I3C protocol used in the STM32 bootloader

Introduction

This application note describes the I3C protocol used in the STM32 microcontroller bootloader, providing details on each supported command.

This document applies to STM32 products embedding bootloader version V14.x, as specified in the application note AN2606 STM32 system memory boot mode, available at www.st.com. These products are listed in Table 1 and referred to as STM32 throughout the document.

For more information about the I3C hardware resources and requirements for your device bootloader, refer to the already mentioned AN2606.

Table 1. Applicable products

Type	Product series
Microcontrollers	STM32H5 STM32H7 STM32U3

1 Bootloader code sequence

The I3C bootloader code sequence for STM32 microcontrollers, based on Arm®(a) core(s), is shown in Figure 1. Bootloader detection (device side).

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

Figure 1. Bootloader detection (device side)

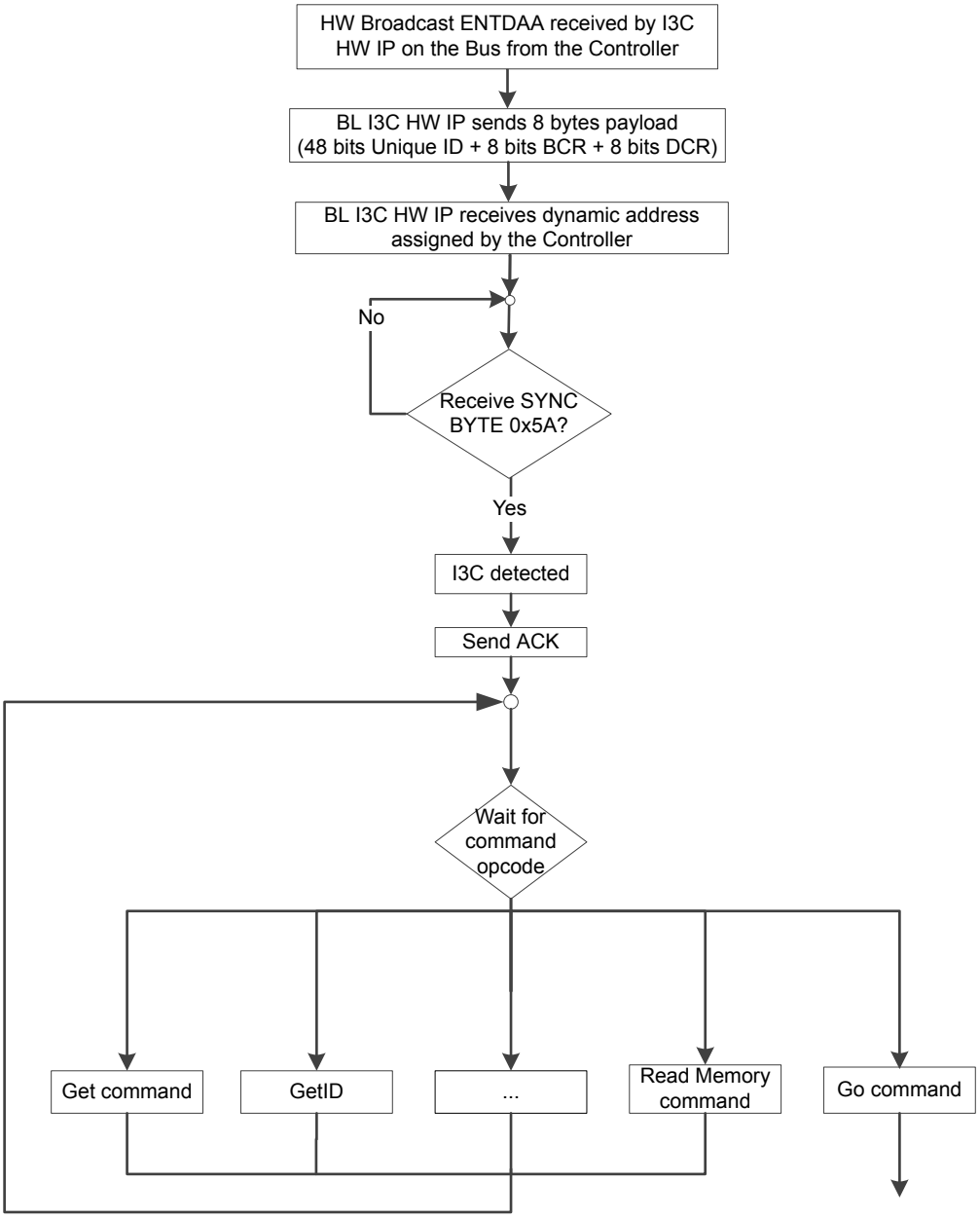
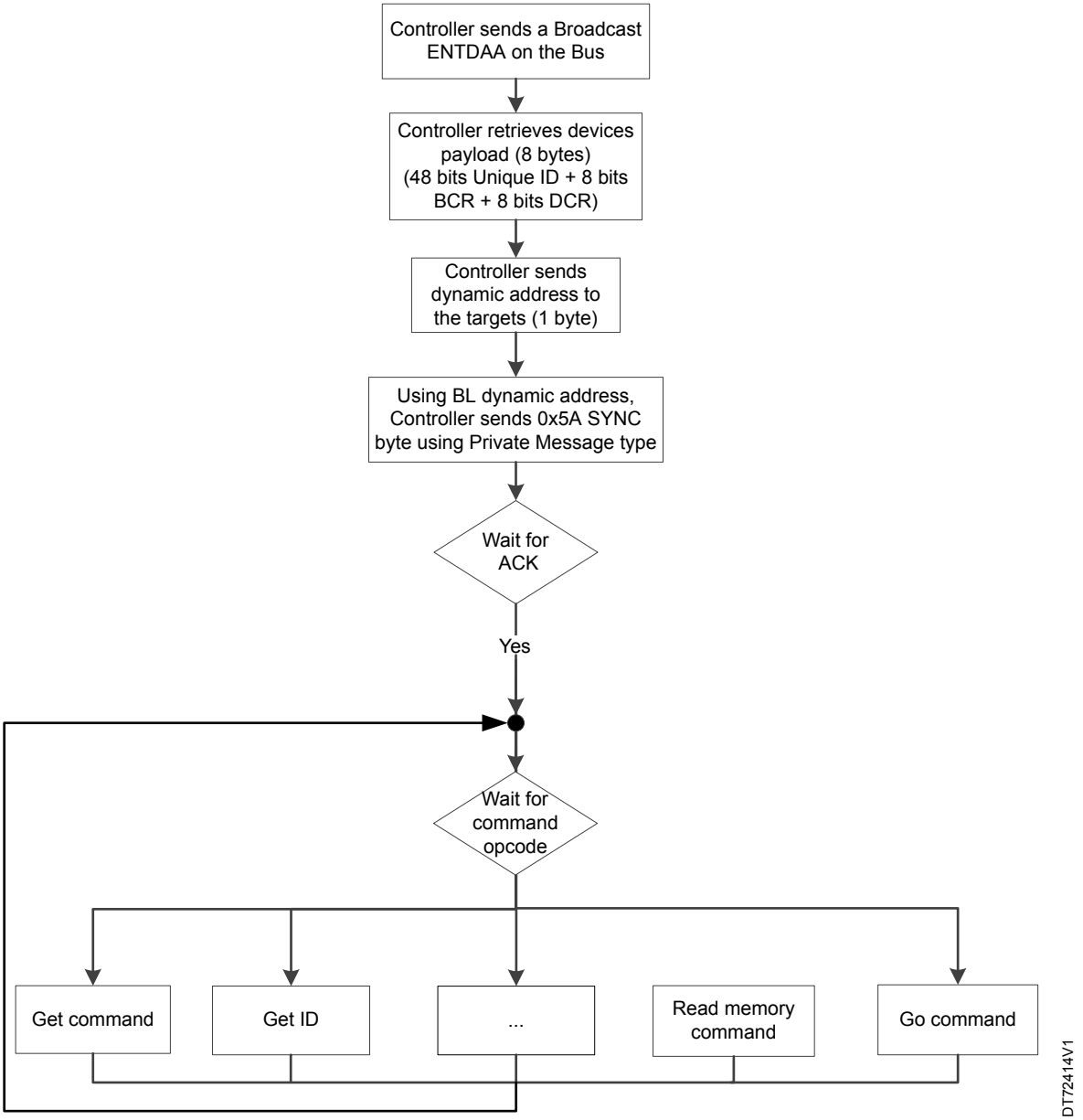


Figure 2. Bootloader detection (host side)



Once the system memory boot mode is entered and the STM32 device has been configured (for more details refer to AN2606), the I3C IP configured by the bootloader listens on the I3C_SDA pin to detect any controller activity on the bus. When the BL detects a controller broadcast and HW negotiation is established (dynamic address assigned to the BL I3C), the BL waits for a synchronization byte “0x5A” to know that the controller is targeting a communication with the bootloader. When detected, the I3C bootloader firmware begins receiving host commands.

2 I3C settings

2.1 Target (bootloader) settings

The BL initialization configures I3C as follows:

- Mode: Target mode
- Aval timing: 0x4E
- DMA Req RX: Disabled
- DMA Req TX: Disabled
- Status FIFO: Disabled
- DMA Req status: Disabled
- DMA Req control: Disabled
- IBI: Enabled
- Additional data after IBI acknowledged: 1 byte
- IBI configuration: Mandatory Data Byte (MDB)
- MIPI instance ID: 0xB (B stands for Bootloader)
- Device characteristics: 0x2
- All IT disabled except RXFNE (Receive FIFO Interrupt)
  - The RXFNE interruption is disabled after SYNC byte detection by the bootloader.

2.2 Controller (host) settings

The host initialization configures I3C as follows:

- Mode: Controller mode
- Clocks waive form: 0x00262726 (1 Mega)

Note: – *It is possible to reach 12.5 Mega by modifying this configuration to (0x00550908) but this depends on the HW quality and configuration. Other baudrates are also possible. 1 Mega here is an example.*

- Bus characteristics: 0x0011004E
- DMA Req RX: Disabled
- DMA Req TX: Disabled
- Status FIFO: Disabled
- DMA Req status: Disabled
- DMA Req control: Disabled
- IBI: Enabled
- Additional data after IBI acknowledged: 1 byte
- IBI configuration: Mandatory Data Byte (MDB)
- Message type: Private message (using dynamic address chosen)
- Send method
  - Data Generate Stop condition after sending a message
  - ACK/NACK: IBI used
- All IT disabled

3 Bootloader command set

The supported commands are listed in Table 2. I3C bootloader commands and described in this section.

Table 2. I3C bootloader commands

Command	Command code	Command description
Get	0x00	Gets the version and the allowed commands supported by the current version of the protocol
Get Version	0x01	Gets the bootloader version
Get ID	0x02	Gets the chip ID
Read Memory	0x11	Reads up to 2048 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 2048 bytes to the RAM or flash memory starting from an address specified by the application
Erase	0x44	Erases from one to all the flash memory sectors
Special	0x50	Generic command that allows adding new features depending on the product constraints, without adding a new command for every feature
Extended Special	0x51	Generic command that allows the user to send more data compared to the Special command
Write Protect	0x63	Enables the write protection for some sectors
Write Unprotect	0x73	Disables the write protection for all flash memory sectors
Readout Protect	0x82	Enables the read protection
Readout Unprotect	0x92	Disables the read protection

Note: Depending on the project some commands can be omitted. The Get command is adapted to give only the commands that are used.

Communication safety

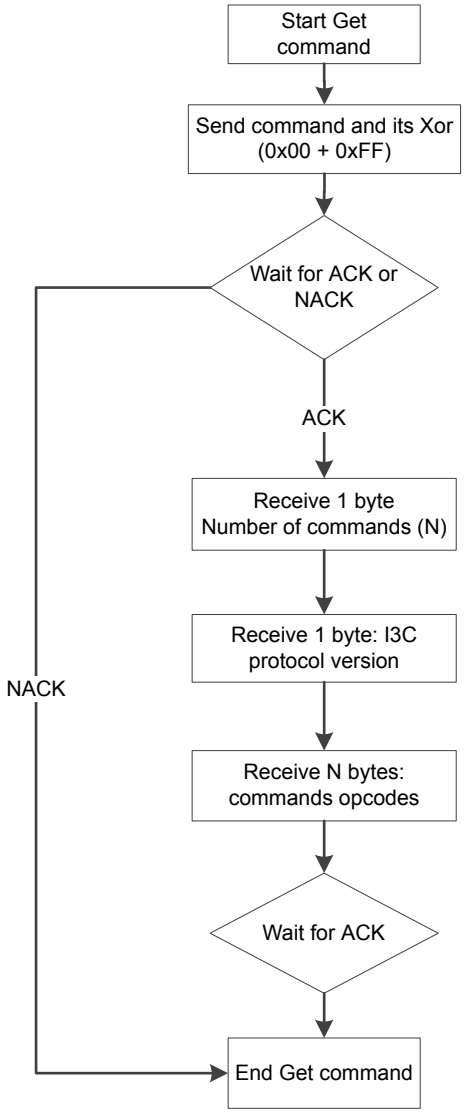
Each packet is either accepted (ACK answer) or discarded (NACK answer):

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

3.1 Get command

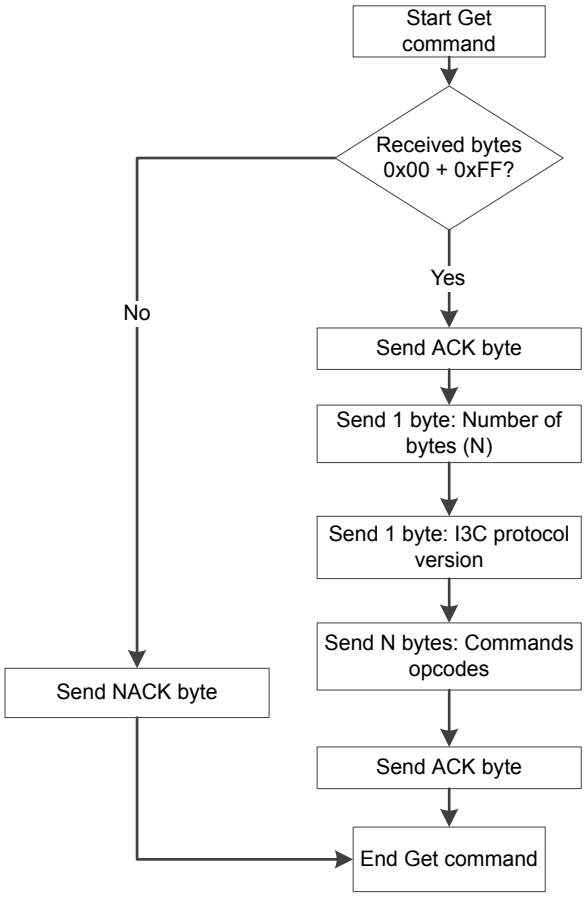
The Get command allows the host to get the version of the I3C protocol and the supported commands. When the bootloader receives the Get command, it transmits the I3C protocol version and the supported command codes to the host.

Figure 3. Get command (host side)



Based on the host command, the device answers the following:

Figure 4. **Get command (device side)**



The data is received on the following order:

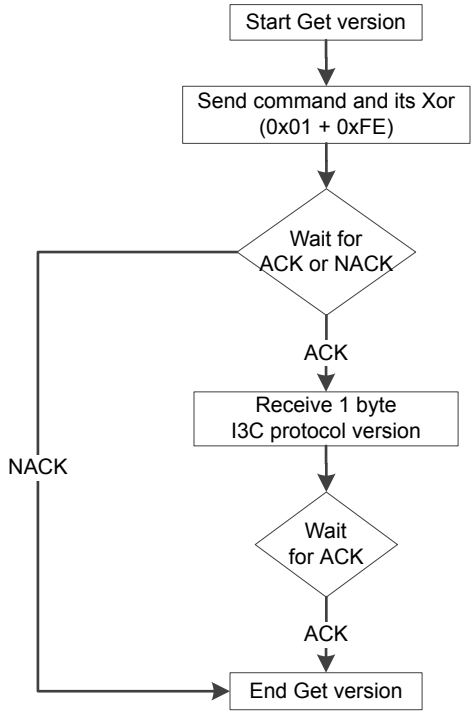
1. Byte 1: ACK
2. Byte 2: N = 10 = Number of commands
3. Byte 3: I3C protocol version 0x10 = Version 1.0
4. Byte 4: 0x00 - Get command
5. Byte 5: 0x01 - Get version
6. Byte 6: 0x02 - Get ID
7. Byte 7: 0x11 - Read Memory command
8. Byte 8: 0x21 - Go command
9. Byte 9: 0x31 - Write Memory command
10. Byte 10: 0x44 - Erase command
11. Byte 11: 0x50 - Special command
12. Byte 12: 0x63 - Write Protect command
13. Byte 13: 0x73 - Write Unprotect command
14. Byte 14: ACK

This is an example as the number of commands and their opcodes depend on the project.

3.2 Get version command

The Get version command is used to get the I3C protocol version. When the bootloader receives the command, it transmits one byte with protocol version to the host.

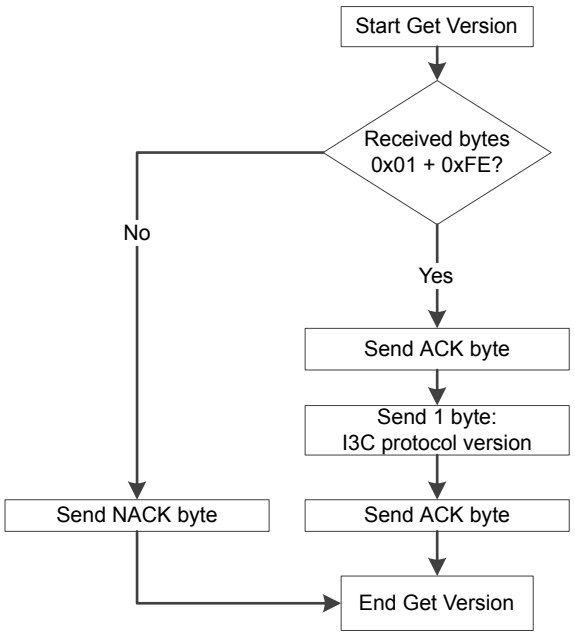
Figure 5. Get version command (host side)



The STM32 sends the bytes as follows:

- Byte 1: ACK
- Byte 2: Bootloader version (0 < Version < 255) (for example, 0x10 = Version 1.0)
- Byte 3: ACK

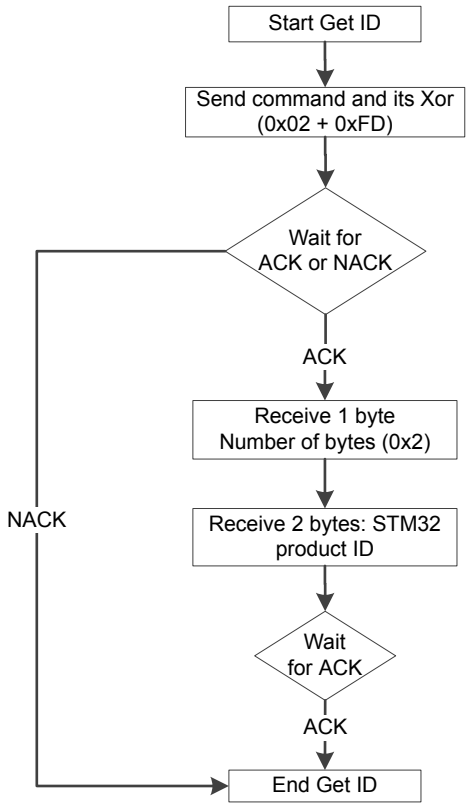
Figure 6. Get version command (device side)



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.

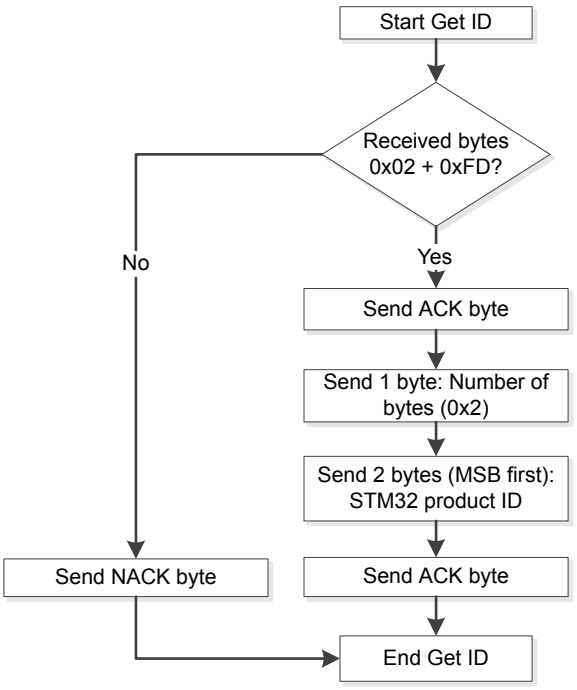
Figure 7. Get ID command (host side)



The STM32 device sends the bytes as follows:

- Byte 1: ACK
- Byte 2: Number of bytes to send (Fixed = 0x2)
- Bytes 3-4: PID (product ID)
  - Byte 3 = MSB
  - Byte 4 = LSB
- Byte 5: ACK

Figure 8. **Get ID command (device side)**



3.4 Read memory command

The Read memory command is used to read data from any valid memory address: RAM, flash memory and in the information blocks (system memory or option byte areas). Read memory is only possible when no protection is set and when the address to read is valid.

The bootloader SW then individually checks if:

- The ID of the command is correct or not
- The protection is disabled or enabled (protection status depends on the product architecture)

If these data are not valid, the bootloader transmits a NACK and leaves the command. Otherwise, it transmits an ACK and then waits to receive the address to read from and its checksum. ACK is transmitted only if the address is valid, and its checksum is ok. If not, NACK is sent, and the command is terminated.

After the ACK, the bootloader waits to receive the data size to be written and its checksum.

- If the checksum is not OK, size received is null, with a size exceeding 2048 or the readable valid area, NACK is sent and the Read memory command is terminated
- If all the conditions above are correct, the BL calculates the number of bytes to read and if a “Loop” is requested to read more or not.
  - Data size and checksum are received on 3 Bytes (data[0], data[1], data[2])

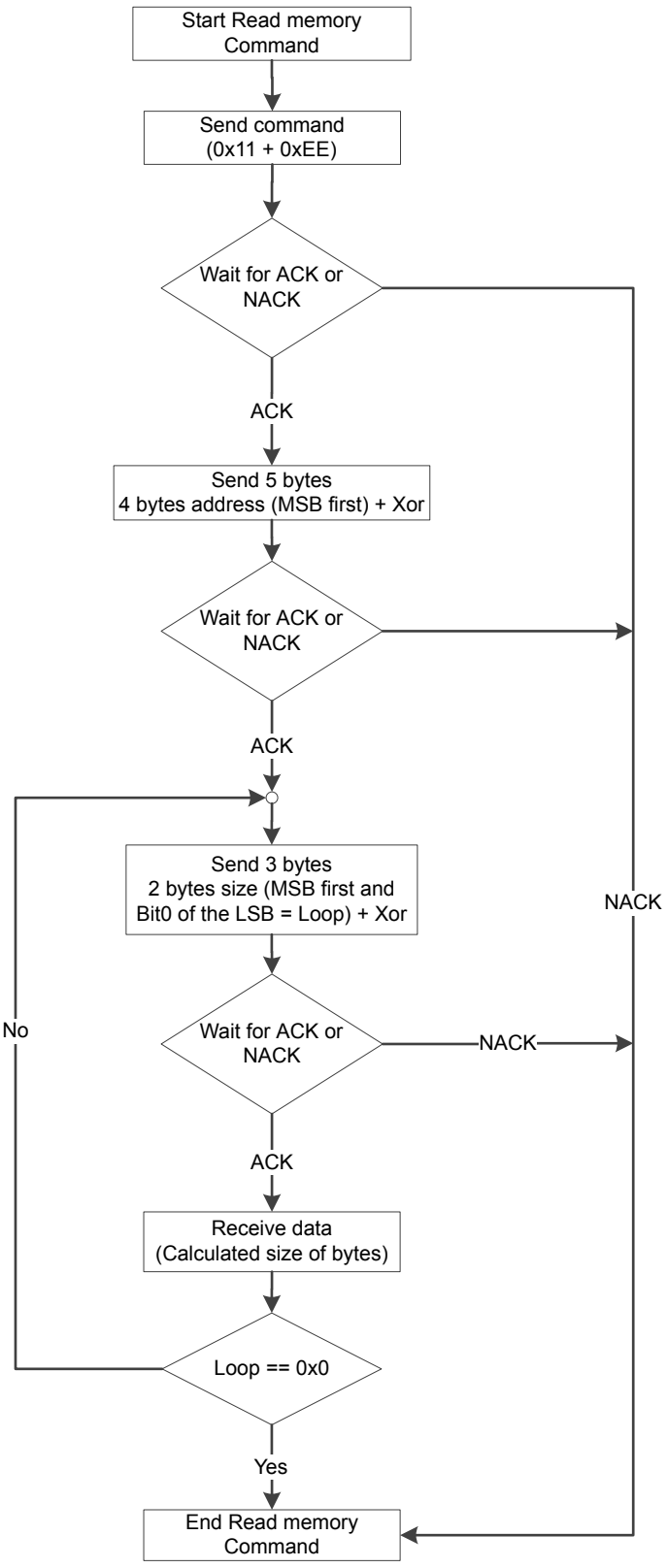
```
Bit number - |15|14|13|12|11|10|9|8|7|6|5|4|3|2|1| 0  |
Content    - |         data[0]     |      data[1]     |
Content    - |     N =(2x number of bytes)       |loop|
```

  - Loop = 1 means that there is still data to send to the host.
    - BL waits to receive the new chunk size and its checksum and repeats the above operation.
  - Loop = 0 means that this is the last chunk of data to read.

Note:

- *The maximum length of the block that can be read for the STM32 is 2048 bytes.*
- *To read more than 2048 bytes, it is not necessary to send the command multiple times. Use the “Loop” feature described above instead.*

Figure 9. Read memory (host side)

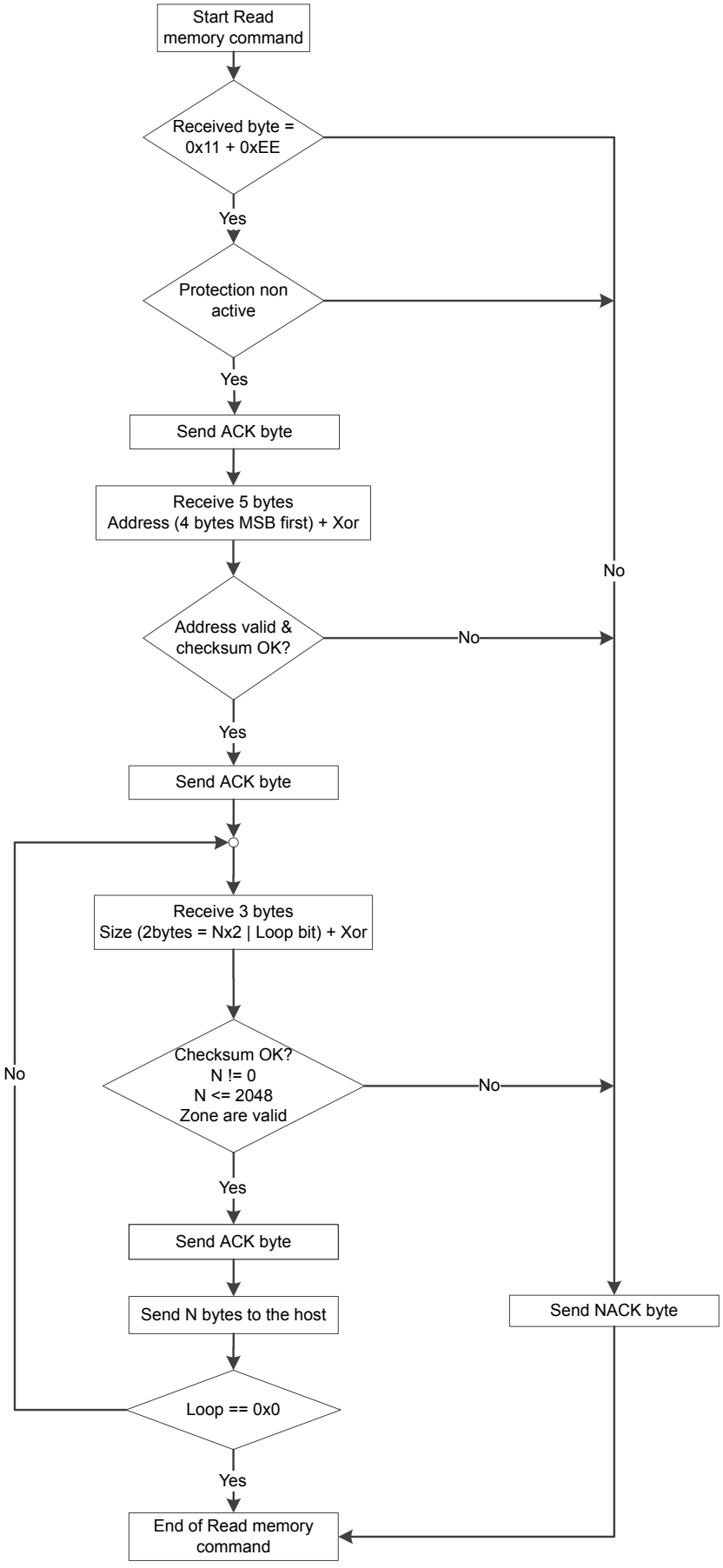


The host sends bytes to the STM32 as follows:

1. Bytes 1-2: 0x11 + 0xEE
2. Wait for ACK or NACK
3. Bytes 3-6: Start address (byte 3: MSB, byte 6: LSB)
4. Byte 7: Checksum: XOR (byte 3, byte 4, byte 5, byte 6)
5. Wait for ACK or NACK
6. Bytes 8-9: The number of bytes to be read (2 bytes 8: MSB, byte 9: LSB)
   - N = (Byte[8] << 8) | Byte[9])
   - Loop = N & 0x1
   - number of bytes = (N >> 1)
7. Byte 10: Checksum: XOR (byte 8, byte 9)
8. Wait for ACK or NACK
9. Byte 11...number of bytes: Data received from the device
10. If loop = 1 go to “6.”

Note: Receiving a NACK terminates the command.

Figure 10. Read memory (device side)



3.5 Go command

The Go command is used to execute the downloaded code or any other code by branching to an address specified by the application. When the bootloader receives the Go command, it starts if the message contains the following valid information:

- Opcode of the command is correct or not
- Protection is disabled or enabled

Note: – Note that the protection depends on the product (RDP, HDPL...)

- Address to jump is valid or not

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

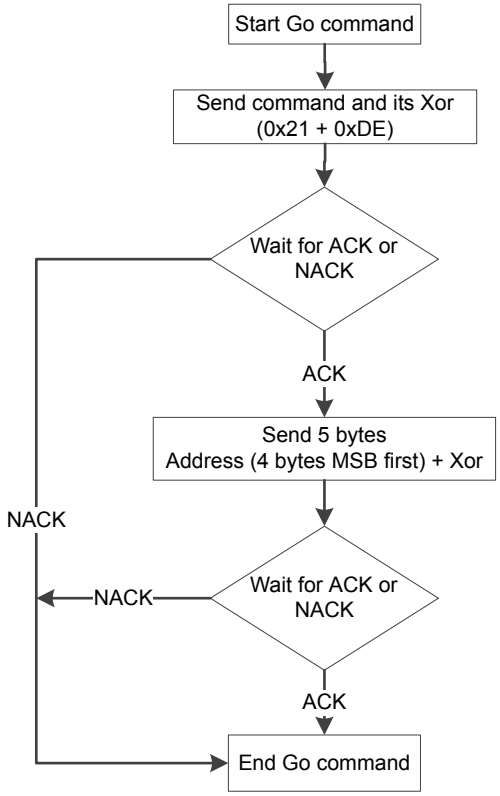
After sending an ACK message to the application, the bootloader firmware:

- Resets the registers of the peripherals used by the bootloader to their default values
- Initializes the user application's main stack pointer
- Jumps to the memory location programmed in the received 'address + 4' "Address +4": 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. The host should send the base address where the application to jump to is programmed.

Note:

- *The Jump to the application works only if the user application sets the vector table correctly to point to the application address.*
- *The valid addresses for the Go command are in RAM or flash memory. All other addresses are considered not valid and are NACK-ed by the device.*
- *Not all addresses in the RAM are considered as valid, application to jump to must consider an offset to avoid overlapping with the first RAM memory used by the bootloader firmware.*

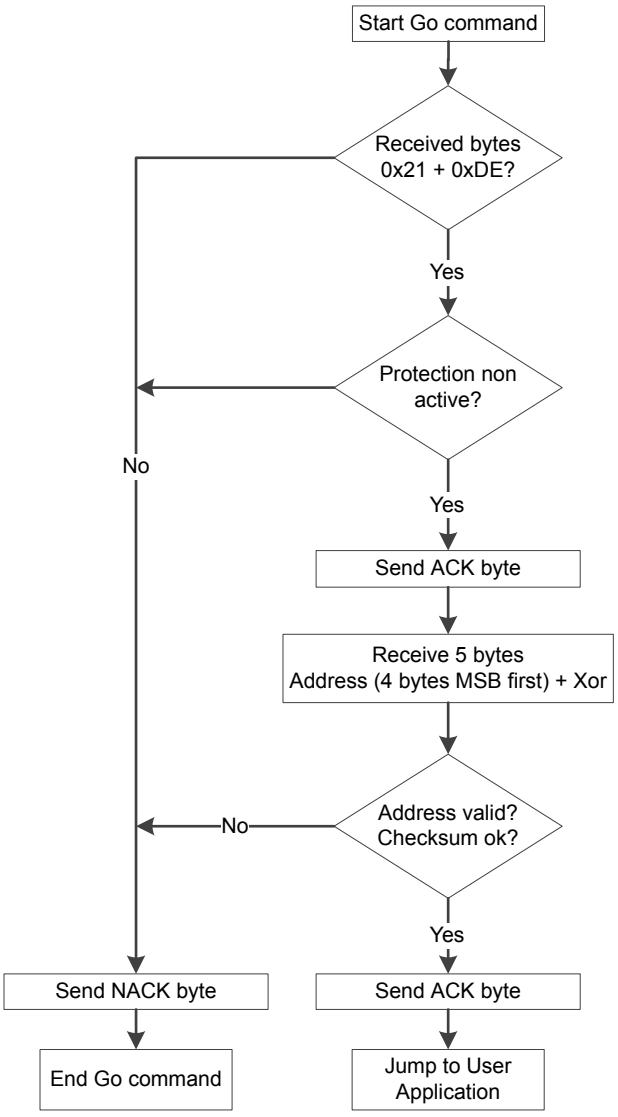
Figure 11. Go command (host side)



The host sends bytes to the STM32 as follows:

1. Byte 1: 0x21
2. Byte 2: 0xDE
3. Wait for ACK or NACK
4. Byte 3 to byte 6: start address
5. Byte 7: checksum: XOR (byte 3, byte 4, byte 5, byte 6)
6. Wait for ACK or NACK

Figure 12. Go command (device side)



3.6 Write memory command

The Write memory command is used to write data to any valid memory address of RAM, Flash OTP, or Option byte area. When the bootloader receives the Write memory command, it starts if the following information are valid:

- Opcode of the command and its checksum are correct
- Protection is disabled

If these data are not valid, the bootloader transmits a NACK and leaves the command. Otherwise, it transmits an ACK and then waits to receive the address to write to and its checksum. ACK is transmitted only if the address is valid, and its checksum is ok. If not, NACK is be sent, and command terminated.

Note: For the Option byte area, only the base address of the Option byte area is accepted to avoid writing inopportunely in this area.

After the ACK, the bootloader waits to receive the data size to be written and its checksum

- If the checksum is not OK, size received is null, with a size exceeding 2048 or the readable valid area, NACK is sent and the Write memory command is terminated
- If all the conditions above are correct, the BL calculates the number of bytes to write and if a “Loop” is requested to write more or not.
  - Data size and checksum are received on 3 Bytes (data[0], data[1], data[2])

```
Bit number - |15|14|13|12|11|10|9|8|7|6|5|4|3|2|1| 0  |
Content    - |          data[0]    |      data[1]       
Content    - |     N = (2x number of bytes)      |loop|
```

  - Loop = 1 means that there are still data to retrieve from the host
    - BL waits to receive the new chunk size and its checksum and repeats the above operation
  - Loop = 0 means that this is the last chunk of data to write

After each write operation, ACK is sent to the host.

In summary, the host sends the data to the BL as follows:

1. Byte 1: 0x31
2. Byte 2: 0xCE
3. Wait for ACK (if NACK received, command is terminated)
4. Byte 3 to byte 6: Start address
   - Byte 3: MSB
   - Byte 6: LSB
5. Byte 7: Checksum: XOR (byte 3, byte 4, byte 5, byte 6)
6. Wait for ACK (if NACK received, command is terminated)
7. Bytes 8-9: The number of bytes to be read (2 bytes 8: MSB, byte 9: LSB)
   - N = (Byte[8] << 8) | Byte[9])
   - Loop = N & 0x1
   - number of bytes = (N >> 1)
8. Byte 10: Checksum byte: XOR (Byte8, byte 9)
9. Wait for ACK (if NACK received, command is terminated)
10. Byte 11 to Byte (N): data to write and its Xor
    - If the Xor is ok, BL writes the data
11. Wait for ACK (if NACK received, command is terminated)
    - If not last chunk (Loop = 1), go to “7”
    - If last chunk (Loop = 0), command is terminated

Note:

- *The maximum length of the block that can be written for the STM32 is 2048 bytes.*
- *To write more than 2048 bytes, it is not necessary to send the command multiple times. Use the “Loop” feature described above instead.*
- *No error is returned when performing write operations on write protected sectors.*

Figure 13. **Write memory command (host side)**

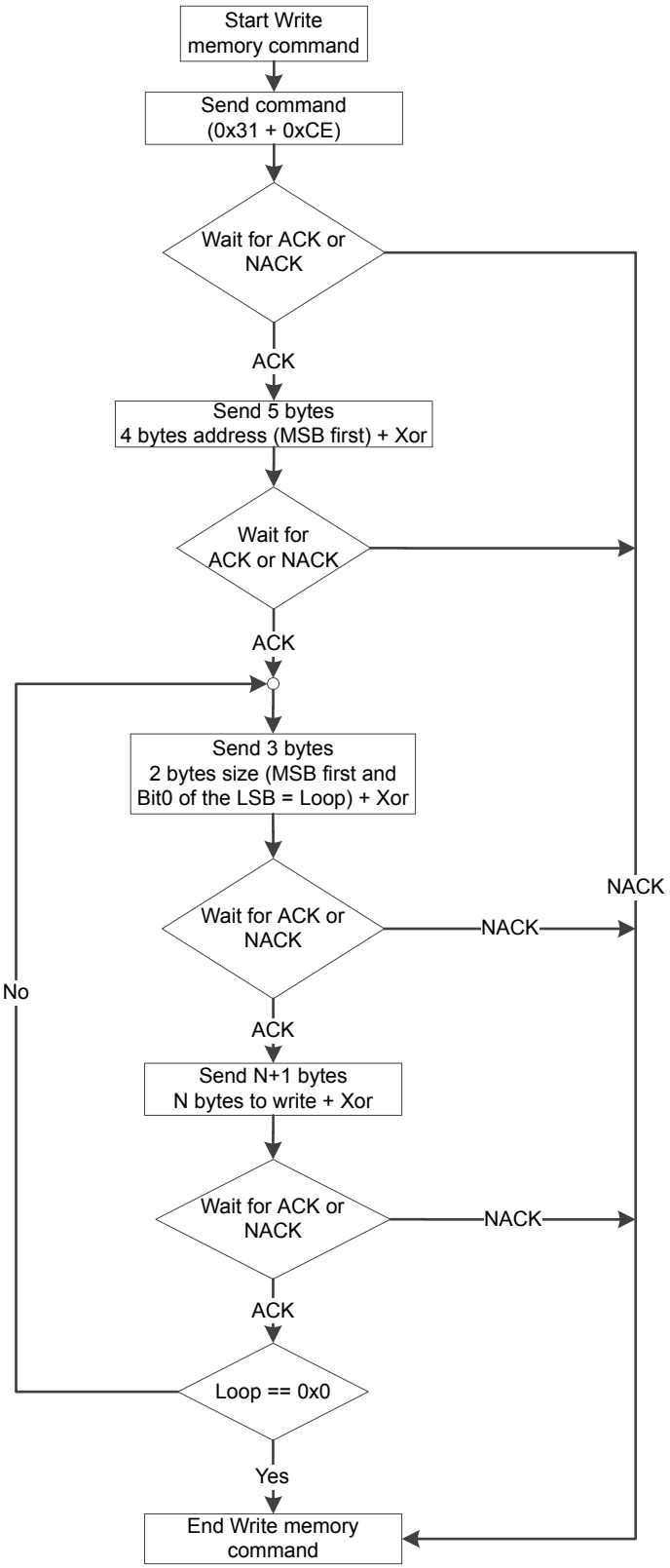
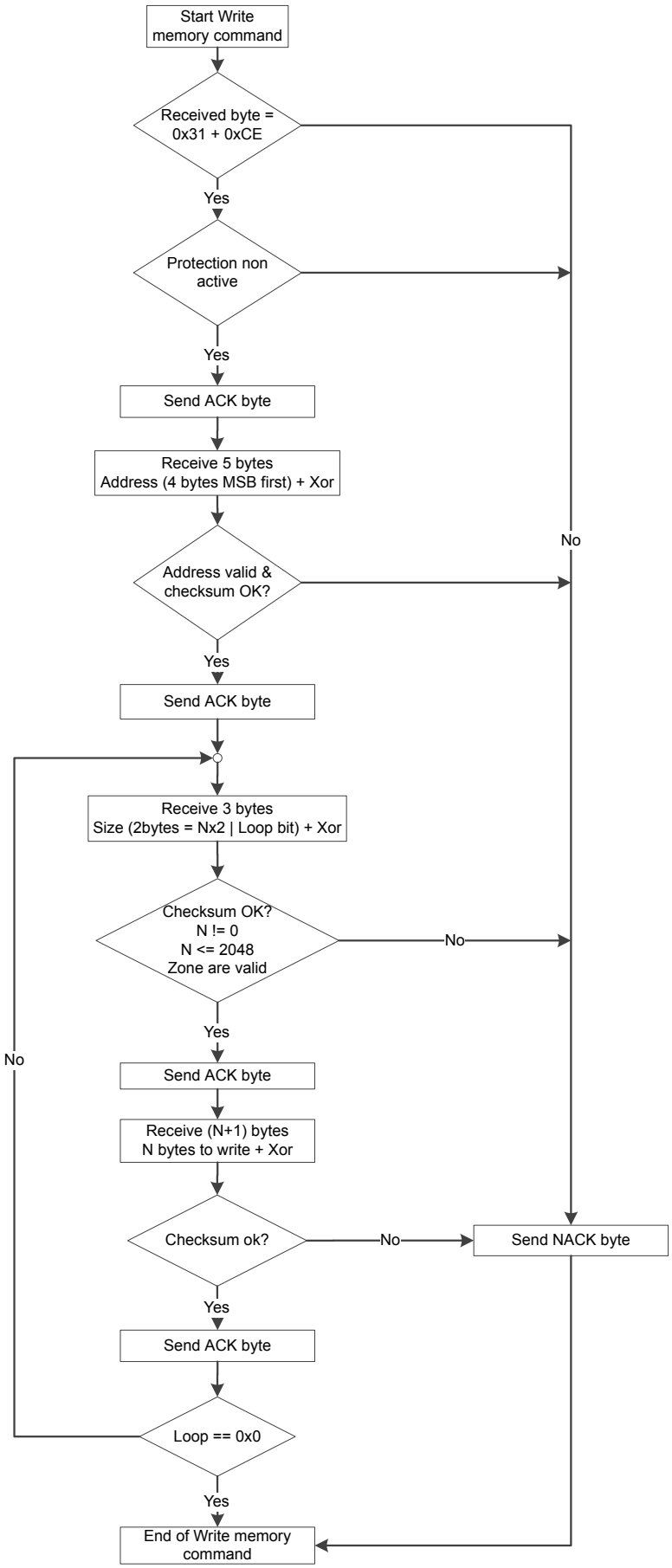


Figure 14. Write memory command (device side)



3.7 Erase memory command

The Erase memory command allows the host to erase flash memory pages. When the bootloader receives the Erase memory command and the protection is disabled, it transmits the ACK message to the host.

After the transmission of the ACK message, the bootloader checks the first 2 bytes data received on a MSB format and constructs the “PageNumber”. Depending on the value of the “PageNumber”, the required erase operation is executed:

- PageNumber = 0xFFFF: Mass erase is requested
- PageNumber = 0xFFFE: Bank1 erase is requested (if supported)
- PageNumber = 0xFFFD: Bank2 erase is requested (if supported)
- PageNumber: if different from the above values, it represents the number of pages to erase

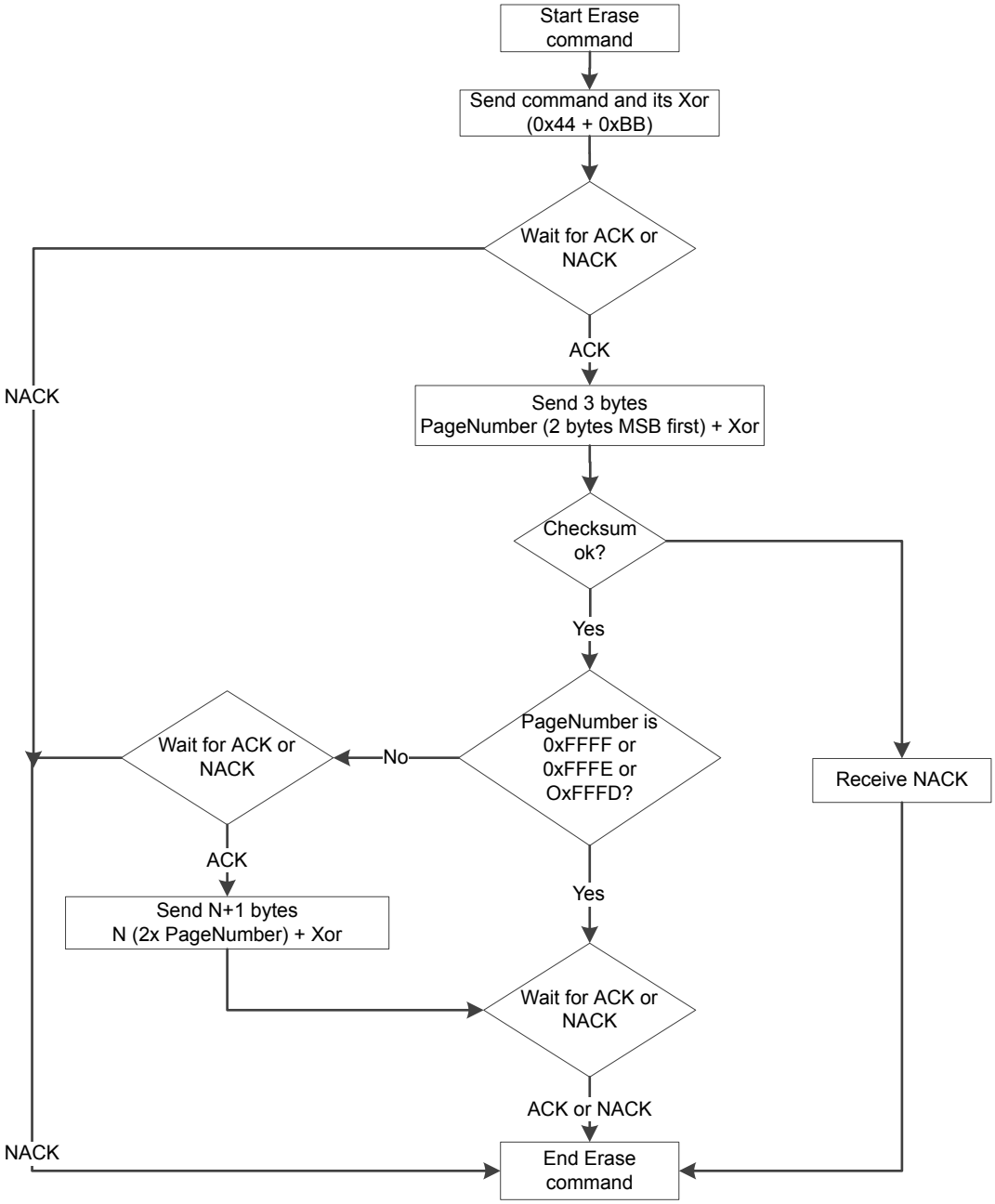
If the PageNumber represents Mass or Bank erase, and once pages erase is complete, it sends an ACK message. Otherwise, the bootloader starts the memory page(s) erase as defined by the host, and after pages erase it sends an ACK message.

- As every PageNumber is represented on 2 bytes (MSB first), the Bytes number is (BytesNumber = 2x PageNumber)
- When erasing different pages, the bootloader receives (BytesNumber + Xor byte)
  - PageNumber (each on 2 bytes)
  - Xor byte for all Pages

Note:

- *ACK message sent after the erase operation indicates that the required command is finished, but it does not guarantee that the erase operation has succeeded.*
- *No error is returned when performing erase operations on write protected sectors.*

Figure 15. **Erase memory command (host side)**



The host sends bytes to the STM32 as follows:

Mass/Bank Erase

1. Byte 1: 0x44
2. Byte 2: 0xBB
3. Wait for ACK or NACK
4. Bytes 3-4: Special erase (0xFFFx)
5. Bytes 5: Checksum of Bytes 3-4
6. Wait for ACK or NACK

Pages Erase

1. Byte 1: 0x44
2. Byte 2: 0xBB
3. Wait for ACK or NACK

4. Bytes 3-4: PageNumber (Represents the number of pages to erase)
   – Byte 3: MSB
   – Byte 4: LSB
5. Bytes 5: Checksum of Bytes 3-4
6. Wait for ACK or NACK
7. Bytes 6 to N – 1 (N = 2x PageNumber + 1)
   – 2x PageNumber; as every page number is represented on 2 bytes (MSB first)
   – “+1”; Checksum for all byte 6 to (N – 1)
8. Wait for ACK or NACK

Example of I3C frame:

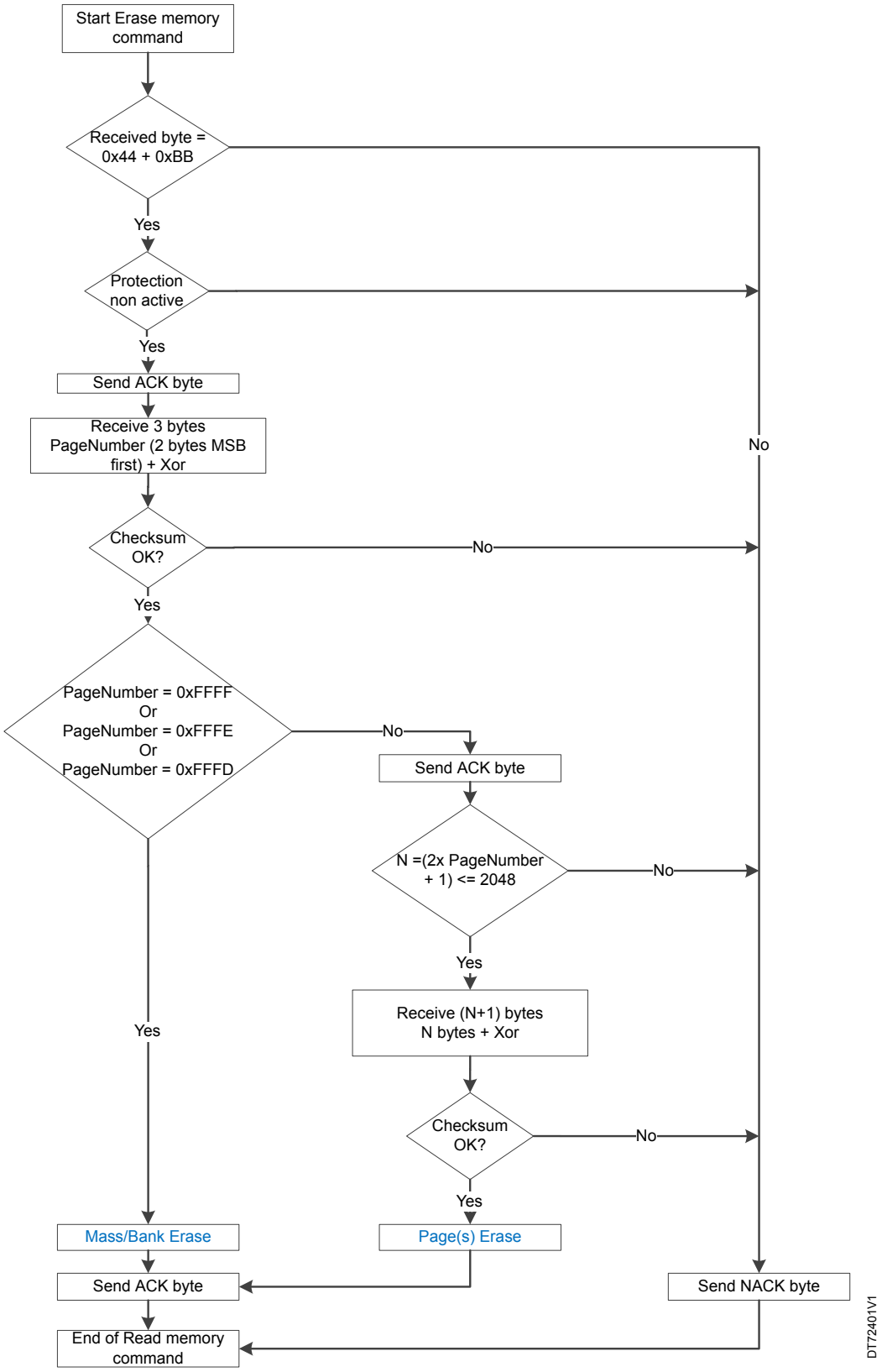
- erase page 3

```
0x44 0xBB Wait ACK 0x00 0x01 0xFE Wait ACK 0x00 0x03 0xFC Wait ACK
```

- erase page 1 and page 2:

```
0x44 0xBB Wait ACK 0x00 0x02 0xFD Wait ACK 0x00 0x01 0x00 0x02 0xFC Wait ACK
```

Figure 16. Erase memory command (device side)



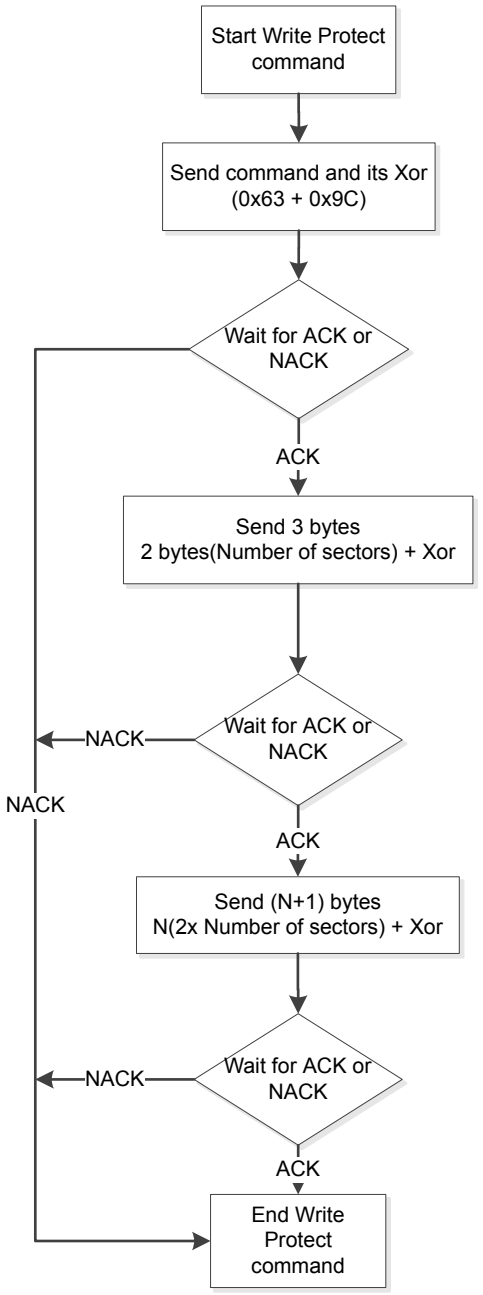
3.8 Write protect command

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 message to the host if protection 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.

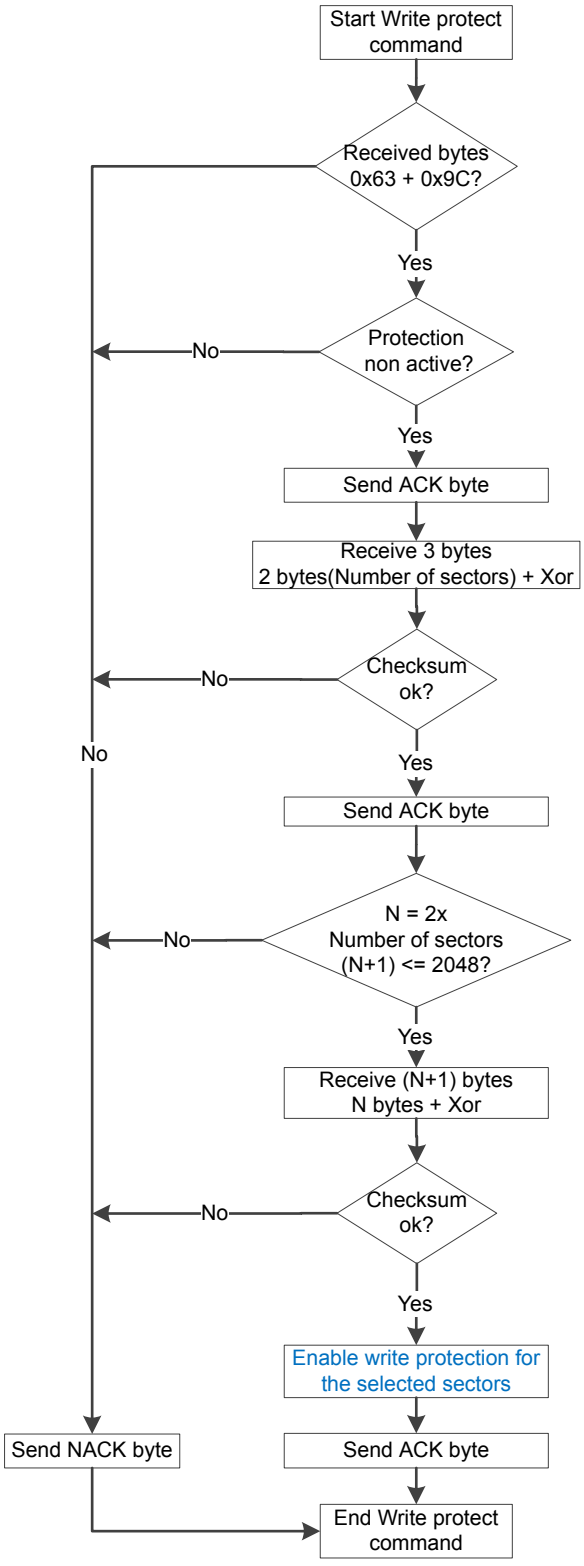
Note: The total number of sectors and the sector number to be protected are not checked, this means that no error is returned when a command is passed with a wrong number of sectors to be protected, or a wrong sector number.

Figure 17. Write protect memory command (host side)



Every sector number is sent on 2 bytes.

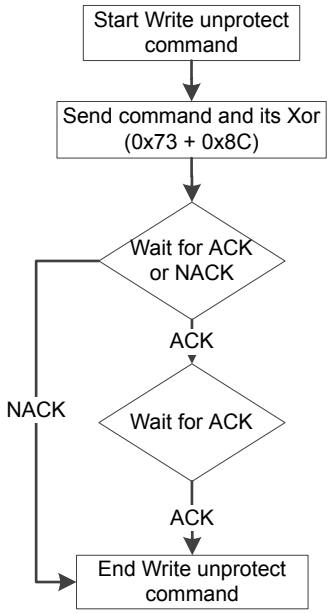
Figure 18. Write protect memory command (device side)



3.9 Write unprotect command

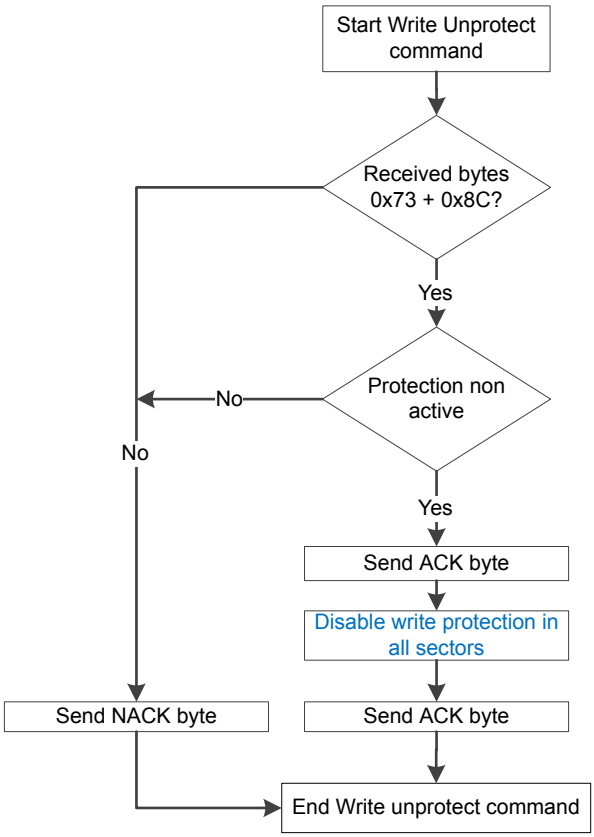
The Write unprotect command is used to disable the write protection of all the flash memory sectors. When the bootloader receives the Write unprotect command, it transmits the ACK message to the host if the protection is disabled, else it transmits NACK. After the transmission of the ACK message, the bootloader disables the write protection of all the flash memory sectors.

Figure 19. **Write unprotect memory command (host side)**



DT72427V1

Figure 20. **Write unprotect memory command (device side)**



DT72413V1

3.10 Read protect command

The Read protect command is used to enable the flash memory read protection. When the bootloader receives the Read protect command, it transmits the ACK message to the host if RDP is disabled else it transmits NACK. After the transmission of the ACK message, the bootloader enables the read protection for the flash memory.

Note

Note: RDP feature isn't present on all the STM32 devices. In this case, the Read protect command is omitted.

Figure 21. Read protect memory command (host side)

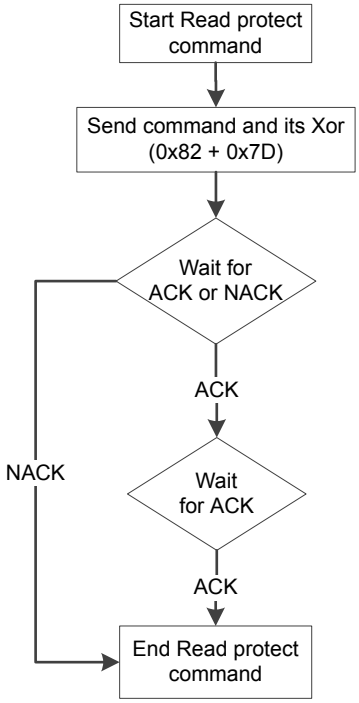
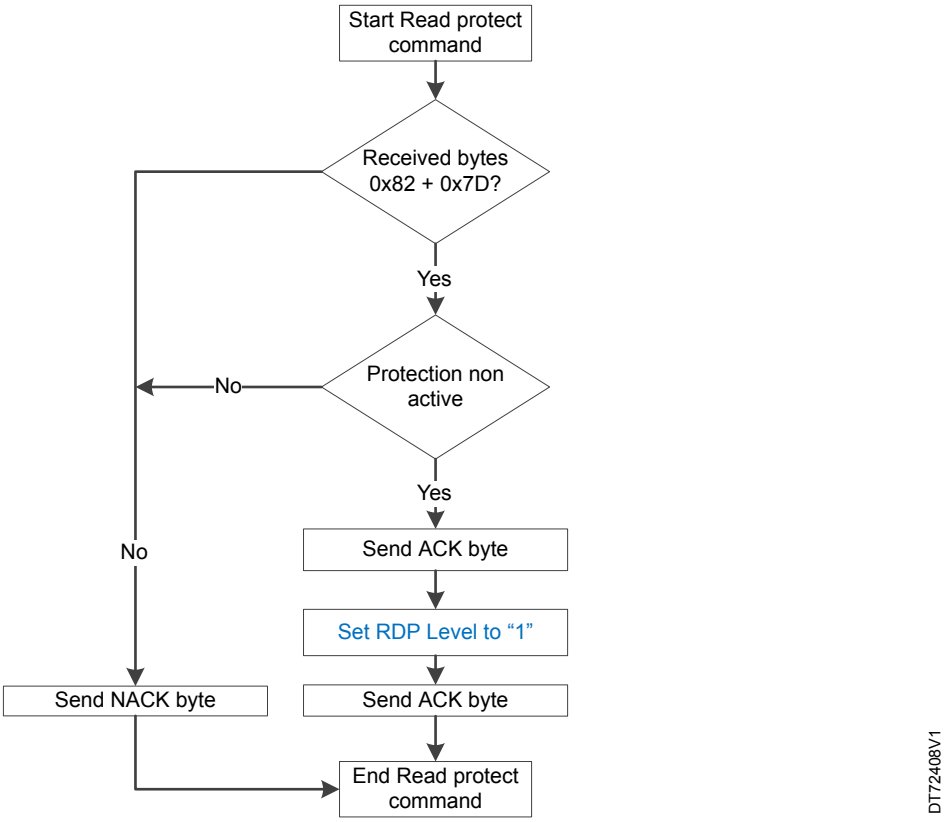


Figure 22. Read protect memory command (device side)



3.11 Read unprotect command

The Read unprotect command is used to disable the flash memory read protection. When the bootloader receives the Read unprotect command, it transmits the ACK message to the host. After the transmission of the ACK message, the bootloader deactivates the RDP.

Figure 23. Read unprotect memory command (host side)

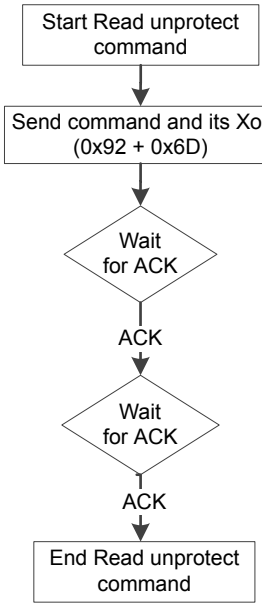
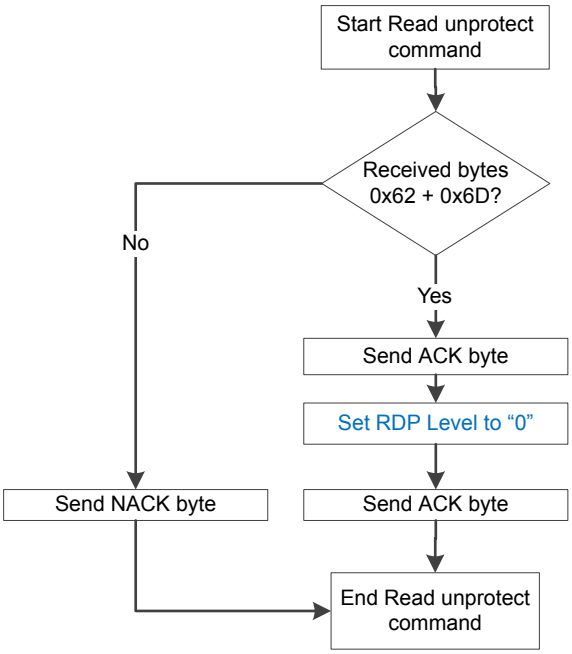


Figure 24. Read unprotect memory command (device side)



3.12 Special command

New bootloader commands are needed to support new STM32 features and to fulfill customer needs. To avoid specific commands for a single project, the Special command has been created to be as generic as possible.

Figure 25. **Special command (host side)**

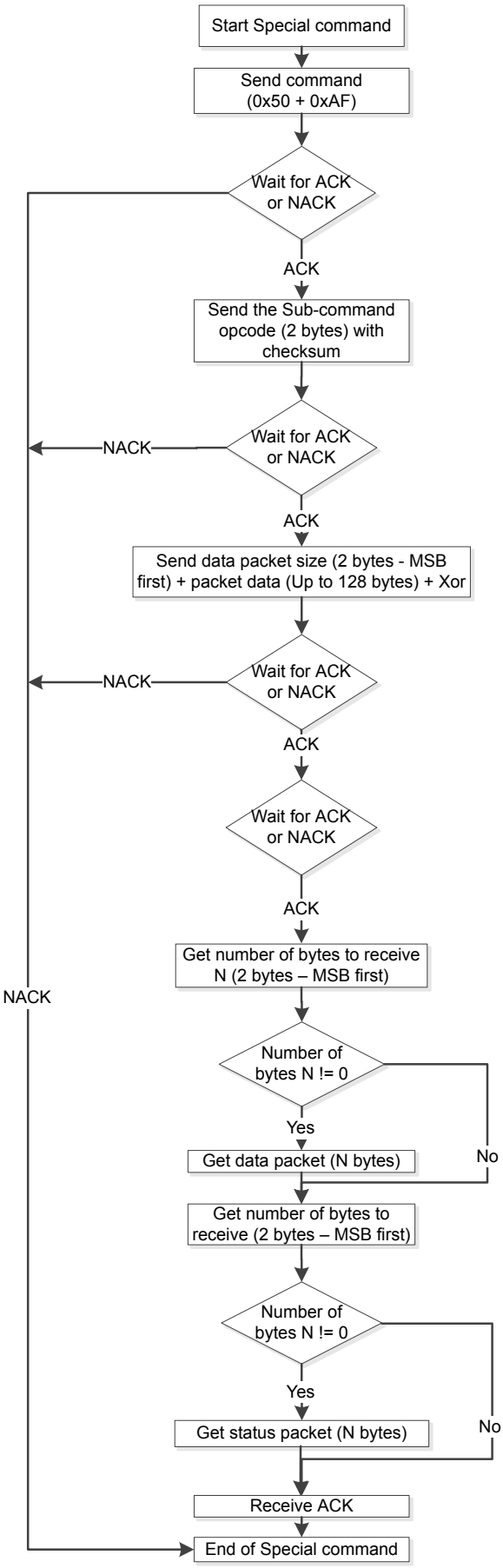
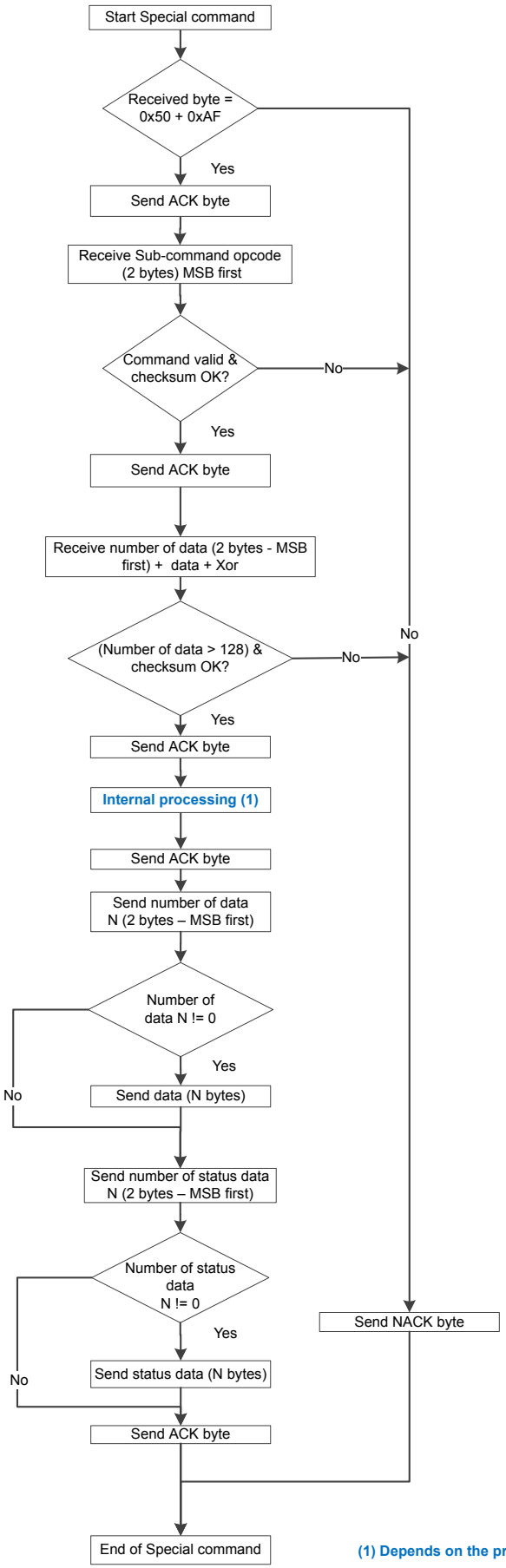


Figure 26. **Special command (device side)**



(1) Depends on the project requirements

When the bootloader receives the Special command, it transmits the ACK byte to the host. Once the ACK is transmitted, the bootloader waits for a sub-command opcode (two bytes, MSB first) and a checksum byte. If the sub-command is supported by the STM32 bootloader and its checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command.

To keep the command generic, the data packet received by the bootloader can have different sizes depending on the sub-command needs. Therefore, the packet is split in two parts:

- Size of the data (2 bytes, MSB first)
- N bytes of data
  - If N = 0, no data is transmitted
  - N must be lower than 128

If all conditions are satisfied (N ≤ 128 and checksum is correct), the bootloader transmits an ACK. Otherwise, it transmits a NACK byte and aborts the command. Once the sub-command is executed using the received data, the bootloader sends a response consisting of two consecutive packets:

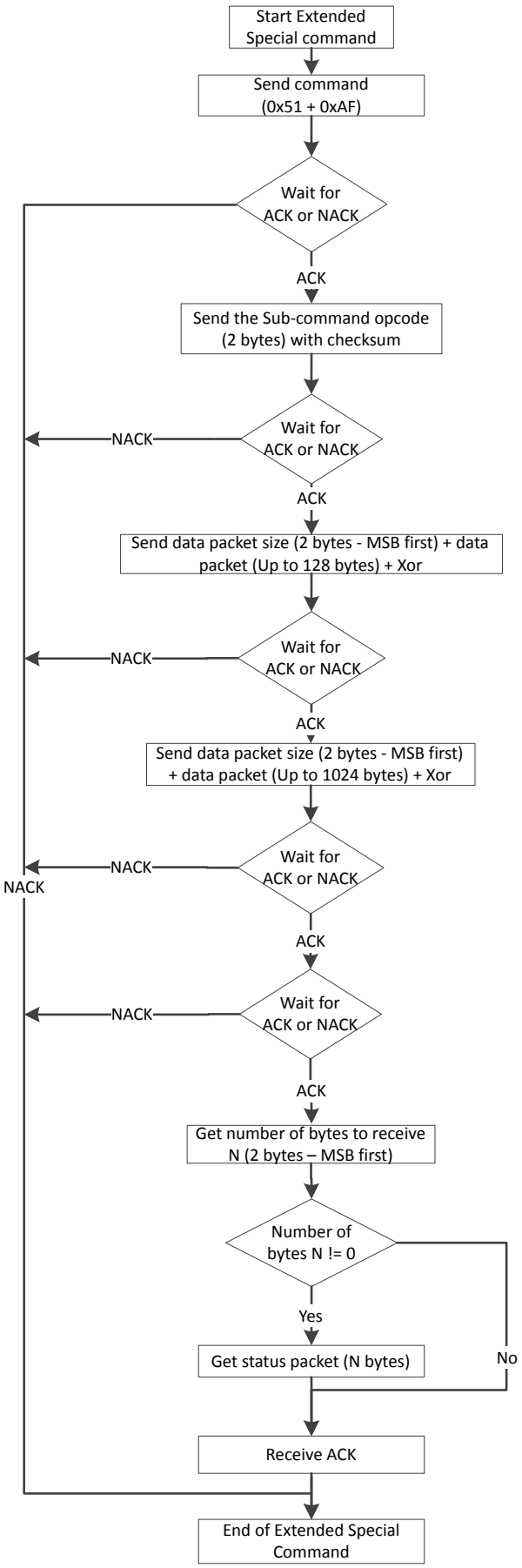
- Data packet
  - Size of the data (2 bytes, MSB first)
  - N bytes of data
  - If N = 0, no data is transmitted
- Status packet
  - Size of the status data (2 bytes, MSB first)
  - N bytes of data
  - If N = 0, no status data is transmitted

An ACK byte closes the Special command interaction between bootloader and the host.

3.13 Extended Special command

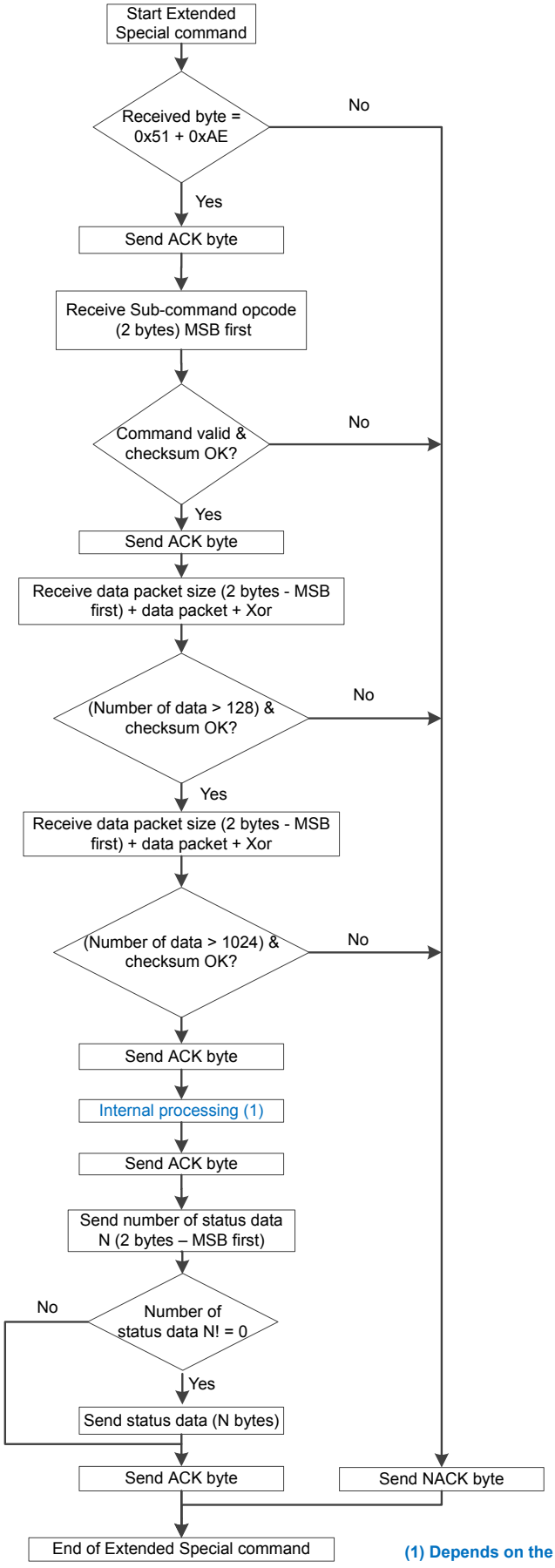
This command is slightly different from the Special command. It allows the user to send more data with the addition of a new buffer of 1024 bytes and, as a response, it only returns the status of the command.

Figure 27. **Extended Special command (host side)**



DT72416V1

Figure 28. **Extended Special command (device side)**



When the bootloader receives the Extended Special command, it transmits the ACK byte to the host. Once the ACK is transmitted, the bootloader waits for a subcommand opcode (two bytes, MSB first) and a checksum byte. If the sub-command is supported by the STM32 bootloader and its checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits a NACK byte and aborts the command.

The two packets then can be received depending on the subcommand needs:

- Packet1: Data1 packet, where number of bytes is limited to 128 bytes.
- Packet2: Data2 packet, where number of bytes is limited to 1024 bytes.

If all conditions are satisfied (Packet1: N ≤ 128 and checksum is correct, Packet2: N ≤ 1024 and checksum is correct), the bootloader transmits an ACK, otherwise it transmits a NACK byte and aborts the command.

Once the sub-command is executed using the received data, the bootloader sends a response consisting of one packet:

- Size of the data (2 bytes, MSB first)
- N bytes of data
  - If N = 0, no data are transmitted

An ACK byte closes the Extended Special command interaction between bootloader and the host.

Revision history

Table 3. **Document revision history**

Date	Version	Changes
22-Feb-2023	1	Initial release.
26-Feb-2024	2	Updated Product series in Section Introduction to add STM32H7.
06-Feb-2025	3	Updated Product series in Section Introduction to add STM32U3.

Contents

List of figures

List of tables