



電子

744

與 電腦



• ELECTRONICS AND COMPUTERS •

“8088系统板故障诊断卡” 使维修计算机变得容易了!

IBM PC, PC/XT, 0520A, 0520CH等以8088为CPU的微型计算机,均可采用“8088系统板故障诊断卡”进行在线故障检测,找出故障点,快速排除。

8088系统板故障诊断卡

(IBM PC, PC/XT, 0520CH及兼容机通用) 售价: 1600.00元

为了帮助大家掌握该卡的用法,本刊自第6期起,在“维修经验谈”栏目,举办有关的辅导讲座及培训班。

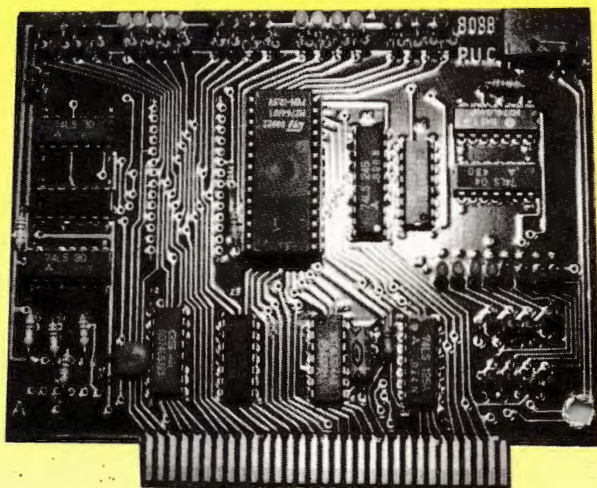
该卡由本刊监制及经销。

银行汇款:

北京工商银行翠微路分理处

891238—72 电子工业出版社《电子与电脑》编辑部

邮局汇款: 北京173信箱 《电子与电脑》编辑部 联系人: 施玉新 邮编: 100036



微机维修技术培训班

本刊邀请中国人民大学信息中心有多年维修经验的专业技术人员任教,在北京不定期举办为期7天的微机维修技术培训班,详细讲授并演示如何利用“诊断卡”准确、迅速地排除系统板故障。同时讲授微机实用维修技术、包括软盘、显示器、电源、键盘。上述内容都以讲课及演示方式授课,使学员在理论及实践的结合中掌握维修技术。

为了保证上课质量,培训班采用小班授课方式,每班不超过30人。欲参加者请复制以下所附报名表寄回。我们收到报名表后,根据报名次序向学员发出上课通知书并通知具体报到、上课时间、地点及乘车路线。

学费200元,资料费实收40元。

本班负责安排食宿(费用自理)并代购回程车票。

联系人: 北京中国人民大学信息中心

胡野红

梁杰熙

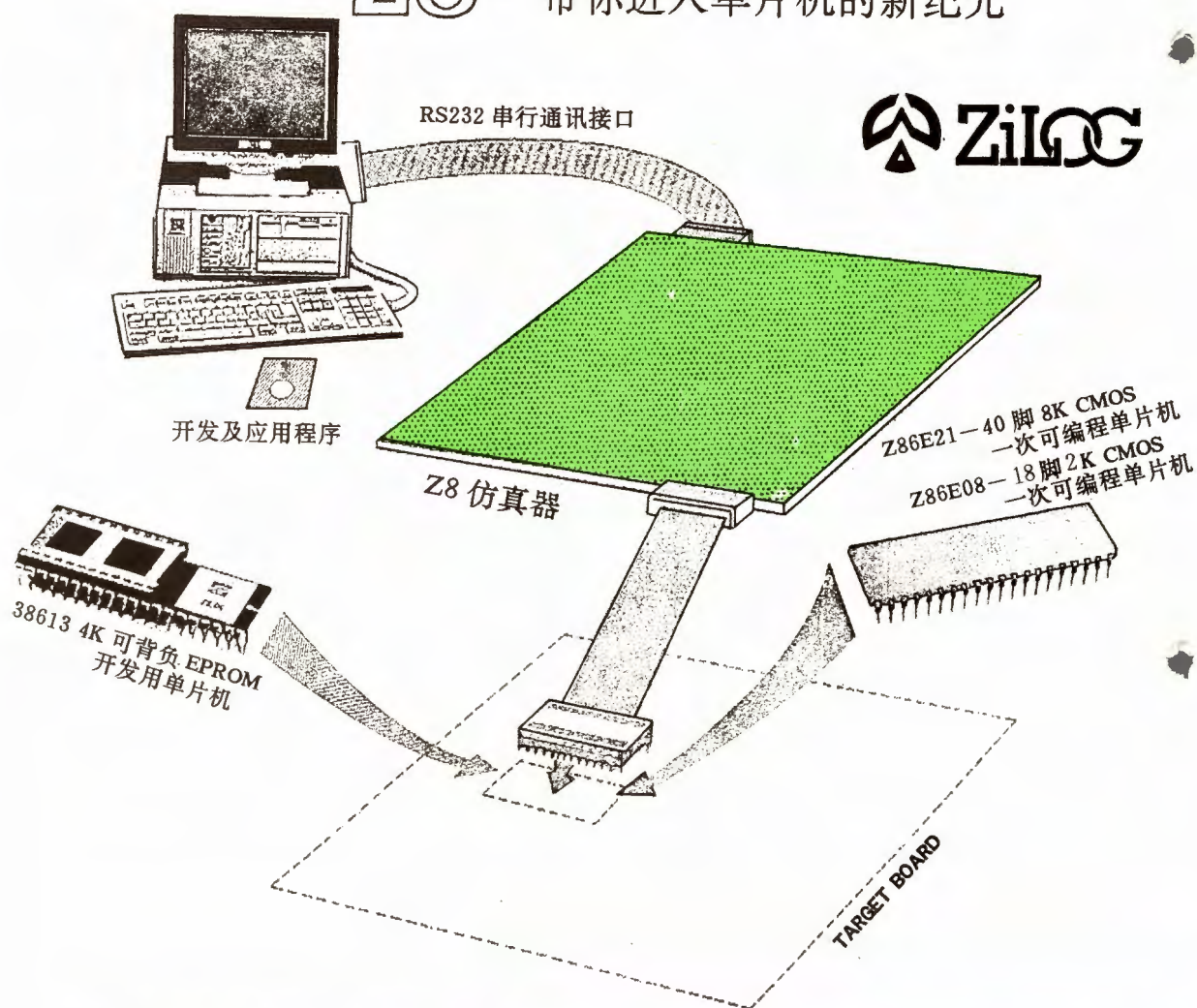
邮政编码: 100872

报名表

姓名	性别	详细通信地址
单位名称	邮政编码	

中国电子器材公司特区公司

‘Z8’ 带你进入单片机的新纪元



使用 Z8 仿真器开发 ZILOG Z8 系列单片机

- 开发成本低
- 开发时间短
- 交货时间准
- 技术支援足
- 售后服务佳

	NMOS	CMOS
ROM SIZE	ROM LESS, 1K, 2K, 4K	ROMLESS, 2K, 4K, 8K
RAM SIZE	76/124/236 BYTE	124/236 BYTE
I/O	14, 22, 32	14, 22, 32
SPEED	8/12MHz	8/12/16MHz
PACKAGE	18, 28, 40 PIN. DIP 44 PIN. PLCC 44 PIN. OFF	18, 28, 40 PIN. DIP 44 PIN. PLCC 44 PIN. OFF
OPTIONS	UART	UART, COMPARATORS, WATCH DOG TIMER, STOP & HALT MODES, RC OSCILLATOR, ROM/RAM PROTECT

购买与租用方法:

现货供应美国 ZILOG 公司 Z8 仿真器每块 4100 元人民币, 或租用每块每月租金 200 元, 欢迎来函来人洽谈。

地址: 深圳市振华路 414 栋 (华发北路口) 电子器材大厦

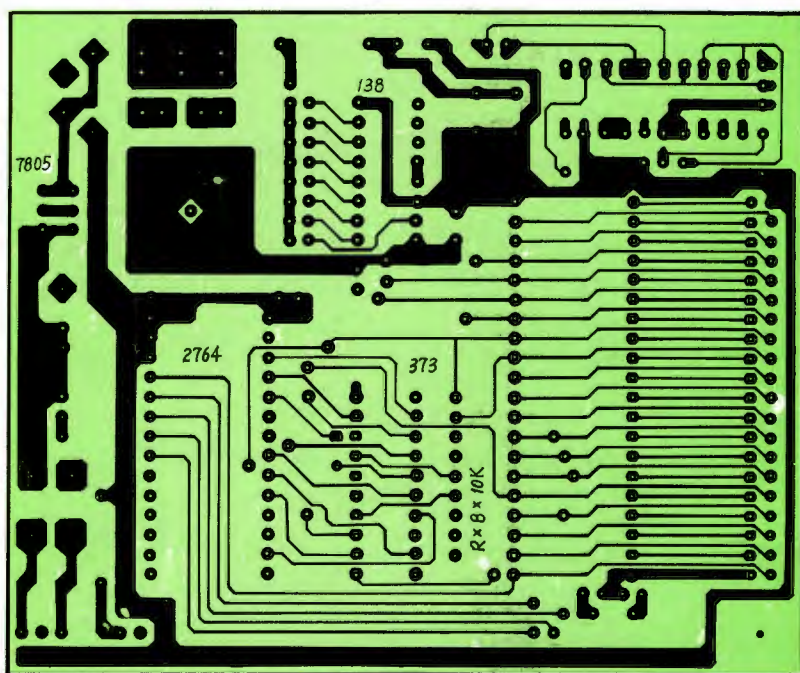
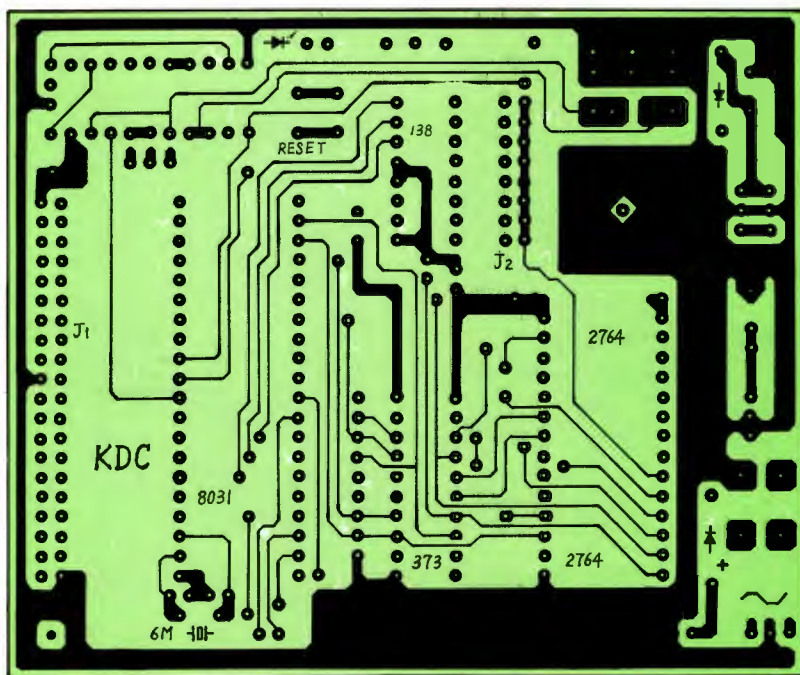
电话: 354214 354345 350877

电挂: 8410 图文传真: 350876

邮编: 518031 银行: 工商银行华强分理处

帐号: 221-06500376

KDC单片机最小系统印制板图 (1 : 1)



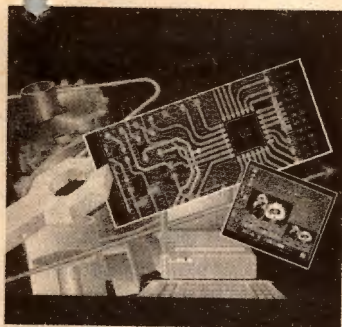


電子
與
電腦

64

一九九一年

总期第75期



ELECTRONICS AND COMPUTERS

電子與電腦

目 录

• 综述 •

- 80年代微机发展回顾及90年代展望(上) 陈志雄(2)

• PC 用户 •

- 汉字同音字查询的实现 李凤明(4)
如何提高 dBASE III 数据库汇总速度 刘昌斌(6)
文件目录修复一例 唐银红(6)
多功能卡日历/时钟功能的开发 张翠玲(8)
Turbo C 问答 施悦华(9)
就 Turbo C 谈充分利用系统内存 宋卫东(9)
女子健美体型电脑咨询程序 黎彦才等(10)
使清屏画面变美妙 王 存(11)
技巧二则 汉 神(12)
利用 DEBUG 进行反汇编存盘的方法 金林樵(13)
90年全国计算机通讯赛题分析 汪五一(14)

• 学习机之友 •

- CEC-I 全功能造字 傅叔平(16)
梵塔问题和奇数幻方的一行程序 陈 纲(18)
APPLE- II 机打印格式的控制 张冠玉(19)
将 BASIC 程序变成 B 型文件并以 BRUN 运行 杨 峰(20)
APPLE DOS 的一处错误 尹进渝(20)
ProDOS 系统盘的修改 廖 凯(21)
制作 40 磁道磁盘的最简方法 瞿 波(21)
PC-1500 机新监控程序 沈玉波(22)

- 关于四舍五入程序的讨论 梁 放 叶德祥(25)

• C 语言初阶讲座 •

- 第五讲 程序控制结构(二) 李 斌 王玉华 蓝智斌(23)

• 学用单片机 •

- 单片机最小应用系统 张培仁(26)
电脑坦克
——KDC 最小系统应用实例之一
..... 张培红 严 琦 阮永光(27)

• 电脑巧开发 •

- 8031 与 PP40 描绘器接口技术 皇甫文忠(29)
解决 TEC-B1 不能使用 AP51-I 卡问题的方法
..... 刘希栋(32)

• 学装微电脑 •

- 学装 μ P-80 微电脑控制电梯模型的实验与制作
..... 易齐干(33)

• 初级程序员级水平考试辅导问答 •

- 汉字 dBASE III 使用技巧(上) 王路敬(37)

• 维修经验谈 •

- 彩色显示器亮度失控的探讨 胡野红(45)
微计算机的维修工具—8088 系统板故障诊断卡
..... 董 珊(46)

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：徐云鹏

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

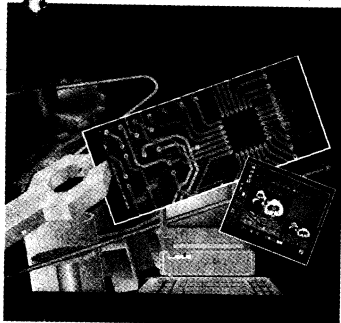


電子
與電腦

64

一九九一年

总期第75期



* ELECTRONICS AND COMPUTERS *

電子與電腦

目 录

• 综述 •

- 80年代微机发展回顾及90年代展望(上) 陈志雄(2)

• PC 用户 •

- 汉字同音字查询的实现 李凤明(4)
 如何提高 dBASE III 数据库汇总速度 刘昌斌(6)
 文件目录修复一例 唐银红(6)
 多功能卡日历/时钟功能的开发 张翠玲(8)
 Turbo C 问答 施悦华(9)
 就 Turbo C 谈充分利用系统内存 宋卫东(9)
 女子健美体型电脑咨询程序 黎彦才等(10)
 使清屏画面变美妙 王 存(11)
 技巧二则 汉 神(12)
 利用 DEBUG 进行反汇编存盘的方法 金林樵(13)
 90年全国计算机通讯赛题分析 汪五一(14)

• 学习机之友 •

- CEC-I 全功能造字 傅叔平(16)
 梵塔问题和奇数幻方的一行程序 陈 纲(18)
 APPLE- II 机打印格式的控制 张冠玉(19)
 将 BASIC 程序变成 B 型文件并以 BRUN 运行 杨 峰(20)
 APPLE DOS 的一处错误 尹进渝(20)
 ProDOS 系统盘的修改 廖 凯(21)
 制作 40 磁道磁盘的最简方法 瞿 波(21)
 PC-1500 机新监控程序 沈玉波(22)

- 关于四舍五入程序的讨论 梁 放 叶德祥(25)

• C 语言初阶讲座 •

- 第五讲 程序控制结构(二) 李 斌 王玉华 蓝智斌(23)

• 学用单片机 •

- 单片机最小应用系统 张培仁(26)
 电脑坦克
 ——KDC 最小系统应用实例之一
 张培红 严 琦 阮永光(27)

• 电脑巧开发 •

- 8031 与 PP40 描绘器接口技术 皇甫文忠(29)
 解决 TEC-B1 不能使用 AP51-I 卡问题的方法
 刘希栋(32)

• 学装微电脑 •

- 学装 $\mu P-80$ 微电脑控制电梯模型的实验与制作
 易齐干(33)

• 初级程序员级水平考试辅导问答 •

- 汉字 dBASE III 使用技巧(上) 王路敬(37)

• 维修经验谈 •

- 彩色显示器亮度失控的探讨 胡野红(45)
 微计算机的维修工具—8088 系统板故障诊断卡
 董 珊(46)

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:徐云鹏

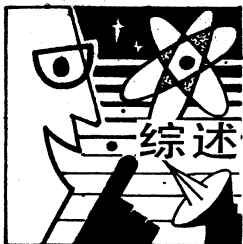
订购处:全国各地邮电局

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



80年代微机发展回顾及90年代展望

艺高电脑国际有限公司总经理 陈志雄

微型计算机是指单芯片微型处理器(microprocessor)为中央处理中心构成的计算机,早期的微处理器功能有限,所以都用于要求低的个人应用计算方面,所以微机同个人计算机(Personal Computer)便常常相提并论。在80年代微处理器技术不断的进步,不少特别设计的微机在工程应用、设计方面有突破性的发展,并一步步代替了不少大型及小型计算机在工程应用、设计上的工作,因为在这一方面的微机好象代替了大型或小型计算机上联网的终端机或工作站,所以该系列的工程应用微机便被称为工作站,与个人电脑在不同的技术及市场领域里发展。

虽然微型机是一个通用名称,但为免得再增加混乱,在这里我以微型机代表个人计算机系列,而工作站则代表为工程应用而设计的计算机系列。

微处理器在70年代初期已制作成功,但微机的发展则到80年代才真正的普及化,自美国IBM公司推出的PC系列之后,市场心理趋使硬件及软件公司相信IBM PC系列将会成为微机标准,因而大量资金投资于PC系列的硬件及软件上。而位于东南亚区的电子厂亦一窝蜂的投资生产IBM PC兼容器,大量的供应迫使价格下降,使微机更容易普及化。在大量软件及低价格的相互作用之下,IBM PC系列微机便发展成一公认的微机标准。80年代便是利用这公认的微机标准不断的发展并普及化,但这标准同时亦带来技术的局限,因为要达至兼容的问题,微机的主要技术指标差不多全部控制于美国Intel公司的手里,在80年代Intel公司推出了8088,80286,及80386系列微处理器,而微机的中央处理器则只能根据Intel公司的发展而进步,直至今天还没有其他比Intel更强的微处理器跟Intel分庭抗礼。

在工作站方面发展跟微机可说是完全不同,工作站到现时还未有一个公认的标准,它的发展要靠针对用户的直接需要,不停的开发新技术,以先进的产品争取市场,很有趣的是微机是以大公司作为领导,但在工作站上比较小的公司反而取得领导地位,在80年代大型计算机公司如IBM, DIGITAL等在工作站的市场上一败涂地,而在不同时间上比较少的公司此起彼落,从早期的Computer Vision, Calma, Apollo, Mentor Graphics,到Daisy, silicon Graphics及近期的HP/Apollo, SUN, MIPS等等都是工作站市场的佼佼者。在技术方面,因不受一个标准所限,各施各法,都以更高的技术及产品取胜,单是微处理器便品类繁多,而且大都比当

时的Intel微处理器功能更强,从早期的Motorola 68000系列, National Semiconductor的32000系列,到现时Motorola的88000新系列, Intel的i860/i960, MIPS的R2000/R3000及SUN的SPARC都是功能强运算速度快的微处理器。除了微处理器外,工作站在其他技术指标方面亦远胜微机如总线速度,网络功能及图形发生及显示器等。

90年代的发展,可能有戏剧性的突破,在微机方面,虽然Intel在1990年推出的80486,垄断可能会被打破,如微机技术的发展不受到标准的局限,而在同一标准的范畴下,微机在性能价格比方面将会直追工作站。现时工作站的领导公司亦正尽再大的努力希望能把本身的技术及产品发展成一公认的标准,如标准化能成为事实90年代工作站的价格亦会大幅回落,直接向微机市场挑战。

90年代最终的发展是怎样,现时还言之过早,本文将对微机及工作站以往的市场技术作分析,并对最近将来的发展作一预测,读者可作自己的结论。

一、微处理器的发展

1. 微机的微处理器发展

在IBM推出PC系列以前,Z80及6502微处理器都是最为普通而分别于S-100总线CP/M操作系统的微机及苹果II型微机之上。在IBM推出PC系列之后,PC系列的Intel 8088微处理器便从此变为主流,在80年代后期Intel推出16位的80286,32位的80386,而在90年推出的80486其实只是在80386的基础上加上协处理器(Co-processor)及8K缓冲存储,在功能上8088只是一个简单适合单用户的8位微处理器,而16位的80286对多任务要求在设计上作出安排,而32位的80386DX是一个同步32字节能支持并行处理的微处理器,80386SX是80386DX的简化体,386SX的内部结构基本上和386DX一产。只是在数据传达/接收时需要两个时钟循环,而不是386DX的一个时钟循环,所以在数据运算速度方面,386SX跟80286基本上相差不远,两者相差的是386SX内部是一个完整的32位微处理器可以运行专为386DX编写的多用户系统软件,而80286则不能。

Intel公司在90年代的发展方向已很明显,所有以后为微机开发的微处理器都会以80386DX为基础,加上外围功能,这些外围功能将包括网络、多媒体传讯(multi-media)如声音、图形辨认、输入/输出、人工智能等等。现时的80486芯片上已集成了一百二十万

个晶体管,但到公元 2000 年则 Intel 公司估计他的 Microchip2000 已可能成功集成一亿个晶体管在一个单芯片上。

Intel 的技术发展方向使很多半导体设计/生产厂家明白到如果他们不能生产一个跟 80386DX 兼容的微处理器,他们在 90 年代便完全没有机会在这个市场上跟 Intel 竞争。问题当然是在要设计一个跟 80386DX 兼容但又不会侵犯 Intel 专利权的微处理器,不管可行性如何,市场上估计最少已有 6 个厂家正在研制 80386DX 的兼容产品,他们能否成功还需拭目以待,但美国的 AMD 公司(Advanced Micro Devices)则成功推出了他的 80386 芯片,但 AMD 的成功并不代表其他厂家可同时取得成功,因为 AMD 跟 Intel 是签订过二线供应合同,即 Intel 与 AMD 根据合同需各自提供本身的设计的产品给对方生产,作为对方的二线供应商。Intel 在 80286 之后已停止供应资料给任何半导体厂家,但 AMD 仍然自行开发 80386DX 微处理器,虽然 Intel 在美国法庭控告 AMD,但市场人士相信因 AMD 过往跟 Intel 签订的合同仍有效,所以 AMD 胜诉的机会很高,而 Intel 将不能以侵犯专利为理由禁止 AMD 生产及销售的 80386DX 芯片。只要 AMD 可以生产 80386DX,80486 便不成问题,而且理论上 AMD 已可以设计比 80486 更多接口功能的微处理器而仍然跟所有标准软件兼容,这可说是直接挑战 Intel 公司十年来的垄断及控制局面。

假定其他正在研制 80386DX 兼容的公司能在技术上或在法庭上解决 Intel 的专利权问题,可以想象不单 80386 和 80486 芯片的价格将会大幅回落,而且更新更强的微处理器将会不断推出,情况将跟工作站用的微处理器的发展一样。

2. 工作站的微处理器发展

象 Intel 的 8088/286/386/486 占领着微机市场一样,早期 Motorola 的 M68000 系列亦差不多占领了整个工作站的市場,现时大部份的工作站厂家都设计微处理器,现将一些标准型而在市场上可购买的微处理器简单作一介绍,介绍暂不包括专用的微处理器(如 IBM RISC,DEC VAX 及 HP/PA 等)。

工作站微处理器可分为两大类:普通指令系统计算机(CISC)及精简指令系统计算机(RISC)。

(1)CISC 普通指令系统计算机

(I)Motorola M68020/030/040

以往大多数的工作站(如 Sun,Apollo,HP 等)都是采用 M68020/030。但 68030 的速度太慢,而 68882 浮点运算速度亦没法跟 RISC 竞争,所以工作站的设计者已一一停止使用 Motorola 的 68000 系列,现时只有 HP/Apollo 还是 Motorola 的唯一最大用户,最新 Motorola 的 68040 含浮点运算器,从 Motorola 提供的数据来看,它确实可与 SPARC 和 MIPS 并驾齐驱。

(II)intel 80286/386/486

虽然 Intel 的微处理器在微机中处于垄断地位,但没有一个工作站厂家能成功的推广 386 为基础的工作

站,SUN 的 386i 可说是完全失败的,但 80486 加上缓冲存储有差不多 20 个 Mips 的速度,所在廉价的无盘工作站(diskless workstation)亦可能会有新的生命力。

(III)National Semiconductors 32032/32332/32536

NS 的 32000 系列微处理器是市面上第一个 32 位微处理器产品,性能优良,但它在工作站市场上则从来没有成功过,反之它作为接口器件的控制器则十分成功(如激光打印机和传真机的控制器)。

(2)RISC 精简指令系统计算机

(I)Sparc

这是目前工作站用的 RISC 处理器中货源最广的一种,至少有五家半导体厂可供应 Sparc 晶片。包括 Fujitsu(门阵列),Cypress(标准元),Bipolar Integrated Technology(放射连结逻辑),LSI Logic(优化门阵列)和 TI(和 SUN 合作的专用 IC)。PHILIPS 与 MATRA 亦宣布加入 Sparc 供应行列。Matsushita 和 Solbourne 正合作开发 64 位的 Sparc,而砷化镓(GaAs)的 Sparc 亦将面世,其速度可达至 200Mips。

(II)MIPS R2000/R3000

MIPS 计算机公司的 R2000/R3000 系列是 Sparc 的强劲竞争对手,在 16—17MHz 主频时速度是 20Mips。该系列起源于斯坦福大学,目前已有许多主要的工作站供应商采用,例如 Digital 已用于他的 DEC station,而 silicon Graphics 更已使用在他全部的系统。MIPS 微处理器的主要弱点是在市场上的数量少,软件的基础较差,但它的长处是内存分配速度快(Memory mapping),而且它的集成度也比较高,每套的芯片数目较少。

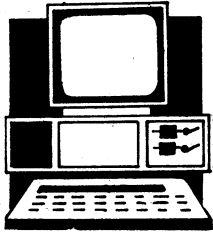
(III)Intel i860/i960

Intel 公司的 RISC 产品主要有两个系列:i960 主要用作接口器件的控制器,而 i860 则是设计用于工作站的。本文将只讨论后者,从性能来说 i860 是众多 RISC 芯片中性能最强的其中一个,它构成了一个整体数运算器(Integer unit),一个浮点运算器(floating point unit),一个图形处理器(Graphics Processing unit),一个在芯片上内存管理器(memory management unit)和缓冲内存(cache)。其速度大约为 25-30Mips 而其浮点运算速度最高可达 40Mflops。单从数字来看 i860 是一种很有竞争能力的产品,但它的问题是缺乏现成的软件。目前 i860 只在 HP/Apollo 的 4000/VRX 中用作图形处理器,在 Floating Point System 公司的 FPS 5000EA 用作阵列处理器。如 i860 的软件充分,它是一个功能极强的工作站中央处理器。

(IV)Motorola M88000

Motorola 的 M88000 RISC 还未在工作站市场上有任何影响,Unisys 公司现时是唯一选择了 M88000 作为工作站的,估计 Motorola 的 M88000 将不会成为主流产品。

可以看得得到工作站的微处理器发展得比微机的 286/386/486 快,这是在没有标准局限的结果,所以 90 年代的微机(PC 机)微处理器的发展(下转 28 页)



PC用户

汉字同音字查询的实现

绵阳市职工社会保险事业管理处

李凤明

在人事档案、户籍、工资等管理系统中,通过输入姓名或其它汉字信息进行查询是较常用的查询方法。这类查询一般先按查询内容进行索引,然后通过定位函数定位来实现查询,它要求输入的查询内容必须和索引内容一致才能正确定位查询。在查询时,如果输入了同音汉字,它就无能为力了。

这是因为使用索引进行查询时,是将查询内容的 ASCII 码与索引内容的 ASCII 码进行比较,两者一致时才认为是查询的记录,而同音字的机内码各不相同,按这种查询方法,若只知道姓名或其它汉字信息,将会出现漏查。在实际工作中,经常会出现同音的情况,这就需要一种汉字同音字查询技术。

根据《通讯用汉字字符集(基本集)及其交换码国家标准(GB2312-80)》,汉字是按拼音字母顺序排列的,同音字基本上在同一区中。每个汉字的机内码为两个字节,其高字节部分确定了它的区号。对汉字的区号进行比较,区号相同即可能为同音字,比较的汉字个数越多,同音的情况也就越确定。

根据这个思想,现介绍一种用 Foxbase+2.00 版本实现的,可对数据库中的姓名字段进行同音查询程序 XMTYCX.PRG(程序附后)。程序为一个过程,可被不同的数据库调用。另外函数 XMCL.PRG(程序附后)对姓名进行处理,将输入的姓名中的空格去掉,在数据库录入时应调用该函数处理姓名。

假定现有职工工资数据库 GZK.DBF(库结构附后),已按姓名字段建立了索引文件 GZK.IDX;显示某人具体情况的程序名为 QKXS.PRG。通过下面的语句即可实现对 GZK 库中姓名的同音查询:

DO XMTYCX WITH "GZK", "GZK", "姓名", "QKXS"

程序只需稍加修改,就可实现对其它汉字信息的同音字查询,也可用其它语言实现。程序在 SPIC 286 微机和浪潮 0520-D 微机上运行通过。

功能说明:1、可对姓名进行同音字查询,一屏显示 18 个记录,也可前后翻页。

2、输入记录号可显示某人的具体情况。

调用参数:1、查询库名;2、查询库按姓名索引文件名;

3、姓名字段名;4、显示具体情况子程序。

调用形式:

DO XCTYCX "查询库名", "姓名索引文件", "姓名字段名", "显示情况子程序"

调用注意事项:本程序将选择 J 工作区打开数据库,对其他工作区无影响。

同音查询程序(XMTYCX.PRG):

```
para x,y,z,w
dime cxm(4)
priv oldse,i,j,n,xxm,ne,con
set stat off
set talk off
set esca off
oldse=ltri(str(sele(),2,0)) && 保护原工作区号
sele j
use &x inde &y
do while .t.
xxm=spac(8)
clea
@24,0 say "请输入姓名(按 ESC 键退出)" get xxm
read
if read()=12 && 按 ESC 键退出
exit
endi
if len(trim(xxm))=0
loop
endi
xxm=xmcl(xxm) && 调姓名处理函数处理查询姓名
cxm=""
cxm(1)=left(xxm,1)
seek cxm(1) && 定位姓同音开始记录
if eof()
@24,0 clea
@24,0 say "查无此人,按任一键继续查询……"
read
loop
endi
n=len(xxm)
i=3
j=2
do while i<=n && 取查询姓名每个字的高字节定区号
cxm(j)=subs(xxm,i,1)
i=i+2
j=j+1
endd
ne=18 && 定义每屏显示记录个数
con="subs(&z,3,1)=cxm(2). and. subs(&z,5,1)=cxm(3)"
&& 定义姓名同音条件
do while .t.
@2,0 clea to 24,79
@24,0 say "正在查询,按 ESC 键停止查询……"
@2,0 say "记录号-----姓 名"
i=1
do while &z=cxm(1). and. i<=ne
```

```

if inke()=27 && 按 ESC 键停止查询
exit
endi
if &.con && 显示满足条件的记录
@2+1,0 say str(recn(),6)+" "+&.z
i=i+1
re=recn()
endi
skip
endd
if i=1 && 文件结束未查到
@24,0 say "查无此人,请按任一键重新输入....."
read
exit
endi
i=0
@24,0 say "请输入 ↑——上翻 ↓——下翻 ESC——退出;
或输入记录号" get i pict "999999" rang 1,recn()
read
if read()=12 && 按 ESC 键退出
exit
endi
do case
case read()=4 && 上翻
skip -ne * 2
case i>0 && 显示某人具体情况
go i
on erro exit && 出错退出
do &.w
go re
clea
othe
go re
endc
if &.z<>cxm(1)
seek cxm(1)
endi
endd
endd
on erro && 取消出错判断
use
sele &.oldse && 恢复原工作区号
retu

* 姓名处理函数(XMCL.PRG)
* 去掉字与字之间的空格
para xx
priv i,j,n,nxm
xx=ltrim(trim(xx))
n=len(xx)

```

```

i=1
nxm=""
do while i<=n
if subs(xx,j,1)<>" "
nxm=nxm+subs(xx,i,1)
endi
i=i+1
endd
retu nxm

* 情况显示子程序(QKXS.PRG)
clea
disp off
wait
retu

```

职工工资数据库 GZK.DBF 结构:
Structure for database: E:\GZK.DBF
Number of data records: 90
Date of last update ,03/08/91

Field	Field Name	Type	Width	Dec
1	姓名	Character	6	
2	基础工资	Numeric	5	2
3	职务津贴	Numeric	6	2
4	粮贴	Numeric	2	
5	燃贴	Numeric	2	
6	综合补贴	Numeric	2	
7	交通费	Numeric	2	
8	书报费	Numeric	2	
9	副贴	Numeric	2	
10	肉贴	Numeric	2	
11	洗理费	Numeric	2	
12	合计	Numeric	5	1
** Total **			39	

程序运行结果

.do xmtycx with "gzk","gzk","姓名","qkxs"
请输入姓名(按 ESC 键退出)李奉鸣

正在查询,按 ESC 键停止查询.....

记录号 ----- 姓名
12 李凤明

请输入 ↑——上翻 ↓——下翻 ESC——退出 或输入记录号
12

姓名	基础工资	职务津贴	粮贴	燃贴	综合补贴	
李凤明	39.00	29.00	0	15	50	
	交通费	书报费	副贴	肉贴	洗理费	合计
	5	6	5	10	6	165.0

Press any key to continue.....

如何提高 CdBASE III 数据库汇总速度

广西玉林建设银行电脑室 刘昌斌

目前,大部分管理机构每月、每年均要用 CdBASE III 程序汇总各个下级机构的报表数据。我们建行系统每月、每年都要定期汇总下级资金平衡表,这些下级的资金平衡表数据库结构、记录个数和顺序各自相同,而且他们与汇总库的结构、记录个数和顺序也相同。汇总各下级的数据实质就是把各个下级的数据库按科目号相同的记录的数据累加到汇总库中。

CdBASE III 可用于汇总数据的语句和方式很多,但是目前被广泛使用的汇总方法基本是:首先打开过渡汇总库,然后用语句“APPE FROM(数据库名)”的形式把各下级的数据库记录一个一个地添加到过渡汇总库,这样就产生了一个庞大的过渡性数据库,对其按某个关键字建立索引后,再打开索引,此后再用“TOTAL TO(汇总库名) ON(关键字) ALL”语句对过渡汇总库具有相同科目号的记录汇总到汇总库内。用这种方法汇总数据速度很慢,因为用 APPE FROM 语句较为耗时,对庞大的数据库建立索引就是耗时的致命的弱点,用这种方法把结构为 14 字段、241 个记录的 14 个下级行数据库汇总需要 1 小时。

虽然用户都知道上述数据汇总方法的速度较慢,但是有不少的用户忽视了应用 REPL 语句进行数据库汇总的技巧。由于各下级的数据库结构都与汇总库相同,这样就可以建立公共索引,然后在两个不同的工作区分别打开汇总库及其索引文件和下级行库及其索引文件(用语句“SET REL TO(关键字) INTO(数据库别名)”,把汇总库和下级行库建立了关系(根据关键字连接两个数据库),此时仅用一个语句“REPL ALL 字段名 WITH 字段名+别名->字段名,...”,就可把一个下级的数据汇总到汇总库内,速度极快。例:应用本方法汇总 14 个下级的数据,汇总库名为 JPY-SUM.DBF,下级行数据库名为 JPYn.DBF(n=1--14),他们的记录个数都是 241,他们的结构是:

字段名	类别	宽度	小数
FKMH	C	4	0
FKMM	C	32	0
FYCYE	N	14	2
FBYS	N	14	2
FBYF	N	14	2
FYE	N	14	2

应用程序(一)的技巧方法汇总数据,不但可极大地提高数据汇总速度,而且避免了产生庞大数据库,同时节省大量的外存空间,简直是一举三得。该数据库汇总方法具有通用性,适用于各个管理机构,现提供各用户借以抛砖引玉。

程序(一)

* * 技巧汇总数据库程序 JPYHZ. PRG

```
sele 1
use jpysum
rep all fycye with 0,fbys with 0,fbyf with 0,fye with 0
inde on fkmh to jpy11
use jpysum inde jpy11
store 1 to n
do while n<15
@ 6,32 say "汇总第"+str(n,2)+"下级行"
if n<10
store "jpy"+str(n,1) to jpy1
else
store "jpy"+str(n,2) to jpy1
endif
sele 2
use &jpy1 inde jpy11
sele 1
set rela to fkmh into &jpy1
repl all fycye with fycye+b->fycye,fbys with fbys+b->fbys,fbyf with fbyf+b->fbyf,fye with fye+b->fye
sele 2
clos inde
n=n+1
enddo
clos data
retur
```

文件目录修复一例

湖北远安县财政局 唐银红

由于病毒侵袭,破坏了硬盘上的文件目录区,用 DIR 命令显示硬盘文件目录时(不带任何参数),目录区前面的绝大部分目录项显示正常,到最后部分显示有 4—5 行乱七八糟的字符,并且每行都要显示到 80 列,除有能正常显示的英文字母、数字及普通字符外,还有大量的控制字符,整个硬盘上有十几个文件丢失。

以上现象表明:虽有十几个文件目录丢失,但系统认为这些文件仍然存在,继续占用硬盘空间。为了验证这一现象,用 PCTOOLS(R4.30 版)显示硬盘根目录,被破坏的部分的文件名、文件长度、建立日期等信息,竟然好端端地显示在目录区的最后,当试图对这些文件进行访问(读写、拷贝操作)时,PCTOOLS 的反应是系统中没有这些文件,访问失败。只能将这些文件名一

一抄录下来。

这些文件既占用了硬盘空间,又不能对其进行读写操作,必须对其进行删除操作。显然,这些文件的文件名都不能正常显示,就无法用 DOS 的命令 DEL FILENAME 进行删除操作。当然可以用 DEL *.* 命令或对硬盘重新进行格式化工作,但却要对硬盘进行备份,恢复后还要重建有关必要数据,不仅显得麻烦,工作量大,而且耗时长。我们知道, DOS 对文件目录的管理,是以文件名的第一个字符来区分一个文件是否是有效文件的,若文件被删除后, DOS 就将该文件名的第一个字符改为十六进制的 E5H。因此,只要将被破坏的文件名的第一个字符改为 E5H 即可。用以下两种方法都可以达到修复的目的。

方法一:用调试程序 DEBUG 修复。

1、用 DIR 命令显示文件目录,记下不能正确显示的文件前面的最后一个文件名,本例为 DBASE. EXE。

2、DEBUG<ENTER>进入调试程序。

3、将硬盘根目录区调入内存。

— L100 2 11 20 <ENTER>

100 为装入内存中的起始地址,2 是 C 盘(硬盘)的代号,11 是 20M 硬盘根目录区的相对扇区起始号,20 是要读入的扇区数。

4、— S 100 4100 'DBASE EXE' <ENTER>

装入的目录长为 16KB(32 个字节表示一个文件名的全部信息,装入的硬盘目录共有 512 个),用十六进制数表示为 4100H,因此要在 100H—4100H 范围内搜寻名为 DBASE. EXE 的文件。注意,文件名必须大写,文件名不足八个字符的部分用空格键补齐,文件名与扩展名之间不能有任何分隔符。

当 DEBUG 找到 DBASE. EXE 文件名后,立即显示文件名在内存中的地址,本例为:

4285: 1CA0

5、— D1CA0 1DBF

从下个目录开始,观察不能正常显示的文件个数,也可省略这一步,根据前面用 PCTOOLS 观察时记录的文件个数。

6、进行修改。具体是从第一个不能正常显示的文件目录开始,用 E 命令修改。

— E1CC0 <ENTER>

4285:1CC0 18. E5<ENTER>

修改第二个不能显示的文件名时,将地址值在前一个文件的地址值基础上,加上 20H 进行修改,以此类推,下一个文件的首地址均在上一个文件名首地址值的基础上加 20H,直到修改完毕。

7、— W 100 2 11 20 <ENTER>

将修改后的目录内容写回硬盘。注意,读盘时用什么参数,写盘时一定要一致。

8、— Q <ENTER> 退出 DEBUG,返回 DOS。

用 DEBUG 进行修改,不需要工具软件,但不方便,对不熟悉 DEBUG 使用方法的用户来说,稍有差错,损失就更大,建议用第二种方法修改。

方法二:用工具软件 PCTOOLS 进行修改(PC-TOOLS 版本号为 R4.30)。

1、PCTOOLS <ENTER>

2、F3 进入磁盘服务功能,选择根目录区。

3、E 进入磁盘编辑功能,此时,显示器上显示磁盘数据信息,从绝对扇区 0 开始显示,每行显示 16 个 ASCII 码,左边为十六进制数,右边显示对应字符,按 PgUp、PgDn 两个键,可以使显示内容上移一屏或下移一屏。首先显示的是三个系统文件的内容,然后是根目录区,连续翻动,直到找到文件 DBASE. EXE 为止。

4、F3 打开编辑开关,允许对磁盘文件进行编辑(读写操作)。用四个光标键(UP ARROW、DOWN ARROW、LEFT ARROW、RIGHT ARROW)移动光标,到不能正常显示的文件目录的第一个字符处,改成 E5H(在左边十六进制区修改,按 F1 键可调换修改的区域为十六进制或 ASCII 字符区),每二行(32 个字符)显示一个文件名及其有关信息,重复上述操作,直到修改完所有的不能正常显示的文件目录为止。一次可修改两页,按 PgUp、PgDn 键可交换上下页进行修改。

5、F5 进行修改确认,保证修改后的内容有效(F6 放弃,使修改作废)。

6、<ESC>退出磁盘编辑功能,再按一次<ESC>键,确认退出 PCTOOLS。

7、DOS 控制下,运行带 F 参数的外部命令 CHKDSK/F,将系统丢失的簇号收回。

8、根据前面所列丢失文件名的清单,将删除的文件逐个从软盘拷回。

至此,整个文件修复工作完成。

注一:修改过程中,多次用 DIR 命令检测硬盘可用空间。修复工作开始前,用 CHKDSK/F 命令,不能检测到系统有丢失的簇,证明系统认为那些文件占用的磁盘空间是合法的;至第六步结束,用 DIR 命令列硬盘目录,显示的硬盘可用空间没有发生变化,表明删除的文件占用的磁盘空间不能正常释放,至第七步结束,用 DIR 命令列硬盘目录,发现硬盘上增加四十多个 FILE * .CHK 文件,将这些文件删除后,硬盘可用空间增加 300KB 之多,表明修复成功。

注二:因为文件目录信息已全部丢失,包括文件名、文件长度、首簇号、建立日期、文件属性、修改标志等,所以修复时不能直接改为原文件名,而必须进行删除操作。

注三:文中带下划线的部分由用户输入,< > 表示为一个键,例如<ENTER>表示回车换行键。

多功能卡日历/时钟功能的开发

辽宁公路专科学校 张翠玲

目前的绝大多数 PC/XT 兼容机和家用 PC 机上,均安装有多功能卡。它为用户提供了五种功能,即软盘控制器接口、并行打印机接口、串行通信接口以及游戏适配器等。在卡上藏有一枚 3 伏锂电池,当计算机关机时,时钟芯片可以通过电池的能量来维持时间和日期。但由于缺少相应的驱动程序,使得日历/时钟功能没有得到充分的发挥,白白浪费了硬件资源。为此,有必要开发日历/时钟驱动程序,以省去每次开机时输入日期和时间的麻烦。

一、多功能卡的硬件特性

在多功能卡的上方有两排各十脚的插座,脚与脚之间的不同连接,可以改变多功能卡的功能。对于实时时钟(参照下图),当第 4 脚与第 3 脚连接时,为时钟 1 有效;而第 4 脚与第 6 脚连接时,为时钟 2 有效;如果第 3、4、6 脚均断开,则时钟功能无效。因此,为了使时钟正常工作,必须保证 4、3 脚或 4、6 脚有一对是相连的。

2	4	6	8	10	12	14	16	18	20	上
1	3	5	7	9	11	13	15	17	19	下

插脚连接方式的不同,在访问时钟芯片时所采用的地址也是不同的。多功能卡时钟芯片上的硬件地址规定如下:

时钟 1: &H240

时钟 2: &H340

二、对时钟芯片的编程

多功能卡上的时钟芯片内包含有时间和日期各部分计数器,见表 1。每一个计数器可以通过时钟芯片地址的相对偏移量来访问。

表 1: 时钟芯片相对偏移地址的内容

时钟芯片口地址	说明
+0	万分之一秒计数器
+1	百分秒计数器
+2	秒计数器
+3	分计数器
+4	小时计数器
+5	星期计数器
+6	日计数器
+7	月计数器
+8	年计数器
+9	上一个月份数
+A	上一年数

为了能对时钟芯片进行编程,首先必须确定出时钟芯片的端口地址。这可以根据插脚的连接方式来判断,但这种方法显得有些笨拙,因为不同的多功能卡其插脚的连接方式可能有所不同。寻找芯片较好的方法是读一些通常被分配给时钟的口地址,如 240H 和 340H 口,与硬件无关的口将返回值 0FEH 或 0FCH。例如要检验秒计数器时,其返回值不能大于 59,否则说明所检验的不是时钟口地址。

寻找时钟芯片的程序段如下:

```

CLKTAB LABEL BYTE
        DW 240H
        DW 340H

        LEA SI,CLKTAB ;SI 指向时钟表地址
        MOV CX,2       ;共两个口地址
FIND:   MOV DX,[SI]     ;第一口地址送 DX
        ADD DX,2       ;加 2 指向秒计数器
        IN AL,DX       ;取秒计数器值到 AL 中
        TEST AL,80H    ;高位设置?
        JZ GO          ;NO,非空口,继续执行
        ADD SI,2       ;否则指向下一口地址
        LOOP FIND
        LEA DX,MESS    ;显示信息
        MOV AH,09H
        INT 21H
        INT 20H
GO:
MESS DB 'No clock chip found! $',0DH,0AH
    
```

虽然时钟芯片本身没有自动增加年数的功能,但从表 1 中可以看到,时钟芯片上的 9 口和 A 口是用来存储上个月和前一年的信息的,我们可以利用这两口的信息对年数进行处理:首先判断上个月和本月两个计数器值的关系,若上个月数大于本月数,则年数增 1,将增加后的年数写回到芯片上,否则年数不变。处理年数的程序段如下:(程序中 MONTH 变量为芯片上 7 口中的内容)

```

MOV DX,24AH ;从 24AH 或 34AH 口取前一年
IN AL,DX    ;存储器的信息放在 AL 中
MOV AH,AL   ;AL 送给 AH
DEC DX      ;指向 249H 口
IN AL,DX    ;取前一个月的信息放在 AL
CMP AL,MONTH ;与本月份数比较
JBE CONT    ;不大于本月数,转 CONT
INC AH      ;否则,前一年数加 1
MOV AL,AH
OUT DX,AL   ;写入 24AH 口
    
```

CONT:

有了上述了解,就可以编写时钟驱动程序了。编程时应注意时钟芯片是采用BCD码来存储信息的,因此使用这些信息时需进行一些转换,如要显示日期和时间,需将读出的BCD数转换成十六进制数,再转换成ASCII码;而在写芯片时,又需将从系统中得到的日期和时钟转换成BCD数。考虑到要缩短机器的启动时间及以后使用上的方便,应分别编写出读时钟和写时钟程序,读时钟程序的执行过程大致如下:从时钟芯片端口分别取出时间和日期内容,转换成十六进制数后,再调用21H中断的2DH号(置系统时间)和2BH号(置系统日期)功能,将芯片上的信息传到系统时钟上。写时钟正好与之相反,它先调用21H中断的2CH(取系统时间)和2AH(取系统日期)号功能,再将时间、日期转换成BCD数写到芯片上。

首次使用时,需给计算机输入正确的时间和日期,之后执行写时钟程序,并在系统盘或硬盘上建立AUTOEXEC.BAT文件,将读时钟程序的文件名及DOS命令DATE和TIME加入到该文件中,这样每次开机后,系统便会自动显示准确的日期和时间,既利用了硬件资源,又省去输入日期和时间的麻烦。

限于篇幅,源程序清单从略。

Turbo C 问答

天津继电器厂 施悦华

微机上的程序设计语言TurboC以它的功能强大和使用方便而深受欢迎,但有些初学者和已能用TurboC编程的用户对某些问题没有给予足够的注意,这里就列出几个问题并给出答案:

TC.EXE与TCC.EXE有何不同?

TurboC带有一个综合编译系统TC.EXE和一个行编译系统TCC.EXE。可以说TC.EXE是带有编辑、连接和跟踪功能的TCC.EXE。两者的配置文件分别为:TCCONFIG.TC和TURBOC.CFG。

为什么TurboC找不到用户的#include文件?

编译系统只在\include目录下寻找包含文件,这个目录可以通过Option/Directory菜单指定。可以把用户的包含文件都放入\include目录。

为什么出现Unable to open include file 'stdarg.h'?

因为超过DOS允许同时打开的文件数,把FILE=20加入配置文件CONFIG.SYS中即可。

如何在TurboC编辑状态下打印源程序?

用键CTRL<K><P>,可以打印标记块,如果没有标记,则打印整个源程序。

怎样产生打印输出?

TurboC在STDIO.H中把打印机定义为一个文件,在使用前用户不必打开stdprn,只须:

```
#include<stdio.h>
```

```
main( )
```

```
{ printf(stdprn, "Hello. \n"); }即可
```

为什么printf()和puts()不能显示彩色字符?

必须使用控制台输入/输出语句cprintf()和语句cputs()。

如何创建.COM文件?

用户手中如有EXE2BIN.EXE,则可以在Tiny模式下建立.EXE文件,然后用它转换为.COM文件。如果没有,也可使用TLINK程序:

```
TCC -mt -lt -tiny
```

它建立tiny.com代替tiny.exe(浮点运算不能转为.COM文件)。

就 Turbo C 谈 充分利用系统内存

天津海洋技术研究所 宋卫东

在各种计算机语言中,由于硬件的原故,突破64K内存是不容易的(特别是高级语言)。这就使用户受到了限制。特别是目前各种机型的内存正迅速扩大,不充分利用很可惜,也是很大的浪费。下面就谈一下在TurboC语言中,如何使用大于64K的数据。

例如我们要定义一个1024×128的二维数组。您也许会写成:

```
char array[1024][128]
```

但因为尺寸超过了64K,它必须直接从系统堆去分配:

```
#include<alloc.h>
```

```
char (huge * array)[128]
```

```
main()
```

```
{.....
```

```
array=farcalloc(sizeof(* array),1024);
```

```
.....}
```

这一个数组的用法同其它直接分配的数组一样。注意,在数组定义过程中必须使用关键字“huge”因为只有“huge”才能寻址到64K以外。这样就扩大了内存的使用范围,提高了计算机的使用效率。

女子健美体型电脑咨询程序

四川泸州医学院 黎彦才 李晓阳 高明扬

近年来,健美运动在我国青少年中得到越来越广泛的开展。在健美训练中,人们希望随时了解自己训练的效果和体型是否符合标准,以及那些部位尚需加强训练等。利用电脑进行健美体型的咨询,无疑是一种高效率而方便的方法。我们根据有关资料在 IBM PC 机上编制了女子健美体型(国内标准)电脑咨询程序,在用户键入测量数据后,电脑将根据输入数据进行计算并打印出咨询单,用户可以从咨询单上看出根据自己身高计算出的自己身体各部位应有的比例以及自己现在在体型各部位的实际比例,科学地进行健美训练。

本程序使用简便,只要按计算机屏幕显示的汉字提示,即可方便地进行操作,得到咨询。

女子健美体型标准为:

1. 体重=(身高[厘米]-100)×0.85[公斤]

2. 胸围:应为身高的一半

3. 腰围:较胸围小 20 厘米

4. 臀围:较胸围大 4 厘米

5. 大腿围:较腰围小 10 厘米

6. 小腿围:较大腿围小 20 厘米

7. 足颈围:较小腿围小 10 厘米

8. 手腕围:较足颈围小 5 厘米

9. 上臂围:等于大腿围的一半

10. 颈围:与小腿围相等

11. 肩宽:应为胸围的一半减 4 厘米

12. 上、下身比例(以肚脐为界):5:8

5 CLS

15 PRINT "女子健美体型电脑咨询"

17 PRINT "请按提示键入测量数据(厘米)"

20 PRINT "身高=", INPUT A

25 PRINT "体重(公斤)", INPUT A1

30 PRINT "胸围=", INPUT A2

40 PRINT "腰围=", INPUT A3

50 PRINT "臀围=", INPUT A4

60 PRINT "大腿围=", INPUT A5

70 PRINT "小腿围=", INPUT A6

80 PRINT "足颈围=", INPUT A7

90 PRINT "手腕围=", INPUT A8

100 PRINT "上臂围=", INPUT A9

110 PRINT "颈围=", INPUT A10

115 PRINT "肩宽=", INPUT A11

118 PRINT "脐高=", INPUT A12

120 X=(5*A12)/(A-A12);B1=(A-100)*.85

130 B2=A/2;B3=B2-20;B4=B2+4;B5=B3-10

135 B6=B5-20;B7=B6-10;B8=B7-5;B9=B5/2

140 B10=B6;B11=B2/2-4

150 A\$="身高";A1\$="体重";A2\$="胸围"

160 A3\$="腰围";A4\$="臀围";A5\$="大腿围"

170 A6\$="小腿围";A7\$="足颈围"

175 A8\$="手腕围";A9\$="上臂围"

180 A10\$="颈围";A11\$="肩宽"

190 A12\$="上身下身比例";CLS

230 LPRINT TAB(18);STRING\$(62,"*")

250 LPRINT TAB(34)"女子健美体型电脑咨询"

252 LPRINT TAB(34)"(国内女子健美体型标准)"

260 LPRINT TAB(18);STRING\$(62,"*")

275 LPRINT TAB(20)"您的身高";A;"(厘米)",

280 LPRINT TAB(53)"您的体型健美标准应是:"

285 LPRINT TAB(20)"体重: ";B1;"(公斤);

290 LPRINT TAB(40)A2\$";B2;"(厘米)"

295 LPRINT TAB(60)A3\$";B3;"(厘米)"

305 LPRINT TAB(20)A4\$";B4;"(厘米)"

310 LPRINT TAB(40)A5\$";B5;"(厘米)"

315 LPRINT TAB(60)A6\$";B6;"(厘米)"

325 LPRINT TAB(20)A7\$";B7;"(厘米)"

330 LPRINT TAB(40)A8\$";B8;"(厘米)"

335 LPRINT TAB(60)A9\$";B9;"(厘米)"

345 LPRINT TAB(20)A10\$";B10;"(厘米)"

350 LPRINT TAB(40)A11\$";B11;"(厘米)"

355 LPRINT TAB(60)A12\$";B12;"(厘米)"

365 LPRINT

368 LPRINT TAB(20)"您体型的实际情况是"

370 LPRINT TAB(20)"体重: ";A1;"公斤"

375 LPRINT TAB(40)A2\$";A2;"(厘米)"

380 LPRINT TAB(60)A3\$";A3;"(厘米)"

390 LPRINT TAB(20)A4\$";A4;"(厘米)"

392 LPRINT TAB(40)A5\$";A5;"(厘米)"

395 LPRINT TAB(60)A6\$";A6;"(厘米)"

400 LPRINT TAB(20)A7\$";A7;"(厘米)"

405 LPRINT TAB(40)A8\$";A8;"(厘米)"

410 LPRINT TAB(60)A9\$";A9;"(厘米)"

420 LPRINT TAB(20)A10\$";A10;"(厘米)"

425 LPRINT TAB(40)A11\$";A11;"(厘米)"

435 LPRINT TAB(60)A12\$";A12;"(厘米)"

440 LPRINT TAB(18);STRING\$(62,"*")

455 LPRINT TAB(43)"测量方法"

470 LPRINT TAB(18)"1. 胸围:由腋下沿胸上方最丰满处测量";

475 LPRINT TAB(54)"7. 颈围:在中部颈最细处"

480 LPRINT TAB(18)"2. 上臂围:在肩关节与肘之间的中部关节";

485 LPRINT TAB(54)"8. 足颈围:在足颈的最细部位"

500 LPRINT TAB(18)"3. 臀围:在体前耻骨平行于臀部最大部位";

510 LPRINT TAB(54)"9. 手腕围:在手腕的最细部位"

520 LPRINT TAB(18)"4. 腰围:在正常情况下,腰的最细部位";

525 LPRINT TAB(54)"10. 肩宽:两肩峰之间的距离"

530 LPRINT TAB(18)"5. 大腿围:在大腿的最上部,臀

折线下”;

```
540 LPRINT TAB(54)“11. 上,下身比例:以肚脐为界”
550 LPRINT TAB(18)“6. 小腿围:在小腿最丰满处”
560 LPRINT TAB(54)“12. 脐高:站立时肚脐至地面高”
610 LPRINT TAB(18);STRING$(62,“*”)
```

```
630 PRINT,PRINT“是否继续咨询(Y/N)”;
640 INPUT C$
645 IF C$ = “y” OR C$ = “Y” THEN GOTO 5
650 END
```

使清屏画面变美妙

邯郸市财政局 王存

很多应用软件,都颇为重视屏幕静态画面的设计,却往往忽视屏幕动态画面的制作。象属于屏幕动态画面之列的清屏过程等,设计者却总习惯采用单一的黑底色将屏幕一清之了,很少考虑清屏过程潜在的积极意义。其实,清屏画面设计得优与劣,不仅体现软件设计水平的高与低,也直接影响软件的应用价值,关联用户的上机情趣。

为了提高应用软件的自身价值,使用户对所用软件一见则喜,百用不厌,始终感到耳目一新,兴味盎然,就需要考虑将清屏过程加以改造,使其变得图案新颖别致,设色鲜艳艳丽,节奏欢快和谐。下面给出的几种清屏程序,是我们在实践中创造的几个代表作。各程序都很短小,但清屏效果很好,其特色可概述如下:

程序一:挂帘垂落 一幅五彩挂帘随着悦耳的声响逐渐垂落,把原先的屏幕内容清除与覆盖。

程序二:帷幕衔启 象戏剧换场一样,伴着音乐节奏,漂亮华丽的帷幕先从两侧合拢一起,继而又被拉开,形成一种可供推出新的屏幕内容的环境。

程序三:百叶窗关合 模拟百叶窗徐徐关合之过程,以达到清除原屏幕内容之目的。

程序四:激光万束 利用产生随机数的函数,生成激光辐射般的万道光束。这些光束自左上向右下照射,参差不齐,错落有致,绚丽多彩,耀眼夺目。

程序五:凤凰展翅 斑斓的凤凰,在烈火中诞生。翎翅,象朝霞殷红;毛羽,似晨晕金黄。当她凌空飞翔之时,旧的屏幕内容便被遮蔽。

程序六:彩蝶翩翩 似在花丛中翻飞,又象在百草间飘旋,翩翩起舞的花蝴蝶舒展艳丽的翅膀,遮掩屏幕上原有的一切。

程序七:碧波涟漪 一泓碧水,明澈如镜。突然,一串石子从天而降,把水面的平静砸得破碎。溅落处,水翻涡旋,波声清脆,美丽的涟漪涌向四围。

程序八:红旗漫卷 红艳艳的旗帜,迎风飘扬;金灿灿的旗穗,飞舞飘荡;《国旗国旗真美丽》的歌声,优美亲切,给人力量。

程序九:土星光环 模拟宇宙微粒排斥与吸引的旋转运动,在屏幕上展现了奇异灿烂的土星光环及土星本身的生成过程。

程序十:星云变幻 如同宇宙中进行不规则运动的网状星云,运动物质以光条形式从几个不同方位同

时交织叠加,逐步弥漫膨胀而演变成一幅色彩纷呈,结构奇特的星云变幻图。

本文所述程序可作为 BASIC 子程序,套用,将它们插入到应用程序中需改变屏幕画面的地方。

程序一

```
10 CLS;KEY OFF;SCREEN 2,0
20 I=3;Y=0;FOR J=0 TO 37;IF 6(I THEN I=0
30 PLAY“mb t250 o4 i24fcd”
40 FOR K=0 TO 300;NEXT K;I=I+1
50 LINE(9,Y)-(630,Y+10),I,BF;Y=Y+12
60 LINE(9,Y-2)-(630,Y),0,BF;NEXT J
```

程序二

```
10 CLS;KEY OFF;SCREEN 2,0;I=4
20 X=0;FOR J=0 TO 19;IF 7(I THEN I=I-7
30 LINE(X,1)-(X+15,450),I,BF
40 LINE(640-X,1)-(625-X,450),I,BF;I=I+1
50 X=X+16;PLAY“mb t180 o3 i24eg”;NEXT J
60 FOR X=-7000 TO 0;NEXT X;FOR J=0 TO 20
70 LINE(320+X,1)-(305+X,450),0,BF
80 LINE(320-X,1)-(305-X,450),0,BF
90 X=X+16;PLAY“mb t180 o4 i12ac”;NEXT J
```

程序三

```
10 CLS;KEY OFF;SCREEN 2,0
20 X=635;Y=0;FOR J=1 TO 28;LINE(5,Y)-(X,Y),
3
30 LINE(5,Y+90)-(X,Y+90),4;LINE(5,Y+180)-
(X,Y+180),5
40 LINE(5,Y+270)-(X,Y+270),6;LINE(5,Y+360)-
(X,Y+360),2
50 PLAY“mb t150 o4 i24ef”;Y=Y+3;NEXT J
```

程序四

```
10 CLS;KEY OFF;SCREEN 2,0
20 I=0;FOR J=0 TO 120;IF 15(I THEN I=1
30 LINE-(RND*639,RND*450),I
40 LINE(RND*777,RND*444)-(RND+999,RND-
333),I
50 I=I+1;PLAY“mb t150 o4 i12ab”;NEXT J
```

程序五

```
10 CLS;KEY OFF;SCREEN 2,0
20 X=40;Y=-1000;FOR J=0 TO 47
```



```

30 LINE(X,Y)-(640-X,450-Y),4,Y=Y+50
40 PLAY"mb t150 o4 i12cd";NEXT J
50 X=100;Y=-1000;FOR J=0 TO 47
60 LINE(X,Y)-(640-X,450-Y),6,Y=Y+50
70 PLAY"mb t150 o4 i12gf";NEXT J

```

程序六

```

10 CLS;KEY OFF;SCREEN 2,0,Y=25
20 FOR J=3 TO 83;LINE(620,450-Y)-(20,Y),J
30 Y=Y+5;PLAY"mb t180 o4 i12d";NEXT J
40 CIRCLE(320,225),150,4,,6.28,6
50 PLAY"mb t120 o2 i12be-fag-d-"
60 Y=-70;PAINT(320,135),4
70 FOR J=0 TO 59;K=INT(J/12+2)
80 LINE(200,450-Y)-(440,Y),K,Y=Y+10
90 PLAY"mb t180 o4 i24f";NEXT J

```

程序七

```

10 CLS;KEY OFF;SCREEN 2,0
20 LINE(0,0)-(640,450),3,BF;X=3
30 FOR J=0 TO 26;CIRCLE(320,225),X,7,... 3
40 PLAY"mb t150 o3 i12dt";Y=X+6
50 CIRCLE(320,225),Y,1,... 3;X=X+12
60 IF J-4*INT(J/4)=0 THEN 90
70 IF J-4*INT(J/4)=2 THEN 100
80 PAINT(320,225),7;GOTO 110
90 PAINT(320,225),3;GOTO 110
100 PAINT(320,225),1
110 PLAY"mb t120 o4 i24gae"
120 CIRCLE(320,225),J;NEXT J

```

程序八

```

10 CLS;KEY OFF;SCREEN 2,0;P=6.28
20 PLAY"mb t120 o3i4gege04co3al2g 14ececge12d"
30 DRAW"BM-320,-224;C7 E10 F10 G10 H10"
40 CIRCLE(14,27),6,7,... 3,P;PAINT(14,27),7
50 LINE(9,39)-(19,450),6,BF
60 LINE(20,42)-(20,432),4;PLAY"i3el6dl4cdegl2a"
70 LINE(20,42)-(320,62),4
80 LINE(20,432)-(320,450),4
90 LINE(320,62)-(630,32),4
100 LINE(320,450)-(630,420),4
110 LINE(630,30)-(600,225),4
120 LINE(600,225)-(630,420),4;PAINT(30,43),4
130 PLAY"i8ageal2gl4gl8del2c"
140 LINE(18,39)-(605,60),6
150 LINE(605,55)-(635,65),2,BF

```

程序九

```

10 CLS;KEY OFF;SCREEN 2,0
20 X=RND*110;Y=RND*255;FOR J=5 TO 31
30 CIRCLE(320,225),X,J,,6.28,.7
40 X=RND*318;PLAY"mb t150 o4 i12dae"
50 CIRCLE(320,225),Y,J,,6.28,.7
60 Y=RND*193;NEXT J;PAINT(320,225),6

```

```
70 PLAY"mb t150 o2 i12cdefilg-l8a"
```

程序十

```

10 CLS;KEY OFF;SCREEN 2,0,X=10
20 FOR J=0 TO 80;LINE(X,X)-(X-640,X-450),6
30 LINE(640-X,450-X)-(X,X),4
40 LINE(X,900-X)-(640-X,X-450),3
50 LINE(320+X,X)-(640-X,X-225),5
60 LINE(X-400,X)-(X,X-600),2,X=X+10
70 PLAY"mb t150 o2 i24agd";NEXT J

```

技巧 二 则

广空 汉神

一、怎样使您的文件隐含或显示？

亲爱的读者，您大概会有这样一种感受吧：每当您编写一个程序或编辑一篇信息文件时，总想使自己的成果或信息文件不被其它人操作（如列文件目录<DIR>、查看文件内容<TYPE>以及拷贝文件<COPY>等）。为了达到此目的，请您不妨试试把文件隐含起来，也许会收到意想不到的效果。运行下面一小段程序就可以把文件隐含或显示。此程序在 IBM-PC/XT 机上运行通过。

说明：

一、本程序是用汇编语言编写（文件名 HAN.ASM），因此，首先必须经过宏汇编（MASM）和连接程序（LINK）后，产生 HAN.EXE 可执行文件，才能直接运行。

二、程序的 12 行至 17 行是提示“请输入您的文件名”。输入您要隐含或显示的文件名后，打回车；第 22 行至 26 行显示“是隐含还是显示”。如要隐含打“Y”，显示打“N”。

三、运行此程序，相当于置一个隐含/显示软开关。

```

1:CODE          SEGMENT PUBLIC
2:              ASSUME CS,CODE,DS,CODE
3:              .ORG 100H
4:START:        JMP BEGIN
5:BUFFER        DB 15
6:              DB ?
7:              DB 15 DUP(?)
8:FILENAME      DB 'PLEASE INPUT YOUR FILE NAME,
               $'
9:HIDEDISP      DB 0DH,0AH,'HIDE ENTER "Y", DIS
               PLAY "N", $'
10:BEGIN:       MOV AX,CS
11:             MOV DS,AX
12:             MOV DX,OFFSET FILENAME
13:             MOV AH,09H
14:             INT 21H
15:             MOV AH,0AH

```

```

16:      MOV DX,OFFSET BUFFER
17:      INT 21H
18:      MOV BL,BUFFER+1
19:      ADD BL,2
20:      MOV BH,0
21:      MOV BUFFER[BX],0
22:      MOV AH,09H
23:      MOV DX,OFFSET HIDEISP
24:      INT 21H
25:      MOV AH,01
26:      INT 21H
27:      CMP AL,59H
28:      JZ A1
29:      CMP AL,79H
30:      JZ A1
31:      MOV CX,00H
32:      JMP A2
33:A1:    MOV CX,02H
34:A2:    MOV AX,4301H
35:      MOV DX,OFFSET BUFFER+2
36:      INT 21H
37:      MOV AH,4CH
38:      INT 21H
39:CODE   ENDS
40:      END START

```

二、怎样给磁盘加上汉字卷标？

我国于1983年推出专为IBM-PC机开发的汉字操作系统CCDOS,使得IBM-PC及其兼容机具备了汉字功能,从而在我国得到广泛使用。通常在机上由键盘输入汉字,并在屏幕上显示出来。实际上这是CCDOS把由键盘输入的汉字输入码(如区位码、拼音码等)转换成汉字机内码(如异形国标码),并将汉字的机内码转换成汉字字模输出显示的过程。汉字的机内码是用国标码最高位置1表示(称为异形国标码),以区别于一般的ASCII码,故一个汉字的机内码用两字节表示。

为了得到汉字卷标,首先我们必须得到汉字的机内码。一般情况下我们采用汉字编辑软件WS编辑一份所需的汉字信息文件,然后用DEBUG把汉字信息文件调入内存。用D命令显示出来的数据即为汉字的机内码(每两个字节为一个汉字机内码)。

修改磁盘卷标信息的操作,可分以下三种情况:

1. 磁盘在格式化时用FORMAT A: /V

```

C>DEBUG
-L100 0 5 1
-E100
CE C4 CF C8 C9 FA 30 31 C5 CC 20 08 20 20 20 20
-W100 0 5 1
-Q

```

2. 磁盘在格式化时用FORMAT A: /S/V

```

C>DEBUG
-L100 0 5 1
-E160

```

```

CE C4 CF C8 C9 FA 30 31 C5 CC 20 08 20 20 20 20
-W100 0 5 1
-Q

```

3. 磁盘在格式化时用FORMAT A:

```

C>DEBUG
-L100 0 5 7
-D

```

此时用D命令来查看目录,如果盘上的文件数目不超过112个,我们就在最后一个文件目录的地址后面加上卷标。假设盘上有50个文件,那么第51个文件目录的地址为:0B00

```
-E0B00
```

```
CE C4 CF C8 C9 FA 30 31 C5 CC 20 20 08 20 20 20 20
```

```
-W100 0 5 7
```

```
-Q
```

用DIR查看目录时,就显示:

```
C>DIR A:
```

Volume in drive A is 文先生 01 盘

Directory of A:

COMMAND	COM	17664	3-08-83	12:00P
FORMAT	COM	6016	3-08-83	12:00P
CHKDSK	COM	6400	3-08-83	12:00P
DEBUG	COM	11904	3-08-83	12:00P

4 File(s) 318464 bytes free

如果是硬盘,上述的L、W命令分别改为:L100 2 11 10, W100 2 11 10,其余步骤不变。

利用DEBUG进行反汇编存盘的方法

浙江电子工业学校 金林樵

用DEBUG的U命令,可方便地反汇编出源程序的清单,但它只能显示和打印,却不能进行反汇编存盘操作,不便于程序的修改和再加工。下面介绍的方法,将为你提供这方面的帮助。

本方法利用DOS系统的输入输出重新定向功能,通过可变元的批处理程序(见程序一),将需反汇编源程序的文件名(包括扩展名)、反汇编起始地址、终止地址及一个退出DEBUG的q命令输出到T.TXT文件中,然后调用DEBUG将需汇编程序从起始地址到终止地址的反汇编清单,通过输出重新定向功能输出到输出源文件名存盘,以后就可利用字处理软件对其输出文件进行修改了,使用十分方便。使用方法如下:

A>UASM 需反汇编文件名 起始地址 终止地址 输出源文件名

本方法在IBM PC/XT机上调式通过。

A>type uasm.bat

ECHO OFF
ECHO U %2 %3 > T.TXT
ECHO Q >> T.TXT

DEBUG %1 < T.TXT > %4
DEL T.TXT

90 年全国计算机通讯赛题分析

西安交大附中 汪五一

1990 年全国青少年计算机通讯赛的试题是一道模拟魔方转动的题目。下面就该题的算法作一探讨(试题详载于本刊 91 年第 1 期,因太长,此处从略)。

综观全题,主要解决两大方面的问题:1. 输入魔方的初始状态,并进行判错处理。2. 若魔方的初态合理,则要求以最少步骤达到目标状态。

判错处理应包括有检查所输入的初始状态是否包含 1~9, A~I, a~i 27 个字符。另外根据魔方转动的特点,还应检查 E 小块是否在魔方的中心,即应在题目给定的编号点 14 上;1、3、7、9、a、c、g 和 i 这 8 小块应落在魔方的 8 个角上的位置,即编号点 1、3、7、9、19、21、25、27 上;5、B、D、F、H 和 e 这 6 小块应在魔方 6 个可见面的中心,即在编号点 5、11、13、15、17 和 23 位置上;其余的各小块应落在魔方其它点位置上。凡上述判断中只要有一个不符合要求,则应作出错处理。这一段检查出错处理程序较简单。

魔方从初始状态到目标状态的移动过程,实质就是从初始态不断按题目要求进行转动搜索,最终找到结果。由于魔方从当前状态转到下一状态有 18 种不同的转法,因此可直接采用广度优先搜索方法。把初始状态作为一个根结点,往下生成 18 个结点,再由这 18 个结点出发,再分别生成其它结点……这样边生成结点,边检查所生成的结点是否到达目标状态,直到目标态为止。这种盲目搜索方法必须把所生成的结点保存在数组中。若从初始状态到目标状态最少要转 N 次,则最少需要生成 $18^{N-1}+1$ 个结点,最多则要生成 18^N 个结点才能达到目标。这对 APPLE II 类微机,从初始态距目标态的步数在一、二步还可行。但三步以上的搜索,就很快由于记录的结点过多,而造成溢出的错误。

本题也可用人工智能中的 A* 算法来处理(1990 年第二届国际信息学奥林匹克竞赛中,我国 4 名选手均采用该算法求解第一道赛题,有关算法可参见《学生计算机世界》报总 143 期或其它有关人工智能方面的书籍)。这种算法主要采用了一个启发函数来估价哪个结点最有希望达到目标,从而优先扩展该结点。因此它仍需要建立数组来记录已扩展的结点。另外这个启发函数要构造得比较准确,才能节约内存,以最快速度达到目标。但对本题由于小方块的转动不是独立的,转动某小块将涉及到该片其它小块也跟着一起转动,要建立一个较好的启发函数比较困难。正由于这个原因,就不能准确地优先发展能到达目标的结点。故用 A* 算法对本题所能解决的魔方移动步数也极为有限。

其实本题可用解答树的方法来处理。其最大特点

是可用最少的内存空间来处理任意步数(原则上可以这么说)的魔方转动,但它也有不足之处,就是这种方法是时间换取空间的。有关解答树在数据结构书籍中均有介绍,现简述如下:

求解解答树的过程是:1. 构造解答树;2. 系统地检查解答树的结点,看它们是否对应于解答位置(所谓解答位置就是符合题目要求的结点)。

为了加快第 2 步中检查结点的速度,通常约定:任何一棵子树,只要它的根不满足条件,它就不会含有解答位置,因而不必继续考虑它,并且在第 1 步构造解答树时就不构造这种子树。

由于所构造的解答树的规模通常很大,因此把第 2 步对结点的检查与第一步构造解答树的工作并行进行,在逐步构造和检查解答树的过程中,对已处理过的结点,若以后不会再用,就不必保留它。

如果逐步构造和检查解答树的工作是从左到右逐步进行,那末一般来说,为检查长度为 n 的树枝 T,只要保留 n 个结点就够了。在 T 的左边的一切结点都已检查过了;T 的右边的一切结点尚未产生。按这种方式得到一条到达树叶的树枝 T 后,得到 T 右边的下一个树枝的方法是:沿 T 回溯到第一个尚未用过的向右的分支点,并由此向右走一步,随后仍然向左走,直至到达树叶为止。

因为本题要求以最少的步数转到目标状态,所以构造树的层数应以 $N=2$ 开始搜索。若 1 步转动不能达到目标,则应构造 $N=3$ 的树,即做 2 步转动试探,以此类推,直到符合目标要求为止。构造检查本题解答树的程序逻辑及 BASIC 程序如下:

1. 输入魔方初始状态作为根结点。
2. 设置生成树层数 N 初值为 2。
3. 设置堆栈指针 I1 为 1。
4. 栈指针 I1 等于树层数 N 时转去 12。
5. 置 S=0。
6. $S=S+1$ 生成下一个扩展结点数。
7. 若当前结点已扩展了 18 个结点则转 12。
8. 将当前结点状态及扩展的枝叶号 S 存入栈,栈指针 I1 加 1。
9. 根据 S 值扩展生成一个结点。
10. 若生成的新结点未达到目标态则返回 4。
11. 完成魔方转动。
12. 栈指针 I1 减 1。
13. 若已下越栈底,生成树层次 N 加 1 后返回 3。
14. 从栈中退出一个结点状态及 S 值,返回 6。

```

1 CLEAR
5 DIM G$(50),G(50),E(18),D(9,5),F(10,2),C$(12),REM 数组 G$,G 作为堆栈,存放魔方转动的各状态及转动的方向,数组 E,D,F 为魔方转动的位置数据,数组 C$ 为存放判错所用数据
10 INPUT "INPUT A STRING: ";A$;IF LEN(A$)<>27 THEN 10;REM 判断输入的初始态字符个数是否为 27 个
20 IF MID$(A$,14,1)<>"E" THEN PRINT "WRONG!";GOTO 1;REM 判断 E 是否在魔方中心
25 FOR Q=1 TO 8;READ C$(Q);NEXT Q;DATA 1,3,7,9,a,c,g,i
30 FOR I=1 TO 8;READ J;R$=MID$(A$,J,1)
35 FOR Q=1 TO 8;IF R$=C$(Q) THEN C$(Q)="";GOTO 45
40 NEXT Q;PRINT "WRONG!";GOTO 1
45 NEXT I;DATA 1,3,7,9,19,21,25,27;REM 判断 1,3,7,9,a,c,g,i 是否在魔方的 8 个角上
50 FOR Q=1 TO 6;READ C$(Q);NEXT Q;DATA 5,B,D,F,H,e
55 FOR I=1 TO 6;READ J;R$=MID$(A$,J,1)
60 FOR Q=1 TO 8;IF R$=C$(Q) THEN C$(Q)="";GOTO 75
65 NEXT Q;PRINT "WRONG!";GOTO 1
75 NEXT I;DATA 5,11,13,15,17,23;REM 判断 5,B,D,F,H,e 是否在魔方可见面的中心
80 FOR Q=1 TO 12;READ C$(Q);NEXT Q;DATA 2,4,6,8,A,C,G,I,b,d,f,h
90 FOR I=1 TO 12;READ J;R$=MID$(A$,J,1)
100 FOR Q=1 TO 12;IF R$=C$(Q) THEN C$(Q)="";GOTO 110
105 NEXT Q;PRINT "WRONG!";GOTO 1
110 NEXT I;DATA 2,4,6,8,10,12,16,18,20,22,24,26
111 REM 以上同时判断输入小方块的标志是否正确
120 B$="123456789ABCDEFGHIabcdehghi";REM B$ 存放目标状态
130 FOR I=0 TO 5;FOR J=1 TO 9;READ D(J,I);NEXT J;NEXT I
140 DATA 7,4,1,8,5,2,9,6,3,3,6,9,2,5,8,1,4,7
150 DATA 19,10,1,20,11,2,21,12,3,3,12,21,2,11,20,1,10,19
160 DATA 7,16,25,4,13,22,1,10,19,19,10,1,22,13,4,25,16,7
170 FOR I=1 TO 18;READ E(I);NEXT I;DATA 0,9,18,0,9,18,0,3,6,0,3,6,0,1,2,0,1,2
180 FOR I=0 TO 2;FOR J=1 TO 9;READ F(J,I);NEXT J;NEXT I;DATA 1,2,3,4,5,6,7,8,9,1,2,3,10,11,12,19,20,21,1,4,7,10,13,16,19,22,25;REM 数组 D,E,F 读入魔方转动的位置数据
200 IF A$=B$ THEN PRINT "RIGHT!";GO TO 380;REM 判断初始态是否为目标态
210 N=2;REM 设置树层数 N 的初值
220 I1=1;REM 设置栈指针初值

```

240 IF I1=N THEN 400;REM 栈指针等于树层数 N,则转 400 进行回溯处理

250 S=0

260 S=S+1;IF S>18 THEN 400;REM 判断当前结点是否已扩展了 18 个树叶

270 GOSUB 2000;I1=I1+1;REM 当前结点及将扩展的叶子号 S 入栈,栈指针加 1

275 J=1;R\$=""

280 FOR I=1 TO 27

285 R=F(J,INT((S-1)/6))+E(S)

290 IF I=R THEN RE\$=MID\$(A\$,D(J,INT((S-1)/3))+E(S),1);J=J+1;GOTO 305

300 RE\$=MID\$(A\$,I,1)

305 R\$=R\$+RE\$;NEXT I;REM 魔方某片实现转动

310 A\$=R\$

315 IF A\$<>B\$ THEN 240;REM 魔方未到目标,继续处理

320 GOSUB 2000;REM 魔方旋转成功,打印处理

329 REM 打印魔方转动过程

330 FOR Q=1 TO I1

340 IF Q=1 THEN PRINT "START: ";GOTO 360

350 PRINT "STEP";Q-1;" ";

360 PRINT G\$(Q)

370 NEXT Q;PRINT "END"

380 END

399 REM 回溯处理

400 I1=I1-1;IF I1=0 THEN N=N+1;GOTO 220;REM 栈指针减 1,判断是否到栈底

410 A\$=G\$(I1);S=G(I1)

420 GOTO 260;REM 结点状态及叶子号 S 出栈,转 260,扩展其它枝叶

1999 REM 结点及 S 入栈子程序

2000 G\$(I1)=A\$;G(I1)=S;RETURN

]RUN

INPUT A STRING;3G12FCahid58HEB4efg69IDA7bc

START;3G12FCahid58HEB4efg69IDA7bc

STEP1;36125CaGidD8HEB4Ffgb9IeA7hc

STEP2;369258aG1dDAHEB4FCgbcIef7hi

STEP3;a23G56189dDAHEB4FCgbcIef7hi

STEP4;123456789GDAHEBIFCabcdehghi

STEP5;123456789ABCDEFGHIabcdehghi

END

但这个程序仍有不足之处,需要大量的搜索时间。从以上分析可看出其探索时间是按几何级数递增的。若有一个需转 n 步才能到达目标的初始态,程序最多运行时间为 $T=t(1+18+18^2+\dots+18^n)=t(\frac{18^{n+1}-1}{18-1})$ 。其中 t 为从一个结点出发,生成 18 个树叶所需的时间。本程序在 APPLE 机上测试 t 约为 20 秒。这样一个需转动五步才能达到目标的初始状态,最多需 $20(\frac{18^5-1}{18-1})=2223021$ 秒 ≈ 26 天才能完成。

改进措施设想:

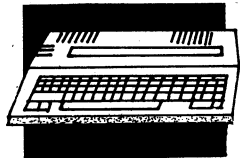
1. 从初态开始,除第一步有 18 种转动方法外,以后每转一步,从该步出发,虽也有 18 种转法,但其中有

一转法又转回到原状态。所以除第一步需扩展 18 个枝叶外,以后的结点只需扩展 17 个枝叶。

2. 由于解答树是按前序生成及搜索的,因此根据题目要求应先生成一棵层次 N 为 2 的树,看其能否达到目标。若不成,然后重新生成一棵层次 N 为 3 的树……这样就造成了搜索时间是按几何级数递增。若能构造一个估价函数,大概估价出魔方需 N 步达到目

标,那么一开始就向 $N+1$ 层树方向生成与搜索。若所生成的 $N+1$ 层树仍未达到目标,就再用估价函数,估价出所生成的树叶离目标最少还需 n 步,下次就生成一棵 $N+1+n$ 层树,再进行搜索。经这样处理也可以加快程序执行速度。

以上讨论不一定正确,希望各位读者提出更为优良的算法。



CEC-I 全 功 能 造 字

成都市成人教育学院 傅叔平

学习机之友

CEC-I 的字库包含了国标一、二级汉字 6763 个和一些外文字母和常用符号,通常情况下已够使用。有时候遇到一些冷僻的汉字或在有的场合需用一些特殊符号,这就需要造字了。但 CEC-I 的汉字管理程序和汉字字库都是固化的,不能改变,这就给造字(特别是在打印输出中造字)带来了较大的困难。本文要解决的问题是要在显示和打印两种输出方式中均能造字并且不影响原有的功能。

一、实现方法

1. 显示造字

CEC-I 的汉字显示用的是高分辨率图形页。汉字系统的输出程序根据汉字或字符的代码在汉字库中查到点阵,再将点阵按高分率图形屏幕上点与其映射地址的对应关系,将点阵数据置入合适的地址,便能在屏幕上显示出汉字来。问题是固化的汉字系统的输出程序只会从系统字库中查找点阵,没法加以改变让它接受用户提供的点阵数据。所以输出自造的字时不能直接调用汉字系统的输出程序,可采用如下较为简单的方法来解决:

在屏幕上显示用户造的字时不用汉字系统的输出程序,而直接调用系统内部的显示子程序来显示,这样就可以由用户程序将所需字的点阵直接送到系统的点阵缓冲区供显示之用。

2. 打印造字

当考虑还要在打印输出中造字时问题就复杂得多了,因为九针打印机打印 16×16 点阵的汉字时,不能象显示输出那样以一个汉字或字符为单位,必须以一个打印行为单位。造的字点阵必须排在打印队列中其它字的点阵之间,还要考虑九针打印机打印一行汉字时必须分为上半行和下半行两次进行,还有字型、字间距、行间距等等一系列问题。

为了能将造的字插入其它的字中间打印出来,在显示时每显示一个字便将该字(包括造的字)的代码排入一个输出队列,打印时有如下几个步骤:

①根据输出队列的长度、字型和字间距计算出打印所需的位图形数据的个数,并将打印机置成图形打印状态;

②到输出队列中依次取代码,若代码属国标一、二级汉字或 ASCII 字符,仍到汉字系统的固化字库中取点阵,若是用户造字则到用户字库中取点阵;

③取完输出队列的点阵后打印出上半行;

④控制行距为 0 换行;

⑤类似①~③的步骤打印出下半行;

⑥按行间距走纸。

从上述看要作的工作是很多的,特别是打印输出,和重写一个汉字打印驱动程序也差不了多少。明智的作法是利用固化的汉字管理程序这个丰富的资源,直接调用其中有用的子程序给我们提供便利,这样可以减少编程工作量并使程序大大地缩短。

二、程序注释

程序 1 是按上述方法编制的一个功能较完善的造字程序,它既能在屏幕上显示用户造的字,也能打印出包括用户造的字在内的输出内容,其字型、字距、行距等仍用原来的命令决定。

该程序调用的汉字系统内部子程序主要有:

\$C392——内码到国码转换;

\$ECCD——查找系统字库中的点阵,调用前须将汉字的国标码高字节放入 \$D6 单元,低字节放入 \$D7 单元,若是 ASCII 字符则 \$D6 单元置 0, \$D7 单元放它的 ASCII 码;

\$F0B5——显示一个汉字,显示前须将汉字的 16×16 点阵数据放入 \$94D0 开始的点阵缓冲区;

\$F0AD——显示 16×16 点阵的左半边字形。当显示 ASCII 字符时,就要用到该子程序来显示,若用前一个显示子程序则字符后将跟一个空格;

\$F0E5——根据一个打印行的字数及字型和字间距计算出一个打印行的位图形数据个数;

\$F630——将打印机设置成位图形打印方式;

\$F66B——打印完上半行汉字后走纸。

程序 1 中 \$6091~\$60B9 是显示子程序,它在显示的同时也把字的代码排入输出队列。\$60BA~\$610A 是打印子程序。\$610B~\$6145 是取点阵子程序。为统一起见,用户造的字被定义为区位码表中的一个未定义区——10 区中的字。该子程序首先判断区

码是否 10(相应的国标码 \$ 2A),若是则到用户字库区查找点阵。在将点阵送入点阵缓冲区之前还有一个工作要做,就是为了和系统字库的点阵数据规律一致,须将点阵数据的每个字节的各位左右颠倒,这由 \$ 6146 ~ \$ 6152 的子程序来完成。

造字程序由 & 命令的入口进入,程序中 \$ 6000 ~ \$ 6034 即是解释执行新定义的几个 & 命令。

三、应用

读者可在监控状态下输入程序 1,然后存盘待用。使用时应先将程序 1 和用户字库调入内存。用户字库起始地址为 \$ 7000,每 32 个字节为一个字的点阵数据,即 \$ 7000 ~ \$ 701F 为用户定义的 1 号字, \$ 7020 ~ \$ 703F 为用户定义的 2 号字,依此类推。然后设置 & 命令的入口向量,使其指向 \$ 6000。

造字程序由如下新定义的 BASIC 语句来调用:

①&C——清除打印队列,在初次打印之前或打印新内容之前须用此语句;

②&“<汉字串>”——显示汉字串前将其代码加入打印队列;

③&n——显示第 n 个用户造字前将其代码加入打印队列,n 对应用户字库区的第 n 个自造字;

④&p——将②、③两种命令形成的输出队列打印出来,和 PRINT 语句一样,字型、字距行距等的确定仍用原来的命令,若 1659 单元的值 0 则不打印,此时此命令无效。在未用命令①清除打印队列之前,可多次使用此命令重复打印同一内容。

从上述可知,若要打印自造字,必须先显示再打印;若只显示则不用打印命令。另外,PRINT 语句的功能照旧,所以,若打印行中没有用户造的字,仍用 PRINT 语句输出即可。当然,PRINT 语句也可和这些新定义的 BASIC 语句混合使用。

程序 2 是调用造字程序的实例,运行结果中第一行字是由 50 行的 PRINT 语句输出的,第二行字是由 60 行输出的,其中第四个字(木旁右边一个童字)不但国标一、二级汉字中没有,就连《新华字典》上也查不到。第三行字中用 16×16 点阵造了一个符号,是由 70 行输出的。

程序 2 的 20 行中“CREATE. CHN. OBJ0”是程序 1 在磁盘上的文件名;30 行中的“CREATE. CHN. LIB”是用户字库在磁盘上的文件名。用户字库如下:

```
7000- 10 80 10 48 17 FC 11 10
7008- FC A4 17 FE 30 08 3B FC
7010- 52 48 53 F8 92 48 13 F8
7018- 10 40 13 F8 10 40 17 FE
7020- 01 00 01 00 01 00 01 00
7028- 03 80 03 80 03 80 07 C0
7030- 07 C0 27 C8 2F E8 2F E8
7038- 3F F8 3F F8 70 1C 80 02
```

程序 1

```
6000- A0 00 B1 B8 C9 43 D0 07
6008- A9 00 8D 53 61 F0 74 C9
```

```
6010- 22 F0 22 C9 50 F0 52 C9
6018- 30 90 6C C9 3A B0 68 20
6020- 7B DD 20 FB E6 8A 18 69
6028- 20 85 D7 A9 2A 85 D6 20
6030- 91 60 4C 86 60 20 8A 60
6038- A0 00 B1 B8 C9 22 F0 43
6040- C9 7F D0 19 20 8A 60 B1
6048- B8 20 92 C3 85 D6 20 8A
6050- 60 A0 00 B1 B8 20 92 C3
6058- 85 D7 4C 63 60 85 D7 A9
6060- 00 85 D6 20 91 60 4C 35
6068- 60 AD 7B 06 F0 03 20 BA
6070- 60 AD 7B 06 48 A9 00 8D
6078- 7B 06 A9 0D 20 2B C3 68
6080- 8D 7B 06 20 8A 60 60 4C
6088- C9 DE E6 B8 D0 02 E6 B9
6090- 60 AE 53 61 A5 D6 9D 55
6098- 61 E8 A5 D7 9D 55 61 E8
60A0- 8E 53 61 20 AB C3 20 0B
60A8- 61 A5 D6 F0 06 20 B5 F0
60B0- 4C B6 60 20 AD F0 20 B9
60B8- C3 60 AD 7B 06 F0 0C 20
60C0- AB C3 20 CC 60 20 8F F6
60C8- 20 B9 C3 60 A9 00 20 DA
60D0- 60 20 6B F6 A9 01 20 DA
60D8- 60 60 8D F5 94 AD 53 61
60E0- F0 28 8D F6 94 20 E0 F5
60E8- 20 30 F6 A2 00 EC 53 61
60F0- B0 18 BD 55 61 85 D6 E8
60F8- BD 55 61 85 D7 8A 48 20
6100- 0B 61 20 B7 F5 68 AA E8
6108- D0 E3 60 A5 D6 C9 2A F0
6110- 06 20 CD EC 4C 45 61 A9
6118- 00 85 06 A9 70 85 07 A5
6120- D7 38 E9 20 AA CA F0 10
6128- A5 06 18 69 20 85 06 A5
6130- 07 69 00 85 07 4C 25 61
6138- A0 1F B1 06 20 46 61 99
6140- D0 94 88 10 F5 60 A2 08
6148- 0A 6E 54 61 CA D0 F9 AD
6150- 54 61 60 A0 A0 C4
```

程序 2

```
1 REM 造字举例
5 B$ = CHR$(13) + CHR$(4) + "BLOAD"
10 PRINT B$; "CREATE. CHN. OBJ0"
20 PRINT B$; "CREATE. CHN. LIB, A $ 7000"
30 POKE 1013, 76; POKE 1014, 0; POKE 1015, 96
40 POKE 1659, 6; POKE 1787, 2; POKE 1915, 6
50 PRINT "造字举例:"
60 & C; & "成都梓"; & 1; & "桥街"; & P
70 & C; & "您能打印出符号"; & 2; & "吗?"; & P
90 POKE 1659, 0
99 DEL 5, 20; END
```

梵塔问题和奇数幻方的一行程序

北京大学 物理系 陈纲

编辑同志：

我是一名北大新生，是贵刊今年的新读者。感谢你们办出这么好的杂志，它已成了我学计算机的好帮手。现在冒昧地寄上两个一程序，望多指教。

第一个程序解决了梵塔问题（即金片移动问题）。梵塔问题是计算机算法学中的一个老问题，历来是用递归方法解决的，而众所周知 BASIC 语言不易实现递归，即使子程序自我调用的方法实现了，若金片太多也会造成堆栈溢出而报错，且不可能写成一程序。本程序采用了一种非传统的算法，亦即将最小的金片按 $A \rightarrow B, B \rightarrow C, C \rightarrow A$ ，（A、B、C 表示三根金片）的顺序移动，每移一步，比较一下剩下两根金片顶端的金片，能怎么移就怎么移，如此反复，移动 $2^N - 1$ 步（N 表示金片数）后必能将 N 片金片都移到另一根金片上。程序中用数组 X(N,2) 存放三根针上的金片号码，用数组 A(2) 存放三根针上的金片数目，并用逻辑代数的技巧，避免了使用 IF.....THEN..... 语句，使此算法能在一行中实现。

第二个程序是一个打印奇数幻方程序。它采用了可能大家很熟悉的构造奇数幻方的数学方法，只是用了一些 SGN 函数以避免使用 IF.....THEN..... 语句，使之能写在一行中。两个程序分别用两种不同的方法替代了 IF.....THEN..... 语句，希望能对读者编写一行程序有些启发。

程序一

```
5 INPUT N;DIM X(N,2);FOR I=1 TO N,X(I,0)=I;
NEXT A(0)=N;FOR I=1 TO 2^N-1,F=E+1-(E)1)*
3;A=X(A(E),E);B=X(A(F),F);M=(A)B)*E+(A)B
```

```
* F;F=F+E-M;PRINT("I" move from " CHR$(65+
M)" to "CHR$(65+F);A(F)=A(F)+1;X(A(F),F)=X
(A(M),M);A(M)=A(M)-1;E=E+2-(E)0)*3;NEXT
```

]RUN

? 3

- (1) move from A to B
- (2) move from A to C
- (3) move from B to C
- (4) move from A to B
- (5) move from C to A
- (6) move from C to B
- (7) move from A to B

程序二

```
5 INPUT P;DIM P(P,P),X=(P+1)/2,Y=P;FOR I=1
TO P * P,P(X,Y)=I;M=X * SGN(P-X)+1;N=Y *
SGN(P-Y)+1;A=SGN(P(M,N));X=M+A * (X-M);
Y=N+A * (Y-1-N);NEXT;FOR I=1 TO P;FOR J=1 TO
P,PRINT RIGHT$( " " +STR$(P(J,I)),4);,NEXT;
PRINT,NEXT
```

]RUN

? 5

11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

原稿还附有梵塔 N=5，幻方 P=31 的运行结果，限于篇幅从略 编者。

（上接第 20 页）因为对顺序文件进行读写时，系统就把读写指针从该顺序文件的首字节向后移动了 p+1 个字节，并把这个位置作为顺序文件的首字节，前面的字节就无法读出或写入了，这样一来，读出的内容肯定不正确，写入的内容亦不能正确读出。

修正办法

修改 DOS 的命令执行过程，使系统先转去对 \$AA6E，\$AA6F 进行初始化置入 0，以确保消除 R_p 参数的遗留影响，修改步骤如下：

```
* A01B,20 C0 B6 JSR $B6C0
* B6C0,A2 00 LDX # $00
* B6C2,8E 6E AA STX $AA6E
* B6C5,8E 6F AA STX $AA6F
* B6C8,0E 5F AA ASL $AA5F
* B6CB,60 RTS
```

然后用 INIT 命令格式化一张具有修正 DOS 的系统盘。经过修改后 DOS 系统就不会有问题了。

更正

本刊 1990 年 12 期《CEC 机汉字放大与缩小显示》一文中，由于笔者的疏忽，在程序三第 1000 行中遗漏了 POKE 8,A。正确的写法是：1000 A=2*LEN(A\$)/3+4; POKE 8,A; POKE 230,64; CALL-3086; VTAB1;PRINT A\$;RETURN。特更正，并向读者表示歉意。

苏州景范中学 赵旭

本刊 1991 年第 2 期《如何解决运行 FOXBASE 内存不够的问题》一文，由于本人不慎，致使第 34 行文字中的式子有误，应改为：

```
(141376 - 65536 * (INT (141376/65536))) =
2840H
```

，谨向读者道歉

湖北天门市 121#信箱 周日初

(2)ESC 0

设定行距为 1/8 英寸。

格式:〈行号〉PRINT CHR\$(27);“0”;

(3)ESC 1

设定行距为 7/72 英寸。

格式:〈行号〉PRINT CHR\$(27);“1”;

(4)ESC 2

设定行距为 1/6 英寸。

格式:〈行号〉PRINT CHR\$(27);“2”;

2. 页长控制码

ESC C+(n)

设定每页的长度,以行数为单位,每页最长不超过 127 行。

格式:〈行号〉PRINT CHR\$(27);“C”;CHR\$(n);

3. 打印机复原码

ESC @

这个控制码使打印机复位、初始化,即设定页始端,页长 66 行,行宽 80 字符等。

将 BASIC 程序变成 B 型文件并以 BRUN 运行

北京师范学院数学系 87(3)班 杨峰

《电子与电脑》杂志曾介绍过将 BASIC 程序以 B 型文件存盘的方法,其缺点是无法以 BRUN 运行,虽起到保密作用,但调盘运行则比较繁。这里介绍一种可以把 BASIC 程序以 B 型文件存盘并且可以 BRUN 运行的方法。

首先介绍一个重要的入口——\$A4FC,这是 DOS 运行 BASIC 程序即 RUN 的执行入口。我们可以在 BASIC 程序内存区(\$800)以前存入一段机器码程序,用以设定 BASIC 简单变量区及指针,然后 JMP \$A4FC 来运行 BASIC 程序。

详细操作过程如下:

1. 键入或调入一个 BASIC 程序到内存区

2. CALL-151 进入监控

3. AF.B0 查看程序尾指针,设 AF 内容为 LL,设 B0 内容为 HH,计算未来程序长度:

长度 = \$HHLL - \$0777

4. ESC @ 清屏,使光标回左上角

5. 键入以下机器码程序:

* 778:A9 LL 85 69 4C F8 7 N 7F8;

A9 HH 85 6A 4C FC A4

6. 用 BSAVE 存盘:BSAVE 〈文件名〉,A\$778,L\$长度

好,现在你的 BASIC 程序以 B 型文件存盘了并且你可以试试是否可以 BRUN 运行呢?

如果在 BASIC 程序中加入 CTRL-C 和 RESET 保护,则以 B 型文件存盘后,本身已经是被加密的了,因为在 \$800 以前的机器码处于文本显示区以外部分,不用特殊工具是无法看到其内容的。

APPLE DOS 的一处错误

重庆市职业技术教育中心 尹进渝

错误现象

在用 READ f[,R_p][,B_b]或 WRITE f[,R_p][,B_b]访问了随机文件后,若接着对顺序文件用 READ f[,B_b]或 WRITE f[,B_b]进行带 B_b 参数的读或写时,将使 READ 后的第一个 GET 或 INPUT 语句,WRITE 后的第一个 PRINT 主句不能正确按 B_b 参数指定的位置读写数据。

错误原因

在程序中访问文本文件时,必须先用 OPEN 命令打开文件,尔后读写。OPEN 命令格式为:

OPEN f[,V_v][,D_d][,S_s][,L_i]

i 表示记录长度,DOS 区分打开的文件是顺序的或是随机的,就是依靠参数项 L_i,系统内部并不区分对这两种文件的操作,而且在文件上不做任何标记。

当对文件进行访问时,如为写,则用 WRITE 命令确定文件写指针位置,格式为:

WRITE f[,R_p][,B_b]

若为读,则用 READ 命令确定文件读指针位置,格式为

READ f[,R_p][,B_b]

这两个命令既可对随机文件,亦可对顺序文件进行读写。读写随机文件时,命令中必须带 R_p 参数,R_p 指文件中的第 p 号记录。DOS 将把 R_p 参数值送入地址 \$AA6E, \$AA6F 中。这些地址里的值在下次被 R_p 参数项修改前一直有效,即对 p 号记录访问完毕后系统并不对 \$AA6E, \$AA6F 进行初始化(置 0)。当随后接着对顺序文件进行读写操作时,READ 或 WRITE 命令不能带 R_p 参数,此时系统将把这个顺序文件视为记录长度为 1,即 L 参数等于 1 的随机文件,并把读写指针定位于 \$AA6E, \$AA6F 所指定的“记录号”位置,再根据 R_p 参数确定字节数。问题就出在这个因使用随机文件后遗留下来的“记录号”上,(转 18 页)

ProDOS 系统盘的修改

廖 凯

编者按 与 APPLE DOS3.3 相比,ProDOS 的性能更为优越,并在许多方面接近 16 位微机的磁盘操作系统,但由于推出较晚,国内介绍、剖析的文献也少,多数中华机和 APPLE 机的用户对它尚了解不多,甚至不知如何操作应用。从本期开始,我们将连载系统介绍 ProDOS 的专文,并特邀北京铁路局的廖凯先生撰稿。

ProDOS 系统以其卓越的功能而受到 APPLE II 及 CEC-I 用户的青睐。由于原版 ProDOS 系统中有一识别机型的子程序,所以只能在 APPLE II 原装机上运行,要想在 APPLE II 兼容机及 CEC-I 上运行,就必须修改 ProDOS 系统。另外,ProDOS 系统只使用 35 磁道,而现在的驱动器已可以使用 40 磁道甚至更多磁道。因此,本人对三种版本的 ProDOS 系统,即 ProDOS V1.0.1、ProDOS V1.1.1 及 ProDOS 8 V1.5 进行了修改,使其系统具有管理 40 磁道或 44 磁道的功能,磁盘的逻辑块总数将达到 320 块或 352 块,大大地扩充了磁盘的容量,经修改过的 ProDOS 系统可以在 APPLE II 兼容机及 CEC-I 上运行。

具体修改如下:

一、系统文件 PRODOS 的修改

1、ProDOS V1.0.1 和 V1.1.1 的修改

在启动 ProDOS 时,由于不是原装机而锁死,此时按 CTRL-RESET 便进入监控状态,然后针对各版本键入以下数据:

V1.0.1 版 23D4;EA EA A9 40 EA EA

V1.1.1 版 2428;EA EA A9 40 EA EA

然后 2000G,启动 ProDOS 系统,再进入 BASIC 状态,键入 BLOAD PRODOS,TSYS,A \$ 2000,按 CALL-151 进入监控状态,然后针对各版本键入以下数据:

V1.0.1 版 23D4;EA EA A9 40 EA EA

265B;EA EA

520C;C9 40(40 道)或 C9 60(44 道)

V1.1.1 版 2428;EA EA A9 40 EA EA

269E;EA EA

56E0;EA EA C9 40(40 道)或 EA C9 60(44 道)

按 CTRL-C 回到 BASIC 状态,键入 BSAVE PRODOS,TSYS,A \$ 2000。若 PRODOS 已上锁,就先解锁再存盘。

2、ProDOS 8 V1.5 的修改 *

用具有磁道扇区编辑功能的软件,如 ZAP、DISK FIXER 或 COPY][PLUS 中的 SECTOR EDITOR 等,读取 \$01 磁道 \$0A 扇区,将 \$45——\$4A 单元内值改为 EA EA A9 40 EA EA,再写回 \$01 道 \$0A 区。再读取 \$04 磁道 \$07 扇区,将 \$E0——\$E3 单元内的值改为 EA EA C9 40(40 道)或 EA EA C9 60(44 道),再写回 \$04 道 \$07 区。

*注:以上参数是以 PRODOS 文件为盘中第一个

文件为基准的。

经修改过的各版本 ProDOS 便具有处理 40 磁道或 44 磁道的功能。

二、文件 FILER 的修改

目前 FILER 文件有两种版本,即 FILE V1.0 与 FILER V1.1。修改如下:

启动 ProDOS 系统进入 BASIC 状态,键入 BLOAD FILER,TSYS,A \$ 2000,按 CALL-151 进入监控状态,然后针对各版本键入以下数据:

V1.0 版 4243;A9 40(40 道)或 A9 60(44 道)

79F3;C9 28(40 道)或 C9 2C(44 道)

V1.1 版 4269;A9 40(40 道)或 A9 60(44 道)

79F3;C9 28(40 道)或 C9 2C(44 道)

按 CTRL-C 回到 BASIC 状态,键入 BSAVE FILER,TSYS,A \$ 2000。若 FILER 已上锁,就先解锁再存盘。

经修改过的 FILER 文件,与修改过的 ProDOS 配合,就可以管理 40 磁道或 44 磁道的磁盘。

最后重新启动修改过的 ProDOS 盘,选择 FILER,格式化一张磁盘,将修改过的 PRODOS、FILER 以及其它文件拷入刚格式完的磁盘,此磁盘便为 40 磁道或 44 磁道的 ProDOS 系统盘。

制作 40 磁道磁盘的最简方法

湖北荆州师专计算机室 翟波

APPLE 机用户为了增大磁盘容量,常将磁盘由 35 磁道扩充为 40 磁道,但所使用的方法大都较繁。本文给出制作 40 磁道磁盘的最简方法。在键入 INIT 格式化命令之前,先键入如下三条 POKE 语句:

]POKE 44725,160✓

]POKE 46063,40✓

]POKE 48894,40✓

再使用 INIT 命令格式化出的磁盘就具有 40 磁道,其 VTOC 表中的有关参数在格式化时已设定好,用不着再去进行修改、释放等工作。用 FID 程序的第 3 项功能,可看到该磁盘的扇区数增加了 80 个。格式化出的 40 磁道盘将工作得很正常,而且,用这张盘作源盘去格式化其它盘,得到的也是 40 磁道的磁盘。

PC-1500 机新监控程序

沈玉波

监控程序是用户编制机器语言程序的有力工具。PC-1500 机原未配监控程序,后来厂家提供了一个 0.5K 的多功能监控程序,使用该程序可观察和修改 64K 内存中任一单元内容(当然对 ROM 及无 RAM 的空缺区修改无效),并可检验 32 个内存单元是否有变码。使用时各状态间可相互转换,也可退出监控,返回 BASIC。在使用过程中,我们感到该监控程序还有一些不尽如人意之处,如在监控状态不能直接关机,须返回 BASIC 方能关机,且关机后再开机时,须重新输入地址,给机器语言程序的调试带来不便。另外,原监控程序不具备增、删功能。

这里向读者介绍我们新编制的监控程序。它除具备原程序的全部功能外,增加了监控下关机和一字节增、删功能,使用更加方便。在监控模式下,按 OFF 键可直接关机,按 ON 键开机后,可保存关机前的显示内容、工作模式和各寄存器(CPU 内)内容,反复调试程序更加方便。在编辑状态,用 SHIFT DEL 键可在光标位置删除一字节,其余后续字节依次前移。按 SHIFT 和 INS 键后,可在光标处加入一条 NOP 指令(38H),其余后续字节依次后移。

监控程序全长 634 字节,可浮动。如使用 PC-1500A 可全部存入 7C01H 为首址的内存中,不受 NEW 或 NEW N 指令影响。如是 PC-1500 机应当首先用 NEW &4340 开辟机器语言保护区,然后再以 40C5H 为首址存入。使用时首先用 CALL &7C01 或 CALL &40C5 指令启动监控,这时屏幕显示 ADR. &__,应输入 4 位十六进制地址,如按下 /、*、一、+、= 及 · 键时,将由监控程序译码为 A、B、C、D、E、F,可快速输入数据。如输入地址有误,可重新输入或用 CL 键清除显示后重输。输入正确地址后,按 ENTER 键进入内容显示状态,一次显示首址及 8 个字节内容,按 ↓、↑ 键可显示后、前 8 字节内容;如按 ▶、◀ 键则进入编辑状态,这时可输入机器语言程序或对内存进行修改、增、删;如按 & 键可在光标处复制前一字节内容,按 ▲ 键则进入校验和状态,可一次显示 32 字节内容,校验和状态按 ↑、↓ 键可快速显示校验和。在各状态均可按 F1 键进入地址输入状态,也可按 ON 键返回 BASIC。不能对 BASIC 区首址后内容进行增删处理。

新监控程序清单如下:

7C00— 0D 6A 45 58 7B 5A 10 E9	7CE0— 7C 81 15 ED 78 9B 80 89	7DC0— 9B 3F 9E 88 BE E3 3F 9E
7C08— 78 9B 00 FD 58 B5 04 8E	7CE8— 01 F1 BE 7B 26 AE 78 7D	7DC8— 70 F2 E9 78 9B 3F 14 B9
7C10— 46 94 BE 7B 15 14 FD C8	7CF0— 14 B9 07 DD DD F1 ED 78	7DD0— F8 1A EB 78 7C 81 9E A2
7C18— F1 BE 7B 26 BE ED 4D FD	7CF8— 9B 80 8B 02 B3 06 BE EE	7DD8— B5 08 8E 13 B5 F8 FD 52
7C20— 8A BE 7B 26 BA ED 4D B9	7D00— 22 CA 7E B5 7F CD 8A BE	7DE0— 8E 0D 68 20 8E 04 68 E0
7C28— 0F F9 B3 30 B7 3A 81 02	7D08— 7B 32 BE ED 7D 81 18 ED	7DE8— FD 52 14 B9 F0 1A A4 FD
7C30— B3 06 9A BE E2 43 48 7B	7D10— 78 9B 80 89 05 F1 59 0F	7DF0— DA E9 78 7C 00 9E C1 14
7C38— 4A 45 6A 05 F7 8B 03 88	7D18— 8E 02 59 F0 1B 1E EF 78	7DF8— B9 F0 1A 56 F2 E9 78 9B
7C40— 05 9A 24 B3 40 9A 2E 3D	7D20— 9B 80 91 7A 54 9E 7D B7	7E00— BF EB 7C 9B 80 14 BB 07
7C48— 2B 2D 2A 2F 41 4A 52 2E	7D28— 0C 9B 0D B7 08 89 DB EF	7E08— 9E 39 B7 0F 89 05 BE E3
7C50— 26 20 20 20 20 5F 0D FD	7D30— 78 9B 80 93 8B 56 9E 8E	7E10— 3F 9E EC B7 1D 8B 11 B7
7C58— CA F5 88 03 48 7B 4A 4B	7D38— EB 78 9B 40 E9 78 7C 00	7E18— 1C 8B 36 B7 16 99 D2 56
7C60— 58 7B 5A B0 6A 0A F5 88	7D40— 9E 99 BE E2 43 B7 0C 8B	7E20— 55 51 E9 78 9B 04 9E F2
7C68— 03 E9 78 80 00 BE E8 CA	7D48— 81 B7 08 8B B4 B7 0A 8B	7C28— 14 AE 40 00 94 AE 40 01
7C70— BE 7B 32 48 7B 4A B5 B7	7D50— 87 B7 0B 8B 87 B7 09 9B	7E30— 54 54 56 57 1E 54 54 54
7C78— 20 81 0B FD 5A 44 F5 F5	7D58— 21 B7 11 9B D9 B7 0F 8B	7E38— 94 A7 78 65 99 0C 14 A7
7C80— F5 1E 9E 1B 9E 2A B7 18	7D60— 05 B7 0E 99 BB 9A BE E3	7E40— 78 66 99 12 56 B5 00 1E
7C88— 9B 2E B7 0E 8B D7 B7 0F	7D68— 3F 9E 26 B9 F0 1A B5 F0	7E48— A5 40 00 1A A5 40 01 18
7C90— 8B 06 B7 0D 99 2D 8E 05	7D70— FD DA FD 52 BE 7B 10 B5	7E50— 9E 2C 44 04 AE 40 00 84
7C98— BE E3 3F 9E 2A BE ED 95	7D78— 7E BE ED 4D B5 00 6A 0F	7E58— AE 40 01 A5 78 65 08 A5
7CA0— 91 46 18 BE ED 95 91 4C	7D80— D9 13 54 88 05 FD C8 56	7E60— 78 66 0A 46 46 45 0E 04
7CA8— 1A F2 E9 78 75 00 B5 41	7D88— BE 7B 10 EF 78 75 06 FD	7E68— 16 99 08 84 96 99 0C B5
7CB0— AE 7B 0E FD 98 14 B9 F8	7D90— 8A BE 7B 15 ED 78 75 40	7E70— 38 1E A5 40 00 04 A5 40
7CB8— ED 78 9B 40 89 AD 1A BE	7D98— 8B 07 EF 78 75 0C 54 9E	7E78— 01 08 9E 57
7CC0— 7B 10 EF 78 75 04 6A 07	7DA0— 2D BE E2 43 FD 1A B7 0A	
7CC8— EF 78 75 04 55 FD A8 BE	7DA8— 8B 38 B7 0B 8B 38 B7 08	
7CD0— 7B 15 FD 2A 88 0E FD 1A	7DB0— 8B 45 B7 0C 8B 13 B7 0F	
7CD8— ED 78 7C 01 8B 64 EB 78	7DB8— 8B 0A B7 09 99 65 E9 78	

第五讲 程序控制结构(二)

李文兵 王玉华 蓝智斌

3. 中断结构

(1) 中断循环结构 这种结构是由中断语句实现的, 这里介绍两个用于 C 程序各种循环结构中的中断语句:

break; —— 跳出循环结构

continue; —— 返回到循环开头

① break 语句 该语句除了用在 switch 结构中, 还常用于各种控制结构中, 其功能是从其所在的循环结构中跳出, 如练习 5.11 所示。

[练习 5.11]

C>type exp5-11. c

```
main ()
{
    int i, j, k;
    for (j=0; j<10; j++)
    {
        k=i+j;
        if (k>15) break;
    }
    printf (" i      j      k      \n");
    printf (" %d %d %d\n", i, j, k);
}
```

C>exp5-11

```
i      j      k
0      10     9
```

② continue 语句用于各种循环结构中, 常作 if 的执行语句, 其功能是返回到循环结构的开关, 如练习 5.12 所示:

[练习 5.12]

C>type a;exp5-12. c

```
main()
{
    int i, j;
    /* continue test 1 */
    for (j=4; j<10; j++)
    {
        for (i=0; i<50; i++)
        {
            if (i%j) continue;
            printf ("%d ", i);
        }
        printf ("\n");
    }
    /* continue test 2 */
    i=0;
    while (++i<=9)
    {
        if (i==5) continue;
        printf ("%d ", i);
    }
}
```

C>exp5-12

```
0  4  8  12  16  20  24  28  32  36  40  44  48
0  5  10  15  20  25  30  35  40  45
0  6  12  18  24  30  36  42  48
0  7  14  21  28  35  42  49
0  8  16  24  32  40  48
0  9  18  27  36  45
1  2  3  4  6  7  8  9
```

(2) 中止函数执行结构 这种程序结构使用 return 语句实现。该语句有两个功能: 一是它将立即中断含有它的那个函数的执行, 返回到调用此函数的语句处往下执行; 二是它可以用来回送一个数值。C 程序中大多数函数是采用 return 语句来终止运行的, 原因是有时函数须返回一个数值; 有时需要在函数中设置多个终止点以提高效率。这就是说, 一个函数可以有多个返回语句, 如练习 5.13 所示。

C>type exp5-13. c

```
main()
{
    char str1[]="abcdefghijkl.";
    char str2[]="efg";
    printf("str2 in str1= %d\n", substr(*str1, *str2));
}

substr(s1, s2)
char *s1, *s2;
{
    register int t;
    char *p2, *p;
    for (t=0; s1[t]; t++)
    {
        p=&s1[t];
        p2=s2;
        while (*p2&&*p2==*p)
        {
            p++; p2++;
        }
        if (!*p2) return(t);
    }
    return(-1);
}
```

C>exp5-13

str2 in str1=4

(3) 无条件转移结构 使用 goto 语句可实现无条件转移, 其结构形式如下:

goto 标号;

功能是: 它能转去执行标有该标号的语句。所谓标号, 同变量一样, 也是用标识符来表示, 只是其后面跟有冒号而已。用 goto 语句转去执行的语句前面要放有标号。

goto 语句可从多重循环的某层次转到任何层次;

而 break 语句只能从里向外跳转一层。这是 goto 语句的优点。但是必须注意,①goto 语句只能在 goto 所在函数内跳转,不能转到函数外。②goto 语句不适于结构化程序,因此,奉劝大家尽量少使用,以免造成程序难读难懂。

这里,介绍一个使用 goto 结构从最里层 for 结构中跳出的程序,如练习 5.14 所示。

[练习 5.14]

A>type exp5-14.c

```
main()
{
    int i,j,k;
    for (i=0;i<10;i++)
    {
        printf("i=%d\n",i);
        for (j=0;j<10;j++)
        {
            printf("j=%d\n",j);
            for (k=0;k<10;k++)
            {
                goto stop;
                printf("%d,%d,%d\n",i,j,k);
            }
        }
        stop;printf("from stop jmp.");
    }
}
```

A>exp5-14

i=0

i=0,j=10

from stop jmp.

4. 递归结构

从 2 讲中,我们知道,一个函数可以调用另一个函数。此外,函数还可以调用其本身。这样的函数,就叫递归函数,递归函数就是一个递归结构。阶乘的算法就可用递归结构来实现,如练习 5.15 所示。

[练习 5.15]

A>type exp5-15.c

```
main()
{
    int i;
    long int fact(int);
    for (i=1;i<10;i++)
        printf("fact(%d)=%8ld\n",i,fact(i));
}

long int fact(i)
int i;
{
    if (i==1)
        return((long)i);
    else
        return((long)i * fact(i-1));
}
```

A>exp5-15

```
fact(1)= 1
fact(2)= 2
fact(3)= 6
fact(4)= 24
fact(5)= 120
fact(6)= 720
```

```
fact(7)= 5040
fact(8)= 40320
fact(9)= 362880
```

这里,函数 fact()就是递归函数。

下面,再介绍 2 个递归结构的用法。

第 1 个是求两个正整数最大公约数的程序,如练习 5.16 所示。

[练习 5.16]

A>type exp5-16.c

```
#include <stdio,h>
main()
{
    int i,j,k;
    char s[80];
    for (k=0;k<4;k++)
    {
        i=atoi(gets(s));
        if (i<0)
            exit(0);
        j=atoi(gets(s));
        printf("hcf(%d,%d)=%d\n",i,j,hcf(i,j));
    }
}

hcf(1,j)
int i,j;
{
    int r;
    r=i%j;
    if (r==0)
        return(j);
    else
        return(hcf(j,r));
}
```

A>exp5-16

9

21

hcf(9,21)=3

8

2

hcf(8,2)=2

40

16

hcf(40,16)=8

5

2

hcf(5,2)=1

这里用的是欧几里得算法,即对于正整数 p 和 q,设 p 除以 q 的余数为 r,就有:

若 r 为 0,则最大公约数就是 q;

若 r 不为 0,则用 q 和 r 分别置换 p 和 q,重复进行上述判断。

第 2 个例子是求费班纳赛(Fibonacci)级数的程序。该级数为:

0,1,1,2,3,5,8,13,21,34,.....

可见,每个元素是由相邻的前两个元素的和组成的,如:

```
f(4)=f(3)+f(2)
    ={f(2)+f(1)}+{f(1)+f(0)}
    ={f(1)+f(0)+f(1)}+{f(1)+f(0)}
    ={1+0+1}+{1+0}
    =3
```

其一般形式为:

$n=0,1$ 时, $f(n)=n$;

$n \geq 2$ 时, $f(n)=f(n-1)+f(n-2)$

实现该级数的程序如练习 5.17 所示

[练习 5.17]

A>type exp5-17.C

```
main()
{ int i=6;
  for(i=0;i<10;i++)
    printf("fib(%d)=%d\n",i,fib(i));
}
```

```
fib(i)
int i;
{ if(i<=1)
  return(i);
  else
  return(fib(i-1)+fib(i-2));
}
```

A>exp5-17

```
fib(0)=0
fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
fib(7)=13
fib(8)=21
fib(9)=34
```

关于四舍六入程序的讨论

辽宁地震局地震大队 梁 放

贵刊一九九一年第二期上刊登了韦肖鸿同志的关于数据 4 舍 6 入处理程序,我对该程序进行了实际使用,在使用中发现该程序虽然能解决数据 4 舍 6 入问题,但是由于采用字符处理方式,使得处理方法比较繁琐,且程序较长、运行速度慢。我根据自己的工作经验,在不改变数据取舍原则的基础上编写了如下程序。从程序清单可以看出,该程序语句很少,步骤简洁明了,特别适合内存容量有限的微型计算机做为子程序使用。

```
1000 INPUT "A=";A;INPUT "P=";P
1001 B=A+0.5/10^P;C=INT(B*10^P)/10^P
1002 IF B*10^P=INT(B*10^P) AND INT((A*10^P)/2)*2-INT(A*10^P)=0 THEN LET C=INT(A*10^P)/10^P
1003 PRINT C
```

说明:

A: 为要处理的数据。

P: 为需要保留的小数位数。

C: 为处理后的结果。

若 A 为负数,可先将 A 变为 |A|,再进行处理,处理后还原。

浙江嵊县中学 叶德祥

对正负数的“舍”和“入”的理解应有区别,如保留小数 1 位,对于 1.55 是“入”变成 1.6,其值比原数大,而对 -1.55 是“入”变成 -1.6,其值比原数小,所以“入”不一定数值变大,同样“舍”数值不一定变小。如果用韦肖鸿同志的程序对 1.55 和 -1.55 进行保留 1 位小数的修约,则分别变成 1.6 和 -1.5;原来互为相反数的两数修约之后就不互为相反数了,这不合理,并有可能使程序潜伏难以发觉的错误,绝不能掉以轻心。此外,这样的程序不用字符串也能解决,而且运算速度更快,下面是我编写的程序,供大家参考。

```
100 Y=X*10^N;Z=INT(Y+0.5)
110 IF Y+0.5>Z THEN 140
120 IF Z/2=INT(Z/2) THEN 140
130 Z=Z-1
140 X=Z/10^N
150 RETURN
```

程序说明:数据 X 保留 N 位小数,对小数点后第 N+1 位进行舍入处理。如小数点后第 N+1 位数字不是 5,或小数点后第 N+1 位数字是 5,但后面的数字不全是零,则与平常的舍入法则相同,此时 $Y+0.5 > \text{INT}(Y+0.5)$ 由 110 句判断。如小数点后第 N+1 位数字是 5,且后面的各位数字皆为零,则舍入后要保证小数点后第 N 位数字为偶数,即 Z 为偶数,由 120、130 句实现。

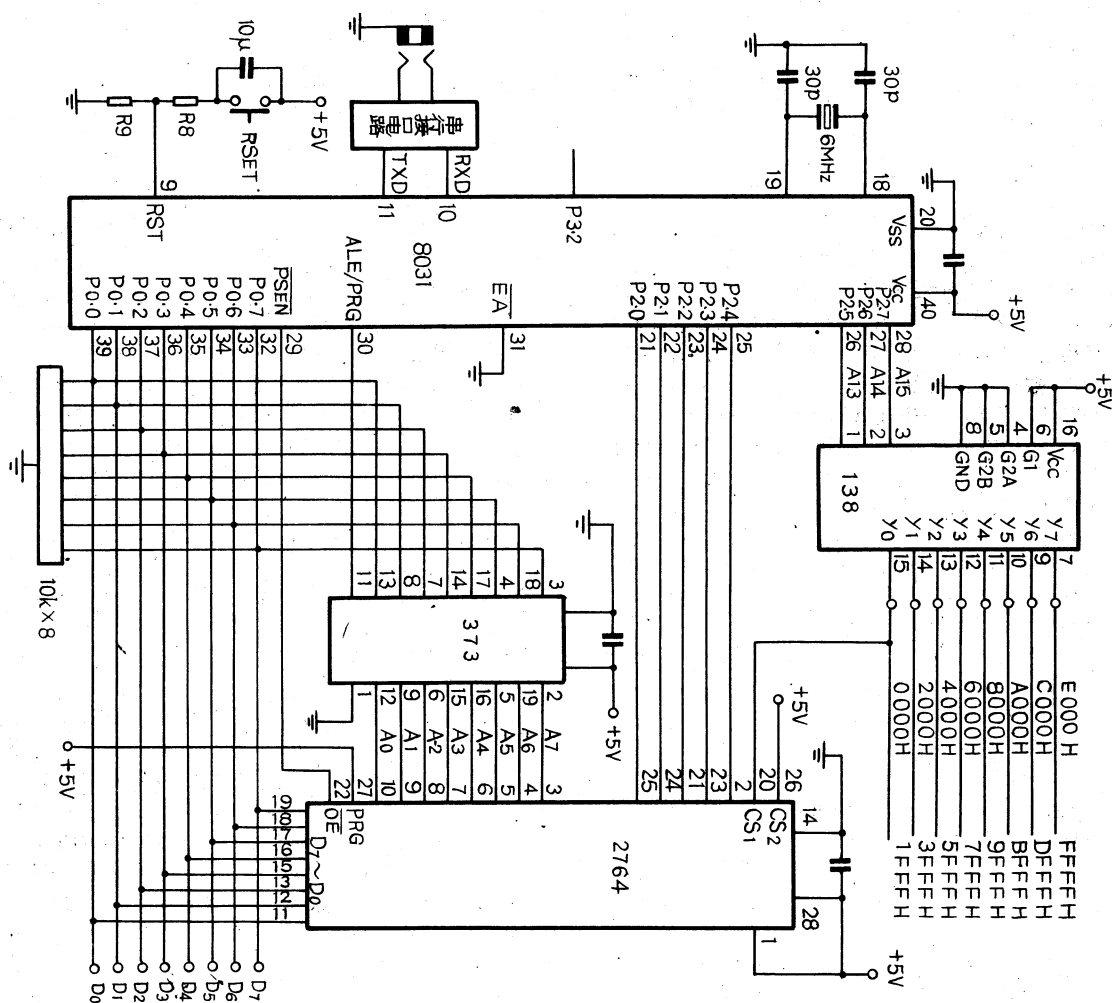


单片机最小应用系统

张培仁

MCS-51 系列中, 8051 片内有 4K 掩膜 ROM, 8031 无 4K 掩膜板 ROM, 8751 片内有 4KEPROM。从国内市场买到 8051 时, 用户无法把程序写到掩膜 ROM 中。因此, 国内组成单片机最小应用系统对 8051 (或 8031) 来说只能外接一片 EPROM (如 2764) 作为用户程序。8051 (或 8031) 从 2764 中取指令时, 必须从 P2 口和 P0 口分别输出 16 位高位地址和低位地址。而 P0 口又是数据输入, 输出的口。因此 P0 要在指令控制下分时输出地址, 再输入或输出数据。这样先输出的地

址要锁存起来, 2716, 2732, 2764 等都是不带地址锁存器的。为了从 EPROM 中取出稳定的正确的程序必须在 8051 外再外接一片 74LS373 作为地址锁存器, 由 3 片组成最小系统。如果 EPROM 是带地址锁存器时只要 2 片就可以组成最小的应用系统。一般为了便于扩充, 常选用 4 片组成最小系统 (增加一片 74LS138), 作为最高 3 位地址的译码器。最小应用系统的电原理图如图 1 所示。印刷电路板图为 1:1, 见本期封三。



原理图中一部分控制信号说明如下:

RST/VPD:复位/掉电时备用电源。复位信号“1”电平有效。片内内接下拉电阻(10K Ω)。

ALE:允许地址锁存。“1”电平有效。此信号是周期出现,频率为主振荡频率的 1/6。

PSEN:此输出是外部程序存储器的读选通信号。它的产生是由程序存储器的取指令来控制。在访问外部数据存储器时此信号不出现。“0”电平有效。

EA:在最小应用系统中它永远是低电平。因为总要不不断地从外部 EPROM 取指令。

在最小的应用系统中,P0 口已不能再作 I/O 口使用,它只能在指令控制下分时输出地址和输入输出数据。P2 口也只能作为 2764 高位地址的输出,也失去作为 I/O 使用的可能。因此可供用户使用只有 P1 口和

P3 口。P3 口是多功能口。

如图所示,此最小应用系统包括 8031,2764,373,138 各 1 片。以及+5V 稳压电源的整流、滤波、稳压电路。还包括用二只晶体管组成非标准的 RS232 接口电路和三芯插座一只。为了便于扩充还留下 40 芯接口插座一个。

当最小系统由三片组成时,省掉 138,把 2764 的 CS1(片选信号)接地。其他不做任何改动。如果 CPU 用 80C31 时要在 P0 口接下拉电阻。

最小系统中用户可用资源还有二个 16 位定时器/计数器,一个可编程的串行口,8031 片内可用 128 字节 RAM 等。用户可以在这样硬件环境下创造出丰富多彩的自己的单片机应用系统。(本刊今后将陆续介绍 KDC 最小应用系统的文章供读者参考——编者注)

电 脑 坦 克

(KDC 最小系统应用实例之一)

张培仁 严 琦 阮永光

KDC 最小系统稍加扩充可构成一个实用的功能强的单片机应用系统。

玩具坦克是两个直流电机驱动的,原来用遥控开关转换电源的极性,以控制电机的正转反转,从而使坦克前进,后退、转弯等。我们改用单片机来控制其运动,并接上小喇叭和发光二极管,可模拟枪炮及音乐音响。具体电路见图 1。驱动电路是这样工作的:8031 的 P1.0~P1.3 口,经 9012 驱动,控制光电耦合器 4N25 的导通与截止,4N25 实现了单片机与功率驱动部分的电

气隔离,最小系统与驱动电路可采用不同的电源,4N25 完成了单片机+5V 到驱动电机±5V 电源的电平转换,8751 与 8778 是同一类型的功率集成开关电路,4N25 的第 4 脚输出接到集成开关的控制端(8751 的第 1 脚,8778 的第 5 脚),8751 输出接电机,它导通时电机正转(图中电流 I₁),同样 8778 导通时电机反转(图中流过电流 I₂)。另外喇叭和发光二极管也是由一片 8751 驱动的。

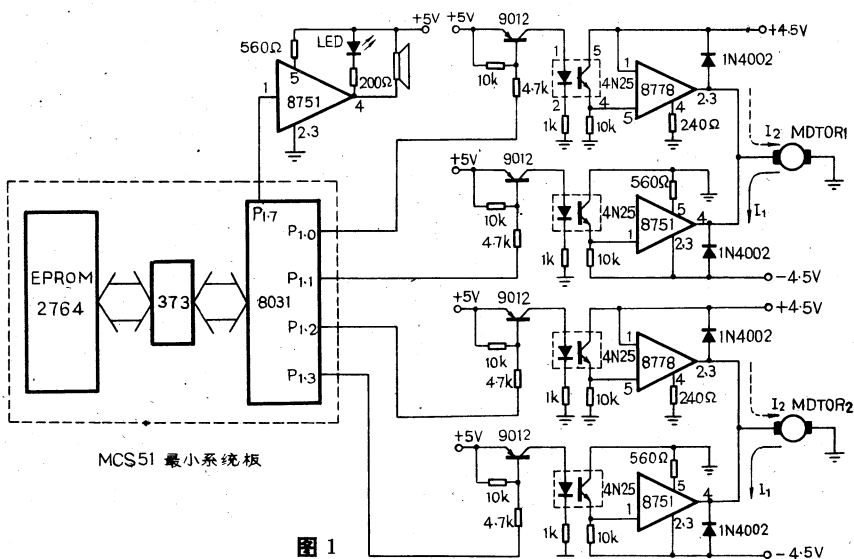


图 1

TWH8751, TWH8778 都是功率开关集成电路。这是专为逻辑电路输出作接口而设计的直流功率开关, 可由 TTL、CMOS 等数字电路直接驱动。该器件工作可靠, 开关速度快, 工作效率高, 寿命长。这是一种广泛应用自控系统、计算机终端接口、测量仪器中微电机控制等电路。TWH8751, 8778 是一种集放大、比较、选通, 整形和功率输出于一体的大规模集成开关功率器件。并自带散热片。TWH8751 是五端器件。①是控制选通端 (ST) 与输入端和输出端相差 180°。②是输入端 IN。③端是地。④端功率输出端 (OUT)。⑤供 TWH8751 放大、比较、驱动电路电源端。电压不超过 +6V。超过 +6V 时要接降压电阻。

TWH8751 性能特点: 1. 器件设计有死区, 有滞回特性, 低噪声、抗干扰性能好。2. 有自我保护, 不易损坏。控制电平输入 $< 0.3V$ 导通, $> 1.1V$ 时输出功率管截止。3. 工作频率可达 1.5MHz。4. 开关性能好, 边沿延时毫微秒级。5. 控制功率大, 功率开关管击穿为 100V, 加散热片可达 3A。典型应用如图 2。

TWH8778 和 TWH8751 不同的是 8778 输出功率管是 PNP 型, 8751 是 NPN 型。其功能性能基本相似。

软件部分主要是对 P1 口的控制。程序主模块是表演部分 (Persorm) 用于设置坦克行走路线与发声控制, 下面有二个子模块, 即方向控制子模块, 包括前进 Forward, 后退 Backward, 左转 LTurn, 右转 RTurn, 发声子模块包括音乐 MUSIC (含五首曲子), 枪炮, 详细注解见程序清单。

将程序写入 EPROM 后, 插入最小系统插座, 再将 P1 口接到控制板上, 坦克就可按预想的方式运动, 并演奏美妙的乐曲, 在停顿时发出激烈的枪炮声。

稍加改动, 该系统还可扩展许多功能, 如增加一超声测距装置, 可使坦克探测到障碍物, 并自动走出迷宫; 接上发光管和光敏接收管, 可实现坦克与目标间的

(接第 3 页) 要看 Intel 的垄断能否被打破。

二、总线的发展

微机的标准总线为 ISA (Industry Standard Architecture) 是 IBM 在 1981 年推出 IBM PC 时订下的, 它是一个颇为简单的八位同步总线含校验位保护 (Parity protection) 及以边缘触发中断指令 (edge-triggered interrupt)。1984 年 IBM 再推出了 16 位的 AT, 为了要跟以前的 8 位 ISA 往下兼容, IBM 在原来的 62 针总线插座下加多一个 38 针的插座, 但事实证明 AT 总线需比 8 位的总线有所进步, 但它仍是微机在技术上发展的一个最大障碍。ISA 的 AT 总线仍是一个低速度 (输入/输出), 及不适合多个微处理器并行工作的总线。

1987 年 IBM 再推出了一系列全新的微机系列 (PS/2 系列), 而且采用了 IBM 自己订立的微通道 (Micro Channel Architecture) 总线。IBM 希望能以 PS/2 系列增加它不断减少的市场占有率, 同时为了要保护他在市场的控制权, IBM 注册的微通道总线的版权, 凡是要生产微通道总线的厂家, 均需付出极为可观的版权费用, 而且追索至厂家以前生产 ISA 总

对抗性演习, 加上红外发射接收管, 很容易实现遥控。单片机为这些功能的扩展提供了极大的便利。

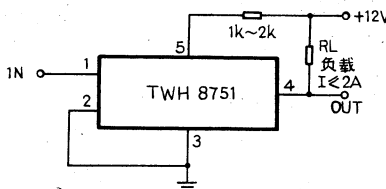


图 2 放大、比较、功放电路

该系统虽是为一玩具坦克设计的, 但实际上, 它有很大的通用性与实用性, 8751 与 8778 属功率集成片。我们曾试用二个 8751 并接可以控制 24V, 3.5A 步进电机可靠工作。用软件改变 P1 口输出脉冲的宽度, 也可方便地控制电机速度, 如果加上各种传感器, 如光栅, 速度计等, 再通过单片机 T0、T1 口输入则可构成一个很实用的闭环的反馈控制系统。

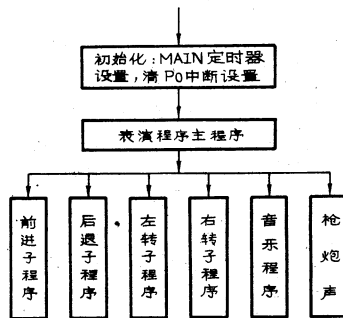


图 3 程序框图

程序框图如图 3 所示。有兴趣的读者可向本刊编辑部函索程序清单, 付复印及邮寄费 3.00 元。

线的所有微机。IBM 的政策使大部分的微机生产商没有兴趣设计及生产微通道微机, 最致命的是 IBM 的微通道总线跟以往的 ISA 在硬件及软件上并不兼容。在前景不明朗的情况下硬件及软件制造商并未在微通道上大量投资, 这是做成微通道总线至今仍未普及或变为主流之原因。

1988/89 年 9 间美国及日本微机制造商联合创制了一种全新的总线 EISA (Extended Industry Standard Architecture)。EISA 的主要目的是解决 ISA 在功能上的不足, EISA 可同步的以 33MHz 速度进行输入/输出数据, 而且从设计上容许多处理器并行工作。EISA 的插座有两排插针可往下兼容所有 ISA 的插板, 这使在未有专为 EISA 而设计的插板时, 用户仍可用现时的 ISA 硬件 (当然, 功能便跟现时 ISA 总线没有分别) 同时 EISA 总线的设计及专利使用权以合理的价格供应给所有微机制造商。这政策看来颇为成功。现时设计 EISA 总线的厂商已远远超过设计 IBM 微通道的厂商。现时支持 EISA 的硬件及软件还未丰富, 估计 EISA 还需数年才能普及及成为主流标准。IBM 现时还很积极的宣传及推广 MCA (微通道), 但它的成功未容乐观。(待续)



8031 与 PP40 描绘器接口技术

皇甫文忠

LASER PP40 是一种体积小,价格低的四色描绘式打印机。可用来描绘字符及图形,可以打印 89 个标准 ASCII 字符和 52 个特定字符,并且具有较强的绘图功能。可在各种智能仪器及控制系统中作为微型绘图机使用。当然,在单片机开发系统中配接 PP40 描绘器是一种很实用的选配方案。

一、LASER PP40 技术参考

1. LASER PP40 接口信号及时序

PP40 的接口插座是 36 芯的,实际上只使用了 13 芯,各芯位信号说明如下表:

表一、PP40 接口插座芯位、信号及功能

芯 位	信 号	功 能
1	$\overline{\text{STROBE}}$	选通信号
2~9	DATA1~8	8 位并行数据总线
10	$\overline{\text{ACK}}$	应答信号,表示 PP40 准备好接收下一批数据
11	BUSY	“忙”信号,表示 PP40 正在工作中,不能接收信号数据
12、15	GND	接地
其余	不接	

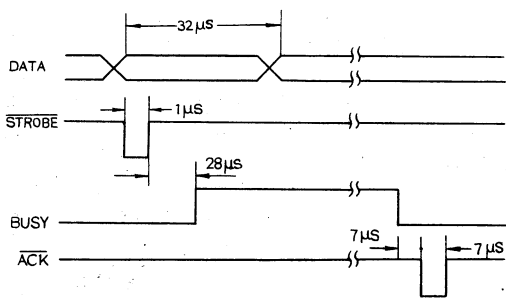
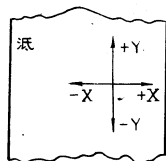
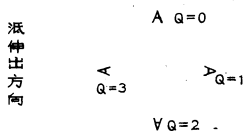


图 1 PP40 接口信号时序



(a) X-Y 方向定义



(b) 字母绘制方向定义

图 2 X、Y 方向及字母描绘方向定义

表二、绘图命令格式及功能

命 令	格 式	功 能
线形式	L_p (p 由 0 ~ 15)	所绘划线的形式,实线 $p=0$, 点线 $p=1 \sim 15$
重置	A	笔架返回 X 轴最左方,而 Y 轴不变动。返回文本模式,并以笔架停留作为起点。
回档	H	笔架升起返回起点。
预备	I	以笔架位置作为起点
绘线	Dx, y, \dots X_n, Y_n ($-999 \leq x, y \leq 999$)	由现时笔嘴位置(x,y)连线
相对绘线	$J \Delta x, \Delta y$ $\dots \Delta x_n, \Delta y_n$ ($-999 \leq \Delta x, \Delta y \leq 999$)	由现时笔嘴位置划一直线至笔嘴点 $\Delta x, \Delta y$ 距离之点上
移动	Mx, y ($-999 \leq x, y \leq 999$)	笔嘴升起,移动至起点相距 x, y 之点上
相对移动	$R \Delta x, \Delta y$ ($-999 \leq \Delta x, \Delta y \leq 999$)	笔嘴升起,移动至现时笔架相距 $\Delta x, \Delta y$ 之新点上
颜色转换	C_n ($n=0 \sim 3$)	颜色转换由 n 指定 0:黑,1:兰,2:绿,3:红
字符尺码	S_n ($n=0 \sim 63$)	指定字符尺码
字母绘制方向	Q_n ($n=0 \sim 3$)	指定文字编印方向(只在图案模式下适用)
编印	PC, C C_n	编印字符(C 为字符, n 无限制)
划轴线	Xp, q, r ($p=0 \sim 1$) ($q=-999 \sim 999$) ($r=1 \sim 255$)	由现时笔架位置绘划轴线 Y 轴: $p=0$, X 轴: $p=1$ q =点距, r =重复次数

PP40 接口信号时序如图 1 所示,在单片机应用系

统中,实际使用时,一般不使用 $\overline{\text{ACK}}$ 信号。

2. PP40 的工作方式

PP40 具有文本模式和图案模式两种绘图工作方式,加电后,初始状态为文本模式,在文本模式下,如果主机将回车控制字符 CR(0DH)和绘图控制字符 DC2(12H)写入 PP40,则 PP40 由文本模式变为图案模式;在图案模式下,再将回车控制字符 CR(0DH)和文本模式控制字符 DC1(11H)写入 PP40,则 PP40 又回到文本模式。

1) 文本模式

PP40 工作于文本模式时,可描绘所有 ASCII 字符及一些特定字符,另外在 PP40 的字符编码中,有以下几个控制字符控制 PP40 的动作。

- 回位 BS(08H):使笔回到前一个字符位置,若描图笔已处于最左边位置,该命令失效。
- 进纸 LF(0AH):将纸推进行
- 退纸 LU(0BH):将纸倒退一行
- 回车 CR(0DH):描图笔返回到最左边位置上。
- 转色 NC(1DH):笔架转动一个位置至另一颜色笔。

当超过一行的字数后,PP40 自动回车并进纸一行。

2) 图案模式

PP40 工作于图案模式时,可通过各种绘图操作命令,绘出各种图形、表格、曲线、绘图命令及格式、功能如表二所示。

X、Y 方向定义、字母描绘方向定义如图 2 所示。

绘图命令的编排有下列约定:

- 参数跟在命令符号后面,参数之间以“,”作分隔符。
- 单字符命令后可直接跟其它命令(返回文本命令除外,它后面必须跟回车符 0DH)
- 一个参数的命令(L,C,S,Q 四条命令),可以在参数后面加“,”后跟其它命令。
- 二个以上参数的命令必须以回车符 CR(0DH)结束。

二、PP40 与 8031 单片机的接口

PP40 与 8031 单片机的接口形式很多,在具体应用时,可根据不同情况加以选择,大致有以下几种接口形式:采取 I/O 口,扩展 I/O 或总线口的连接方法。

1. 采取 I/O 口的接口方法

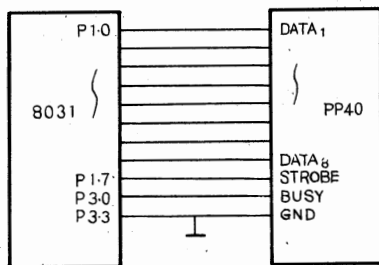


图 3 PP40 通过 I/O 口与 8031 的接口方法

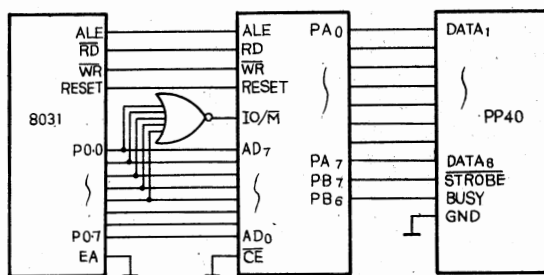
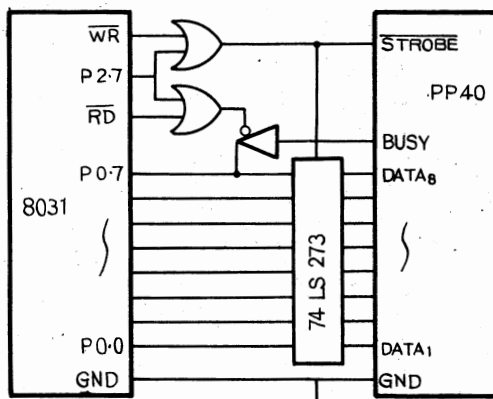
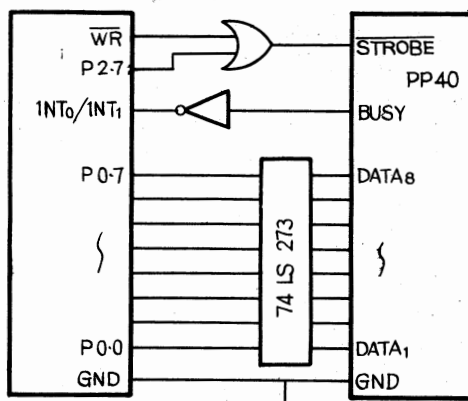


图 4 PP40 通过 8155 与 8031 的接口方法



(a) 查询方式接口



(b) 中断方式接口

图 5 PP40 与 8031 通过 P0 口的接口方法

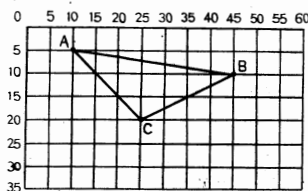


图 6

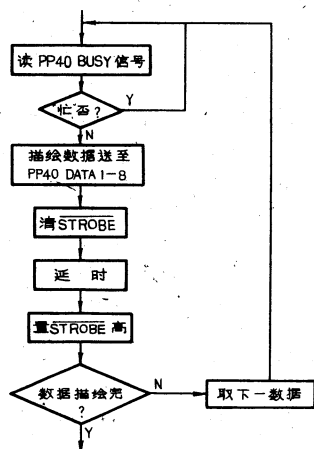


图 7 查询方式程序框图

通过 I/O 口的连接方法比较简单,但要占用 I/O 口线,这里占用 P1 口作为数据通道。

2. 采取扩展 I/O 口的接口方法

这里是用 8155 的 PA 口作为数据通道,也可用 8255 代替 8155,一般来说,只需再加上一对握手线,扩展 I/O 口可以用应用系统中任何并行 I/O 口线。

3. 通过总线口的接口方法

这里在 P0 口的输出端加一锁存器 74LS273,使用 \overline{WR} 信号进行锁存控制。

三、PP40 描绘驱动程序设计

1. 文本模式及图案模式的编码设计

在设计 PP40 的字符及图案描绘程序之前,要对绘制的字符或图案进行编码设计。

首先将所要绘制的字符、图案变成一系列由命令码,控制码及文字字符组成的字符串,然后再将它们“翻译”成相应的以十六进制表示的数据串,作为提供给 PP40 使用的数据表。

举例如下:

1) 文本模式的编码设计

绘制: COMPUTER SYSTEM

可采取的编码如下:

DC2, S3, C1 CRLF

DC1 CR LF

COMPUTE2 SYSTEM CRLF

其中 DC2 是图案模式编码,文本模式中可利用图案模式选择字符尺寸,颜色转换等, S3 表示选 3 号尺寸, C1 为选蓝色笔, DC1 是文本模式命令。

在程序中将上述编码转换成下列十六进制数据,存放在存储器中,然后逐个送入 PP40 中,即可绘出要求的字符。

12H, 2CH, 53H, 33H, 2CH, 43H, 31H, 0DH, 0AH, 11H, 0DH, 0AH, 20H, 20H, 43H, 4FH, 4DH, 50H, 55H, 54H, 45H, 52H, 20H, 53H, 59H, 53H, 54H, 45H, 4DH, 0DH, 0AH

2.1 图案模式的编码设计

绘制如图 6 所示的三角形

编码设计如下:

CR DC2 CR HM10, -5CR; 笔嘴抬起先回起点再移至 A 点

L0, J35, -10, -20, -5, -15, 15 CR; 描绘 AB、BC、CA 线段。

翻译成 16 进制数据如下:

0DH, 12H, 0DH, 48H, 4DH, 31H, 30H, 2CH, 2DH, 35H, 0DH, 4CH, 30H, 2CH, 4AH, 33H, 35H, 2CH, 2DH, 31H, 30H, 2CH, 2DH, 32H, 30H, 2CH, 2DH, 35H, 2CH, 2DH, 31H, 35H, 2CH, 31H, 35H, 0DH

将上述数据送入 PP40, 即可绘出如图 6 的三角形。

2. 描绘程序设计

1) 中断方式接口程序设计

以图 3 接口电路为例, PP40 接口采用中断方式主程序:

```

MAIN:      ;
            ORL PSW, #18H      ;描绘工作初始化,选用
                                ;工作寄存器区 3
            MOV R7, #20H      ;20H 个 ASCII 代码数据
            MOV R5, #00H      ;描绘数据区首址 3000H
            MOV R6, #30H
            ANL PSW, #07H      ;恢复工作寄存器区 0
            MOV P1, #20H      ;启动 PP40 描绘空格符
            SETB P3.0          ;选通信号
            CLR P3.0
            SETB IT1           ;置 INT1 为下降沿触发方式
            MOV IE, #84H      ;允许 INT1 中断请求
            MOV PSW, #00H      ;CPU 可做其它工作
            ;
  
```

中断服务程序:

```

PRINT:     PUSH ACC           ;保护 8031 现场
            PUSH PSW
            PUSH DPH
            PUSH DPL
            ORL PSW, #18H      ;选用工作寄存器 3
            MOV DPL, R5        ;送数据区地址
            MOV DPH, R6
            MOVX A, @DPTR      ;送数据至 PP40
            MOV P1, A
            CLR P3.0           ;选通信号
            NOP                ;延时
            NOP
            SETB P3.0
            INC DPTR           ;指向下一数据
            MOV R5, DPL
            MOV R6, DPH
            DJNZ R7, PRINT      ;数据未完返回继续
            SETB 00H           ;设置结束标志供主程序查询
            CLR EX1            ;关 INT1
PIRI:      POP DPL             ;恢复 8031 现场
  
```

```

POP DPH
POP PSW
POP ACC
RETI

```

说明:

主程中已将 20H 个描绘数据送入首址为 3000H 的数据存储器中。

2) 查询方式接口程序设计

查询方式接口程序的主要思想是利用软件查询 PP40 是否处于“忙”状态,是则等待,否则将数据送至 PP40,置 $\overline{\text{STRUBE}}$ 低电平,选通 PP40,完成一次描绘,主框图如下:

以图 4 的接口电路为例,程序清单如下所示,其中 8155 命令口地址 4100H,PA 口地址 4101H,PB 口地址 4102H。

```

MAIN:  MOV DPTR, #4100H  ;置 8155 工作方式
        MOV A, #0DH      A 口输入, B 口输出
        MOVX @DPTR, A
        MOV DPTR, #4102H ;置  $\overline{\text{STRUBE}}$  高电平
        MOVX A, @DPTR
        SETB A, 7
        MOVX @DPTR, A
        ;
PRINT:  MOV DPTR, #4102H
        MOVX A, @DPTR
        JB A.6, PRINT    ;BUSY 为 1 则转 PRINT

```

```

MOV A, R3      ;R3 中为欲描绘数据
MOV DPTR, #4101H ;数据送至 PP40
MOVB @DPTR, A
MOV DPTR, #4100H ;置 8155 工作方式
MOV A, #0FH      A 口输出, B 口输出
MOV @DPTR, A
MOV DPTR, #4102H ;清  $\overline{\text{STROBE}}$ , 选通 PP40,
MOVX A, @DPTR
CLR A, 7         ;启动一次描绘
MOVB @DPTR, A
PUSH A
MOV A, #08H      ;延时
DEC A
JNZ PPP
POP A
SETB A, 7        ;置  $\overline{\text{STROBE}}$  高
MOVB @DPTR, A
MOV DPTR, #4100H
MOV A, #0DH
MOVB @DPTR, A
RET

```

PPP:

说明:本程序清单中只完成一个数据的描绘,用户可在主程序中将数据送入 R3,然后调用 PRINT,取下一个数据送入 R3,调用 PRINT,……即可完成一批数据的描绘。

解决 TEC—B1 不能使用 AP51—I 卡问题的方法

大庆新华发电厂 刘希栋

《电子与电脑》90 年第 2 期封底邮购消息介绍的 AP51 卡,是用于 APP LE II 及其兼容机的接口卡,但笔者在使用中发现该卡不能用于天坛机 TEC—B1,天坛机插上该卡引导 AP51DEBUG 后,即死机,无法运行。开始认为是卡出了毛病,但插在 CEC—I 中华学习机和 APPLE II 机上,能正常运行,证明卡是好的。

把 AP51 卡分别插在 TEC 和 CEC 机上用示波器查 $\overline{\text{DMA}}$ 信号,可以看出信号是一样的,在引导 DEBUG 后都有 $\overline{\text{DMA}}$ 信号产生。观察 TEC 和 CEC 机上 CPU37 脚(时钟),信号就不一样了,CEC 机在 $\overline{\text{DMA}}$ 低电平期间 CPU 的 37 脚无时钟,而 TEC 机在 $\overline{\text{DMA}}$ 低电平期间 CPU37 脚时钟是连续的,不受 $\overline{\text{DMA}}$ 控制,这就是 TEC 机无法运行 AP51 卡的根本原因所在。

中华学习机 CEC—I 中信号对时钟信号的控制过程见图 1,在 $\overline{\text{DMA}}$ 高电平时 CPU37 脚有时钟,在 $\overline{\text{DMA}}$ 为低电平时切断时钟,6502 等待。而 TEC 机 CPU37 脚时钟信号从 U31 74LS04 第 6 脚引出,不受 $\overline{\text{DMA}}$ 控制,AP51 卡在 TEC 机上由 DEBUG 启动后,发出 $\overline{\text{DMA}}$ 信号,6502 即将控制权交给 MCS—51,但因时钟不受 $\overline{\text{DMA}}$ 控制继续提供,造成 6502 与 MCS—51 同时工作,产生了冲突而死机。

针对上述原因,可按图 2 方法使 AP51 卡在 TEC 机上运行。图中 74LS08 是增添的 2 输入 4 与门(只用其中一个与门),把它放在 TEC 主板上适当位置,7 脚接 0V,14 脚接十

5V,1 脚连续 I/O SLOT22 脚 $\overline{\text{DMA}}$,2 脚连接 U31 74LS04 第 6 脚,3 脚连接至 6502 37 脚,断开原主板上 CPU37 脚到 U31 74LS04 第 6 脚之间的连线。

改动完成后,插上 AP51 卡,引导 DEBUG,AP51 卡将在 TEC 机上正常运行,TEC 机用户要使用 AP51 卡不妨一试。

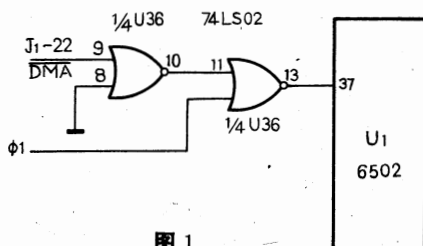


图 1

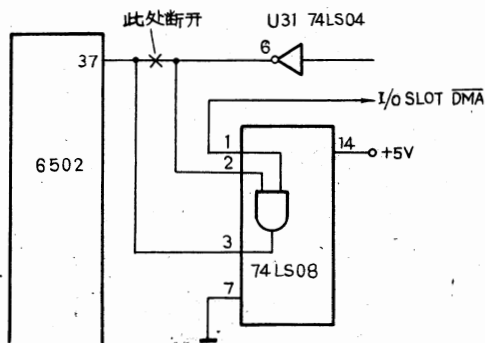


图 2



学装 $\mu\text{p}-80$

微电脑控制电梯模型的 实验与制作

易齐干

该电梯模型假想用于四层楼房。主要动作与实际电梯相同。如：按动指定楼层的按钮，电梯门关闭，电梯起动，到达后，稍停片刻电梯门自动打开。与实际电梯不同之处是不接受一次按押 2~3 个楼层按钮，也不能预约。

该模型大体分为升降、门开闭和基体三部分。

1. 卷绕部分

电梯模型可用厚度为 5~10mm 的有机玻璃制成，强度薄弱处以及电机的固定使用不锈钢板加固。

驱动源使用任意型号的小型直流 (DC) 电机，由齿轮传动装置组成三级减速器 (减速比约 3000:1) 带动装在该轴上的滑轮卷绕钢丝绳提升电梯轿厢。轿厢的下降由本身自重实现。轿厢的重量较轻，使用 $\varnothing 1\text{mm}$ 的钢丝绳就足以保证强度。

“现在，到了几层？”这种位置的检测由安装在各层的光传感器完成。在轿厢的下方突出一块板，遮断光传感器的光路进行位置检测 (图 1 未画出)。

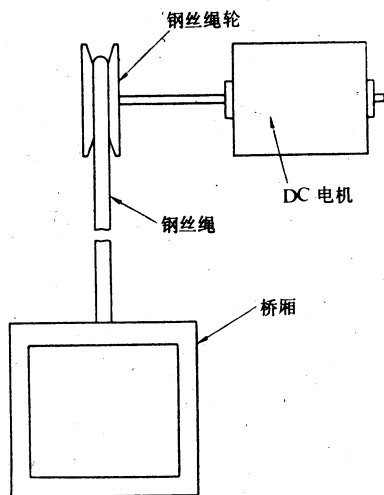


图 1

2. 轿厢门的开闭

轿厢门是用有机玻璃制成。各层楼的横梁是轿厢门的轨道。各层楼都装有直流 (DC) 电磁铁，电磁铁的吸引力通过厢门与电磁铁之间的杠杆，拉动打开厢门。厢门的关闭是靠弹簧的作用力完成。

DC 电磁铁衔铁的行程 (伸出长度) 约 15mm，由于各层轿厢门长度是 40mm，所以，利用杠杆的原理实现行程加大。

3. 基体

四层楼的模型也是由有机玻璃制成，各层设有轿厢门，最上方装卷绕部份，最下面为轿厢。在楼的正面与卷绕部份平行方向并排四支 LED (发光二极管) 作为监控器，表示电梯所处位置。

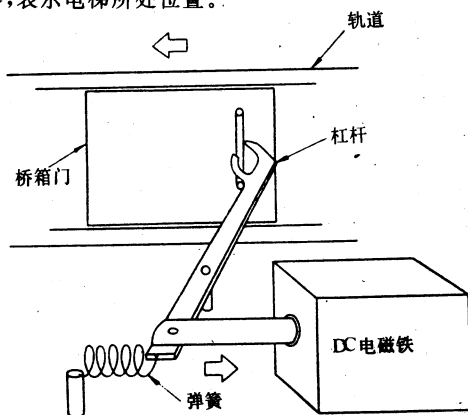


图 2

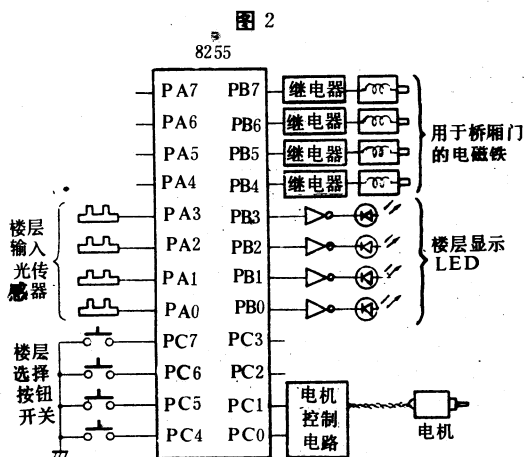


图 3

3. 控制

微电脑控制电梯模型可使用 $\mu\text{p}-80$ 单板机，端口分配如图 3 所示。读者可根据端口分配，设计出各接口电路。如光传感器、按钮开关、电磁铁电路、楼层显示 LED 亮灯电路及 DC 电机驱动电路如图 4~图 8 所示。亮灯电路、电机驱动电路，以及 12V 直流电源。

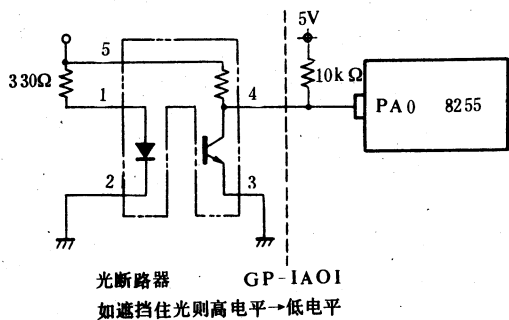


图 4

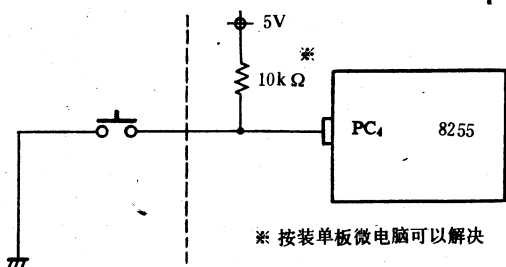


图 5

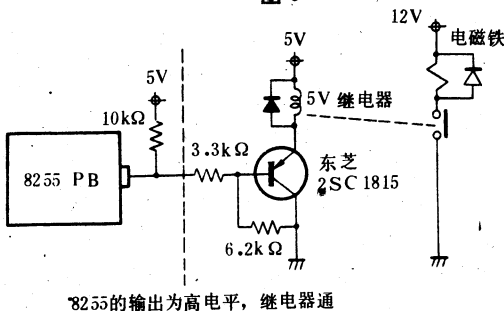


图 6

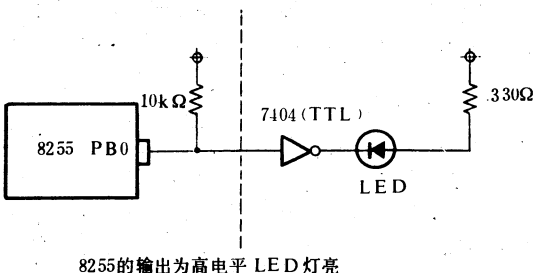


图 7

4. 软件设计

控制电梯的简单流程图如图 9 所示。

该软件是比较按钮开关的状态(由 PC7~PC4 读入,存储于 Z80 的 D 寄存器)与光开关的状态(由 PA3~PA0 读入,存储于 Z80 的 E 寄存器),选择电机应该正转还是停止或是反转。电机停止旋转时(希望轿厢

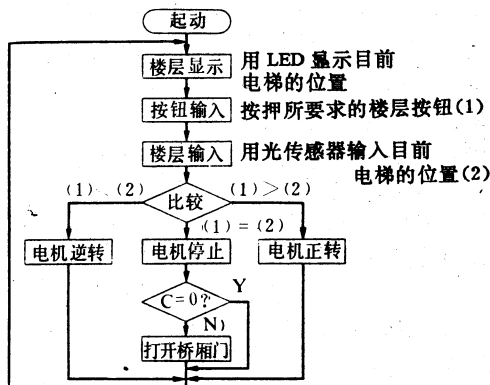


图 9

处于某一层),继电器电路驱动电磁铁,打开该楼层的轿厢门。轿厢门之所以打开,仅是电机停止旋转和判断是正转或反转变之故。电机停止旋转由 01H 进行判断。

Z80 各寄存器的分配如表 1 所示。程序清单与流程图如表 2 所示。

表 1 Z80 各寄存器的分配表:

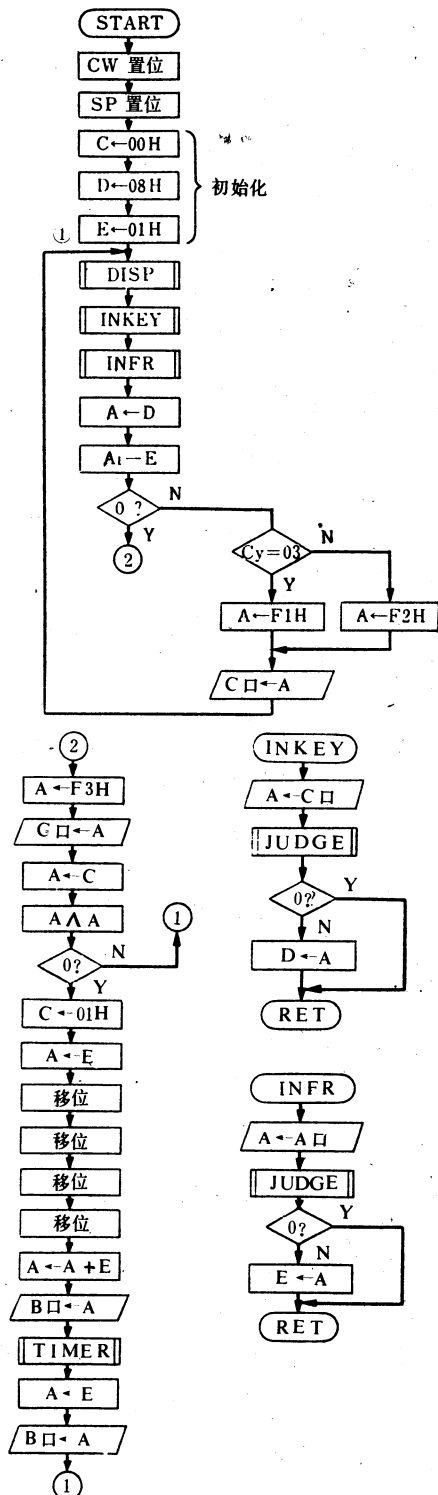
A 寄存器	与 8255 端口交换信息
B 寄存器	不使用
C 寄存器	打开轿厢门的标志
D 寄存器	存储按钮开关的输入值
E 寄存器	存储当前电梯处在的楼层
H,L 寄存器	用于延时的计数器

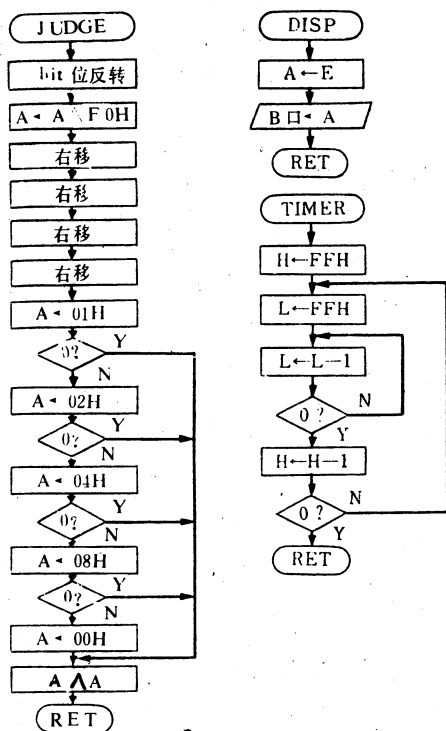
表 2 流程图及其程序清单

标号	助记符	地址	机器语言	注释
START	LD A,98H	00	3E 98	} CW 置位
	OUT(03H),A	02	D3 03	
	LD SP 8100H	04	3100 81	SP 置位
	LD C,00H	07	0E 00	} 初始化
	LD D,08H	09	16 08	
	LD E,01H	0B	1E 01	
MAIN	CALL DISP	0D	CD 7F 00	楼层显示
	CALL INKEY	10	CD 4C 00	按钮输入(D)
	CALL INFR	13	CD 56 00	当前楼层输入(E)
	LD A,D	16	7A	} 比较
	CP E	17	BB	
	JPZ,STOPM	18	CA ZC 00	
	LD C,00H	1B	0E 00	门标志复位
	JP C,DOWN	1D	DA 27 00	
UP	LD A,F1H	20	3E F1	} 电机正转
J1	OUT(02H),A	22	D3 02	
	JP MAIN	24	C3 0D 00	
DOWN	LD A,F2H	27	3E F2	} 电机反转
	JP J1	29	C3 22 00	
STOPM	LD A,F3H	2C	3E F3	} 电机停止
	OUT(02H),A	2E	D3 02	

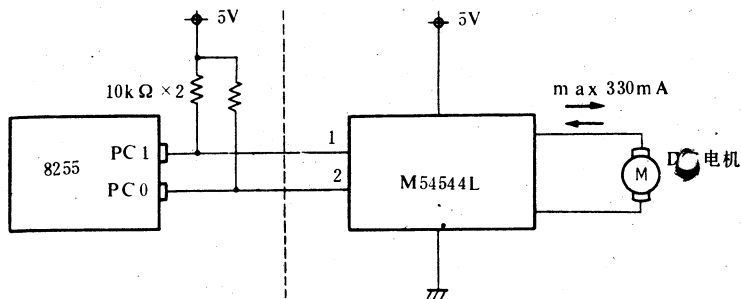
LD A,C	30	79	} 门标志是 否为 0
AND A	31	A7	
JP NZ,MAIN	32	C2 0D 00	
LD C,01H	35	0E 01	置门标志
LD A,E	37	7B	
RLC A	38	CB 07	} 左移 4 位, A 的低位 移进高位
RLC A	3A	CB 07	
RLC A	3C	CB 07	
RLC A	3E	CB 07	
ADD A,E	40	33	
OUT(01H),A	41	D3 01	} 用电磁铁 打开门
CALL TIMER	43	CD 83 00	
LD A,E	46	7B	
OUT 01H,A	47	D3 01	
JP MAIN	49	C3 0D 00	
INKEY IN A,(02H)	4C	DB 02	由 C 口输入
CALL JUDGE	4E	CD 60 00	
JP Z,J2	51	CA 5F 00	
LD D,A	54	5F	按钮输入(D)
J2 RET	55	C9	
INFR IN A(00H)	56	DB 00	由 A 口输入
CALL JUDGE	58	CD 60 00	
JP Z,J3	5B	CA 5F 00	
LD E,A	5E	5F	楼层输入(E)
J2 RET	5F	C9	
JUDGE CPL	60	2F	位反转
AND FOH	61	E6 F0	低位无效
RRA	63	1F	} 高 4 位进 入低位
RRA	64	1F	
RRA	65	1F	
RRA	66	1F	
CP 01H	67	FE 01	1 层?
JP Z,J4	69	CA 7D 00	
CP 02H	6C	FE 02	2 层?
JP Z,J4	6E	CA 7D 00	
CP 04H	71	FE 04	3 层?
JP Z,J4	73	CA 7D 00	
CP 08H	76	FE 08	4 层?
JP Z,J4	78	CA 7D 00	
LD A,00H	7B	3E 00	
AND A	7D	A7	A=0?
RET	7E	C9	
DISP LD A,E	7E	7B	} 端口 B 输 出(E)的值
OUT(01H),A	80	D3 01	
RET	82	C9	
TIMER LD H,FFH	83	26 FF	延时
J5 LD L,FFH	85	2E FF	
J6 DEC L	87	2D 01	
JP NZ,J6	88	C2 87 00	
DEC H	8B	25	

JP NZ,J5 8C C2 85 00
RET 8F C9





上述软件表达的动作有: 按动一次指定楼层开关; 用光传感器检测电梯所处位置; 与指定楼层进行比较; 电梯升、降; 厢门的开、闭等动作。对于多层楼的呼叫记忆与依次的动作; 上升、下降时加速、减速的微妙控制; 厢门的开闭消除不适感的同步方式等等还应该下功夫解决。



输入		动作
1	2	
1	1	制动
1	0	正转
0	1	逆转
0	0	停止

1: 高电平
0: 低电平

图 8

尤其是考虑对多架电梯群控管理, 会深刻地体会到电梯控制的实用化是一件很了不起的事情。

编者注: M54544L 为日本三菱公司生产的 DC 电机控制电路, 读者可根据图 8 状态表, 可用开关三极管电路替代。

LD 磁盘文件查询程序

软件开发: 杜西延

LD 是英文 LABEL DISPLAY 的缩写, 意为标识显示。曾经荣获一九八九年度军队级科技进步四等奖。它能在诸多的文件中迅速、准确、方便地找到自己所需的 (ASCII 码) 文件, 是提高文件管理必不可少的工具软件。LD 不但能显示文件名、文件标示内容、文件字节数, 而且显示文件具体内容、文件的个数和磁盘空间, 它还有文件删除、子目录查询和标识打印等功能。LD 的操作采用屏幕对话方式, 使用起来简单方便。

本软件已由电子工业出版社软件部出版

上述软件已由电子工业出版社软件部出版, 欲购者请与北京 173 信箱软件部联系, 邮编 100036。

CTE 通用表格编辑软件

软件开发: 张伟

CTE 是全屏编辑软件, 用于表格和文书的编辑。它除具有同类文字编辑软件的各项功能外, 还有如下特点:

1. 表格编辑是以一个窗口方框为一个基本操作单元, 制表过程中操作人员始终不需要与表格线打交道, 他所关心的只是方框中要输入什么内容以及方框所在的位置, 方框的大小随输入文字的多少而变化。制表难度不随表格复杂程度的增加而增加, 可编辑任意复杂的表格。
2. 可打印超过打印机宽度的表格。
3. 可对文件进行排版, 版面宽度可在 40~240 间任意选择。
4. 提供了用光标画线抹线的方法。

初级程序员级水平考试辅导问答

汉字 dBASE III 使用技巧(上)

王 路 敬

1. 运行 dBASE III 以前,操作系统配置要作哪些改变?

在 MSDOS 或 PC DOS 操作系统下,必须对操作系统的两个缺省值进行改变。在一个名为 CONFIG.SYS 的 ASCII 文本文件中重新设定两个系统参数就可以做到这一点。

CONFIG.SYS 是启动 DOS 时自动执行的,它在 AUTOEXEC.BAT 之前执行,必须放在系统盘上。要改变的两个系统参数是: DOS 可以使用的缓冲区数目和 DOS 可以同时打开的文件数目。CONFIG.SYS 文件在 dBASE III 文件应用中应包括这样两行:

```
BUFFERS=24
```

```
FILES=20
```

应当注意的是,每增加一个缓冲区就会使 DOS 在内存多占 528 个字节。由于这样会使 RAM 的开销增加,故在内存容量较小的情况下应设置较少的缓冲区。然而,如果想充分使用 dBASE III,计算机的内存容量应允许设置一定数量的缓冲区。实践证明,设置 10~24 个缓冲区系统就可以运行较快了。

DOS 2.0 以上各版本一般设置可打开的文件个数为 20。当超过约定值 8 以后,每增加一个文件就使 DOS 增加 39 个字节。在这些打开的文件中, DOS 保留 5 个分别用作标准输入、标准输出、标准错误输出、辅助设备和标准打印机。这就意味着,在整个系统里各种类型的文件最多可以同时打开 15 个,包括 dBASE III 在内。dBASE III 应用程序只可以打开 13 个文件。这是因为, dBASE III 本身用了两个,一个供主程序用,另一个供覆盖文件用。

dBASE III 运行时没有这个配置要求,但当缓冲区数目增加时,它的速度也会大大提高。文件数的调置并不影响 dBASE III,因为它是直接处理文件控制块 FCB。因而允许在 dBASE III 内同时打开 16 个文件,不包括 dBASE III 本身在内。

汉字 dBASE III 的运行环境在中文操作系统支持下,在启动操作系统时, CONFIG.SYS 文件同时被执行,从而完成了对系统的初始化设置。

2. 在 dBASE III 系统盘上 CONFIG.DB 文件起什么作用?怎样建立和使用该文件?

在 dBASE III 系统盘上的 CONFIG.DB 文件是对 dBASE III 的运行环境作初始设置。CONFIG.DB 中的内容是一些具有特殊格式的命令行,当 dBASE III 启动时,系统自动查找 CONFIG.DB 文件,若能找到,就执

行 CONFIG.DB 文件中的命令,执行后回到命令状态;若找不到 CONFIG.DB 文件,则系统按既定(默认)初始化 dBASE III,然后进入命令状态。

CONFIG.DB 是标准文本文件,可以用任一字处理或编辑程序建立,也可以在操作系统的提示符下键入以下命令来建立:

```
C>COPY CON:CONFIG.DB<
```

```
DEFAULT=C:<
```

```
TEDIT=EDLIN<
```

```
WP=WS<
```

```
:
```

```
^Z<(存盘)
```

CONFIG.DB 中的命令可以根据需要自行选择。

在实际应用中, CONFIG.DB 可以实现下面的功能:

(1) 设置用户文件所存放的默认盘

命令格式:

```
DEFAULT=(盘符)
```

这样设置后,在 dBASE III 的所有操作中,如果文件名前不加盘符,系统都将在所定义的磁盘上进行文件的存取。

(2) 设置屏幕的颜色

命令格式:

```
COLOR=<标准显示>[,<加强显示>][,<边框>]
```

例如希望把屏幕设置成:标准部分为蓝色,其背景为黄色,加强显示时显示红色,背景为白色,并且处于闪烁方式下,边框为绿色,则:

```
COLOR=B/GR,*R/W,G
```

或者

```
COLOR=1/6,*4/7,2
```

(3) 设置功能键

命令格式:

```
<功能键名称>=<命令>[,<命令>;...]
```

其中<功能键名称>指的是键盘上左面 F2 至 F10 键位,<命令>为 dBASE III 命令,分号代表回车键。例如:

```
F2=USE CTSU;LIST;
```

该命令行定义按 F2 键一次等效于执行下列两条语句:

```
USE CTSU
```

```
LIST
```

可以用类似的方法定义其他功能键(F1 键不能定

义),以使得以后的键盘操作方便。

(4)设置 SET 命令的 ON/OFF 状态

例:TALK=ON

该命令使 TALK 为开,也就是说进入 dBASE III 时就能自动执行 SET TALK ON 语句。

(5)设置 dBASE III 系统的最大可用内存

命令格式:

MAXMEM=<数值>

其中<数值>范围为 200K~720K,该命令为 dBASE III 运行时规定了最大可用内存数范围。

例:设 IBM PC/XT 配置有 640K 的内存容量,希望给 dBASE III 系统 540K,以便在 dBASE III 系统中运行较大的外部高级语言程序。CONFIG·DB 命令行中可设置一条语句。

MAXMEM=540K

这种命令的使用随情况而定,有时使用 540K 内存运行外部文件时,系统还显示“无足够的内存空间”的提示。此时可以用 MAXMEM=640K 命令将可用内存设置到最大限度。

(6)规定内存变量可占用的最大内存空间。

命令格式:

MVARSIK=<数值>

其中<数值>范围为 1K~31K,该命令为 dBASE III 运行时规定了内存变量可占用的最大空间范围。

dBASE III 允许内存变量为 256 个,空间为 6000 字节,但由于有时某些内存变量占据字节数太多,所用变量个数虽未超过 256 个,但总字节数已经超过了 6000 字节,在这种情况下需要首先扩大内存变量所允许使用的内存空间数目。

例:MVARSIK=20K

此命令使内存变量可使用的内存扩大到 20480 个字节。保证程序的顺利执行。

(7)规定 dBASE III 使用的编辑方式

命令格式:

TEDIT=<程序名>

WP=<程序名>

规定 MODIFY 命令所用的编辑程序和建立或修改明细型字段所使用的编辑程序。

我们知道 dBASE III 可以用三种方法进行编辑(EDLIN),第三种是字处理程序(WORDSTAR)。未规定时,dBASE III 默认编辑为它本身所带字处理程序,但我们可以用上述命令重新规定编辑程序。

例如:TEDIT=EDLIN

WP=WS

这样执行“MODIFY COMMAND<文件名>”时,自动调 EDLIN 行编辑,编辑明细型字段时,装入字处理程序 WORDSTAR。

(8)进入 dBASE III 系统后立即执行特定的命令。

命令格式:

COMMAND=<命令>

功能是执行列出的各<命令>

例:COMMAND=CLEAR

系统进入 dBASE III 自动执行 CLEAR

在 CONFIG·DB 文件中,如果将上述命令放在一起就形成一个命令串。例如:用 EDLIN 建立 CONFIG·DB 文件,其内容如下:

DEFAULT=A

DECIMALS=3

✓ F2=USE CTSU;LISI;

MAXMEM=640K

MAVARISZ=20K

TEDIT=EDLIN

WP=WS

COMMAND=CLEAR

进入 dBASE III 后,执行 CONFIG·DB 文件,使得系统配置为:指定当前磁盘驱动器号为 A,小数点后取三位;F2 功能键代替 USE、LIST 命令;dBASE III 最大可用内存为 640K;内存变量内存区为 20480 字节,修改命令文件时用 EDLIN 文本编辑程序。最后自动执行 CLEAR 命令。

3. 汉字 dBASE III 在硬盘 C 的根目录上,信息管理系统在 A 盘上,如何设置才能正常工作?

当从 A 驱动器启动 CC-DOS 或者 PC-DOS 时,当前工作驱动器系统默认为 A。驱动器号可以为 A:、B:或 C:。

汉字 dBASE III 系统若在 C 盘的根目录上,信息管理系统若在 A 盘上,为了调试 A 盘上的程序,我们利用定义默认盘的命令 SET DEFAULT TO 来实现,进行如下的操作:

C>DBASE

.SET DEFAULT TO A:✓

此后,A 盘即为系统 dBASE III 的默认盘,调试程序时无需再指明盘号。

或者在 CONFIG·DB 文件中进行设置。设置的方法用字处理程序或编辑程序建立,或用其它方法建立。

例如:C>EDLIN CONFIG·DB

* I✓

1:DEFAULT=A:✓

2:✓

:~ Z✓

* E✓

C>

一旦对 CONFIG·DB 文件进行改变,需要重新启动 dBASE III。

4. 在进行信息查询时,被查询的内容确实存在,但有时就是查不到,为什么?

出现这种情况往往是由于当前盘和系统定义的默认盘不一致而造成的。每当汉字 dBASE III 被启动之后,汉字 dBASE III 即自动地在其所在盘上寻找是否有 CONFIG·DB,若有这个文件,则首先执行 CONFIG·

DB 所包含的各条命令,执行完,进入汉字 dBASE III 命令状态;如果没有,便立刻进入汉字 dBASE III 命令状态。若被查询的内容确实存在,但又查不到,解决的方法是将当前盘和系统默认盘取得一致,或者被查询的文件放在默认盘上,就不会发生查找不到的现象。

5. 运行汉字 dBASE III 程序时,在什么情况下会出现“内存空间不够”的提示信息?

汉字 dBASE III 的运行环境,最好要有 640KB 内存空间的支持,但在 512K 内存空间的条件下,基本上也可以运行。不过在这种情况下,必须小心谨慎地进行程序设计,否则,在程序运行过程中,系统会提示“内存空间不够”,程序再无法向下运行。在信息管理系统实际开发过程中,出现“内存空间不够”的提示,归纳起来有以下三种情况:

(1)利用 RUN 命令执行 CC-DOS 命令。因为 CC-DOS 命令是编译过的目的程序,它的执行需要将目的程序一次全部调入内存空间,因为内存空间已经存放了汉字 dBASE III。

(2)命令文件中,出现语法错误。汉字 dBASE III 再将“错误信息提示”模块调入内存空间,这种复盖模块要比其他复盖模块的容量大,因此,也往往出现“内存空间不够”的错误信息提示。

(3)同时打开多个文件,特别是又要打开索引文件,这时也会出现上述情况。

所有这三种情况,只要我们在程序设计过程中认真处理好,在 512K 内存空间的条件下,仍然是可以运行汉字 dBASE III 的。

6. 当需要打开多个数据库时,dBASE III 系统显示“打开文件太多”的提示信息是怎么回事?如何解决?

出现这个问题主要原因是没有修改中文操作系统 CC-DOS 中的系统结构设置文件 CONFIG.SYS 内容。

若 CC-DOS 中的 CONFIG.SYS 原来的内容为:

DEVICE=ANSI.SYS

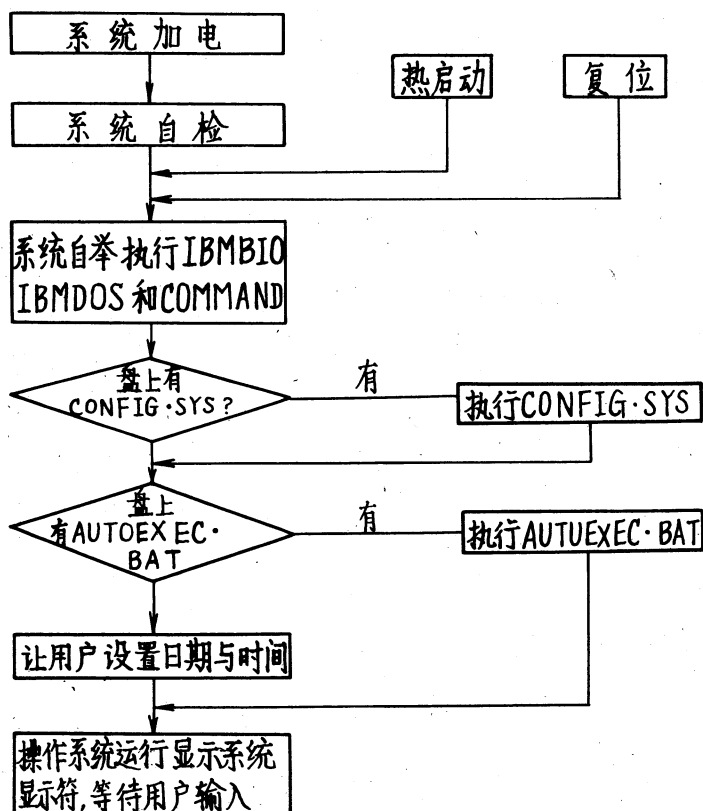
应修改为:

DEVICE=ANSI.SYS

FILES=20

BUFFERS=24

在未经设置的情况下,DOS 同时可以打开的文件个数为 8,而扩充的输入输出操作将占去 5 个文件,留给 dBASE III 用的文件个数只有 3 个,显然这是不够的。运行 dBASE III 需要设置 FILES=20,此时汉字 dBASE III 可以打开各类文件 15 个;不仅如此,修改后的 CONFIG.SYS 还必须和 CC-DOS 放在一张磁盘上,否则,CONFIG.SYS 仍然不起作用,这是由于 CC-DOS 的启动过程所决定的。CC-DOS 的启动过程如图所示。



从上图可以清楚地看到,CONFIG.SYS放在其他磁盘上,例如放在汉字 dBASE III 系统盘上,CC-DOS 启动过程中并不执行此文件,必须放在 CC-DOS 启动盘上。

7. 在硬盘不同子目录下运行 dBASE III 命令文件有什么缺点? 怎样克服?

在操作系统下先进入某个子目录,调用 dBASE III 系统,然后再运行另一个子目录下的 dBASE III 命令文件。若需要继续运行另一个子目录下的 dBASE III 命令文件,往往是退出当前的 dBASE III 状态,退出该子目录,然后再进入另一个子目录,调入 dBASE III,再运行该子目录下的 dBASE III 命令文件。这样多次调用 dBASE III 会浪费用户的大量时间,而且在不同的子目录下至少要存放两个 dBASE III 系统文件:DBASE.EXE 和 DBASE.OVL,占去了硬盘不少的空间(约 260K)。

解决这种问题的方法是调用根目录下的 dBASE III。进入 dBASE III 状态,然后进入子目录,运行该子目录下的 dBASE III 命令文件。若继续运行另一子目录下的命令文件,退出当前目录状态,进入该子目录,运行该子目录下的 dBASE III 命令文件。但在操作时,为保证 dBASE III 系统只调用一次,在进入和退出子目录操作中要在 dBASE III 系统下进行,为了做到这点使用 dBASE III 的 RUN/! 执行 dBASE III 外部程序的命令。这样就要求内存容量大于 640KB 的机器。

8. 多个子目录怎样公用 dBASE III?

可以这样进行操作:

(1) 把 dBASE III 系统盘插入 A 驱动器,并将 dBASE III 文件复制到硬盘的根目录下。

C>COPY A:*. * C:

(2) 进入 dBASE III

C>DBSAE

(3) 进入所要的子目录

.RUN CD/子目录名

这样即进入了 dBASE III 又进入了所要的子目录中。很显然要使硬盘上的若干子目录公用一个 dBASE III,借助于 RUN 命令即可解决。

9. 在 dBASE III 系统下如何对子目录文件进行操作?

为了避免一机多人使用造成数据和文件丢失的现象,常使用的办法是建立子目录文件。使用子目录对子目录文件操作运行速度也较快。在 dBASE III 命令状态下对子目录文件进行操作可采用以下几种方法:

(1) 在 dBASE III 园点“.”提示符下,建立子目录操作:

.RUN MD\一级子目录文件名\二级子目录文件名

假若一级子目录文件名为 DBASE,二级子目录文件名为 ABC,其操作方法是:

.RUN MD\DBASE\ABC

(2) 使子目录上的文件或运行程序变为当前目录操作:

.RUN CD\DBASE\ABC

若对文件直接进行编辑,可进行如下的操作:

.MODIFY COMMAND 文件名

若运行程序,可进行如下操作:

.DO 文件名

(3) 直接对子目录进行操作

若编写程序文件操作如下:

.MODIFY COMMAND\DBASE\ABC\文件名

若运行程序,其操作如下:

.DO\DBASE\ABC\文件名

若打开数据库文件,其操作如下:

.USE\DBASE\ABC\文件名

.LIST

(4) 使用批处理直接进入子目录

如果当前目录没有批处理文件时,可以随时建立批处理文件。其操作如下:

.MODIFY COMMAND 文件名.BAT

CD\DBASE\ABC

CLS

批处理文件建立好之后,在以后的使用中十分方便,只需在 dBASE III 命令状态下键入 RUN 命令,批处理文件的文件名,而后打回车键就可进入子目录。其操作如下:

.RUN 文件名(这里的文件名指批处理文件名)

10. 使用建库命令时,文件的扩展名是否必须为 .DBF?

不必。在使用建库命令 CREATE 时,若省略扩展名,则系统自动为其加上扩展名.DBF。用户也可以用别的扩展名,但在建库文件时,扩展名部分不能省略。为简便起见,用户一般不必另起扩展名。

11. 数据库文件结构的特点是什么?

DBF 文件由两大部分组成。第一部分是文件结构的说明,第二部分是实际的数据。

(1) 文件结构说明部分

文件结构说明部分的长度为:(文件字段数+1)×32+2。前 32 个字节存放文件结构中除字段描述信息外的其它数据。有关字节的内容及含义见表 1-1。

表 1-1 数据库文件结构

字 节	内 容	含 义
1	83H 或 03H	DBF 文件标识符, 含 M 型字段为 83H, 否则为 03H
2-4	日期	最后一次修改 DBF 文件的日期, 依次为年、月、日
5-8	记录数	DBF 文件中当前记录个数, 低位在前, 高位在后
9-10	地址	DBF 文件实际数据的相对起始地址
11-12	记录数	存放记录长度, 低位在前, 高位在后

从第 33 字节开始, 存放各段说明部分, 每字段占 32 个字节, 有关字节内容及含义见表 1-2。

表 1-2 字段说明部分有关字节内容及含义

字 节	内 容 及 含 义
1-10	存放字段名, 不足 10 个后补 00
12	类型标志(N, C, D, L, M)
17	字段宽度
18	若字段为实型, 存放小数位数, 否则为 00

文件结构说明部分除字段名和类型标志用 ASCII 码表示外, 其余均为 16 进制数。文件说明部分以 0DH, 00H 为结束标志。

(2) 数据部分

DBF 文件的实际数据紧接 0DH, 00H 之后存放。每个记录的第一个字节为记录删除标记, 若用 DELETE 命令删除记录(未用 PACK 命令), 则其内容为 2A(* 的 ASCII 码), 否则为 20(空格), 其后连续以 ASCII 码形式按顺序和长度存放各字段数据, 其中字符型的数据是左对齐, 数值型的数据是右对齐, 不足字段长部分用空格对齐。每个字符占一个字节, 数据部分以 1A(CTRL-Z)结束。

知道 dBASE III DBF 文件中数据安排后, 就可以用调试程序(DEBUG)查看或修改有关字节的内容, 达到恢复 DBF 文件(用 USE 可以打开)和恢复数据的目的; 也可用改变 DBF 文件标识字, 记录计数器或其它有关字节的内容给 DBF 文件加密。

12. 建立数据库可通过哪些途径?

建立数据库可通过多种途径, 归纳起来有三种方法, 即直接用 CREATE 命令建立新库; 用 COPY 命令将已建旧库改造成新库; 将各高级语言建的数据文件转换成新库。分 6 种途径:

(1) 第一次建新库

使用 CREATE 命令即可, 这是大家比较熟悉的一种方法。

(2) 由已建立的旧库改造成新库

① 照旧库原样复制出新库

格式:

.USE<待复制的旧库名>

.COPY TO <新库名>

② 复制旧库的结构到新库

格式:

.USE<被复制的旧库名>

.COPY TO <新库名> STRU

③ 仅复制旧库中的几个字段到新库

格式:

.USE<被复制的旧大库名>

.COPY TO <新库名> STRU FIELDS <字段名

清单>

④ 复制部分结构连同部分记录到新库

格式:

.USE<旧库名>

.COPY TO <新库名> FIELDS <字段名清单

><FOR(条件)>

⑤ 保留旧库全部内容, 同时需增加字段名

格式:

.USE <旧库名>

.COPY TO <备份库名>

.MODIFY STRU

.APPEND FROM <备份库名>

.ERASE <备份库名>

⑥ 保留原有数据, 并改变字段名

格式:

.USE <旧库名>

.COPY TO <暂存外部文件名> SDF

.MODIFY STRU

.APPEND FROM <暂存外部文件名>.TXT SDF

.ERASE<暂存外部文件名>.TXT

(3) 由两个相关的数据库联接产生第三个新库。

格式:

.SELECT 2

.USE<第二个库名>

.SELECT 1

.USE <第一个库名>

.JOIN TO <新库名> FOR <条件表达式>
(FIELDS <字段名清单>)

(4) 由一个旧库产生一个同类项合计新库
格式:

.USE <旧库名>

.INDEX ON <关键字段名> TO <索引文件名>

>

.USE <旧库名> INDEX <索引文件名>

.TOTAL <新库名> ON <关键字段索引> (FOR
<表达式>) (FIELDS <字段名表>)

(5) 从间接数据库产生新库

格式:

.CREATE <新库名> FROM <间接库名>

间接库名可用如下格式建立:

.USE <旧库名>

.COPY TO <间接库名> STRU EXTE

(6) 更改旧库名而变成新库

格式:

RENAME <旧库名> TO <新库名>

建立数据库的时候,用户可根据应用的具体情况,
选择上述建库的途径。

13. 有时打开 dBASE III 数据库时屏幕显示“不是数据库文件”的信息,而用 DOS 的 DIR 列目录时仍为数据库文件,扩展名为 DBF,原因何在?

产生这现象的原因很可能是:

在用 USE 命令打开此数据库前,曾用过 dBASE III
的 EDIT 命令修改过记录内容,或用 APPEND 命令追加
了记录,亦即往数据库中写入了内容,随后即没有用
USE 命令关闭此数据库文件,也没有用 QUIT 命令退
出 dBASE III。那末在下次打开此数据库时,dBASE III
就可能不认为这是数据库文件了。

顺便说明一下,USE 命令不带任何参数时,为关闭
原先打开的数据库文件。QUIT 命令用于关闭所有数
据库文件和命令文件,并回到操作系统。

14. 在使用汉字 dBASE III 时数据库数据的丢失是怎样造成的? 如何解决?

数据库数据的丢失常常是由于下列几种情况造成:

(1) 没有关闭数据库文件

在 dBASE III 中,输入和修改数据常用的命令有:
APPEND (BLANK), INSERT (BLANK) (BEFORE),
BROWSE, DELETE, PACK 等。这几条命令有一特点,就
是每执行一条新的命令后,才将前一命令的执行结果
写盘。如果不了解这一点,只顾修改,输入操作,而没考
虑到数据是否真正存盘,就会出问题。比如,若最后执
行 APPEND 命令,执行完后就抽出盘片或关机,那么
由 APPEND 输入的的数据没有写盘,这部分数据就将
丢失。解决的办法是在最后增加一个关闭现有数据库

文件的操作。

(2) 使用 INSERT (BEFORE) 命令时,对 dBASE III
1.0 版会丢掉最后一条记录。

解决办法是在执行 INSERT (BEFORE) 之前,先执
行 APPEND BLANK 命令。

(3) 使用 INSERT (BEFORE) 没有建立正确文件结
束标记。

dBASE III 1.00 版本中,执行 INSERT (BEFORE) 不
能建立正确的文件结束标记。

例如:

.USE <库文件名>

.GOTO BOTTOM

.DISPLAY 姓名,性别

.APPEND BLANK

.INSERT BEFORE

.GO BOTTOM

屏幕上显示:

end of file encountered

goto bottom?

Do You Want Some help (Y/N)?

由上可知,在执行 INSERT BEFORE 后,发出 GO
BOTTOM,显示出错误信息。因为 dBASE III 对所有非索
引文件,在执行 GO BOTTOM 后,指针指向最后一条记
录,但 EOF() 为 .F.。EOF() 为真的条件是指向最后
一条记录+1。解决办法是在执行 INSERT (BEFORE) 后,
再执行一条 PACK。

(4) 有时用 DIR 查目录时,看到文件的记录数正
确,但又找不出全部记录,指定记录号查询或显示时,
又出现“记录超出范围”的错误信息。解决办法是用
APPEND BLANK 命令找回数据。

15. 打开库后发现记录数少若干怎样处理?

一般可采用这样的步骤:

.USE <库文件名>

.GO BOTTOM

.SKIP

若能用 LIST 显示下面的记录则修复成功:

若不能奏效可以重复执行 SKIP 命令。若仍不成
功时则可用:

.USE <库文件名>

.GO BOTTOM

.DELE

.PACK

若还不成功,调用 DEBUG 调试程序,查看被破坏
的数据库的内存映像,从中常常可以找到问题的所在。
操作步骤:

(1) 将破坏的数据库拷贝到一个格式化好的空白
盘上,并将该文件改名。

(2) 调用 DEBUG 程序,查看,找出问题所在,修
改、存盘、退出。

(3) 复原文件名。

(4)进入 dBASE III。打开库文件,显示,即可检查是否修复成功。

16. 向数据库录入数据有哪些方法? 各有何特点?

数据库文件结构建立之后大量的工作是向库中输入数据,录入数据的方法有以下几种:

(1)建立数据库文件结构的同时录入数据

当数据库结构建立完毕,dBASE III 系统询问:现在就输入数据记录?(Y/N)。这时如果按“Y”键,dBASE III 则显示第一个记录的全部字段名,光标停留在第一个字段名后,我们就可以一个字段,一个字段地,一个记录,一个记录地输入数据,当输到明细型字段时,按 CTRL+PgDn 键则进入 dBASE III 字处理,在字处理方式下,便可给明细型字段录入数据,录入完毕,按 CTRL+W 则退出文字处理状态,输入下一个字段数据,如此这般进行下去。这种方法适用于初学者。

(2)简易录入法

所谓简易录入法就是利用 APPEND 命令在当前工作库文件或索引文件末尾增加一些新的记录,一次增加记录的数目是任意的,只要总的库文件内容中超出允许的最大容量就行。

操作:

.USE <库文件名>

.APPEND

采用全屏方式向库文件或索引文件添加新的记录。

(3)调用格式文件录入法

利用 dBASE III 的格式文件(.FMT)的方法,就是采用 APPEND 等命令调用格式文件的方法。这种方法好象以卡片形式向库文件添加新的记录,这样,显示的画面比较直观、舒适、漂亮。

操作的步骤:

①打开库文件。

.USE<库文件名>

②打开格式文件

.SET FORMTA TO <格式文件名>

③用 APPEND 命令调用格式文件录入数据。填完最后一个数据后,按回车键,表示一个记录输入完毕,dBASE III 则显示下一个记录,再重复上述输入数据的操作。

(4)修改录入法

修改录入法就是将 APPEND BLANK 和 READ 配合起来使用,先向库文件追加一个空记录,然后再用 READ 命令读入键盘输入的数据,并用此数据修改空记录的内容。

操作步骤:

①打开库文件

.USE<库文件名>

②追加一个空记录

.APPEND BLANK

③打开格式文件并用 READ 命令读入一组数据

.SET FORMAT TO <格式文件名>

.REAN

如果还要追加数据,再追加一个空记录,重复上一次的操作过程。这样追加的记录数可以是任意的,如果编写一个循环程序,就可实现数据的自动录入。

这种方法要注意的是用 READ 命令不能进入 dBASE III 字处理状态,所以无法向明细型字段录入数据。

(5)传递快速录入法

这种方法是把上一个记录的内容传递到下一个记录里。如果这两个记录的内容完全相同,那么,把上一个记录传递过来以后,只要将不同的部分稍作修改,就实现了一个记录的数据输入。从而提高了数据的输入速度。

操作步骤:

①打开库文件

.USE<库文件名>

②设置记录传递开关

.SET CARRY ON

.APPEND

这种传递功能很有实用价值,如果两个相邻记录有很多数据项内容相同,只要对上一个记录作少许修改,就可得到下一个记录。

如果将记录传递开关设置成关闭状态,dBASE III 将不进行记录的传递,在屏幕上显示的是空记录。

例如进行如下的操作:

.SET CARRY OFF

.SET FORMAT TO <格式文件名>

.APPEND

在屏幕上仅显示一个空记录。

17. 加快数据输入有哪些措施?

一个数据库文件的建立,归纳起来无非是做两个方面的工作:一是建立数据库文件的结构;二是输入数据。当输入数据多,尤其输入汉字信息多时,加快数据输入的速度感到尤为必要。实践证明可以从以下几个方面考虑:

①使用 SET FUNCTION 命令

这个命令用来定义功能键的功能。键盘的 F1~F10 10 个功能键除 F1 不能改变其功能外,其他几个功能键均可由用户使用 SET FUNCTION 命令来改变。该命令使用的格式:

SET FUNCTION <字数> TO <字符串>

每个被改变的功能键的功能可以定义成含有 30 个字符的序列,这些字符可以是汉字,数字及各种符号。这样按一次功能键便可以输入一串预先定义的字符。9 个功能键即可以预先定义成 9 个常用字符串。例如,若数据库文件有“单位名称”这个字段,那么可以预先先把 9 个常出现的单位名称定义在 9 个功能键上,当光标在字段“单位名称”下时,按下某一定义后的功能键,即可一次输入这个字段。功能键所定义的内容可根据

据需要及时更换。

②利用替换命令 REPLACE

输入数据时,常常有许多字段内容相同,若输入的数据是以汉字形式出现,输入工作就更麻烦,可以用简单的符号来代替,最后用 REPLACE 命令来更改。

例如,“大豆品种资源数据库”中有一字段“花色”,在这字段下的数据要么输入“紫花”,要么输入“白花”,在输入时“紫花”处用“Z”代表;“白花”处用“B”代替,2000个记录的数据都输入完后,最后,用 REPLACE 命令进行替换。这样可节省很多输入数据的时间。

上述问题的实现可以这样做:

USE<库文件名>

REPLACE ALL 花色 WITH “紫花” FOR 花色 = “Z”

REPLACE ALL 花色 WITH “白花” FOR 花色 = “B”

③应用 SET CARRY ON 命令

此命令可以免去输入相邻两个记录字段内容相同的数据。执行 SET CARRY ON 后,当执行 APPEND 命令时,将把前面一个记录的数据带到新记录中,当执行插入命令 INSERT (BLANK) 时,将把当前记录的数据带到新记录中去。这时输入新记录的操作就变成了修改记录中的有关数据了。

④应用 APPEND FROM 命令

APPEND FROM <文件名> (FOR<条件>), (SDF)

(DELIMITED)命令是从其他文件追加记录到当前数据库文件之中。利用这一特性,对需要输入数据的数据库中某些字段及其内容与原有的数据库的对应字段及其数据相同时,不必重新输入这些数据,只需用上述命令,将这部分数据追加到已打开的数据库中,然后对其他字段输入相应的数据即可。

18. 明细文件的扩展名是否一定为 .DBT?

是的。明细文件的扩展名也是系统自动加的。当用户所建的库中有明细型字段时,存盘后就自动在盘上生成一个与库文件名同名其扩展名为 .DBT 的明细文件。

19. 明细字段里的空格是否占用空间?

是的。明细字段里的 1 个空格占 1 个字符的空间。因为明细字段的数据均可以用一段文字来表示,而这段文字是按 512 个字节为一个数据块来存放的。明细字段的数据最多允许 4K 字符,即 8 个数据块。当写入的文字为 512 个字节时,系统就为此明细字段分配一个数据块。当写入的内容大于 512 个字节而小于 1024 个字节时,系统就分配两个数据块。

20. 使用 Memo 明细型字段节省存储空间吗?

这要看 Memo 型字段中的内容而言。如果 Memo 型字段中的内容较少,则并不节省存储空间。这是因

为,一旦定义一个 Memo 型字段,则不管其中有无内容,系统都要开销 512 个字节的存储空间。也就是说,又要有一个 .DBF 文件,哪怕它是空的,也至少要占据 512 个字节的盘空间。dBASE III 为 Memo 型字段分配磁盘空间是以块为单位进行的。每个块为 512 个字节的 1 个扇区。最多可有 8 块,即 4096 个字节。

21. 怎样输出 Memo 明细型字段的内容?

dBASE III 在数据库文件中增加了 Memo 型字段,这对处理内容较多且长短相差较大的字符数据提供了方便。但是,dBASE III 提供的输出 Memo 字段的手段较少,能显示或修改 Memo 型字段的命令只有 6 条,即 EDIT,CHANGE,?,DISPLAY,LIST 和 REPORT。其中,EDIT 和 CHANGE 主要用于编辑和修改 Memo 型字段的内容。?,DISPLAY 和 LIST 主要用于输出 Memo 型字段的内容。这 3 条命令中,? 命令更适合于在程序设计中使。只要在? 命令中给出 Memo 型字段的名字,便可输出其内容。如果用? 命令造表,则可以在表格中输出 Memo 型字段的内容。REPORT 命令是专用于输出报表的。在用该命令建立报表格式时,只需针对要求输入栏目内容的提问键入需要输出的 Memo 型字段的名字,便可在报表中输出 Memo 型字段的内容。

22. 怎样使用行编辑程序或字处理软件 WORDSTAR 编辑 Memo 字段?

dBASE III 提供了自己内部的字处理程序用以编辑 Memo 型字段,但它的功能有限,最多只能编辑 4096 个字节的文件。dBASE III 还提供了另一种方法:在不退出 dBASE III 的情况下调用 EDLIN 或 WORDSTAR。这是通过在 CONFIG·DB 文件中设置如下两个参数来实现的:

WP=<外部程序名>

TEDIT=<外部程序名>

WP 和 TEDIT 定义了用于编辑 Memo 型字段的外部程序。在用 EDLIN 或 WORDSTAR 编辑 Memo 型字段的情况下,这两个参数设置如下:

WP=EDLIN

或

TEDIT=WORDSTAR

但是,一旦进行了上述设置,在 dBASE III 系统下打入:

MODIFY COMMAND <文件名>

正常情况下应当进入到相应的编辑状态,或者是行编辑状态,或是字处理状态。不过有时候出现这样一种现象,执行上述命令后没有进入到所指定的编辑状态,而是又返回 dBASE III 命令状态,即圆点提示符的状态。这是为什么呢?当出现这种现象时候要检查所指定的编辑软件是否在指定磁盘上存在,若指定的磁盘上不存在该软件,就返回 dBASE III 圆点提示符状态等待。



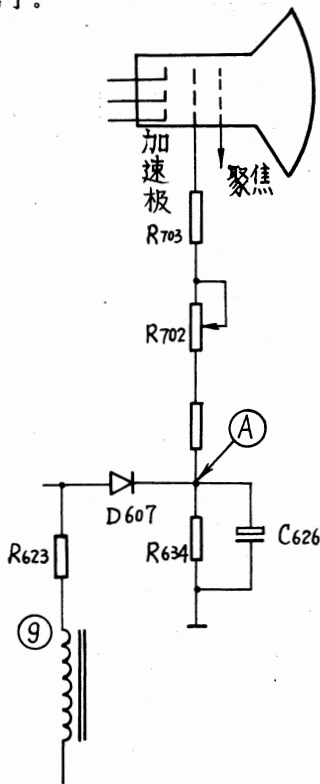
彩色显示器亮度失控的探讨

中国人民大学信息中心 胡野红

例 1: 亮度失控

亮度失控是显示器常见故障之一,而造成这种故障的原因大部分都是忽视了机器的使用环境如:机房温度过高或湿度过高。也有少量的显示器故障是由于设计不合理或使用了质量不好的元器件所致。本文对后一种原因所引起的故障结合两个实例予以分析,供大家参考。例 1. 故障现象为亮度下不去,调节亮度电位器不能使屏幕的亮度发生变化此故障多次出现在 BRI-UP 彩显中。

分析与维修:首先检查了亮度控制电路没有发现异常现象,再检查显像管加速极电路发现该极电压高达 700V,而正常工作时该极电压为 420V,由以下附图可以看出,加速极电压出自高压包 T 的 ⑨端经 R623 限流, D607 限流 R634 和 R701 分压而得。进一步检查发现 A 点电压为 640V,而正常时应为 200V。这时可以明显看出,故障范围在 A 点以下电路,结果是 R634 开路和 C626 失效。换上同等参数的元件后该彩显工作正常了。



该彩显修好用了一段时间后旧戏重演,再次出现亮度失控现象,经查还是 C626 失效,原电路中 C626 为 $4.7\mu\text{f}/200\text{V}$ 而在实测中 A 点正常工作电压为 200V。很明显 C626 耐压不够。由电工原理而知 C626 的耐压应选 $\sqrt{2} 200(\text{V}) = 282(\text{V})$,故选用标称值为 $4.7\mu\text{f}/300\text{V}$ 的电容换到该显示器上后,彩显一直正常工作至今未发现这种故障。

仔细分析这部分电路可以发现:①高压包 T ⑨端的另一端接行输出管,而行输出的行脉冲虽经 D607 整流,但到达 A 点时还是脉动直流,对 C626 形成冲击。②由于高压包 T 是储能元件,每次开关机时均会产生数倍于正常工作电压的反电势,形成对 C626 的冲击。又由于反电势虽然电压高,但能量小,不至于数次开关机就损坏 C626,而长而久之的冲击使 C626 失效而起不到滤波作用。所以由以上事实可以得出结论:由于原线路对 C626 的耐压选择不当,造成 C626 失效、分压电阻 R634 开路 A 点电压升高,使显像管加速极电压脱离工作范围而引起显示器亮度失控。

例 2:故障现象为刚开机时光栅抖动呈浅绿色且光栅行幅不满,调节亮度电位器光栅亮度无变化,几分钟后工作正常。

显示器牌号为 BRI-UP

分析与维修:先检查亮度控制电路未发现异常现象,再用万用表测机内直流电源输出端,发现各挡电压均在低于额定电压 30V 内随光栅抖动而变。故怀疑电源带载能力差。用替换法处理证明电源无问题。又怀疑行扫描电路有问题,结果行管、逆程电容等有关元器件均完好无损,断开行输出后电源电压工作正常,这说明短路现象发生在行输出以后部分。而行输出的最大负载在显像管,因为屏幕为浅绿色故怀疑 G 极(显像管)线路有问题,经仔细检查, G 极线路无问题。当把 G 级引线从显像管尾部线路板上焊下时,光栅满幅电源电压也工作正常了。所以怀疑显像管本身有问题。用摇表测 R、G、B 极对地电阻,发现 G 极电阻明显小于其它两极。换同一规格的显像管后,彩显工作正常了。可以看出,由于显像管 G 极对地绝缘性能变差,当开机初始、呈短路状态,使电源电压下降、光栅不满、亮度失控。随着时间的推移,显像管 G 极极板温度升高,电阻逐步增大到正常值使彩显正常工作,而关机后显像管极板逐渐变凉电阻也逐渐变小,而再次开机时,重现故障。

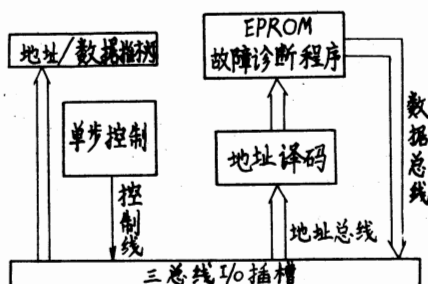
微计算机的维修工具——8088 系统板故障的诊断卡

董 珊

微型计算机的系统板是整个计算机的核心。板上有 8088CPU、ROM 和 RAM、DMA 控制器,中断控制器,定时控制器,并行接口控制器及总线控制等多种功能,这些功能大多采用集成电路芯片实现。当系统板出现故障时绝大多数情况要造成“死机”,一旦“死机”,系统板上多数芯片的输入、输出状态都不正常,尤其对那些大规模集成电路芯片很难确定故障点,给故障修复工作造成很多困难,本刊第 5 期推荐的“8088 系统板故障诊断卡”是维修 IBM PC/XT 及其兼容机系统板的实用工具。对于初次从事维修的同志,它将使你很快胜任常见故障的修复工作,对于已从事过系统板维修的同志,它将会提高你的工作效率。

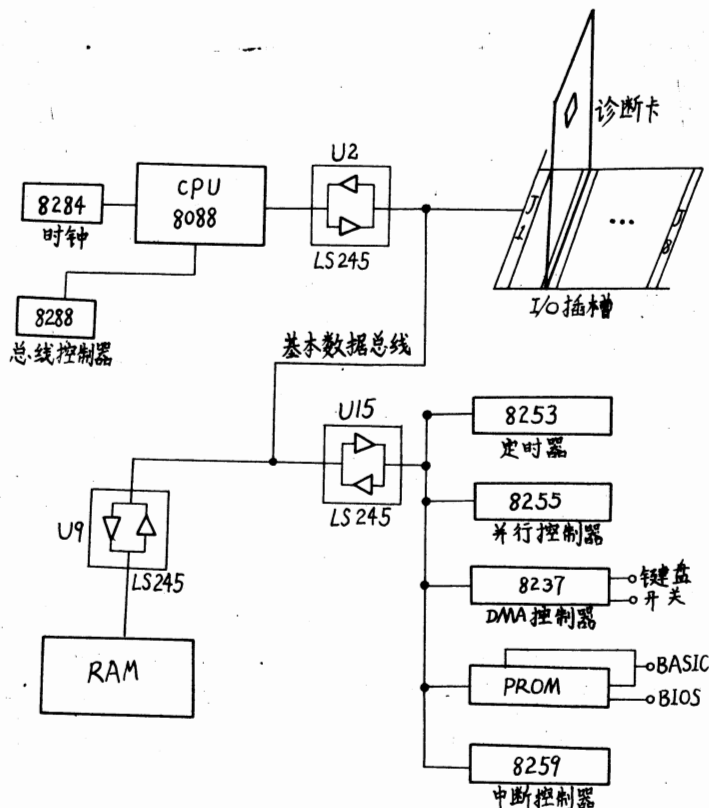
一、工作原理

8088 系统板故障诊断卡(以下简称诊断卡)主要包括三部分电路,(一)存放诊断程序的 EPROM 芯片;(二)地址译码电路;(三)单步控制电路。三部分框图如一所示。



图一 诊断卡原理图

其工作过程是:从 I/O 总线插槽取得首地址,经地址译码器译码,选中故障诊断程序相应的地址单元,然后读取该地址的内容,这一定是诊断程序第一条指



图二 IBM PC/XT 系统板数据流程图

令的内容,经数据总线送往 I/O 插槽,CPU 读取并执行该指令又送出下一个地址,取出第二条指令...如此循环。这样 EPROM 中的诊断程序将自动地诊断系统板上各部分的故障情况,并且直观地显示在屏幕上。当系统板上某些故障(约占系统板故障的 25%)影响到诊断程序正常运行时,为了进一步查找故障源,诊断卡上的单步控制线路可完成这一工作,在单步工作方式下,每执行一个总线周期停机一次,这样我们可以从地址/数据指示灯上清楚地看到地址/数据总线的故障,也可从中分析控制总线的故障。只要排除了总线故障,便可重复上述连续工作方式,全面地检查系统板上其它各部分故障。

二、从维修角度看系统板的结构及诊断卡位置

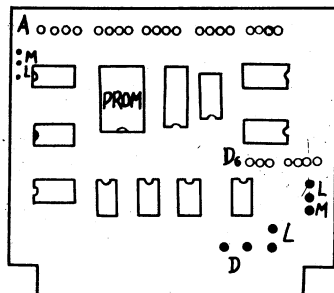
从图二中我们可得到哪些信息呢?首先看 U2,在非 DMA 方式下,数据传输由 8088CPU 控制,所有的数据输入,输出都要经过 U2;CPU 与 I/O 插槽中控制卡的信息联系要经过 U2,CPU 对 RAM 的数据读,写首先经过 U2 再经 U9 到达存储器,同样,CPU 对 PROM,8253、8255 的访问也要经 U2 和 U15,就是在 DMA 操作过程的开始,CPU 也必须经 U2 和 U15 对 8237DMA 控制器进行初始化。可见,U2 的好坏将影响整个系统板上其它各模块的正常工作,故该芯片尤其重要。

从诊断卡在整个图二中的位置可以看到,只要保证基本总线是好的,诊断卡就可以正常工作,直接对 8253、8255、8237、8259、RAM 等各部分进行故障诊断。但是必须封锁 CPU 对原 PROM 的访问。

利用图二我们还可以分析某些故障现象的故障源。如对系统板的诊断结果为 8237、8259、8255 均有故障,一般说几个芯片同时损坏的情况机会不多,可能性较大的是 U15,这在实际维修工作中会少走些弯路。

三、诊断卡的运行启动

诊断卡是系统板的维修工具,因此,在维修有故障的系统板之前先要保证诊断卡本身是完好的,所以有必要先在一个好系统板上学会正确地使用诊断卡的方法,并验证诊断卡是否完好。



图三 诊断卡外形图

在诊断卡插入 I/O 槽之前,先检查一下两种工作方式跳线开关的位置。若将跳线连接右下角 L 两点,则为单步工作方式;若跳线连接的是 D 两点,这时将工作在连续诊断方式。如图三所示。

左上角若连接跳线在 M 两点,地址发光二级管

灭,若连接跳线在 L 两点,发光二级管亮。右中下位置若连接跳线在 L 位置,数据发光二级管灭,若连接跳线在 M 位置,数据发光二级管亮,一般情况要保证发光二级管亮,以便观察地址、数据总线的状态。

在跳线开关接好之后,诊断卡可以插在系统板的任一 I/O 槽中(PC/XT 的 I/O 插槽 J8 除外)。但要注意:诊断卡的元件面要与其它卡的元件面方向要一致,千万不要插反,以免烧坏元件。

如果诊断卡上的 EPROM 芯片是 CGA 方式显示的,这时系统板上应插一块 CGA 卡,并连接相应的显示器;如果诊断卡上的 EPROM 芯片是单色图形方式显示,则应在系统板上插一块单色图形卡并连接单色显示器。两种 EPROM 在拔插替换时也应特别注意:EPROM 芯片不要插反,否则将破坏里面的诊断程序。

为了保证诊断卡上的 EPROM 能正常工作,还应将该系统板上原 PROM 芯片“置死”。实现方法是:PC 机,在系统板上将 U46(LS138)-5 脚或 U64(LS20)-6 脚断开;在 PC/XT 机上,位于系统板上靠近键盘插座附近,有两个用白色框线围起来的焊点,标称 E1(1、2),在 PC/XT 系统板图纸上的第三项标以 E7(1、2),将两点短路即可;在 0520CH 机上,将系统板上原 B105 芯片标有 01 的 PROM 芯片拔下,换上专为 0520CH 所写的 EPROM 芯片,即可完成对 0520CH 系统板的测试诊断。其它一些 XT 兼容机,并可采用这一方法,如台湾产的 PC/XT 机等。

在完成上述准备工作之后,现在可以进行加电测试,当电源开关接通后,显示器的屏幕上应立即现出大字“8088TEST”字样,之后换屏,在 PC/XT 机系统板条件下,出现如下的标准显示结果:

CPU	OK	SW1 1 2 3 4 5 6 7 8	
8237 REG	OK	0 1 0 0 1 0 0 1	
8237 CHAN0	OK	SW2 1 2 3 4 5 6 7 8	
8253 CHAN1	OK	0 1 0 0 1 0 1 1	
8259 REG	OK	* RS232(1)PORT TEST RESULT;	
* MEMORY TEST 512KB OK	8250 REG	OK	
* MEMORY.....80000 55 ERROR	8250 REG(STATUS)	OK	
SI=0000	8250 DATA TRANSFER	OK	
	8250 INTR	OK	
8259 INTRR FLAG OK			
8253-0	OK	* RS232(2) PORT TEST RESULT;	
	8250 REG	OK	
P10 DATA CHAN	OK	8250 REG	OK
P10 CONTROL PORT	OK	8250 DATA TRANSFER	OK
P10 STATUS	OK	8250 INTR TEST	OK
* P10 INTR TEST	ERROR		

对于上面结果中打“*”号的部分作如下说明

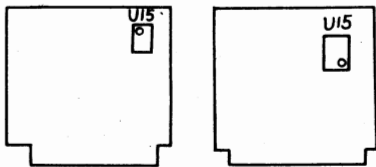
1 MEMORY TEST 512 KB...OK

MEMORY...8000 55 ERROR

此项为系统板内存实际容量的测试结果,若被测系统板实际容量发生变化,显示结果将相应地改变,如实际容量为 256KB,则该项显示变为:

MEMOR TEST 256 KB OK

MEMOR...4000 55 ERROR



图四串卡

图五串卡

2 P10 INTR TEST ERROR

此项内容表示对 P10 的中断部分进行诊断后的结果,这部分的测试程序是针对某一并行接口编号的,故对通用的并口报错,我们准备对该部分的程序作进一步修改,满足对通用并口的测试。

3 RS232(1)PORT TEST RESULT:

要在这个测试结果区显示串卡一口的测试情况,应将串行卡的 U15 跳接器按图四所示方法接插。

4 RS232(2)PORT TEST RESULT:

要在该区显示测试串行卡 2 口的结果,只要将串行卡中 U15 的跳接器取出变换方向再插入,如图五所示。若有两个 RS232 卡同时测试,只要按上述图四、图五分别插入 U15 的位置,即可对二个串行口同时测试,其中一个测的是 RS232(1)另一个检测的是 RS232(2),恰如标准结果所示。

需要对并行卡和串行卡做故障诊断时,应分别在并卡、串卡的插座插头位置上,插上诊断插头、插座,这样才可以使诊断程序正常工作,完成对并行卡、串行卡的测试,否则串、并卡显示的测试结果将都是 ERROR,测试是无效的。

这里给出的系统板诊断的标准结果,它的全部内容都直观地显示在一个屏幕上,如果发现某一项或几项与标准结果不同,则说明该部分发现了故障,如何针对具体的显示内容排除故障,我们下边分别给以介绍。

在对有故障的系统板进行加电检测之前,一般先要测一下系统板+5V 与地之间的电阻值,一般应在 300Ω 左右,最低不小于 100Ω,然后交换表笔测反向电阻,其阻值不应有太大的差异。

四、系统板上内存存储器故障点的定位

为了准确地判断内存存储器中的故障点,我们再仔细的分析一下内存显示信息:

MEMORY TEST 512 KB OK

MEMORY 80000 55 ERROR

该项信息中的 8 表示第 8 体。它是这样计算的,每 64K 的存储空间定义为一个体,现在共有 512K 的存储空间 OK,即有第 0—第 7 体共 8 个体 OK,这里出现的第 8 体已超出内存的实际容量,故报出 55 ERROR 信息,这是正常的超量错误信息。

如果待测的存储器是 512K,而测试信息显示的是:MEMORY TEST 256 KB OK

MEMORY 40000 ×× ERROR

按 64K 为一个体计算,有第 0—第 3 体共 4 个体

OK,第四体则表示为有故障的体位置。所以这一位数字表示的是从 0 开始计数的有故障的内存体。

因为每个体都有 9 个芯片组成,所以只知道哪个体有故障,并没有最后找到故障点。标准显示结果的“55”两位数字位置,将根据实际测试的结果给出不同的数字,表示出错体中,哪一位芯片出了故障,这些数字及其含意如表 1 所示。

表 1 内存故障定位表

显示数字	出错芯片
0 0	奇偶校验位
0 1	D0
0 2	D1
0 4	D2
0 8	D3
1 0	D4
2 0	D5
4 0	D6
8 0	D7

例如:当看到下面信息:MEMORY TEST 000KB OK

MEMORY 00000 04 ERROR

我们便知道是 0 体、D2 位芯片出了故障,一般情况下,为了慎重起见,把 0 体 D2 位芯片取出换到该体的其它位置上再加电测试,如果又在其它位置上给出了错误信息,则该片无疑是有故障的芯片。再比如出现:60000 00 ERROR 可以知道是第六体的奇偶校验信息片出了故障。

如果 PC/XT 机内存芯片是 256KB 的 41256 芯片,体号的计算方法不变,但坏片的物理位置应拆合为每一排芯片包括 4 个体。

当见到如下的错误信息:

MEMORY TEST 000 KB OK

MEMORY 00000 FF ERROR

这里报告 0 体 8 位芯片均有故障。但一般情况下,我们知道一个体中 8 个 RAM 存储器芯片突然同时损坏是很少见的,较大的可能是内存控制电路出了故障,我们可以检查行选通(RASX),列选通(CASX),等公用部分的电路。

利用诊断卡,我们在 PC 机上排除过这样一个故障:MEMORY TEST 000 KB OK

MEMORY 00000 00 ERROR

这里给出的信息表示 0 体的奇偶校验位坏但实际上该芯片没坏,我们从 PC 图纸上分析,引起奇偶校验错的原因还可能有两个,其一是 PCK 信号变成高电平,可能是 U96、U27、U97、U37 坏,其 2 是 I/O CHCK 信号变为高电平,可能为 U52 坏,经检查 U52—11 脚 I/O CHCK 信号“死记”,焊下 U52 芯片检查,发现 U52—13 对地短路,换 U52 芯片(LS00),机器恢复正常。