同济大学

嵌入式系统实验大作业报告

课题名称	基于 RTOS 的密码锁设计
学院(系)	电子与信息工程学院
专　　业	电子信息工程
年　　级	2020 级
姓　　名	黄锴康
学　　号	1951706
指导教师	喻剑
起讫时间	2023.6.5--2023.6.17
成　　绩	

目录

一、设计目的 1
二、设计环境 1
三、设计内容 1
四、设计方案及流程 1
五、设计结果及体会 4
1、设计结果 4
2、设计体会 ..... **错误！未定义书签。**
3、思考题 ..... **错误！未定义书签。**
六、参考文献 ..... **错误！未定义书签。**
七、主要程序清单 5

一、设计目的

通过设计制作密码锁项目，熟悉 STM32 开发流程，了解相关软件与硬件相互配合的开发方法，并掌握基于操作系统的设计思路。

二、设计环境

本实验硬件环境为 PC,STM32F103ZE 开发板、TFTLCD 显示屏。

在本实验软件环境为 Keil5 集成开发环境，并采用 ARM 开发的编译器进行编译，可以直接生成二进制文件。PC 系统为 Windows10，同时还有普中软件烧录程序等。

三、设计内容

利用 STM32F 实验平台的相对应模块，设计方案、编写程序(C 语言)、进行调试，实现密码锁，该系统具体功能如下：

- 在 TFTLCD 显示屏上显示密码锁界面，包括数字密码键盘和密码输入框。
- 用户可以通过按键在密码输入框中输入数字密码，并以星号形式显示。
- 密码验证功能，当输入的密码达到预设长度后，系统会进行密码验证：如果输入的密码与预设的密码匹配，TFT LCD 上显示成功图像；如果输入的密码不正确，TFT LCD 提示重新输入。

四、设计方案及流程

4.1 设计方案

1.基于操作系统的设计

在本项目中我们使用 CMSIS-RTOS2 实时操作系统，通过调用 CMSIS-RTOS2 API 即可实现操作系统的多个操作，包括多任务处理、同步和通信等功能。Keil RTX5 的优点包括高效、可靠、易于使用和与 Keil MDK 紧密集成。通过使用 CMSIS-RTOS2 API 和 Keil RTX5，开发人员可以更方便地开发和管理嵌入式系统中的多任务处理、同步和通信等功能。

为了通过操作系统完成设计，必须理解以下的一些概念：

- **线程（Thread）：** 线程是操作系统中最小的执行单位。一个进程可以包含多个线程，每个线程都有自己的执行路径和执行状态。线程可以独立执行任务，并与其他线程共享进程的资源。在本项目中不同线程完成不同的任务，以实现密码锁的功能。
- **任务间通信（Inter-Process Communication, IPC）：** 任务间通信是指不同线程或不同进程之间进行数据交换和信息传递的机制。任务间通信允许线程或进程之间共享数据、同步操作和进行协作。常见的任务间通信方式包括消息队列、信号量、互斥量、管道、共享内存等。在本项目中不同线程之间的通信方式为标志位通信。

2.任务设计

基于操作系统的设计，不同任务分类与编写对于系统最后能否正常运行十分重要，在本项目中主要将任务分为以下几个：

- 显示任务：负责 TFTLCD 的显示和更新，包括初始 UI 界面的显示，按键的焦点显示、输入框的星号显示。
- 输入任务：处理按键输入和密码输入的逻辑。根据不同的按键与不同的线程通信，从而实现密码的输入与显示更新。
- 验证任务：验证输入的密码是否与预设的密码匹配。也就是密码判断功能。

3.UI 设计

本项目包含一个数字键盘 UI 设计，也是用户和系统的主要交互界面，对于 UI 设计我主要考虑以下三个方案：

vvgl 图形库移植：

vvgl 提供丰富的图形库，可以帮助简化界面设计和开发过程，同时 stm32 平台支持 vvgl 图形库，因此移植 vvgl 到项目中是可行的。但由于 vvgl 的资源占用较高，对于本项目中资源消耗并不合适。因此放弃选择该方案。

图片显示：

准备好预先设计好的 UI 界面图片，并在显示屏上显示这些图片。这也是指导书提供的主流方案，通过图片显示界面简单且直观，不需要复杂的图形库或绘制逻辑。但图片显示方式限制了交互性，同时在实际调试中发现，图片显示 UI 会存在一定的延迟，图片刷新较慢，不满足实时性的要求，因此该方案不合适

通过画图函数显示 UI：

最后使用的方案为使用画图函数（如 LCD_DrawRect、LCD_DrawLine 等）在屏幕上绘制界面，这些函数由显示驱动提供。实际上这种方案的逻辑与 vvgl 图形库的逻辑一样，只是没有使用专业的图形库，使得显示 UI 比较简单。同时在使用画图函数绘制界面时，可需要手动计算坐标、尺寸等参数。在后续也会简述 UI 设计的手稿。

4.2 设计流程

1. 线程设计

针对以上设计方案，做出来以下线程设计，如图 4.1 所示。

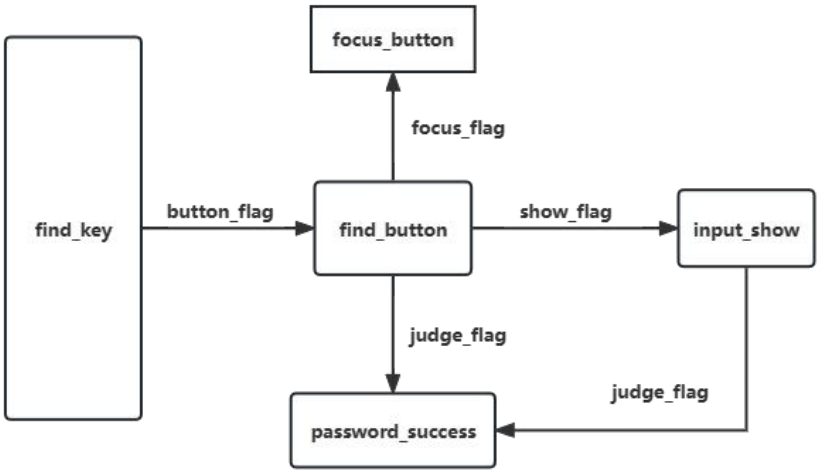

图 4.1　系统线程运行图

- find_key 任务

不断调用 KEY_Scan 函数扫描哪个按键被按下，该任务是一直在执行而没有阻塞。有考虑过

将 find_key 和 find_button 合并，但这会造成扫描功能错误。

- find_button 任务

由于开发板只有 4 个按键，因此将按键功能定义为如表 4.1 所示，

左键	右键	上键	下键
选择框往左移	选择框往右移	直接验证密码	确认选择

当选择框在键盘最左端还按左键，则移动到上一行最后一个数字；当选择框在键盘最右端还按右键，则移动到下一行第一个数字。根据 button_flag 标志决定发送哪一个信号，移动选择框的功能，发送 focus_flag 标志位通知其他任务；验证密码的功能，发送 judge_flag 标志位通知其他任务；显示输入功能通过 show_flag 实现。

- focus_button 任务

为了简化代码，在此我通过创建 NumberInfo 的结构体，描述数字按键的位置、值以及显示的字符，通过创建结构体数组，并且通过接受到的 focus_flag 找到现在选择框所在的数字，同时要将上一个被选择框选中的数字清空，从而实现了选择框的显示和更新。

- input_show 任务

该线程负责根据接收到的标志位信息处理输入密码的逻辑并更新 UI 界面的显示。判断输入密码的情况。如果已输入 5 位密码，此时向 judge_flag 标志位所在的线程发送消息，通知密码输入完成。

否则，根据接收到的消息进行处理：如果接收到的消息的取值在 0x0001 到 0x0009 之间，表示输入的是数字键，将对应的星号字符显示在 LCD 屏幕上，并将输入的密码保存到密码数组中，增加已输入密码的位数。如果接收到的消息等于 0x000B，表示输入的是清除键，将对应的星号字符显示在 LCD 屏幕上，并将 0 存入密码数组中，然后将输入密码位数递增。如果接收到的消息等于 0x00A，表示输入的是退格键，将 input_num 递减，清除 LCD 屏幕上的输入框，并重新显示已输入的密码。如果接收到的消息等于 0x000C，表示输入的是确认键，向 judge_flag 标志位所在的线程发送消息，通知密码输入完成。

- password_success 任务

当接收到标志位通知后，线程会调用 JudgePassword()函数进行密码验证。如果密码验证成功，线程会清除 LCD 屏幕上的内容，并显示一个正确的图片。如果密码验证失败，线程会清除 LCD 屏幕上的内容，并显示一个错误的图片。最后，线程会重置一些变量（num_state、old_state 和 input_num）的值，并调用 ui_setup()函数重新设置 UI 界面。然后线程继续循环，等待下一次标志位的通知。

2. UI 设计

由于采用 TFTLCD 的画图函数实现 UI，因此必须考虑 UI 的像素位置，UI 设计如图 4.2 所示

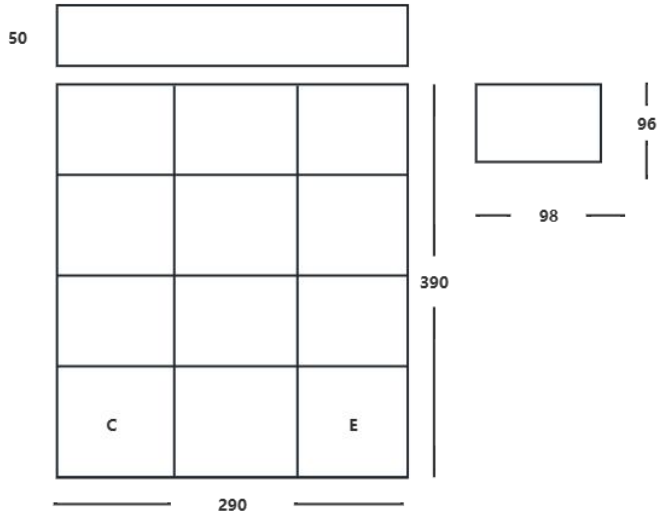

图 4.2　UI 设计图及参数

输入框和数字键盘的参数也在表中，通过 TFTLCD_DrawLine、TFTLCD_Fill 等函数实现 UI 的设计。

五、设计结果及体会

1、设计结果

开发板运行初始界面，如图 5.1 所示。

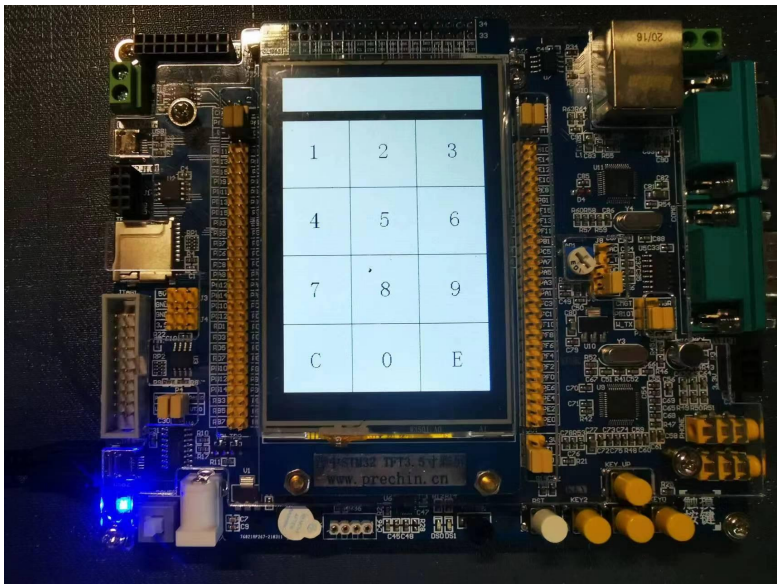

图 5.1　系统初始界面

控制按钮选择密码输入，输入选中密码的焦点以及密码正确图案如图 5.2 所示，

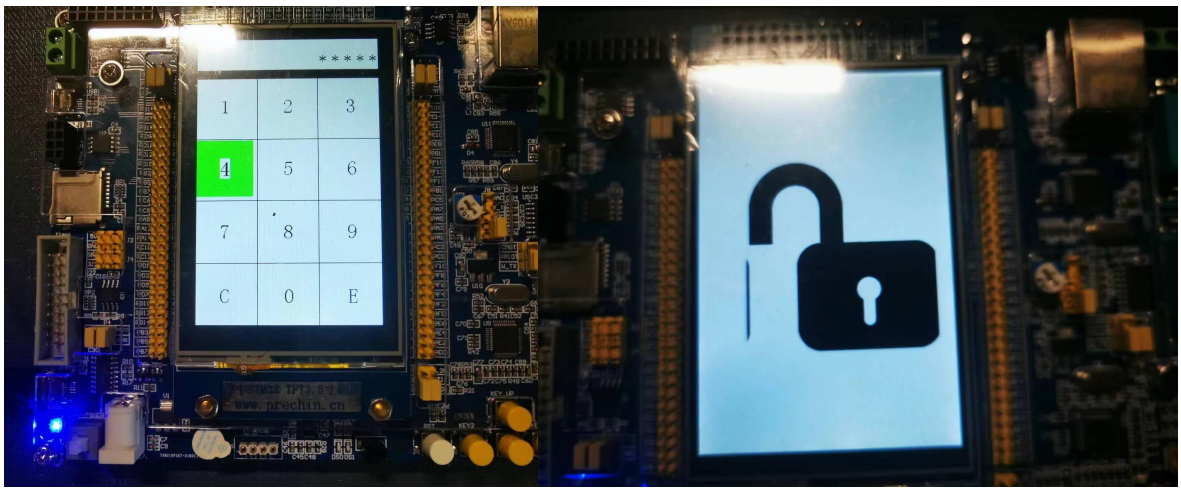

图 5.2　选中密码和密码正确图

当输入密码错误时，系统显示错误图案，并且在 5s 后自动返回初始化界面。

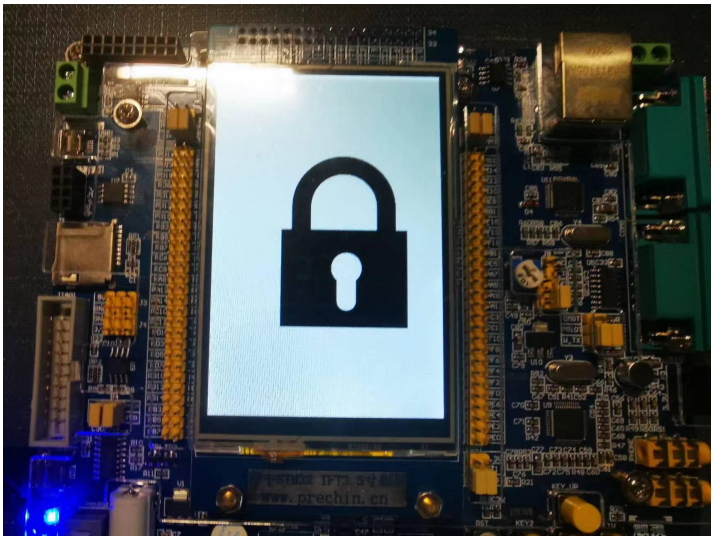

图 5.2 选中密码和密码正确图

2、设计体会

当我完成这个程序时，我深入学习了多线程并发设计和线程间通信的概念和应用，我理解了多线程并发的好处，可以充分利用处理器资源，提高系统的响应性能。线程间的通信机制让不同线程之间能够传递信息和协作，实现复杂的功能和任务分配。

在这个实验中，我注意到了一些关键点。首先，代码的结构清晰，函数和变量的命名具有描述性，这便于增加更容易理解代码的逻辑和功能。其次，通过使用标志位来管理和切换系统的状态，能够控制不同功能的执行和行为。这种状态管理的方式让代码更加灵活和可扩展。

同时，在多线程环境中，需要小心处理竞争条件和数据访问冲突，以避免出现意外的行为或结果。在实际应用中，我需要更加细致地检查代码，确保线程间的操作不会相互干扰或冲突。

通过这个实验，我对多线程并发设计有了更深入的理解，并掌握了线程间通信的基本方法。这将对我在嵌入式系统开发和并发编程方面的工作和学习有很大帮助。我计划进一步探索并发编程的相关概念和技术，并在未来的项目中应用和拓展这些知识。

七、主要程序清单

附录

完整的工程项目

```
#include "tftlcd.h"
#include "rtx_os.h"
#include "stm32f10x_gpio.h"
#include "key.h"
#include "wrong.h"
#include "right.h"


osThreadId_t key_flag,button_flag,focus_flag,show_flag,judge_flag;
uint32_t flagsX1,flagsX2,flagsX3,flagsX4,flagsX5;


#define PASSWORD_LENGTH 5
uint8_t password[PASSWORD_LENGTH];
uint8_t presetPassword[PASSWORD_LENGTH] = {1, 1, 1, 1, 1};
static u8 fac_us=0;
static u16 fac_ms=0;
static u16 state=1;
```

```
static u16 old_state=1;
static u16 input_num = 0;
typedef struct {
    uint8_t value;
    uint16_t x1, y1, x2, y2;
    const char* displayChar;
} NumberInfo;

NumberInfo numbers[] = {
    {0x0001, 15, 75, 104, 164, "1"},
    {0x0002, 115, 75, 204, 164, "2"},
    {0x0003, 215, 75, 304, 164, "3"},
    {0x0004, 15, 175, 104, 264, "4"},
    {0x0005, 115, 175, 204, 264, "5"},
    {0x0006, 215, 175, 304, 264, "6"},
    {0x0007, 15, 275, 104, 364, "7"},
    {0x0008, 115, 275, 204, 364, "8"},
    {0x0009, 215, 275, 304, 364, "9"},
    {0x000A, 15, 375, 104, 464, "C"},
    {0x000B, 115, 375, 204, 464, "0"},
    {0x000C, 215, 375, 304, 464, "E"}
};


void SysTick_Init(u8 SYSCLK){
    SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
    fac_us=SYSCLK/8;
    fac_ms=(u16)fac_us*1000;
}

void delay_us(u32 nus){
    u32 temp;
    SysTick->LOAD=nus*fac_us;
    SysTick->VAL=0x00;
    SysTick->CTRL|=SysTick_CTRL_ENABLE_Msk ;
    do{
        temp=SysTick->CTRL;
    }while((temp&0x01)&&!(temp&(1<<16)));
    SysTick->CTRL&=~SysTick_CTRL_ENABLE_Msk;
    SysTick->VAL =0X00;
}

void delay_ms(u16 nms){
    u32 temp;
    SysTick->LOAD=(u32)nms*fac_ms;
    SysTick->VAL =0x00;
    SysTick->CTRL|=SysTick_CTRL_ENABLE_Msk ;
    do{
        temp=SysTick->CTRL;
    }while((temp&0x01)&&!(temp&(1<<16)));
    SysTick->CTRL&=~SysTick_CTRL_ENABLE_Msk;
    SysTick->VAL =0X00;
}

void KEY_GPIO_Init(void){
    GPIO_InitTypeDef GPIO_InitStruct;
```

装 订 线

```
    RCC_APB2PeriphClockCmd(KEY0_RCC_APB2Periph_CLK | KEY_UP_RCC_APB2Periph_CLK,
ENABLE);

    GPIO_InitStruct.GPIO_Pin = KEY0_GPIO_Pin;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_Init(KEY0_GPIO_Port, &GPIO_InitStruct);

    GPIO_InitStruct.GPIO_Pin = KEY1_GPIO_Pin;
    GPIO_Init(KEY1_GPIO_Port, &GPIO_InitStruct);

    GPIO_InitStruct.GPIO_Pin = KEY2_GPIO_Pin;
    GPIO_Init(KEY2_GPIO_Port, &GPIO_InitStruct);

    GPIO_InitStruct.GPIO_Pin = KEY_UP_GPIO_Pin;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(KEY_UP_GPIO_Port, &GPIO_InitStruct);
}


u8 KEY_Scan(u8 mode){
    static u8 key_up = 1;
    if(mode) key_up = 1;
    if(KEY0 == 1 && KEY1 == 1 && KEY2 == 1 && WK_UP == 0)
    {
        key_up = 1;
    }
    if(key_up && (KEY0 == 0  || KEY1 == 0 || KEY2 == 0 || WK_UP == 1))
    {
        //delay_ms(10);
        //if(KEY0 == 0 || KEY1 == 0 || KEY2 == 0 || WK_UP == 1)
        //{
            key_up = 0;
            if(KEY0 == 0) return KEY0_PRESS;
            if(KEY1 == 0) return KEY1_PRESS;
            if(KEY2 == 0) return KEY2_PRESS;
            if(WK_UP == 1) return WK_UP_PRESS;
        //}
    }
    return 0;
}


void find_key(void *argument){
    uint8_t key_sta = 0;
    while(1){
        key_sta = KEY_Scan(0);
        if(key_sta == KEY0_PRESS){
            osThreadFlagsSet(button_flag, 0x0001U);
            osDelay(50);
        }
        if(key_sta == KEY1_PRESS){
            osThreadFlagsSet(button_flag, 0x0002U);
            osDelay(50);
        }
```

```
        if(key_sta == KEY2_PRESS){
            osThreadFlagsSet(button_flag, 0x0004U);
            osDelay(50);
        }
        if(key_sta == WK_UP_PRESS){
            osThreadFlagsSet(button_flag, 0x0008U);
            osDelay(50);
        }
    }
}
void find_button(void *argument){// 320*480
    while(1){
        flagsX2=osThreadFlagsWait(0x000FU, osFlagsWaitAny, osWaitForever);
        old_state = state;
        uint32_t hexValue = 0;
        if((flagsX2 & 0x0001)==0x0001){
            if(state >= 12)
                state = 1;
            else
                state = state+1;
        }
        if((flagsX2 & 0x0002)==0x0002){
            hexValue = (uint32_t)state;
           osThreadFlagsSet(show_flag, hexValue);
        }
        if((flagsX2 & 0x0004)==0x0004){
            if(state <= 1)
                state = 12;
            else
                state = state-1;
        }
        if((flagsX2 & 0x0008)==0x0008){
            osThreadFlagsSet(judge_flag, 0x0001U);
        }
        hexValue = (uint32_t)state;
    osThreadFlagsSet(focus_flag, hexValue);

    }
}
void ui_setup(){
    LCD_Clear(BLACK);
    LCD_Fill(15,10,304,59,WHITE);//input
    LCD_Fill(15,75,304,464,WHITE);//body

    LCD_DrawLine(15+96,75,15+96,464);
    LCD_DrawLine(15+96*2,75,15+96*2,464);
    for(int i=1;i<4;i++)//horizental
    {
        LCD_DrawLine(15,75+98*i,304,75+98*i);
    }

    LCD_ShowString(15+37,275+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"7
");
    LCD_ShowString(115+37,275+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"
8");
    LCD_ShowString(215+37,275+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"
```

```
9");

    LCD_ShowString(15+37,75+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"1"
);
    LCD_ShowString(115+37,75+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"2
");
    LCD_ShowString(215+37,75+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"3
");


LCD_ShowString(15+37,175+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"4");
    LCD_ShowString(115+37,175+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"
5");

LCD_ShowString(215+37,175+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"6");

    LCD_ShowString(15+37,375+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"C
");
    LCD_ShowString(115+37,375+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"
0");
    LCD_ShowString(215+37,375+29,tftlcd_data.width,tftlcd_data.height,32,(u8*)"
E");
}




void focus_button(void *argument){
    while(1){
        flagsX3=osThreadFlagsWait(0x000FU, osFlagsWaitAny, osWaitForever);
        for (int i = 0; i < sizeof(numbers) / sizeof(numbers[0]); i++) {
            if ((flagsX3 & 0x000F) == numbers[i].value) {
                LCD_Fill(numbers[i].x1, numbers[i].y1, numbers[i].x2,
numbers[i].y2, GREEN);
                LCD_ShowString(numbers[i].x1 + 37, numbers[i].y1 + 29,
tftlcd_data.width, tftlcd_data.height, 32, (u8*)numbers[i].displayChar);
            }
                    else if ((old_state & 0x000F) == numbers[i].value) {
                LCD_Fill(numbers[i].x1, numbers[i].y1, numbers[i].x2,
numbers[i].y2, WHITE);
                LCD_ShowString(numbers[i].x1 + 37, numbers[i].y1 + 29,
tftlcd_data.width, tftlcd_data.height, 32, (u8*)numbers[i].displayChar);
            }
        osThreadFlagsSet(focus_flag, 0x0000U);
        }
    }
}


u8 JudgePassword() {
    for (int i = 0; i < PASSWORD_LENGTH; i++) {
        if (password[i] != presetPassword[i]) {
            return 0;
        }
```

```
    }
    return 1;
}

void input_show(void *argument){
    while(1){
    flagsX4=osThreadFlagsWait(0x000FU, osFlagsWaitAny, osWaitForever);
    if(input_num>=5){
      osThreadFlagsSet(judge_flag, 0x0001U);
    }
    else{
        if(flagsX4>=0x0001 && flagsX4<=0x0009){
            LCD_ShowChar(290-20*input_num,30,'*',24,0);
          password[input_num++]=flagsX4;
        }
        else if(flagsX4 == 0x000B){
           LCD_ShowChar(290-20*input_num,30,'*',24,0);
           password[input_num++]=0;
        }
        else if(flagsX4 == 0x00A){
           if(input_num ==0);
             else{
              input_num--;
                LCD_Fill(15,10,304,59,WHITE);//input
                for(int i =0; i<input_num; i++)
                {
                    LCD_ShowChar(290-20*i,30,'*',24,0);
                }
            }
        }
        else if(flagsX4 == 0x000C){
           osThreadFlagsSet(judge_flag, 0x0001U);
        }
        else;
    }
    osThreadFlagsSet(show_flag, 0x0000U);
    }
}

void password_success(void *argument){
    while(1){
    flagsX5=osThreadFlagsWait(0x000FU, osFlagsWaitAny, osWaitForever);
    if(JudgePassword()){
        LCD_Clear(WHITE);
        LCD_ShowPicture(15+45,75+45,240, 240, (u8 *)gImage_right);
        }
    else{
        LCD_Clear(WHITE);
        LCD_ShowPicture(15+45,75+45,240, 240, (u8 *)gImage_wrong);
        }
    delay_ms(5000);
    state=1;
  old_state=1;
  input_num = 0;
    ui_setup();
    }
```

装 订 线

```
}


void app_main(void *argument){
    key_flag = osThreadNew(find_key, NULL, NULL);
    button_flag = osThreadNew(find_button, NULL, NULL);
    focus_flag = osThreadNew(focus_button, NULL, NULL);
    show_flag = osThreadNew(input_show, NULL, NULL);
    judge_flag = osThreadNew(password_success, NULL, NULL);
    while(1);
}

int main(void){
    KEY_GPIO_Init();
    SysTick_Init(72);
    TFTLCD_Init();
    FRONT_COLOR=BLACK;
    ui_setup();

    SystemCoreClockUpdate();
    osKernelInitialize();
    osThreadNew(app_main, NULL, NULL);
    osKernelStart();

    while(1);

}
```

装 订 线