

STM32通过中断方式实现USART1通信

 blog.csdn.net/u013082827/article/details/77726520

1.前言

这个例程花了我好几天的时间，涉及到的内容比较多，走了很多弯路，一定要总结一下！

首先说明，我的开发板不是比较流行的正点原子家的，而是普中的STM32F1，原子家的开发板太贵了。到目前为止，我还没有发现我的开发板出现硬件上的问题，就是客服很不给力，关于仿真器配置的一个小问题就解决不了。好在网上资料比较丰富，慢慢也能解决自己遇到的一些问题。

思路和程序网上大家列举的都一样。在上网查资料的过程中发现，ARM开发板就是厉害，我想找什么问题，不管大家用什么型号的开发板，程序几乎一点都没有变化，问题讨论起来方便多了！

2.程序摘要

分为4部分：GPIO配置、USART参数配置、NVIC总中断配置、中断函数；
在主函数中初始化以上配置，直接进入while(1)就行了。

首先配置管脚GPIOA：GPIO_Init

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
```

```
GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
```

```
GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
```

```
.....
```

注意：这里使用了管脚复用功能,但是并不需要下面这句话：

```
"RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);"
```

原因：STM32F103：什么时候需要复用IO（AFIO）？

- 串口通信USART1参数配置：USART_Init

使用USART_Init函数，配置包括波特率、停止位、数据位、校验位等常见的串口通信的参数；

```
USART_InitStructure.USART_Mode=USART_Mode_Rx|USART_Mode_Tx;
```

```
USART_Cmd(USART1, ENABLE);
```

```
USART_ITConfig(USART1, USART_IT_RXNE ,ENABLE);
```

```
USART_ClearITPendingBit(USART1, USART_IT_RXNE);
```

- 5

这里要弄清楚，可以操作USART1中断标志位（也就是输入参数是USART的中断类型）的库函数有三个：

USART_ITConfig：使能/失能中断

USART_GetITStatus：读取中断状态

USART_ClearITPendingBit：清除中断标志位

其中，清除中断标志位用： USART_ClearITPendingBit；

总中断设置：NVIC_Init

```
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
```

```
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
```

- 1
- 2
- 3
- 4
- 5

• 中断函数

这里要实现的功能是：

PC向单片机发送字节数据，单片机接收到数据后+2，并返回到PC端；

```

void USART1_IRQHandler ()
{
    u8 k;

    if(USART_GetITStatus(USART1,USART_IT_RXNE)== SET)
    {

        k=USART_ReceiveData(USART1);
        k=k+2;

        USART_SendData(USART1,k);

        while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    }
}

```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16

第一个问题：

进入中断之后，判断中断发生的类型时，可以使用USART_GetFlagStatus吗？

回答：

可以。把：if(USART_GetITStatus(USART1,USART_IT_RXNE) == SET)

替换成：if(USART_GetFlagStatus(USART1,USART_FLAG_RXNE))，

结果证明可以。

事实上，USART_IT_RXNE 只是中断类型名，要接ENABLE/DISABLE的，不是中断标志位，我猜测 USART_GetITStatus 函数其实判断的也是发送完成的flag：

USART_FLAG_TXE。

但是，明明已经进入了中断，查询中断状态来判断不是更加顺其自然吗！

第二个问题：

执行发送语句、等待发送完成的时候，可以用 USART_GetITStatus 函数判断是否发送完成吗？

回答：

不可以。因为发送中断根本没打开，中断状态也是无效的。

3.遇到的问题

软件问题

写好程序之后，通过仿真器调试发现，总是不能进入接收字节的串口中断；

当时有一种病急乱投医的感觉，在网上查了很多资料，有说“不该打开发送中断”的，有说“在中断函数中没有清楚中断标志位”的，也有说“管脚复用没有打开”的.....

再说明一下：本次管脚复用不需要用那句话打开的，上面链接讲的很详细了。

等我调试好，把上面的说法进行逐个验证，最后不需要的都没加上，最终得到了上面所示的最简洁的，并且实现预期效果的通信程序。

硬件问题

其实，最开始导致程序不能进入中断的原因，是硬件的原因。

232串口通信，根据我的经验，找了一根USB转9针串口，连接上发现不进入中断.....

后来各种调试没结果，实在没办法了，看视频才知道，这个开发板的232有两种模式，USART1使用的是：

PC→USB转microUSB数据线→CH340芯片→USART1接口→MCU



而常见的却是这种模式：

PC→USB转9针串口数据线→max232芯片→USART1串口→MCU



我误以为USART1是后者，调了很久没结果.....

4.总结

这个例程已经花了好几天时间了.....

今天上午实验室的师兄师姐们进行招聘网上答题，给出一个问题，在很短的时间内提出解决方法并编出程序。感觉这种考验真的能体现出水平，不妨偶尔把平时遇到的问题当做考试来解决，换一换思路和环境，毕竟也不是做研究，没必要按部就班像上课一样学习。

遇到一个问题就像面临一张空白的试卷，我按照流程不慌不忙的花个几天解决问题，就像把试卷当做练习题来做；

但是总做练习题，水平提高的并不是很快，不如偶尔认真起来，把这一张试卷当做考试，不是为了应试，而是为了在考试中发现自己的薄弱部分，毕竟考试完毕也可以好好总结。

从下一个例程开始！