AN0042

Application Note

AT32 SPIM Application Note

Introduction

This application note introduces how to use AT32 MCU SPIM to expand the external memory.

Applicable products:

Part number	AT32F403 series AT32F403A series AT32F407 series AT32F413 series AT32A403A series

Contents

List of tables

List of figures

1 Overview

The SPIM (External SPI FLASH memory interface), with the maximum address field of 0x08400000 - 0x093FFFFF (16MB), is a unique Flash access method of AT32 MCUs, which supports user's Flash to be used as an external memory of AT32 MCUs. Different from on-chip Bank1/Bank2, users can enable or disable SPIM as needed. Once enabled, SPIM supports Flash expansion to realize the following functions:

- Store user program at the SPIM address, similar to Bank1/Bank2;
- User program directly accesses the SPIM address, which is used as a memory storing font library and images.

2 SPIM configuration

SPIM is operated by words (32 bits) or half-words (16 bits) only. It should be initialized ad unlocked before being read, programmed and erased.

For different AT32 MCU series and packages, pins used by SPIM are different. For details, please refer to the corresponding reference manual and datasheet. In this application note, the AT32F403A series is used as an example for demonstration.

2.1 Initialization and unlock operation

The initialization and unlock procedures are encapsulated as library functions in AT32 BSP, which can be directly called by users.

1) Enable GPIOA, GPIOB and IOMUX CRM clock;
2) Configure the corresponding PA8, PA11, PA12, PB1, PB6 and PB7 as push-pull output;
3) Enable the SPIM interface in the IOMUX_REMAP2 register;
4) Configure the FLASH_SELECT register and select the SPIM Flash type;
5) Use KEY to unlock SPIM: write values (0x45670123 and 0xCDEF89AB, respectively) to the FLASH_UNLOCK3 register;
6) Check whether the SPIM is unlocked successfully; read the OPLK bit in the FLASH_CTRL3 register: if this bit is cleared, SPIM is ready for operation.

2.2 Flash model selection

SPIM can be configured to support different models of spi Flash, and the supported instruction sets are listed in Table 1. For more details, please refer to the FLASH section in reference manual.

Table 1. Instruction sets supported by SPIM

Instruction	Instruction code	FLASH_SELECT register configuration	Notes
Write Enable	0x06	0x1/0x2	Both models of Flash support 0x06 instruction
Quad Page Program	0x32	0x1/0x2	Both models of Flash support 0x32 instruction
Sector Erase	0x20	0x1/0x2	Both models of Flash support 0x20 instruction
Chip Erase	0xC7	0x1/0x2	Both models of Flash support 0xC7 instruction
Read Status Register	0x05	0x1/0x2	Both models of Flash support 0x05 instruction
Quad I/O Read	0xEB	0x1/0x2	Both models of Flash support 0xEB instruction 24bit Addr + 6 x Dummy cycles
Volatile status Register write enable	0x50	0x1	When these three instructions are used for model 1 Flash selection, the hardware automatically sends an instruction to configure the Quad Enable (QE bit) in the Status Register. Model 1 Flash supports: 0x50 and 0x01, or 0x50 and 0x31
Write Status Register-1	0x01		
Write Status Register-2	0x31		

2.3 Read operation

The address field (0x08400000 – 0x093FFFFF) is directly accessed by words (32 bits) or half-words (16 bits).

2.4 Programming operation

Programming procedures are encapsulated as library function in AT32 BSP, which can be called by users directly.

1) Enable programming and set FPRGM=1 in the FLASH_CTRL3 register;
2) Write values to the address to be programmed by words (32 bits) or half-words (16 bits);
3) Check whether the OBF bit in the FLASH_STS3 register is cleared; if it is cleared, the write operation is completed;
4) Disable programming and set FPRGM=0 in the FLASH_CTRL3 register;
5) Check the PRGMERR and EPPERR bits in the FLASH_STS3 register; if both bits are set to 0, the write operation is completed successfully.

2.5 Erase operation

SPIM erase operations include Mass Erase and Sector Erase (each sector is 4 KB). Erase procedures are encapsulated as library function in AT32 BSP, which can be called by users directly.

Mass Erase

1) Enable mass erase by setting CHPERS=1 in the FLASH_CTRL3 register;
2) Start mass erase by setting ERSTR=1 in the FLASH_CTRL3 register;
3) Disable mass erase by setting CHPERS=0 in the FLASH_CTRL3 register.

Sector Erase

1) Enable sector erase by setting SECERS=1 in the FLASH_CTRL3 register;
2) Select and write the sector address to be erased to the FLASH_ADDR3 register;
3) Start sector erase by setting ERSTR=1 in the FLASH_CTRL3 register;
4) Disable sector erase by setting SECERS=0 in the FLASH_CTRL3 register.

2.6 Encryption

The SPIM circuit is exposed outside the MCU chip. In order to protect data in SPIM Flash being read directly, SPIM is designed with encryption function so that the original data can be encrypted by specific algorithm before being written to the Flash. When AT32 MCU reads data from SPIM, it performs decryption to obtain the original data to ensure data security. The “SPIM scrambled KEY” used by encryption algorithm is the data within 0x1FFFF820-0x1FFFF82F in user system data area.

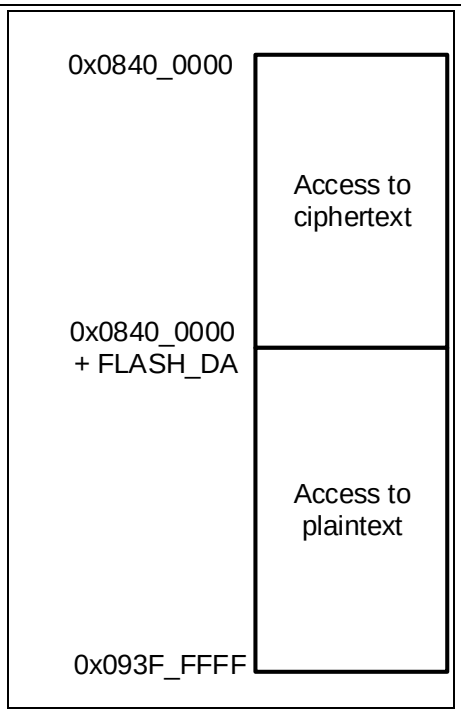
- When “SPIM scrambled KEY” values are all 0xFFFFFFFF, the encryption function is disabled.
- When “SPIM scrambled KEY” values are not all 0xFFFFFFFF, the encryption function is enabled. AT32 MCU delimits the encryption range according to the values in the FLASH_DA register (data within the address field less than 0x08400000+FLASH_DA is ciphertext; other data is plaintext).

Note: The data encryption status in write operation must be the same as that in read operation; otherwise, the data may be read as messy codes and cannot be used or executed properly. That is, if SPIM scrambled KEY and FLASH_DA are configured when writing the data, the SPIM scrambled KEY and FLASH_DA must be configured as the same when reading the data.

Figure 1. SPIM scrambled KEY storage address

0x1FFF_F820	nBANK3KEY1	BANK3KEY1	nBANK3KEY0	BANK3KEY0
0x1FFF_F824	nBANK3KEY3	BANK3KEY3	nBANK3KEY2	BANK3KEY2
0x1FFF_F828	nBANK3KEY5	BANK3KEY5	nBANK3KEY4	BANK3KEY4
0x1FFF_F82C	nBANK3KEY7	BANK3KEY7	nBANK3KEY6	BANK3KEY6

Figure 2. SPIM address range

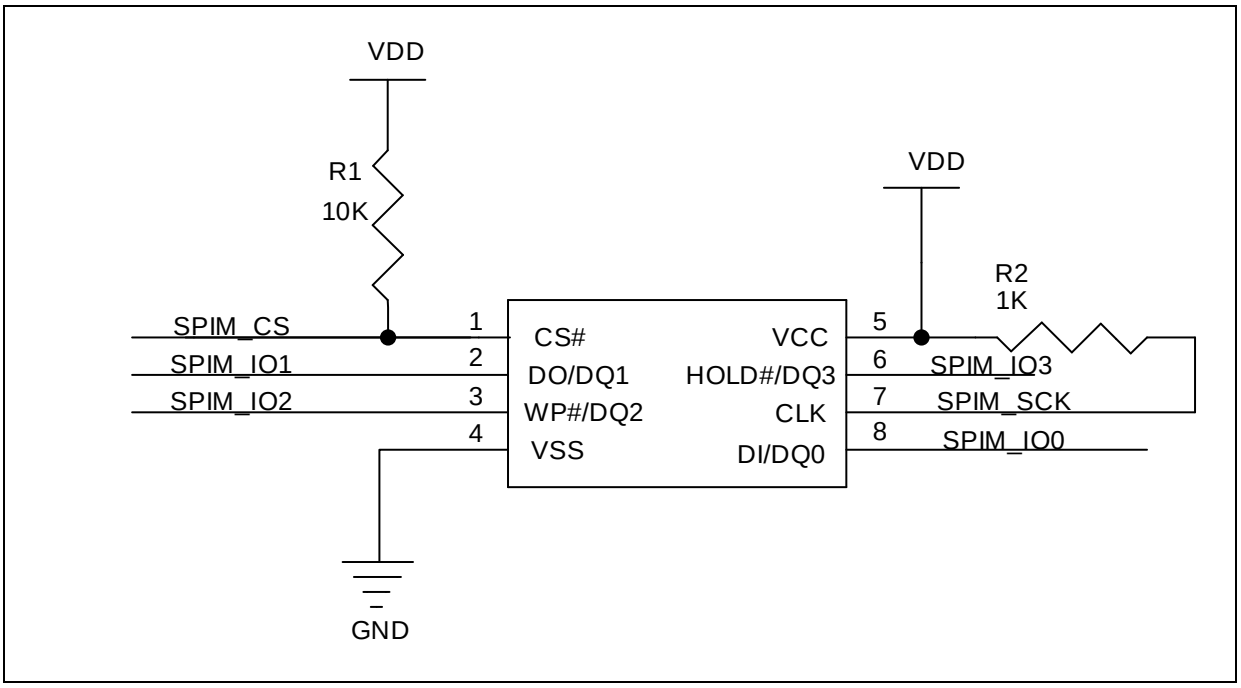


2.7 Hardware circuit

Since the SPIM is connected to an external circuit and is greatly affected by the environment, the PCB wiring length should be minimized to ensure circuit stability.

Note: The frequency is 1/2 of the MCU AHB clock frequency. When the SPIM is enabled, the frequency is limited by the corresponding MCU AHB clock. For details about the maximum frequency with the SPIM enabled, please refer to the general operating conditions in the datasheet of the corresponding MCU.

Figure 3. Reference circuit



2.8 Multiplexed function I/Os (IOMUX)

Pay attention to the multiplexed function I/Os with other IPs when the SPIM is used.

If the IO used by SPIM is also enabled for other IPs, even if this IO is not used in application, it may be occupied.

For example, when XMC and SPIM are used at the same time, PB7 serves as IO2 of SPIM on AT32F403A; if XMC is enabled, even if the XMC_NADV is not enabled, PB7 will be enabled and occupied by XMC_NADV by default, resulting in abnormal operation of SPIM. In this case, configure the IOMUX_REMAP2 register manually and disable the XMC_NADV by calling the following library function:

gpio_pin_remap_config (XMC_NADV_MUX, TRUE)

Figure 4. Shared pins of XMC and SPIM

PB7	I2C1_SDA(7) / XMC_NADV / SPIM_IO2 / TMR4_CH2(7)	USART1_RX / SPI4_SCK / I2S4_CK

Figure 5. XMC_NADV disabled

Bit 10	XMC_NADV_MUX	0x0	rw	XMC_NADV_MUX: XMC NADV connection. This bit is used to choose whether to use XMC_NADV signal. 0: XMC_NADV is connected to pin. (default) 1: XMC_NADV is unused, and the corresponding pin can be used by other peripherals.

3 Demo

In this application note, the operate_spim and run_in_spim in AT32 BSP are used to demonstrate how to use SPIM.

3.1 User program accesses SPIM

Once the SPIM is initialized and unlocked, the user program can execute read, programming and erase operation. The examples\flash\operate_spim in BSP executes SPIM initialization, erase, programming and read operations, and the result is shown on LED.

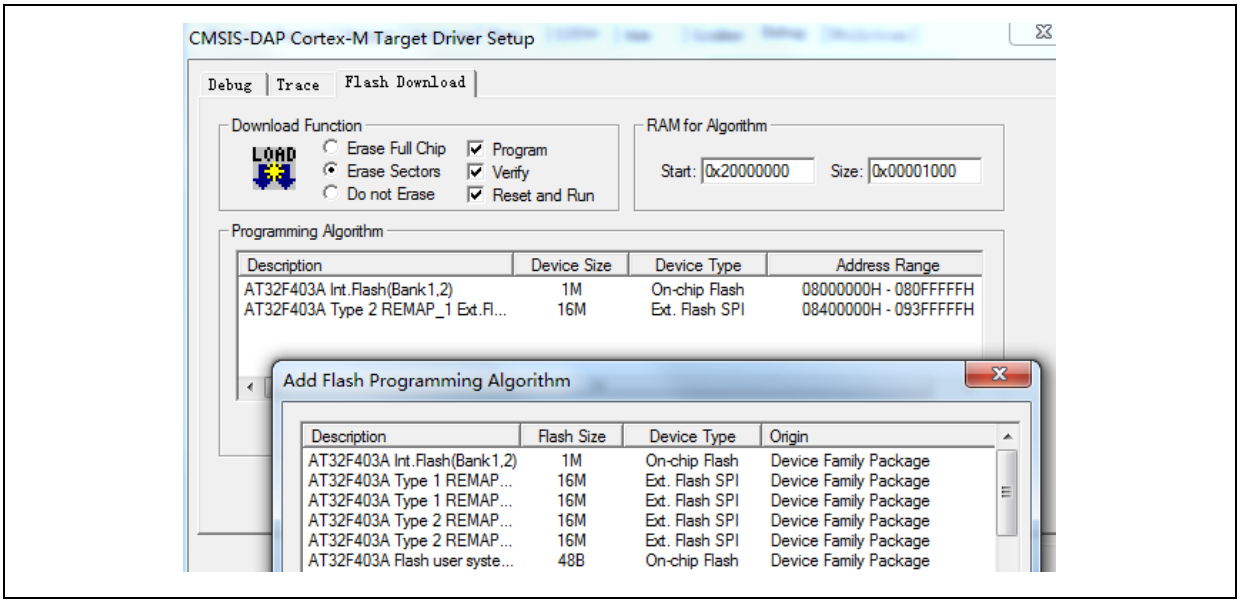
3.2 Download user program to SPIM or execute user program in SPIM

If the user program is downloaded to SPIM through Keil or execute the user program in SPIM, some additional operations are required.

The accessory project “run_in_spim” demonstrates how the code runs on SPIM by LED flashing status.

1) Click Options-Debug-Settings-Flash Download, and select the external Flash model;

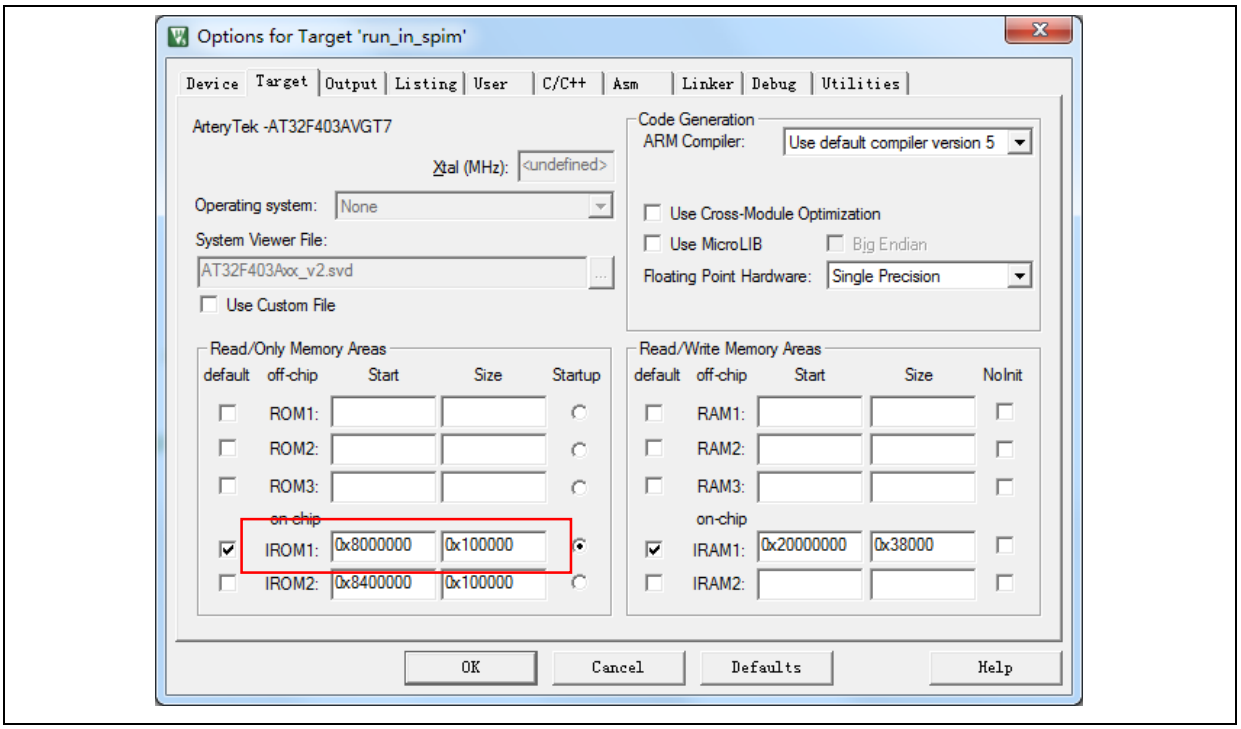
Figure 6. Configure SPIM Flash model



2) Click Options-Target to add the SPIM start address and size (do not tick); in this Demo, the SPIM is defined at IROM2 to store C files.

Note: If the SPIM start address and size are ticked, KEIL may compile unnecessary functions to the SPIM address field.

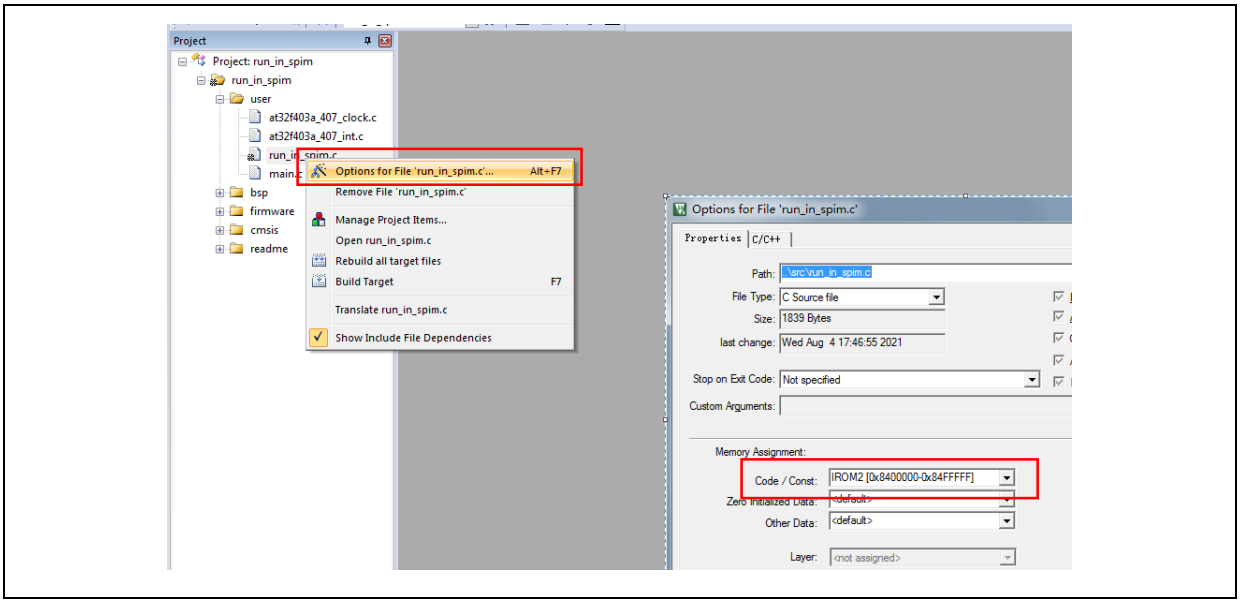
Figure 7. Configure SPIM start address and size



3) Click Project and select C files to run in SPIM; right click Options to modify the code address in Memory Assignment to the corresponding address field.

Note: If the function in project requires compilation with specified address fields, add the address to the corresponding ROM1/2/3 or modify sct. files manually.

Figure 8. Configure codes to run in the first part of SPIM



4) Tick to automatically generate sct. file. After the compilation is complete, browse the sct. file, and it can be found that functions to run in SPIM have been compiled to the corresponding area correctly.

Figure 9. Tick to automatically generate sct. file

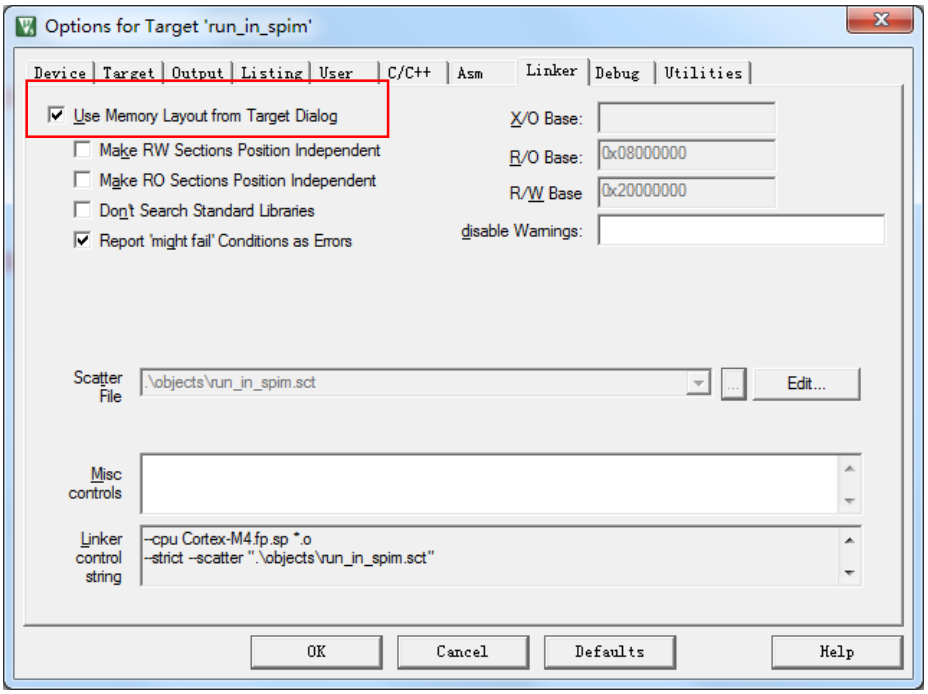


Figure 10. Automatically generated sct. file

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************

LR_IROM1 0x08000000 0x00100000  {    ; load region size_region
  ER_IROM1 0x08000000 0x00100000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x00038000  {  ; RW data
   .ANY (+RW +ZI)
  }
}

LR_IROM2 0x08400000 0x00100000  {
  ER_IROM2 0x08400000 0x00100000  {  ; load address = execution address
    run_in_spim.o (+RO)
  }
}
```

Note: The entire code must start from bank1, and the SPIM flash initialization codes should run before the program executes in SPIM.

4 Revision history

Table 2. Document revision history

Date	Version	Revision note
2022.01.19	2.0.0	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant 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.