

## AT-Link Console User Manual

### Introduction

This user manual gives an overview of AT-Link Console. AT-Link Console is a command-line application based on AT-Link. With the help of this software, users can configure ARTERY MCU devices through SWD ports.

List of applicable Artery MCUs:

|             |              |
|-------------|--------------|
| Part number | AT32F series |
|-------------|--------------|

## Contents

|          |                                   |           |
|----------|-----------------------------------|-----------|
| <b>1</b> | <b>Introduction .....</b>         | <b>5</b>  |
| 1.1      | Environmental requirements.....   | 5         |
| 1.2      | Glossary .....                    | 5         |
| <b>2</b> | <b>Installation .....</b>         | <b>6</b>  |
| <b>3</b> | <b>Software operation .....</b>   | <b>7</b>  |
| 3.1      | Operating mode.....               | 7         |
| 3.1.1    | Used in Windows.....              | 7         |
| 3.1.2    | Used in Linux.....                | 8         |
| 3.2      | Command line parameters .....     | 9         |
| 3.3      | AT-Link Console return codes..... | 11        |
| 3.4      | Flow chart.....                   | 13        |
| 3.5      | Write serial number .....         | 14        |
| <b>4</b> | <b>Revision history.....</b>      | <b>15</b> |

## List of tables

|  |    |
|--|----|
| Table 1 List of command line parameters..... | 9  |
| Table 2 List of return codes.....            | 11 |
| Table 3 Document revision history.....       | 15 |

## List of figures

|   |    |
|---|----|
| Figure 1 Command line window .....                    | 7  |
| Figure 2 Batch file processing .....                  | 7  |
| Figure 3 Linux OS .....                               | 8  |
| Figure 4 Shell script file .....                      | 8  |
| Figure 5 Flow chart .....                             | 13 |
| Figure 6 Serial number write configuration file ..... | 14 |

# 1 Introduction

## 1.1 Environmental requirements

### ■ Software resources

#### **Windows OS**

Windows 7 and above.

No driver is required when using AT-Link debugger.

#### **Linux OS**

Linux OS with x86\_64 architecture, such as Ubuntu, Federa, etc.

### ■ Hardware resources

AT-Link debugger

USB communication port.

## 1.2 Glossary

### ■ AT-Link debugger

AT-Link is a debugger, released by ARTERY, for the sake of MCU development.

## 2 Installation

### ■ Hardware installation

Step 1: Connect AT-Link debugger to the USB port of PC.

Step 2: Connect AT-Link debugger to the ICE interface of the target evaluation board.

### ■ Software installation

This software doesn't need to be installed, just run the executable program directly.

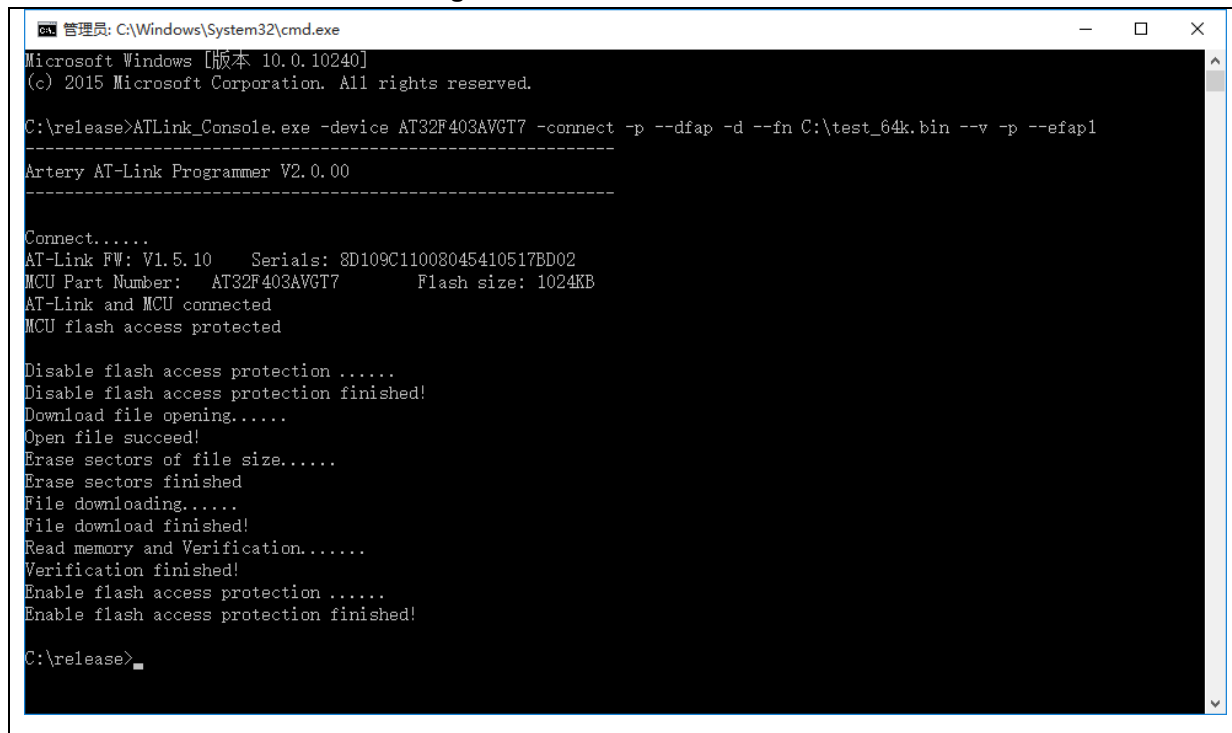
## 3 Software operation

### 3.1 Operating mode

#### 3.1.1 Used in Windows

Mode 1: Input parameters on the command line

Figure 1 Command line window



```
管理员: C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\release>ATLink_Console.exe -device AT32F403AVGT7 -connect -p --dfap -d --fn C:\test_64k.bin --v -p --efap1

Artery AT-Link Programmer V2.0.00

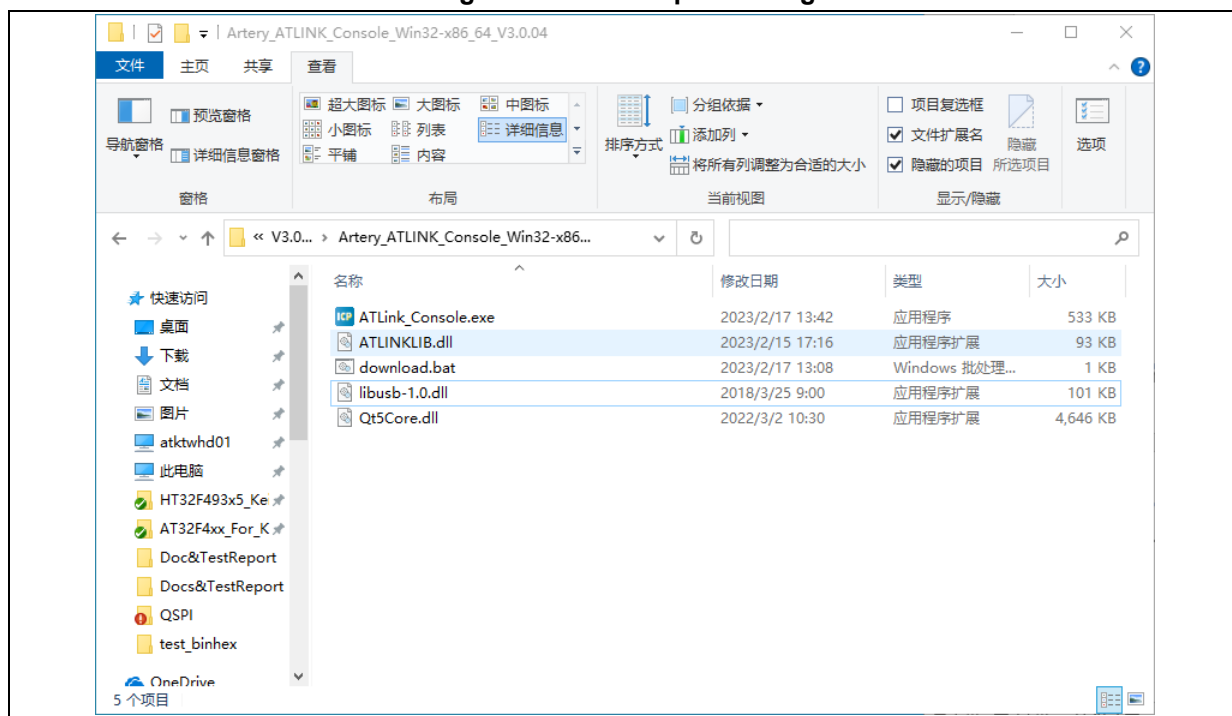
Connect.....
AT-Link FW: V1.5.10   Serials: 8D109C11008045410517BD02
MCU Part Number:   AT32F403AVGT7       Flash size: 1024KB
AT-Link and MCU connected
MCU flash access protected

Disable flash access protection .....
Disable flash access protection finished!
Download file opening.....
Open file succeed!
Erase sectors of file size.....
Erase sectors finished
File downloading.....
File download finished!
Read memory and Verification.....
Verification finished!
Enable flash access protection .....
Enable flash access protection finished!

C:\release>
```

Mode 2: Batch file processing (Refer to DFU\_download.bat for details on common operations)

Figure 2 Batch file processing



### 3.1.2 Used in Linux

1. The executive program "ATLink\_Console" and the script "ATLink\_Console.sh" needs an execution permission.  
Command: `chmod +x ATLink_Console ATLink_Console.sh`
2. Edit the script "download.sh", add operation steps based on command line parameters shown in Section 4.2 (Seedownload.sh in the example) and give an execution permission.  
Command: `chmod +x download.sh`
3. To execute the script download.sh in the terminal, a sudo is required, for either a serial interface or USB device needs a root user authority.  
Command: `sudo ./download.sh`

Figure 3 Linux OS

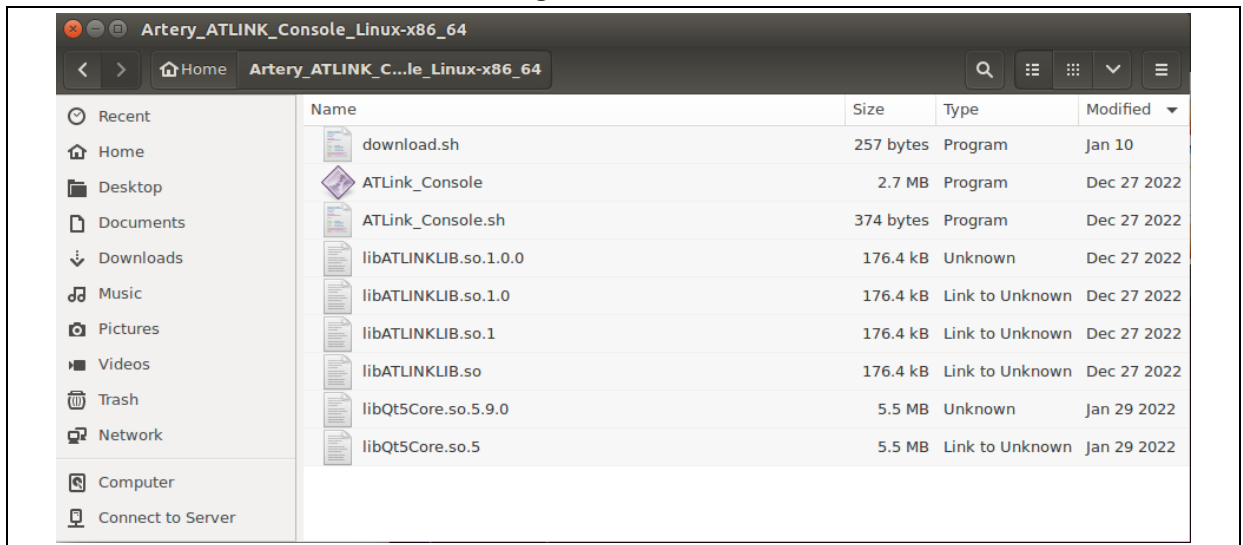


Figure 4 Shell script file





## 3.2 Command line parameters

**Table 1 List of command line parameters**

| Command  | Sub Command      | Remarks  |
|----------|------------------|--|
| -?       |                  | Show the help.   |
| -device  |                  | Set device, e.g. AT32F403ARGT7. You must set device when the device access protection. |
| -connect |                  | Establish connection.  |
| -e       |                  | Erase flash.   |
|          | --all            | Erase all sectors of flash, spim (spim enabled), boot memory(AP mode enabled).         |
|          | --flash          | Erase all sectors of flash. (--all --flash)  |
|          | --spim           | Erase all sectors of SPIM. (--all --spim)  |
|          | --bootm          | Erase all sectors of boot memory. (--all --bootm)                                      |
|          | --sec n-m        | Erase selected sectors, begin sector-end sector, e.g. 0-20.                            |
| -eble    |                  | Erase BLE module flash.  |
|          | --all            | Erase all sectors of BLE module flash.   |
|          | --main           | Erase all sectors of main code space. (--all --main)                                   |
|          | --nvr            | Erase all sectors of NVR space. (--all --nvr)  |
|          | --rdn            | Erase all sectors of RDN space. (--all --rdn)  |
|          | --sec n-m        | Erase selected sectors of main space, begin sector-end sector, e.g. 0-20.              |
| -u       |                  | Upload flash contents to the specified file.   |
|          | --sec n-m        | Upload selected sectors, begin_sector-end_sector, e.g. 0-20.                           |
|          | --fn file_name   | Full path name (bin, hex file; the file type is recognized by its extension).          |
| -uble    |                  | Upload BLE module main code space flash contents to the specified file.                |
|          | --sec n-m        | Upload selected sectors, begin_sector-end_sector, e.g. 0-20.                           |
|          | --fn file_name   | Full path name (bin, hex file; the file type is recognized by its extension).          |
| -d       |                  | Download the content of the specified file into flash                                  |
|          | --a address(hex) | Start address, default 0x08000000; ignored if the target file is not a binary file.    |

| Command | Sub Command          | Remarks  |
|---------|----------------------|--|
|         | --fn file_name       | Full path name (bin/hex file; the file type is recognized by its extension).   |
|         | --v                  | Verify after download.   |
|         | --ne                 | Don't erase sector before downloading file.  |
| -p      |                      | Enable or disable protection.  |
|         | --efap1              | Enable access protection, all options following this one will fail.  |
|         | --efap2              | Enable high level access protection, all options following this one will fail.   |
|         | --y                  | If the MCU is AT32F425/F423/L021/F402/F405/F490/M412/M416, you must enter "--y" for confirmation.<br>(--efap2 --y)                                     |
|         | --dfap               | Disable flash access protection.   |
|         | --depp               | Disable erase and program protection.  |
|         | --eepp n-m           | Enable erase and program protection for sector codes, begin_sector-end_sector, e.g. 0-20.  |
|         | --ebfap              | Enable BLE module access protection.   |
|         | --dbfap              | Disable BLE module access protection.  |
| -usd    |                      | Set user system data to MCU.   |
|         | --get --fn file_name | Get user system data from the device and write it in the specified file, full path name (bin/hex file, the file type is recognized by its extension).  |
|         | --set --fn file_name | Load user system data from the specified file and write it to the device, full path name (bin/hex file, the file type is recognized by its extension). |
| -otp    |                      | Download One-Time Programmable data.   |
|         | --fn file_name       | Full path name (attp file).  |
| -enspim |                      | Enable to access SPIM.   |
|         | --ft type            | SPIM flash type, value 1 or 2. Default value 1.  |
|         | --fs size            | SPIM flash size (MB).  |
|         | --fda FA(hex)        | SPIM FLASH_DA, hexadecimal.  |
|         | --remap 0/1          | Remap IO pin used by SPIM.<br>0: remap0 (Use PA11/PA12 pins)<br>1: remap1 (Use PB10/PB11 pins)   |
| -bmapm  |                      | Set boot memory AP mode.   |
|         | --key value          | Hexadecimal, must be 0xA35F6D24.   |

| Command | Sub Command | Remarks   |
|---------|-------------|---|
| -w4     |             | MCU debug mode, write 32-bit data.                                    |
|         | addr(hex)   | The address of the data to be written, 32-bit address. e.g. 20000000. |
|         | value(hex)  | The value of the data to be written, 32-bit data. e.g. 00112233.      |
| -w2     |             | MCU debug mode, write 16-bit data                                     |
|         | addr(hex)   | The address of the data to be written, 32-bit address. e.g. 20000000. |
|         | value(hex)  | The value of the data to be written, 16-bit data. e.g. 0011.          |
| -mem32  |             | MCU debug mode, read 32-bit data.                                     |
|         | addr(hex)   | The address of the data to be read, 32-bit address. e.g. 08000000.    |
| -mem16  |             | MCU debug mode, read 16-bit data.                                     |
|         | addr(hex)   | The address of the data to be read, 32-bit address. e.g. 08000000.    |
| -r      |             | Reset and run. When MCU access protection, this command is invalid.   |
| -wsn    |             | Write serial number.  |
|         | --ne        | Don't erase sector before writing serial number.                      |

### 3.3 AT-Link Console return codes

In case of error, while executing AT-Link Console commands, the return code (Errrolevel) is greater than 0.

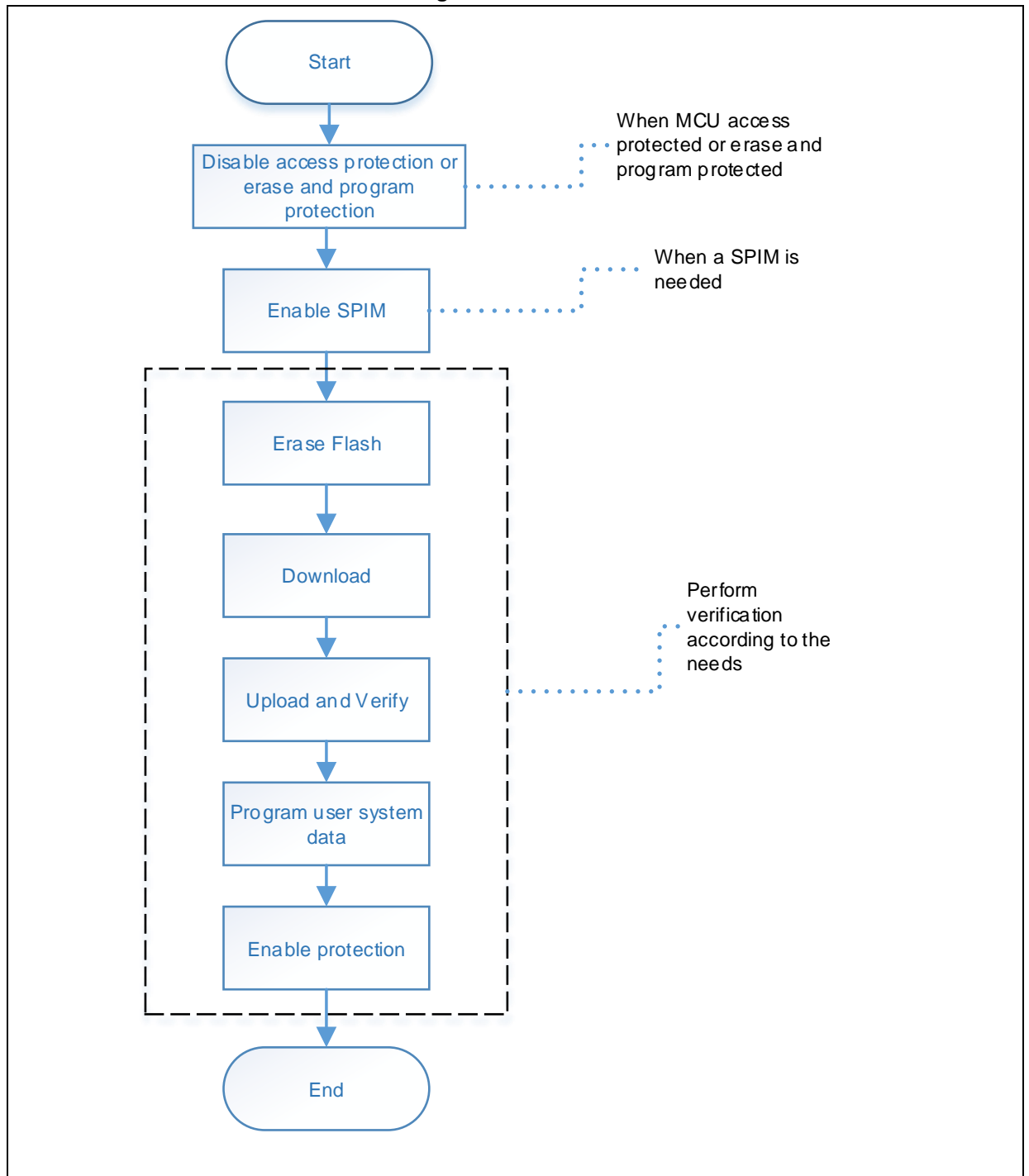
Table 2 List of return codes

| Return code | Command | Error   |
|-------------|---------|---|
| 0x00        | All     | Finished successfully                                       |
| 0x01        | All     | Command arguments error.                                    |
| 0x02        | All     | Connection problem.   |
| 0x03        | -d      | Flash memory programming/verification error.                |
| 0x04        | -u      | Failed to upload Flash memory contents                      |
| 0x05        | -uble   | Failed to upload BLE module main code space Flash contents. |
| 0x06        | -usd    | Failed to get user system data from the device.             |
| 0x07        | -usd    | Failed to write user system data to the device.             |

| Return code | Command            | Error  |
|-------------|--------------------|--|
| 0x08        | -enspim            | Failed to enable SPIM.   |
| 0x09        | -bmapm             | Failed to set boot memory AP mode                                  |
| 0x0A        | -w4/-w2            | Error occurred while writing data to the specified flash address   |
| 0x0B        | -mem32/-mem16      | Error occurred while reading data from the specified flash address |
| 0x0C        | -r                 | Reset and run error.   |
| 0x0D        | -wsn               | Failed to write serial number.                                     |
| 0x20        | -p --efap1         | Failed to enable access protection.                                |
| 0x21        | -p --efap2         | Failed to enable high level access protection.                     |
| 0x22        | -p --dfap          | Failed to disable flash access protection.                         |
| 0x23        | -p --depp          | Failed to disable erase and program protection.                    |
| 0x24        | -p --epp           | Failed to enable erase and program protection.                     |
| 0x25        | -p --ebfap         | Failed to enable BLE module access protection.                     |
| 0x26        | -p --dbfap         | Failed to disable BLE module access protection.                    |
| 0x30        | -e --all --flash   | Failed to erase all sectors of Flash.                              |
| 0x31        | -e --all --spim    | Failed to erase all sectors of SPIM.                               |
| 0x32        | -e --all --bootm   | Failed to erase all sectors of boot memory.                        |
| 0x33        | -e --sec           | Failed to erase selected sectors.                                  |
| 0x34        | -eble --all        | Failed to erase all sectors of BLE module Flash.                   |
| 0x35        | -eble --all --main | Failed to erase all sectors of BLE main code space.                |
| 0x36        | -eble --all --nvr  | Failed to erase all sectors of BLE NVR space.                      |
| 0x37        | -eble --all --rdn  | Failed to erase all sectors of BLE RDN space.                      |
| 0x38        | -eble --sec        | Failed to erase selected sectors of BLE main code space.           |
| 0x40        | -otp               | Failed to download One-Time Programmable data.                     |

### 3.4 Flow chart

Figure 5 Flow chart



### 3.5 Write serial number

Use the “-wsn” command to write the serial number.

To write the serial number, users need to modify three parameters in the *WriteSN.ini* file, including:

- WriteAddr

Serial number write address

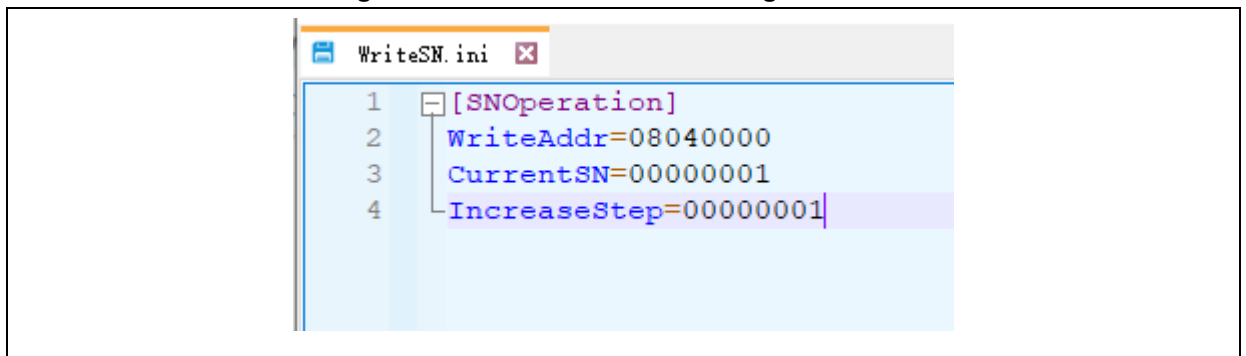
- CurrentSN

Current serial number. Every time the serial number is written successfully, this value will be automatically modified and incremented according to the step.

- IncreaseStep

Serial number increment step.

Figure 6 Serial number write configuration file



## 4 Revision history

**Table 3 Document revision history**

| Date       | Version | Revision note   |
|------------|---------|---|
| 2025/02/18 | V2.10   | 1. Added serial number write feature.   |
| 2024/10/29 | V2.09   | 1. Support for AT32M412/M416 series.<br>2. Added downloading One-Time Programmable data.  |
| 2023/08/10 | V2.08   | 1. Support for AT32F423VCW.<br>2. Support for AT32F402/F405 series.   |
| 2023/07/06 | V2.07   | 1. Support for AT32A403A series.  |
| 2023/02/12 | V2.06   | 1. Supports multiple platforms, including Windows, Linux (Ubuntu, Fedora) OS.<br>2. Added AT32F423 series.  |
| 2022/08/12 | V2.05   | 1. Added return codes.  |
| 2022/07/15 | V2.04   | 1. Added AT32L021 series.   |
| 2022/04/27 | V2.03   | 1. Support reset and run.   |
| 2022/01/26 | V2.02   | 1. Added AT32WB415CCU7-7  |
| 2022/01/04 | V2.01   | 1. Added SPIM support.<br>2. Support disable/enable of the advanced access protection.<br>3. Support disable/enable of the programming protection.<br>4. Support upload feature.<br>5. Support direct read/write feature. |
| 2021/11/26 | V2.00   | 1. Initial release. Support AT32F403/F413/F415/F421/F403A/F407/F435/F437.<br>2. Added AT32F425 series.  |

### IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers understand and agree that purchasers are solely responsible for the selection and use of Artery's products and services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any unjudicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Aerospace applications or environment; (D) Weapons, and/or (E) Other applications that may cause injuries, deaths or property damages. Since ARTERY products are not intended for the above-mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form.

© 2025 ARTERY Technology – All Rights Reserved