



● 一九九一年 ● 总期第72期

3

電子

144
744

與 電腦



• ELECTRONICS AND COMPUTERS •

广东省潮阳电声磁带厂

“曲泉”“曲泉”录音优美动听似流泉。“曲泉”

“曲泉”录像明澈透亮赛清泉。

潮阳电声磁带厂生产的“曲泉”牌卡式空白录音带和卡式空白录像带，

是引进国内外先进技术设备，进口美国西德磁带材料装配的产品。

“曲泉”牌卡式空白录音带一九八二年经国家第四机械工业部测试鉴定。各项质量指标均符合录制音乐、语言等要求。产品畅销全国。“曲泉”牌卡式空白录像带色彩鲜艳，明亮度强、品种多样、规格齐全。还可根据用户需要，提供各种特定长度的产品。

厂址：广东省潮阳县棉城东山

电话：2585 电挂：7820



大连磁带厂是化工部所属专业化磁带厂。厂生产的大连牌500系列广播录音磁带。在全国同行业评比中多次获奖。产品畅销全国，在用户中享有盛誉。由引进生产线生产的600系列盒式录音磁带已通过国家鉴定，质量可与国外先进水平媲美。

大连牌广播录音磁带荣获一九八〇年全国评比第一名、一九八四年化工部优质产品奖、一九八一年获国家银质奖。

厂址：辽宁省大连市金州区龙王庙

电话：230397 电挂：4318



大连磁带厂



中国电子器材公司特区公司

本公司是机械电子工业部派驻特区的电子产品管理和经营机构。本公司立足特区，全部产品直接货源，价格性能比均优，来人来函订购均可。竭诚为广大用户提供资料咨询，引进新产品代办进口手续，欢迎比较和建立长期合作关系。

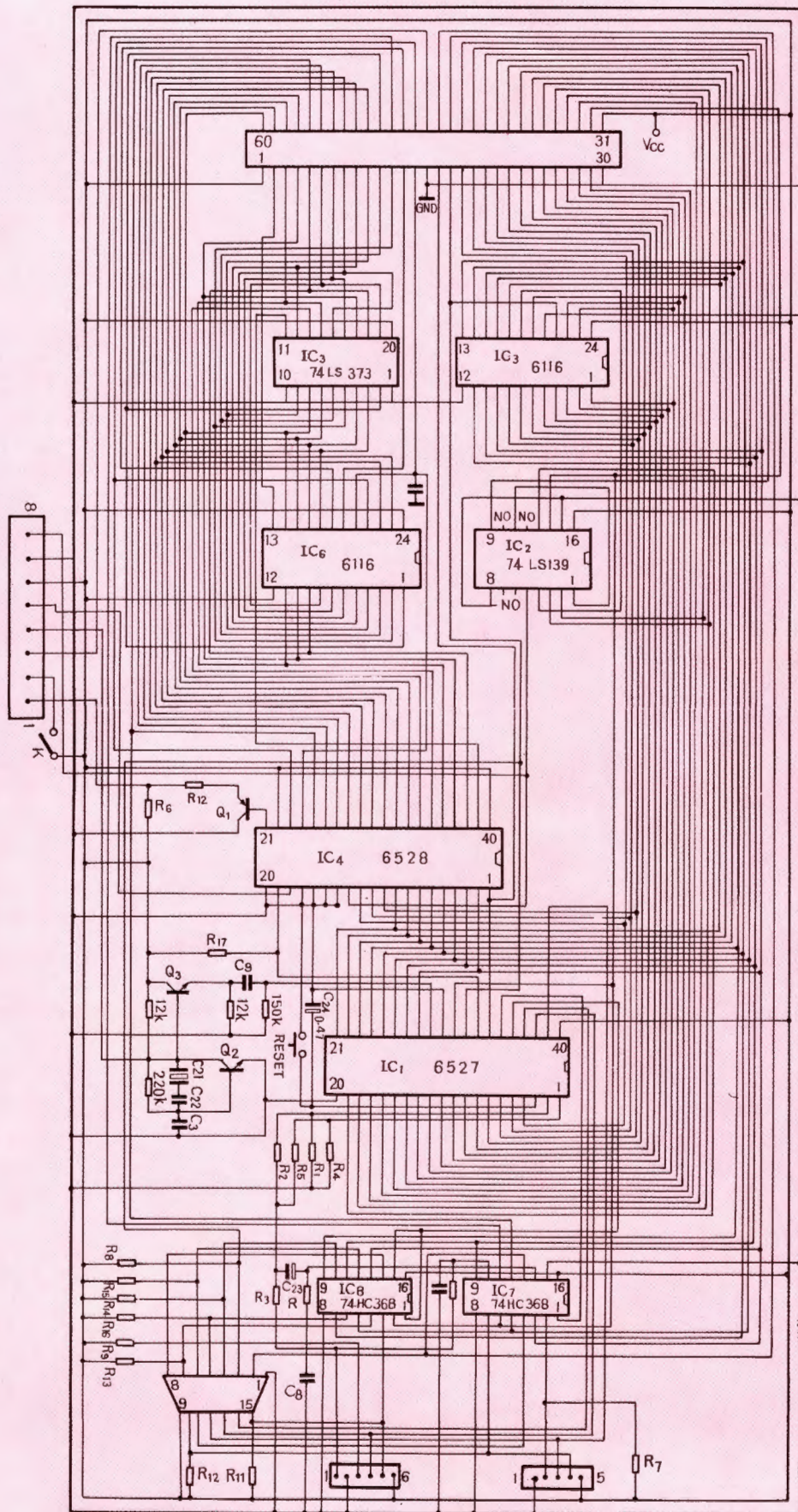
以下选介美国ZILOG公司部分集成电路期货价目表（参考）

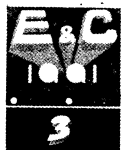
1991年3月1日

单价（元）

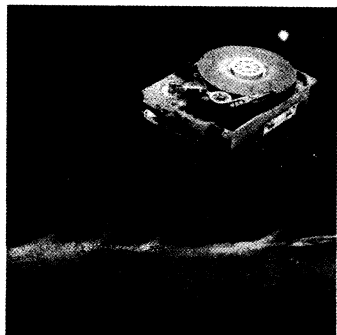
型号	功能	500只	1000只	2000只
Z0840004PSC	Z80 NMOS 4MHZ. CPU.	5.36	5.28	5.12
Z0840006PSC	Z80 NMOS 6MHZ. CPU.	7.04	6.96	6.72
Z84C0006PEC	Z80 CMOS 6MHZ. CPU.	11.60	11.36	11.04
Z0841004PSC	Z80 NMOS 4MHZ. DMA.	22.72	22.24	21.60
Z84C1006PEC	Z80 CMOS 6MHZ. DMA.	27.76	27.20	26.40
Z0842004PSC	Z80 NMOS 4MHZ. PIO.	5.36	5.28	5.12
Z0842006PSC	Z80 NMOS 6MHZ. PIO.	7.04	6.96	6.72
Z84C2006PEC	Z80 CMOS 6MHZ. PIO.	11.60	11.36	11.04
Z0843004PSC	Z80 NMOS 4MHZ. CTC.	5.36	5.28	5.12
Z0843006PSC	Z80 NMOS 6MHZ. CTC.	7.04	6.96	6.72
Z84C3006PEC	Z80 CMOS 6MHZ. CTC.	11.60	11.36	11.04
Z0844004PSC	Z80 NMOS 4MHZ. SIO/0	13.84	13.60	13.20
Z0844006PSC	Z80 NMOS 6MHZ. SIO/0	14.08	13.84	13.44
Z84C4006PEC	Z80 CMOS 6MHZ. SIO/0	21.60	21.20	20.56
Z0844104PSC	Z80 NMOS 4MHZ. SIO/1	13.84	13.60	13.20
Z0844106PSC	Z80 NMOS 6MHZ. SIO/1	14.08	13.84	13.44
Z84C4106PEC	Z80 CMOS 6MHZ. SIO/1	21.60	21.20	20.56
Z0844204PSC	Z80 NMOS 4MHZ. SIO/2	13.84	13.60	13.20
Z0844206PSC	Z80 NMOS 6MHZ. SIO/2	14.08	13.84	13.44
Z84C4206PEC	Z80 CMOS 6MHZ. SIO/2	21.60	21.20	20.56
Z84C4306FEC	Z80 NMOS 4MHZ. SIO/3	23.76	23.28	22.64
Z0844404PSC	Z80 NMOS 4MHZ. SIO/4	13.84	13.60	13.20
Z0844406PSC	Z80 NMOS 6MHZ. SIO/4	14.08	13.84	13.44
Z84C4406VEC	Z80 CMOS 6MHZ. SIO/4	21.84	21.44	20.80
Z0847004PSC	Z80 NMOS 4MHZ. DART.	13.84	13.60	13.20
Z0847006PSC	Z80 NMOS 6MHZ. DART.	14.08	13.84	13.44
Z84C0106VEC	Z80 CMOS 6MHZ CPU+OSC	11.20	10.96	10.64
Z84C9008VSC	Z80 CMOS 8MHZ KIO: SIO+PIO+CTC+PORT.	69.76	68.40	66.40
Z0765A08PSC	8MHZ FLOPY DISK CONTROLLER	16.80	16.40	16.00
Z8018006PSC	Z180 CMOS ENHANCED MPU	39.04	38.32	37.20

任天堂游戏机参考电路





電子
與電腦



• ELECTRONICS AND COMPUTERS •

744
3

一九九一年
总期第 72 期

電子與電腦

目 录

• 综述 •

磁盘和磁盘机(一) 林兼(2)

• PC 用户 •

数据库通用维护模块设计 徐维祥(4)

字典库的编辑和使用 黄焕如(7)

用 DOS 命令设计菜单 陈凯(8)

在 IBM PC/AT 机上配置虚盘 何管略(9)

IBM PC/XT 机异步通信适配器的自检

方法 杨沂 杨耘华(10)

产生真正的随机数 应海涛(11)

利用屏幕缓冲区实现语言的三重调用 张建新(11)

谈谈子目录加密与解密的一种方法 李连德(12)

PC 机显示氢原子轨道波函数

..... 陈可中 曾庆涛 郭冰莹(12)

BASIC 程序 P 加密的简易解密法 李志刚(13)

• 学习机之友 •

BASIC 语言实现递归 苏亚华(14)

数据处理原理和技巧 朱国江(15)

哥德巴赫偶数猜想的 BASIC 程序 陈君佐(18)

梵塔移动的动态模拟 刘闯(19)

查找内存中的代码 陈宇(20)

奇妙的图案 李明(20)

录

最简数据盘生成法 江东(20)

超级 8~40 单元打印 王壮(21)

将 16 进制代码翻译成 DATA 语句的数据

..... 张振堂(21)

关于多目录磁道问题 张剑平(22)

电子表键盘输入方式的改进 田洪瑜(22)

LASER310 兼做数字石英钟 郑嘉琦(23)

• C 语言初阶 •

第三讲 命令行 Turbo C 和实用程序 李文兵(24)

• 学用单片机 •

普及型单片机开发工具的软件设计

..... 张培仁 刘振安(28)

• 学装微电脑 •

学装 μ p-80 8 位 A/D 变换部件的制作 易齐干(31)

• 电脑巧开发 •

MP-I 微电脑加装编码键盘 李志平(36)

扫描式红外线十路遥控器 王刚勤(37)

• 初级程序员级水平考试辅导问答 •

微型计算机操作系统使用问答 王路敬(40)

• 封三 •

任天堂游戏机参考电路

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:北京市邮政局

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 副主编:王昌铭

责任编辑:张 丽

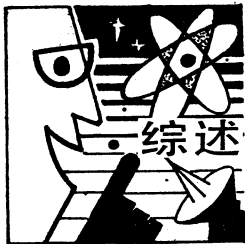
订购处:全国各地邮电局

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:0.95 元



磁盘和磁盘机(一)

林 兼

近来, IBM-PC 和它的兼容机(如长城 0520, ST-PC, HH-PC)逐步进入学校和家庭, 引起了越来越多的人的兴趣, 许多人也想了解它所用的各种外部设备。今天, 我们首先介绍 IBM-PC 所用的磁盘和磁盘机。

所有各种计算机, 不论大小, 都要有存储设备。就象人的头脑一样, 记住各种公式和数据。有些数据需要长期保存, 有些数据需要在计算机间交换, 就要把它存储在一个单独的“地方”, 就象人把数据写在纸上保存一样。这个“地方”在 Apple 机和中华学习机中用磁带和软盘片, 在 IBM-PC 中, 不用录音磁带, 因为它存取速度太慢, 可靠性也低。大部分管理程序, 如 MS-DOS 操作系统, 编好的程序, 游戏等等都是记录在软盘片上, 然后输入到机器中去执行。计算的结果也是记录在软盘片上取走。这样, 输入速度快, 交换也方便。要注意的是, Apple 机所用的软盘片和软盘机, 和 IBM-PC 所用的是不一样的, 并不通用。差别下面我们会讲到。

除了软盘外, IBM-PC 还可以配置硬盘机。目的是扩大存储容量和提高工作速度。

软盘片的种类

软盘片是目前微机最常用的存储介质。它由圆形的聚酯薄膜作片基, 两面涂复有磁性材料。外面有一个方形的塑料保护套。早期盘片直径是 8 英寸(200mm), 这种盘片现在不常用。最常见的是 $5\frac{1}{4}$ 英寸(130mm) 和 3.5 英寸(90mm) 的两种。前一种用于 Apple 机和 IBM-PC 机, 后一种用于 PS/2 和手提式计算机。它们的外形尺寸如图 1。除了尺寸大小不同外, $5\frac{1}{4}$ 英寸盘片保护套是低质、柔软的。3.5 英寸盘片保护套是塑料的, 硬的, 很好区别。

除了上述这几种外, 新研制的还有 3 英寸, 2 英寸等小型软盘片。但目前都没有广泛使用。

$5\frac{1}{4}$ 英寸软盘的种类

$5\frac{1}{4}$ 英寸软盘按用途分有三种:

1. 工作(记录)用盘片, 即通常存储数据用的盘片。

从图 1 可以看到, 这种盘片保护套上有几个孔。中间大圆孔是主轴孔, 当软盘片放进软盘机时, 主轴把盘片夹紧, 带动软盘片在保护套内旋转。保护套是不动的。中间主轴孔旁有一个小孔, 叫做索引孔, 它用来确定磁道的起始位置, 检查盘片转速。下面有一长圆孔, 是供磁头运动进行读写的。侧边有一缺口。当用胶纸把缺口封上时, 就不能进行写入, 以避免因误操作把软盘上重要的数据破坏了。通常叫它“写保护”。

2. 校准用盘片, 用来调整和检查软盘机。通常称为

CE 盘, 通俗叫法是“猫眼盘”。

这种校准盘片上面写有特殊的信号, 不是用一般复制的办法可以做出来的。因此价钱比普通的盘片贵几十倍。可是它在外形上和普通盘片没有差别, 如果不注意, 把它当普通盘片使用, 就破坏了它的功能, 造成不应有的损失。

3. 清洗盘片。由于软盘机的磁头是暴露在周围空气中, 难免沾上灰尘或其他脏物。最好要定期清洗。清洗有许多方法。采用清洗盘片是一种方法。这种盘片在外观颜色上与普通盘片不同, 因此不易混淆。

不同规格的工作盘片

我们可以把一张软盘片看成是一页书, 盘片的两面好比是一页书的两面。盘片的每面分成许多磁道, 一条磁道好比是书的“一行”。只不过磁道是一个个同心圆, 而不是一行行直线。每条磁道由许多位组成, 好比是每行中的一个字。数据就保存在这些位上, 每个位存储一个数据。在计算机中, 存储容量通常用字节(Byte)或兆字节(MB)表示。一个字节有 8 位。

对使用者来说, 最关心的事是一片盘能保存尽量多的数据。用计算机的术语说就是存储容量尽可能大。但软盘片存储面积是一定的, 要存储多必须提高单位面积的存储量。在每条磁道上单位长度上存储的位数叫做“位密度”, 单位是每毫米位数或每英寸位数(bpi)。而半径方向单位长度上的磁道数称为“道密度”, 单位用每毫米道数或每英寸道数(tpi)。

从存储容量上考虑, 当然是位密度和道密度越大越好。但是, 由于软盘片要互换, 要求在一台软盘机上写好的软盘片在其他同类型的软盘机上也能读出, 必须有统一的标准。目前, 国际标准规定了每个磁道的位置(同心圆的半径), 也规定了道密度。目前 $5\frac{1}{4}$ 英寸盘片只有两种道密度 48tpi 和 96tpi。位密度也只有两种常用, 采用改进调频制的 5876bpi, 称为“倍密度”。采用 9646bpi 的称为“高密度”。

总的说来, 软盘片有单面和双面, 有倍密度和高密度, 有 48tpi 和 96tpi, 排列组合, 可以有好几种不同的规格。

Apple 机和中华学习机采用的是单面软盘片, IBM-PC 机只用双面盘片。这是它们间不能互换使用的原因之一。

对于高密度, 只用 96tpi。因此, IBM-PC 机所用的盘片实际上只有三种:

双面	倍密度, 48tpi
	倍密度, 96tpi
	高密度, 96tpi

存储容量的计算

当我们打开一本书时,可以看到每面有多少行,每行有若干字。这两个数相乘,就得到每面可排下的最大的字数。但由于空行空格,标题等等占去不少位置,实际的字数要比最大字数少。不同的开本最大字数也不一样。

软盘片也一样,我们可以用每面磁道数乘上每磁道存储字节数得到它的“最大存储容量”。这个容量通常称为“非格式化容量”。例如,双面 48tpi 的盘片,每面 40 个磁道,每磁道存储 6250 字节,它的非格式化容量就是:

$$2 \times 40 \times 6250 = 500000 \text{ 字节} = 500 \text{ 千字节(KB)}$$

在使用时,不同的操作系统把软盘分成不同的格式。在 IBM-PC 机中通常用的 PC-DOS,把一条磁道分成 8' 个或 9 个扇区,每个扇区有 512 字节。因此,一张软盘片可存储的数据量:

当用 8 个扇区时:

$$8 \times 512 \times 2 \times 40 = 328 \text{ KB}$$

当用 9 个扇区时:

$$9 \times 512 \times 2 \times 40 = 368 \text{ KB}$$

这个容量称为“格式化容量”。

由此可见,格式化容量与采用的格式有关。如果我们把这样的软盘片放在 Apple 机上,它会得出另一种结果来。因此,当我们选择盘片时要注意到非格式化容量,格式化容量的差别。目前,由于互换的要求,大多数盘片采用 PC-DOS 的每磁道 9 个扇区的标准。格式化容量都写明是 360KB。这是 368KB 取整的结果。

对于双面倍密度 96tpi 的盘片,非格式化容量是 1000KB,格式化容量是 720KB。

对于双面高密度 96tpi 的盘片,非格式化容量是 1.6MB(兆字节)。格式化容量是 1.2MB。

软盘片的选择

既然有这么多种规格,在使用中如何选择呢?

第一,要采用和微机规格一致的盘片。如 Apple 机还是 IBM-PC 机。当然,双面盘当单面盘用是可以的,就是价钱贵一些。但单面盘一般不能用作双面盘。因为不用的那面缺陷多,可靠性差。

高密度盘片不能当倍密度用。因为磁特性不同,在倍密度的软盘机上写不进去。

盘片的规格在保护套上都有标明。如倍密度用 DD(Double Density),高密度用 HD(High Density),双面用 DS(Double Side),48tpi 或 96tpi 直接标明,只不过有的用大写字母 TPI 字样。另外,有的厂家不同盘片的标签颜色也不同,容易区别开来。

在购买盘片时,可以用肉眼观察盘片表面的情况。从保护套的长圆孔处可以看到盘表面的磁性层,应当平整,光滑,无小凸起物,也没有小针孔。表面应没有划伤痕迹,也应没有霉斑。颜色应均匀,没有深浅不一。

注意不要用手或其他物体触摸盘片表面。

在外观检查后,应把盘片放到 IBM-PC 机上进行格式化。新盘片格式化后应没有缺陷。

三种 $5\frac{1}{4}$ 英寸盘片规格汇总如下:

规格	位密度 bpi	道密度 tpi	非格式容量 KB	格式化容量 KB
双面倍密度 48tpi (DS,DD)	5876	48	500	360
双面、倍密度,96tpi (DS,DD)	5876	96	1000	720
双面、高密度,96tpi (DS,HD)	9646	96	1600	1200

软盘片使用注意事项

软盘片使用环境要求:

温度 $10^{\circ}\text{C} \sim 44^{\circ}\text{C}$ ($50^{\circ}\text{F} \sim 112^{\circ}\text{F}$)

相对湿度 $20\% \sim 80\%$ (不结露)

这些要求一般都能达到。也有个别情况会超出上述范围。例如北方的冬天,相对湿度可能低于 20% ,这时盘片因过于干燥,摩擦产生静电,使用时会出错。又例如在南方夏天,机箱内温度会高于 44°C ,盘片受热后变形,当然也会出错。

除了环境条件外,使用中还应注意:

1. 要保持盘片使用环境和保存环境清洁。以免污染盘片表面。盘片划伤报废多半是灰尘造成的。因此,微机使用的地方应保持清洁。盘片用后放回纸套内。

2. 不要把软盘片放在靠近有磁场的地方,如:电机、变压器,显示器等设备附近。过大的磁场会把记录的数据抹去。

3. 不要弯曲、折叠,重压盘片。因为盘片上有折痕或变形,都会使读写出错。

4. 严禁用手触摸盘片表面。也不能用酒精或其他溶剂清洗盘片表面。

5. 不要把盘片放在高温处或在烈日下曝晒。

6. 标签应用软水笔写好后贴上去,不要用铅笔或圆珠笔写。也不能用橡皮擦标签。因为粉末或油性物会落到盘片表面上,造成读写出错。

软盘片注意使用,一方面延长使用寿命。按标准,软盘片可使用三百万次。但由于使用不小心,使盘片过早报废,没有达到这个指标。另一方面,盘片损坏造成数据丢失,使辛辛苦苦编成的软件或工作结果,毁于一旦。这种情况,也是常见到的。

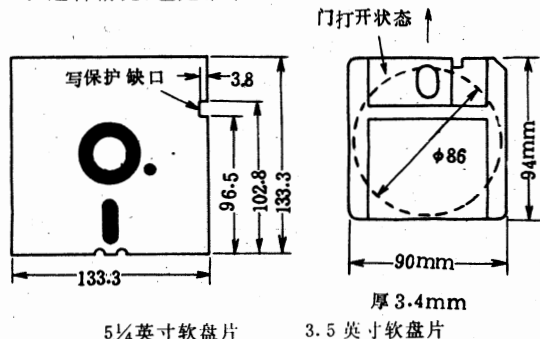


图 1



PC 用户

数据库通用维护模块设计

锦州铁路运输学校 徐维祥

绝大多数的数据库在实际运用中都要经常进行数据的更新,因而数据库的维护是一项经常性工作。目前不外乎采用两类维护方法:一是为每个数据库编制一套维护模块。对于不同的数据库要设计不同的维护模块,这种重复劳动浪费了程序设计人员很多时间;二是把维护工作留给通晓 DBASE III 命令的人员去完成。这样提高了对用户的要求,客观上限制了数据库的普及使用。

能否提供一个通用维护模块,既不要求操作者了解程序原理和 DBASE III 命令,又能完成对各种数据库的维护工作呢?笔者进行了尝试,本文把一个操作简便,功能齐全的通用维护程序奉献给读者。

一、通用维护模块应当具备的功能

维护数据库的工作可以归结为:1.删除一条或一批记录;2.修改一条或一批记录;3.替换一批记录;4.中间插入记录;5.末尾添加记录。有了这五种基本操作就可以满足对数据库更新的各种需求。

对单条记录的处理以采用记录号定位的方法最简便;处理一批记录则以条件定位的方法最灵活。

为了通用,维护模块应能自动向操作者提供要维护的数据库的结构信息。

为使操作尽可能简便,应对字段名编号,使操作者通过选号而不必输入汉字就能使用字段名。

维护模块应在记录范围、记录应满足的条件及摘录出必要的字段加以处理等方面向用户完全敞开。

二、主要技术问题的处理

首先要解决的问题是能提供任意选定的数据库的结构信息。对此采用的方法是,在第一工作区打开要维护的数据库,用拷贝命令把库的结构拷贝成结构描述库 JBJ.DBF,结构描述库的记录就是要维护的数据库各字段的有关信息。然后在第二工作区打开 JBJ.DBF 库,准备随时提供有关信息。

DBASE III 命令的一般形式为:〈命令字〉[范围][字段名清单][条件]。由于要针对未知结构的数据库进行设计,方括号中的三个可选项必须是通用的。只有充分运用这三个选择项,才能保证在任何情况下都能满足维护操作的要求。

在构成定位记录的条件时,要组成〈字段名〉〈关系符〉〈字段值〉的条件表达式。在需要使用字段名时,屏幕提示子模块采用循环方式将结构描述库中的字段序号和字段名一一显示出来,操作者一经指定字段序号,程序就用变量传递及宏代换替换成字段名,免除了输入汉字字段名时既麻烦又易错的弊端。

为了组成灵活的条件表达式,关系符可以采用多种形式。例如,对数值型和日期型字段可采用=,<,>,<=,>=,<>;对字符型字段还可采用\$。\$引进了包含功能,只要字段值中包含给定值,条件就成立,选择性极好。为了操作方便,对这些关系符也做了编号处理,选号即可输入关系符。

不同类型的字段构成表达式时有不同的要求。为了通用,在指定字段之后,程序要通过结构描述库了解该字段的类型,然后用相应的方法接受手工输入的字段值,组成合法的条件表达式。譬如,对数值型字段只接受数码;对日期型字段提示 MM/DD/YY,将数据转换成日期型等。这样防止了字段值类型出错,也避免了输入的内容超过允许的长度。

程序还能将简单的条件表达式复合起来,构成〈简单条件 1〉.AND./OR.〈简单条件 2〉.AND./OR.……的形式,使操作者对数据拥有极强的控制力,方便灵活地筛选出要处理的记录。

为了实现在某一范围内进行更新操作,设置了范围选择子模块。自动提示指定库内现有记录数,并接受处理范围的起止记录号,选择满足条件的记录进行处理。对于单条记录,给出相同的起止记录号就可以越过范围和条件选择进入实质性处理。

有时只需修改少数几个字段,为此使用屏幕提示子模块将字段序号和字段名全部显示出来,操作者只需指定序号就可以取出字段名。这段程序包含在循环中,循环一次取出一个字段名,并自动将其用逗号连接起来,再经宏代换处理,放在修改命令的[字段名清单]位置。于是,修改时只列出点到的字段,非常清晰。考虑到有时要对一批记录中的某一字段作同样的修改,程序设置了用输入值一次更改一批记录的替换功能。

实用系统中维护模块内使用的格式文件,通常是针对具体数据库设计的。在通用维护中设计通用格式文件比较复杂。为避开这个难点,程序巧妙地借用了 CHANGE 命令提供的格式,解决了这个问题。

* 通用维护模块(TYWH. PRG)

```
SET TALK OFF
ACCE "请输入数据库名:" TO KM
USE &KM
COPY TO JBJ.DBF STRU EXTE
SELE 2
USE JBJ
SELE 1
H=" "
DO WHILE H(">0")
CLEA
@1,28 SAY "数据库维护模块功能:"
@3,28 SAY "1 2 3 4 5 0"
@4,28 SAY "追插替修删退"
```



```

@5,28 SAY "加入换改除出"
@7,31 SAY "请选号(0-5):" GET H
READ
CLEA
STOR 0 TO B,C,D,E,J,T,Z
STOR " " TO G,MB,TJ
DO CASE
CASE H="0"
LOOP
CASE H="1"
APPE BLAN
CHAN NEXT 1
CASE H="2"
INPU "新记录插入位置:" TO B
GO B
INSE BLAN BEFO
CHAN NEXT 1
CASE H="3"
DO TFW
DO TTJ
SELE 2
DO WHIL E<10
@E+11,0 CLEA
E=E+1
ENDD
@12,5 SAY "要替换哪个字段:"
@12,21 GET B PICT "@Z 999" RANG 1,T
READ
GO B
MB=TRIM(FLELD-NAME)
@12,21 SAY "&MB"
SELE 1
APPE BLAN
MD=&MB
@13,13 SAY "替换成:" GET MD
READ
DELE
PACK
GO C
LOCA NEXT D FOR &TJ
DO WHIL RECN()<D+C-1
REPL &MB WITH MD
CONT
ENDD
CASE H="4"
DO TFW
IF D=2
GO C
CHAN NEXT 1
LOOP
ENDI
SELE 2
COUN TO T
DO TPM
DO WHIL E<=T

```

```

B=0
@18,5 SAY "指定要改的字段,回车退出:"
@18,32 GET B PICT "@Z 999" RANG 0,T
READ
IF B=0
EXIT
ENDI
GO B
MB=MB-G-FIELD-NAME
@12,5 SAY MB
G=","
ENDD
MB=TRIM(MB)
DO TTJ
GO C
LOCA NEXT D FOR &TJ
DO WHIL RECN()<D+C-1
CHAN FIEL &MB NEXT 1
SKIP-1
CONT
ENDD
CASE H="5"
DO TFW
IF D>2
DO TTJ
ELSE
TJ="RECN()=C"
ENDI
GO C
LOCA NEXT D FOR &TJ
DO WHIL RECN()<D+C-1
CLEA
DISP
ACCE "拟删除对否(Y/N)?" TO X
IF UPPE(X)="Y"
DELE
ENDI
CONT
ENDD
PACK
ENDC
ENDD
CLOS DATA
DELE FILE JBJ.DBF
SET TALK ON
RETU
* 屏幕提示子模块(TPM. PRG)
CLEA
STOR 0 TO I,M,N
GO TOP
DO WHIL .NOT. EOF()
M=INT(I/5)
N=16*(I-5*M)
@M+1,N SAY RECN() PICT "999"
@M+1,N+3 SAY " "+FIELD-NAME

```

I=I+1

SKIP

ENDD

RETU

* 条件选择子模块(TTJ. PRG)

SELE 2

A1="="

A2="<"

A3=">"

A4="<="

A5=">="

A6="<)"

A7="\$"

SET FILT TO FIELD-TYPE<)"M"

COUN TO T

DO TPM

@11,2 SAY "记录需满足的条件:"

DO WHIL J<6

STOR " " TO Q,K,Z,L

B=0

@18,0 CLEA

@18,8 SAY "指定字段名代号:"

@18,24 GET B PICT "@Z 999"RANG 1,T

READ

GO B

Q=Q-FIELD-NAME

@11+J,20 SAY Q

F=SPAC(FIELD-LEN)

B=0

Y=6

DO CASE

CASE FIELD-TYPE="C"

Z=" 7:\$"

Y=7

R="@R"

DO TFH

IF A&K="\$"

Q=["&F"]-A&K-Q

ELSE

Q=Q-A&K-["&F"]

ENDI

CASE FIELD-TYPE="N"

P="@Z #####"

DO TFH

Q=Q-A&K-[VAL("&F")]

CASE FIELD-TYPE="D"

P="@D"

DO TFH

Q=Q-A&K-[CTOD("&F")]

CASE FIELD-TYPE="L"

@18,8 SAY "请输入期望值(T/F):" GET L

READ

@ 11+J,30 SAY "=" + "&L"

IF L="F"

Q=" NOT." + Q

ENDI

ENDC

B=0

@18,6 SAY "选择{1: 与 2: 或 3: 修改 0: 结束}:"

@18,36 GET B PICT "@Z 9" RANG 0,3

READ

Q=TRIM(Q)

DO CASE

CASE B=1

TJ=TJ-Q- ". AND."

@12+J,15 SAY ". AND."

CASE B=2

TJ=TJ-Q- ". OR."

@12+J,15 SAY ". OR."

CASE B=3

LOOP

CASE B=0

TJ=TJ-Q

EXIT

ENDC

J=J+1

ENDD

SET FILT TO

SELE 1

RETU

* 关系符号子模块(TFH. PRG)

@18,0 CLEA

@18,0 SAY "{1:= 2:< 3:> 4:(= 5:)= 6:<}"

@18,27 SAY Z+"}选号"

@18,37 GET B PICT "@Z 9" RANG 1,Y

READ

K=STR(B,1)

@11+J,30 SAY A&K

@18,0 CLEA

@18,25 SAY "输入期望值:"

@11+J,32 GET F PICT "&P"

READ

F=TRIM(F)

RETU

* 范围选择子模块(TFW. PRG)

CLEA

COUN TO S

@11,10 SAY "共有"+STR(S,5)+"条记录"

@13,10 SAY "范围:从第"

@13,20 GET C PICT "@RZ 99999 条" RANG 1,S

READ

@13,28 SAY "至第"

@13,33 GET D PICT "@RZ 99999 条" RANG C,S

READ

D=D-C+2.

RETU

字典库的编辑和使用

江西拖拉机发动机厂 黄焕如

在 DBASE 应用软件中,经常遇到大量的汉字词组输入。为减少数据冗余,节约存储空间,提高输入速度,一个有效途径就是利用字典库。

所谓字典库就是把可能出现的频率较高的信息,分门别类地存入一系列数据库中,然后把原库或其他库中的这些信息以相应的代码替换。例如,在一个财务凭证传票数据库中,科目名称一般需要 15 个汉字左右(30 字节),如果每个记录都占有这 30 个字节,记录数增加时不仅浪费存储空间,而且输入费时又易出错。于是可以单独建立科目字典库 PZKM.DBF,结构和部分数据如下:

。display structure

字段名	类型	宽库	
DH	字符	6	(科目代号)
KM	字符	30	(科目名称)

Record #	DH	KM
1	101000	在途出口商品
2	102000	库存出口商品
3	103000	委托加工商品

在原凭证传票数据库中就不需要描述科目名称这个字段了,保留科目代号字段,利用它读取相应的科目名称。为提高查询速度,在建字典库的同时也建立相应的索引文件。索引的关键字为同主库的同一字段。如上例以关键字 DH 建立索引文件 PZKM.NDX。

字典库的内容应采取既可另行编辑,也可输入时现场编辑的方法。单独编辑字典库比较简单,使用时要注意输入字典内容不应重复或太全,以免一些根本用不到或不常用的内容进入字典库而降低查找速度。而现场编辑字典库变化比较多,应结合实际灵活使用。如上例,在一个输入财务凭证传票过程中,往往仅键入科目代号而调用科目字典库内相应的科目名称,由于科目数量太多且各单位常用科目各不相同,显然仅采用单独编辑科目字典库是不适宜的,应结合采用现场编辑、自动进入字典库的方法。以下程序可插入在输入部分程序中,其主要功能是:当输入任一科目代号时,首先在科目字典库内查找该代号,如有此代号则显示相应的科目名称;否则提示是否重建该科目名称,如重建将这新的科目设置在科目字典库中。由于其索引文件和数据库被同时打开,新建立的科目又可随时被调用,这就实现了现场编辑字典库的功能。(参考程序一)

字典库的使用很广泛,其中最重要的就是汉字词组的选择输入。即通过程序在屏幕上显示一定数量的汉字词组和对应的数值,输入某一数值,则相当于输入

了相应的汉字词组,大大地提高了输入汉字的速度和准确性。程序二给出了示范,该程序的特点是:字典库内容显示在屏幕右上角窗口内,每页显示五个词组(可根据需要修改),既可向前翻又可向后翻页。选择某一数值,其对应的词组就可调出,另外还可随时退出选择。当字典库内记录数不是其显示个数的整数倍,下翻最后一页时,按最后五条词组排列显示,充分利用了屏幕,且新颖别致。程序二可作为一子程序供输入、检索等模块调用。

附程序一

```
.....
SET TALK OFF
CLEAR
USE PZKM INDE PZKM
DH1=SPACE(6)
KM1=SPACE(30)
JJ=. T.
DO WHILE JJ
@2,8 SAY "科目代号:" GET DH1
READ
SEEK DH1
FF="N"
IF EOF()
@24,0
@24,14 SAY "字典库内无此代号,重建吗?(Y/N)"
GET FF PICT "!"
READ
@24,14
ELSE
JJ=. F.
LOOP
ENDIF EOF()
IF FF="Y"
@24,14 SAY "请输入科目名称:" GET KM1
READ
@24,0
APPE BLAN
REPL KM WITH KM1,DH WITH DH1
JJ=. F.
ENDIF FF="Y"
ENDDO JJ=. T.
@2,40 SAY "科目名称:"+KM
.....
```

附程序二

```
SET TALK OFF
USE PZKM
CLEAR
```

```

@24,0
KK=. T.
DO WHILE KK
N=1
A=" "
@2,40 CLEAR
@0,0
DO WHILE N<6
IF .NOT. EOF( )
@ROW( )+1,40 SAY STR(N,4)+"."+KM
SKIP
ENDIF
IF EOF( ). AND. N<5
N=0
SKIP-5
EXIT
ENDIF
N=N+1
ENDDO
IF N=0
LOOP
ENDIF
SKIP-1
M=RECNO( )
DO WHILE .T.
@24,10 say "请选择:1-5 或 U/D-上/下换页 Q-退出"
SET CONS OFF
WAIT TO A
SET CONS ON
@24,0
DO CASE
CASE VAL(A)<6 . AND. VAL(A)>0
GO M
SKIP VAL(A)-5
@14,0 SAY DH+" " +KM (取出选择的词组)
KK=. F.
EXIT
CASE UPPER(A)="U" (上翻一页)
GO M
SKIP-9
EXIT
CASE UPPER(A)="D" (下翻一页)
GO M
IF .NOT. EOF( )
SKIP
ENDIF
EXIT
CASE UPPER(A)="Q" (退出)
KK=. F.
EXIT
ENDCASE
ENDDO
ENDDO KK
RETURN

```

用 DOS 命令设计菜单

陈凯

灵活地使用 DOS 的批文件命令、设置提示命令 PROMPT 和显示命令 TYPE, 可以设计出用户界面友好的菜单。兹介绍一种设计菜单的方法和技巧。

一、建立标准的 ASCII 码菜单选择项文件。例如, 使用 WORDSTAR、EDLIN 等建立《通用软件设计工具包》菜单选择项文件, 其名称是 GJ.ACD, 内容为:

【通用软件设计工具包】

G1—制表准备	G5—字母转换
G2—打印输出	G6—软件保护
G3—居中输出	G7—拷贝扩充
G4—格式转换	G8—通用编辑

二、用批文件命令(扩展名 BAT)、设置提示符命令和显示命令将菜单选择项与操作提示信息相连接形成一个完整的菜单。批文件命令的文件名是“GJ.BAT”, 设置 DOS 系统的提示信息为“选择 G0(G0—退出)~8 后, 按 Enter 键:”

ECHO OFF

REM 文件名是: GJ. BAT

PROMPT 选择 G0(G0—退出)~8 后, 按 Enter 键:

TYPE GJ. ACD

三、分别建立各功能模块的批文件命令。制表准备子模块批文件命令的文件名是: G1. BAT”, 退出《通用软件设计工具包》子模块批文件的文件名是“G0. BAT”, 其它功能模块的文件名的定义类推。

ECHO OFF

REM 文件名是: G1. BAT

DBASE G1. PRG

TYPE GJ. ACD

ECHO OFF

REM 文件名是: G0. BAT

PROMPT: \$p \$g

CLS

四、批文件命令 GJ. BAT 和 G0. BAT 的运行结果:

C:\>GJ

键入回车后即显示上面的菜单, 并在其下显示:

选择 G0(G0—退出)~8 后, 按 Enter 键:

由此可知, 这种菜单主要作用是显示操作提示的信息, 对于一切可以执行文件(扩展名为“.BAT”、“.COM”和“.EXE”)功能的实现没有任何影响。

在 IBM PC/AT 型机上配置虚盘

广东 汕头 金山中学 何管略

DOS 直接使用 IBM PC/AT 型机的内存主区,占 RAM 640KB。运行汉字 dBASE III 时,如果采用 CCDOS 2.10,其内存分配大致如下(误差不超过±10KB)

MS-DOS	50KB
CC-DOS	70KB
HZK16	260KB
DBASE—III	260KB

它将 260KB 的两级字库,全部装入内存,运行速度确实不慢;但是,由于内存空间用完,没有一点自由空间,以致 dBASE III 不能调用 DOS 的外部命令(如 Edlin),使用起来,甚为不便。

CCDOS 4.0 对此作了改进。要装入多少汉字,可由用户指定。通常是装入一半,这样就剩余 120KB 左右的自由空间,可供 dBASE III 调用 DOS 的外部命令。但是,对于未装入内存中的汉字的 I/O 操作,每次都要读盘,降低了运行速度,难免美中不足。

有没有既活且快、两全其美的方案呢?有的,这就是下面我们所要介绍的“配置虚盘方案”。

IBM PC/AT 型机一般配有总量达 1MB 的 RAM。其中,主区 640KB,扩展区 384KB。可以开发这 384KB 的扩展内存区,建立虚盘,将 260KB 的全部汉字库,装在虚盘里。于是,对于任何汉字的各种 I/O 操作,都不必去读硬盘或软盘;只读虚盘,可以明显提高运行速度。另外,在 640KB 的主内存区中,又可剩余 250KB 左右的自由空间,供 dBASE III 调用 DOS 的各种外部命令;例如长达 170KB 的 PCTOOL 工具软件,也可由 dBASE III 的 RUN 命令随时调用,非常灵活方便。怎样建立虚盘?以 CCDOS 2.13E 操作系统为例:

(1)在系统配置文件 Config.SYS 中,增写一项设备驱动程序:Device=Vdisk.SYS 384 512 32/E。当然,建立虚盘的系统文件 Vdisk.SYS,必须存在根目录中。参数 384 规定虚盘的大小(KB),512 规定扇区的字节数(B),参数 32 规定根目录的项数,开关/E 规定虚盘建在扩展内存区(否则虚盘建在主区)。

DOS 对设备驱动程序的项数没有限制,所以在 Config.SYS 文件中,原有的 Device=Ansi.SYS 可以保持不动,它的加强键盘和屏幕管理的功能,也不受影响。

(2)运行 CCDOS 2.13E 系统,选用将 16 点阵汉字库全部装入虚盘(其盘符规定为 D:)的相应项目。

(3)考虑到虚盘的 384KB 容量,装了 260KB 的字库之外,还有 120KB 的空间,可以用来装载一些常用的命令文件;如果键入 Path D:\,又可提高常用命令文件的运行速度。此项工作,重复性大,宜写成批处理文件,每次建盘后运行一遍。

(4)在虚盘的建立、装载和调试过程中,有时难免丢失 Setup 参数,导致下次开机失灵。建议首先运行 Setup 程序,记录本机的各项参数,特别是硬盘的型号。在关机之前,再次运行 Setup 程序,检查有关参数;如果发现丢失,就及时键入正确参数。

配置虚盘的具体情况,项目如下:

VDISK Version 3.30 virtual disk D:

Transfer size adjusted

Buffer size: 384 KB

Sector size: 512

Directory entries: 32

Transfer size: 8

Strike a key when ready.....

C:\>

C:\>dir *.SYS

Volume in drive C is JIN—ZHONG

Directory of C:\

CONFIG	SYS	86	4-19-90	10:08p
ANSI	SYS	1664	7-11-86	9:07a
VDISK	SYS	3455	3-17-87	12:00p

3 File(s) 21495808 bytes free

C:\>type config.sys

Device=Vdisk.sys 384 512 32/E

Device=Ansi.sys

Files=20

Buffers=24

C:\>dir d:

Volume in drive D is VDISK V3.3

Directory of D:\

HZK16		261696	3-19-88	8:59a
EDLIN	COM	7526	3-17-87	12:00p
BASICA	COM	36403	3-17-87	12:00p
SETUP	EXE	15793	1-07-88	5:29p
MODE	OCM	15440	7-24-87	12:00a
CC11	COM	3991	2-11-89	8:53a
CE21	COM	3991	2-11-89	4:38p
CE25	COM	3991	2-14-89	1:25p
CS	COM	98	8-21-87	1:36p

9 File(s) 38912 bytes free

C:\>CS

内存自由空间 517424 字节

.Run CS

内存自由空间 251312 字节

IBM PC/XT 机异步通信适配器的自检方法

中国金融学院 杨沂

中科院地理所 杨耘华

微机的异步通信方式,也就是串行通信方式。通信的发送方和接受方之间的数据信息的传输是在单根数据线上,以每次一个二进制位传送。但是在联机的过程中,一旦数据未能传输成功,就需对每个微机的异步通信适配器(UART)进行检查。一般是采用物理接线的方法,使计算机发出的信号,变成自身接收的信号,以此来检验 UART 的工作情况。除上述方法,还可用软件检验的方法而无需物理接线。

IBM PC/XT 系列微机使用的 UART 是 8250 器件,采用电压和电流环接口,电压接口采用 EIA RS232C 串行通信标准,用 $-3\sim-15V$ 之间的电压表示二进制 1, $+3\sim+15V$ 之间表示二进制 0,允许最大数据传输率为 20kbps,最长可驱动电缆 15 米。8250 有两个信道,即 COM1 和 COM2。我们只对 COM1 进行讨论,对 COM2 只需将端口改一下即可。

在编制通信软件时,对芯片的通信速率、奇偶校验和字长等初始化之后,就可采取程序查询方式或中断请求方式实现通信。查询式通信比较直观,原理也很简单,只要通过对 8250 通信线路状态寄存器进行访问,由其内容来判断能否向 8250 输出数据或由 8250 输入数据。它的主要缺点是要占用大量的机时去查询 8250 的状态,既不经济,效率也低。

另一种方式就是通过硬件中断请求来为输入或输出建立一个循环队列,以完成串行通信的工作。8250 一旦产生了符合用户规定的条件,就通知 CPU。在 CPU 允许后,就将现在运行的程序挂起,先运行中断服务程序,然后再运行原来的程序。这种方式可大量节约机时,提高计算机使用效率。

使用 8250 中断方式进行通信的编程,除了通常的初始化,还需做以下几步:

1. 对 8259A 中断控制器的中断屏蔽寄存器初始化,即允许第 4 级中断。
2. 将 8250 中断服务程序的入口地址填入中断向量的向量单元中。
3. 初始化 8250 芯片,包括置成中断工作方式。
4. 根据需要,设置中断允许寄存器相应的允许/屏蔽位。8250 允许四级中断:(一)数据接受就绪,(二)发送保持寄存器空,(三)允许接受线路中断(四)允许 MODEM 状态中断,其优先级由(一)至(四)。

IBM PC/XT 系列微机的 UART 有一种特殊的功能,即在调制解调器寄存器初始化时将第 4 位置成 1,这样就使 8250 发出的数据变成了自己接收的数据,而无需外部接线。我们利用这种特殊功能和通信硬件中断的特性,编制了一个小程序,来完成对 UART 的自我检查。程序的主要特点是:

1. 无接线情况下,利用 UART 的循环特性;
2. 使用了 8250 数据接收就绪中断。

程序的主要流程如下:选择是 COM1 口还是 COM2 口;取出原来中断对应的段地址和偏移值,加以保存;然后对 8250 进行初始化;接着利用 MS-DOS 的 INT 21H 的 25H 号中断,将我们所编的中断服务程序入口通知 CPU;并接过异步通信中断向量,写入 8259A 的中断屏蔽寄存器;在调制解调器寄存器初始化时,将第四位置成 1,以建立 UART 的循环方式;并在屏幕上建立两个窗口;从而打开中断,程序进入一个循环。它自动查看是否有信息从键盘输入,有则在键盘输入窗口显示出来,并由 8250 发出,接着再看串口输入缓冲区是否收到新的码,有则在串口输入窗口显示出来。这两个窗口是分别滚动的,一旦按了 ESC 键,则程序恢复串行口中断向量、8259A 中断屏蔽器的原来状态,从而关中断并返回 DOS。通过两个窗口的数据可判断 UART 的工作情况,如果两个窗口的数据能一一对应,就证明 8250 工作正常,否则需进一步诊断。

由于篇幅关系,我们只给出对 8250 的 COM1 方式中断器的建立过程及中断服务程序,如对此有兴趣者,可与编辑部联系。

.....

```
mov    ah,35h                ;取原中断段址偏移址
mov    al,0ch
int     21h
mov     int-seg,es
mov     int-offs,bx
mov     dx,offset com-int    ;设新中断
mov     ah,25h
mov     al,0ch
int     21h
in      al,21h                ;在 8259A 中打开 COM1
and     al,not(10h)           ;中断
out     21h,al
mov     dx,03fch              ;在 8250 中置循环特性
mov     al,11011b             ;和收码中断
out     dx,al
mov     dx,03f9h
mov     al,1
out     dx,al
```

.....

```
com-int proc far
        sti                    ;保存现场
```



```

push ax
push bx
push dx
push ds
mov ax,cs          ;建立可寻址性
mov ds,ax
mov dx,03f8h       ;取数据
in al,dx
cli
mov bx,con-in
mov [com-buf+bx],al ;将数据放入缓冲区
inc bx              ;确定下一数据在缓冲区
cmp bx,com-buf-len ;的位置
jne com-int1
xor bx,bx
com-int1:
mov com-in,bx
sti
mov al,20h          ;中断结束
out 020h,al
pop ds              ;恢复现场
pop dx
pop bx
pop ax
iret                ;中断返回
com-int endp

.....

int-offs dw 0       ;保存原中断地址
int-seg dw 0
com-in dw 0          ;数据位置
com-buf-len equ 100 ;缓冲区长度
com-buf db 100 dup('0') ;缓冲区

.....

```

产生真正的随机数

上海机械学院 应海涛

程序设计中常用到随机数。虽然 BASIC 语言中有 RND 函数可用来生成随机数,但每一次运行程序,它总是生成同一系列的“随机数”,往往不能达到某些应用目的。本人使用了一点小技巧解决了这个问题,只要在程序前部加上如下语句:

```

10 PRINT "Press any key"
20 IF INKEY$="" THEN A=RND;GOTO 20

```

运行程序时,只要按下任意键,即可生成难以预测的某一系列随机数供使用。20 句利用了运行程序到按一任意键之间的时差(足以让第 20 句循环无数遍)的不确定性,使随机系列的第一个数无法确定。这样就达到每次运行程序,产生的随机数系列难以雷同。

利用屏幕缓冲区实现语言的三重调用

淄博市建设银行 张建新

PC 机分配给屏幕缓冲区的内存为 16K。在段地址 47104(&HB800)中,偏移地址为 0 至 16383。屏幕的分辨率为 640×200。缓冲区每一个字节的一位对应屏幕上的一个点。因此,一显示行的 640 点需要 640/8=80 字节。屏幕显示分偶数行和奇数行,其缓冲区各分配 8K 字节,各为 100 显示行。但偶数行和奇数行实际需要的内存量各为 100×80=8000 字节,比 8K(8×1024)各少 192 字节。这样,分配给屏幕缓冲区的内存比实际使用的要多出 384 字节。偶数行分配地址为 0 至 8191。其中 8000 至 8191 为空余区。奇数行分配地址为 8192 至 16383。其中 16192 至 16383 为空余区。

在混合语言程序设计中,常会遇到数据传递问题。数据传递速度最快、手段最简捷的方法便是内存中转法,即把参与传递的数据置入一足够安全的内存区,进入另一语言系统时,再从此内存区读取。综上所述,在传递数据较少,不影响屏幕美观的情况下,可以利用屏幕缓冲区空余的 384 字节实现数据的中转,也可以存放简短的机器语言子程序。

以下二程序展示利用屏幕缓冲区从 dBASE 调用 BASIC 程序,BASIC 程序又调用机器语言子程序,最后返回 dBASE,同时进行数据相互传递诸功能。

```

10 DEF SEG=&HB800:P=PEEK(8000)
110 CLS,KEY OFF,SCREEN 1,0,COLOR 1,0
120 LOCATE 1,12
130 LINE(10,16)-(289,36),2,B
140 LINE(11,15)-(290,37),2,B
150 PRINT "从 DBASE 中取 P 值"
185 LOCATE 5,12:PRINT P
187 LOCATE 7,12:INPUT "输入 1 数":A
190 SCREEN 2,0
195 PRINT "调机器语言子程序,做屏幕硬拷贝!"
220 POKE 8000,A
230 PRINT "把 A 的值送入 DBASE"
320 DATA &H55,&Hcd,&H05,&H5D,&HCB
335 DEF SEG=&HB800;FOR I=8003 TO 8007
340 READ J;POKE I,J;NEXT I
350 X=8003;CALL X
400 SYSTEM

```

```

C)type qwe. prg
clear
set talk off
SET SEGMENT TO 47104
input "输入1数:" TO A
I=0
STOR "POKE"+STR(I+8000)+"", "+STR(A) TO L
&L
@3,12 say "A 的值送入 BASIC 程序 AAA"
run basica AAA
set segment to 47104
@4,12 SAY "从 BASIC 程序 AAA 中取数"
k=peek(8000)
? k
return

```

谈谈子目录加密与解密的一种方法

黑龙江鸡西市公共汽车公司 李连德

为了使子目录中的程序不被非法用户进入,下面介绍一种用半个汉字对目录名加密的方法:

1. 用 CCDOS 启动,使机器进入汉字操作。
2. 打入 MD 密。
3. 用退格键将“密”字删去一半,然后回车。

此时再执行 DIR 命令,则只能看到<DIR>而看不到子目录名。这就使半个汉字成为了“保密字”,从而达到保密目的。

如果遗忘了保密字,子目录就无法进入。

解密方法如下:

1. 用 PC DOS 启动,使其进入西文操作系统。
2. 用 DIR 或显示磁盘目录命令:

C>DIR *. <CR>

```

  1  (DIR)89-09-27 8.25

```

```

1File(1) 19555529 bytes free

```

此时就能看到“密”字的半个汉字的字符“𠂇”,然后在 ASCII 码表中查出“𠂇”所对应的值 211。

3. 启动 CCDOS 进入汉字系统,并进入 BASIC。

4. 用 PRINT CHR\$(211)+CHR\$(A);

A 是 161 到 254 之间的任意 ASCII 码值。

假设 A 为 178 则:

```
PRINTCHR$(211)+CHR$(178)<CR>
```

硬

OK

5. SYSTEM<CR>返回 CCDOS

6. C>CD 硬,然后用退格键删去一半<CR>回车。

此时便进入到保密字为“密”的子目录中。

一个汉字是由 161 到 254 之间的两个 ASCII 码

组成的,当一个汉字被删半后就变成了单 ASCII 码,只有在 PC DOS 才能显示半字的 ACSII 符。可见一个半字所建立的子目录可用多个汉字的半字进入。利用汉字的这种组字规律,可将所有汉字库打印出来。

PC 机显示氢原子轨道波函数

陈可中 曾庆涛 郭冰莹

我们在 IBM-PC 机上用编译 BASIC 成功地模拟了氢原子中电子轨道波函数的图象,一则可以帮助同学们澄清一些物理概念和模糊的图象;二则可以激发同学们学习物理理论和电脑的兴趣。现将氢原子中 3S 电子轨道波函数立体图及程序附后:

```

10 SCREEN 2;CLS;PI=3.14159;DIM YH(2800),YL(2800)

```

```

20 UM=1000;VM=1000;A=-58*PI/180;A1=COS(A);A2=SIN(A)

```

```

30 DU1=40;DV1=40;DU2=10;DV2=10;UP=5500
VP=0;WP=1600;US=600

```

```

40 K=.015;W0=6001000!*3.14159^(-.5)

```

```

50 DEF FNF(U,V)=W0/1.732/81*K^1.5*(27-18*K*(U^2+V^2)^.5+2*K^2*(U^2+V^2))*EXP(-K*(U^2+V^2)^.5/3)-400

```

```

60 GOSUB 170;FOR U=UM TO -UM STEP -DU1

```

```

70 FOR V=VM TO -VM STEP -DV2;GOSUB 150

```

```

80 IF V=VM THEN PLOT=0 ELSE PLOT=2

```

```

90 GOSUB 180;GOSUB 330;NEXT V;NEXT U

```

```

100 GOSUB 170;FOR V=VM TO -VM STEP -DV1

```

```

110 FOR U=UM TO -UM STEP -DU2;GOSUB 150

```

```

120 IF U=UM THEN PLOT=0 ELSE PLOT=2

```

```

130 GOSUB 180;GOSUB 330;NEXT U;NEXT V

```

```

140 K$=INPUT$(1);END

```

```

150 UU=U*A1-V*A2;VV=V*A1+U*A2;TRAN=(UP-US)/(UP-UU)

```

```

160 W=FNF(U,V);X=VP-TRAN*(VP-VV);Y=WP-TRAN*(WP-W);RETURN

```

```

170 FOR I=0 TO 2800;YH(I)=-1000;YL(I)=1000;NEXT I;RETURN

```

```

180 XNOW=1300+CINT(X);IF PLOT=0 THEN XLAST=XNOW

```

```

190 IF Y>YH(XNOW)THEN GOSUB 230

```

```

200 IF Y<YL(XNOW)THEN GOSUB 280

```

```

210 IF Y<YH(XNOW)AND Y>YL(XNOW)THEN XLAST=XNOW;PLOT=0

```

```

220 RETURN

```

```

230 IF XNOW>XLAST THEN XSMA=XLAST;XBIG=XNOW

```

```

240 IF XNOW<XLAST THEN XBIG=XLAST;XSMA=XNOW

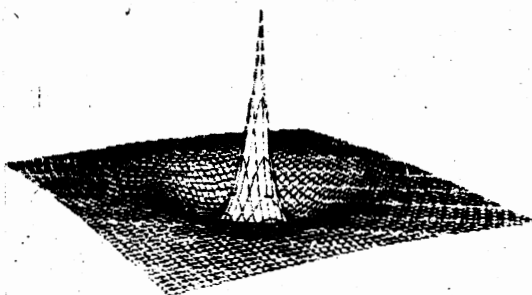
```



```

250 YH(XNOW)=Y;FOR SCAN=XSMA+1 TO XBIG-1
260 YH(SCAN)=Y+(YH(XLAST)-Y)*(SCAN-
XNOW)/(XLAST-XNOW)
270 NEXT SCAN;XLAST=XNOW;RETURN
280 IF XNOW>XLAST THEN XSMA=XLAST;XBIG=
XNOW
290 IF XNOW<XLAST THEN XBIG=XLAST;XSMA=
XNOW
300 YL(XNOW)=Y;FOR SCAN=XSMA+1 TO XBIG-1
310 YL(SCAN)=Y+(YL(XLAST)-Y)*(SCAN-
XNOW)/(XLAST-XNOW)
320 NEXT SCAN;XLAST=XNOW;RETURN
330 IF PLOT=2 THEN LINE-(.24*X+320,100-Y/
10);RETURN
340 PSET(.24*X+320,100-Y/10);POINT(.24*X+
320,100-Y/10);RETURN

```



BASIC 程序 P 加密 的简易解密法

潍坊市产品质量监督检验所 李志刚

BASIC 程序可以加 P 参数存盘,这类程序被调入内存后将不能用 LIST 列出,从而达到加密的目的。以前本杂志上介绍过不少解密方法,但大都较为复杂。现向读者介绍一种十分简单的解密方法。

先用 DEBUG 建立一个名为 PJ.BAS 的文件:

```

A>DEBUG
-E DS,0100 FF01
-N PJ.BAS
-R CX
CX 0000
: 0002
-W
-Q

```

需要解密时,在 BASIC 状态下用 LOAD “文件名”

调入要解密的程序,之后用 LOAD “PJ”调入 PJ.BAS,用 LIST 就可列出程序的内容。

我们还可以利用 PJ.BAS 恢复被 NEW 清除的程序,方法是:当执行 NEW 后又想恢复时,就可执行 LOAD “PJ”这样的操作,被清除了的程序便又恢复。

(接第 27 页)

• \$<(全文件名宏) 展开成含扩展名的目标名,例如

```

target.obj;target.c
copy $< A:\usrobj
tcc -c $*

```

可展开成:target.obj;target.c

```

copy target.obj A:\usrobj
tcc -c target

```

• \$:(文件路径名宏) 可展开成如下路径名:当文件名为 A:a\b\c\c.c 时,可展开成 A:a\b\.

• \$. (带扩展名的文件名宏) 可展开成带扩展名的文件名。如用 \$. 可把 a\b\c.c 展开成 c.c。

• \$&(文件名宏) 该宏可展开成不带扩展名的文件名。例如用 \$& 可把 A:a\b\c.c 展开成 C。

4. Touch

用于修改一个或多个文件的时间和日期为当前时间和日期,使比依赖于他们的文件更新。用法如下:

Touch 文件名表

文件名可用符 * 或?。对于执行过 Touch 的文件名,执行 MAKE 时,是要重新构造的。

5. Grep

Grep 是功能很强的查找实用程序,用法如下:

Grep 任选项 查找对象 指定文件

Grep 任选项如下表所示。

-c	只输出匹配行数	-o	UNIX 输出格式
-d	查找目录	-r	正则表达式查找
-i	忽略大小写区别	-u	更新选择项
-l	列出匹配文件	-v	打印不匹配的行
-n	行前有标号	-w	字查找
		-z	输出查找的文件名

[练习 3.5]

C>dir *.c

```

LI4 C 102 1-01-80 12:03a
LI5 C 46 1-01-80 12:29a
A C 38 1-01-80 12:35a
3 File(s) 13687072 bytes free

```

C>grep printf *.c

File LI4.C:

```

printf("Tianjin Shida Computer");
printf("Turbo C 2.0 exercise");

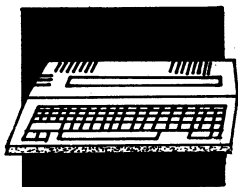
```

File A.C:

```

printf("Now is a.c");

```



学习机之友

BASIC 语言实现递归

福建宁德师专 苏亚华

一个子程序中,如果包含对自己的调用,则称为递归调用。一般的结构化程序设计语言中都支持递归。递归是解决许多带有回溯性质的算法问题的有效方法。递归的应用还使得程序具有良好的结构和可读性。然而,BASIC 语言是一种非结构化语言,一般都规定子程序不能调用自己,其原因有二:

1. BASIC 语言中没有局部变量的概念,程序中所有变量均为全局变量。同时,子程序的调用不带参数,子程序内部的变量与外部变量只要同名,就是同一变量,不产生屏蔽。如果允许递归,则必然导致许多同名变量出现在不同的各层次子程序中,造成子程序内部变量的混乱。

2. 子程序的嵌套深度有限制,如 APPLE 机 BASIC 语言规定:子程序嵌套深度不得超过 24。递归调用如控制不当,极易造成嵌套深度的越界。

囿于此,在 BASIC 语言的编程中,带有递归性质的问题以往总是靠大量 GOTO 语句来解决,从而使程序结构性、可读性都很差。那么,能不能将结构化编程思想引入 BASIC 语言,实现递归调用呢?事实上,只要我们了解机器对于子程序嵌套的处理过程,对上述两点作适当控制,妥善解决好子程序内部的变量保护和恢复等问题,递归调用是可能的。

为解决 BASIC 语言无局部变量的问题,在递归时,可用数组来区分不同层次中的同名变量。数组的下标表示子程序嵌套的深度,在每层调用之前,先设置好深度值,返回后再恢复。这样,在不同层次子程序中,变量各自独立,互相屏蔽。

我们用一道例题来具体分析如何在 BASIC 语言编程中实现递归。此题是一九八八年福建省青少年计算机程序设计竞赛 BASIC 语言组试卷的最后一题。

[试题] 试编程找出下列字母方阵(图一)中所有隐藏的单词 FUJIAN,单词中字母的走向可以是图二中编号为 1—8 的八个方向,要求打印格式为:

(X,Y),a1,a2,a3,a4,a5

(X,Y)为首字母“F”所在的行、列坐标,a1,a2,a3,a4,a5 是后续每个字母相对前一个字母的走向代号。例如,图一中箭号所示的 FUJIAN 是其中的一种,其打印结果应为

(1,1),3,5,4,6,8

[分析] 题目要求找出所有满足条件的解。因此,必须对所有可能的起点和走向进行搜索,判断是否符合条件。具体过程是:对每一个起点,先选择某一方向前进一步,如果这一步可以接受,就找出下一个字符开始匹配的所有解与其结合,然后返回去再搜索另一方向,直

		列坐标					
		1	2	3	4	5	6
行坐标	1	F	U	J	U	A	N
	2	J	J	F	I	U	N
	3	N	U	I	J	U	F
	4	U	A	N	J	N	A
	5	I	A	J	F	U	I
	6	J	N	N	I	N	F

(图一)

方向代号表		
8	1	2
7	3	3
6	5	4

(图二)

至八个方向走完。“找出下一个字符开始匹配的所有解”这一过程可以用一个子程序来实现,它是这样描述的:对当前搜索到的点,选择每个走向前进一步,如果这一步可接受,再“找出下一个字符开始匹配的所有解”与其结合,否则换另一走向。这就体现了递归的思想。

设计子程序时,我们用变量 K 表示搜索深度,用数组 T(N)来存放走向代号。每次调用子程序时变量 K 加 1,而把该层所选择的走向代号放入 T(K)中。待调用返回时,再将变量 K 减 1。这样,就能有效地保护有关数据(相当于被屏蔽)。程序如下:

```

10 F$="FUJIAN":N=LEN(F$):DIM H$(6,6),A
(8),B(8),X(N),Y(N),T(N)
20 FOR I=1 TO 6:FOR J=1 TO 6:READ H$(I,J):
NEXT I,J
30 FOR I=1 TO 8:READ A(I),B(I):NEXT I
50 FOR I=1 TO 6:FOR J=1 TO 6:X(1)=I:Y(1)=J
60 IF H$(X(1),Y(1))=MID$(F$,1,1)THEN K=2:
T(2)=0:GOSUB 500
70 NEXT J,I:END
101 DATA "F","U","J","U","A","N":DATA "J","J",
"F","I","U","N"
102 DATA "N","U","I","J","U","F":DATA "U",
"A","N","J","N","A"
103 DATA "I","A","J","F","U","I":DATA "J","N",
"N","I","N","F"
104 DATA -1,0,-1,1,0,1,1,1,1,0,1,-1,0,-1,-1,-1
500 REM 子程序入口
510 T(K)=T(K)+1:IF T(K)>8 THEN 590
520 X(K)=X(K-1)+A(T(K)):Y(K)=Y(K-1)+B
(T(K))
530 IF X(K)<1 OR X(K)>6 OR Y(K)<1 OR Y(K)>6
THEN 510
540 IF H$(X(K),Y(K))=MID$(F$,K,1)THEN 510
550 IF K=N THEN K=K+1:T(K)=0:GOSUB 500:K=
K-1:GOTO 510

```

```

560 PRINT "(",X(1);",",Y(1);")";:FOR I=2 TO N:
PRINT "(",T(I);",NEXT
570 PRINT:GOTO 510
590 RETURN

```

60 语句判断首字符的选择。若正确则转子程序执行 500 语句;否则选择下一个首字符位置。由于子程序采用递归算法,主程序的思想十分简单。

子程序的 510 语句进入下一个走向,八个走向都走完返回。520—540 语句判断该走向是否可接受。

如不可接受则转 510 语句选择下一个走向。在可以接受的情况下,550 语句判断是否到达终点;如到达,则打印出一个解;否则再调用子程序(递归),找出下一个字符开始匹配的所有解。

【结论】在非结构化语言 BASIC 编程中,一定程度地体现程序的层次关系和模块结构是可能的。程序的模块化,使得设计思想易于理解,程序可读性强,调试和修改也比较容易。

数据处理原理和技巧

南京大学 朱国江

本文介绍数据处理的四种基本操作:数据插入、删除、修改和合并。读者可从中体会菜单技术的应用;模块化程序结构的基本思想;在程序中加注中文说明的方法以及信息处理的设计原理。

一、数据处理原理

1. 数据插入

插入数据的方法比较简单。例如,有下列数组:

33 44 55 61 29

若要在 55 和 61 之间插入 99 这一个数,首先应设法在 55 和 61 之间空出一个空位来,为此,必须把 55 后面的两个数相应地后移一个存储单元。

这种移动必须是最后面一个数据先向右移,然后倒数第二个数据再向右移,这样就空出 55 以后的位置,并保留了 55 以后的其它数据。反之,如果 55 以后的第一个数据先向右移动一个存储单元,则最后一个数据(本例是 29)将被冲掉。

用 BASIC 语句来处理,上述思想简单归结为: $D(I+1)=D(I)$ 。这个语句的作用是将第 I 个存储单元之值,移动到第 $I+1$ 个存储空间去。

为保存数据并依次向右移,用循环语句来控制将是十分方便的。由于要插入新的数据,原有数组开辟的内存就不够用了。故 DIM 说明语句可用 DIMD(20)来解决(设原有 10 个数,增加到最多 20 个数)。

插入数据的实现,见程序中 140—360 句程序段。

2. 数据删除

删除数据的概念是不难理解的。如删除下列数据中的一个记录 66

98 97 66 53 42

首先将 66 删去,然后让 53 填入 66 的位置,最后将 42 再填补到原来 53 的地方。

上述思想是将后面的数据向前移,从而删除了原有的信息。但是,数据删除必须考虑:

· 除要删除的数据外,所有原来在被删除信息后面的全体数据,应该依次向前顺移;

· 删除一个数据后,原数组的最后一个记录也应删去。

总之,删除数据(或信息)的思路,可以概括为一个语句:即 $D(I)=D(I+1)$ 。为了实现上述思想,合理安排循环控制是至关重要的。

删除数据的原理,见程序段 370—550 句。

3. 数据修改

数据修改的问题相对于信息的插入、删除要简单得多。而利用下标变量或者字符串变量更为方便。例如,要更改字符串,只要设法指出存储在那一个单元的信息要改,然后重新赋给修改的内容即可。更简单地说是这样的:若 D(5)$ 中存有 PPI,今要改成 RHI,只要键入 $N=5$ 和 N=RHI$,那么, D(N)=N$$ 即可完成信息更改。

数字修改更为方便,可令 $D(N)=M$,其中 N 指修改的位置,而 M 为修改的数据。

完成修改数据的程序段,见 560—740 句。

4. 数据合并

数据合并是指将若干组数据合并在一起。为叙述方便,暂定一组数为 8 个,另一组为 10 个,各自已按数值由大到小顺序排好。现在要求把这两组数合并在一起,并按从大到小的顺序重新存储。

用数组 A(9)、B(11)作为原来两组数的存储单元,每组数组的最后一个存储单元作为附加单元备用。用数组 X(19)作为合并后全部数据记录的存储单元。

设置三个指针: I、J、N,让它们分别指向数组 A、B、X 的位置。开始时令 $I=1, J=1, N=0$,这就表示两组数据尚未合并时, I 指向 A 数组首地址的数据 A(1), J 指向 B 数组的首地址数据 B(1), 而 $N=0$,则表示指向 X 数组的地址下标为零的位置,此时 $X(0)=0$,表示没有数据存入。

程序设计的主要思路:

开始,让每个数组中各自最大的一个值进行比较,那一个值大,就把最大值存入 X 数组,并把对应数组

的指针加1。例如 A(I) < B(J), 则大数 A(I) 的值存入 X(N) 中, 并使指针 I=I+1; 若 A(I) > B(J), 大数是 B(J) 里的值, 应让 B(J) 的值存入 X(N) 中, 同时使指针 J 加1。如此不断循环, 直至实现两组数据的合并, 而合并后的数据也是顺序排列的。为判断合并是否完成, 可以安排 N 指针作为计数器, 本例中 N=18 时, 表示合并完成。

数据合并的程序段安排在 750—910 句。

二、几点技巧说明

1. DIM 语句安排

各功能块所用数组说明, 全部放在 20 句, 这样可以防止多次运行功能块时, 出现数组重复说明的错误。

2. RESTORE 设置

30 句中设置了恢复数据语句 RESTORE, 这是整个程序连接中必不可少的, 否则各模块完成该模块功能后, 返回总控时, 其它各模块无数可读。

3. HOME 语句作用

130 句用了清屏语句 HOME, 目的是将屏幕清除干净, 否则菜单提示和运行结果混在一起。但在打印程序运行结果时, 应删去 HOME 语句, 改 130 句为 PRINT; PRINT; GOTO 30; 否则发生打印格式不统一, 以及打印机空走等情况。中华学习机就属此例。

4. 循环延时技巧

730 句和 900 句各安排一个空循环, 起延时作用, 否则由于迅速返回总控显示菜单而看不清运行结果, 这是 130 句 HOME 语句造成的。

5. 虚读技术

760 句安排了一个虚读技术, 这是因为 750—910 句是处理两组数据的合并问题, 两组数据放在 880—890 句两个 DATA 语句中, 如果不安排 760 句的条件判断和虚读技巧, 则发生 350 句的数据和 880 句的数据合并, 从而造成程序错误并中断。

6. 防止出错处理

80 句中 INPUT K\$ 是等待用户响应, 由操作者输入希望完成的功能代号。110 句是防止程序乱跑 (BOMB PROOFING)。程序中四种操作是用 1 到 4 这四个功能代号来响应的, 若使用者按错数字键, 则程序可能“爆炸”, 所以安排了“防爆”措施。这样做实际上是增加了是否在 1 到 4 以内的逻辑检查, 如出错, 则自动返回重新输入。它属于提高程序质量的方法。

7. 输入采用字符串变量

程序中的输入采用字符串变量 K\$ 而不用数值变量, 有两方面的考虑: 其一是防止和处理的数值混淆; 其二是输入方式灵活方便, 既可输入字符, 也可输入数字。例如, 要结束可键入“#”号, 经 90 句结束处理。而输入“1”, “2”, “3”, “4”, 也可以理解是字符, 经 100 句 VAL(K\$) 函数变成数值。

8. 模块化的程序结构

这是编制程序常用的方法。120 句是在分析用户的要求后进行分支处理, 显然用开关控制语句 ON/ GOSUB 最为合适。120 句后面的行号, 则是本程序相

应 4 个功能模块的入口。每个独立的模块, 分别完成注释中的相应功能, 在每个模块结尾处都设置了 RETURN 语句, 以使在完成本模块功能后, 正确返回总控, 根据需要再作其它操作。

```

10 REM 数据处理原理
20 DIM D(20); A(9), B(11), X(19)
30 RESTORE
40 PRINT "1. 数据插入"
50 PRINT "2. 数据删除"
60 PRINT "3. 数据修改"
70 PRINT "4. 数据合并"
80 INPUT K$
90 IF K$ = "#" THEN END
100 MO = VAL(K$)
110 IF MO < 1 OR MO > 4 THEN 80
120 ON MO GOSUB 140, 370, 560, 750
130 HOME; GOTO 30
140 REM 数据插入
150 FOR I=1 TO 10
160 READ D(I)
170 PRINT D(I); " ";
180 NEXT I
190 PRINT
200 K=0
210 INPUT "N="; N; REM 数据插入位置
220 IF N=-1 THEN 360
230 FOR I=10+K TO N STEP -1
240 D(I+1)=D(I)
250 NEXT I
260 K=K+1
270 INPUT "A="; A; REM 插入的数据
280 D(N)=A
290 PRINT
300 FOR I=1 TO 10+K
310 PRINT D(I); " ";
320 NEXT I
330 PRINT
340 GOTO 210
350 DATA 12, 47, 35, 26, 61, 83, 90, 55, 27, 88
360 RETURN
370 REM 数据删除
380 FOR I=1 TO 10
390 READ D(I)
400 PRINT D(I); " ";
410 NEXT I
420 PRINT
430 K=0
440 INPUT "N="; N; REM 删除的位置
450 IF N=-1 THEN RETURN
460 FOR I=N TO 9
470 D(I)=D(I+1)
480 NEXT I
490 K=K+1
500 PRINT
510 FOR I=1 TO 10-K

```

```

520 PRINT D(I);“”;
530 NEXT I
540 PRINT
550 GOTO 440
560 REM 数据修改
570 FOR I=1 TO 10
580 READ D(I)
590 PRINT D(I);“”;
600 NEXT I
610 PRINT
620 REM 连续修改
630 FOR I=1 TO 10
640 INPUT “N=”,N;REM 修改的位置
650 IF N=-1 THEN 690;REM 修改结束,显示,返回
660 INPUT “M=”,M;REM 修改的数据
670 D(N)=M
680 NEXT I
690 PRINT
700 FOR I=1 TO 10
710 PRINT D(I);“”;
720 NEXT I
730 FOR I=1 TO 2000;NEXT I
740 RETURN
750 REM 数据合并
760 IF MO=4 THEN FOR I=1 TO 10;READ S;NEXT I
770 FOR I=1 TO 8;READ A(I);NEXT
780 FOR I=1 TO 10;READ B(I);NEXT
790 I=1;J=1;N=0
800 N=N+1
810 IF A(I)>B(J) THEN LET X(N)=A(I);I=I+1;GO-
TO 830
820 X(N)=B(J);J=J+1
830 IF N<18 THEN 800
840 FOR I=1 TO 18
850 PRINT X(I);“,”;
860 NEXT
870 PRINT
880 DATA 98,97,89,85,84,71,69,63
890 DATA 96,94,92,90,87,83,72,69,60,55
900 FOR I=1 TO 2000;NEXT I
910 RETURN
]RUN
1. 数据插入
2. 数据删除
3. 数据修改
4. 数据合并
? 1
12 47 35 26 61 83 90 55 27 88
N=3
A=5555

12 47 5555 35 26 61 83 90 55 27 88
N=-1

```

```

1. 数据插入
2. 数据删除
3. 数据修改
4. 数据合并
? 2
12 47 35 26 61 83 90 55 27 88
N=3

```

```

12 47 26 61 83 90 55 27 88
N=-1

```

```

1. 数据插入
2. 数据删除
3. 数据修改
4. 数据合并
? 3
12 47 35 26 61 83 90 55 27 88
N=3
M=7777
N=-1

```

```

12 47 7777 26 61 83 90 55 27 88

```

```

1. 数据插入
2. 数据删除
3. 数据修改
4. 数据合并
? 4

```

```

98,97,96,94,92,90,89,87,85,84,83,72,71,69,69,
63,60,55,

```

```

1. 数据插入
2. 数据删除
3. 数据修改
4. 数据合并
? #

```

~~~~~  
(接第 48 页)

31. 使用打印汉字驱动程序应当注意哪些问题?  
应当注意下面三个问题:

(1)打印汉字驱动程序应与打印机的型号配套。不同型号的打印机,其驱动程序不同。

(2)打印汉字驱动程序应与要求的汉字库配套。例如长城 0520CH 机,在 GWBIOS 3.00 支持下,3070 打印机的打印驱动程序 3070C·EXE 要求 CLIB24 字库文件的长度为 588K,而 3·COM 则要求 CLIB24 字库文件为 607K。

(3)打印汉字驱动程序应与中文操作系统 CCDOS 版本配套。IBM PC/XT, GW0520A 型机等在 CCDOS 2.0 支持下,3070 打印机的驱动程序是 ALL24P·EXE,而在 CCDOS 4.0 版本下,3070 打印机的驱动程序是 3070P·COM 或 TH3070E·COM。

# 哥德巴赫偶数猜想的 BASIC 程序

汕头四中 陈君佐

哥德巴赫(Goldbach, 1690—1764)是德国人。他在1725年当选俄国彼得堡科学院院士。1742年,他在和大数学家欧拉的几次通信中,提出了关于正整数和素数之间关系的两个推测,用现在的话来说,就是:

1. 每一个不小于6的偶数都是两个奇素数之和;
2. 每一个不小于9的奇数都是三个奇素数之和。

二百多年来,这两个猜想吸引了许多数学家的兴趣。我们以符号(a,b)表示任一个大偶数都可表为一个不超过a个素因子的乘积和另一个不超过b个素因子的乘积之和。据潘承洞教授《哥德巴赫猜想》介绍,1920年前后,有人证明了命题{9,9}。1924年{7,7}被证明。1932年{6,6}被证明。1937年{5,7},{4,9},{3,15}被证明。1938年{5,5}被证明。1939年{4,4}被证明。1956年我国数学家王元证明了{3,4},1957年又证明了{2,3}。1962年潘承洞证明了{1,5}。1962年王元和潘承洞又分别证明了{1,4}。直到1966和1978年陈景润证明了{1,2},人们正一步步向{1,1}逼近。

下面,我用BASIC语言,编出一个快速寻求{1,1}素数对的程序:

```

10 INPUT A
20 B=INT(A/2)
30 M=0
40 FOR Y=B TO A
50 X=A-Y
60 IF Y=2*INT(Y/2) THEN 250
70 IF X=2*INT(X/2) THEN 250
80 IF X=3 THEN 200
90 IF X=1 THEN 250
100 SY=INT(SQR(Y+0.5))
110 SX=INT(SQR(X+0.5))
120 IF SX=2*INT(SX/2) THEN 150
130 SS=SX+2
140 GOTO 160
150 SS=SX+1
160 FOR N=3 TO SX STEP 2
170 IF Y=N*INT(Y/N) THEN 250
180 IF X=N*INT(X/N) THEN 250
190 NEXT N
200 FOR N=SS TO SY STEP 2
210 IF Y=N*INT(Y/N) THEN 250
220 NEXT N
230 M=M+1
240 PRINT M,X,"",Y
250 NEXT Y
280 PRINT
290 PRINT A,M
300 GOTO 10
    
```

运行程序后,输入100即得出

|     |    |    |
|-----|----|----|
| 1   | 47 | 53 |
| 2   | 41 | 59 |
| 3   | 29 | 71 |
| 4   | 17 | 83 |
| 5   | 11 | 89 |
| 6   | 3  | 97 |
| 100 | 6  |    |

用《哥德巴赫猜想》介绍的把偶数A表成的不同素数对的函数,就是:

$$D(100)=6$$

表示偶数100可表为6种不同数值的Goldbach素数对。

当分别输入200、300、400,可得 $D(200)=8$ , $D(300)=21$ 和 $D(400)=14$ 。

在汕头大学计算中心的VAX-Ⅱ机以及两台中型机上收集了大量的 $D(x)$ 值,发现,有65%的偶数x,满足:

$$D(x) \leq \frac{[\pi(x)]^2}{x} \quad (1)$$

如100以内有25个素数,用数论函数表示,就是 $\pi(100)=25$ ,代入知:

$$D(100)=6 \leq \frac{[\pi(100)]^2}{100} = \frac{25^2}{100} = 6.25$$

$$D(200)=8 \leq \frac{[\pi(200)]^2}{200} = \frac{46^2}{200} = 10.58$$

也即含素因子“2”和“5”的偶数,满足(1)式。

有30%左右的含素因子“3”的偶数,满足:

$$\frac{[\pi(x)]^2}{x} \leq D(x) \leq 2 \times \frac{[\pi(x)]^2}{x} \quad (2)$$

如:  $D(300)=21 \leq 2 \times \frac{[\pi(300)]^2}{300} = 2 \times \frac{62^2}{300}$

$$= 2 \times 12.81 = 25.62$$

$$D(600)=32 \leq 2 \times \frac{[\pi(600)]^2}{600}$$

$$= 2 \times \frac{109^2}{600} = 2 \times 19.8$$

$$= 39.6$$

只有极少数的富含素因子的偶数,才满足:

$$D(x) \geq 2 \times \frac{[\pi(x)]^2}{x} \quad (3)$$

如 $x=2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17=510510$ 这偶数

$$D(510510)=9493 \geq 2 \times \frac{[\pi(510510)]^2}{510510}$$

$$= 2 \times \frac{42331^2}{510510} = 2 \times 3510 = 7020$$

根据我们电算结果,认为1920年Hardy的渐近公式:

$$D(x) \sim 2C(x) \frac{x}{\log^2 x} \quad (A)$$



系数  $2C(x)$  偏大, 这里

$$C(x) = \prod_{p \leq x} \left(1 - \frac{1}{(p-1)^2}\right) \prod_{p \leq x} \frac{p-1}{p-2}$$

而陈景润 1978 年给出: 对充分大的  $N$ , 有

$$D(N) < 7.928C(N) \frac{N}{\log^2 N}$$

就更与实际计算有很大的偏离了。

1990 是一个偶数, 我把它送入 LASER310 机, 在 2 分钟 20 秒内, 她就给出 42 对 Goldbach 素数对来。

? 1990

|    |     |      |
|----|-----|------|
| 1  | 977 | 1013 |
| 2  | 971 | 1019 |
| 3  | 941 | 1049 |
| 4  | 929 | 1061 |
| 5  | 887 | 1103 |
| 6  | 881 | 1109 |
| 7  | 839 | 1151 |
| 8  | 827 | 1163 |
| 9  | 809 | 1181 |
| 10 | 797 | 1193 |
| 11 | 773 | 1217 |
| 12 | 761 | 1229 |
| 13 | 701 | 1289 |
| 14 | 683 | 1307 |
| 15 | 617 | 1373 |
| 16 | 563 | 1427 |
| 17 | 557 | 1433 |
| 18 | 509 | 1481 |
| 19 | 503 | 1487 |
| 20 | 491 | 1499 |
| 21 | 479 | 1511 |
| 22 | 467 | 1523 |
| 23 | 431 | 1559 |
| 24 | 419 | 1571 |
| 25 | 389 | 1601 |
| 26 | 383 | 1607 |
| 27 | 353 | 1637 |
| 28 | 293 | 1697 |
| 29 | 281 | 1709 |
| 30 | 269 | 1721 |
| 31 | 257 | 1733 |
| 32 | 179 | 1811 |
| 33 | 167 | 1823 |
| 34 | 113 | 1877 |
| 35 | 101 | 1889 |
| 36 | 89  | 1901 |
| 37 | 83  | 1907 |
| 38 | 59  | 1931 |
| 39 | 41  | 1949 |
| 40 | 17  | 1973 |
| 41 | 11  | 1979 |
| 42 | 3   | 1987 |

我从大量计算发现, 偶数越大, 能找到的 Goldbach 素数对越多。但不知为什么? 至今, 世界还有些数学家, 却坚持说大偶数, 找不到  $(1+1)$  的素数对。

## 梵塔移动的动态模拟

天津市大港一中 刘闯

1990 年七月举行的天津市计算机团体赛的第五题是: 动态模拟梵塔的移动, 要求打印移动过程如 “A → B” 等, 且移动速度可调。我编写了如下程序, 较好地完成了整个过程的模拟。为使移动过程看得更清楚并可暂停, 把移速可调改为响铃提示按某键继续。这里对程序稍作说明:

B 数组和 B, F 数组和 F, S 数组分别表示源塔, 目标塔, 中间塔;

X 数组记录三塔位置; P 数组记录三个塔上的塔盘数; A(a, b) 数组中 a 记录塔号, b 为层数, 整个记录各塔各盘的宽度。

程序 10~40 设置必要参量。

60 画初始时的 A 柱上各盘。

190~210 打印移动步数及方法, 模拟移动过程。

290~320 为画图子程序。

120~280 是求解移动路径。

5 REM 河内塔的动态模拟

10 INPUT “塔盘个数:”; N

20 INPUT “塔盘高度:”; H

30 X(1)=40; X(2)=140; X(3)=239; Y=140; P(1)=N; A(1,1)=48; B=1; CALL 5576; HPLLOT 0, 141-H TO 279, 141-H

35 VTAB 10; PRINT TAB (6) “A” TAB (18) “B” TAB (30) “C”

40 FOR I=2 TO N; A(1,I)=A(1,I-1)-2; NEXT

60 FOR I=1 TO N; P=I; GOSUB 290; NEXT

70 DIM N(20), B(20), F(20), S(20)

80 I=0; N(1)=N; J=1

90 B(1)=1; F(1)=3; S(1)=2

100 GOSUB 120

110 END

120 I=I+1

130 IF N(I)=0 THEN I=I-1; RETURN

140 N(I+1)=N(I)-1

150 B(I+1)=B(I)

160 F(I+1)=S(I)

170 S(I+1)=F(I)

180 GOSUB 120

190 VTAB 2; HTAB 16; PRINT J; J=J+1; B=B(I); F=F(I); PRINT CHR \$(7); GETA \$

200 VTAB 3; HTAB 14; PRINT CHR \$(64+B) “→” CHR \$(64+F); HCOLOR=4; P=P(B); GOSUB 300; P(F)=P(F)+1

210 A(F,P(F))=A(B,P(B)); P(B)=P(B)-1; B=F; P

```

=P(B);GOSUB 290
220 N(I+1)=N(I)-1
230 B(I+1)=S(I)
240 F(I+1)=F(I)
250 S(I+1)=B(I)
260 GOSUB 120
270 I=I-1
280 RETURN
290 HCOLOR=3
300 HPOINT X(B)-A(B,P),Y-H*P TO X(B)-A(B,
P),Y-H*(P+1) TO X(B)+A(B,P),Y-H*(P+1) TO X
(B)+A(B,P),Y-H*P
320 RETURN

```

## 查找内存中的代码

陈 宇

将程序清单的代码键入内存指定地址后,以“MEMORY SCAN”为名存盘,备以后使用。然后,再在监控状态下键入 300G 即可。将要寻找的字符串长度值放入 \$0 单元,要找的字符串依次从 \$1 单元放起,如要寻找“A9 00”字符串,可键入“0:2 A9 00”,接着键入“首地址。末地址 CTRL-Y”,程序便可在首地址至末地址之间的内存中为你寻找这一字符串,若找到一处,便印出地址数值。在寻找过程中,可按 CTRL-C 退出。

```

0300- A9 4C 8D F8 03 A9 10 8D
0308- F9 03 A9 03 8D FA 03 60
0310- A9 00 85 FD 85 FE A0 00
0318- A5 00 85 FF B1 3C D9 01
0320- 00 D0 11 C6 FF F0 04 C8
0328- 4C 1C 03 20 66 03 E6 FD
0330- D0 02 E6 FE AD 00 C0 C9
0338- 83 F0 13 E6 3C D0 02 E6
0340- 3D A5 3D 18 C5 3F 90 CE
0348- A5 3C C5 3E 90 C8 A9 8D
0350- 20 ED FD A0 05 B9 76 03
0358- 20 ED FD 88 10 F7 A5 FE
0360- 85 3D A5 FD 85 3C A5 3D
0368- 20 DA FD A5 3C 20 DA FD
0370- A9 A0 20 ED FD 60 A0 BA
0378- C4 CE C9 C6

```

## 奇妙的图案

宁夏中卫县中学 高一班 李明

我自己根据一个智能绘图仪的原理,编了下面这个程序,别看它很短,画出的图案却非常奇妙。

```

10 HGR;HCOLOR=3;INPUT "R,L=";R,L;D=
75/R;T=3.14159/180
20 HPOINT 140,5+R-L
30 A=A-15;A=A-INT(A/360)*360;I=I+
15/D;I=I-INT(I/360)*360
40 X=140+SIN(I*T)*(75-R)+SIN(A*T)
*L;Y=80-COS(I*T)*(75-R)-COS(A*T)*L
50 HPOINT TO X,Y;GOTO 30

```

此程序在 CEC-I(中华学习机)上通过。程序参数要求:R≤75,L≤R+5,下面参数可供参考:R,L=50,45 R,L=35,25 R,L=35,15 R,L=35,40 R,L=45,30 R,L=55,45 R,L=25 30 R,L=30,35 R,L=30,15 R,L=15,10 R,L=19,13 R,L=37,42

此程序原理如下:一个大圈,圈内侧有齿轮,还有许多大小不同的小圈,圈外侧也有齿轮,当两圈互相咬合,小圈在大圈做圆周运动时,小圈半径上任一点所走的轨迹就是图。

注:图案的精密程度,可改 30 行语句 A=A-5 和 I=I+15/D 中常数的大小(越小越精密)来调整。

## 最简数据盘生成法

江 东

利用微机进行各种事务管理,常把系统盘和数据盘各用一张磁盘,以获得最大存储空间。利用下面这个一程序,可以生成一个不含 HELLO 和 DOS 的 40 磁道纯数据盘。经过两年多的运行使用,在中英文状态下效果良好。附程序清单如下:

```

0 HOME;POKE42344,76;POKE44723,4;POKE
46922,96;POKE44725,164;POKE46063,40;
POKE48894,40;PRINT CHR$(4);"INIT HELLO,D2"

```

若去掉最后三个 POKE 语句,则可生成 35 磁道的纯数据盘(均能用 CATALOG 列目录)。

## 超级 8~40 单元打印

青岛九中微机室 王壮

在 APPLE 与中华学习机中,监控状态下对内存单元的列印只有一种形式,即一行列出八个单元的内容。在显示器上看很清楚,但输出至打印机时,却特别浪费纸。为此本人编写了这个打印程序以节省打印纸不必要的消耗。

本程序使用了内存中 \$ 2000~\$ 20A6 单元,主要是为了避免与其他机器语言程序发生冲突。程序设置了五种打印格式,分别为一行打印 8、16、24、32、40 个

```
2000-A9 0B 8D F9 03 A9 20 8D FA 03 60 A2 04 B5 3B 95
2020-FD A5 34 85 B8 20 B7 00 E6 B8 49 B0 F0 07 8D 84
2040-FD E6 08 A2 00 A0 00 A5 06 29 07 F0 02 A2 01 A5
2060-20 DA FD A9 A0 20 ED FD E6 06 A5 06 F0 21 4C 91
2080-ED FD E8 E0 04 D0 D7 A2 00 20 8E FD 4C 4F 20 E6
20A0-FD 20 93 FE 4C EA 03
```

## 将 16 进制代码翻译成 DATA 语句的数据

天津第七印刷厂设备科 张振堂

在 BASIC 语言编程过程中,经常需要设置一些 READ/DATA 语句,把一段机器语言子程序或一段图形数据 POKE 到指定单元。现将上述过程反过来,即把指定单元的数据由 16 进制变为 10 进制,并存放在 DATA 语句中。

键入程序代码,并以“MAKE DATA”为名存盘:

BSAVE MAKE DATA, A \$ 300, L \$ CE

使用时用 BRUN 将其调回内存运行。

也可根据需要在本程序调往其它单元运行,如:

BRUN MAKE DATA, A \$ 9500

此后,在编制程序过程中,可随时使用 & 命令将内存中代码转换为 DATA 语句。命令格式为

&m,n

m 为代码的首地址,n 为长度。所生成的 DATA 语句行号为原 BASIC 程序最大行号再加 10。

例一,机器语言代码首地址 1653,长度 34

&1653,34

本程序生成的 DATA 语句不应超过 249 个字节。对于过长的机器语言代码,可将其分为数段,再通过多次运行本程序生成多条 DATA 语句。

例二,机器语言代码首地址 4000,长度 179

可分为三段:

内存单元内容。您可以根据所要打印的程序的长短采用不同的打印格式。程序使用前不必用 PR#1 接通打印机。下面介绍使用方法:

首先运行本程序进行最初的设置,然后在您需要打印时键入如下命令:

首地址.末地址 **CTRL** **Y** 一行打印内存单元个数除以 8。例如您想打印内存中 \$ 2000~\$ 20A6 的内容,并且一行打印 40 个单元,在监控状态下可键入 2000.20A6 **CTRL** **Y** 5

在一行打印单元个数超过 24 时,程序会自动设置打印压缩字体。本程序适用于 CP80、DP80 等 EPSON 兼容打印机。

```
05 CA D0 F9 A9 C1 A2 36 20 A3 FE 20 EA 03 20 8E
20 C9 06 30 03 4C 99 20 C9 04 30 05 A9 0F 20 ED
07 20 DA FD A5 06 20 DA FD A9 AD 20 ED FD B1 06
20 A5 06 C5 08 F0 22 A5 06 29 07 D0 E1 A9 A0 20
07 A5 07 C5 09 30 E0 F0 D8 20 8E FD A9 12 20 ED
```

| 序 号 | 首 地 址        | 长 度 |
|-----|--------------|-----|
| 1   | 4000         | 60  |
| 2   | 4000+60=4060 | 59  |
| 3   | 4060+59=4119 | 60  |

分别运行本程序,可生成三条 DATA 语句。

&4000,60

&4060,59

&4119,60

```
0300- D8 18 AD 72 AA 69 18 8D
0308- F6 03 AD 73 AA 69 00 8D
0310- F7 03 A9 4C 8D F5 03 60
0318- 20 46 E7 86 F9 A9 00 85
0320- FC 85 FE 85 FF A9 FC 85
0328- FB A5 67 85 69 A5 68 85
0330- 6A A0 01 B1 69 F0 13 48
0338- 88 B1 69 A6 69 86 FB A6
0340- 6A 86 FC 85 69 68 85 6A
0348- D0 E7 98 91 69 C8 18 B1
0350- FB 69 0A 91 69 C8 A9 00
0358- 71 FB 91 69 C8 A9 83 91
0360- 69 18 98 65 69 85 69 90
0368- 02 E6 6A A9 60 8D 61 AE
0370- A4 FF B1 50 85 44 A2 00
0378- A9 02 85 FD A4 FD 20 44
0380- AE 84 FD D0 06 A4 FE E0
0388- 00 F0 09 E6 FE A4 FE 09
0390- 30 91 69 E8 C6 FD 10 E4
0398- E0 00 D0 05 A9 30 C8 91
03A0- 69 A9 2C C8 91 69 84 FE
```



03A8— E6 FF C6 F9 D0 C2 A9 00  
 03B0— 91 69 C8 91 69 C8 91 69  
 03B8— C8 98 18 65 69 85 69 90  
 03C0— 02 E6 6A A9 09 8D 61 AE  
 03C8— 20 74 D6 4C F2 D4

## 关于多目录磁道问题

天津市南开中学高二 张剑平

贵刊一九九〇年第八期刊登关炳坤同志的《如何在 APPLE II 及其兼容机上建立多个目录磁道》(下称《关》文)中,曾有编者按说,此文没有考虑各 VTOC 间协调管理问题。的确,《关》文提供的方法不完善,它未在各 VTOC 中把各目录磁道都置为已使用,因而可能遭到破坏。我对此进行了研究,发现可下列方法解决。

第一种方法:每一次存盘后,把相应的 VTOC 内容读入内存,利用《关》文所附 RWTS 程序,重复其操作步骤 2 即可。

第二种方法是利用本文所附程序对 VTOC 进行改制。其实根本不需每个目录道都有一个 VTOC,只要编一个小程序不断改变 VTOC 中 \$01 和 \$02 字节中的目录表指针即可。本程序即利用了这种思路。

程序在使用前要实际建立多目录道的情况改写 0 语句行的 N 值,然后按照 200 句中 DATA 的形式输入目录道中目录表首区所在磁道与扇区位置。运行后会自动显示已建的各目录道所在磁道位置,并让您选择使用哪一道,再进行 VTOC 的改制。

最好能以本程序作为 HELLO 程序初始化一张新盘,然后按《关》文的方法建立目录(当然不需每个子目录都拥有一个 VTOC,即不用执行操作步骤 2),接着在 VTOC 中把各目录道所占磁道的自由扇区图置为 \$00(表示已全部使用。)或 \$01(表示除第 0 区外其它扇区已被占用)。为了在任意目录道下都能正确调用 HELLO 程序,可把标准目录道的 \$0F 扇区的内容分别拷入各目录道的 \$0F 扇区中,当然不要忘了对各区中的 \$01~\$02 字节进行相应改动!

第三种方法最简单且行之有效。此法就是把磁道分为若干区域,使每个子目录分区自治,互不干涉。具体方法如下(请参考《关》文来阅读):

1. 初始化一张新盘。
2. 读出 \$11 道 0 区,把 VTOC 中的 \$84 字节后所有字节都置为 \$00,表示这些磁道已使用,再写入 \$11 道 0 区。
3. 接着把 \$01~\$02 改为 \$1E, \$0F。把 \$38~\$83 中所有内容置为 \$00。把 \$84 字节后的所有字节按以下形式改制:FFFF0000FFFF0000,改到 \$C4 字节即可。然后把它写入 \$1E 道 \$00 扇区。(注意: \$A8~\$AB 中仍为 00)。
4. 读出 \$11 道 \$0F 扇区,把 \$01~\$02 字节改

为 \$1E, \$0E 再写到 \$1E 道 \$0F 扇区中。

5. 把 \$1E 道第 \$0E~\$02 扇区的 \$01~\$02 字节分别改为 \$1E, \$0D;...; \$1E, \$01,再写回到原扇区。

至此一张有 2 个目录道的磁盘就建立好了。

```
0 BUFP=8192;TRK=788;SEC=789;
  CMD=796;N=3
5 HOME;FLASH;HTAB 13;PRINT
  "CHANGE DIR TRACK";NORMAL
10 HTAB 4;FOR I=1 TO 32;PRINT
  " * ";NEXT I;PRINT
20 FOR I=768 TO 807;READ A,POKE
  I,A;NEXT
30 DATA 169,3,160,16,32,217,3,169,0,133,72,96,
  0,0,0,0,1,96,1,0,17,0,37,3,0,32,0,0,1,0,0,
  96,1,255,255,255,0,1,239,216
35 PRINT,PRINT
40 FOR K=1 TO N
45 READ C(K,1),C(K,2)
50 PRINT TAB(11);K;"==> SUB T
  RACK:";C(K,1)
55 NEXT,PRINT,PRINT,PRINT
60 HTAB 8; FLASH : INPUT "USE WH
  ICH DIR TRACK?";X; NORMAL
70 POKE TRK, 17;POKE SEC,0;POKE
  CMD,1;CALL 768
80 POKE BUFP+1,C(X,1);POKE BU
  FP+2,C(X,2)
90 POKE CMD, 2;CALL 768
100 NEW
200 DATA T1,81,T2,82,Tn,Sn
```

## 电子表键盘 输入方式的改进

上海市新昌中学 田洪瑜

《电子与电脑》90 年第 7 期《具有电子表功能的键盘输入方式》一文(以下简称《表》文),提供了一种对键盘输入过程进行计时、限时和显时的方法。然而经用秒表测定发现,该程序还存在着一个根本的问题:无键盘输入时的计时虽准确,而有键盘输入时的计时误差却很大。例如,用标准时间 10 秒连续按某数字键输入,电脑电子表的显示却只有 5 秒,如此大的误差,严重影响了实际应用。

经分析发现,《表》文误差产生的原因是:有键盘输入时,程序要多运行 3030 行~3050 行,需要一定的时间,而 3015 行中的计时累加增量 0.079 却没有变化,从而产生了误差。改进的方法是使计时累加增量根据是否有键盘输入而变化,即增加

3008 N=0.079

3012 IF B>127 THEN N=0.279

3032 N=0.079

并将 3015 句改为

3015 K=K+N;IF K<60 THEN 3025

经如此改进,电脑电子表就能准确计时了。

另外,《表》文有三处排版失误,请作如下更正:

3005 VTAB 1;HTAB 3;PRINT“分 ”

3015 K=K+.079;IF K<60 THEN 3025

3025 VTAB1;HTAB 1;PRINT K0;VTAB 1;HTAB  
5;PRINT INT(K);“秒”

## LASER—310 兼做数字石英钟

北京六十一中 郑嘉琦

在 LASER310 上,不必增加硬件,只用软件方法就能使它象一台数字石英钟一样,在屏幕右上角显示时、分、秒。有趣的是,在走时的同时,计算机仍然可以照常使用。感兴趣的诸君不妨一试。其方法如下。

输入并运行下面程序。屏幕第二行将显示“START TIME:?”这时请键入开始走时的时、分数。例如再过一两分钟就到 9 点 25 分,就键入 9.25。之后屏幕右上角将以反白方式显示出 09.25:00 并同时在第四行显示出“PRESS RETURN TO START!!”。这时请对照手表,等待 9 点 25 分到点时立即按下回车键,屏幕右上角的显示开始走时。再键入 NEW 将程序清除就可以象往常一样使用计算机了。计算机的使用与走时显示将互不相干,“并列”进行。只是不要运行有 MODE(1)的程序,不要使用内存中的高分辨度显示区。

走时的快慢可以用改变两个内存单元中的数值进行调节。29340 单元管粗调,29361 单元管细调。数值加大走时变慢,数值减小走时变快。具体做法可先用 PEEK 查阅这两个单元的当前值,然后再根据实际走时的快慢用 POKE 来改变其中的数值。通过调节可以达到每天走时误差不超过 0.5 秒。一经调定,走时将非常稳定。这是由于走时的稳定程度决定于主机频率稳定度极高的石英晶体振荡器。

```
5 POKE 30845,201
10 DATA 62,50,237,71,62,114,50,127,120,62,101,
50,126,120
11 DATA 62,195,50,125,120,201,0,0,0,0,0
12 DATA 71,203,63,203,63,203,63,203,63,246,48,
119,35
13 DATA 120,230,15,246,48,119,201,0,0,0,0,0
14 DATA 33,24,112,58,1,116,205,75,114,35,54,46,
35,58
15 DATA 2,116,205,75,114,35,54,58,35,58,3,116,
205,75,114
```

```
16 DATA 237,87,61,237,71,192,62,50,237,71,58,3,
116,60
17 DATA 39,50,3,116,214,96,39,192,50,3,116,62,
61,237,71
18 DATA 58,2,116,60,39,50,2,116,214,96,39,192,
0,0
19 DATA 50,2,116,62,59,237,71,58,1,116,60,39,
50,1,116
20 DATA 214,19,39,192,0,0,62,1,50,1,116,201
100 POKE 30862,50
105 POKE 30863,114
110 FOR I=29234 TO I+149
115 READ A;POKE I,A;NEXT I
120 CLS;PRINT;INPUT“START TIME:”;A;T=A;E=0
130 IF A>=13 THEN T=A-12
135 IF A-INT(A)>=0.6 OR A>=24 THEN PRINT
“REDO”;GOTO 120
140 IF T>=10 THEN E=1
150 T=T*100;A$=STR$(T)
152 IF T<100 THEN A$=“H”+A$
153 IF T<10 THEN A$=“H”+A$
155 FOR J=1 TO 4;A$(J)=MID$(A$,J+E,1);
NEXT J
160 S=VAL(A$(1))*16+VAL(A$(2));F=VAL
(A$(3))*16+VAL(A$(4))
170 POKE 29697,S;POKE 29698,F
175 POKE 29699,0;N=0
180 FOR I=28696 TO I+7
190 IF I=28698 THEN 205
195 IF I=28701 THEN 205
200 N=N+1;POKE I,VAL(A$(N))+48
205 NEXT I
210 PRINT;PRINT;INPUT“PRESS RETURN TO
START!!”;B$
220 A=USR(0)
240 END
```

## 第三讲 命令行 Turbo C 和实用程序

天津师大计算机系 李文兵

本讲介绍用 DOS 命令行可执行的命令行 Turbo C 和实用程序 CPP、MAKE、TOUCH、GREP 的用法。

### 1. 命令行 Turbo C

命令行 Turbo C 包括如下程序：TCC——C 编译程序；TLINK——链接程序；TLIB——库管理程序。

命令行 Turbo C 还具有直接插入汇编指令的功能和汇编源程序输出功能(TC 没有这些功能)。

#### (1) TCC——命令行编译程序

TCC 比 TC 有更多的选项(表 3.1)，其用法是：

TCC 任选项 要编译的文件名

表 3.1 TCC 任选项

|       |               |        |              |
|-------|---------------|--------|--------------|
| -I    | 80/86/286 指令集 | -gN    | N 次警告后停止     |
| -A    | 禁止非 ANSI 扩展   | -iN    | 最大标识符长度      |
| -B    | 通过汇编程序编译      | -jN    | N 次错误后停止     |
| -C    | 允许注释嵌套        | -k     | 标准堆栈结构       |
| -Dxxx | 定义宏           | -IX    | 传 X 选项到链接程序。 |
| -Exxx | 选择汇编程序名       | -mc    | 紧缩模式         |
| -G    | 按时间优化         | -mh    | 巨型模式         |
| -Ixxx | 包含文件目录        | -ml    | 大型模式         |
| -K    | 缺省字符无符号型      | -mm    | 中型模式         |
| -L    | 库文件目标         | -ms    | 小型模式 *       |
| -O    | 优化转移指令        | -mt    | 微型模式         |
| -S    | 产生汇编输出        | -nxxx  | 输出文件目录       |
| -M    | 产生链接映射        | -oxxx  | 目标文件名        |
| -N    | 堆栈溢出检查        | -p     | pascal 调用    |
| -Uxxx | 未定义宏          | -r     | 寄存器变量 *      |
| -Z    | 寄存器优化         | -u     | 外部名下打下线符 *   |
| -a    | 产生字对齐         | -v     | 源级调试。        |
| -c    | 只编译，不链接       | -w     | 允许所有警告       |
| -d    | 合并重复字符串       | -Wxxx  | 允许 X×X 警告    |
| -exxx | 执行文件名         | -w-xxx | 不允许 X×X 警告   |
| -f    | 浮点仿真          | -y     | 产生行号信息       |
| -f87  | 8087 浮点运算     | -zxxx  | 设置段名         |

几点说明：

①打 \* 者为缺省项；②打 · 者为 2.0 版新增任选项；③-×- 表示任选项 X 之否定；

表 3.2 -Wxxx 任选项

(不符合 ANSI 标准)

|       |                                |
|-------|--------------------------------|
| -wdup | 重定义的标识符不一致                     |
| -wret | 同时使用了 return 与 return of value |
| -wstr | 标识符不是结构部分                      |
| -wstu | 标识符是未定义的结构                     |
| -wsus | 可疑的指针变换                        |
| -wvoi | void 函数不能返回值                   |
| -wzst | 长度为零的结构                        |

(常见错误)

|       |              |
|-------|--------------|
| -waus | 赋予标识符一个未用过的值 |
| -wdef | 标识符已定义过      |
| -weff | 无效代码         |
| -wpar | 标识符为从未用过的参数  |
| -wpia | 象是不正确的赋值     |
| -wrch | 不能实现的编码      |
| -wrvi | 函数应返回值       |

(不常见错误)

|       |                 |
|-------|-----------------|
| -wamb | 二义性运算符(要加括号)    |
| -wamp | 函数或数组有多余的 & 运算符 |
| -wnod | 函数未定义           |
| -wpro | 无函数原型           |
| -wstv | 用结构传值           |
| -wuse | 标识符未使用          |

(移植性错误)

|       |                               |
|-------|-------------------------------|
| -wapt | 无互换性指针赋值                      |
| -wcln | 常数太长                          |
| -wcpt | 无互换性指针比较                      |
| -wdgn | 比较的常数超出范围                     |
| -wrpt | 返回值转换中无互换性                    |
| -wsig | 类型变换中丢失位                      |
| -wucp | Signed 与 unsigned char 型指针相混合 |

[练习 3.1] 是 DOS 下使用 TCC 的一个实例：

```
C>tcc-IA;\include-IA;\lib-etest li.c
Turbo C Version 2.0 Copyright (c).....
li.c;
Turbo Link Version 2.0 Copyright (c).....
Available memory 408688
C>type li.c
main()
{printf("abcd.");}
```

#### (2) TLINK——链接程序

TLINK 是不受操作系统控制的。它把目标模块、库模块等链接成可执行的文件。用法如下：

Tlink 目标文件，执行文件，列表文件，库文件

表 3.3 TLINK 任选项

|    |                   |
|----|-------------------|
| /c | 在公共的和外部的符号中大小写有意义 |
| /d | 发现库中符号重复的警告       |
| /i | 初始化全部段            |
| /l | 插入源文件行号           |
| /m | 映象文件含公共符号         |
| /n | 无缺省库              |
| /s | 详尽的段映象            |
| /x | 无映象文件             |

关于 TLINK 的用法，请注意以下几点：

①被链接文件的扩展名可缺省。

②执行文件和列表文件可缺省,但其后的逗号不能缺省;执行文件缺省表示与目标文件同名;列表文件缺省表示与执行文件同名。

③任选项可放在命令行任何位置,如不想产生映象文件,且把源程序行号放入可执行文件时:

TLINK /x /l c0s MyFILE,,,cs

④多个目标文件用空格隔开;第一个目标文件必须是与编译该程序时所用存储模式一致的初始化模块名。初始化模块名与相应存储模式类型见表 3.4。

⑤库表用空格分开。标准库和数学库也须与编译程序所用存储模式一致,其对应关系也见表 3.4。

如果程序使用浮点数,须在命令行写入 EMU. Lib (无 8087/80287,使用浮点仿真库),或 FP87. Lib (有 8087/80287,使用 80x87 浮点库)。

表 3.4 存储模式及其相应的文件

| 存储模式 | 初始化模块   | 标准库    | 数学库       |
|------|---------|--------|-----------|
| 微型   | COT.obj | CT.lib | MATHt.lib |
| 小型   | COS.obj | CS.lib | MATHs.lib |
| 紧凑型  | COC.obj | CC.lib | MATHc.lib |
| 中型   | COM.obj | CM.lib | MATHm.lib |
| 大型   | COL.obj | CL.lib | MATHl.lib |
| 巨型   | COH.obj | CH.lib | MATHh.lib |

例如:使用小型存储模式,浮点仿真,链接名为 li 的目标文件,其 TLINK 命令行为:

Tlink c0s li,,,EMU CS MATHS

### (3)TLIB——库管理程序:

①功能 为建立目标形式的函数库提供方便。它有增加模块到库、从库中消除模块和从库中抽出一个.obj 文件三种功能。其命令的一般形式是:

TLIB 库名 [OP] 模块名 [OP]模块名...

(所指定的库)(操作符)(操作模块)

### ②注意事项:

- 库扩展名为 .Lib,操作模块扩展名为 .obj。
- 允许使用的操作符如表 3.5 所示

表 3.5 Tlib 的操作符

| 操作符   | 功能              |
|-------|-----------------|
| +     | 增加模块到指定库        |
| -     | 从指定库中消去模块       |
| *     | 从指定库中抽取 .obj 文件 |
| + 或 - | 消失同时又增加(即重新拷贝)  |
| * 或 * | 抽取同时又消去         |

• TLib 只能对整个模块进行操作

下面通过实例综合练习 Tcc, Tlib, Tlink 三个程序的用法。练习 3.2 中给出的序号为操作顺序。

### [练习 3.2]

(1)编辑求  $x^y$  的文件 sqr. c

```
C>type sqr. c
long sqr(x,y)
int x,y;
{ int i;
```

```
long j=1;
for(i=0;i<y;i++)
j=(long)j*x;
return(j);
```

(2)用 Tcc 编译该函数

C>tcc -c sqr. c

Turbo C Version 2.0 Copyright(c)...

sqr. c:

Available memory 419722

(3)用 dir 查看编译结果

C>dir sqr. \*

```
SQR  OBJ      260   1-01-80   12:05a
SQR  C       104   1-01-80   12:33a
2 File(s) 13703168 bytes free
```

(4)用 Tlib 把 sqr. lib 加入到 maths 库

C>tlib c:\turboc\lib\maths+sqr

TLIB Version 2.0 Copyright (c)...

(5)编辑调用函数 sqr()的文件 test. c

C>type test. c

#include <stdio. h>

#include "sqr. c"

main()

```
{ int x,y;
printf("Please Input:\n");
scanf("%d,%d",&x,&y);
printf("%d ^ %d = %ld\n",x,y,sqr(x,y));
}
```

(6)编译文件 test. c

C>tcc -c test. c

Turbo C Version 2.0 Copyright (c)....

test. c:

Available memory 85560

(7)查看编译结果

C>dir test. \*

```
TEST  C       162   1-01-80   12:14a
TEST  OBJ      459   1-01-80   12:17a
2 File(s) 13711360 bytes free
```

(8)用 Tlink 对 test. obj 文件进行链接

C>tlink c:\turboc\lib\c0s test,test,,c:\turboc\lib\emu

c:\turboc\lib\maths c:turboc\lib\cs

Turbo Link Version 2.0 Copyright (c)...

(9)查看链接结果

C>dir test. \*

```
TEST  C       162   1-01-80   12:14a
TEST  OBJ      459   1-01-80   12:17a
TEST  MAP      514   1-01-80   12:30a
TEST  EXE     9544   1-01-80   12:30a
```

4 File(s) 13709312 bytes free

(10)执行文件

C>test



Please Input:

2,4

2,4

2 ^ 4=16

对于源文件 test.c,若使用如下任选项编译,则可  
直接得到执行文件 test.exe

```
C>tcc -Ic:\turboc\include -Ic:\turboc\lib -ea;  
test a:test.c
```

Turbo C Version 2.0 Copyright (c) ....

a:test.c;

Turbo Link Version 2.0 Copyright (c) ....

Available memory 404038

C>dir a:test. \*

```
TEST OBJ 562 1-01-80 1:08a  
TEST C 162 1-01-80 12:14a  
TEST EXE 9544 1-01-80 12:04a  
3 File(s) 71680 bytes free
```

## 2. CPP——预处理程序

CPP 也是不受操作系统控制的应用程序,其功能  
是对源程序预处理。它最初由编译程序启动。使用 TC  
时,我们完全意识不到它。其用法如下:

### CPP 文件名

源文件经 CPP 预处理后,其中的宏指令均被预处  
理变为扩展名为 .i 的文件,如练习 3.3:

#### (1) 编辑含有宏指令的程序

```
C>type exp3-3.c  
#define MAX(x,y) (x>y)? x:y  
main()  
{int a=5,b=3  
printf("MAX=%d\n",MAX(a,b));  
}
```

#### (2) 用 CPP 预处理

```
C>CPP exp3-3.c  
CPP version 2.0 copyright (c) ....  
exp3-3.c  
Available memory 525430
```

#### (3) 用 type 命令查看预处理后的文件

```
C>type exp3-3.i  
exp3-3.c 1;  
exp3-3.c 2:main()  
exp3-3.c 3:{int a=5,b=3;  
exp3-3.c 4:printf("%d\n",(a>b)? a:b);  
exp3-3.c 5;}  
exp3-3.c 6;
```

表 3.6 CPP 任选项

|       |              |       |        |
|-------|--------------|-------|--------|
| -A    | 禁止非 ANSI 扩展  | -ml   | 大型模式   |
| -C    | 允许注释嵌套       | -mm   | 中型模式   |
| -Dxxx | 定义宏          | -ms   | 小型模式   |
| -Ixxx | Include 文件目录 | -mt   | 微型模式   |
| -P    | 包含源文件信息      | -nxxx | 输出文件目录 |
| -xxx  | 未定义宏         | -oxxx | 目标文件名  |

|     |          |        |           |
|-----|----------|--------|-----------|
| -gN | N 次警告后停止 | -p     | pascal 调用 |
| -iN | 最大标识符长度  | -w     | 允许所有警告    |
| -jN | N 次错误后停止 | -wxxx  | 允许×××警告   |
| -mc | 紧凑模式     | -w-xxx | 不允许×××警告  |
| -mh | 巨型模式     |        |           |

## 3. MAKE——程序管理工具

MAKE 也是不受操作系统控制的软件工具。

(1)功能 自动重新编译修改过的文件,避免对所有  
文件重新编译而造成时间浪费。

(2)Make 的启动 启动 MAKE 的命令是:

A>MAKE

该命令将编译所需的模块,建立可执行文件。当不  
指定其它文件且 MAKE 存在时,就执行其内容。要想  
使用其它文件(如 MYMAKE),就要用任选项-f,如:

A>MAKE -fMYMAKE

(3)任选项 MAKE 命令行的选项见下表。

| 任选项           | 含 义                      |
|---------------|--------------------------|
| -Didentifier  | 定义标识符 identifier         |
| -Diden=string | 把标识符定义为字符串(不能含空格与制表符)    |
| -Idirectory   | 在 directory 目录内检索包含      |
| - identifier  | 解除前面所定义的标识符              |
| -s            | 无该选项,MAKE 显示全部命令;否则不显示。  |
| -n            | 显示命令但不执行。用于观察 make 文件。   |
| -ffilename    | 把 filename 作为 MAKE 文件使用。 |
| -?,-h         | 显示帮助信息。                  |

#### [练习 3.4]

```
C>make -s -fsample.mak  
MAKE Version 2.0 Copyright (c) ....  
Available memory 247615 bytes
```

(4)MAKE 文件 由目标文件、源文件和命令表组  
成。目标文件由它所依存的源文件经编译产生。MAKE  
文件的一般形式如下:

```
目标文件 1:源文件 1  
命令序列  
目标文件 2:源文件 2  
命令序列
```

①目标文件须从最左列写起,其后紧跟冒号;

②命令形式为:若干空格[词头]命令

目标文件的词头有三个,即@、-num 和-。

@:表示执行命令时,不显示该命令

-num:若指定数值 num,则当退出状态大于 num  
时,MAKE 停止执行;若未指定,则当退出值不为零时,  
MAKE 停止执行,且删除目前的目标文件。

-:不检验退出状态,因此,继续执行 MAKE。

命令能够使用 MS-DOS 的全部命令,但改道命  
令和滚边命令例外。命令序列中的命令、任选项、文  
名等用空格或制表符隔开。

③注释以#开头,可跟在源文件表或命令序列之  
后;若注释单独占一行,就必须从最左列开始。

④目标文件定义与下一定义至少间隔为一空行。

(5)MAKE 文件组成要素 由注释、法则、隐含法则、宏定义、指令等五个基本要素组成。

①注释 是为便于阅读 MAKE 文件,而设置的以 # 为开头的说明,并不被 MAKE 执行。

②法则 用来描述文件间的依存关系与命令,其形式如下:

目标文件[目标文件...]:[源文件·源文件...]

[命令]

[命令]

例如:C.obj:c.c d.h

tcc -c -mm -fc.c

其中 C.obj 为目标文件,c.c 为源文件,d.h 为源文件,tcc -c -mm -f c.c 为命令。

法则执行过程如下:

- 若目标文件不存在则生成目标文件。
- 若目标文件存在,则检查其生成日期与时间,并与对应的每个源文件的生成日期与时间比较。若目标文件较新,则不执行命令;而当源文件较新时,则执行命令,制作相应的新目标文件。

③隐含法则 请看下面这些法则:

a.obj:a.c

tcc -c -mm -f a.c

C.obj:c.c

tcc -c -mm -f c.c

它们的形式完全相同,目标文件名与对应的源文件也分别相同,可以用如下一条隐含法则来表示。

-C.obj:

tcc -c -mm -f \$<

这条隐含法则表示,所有 .obj 文件皆依存于对应的 .C 文件,且通过 tcc 命令实现由 .C 到 .obj 的转变。这种隐含法则也适用于不用法则指定命令的情况。

④宏定义 MAKE 文件用宏定义的形式是:

宏名=宏展开字符

宏名用不含空格的英文字母、数字序列指定。宏展开字符可用英文字母、数字、空格等构成,末尾是回车换行符。所定义的宏名可按 \$(宏名)形式用在 MAKE 文件中。MAKE 将把 \$(宏名)替换为宏展开字符。如果宏名未定义,将替换为 Null 字符。例如下面的法则,第一行为宏定义。第 3 行为该宏定义的使用,MAKE 将把 \$(MCR)替换为 S。

MCR=S

a.obj:a.c

tcc -c -m \$(MCR) -f a.c

要想改变存储模式,只要改变宏定义即可。

⑤控制指令 MAKE 的控制指令(directives)有:

- 包含指令(! include) 该指令的功能是把 MAKE 文件所指定的文件包含进来,其形式为

!include "文件名"

有了这条指令就可把不同程序共用的部分放入一个专门文件,这样不管哪个 MAKE 文件就都能使用。

- 条件指令(!if, !elif, !else, !endif)

[用法 1] !if 表达式 若表达式成立  
语句 1 则执行语句 1、  
语句 2 语句 2...

!endif

[用法 2] !if 表达式 若表达式成立,  
语句 1 则执行语句 1,  
!else 否则执行语句 2

语句 2

!endif

[用法 3] !if 表达式 1 表达式 1 为真则执行  
语句 1 行语句 1;表达式 1  
!elif 表达式 2 为假,表达式 2 为  
语句 2 真则执行语句 2;若  
!endif 表达式 1、表达式 2  
皆假,则什么也不  
执行

语句 1、语句 2 包括宏定义、法则、隐含法则、include 指令等。!if 和 ! elif 中的表达式是 32 位带符号的整型表达式,由操作数、运算符构成。操作数可用 10、8、16 进制表示。可用运算符如下:

单项运算符有-、~、!;

双项运算符有+、-、\*、/、%、>、<、&、|、^、&&、||、>、<、>=、<=、==、!=;

三项运算符有?:。

- 错误指令(! error) 功能是使 MAKE 停止执行,显示文本。一般与条件命令组合使用,如:

!if !\$d(MCR)

!error MACRO not defined

!endif

- 解除命令(!undef) 解除以宏名定义的宏,用法:

!undef 宏名

⑥包含宏定义 MAKE 的包含宏定义用法有:

- \$(宏名) 宏名定义过,\$d(宏名)返回 1;否则返回 0。该宏可与条件命令组合使用,如:

!if !\$d(MCR)

MCR=S

!endif

表示若 MCR 未定义,则把 MCR 定义为 S。

- \$\*(基文件名宏) 它可以展开为不带扩展名的目标名。如:

OBJ=a.obj b.obj c.obj

target.exe=\$(OBJ)

tcc c0\$(MCR)a b c, \$\*, \$\*, \emu

math\$(MCR) c\$(MCR)

其中,\$\*用不带扩展名的 target.exe 替代即可。该例中的\$(OBJ)用宏所定义的 OBJ 的目录文件替代即可。此外,该宏也适用于隐含规则,如:

.c.obj

tcc \$\*

(转第 13 页)



# 普及型单片机开发工具的软件设计

中国科学技术大学 张培仁 刘振安

上期刊出了 KDC-III 的硬件结构,这里介绍其软件设计。

单片机开发机软件由在 IBM PC 运行的一个组合软件包和 KDC-III 上的监控程序两部分组成。

KDC-III 的软件结构如下:

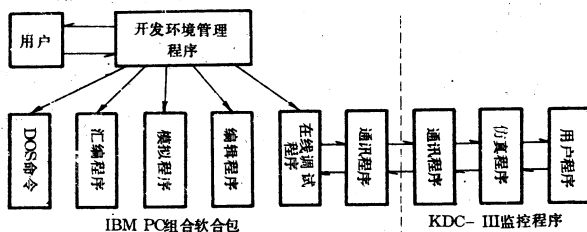


图 1

一、IBM PC 组合软件包主要由编辑程序,汇编程序,模拟调试程序,通信程序,在线调试程序共同组成。编辑程序用于用户汇编语言程序的输入;汇编程序将用户的源程序汇编成可执行的机器码文件;调试程序提供开发机调试命令,用户可通过这些命令控制开发机从而完成一系列的调试功能;组合软件包的各程序均可以直接在 DOS 系统下独立直接运行。

本软件动态地,通过人机对话的方式监督,控制和改变单片机系统的硬软件之间关系,为用户调试用户程序提供了灵活的调试手段。它允许用户检查,修改系统中 8031 内部寄存器 RAM。可单步,连续地执行用户程序。并可随意启停程序的执行;可采用设置断点的方法灵活实现硬软件的查错。

IBM PC 主机方面软件模块框图如下:

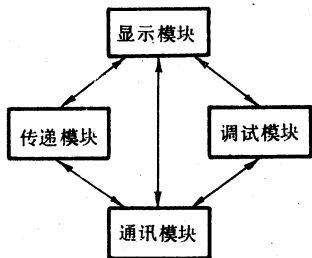


图 2

## 二、开发机的监控程序

监控程序主要由通信程序和仿真程序组成。通信程序是接收 PC 发来的各种命令,并根据命令把相应的信息返回给 PC;仿真程序主要包括单步,断点,全速

运行等命令的处理程序和中断处理程序。

单片机的总监控框图如图 5 所示。

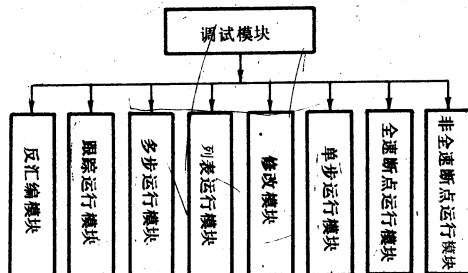


图 3

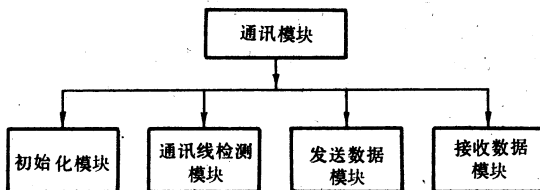


图 4

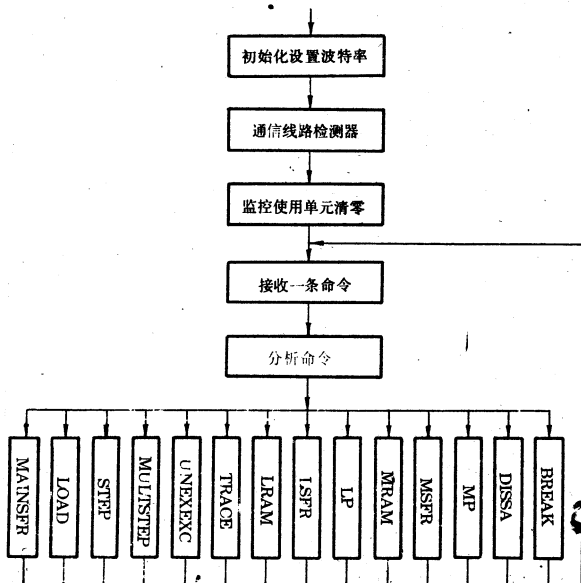


图 5

单片机调试程序框图如图 6

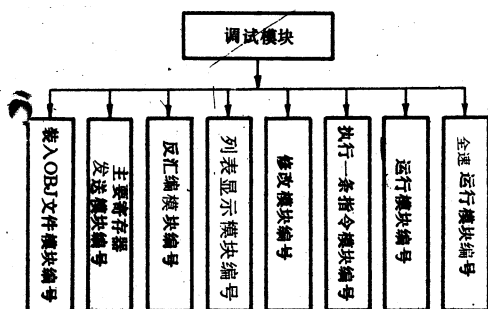


图 6

这里讲的调试程序就是仿真程序的用户界面，也可以讲仿真程序是调试程序的服务程序。调试程序的每一命令都要通过仿真程序中的相应模块在开发机上执行而实施。

### (三) 监控程序一些设计方法和说明。

监控程序主要由初始化模块、接收分析命令模块，以及各个命令相对应各个执行模块组成和 PC 通信的模块。

初始化模块是仿真器加电或复位后立即执行模块。它包括通信初始化模块和系统环境初始化模块组成。系统环境初始化主要负责系统堆栈，监控使用单元清零。

接收分析命令模块，主要通过调用通信程序的接收模块接收调试程序发送过来的命令码，并根据此命令码调不同的模块完成调试操作。各个命令子模块如 OBJ 文件加载模块，修改模块，单步模块，多步模块，跟踪模块，非全速断点运行模块，全速断点运行模块等。这些子模块都将调用两个最基本最低层两个通用子模块。即一个是将 8031 当前状态要内容送到 PC 的模块，另一个是控制 8031 执行一条指令的模块。

监控程序在设计时要解决主要问题如下：

1. 单片机资源的争夺的问题。因为我们开发机是单 CPU 的处理机。监控程序(即仿真处理机)和用户程序(目标系统处理机)合用一个 CPU。开发机一般工作方式是用户通过开发机的监控程序来控制用户程序运行和调试，一般情况下运行或调试一段用户程序后又返回监控。这样就产生了监控程序 and 用户程序运行时都必然会用一些寄存器如累加器 ACC, DPTR 等。为了确保用户程序正常运行。必须在 8031 内部(也可以在片外)设两个堆栈，从而把监控和用户程序有关数据分开。如果共用一个堆栈，就可能产生不可预知的现象。另外在任何时候都要使开发机知道目前是执行用户程序还是监控程序，为此必须有明确的标志。因为监控和用户在中断矢量，寄存器内容，执行指令都会有所不同，所以我们在 8031 内部开了一区域即 6AH~7FH 作为保存用户寄存器的内容和系统堆栈区的区域。从而较好解决单 CPU，监控和用户争夺资源的矛盾。这个方法的缺点是占用 8031 内部 22 字节 RAM。

2. 监控程序要快速地执行各个子命令的模块。

为了提仿真程序的速度，减少代码量，当监控通过接收分析模块接收到某个命令，经过查表跳转的方法

完成控制代码分析和相应模块的调用。

我们把命令按顺序编好码如下表所示：

|    |            |
|----|------------|
| 00 | 传送主要寄存器的内容 |
| 01 | OBJ 文件加载   |
| 02 | 单步         |
| ⋮  | ⋮          |
| 0E | 非全速断点运行    |
| 0F | 仿真程序复位     |
| 10 | 汇编         |

在仿真程序中再建立一个 17 个长跳转语句组成的跳转散表，(长跳转语句是 LJMP)TABEL 表中每一个语句分别指向一个模块的入口。

由于长跳转指令占三个字节，因此当接收到调试命令编码以后乘以 3 再用散转指令 JMP @A+DPTR 来实现相应模块的调用。

此模块程序如下：

```
BEGIN: LCALL RECEIV
        MOV DPTR, #TABEL
        MOV B, #03H
        MUL AB
        PUSH ACC
        MOV A, B
        ADD A, DPH
        MOV DPH, A
        POP ACC
        JMP @A+DPTR

TABEL: LJMP MAINSFR
        LJMP ....
        ⋮
```

### (三) 单步的实现。

对于一个调试程序控制用户程序一次执行一条指令是必不可少的基本手段。开发机通过单步可进一步实现多步运行，跟踪运行等调试手段。MCS—48 单片机有一个 SS 引脚，可用控制程序单步执行，MCS—96/98 有一条陷井软中断指令 TRACE 可用于控制单步执行。但 MCS—51 没有单独控制单步的引脚。但是利用 MCS—51 中断结构也可以实现单步。

在 MCS—51 中断系统中当另一个同级中断还在进行时，一个中断请求是不会得到响应的，而且 RETI 指令之后，至少要执行一条其它指令(非 RETI，或控制中断的指令)，这个中断请求才会得到响应。具体方法是开始确定单片机 CPU 控制权是否在监控程序手中，而后执行一段监控程序做好执行单步准备工作(包括置中断允许，置中断优先级，置外部中断 0 激发方式是电平方式等)准备工作完成之后，便激活外部中断 0，紧接着其后是一条返回指令 RETI。在执行这条指令时那个中断不会被响应，在执行这条指令之后，还要执行一条用户的程序，才又会响应那个中断，从而又一次返回监控。流程图如图 7。

多步，跟踪是基于单步基础上执行的。

多步运行步骤：

①接收程序开始地址。

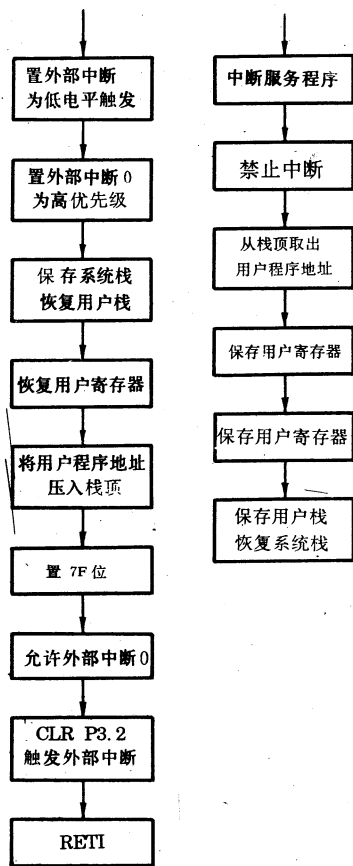


图 7

- ②接收执行步数。
- ③调单步操作模块。
- ④执行步数减 1。
- ⑤步数不等 0 转到③
- ⑥传送主要寄存器的值到 PC。
- ⑦结束。

跟踪运行步骤:

- ①接收程序的起始地址。
- ②调单步操作模块。
- ③传送主要寄存器的值到 PC。
- ④调反汇编模块。
- ⑤接收是否继续命令,若继续转到②。
- ⑥结束。

(四)断点运行的管理与实现。

断点运行对于一个完善的调试系统而言同单步一样是必不可少的调试手段。断点实现有两种方法即硬断点和软断点。我们为了降低成本使用软断点。

用软件方式实现断点运行,一种方法是每执行一条指令(由单步模块完成)就比较一下当前地址是否为断点地址,若已到断点地址就停下来。监控中仿真程序就用这个方法实现非全速断点运行。非全速断点运行由于每执行一条用户指令就需要有一系列的比较工作和监控,用户之间控制权转换的工作要进行。所以相对来说比较慢,我们称它为非全速运行。

非全速断点流程如下:

- ①接收程序起始地址。
- ②接收并保存断点地址。
- ③调单步操作模块。
- ④比较当前指令地址是否等于断点。若不等转③。
- ⑤传送主要寄存器的值到 PC 机。
- ⑥结束。

用软件实现断点运行还有一个方法。这个方法是参照 8086 系统中断点管理方式实现的。即先将断点处内容保存起来,取而代之一条转移到断点服务的程序的跳转指令,然后将控制权交给用户程序,并运行用户程序。直到用户程序运行到断点处为止,通过预先设置好的跳转指令返回监控。此方法为了区别前一种方法,所以称全速断点运行。

流程图如下:

- ①接收程序起始地址。
- ②接收并保存断点地址。
- ③保存断点起的三个字节的用户信息,将长转移指令 LJMP 0016 放入用户设置断点处。
- ④保存系统栈,恢复用户栈,恢复用户寄存器。
- ⑤程序起始地址进栈。
- ⑥执行 RET 指令,启动用户程序。

用户程序运行到断点处,便遇到 LJMP 0016 指令,返回到 0016 监控程序中去,从而控制权又回到监控程序中。

0016 处全速断点运行服务程序流程图如下:

- ①保存用户寄存器,保护用户栈,恢复系统栈。
- ②恢复断点处三个字节的用户信息。
- ③传送主要寄存器到 PC 机。
- ④结束。

全速断点运行速度快,而且用户程序完全脱离仿真程序(监控)的控制,故用户可占用全部的资源。这是全速断点运行的特点。但是由于这一方法修改断点处的内容,所以用户程序执行时不能和断点处后三个字节有联系,否则可能出错。

(五)用户程序的中断矢量的确定。

监控程序 and 用户程序都会利用中断执行相应的中断服务程序。MCS—51 规定五个中断源的中断矢量分别在 0003H,000BH……5 个相应地址中。这些地址已由 ROM(2764)构成。也就是不能修改了,所以进入中断以后,首先确定是监控使用中断还是用户使用中断。如果是用户,这时监控程序将用户中断服务程序的入口地址转到 2003H,200BH……相应地址处。2000H 开始给用户装入文件的 RAM 区,所以可以用户写入自己所需要中断服务程序入口地址。

总之单片机开发机的仿真程序还要解决一些其他模块问题,如 OBJ 文件装载,小汇编模块等问题。这里不再一一介绍了。

KDC—Ⅲ 开发机在线仿真软件文件名 ASI.PAS,单片机的监控程序文件名 P4.PRT。两个文件全部程序清单可向本刊编辑部函索,请寄印、邮费 15 元。





## 学装微电脑

学装  $\mu\text{P}-80$

# 8 位 A/D 变换部件的制作

易齐干

当连续变化模拟量的数据输入给微电脑时,必须把模拟量变换为数字量,本文介绍使用模拟量/数字量

表 1 TC5090AP 特性

|                                                                                  |
|----------------------------------------------------------------------------------|
| • 高精度 $\pm 1\text{LSB Max}$                                                      |
| • 功耗 $10\text{mW}$ ( $V_{\text{DD}}=5\text{V}$<br>$f_{\text{osc}}=1\text{MHz}$ ) |
| • 电源 $5\pm 1.5\text{V}$                                                          |
| • 变换时间 $2\text{ms}(\text{Max})$ ( $f_{\text{osc}}=1.5\text{MHz}$ )               |
| • 内部有 CR 基准振荡电路                                                                  |
| • 三态锁存输出                                                                         |
| • 输入/输出端为 TTL/CMOS 兼容                                                            |
| • 有自动补偿                                                                          |

(A/D)变换 IC(东芝 TC5090AP),将模拟量变换为 8 位数字量的方法。

1. TC5090AP—低功耗积分型 8 位 A/D 变换器。特性如表 1 所示,管脚分配如图 1,管脚说明如表 2 所示。

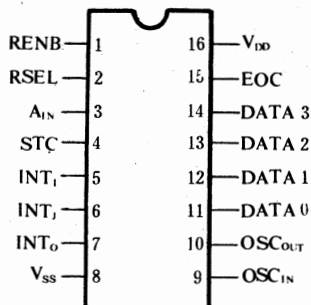


表 2 TC5090AP 管脚信号及说明

| No | 符号    | 名称与功能                                                                                          | No | 符号      | 名称与功能                                                                        |
|----|-------|------------------------------------------------------------------------------------------------|----|---------|------------------------------------------------------------------------------|
| 1  | RENB  | (Read Enable)<br>数据读出信号<br>“H”; DATA 0~3 输出<br>“L”; DATA 0~3 高阻抗                               | 9  | OSC IN  | Oscillator Input<br>Oscillator Output<br>输入基准时钟。<br>能使用石英振荡器或外部<br>电阻产生时钟信号。 |
| 2  | RSEL  | (Read Select)<br>4 位数据输出是选择高 4 位或是<br>选择低 4 位。<br>“H”; 高 4 位输出。<br>“L”; 低 4 位输出。               | 10 | OSC OUT | OSCIN 为输入,也可由<br>外部供给时钟信号。                                                   |
| 3  | AIN   | (Analog Input)<br>模拟量输入<br>电压输入范围 $V_{\text{SS}} \sim V_{\text{DD}}$                           | 11 | DATA 0  | 3 State Parallel Data<br>Output<br>输出变换数据<br>DATA 0 为 LSB<br>DATA 3 为 MSB    |
| 4  | STC   | (Start Conversion)<br>变换开始信号,STC 下降沿变换<br>开始。在 STC 下降过程中,<br>RSEL 以及 RENB 至少其中之<br>一为“H”电平。    | 12 | DATA 1  |                                                                              |
| 5  | INT I | Integrator Input<br>Integrator Junction<br>Integrator Output<br>外部电阻 $R_i$ 与电容 $C_i$<br>组成积分器。 | 13 | DATA 2  |                                                                              |
| 6  | INT J |                                                                                                | 14 | DATA 3  |                                                                              |
| 7  | INT 0 |                                                                                                | 15 | EOC     | (End of Conversion)<br>变换结束信号<br>如果 STC 下降,则为“L”电平,<br>如果变换结束,返回“H”电平。       |
| 8  | VSS   | (Digital Ground)<br>通常为 0V                                                                     | 16 | VDD     | (Power Supply)<br>$5\text{V} \pm 1.5\text{V}$                                |

图 1 TC5090AP

## 2. 8 位 A/D 变换部件的制作

8 位 A/D 变换部件的电路如图 2 所示。

采用外接电阻可获得时钟频率,外接电阻  $R_t = 47\text{k}\Omega$ ,时钟频率为  $500\text{kHz}$ 。两种关系如下:

$10\text{k}\Omega$  为  $1.5\text{MHz}$ ,  $20\text{k}\Omega$  约为  $1\text{MHz}$ ;  $30\text{k}\Omega$  约为  $800\text{kHz}$ 。

积分电路的电阻  $R_i$  与电路  $C_i$  应满足下式:

$$R_i \cdot C_i = (1.2 \sim 1.75) \cdot 1/f_{\text{osc}}$$

式中:  $R_i$ : 电阻 ( $\text{M}\Omega$ )

$C_i$ : 电容 ( $\mu\text{F}$ )

$f_{\text{osc}}$ : 振荡频率 ( $\text{kHz}$ )

本文为  $R_i = 1.5\text{M}\Omega$ ,  $C_i = 0.0015\mu\text{F}$

模拟量输入出脚 AIN 连续切换开关,可以输入由电位器得到的  $0 \sim 5\text{V}$  电压(切换开关在检验侧),也可输入由热敏电阻温度传感器获得的模拟电压(切换开关在测试侧)。

8 位 A/D 变换数据与 RSEL 端的高电平(H)、低电平(L)相对应(因为高 4 位和低 4 位均在 4 根数据线上输出)。控制端与数据线出脚能够与  $\mu\text{P}-80$  输入输出部件的端口 C 连接。

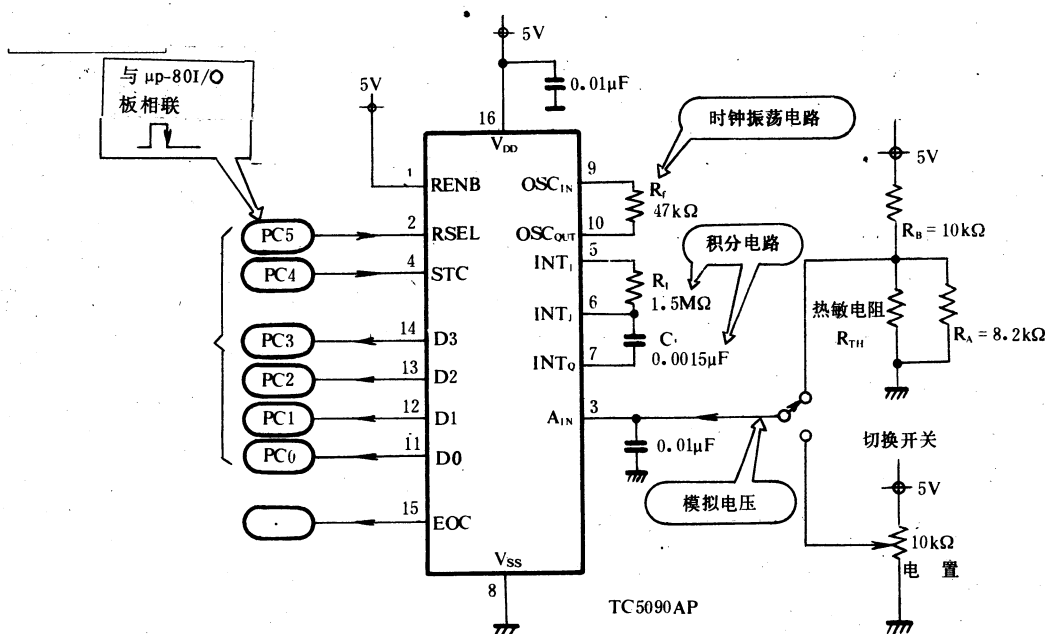


图 2

表 3 温度、热敏电阻与模拟电压的关系

| 温度 T (°C)                | 0                                                                       | 10  | 20  | 30  |
|--------------------------|-------------------------------------------------------------------------|-----|-----|-----|
| 热敏电阻 $R_{TH}$ (kΩ)       | 13.3                                                                    | 8.8 | 6.0 | 4.2 |
| 合成电阻 $R_T$ (kΩ)          | 5.1                                                                     | 4.2 | 3.5 | 2.8 |
| 模拟电压 $E_A$ (V)           | 1.7                                                                     | 1.5 | 1.3 | 1.1 |
| 每度的模拟电压 $\Delta E_A$ (V) | $\frac{0.2}{10} = 0.02$ $\frac{0.2}{10} = 0.02$ $\frac{0.2}{10} = 0.02$ |     |     |     |

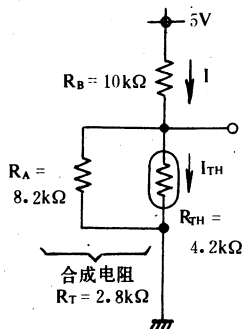
注 ①热敏电阻为石冢电子(株)的5DT型。

②合成电阻  $R_T$  是热敏电阻  $R_{TH}$  与电阻  $R_A$  的合成

$$R_T = \frac{R_{TH} \cdot R_A}{R_{TH} + R_A}$$

③模拟电压由下式求出

$$E_A = 5 \times \frac{R_T}{R_T + R_B}$$



$$I = \frac{5}{R_T + R_H} = \frac{5}{2.8 + 10} = 0.39 \text{ mA}$$

$$I_{TH} = I \times \frac{R_A}{R_{TH} + R_A} = 0.39 \times \frac{8.2}{4.2 + 8.2} = 0.26 \text{ mA}$$

注:  $R_A$ 、 $R_T$ 、 $R_{TH}$  的合成电阻

图 3

表 4 温度与 A/D 变换数据对应表

| 温度 (°C)  | 0   | 10  | 20  | 30  |
|----------|-----|-----|-----|-----|
| 模拟电压 (V) | 1.7 | 1.5 | 1.3 | 1.1 |
| 16进制数    | 57H | 4DH | 43H | 39H |

注: ①模拟电压的计算方法参照表 A。

②温度 0°C 时讨论 16 进制计算方法如下式。

因为 A/D 变换 IC 为 8 位, 所以平均 1H 等于 5/256 (V), 故下式成立。

$$x \times \frac{5}{256} = 1.7 \Rightarrow x = \frac{256 \times 1.7}{5} \approx 87$$

将 87 化为 16 进制数 87 : 16 = 5...7 故为 57H。

严格地讲, 微电脑通过检测 EOC 输出电平, 如果变换结束则向微电脑输入数据, STC 脉冲信号的下降沿是变换开始, 到变换结束需要 6ms 时间 (振荡频率为 500kHz 时) 之后, 要用软件向微电脑输入数据。

### 3. 应用举例

本实验 8 位 A/D 变换部件应用于输入温度并显示其结果。

热敏电阻本身温度与电阻值的变化不是线性关系, 为谋求线性关系, 要并联附加电阻, 由热敏电阻获得电压的变化, 还要串联电阻  $R_B$  (见图 2)。

TC5090AP 的分辨率为  $5 \div 256 \approx 0.02 \text{ V}$  即温度变化 1 度电压变化 0.02V, 按照这原则决定  $R_A$  和  $R_B$  的大小, 给以后程序编制带来很大方便。首先, 计算出各个温度值的模拟电压这样即可求出  $R_A$  和  $R_B$  的大小。现在,  $R_A = 8.2 \text{ k}\Omega$ ,  $R_B = 10 \text{ k}\Omega$ , 与各个温度相对应的模

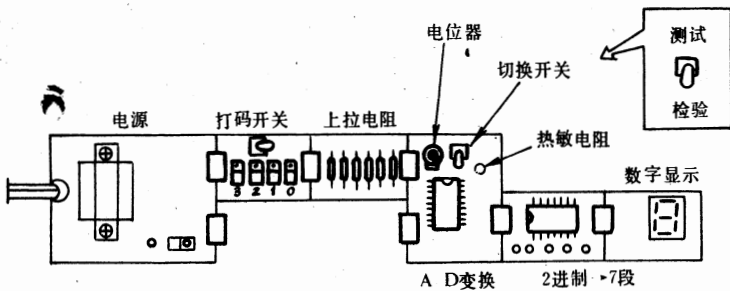


图 4

表 5 A/D 变换部件工作检验表

| No | 操作电位器              | 操作打码开关 $S_2$  | 操作结果                      |
|----|--------------------|---------------|---------------------------|
| 1  | 最左端<br>(模拟电压为 0V)  | ON → OFF → ON | $S_1$ OFF 变换数据的高 4 位显示 0. |
|    |                    |               | $S_1$ ON 变换数据的低 4 位显示 0.  |
| 2  | 中间<br>(模拟电压约 2.5V) | "             | $S_1$ OFF 变换数据的高 4 位显示 7. |
|    |                    |               | $S_1$ ON 变换数据的低 4 位显示 F.  |
| 3  | 最右端<br>(模拟电压为 5V)  | "             | $S_1$ OFF 变换数据的高 4 位显示 F. |
|    |                    |               | $S_1$ ON 变换数据的低 4 位显示 F.  |

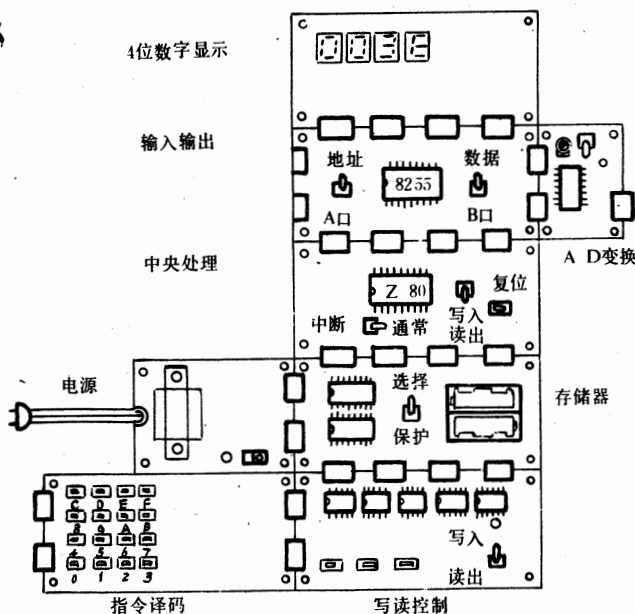


图 9

拟电压如表 3 所示。另外,为获得模拟电压的绝对值,要串联  $R_B=10k\Omega$  的电阻,其实,  $R_B<10k\Omega$  也可以,但是,如果  $R_B$  值过小,流过热敏电阻的电流变大,本身发热造成误差,这点决不能忽视,一般流过热敏电阻的电流不超过 0.5mA,按图 3 所列公式可以计算出流过热敏电阻的电流(30℃时)。

然后,计算各代表性温度所对应的 8 位数据。结果如表 4 所示。

制作完毕,要进行硬件检验。确认没有连线错误、虚焊、跨接等问题之后,将微电脑教育部件的打码开关、上拉电阻、2 进制—7 段、数字显示等部件。按图 4 所示连接。将 A/D 变换部件的切换开关扳至检验侧,操作打码开关  $S_0$ (变换开始)与  $S_1$ (高位输出数据与低位输出数据切换),数字显示结果如果与表 5 所示一致,说明工作正确,这样,预先对最初的基本动作进行检验,会給以后带来方便。

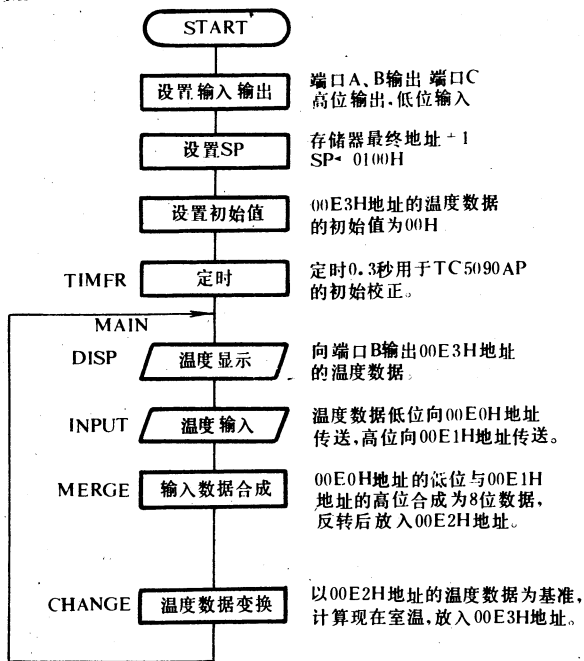
#### 4. A/D 变换程序

以上所述表 4 绘制温度与 A/D 变换数据对应表,发现随温度上升,A/D 变换数据变小,这样,使用起来很困难。因此,要如表 6 所示,将 A/D 变换数据进行反转,随温度由低变高,A/D 变换数据也与此相对应由小变大。由表 6 可知,A/D 变换的任何一个数据减去 A8H 之后,进行 10 进制调整,即为温度。

表 6 温度与 A/D 变换数据对应表

| 温度<br>(℃) | 模拟电压<br>(V) | A/D 变换数据<br>(16H) |    |
|-----------|-------------|-------------------|----|
|           |             | 正码                | 反码 |
| 0         | 1.7         | 57                | A8 |
| 1         |             | 56                | A9 |
| 2         |             | 55                | AA |
| 3         |             | 54                | AB |
| 4         |             | 53                | AC |
| 5         |             | 52                | AD |
| 6         |             | 51                | AE |
| 7         |             | 50                | AF |
| 8         |             | 4F                | B0 |
| 9         |             | 4E                | B1 |
| 10        | 1.5         | 4D                | B2 |
| 11        |             | 4C                | B3 |
| 12        |             | 4B                | B4 |
| 13        |             | 4A                | B5 |
| 14        |             | 49                | B6 |
| 15        |             | 48                | B7 |
| 16        |             | 47                | B8 |
| 17        |             | 46                | B9 |
| 18        |             | 45                | BA |
| 19        |             | 44                | BB |
| 20        | 1.3         | 43                | BC |
| 21        |             | 42                | BD |
| 22        |             | 41                | BE |
| 23        |             | 40                | BF |
| 24        |             | 3F                | C0 |
| 25        |             | 3E                | C1 |
| 26        |             | 3D                | C2 |
| 27        |             | 3C                | C3 |
| 28        |             | 3B                | C4 |
| 29        |             | 3A                | C5 |
| 30        | 1.1         | 39                | C6 |

现在,由流程图依次考虑温度测试显示程序。简略流程图与 RAM 区域图如图 5 所示。主要子程序的详细流程图如图 7~图 8 所示。要熟练地使用 A/D 变换



● RAM区域

| 地址  | 00E0H  | 00E1H  | 00E2H | 00E3H |
|-----|--------|--------|-------|-------|
| 分配  | 输入(低位) | 输入(高位) | 合成数据  | 温度    |
| 初始值 |        |        |       |       |

数据变换。

图 5

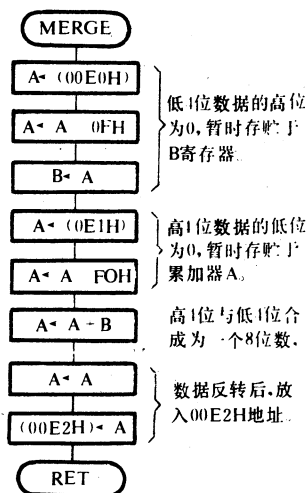


图 7

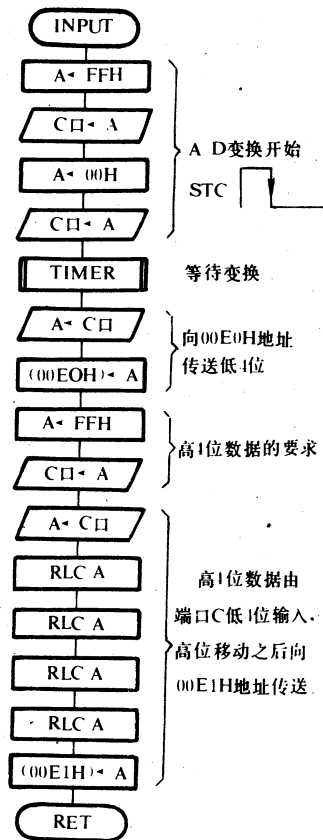


图 6

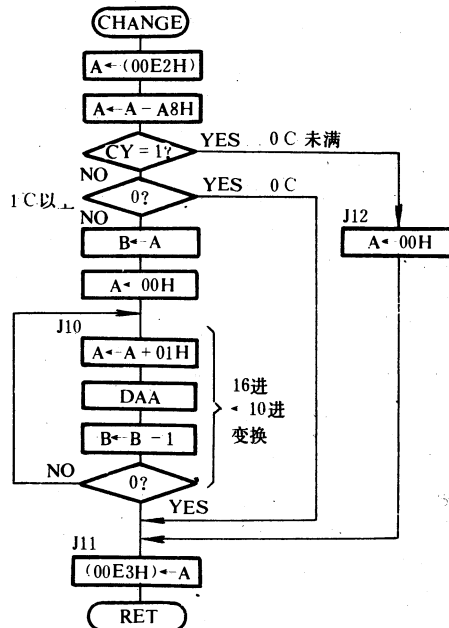


图 8

表7 程序清单

| 标记                                         | 助记符           | 地址   | 机器语言     | 注释             |
|--------------------------------------------|---------------|------|----------|----------------|
| ART                                        | LD A, 8H      | 0000 | 3E 81    | 设置输入输出         |
|                                            | OUT(03H), A   | 0002 | D3 03    |                |
| MAIN                                       | LD SP, 0100H  | 0004 | 31 00 01 | 设置 SP<br>设置初始值 |
|                                            | LD A, 00H     | 0007 | 3E 00    |                |
|                                            | LD(00E3H), A  | 0009 | 32 E3 00 |                |
|                                            | CALL TIMER    | 000C | CD 20 00 |                |
|                                            | CALL DISP     | 000F | CD 30 00 |                |
|                                            | CALL INPUT    | 0012 | CD 40 00 |                |
|                                            | CALL MERGE    | 0015 | CD 70 00 |                |
|                                            | CALL CHANGE   | 0018 | CD 90 00 |                |
|                                            | JP MAIN       | 001B | C3 0F 00 |                |
| TIMER<br>J40<br>J41                        | LD D, 5CH     | 0020 | 16 5C    | 延时子程序          |
|                                            | LD E, 5FH     | 0022 | 1E 5F    |                |
|                                            | DEC D         | 0024 | 1D       |                |
|                                            | JP NZ, J41    | 0025 | C2 24 00 |                |
|                                            | DEC D         | 0028 | 15       |                |
| DISP                                       | JP NZ, J40    | 0029 | C2 22 00 | 温度显示<br>子程序    |
|                                            | RET           | 002C | C9       |                |
|                                            | LD A, (00E3H) | 0030 | 3A E3 00 |                |
|                                            | OUT(01H), A   | 0033 | D3 01    |                |
|                                            | RET           | 0035 | C9       |                |
| INPUT                                      | LD A, FFH     | 0040 | 3C FF    | 温度输入<br>子程序    |
|                                            | OUT(02H), A   | 0042 | D3 02    |                |
|                                            | LD A, 00H     | 0044 | 3E 00    |                |
|                                            | OUT(02H), A   | 0046 | D3 02    |                |
|                                            | CALL TIMER    | 0048 | CD 20 00 |                |
|                                            | IN A, (02H)   | 004B | DB 02    |                |
|                                            | LD(00E0H), A  | 004D | 32 E0 00 |                |
|                                            | LD A, FFH     | 0050 | 3E FF    |                |
|                                            | OUT(02H), A   | 0052 | D3 02    |                |
|                                            | IN A, (02H)   | 0054 | DB 02    |                |
|                                            | RLCA          | 0056 | CB 07    |                |
|                                            | RLCA          | 0058 | CB 07    |                |
|                                            | RLCA          | 005A | CB 07    |                |
|                                            | RLCA          | 005C | CB 07    |                |
|                                            | LD(00E1H), A  | 005E | 32 E1 00 |                |
|                                            | RET           | 0061 | C9       |                |
| MERGE                                      | LD A, (00E0H) | 0070 | 3A E0 00 | 输入数据<br>合成子程序  |
|                                            | AND 0FH       | 0073 | 36 0F    |                |
|                                            | LD B, A       | 0075 | 47       |                |
|                                            | LD A, (00E1H) | 0076 | 3A E1 00 |                |
|                                            | AND 0FH       | 0079 | E6 F0    |                |
|                                            | ADD A, B      | 007B | 80       |                |
|                                            | CPL           | 007C | 2F       |                |
|                                            | LD(00E2H), A  | 007D | 22 E2 00 |                |
| CHANGE<br><br>J10<br><br>J11<br>J12<br>J12 | RET           | 0080 | C9       | 温度数据<br>变换子程序  |
|                                            | LD A, (00E2H) | 0090 | 3A E2 00 |                |
|                                            | SUB A8H       | 0093 | D6 A8    |                |
|                                            | JP C, J12     | 0095 | DA A9    |                |
|                                            | JP Z, J11     | 0098 | 0A A5    |                |
|                                            | LD B, A       | 009B | 47       |                |
|                                            | LD A, 00H     | 009C | 3E 00    |                |
|                                            | ADD 01H       | 009E | C6 01    |                |
|                                            | DAA           | 00A0 | 27       |                |
|                                            | DEC B         | 00A1 | 05       |                |
|                                            | JP NZ, J10    | 00A2 | C2 9E 00 |                |
|                                            | LD(00E3H), A  | 00A5 | 32 E3 00 |                |
|                                            | RET           | 00A8 | C9       |                |
|                                            | LD A, 00H     | 00A9 | 3E 00    |                |
|                                            | JP J11        | 00AB | C3 A5 00 |                |

6)、MERGE:输入数据合成子程序(图7)、CHANGE:温度数据变换子程序(图8)。以流程图为基础,编制程序清单。如表7所示。

#### 5. 执行程序

首先,参照图9、连接 $\mu P-80$ 与8位A/D变换部件。然后,写入表7的程序,将A/D变换部件的切换开关扳在检验侧,旋转电位器,端口B显示应该由00变到87(输入输出部件的显示切换开关在端口A、端口B一侧)。

检验结束,A/D变换部件的切换开关在测试侧,接通代替热敏电阻的 $6k\Omega$ 电阻,它相当于 $20^{\circ}\text{C}$ 。此时,应该显示 $20^{\circ}\text{C}$ 左右(电源电压与电阻平衡,可能有 $\pm 2$ 度误差)。这时,要调整温度输入电路的串联电阻RB的数值,希望显示为 $20^{\circ}\text{C}$ ,如果显示 $20^{\circ}\text{C}$ ,连接热敏电阻。

本实验温度显示以1度为单位,如果使用运算放大器放大模拟电压,平均1度的电压变化量由20mV到40mV时,测试精度可达到0.5度。在模拟量输入端,如果连接热敏电阻以外的传感器,也能显示由该传感器决定的测试量。有兴趣的读者不妨动手实验。

### 首届电子版图书全国大联展即将开幕

为了贯彻中央十三届七中全会关于加速发展电子工业的方针,进一步落实中央关于“一手抓整顿、一手抓繁荣”的指示精神,繁荣文化图书市场,缓解“卖书难和买书难”的矛盾,电子工业出版社定于1991年4月25日至5月10日在北京、上海、天津、广州、南京、西安、武汉、沈阳、重庆、济南、杭州等30多个大中城市的40多家新华书店同时举办“电子工业出版社图书首届全国大联展”。

本届联展共将展销电子版图书十五大类,近千个品种。其中九〇年以来出版的新书及重印书300余种。这些图书基本上覆盖了电子科学技术的各个专业;兼顾了不同层次读者的需求,除了有大批实用性很强的工程技术图书和科普读物参展外,还有不少具有较高学术水平的著作和读者见面。在展销期间,电子工业出版社软件出版部将提供一百余种计算机辅助教学软件和近百种PC机应用软件供读者选购。此外还有若干音像出版物上市。

#### 电子工业出版社软件部

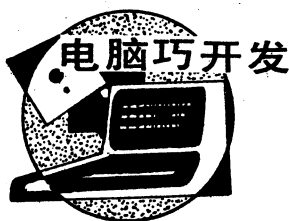
#### 举办软件优惠酬宾活动

为促进计算机软件的推广应用,软件部决定在我社图书首届全国大联展期间举办“软件优惠酬宾活动”,即在1991年4月1日~5月31日期间,实行特优惠价销售软件:CEC—I和Apple—III软件优惠价为目录定价的75%,IBM PC及其兼容机软件为目录定价的85%。欲购者从速函索“软件优惠酬宾”卡和“软件目录”。联系地址:北京万寿路173信箱软件部 邮码:100036

联系人:谈众安 电话:815342

部件,必须认真理解 INPUT:温度输入子程序(图





## MP-I 微电脑加装编码键盘

李志平

MP-1 学习机结构简单,实用性强,经过开发能胜任复杂的工作。只是原机的数据键为 8 位拨动开关,输入程序时颇费心思,且容易出错。本人给 MP-1 机加装了一个简单的编码键盘,输入程序时甚为简便。

### 一、工作原理

电路见图一,共有十六个键,可以打出 0 到 F 十六个数据,当没有按下按键时  $S_3, S_2, S_1, S_0$  均为高电平,当一个键被按下时,即有一根  $y$  线和  $x$  线被短接,对应的三极管有导通电压而导通,于是被短接的  $y$  线通过  $be$  极接地,变为低电平, $x$  线也因三极管导通而变为低电压,也就是说每按下一个键,都导致和此键相连的  $x$  线和  $y$  线变为低电压,把  $x$  线和  $y$  线的电位变

化通过二极管矩阵编码,即获得与被按键的键面数字相符的对应的二进制数码(高电平为 0),同时  $x$  电位变化也引起  $D$  点变化, $D$  为按键检测点,各按键按下时  $S_3, S_2, S_1, S_0$   $D$  输出电位及代码如表所列。

$S_3, S_2, S_1, S_0$  输出电位通过  $IC_1$  (施密特触发器)整形反相输入到  $IC_3, IC_4$  存储起来。 $IC_3$  和  $IC_4$  为 4 位  $D$  锁存触发器, $D$  的电位通过整形作为  $IC_5$  (JK 触发器)的时钟信号。每来一次  $D$  信号,JK 就翻转一次,JK 的  $Q$  和  $\bar{Q}$  电位通过电容器,二极管变为正尖脉冲,以控制两组  $D$  触发器轮流触发,按下键时  $S_3, S_2, S_1, S_0$  的信号在两组  $D$  触发器输入端等候,假定按键前 JK 的  $Q=1$ ,按下键时  $\bar{Q}=1$ ,高 4 位  $D$  触发器有时钟脉

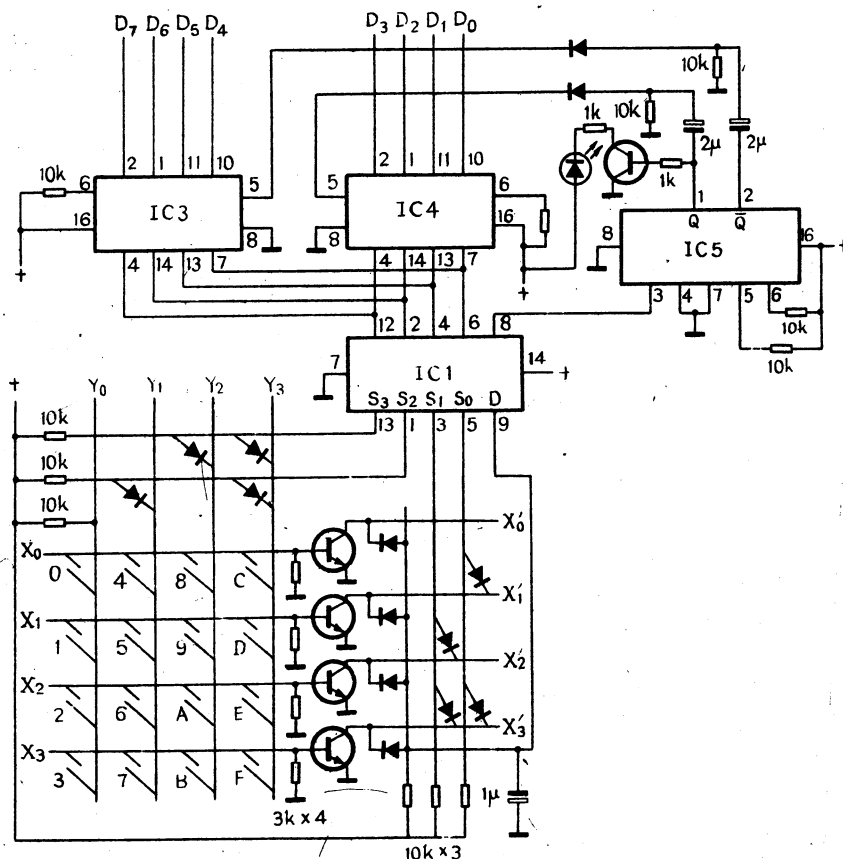


图 1

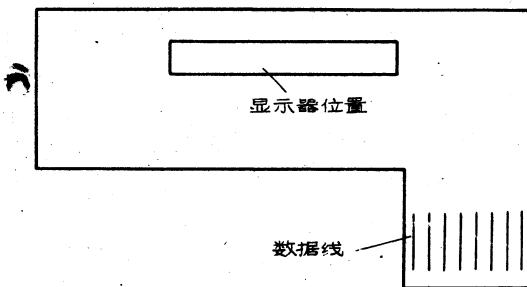


图 2

冲,被触发(即存入了  $S_3, S_2, S_1, S_0$ 。再按一次键时,低 4 位 D 触发器被触发,第三次按键时,高 4 位又工作,如此反复。在 JK 的 Q 端有一指示灯,灯亮时即可开始输入数据,第一次输入是高 4 位。

另外,多数程序输入时都按顺序从小到大输入的。在  $A_1$  键上并联一个三极管开关,由 WR 键控制输入程序就方便多了。输入程序时用  $A_1$  或  $A_n$  找到需要的入口地址,按数据键使指示灯亮,输入高 4 位,接着输入低 4 位,按 WR 键数据就输入内存单元,放开 WR 键,程序计数器加 1,再输入下一个数据。

## 二、安装与元件选择

电路板设计如图二形状,以便于安装在原机上,按键时宜选用微型微动开关,IC<sub>1</sub> 为 CD40106,IC<sub>4</sub> 和 IC<sub>3</sub> 为 CD4042,IC<sub>5</sub> 用 CD4027,三极管可用 3DG201 等廉

|   | $S_3$ | $S_2$ | $S_1$ | $S_0$ | D | 代码   |
|---|-------|-------|-------|-------|---|------|
| 0 | △     | △     | △     | △     | / | 0000 |
| 1 | △     | △     | △     | ↓     | / | 0001 |
| 2 | △     | △     | ↓     | △     | / | 0010 |
| 3 | △     | △     | ↓     | ↓     | / | 0011 |
| 4 | △     | ↓     | △     | △     | / | 0100 |
| 5 | △     | ↓     | △     | ↓     | / | 0101 |
| 6 | △     | ↓     | ↓     | △     | / | 0110 |
| 7 | △     | ↓     | ↓     | ↓     | / | 0111 |
| 8 | ↓     | △     | △     | △     | / | 1000 |
| 9 | ↓     | △     | △     | ↓     | / | 1001 |
| A | ↓     | △     | ↓     | △     | / | 1010 |
| B | ↓     | △     | ↓     | ↓     | / | 1011 |
| C | ↓     | ↓     | △     | △     | / | 1100 |
| D | ↓     | ↓     | △     | ↓     | / | 1101 |
| E | ↓     | ↓     | ↓     | △     | / | 1110 |
| F | ↓     | ↓     | ↓     | ↓     | / | 1111 |

△ 代表高电平 ↓ 代表低电平

图 3

价管子。安装时不能带电焊接 IC。最好能用 IC 插座安装。电路板上还可装一个 8 位插座,可把数据输出到 MP-1 机的输入口进行“现场”控制。

# 扫描式红外线十路遥控器

王刚勤

本遥控器是采用电子扫描而实现对音频译码器中心频率的捕捉,且只用一个锁相环音频译码器,就可分离出十种不相同的中心频率信号,从而大大减化了控制电路。该遥控电路设计新颖、实用性强,与其它遥控电路相比,具有体积小、重量轻、性能价格比高等优点。可对各种家用电器及其它需要多路控制的装置和器具进行遥控。

## 一、工作原理

图 1 和图 2 是扫描式红外线十路遥控器的接收电路。图 3 是扫描式红外线十路遥控器的发射电路。现把以上两种电路的工作情况分别叙述如下:

1. 发射电路 由 IC 等元件组成一个多谐振荡器,其振荡频率由  $1.443/(r+2R_{11})C_2$  决定,  $r$  为  $R_1 \sim R_{10}$  其中某一个电阻。  $AN_1 \sim AN_{10}$  为 10 个按键开关。倘若按下某一个按键遥控开关时,使得 IC 所组成的多谐振荡器工作,则振荡信号从③脚输出,故红外发射管 FG 发出一定频率的红外光。只要轮流按一下  $AN_1 \sim AN_{10}$

开关,它便产生出十种不相同的红外线遥控发射频率。

2. 接收电路 主要由红外线接收探头、放大电路、音频译码器、脉冲产生器、十进制计数器/脉冲分配器、模拟数字传输开关、双稳态电路、与非门和非门电路、电子开关、复位电路、执行电路等组成。LM567 是一种锁相环(PLL)音频译码器,它具有高噪声抑制、窄通频带、抗干扰性好、中心频率稳定等特点。LM567 的中心频率由  $1/1.1RC_9$  决定( $R$  为  $R_{20} \sim R_{29}$  其中某一个电阻);工作频率可在 0.01~500 千赫;工作电压 4.75~9 伏;第③脚输入端,要求输入信号 > 25 毫伏;第⑧脚输出端,允许最大灌电流为 100 毫安;静态工作电流为 8 毫安。LM567 的基本工作过程如下:当在器件③脚输入 > 25 毫伏的振荡信号,且输入信号频率与 LM567 的中心频率相同时,则⑧脚由高电平跳变到低电平。

平时,红外接收管 HG 不受到红外光作用时,其音频译码器 IC<sub>4</sub> 的输出端⑧脚呈高电平,IC<sub>1</sub>、IC<sub>2</sub>、IC<sub>5</sub>~IC<sub>7</sub> 均处于工作状态,并且快速向 LM567 的 10 路遥控

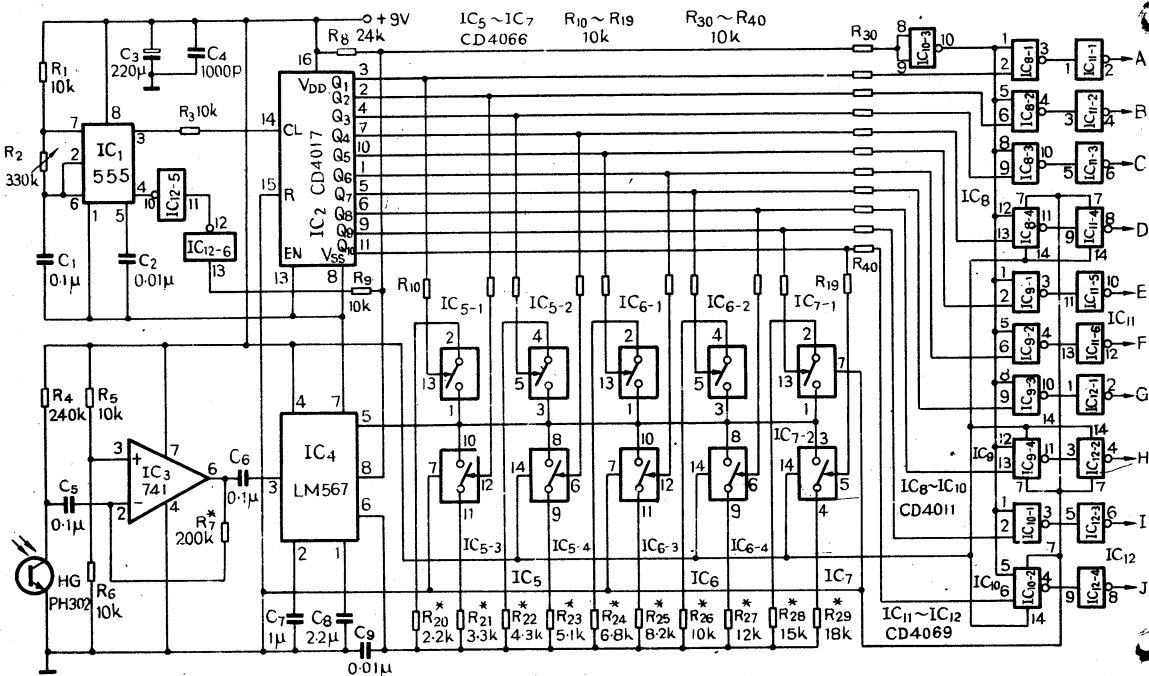


图 1

管 FG 的微弱红外光信号,经  $C_5$  耦合加到运算放大器  $IC_3$  的②脚上进行放大。被放大后的输入振荡信号从  $IC_3$  的⑥脚输出,由  $C_6$  耦合加至音频译码器  $IC_4$  的输入端③脚上去捕捉  $IC_4$  的中心频率。与此同时,  $IC_1$  产生的脉冲振荡信号从③脚输出,其振荡频率由  $1.443 / (R_1 + 2R_2)C_1$  决定,频率  $f$  约为 20 赫左右。然后,由  $R_3$  加到十进制计数器/脉冲分配器  $IC_2$  的⑭脚上进行计数译码输出,使得  $Q_1 \sim Q_{10}$  轮流地输出正脉冲信号。由于  $IC_1$  的振荡周期  $T$  约 0.05 秒,故使  $Q_1 \sim Q_{10}$  的输出端分别在 0.5 秒时获得正脉冲信号一次。当  $IC_2$  的③脚输出正脉冲信号,使得  $IC_{5-1}$  模拟开关接通,相应接通  $IC_4$  的⑤脚与⑥脚之间的电阻  $R_{20}$ ,随之  $IC_4$  产生的中心频率与接收到的发射频率相同,此时⑧脚由高电平跳变到低电平。然后,  $IC_{10-3}$  的⑩脚输出高电平,其  $IC_{8-1}$  的两个输出端①和②脚均为高电平,③脚输出低

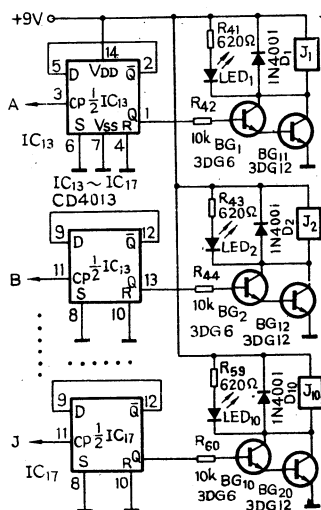


图 2

通道中心频率进行扫描。由于输入振荡信号未加到 LM567 上,其输出端⑧脚为高电平,则  $IC_{10-3}$  的⑩脚输出低电平,使得  $IC_{8-1} \sim IC_{8-4}$ ,  $IC_{9-1} \sim IC_{9-4}$ ,  $IC_{10-1}$ ,  $IC_{10-2}$  上的一个输入端分别为低电平,故 A~J 端均输出低电平,致使继电器  $J_1 \sim J_{10}$  都为释放状态。

当按下  $AN_1$  时,红外接收管 HG 接收到红外发射

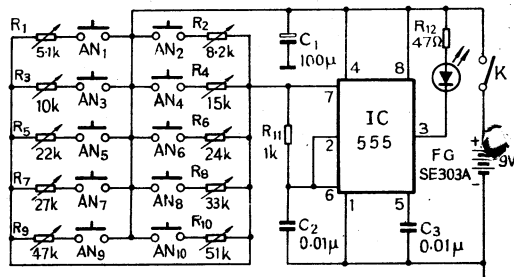


图 3

发射电路中, IC 使用 NE555 时基集成电路, 亦可用 5G1555、FX555、LM555 等。红外发射管 FG 使用 SE303A, 或 HG503 等。AN<sub>1</sub>~AN<sub>10</sub> 均用小型按键开关。

接收电路中, IC<sub>1</sub> 使用时基集成电路 NE555, 其它的 555 型号亦可。IC<sub>2</sub> 使用十进制计数器/脉冲分配器 CD4017 或 C187、MC4017 等, 其外引出脚排列和功用见图 4。IC<sub>3</sub> 为通用型运算放大器  $\mu$ A741, 亦可用 CF741、5G24 等。IC<sub>4</sub> 使用锁相环音频译码器 LM567,

发射电路中, IC 使用 NE555 时基集成电路, 亦可 G1555、FX555、LM555 等。红外发射管 FG 使用

接收电路中, IC<sub>1</sub> 使用时基集成电路 NE555, 其它的 555 型号亦可。IC<sub>2</sub> 使用十进制计数器/脉冲分配器 CD4017 或 C187、MC4017 等, 其外引出脚排列和功用见图 4。IC<sub>3</sub> 为通用型运算放大器  $\mu$ A741, 亦可用 CF741、5G24 等。IC<sub>4</sub> 使用锁相环音频译码器 LM567,



此外引出脚排列和功用见图 7。IC<sub>5</sub>~IC<sub>7</sub> 均使用四双向模拟数字传输开关 CD4066, 亦可用 5G811、CH4066、C544 等, 其外引出脚排列和功用见图 5。IC<sub>8</sub>~IC<sub>10</sub> 均使用四与非门集成电路 CD4011, 或用 5G801、CH4011 等。IC<sub>11</sub>、IC<sub>12</sub> 均用六非门集成电路 CD4069, 或 C033、5G806 等。IC<sub>13</sub>~IC<sub>17</sub> 均选用双 D 触发器 CD4013 或 CH4013、5G822 等, 其外引出脚排列和功用见图 6。BG<sub>1</sub>~BG<sub>10</sub> 均选用放大系数大于 60 的硅 NPN 型晶体管, 如 3DG6 等。BG<sub>11</sub>~BG<sub>20</sub> 均选用放大系数大于 80 的硅 NPN 型晶体管, 如 3DG12 等。HG 为红外线接收管 PH302, 若此管没有可用一般的光敏三极管代替。J<sub>1</sub>~J<sub>10</sub> 均使用直流工作电压 12 伏、线圈的直流电阻 450 欧、吸合电流 ≤ 20 毫安的 JQX-4 小型灵敏继电器。

整机安装完毕,经检查无误方可通电调试。先不接用电器,把一支 620 欧的电阻与一支普通发光二极管串联后接在 IC<sub>4</sub> 的 ⑧脚和电源正端之间,再把发射电路中的第一路遥控按键开关 AN<sub>1</sub> 用导线短接,然后分别接通接收电路和发射电路的电源,将红外发射管 FG 对准红外接收管 HG 的受光面,缓慢的转动可调电阻 R<sub>1</sub>,同时观察接在 IC<sub>4</sub> 的 ⑧脚上的发光二极管,当该发光二极管刚好点亮时,则停止调整 R<sub>1</sub>,此时第一路发射频率与 IC<sub>4</sub> 相应的第一通道中心频率相同。最后,取下按键开关上的短接导线,第一路遥控通道就算调整好了。同理,用相同的方法可调整好 2~10 遥控通道,使其 IC<sub>4</sub> 的 2~10 通道中心频率与发射电路的 2~10 通道发射频率相对应。发光二极管 LED<sub>1</sub>~LED<sub>10</sub> 分别作 1~10 路电器的通电工作指示。倘若接收机 1~10 通道的某个中心频率偏移相对应的发射频率的话,可适当调整 R<sub>20</sub>~R<sub>29</sub>与之相对应的电阻的阻值,使其每一路遥控通道均能可靠的对电器进行遥控。

# 初级程序员水平考试辅导

## 微型计算机操作系统使用问答

王路敬

一台计算机安装起来后,怎样使这套系统正常地运转起来呢?计算机的各组成部分又是怎样相互配合、协调一致有效地进行工作呢?这一切都是通过操作系统去控制的。操作系统是计算机所有硬件资源和软件资源的组织者和管理者,是对计算机系统进行统一管理、分配和调度的系统软件,是由许多具有控制和管理功能的子程序组成。

任何一个用户都是通过操作系统使用计算机的。它是人机交互的接口,是所有软件编制的基础,所有应用软件的使用方法及原理都与操作系统有着许多内在的联系。因此,了解操作系统的一般工作原理以及有关的基本概念,基本组成及使用方法对操作人员和软件工作人员都是十分必要的。随着 IBM PC 机及其兼容机的广泛普及,PCDOS、CCDOS 操作系统已成为一种最流行的单用户操作系统。作为使用微型计算机的初级程序员,迅速正确的解决使用操作系统中碰到的一些问题显然是极为重要的。下面就将大家在使用 PCDOS 和 CCDOS 过程中常见的而又具有普遍性的问题解答如下。对报考初级程序员的考生进行复习提供一些帮助。

### 1. PCDOS 的发展过程分为哪几阶段?

到目前为止,DOS 的发展过程可以分为两个阶段:

(1)第一个阶段是指 DOS1.00 及 DOS1.10 版时期。这些早期的版本是由微软公司(Microsoft)将所购置的 8 位 CP/M 操作系统予以改进而来。事实上,早期的 DOS 可以说是 8086/8088 CP/M 的扩充版。

(2)第二个阶段则始于 2.00 版的发表。这个版主要改革部分是将原来版本再加上结构上的扩充。主要增加的部分是提供 UNIX 的部分使用者界面。而其他结构上的重大改革是提供了硬盘驱动器、树形文件目录系统、输入/输出定向及管道功能、可增设的磁盘机、提供更方便操作系统与应用程序的文件输入/输出界面等。

PC-DOS2.10 版仅对系统内部作了修改,使能适用于 PC/JR 及 PC Portable(轻便型)的半高型软盘驱动器特性。没有强加任何新命令。长城 0520CH 就使用了该版本。

PC-DOS3.00 版本为 PC/AT 而发表的。它除了提供 PC/AT 的 1.2MB 的软盘驱动器功能外,外部较明显的改进是命令可以加上一个前列字串,用以描述命令所在的子目录。这一点改进在 3.00 及后续版本中体现得比较明显。在内部,文件分配表是(FAT)做了重

大的改变,文件分配表的每一项以 16 位二进制数来表示,而不是以 12 位二进制数来表示。所以一个磁盘中最多可以有 65536 簇,而不是以前的 4096 簇。簇的数量增加意味着在同样大小的磁盘,每一个簇可以有较小的空间,因此,磁盘空间的使用会更有效率。

PC-DOS3.10 实现了网络功能,它还提供了 SUBST 命令和 JOIN 命令。SUBST 命令可以将一个磁盘机字母代替任一路径。JOIN 命令,它可以把整个磁盘机处理成为其他装置的子目录。除此之外,PC-DOS3.10 改正了 PC-DOS3.00 的 BACKUP 命令。早期版本的 BACKUP 在处理到超过磁盘容量限制的文件时会失去它的磁盘计数,而新版的 BACKUP 则完全正常。

PC-DOS3.20 版是为轻便型 PC 机所发表的,现在已广泛的应用到长城 0520DH,长城 286 及其兼容机上。这种版本主要加强部分是提供对 3.5 英寸 720B 软盘驱动器的驱动功能,以及 IBM 的 Token-Ring 的网络功能。也有一般性的改进,如提供 REPLACE、XCOPY 等,它们是 COPY 命令的补充和加强。这种版本的其他特点是提供一个称为 DRIVER.SYS 的设备驱动程序。该程序的主要功能是提供外部的 3.5 英寸磁盘机驱动功能,也可以处理其他的磁盘机。增加了一些命令,扩充了一项新的输入/输出控制。

PC-DOS3.30 除了可以用于现有的 PC/XT,PC/AT 及长城系列微机以外还可以用于 IBM PS/2 微型机。在 PC-DOS3.30 中引进了一些新的概念,如码页的概念,也配备了一些新的设备驱动程序和外部命令程序,如 DISPLAY.SYS、APPEND 等。

DOS4.00 是最新版本,它包括许多新命令,还包括许多较以前各版本有所增强的命令。它的新功能支持大容量存储介质(硬盘分区可以超过 32 兆字节)。提供了全屏幕交互式安装实用程序 SELECT 以及由鼠标器或键盘驱动的全屏幕用户接口程序 Shell,使你能建立用户定义的菜单系统。4.00 版加强了对命令语法的检查,出错信息,而且作了更详尽的区分和说明。DOS4.00 中较大的文件分配表将加速对文件的访问。

可见,DOS 的版本不断变化,其功能不断增强。综上所述,除 PC-DOS3.10 版是为了网络的需要,其他版本变更的主要原因是因为磁盘的升级。

### 2. PCDOS 的各个版本之间的兼容性是什么含义?

目前社会上使用了 PCDOS 的各种版本,较多的是 PCDOS3.XX 版。这些不同的 PCDOS 版本之间基本上是兼容的。所谓兼容是指低版本上能操作的命令,在高版本上同样可以操作。版本号越高,功能越强。选用



PCDOS 的版本要根据机型,低档次的微机不要追求高版本的 PCDOS。

### 3. 组成 PCDOS 的四个模块各有哪些主要功能?

PC-DOS 由引导程序、隐含文件 IBMBIO.COM 基本输入输出管理程序、隐含文件 IBMDOS.COM 中断 21H 的系统功能调用和用户文件管理程序以及命令处理程序 COMMAND.COM 四部分组成。它们各自的功能是:

#### (一)引导程序

系统加电后,执行 ROM 区地址在 FFFF:0000 的启动程序,进行初始化,并自检。然后进入 ROM BIOS 的 INT 19H 检查驱动器插盘否。若插有 DOS 磁盘,则读入引导程序,否则再进入 ROM BIOS。引导程序放在内存地址 0000:7C00。

其引导过程是:

(1)检查 0 道 5 扇区的目录块,应有 IBMBIO.COM 和 IBMDOS.COM

(2)将 IBMBIO.COM 和 IBMDOS.COM 两个文件放在地址 0060:0000 处。

(3)转到 0060:0000 处执行 IBMBIO.COM

(4)以上没有,则为非系统盘显示错误,等待换盘。

#### (二)IBMBIO.COM

该程序依赖机器硬件,扩充了 ROM BIOS 与 IBMDOS.COM 接口,处理同全体外设的通讯。它的功能是:

第一,初始化功能,包括:

(1)建立磁盘参数表(INT-1EH)入口;

(2)初始化 RS232 端口和打印机口;

(3)建立 01H,03H,04H,1BH 类中断入口;

(4)建立通信区 0050 段中的标志单元;

(5)把 IBMDOS 从当前位置移到 BFH 段,要复盖一些 BIOS.COM 程序;

(6)确定磁驱动器数和 RAM 的大小;

(7)调 IBMDOS 初始化程序,建立用户区段;

(8)填写 25H,29H 类中断入口地址;

(9)把 COMMAND.COM 读到用户区段 0100 处;

(10)建立磁盘缓冲区(DAT)地址,偏移量为 80H;

(11)执行 COMMAND.COM 程序。

第二,接口功能,有 11 个设备驱动程序:

05H,08H,0AH,10H,11H,12H,13H,14H,16H,17H,19BH

#### (三)IBMDOS.COM

该程序不依赖机器硬件,与 IBMBIO.COM 和应用程序接口。它的功能是:

第一,初始化功能

(1)按照 IBMBIO.COM 的磁盘参数表 DPT,再建立新的磁盘参数表 DPT;

(2)每个驱动器建立一个文件分配表(FAT);

(3)在最后一张文件分配表末尾建立用户区段地址;

(4)检查 RAM 的大小;

(5)填写中断向量 0H,20H,22H,23H,24H,25H,26H,30H 的入口;

(6)在程序段建立前缀控制块 PSP。

第二,中断 21H 功能调用

共有 12 组 57 个功能块,调用时 AH 放功能块号。

#### (四)COMMAND.COM

COMMAND.COM 由暂驻内存、常驻内存和初始化部分组成。

第一,暂驻部分

在 COMMAND.COM 建立段前缀 PSP 后,由 IBMBIO 的初始化程序将 COMMAND.COM 的常驻部分调入 CS:0100,由此开始执行 COMMAND.COM 的初始化。暂驻部分含:命令处理程序本身、所有的内部命令处理程序和批命令处理程序。对 DOS2.1 还包括装入外部命令程序,执行外部命令程序。

第二,COMMAND.COM 的内存常驻部分的功能是:

(1)与 IBMDOS 用户区程序,COMMAND.COM 的暂存部分通讯;

(2)建立 INT22H 管理程序终止,INT23H 中断退出,INT24H 出错处理,INT27H 结束并保留程序;

(3)在用户程序终止后,检查暂存部分被用户程序覆盖否,若覆盖了则重新调入。

第三,初始化部分。这部分包含 AUTOEXEC.BAT 文件处理程序的设置程序。它决定被装入程序的段地址。

第四,COMMAND.COM 还产生 DOS 提示符(如 A>或 C>),读来自键盘(或批命令文件)的命令并执行之。对于外部命令,它建立一个命令行并发出 EXEC 功能调用,以装入并把控制传送给这个外部命令处理程序。

#### 4. 启动 PCDOS 的含义是什么?

组成 PCDOS 的文件都装于磁盘之中,只有将磁盘的文件按一定的规则装入内存,才能使 PCDOS 操作系统激活,从而把系统的硬件资源和软件资源管理起来。所谓 PCDOS 的启动就是把系统盘上基本文件:IBMBIO.COM 基本输入输出系统、IBMDOS.COM DOS 核心、COMMAND.COM 命令处理程序装入内存的过程。

启动 PCDOS 有两种方式:一种是冷启动,即加电开机的启动方式。第二种是热启动,这是系统主机已经加电,或你想摆脱某种困境而要回到 PCDOS 状态的启动方式。建议你最好从硬盘系统启动 PCDOS(或 CCDOS),这样做操作简单,启动速度快,而且可减少计算机病毒感染的机会。

#### 5. PCDOS 启动的过程是怎样进行的?

PCDOS 的启动过程如下:

(1)当系统加电或热启动后,程序从 FFFF:0000H 开始执行,进入自检程序和 ROM 引导装入程序。

(2)ROM 引导装入程序,从磁盘的第 1 扇区(引导扇区)读入磁盘引导装入程序到 0000:007CH,然后将控制权转交给磁盘引导装入程序。

(3) 磁盘引导装入程序检查盘上是否有 DOS 系统。检查过程为: 读入根目录的第 1 扇区, 然后检查其中前两个文件是否为 IBMBIO.COM 和 IBMDOS.COM, 若盘上无这两个文件, 系统将提示用户插入另一磁盘, 然后按下任一健继续。若发现了这两个文件, 磁盘引导装入程序便将 IBMBIO.COM 读入内存, 并将控制转向 IBMBIO.COM 的初始化入口。

(4) IBMBIO.COM 由两个模块组成: 第 1 个模块是 BIOS, 它由一组设备驱动程序组成; 第 2 个模块是 SYSINIT。

BIOS 模块的初始化部分将 IBMDOS.COM 加载到内存。

(5) SYSINIT 由 BIOS 的初始化代码调用。该程序确定连续的内存空间的大小, 并将自身重新装入内存高端。然后, SYSINIT 将 DOS 核心 (IBMDOS.COM) 从初始化位置重新加载到最后位置, 并将 IBMBIO.COM 中可覆盖的初始化程序和内存低端的 SYSINIT 覆盖掉。

(6) 随后, SYSINIT 调用 IBMDOS.COM 的初始化程序。作为系统初始化工作的一部分, DOS 核心检查由驻留的块设备驱动程序返回的磁盘参量, 确定系统所使用的最大扇区的大小, 建立一些磁盘参数块 (DPB) 并设立磁盘缓冲区。

执行完 DOS 核心的初始化程序后, SYSINIT 便可调用 DOS 的文件服务程序 (INT21H) 去打开 CONFIG.SYS 文件, 用户可在 CONFIG.SYS 中指定新加的设备驱动程序, 确定磁盘缓冲区的数目, 最多可同时打开的文件数目以及定义命令处理程序的文件名等。

(7) 加载完所有可安装的设备驱动程序后, SYSINIT 关闭所有的文件指针, 重新打开控制台 (CON)、打印机 (PRN) 以及辅助设备 (AUX)。此时允许用户安装的字符设备驱动程序覆盖 BIOS 为标准设备驻留的设备驱动程序。

最后, SYSINIT 调用 DOS 的 EXEC 功能将命令处理程序 (COMMAND.COM) 加载到内存, 然后将控制转交给命令处理程序。

命令处理程序显示 DOS 提示符, 等待用户输入命令。此时 DOS 进入正常工作状态, SYSINIT 消失。

6. 为什么有时热启动不起作用? 此时应采取什么办法?

主要原因有两个: 其一, 突然改变了键盘中断处理程序的入口地址, 使中断程序不能正确地唤醒。由 PC DOS 中断向量表可知, 地址 0000:0036~0000:0039 这 4 个字节是 PC DOS 的 9 类中断入口地址, 它对应的是一个键盘中断处理程序。当由于某种原因改变了该类中断入口地址时, 9 类中断程序不能正确执行, 因而热启动也不能实现。其二, 由于程序停留在关中断状态。当一个程序由于某种原因而突然停止运行时, 如果程序正好停止在 CLI (关中断) 指令处而使系统处于禁止中断状态, 那么一切可屏蔽的中断都被挂起, 9 类中断是一个可屏蔽中断, 因此也不再起作用了。

当遇到上述两种情况时, 要重新加电冷启动系统。

7. 在什么情况下需要启动 PC DOS (或 CCDOS)? 一般在下述情况下需要启动 DOS?

(1) 系统从休息状态转入到工作状态。

(2) 因软件故障或操作不当而使系统无法继续运行, 出现系统“死锁”(或称“死机”)。

(3) 当改变系统结构设备而必须对系统结构设备文件 CONFIG.SYS 进行修改或重新建立时。

8. 在 PC DOS (或 CCDOS) 下当前盘与启动盘有什么关系? 系统启动后能改变启动盘吗? 系统在什么情况下才会用到启动盘?

操作系统启动后, 在未改变当前行定义的情况下, 当前盘和启动盘是指同一个磁盘驱动器。以后当前盘可以任意改变, 而启动盘不可改变。系统在“COMMAND.COM”的暂驻部分被覆盖时 (即软件的数量过大), 才会用到启动盘, 以便把覆盖部分的代码读入内存。

例如长城 0520 或 IBM PC/XT 的启动盘是 A 或 C, 当前盘可为 A、C 也可以是 B。

9. 长城 0520CH 系统启动后, 为什么有时显示的日期和时间为标准时间, 而有时则不是?

长城 0520CH 主机内有一个可在长时间切断电源后仍能保持准确走时的时钟, 称之实时时钟。只有使用了外部命令 REALTIME 后, 才能将实时时钟的内容传送到系统时钟, 从而得到标准时间。系统时钟是操作系统及各种应用软件所使用的时钟。如果每次启动后希望使用正确时钟, 就应使用 REALTIME 命令进行传送, 或在自动执行批处理文件中增加 REALTIME/C 这一命令。

10. 什么是批处理文件? 文件的扩展名为 .BAT 的 ASCII 码文件称之为批处理文件。

对于经常要执行的一系列命令, 可以把这些命令语句象编程序一样写入到一个批处理文件中。然后此文件可以作为外部命令被调用, 这时系统将自动地依次执行该文件中的全部命令语句。

对于已存在磁盘上的批处理文件, 可以用下列方式调用:

>批处理文件名

这时, 系统将自动地依次执行当前盘上的命令语句。系统在执行批处理文件时, 并不是把所含的命令文件全部装入内存而是读入一条并执行后再读入下一条执行, 这样一直到最后一条命令语句执行完为止。

批处理文件可用 COPY CON: (文件名).BAT 或用 EDLIN (文件名).BAT 命令来建立。

例如, 我们希望系统连续执行如下操作:

先把 A 盘的所有 .BAS 文件复制到 B 盘, 然后显示 A 盘的所有 .BAS 文件目录, 再显示 B 盘的所有 .BAS 文件目录, 最后打印出批处理文件自身的内容。

建立如下的批处理文件可达到上述要求:

A>COPY CON:WW.BAT

COPY A:\*.BAS B:

```
DIR A: * . BAS
DIR B: * . BAS
TYPE WW . BAT > PRN
AZ
```

AZ 即 <CTRL>+Z 结束 COPY CON; 命令并将 WW . BAT 文件存盘。按下述方式便可执行 WW . BAT 文件:

```
A>WW
```

11. 在批处理文件中可以使用哪些形式参数?为什么要使用?

PCDOS 系统规定,用户可使用 10 个形式参数。它们是 %0, %1, %2, …… %9。其中 %0 表示批处理文件自身。值得注意的是“%”与数字之间不能有空格。调用时如果实际参数个数少于形式参数,则会显示出“Syntaxerror”的出错信息;如果实际参数个数多于形式参数,系统将不理睬多余的实际参数。

在批处理文件中使用时形式参数目的是为了增加批处理文件的灵活性。

例如批处理文件 BATN . BAT 内容如下:

```
A>TYPE BATN . BAT
```

```
DIR A;
DISKCOPY A: B;
DISKCOMP A: B;
DIR B;
```

在该批处理文件内的盘符“A:”和“B:”用两个形式参数来代替,即:

```
A>COPY CON;BATN . BAT
```

```
TYPE %0 . BAT ;显示该批处理文件内的
                全部内容
```

```
DIR %1 ;显示由第一个参数所定义的磁盘
                的目录
```

```
DISKCOPY %1 %2
```

```
DISKCOMP %1 %2
```

```
DIR %2 ;显示由第二个参数所定义的磁
                盘目录
```

```
AZ
```

在批处理文件中不引入形式参数,由于各条命令的参数都是固定的,故每次运行批处理文件只能执行固定的操作。引入形式参数后,就可以在调用批处理文件时再决定命令中的参数,给批处理文件赋予了一定的灵活性。

12. 在批处理文件中可包含哪些内容?

可以包含 PCDOS 内部命令、外部命令、可执行的程序、形式参数以及专用于批处理文件的子命令。这些子命令是专用于批作业的内部命令和语句的标号,以实现稍微复杂一些的作业结构。

这些子命令主要有:

(1)ECHO—禁止或允许屏幕的命令显示语;

命令格式:

```
ECHO
```

或: ECHO ON/OFF

或: ECHO <显示信息>

(2)REM—注释语句

命令格式:

```
REM <注释语>
```

(3)PAUSE—暂停语句

命令格式:

```
PAUSE [<注释语>]
```

(4)GOTO—转向语句

命令格式:

```
GOTO <语句标号>
```

(5)IF—条件判断语句

命令格式:

```
IF [<NOT>]<条件><DOS 命令>
```

(6)FOR—重复执行语句

命令格式:

```
FOR %%<变量名>IN <文件名集> DO <DOS 命
令>
```

(7)SHIFT—左移参数语句

命令格式:

```
SHIFT
```

上述 7 个批处理子命令虽然也可以在操作系统的提示符下作为命令输入,但这样执行不是没有实际意义,就是不引起真正的执行。一般与 PCDOS 的命令,可执行程序结合起来组成一个批处理作业,使该处理文件更具有人机交互功能和友好性。

例如 CCDOS2.13G 中文操作系统的自动执行批处理文件 AUTOEXEC . BAT 的内容如下:

```
ECHO OFF
CLS
CD\213
MENUFF
ECHO                PLEASE WAIT
IF ERRORLEVEL 52 GOTO E
IF ERRORLEVEL 51 GOTO C
IF ERRORLEVEL 50 GOTO B
IF ERRORLEVEL 49 GOTO A
;E
FILE1 2
GOTO D
;A
FILE0 82
GOTO D
;C
IF NOT EXIST D:\HZK16 COPY HZK16 D;
FILE3 D2
;D
CCCC
CC11
PRT
FILE16B
FILE24 1SFHK
FILE40 1FHK
FILE49 1
```

```
ZF24 3
YX1
:B
PATH C:\;C:\213
KEY
CD\
```

13. 为什么说 AUTOEXEC · BAT 是 CCDOS 启动的必备文件?

CCDOS 是在 PC DOS 的基础上开发而成的中文操作系统。它既保留了 PC DOS 原有的全部功能,又增加了有关汉字方面的处理能力。作为一个汉字操作系统,必须在系统启动后调用一些处理汉字的系统软件实现汉字的输入、显示和打印。如果不使用 AUTOEXEC · BAT 文件,就要在每次系统启动后手工调用,这样会使不了解情况的用户无法进入中文操作系统,建立这个文件后,每当用户启动 PC DOS 时,命令处理程序 COMMAND · COM 就在启动盘的根目录中搜索 AUTOEXEC · BAT,并在找到之后立即执行,从而使系统自动处在中文操作系统下。

例如 IBM PC/XT 机使用的 CCDOS2.0/2.10,在该系统下 AUTOEXEC · BAT 一般由下列基本语句组成:

```
ECHO OFF
CLS
FILE1
CCCC
ALL24P
```

其中 FILE1,CCCC 分别为 CCDOS2.0/2.1 的第 1 个和第 2 个启动文件是必不少的。ALL24P 是 3070 打印机打印汉字的驱动程序,其他型号的打印机可选取相应打印驱动程序。

AUTOEXEC · BAT 是文本型文件可由用户建立和修改。在使用中需要注意的是当 AUTOEXEC · BAT 的内容一旦进行修改,需要重新启动系统。

14. DOS 为用户提供哪些实用功能?

DOS 为用户提供了大量的实用功能,包括设置一组中断和各种功能调用,其中有键盘输入程序,终端和打印机输出程序,形成文件控制块程序,内存管理程序,日期和时间程序以及各种磁盘、目录、文件的控制管理程序。作为文件管理功能,DOS 提供了两种类型的功能调用,它们是:

(1) 文件控制块 (FCB) 功能调用。

(2) 扩充控制功能调用。

15. DOS 在磁盘上分配分哪些区域? 它们的含义是什么?

由 DOS 格式化的软盘或者磁盘,其扇区的大小均为 512 字节。DOS 将磁盘划为如下几个区:

(1) 引导记录区,占 1 个扇区。它位于软盘的 0 面,0 磁道,1 扇区。如果试图用驱动器 A 中的非系统盘启动系统,它被用来产生出错误信息。对硬盘来说,引导记录位于 DOS 部分和第 1 个扇区。

(2) 文件分配表 FAT 区,可变长。

FAT 用来为文件分配空间,每次一组。对磁盘上的每一个 FAT 由 12 位 (1.5 字节) 的表项或者 16 位 (2 个字节) 的表项组成。凡是少于 20740 个扇区的软盘或硬盘都用 12 位 FAT;多于 20740 个扇区的,用 16 位 FAT。FAT 总是位于引导记录接着的扇区。如果 FAT 大于一个扇区,则占用下面连续的扇区号。

(3) 备份文件分配表 FAT 区,可变长。

(4) 根目录区,可变长。

紧挨着文件分配表 FAT 存放。

FORMAT 命令,在磁盘上建立根目录。不同种类的软盘片,根目录的个数不同。

(5) 数据区

文件的空分配 (在数据区) 只有在需要的时候才做。每次分配一个簇。一个簇总是一个或多个连续的扇区号,一个文件所有的簇在 FAT 中链在一起。数据区排在备份文件分配表区之后。

16. DOS 怎样使用文件目录表和文件分配表 FAT?

DOS 在分配磁盘空间时是以簇为最小单位进行的,而不是以扇区为单位。对于不同形式的磁盘,每簇所拥有的扇区数是不同的。对一个长度仅为一个字节的文件来说,在双面软盘上要占用 1024 字节的空间,而在硬盘上就要占用 4096 字节的空间。读文件时,DOS 首先在目录中拉到该文件的目录项,根据目录项中第 25~27 字节的起始簇号值,求出文件的起始相对扇区号和在 FAT 的入口。起如扇区号与起始簇号之间的关如下 (十进制):

| 磁盘形式    | 起始扇区号 S 与起始簇号 C |
|---------|-----------------|
| 单面 8 扇区 | $S = C + 5$     |
| 双面 8 扇区 | $S = 2C + 6$    |
| 单面 9 扇区 | $S = C + 7$     |
| 双面 9 扇区 | $S = 2C + 8$    |
| 10M 硬盘  | $S = 8C + 33$   |

FAT 表中的入口地址即为字节的顺序号 (从 0 开始编号)。计算方法:

(1) 将簇号乘以 1.5

(2) 若所得结果为一整数,比如说是 6,则 FAT 中的第 6 个字节的 8 位二进制数作为  $b_7 \sim b_0$  位,将相邻的下一个字节的低 4 位作为  $b_{11} \sim b_8$ ,组成一个 12 位二进制数,这个 12 位的二进制数即表示对应簇的状态。若所得结果是一小数,比如是 7.5,则 FAT 中的第 7 个字节的高 4 位作为  $b_3 \sim b_0$  位。将下邻的一个字节作为  $b_{11} \sim b_4$ ,组成一个 12 位的二进制数。

写文件时过程正好相反,DOS 首先在目录中检索是否有同名文件,若无则将该文件名写入第一个找到的作过删除标记的目录中,否则开辟一新的未用目录项。然后,顺序检索 FAT 中的每个表项。找到第一个值为 000H 的表项后,将文件写入该表项对应的簇中,并将其簇号写入一个表项中。如果文件写完,则该表项位置为 FFFH。

一个文件可以是放在互相衔接的几个连续簇中,也可能是断续放在几个不相邻的簇中,中间隔着其他文件占用的簇或空簇。但不论哪种情形,一个文件所占用的簇,其簇号一定是从小到大顺序递增的,除非该文件不是采用正常的拷贝命令和编辑手段建立的。当用 COPY 命令将一些文件拷贝到一张空盘上时,所有的文件一定是放在互相衔接的连续簇中的。一旦对文件进行了多次增删和修改后,这种连续的情形就不存在了。有时在某个文件中间还会出现释放的空簇,这些情况都给恢复文件带来极大困难,甚至使恢复成为不可能的。

#### 17. 文件分配表 FAT 向用户提供哪些信息?

FAT 是磁盘上一个相当重要的区域,向用户提供以下 4 个方面的信息:

- (1) 磁盘性质;
- (2) 扇区封锁信息表;
- (3) 扇区使用情况;
- (4) 指示一个文件的后续簇。

FAT 中每一个表项的值由一个 12 位或 16 位二进制数组成。第 1~2 字节总是 FFH,第 0 字节(首字节)表示磁盘的性质,其意义如下:

- FEH—单面每磁道 8 扇区软盘。
- FFH—双面每磁道 8 扇区软盘。
- FCH—单面每磁道 9 扇区软盘。
- FDH—双面每磁道 9 扇区软盘。
- F8H—硬盘

从第 3 字节开始的第 2 表项表示磁盘上数据区簇的情况。数据区一定是从 002H 簇开始的。第 3~5 字节为第 2~3 表项,表示磁盘上 002H 簇和 003 簇的情况。表项号,簇号,磁盘上的簇域是一一对应的。这里要注意表项的值在字节中的位置。例如第 4 字节一半属于第 2 表项,而另一半属于第 3 表项。具体分配时其高 4 位应属于第 3 表项的低 4 位,而其低 4 位则是第 2 表项的高 4 位。

磁盘在格式化时如发现损坏的扇区便在对应的扇区表项中写入 FF7H,表示该扇区所在簇不能使用。当表项的值为 000H 时表示对应的簇是空的,可以使用。FF8H~FFFH 表示该簇为文件中最后一簇。FF0H~FF6H 表示保留的簇,其他在表项中出现的指示文件中的一个簇簇号,也是该文件在 FAT 中的下一个入口。

#### 18. 文件目录表向用户能提供哪些信息?

文件目录表是存放文件目录的地方,每个文件的目录由 32 个字节组成,包含了有关该文件的重要信息,各字节的意义如下:

##### (1) 第 0~7 字节:

文件名,当第 0 字节的值为:

00H——表示该目录项从未使用过。DOS 在有关的操作中遇到 00H 后即停止向下继续检索。

E5H——表示该文件已被删除,DOS 跳过该目录项继续向下检索。

2EH——表示目录表为一子目录表。

除此之外出现的其他值为文件名第一个字符的 ASCII 代码。

##### (2) 第 8~10 字节:

文件的扩展名。

##### (3) 第 11 字节:

表示文件的属性,其含意如下:

01H—表示只读文件,拒绝 COPY、ERASE 命令。

02H—表示隐含文件,拒绝 COPY、ERASE、DIR 命令。

04H—表示系统文件,拒绝 COPY、ERASE、DIR 命令。

08H—表示该目录项前 11 个字符为磁盘卷标,在 FAT 中没有入口,只有存在于根目录中。

10H—表示该文件名为一子目录的父名。

20H—表示档案位。在建立或修改文件时,该值加到文件属性中表示关闭,对硬盘上文件使用 BACKUP 命令后,该位被打开,从文件属性中减去该值。归档位主要和 BACKUP 命令中的 M 开关及 RESTOR 命令中的 P 开关有关。

##### (4) 第 12~21 字节:

保留未用。

##### (5) 第 22~23 字节:

文件生成或最后一次更新的时间。

##### (6) 第 24~25 字节:

文件生成或最后一次更新的日期。

##### (7) 第 26~27 字节:

文件的起始簇号,也是指向 FAT 的入口地址。第 26 字节是低位字节,第 27 字节是高位字节。当第 0 字节为 2EH 时表示子目录所在的簇号。当第 0 字节与第 1 字节均为 2EH 时,表示该子目录表的父目录所在的簇号。

##### (8) 第 28~31 字节:

表示文件长度,以字节为单位。第 28 字节为低位字节,第 31 位字节为高位字节。

#### 19. CCDOS 系统盘上的 DONFIG·SYS 文件有何用途?

CONFIG·SYS 为系统结构设置文件。在 CCDOS 启动过程中,CCDOS 自动在系统盘中检索这个文件。如该文件存在,则将该文件中各语句的内容对系统进行一些初始化设备,如果不存在,CCDOS 就按约定值进行初始化设备。

以 CCDOS 2.0/2.10 为例,CONFIG·SYS 文件包含如下 7 条系统设备命令完成以下功能:

##### (1) BREAK=ON/OFF(约定状态:OFF)

该语句用于对中止键(CTRL)+(BREAK)进行开(ON)关(OFF)设置。

当 BREAK=ON 时,无论系统在执行什么操作,都可以用(CTRL)+(BREAK)键中止系统的操作。

当 BREAK=OFF 时,只有当系统正在执行外设操作(屏幕、键盘、打印机和异步通讯适配器操作)时,才能用(CTRL)+(BREAK)键中止系统的操作。当系统执



行其它操作时,〈CTRL〉+〈BREAK〉键无效。

也可以用内部命令 BREAK 设置〈CTRL〉+〈BREAK〉的状态。但需要注意的是,在某些应用软件中(例如 dBASE III),〈CTRL〉+〈BREAK〉键是无效的。因此,BREAK 的设置对这类软件是无效的。

#### (2)BUFFERS=n

其中 n 可在 1~99 之间选择,约定值为 2,表示设置缓冲区的数量。

缓冲区是磁盘文件管理的重要设置。当 DOS 读磁盘时,先将磁盘上的内容读入缓冲区中,然后再对其中的信息进行处理。在 DOS 进行反复读盘操作时,它先去缓冲区中检查要读的内容是否已在缓冲区中,若在,则勿须再执行读盘操作,从而节省了大量的读盘时间。

当 DOS 写磁盘时,要在缓冲区中存放写的信息,DOS 将缓冲区中的数据组织好后将其写入磁盘。

每个缓冲区要占用 528 个字节的内存。在某些应用中,多几个缓冲区可以提高程序的执行速度,但要以多开销内存为代价。并不是说缓冲区越多,速度就一定会更快。对于要求执行大量随机磁盘存取的应用程序(例如基本的数据库应用程序)缓冲区应多一些(例如在 10~20 之间可以取得较好的效益);而对于执行大量顺序磁盘存取的应用程序(例如对整个文件读写),系统默认的两个缓冲区是足够的了,没有必要再去设置 BUFFERS 的数量。

#### (3)FILES=n

FILES 用于规定 DOS 同时可打开的文件的数量,其中 n 可在 8~99 之间选择。系统的默认值是 8。但这 8 个 FILES 并不是都交给用户使用,用户只能使用其中的 3 个,另外 5 个 DOS 早已分配给标准输入、标准输出、标准错误输出、辅助设备和标准打印机了。用户如果需要同时打开多个文件,必须对 FILES 重新设置。

FILES 也是 DOS 进行文件管理的依据。DOS 每打开一个文件,就要在内存中为该文件建立一个文件控制块(简称 FCB),FILES 实质上规定了 DOS 所能建立的 FCB 的个数。

每个 FCB 在内存中要占用 39 个字节。一区 39 个字节对几百 K 的内存来说是微不足道的,却赢得了可打开多个文件的方便,所以用户多设置几个 FILES 是明智的作法。

#### (4)DEVICE=〈文件名全称〉

DOS 规定了 5 种标准设备文件,它们是:标准输入设备;标准输出设备;出错信息输出设备;异步通讯控制器及打印机。如果用户想要扩充系统的设备,就要用 DEVICE 语句来定义扩充设备的驱动程序文件的名字。这样,在 DOS 启动时,系统就能把这个文件装入内存作为 DOS 基本设备的扩充。例如,在 0520CH 上使用的 GWBIOS 3.00 在支持高级语言作图功能时,要把 DEVICE 设置为:

DEVICE=GRD·SYS

其中 GRD·SYS 是一个系统图形驱动文件,该文件作为系统的虚拟图形设备,当系统启动时,将自动把

GRD·SYS 作为系统设备驱动文件装入内存。这样,用户在使用高级语言中的打开(OPEN)设备文件,写(WRITE)数据到该设备等语句时就可以进行画图操作了。

在一个 CONFIG·SYS 文件中可以使用多条 DEVICE 语句来定义多个扩充设备。DEVICE 通常是按应用软件的要求而设置的。

#### (5)SHELL=〈文件名全称〉

此语句可以用用户自己开发的控制台命令处理程序代替系统所规定使用的 COMMAND·COM 文件。用户在使用这条语句时一定要谨慎从事,除非确有必要使用另行开发的控制命令处理程序,一般是不使用此语句的。

#### (6)AVAILDEV=F

AVAILDEV 命令用于关闭字符设备名检查,其缺省值是 T(开)。当字符设备名检查开关被打开(T)时,每当进行指针型文件操作时,要检查文件名是否为字符设备名,否则(F),则不对其进行检查。

#### (7)SWITCHAR=〈开关引导符〉

SWITCHAR 命令用于设置命令行参数开关引导符,其缺省值是“/”。开关引导符是 COMMAND·COM 所特需的。

综上所述,CONFIG·SYS 文件是十分有用的。特别是在某些应用领域,例如 dBASE III,它要求 FILES 和 BUFFERS 必须大于约定值,如若不设置就不能正常使用 dBASE III。而 CONFIG·SYS 是在 DOS 启动时使用的,用户容易忽略,所以要提醒用户注意。

20. PCDOS 的内部命令和外部命令是按什么划分的? 使用外部命令时注意什么问题?

DOS 的命令有两类:一类是 DOS 的内部命令,另一类是 DOS 的外部命令。内部命令与外部命令是按此命令是否常驻内存中来划分的。DOS 任何命令的执行即执行一段程序,DOS 的内部命令的程序已随 DOS 的启动而装入内存,其命令动词为系统保留词,如 DIR, COPY, TYPE, ERASE 等等均为内部命令。其外部命令的程序存于命令文件中,调用时装入内存,执行完一般所占用的内存释放,其命令动词为命令文件的文件名。使用外部命令时应注意:

(1)外部命令的执行依赖于命令文件的存在,即执行任一个外部命令,系统首先要从磁盘读取相应的文件,如此命令不存在,则命令不可执行,因此在当前盘上应当存在要执行的命令文件。

(2)在调用外部命令时,命令文件的扩展名是不必给出的。系统约定的扩展名为:

·COM

·EXE

·BAT

三种类型的文件的文件名 DOS 可直接调用。

(3)当调用外部命令时,若系统显示“Bad command or file name”错误信息,这时可以从两个方面检查:一是命令拼错了没有;一是当前盘是否存在该命令文件。

## 21. ANSI · SYS 程序有何功能？怎样设置？

PCDOS 装载的标准设备驱动器支持标准输入、标准输出、标准打印机、软盘以及硬盘设备。PCDOS 支持这些设备不需要用户用任何命令指定，如果要扩充输入设备、输出设备的控制特性，就需要把编写的有关设备的驱动程序，在启动 PCDOS 时把它装入，而且是通过在系统结构设置文件 CONFIG · SYS 中加入一个 DEVICE = (命令) 这种形式指定。其中 DEVICE = ANSI · SYS 就是一种指定形式。

ANSI · SYS 是 PCDOS 提供的一个系统文件，在此文件的控制下可以实现屏幕和键盘的功能扩充，使 PCDOS 用扩展功能取代标准输入和标准输出。中文操作系统 CCDOS 2.0/2.1 是在 PCDOS 2.0/2.1 基础上扩充汉字输入和输出功能后形成的，键盘和屏幕输入和输出功能的扩充就是利用了 ANSI · SYS 的控制特性。如果在 CONFIG · SYS 文件中不加入 DEVICE = ANSI · SYS 语句，IBM-PC/XT，长城 0520A 及其兼容机，使用 CCDOS 2.0/2.1，在显示 11 行汉字状态下，PCDOS 的清屏命令 CLS 执行后光标不显示，屏幕上什么信息也没有，给使用带来不便。相反如果在 CONFIG · SYS 文件中加入语句 DEVICE = ANSI · SYS 就很好地解决了清屏命令执行后屏幕无光标显示的毛病。

## 22. IBMBIO · COM、IBMDOS · COM 和 COMMAND · COM 版本不同时，会发生什么现象？怎样解决？

随着 DOS 功能的不断增加，其操作系统的大小也不断增加。在不同版本下，随版本号提高，其文件的长度也明显加大。对于 PCDOS 的基本文件：IBMBIO · COM、IBMDOS · COM、COMMAND · COM 的长度，以及它们在磁盘上占用的空间，系统程序占用区，暂态区，所占内存空间的情况在不同版本下其文件长度不同。

当 DOS 中两个隐含文件 IBMBIO · COM、IBMDOS · COM 和外部文件 COMMAND · COM 不是同一版本时，往往出现引导失败，或屏幕提示 DOS 版本不对，或自举后不能进入硬盘，运行应用软件出错等现象。产生这种问题的原因绝大多数是由于软盘之间，或者软盘与硬盘之间全盘拷贝造成的。例如，把整个软盘拷贝到硬盘，这时若盘上有 COMMAND · COM 文件也将它拷贝到了硬盘，而与它同版本的两个隐含文件 IBMBIO · COM 和 IBMDOS · COM 却没有拷贝到硬盘，造成上述现象。

每当出现这些现象时，不要急忙使用 FORMAT 命令格式化硬盘，或用检测盘测试微机，更不要把微机送出去修理。否则，既浪费人力、物力、财力、机时，又容易丢失大量数据。

在应用中可以采取一定措施来防止这种问题的发生。例如，所有应用程序只能拷贝到各自子目录下，或少或不用全盘拷贝命令。如果万一发生这种问题，可以针对不同情况，采取不同的解决办法。例如，当能自举时，可以将自举后屏幕显示的 COMMAND 中的版本号与用命令 VER 显示的两个隐含文件中的版本号相比较，看是否匹配。因为 DOS 的版本不同 IBMBIO ·

COM、IBMDOS · COM 和 COMMAND · COM 文件长度不同，通过命令文件的大小可以确定版本号是否匹配。当不能自举或屏幕提示 DOS 版本不对时，可以用比较高版本 DOS 从软盘自举，然后查看产生问题的原因。

## 23. 使用 FORMAT 命令时应注意什么问题？

磁盘格式化命令 FORMAT 是 DOS 的一个重要外部命令，执行该命令时它删除磁盘上所有的数据。因此在决定对一个磁盘，尤其是硬盘进行格式化时应当格外小心，在格式化之前需要备份又能够备份的文件最好备份下来，以便格式化后恢复。如果不能肯定哪个驱动器是默认的，在没有驱动器字母标号的情况下，就不要输入 FORMAT，否则就有可能把默认盘上的所有数据删除，造成损失。为保险起见，在 FORMAT 命令行中明确指出要格式化的驱动器的字母标号。

## 24. 为什么 FORMAT 命令不能消除硬盘上的大麻病毒？

因为大麻病毒存放在硬盘的主引导扇区，不属于 PCDOS 分区，所以与机器使用的 DOS 无关，故对硬盘格式化不能改变主引导扇区，寄生在主引导扇区的大麻病毒不能被消除。

利用下面两种方法可以人工消除硬盘感染的大麻病毒。

### 方法之一：

对硬盘进行初级格式化。如果系统带有初级格式化程序，可以在操作系统下直接执行；若没有初级格式化程序可用 DEBUG 程序调用 ROMBIOS 中的初始化程序并执行之。缺点是破坏硬盘的全部数据。

### 操作如下：

A>DEBUG

—G=C800:0005

该操完成后，即把硬盘主引导扇区的大麻病毒清除。然后进行硬盘分区，建立硬盘系统。

### 方法之二：

利用 DEBUG 程序进行如下操作：

A>DEBUG

—R IP ; 设置 IP 指针

:100

—A 100 ; 开始汇编

××××:0100 MOV DX,0080

MOV CX,0007

MOV BX,0200

MOV AX,0201

INT 13H ;读硬盘 0 柱面 0 磁头  
7 扇区到内存 200 处读  
原硬盘主引导程序

MOV DX,0080

MOV CX,0001

MOV BX,0200

MOV AX,0301

××××:011A INT 13H ;写硬盘 0 柱面 0 磁头  
1 扇区恢复硬盘主  
引导程序

25. FORMAT 和 SYS 两个命令在使用中有何区别?

FORMAT 命令选用 S 参数和 SYS 命令都可以复制系统文件,但在使用中有所区别表现在:

(1)若想在已保存有数据的盘上再复制系统文件,应该用 SYS 命令,因为格式化时会删除磁盘上原有数据。

(2)使用 FORMAT /S 可以按如下顺序将 IBMBIO · COM、IBMDOS · COM 和 COMMAND · COM 复制系统文件,而使用 SYS 命令只能复制前两个隐含的系统文件,不能传送 COMMAND · COM。

(3)使用 SYS 命令传送系统文件,要求格式化时最好选用 /B 参数,这样可以存入任一 DOS 版本的系统文件,否则只能存入指定的版本,还有一点要注意的是若格式化时选用了 /V 参数,则不能用 SYS 命令传送系统文件,此时目标盘上没有为 DOS 系统保留存储空间。

26. 使用 BACKUP 和 RESTORE 命令应当注意什么?

BACKUP 和 RESTORE 这两个命令必须配合使用。具体来说,用 BACKUP 命令把硬盘根目录下的数据文件备份到软盘,用 RESTORE 命令还原备份文件时要还原到硬盘的根目录下;若用 BACKUP 命令将硬盘子目录下的文件备份到软盘,用 RESTORE 还原备份文件时要还原到硬盘相应的子目录下,否则不会成功。

27. COPY 和 DISKCOPY 两个命令使用上的不同点有哪些?

COPY 和 DISKCOPY 都是复制类命令,都能完成指定磁盘或文件的拷贝,它们的不同点是:

(1)工作方式不同。COPY 命令是以文件对文件的形式进行复制,而 DISKCOPY 则以磁道对磁道的方式进行复制。由于增、删和修改,一个文件常处于磁盘上不连续的扇区。读、写磁盘上不连续的文件要多花费时间。这时可利用 COPY 命令工作方式的特点。将原盘中的文件一个个复制到新盘上去,这样新盘上的文件就都处于连续的扇区中,即通过拷贝可以整理文件的存放空间。

(2)COPY 命令可复制全盘文件,也可以复制某个或某些指定的文件,而 DISKCOPY 只有全盘复制的功能。

(3)COPY 命令既可以在软盘和软盘之间,也可以在软盘和硬盘之间复制文件;DISKCOPY 只适用于软盘之间的复制。

(4)使用 COPY 命令复制前,目标盘必须已格式化,而用 DISKCOPY 若目录盘未格式化成与源盘一致的格式,它可在复制操作时格式化该盘。

(5)COPY 命令不能复制隐含文件(列目录时不显示的文件),故复制 DOS 系统盘(有两个隐含文件)时

应选用 DISKCOPY 命令。

值得注意的是: COPY 命令只能从当前目录或指定目录中复制文件,若要复制包括子目录的所有文件,就要用 BACKUP 命令。该命令既可以将硬盘上的指定文件复制到软盘作为备份存档,也能传递长度大于一张软盘容量的文件。但用 BACKUP 命令得到的文件只能作为备份,调用时必须通过 RESTORE 命令将其恢复。DOS 3.20 和 DOS 3.30 版本还增加了 XCOPY 命令,用这个命令可以拷贝包括较低层次子目录在内的整个目录,XCOPY 复制的文件可以直接调用。

28. DIR 和 TREE 命令在使用上有何不同?

DIR 是 PC DOS 的内部命令, TREE 则属 PC DOS 的外部命令。两个命令都可以显示磁盘上文件的目录,但二者在使用中有所不同。

DIR 命令列出的是根目录或当前目录上的所有目录项或只列出指定文件的目录项。我们知道,从 DOS 2.00 版起提供了一种新的目录结构(即由根目录和多级子目录一层层展开形成的目录结构),当磁盘上建有子目录且子目录为当前目录时,执行 DIR 命令显示会有:

• <DIR> 日期 时间

• • <DIR> 日期 时间

这样两项。一个圆点标志的项表示该项是一个子目录,两个圆点标志的项表示该子目录属于那个上级目录。如果你想知道每一个子目录的名称并了解磁盘上的全部目录结构,则要用 TREE 命令。要使用好文件目录,必须搞清楚根目录、子目录、当前目录、目录路径这几个基本概念,认清它们的区别。对子目录进行操作,有专门的建立、删除、显示和改变子目录的命令。

29. 删除子目录时应当注意那几点?

应当注意以下几点:

(1)被删除的子目录必须是空目录。如果子目录下存在文件先用删除文件的命令 ERASE(或 DEL)将其全部文件删除,再进行删除子目录的操作。

(2)被删除的子目录不为当前目录。

(3)根目录不可删除。子目录下有子目录,要由下向上逐级删除。

30. 使用 TYPE 命令显示文件内容时,为什么有时使屏幕显示发生紊乱而得不到要显示的结果?

TYPE 命令是在屏幕上显示 PC DOS 操作系统的 ASCII 码的文件的内容。所谓 ASCII 码文件即文本文件。如各种编译型语言的源程序(COBOL 语言, PASCAL 语言, FORTRAN 语言等),解释型语言的源程序(BASIC 语言, dBASE II, III 等),批处理文件(·BAT),系统结构设置文件(CONFIG · SYS),应用软件的说明书,用编辑软件写的文章,建立的表格和书信等。若试图用 TYPE 命令显示非 ASCII 码的文件,例如文件扩展名为 · EXE, · COM 等类型的文件,则使屏幕显示紊乱,或显示一些不可标识的图形,而不能得到要显示的文件内容。

(转第 17 页)