三种方式：查询，中断，DMA

通用同步异步收发器（USART）提供了一种灵活的方法来与使用工业标准 NR 异步串行数据格式的外部设备之间进行全双工数据交换。 USART 利用分数波特率发生器提供宽范围的波特率选择。
它支持同步单向通信和半双工单线通信。它也支持 LIN(局部互连网)，智能卡协议和 IrDA(红外数据组织)SIR ENDEC 规范，以及调制解调器(CTS/RTS)操作。它还允许多处理器通信。用于多缓冲器配置的 DMA 方式，可以实现高速数据通信。

主要特性：
全双工的，异步通信
NR 标准格式
分数波特率发生器系统
-发送和接收共用的可编程波特率，最高到 4.5Mbits/s
可编程数据字长度（8 位或 9 位）
可配置的停止位 -支持 1 或 2 个停止位
LIN 主发送同步断开符的能力以及 LIN 从检测断开符的能力
- 当 USART 硬件配置成 LIN 时，生成 13 位断开符；检测 10/11 位断开符
发送方为同步传输提供时钟
IRDA SIR 编码器解码器
- 在正常模式下支持 3/16 位的持续时间
智能卡模拟功能
- 智能卡接口支持 ISO7816 -3 标准里定义的异步协议智能卡
- 智能卡用到的 0.5 和 1.5 个停止位
单线半双工通信
使用 DMA 的可配置的多缓冲器通信
- 在保留的 SRAM 里利用集中式 DMA 缓冲接收/发送字节
单独的发送器和接收器使能位
检测标志
- 接收缓冲器满
- 发送缓冲器空
- 传输结束标志
校验控制
- 发送校验位
- 对接收数据进行校验
四个错误检测标志
- 溢出错误
- 噪音错误
- 帧错误
- 校验错误
10 个带标志的中断源
- CTS 改变

- LIN 断开符检测
- 发送数据寄存器
- 发送完成
- 接收数据寄存器
- 检测到总线为空
- 溢出错误
- 帧错误
- 噪音错误
- 校验错误

多处理器通信 - - 如果地址不匹配，则进入静默模式
从静默模式中唤醒（通过空闲总线检测或地址标志检测）
两种唤醒接收器的方式

- 地址位（MSB）
- 空闲总线

STM32 的串口配置 也挺方便的

首先是配置 UART 的 GPIO 口

```
/*******************************************************************************
***********
* Function Name   : UART1_GPIO_Configuration
* Description        : Configures the uart1 GPIO ports.
* Input                    : None
* Output                 : None
* Return                 : None
*******************************************************************************
**********/
void UART1_GPIO_Configuration(void)
{
GPIO_InitTypeDef GPIO_InitStructure;
// Configure USART1_Tx as alternate function push-pull
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);

// Configure USART1_Rx as input floating
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

}

然后是配置串口参数

```
/* 如果使用查询的方式发送和接收数据 则不需要使用串口的中断
   如果需要使用中断的方式发送和接收数据 则需要使能串口中断
    函数原形 void USART_ITConfig(USART_TypeDef* USARTx, u16 USART_IT,
FunctionalState NewState)
    功能描述 使能或者失能指定的 USART 中断

        USART_IT                    描述
        USART_IT_PE       奇偶错误中断
        USART_IT_TXE    发送中断
        USART_IT_TC       传输完成中断
        USART_IT_RXNE 接收中断
        USART_IT_IDLE    空闲总线中断
        USART_IT_LBD    LIN 中断检测中断
        USART_IT_CTS    CTS 中断
        USART_IT_ERR    错误中断

*/


/*************************************************************************
***********
* Function Name   : UART1_Configuration
* Description       : Configures the uart1
* Input                   : None
* Output                : None
* Return                : None
*************************************************************************
**********/
void UART1_Configuration(void)
{

   USART_InitTypeDef USART_InitStructure;
   /* USART1 configured as follow:
       - BaudRate = 9600 baud
       - Word Length = 8 Bits
       - One Stop Bit
       - No parity
       - Hardware flow control disabled (RTS and CTS signals)
       - Receive and transmit enabled
```

```
   */
   USART_InitStructure.USART_BaudRate = 9600;
   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
   USART_InitStructure.USART_StopBits = USART_StopBits_1;
   USART_InitStructure.USART_Parity = USART_Parity_No ;
   USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
   USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

   /* Configure the USART1*/
   USART_Init(USART1, &USART_InitStructure);

   /* Enable USART1 Receive and Transmit interrupts */
   USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);


   /* Enable the USART1 */
   USART_Cmd(USART1, ENABLE);
}
```

发送一个字符

```
/*******************************************************************************
***********
* Function Name   : Uart1_PutChar
* Description        : printf a char to the uart.
* Input                    : None
* Output                 : None
* Return                 : None
*******************************************************************************
**********/
u8 Uart1_PutChar(u8 ch)
{
/* Write a character to the USART */
USART_SendData(USART1, (u8) ch);
while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
{
}
return ch;
}
```

发送一个字符串

```
/*******************************************************************************
```

```
***********
* Function Name   : Uart1_PutString
* Description        : print a string to the uart1
* Input                    : buf 为发送数据的地址 , len 为发送字符的个
数
* Output                  : None
* Return                  : None
*******************************************************************
**********/
void Uart1_PutString(u8* buf , u8 len)
{
for(u8 i=0;i<len;i++)
{
       Uart1_PutChar(*buf++);
}
}
```

如果 UART 使用中断发送数据 则需要修改 stm32f10x_it.c 中的串口中断函数 并且需要修改 void NVIC_Configuration(void)函数

在中断里面的处理 原则上是需要简短和高效 下面的流程是 如果接收到 255 个字符或者接收到回车符 则关闭中断 并且把标志位 UartHaveData 置 1

```
/*******************************************************************
***********
* Function Name   : USART1_IRQHandler
* Description        : This function handles USART1 global interrupt
request.
* Input                    : None
* Output                  : None
* Return                  : None
*******************************************************************
**********/
void USART1_IRQHandler(void)
{
   if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
   {
/* Read one byte from the receive data register */
RxBuffer[ RxCounter ] = USART_ReceiveData(USART1);
if( RxCounter == 0xfe || '\r' == RxBuffer[ RxCounter ] )
{
   /* Disable the USART1 Receive interrupt */
   USART_ITConfig(USART1, USART_IT_RXNE, DISABLE);
```

```
   RxBuffer[ RxCounter ] = '\0';
   UartHaveData = 1;
}

RxCounter++;
   }
}
```

修改 NVIC_Configuration 函数

```
/*******************************************************************************
***********
* Function Name   : NVIC_Configuration
* Description        : Configures NVIC and Vector Table base location.
* Input                    : None
* Output                  : None
* Return                  : None
*******************************************************************************
**********/
void NVIC_Configuration(void)
{
   NVIC_InitTypeDef NVIC_InitStructure;

#ifdef   VECT_TAB_RAM
/* Set the Vector Table base location at 0x20000000 */
NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
#else   /* VECT_TAB_FLASH   */
/* Set the Vector Table base location at 0x08000000 */
NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
#endif

   /* Configure the NVIC Preemption Priority Bits */
   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

   /* Enable the USART1 Interrupt */
   NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQChannel;
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
   NVIC_Init(&NVIC_InitStructure);

}
```

至此 串口就可以工作起来了 附件中的程序功能是 开机后 从串口中输出 2 行信息 然后就等待接收串口数据 并且把接收到的数据发回到 PC 机上来 附件有 2 个 一个是查询方式的 一个是中断方式的