电子科学与技术专业课

嵌入式系统

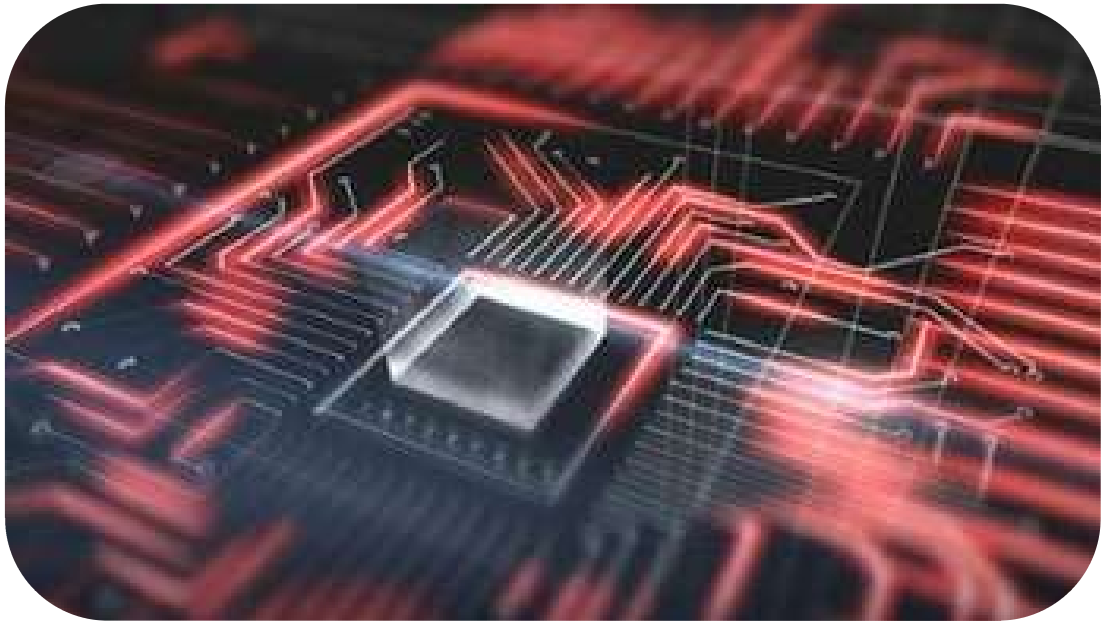
Embedded System

信息学院 光电子系

第5章 通信接口与总线

- 5.1 通信概述
- 5.2 异步串行通信UART
- 5.3 串行外设接口SPI
- 5.4 集成电路总线I^2C

嵌入式系统(EMBEDDED SYSTEM)



第5章 通信接口与总线

❀ **5.1 通信概述**

本节内容

- 通信的定义
- 通信的方式
  - 串行/并行
  - 点对点/Bus总线
  - 同步/异步
  - 半双工/全双工
  - 主从/对等
- 通信信号区分

通信 Communication

- **通信**是传递信息的一种活动，通过各种各样的手段，包括语言、信号、书写、行为等等来交换想法、消息或信息。

—— 维基百科

- **通信**是在点与点之间，人与人之间或设备与设备之间进行信息交换的过程。

—— 牛津词典

通信的本质就是信息的交换！

通信 Communication

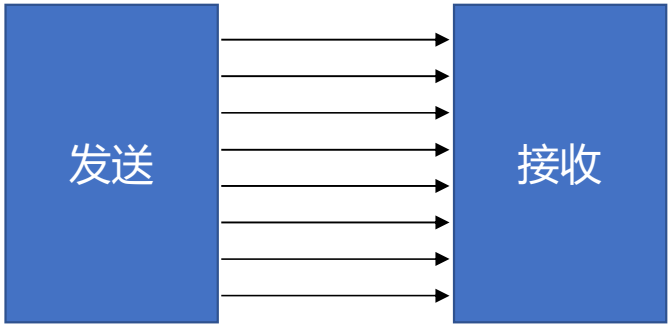
特征

- 串行/并行　Serial/Parallel
- 同步/异步　Synchronous/Asynchronous
- 点对点/总线　Point-to-point/Bus
- 半双工/全双工　Halp-duplex/Full-duplex
- 主从/对等　Master-slave/Equal partners
- 单端/差分　Single-ended/Differential

通信方式

- 串行 Serial
  - 数据一位一位地传送
  - 数据线需求少 – 硬件资源利用率高
  - 带宽小，速度慢
  - 长距离通信有优势　　硬件成本低

- 并行 Parallel
  - 数据以字节或字为单位传送 – 8位、16位
  - 数据线需求多 – PCB更多布线
  - 速度快
  - 适合短距离通信



通信方式

- 点对点 Point-to-point
  - 仅两个端点或者两个设备之间
  - 无需地址
- Bus总线
  - 若干终端或者设备之间
  - 需要地址 ip地址

通信方式

- 同步 Synchronous
  - 发送端和接收端共享一个时钟信号
  - 传输速度快
- 异步 Asynchronous
  - 发送端和接收端的时钟是独立的
  - 接收端提前知道数据传输速率
  - 需要数据包，并带起始位/停止位
  - 正常需要一个防过采样的方案
  - 传输速度较慢

通信方式

- 半双工 Half duplex
  - 允许二台设备之间双向传输信息，但不能同时进行。也就是说，同一时间，只允许一方发送，另一方接收，或者反过来。
  - 可以比喻为单线铁路。无列车时，任一方向的车都可以通过。有列车时，另一方向必须等待该车通过后才能通行。
  - 比如：无线电对讲机，长江:“……”，黄河：“……”
- 全双工 Full duplex
  - 允许二台设备同时进行资料传输
  - 可以用双向车道来形容。两个方向的来车使用不同车道，不会相互影响。
  - 举例：电话机.....

通信方式

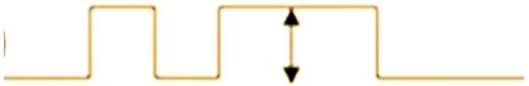
- 对等方式 Equal partners
  - 在通道空闲时，任何节点都可以传送数据
  - 需要仲裁
- 主从方式 Master-slave
  - 只有主机能发起通信
  - 从机在得到允许情况下与主机通信

主从方式：主机为电脑

通信信号

- 单端信号 Single-ended
  - 以单电压代表逻辑1或0，接地作为电压参考基准
  - 通信双方共享接地端

以相对接地电压作为标准



- 差分(模)信号 Differential
  - 以两根线的电压差来传递信号
  - 收发双方不需要共地端
  - 能获得更高速通信

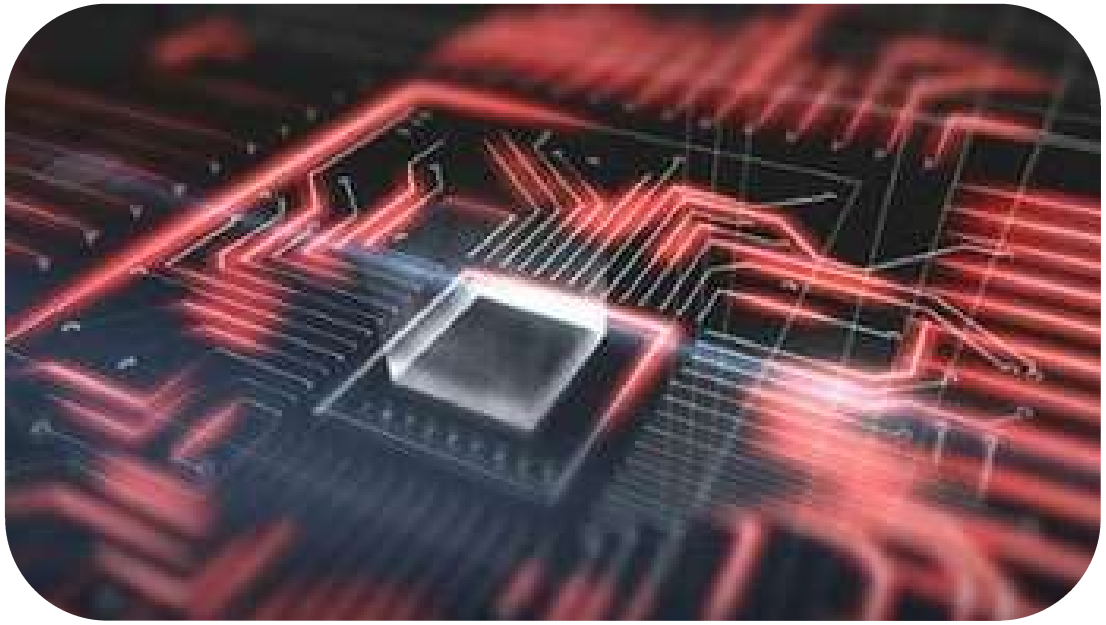
电压差作为标准

通信总线

- 嵌入式技术中的通信总线

总线类型	线数	通信类型	多主机	波特率 (bps)	挂接 设备数目	总线长度 (m)
UART	2	异步	No	3K~4M	2	1.5
SPI	3	同步	Yes	>1M	<10	<3
I2C	2	同步	Yes	<3.4M	<10	<3
CAN	2	异步	Yes	20K~1M	128	40
USB	2	异步	No	480M	127	5

嵌入式系统(EMBEDDED SYSTEM)



第5章 通信接口与总线

- **5.2 异步串行通信UART**

5.2 异步串行通信UART

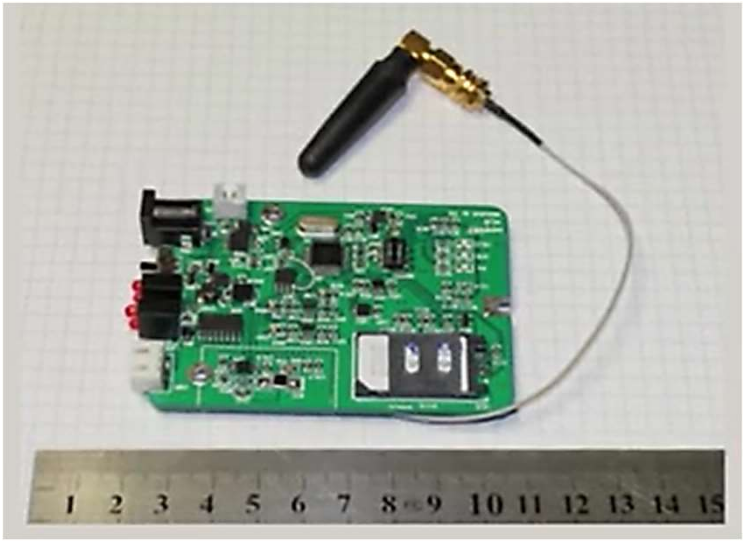
- 本节内容
  - UART原理 I、II
  - RS232串口和USB虚拟串口
  - STM32的串口
  - 常用库函数与用户建库
  - UART编程实例 – PC通信
  - 编程练习E3 ： UART串口通信编程

UART原理 I

- **术语**
  - UART – Universal Async. Receiver Transmitter
    - 通用异步接收/发送装置（总称）
  - SCI – Serial Communication Interface
    - 串行通信接口（Motorola/Freescale）
  - 特征 Characteristics
    - 串行 Serial
    - 异步 Asynchronous
    - 点对点 Point-to-point
    - 全双工 Full-deplex
    - 对等 Equal partners

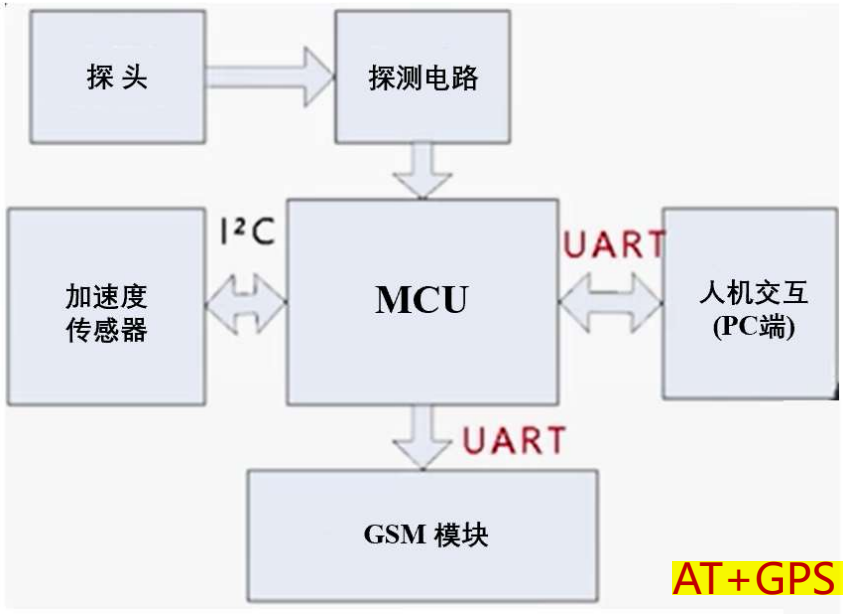
UART原理 I

- 一个放射源探测终端



UART原理 I

- 终端中的UART通信单元

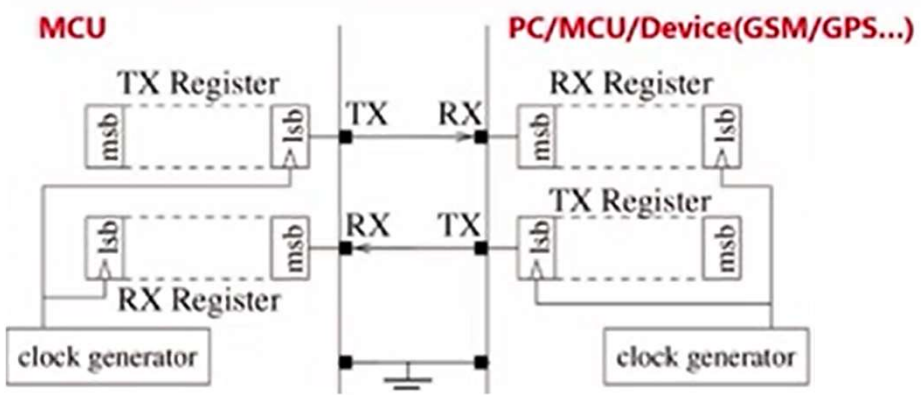


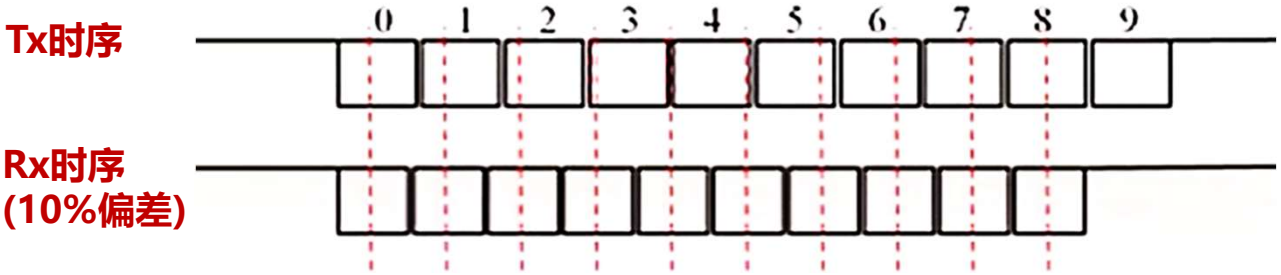
AT+GPS = “13912345678”

“$GPRMC,092427.604,V,4002.1531,N,11618.3097,E,0.000,0.00,280819,,E,N*08”

UART原理 I

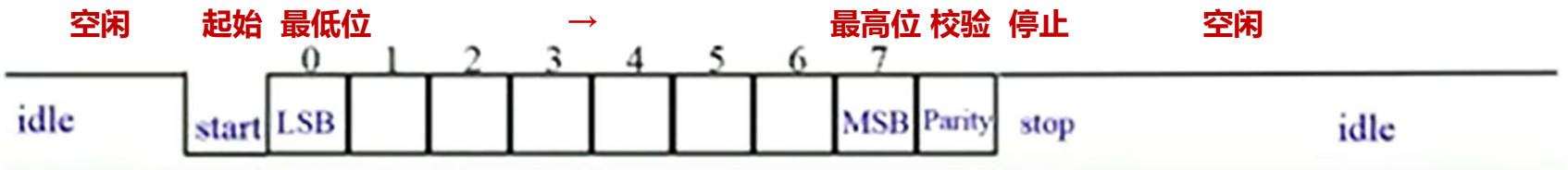
- 如何通信?
  - 发端TX，收端RX
  - 收发引脚，两对　全双工
  - 共地
  - 独立时钟，约定速率





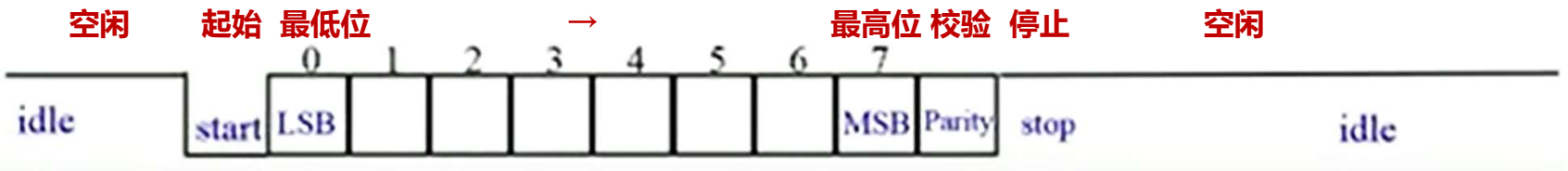
UART原理 I

- 帧格式 Frame format



- 编码：NRZ（not return zero） Encoding “不归零编码”
- 起始/停止位：帮助接收端获知一个字节的起始和停止
  -- 将空闲状态拉低即发出起始位，起始位永远是0，它出现在每个字节的开头
  -- 停止位永远是1，它出现在每个字节数据的末尾
- 波特率：Baud rate，每秒传输的数据位数（bits per second，bps）
- 避免了异步时钟误差的累积！
  -- 想一下，收发双方允许的波特率误差是多少?

UART原理 I



- 数据位：字节为单位，LSB ~ MSB
  -- LSB – Least significant bit 最低位
  -- MSB – Most significant bit 最高位
- 若干描述参数
  -- Baud rate，#data bits，parity，#stop bits
- 描述术语：9600，8N1（简写）
  -- 如果采用9600，8N1，则每秒可以完成多少字节数据量的传输?
  - Hints：考虑起始位和停止位 一秒9600bits，8位一帧，N表示不需要校验位，1是停止

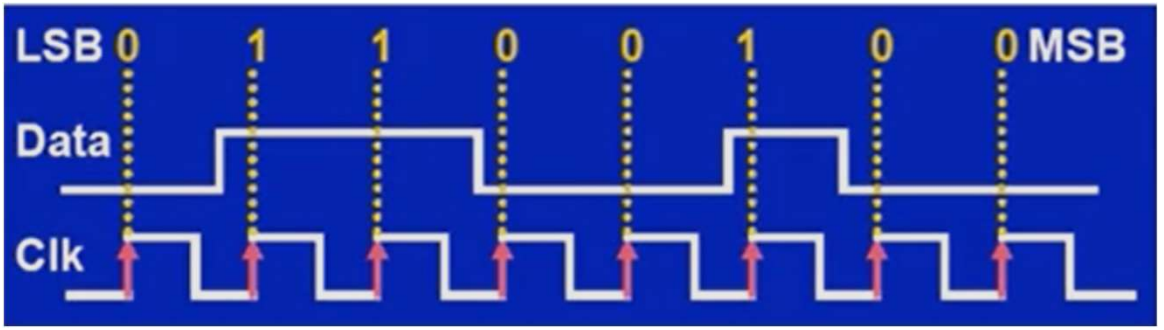
9600/10

UART原理 I

- 概念小结
  - 特征
    - Serial、 Asynchronous 、Point-to-point、Full-deplex ……
  - UART – universal Async. Receiver Transmitter
    - 通用异步接收/发送装置（总称）
    - 一种统称，第一块UART芯片出现与1971年
    - 具体实现上，可以是RS-232、RS485、RS422等
  - SCI – Serial Communication Interface
    - 串行通信接口
    - Motorola/Freescale，since 1975

UART原理 II

- 0与1的准确识别



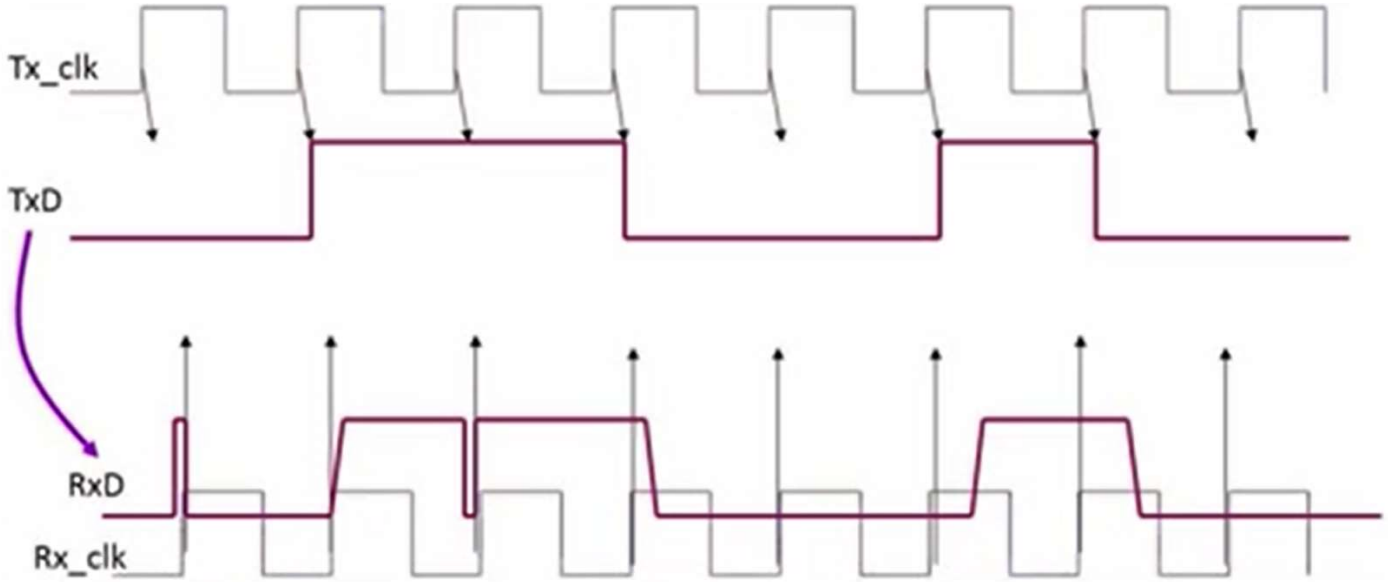
- 接收端在每个时钟上升沿采样数据
- 上面传输的数据是什么?

  -- 发时 LSB在先，MSB在后

  -- 二进制格式 0b00100110，十六进制格式 0x26

UART原理 II

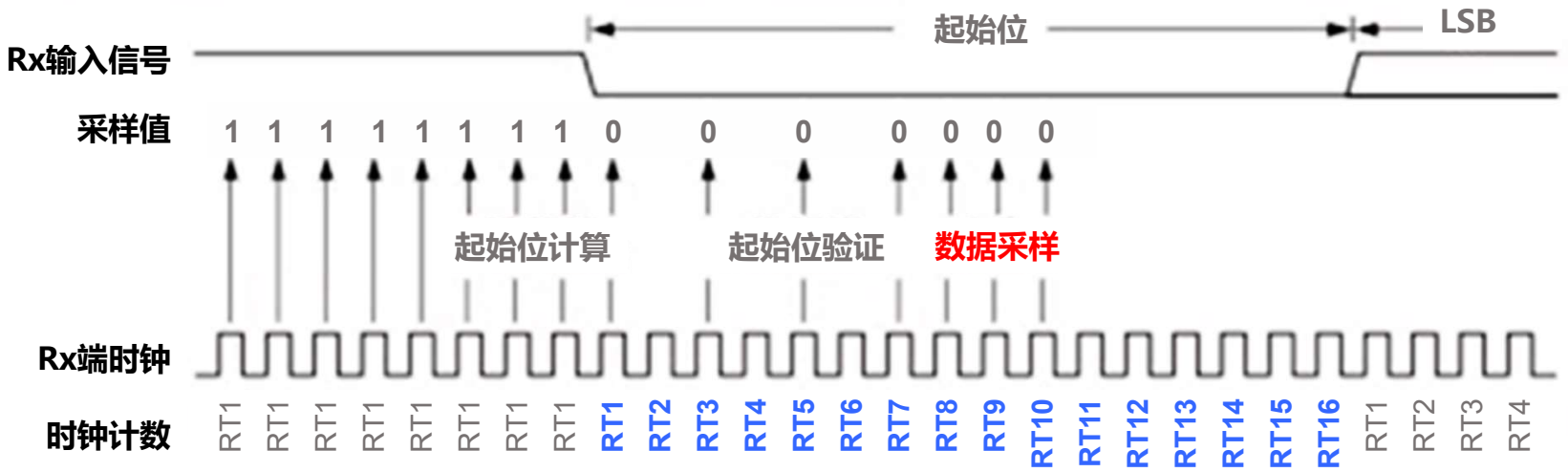
- 信号失真



  - 经过线缆传输后，接收到的信号可能夹杂毛刺，信号边沿也不再陡峭

UART原理 II

过采样(Oversampling)



- 接收端采样时钟速率是波特率的16x
- 选择16次采样的中间3个bits用于“投票”（why?）

  -- “少数服从多数原则”来决定信号电平

  -- 如果中间3个bits值不一致，Noise flag将被置位

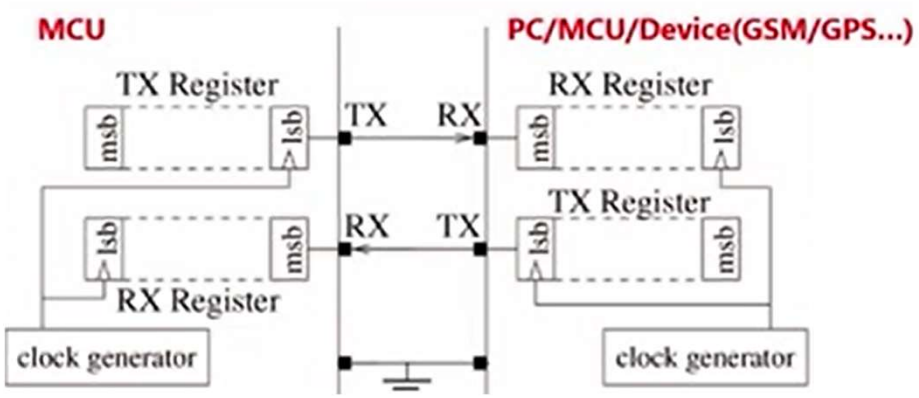
UART原理 II

RT8，RT9，RT10 Samples	Data Bit Determination	Noise flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

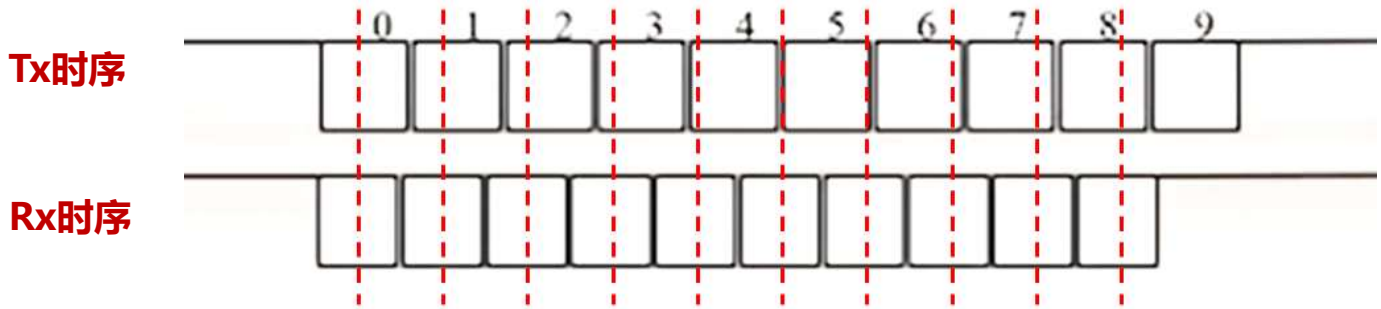
- **Question**：一个最高时钟为16MHz的MCU，其UART能达到的最高波特率是多多少?

UART原理 II

- 如何通信? （回顾）

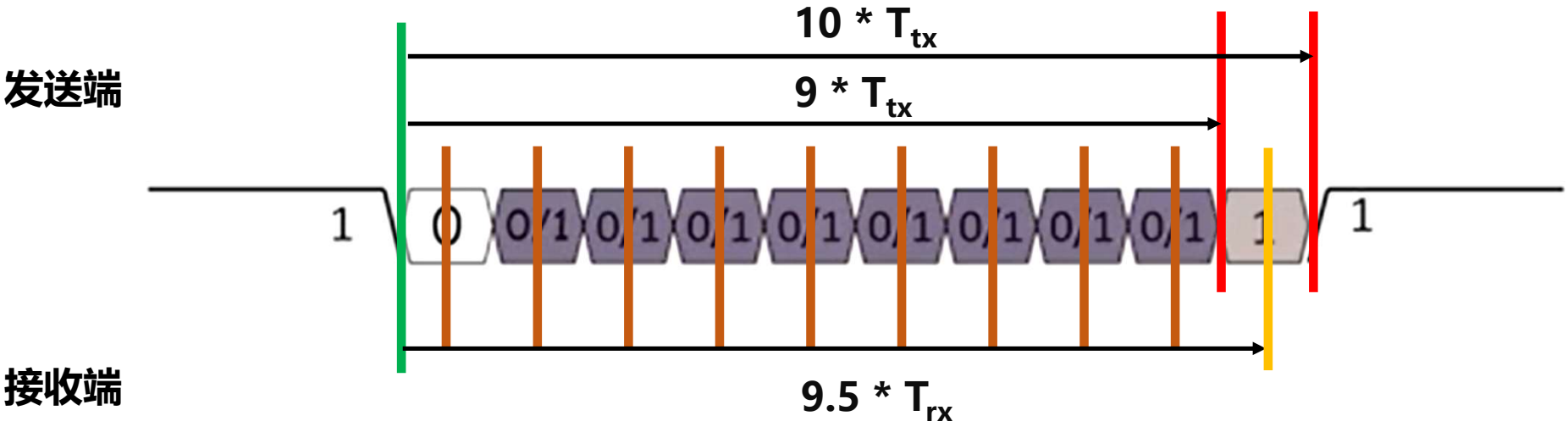


收发双方的时钟偏差多小才够小?



UART原理 II

◆ **时钟容许偏差**



- 接收端要能正确采样到停止位

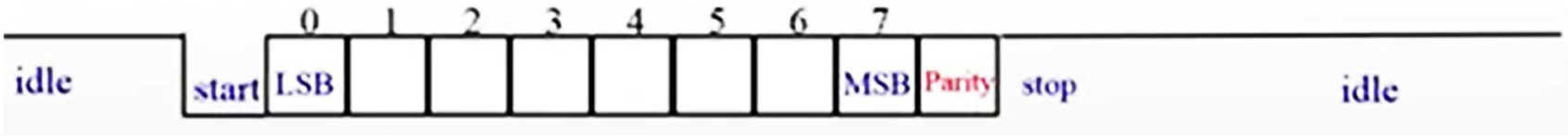
$$9 * T_{tx} < 9.5 * T_{rx} < 10 * T_{tx}$$

Clock difference tolerance

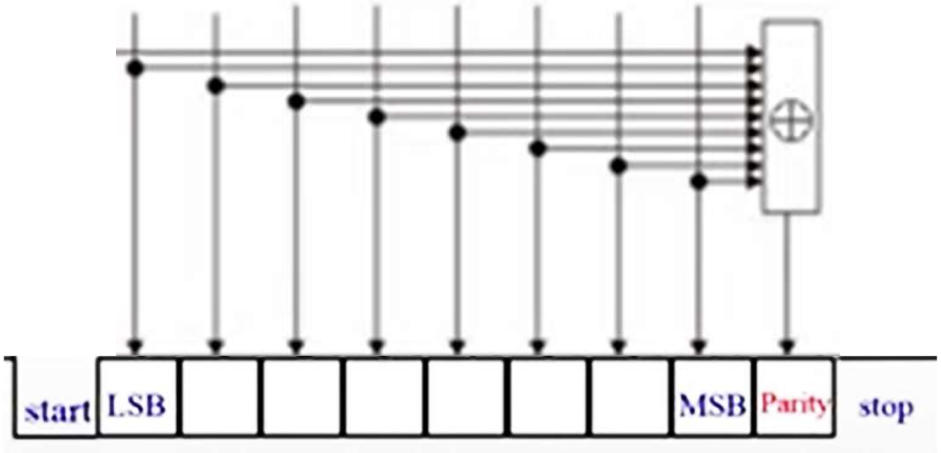
≈ 5%

UART原理 II

数据帧校验



- 校验位(Parity bit)被添加在帧尾，用于检查数据的完整性



- **奇偶校验位**用一个bit位来表示传输的数据中“1”的个数是奇数还是偶数

UART原理 II

奇偶校验释义

- 偶校验（Even parity）

  -- 要求数据中1的总个数是偶数（包括校验位）

- 奇校验（Odd parity）

  -- 要求数据中1的总个数是奇数（包括校验位）

- 举例：

  -- 偶校验，“10011100”，校验位 = 0

  -- 奇效验，“01010110”，校验位 = 1

- Questions：-- 奇偶校验能发现所有的通信错误吗？ 不能，偶数位错误

  -- 奇偶校验能帮助纠正错误吗？ 不能发现出现错误的bit位

UART小游戏

- Let’s paly a game!
  - 老师是UART发送方，同学们是接收方
  - 协议为 1，8N1（1bpf)
  - 举旗代表“1”，放下代表“0”
  - 发送ASCII
    - -- ‘U’ = 0b01010101
    - -- ‘Z’ = ? = 0b01011010

单选题 1分

设置

某UART通信参数为：9600，8N1，则每秒可以完成多少字节数据量的传输?

A 9600字节

B 1200字节

C 960字节

D 8字节

提交

5.2 异步串行通信UART

- Next……
  - UART原理 I、II
  - RS232串口和USB虚拟串口
  - STM32的串口
  - 常用库函数与用户建库
  - UART编程实例 – PC通信
  - 编程练习E3 ： UART串口通信编程

RS232标准(串口)和USB虚拟串口

RS232标准(串口Serial Port)

- **EIA RS-232-C**

  美国电子工业协会正式公布的异步串行通信标准，也是目前最常用的异步串行通信标准，用于实现计算机与计算机之间、计算机与外设之间的数据通信。

- RS-232-C电平采用**负逻辑**，即逻辑1：-3V~-25V，逻辑0：+3~+25V（需要电平转换）

  RS-232-C接口标准有22根线，采用标准25芯D型插头座，**PC上使用简化的9芯D型插座**

  多种通信应答（握手）方式：硬件握手，自应答，XON/XOFF模式

- 波特率：300bps~4Mbps，物理层有多种实现方式（铜缆，光缆，红外，无线，微波）

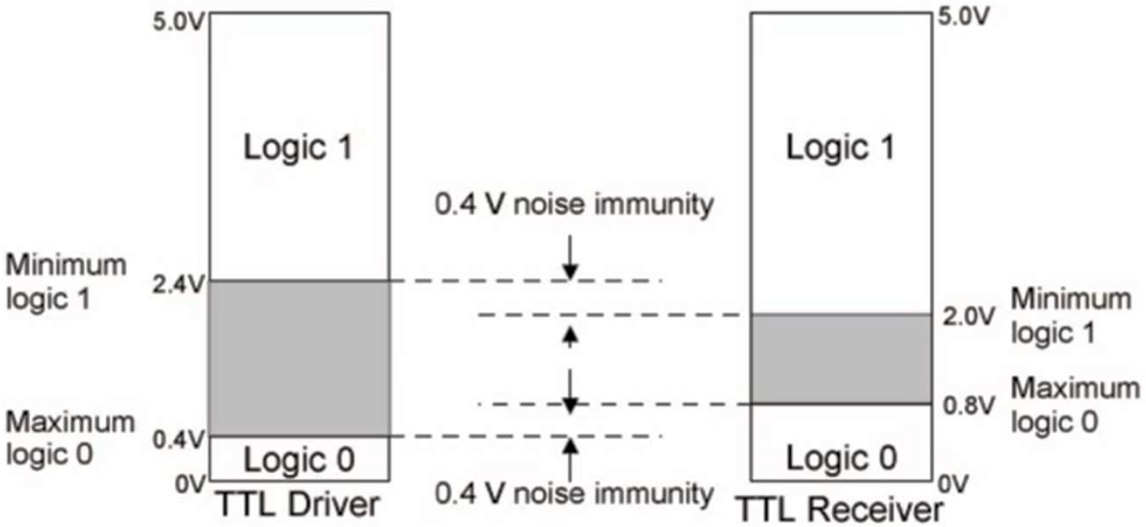
  起始位/停止位/数据位/奇偶校验/全双工半双工模式

- 广泛应用：Modem，20mA电流环，RS485

  个人计算机都有标准的RS232接口，驱动程序，超级终端

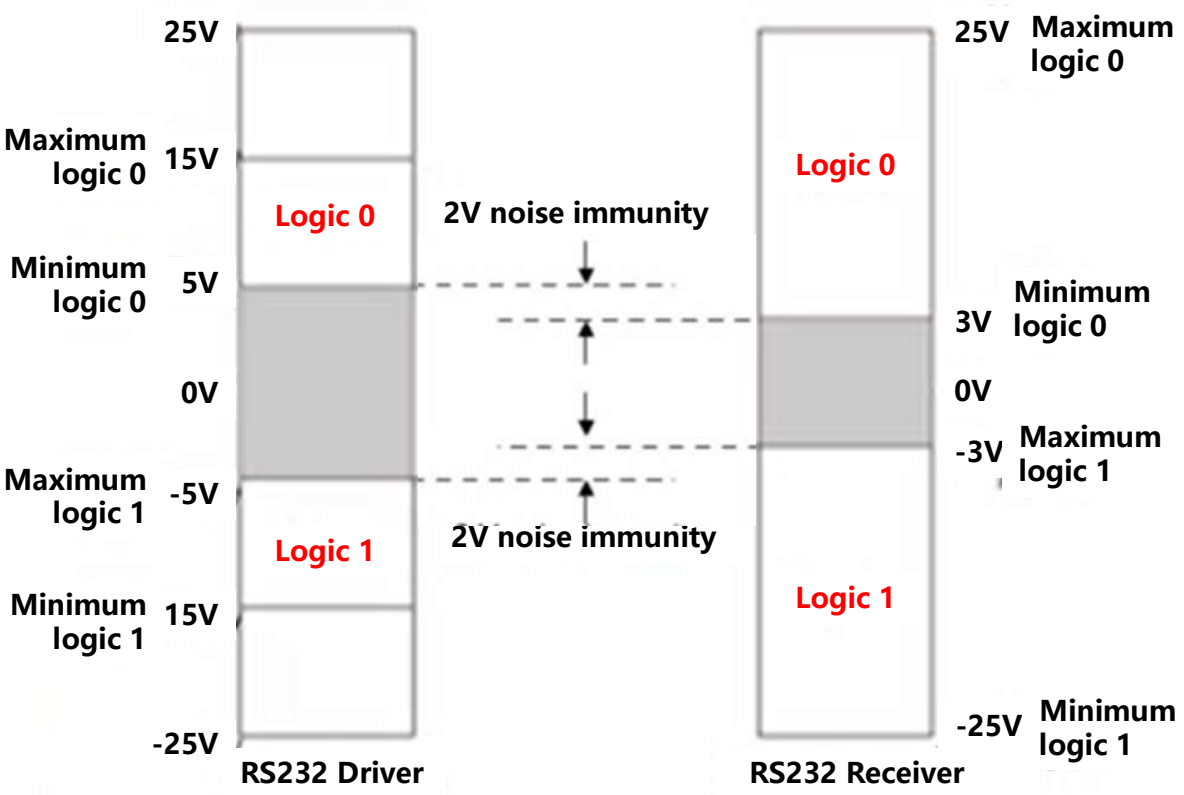
RS232标准

- TTL Level



RS232 standard

- **RS232 Level**

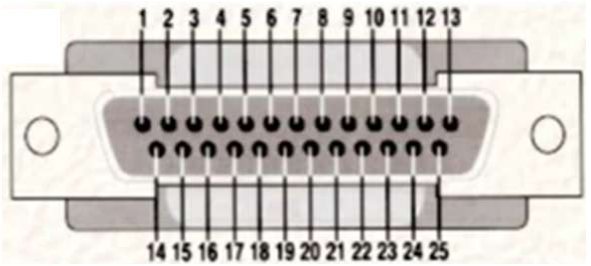


RS232 标准

- **RS232 DB-25 Pins**

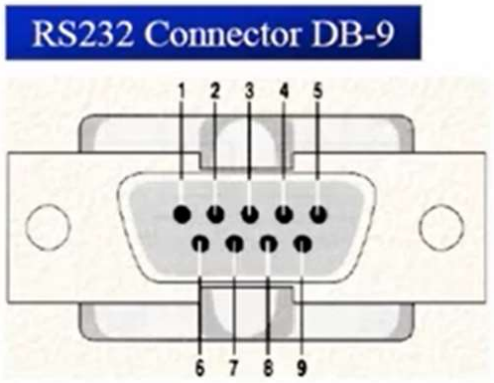
Pin	Description	Pin	Description
1	Protective ground	14	Secondary transmitted data
2	Transmitted data (TxD)	15	Transmitted signal element timing
3	Received data (RxD)	16	Secondary receive data
4	Request to send (-RTS)	17	Receive signal element timing
5	Clear to send (-CTS)	18	Unassigned
6	Data set ready (-DSR)	19	Secondary receive data
7	Signal ground (GND)	20	Data terminal ready (-DTR)
8	Data carrier detect (-DCD)	21	Signal quality detector
9/10	Reserved for data testing	22	Ring indicator (RI)
11	Unassigned	23	Data signal rate select
12	Secondary data carrier detect	24	Transmit signal element timing
13	Secondary clear to send	25	Unassigned

- RS232 Connector DB-25



RS232 standard

- IBM引入了DB9版本的串口标准

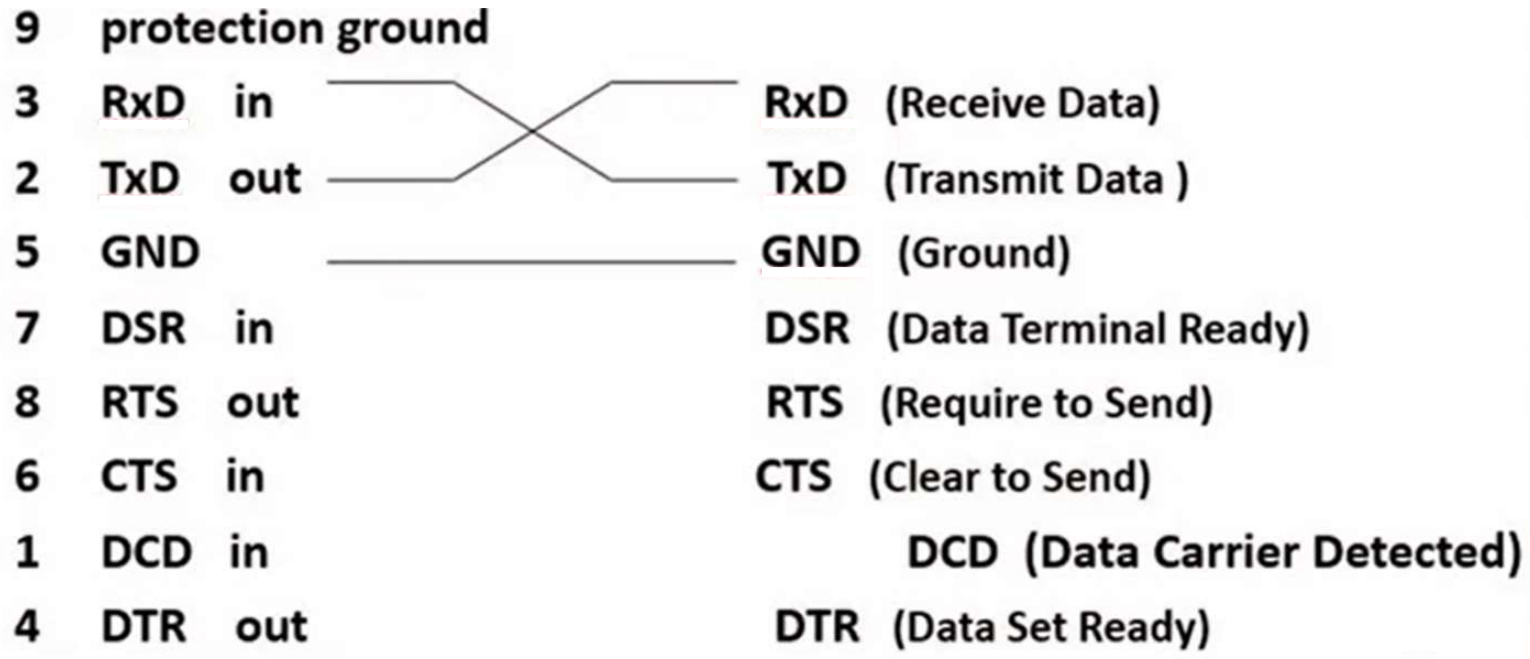


RS232 DB-9 Pins

Pin	Description
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)
8	Clear to send (-CTS)
9	Ring indicator (RI)

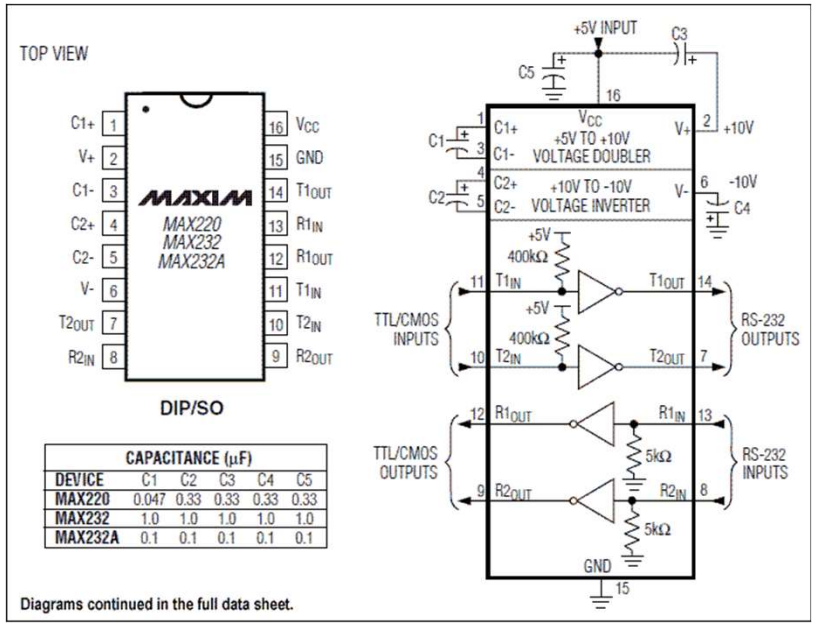
RS232标准

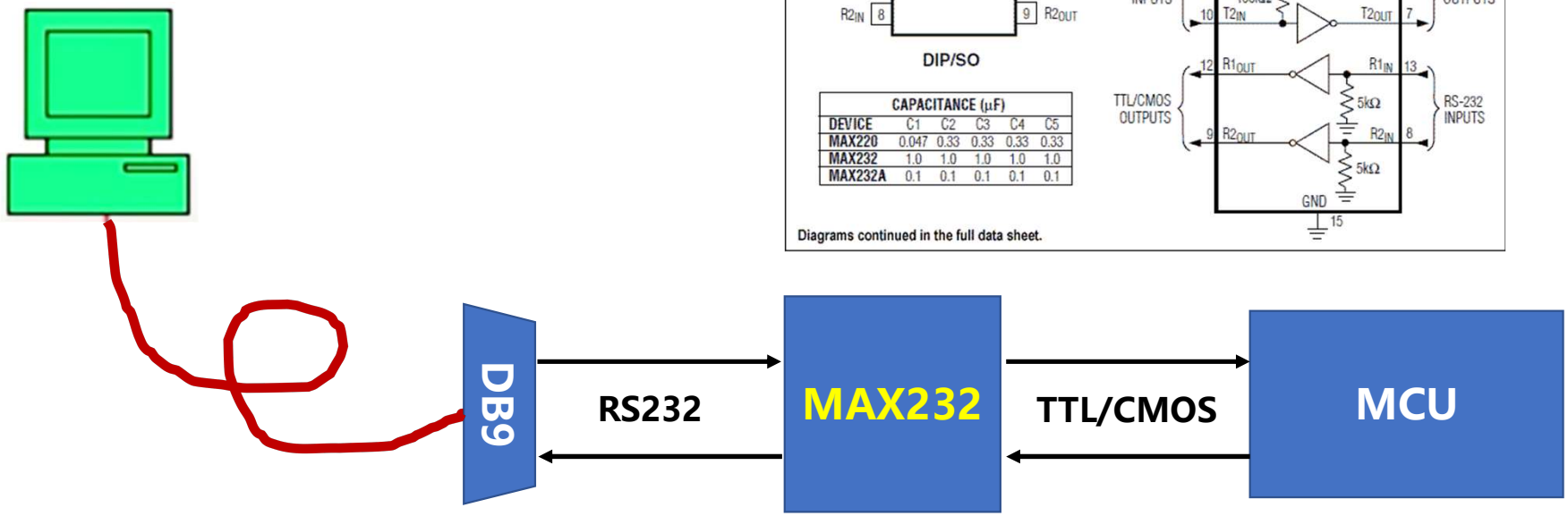
- 简化的9芯D型接头信号定义
  - 三线制无应答方式



RS232 ⇌ TTL logic

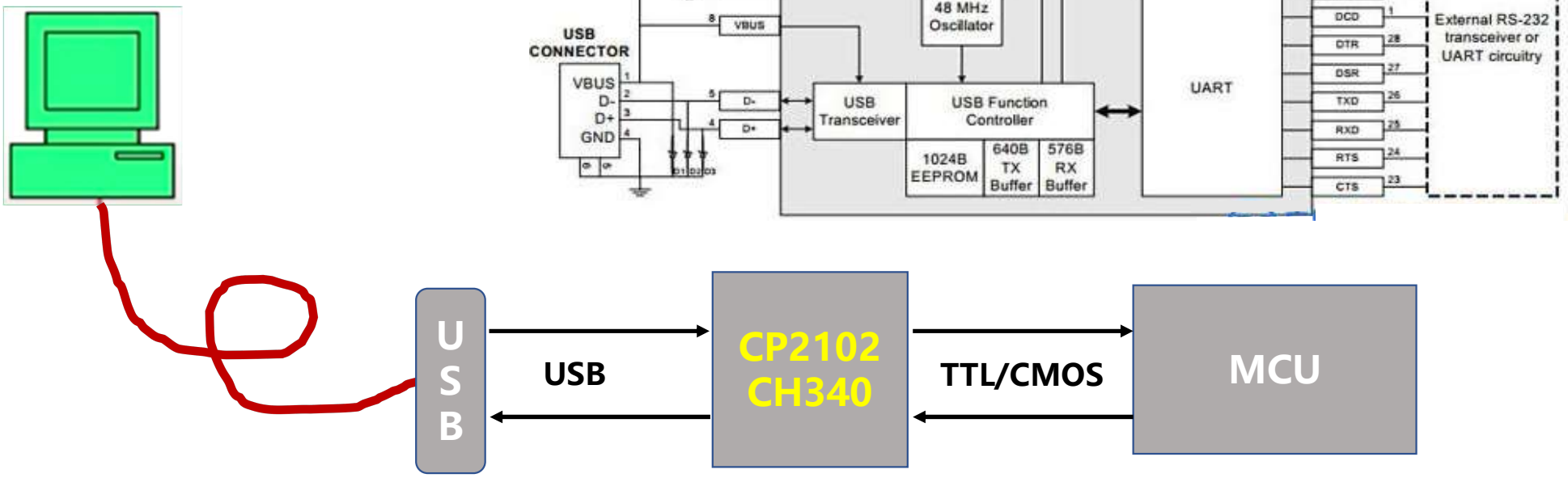
- RS232的出现早于TTL逻辑
- 需要一个接口芯片来完成转换





USB ⇌ UART

- 接口芯片将USB转为TTL UART
- PC上会出现一个虚拟串口设备
  (需安装驱动)



5.2 异步串行通信UART

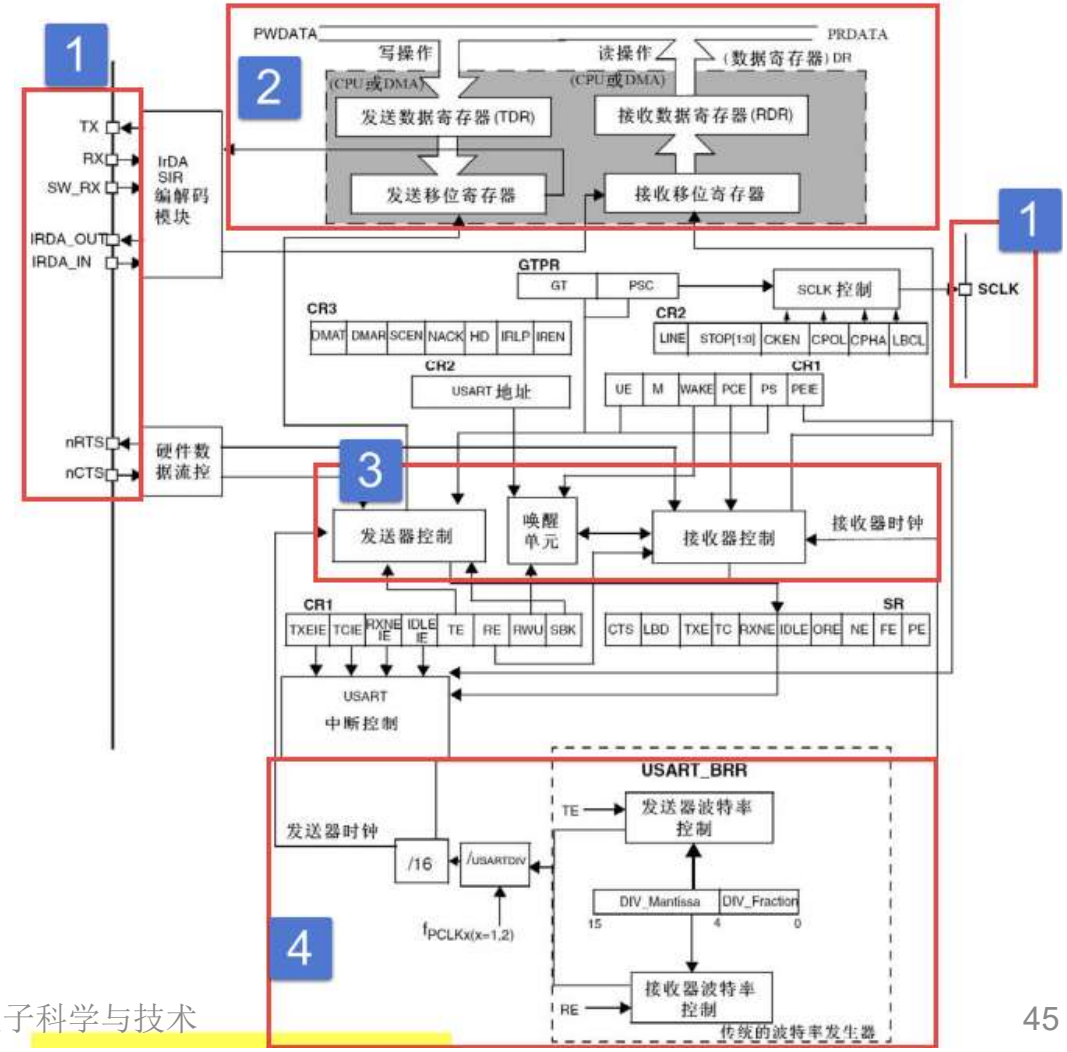
- Next……
  - UART原理 I、II
  - RS232串口和USB虚拟串口
  - STM32的串口
  - 常用库函数与用户建库
  - UART编程实例 – PC通信
  - 编程练习E3 ： UART串口通信编程

STM32的串口

- USART
  - -- Universal Synchronous Asynchronous Receiver and Transmitter
  - -- 通用同步异步收发器
  - -- 能对外提供时钟输出，实现同步传输
  - -- 可配置为UART使用
- UART
  - -- Universal Asynchronous Receiver and Transmitter
  - -- 通用异步收发器
  - -- 收发双方独立时钟

USART结构

- **① 功能引脚**
  - -- TX 发送端
  - -- RX 接收端
  - -- RTS、CTS 握手信号
  - -- SCLK 时钟输出（同步模式）
- **② 数据寄存器 DR**
  - -- TDR 发送寄存器
  - -- RDR 接收寄存器
- **③ 控制器**
  - -- 收发控制
  - -- 使能控制
- **④ 波特率生成**
  - -- 每秒传输比特数（bps）
  - -- 常用 2400, 9600, 19200, 115200



STM32串口引脚

- **F407IG芯片**

  --数据手册 Chapter 2.2.23：USART description

  --数据手册 Chapter 3：Pinouts…description – Table 9. Alternate function mapping

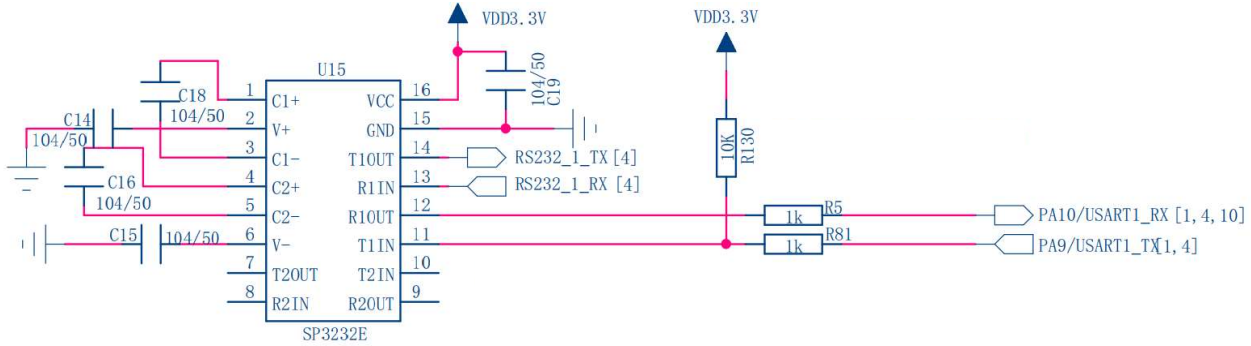
引脚	APB1 总线 (max. 42MHz)				APB2 总线 (max. 84MHz)	
	USART2	USART3	UART4	UART5	USART1	USART6
TX	PA2/PD5	PB10/PC10 /PD8	PA0/PC10	PC12	PA9/PB6	PC6/PG14
RX	PA3/PD6	PB11/PC11 /PD9	PA1/PC11	PD2	PA10/PB7	PC7/PG9
SCLK	PA4/PD7	PB12/PC12 /PD10	-	-	PA8	PC8/PG7
CTS	PA0/PD3	PB13/PD11	-	-	PA11	PG13/PG15
RTS	PA1/PD4	PB14/PD12	-	-	PA12	PG8/PG12

开发板引出串口

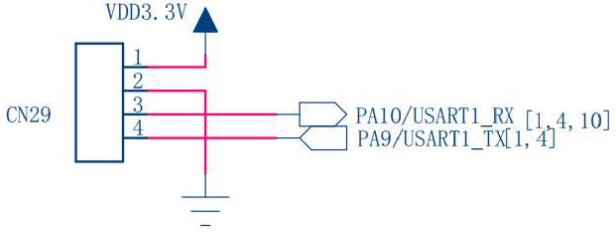
- **开发板提供的串口**

  -- 开发板电路原理图、用户手册

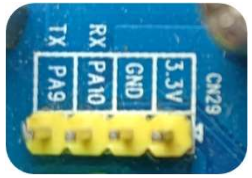
  -- USART1：TX PA9，RX PA10



RS232 DB9接口/插座



TTL 电平接口/插座



5.2 异步串行通信UART

- Next……
  - UART原理 I、II
  - RS232串口和USB虚拟串口
  - STM32的串口
  - 常用库函数与用户建库
  - UART编程实例 – PC通信
  - 编程练习E3 ： UART串口通信编程

常用库函数

- USART_Init (USART_TypeDef* USARTx, USART_InitTypeDef* USART_InitStruct)

  功能：将串口 USARTx 按照结构体 USART_InitStruct 的参数进行初始化

- 串口初始化结构体

```
typedef struct
{
  uint32_t      USART_BaudRate;       /*波特率，9600或115200 .....*/
  uint16_t      USART_WordLength;     /*数据帧字长，8或9*/
  uint16_t      USART_StopBits;       /*停止位，1或1.5 .....*/
  uint16_t      USART_Parity;         /*奇偶校验，No或Even或Odd*/
  uint16_t      USART_Mode;           /*模式，Rx或Tx或二者*/
} USART_InitTypeDef;
  //定义初始化结构体变量
  USART_InitTypeDef  USART_InitStructure;
```

常用库函数

- USART_SendData(USART_TypeDef* USARTx, uint16_t Data)

  功能：将数据Data通过串口USARTx发送出去，Data有效部分情况

  -- Data[8:0]，9位，含一位校验位

  -- Data[7:0]，8位，无校验位

- uint16_t USART_ReceiveData(USART_TypeDef* USARTx)

  功能：从串口USARTx接收一个数据并将其返回，该数据两种有效位数同上

- USART_Cmd(USART_TypeDef* USARTx, FunctionalState NewState)

  功能：使能/关闭串口USARTx，NewStat = ENABLE或DISABLE

常用库函数

- FlagStatus USART_GetFlagStatus (USART_TypeDef* USARTx, uint16_t USART_FLAG)
  - 检测指定的USARTx 的状态标志并返回其值， USART_FLAG常用标志有

-- USART_FLAG_TC: Transmission complete，= 1 数据发送完成，=0 未发完

-- USART_FLAG_RXNE: Receive data register not empty，=1 数据已接收待读取，=0 未收完

  - 返回值类型 FlagStatus 为枚举，定义如下

```
typedef enum { RESET = 0, SET = 1} FlagStatus;
```

用户自建函数库

- 以标准库函数为基础，构建更高一级的外设操作函数

  (自建uart.c & uart.h)

  - **//字节发送函数** – 从串口USARTx发送一个字节数据

```
void  UartSendByte (USART_TypeDef  *USARTx,  uint8_t  ch)
{
   USART_SendData (USARTx,  ch);   /*调用标准库的发送函数*/
   while( USART_GetFlagStatus(USARTx,  USART_FLAG_TC) == 0) { };  /*检查发送完成标志*/
 }
```

用户自建函数库

再谈 USART_FLAG_TC 标志

《寄存器手册》Chapter 30.6.1 - USART status register

30.6.1 Status register (USART_SR)

Address offset: 0x00

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

值=1表示完成

Bit 6 **TC:** Transmission complete

0: Transmission is not complete

1: Transmission is complete

系统复位时 TC = 1

用户自建函数库

- **//字符串发送函数** – 将str指向的字符串通过USARTx发送

```
void  UartSendString (USART_TypeDef  *USARTx,  char  *str)
{   uint16_t  k =0;
    do {     /*循环方式逐个字符*/
        USART_SendData (USARTx,  *(str+k) );    /*通过指针*(str+k)定位字符*/
        k++;
        while (USART_GetFlagStatus(USARTx,  USART_FLAG_TC) == 0){ };
      } while ( *(str+k) != '\0' );
 }
```

示例： char info[ ] = “This is a string!”； /* 编译时会在字串结尾自动加 \0 */

```
        UartSendString(USART1,  info);
```

用户自建函数库

- **//字节接收函数** – 从串口USARTx接收一个字节数据

```
uint8_t  UartReceiveByte(USART_TypeDef  *USARTx)
{
    uint8_t  ch;
    while ( USART_GetFlagStatus(USARTx,  USART_FLAG_RXNE) == 0) {};  /*检查接收标志*/
    ch = USART_ReceiveData( USARTx );       /*调用标准库的接收函数*/
    return  ch;                                          /*返回接收值*/
}
```

用户自建函数库

再谈 USART_FLAG_RXNE 标志

《寄存器手册》Chapter 30.6.1 - USART status register

30.6.1 Status register (USART_SR)

Address offset: 0x00

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bit 5 **RXNE:** Read data register not empty

0: Data is not received

1: Received data is ready to be read

系统复位时 RXNE = 0

5.2 异步串行通信UART

- Next……
  - UART原理 I、II
  - RS232串口和USB虚拟串口
  - STM32的串口
  - 常用库函数与用户建库
  - UART编程实例 – PC通信
  - 编程练习E3 ： UART串口通信编程

UART编程实例 – PC通信

功能：从PC端虚拟串口接收一个字节数据并原样发回

```
main()
{

  // 用MCU串口1收发
  ……


}
```

串口编程三步走

- **Step 1：UART初始化**
  - 打开GPIO时钟、打开UART时钟
  - 开启GPIO引脚复用功能
  - 配置GPIO参数、配置USAR参数T
- **Step 2：UART高一级通信函数**
  - 标准函数之上的更高一级封装
  - 通常为字节/字串收发（前已介绍）
- **Step 3：UART高级函数应用**

串口编程1 - UART初始化

```
/* 串口1 TX = PA9，RX = PA10 */
void UartInit (void)
{    //定义初始化用结构体变量
     GPIO_InitTypeDef       GPIO_InitStructure;
     USART_InitTypeDef      USART_InitStructure;
    //打开设备时钟
     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA,  ENABLE);    /* GPIOA   时钟*/
     RCC_APB2PeriphClockCmd (RCC_APB2Periph_USART1, ENABLE);    /* USART1 时钟*/
    //开启GPIO复用USART功能
     GPIO_PinAFConfig(GPIOA, GPIO_PinSource9,  GPIO_AF_USART1);   /*PA9  复用开启*/
     GPIO_PinAFConfig(GPIOA, GPIO_PinSource10, GPIO_AF_USART1);  /*PA10复用开启*/
    //配置GPIO
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9 | GPIO_Pin_10;   /*PA9、PA10同时*/
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;              /*工作在复用模式*/
     GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;              /*使用内部上拉电阻*/
     GPIO_Init(GPIOA, &GPIO_InitStructure);                    /*完成初始化*/
```

串口编程1- UART初始化

```
/* 串口1 TX = PA9，RX = PA10 */
    //配置USART
    USART_InitStructure.USART_BaudRate = 9600;                          /*波特率*/
    USART_InitStructure.USART_WordLength = USART_WordLength_8b ;        /*8bit数据位*/
    USART_InitStructure.USART_StopBits = USART_StopBits_1;              /*1bit停止位*/
    USART_InitStructure.USART_Parity = USART_Parity_No ;                /*无校验位*/
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;  /*收发模式*/
    USART_Init(USART1，&USART_InitStructure);                    /*完成初始化*/
    USART_Cmd(USART1，ENABLE);                                   /*使能串口*/
    USART1->SR &= ~0x0040;                                       /*清TC标志位*/
}          系统上电时TC = 1，清除其标志位
```

同时控制

串口编程2 - UART高一级操作函数

(uart.c & uart.h)

- **//字节发送函数**

  void UartSendByte (USART_TypeDef *USARTx, uint8_t ch)

- **//字节接收函数**

  uint8_t UartReceiveByte (USART_TypeDef *USARTx)

- **//字符串发送函数**

  void UartSendString (USART_TypeDef *USARTx, char *str)

  .....

串口编程3 – 高级函数应用

```
//串口与PC端数据通信主程序
main{
    char  info[ ] =  “串口收发功能测试\n”；    /*定义字符串*/
     ……
    UartSendString (USART1, info);              /*发送info字符串*/
     //从串口接收一个字节，并原样发回
     while(1) {
        uint8_t  ch = 0;
        ch = UartReceiveByte (USART1);   //接收一个字节
        if (ch != 0) {
            UartSendByte (USART1, ch);     //发送一个字节
        }
     }
}
```

PC端 - USB虚拟串口收发工具

- ✓ 编程练习E3.1&E3.2
  UART串口通信编程
  (配套视频E3.1)

思考题 – 第5章通信接口与总线

- 解释UART通信方式为什么是“串行，异步，点对点，全双工，对等”
- 解释UART通信的数据帧格式(Frame format)
- UART术语：115200，8N1 是何意？每秒可完成多少字节数据量的传输？
- ~~解释UART过采样(Oversampling)，包括原因及方法。~~
- 何为偶校验？数据“10010111”做偶校验，校验位的值是多少？
- UART简化的三线制连接是哪三条线？各自用途是什么？
- 如何使MCU的UART和你笔记本电脑的USB接口进行通信？
- STM32F407IG的UART初始化时，为什么要加一条清除TC标志的指令？(TC – Transmission complete)