一、编写UART\_2串口发送程序时，初始化需要设置哪些参数?

时钟使能寄存器地址：RCC\_AHB2 和 RCC\_APB1 用于使能GPIO和UART的时钟。

端口基地址：gpio\_ptr 和 uart\_ptr 分别指向 GPIOA 端口和 UART2 端口的基地址。

引脚模式寄存器地址：gpio\_mode 用于配置引脚模式。

GPIO复用功能寄存器地址：gpio\_afrl 用于选择引脚的端口复用功能。

波特率寄存器地址：uart\_brr 用于配置波特率。

UART控制寄存器基地址：uart\_cr1、uart\_cr2 和 uart\_cr3 用于配置 UART 控制寄存器。

UART发送数据寄存器地址：uart\_tdr 用于发送数据。

USARTDIV 变量：用于存储波特率寄存器应赋的值。

//设置的参数如下

//uart寄存器相关地址

**volatile** uint32\_t\* RCC\_AHB2; //GPIO的A口时钟使能寄存器地址

**volatile** uint32\_t\* RCC\_APB1; //UART的2口时钟使能寄存器地址

**volatile** uint32\_t\* gpio\_ptr; //GPIO的A口基地址

**volatile** uint32\_t\* uart\_ptr; //uart2端口的基地址

**volatile** uint32\_t\* gpio\_mode; //引脚模式寄存器地址=口基地址

**volatile** uint32\_t\* gpio\_afrl; //GPIO复用功能低位寄存器

**volatile** uint32\_t\* uart\_brr; //UART波特率寄存器地址

**volatile** uint32\_t\* uart\_isr; // UART中断和状态寄存器基地址

**volatile** uint32\_t\* uart\_cr1; //UART控制寄存器1基地址

**volatile** uint32\_t\* uart\_cr2; // UART控制寄存器2基地址

**volatile** uint32\_t\* uart\_cr3; // UART控制寄存器3基地址

**volatile** uint32\_t\* uart\_tdr; // UART发送数据寄存器

uint16\_t usartdiv; //BRR寄存器应赋的值

//变量赋值

RCC\_APB1=0x40021058UL; //UART时钟使能寄存器地址

RCC\_AHB2=0x4002104CUL; //GPIO的A口时钟使能寄存器地址

gpio\_ptr=0x48000000UL; //GPIOA端口的基地址

uart\_ptr=0x40004400UL; //UART2端口的基地址

gpio\_mode=0x48000000UL; //引脚模式寄存器地址=口基地址

gpio\_afrl=0x48000020UL; // GPIO复用功能低位寄存器

uart\_cr1=0x40004400UL; //UART控制寄存器1基地址

uart\_brr=0x4000440CUL; // UART波特率寄存器地址

uart\_isr=0x4000441CUL; // UART中断和状态寄存器基地址

uart\_tdr=0x40004428UL; //UART发送数据寄存器

uart\_cr2=0x40004404UL; // UART控制寄存器2基地址

uart\_cr3=0x40004408UL; //UART控制寄存器3基地址

//使能GPIOA和UART2的时钟

\*RCC\_APB1|=(0x1UL<<17U); //UART2时钟使能

\*RCC\_AHB2 |=(0x1UL<<0U); //GPIOA时钟使能

//将GPIO端口设置为复用功能

//首先将D7、D6、D5、D4清零

\*gpio\_mode &= ~((0x3UL<<4U)|(0x3UL<<6U));

//然后将D7、D6、D5、D4设为1010，设置PTA2、PTA3为复用功能串行功能。

\*gpio\_mode |=((0x2UL<<4U)|(0x2UL<<6U));

//选择引脚的端口复用功能

//首先将D15~D8清零

\*gpio\_afrl &= ~((0xFUL<<8U)|(0xFUL<<12U));

//然后将D15~D8设置为01110111，分别将PTA3、PTA2引脚设置为USART2\_RX、USART2\_TX

\*gpio\_afrl=(((0x1UL<<8U)|(0x2UL<<8U)|(0x4UL<<8U))|((0x1UL<<12U)

|(0x2UL<<12U)|(0x4UL<<12U)));

//暂时禁用UART功能，控制寄存器1的第0位对应的是UE—USART使能位。

//此位清零后，USART预分频器和输出将立即停止，并丢弃所有当前操作。

\*uart\_cr1 &= ~(0x1UL);

//暂时关闭串口发送与接收功能，控制寄存器1的发送器使能位（D3）、接收器使能位（D2）

\*uart\_cr1 &= ~((0x1UL<<3U)|(0x1UL<<2U));

//配置波特率

**if**(\***uart\_cr1&**(0x1UL<<15) == (0x1UL<<15))

usartdiv = (uint16\_t)((SystemCoreClock/115200)\*2);

**else**

usartdiv = (uint16\_t)((SystemCoreClock/115200));

\*uart\_brr = usartdiv;

//初始化控制寄存器和中断状态寄存器、清标志位

//关中断

\*uart\_isr = 0x0UL;

//将控制寄存器2的两个使能位清零。D14—LIN模式使能位、D11—时钟使能位

\*uart\_cr2 &= ~((0x1UL<<14U)|(0x1UL<<11U));

//将控制寄存器3的三个使能位清零。D5 (SCEN) —smartcard模式使能位、

//D3 (HDSEL) —半双工选择位、D1 (IREN) —IrDA 模式使能位

\*uart\_cr3 &= ~((0x1UL<<5U) | (0x1UL<<3U) |(0x1UL<<1U));

//启动串口发送与接收功能

\*uart\_cr1 |= ((0x1UL<<3U)|(0x1UL<<2U));

//开启UART功能

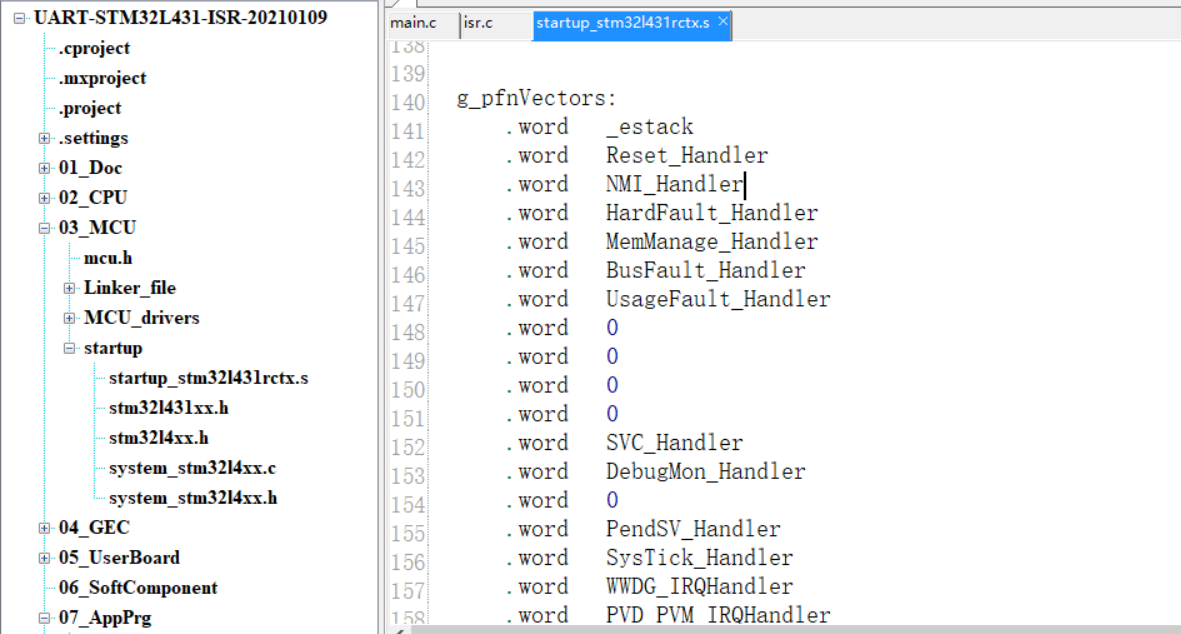
\*uart\_cr1 |= (0x1UL<<0U);

二、假设速度为115200，系统时钟为72MHz，波特率寄存器BRR中的值应该是多少？

USARTCR1中第15位“过采样”模式为0,（系数为8）：USARTDIV = 2\*72MHz / 115200 = 1250

USARTCR1中第15位“过采样”模式为1,（系数为16）：USARTDIV = 72MHz / 115200 = 625

## **三、中断向量表在哪个文件中？表中有多少项？给出部分截图。**

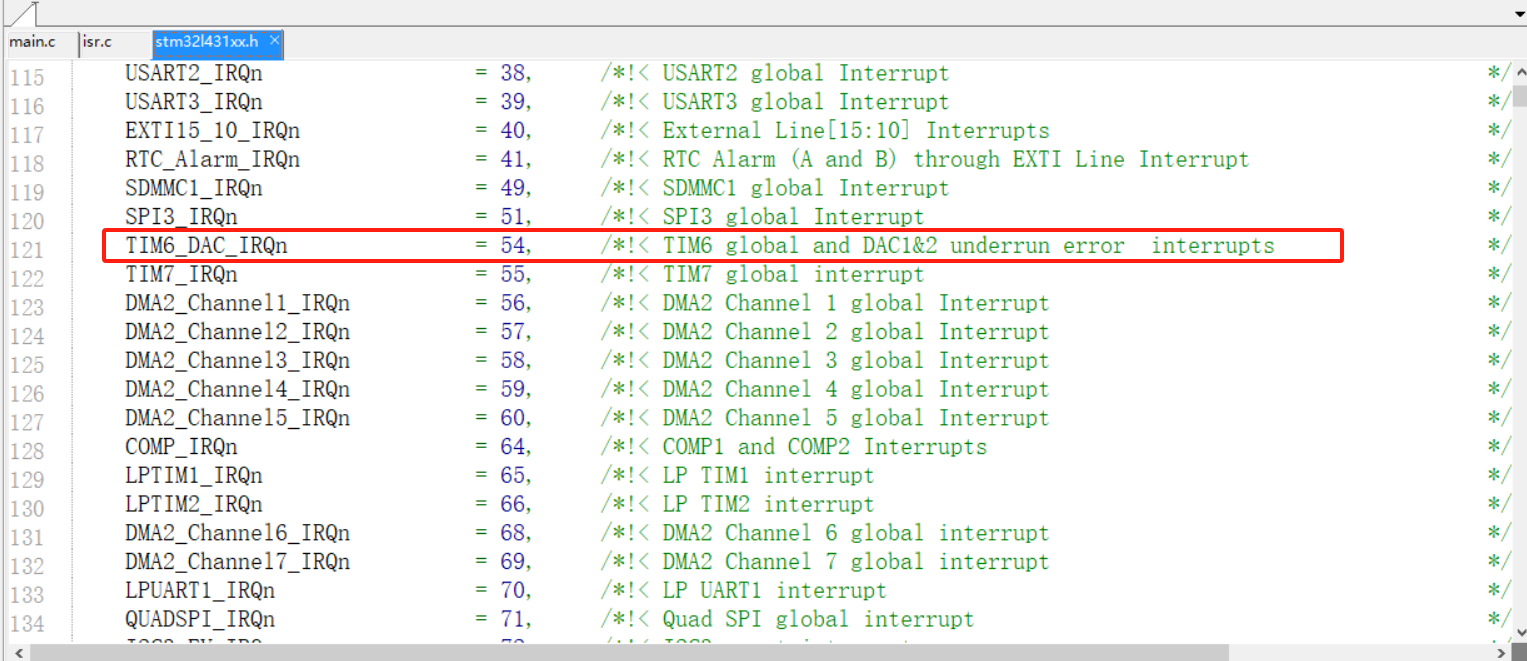


在

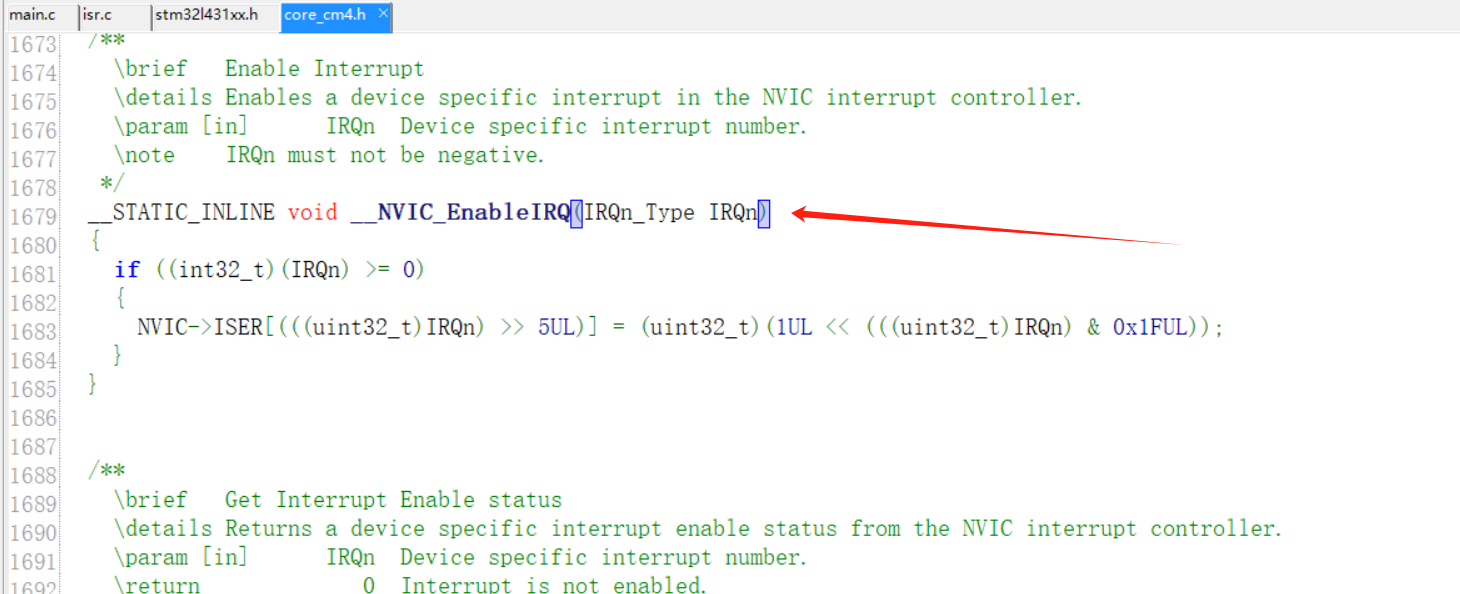
AHL-MCU6-V2.0-20240118\AHL-MCU6-V2.0-20240118\04-Software\CH06\UART-STM32L431-ADDR-20210103

表中共有99项

## **四、以下是中断源使能函数，假设中断源为TIM6，将函数实例化（写出各项具体数值）。**



找到TIM6中断号54



当传入的参数IRQn为54时，首先会执行以下步骤：

if ((int32\_t)(IRQn) >= 0)判断IRQn是否大于等于0，因为IRQn是一个枚举类型，实际上是一个整数值，因此这一步是为了确保传入的中断号是有效的。

通过位运算确定需要设置的寄存器和位。这里涉及到三个部分：

((uint32\_t)IRQn) >> 5UL 表示将中断号右移5位，这是因为每个寄存器可以管理32个中断，所以需要用中断号除以32得到对应的寄存器号。

(1UL << (((uint32\_t)IRQn) & 0x1FUL)) 表示将1左移中断号对32取余的位数，这是因为每个寄存器中的位号对应一个中断号，需要用中断号对32取余得到在寄存器中的位置。

最终结果为将1左移中断号对32取余的位数，得到需要设置的中断位。

NVIC->ISER[1] = 1 << 22将得到的中断位写入到对应的NVIC->ISER寄存器中，使得对应的中断被使能。

综上所述，当传入的参数IRQn为54时，经过上述步骤处理后，最终的操作是将第22位设置为1，从而使得中断号为54的中断被使能。

五、假设将UART\_2和TIM6交换其在中断向量表中的位置和IRQ号，UART\_2可以正常中断吗？

不能，将UART\_2和TIM6在中断向量表中的位置和IRQ号进行交换，那么在代码中相应的中断处理程序可能会被错误地调用，因为中断向量表中的位置和IRQ号是与特定的外设和中断服务函数相关联的。