GigaDevice Semiconductor Inc.

GD32L23x 系列 FLASH 模拟 EEPROM

应用笔记

AN201

1.0 版本

（2024 年 4 月）

目录

图索引

表索引

1. 简介

FLASH 和 EEPROM 都是非易失性储存器，在掉电后数据还会保留。FLASH 和 EEPROM 最大的区别是擦除方式不同，EEPROM 可以按字节进行擦除，但 FLASH 最小擦除单位是页，一页通常包含多个字节甚至几 K 字节。FLASH 和 EEPROM 擦写属性决定了 EEPROM 容量小但擦写寿命比较高，而 FLASH 容量非常大，但擦写寿命短。

在 MCU 主频高的情况下，可以采用 FLASH 模拟 EEPROM 来降低成本。本文介绍了一种采用 FLASH 模拟 EEPROM 的方法，实现了 EEPROM 按字节进行数据修改，可防止复位或掉电产生的数据丢失，当采用的 FLASH 存储空间越大，EEPROM 性能越好。

2. EEPROM 备份区结构

本文以GD32L235RBT6的后63页FLASH模拟2K字节EEPROM来介绍FLASH模拟EEPROM的方法。

2.1. GD32L23x FLASH 简介

GD32L23x提供高达256KB片上FLASH。本文使用的GD32L235RBT6闪存基地址和大小如*表 2-1. GD32L235xx 128KB闪存基地址和大小*所示。

表 2-1. GD32L235xx 128KB 闪存基地址和大小

闪存块	名称	地址范围	大小（字节）
主存储闪存块	第0页	0x0800 0000 – 0x0800 03FF	1KB
	第1页	0x0800 0400 – 0x0800 07FF	1KB
	第2页	0x0800 0800 – 0x0800 0BFF	1KB
	.		
	第127页	0x0801 FC00 – 0x0801 FFFF	1KB
信息块	引导装载程序	0x1FFF D000- 0x1FFF F7FF	10KB
选项字节块	选项字节	0x1FFF F800 – 0x1FFF F80F	16B
一次性编程块	OTP字节	0x1FFF_7000~0x1FFF_71FF	512B

2.2. EEPROM 数据备份结构

表 2-2. EEPROM 页数据备份存储结构

功能		大小（字节）
EEPROM 页 0	FLASH 页使用标记	8
	EEPROM 页开始标记	8
	EEPROM 页结束标记	8
	EEPROM 数据	2048
EEPROM 页 1	FLASH 页使用标记	8
	EEPROM 页开始标记	8
	EEPROM 页结束标记	8
	EEPROM 数据	2048
…	…	…
EEPROM 页 n	FLASH 页使用标记	8
	EEPROM 页开始标记	8
	EEPROM 页结束标记	8

功能		大小（字节）
	EEPROM 数据	2048

3. FLASH 模拟 EEPROM 方案

3.1. 算法实现

3.1.1. 参数宏

表 3-1. 参数宏

名称	功能
EEPROM_DATA_SIZE	模拟 EEPROM 大小
EEPROM_FLASH_PAGE_NUM	使用的 FLASH 页数
FLASH_PAGE_SIZE	FLASH 页大小
EEPROM_PAGE_SIZE	EEPROM 页大小
EEPROM_PAGE_DW_NUM	每页 EEPROM 包含的双字数
EEPROM_PAGE_NUM	一页 FLASH 中包含的 EEPROM 页数
EEPROM_BACKUP_SIZE	EEPROM 备份区大小
EEPROM_BACKUP_END_ADDR	EEPROM 备份区结束地址
EEPROM_BACKUP_START_ADDR	EEPROM 备份区起始地址
EEPROM_PAGE_HEAD_FLAG	EEPROM 起始标记
EEPROM_PAGE_END_FLAG	EEPROM 结束标记
EEPROM_WORK_PAGE_FLAG	FLASH 页使用标记
FLASH_PAGES_PER_EEPROM_PAGE	每页 EEPROM 包含的 FLASH 页数

3.1.2. API 函数

函数 eeprom_init

函数 eeprom_init 用于初始化 EEPROM 备份区，并获取当前正在用于 EEPROM 备份的 FLASH 页相对编号。

表 3-2. EEPROM 初始化

```
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;
                    }
                }
            }
        }
}
```

函数 eeprom_write

函数 eeprom_write 用于索引当前可写的地址，并将数据写到对应 FLASH 地址。注意该函数的入参 ee_addr 是模拟 EEPROM 地址，范围为 0-135。

表 3-3. EEPROM 写函数

```
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;
}
```

函数 eeprom_read

函数 eeprom_read 用于读取 EEPROM 备份区最新的数据。注意该函数的入参 ee_addr 是 EEPROM 地址，范围为 0-135。

表 3-4. EEPROM 读函数

```
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. 测试结果

测试在 EEPROM 中改写第一个字节共 16 次。写入数据如*图 3-1. 读写数据*所示。

图 3-1. 读写数据

```
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
                      };
```

代码如*表 3-5. 测试 demo* 所示。

表 3-5. 测试 demo

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

测试结果如*图 3-2. EEPROM 备份区数据*所示。

图 3-2. EEPROM 备份区数据

```
0x08010400: F0F0F0F0 A55AA55A F0F0F0F0 AAAA5555 F0F0F0F0 5555AAAA 03020100 07060504 0B0A0908 0F0E0D0C 13121110 17161514 1B1A1918
0x08010434: 1F1E1D1C 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010468: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x0801049C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080104D0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010504: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010538: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x0801056C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080105A0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080105D4: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010608: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x0801063C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010670: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080106A4: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080106D8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x0801070C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010740: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010774: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080107A8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080107DC: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010810: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010844: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010878: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080108AC: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080108E0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010914: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010948: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x0801097C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080109B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080109E4: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010A18: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010A4C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010A80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010AB4: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010AE8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010B1C: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010B50: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010B84: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010BB8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08010BEC: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 FFFFFFFF FFFFFFFF
0x08010C20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
```

4. 版本历史

表 4-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2024 年 04 月 10 日

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.