电工电子综合设计

黄永嘉　编

重庆大学电气工程学院

2004 年 7 月

目　　录

电工电子综合设计开放性实验讲义

第一章　概述

本课程是综合及开放性独立实验课程，其前导课程是《电工学》、《电子学（模电及数电）》、《单片机》、《工业 PC 及测控系统》、《TC 语言》、《电工电子综合实验》等。

本课程是工科电气类的一门通用实践课程。目的是使学生掌握当代的新型机电测控技术，从集散测控系统和集散测控网络的层次进行设计制作。

本课程给学生提供完整的集散系统和网络，让学生制作其中某部分装置，编制某一部分程序，然后让学生完成全系统全网络化的实验，以资在有限的 36 学时内掌握全系统全网络。

本课程设计了串行 BuS，设立了网络自主运作的通讯程序，网络是骨干是系统本体，其余各类站点（含各类计算机、工控机、单片机、智能仪器表）皆为从属单元，辅以专用实时操作系统，实时数据库，组态软件，使学生掌握一个高度灵活、功能齐全、实用价值极高的系统。

从《工业 PC 及测控系统》课程中知道，一套集中式测控系统可以处理若干模拟量、数字量、开关量等工业现场的各类信息，其特点是全部信息集中收集在一套计算机上同时处理。这种测控系统能适合于现场信息物理位置相对集中、现场信息要求高速同时处理、现场信息要求大容量超大容量处理等场合。现场信息并非都是高速大容量，更多的是较慢较少的信息。合理的结构设计应该把不同要求的信息分别处理，把不同物理位置的信息分别就地处理，然后用通讯网络将各点各类进行过一次处理的信息集中起来作二次处理。这样就产生了分布式测控网络系统。显然分布式测控网络系统更能适应广泛的工业现场。

本讲义采用的文献：

陈曾汉编著的《工业 PC 及测控系统》　　机械工业出版社 2004 年出版

李朝青编著的《PC 机及单片机数据通信技术》　　北航出版社 2000 年出版

第一章　测控网络结构

测控网络结构取决与其通讯方式，通常把一特定的工业现场，比如一个车间或一个工厂作为检测对象，在这个现场用工业局域网管理各检测点信息，然后再通过广域网向上级管理层联结。

如图就是一个通用的工业测控网络结构。

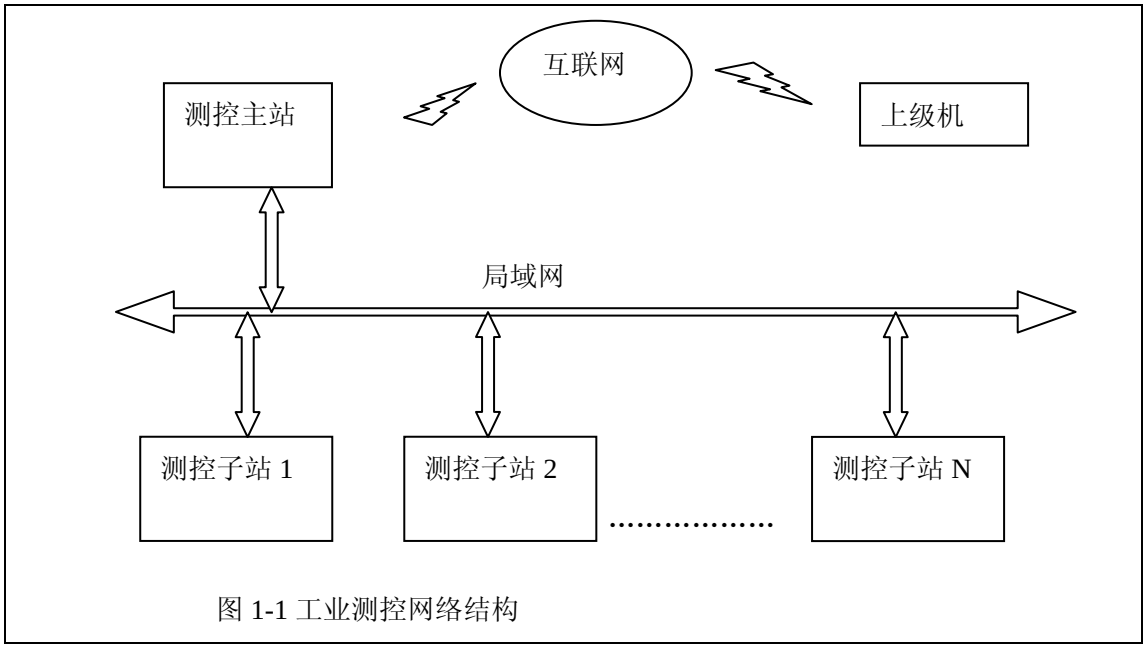


图 1-1 工业测控网络结构

图中的局域网是结构的主干，挂在网上的任一单元都必须从网上接收信息（控制命令和数据）并且向网上发布信息（应答命令和数据）。

一．局域网网络的硬件选择

1．常规网卡
2．CAN 总线（控制器局域网）
3．串行总线（RS232、RS485）

二．局域网网络的通讯协议
1．TCP/IP 协议
2．呼叫应答方式
3．广播方式

三．实验室提供的局域网网络的硬件及通讯方式

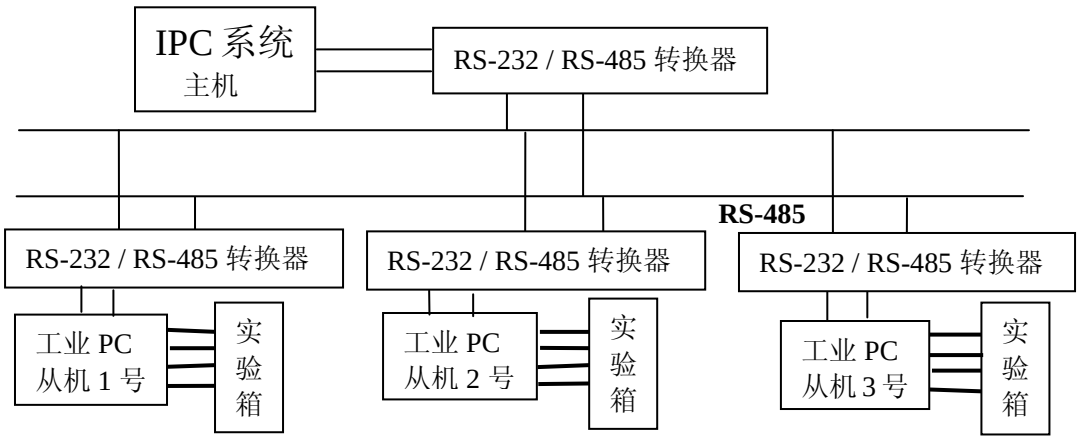


图 1-2 由四台 IPC 构成的两级 DCS 实验系统

如图是实验室在 A6220 室装备的一套局域网测控系统。其通讯网络由 RS485 串行总线构成，通过 RS232/RS485 转换器与主机和三个从机相连。通讯规约是主从呼叫应答方式，主机向三个从机分别索要信息，从机按照主机的命令向主机发送信息。全部信息（控制命令及数据流）交换通过 RS485 串行总线进行。主机用广播方式呼叫某一台从机，三台从机同时收到主机的命令呼叫，从机对命令进行识别，确认与己相关后，按呼叫命令的要求把数据发给主机。

RS485 串行总线是对 RS232 串行总线电气驱动方式的扩展，把 RS232 串行总线的 30M 直接传输距离扩大到 RS485 串行总线的 2000M，传输速度从 RS232 串行总线的 9600bit/s 提高到 RS485 串行总线的 1Mbit/s，RS485 串行总线还具有抗雷击能力及总线上带电拔插的功能。

这两种串行总线的通讯规约相同，它们的通讯程序也就相同。

通过对这套局域网测控系统的结构进行实物观察和操作运行，能够全面掌握局域网测控系统。

由于实验设备数量的限制，为了让实验者着手进行系统设计、编程、调试及运行系统，实验室为实验者设计了一种简化的局域网测控系统。该系统把从机减少为一台，在一台从机的硬件平台上仿真多台从机，从机的数据采样也用仿真手段代替。这在学习了前导课程《工业 PC 及测控系统》的基础上，不失为一种合理实用的替代方法。

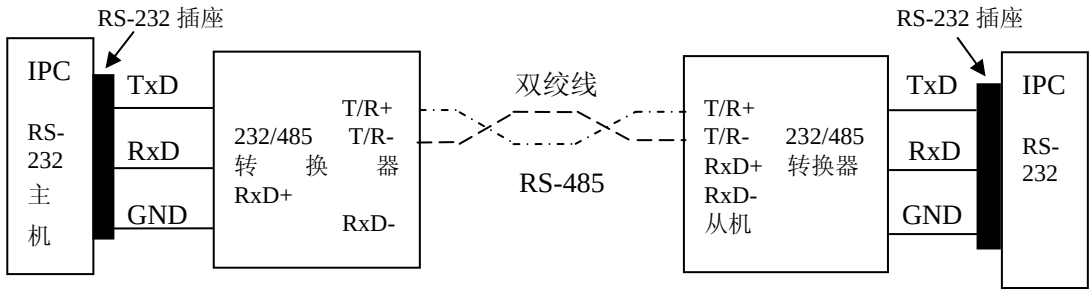


图 1-3 两台 IPC 通过 RS-232/485 实现局域网测控系统仿真

第二章 数据通信基础

在微机测控技术领域，要构成一个较大规模的测控系统，都不可避免地要采用多机系统。由于微机与微机间的距离可能是近程的(几米之内)，也可能是远程的(几百米甚至上千米)，那么信息交换的方式可能采用并行通信，也可能采用串行通信。而一般的远程通信须采用串行通信方式，现在单片机及 PC 机在结构、性能和经济上为实现远程串行通信特别是多机系统提供了很好的条件。

本书主要讲述单片机与单片机，PC 机与单片机之间串行通信技术。

2.1 串行数据通信

2.1.1 数据通信的概念

在实际工作中，计算机的 CPU 与外部设备之间常常要进行信息交换，一台计算机与其他计算机之间也往往要交换信息，所有这些信息交换均可称为数据通信。

数据通信方式有两种，即并行数据通信和串行数据通信。通常根据信息传送的距离决定采用哪种通信方式。例如，在 PC 机与外部设备(如打印机等)通信时，如果距离小于 30m，可采用并行数据通信方式；当距离大于 30m 时，则要采用串行数据通信方式。8051 单片机具有并行和串行二种基本数据通信方式。

并行数据通信是指数据的各位同时进行传送(发送或接收)的通信方式。其优点是传递速度快；缺点是数据有多少位，就需要多少根传送线。例如 8051 单片机与打印机之间的数据传送就属于并行数据通信。图 2.1—1(a)所示为 8051 单片机与外设间 8 位数据并行通信的连接方法。并行通信在位数多、传送距离又远时就不太适宜。

串行数据通信指数据是一位一位顺序传送的通信方式，它的突出优点是只需一对传送线(利用电话线就可作为传送线)，这样就大大降低了传送成本，特别适用于远距离通信；其缺点是传送速度较低。假设并行传送 N 位数据所需时间为 t，那么串行传送的时间至少为 Nt，实际上总是大于 Nt 的。图 2.1—1(b)所示为串行数据通信方式的连接方法。

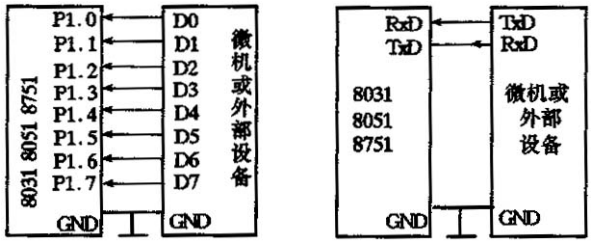


图 2.1-1 两种通信方式连接

2.1.2 串行通信的传送方式

串行通信的传送方式通常有三种：一种为单向（或单工）配置，只允许数据向一个方向传送；另一种是半双向（或半双工）配置，允许数据向两个方向中的任一方向传送，但每次只能有一站发送；第三种传送方式是全双向（全双工）配置，允许同时双向传送数据，因此，全双工配置是一对单向配置，它要求两端的通信设备都具有完整和独立的发送和接收能力。图 2.1-2 所示为串行通信中数据传送方式。

图 2.1-2（a）为单工方式，（b）为半双工方式，（c）为全双工方式。

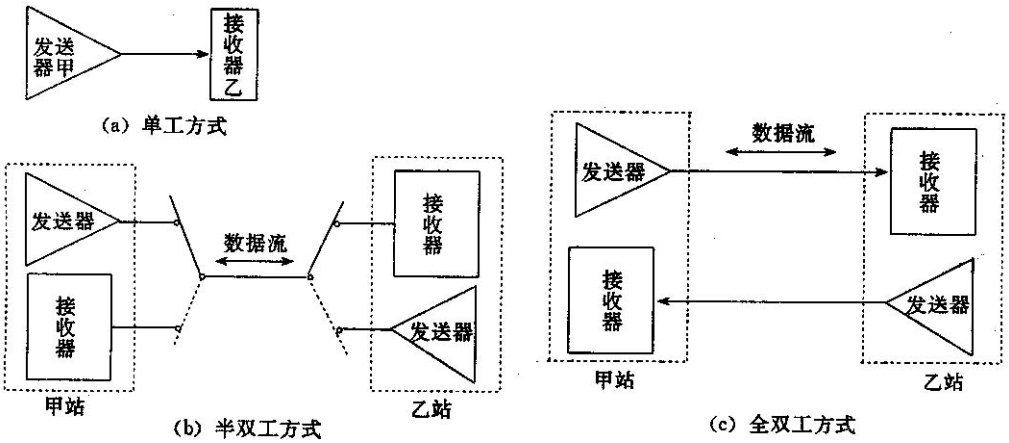


图 2.1-2 串行通信传输方式

2.1.3 异步通信和同步通信

串行通信有两种基本通信方式，即异步通信和同步通信。

1、异步通信

在异步通信中，数据是一帧一帧（包含一个字符代码或一字节数据）传送的，每一串行帧的数据格式如图 2.1-3 所示。

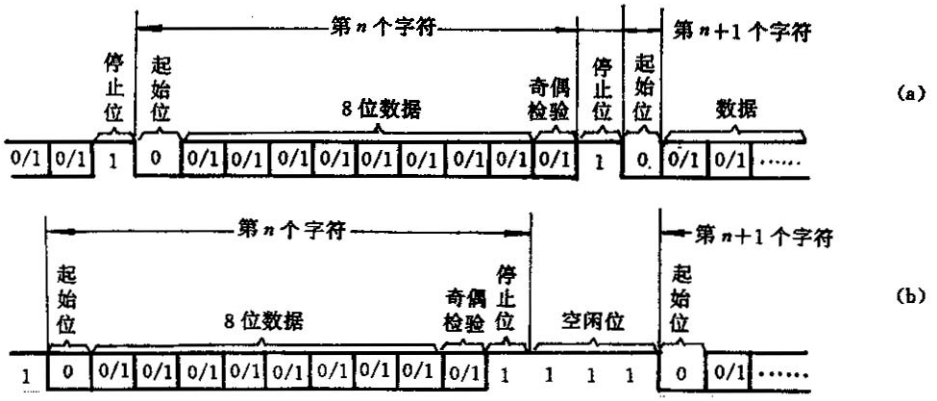


图 2.1-3 异步通信的一帧数据格式

在帧格式中，一个字符由四个部分组成：起始位、数据位、奇偶校验位和停止位。首先是一个起始位“0”，然后是 5～8 位数据(规定低位在前，高位在后)，接下来是奇偶校验位(可省略)，最后是停止位“1”。起始位“0”信号只占用一位，用来通知接收设备一个待接收的字符开始到来。线路上在不传送字符时应保持为“1”。接收端不断检测线路的状态，若连续为“1”以后又测到一个“0”，就知道发来一个新字符，应马上准备接收。字符的起始位还被用作同步接收端的时钟，以保证以后的接收能正确进行。

起始位后面紧接着是数据位，它可以是 5 位(D0~D4)、6 位、7 位或 8 位(D0~D7)。

奇偶校验(D8)只占一位，但在字符中也可以规定不同奇偶校验位，则这时这一位就可省去。也可用这一位(1 / 0)来确定这一帧中的字符所代表信息的性质(地址 / 数据等)。

停止位用来表征字符的结束，它一定是高电位(逻辑“1”)。停止位可以是 1 位、1.5 位或 2 位。接收端收到停止位后，知道上一字符已传送完毕，同时，也为接收下一个字符作好准备——只要再收到“0”就是新的字符的起始位。若停止位以后不是紧接着传送下一个字符，则让线路上保持为“1”。图 2.1—3(a)表示一个字符紧接一个字符传送的情况，上一个字符的停止位和下一个字符的起始位是紧相邻的，图 2.1—3(b)则是两个字符间有空闲位的情况，空闲位为“1，线路处于等待状态。存在空闲位正是异步通信的特征之一。

例如，规定用 ASCII 编码，字符为七位，加一个奇偶校验位、一个起始位、一个停止位，则一帧共十位。

2、同步通信

同步通信中，在数据开始传送前用同步字符来指示(常约定 1～2 个)，并由时钟来实现发送端和接收端同步，即检测到规定的同步字符后，下面就连续按顺序传送数据，直到通信告一段落。同步传送时，字符与字符之间没有间隙，也不用起始位和停止位，仅在数据块开始时用同步字符 SYNC 来指示，其数据格式如图 2.1—4 所示。

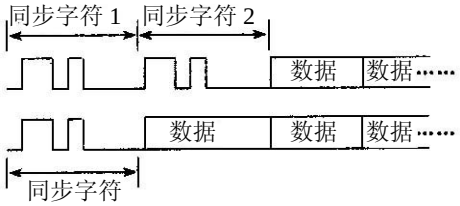


图 2.1-4 同步传送的数据格式

同步字符的插入可以是单同步字符方式或双同步字符方式，如图 2.1—4 所示。然后是连续的数据块。同步字符可以由用户约定，当然也可以采用 ASCII 码中规定的 SYN 代码，即 16H。按同方式通信时，先发送同步字符，接收方检测到同步字符后，即准备接收数据。

在同步传送时，要求用时钟来实现发送端与接收端之间的同步。为了保证接收正确无误，发送方除了传送数据外，还要把时钟信号同时传送。

同步传送的优点是可以提高传送速率(达 56kb / s 或更高)，但硬件比较复杂。

2.1.4 波特率和接收 / 发送时钟

1.波特率

波特率，即数据传送速率，表示每秒钟传送二进制代码的位数（bit），它的单位是位 / 秒(b / s)。波特率对于 CPU 与外界的通信是很重要的。假设数据传送速率是 120byte / s，而每个字符格式包含 10 个代码位(1 个起始位、1 个终止位、8 个数据位)，这时传送的波特率为：

$$10\times120\text{byte/s}=1200\text{b/s}$$

每一位代码的传送时间 t_d 为波特率的倒数。

$$t_d = \frac{1}{1200} = 0.883ms$$

波特率是衡量传输通道频宽的指标，它和传送数据的速率并不一致。如上例中，因为除掉起始位和终止位，每一个数据实际只占 8 位。所以数位的传送速率为：

$$8\times120byte/s=960b/s$$

异步通信的传递速度在 50~19200b/s 之间。常用于计算机到终端机和打印机之间的通信、直通电报以及无线电通信的数据发送等。

2、接收/发送时钟

在串行通信过程中二进制数字系列以数字信号波形的形式出现。不论接收还是发送，都必须有时钟信号对传送的数据进行定位。接收/发送时钟就是用来控制通信设备接收/发送字符数据速度的，该时钟信号通常由微机内部时钟电路产生。

在接收数据时，接收器在接收时钟的上升沿对接收数据采样，进行数据位检测；在发送数据时，发送器在发送时钟的下降沿将移位寄存器的数据串行移位输出，如图 2.1-5 和图 2.1-6 所示。

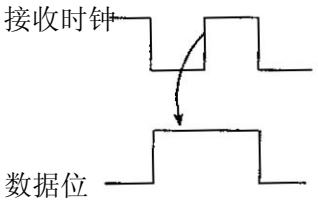


图 2.1-5　接收时钟

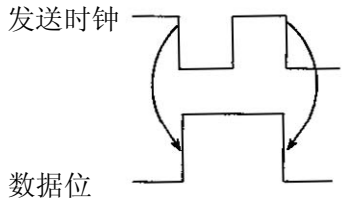


图 2.1-6 发送时钟

接收/发送时钟频率与波特率有如下关系：

收/发时钟频率=n×收/发波特率

$$收/发波特率=\frac{收/发时钟频主率}{n}$$

其中频率系数 n=1,16,64。

对于同步传送方式，必须取 n=1，即接收/发送时钟的频率等于收/发波特率。对于异步传送方式，n=1,16,64，即可以选择的接收/发送时钟频率是波特率的 1,16 或 64 倍。因此，可由要求的传送波特率及所选择的倍数 n 来确定接收/发送时钟的频率。

例如，若要求数据传送的波特率为 300b/s，则

接收/发送时钟频率=300Hz（n=1）

接收/发送时钟频率=4800Hz（n=16）

接收/发送时钟频率=19.2kHz（n=64）

接收/发送时钟的周期 T_c 与发送的数据位宽 Td 之间的关系是：

$$T_c = \frac{T_d}{n}(n = 1,16,64)$$

若取 n=16，那么异步传送接收数据实现同步的过程如下：接收器在每一个接收时钟的上升沿采样接收数据线，当发现接收数据线出现低电平时就认为是起始位的开始，以后若在连续的 8 个时钟周期(因 n=16，故 Ta=16Tc)内检澜到接收数据线仍保持为低电平，则确定它为起始位(不是干扰信号)。通过这种方法，不仅能够排除接收线上的噪声干扰，识别假起始

位；而且能够相当精确地确定起始位的中间点，从而提供一个准确的时间基准。从这个基准算起，每隔 16Tc。采样一次数据线，作为输入数据。一般来说，从接收数据线上检测到一个下降沿开始，若其低电平能保持号 nTc/2(半位时间)，则确定为起始位，其后每间隔 nTc 时间(一个数据位时间)在每个数据位的中间点采样。

由此可见，接收／发送时钟对于收／发双方之间的数据传送达到同步是至关重要的。

3、允许的波特率误差

为分析方便，假设传递的数据一帧为 10 位，若发送和接收的波特率达到理想的一致，那么接收方时钟脉冲的出现时间保证对数据的采样都发生在每位数据有效时刻的中点。如果接收一方的波特率比发送一方大或小 5%，那么对 10 位一帧的串行数据，时钟脉冲相对数据有效时刻逐位偏移，当接收到第 10 位时，积累的误差达 50%，则采样的数据已是第 10 位数据有效与无效的临界状态，这时就可能发生错位，所以 5%是最大的波特率允许误差。对于常用的 8 位、9 位和 11 位一帧的串行传送，其最大的波特率允许误差分别为 6.25%、5.56%和 4.5%。

2.2　串行通信的过程及通信协议

2.2.1　串—并转换与设备同步

两个通信设备在串行线路上实现成功的通信必须解决两个问题：一是串—并转换，即如何把要发送的并行数据串行化，把接收的串行数据并行化，二是设备同步，即同步发送设备和接收设备的工作节拍相同，以确保发送数据在接收端被正确读出。

1、串—并转换

串行通信是将计算机内部的并行数据转换成串行数据，将其通过一根通信线传送，并将接收的串行数据再转换成并行数据送到计算机中。

在计算机发送串行数据之前，计算机内部的并行数据被送入移位寄存器并一位一位地移出，将并行数据转换成串行数据，如图 2.2—1 所示。在接收数据时，来自通信线路的串行数据被送入移位寄存器，满 8 位后并行送到计算机内部，如图 2.2—2 所示。在串行通信控制电路中，串—并、并—串转换逻辑被集成在串行异步通信控制器芯片中。8051 单片机中的串行口和 PC 机中的 8250 芯片都可完成这一功能。

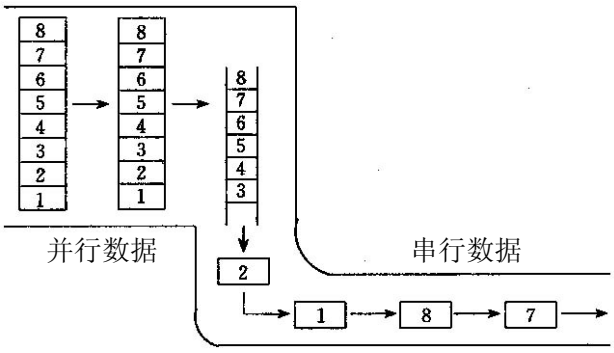


图 2.2-1 发送时的并-串转换

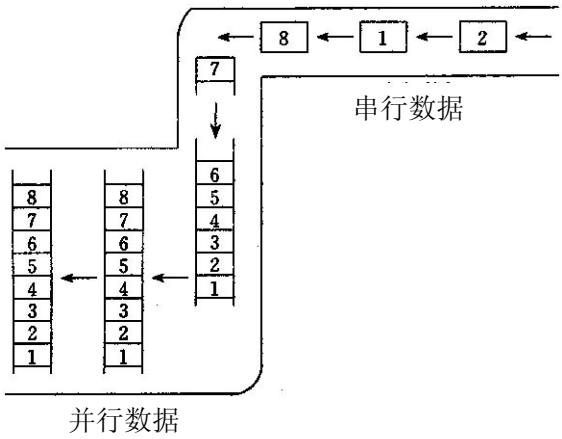


图 2.2-2 接收时的串-并转换

2、设备同步

进行串行通信的两台设备必须同步工作才能有效地检测通信线路上的信号变化，从而采样传送数据脉冲。设备同步对通信双方有两个共同要求：一是通信双方必须采用统一的编码方法；二是通信双方必须能产生相同的传送速率。

采用统一的编码方法确定了一个字符二进制表示值的位发送顺序和位串长度，当然还包括统一的逻辑电平规定，即电平信号高低与逻辑“l”和逻辑“0”的固定对应关系。

通信双方只有产生相同的传送速率，才能确保设备同步，这就要求发送设备和接收设备采用相同频率的时钟。发送设备在统一的时钟脉冲上发出数据，接收设备才能正确地检测出与时钟脉冲同步的数据信息。

2.2.2 串行通信协议

通信协议是对数据传送方式的规定，包括数据格式定义和数据位定义等。通信方式必须遵从统一的通信协议。串行通信协议包括同步协议和异步协议两种，本书只讨论异步串行通信协议。异步串行协议规定字符数据的传送格式，前面已讲过，这里不再赘述。

要想保证通信成功，通信双方必须有一系列的约定，比如：

作为发送方，必须知道什么时候发送信息、发什么、对方是否收到、收到的内容有没有错、要不要重发、怎样通知对方结束等；作为接收方，必须知道对方是否发送了信息、发的是什么、收到的信息是否有错、如果有错怎样通知对方重发、怎样判断结束等。

这种约定就叫做通信规程或协议，它必须在编程之前确定下来。要想使通信双方能够正确地交换信息和数据，在协议中对什么时候开始通信、什么时候结束通信、何时交换信息等问题都必须作出明确的规定。只有双方都正确地识别并遵守这些规定才能顺利地进行通信。

1、起始位

当通信线上没有数据被传送时处于逻辑“l”状态。当发送设备要发送一个字符数据时，首先发出一个逻辑“0”信号，这个逻辑低电平就是起始位。起始位通过通信线传向接收设备，接收设备检测到这个逻辑低电平后，就开始准备接收数据位信号。起始位所起的作用就是使设备同步，通信双方必须在传送数据位前协调同步。

2、数据位

当接收设备收到起始位后，紧接着就会收到数据位。数据位的个数可以是 5、6、7 或 8，PC 机中经常采用 7 位或 8 位数据传送，8051 串行口采用 8 位或 9 位数据传送。这些数据位被接收到移位寄存器中，构成传送数据字符。在字符数据传送过程中，数据位从最低有效位开始发送，依次在接收设备中被转换为并行数据。

3、奇偶校验位

数据位发送完之后，便可以发送奇偶校验位。奇偶校验用于有限差错检测，通信双方应

约定一致的奇偶校验方式。如果选择偶校验，那么组成数据位和奇偶位的逻辑“1”的个数必须是偶数：如果选择奇校验，那么逻辑“1”的个数必须是奇数。

4、停止位约定

在奇偶位或数据位(当无奇偶校验时)之后发送的是停止位。停止位是一个字符数据的结束标志，可以是 1 位、1.5 位或 2 位的低电平。接收设备收到停止位之后，通信线路上便又恢复逻辑“1”状态，直至下一个字符数据的起始位到来。

5、波特串设置

通信线上传送的所有位信号都保持一致的信号持续时间，每一位的宽度都由数据传送速率确定，而传送速率是以每秒多少个二进制位来度量的，这个速率叫波特率。如果数据以每秒 300 个二进制位在通信线上传送，那么这个传送速率为 300 波特。

6.软件挂钩(握手)信号约定

第三章　PC 机通信

IBMPC(简称 PC 机)是 1981 年 8 月 12 日，由计算机行业巨人 IBM 公司推出的。至今 PC 机已发展为不可取代的世界主流机型。PC 系统支持的串行通信卡符合 RS 一 232C 工业标准。PC 机这种兼容性的设计方法主要体现在其通用的接口设计上。1996 年 PC 机上开始增加 USB 万能接口。

研究开发 PC 机的通信技术，特别是在进入信息网络时代的今天就变得十分重要。

3.1　异步通信控制器(适配器)I／O 接口芯片 8250 及应用

在微机数据通信中，经常使用大规模集成串行接口电路芯片，它们的种类和型号很多，如 UART、USRT 和 USART 等。能够完成异步通信的硬件电路称为 UART，即通用异步接收器／发送器(Universal Asynchronous Receiver／Transmitter)；能够完成同步通信的硬件电路称为 USRT(Universal Synchronous Receiver／Transmitter)；既能异步又能同步通信的硬件电路称为 USART(Universal Synchronous Asynchronous Receiver／Transmitter)。

从本质上讲，所有的串行接口电路都是以并行数据形式与 CPU 接口、而以串行数据形式与外部逻辑接口。它们的基本功能是从外部逻辑接收串行数据，转换成并行数据后传送给 CPU 或者从 CPU 并行输出的数据，转变成串行数据后输出给外部逻辑。图 3.1—l 和图 3.1—2 分别给出了 UART 电路中发送和接收数据操作的情况。

从图 3.1—1 和图 3.1—2 可以看出，串行通信接口电路至少包括一个接收器和一个发送器，而接收器和发送器都分别包括一个数据寄存器和一个移位寄存器，以便实现 CPU 输出一并行一串行一发送或接收一串行一并行一 CPU 输入操作。

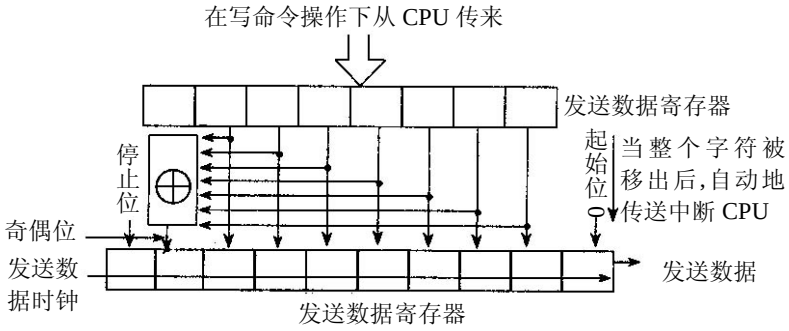


图 3.1-1 UART 发送操作

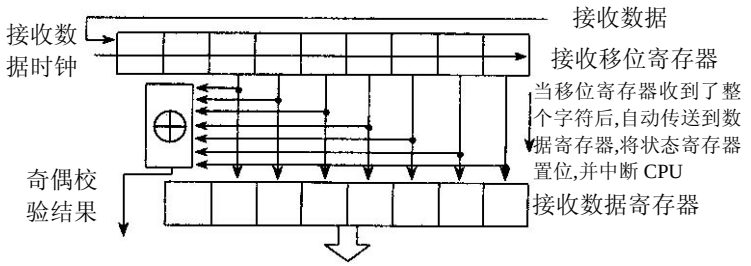


图 3.1-2 UART 接收操作

为实现串行通信，可以根据通信协议的要求，用编写程序的方法完成串行通信中数据字符的接收和发送(包括并—串转换等)，但这些比较麻烦。为了有效地实现串行通信，PC 系列及其兼容机都采用一个可编程异步串行通信接口芯片来执行异步串行通信协议。这种异步串行通信接口芯片的核心是一个大规模集成通信组件，称为通用异步接收／发送器，或简称 UART。在使用了 UART 后，异步串行通信的实现就方便多了。

作为可编程的异步串行通信芯片 UART，可根据协议的要求对其初始化。初始化后，当要发送一个数据字符时，如果 UART 发送保持寄存器为空，可用 CPU 的输入输出指令把该数据字符输出到 UART 的发送保持寄存器。UART 按初始化时设置的要求，把相应的起始位、奇偶位和停止位加到来自于 CPU 的 8 位数据上，然后接设置的波特率把这个二进制位串发送到串行通信上。同样，UART 能自动从通信线上接收串行数据，并挑出有效的数据位，转换成数据字符存入接收数据寄存器。

UART 的产品型号颇多。PC 和 XT 采用的是 INS8250 芯片，AT 采用的是 NSl6450 芯片，二者的设计思想和内部结构相似。

下面从 UART 编程的角度出发，介绍 INS8250 的组织结构及 l0 个可被 CPU 访问的 8 位寄存器。

可编程串行异步通信控制器(适配器)8250 是 IBMPC 串行通信控制器 I／O 接口电路的核心。通过对 8250 的编程，可以控制串行数据传送格式和速度在 PC 机中，并将其做成接口卡的形式，其端口地址范围 3F8H~3FFH(若是辅卡，则端口地址为 2F8~2FFH)。它可以与调制解调器配合进行远距离通信，传送波特率为 50～9 600b／s。

图 3。1—3 是 PC 机异步通信适配器接口结构框图。适配器的核心是 1NS8250 异步通信适配器芯片，它把 PC 机的并行数据转换成串行数据送往 MODEM 或外部设备；或者将 MODEM 或外部设备传来的串行数据经 RS--232C 接口接收进来，并且转换成 8 位并行数据送往 PC 机。

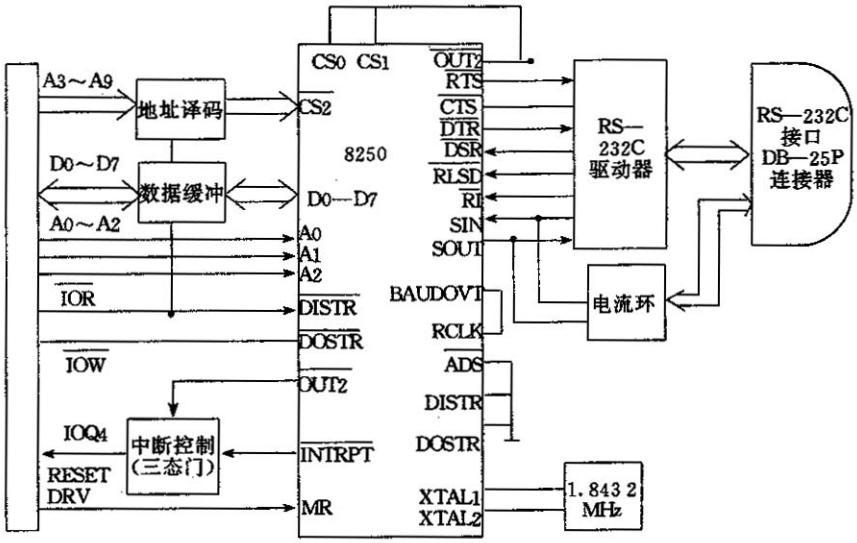


图 3.1-3 PC 机异步通信适配器接口结构框图

3.1.1 INS8250 的组成及工作原理

1．8250 芯片的组成及引脚功能

8250 芯片由各种控制逻辑和寄存器组成，主要包括 6 部分，数据输入/输出总线缓冲器和芯片内部选择和读/写控制逻辑、接收控制电路、发送控制电路、传送速度控制电路、调制解调控制电路、中断控制电路和 10 个寄存器。8250 芯片的内部结构框图如图 5.1-4 所示。

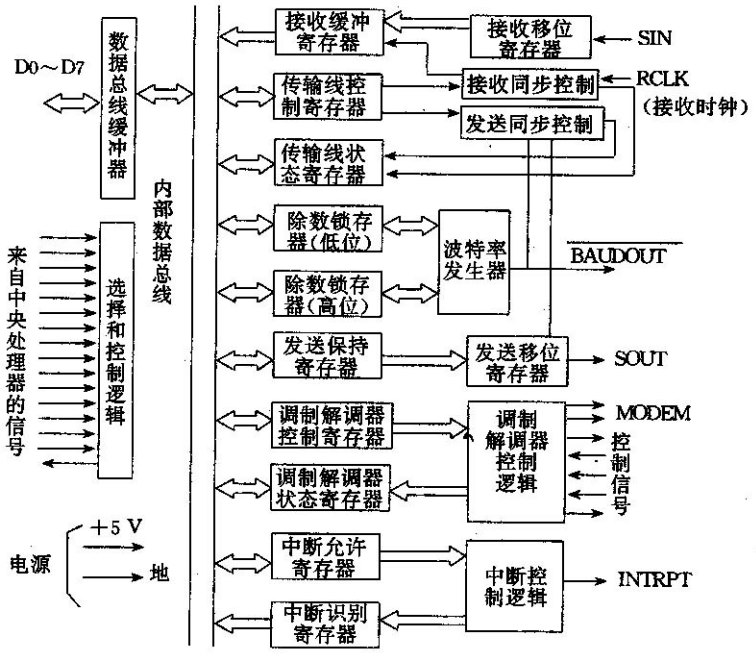


图 3.1-4 8250 的内部结构框图

8250 的引脚及功能框图如图 3.1-5 所示。

各部分的功能及引脚含义分别说明如下：

（1）数据输入/输出总线缓冲器

D0~D7，八条三态双向数据线，实现 8250 芯片与 CPU 之间的双向通信、包括数据、控

制字及状态信息的传输。

（2）选择和读/写控制逻辑

该逻辑包括如下引脚功能；

CS0，CS1，$\overline{CS2}$，芯片选择。当 CS0，CS1 为高，$\overline{CS2}$ 为低时，8250 芯片被选中。

$\overline{ADS}$，地址选通。当 $\overline{ADS}$ 为低时，则锁存片选号（CS0、CS1、$\overline{CS2}$）和寄存器选择信号（A0、A1、A2），即允许芯片和 CPU 进行数据传输。

A0、A1、A2，寄存器选择。芯片内部供 CPU 访问的 10 个寄存器可由这三个信号和线路控制寄存器最高位 DLAB 共同来选择。

DIATR、$\overline{DIATR}$，数据输入选通。当 DISTR 为高电平或 $\overline{DISTR}$ 为低电平且芯片被选中时，则允许 CPU 从选定的寄存器中读出数据或状态信息。

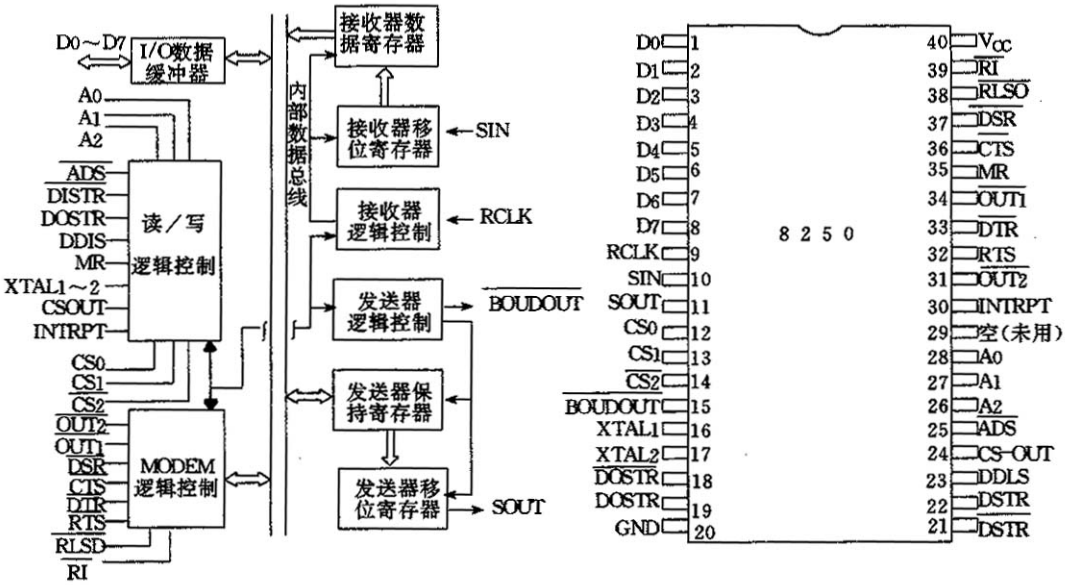


图 3.1-5 8250 的引脚及功能框图

DOSTR、$\overline{DOSTR}$，数据输出选通。当 DOTR 为高电平或 $\overline{DOTR}$ 为低电平且该片被选中时，则允许 CPU 将数据或控制命令写到选定的寄存器中。

DDIS，收发器禁止。该端输出为高电平时，禁止 CPU 对 8250 写操作。

MR，主复位。当 MR 为高平时，除接收缓冲器、发送保持寄存器、除数锁存器之外，其余寄存器和控制逻辑均被复位。有关复位的进一步细节见表 3.1-3。

表 3.1-3 异步通信的复位

寄存器/信号	复位控制	复位后状态
中断允许寄存器	MR	所有位为低电平
中断标识寄存器	MR	0 位为高其余位为低电平
通信线路控制寄存器	MR	所有位为低电平
MODEM 控制寄存器	MR	所有位为低电平
通信线路状态寄存器	MR	除 5，6 位外其余均为高
MODEM 状态寄存器	MR	0~3 位为高，4~7 位输入信号
SOUT	MR	高电平

INTRPT（线路状态错）	读线路状态寄存器/MR	低电平
INIRPT（发送保持器空）	读中断标识寄存器,写发送保持寄存器/MR	低电平
INIRPT（接收数据准备好）	读接收数据寄存器/MR	低电平
INTRPT（MODEM 状态改变）	读 MODEM 状态寄存器/MR	低电平
OUT2、RTS、DTR	MR	高电平
OUT1	MR	高电平

XTAL1、XTAL2，外部时钟输入/输出，由外部基准定时振荡器提供时钟。

CSOUT，片选输出。当 CSOUT 为高电平时，8250 已被 CS0、CSl、CS2 信号选中，数据传送可以开始。

INTRPT，中断。每当任一种中断类型变成有效且中断允许时，该输出端为高电平。

由上述各引脚的含义可知：读 / 写控制逻辑主要提供芯片与 CPU 之间的数据传输，并接受 CPU 的控制。

(3)调制解调器(MODEM)控制逻辑

该逻辑包括如下引脚功能：

$\overline{\text{DSR}}$，数据装置准备好，输入信号。当 DSR 为低电平时，表示 MODEM 已作好准备与 8250 进行数据传输。DSR 信号的状态可由调制解调器状态寄存器的第 5 位(DSR)检测出来。

$\overline{\text{CTS}}$，清除发送，输入信号。每当 MODEM 状态寄存器的 CTS 位改变状态时，如果允许 MODEM 状态中断，则产生一次中断。

$\overline{\text{DTR}}$，数据终端准备好，输出信号。当 DTR 为低电平时，通知 MODEM，8250 准备好通信，RTS，请求发送，输出信号。当 RTS 为低电平时，通知 MODEM，8250 准备好发送数据。

$\overline{\text{RLSD}}$，接收线路信号检测，输入信号。当 RLSD 为低电平时，表示 MODEM 已经检测出数据载波。每当 MODEM 状态寄存器的 RLSD 位改变状态时，如果允许 MODEM 状态中断，则产生一次中断。

RI，振铃指示，该端是 MODEM 控制功能的输入。若为低时，表示 MODEM 已收到一个电话响铃信号。若 MODEM 状态中断被允许，每当 MODEM 状态寄存器的 RI 位由高变低时产生一次中断。

$\overline{\text{OUT1}}$，输出 1，用户指定的 MODEM 控制功能的输出。通过对 MODEM 控制寄存器第 2 位编程置 1，OUTl 引脚就变成低电平。

$\overline{\text{OUT2}}$，输出 2，用户指定的 MODEM 控制功能的输出。通过对 MODEM 控制寄存器第 3 位编程置 1，OUT2 引脚就变成低电平。

综上所述，MODEM 控制逻辑实现 8250 与 MODEM 之间通信传输的控制。

(4)接收器逻辑

该逻辑包括接收器移位器和数据寄存器及相应的接收控制逻辑，其引脚功能如下：

RCLK，接收器时钟，输入接收波特率的 16 倍时钟信号。

SIN，串行输入，来自通信链路(如外设、MODEM、数据设备)的串行数据输入。

(5)发送器逻辑

该逻辑包括发送器保持寄存器和移位器及相应的发送控制逻辑，其引脚功能如下：

BOUDOUT，波特率输出，输出发送波特率的 16 倍时钟信号。

SOUT，串行输出，送到通信链路(如外设、MODEM、数据设备)的串行数据输出。

2．8250 的内部寄存器

8250 发送或接收数据之前，必须先进行初始化，即对它的一些内部寄存器写入控制字，以规定传输的波特率、数据格式、信号检测的要求、是否允许中断等等。通过读出或写入 8250 的内部寄存器来实现对串行通信的控制。

8250 内部有 10 个寄存器，如表 3．1—4 所示。这 10 个寄存器由 8250 的引脚 A2~A0 及通信线路控制寄存器的最高位(即除数锁存器访问位 DLAB)共同来选择。

表 3.1-4 8250 内部寄存器的选择

DLAB	A2	A1	A0	寄存器
0	0	0	0	接收缓冲器（读），发送保持寄存器（写）
0	0	0	1	中断允许
×	0	1	0	中断识别（只读）
×	0	1	1	通信线路控制（LCR）
×	1	0	0	调制解调器控制
×	1	0	1	通信线路状态
×	1	1	0	调制解调器状态
×	1	1	1	无用
1	0	0	0	除数锁存器（低有效位）
1	0	0	1	除数锁存器（高有效位）

8250 的 10 个内部寄存器端口地址及读写情况见表 3.1-5。第一异步通信适配器（COM1）内部寄存器端口地址为 3F8H~3FEH，如果是第二通信口（COM2），则 3F8H~3FEH 的地址应分别改为 2FBH~2FEH。

表 3.1-5 UART 内部寄存器地址分配

端口地址（COM1）	端口地址（COM2）	输入/输出	寻址条件	寄存器名称
3F8H	2F8H	输出	DLAB=0	发送保持寄存器（THR）
3F8H	2F8H	输入	DLAB=0	接收数据寄存器（RBR）
3F8H	2F8H	输出	DLAB=1	波特率因子（除数）低字节 LSB（DR）
3F9H	2F9H	输出	DLAB=1	波特率因子（除数）高字节 MSB（DR）
3F9H	2F9H	输出	DLAB=0	中断允许寄存器（IER）
3FAH	2FAH	输入	-	中断标识寄存器（IIR）
3FBH	2FBH	输出	-	线路控制寄存器（LCR）
3FCH	2FCH	输出	-	MODEM 控制寄存器（MCR）
3FDH	2FDH	输入	-	线路状态寄存器（LSR）
3FEH	2FEH	输入	-	MODEM 状态寄存器（MSR）

从表中可看到，发送保持寄存器、接收数据寄存器和波特率因子低字节寄存器合用一个口地址；中断允许寄存器和波特率因子高字节寄存器合用一个口地址；线路控制寄存器中的位 7（表中用 DLAB 表示）决定复用的口地址当前对应哪一个寄存器；发送保持寄存器和接收数据寄存器总是合用一个口地址，当向该口地址写时，就对应发送保持寄存器，当从该口地址读时，就对应接收数据寄存器。

用户可编程使用的 10 个内部寄存器功能和规定介绍如下：

（1）通信线路控制寄存器（LCR）

通信线路控制寄存器用于控制通信数据格式，其 I/O 口地址为 3FBH，寄存器各位含义如图 3.1-6 所示。

（2）通信线路状态寄存器

该寄存器向处理器提供有关数据传送的状态信息，I/O 端口地址为 3FDH，各位含义如图 3.1-7 所示。

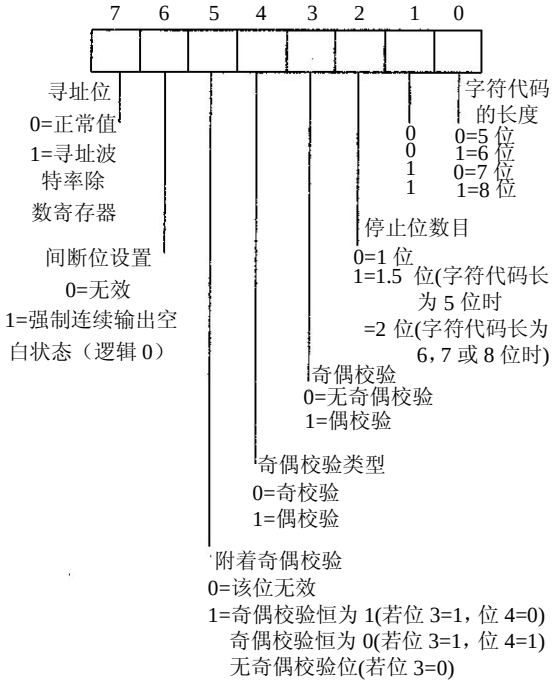


图 3.1-6 通信线控制寄存器

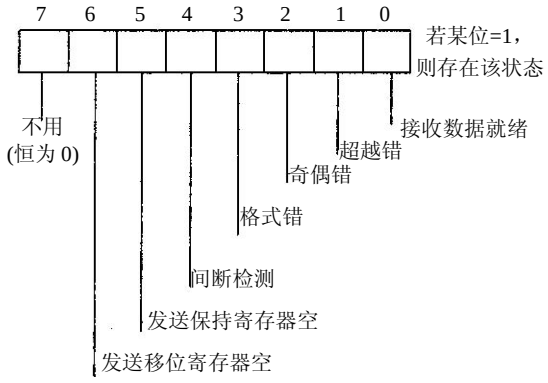


图 3.1-7 通信线状态寄存器

其中，位 0~位 3 为出错显示位，当位 4 为 1 时，表示接收到的数据在大于一个完整的数据字间内均为空（0 状态），实际表示线路间断。当处理器读取该寄存器时，该位复位。以上 5 位一旦某位为 1，就发出接收线路出错中断。

（3）发送保持寄存器

该寄存器用于保存待发送的一个字节的数据，该字节数据经发送移位寄存器串行发出。

（4）接收数据缓冲器寄存器

该寄存器保存从接收移位寄存器移入的字节数据。

（5）除数锁存器

除数锁存器的值必须在8250初始化时设置，即把通信线路控制寄存器的最高位（DLAB）置 1，然后分两次把除数锁存器的高 8 位和低 8 位分别写入端口地址 3F8H 和 3F9H。

8250 使用频率为 1.8432MHz 的基准时钟输入信号，所以需用分频的方法产生所需的波特率。8250 传送或接收串行数据时，使用的时钟信号的频率是数据传送波特率的 16 倍，锁存器的除数值可以由以下公式推算：

除数=1843200÷（16×波特率）

表 3.1-6 列出了获得 15 种波特率所需设置的除数值。

表 3.1-6 波特率除数锁存器的值与波特率的对应关系

波特率/$b.s^{-1}$	波特率除数锁存器的值	
	高 8 位（H）	低 8 位（M）

50	09	00
75	06	00
110	04	17
134.5	03	59
150	03	00
300	01	80
600	00	C0
1200	00	60
1800	00	40
2000	00	3A
2400	00	3A
3600	00	20
4800	00	18
7200	00	10
9600	00	0C

（6）中断允许寄存器

中断允许寄存器的端口地址为 3F9H，其各位含义如图 3.1-8 所示。8250 芯片可以处理四种类型的中断，按优先次序排列为：

①接收线路出错；

②接收数据就绪；

③发送保持寄存器已空；

④MODEM 中断。

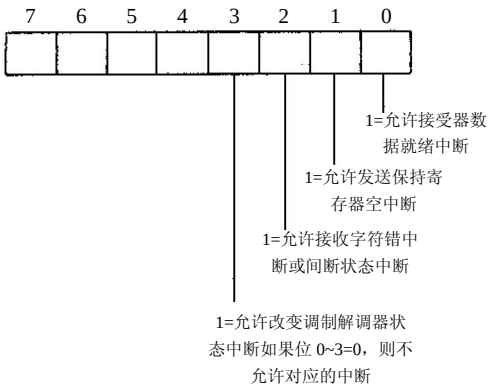
中断允许寄存器的低 4 位分别对应上述 4 种中断，当对应位为 1 时，则允许以应中断信号输入。如果位 0~位 3 全为 0，则表示禁止 8250 中断，从而中断标识寄存器和中断请求信号均被禁止输出。

（7）中断识别寄存器

由于 8250 仅能向外发出一个总的中断请求信号，程序为了识别是哪一个中断源引起的中断，以便转入相应的中断处理程序，必须从该寄存器（端口地址为 3FAH）读出其内容，各位含义如图 3.1-9 所示。

（8）MODEM 控制寄存器

异步通信适配在近距离内通信可以直接和具有 RS-232C 接口的设备相连，以实现两个设备之间的异步串行通信。当通信距离较远时，由于电话线的频带很窄（一般在 300~3000Hz 之间），所以必须对计算机 RS-232C 或 20mA 电流环接口输出的数字信号用专用调制解调器进行调制。调制后的信号经过电话线进行传输，在接收端再通过 MODEM 调解，检出的同 RC-232C 送到另一个计算机或终端，以实现远程频串行通信。由于 MODEM 既可调制又可解调，故可实现双向通信。利用 MODEM 进行通信，处理器必须对 MODEM 控制寄存器和 MODEM 状态寄存器控制并读取它的状态。



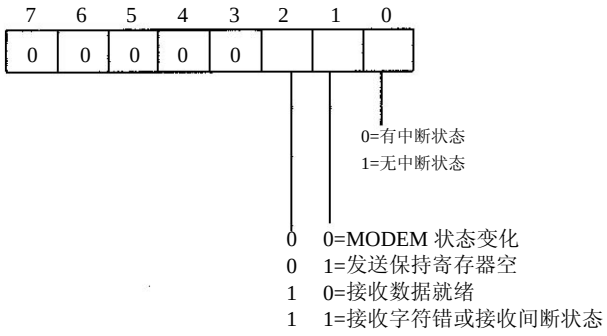
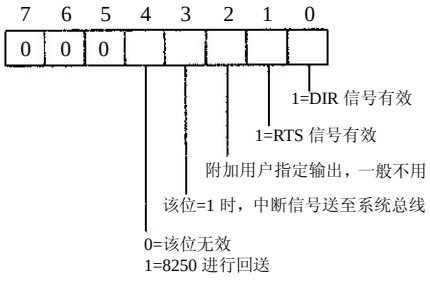


图 3.1-8 中断允许寄存器　　图 3.1-9 中断识别寄存器

MODEM 控制寄存器的 I/O 端口地址为 3FCH，其控制字格式如图 3.1-10 所示。

（9）MODEM 状态寄存器

该寄存器的端口地址为 3FEH，各位含义如图 3.1-11 所示，低 4 位是控制输入信号发生变化的状态标志，初值应置 0，如果读入 MODEM 状态字节中出现置 1 的位，则说明对应输入信号发生变化；高 4 位保存 MODEM 控制信号。



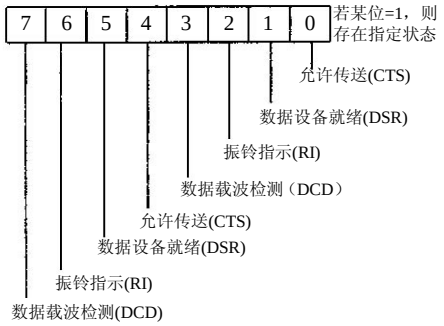


图 3.1-10 MODEM 控制寄存器　　图 3.1-11 MODEM 状态寄存器

3、通信原理

对 8250 异步通信控制器编程实现 PC 机通信的发送器输出端和接收器输入端的一种数据式，如图 3.1-12 所示。

启动位	D0	D1	D2	D3	D4	D5	D6	D7	校验	停止位

图 3.1-12 8250 接收和发送的数据格式

8250 通信线路控制寄存器的位 1、位 0 设置为“11”，设定传送的每帧字符代码长度为 8 位；位 2 为“0”，设定停止位数目为 1 位；位 3 设置决定有无奇偶校验位；位 4 决定奇偶校验位的类型，是奇校验还是偶校验；通过除数锁存器来设定波特率；对 8250 的 10 个寄存器的操作，决定了 PC 机的通信形式。

8051 单片机内含有一个全双工异步串行通信口，数据格式如图 3.1-13 所示。每个字符用一个起始位（低电平）表示字符的开始，接着是 8 位或 9 位数据位，最后以一个停止位（高电平）表示字符的结束，构成一帧信息。

8051 串行口具有两个物理上独立的发送/接收缓存器；另外，特殊功能寄存器 SCON 和 PCON 控制串行口的工作方式以及波特率，定时器 T1 作为波特率发生器。

起始位	D0	D1	D2	D3	D4	D5	D6	D7	第 9 位数据	停止位

图 3.1-13 8051 的数据传输格式

3.1.2 对 8250 的编程

仅就 8250 而言，异步串行通信编程步骤如下：

第一步　设定通信的规程，如波特率、奇偶校验方式、数据格式、数据字节长度等；

第二步　读取通信线路（或 MODEM）的状态，判断是否可进行通信；

第三步　送出（或接收）一个数据字节；

第四步　重复第二步和第三步直到通信完毕。

当允许中断时，CPU 送出（或收到）一个字节之后，并不需要不断查询控制器的状态，而可转向执行其他任务。当有中断信号 INT4 发生并响应后，再按上述第二步和第三步处理即可。

应用 8250 进行串行通信时，首先要对其初始化，即设置波特率、通信采用的数据格式、是否使用中断、是否自测试操作等。初始化后，则可采取程序查询方式或中断方式进行通信。

8250 的初始化一般分成三步：

1、设置波特率（假设为 1200）

```
MOV   AL，1000000B    ；置 DLAB=1
MOV   DX，3FBH        ；写入通信线控制寄存器
OUT   DX，AL
MOV   AL，60H         ；置产生 1200 波特率除数低位
MOV   DX，3F8H
OUT   DX，AL          ；写入除数锁存器低位
MOV   AL，00H         ；置产生 1200 波特率除数高位
MOV   DX，3F9H
OUT   DX，AL          ；写入除数锁存器高位
```

2、设置通信数据格式

假设 7 个数据位，1 个停止位，偶校验，编程如下：

```
MOV   AL，00011010B       ；设置数据格式
MOV   DX，3FBH            ；写入通信线控制寄存器
OUT   DX，AL
```

3、设置操作方式

PC 机异步通信适配器中的 8250 中断输出（INTRPT）外接成受引脚 $\overline{OUT2}$ 控制输出的三态门控制，见图 3.1-3。只有当 $\overline{OUT2}$ 为低电平，并有 INTRPT 产生时，中断信号才可通过此三态门。因此，只要控制 $\overline{OUT2}$ 输出，即可控制是否允许中断信号通过。对 MODEM 控制寄存器写入所要求的控制字，置位 3 为 1，便可使 $\overline{OUT2}$ 为低电平，三态门变成常通状态，可以在中断方式下工作（中断是否产生，还受中断允许寄存器的控制）。编程示例如下：

```
；不允许中断输出：
MOV   AL，03H             ；使 OUT2 为高， DTR, RTS 有效
MOV   DX，3FCH
```

```
OUT   DX，AL
；允许中断输出：
MOV   DX，3FCH
MOV   AL，0BH              ；使OUT2为低，DTR,RTS有效
OUT   DX，AL
；自测试工作方式：
MOV   AL，13H              ；自测试下若允许中断则应为1BH
MOV   DX，3FCH
OUT   DX，AL
```

4、设置中断允许寄存器

假设禁止中断，编程如下：

```
MOV   AL，00H              ；禁止所有中断的控制字
MOV   DX，3F9H
OUT   DX，AL               ；写入中断允许寄存器
```

若允许中断或仅允许四种类型中的某几类中断，则可写入相应的控制字到中断允许寄存器中去。

3.1.3 串行口硬件中断及处理

UART 不但支持查询方式发送和接收，也支持中断方式发送和接收，现从编程的角度简单介绍有关串行口硬件中断及其处理方面的内容。

在绝大多数 PC 系列及其兼容机上，串行口 1 和串行口 2 作为两个外部中断源连接在中断控制器 8259 上。串行口 1 作为外部中断级 4，而串行口 2 作为外部中断级 3。图 3.1-14 给出了串行口 1 作为外部中断级 4 的连接示意图。

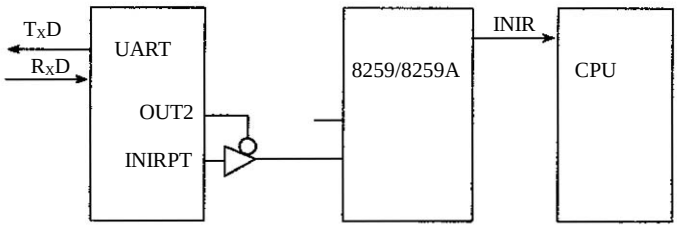


图 3.1-14　串行口 1 作为外部中断级 4 的连接示意图

UART 支持 4 种类型的中断，程序可通过设置 UART 的中断允许寄存器达到允许或禁止某些类型中断的目的。例如，设置 UART 中断允许寄存器的值为 0，UART 就不会发生中断，从而工作于查询方式。设置 UART 中断允许寄存器的值为 01H，当接收器数据就绪时，UART 的 INTRPT 信号将有效。但 UART 的 INTRPT 信号能否传到系统的中断控制器(8259／8259A)，还取决于 UART 的输出信号 OUT2 是否有效。所以，为了保证中断信号能传到系统的中断控制寄存器，程序事先还要通过设置 UART 的 MODEM 控制寄存器来使 OUT2 输出有效。

系统的中断控制器负责管理多级外部中断。在中断控制器中有一个中断屏蔽寄存器，其口地址为 21H。该寄存器的各位分别对应 IRQ0～rRQ7 的请求信号，当某位为 1，相应的中断请求被屏蔽，否则，相应的中断请求不被屏蔽。仅当对应于串行口 1 的中断屏蔽寄存器的位 4 为。时，由 UART 发出的中断请求才可能通过系统的中断控制器传到 CPU。同样，对应串行口 2 的中断屏蔽寄存器中的位是位 3，对串行口 2 也有类似的作用。为了使由串行口发出的中断请求能传到 CPU，程序要事先根据要求设置系统中断控制器中的中断屏蔽寄存器。

CPU 在接收到外部中断请求信号时，仍可能不响应中断。为了使 CPU 真正响应由串行口发出的中断请求，CPU 还必须处于开中断状态，即 CPU 标志寄存器中的 I 位必须为 1。指令“STI"和"CLI”是专门用于控制该中断标志的。

外部中断级 3 和级 4 分别对应 0BH 和 0CH 号中断向量，当 CPU 响应由串行口 COM]发出的中断请求时，将根据 0CH 号中断向量执行对应的中断处理程序，当 CPU 响应由串行口 COM2 发出的中断请求时，将根据 0BH 号中断向量执行对应的中断处理程序。为此，程序要事先根据需要正确设置好有关的中断向量。

综上所述，为采用中断方式接收和发送，有关应用程序在初始化时要完成如下工作，不过，它们只是中断的必要条件：

(1)根据需要设置 UART 的中断允许寄存器；

(2)设置 UART 的 MODEM 控制寄存器，使 OUT2 输出有效；

(3)设置系统中断控制器中的中断屏蔽寄存器，保证不再屏蔽有关中断；

(4)设置有关的中断向量；

(5)保证能及时使 CPU 处于开中断状态。

CPU 在响应中断后，将执行对应的中断处理程序。作为处理串行口中断请求的中断处理程序也应像别的中断处理程序一样，必须遵守有关的基本原则。串行口中断处理程序应遵守的基本原则如下：

(1)尽可能及时开中断；

(2)保护要使用到的寄存器；

(3)尽快结束中断处理；

(4)向系统中断控制器发出中断结束通知；

(5)正确恢复受保护的寄存器。

3.3 两台 PC 机间 DOS 环境下 C 语言通信程序

有三种方式访问 PC 机或兼容机的串口：通过 DOS，通过 BIOS，或绕过 DOS 和 BIOS 直接对硬件控制。通过 DOS 访问串行口通常不是一个好办法，因为它对端口的状态不提供反馈信息，只是对端口盲目的读写，这样，DOS 中断将不被使用。直接对硬件控制对串口不是必须的，因为这样将增加编程的工作量和复杂程度。而采用 BIOS 中断调用编写串行通信程序是很方便的。

四个 BIOS 功能支持对串行口的访问。这些功能是通过中断 0x14 实现的。

3.3.1 C 语言关于 BIOS 的调用

这里简单介绍一下 C 语言关于 BIOS 与串行口有关的中断调用函数。BIOS 程序是独立于任何操作系统的，共有十多个中断服务（因机型不同而不同）。它的服务是最底层的服务，C 语言的函数可以方便的调用这些服务功能，这是 C 语言接近硬件层的特色。

所有与 DOS 有关的函数原型定义均包含在 dos.h 文件中，所以，在头文件中应包含 dos.h。同时，在 dos.h 中也包含了一些相应结构及联合类型的定义，其中有下面要用到的 REGS 类型。

1、RDGS 定义格式如下：

```
Union REGS
{
    struct WORDREGSx;
    struct BYTEREGSh;
}
```

其中结构 WORDREGS 和 BYTEREGS 定义如下：

```
struct WORDREGS
{
    unsigned int ax, bx,cx,dx,si,di,cflag,fflags;
};
struct BYTEREGS
{
    unsigned char al,ah,bl,bh,cl,ch,dl,dh;
};
```

2、下面介绍一下 C 语言的中断调用函数 int86( )。

int86( )函数的格式如下：

```
int86(int int_ num,union REGS*in_negs,union RES*ou_ regs)
```

其中各个参数含义如下：

int_num:指定的中断号。

*in_negs:调用中断前 CPU 的 ax,bx,cx,dx,si,di,cflag,fflags 寄存器的入口值。

*out_regs：中断返回后 CPU 的 ax,bx,cx,dx,si,di,cflag,fflags 寄存器的出口值。

使用 int86()函数可以调用任何一个由 int_num 指定的软中断。执行时，首先把 in_negs 中的内容复制到 CPU 的各对应寄存器中，然后产生相应的中断，中断返回后再将 CPU 中寄存器的内容复制到 out-regs 中。

3.3.2 端口初始化

BIOS 串行口管理程序作为 0x14 中断处理程序，并且其返回结果不影响标志寄存器各标志，所以可以利用一般 C 编译都提供的 int86( )函数来调用 BIOS 串行口管理程序。为了利用 int86()函数需要定义一个类型的 union 的 REGS 变量。这此函数均带有一个用于指定串行口的参数。

再标准的个人微机上最多可达 7 个串行口可供使用。0x14 中断的 0 功能用来初始化一个串行口。如同其他 BIOS 中断，AH 寄存器保存功能 0，DX 置所要初始化的串行口端代码，第一个串行口是 0，第二个是 1，依次类推，AL 寄存器保存被编入 1 个字节的初始参数。下面名为 init_port()的函数对系统中任何串行口的值初始化：

```
void port_init(int port,unsigned char code)
{
union REGSr;
r.x.dx=prot;     /*定义端口号*/
r.h.ah=0;       /*初始化端口功能*/
r.h.al=code; /*初始参数*/
int86(0x14,&r,&r);
}
```

此函数依靠 int86()函数对串行口初始化。

3.3.3 传输字符

BIOS 中断 0x14，功能 1 通过在 DX 中指定的串行口（第一个或第二个串行口）传输一个字符。把要发送字符的 ASCII 码保存在 AL 中。传输状态在 AH 寄存器中得到返回。如果成功，则 AL 中的数据被发送出去。下面函数 sport()通过指定的串行口传输一个字节：

```
/*从串行口发送一个字符*/
void sport(int port,char c)
{
```

```
地址 union REGS r;
r.x.dx=port;            /*端口号*/
r.h.al=c;               /*要发送的字符*/
r.h.ah=1;               /*要送字符功能*/
int86(0x14,&r,&r);/*中断调用*/
if(r.h.ah &128)     /*如果发送不成功，则打印错误信息并退出*/
{
printf(“send error detected in serial port”;)
exit(1);
}
}
```

如果 AH 的 7 位被中断返回置位，就发生了传输错误，所以打印错误信息并退出。

3.3.4 端口状态检测

BIOS 中断 0x14，功能 3 可用于检测端口状态。欲检测的端口在 DX 寄存器中指定。从中断返回，AH 和 AL 中将保存有端口的状态。可以通过检测“数据就绪（AH 的 0 位）”确定何时一个字节的数据已被端口接收，并为读取作准备。下面的函数 check_stat( ) 返回端口状态信息：

```
/*检测串行端口状态*/
check-stat(int port)            /*检测端口程序*/
{
union REGSr;
r.x,dx=port,            /*定义端口*/
r.h.ah=3，              /*定义检测端口功能*/
int86(0x14,&r,&r) /*中断调用*/
return r.x.ax;        /*返回 AX 寄存器中的内容*/
}
```

3.3.5 字符接收

BIOS 中断 0x14，功能 2 从串行口读取一个字符。在 DX 寄存器中指定端口，从中断返回时，所读取的字符在寄存器 AL 中，中断功能在 AH 中，中断返回之后，AH 的位 7 用于指明成功与否。下面的 rpotr()函数为从指定端口读取一个字节：

```
/*从指定端口读取一个字符*/
rport(int port)
{
   union REGSr;
   /*等待一个字符，若“数据就绪”位为 0 则等待，直到此位为 1，或按任意键退出*/
   while(! (check_stat(port)&256))
   {
    if(kbhit())
   getch();
   exit(1);
   }
r.x,dx=port;            /*定义端口号*/
r.h.ah=2;               /*定义读取字符功能*/
int86(0x14,&r,&r);   /*中断调用*/
```

```
if(r.h.ah&128)        /*若 AH 位 7 被置位，则打印出错信息*/
   printf("read error detected in serial port");
return r.h.al;        /*返回 AL 中的读取字符*/
}
```

在一个字符被串行口接收并返回之前，读取端口中断将等待。若有连接等错误导致系统锁定，可通过按键盘退出接收程序。若接收成功，则返回 AL 中的接收字符。

第四章　　局域网测控系统程序及其解读

4．1 系统人机界面

主机运行画面

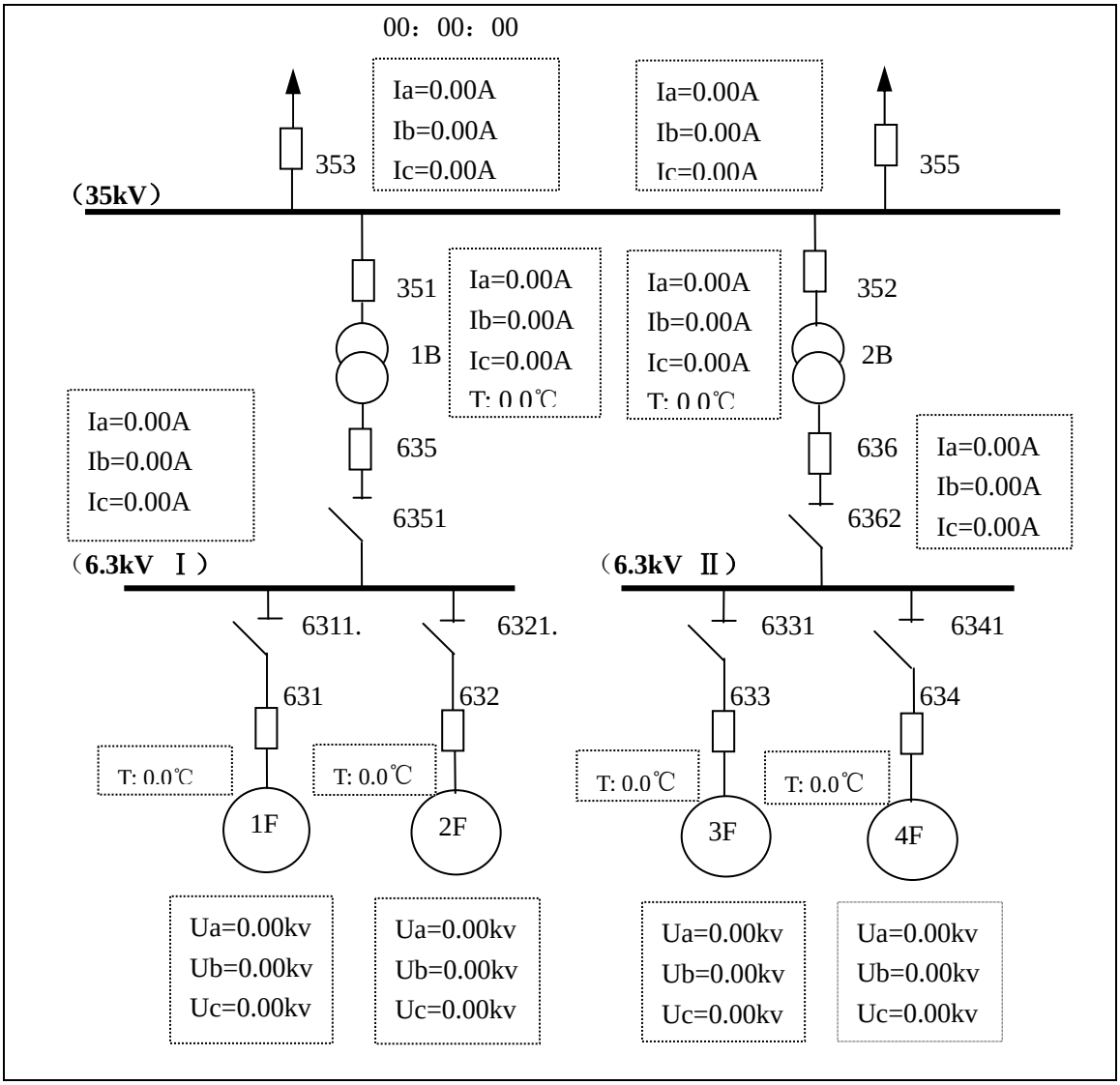


图 4—1 小型电站主站电气主接线图及实时参数显示

上图是局域网测控系统的主站（主机）运行时的显示画面。本实验系统是小型电站（发电厂）的局域网式巡回监测系统，即对电站的运行状态进行不间断的监测，对电站运行的异

常状态及时处理、报警、记录，对电站各类运行参数作实时记录并制成报表存档。主站的显示参数由从站通过 RS485 串行总线发送。

从机运行画面

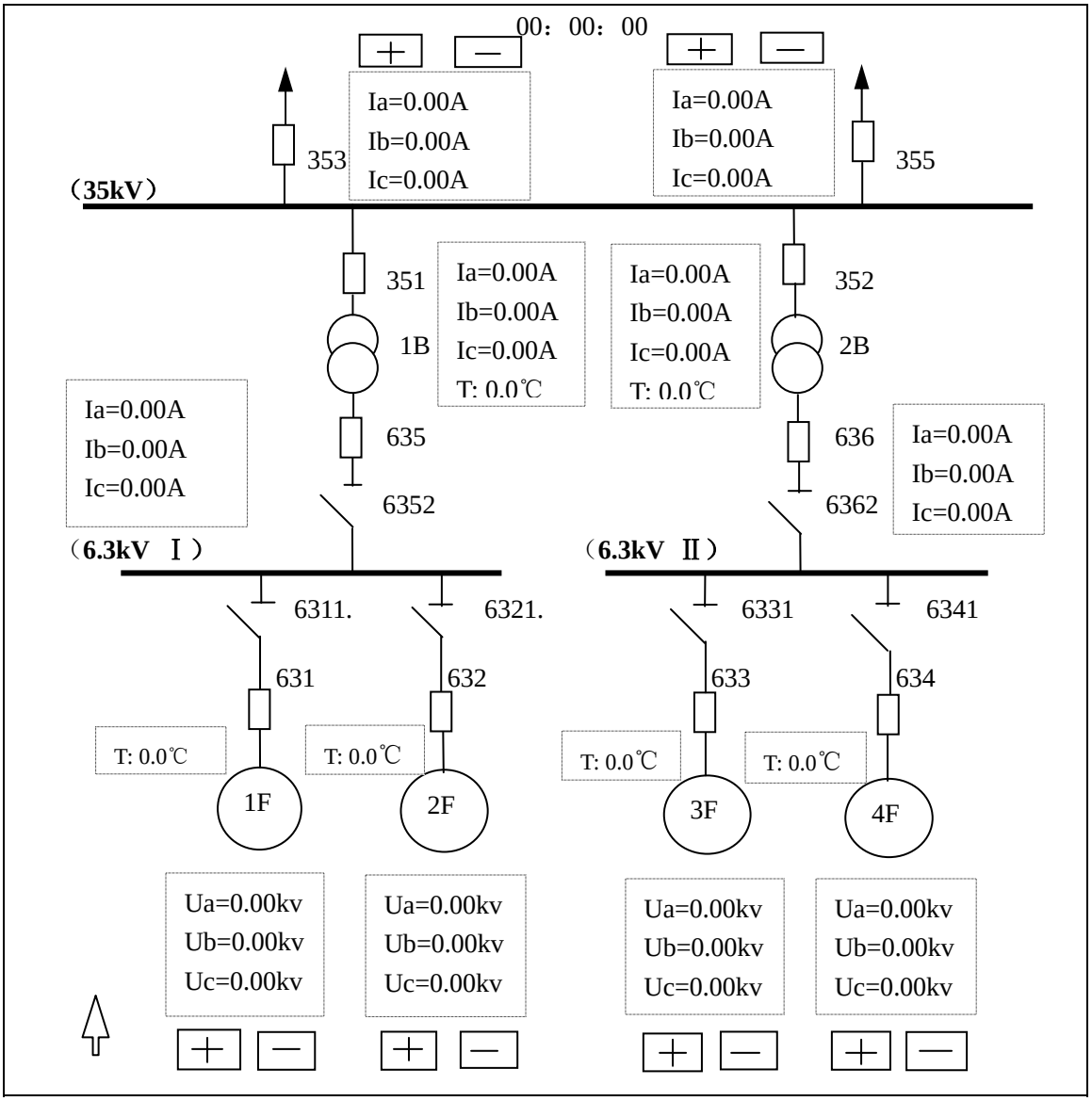


图 4—2 小型电站子站电气主接线图及实时参数显示

上图是局域网测控系统的子站（从机）运行时的显示画面。与主站画面不同的是增加了鼠标控制的箭头形状的光标（图左下角）和带“＋”“－”符号的“按键”。当光标移动到带“＋” 符号的“按键”上，按下鼠标左键你能看到带“＋” 符号的“按键”所属的参数增加，按“－”符号的“按键”则减少。当光标移动到开关符号和断路器符号的位置，按下鼠标左键你能看到开关符号和断路器符号发生从接通到断开或从断开到接通的变化。这样一来，你可以操纵鼠标设计一个小型电站运行状态。

图 1-3 是两台 IPC 通过 RS-232/485 实现局域网测控系统电气连结方式。这是专为学习设计局域网测控系统准备的实验装置，在这个装置上可以完成本实验要求。

4．2 程序从属关系

4．2．1 两台 IPC 通过 RS-232/485 实现局域网测控系统仿真的程序从属关系

本系统编程使用 BORLAND C＋＋ 3 • 1 语言。

其程序结构关系如下：

图 4。2。1-1 主站程序关系，文件夹名：sheji_c1

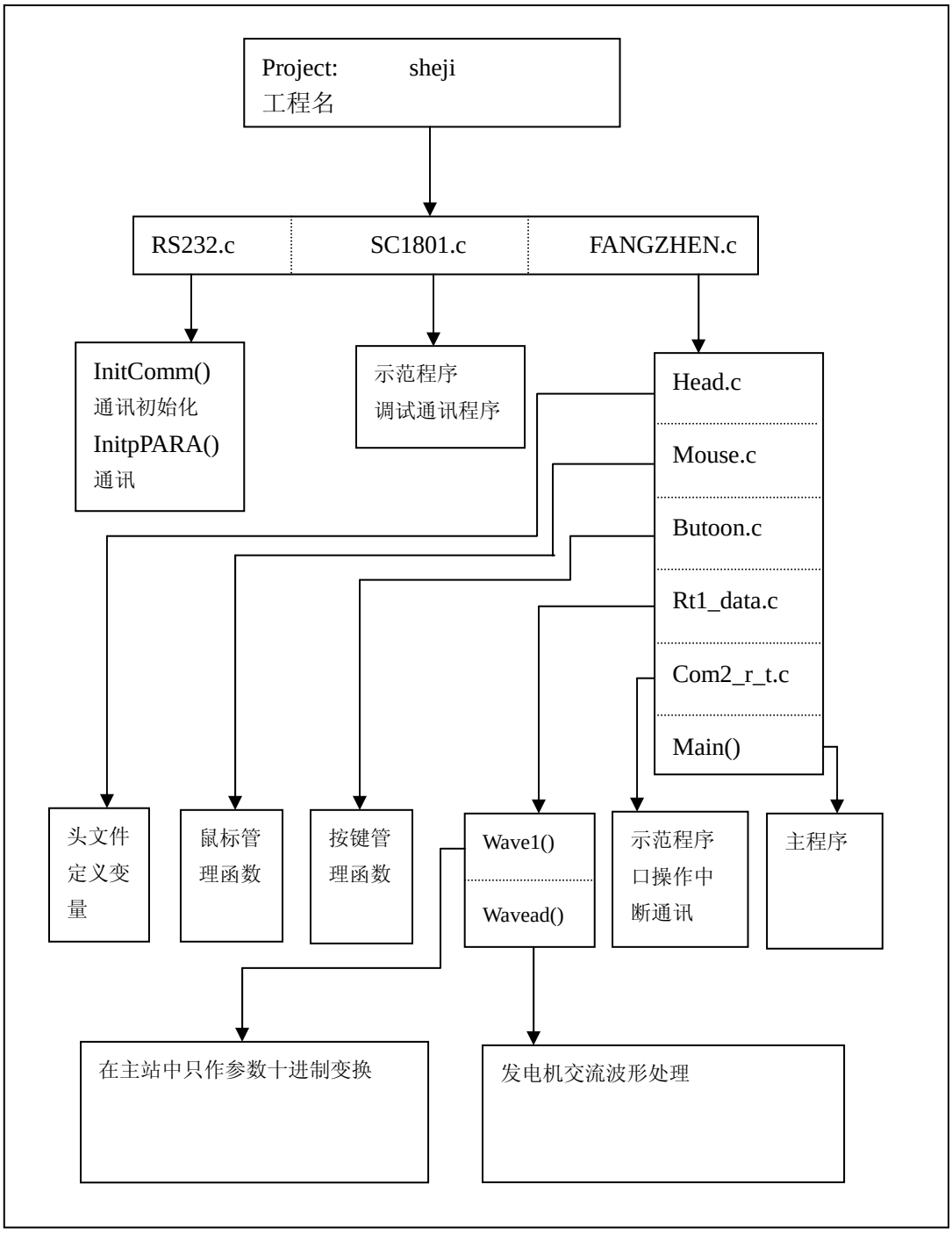


图 4。2。1-1 主站程序关系

图 4。2。1-2 从站程序关系，文件夹名：sheji_c2

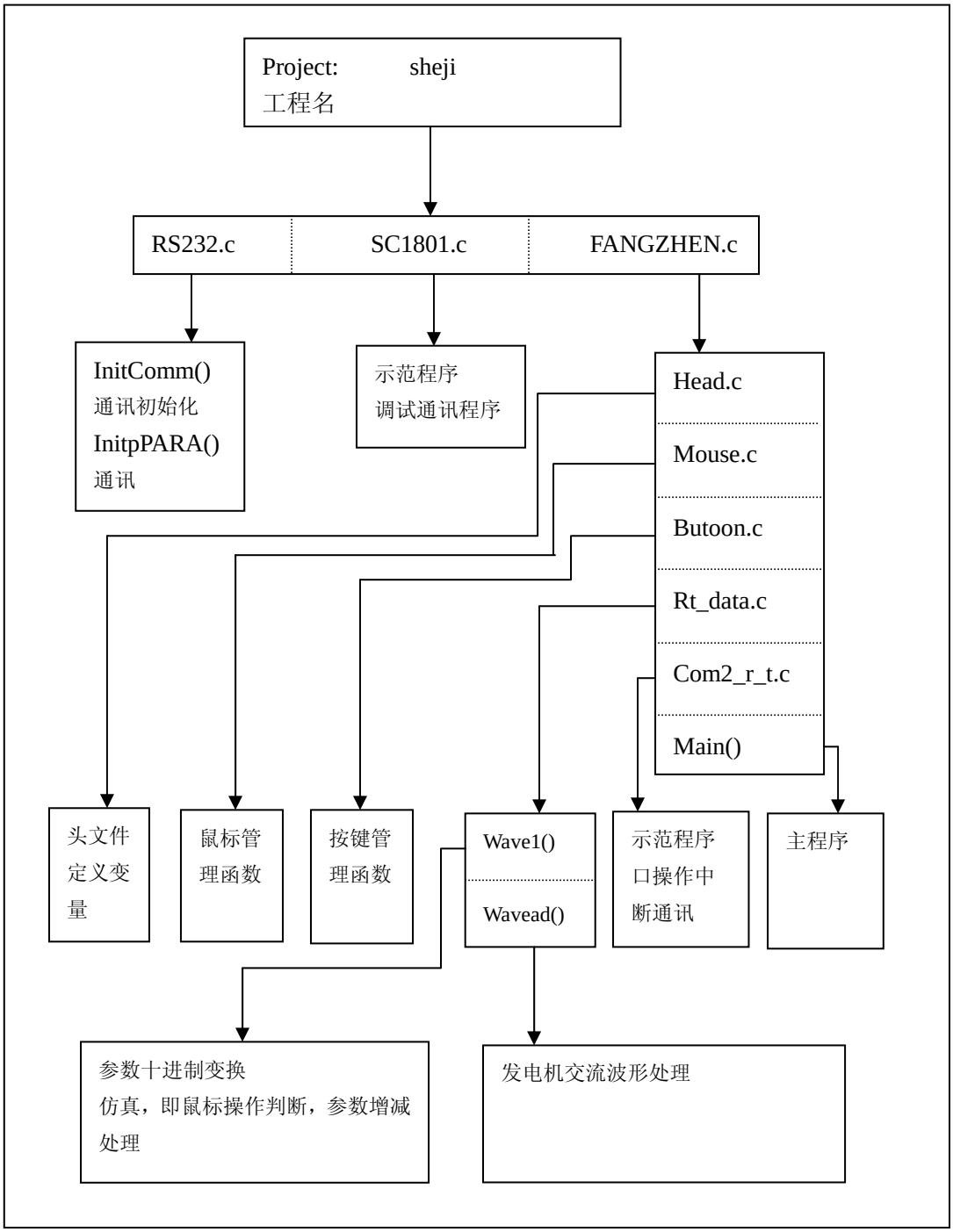


图 4。2。1-2 从站程序关系

4．2．2 由四台 IPC 构成的两级 DCS 实验系统的程序从属关系

图 1-2 是由四台 IPC 构成的两级 DCS 实验系统,其功能同前面介绍的仿真系统一样，也是小型电站（发电厂）的局域网式巡回监测系统。在仿真系统程序结构的基础上，加入了主站呼叫从站及相应的画面。熟悉仿真系统程序结构后，能够顺利读懂这种系统的程序。

图 4。2。1-3 由四台 IPC 构成的两级 DCS 实验系统主站程序关系，文件夹名：mast741

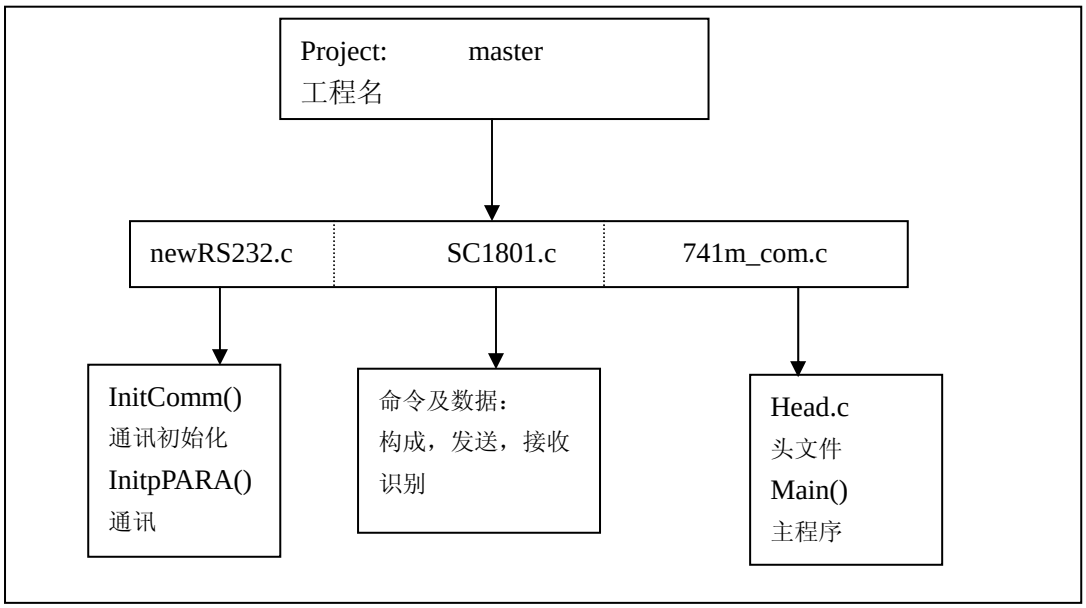
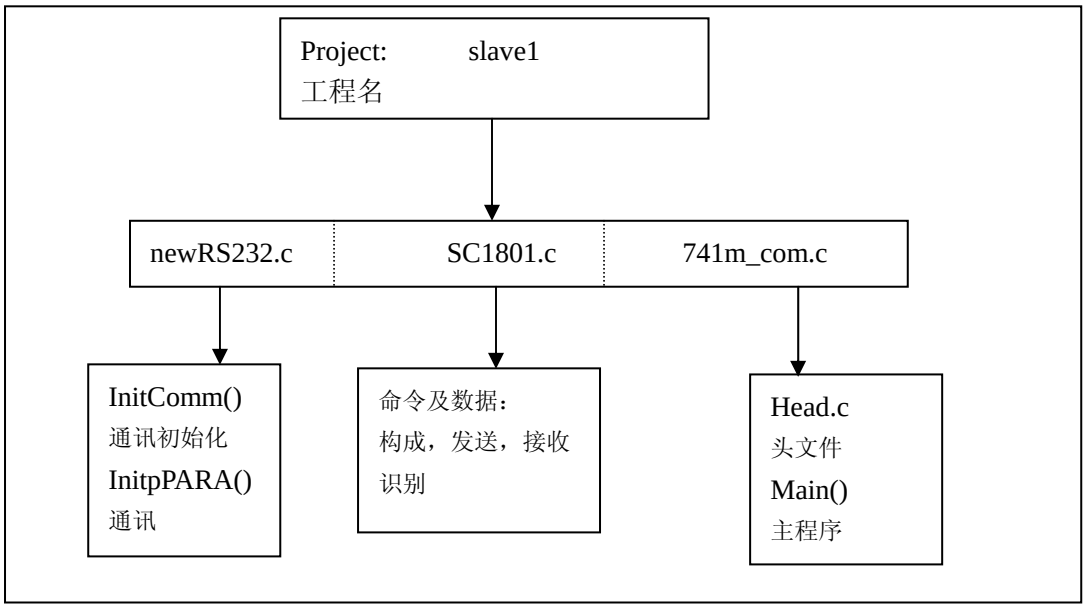


图 4。2。1-4 由四台 IPC 构成的两级 DCS 实验系统从站程序关系，文件夹名：slav1741

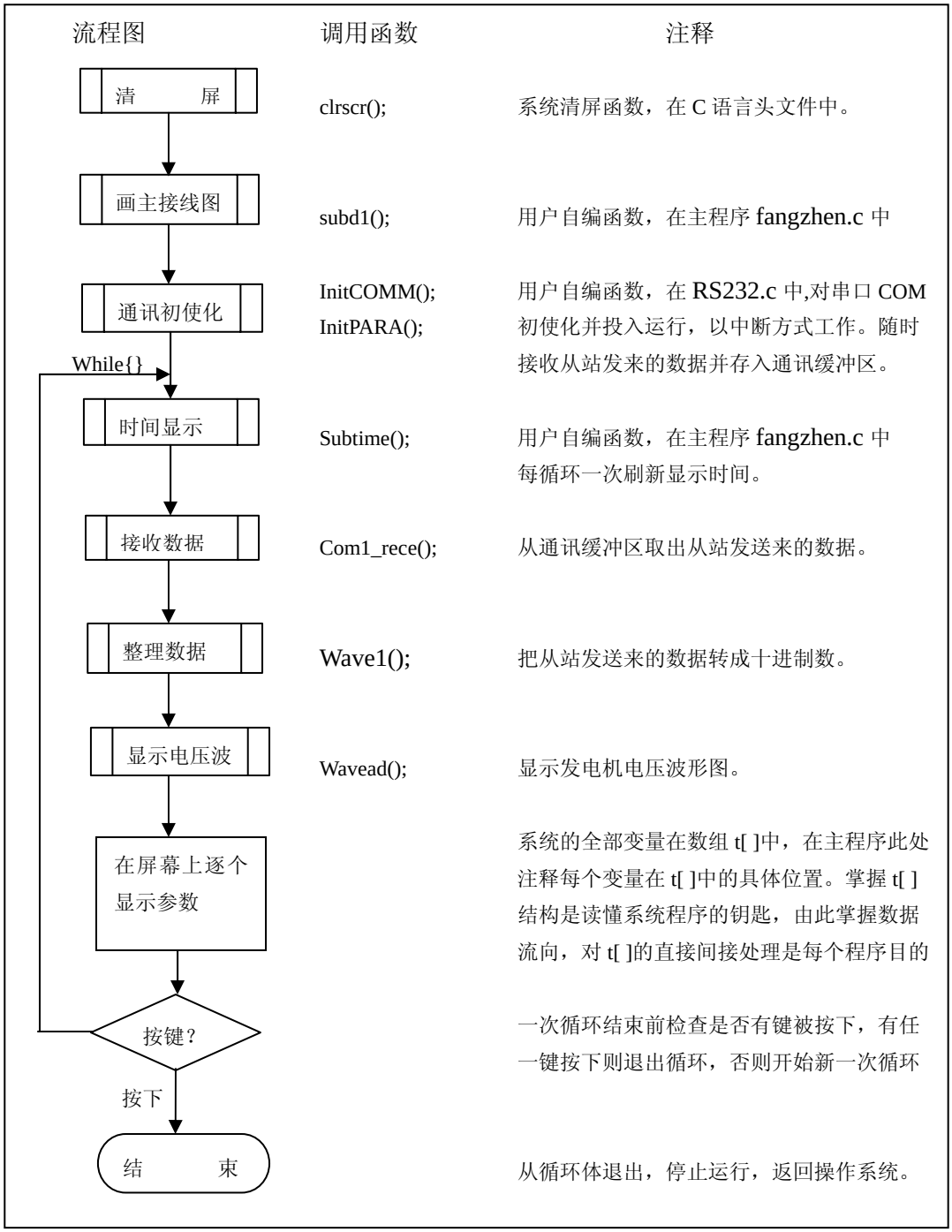


从站文件夹名分别是：slav1741,slav2741,slav3741.
从站工程名分别是：slave1,slave2,slave3.

4．3 主程序

4．3．1 主站主程序

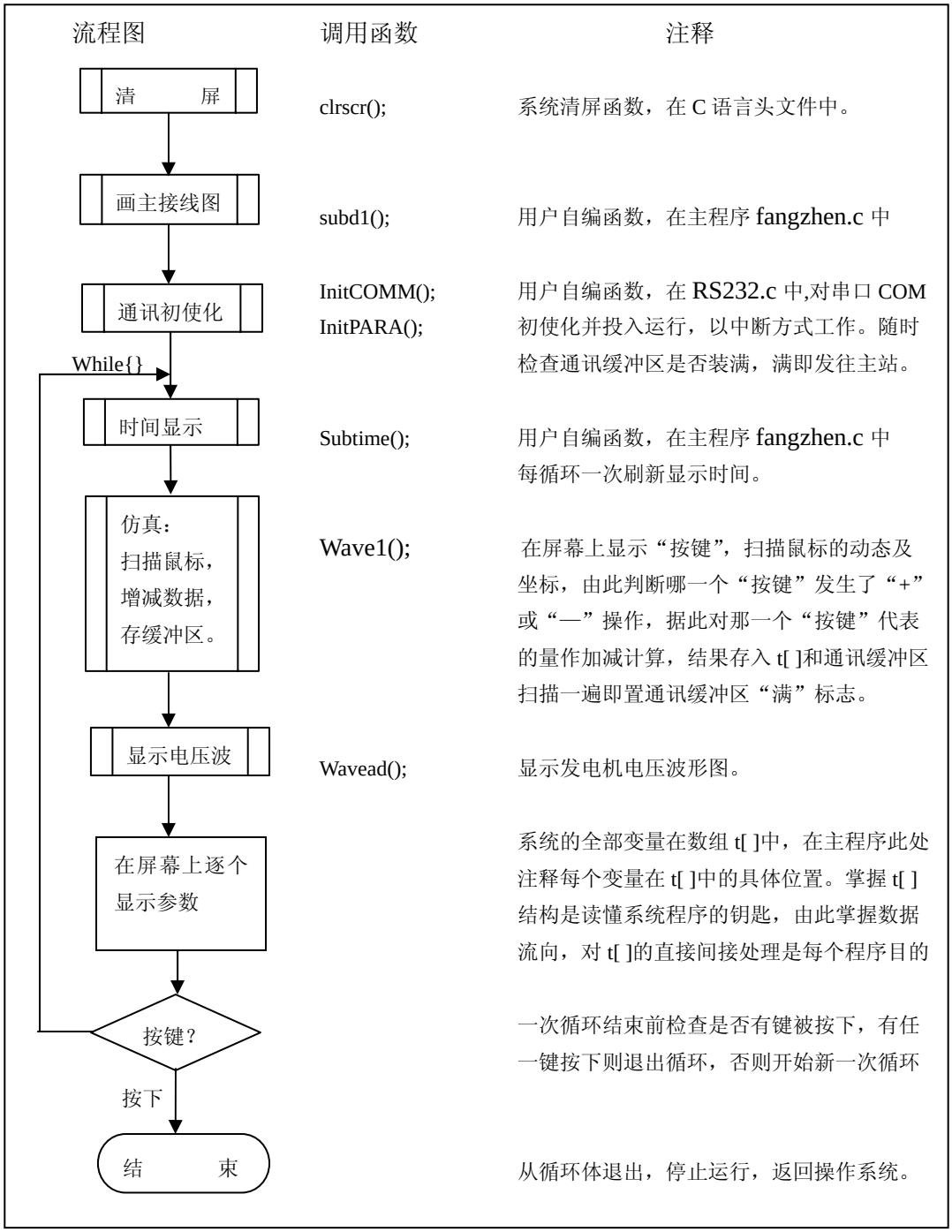
在前面介绍的 sheji_c1 文件夹中的有仿真主站主程序 fangzhen.c,其文件清单几乎是逐句汉字注释，结合下面的程序结构分析图，可以清晰地了解程序全部细节。



4．3．2 从站主程序

在前面介绍的 sheji_c2 文件夹中的有仿真从站主程序 fangzhen.c,其文件清单几乎是逐句汉字注释，结合下面的程序结构分析图，可以清晰地了解程序全部细节。

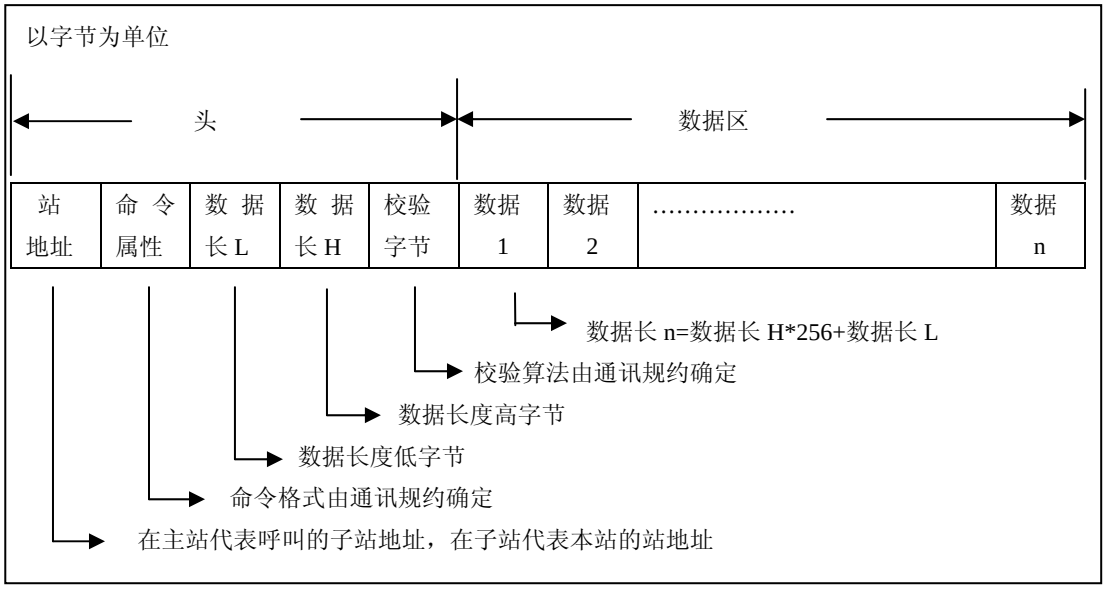
与主站主程序不同之处是少通讯接收程序，而 Wave1()是仿真程序。



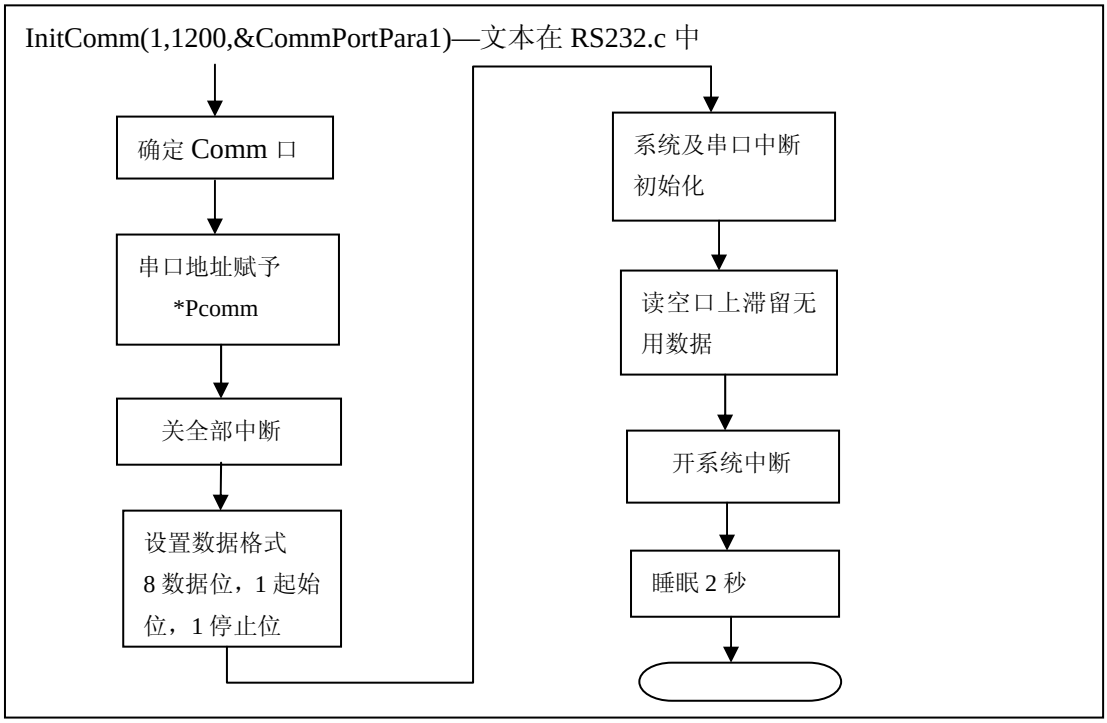
4．4 通讯程序

系统通讯程序是 PC 机与 PC 机之间通过 COMM 串口通讯程序，用 C 语言编写，采用中断方式工作。主站和从站的通讯程序相同，都具备收发功能，是一个实用程序。通讯策略是主站向局域网上广播索取数据的命令，每次指明某一从站回答，网上每一个从站同时收到广播命令，确认是对本站有效命令后立即向局域网发送本站数据。命令和数据按事先约定的格式装配。

网上传送的数据流结构如下：



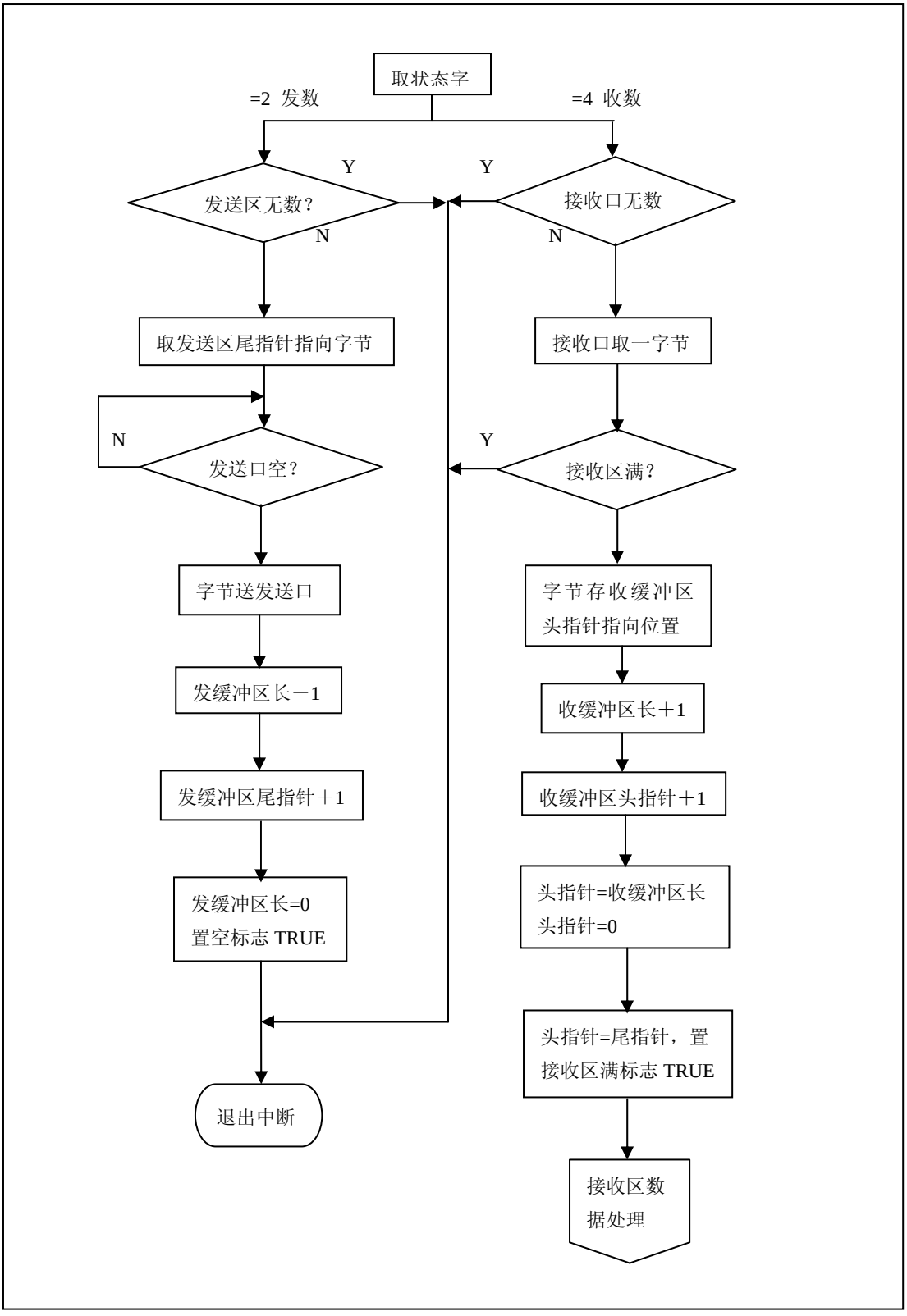
由主程序框图知道，主程序调用了 InitComm(1,1200,&CommPortPara1) 串口初使化函数和 InitpPARA()串口数据缓冲区初使化函数，其流程图如下：



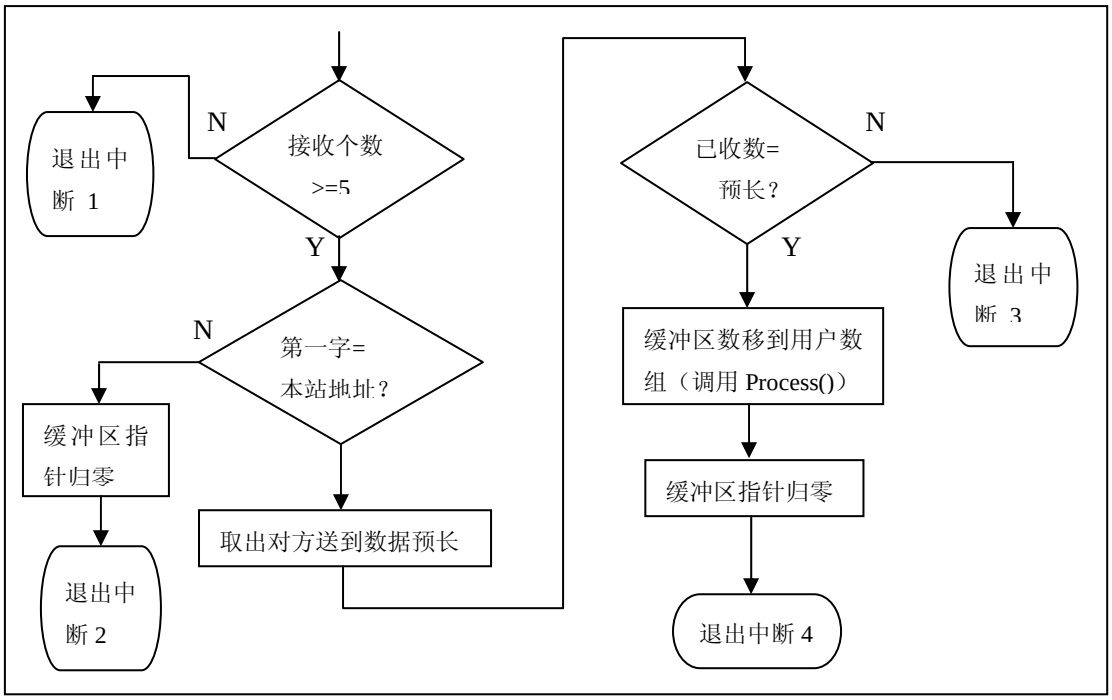
InitpPARA()函数把串口接收发送数据缓冲区清零，缓冲区头指针、尾指针及长度归零，缓冲区满标志置 FALSE，缓冲区空标志置 TRUE。

按中断方式运行的通讯中断服务程序（函数）在系统程序装载时，被作为常驻程序安排在内存中，由串口接收或发送数据时发出的中断请求信号激活。在主程序及其相关函数中看不到通讯中断服务程序（函数）的调用，要了解通讯过程，应读通讯中断服务程序（函数）。

通讯中断服务程序（函数）CommIntHandle1（）在 RS232.c 中，其流程图如下：



接上页接收缓冲区数据处理

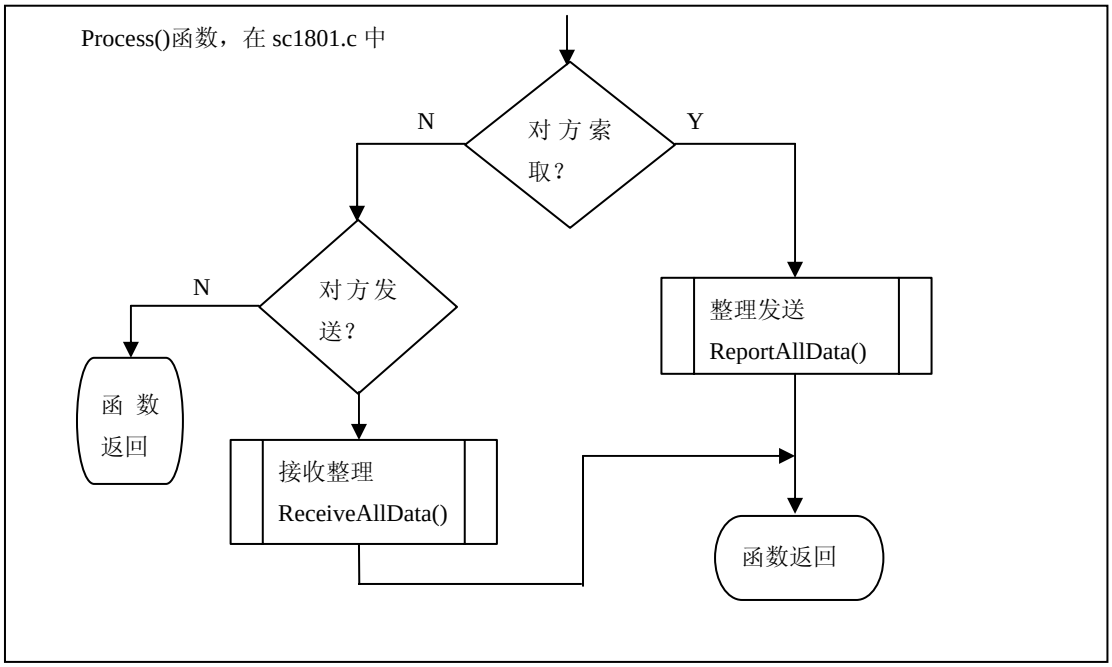


上图调用 Process()函数，把缓冲区数移到用户数组， 程序步骤分成

首先区别通讯对方是索取数据还是发送数据，如果是对方索取，则调用 ReportAllData()把 t[],sd1,sd2 按通讯规约转换后放入发送缓冲区，然后执行 outportb(CommPortPara1.DataPort, ucTemp)发送第一字节，启动发送的中断进程。

如果是对方送数，则调用 ReceiveAllData()，把接收缓冲区数据取出，按通讯规约转换后放入 t[],sd1,sd2。

其流程图如下：



4．5 数据采集程序（数据仿真程序）

数据仿真程序（Wave1()）在主站（sheji_c1）中仅完成数据十进制转换一项功能，在从站（sheji_c1）则要完成全部数据仿真。

要读懂数据仿真程序（Wave1()），必须清楚系统核心数据结构。系统核心数据结构分为模拟量数组 t[21]和开关量两个变量字节 sdi、sd2。其结构如下：

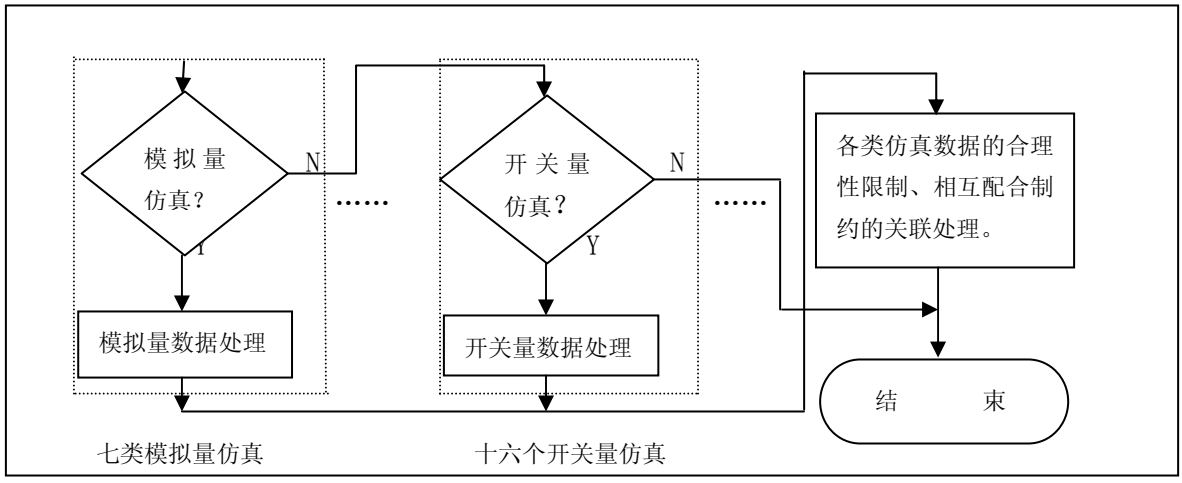
T[ ]	模拟量
0	空
1	1F 一号发电机三相电压有效值
2	2F 二号发电机三相电压有效值
3	3F 三号发电机三相电压有效值
4	4F 四号发电机三相电压有效值
5	1F 一号发电机温度值
6	2F 二号发电机温度值
7	3F 三号发电机温度值
8	4F 四号发电机温度值
9	6．3KVⅠ段母线电压有效值
10	6．3KVⅡ段母线电压有效值
11	1B 一号主变压器副边电流有效值
12	1B 一号主变压器原边电流有效值
13	1B 一号主变压器温度值
14	2B 二号主变压器副边电流有效值
15	2B 二号主变压器原边电流有效值
16	2B 二号主变压器温度值
17	35KV 母线电压有效值
18	353 出线电流有效值
19	355 出线电流有效值
20	35KV 系统频率

SD1 字节	十六进制位码	开关量名
Sd1.0	0X01	353 断路器
Sd1.1	0X02	351 断路器
Sd1.2	0X04	635 断路器
Sd1.3	0X08	6351 隔离刀闸
Sd1.4	0X10	6311 隔离刀闸
Sd1.5	0X20	631 断路器
Sd1.6	0X40	6321 隔离刀闸
Sd1.7	0X80	632 断路器

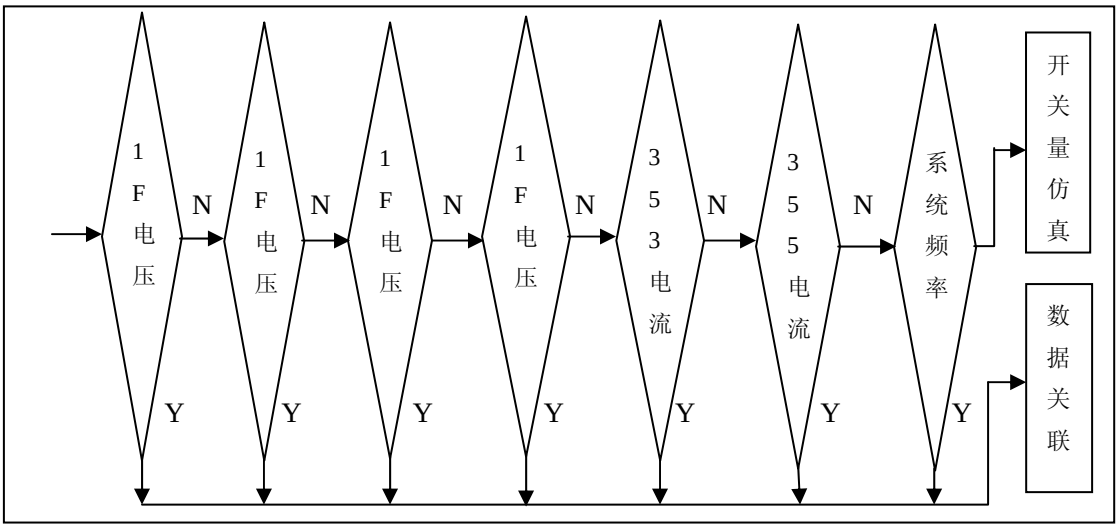
SD2 字节	十六进制位码	开关量名
Sd2.0	0X01	355 断路器
Sd2.1	0X02	352 断路器
Sd2.2	0X04	636 断路器
Sd2.3	0X08	6361 隔离刀闸
Sd2.4	0X10	6331 隔离刀闸
Sd2.5	0X20	633 断路器
Sd2.6	0X40	6341 隔离刀闸
Sd2.7	0X80	634 断路器

仿真程序分成三个层次介绍。

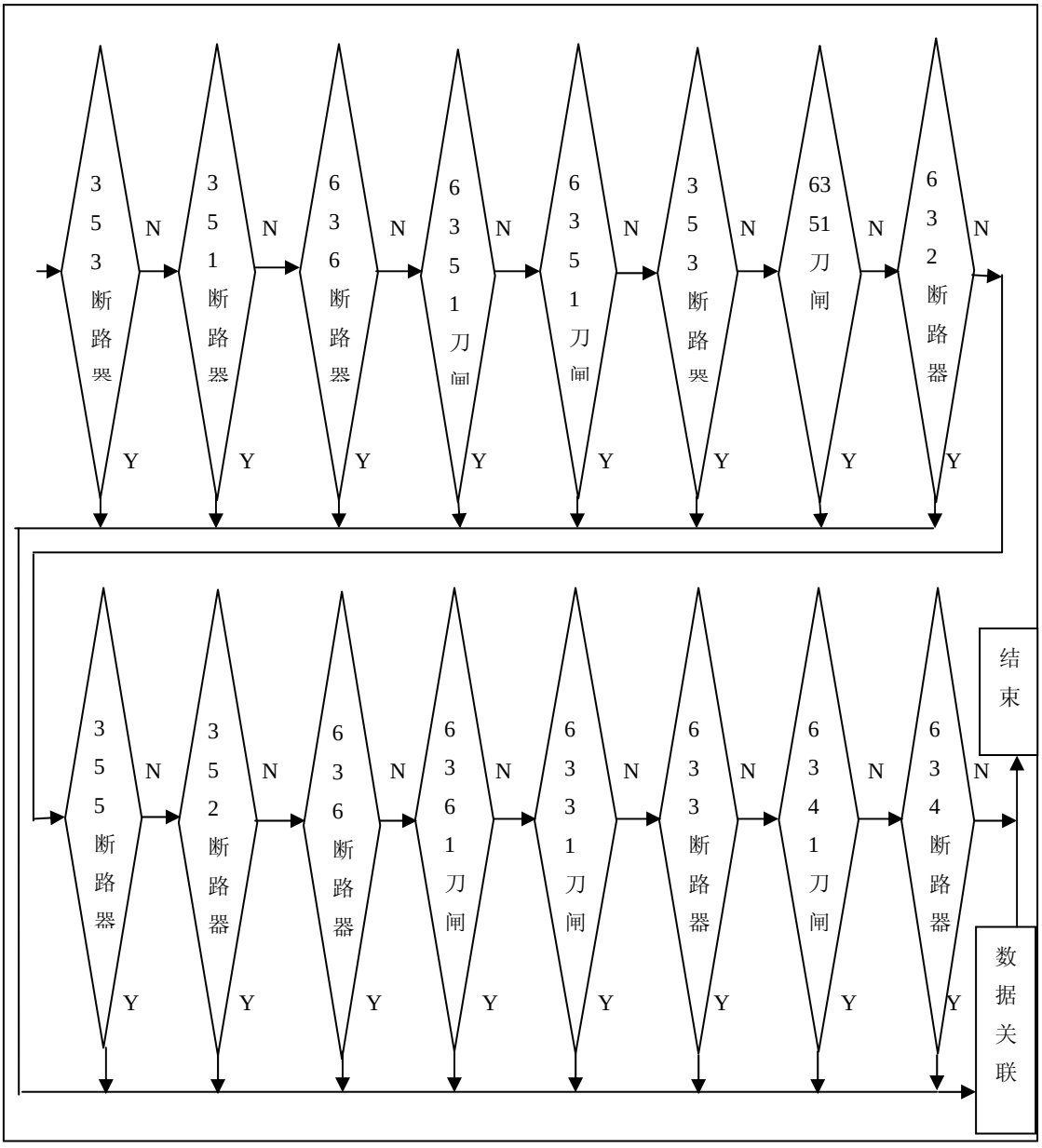
第一层是二类数据仿真程序之间的流程，也是系统程序的总框图，见下图：



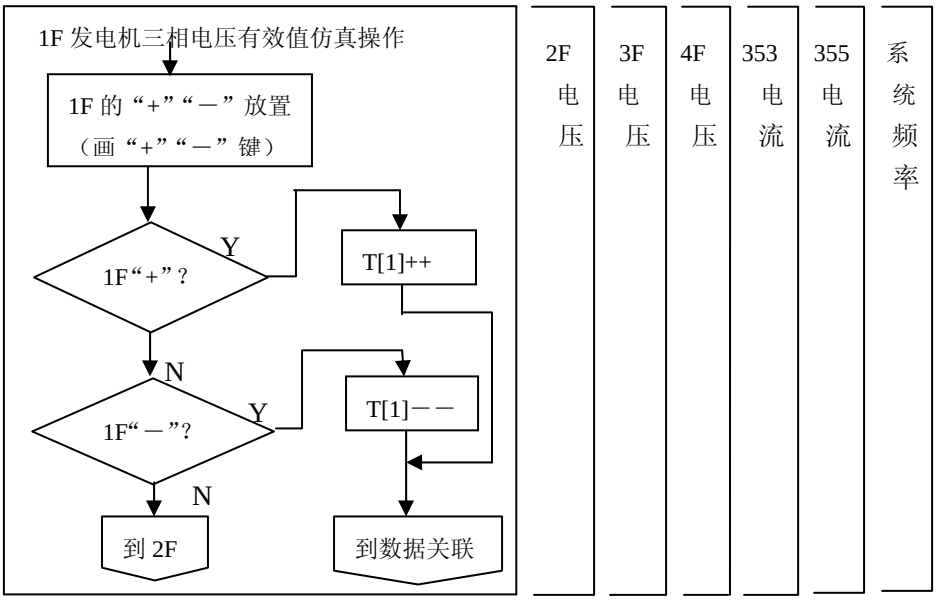
第二层七类模拟量仿真流程图



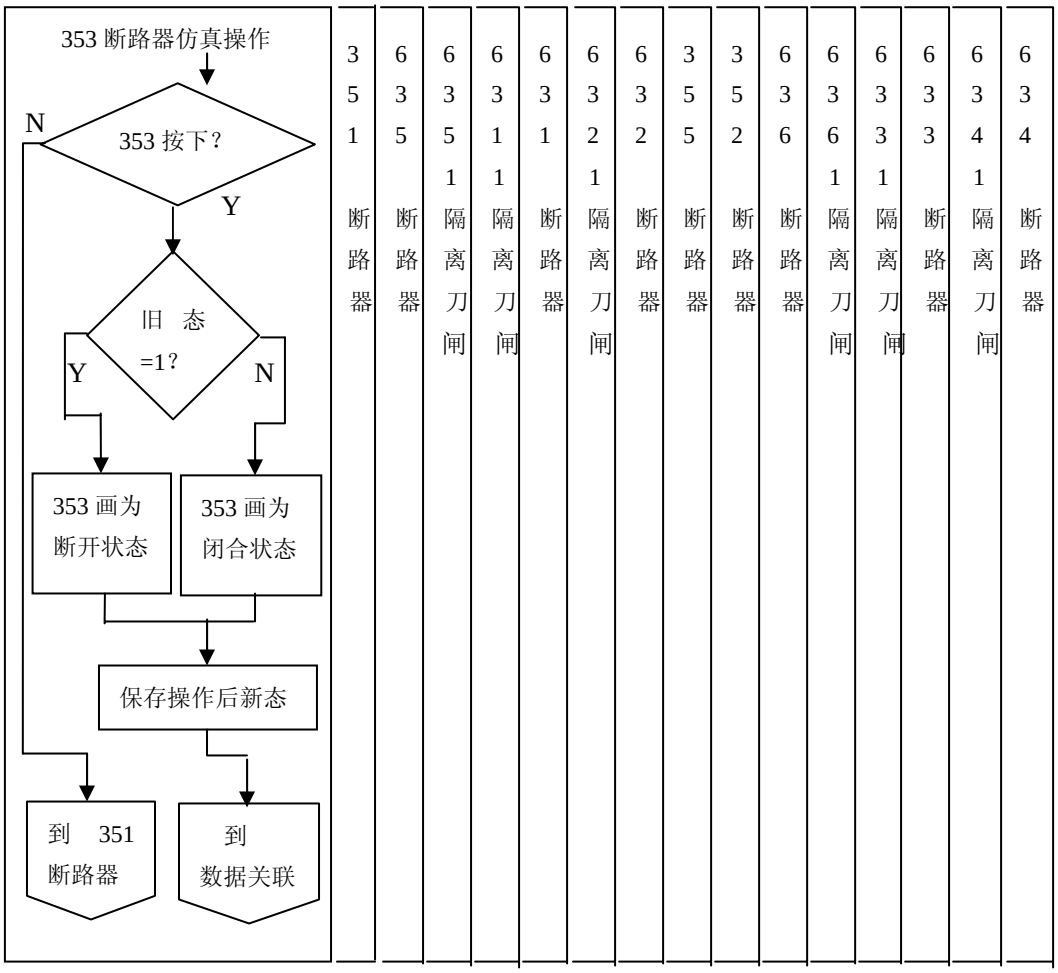
第二层十六类开关量仿真流程图



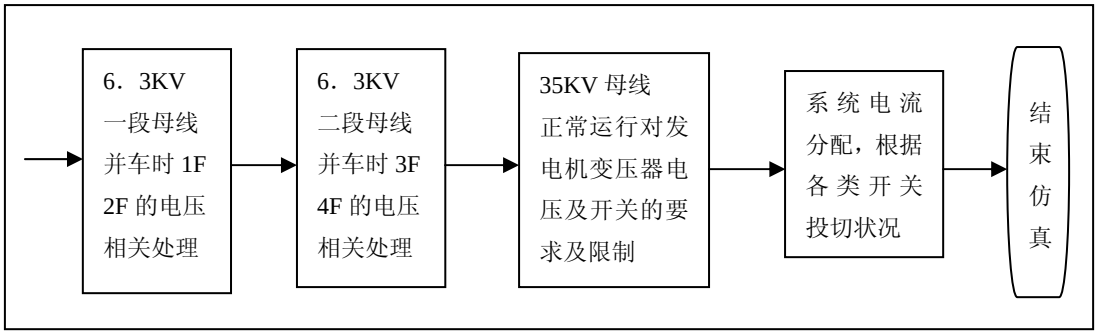
第三层模拟量仿真操作流程图



第三层开关量仿真操作流程图



流程图中把各类仿真数据的合理性限制、相互配合制约的关联处理称为数据关联程序，其结构与功能如下：



4．6 鼠标管理程序

仿真程序 Rt_data.c 调用了 button..c 中的鼠标移动坐标识别函数和在屏幕上显示“＋”“－”键函数，分别是：

光标坐标识别函数 getkey12n(int x,int y)，用于“＋”“－”键的位置及是否被按下识别。

光标坐标识别函数 getkey10n(int x1,int y1 ，int x2,int y2)，用于断路器及隔离刀闸位置及是否被切换识别。

画“＋”键函数 Button30n(int x,int y)，画出凸形按钮。

画“－”键函数 Button32n(int x,int y)，画出凸形按钮。

画“　”键函数 Button31n(int x,int y)，画出凹形按钮。

4．7 画面显示程序

在 fangzhen.c 中调用 subd1()函数在屏幕上画系统电气主接线图。

第五章 BC30 使用及程序调试

BORLAND C++（本讲义简称 BC30）对 TURBO.C 向下兼容，这里只介绍与 TURBO.C 不同的相关操作。

5.1.1 Borland C++窗口

集成环境中的大多数操作发生在窗口中，窗口是屏幕的一块区域，可以打开、关闭、移动、重定义大小、放大、缩小、覆盖。

集成环境可以打开多个窗口，但在任何时候只能有一个窗口是活动的，活动窗口是当前用户正在上面进行操作的窗口，选取的命令、键入的正文一般只作用于活动窗口。如果在在几个窗口中打开同一个文件，对该文件的操作将适用于所有打开该文件的窗口。

活动窗口是容易识别的，带有双线边框窗口即是。活动窗口总是有一关闭图标、放大图标及滚动条。如果窗口间有覆盖，活动窗口总是位于所有窗口的项端（即最前面）。

窗口分几种类型，但大多有如下的共同点：

- 一标题条。
- 一关闭图标。
- 滚动条。
- 一放大图标。
- 窗口号（1~9）。

编辑窗口在下角还显示前行号、列号，如果对文件作了修改，行号、列号的左边将出现

一个星号（*）。

一个典型的窗口如图 1.1。

窗口的关闭框位于窗口的左上角，单击关闭框能快速关闭该窗口，也可以选取菜单命令 Window|Close。Inspector 窗口和 Help 窗口被看作是临时窗口，可以按 ESC 来关闭。

标题条是窗口顶端的一水平条，标题条上有窗口名和窗口号。在标题条上快速双击将放大该窗口，拖标题条可以移动窗口位置。

放大框位于窗口的右上角。如果该处的图标为一箭头（↑），则可以单击该箭头图标把窗口放至最大；如果该处的图标为一双向箭头，则窗口已处于最大状态，这时可以单击该图标把窗口还原成原先的尺寸。从键盘可以选择 Window|Zoom 菜单命令缩放窗口。

Borland C++中先打开的九个窗口在其右上角有窗口号。按 Alt 与窗口号的组合键可以激活对应的窗口。例如，如果求助信息窗口的窗口号是 5，但被其它窗口盖住了，按 Alt-5 将把求助信息窗口调至最前面。

滚动条有水平和垂直的两种。

以滚动条上按鼠标键可以使窗口中的文件滚动。在两端的箭头上单击鼠标，每次单击都滚动一行，按住鼠标键连续滚动。在滚动条两侧的阴影区域上单击鼠标，每次单击都滚动一页。把滚动条指针拖 至某一位置可以快速使窗口中的文件作相应移动。

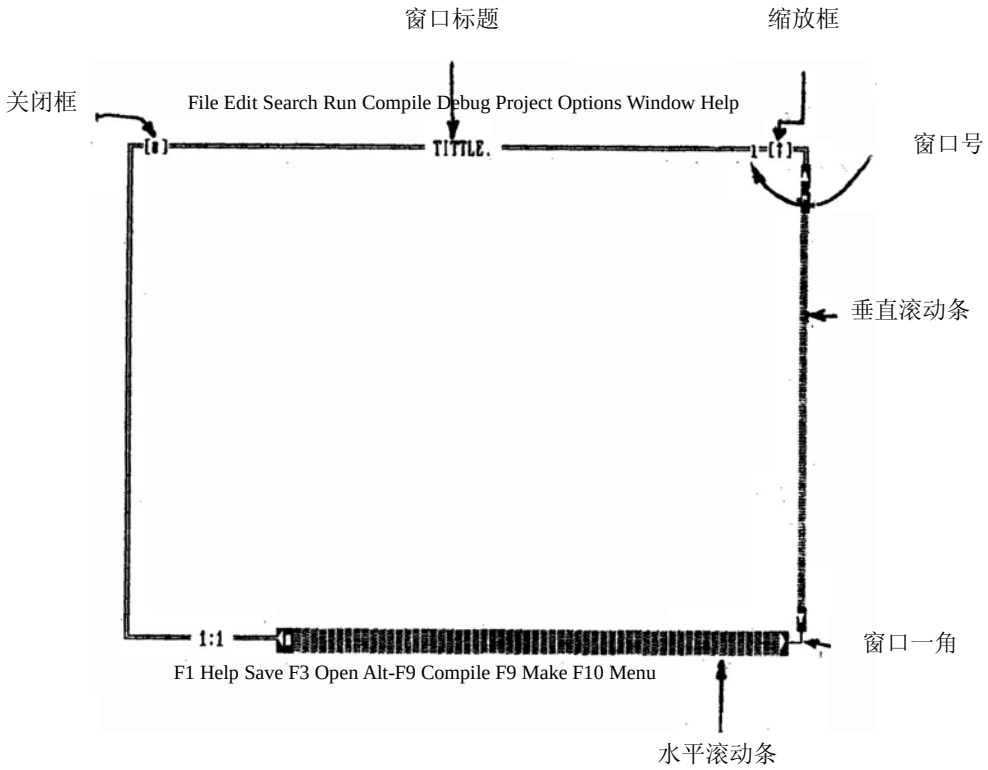


图 1.1 一个典型的窗口

滚动条显示出当前光标在文件中的相对位置。

拖曳窗口的四个角都可以使窗口变大或变小，对键盘来说可以在 Window 菜单中选取 Size/Move 命令重定义窗口大小。

5.1.2 窗口管理

表 5.1 列出了 Borland C++中的各种窗口操作，注意这些操作不是非用鼠标器不可，键

盘一样可行。

目的	可用的方法
打开一个编辑窗口	选取菜单命令 File|Open 打开一文件并显示一窗口中。
打开其它窗口	从 Window 菜单中选项所需的窗口。
关闭一窗口	从 Window 菜单中选取 Close 命令，或者在关闭图标上按鼠标键。
激活一窗口	在窗口的任意部位上按鼠标键；或者按 Alt 与窗口号（1~9，位于窗口右上角）的组合键；或者选取 Window|List 或者按 Alt—O 从窗口列表中选取一窗口；或者选择 Window/NEXT 激活下一个窗口（按打开窗口的顺序）。
	移动活动窗口拖曳标题条或选菜单命令 Window/Size/Move，用箭头键把窗口移至想安放的位置，再按 Enter。
重定义活动窗口的尺寸	拖曳窗口的角；或者选取菜单命令 Window|Size/Move，在按住 Shift 键的同时用箭头键改变窗口大小，再按 Enter。
放大活动窗口	在窗口右上角的放大图标上按鼠标键;或者在窗口标题条上双击；或者选取菜单命令 Window|Zoom。

表 5.1 窗口操作

5.1.3 状态行

状态行位于屏幕的底端，作用如下：

- 提示活动窗口中当前可用的基本击键入或简捷键（或热键）输入
- 允许用户按鼠标键来触发相应操作，而不必从菜单中选取或按简捷键。
- 告诉程序在干什么，如，保存编辑文件时显示“Saving filename…”。
- 对所选菜单命令和对话框项提供提示信息。

选择了一个菜单标题或命令，状态行显示一行信息表示所选项的功能。

5.1.4 对话框

后面跟着省略号（…）的菜单命令将会打开一个对话框。对话框用来显示或设置多个选项是挺合适的。在对话框中用户与五种基本的屏幕控制打交道，它们是：单选按钮，复选框、操作按钮，输入窗和列表窗。

如果有彩显，Borland C++可用不同的颜色显示对话框的不同部分。

该对话框有三个标准按钮：OK，Cancel 和 Help，如果选 OK 则确认对话框中的选择，如选 Cancel 则不改变原先的选择设置，即使发生任何操作出只是清除对话框。选择 Help 将打开一个求助窗口显示有关该对话框的求助信息。ESC 约定为 Cancel 的简捷键即使对话框中没有 Cancel 按钮，该约定成立。

如使用鼠标器，只要在相应按钮上按鼠标键就可以了。若使用键盘，可按 Alt 加上加亮字符的组合键。如，Alt+K 表示选 OK，按 Tab 或 Shift+Tab 分别表示在对话框中前后移动，成为活动状态则该处加亮显示。

在该对话框，缺省值是 OK，也就是说按 Enter 就可选择该按钮（在单显系统中用箭头

号表示缺省按钮，彩显中用加亮表示）。注意按 Tab 移至一按钮使该按钮成为缺省按钮。

5.1.4.1 复先框与单选按钮

对话框中有一种复选框，[X]表示该项被选通，[ ]表示没被选通。在复选框或其标题正文上按鼠标键，或者按 Tab 使复选框加亮然后按 Spacebar，或者按 Alt 与加亮字符的组合键，可以对应的复选框。用户可以同时选通多个复选框。

如果有几个复选框对应于某种含义那么它们将以复选框组的形式出现，按 Tab 可以在组间切换。选定复选框组后可以用箭头键选取用户要的项然后按 Spacebar 选通。对于单色显示器活动复选框或复选框组用符号“>>”指示，按 Tab 键“>>”标记将移至一下一个复选框组或单选按钮组。

对话框中还有一种单选按钮，其特性类似汽车收音机的选台按钮，任何时刻有且只有一个按钮被选通，选通某按钮将自动使原来选通的按钮断开。

单选按钮与复选框的不同之处在于单选按钮表示一个互斥的选择，正因为选择，单选按钮总是成组出现，任何时记刻刚好有一个（不多，也不少）单选按钮被选通。在单选按钮或其标题正文上按鼠标键可以选通该按钮。对键盘输入来说，可以按 Alt 与加亮字符的组合键选通一单选按钮，或者按 Tab 使该单选按钮组加亮然后用箭头键选通一单选按钮，再次按 Tab 或 Shift—Tab 离开该单选按钮组则单选按钮选通完毕。

5.1.4.2 输入框与列表

对话框可以包含输入框。在输入框中可以输入适当的正文。大多数正文编辑功能键（如箭头键，Home,End,Ins 等）在正文框中有效。输入框还有自动翻卷功能（键入正文达到输入框边界后），输入框边界上的箭号表示另外还有正文没能显示出来，可以在箭头上按鼠标键翻卷，也可以拖曳正文。假如想在输入框中输入控制字符（如ˆLˆM），那么必须前缀ˆP。这样ˆPˆL。（这有利于查找字符串。）

如果输入框的右边有一↓图标，则表示对应有一历史输入列表，按 Enter 可以从该列表中选取一项，历史输入列表记录对应该输入框的最近一些输入。例如查找输入框有一个历史输入列表保存有以前查找过的正文。按下箭头键↓或者在↓图标上按鼠标键可以再次输入用户已经输入过的正文。也可以对历史输入列表中的项进行编辑。按 Esc 可以不作选择退出历史输入列表。

用菜单命令 Options|Enviromnmet|Desktop 可以控制是否把历史输入列表存入桌面。图 1.4 是查找输入框的一个历史输入列表，假定原先的有过六次输入。

列表框是许多对话框的终结成分，列表框让用户能够不退出对话框就能在可变长度列表（通常是文件名）中作浏览和选取操作。当一闪烁光标出现在列表框中时，如果知道要找寻的工程，可以键入该工程（或者打开头几个字母），Borland C++能够自动为用户查找出来。

在列表框上按鼠标键或者选列表框标题上的加亮字符（或按 Tab 键加亮列表框），可以激活列表框。在显示列框之后用户可以用滚动框在列表中作移动操作，也可以用上下箭头键来移动。

5.2 配置和工程文件

利用配置文件用户可指定在 IDE 中的工作方式，工程文件包含构造工程所需的所有信息，但不影响用户使用 IDE。

5.2.1 配置文件

配置文件 TCCONFIG，TC 只包含环境（或者说全局）的信息，其信息包括

• 编辑组合键和宏。

• 编辑模式设置（如自动缩进，制表键等）。

• 鼠标器性能。

• 自动保存标志。

构造工程定义的程序，不需要配置文件。

当开始程序对话时，Borland C++首先在当前目录下查找 TCCONFIG，TC，然后进入包含 BC，EXE 的目标，Turbo C++也在当前目标中查找，但它不查找 TCCONFIG，TC，而是查找包含 TCW.EXE 的目录。

5.2.2 工程文件

IDE 把构造程序所需要的信息放入二进制工程文件，扩展名为，PRJ，工程文件包含的其它信息如下：

• 编译器，连接器，make 和库管理程序选项。

• 目录路径。

• 构成工程的所有方便列表。

• 特别的译码器（如 Turbo Assembler）。

另外，工程文件还包含有关该工程的一般性信息，如编译统计资源（显示在工程窗口）和带高速缓存的自身依赖性信息。

IDE 的工程文件对应于提供的命令行编译器的，CFG 配置文件（缺省的命令行编译器配置文件是 TURBOC.CFG），实用程序 PRJCFG 可将，PRJ 文件转换成.CFG 文件，可也是把.CFG 文件转换为.PRJ 文件。

可以用以下三种方法装入工程文件。

1、启动 Borland C++时，在 BC 命令后给出带扩充名，PRJ 的工程名，例如

BC myproj.prj

2、必须带扩充名.PRJ 区别于源程序文件。

3、如果当前目录只有一个.PRJ 文件，那么 IDE 假定该目录对应于该工程并自动装入，因此当前目录中包含一个工程文件时键入 BC 将自动装该工程文件。

4、在 IDE 中，用菜单命令 Project|Open Project 装入一个工程文件。

5.2.2.1 工程目录

如果工程文件是从非当前目录的另一目录装入的，则该目录成为当前 DOS 目录，这样可以根据 Options|Directories 对话框中的相对路径定义用户的工程，也使工程可以从一个驱动器转换到另一驱动器或从一个目录转换到另一目录。注意，工程装入完毕再改变目录将使得相对路径不正确，也使工程不能构造，在这种情况下可以把当前目录仍然改回到装入工程的目录。

5.2.2.2 桌面文件

每一个工程文件有一相应的桌面文件（prjname.DSK），该文件包含有对应工程的状态信息。桌面文件包含的信息不是用来构造工程的，其所有信息直接对应于该工程。桌面文件包含有如下一些信息：

（可以用菜单命令 Options|Environment|Desktop）把某些选项设置通或断状态。

•工程中每个文件的上下文相关的信息（context information）（例如在文件中位置信息）。

• 各种输入窗（如查找字符串，文件名掩码等等）的历史输入列表。

• 桌面上窗口的布局。

• 剪贴板的内容。

• 监视表达式。

• 断点。

5.3 Search 菜单（Alt+S）

Search 菜单让用户在程序文件中查找正文，函数声明和出错位置

5.3.1 Find(Ctrl+Q+F)

Find Text 对话框包括几个按钮和复选方框：

[X]Case sensitive 选通该复选方框指示 IDE 区分大小写。

[ ]Whole word only 选通它表示 IDE 只查找单词（即字符串的两端必须有标点或空格）。

[ ]Regular expression 选通它表示 ID 识别类似 GREP 的通配符，通配符有^，$，

*，+，[ ]，和\。它们的含义如下：

^ 字符串开始处的^匹配一行的开始

$ 表达式尾部的美元符号匹配行的末尾

. 句点匹配任意字符

* 一个字符后跟星号匹配字符重复任意次（包括零次）如 bo*匹配 bot,b,boo 和 be。

+ 一个字符后跟加号匹配字符重复除零外次的任意次。如 bo+匹配 bot 和 boo,但不匹配 be 和 b。

[ ] 若干字符包在方括号内匹配方括号内的任一字符（而不是别字符）。如[bot]匹配 b,o 和 t。

[ ] ^为方括号的第一个字符其含义是“非”或“除”…以外，所以[^bot]匹配除 b,o,t 以外的任一字符。

[-]方括号中的连字符表示某一范围内的字符，如[b-o]匹配从 b~o 的任一字符。

\ 位于通配符前的反斜杠表示 BorlandC++不把它们作为通配符，而只是一般的字符。如\^匹配符号^而不是匹配一行的开始。

在输入窗内键入字符串并选取 OK 按钮则开始查找，如果选取 Cancel 则取消，假如想输入以前查找过的字符串，可以用↓调出历史输入表再从表中选出所要输入的字符串。

用户可以选取编辑窗口中当前光标处的单词然后调用 Search|Find 命令查找该单词。按→可以从编辑窗口中取出附加的字符。

Direticon

( )Forward

( )Backward

Direction 单选按钮决定从起始位置（可以用 orign 单选按钮设置）开始的查找方向。

Scope

（ ）Global

（ ）Selected text

Scope 单选按钮决定查找的范围，可以整个文件（全局）或近选正文

Origin

（ ）From Cursor

（ ）Entire Scope

Origin 单选按钮决定查找的起始位置，如果选 Entire Scope，则由 Direction 单选按钮决定查找是从查找范围的头开始还是查找范围的尾开始，查找范围由 Scope 单选按钮决定。

5.3.2 Replace（仅限 Alternate Ctrl+Q+A）

Replace Text 对话框有一些单选按钮和复选方框，其中许多同 Find Text 对话框中的相同（前面已讨论过），附加的复选方框 Prompt on Replace 控制每次替代时是否给予提示。

输入查找字符串和替换字符串后按 OK 按钮或 Change All 则会开始查找，也可以按 Cnacel 按钮取消。如果想输入以前输入过的字符串可以按↓调出历史输入列表加以选。

如果找到指定的正文且提示是否作替换，若选 OK，它就会查找且替换最先找到的正文，选取 Chandge All 按钮就会替换所有查到的正文，能查到的正文由 Direction,Scope 和 Origin 三组单选钮钮决定。

5.3.3 Search Again(CPU 用 F3，Alternate 用 Ctrl+F)

Search|Search Against 重复上次 Find 或 Replace 命令，上一次 Find 或 Replace 对话框中的设置信息保持不变。

5.3.4 Go to Line Number

Search|Go to Ling Number 命令提示用户所要查找一行号。

IDE 在编辑窗口在左下角显示当前行号和列号。

5.3.5 Previous Error(Alt+F7)

Search|Previous Error 命令把光标移到前一个出错或警告位置处。只有当消息窗口中有与行号相关的信息时该命令才可执行。

5.3.6 Next Error(Alt+F8)

Search|Next Error 命令把光标移到下一个出错或警告位置处，只有当消息窗口中有与行号相关的信 息时该命令才可执行。

5.3.7 Locate Function

Search|Locate Function 命令显示一个对话框以便让用户输入要查找的函数名，只有调试过程中该命令下才可执行。

键入函数名或从历史表明↓选一个名字，与 Find 命令相反，该命令查找函数声明，而不是其使用的实例。

5.4 Project 菜单（Alt+P）

Project 菜单包含了所有工程管理命令：

- 创建一个工程。
- 从工程中删除文件或向工程添加文件（如何使用工程管理程序见第五章“多文件工程管理”）。
- 指定与用户的源程序文件一起被译码（编译、汇编）的程序（Borland C++独有）。
- 设置工程中文的选项（Borland C++独有）。
- 指定切换程序使用的命令行取代（Override）选项（Borland C++独有）。
- 指定目标模块名、放置的目录是否覆盖、是否包含调试信息（Boralnd C++独有）。
- 显示工程中特定文件的嵌入文件。

5.4.1 Open Project

Open Project 命令显示 Load Project File 对话框，让用户输入一个工程文件名以选取、装入一个工程或创建一个新工程。

在该对话框中选择文件的方法类似于 File|Open 对话框，选择的文件将用作一个工程文件，里面包含有构造程序执行文件需要的一切信息，创建，EXE 和.MAP 文件时 WindowsC++用到工程文件。典型的工程文件有扩展名.PRJ。

5.4.2 Close Project

Project|Close Project 命令可以删除用户的工程并返回缺省工程（TCDEF.DPR）。

5.4.3 Add Item

Close Project|Add Item 命令在工程列表中加入一个文件，调出一个 Add to project 对话框。

该对话框类似于 Open a File 对话框（命令 File/Open）。选取 Add 按钮把文件列表中的当前加亮文件加进工程窗口，亮条移进工程窗口之后所选文件立即被加入工程窗口的文件列表，每加进一个文件亮条前进一个位置。（如果工程窗口是活动窗口用户可以按 Ins 加入一个文件。）

5.4.4 Delete Item

Project|Delete Item 命令从工程窗口删除一个文件。如果工程窗口是活动窗口用户也可以按 Del 删除一个文件。

5.4.5 Local Options

Local Options 命令打开如图 2.8 所示的对话框。

不支持下列命令行选项；c，Efilename,e,Ipathname,L,Ix,M,Q,Y。

Override Options 对话框让用户给特定的工程文件模块选用命令行取代选项。

所有选 Translator 选项装入 Transfer 对话框的程序显示在 Project File Translaiors 列表中。

[ ]Overlay this module

如果想使所选模块或库（或工程项）使用覆盖，选通 Overlay this module 选项，该选项局部于一个文件。如果没有选通 Options|Linker|Settings 中的 Overlaid DOX EXE 选项，该被忽略。

[ ]Exclude debug information

选通 Exclude Debug Information 选项禁止把该模块中的调试信息带入.EXE 文件。

调试大型程序时用到该选项开关。只要改变该选项开关重新连接（不需要重编译）就可以了。

[ ]Exclude from link

选通 Exclude from Link 选项禁止连接该模块。

5.4.6 Include Files

选取 Project[Include Files]可以显示如图 2.9 所示的 Include Files 对话框，想查看用户从工程窗口选取的文件包含了哪些嵌入文件时使用该命令。在工程窗口中可以按 Spacebar 调出 Include Files 对话框，如果还没有构造一个工程该命令不可执行。

可以滚动查看该文件列表。按 Enter 调用所省操作即把受选嵌入文件显示在编辑窗口中。

5.5 Window 菜单（Alt+W）

Window 菜单包含窗口管理命令，用该菜单打开的窗口多数有所的标准字符串成份，如滚动条、关闭图标和放大图标，有关标准窗口成分的介绍见第一章“IDE 基础的 BorlandC++窗口”一节。

Tubo C++IDE 的 Window 菜单与 Borland C++IDE Window 菜单有点不同，窗口管理命令没有 BoraldnC++多，但有些命令相同，如果用户会使用 Windows，那么用户将会知道在 Turbo C++IDE 下如何管理窗口，只需用同样的命令。

5.5.1 Cascade（CUA 用 Shift+F4）

Windlw|Cascade 命令层叠所有打开的窗口。

5．6 Option 菜单

Directoreis 选项必须正确填写。

Include Directoreis 选项，应填写系统头文件的存放路径，比如本实验是 c:\bc30\include.

Library Directoreis 选项，应填写系统库文件的存放路径，比如本实验是 c:\bc30\lib.

Output Directoreis 选项，应填写编译的运行文件存放路径，比如本实验是 d:\XXXX.

Source Directoreis 选项，应填写被编译的 C 程序存放路径，比如本实验是 d:\XXXX.

“XXXX”是用户文件夹名。

注意：上面介绍的以外的选项，初学者尽可能不作修改。

开机步骤：

一．开电源，等待 WIN98 系统完全建立。

二．点击 WIN98 桌面的 MSDOS 快捷键，进入 MSDOS 系统。

三．键入“C:”，或“CD\”,然后按“Enter”回车键，进入 C 盘根目录。

四．键入“UCDOS”，运行 UCDOS 中文操作系统。

五．键入“D:”，进入 D 盘根目录。

六．键入“cd\XXXX”，进入用户路径，假定用户文件夹在 D 盘，“XXXX”是用户文件夹名。

七．键入“BC30”，运行 Borland C++，进入其窗口集成环境。