AN3155
Application note

USART protocol used in the STM32™ bootloader

Introduction

This application note describes the USART protocol used in the STM32 microcontroller bootloader. It details each supported command. For more information about the USART hardware resources and requirements for your device bootloader, please refer to the “STM32 system memory boot mode” application note (AN2606).

Related documents

Available from www.st.com:

AN2606 “STM32 system memory boot mode”

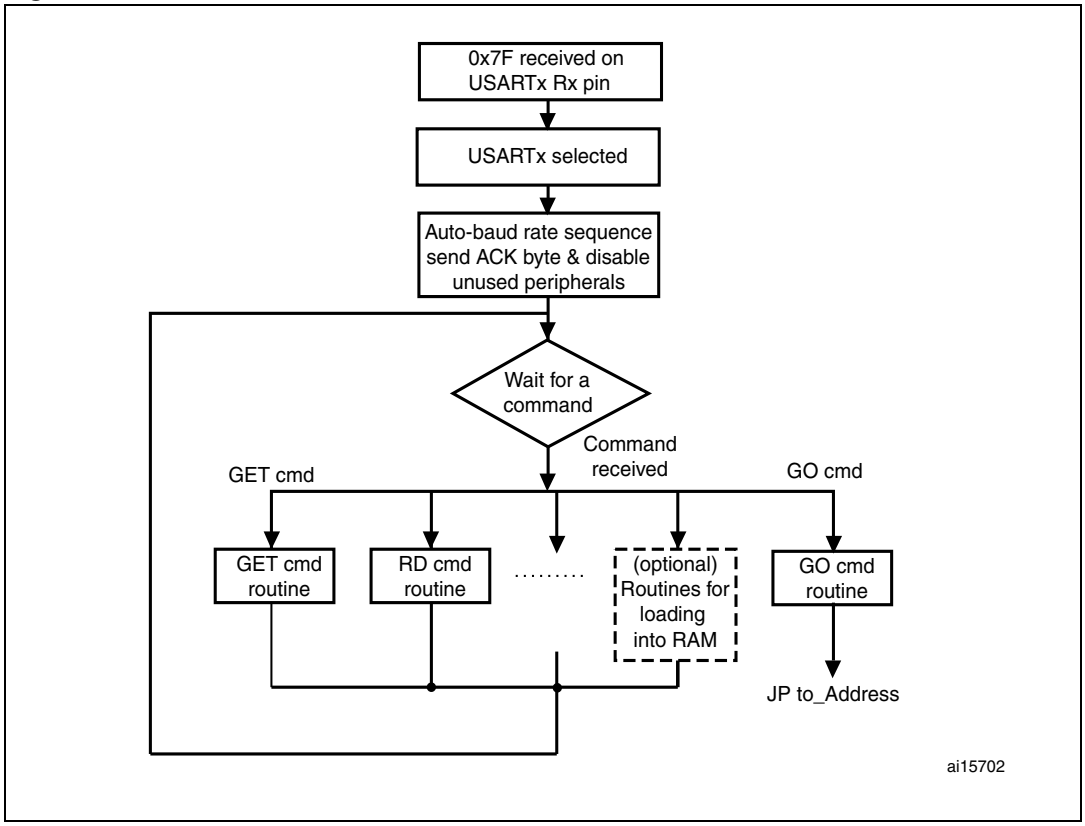
Contents

List of tables

List of figures

1 USART bootloader code sequence

Figure 1. Bootloader for STM32 with USART



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

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

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

2 Choosing the USARTx baud rate

The calculation of the serial baud rate for USARTx, 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 microcontroller, the maximum deviation between the internal initialized baud rate for USARTx 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 microcontroller baud rate can be calculated using the formula below:

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

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.

2.1 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, USARTx would not be correctly initialized.

2.2 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.

3 Bootloader command set

The supported commands are listed in *Table 1* below. Each command is further described in this section.

Table 1. USART 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 application
Go	0x21	Jumps to user application code located in the internal Flash memory or in SRAM
Write Memory	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the application
Erase(3)	0x43	Erases from one to all the Flash memory pages
Extended Erase(3)	0x44	Erases from one to all the Flash memory pages using two byte addressing mode (available only for v3.0 usart bootloader versions and above).
Write Protect(4)	0x63	Enables the write protection for some sectors
Write Unprotect(4)	0x73	Disables the write protection for all Flash memory sectors
Readout Protect	0x82	Enables the read protection
Readout Unprotect(2)	0x92	Disables the read protection

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. Erase (x043) and Extended Erase (0x44) are exclusive. A device may support either the Erase command or the Extended Erase command but not both.
4. See *Section 3.1: Device-dependent bootloader parameters*.

Communication safety

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

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

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

- ACK = 0x79
- NACK = 0x1F

3.1 Device-dependent bootloader parameters

While the USART bootloader protocol's command set and sequences are the same for all the STM32 devices, some parameters are device-dependent. For a few commands, the value of some parameters may depend on the device used. The concerned parameters are listed below:

- PID (product ID), which changes with the device
- Valid memory addresses (RAM, Flash memory, system memory, option byte areas) accepted by the bootloader when the Read Memory, Go and Write Memory commands are executed.
- Size of the Flash memory sector used when executing the Write Protect command.

For more details about the value of these parameters for the device you are using, please refer to the "Device-dependent bootloader parameters" section in the "STM32 system memory boot mode" application note (AN2606).

3.2 Get command

The Get command allows 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 host, as described in *Figure 2*.

Figure 2. Get command: host side

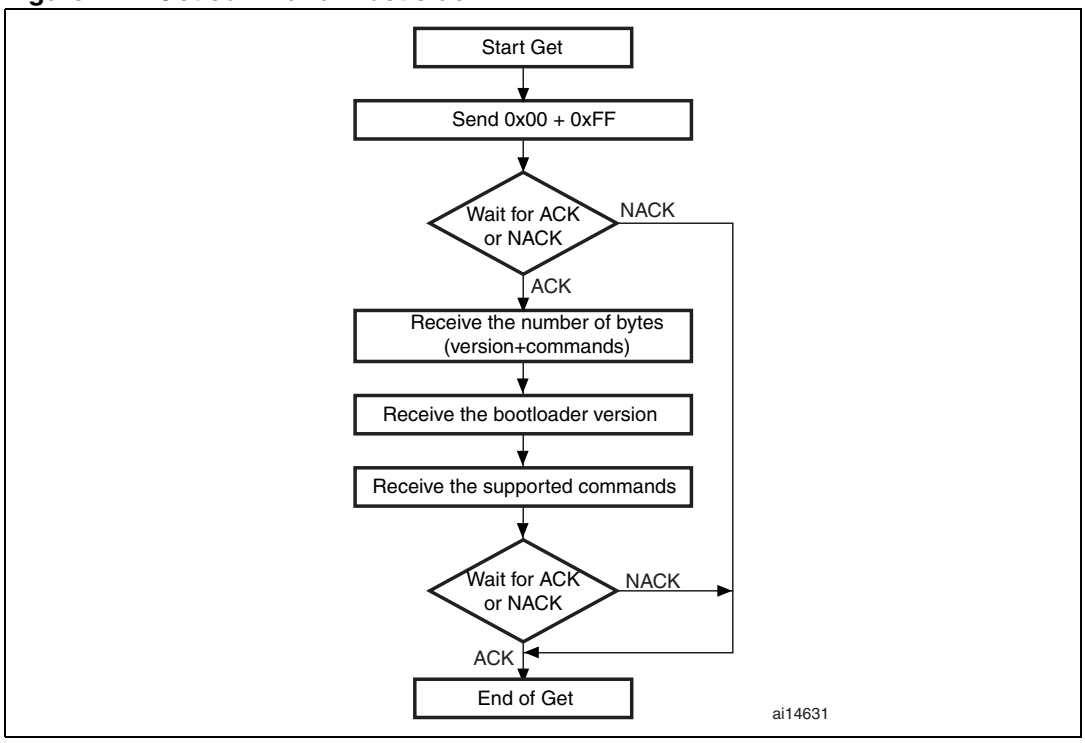
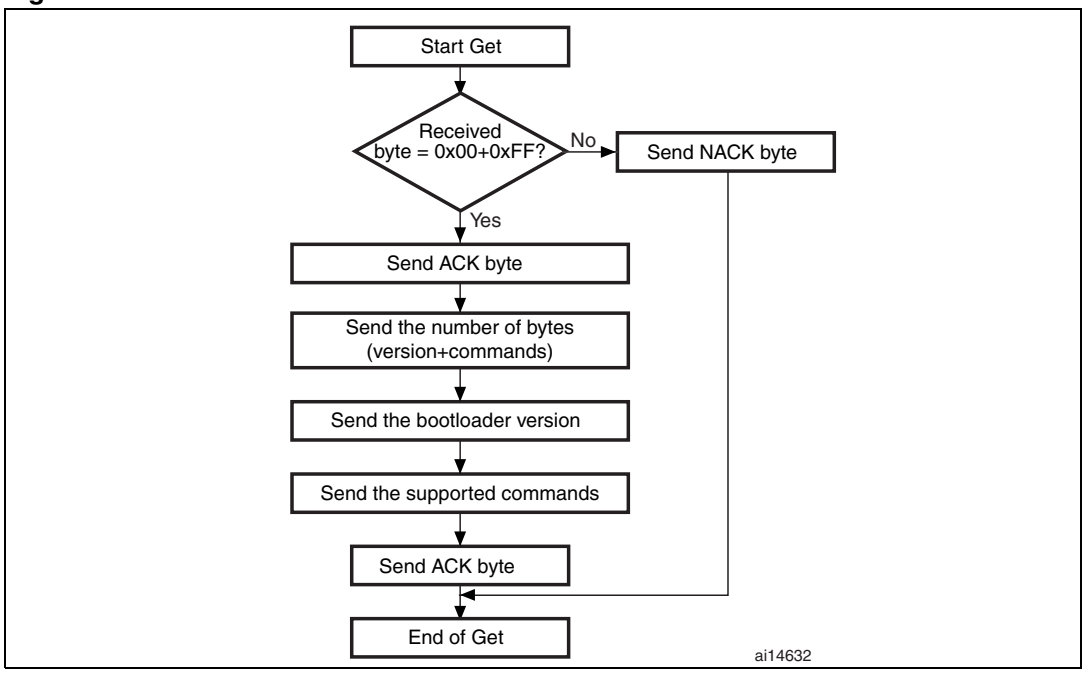


Figure 3. Get command: device side



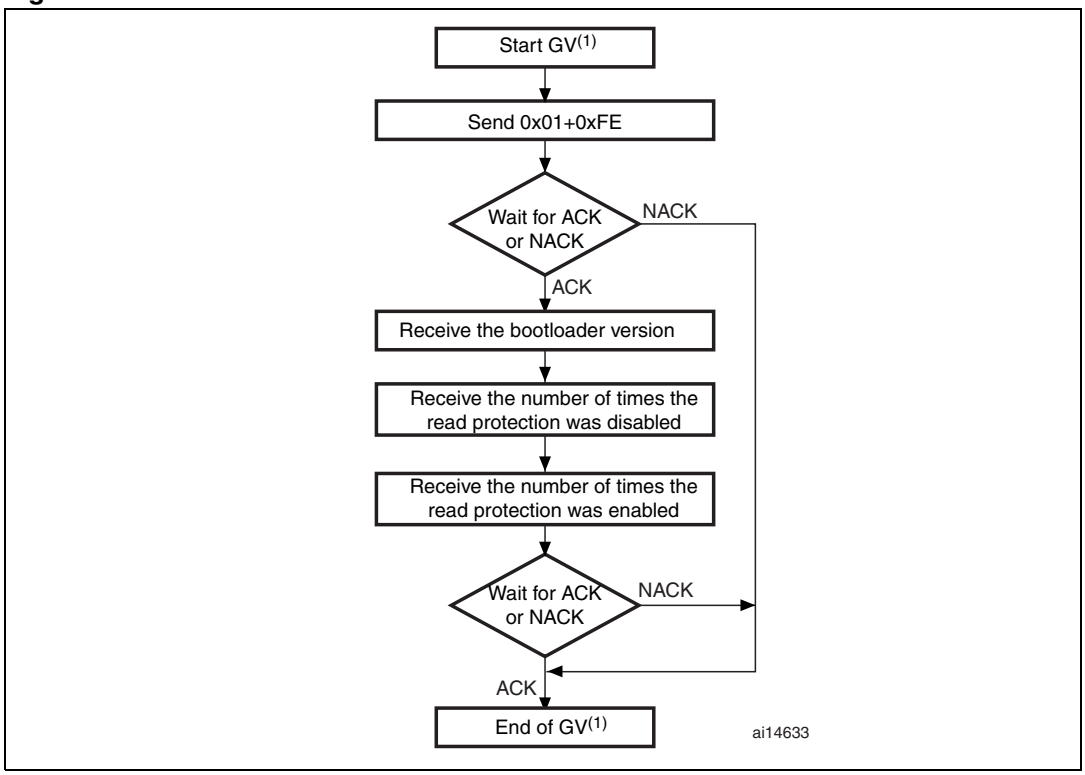
The STM32 sends the bytes as follows:

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

3.3 Get Version & Read Protection Status command

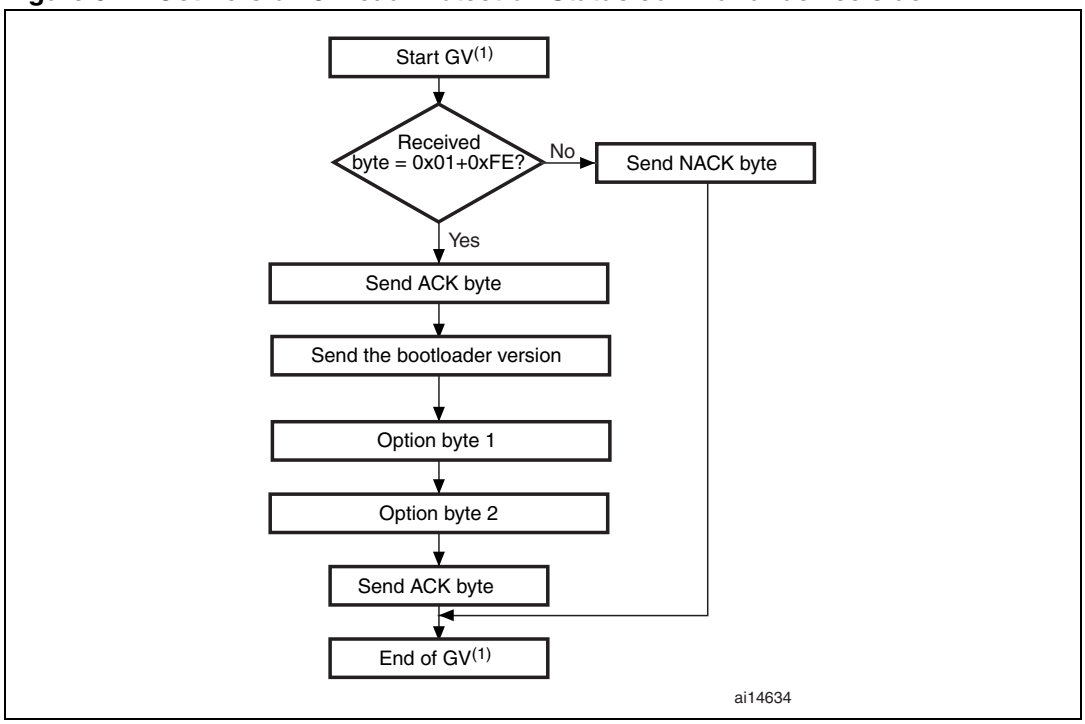
The Get Version & Read Protection Status command is used to get the bootloader version and the read protection status. When the bootloader receives the command, it transmits the information described below (version, read protection: number of times it was enabled and disabled) to the host.

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 STM32 sends the bytes as follows:

Byte 1: ACK

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

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

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

Byte 5: ACK

3.4 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 host.

The STM32 device sends the bytes as follows:

Byte 1: ACK

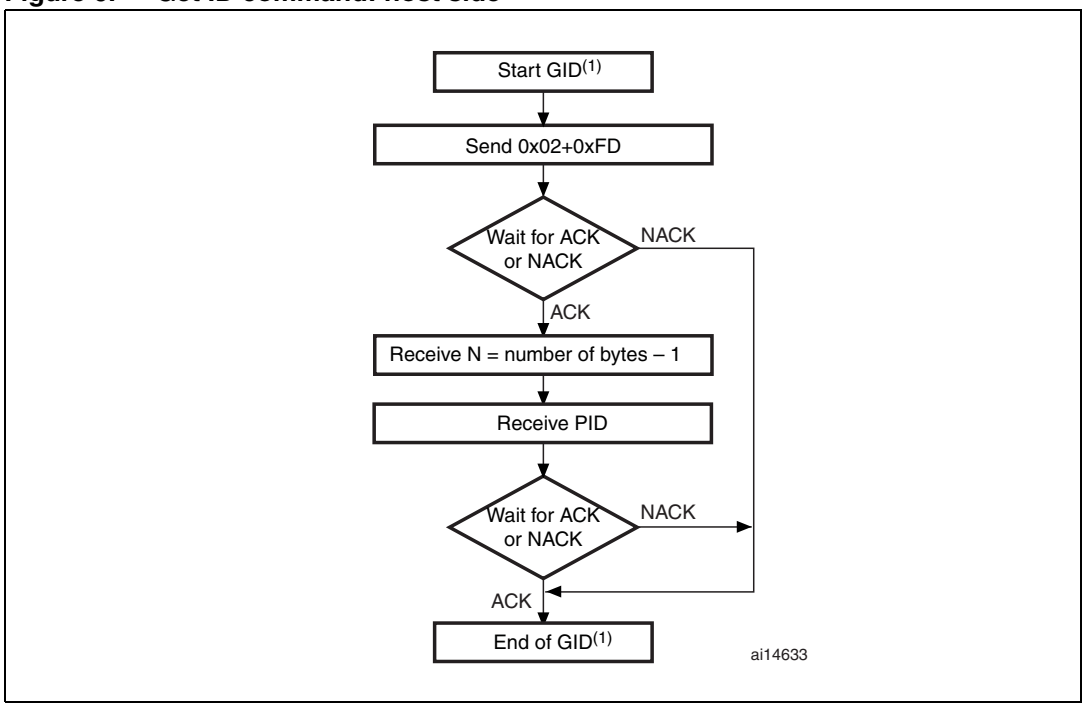
Byte 2: N = the number of bytes – 1 (N = 1 for STM32), except for current byte and ACKs.

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

Byte 5: ACK

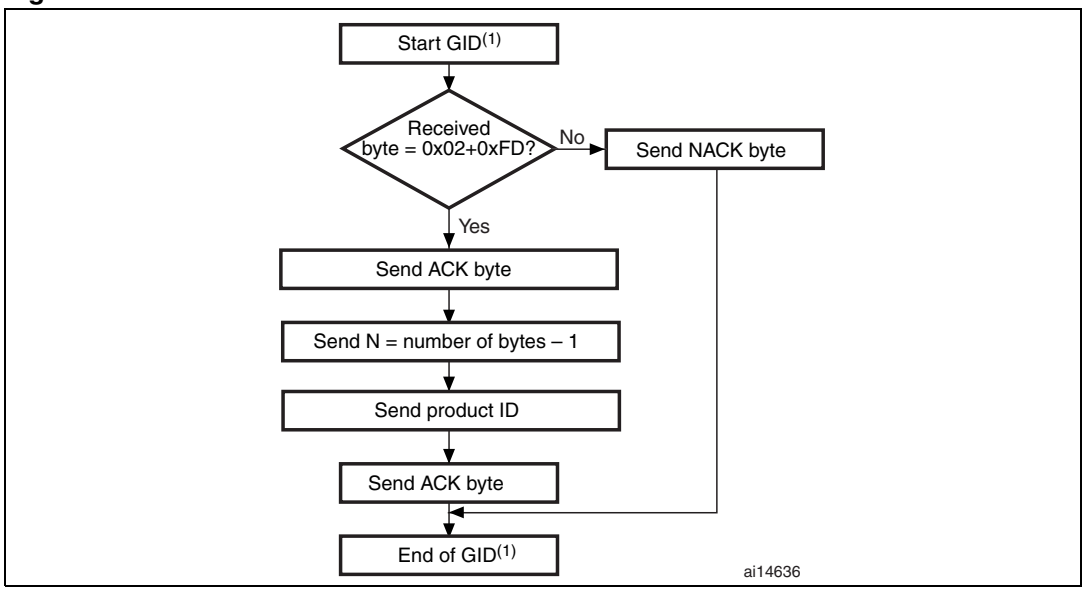
1. PID stands for product ID. Byte 1 is the MSB and byte 2, the LSB of the address. Refer to *Section 3.1: Device-dependent bootloader parameters* for more details about the PID of the device you are using.

Figure 6. Get ID command: host side



1. GID = Get ID.

Figure 7. Get ID command: device side



1. GID = Get ID.

3.5 Read Memory command

The Read Memory command is used to read data from any valid memory address (see note) in RAM, Flash memory and the information block (System memory or option byte areas).

Note: Refer to Section 3.1: Device-dependent bootloader parameters for more details about the valid memory addresses for the device you are using.

When the bootloader receives the Read Memory command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (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 – 1 (N bytes) and for its complemented byte (checksum). If the checksum is correct it then transmits the needed data ((N + 1) bytes) to the application, starting from the received address. If the checksum is not correct, it sends a NACK before aborting the command.

The host sends bytes to the STM32 as follows:

Bytes 1-2: 0x11+0xEE

Wait for ACK

Bytes 3 to 6: start address

- byte 3: MSB
- byte 6: LSB

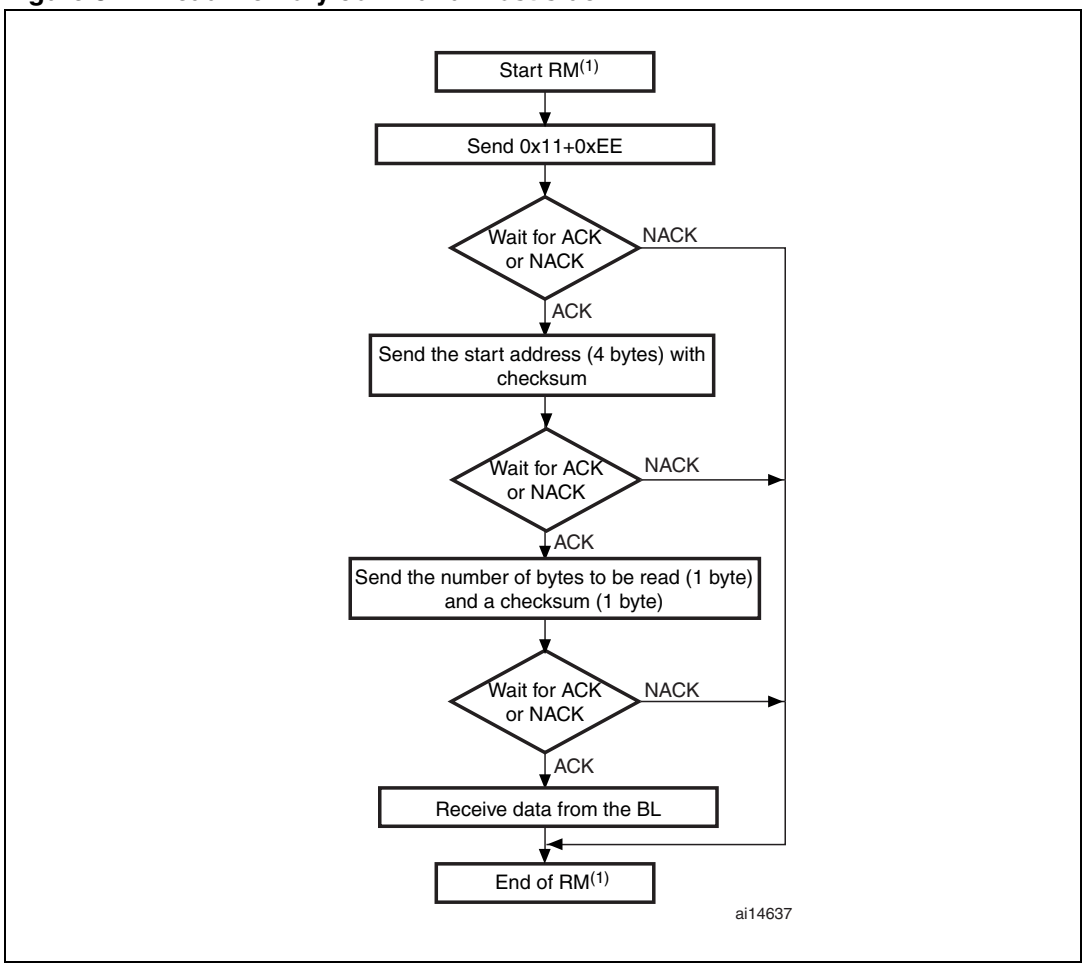
Byte 7: Checksum: XOR (byte 3, byte 4, byte 5, byte 6)

Wait for ACK

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

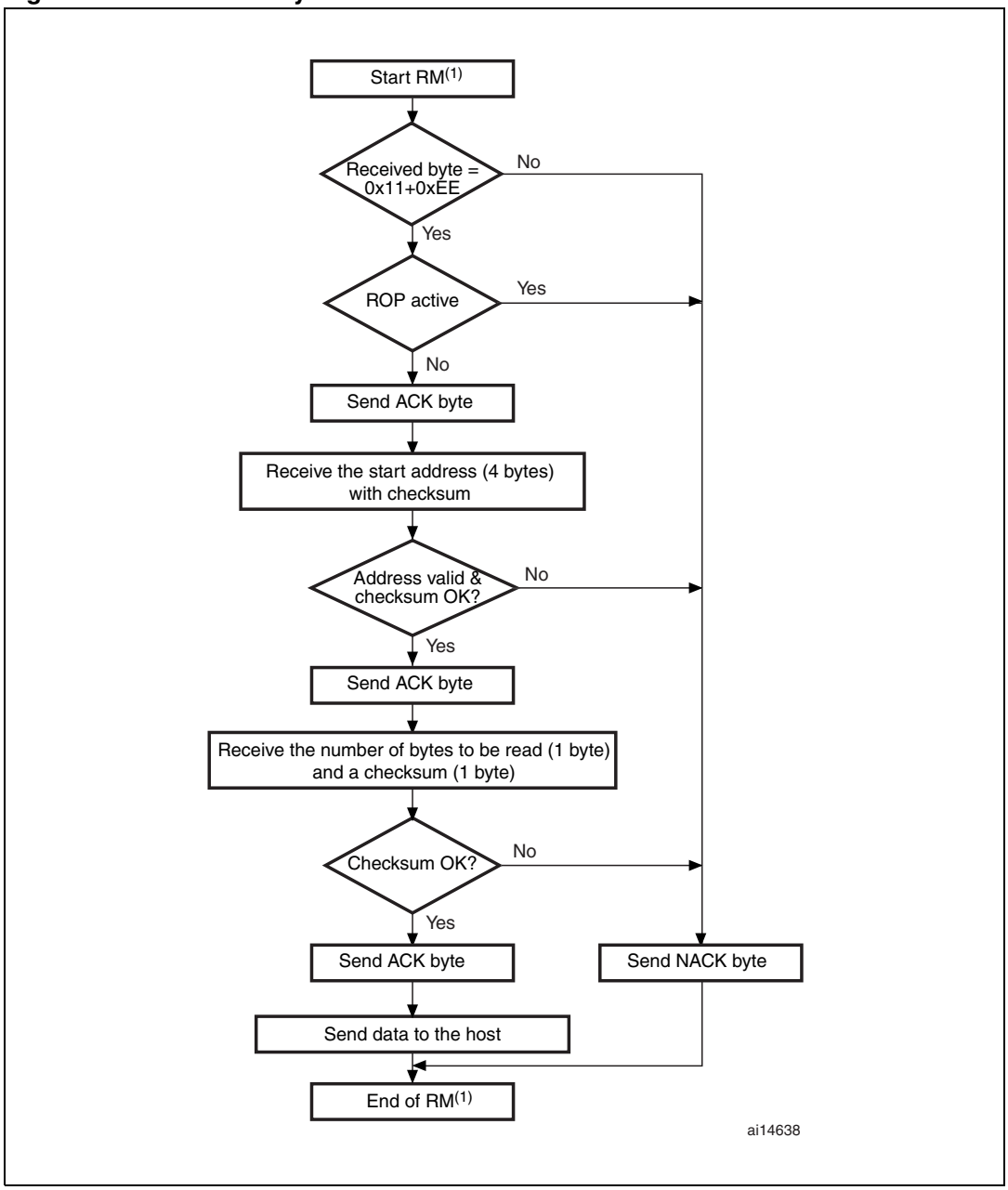
Byte 9: Checksum: XOR byte 8 (complement of byte 8)

Figure 8. Read Memory command: host side



1. RM = Read Memory.

Figure 9. Read Memory command: device side



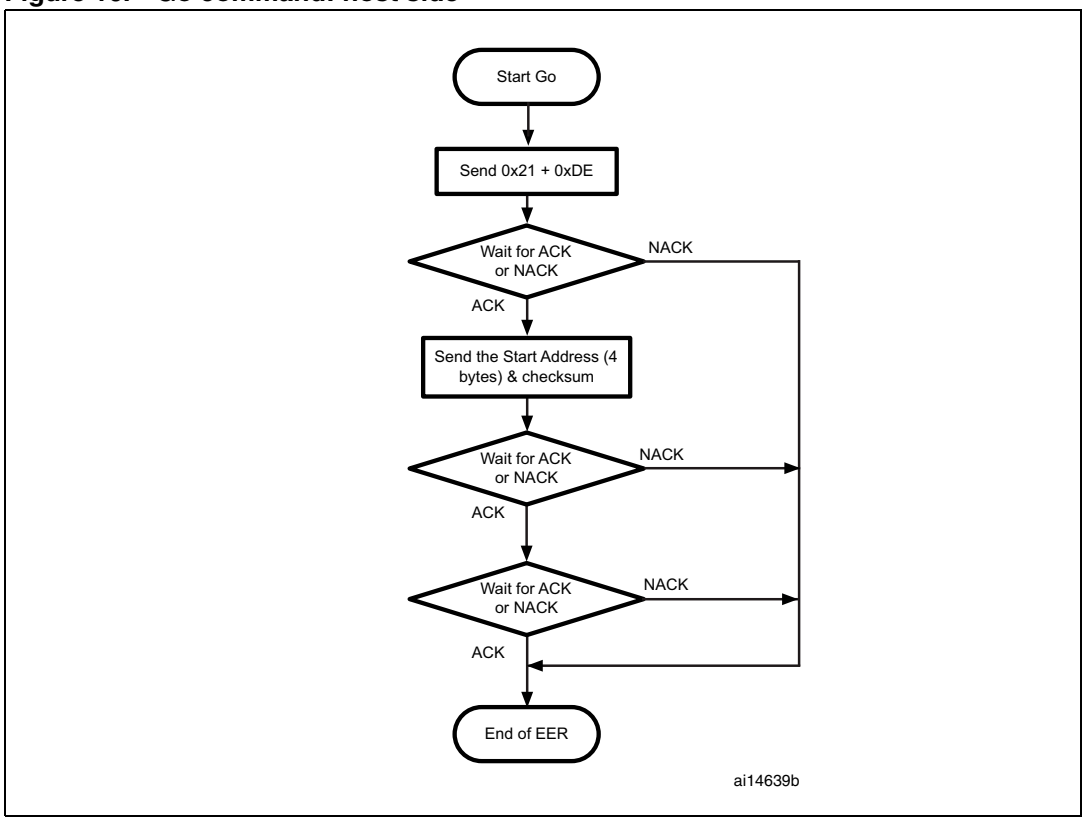
1. RM = Read Memory.

3.6 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 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 bootloader firmware performs the following:

- it initializes the registers of the peripherals used by the bootloader to their default reset values
- it initializes the user application's main stack pointer
- it 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 0x0800 0000, the bootloader will jump to the memory location programmed at address 0x0800 0004.
  In general, the host should send the base address where the application to jump to is programmed

Figure 10. Go command: host side

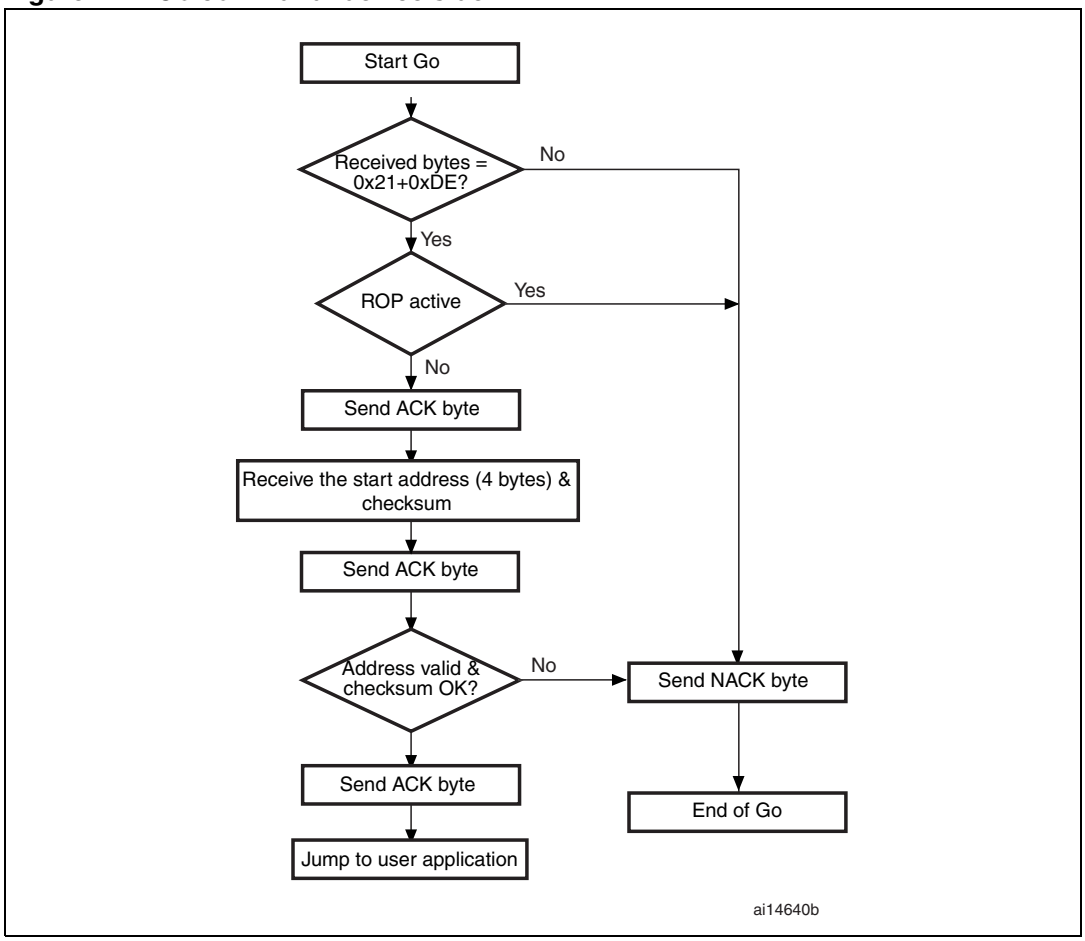


Note: 1 Valid addresses for the Go command are in RAM or Flash memory (refer to Section 3.1: Device-dependent bootloader parameters for more details about the valid memory addresses for the device you are using). All other addresses are considered not valid and are NACKed by the device.

2 When an application is loaded into RAM and then a jump is made to it, the program must be configured to run with an offset to avoid overlapping with the first RAM memory used by the bootloader firmware (refer to Section 3.1: Device-dependent bootloader parameters for more details about the RAM offset for the device you are using).

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

Figure 11. Go command: device side



The host sends bytes as follow to the STM32:

Byte 1: 0x21

Byte 2: 0xDE

Wait for ACK

Byte 3 to byte 6: start address

byte 3: MSB

byte 6: LSB

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

3.7 Write Memory command

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

When the bootloader receives the Write Memory command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (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 the base address of the Option byte area (see note) to avoid writing inopportunely in this area.

Note: 1 Write operations to Flash memory/SRAM must be word (32-bit) aligned and data should be in multiples of four bytes. If less data are written the remaining bytes should be filled by 0xFF.

2 Refer to Section 3.1: Device-dependent bootloader parameters for more details about the valid memory addresses for the device you are using.

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

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

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

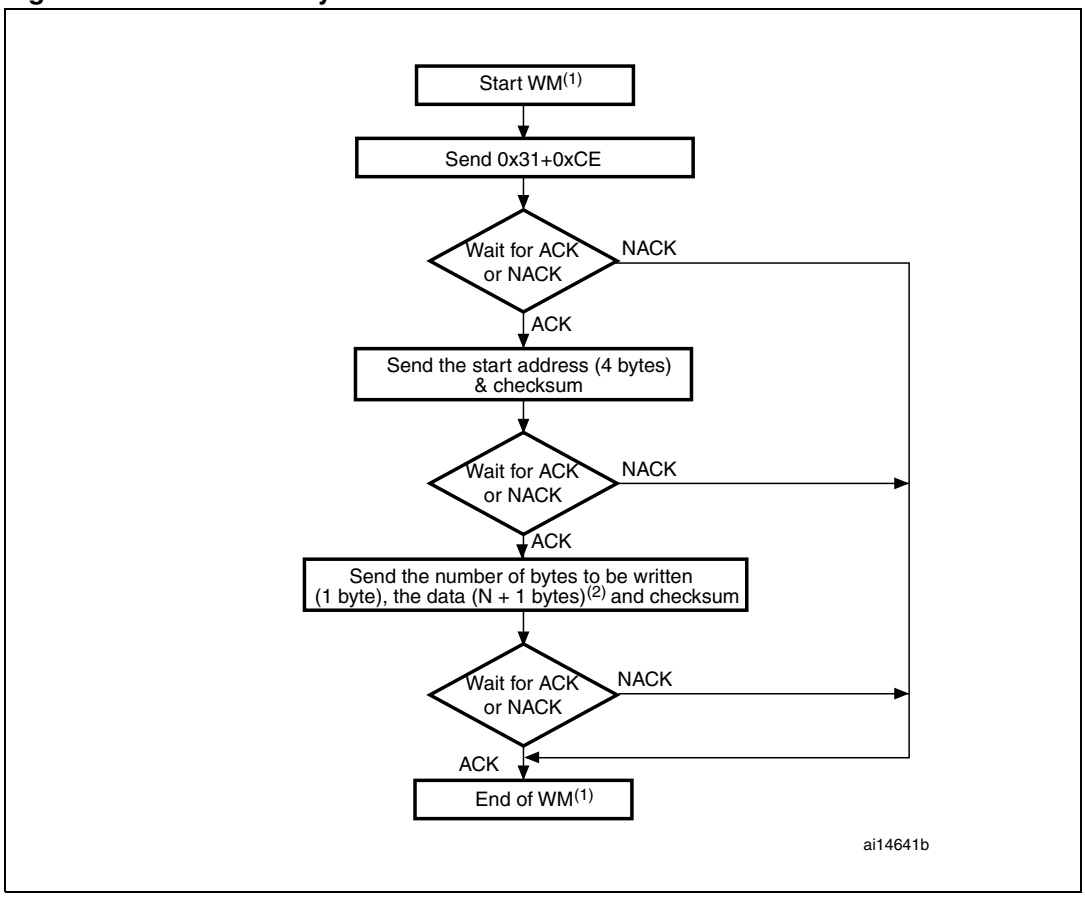
If the Write Memory command is issued to the Option byte area, all options are erased before writing the new values, and at the end of the command the bootloader generates a system Reset to take into account the new configuration of the option byte.

Note: 1 When writing to the RAM, you should take care not to overlap the first RAM memory used by the bootloader firmware.

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

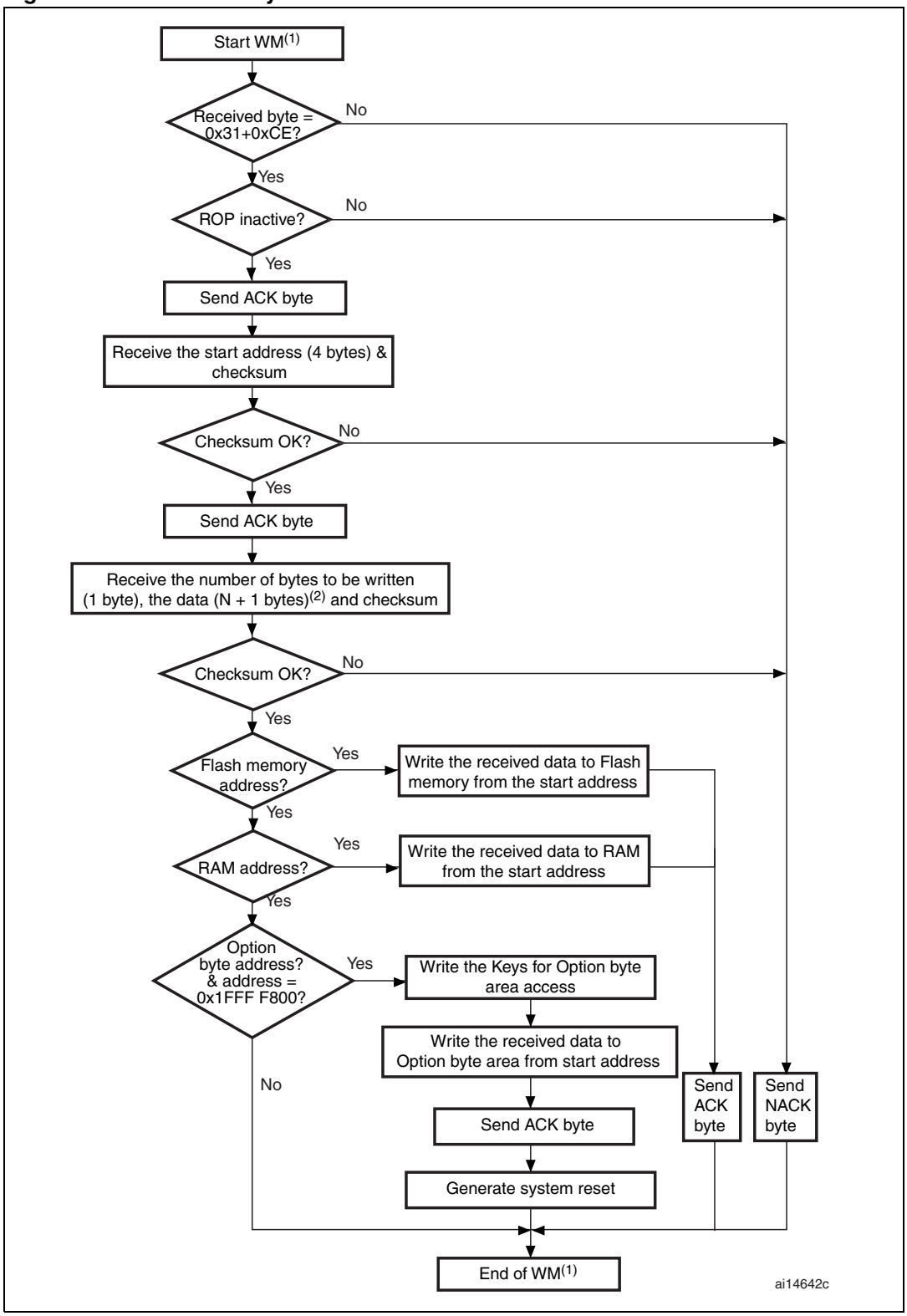
3 No error is returned when the start address is invalid.

Figure 12. Write Memory command: host side



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

Figure 13. Write Memory command: device side



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

The host sends the bytes to the STM32 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)

3.8 Erase Memory command

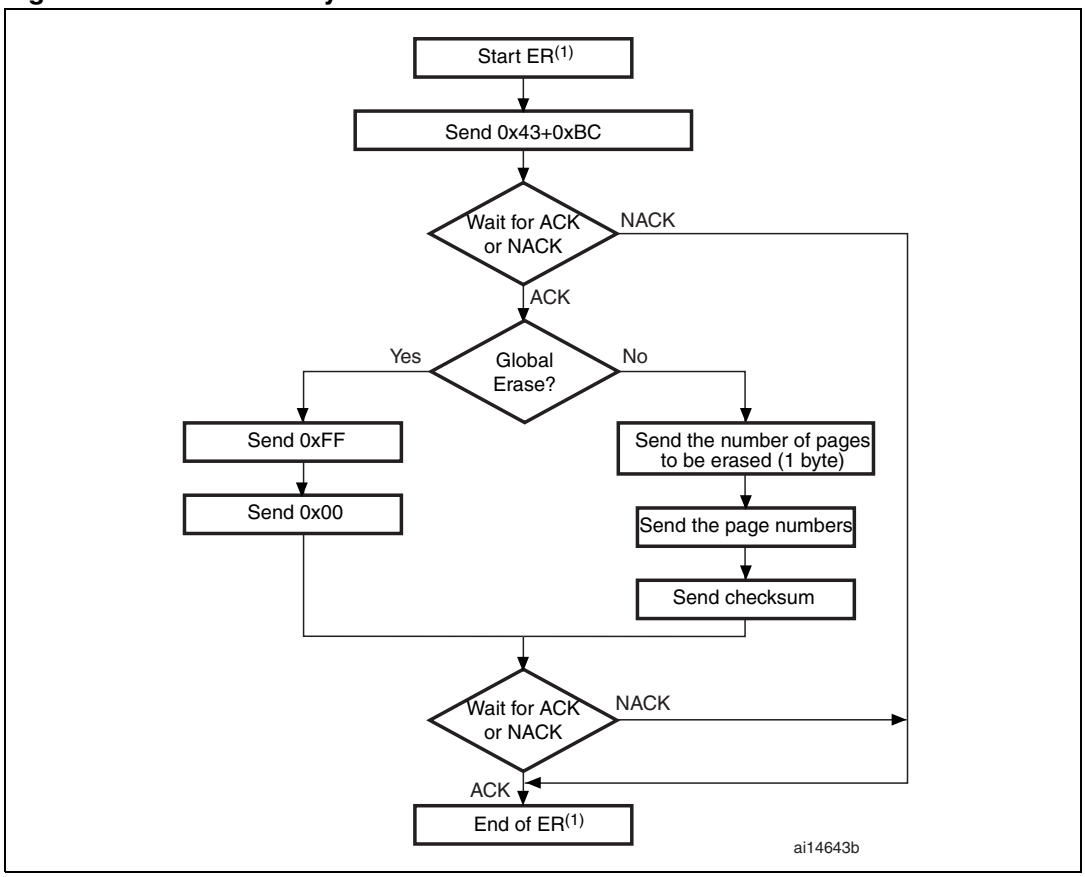
The Erase Memory command allows the host to erase Flash memory pages. When the bootloader receives the Erase Memory command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader receives one byte (number of pages to be erased), the Flash memory page codes and 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

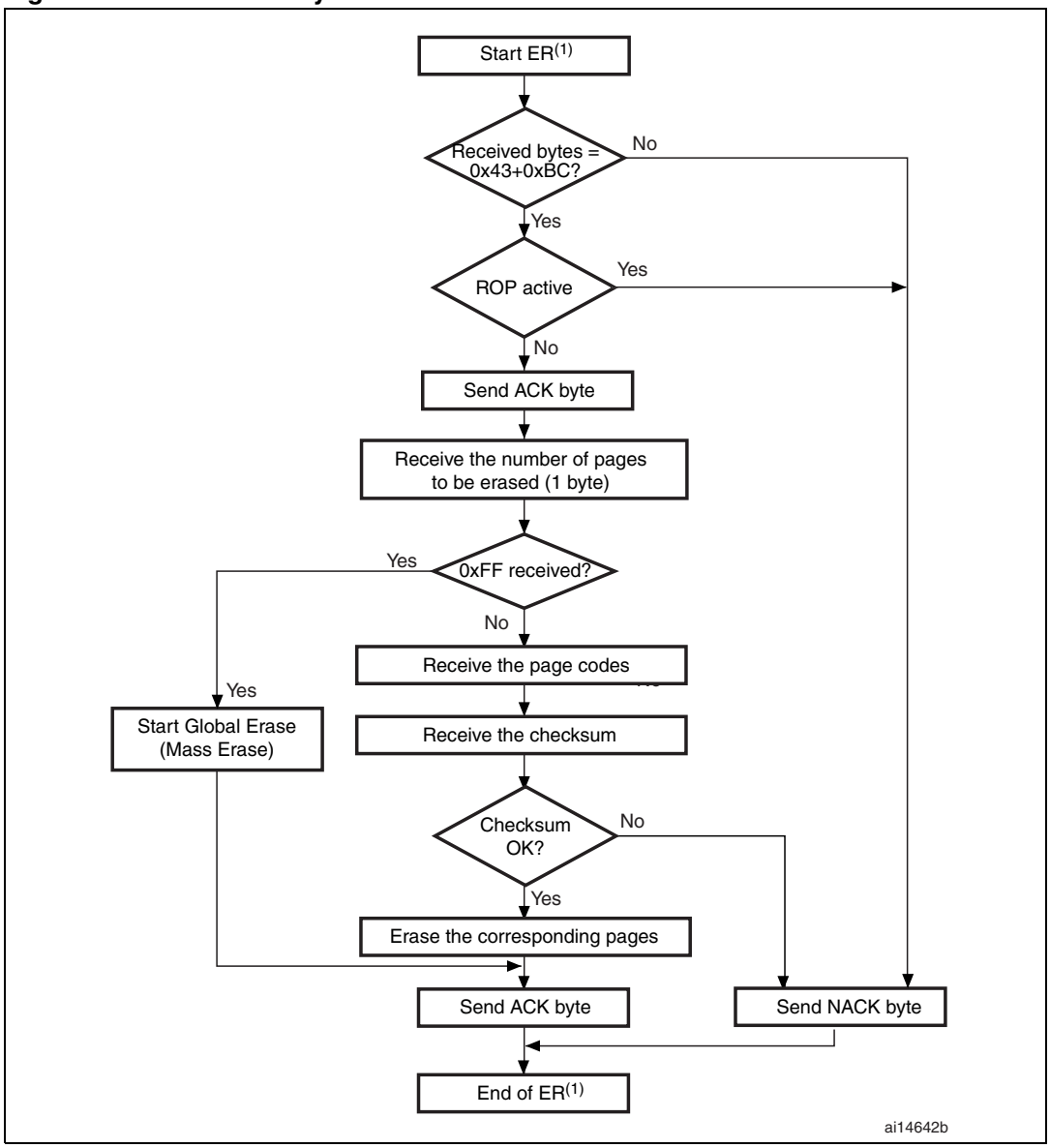
Note: No error is returned when performing erase operations on write protected sectors.

Figure 14. Erase Memory command: host side



1. ER = Erase Memory.

Figure 15. Erase Memory command: device side



1. ER = Erase Memory.

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

The host sends bytes to the STM32 as follows:

Byte 1: 0x43

Byte 2: 0xBC

Wait for ACK

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

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

3.9 Extended Erase Memory command

The Extended Erase Memory command allows the host to erase Flash memory pages using two bytes addressing mode. When the bootloader receives the Extended Erase Memory command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader receives two bytes (number of pages to be erased), the Flash memory page codes (each one coded on two bytes, MSB first) and a checksum byte (XOR of the sent bytes); if the checksum is correct, the bootloader erases the memory and sends an ACK byte to the host. Otherwise it sends a NACK byte to the host and the command is aborted.

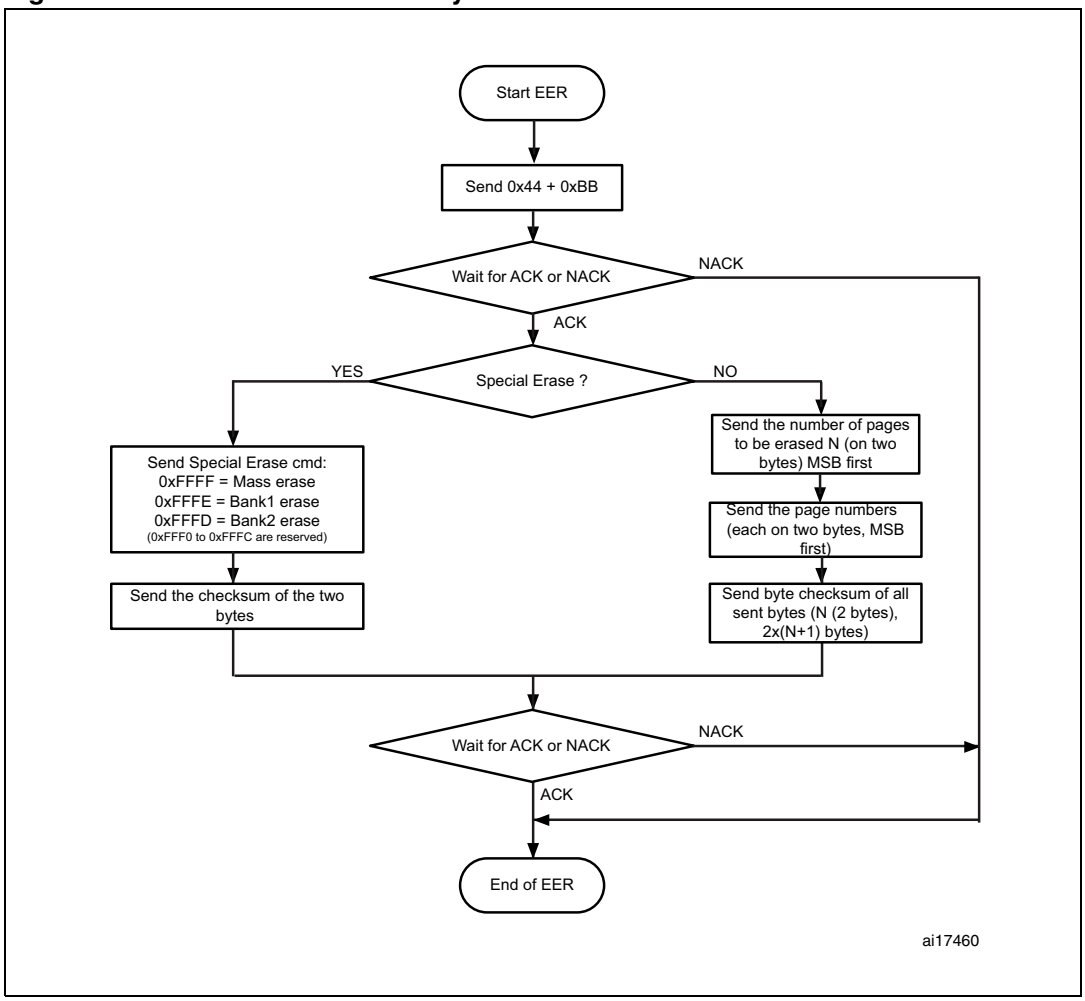
Extended Erase Memory command specifications:

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

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

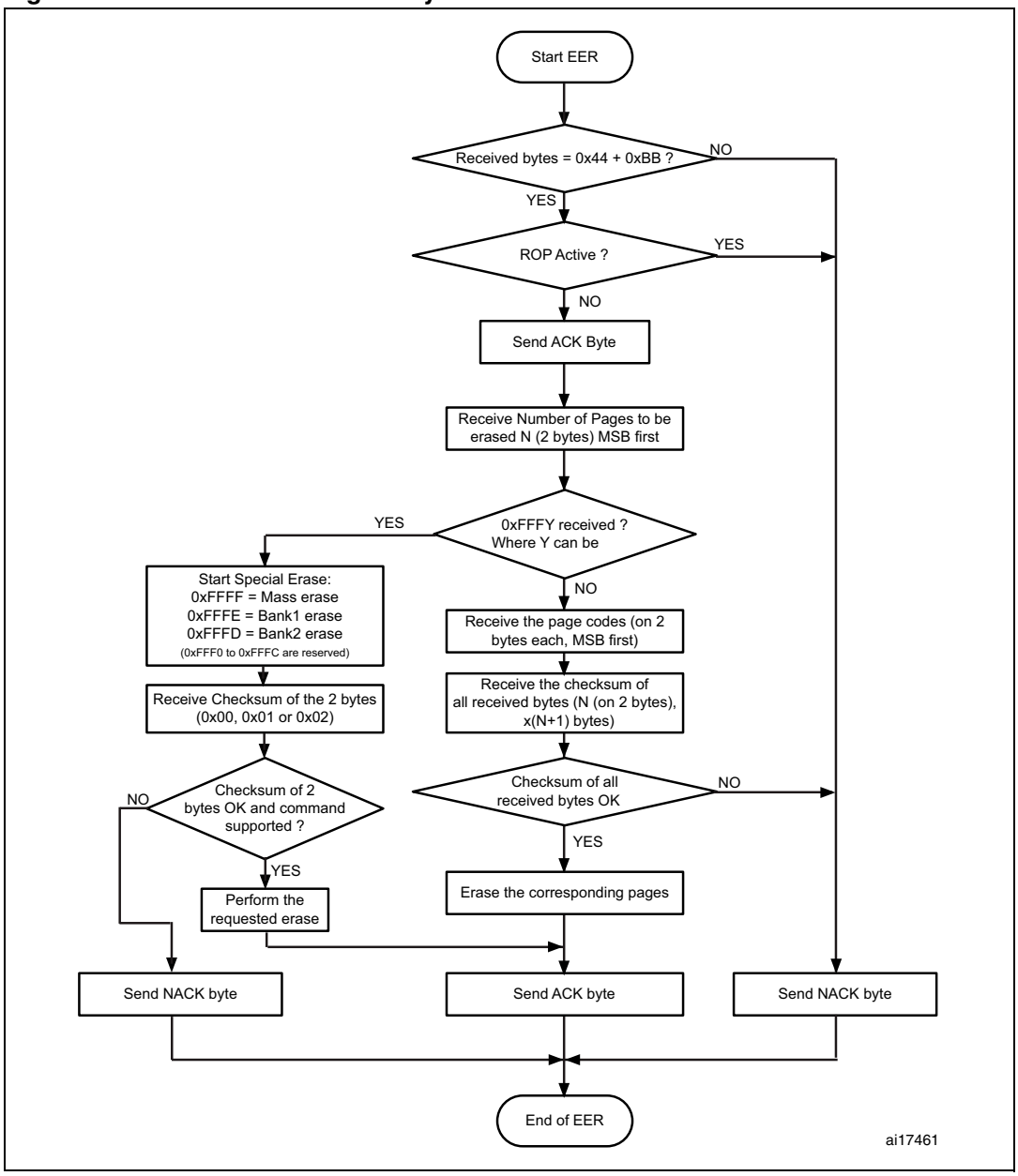
The maximum number of pages is relative to the product and thus should be respected.

Figure 16. Extended Erase Memory command: host side



1. EER = Extended Erase Memory

Figure 17. Extended Erase Memory command: device side



1. EER = Extended Erase Memory.

The host sends the bytes to the STM32F1xxx as follows:

Byte 1: 0x44

Byte 2: 0xBB

Wait for ACK

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

OR

- Number of pages to be erased (N+1 where: 0 ≤N < Maximum number of pages).

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

OR

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

3.10 Write Protect command

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

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

Note: Refer to Section 3.1: Device-dependent bootloader parameters for more details about the sector size for the device you are using.

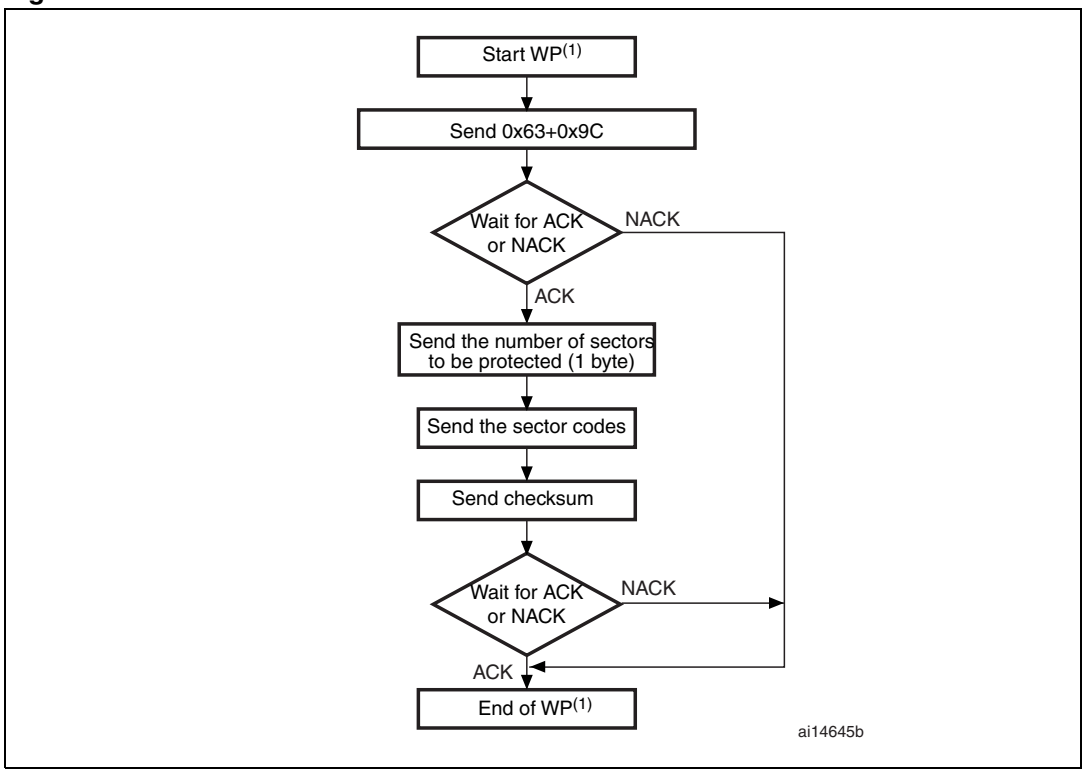
The Write Protect command sequence is as follows:

- the bootloader receives one byte that contains N, the number of sectors to be write-protected – 1 (0 ≤ N ≤ 255)
- the bootloader receives (N + 1) bytes, each byte contains a sector code

Note: 1 The total number of sectors and the sector number to be protected are not checked, this means that no error is returned when a command is passed with a wrong number of sectors to be protected or a wrong sector number.

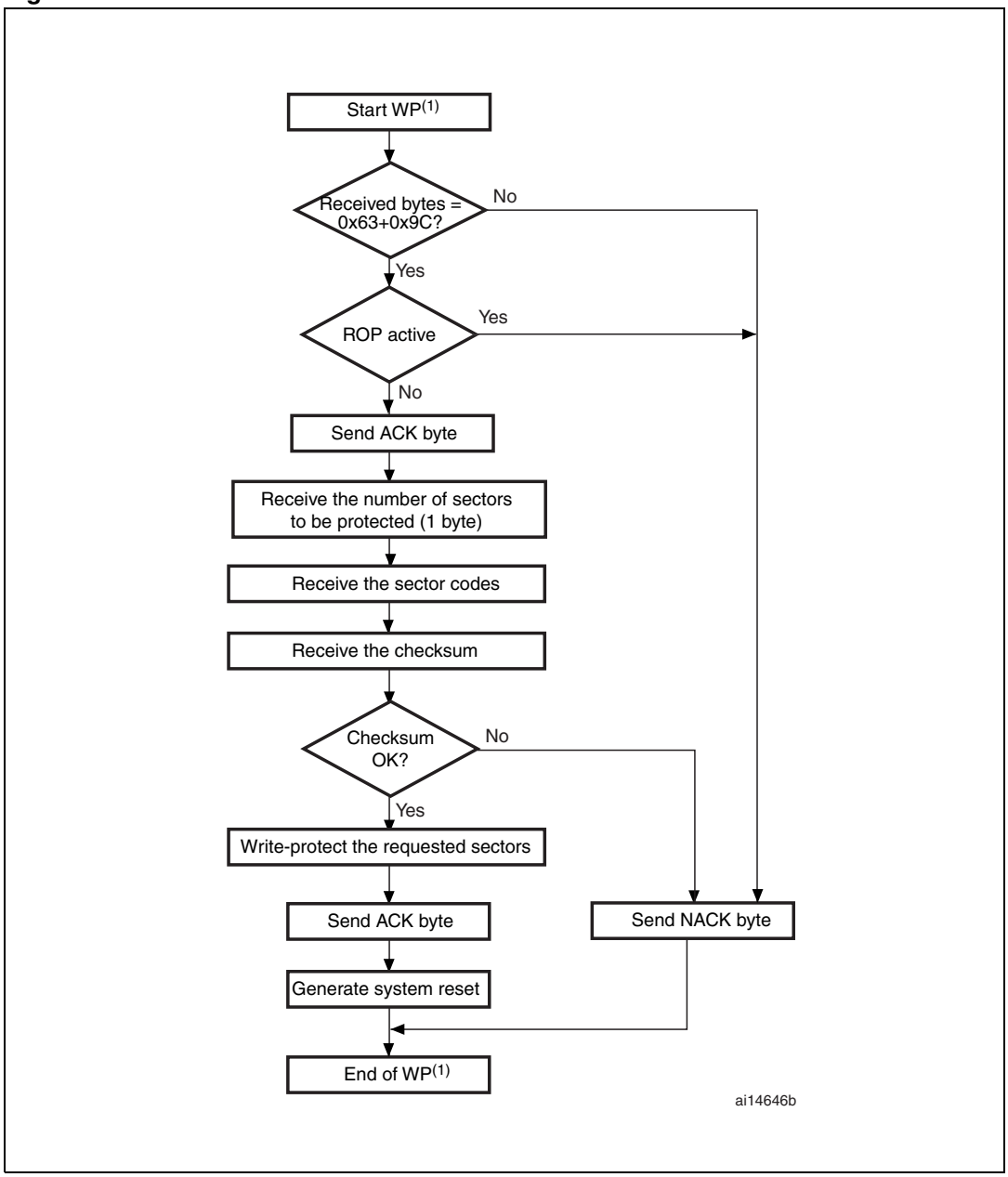
2 If a second Write Protect command is executed, the Flash memory sectors that had been protected by the first command become unprotected and only the sectors passed within the second Write Protect command become protected.

Figure 18. Write Protect command: host side



1. WP = Write Protect.

Figure 19. Write Protect command: device side



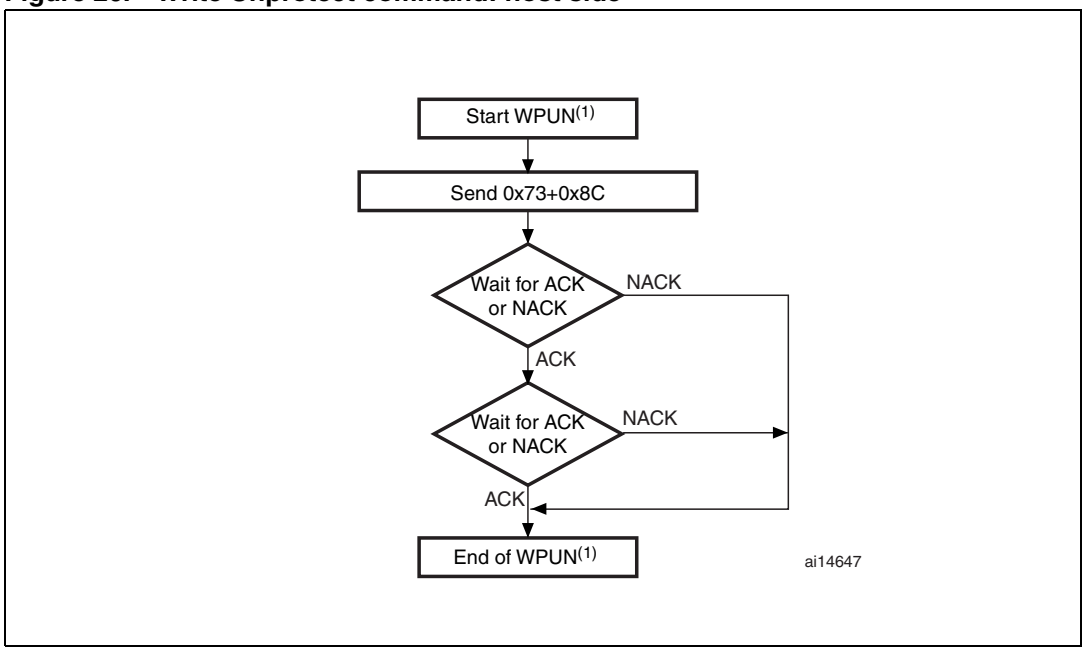
1. WP = Write Protect.

3.11 Write Unprotect command

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

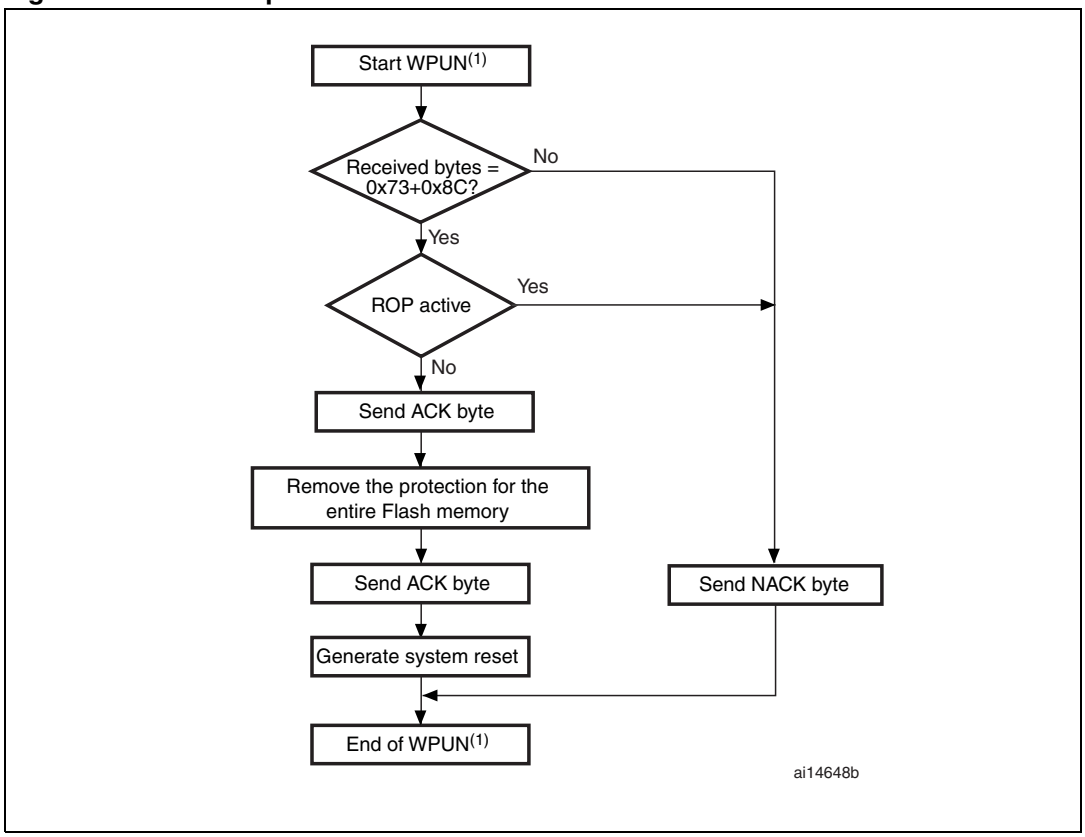
At the end of the Write Unprotect command, the bootloader transmits the ACK byte and generates a system Reset to take into account the new configuration of the option byte.

Figure 20. Write Unprotect command: host side



1. WPUN = Write Unprotect.

Figure 21. Write Unprotect command: device side



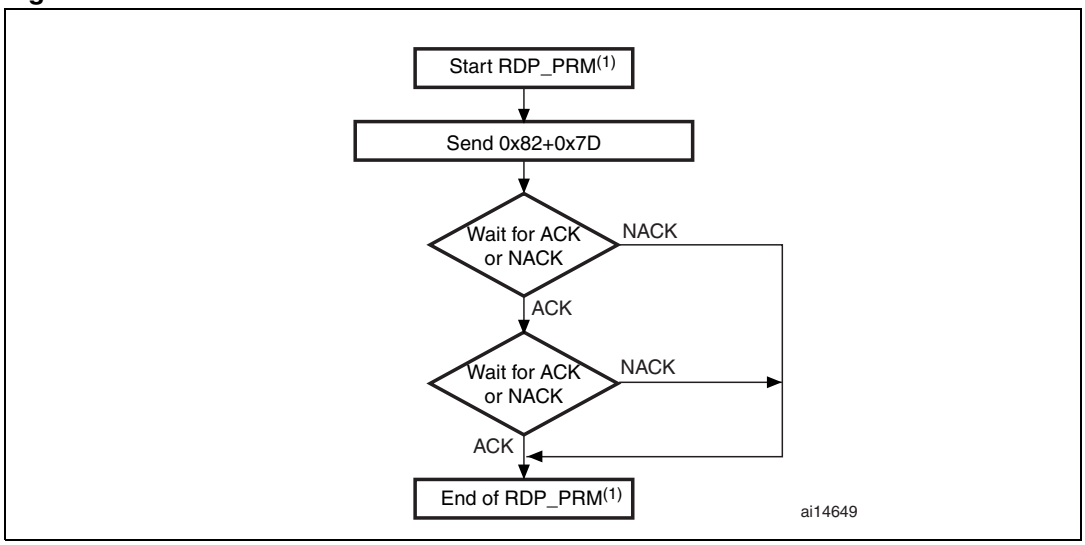
1. WPUN = Write Unprotect.

3.12 Readout Protect command

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

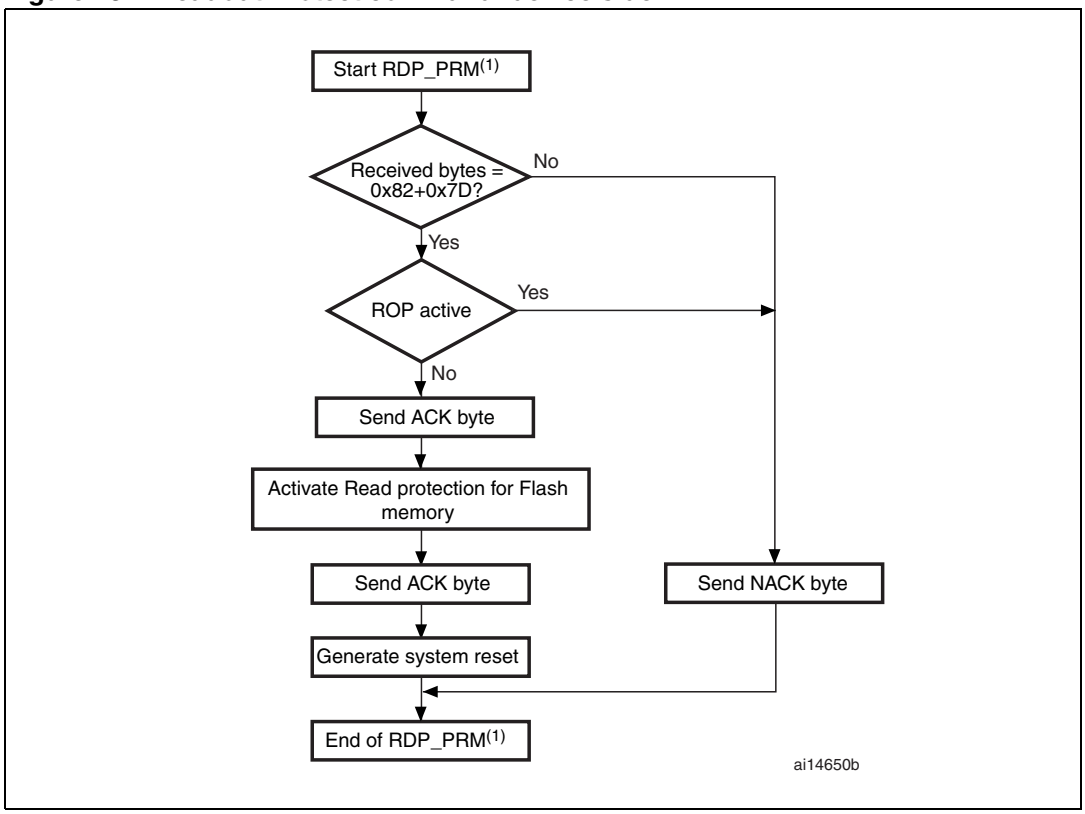
At the end of the Readout Protect command, the bootloader transmits the ACK byte and generates a system Reset to take into account the new configuration of the option byte.

Figure 22. Readout Protect command: host side



1. RDP_PRM = Readout Protect.

Figure 23. Readout Protect command: device side



1. RDP_PRM = Readout Protect.

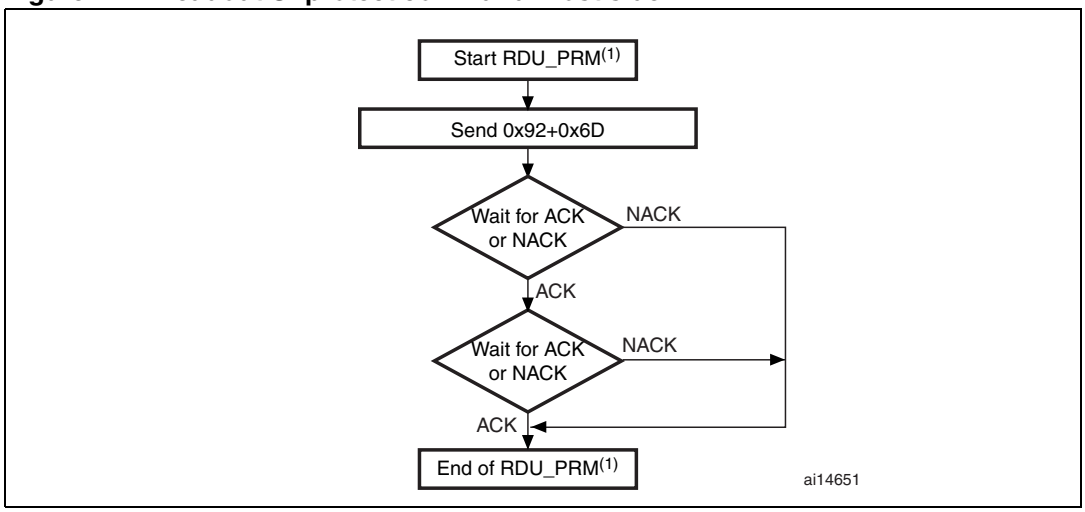
3.13 Readout Unprotect command

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

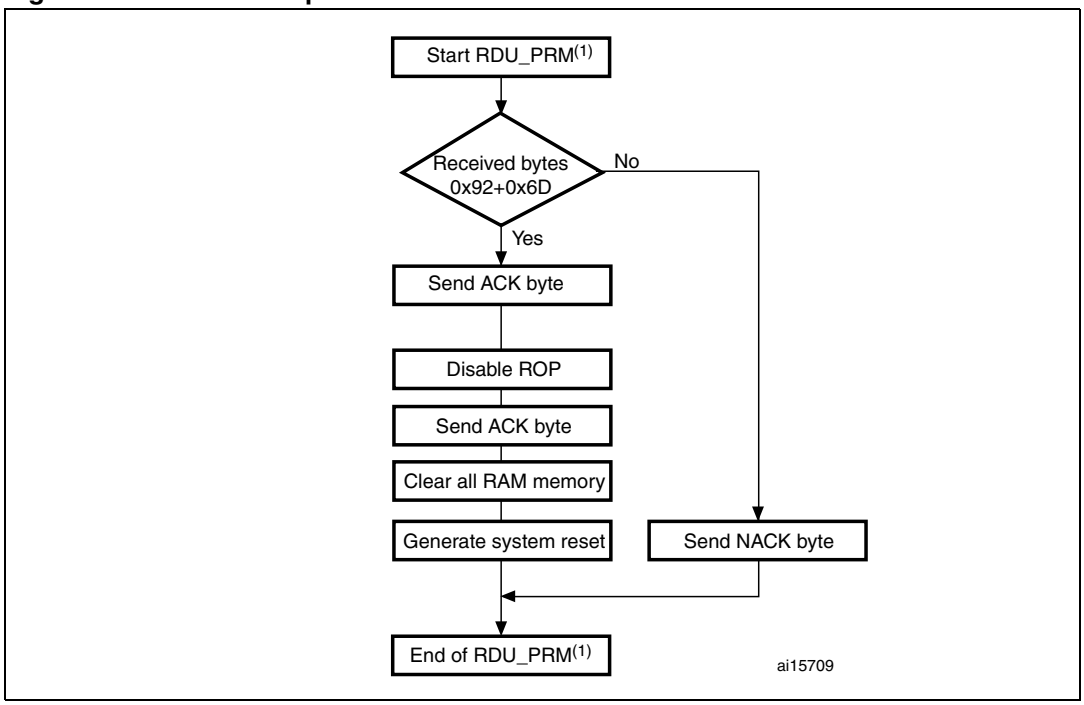
At the end of the Readout Unprotect command, the bootloader transmits an ACK and generates a system Reset to take into account the new configuration of the option byte.

Figure 24. Readout Unprotect command: host side



1. RDU_PRM = Readout Unprotect.

Figure 25. Readout Unprotect command: device side



2. RDU_PRM = Readout Unprotect.

4 Bootloader protocol version evolution

Table 2 lists the bootloader versions.

Table 2. Bootloader protocol versions

Version	Description
V2.0	Initial bootloader version.
V2.1	– Update Go command to initialize the main stack pointer – Update Go command to return NACK when jump address is in the Option byte area or System memory area – Update Get ID command to return the device ID on two bytes – Update the bootloader version to V2.1
V2.2	– Update Read Memory, Write Memory and Go commands to deny access, with a NACK response, to the first bytes of RAM memory used by the bootloader – Update Readout Unprotect command to initialize the whole RAM content to 0x0 before ROP disable operation
V3.0	– Extended Erase command added to support number of pages larger than 256 and separate bank mass erase. – Erase command has not been modified in this version but, due to addition of the Extended Erase command it is no longer supported (Erase and Extended Erase commands are exclusive).

5 Revision history

Table 3. Document revision history

Date	Revision	Changes
09-Mar-2010	1	Initial release.
20-Apr-2010	2	*Table 1: USART bootloader commands*: added Extended Erase command; removed footnote 2 concerning read protection from the Readout Protect command. *Communication safety*: amended *Note 1*. *Section 3.2: Get command*: updated byte 10. Updated *Figure 10: Go command: host side* for missing ACK state. *Section 3.7: Write Memory command*: added *Note 1* and *Note 2*. *Figure 12*, and *Figure 13*: added notes regarding N+1. Added *Section 3.9: Extended Erase Memory command*. *Table 2: Bootloader protocol versions*: added v3.0.