计算机科学与技术学院

嵌入式系统实验报告

（五）

姓　　名：　Banban

专　　业：　计算机科学与技术

班　　级：

学　　号：

指导教师：

2023 年 4 月 10 日

一、任务要求

1、 结合示例工程项目、串口调试助手理解 USART 操控原理和方法

2、 创建 mdk 工程，编写程序，实现如下任务（任务 1、2 选做其一）：

- 任务 1：根据教材习题 7-9 提供的方法，获取实验用开发板上 MCU 的闪存容量数据，并通过 USART1 发送给 pc 显示。
- 任务 2：STM32 通过串口 1 和上位机（pc）对话，STM32 在收到上位机（pc）发过来的字符串(以回车换行结束)后，原样发送给上位机。
- 任务 3（选做）：开发板读取独立按键或矩阵键盘，将按键信息（如按键编号或按键行列信息）发送给上位 pc 机。

二、实验报告要求

1、 任务 1 - 任务 3 中自编程序的源代码（加上注释）

2、 能说明软件仿真结果的截图、反映硬件电路连接和硬件验证结果的图片或视频

三、实验过程

一. 任务一：获取实验用开发板上 MCU 的闪存容量数据，并通过 USART1 发送给 pc 显示

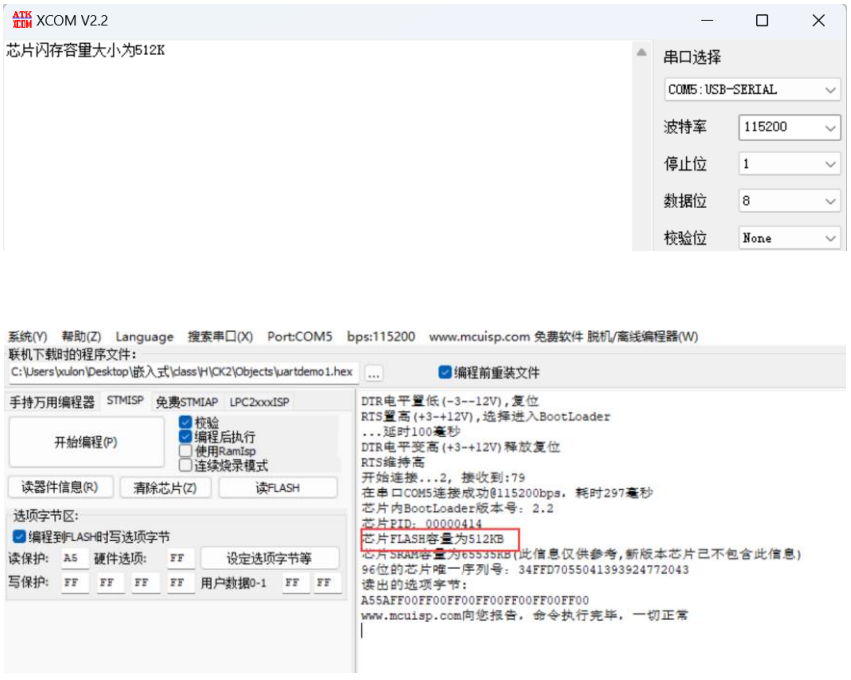
1. 代码：获取 MCU 的闪存容量

```
// main.c
// -----------------------------------------------------------
#include <stdio.h>
#include "stm32f10x.h"
#include "stm32f10x_usart.h"

void RCC_Config(void);
void GPIO_Config(void);
void USART_Config(void);
```

```c
int fputc(int ch, FILE *f);
u16 flash_size;

int main() {
    RCC_Config();
    GPIO_Config();
    USART_Config();
    USART_ClearFlag(USART1, USART_FLAG_TC);
    flash_size = *(unsigned int *)(0x1FFFF7E0);  // 闪存容量
    printf("芯片闪存容量大小为%dK\r\n", flash_size);
}

void RCC_Config(void) {
    // SystemInit();
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
}

void GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    // GPIO_StructInit(&GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;  // USART1_TX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;  // USART1_RX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void USART_Config(void) {
    USART_InitTypeDef USART_InitStructure;
    USART_InitStructure.USART_BaudRate = 115200;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No;
```

```
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Tx;
    USART_Init(USART1, &USART_InitStructure);
    USART_Cmd(USART1, ENABLE);
}

int fputc(int ch, FILE *f) {
    if (ch == '\n') {
        while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
        USART_SendData(USART1, '\r');
    }
    while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
    USART_SendData(USART1, ch);
    return ch;
}
```

2. 图片效果



二. 任务二：STM32 通过串口 1 和上位机（pc）对话，STM32 在收到上位机（pc）发过来的字符串(以回车换行结束)后，原样发送

给上位机

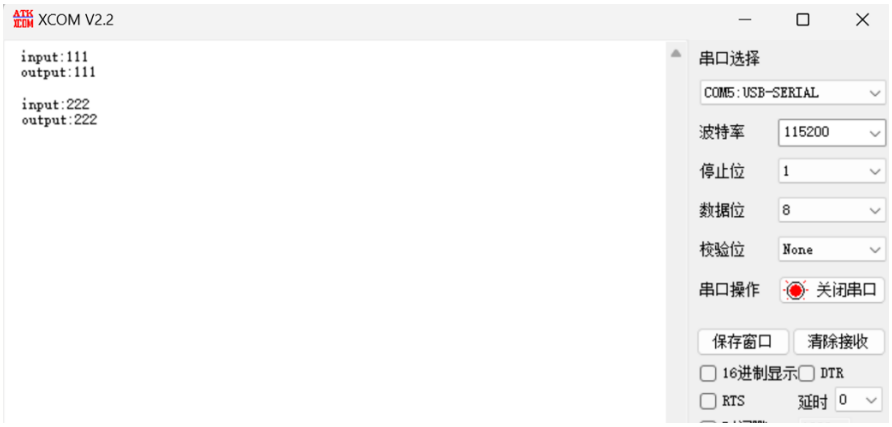
1. 代码

```
// main.c
// -----------------------------------------------------------
#include <stdio.h>
#include "stm32f10x.h"
#include "stm32f10x_usart.h"

void RCC_Config(void);
void GPIO_Config(void);
void USART_Config(void);
int fputc(int ch, FILE *f);
int fgetc(FILE *f);

int main() {
    char msg[100];   // 输入字符串长度最大不超过 100
    int temp = 1;    // 作为对话次数的标志
    RCC_Config();    // 时钟初始化
    GPIO_Config();   // GPIO 初始化
    USART_Config();  // USART 初始化

    while (1) {
        USART_ClearFlag(USART1, USART_FLAG_TC);    // 清除发送标志
        USART_ClearFlag(USART1, USART_FLAG_RXNE);  // 清除接受标志
        printf(" input:\n");
        scanf("%s", msg);  // 键盘输入
        printf("\r\n output:%s\r\n", msg);
    }
}

void RCC_Config(void) {
    // SystemInit();
    // 串口 1 挂在 APB2 上
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    // USART_Tx(PA9)  USART_Rx(PA10)
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
}

void GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;        // USART1_TX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  // 复用推挽输出
    GPIO_Init(GPIOA, &GPIO_InitStructure);           // GPIOA 初始化
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;             // USART1_RX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;  // 浮空输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);                 // GPIOA 初始化
}

void USART_Config(void) {
    USART_InitTypeDef USART_InitStructure;
    USART_InitStructure.USART_BaudRate = 115200;  // 通信波特率
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;  // 一个字节
8bit
    USART_InitStructure.USART_StopBits = USART_StopBits_1;  // 一位停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;     // 无校验
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;  // 收发
模式
    USART_Init(USART1, &USART_InitStructure);  // USART1 初始化
    USART_Cmd(USART1, ENABLE);  // 允许其工作
}

int fputc(int ch, FILE *f) {
    if (ch == '\n') {  // 换行
        // 是否发送完成
        while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
        USART_SendData(USART1, '\r');  // 发送完成则回车
    }
```

```
    while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
    USART_SendData(USART1, ch);  // 继续发送
    return ch;
}

int fgetc(FILE *f) {
    int ch;
    // 是否已接收数据
    while (USART_GetFlagStatus(USART1, USART_FLAG_RXNE) == RESET);
    ch = USART_ReceiveData(USART1);
    // 将接收的数据发送回去，实现键盘的回显功能
    while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET);
    USART_SendData(USART1, (uint8_t)ch);
    return ch;
}
```

2. 图片效果



三. 任务三：（选做）：开发板读取独立按键或矩阵键盘，将按键信息（如按键编号或按键行列信息）发送给上位 pc 机

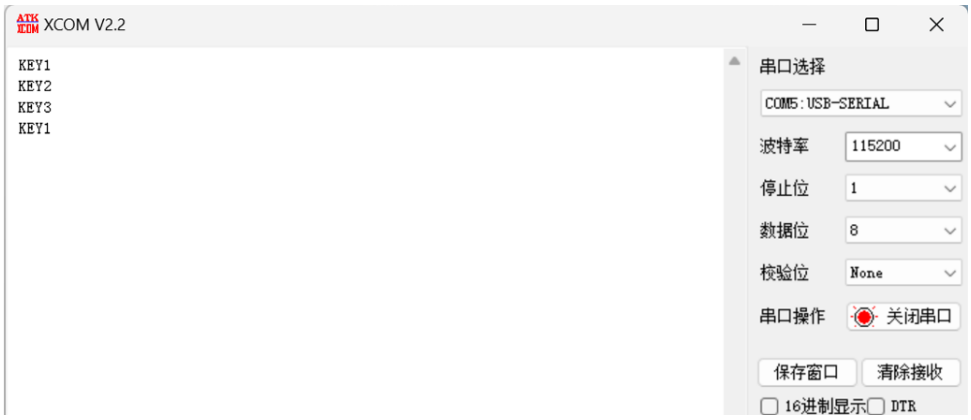
1. 代码

```
// main.c
// -----------------------------------------------------------
int main() {
    int i = 0;
    RCC_Config();   // 时钟初始化
```

```
    GPIO_Config();  // GPIO 初始化
    Key_GPIO_Config();
    USART_Config();                          // USART 初始化
    USART_ClearFlag(USART1, USART_FLAG_TC);  // 清除发送标志
    while (1) {
        for (i = 0; i < 4; i++) {
            if (Key_Scan(GPIOE, a[i]) == KEY_ON) {
                printf("KEY %d\r\n", i);
                USART_ClearFlag(USART1, USART_FLAG_TC);  // 清除发送标志
                break;
            }
        }
        i = 0;
    }
}

void RCC_Config(void) {
    // SystemInit();
    // 串口 1 挂在 APB2 上
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    // USART_Tx(PA9)  USART_Rx(PA10)
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE);
}

void GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;  // USART1_TX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  // 复用推挽输出
    GPIO_Init(GPIOA, &GPIO_InitStructure);           // GPIOA 初始化
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;             // USART1_RX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;  // 浮空输入
    GPIO_Init(GPIOA, &GPIO_InitStructure);                 // GPIOA 初始化
}
```

```
void Key_GPIO_Config(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 |
GPIO_Pin_4;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOE, &GPIO_InitStructure);
}

void USART_Config(void) {
    USART_InitTypeDef USART_InitStructure;
    USART_InitStructure.USART_BaudRate = 9600;  // 通信波特率
    USART_InitStructure.USART_WordLength = USART_WordLength_8b; // 8bit
    USART_InitStructure.USART_StopBits = USART_StopBits_1;  // 一位停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;     // 无校验
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Tx;  // 发模式
    USART_Init(USART1, &USART_InitStructure);        // USART1 初始化
    USART_Cmd(USART1, ENABLE);  // 允许其工作
}
// • • •
```

2. 图片效果



四、总结与分析

在老师的帮助指导下，我顺利完成了本次实验，在获取 MCU 闪存信息的程序中，我们首先获取 MCU 的闪存容量数据，然后初始化 USART1 串口，并通过 USART1 的发送中断将闪存容量数据发送给 PC 端进行显示。在中断处理函数 USART1_IRQHandler()中，当 USART1 接收到 PC 端的请求时，将闪存容量数据通过 USART1 发送给 PC 端。

之后我们使用 USART1 串口进行通信的程序。程序中定义了 USART_Init 函数，用于初始化 USART1 串口并设置相关参数，包括波特率、数据位、停止位、奇偶校验位等。USART1_IRQHandler 函数则是中断服务函数，用于处理 USART1 串口接收中断，并将接收到的数据通过 USART1 发送回去。

最后，在 main 函数中，程序使用 NVIC_PriorityGroupConfig 函数进行中断优先级组的配置，并调用 USART_Init 函数进行初始化，并在 while(1)循环中等待接收和发送数据。