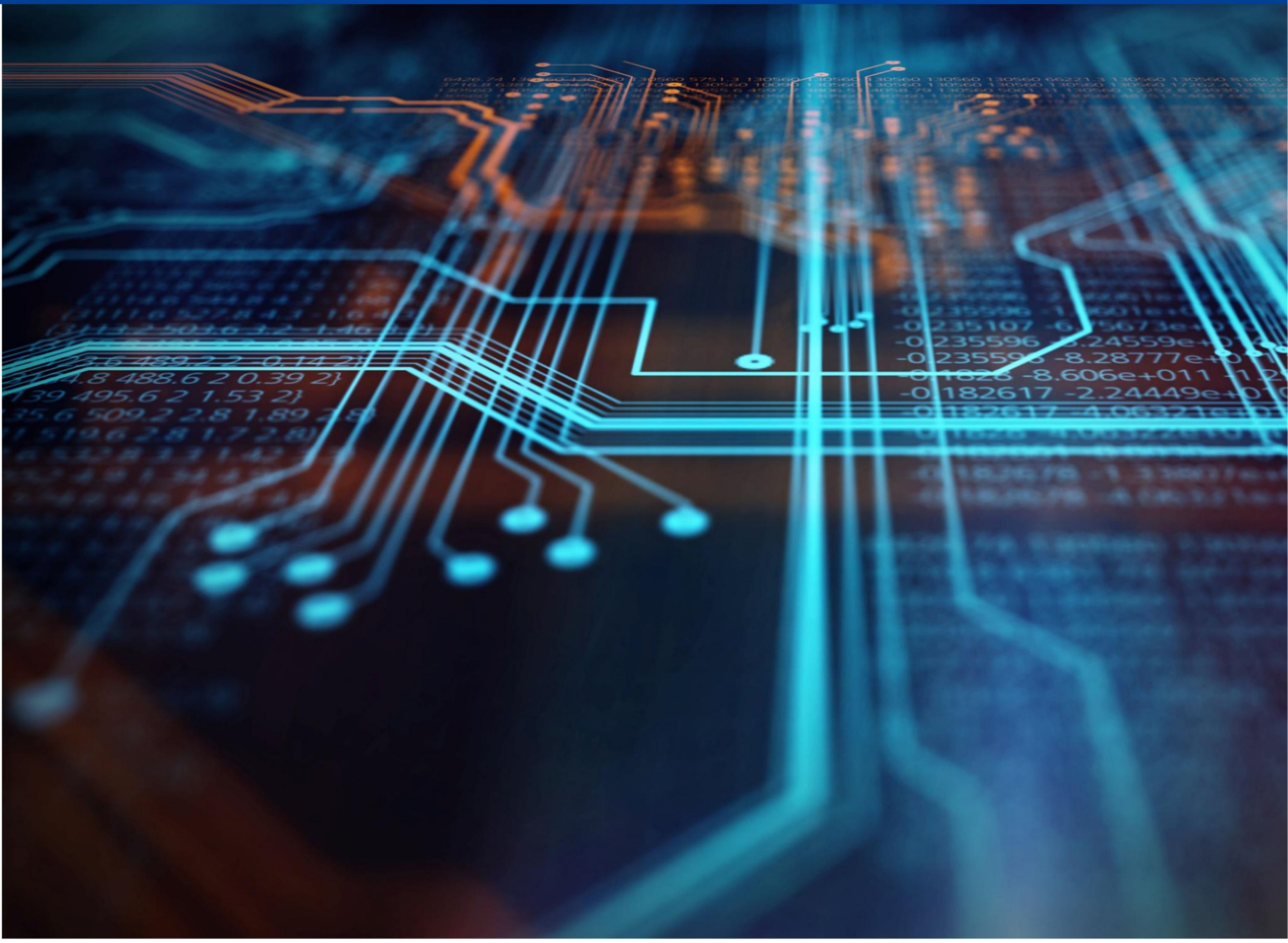


TPS32 Bootloader User Guide

Version: A2

Chapter 1 Introduction

1.1 Introduction

The bootloader is stored in the TPS325Mx device's internal system memory and is programmed by 3PEAK during the production process. Its main task is to download the application program to the internal main flash memory using available serial peripherals, such as USART and IIC.

Communication protocols are defined for each serial interface to ensure compatibility with command sets and sequences. This document applies to the products listed in Table 1-1 and describes the supported peripherals and hardware requirements when using the TPS32 device's bootloader.

Table 1-1 Applicable Devices

Product Series		Part Number
TPS325Mx	TPS325M01	TPS325M0177Q-QP7T
		TPS325M0166Q-QP7T
		TPS325M0156Q-QP7T
		TPS325M0177Q-QP5T
		TPS325M0166Q-QP5T
		TPS325M0155Q-QP5T
		TPS325M0177Q-QP6T
		TPS325M0166Q-QP6T
		TPS325M0155Q-QP6T
	TPS325M51	TPS325M5177Q-QP7T
		TPS325M5165Q-QP7T
		TPS325M5166Q-QP7T
		TPS325M5167Q-QP7T
		TPS325M5155Q-QP7T
		TPS325M5156Q-QP7T
		TPS325M5177Q-QP5T
		TPS325M5166Q-QP5T
		TPS325M5155Q-QP5T
		TPS325M5177Q-QP6T
		TPS325M5166Q-QP6T
		TPS325M5177Q-FSDR
		TPS325M5177I-FSDR
		TPS325M5166I-FSDR
		TPS325M5156I-FSDR
	TPS325M0A	TPS325M0A57Q-QP5T
	TPS325M5A	TPS325M5A57Q-QP5T

1.2 Related Documents

For device listed in Table 1-1, refer to the following documents and tools from www.3peak.com:

- TPS325M Datasheet
- TPS325M Technical Reference Manual
- TPS32 Programmer User Guide
- TPS32 Programmer Tool

1.3 Documentation Conventions

Convention	Usage
Bold	Displays commands, menu paths, and icon names in procedures. For example: Click the **File** icon and then click **Open**.
File > **New**	Represents menu path. For instance: **File** > **New** > New Project

`Courier New`	Displays file locations, user entered text, and source code.
	For example:
	`<your_sdk_path>/example/tpsensor/`
	`tpsensor_exp/source`

Chapter 2 Description

2.1 Boot Mode

Flash memory, system memory, and SRAM are the three common types of memory that can be used for storing programs. TPS325M devices support booting from these three types of memory.

The boot mode is controlled by following user option byte bits in the FMC_OPTR register.

Table 2-1 Boot Mode

BOOT_ LOCK	nSWBOOT0 FMC_OPTR[26]	BOOT0 Pin	nBOOT0 FMC_OPTR[27]	nBOOT1 FMC_OPTR[23]	DBANK FMC_OPTR[22]	SWAPBANK FMC_OPTR[20]	Boot Memory Space Alias
1	X	X	X	X	X	X	Main Flash memory Bank0 is selected
0	1	0	X	X	0	X	Main Flash memory Bank0 is selected
					1	0	Main Flash memory Bank0 is selected
					1	1	Main Flash memory Bank1 is selected
0	0	X	1	X	0	X	Main Flash memory Bank0 is selected
					1	0	Main Flash memory Bank0 is selected
					1	1	Main Flash memory Bank1 is selected
0	1	1	X	0	X	X	Embedded SRAM1 is selected
0	0	X	0	0	X	X	Embedded SRAM1 is selected
0	1	1	X	1	X	X	System memory is selected
0	0	X	0	1	X	X	System memory is selected

Boot Process

The boot process is shown in Figure 2-1.

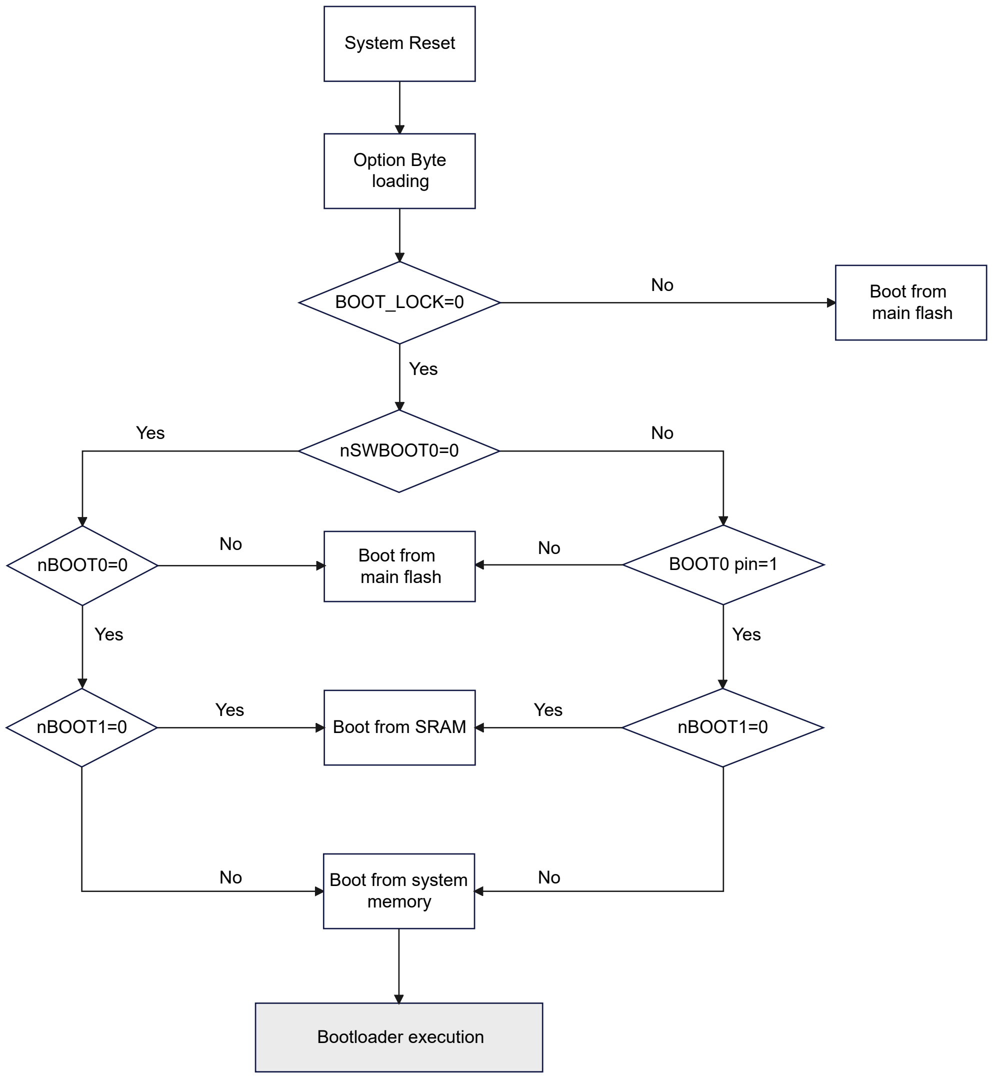


Figure 2-1 Boot Process

If users need the device to execute the bootloader function, they must configure the appropriate control bits in option byte. Setting these control bits will force the device to boot from system memory.

2.2 Serial Peripherals

The bootloader for a given TPS32 device may be compatible with various embedded serial peripherals to download code into internal Flash memory. Bootloader identifiers (IDs) are used to identify the supported serial peripherals.

The bootloader ID for the TPS32 device is encoded in one byte format, 0xAB, where:

- A indicates serial peripherals used

  X=1: One USART and one IIC
- B indicates the software version

Moreover, the communication protocols for the various peripherals vary and are indicated by the bootloader version code.

Product Series	Support Serial Peripherals	Bootloader ID		Bootloader Version
		ID	Memory	

TPS325Mx	USART0
	IIC1
		0x11	0xFFE0000	USART(V1.1)
				IIC(V1.1)

Users may use the `GET ID` and `GET VERSION` commands to retrieve the related information.

2.3 Memory Management

TPS325M series devices have several internal memory types:

- Main Flash Memory

  This memory is designated for storing the user's application program.

- System Memory

  The bootloader is programmed into system memory by 3PEAK during manufacturing and is not user-accessible. It offers a straightforward method for users to upload their applications.

- SRAM

  Used to store data while the program is running. When power off, the data in SRAM will be lost.

- Other Memories
  - OTP
  - NVR
  - Option bytes

  Both OTP and NVR can store specific user data, while option bytes allow users to configure the device before running the application.

Currently, the bootloader only supports modification of the main Flash and option bytes areas, as shown in Table 2-2.

Table 2-2 Supported Memory Operations

Memory Area	Write Command	Read Command	Erase Command	Go Command
Main Flash	√	√	√	√
System memory	×	×	×	×
SRAM	×	×	×	×
Option bytes	√	√	×	×
NVR	×	×	×	×
OTP	×	×	×	×

(1) √ - Supported

(2) × - Not supported

2.4 Programming Constraints

Bootloader write commands require data boundary alignment:

- Single bank mode: Addresses must be aligned to 16 bytes.
- Dual bank mode: Addresses must be aligned to 8 bytes.

Chapter 3 Bootloader

3.1 Configuration

When the device boots from system memory, the bootloader will execute and occupy the following hardware resources.

Bootloader	Peripherals	State	Comment
Common to all bootloaders	RCC	IHS	System clock is 8 MHz.
	SYSCFG	Enabled	Enable the clock so that the SYSCFG registers can be accessed.
	System memory	-	32 Kbytes starting from address 0xFFE0000 contain the bootloader firmware.
	FMC	-	All Flash operations through Flash Memory Controller (FMC).
	Main Flash	-	Flash memory for users to store their application program.
	SRAM	-	336 Kbytes starting from address 0x20000000 are used by the bootloader.
	Systick	-	Used to handle timeout in communication
USART bootloader	USART0	Enabled	Once initialized, the USART0 configuration is 8 bits, even parity and 1 Stop bit.
	USART0_RX pin	Input	PA2 pin: USART0 in reception mode. Used in input pull-up mode.
	USART0_TX pin	Output	PA1 pin: USART0 in transmission mode. Used in alternate push-pull, pull-up mode.
IIC bootloader	IIC1	Enabled	The IIC1 configuration is IIC1 speed: up to 200 kHz, 7-bit address, slave mode, analog filter ON. Slave 7-bit address: 0b0000100x (where x = 0 for write and x = 1 for read)
	IIC1_SCL pin	Input/ Output	PC9 pin: clock line is used in open-drain no pull mode.
	IIC1_SDA pin	Input/ Output	PC8 pin: data line is used in open-drain no pull mode.

3.2 Flow Chart

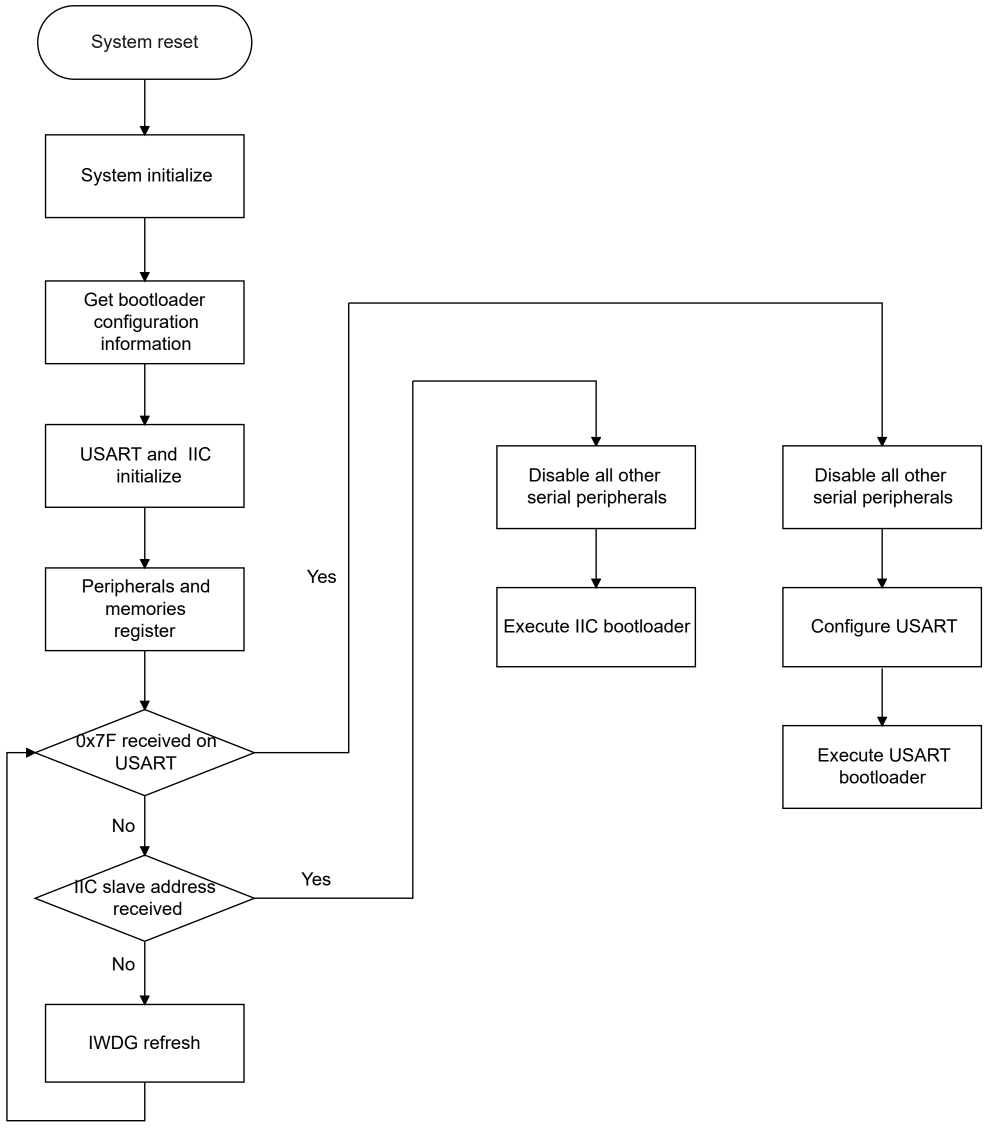

System reset
System initialize
Get bootloader configuration information
USART and IIC initialize
Peripherals and memories register
0x7F received on USART
Yes
No
IIC slave address received
Yes
No
IWDG refresh
Disable all other serial peripherals
Execute IIC bootloader
Disable all other serial peripherals
Configure USART
Execute USART bootloader

Figure 3-1 Bootloader Flow Chart

3.3 Command Set

Type	Command [1]	Code	Description
Get Infomation	Get [2]	0x11	Get the version and the allowed commands supported by the current version of the bootloader.
	Get Version & Read Protection Status [2]	0x12	Get the bootloader version.
	Get ID [2]	0x13	Get the bootloader ID.

Type	Command [(1)]	Code	Description
Memory Operation	Read [(3)]	0x31	Read up to 256 bytes of memory starting from an address specified by the application.
	Go [(3)]	0x32	Jump to the user application code located in the internal Flash memory or in the SRAM.
	Write [(3)]	0x33	Write up to 256 bytes to the RAM or Flash memory starting from an address specified by the application.
	Erase [(3)]	0x35	Erases from one to all the Flash memory pages using a two-byte addressing mode.

(1) If a denied command is received or an error occurs during the command execution, the bootloader sends NACK byte and goes back to the command checking.

(2) Read protection. When the Read protection (RDP) 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.

(3) Refer to TPS32 product datasheets and technical reference manual to know the valid memory areas for these commands.

3.4 Communication Security

All communication from the programming tool (PC) to the device undergoes verification:

1. The received data block bytes are XORed. The computed XOR value of all previous bytes is added to the end of each communication (checksum byte).
2. For each command, the host sends a byte and its complement (XOR = 0x00).
3. UART: Parity check is active (even parity).
4. Each data packet is either accepted (ACK response, 0xA3) or discarded (NACK response, 0x1A).

3.5 USART Bootloader

3.5.1 USART Bootloader Process

When the system memory boot mode is selected and a system reset occurs, the bootloader program starts to scan the USART interface for auto-baud rate detection byte(0x7F). Once the 0x7F data frame is captured, the baud rate is updated at the end of the 6th bit. Then bootloader program sends the ACK byte (0xA3) to the host and is ready to get command.

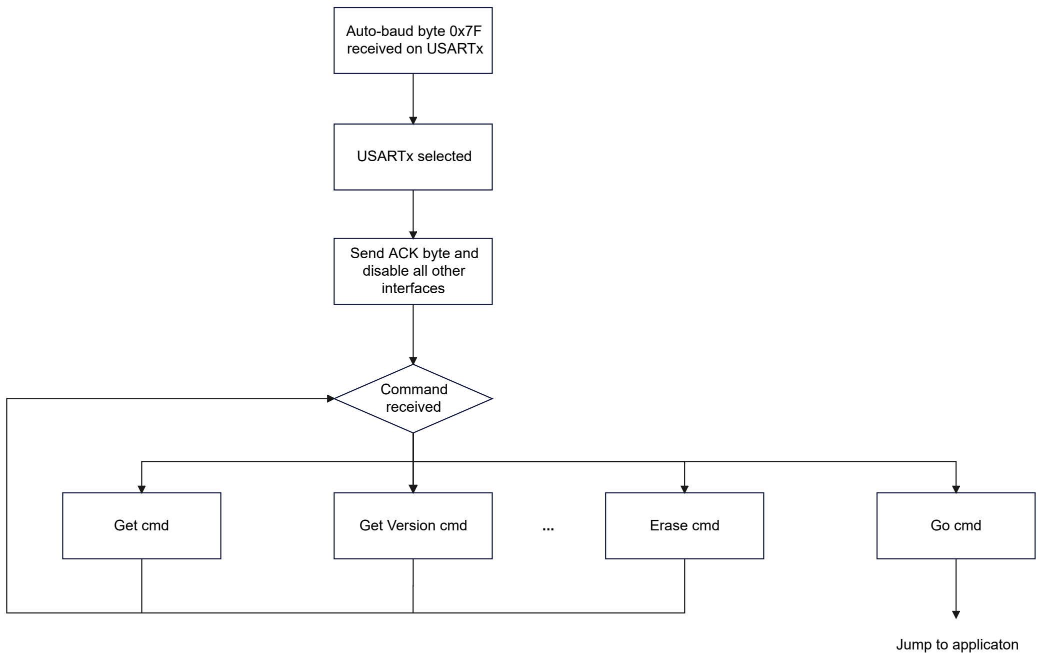


Figure 3-2 USART Bootloader Flow Chart

3.5.2 USART Baud Rate

Calculating the USARTx baud rate from the first received byte enables boot loader operation across a wide range of baud rates. However, upper and lower limits must be enforced to ensure reliable data transmission.

To properly transfer data from host to microcontroller, the maximum deviation between USARTx's initial internal baud rate and the host's actual baud rate must be less than 2%. The deviation between the host and microcontroller baud rates (E, expressed in percentage) is calculated as:

$$E = \frac{|\text{baud rate} - \text{host baud rate}|}{\text{baud rate}} \times 100\% \qquad (1)$$

- Minimum baud rate: 1200 bit/s
- Maximum baud rate: 115200 bit/s

3.5.3 Command Implementation

3.5.3.1 GET Command

Upon receiving the GET command, the bootloader transmits the supported command codes, command count, and bootloader version to the host. See the following figures for the exchange flow chart:

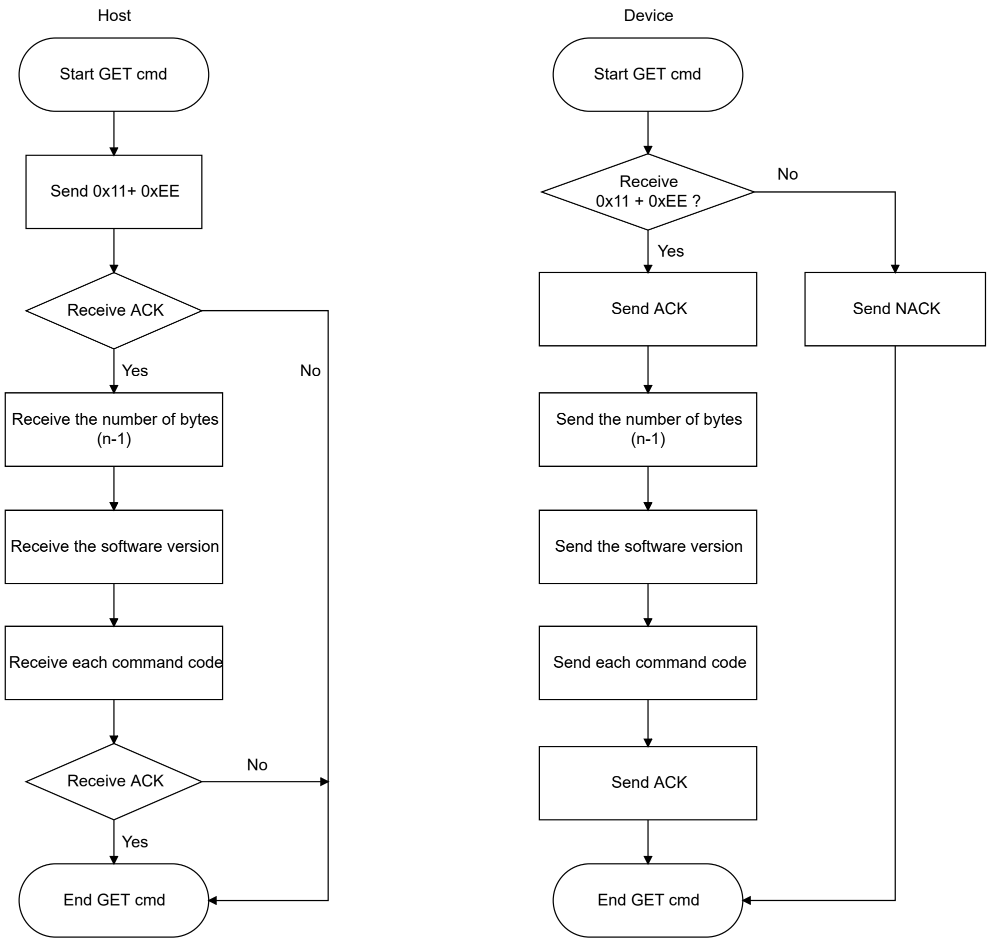
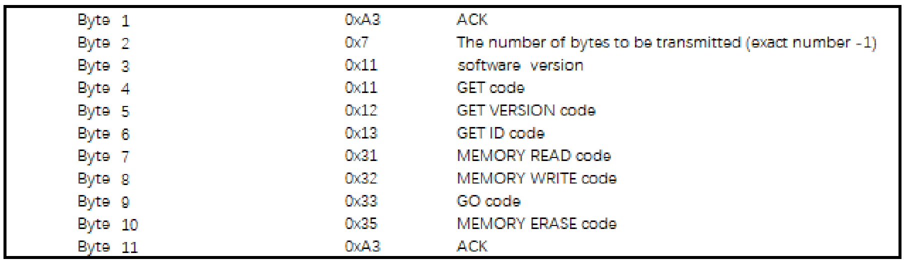


Figure 3-3 USART Bootloader GET Flow Chart

NOTE: All the number of bytes exchanged during communication is the actual byte number minus one.

The device sends the bytes as below:



Byte 1	0xA3	ACK
Byte 2	0x7	The number of bytes to be transmitted (exact number -1)
Byte 3	0x11	software version
Byte 4	0x11	GET code
Byte 5	0x12	GET VERSION code
Byte 6	0x13	GET ID code
Byte 7	0x31	MEMORY READ code
Byte 8	0x32	MEMORY WRITE code
Byte 9	0x33	GO code
Byte 10	0x35	MEMORY ERASE code
Byte 11	0xA3	ACK

3.5.3.2 GET VERSION Command

The Get VERSION command is used to obtain the bootloader version. Once the command is received, the bootloader sends the version to the host.

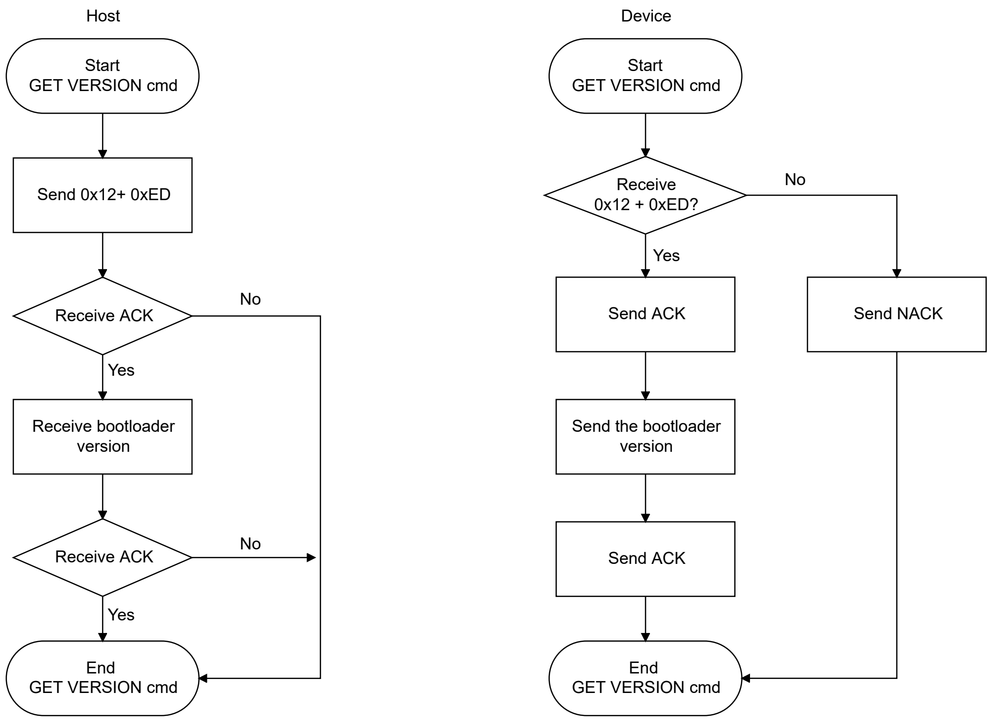


Figure 3-4 USART Bootloader GET VERSION Flow Chart

The device sends the bytes as below:

Byte 1	0xA3	ACK
Byte 2	0x11	The bootloader version
Byte 3	0xA3	ACK

3.5.3.3 GET ID Command

The Get ID command is used to get the chip device ID that has information about serial interfaces supported and communication protocol version.

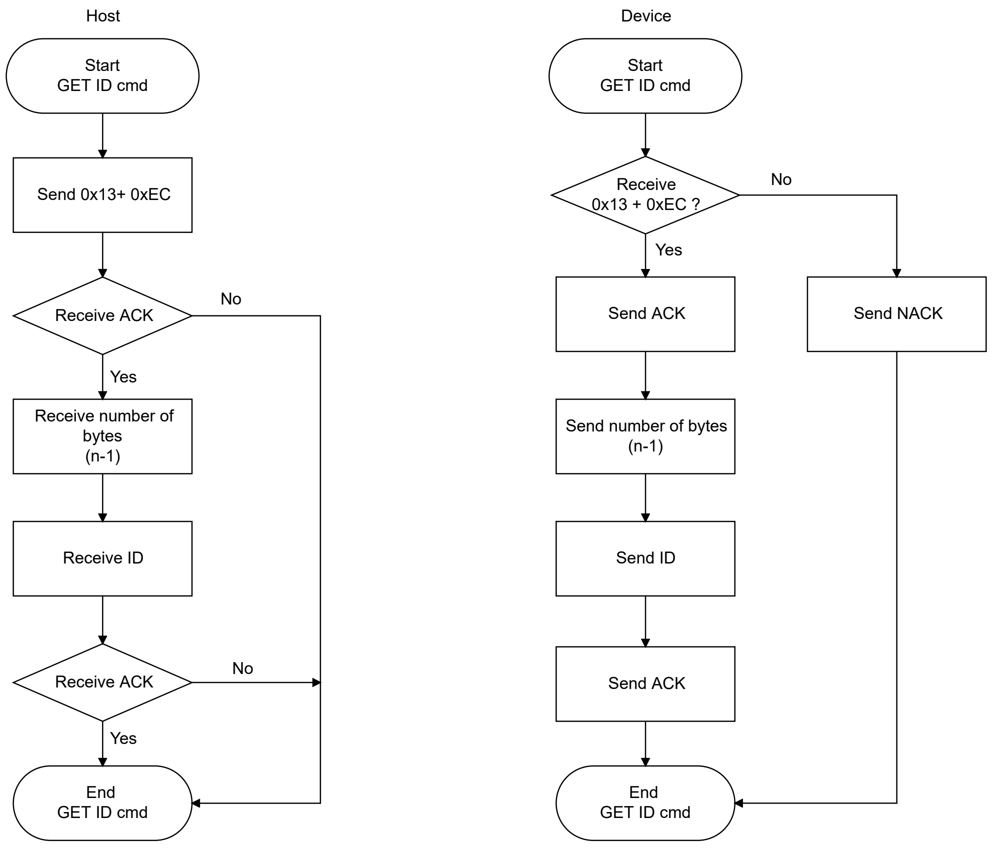


Figure 3-5 USART Bootloader GET ID Flow Chart

The device sends the bytes as below:

Byte1	0xA3	ACK
Byte2	0x3	The number of bytes
Byte3	0x23	ID MSB
Byte4	0x0	
Byte5	0x0	
Byte6	0x1	ID LSB
Byte7	0xA3	ACK

3.5.3.4 READ Command

The READ command is used to read data from memory supported by bootloader, including main Flash memory and option byte area. The maximum size of the data to be read is 256 bytes.

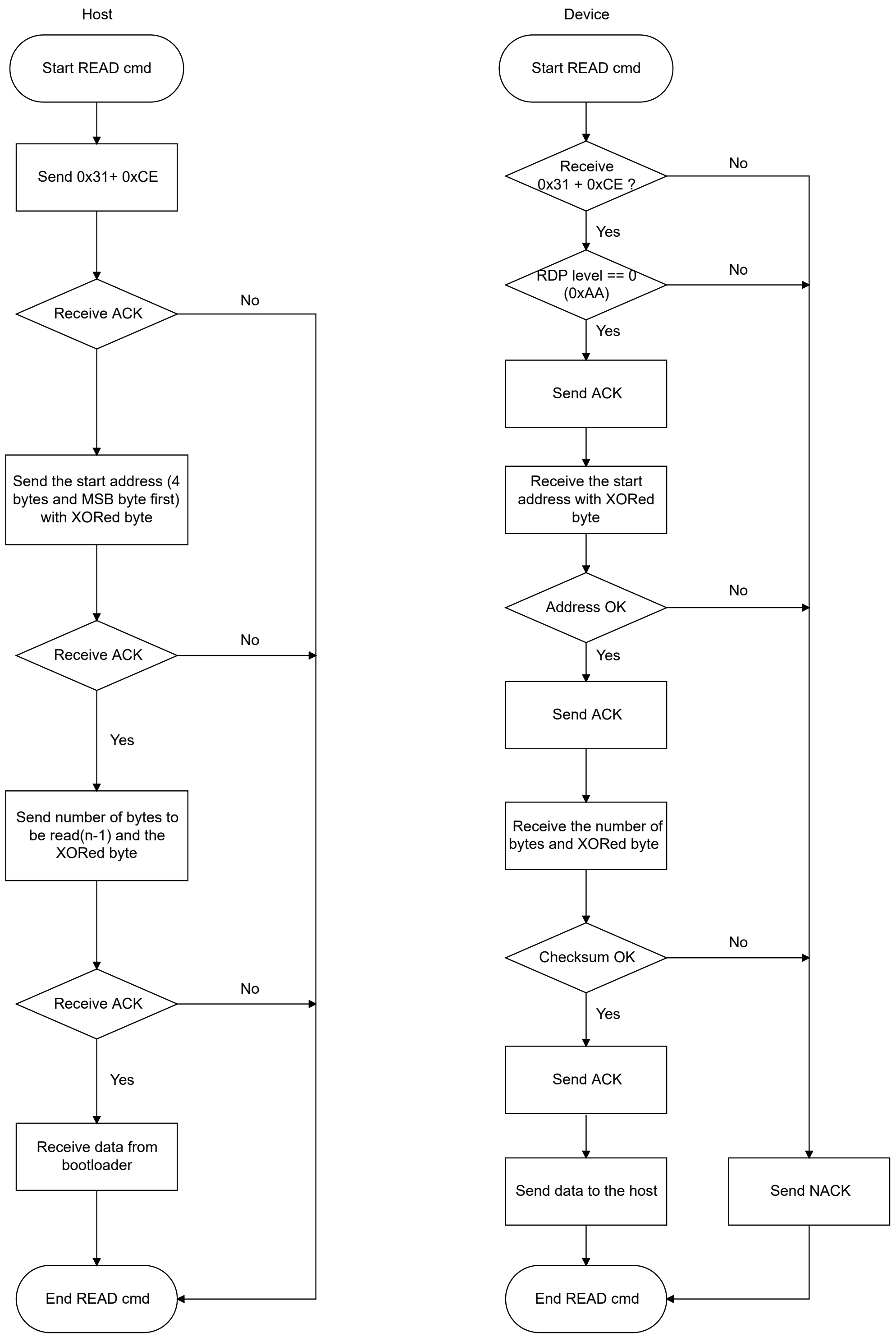
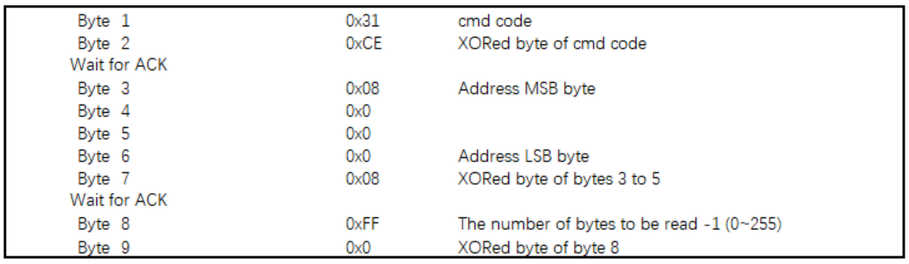


Figure 3-6 USART Bootloader READ Flow Chart

The host sends the bytes as an example:



Byte 1	0x31	cmd code
Byte 2	0xCE	XORed byte of cmd code
Wait for ACK		
Byte 3	0x08	Address MSB byte
Byte 4	0x0	
Byte 5	0x0	
Byte 6	0x0	Address LSB byte
Byte 7	0x08	XORed byte of bytes 3 to 5
Wait for ACK		
Byte 8	0xFF	The number of bytes to be read -1 (0~255)
Byte 9	0x0	XORed byte of byte 8

3.5.3.5 GO Command

The GO command is used to jump to application code, which is programmed in the main Flash through the bootloader.

When device gets the GO command and a valid address to jump, the bootloader will:

1. Deinitialize all the peripherals used in the bootloader code.
2. Initialize the main stack pointer of the application code.
3. Jump to the application entrance address.

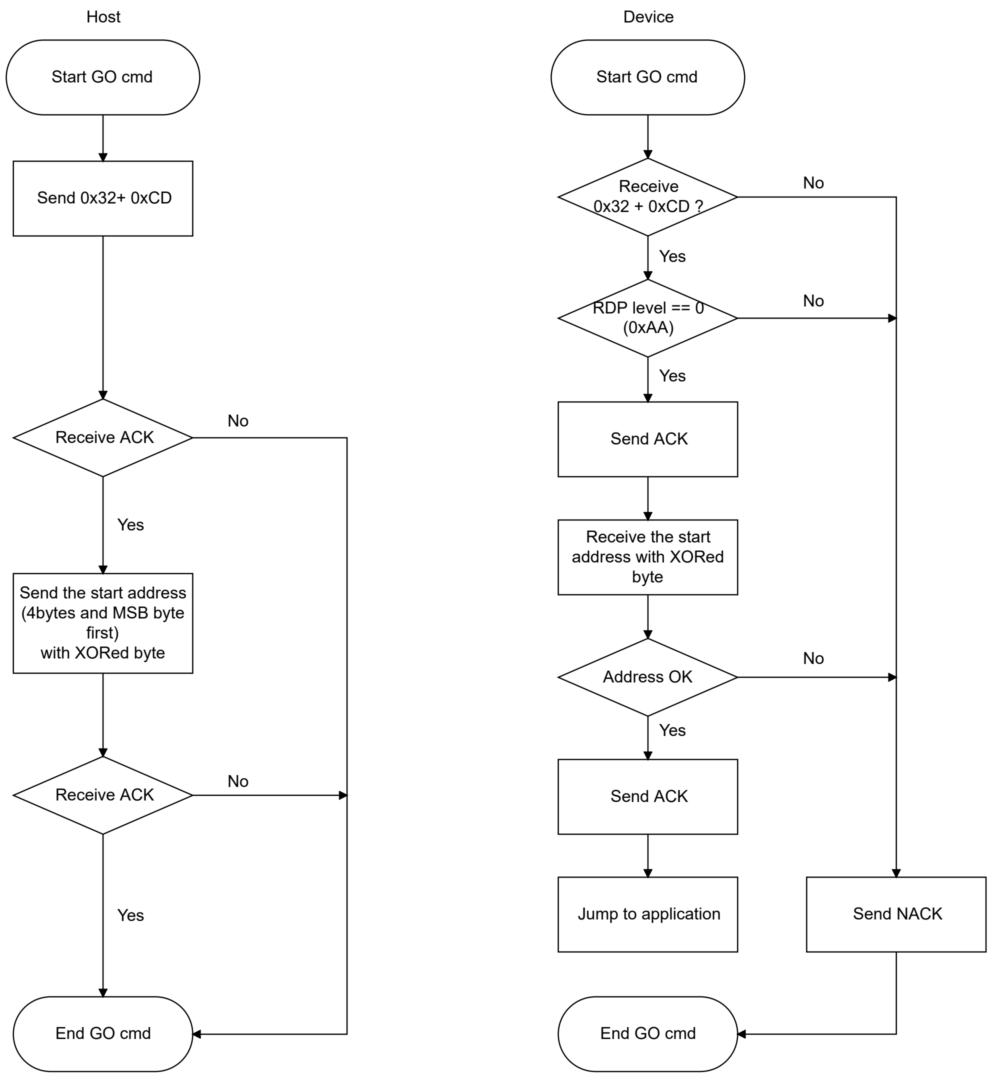


Figure 3-7 USART Bootloader GO Flow Chart

The host sends the bytes as an example:

Byte 1	0x32	cmd code
Byte 2	0xCD	XORed byte of cmd code
Wait for ACK		
Byte 3	0x08	Address MSB byte
Byte 4	0x0	
Byte 5	0x0	
Byte 6	0x0	Address LSB byte
Byte 7	0x08	XORed byte of bytes 3 to 6
Wait for ACK		

3.5.3.6 WRITE command

The WRITE command is used to write data to memory supported by the bootloader, including the main Flash memory and option byte area. The maximum size of the data to be written is 256 bytes.

NOTE:

1. The main Flash and OTP area must be erased first before the write operation being performed.
2. For OTP, the address to be written must be 8 bytes aligned.
3. For main Flash, the address must be:
   - 16 bytes aligned in single bank mode.
   - 8 bytes aligned in dual bank mode.

Host sends the bytes like below to write 0x11111111 and 0x22222222 to address 0x8000000.

Byte 1	0x33	cmd code
Byte 2	0xCC	XORed byte of cmd code
Wait for ACK		
Byte 3	0x08	Address MSB byte
Byte 4	0x0	
Byte 5	0x0	
Byte 6	0x0	Address LSB byte
Byte 7	0x08	XORed byte of bytes 3 to 5
Wait for ACK		
Byte 8	0x7	N: the number of bytes to be write -1 (0~255)
Byte 9	0x11	The first byte of data
Byte 10	0x11	
Byte 11	0x11	
Byte 12	0x11	
Byte 13	0x22	
Byte 14	0x22	
Byte 15	0x22	
Byte 16	0x22	The last byte of data
Byte 17	0x7	XORed checksum
Wait for ACK		

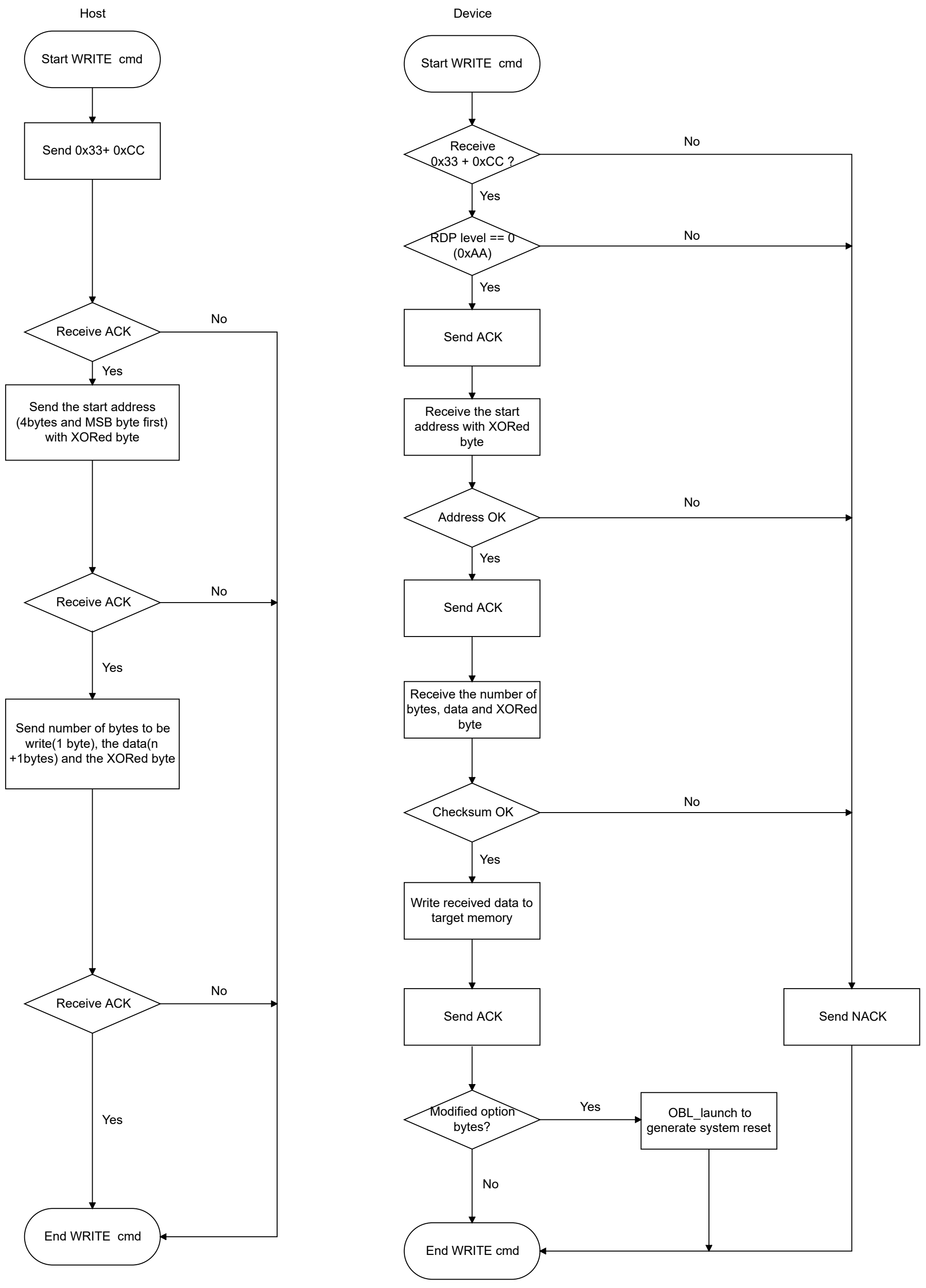

Host
Device
Start WRITE cmd
Send 0x33+ 0xCC
Receive ACK
No
Yes
Send the start address (4bytes and MSB byte first) with XORed byte
Send number of bytes to be write(1 byte), the data(n +1bytes) and the XORed byte
End WRITE cmd
Receive 0x33 + 0xCC ?
RDP level == 0 (0xAA)
Send ACK
Receive the start address with XORed byte
Address OK
Receive the number of bytes, data and XORed byte
Checksum OK
Write received data to target memory
Send NACK
Modified option bytes?
OBL_launch to generate system reset

Figure 3-8 USART Bootloader WRITE Flow Chart

3.5.3.7 ERASE command

The ERASE command allows the host to erase the main Flash memory of the device, and it supports mass erase, bank erase, and sector erase.

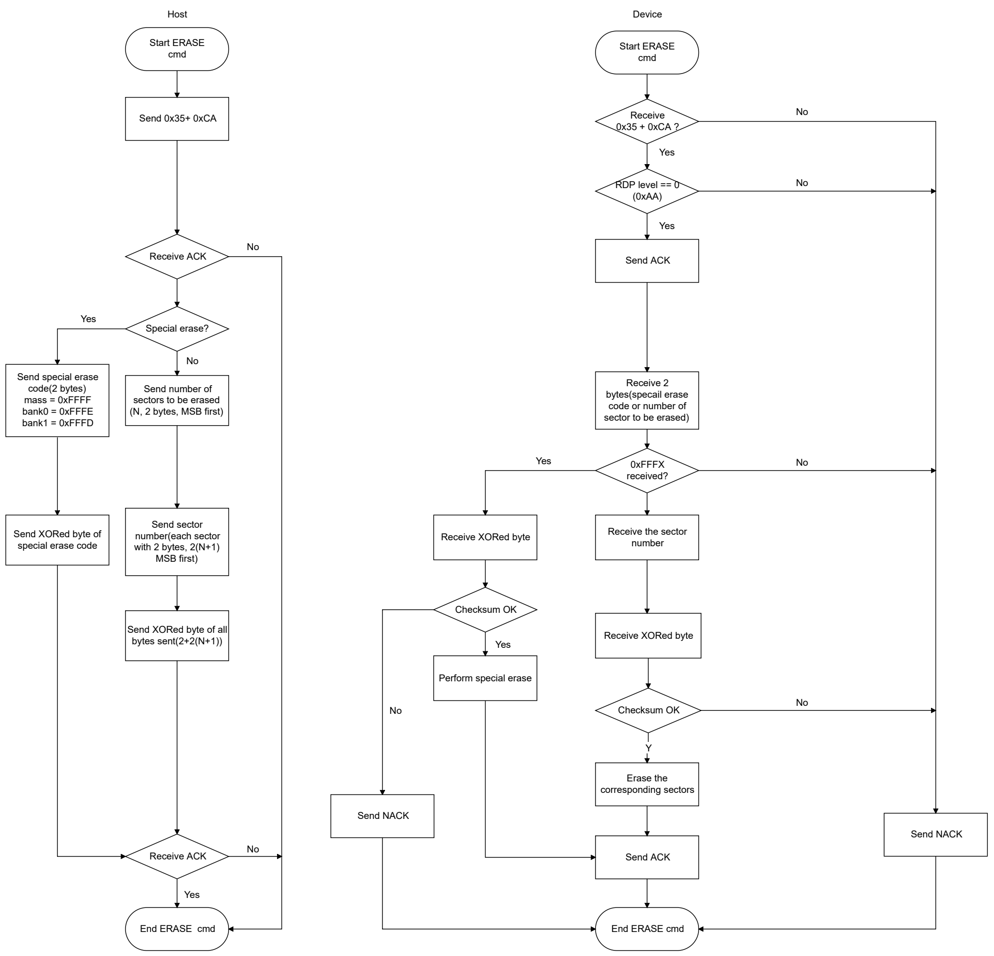

Host
Device
Start ERASE cmd
Send 0x35+ 0xCA
Receive ACK
No
Special erase?
Yes
No
Send special erase code(2 bytes) mass = 0xFFFF bank0 = 0xFFFE bank1 = 0xFFFD
Send number of sectors to be erased (N, 2 bytes, MSB first)
Send XORed byte of special erase code
Send sector number(each sector with 2 bytes, 2(N+1) MSB first)
Send XORed byte of all bytes sent(2+2(N+1))
Receive ACK
No
Yes
End ERASE cmd
Start ERASE cmd
Receive 0x35 + 0xCA ?
No
Yes
RDP level == 0 (0xAA)
No
Yes
Send ACK
Receive 2 bytes(specail erase code or number of sector to be erased)
Yes
0xFFFX received?
No
Receive XORed byte
Receive the sector number
Checksum OK
Yes
Perform special erase
Receive XORed byte
No
Checksum OK
No
Y
Erase the corresponding sectors
Send NACK
Send NACK
Send ACK
End ERASE cmd

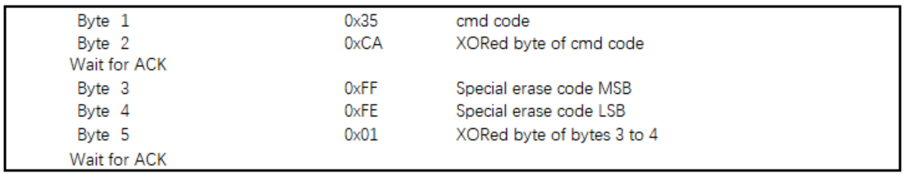
Figure 3-9 USART Bootloader ERASE Command Flow Chart

From perspective of the device, when receiving the ERASE command and sending ACK to the host, it starts to wait for 2 bytes (N):

- Either special erase code in special erase mode,
  - N = 0xFFFF: mass erase
  - N = 0xFFFE: bank0 erase
  - N = 0xFFFD: bank1 erase
- Or the number of sectors to be erased in sector erase mode.
  - 0 ≤ N < the maximum sector number of the device

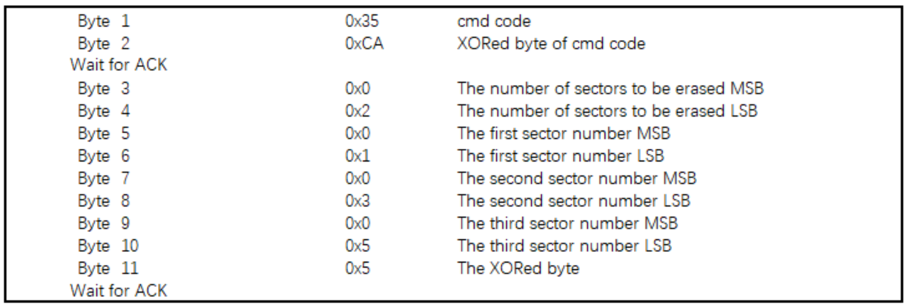
NOTE: The exact number of sectors to be erased is (N+1), and the bootloader receives 2× (N+1) bytes representing corresponding sectors.

The host sends the bytes as below to perform bank 0 erase:



Byte 1	0x35	cmd code
Byte 2	0xCA	XORed byte of cmd code
Wait for ACK		
Byte 3	0xFF	Special erase code MSB
Byte 4	0xFE	Special erase code LSB
Byte 5	0x01	XORed byte of bytes 3 to 4
Wait for ACK		

The host performs sector erase to erase sector 2, 4, 6 (the sector index starts from 0).



Byte 1	0x35	cmd code
Byte 2	0xCA	XORed byte of cmd code
Wait for ACK		
Byte 3	0x0	The number of sectors to be erased MSB
Byte 4	0x2	The number of sectors to be erased LSB
Byte 5	0x0	The first sector number MSB
Byte 6	0x1	The first sector number LSB
Byte 7	0x0	The second sector number MSB
Byte 8	0x3	The second sector number LSB
Byte 9	0x0	The third sector number MSB
Byte 10	0x5	The third sector number LSB
Byte 11	0x5	The XORed byte
Wait for ACK		

3.6 IIC Bootloader

3.6.1 IIC Bootloader Process

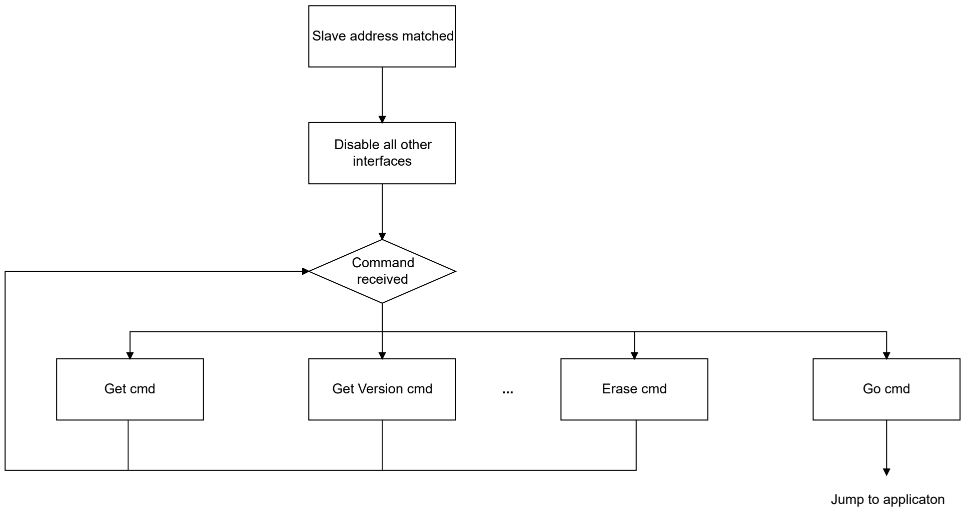


Figure 3-10 IIC Bootloader Flow Chart

3.6.2 Command Implementation

The application layer data exchanged between the host and the device are same comparing with USART bootloader，but the logical link layer data frame is different.

In USART transmission, each byte is a data frame.

The order and definition of the bits appearing on the IIC bus and how they are clocked are determined by IIC protocol. The IIC data frame is shown below:

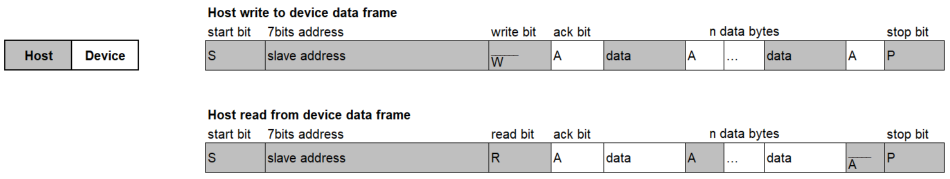


Figure 3-11 IIC Data Frame

NOTE: The slave address of device is 0x8.

3.6.2.1 GET Command

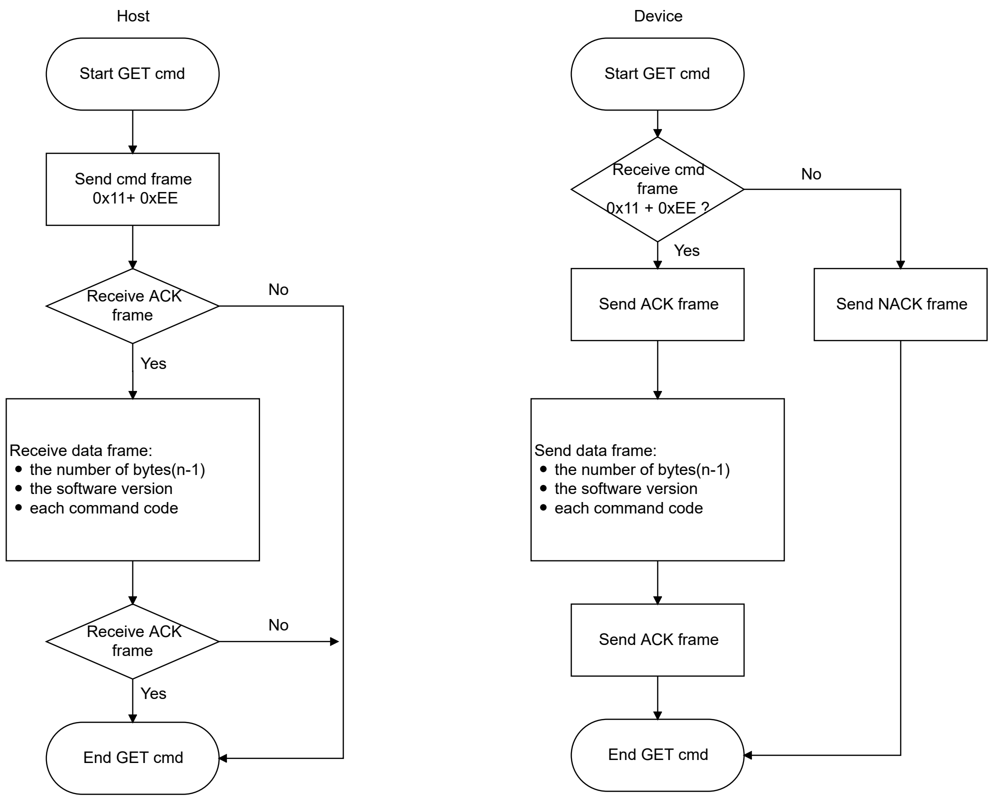


Figure 3-12 IIC Bootloader GET Flow Chart

The device sends the frames and bytes:

Frame 1	Byte 1	0xA3	ACK frame
Frame 2	Byte 1	0x7	The number of bytes to be transmitted (exact number -1)
	Byte 2	0x11	Software version
	Byte 3	0x11	GET code
	Byte 4	0x12	GET VERSION code
	Byte 5	0x13	GET ID code
	Byte 6	0x31	MEMORY READ code
	Byte 7	0x32	MEMORY WRITE code
	Byte 8	0x33	GO code
	Byte 9	0x35	MEMORY ERASE code
Frame 3	Byte 1	0xA3	ACK frame

3.6.2.2 GET VERSION Command

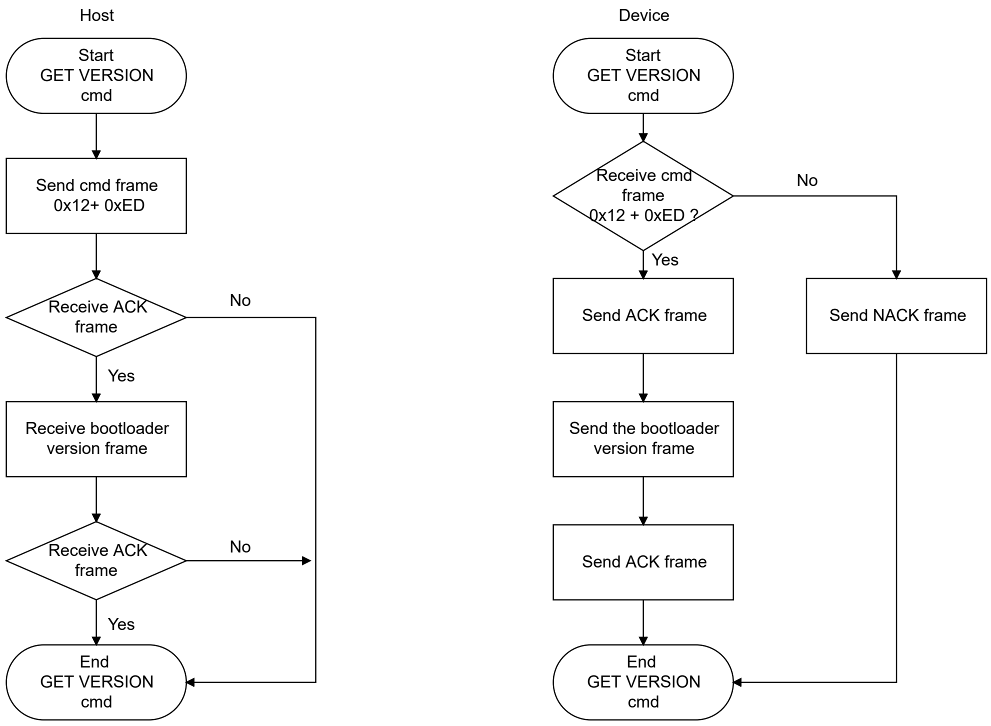


Figure 3-13 IIC Bootloader GET VERSION Flow Chart

The device sends the frames and bytes:

Frame 1	Byte 1	0xA3	ACK
Frame 2	Byte 1	0x11	The bootloader version
Frame 3	Byte 1	0xA3	ACK

3.6.2.3 GET ID Command

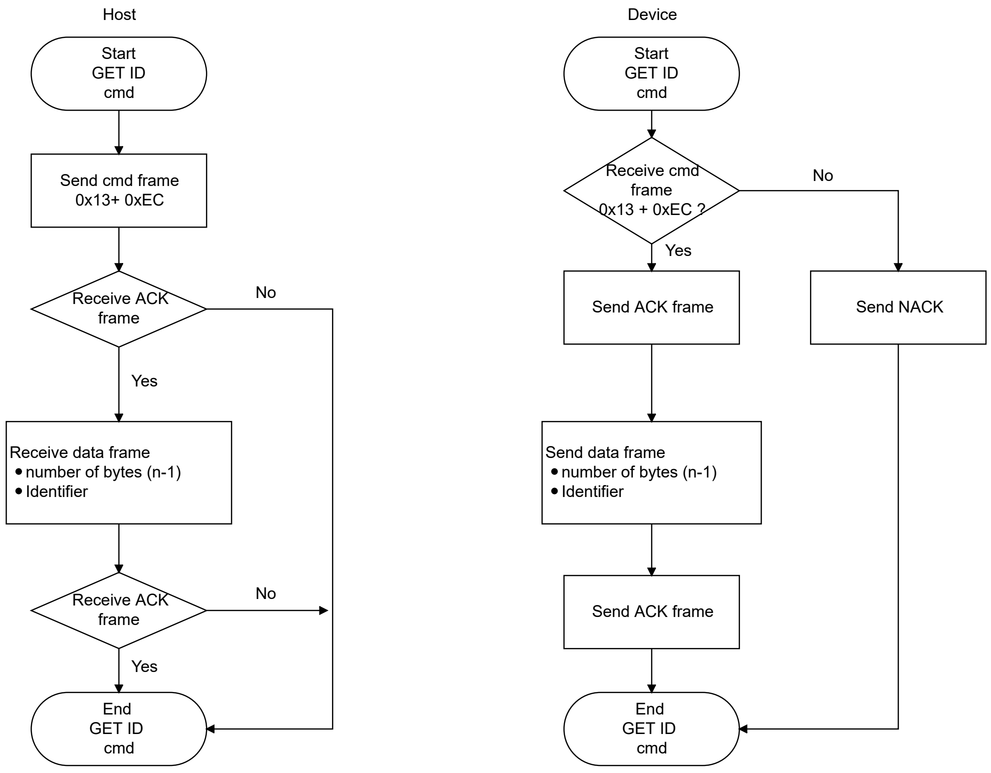


Figure 3-14 IIC Bootloader GET ID Flow Chart

The device sends the frames and bytes:

Byte1	0xA3	ACK
Byte2	0x3	The number of bytes
Byte3	0x23	ID MSB
Byte4	0x0	
Byte5	0x0	
Byte6	0x1	ID LSB
Byte7	0xA3	ACK

3.6.2.4 READ Command

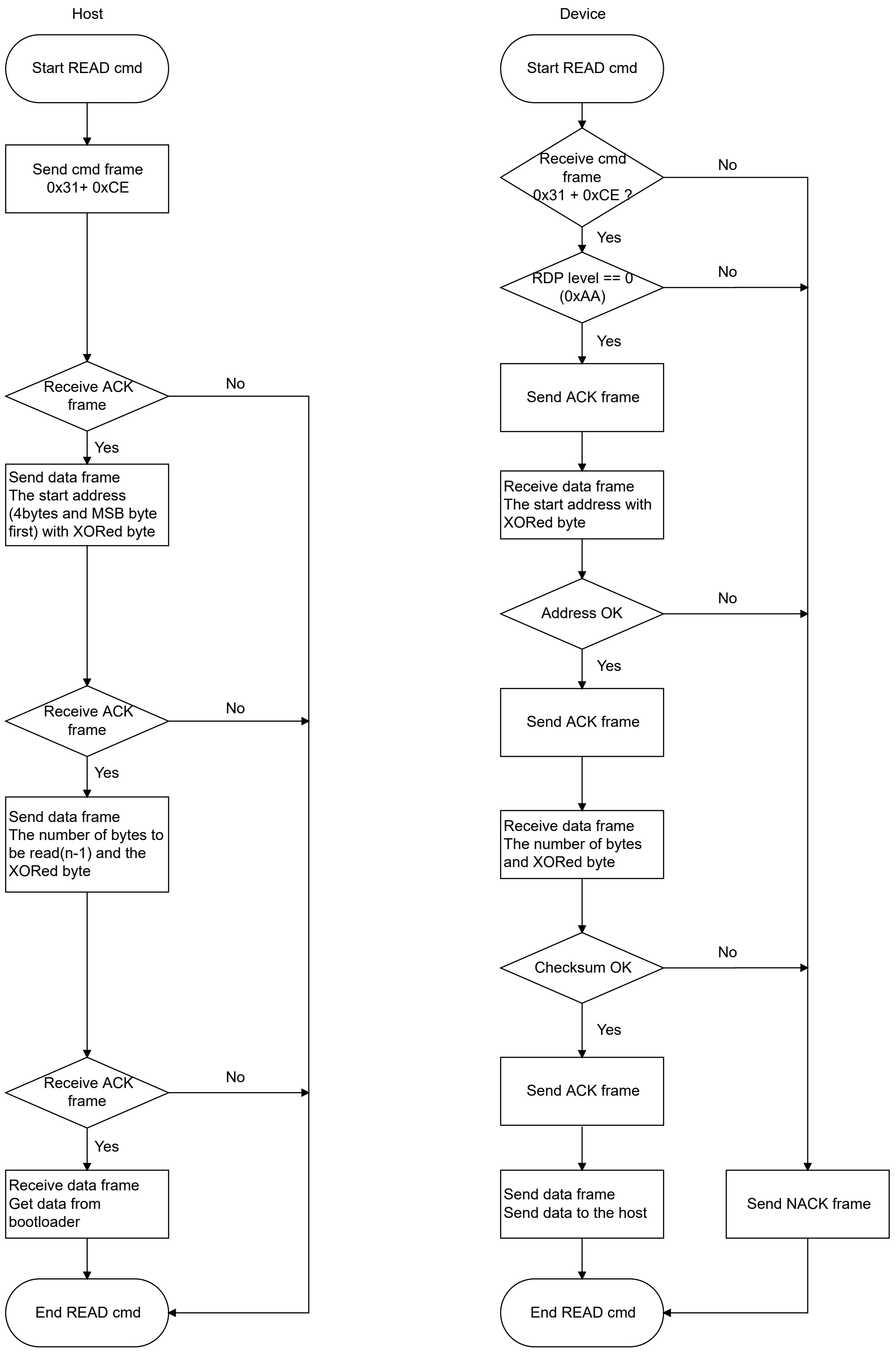
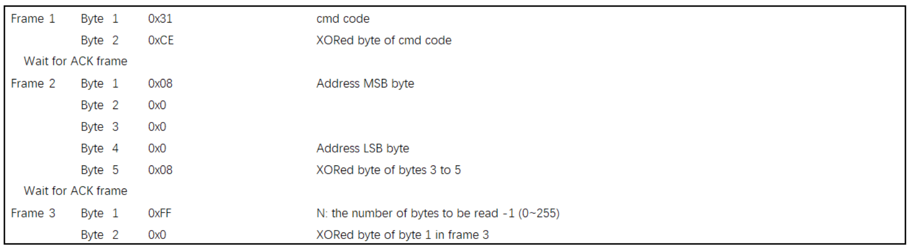

Host
Device
Start READ cmd
Send cmd frame 0x31+ 0xCE
Receive ACK frame
No
Yes
Send data frame The start address (4bytes and MSB byte first) with XORed byte
Send data frame The number of bytes to be read(n-1) and the XORed byte
Receive data frame Get data from bootloader
End READ cmd
Receive cmd frame 0x31 + 0xCE ?
RDP level == 0 (0xAA)
Send ACK frame
Receive data frame The start address with XORed byte
Address OK
Receive data frame The number of bytes and XORed byte
Checksum OK
Send data frame Send data to the host
Send NACK frame

Figure 3-15 IIC Bootloader READ Flow Chart

The host sends the frames and bytes:



Frame 1	Byte 1	0x31	cmd code
	Byte 2	0xCE	XORed byte of cmd code
Wait for ACK frame			
Frame 2	Byte 1	0x08	Address MSB byte
	Byte 2	0x0	
	Byte 3	0x0	
	Byte 4	0x0	Address LSB byte
	Byte 5	0x08	XORed byte of bytes 3 to 5
Wait for ACK frame			
Frame 3	Byte 1	0xFF	N: the number of bytes to be read -1 (0~255)
	Byte 2	0x0	XORed byte of byte 1 in frame 3

3.6.2.5 GO Command

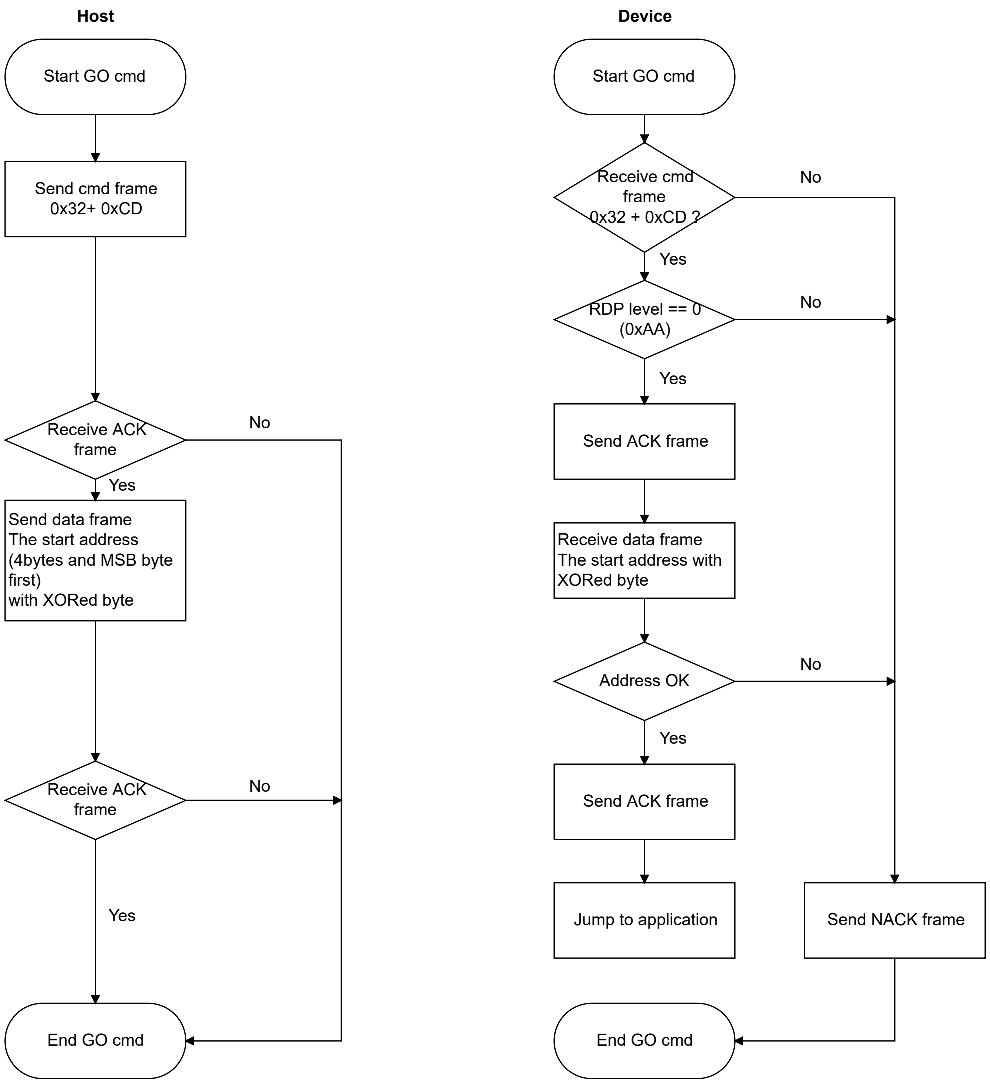
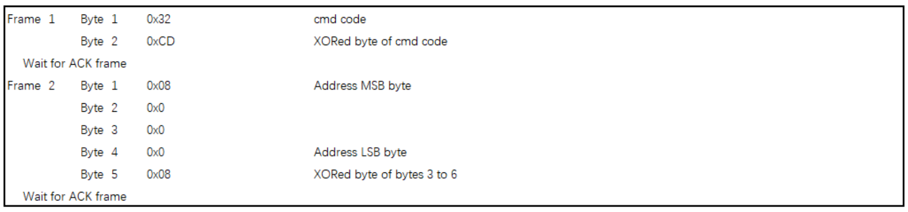


Figure 3-16 IIC bootloader GO flow chart

The host sends the frames and bytes:



Frame 1	Byte 1	0x32	cmd code
	Byte 2	0xCD	XORed byte of cmd code
Wait for ACK frame			
Frame 2	Byte 1	0x08	Address MSB byte
	Byte 2	0x0	
	Byte 3	0x0	
	Byte 4	0x0	Address LSB byte
	Byte 5	0x08	XORed byte of bytes 3 to 6
Wait for ACK frame			

3.6.2.6 WRITE command

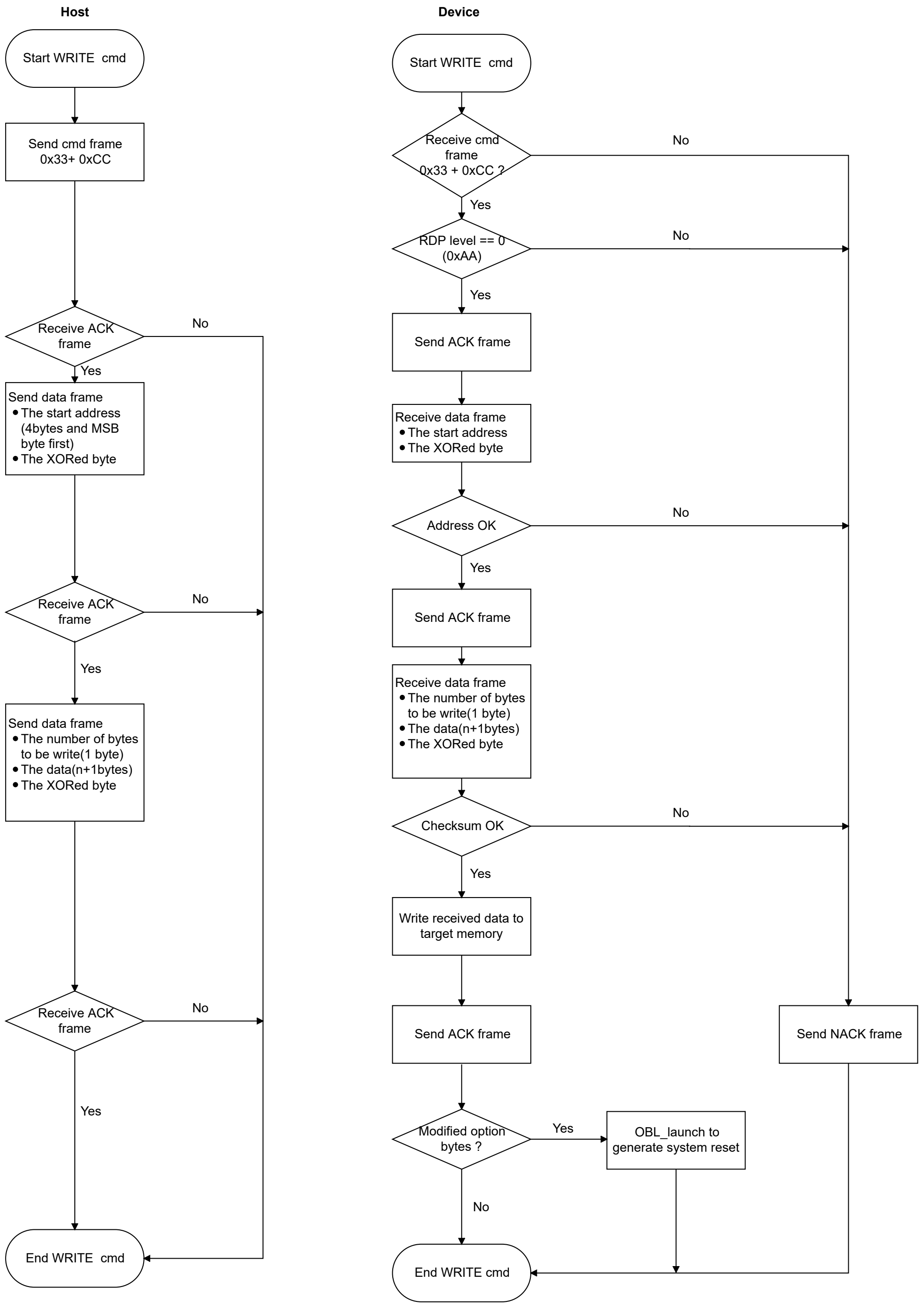


Figure 3-17 IIC Bootloader WRITE Flow Chart

The host sends the frames and bytes:

Frame 1	Byte 1	0x33	cmd code
	Byte 2	0xCC	XORed byte of cmd code
Wait for ACK frame			
Frame 2	Byte 1	0x08	Address MSB byte
	Byte 2	0x0	
	Byte 3	0x0	
	Byte 4	0x0	Address LSB byte
	Byte 5	0x08	XORed byte of bytes 1 to 4
Wait for ACK frame			
Frame 3	Byte 1	0x7	N: the number of bytes to be write -1 (0~255)
	Byte 2	0x11	The first byte of data
	Byte 3	0x11	
	Byte 4	0x11	
	Byte 5	0x11	
	Byte 6	0x22	
	Byte 7	0x22	
	Byte 8	0x22	
	Byte 9	0x22	The last byte of data
	Byte 10	0x7	XORed checksum
Wait for ACK frame			

3.6.2.7 ERASE command

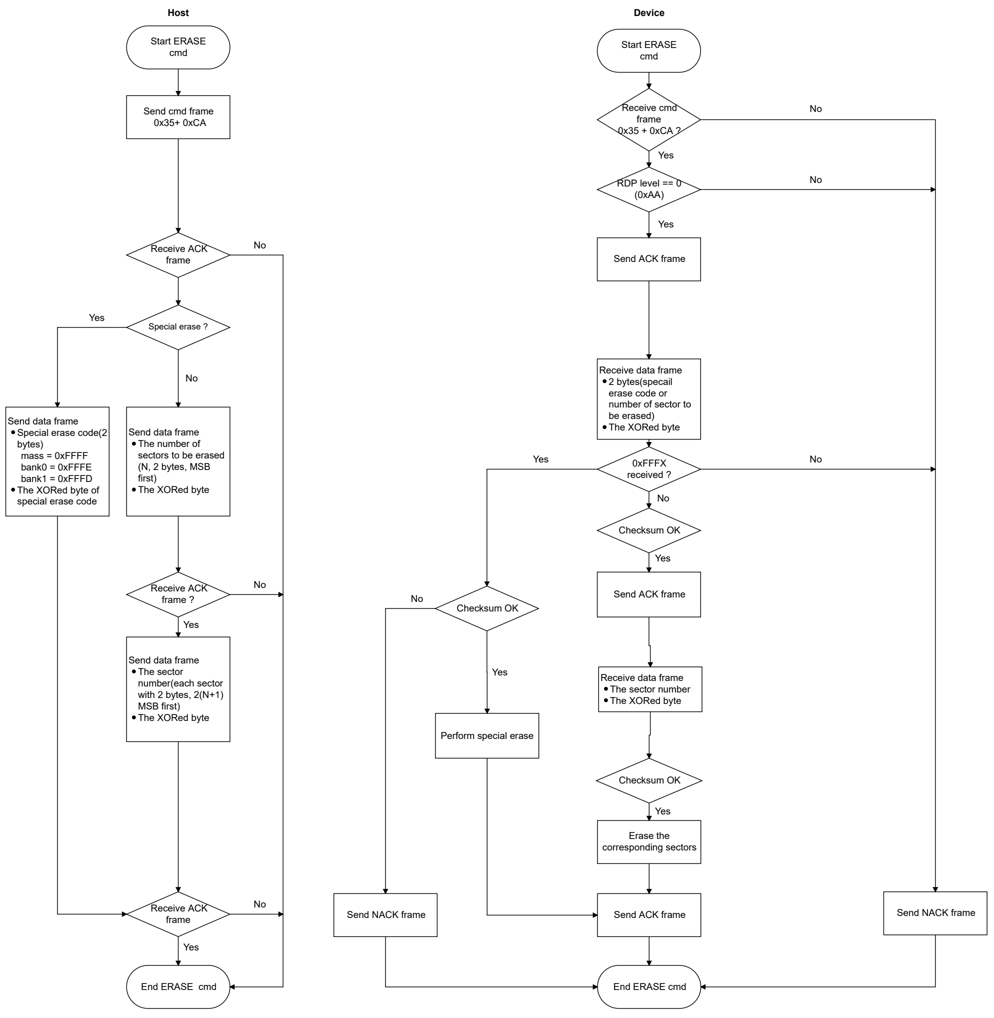
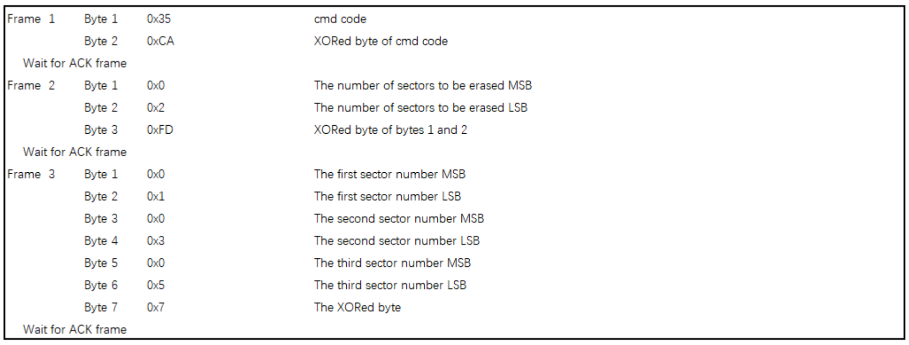


Figure 3-18 IIC Bootloader ERASE Flow Chart

The host sends the frames and bytes:

Frame 1	Byte 1	0x35	cmd code
	Byte 2	0xCA	XORed byte of cmd code
Wait for ACK frame			
Frame 2	Byte 1	0xFF	Special erase code MSB
	Byte 2	0xFE	Special erase code LSB
	Byte 3	0x01	XORed byte of bytes 3 to 4
Wait for ACK frame			



Frame 1	Byte 1	0x35	cmd code
	Byte 2	0xCA	XORed byte of cmd code
Wait for ACK frame			
Frame 2	Byte 1	0x0	The number of sectors to be erased MSB
	Byte 2	0x2	The number of sectors to be erased LSB
	Byte 3	0xFD	XORed byte of bytes 1 and 2
Wait for ACK frame			
Frame 3	Byte 1	0x0	The first sector number MSB
	Byte 2	0x1	The first sector number LSB
	Byte 3	0x0	The second sector number MSB
	Byte 4	0x3	The second sector number LSB
	Byte 5	0x0	The third sector number MSB
	Byte 6	0x5	The third sector number LSB
	Byte 7	0x7	The XORed byte
Wait for ACK frame			

Revision History

Date	Version	Changes
2024-04-10	Rev.A.2	• Modified part of the table data information in Section 3.1 Configuration. • Some editorial changes have been made.
2023-12-07	Rev.A.1	• Updated Figure 2-1. • Updated Table 2-1.
2023-11-30	Rev.A.0	Initial release