串 行 通 信

信息工程学院　　陈梅

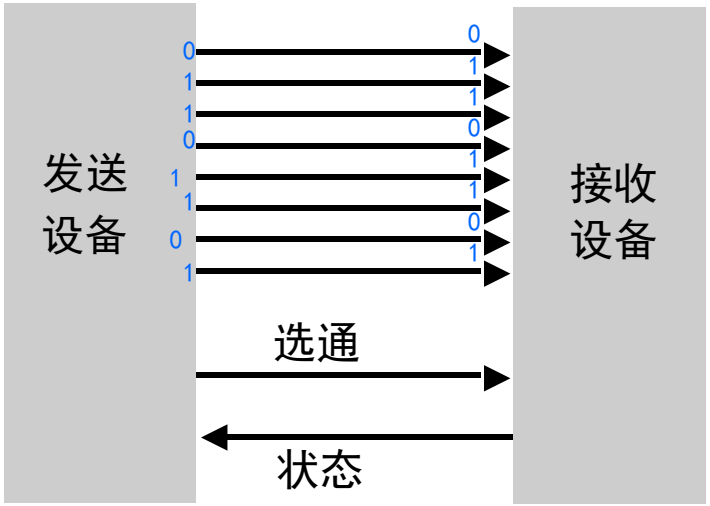
STC89C52单片机串行通信 要点

要点：

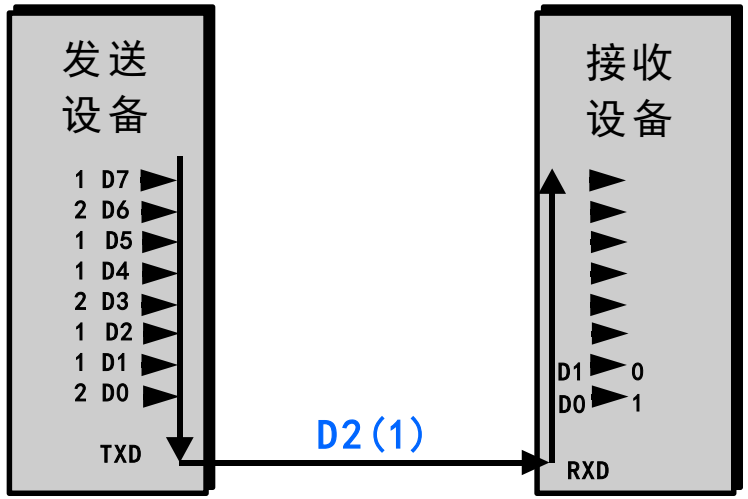
- 串行通信基本概念
- 串行口的结构与工作方式
- 串行口的初始化及波特率的设定与计算
- 串行口的应用
  - 单片机之间的通信
  - PC与单片机间的通信

并行通信和串行通信（数据通信的两种常用形式）

（1）**并行方式**—数据同时发送或同时接收。



（2）**串行方式**—数据依次逐位传送。

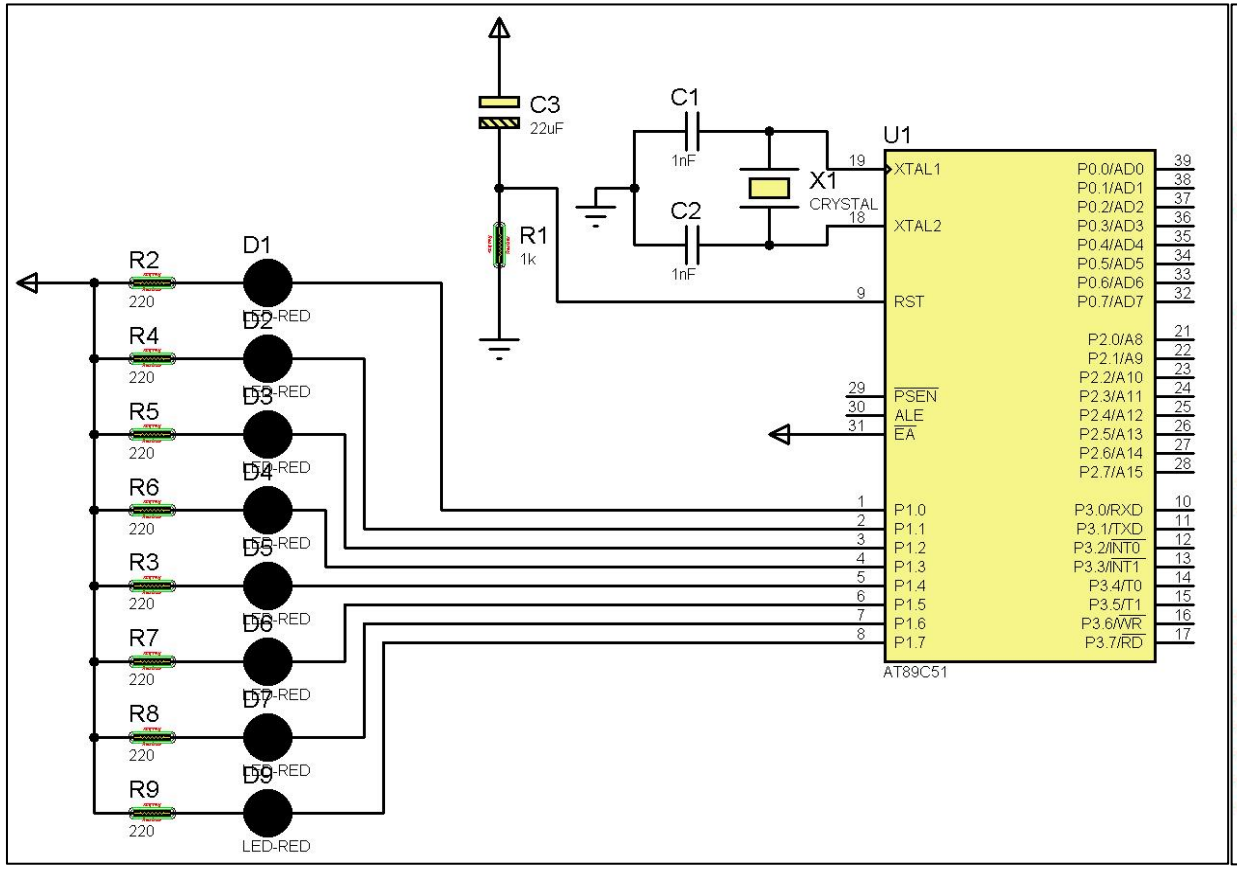


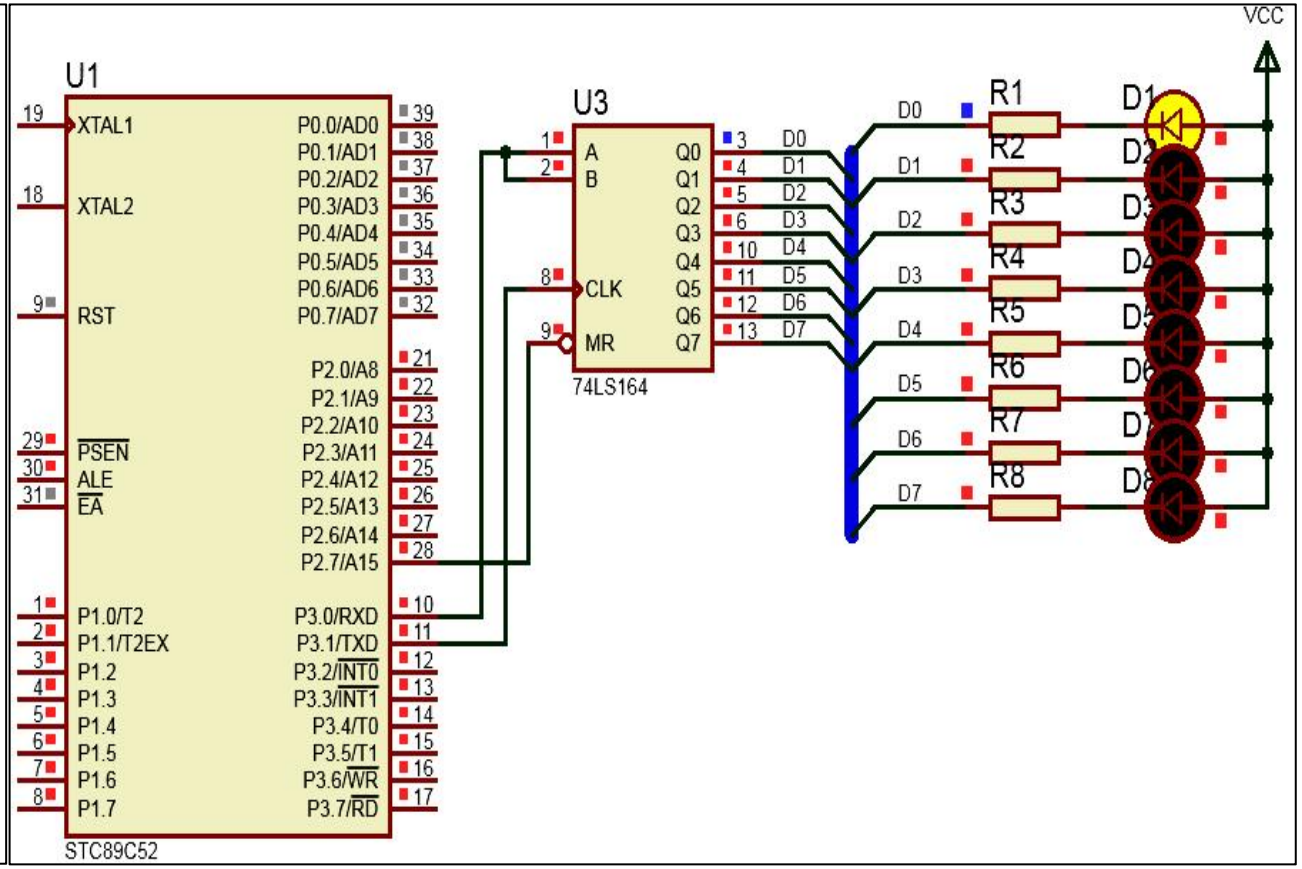
并行传送**优点**：控制简单，传送速度快；
并行传送**缺点**：多根传输线，成本高，一般适于近距离通信。

串行传送**优点**：较少传输线，成本低，适合于远距离通信。
串行传送**缺点**：控制较并行传送复杂，传输速度慢

并行通信

串行通信


C3
22uF
C1
1nF
C2
1nF
X1
CRYSTAL
R1
1k
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
AT89C51
R2
R4
R5
R6
R3
R7
R8
R9
220
D1
D2
D3
D4
D5
D6
D7
D9
LED-RED


VCC
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
STC89C52
U3
A
B
CLK
MR
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
74LS164
D0
D1
D2
D3
D4
D5
D6
D7
R1
R2
R3
R4
R5
R6
R7
R8

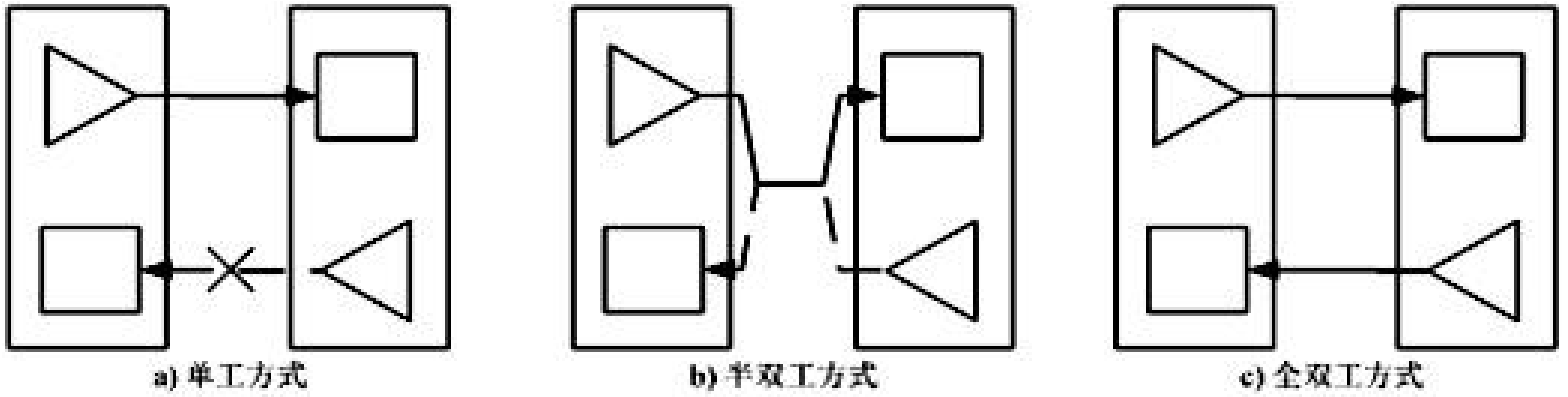
此章所述的串行通信是**以微处理器为核心的系统之间**的数据交换方式；

IIC、SPI、USB等标准串行通信接口是**微处理器系统与非微处理器型外设之间**的数据交换方式。

前者可以是对等通信，而后者只能采用主从方式。

串行通信传输方向

按照数据流向划分3种传输形式：单工、半双工、全双工



单工方式：	一端为发送器TXD，另一端为接收器RXD，数据只能按照一个固定的方向传送。
半双工方式：	由一个TXD和一个RXD组成，不能同时在两个方向上传送，收发开关由软件方式切换。
全双工方式：	每端都有TXD和RXD，可以同时发送和接收，即数据可以在两个方向上同时传送。

异步通信和同步通信

同步和异步的最本质区别在于：通信双方是否使用相同的时钟源。

- 异步通信：指收发设备使用各自的时钟控制数据的传输过程，为使收发协调，要求收发设备的时钟频率在允许的误差范围内尽可能一致。
- 同步通信：指发送方时钟对接收方时钟的直接控制，使数据传送完全同步。

STC89C52单片机只支持异步通信。

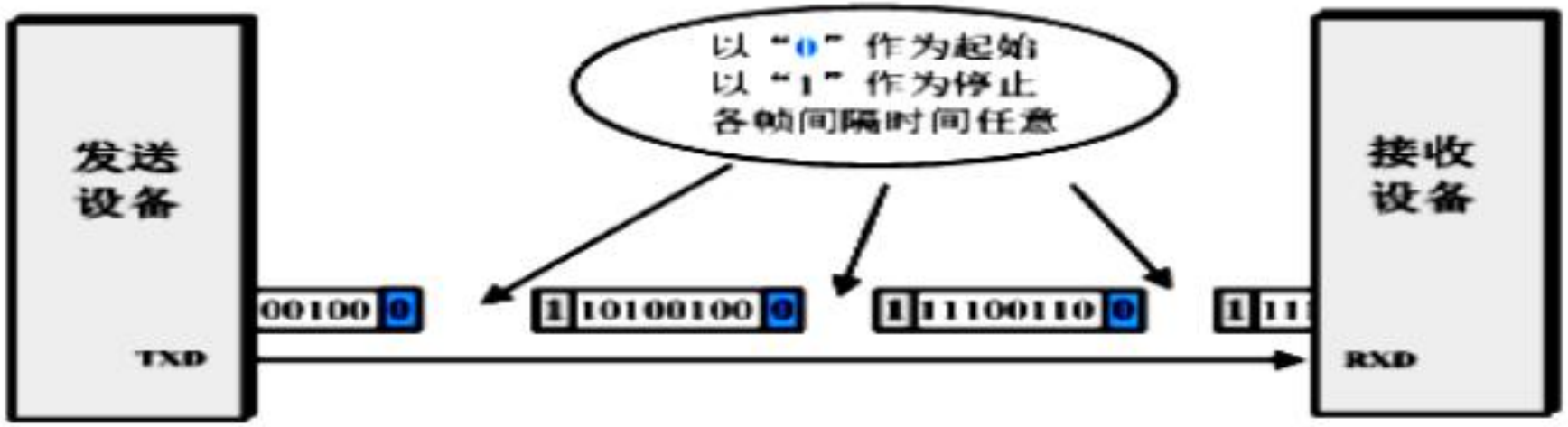
异步通信和同步通信

1．**异步通信**——以字符（帧）为单位传送

传送的每一帧都用起始位进行收发双方的同步，停止位和间隙作为时钟频率偏差的缓冲，故误差仅限在本帧之内。收发双方不同步时，能依靠在每帧开始时的不断对齐，自行纠正偏差，故每个字节都要建立一次同步，适用传送速度较低的场合。串行异步通信是单片机中常用的通信方式。

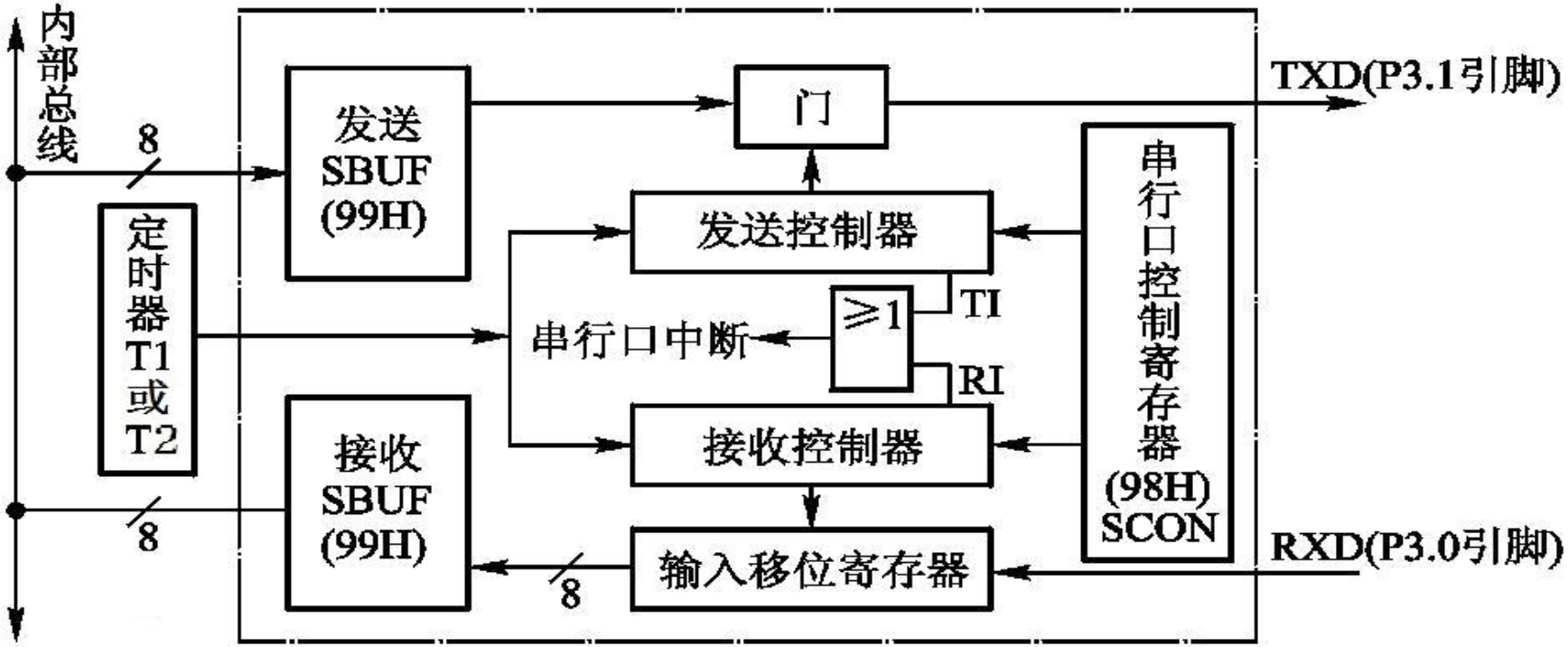
2．**同步通信**——数据以块为单位进行的数据传送

发送方先发送1-2个字节的同步字符(由硬件确定)，接收方检测到后，即准备接收后续的字符数据流。多字节数据之间不允许有空隙，每位占用的时间相等；空闲位需发送同步字符。由于省去了开始和结束标志，而且字节连续没有空闲，其速度高于异步通信。但要求有准确的时钟来实现收发双方的严格同步，对硬件要求较高，适用于成批数据传送。

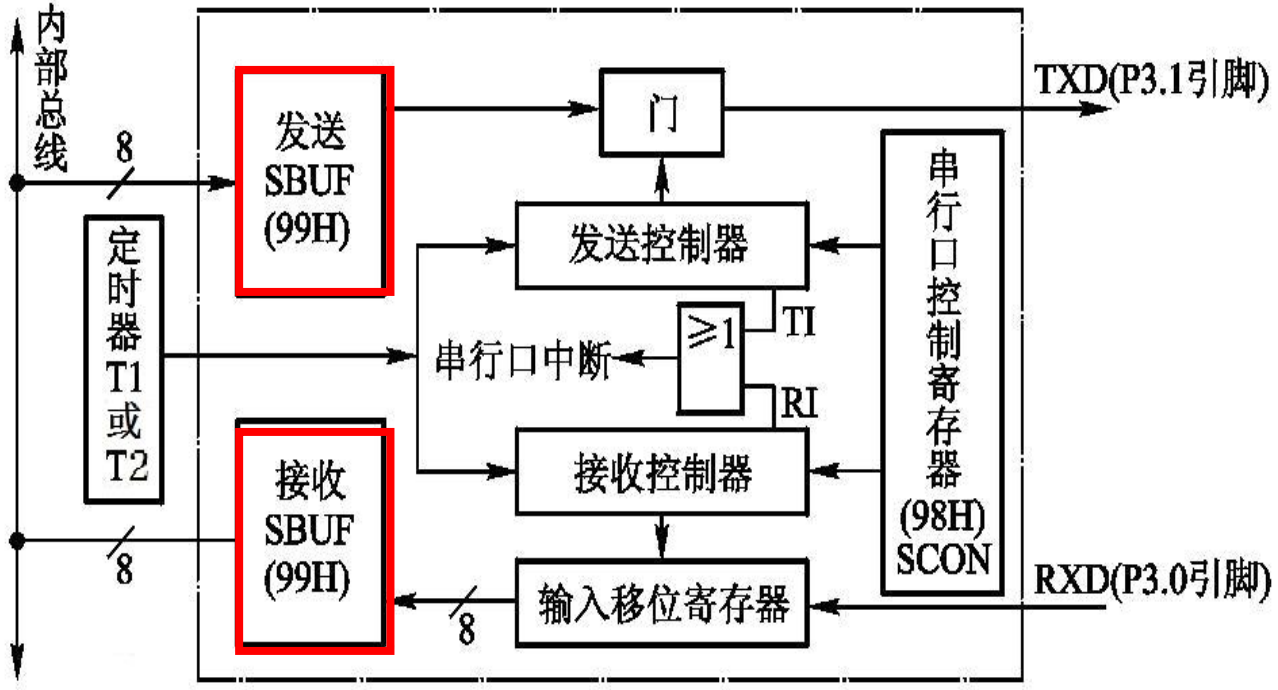


- **起始位：1位，逻辑0。**
  - 用于实现发送方和接收方之间的同步。当不进行数据通信时，通信线路保持高电平，当发送端准备向接收端传输数据时，首先发送起始位0电平，使得电平由高变低，接收端在检测到这一电平变化后，可以准备接收数据。
- **数据位：8位。**
  - 从D0低位到D7高位依次传送。
- **校验位：1位。**
  - 是用户自定义的特征位，用于通信过程中数据差错的校验，或传送多机串行通信的联络信息。常用的差错校验方法有奇偶校验、和校验及循环冗余码校验。
- **停止位：1位，逻辑1。**
  - 位于数据位末尾，用于告知一帧结束，始终为高电平。数据传输结束后，发送端发送逻辑1，将通信线路再次置为高电平，表示一帧数据发送结束。

串行口的结构



STC89C52RC单片机内部集成有一个可编程的全双工的通用异步收发器（UART），也可作为同步移位寄存器使用。



- 物理上**独立**的两个**数据缓冲器SBUF**；
- 一个用于发送数据、另一个用于接收数据。
- 两个SBUF共用一个地址（**99H**），
- 通过读写指令区别是对哪个SUBF的操作。

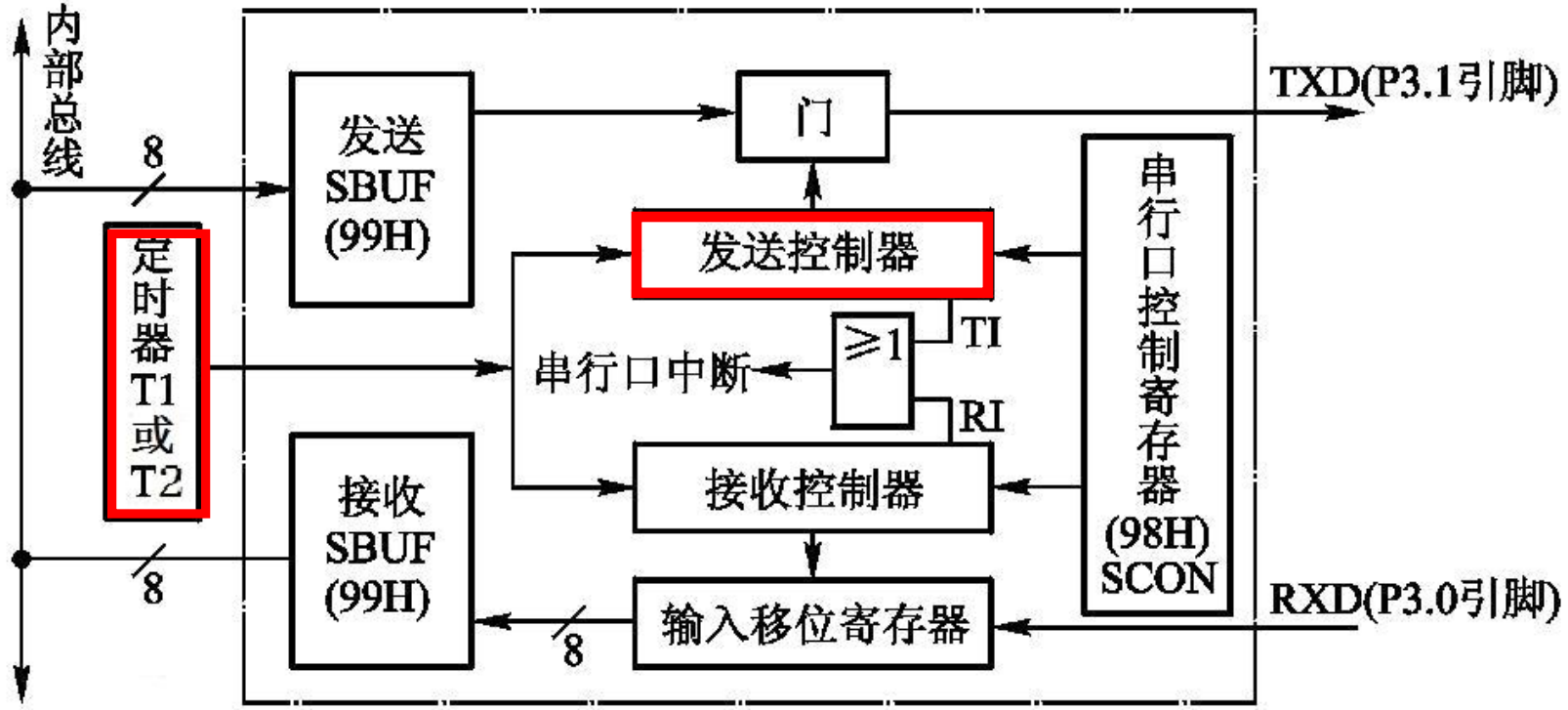
C51:

```
uchar counter;  待发送存放变量
…
SBUF = counter; 完成一次数据发送
…
counter = SBUF; 完成一次数据接收
```

汇编：

```
MOV  SBUF，A   ;完成一次数据发送
…

MOV A，SBUF   ; 完成一次数据接收
…
```

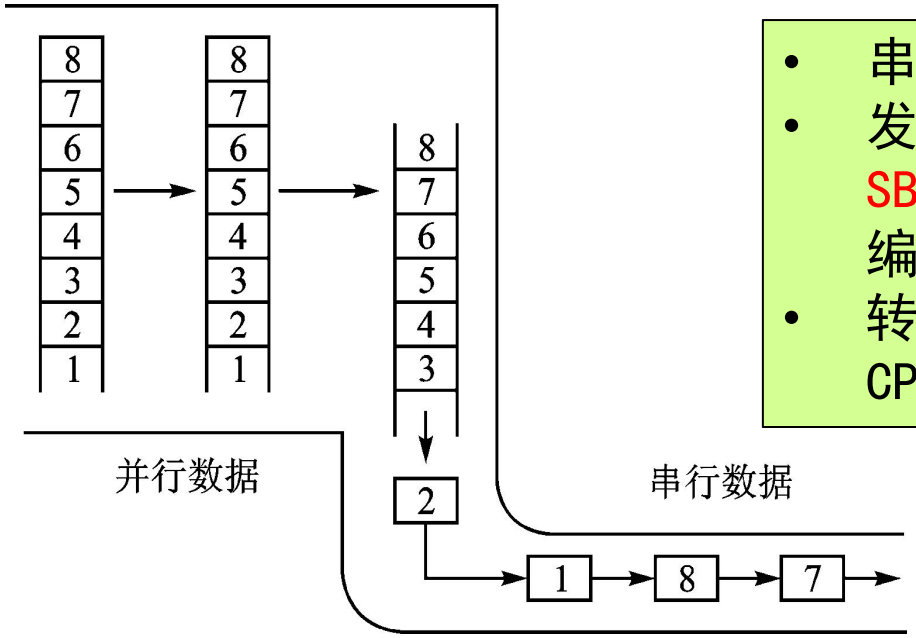


启动一次数据发送:

```
MOV SBUF，A
JNB  TI,   $
CLR  TI
```

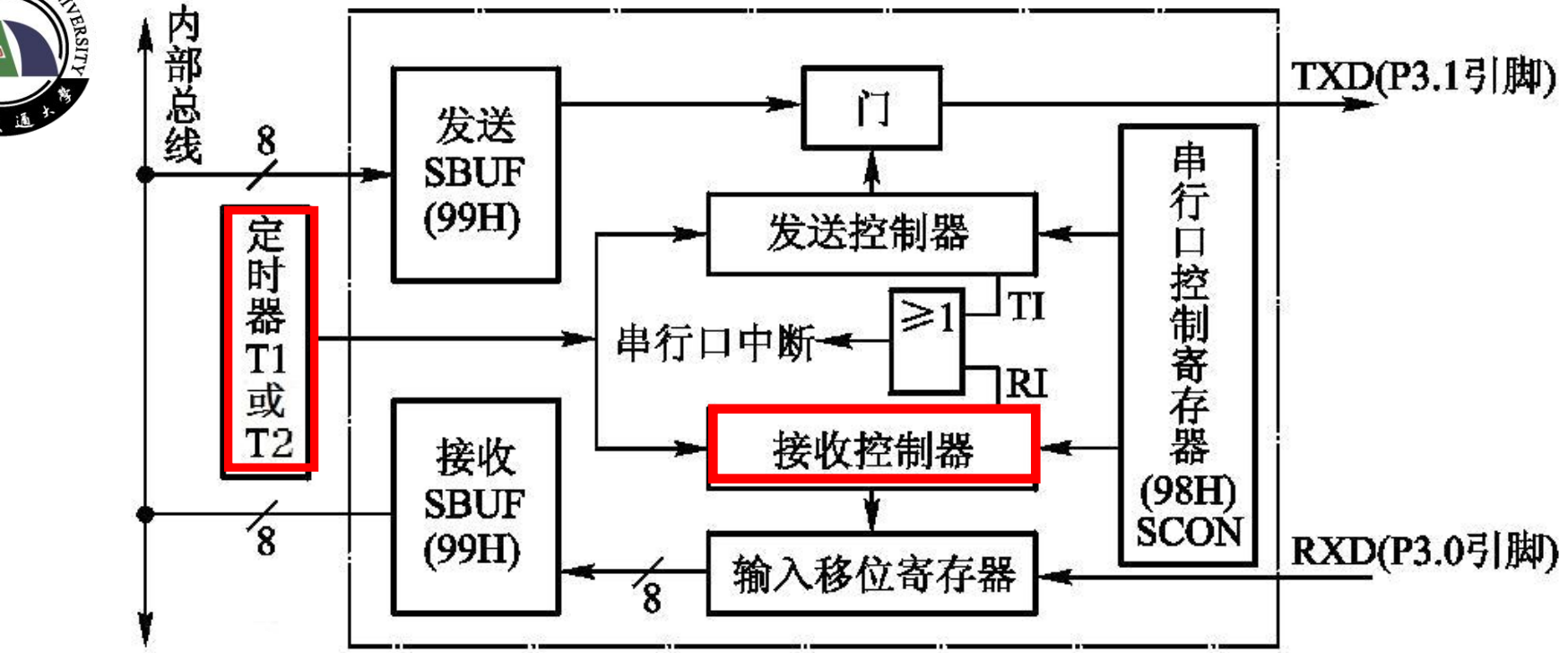
启动一次数据发送:

```
SBUF=counter;
while(TI==0);
TI = 0;
```



- 串行口的发送是主动的，执行MOV SBUF，A启动发送
- 发送控制器是在门电路和定时器T的配合下，将发送缓冲器SBUF中的并行数据——>串行数据，并自动添加起始位、可编程位、停止位。
- 转换结束后自动使发送中断请求标志位TI置1，用以通知CPU已将发送缓存器SBUF中的数据输出到了TXD引脚。

TXD（P3.1）用于串行信号的传出。

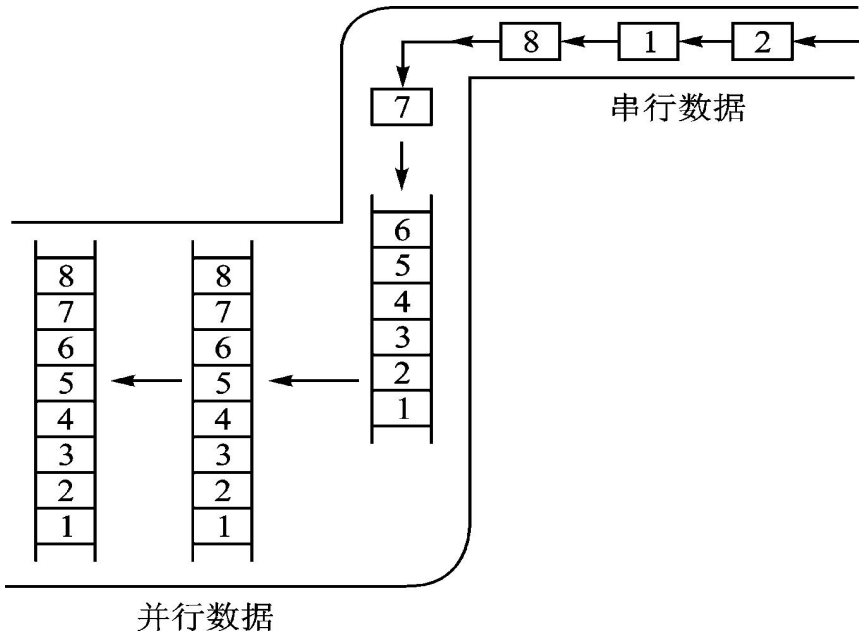


完成一次数据接收:

```
JNB   RI,$
CLR   RI
MOV A，SBUF
```

完成一次数据接收:

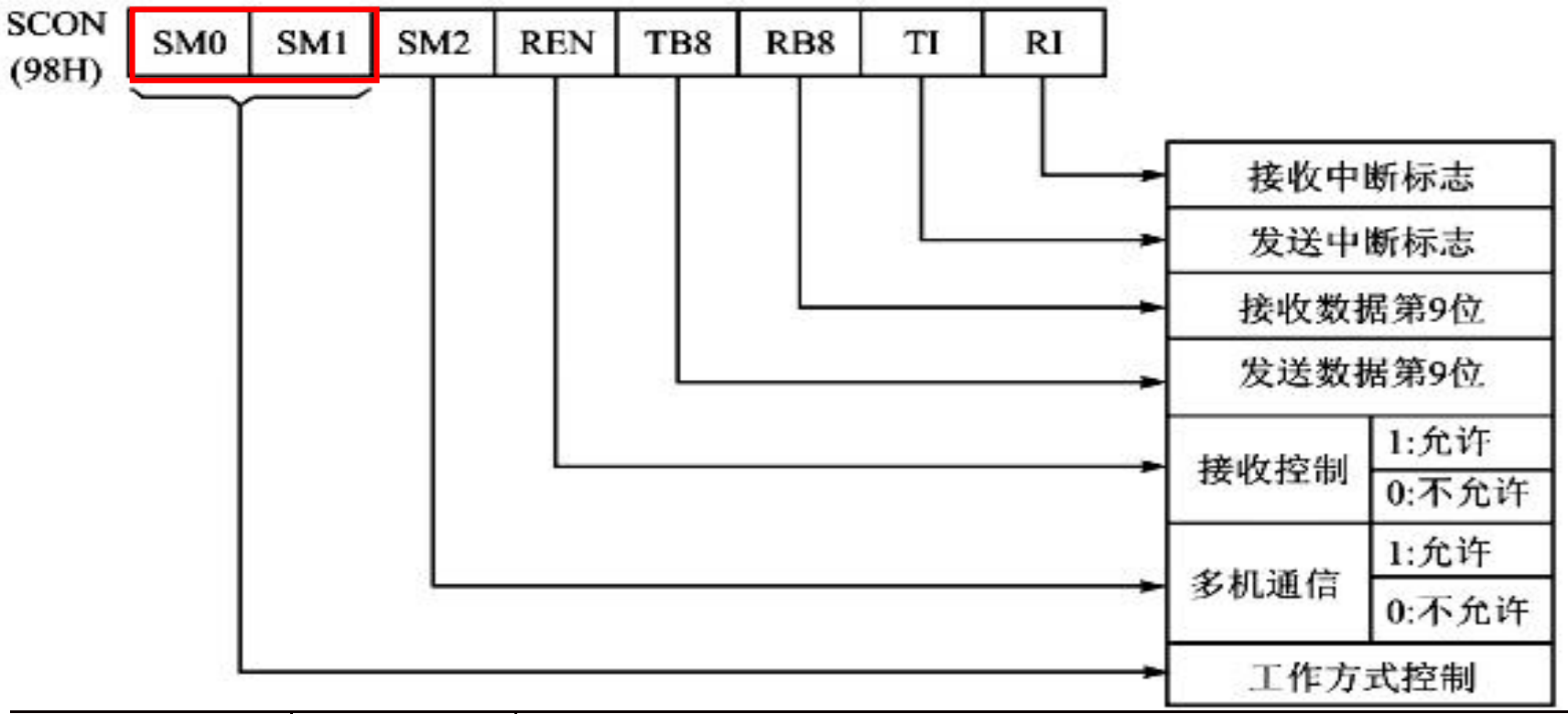
```
while(!RI);
RI=0;
counter=SBUF;
```



- 串行口的接收是被动的
- 接收控制器在输入移位寄存器和定时器T的配合下，当接收到来自RXD引脚的数据，将串行数据——>并行数据，并自动过滤掉起始位、可编程位、停止位
- 转换结束后自动使接收中断请求标志位RI置1，用以通知CPU接收的数据已存入接收缓冲器SBUF。因此当读到RI为1后，执行MOV A,SBUF指令，完成一次数据的接收，使用CPU将SBUF中的数据取走。

RXD（P3.0）用于串行信号的传入。

串行口特殊功能寄存器SCON（98H）



SM0 SM1	方　式	功　能　说　明
0　0	0	同步移位寄存器方式（用于扩展I/O口）
0　1	1	10位异步收发，波特率可变（由定时器控制）
1　0	2	11位异步收发，波特率为$f_{osc}/64$或$f_{osc}/32$
1　1	3	11位异步收发，波特率可变（由定时器控制）

SCON (98H)	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

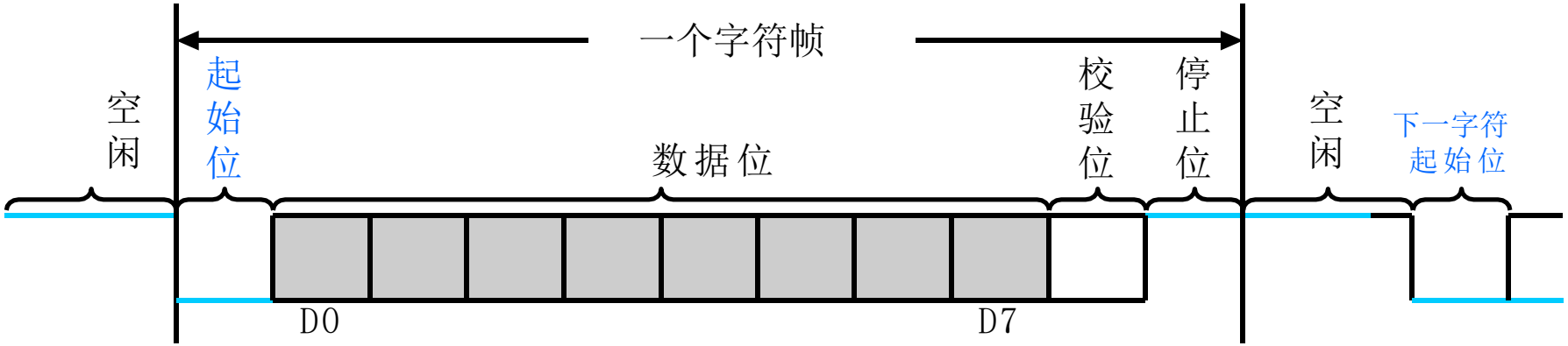
- REN：允许串行接收位，由软件置1或清0。
  REN=1　允许串口接收　REN=0　禁止串口接收

- TI：　发送中断标志位，必须由软件清“0”
  TI=1，表示一帧数据发送结束，由硬件置1。可供软件查询也可申请中断。

- RI：　接收中断标志位，必须由软件清“0”
  RI=1，表示一帧数据接收完毕，由硬件置1。可供软件查询也可申请中断。

TI或RI为0才表示发送或接收缓冲器的数已被取走，可以传送下一帧数据。

波特率的概念



波特率：每秒钟传送码元数目，即每秒发送二进制数码的位数。

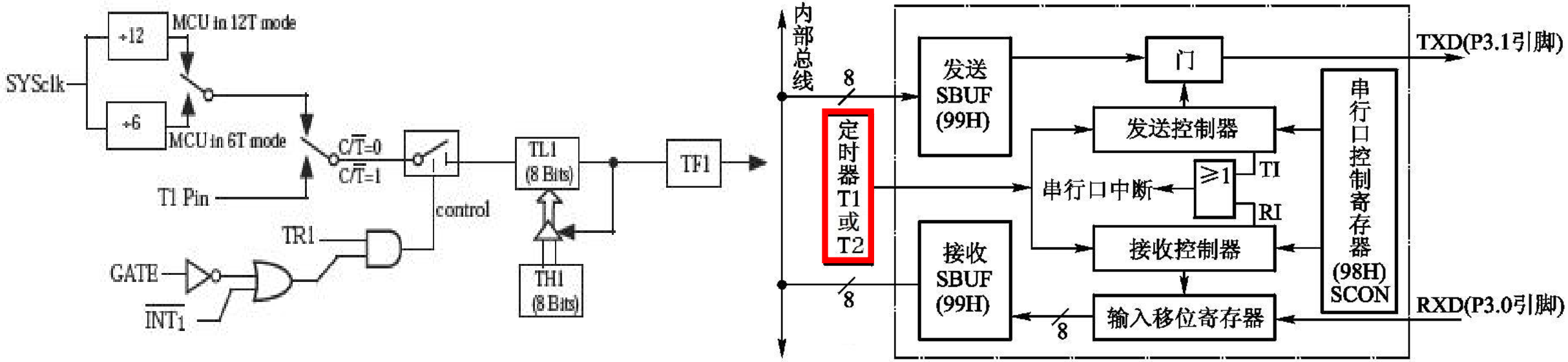
单位：波特Baud， 位/秒bps。

常用波特率为：2400、4800、 9600、14.4K、19.2K……等

- 波特率是表征串行通信数据传输快慢的，传输距离随波特率的增加而减小。

串行通信中，收发双方必须采用相同的波特率通讯。

波特率的产生与计算



STC89C52串行通信以**定时器T1或定时器T2**作为波特率信号发生器，其溢出脉冲经过分频单元后送到收、发控制器中。

波特率与系统时钟频率f_{CLK}有关。以传统单片机的12T时钟计算

1. 串口方式0的波特率

$$串口方式0波特率=\frac{1}{12}\times f_{osc}$$

2. 串口方式2的波特率

$$串口方式2波特率=\frac{2^{SMOD}}{64}\times f_{osc}$$

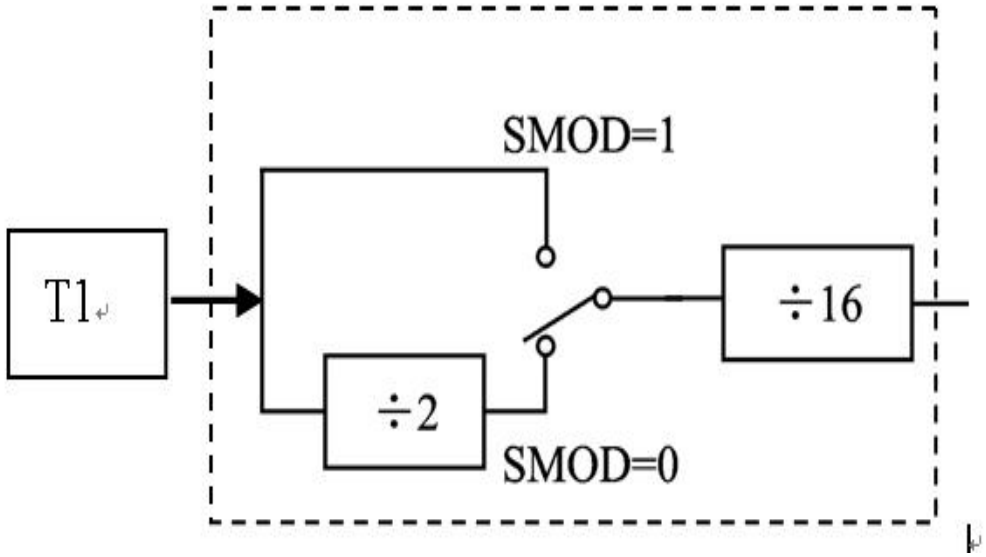
3. 串口方式1和方式3的波特率

$$串口方式1/3波特率=\frac{2^{SMOD}}{32}\times 定时器的溢出率$$

电源控制寄存器PCON

	D7	D6	D5	D4	D3	D2	D1	D0
PCON	**SMOD**	SMOD0	-	POF	GF1	GF0	PD	IDL

- **SMOD：波特率倍增位。**
  当SMOD = 1时，要比SMOD = 0时的波特率加倍。



传统12T模式下，串口方式1/3波特率的定时常数的计算

- 串口方式1和方式3的波特率

$$串口方式1/3波特率=\frac{2^{SMOD}}{32}\times 定时器的溢出率$$

- 设计数初值为X, 那么每经过“$2^n - X$”个机器周期，定时器产生一次溢出，则溢出周期=$T_{机}\times(2^n-X)$，$T_{机}=12 * 1/f_{osc}$，故：

$$定时器溢出率=\frac{f_{osc}}{12\times(2^n-X)}$$

- 定时器不同工作方式计数位数n不同，避免因重装初值带来的定时误差，定时器通常选用方式2产生波特率。

$$定时器方式2的溢出率=\frac{f_{osc}}{12\times(2^8-X)}$$

$$串行口方式1/3波特率=\frac{2^{SMOD}}{32}\times 定时器方式2的溢出率=\frac{2^{SMOD}}{32}\times\frac{f_{osc}}{12\times(2^8-X)}$$

解出此时定时器初值为：

$$X=256-\frac{f_{osc}\times(SMOD+1)}{384\times 波特率}$$

实际使用时，已知波特率和时钟频率，求定时器的初值

表 7-2　　用定时器 T1 产生的常用波特率

波 特 率	f_{CLK}=12MHz		f_{CLK}=11.0592 MHz	
	SMOD	TH1/TL1	SMOD	TH1/TL1
19.2 kbps	1	FCH	1	FDH
9.6 kbps	1	F9H	0	FDH
4.8 kbps	1	F3H	0	FAH
2.4 kbps	0	F3H	0	F4H
1.2 kbps	0	E6H	0	E8H

时钟频率为12MHz时，计算出的波特率有一定误差。为减小波特率误差，可使用的时钟频率为11.0592MHz，此时定时初值为整数。

时钟频率为11.0592MHz，用于系统定时服务不理想。例如：单片机外接11.0592MHz晶振时，$T_{机器}$=12/11.0592MHz≈1.085us，是无限循环小数。

故：定时器用于产生波特率时用11.0592MHz，用于实现精确定时则用12MHz。

【例7-4】STC89C52系统时钟频率f_{OSC}为11.0592MHz，工作在12T模式，串口工作于方式1，用T1定时器工作在方式2作为波特率发生器，波特率为9600波特，求初值。

解：取SMOD=0。

将已知条件带入公式 $X = 256 - \frac{f_{osc} \times (SMOD + 1)}{384 \times 波特率}$

解得X=FDH。另查表也可得。

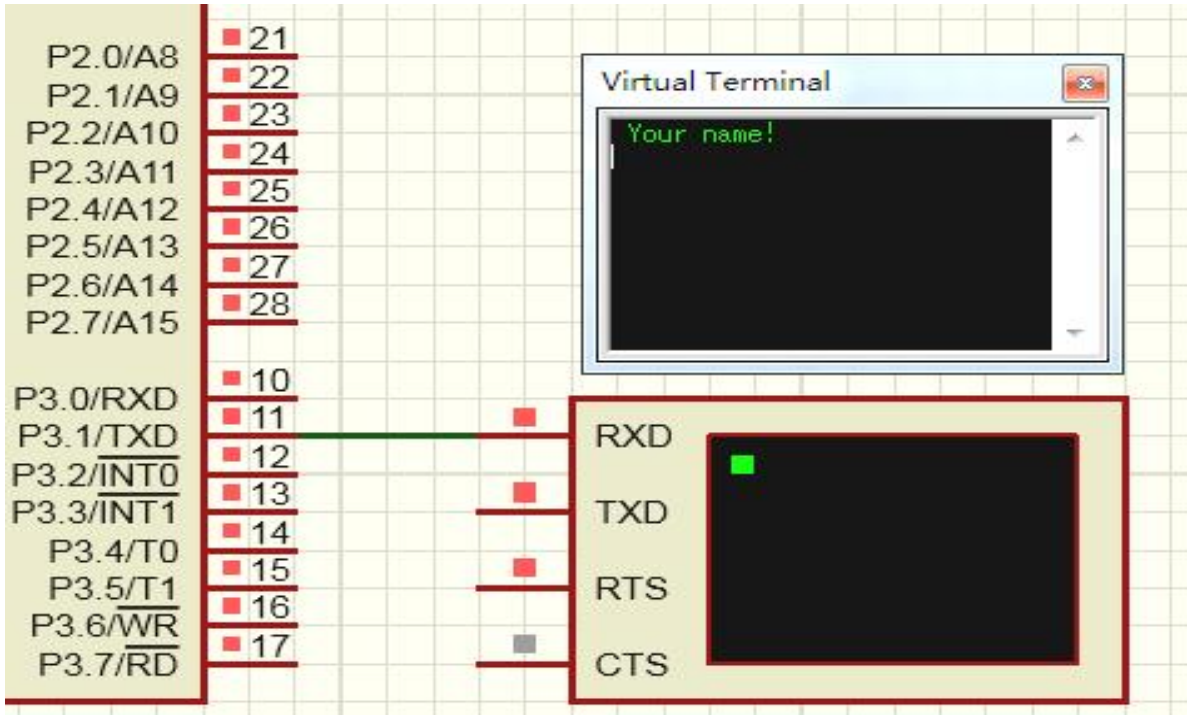
串口的初始化步骤：

- 置串行口工作方式——SCON
- 置波特率倍增位———PCON
- 置定时器T1工作方式——TMOD
- 置定时初值————————TH1,TL1
- 启动定时器————————TR1
- 用中断则开放中断———EA,ES

```
#include <reg52.h>
#include <stdio.h>

void UartInit(void)
{
  SCON=0x40;              //串口工作于方式1发送
  TMOD=0x20;              //T1工作于方式2用于产生波特率
  TH1=0xFD;               //晶振为11.0592MHz,波特率为9.6KHz
  TL1=0xFD;
  TR1=1;                  //启动定时器
}
```

```
MOV   SCON,#40H ；设串口工作方式
MOV   PCON,#00H ； 可省略
MOV   TMOD,#20H ； 设T1工作方式
MOV   TH1,#0FDH ；设T1定时初值
MOV   TL1,#0FDH
SETB   TR1              ；启动T1
```



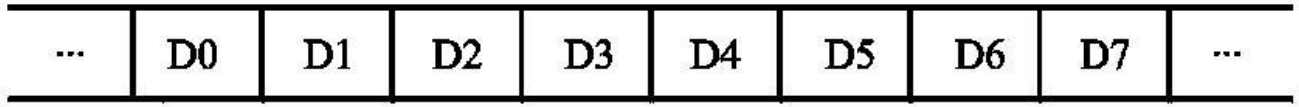

```c
#include <reg52.h>
#include <stdio.h>

void UartInit(void)
{
    SCON=0x40;              //串口工作于方式1发送
    TMOD=0x20;              //T1工作于方式2用于产生波特率
    TH1=0xFD;               //晶振为11.0592MHz,波特率为9.6KHz
    TL1=0xFD;
    TR1=1;                  //启动定时器
}

void main(void)
{
    UartInit();
    TI=1;                   //表示一帧数据发送结束
    printf(" My name!\n"); //把输出内容改为你姓名的全拼
    TI=0;
    while(1);
}
```

串行口的4种工作方式

- **方式0——8位同步移位寄存器工作方式** 常用于串并转换扩展并行口通信



- 一帧信息：8位数据，无起始和停止位， 从最低位开始。
- RXD：数据输入或输出。TXD：移位脉冲输出。
- 默认在12T模式下波特率固定：$f_{CLK}/12$， 唯一的同步通信方式。

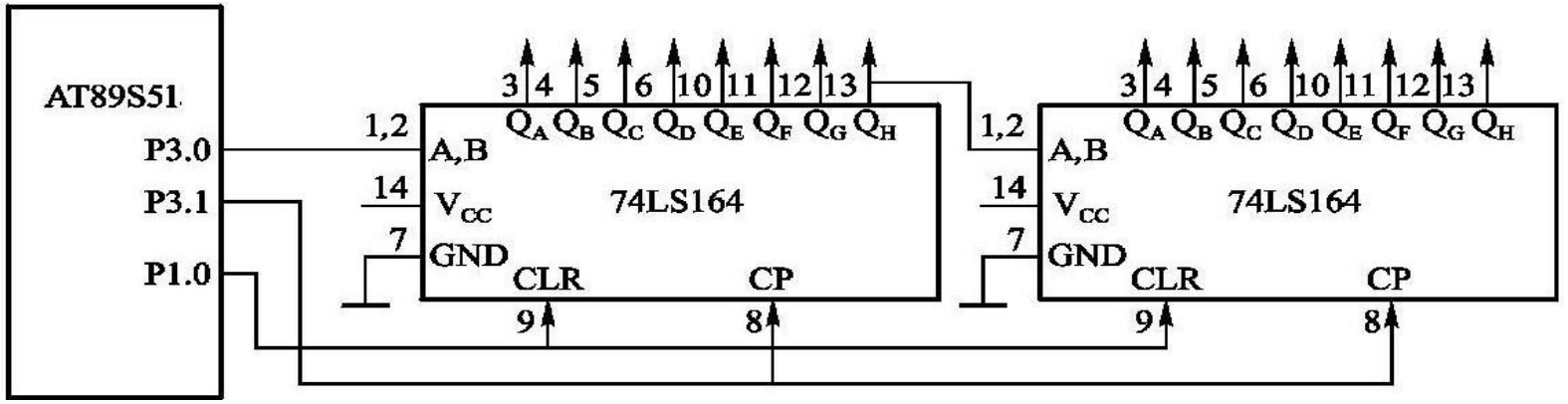
方式0主要用于当并行口数量不足时，采用移位寄存器进行**串并转换**扩展，常用移位寄存器有：74LS164——8位串入并出移位寄存器，74LS165——8位并入串出移位寄存器。

该扩展方法

- 节省硬件开销，只需增加1根I/O口线
- 扩展的移位寄存器芯片越多，操作速度越慢
- 不占用片外RAM存储空间的地址

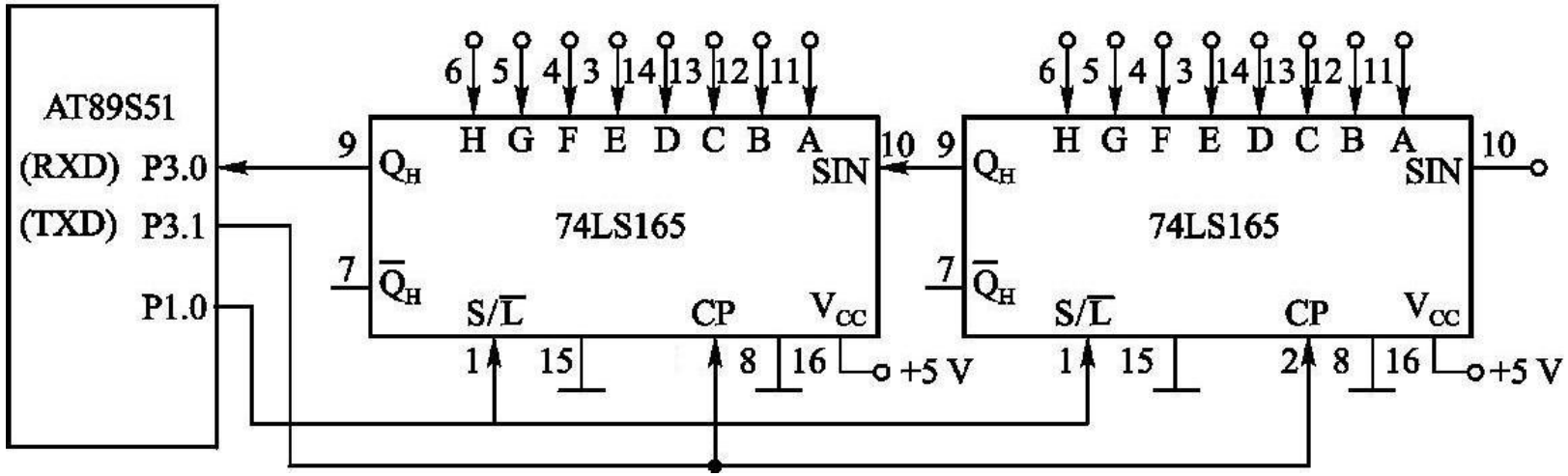
利用74LS164串入并出移位寄存器 实现串行口扩展多个并行输出口：

P3.0（RXD）:串行输出数据　P3.1 （TXD）:同步移位脉冲



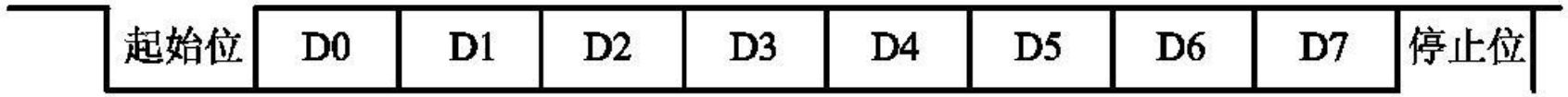
利用74LS165并入串出移位寄存器实现串行口扩展多个并行输入口：

P3.0（RXD）:串行输入数据　P3.1 （TXD）:同步移位脉冲



串行口的4种工作方式

- **方式1—— 10位异步通信模式** 常用于点对点双机通信

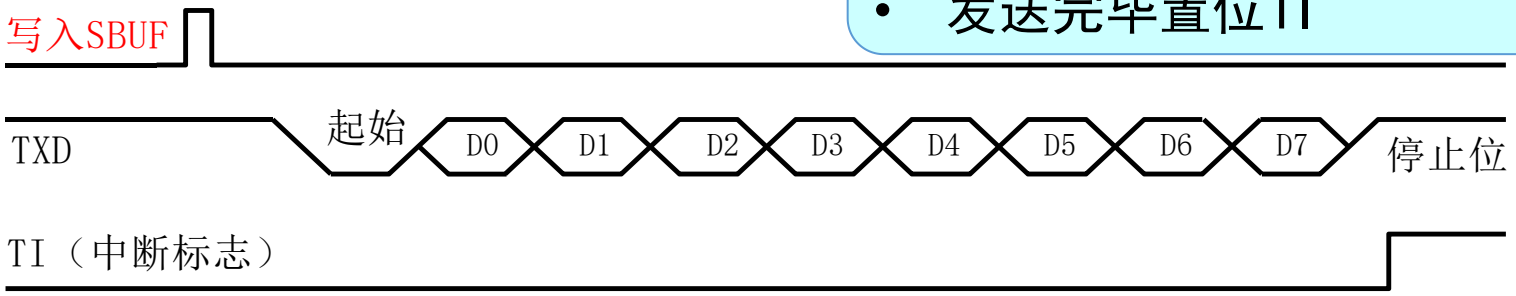


- 一帧信息：1个起始位(0)+8位数据位+1个停止位(1)
- TXD：发送数据，RXD：接收数据
- 波特率可变

方式1

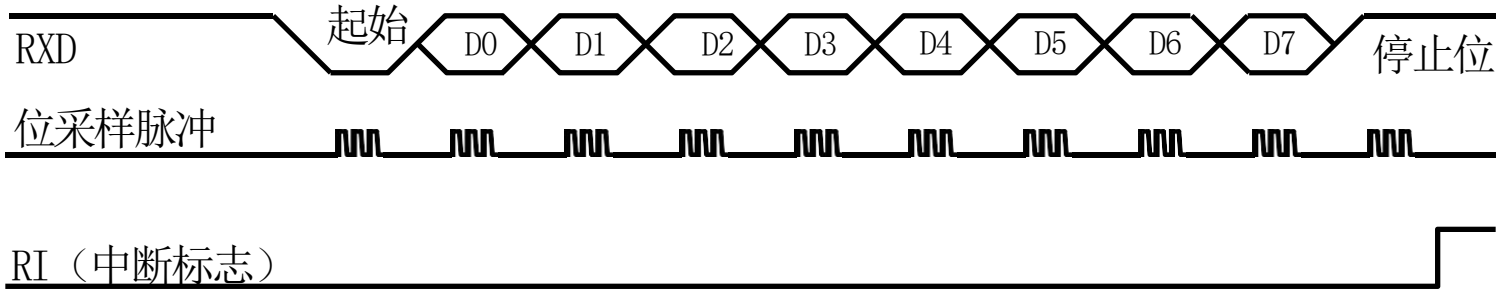
发送：（写SBUF启动发送）

- MOV SBUF,A指令启动发送
- TXD引脚：输出一帧10位数据
- 发送完毕置位TI



接收：(置REN=1启动接收)

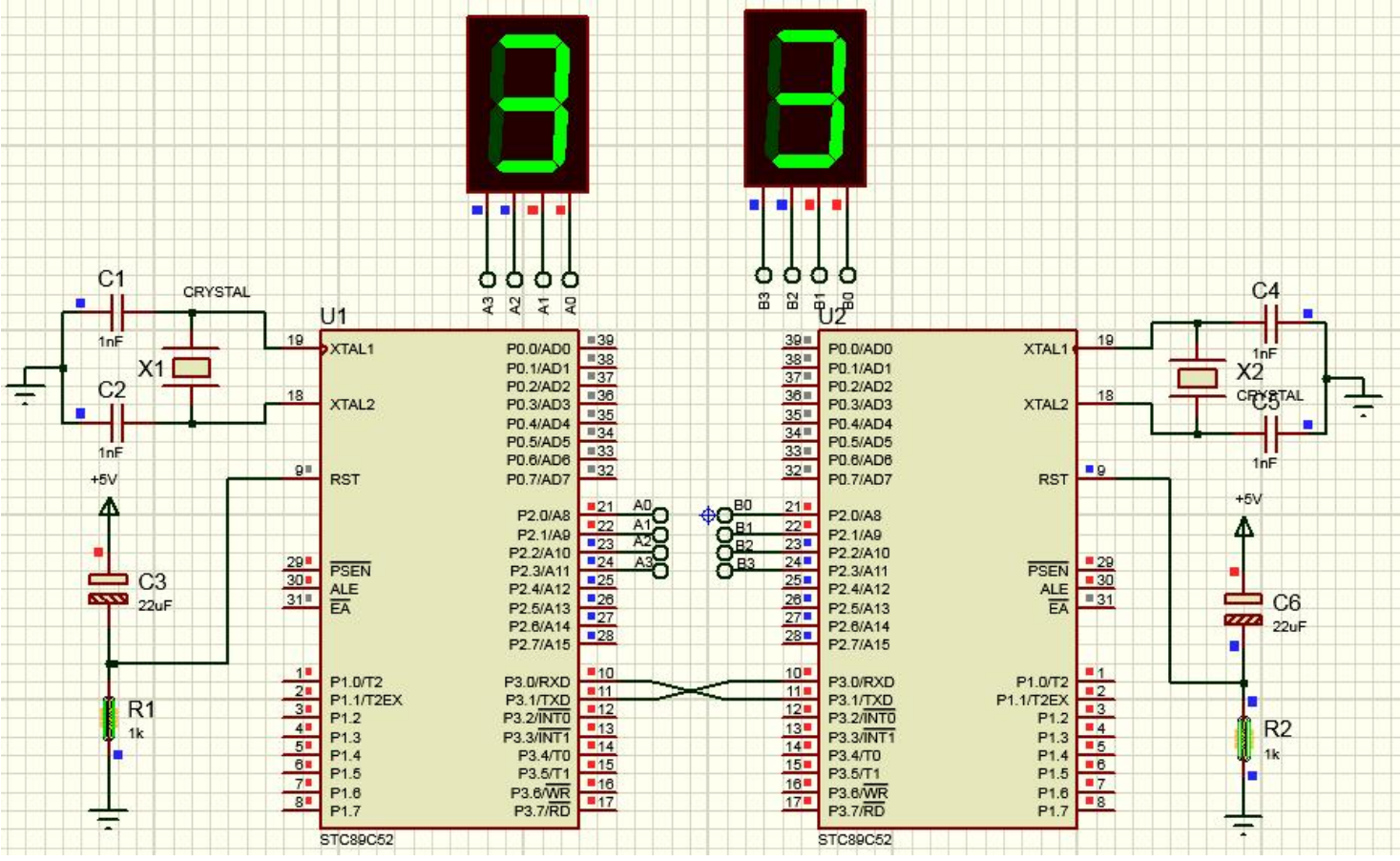
- 设置SCON使REN=1启动接收
- 位检测器以16倍速检测到RXD引脚负跳变时启动，3次连续采样取两次相同的值，以保证数据的准确性
- RXD引脚：输入一帧10位数据
- 接收完毕置位RI



单片机与单片机双机串行通信

【例7-6】串口以方式1进行异步通信，如下图甲机发送，乙机接收。双方晶振频率为11.0592MHz，通信波特率为2400bps。甲机循环发送数字0～F，乙机接收后返回接收值。若发送值与返回值相等，则继续发送下一数字，否则需重发当前数字。

数码管（7SEG-BCD）——带译码电路可直接输入数据0～F，无需显示字模）。



甲机发送程序思路

分析：① 串口、定时器初始化：

- 串口工作在方式1，允许接收，清中断标志 **SCON=0x50**
- **PCON=0即SMOD=0，可省略**
- **T1采用方式2产生波特率 TMOD=0x20**
- 产生**2400**波特率，查表与计算均得**TH1=TL1=0xf4;**

f=11.0592MHz		
波特率	SMOD	a
62500	1	0xff
19200	1	0xfd
9600	0	0xfd
4800	0	0xfa
2400	**0**	**0xf4**
1200	0	0xe8

串口的初始化步骤：

- 置串行口工作方式——SCON
- 置波特率倍增位———PCON
- 置T1工作方式————TMOD
- 置定时初值————TH1,TL1
- 启动定时器————TR1
- 用中断则开放中断——EA,ES

```
MAIN:  MOV   SCON,#50H
       MOV   PCON,#00H
       MOV   TMOD,#20H
       MOV   TH1,#0F4H
       MOV   TL1,#0F4H
       SETB  TR1
```

```
void initUart(void){
   SCON = 0x50;  //串口方式1，接收允许
   PCON=0;       //波特率不加倍
   TMOD=0x20;    //定时器T1方式2
   TH1=0xF4;
   TL1=0xF4;
   TR1=1;        //启动T1产生波特率
   }
```

甲机发送程序

```
#include<reg52.h>
#define uchar unsigned char
void time(unsigned int ucMs);
void initUart(void);
void main(void){
 uchar counter=0;
 time(1);
 initUart();
 while(1){
     SBUF=counter;
     while(TI==0);
     TI = 0;
     while(RI==0);
     RI = 0;
     if(SBUF==counter){
      P2 = counter;
      if(++counter>15) counter=0;
      time(500);
      }
  }
}
```

```
void time(unsigned int ucMs){ //延时
   #define DELAYTIMES 239
   unsigned char  ucCounter;  // 循环次数
   while (ucMs!=0) {
       for (ucCounter=0; ucCounter<DELAYTIMES; ucCounter++){}
       ucMs--;
    }
}
```

//延时1ms等待外围器件复位
//初始化串口

//启动发送
//等待发送完成
//清TI标志位
//等待乙机回答
//清RI标志位
//判断接收数据是否等于发送值
//显示已发送值
//修正计数器，超过15从0重新开始

甲机发送程序 改为汇编代码

```
#include<reg52.h>
#define uchar unsigned char
void time(unsigned int ucMs);
void initUart(void);
void main(void){
 uchar counter=0;
 time(1);            //延时1ms等待外围器件复位
 initUart();         //初始化串口
 while(1){
     SBUF=counter;          //启动发送
     while(TI==0);          //等待发送完成
     TI = 0;                //清TI标志位
     while(RI==0);          //等待乙机回答
     RI = 0;                //清RI标志位
   if(SBUF==counter){       接收数据是否等于发送值
      P2 = counter;                    //显示已发送值
      if(++counter>15) counter=0; //修正计数器
      time(500);
          }
     }
}
```

```
        ORG   0000H
        AJMP  MAIN
        ORG   0030H
MAIN:   MOV   SCON,#50H
        MOV   PCON,#00H
        MOV   TMOD,#20H
        MOV   TH1,#0F4H
        MOV   TL1,#0F4H
        SETB  TR1
        MOV   30H,#00H
LOOP1:  MOV   A,30H
        MOV   SBUF,A
HERE1:  JNB   TI,HERE1
        CLR   TI
HERE2:  JNB   RI,HERE2
        CLR   RI
        MOV   A,SBUF
HERE3:CJNE    A,30H,HERE3
        MOV   P2,A
        ACALL DELAY
        INC   30H
        AJMP  LOOP1
DELAY:  MOV   R3,#80H
DELAY1:MOV    R4,#0FFH
DELAY2:DJNZ   R4,DELAY2
        DJNZ  R3,DELAY1
        RET
        END
```

乙机接收程序

```
#include<reg52.h>
#define uchar unsigned char
void time(unsigned int ucMs);
void inituart(void);
void main(void){
    uchar  receive;
    time(1);
    initUart();
    while(1){
            while(RI==1){
            RI=0;
            receive=SBUF;
            SBUF=receive;
            while(TI==0);
            TI=0;
            P2=receive;
             }
       }
}
```

```
void time(unsigned int ucMs){ //延时
    #define DELAYTIMES 239
    unsigned char  ucCounter;  // 循环次数
    while (ucMs!=0) {
        for (ucCounter=0; ucCounter<DELAYTIMES; ucCounter++){}
        ucMs--;
     }
}
```

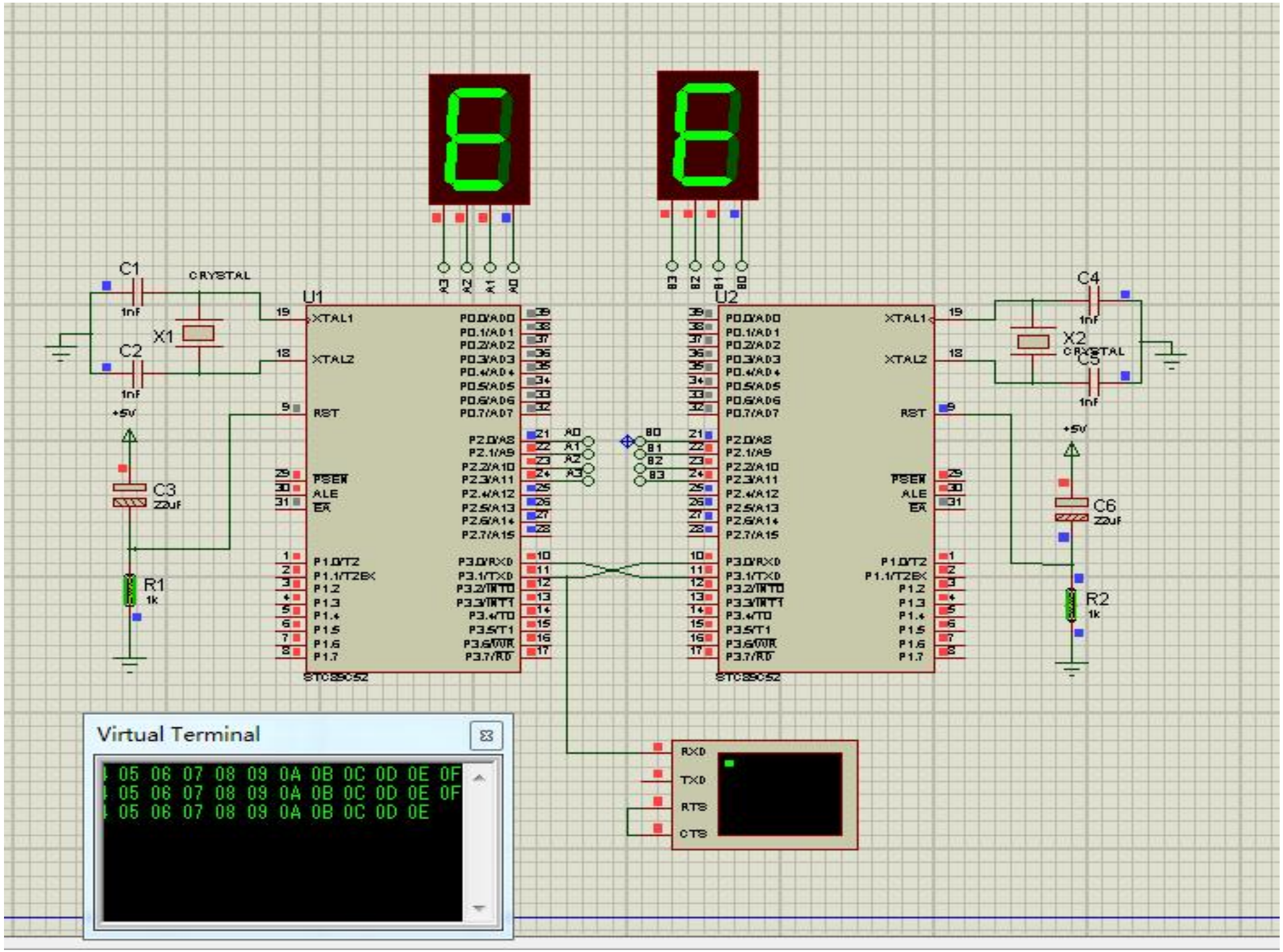
//延时1ms等待外围器件复位
//初始化串口

//先判断接收是否完成
//清RI标志位
//取接收值
//接收值返送给甲机
//等待发送结束
//清TI标志位
//显示接收值

乙机接收程序改为汇编代码

```
#include<reg52.h>
#define uchar unsigned char
void time(unsigned int ucMs);     //延时单位
void inituart(void);              //初始化串口
void main(void){
    uchar  receive;               //定义接收缓冲
    time(1);                      //延时等待外围器件完成复位
    initUart();
    while(1){
            while(RI==1){                 //等待接收完成
            RI=0;                         //清RI
            receive=SBUF;                 //取接收值
            SBUF=receive;                 //结果返送发送缓冲器
            while(TI==0);                 //等待发送结束
            TI=0;                         //清TI
            P2=receive;                   //显示接收值
             }
      }
}
```

```
        ORG   0000H
        AJMP  MAIN
        ORG   0030H
MAIN:   MOV   SCON,#50H
        MOV   PCON,#00H
        MOV   TMOD,#20H
        MOV   TH1,#0F4H
        MOV   TL1,#0F4H
        SETB  TR1
HERE1:  JNB   RI,HERE1
        CLR   RI
        MOV   A,SBUF
        NOP
        MOV   SBUF,A
HERE2:  JNB   TI,HERE2
        CLR   TI
        MOV   P2,A
        ACALL DELAY
        AJMP  HERE1
DELAY:  MOV   R3,#0FFH
DELAY1: MOV   R4,#0FFH
DELAY2: DJNZ  R4,DELAY2
        DJNZ  R3,DELAY1
        RET
        END
```


U1
U2
STC89C52
STC89C52
C1
1nF
CRYSTAL
X1
C2
1nF
C3
22uF
R1
1k
+5V
C4
1nF
X2
CRYSTAL
C5
1nF
C6
22uF
R2
1k
Virtual Terminal
05 06 07 08 09 0A 0B 0C 0D 0E 0F
05 06 07 08 09 0A 0B 0C 0D 0E 0F
05 06 07 08 09 0A 0B 0C 0D 0E
RXD
TXD
RTS
CTS

思考：

汇编程序有没有判断是否大于15？

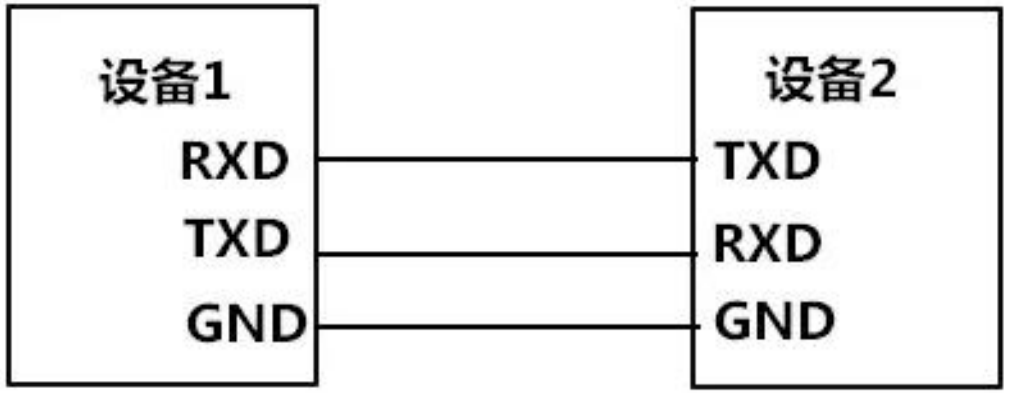
答：显然没有，为什么不用判断？

如果要求只显示0～9，C语言的程序修改简单，只需要将if(++counter>15)中的15改为10即可，那汇编程序要如何修改？

答：汇编程序在循环开始，即发送数据之前先判断是否>=0AH，如果大于则不发送不显示，直接加1进入下一个数，如果<0AH，则发送并显示。

```
        ORG     0000H
        AJMP    MAIN
        ORG     0030H
MAIN:   MOV     SCON,#50H
        MOV     PCON,#00H
        MOV     TMOD,#20H
        MOV     TH1,#0F4H
        MOV     TL1,#0F4H
        SETB    TR1
        MOV     30H,#00H
LOOP1:  MOV     R0,30H
        CJNE    R0,#0AH,CONTU
CONTU:  JNC     NEXT
        MOV     A,30H
        MOV     SBUF,A
HERE1:  JNB     TI,HERE1
        CLR     TI
HERE2:  JNB     RI,HERE2
        CLR     RI
        MOV     A,SBUF
HERE3:  CJNE    A,30H,HERE3
        MOV     P2,A
        ACALL   DELAY
NEXT:   INC     30H
        AJMP    LOOP1
DELAY:  MOV     R3,#80H
DELAY1: MOV     R4,#0FFH
DELAY2: DJNZ    R4,DELAY2
        DJNZ    R3,DELAY1
        RET
        END
```

原生的串口到串口



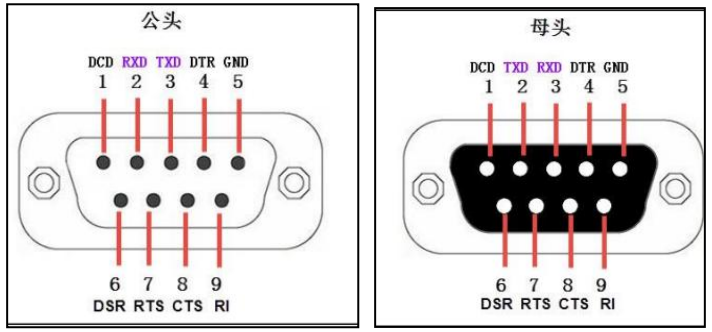
- 原生的串口通信主要是控制器跟串口设备不需要经过电平转换芯片，直接就用TTL电平通信。如单片机与GPS模块、GSM模块、WIFI模块等

两个单片机相距在1.5m之内，可直接用TTL电平实现通信。

标准串行接口

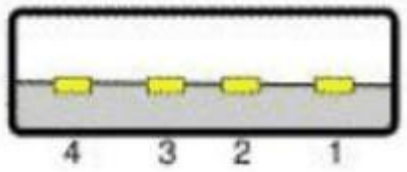
采用标准串行接口提高串行通信的可靠性，增大通信距离以及提高传输速率。





	功能	符号		功能	符号
1	数据载波检测	DCD	6	数据设备准备好	DSR
2	接收数据	RXD	7	请求发送	RTS
3	发送数据	TXD	8	清除发送	CTS
4	数据终端准备	DTR	9	振铃指示	DELL
5	信号地	GND			





Type A

Pin	Signal
1	VCC
2	D-
3	D+
4	GND

- RS232标准串口与TTL电平的转换常用MAX232芯片

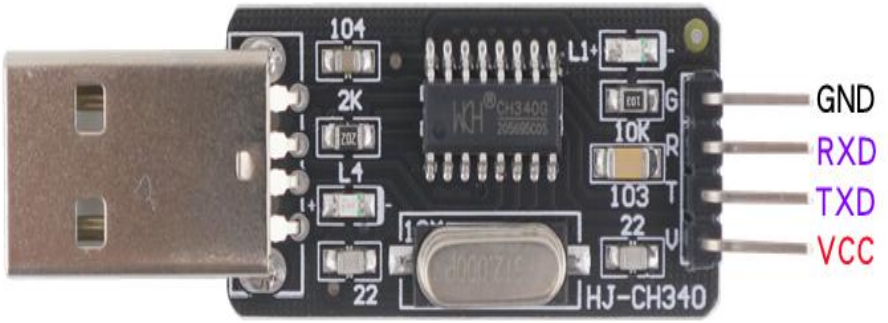


PC机与单片机之间通过RS232串口通信



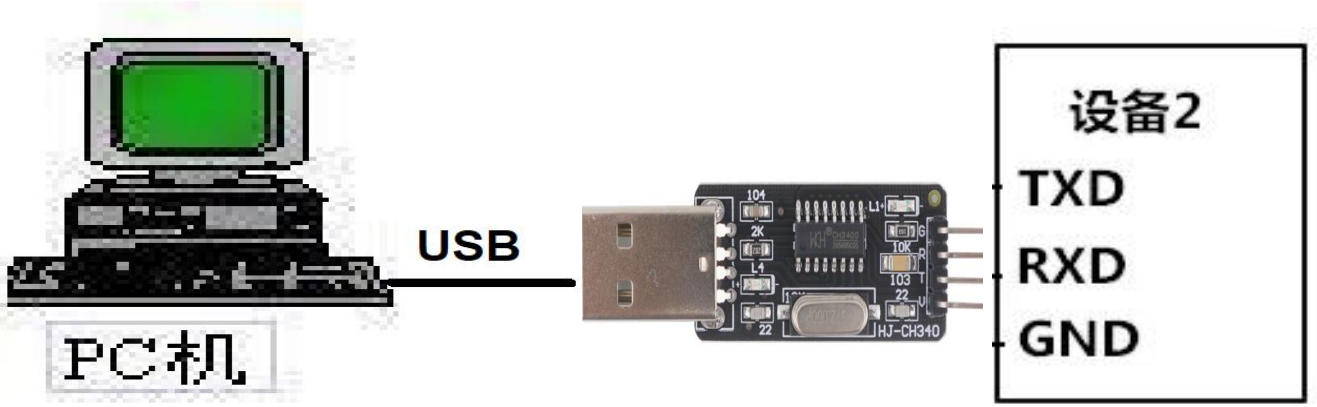
- USB标准串口与TTL电平转换

  常用CH340电路模块



PC机与单片机之间

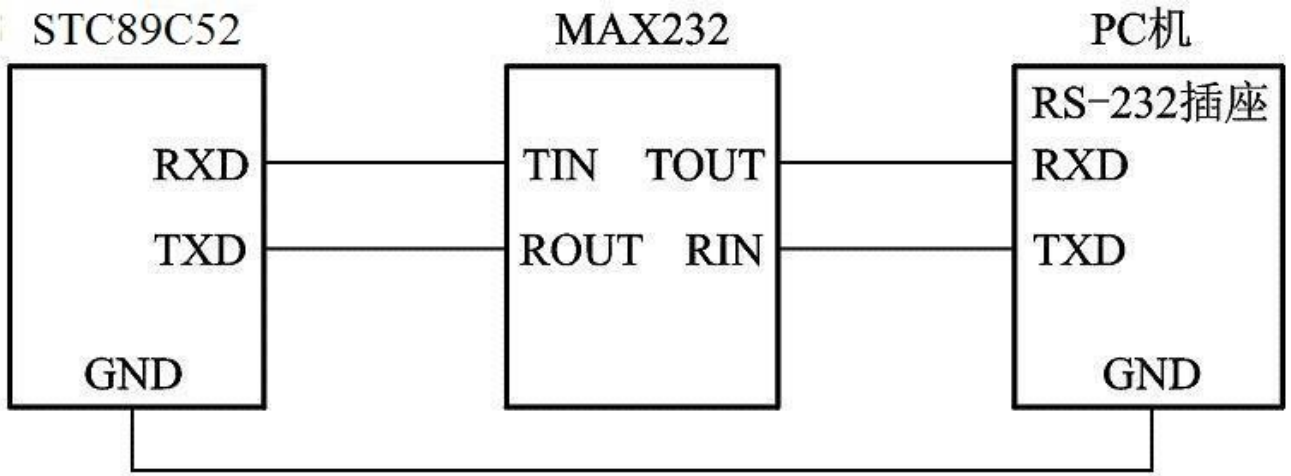
通过USB串口通信



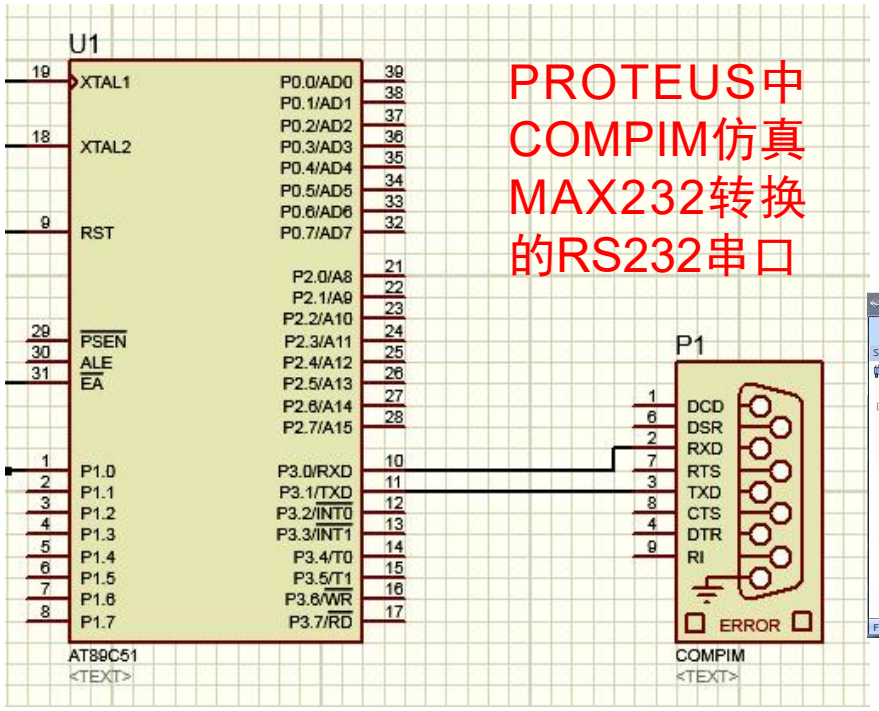
电脑端需要安装CH340驱动

单片机与PC机双机串行通信

单片机（下位机）

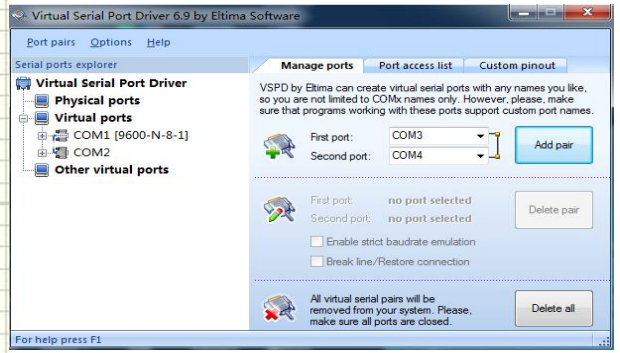


PC（上位机）

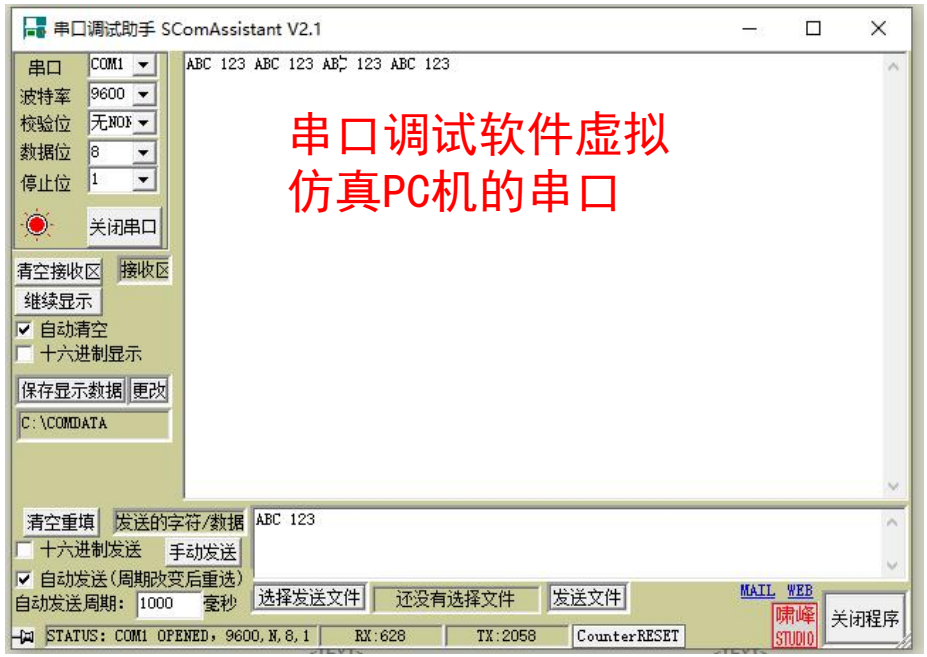


PROTEUS中COMPIM仿真MAX232转换的RS232串口

由虚拟串口配置软件将单片机与PC机的串口配对，实现仿真连接



串口调试软件虚拟仿真PC机的串口

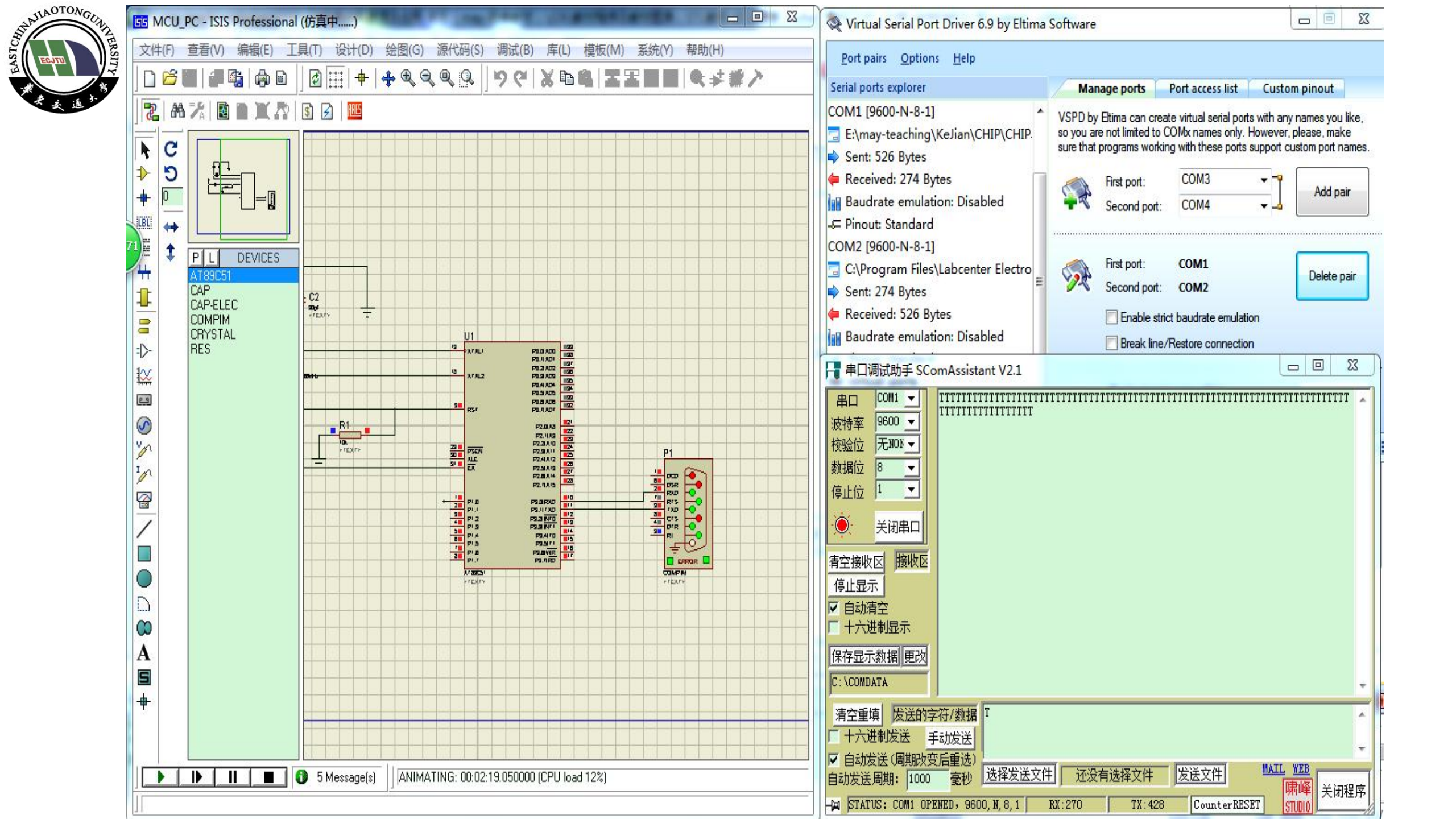


;先通过串口工具即PC发送字符给单片机，单片机接收数据后再发给PC， 再通过串口工具观察传送结果
;由于本地仿真，需要安装虚拟串口，PC用COM1，MCU的COMPIM设为COM2

```
#include <reg52.h>
#define uchar  unsigned char
#define uint  unsigned int
void Uart_Init(void)
{               SCON=0x50;    //串口工作于方式1发送
                TMOD|=0x20;   //T1工作于方式2产生波特率
                TH1=0xFD;     //设定时初值
                TL1=0xFD;
                TR1=1;        //启动定时器
                ES=1;         //开中断
                EA=1;
                TI=1;         //表示一帧数据发送结束
}
void Uart_Isr(void)   interrupt 4
{               uchar Recv;
                while(!RI);             //判断接收
                RI=0;
                Recv=SBUF;              //取接收值
                SBUF=Recv;              //启动发送
                while(!TI);
                TI=0;
}
void main(void)
{               Uart_Init();
                while(1);               //等待接收中断
}
```

```
                org     0000h
                ajmp    main
                org     0023h
                ajmp    aa
                org     0030h
main:           mov     scon,#50h
                mov     tmod,#20h
                mov     pcon,#00h
                mov     th1,#0fdh
                mov     tl1,#0fdh
                setb    ea
                setb    es
                setb    tr1
                CLR     F0          ;F0=0接收
                mov     a,sbuf      ;启动串口接收
                sjmp    $           ;等待串口中断

aa:             JB      F0,TRS      ;若F0=0接收，若F0=1转发送
                SETB    F0          ;进入接收将F0置1为发送准备
                CLR     RI          ;进入接收中断清RI标志
                MOV     scon,#40H ;设置串口方式1发送
                MOV     sbuf,a      ;启动串口发送
                RETI
TRS:            CLR     TI          ;进入发送中断清TI标志
                CLR     F0          ;进入发送将F0清0为接收准备
                MOV     scon,#50H;设置串口方式1接收
                mov     a,sbuf      ;启动串口接收
                RETI
                end
```


MCU_PC - ISIS Professional (仿真中.....)
文件(F) 查看(V) 编辑(E) 工具(T) 设计(D) 绘图(G) 源代码(S) 调试(B) 库(L) 模板(M) 系统(Y) 帮助(H)
DEVICES
AT89C51
CAP
CAP-ELEC
COMPIM
CRYSTAL
RES
U1
R1
C2
P1
5 Message(s)
ANIMATING: 00:02:19.050000 (CPU load 12%)
Virtual Serial Port Driver 6.9 by Eltima Software
Port pairs Options Help
Serial ports explorer
COM1 [9600-N-8-1]
E:\may-teaching\KeJian\CHIP\CHIP.
Sent: 526 Bytes
Received: 274 Bytes
Baudrate emulation: Disabled
Pinout: Standard
COM2 [9600-N-8-1]
C:\Program Files\Labcenter Electro
Sent: 274 Bytes
Received: 526 Bytes
Baudrate emulation: Disabled
Manage ports
Port access list
Custom pinout
VSPD by Eltima can create virtual serial ports with any names you like, so you are not limited to COMx names only. However, please, make sure that programs working with these ports support custom port names.
First port: COM3
Second port: COM4
Add pair
First port: COM1
Second port: COM2
Delete pair
Enable strict baudrate emulation
Break line/Restore connection
串口调试助手 SComAssistant V2.1
串口 COM1
波特率 9600
校验位 无NON
数据位 8
停止位 1
关闭串口
清空接收区
接收区
停止显示
自动清空
十六进制显示
保存显示数据
更改
C:\COMDATA
清空重填
发送的字符/数据
T
十六进制发送
手动发送
自动发送(周期改变后重选)
自动发送周期: 1000 毫秒
选择发送文件
还没有选择文件
发送文件
MAIL WEB
关闭程序
STATUS: COM1 OPENED, 9600, N, 8, 1
RX:270
TX:428
CounterRESET

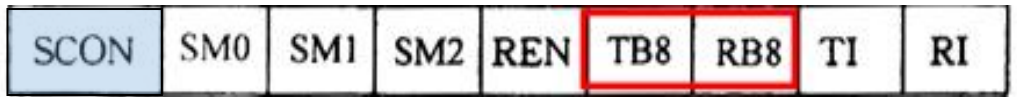
串行口的4种工作方式

- 方式2和方式3——11位数据异步通信方式　　常用于多机通信



➢ 方式2和方式3不同点：
- 方式2的波特率是固定的；
- 方式3的波特率由定时器T的溢出率决定。

➢方式2和方式3共同点：一帧信息都是11位数据：
- 1位起始位、8位数据位、1位TB8或1位RB8第9位数据、1位停止位



第9位数据D8是硬件插入数据帧，发送时是SCON中的TB8，接收时是SCON中的RB8

可作为多机通信中地址帧或数据帧的标识位（如地址帧可约定为1，数据帧时为0），也可作为数据的奇偶校验位。

SCON (98H)	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

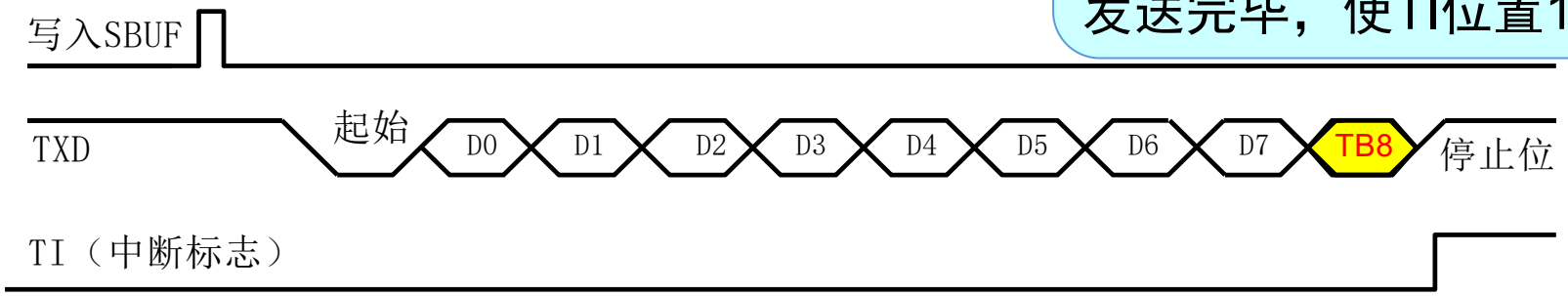
- **SM2—用于方式2或方式3多机通信控制位**
  - –当SM2=1时：
    - •若收到的RB8=1，则使RI=1，前8位数据有效，送入SBUF。
    - •若收到的RB8=0，则不会使RI=1,前8位数据无效，丢弃。
  - –当SM2=0时：
    - •无论收到的RB8=0或1，都会使RI=1，前8位数据有效，送到SBUF中。

- TB8:发送的第9位数据。
  - – 方式2和方式3，TB8是要发送的第9位数据，由软件置1或清0。
    - 在多机串行通信中用来表示主机发送的是地址帧还是数据帧，TB8=1为地址帧，TB8=0为数据帧。（双机通信时，一般作为奇偶校验位使用）
- RB8:接收的第9位数据 。
  - – 方式2和方式3，RB8存放接收到的第9位数据。

方式2和方式3

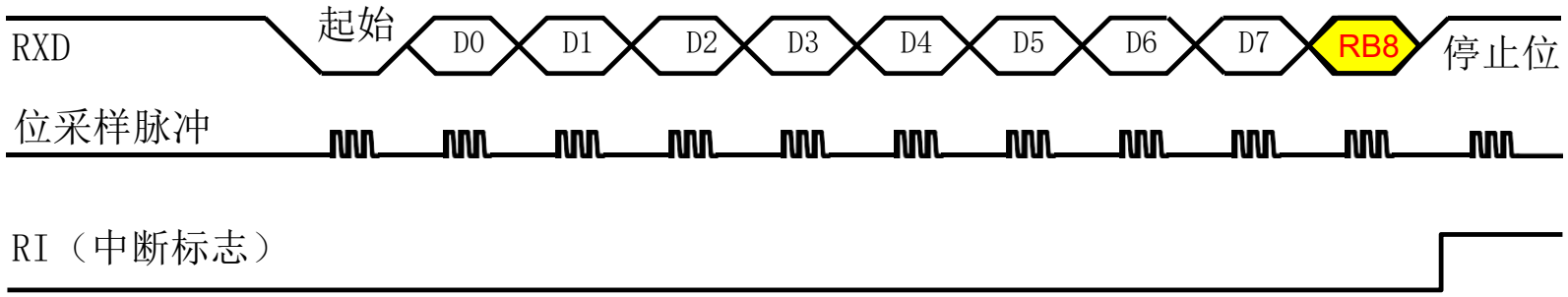
串行发送：（写SBUF启动发送）

先设置SCON的TB8位
再执行MOV SBUF, A启动发送
　　TB8自动装入数据帧第9位一并发送
发送完毕，使TI位置1

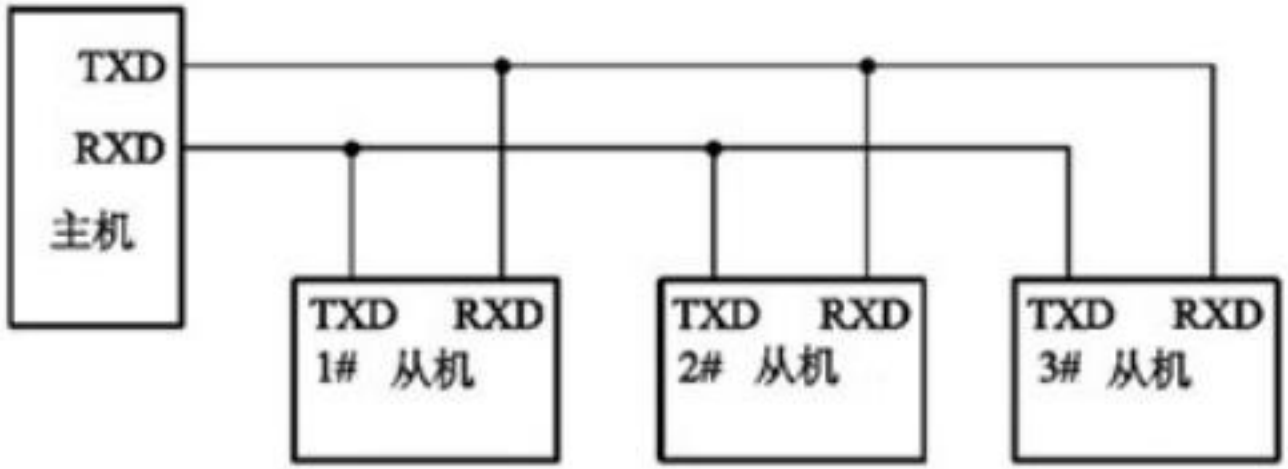


串行接收：(置REN=1启动接收)

- 设置SCON使REN=1启动接收
- 位检测器同方式1连续采样，使RXD接收数据.
- 当RB8=1时，不论SM2=0或SM2=1，接收到的信息会自动装入SBUF，置RI=1。
- 当RB8=0时，且SM2=1时，数据丢弃不被接收，不置位RI。



方式2和方式3的应用——多机通信



- 多机通信，常采用主从式结构。主机与各从机间能实现双向通信，各从机之间不能直接通信。
- 主机的RXD与所有从机的TXD相连，主机的TXD与所有从机的RXD相连。
- 每个从机都被赋予唯一的地址。一般还要预留1~2个”广播地址”即所有从机共有的地址。

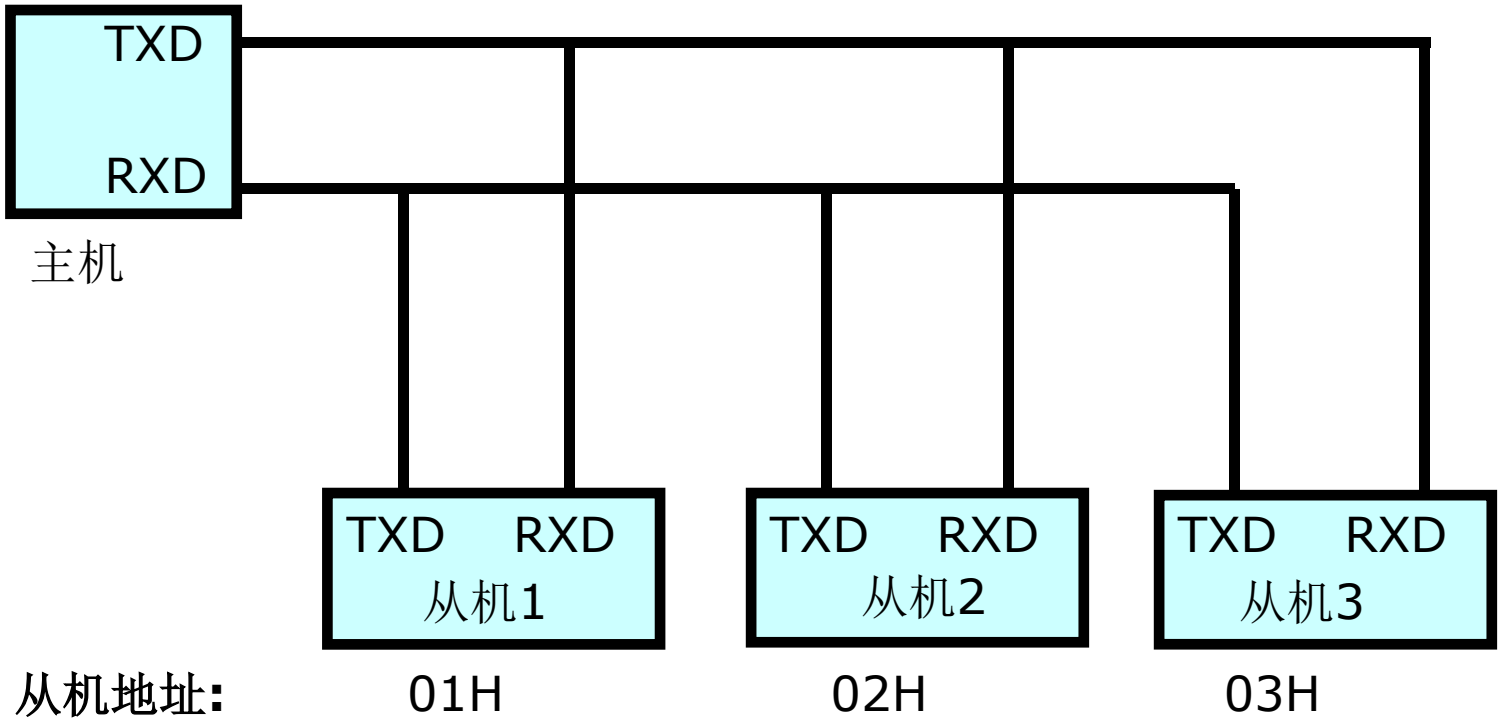
多机通信过程

主机第1次发出的地址信息要能被所有的从机响应，

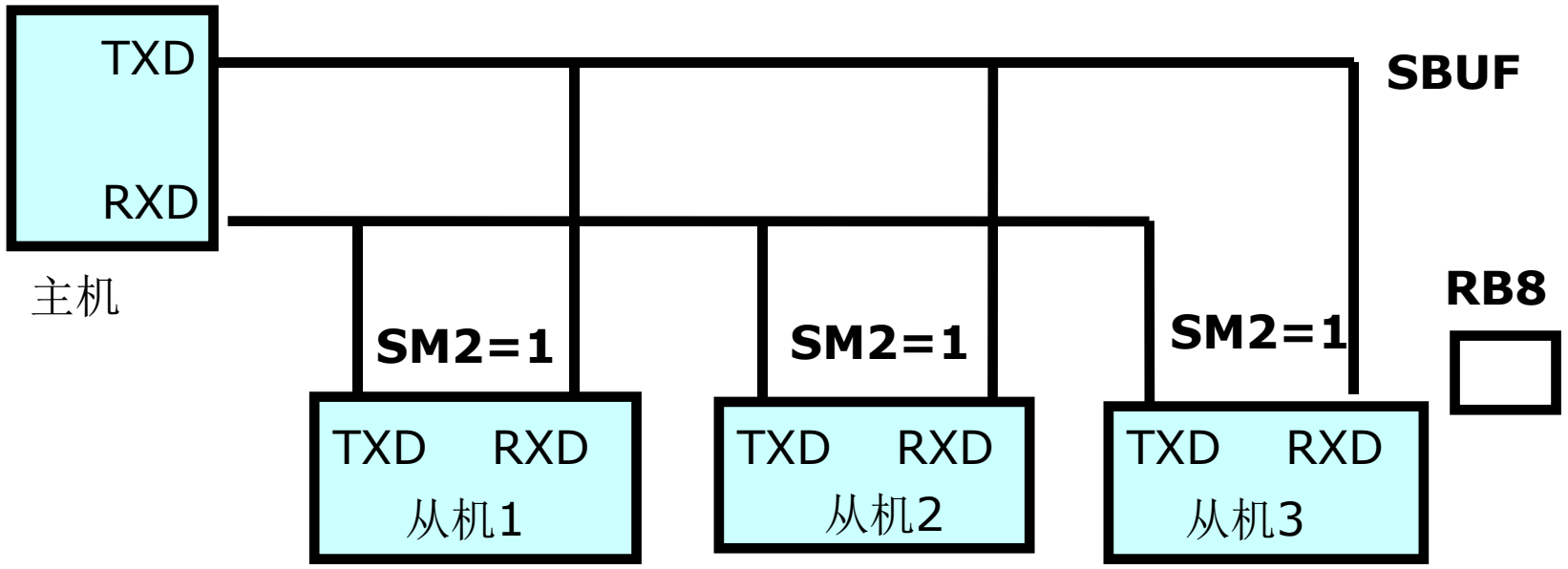
而第2次的数据信息只能被第n#从机响应

发送的第9位数据TB8作为区分地址/数据的标识。

TB8=1表示地址，TB8=0表示数据。



SM2=1 ≺ RB8=1,数据有效装入SBUF，RI=1可中断申请
RB8=0,数据无效丢弃，RI=0不发中断申请
SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请



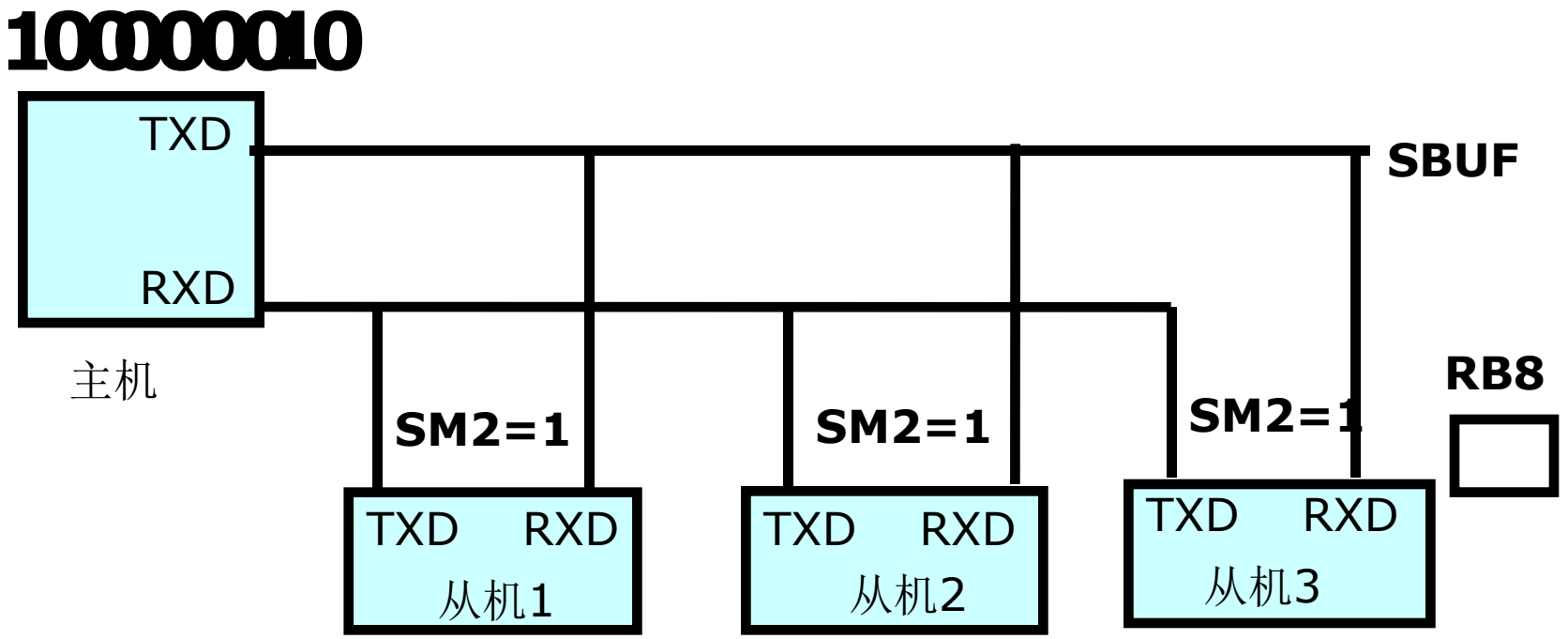
1）各从机初始化:MOV SCON,#0B0H;或#0F0H

(方式2或方式3，SM2和REN均为1）

	D7	D6	D5	D4	D3	D2	D1	D0
SCON	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

2）主机发送信息:先发送地址信息，设主机发送地址帧信息02H，02H是从机2的地址。则TB8=1表示地址。

SM2=1 ⟨ RB8=1,数据有效装入SBUF，RI=1可中断申请
RB8=0,数据无效丢弃，RI=0不发中断申请
SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请

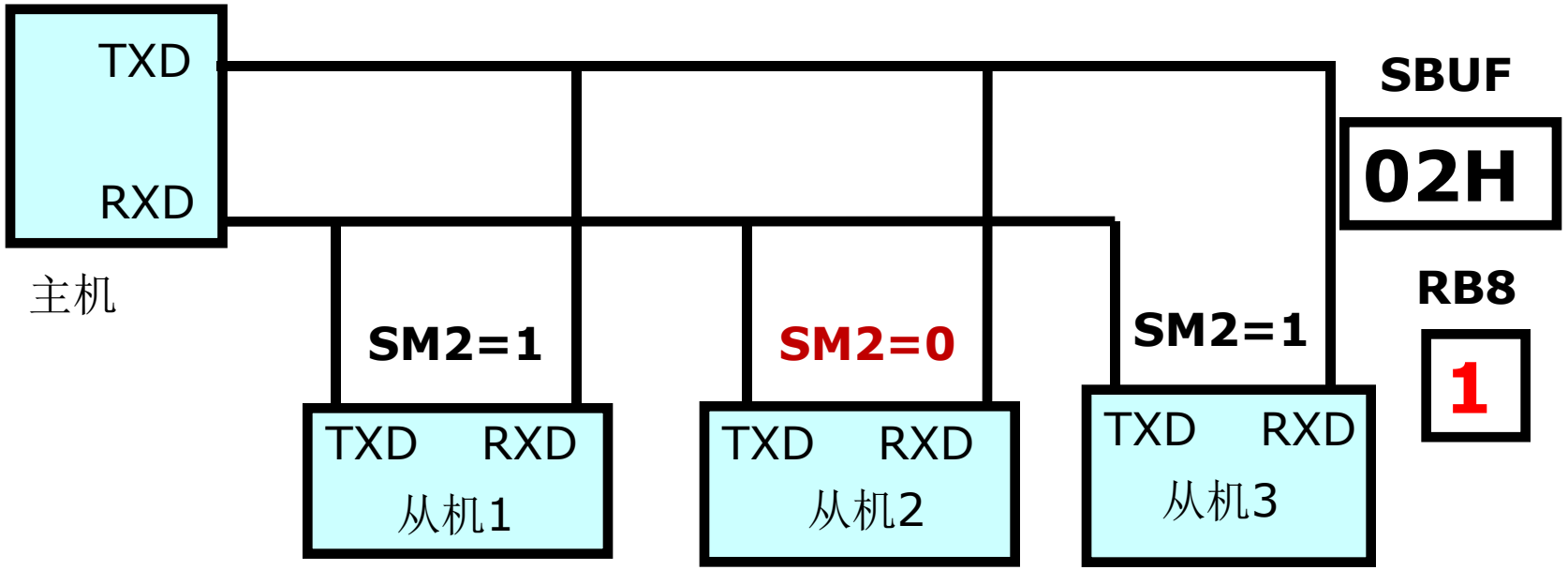


设主机发送地址帧信息02H时，各从机已初始化SM2和REN均为1

SM2=1 ⟨ RB8=1,数据装入SBUF，RI=1各从机响应中断
RB8=0,数据无效丢弃，RI=0不发中断申请

SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请

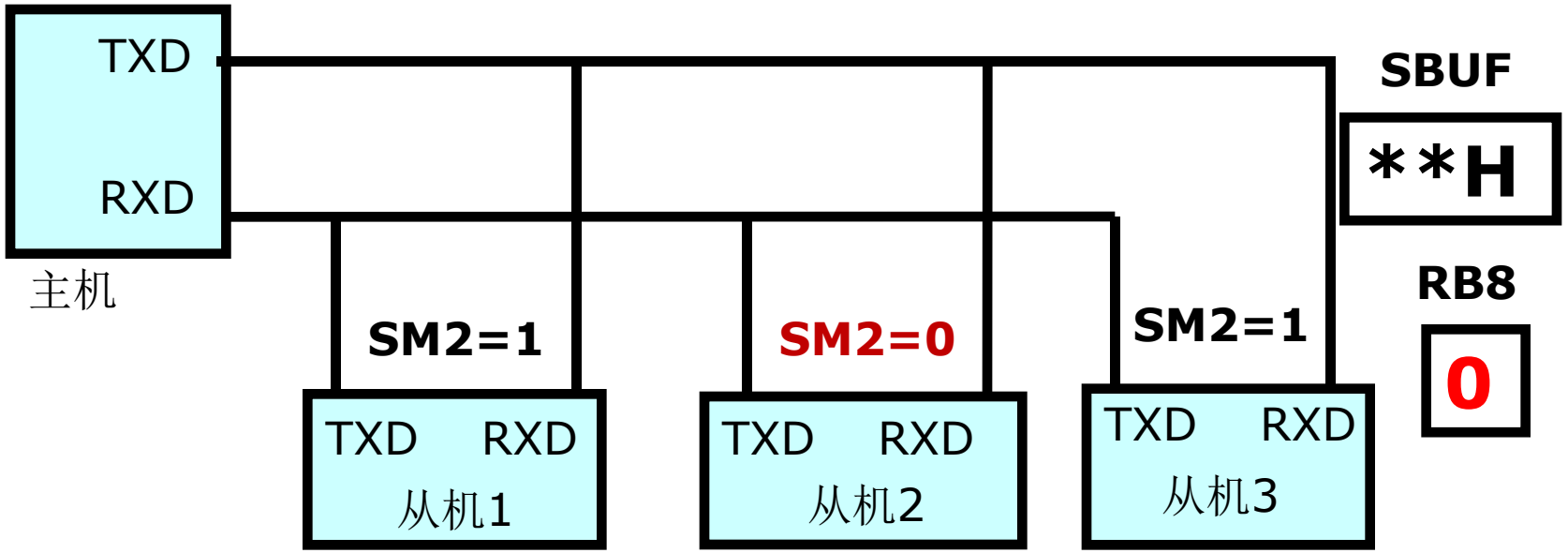
SM2=1 ≺ RB8=1,数据有效装入SBUF，RI=1可中断申请
RB8=0,数据无效丢弃，RI=0不发中断申请
SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请



3) 各从机在中断程序中判断：接收到的地址与本机是否相符?

地址相符,则该从机置SM2=0；地址不符,保持SM2=1不变。

SM2=1 ⟨ RB8=1,数据有效装入SBUF，RI=1可中断申请
RB8=0,数据无效丢弃，RI=0不发中断申请
SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请

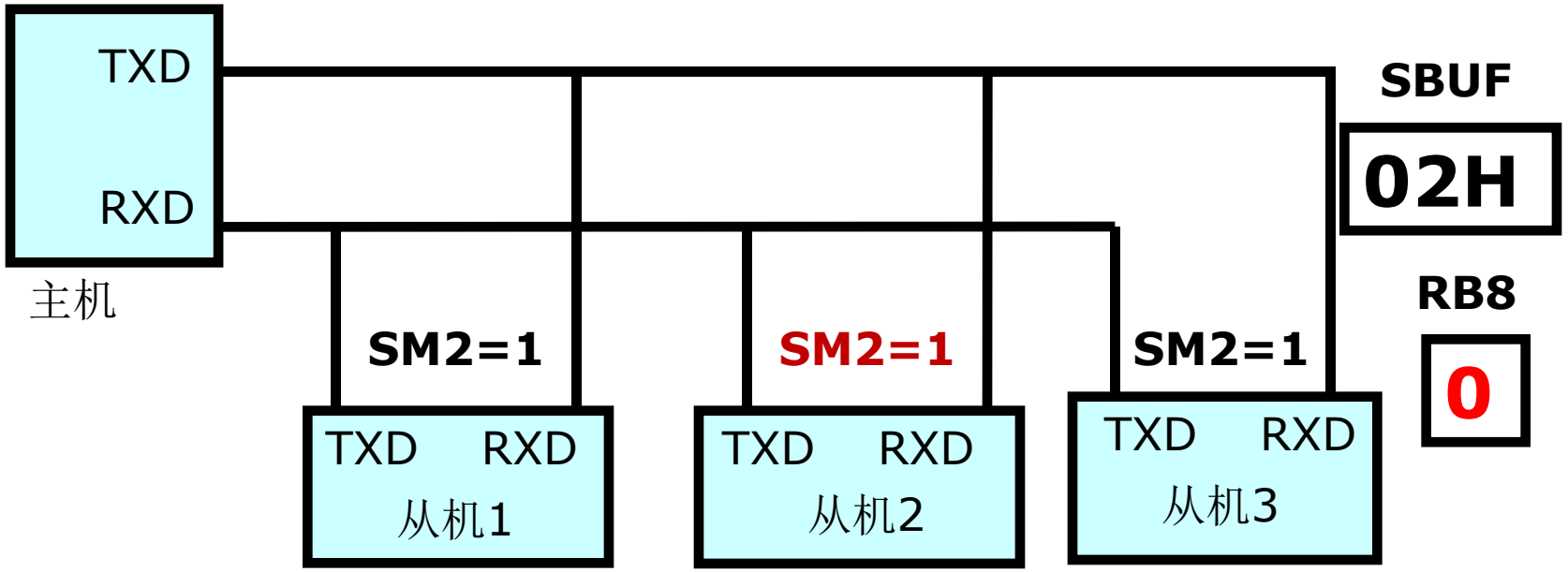


4)主机再发送数据帧，置TB8=0(表示数据)，因此从机接收到会使RB8=0，此时只有SM2=0即地址相符的从机可接收数据，进入中断处理。

SM2=1 ⟨ RB8=1,数据有效装入SBUF，RI=1可中断申请
RB8=0,数据无效丢弃，RI=0不发中断申请

SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请

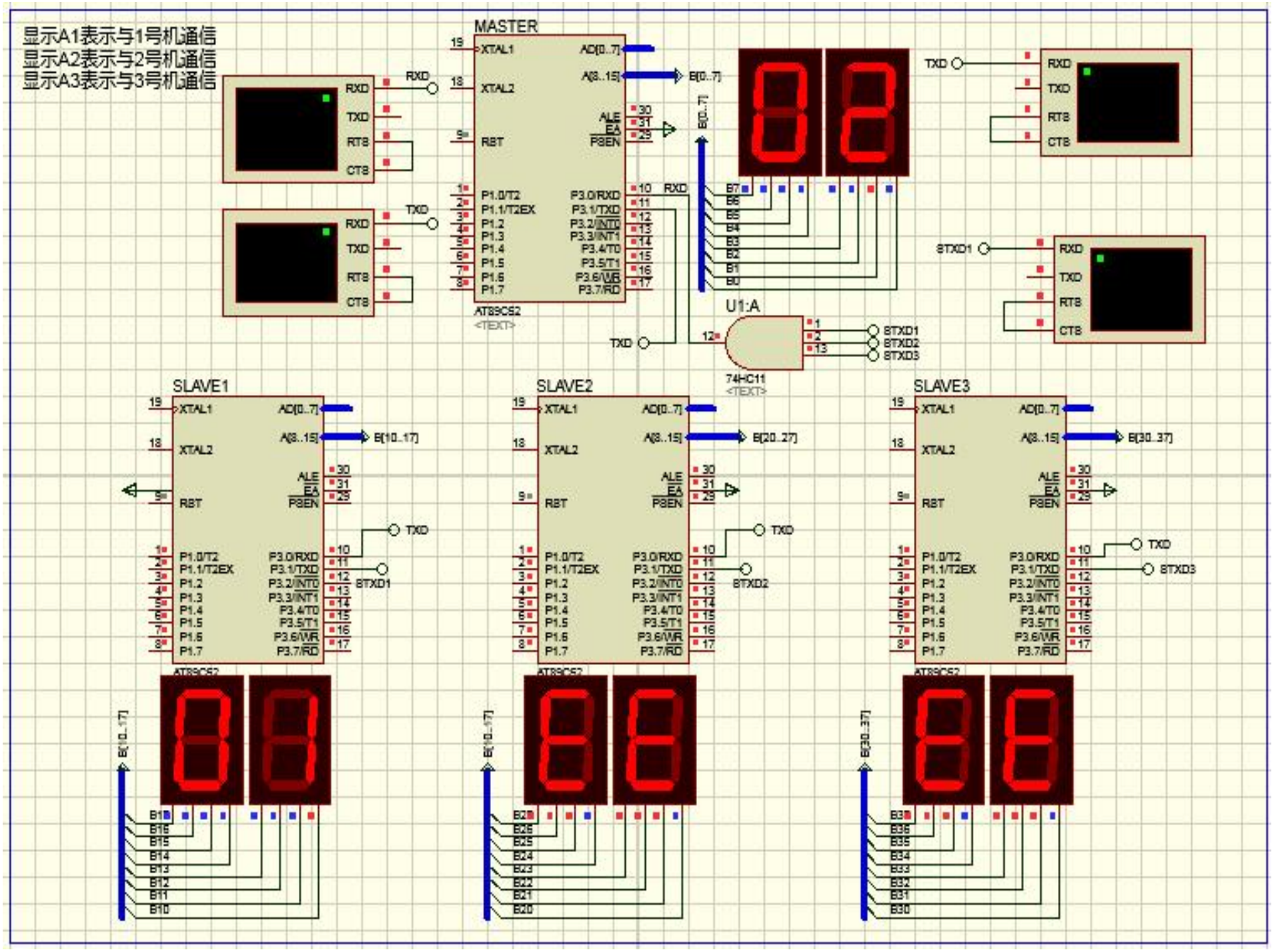
SM2=1 ≺ RB8=1,数据有效装入SBUF，RI=1可中断申请
RB8=0,数据无效丢弃，RI=0不发中断申请
SM2=0,无论RB8=0或1,数据都有效装入SBUF，RI=1中断申请



5) 结束此次通信，为下一次多机通信做准备。

在多机系统，每个从机地址是唯一的，但要预留1～2个广播地址，即所有从机共有的地址。

当主机与从机的数据通信结束后，一定要将从机再设置为多机通信模式，即再将所有从机的SM2置为1，以便下次多机通信。


显示A1表示与1号机通信
显示A2表示与2号机通信
显示A3表示与3号机通信
MASTER
AT89C52
U1:A
74HC11
SLAVE1
SLAVE2
SLAVE3
RXD
TXD
RTS
CTS
STXD1
STXD2
STXD3

串行通信要点

重点

- 串行通信基本概念和串口结构
- 串行口相关的SFR
- 串行口的发送和接收的启动
- 4种工作方式的特点、帧格式和波特率

- 串行口的初始化
- 波特率的设定与计算
- 串行口的应用
  - 单片机之间的通信
  - PC与单片机间的通信

复习

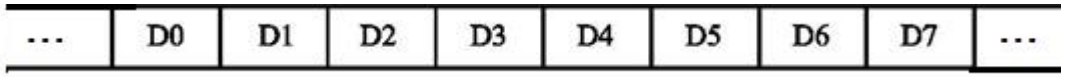
1. 与串口通信相关寄存器：SCON(98H) PCON(87H)

	D7	D6	D5	D4	D3	D2	D1	D0
SCON:	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
位地址	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H

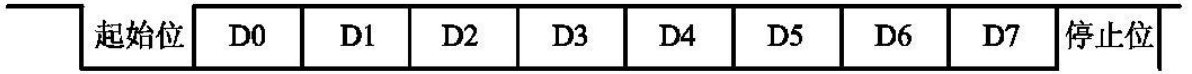
	D7	D6	D5	D4	D3	D2	D1	D0
PCON	SMOD	SMOD0	-	与	串	行	通	信 无 关

复习　2. 工作方式和帧格式：

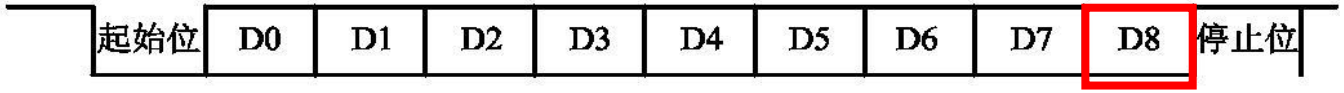
方式0：　8位同步移位寄存器方式，波特率=$F_{CLK}/12$（12T）或=$F_{CLK}/6$（6T）



方式1：　10位异步通信方式，波特率=$(2^{SMOD}/32)*$T1（或T2）的溢出率



方式2：　11位异步通信方式，波特率=$(2^{SMOD}/64)*F_{CLK}$



方式3：　11位异步通信方式，波特率计算与方式1波特率计算方法相同

方式3帧格式与方式2帧格式相同。

复习　3. 波特率设定与计算

串口工作在方式1或方式3需要计算产生波特率的计数器的定时计数初值，

方式1或方式3的波特率=(2^{SMOD}/32)*定时器的溢出率

溢出率=1/溢出周期

产生波特率常用定时器的工作方式2，此时的波特率计算公式：

$$\text{波特率} = \frac{2^{SMOD}}{32} \times \frac{f_{CLK}}{12 \times (256 - X)}$$

计数初值计算公式：

$$X = 256 - \frac{f_{osc} \times (SMOD + 1)}{384 \times \text{波特率}}$$