实验五 UART 通信

一、实验目的：

1、了解 ARM 处理器中的的 UART 通信原理；
2、熟悉 UART 控制器的工作原理；
3、掌握与 UART 通信相关的寄存器设置；

二、实验环境：

1、proteus 仿真软件
2、VScode 开发环境
3、ARM gun 交叉编译工具链

三、实验内容

1、配置 USART1 接口，进行串口通信实验，实现发送数据的回显，即通过串口调试工具给处理器发送数据，然后经 USART1 接口接收在发送调试工具显示出来。采用轮询或中断的方式完成。
2、将串口的操作封装成函数。

四、实验步骤

1、查看电路原理图，分析 UART（实验使用串口 1，即 USART1）对应的 GPIO 引脚；
2、串口时钟使能。（APB2）
3、配置引脚对的功能为复用并映射功能； （7.3.2）
4、串口波特率设置
5、串口控制；
6、数据发送与接收（轮询）；
7、完成实验报告

五、进阶

使能接收中断，利用中断处理函数进行串口数据的接收。

附录

ISER[8]：ISER 全称是：Interrupt Set-Enable Registers，这是一个中断使能寄存器组。M4 内核支持 256 个中断，用 8 个 32 位寄存器来控制，每个位控制一个中断。但是 STM32F4 的可屏蔽中断最多只有 82 个，所以有用的就是三个（ISER[0~2]），总共可以表示 96 个中断。而 STM32F4 只用了其中的前 82 个。ISER[0]的 bit0~31 分别对应中断 0~31；ISER[1]的 bit0~32 对应中断 32~63；ISER[2]的 bit0~17 对应中断 64~81；这样总共 82 个 中断就分别对应上了。要使能某个中断，必须设置相应的 ISER 位为 1，使该中断被使能(这里仅仅是使能，还要配合中断分组、屏蔽、IO 口映射等设置才算是一个完整的中断设置)。

IP[240]：全称是：Interrupt Priority Registers，是一个中断优先级控制的寄存器组。这个寄存器组相当重要！STM32F4 的中断分组与这个寄存器组密切相关。IP 寄存器组由 240 个 8bit 的寄存器组成，每个可屏蔽中断占用 8bit，这样总共可以表示 240 个可屏蔽中断。而 STM32F4 只用到了其中的 82 个。IP[81]~IP[0]分别对应中断 81~0。而每个可屏蔽中断占用的 8bit 并没有全部使用，而是只用了高 4 位。这 4 位，又分为抢占优先级和子优先级。抢占优先级在前，子优先级在后。而这两个优先级各占几个位又要根据 SCB->AIRCR 中的中断分组设置来决定。 这里简单介绍一下 STM32F4 的中断分组：STM32F4 将中断分为 5 个组，组 0~4。该分组的设置是由 SCB->AIRCR 寄存器的 bit10~8 来定义的。具体的分配关系如下表所示。

组	AIRCR[10：8]	bit[7：4]分配情况	分配结果
0	111	0：4	0 位抢占优先级，4 位响应优先级
1	110	1：3	1 位抢占优先级，3 位响应优先级
2	101	2：2	2 位抢占优先级，2 位响应优先级
3	100	3：1	3 位抢占优先级，1 位响应优先级
4	011	4：0	4 位抢占优先级，0 位响应优先级

通过这个表，我们就可以清楚的看到组 0~4 对应的配置关系，例如组设置为 3，那么此时所有的 82 个中断，每个中断的中断优先寄存器的高四位中的最高 3 位是抢占优先级，低 1 位是响应优先级。每个中断，你可以设置抢占优先级为 0~7，响应优先级为 1 或 0。抢占优先级的级别高于响应优先级。而数值越小所代表的优先级就越高。

时钟配置

USART1 链接在 APB2S 上，在配置波特率时要清楚设置好 APB2 的时钟频率，以下为时钟配置函数。

```
//外部晶振为 8M 的时候,推荐值:plln=336,pllm=8,pllp=4,pllq=7.
//得到:Fvco=8*(336/8)=336Mhz
//      Fsys=336/4=84Mhz
//      Fusb=336/7=48Mhz
//返回值:0,成功;1,失败。
u8 Sys_Clock_Set(u32 plln,u32 pllm,u32 pllp,u32 pllq)
{
        u16 retry=0;
        u8 status=0;
        RCC->CR|=1<<16;              //HSE 开启
        while(((RCC->CR&(1<<17))==0)&&(retry<0X1FFF))retry++;//等待 HSE R
DY
        if(retry==0X1FFF)
                status=1;    //HSE 无法就绪
        else
        {
                RCC->APB1ENR|=1<<28;     //电源接口时钟使能
                RCC->CFGR|=(0<<4)|(4<<10)|(0<<13);//HCLK 不分频;APB1 2 分
频;APB2 不分频.
                RCC->CR&=~(1<<24);  //关闭主 PLL
                RCC->PLLCFGR=pllm|(plln<<6)|(((pllp>>1)-1)<<16)|
(pllq<<24)|(1<<22);//配置主 PLL,PLL 时钟源来自 HSE
                RCC->CR|=1<<24;          //打开主 PLL
                while((RCC->CR&(1<<25))==0);//等待 PLL 准备好
                RCC->CFGR&=~(3<<0);      //清零
                RCC->CFGR|=2<<0;         //选择主 PLL 作为系统时钟
                while((RCC->CFGR&(3<<2))!=(2<<2));//等待主 PLL 作为系统时钟
成功.
        }
```

```
        return status;
}

//系统时钟初始化函数
//plln:主 PLL 倍频系数(PLL 倍频),取值范围:64~432.
//pllm:主 PLL 和音频 PLL 分频系数(PLL 之前的分频),取值范围:2~63.
//pllp:系统时钟的主 PLL 分频系数(PLL 之后的分频),取值范围:2,4,6,8.(仅限这 4 个
值!)
//pllq:USB/SDIO/随机数产生器等的主 PLL 分频系数(PLL 之后的分频),取值范
围:2~15.
void Stm32_Clock_Init(u32 plln,u32 pllm,u32 pllp,u32 pllq)
{
        RCC->CR|=0x00000001;          //设置 HISON,开启内部高速 RC 振荡
        RCC->CFGR = 0x00000000;       //CFGR 清零
        RCC->CR &= 0xFEF6FFFF;        //HSEON,CSSON,PLLON 清零
        RCC->PLLCFGR=0x24003010;      //PLLCFGR 恢复复位值
        RCC->CR&=~(1<<18);            //HSEBYP 清零,外部晶振不旁路
        RCC->CIR=0x00000000;          //禁止 RCC 时钟中断
        Sys_Clock_Set(plln,pllm,pllp,pllq);//设置时钟
}
```

添加在 main.c 中即可，在 main 函数中调用 Stm32_Clock_Init 函数，设置 n,m,p,q 参数，APB2 的时钟 CFGR 寄存器相应位设置。