GigaDevice Semiconductor Inc.

FLASH emulate EEPROM

Application Note
AN132

Revision 1.0

(May. 2023)

Table of Contents

Table of Contents **2**
List of Figures **3**
List of Tables **4**
1. Introduction **5**
2. Structure of EEPROM backup **6**
2.1. GD32A50x FLASH introduction **6**
2.2. Data structure of EEPROM data **6**
3. FLASH emulate EEPROM solution **8**
3.1. Algorithm implementation **8**
3.1.1. Parameter macro 8
3.1.2. API function 8
3.1.3. Test result 12
4. Revision history **15**

List of Figures

Figure 3-1. Read and write data 12
Figure 3-2. Data in EEPROM backup 14

List of Tables

Table 2-1. Base address and size for 384 KB flash memory 6
Table 2-2. Data structure of flash page 6
Table 3-1. Parameter macro 8
Table 3-2. EEPROM initialization 8
Table 3-3. EEPROM write function 10
Table 3-4. EEPROM read function 12
Table 3-5. Test demo 12
Table 4-1. Revision history 15

1. Introduction

Both FLASH and EEPROM are non-volatile storage devices that can retain data after a power reset. The distinguish between FLASH and EEPROM is that the erase mode is different. EEPROM can be erased by bytes, but the minimum erase unit of FLASH is page. A page usually contains several bytes or even several K bytes. The erase attributes of FLASH and EEPROM determine that EEPROM has a small capacity but a high erasure life, while FLASH has a very large capacity but a short erasure life.

As the MCU frequency is high, FLASH can be used to simulate EEPROM to reduce the cost. In this paper, a method of simulating EEPROM with FLASH is introduced, which realizes the EEPROM data modification by byte, and can prevent the data lost by software reset or power reset. The larger the FLASH storage space, the better the performance of EEPROM.

2. Structure of EEPROM backup

In this paper, 63 pages of DATA FLASH simulate 2K bytes EEPROM are used to introduce the method of FLASH simulation EEPROM.

2.1. GD32A50x FLASH introduction

The GD32A50x consists up to 384KB on-chip FLASH, with bank0 256KB and bank1 128KB. shows the base address and size.

Table 2-1. Base address and size for 384 KB flash memory

Block		Name	Address	size(bytes)
Main Flash Block	bank0	Page 0	0x0800 0000 - 0x0800 03FF	1KB
		Page 1	0x0800 0400 - 0x0800 07FF	1KB
		Page 2	0x0800 0800 - 0x0800 0BFF	1KB
		. . .	. . .	. . .
		Page 255	0x0803 FC00 - 0x0803 FFFF	1KB
	bank1	Page 256	0x0804 0000 - 0x0804 03FF	1KB
		Page 257	0x0804 0400 - 0x0804 07FF	1KB
		Page 258	0x0804 0800 - 0x0804 0BFF	1KB
		. . .	. . .	. . .
		Page 383	0x0805 FC00 - 0x0805 FFFF	1KB
Extend Flash Block		Data Flash	0x0880 0000 - 0x0880 FFFF	64KB
		EEPROM backup	-	

2.2. Data structure of EEPROM data

Table 2-2. Data structure of flash page

Function		size(bytes)
EEPROM page 0	FLASH page use flag	8
	EEPROM page start flag	8
	EEPROM page end flag	8
	EEPROM data	2048
EEPROM page 1	FLASH page use flag	8
	EEPROM page start flag	8
	EEPROM page end flag	8

Function		size(bytes)
	EEPROM data	2048
…	…	…
EEPROM page n	FLASH page use flag	8
	EEPROM page start flag	8
	EEPROM page end flag	8
	EEPROM data	2048

3. FLASH emulate EEPROM solution

3.1. Algorithm implementation

3.1.1. Parameter macro

Table 3-1. Parameter macro

Name	function
EEPROM_DATA_SIZE	Size of emulated EEPROM
EEPROM_FLASH_PAGE_NUM	FLASH pages
FLASH_PAGE_SIZE	FLASH page size
EEPROM_PAGE_SIZE	EEPROM page size
EEPROM_PAGE_DW_NUM	Number of double words in EEPROM page
EEPROM_PAGE_NUM	EEPROM pages in a FLASH page
EEPROM_BACKUP_SIZE	Size of EEPROM backup
EEPROM_BACKUP_END_ADDR	End address of EEPROM backup
EEPROM_BACKUP_START_ADDR	Start address of EEPROM backup
EEPROM_PAGE_HEAD_FLAG	EEPROM start flag
EEPROM_PAGE_END_FLAG	EEPROM end flag
EEPROM_WORK_PAGE_FLAG	FLASH page use flag
FLASH_PAGES_PER_EEPROM_PAGE	FLASH page number per EEPROM page

3.1.2. API function

Function eeprom_init

The eeprom_init function is used to initialize the EEPROM backup area and obtain the relative number of the FLASH page currently being used for EEPROM backup.

Table 3-2. EEPROM initialization

```
void eeprom_init(void)
{
    uint16_t i = 0;
    uint8_t flag_mark_num = 0;
    uint64_t flag_work_page = 0;
    uint8_t check_ff_flag = 0;
    for(i = 0; i < EEPROM_FLASH_PAGE_NUM; i += FLASH_PAGES_PER_EEPROM_PAGE) {
        flag_work_page = REG64(EEPROM_BACKUP_START_ADDR + i * FLASH_PAGE_SIZE);
        check_ff_flag = check_ff(EEPROM_BACKUP_START_ADDR + i * FLASH_PAGE_SIZE,
FLASH_PAGE_SIZE * FLASH_PAGES_PER_EEPROM_PAGE);
        /* if the flash is without EEPROM_WORK_PAGE_FLAG but the page is not empty, erase
```

```
the flash page */
        if(((0xffffffffffffffff == flag_work_page) && (0x01 != check_ff_flag)) || ((0xffffffffffffffff != flag_work_page) && (EEPROM_WORK_PAGE_FLAG != flag_work_page))) {
            eeprom_block_erase(EEPROM_BACKUP_START_ADDR + i * FLASH_PAGE_SIZE);
        } else if(REG64(EEPROM_BACKUP_START_ADDR + i * FLASH_PAGE_SIZE) == EEPROM_WORK_PAGE_FLAG) {
            /* find the flash page with EEPROM_WORK_PAGE_FLAG marked */
            current_page = i;
            flag_mark_num++;
        }
    }
    /* no EEPROM_WORK_PAGE_FLAG is found */
    if(flag_mark_num == 0) {
        current_page = 0;
    }
    if(flag_mark_num > 1) {
        /* the first block is not the marked page */
        if(REG64(EEPROM_BACKUP_START_ADDR) == 0xffffffffffffffff) {
            if(REG64(EEPROM_BACKUP_START_ADDR + current_page * FLASH_PAGE_SIZE + 8 * 2) == EEPROM_PAGE_END_FLAG) {
                /* erase the page whose index is current_page-FLASH_PAGES_PER_EEPROM_PAGE(the forward EEPROM block) */
                eeprom_block_erase(EEPROM_BACKUP_START_ADDR + (current_page - FLASH_PAGES_PER_EEPROM_PAGE)*FLASH_PAGE_SIZE);
            } else {
                /* erase the page whose index is current_page, because EEPROM_PAGE_END_FLAG is not found, and the data is incomplete, discard the data */
                eeprom_block_erase(EEPROM_BACKUP_START_ADDR + current_page * FLASH_PAGE_SIZE);
                current_page -= FLASH_PAGES_PER_EEPROM_PAGE;
            }
        /* the first block is the marked block */
        } else {
            /* the marked block is the first block and the last block */
            if(FLASH_PAGES_PER_EEPROM_PAGE != current_page) {
                if(REG64(EEPROM_BACKUP_START_ADDR + 8 * 2) == EEPROM_PAGE_END_FLAG) {
                    eeprom_block_erase(EEPROM_BACKUP_START_ADDR + current_page * FLASH_PAGE_SIZE);
                    current_page = 0;
                } else {
                    eeprom_block_erase(EEPROM_BACKUP_START_ADDR);
                }
```

```
                /* the marked block is the first block and the second block */
                } else {
                    if(REG64(EEPROM_BACKUP_START_ADDR + current_page *
FLASH_PAGE_SIZE + 8 * 2) == EEPROM_PAGE_END_FLAG) {
                        eeprom_block_erase(EEPROM_BACKUP_START_ADDR);
                    } else {
                        eeprom_block_erase(EEPROM_BACKUP_START_ADDR + current_page *
FLASH_PAGE_SIZE);
                        current_page = 0;
                    }
                }
            }
        }
}
```

Function eeprom_write

The eeprom_write function is used to index the current writable address and write data to the corresponding FLASH address. Note that the parameter ee_addr of this function is an emulated EEPROM address, ranging from 0 to 2047.

Table 3-3. EEPROM write function

```
uint8_t eeprom_write(uint16_t ee_addr, uint8_t *data, uint16_t size)
{
    uint8_t ee_state = 0x01, i = 0;
    uint32_t block_addr = 0, ee_data_addr = 0;
    uint64_t temp_flag = 0;
    uint16_t tmp_size = 0, addr_tmp = 0;
    uint8_t *p_tmp = data;
    if(ee_addr + size > EEPROM_DATA_SIZE) {
        ee_state = 0x00;
        size = EEPROM_DATA_SIZE - ee_addr;
    }
    eeprom_read(0, (uint8_t *)record_buf, EEPROM_DATA_SIZE);
    tmp_size = size;
    addr_tmp = ee_addr;
    /* refresh the data in EEPROM */
    while(tmp_size--) {
        ((uint8_t *)record_buf)[addr_tmp++] = *p_tmp++;
    }
    /* find the block start address to write data */
    if(0xffffffffffffffff != REG64(EEPROM_BACKUP_START_ADDR + current_page *
FLASH_PAGE_SIZE + 8)){
        if((EEPROM_FLASH_PAGE_NUM - FLASH_PAGES_PER_EEPROM_PAGE) ==
```

```
current_page){
                current_page = 0;
            }else{
                current_page = current_page + FLASH_PAGES_PER_EEPROM_PAGE;
            }
        }
        block_addr = EEPROM_BACKUP_START_ADDR + current_page * FLASH_PAGE_SIZE + 8;
        ee_data_addr = block_addr + 8 * 2;
        temp_flag = EEPROM_WORK_PAGE_FLAG;
        if(0 == flash_program(block_addr - 8, &temp_flag, 1)) {
            ee_state = 0x00;
        }
        /* write the EEPROM_PAGE_HEAD_FLAG */
        temp_flag = EEPROM_PAGE_HEAD_FLAG;
        if(0 == flash_program(ee_data_addr - 8 * 2, &temp_flag, 1)) {
            ee_state = 0x00;
        }
        /* write the data */
        if(0 == flash_program(ee_data_addr, record_buf, EEPROM_PAGE_DW_NUM)) {
            ee_state = 0x00;
        }
        /* read back data */
        flash_read_word(ee_data_addr, (uint8_t *)record_buf, EEPROM_DATA_SIZE);
        tmp_size = size;
        addr_tmp = ee_addr;
        while(tmp_size--) {
            /* check the data */
            if(((uint8_t *)record_buf)[addr_tmp++] != *data++) {
                ee_state = 0x00;
            }
        }
        /* write the EEPROM_PAGE_END_FLAG */
        if(ee_state == 0x01) {
            temp_flag = EEPROM_PAGE_END_FLAG;
            if(0 == flash_program(ee_data_addr - 8, &temp_flag, 1)) {
                ee_state = 0x00;
            }
        }
        if(ee_data_addr == block_addr + 8 * 2) {
            if(block_addr == EEPROM_BACKUP_START_ADDR + 8) {
                /* the current page is the last flash page */
                eeprom_block_erase(EEPROM_BACKUP_START_ADDR                                    +
(EEPROM_FLASH_PAGE_NUM - FLASH_PAGES_PER_EEPROM_PAGE)*FLASH_PAGE_SIZE);
```

```
        } else {
            /* the current page is not the last flash page */
            eeprom_block_erase(EEPROM_BACKUP_START_ADDR + (current_page -
FLASH_PAGES_PER_EEPROM_PAGE)*FLASH_PAGE_SIZE);
        }
    }
    return ee_state;
}
```

Function eeprom_read

The eeprom_read function is used to read the latest data in the EEPROM backup area. Note that the entry ee_addr to this function is an EEPROM address, ranging from 0 to 135.

Table 3-4. EEPROM read function

```
uint8_t eeprom_read(uint16_t ee_addr, uint8_t *data, uint16_t size)
{
    uint8_t ee_state = 1, i = 0;
    uint32_t page_addr, ee_data_addr;
    /* find the page start address to write data */
    page_addr = EEPROM_BACKUP_START_ADDR + current_page * FLASH_PAGE_SIZE + 8;
    /* locate at the address to read data */
    ee_data_addr = page_addr + 8 * 2;
    if(ee_addr + size > EEPROM_DATA_SIZE) {
        ee_state = 0x00;
        size = EEPROM_DATA_SIZE - ee_addr;
    }
    /* read data */
    flash_read_word(ee_data_addr + ee_addr, data, size);
    return(ee_state);
}
```

3.1.3. Test result

Overwrites the first byte in EEPROM 16 times. The data to write is shown in ***Figure 3-1. Read and write data***.

Figure 3-1. Read and write data

```
uint8_t data[2048] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
                      0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F
                      };
```

The code is shown in ***Table 3-5. Test demo***.

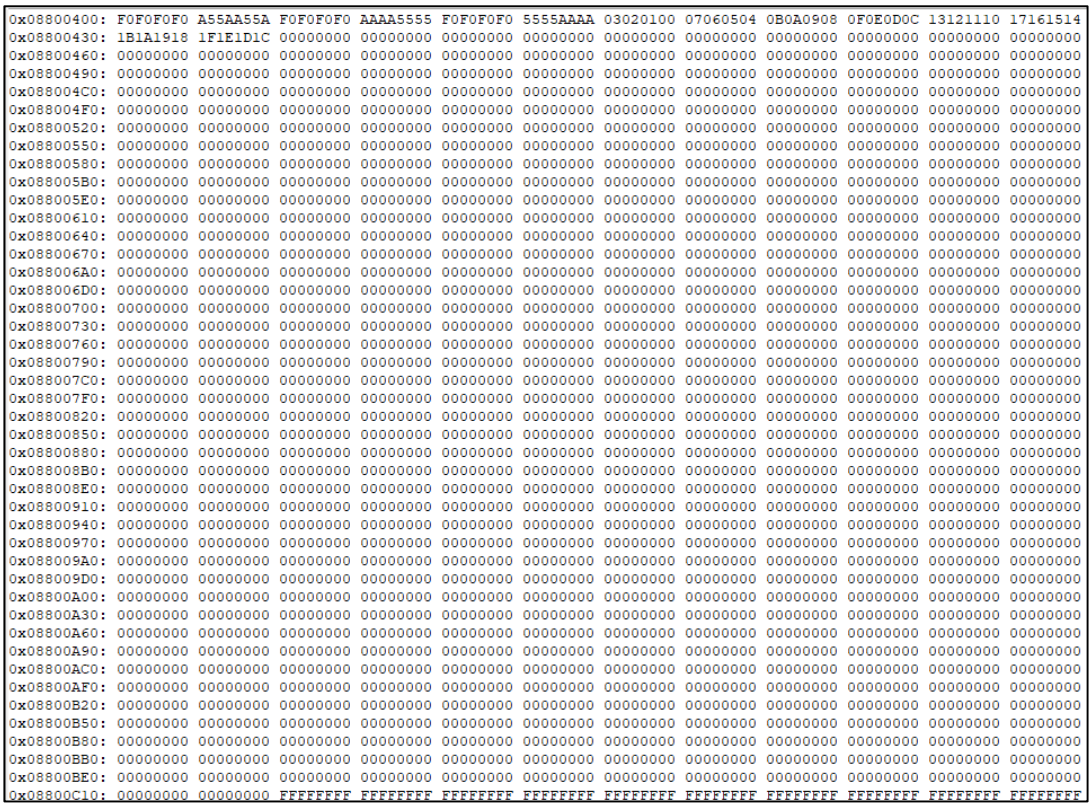
Table 3-5. Test demo

```
int main(void)
{
```

```
    fmc_unlock();
    ob_unlock();
    /* configure the data flash size as 64k */
    if(0x10000 != dflash_size_get()){
        ob1_eeprom_parameter_config(OB1CS_DF_64K_EF_0K, OB1CS_EPSIZE_NONE);
        ob_reset();
    }
    ob_lock();
    fmc_lock();
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    eeprom_init();
    eeprom_read(0, data_read, 2048);
    for(int i=0; i<16; i++){
        data[0] = i;
        eeprom_write(0, data, 2048);
        eeprom_read(0, data_read, 2048);
        if(SUCCESS != byte_memory_compare(data, data_read, 2048)) {
            gd_eval_led_on(LED2);
            return ERROR;
        }
    }
    gd_eval_led_on(LED1);
    while(1) {
    }
}
```

The test result is shown in ***Figure 3-2. Data in EEPROM backup***.

Figure 3-2. Data in EEPROM backup



```
0x08800400: F0F0F0F0 A55AA55A F0F0F0F0 AAAA5555 F0F0F0F0 5555AAAA 03020100 07060504 0B0A0908 0F0E0D0C 13121110 17161514
0x08800430: 1B1A1918 1F1E1D1C 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800460: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800490: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088004C0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088004F0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800520: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800550: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800580: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088005B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088005E0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800610: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800640: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800670: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088006A0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088006D0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800700: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800730: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800760: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800790: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088007C0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088007F0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800820: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800850: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800880: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088008B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088008E0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800910: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800940: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800970: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088009A0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x088009D0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800A00: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800A30: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800A60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800A90: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800AC0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800AF0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800B20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800B50: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800B80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800BB0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800BE0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08800C10: 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
```

4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	May.30, 2023

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.