

# E&C

# 1993

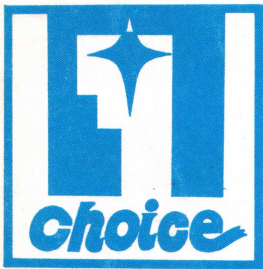
●一九九三年 ●总期第96期

# 3

# 電子

ISSN 1000-1077

# 與 電腦



## 武汉创意电子研究所隆重推出 CYSCB-2 MCS-51、8098兼容单片单板机

集51与98于一身  
融应用与开发于一体

欢迎选用!



# ● ELECTRONICS AND COMPUTERS ●





# ● 智力开发 明智选择



# 电子工业出版社 93年向读者奉献部分优秀新书

代号	书 名	邮购价
F—1	标准集成电路数据手册—TTL 电路	46.00
F—2	标准集成电路数据手册—运算放大器	37.00
F—3	标准集成电路数据手册—音响电路	46.00
F—4	标准集成电路数据手册—通信电路	40.30
F—5	标准集成电路数据手册—PAL 电路	70.00
F—6	标准集成电路数据手册—高速 CMOS 电路	52.00
F—7	微机常用集成电路手册	73.60
F—8	微机显示技术实用手册	69.00
F—9	中外集成电路简明速查手册 TTL, CMOS 电路	64.40
F—10	机电元件技术手册—设计、制造、使用、维修	109.00
F—11	半导体器件制造工艺常用数据手册	52.00
F—12	中国机械电子工业年鉴鉴电子卷—92 年	52.00
F—13	中国机械电子工业企业单位名片集	55.00
F—14	555 集成电路应用 800 例	28.80
F—15	中外电信集成电路使用手册	26.00
F—16	传感器应用及其电路精选(上、下册)	25.30
F—17	万用表测量技巧	20.00
F—18	显示器电路原理与维修	34.50
F—19	英汉计算机技术辞典	43.70
F—20	日英汉机电技术大词典	188.00
F—21	现代电子科学技术大词典	138.00
F—22	电子测量仪器器材技术手册	86.00
F—23	FOX BASE 实用大全	21.00
F—24	表形码编排汉语字典	13.00
F—25	实用 DOS 详解词典	15.00
F—26	92 年电子与电脑合订本	19.00
F—27	现代汽车电路图集(93 年新书 1~5 册)	96.00

邮购办法: 邮局汇款: 北京万寿路电子工业出版社发行部 邮购科

银行汇款: 开户行: 北京工商银行翠微路分理处

帐号: 661036—40 户名: 电子工业出版社发行部

电话: 813693 邮编: 100036

注: (1) 发书周期 30 天左右, 如邮路丢失, 向我部索赔, 如订购千元以上优惠 5% (自行扣除)。

(2) 购书清单请写在汇款单“附言栏”内, 或同汇款一起寄来(也可写书代号), 不必另寄信函。



表一 Z8611, INTEL8051, MC6801 结构比较表

特 性	ZiLOG, Z8611	INtel8051	Motorla Mc6801
在片 ROM	4K×8	4K×8	2K×8
通用目的寄存器	124	128	128
特殊功能寄存器			
状态寄存器	16	16	17
I/O 控制寄存器	4	4	4
I/O 端口			
平行线	32	32	29
端口	4 个 8 位	4 个 8 位	3 个 8 位, 1 个 5 位
交握控制	3 个端口可以	没有	1 端口可以
中断			
源	8	5	7
外部源	4	2	2
矢量	6	5	7
优选数	48 可编程	2 个可编程	没有可编程
屏蔽	6	5	6
可外扩存储器	120KB	124KB	64KB
堆栈			
堆栈指针	16 位	8 位	16 位
内部堆栈	可以 8 位指针	可以	可以
外部堆栈	可以	不可以	可以
计数器/定时器			
计数器	2 个 8 位	2 个 16 位或 2 个 8 位	一个 16 位
预置器	2 个 6 位	16 位设预置 8 位可 5 位预置	没有
寻址方式			
寄存器寻址	可以	可以	没有
间接寄存器寻址	可以	可以	没有
间址	可以	可以	可以
直接寻址	可以	可以	可以
相对寻址	可以	可以	可以
立即寻址	可以	可以	可以
隐含寻址	可以	可以	可以
变址寄存器	124(任一通用寄存器)	1.(利用累加器做 8 位移位)	1.(利用 16 位变址寄存器)
串行通讯接口			
全双工 UART	可以	可以	可以
中断方式送/收	送/收均需中断	每次中断可送/收	每次中断可送/收
寄存器双缓冲	接收器	接收器	传送/接收
串行传输速率	8MC 时:62.5KB/S 12MC 时:93.5KB/S	12MC 时:187.5KB/S	4MC 时:62.5KB/S
速度			
指令执行平均	2.2 $\mu$ s	1.5 $\mu$ s	3.9 $\mu$ s
速度	1.5 $\mu$ s/12MC 时		
长指令	4.25 $\mu$ s 2.8 $\mu$ s/12MC 时	4 $\mu$ s	10 $\mu$ s
时钟频率	8MC 或 12MC	12MHz	4MHz
掉电模式	保护 124 个寄存器	保护 128 个寄存器	保护 64 个寄存器
程序交换时 所需保护内容	保护程序指针和标志	保护程序指针编程必须保护 所有寄存器内容	保护程序指针程序状态字累 加器和变址寄存器的内容
开发装置	40 脚背靴式(8613) 64 脚(8612) 40 脚无 ROM(Z8681)	40 脚 8751	40 脚 68701
EPROM	4KB(2732)	4KB(2732)	2KB(2716)
	2KB(2716)		



# 电子与电脑

一九九三年 总期第 96 期

## 目 录

- 综述 •
  - 从三种通用的八位单片机硬、软件比较来看 z8 系列单片机的优越性 ..... 董伯明(2)
- PC 用户 •
  - 如何用 C 语言编制数据采集软件 ..... 冯品如 张新民(4)
  - 谈谈 MS-DOS 5.0 新增命令 DOSKEY ..... 唐银红(7)
  - 用 SET 命令解决 COMMAND.COM 的重新调入 ..... 叶 斌(11)
  - 任意盘任意子目录下调用 WS 的方法 ..... 何 捷(11)
  - C-dBASE III 数据文件结构及其与其它语言的共享 ..... 卢小平(12)
  - “中国青蛙”病毒 ..... 罗 亚(13)
  - 以毒攻毒——消除 DIR-2 病毒的简便方法 ..... 刘山水(14)
- 学习机之友 •
  - XMF-BASIC 全功能扩充 ..... 石永琳(17)
  - 实用程序二则 ..... 李成功(19)
  - 任意数的高精度乘法 ..... 张新莲(20)
  - ProDOS 系统内部结构剖析(续) ..... 廖 凯(21)
- 语言讲座 •
  - 第四讲 FORTH 中的数 ..... 丁志伟(23)
- 初、中级程序员软件水平考试 •
  - 基本算法(一) ..... 李 宁(26)
- 新书与软件 •
  - Turbo PASCAL V6.0 介绍 ..... 欧阳慎(29)
- 学用单片机 •
  - BJS-98 电路原理与 CDW 多窗口调试器 ..... 袁 涛 魏 峰(31)
  - MCS-51 单片单板机监控程序分析(中) ..... 江 琪 刘 葳(34)
- 电脑巧开发 •
  - PC 电力拖动控制 ..... 李文兵(36)
- 维修经验谈 •
  - 彩色显示器工作原理简述及故障诊断(上) ..... 胡野红(42)
- 电脑游戏机 •
  - 第五讲 游戏程序的编程特技(中) ..... 于 春(44)
- IC 应用 •
  - 高频开关稳压电源 IC 讲座
  - 第二讲 开关电源基本组态和企业标准 ..... 李秀华 张占松 梁全稳(47)
  - 集成电路滤波器及使用(一)(续) ..... 李兰友(50)
- 电脑通信 •
  - 计算机多媒体技术的新信息 ..... 文 波(51)
  - 传真机专题讲座 •
    - 第一讲 传真概述 ..... 张景生 张建军(53)
  - 电脑通信大家谈 •
    - PC 机之间最简单的通信 ..... (55)
- 读者联谊 •
  - PC 机最新教学、游戏软件介绍;华容道 ..... (49)

机械电子工业部电子工业出版社主办

编辑、出版:《电子与电脑》编辑部  
(北京 173 信箱 邮政编码:100036)

印刷:北京三二〇九厂

国内总发行:北京报刊发行局

国内统一刊号:CN11-2199

邮发代号:2-888

国外代号:M924

出版日期:每月 23 日

主编:王惠民 特约编审:苏子栋

责任编辑:杨逢仪

订购处:全国各地邮电局

国外总发行:中国国际图书贸易总公司

(北京 399 信箱 邮政编码 100044)

广告经营许可证:京海工商广字 147 号

定价:1.60 元



# 从三种通用的八位单片机硬、 软件比较来看 Z8 系列单片机的优越性

Zilog 中国区高级工程师——董伯明

目前我国从国外引进的四位、八位、以至十六位单片机有 Intel, Motorola, Philips, NS, psicrochip, Zilog 等多种。为帮助应用者在选型时能做到心中有数, 现就 Zilog Z8611, Intel 8051, Motorola MC6801 的三种八位的单片机在硬件构造及应用软件实践作一比较, 可使大家信服地看到 Zilog 公司推出的 Z8 系列单片机由于其设计思想的独一新颖, 导致性能是最优越的。

## 一、硬件结构比较(见本期封三)

1. Z8611 的显著优点: Intel 8051 和 MC6801 的 CPU 硬件结构设计采用传统的设计思想, 仅有一个累加器。而 Z8 是多累加器结构, 即在每一个 MCU 中的所有通用寄存器阵列均可视为累加器处理。NMOS 工艺的 Z8 系列 MCU 中有 124 个通用寄存器, CMOS 工艺的 Z8 系列 MCU 中有 236 个通用寄存器均可作为累加器。彻底解决了 MCU 中传统存在的“瓶颈”问题。因而采用 Z8 系列单片机来完成同一目标而编制软件显而易见的方便、灵活、短小、占内存少, 执行速度快, 举一个一次加法为例来说明上述问题, 要完成  $M_1$  加  $M_2$ , 再存放到  $M_1$  的功能:

如采用其它同类的单片机则至少需三条指令:

```
LD   M1;
ADD  M2; } 6 字节
STD  M1;
```

如采用 Z8 系列单片机只要一条指令:

```
ADD M1 M2; 3 字节
```

如采用 Z8 提供的寄存器寻址指令, 则该指令字节可短到 2 字节。

2. 存储器空间: Z8611 的 CPU 是与四个存储器空间打交道的, 即:

- 60 KB 的外部数据存储器; • 60 KB 的外部程序存储器; • 4 KB 的内部程序存储器(ROM); • 144B (NMOS)或 256B(CMOS)的寄存器阵列(RAM)均可作累加器;

8051CPU 亦和四个存储器空间打交道:

- 64KB 外部数据存储器; • 60KB 外部程序存储器; • 4KB 内部程序存储器; • 148B 寄存器(RAM)。

MC6801CPU 只与 3 个存储器空间打交道:

- 62KB 外部存储器; • 2KB 内部程序存储器; • 149B 寄存器(RAM)。

因而, MC6801 内部 ROM 比 Z8611 和 8051 均少 2KB。所有三个芯片均用 RAM 作为寄存器使用, 这些寄存器阵列均被分成二个部分, 即通用寄存器阵列和特殊用途寄存器阵列(SFR)。

对 Z8611 讲, 124 个通用寄存器可分成 8 组, 每组 16 个寄存器。第一组的低 4 个寄存器与 4 个 I/O 端口成一一对应的映射关系, 故而对这些寄存器的存、取就是对相应的 I/O 端口进行输入、输出操作, 故而 Z8 系列不需专门的输入、输出指令。其中通用寄存器可以通过八位来寻址, 或一种短的四位寻址(寄存器指针寻址)方式; 采用四位寻址方式可节省指令的字节与缩短指令执行的时间。这些通用寄存器均可当作累加器、地址指针, 变址寄存器来使用。

8051 的 128 个通用寄存器分成二组, 低 32 个寄存器分成四组, 每组 8 个寄存器, 而其余的寄存器可被用作堆栈或通用目的寄存器。这些寄存器均不能当作累加器、变址寄存器或地址指针。

MC6801 亦有 128 个通用寄存器, 它们可以当作堆栈或地址指针, 但均不可作累加器和变址寄存器。

从表 1 和上述比较可得到如下结果, 只有 Z8611 的通用寄存器可作为变址寄存器; 只有 Z8611 的通用寄存器可作为非常有用的存储器的指针; 亦只有 Z8611 的通用寄存器可作为多累加器使用。

3. 特殊用途寄存器阵列 Z8611 有 20 个特殊用途的寄存器用作状态、控制和 I/O 端口, 它们是:

- 二个八位寄存器当作一个十六位的堆栈指针(SPH, SPL);
- 一个寄存器定义为寄存器指针(RP);
- 一个寄存器定义为状态标志(Flags);
- 一个寄存器定义为中断优先权(IPR);
- 一个寄存器定义为中断屏蔽(IMR);
- 一个寄存器定义为中断请求(IRQ);
- 三个模式控制寄存器控制四个端口(P0M, P1M, P2M, P3M);
- 一个寄存器定义为串行通信口(SIO);
- 二个计数器/定时器寄存器(T0, T1);
- 一个定时器模式控制寄存器(TMR);
- 二个预置寄存器(PRE0, PRE1);
- 四个寄存器作为 I/O 端口(PORT0, PORT1, PORT2, PORT3)。

8051 同样有 20 特殊用途寄存器作为状态、控制和 I/O:

- 一个堆栈指针(SP);
- 二个累加器(A, B);
- 一个作程序状态字(PSW);
- 二个寄存器当作数据存储器指针(DPH, DPL);
- 四个寄存器引成二个十六位的定时器/计数器(TH0, TH1, TL0, TL1);
- 一个定时器/计数器模式控制寄存器(TCON);
- 一个中断允许寄存器(IEC);
- 一个中断优先权寄存器(IPC);
- 一个串行通信缓冲寄存器(SBUF);
- 一个串行通信控制寄存器(SCON);
- 四个寄存器作 I/O 端口(P0, P1, P2, P3)。

MC6801 有 21 个特殊用途寄存器作状态、控制和



I/O:

- 一个 RAM/EROM 控制寄存器; • 一个串行接收寄存器; • 一个串行发送寄存器; • 一个串行控制、状态寄存器; • 一个串行速率、模式寄存器; • 一个端口 3 状态和控制寄存器; • 一个定时器的状态和控制寄存器; • 二个寄存器组成一个 16 位的定时器; • 二个寄存器与定时器配合作为 16 位的输入; • 二个寄存器与定时器配合作为 16 位的输出比较; • 四个 I/O 端口; • 四个寄存器辅助四个端口作为数据方向寄存器。

通过上述比较可知,MC6801 需要 5 个寄存器来完成对 I/O 端口的控制,而 Z8611 完成相同功能只需 3 个寄存器;MC6801 要 4 个寄存器完成串行通信功能,但 Z8611 只需 1 个就可以了。

8051 只有二个累加器,且要用掉二个寄存器,而 Z8611 所有的通用寄存器均可作累加器。8051 也需 2 个寄存器作为串行通信控制,而 Z8611 只需 1 个寄存器;8051 又要用 2 个寄存器当作数据指针,而 Z8611 任何一个寄存均可作为数据指针。

因而 Z8611 的特殊用途寄存器阵列比 MC6801 和 8051 的利用率要有效得多,这种寄存器阵列的典型设计方法需要强的中断能力和短寻址的寄存器指针,同时 Z8611 可用外部的 60KB 的外存来做外部堆栈,这些都是 MC6801 和 8051 所不能达到的。

4. 外部存储器 三种 MCU 都能随机存取外部存储器。Z8611 和 8051 有一选择信号可以随机地选择外部数据存储器和外部程序存储器,故而 Z8611 可使用 60KB 的外部数据存储器和 60KB 的外部程序存储器。8051 可使用 64KB 的外部数据存储器和 60KB 的外部程序存储器,但 MC6801 只能使用 62KB 的外部存储器。其中包含数据和程序,所以以存取外部存储器的功能来比较,Z8611 和 8051 明显强过 MC6801。

5. 在片外设功能 除 CPU 和存储器空间外,三个单片机均提供了中断系统,I/O 口,并行 I/O 端口,双向的地址/数据总线和一个 I/O 扩展的串行端口。

6. 中断 Z8611 中断源有 8 个,4 个外部中断源来自端口 3 的低四位 IR<sub>0</sub>~IR<sub>3</sub>,4 个内部中断源是串行输入,串行输出,2 个计数器/定时器。所有的中断都是可屏蔽的,通过中断屏蔽寄存器和中断优先权寄存器,这些中断可有 48 个优先权,所有中断都是矢量中断,在芯片 ROM 的最低 12 字节就是六个 16 位的矢量中断地址。因而外存的整个范围均可存放中断服务子程序。

8051 有 5 个中断源,2 个外部中断源是 INT0 和 INT1,3 个内部中断源是 2 个定时器/计数器,1 个串行 I/O 口,所有的中断均可独立地禁止或全部禁止。任一个均可有高或低的优先权状态,5 个中断均是矢量中断,在存储器中每个矢量中断都有一个 8 字节长的存储器存放中断服务。

MC6801 有一个外部中断,一个非屏蔽中断,一个内部查询中断和一个软件中断。串行 I/O 口,定时器溢出,定时器输出比较,定时器输入均是内部中断,每个中断的优先权都是预置的且不能更换,外部中断可以通过状态码寄存器进行屏蔽,MC6801 在存储器中

有 7 个固定地址的矢量,它们是 16 位的服务子程序的入口地址。

当一个中断产生时,8051 只有程序指针 PC 被保护,用户必须保护标志、累加器和其中一些寄存器内容,MC6801 会保护程序指针 PC、累加器、变址寄存器和程序状态字,用户必须保护所有中断服务子程序所需的寄存器内容,而 Z8611 保存程序指针 PC 和标志寄存器,为保护 16 个寄存器内容,仅仅寄存器指针 RP 的内容需进入堆栈,而另一组工作寄存器用来作中断子程序服务。可见 Z8611 的中断明显地是最强的,它只需一个命令就能保护所有的工作寄存器,大大提高了程序间的转换效率。

7. I/O Z8611 有 32 条引线具有 I/O 的功能,这 32 条引线组成四个具有八个端口,四个端口均可通过软件可以设置成输入、输出、多重地址/数据线,定时或状态。输入或输出可以是串行、并行、带交互式或不带交互式。一个端口可以被设置成串行传输,四个端口均可被设置成并行传输,在并行传输时,端口 3 提供交互控制信号,其余三个端口传输数据。

8051 同样具有 32 个口被组成四个端口,每个端口有 8 个口,程序可将口设置成串行或并行的 I/O,亦可设置成地址/数据线、定时、状态,通过软件可以完成交互控制 I/O。

MC6801 有 29 个端口为 I/O,三个八位的端口和一个五位的端口,一个端口有二条线作为交互控制,端口提供输入输出、串行、并行、地址、数据等所需的各种控制信号,它们亦可用作与外部存储器的接口。

三种芯片在 I/O 方面最主要的区别是 Z8611 可以用硬件来完成三个交互式控制的端口,而 MC6801 只有一个端口可用硬件完成交互式控制,8051 无硬件交互式控制。

8. 计数器/定时器 Z8611 有二个 8 位计数器,2 个 6 位可编程的预置器,一个预置器可被内部或外部时钟驱动,而另一个只能被内部时钟驱动,二个计数器完成计数后均可产生中断信号中断 CPU,二个计数器均可工作于 2 个状态之一,其一是单项模式,减法计数到零产生中断;其二是重复模式,减法计数到零再次从预置数开始减法计数。这二个计数器可用来测量时间、脉冲宽度、产生可变脉冲宽度、计数转换,或产生周期性的中断。

8051 有二个 16 位的计数器/定时器,可测量时间、脉冲宽度、产生脉冲宽度、计数器转换与周期中断,它们有几种工作模式;它们可设置成带二个 5 位预置的 8 位计数器;它们可以设置成二个 16 位的计数器/定时器;最后它们可以作模 8 的计数器,将结果值存放到 16 位寄存器的高八位,当计数器/定时完成计数,将产生中断信号。

MC6801 有一个 16 位的计数器用于脉宽测量和产生。这个计数器/定时器实际上是三个 16 位的寄存器和一个 8 位的状态控制寄存器,该定时器有一个输入寄存器,一个输出比较寄存器和一个独立工作的计数器,三个 16 位的寄存器的均可产生中断。

(待续)



# 如何用 C 语言编制数据采集软件

无锡轻工业学院自动化系(214036) 冯品如

无锡小天鹅集团洗衣机公司 张新民

本文针对洗衣机电参数计算机辅助测试流水线的研制,在软件上采用 Turbo C 进行编程,对在汉字系统下如何用 C 采集、处理数据,显示、打印汉字数据结果,以及如何按 dBASE 数据格式或二进制数据格式保存结果等等进行了一些讨论。

用 C 语言编制洗衣机电气参数流水检测线数据采集程序时,遇到了不少问题,如何解决这些问题这里谈一些体会。

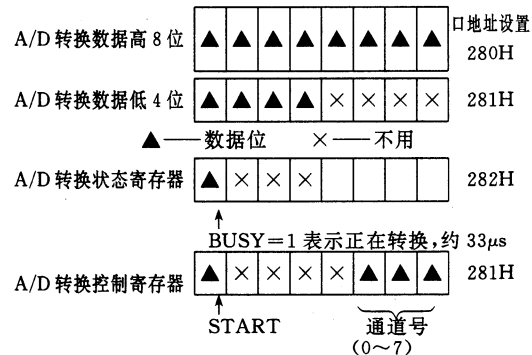
## 一、数据的采集:

数据的采集用 C 语言中的:inportb()

数据的输出用:output()

通过 x=inportb()得到经 A/D 板转换后的八位二进制值及开关量状态。对于 12 位的 AD 转换器,结果一般分两个字节保存,要进行合并计算。如下例:

ADC-30 转换及数据寄存器的设定如下:



```
int zadc(int i)
{
    int data.h,data.l,busy,temp;
    int busy=1;
    output(0x281,0x80+i); /* 启动 i 通道转换 */
    while(busy){ /* 等待转换完毕 */
        busy.temp=inportb(0x282); /* 取状态转换字 */
        busy=busy.temp&0x80;
    }
    data.h=inportb(0x280); /* 读高八位数据 */
    data.l=inportb(0x281); /* 读低四位数据 */
    return((int)(data.h*16+data.l/16)); /* 返回测量结果 */
}
```

## 二、数据的处理:

从 A/D 口采集得到的数据经转换成十进制整数

后,再进入数据处理部分。

对于流水线,由于在同一时间不同的工位测得的参数分属不同的被测体,故需根据测试位置和顺序对结果进行定位分配。这通过数组很容易实现,这里不多谈。

对数据进行定位转换、滤波和工程量转换,从而可得到流水线上每台被测体的测试结果和流水线每一对应工位上的参数值。

## 三、数据的保存和读取:

经 A/D 转换采集,并转换成十进制数后的结果数据,保存可用二进制形式或 ASCII 码形式进行,而 dBASE 即以 ASCII 码形式保存。

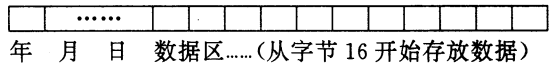
### ①以二进制方式:

特点:便于统计,节省空间。

用二进制保存时,一个字节的 00~FF 即表示 0~255,你可规定你的结果文件格式。

如:规定文件的前 16 个字节为文件的有关信息,后面开始存放数据。

字节 0~15 16



在 C 语言中,读写既可用标准 I/O 进行,也可用系统级的读写指令进行。要进行读写,必须掌握对文件部分字节或一个字节的读写方法。

写:

```
ch[0]=day.da.year;
ch[1]=day.da.mon;
ch[2]=day.da.day;
lseek(temp,0L,0);
write(temp,ch,16);
```

读出:

```
lseek(temp,0L,0);
read(temp,ch,16); /* ch 定义为字符数组 */
读写数组中的数据是这样进行的:
```

写:

```
lseek(temp,16L,0);
write(temp,&data,sizeof(data));
```

读出:

```
read(temp,&data,sizedata);
```

### ②以 dBASE 格式保存:

特点:通用、便于查看,但占空间。

采用 dBASE 方式存盘,在系统工作程序上可利用 C 语言编制相应的功能查看,亦可在普通的 PC 系

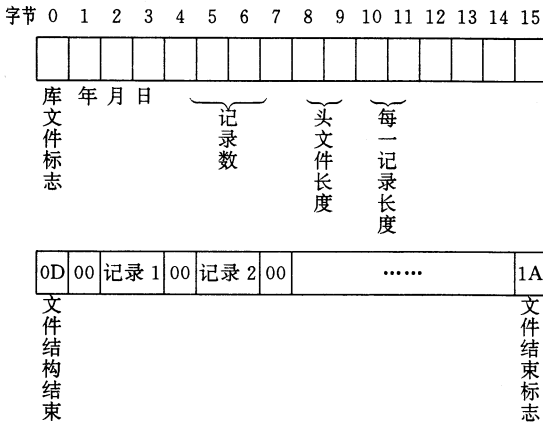


列机上用 dBASE 或 FoxBASE 直接查看处理。

为此先说明 dBASE 数据库文件的格式。(dBASE 和 FoxBASE 文件格式相同)

dBASE 库文件格式:

dBASE 库文件内部是以 ASCII 码形式存放数据的,库文件由两部分组成,第一部分是数据库结构描述部分,以 0D 00 结尾;第二部分是记录数据,紧接着 0D 00 存放,记录之间用空字符隔开。库文件以 1A 标志结束。



读 dBASE 的信息部分,可这样进行:

```
lseek(fr,0L,0);
read(fr,ch,16);
recyear=ch[1];
recmonth=ch[2];
recday=ch[3];
recnumb=ch[4]+256*ch[5];
reclen.h=ch[8]+256*ch[9];
reclen=ch[10]+256*ch[11];
```

由于 dBASE 的数据文件是以 ASCII 码形式保存,所以先要把数组中的数据按记录的要求转换成字符串,字符串的第一为空格,接着为数据最后加一库结束标记 1A,在下次添加记录时,再将 1A 覆盖,重复上一步,这样就将数据保存了。但注意,仅存记录是不够的,还要修改库结构描述部分的记录数。

将数据转换成字符串,可用 sprintf 进行:

```
sprintf(dataline," \ x20%3d%6.4f \ x1a", no, name,r);
```

将字符串存入文件:

```
length=reclen.h+reclen*recnumb;
lseek(fr,length,SEEK.SET);
write(fr,&dataline,sizeof(dataline));
```

重新修改日期、记录数。

```
recnumb=recnumb+1;
ch[4]=recnumb%256;
ch[5]=recnumb/256;
getdate(&day);
ch[1]=day.da.year-1900; /* 改日期 */
ch[2]=day.da.mon;
```

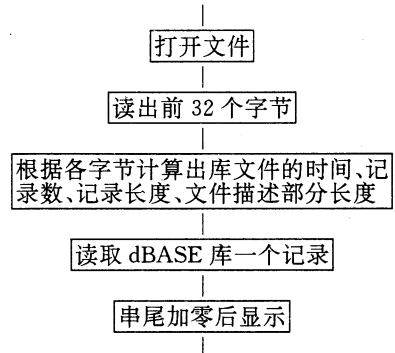
```
ch[3]=day.da.day;
lseek(fr,0L,0); /* 存结构描述部分 */
write(fr,ch,16);
```

经这样处理的文件可直接由 dBASE 处理。

③dBASE 文件的读出显示:

dBASE 库文件是以 ASCII 码保存,其记录格式可用 C 语言的结构模拟,串尾加零后即可显示。

读 dBASE 文件头文件模块框图:



#### 四、数据的显示

数据结果的显示离不开汉字提示,而目前软汉字系统一般以图形方式显示汉字,进入汉字系统即进入图形显示方式。

Turbo C 虽然编辑编译时可以接受汉字字符串,但只有进入图形方式,并且选择了正确的显示模式后,在汉字系统下 printf(...)才能显示汉字,而且只能显示单一的黑底白字,其余输出语句如 cprintf、out-text 等均不能正确显示汉字。目前已有的汉化 Turbo C 一种是长城机用的版本,一种是 PC 机用的版本,长城机版本不适用于 PC 系列机,而 PC 机版本经测试发现仅仅集成编辑环境可使用汉字,其输出结果的显示语句功能并没有改变,最后效果仍同西文的 Turbo C 一样。正确处理好 Turbo C 图形模式和汉字系统的关系,创造出一个好的用户界面,这对于面向用户的软件来说是非常重要的。

对于带汉卡的机器,由于汉字是以字符方式显示,用西文 Turbo C 的文本或图形输出函数可以显示出正确的汉字。所以长城机、PC5550 等(内带字库的)机不经处理直接即可显示汉字。而不带汉卡的机器有更高的灵活性,下面谈谈占有绝大多数的带软汉字系统的机器显示汉字的方法。

实现汉字显示两种较常用的方法:

①调用 BIOS 中断 10H 实现:

调用 BIOS 中断 10H,对需显示的字符串(包括汉字)逐个读取显示,可以控制显示的颜色和背景色,这种方法要求系统工作在中文方式下,此时可以直接利用 Turbo C 的图形函数作图。程序如下:

```
union REGS regs;
int cprintfs(char * string,int row,int col,int color,int orxor)
{
```



```

int j, ncol;
for (j = 0, ncol = col; j < strlen (string); ++j, ++ncol)
    print.char (string[j], row, ncol, color, orxor);
}
int print.char (char ch, int row, int col, int color, int orxor)
{
regs. h. ah = 2;
regs. h. bh = 0;
regs. h. dh = row;
regs. h. dl = col;
int86 (0x10, &regs, &regs);
regs. h. ah = 9;
regs. x. cx = 1;
regs. h. al = ch;
regs. h. bl = color | orxor;
int86 (0x10, &regs, &regs);
}

```

②直接读取字库字模,用绘点方法实现:

通过直接读取 16 点阵字库字模,再根据字模的排列规则用绘点方法实现汉字的显示。该方法选用不同的点阵字库将显示不同的字体,如可选用 16 点阵简体、繁体或 24 点阵宋、楷、黑、仿宋体,将显示出对应的 16 种颜色的字体,经处理后还能放大、变形,同时不论在中西文方式下都能显示汉字。下面是利用 213H 字库显示四种字体的程序。

```

int hz24 (char * p, int x, int y, int type, char color, int h, int w)
{
FILE * fphz24;
long length;
unsigned char c, by[72];
int i1, i2, i3, i4, i5, cl, flag = 0;
switch (type) {
case 0: /* 宋体 */
fphz24 = fopen ("c:\\213\\hzk24s", "rb");
break;
case 1: /* 仿宋体 */
fphz24 = fopen ("c:\\213\\hzk24f", "rb");
break;
case 2: /* 楷体 */
fphz24 = fopen ("c:\\213\\hzk24k", "rb");
break;
case 3: /* 黑体 */
fphz24 = fopen ("c:\\213\\hzk24h", "rb");
break;
default: break;
}
while ((c = * p++) != 0)
{

```

```

if (c > 0xa0)
if (flag == 0)
{c1 = c;
flag++;}
else
{
flag = 0;
length = (c1 - 0xa0) * 94 + c - 1665L;
length = 72L * length;
fseek (fphz24, length, SEEK.SET);
fread (by, 1, 72, fphz24);
for (i1 = 0; i1 < 24; i1++)
for (i2 = 0; i2 < 3; i2++)
for (i3 = 0; i3 < 8; i3++)
if ((by[i1 * 3 + i2]) >> (7 - i3)) & 1)
for (i4 = 0; i4 < h; i4++) {
for (i5 = 0; i5 < w; i5++)
putpixel (x + i4 + h * i1, y + i5 + w * i2 * 8 + w *
i3, color);}
}
x += 12 * h;
}
fclose (fphz24);
}

```

以上两种方法均可实现汉字和 Turbo C 图形的统一,为编制良好的中文用户界面提供了可能。

## 五、数据的打印:

结果输出中如何打印出美观的各种字型汉字? 经实践本人很好地解决了这个问题。

对于西文 Turbo C,打印是通过向规定的标准输出设备写串实现的,即

```
fprintf (stdprn, fmt...);
```

①如系统是在汉字系统下工作,调入汉字打印驱动程序,可打印出汉字。

②如系统是在纯西文系统下工作,单独调入 2.13H 汉字系统打印驱动程序,亦可打印出各种汉字。这时,在中文系统下编辑好的汉字尽管不能正确显示,但仍能打印出各种字体的汉字,中文下的各种打印控制符仍起作用。如:

```
fprintf (std, "BM 打印\n");
```

```
2.13 打印驱动部分:
```

```

CD\213
PRTA
FILE16B
FILE24A 1SFHK
FILE40A 1SFHK
ZF24 3
CD\

```

2.13H 的打印功能是很强的,它可以打印出多种不同大小不同字型的字体和背景,而且打印控制符使用简单方便。但汉字系统的调入,将会对某些用于控

制的系统的运行产生一些影响。

如再利用前面介绍的西文显示汉字的方法,可以使您的程序工作在纯西文方式下,而仍能实现汉字提示的显示和汉字打印输出。对于某些程序来说,这样能更进一步提高程序运行的可靠性。

另外,打印时经常会出现的打印机故障或误操作,如电源未开、无纸等,将会使计算机停机等待纠正,影响系统的正常运行,这对于一个控制系统是不允许的。本系统通过在打印一行前先检测打印机状态,以决定是否打印的方法克服了这个问题,如打印机未准备好,则跳过。本人使用此法编制的程序,即使打印中途无纸或按某些按钮,都不影响系统的正常运行。如:

```
prtstate = biosprint(2,0,0); /* 测打印机状态 */
if(prtstate=0x90){ /* Ready=1,打印 */
fprintf(stdprn,"编号:%10ld\n",no);}
```

#### 六、系统的抗干扰问题:

工业控制系统的可靠性是一个很重要的问题,由于外界的干扰可能会使系统死机,使数据丢失。死机的主要原因是程序进入了某个死循环,这时数据仍然在内存中,如果能中止死循环,同时使程序自动继续执行,系统就能连续运行。

Turbo C 有一设置 Ctrl.Break 处理程序的函数,利用它来打断死循环,再利用另一函数 longjmp 返回到程序的主循环开始处,从而实现系统工作的恢复。如下例:不论何时按 Ctrl.C 或 Ctrl-BREAK,程序都将从 start 开始。

```
#include<stdio.h>
#include<setjmp.h>
#include<dos.h>
int value=0;
jmp.buf jumper;
int c.break(void)
{
printf("Control-Break hit. Program aborting...\n");
longjmp(jumper,1);
}
main()
{
value=setjmp(jumper);
printf("start\n");
ctrlbrk(c.break);
for(;;){ printf("Looping...value=%d\n",value);}
```

## 谈谈 MS-DOS 5.0 新增命令 DOSKEY

湖北省远安县财政局 (444200) 唐银红

MS-DOS 5.0 是美国微软(MICROSOFT)公司于一九九一年底推出的最新一代个人电脑的磁盘操作系统,最近我国的长城、联想等集团公司向美国微软公司购买了 MS-DOS 5.0 的使用许可,购买数达七万件之多,加上外国进口的原装机及台湾生产的机器大都配有 MS-DOS 5.0,可以预测,MS-DOS 5.0 将取代目前在国内最为流行的 MS-DOS 3.30 从而成为国内使用最普遍的操作系统。

MS-DOS 5.0 较以前版本的 DOS 相比较有了明显的改善,诸如在内存管理、文件操作和磁盘维护、数据安全、在线帮助等方面,还有本文即将介绍的新增命令 DOSKEY。

DOSKEY 是 MS-DOS 5.0 的一个外部命令,可用于网络服务器中,它是一个驻留内存(TSR)程序,使用缺省参数驻留内存后,其自身大约占用 3KB 左右的存储空间,加上 512 字节的命令缓冲区,约需 4KB 左右

的存储空间。

实际上,DOSKEY 是键盘输入与 MS-DOS 5.0 键盘接收信息解释程序的一个高级接口,它是在内存中开辟一定数量的存储器区域作为缓冲区,用于记录自 DOSKEY 驻留内存后用户所键入的所有命令(包括用户调用的各种应用软件),用户可以用 DOSKEY 定义的编辑键将其中的任何一条调出来,直接重发或进行编辑修改后重发。此外在 DOSKEY 的支持下,用户可以在一个命令行中一次输入多条 DOS 命令,或者创建一个命令宏定义(简称为宏),所谓宏,就是一次定义多条命令到宏体中,要执行这些命令,键入宏名就可以了。

#### 一、DOSKEY 命令的格式

命令格式: DOSKEY [/REINSTALL][/BUFSIZE=size][/MACROS][/HISTORY][/INSERT|OVERSTRIKE][MACRONAME=[TEXT]]



(其中;方括号中的内容表示是可选项,管道符号表示两者中只能取一种)。

参数和开关:

MACRONAME 为创建的宏指定名称。

TEXT 宏中记录的 DOS 命令。

/REINSTALL 安装 DOSKEY 的拷贝到内存中,而不管内存中是否已经安装过 DOSKEY。如果已经安装过 DOSKEY,该开关将清除 DOSKEY 缓冲区中的命令。

/BUFSIZE=size 设置 DOSKEY 用于保存命令和宏的缓冲区的大小,以字节为单位,缺省设置是 512 字节,最小值是 256 字节。

/MACROS 显示缓冲区中所有的 DOSKEY 宏,可缩写为/M。

/HISTORY 显示缓冲区中记录的用户键入的所有 DOS 命令,本开关可缩写为/H。

/INSERT/OVERSTRIKE 指定修改 DOSKEY 命令行时的编辑方式。/INSERT 为插入方式,使用此开关时,将编辑方式设置为插入方式,在编辑时,就象按下了 INS 键一样,使键入的字符插入到原文的两个字符之间。/OVERSTRIKE 设置覆盖编辑方式,键入的字符替换掉原文光标处的字符,这也是 DOSKEY 的缺省设置。

以上两种编辑方式中,小键盘上的 INS 键仍然是这两种编辑方式的转换开关,按一次就从一种编辑方式转换到另一种编辑方式,直到按下回车键转换到缺省的设置。

DOSKEY 中将两种编辑方式的光标形状设计得不同,覆盖编辑方式时,光标与 DOS 光标一样;插入编辑方式时,光标高度加增一倍。

在 DOSKEY 管理期间,可能出现这种情况:定义的缓冲区不是太大,而用户使用了较多的 DOS 命令,使得缓冲区容纳不下。在这种情况下,DOSKEY 按照队列(先进先出)的管理方式,将最先使用的 DOS 命令从缓冲区中清除。

## 二、DOSKEY 命令使用的编辑键

为了能再次调用或编辑 DOSKEY 缓冲区中保存的 DOS 命令,DOSKEY 程序定义了一些编辑键,其中有些与 DOS 命令中的编辑键相类似,有些是重新定义的。以下是 DOSKEY 中使用的编辑键及其功能的主要描述:

UP ARROW(↑) 调出 DOSKEY 缓冲区中最近使用的一条命令,每按一次,就调出前一条命令。

DOWN ARROW(↓) 调出正在显示的命令的下一条命令,每按一次,就向后推一条。

PAGE UP(PgUp)调出缓冲区中最早一条命令。

PAGE DOWN(PgDn) 调出缓冲区中最后一条命令。

F7 显示所有存储在 DOSKEY 缓冲区中的命令及其在缓冲区中的顺序号,每行显示一条,前面是顺序号,后面是命令名,顺序号和命令之间用冒号分隔。顺序号是根据命令的执行顺序(即进入 DOSKEY 缓

冲区的顺序)而指定的,最先执行(进入缓冲区)的就是 1 号。

ALT+F7 清除保存在 DOSKEY 缓冲区中的所有命令。

F8 在 DOSKEY 缓冲区中搜索以指定字符或字符串开头的命令,要使用这个编辑键,首先在 DOS 提示符后输入要 DOSKEY 搜索的命令的第一个字符或前几个字符,不区分大小写,但在 DOS 提示符与指定字符之间不能有任何空格,然后按 F8 键,DOSKEY 就显示最近使用的一条与之相匹配的命令,再按一次 F8 键,DOSKEY 继续向前搜索与之相匹配的命令,直到搜索完整个缓冲区。

F9 根据顺序号调出命令。当用户按下 F9 键后,DOSKEY 就要求输入命令在缓冲区中的顺序号(line number)用户输入顺序号后,DOSKEY 就调出缓冲区中相应的命令,显示在当前行上。要想知道各个顺序号所表示的命令名,可按 F7 键列出所有命令及其顺序号。

ALT+F10 清除 DOSKEY 缓冲区中已定义的宏。如果要删除某个已定义的宏,用带 MACRONAME=参数的 DOSKEY 命令。

ARROW LEFT(←)光标向后(左)移动一个字符。

ARROW RIGHT(→)光标向前(右)移动一个字符。

CTRL+ARROW LEFT(←)光标移到前一个字段的第一个字符上。所谓字段,就是用空格分隔的字符串。CTRL+ARROW RIGHT(→) 光标移到下一个字段的第一个字符上。

HOME 光标移到当前编辑的 DOSKEY 命令的第一个字符上(行首)

END 光标移到当前编辑的 DOSKEY 命令的最后一个字符后(行尾)

CTRL+HOME 删除光标左面的所有字符,但不包括光标处的那个字符。

CTRL+END 删除光标右面的所有字符,包括光标处的那个字符。

INS 插入编辑方式和覆盖编辑方式的转换开关。

DEL 删除光标处的字符。

BACKSPACE 删除光标前的一个字符。

ESC 清除显示的命令,并将光标移到行首(DOS 提示符之后)。此键与 DOS 命令中的编辑键的定义稍有不同,它是清除当前行上显示的所有字符,并将光标移到行首,而在 DOS 中,它表示使当前行作废,另起一行以重新输入命令,作废的行依然显示在显示器上,只不过在行尾加上一个反斜杠表示该行作废。

## 三、重新调用 DOSKEY 缓冲区中的命令

用以下三种方法,可以调出、编辑和重发保存在 DOSKEY 缓冲区中的命令。

①如果缓冲区中的命令不是很多,可用 UP ARROW(↑)和 DOWN ARROW(↓)依次向前或向后调出各条命令。

②紧接 DOS 提示符后输入命令的第一个字母或前几个字符,然后按 F8 搜索缓冲区中最近使用的一条以指定字符或字符串开头的命令,找到后就将其显示在当前行上,如果不是所需要的命令,可按 F8 键继续向前搜索。

③先按 F7 键,将 DOSKEY 缓冲区中的所有命令在屏幕上显示出来,观察所要调用的命令的序号,然后按 F9 键,此时, DOSKEY 就要求输入命令的序号,键入序号,就可以调出相应的命令。

例如,某用户在 DOSKEY 驻留内存后,已经使用了以下四条命令(可按 F7 键观察):

```
213          ;调用 2,13 汉字系统
CD/CRPG      ;进入报表处理子目录
CRPG         ;调用汉字报表处理软件
COPY *.DBF A;将修改后的数据文件备份
```

如果按 ↑ 键,最后一条命令 COPY \*.DBF A:就显示在当前行上,继续按 ↑,就显示第三条命令 CRPG,如果此时按 ↓ 键,第四条命令又出现在显示器上,按 PgUp,显示第一条命令 213,按 PgDn 键显示最后一条命令。

如果要调用第三条命令,可以连按 ↑ 键两次,或键入 C(可不分大小写),然后按两次 F8 键(因第一次按 F8 键搜索的是最近使用的一条以 C 开头的命令 COPY),还可以按 F9 键,再键入 CRPG 命令的缓冲区中的编号 3。

无论用哪种方式调出缓冲区中的命令,都可以用前面介绍的编辑键对命令进行任意编辑。编辑时,整个命令行都显示在显示器上,编辑非常方便。无论光标在什么位置,只要按下了回车键,命令就被重发了。

④一行输入多条命令。在 DOSKEY 的支持下,可以在一行中输入多条命令,各个命令之间用 CTRL+T 分隔,每行的字符总数,包括空格在内,不能超过 127 个字符。

例如,象下面一样将上述四条命令放在一行之内:

```
213^ TCD/CRPG^ TCRPG^ TCOPY *.DBF A;
```

在敲入回车后四条命令就被依次执行。

⑤利用 DOSKEY 创建批程序。在 DOSKEY 中利用 /H 开关和输出重定向,能方便地创建一个批程序。例如要在 C 盘根目录下建立一个名为 CRPG.BAT 的批程序,包含以上四条命令,可按以下步骤进行:首先按 ALT+F7 键清除 DOSKEY 缓冲区中的所有命令,然后依次调用以上四条命令,全部执行完后,可按 F7 键察看,确定缓冲区中的命令是否全部都是批程序中所要求的,最后键入以下命令:

```
DOSKEY /H>C:\CRPG.BAT
```

#### 四、DOSKEY 宏

前面,我们已经多次提到过 DOSKEY 宏,下面就详细地介绍 DOSKEY 宏。

①DOSKEY 宏中的专用字符。

以下的字符,在 DOSKEY 宏定义中具有特殊的意义:

\$G 或 \$g 这个符号等价于 DOS 中的输出重定向符号(>),是将输出送到文件或设备中。

\$G \$G 或 \$g \$g 这个双写的符号等价于 DOS 中的“添加”输出重定向符号(>>),是将输出添加到已经存在的文件的末尾。

\$L 或 \$l 该符号等价于 DOS 中的输入重定向符号(<),用它指定从设备或文件中读取输入。

\$B 或 \$b 等价于命令行中的管道符号(|),用于将宏输出到一条命令之中。

\$T 或 \$t 如果创建 DOSKEY 宏时的命令不止一条,则各条命令之间用 \$T 或 \$t 分隔。

\$ \$ 输入美元符号(\$)。

\$1 到 \$9 定义宏时使用的可替换参数,调用时在宏名后依次给定相应的参数。这很象批程序中的 %1 到 %9,使宏能完成不同的任务。

定义宏中使用的可替换参数上面的 \$1

到 \$9 只能代表一个可替换参数,而 \$

\* 是将宏名后的所有信息作为一个参数。

#### ②宏的创建、运行和删除

例如,要创建一个名为 QFC 的宏,完成以下任务:对驱动器 A 中的盘进行无条件(/U 开关)、快速(/Q 开关)格式化,然后用 SYS 命令将系统传送到 A 盘中,最后将 C 盘 DOS 目录下的所有扩展名为 .EXE 的文件拷贝到 A 盘中。可用以下命令来创建这个宏:

```
DOSKEY QFC=FORMAT A: /Q/U $TSYS A:
$TCOPY C:\DOS\*.EXE A;
```

每用一条带 MACRONAME = TEXT 参数的 DOSKEY 命令,就创建了一个宏。这些宏被依次存放在 DOSKEY 缓冲区中,直到被删除为止。

要调用已定义的宏,键入宏名即可。例如,在光标处打入:

```
QFC
```

则 QFC 宏被调用,完成对 A 盘的格式化、系统传送和文件拷贝任务。

因为设置的 DOSKEY 缓冲区数量有限,所以应及时删除无用的宏。删除一个宏,就是定义该宏时,仅有宏名,而不赋给宏任何内容,就象下面一样,将 QFC 宏从缓冲区中清除:

```
DOSKEY QFC=
```

这就是说,对已经存在的宏,可以随时重新定义。

如果要删除所有的已经定义的宏,按 ALF+F10 键。

#### ③在宏中使用可替换参数

象前面定义的 QFC 宏中,有时感觉使用不方便,例如要对 B 盘做相同的工作,就不能用 QFC 宏,还有一种可能,如果 A 为 1.2MB 的驱动器,要格式化低密盘时,必须带 /4 开关,如果在定义宏时使用了可替换参数 \$1 到 \$9,就可以完成诸如上述的操作,使用时就感觉到非常方便。就象下面一样:



```
DOSKEY QF=FORMAT $1/U/Q $2 $TSYS
$1$TCOPY C:\DOS\*.EXE $1
```

如果是对 B 盘进行上述操作,键入下面的命令:

```
QF B;
```

如果是格式化 A 驱(1.2MB)中的 360KB 的软盘,并进行相应的操作,使用以下命令格式:

```
QF A: /4
```

有时需要替换的参数不能确定时,用 \$\*。

例如,用列目录命令 DIR 观看 C 盘 DTP 目录下的 SAMPLES 子目录下的扩展名为 .DOC 的文件,分别用不带开关、带/P/W 开关和/P/W/OD/L 开关,则可用下面的命令定义为 DDIR 的宏:

```
DOSKEY DDIR=DIR $* C:\DTP\SAMPLES
\*.DOC
```

在调用时,参数跟在宏名之后:

```
DDIR 相当于 DIR C:\DTP\SAMPLES\*.DOC
```

```
DDIR /P/W 相当于 DIR /P/W C:\DTP\SAM-
PLES\*.DOC
```

```
DDIR /P/W/OD/L 相当于 DIR /P/W/OD/L C:\
DTP\SAMPLES\*.DOC
```

#### ④宏名与 DOS 命令同名

定义宏时,允许将宏名取得与 DOS 命令名相同。以下规则使系统能够识别调用的是 DOSKEY 宏,还是 DOS 命令:

调用宏时,必须在 DOS 提示符(>)之后,立即输入宏名,在宏名与 DOS 提示符之间不能有任何空格。

调用 DOS 命令时,在 DOS 提示符后输入一个或多个空格,然后再输入 DOS 的命令名。

#### ⑤利用批程序保存宏

由于宏是保存在 DOSKEY 缓冲区中,也就是说它位于存储器中,因此,也就具有易失性,一是重新引导系统时,二是用带/REINSTALL 开关的 DOSKEY 命令时,三是按 ALT+F10 时,都将清除所有的宏。如果这些宏还有用,就必须重新定义,这就显得有些麻烦。此时可将缓冲区中的宏,输出到一个批程序中保存起来,必要时利用批程序来定义宏。例如在前面已经定义了 QFC、QF 和 DDIR 三个宏,利用/M 开关和输出重定向,创建一个名为 MACROS.BAT 的批程序:

```
DOSKEY /M>MACROS.BAT
```

此时,MACROS.BAT 文件中包括以下三条语句:

```
QFC=FORMAT A:/Q/U $TSYS A: $TCOPY
C:\DOS\*.EXE
```

```
QF = FORMAT $1/Q/U $2 $TSYS $1
$TCOPY C:\DOS\*.EXE $1
```

```
DDIR=DIR $* C:\DTP\SAMPLES\*.DOC
```

用各种文本编辑软件(象 WS 等)在每条语句前面上加 DOSKEY,当编辑好后,每次调用 MACROS.BAT 批程序时,就定义了 QFC、QF 和 DDIR 三个宏。

注意:不能在一个宏中调用另一个宏,也不能在一个批程序中调用一个宏。

## 五、宏与批程序的区别

宏和批程序很象似,当输入其名称时,就调用了它们之中所包含的命令集合,但它们之间还是有很大的区别:

①宏保存在内存中,可以从任何盘、任何目录中调用,且速度相当快;而批程序保存在磁盘上,调用时必须指明它存放的路径,否则,DOS 就找不到它,并且批程序总是从磁盘中调到内存,因而执行的速度慢于宏。

②在定义宏时,将所有的命令放于一行之中,各命令之间用 \$T 分隔,而且字符总和不能超过 127 个;而在批程序中,一条命令单独占用一行,且对命令的个数没有限制。

③宏和批程序都对 CTRL+C (CTRL+BREAK) 非常敏感。在宏中,按 CTRL+C 只能中断正在执行的那条命令,而且是不等用户确认,转而去执行下一条命令。如果一个宏中定义了三条命令,就得按三次 CTRL+C 才能完全中断这个宏,如果按 CTRL+C 的次数少于宏中定义的命令个数,那么,下个余留的命令将被立即执行;而在批程序中,只要用户按下了 CTRL+C,批程序就会停下来,询问用户是否确实要退出批程序,如果用户确认,就停止整个批程序,而不管还余下多少个命令还没有被执行。

④虽然两者都能使用可替换参数和重定向符号,但二者的表示方式不同。宏中的可替换参数是 \$1——\$9,并使用专用重定向符号 \$G、\$G\$G 和 \$L;而在批程序中的可替换参数是 %1——%9,并使用 DOS 中的重定向符号 >、>> 和 <。

⑤宏中不能使用 GOTO 命令,也就是说,宏只能完成一些较为简单的任务,而要完成比较复杂的任务,还必须用批程序。

⑥可以在宏中定义环境变量,但不能在宏中使用环境变量。例如设定 ABC 为 C:\DOS 目录 (SET ABC=C:\DOS),但不能在宏中用 ABC 代替 C:\DOS,象 DIR ABC 在宏中是错误的;然而,在批程序中使用所有的环境变量。

#### 备注:

①在 DOSKEY 中,只能将用户从键盘中输入的信息保存到缓冲区中,如果用户调用一个批程序,DOS 将顺序地执行批程序中的每条命令,而不将这些命令保存到缓冲区中。

②DOSKEY 命令行与 DOS 命令行的区别: DOS 的命令行只能显示光标以前的字符,而 DOSKEY 是将整条命令都显示在当前行上,这对编辑修改非常方便。

③本文所说的命令,是广义的 DOS 命令,意即所有扩展名为 .COM、.EXE 和 .BAT 的可执行文件。

④如果 DOSKEY 用缺省的参数驻留内存,则缓冲区为 512 字节,大约可以记录 35 条命令。

⑤当用户键入 CTRL+T 时,在屏幕上显示的是一个特殊符号,本文中用 ^T 表示。

# 用 SET 命令解决 COMMAND.COM 的重新调入

云南省科学技术协会信息中心(650031) 叶 斌

COMMAND.COM 是 DOS 的命令处理器,它是负责执行用户命令的程序。命令处理器由三部分组成:启动部分、常驻部分、暂存部分。

启动部分的作用仅在于执行自动批处理文件 AUTOEXEC. BAT,当 AUTOEXEC. BAT 执行结束后,启动部分终止运行,释放所占的内存空间。

常驻部分则是始终保留在主存中的。该部分实现 DOS 必须能立即响应的所有必不可少的基本功能,例如常驻部分的调入、程序执行或者程序被用户通过 CTRL-BREAK 中止。另外,常驻部分还执行 DOS 关键性错误处理,该处理产生如下信息:

Not ready error reading device

Abort, Retry, Ignore?

第三部分是暂存部分。许多 DOS 应用程序会覆盖掉内存中这一部分。这时,常驻部分将重新由磁盘调入常驻部分。

在 DOS 状态下打入 SET 命令,我们可以看见

COMSPEC=<path>COMMAND.COM

一行,<path>是根据启动时 COMMAND.COM 所在的驱动器而定。该行表示在暂存部分被重新调入时应查找哪一个驱动器并调入。

如果是用硬盘启动的机器,SET 命令显示

COMSPEC=C:\COMMAND.COM

表示如果暂存部分需重新调入时,应从 C 盘根目录调入。因硬盘速度较快,所以重新调入时是感觉不到的。但如果你的机器是不带硬盘的机器,或是用软盘启动的,重新调入时盘上不存在 COMMAND.COM,就会

出现

Insert disk with COMMAND.COM in drive A  
and strike any key when ready

的信息,你就要重新插入 DOS 盘然后按任意键。这样处理多次后,就会感觉到很不方便,而我们只要用 DOS 的 SET 命令修改环境变量中的 COMSPEC 就可以解决这一问题。

方法一:如果你的驱动器是 1.2M+360K 的配置,而高密驱动器又用得较多,你可以在 DOS 盘的自动批处理文件中加上

SET COMSPEC=B:\COMMAND.COM

一行,机器启动后就把 DOS 盘放入 B 驱(当然你的 DOS 盘也必须是低密盘),这样每次重新调入 COMMAND.COM 时就会从 B 盘调入,而你也不须频繁地更换 A 驱的盘了。

方法二:建立一个容量稍大于 COMMAND.COM 长度的虚拟盘(有扩展内存就建在扩展内存,建立虚拟盘的方法很多 DOS 的书上都有,这里就不说了),然后在自动批处理里加上

COPY COMMAND.COM C;

SET COMSPEC=C:\COMMAND.COM

两行,这样虽然启动时因拷贝文件慢了一点,但在使用时,因为重新调入 COMMAND.COM 是从虚拟盘调入,所以甚至比硬盘还快。

以上两种方法也可以 DOS 状态下直接输入。

此外,笔者很想求答全部 DOS 环境变量的内容及用法。

# 任意盘任意子目录下调用 WS 的方法

甘肃 长庆石油勘探局第二机械厂计算中心(745100) 何 捷

汉字 WordStar 是使用很广泛的字处理软件,但由于 WordStar 是从 CP/M 操作系统下移植过来的,所以不支持子目录,其覆盖模块也只能从指定的系统盘读出。本刊 1991 年第七期刊登的《若干子目录下共用一个 dBASE III (或 WS) 软件包》一文中,提供了一种子目录下文件共享的方法,适用于 dBASE III 等软件;但对 WS 则无效。这里,提供解决这一问题的两种方法,供大家参考。

先将 WordStar 诸文件拷入 C 盘 WS 子目录中(C:\WS);然后,修改 WS.COM 文件中的系统盘号。过程如下:

C>DEBUG\WS\WS.COM

-E2DC n

-W

-Q

其中,n 为最大驱动器号或虚拟盘号,若为 D:盘,则 n=4;为 E 盘时,则 n=5,余类推;

在 AUTOEXEC. BAT 启动文件中增加二行:

SUBST X: C:\WS

COPY \WS\\*. \* X:>NUL

第一行是将子目录用盘符 X:代替,X:为第一个自由盘符(即最后驱动器号)。如系统已有 C、D、E、F 盘,则 X=G。X:盘可在 CONFIG. SYS 中配置:LAST-DRIVE=X

X 为确定可访问的最大驱动器数。

第二行是将子目录 WS 中的 WordStar 诸文件拷贝到 X 盘上,>NUL 为不显示信息。

另外,也可使用虚拟盘(d:)的方法:

在 CONFIG. SYS 文件中增加一行:

DEVICE=VDISK. SYS Y 512 64/E

Y——虚拟盘空间,由机器内存大小而定。如 1M 内存,则 Y=384。

在 AUTOEXEC. BAT 文件中增加一行:

COPY \WS\\*. \* d:>NUL

其中,d:为虚拟盘号。

这样,在启动机器后,便可在任何路径下调用 WordStar 软件。



# C—dBASE III 数据库文件的结构 及其与其它语言的共享

南京百子亭 43 号 201 室(210009) 卢小平

在数据库语言中,C—dBASE III 要算众所周知的了。它的数据管理通常是由命令文件对数据库文件(.DBF 文件)进行操作。其它语种如要调用数据库文件,一般是把 .DBF 文件转换成标准文本文件(.TXT 文件),再由其它语言程序对文本文件进行操作。但这样做有两个缺点:①在硬盘上必须再建立一个 .TXT 文件,而它的大小和 .DBF 文件不相上下,这样就增加了大量重复的信息,占用了硬盘资源;②. TXT 文件一般都是等长数据记录,不含有文件结构信息,使得其它程序处理时不易控制。我们在实际工作中,需要在图形处理状态下调用数据库文件,为了解决上述问题,我们采取了用 C 语言直接调用数据库文件的方法,收效甚佳。

其它语言程序对 .TXT 文件的读写,是利用了它是等长记录这一特点。实际上 .DBF 文件也是等长记录,不同之处是在数据记录的前面放置了文件结构信息,文件结构如下:

文件开始的 32 个字节是数据库文件的总体参数说明,以后是字段说明,每个字段说明都占 32 个字节,全部字段说明之后是一个回车符(0DH)和一个空字符。这样,文件结构部分共占  $32(n+1)+2$  个字节, $n$  为字段数。

文件总体参数说明部分的第一个字节必须是 03,第 2~4 字节是日期,第 5 至 8 字节是记录条数,第 9、10 字节是文件结构长度,第 11、12 字节记录长度,其它都为 0。

字段说明部分的开头 10 个字节是字段名,可放 5 个汉字或 10 个 ASCII 字符,第 12 字节表示字段类型,如 N(4EH)为数值型,C(43H)为字符型,第 13、14 字节表示同一记录中下一字段的位置,第 17、18 字节是字段宽度。

上述文件结构的后面就是等长数据记录,其长度和各字段已在文件结构中说明,数据记录以 ASCII 字符形式连续存放,相邻记录间、字段与字段之间没有间隔。

如以下的文件结构

```
03 5B 0C 17 10 00 00 00 61 01 41 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
41 42 43 44 00 00 00 00 00 00 43 05 00 3F 53
04 00 00 00 01 00 00 00 00 00 00 00 00 00 00
58 59 5A 00 00 00 00 00 00 00 4E 08 00 3F 53
03 00 00 00 01 00 00 00 00 00 00 00 00 00 00
.....
```

第一行第 5、6 字节 1000 表示有 10H 个记录,第 9、10 字节 6101 表示文件结构长度为 0161H,第 11、12 字节 4100 表示记录长度为 41H,第三行前 4 个字符表示第一个字段名为 ABCD,第 12 字节 43 表示是字符型,第 13、14 字节 0500 表示下一字段从同一记录中 05H 开始,第四行开头 0400 表示字节段宽度为 4,第五行为后字段说明,字段名为 XYZ,类型为数值型。

现举例说明,用 C 语言读取 EXAM.DBF 数据库文件的结构信息,并读取第一个记录,以便显示或打印。C 程序如下:

```
#include <stdio.h>
main()
{long int struc.len,record.len;
 int fd,i,j,*n;
 char *record,**name;
 fd=open("EXAM.DBF",0);
 fseek(fd,8,0);
 read(fd,record,2);/*读文件结构长度*/
 struc.len=*(record+1)*0x100+*record;
 read(fd,record,2);/*读记录长度*/
 record.len=*(record+1)*0x100+*record;
 fseek(fd,32,0);
 for(i=1;32*(i+1)+2>struc.len;i++){
 read(fd,record,32);/*读字段说明*/
 for(j=0;*(record+j);j++){
 ***(name+j)=*(record+j);/*字段名赋值*/
 name++;
 *n=*(record+18)*0x100+*(record+17);/*字段宽度赋值*/
 n++;
 }
 fseek(fd,struc.len+1,0);
 read(fd,record,record.len);/*读记录*/
 close(fd);
 }
```

读程序首先打开 EXAM.DBF 文件,读取数据库文件总体参数,分别读每一字段说明(本程序只读取了字段名和字段宽度),最后读取第一个完整的记录,以后就可以根据具体的需要,编制屏幕格式或打印格式的程序段,程序中的数组都采用指针变量,可使得数组大小较为灵活。以上程序在 TurboC 上编译通过。

# “中国青蛙”病毒

成都乡农市街 53 号(610031)罗亚

笔者最近在 IBM 机上发现了一种新病毒,用当前流行的 CPAV、TNTVIRUS、SCAN、KILL 等均不能发现和清除。

此病毒只感染 A 盘和 C 盘,且感染和发作条件很特殊:

1. 用引导区染毒的盘启动时,病毒驻留内存,占用高端的 2048 字节,修改 13H 号中断向量,感染 C 盘的引导区(修改原引导程序,另外占用两个扇区存放病毒程序)。以后若使用 A 盘,21H 号中断向量也将改变。

2. 当病毒驻留内存后,若读写 A 盘则感染 A 盘的引导区。

3. 病毒驻留内存后,若当前盘为 A 盘,则每次用 INT 21H 的 4B 功能调入一程序时,感染 A 盘的第一个未染毒的 COM 文件,在文件尾部增加 122 字节的部分病毒程序。

4. 运行任何盘上的染毒 COM 文件时,若当前盘引导区染毒,则从当前盘调入占两个扇区的病毒程序整体(不驻留内存),感染 C 盘。

5. 当病毒驻留内存后,若机内时钟的秒计数器 0046CH 单元恰为 1 时读写 A 盘,则病毒发作,在屏幕上打出“—China frog 1990.10.1—”。

6. 若只有 COM 文件染毒,而无任何盘引导区染毒,则病毒不能驻留、传播或发作。

7. 病毒对引导区和 COM 文件均不重复感染。COM 文件的第四字节若为 4BH 则认为已染毒,可利用这一点进行免疫。引导区的病毒标记是病毒程序的头十六个字节。

根据以上特点,可知这是一个良性病毒,发现此病毒的办法是用 CHKDSK 查看内存总量是否比以往减少 2K,或 COM 文件是否莫名其妙地增加了 122 字节,或用 DEBUG 查看 COM 文件的开头是否为一条三字节 JMP 指令后跟 4BH(DEC BX 指令)。

由于病毒感染 COM 文件和引导区,手工消毒工作量太大,所以笔者编了两个小程序,在 DEBUG 下照文末清单输入即可建立 CHBOOT.COM 和 CHCOM.COM 两文件,分别用于检查清除当前盘的引导区和 COM 文件。CHBOOT 检查到染毒引导区则显示“boot infected”并消毒,CHCOM 检查到染毒文件则输出文件名并消毒。

例如,CHBOOT.COM 和 CHCOM.COM 在 C 盘:要清查 A 盘,需先将当前盘设为 A 盘,再键入 C:CHBOOT 或 C:CHCOM 消毒。注意,若 A 盘有子目录,需分别在各子目录下键入 C:CHCOM,因 CHCOM

只清查当前盘当前路径中的文件。

```
-A100
XOR     DH,DH
MOV     CX,0001
MOV     AH,19
INT     21
CMP     AL,02
JB      010F
ADD     AL,7E
MOV     DL,AL
MOV     AX,0201
MOV     BX,0200
INT     13
MOV     AL,EB
CMP     [BX],AL
JZ      013A
MOV     SI,01BE
MOV     AL,80
CMP     [BX+SI],AL
JZ      0132
ADD     BL,10
CMP     BL,40
JNZ     0124
JMP     0173
INC     BX
MOV     DH,[BX+SI]
INC     BX
MOV     CX,[BX+SI]
JMP     0111
ADD     BL,[0201]
INC     BX
INC     BX
MOV     AX,2BEA
CMP     [BX+22],AX
JNZ     0173
PUSH    DX
PUSH    CX
MOV     DX,0175
MOV     AH,09
INT     21
MOV     CX,[BX+27]
MOV     DX,[BX+29]
MOV     AX,0202
MOV     DI,BX
MOV     BX,0400
INT     13
MOV     CX,002B
MOV     SI,07B4
REPZ   MOVSB
POP     CX
POP     DX
MOV     AX,0301
MOV     BX,0200
INT     13
INT     20
```



```

DB      "boot infected" 0A 0D 24
-R CX
-85
-N CHBOOT.COM
-W
-A100
MOV     SP,03FE
MOV     DX,0200
MOV     AH,1A
INT     21
MOV     AH,4E
MOV     DX,0182
XOR     CX,CX
INT     21
JB      0180
MOV     DX,021E
PUSH    DX
MOV     AX,3D00
INT     21
MOV     BX,AX
MOV     AX,DS
ADD     AX,0040
MOV     DS,AX
XOR     DX,DX
MOV     AH,3F
MOV     CX,FFFF
INT     21
SUB     AX,007A
MOV     DI,AX
MOV     AX,1E8B
CMP     [DI],AX
MOV     AH,3E
INT     21
POP     DX
JNZ     017A
PUSH    DS
PUSH    ES
POP     DS

```

```

MOV     AH,3C
XOR     CX,CX
INT     21
MOV     CX,DI
MOV     BX,AX
MOV     AH,0E
MOV     SI,DX
LODSB
OR      AL,AL
JZ      015B
INT     10
JMP     0152
MOV     AL,0A
INT     10
MOV     AL,0D
INT     10
POP     DS
XOR     DX,DX
MOV     AX,[DI+72]
MOV     [0000],AX
MOV     AX,[DI+74]
MOV     [0002],AX
MOV     AH,40
INT     21
MOV     AH,3E
INT     21
PUSH    ES
POP     DS
MOV     AH,4F
JMP     010C
INT     20
DB      "*.COM"00
-R CX
-88
-N CHCOM.COM
-W
-Q

```

## 以毒攻毒——消除 DIR-2 病毒的简便方法

新疆石河子卫生学校微机室(832000) 刘山水

当今,各种计算机病毒泛滥成灾,为了对付它们的攻击,各种防病毒技术也不断涌现。然而,道高一尺,魔高一丈,病毒制造者们也在以更高明的手段对抗、躲避防病毒软件和硬件(卡)。最近发现的 DIR-2 病毒就是一个很好的例子。经笔者测试,该病毒能够成功地躲过一些较先进的防病毒软件的检测和清除,如公安部推出的 SCAN 3.0 和 KILL V50.00, PCTools V7.0 所

带的 VDEFEND 和 CPAV, CARMEL 公司的 TNTVIRUS V7.10A, 据《软件报》(第 33 期)称连防病毒卡也投降了,可见该病毒的威力是如何之大。但是,既然病毒是人造出来的,那么,人们也能够清除它。

### 一、病毒表现

尽管该病毒很隐蔽,本身没有可表现的信息,一般

不易被发现,但因为它是操作系统中一种不正常的程序,具有病毒的共性——繁殖和传播,所以它总会暴露出一些异常状态。例如,有时发现某个可执行文件不能运行或某个非可执行文件不能正常读出;使用 CPAV 检测病毒时产生死机;有时读写保护的软盘或运行其中的文件时(假设被操作的盘在 A 驱动器)出现:

File allocation table bad,drive A.

运行 CPAV 的 VSAFE 或 TNTVIRUS 的 TSAFE,打开一般写保护开关后,再读软盘会产生程序要写盘的警告,这是不正常的,因为 DOS 的 DIR 命令是不会产生写操作的,此时病毒的“狐狸尾巴”就露出来了。

## 二、病毒危害

1. 占据磁盘的末簇并覆盖其中的数据,从而造成数据的丢失或程序的破坏,其后果是不可恢复的。

2. 病毒的传播速度非常快,在病毒环境中,只要一列目录,就执行传染。病毒的名称也由此而得。

3. 在正常系统环境下,使用 CHKDSK 命令检查感染盘时,会发现文件目录表有问题,因为结果造成所有的被感染的文件丢失,若没有备份,则是一大损失。例如:执行 CHKDSK /F 对感染盘修复,回答 Y,则所有丢失链的可执行文件都变为 0 字节;回答 N,则所有丢失链的可执行文件都变为 1024 字节(软盘)或 4096 字节(硬盘),实为病毒体,若不注意文件大小的改变,直接运行,则又重新感染病毒。

4. 在正常环境下对感染的盘写操作时,有可能破坏该盘所有文件的链,从而造成更严重的后果。

## 三、病毒特征

1. 无论在正常系统环境下还是在病毒环境下,都可用 PCTools 或 DEBUG 工具在磁盘的末簇的第一个扇区搜寻到病毒标志“BC0006FF06EB04”。只有在正常系统环境下,才能在可执行文件的头部找到这个标志。

2. 在正常系统环境下,用 PCTools 的磁盘文件映像功能,可清楚地看到,所有可执行文件都占用磁盘的末簇。而在病毒环境下却无此现象。

3. 与以往所发现的病毒不同,该病毒有以下五个新特点:

- 1) 不改变文件长度、属性、日期和内容。
  - 2) 不改变内存大小。
  - 3) 不修改中断向量。
  - 4) 不修改或占用系统引导区。
  - 5) 没有激活条件和表现部分。
- 这五个新的特点使病毒的概念发生了变化。

## 四、病毒原理

该病毒既不以程序为载体,也不修改中断向量,又不占用或修改系统引导区,那么,它是如何传播的呢?这就是该病毒设计上的奥秘所在。

该病毒是一种独立存在,以磁盘和内存为宿主的新型病毒。其主要功能步骤如下:

1) MOV SP,600H;这是病毒的第一条指令,目的是将堆栈移至安全区,保护自己。

2) 调用 DOS 的 30H 功能,针对不同版本的 DOS 进行传染。

3) 调用 DOS 的 4AH 功能,改变所分配的内存块,将病毒体驻留内存,这种驻留方式很特别,没有使用“显眼”的 DOS 中断和功能调用。

4) 调用 DOS 的 52H 功能,取 DOS 内部值,根据其值修改 DOS 常驻内存的功能调用的地址入口为病毒所在地址。若地址已为病毒地址(即病毒已在内存),则不修改。

5) 执行 DOS 的 3DH 功能,虽然这是一个读盘功能,但却执行了病毒的写盘操作,将病毒体写入硬盘的末簇的第一扇区,修改 FAT 表使末簇为已用标志,防止被覆盖。同时将硬盘当前目录的可执行文件的起始簇改为末簇。即首先感染硬盘。因为一般都以硬盘为系统引导盘,所以,病毒以硬盘为宿主,从而保证每次从硬盘启动都能够先进入内存,进行传播。

6) 执行 DOS 的 4B00H 功能,加载用户程序并执行。因为被感染的可执行文件的开始簇为病毒所在地址——磁盘的末簇的第一个扇区,所以每次运行可执行文件时,必先执行病毒程序,然后由病毒执行 DOS 的加载功能来运行可执行文件,使得用户在运行程序时不至于出错而暴露出自己。

病毒一旦驻留内存,就开始传播了。其部分传播程序段如下:

```

8CCF:0406 3D4558      CMP AX,5845;是 EXE 文件吗?
8CCF:0409 7505       JNZ 0410
8CCF:040B 38440A     CMP [SI+0A],AL
8CCF:040E 740B       JZ 041B
8CCF:0410 3D434F     CMP AX,4F43;COM 文件吗?
8CCF:0413 753E       JNZ 0453
8CCF:0415 807C0A4D   CMP BYTE PTR[SI+0A],
4D
8CCF:0419 7538       JNZ 0453
8CCF:041B F7441EC0FF  TEST WORD PTR[SI+1E],
FFC0
8CCF:0420 7531       JNZ 0453
8CCF:0422 F7441DF83F  TEST WORD PTR[SI+1D],
3FF8;大于 2K 吗?
8CCF:0427 742A       JZ 0453
8CCF:0429 F6440B1C   TEST BYTE PTR[SI+0B],1C;
判断是否为系统文件、
子目录或卷标
8CCF:042D 7524       JNZ 0453
8CCF:042F 84D2       TEST DL,DL
8CCF:0431 7513       JNZ 0446
8CCF:0433 B86301     MOV AX,0163;判断是否已感
染过(163H 为 低密软盘

```

```

                                的最后簇号)
8CCF:0436 3B441A    CMP AX,[SI+1A]
8CCF:0439 7418      JZ 0453
8CCF:043B 87441A    XCHGAX,[SI+1A]
8CCF:043E 3517FE    XOR AX,FE17;低密软盘加
                                密初始值
8CCF:0441 894414    MOV [SI+14],AX
8CCF:0444 E20D      LOOP 0453
8CCF:0446 31C0      XOR AX,AX;首簇号解密段
8CCF:0448 874414    XCHGAX,[SI+14]
8CCF:044B 2E        CS;
8CCF:044C 33063F04  XOR AX,[043F]
8CCF:0450 89441A    MOV [SI+1A],AX
8CCF:0453 2E        CS;
8CCF:0454 D1063F04  ROL WORD PTR[043F],1
8CCF:0458 83C620    ADD SI,+20
8CCF:045B 39F7      CMP DI,SI
8CCF:045D 75A4      JNZ 0403
8CCF:045F C3          RET

```

病毒将所有满足条件的可执行文件的开始簇号加密后,放到目录项的保留区内,所用的加密初始值与磁盘规格有关(360K 软盘为 FE17H,高密盘为 03EBH,硬盘为 FDADH),加密算法为每找一个目录表的表项,初始值就循环左移一位,把满足条件的可执行文件的开始簇号与之进行异或运算。解密的过程与之相同。这就是在病毒环境中一切正常,而在正常环境中一切异常的原因。

由此可见,由于所有的可执行文件的开始簇都是病毒体的所在地,所以一运行可执行文件,就首先运行病毒,病毒虽然没有以程序为载体,却达到了以程序为载体的目标,同时,克服了靠修改文件(文件型病毒)或系统引导区(系统型病毒)而易暴露的缺点,可见其设计手段是相当高明的。这样,在病毒环境下,检查文件内容是不会发现异常的。另外,病毒中使用 DOS 功能调用的方法更为独特,这是对以往病毒的一个重大改进,也是该病毒设计上最精彩的部分。

具体程序如下:

```

8CCF:0114 B430      MOV AH,30
8CCF:0116 E82401    CALL 023D;调用子程序
.
.
.
.
8CCF:023D 9C        PUSHF
8CCF:023E 2E        CS;
8CCF:023F FF1EFC05    CALL FAR[05FC];调用 258:
                                1467 程序段,该程序段就是
                                DOS 驻留在内存中的 INT 21H
                                部分
8CCF:0243 C3          RET

```

INT 13H 的调用也是采用同样方法,这样直接引用系统功能的程序段来使用中断和功能调用比修改中断向量要高明得多,所以能够成功地逃避一些靠监视中断向量来检测病毒的防病毒软件。

## 五、病毒的检测

用 McAfee Associates 提供的 SCAN 8.3B86 可直接检测出内存和可执行文件的病毒,若没有该软件,则可用正常的 DOS 启动后,用 CHKDSK 命令检查磁盘,若发现所有可执行文件的链都指向末簇时,就可初步断定为该病毒,这时再用 DEBUG 或 PCTools 查出病毒标志,就可完全断定为该病毒。

## 六、病毒的清除

弄清了病毒的原理之后,就好对付它了,即所谓知己知彼,百战不殆。

最简单的方法莫过于以毒攻毒,这里就介绍一种不用清病毒软件就能清除该病毒的方法,具体作法如下:

### 1. 取病毒体

(1)发现病毒后,在病毒环境下,取一张没有文件的软盘,用 DIR 命令读一次该软盘。

(2)用正常的 DOS 系统启动,对刚才那张盘执行 CHKDSK /F 命令,回答 Y 之后,发现盘上有一个文件 FILE0000.CHK,把它的后缀改为 COM。

### 2. 修改病毒

用 DEBUG 调入并执行以下命令:

```

-E CS:22C 90 90 90
-E CS:439 75
-E CS:43D 14
-E CS:443 1A
-W
-Q

```

### 3. 消毒

在 DOS 提示符下,运行修改后的病毒,用 DOS 的 PATH 命令设置所有含有可执行文件的磁盘子目录,然后键入一个磁盘上没有的可执行文件名,则磁盘上的可执行文件就能恢复正常。

### 4. 清除病毒体

以上步骤操作完毕后,切记不要对磁盘执行写操作!重新启动机器,执行 CHKDSK /F,会发现只有一个丢失簇,回答 N,病毒就被彻底清除。

### 七、病毒的预防

该病毒的预防较困难,只有加强病毒防护知识,坚决杜绝使用未检查过的软盘,建立严格的病毒预防、监测制度才能真正防患于未然。

[注]文章中所用程序和内存的地址是在 PC-DOS 3.30 系统环境中取得的。



# XMF—BASIC 全能扩充

甘肃临洮县洮阳镇临洮农业学校(730500)石永琳

中华学习机“小蜜蜂—1”的驻机解释程序 XMF—BASIC 是同类机型中功能较为完善的系统程序。它具备一些在 APPLE、CEC—1 等同类机器中需要扩充才能实现的功能,如用表达式作转移目标、表达式中直接使用十六进制数、十进制数与十六进制数的转换、自动行号、行编辑等。特别是该机的行编辑命令 EDIT,使一行程程序的编辑变得简单,容易掌握。

遗憾的是 XMF—BASIC 有 2 个命令不支持汉字系统。它们是 GET 和 XMF—BASIC 独有的 EDIT 命令。GET 只能读 ASCII 字符,不能正确读取汉字(实际读取的是汉字国标码的低 2 位),EDIT 则完全不能执行,使用户深感不便。为了弥补这个不足,笔者扩充了下面程序,在汉字系统中实现了原系统的 GET 和 EDIT 命令的功能,增加了 RESTORE(行号)功能,使 XMF—BASIC 成为完整的系统程序。

将本文所附的机器码程序输入机内,核对无误后用 BSAVE BAS.EXPANSION.OBJ, A \$7C00, L \$400 存盘备用。扩展程序安排在 \$7C00~\$7FFF 区。首次用 BRUN BAS.EXPANSION.OBJ 启动, BASIC 程序中用 HIMEM: \$7C00 保护。扩展功能用 & 命令调用,现说明如下:

## 1. &RESTORE(行号)

功能:按指定的 DATA 语句行号恢复数据。

## 2. &GET(变量)

功能:给变量从键盘读一个字符或汉字,对变量的类型没有限制。

## 3. &EDIT(行号)

功能:编辑指定的一行程程序,设置的功能键如下。

→:光标右移一个字符或汉字

←:光标左移一个字符或汉字

CTRL—T:光标上移一行

CTRL—V:光标下移一行

CTRL—I:光标处插入一个空位

CTRL—I:删除光标位置的字符或汉字

CTRL—X:放弃对程序行的编辑并退出

RETURN:结束编辑。

在汉字系统中,编辑一行程程序时,由于要涉及到西文字符和汉字这两类代码形式、字符宽度都不相同的对象,情况较为复杂。在程序中已对这些问题作了周密考虑,使光标移动、插入、删除、替换等操作对汉字与西文字符都能自适应。

另外,也允许用 &EDIT 命令编辑一行新程序,这样做的好处,是在新输入程序时即有上述的编辑功能。

程序中, \$7C00—\$7C17 设置 & 命令入口地址; \$7C18—\$7C33 是 &RESTORE 命令的解释程序; \$7C34—\$7C8F 是 &GET 命令的解释程序; \$7C90—\$7FFB 是 &EDIT 命令的解释程序。其中 &GET 是一个非常有用的子程序,读者可剪接到自编程序中。

扩充程序中留有用户接口,当用户需要用 & 扩充新的命令时,把入口地址按先低位后高位填入 \$7CA4 和 \$7CA5 二单元。这二单元现在指向 \$DEC9,即显示出错信息“SYNTAX ERROR”。

程序调用了原系统中 GET、EDIT 命令的部分子程序。对汉字系统程序的不适应部分作了屏蔽处理。如

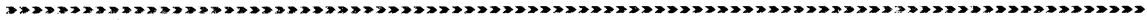
→ ← RETURN ESC 四个键的处理权均在扩充程序中,不会因误按键而发生错误操作。

7C00— A9 4C 8D F5 03 A9 18 8D  
7C08— F6 03 A9 7C 8D F7 03 A9  
7C10— 00 85 73 A9 7C 85 74 60  
7C18— 48 20 B1 00 68 C9 AE D0  
7C20— 13 20 7B DD 20 52 E7 20  
7C28— 1A D6 A5 9B A4 9C 38 E9  
7C30— 01 4C 50 D8 C9 BE D0 58  
7C38— A5 ED D0 03 4C A0 DB 20  
7C40— 06 E3 A9 40 A2 01 A0 02  
7C48— 85 15 86 7F 84 80 20 E3  
7C50— DF 85 85 84 86 A5 B8 A4  
7C58— B9 85 87 84 88 20 67 7C  
7C60— E8 20 39 D5 4C 26 DC A9  
7C68— D0 8D 6E AB 8D 80 AB A9  
7C70— 01 8D 6F AB 8D 81 AB A2

7C78— 00 20 0C FD 9D 00 02 A9  
7C80— 20 8D 6E AB 8D 80 AB A9  
7C88— 0A 8D 6F AB 8D 81 AB 60  
7C90— C9 EC D0 0F 20 B7 00 90  
7C98— 03 4C C9 DE A5 ED D0 06  
7CA0— 4C 25 D0 4C C9 DE 20 76  
7CA8— D0 A0 05 B9 2C D1 20 5C  
7CB0— DB 88 D0 F7 84 84 84 86  
7CB8— 20 B7 00 20 0C DA 20 1A  
7CC0— D6 20 D8 D6 A9 00 8D AF  
7CC8— AA 8D B3 AA A9 F9 8D CF  
7CD0— A7 A9 60 8D E1 A5 A2 00  
7CD8— A0 00 A9 0F 84 EF 85 2A  
7CE0— 2C 4C C0 20 0D D8 B1 28  
7CE8— C9 80 B0 10 48 AD 0A AE

7CF0— 9D 03 02 E8 68 9D 03 02  
7CF8— E8 C8 B1 28 9D 03 02 E4  
7D00— 86 B0 06 20 31 D8 4C E6  
7D08— 7C A9 00 9D 03 02 20 6A  
7D10— D8 A2 00 A0 00 A9 0F 85  
7D18— 2A 20 0D D8 A5 2A 85 25  
7D20— 84 24 8A 48 98 48 20 67  
7D28— 7C 68 A8 68 AA AD 00 02  
7D30— C9 95 D0 06 20 71 7D 4C  
7D38— 1C 7D C9 88 D0 06 20 83  
7D40— 7D 4C 1C 7D C9 8D D0 03  
7D48— 4C C8 7D C9 98 D0 03 4C  
7D50— E5 7D C9 84 D0 03 4C F8  
7D58— 7D C9 89 D0 03 4C 85 7E  
7D60— C9 94 D0 03 4C DF 7F C9

7D68-- 96 D0 03 4C B3 7F 4C E8	7E48-- 20 0A A0 A5 2A 85 25 84	7F28-- 7E 20 6A D8 20 7D D8 20
7D70-- 7E E0 E0 90 01 60 BD 03	7E50-- 24 68 C9 FE 90 0F 20 0A	7F30-- 71 7D E0 E3 90 03 20 83
7D78-- 02 C9 FE 90 03 E8 E8 C8	7E58-- A0 E8 BD 03 02 20 0A A0	7F38-- 7D 4C 1C 7D E0 E1 B0 D4
7D80-- 4C 2D D8 E0 03 90 1B BD	7E60-- E8 C8 BD 03 02 20 0A A0	7F40-- 20 50 7F 20 32 7E 20 6A
7D88-- 00 02 C9 FE 90 14 CA CA	7E68-- 20 31 D8 60 EA A5 9E 8D	7F48-- D8 A9 00 9D 03 02 F0 E2
7D90-- 88 20 A2 7D C0 20 D0 09	7E70-- 6C 7E 20 6A D8 CC 6C 7E	7F50-- AD 00 02 9D 03 02 AD 01
7D98-- C8 B1 28 88 C9 80 90 01	7E78-- F0 0A A9 A0 20 35 7E CC	7F58-- 02 9D 04 02 AD 02 02 9D
7DA0-- 88 60 A5 2A C9 0F F0 03	7E80-- 6C 7E D0 F6 60 E4 9F B0	7F60-- 05 02 60 BD 03 02 C9 FE
7DA8-- 4C 39 D8 CA 88 4C 41 D8	7E88-- 1E A5 9F C9 E1 90 1B 8E	7F68-- 90 24 AD 00 02 9D 03 02
7DB0-- 2C 4D C0 A9 95 8D AF AA	7E90-- 6C 7E A6 9F BD 00 02 AE	7F70-- 20 87 D8 E8 BD 05 02 9D
7DB8-- A9 88 8D B3 AA A9 A9 8D	7E98-- 6C 7E C9 FE 90 06 A5 9F	7F78-- 03 02 D0 F7 20 7D D8 20
7DC0-- E1 A5 A9 F3 8D CF A7 60	7EA0-- C9 E3 90 06 20 3A FF 4C	7F80-- 27 7E 20 6D 7E 20 7D D8
7DC8-- 20 B0 7D A9 A0 8D 00 02	7EA8-- 1C 7D 20 87 D8 20 C4 7E	7F88-- 20 71 7D 4C A9 7F A5 2A
7DD0-- 8D 01 02 8D 02 02 A5 9D	7EB0-- 20 7D D8 A9 A0 9D 03 02	7F90-- 85 25 84 24 AD 00 02 9D
7DD8-- 85 25 A6 9F E8 E8 E8 20	7EB8-- 20 27 7E 20 6A D8 20 7D	7F98-- 03 02 20 65 7E E4 9F 90
7DE0-- 39 D5 4C 44 D4 20 B0 7D	7EC0-- D8 4C 1C 7D A6 9F BD 03	7FA0-- 0F A9 00 9D 03 02 20 6A
7DE8-- A5 9D A4 9E 85 25 84 24	7EC8-- 02 9D 04 02 E0 00 F0 05	7FA8-- D8 E0 E1 90 03 20 83 7D
7DF0-- A9 DC 20 0A A0 4C 3C D4	7ED0-- CA E4 A2 B0 F1 60 A6 9F	7FB0-- 4C 1C 7D A5 2A C5 9D B0
7DF8-- E4 9F B0 12 20 87 D8 20	7ED8-- BD 03 02 9D 05 02 E0 00	7FB8-- 23 20 87 D8 E6 A0 20 71
7E00-- 11 7E 20 7D D8 20 27 7E	7EE0-- F0 05 CA E4 A2 B0 F1 60	7FC0-- 7D 14 9F B0 14 E0 E0 F0
7E08-- 20 6D 7E 20 7D D8 4C 1C	7EE8-- C9 FE 90 77 BF 03 02 C9	7FC8-- 13 A5 2A C5 A0 90 EF C4
7E10-- 7D BD 03 02 C9 FE E8 90	7EF0-- FE 90 09 20 50 7F 20 32	7FD0-- A1 B0 09 C0 1F 90 E7 B0
7E18-- 05 BD 05 02 B0 03 BD 03	7EF8-- 7E 4C 32 7F E4 9F B0 3C	7FD8-- 03 20 83 7D 4C 1C 7D A5
7E20-- 02 9D 02 02 D0 F0 60 BD	7F00-- E8 E4 9F CA 90 08 A5 9F	7FE0-- 2A C9 0F F0 14 20 87 D8
7E28-- 03 02 F0 05 20 35 7E D0	7F08-- C9 E2 90 0E B0 06 A5 9F	7FE8-- C6 A0 20 83 7D A5 2A C5
7E30-- F6 60 BD 03 02 48 C0 1F	7F10-- C9 E0 90 06 20 3A FF 4C	7FF0-- A0 B0 F7 C4 A1 F0 02 B0
7E38-- 90 11 C0 21 B0 0D A5 2A	7F18-- 1C 7D 20 87 D8 20 D6 7E	7FF8-- F1 4C 1C 7D
7E40-- 85 25 A9 21 85 24 A9 A0	7F20-- 20 7D D8 20 50 7F 20 27	



(上接第 22 页)

文件类型(1 字节):一个文件的内部结构的描述。

关键指针(2 字节):被这文件项目描述的子目录或标准文件的关键块的块地址。

使用块数(2 字节):被文件实际使用的总块数。对于一个子目录文件,这包含子目录信息的块,但不包含指向文件的块。对于一个标准文件,这既包含资料块(索引块)又包含数据块。

EOF(3 字节):一个 3 字节整数,低字节在前,表示可从文件读取的字节总数。注意,就稀疏文件而论,EOF 可能比在磁盘上实际分配的字节数大。

建立部分(4 字节):由此项目指向的文件的建立日期和时间。这些字节的格式在后面描述。

版本号(1 字节):建立由此项目指向的文件时的 ProDOS 版本号。这字节允许新版本的 ProDOS 确定文件的格式,并相应地调整分析过程。ProDOS 1.1.1:版本号=0

最小版本号(1 字节):可以存取这文件内资料的

ProDOS 的最低版本号,此字节允许低版本的 ProDOS 确定它们是否可以存取新版文件,ProDOS 1.1.1:最低版本号=0

存取标记(1 字节):确定这文件是否可以被读取、写入、删除和改名,和文件是否需要备份。此字段的值可以由 SET FILE INFO 调用改变。

辅助类型(2 字节):这是一个可以存储有关一个文件内部格式的附加信息的通用字段。例如,ProDOS BASIC 系统程序用这字段记录一从二进制文件的载入地址或一个文本文件的记录长度。

最后修改日期及时间(4 字节):在这文件执行一个 WRITE 之后,最后 CLOSE 操作的日期和时间。这些字节的格式在后面描述。此字段可由 SET FILE INFO 调用改变。

标题指针(2 字节):这字段是拥有此文件项目的目录的关键块的块地址。这两字节指针低字节在前,高字节在后。  
(未完待续)

# 实用程序二则

新疆哈密铁一中(839001)李成功

## 为《儿歌拼图》游戏软件制作续集

《儿歌拼图》是一个专为儿童设计的开发和增强智力的游戏软件。儿童们既可以在计算机上玩相当于六面画的拼图游戏,又可以欣赏那优美动听的音乐。该软件的二十幅图案都相当漂亮,深受儿童们的喜爱。但是,时间久了,儿童们还是想换一些图案进行拼图游戏。笔者通过分析摸索,发现该磁盘中的“P1~P20”是进行拼图的二十个图形文件。但是,用普通的图形文件是不能替换它们的,因为这些图形文件采用了压缩存储的技术。若用本文介绍的方法可以换掉这些图形文件,玩自己所喜爱的图形拼图游戏。笔者曾将《Easy as ABC》游戏软件中的二十动物图案替换《儿歌拼图》的二十幅图案,效果很好。具体步骤如下:

1. 首先,将《儿歌拼图》软盘的写保护纸揭掉,并插入一号驱动器。在监控状态下键入一段图形压缩存储的机器语言程序,并以 YASUO - SAVEPICT, A \$ 165A, L \$ 9F 存入磁盘。

2. 用 DELETE 命令将《儿歌拼图》软盘中的名为“P1~P20”的图形文件全部删除,在二号驱动器中插入一张存有名为“P1~P10”的未经压缩图形库磁盘。

3. 在 BASIC 状态下,输入一小段 BASIC 程序并运行。当计算机发出“嘀”的一声后,将二号驱动器中的图库盘抽出,再插入另外一张存有名为“P11~P20”的未经压缩图形库磁盘。按任意键,等机器再次发出“嘀”的一声后,取出一号驱动器中的磁盘,重新贴上写保护纸,至此,一张《儿歌拼图》游戏软件续集制作完毕。

对只有一个驱动器的用户,可以修改 BASIC 程序的某些语句,反复调换磁盘即可。本方法已在 APPLE - II 和 CEC - I 中华学习机上调试通过。

### 图形压缩程序

```

165A- 8A 29 C0 85 05 4A
1660- 4A 05 05 85 05 8A 85 06
1668- 0A 0A 0A 26 06 0A 26 06
1670- 0A 66 05 A5 06 29 1F 05
1678- E6 85 06 60 A0 01 84 04
1680- 84 03 88 84 02 A5 E6 09
1688- 04 85 06 84 05 B1 05 D0
1690- 02 09 80 A2 01 86 08 85
1698- 07 A4 02 A6 03 E8 E8 E0
16A0- C0 90 0F C8 C0 28 90 06
16A8- C6 04 30 1B A0 00 84 C2
16B0- A6 04 86 03 20 5A 16 B1
16B8- 05 D0 02 09 80 C5 07 D0
16C0- 06 E6 08 D0 D4 C6 08 48
16C8- A0 00 A6 08 F0 0E E0 04
16D0- B0 0A A5 07 20 FE 16 CA

```

```

16D8- D0 FA F0 0D 98 20 FE 16
16E0- 8A 20 EF 16 A5 07 20 EF
16E8- 16 68 24 04 10 A5 60 91
16F0- 00 E6 00 D0 02 E6 01 60
]BSAVEYASUO-SAVEPICT,A $ 165A,L $ 9F

```

主程序

```

10 D$ =CHR$(13)+CHR$(4):HGR
20 PRINT D$ "BLOAD YASUO-SAVEPICT,D1"
30 FOR I=1 TO 20: IF I=11 THEN PRINT CHR$(7):
   GET A$
40 PRINT D$ "BLOADP";I;","A$ 2000,D2"
50 POKE 230,32:POKE 0,0:POKE 1,96:CALL 5756
60 PRINT D$ "BSAVEP";I;","A$ 6000,L";PEEK(1)*
   256+PEEK(0)-24575;","D1"
70 NEXT:PRINT CHR$(7)

```

## 《Easy as ABC》软件的 图形打印与存储

《Easy as ABC》是美国 Spring board 软件公司出版的幼儿教育软件。二十多个色彩鲜艳、栩栩如生的动物图象引起了儿童们的浓厚兴趣。该软件的主要特点是把枯燥无味的英文识字变成了引人入胜的游戏。这张软盘上共有三十个高分辨图案。若将这些图案打印出来进行欣赏或将它们存入自己的磁盘中随时调用定会使你的软件增添许多光彩。可是这些图形在存储方式上都做了特殊处理,用 CATALOG 命令列磁盘目录是找不到这些图形文件的,就是使用 COPY I PLUS 4.3 等工具软件也无法列出。笔者通过分析摸索编制了一小段 BASIC 程序,运行它可以将《Easy as ABC》软盘中所有的图形存入另外两张磁盘中从而建立一个图形库。若只打印图形不建立图库,可将程序中的第 60 行删除,并增加第 65 行进行打印:

```

65 PRINT D$ ; "PR #1";PRINT CHR$(17);PRINT
   D$ ; "PR #0"

```

若要既打印图形又存储图形则保留程序的第 60 行。具体操作如下:

1. 准备两张已用 DOS 3.3 格式化好的磁盘并在二号驱动器中插入其中的一张。

2. 在一号驱动中插入《Easy as ABC》软盘后键入程序并运行。

3. 由于二号驱动器中磁盘存满后程序会中断,此时,在二号驱动器中换上另外一张已格式化好的空盘。

4. 将程序中的第 30 行改为“30 FOR I=16 TO 30”后再次运行程序即可得到后 14 个图形文件。

```

10 D$ =CHR$(13)+CHR$(4):PRINT D$ "BLOADB.
   ALPH+SHAPES+FONT"
20 POKE 230,32:POKE --16297,0:POKE --16302,0:POKE
   --16300,0:POKE --16304,0
30 FOR I=0 TO 29
40 POKE 43624,1
50 CALL 26020,150,I
60 PRINT D$ "BSAVEP";I;","A$ 2000,L $ 2000,D2"
70 NEXT

```



# 任意数的高精度乘法

山东苍山县酒厂(277700)张新莲

数学中把左右对称的数叫回文数,你见过下面这些回文数吗?

- ①12345678987654321
- ②121242363484363242121
- ③121024203630484036302420121
- ④12321024642036963024642012321
- ⑤102030405060708090807060504030201
- ⑥1002003004005006007008009008007006005004003002001

这些数决不是杜撰的,是经过精确计算而得出的。有趣的是这些数竟然都是平方幂,你能找出它们的根吗?

众所周知,所有计算机都能进行乘法运算。但是,所能处理数据的能力及运算精度却有天壤之别,对于中华学习机,整数 BASIC 要求乘数、被乘数、乘积分别限于 $-32768 \sim +32767$ 之间,超出这一范围就不能运行,而且不能运算小数。浮点 BASIC 虽能进行小数运算,并采用了科学计数法,使处理数据的能力得以极大地提高,但处理数字的最大值仅为 $10^{38}$ 次方,且运算精度较差,仅有九位有效数字。这样的精度虽能满足一般使用要求,但在某些特殊场合,如金融、航天等要求高精度计算的领域,则不能满足使用。以前曾有发表高精度乘法的文章,但只能进行整数乘法,不能处理小数的运算,而且还要预先输入被乘数、乘数的位数,再输入被乘数、乘数,才能得出乘积,输入操作较烦琐。本文介绍的通用高精度乘法程序,可以进行任意位带符号整数、小数的高精度乘法运算。

本程序以整数 BASIC 为出发点,尽量采用一般语句进行编写,已在中华学习机上通过,适用其兼容机。修改个别语句可方便地移植到 LASER 310、PC-1500、CAC、键盘游戏机等微机中。

程序的设计思路是采用下标变量,把被乘数、乘数、乘积分别装入三个数组中,把数字当作字符串输入微机,因此可以处理 255 位以内数字的运算。运算中仅限于两位数之内的加、减、乘法运算、因而不用担心溢出问题。设计中模仿笔算乘法的过程,采用循环控制语句,避免子程序的调用,尽量减少 GOTO 语句的使用,尽量缩小 GOTO 跳转的距离,从而使整个程序平铺直叙、分段清晰、功能明确,便于阅读理解,运行速度较快。经在 CEC-I 上测试典型数据如下:

计算 10 位数乘 10 位数,积为 20 位数,8 秒,

计算 30 位数乘 30 位数,积为 60 位数,60 秒,

计算 50 位数乘 50 位数,积为 100 位数,190 秒。

计算小数点后 97 个“0”后 999 乘以小数点后 97 个“0”后 888 积为 200 位小数,用时仅 3 秒钟。

程序说明如下:(程序清单单位附后)

1. 5、10 行是数字输入语句,把两个数据作为字符分别赋给 B\$, C\$ 字符变量,并测试出它们的字符长度。
2. 20~30 行的功能是判别乘积的符号,两乘数符号相同时为正,否则为负。
3. 35~60 行的作用是去掉小数点,并由 H 记录乘积尾数的长度。
4. 70~85 行的功能是对字符数据整形。整形的目的如下:如有算式  $0.00038 \times 0.00059$ ,去小数点后分别变为 000038、000059。这种形式直接进行乘法运算,并不影响乘积的正确,但却增加了循环次数,减慢了运算速度。可以算出,无效循环次数为 32 次,是有效循环的 8 倍。这无疑多占用了机时,浪费了时间。若小数的尾数更长,则浪费的时间更多。因此,必须在运算前除去有效数字前面的“0”,把两数整形为  $38 \times 59$ 。
5. 90~100 行的作用是把被乘数装入 B 数组,乘数装入 C 数组。
6. 105~125 行的功能是计算乘积。该处使用了双重循环。外循环作用于乘数,内循环作用于被乘数。完全模仿笔算过程,先从乘数的个位开始分别与被乘数各位相乘,乘积放入 A 数组的对应下标变量中;然后取出乘数中的十位数分别与被乘数各位相乘,各位乘积分别与上一次的乘积对应相加;然后再取乘数的百位,如此循环往复,直到所有乘数运算完毕。其中 115、120 行为乘积的进位判断和处理程序,以保证每个下标变量中的乘积都是单位数。
7. 130 行的作用是去掉乘积整数前面的“0”,使输出的乘积符合人们的记数习惯。如 03587 经过 130 行的处理变为 3587。
8. 135 行是对乘积尾数的添“0”处理。如  $0.00038 \times 0.00059$ ,乘积尾数  $H=10$ ,由于进行了去小数点整形,运算结果为  $38 \times 59 = 2242$ ,在输出乘积时必须在 2242 前面添 6 个“0”后再点小数点。
9. 140 行是对无整数部分小数乘积的小数点前加“0”。
10. 145~155 行是乘积的输出程序。功能是把 A 数组中各下标变量的内容以高下标在前、低下标在后的顺序依次打印。若乘积中有尾数,则通过 150 行在恰当的位置打印小数点。
11. 160~170 行是结束处理语句。若继续运算,按除“N”键外的任意键;若结束运算,则按“N”键。

输入程序,上机运算,可得出以上回文数分别是下列 6 个数的平方幂。

- ①111111111
- ②11011011011
- ③11001100110011
- ④111000111000111
- ⑤10101010101010101
- ⑥1001001001001001001001

你想验证一下吗?

相信你会找出更多更有趣的回文数。

```

1 REM "GAO JING DU CHENG FA"
5 HOME: CLEAR: INPUT "BEI CHENG SHU = ?"; B$: B
  = LEN(B$)
10 INPUT "CHENG SHU = ?"; C$: C = LEN(C$)
15 E = 0; F = 0; H = 0; U = 0; V = 0; PRINT B$; "X"; C
  $; "=";
20 IF MID$(B$, 1, 1) = "-" THEN U = 1; B$ = MID
  $(B$, 2); B = B - 1
25 IF MID$(C$, 1, 1) = "--" THEN V = 1; C$ = MID
  $(C$, 2); C = C - 1
30 IF (U = 1 AND V = 0) OR (U = 0 AND V = 1) THEN
  PRINT "--";
35 FOR I = 1 TO B: IF MID$(B$, I, 1) = "." THEN E =
  I; GOTO 45
40 NEXT; GOTO 50
45 B$ = LEFT$(B$, E - 1) + RIGHT$(B$, B - E); H =
  B - E; B = B - 1
50 FOR J = 1 TO C: IF MID$(C$, J, 1) = "." THEN F =
  J; GOTO 60
55 NEXT; GOTO 65
60 C$ = LEFT$(C$, F - 1) + RIGHT$(C$, C - F); H

```

```

= H + C - F; C = C - 1
65 DIM A(B + C)
70 FOR I = 1 TO B: IF MID$(B$, I, 1) = "0" THEN
  NEXT
75 B = B - I + 1; B$ = RIGHT$(B$, B)
80 FOR J = 1 TO C: IF MID$(C$, J, 1) = "0" THEN
  NEXT
85 C = C - J + 1; C$ = RIGHT$(C$, C)
90 G = B + C; DIM B(B), C(C)
95 FOR I = 1 TO B: B(B - I + 1) = VAL(MID$(B$, I,
  1)); NEXT
100 FOR J = 1 TO C: C(C - J + 1) = VAL(MID$(C$, J,
  1)); NEXT
105 FOR K = 1 TO C: X = K
110 FOR J = 1 TO B: D = B(J) * C(K); A(X) = A(X) + D
115 IF A(X) > 9 THEN A(X + 1) = A(X + 1) + INT(A(X) /
  10); A(X) = A(X) - INT(A(X) / 10) * 10
120 IF A(X + 1) > 9 THEN A(X + 2) = A(X + 2) + 1; A(X
  + 1) = A(X + 1) - INT(A(X + 1) / 10) * 10
125 X = X + 1; NEXT J; NEXT K
130 IF A(G) = 0 THEN G = G - 1
135 IF G < H THEN G = H
140 IF G = H THEN PRINT "0";
145 FOR I = G TO 1 STEP -1
150 IF I = H THEN PRINT ". ";
155 PRINT A(I);; NEXT
160 PRINT; PRINT; PRINT "JI XU MA? (Y/N)";; GET A
  $
165 IF A$ <> "N" THEN RUN
170 HOME: VTAB 10; HTAB 10; PRINT "GOOD BY!!!";
  END

```

## ProDOS 系统内部结构剖析(续)

廖 凯

每块项目数(1字节):被存在目录文件的每一块内的项目数量。ProDOS 1.1.1:每块项目数 = \$0D

文件数(2字节):在此目录文件内现有的文件项目数量。一个现有文件的存储类型非零。请看后面有关文件项目的描述。

位图指针(2字节):卷位图的第一块的块地址。位图占据相连的块。一块位图对应于卷上每 4096 个块(或它的一部分)。你可以用后面描述的总块数字段计算块的数量。

位图的一位元(bit)对应于卷上每个块:1 意味着块是空的,0 意味着块已用。如果被卷上所有文件使用的块数与在位图内记录的数不同,那么卷的目录结构

已被破坏。

总块数(2字节):卷上块的总数。

3. 子目录标题

每个子目录文件的关键块被目录中某一项指定。一个子目录的标题从子目录文件的关键块的 \$0004 字节处开始,紧跟在两个指针之后。

它的内部结构与一个卷目录标题很相似。它有十四个字段,包含所有有关子目录的主要资料。下图显示了一个子目录标题的结构。

存储类型 and 名字长度(1字节):两个四位元字段组成此字节。在高四位(存储类型)的 \$E 值表示当前块为一个子目录的关键块。低四位包含子目录名的长

度。名字长度可由 RENAME 调用改变。

文件名(15 字节):此字段包含子目录的名字。此字段可由 RENAME 调用改变。

保留(8 字节):保留给文件系统未来扩展。

建立部分(4 字节):子目录被建立时的日期和时间。这些字节的格式在后面描述。

版本号(1 字节):子目录被建立时的 ProDOS 的版本号。这字节允许新版的 ProDOS 确定子目录的格式,并相应调整目录分析。ProDOS 1.1.1:版号=0

最小版号(1 字节):可以存取这子目录内资料的 ProDOS 的最低版号。这字节允许低版本的 ProDOS 确定它们是否能存取较新的子目录。ProDOS 1.1.1:最低版号=0

存取标记(1 字节):确定这子目录是否能被读取、写入、删除和改名,和文件是否需要备份。这字段的格式在后面描述。一个子目录的存取标记字节可由 SET-FILE.INFO 调用改变。

项目长度(1 字节):子目录内每个项目的字节长度。ProDOS 1.1.1:项目长度= \$ 27

每块项目数(1 字节):目录文件的每个块内的项目数量。ProDOS 1.1.1:每块项目数= \$ 0D

字段长度	相对字节	
1 字节	\$ 04	存储类型
15 字节	\$ 05	名字长度
8 字节	\$ 14	文件名
4 字节	\$ 1C	保留
1 字节	\$ 20	建立日期 建立时间
1 字节	\$ 21	版本号
1 字节	\$ 22	最小版本号
1 字节	\$ 23	存取标记
1 字节	\$ 24	项目长度
1 字节	\$ 25	每块项目数
2 字节	\$ 27	现有文件数
2 字节	\$ 29	根指针
1 字节	\$ 2A	根项目号
1 字节		根项目长度

文件数(2 字节):子目录文件内现有文件项目的数量。一个现有文件是存储类型非零的文件。

父指针(2 字节):包含这子目录项目的目录文件块的块地址。这两字节指针,低字节在前,高字节在后。

父项目号(1 字节):父指针指定的块内的此子目录的项目号。

父项目长度(1 字节):拥有此子目录文件的目录

项目的长度。注意,用最后 3 个字节你可以计算此子目录的文件项目在一个卷上的准确位置。ProDOS 1.1.1:父项目长度= \$ 27

#### 4. 文件项目

在一个目录文件的任何块内,紧接指针后面是一系列登记项。目录文件的关键块内的第一个项目是一个标题;所有其它登记项是文件项目。每个项目的长度记录在目录的项目长度字段内,一个文件项目包含描述和指向一个单个子目录文件或标准文件的信息。

在一个目录文件内的一个项目可能是有效的(active)或无效的(inactive);也就是说,它可能或不可能描述一个当前在目录内的文件。如果它无效,那么项目的第一字节(存储类型和名字长度)为零值。

在一个目录块内项目的最大数量(包括标题)被记录在那个目录的标题的“每块项目数”字段内。有效的文件项目的总数(不包括标题)被记录在那个目录的标题的“文件数”字段内。下图描述一个文件项目的格式。

字段长度	相对字节	
1 字节	\$ 00	存储类型
1 字节	\$ 01	名字长度
15 字节	\$ 10	文件名
1 字节	\$ 11	文件类型
2 字节	\$ 13	关键块指针
2 字节	\$ 15	使用块数
3 字节	\$ 18	EOF
4 字节	\$ 1C	建立日期 建立时间
1 字节	\$ 1D	版本号
1 字节	\$ 1E	最小版本号
1 字节	\$ 1F	存取标记
2 字节	\$ 21	辅助类型
4 字节	\$ 25	最后修改日期 和时间
2 字节	\$ 29	标题指针

存储类型及名字长度(1 字节):两个四位元字段被压缩到此字节内。在高四位内的值(存储类型)指定被此文件项目指向的文件的类型。值 \$ 1, \$ 2, \$ 3 和 \$ D 分别表示树苗型,幼树型,树型和子目录文件。树苗型,幼树型和树型文件是标准文件的三种格式。在后面将详细描述这三种文件。低四位包含文件名字的长度,它可由 RENAME 调用改变。

文件名(15 字节):此字段包含文件的名字。它可由 RENAME 调用改变。

(下转第 18 页)



## 第四讲 FORTH 中的数

北京 FORTH 应用研究会(100034) 丁志伟

这一讲介绍 FORTH 中数据的计算、变量的用法、单精度数的用法等问题。这方面内容较多,因此介绍只能是较粗略的。

• FORTH 中的运算 •

前几讲中介绍过的一些算术运算,都比较简单。下面介绍比较复杂的运算如何进行。

已经知道,3 与 5 两数相加,用逆波兰表示为:

```
3 5 +
```

这是两个数相加。现在有 3、5、7 三个数要相加该怎样做呢?可以这样看:3 与 5 相加等于 8,8 再与 7 相加等于 15。整个过程就是 $(3+5)+7$ 。于是逆波兰表达式就可以写作:

```
3 5 + 7 +
```

看一下栈状态:

```
3 5 ( -3 5)
```

```
+ ( 3 5 --8)
```

```
7 ( 8 --8 7)
```

```
+ ( 8 7 --15)
```

再看看  $3+5\times 7$  该怎样做。这回要先做 5 与 7 相乘,再与 3 相加。用逆波兰表示就是

```
3 5 7 * +
```

请看栈注释:

```
3 5 7 ( --3 5 7)
```

```
* ( 3 5 7 --3 35)
```

```
+ ( 3 35 --38)
```

相应地,3(5-7)的逆波兰式子则为

```
3 5 7 - *
```

在逆波兰表达式中,并没有出现象括号这样的额外符号,也没有涉及到象先乘除后加减这样的运算优先级概念。虽然与人们的习惯不一样,但它简单、不会产生歧义,便于计算机处理。有些编译程序就是把逆波兰做为高级语言和机器语言之间的中间表示法的。

再看一个稍复杂些的例子。比如有些计算中, $(a+b)/(a-b)$  会反复出现,就可以定义一个词:

```
: CAL2 OVER OVER + ROT ROT
- / ;
```

程序中两个 OVER 用来复制 a 和 b。做过加法之后,栈顶是 a+b,两个 ROT 用来把 a、b 移到栈顶,做减法之后,栈中有两个数,分别是 a+b 和 a-b,然后做除法。请读者自己做一下栈注释,以加深理解。

再举一个例子,定义一个名为  $N+M=L$  的词,如果输入

```
5 6 N+M=L
```

要显示出  $5+6=11$

问题有些复杂,不妨分步编程。先定义一个词 N+

M

```
: N+M SWAP . ." + " . ;
```

其中“+”是显示出加号。因为后进先出,要先用 SWAP 交换两数位置。试一下:

```
8 3 N+M
```

显示出 8+3

接下来可以定义出  $N+M=L$

```
: +M=L OVER OVER N+M ."
="
```

```
+ . CR ;
```

由于 N+M 和加法会分别消耗掉两个数,因此要事先用两个 OVER 把两数复制一遍。看看程序是否符合要求:

```
7 4 N+M=L
```

显示 7+4=11

当然,也可以不先定义出 N+M,直接用展开式定义  $N+M=L$ 。象下面这样:

```
: N+M=L OVER OVER SWAP
```

```
. ." + " . ." = " + . CR ;
```

• 常数和变量 •

FORTH 使用的数据,一般放在堆栈上,用于词与词之间的参数传递,能使单词的接口简单,因而定义新词方便,扩充系统易于进行。这是 FORTH 的特点之一。但如果只使用堆栈,就好象一个工厂只有加工车间而无仓库,因此 FORTH 也使用变量和常数。另外, FORTH 语言还可以定义出各种各样的数据结构,用来满足对数据处理的多方面要求。

使用常数和变量,需要事先定义。定义出的变量和常数,也是词典中的单词。系统的词典可以容纳各种成分。先看常数。其定义方法是

```
n CONSTANT name (对于 PC/FORTH)
```

```
n CONST name (对于 D1.0 版本)
```

定义之后,引用 name 时,堆栈中会留下 n 这个数。作为常数,它是不能改变的。

比如定义一个常数 S/M,它是每分钟秒数。

```
60 CONSTANT S/M
```

```
S/M .
```

显示出 60

利用 S/M 定义一个单词,可以把分钟化为秒。

```
: M>S DUP . ." minutes = "
```

```
S/M * . ." seconds." CR ;
```

7 M>S

会显示出 7 minutes = 420 seconds。  
(7分钟=420秒)

变量比起常数更常用,使用方法稍复杂些。先看变量的定义方法。对于 PC/FORTH,要用

VARIABLE name

对于 D1.0 版本,有两种定义方法。

① n VARIABLE name

n 是给变量赋的初值,一般用 0。

② VAR name

这种形式相当于 PC/FORTH 中的 VARIABLE,定义后 name 的初值为 0。实际上,D1.0 版本中的 VAR 是由 VARIABLE 定义出来的。

: VAR 0 VARIABLE ;

再看看变量的用法。如果编一个温度监测程序,可能需要定义出一个变量 T,用来保存测得的温度。

VARIABLE T

T ( --a) 引用变量时,系统会把变量所在地址放到堆栈上,并不把变量值取出。

向变量中存一个数,要用叹号“!”。

! ( n a--) 把 n 存入 a 地址中去。

比如测得的温度是 25 度,就可以用:

25 T !

想从变量中取出所存的值是用 @。

@ ( a--n) 把 a 地址中的数取出来放到堆栈上。

想看看 T 中是什么数,就用

T @ .

T 向堆栈提供一个地址 a,@ 将其内容取出。因此上边一行操作会显示出刚存入的 25。

如果把一个变量比做一个信箱,每个信箱有其位置(地址 a),叹号就象把一个数当做一封信投入这个信箱。@ 则看成用来打开信箱的钥匙,能将其内容取出。

D1.0 版中有一个词“?”,问号。这个词用来显示出一个变量中的数。其定义是:

: ? @ U. CR ;

前边的变量 T,第二次测量时比第 1 次升高了 2 度,该怎么做呢? 可以这样:

2 T @ + T !

虽然能满足要求,却有些麻烦。词典中有一个词 +!

+! ( n a--) 把 n 加到 a 的内容中。

上述操作就可以改成

2 T +!

这样操作较为简单、直观,速度也快些。

词典中有一些供系统使用的变量。读者已经知道,FORTH 语言有一个功能比其他语言方便,就是用一命令可以改变系统进位制。比如 HEX 和 DECIMAL (或 DEC)能分别把系统设置成 16 进制和 10 进制。改

变进位制,就是通过变量进行的。

系统中有一个变量名为 BASE,存放着当前进位制数基。数基是几,系统就处于几进制状态。请看 HEX 和 DECIMAL 的定义:

: HEX 16 BASE ! ;

: DECIMAL 10 BASE ! ; (对于 PC/FORTH)

: DEC 10 BASE ! ; (对于 D1.0)

设定系统进位制原来如此简单。请注意,定义中的 16 和 10 都是用 10 进制表示的,如果在 16 进制下,就分别是 10 和 A。当然,还可以把系统改变为其他进位制,比如二进制等。

在 FORTH 中,对数据做不同进位制的转换很方便。比如字母 A 的 ASCII 码是 16 进制的 41,想知道它的 10 进制是多少,就可以:

HEX 41 DECIMAL .

显示出 65。

很多读者都有体会,对于 BASIC 和大多数高级语言,做这种变换就麻烦多了。

• FORTH 中的数 •

FORTH 中数据的基本形式,是单精度数。这种数据处理方便,运行效率较高。

一个单精度数在内存中占两个字节,可以看作带符号数,也可以看作无符号数。对于带符号数,可以表示的数值范围对于 10 进制是 -32768~+32767,对于 16 进制是 -8000H~+7FFFH。与大多数 BASIC 语言中整型数相同。

下面的小程序,可以测出系统单精度数的最大值。

: TEST1 1 BEGIN DUP DUP 1 + <

WHILE 1 + REPEAT . ;

这是一个循环程序,关于循环,以后会介绍的。

程序将 1 放在堆栈,然后不断将栈中数加 1,如果达到可表示的最大值,加 1 就会超出它可以表示的范围,数值就会溢出,程序终止,然后显示出这个最大值。

执行 TEST1,稍等一会,在 10 进制状态下,屏幕上会显示出 32767,说明程序最大值是 32767(16 进制下为 7FFF)。

把程序稍加改变,能测出最小值。

: TEST2 1 BEGIN DUP DUP 1 - >

WHILE 1 - REPEAT . ;

执行 TEST2 的结果,是 10 进制的 -32768 或 16 进制的 -8000。

对于某些数,比如内存地址,是不出现负值的。这时可以把栈中数当做无符号数处理,其范围是 0~65535。在 FORTH 中,有无符号数在内存中是没有区别的,差别是对数据的解释不同。请看:

32767 1 + DUP .

显示 -32768,这是带符号数,大于 32767 时就无法正确表示了。栈中还剩下一个数。

U. ✓

显示 32768。这是无符号数,大于 32767 而又小于 65536 的数是可以正确表示的。上边的 32767 加 1,对于点单词“.”,认为它是带符号数,而对于“U.”(无符号显示词),则认为它是无符号数,因而显示结果不同。

限于篇幅,无法对有无符号数做更深入的讨论,这涉及到计算机基础知识,请读者参考有关资料。

无论有无符号,两个字节所能表示的范围可能太小。有几种解决办法,如利用某些特殊单词、用双精度数、用浮点数等等。下面重点介绍特殊单词。

数值范围小的问题在做乘法时比较突出。对于乘法,词典中有两个词,分别是\*/和MOD。与之相近的还有\*/MOD和UM/MOD等。

$*/ (n_1 n_2 n_3 \dots (n_1 \times n_2) / n_3)$

先把  $n_1$  与  $n_2$  相乘,再除以  $n_3$

这个词把  $n_1$  乘以  $n_2$  做为中间值,是作为双字长(4字节)数出现的,因而它的精度较高。可以利用它定义一个百分数词。

: % 100 \*/ ;

比如一个学校共有 320 位学生,其中男生占 55%,求有多少位男生,就可以

320 55 % . ✓

答案是 176 位男同学。

再举一例。一个直径为 5000 的圆环,其周长是多少?求周长的公式是:

$L = \pi D$

L 是周长,  $\pi$  是圆周率, D 是直径。

$\pi$  值是 3.14159..., 单精度数是无法表示的。这里使用分数 355/113, 这是祖冲之用来表示  $\pi$  值的分数“密率”。于是公式就可以改成:

$L = \frac{355}{113} D$

利用 \*/ , 逆波兰表示就是

5000 355 113 \*/

5000 先乘以 355, 再除以 113, 其结果是 15707。

假如不用 \*/ 而分别用 \* 和 /, 就该是这样:

5000 355 113 /

结果是 48.5000 与 355 相乘用二字节数已无法表示, 因而结果会出现错误。

上一讲介绍过词典中还有一个单词名为/MOD。它的功能是除法, 得出商和余数。对余数继续除, 就可以计算出小数。

比如计算 10/7, 可以

10 7 /MOD . ✓

显示 1, 这是商的整数部分。栈中还留下一个余数 3, 把余数乘以 10 再除以 7, 就会得出第 1 位小数。

10 \* 7 /MOD . ✓

显示 4, 这是第 1 位小数, 此时栈中会留下一个余数 2, 再乘以 10, 除以 7。

10 \* 7 /MOD . ✓

显示 2, 这样的步骤可以一直继续下去。结果是 1.

这种除法步骤较多, 比较麻烦, 如果用循环编一个程序, 就可以简单得多。

可以看出, 虽然 FORTH 使用单精度数, 仍然可以用不太复杂的方法提高运算精度。

当然, \*/和/MOD 并不能全部解决精度问题。这时就应该考虑使用双精度数。PC/FORTH 中和大多数 83 标准的系统, 都具有双精度数功能。

双精度数在内存中占 4 个字节, 在堆栈中占两个单精度数的位置。其表示范围在 -2147483648 ~ +2147483647, 达到 10 位 10 进制, 这对大多数应用来说已经足够了。

输入双精度数, 要在数后边加上一个小数点。如: 12345. ✓

这个数小于 65536, 系统会把一个 12345 和一个 0 分别入栈。如果用点单词,

. ✓  
会显示出 0。再

. ✓  
才显示出 12345。

正确显示出一个双精度数, 要用“D.”, 点前边的字母 D, 表示对双精度操作。

12345. D. ✓

显示 12345。

加法要用 D+:

11223344. 88776655. D+ D. ✓

显示 99999999

供双精度数操作作用的词还有: D\*、D.R、DMAX 等等。

双精度数还可以做为小数使用, 或者按 05:16:03.92/10/1 等各种形式显示, 非常灵活。

关于 FORTH 中的数, 还有许多问题, 限于篇幅, 无法一一介绍。下面把这讲新遇到的一些词归纳一下。

CONSTANT 或 CONST 定义常数

VARIABLE 或 VAR 定义变量

! ( n a-- ) 将 n 存入 a 地址

@ ( a--n ) 将 a 地址中数取出

? ( a-- ) 显示 a 地址中内容

HEX ( -- ) 将系统设置为 16 进制

DECIMAL 或 DEC ( -- ) 设置成 10 进制

U. ( U-- ) 将堆栈中数做为无符号数显示

\*/ ( n<sub>1</sub> n<sub>2</sub> n<sub>3</sub>...n<sub>3</sub> ) n<sub>1</sub> 乘以 n<sub>2</sub> 再除以 n<sub>3</sub>

D. ( n<sub>1</sub> n<sub>2</sub>-- ) 将 n<sub>1</sub>、n<sub>2</sub> 作为双精度数显示

D+ ( n<sub>1</sub> n<sub>2</sub> n<sub>3</sub> n<sub>4</sub>...n<sub>5</sub> n<sub>6</sub> ) 双精度加法

有需要讲座中介绍的 APPLE][FORTH D1.0 软件者, 请将款汇到编辑部。软件包括一张软盘和一本说明书。定价 35 元(含邮挂费), 如有其他问题, 请与作者联系。汇款地址: 北京 173 信箱 邮编 100036[电子与电脑]编辑部张丽收。



## 基 本 算 法

北京航空航天大学计算机中心(100083)李宁

### 一排序(Sorting)

在计算机应用的各个方面,有一项很重要的工作就是将数据按某种要求的顺序排列好,以便对这些数据进行下一步处理。比如考生的分数要从高到低排好,这样才便于录取;实验测得的数据要按测量时间的先后排好,这样才便于比较。

表一中是一张工资表,列出了若干位职工的编号、姓名和工资,各位职工领了工资后应在自己名字后面签个字。若凭这张表发工资,职工们都会感到不方便,因为他们都要费点劲才能找到自己的名字。若这张表较长,那费的劲就很大了。根本的原因就是这张表对编号、姓名和工资这些数据项都没有顺序。若这张表对某个数据项(比如编号)有顺序(比如从小到大排),那发工资来就方便多了。排好顺序对于查找数据有重要意义,另外,对分析、比较和统计等工作也有重要意义。

表一 编号 姓名 工资	表二 编号 姓名 工资
0421 周川 122.00	0037 刘文 105.00
0186 陈伟 113.00	0186 陈伟 113.00
0295 杨光 98.00	0295 杨光 98.00
0037 刘文 105.00	0421 周川 122.00

我们把工资表按编号排一下序,排完以后如表二所示。这张表就是一个被排序的序列,表中的每一行是这个序列的一个元素,编号这个数据项则是这次排序的“关键字”。排序就是将若干个元素组成的任意序列按某个关键字重新排列成有序的序列。不是任何数据项都可以作为关键字的,比如复数数据项就不行,因为复数不能比较大小。另外,关键字的值谁大谁小(即顺序)是应该有标准的。象工资,200元比100元大,这在人们的常识里和计算机里都是这样规定的,所以工资这个数据项可以作为关键字。象姓名,美国人的姓名可由英文字母顺序排先后,中国人的姓名则可由汉字的笔划数或汉字的汉语拼音字母顺序排先后,但是在计算机里可没有遵循这些规定。使用美国标准信息交换码(ASCII码)的计算机里规定:‘A’<…<‘Z’<‘a’<…<‘z’,因此‘MacDonald’<‘Macaulay’;汉字的顺序则是根据它们在国家标准汉字字符集中的位置,靠前者较小,因此‘虞’<‘俞’<‘荀’。这样一来,若用姓名这一数据项作为关键字时就要小心,计算机排出的顺序与人们通常希望的顺序可能不一样。如果你需要一种特殊的顺序,象中国人的姓名按笔划数排,那就必须在表中加一个辅助的数据项“笔划数”作为关键字。

排序的算法有许多种,但都离不开“比较”和“交

换”这两个基本操作。排序时一般都是从头开始,按照某种规则在被排序列中取出两个元素,先比较一下它们的关键字的值,若是“逆序”(即与规定的顺序标准相反),就把这两个元素的值交换一下,一直做到尾,这叫排了一趟。一般都要经过很多趟的多次比较和交换才能将序排好。我们对那张工资表以编号作为关键字进行排序,被比较的是两个职工的编号,但交换的时候,则是把该职工的编号、姓名和工资这一行数据作为一个单位(元素)去交换值的。从这里可以看到序列的关键字和序列的元素之间是有区别的。下面我们讨论排序算法时,假定被排序列是有  $m$  个元素的整数序列,即序列中的每一个元素只是一个整数,因此被比较的关键字是整数,被交换的元素也是整数,这将使算法的表述和所给的程序清晰简单,但这并不改变算法的本质。当要排工资表这样有多个数据项的序列时,只要对程序稍加修改就行了。下面讨论三种典型的排序算法。

#### 1 简单选择排序(Simple Selection Sort)

这种算法很简单。第一趟从第一个元素开始,逐个与后面的元素比较,找到值最小的那个元素,然后交换此元素与第一个元素的值,这样第一个元素的值就是整个序列中最小的了;第二趟从第二个元素开始,逐个与后面的元素比较,找到值最小的那个元素,然后交换此元素与第二个元素的值,这样第二个元素的值就是整个序列中第二小的了;…;第  $i$  趟从第  $i$  个元素开始,逐个与后面的元素比较,找到值最小的那个元素,然后交换此元素与第  $i$  个元素的值,这样第  $i$  个元素的值就是整个序列中第  $i$  小的了;…;一共做  $m-1$  趟,整个序列就从小到大排好了。

表三中列出了对一序列用这种算法排序的过程,每一行中有括号的两个元素曾交换过值。用C语言写的函数 Select 用于实现这个算法,内层循环控制依次向后进行比较,并将值最小的那个元素的下标用  $k$  记下来,这层循环依次要做  $(m-1)$ ,  $(m-2)$ , …,  $1$  次;外层循环控制从哪个元素开始向后进行比较,并调用函数 Xchang 交换此元素与找到的值最小的那个元素的值,这层循环要做  $m-1$  次。这里要注意,C语言中数组的下标都从 0 开始,第  $i$  个元素的下标是  $i-1$ 。函数 Xchang 用于交换两个元素的值,它要占用一个辅助空间  $t$ ,这个函数以后还要用到。

表三 元素的下标	0	1	2	3
初始被排序列( $m=4$ )	122	113	98	105
第1趟从第1个开始找最小的			↑	

交换其与第 1 个的值 (98) 113 (122) 105  
 第 2 趟从第 2 个开始找最小的 ↑  
 交换其与第 2 个的值 98 (105) 122 (113)  
 第 3 趟从第 3 个开始找最小的 ↑  
 交换其与第 3 个的值 98 105 (113) (122)

简单选择排序的特点是每一趟中被比较的两个元素间的“距离”是逐渐加大的,先与邻近的比,再与远处的比,要经过 $(m-1)m/2$ 次比较才能将序列排好,因此说平均排序时间是 $m^2$ 量级的(即平均排序时间基本与 $m^2$ 成正比),记作 $O(m^2)$ 。显然,随着被排序列中元素个数的增多,所要花的平均排序时间增长得很快,因此有必要寻找速度更快的排序算法。

### 2. shell 排序(Shell's Sorting Method)

这种算法是 1959 年由 D. L. Shell 提出的,其特点是每一趟排序中被比较的元素间的距离是逐渐减小的。具体做法是:开始时第一个元素  $aa[0]$  不是与它邻近的元素比较,而是与和它隔  $k$  个元素的元素  $aa[k]$  比较,若小的在后,就交换它们的值,接着第二个元素

$aa[1]$ 与元素  $aa[1+k]$ 比较,若小的在后,就交换它们的值,……,一直做到第  $n$  个元素为止( $n+k=m$ ),这就排了一趟。对某一趟排序, $k$  是固定的,若这一趟中没有发生过交换值的情况,则  $k$  不变去进行下一趟排序,若这一趟中发生过交换值的情况,则  $k$  按某种规律减少一些再进行下一趟排序。显然当  $k$  减为 1 时,若某一趟排序中没有发生过交换值的情况,整个序列就从小到大排好了。 $k$  的递减规律有多种,一般第一趟取  $k=m/2$ ,以后  $k$  逐次减半,由于  $k$  是整数,所以总能达到  $k=1$  的情况。

表四中列出了对一序列用这种算法排序的过程,每一行中有相同括号的两个元素曾交换过值。用 C 语言写的函数 Shells 用于实现这种算法,内层循环控制哪两个元素相比较,若小的在后,就调用函数 Xchang 交换它们的值,并令  $g=1$  记住这一趟中曾发生过交换值的情况;中层循环根据  $g$  决定是否用当前的  $k$  值进行下一趟排序;外层循环控制  $k$  的递减。这种算法也需要一个辅助空间用于交换两个元素的值。

表四

元素的下标	0	1	2	3	4	5	6	7	8	交换次数
初始被排序列( $m=9$ )	5	7	6	4	9	1	3	2	8	
第 1 趟, $k=m/2=9/2=4$										
第 1 次 $aa[0]$ 与 $aa[k]$ 比较	↑				↑					
∴"顺序",∴不交换值	5	7	6	4	9	1	3	2	8	
第 2 次 $aa[1]$ 与 $aa[1+k]$ 比较		↑				↑				
∴"逆序",∴交换值	5	1	6	4	9	7	3	2	8	
第 3 次 $aa[2]$ 与 $aa[2+k]$ 比较			↑				↑			
∴"逆序",∴交换值	5	1	3	4	9	7	6	2	8	
第 4 次 $aa[3]$ 与 $aa[3+k]$ 比较				↑				↑		
∴"逆序",∴交换值	5	1	3	2	9	7	6	4	8	
第 5 次 $aa[4]$ 与 $aa[4+k]$ 比较					↑				↑	
∴"逆序",∴交换值	5	1	3	2	8	7	6	4	9	
第 1 趟排完	5	(1)	(3)	[2]	{8}	<7>	(6)	[4]	{9}	4
第 2 趟, $k=4$	5	1	3	2	8	7	6	4	9	0
第 3 趟, $k=k/2=4/2=2$	<3>	1	<5>	2	(6)	4	(8)	7	9	2
第 4 趟, $k=2$	3	1	5	2	6	4	8	7	9	0
第 5 趟, $k=k/2=2/2=1$	<1>	<3>	(2)	(5)	[4]	[6]	{7}	{8}	9	4
第 6 趟, $k=1$	1	<2>	<3>	(4)	(5)	6	7	8	9	2
第 7 趟, $k=1$	1	2	3	4	5	6	7	8	9	0

Shell 排序的平均排序时间从理论上很难估计,但大量的实际运行可以证明它是 $m^{1.3}$ 量级的。由于先在大范围里排得大致有序以后才在小范围里排,所以小范围里的交换次数就会减少,因此 Shell 排序的速度要比简单选则排序快得多, $m$  越大快得越多。还有更快的排序算法吗?有!

### 3. 快速排序(Quick Sort)

这种排序算法是 1962 年由 C. A. R. Horare 提出的,其特点是对被排序列采用分而治之的策略。具体做法是:每一趟排序前都在被排序列中选出一个元素作为标准(aak),根据它将被排序列分成两个小序列,使

前一个小序列中的所有元素都不大于 aak,使后一个小序列中的所有元素都不小于 aak。然后分别对这两个小序列再选 aak 和再分,相应产生四个更小的序列,……到分出的每一个小序列中都只有一个元素时,整个序列就从小到大排好了。显然,这是一个递归算法。选出的 aak 对排序速度有影响,从理论上讲,当它使被分成的两个小序列中的元素个数相等(或差一)时,排序速度最快,但实际上预先选出这样的 aak 与排序的目的是有矛盾的(选出这样的 aak 显然要求被排序列已经有序),一个简单方法是取被排序列的第一个元素作为 aak。

怎样根据 aak 将被排序列分成上述两个小序列呢?请参见表五中的具体实例。先设两个指针 i 和 j,开始时分别指向被排序列的第一个(即 aak,加括号以示区别)和最后一个元素。然后先前移 j,停在找到的第一个小于等于 aak 的元素处,交换 i 和 j 所指的这两个元素的值,同时 i 后移一个元素,此时 j 指的就是 aak;然后再后移 i,停在找到的第一个大于等于 aak 的元素处,交换 i 和 j 所指的这两个元素的值,同时 j 前移一个元素,此时 i 指的就是 aak;...一直做到  $i \geq j$  为止,

此时 aak 前边的和后边的元素就组成了上述的两个小序列。

用 C 语言写的函数 Qksort 用于实现这个算法,把它改写成非递归调用的程序也可以。第一个内层循环用于移动指针 j,第二个内层循环用于移动指针 i,外层循环控制分两个小序列的工作何时结束。另外,由于每次交换都是与 aak 进行的,所以交换只需一次赋值,再用 k 记下 aak 应在的位置,到每一趟排序的最后将 aak 放到它应在的位置上就可以了,这也加快了排序速度。

表五

元素的下标	0	1	2	3	4	5	6	7
aak=73,开始时 i=0,j=7	(73)	91	72	11	75	23	14	11
前移 j 找到(=aak 的元素(j=7))	i							j←
交换 i 和 j 所指的元素的值	11	91	72	11	75	23	14	(73)
后移 i 找到(=aak 的元素(i=1))		→i						j
交换 i 和 j 所指的元素的值	11	(73)	72	11	75	23	14	91
前移 j 找到(=aak 的元素(j=6))		i						j←
交换 i 和 j 所指的元素的值	11	14	72	11	75	23	(73)	91
后移 i 找到(=aak 的元素(i=4))			→		i		j	
交换 i 和 j 所指的元素的值	11	14	72	11	(73)	23	75	91
前移 j 找到(=aak 的元素(j=5))					i	j←		
交换 i 和 j 所指的元素的值	11	14	72	11	23	(73)	75	91
后移 i 找到(=aak 的元素(i=6))						→j	i	
∴i>j,∴不交换	11	14	72	11	23	(73)	75	91
排完一趟,下面排后一小序列								
aak=75,开始时 i=6,j=7						(75)	91	
前移 j 找到(=aak 的元素						ij	←	
91i)>j,∴不交换								(75)
排完一趟,只分出后一小序列								91
只有一个元素,不用再排								
(请读者完成排前一小序列)								

理论上可以证明快速排序的平均排序时间是  $m \log_2 m$  量级的,显然比 Shell 排序还快得多。实际上快速排序是常用排序算法中速度最快的。但是当初始被排序列基本有序时,取被排序列的第一个元素作为 aak 的快速排序是较慢的( $m^2$  量级),改进的方法是取被排序列的第一个、中间一个和最后一个元素当中值居中的那个作为 aak,而且仍要将 aak 放在被排序列的第一个。另外,快速排序是个递归过程,所以需要若干额外的空间作为栈(C 程序会自动分配),在最坏的情况下,要  $m$  个额外的空间,这是个不小的数目(与被排序列的元素一样多),改进的方法是除尽量使分成的两个小序列长度相近外,还应该对长度较小的那个小序列先进行排序。

#### 4 排序算法小结

1) 比较和交换是排序算法的两个基本操作,比较和交换的次数决定了排序算法的速度。一般一次比较所花的时间比一次交换所花的时间少,但排序过程中比较的次数比交换的次数多。

2) 各种排序算法在时间和空间上各有利弊,且往往是时间上占优的算法空间上显劣,反之亦然,所以

要根据具体情况选用,甚至可将几种排序算法结合起来使用。

3) 对于同一随机整数序列,在 386 微机上用上面的三种算法排序所用的时间(秒)如下(供参考):

表六

	平均时间	最坏情况	m=2048	m=4096
简单选择排序	$O(m^2)$	$O(m^2)$	4.40	17.64
Shell 排序	$O(m^{1.3})$	$O(m^{1.5})$	0.88	2.58
快速排序	$O(m \log_2 m)$	$O(m^2)$	0.05	0.11

## 二 查找(Searching)

查找也叫检索或搜索,比如要在工资表中查找姓名是'杨光'的人,那么姓名这个数据项就是这次查找的关键字。显然,作为查找关键字的数据项不必非得能比较大,只要能比较是否相等就可以了,因此复数数据项可以作为查找的关键字。查找是要在若干元素所组成的序列中查找关键字等于给定值的那个元素。找到时称查找成功,要给出该元素在序列中的位置;若没找到则称查找不成功,要给出没找到的信息(比如-1)。下面可以看到,当被查序列按查找关键字有序时,

查找速度可以快得多。常用的查找算法有下面两种,和上面的原因一样,我们假定被查序列是有  $m$  个元素的整数序列。

### 1 顺序查找(Sequential Search)

顺序查找也叫线性查找,算法很简单,只要从某个元素(通常是第一个)开始,逐个比较各元素的值与给定值是否相等就可以了。有相等的情况时,查找成功;若比较到最后一个元素也没有相等的情况,则查找不成功。若被查序列已从小到大排好了序,则只要查到大于给定值的那个元素时,就可以知道查找不成功,后面的元素就可以不查了。

用 C 语言写的函数 `Sequen` 用于实现这个算法,但是没有考虑被查序列已排序的情况。查找成功时给出元素的下标,查找不成功时给出 -1。

顺序查找平均要查  $m/2$  个元素,因此说平均查找时间是  $m$  量级的(即平均查找时间基本与  $m$  成正比),记作  $O(m)$ 。当  $m$  较大时时间效率是较低的,因此有必要寻找速度更快的查找算法。

### 2 对分查找(Binary Search)

对分查找也叫折半查找,它要求被查序列必须是已从小到大排好序的。查找开始时先将被查序列中间的那个元素的值与给定值比较,若等于则查找成功;若大于则将该元素前面的所有元素当做被查序列继续查找;若小于则将该元素后面的所有元素当做被查序列继续查找,……,一直做到被查序列中只有一个元素时还没有相等的情况出现,则查找不成功。显然,这也是一个递归算法。

用 C 语言写的函数 `Binary` 用于实现这个算法,把它改写成非递归调用的程序较容易。查找成功时给出元素的下标,查找不成功时给出 -1。

对分查找平均要查  $(\log_2 m)/2$  个元素,因此说平均查找时间是  $\log_2 m$  量级的,显然比顺序查找快得多。

### 3 查找算法小结

1) 比较是查找算法的基本操作,比较次数越少,查找速度越快。

2) 对同一  $m$  个元素的随机整数序列  $M$  和同一  $k$  个元素的随机整数序列  $K$ ,在 386 微机上用上面的两种算法在  $M$  中查找  $K$  中的各个元素所用的时间(秒)如下(供参考):

表七	平均时间	$m=2048$ $K=2048$	$m=4096$ $K=1024$
顺序查找	$O(m)$	2.64	4.51
对分查找	$O(\log_2 m)$	0.11	0.05

### 三合并(Merging)

合并也叫归并,它是将两个有序序列合成一个有序序列的过程。如果先将一个有序序列接在另一个有序序列之后,然后再排序形成一个大的有序序列,所花的时间要比利用合并慢得多。有时要排序的序列非常大,以致计算机的内存放不下(只能放在外存里),这时可将被排序列先分成若干段,把这些小序列排好序后

再用合并的算法将它们合成大的有序序列,当然此时要借助计算机的外存来工作。和上面的原因一样,我们假定被合并序列是整数序列。

合并算法利用了被合序列已从小到大的有序特征,所以是一个单向前进的过程。设两个被合序列已分别放在了数组 `aa`(有  $m$  个元素)和 `bb`(有  $n$  个元素)中,另外准备一个数组 `cc` 放合并后的序列(至少能放  $m+n$  个元素)。先比较 `aa[0]` 和 `bb[0]`,将较小的放入 `cc[0]`。若放的是 `aa[0]`,就再取 `aa[1]` 与 `bb[0]` 比较;若放的是 `bb[0]`,就再取 `bb[1]` 与 `aa[0]` 比较,将较小的放入 `cc[1]`。……以后都取一个新的元素进行比较,将较小的放入 `cc`,一直到 `aa` 或 `bb` 中的元素被取完为止,再把 `bb` 或 `aa` 中剩下的元素依次放入 `cc`。

用 C 语言写的函数 `Merge` 用于实现这个算法。

## Turbo Pascal V6.0 介绍

洛阳铁路分局 欧阳慎

当广大 PC 用户正在细细体味 Borland 公司 Turbo 系列语言软件产品的独具特色时,该公司却令人目不暇接地推出更新的版本。现在就为大家介绍 Turbo Pascal 的新版本 V6.0。

### 一、Turbo Pascal v 6.0 新增特性:

1. 新的集成开发环境 (IDE),除了保留原有的用户界面外,另增:

\* 鼠标支持(需要在 DOS 中配置与 Microsoft 6. x 及以上版鼠标驱动程序或相兼容的驱动程序);

\* 可覆盖的多窗口;

\* 多文件编辑器,可一次编辑多个文件总长不超过 1M 字节;

\* 使用书写板 (clip board) 切块、粘贴、复制文本;

\* 扩充了联机求助;

\* 可设置条件断点;

\* 含 CPU 寄存器窗口;

\* 充分使用扩充内存 EMS,以提高性能和增加容量;

\* 可内定色彩及启动时的选项。

2. 面向对象的高性能应用结构和库 Turbo Vision 用 Turbo Vision 编写的程序将继承所有特性,如鼠标支持、对话框、菜单、可覆盖窗口以及自动的桌面管理(联机求助、计算器、其他工具)。据称 Turbo Pascal v6.0 的集成开发环境就用 Turbo Vision 编写而成;

3. 可嵌入汇编语言

可直接在 Pascal 源程序内嵌入汇编语言指令;

4. 对象说明中增加支持私用的域 (fields) 和方法 (methods)



## 5. 其他

- \* 新的对象库单元(Objects Unit);
- \* 更快、更安全的堆管理;
- \* 增加过程指示符:far, near 和 assembler;
- \* 增加编译指示符;

{ \$x } 放弃函数结果

{ \$G } 产生 80286 代码

{ \$L } 从 .OBJ 文件中链入初始化数据;

- \* 生成更短的代码

### 二、与低版本和其他软件的兼容性:

1. Turbo Pascal v5.5 的所有程序只要在 6.0 下重新编译就可执行,但 5.5 版的对象库单元不能和 6.0 的 Turbo Vision 同用;

2. 4.0 版的工具箱(Toolboxes)可和 6.0 同用;

3. 可和任意版的 Turbo Debugger 同用,但由于堆管理的改进,为了保证对这部份操作的正确性,应使用 Turbo Debugger v2.01 成更高版。

### 三、安装 6.0 版

1. 文件列表

6.0 版源盘为二张高密度盘(1.2M)

其中:

盘 1: Install, Compiler, & Turbo Vision

INSTALL.EXE - 6.0 安装程序

README.COM

WNZIP.EXE - 释放压缩文件.ZIP 程序

TURBO.ZIP - 一经释放后产生:

TURBO.EXE

TURBO.TPL

TURBO.TP

TPC.EXE

TVISION.ZIP - 一经释放后产生:

APP.PAS BUFFERS.PAS COLOSEL.PAS

EDITORS.PAS

MSGBOX.PAS STDDL.G.PAS APP.TPU

BUFFERS.TPU

COLOSEL.TPU DIALOGS.TPU DRIVER-

S.TPU EDITORS.TPU

HISTLIST.TPU MEMORY.TPU MENUS.T-

PU MSGBOX.TPU

OBJECTS.TPU STDDL.G.TPU

TEXTVIEW.TPU NEWS.TPU

TVDEMOS.ZIP - 一经释放后产生:

TVDEMO.PAS DEMOHELP.HLP ASCI-

ITAB.PAS CALC.PAS

CALENDAR.PAS DEMOHELP.PAS FVIEW-

ER.PAS GADGETS.PAS

HELPHFILE.PAS MOUSEDLG.PAS PUZZLE.-

PAS DEMOCMDS.PAS

TVHC.PAS DEMOHELP.TXT TVRDEMO.-

PAS MKRDEMO.BAT

GENRDEMO.PAS TVEDIT.PAS TVFORMS.-

PAS FORMCMDS.PAS

DATA COLL.PAS LISTDLG.PAS FIELDS.-

PAS FORMS.PAS

PARTS.TVF PHONENUM.TVF GEN-

FORMS.BAT GENFORM.PAS

GENPARTS.PAS GENPHONE.PAS FILE-

VIEW.PAS TVTXTDMO.PAS

TVBGI.PAS GRAPHAPP.PAS

INTRFACE.ZIP

TURBO3.ZIP

TOUR.ZIP

REAME

盘 2. Online Help, BGI & Utilities

HELP.ZIP - 一经释放后产生:

TURBO.HLP

BGI.ZIP - 一经释放后产生

GRAPH.TPU ATT.BGI CGA.BGI EGAV-

GA.BGI

HERC.BGI PC3270.BGI IBM8514.BGI

GOTH.CHR

LITT.CHR SANS.CHR TRIP.CHR BGIDE-

MO.PAS

ARTY.PAS BGILINK.PAS BGIDRIV.PAS

BGIFONT.PAS BGILINK.MAK

UTILS.ZIP - 一经释放后产生文件:

THELP.COM TPUMOVER.EXE TEMC.EXE

MAKE.EXE

GREP.COM TOUCH.COM BINOBJ.EXE EM-

STEST.COM

DOCDEMOS.ZIP

TCALC.ZIP

DEMOS.ZIP

ONLINE.ZIP - 一经释放后产生文件:

HELPME!.DOC FIXES.DOC TVISION.DOC

BUFFERS.DOC

EDITORS.DOC TEMC.DOC THELP.DOC U-

TILS.DOC

2. 安装方法:

6.0 版安装可以自动安装和手工安装;

\* 自动安装:

把盘 1 插入 A 驱, 盘符转至 A>, 键入 INSTALL, 根据提示进行, 按照自己需要修改就可以完成了。

\* 手工安装:

用 UNZIP 程序释放.ZIP 文件即可, 如:

UNZIP A:TURBO ←

释放 A 盘 TURBO.ZIP 压缩文件。

尽管 6.0 版可安装在软盘上, 但一般建议安装在硬盘上, 省事方便。

Turbo Pascal V6.0 的功能显而易见, 是以往各版所不能比拟的, 相信它会给编程者带来更多的便利。

# BJS-98 电路原理与 CDW 多窗口调试器

清华大学 袁涛 魏峰

BJS-98 的总体结构图已在“BJS-98 单片机多功能实验系统”一文中提供给读者。为使读者能进一步有效地学习和掌握 8098 单片机原理和应用技术,便于进行单片机基础实验和高级实验,下面将 BJS-98 电路原理图提供给读者,见图 1。

原理图中的译码电路并没有使用通常的逻辑电路芯片组成,而是使用了一片 GAL16V8 可编程逻辑阵列芯片。过去多片 TTL 电路才能实现的功能,一片 GAL 即可实现;印刷电路板设计完成后,使用 TTL 电路译码很难更改地址分配,而 GAL 芯片更改地址分

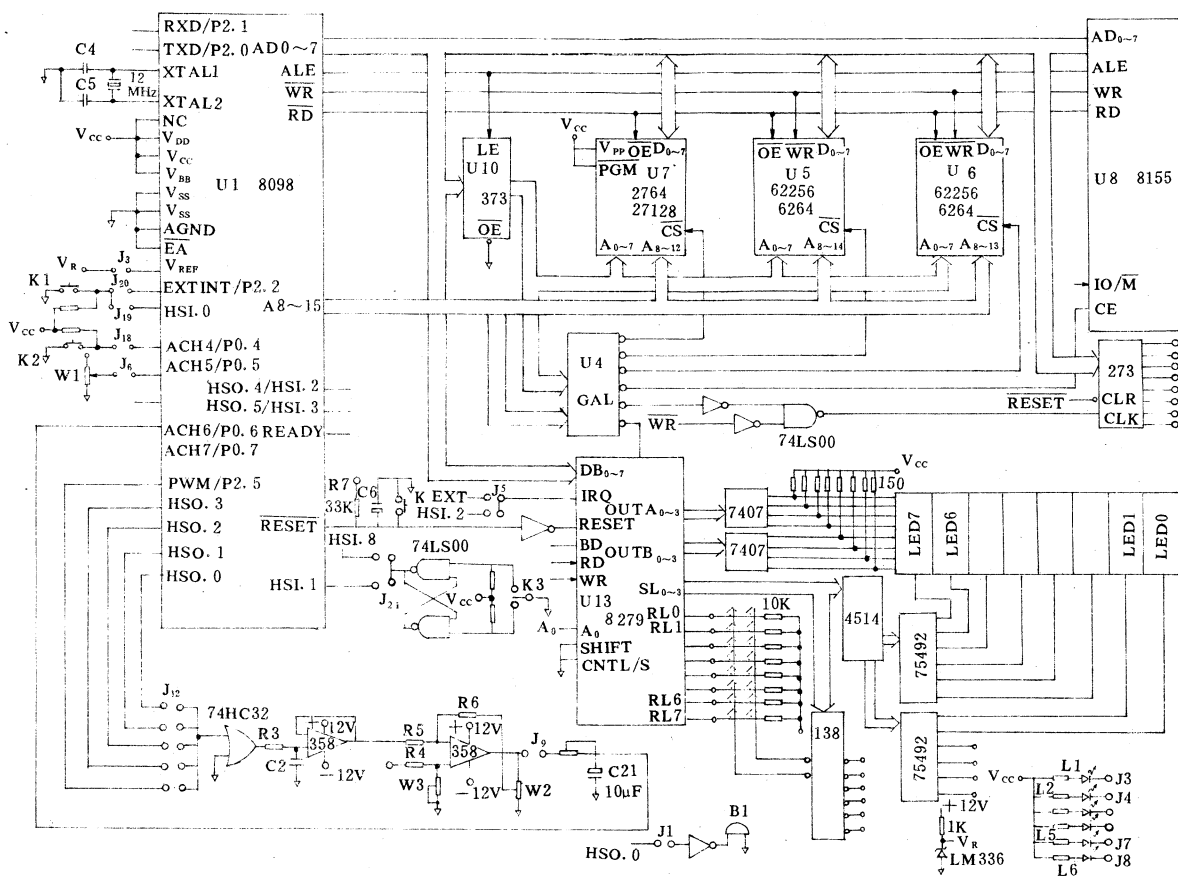


图 1

配则是轻而易举的事;使用 GAL 芯片不仅使印刷电路板体积减小,在不少情况下还会使成本降低。可靠性如何呢?有人因误操作而将电源和地线接反了,电源线都发热了,但 GAL 芯片并未损坏,如果正常使用则更不易损坏了。

BJS-98 并没有采用通常的多块板形式,整体都装在一块电路板上,其优点是总体成本和价格降低,实验方便,由于不存在板与板之间的接插件,可靠性相应提高。一块电路板上可以做单片机的差不多是全部的基础实验和主要高级实验,如数据处理和运算、浮点数处理和运算、数字量 I/O 实验、中断实验、高速输入和高速输出实验、定时器实验、报警器实验、数字显示实验、键盘输入实验、A/D 实验、D/A 实验、串行通信实验、交通灯控制实验、测量仪器实验、控制器实验等内容丰富的实验。

由于 PC 计算机已经很普及,在实际的单片机开发中,PC 计算机已成为开发人员的基本工具,为提高学生的适应能力,为了使学生将精力和时间集中在掌握知识和提高分析问题和解决问题的能力上,在设计 BJS-98 多功能教学实验系统时,是以有 PC 机支持为最基本运行环境的。

BJS-98 使用 CDW 多窗口调试技术,在 PC 机上操作,可以直接面向 PL/M-96 结构化高级语言调试,也可直接面向 ASM96 宏汇编语言调试。由于使用了多窗口方式,各种信息和操作都提示在 CRT 上,便于学生掌握。

#### CDW 多窗口调试器功能包括:

##### 1. 状态显示功能:

可以显示 8098 单片机 256 字节片内寄存器的内容,包括可以读出特殊功能寄存器的值,可以显示全部外部存储空间中的内容,可以字节型、字型、双字型、短整型、整形、长整型、实型(浮点型)、ASCII 字符方式直接显示存储器中的内容。还可设置经常观察项。

##### 2. 状态修改功能:

可对片内寄存器、特殊功能寄存器、外部 RAM 区、外部仿真程序存储器区按字节修改、按字修改、按双字修改、按短整型修改、按整型修改、按长整型修改、按浮点数(实型)修改。

##### 3. 运行控制功能:

可以控制单片机进行单步运行、不追踪过程内部的单步运行、多步跟踪运行、断点运行、连续全速运行。除连续全速运行外,其他各种运行方式都能在运行结束后,向用户提供运行后单片机系统的状态信息。

##### 4. 数据传送功能:

可以将 RAM 中、EEPROM 中、EPROM 中内容存到 PC 机的磁盘上,也可将磁盘文件写入 RAM、EEPROM、EPROM 中。

5. PL/M-96 高级语言调试窗口与反汇编窗口切换功能。

6. 反汇编功能。可以将其显示、存盘,可以在反汇编窗口进行调试运行。

7. 字符串搜索功能。

8. EPROM 或 EEPROM 编程功能。包括读出和写入功能(必须与 EPROM 编程板配合)。

BJS-98 的 CDW 多窗口调试器操作分为窗口命令、菜单选项、会话命令三大部分。这些操作都在同一画面上的不同区域操作,菜单项在使用时将有关项弹出,观察区、源程序区、会话区的大小可根据需要由使用随时调整。CDW 的窗口命令见表 1。

CDW 的菜单选择项主要内容见表 2。

CDW 的会话命令见表 3。

表 3 中的 type 项是指类型,类型共有 8 种,见表 4。会话命令中,如果将一个变量或内存以某种类型的形式进行操作,则应在表 3 中的 type 位置写上相应类型字母,如果不写类型,则表示按源文件中变量类型进

表 2 CDW 的菜单选择项

菜单	选项	功能
Alt-File	Open...	装入新的可执行目标文件
	Load...	装入数据文件
	Save...	内存存盘
	DOS-shell	MS-DOS shell 调用
	Exit	退出 CDW
Alt-View	Source	源文件显示方式
	Disassemble	反汇编显示方式
	Registers. F2	开/关寄存器窗口
Alt-Search	Find...Ctrl+F	查找正文串
	Next	下一个匹配项
	Previous	上一个匹配项
	Label	查找标号
Alt-Run	Start	开始启动
	Reload	重新装入
	Restart	重新装入并复位系统
	Execute	自动步进执行
	Clear-Breakpoints	清除所有断点
Alt-Watch	Add.watch...Ctrl+W	增加观察项
	Delete.watch...Ctrl+W	删除观察项
	Delete.All.Watch	删除所有观察项
	Add.register	增加寄存器
	Delete.register	删除寄存器
Alt-Language	Auto	自动选择语言处理器
	PLM	PLM 表达式处理器
	Assembler	汇编语言表达式处理器
Alt-EPROM	TYPE	选择 EPROM 类型
	TEST	测试 EPROM 是否全 FFH
	WRITE	EPROM 编程
	READ	EPROM 存盘
	COMPARE	比较文件与 EPROM
	EXIT	退出 EPROM 操作

表 1 CDW 的窗口命令

键	功 能
F1	帮助
F2	开/关寄存器窗口
F3	开/关观察窗口
F4	当前行定位
F5	运行(GO)用户程序
F6	显示/会话光标转换
F7	执行到当前光标行
F8	追踪过程单步
F10	跳过过程单步
F9	在当前光标行设置/清除断点
Ctrl—G	扩大当前光标所在窗口
Ctrl—T	缩小当前光标所在窗口
Ctrl—R	重新启动程序
PGUP	显示窗口内容上卷一页
PGDN	显示窗口内容下卷一页
UP ARROW(↑)	光标上移一行
DOWN ARROW(↓)	光标下移一行

行操作。例如下面两条命令操作：

ER .X  
ED .X

第一条命令是将变量作为实型数操作，第二条命令是将变量作为双字型操作。

表 4 type 项的 8 种类型

type 项	对应类型	说 明
A	ASCII	ASCII 字符(8 位)
B	BYTE	字节(8 位)十六进制数值
W	WORD	字(16 位)十六进制数值
D	DWORD	双字(32 位)十六进制数值
S	SHORTINT	短整数(8 位)十进制数值
I	INTEGER	整数(16 位)十进制数值
L	LONGINT	长整数(32 位)十进制数值
R	REAL	实型数(32 位)浮点数值

表 3 CDW 的会话命令

命令类型	命令格式	功 能
断点清除	BC <b>【list】</b>	清除表 list 中的所有中断点。list 是断点序号，彼此逗号分开
断点屏蔽	BD <b>【list】</b>	屏蔽表，list 中的所有中断点。list 是断点序号，彼此逗号分开
断点启动	BE <b>【list】</b>	启动表 list 中的所有中断点。list 是断点序号，彼此逗号分开
断点列表	BL	列出所有的中断点及其状态
断点设置	BP <b>【address count】</b>	在 address 处设置断点，count 是通过次数
注解	*	显示注解正文
延迟	:	延迟从被重定向文件中来的命令的执行(可重复使用)
显示表达式	? <b>expression 【,format】</b>	按 format 格式显示表达式值
内存显示	D <b>【type】 【range】</b>	按 type 格式显示内存范围 range 中的内容，range 用 16 进制表示，首址和末址用空格分开。
	D <b>【type】 【首地址】 L【长度】</b>	显示从首地址开始指定长度的值
键入	E <b>【type】 address 【list】</b>	按 type 格式把值输入内存中
检查符号	X? <b>【mod!】 【func#】 【sym】 【*】</b>	显示给定的符号
执行	E	慢速执行
走	G <b>【address】</b>	执行到 address 处或尾
帮助	H	显示会话命令及语法
暂停	"	暂停从被重新定向的文件中来的命令的执行直到按某一键
程序步	p <b>【count】</b>	执行源代码或指令，越过函数、子程序及中断调用；重复执行 count 次
退出	Q	返回到 MS-DOS



# MCS—51 单片单板机监控程序分析(中)

北京电信学校 江琪 刘葳

## 4. 用 JMP @A+DPTR 指令完成散转操作

由图 2 可知,本单板机一共安排了二十四只按键:十六只数字键,八只命令键。为了更好地利用硬件资源,一般都设置命令键为双重功能,即所谓的上下档功能。这样如果除去一只换档键,或称监控键(MONITOR),剩下七只命令键,可以安排十四种功能操作。对于如何完成上下档功能,后面会有介绍。我们将每只键的上档功能定义如表 1 所示(可根据情况自己重新安排),这样在按表 1 的顺序安排好散转表以后,就可以利用 JMP @A+DPTR 指令完成散转操作了。由 KEY-PRO 子程序我们已在累加器 A 中得到命令键的键值,这样在执行了下面这段程序后,程序就会转到相应的命令键处理程序去执行。

```
FIND-F: ANL A, #07H      ;屏蔽掉高位的 1
        MOV R0, A        ;键值×3→A
        RL A
        ADD A, R0
        MOV DPTR, #FUNC ;散转表首地址→DPTP
        JMP @A+DPTR     ;完成散转操作
FUNC: LJMP MON-KEY     ;散转表
      LJMP EXAM-KEY
      LJMP NEXT-KEY
      LJMP LAST-KEY
      LJMP MOV-KEY
      LJMP EXE-KEY
      LJMP AD-KEY
      LJMP OPR-KEY
```

假设用户按下了 NEXT 键,则 KEY-PRO 子程序返回值 A=12H,经过本程序段的逻辑与操作使 A=02H,乘了 3 以后, A=06H,所以 JMP @+DPTR 指令使程序转到 LJMP NEXT-KEY 处,接着转到 NEXT 键处理程序处去执行了。

## 5. 数字键的处理

完成地址码的输入或数据的改写都需要输入数字键,这样当操作单板机时,如果按下了某数字键,会出现三种情况:

- 输入地址码
- 改写片内 RAM 内容
- 改写片外 RAM 内容

对于这三种情况监控程序要分别进行处理,那么如何区别按下的数字是地址码还是单元内容呢?如何区别是修改片内 RAM 还是片外 RAM 内容呢?我们分别讨论。

①首先,我们简单地描述一下操作过程(不同的机型,操作过程可能不太一样)。开机后,按一下 MON

键, DG1 显示 P, 其余 DG2—DG6 均灭, 如这时按某个数字键, 则在 DG4 处显示出来, 以后不管按多少次数字键, 则依次由 DG4 向 DG1 方向进位, 而 DG5、DG6 始终不亮, 这个过程是地址码输入过程。当地址确定以后, 按一下 EXAM 命令键, 则原地址保持不变, 而在 DG5、DG6 上显示出单元内容。如这时再按数字键, 则 DG1—DG4 内容不变, 最新按的数字显示在 DG6 上, 同时原 DG6 内容(如果有的话)在 DG5 上显示, 即 DG6 向 DG5 进位, 这个过程也正是修改数据的过程。程序是怎样辨别按下的数字是在 DG4 上显示出来还是在 DG6 上显示出来呢? 我们说, 当程序初始化以后, 仅 DG1 显示 P, 其余 DG2—DG6 均灭, 由段值表可知, 此时 79H—7DH 单元内容均为 16H, 特别要指出的是 7AH(DG5)单元内容是 16H, 所以如这时有数字输入, 程序判断 7AH 单元内容为 16H(灭), 则在 DG4 上显示, 地址依次进位。当地址确定以后, 按一下 EXAM 命令键, DG5、DG6 显示出对应的单元内容, 则 7AH(DG5)单元内容必为数字, 即小于 10H, 所以如这时有数字输入, 程序判断 7AH 单元内容小于 10H, 则在 DG6 上显示, 进行数据修改。

②程序通过判断 DG1 的内容是什么来区别修改片内 RAM 或片外 RAM。如果是修改片外 RAM 的内容, 地址码是四位, 则 7EH 单元(DG1)的内容小于 10H。如果是修改片内 RAM 内容, 地址码是两位, 则 7EH 单元的内容为 16H(灭), 大于 10H。

数字键的处理程序清单如下:

```
PUSH A      ;暂时将输入的数字保护到堆栈中
MOV A, 7AH   ;准备判断 DG5 是否有显示
JNB ACC. 4, KK-4 ;低两位有显示, 转去修改低两位
MOV 7EH, 7DH ;低两位无显示, 则继续地址码输入
MOV 7DH, 7CH
MOV 7CH, 7BH
POP A        ;恢复输入的数字到累加器 A 中
MOV 7BH, A   ;写入到 DG4 中
AJMP K-P0    ;完成操作, 转到主程序 K-P0 处
KK-4: MOV 7AH, 79H ;修改低两位显示, DG6→DG5
      POP A        ;恢复输入的数字写入 DG6 中
      MOV 79H, A
      MOV A, 7EH   ;准备检查 DG1 有否显示
      JB ACC. 4, KK-5 ;DG1 无显示, 表明是片内 RAM, 转走
      ACALL DSD2   ;DG1 有显示, 修改片外 RAM, DG1→DG4→DPTP
KK-6: MOV R0, #79H ;将 DG5、DG→A
      ACALL DZ0
```

```

MOVX @DPTP, A ;完成 RAM 内容的真正改写
AJMP K_P0
KK. 5: MOV R0, # 7BH ;完成将 DG3、DG4 所显示的内
      ;容→DPL
ACALL DZ0
MOV DPL, A
MOV DPH, # 17H ;#17H→DPH
AJMP KK. 6

```

在这段程序调用了子程序 DSD2 和 DZ0, 此处略。另外, 完成片内 RAM 内容的真正修改过程实际上是针对从地址 1700H 开始的片外片内 RAM 的修改过程。为什么要这样做呢? 这在单步命令键处理程序中会做进一步的说明。

#### 6. 监控程序的总体结构

监控程序的总体结构框图如图 4 所示。需要说明

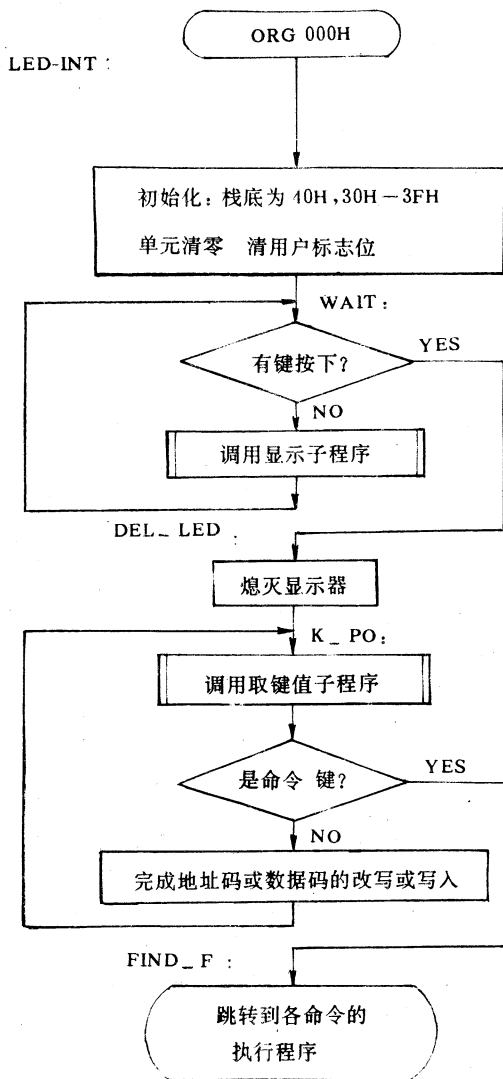


图 4 监控程序总体结构框图

的是 RAM 30H—3FH 单元为监控程序所用, 初始化时需清“0”。设置栈底为 40H。用户标志位 PSW. 5 用来表明命令键的上下档, 当 PSW. 5=0 时, 表明单板机处于上档状态; 当 PSW. 5=1 时, 为下档状态。

程序清单如下:

```

ORG 0000H
NOP
AJMP LED-INT
..... ;空过中断服务程序入口地址段
ORG 0026H
LED-INT: : : : : :
MOV SP, 40H ;栈底是 40H
MOV R0, # 30H ;将 30H—3FH 单元清
           "0"
MOV R1, # 10H
CLR A
D0: MOV @R0, A
    INC R0
    DJNZ R1, D0
    CLR PSW. 5 ;清用户标志位, 使单板机处
              ;于上档状态
WAIT: ACALL KEY_P ;调键盘扫描子程序, 检
        ;查是否有键按下
      JNZ DEL_LED ;有键按下, 转去译码
      ACALL LED_D ;无键按下, 调显示子程序
      AJMP WAIT
DEL_LED: MOV R0, # 79H ;熄灭显示器
DEL_LED0: MOV @R0, # 16H
          INC R0
          CJNZ R0, # 7FH, DEL_LED0
K_P0: ACALL KEY_PRO ;调用取键值子程序
      JB ACC. 4, FIND_F ;如 A> #10H, 转去
        ;完成散转操作
        ;A<10H, 接数字键
        ;处理程序, 而完成数字
        ;处理

```

#### 二、命令键的上下档转换及处理

前面提到过八只命令键, 除去一只监控键 (MON), 其余的七只命令键共完成十四种操作, 它们是通过用户标志位区分的上下档来完成的。当我们按一下 MON 键时, 单板机处于上档状态 PSW. 5=0, DG1 上显示“P”; 我们再点击一下 MON 键, 单板机则改变到处于下档状态 PSW. 5=1, DG1 上显示“P”。可见 MON 键相当于一个乒乓开关, 用来改变单板机的档位。这样 MON 键处理程序可编制如下:

```

MON.KEY: CPL PSW. 5 ;换档
MON0: JNB PSW. 5, MON2 ;此时是上档还是下档
      MOV 7EH, # 1BH ;PSW. 5=1 下档状态, 显
        ;示 P。
MON1: MOV A, # 16H ;DG2-DG6 均灭
      MOV 7DH, A
      MOV 7CH, A
      MOV 7BH, A
      MOV 7AH, A

```

```

MOV 79H, A
AJMP K. P0 ; 转回到主程序段 K. P0 处
MON2: MOV 7EH, #10H ; PSW. 5=0, 上档状态, 显示
      P
AJMP MON1

```

有了 PSW. 5 位, 我们就可以区别上下档了。比如我们安排单步执行命令 (STEP) 与全速执行命令 (EXE) 为同一键。与单板机处于上档状态时, 按一下此键, 监控程序转到 EXE 处理程序处去执行, 当单板机处于下档状态时, 按一下此键, 监控则转到 STEP 处理

程序处去执行。那么监控程序是如何完成上面的转移的呢? 由散转指令处我们知道, 当你按下了 EXE/STEP 键以后, 程序自动转到 EXE-KEY 处去执行, 然而在 EXE-KEY 处, 我们首先安排了这样一条指令: EXE-KEY: JB PSW. 5, ; 当 PSW. 5=1 时, 转去处理 STEP-

```

STEP-KEY命令
; 当 PSW. 5=0 时, 进行 EXE 命令
处理

```

可见, 仅一条指令即完成了上下档处理。

(待续)

# PC 电力拖动控制

天津师大 李文兵

随着微电子技术的发展, 由微电脑构成的 PC (可编程控制器), 其功能齐全、抗干扰力强, 使用方便, 在工业自动化、电力拖动控制、机电一体化等方面得到广泛应用。本文以日本三菱公司产品为例, 介绍 PC 在电力拖动控制方面的应用。

## 一、三相感应电机的 Y/△ 起动控制

### 1. 由电磁开关和定时器组成的 Y/△ 起动电路

三相感应电动机在起动时, 需要很大的起动电流, 对电网有很大的影响。为此, 要求 5.5kW 以上的电机要有 Y/△ 起动器, 使得起动时, 其线圈接成 Y 形; 起动之后, 线圈改为 △ 接法。一般来说, 该起动器由 2 个电磁开关 (接触器), 以及温度继电器和定时器组成, 如图 1 所示。

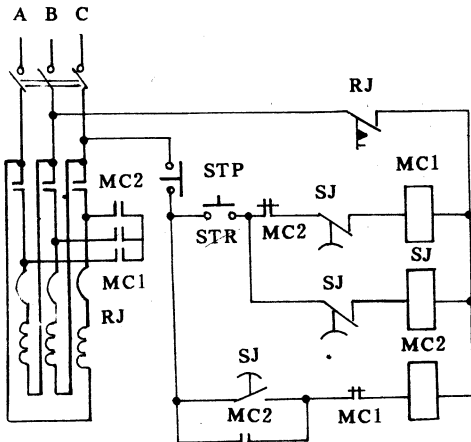
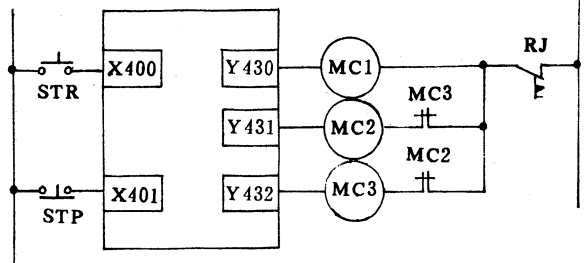


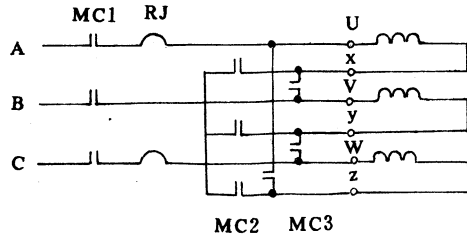
图 1 Y/△ 起动器

### 2. PC Y/△ 起动器

上述起动器可由 PC 取代。PC 的接线如图 2 所示。起动按钮 STR 和停止按钮 STP 作为输入接到 PC 输入端 X400 和 X401 上, PC 的输出端 Y430、Y431 和 Y432 分别接到接触器线圈 MC1、MC2 和 MC3 上。电



(a) PC 外接线图



(b) 主回路

图 2 PC 控制电路

机的三相绕组在触点 MC1 闭合时接通电源; 在触点 MC2 闭合时接成 Y 形; 在触头 MC3 闭合时接成 △ 形。

该电路采用了两种继电器保护。其一是过流保护: 当主回路有过大电流流过时, 接在主回路中的温度继电器工作, 使得与 MC1、MC2 和 MC3 都串接在一起的常闭触点 THR 断开, MC1、MC2 和 MC3 都失电。其二是互锁保护: 从图 2(a) 可以看出, MC2 和 MC3 是互锁的, 即只有对方不带电时, 才能工作。这就避免了 MC2 和 MC3 同时带电所造成的短路事故。

为使 MC2 和 MC3 不同时带电, 当然可以使用 PC 程序, 使 Y2 和 Y3 不同时工作, 软件实现互锁。但是, 万一 PC 发生故障, 仅软件对策并不安全。由于故障, 有可能 Y2 和 Y3 同时工作, 造成 MC2 和 MC3 同时带电。可见, 互锁不仅要软件处理, 而且还要在 PC 外部用硬件处理。这种解决互锁问题的方法在 PC 电力拖动控制中被普遍采用。

### 3. PC 软件设计

要设计 PC 软件, 首先要弄清电机启动/运行的时序。根据电机起动要求, 其时序图如图 3 所示。图 3 说明, 一按起动按钮, MC1 带电, 主回路接通电源, 同时

MC2 也带电,电机三相线圈接成 Y 形,电机进入起动状态。按起动按钮  $t_0$  秒后,起动结束,线圈改为  $\Delta$  接法,进入运行状态。由 Y 接法切换  $\Delta$  接法时,MC2 的触点会产生很大的火花。这火花有可能造成电源短路。为消除火花,在起动完了,MC2 失电后,需再延时  $t_1$  秒后,才让 MC3 再带电。

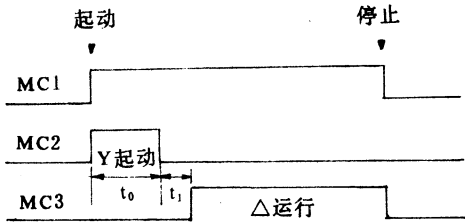


图 3 起动/运行时序图

根据图 3 便可以设计阶梯图,其步骤如下:

首先设计 Y430(MC1)输出,这里采用了停止优先与自保,阶梯图如图 4 所示。

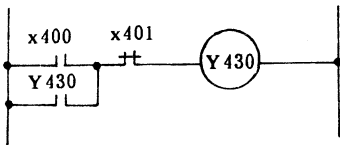


图 4 Y1 阶梯图

其次设计 Y431(MC2)输出,这里使用了 PC 中定时器 T450,用来产生  $t_0$ ,阶梯图如图 5 所示。

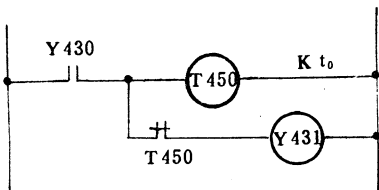


图 5 Y2 阶梯图

最后设计 Y432(MC3)输出,这里使用了 PC 中定时器 T451,用来产生  $t_1$ ,阶梯图如图 6 所示。

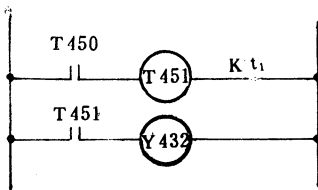
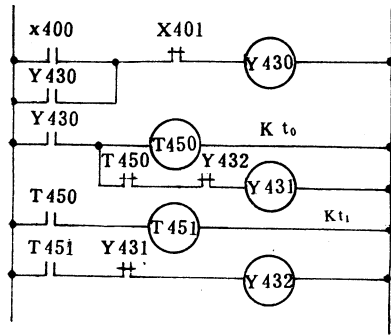


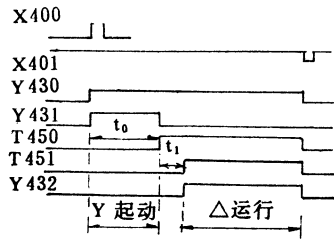
图 6 Y3 阶梯图

把图 4、图 5 和图 6 装配在一起,再加上 Y431 和 Y432 的互锁,便得到所需要的阶梯图,如图 7 所示。图 7(b)是根据阶梯图得到的时序图。

从图 7(b)时序图可以看出, T450 和 T451 一旦时间到,就一直带电,这是没有必要的。对该阶梯图稍加修改,就能使 T450 和 T451 工作后即失电,如图 8 所示。



(a) 阶梯图



(b) 时序图

图 7

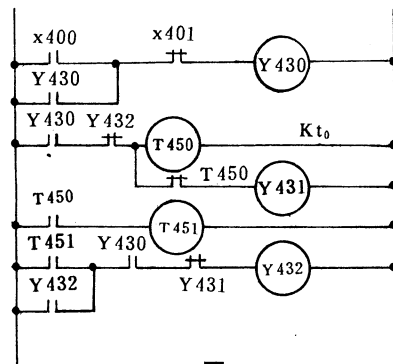


图 8

## 二、起动和停止的延时控制

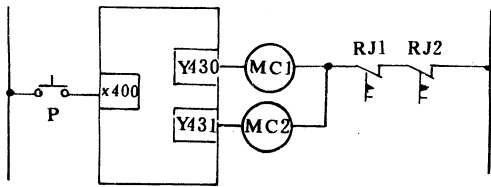
### 1. 起动延时(ondelay)和停止延时(offdelay)

在电力拖动控制中,有些装置按电源 ON 按钮后,需延时一段时间才工作,这叫起动延时;相反,即使按了电源 OFF 按钮,若不经过一段时间,电源实际上并没有关掉,这叫停止延时。家电中,组成暖风机的风机就是这样的装置。

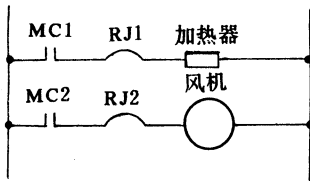
接通电源开关,加热器就首先接通电源;而风机要待加热器预热一段时间后才开始运转。切断电源开关,加热器电源也切断,而风机还要继续运转,直到加热器余热消失时为止。风机的开关控制,用温度传感器是可以实现的。这样的电路很多,这里不做介绍。下面,介绍 PC 控制电路。

控制加热器及风机的 PC 电路如图 9 所示。其中,

图 9(a)是 PC 控制电路,图 9(b)是暖风机的主回路。按钮 P 不是按下会自动弹开的,而是按一次,它就自锁在接通状态;再按一次,才断开。



(a) PC 外接线路



(b) 主回路

图 9 PC 控制电路

根据风机的起动延时和停止延时的要求,加热器和风机的的工作时序,应如图 10 所示。

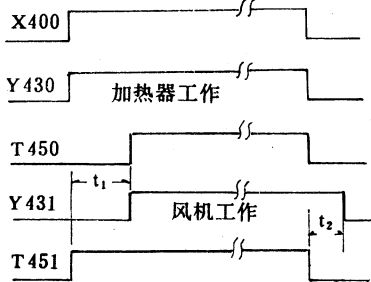


图 10 加热器和风机工作时序

### 3. PC 软件设计

根据时序图,选取  $t_1$  和  $t_2$  的合理值,就可以设计 PC 阶梯图了。

Y 430(MC1)输出的阶梯图如图 11 所示。这里,同时输出 T450,其定时时间为  $t_1$ (100 秒),用来驱动 Y 431。

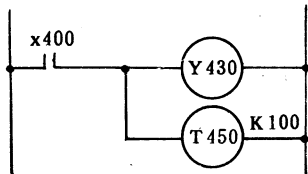


图 11

Y 431(MC2)输出的阶梯图如图 12 所示。为了使风机能在加热器停止工作 50 秒后停机,这里使用了定时器 T 451,定时时间就是 50 秒( $t_2$ )。

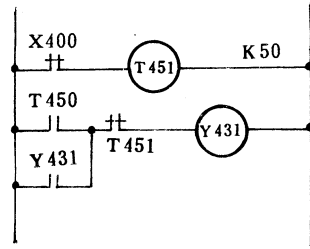
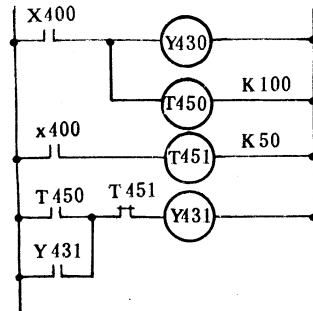


图 12

把图 11 和图 12 装配在一起,即可得到所需阶梯图,如图 13(a)所示。其对应的程序如图 13(b)所示。



(a) 阶梯图

```

LD X400
OUT Y430
OUT T450
K100
LDI X400
OUT T451
K050
LD T450
OR Y431
ANI T451
OUT Y431

```

(b) 程序

图 13

## 三.单相感应电机正反转控制

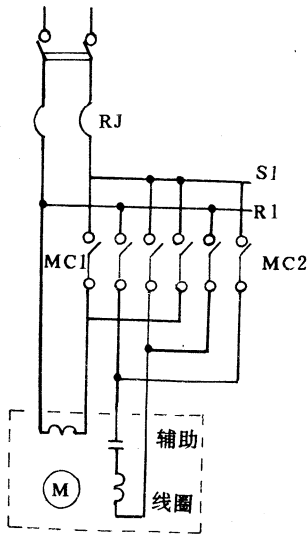
### 1. 继电器控制电路

单相感应电机的正反转控制可以使用继电器实现,这只要改变电容 C 所接线圈的相位即可,其主回路和控制回路如图 14 所示。

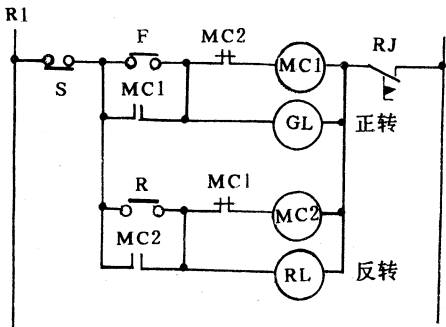
在该电路中,只要一按正转按钮 F,接触器 MC1 即带电,并自保,电机正转。这时,即使按反转按钮 R,电机仍处于正转状态,并不能反转运行。这是因为 MC1 和 MC2 互锁。这样,即使你按了 F 按钮,只要 MC2 还没有失电,MC1 就不能带电;同样,即使按了 R 按钮,只要 MC1 还没有失电,MC2 就不能带电。因此,



使用这个电路进行转向切换,只要没有按停止按钮 S,使 MC1 和 MC2 都失电,就不能实现转向切换。



(a) 主回路



(b) 控制回路

图 14 单相感应电机正反转控制电路

### 2. PC 控制电路

当电机工作在可逆操作时,运行方向的突然变换,会产生过大的冲击,造成机械损坏,或电路里有过大电流流过,对电网和电源产生较大影响。为此,用 PC 取代继电器控制。PC 接线如图 15 所示,阶梯图如图 16。

图 16 完全是由图 14 所示继电器控制电路对应过来的。该阶梯图存有两个问题:一是不按 S 停止按钮,就不能进行转向切换。二是一旦按了 S 按钮, Y 431 (MC1) 和 Y 432 (MC2) 便返回到失电状态,即使电机并没有停止运转,也能马上进行转向切换操作。这样,会在机械上或电气上产生大的冲击,发生事故。

这些问题,若使用 PC 控制,即使不增加电气部件,在程序上稍下功夫,是可以解决的。

### 3. PC 控制软件

能解决图 16 中所存在问题的阶梯图如图 17 所示。其对应程序如图 18 所示。下面说明各部分的作用。

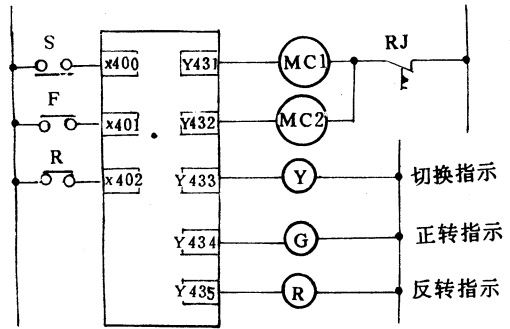


图 15 PC 接线图

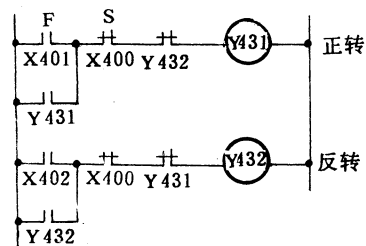


图 16 阶梯图

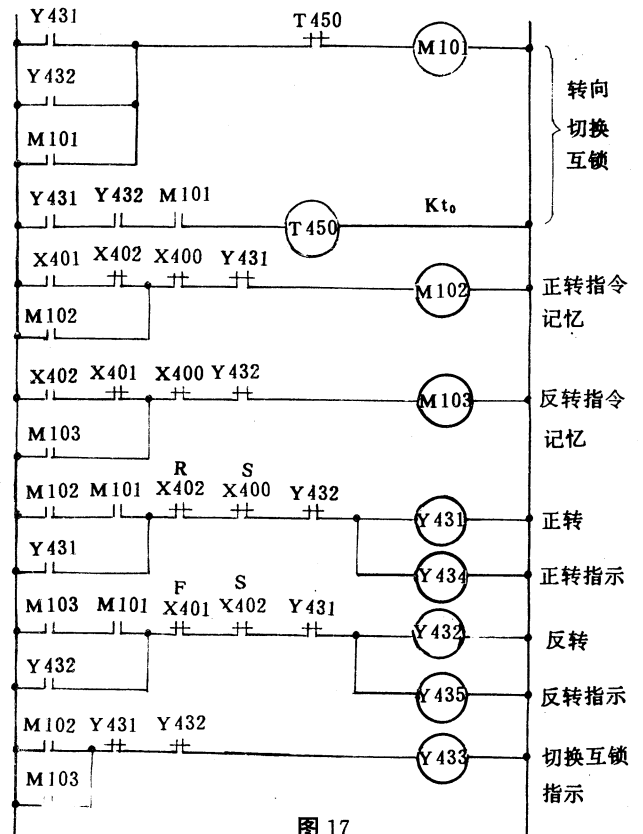


图 17

```

LD Y431
OR Y432
OR M101
ANI T450
OUT M101
LD Y431
AND Y432
AND M101
OUT T450
K t0
LD X401
ANI X402
OR M102
ANI X400
ANI Y431
OUT M102
LD X402
ANI X401
OR M103
ANI X400
ANI Y432
OUT M103

LD M102
AND M101
OR Y431
ANI X401
ANI X400
ANI Y431
OUT Y432
LD M102
OR M103
ANI Y431
ANI Y432
OUT Y433

```

图 18

第一部分是提供时间  $t_0$  的 on 延时部分。 $t_0$  的设定时间比电机惯性运转时间稍长即可。这样就能保证待电机转动停止后,才进行转向切换,以防止转向切换时产生过大的机械的、电气的冲击。

第二部分是正反转指令存贮部分。M102 和 M103 分别存贮正转指令和反转指令。这部分有自保和互锁。

第三部分是正反转执行部分。为切换转向,必须在 M101 失电以前,连续按 F 和 R 按钮,待 M101 失电后,再按 F 或 R。这部分中的 Y434 和 Y435 分别用来指示正转和反转。

第四部分是正处于切换中的显示。

#### 四、工作台往返操作

##### 1. 工作台的工作方式

工作台示意图如图 19 所示。其前进与后退是由蜗轮、蜗杆带动的。蜗杆由电机 M 带动。LS1、LS2 和 LS3 是三个限位开关,它们均由碰锤操作。

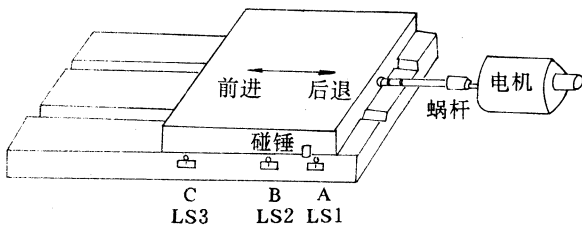


图 19 工作台示意图

工作台一般设有两种工作方式,即自动工作方式和步进工作方式。具体要求如下:

- 自动工作方式起动 只有在工作台处于原点(A 点)时才能进行。
- 自动工作方式停止 根据停止指令发出的位

置,分两种情况。一种情况是,工作台处于作业区,接到停止指令。这时,工作台待结束该周期后,返回到原点。若工作台在从 B 点向 C 点移动时,接到停止命令,则移动到 C 点后后退,检测到 B 点,仍继续后退,停止在 A 点。若工作台在从 C 点向 B 点移动时,接到停止指令,照样通过 B 点,停止在 A 点。另一种情况是,工作台未进作业区,虽然按了自动工作方式起动按钮,这时,工作台返回到原点并停止。自动工作方式的起动和停止处理如图 20 所示。其中,ENDING 为停止处理。

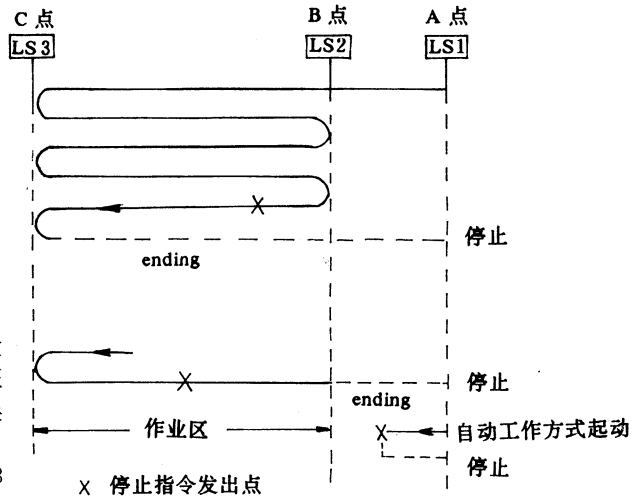


图 20 自动工作方式

• 步进工作方式起动 在前进指令输出期间(即按前进按钮后),工作台前进,到达 C 点停止。可见,C 点的 LS3 具有前端过驱动警界作用。

• 步进工作方式停止 后退指令输出期间(即按后退按钮后),工作台后退,返回到 A 点停止。A 点的 LS1 除表示原点外,也有过驱动警界的作用。

设步进工作方式的目的是有两个:一是用来处理非常停止和停电。非常停止和停电所造成的随机停止,使工作台停止在工程周期途中。这时,即使按动自动起动按钮,也不能再起。要想再起,就必须使用步进操作,把工作台返回到能够起动周期运动的原点位置。二是用来进行机械调整。有了步进操作,进行这项工作就方便多了。

##### 2. PC 的接线

假定我们使用非常停止、自动起动、自动停止、前进、后退五个按钮,再加上三个限位开关,则 PC 的输入信号为八个。从被控的电机驱动电路来看,PC 的输出信号使用两个即可。

分析输入信号的正负逻辑后,不难得到 PC 的接线图,如图 21 所示。

##### 3. PC 控制软件

PC 阶梯图如图 22 所示。其中,Y431 和 Y432 用来控制工作台的移动方向;M101 和 M102 分别用来记

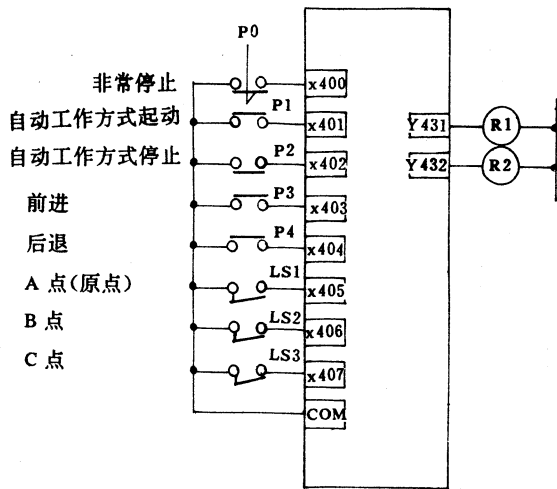


图 21 PC 外接线图

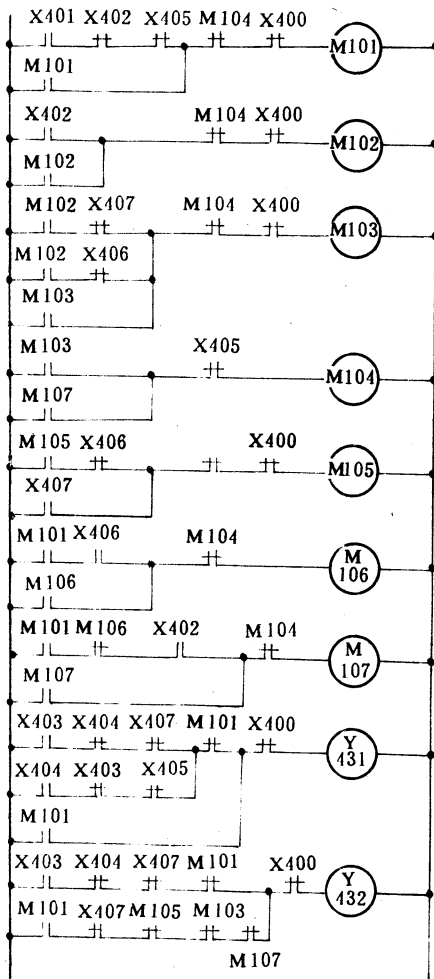


图 22

忆自动工作方式的起动的停止信号;M103 是用于结束周期操作的 ENDing 处理信号;M104 是周期操作结束信号;M105 是后退保持信号;M106 是进入作业区的记忆信号;M107 是工作停止信号。

该阶梯图的详细设计过程,这里不做介绍。该阶梯图先设计步进工作方式;再设计自动工作方式的启动和停止;最后设计其余信号。这样设计比较顺当。设计过程中,应一边设计后面的,一边修改前面的。图 22 就是设计最终结果,其对应的 PC 程序如图 23 所示。

LD X401	OR M106
ANI X402	ANI M104
ANI X405	OUT M106
OR M101	LD M101
ANI M104	ANI M106
ANI X400	AND X402
OUT M101	OR M107
LD X402	ANI M104
OR M102	OUT M107
ANI M104	LD X403
ANI X400	ANI X404
OUT M102	ANI X407
LD M102	LD X404
ANI X407	ANI X403
LD M102	ANI X405
ANI X406	ORB
ORB	ANI M101
OR M103	OR M101
ANI M104	ANI X400
ANI X400	OUT Y431
OUT M103	LD X403
LD M103	ANI Y404
OR M107	ANI X407
ANI X405	ANI M101
OUT M104	LD M101
LD M105	ANI X407
ANI X406	ANI M105
OR X407	ANI M103
ANI M104	ANI M107
ANI X400	ORB
OUT M105	ANI X400
LD M101	OUT Y432
AND X406	

图 23

# 彩色显示器工作原理简述及故障诊断(上)

中国人民大学信息中心 胡野红

CGA, EGA, VGA 以及其它类型的 CRT 彩显、单显除了工作频率不同、分辨率不同和所显示的图形或文本的彩色种类有不同外,其工作原理大同小异。本文以工作实例中挑出的一些具有代表意义的典型故障为例,从彩显工作原理的角度出发通过现象一步步测试分析,最终找到故障点,同时发现另外一些元器件损坏也能引发这种故障出现。愿本文对读者有所帮助。

## 一、彩显的一般工作原理

从图 1 可以看出,由适配器通过“D”型插座送来的场(V)行(H)同步信号、R、G、B、I(对应 CGA)经过整形和鉴别工作方式(对应 EGA、VGA 和多频自适应彩显)送到视放级进行功率放大,然后加到 CRT 的 R、G、B 极,该电子束在灯丝电压的激发下,在阳极电压的吸引下,高速地到达显像管底部轰击 CRT 内壁的荧光粉,使屏幕对应点发光。

若没有行、场偏转线圈的作用,则电子束只固定打在屏幕中心并出现一个亮点。在显像管的脖径上装上两对行场偏转线圈以后,由于其内部流过不同强度,不同频率的锯齿波电流产生了垂直和平行于屏幕的两个力,当电子束经过这两个均匀地、有规律地变化的合力点时(又称为偏转点、偏转线圈因此得名)将受到力的作用(这种力在物理中称为“洛仑兹力”)在屏幕上从左到右从上到下扫描。扫描的频率和显像管的参数共同决定了显示器是工作在 CGA 或 EGA 或 VGA 方式。若在这个过程中没有阳极电压的吸引和加速极的作用,电子束是不能按时、按格式到达荧光屏底部的对应位置。所以 2 万伏左右的阳极电压(视 CRT 的尺寸不同而不同)和加速极电压不可少。

阳极高压和加速极电压均来自高压包(又称高频变压器)。行偏转线圈、行管和高压包共同组成了行扫描电路。行管和行偏转线圈用来产生行扫描所需的锯齿波电流,同时还经过高压包升压得到所需的阳极高压和加速极电压。

行管的控制电压(基极电压)来自行扫描振荡集成电路,常见的型号为 HA11235、HA11423 等。由整形电路送出的 H、V 同步信号加到扫描集成电路中进行处理,在其输出端得到一个和主机同步的脉冲经预激励耦合到行、场功放电路用以产生行偏和场偏所需的不同周期不同强度的锯齿波电流。

由于行扫描电路工作频率高,电压高,功率大,所以容易出现故障。

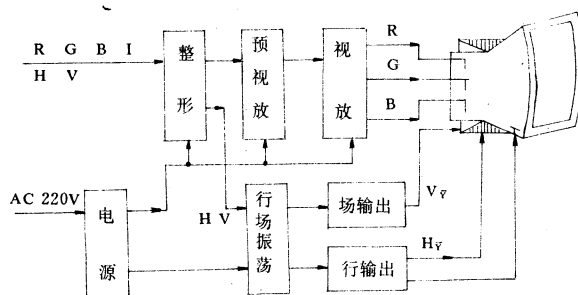


图 1

大部分彩显共用一块集成电路产生行、场所需的脉冲信号,但也有彩显各使用一块集成电路共同完成此项任务。如 Compaq VGA 彩显场振荡电路就单独使用专用芯片 TDA1170。

彩显电源输出数档不同功率、不同数值的直流电压,不同型号的彩显所需的直流电压也不同,如 AST EGA 彩显,电源输出 +150V、+63V、+15V 三档直流。彩显和其它现代化办公设备、家电一样,大都采用开关电源。这种线路效率高,保护功能可靠,所以得到了广泛的应用。

## 二、视放级出现的故障

视放级的典型故障现象为偏色。

例:机型:AST EGA 彩显

故障现象:满屏光栅偏蓝色,且有向左下方倾斜的几条稀疏回扫线,如图 2a 所示。

分析诊断:由彩显工作原理可知,屏幕上出现回扫线大多数为扫描电路在逆程时消隐电路没起到应起的作用而引起。但显像管电路、电源电路、行扫描电路、视放级电路出现故障也可能引起这种故障出现。由本例看出,若消隐电路出了问题,屏幕偏色和回扫线稀疏同时出现的可能性很小。因为消隐信号在行扫描逆程时同时把消隐信号加到了视放级的 Q801(对应 CRT 的 R 极)、Q802(对应 CRT 的 G 极)、Q803(对应 CRT 的 B 极)的发射极,如图 3 所示(本图省略了 Q801、Q802),若无消隐信号或消隐信号不正常,CRT 的三个阴极均不会正常工作,而此时测 Q801、Q802、Q803 的基极电压发现,Q801 和 Q802 基极电压为 4.6V 正常。而 Q803 为 8.4V 属不正常。只能说 B 极有问题引起偏蓝和出现回扫线,消隐电路出故障的可能性可以排除。

仔细检查该显示器,发现亮度和对比度均可调,显像管电路的加速极电压和视放极工作电压均来自电源+150V档,该电压值正常且亮度、对比度可调,说明显像管电路故障引起出现回扫线的可能性也可以排除。

本机电源模块输出的直流电压分别为+150V、+63V、+15V,实测与标称值相等。电源工作正常。

若行电路出了问题,也可能引发此故障。用示波器(万用表)测行管集电极电压波形(该电路行管为Q702),峰值为1100V,波形如图2b所示,均为正常。

在排除上述模块发生故障的可能性之后,判定故障只能出在预视放和视放级。视放级分为预视放和视放。预视放为小信号放大,输入端为整形电路(又称信号通道)输出到视放级。预视放和视放级共同完成功率放大。被放大后的信号加到CRT的阴极。

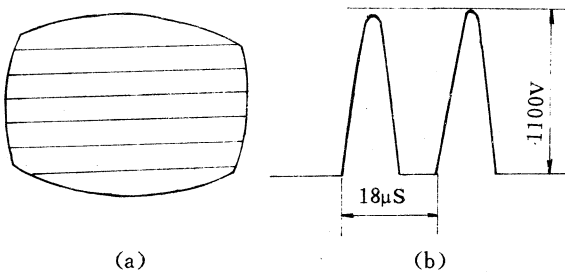


图 2

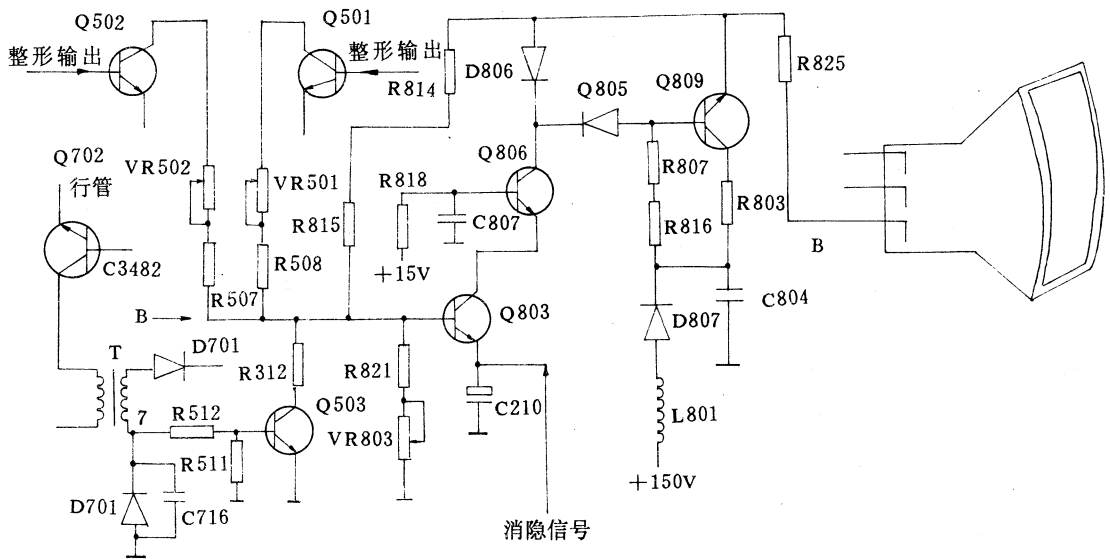


图 3

由本机结构可以看出,预视放和信号通道,行、场扫描电路,电源均在主板,视放级和显像管电路均在尾板上。正确区分故障在主板或尾板的哪一极,是十分重要的,否则将会走许多弯路。

由本电路的对称性可以很快地判断出故障在主

板或尾板。方法是测主板到尾板的R、G、B三极的静态电压。结果发现R、G均为4.6V,而B极为7.4V。B板电压偏高和屏幕偏蓝色相吻合,然后依次分别断开主板到尾板视放级的信号电缆,测主板R、G、B信号电压,结果均为4.6V。同时发现当断开B板信号线时,屏幕上回扫线消除,以上测量结果和屏幕现象说明故障出在尾板视放级的B极电路。再测断开信号输入线的尾板视放级R、G、B入口静态电压,结果为R=4.6V,G=4.6V,B=8.24V,拔下尾板进行外观检查,没有发现明显的烧糊、炸裂元器件,由图3可以看出,尾板视放级B极入口为Q803的基极。该点电位升高无非是上下偏电阻R814,R815,R821短路或开路或是Q809工作异常。事实证明该电路故障的难点就在于以上判断与事实不符。利用本线路的对称性这一有利因素把Q803周围的元器件与另外两极即R或G极对应的元器件对调没有发现异常现象。再加电调VR803发现光栅颜色有变化但回扫线仍然存在。再怀疑Q806,实测静态工作点发现和另外二极的对应管一致。最后只有焊下Q803测试,发现Q803的BE结开路,CB结电阻很小,该管明显损坏,这种损坏现象实为少见。Q803为高速开关管,型号为BSX20,耐压为40V 0.5A,这类管子在市场上难以购到,用参数相似的硅管2SC2235替换后本机工作正常。

从上例可以看出,Q803的CB结短路,BE结开路

后在行扫描逆程时,Q803因CB结短路该截止而没截止反而集电结有较大电流流到基极,基极电位升高致使CRT的B极仍然发射电子束,屏幕呈现兰色。因消隐信号正常CRT其它两极不工作,所以回扫线稀疏。

(未完待续)



## 第五讲 游戏程序的编程特技(中)

山东苍山机械化学电子工业局(277700) 于 春

## 二、卡通特技

## 1. 卡通的自动运动

关于卡通的自动运动问题,我们已在例四“企鹅看外婆”游戏程序中涉及,现予以系统介绍。

卡通的自动运动有多种形式:第一是卡通的种类不确定;第二是卡通的运动方向不确定;第三是卡通的运动速度不确定;第四是卡通的运动量不确定;第五是卡通的色彩不确定;第六是卡通显示于背景的优先度不确定;第七是卡通运动的起点位置不确定。游戏程序中选择哪几种自动运动形式,一般要根据游戏的实际需要而确定。“企鹅看外婆”游戏中我们仅选用了卡通的颜色和起点纵坐标随机产生两种形式。程序 No. 5—12 演示了七种形式的综合表现。

```
5 REM "No. 5—12"
10 CLS.;SP.O.
15 PAL.B 0,33,32,38,38
20 F.I=0 TO 55
25 P.CH.(216);CH.(217);CH.(218);
30 N.
35 LOC.0,15
40 F.I=0 TO 223
45 P.CH.(255);;N.
50 F.I=0 TO 7;GOS.75;N.
55 F.I=0 TO 7
60 IF M(I)=0 T.GOS.75
65 N.
70 G.55
75 DE.M.(I)=SP.(RND(16),RND(8)+1,RND(9)+1,
RND(155)+100,RND(2),RND(4))
80 POS.I,RND(236)+20,RND(240);M.I
85 RE.
```

运行程序,在天蓝色的背景上天上一片白云,地下一块绿地,16种卡通穿梭运动,构成一幅美丽的画卷。读者可以从中深刻体会到七种运动形式的奇妙。

卡通自动运动的目的一般分为两类:一类是无目的运动,如程序 No. 5—12 中的运动;一类是有目的运动,如“成龙救金凤”游戏中妖怪的运动。实用中,有目的运动使用的较多。无目的运动较容易实现,有目的运动的实现要复杂得多。“成龙救金凤”中用了较大的程序量实现妖怪自动向成龙运动的控制。限于篇幅,不再列程序清单,读者可仔细阅读模块 I 的程序。

## 2. 用操纵器控制卡通跳跃

卡通的跳跃有两种情况:一种是原地跳;一种是运动中跳。运动中跳还有向左跳、向右跳。F BASIC 语言中没有令卡通跳跃的语句,而 MOVE 语句只能令

卡通在确定的位置开始运动,也不能产生跳跃。因此,只能使用 SPRITE 语句以不同的时间间隔、在不同的位置显示卡通的不同的姿式,从而产生卡通跳跃的画面。设计卡通跳跃的思路是:(以玛丽为例)卡通图形库有玛丽走、跳、跌、滑等七个姿式。设计运动卡通跳跃时,开始先显示玛丽走的姿式,随后在跳高座标上显示玛丽跳的姿式,在落地座标上再显示玛丽走的姿式,同时加入在 X 方向的位移,使跳跃后向前移动一段距离。若是原地跳则显示玛丽跌下的姿式。

具体实现方法是:定义 0<sup>#</sup> 显示为卡通跌下姿式,1<sup>#</sup> 为面向右跳姿式,2<sup>#</sup> 为面向左跳姿式,7<sup>#</sup> 动作为运动卡通,选用操纵器的 A 键控制卡通跳跃。假定玛丽在向右运动中按 A 键,则使用“SP. 1,X,Y”语句在跳高座标上显示 1<sup>#</sup> 姿式,随后使用“ERA 7”语句令 7<sup>#</sup> 动作消失,使用“PAU. 5”延时一段时间,再使用“POS. 7,X1,Y1;M. 7”语句在新的座标上令 7<sup>#</sup> 动作运动,同时“SP. 1”令 1<sup>#</sup> 显示卡通消失。这样就实现了卡通在向右运动过程中的一次跳跃。同理,在向左运动时则显示 2<sup>#</sup> 显示卡通,原地跳时显示 0<sup>#</sup> 显示卡通。演示程序见 No. 5—13。

```
5 REM "No. 5—13"
10 CLS.;SP.O.;X=100;Y=100
15 DE.SP.0,(0,1,0,1,0)=CH.(25)+CH.(24)+CH.
(27)+CH.(26)
20 DE.SP.1,(0,1,0,1,0)=CH.(13)+CH.(12)+CH
(15)+CH.(14)
25 DE.SP.2,(0,1,0,0,0)=CH.(12)+CH.(13)+CH.
(14)+CH.(15)
30 DE.M.(7)=SP.(0,V,1,4);POS.7,X,Y;M.7;SP.S
35 S=STL.(0);IF (S=1)+(S=2)=0 T.S=0;V=0
40 K=STR.(0);IF K=8 T.60
45 IF S=1 T.V=3
50 IF S=2 T.V=7
55 X=XP.(7);Y=YP.(7);G.30
60 X=X+6*((S=2)-(S=1));IF X<12 T.X=240
65 IF X>240 T.X=12
70 Y=Y+20*((K=8)-(F=1))
80 IF F=0 T.SP.S,X,Y;ERA 7;PAU.5;F=1;K=0;G.
60
85 POS.7,X,Y;F=0;V=0;G.30
```

程序中 35 行的“IF—THEN”语句的作用是封锁操纵器除左、右运动外的六个方向键。60 行产生 X 方向的位移。70 行产生 Y 方向的位移。两行均使用了运算符判别式,其作用如下:在 X 轴方向若卡通向右运动则 S=1,那么(S=2)=0,(S=1)=-1,则 X=X+6,跳起时卡通向前移动六点,落地时再向前移动六点。一

次跳跃在 X 方向向前移动12点。同理,当卡通向左运动时  $S=2$ ,则有  $(S=2)=-1$ ,  $(S=1)=0$ ,  $X=X-6$ , 一次向左跳跃也向前移动12点。在 Y 轴方向若按 A 键则  $K=8$ ,这时  $F=0$ ,那么  $Y=Y-20$ 。在  $(X,Y-20)$  座标上显示跳的姿式,随后  $K=0$ , $F=1$ ,再返回70行时则  $(K=8)=0$ ,  $(F=1)=-1$ ,有  $Y=Y-20+20=Y$ ,卡通又落回到跳起时的座标。

### 3. 用操纵器控制卡通运动的速度

在“采蘑菇”游戏中,按 B 键可令玛丽运动速度增加一倍,在“火箭车”游戏中按 B 键也可以提高车速。这一过程是如何实现的呢?说穿了很简单。方法是在定义受操纵卡通时,用一变量 D 代替它的速度值。正常时  $D=2$ 。当读到  $STRIG(0)=4$ 时,(按 B 键)则令  $D=1$ ,若松开 B 键,仍有  $D=2$ ,从而实现了用操纵器控制卡通的运动速度。

在程序 No. 5—13中,更改10、30行,添43行即可达到目的,三行程序如下:

```
10 CLS;SP.O.;X=100,Y=100;D=2
30 DE.M(7)=SP.(0,V,D,4);POS.7,X,Y;M.7;
   SP.S
43 D=2;IF K=4 T.D=1
```

运行程序,按 B 键就可使玛丽的运动速度提高一倍。

### 4. 双操纵器的使用

“魂斗罗”、“双截龙”、“忍者”等游戏都是双打游戏。游戏中使用两个操纵器分别控制两个主人翁的动作。由于参战人员增加,出现了战斗的配合和任务的分工,从而使游戏更加激烈有趣,富有吸引力。

有了单操纵器的使用经验,使用双操纵器并不难。一般只要编制两个单操纵器控制程序连接在一起,令1#操纵器控制1#动作,用2#操纵器控制2#动作。程序运行中,先测试1#操纵器、执行1#动作后,再测试2#操纵器,执行2#动作。轮流检测,轮流控制,就可达到使用双操纵器的要求。由于 F BASIC 语言程序执行速度较慢,致使双操纵器控制卡通的动作速度也慢一些。这一缺陷是先天的,请不要强求。程序 No. 5—14为双操纵器控制玛丽、丽莎运动示范程序。

```
5 REM "No. 5-14"
10 CLS.;SP.O.
15 F.I=0 TO 1
20 DE.M(I)=SP.(I,0,1,8);POS.I,100,I*80+80;M.
   I,N.
25 F=F+1;IF F>1 T.F=0
30 S=STI.(F);IF S=0 T.25
35 IF S=1 T.V=3
40 IF S=2 T.V=7
45 IF S=4 T.V=5
50 IF S=8 T.V=1
55 X=XP.(F);Y=YP.(F)
60 DE.M.(F)=SP.(F,V,1,8);POS.F,X,Y;M.F
65 G.25
```

### 5. 卡通游泳

实现卡通游泳的方法就是画一水面,令卡通在背景后面缓慢移动,并使其头部露出水面,这样就形成了卡通游泳的动画。实现程序如下:

```
10 CLS;SP.O.
20 LOC.0,10;F.I=0 TO 140;P.CH.(254);;N.
30 F.I=0 TO 140;P.CH.(202);;N.
40 LOC.6,0;P.CH.(216);CH.(217);CH.(218)
50 LOC.7,1;P.CH.(216);CH.(217);CH.(217);CH.
   (218)
60 DE.M.(0)=SP.(0,3,4,100,1,1);POS.0,20,95;M.0
   若使用操纵器控制卡通游泳,可使其潜入水下、
   跃出水面、从水中走出在沙滩上奔跑,再走入水内等。
   见程序 No. 5—15
5 REM "No. 5-15"
10~60行语句同上
70 S=STI.(0);IF S=0 T.70
80 IF S=1 T.V=3
90 IF S=2 T.V=7
100 IF S=8 T.V=1
110 IF S=4 T.V=5
120 X=(XP.(0)-12)/8;IF X>24 T.X=4
130 IF X<4 T.X=24
140 Y=(YP.(0)-16)/8;IF Y<10 T.Y=10
150 IF Y>B T.170
160 DE.M.(0)=SP.(0,V,3,4,1,1);POS.0,X*8+12,Y
   *8+16;M.0;G.180
170 DE.M.(0)=SP.(0,V,1,4,0,1);POS.:.0,X*8+12,
   Y*8+16;M.0
180 G.70
```

程序170行是当卡通运行到水边沙滩附近时,使其从水中显示于背景前面。读者若喜欢卡通跳跃可在70行之后加入跳跃语句。另外若使用 BGTOOL 语句在水面上画上轮船,岸边画上礁石、悬崖、亭台楼阁等将使画面更加形象。

运行程序,可用操纵器控制玛丽在水中游泳,按向下键则潜入水中,继续按向上键,则从水中出来站立在沙滩上。按左右键可控制其在沙滩上奔跑。按向上键则走向水中,走到一定深度则潜入水中。继续按向上键则从水中露出头部,可继续游泳。

### 6. 运动卡通的放大与缩小

在“采蘑菇”游戏中,玛丽吃到红蘑菇后身体立即长大一倍,不少朋友感到很奇怪,这种身体大小的变化是如何实现的呢?现在介绍实现的方法。

我们知道在立体分解画面指令中有一条 DEF SP RITE 语句,使用该语句通过变量 B 可以定义显示在 SP 面上的卡通图案发生大小变化。当  $B=1$ 时,卡通图案有4个字符大小,当  $B=0$ 时,卡通只有一个字符大小。我们只要解决了用操纵器控制显示卡通运动就可实现运动卡通的放大与缩小。程序见 No. 5—16。

```
5 REM "No. 5-16"
10 CLS;SP.O.;CG.1,0;PAL.B 0,41,48,48,48
20 DE.SP.0,(0,0,0,1,0)=CH.(29)+CH.(28)+CH.
   (31)+CH.(30)
```

# 封面说明

```
30 DE. SP. 1, (0, 0, 0, 1, 0) = CH. (33) + CH. (32) + CH.
(35) + CH. (34)
40 DE. SP. 2, (0, 0, 0, 1, 0) = CH. (37) + CH. (36) + CH.
(39) + CH. (38)
50 DE. SP. 3, (0, 1, 0, 1, 0) = CH. (29) + CH. (28) + CH.
(31) + CH. (30)
60 DE. SP. 4, (0, 1, 0, 1, 0) = CH. (33) + CH. (32) + CH.
(35) + CH. (34)
70 DE. SP. 5, (0, 1, 0, 1, 0) = CH. (37) + CH. (36) + CH.
(39) + CH. (38)
80 X=0; Y=100
90 X=X+9; IF X>250 T. 140
100 SP. 0, X, Y; SP. 3; PAU. 20
110 SP. 1, X+3, Y; SP. 0; PAU. 20
120 SP. 2, X+6, Y; SP. 1; PAU. 20
130 G. 90
140 X=0
150 X=X+9; IF X>250 T. 80
160 SP. 3, X, Y; SP. 5; PAU. 20
170 SP. 4, X+3, Y; SP. 3; PAU. 20
180 SP. 5, X+6, Y; SP. 4; PAU. 20
190 G. 150
```

运行程序可见较小的丽莎小姐以屏幕左侧走向右侧,而后又以大一倍的身形从左侧走向右侧,如此反复演示。若在程序中加入自动运动的乌龟和蘑菇(注——蘑菇可从背景图形库中选取四个图案形成)并加入操纵器控制。令丽莎碰到蘑菇则长大,碰到乌龟则变小,更形象地再现了运动卡通的放大与缩小。

为便于阅读理解,程序 No. 5—16 中使用重复语句行较多,我们可以通过循环语句的使用化简程序。简化程序为 No. 5—17。

```
5 REM "No. 5—17"
10 CLS; SP. 0; CG. 1, 0; PAL. B 0, 41, 48, 48, 48
20 F. I=0 TO 2
25 F. J=0 TO 3 ST. 2
30 DE. SP. I+J, (0, -(J=3), 0, 1, 0) = CH. (29+I*4) +
CH. (28+I*4) + CH. (31+I*4) + CH. (30+I*4)
40 N. :N.
50 X=0; Y=100; F=0; A=0
60 X=X+9; IF X>250 AND F=0 T. X=5; SP. A+2; A
=3; F=1
70 IF X>250 AND F=1 T. X=5; SP. A+2; A=0; F=0
80 SP. A, X, Y; SP. A+2; PAU. 20; SP. A+1, X+3, Y; SP.
A; PAU. 20; SP. A+2, X+6, Y; SP. A+1; PAU. 20
90 G. 60 (待续)
```

为了方便广大用户调试目标机,扩展CYSCB-2 51、98单片单板机的功能,武汉创意电子研究所特向用户推出CYF-1仿真接口板,在CYSCB-2 51、98单片单板机上接上该接口板,便能对MCS-51用户板进行仿真调试(高32K),进行慢速、快速断点运行,单步运行。这样不仅能开发自身(见前期封面说明),而且能开发目标机,大大扩展了开发功能。

本所开发的CYLED-1型电子广告牌,采用高亮度发光二极管,分辨率高、视角宽广。能方便地进行汉字、图形编辑,进行多种模式显示,可靠性好。是车站、机场、体育馆场进行信息显示的理想装置,欢迎选用。

CYSCB-2 单片单板机硬件配置 CPU: 8031或8098。RAM 6264×2, 高8K带掉电保护。A/D: 0809。D/A: 0832。PIO: 8255。外部Watch Dog。40芯总线输出供扩展用。CYF-1仿真接口板。

软件配置 高位机: 编辑、汇编、反汇编、DEBUG、通讯。本机: 本机监控和CRT监控, 8031/8098应用子程序库。

特点 1. 独特的软单步和断点设置, 提供良好的调试环境。最优! 2. 本机板提供A/D、D/A参考电平、RS232电平。单5V供电。最简! 3. 硬件、51+98; 功能: 学习+应用+仿真。性能价格比最高! 4. 应用软件丰富,(数值计算、显示、键盘、A/D、D/A、多机通讯、打印机驱动、实际应用系统等)直接供用户使用。最省! 5. 选材严格、工艺考究。可靠!

资料配备①《单片机应用文集》②《MCS-51、8098单片单板机系统设计及应用》③《MCS-51、8098单片机实验指导书》

本产品保修一年,一月内可退货(不问原因)。

下面是本所产品的报价单:

- |                                                                                 |           |
|---------------------------------------------------------------------------------|-----------|
| 1. CYSCB-2 MCS-51、8098单片单板机(附系统盘两张、通讯电缆、CYF-1仿真接口板、8031/98应用子程序库软盘一张,资料配备①、②、③) | 1198.00元  |
| 2. CYSCB-2 MCS-51单片单板机(附系统盘一张、其它同上)                                             | 698.00元   |
| 3. CYD-1 TV/CRT 显示接口板                                                           | 450.00元   |
| 4. CYD-1全系列智能EPROM编程器(单+5V供电)                                                   | 350.00元   |
| 5. CYGAL-1 GAL编和器(单5V供电, 16V8、20V8)                                             | 788.00元   |
| 6. 单色电子广告牌、三色电子广告牌                                                              |           |
| 7. CYPRN-1 256×256×8BITS帧存卡                                                     | 3500.00元  |
| 8. CYP-1电脑打像机(主机 IEMPC)                                                         | 14800.00元 |
| 9. CYP-2一体化电脑打像机                                                                | 10500.00元 |

地址: 武汉武昌珞瑜路100-1号(430070)

武汉创意电子研究所

电话: (027) 712761 传真: (027) 701803

开户行: 武汉市洪山区科技信用社 帐号: 967-517

联系人: 吴微 罗维国 傅彦 宋咏良

## 裕兴中英文游戏机电脑键盘

质量高、功能强、同类产品价最优

带有中文功能的、裕兴中英文游戏机电脑键盘问世半年多来,以其鲜明的特色、低廉的价格、完善的售后服务,受到广大用户的欢迎,成为同类产品中的名牌。它独有掉电保持、录音机测试功能,带有音乐、英

语、数学计算等多种演示程序,同类产品性能价格比最高,是中、小学学生学习计算机知识的最佳选则。配中英文2合1卡,可学习中英文、练打字、作曲演奏。利用F BASIC语言编制学习和游戏程序。手册详尽。邮购每套315元,免邮费并保修一年。索取资料附回信封,北京西城西教场胡同35号,裕兴机械电子研究所,电话2252309,邮编100035。

# 第二讲 开关电源基本组态和企业标准

李秀华 张占松 梁金稳

## 一、开关电源的基本组态

如果以一个开关工作,输入与输出不需要隔离而言,开关电源的基本组态可认为有三种。它们称为降压变换器(buck converter),升压变换器(boost converter)与降-升压变换器(buck-boost converter)。在此基础上加多开关数量,达到输入-输出隔离时,则可以派生出许多电路来,如半桥、全桥、推挽、正向、反向和很具特色的古卡(cuk)变换器等。

下面是各电路主要电量关系式

### (1) 降压变换器

电路及波形图如图4、5所示

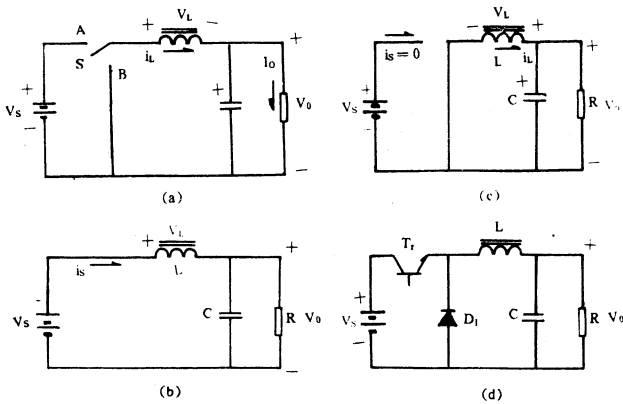


图4 降压变换器电路原理图

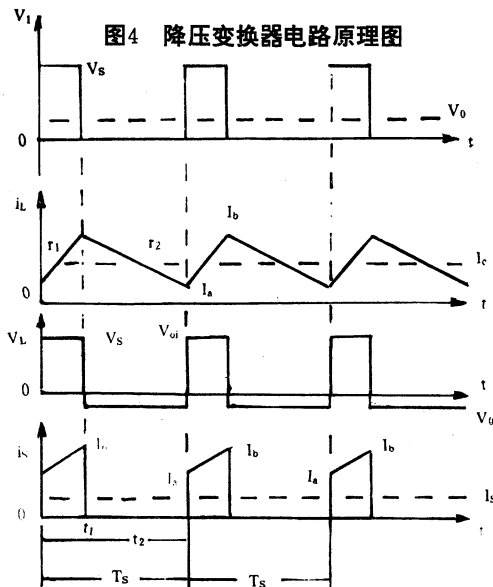


图5 工作在降压变换器波形图

在一个工作周期  $T_s$  中,开关分别在 A、B 位置上。在 A 位时,  $V_s$  是  $V_L$  与  $V_o$  之和,  $i_L$  在  $t_1$  时间里直线上升。在 B 位时,  $V_L$  等于  $V_o$ ,  $i_L$  在  $t_2 - t_1$  时间里线性下降。在  $T_s$  时间里,  $i_L$  呈锯齿状,当锯齿下点与横轴相交时,为电流连续与不连续的分界点。电流连续时:

$t = 0 \sim t_1$  磁通增量  $\Delta\Phi_{inc}$ , 则有:

$$V_L = V_s - V_o = N \frac{\Delta\Phi_{inc}}{\Delta t} = N \frac{\Delta\Phi_{inc}}{t_1}$$

$t = t_1 \sim t_2$  磁通减少  $\Delta\Phi_{dec}$

$$V_L = -V_o = N \frac{\Delta\Phi_{dec}}{\Delta t} = N \frac{\Delta\Phi_{dec}}{t_2 - t_1}$$

稳态时  $\Delta\Phi_{inc} = \Delta\Phi_{dec}$

$$\therefore V_o = \frac{t_1}{t_2} V_s = \frac{t_1}{T_s} V_s = \Theta V_s \quad (1)$$

$\Theta = \frac{t_1}{T_s}$  —— 占空比;  $N$  为  $L$  的函数。

如果电流不连续时,按照输入功率  $P = \frac{V_s(V_s - V_o)}{L} t$  在一周内平均值与输出功率  $V_o I_o$  相等的原则有:

$$\frac{1}{T_s} \int_0^{t_1} \frac{(V_s - V_o)}{L} t dt = V_o I_o \text{ 整理得输出电压}$$

$$V_o = \frac{(V_s t_1)^2}{V_s t_1^2 + 2I_o L T_s} \quad (2)$$

另外,平均电流  $I_o$  可表为:

$$I_o = I_a + \frac{V_o}{2L} (t_2 - t_1), I_a = 0 \text{ 时, 电流开始不连续, 保}$$

证电流连续的电流条件是  $I_o > \frac{V_o}{2L} (t_2 - t_1)$ 。电感量足够大,足以保证线圈纹波电流峰——峰值小于负载电流  $I_o$  一半时,电流为连续。即

$$\frac{V_s - V_o}{L} t_1 < \frac{1}{2} I_{o\max}$$

$$\therefore L > \frac{2(V_s - V_o)}{I_{o\max}} t_1 \quad (3)$$

一般电压纹波表示式

$$\Delta V = \frac{V_o}{8LC} (t_2 - t_1) T_s \quad (4)$$

### (2) 升压变换器

电路及波形图如图6、7所示

开关  $S$  在位置 a 时  $V_L = V_s$ , 电感上的电流  $i_L$  呈线性上升,电感逐渐积累磁场能量。续流二极管因承受反偏压而截止,其反偏电压数值就是  $V_o$ 。在这区间输出滤波电容  $C$  放电提供负载  $R$  的电流  $I_o$ 。当开关  $S$  在位置 b 时,  $T_1$  截止,  $L$  由于自感作用极性颠倒,于是  $L$  上的电压与输入电源电压叠加后,通过  $D_1$  加给负载。同时向电容  $C$  充电。因此,在负载上得到一个比输入电

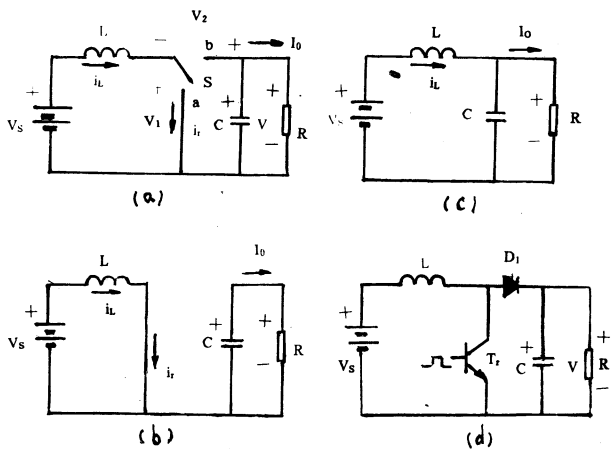


图6

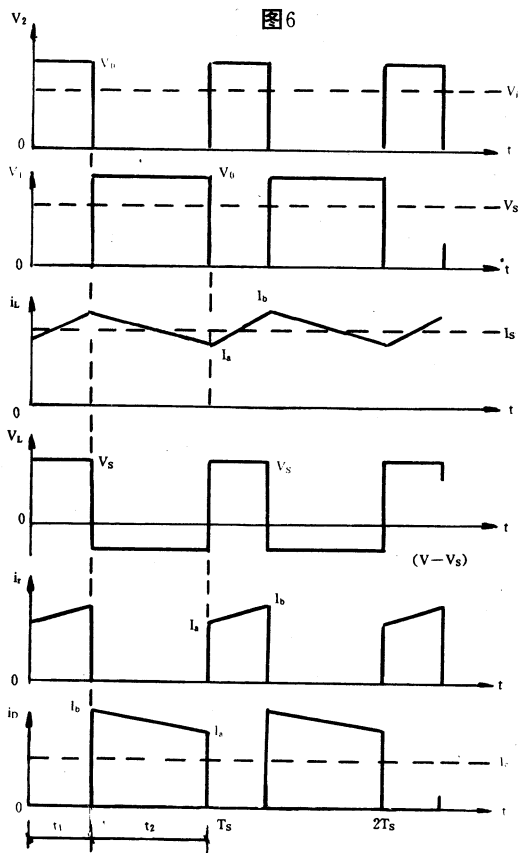


图7 升压变换器连续状态

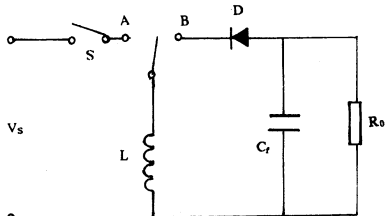


图8

压  $V_s$  高的  $V_o$ 。控制  $T_r$  的基极脉冲, 则可调输出电压  $V_o$ , 在构成闭环后, 自动控制占空比则能稳压。由于电流是连续的, 所以在开关周期  $T_s$  最后时刻, 电感  $L$  上流过的电流  $I_o$  值, 就是下一个  $T_s$  周期中电流  $I_o$  的开始值。但是如果电感太小, 电流线性下降过快。即在能量释放完时, 尚未达到晶体管重新导通的时刻, 因而能量得不到及时的补充, 这样出现了电流不连续状态(未示出波形)根据电流上升和下降的变化量相等的原则:

$$\frac{V_s}{L} t_1 = \frac{V_o - V_s}{L} (t_2 - t_1) \text{ 整理得:}$$

$$V_o = \frac{1}{1 - \Theta} V_s \quad (5)$$

式中  $\Theta = \frac{t_1}{T_s}$ , 称占空比。

如果电流不连续时, 依照降压相似推导方式可得:

$$\frac{1}{T_s} \left( \int_0^{t_1} \frac{V_s^2}{L} t dt + \int_{t_2}^{t_1} \frac{V_s(V_o - V_s)}{L} t dt \right) = V_o I_o \text{ 整理得:}$$

$$V_o = \frac{(V_s T_s)^2}{2 I_o L T_s} + V_s \quad (6)$$

保证电流连续的电流条件是  $I_o > \frac{V_s}{2L} t_1$

电感量  $L$  设计原则与降压相同, 即

$$\frac{V_s}{L} t_1 < 0.5 I_o (\max)$$

$$\therefore L > \frac{2 V_s}{I_o (\max)} t_1 \quad (7)$$

(3) 反极性开关变换器或降-升压变换器。

电路图如图8所示。  $S_1$  在 A 位置时电流  $i_L$  流过电感线圈  $L$ , 电感中积累能量。当  $S$  在 B 位时,  $i_L$  有减小趋势, 电感线圈产生自感电势, 反向二极管  $D$  正偏导通, 负载上电压  $V_o$  与输入电压  $V_s$  是反极性的。

在稳态时,  $t_1$  与  $(t_2 - t_1)$  电流变化量相等。

$$\frac{V_s}{L} t_1 = \frac{V_o}{L} (t_2 - t_1) \text{ 整理得:}$$

$$V_o = \frac{t_1}{t_2} V_s \quad (8)$$

如果电流不连续时,

$$\frac{1}{T_s} \int_0^{t_1} \frac{V_s^2}{L} t dt = V_o I_o \text{ 整理得:}$$

$$V_o = \frac{(V_s t_1)^2}{2 I_o L T_s} \quad (9)$$

电流连续的临界值与降压变换器相同。  $I_o \geq \frac{V_o}{2L} (t_2 - t_1)$  保证电流连续的电感量与升压变换器相同。

$$\frac{V_s}{L} t_1 < 0.5 I_o (\max)$$

$$L > \frac{2 L_s}{I_o (\max)} t_1 \quad (10)$$

二、国内开关电源的状况简述

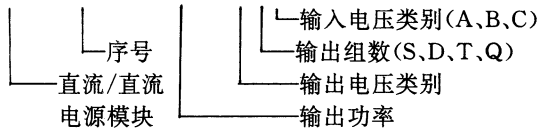
我国在1974年就有10KHz、5V、50A 无工频变压器开关稳压电源研制成功的报导。以后又生产了SL-64等串联开关稳压电源控制器。上海无线电七厂的W723在功能上与美国仙童的MA723相仿, 国产的



CW3420、CW3520 及 CW1524 与美国 Motorola 的 MC3420、MC3520 及 SG1524 系列相仿。

目前,我国已研制出适用于32位16位微机的开关电源。有的通过部局一级的鉴定。邮电部科技局对珠海经济特区通用电源厂生产的模块电源、铃流信号发生器进行了技术鉴定,认为达到了世界八十年代末的先进水平。珠海经济特区通用电源厂在1992年发布了国家认可的 Q/ZTD001-92 和 Q/ZTD002-92 二个企业标准,内容分别是 DMZ 系列直流变换器通用技术条件和 DMA01 系列铃流信号发生器通用技术条件。直流变换器可按输出功率的不同来划分成若干个系列,表示为:

DMZ ××—NNN XYZ<sup>注①</sup>



输出功率:几十瓦到单体1千瓦,允许多个并联序号1~6

开关电源的基本参数定义为:

(1)输出电压  $V_o$ ——在环境温度为25℃下,负载为额定负载,输入电压调至标称值时测得的输出电压;

(2)输出电压调节范围  $V_{oadj}$ ——电路中  $T_{rim}$  (调整端)与S端(或-S)短路情况下输出电压所得最低(最高)值的百分比。即  $\frac{V_{omin}-V_o}{V_o} \times 100\%$  和  $\frac{V_{omax}-V_o}{V_o} \times 100\%$  值

(3)电压调整率  $V_{ov}$ ——输入电压作上、下限波动时输出电压变化  $V_{o1}, V_{o2}$  差值与标称值的百分比,即  $V_{ov} = \frac{V_{o1}-V_{o2}}{V_o} \times 100\%$

(4)负载调整率  $V_{ol}$ ——负载在满载和20%满载下输出电压变化  $V_{o1} \sim V_{o2}$  差值与标称电压百分比。即  $V_{ol} = \frac{V_{o2}-V_{o1}}{V_o} \times 100\%$ ;

(5)动态响应  $V_{od}$ ——动态响应是指当电子负载以高频(1KHZ)在百分之二十额定负载(0.2 $I_{nom}$ )及满载( $I_{nom}$ )之间阶跃变化时输出电压的恢复时间。恢复时间是指负载阶跃时输出电压恢复并稳定在98%~102% $V_o$ 范围内的时间;

(6)限流点——本系列DC/DC变换器在过载状态时具有恒流保护功能。调节电子负载超过  $I_{nom}$  直至输出电压下降,输出电流不变,此恒定电流值即为限流点。

目前电力电子学术活动都不约而同的向开关电源、不间断电源等专题倾斜,无疑对我国赶上世界先进水平是大有裨益的。

① 注:输入电压类别 A:9~18VDC, B:18~36VDC, C:36~72VDC

输出组数 S 单组, D 双组, T 三组, Q 四组

## PC 机教学游戏软件

PC 机现正以较高的性能价格比为顾客所接受,走进千家万户。“好马配好鞍”“红花虽好,还要绿叶来扶,”国家教委中小学计算机教育研究中心组织力量,研制出了一批专供 PC 机用户使用的教学游戏软件,我刊将陆续予以介绍,并在适当时通过读者服务部经销,具体时间将在杂志刊登。请您留意——编者。

## 华容道

华容道游戏起源于我国三国时期的一段历史故事:东汉献帝建安年间,首相曹操率兵沿着华容的小道逃跑碰到诸葛亮的伏兵,后来在关羽的帮助下,才逃出华容道。软件设计者将民间传说故事搬上了计算机,在屏幕上再现了曹操以及关、张、赵、马、黄等历史人物在华容道上的一场智斗。

本软件是做为益智性游戏供用户娱乐用的,包括标准布局、人控布局、标准走法、人控走法几个项目。用户可以利用软件自己动手设计游戏布局,也可以采用标准布局,可由自己操作走,也可以在标准布局下查看标准走法,使用方便,操作灵活。

游戏共设十个棋子,要求游戏者利用棋盘上的两个空格,移动大小不同的十个棋子,游戏的结局是将曹操移出底部的缺口。

软件装在一张1.2MB或1.44MB的高密软磁盘上,可在相应配置的IBM-PC系列及其兼容机上运行,适用VGA彩色显示器。界面美观、大方,使用方便。

简单操作:

游戏主屏幕的下方,开设有用户操作窗口,对每一操作,都给出了具体提示。用户可利用这个窗口,辅助自己操作游戏。

(一)软件的启动:

开机引导后,软件显示片头、研制出版单位等。按任意一键后,软件进入主屏幕。主屏幕的右上方设置了选择菜单区域,用户可利用它进入每个选项。右下方有走步计数区域和当前时间表示。

(二)选择布局:

标准布局是流传最广的布局“横刀立马”,软件对它配有标准走法。

用户也可利用软件进行自控布局,进入自控布局后,依棋子大小显示选择框,当选择框移到用户满意的位置后,可按回车键确定放置,十子放完,返回主菜单。

(三)走法

对标准布局,软件给出当今世界记录即81步的标准走法,用户可以在人控走法项中,自己对各种布局包括标准布局和自控布局进行操作游戏。

# 电路滤波器及使用方法(一)

李兰友

(续上期)

## 4. LTC1064

LTC1064是八阶开关电容滤波器。LTC1064系列有五种型号,即多功能滤波器LTC1064,巴特渥斯特性滤波器LTC1064-2,贝塞尔特性滤波器LTC1064-3和考尔特性滤波器LTC1064-1、LTC1064-4。

典型参数如下:

### LTC1064-2

滤波器特性	八阶巴特渥斯特性
衰减特性	48dB/oct
截止频率范围	0~140KHz
截止频率控制方式	时钟, $f_{CLK}/f_c = 50$
失真率	0.03以下
噪声	$83\mu V_{rms}$
失调电压	$\pm 30mv$
工作电压	$\pm 2.37V \sim \pm 8V$
消耗电流	$\pm 12mA$ (typ)

### LTC1064-4

滤波器特性	8阶考尔特性
衰减特性	80dB/oct
截止频率	0~140KHz(70KHz)
截止频率控制方式	时钟 $f_{CLK}/f_c = 50(100)$
失真率	0.03以下
噪声	$80\mu V_{rms}$
失调电压	$\pm 30mV$
消耗电流	$\pm 12mA$
外型	14脚 DIP

图10为LTC1064典型用法线路,图10(a)

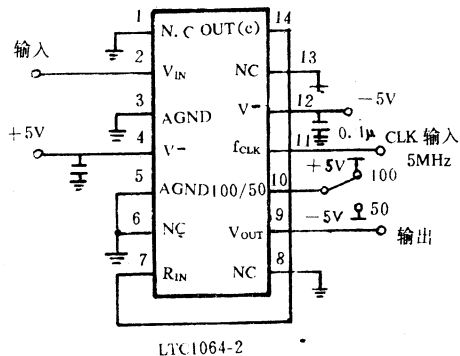


图10(a)

为LTC1064-2(巴特渥斯特性),图10(b)为LTC1064-4(考尔特性)。这里注意LTC1064-2的引脚①和引脚③均不使用而接地,而在LTC1064-4的用法中,引脚①和⑬之间接一补偿电容。

图10(a)电路的实测参数如表4所示,实测频率特

性如图11。

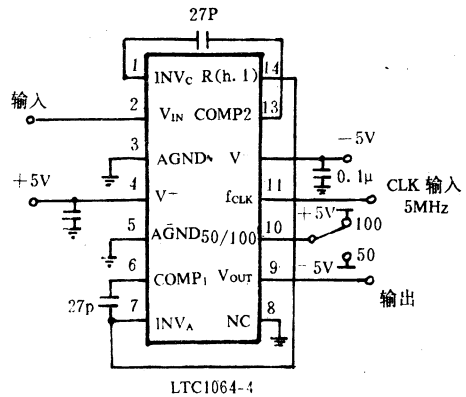


图10(b)

表4

$f_{CLK}$ (Hz)	$f_c$ (Hz)	失真率 (%)	噪声 ( $\mu V_{rms}$ )	失调电压 (mv)	消耗电流 (mA)
5M	100K	0.05	90	39.6	$\pm 13.8$
500K	10K	0.16	15mv	41.5	$\pm 11.1$
50K	1K	1.78	430	40.7	$\pm 10.8$

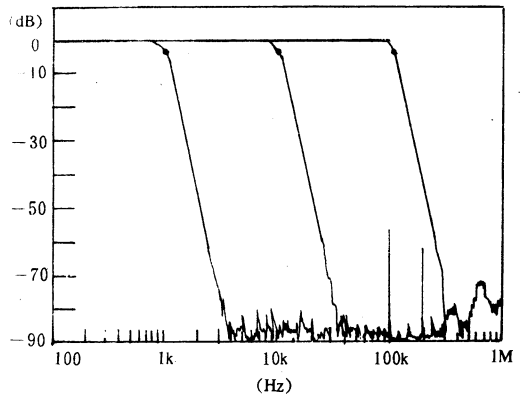


图11

LTC1064-4电路(图10(b))实测参数如表5。

表5

$f_{CLK}$ (Hz)	$f_c$ (Hz)	失真率 (%)	噪声 ( $\mu V_{rms}$ )	失调电压 (mv)	消耗电流 (mA)
5M	100K	0.12	90	-22.5	$\pm 15.5$
500K	10K	0.32	700	-20.1	$\pm 12.7$
50K	1K	0.18	280	-21.5	$\pm 12.4$

# 计算机多媒体技术的新信息

文 波

多媒体是通信工业所发生的深刻变化的最显著的体现。

多媒体技术的产生既塑造着变化中的通信工业，也受着通信工业的影响。

长期以来作为计算基础的数字技术在过去十年间广泛应用于声音、语音和数据通信以及图象。而多媒体正是这些数字技术的结合，并以不同的方式操作。

不久的将来，对于原始程序来说，它有可能通过微机上的展开页来完成，也有可能通过语音在办公室的电话机上完成，它既可通过麦克风来记录，也可通过摄像机来存档，具体使用何种方式已无关紧要，因为它们只是以不同的方式分享相同的数字语言。

从许多方面来讲，多媒体的影响已经产生了一个重要的组合：IBM 和时代华纳公司合作提供新的有线电视节目；日本的索尼公司已经进入好莱坞和音乐行业；苹果公司和东芝公司联合开发 Newton 技术的个人数字助理。飞利浦公司已开始销售数字音带和光盘。

尽管这些机构的主要目标是利用多媒体的一些专门技术，但是他们销售的产品只限于一些为人所熟知的模式。

远程通信行业部分变化更深刻，这些变化直接影响其本质。

已经对通信服务有很大影响的软件将成为这一行业的推动力量。

年收入将不再来自于过去100年间为人所熟悉的、非常规范化的基础服务，而将来自于象多媒体这样的难于规范的、更先进的、网络独立的服务。

象多媒体这样，把网络和服务分开，其意义是深远的：远程通信管理和网络操作员将不再因为他们控制网络而拥有网络控制。在载体从使用网络的服务提供者那里获得收入的同时，大笔的收入将从这些服务中得到，而不是从网络得到。

观察家们说，多媒体是这些变化的最明显的体现，同时也是一个重要的推动者。

英国的 Cambridge Analysys Ltd 的经理 David Cleevly 先生认为，多媒体将成为从基本服务向增值服务转变的关键因素。

“基于局域网和住宅消费商品的多媒体的发展会产生大量的交通以及高值连接和处理方面的要求，” Cleevly 先生认为“最终这将为远程通信操纵者提供大笔的收入”。

从许多方面来说，这些变化是对用户经营方式的复杂化的反应（用户希望得到容易使用、性能价格比高的产品）：通过电视电话会面，在个人机上进行项目合

作，而不必考虑合作双方之间的实际距离；为不能去医院的病人作诊断；根据音像信息注释处理保险赔偿请求；通过集成的实时音像和计算机显示进行货币交易；还可进行远程法庭审判等。

这其中的某些应用正在试验过程中，尽管价格非常昂贵，有些已投入使用。这一新型产业的带头人——高技术系统和服务的提供者，已经为下一代通信安排好了日程。

## ISDN 的角色

可笑的是，多媒体的成功将依赖姗姗来迟、并受到许多人攻击的综合服务数字网。

ISDN，尽管它有许多局限性——低性能，通信拥挤，不同国家间多变的标准和安装，以及昂贵的终端系统，但它至少提供了终端数字通信和对多媒体应用的推广至关重要的、图象传输所必需的足够的带宽。

而且，它的多路64KB/S 通道允许对语音、数据、图象的集成和用户之间的交互通信。例如，几个 PC 用户可以在通道上讨论他们都能看到并能操作的文件。

但是在美国 ISDN 的安装是很粗糙的，欧洲的情况也不好。欧洲的商业客户能获得 ISDN 服务的范围是从希腊、意大利、葡萄牙、西班牙的40%以下到丹麦、法国、英国的80%以上。

而国家之间的连接即使有，也只有有限的线路。

广泛的安装 ISDN 是对欧洲的通信产业的潜在帮助。连同 ITTCC 的视频通信和视频压缩标准 H. 320 和 H. 261，安装 ISDN 网将使欧洲在视频应用方面超过美国。

当然限制还是有的：窄带 ISDN 只能允许用最小的带宽来传输相当原始的视频图象，它不能支持高质量、全彩色、大屏幕、全动作的视频图象。

下一步将是在局域网中使用宽带和光纤。这将消除窄带 ISDN 性能方面的限制，但其广泛应用恐怕得要到下一世纪。

飞利浦公司音像服务市场经理 Johann Bialezki 先生认为，未来五年，以异步传输模式宽带交换技术为基础的宽带网络将支持多媒体技术的发展。

## 夸大还是现实？

目前，存在着推出独立于电话系统及其操作者的公共国际市场服务的可能性。

随着音频电传和智能网服务的发展，这一可能已经开始变为现实。

但是由于任何人都有可能提供增值的、多媒体方式的服务，网络和服务间的关系可能被整个破坏。它将象那些使用额外付费电话号来提供商业建议和彩票的

独立的音频电传提供者那样工作：电话公司提供号码，从服务提供者的电话费里分成。更好一点，电话公司可以自己提供这种服务或承包出去。

多媒体的提倡者们认为多媒体服务将成为比音频电传大得多的行业。

但是，所有这一切都是不确定的。位于麻省的 Cambridge 的 Forrester Research Inc. 认为围绕着多媒体有许多夸大之处，多媒体的需求非常之小，以至于多媒体系统提供者将不得不免费赠送。Forrester 公司预测多媒体最终将占有 15% 的应用市场，“任何把注意力放在虚幻的‘多媒体市场’的供应商将死亡”。

在四月份一个比较乐观的调查中，瑞士 Basel 的 Prognos 公司估计到 1995 年，美国市场的多媒体系统和服务将达到 56 亿欧洲货币单位 (68 亿美元)，而同时西欧市场将达 28 亿欧洲货币单位。Prognos 公司估计，其中 9 亿将来自于象基础课程的创建、培训信息、商业销售支持以及电视会议等服务。

Prognos 公司的研究仅限于用户提供的的应用，它并未研究与网络有关的服务的市场。

Prognos 公司相信除去用户市场，大多数的需求将是电视会议、培训以及商业销售支持。公司顾问 Holger Delpho 先生认为，到那时公共教育在美国也将成为一个因素，但在欧洲。

位于旧金山的资产管理公司 RMC Capital Management 的高级项目经理 David Palmer 对多媒体及其相关服务的重要性充满信心。

Palmer 说：“它将成为我们这个世界的一部分，并将对我们的生活方式产生重大影响”。

他相信在美国至少有线电视公司和贝尔公司会争夺提供新服务的市场。“网络操作者将比硬件制造者投入更多去定义多媒体”。

### 谨慎的计划

在英国，BT 和 IBM 已经开始谨慎地讨论多媒体所允许的服务。

BT 说它打算通过租赁的线路和其它 ISDN 载体向全球推出多媒体服务。由于是数字传输，传输质量与传输距离无关，用来提供服务的系统可以放在任何地方。它的服务甚至可以卖给第三方。

如果载体系统能处理这些新的服务以及新的服务规定，这一切至少是可以实现的。正如 IBM 英国公司视频电话部经理 Chris Frost 先生所说，在大多数情况下，载体系统是不够灵活的。

在许多国家，规范也不十分清楚：如果实时视频图象是多媒体的一部分，那么是否因为它包含语音而构成基本服务？或者因为它不只是简单的声音而把它认为是增值服务？

IBM 的 Frost 先生将服务分为四类：

象电话簿一样的，能提供很多人都想获取的相同信息这样一类的传统的服务器；

人们可以在不同地点相互通信的虚拟会议室；

个人的，人与人之间的信息服务；

虚拟购物所，用户可以从屏幕上的“商店”浏览购物：有些对所有人开放，有些只对那些付门票的人开

放。

一种服务可以是一经请求即提供电视服务。例如。一个游客可以接通电视公司的视频服务器——一种存储数字视频图象而不是数据的数据库——来看一下当天的主要新闻，甚至可以看实况节目。最终付费电话也将有视频屏幕；AT&T 和日本电报电话公司已经提供屏幕电话和 ISDN 付费电话，尽管它们还没有视频能力。

一个主要是面对商业用户的早期多媒体服务可能是多点的个人机电视会议服务，BT 和 IBM 把它作为联合开发 PC 视频电话的起点。这个联合体使用 IBM 的软件和一个外插在 BT 和 Motorola 共同开发的视频/图象/数据片上的外接板。

BT 公司音像服务部证券和商业发展经理 Graham Mills 先生认为，完善的视频电话产品会在 93 年下半年得到。因为 IBM 和 BT 想鼓励其他厂家采纳这一技术，所以可以免费得到设计规则。

视频电话，不管是数字的、模拟的、还是基于 PC 的，都代表了 BT 公司通向多媒体的三个方向中的一部分。其它两部分是向宽带 ISDN 发展的 ISDN 网和增值多媒体服务。

Mills 先生认为早期的服务将会和 PC 视频电话一起推出，尽管他没有提供细节。

### IBM 的参与

IBM 的 Frost 先生也不愿意在新的服务推出之前讨论它们。他说 IBM 对提供服务没有兴趣，它集中精力于向服务提供者销售设备。

但是，IBM 德国分公司已申请向它的主要客户提供多媒体网络服务的执照。德国成了 IBM 希望向用户提供的语音——数据网络服务的试验场。这一服务将使用户可以通过 IBM 全球信息网的德国部分，利用 IBM 的多功能工作站进行语音、数据、视频通信。IBM 将成为第一批向德国电话服务垄断挑战的挑战者。

同时，在美国 IBM 正在开发信息和家庭购物服务的伪多媒体服务。

视频服务器的一个变种是存储可立即调用的图象的视频图象库。

例如，在巴黎一个由 IBM、GSI 和 Eucom 公司共同组建的 Eurotop 公司最近推出了一个叫作“电子手册”的为旅行社服务的 PC 产品。

旅行代理和顾客可以在屏幕上看旅行手册的静止视频画面。画面可以通过与 IBM 3090 上运行的视频图象库相连而不断地更新。旅行代理可以利用在 OS/2 操作系统上运行的 PS/2 系统，通过 ISDN 网来操作恢复图象。

Eurotop 软件包是作为欧共体支持的信息市场政策法的一部分开发的，包括了欧洲几个最大的旅行社。

但是 IBM 的主机过于昂贵，而 OS/2 操作系统的使用也很有限。

尽管如此，越来越多的 PC 多媒体应用和系统正在涌现，正如 Forrester Research 公司所预测的那样，只要使用户相信他们真的需要多媒体，到 1995 年多媒体的许多特征将成为标准。

# 传 真 概 述

张景生 张建军

编者按:传真机作为办公自动化设备在国内的军事、邮政、金融、政府、企业等社会领域已广泛应用,在发达国家及地区已进入家庭。上海年产40~60万台传真机生产线已经组建。预测在未来的4年时间里,国内传真机将按年增长率30%左右的速度递增。为使读者了解掌握传真机知识,本刊将开辟讲座,系统介绍,分九期登出。

### 一、传真通信的用途及特点

把记录在纸上的文字、图表、相片等静止的图象转换成电信号,经过线路传输到遥远的地方,在收信端获得与发送原稿相似的记录图象的通信方式,称为传真通信。

传真通信不但可以传送文字、数据、图表,还可以传送真迹。也就是说,利用传真机可以在收信端获得发报人的签名、手迹,有特殊的应用价值。既可节省通信费用,又可提高办事效率。所以,在高速运转的信息社会中,传真通信给现代通信技术赋予了新生命,给办公自动化带来了高效率。

尤其是半导体器件的超大规模集成化和采用微电脑技术后,使得传真机体积小、功耗低、功能全,使用简便、可靠性强,达到无人值守的全自动化程度。利用先进的传真机作为电子计算机外部设备,可完成输入图片、文件、手迹、签名、印章和记录显示、打印等任务。在通信管理方面也可以利用数字化技术寻址、标识、登记计数等。传真通信趋于系统化、网络化、智能化。

### 二、传真通信的基本过程

要将一张图象(文件、图表、相片)完整无缺地传送到对方,首先需要将图象分解编码,也就是把发送的图象分解成很多微小单元,并按照一定的顺序利用一种光电器件 CCD 转换成不同强度的电信号,再经过整形、放大编码、调制等处理后,通过有线或无线信道传送到接收端。在接收端把接收到的电信号加以放大,解调处理后,再转换成相应的微小单元象素,并把把这些微小单元按照与发送端同样的顺序采用电热能的转换器件进行转换——感热记录合成图象记录下来。这就是传真通信的基本过程,即发送扫描→光电变换编码→调制→传输→解调→译码→接收扫描→记录变换。

### 三、传真通信的同步和同相

要在收端获得与发端完全相同的图象,发送机和接收机必须“步调一致”。发送扫描自左至右自上而下一行一行地扫描,就要求接收扫描也按照同样的顺序自左至右自上而下地进行,就是要求“分解”与“合成”按相同的规律进行,否则记录出的图象就会出现歪斜,乱七八糟甚至难以辨认。

“同步”就是收发双方扫描速度的一致性。如发送原稿中部有一条垂直黑线,在发送时自左至右自上而下地扫描,并把图形传到对方,在收方也自左至右自上而下地扫描记录图象。如果发收扫描速度完全一致,就是“同步”,那么记录的图象也是一条垂直的黑线,如果发的扫描速度快而收的扫描速度慢,那么每收一条就要比发一条的时间长。就会造成记录的图象向左倾斜失真。反之如果发的速度慢而收的速度快,那么每次收的扫描要超前发的扫描,记录出的图象向右倾斜失真。同步是保证记录出一样图形的重要措施。同步的方法很多,有电源同步、独立同步、传输同步等等。

“同相”就是发送机和接收机每行扫描的起点要相同。即发送机和接收机扫描图象时,必须都从发送原稿和记录纸的边缘开始,如果发端扫描从图片的边缘开始,而收端扫描从中间开始,结果就会造成记录出的图象分离、造成不同相。传真通信中的对相是由发方发送的相位信号和收方发送的相位信号相吻合而完成的。

### 四、传真机的主要技术参数

1. 扫描尺寸 无论发送图象和接收图象都是利用扫描把图象分成许多微小单元。因此,扫描的尺寸大小取决于图片的性质和对记录图象的要求。扫描点越小,复制出的图形越逼真。但扫描尺寸太小时,反射到光电器件的光通量又太小,给信号处理带来了困难。

在平面扫描中,一般采用 CCD 作为发送扫描过程中的光电传感器,接收记录扫描是利用感热头作为电——热能转换器。主扫描方向上的点数为8点/mm。

2. 扫描线密度 由于扫描是连续进行,所以画面是象素的集合或是象素宽度的许多线的集合,这些集合组成的线称为扫描线。扫描线在一定宽度的图象中越多,收信图象的分辨率就越高。因此,扫描线根数可作为分辨率的单位,在副扫描方向上,其单位长度内的扫描线根数称为扫描线密度。单路三类机扫描线密度有标准的3.85线/mm,精细7.7线/mm和超精细15.4线/mm。

3. 扫描线长度 扫描点沿主扫描方向扫描一行距离称扫描线长度,单位为mm。在滚筒扫描中,即为滚筒的周长  $L=D$ 。在平面扫描中,扫描线长度等于扫描头的有效宽度。由于平面扫描器在送纸时,可能有左右偏移,因此扫描线全长应稍大于纸宽。

4. 主扫描速度  $N$  是指单位时间内对图象进行主扫描的次数。在滚筒扫描中, $N$ =滚筒转速。CDD 平面扫描方式中, $N$ =时钟频率。

副扫描速度  $V$  是指在单位时间内扫描元件在副扫描方向上所扫描的距离。它是建立主副扫描之间联



系的一个重要参数。在滚筒扫描方式中,即为光学系统和记录器移动的速度,在平面扫描中,即为记录纸行走的速度。

5. 合作系数 “合作系数”是表征发送图象和接收图象的长宽尺寸应符合一定比例的参数。收发两端所用的传真机尽管是不同的式样型号、不同厂家,只要它们的合作系数相同,接收到与发送图象呈比例的记录图象,也就是说,复制出来的图象可以同发送图象一样大小,也可以按某一比例放大或缩小。在平面扫描中  $D = L/\pi$ ,其中  $L$  为主扫描行长度,  $d$  为扫描线密度。

6. 图象频率 传真通信中把由图象经光电转换器转换成电信号的频率统称为图象信号频率。由于实际图象中线条宽度是随机的,所以图象频率的频率范围较宽。在传真通信中,把宽度等于扫描线密度倒数的线条转换成电信号的频率称为最高图象频率。最高图象频率是传真机在考虑传输频带时的重要参数,也是考虑频带压缩时用的基准频率。

7. 传送时间 传真机中传送一页文稿的时间叫传送时间。传送时间由原稿尺寸,扫描线密度以及传输线路中允许传输频带的宽度决定。

单路传真机当传送 A4 幅面的文件,扫描线密度为 3.85 线,传输频带为 300Hz—3400Hz 时,传送时间的标称值可为 15 秒种至 1 分钟之间。

8. 输出电平 考虑到传真信号在市内中继线上传输衰减相差很大,所以发送机输出到线路的电平能在一定范围内调整(三类机 0db 到 -15db 之间)。对于衰减较大的中继线线路,需提高传真机的发送电平,对于衰减较小的线路,则需要降低发送电平。发送电平合适值应以传真信号送入长途电话接口处载波机对该点传真信号功率电平额定值为准。如果传输电平比额定电平低得多,传输中容易受到线路串音和杂音的干扰,而且长途电话有其净衰减,这个净衰减随着长途音频转换段数的增加而增大。这样,传到终端时,传真接收机的输入电平就很低了,信杂比就恶化,记录质量将变差。反之,信号电平太高,造成载波机内非线性元件和部件的过负荷而产生非线性失真,也会使传真质量受到影响。一般传真机在出厂时发送电平平均调整在 0db 电平上。

9. 输入电平 传真机的输入电平限值在保证正常接收的条件下不低于 -40db,传真机的接收电平如果过低。则对话路的杂音要求就高,传真信号的传输质量就不易保证。所以一般规定在 -40db。

## 五、传真机分类

### 1. 真迹传真机

它主要用来传送文字和图表。而且传送的是黑白两种色调,故又称“黑白传真机”。真迹传真机可以用电话机配合使用,不论市内或长途,在接通电话进行联系后,就可利用这条话路来传送真迹传真。

真迹传真机又分为用一个话路传输的单路真迹传真机和用载波机的一个基群或一个超基群传输的多路真迹传真机。

### 2. 相片传真机

用于传送照片的传真机称为相片传真机。它不但传送“黑白”信号,还可传送介于黑白之间的灰色中间层次,以保证照片的清晰和逼真。相片传真机也用同一个电话电路传送,扫描速度比真迹传真机慢。

### 3. 报纸传真机

它是一种高速大滚筒的真迹传真机,它可以将整版报纸进行传送,报纸传真机一般利用微波信道传输。

### 4. 气象传真机

它被气象部门用来发送、接收气象图,以便及时预报气象消息。气象传真一般采用无线广播发送的方式,它一般采用无线短波、超短波和微波信道。

另外以速度划分,国际上把单路真迹传真机分以下四类:

一类机简称 G1 凡是采用双边带传输,其发送信号不采取特殊压缩措施,并能在一个电话电线上用约 6 分钟的时间传送一份 A4 幅面(标称尺寸 210mm × 297mm)文件的传真机称为一类机。

二类机简称 G2 凡是采用模拟频带压缩技术,达到约在 3 分钟的时间内,在一个电话电路上传送一份 A4 幅面文件的传真机称之为二类机。

三类机简称 G3 凡是采用数字频带压缩技术(即用压缩信息多余度的方式),在电话电路上约 1 分钟时间内送一份 A4 幅面文件的传真机称之为三类机。

四类机简称 G4 由于回线传输质量的限制,接收的图象质量与传输时间已达到极限。而四类机利用数据网进行传真,可以越过这些,得到我们满意而理想的传真信息。此类设备具有在传输前减少信号中冗长度的手段,并采用适合数据网的传输控制程度程序,可以进行无错码接收。此外,采用适当的调制方式,还可在一般电话交换网内使用。

## 六、传真机通信线路

1. 传真通信的有线电路有下列四种:

- (1) 音频实线电路
- (2) 载波电话电路
- (3) 群载波电话电路(即由若干话路组成的群路)
- (4) 脉冲编码调制和增量调制的数字电路

短距离的传真电报可在实线电路上进行。例如市内传真通信网,企业内部的传真通信线,传真机到载波电话机的中继线,以及短距离的传真专线等。

目前使用最广泛的传真通路是载波电话市话通路,其通频带宽度为 300Hz—2700Hz 和 300Hz—3400Hz 两种,分别称窄带电路、宽带电路,在载波电话通路上的传真通信距离相当于长途话路的传输距离。

2. 传真通信的无线通路主要有下列三种:

- (1) 短波通路
- (2) 微波通路
- (3) 卫星通信通路

采用短波通信时,传真通信的速率将受到一定程度的限制,一般不超过 275—330 毫米/秒。微波通路工作于厘米波或分米波。

## PC 机之间最简单的通信

申老师：小王，小李，今天给你们介绍上次讲过的“三根线，四语句”的 PC 机之间最简单的通信。咱们一边讲，一边共同动手做。所需的设备和材料是：二台 IBM PC/XT、AT 或兼容机（均需带有串行通信口），RS-232C 通信电缆一根，BASICA 解释程序一套。

注意，在 IBM PC 原装机上可以使用 IBM BASICA，这次的实验甚至用 BASIC 也可以，但是，若使用兼容机，则一般不能运行 IBM 的 BASICA，而应当使用 GWBASIC 等功能与 BASICA 相当的其他版本的 BASIC 解释程序。

RS-232C 通信电缆，最好有 3 米长，两头都应当接有 DB-25 插座，而且两头都是阴头。如果手头没有 DB-25 插座，也可以直接用三根导线，每根导线的两端均焊上一个筒形小簧片，以便直接套插到 PC 机串行口的插针上。另外还需再焊出 4 根短路线来。按图 1 所示的连接方法，按编号找准插针，将两台 PC 机的串行口连接起来。

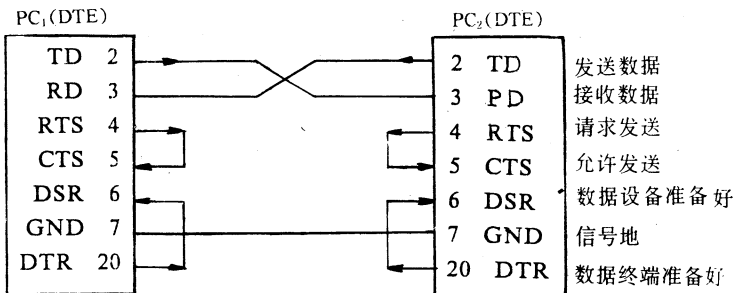


图 1

小王：2和3为什么要交叉连接？

申老师：请看这二个信号的名称，TD是“发送数据”，RD是“接收数据”，一台机的发送，必定是另一台机的接收，所以要交叉。7是信号地，收、发信号都以它为参考电位。

小王：4到5，6到20的短路线起什么作用？

申老师：这些本来是要互相连向对方的控制和应答信号，也称“握手信号”。为了简单起见，就把自己发出的请求信号，当作是从对方发来的回答信号进行接收，例如从4发出，5接收，以便使异步通信接口电路能正常工作。从20发至6也是这个道理。

小王：申老师，这一台 PC 机的串行口，用的不是 25 针插头，而是 9 针的小插头，怎么回事？

申老师：是的，原 RS-232C 规定的 DB-25 插头 25 芯的信号，太繁杂了，比如其中还有两组老式设备才用的“20mA 电流环”等等。在实际使用中常常简化到

只需 9 线以下，所以近来越来越多的 PC 机采用 DB-9 九针插头，其与 DB-25 插头常用信号的对应关系表如图 2 所示。可按此表进行连接。

DB25	DB-9
2	3
3	2
4	7
5	8
6	6
7	5
8	1
20	4

图 2 DB-25 与 DB-9 插座信号对照表

连接完成后，启动两台 PC 机，并且都运行 BASICA，显示 OK 后，在其中一台上键入如下程序，作为发送方：

```

10 OPEN "COM1: 1200, E, 7, 1, RS, CS, DS"
   AS #1
   (打开通信口 COM1 至 1 号通道；)
20 K$ = INKEY$ : PRINT #1, K$ : GOTO
   20
   (键入一字符，并从通信口发出。循环直至用 Ctrl
   -Break 打断。)
   在另一台机上键入如下接收程序：
30 OPEN "COM1: 1200, E, 7, 1, RS, CS, DS"
   AS #1
40 INPUT #1, A$ : PRINT A$ ; : GOTO
   40

```

(从通信口接收一字符，并在屏幕上显示出来。循环直至用 Ctrl-Break 打断。)

两台机都键入 RUN(ENTER)，使程序运行。在发送方的机上从键盘输入一些信息，如“Good morning!”，就可以发现信息已传到接收方的屏幕上。

(小王、小李按照此法做了，一次就连接并传输成功，很是兴奋。但他们不满足于已取得的成功，而是想学习更多的知识，他们继续向申老师请教。)

小王：语句 10 的意思，是把设备“COM1:”当成一个文件看，把它打开在 1 号通道上。但其中有几个参数“1200, E, 7, 1, RS, CS, DS”是什么意思？

申老师：表示采用传输速率为每秒 1200 二个进制数位 (1200bps)；校验位采用偶校验 (E)；数据位为 7 位；停止位为 1 位。串行通信数字信号的波形见图 3 所示。

"M" = 

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

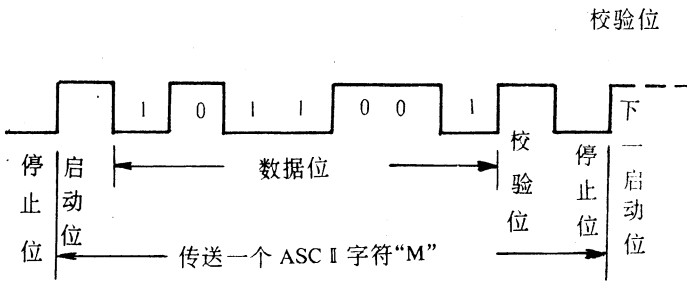


图3 字符“M”的异步串行通信数字信号

从图上可见,在这种异步通信方式中,除了数据,还要加上联络用的启动位、停止位,以及为了提高传输准确率而采用的奇偶校验位等信号一起传送。为了传送一个 ASCII 字符(7位),差不多要用10个比特。所以1200bps 速率用来传送字符时,实际只有每秒120个字符左右。

校验位你们知道吧?就是用该位的置“0”或置“1”,来凑足该组信息中“1”的个数为奇数或偶数,分别称为奇校验(O)或偶校验(E)。传送到对方后再核查,如果发现“1”的奇偶数不对了,说明传输过程发生了错误,可以要求重发。这在用电话线进行传送的情况常常发生,因为我国不少地区电话线质量不够好。

停止位选1位或2位即可,一般选1位。

RS,CS,DS,是表示对控制信号的某些控制,有关的说明要查手册了。

小王:传输速率可以在什么范围内选取?

申老师:在这种用 RS-232 电缆直接传送的情况,可在300-9600bps 范围内选择,最高可达9600bps;但以后要学到的用电话线传送的情况,常常只能在300~2400bps 内选取。电话线质量好时,可选2400,差一些降一半为1200,再差再降为600,甚至300。

小王:这个程序可用来传送磁盘文件吗?

申老师:以上这个简单实验,只能传键盘信息,目的是为了让大家捅破 PC-PC 通信这个“窗户纸”。能够传磁盘文件的程序要复杂一些,以后再介绍。

小王:今天我们在老师辅导下实验做得很顺利,如果是自己做不通时,应怎样检查故障?

申老师:如果做不通,可以从下面几个方面去检查:

1. 传输电缆接线是否接错?
2. BASIC 版本是否正确,注意兼容机往往不能运行 IBM BASIC,这时象是“死机”,出不来 BASIC 提示符“OK”。更换 BASIC 解释程序(例如 GWBASIC)往往就可以解决问题。
3. 串行通信口 COM1可能是坏的,因为平时不用不知道坏。可以用随机的诊断程序查查看;
4. 错把别的接口当成串行通信口;
5. 兼容机在配置时既安装了专用的通信接口板

(上有串行通信口),又插上了含有串行口的多功能板,造成有二个 COM1,无法正常工作;

6. 有的机器 COM1已接有鼠标器等其他设备,这时应改选 COM2,甚至 COM3,COM4;

7. 病毒污染,影响正常操作,应先清除病毒。

## 中软总公司库压品门市部

### 优惠向您提供下列产品

#### 一、计算机盘片:

3M. BASF. 五英寸,单面/双密,20元/盒。100片以上,15元/盒;3M. 五英寸,单面/双密,硬分段,16扇区,15元/盒。CE. BASF 五英寸,双面/双密,28元/盒。100片以上24元/盒。DATALIFE. 五英寸,单面/双密,20元/盒。100片以上18元(塑料盒);DATALIFE. 五英寸,单面/双密,30元/盒。100片以上28元/盒(塑料盒);BASF. MASTER,八英寸,单面/单密,60元/盒;3M 八英寸,单面/双密。80元/盒;JANUS. 五英寸软加密盘。双面/双密,10元/张(低版本)。

#### 二、盒式磁带:

BASF. 1/2×1200英尺。60元/盘;  
3M. 1/2×1200英尺。80元/盘;

三、8086单板机,500元/台,9250智能中英文打字机4200元/台;铅字打字机,油印机;日本1550九针打印机,1200元/台。60M 磁带机,1000元/台。24针打印机9400,2800元/台;

#### 四、绘图仪:

SPL——400,3000元/台;LP——3700,2万元/台;  
DXY——800,3000元/台;数字化仪,WT——4000,3000元/台。

#### 五、显示器:

12英寸彩显,800-1000元/台。14英寸彩显,1000-1300元/台。

#### 六、各种配置计算机:

IBM PC/XT. AT/286. 中华 386/16. 长城 GW0520CH. GW286。

七、主板机、各种显示卡、长城打印机汉卡;UPS400W、UPS1000W。

#### 八、其它:

0520A 电源,TP-80电源,打印机电缆线;光电鼠标器、稳压电源500W、1000W,扁线压接工具、实验版、起拔器、1500K 盒式色带、P3/P7盒式色带、元器件;5英寸、8英寸盘盒。

联系人:温友良 孟秀华

电话:8317722-1112

地址:北京市海淀区学院南路55号

电脑大厦112房间

邮政编码:100081

乘车路线:16路汽车电脑大厦下车