



---

## Guidelines for UART Bootloader on BlueNRG-LP/BlueNRG-LPS, and STM32WB0 series MCUs

### Introduction

BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices are very-low power Bluetooth® Low Energy (Bluetooth® LE) single-mode systems-on-chip, compliant with Bluetooth® specifications. The architecture core is a 32-bit Cortex-M0+ processor. This application note contains the specifications of the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series bootloader.

**Note:** *The document content is valid for the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices. Any reference to the BlueNRG-LP device and platform is also valid for the BlueNRG-LPS and STM32WB0 series devices and platforms. Any specific difference is highlighted whenever it is needed.*

## 1 UART bootloader configuration

To communicate with the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series bootloader the host UART has to be configured as follows:

- UART data 8-bit
- NO parity
- Stop bit 1
- NO flow control
- Baud rate range [500 – 460800]

The bootloader is configured to use the UART pins:

- BlueNRG-LP and STM32WB06, STM32WB07:
  - UART RX: PA8
  - UART TX: PA9
- BlueNRG-LPS and STM32WB05, STM32WB09:
  - UART RX = PB0
  - UART TX = PA1

*Note:*

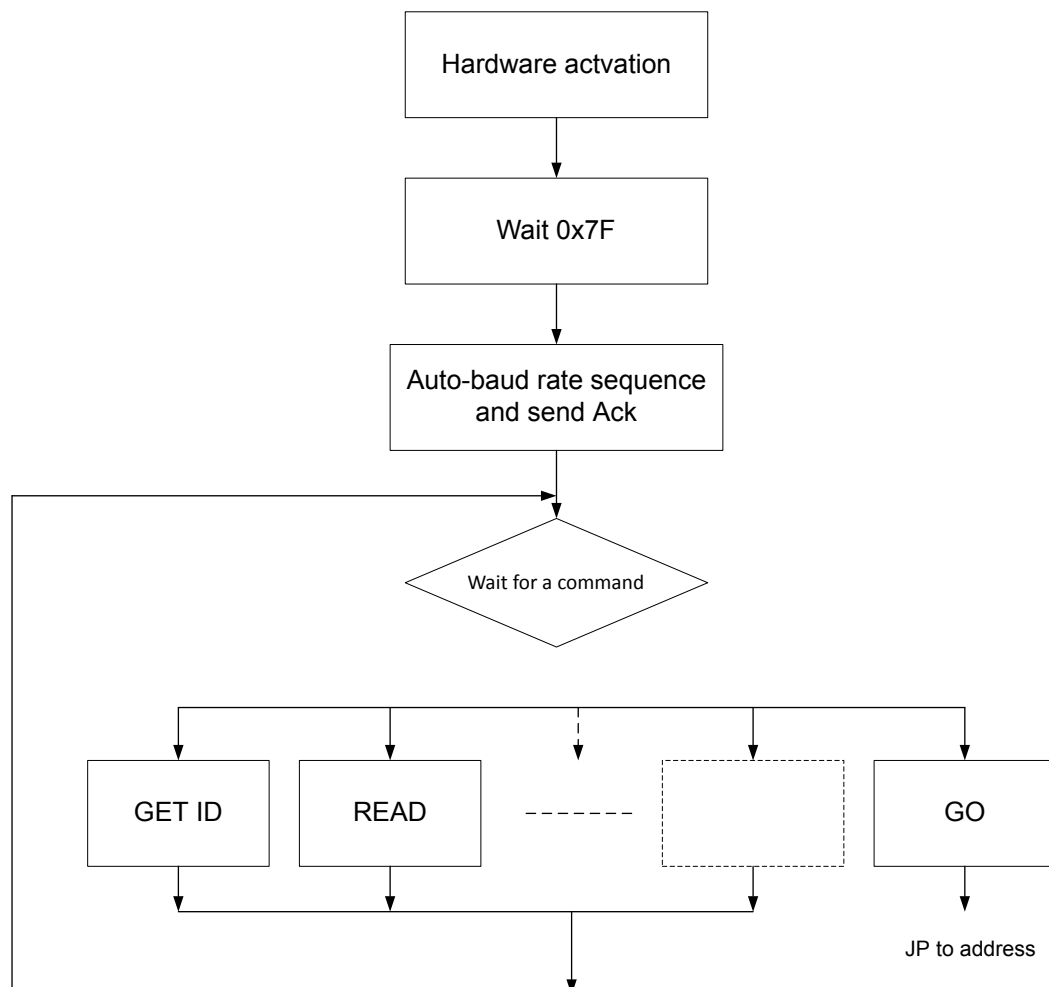
1. Bootloader is available only on pins PA8, PA9 (BlueNRG-LP, STM32WB06, STM32WB07) and PA1, PB0 (BlueNRG-LPS/STM32WB05, STM32WB09)
2. Bootloader is activated by hardware by forcing PA10 high during hardware reset (otherwise, the application residing in flash is launched)

## 2 UART bootloader activation

The BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series bootloaders are activated by forcing PA10 high at device hardware reset. Once the bootloader is activated, the firmware initializes the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series serial interface and starts a procedure to auto-detect the host UART baud rate and begins to scan the USART RX line pin, waiting for the 0x7F data frame from the host: one start bit, 0x7F data bits, no parity bit and one stop bit.

Once the baud rate is calculated an acknowledge byte (0x79) is returned to the host, which signals that the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series are ready to receive commands.

**Figure 1. UART bootloader for BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices**



DT57267V1

### 3 UART bootloader commands

The table below lists the supported commands, fully detailed in the following subsections.

**Table 1. BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series UART bootloader commands**

Command	Command code	Command description
Get List Command	0x00	Gets the version and the allowed commands supported by the current version of the bootloader
Get Version	0x01	Gets the bootloader version
Get ID	0x02	Gets the chip ID
Read Memory	0x11 <sup>(1)</sup>	Reads up to 256 bytes of memory starting from an address specified by the application
Go	0x21 <sup>(1)</sup>	Jumps to user application code located in the internal flash memory or in RAM
Write Memory	0x31 <sup>(1)</sup>	Writes up to 256 bytes to the RAM or flash memory starting from an address specified by the application
Erase	0x43	Erases from one to all the flash memory pages
Readout Protect	0x82	Enables the read protection
Readout Unprotect	0x92	Disables the read protection
OTP Write	0xA2	Write a 32-bit word in the OTP sector

1. This command is disabled when readout protection is enabled.

All communications from the host to the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices are safe because they are verified by:

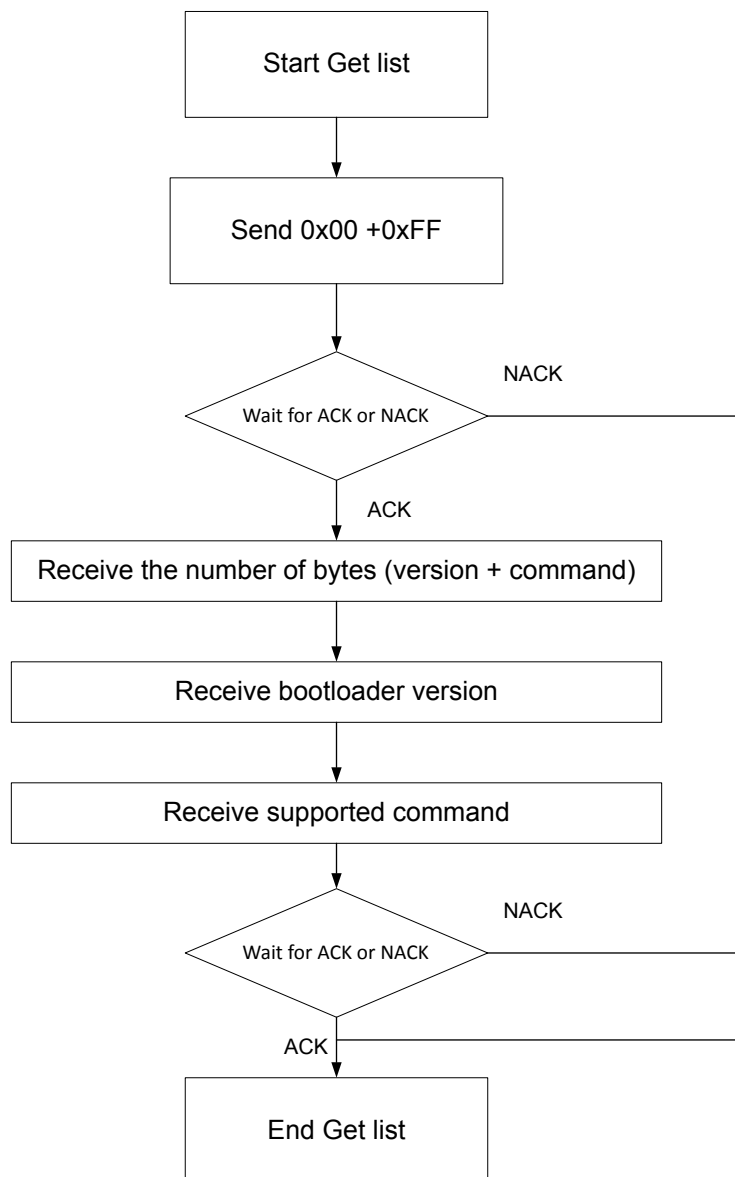
- Checksum: received blocks of data bytes are XORed. A byte containing the computed XOR of all previous bytes is added to the end of each communication (checksum byte). By XORing all received bytes, data + checksum, the result at the end of the packet must be 0x00
- For each bootloader command, the host sends a byte and its complement
- Each packet is either accepted (ACK answer) or discarded (NACK answer):
  - ACK = 0x79
  - NACK = 0x1F

### 3.1 Get List command

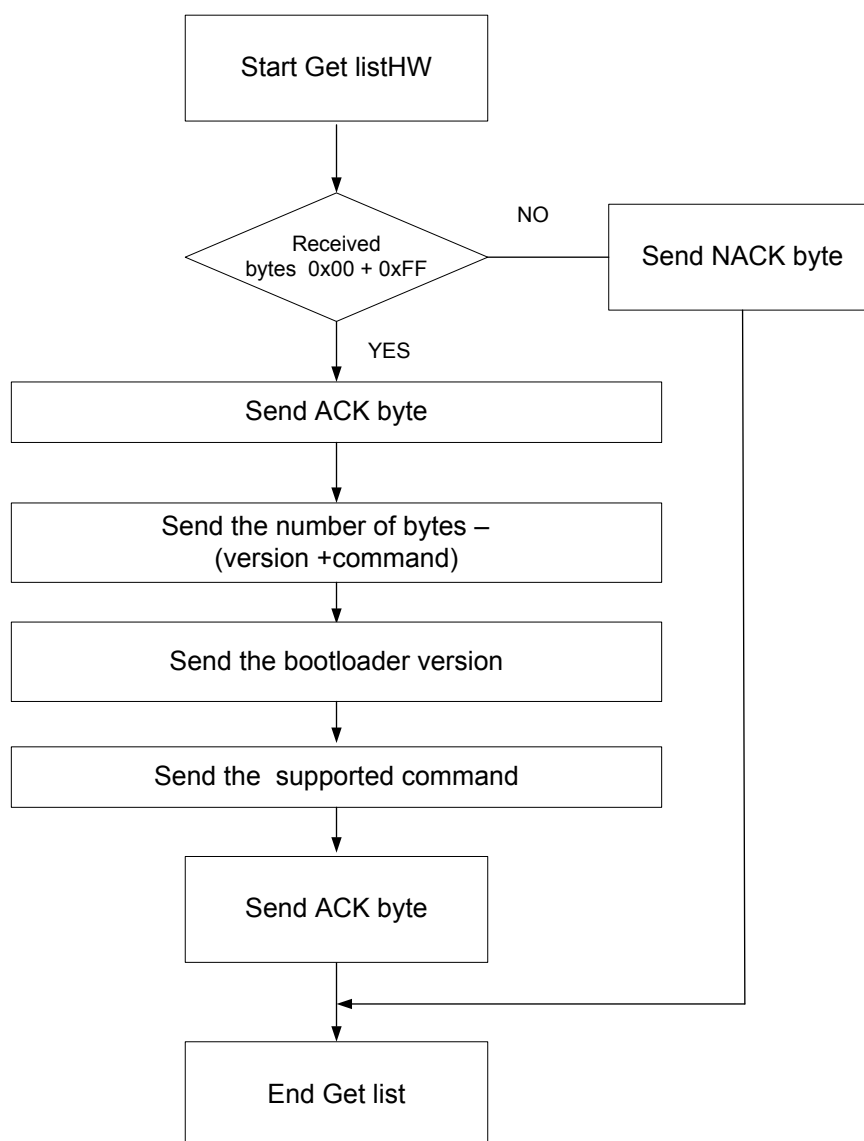
The Get List command allows getting the version of the bootloader and the supported commands.

When the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series bootloader receives the Get List command, it transmits the bootloader version and the supported command codes to the host.

Figure 2. Get List command: host side



DT57268V1

**Figure 3. Get list command: device side**


DT57269V1

The BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series send the bytes as follows:

Byte 1: ACK

Byte 2: N = 9 = the number of bytes to follow -1 except current and ACKs.

Byte 3: Bootloader version (0 < Version <= 255), example: 0x01 = Version 1.0

Byte 4: 0x00 – Get List command

Byte 5: 0x01 – Get Version command

Byte 6: 0x02 – Get ID command

Byte 7: 0x11 – Read Memory command

Byte 8: 0x21 – Go command

Byte 9: 0x31 – Write Memory command

Byte 10: 0x43 – Erase command

Byte 11: 0x82 – Readout Protect command

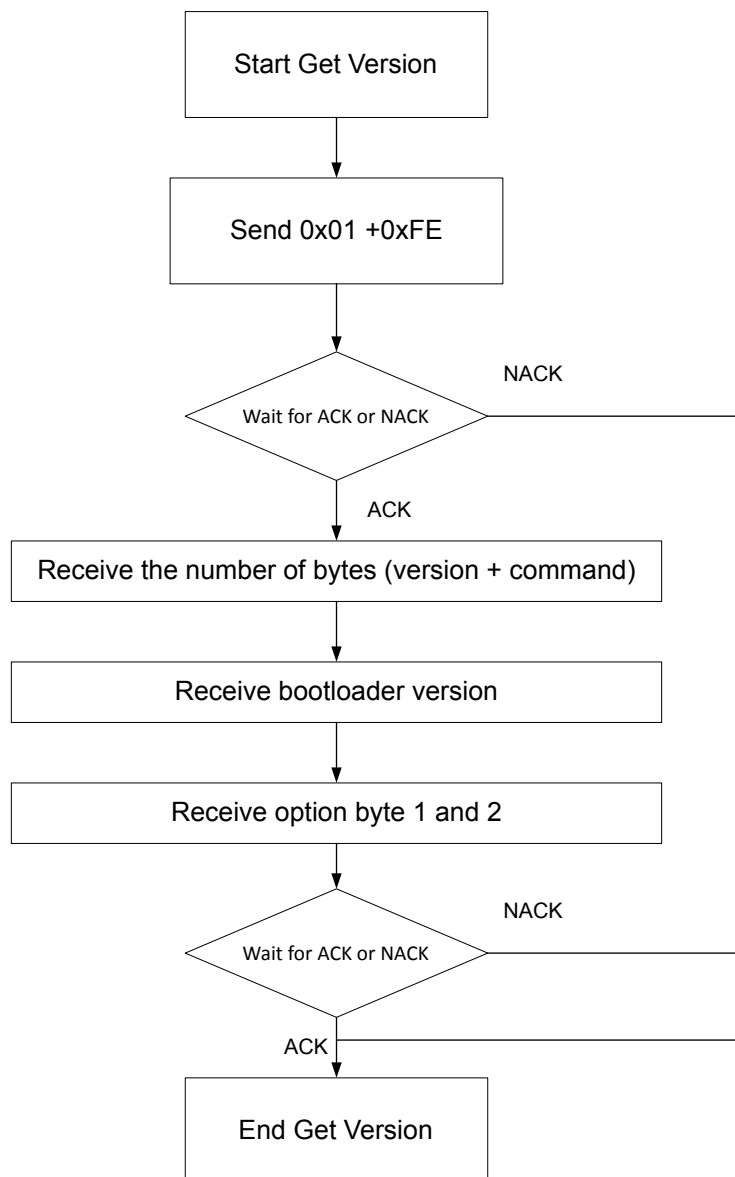
Byte 12: 0x92 – Readout Unprotect command

Byte 13: ACK. This is the last byte

## 3.2 Get Version command

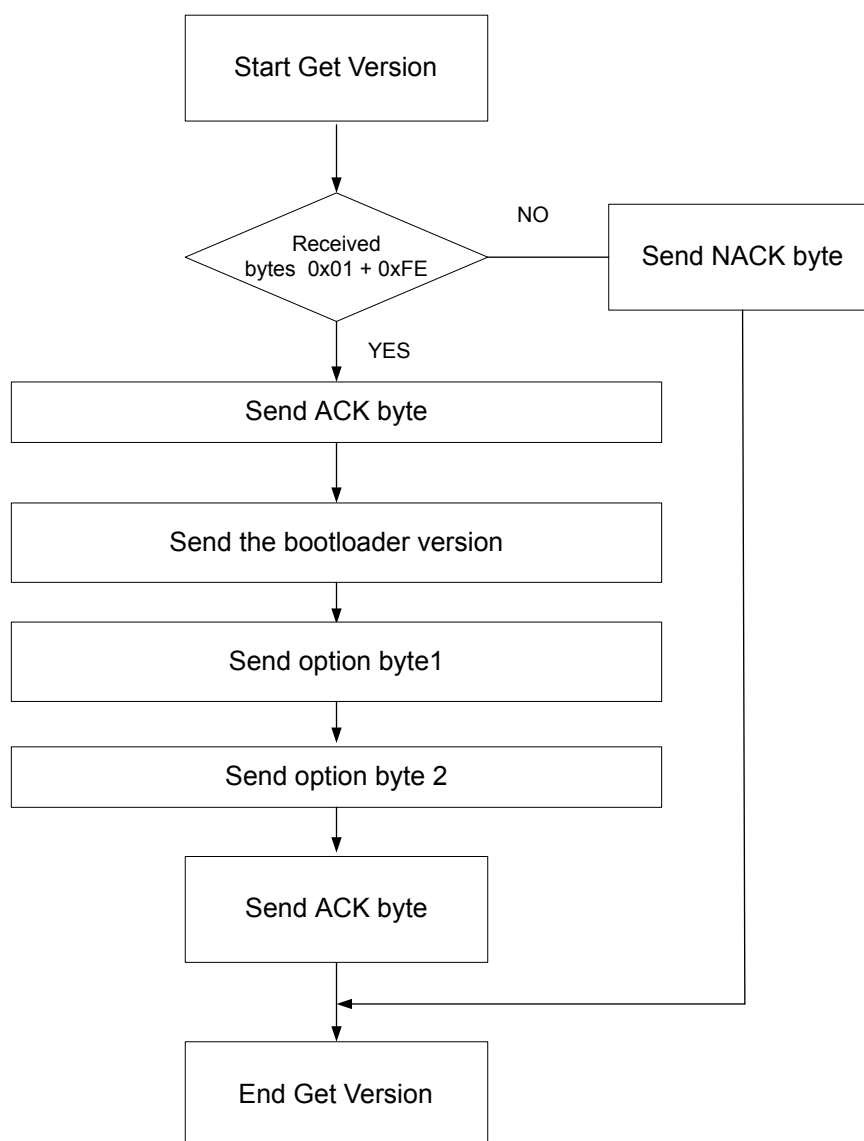
The Get Version command is used to get the bootloader version. When the bootloader receives the command, it transmits the information shown below to the host.

Figure 4. Get Version command: host side



DT57270V1

Figure 5. Get Version command: device side



DT57271V1

The BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices send the bytes as follows:

Byte 1: ACK

Byte 2: Bootloader version ( $0 < \text{Version} \leq 255$ ), example: 0x01 = Version 1.0

Byte 3: Option byte 1: 0x00

Byte 4: Option byte 2: 0x00

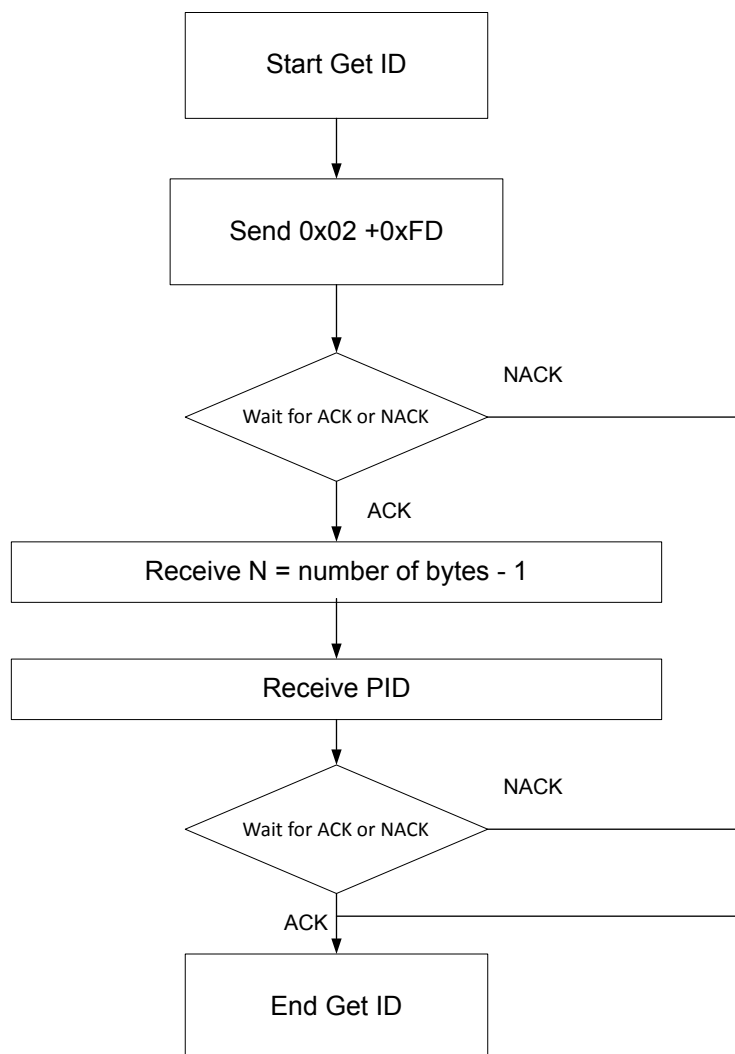
Byte 5: ACK



### 3.3 Get ID command

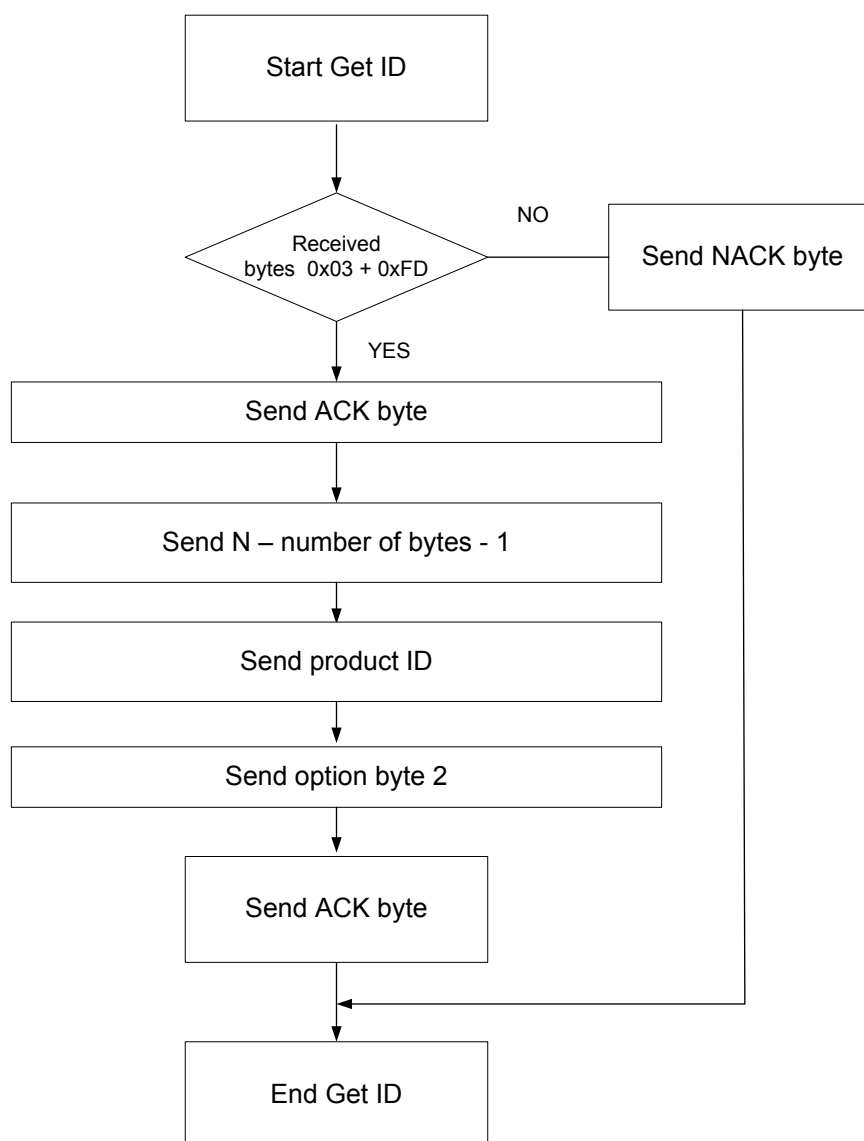
The Get ID command is used to get the version of the chip ID (identification). When the bootloader receives the command, it transmits the product ID to the host.

Figure 6. Get ID command: host side



DT57272V1

Figure 7. Get ID command: device side



DT57273V1

The BlueNRG-LP/BlueNRG-LPS/STM32WB05/STM32WB06/STM32WB07 devices send the bytes as follows:

Byte 1: ACK

Byte 2: 0x02: the number of bytes -1, except for current byte and ACKs

Byte 3-5: PID

Byte 6: ACK

The STM32WB09 device sends the bytes as follows:

Byte 1: ACK

Byte 2: 0x03: the number of bytes -1, except for current byte and ACKs

Byte 3-6: PID

Byte 7: ACK

- Note:
1. The Get ID command format returns three bytes for BlueNRG-LP/BlueNRG-LPS/STM32WB05/STM32WB06/STM32WB07:
    - BYTE1: Metal fix version (for example cut 1.0 is 0)
    - BYTE2: Mask set version (for example cut 1.0 is 1)
    - BYTE3: is split into two nibbles: 0xHL:
      - H is the product ID:
        - 3 means BlueNRG-LP/BlueNRG-LPS/STM32WB05/STM32WB06/STM32WB07
      - L is the flash size code:
        - F: 256 Kbyte (for the BlueNRG-LP/STM32WB06/STM32WB07)
        - B: 192 Kbyte (for the BlueNRG-LPS/STM32WB05)
  2. The Get ID command format returns four bytes for the STM32WB09:
    - BYTE1: Metal fix version (for example cut 1.0 is 0)
    - BYTE2: Mask set version (for example cut 1.0 is 1)
    - BYTE3: Product ID, 6 means STM32WB09
    - BYTE4: flash size code as follow:
      - 0x1F: 512 Kbytes

### 3.4 Read Memory command

The Read Memory command is used to read data from any valid memory address in RAM and flash memory.

When the bootloader receives the Read Memory command, it transmits the ACK byte to the application. After which, the bootloader waits for an address (4 bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte. 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.

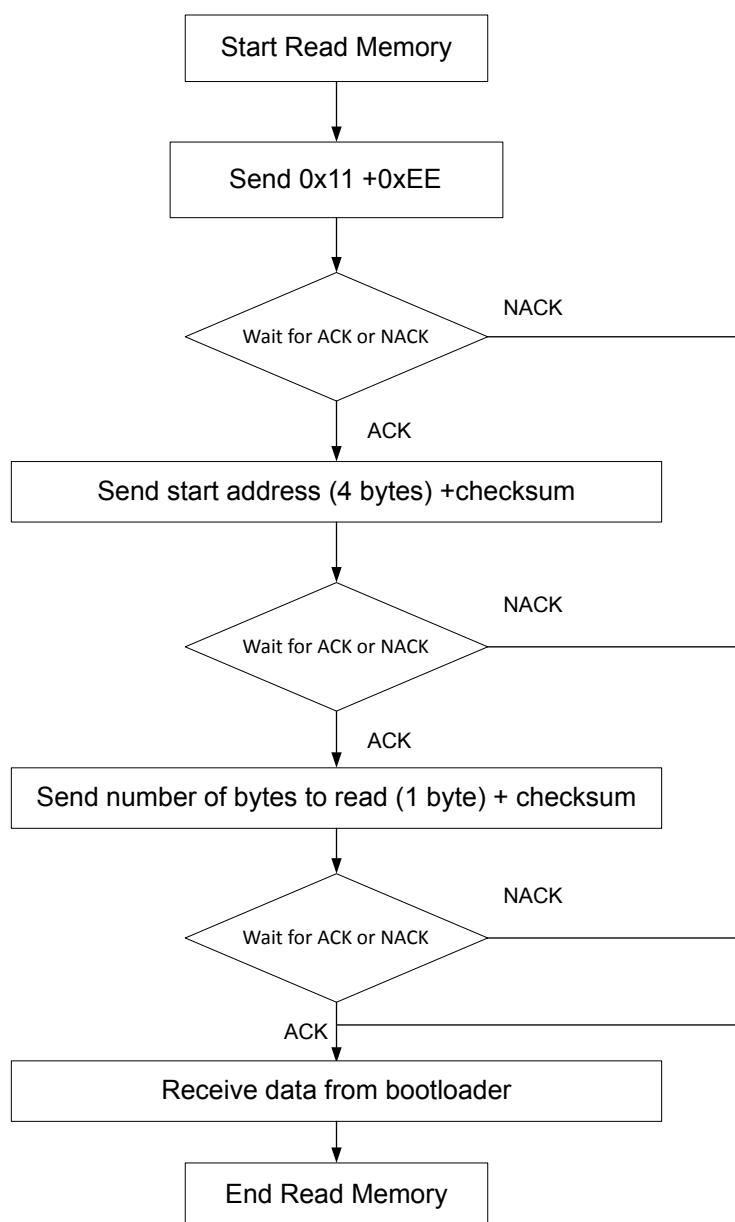
When the address is valid and the checksum is correct, the bootloader waits for N that is the number of bytes to be transmitted – 1 and for its complemented byte (checksum).

If the checksum is correct it 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.

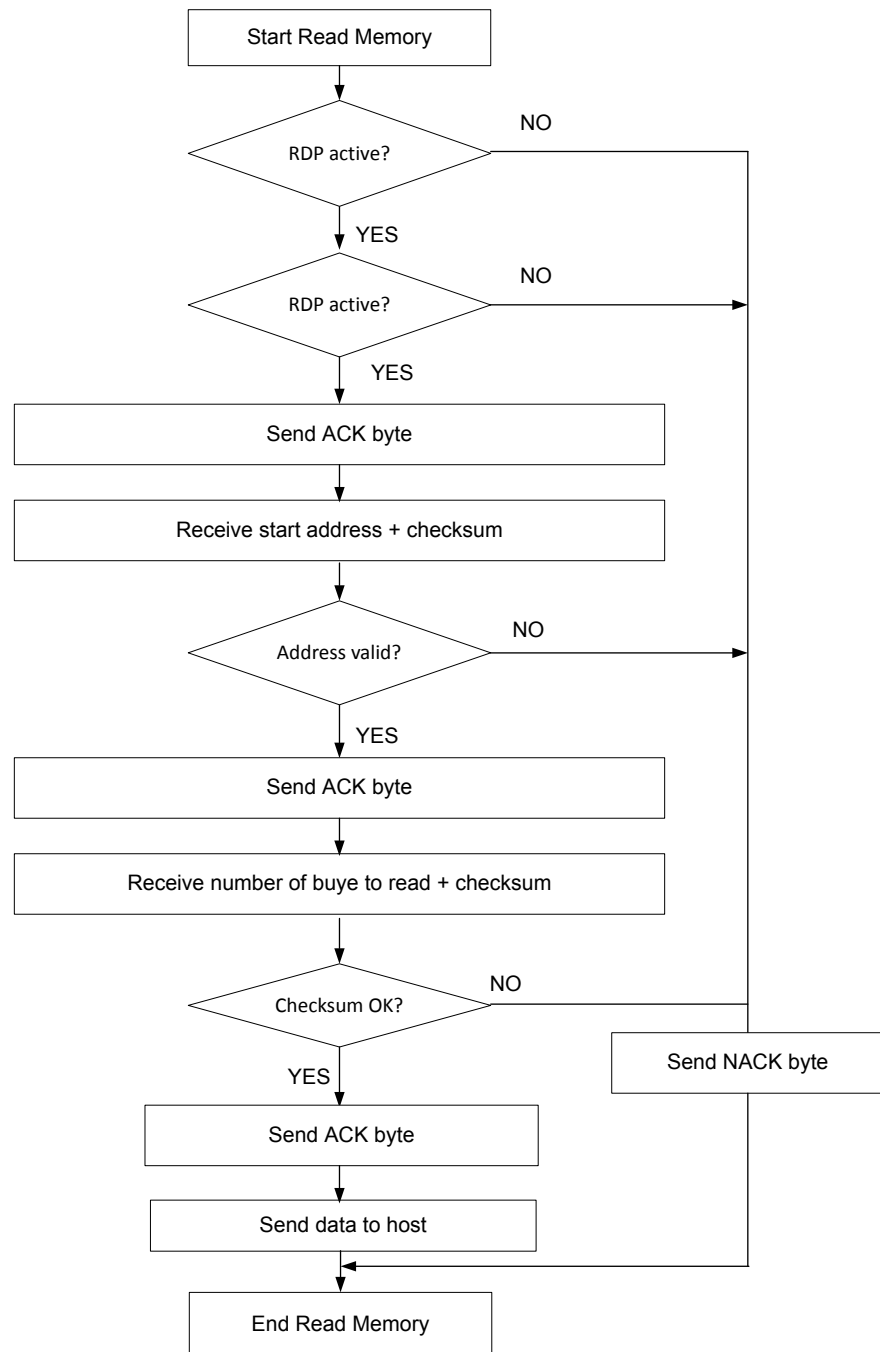
If the readout protection is active, a NACK byte is sent to the host when the Read Memory command is received.

Figure 8. Read Memory command: host side



DT57274V1

Figure 9. Read Memory command: device side



DT57275V1

The host sends bytes to the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series as follows:

Byte 1: 0x11

Byte 2: 0xEE

Wait for ACK

Byte 3 to 6: start address

- Byte 3: MSB
- Byte 6: LSB

Byte 7: Checksum: XOR of address bytes

Wait for ACK

Byte 8: Number of bytes to read -1 ( $0 < N \leq 255$ )

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

### 3.5

## Go command

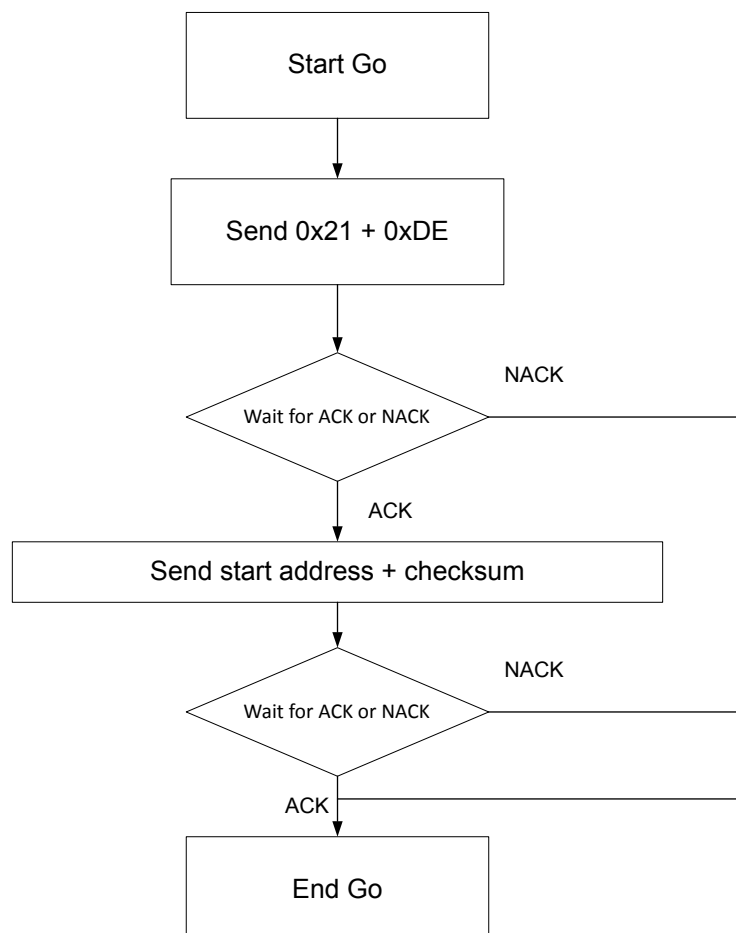
The Go command is used to execute the downloaded code or any other code by jumping to an address specified by the application. When the bootloader receives the Go command, it transmits the ACK byte to the application. After which, the bootloader waits for an address (4 bytes, byte 1 is the address MSB and byte 4 is LSB) and a checksum byte.

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 actions:

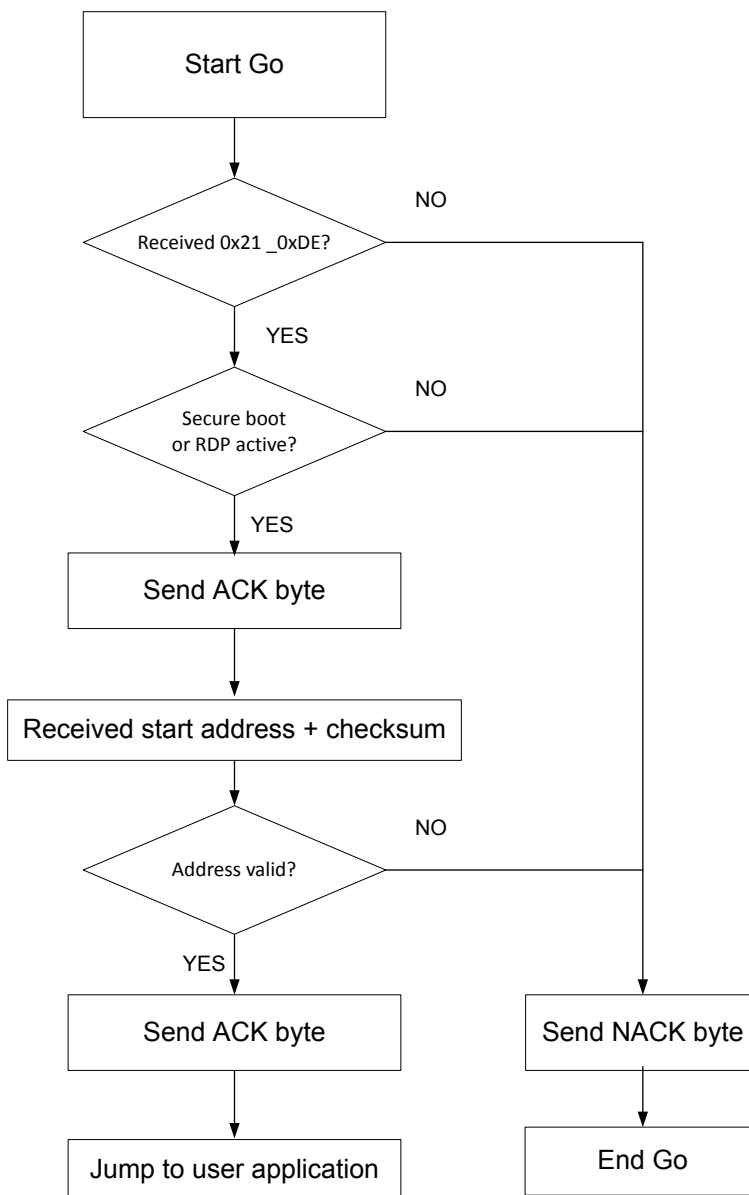
- initializes the registers of the peripherals used by the bootloader to their default reset values
- initializes the user application main stack pointer
- jumps to the memory location programmed in the received 'address + 4' (which corresponds to the address of the application reset handler). For example, if the received address is 0x10040000, the bootloader jumps to the memory location programmed at address 0x10040004.

Figure 10. Go command: host side



DT57276V1

Figure 11. Go command: device side



DT5727V1

The host sends bytes to the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series as follows:

Byte 1: 0x21

Byte 2: 0xDE

Wait for ACK

Byte 3 – 6: start application address

- Byte 3: MSB
- Byte 6: LSB

Byte 7: checksum: XOR of address bytes

Wait for ACK

### 3.6 Write Memory command

The Write Memory command is used to write data to any valid memory address RAM or flash memory.

When the bootloader receives the Write Memory command, it transmits the ACK byte to the application. After which, the bootloader waits for an address (4 bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte.

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 performs these actions:

- 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 is 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 BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series is 256 bytes.

If the readout protection is active, a NACK byte is sent to the host when the Write Memory command is received.

Figure 12. Write Memory command: host side

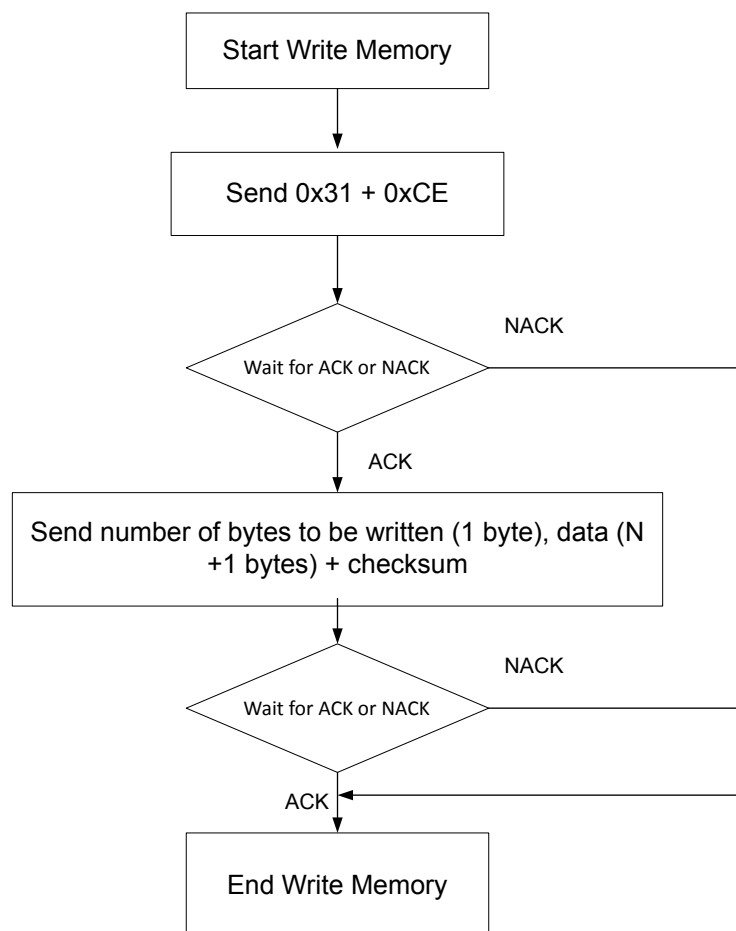
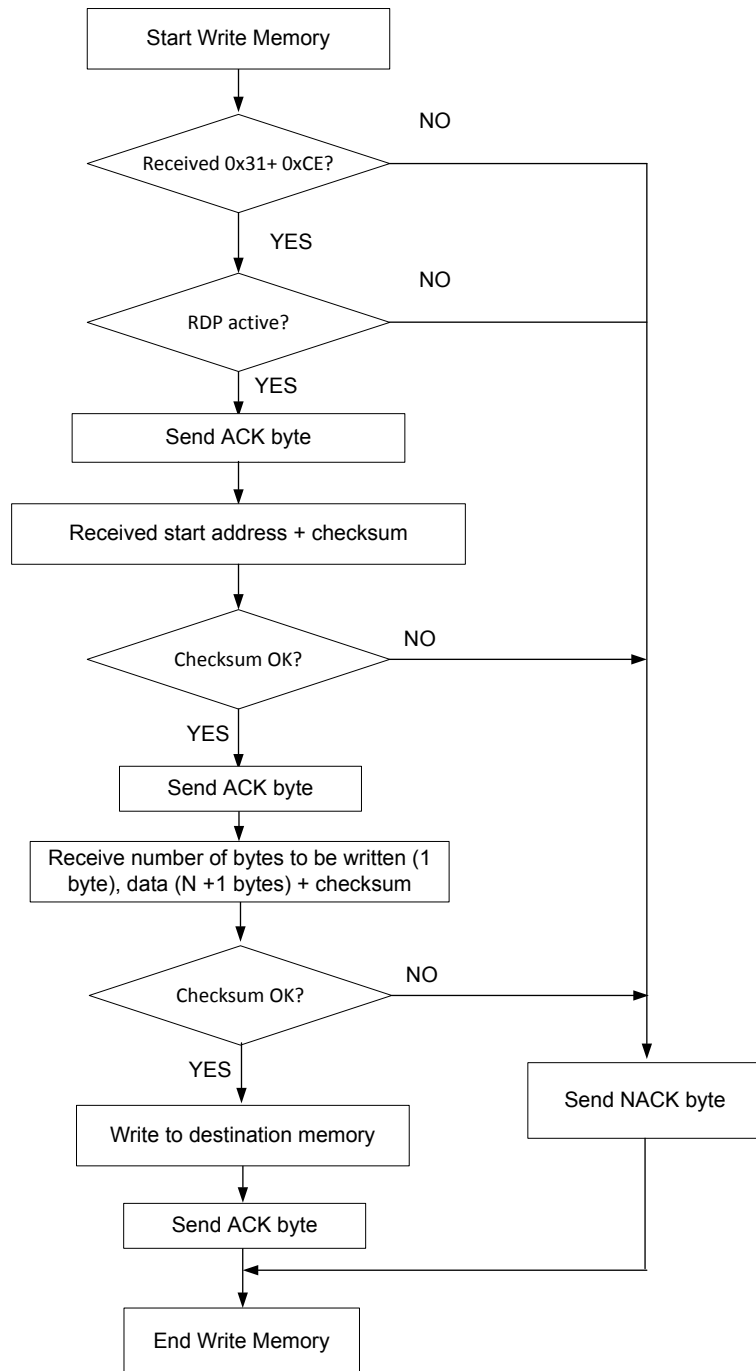




Figure 13. Write Memory command: device side



DT57279V1

The host sends the bytes to the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series 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 (byte 3, byte 4, byte 5, byte 6)

Wait for ACK  
 Byte 8: number of bytes to be received ( $0 < N \leq 255$ )  
 N + 1 data bytes: (Max. 256 bytes)  
 Checksum byte: XOR (N, N+1 data bytes)  
 Wait for ACK

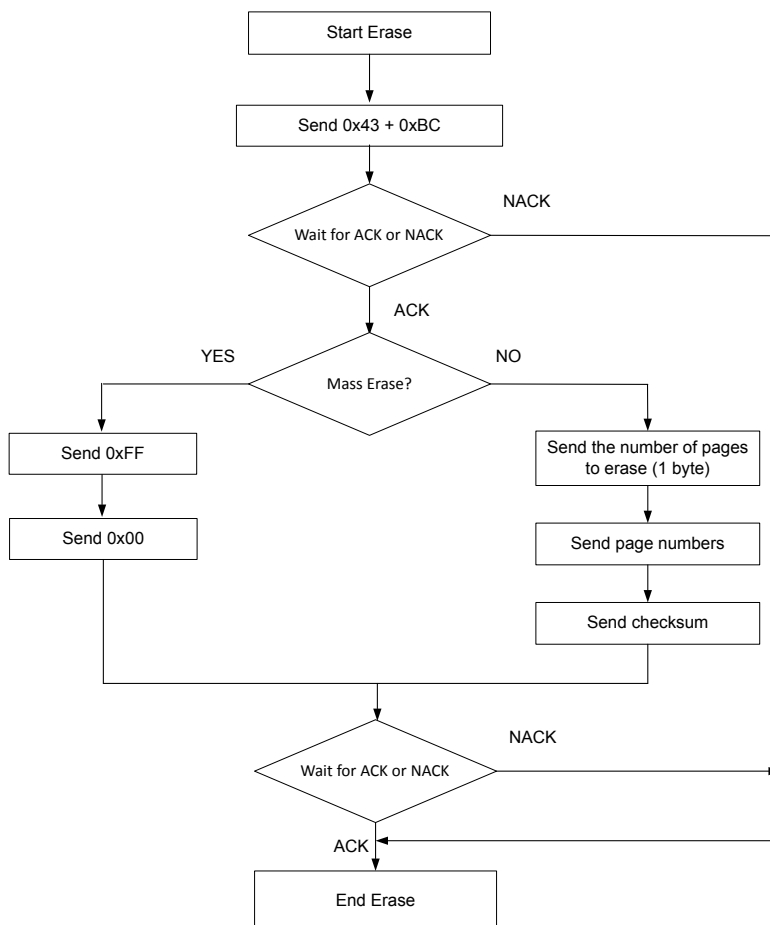
### 3.7 Erase Memory command

The Erase Memory command allows the host to erase flash memory pages. When the bootloader receives the Erase Memory command, it transmits the ACK byte to the host. After which, 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, the bootloader then 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 are:

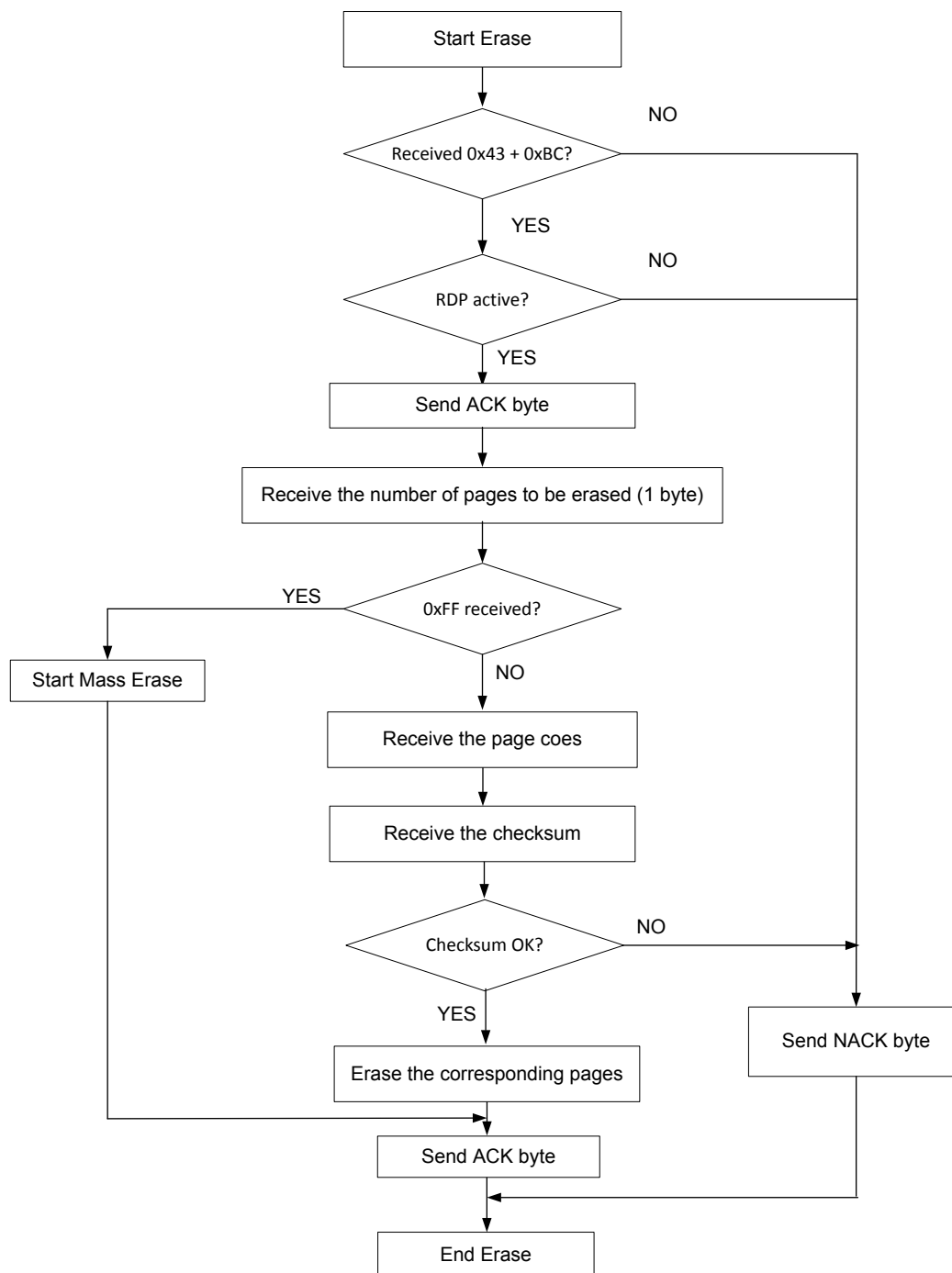
1. the bootloader receives one byte that contains N, the number of pages to be erased – 1. N= 255 is reserved for mass erase request. For  $0 < N \leq 79$ , N + 1 pages are erased.
2. the bootloader receives (N + 1) bytes, each byte containing a page number

**Figure 14. Erase Memory command: host side**



DT157280V1

Figure 15. Erase Memory command: device side



DT57281V1

The host sends bytes to the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series 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 mass erase) or ((N+1 bytes (page numbers) and then checksum XOR(N, N+1 bytes))

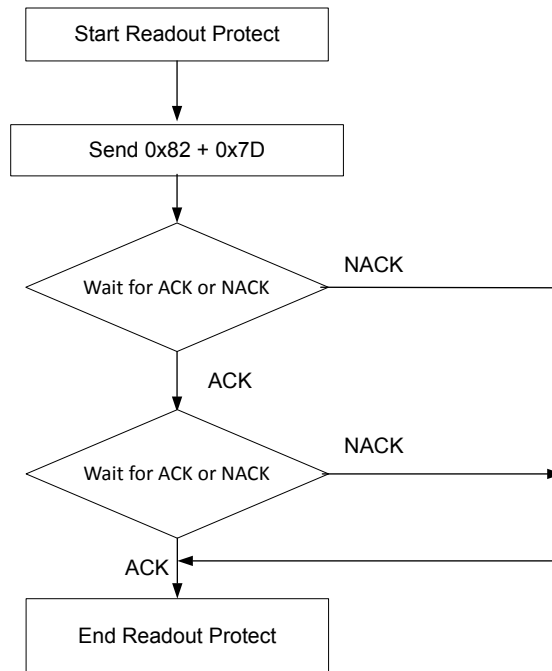
Wait for ACK

### 3.8 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 which, the bootloader enables the read protection for the flash memory.

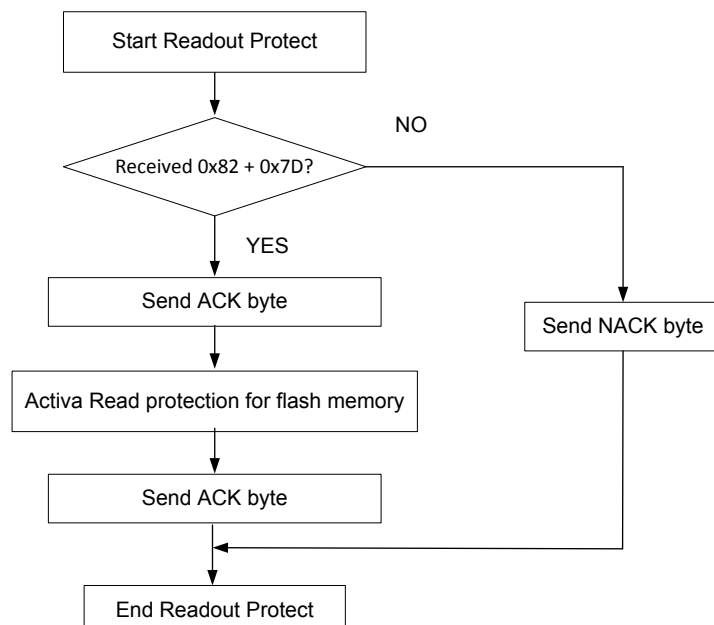
At the end of the Readout Protect command, the bootloader transmits the ACK byte to signal the end of the command.

Figure 16. Readout Protect command: host side



DT57282V1

Figure 17. Readout Protect command: device side



DT57283V1

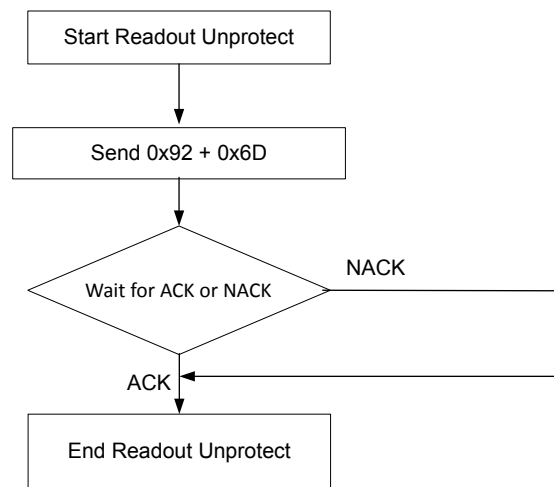
### 3.9 Readout unprotect command

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

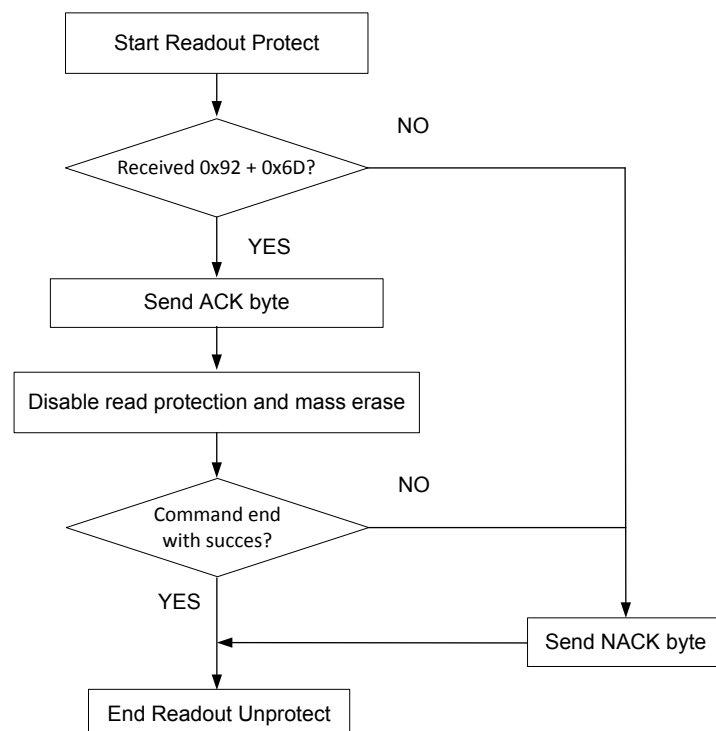
At the end of the Readout Unprotect command, the bootloader transmits an ACK and generates a system Reset. If other bootloader commands should be executed, the UART bootloader activation needs to be re-executed, that is HW reset plus PA10 high, as described in the UART bootloader activation.

**Figure 18. Readout unprotect command: host side**



DT57284V1

**Figure 19. Readout unprotect command: device side**



DT57285V1

### 3.10 OTP Write command

The OTP Write command is used to write a 32-bit word to any valid memory address in OTP memory. When the bootloader receives the OTP Write command, it transmits the acknowledgment (ACK) byte to the application. Afterward, the bootloader waits for an address (4 bytes, with byte 1 as the most significant byte (MSB) and byte 4 as the least significant byte (LSB)) and a checksum byte. If the received address is valid and the checksum is correct, the bootloader transmits an ACK byte; otherwise, it transmits a negative acknowledgment (NACK) byte and aborts the command. When the address is valid and the checksum is correct, the bootloader performs the following actions:

- Receives the user data (4 bytes) and the checksum (XOR of all data bytes).
- Programs the user data to memory at the received address.

At the end of the command, if the OTP write operation is successful, the bootloader transmits the ACK byte; otherwise, it transmits a NACK byte to the application and aborts the command.

Figure 20. OTP Write command: host side

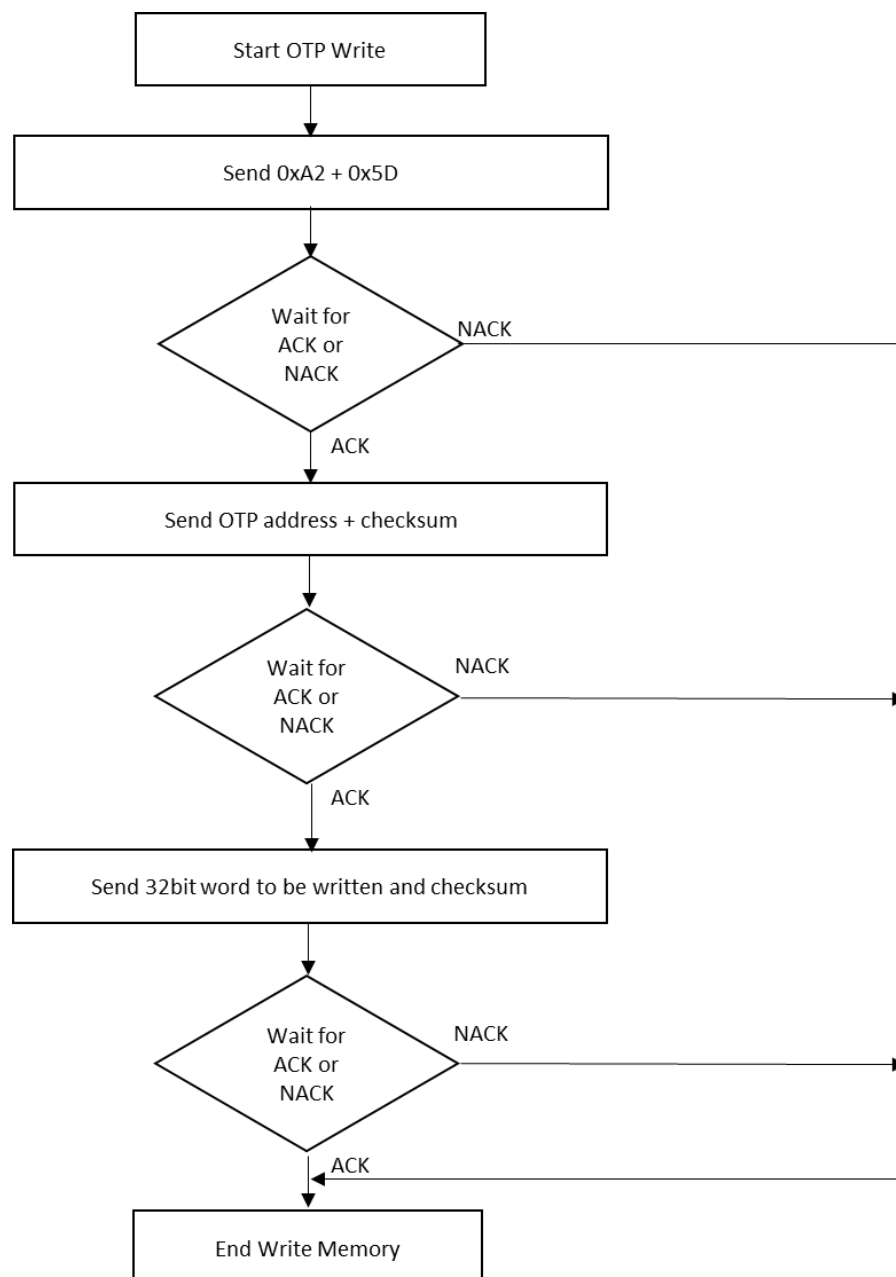
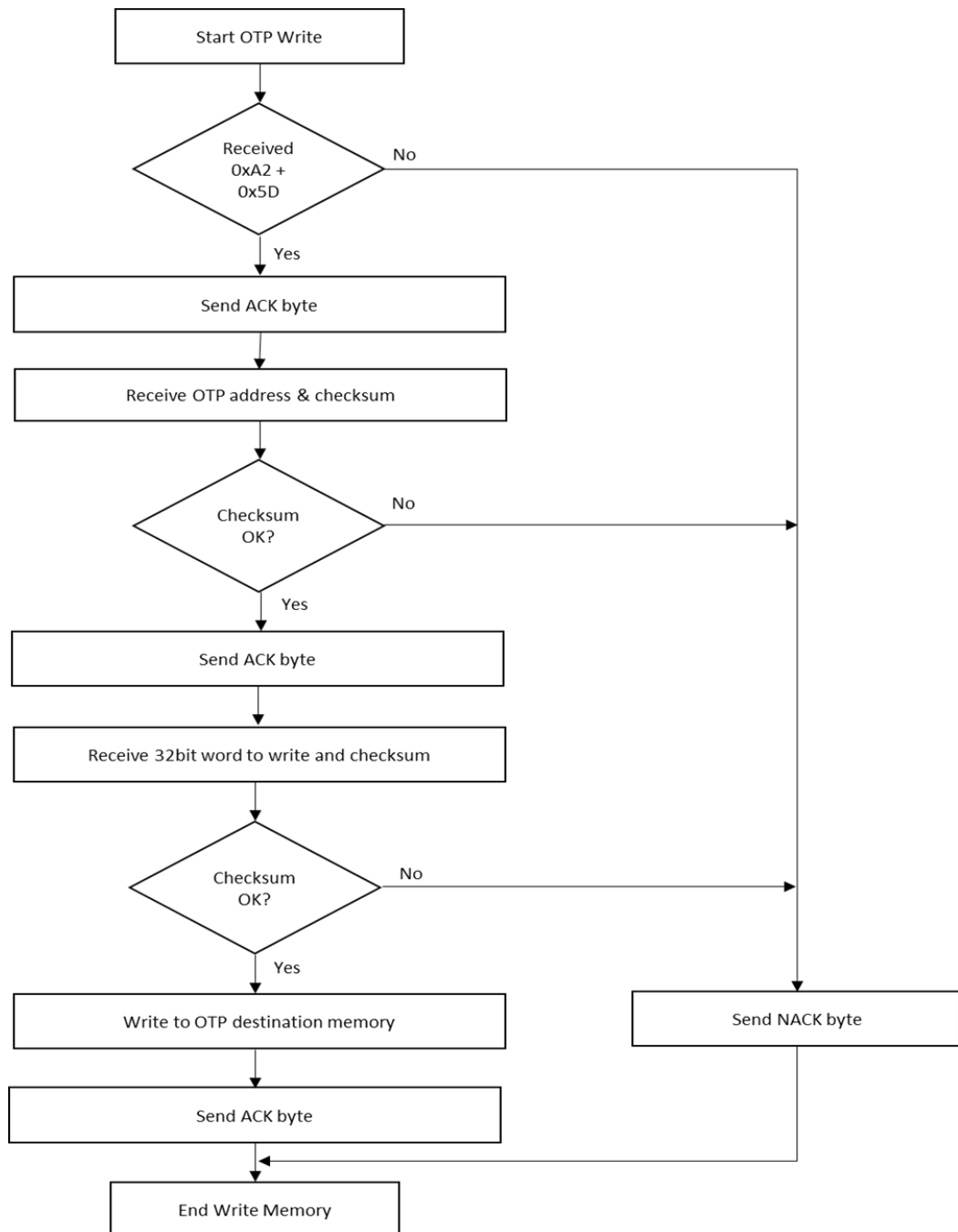


Figure 21. OTP Write command: device side



DT57272V1

The host sends the bytes to the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series as follows:

1. Byte 1: 0xA2
2. Byte 2: 0x5D
3. Wait for acknowledgment.
4. Byte 3 to Byte 6: OTP address
  - Byte 3: MSB
  - Byte 6: LSB
5. Byte 7: Checksum: XOR (Byte 3, Byte 4, Byte 5, Byte 6)

6. Wait for acknowledgment.
7. Byte 8 to Byte 11: Data word to write
8. Checksum byte: XOR data word bytes
9. Wait for acknowledgment.



## 4 Secure bootloader

Secure bootloader feature is part of the UART bootloader preprogrammed on the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series at manufacturing time: it allows a user application to be authenticated in order to understand whether it is an authorized FW or not. Only authorized, trusted applications are allowed to run.

The BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series secure bootloaders use asymmetric cryptographic algorithms with key pair (public, private).

RSA-2048 is used with 256 bytes public key size.

This authentication method allows an Application FW to be authenticated by generating and appending a digital sign to it.

Secure bootloader feature can be activated by writing a specific value on OTP area.

### 4.1 How it works

The following steps describe the process to be followed in order to correctly authenticate an application firmware at user production time:

User generates the key pair (public and private) and programs its device using signed firmware:

- Application firmware is signed by owner using the private key and the generated sign is appended to the application firmware (digital sign size is 2048 bits)
- The BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series OTP sections are used to store the generated public key

Secure bootloader uses the following components to verify/authenticate the application image:

- Stored public key on the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series OTP
- Application firmware image on flash (with information related to the size)
- Digital signature appended at the end of the application firmware image
- OTP dedicated location for enabling secure bootloader

Only the application firmware image signed with the correct private key is executable and the private key is never shared or stored inside the devices.

The key point of this method is that the application firmware signing is done using the private key (not stored on the device) and authentication/verification is done with associated public key. So if the application firmware has not been signed with the correct private key (owned by user), the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series are not able to execute it.

#### 4.1.1 How to store the public key in the OTP memory

The OTP memory is used to store the public key and other basic information needed to activate the secure bootloader feature.

To enable the secure bootloader feature, the OTP memory must contain the following information at specified addresses:

1. **Activation Word@ OTP address 0x10001800**
  - The activation word is: **0xEC1CC10B**
  - This word only is able to activate the secure boot feature in the bootloader code
2. **Start flash location of signed application firmware @ OTP address 0x10001804**
  - This is the interrupt vector table start address of the signed application firmware (i.e. 0x10040000)
3. **R2 public key @ OTP address 0x10001808**
  - R2 public key with length 256 bytes
4. **Modulus public key @ OTP address 0x10001908**
  - Modulus public key with length 256 bytes
5. **Exponent public key @ OTP address 0x10001A08**
  - Exponents public key with length 4 bytes

*Note: To guarantee the correct secure verification, the application must start from the flash bottom at address 0x10040000. The same address must be stored at OTP address 0x10001804. In this way, only genuine signed firmware can be executed from the flash.*

The total size of the information to store inside the OTP area is 524 bytes.

When all the data are stored inside the OTP area, it is **mandatory** to enable the OTP lock memory mechanism by writing a word different from 0xFFFFFFFF at OTP address 0x10001BFC.

#### 4.1.2 How to sign application firmware

The following section provides a basic example of the following steps:

1. Generate public/private keys.
2. Generate a digital sign from an input file.
3. Append the digital sign to the input file.
4. Store public key on the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series OTP.
5. Enable secure bootloader on the BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices.

To enable the secure bootloader feature, the following secure bootloader utilities are released:

- **key\_generation.exe**: this utility generates the public and private keys for the RSA-2048 algorithm, using the OPENSSL commands. The OPENSSL SW must be installed and the path must be defined within the Windows Path environment variable. This utility requires the following parameters:

- `-k` <Destination folder for private and public key generated>
- `-f` <Destination folder to save the public\_key.c and public\_key.py files>

The option “k” creates a folder where the public and private key are saved.

The option “f” creates a folder that contains the public key saved in two formats. The first format is compatible for C project and the second one for a python script.

It is the choice of the user to create a simple C project that uses the “*public\_key.c*” file to store the information in the OTP area. Alternatively, the “*public\_key.py*” file is used from the “*store\_key\_OTP.exe*” utility.

*Note:* Log tab is also available. It shows the outcomes of the selected operations

*Note:* The folder names “key” and “code” are not allowed when defining the destination folder name to be used for key generation and storage.

*Note:* The defined destination folder is automatically generated under the Secure\_bootloader folder within the utility installation path. The destination folder location cannot be modified.

- **create\_signed\_bin.exe**: this utility creates a signed firmware starting from a binary file. This utility takes a not signed raw binary file in input and executes the following operations in the following order:

1. Add the firmware size inside the binary file at offset 0x18.
2. Generate the signature for the application firmware. To create the signature, the utility uses the private key already generated with the key\_generation.exe utility.
3. Add the signature to the end of the binary file and create a new signed file.

This utility requires the following parameters:

- `k` <Folder with private and public key used to create the signature (the same folder created with the option “k” of the key\_generation.exe utility)>
- `f` <Input raw binary image file not signed>
- `o` <Output raw binary image file signed>

The file generated with the option “o” is the application signed that can be stored in the main flash.

*Note:* Log tab is also available. It shows the outcomes of the selected operations.

- **store\_key\_OTP.exe**: this utility stores all the information inside the OTP area. It uses the bootloader engine, so, when this utility runs the device must be in bootloader mode (see [Section 2: UART bootloader activation](#)).

The utility requires the following parameters:

- `f` <Folder with the public\_key.py file to store in OTP (the same folder created with the option “f” of the key\_generation.exe utility)>
- `p` <Device COM Port, used to run the bootloader utility>
- `a` <Start firmware address (i.e. 0x10040000)>
- `l` <Lock of the OTP area>

*Note:* Log tab is also available. It shows the outcomes of the selected operations.

*Note:* The secure bootloader utilities are available on the BlueNRG-LP, BlueNRG-LPS, STSW-BNRGLP-DK, application, secure\_bootloader folder and on the STM32CubeProgrammer PC tool.

The secure boot feature is not reversible and cannot be disabled. When the secure boot feature is activated at reset, the device verifies whether the firmware, in the main flash, is authenticated or not. The public key stored inside the OTP is used. The OTP cannot be changed, so, it is important to save the keys generated with the utility "key\_generation.exe", because if a new firmware update is required, the same keys must be used to generate the signature for the new signed firmware, otherwise the secure bootloader detects the firmware as not authenticated and it is not executed.

#### 4.1.3 How to perform an authenticated OTA firmware upgrade (BlueNRG-LP, BlueNRG-LPS devices, and related STSW-BNRGLP-DK SW package)

In the context of the STSW-BNRGLP-DK SW package, OTA reset and service managers are also able to perform the authentication procedure in a device where the secure bootloader capability has been enabled, following the procedure described in [Section 4.1.2: How to sign application firmware](#).

OTA reset and the lower application and OTA service manager must also be signed/authenticated. The keys used are those generated for signing the user application to be loaded through the OTA firmware upgrade process.

When the secure bootloader feature is activated with specific keys on OTP area, the authenticated OTA firmware upgrade support is possible by:

1. Select the specific OTA firmware upgrade framework (OTA reset or service manager)
  - a. OTA reset manager: authenticate and load the OTA reset manager and the user lower application, following the steps described in [Section 4.1.2: How to sign application firmware](#), and load both on the device flash
  - i. OTA service manager: authenticate and load the OTA service manager following the steps described in [Section 4.1.2: How to sign application firmware](#), and load it on the device flash
2. Authenticate the specific user application to be loaded through the selected OTA firmware upgrade process

The user can then follow the usual OTA firmware upgrade steps described on "The BlueNRG-LP, BlueNRG-LPS, (over-the-air) firmware upgrade (AN5463)". Since the selected device and OTA framework is authenticated (secure bootloader has been activated through OTP), OTA reset or service manager automatically applies the secure bootloader authentication procedure as final criteria to give control to new loaded and valid/authenticated application. If the loaded application is not a valid/authenticated application, no application is executed.

The OTA reset/service managers perform the authentication procured by adding this code before jumping to the loaded application:

```
/* Secure boot activation*/
if (*(volatile uint32_t*)SECURE_BOOT_OTP_ACTIVATION_ADDR == SECURE_BOOT_MAGIC_WORD)
    verifyFW(appAddress);
```

where the "if" instruction simply checks if secure bootloader has been activated on the device (refer to How to store the public key in the OTP memory). The `verifyFW()` function uses the device UART bootloader, secure bootloader functions to authenticate the loaded image through OTA firmware upgrade.

**Note:** *If the `verifyFW()` function is used for authentication purposes on a specific user application, it must reserve two RAM words, respectively, at address 0x200000C4 and 0x200000C8 for the BlueNRG-LP, BlueNRG-LPS devices.*

**Note:** *OTA\_ServiceManager is not supported in the context of the STM32CubeWB0 SW package.*

## Revision history

**Table 2. Document revision history**

Date	Version	Changes
20-Jul-2020	1	Initial release.
29-Sep-2020	2	Update Section Introduction, Section 3.1 Get List command, Section 4.1.2 How to sign an application FW.
03-May-2021	3	Added Section 4.1.3 How to perform an authenticated OTA FW upgrade procedure.
06-Apr-2022	4	Updated Section Introduction, Section 1 UART bootloader configuration, Section 2 UART bootloader activation, Section 3 UART bootloader commands, Section 3.3 Get ID command.  Added the reference of the BlueNRG-LPS to the all document.
30-Aug-2023	4.1	Added the reference of the STM32WB0 series to the whole document.
12-May-2025	5	Updated: <ul style="list-style-type: none"> <li>Document title</li> <li>Section 3: UART bootloader commands</li> </ul> Added Section 3.10: OTP Write command
13-Aug-2025	6	Added references to STM32WB0 series devices.  Updated Figure 21. OTP Write command: device side.

## Contents

<b>1</b>	<b>UART bootloader configuration</b>	<b>2</b>
<b>2</b>	<b>UART bootloader activation</b>	<b>3</b>
<b>3</b>	<b>UART bootloader commands</b>	<b>4</b>
3.1	Get List command	5
3.2	Get Version command	7
3.3	Get ID command	9
3.4	Read Memory command	11
3.5	Go command	14
3.6	Write Memory command	16
3.7	Erase Memory command	18
3.8	Readout Protect command	20
3.9	Readout unprotect command	21
3.10	OTP Write command	22
<b>4</b>	<b>Secure bootloader</b>	<b>25</b>
4.1	How it works	25
4.1.1	How to store the public key in the OTP memory	25
4.1.2	How to sign application firmware	26
4.1.3	How to perform an authenticated OTA firmware upgrade (BlueNRG-LP, BlueNRG-LPS devices, and related STSW-BNRGLP-DK SW package)	27
	<b>Revision history</b>	<b>28</b>

## List of figures

<b>Figure 1.</b>	UART bootloader for BlueNRG-LP, BlueNRG-LPS, and STM32WB0 series devices . . . . .	3
<b>Figure 2.</b>	Get List command: host side . . . . .	5
<b>Figure 3.</b>	Get list command: device side . . . . .	6
<b>Figure 4.</b>	Get Version command: host side. . . . .	7
<b>Figure 5.</b>	Get Version command: device side . . . . .	8
<b>Figure 6.</b>	Get ID command: host side . . . . .	9
<b>Figure 7.</b>	Get ID command: device side. . . . .	10
<b>Figure 8.</b>	Read Memory command: host side . . . . .	12
<b>Figure 9.</b>	Read Memory command: device side . . . . .	13
<b>Figure 10.</b>	Go command: host side . . . . .	14
<b>Figure 11.</b>	Go command: device side . . . . .	15
<b>Figure 12.</b>	Write Memory command: host side . . . . .	16
<b>Figure 13.</b>	Write Memory command: device side . . . . .	17
<b>Figure 14.</b>	Erase Memory command: host side. . . . .	18
<b>Figure 15.</b>	Erase Memory command: device side . . . . .	19
<b>Figure 16.</b>	Readout Protect command: host side . . . . .	20
<b>Figure 17.</b>	Readout Protect command: device side . . . . .	20
<b>Figure 18.</b>	Readout unprotect command: host side . . . . .	21
<b>Figure 19.</b>	Readout unprotect command: device side . . . . .	21
<b>Figure 20.</b>	OTP Write command: host side . . . . .	22
<b>Figure 21.</b>	OTP Write command: device side . . . . .	23

**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved