GigaDevice Semiconductor Inc.

FLASH emulate EEPROM for GD32C2x1 series

Application Note
AN213

Revision 1.1

(Jun. 2025)

Table of Contents

List of Figures

List of Tables

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 byte, 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 application note, 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 application note, GD32C2x1 33 pages of FLASH simulate 2K bytes EEPROM are used to introduce the method of FLASH simulation EEPROM.

2.1. GD32C2x1 FLASH introduction

The GD32C2x1 consists up to 64KB on-chip FLASH. The flash base address and size of GD32C2x1 used in this application note are shown as ***Table 2-1. Base address and size for GD32C2x1 64 KB flash memory***.

Table 2-1. Base address and size for GD32C2x1 64 KB flash memory

Block	Name	Address range	size(bytes)
Main flash block	Page 0	0x0800 0000 – 0x0800 03FF	1KB
	Page 1	0x0800 0400 – 0x0800 07FF	1KB
	Page 2	0x0800 0800 – 0x0800 0BFF	1KB
	Page 63	0x0800 FC00 - 0x0800 FFFF	1KB
Information block	Boot loader area	0x1FFF 0000 - 0x1FFF 0BFF	3KB
Option bytes block	Option bytes	0x1FFF 7800 - 0x1FFF 787F	128B
One-time program block	OTP bytes	0x1FFF 7000~0x1FFF 73FF	1KB

2.2. Data structure of EEPROM backup data

Table 2-2. Data structure of EEPROM page backup

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
	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_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 2047.

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)
{
    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

```
0x08007C00: F0F0F0F0 A55AA55A F0F0F0F0 AAAA5555 F0F0F0F0 5555AAAA 03020100 07060504 0B0A0908 0F0E0D0C 13121110 17161514 1B1A1918 1F1E1D1C
0x08007C38: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007C70: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007CA8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007CE0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007D18: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007D50: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007D88: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007DC0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007DF8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007E30: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007E68: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007EA0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007ED8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007F10: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007F48: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007F80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007FB8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08007FF0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008028: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008060: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008098: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080080D0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008108: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008140: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008178: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080081B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080081E8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008220: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008258: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008290: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080082C8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008300: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008338: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008370: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080083A8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x080083E0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x08008418: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 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	Jun.03, 2024
1.1	Updated the Important Noctice	Jun.03, 2025

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 according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. 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.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its 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 Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.