```
#include "./SYSTEM/sys/sys.h"
#include "./SYSTEM/usart/usart.h"
#include "./SYSTEM/delay/delay.h"
#include "./BSP/LED/led.h"
#include "./BSP/LCD/lcd.h"
#include "./BSP/KEY/key.h"
#include "./BSP/TOUCH/touch.h"
#include "./BSP/SRAM/sram.h"
#include "./MALLOC/malloc.h"
#include "./BSP/TIMER/btim.h"
#include "GUI.h"
#include "WM.h"



int main(void)
{
```

```
    HAL_Init();                                 /* 初始化 HAL 库
*/

    sys_stm32_clock_init(336, 8, 2, 7); /* 设置时钟, 168Mhz */

    delay_init(168);                            /* 延时初始化 */

    usart_init(9600);                           led_init();
/* 初始化继电器输出*/

    lcd_init();                                 /* 初始化 LCD */

    key_init();                                 /* 初始化光电输入
*/

    tp_dev.init();                              /* 触摸屏初始化 */

    sram_init();                                /* SRAM 初始化 */


    my_mem_init(SRAMIN);                        /* 初始化内部
SRAM 内存池 */

    my_mem_init(SRAMEX);                        /* 初始化外部
SRAM 内存池 */

    my_mem_init(SRAMCCM);                       /* 初始化内部
CCM 内存池 */
```

```
    btim_timx_int_init(999,83);             /* 定时 1ms 提供
emwin 时基 */
    __HAL_RCC_CRC_CLK_ENABLE();
    GUI_Init();



    usart_init(9600);                       /* 串口初始化为 115200
*/
    emwin_main(); //进入主程序


    while(1);


}
void sram_init(void)
{
    GPIO_InitTypeDef gpio_init_struct;
```

```
    FSMC_NORSRAM_TimingTypeDef fsmc_readwritetim;


    SRAM_CS_GPIO_CLK_ENABLE();      /* SRAM_CS 脚时钟使能 */
    SRAM_WR_GPIO_CLK_ENABLE();      /* SRAM_WR 脚时钟使能 */
    SRAM_RD_GPIO_CLK_ENABLE();      /* SRAM_RD 脚时钟使能 */
    __HAL_RCC_FSMC_CLK_ENABLE();    /* 使能 FSMC 时钟 */
    __HAL_RCC_GPIOD_CLK_ENABLE(); /* 使能 GPIOD 时钟 */
    __HAL_RCC_GPIOE_CLK_ENABLE(); /* 使能 GPIOE 时钟 */
    __HAL_RCC_GPIOF_CLK_ENABLE(); /* 使能 GPIOF 时钟 */
    __HAL_RCC_GPIOG_CLK_ENABLE(); /* 使能 GPIOG 时钟 */
```

```
    gpio_init_struct.Pin = SRAM_CS_GPIO_PIN;
    gpio_init_struct.Mode = GPIO_MODE_AF_PP;
    gpio_init_struct.Pull = GPIO_PULLUP;
    gpio_init_struct.Speed = GPIO_SPEED_FREQ_HIGH;
    gpio_init_struct.Alternate = GPIO_AF12_FSMC;
    HAL_GPIO_Init(SRAM_CS_GPIO_PORT, &gpio_init_struct);
/* SRAM_CS 引脚模式设置  */

    gpio_init_struct.Pin = SRAM_WR_GPIO_PIN;
    HAL_GPIO_Init(SRAM_WR_GPIO_PORT,
&gpio_init_struct); /* SRAM_WR 引脚模式设置  */

    gpio_init_struct.Pin = SRAM_RD_GPIO_PIN;
    HAL_GPIO_Init(SRAM_RD_GPIO_PORT, &gpio_init_struct);
/* SRAM_CS 引脚模式设置  */
```

```
    /* PD0,1,4,5,8~15 */

    gpio_init_struct.Pin = GPIO_PIN_0 | GPIO_PIN_1 |
GPIO_PIN_8 | GPIO_PIN_9 |

                           GPIO_PIN_10 | GPIO_PIN_11 |
GPIO_PIN_12 | GPIO_PIN_13 |

                           GPIO_PIN_14 | GPIO_PIN_15;

    gpio_init_struct.Mode = GPIO_MODE_AF_PP;          /* 推
挽复用 */

    gpio_init_struct.Pull = GPIO_PULLUP;              /* 上拉
*/

    gpio_init_struct.Speed = GPIO_SPEED_FREQ_HIGH; /* 高
速 */

    HAL_GPIO_Init(GPIOD, &gpio_init_struct);



    /* PE0,1,7~15 */

    gpio_init_struct.Pin = GPIO_PIN_0 | GPIO_PIN_1 |
GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9 |

                           GPIO_PIN_10 | GPIO_PIN_11 |
```

```
GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 |
                                        GPIO_PIN_15;
        HAL_GPIO_Init(GPIOE, &gpio_init_struct);

        /* PF0~5,12~15 */
        gpio_init_struct.Pin = GPIO_PIN_0 | GPIO_PIN_1 |
GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
                                        GPIO_PIN_5 | GPIO_PIN_12 |
GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
        HAL_GPIO_Init(GPIOF, &gpio_init_struct);

        /* PG0~5,10 */
        gpio_init_struct.Pin = GPIO_PIN_0 | GPIO_PIN_1 |
GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5;
        HAL_GPIO_Init(GPIOG, &gpio_init_struct);

        g_sram_handler.Instance = FSMC_NORSRAM_DEVICE;
```

```
    g_sram_handler.Extended =
FSMC_NORSRAM_EXTENDED_DEVICE;


    g_sram_handler.Init.NSBank = (SRAM_FSMC_NEX == 1) ?
FSMC_NORSRAM_BANK1 : \
                                 (SRAM_FSMC_NEX ==
2) ? FSMC_NORSRAM_BANK2 : \
                                 (SRAM_FSMC_NEX ==
3) ? FSMC_NORSRAM_BANK3 :

FSMC_NORSRAM_BANK4; /*  根据配置选择 FSMC_NE1~4 */

    g_sram_handler.Init.DataAddressMux =
FSMC_DATA_ADDRESS_MUX_DISABLE;        /*  地址/数据线不
复用  */

    g_sram_handler.Init.MemoryType =
FSMC_MEMORY_TYPE_SRAM;                /* SRAM */

    g_sram_handler.Init.MemoryDataWidth =
FSMC_NORSRAM_MEM_BUS_WIDTH_16;       /* 16 位数据宽度  */

    g_sram_handler.Init.BurstAccessMode =
```

FSMC_BURST_ACCESS_MODE_DISABLE; /* 是否使能突发访问,仅对同步突发存储器有效,此处未用到 */

g_sram_handler.Init.WaitSignalPolarity = FSMC_WAIT_SIGNAL_POLARITY_LOW; /* 等待信号的极性,仅在突发模式访问下有用 */

g_sram_handler.Init.WaitSignalActive = FSMC_WAIT_TIMING_BEFORE_WS; /* 存储器是在等待周期之前的一个时钟周期还是等待周期期间使能 NWAIT */

g_sram_handler.Init.WriteOperation = FSMC_WRITE_OPERATION_ENABLE; /* 存储器写使能 */

g_sram_handler.Init.WaitSignal = FSMC_WAIT_SIGNAL_DISABLE; /* 等待使能位,此处未用到 */

g_sram_handler.Init.ExtendedMode = FSMC_EXTENDED_MODE_DISABLE; /* 读写使用相同的时序 */

g_sram_handler.Init.AsynchronousWait = FSMC_ASYNCHRONOUS_WAIT_DISABLE; /* 是否使能同步传输模式下的等待信号,此处未用到 */

```
        g_sram_handler.Init.WriteBurst =
FSMC_WRITE_BURST_DISABLE;                   /* 禁止突发写 */

        /* FMC 读时序控制寄存器 */

        fsmc_readwritetim.AddressSetupTime = 0x00;
/* 地址建立时间（ADDSET）为 1 个 HCLK 1/72M=13.8ns */

        fsmc_readwritetim.AddressHoldTime = 0x00;
/* 地址保持时间（ADDHLD）模式 A 未用到 */

        fsmc_readwritetim.DataSetupTime = 0x06;
/* 数据保存时间为 6 个 HCLK = 6*1 = 6ns */

        fsmc_readwritetim.BusTurnAroundDuration = 0x00;

        fsmc_readwritetim.AccessMode = FSMC_ACCESS_MODE_A;
/* 模式 A */

        HAL_SRAM_Init(&g_sram_handler, &fsmc_readwritetim,
&fsmc_readwritetim);

    }


    void sram_write(uint8_t *pbuf, uint32_t addr, uint32_t datalen)

    {
```

```
        for (; datalen != 0; datalen--)
        {
            *(volatile uint8_t *)(SRAM_BASE_ADDR + addr) =
*pbuf;
            addr++;
            pbuf++;
        }
    }


    void sram_read(uint8_t *pbuf, uint32_t addr, uint32_t datalen)
    {
        for (; datalen != 0; datalen--)
        {
            *pbuf++ = *(volatile uint8_t *)(SRAM_BASE_ADDR +
addr);
            addr++;
```

```
        }

    }
```