AN3154
Application note

CAN protocol used in the STM32 bootloader

Introduction

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

This document applies to the STM32 products embedding bootloader versions V3.x, V7.x and V9.x, as specified in *STM32 microcontroller system memory boot mode* (AN2606) available on *www.st.com*. These products are listed in *Table 1*, and are referred to as STM32 throughout the document.

For more information about the CAN hardware resources and requirements for the bootloader of the used device, refer to the already mentioned AN2606.

Table 1. Applicable products

Type	Part number or product series
Microcontrollers	STM32F1 Series STM32F2 Series STM32F4 Series STM32F7 Series STM32L4 Series

Contents

1 Bootloader code sequence 5

2 CAN settings 7

3 Bootloader command set 8

3.1 Get command 8

3.2 Get Version & Read Protection Status command 11

3.3 Get ID command 13

3.4 Speed command 14

3.5 Read Memory command 17

3.6 Go command 18

3.7 Write Memory command 20

3.8 Erase Memory command 23

3.9 Write Protect command 25

3.10 Write Unprotect command 26

3.11 Readout Protect command 28

3.12 Readout Unprotect command 30

4 Bootloader protocol version evolution 32

5 Revision history 33

List of tables

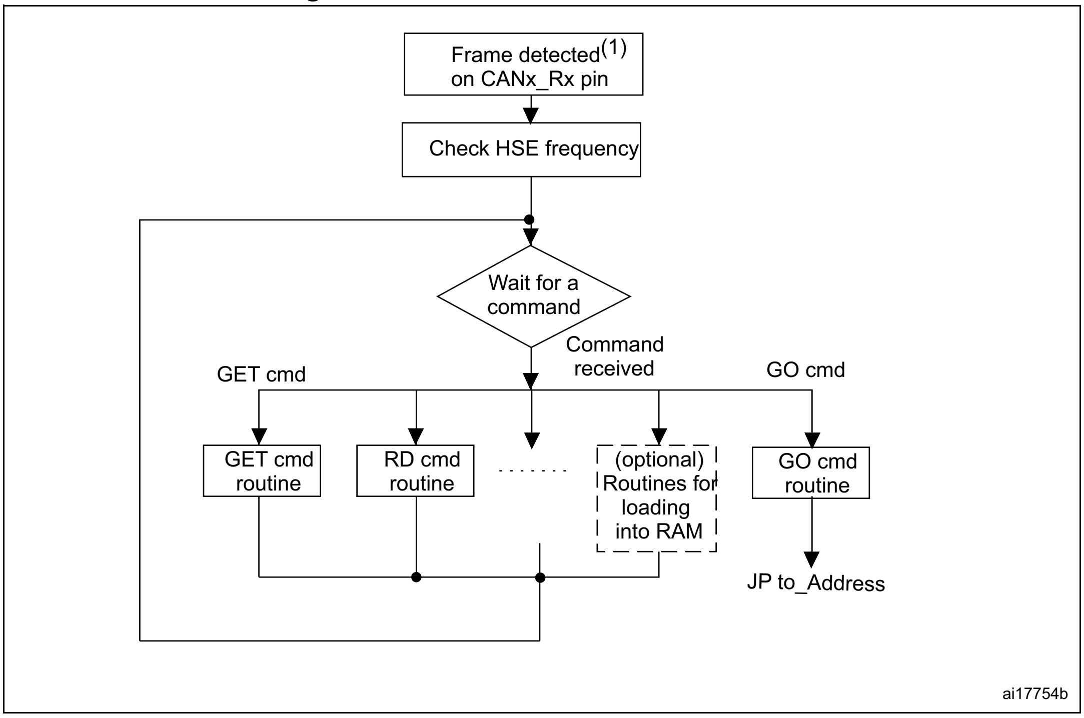
Table 1. Applicable products 1
Table 2. CAN bootloader commands 8
Table 3. Bootloader protocol versions 32
Table 4. Document revision history 33

List of figures

Figure 1. Bootloader for STM32 with CAN 5
Figure 2. Check HSE frequency 6
Figure 3. CAN frame 7
Figure 4. Get command: host side 9
Figure 5. Get command: device side 10
Figure 6. Get Version & Read Protection Status command: host side 11
Figure 7. Get Version & Read Protection Status command: device side 12
Figure 8. Get ID command: host side 13
Figure 9. Get ID command: device side 14
Figure 10. Speed command: host side 15
Figure 11. Speed command: device side 16
Figure 12. Read memory command: host side 17
Figure 13. Read memory command: device side 18
Figure 14. Go command: host side 19
Figure 15. Go command: device side 20
Figure 16. Write Memory command: host side 21
Figure 17. Write memory command: device side 22
Figure 18. Erase Memory command: host side 23
Figure 19. Erase Memory command: device side 24
Figure 20. Write Protect command: host side 25
Figure 21. Write Protect command: device side 26
Figure 22. Write Unprotect command: host side 27
Figure 23. Write Unprotect command: device side 27
Figure 24. Readout Protect command: host side 28
Figure 25. Readout Protect command: device side 29
Figure 26. Readout Unprotect command: host side 30
Figure 27. Readout Unprotect command: device side 31

1 Bootloader code sequence

Figure 1. Bootloader for STM32 with CAN



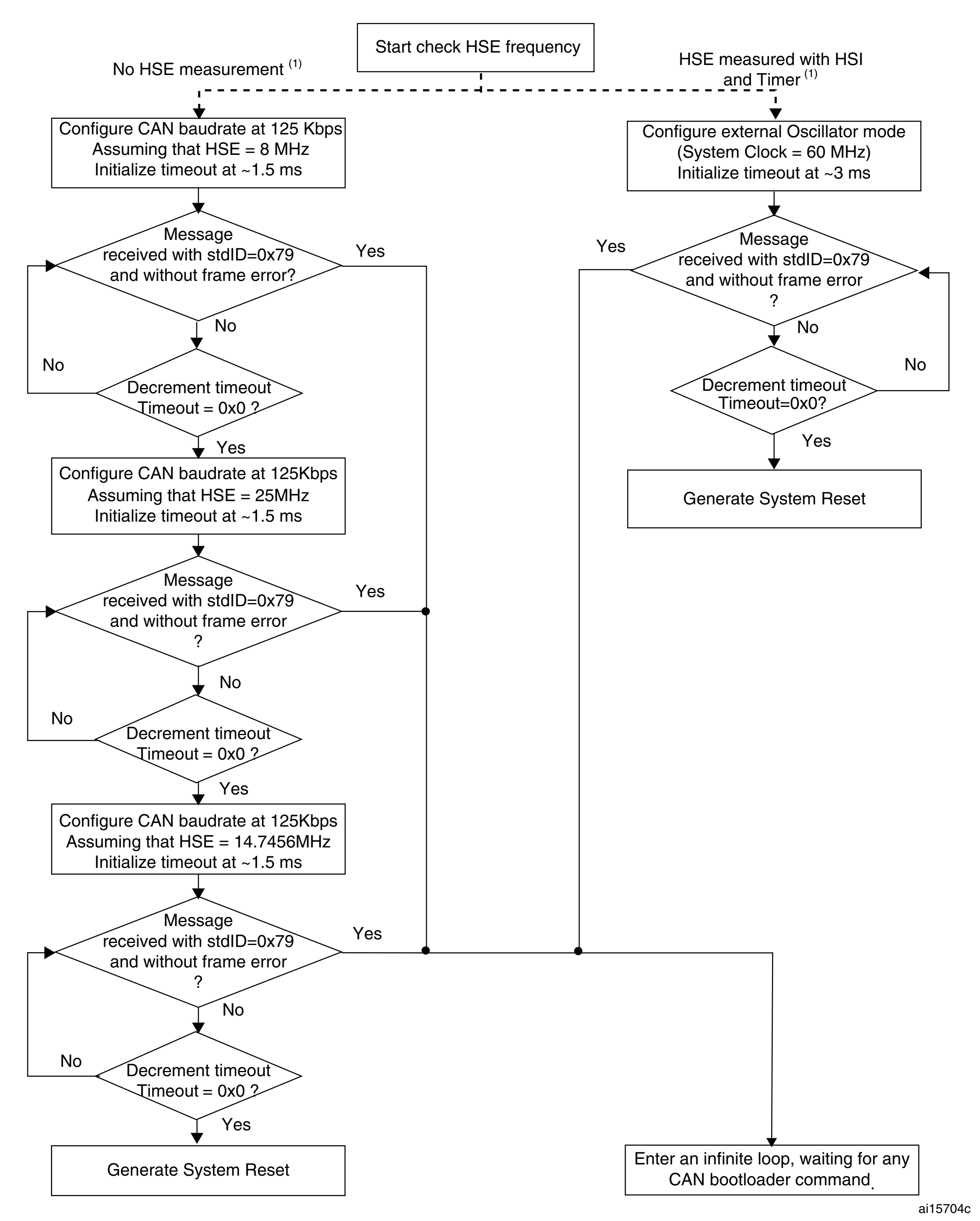
1. It is recommended to send a frame with a Standard ID = 0x79.

Once the system memory boot mode is entered and the STM32 device (based on Arm®[a] cores) has been configured (for more details refer to AN2606), the bootloader code waits for a frame on the CANx_Rx pin. When a detection occurs the CAN bootloader firmware starts to check the external clock frequency.

Figure 2 shows the flowchart of the frequency check.

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

Figure 2. Check HSE frequency



1. For some devices the HSE frequency is calculated using HSI oscillator connected to a timer. For other devices this measurement is not implemented. For the devices without HSE frequency measurement, only the flow represented on the left is executed, while for the devices with HSE frequency measurement only the flow on the right is executed. To know the flow for the used device refer to AN2606.

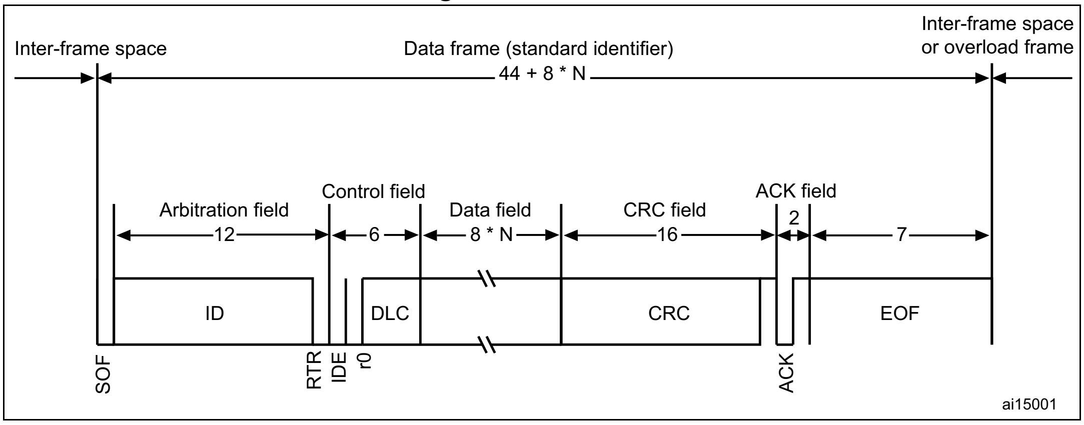
Next, the code initializes the serial interface accordingly. Using this calculated baud rate, an acknowledge byte *(0x79)* is returned to the host, indicating that the STM32 is ready to receive commands.

2 CAN settings

The STM32 CAN is compliant with the 2.0A and B (active) specifications with a bit rate up to 1 Mbit/s. It can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

Figure 3 shows a CAN frame that uses the standard identifier only.

Figure 3. CAN frame



In this application the CAN settings are:

- Standard identifier (not extended)
- Bit rate: at the beginning it is 125 kbps; during runtime it can be changed via the speed command to achieve a maximum bit rate of 1 Mbps.

The transmit settings (from the STM32 to the host) are:

- Tx mailbox0: On
- Tx mailbox1 and Tx mailbox2: Off
- Tx identifier: (0x00, 0x01, 0x02, v03, 0x11, 0x21, 0x31, 0x43, 0x63, 0x73, 0x82, 0x92)

The receive settings (from the host to the STM32) are:

- Synchronization byte, 0x79, is in the RX identifier and not in the data field.
- RX identifier depends on the command (0x00, 0x01, 0x02, 0x03, 0x11, 0x21, 0x31, 0x43, 0x63, 0x73, 0x82, 0x92)
- Error checking: If the error field (bit [6:4] in the CAN_ESR register) is different from 000b, the message is discarded and a NACK is sent to the host
- In FIFO overrun condition, the message is discarded and a NACK is sent to the host
- Incoming messages can contain from 1 to 8 data bytes.

Note: The CAN bootloader firmware supports only one node at a time. This means that CAN network management is not supported by the firmware.

3 Bootloader command set

The supported commands are listed in *Table 2*, each of them is described in this section.

Table 2. CAN bootloader commands

Command	Command code	Command description
Get(1)	0x00	Gets the version and the allowed commands supported by the current version of the bootloader
Get Version & Read Protection Status(1)	0x01	Gets the bootloader version and the Read Protection status of the Flash memory
Get ID(1)	0x02	Gets the chip ID
Speed	0x03	The speed command allows the baud rate for CAN run-time to be changed.
Read Memory(2)	0x11	Reads up to 256 bytes of memory starting from an address specified by the application
Go(2)	0x21	Jumps to user application code located in the internal Flash memory or in SRAM
Write Memory(2)	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the application
Erase(2)	0x43	Erases from one to all the Flash memory sectors
Write Protect	0x63	Enables the write protection for some sectors
Write Unprotect	0x73	Disables the write protection for all Flash memory sectors
Readout Protect(1)	0x82	Enables the read protection
Readout Unprotect(1)	0x92	Disables the read protection

1. Read protection – When the RDP (read protection) option is active, only this limited subset of commands is available. All other commands are NACK-ed and have no effect on the device. Once the RDP has been removed, the other commands become active.
2. Refer to STM32 product datasheet and AN2606 to know the memory spaces valid for these commands.

Communication safety

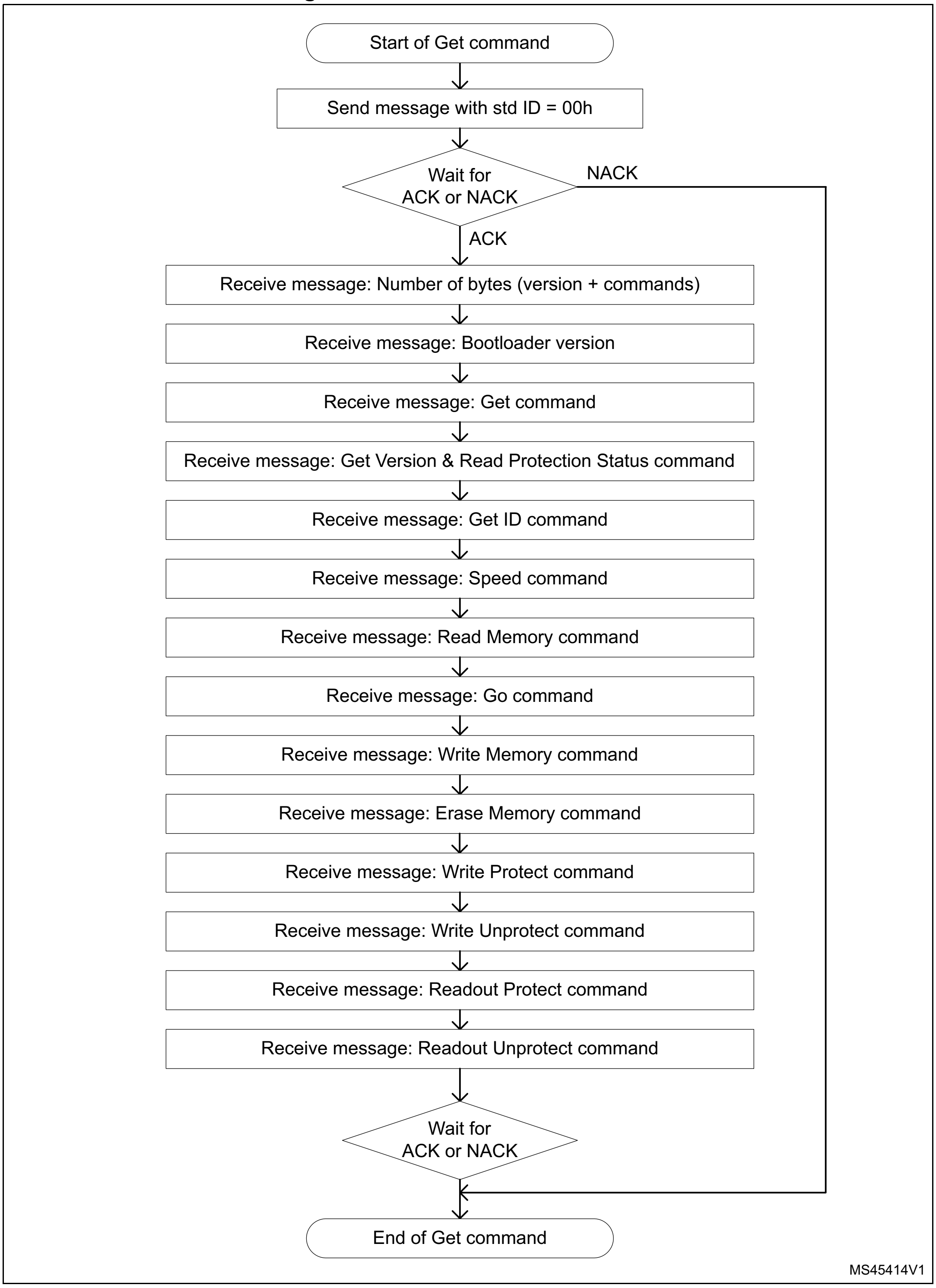
Each packet is either accepted (ACK answer) or discarded (NACK answer):

- ACK message = 0x79
- NACK message = 0x1F

3.1 Get command

The Get command allows the host to get the version of the bootloader and the supported commands. When the bootloader receives this command, it transmits the bootloader version and the supported command codes to the host.

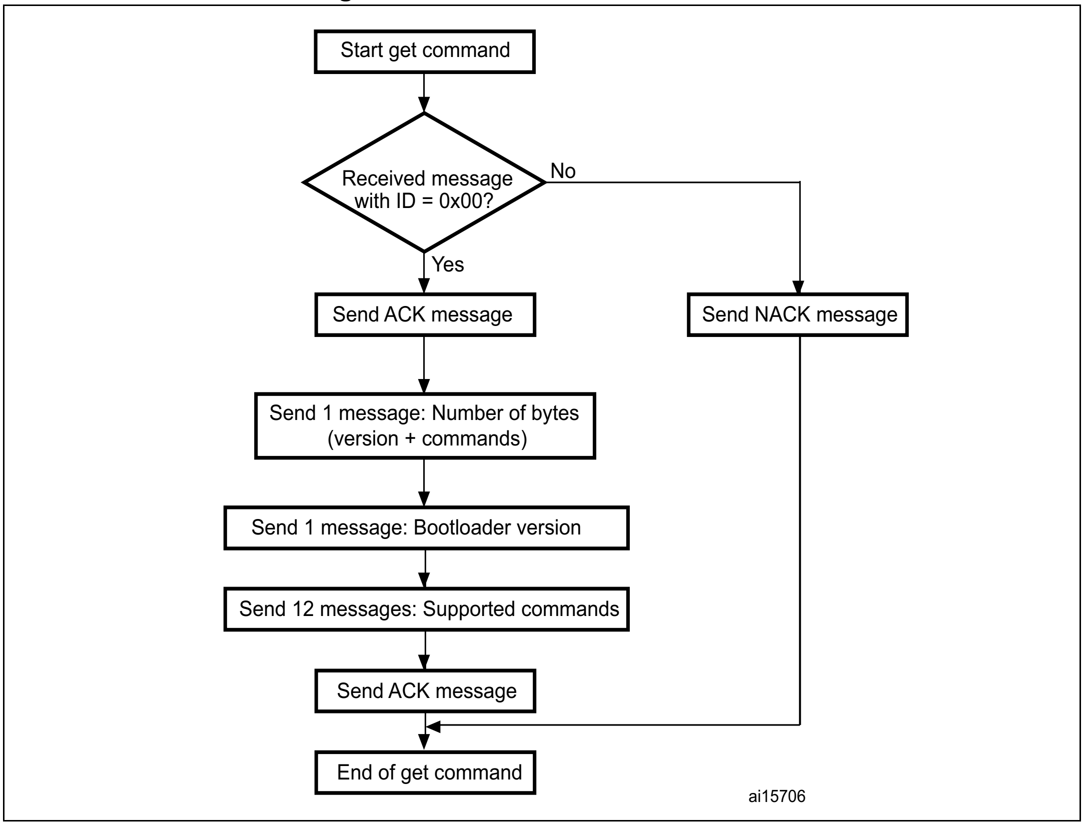
Figure 4. Get command: host side



The host sends messages as follows:

Command message: Std ID = 0x00, data length code (DLC) = ‘not important’.

Figure 5. Get command: device side



The STM32 sends messages as follows:

Message 1: Std ID = 0x00, DLC = 1, data = 0x79 - ACK

Message 2: Std ID = 0x00, DLC = 1 data = N = 12 = the number of bytes to be sent -1 ($1 \leq N + 1 \leq 256$)

Message 3: Std ID = 0x00, DLC = 1, data = bootloader version ($0 <$ version ≤ 255)

Message 4: Std ID = 0x00, DLC = 1, data = 0x00 - Get command

Message 5: Std ID = 0x00, DLC = 1, data = 0x01 - Get Version & Read Protection

Status command

Message 6: Std ID = 0x00, DLC = 1, data = 0x02 - Get ID command

Message 7: Std ID = 0x00, DLC = 1, data = 0x03 - Speed command

Message 8: Std ID = 0x00, DLC = 1, data = 0x11 - Read memory command

Message 9: Std ID = 0x00, DLC = 1, data = 0x21 - Go command

Message 10: Std ID = 0x00, DLC = 1, data = 0x31 - Write memory command

Message 11: Std ID = 0x00, DLC = 1, data = 0x43 - Erase memory command

Message 12: Std ID = 0x00, DLC = 1, data = 0x63 - Write Protect command

Message 13: Std ID = 0x00, DLC = 1, data = 0x73 - Write Unprotect command

Message 14: Std ID = 0x00, DLC = 1, data = 82h - Readout Protect command

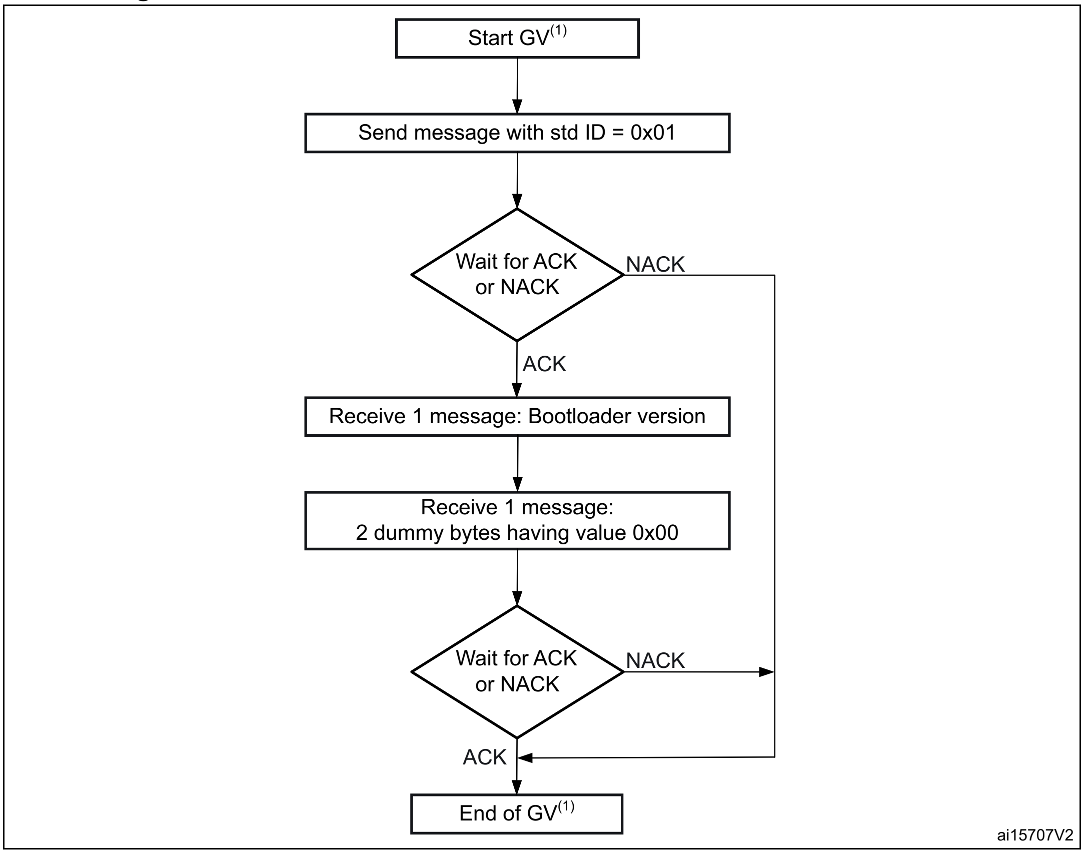
Message 15: Std ID = 0x00, DLC = 1, data = 92h - Readout Unprotect command

Message 16: Std ID = 0x00, DLC = 1, data = 0x79 - ACK

3.2 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 and two dummy bytes having value 0x00) to the host.

Figure 6. Get Version & Read Protection Status command: host side



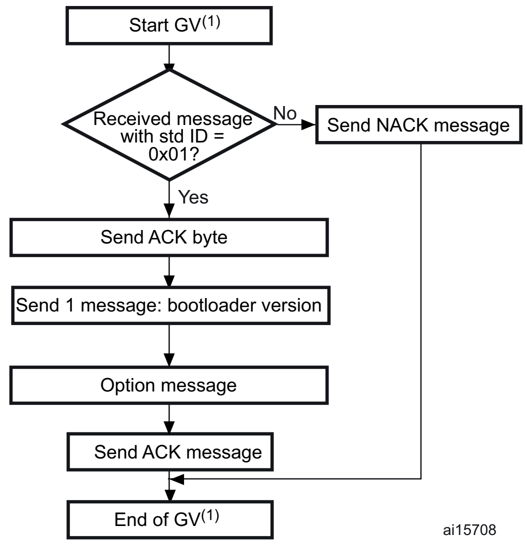
1. GV = Get Version & Read Protection Status.

The host sends messages as follows:

Command message: Std ID = 0x01, data length code (DLC) = ‘not important’.

ACK Message contain: Std ID = 0x01, DLC = 1, data = 0x79 - ACK

Figure 7. Get Version & Read Protection Status command: device side



1. GV = Get Version & Read Protection Status.

The STM32 sends messages as follows:

Message 1: Std ID = 0x01, DLC = 1, data = ACK

Message 2: Std ID = 0x01, DLC = 1, data[0] = bootloader version (0 < version ≤ 255), example: 0x10 = Version 1.0

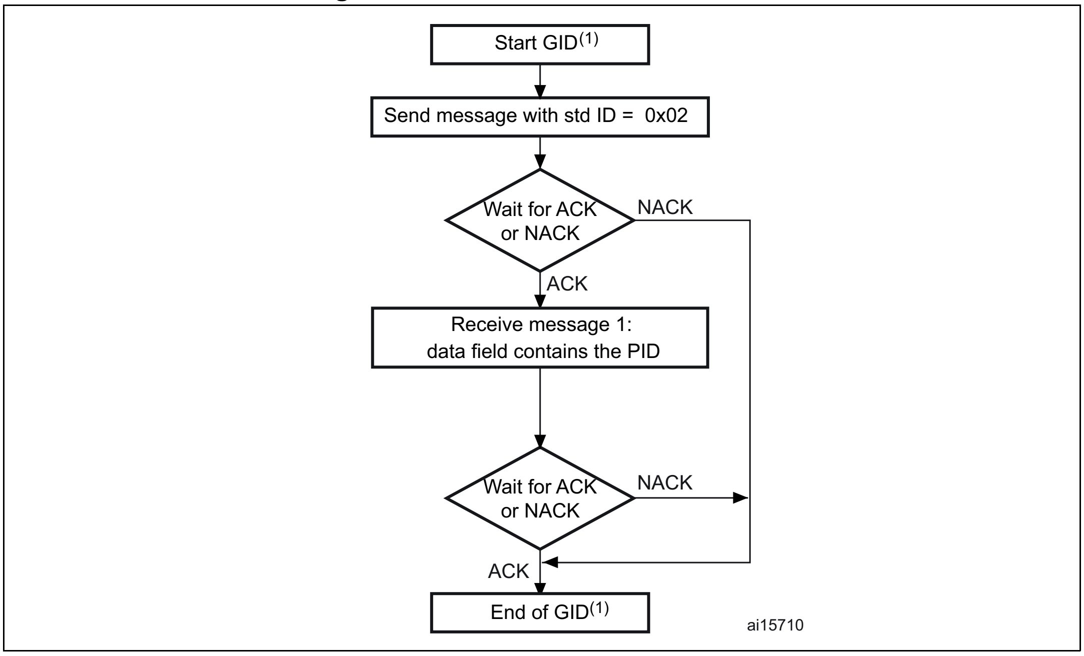
Message 3: Option message 1: Std ID = 0x01, DLC = 2, data = 0x00 (byte1 and byte 2)

Message 4: Std ID = 0x01, DLC = 1, data = ACK

3.3 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 8. Get ID command: host side



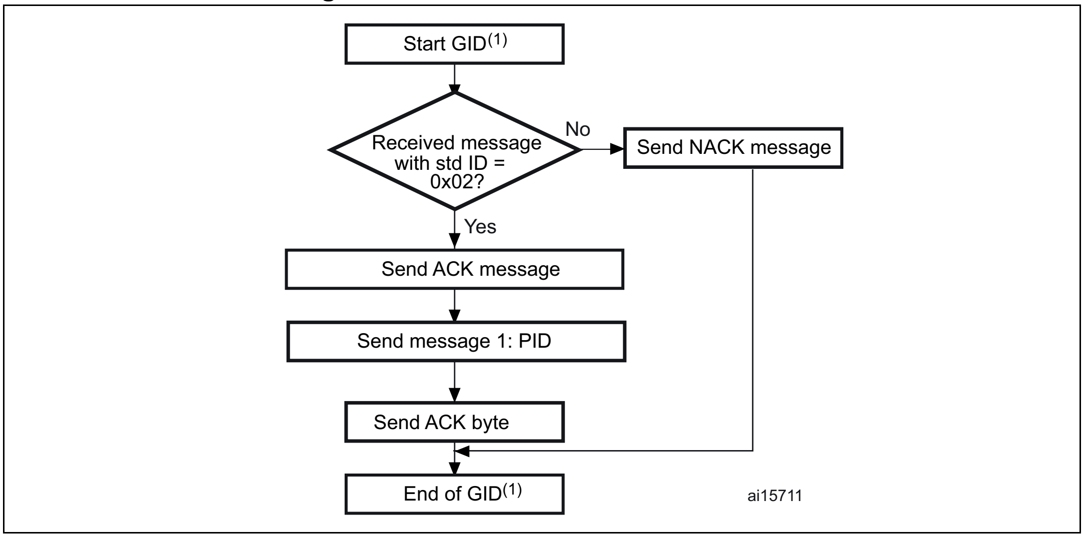
1. GID = Get ID.
2. PID stands for product ID. Byte 1 is the MSB and byte 2, the LSB of the address. Refer to *Section 3.1: Get command* for more details about the PID of the used device.

The host sends messages as follows:

Command message: Std ID = 0x02, data length code (DLC) = ‘not important’.

ACK Message contains: Std ID = 0x02, DLC = 1, data = 0x79 - ACK

Figure 9. Get ID command: device side



1. GID = Get ID.
2. PID stands for product ID. Byte 1 is the MSB and byte 2 is LSB of the address.

The STM32 sends the bytes as follows:

Message 1: Std ID = 0x02, DLC = 1, data = ACK with DLC except for current message and ACKs.

Message 2: Std ID = 0x02, DLC = N (the number of bytes – 1. For STM32, N = 1), data = PID with byte 0 is MSB and byte N is the LSB of the product ID

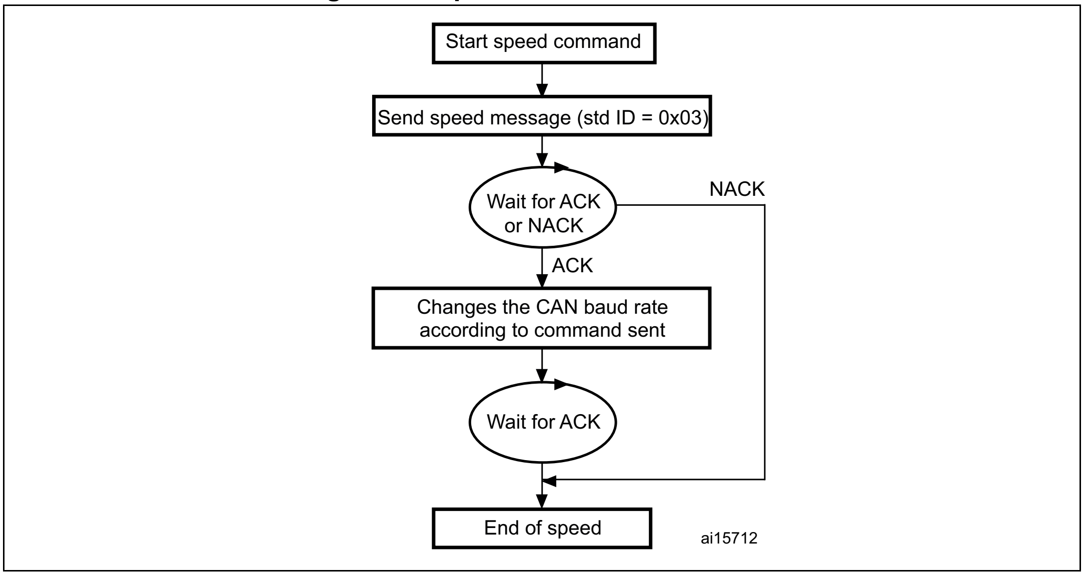
Message 3: Std ID = 0x02, DLC = 1, data = ACK = 0x79

3.4 Speed command

The speed command allows the baud rate for CAN run-time to be changed. It can be used only if CAN is the peripheral being used.

A system reset is generated if the CAN receives the correct message but the operation to set the new baud rate fails, which prevents it from entering or leaving initialization mode.

Figure 10. Speed command: host side



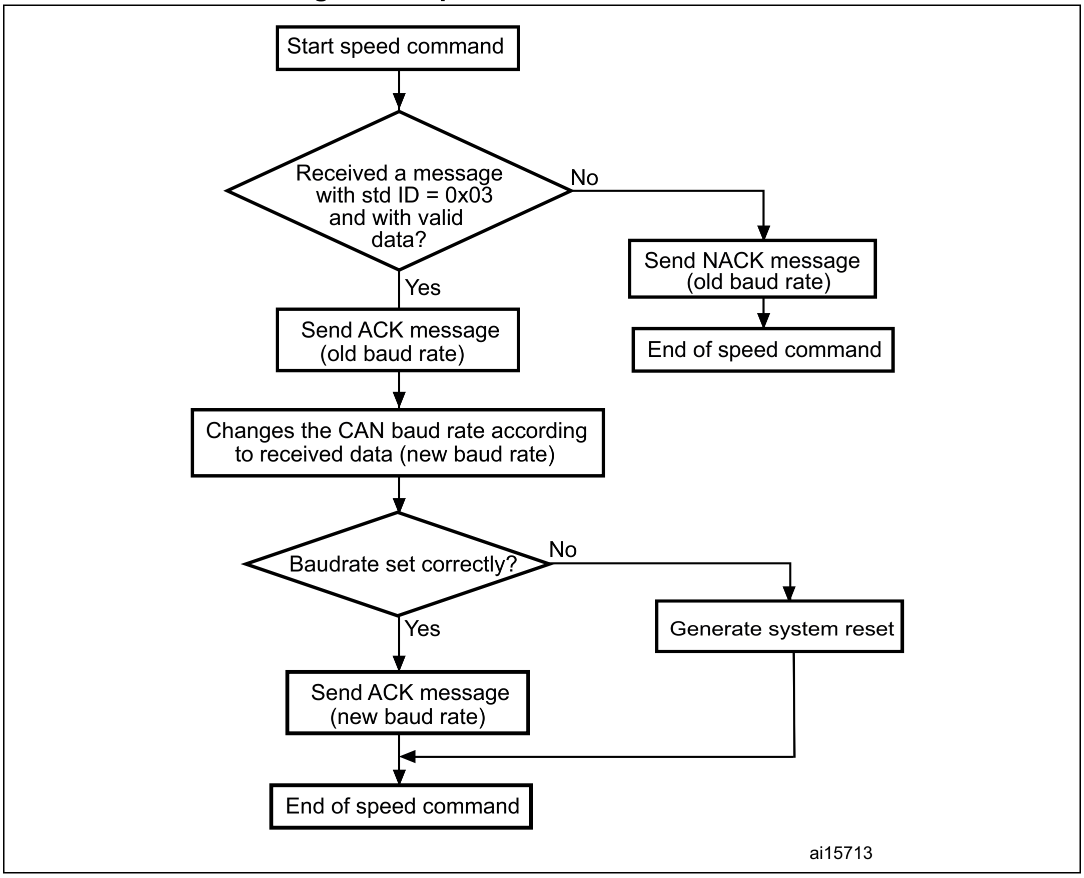
1. After setting the new baud rate, the bootloader sends the ACK message. Therefore, the host sets its baud rate while waiting for the ACK.

The host sends the message as follows:

Command message: Std ID = 0x03, DLC = 0x01, data[0] = XXh where XXh takes the following values depending on the baud rate to be set:

- 0x01: baud rate = 125 kbps
- 0x02: baud rate = 250 kbps
- 0x03: baud rate = 500 kbps
- 0x04: baud rate = 1 Mbps

Figure 11. Speed command: device side



The STM32 sends the bytes as follows:

Message 1: Std ID = 0x03, DLC = 1, data[0] = ACK= 0x79: with old baud rate if the receive message is correct else data[0] = NACK= 0x1F

Message 2: Std ID = 0x03, DLC = 1, data[0] = ACK = 0x79 with new baud rate

3.5 Read Memory command

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

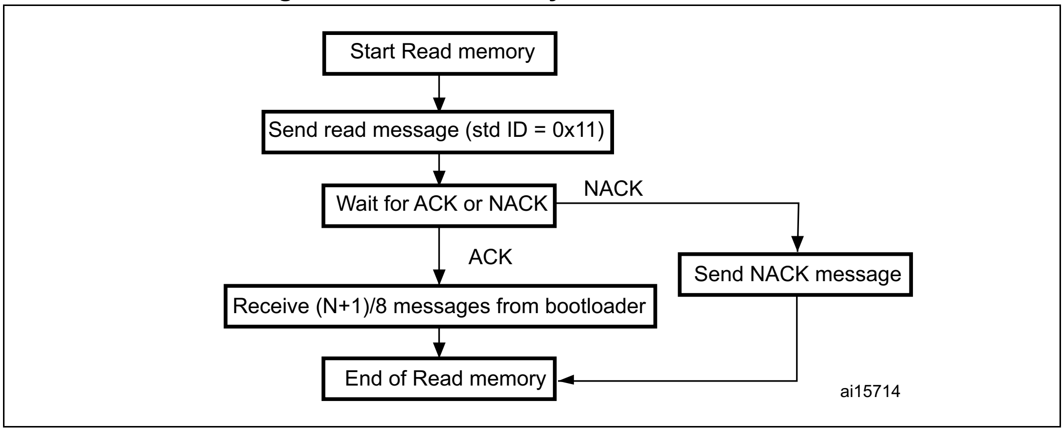
When the bootloader receives the Read Memory command, it starts to verify the contents of the message:

- ID of the command is correct or not
- ReadOutProtection is disabled or enabled
- Address to be read is valid or not

If the message content is correct it transmits an ACK message otherwise it transmits a NACK message.

After sending an ACK message, it transmits the required data to the application ((N + 1) bytes) via (N+1) messages /8 (as each message contains 8 bytes), starting from the received address.

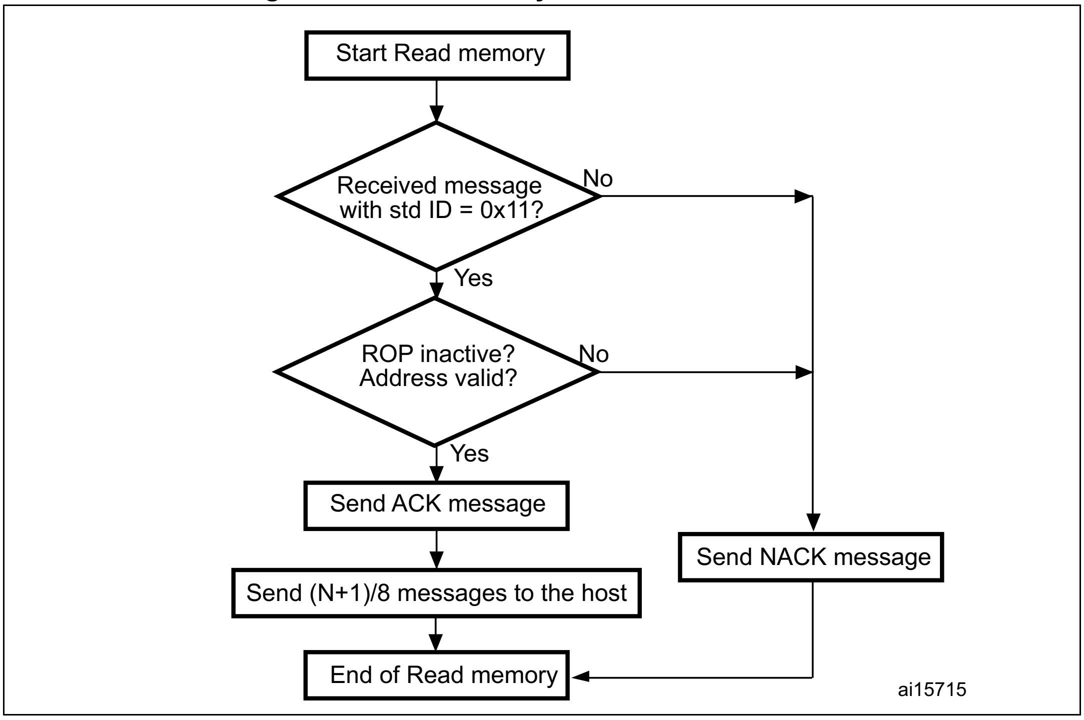
Figure 12. Read memory command: host side



The host sends messages as follows:

Command message:
Std ID = 0x11, DLC = 0x05, data[0] = 0xXX: MSB of the address... data[3] = 0xYY: LSB of the address, data[4] = N: number of bytes to be read (where $0 < N \leq 255$).

Figure 13. Read memory command: device side



The STM32 sends messages as follows:

ACK message: Std ID = 0x11, DLC = 1, data[0] = ACK if content of the command is correct else data[0] = NACK

Data message (N+1) / 8: Std ID = 0x11, DLC = Number of Byte, data[0] = 0xXX... data[Number of Byte - 1] = 0xYY

ACK message: Std ID = 0x11, DLC = 1, data[0] = ACK

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 starts if the message contains the following valid information:

- ID of the command is correct or not
- ReadOutProtection is disabled or enabled
- branch destination address is valid or not(data[0] is the address MSB and data[3] is LSB

If the message content is correct it transmits an ACK message, otherwise it transmits a NACK message.

After sending an ACK message to the application, the bootloader firmware

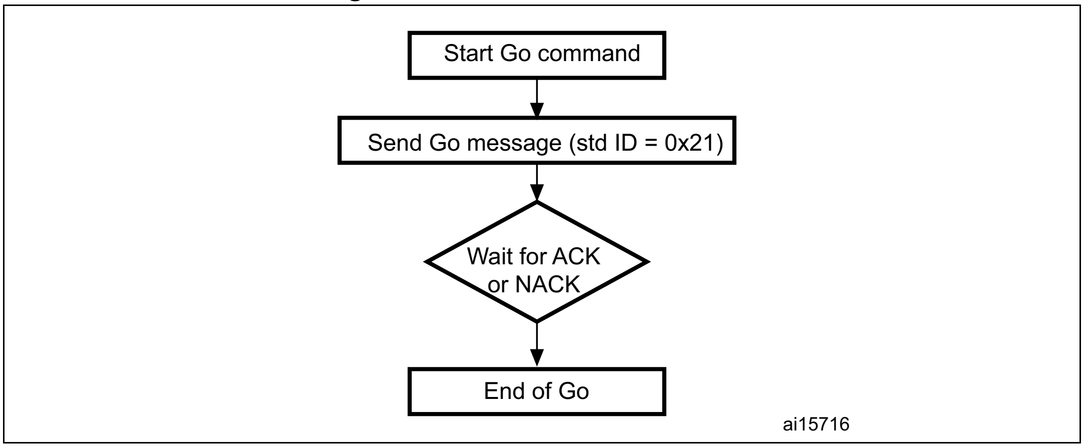
- Initializes the registers of the peripherals used by the bootloader to their default reset values
- Initializes the user application main stack pointer
- Jumps to the memory location programmed in the received ‘address + 4’ (corresponding to the address of the application’s reset handler).
  For example, if the received address is 0x0800 0000, the bootloader jumps to the memory location programmed at address 0x0800 0004.
  In general, the host sends the base address where the application to jump to is programmed.

Note: 1 The Jump to the application works only if the user application sets the vector table correctly to point to the application address.

2 The valid addresses for the Go command are in RAM or Flash memory (refer to Section 3.1 for more details about the valid memory addresses for the used device). All other addresses are considered not valid and are NACK-ed by the device.

3 When an application is loaded into RAM and a jump is made to it, the program must be configured to run with an offset to avoid overlapping with the first area used by the bootloader firmware (refer to Section 3.1 for more details about the RAM offset for the used device).

Figure 14. Go command: host side

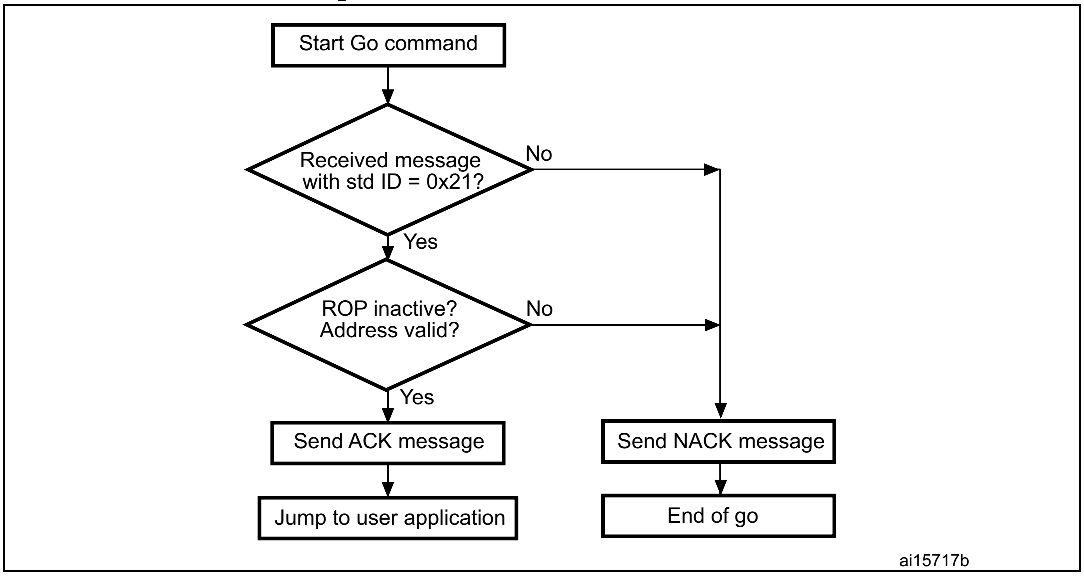


1. See product datasheet for valid addresses.

The host sends the bytes as follows

Go command message: Std ID = 0x21, DLC = 0x04, data[0] = 0xXX: MSB address,...data[3] = 0xYY LSB address.

Figure 15. Go command: device side



The STM32 send the messages as follows:

ACK message: Std ID = 0x21, DLC = 1, data[0] = ACK if content of the command is correct else data[0] = NACK

3.7 Write Memory command

The Write Memory command is used to write data to any valid memory address (see note) of RAM, Flash memory, or Option byte area. When the bootloader receives the Write Memory command, (message with 5 bytes data length, data[0] is the address MSB, data[3] is the LSB and data[4] is the number of data bytes to be received), 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: Refer to Section 3.1 for more details about the valid memory addresses for the used device.

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

- Receives the user data (N bytes) so the device receives N/8 messages (each message contains 8 data bytes)
- Programs the user data into memory starting from the received address
- At the end of the command, if the write operation was successful, the bootloader transmits the ACK message; otherwise it transmits a NACK message to the application and aborts the command

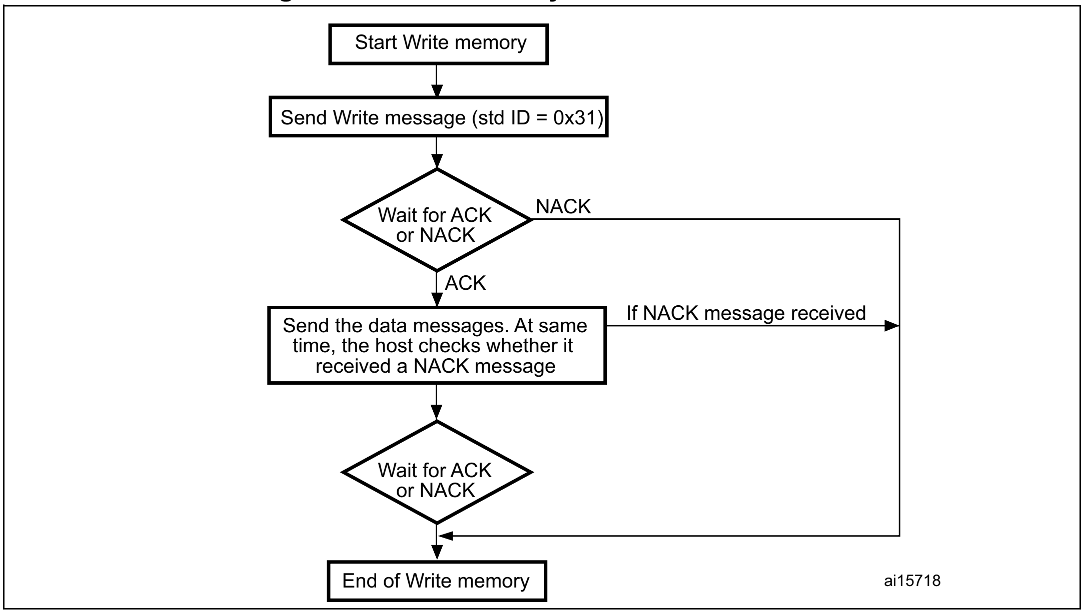
The maximum length of the block to be written for the STM32 is 256 bytes.

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, user must not overlap the memory used by the bootloader firmware.

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

Figure 16. Write Memory command: host side



Note: If the start address is invalid, the command is NACK-ed by the device.

The host sends the messages as follows:

Command message: Std ID = 0x31, DLC = 0x05, data[0] = 0xXX: MSB address,..., data[3] = 0xYY: LSB address, data[4] = N-1 (number of bytes to be written - 1), 0 < N ≤ 255).

Then the host sends N/8 message

Data message: Std ID = 0x31, DLC_1 = to 8, data = byte_11, … byte_18…

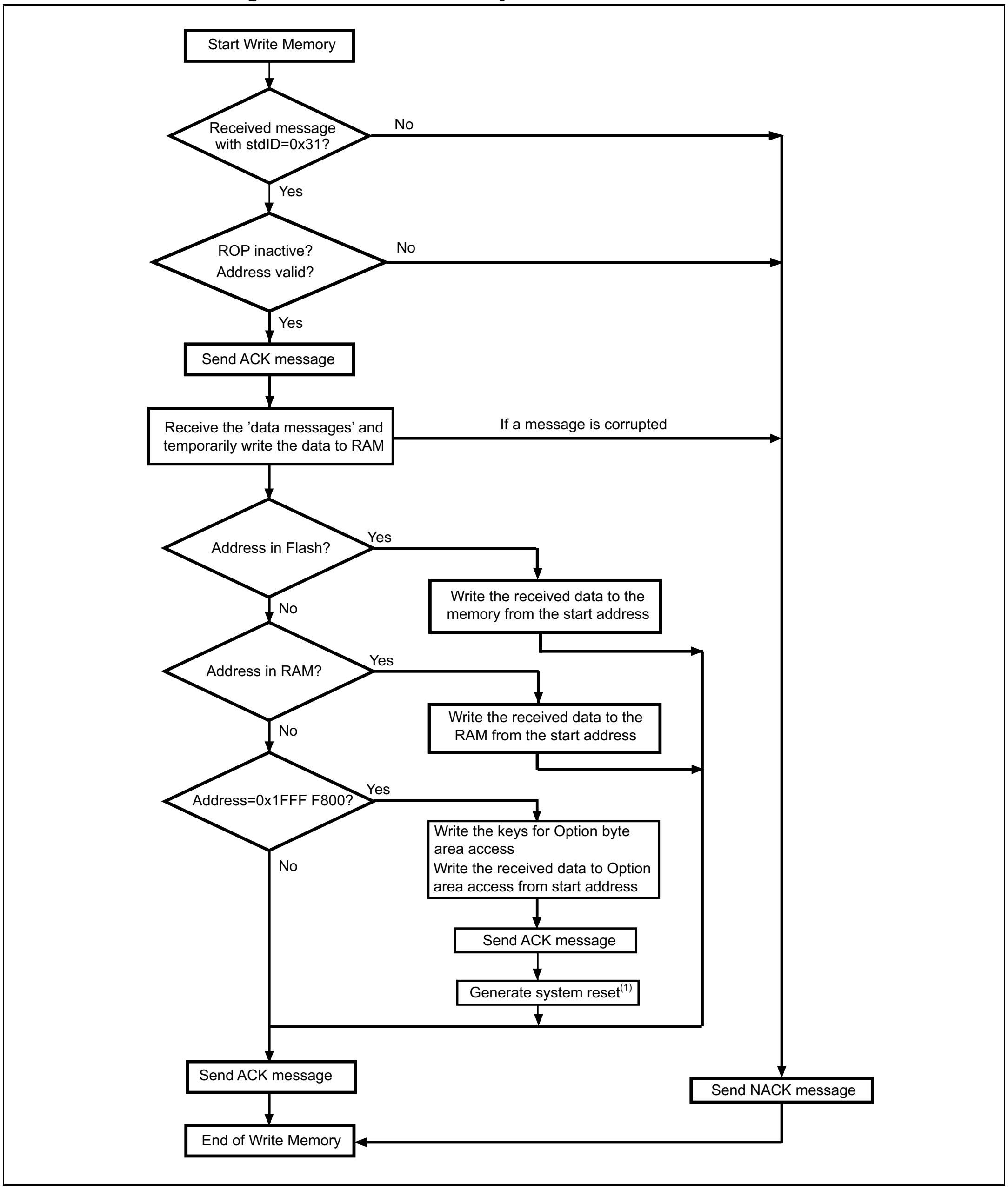
Data message_M: Std ID = 0x04, DLC_M = 1 to 8, data = byte_m1, …, byte_M8

Note: 1 DLC_1 + DLC_2 + ... DLC_M = 256 maximum

2 After each message the host receives the ACK or NACK message from the device

3 The bootloader does not check the standard ID of the data, so any ID from 0h to 0xFF can be used. It is recommended to use 0x04.

Figure 17. Write memory command: device side



1. System reset is called only for some STM32 BL (STM32F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

The STM32 sends messages as follows:

ACK message: Std ID = 0x31, DLC = 1, data[0] = ACK if the content of the command is correct else data[0] = NACK

After each received message, the device sends an ACK if the content of the command is correct else a NACK. However, as described in *Figure 17*, after receiving all the 'data messages' (N/8 messages) and temporarily write the data to RAM, if none of the messages content is corrupted, the bootloader writes the N bytes at the requested address (Flash memory, RAM or option byte), then if the write operation is successfully completed, the device sends an ACK message to the host.

In other terms, after sending the N/8 messages, host receives two successive ACK; the first is sent by the device if the last message of the N/8 is correctly received, and the second ACK is sent after writing correctly the N/8 message at the requested address.

3.8 Erase Memory command

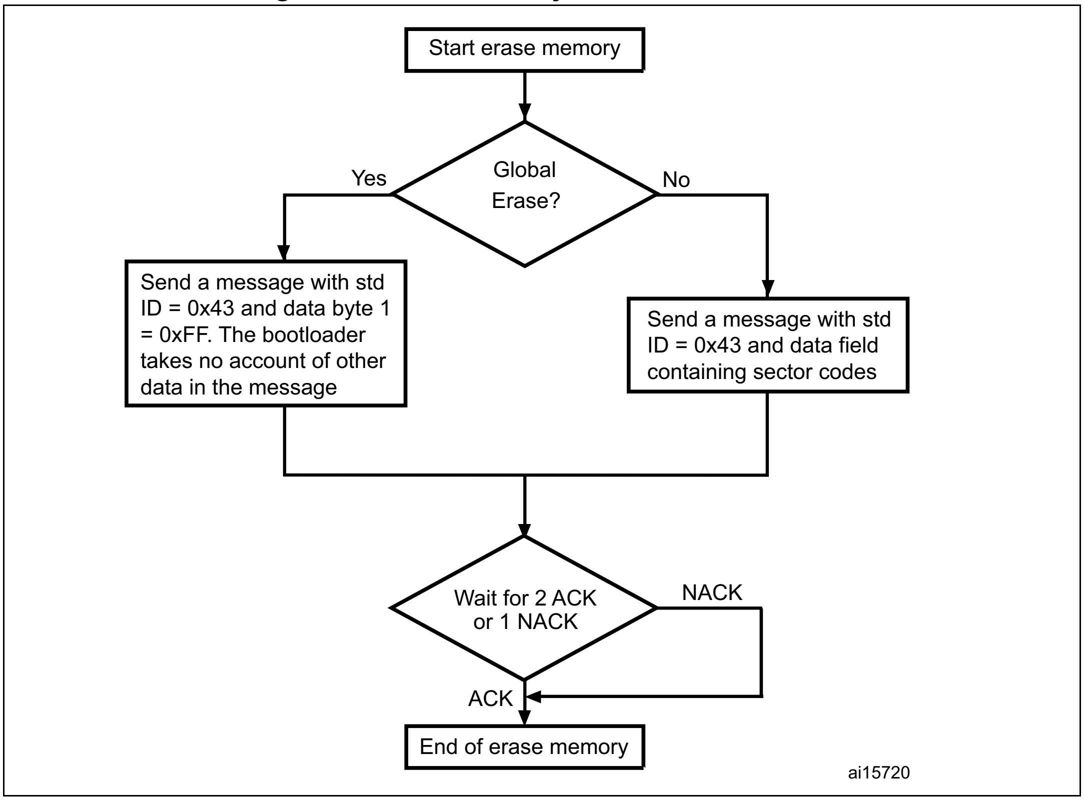
The Erase Memory command allows the host to erase Flash memory pages. When the bootloader receives the Erase Memory command and ROP is disabled, it transmits the ACK message to the host. After the transmission of the ACK message, the bootloader checks if data[0] is equal to 0xFF, if this is the case a global memory erase operation is started and when finished it sends ACK message. Otherwise (data[0] is different from 0xFF), the bootloader starts the memory page(s) erase as defined by the host, and after each page erase it sends an ACK or NACK message.

Erase Memory command specifications:

1. The bootloader receives a message containing N (the number of pages to be erased – 1). N = 255 is reserved for global erase requests. For $0 \leq N \leq 254$, 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 18. Erase Memory command: host side



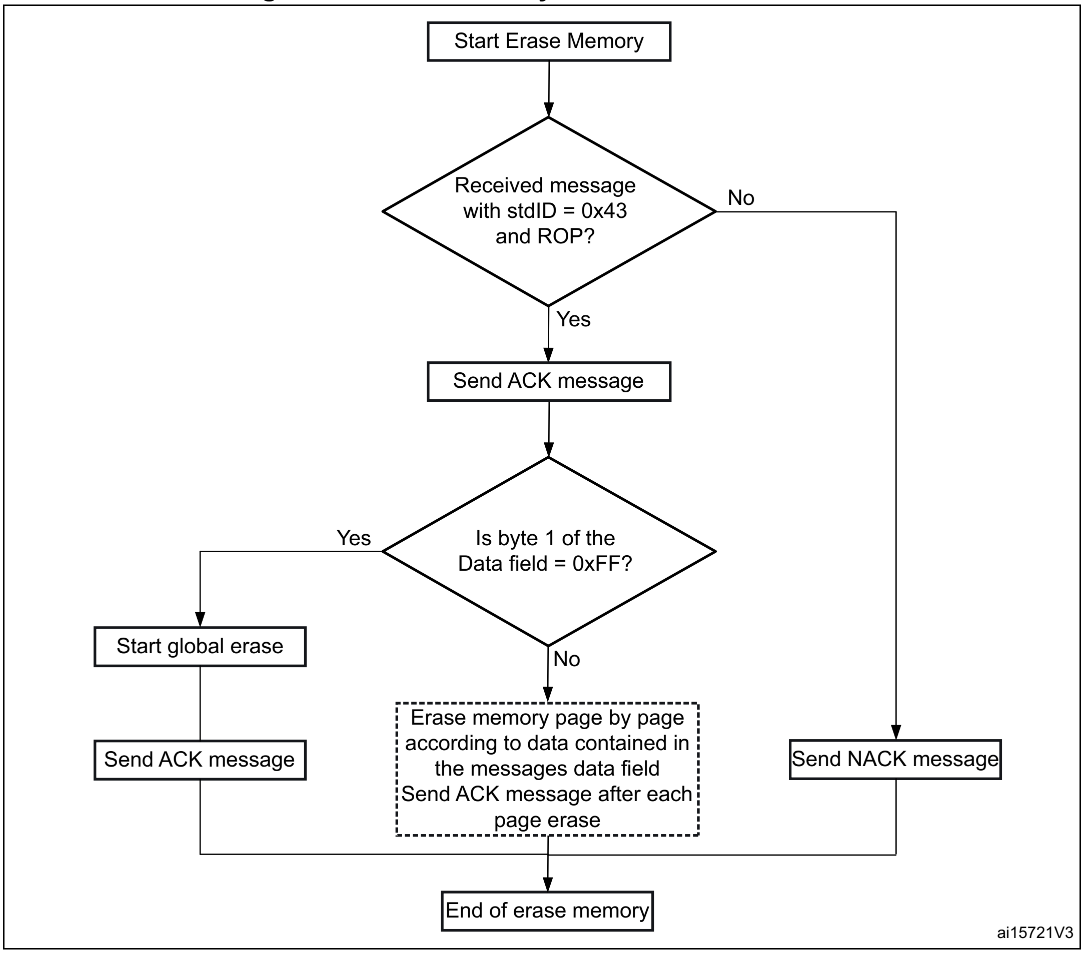
The host sends the message as follows:

The ID contains the command type (0x43):

- Total erase message: Std ID = 0x43, DLC = 0x01, data = 0xFF.
- Erase sector by sector message: Std ID = 0x43, DLC = 0x01 to 0x08, data = see product datasheet.

In case of page by page erase, after each message the host receives the ACK or NACK message from the device.

Figure 19. Erase Memory command: device side



The STM32 sends messages as follows:

ACK message: Std ID = 0x43, DLC = 1, data[0] = ACK if content of the command is correct and ROP is not active else data[0] = NACK.

3.9 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 message to the host if ROP is disabled else it transmits NACK.

After the transmission of the ACK byte, the bootloader waits to receive the Flash memory sector codes from the application.

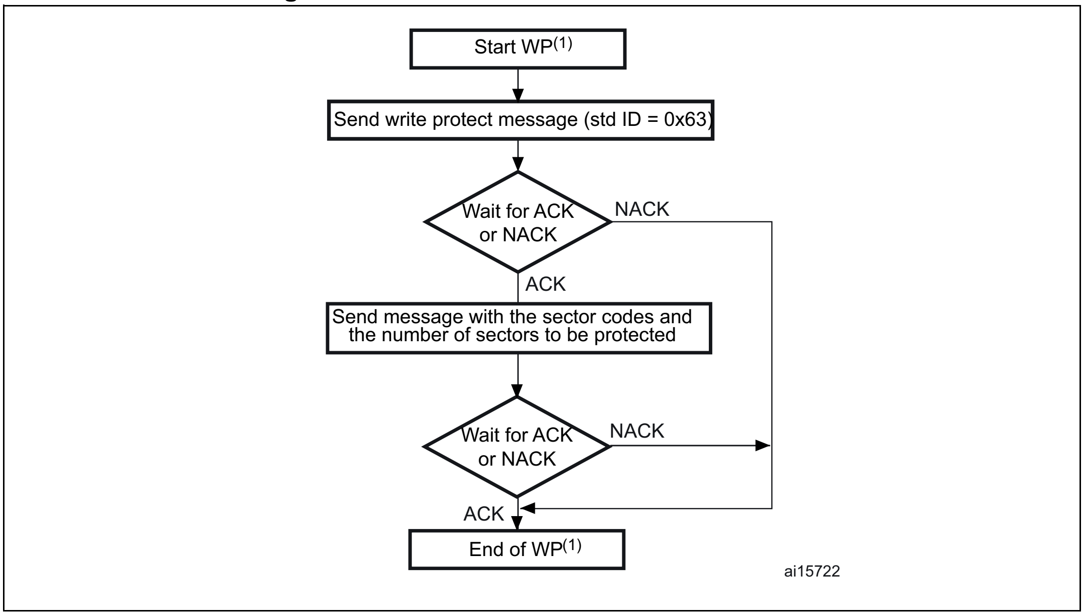
At the end of the Write Protect command, the bootloader transmits the ACK message and generates a system Reset to take into account the new configuration of the option byte.

Note: 1 Refer to Section 3.1 for more details about the sector size of the used device.

2 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.

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

Figure 20. Write Protect command: host side



1. WP = Write Protect.

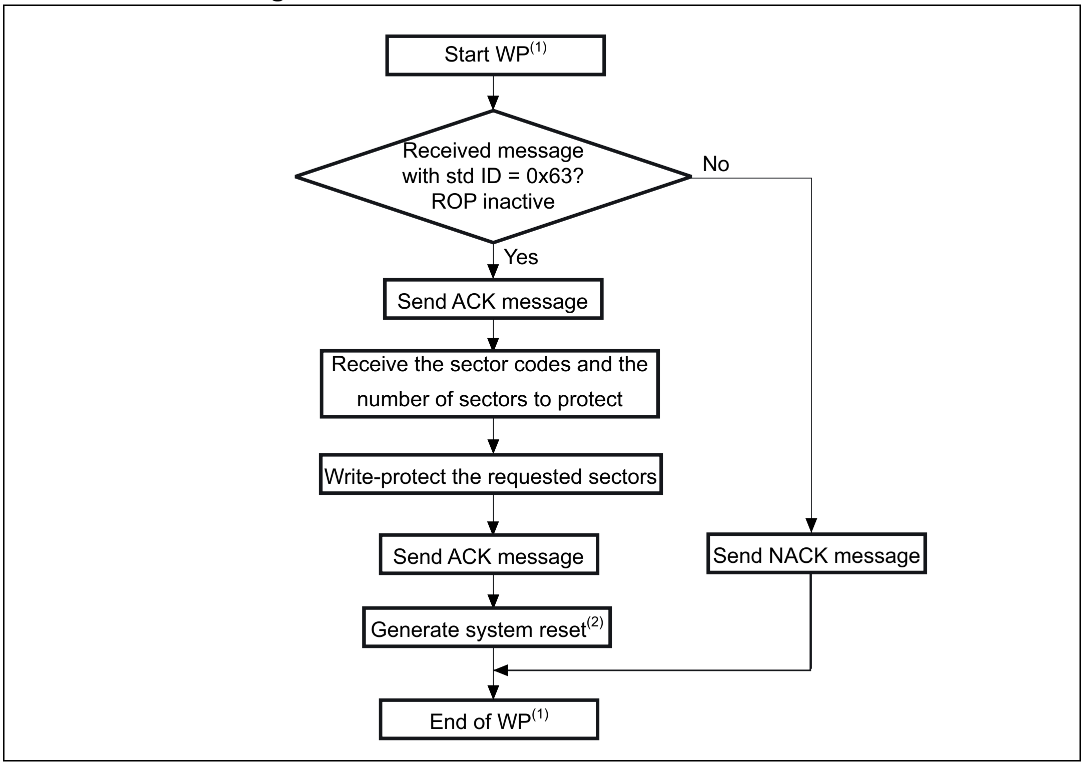
The host sends messages as follows:

Command message: Std ID = 0x63, DLC = 0x01, data[0] = N (where $0 < N \leq 255$).

Command message: Std ID = 0x63, DLC = 0x01..08, data[0] = N (where $0 < N \leq 255$).

After each message the host receives the ACK or NACK message from the device.

Figure 21. Write Protect command: device side



1. WP = Write Protect
2. System reset is called only for some STM32 BL (STM32F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

The STM32 sends messages as follows:

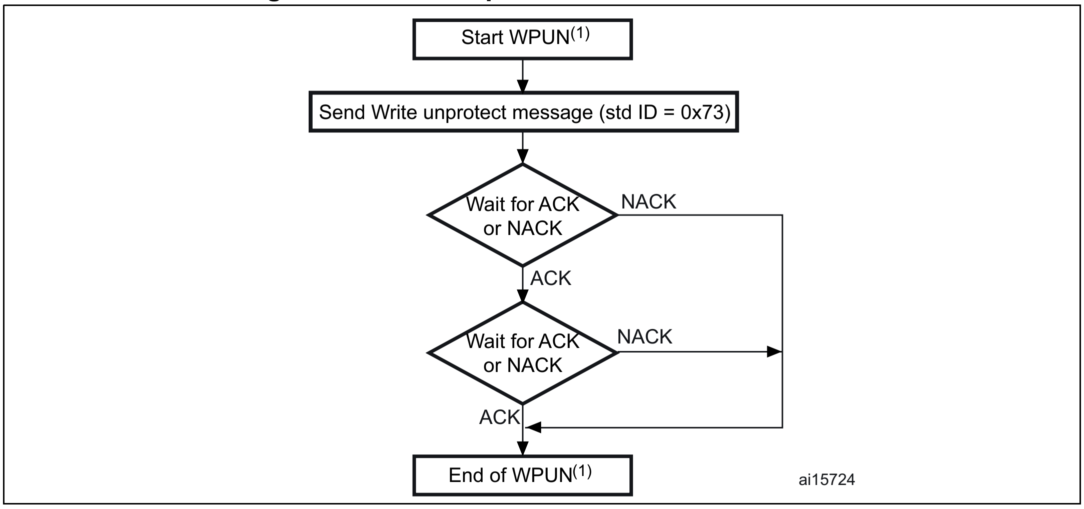
ACK message: Std ID = 0x63, DLC = 1, data[0] = ACK if the content of the command is correct and ROP is not active else data[0] = NACK.

3.10 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 message to the host if ROP is disabled, else it transmits NACK. After the transmission of the ACK message, the bootloader disables the write protection of all the Flash memory sectors.

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

Figure 22. Write Unprotect command: host side

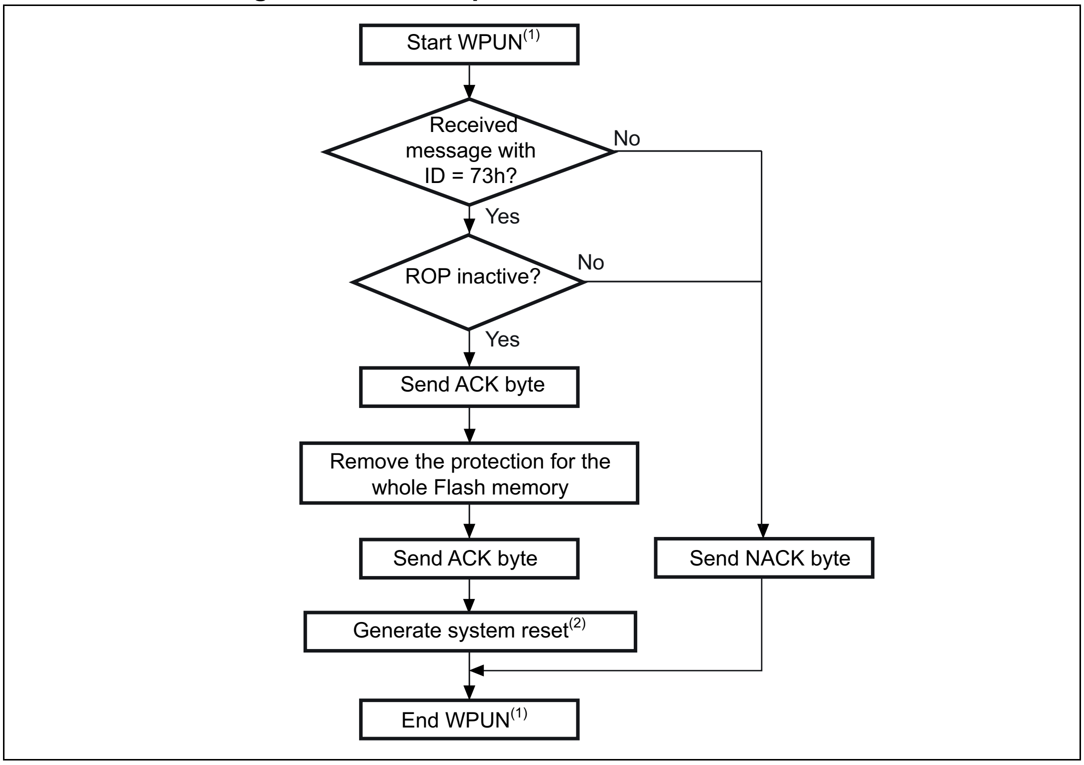


1. WPUN = Write Unprotect.

The host sends messages as follows:

Command message: Std ID = 0x73, DLC = 0x01, data = 00.

Figure 23. Write Unprotect command: device side



1. WPUN = Write Unprotect.
2. System reset is called only for some STM32 BL (STM32F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

The STM32 sends messages as follows:

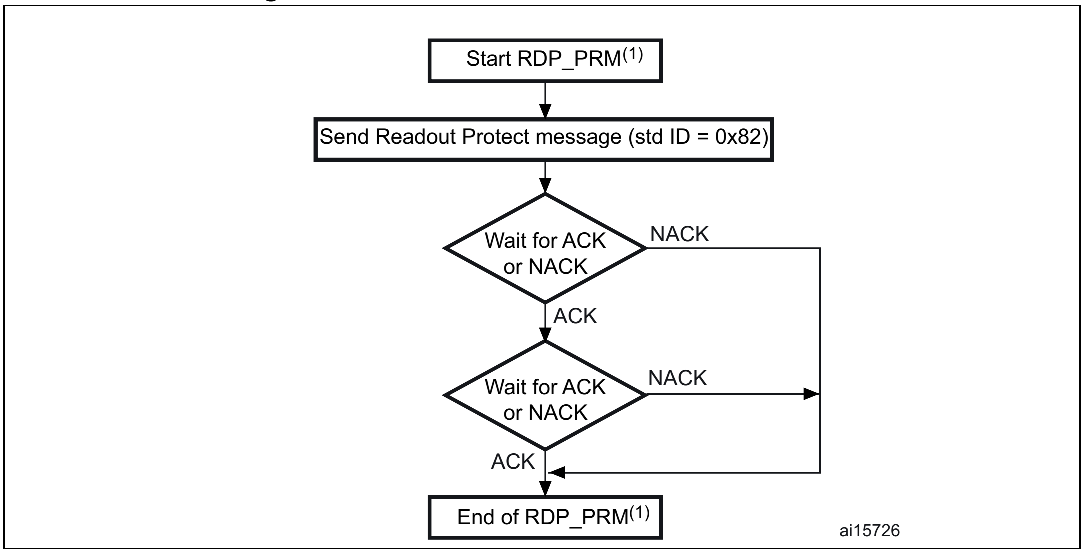
ACK message: Std ID = 0x73, DLC = 1, data[0] = ACK if the content of the command is correct and ROP is not active else data[0] = NACK.

3.11 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 message to the host if ROP is disabled else it transmits NACK. After the transmission of the ACK message, the bootloader enables the read protection for the Flash memory.

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

Figure 24. Readout Protect command: host side

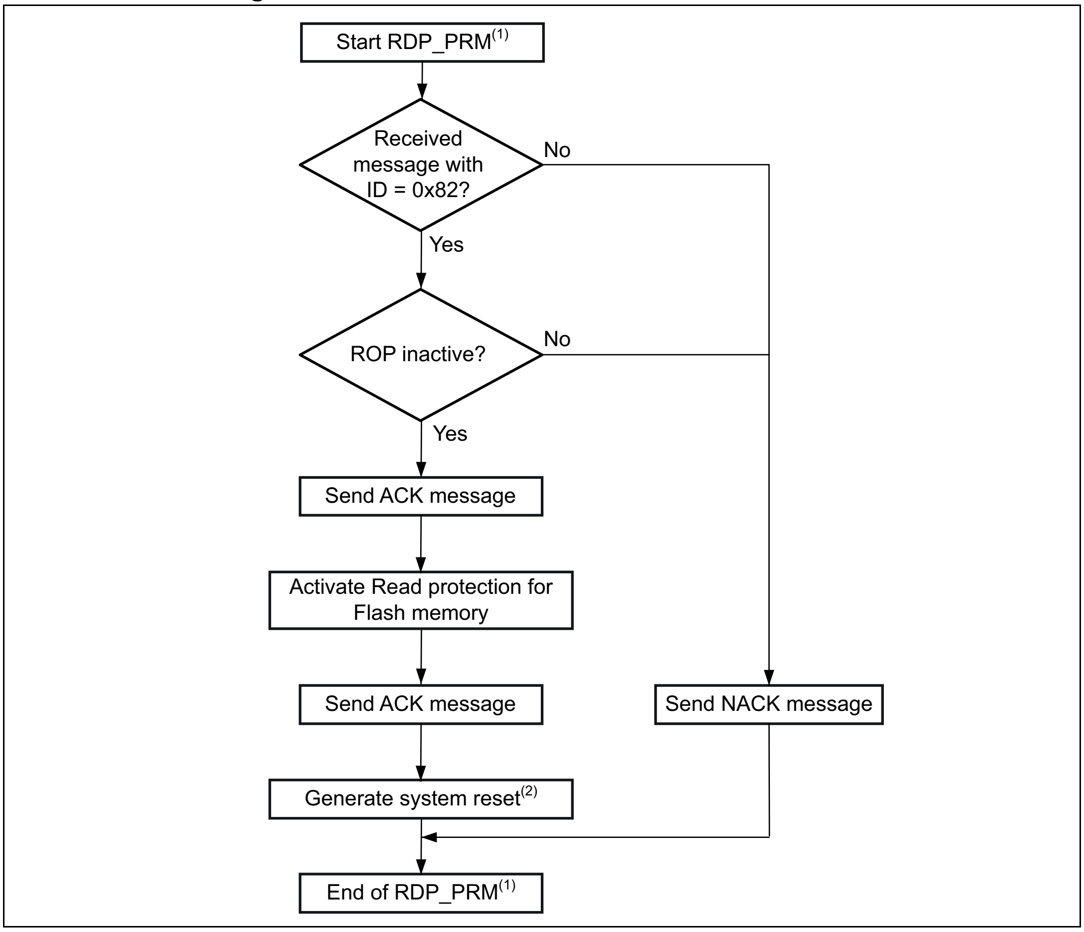


1. RDP_PRM = Readout Protect.

The host sends the messages as follows

Command message: Std ID = 0x82, DLC = 0x01, data[0] = 00.

Figure 25. Readout Protect command: device side



1. RDP_PRM = Readout Protect
2. System reset is called only for some STM32 BL (STM32F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

The STM32 sends messages as follows:

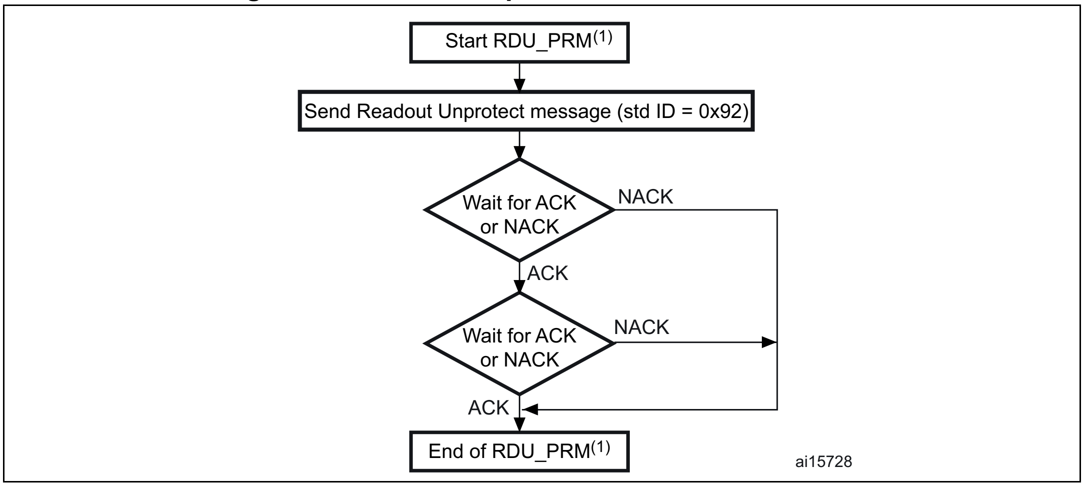
ACK message: Std ID = 0x82, DLC = 1, data[0] = ACK if the content of the command is correct and ROP is not active else data[0] = NACK.

3.12 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 message to the host. After the transmission of the ACK message, the bootloader erases all the Flash memory sectors and disables the read protection for the entire Flash memory. If the erase operation is successful, the bootloader deactivates the RDP.

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

Figure 26. Readout Unprotect command: host side

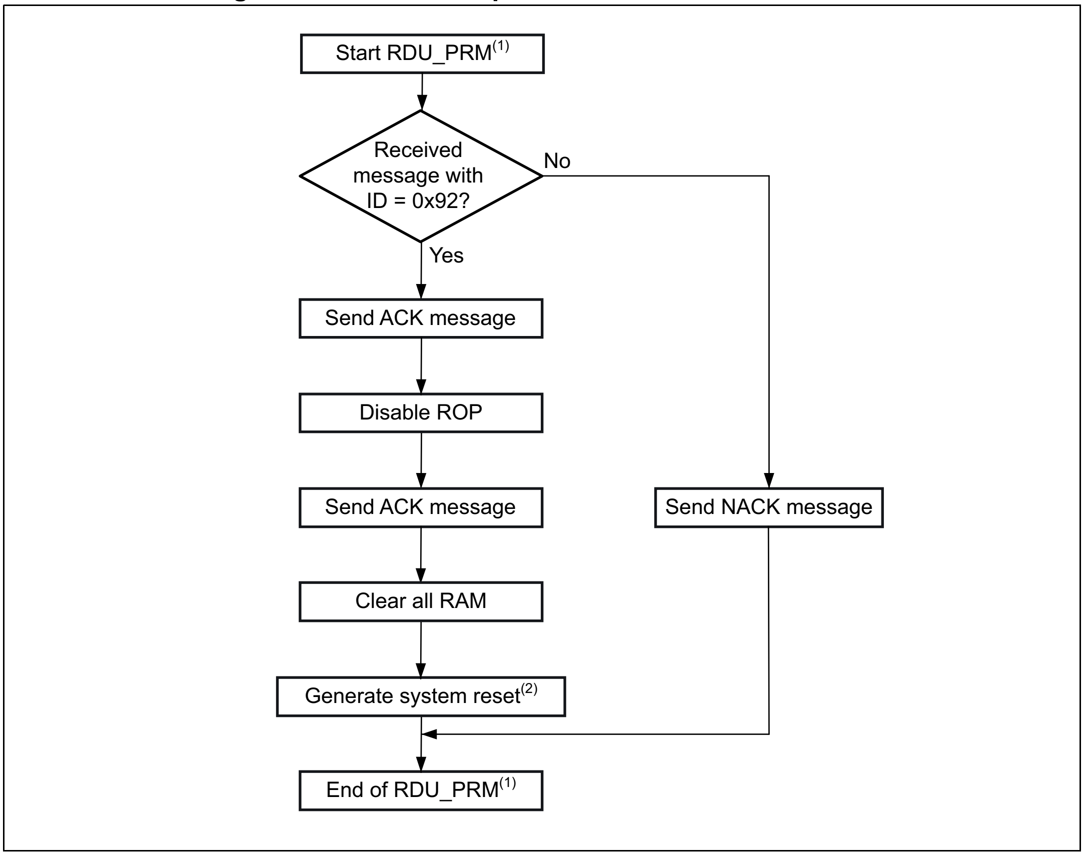


1. RDU_PRM = Readout Unprotect.

The host sends messages as follows

Command message: Std ID = 0x92, DLC = 0x01, data = 00.

Figure 27. Readout Unprotect command: device side



1. RDU_PRM = Readout Unprotect.
2. System reset is called only for some STM32 BL (STM32F2/F4/F7) and some STM32L4 (STM32L412xx/422xx, STM32L43xxx/44xxx, STM32L45xxx/46xxx) products. In all other STM32 products no system reset is called.

The STM32 sends messages as follows:

ACK message: Std ID = 0x92, DLC = 1, data[0] = ACK if the content of the command is correct and ROP is not active else data[0] = NACK.

4 Bootloader protocol version evolution

Table 3 lists the bootloader versions.

Table 3. Bootloader protocol versions

Version	Description
V2.0	Initial bootloader version.

5 Revision history

Table 4. Document revision history

Date	Revision	Changes
09-Mar-2010	1	Initial release.
15-Apr-2011	2	Updated *Figure 2: Check HSE frequency* and added *Note 1*.
22-Apr-2011	3	Update Std ID for message 2 in *Section 3.4: Speed command*.
24-Oct-2012	4	Updated *Chapter 3.2: Get Version & Read Protection Status command* *Figure 6: Get Version & Read Protection Status command: host side* *Chapter 3.7: Write Memory command* *Chapter 3.8: Erase Memory command* *Figure 19: Erase Memory command: device side* *Chapter 3.9: Write Protect command* *Figure 25: Readout Protect command: device side*
02-May-2014	5	Updated *Table 1: Applicable products* and *Table 2: CAN bootloader commands*. Removed Section dedicated to Device-dependent bootloader parameters. Updated *Section 3.5: Read Memory command* and *Section 3.7: Write Memory command*.
21-Oct-2016	6	Updated *Introduction*, *Section 1: Bootloader code sequence* and *Section 3.1: Get command*. Updated *Table 1: Applicable products*.
03-Dec-2019	7	Updated *Table 1: Applicable products*. Updated *Section 1: Bootloader code sequence*, *Section 3.6: Go command* and *Section 3.7: Write Memory command*. Updated *Figure 4: Get command: host side* and *Figure 17: Write memory command: device side*. Minor text edits across the whole document.
02-Jun-2021	8	Updated *Figure 17: Write memory command: device side*, *Figure 21: Write Protect command: device side*, *Figure 23: Write Unprotect command: device side*, *Figure 25: Readout Protect command: device side* and *Figure 27: Readout Unprotect command: device side* and added footnotes to them. Minor text edits across the whole document.