

**GigaDevice Semiconductor Inc.**

**Arm<sup>®</sup> Cortex<sup>®</sup>- M3/M4/M23/M33 32-bit MCU**

**应用笔记**

**AN018**

# 目录

目录.....	2
图索引.....	3
表索引.....	4
1. 简介.....	5
2. MPU 应用.....	6
2.1. 开发环境.....	6
2.2. 开发目标.....	6
2.3. 代码解读.....	6
3. 特权模式和用户模式切换.....	11
3.1. 开发环境.....	11
3.2. 特权模式和用户模式切换简介.....	11
3.3. 开发目标.....	11
3.4. 代码解读.....	11
4. 版本历史.....	15

## 图索引

图 2-1. MPU 流程图.....	7
图 3-1. 特权模式与用户模式的切换.....	11

## 表索引

表 2-1. 主程序代码.....	7
表 2-2. mpu_setup()函数代码.....	8
表 2-3. mpu_access_permission()函数代码.....	10
表 3-1. 主程序代码.....	12
表 4-1. 版本历史.....	15

## 1. 简介

存储保护单元（MPU）是 Cortex-M3/M4 控制器中一个可以选配的单元，它可以实施对存储器的保护，从而使软件系统更加健壮和可靠。更多关于 MPU 的基础知识在 Cortex-M3 权威指南中文版的第 14 章有非常详细的介绍，或者查看 Cortex-M4 权威指南英文版第 11 章。本应用笔记基于 GD32F4xx 系列，分别介绍如何根据应用设置 MPU 以及如何实现特权模式和用户模式之间的切换。

## 2. MPU 应用

IAP 程序通常由两个部分组成: Bootloader 和 APP。Bootloader 和 APP 分别为两个工程程序, 存放在 Flash 的 Main Flash 区, 即 0x08000000 开始的区域。

### 2.1. 开发环境

开发板: GD32450Z-EVAL

开发工具: GD32F4xx\_MPU\_example

Keil/IAR

### 2.2. 开发目标

在软件中定义一个数组 `PrivilegedReadOnlyArray`, 并把该数组所属的区域属性定义为特权级下只读, 验证对该区域进行写操作时会导致存储管理异常。

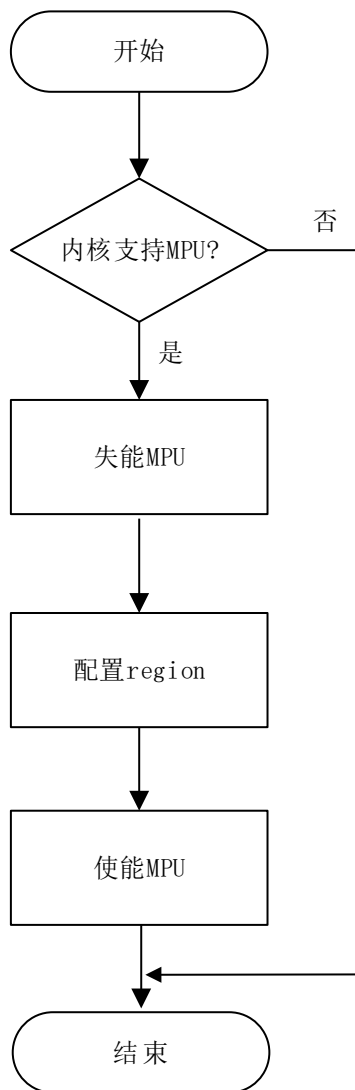
### 2.3. 代码解读

MPU 的编程可以分成四个步骤:

- (a) 通过 MPU 类型寄存器检查内核是否支持 MPU;
- (b) 失能 MPU;
- (c) 根据应用需要配置每个 region 的大小和属性;
- (d) 使能 MPU。

流程图如 [图 2-1. MPU 流程图](#) 所示。

图 2-1. MPU 流程图



在主程序中，先配置 `systick`，然后初始化 `LED1` 和 `LED2`，接着判断内核是否支持 `MPU`。如果不支持，程序就进入无限循环。如果支持，就对 `MPU` 进行配置。最后让 `LED1` 每隔一秒钟翻转一次。主程序代码如[表 2-1. 主程序代码](#)所示。

表 2-1. 主程序代码

```
int main(void)
{
    systick_config();
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    gd_eval_led_off(LED2);

    /* if core dose not support MPU, go to infinite loop */
    if(FALSE == mpu_test()){
        while(1);
    }
}
```

```

    }

    mpu_setup();
    mpu_access_permission_config();
    while(1){
        /* toggle led1*/
        gd_eval_led_toggle(LED1);
        delay_1ms(1000);
    }
}

```

在 `mpu_setup()` 函数中对 RAM、FLASH、外设所属区域的属性进行配置，在 `mpu_access_permission_config()` 函数中对数组 `PrivilegedReadOnlyArray` 所属区域的属性进行配置。下面着重介绍 `mpu_setup()` 函数和 `mpu_access_permission_config()` 函数。

在 `mpu_setup()` 函数中，程序开始先失能 MPU，然后对 RAM 所在的区域进行属性设置。设置 RAM 为 region 0，起始地址为 `0x20000000`，大小为 8kB，在特权级和用户级下可读可写。不可共享，不可缓存，不可缓冲，允许取指。接着对 FLASH 所在的区域进行属性设置。设置 FLASH 为 region 1，起始地址为 `0x08000000`，大小为 1MB，在特权级和用户级下可读可写。不可共享，不可缓存，不可缓冲，允许取指。最后对外设所在的区域进行属性设置。设置外设为 region 2，起始地址为 `0x40000000`，大小为 512MB，在特权级和用户级下可读可写。不可共享，不可缓存，不可缓冲，不允许取指。设置完成后，使能 MPU。`mpu_setup()` 函数的代码如 [表 2-2. mpu\\_setup\(\) 函数代码](#) 所示。

**表 2-2. mpu\_setup() 函数代码**

```

void mpu_setup(void)
{
    mpu_region_init_struct mpu_init_struct;

    /* disable MPU */
    mpu_disable();

    /* configure RAM region as region 0, 8kB of size and R/W region */
    mpu_init_struct.enable           = MPU_REGION_ENABLE;
    mpu_init_struct.base_address    = RAM_BASE_ADDRESS;
    mpu_init_struct.region_size     = RAM_SIZE;
    mpu_init_struct.access_permission=MPU_REGION_PRIV_READ_WRITE_USER_READ_WRITE;

    mpu_init_struct.bufferable      = MPU_ACCESS_NOT_BUFFERABLE;
    mpu_init_struct.cacheable      = MPU_ACCESS_NOT_CACHEABLE;
    mpu_init_struct.shareable      = MPU_ACCESS_NOT_SHAREABLE;
    mpu_init_struct.number         = MPU_REGION_NUMBER_0;
    mpu_init_struct.type_extension_field = MPU_TEX_LEVEL_0;
    mpu_init_struct.sub_region_disable = MPU_SUB_REGION_ALL_DISABLE;
    mpu_init_struct.instruction_accessable = MPU_INSTRUCTION_ACCESS_ENABLE;
}

```



```
mpu_region_config(&mpu_init_struct);

/* configure FLASH region as region 1, 1MB of size and R/W region */
mpu_init_struct.base_address      = FLASH_BASE_ADDRESS;
mpu_init_struct.region_size      = FLASH_SIZE;
mpu_init_struct.number           = MPU_REGION_NUMBER_1;

mpu_region_config(&mpu_init_struct);

/* configure peripheral region as region 2, 512MB of size, R/W and execute never region */
mpu_init_struct.base_address      = PERIPH_BASE_ADDRESS;
mpu_init_struct.region_size      = PERIPH_SIZE;
mpu_init_struct.number           = MPU_REGION_NUMBER_2;
mpu_init_struct.instruction_accessible = MPU_INSTRUCTION_ACCESS_DISABLE;

mpu_region_config(&mpu_init_struct);

/* enable MPU */
mpu_enable(MPU_HFNMI_DISABLE_PRIVDEF_ENABLE);
}
```

在 `mpu_access_permission_config()` 函数中，程序开始先失能 MPU，然后对数组 `PrivilegedReadOnlyArray` 所在的区域进行属性配置。设置数组为 region 3，起始地址为 `0x20002000`，大小为 `32byte`，仅在特权级下可读。不可共享，不可缓存，不可缓冲，允许取指。设置完成后，使能 MPU。由于设置为仅在特权级下可读，因此执行 `read_data = PrivilegedReadOnlyArray[0];` 后不会进入 `MemManage_Handler`。但是，如果执行了写操作 `PrivilegedReadOnlyArray[0] = 'a';` 就会进入 `MemManage_Handler`，点亮 LED2。`mpu_access_permission_config()` 函数的代码如 [表 2-3. mpu\\_access\\_permission\(\) 函数代码](#) 所示。

表 2-3. mpu\_access\_permission()函数代码

```

void mpu_access_permission_config(void)
{
    mpu_region_init_struct mpu_struct;

    /* disable MPU */
    mpu_disable();

    /* configure region for privileged read only array as region 3, 32 byte and read only in
privileged mode */
    mpu_struct.enable                = MPU_REGION_ENABLE;
    mpu_struct.base_address          = ARRAY_BASE_ADDRESS;
    mpu_struct.region_size           = ARRAY_SIZE;

    mpu_struct.access_permission=MPU_REGION_PRIV_READ_ONLY_USER_DISABLE
;
    mpu_struct.bufferable            = MPU_ACCESS_NOT_BUFFERABLE;
    mpu_struct.cacheable            = MPU_ACCESS_NOT_CACHEABLE;
    mpu_struct.shareable            = MPU_ACCESS_NOT_SHAREABLE;
    mpu_struct.number               = MPU_REGION_NUMBER_3;
    mpu_struct.type_extension_field  = MPU_TEX_LEVEL_0;
    mpu_struct.sub_region_disable   = MPU_SUB_REGION_ALL_DISABLE;
    mpu_struct.instruction_accessible = MPU_INSTRUCTION_ACCESS_ENABLE;

    mpu_region_config(&mpu_struct);

    /* enable MPU */
    mpu_enable(MPU_HFNMI_DISABLE_PRIVDEF_ENABLE);

    /* read from privileged read only array. This will not generate error */
    read_data = PrivilegedReadOnlyArray[0];

    /* uncomment the following line to write to privileged read only array. This will generate
error */
    // PrivilegedReadOnlyArray[0] = 'a';
}

```

## 3. 特权模式和用户模式切换

### 3.1. 开发环境

开发板: GD32450Z-EVAL

开发工具: GD32F4xx\_privileged\_mode\_unprivileged\_mode\_switch

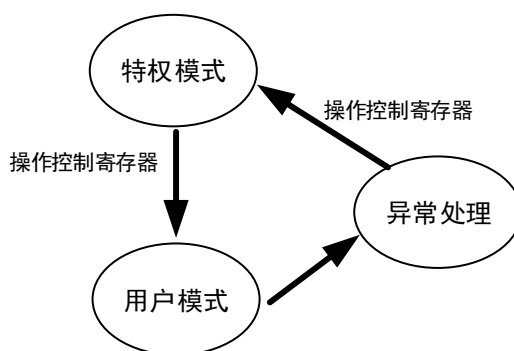
Keil

### 3.2. 特权模式和用户模式切换简介

特权模式和用户模式是 Cortex-M3/M4 支持的两种模式,可以通过控制寄存器的位 0 进行设定。在特权模式下,可以通过置位 CONTROL[0]进入用户模式。用户模式下的代码不能再试图修改 CONTROL[0]来回到特权模式,必须通过一个异常处理操作,比如 SVC(Supervisor Calls)。在异常处理中修改 CONTROL[0],可以从用户模式回到特权模式。特权模式和用户模式的切换过程如 [图 3-1. 特权模式与用户模式的切换](#)所示。在不同的模式下,代码拥有不同的访问权限。当代码运行在特权模式下,它拥有所有的访问许可;而在用户模式下,它的访问权限受到限制。

特权模式和用户模式不是 MPU 的一部分,但是与 MPU 有关联。通过 MPU,可以设定内存的访问规则,这些规则包括内存所在空间在特权级下是否可读可写。因此,通过 MPU 设定内存的读写属性,对该内存进行读写操作,可以更好的理解特权模式和用户模式的切换。

图 3-1. 特权模式与用户模式的切换



### 3.3. 开发目标

通过本例程的学习,开发者可以更好的理解特权模式和用户模式的切换。

### 3.4. 代码解读

在主程序中,先配置 systick 并初始化 LED1、LED2。然后判断内核是否支持 MPU。接着把堆

栈指针从主堆栈指针切换为进程堆栈指针，对 MPU 进行配置。在 `mpu_access_permission_config()` 函数中把数组 `PrivilegedReadOnlyArray` 所属区域的属性定义为特权级下可读可写。把线程模式从特权模式切换为用户模式，此时如果对数组 `PrivilegedReadOnlyArray` 进行写操作，会触发存储管理异常，点亮 LED2。直接修改控制寄存器不能切换回特权模式，必须在异常中修改。程序中进入 SVC 异常，在异常中切换回特权模式，此时在主程序中对数组 `PrivilegedReadOnlyArray` 进行写操作不会触发存储管理异常，进入 `while` 循环，实现 LED1 翻转。主程序代码如 [表 3-1. 主程序代码](#) 所示。

**表 3-1. 主程序代码**

```
int main(void)
{
    systick_config();
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    gd_eval_led_off(LED2);

    /* core dose not support MPU, go to infinite loop */
    if(FALSE == mpu_test()){
        while(1);
    }
    /* switch thread mode stack from main to process */
    /* initialize memory reserved for process stack */
    for(Index = 0; Index < SP_PROCESS_SIZE; Index++)
    {
        PSPMemAlloc[Index] = 0x00;
    }

    /* set process stack value */
    __set_PSP((uint32_t)PSPMemAlloc + SP_PROCESS_SIZE);

    /* select process stack as thread mode stack */
    __set_CONTROL(SP_PROCESS);

    /* execute ISB instruction to flush pipeline as recommended by arm */
    __ISB();

    /* get the thread mode stack used */
    if((__get_CONTROL() & 0x02) == SP_MAIN){
        /* main stack is used as the current stack */
        CurrentStack = SP_MAIN;
    }else{
        /* process stack is used as the current stack */
        CurrentStack = SP_PROCESS;
    }
}
```

```

    /* get process stack pointer value */
    PSPValue = __get_PSP();
}

mpu_setup();
mpu_access_permission_config();

/* switch thread mode from privileged to unprivileged */
/* thread mode has unprivileged access */
__set_CONTROL(THREAD_MODE_UNPRIVILEGED | SP_PROCESS);

/* execute ISB instruction to flush pipeline as recommended by arm */
__ISB();

/* unprivileged access mainly affect ability to:
- use or not use certain instructions such as MSR fields
- access System Control Space (SCS) registers such as NVIC and SysTick */

/* check thread mode privilege status */
if((__get_CONTROL() & 0x01) == THREAD_MODE_PRIVILEGED){
    /* thread mode has privileged access */
    ThreadMode = THREAD_MODE_PRIVILEGED;
}else{
    /* thread mode has unprivileged access */
    ThreadMode = THREAD_MODE_UNPRIVILEGED;
}

/* the thread mode is unprivileged now. It will cause MemManage fault if uncomment the following
four lines.*/
// if(PrivilegedReadOnlyArray[0])
// {
//     PrivilegedReadOnlyArray[0] = 'e';
// }

/* switch back thread mode from unprivileged to privileged */
/* try to switch back thread mode to privileged (Not possible, this can be
done only in Handler mode) */
__set_CONTROL(THREAD_MODE_PRIVILEGED | SP_PROCESS);

/* execute ISB instruction to flush pipeline as recommended by arm */
__ISB();

/* generate a system call exception, and in the ISR switch back thread mode to privileged */

```

```
__SVC();

/* check Thread mode privilege status */
if((__get_CONTROL() & 0x01) == THREAD_MODE_PRIVILEGED)
{
    /* Thread mode has privileged access */
    ThreadMode = THREAD_MODE_PRIVILEGED;
}else{
    /* Thread mode has unprivileged access */
    ThreadMode = THREAD_MODE_UNPRIVILEGED;
}

/* the thread mode is privileged now ,write to PrivilegedReadOnlyArray[0] will not cause
MemManage fault */
if(PrivilegedReadOnlyArray[0])
{
    PrivilegedReadOnlyArray[0] = 'e';
}

while(1){
    /* toggle led1*/
    gd_eval_led_toggle(LED1);
    delay_1ms(1000);
}
}
```

## 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	2021 年 11 月 30 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 its 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 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 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.