```
//main.c 程序
//<-只发送一个的可以看一下你串口 c 文件的 TXE 是否在 for 函数循环里面，不然就只会发送高位的数据
//还是不能解决
#include "stm32f10x.h"
#include "bsp_led.h"
#include "bsp_usart.h"

int main(void)
{
	uint8_t a[10] = {1,2,3,4,5,6,7,8,9,10};
	USART_Config();
//	Usart_SendByte( DEBUG_USARTx,	'A');
	Usart_SendHalfWord( DEBUG_USARTx,	0xff56);
	//Usart_SendArray( DEBUG_USARTx,	a , 10 );
	 //Usart_SendStr(DEBUG_USARTx, "你好");


	while(1)
	{
	}
}


////////////////////////////////////////////////////////////////////////////////////////////////////
//bsp_usart.c 程序
#include "bsp_usart.h"

static void NVIC_Configuration(void)
{
  NVIC_InitTypeDef NVIC_InitStructure;

  /* 嵌套向量中断控制器组选择 */
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);

  /* 配置 USART 为中断源 */
  NVIC_InitStructure.NVIC_IRQChannel = DEBUG_USART_IRQ;
  /* 抢断优先级*/
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
  /* 子优先级 */
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  /* 使能中断 */
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  /* 初始化配置 NVIC */
```

```
    NVIC_Init(&NVIC_InitStructure);
}


void USART_Config(void)
{
      GPIO_InitTypeDef GPIO_InitStructure;
      USART_InitTypeDef USART_InitStructure;

      // 打开串口 GPIO 的时钟
      DEBUG_USART_GPIO_APBxClkCmd(DEBUG_USART_GPIO_CLK, ENABLE);

      // 打开串口外设的时钟
      DEBUG_USART_APBxClkCmd(DEBUG_USART_CLK, ENABLE);

      // 将 USART Tx 的 GPIO 配置为推挽复用模式
      GPIO_InitStructure.GPIO_Pin = DEBUG_USART_TX_GPIO_PIN;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(DEBUG_USART_TX_GPIO_PORT, &GPIO_InitStructure);

    // 将 USART Rx 的 GPIO 配置为浮空输入模式
      GPIO_InitStructure.GPIO_Pin = DEBUG_USART_RX_GPIO_PIN;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
      GPIO_Init(DEBUG_USART_RX_GPIO_PORT, &GPIO_InitStructure);

      // 配置串口的工作参数
      // 配置波特率
      USART_InitStructure.USART_BaudRate = DEBUG_USART_BAUDRATE;
      // 配置 针数据字长
      USART_InitStructure.USART_WordLength = USART_WordLength_8b;
      // 配置停止位
      USART_InitStructure.USART_StopBits = USART_StopBits_1;
      // 配置校验位
      USART_InitStructure.USART_Parity = USART_Parity_No ;
      // 配置硬件流控制
      USART_InitStructure.USART_HardwareFlowControl                                    =
USART_HardwareFlowControl_None;
      // 配置工作模式，收发一起
      USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
      // 完成串口的初始化配置
      USART_Init(DEBUG_USARTx, &USART_InitStructure);

      // 串口中断优先级配置
```

```
    NVIC_Configuration();

    // 使能串口接收中断
    USART_ITConfig(DEBUG_USARTx, USART_IT_RXNE, ENABLE);

    // 使能串口
    USART_Cmd(DEBUG_USARTx, ENABLE);
}


void Usart_SendByte(USART_TypeDef* pUSARTx, uint8_t data)
{
    USART_SendData(pUSARTx, data);
    while( USART_GetFlagStatus(pUSARTx,  USART_FLAG_RXNE) == RESET);

}
/*发送半字数据，这个也是这次我要问的问题，为什么只有高位字节能发送*/
void Usart_SendHalfWord(USART_TypeDef* pUSARTx, uint16_t data)
{

    uint8_t temp_h,temp_l;
    temp_h = (data & 0xff00) >> 8;
    temp_l = data & 0xff;

    USART_SendData(pUSARTx, temp_h);
    while( USART_GetFlagStatus(pUSARTx,  USART_FLAG_RXNE) == RESET);

    USART_SendData(pUSARTx, temp_l);
    while( USART_GetFlagStatus(pUSARTx,  USART_FLAG_RXNE) == RESET);
}

/*发送 8 位数据的数组*/
void Usart_SendArray(USART_TypeDef* pUSARTx, uint8_t *array , uint8_t num)
{
    uint8_t i;
    for (i = 0;i < num;i++)
    {
        Usart_SendByte( pUSARTx,  array[i]);
    while( USART_GetFlagStatus(pUSARTx,  USART_FLAG_TXE) == RESET);

    }
    while( USART_GetFlagStatus(pUSARTx,  USART_FLAG_TC) == RESET);

}//这个数组也是只能发送一个数据，不知道咋回事
```

```
/*发送字符串*/
//乱码的兄弟，去 stm32f10x.h 头文件搜 HSE_VALUE，默认是 8000000 即 8MHz）比如我单片机外部晶振为 12MHz,那这个值就改为 12000000。乱码是系统波特计算错导致，用此法
void Usart_SendStr(USART_TypeDef* pUSARTx, uint8_t *str)
{
	uint8_t i = 0;
	do
	{
		Usart_SendByte( pUSARTx,  *(str + i));
		i++;

	}while(*(str+i) != '0');
	while( USART_GetFlagStatus(pUSARTx,  USART_FLAG_TC) == RESET);

}//确实是乱码


////////////////////////////////////////////////////////////////////
//bsp_usart.h 头文件程序
#ifndef __BSP_USART_H
#define __BSP_USART_H

#include "stm32f10x.h"

#define   DEBUG_USARTx                    USART1
#define   DEBUG_USART_CLK                 RCC_APB2Periph_USART1
#define   DEBUG_USART_APBxClkCmd          RCC_APB2PeriphClockCmd
#define   DEBUG_USART_BAUDRATE            115200

// USART GPIO 引脚宏定义
#define   DEBUG_USART_GPIO_CLK            (RCC_APB2Periph_GPIOA)
#define   DEBUG_USART_GPIO_APBxClkCmd     RCC_APB2PeriphClockCmd

#define   DEBUG_USART_TX_GPIO_PORT        GPIOA
#define   DEBUG_USART_TX_GPIO_PIN         GPIO_Pin_9
#define   DEBUG_USART_RX_GPIO_PORT        GPIOA
#define   DEBUG_USART_RX_GPIO_PIN         GPIO_Pin_10

#define   DEBUG_USART_IRQ                 USART1_IRQn
#define   DEBUG_USART_IRQHandler          USART1_IRQHandler

void USART_Config(void);
void Usart_SendByte(USART_TypeDef* pUSARTx, uint8_t data);
```

```
void Usart_SendHalfWord(USART_TypeDef* pUSARTx, uint16_t data);
void Usart_SendArray(USART_TypeDef* pUSARTx, uint8_t *array , uint8_t num);
void Usart_SendStr(USART_TypeDef* pUSARTx, uint8_t *str);

#endif /* __BSP_USART_H */
```