GigaDevice Semiconductor Inc.

Arm® Cortex®- M3/M4/M23/M33 32-bit MCU

应用笔记

AN043

目录

图索引

表索引

1. 介绍

Flash作为一种非易失性存储器，在微控制器系统中起着不可或缺的作用，其性能的好坏将影响整个系统的运行效率。其性能主要反映在对flash的操作时间上，包括全片擦除、页擦除、字编程时间。

该应用笔记提供了两种对flash操作时间测量的方法。

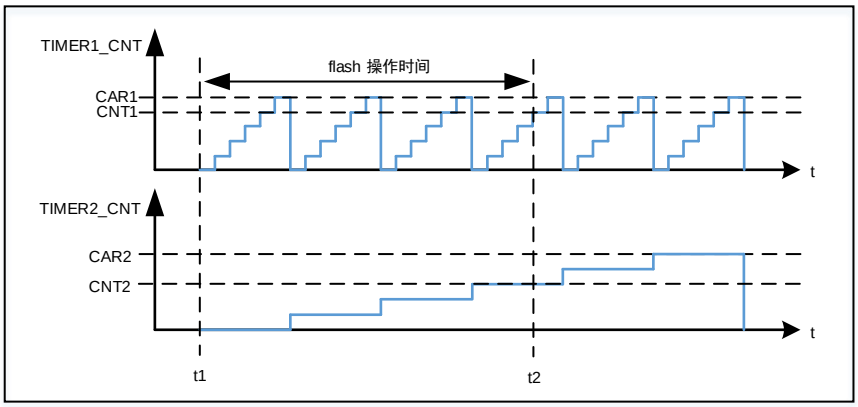
2. Flash 操作时间测量方法

2.1. 定时器计数法

定时器计数法采用 MCU 内部定时器计数，并利用计数值计算对 flash 的操作时间。该方法通过在进行 flash 操作之前清除定时器计数值并启动计数，在 flash 操作完成之后读取计数器值并关闭计数。为了提高测量精度，我们将系统运行在最高主频 64MHz，并将定时器时钟分频成 8MHz，即定时器计数周期为 0.125us。由于 L23x 定时器为 16 位定时器，则单个定时器最大只能测量 65536*0.125us = 8.192ms，为了提高测量时间范围，可采用定时器级联的方法。将其中一个定时器更新事件脉冲作为另一个定时器的时钟源，这样可以在保证最大精度的同时，将测量时间扩大为 65536*8.192ms=536.870912s。如*图 2-1. 定时器测量 flash 操作时间的方法*所示，flash 操作时间 tf：

$$tf = t2\text{-}t1 = (CNT1 + (CAR1 * CNT2)) * 0.125us \quad (2\text{-}1)$$

图 2-1. 定时器测量 flash 操作时间的方法

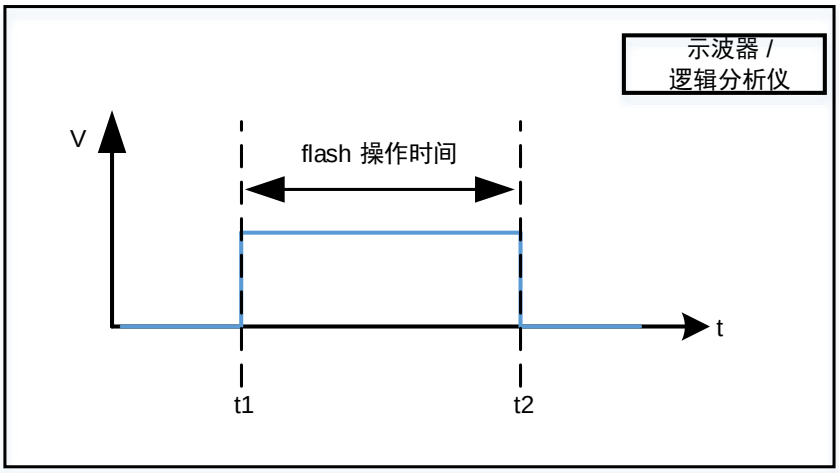


2.2. I/O 电平翻转法

采用 MCU 外部引脚输出高低电平，并使用示波器/逻辑分析仪测量该脉冲时间。

该方法通过在进行 flash 操作之前，将普通 IO 口设置为高电平；在 flash 操作完成之后，将普通 IO 口设置为低电平。通过示波器/逻辑分析仪测量测量正脉冲时间。如*图 2-2. IO 口测量 flash 操作时间的方法*所示，flash 操作时间为 tf = t2-t1。

图 2-2. IO 口测量 flash 操作时间的方法

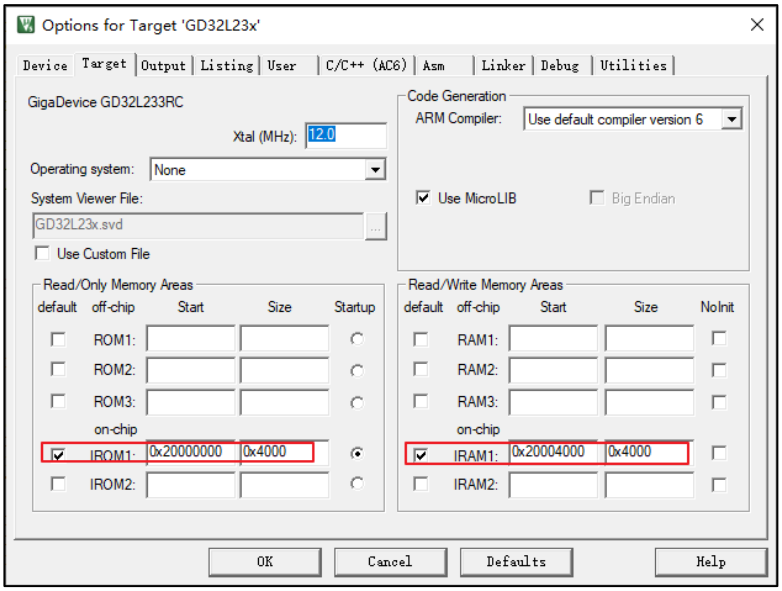

示波器 /
逻辑分析仪
V
flash 操作时间
t
t1
t2

3. Flash 操作时间测量具体实现

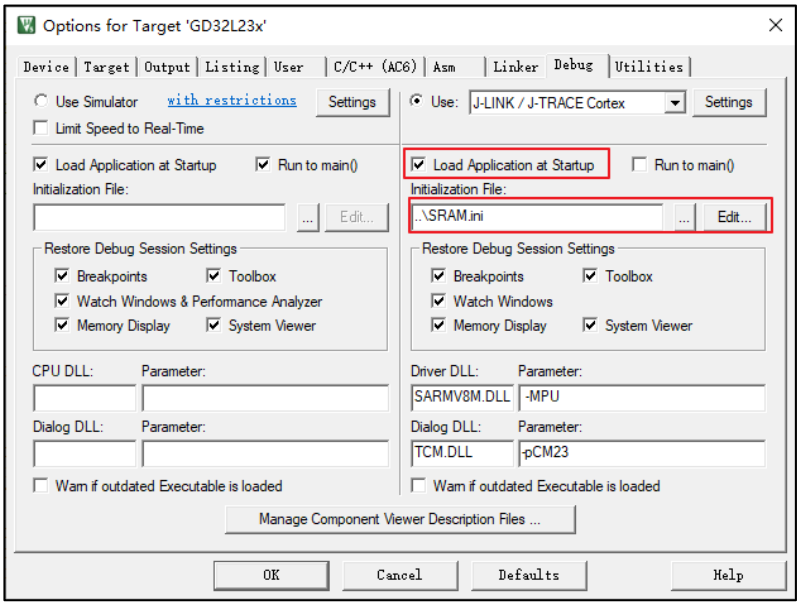
3.1. SRAM 中启动配置

为测量 flash 特性，需要把测量程序放在 sram 中运行。在 sram 中启动调试配置步骤如下：

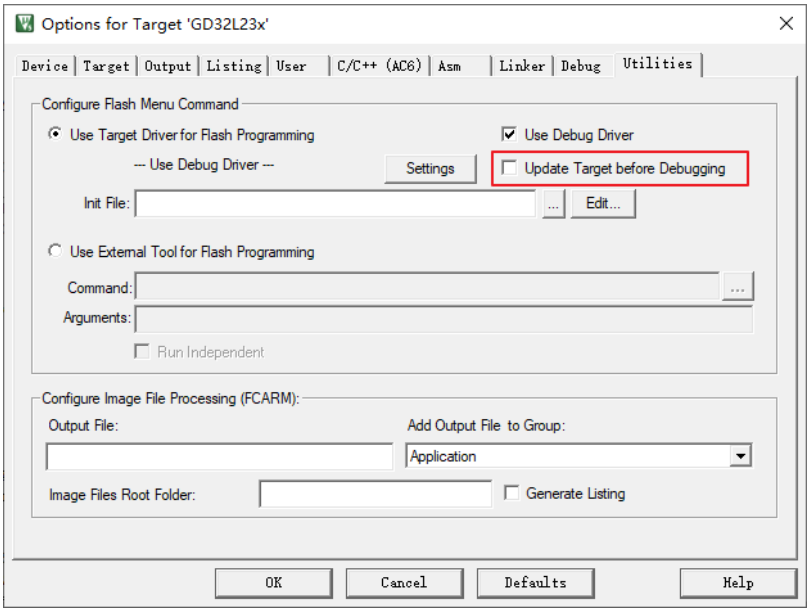
1. 根据实际 sram 空间，配置分散加载区域



2. 添加从 SRAM 中启动的初始化文件



3. 配置 Utilities 选项



4. SRAM.ini 初始化文件

```
FUNC void Setup (void) {
/* Setup Stack Pointer */
SP = _RDWORD(0x20000000);
/* Setup Program Counter */
PC = _RDWORD(0x20000004);
/* Setup Vector Table Offset Register */
_WDWORD(0xE000ED08, 0x20000000);
}
/* Download, Project.axf, the same with your project name */
LOAD Project.axf INCREMENTAL
/* Setup for Running */
```

3.2. 软件实现

通过宏的方式，可以使用定时器方法和 I/O 电平翻转方法分别对 flash 的全片擦除、页擦除、字编程时间进行测量。具体代码实现如下：

1. TIMER 配置

```
void timer_config(void)
{
    timer_parameter_struct timer_initpara;

    rcu_periph_clock_enable(RCU_TIMER);
    rcu_periph_clock_enable(RCU_TIMER2);

    timer_deinit(TIMER_USE);
    /* TIMER1 configuration */
```

```
    timer_initpara.prescaler         = TIMER_PRESCALER;
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            =65535;
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
    timer_init(TIMER_USE, &timer_initpara);

    /* auto-reload preload enable */
    timer_auto_reload_shadow_enable(TIMER_USE);
    /* configure TIMER1 master slave mode */
    timer_master_slave_mode_config(TIMER_USE, TIMER_MASTER_SLAVE_MODE_ENABLE);
    timer_master_output_trigger_source_select(TIMER_USE, TIMER_TRI_OUT_SRC_UPDATE);

    timer_deinit(TIMER2);
    /* TIMER2 configuration */
    timer_initpara.prescaler         = 0;
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            = 65535;
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
    timer_init(TIMER2, &timer_initpara);

    timer_auto_reload_shadow_enable(TIMER2);
    /* slave mode selection: TIMER2 */
    timer_slave_mode_select(TIMER2, TIMER_SLAVE_MODE_EXTERNAL0);
    timer_input_trigger_source_select(TIMER2, TIMER_SMCFG_TRGSEL_ITI0);
}
```

2. 主程序代码

```
/* macro definition */
#define GD32L233RC
#define TEST_MASS_ERASE              1
#define TESE_PAGE_ERASE              1
#define TEST_WORD_PROGRAMME          1
#define TIMER_CNT_MESURE_METHOD      1

#define TIMER_PRESCALER              (8 - 1)
#define TIMER_USE                    TIMER1
#define RCU_TIMER                    RCU_TIMER1
#define PROGRAMME_DATA               0xaa55aa55
#define ADDRESS_TO_PROGRAMME         0x08000000
#define PAGE_TO_ERASE1               0x08000000

#define MEASURE_NUMS                 1
```

```
#define AVERAGE_VALUE_POSITION    (MEASURE_NUMS)

#if defined(GD32L233RC)
#define PAGE_SIZE1_WORD                 ((4*1024)/4)
#define FLASH_SIZE_WORD                 ((256*1024) /4)
#endif
#define    USART_COM          USART1
/* flash operation time struct definition */
typedef struct {
    uint32_t word_programme[MEASURE_NUMS+1];
    uint32_t page_erase[MEASURE_NUMS+1];
    uint32_t mass_erase[MEASURE_NUMS+1];
}flash_operation_time_struct;

flash_operation_time_struct    flash_operation_time;

int main(void)
{
    uint16_t measure_counts = 0;
    /* gpio, timer, usart configuration */
    rcu_periph_clock_enable(RCU_GPIOC);
    gpio_mode_set(GPIOC, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,GPIO_PIN_0);
    gpio_output_options_set(GPIOC, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_0);
    gpio_bit_reset(GPIOC, GPIO_PIN_0);
    gd_eval_com_init(USART_COM);
    timer_config();

    fmc_unlock();
    /* mass erase measure */
#if TEST_MASS_ERASE
    do{
      {
          uint32_t i =0;
          /* mass erase, then programme full flash with PROGRAMME_DATA */
          fmc_mass_erase();
          for(i = 0; i < FLASH_SIZE_WORD; i++){
              fmc_word_program((ADDRESS_TO_PROGRAME + (i * 4)),
PROGRAME_DATA);
          }
      }
#if TIMER_CNT_MESURE_METHOD
        /* clear timer count and enable timer */
        timer_disable(TIMER_USE);
```

```
            timer_disable(TIMER2);
            TIMER_CNT(TIMER_USE) = 0;
            TIMER_CNT(TIMER2) = 0;
            timer_enable(TIMER_USE);
            timer_enable(TIMER2);
#else
            /* set gpio pin to high level */
            gpio_bit_set(GPIOC, GPIO_PIN_0);
#endif
            /* start mass erase */
            fmc_mass_erase();
#if TIMER_CNT_MESURE_METHOD
            /* get the mass erase time */
            flash_operation_time.mass_erase[measure_counts] = TIMER_CNT(TIMER_USE) +
65536*TIMER_CNT(TIMER2);
#else
            /* set gpio pin to low level */
            gpio_bit_reset(GPIOC, GPIO_PIN_0);
#endif
      }while(++measure_counts < MEASURE_NUMS);
#if TIMER_CNT_MESURE_METHOD
      /* get the average mass erase time */
      {
            uint32_t temp =0;
            for(measure_counts =0; measure_counts < MEASURE_NUMS; measure_counts++)
            {
                  temp += flash_operation_time.mass_erase[measure_counts];
            }
            flash_operation_time.mass_erase[AVERAGE_VALUE_POSITION] = temp /
MEASURE_NUMS;
      }
#endif
#endif
      /* page erase measure */
#if TESE_PAGE_ERASE
measure_counts = 0;
do{
          {
                uint32_t i =0;
                /* page erase, then programme this page with PROGRAMME_DATA */
                fmc_page_erase(PAGE_TO_ERASE1);
                for(i = 0; i < 512; i++){
                      fmc_word_program((ADDRESS_TO_PROGRAME + (i * 4)),
```

```
PROGRAME_DATA);
                }
            }
#if TIMER_CNT_MESURE_METHOD
            /* clear timer count and enable timer */
            timer_disable(TIMER_USE);
            timer_disable(TIMER2);
            TIMER_CNT(TIMER_USE) = 0;
            TIMER_CNT(TIMER2) = 0;
            timer_enable(TIMER_USE);
            timer_enable(TIMER2);
#else
            /* set gpio pin to high level */
            gpio_bit_set(GPIOC, GPIO_PIN_0);
#endif
            /* start page erase */
            fmc_page_erase(PAGE_TO_ERASE1);
#if TIMER_CNT_MESURE_METHOD
            /* get the page erase time */
            flash_operation_time.page_erase[measure_counts] = TIMER_CNT(TIMER_USE) +
65536*TIMER_CNT(TIMER2);
#else
            /* set gpio pin to low level */
            gpio_bit_reset(GPIOC, GPIO_PIN_0);
#endif
    }while(++measure_counts < MEASURE_NUMS);
#if TIMER_CNT_MESURE_METHOD
    /* get the average page erase time */
    {
        uint32_t temp =0;
        for(measure_counts =0; measure_counts < MEASURE_NUMS; measure_counts++)
        {
            temp += flash_operation_time.page_erase[measure_counts];
        }
        flash_operation_time.page_erase[AVERAGE_VALUE_POSITION] = temp /
MEASURE_NUMS;
    }
#endif
#endif
    /* word programme measure */
#if TEST_WORD_PROGRAMME
measure_counts = 0;
do{
```

```
        fmc_page_erase(PAGE_TO_ERASE1);
#if TIMER_CNT_MESURE_METHOD
        /* clear timer count and enable timer */
        timer_disable(TIMER_USE);
        timer_disable(TIMER2);
        TIMER_CNT(TIMER_USE) = 0;
        TIMER_CNT(TIMER2) = 0;
        timer_enable(TIMER_USE);
        timer_enable(TIMER2);
#else
        /* set gpio pin to high level */
        gpio_bit_set(GPIOC, GPIO_PIN_0);
#endif
        /* start word programme */
        fmc_word_program(ADDRESS_TO_PROGRAMME +(4*measure_counts) ,
PROGRAMME_DATA);
#if TIMER_CNT_MESURE_METHOD
        /* get the word programme time */
        flash_operation_time.word_programme[measure_counts] = TIMER_CNT(TIMER_USE)+
65536*TIMER_CNT(TIMER2);
#else
        /* set gpio pin to low level */
        gpio_bit_reset(GPIOC, GPIO_PIN_0);
#endif
    }while(++measure_counts < MEASURE_NUMS);
#if TIMER_CNT_MESURE_METHOD
    /* get the average word programme time */
    {
        uint32_t temp =0;
        for(measure_counts =0; measure_counts < MEASURE_NUMS; measure_counts++)
        {
            temp += flash_operation_time.word_programme[measure_counts];
        }
        flash_operation_time.word_programme[AVERAGE_VALUE_POSITION] = temp /
MEASURE_NUMS;
    }
#endif
#endif

#if TIMER_CNT_MESURE_METHOD
    /* print flash operation time by usart */
    printf("word programme time:%.2f(us)\r\npage erase time:%.2f (us)\r\nmass erase time:%.2f
(us)\r\n",
```

```
                flash_operation_time.word_programme[AVERAGE_VALUE_POSITION]*0.125, \
                flash_operation_time.page_erase[AVERAGE_VALUE_POSITION]*0.125, \
                flash_operation_time.mass_erase[AVERAGE_VALUE_POSITION]*0.125);
#endif
/* infinite loop */
    while(1){

    }
}
```

4. 测试结果

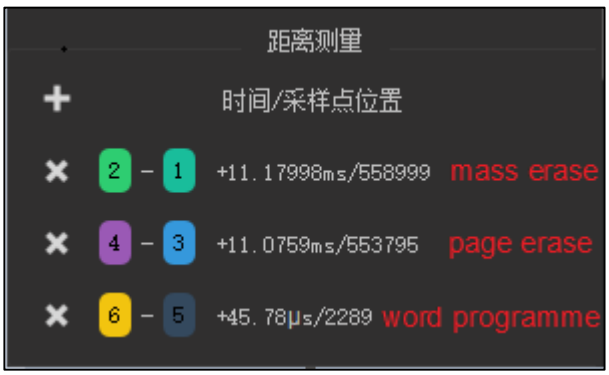
打开宏 TEST_MASS_ERASE、TESE_PAGE_ERASE、TEST_WORD_PROGRAMME、TIMER_CNT_MESURE_METHOD。编译工程，点击调试按钮，点击全速运行。*图 4-1.串口输出 flash 操作时间*结果如下。

图 4-1.串口输出 flash 操作时间

```
word programe time:46.00(us)
page erase time:11072.63(us)
mass erase time:11176.75(us)
```

打开宏 TEST_MASS_ERASE、TESE_PAGE_ERASE、TEST_WORD_PROGRAMME，关闭宏 TIMER_CNT_MESURE_METHOD。编译工程，点击调试按钮，点击全速运行。*图 4-2. 逻辑分析仪输出 flash 操作时间*结果如下。

图 4-2. 逻辑分析仪输出 flash 操作时间



5. 版本历史

表 5-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2021 年 11 月 8 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.