第七章 单片机串行通信接口

7.1 Hello,world!

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

单片机C语言程序设计中的标准输入与输出

- 在传统的C语言程序设计中，标准输出对应的设备是屏幕，通过函数printf()将字符串输出到显示屏上；标准输入对应的是键盘，通过函数scanf()从键盘读取用户输入的字符；
- 单片机本身硬件资源有限，并没有屏幕和键盘，所以C51设计的标准输出和标准输入所对应的设备都是串行口。即在用C51编写的单片机程序中，printf()函数输出的字符串将送单片机的串行口发送，而scanf()函数则从单片机的串行口读取输入的数据。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

单片机C语言程序设计中的标准输入与输出

- 在使用单片机的串行口进行通信前，首先要对其进行初始化，对通信帧格式及波特率进行设定，保证通信双方通信帧格式及波特率一致；
- 在硬件电路上，单片机系统之间短距离的通信可以将引脚直接用导线相连；
- 如果通信距离较远或者单片机需要和PC机进行通信，则必须进行TTL/CMOS电平与RS-232C电平的变换方能正常通信；
- 关于串行口波特率设置、电平变换等内容在第二章中已有详细讲解，请自行参阅。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

单片机C语言程序设计中的Hello world程序

```
#include <reg51.h>     // 对单片机的端口、寄存器等内部资源进行定义的头文件
#include <stdio.h>     // 程序中如果使用输入/输出函数，必须包含此头文件
// 使用define定义系统中的常数，既清晰明了，又方便修改
#define    OSC          11059200
#define    BAUD         9600
void main(void)
{
    TMOD  = 0x20;                          // T1选择模式2，用于波特率发生器
    SCON  = 0x50;                          // 串口方式1（N81），REN=1，允许接收
    PCON  |= 0x80;                         // SMOD=1，波特率倍增
    TL1   = 256 - (OSC/12/16/BAUD);
    TH1   = 256 - (OSC/12/16/BAUD);        // 计算并设置波特率常数（串行口硬件以设定
                                           // 波特率的16倍速率扫描串行通信线路）
    TR1   = 1;                             // 启动定时器，波特率发生器开始工作
    TI    = 1;                                    // 使用printf函数的要求

    printf("\r\nHello, world!\r\n");       // 输出Hello, world!，前后均换行
    while(1);                              // 执行完毕，在此处循环，以免程序“跑飞”
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

单片机C语言程序设计中的putchar程序

- printf函数可通过串行口输出各种格式化的字符或字符串，而每个字符的输出则是调用putchar函数实现的：

```
char putchar (char c)
{
    if (c == '\n')              // 如果输出的是换行符
    {
        while (!TI);            // 如TI不为1，等待其变为1。即等待上一字符发完
        TI = 0;
        SBUF = 0x0d;            // 自动增加一个回车符
    }
    while (!TI);                // 每次都等待前一个字节发完再发下一字节
    TI = 0;                     // 清除TI标志
    return (SBUF = c);          // 发送待发字节并返回
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

单片机C语言程序设计中的getkey程序

- scanf函数可通过串行口输入各种格式化的字符或字符串，而每个字符的输入则是调用getkey函数实现的：

```
#include <reg51.h>
char _getkey ()
{
    char c;

    while (!RI);              // 等待串行口接收完一个字节的数据
    c = SBUF;                 // 将串行口接收的数据存入临时变量
    RI = 0;                   // 清RI标志
    return (c);               // 返回接收到的数据字节
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

单片机C语言程序设计中输入输出的重定向

- 在实际应用中，程序员可通过更改putchar函数，例如将putchar更改为向液晶显示器模块(LCM)输出，即可实现printf函数在不同外设上的格式化输出；
- 在实际应用中，程序员可通过更改_getkey函数，例如将_getkey更改为从扩展的行列式键盘读取数据，即可实现scanf函数在不同外设上的格式化输入；
- 实际编程时，只要将putchar或_getkey函数加入到工程文件中，编译链接即可取代标准库函数中相应的函数。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

第七章 单片机串行通信接口

7.2单片机串行口
查询方式通信

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.2.1 设计思路分析

- 特殊功能寄存器SCON中的TI位和RI位分别表示当前串行口数据收发的状态；
- TI为发送结束标志位，在一字节数据发送完毕时置1；
- RI为接收完成标志位，在一字节数据接收完成时置1；
- TI和RI中任何一个为1，都可向CPU申请串行口中断。即使单片机设置为不响应任何中断，TI和RI仍会根据串行帧的收发情况置位或复位；
- 通过TI和RI可实现查询方式的串行通信。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.2.1 设计思路分析

- 单片机发送数据时，可采用两种查询方式：
  - 方式一：发送数据前先查TI是否为1，TI=1表示上一个数据已经发完，则先置TI=0，再发送数据，否则等待TI=1。此方式下发送数据指令执行完毕后即可返回。注意此方式要求在初始化串口时先置TI=1；
  - 方式二：串口初始化时置TI=0，此后向串行口发送数据时，将待发送数据写入SBUF后，必须等待TI为1，并将TI清零后再返回。
- 单片机接收数据时，只要查询到RI为1，就表示有一个有效的数据字节已被接收到SBUF中。
- 单片机应及时从SBUF中读取本次接收到的数据，以免该数据被下一个接收的数据覆盖。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.2.1 设计思路分析

- 当单片机需要同时进行数据的接收和发送时，要采用轮询的方式进行，在一个大循环中分别查询各标志，使用if或switch语句而不是while语句来判断标志是否满足条件，以避免某一标志始终不出现而导致系统死锁。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.2.2 串行口查询方式通信程序实例

- 使用C51设计单片机串行口控制程序，设单片机的晶振频率为11.0592MHz，串行口工作在9600bps、8位数据位、1位停止位、无校验模式。单片机采用查询方式控制数据的收发流程，将串行口接收到的除回车/换行之外的数据加1后送回。

➢ 根据程序要求，单片机应被动等待对方先发数据，再根据收到的数据判断如何回送，因此单片机初始化完成后首先应等待接收数据，结合晶振频率及要求的单片机串行口工作方式，设计程序如下：

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.2.2 串行口查询方式通信程序实例

- 初始化部分代码

```
#include <reg51.h>

#define OSC          11059200
#define BAUDRATE     9600

void main(void)
{
    unsigned char c;

    TMOD     = 0x20;                          // T1工作在方式2,作为波特率发生器
    SCON     = 0x50;                          // 串口方式1(N81模式),REN=1(允许接收)
    PCON     |= 0x80;                         // SMOD=1
    TL1      = 256-(OSC/12/16/BAUDRATE);
    TH1      = 256-(OSC/12/16/BAUDRATE);      // 设置定时器初始值
    TR1      = 1;                             // 启动定时器,输出波特率时钟

    RI       = 0;
    TI       = 0;                             // 确保程序开始时RI/TI标志无效
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.2.2 串行口查询方式通信程序实例

- 功能实现部分代码

```
    while(1)                                    // 主循环
    {
        while(RI==0);                           // 等待对方发送数据
        RI = 0;                                 // 能执行到这里，即说明RI=1，接收数据有效
        c = SBUF;                               // 将接收数据从SBUF读出至内部变量中
        switch(c)                               // 使用switch()进行分支判断
        {
            case 0x0d:
            case 0x0a:                          // 回车/换行原样送回
                SBUF = c;
                break;
            default:
                SBUF = ++c;                     // 其余数据加1后送回
                break;
        }

        while(!TI);                             // 等待串行帧发送完毕
        TI = 0;                                 // 清零TI后开始下次循环
    }
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

第七章 单片机串行通信接口

7.3单片机串行口
中断方式通信

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

- 当CPU开放串行口中断时，RI或TI中任一个为1（即收到或发完一个字节）时都会向CPU申请中断，在中断服务程序中进行数据处理。在没有中断发生的时候，CPU可执行其它任务；
- 在实际程序设计中，根据具体情况，通过中断服务程序可选择处理某一个或全部中断事件；
- 中断服务程序应遵循“先保存后处理”的原则，尽量减少中断服务程序（ISR）的执行时间；
- 使用中断方式管理串行数据的收发，可极大提高代码的执行效率。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

1、只处理串行口接收数据中断

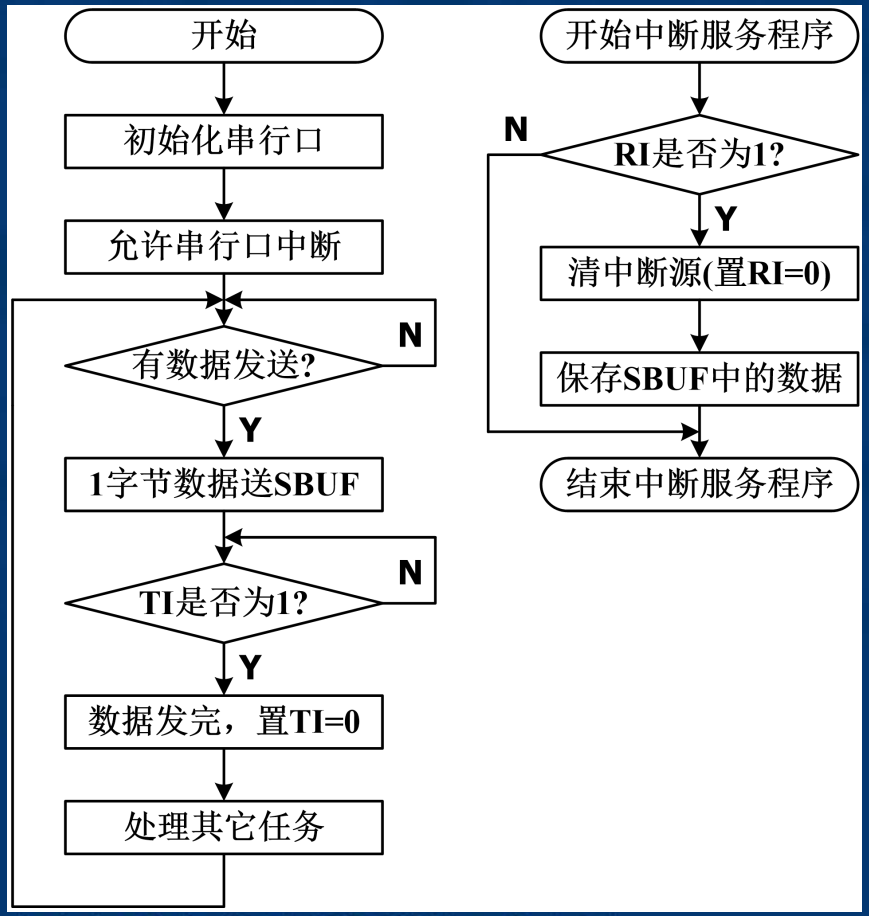
- 由于串行通信时发送数据是在程序的控制下主动进行的，程序只要在发送数据前查询TI位，确保前一个串行帧发送完毕前不发送下一帧即可，因此串行数据的发送可以不使用中断；
- 而串行数据的接收则是随机的，何时接收数据完全取决于对方，故应该采用中断方式进行接收；
- 在这种情况下，每个数据字节发送完成时，硬件仍会自动置TI=1，CPU也响应该中断。但是只要在中断服务程序中不清除TI，TI将保持不变，供主程序查询。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

1、只处理串行口接收数据中断

- 程序流程图



南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

2、处理串行口接收数据和发送数据中断

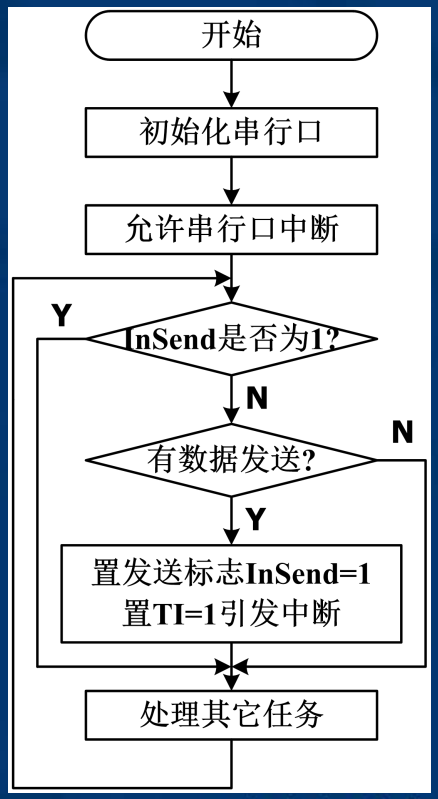
- 为了进一步提高串行口数据收发的效率，串行口发送数据也可以采用中断驱动的方式；
- 程序首先将待发送的数据送入数据发送缓冲区；
- 在第一个数据发送前强制置TI=1引发串行口发送中断；
- CPU响应中断，进入中断服务程序进行数据发送处理，数据发完后将引发中断，继续后续数据的发送；
- 中断方式数据接收处理过程和前述的方法相同。

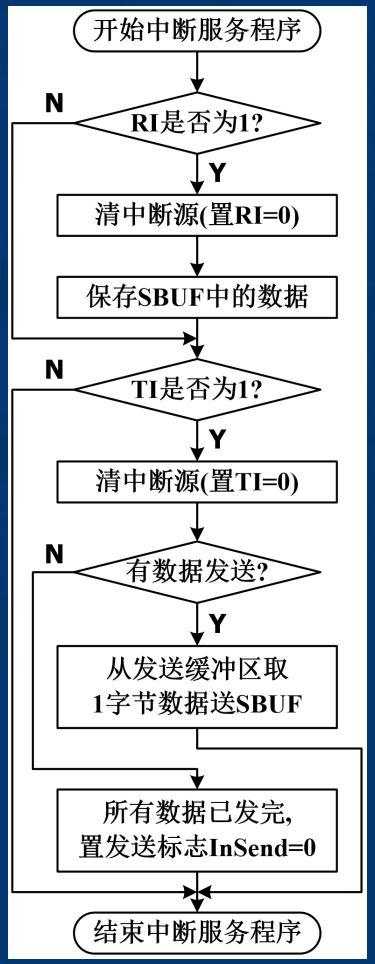
南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

2、处理串行口接收数据和发送数据中断

- 程序流程图





南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

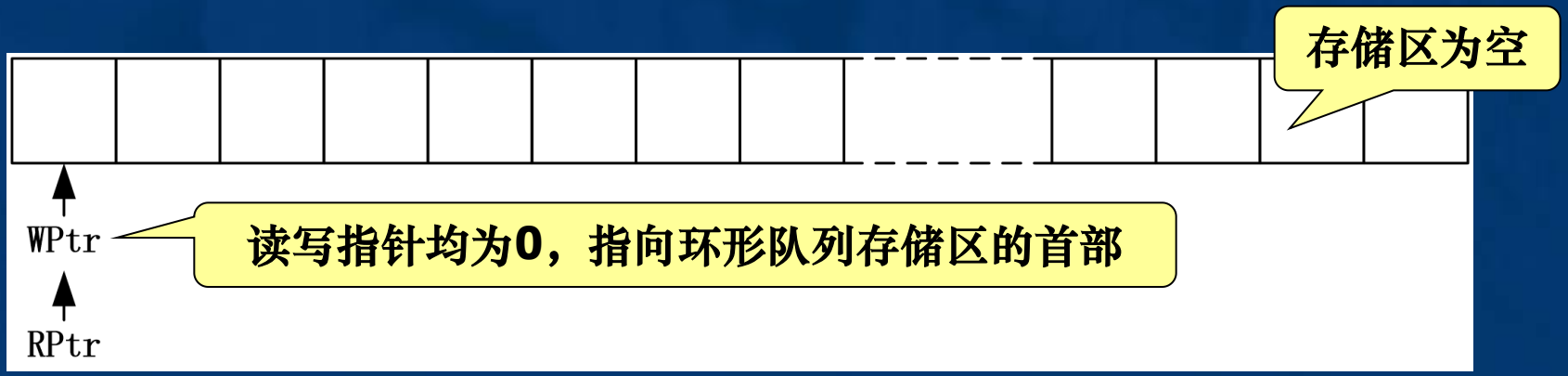
- 数据缓冲的目的是为了加快CPU对数据收发的响应速度和系统事件处理的效率；
- 数据缓冲通常采用“先存储，后处理”的方式，即在ISR中只进行数据的收发，数据的处理则由其他模块完成；
- 在存储空间比较紧张的单片机系统中，一般通过内存需求较小的环形队列来实现数据的缓冲。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 环形队列由存储区和读写指针构成；
- 一个空的环形队列的结构如下图所示：

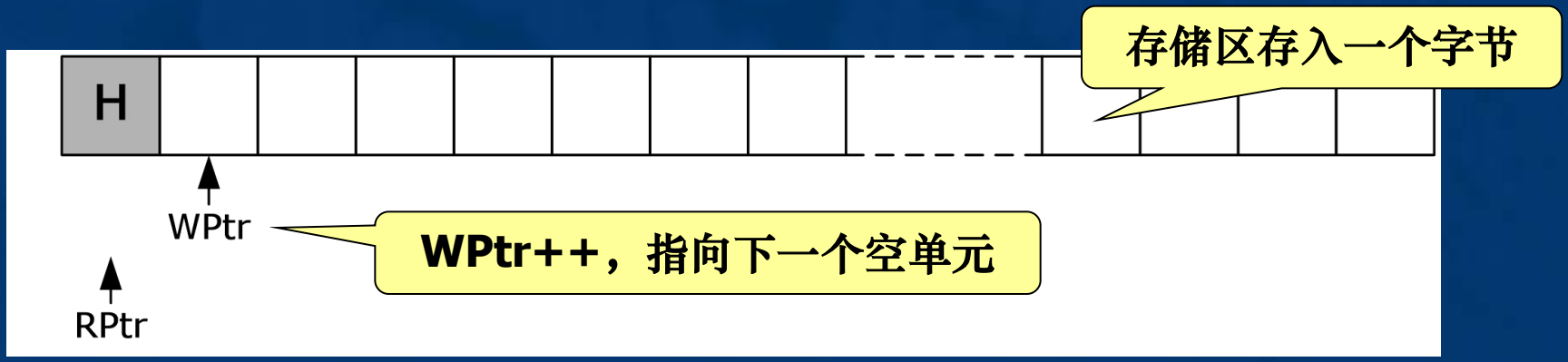


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 写入一个字节后的环形缓冲区：

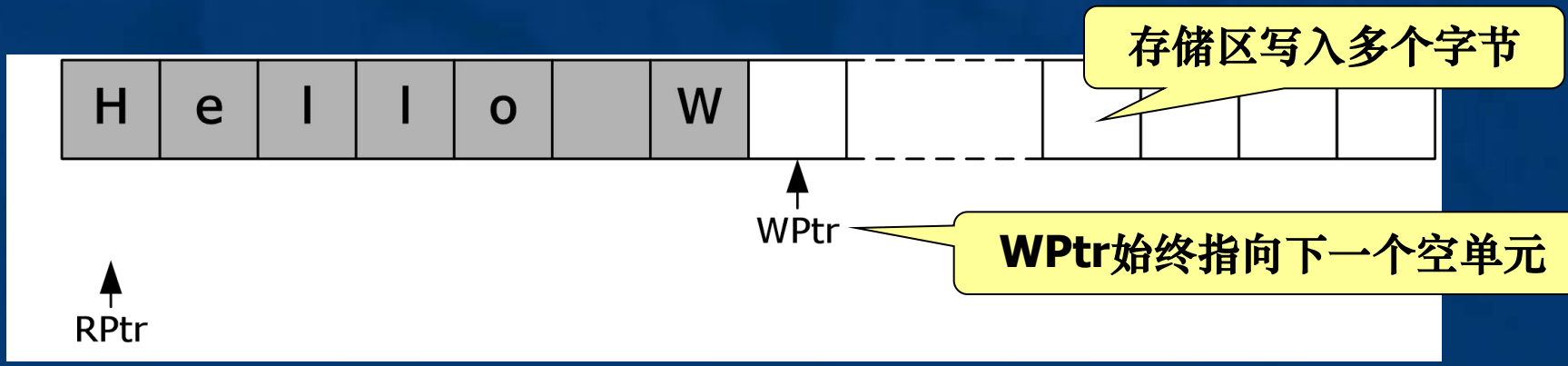


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 写入多个字节后的环形缓冲区：

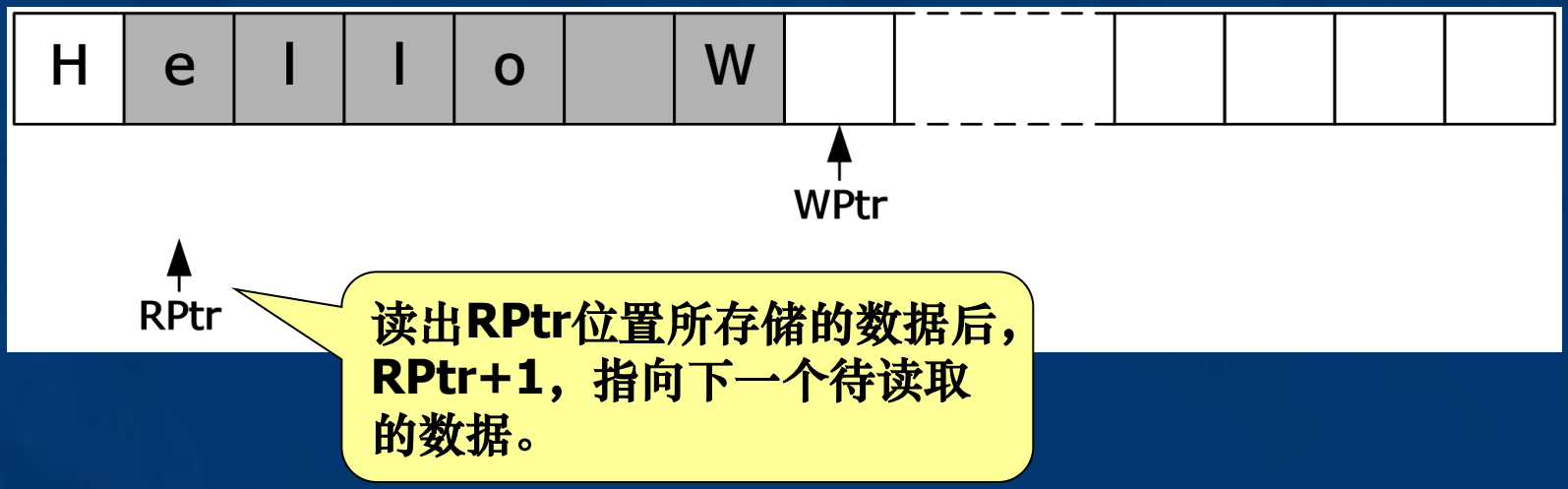


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 读出一个字节后的环形缓冲区：

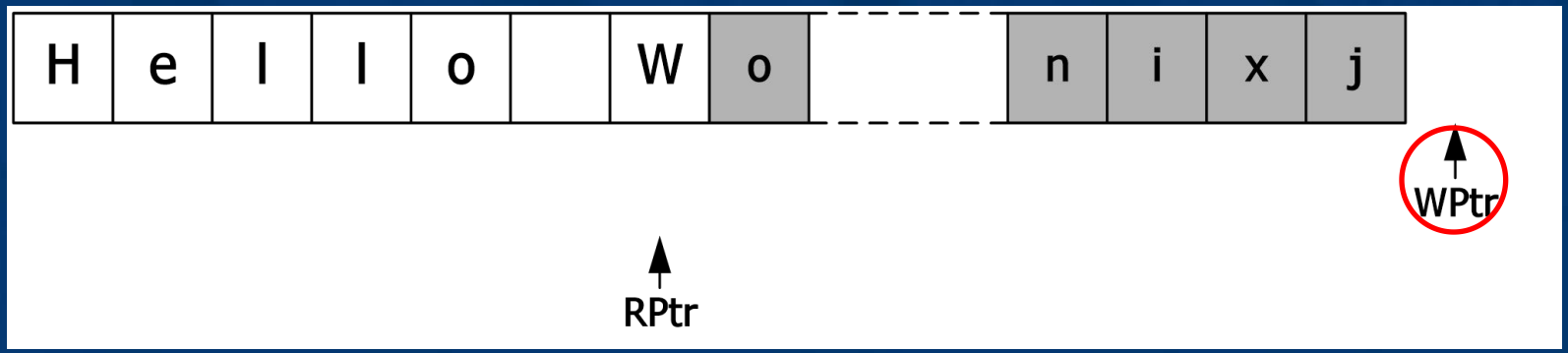


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 当写入某字节时写指针超过缓冲区末尾：

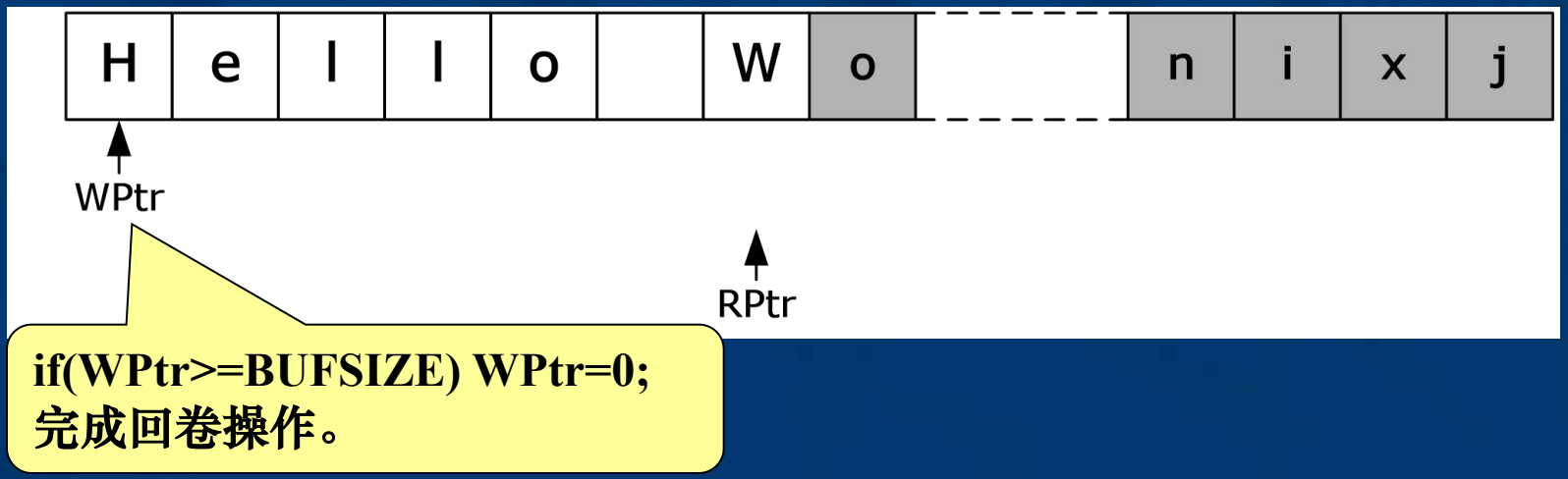


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

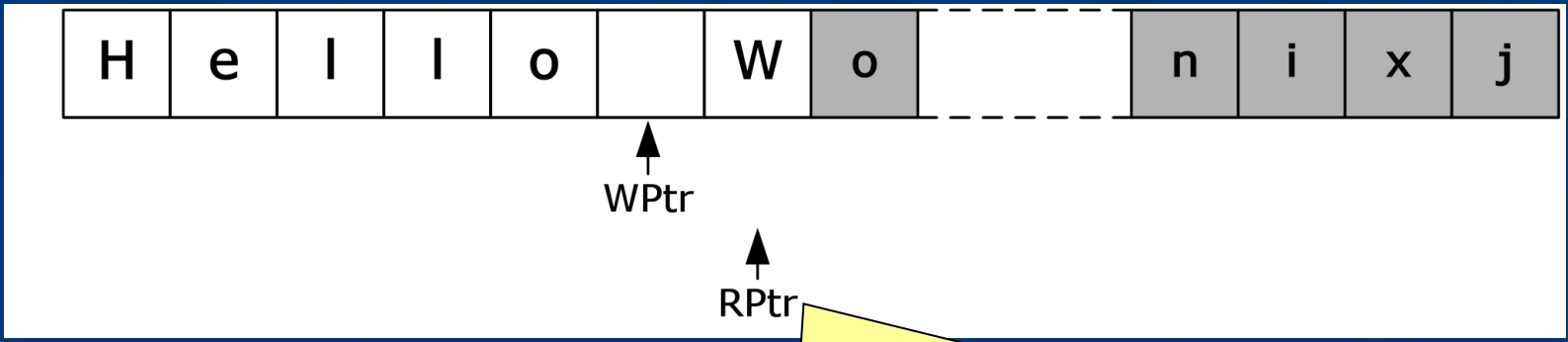
- 写指针回卷(Rewind)，读指针与此类似：



7.3.1 设计思路分析

3、数据的缓冲

- 缓冲区满：



假设缓冲区大小为n字节，由于WPtr和RPtr相等表示缓冲区空，因此特规定WPtr和RPtr相差1字节时，即存入n-1字节时缓冲区满

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 缓冲区存储字节数的计算
  - 缓冲区中的数据是“先存储，后读取”，缓冲区中存储字节数的计算方法为：

```
bytes = WPtr - RPtr;                    // 写指针-读指针
if(bytes < 0) bytes += BUFSIZE;         // 写指针回卷将计算出负值
                                        // 此时应加上缓冲区长度
```

- 根据前面的分析，如果计算结果表明环形队列中已经存储了BUFSIZE-1字节的数据时，表示队列区已满，此后不应再向队列中存储数据（直到队列中的一个或多个数据被读出为止）。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 如果缓冲区大于256字节，WPtr和RPtr必须采用整型数，8位单片机处理时将分多条指令完成；
- 此时有可能出现这样的情况：主程序根据读写指针值计算缓冲区存储字节数时被串行口中断打断，而串行口中断服务程序读写缓冲区数据时，又更改了缓冲区的读写指针；
- 当中断服务程序执行完毕返回主程序后，主程序继续计算出的缓冲区存储字节数就有可能出错；
- 为了避免这种情况的发生，当计算环形队列存储字节数时，要暂时禁止串行口中断，计算完毕再开放串口中断。

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.1 设计思路分析

3、数据的缓冲

- 暂时禁止串行口中断并不会影响串行口的数据收发；
- 这是因为当初始化串行口时，串行数据的每一位都要进行至少16次扫描，假设波特率设置到当前晶振下的最高值，即定时器T1初值设置为255，每个机器周期溢出一次，那么收发一个串行帧(10bit)，则至少需要160个机器周期；
- 而整型数的加减法仅需不超过二十个机器周期即可计算完毕，因此不会因暂时禁止串口中断而影响串行数据收发。

7.3.2 串行口中断方式通信程序实例

- 使用C51设计单片机串行口控制程序，设单片机的晶振频率为11.0592MHz，串行口工作在9600bps、8位数据位、1位停止位、无校验模式。单片机采用全中断方式控制数据的收发流程，使用环形缓冲区存储串行口接收到的数据，并将收到的除回车/换行之外的数据加1后送回。

➢ 根据题目要求，单片机处理串行口的接收中断和发送中断，存储接收数据的环形缓冲区的大小由BUFSIZE定义，编写程序如下：

南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.2 串行口中断方式通信程序实例

- 全局变量定义及初始化部分代码

```
#include <reg51.h>

#define OSC          11059200
#define BAUDRATE     9600
#define BUFSIZE      8

unsigned char Buf[BUFSIZE];
int RPtr,WPtr;
bit InSend;
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.2 串行口中断方式通信程序实例

- 通用功能函数代码

```
// 指针加1函数，同时判断是否需要回卷
void IncPtr(int *ptr)                    // 通过指针方式操作，可直接修改原变量
{
    ES = 0;                              // 禁止串行口中断
    (*ptr) ++;                           // 指针内容+1
    if((*ptr)>=BUFSIZE) (*ptr)=0;        // 判断是否需要回卷
    ES = 1;                              // 恢复允许串行口中断
}

// 计算环形缓冲区存储字节数的函数
int BytesInBuf(void)
{
    int bytes;
    ES = 0;
    bytes = WPtr - RPtr;
    if(bytes < 0) bytes += BUFSIZE;
    ES = 1;
    return bytes;
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.2 串行口中断方式通信程序实例

- 串口中断服务程序

```
void ComISR(void) interrupt 4
{
    unsigned char c;

    if(RI)                                  // 接收数据中断
    {
        RI = 0;                             // 首先清RI
        c = SBUF;
        if((c!=0x0d)&&(c!=0x0a)) c++;       // 对非回车/换行数据加1
        if(BytesInBuf() < BUFSIZE-1)        // 如果缓冲区未满
        {
            Buf[WPtr] = c;                  // 数据写入缓冲区
            IncPtr(&WPtr);                  // WPtr+1并判断是否需要回卷
        }
    }

    if(TI)                                  // 发送完数据中断
    {
        TI = 0;                             // 首先清TI
        if(BytesInBuf()>0)                  // 如果缓冲区中有数据
        {
            SBUF = Buf[RPtr];               // 从缓冲区取出一字节数据发送
            IncPtr(&RPtr);                  // RPtr+1并判断是否需要回卷
        }
        else InSend = 0;                    // 如果缓冲区已无数据待发，说明此TI为最后
                                            // 一字节数据发完引起的，清除InSend标志
    }
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269

7.3.2 串行口中断方式通信程序实例

- 主程序代码

```
void main(void)
{
    SCON    = 0x50;                          // 初始化串行口为模式1，允许接收
    PCON    |= 0x80;                         // 波特率加倍
    TMOD    = 0x20;                          // T1工作在方式2，作为波特率发生器
    TH1     = 256-(OSC/12/16/BAUDRATE);
    TL1     = 256-(OSC/12/16/BAUDRATE);      // 计算波特率常数
    TR1     = 1;                             // 启动波特率发生器
    TI      = 0;
    RI      = 0;                             // 初始化TI、RI标志为0

    RPtr    = 0;
    WPtr    = 0;                             // 读写指针均初始化为0，即清空缓冲区
    InSend  = 0;                             // 清除发送标志

    ES      = 1;                             // 允许串行口中断
    EA      = 1;                             // 总中断允许

    while(1)                                 // 主循环
    {
        if(!InSend)                          //如果没有处于数据发送状态
        {
            if(BytesInBuf()>0)               // 如果缓冲区中有数据
            {
                InSend = 1;                  // 置位开始发送标志
                TI = 1;                      // 强行设置TI为1，引发串行口发送中断
            }
        }
        // 此处可加入其它任务的处理程序…
    }
}
```


南京邮电大学计算机学院 倪晓军
nixj@njupt.edu.cn 13505151269