目录

[一、 串口通信 1](#_Toc33634691)

[1. 通信简介 1](#_Toc33634692)

[1) 并行通信 1](#_Toc33634693)

[2) 串行通信 1](#_Toc33634694)

[2. 串行通信 1](#_Toc33634695)

[1) 串行通讯数据传送方向 1](#_Toc33634696)

[2) 串行通信基本方式 1](#_Toc33634697)

[3) 异步串行通信数据格式 2](#_Toc33634698)

[4) 串行接口标准 2](#_Toc33634699)

[3. 通信实验 3](#_Toc33634700)

[1) 实验环境的搭建 3](#_Toc33634701)

[2) 实验内容说明 4](#_Toc33634702)

[3) 单片机程序的编制 5](#_Toc33634703)

[4) 窗口应用程序的编制 5](#_Toc33634704)

[5) 实验过程 7](#_Toc33634705)

[二、 TCP通信 10](#_Toc33634706)

[1. Socket套接字 10](#_Toc33634707)

[1) 简介 10](#_Toc33634708)

[2) 分类 10](#_Toc33634709)

[2. TCP简介 10](#_Toc33634710)

[3. TCP通信过程 10](#_Toc33634711)

[4. TCP服务器程序流程 11](#_Toc33634712)

[1) 建立连接阶段 11](#_Toc33634713)

[2) 数据交互阶段 11](#_Toc33634714)

[3) 关闭连接 11](#_Toc33634715)

[5. TCP客户端程序流程 11](#_Toc33634716)

[1) 建立连接阶段 12](#_Toc33634717)

[2) 数据交互阶段 12](#_Toc33634718)

[3) 关闭连接 12](#_Toc33634719)

[6. 通信实验 12](#_Toc33634720)

[1) 实验环境的搭建 12](#_Toc33634721)

[2) 实验内容说明 12](#_Toc33634722)

[3) 服务器端程序的编制 12](#_Toc33634723)

[4) 客户端程序的编制 13](#_Toc33634724)

[5) 实验过程 15](#_Toc33634725)

[三、 UDP通信 16](#_Toc33634726)

[1. UDP简介 16](#_Toc33634727)

[2. UDP通信过程 16](#_Toc33634728)

[3. UDP服务器程序流程 16](#_Toc33634729)

[4. UDP客户端程序流程 16](#_Toc33634730)

[5. 通信实验 16](#_Toc33634731)

[1) 实验内容说明 16](#_Toc33634732)

[2) 服务器端程序的编制 17](#_Toc33634733)

[3) 客户端程序的编制 17](#_Toc33634734)

[4) 实验过程 18](#_Toc33634735)

[四、 附件 19](#_Toc33634736)

[1. 附件一（单片机程序） 19](#_Toc33634737)

[2. 附件二（VB.Net程序） 23](#_Toc33634738)

[3. 附件三（TCP Server程序） 28](#_Toc33634739)

[4. 附件四（TCP Client程序） 35](#_Toc33634740)

[5. 附件五（UDP Server程序） 40](#_Toc33634741)

[6. 附件六（UDP Client程序） 43](#_Toc33634742)

[7. 附件七（单片机原理图） 46](#_Toc33634743)

# 串口通信

## 通信简介

通信有并行通信和串行通信两种方式。在多微机系统以及现代测控系统中信息的交换多采用串行通信方式。

### 并行通信

并行通信是指所传送的数据各位同时进行传送。其优点是传送速度快，缺点是传输线多，通信线路费用较高，并行传送适用于近距离、传送速度高的场合。

### 串行通信

串行通信时，传送数据的各位按分时顺序一位一位地传送(例如先低位、后高位)。其优点是传输线少，传送通道费用低，故适合长距离数据传送。缺点是传送速度较低。

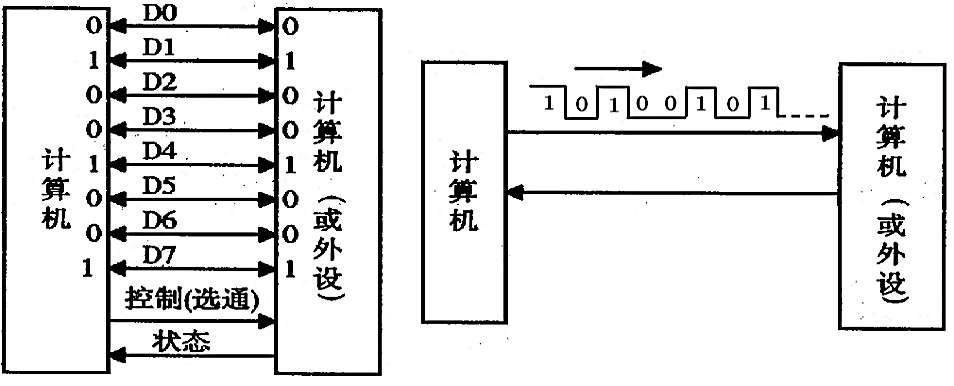


图 1

## 串行通信

### 串行通讯数据传送方向

#### 单工方式

通信双方只有一条单向传输线，只允许数据由一方发送，另一方接收。

#### 半双工方式

通信双方只有一条双向传输线，允许数据双向传送，但每时刻上只能有一方发送，另一方接收，这是一种能够切换传送方向的单工方式。

#### 全双工方式

通信双方只有两条传输线，允许数据同时双向传送，其通信设备应具有完全独立的收发功能。

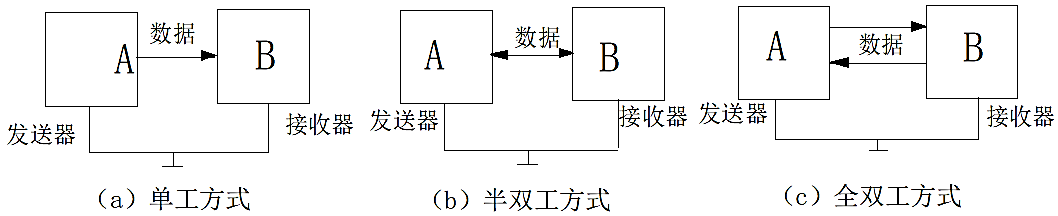


图 2

### 串行通信基本方式

为了准确地发送、接收信息，发送者和接受者双方必须协调工作。这种协调方法，从原理上可分成两种：同步串行I/O和异步串行I/O。

#### 同步通信方式

在同步通信中，在数据或字符开始传送前用同步字符(SYNC)来指示(常约定l一2个)，由时钟来实现发送端和接收端同步，当检测到规定的同步字符后，接下来就连续按顺序传送数据。同步字符是一特定的二进制序列，在传送的数据中不会出现。

同步通信方式由于不采用起始和停止位，是在同步字符后可以接较大的数据区，同步字符所占部分很小，因此有较高的传送效率。

发送、接收双方都由统一的时钟来发送、接收数据。也就是说，不光有数据传输线，还要附加同步时钟线。这种方式，硬件较复杂、成本较高、传送速率高。

#### 异步通信方式

异步通信方式时，数据一帧一帧地传送，不需要同步时钟，实现简单。

在异步方式中，为了避免连续传送过程中的误差积累，每个字符都要独立确定起始和结束(即每个字符都要重新同步)，字符和字符间还可能有长度不定的空闲时间。

### 异步串行通信数据格式

一个字符在异步传送中称为一帧数据。一帧数据由4部分组成：起始位、数据位、奇偶位、停止位。

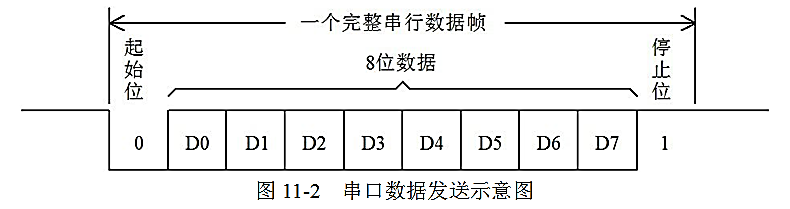


图 3

上图所示为不包含奇偶检验位的一个数据帧。奇偶检验位应位于数据位后，停止位前。

#### 起始位

在没有数据传输时，通信线上处于逻辑“1”状态。当发送端要发送1个字符数据时，首先发送1个逻辑“0”，这个低电平便是帧格式的起始位。其作用是向接收端表示发送端开始发送一帧数据了。接收端检测到这个低电平后，就准备接收数据。

#### 数据位

在起始位之后，发送端发出（或接收端接收）的是数据位，数据的位数没有严格的限制，5~8 位均可。由低位到高位逐位发送。

#### 奇偶校验位

数据位发送完（接收完）之后，可发送一位用来验证数据在传送过程中是否出错的奇偶校验位。奇偶校验是收发双发预先约定的有限差错校验方法之一。有时也可不用奇偶校验。

#### 停止位

字符帧格式的最后部分是停止位，逻辑“高（1）”电平有效，它可占 1/2 位、1 位或 2 位。停止位表示传送一帧信息的结束，也为发送下一帧数据做好了准备。

### 串行接口标准

#### RS-232

RS-232是美国电子工业协会EIA(Electronic Industry Association)制定的一种串行物理接口标准。RS是英文“推荐标准”的缩写，232为标识号。RS-232是对电气特性以及物理特性的规定，只作用于数据的传输通路上，它并不内含对数据的处理方式。RS-232标准是逻辑1为-3V～-15V，逻辑0为+3～+15V。

#### RS-422

RS-422的电气性能与RS-485完全一样。主要的区别在于：RS-422 有4 根信号线：两根发送、两根接收。由于RS-422 的收与发是分开的所以可以同时收和发(全双工)，也正因为全双工要求收发要有单独的信道，所以RS-422适用于两个站之间通信，星型网、环网，不可用于总线网;RS-485 只有2 根信号线，所以只能工作在半双工模式，常用于总线网。

#### RS-485

RS-232接口可以实现点对点的通信方式，但这种方式不能实现联网功能。于是，为了解决这个问题，一个新的标准RS-485产生了。RS-485的数据信号采用差分传输方式，也称作平衡传输，它使用一对双绞线，将其中一线定义为A，另一线定义为B。通常情况下，发送驱动器A、B之间的正电平在+2～+6V，是一个逻辑状态，负电平在-2～-6V，是另一个逻辑状态。另有一个信号地C，在RS-485中还有一“使能”端，而在RS-422中这是可用可不用的。

## 通信实验

### 实验环境的搭建

为实现在无附加硬件的条件下，使用PC机完成串行口通信实验。通过软件对实验环境配置如下：

#### 虚拟串口的创建

使用Configure Virtual Serial Port Driver软件在PC上创建一对虚拟串口（COM1和COM2），如图所示。

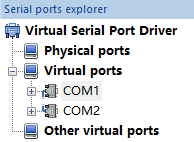


图 4

COM1和COM2为已连接的一对串口，分别为两台设备的接口（已虚拟连接）。设备A，B分别连接至COM1和COM2，即可收发数据。

#### 设备B的创建

此次实验中，设备B为单片微机（MCU）。其选型为AT89C52型，具备硬件UART功能。设备B电路实现由Protues软件进行模拟仿真。电路原理图如图所示。

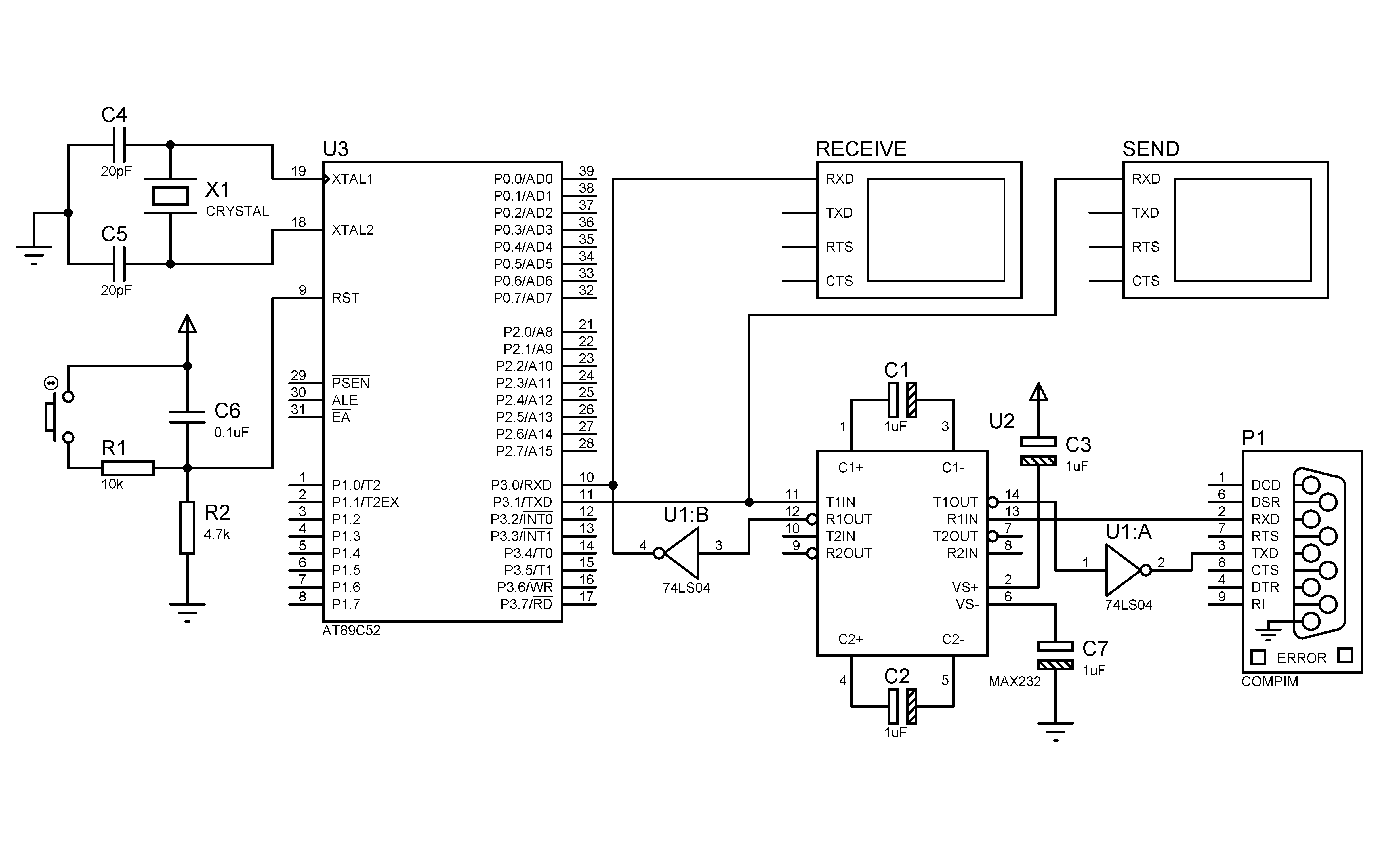


图 5

由于RS-232标准采用负逻辑，即逻辑1为-3V～-15V，逻辑0为+3～+15V。而本单片机系统为TTL信号系统。TTL电平规定，+5V等价于逻辑“1”，0V等价于逻辑“0”。因此，DB-9与单片机的连接需要进行电平转换，如上图所示，选用了MAX232芯片。

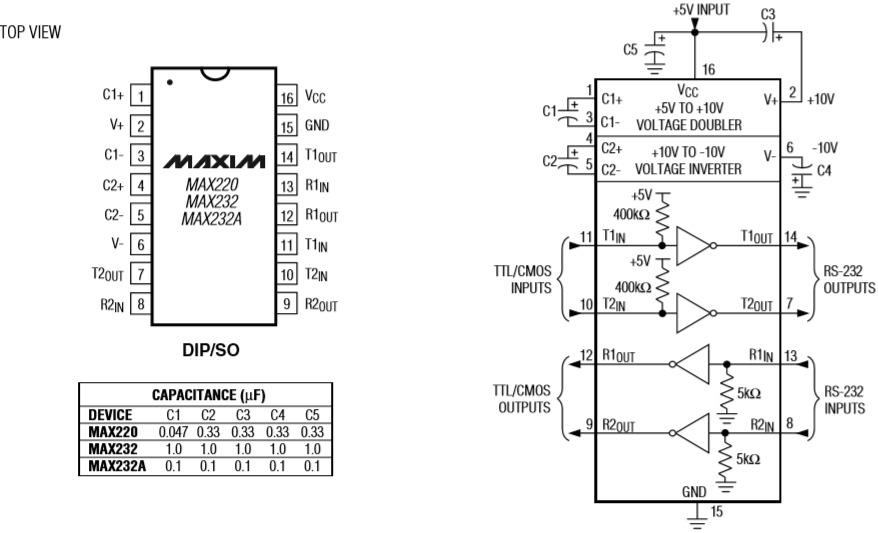


图 6

如图所示，P1组件（COMPIM）即为DB-9接口，使用MAX232芯片进行电平转换后与AT89C52串行口P3.0/RXD，P3.1/TXD分别相连接。（由于软件特殊原因，在MAX232芯片1通道R1OUT及T1OUT处增加74LS04非门进行处理，实际中无需取反处理。）

图中组件RECEIVE和SEND为Protues软件提供的VIRTUAL TERMINAL（虚拟终端）可用于观察AT89C52接收和发送到的数据。仅用于调试观察。

### 实验内容说明

本次实验涉及两个设备，设备B为上述单片微机，用以模拟一个外部设备。设备A为PC机本身。

设备A，B分别都运行有简单的程序。设备A的程序存储器中加载有简单回传程序，负责将接收到的字符/字符串（以回车换行结尾）保持原接收状态并向外发送。设备B即PC，其上运行有窗口程序，负责向设备A发送字符/字符串，并接收显示其回传的数据。

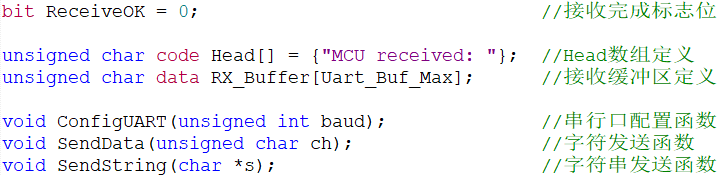
实验时设备A连接COM1口，设备B连接COM2口。由PC窗口程序输入测试数据后发送至单片微机后，由后者保持原样回传并显示在PC窗口程序接收窗口。

### 单片机程序的编制

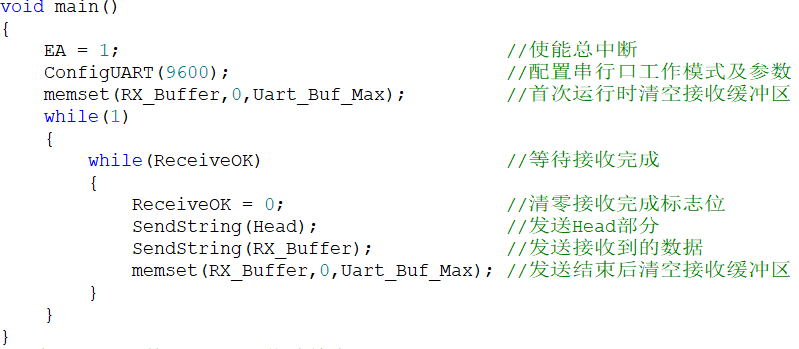
由于该芯片自带硬件UART功能且由于实验环境所限，此次实验中，发送接收通过配置相关寄存器控制硬件模块实现，未采用软件定时器模拟通信时序的方式。

程序主要包括以下三个部分：主函数，串行口配置函数，发送函数及串行口中断服务函数。

#### 程序变量定义及函数声明



#### 程序主函数



#### 其他子函数

其他在主函数中调用的具体子函数实现由附件1列出。

### 窗口应用程序的编制

本次实验的窗口程序使用VB.NET语言在VS2010环境下编写。

程序中主要由.NET提供的SerialPort控件实现，辅之以必要的其他窗口控件。窗口布局如图所示：

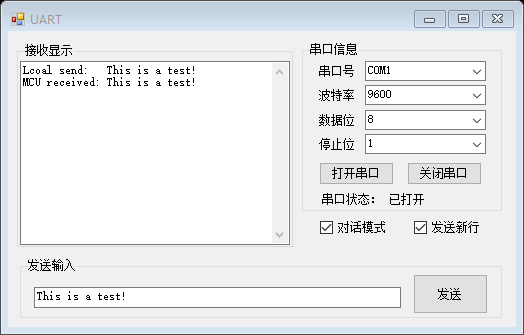
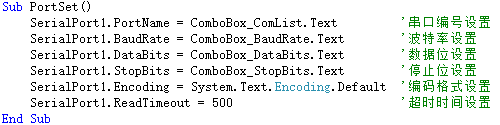


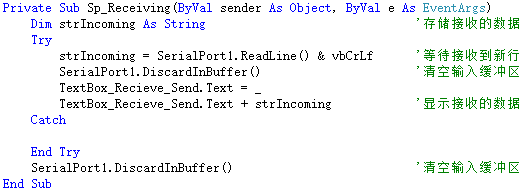
图 7

#### 参数设置函数



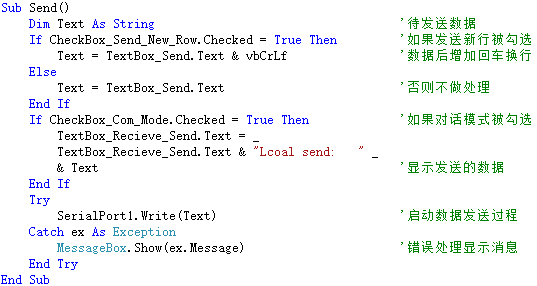
#### 数据接收函数

SerialPort的事件DataReceived调用Sp\_Receiving实现数据接收。当输入缓冲区接收到1个字符后驱动该事件发生，并调用Sp\_Receiving函数准备开始接收数据。当读取到缓冲区中的回车换行符后，将接受到的数据显示在TextBox控件中。若在超时以后仍未读取到回车换行符，则执行空操作后退出接受程序，退出前清空接收缓冲区。既本窗口程序仅接受并显示以回车换行结尾的数据包。



#### 数据发送函数

数据发送过程由按键按下事件驱动调用。判断是否发送新行（本次实验中必须勾选）后是否为对话模式后，发送数据。



#### 其他子函数

其他子函数包括PC可用串口读取，串口打开，串口关闭，及相关事件处理函数。其具体实现由附件2给出。

### 实验过程

#### 串口有关参数设置

表 1

|  |  |  |
| --- | --- | --- |
| 串口参数约定 | | |
|  | 设备A（PC） | 设备B（MCU） |
| COM编号 | COM1 | COM2 |
| 波特率（bps） | 9600 | |
| 数据位 | 8 | |
| 校验位 | 无 | |
| 停止位 | 1 | |

按照表1设置串口参数如下。

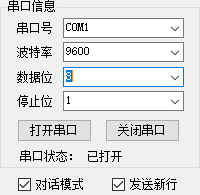


图 8

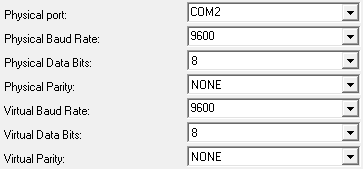
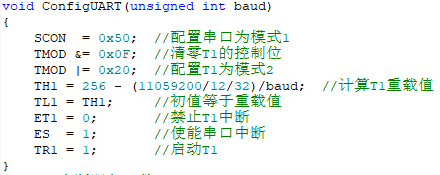


图 9

左图为窗口程序设置，右图为仿真软件中对与MCU接口的DB-9接头的属性设置。MCU中的配置由配置相关寄存器完成，如下所示。



#### 启动MCU仿真

点击下图第一个按钮启动仿真。



图 10

出现如下图所示内容说明仿真成功执行。



图 11

#### 打开上位机串口

点击打开串口按键，若在串口状态处显示已打开，则说明上位机以成功打开串口。



图 12

#### 输入信息并发送

在发送输入下的文本框中输入任意字符串（长度不可超出单片机接收缓冲区长度-2）后，点击右侧发送按钮。此处发送“This is a test！”。

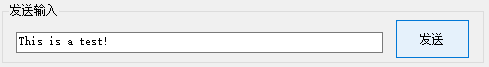


图 13

#### 单片机回传数据

上位机发送数据“This is a test！”之后，单片机回传“MCU received: This is a test!”并显示在接收显示窗口中，如下图所示。

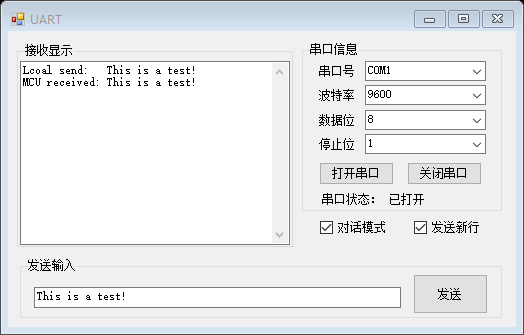


图 14

#### 单片机接收/发送

在步骤e）执行之前，单片机接收到的内容和发送的内容如下图所示。

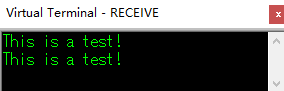


图 15

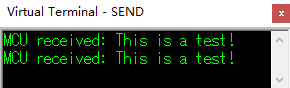


图 16

# TCP通信

## Socket套接字

### 简介

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个socket。

套接字（socket）是通信的基石，是支持TCP/IP协议的网络通信的基本操作单元。它是网络通信过程中端点的抽象表示，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。

Socket是应用层与TCP/IP协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket其实就是一个门面模式，它把复杂的TCP/IP协议族隐藏在Socket接口后面，对用户来说，一组简单的接口就是全部，让Socket去组织数据，以符合指定的协议。

### 分类

TCP/IP的socket提供下列三种类型套接字。

#### 流式套接字（SOCK\_STREAM）

提供了一个面向连接（TCP）、可靠的数据传输服务，数据无差错、无重复地发送，且按发送顺序接收。内设流量控制，避免数据流超限；数据被看作是字节流，无长度限制。文件传送协议（FTP）即使用流式套接字。

#### 数据报式套接字（SOCK\_DGRAM）

提供了一个无连接服务（UDP）。数据包以独立包形式被发送，不提供无错保证，数据可能丢失或重复，并且接收顺序混乱。网络文件系统（NFS）使用数据报式套接字。

#### 原始式套接字（SOCK\_RAW）

该接口允许对较低层协议，如IP、ICMP直接访问。常用于检验新的协议实现或访问现有服务中配置的新设备。

## TCP简介

TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议。利用TCP协议进行通信时，首先要建立通信双方的连接。一旦连接建立完成，就可以进行通信。TCP提供了数据确认和数据重传机制，保证了发送的数据一定能到达通信的对方。

## TCP通信过程

TCP通信过程如下图所示。

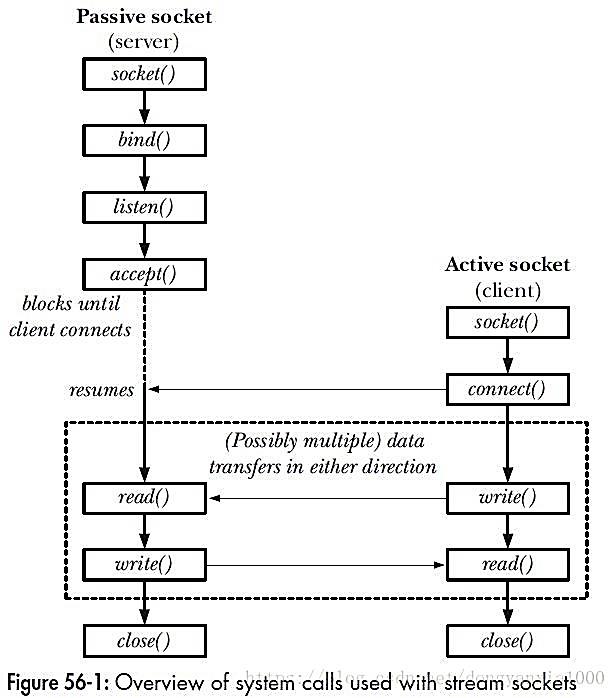


图 17

服务器端先初始化Socket，然后与端口绑定(bind)，对端口进行监听(listen)，调用(accept)阻塞，等待客户端连接。在这时如果有个客户端初始化一个Socket，然后连接服务器(connect)，如果连接成功，这时客户端与服务器端的连接就成功建立。客户端发送数据请求，服务器端接收请求并处理请求，然后把回应数据发送给客户端，客户端读取数据，最后关闭连接，一次交互结束。

## TCP服务器程序流程

### 建立连接阶段

* 调用socket()，分配文件描述符，即监听套接字。
* 调用bind()，将套接字与本地IP地址和端口绑定。
* 调用listen()，监听特定端口，socket()创建的套接字是主动的，调用listen使得该文件描述符为监听套接字，变主动为被动。
* 调用accept()，阻塞等待客户端连接。

### 数据交互阶段

* 调用read()，阻塞等待客户端发送请求，收到请求后从read()返回，处理客户端请求。
* 调用write()，将处理结果发送给客户端，然后继续调用read()等待客户端请求。

### 关闭连接

* 当read()返回0的时候，说明客户端发来了FIN数据包，即关闭连接，也会调用close()关闭连接套接字和监听套接字。

## TCP客户端程序流程

### 建立连接阶段

* 调用socket()，分配文件描述符。
* 调用connect()，向服务器发送建立连接请求。

### 数据交互阶段

* 调用write()，将请求发送给服务器。
* 调用read()，阻塞等待服务器应答。

### 关闭连接

* 当没有数据发送的时候，调用close()关闭连接套接字，即关闭连接，向服务器发送FIN数据报。

## 通信实验

### 实验环境的搭建

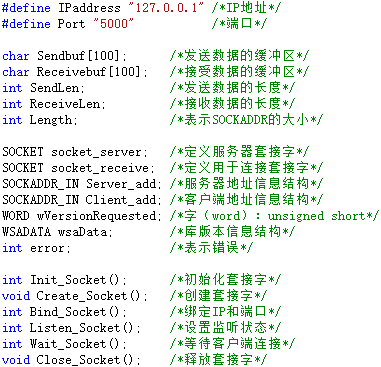
在编写实验程序前，须在编译器中添加库文件ws2\_32.lib。可通过编译器软件设置实现，也可通过包含如下代码实现：#pragma comment(lib,"ws2\_32.lib")

### 实验内容说明

本次实验包括服务器端程序和客户端程序，由C语言编写的控制台程序实现。程序顺序执行。实验实现客户端程序和服务器端程序的通信（字符串）。

### 服务器端程序的编制

#### 程序变量定义及函数声明



#### 程序主函数

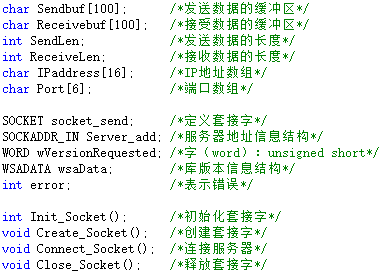


#### 其他子函数

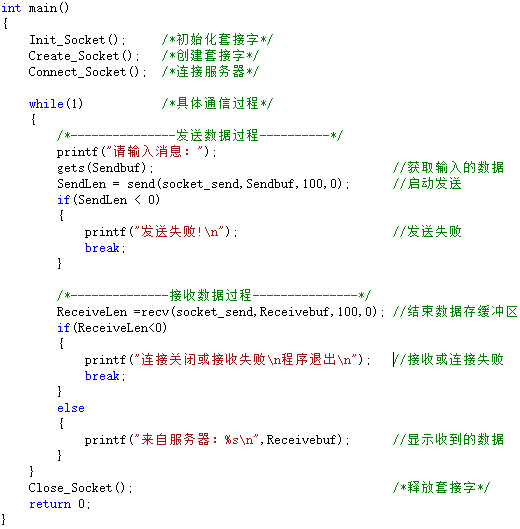
其他子函数包括用于初始化套接字，创建套接字，绑定IP和端口，设置监听状态，等待客户端发起连接的程序，其中分别调用了Socket中提供的功能函数，但是加入了一些简单的显示和错误处理。其具体实现由附件3给出。

### 客户端程序的编制

#### 程序变量定义及函数声明



#### 程序主函数



#### 其他子函数

其他子函数包含用于初始化套接字，创建套接字，连接服务器及释放套接字的函数。为了适应在本地实验和同网段实验的需求，连接服务器的子函数Connect\_Socket()中从键盘获取输入的IP地址和端口实现连接。其具体实现由附件4给出。

### 实验过程

#### 启动服务器端程序并等待连接

#### 启动客户端程序并发起连接

输入服务器IP地址和端口后即可发起连接。

#### 客户端发送消息

输入消息并回车即可发送。

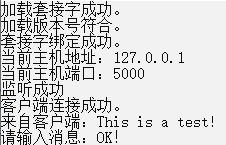


图 18

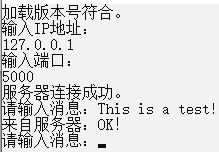


图 19

#### 服务器端发送消息

输入消息并回车即可发送。

注意：由于程序顺序/循环执行，非事件驱动结构，因此必须在接收到对方发送的数据后，己方方可发送。

# UDP通信

## UDP简介

UDP是无连接的不可靠的传输协议。采用UDP进行通信时，不需要建立连接，可以直接向一个IP地址发送数据，但是不能保证对方能收到。

对于基于UDP面向无连接的套接字编程来说，服务器端和客户端这种概念不是特别的严格。可以把服务器端称为接收端，客户端就是发送数据的发送端。

## UDP通信过程

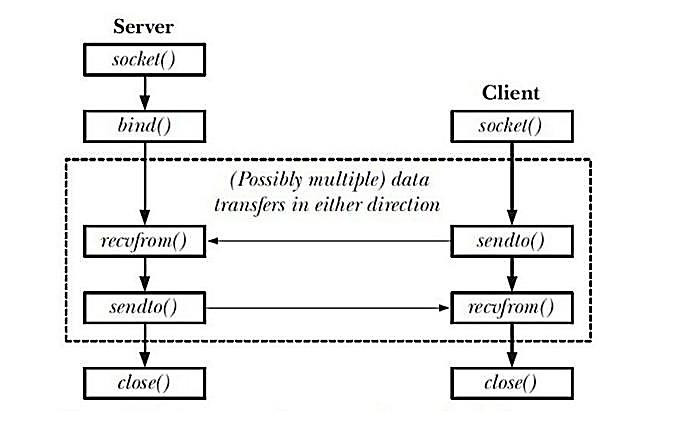


图 20

服务器端先初始化Socket，然后与端口绑定(bind)，在这时如果有个客户端初始化一个Socket，客户端发送数据请求，服务器端接收请求并处理请求，然后把回应数据发送给客户端，客户端读取数据，一次交互结束。

注意到，在进行端口绑定(bind)之后，服务器端不需要对端口进行监听(listen)，也不需要调用等待连接(accept)阻塞，等待客户端连接。而客户端无需使用连接(connect)事先与服务器建立连接。

## UDP服务器程序流程

* 建立套接字文件描述符，使用函数socket()，生成套接字文件描述符。
* 设置服务器地址和侦听端口，初始化要绑定的网络地址结构。
* 绑定侦听端口，使用bind()函数，将套接字文件描述符和一个地址类型变量进行绑定。
* 接收客户端的数据，使用recvfrom()函数接收客户端的网络数据。
* 向客户端发送数据，使用sendto()函数向服务器主机发送数据。
* 关闭套接字，使用close()函数释放资源。UDP协议的客户端流程.

## UDP客户端程序流程

* 建立套接字文件描述符，socket()。
* 设置服务器地址和端口，struct sockaddr。
* 向服务器发送数据，sendto()。
* 接收服务器的数据，recvfrom()。
* 关闭套接字，close()。

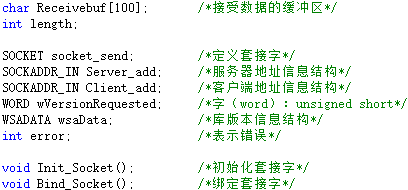
## 通信实验

### 实验内容说明

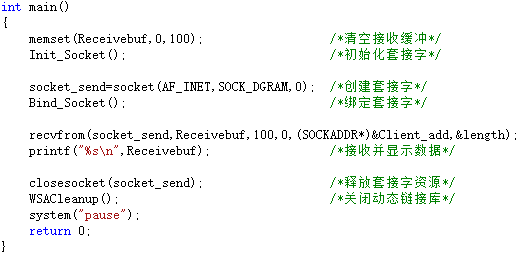
本次实验进行了更为简单的验证次实验启动服务器后监听某端口后，由客户端发送一固定数据给服务器，服务器接收并显示后。双方释放套接字并结束程序。

### 服务器端程序的编制

#### 程序变量定义及函数声明



#### 程序主函数

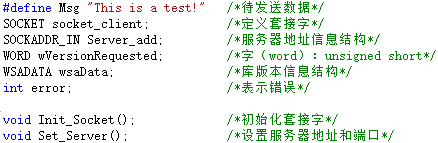


#### 其他子函数

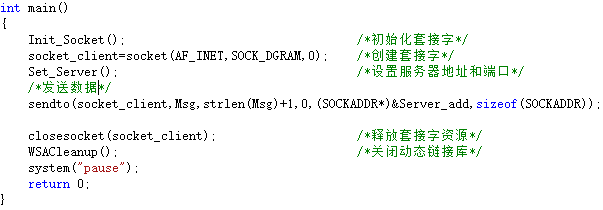
其他子函数包括用于初始化套接字，绑定IP和端口的函数，其中分别调用了Socket中提供的功能函数。其具体实现由附件5给出。

### 客户端程序的编制

#### 程序变量定义及函数声明



#### 程序主函数



#### 其他子函数

其他子函数包括用于初始化套接字，设置服务器IP和端口的函数，其中分别调用了Socket中提供的功能函数。其具体实现由附件6给出。

### 实验过程

#### 启动服务器并监听端口

#### 启动客户端向服务器发送

#### 服务器接收并显示

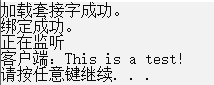


图 21

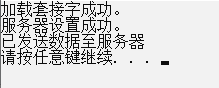


图 22

# 附件

## 附件一（单片机程序）

1. #include <reg52.h>
2. #include <string.h>
4. #define Uart\_Buf\_Max 50                          //接收缓冲区长度
6. bit ReceiveOK = 0;                               //接收完成标志位
8. unsigned **char** code Head[] = {"MCU received: "};  //Head数组定义
9. unsigned **char** data RX\_Buffer[Uart\_Buf\_Max];      //接收缓冲区定义
11. **void** ConfigUART(unsigned **int** baud);              //串行口配置函数
12. **void** SendData(unsigned **char** ch);                 //字符发送函数
13. **void** SendString(**char** \*s);                        //字符串发送函数
15. /\* 主函数 \*/
16. **void** main()
17. {
18. EA = 1;                                      //使能总中断
19. ConfigUART(9600);                            //配置串行口工作模式及参数
20. memset(RX\_Buffer,0,Uart\_Buf\_Max);            //首次运行时清空接收缓冲区
21. **while**(1)
22. {
23. **while**(ReceiveOK)                         //等待接收完成
24. {
25. ReceiveOK = 0;                       //清零接收完成标志位
26. SendString(Head);                    //发送Head部分
27. SendString(RX\_Buffer);               //发送接收到的数据
28. memset(RX\_Buffer,0,Uart\_Buf\_Max);    //发送结束后清空接收缓冲区
29. }
30. }
31. }
32. /\* 串口配置函数，baud-通信波特率 \*/
33. **void** ConfigUART(unsigned **int** baud)
34. {
35. SCON  = 0x50;                                //配置串口为模式1
36. TMOD &= 0x0F;                                //清零T1的控制位
37. TMOD |= 0x20;                                //配置T1为模式2
38. TH1 = 256 - (11059200/12/32)/baud;           //计算T1重载值
39. TL1 = TH1;                                   //初值等于重载值
40. ET1 = 0;                                     //禁止T1中断
41. ES  = 1;                                     //使能串口中断
42. TR1 = 1;                                     //启动T1
43. }
44. /\* UART中断服务函数 \*/
45. **void** InterruptUART() interrupt 4
46. {
47. **static** unsigned **char** data RX\_Count = 0;      //静态计数变量
48. **if**(RI)                                       //接收到字节
49. {
50. RI = 0;                                  //清零接收中断标志位
51. RX\_Buffer[RX\_Count++] = SBUF;            //接收字节存缓冲区
52. **if**(RX\_Count >= Uart\_Buf\_Max)             //判断溢出
53. {
54. RX\_Count = 0;                        //溢出后覆盖原数据
55. }
56. **if**(SBUF == '\n')                         //判断帧结束
57. {
58. ReceiveOK = 1;                       //接收完成
59. RX\_Count = 0;                        //清零计数
60. }
61. }
62. **if**(TI)                                       //字节发送完毕
63. {
64. TI = 0;                                  //清零发送中断标志位
65. }
66. }
67. /\* UART字符发送函数 \*/
68. **void** SendData(unsigned **char** ch)
69. {
70. SBUF = ch;                                   //启动发送
71. **while**(!TI);                                  //等待结束
72. }
73. /\* UART字符串发送函数 \*/
74. **void** SendString(**char** \*s)
75. {
76. **while**(\*s)                                    //循环发送
77. {
78. SendData(\*s++);
79. }
80. }

## 附件二（VB.Net程序）

1. **Imports** System.IO.Ports
2. **Imports** System.Text
4. **Public** **Class** Form1
6. '触发接收事件
7. **Public** **Sub** Sp\_DataReceived(**ByVal** sender **As** **Object**, **ByVal** e **As** System.IO.Ports.SerialDataReceivedEventArgs) **Handles** SerialPort1.DataReceived
8. **Me**.Invoke(**New** EventHandler(**AddressOf** Sp\_Receiving))    '调用接收数据函数
9. **End** **Sub**
11. '接收数据（等待新行）
12. **Private** **Sub** Sp\_Receiving(**ByVal** sender **As** **Object**, **ByVal** e **As** EventArgs)
13. **Dim** strIncoming **As** **String**                              '存储接收的数据
14. **Try**
15. strIncoming = SerialPort1.ReadLine() & vbCrLf      '等待接收到新行
16. SerialPort1.DiscardInBuffer()                      '清空输入缓冲区
17. TextBox\_Recieve\_Send.Text = \_
18. TextBox\_Recieve\_Send.Text + strIncoming            '显示接收的数据
19. **Catch**
21. **End** **Try**
22. SerialPort1.DiscardInBuffer()                          '清空输入缓冲区
23. **End** **Sub**
25. **Private** **Sub** Form1\_Load(**ByVal** sender **As** System.**Object**, **ByVal** e **As** System.EventArgs) **Handles** **MyBase**.Load
26. **Me**.KeyPreview = **True**                                   '注册窗体的键盘事件
27. GetSerialPortNames()                                   '获取系统可用串口号
28. **End** **Sub**
30. **Private** **Sub** Form1\_Closing(**ByVal** sender **As** System.**Object**, **ByVal** e **As** System.EventArgs) **Handles** **MyBase**.Closing
31. **If** SerialPort1.IsOpen **Then**
32. SerialPort1.Close()                                '如果串口打开则关闭
33. **End** **If**
34. **End** **Sub**
36. **Sub** GetSerialPortNames()                                   '计算机串口读取
37. **For** **Each** sp **As** **String** **In** My.Computer.Ports.SerialPortNames
38. ComboBox\_ComList.Items.Add(sp)                     '添加串口号
39. **Next**
40. ComboBox\_ComList.Sorted = **True**
41. ComboBox\_ComList.SelectedIndex = 0
42. **End** **Sub**
44. **Sub** PortSet()                                              '串口设置
45. SerialPort1.PortName = ComboBox\_ComList.Text           '串口编号设置
46. SerialPort1.BaudRate = ComboBox\_BaudRate.Text          '波特率设置
47. SerialPort1.DataBits = ComboBox\_DataBits.Text          '数据位设置
48. SerialPort1.StopBits = ComboBox\_StopBits.Text          '停止位设置
49. SerialPort1.Encoding = System.Text.Encoding.**Default**    '编码格式设置
50. SerialPort1.ReadTimeout = 500                          '超时时间设置
51. **End** **Sub**
53. **Sub** PortOpen()
54. PortSet()                                              '串口设置
55. **Try**
56. SerialPort1.Open()                                 '尝试打开串口
57. **Catch** ex **As** UnauthorizedAccessException
58. MsgBox("串口被占用或串口错误！", MsgBoxStyle.Information, "提示！")
59. **End** **Try**
60. Label\_ConnectStatus.Text = "已打开"
61. **End** **Sub**
63. **Sub** PortClose()
64. **Try**
65. SerialPort1.Close()                                '尝试关闭串口
66. **Catch** ex **As** Exception
67. MsgBox("串口未打开或串口异常！", MsgBoxStyle.Information, "提示！")
68. **End** **Try**
69. Label\_ConnectStatus.Text = "已关闭"
70. **End** **Sub**
72. **Private** **Sub** Button\_Connect\_Click(**ByVal** sender **As** System.**Object**, **ByVal** e **As** System.EventArgs) **Handles** Button\_Connect.Click
73. PortOpen()                                             '打开串口
74. **End** **Sub**
76. **Private** **Sub** Button\_Disconnect\_Click(**ByVal** sender **As** System.**Object**, **ByVal** e **As** System.EventArgs) **Handles** Button\_Disconnect.Click
77. PortClose()                                            '关闭串口
78. **End** **Sub**
80. **Private** **Sub** Button\_Send\_Click(**ByVal** sender **As** System.**Object**, **ByVal** e **As** System.EventArgs) **Handles** Button\_Send.Click
81. Send()                                                 '启动发送
82. **End** **Sub**
84. **Sub** Send()
85. **Dim** Text **As** **String**                                     '待发送数据
86. **If** CheckBox\_Send\_New\_Row.Checked = **True** **Then**           '如果发送新行被勾选
87. Text = TextBox\_Send.Text & vbCrLf                  '数据后增加回车换行
88. **Else**
89. Text = TextBox\_Send.Text                           '否则不做处理
90. **End** **If**
91. **If** CheckBox\_Com\_Mode.Checked = **True** **Then**               '如果对话模式被勾选
92. TextBox\_Recieve\_Send.Text = \_
93. TextBox\_Recieve\_Send.Text & "Lcoal send:   " \_
94. & Text                                             '显示发送的数据
95. **End** **If**
96. **Try**
97. SerialPort1.Write(Text)                            '启动数据发送过程
98. **Catch** ex **As** Exception
99. MessageBox.Show(ex.Message)                        '错误处理显示消息
100. **End** **Try**
101. **End** **Sub**
103. **Private** **Sub** Form1\_KeyDown(**ByVal** sender **As** **Object**, **ByVal** e **As** System.Windows.Forms.KeyEventArgs) **Handles** **Me**.KeyDown
104. **If** e.KeyCode = Keys.Enter **Then**                         '回车键检测
105. Send()                                             '启动发送
106. **End** **If**
107. **End** **Sub**
109. **Private** **Sub** TextBox\_Recieve\_Send\_TextChanged(**ByVal** sender **As** System.**Object**, **ByVal** e **As** System.EventArgs) **Handles** TextBox\_Recieve\_Send.TextChanged
110. TextBox\_Recieve\_Send.SelectionStart = Len(TextBox\_Recieve\_Send.Text)
111. TextBox\_Recieve\_Send.ScrollToCaret()                   '接收文本框滚动控制
112. **End** **Sub**
113. **End** **Class**

## 附件三（TCP Server程序）

1. #include<stdio.h>
2. #include<winsock.h>       /\*引入winsock头文件\*/
3. #pragma comment(lib,"ws2\_32.lib")
5. #define IPaddress "127.0.0.1" /\*IP地址\*/
6. #define Port "5000"           /\*端口\*/
8. **char** Sendbuf[100];      /\*发送数据的缓冲区\*/
9. **char** Receivebuf[100];   /\*接受数据的缓冲区\*/
10. **int** SendLen;            /\*发送数据的长度\*/
11. **int** ReceiveLen;         /\*接收数据的长度\*/
12. **int** Length;             /\*表示SOCKADDR的大小\*/
14. SOCKET socket\_server;   /\*定义服务器套接字\*/
15. SOCKET socket\_receive;  /\*定义用于连接套接字\*/
16. SOCKADDR\_IN Server\_add; /\*服务器地址信息结构\*/
17. SOCKADDR\_IN Client\_add; /\*客户端地址信息结构\*/
18. **WORD** wVersionRequested; /\*字（word）：unsigned short\*/
19. WSADATA wsaData;        /\*库版本信息结构\*/
20. **int** error;              /\*表示错误\*/
22. **int** Init\_Socket();      /\*初始化套接字\*/
23. **void** Create\_Socket();   /\*创建套接字\*/
24. **int** Bind\_Socket();      /\*绑定IP和端口\*/
25. **int** Listen\_Socket();    /\*设置监听状态\*/
26. **int** Wait\_Socket();      /\*等待客户端连接\*/
27. **void** Close\_Socket();    /\*释放套接字\*/
29. **int** main()
30. {
31. Init\_Socket();                                            /\*初始化套接字\*/
32. Create\_Socket();                                          /\*创建套接字\*/
33. Bind\_Socket();                                            /\*绑定IP和端口\*/
34. Listen\_Socket();                                          /\*设置监听状态\*/
35. Wait\_Socket();                                            /\*等待客户端连接\*/
37. **while**(1)            /\*具体通信过程\*/
38. {
39. /\*--------接收数据---------\*/
40. ReceiveLen =recv(socket\_receive,Receivebuf,100,0);    //接收数据存缓冲区
41. **if**(ReceiveLen<0)                                      //连接或接收失败
42. {
43. printf("客户端中断连接或接收失败\n程序退出\n");
44. **break**;
45. }
46. **else**
47. {
48. printf("来自客户端：%s\n",Receivebuf);            //显示接收到的数据
49. }
51. /\*--------发送数据---------\*/
52. printf("请输入消息：");
53. gets(Sendbuf);                                        //获取输入的数据
54. SendLen=send(socket\_receive,Sendbuf,100,0);           //启动发送
55. **if**(SendLen<0)
56. {
57. printf("发送失败。\n");                           //本次发送失败
58. **break**;
59. }
60. }
61. Close\_Socket();                                           /\*释放连接\*/
62. **return** 0;
63. }
65. **int** Init\_Socket()
66. {
67. /\*------------初始化套接字库---------------\*/
68. /\*定义版本类型。将两个字节组合成一个字，前面是第字节，后面是高字节\*/
69. wVersionRequested = MAKEWORD( 2, 2 );
70. /\*加载套接字库，初始化Ws2\_32.dll动态链接库\*/
71. error = WSAStartup( wVersionRequested, &wsaData);
72. **if**(error!=0)
73. {
74. printf("加载套接字失败。\n");
75. **return** 0;                                             /\*程序结束\*/
76. }
77. **else**
78. {
79. printf("加载套接字成功。\n");
80. }
81. /\*判断请求加载的版本号是否符合要求\*/
82. **if**(LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion ) != 2)
83. {
84. WSACleanup( );                                        /\*不符合，关闭套接字库\*/
85. **return** 0;                                             /\*程序结束\*/
86. }
87. **else**
88. {
89. printf("加载版本号符合。\n");
90. }
91. **return** 1;
92. }
94. **void** Create\_Socket()
95. {
96. /\*------------设置连接地址-----------------\*/
97. Server\_add.sin\_family=AF\_INET;/\*地址家族，对于必须是AF\_INET，注意只有它不是网络网络字节顺序\*/
98. Server\_add.sin\_addr.S\_un.S\_addr=inet\_addr(IPaddress);     /\*主机地址\*/
99. Server\_add.sin\_port=htons(atoi(Port));/\*端口号\*/
101. /\*------------创建套接字-------------------\*/
102. /\*AF\_INET表示指定地址族，SOCK\_STREAM表示流式套接字TCP，特定的地址家族相关的协议。\*/
103. socket\_server=socket(AF\_INET,SOCK\_STREAM,0);
104. }
106. **int** Bind\_Socket()
107. {
108. /\*---绑定套接字到本地的某个地址和端口上----\*/
109. /\*socket\_server为套接字，(SOCKADDR\*)&Server\_add为服务器地址\*/
110. **if**(bind(socket\_server,(SOCKADDR\*)&Server\_add,**sizeof**(SOCKADDR) )==SOCKET\_ERROR)
111. {
112. printf("绑定失败。\n");
113. **return** 0;
114. }
115. **else**
116. {
117. printf("套接字绑定成功。\n");
118. printf("当前主机地址：");
119. printf(IPaddress);
120. printf("\n当前主机端口：");
121. printf(Port);
122. printf("\n");
123. }
124. **return** 1;
125. }
127. **int** Listen\_Socket()
128. {
129. /\*------------设置套接字为监听状态---------\*/
130. /\*监听状态，为连接做准备，最大等待的数目为5\*/
131. **if**(listen(socket\_server,5)<0)
132. {
133. printf("监听失败\n");
134. **return** 0;
135. }
136. **else**
137. {
138. printf("监听成功\n");
139. **return** 1;
140. }
141. }
143. **int** Wait\_Socket()
144. {
145. /\*------------接受连接---------------------\*/
146. Length=**sizeof**(SOCKADDR);
147. /\*接受客户端的发送请求,等待客户端发送connect请求\*/
148. socket\_receive=accept(socket\_server,(SOCKADDR\*)&Client\_add,&Length);
149. **if**(socket\_receive==SOCKET\_ERROR)
150. {
151. printf("客户端连接失败。");
152. **return** 0;
153. }
154. **else**
155. {
156. printf("客户端连接成功。\n");
157. **return** 1;
158. }
159. }
161. **void** Close\_Socket()
162. {
163. /\*---------释放套接字，关闭动态库----------\*/
164. closesocket(socket\_receive);                              /\*释放客户端的套接字资源\*/
165. closesocket(socket\_server);                               /\*释放套接字资源\*/
166. WSACleanup();                                             /\*关闭动态链接库\*/
167. }

## 附件四（TCP Client程序）

1. #include<stdio.h>
2. #include<winsock.h>       /\*引入winsock头文件\*/
3. #pragma comment(lib,"ws2\_32.lib")
5. **char** Sendbuf[100];      /\*发送数据的缓冲区\*/
6. **char** Receivebuf[100];   /\*接受数据的缓冲区\*/
7. **int** SendLen;            /\*发送数据的长度\*/
8. **int** ReceiveLen;         /\*接收数据的长度\*/
9. **char** IPaddress[16];     /\*IP地址数组\*/
10. **char** Port[6];           /\*端口数组\*/
12. SOCKET socket\_send;     /\*定义套接字\*/
13. SOCKADDR\_IN Server\_add; /\*服务器地址信息结构\*/
14. **WORD** wVersionRequested; /\*字（word）：unsigned short\*/
15. WSADATA wsaData;        /\*库版本信息结构\*/
16. **int** error;              /\*表示错误\*/
18. **int** Init\_Socket();      /\*初始化套接字\*/
19. **void** Create\_Socket();   /\*创建套接字\*/
20. **void** Connect\_Socket();  /\*连接服务器\*/
21. **void** Close\_Socket();    /\*释放套接字\*/
23. **int** main()
24. {
25. Init\_Socket();                                      /\*初始化套接字\*/
26. Create\_Socket();                                    /\*创建套接字\*/
27. Connect\_Socket();                                   /\*连接服务器\*/
29. **while**(1)                                            /\*具体通信过程\*/
30. {
31. /\*---------------发送数据过程----------\*/
32. printf("请输入消息：");
33. gets(Sendbuf);                                  //获取输入的数据
34. SendLen = send(socket\_send,Sendbuf,100,0);      //启动发送
35. **if**(SendLen < 0)
36. {
37. printf("发送失败!\n");                      //发送失败
38. **break**;
39. }
41. /\*--------------接收数据过程---------------\*/
42. ReceiveLen =recv(socket\_send,Receivebuf,100,0); //结束数据存缓冲区
43. **if**(ReceiveLen<0)
44. {
45. printf("连接关闭或接收失败\n程序退出\n");   //接收或连接失败
46. **break**;
47. }
48. **else**
49. {
50. printf("来自服务器：%s\n",Receivebuf);      //显示收到的数据
51. }
52. }
53. Close\_Socket();                                     /\*释放套接字\*/
54. **return** 0;
55. }
57. **int** Init\_Socket()
58. {
59. /\*------------初始化套接字库---------------\*/
60. /\*定义版本类型。将两个字节组合成一个字，前面是第字节，后面是高字节\*/
61. wVersionRequested = MAKEWORD( 2, 2 );
62. /\*加载套接字库，初始化Ws2\_32.dll动态链接库\*/
63. error = WSAStartup( wVersionRequested, &wsaData);
64. **if**(error!=0)
65. {
66. printf("加载套接字失败。\n");
67. **return** 0;                                       /\*程序结束\*/
68. }
69. **else**
70. {
71. printf("加载套接字成功。\n");
72. }
73. /\*判断请求加载的版本号是否符合要求\*/
74. **if**(LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion ) != 2)
75. {
76. WSACleanup( );                                  /\*不符合，关闭套接字库\*/
77. **return** 0;                                       /\*程序结束\*/
78. }
79. **else**
80. {
81. printf("加载版本号符合。\n");
82. }
83. }
85. **void** Create\_Socket()
86. {
87. /\*-------------进行连接服务器--------------\*/
88. /\*客户端创建套接字，但是不需要绑定的，只需要和服务器建立起连接就可以了。\*/
89. /\*socket\_sendr表示的是套接字，Server\_add服务器的地址结构\*/
90. socket\_send=socket(AF\_INET,SOCK\_STREAM,0);
91. }
93. **void** Connect\_Socket()
94. {
95. /\*------------设置服务器地址---------------\*/
96. Server\_add.sin\_family=AF\_INET;/\*地址家族，对于必须是AF\_INET，注意只有它不是网络网络字节顺序\*/
97. /\*服务器的地址，将一个点分十进制表示为IP地址，inet\_ntoa是将地址转成字符串\*/
98. puts("输入IP地址：");
99. gets(IPaddress);
100. puts("输入端口：");
101. gets(Port);
102. Server\_add.sin\_addr.S\_un.S\_addr = inet\_addr(IPaddress);
103. Server\_add.sin\_port=htons(atoi(Port));              /\*端口号\*/
104. /\*-------------创建用于连接的套接字--------\*/
105. /\*AF\_INET表示指定地址族，SOCK\_STREAM表示流式套接字TCP，特定的地址家族相关的协议。\*/
106. **if**(connect(socket\_send,(SOCKADDR\*)&Server\_add,**sizeof**(SOCKADDR)) == SOCKET\_ERROR)
107. {
108. printf("服务器连接失败。\n");
109. }
110. **else**
111. {
112. printf("服务器连接成功。\n");
113. }
114. }
116. **void** Close\_Socket()
117. {
118. /\*---------释放套接字，关闭动态库----------\*/
119. closesocket(socket\_send);   /\*释放套接字资源\*/
120. WSACleanup();               /\*关闭动态链接库\*/
121. }

## 附件五（UDP Server程序）

1. #include<stdio.h>
2. #include<winsock.h>                           /\*引入winsock头文件\*/
3. #pragma comment(lib,"ws2\_32.lib")
5. **char** Receivebuf[100];       /\*接受数据的缓冲区\*/
6. **int** length;
8. SOCKET socket\_send;         /\*定义套接字\*/
9. SOCKADDR\_IN Server\_add;     /\*服务器地址信息结构\*/
10. SOCKADDR\_IN Client\_add;     /\*客户端地址信息结构\*/
11. **WORD** wVersionRequested;     /\*字（word）：unsigned short\*/
12. WSADATA wsaData;            /\*库版本信息结构\*/
13. **int** error;                  /\*表示错误\*/
15. **void** Init\_Socket();         /\*初始化套接字\*/
16. **void** Bind\_Socket();         /\*绑定套接字\*/
18. **int** main()
19. {
20. memset(Receivebuf,0,100);                  /\*清空接收缓冲\*/
21. Init\_Socket();                             /\*初始化套接字\*/
23. socket\_send=socket(AF\_INET,SOCK\_DGRAM,0);  /\*创建套接字\*/
24. Bind\_Socket();                             /\*绑定套接字\*/
26. recvfrom(socket\_send,Receivebuf,100,0,(SOCKADDR\*)&Client\_add,&length);
27. printf("客户端：%s\n",Receivebuf);         /\*接收并显示数据\*/
29. closesocket(socket\_send);                  /\*释放套接字资源\*/
30. WSACleanup();                              /\*关闭动态链接库\*/
31. system("pause");
32. **return** 0;
33. }
35. **void** Init\_Socket()
36. {
37. /\*-------------------------初始化套接字库---------------------------\*/
38. /\*定义版本类型。将两个字节组合成一个字，前面是第字节，后面是高字节\*/
39. wVersionRequested = MAKEWORD( 2, 2 );
40. /\*加载套接字库，初始化Ws2\_32.dll动态链接库\*/
41. error = WSAStartup( wVersionRequested, &wsaData);
42. **if**(error!=0)
43. {
44. printf("加载套接字失败！\n");
45. **return** 0;                              /\*程序结束\*/
46. }
47. /\*判断请求加载的版本号是否符合要求\*/
48. **if** (LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion ) != 2)
49. {
50. WSACleanup( );                         /\*不符合，关闭套接字库\*/
51. **return** 0;                              /\*程序结束\*/
52. }
53. printf("加载套接字成功。\n");
54. }
56. **void** Bind\_Socket()
57. {
58. /\*----------------------设置服务器地址-----------------------\*/
59. Server\_add.sin\_family=AF\_INET;/\*地址家族，对于必须是AF\_INET，注意只有它不是网络网络字节顺序\*/
60. Server\_add.sin\_addr.S\_un.S\_addr = inet\_addr("127.0.0.1");
61. Server\_add.sin\_port=htons(5000);/\*端口号\*/
63. /\*绑定套接字\*/
64. bind(socket\_send,(SOCKADDR\*)&Server\_add,**sizeof**(SOCKADDR));
65. length=**sizeof**(SOCKADDR);
66. printf("绑定成功。\n正在监听\n");
67. }

## 附件六（UDP Client程序）

1. #include<stdio.h>
2. #include<winsock.h>               /\*引入winsock头文件\*/
3. #pragma comment(lib,"ws2\_32.lib")
5. #define Msg "This is a test!"   /\*待发送数据\*/
6. SOCKET socket\_client;           /\*定义套接字\*/
7. SOCKADDR\_IN Server\_add;         /\*服务器地址信息结构\*/
8. **WORD** wVersionRequested;         /\*字（word）：unsigned short\*/
9. WSADATA wsaData;                /\*库版本信息结构\*/
10. **int** error;                      /\*表示错误\*/
12. **void** Init\_Socket();             /\*初始化套接字\*/
13. **void** Set\_Server();              /\*设置服务器地址和端口\*/
14. **int** main()
15. {
16. Init\_Socket();                                 /\*初始化套接字\*/
17. socket\_client=socket(AF\_INET,SOCK\_DGRAM,0);    /\*创建套接字\*/
18. Set\_Server();                                  /\*设置服务器地址和端口\*/
19. /\*发送数据\*/
20. sendto(socket\_client,Msg,strlen(Msg)+1,0,(SOCKADDR\*)&Server\_add,**sizeof**(SOCKADDR));
21. printf("已发送数据至服务器\n");
22. closesocket(socket\_client);                    /\*释放套接字资源\*/
23. WSACleanup();                                  /\*关闭动态链接库\*/
24. system("pause");
25. **return** 0;
26. }
28. **void** Init\_Socket()
29. {
30. /\*-------------------------初始化套接字库---------------------------\*/
31. /\*定义版本类型。将两个字节组合成一个字，前面是第字节，后面是高字节\*/
32. wVersionRequested = MAKEWORD( 2, 2 );
33. /\*加载套接字库，初始化Ws2\_32.dll动态链接库\*/
34. error = WSAStartup( wVersionRequested, &wsaData);
35. **if**(error!=0)
36. {
37. printf("加载套接字失败！\n");
38. **return** 0;                                  /\*程序结束\*/
39. }
40. /\*判断请求加载的版本号是否符合要求\*/
41. **if** ( LOBYTE( wsaData.wVersion ) != 2 ||
42. HIBYTE( wsaData.wVersion ) != 2 )
43. {
44. WSACleanup( );                             /\*不符合，关闭套接字库\*/
45. **return** 0;                                  /\*程序结束\*/
46. }
47. printf("加载套接字成功。\n");
48. }
50. **void** Set\_Server()
51. {
52. /\*----------------------设置服务器地址-----------------------\*/
53. Server\_add.sin\_family=AF\_INET;/\*地址家族，对于必须是AF\_INET，注意只有它不是网络网络字节顺序\*/
54. Server\_add.sin\_addr.S\_un.S\_addr = inet\_addr("127.0.0.1");
55. Server\_add.sin\_port=htons(5000);               /\*端口号\*/
56. printf("服务器设置成功。\n");
57. }

## 附件七（单片机原理图）

