2: 0x00 to keep the compatibility with generic bootloader protocol

Byte 5: ACK

2.3 Get ID command

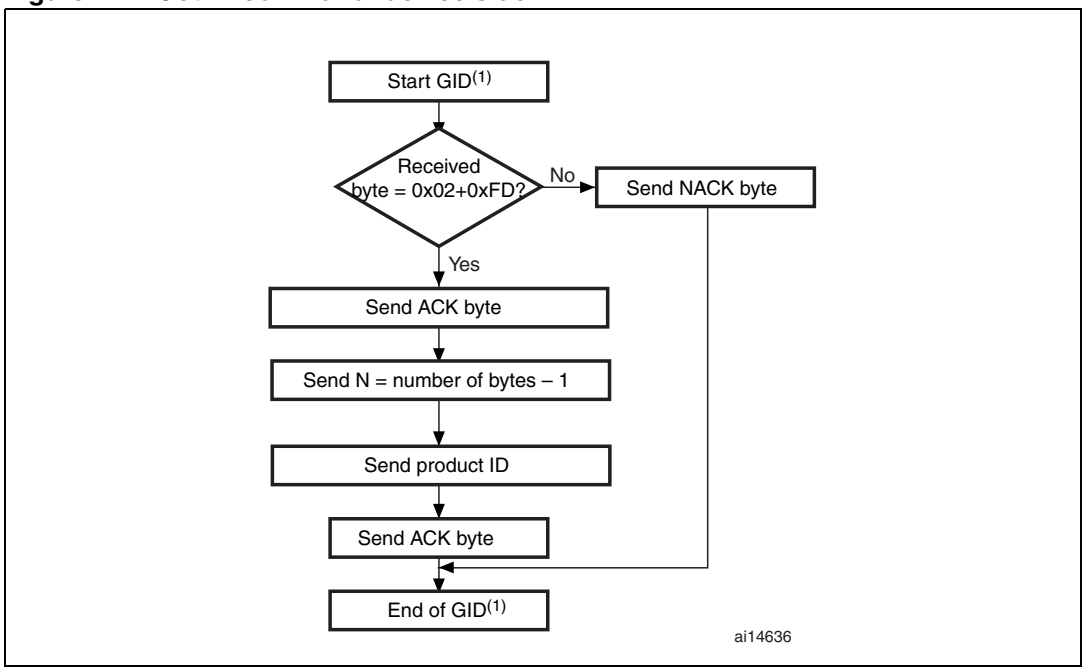
The Get ID command is used to get the version of the chip ID (identification). When the bootloader receives the command, it transmits the product ID to the host.

Figure 6. Get ID command: host side



1. GID = Get ID.

Figure 7. Get ID command: device side



1. GID = Get ID.

The STM32F10xxx sends the bytes as follows:

Byte 1: ACK

Byte 2: N = the number of bytes – 1 (N = 1 for STM32F10xxx), except for current byte and ACKs.

Bytes 3-4: PID(1) byte 3 = 0x04, byte 4 = 0x1X

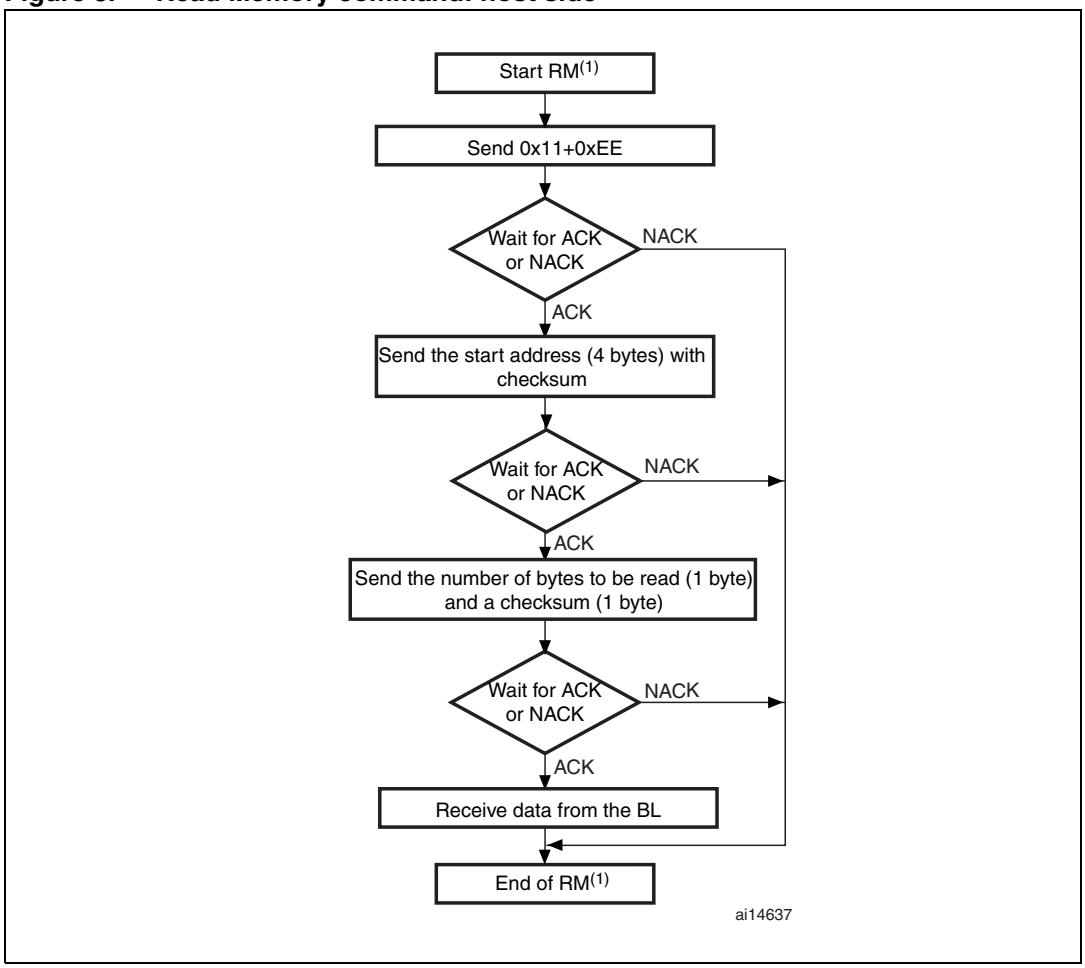
Byte 5: ACK

1. PID: is the product ID, which may be 0x0410, 0x0412 or 0x0414 according to the STM32F10xxx product.

2.4 Read Memory command

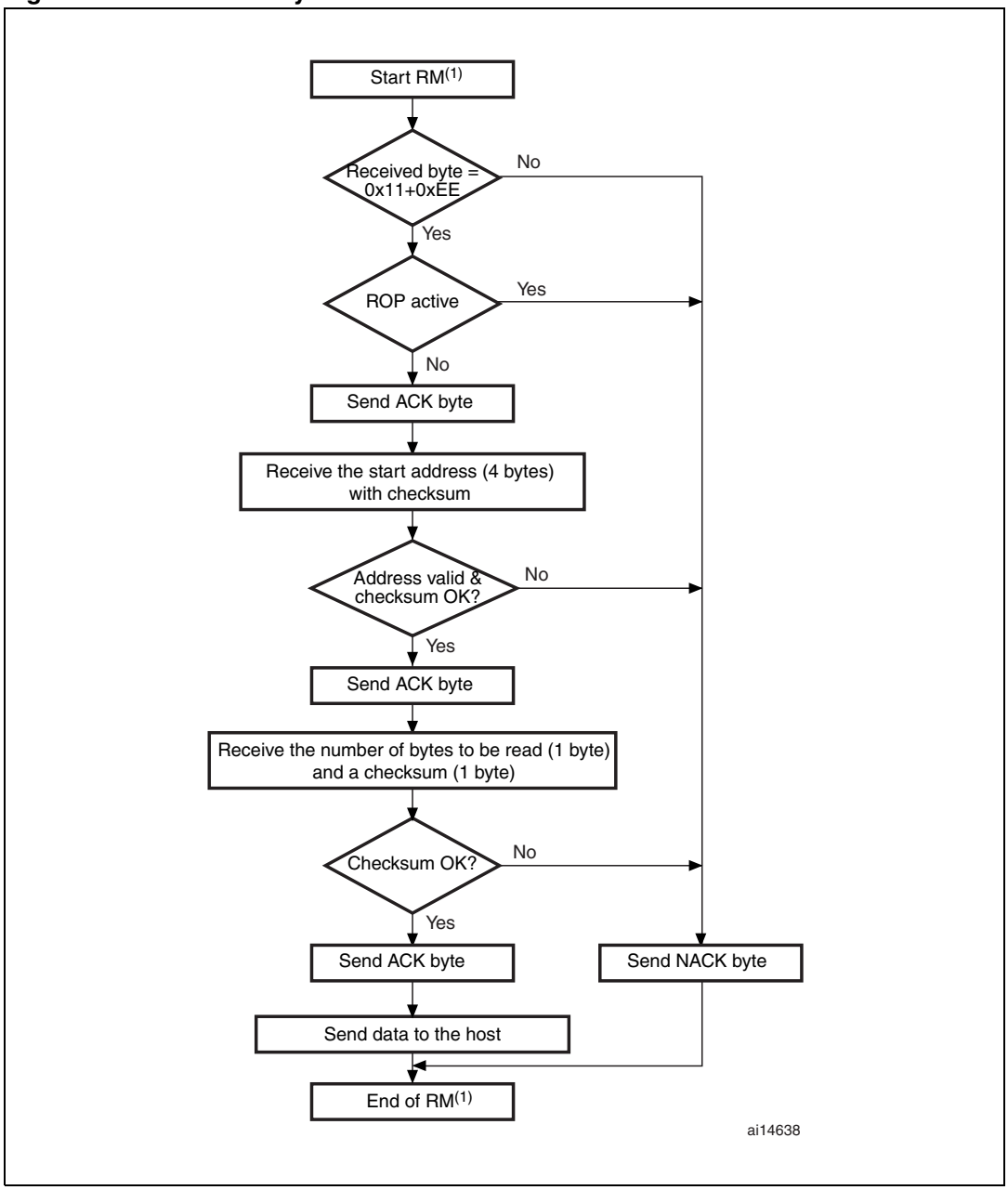
The Read Memory command is used to read data from any memory address in RAM, Flash memory and information block (System memory or option byte areas).
When the bootloader receives the Read Memory command, it transmits the ACK byte to the user. After the transmission of the ACK byte, the bootloader waits for an address (4 bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte, then it checks the received address. If the address is valid and the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command.
When the address is valid and the checksum is correct, the bootloader waits for the number of bytes to be transmitted (N bytes) and for its complemented byte (checksum). If the checksum is correct it then transmits the needed data ((N + 1) bytes) to the user, starting from the received address. If the checksum is not correct, it sends a NACK before aborting the command.

Figure 8. Read Memory command: host side



1. RM = Read Memory.

Figure 9. Read Memory command: device side



1. RM = Read Memory.

The host sends the bytes to the STM32F10xxx as follows:

Bytes 1-2: 0x11+0xEE

Wait for ACK

Bytes 3 to 6: start address

- byte 3: MSB
- byte 6: LSB

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

Wait for ACK

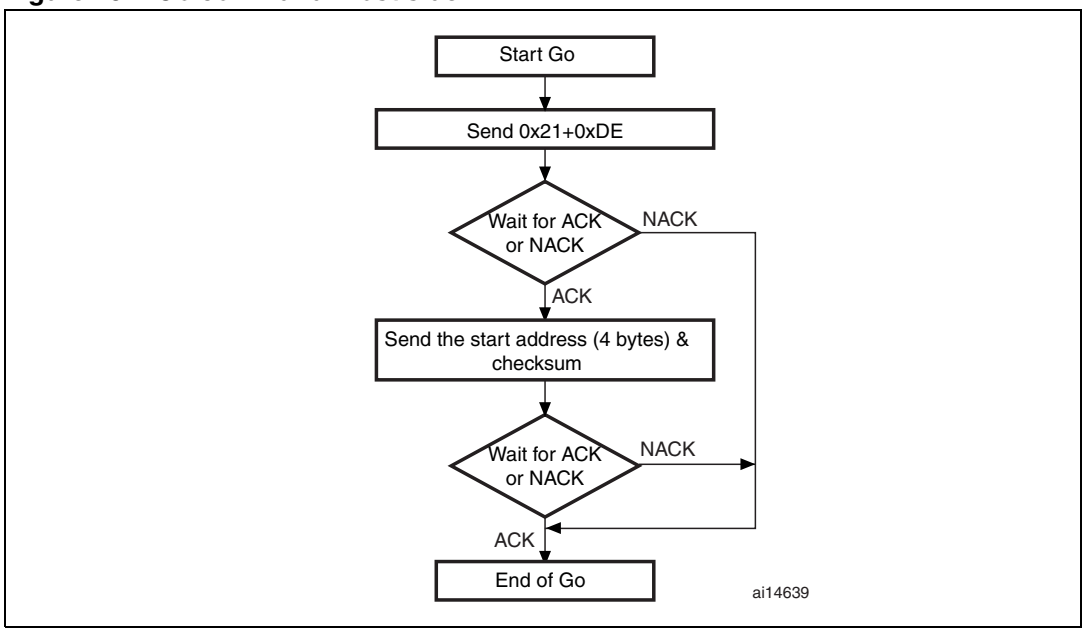
Byte 8: The number of bytes to be read – 1 (0 < N ≤255);

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

2.5 Go command

The Go command is used to execute the downloaded code or any other code by branching to an address specified by the user. When the bootloader receives the Go command, it transmits the ACK byte to the user. After the transmission of the ACK byte, the bootloader waits for an address (4 bytes, byte 1 is the address MSB and byte 4 is LSB) and a checksum byte, then it checks the received address. If the address is valid and the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command.
When the address is valid and the checksum is correct, the program counter of the CPU automatically jumps to the address.

Figure 10. Go command: host side

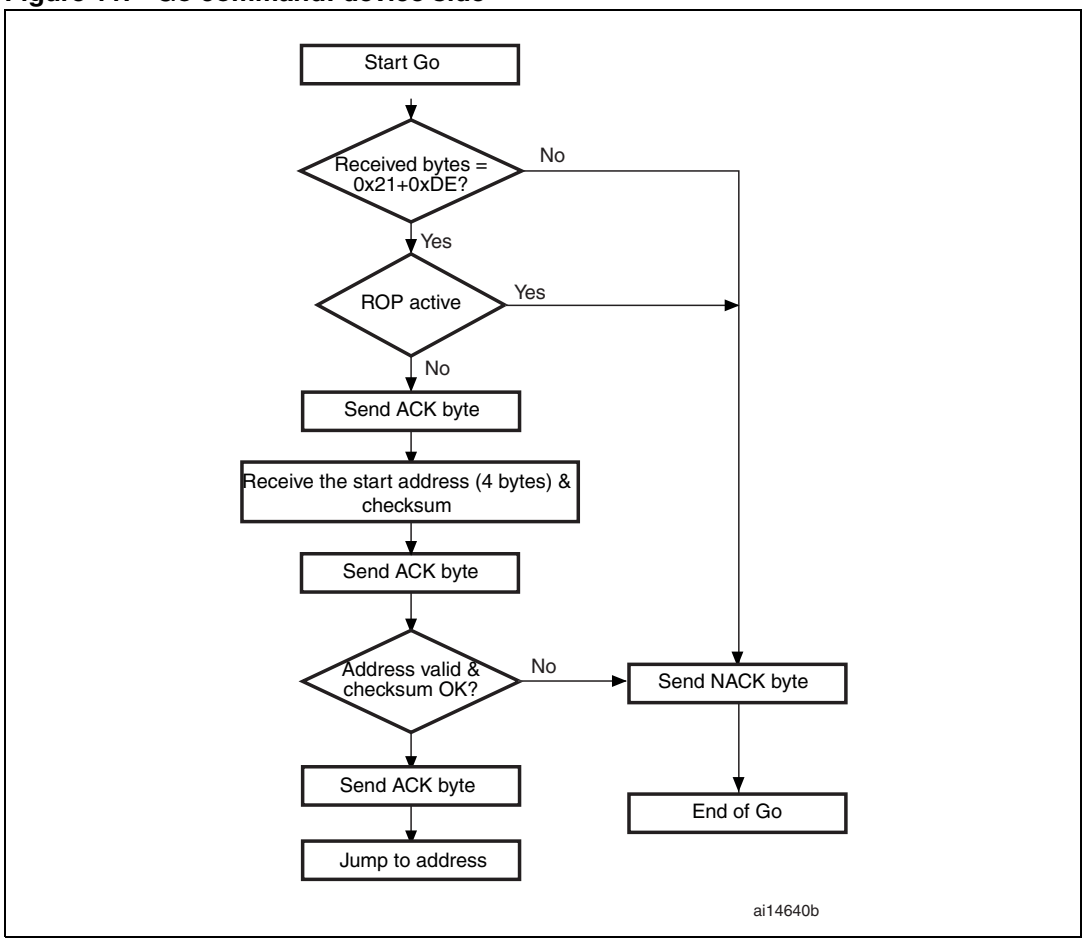


Note: Valid addresses are RAM (starting from 0x2000 0000 to the end of the RAM) or Flash memory (starting from 0x800 0000 to the end of the Flash memory) addresses. All other addresses are considered not valid and will be NACKed by the device.

When an application is loaded into RAM and then a jump is made to it, the program must be configured to run with an offset of at least 0x200 to avoid overlapping with the first 0x200 RAM memory used by the bootloader firmware.

The Go command must be used after loading an application into RAM or user Flash memory. It will initialize the main stack pointer and jump to the loaded code.

Figure 11. Go command: device side



The host sends the bytes as follow to the STM32F10xxx:

Byte 1: 0x21

Byte 2: 0xDE

Wait for ACK

Byte 3 to Byte 6: start address

byte3: MSB

byte6: LSB

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

2.6 Write Memory command

The Write Memory command is used to write data to any memory address of RAM, Flash memory, or Option byte area. Refer to the STM32F101xx and STM32F103xx Flash programming manual. When the bootloader receives the Write Memory command, it transmits the ACK byte to the user. After the transmission of the ACK byte, the bootloader waits for an address (4 bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte, it then checks the received address. For the Option byte area, the start address must be 0x1FFFF800 to avoid writing inopportunely in this area.
If the received address is valid and the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command. When the address is valid and the checksum is correct, the bootloader:

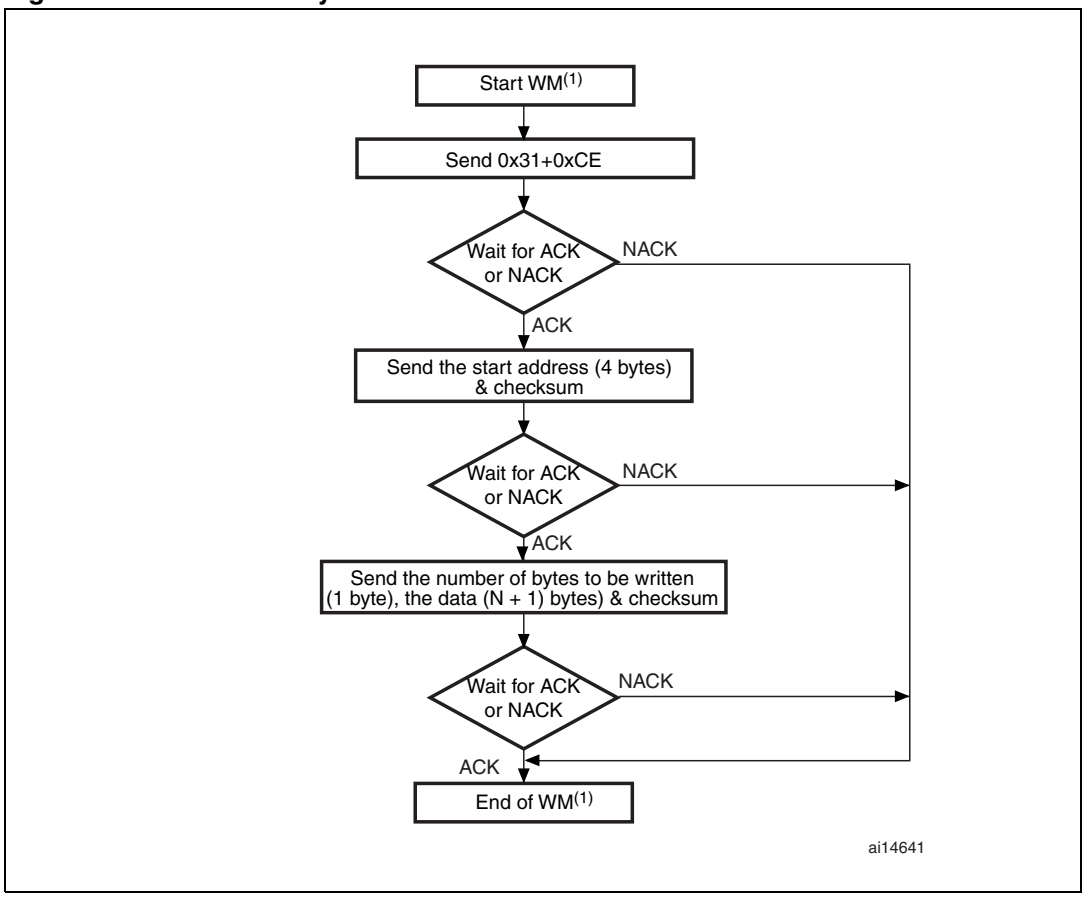
- gets a byte, N, which contains the number of data bytes to be received
- receives the user data ((N + 1) bytes) and the checksum (XOR of N and of all data bytes)
- programs the user data to memory starting from the received address
- at the end of the command, if the write operation was successful,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

The maximum length of the block to be written for the STM32F10xxx is 256 bytes.

If the Write Memory command is issued to the Option byte area, then at the end of the command the BL generates a system Reset to take into account the new configuration of the option byte.

Note: When writting to the RAM, care must be taken to avoid overlapping with the first 512 bytes (0x200) in RAM because they are used by the bootloader firmware.

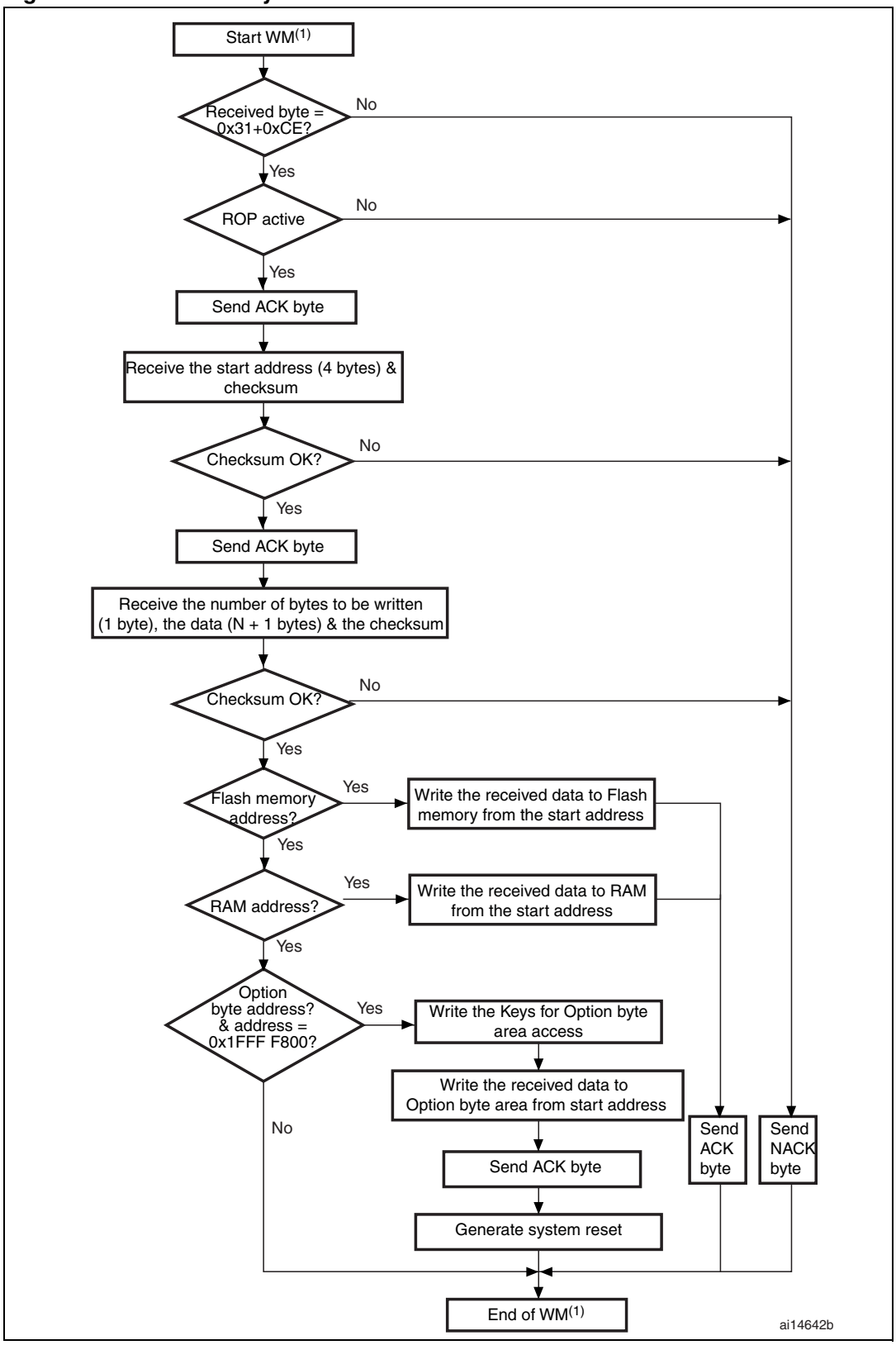
Figure 12. Write Memory command: host side



1. WM = Write Memory.

Note: If the start address is invalid, the command is NACKed by the device.

Figure 13. Write Memory command: device side



1. WM = Write Memory.

The host sends the bytes to the STM32F10xxx as follows:

Byte 1: 0x31

Byte 2: 0xCE

Wait for ACK

Byte 3 to byte 6: start address

byte 3: MSB

byte 6: LSB

Byte 7: Checksum: XOR (Byte3, Byte4, Byte5, Byte6)

Wait for ACK

Byte 8: Number of bytes to be received (0 < N ≤255)

N +1 data bytes: (Max 256 bytes)

Checksum byte: XOR (N, N+1 data bytes)

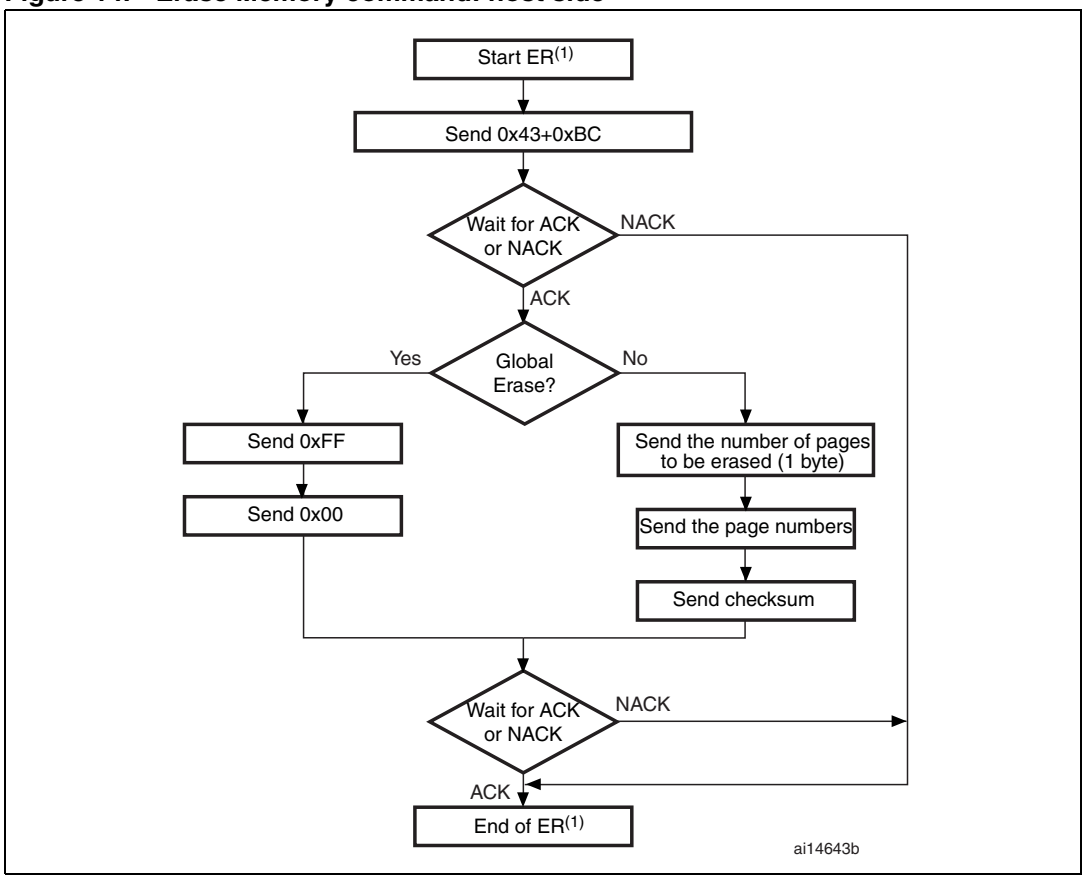
2.7 Erase Memory command

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

Erase Memory command specifications:

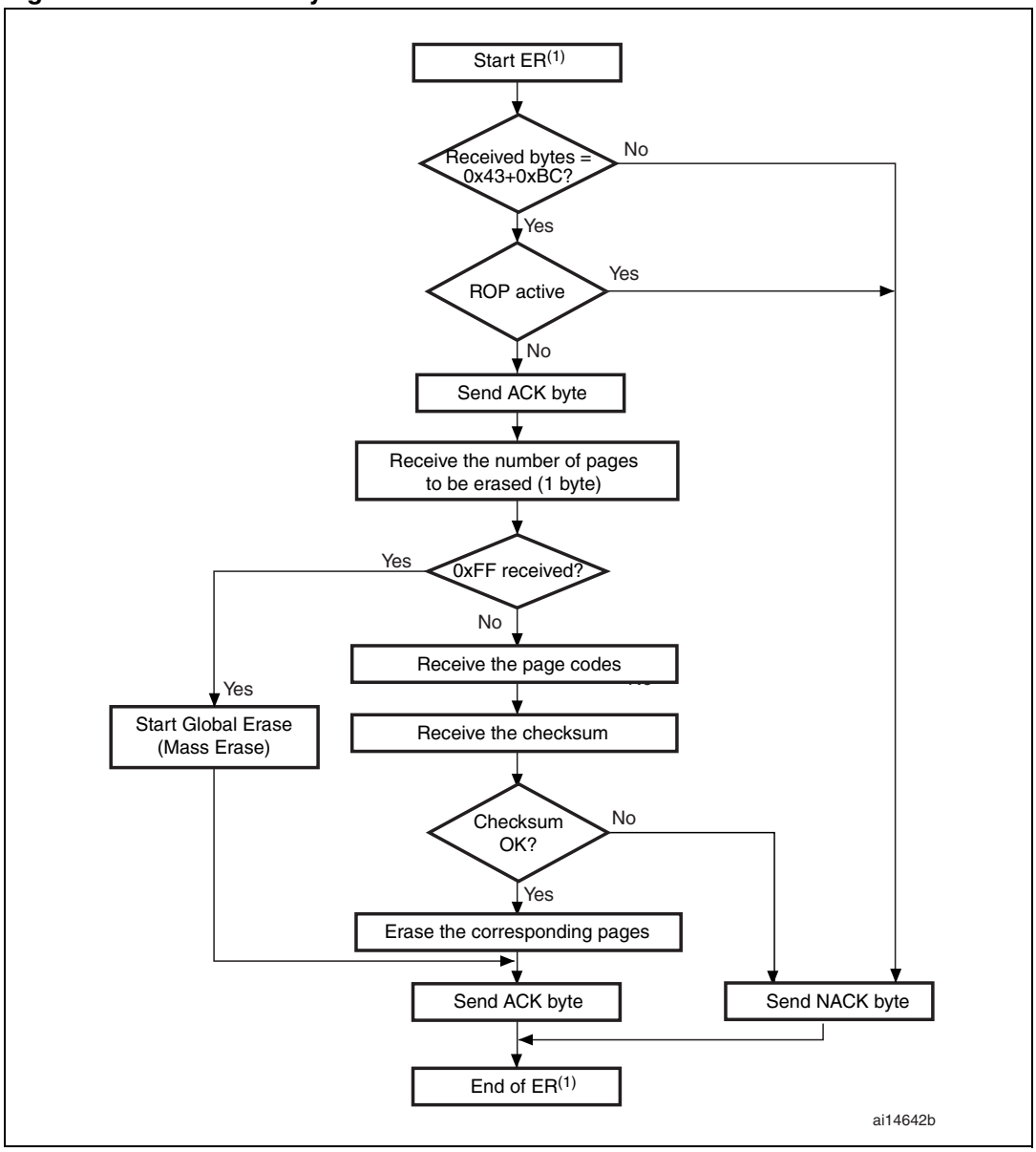
1. the bootloader receives one byte that contains N, the number of pages to be erased – 1.
   N = 255 is reserved for global erase requests. For 0 ≤N ≤254, N + 1 pages are erased.
2. the bootloader receives (N + 1) bytes, each byte containing a page number

Figure 14. Erase Memory command: host side



1. ER = Erase Memory.

Figure 15. Erase Memory command: device side



1. ER = Erase Memory.

The host sends the bytes to the STM32F10xxx as follows:

Byte 1: 0x43

Byte 2: 0xBC

Wait for ACK

Byte 3: 0xFF or number of pages to be erased (0 ≤N ≤maximum number of pages)

Byte 0x00 or (N + 1 bytes (page numbers) and then the checksum for byte 3 and the following bytes)

2.8 Write Protect command

The Write Protect command is used to enable the write protection for some or all Flash memory sectors. When the bootloader receives the Write Protect command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader waits for the number of bytes to be received (sectors to be protected) and then receives the Flash memory sector codes from the user.
If the write protection was enabled successfully, the bootloader transmits the ACK byte, otherwise it transmits a NACK byte to the user and the command is aborted.

At the end of the Write Protect command, the BL generates a system Reset to take into account the new configuration of the option byte.

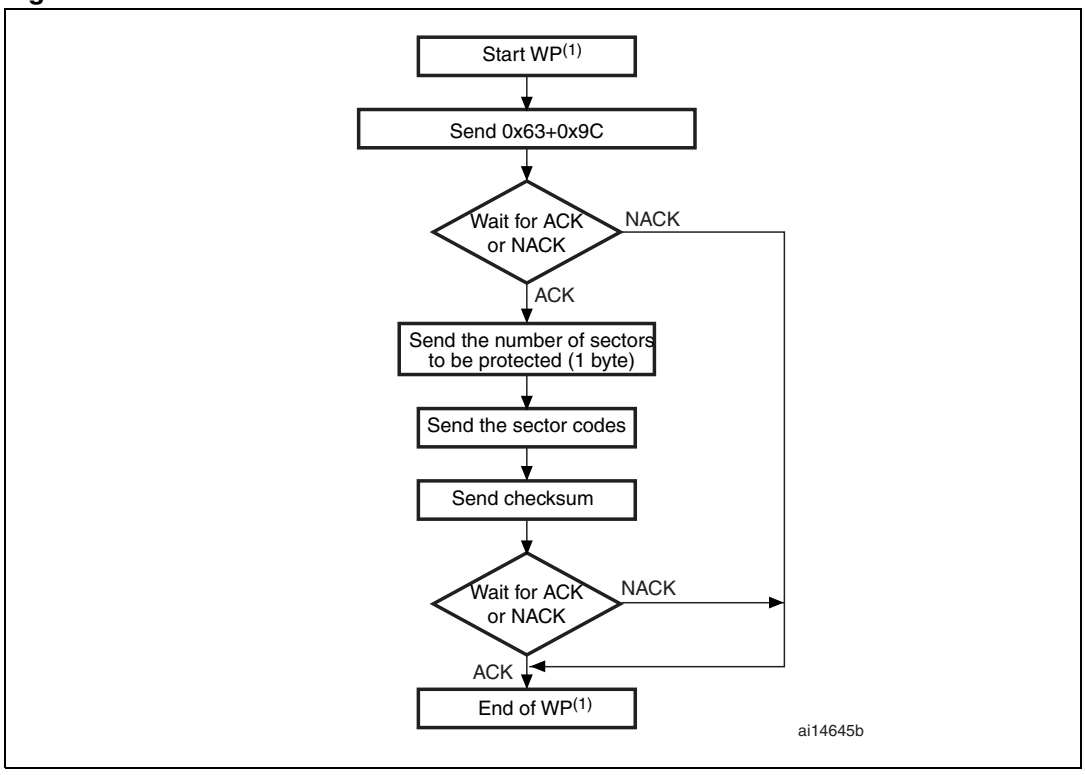
Note: On the STM32F10xxx, the sector size is 4 Kbytes (4 pages when the page size is 1 Kbyte and 2 pages when the page is 2 Kbytes) for the Write Protect command.

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

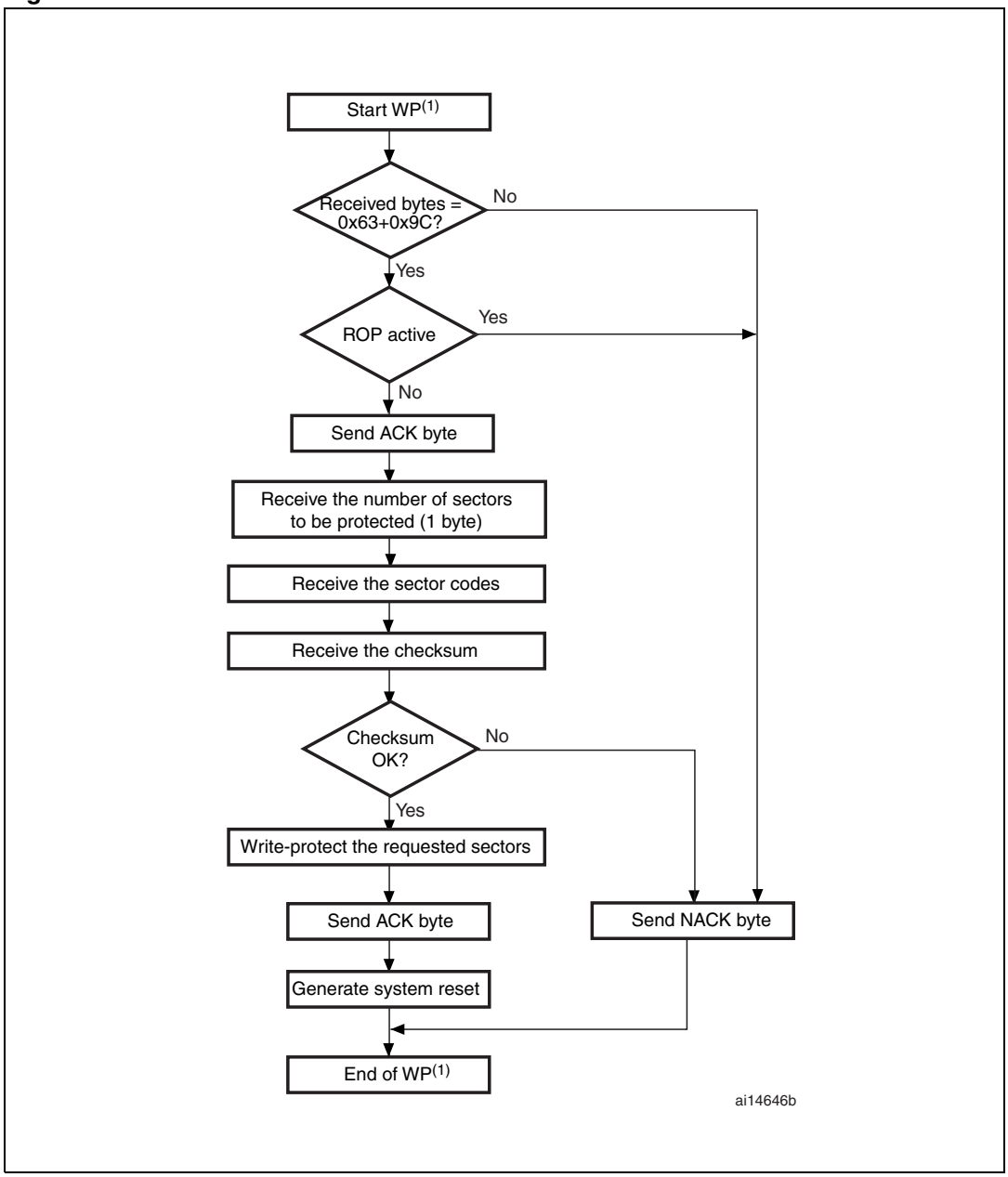
Note: If a second Write Protect command is executed, the Flash memory sectors that had been protected by the first command become unprotected and only the sectors passed within the second Write Protect command become protected.

Figure 16. Write Protect command: host side



1. WP = Write Protect.

Figure 17. Write Protect command: device side



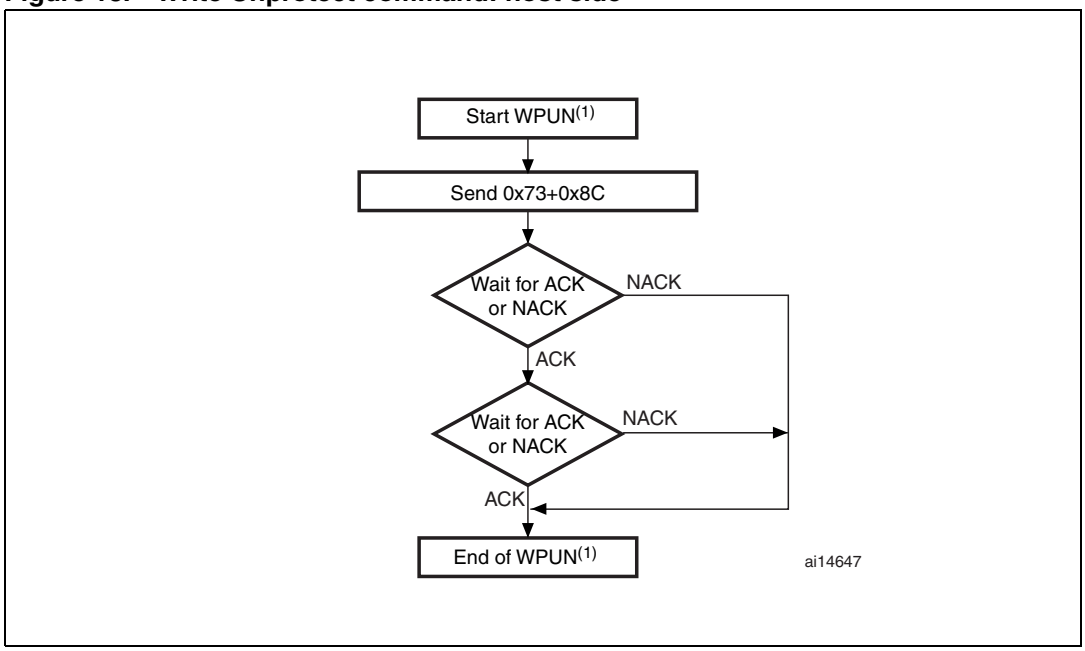
1. WP = Write Protect.

2.9 Write Unprotect command

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

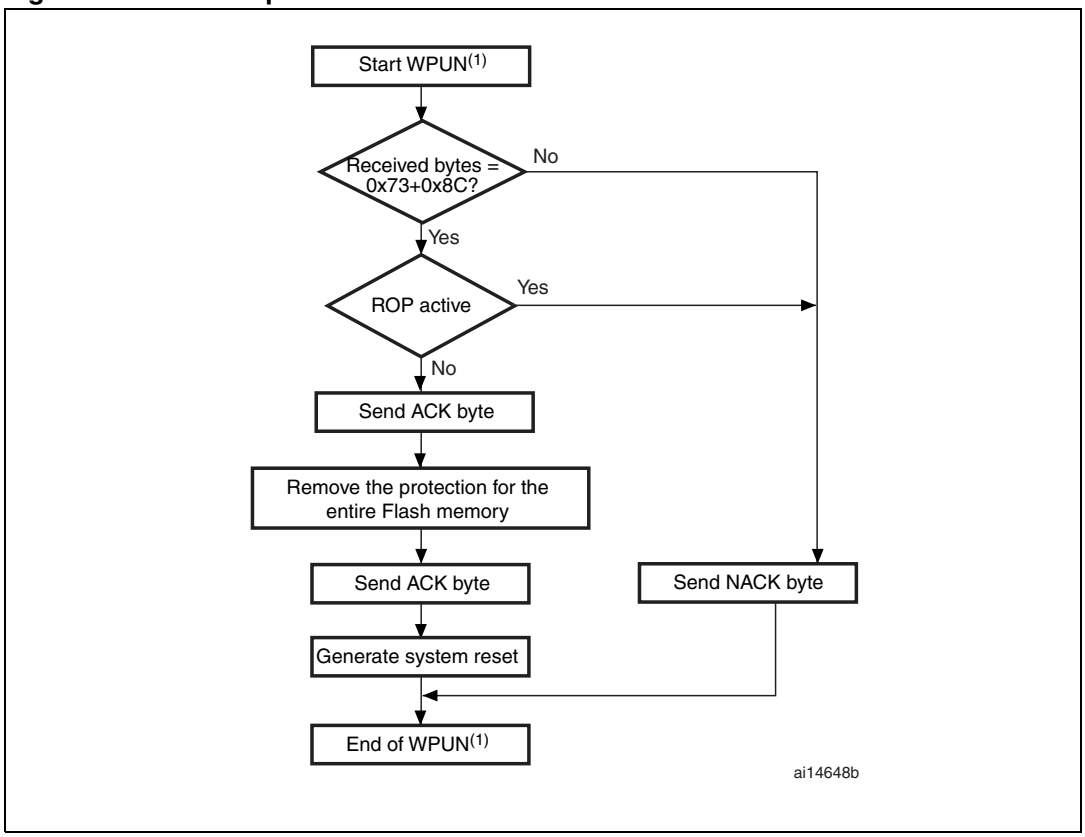
At the end of the Write Unprotect command, the BL generate a system Reset to take into account the new configuration of the option byte.

Figure 18. Write Unprotect command: host side



1. WPUN = Write Unprotect.

Figure 19. Write Unprotect command: device side



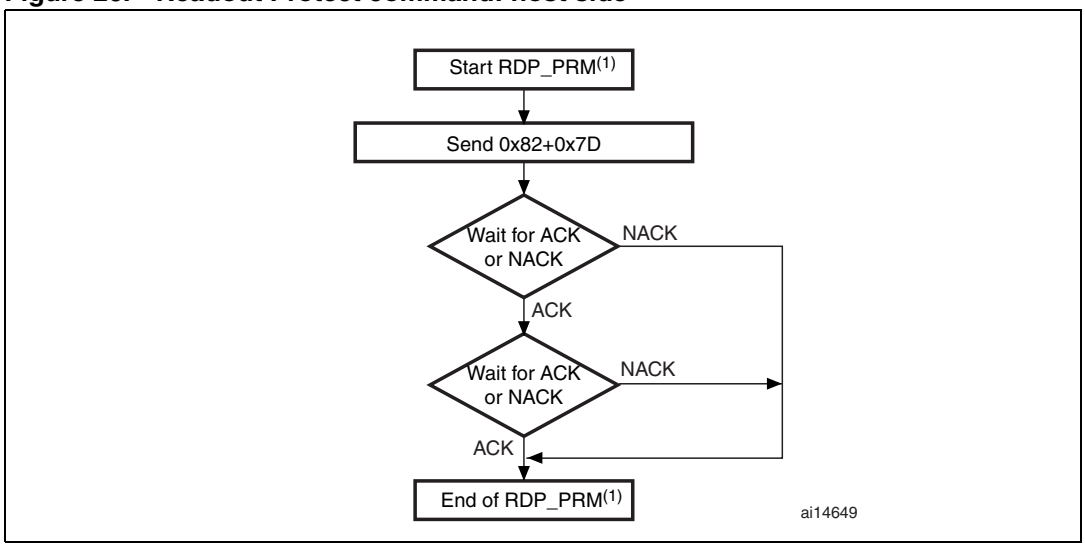
1. WPUN = Write Unprotect.

2.10 Readout Protect command

The Readout Protect command is used to enable the Flash memory read protection. When the bootloader receives the Readout Protect command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader enables the read protection for the Flash memory. If the protection operation was successful the bootloader transmits the ACK byte, otherwise it transmits a NACK byte to the user.

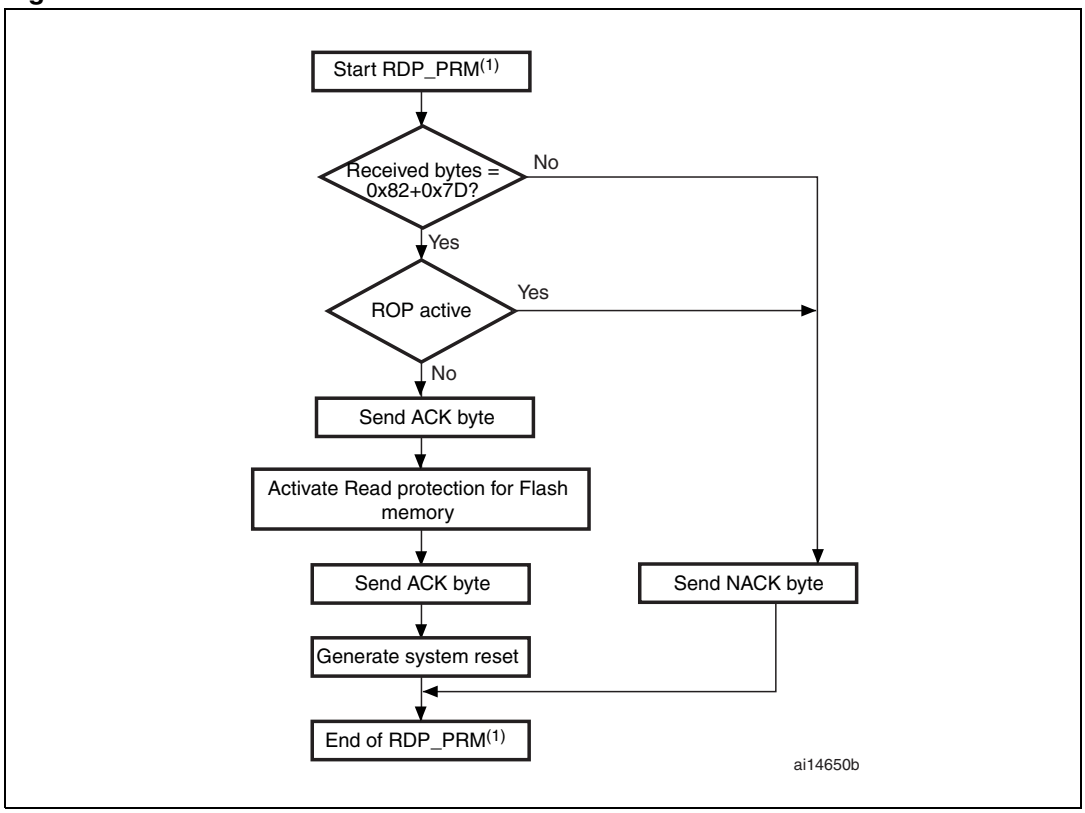
At the end of the Readout Protect command, the BL generates a system Reset to take into account the new configuration of the option byte.

Figure 20. Readout Protect command: host side



1. RDP_PRM = Readout Protect.

Figure 21. Readout Protect command: device side



1. RDP_PRM = Readout Protect.

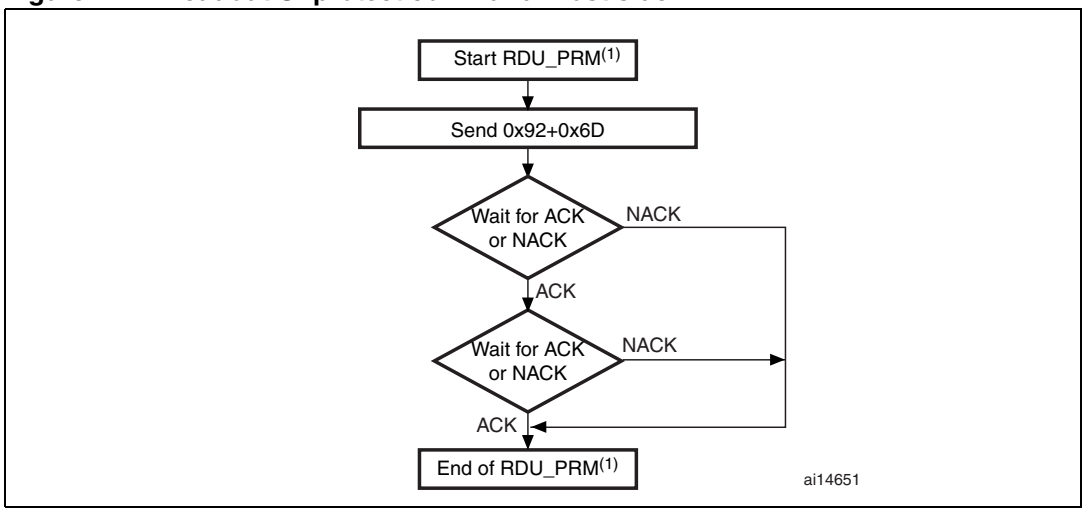
2.11 Readout Unprotect command

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

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

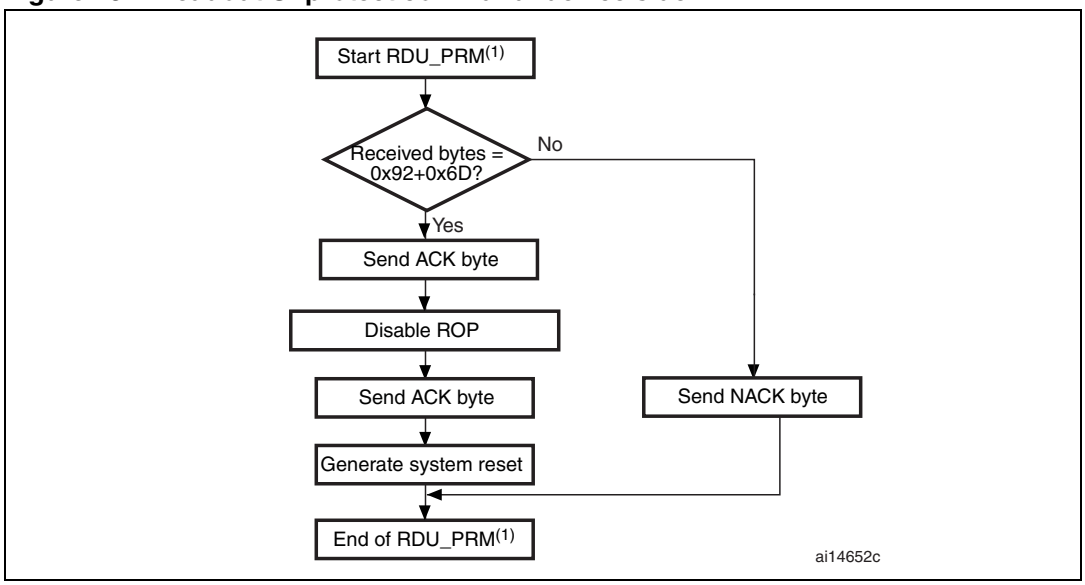
At the end of the Readout Unprotect command, the BL generates a system Reset to take into account the new configuration of the option byte.

Figure 22. Readout Unprotect command: host side



1. RDU_PRM = Readout Unprotect.

Figure 23. Readout Unprotect command: device side



1. RDU_PRM = Readout Unprotect.

3 Bootloader version evolution

Table 4 lists the bootloader versions.

Table 4. Bootloader versions

Bootloader version number	Description
V2.0	Initial bootloader version.
V2.1	– Update Go Command to initialize the main stack pointer – Update Go command to return NACK when jump address is in the Option byte area or System memory area – Update Get ID command to return the device ID on two bytes – Update the bootloader version to V2.1

Revision history

Table 5. Document revision history

Date	Revision	Changes
22-Oct-2007	1	Initial release.
22-Jan-2008	2	All STM32F10xxx in production (rev. B and rev. Z) include the bootloader described in this application note. Modified: *Section 1.2: Bootloader activation* and *Section 1.4: Bootloader code sequence*. Added: *Section 1.3: Hardware requirements*, *Section 1.5: Choosing the USART baud rate*, *Section 1.6: Using the bootloader* and *Section 1.6: Exiting System memory boot mode*. *Note 2* linked to Get, Get Version & Read Protection Status and Get ID commands in *Table 3: Bootloader commands*, *Note 3* added. Notion of “permanent” (Permanent Write Unprotect/Readout Protect/Unprotect) removed from document. Small text changes. Bootloader version upgraded to 2.0.
26-May-2008	3	Small text changes. RAM and System memory added to *Table 2: STM32F10xxx configuration in System memory boot mode*. *Section 1.6: Using the bootloader on page 8* removed. Erase modified, *Note 3* modified and *Note 1* added in *Table 3: Bootloader commands on page 9*. *Byte 3: on page 11* modified. *Byte 2: on page 13* modified. *Byte 2:*, *Bytes 3-4:* and *Byte 5: on page 15* modified, *Note 3* modified. *Byte 8: on page 18* modified. Notes added to *Section 2.5: Go command on page 18*. *Figure 11: Go command: device side on page 19* modified. Note added in *Section 2.6: Write Memory command on page 20*. *Byte 8: on page 23* modified. *Figure 14: Erase Memory command: host side* and *Figure 15: Erase Memory command: device side* modified. *Byte 3: on page 25* modified. *Table 3: Bootloader commands on page 9*. Note modified and note added in *Section 2.8: Write Protect command on page 26*. *Figure 16: Write Protect command: host side*, *Figure 17: Write Protect command: device side*, *Figure 19: Write Unprotect command: device side*, *Figure 21: Readout Protect command: device side* and *Figure 23: Readout Unprotect command: device side* modified.