# STM32培训5 —— USART

# 1. 什么是串口?

串行接口

# 2. 什么是USART

USART: Universal Synchronous/Asynchronous Receiver/Transmitter
通用同步/异步串行接收/发送器

`断句：通用 同步/异步 串行 接收/发送 器`

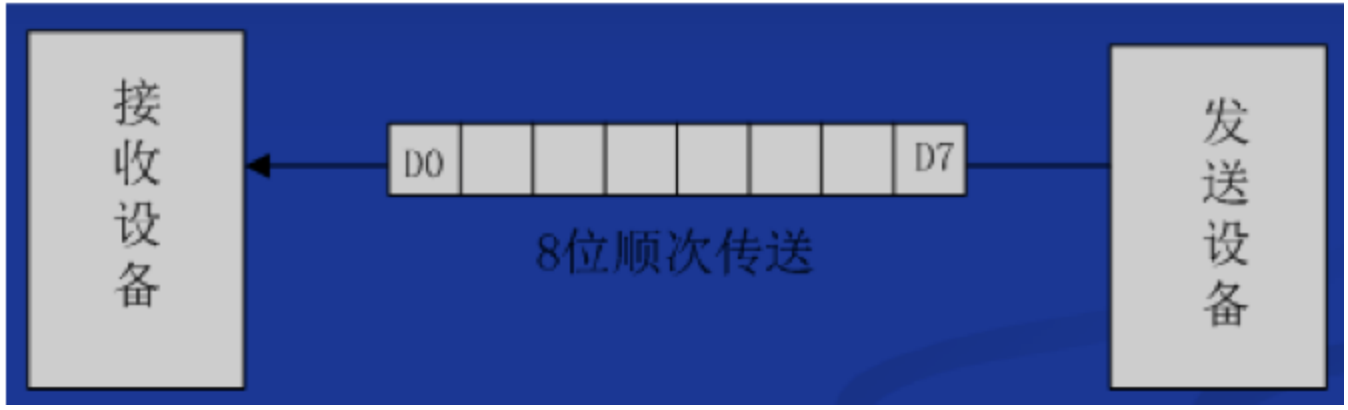
USART是一个全双工通用同步/异步串行收发模块，该接口是一个高度灵活的串行通信设备。

## 与UART区别

`UART：通用异步收发传输器（Universal Asynchronous Receiver/Transmitter）`

比UART多了同步功能，为UART增强版

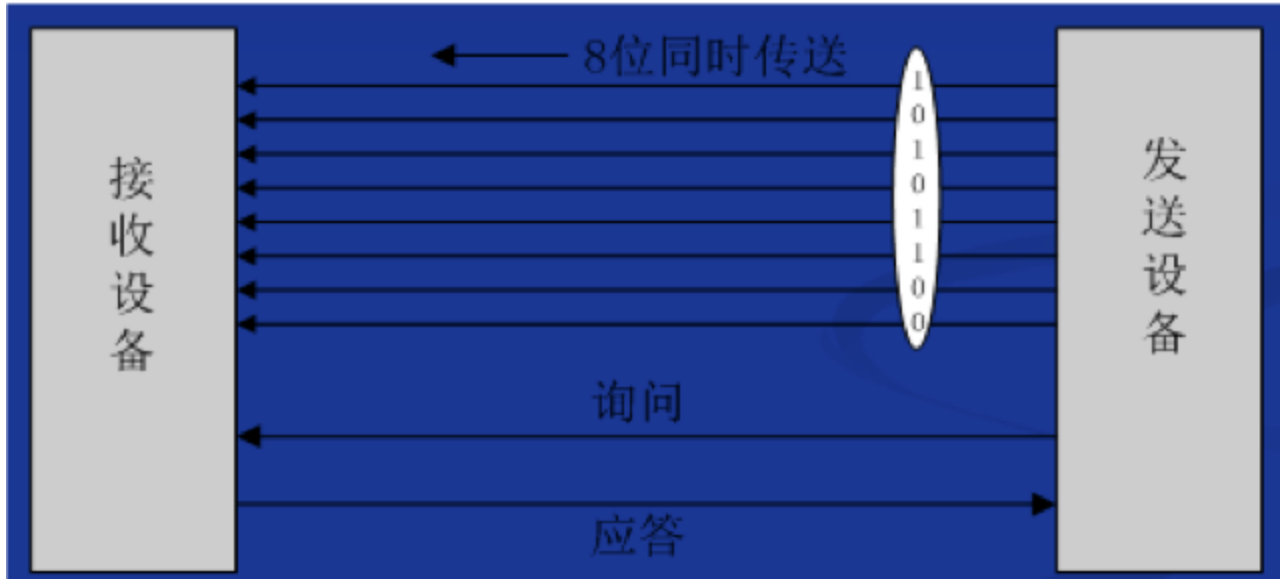
## 3. 串行通讯




串行通信是指使用一条数据线，将数据一位一位地依次传输，每一位数据占据一个固定的时间长度。其只需要少数几条线就可以在系统间交换信息，特别适用于计算机与计算机、计算机与外设之间的远距离通信

- 特点：线少，控制较复杂

# 4. 并行通讯

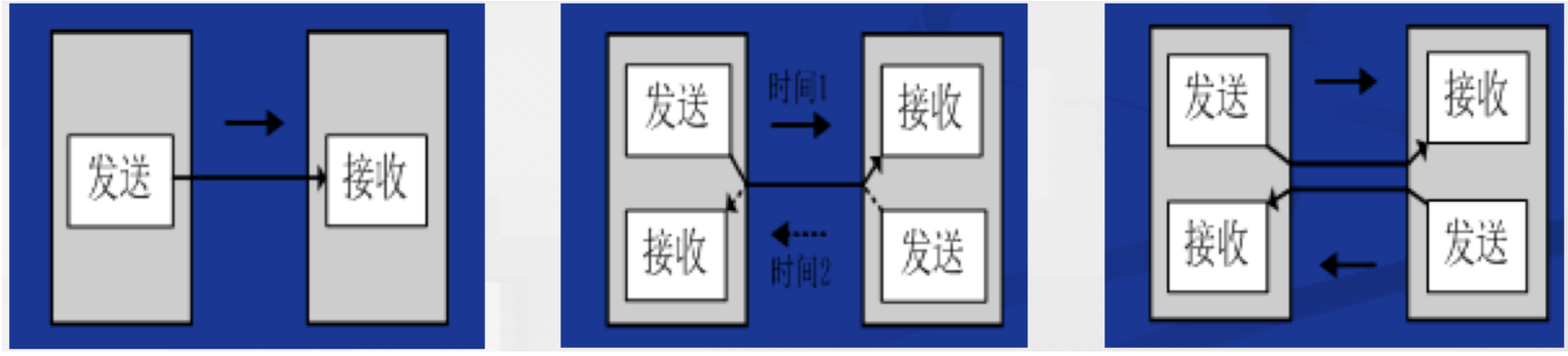



并行通信通常是将数据字节的各位用多条数据线同时进行传送，通常是8位、16位、32位等数据一起传输。

- 特点： 控制简单、传输速度快，接收方的各位同时接收存在困难

# 5. 数据传输方式

- 单工：只允许A向B发送数据或者只允许B向A发送数据
- 半双工：数据可以在一个信号载体的两个方向上传输，但是不能同时传输
- 全双工：数据可以在一个信号载体的两个方向上同时传输



# 6. STM32 USART

| |
|---|
| **USART3_REMAP[1:0]**：USART3的重映像 (USART3 remapping)<br>这些位可由软件置'1'或置'0'，控制USART3的CTS、RTS、CK、TX和RX复用功能在GPIO端口的映像。<br>00: 没有重映像(TX/PB10，RX/PB11，CK/PB12，CTS/PB13，RTS/PB14)；<br>01: 部分映像(TX/PC10，RX/PC11，CK/PC12，CTS/PB13，RTS/PB14)；<br>10: 未用组合；<br>11: 完全映像(TX/PD8，RX/PD9，CK/PD10，CTS/PD11，RTS/PD12)。 |
| **USART2_REMAP**：USART2的重映像 (USART2 remapping)<br>这些位可由软件置'1'或置'0'，控制USART2的CTS、RTS、CK、TX和RX复用功能在GPIO端口的映像。<br>0: 没有重映像(CTS/PA0，RTS/PA1，TX/PA2，RX/PA3，CK/PA4)；<br>1: 重映像(CTS/PD3，RTS/PD4，TX/PD5，RX/PD6，CK/PD7)； |
| **USART1_REMAP**：USART1的重映像 (USART1 remapping)<br>该位可由软件置'1'或置'0'，控制USART1的TX和RX复用功能在GPIO端口的映像。<br>0: 没有重映像(TX/PA9，RX/PA10)；<br>1: 重映像(TX/PB6，RX/PB7)。 |

# 7. STM32 USART配置

## 7.1 使能时钟

需要使能的时钟有: 串口引脚对应GPIO_Port， USART时钟， GPIO复用时钟

```
RCC_APB2PeriphClockCmd( RCC_APB2Periph_USART1 |
                        RCC_APB2Periph_GPIOA |
                        RCC_APB2Periph_AFIO  , ENABLE);
```

## 7.2 初始化GPIO

```
GPIO_InitTypeDef GPIO_InitStructure;
//USART1 TX
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
//复用推挽输出
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);

//USART1 RX
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
//复用开漏输入
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

## 7.3 初始化中断优先级

```
void initNVIC(void) {
    /*  结构声明*/
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Configure the NVIC Preemption Priority Bits */
    /* Configure one bit for preemption priority */
    /* 优先级组 说明了抢占优先级所用的位数，和子优先级所用的位数 */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

## 7.4 初始化USART

```
void USART_Config(USART_TypeDef* USARTx){
    USART_InitTypeDef USART_InitStructure;
    //速率19200bps
    USART_InitStructure.USART_BaudRate = 19200;
    //数据位8位
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    //停止位1位
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    //无校验位
    USART_InitStructure.USART_Parity = USART_Parity_No;
    //无硬件流控
    USART_InitStructure.USART_HardwareFlowControl =
                                USART_HardwareFlowControl_None;
    //收发模式
    USART_InitStructure.USART_Mode = USART_Mode_Rx |
                                     USART_Mode_Tx;
    USART_Init(USARTx, &USART_InitStructure);
    //使能接收中断
    USART_ITConfig(USARTx, USART_IT_RXNE, ENABLE);
    //使能发送缓冲空中断
    USART_ITConfig(USARTx, USART_IT_TXE, ENABLE);
    USART_Cmd(USARTx, ENABLE);
}
```

## 7.5 USART1中断函数

```
void USART1_IRQHandler(void)      //串口1 中断服务程序
{
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        //将读寄存器的数据缓存到接收缓冲区里
        RxBuffer1[RxCount++] = USART_ReceiveData(USART1);
        if(RxBuffer1[RxCount - 2]==0x0d
            &&RxBuffer1[RxCount - 1]==0x0a)
            //判断结束标志是否是0x0d 0x0a
        {
            RxCount=0;
            /* 读取串口结束， 这里可以写一些结束后的操作 */
            RxBuffer1[RxCount - 2] = '\0';
            USART_SendStr(USART1, RxBuffer1);
        }
    }
    //这段是为了避免STM32 USART 第一个字节发不出去的BUG
    if(USART_GetITStatus(USART1, USART_IT_TXE) != RESET)
    {
        //禁止发缓冲器空中断,
        USART_ITConfig(USART1, USART_IT_TXE, DISABLE);
    }
}
```