一、串口概述

1. 定义

串口通信是一种设备间非常常用的**串行**，以比特位的形式发送或接收数据，电子工程师经常使用这种方式来调试数据。

2. 开发板硬件用于串口跟 PC 相连的时候有以下注意事项：

A. 使用到 usb 转串口，所以得安装驱动

B. 跳线帽要进行短接，**参考 01_串口 1 硬件连接.BMP**

二、程序设计

1. 设置引脚功能复用

```
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA,ENABLE);                    //使能 GPIOA 时钟
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);                   //使能 USART1 时钟

//串口 1 对应引脚复用映射
GPIO_PinAFConfig(GPIOA,GPIO_PinSource9,GPIO_AF_USART1);                 //GPIOA9 复用为 USART1
GPIO_PinAFConfig(GPIOA,GPIO_PinSource10,GPIO_AF_USART1);                //GPIOA10 复用为 USART1

//USART1 端口配置
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9 | GPIO_Pin_10;                 //GPIOA9 与 GPIOA10
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;                            //复用功能
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;                       //速度 50MHz
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;                              //推挽复用输出
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;                            //上拉
GPIO_Init(GPIOA,&GPIO_InitStructure);                                       //初始化 PA9，PA10
```

2. 串口参数配置

```
//USART1 初始化设置
USART_InitStructure.USART_BaudRate = baud;                                  //波特率设置
USART_InitStructure.USART_WordLength = USART_WordLength_8b;                 //字长为 8 位数据格式
USART_InitStructure.USART_StopBits = USART_StopBits_1;                      //一个停止位
USART_InitStructure.USART_Parity = USART_Parity_No;                         //无奇偶校验位
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//无硬件数据流控制
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;             //收发模式
USART_Init(USART1, &USART_InitStructure);                                       //初始化串口 1
USART_Cmd(USART1, ENABLE);                                                      //使能串口 1

//USART_OverSampling8Cmd(USART1,ENABLE);//在超高速的波特率才打开，如 5MHz 频率。正常很少有设备达到这么高的速度
```

3. 串口中断配置

```
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);                          //开启相关中断

//Usart1 NVIC 配置
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;                       //串口 1 中断通道
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=3;                 //抢占优先级 3
NVIC_InitStructure.NVIC_IRQChannelSubPriority =3;                     //子优先级 3
```

```
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;                    //IRQ 通道使能
    NVIC_Init(&NVIC_InitStructure);                                    //根据指定的参数初始化 VIC 寄存器
```

4. 发送函数的编写

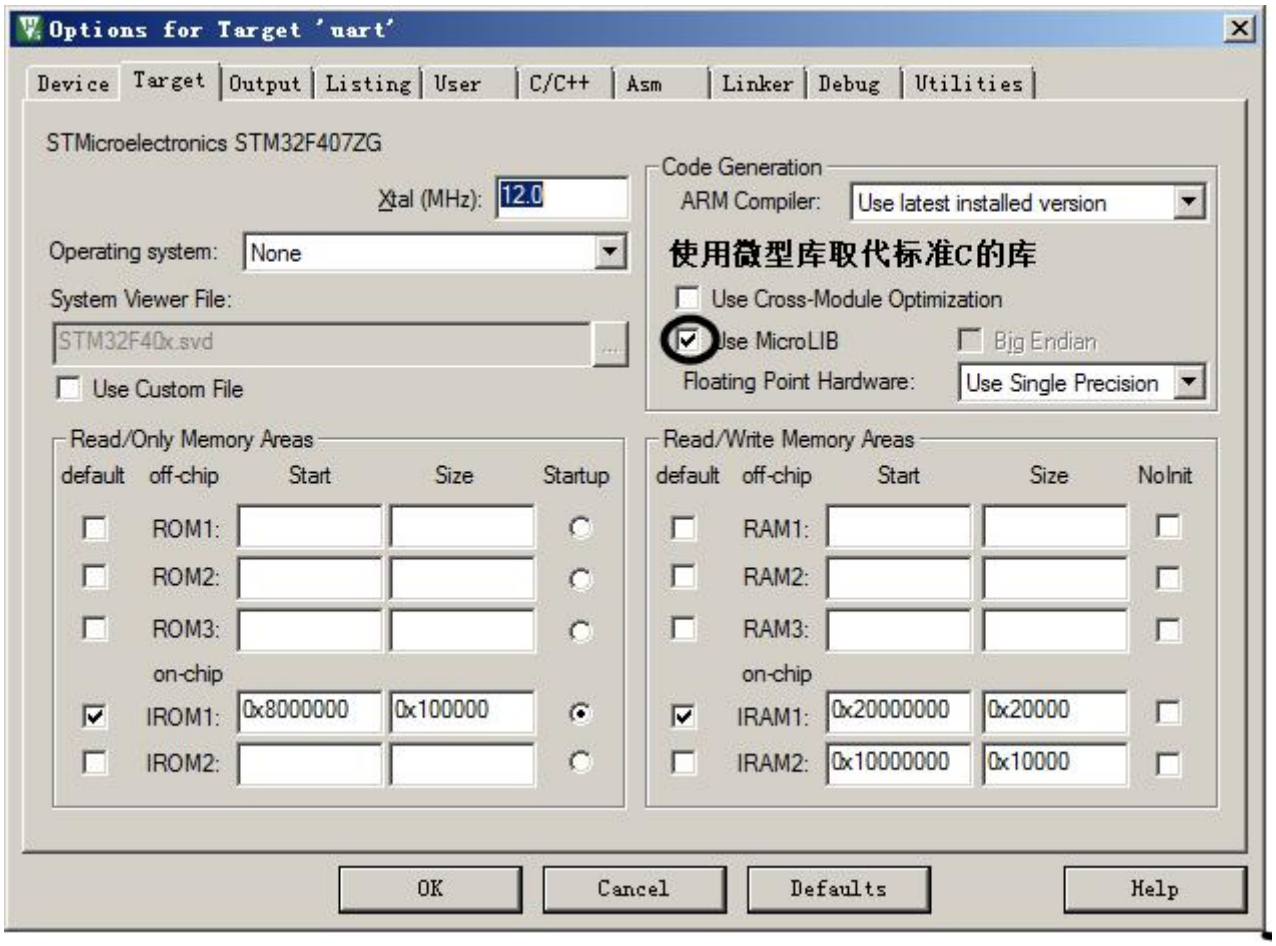
```
void usart1_send_bytes(uint8_t *pbuf,uint32_t len)
{
    while(len--)
    {
        USART_SendData(USART1,*pbuf++);
        while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    }
}

void usart1_send_str(char *pbuf)
{
    while(pbuf && *pbuf)
    {
        USART_SendData(USART1,*pbuf++);
        while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    }
}
```

有些同学发送数据是乱码，请检查 PLL 相关的配置是否正常！

三、重定向 printf 函数

参考 03_自定义 printf 函数.bmp

1.在 Keil 选项中勾选支持“Micro LIB”，如下图。



2.因为 printf 的打印输出最后由 fputc 实现，所以在 main.c 当中，重写 fputc 函数。

```
void printf(char ....)
{
    //输出单个字符
    fputc

    ......

}
```

```
//重定义fputc函数
int fputc(int ch, FILE *f)
{
      USART_SendData(USART1,ch);
      while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);

      return ch;
}



//printf打印输出
printf("hello gecedu %d\r\n",year);
```

```
//重定义 fputc
int fputc(int ch,FILE *f)
{
    USART_SendData(USART1,ch);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);

    return ch;
}
```

练习 1

使用 PC 通过串口发送数据给开发板实现灯的控制，要求如下：

接收到数据 0x00，则 LED0 点亮；接收到数据 0xF0，则 LED0 熄灭！

接收到数据 0x01，则 LED1 点亮；接收到数据 0xF1，则 LED1 熄灭！

接收到数据 0x02，则 LED2 点亮；接收到数据 0xF2，则 LED2 熄灭！

接收到数据 0x03，则 LED3 点亮；接收到数据 0xF3，则 LED3 熄灭！