

E&C
1992
 ●一九九二年 ●总期第91期
10

電子

ISSN 1000-1077

479

與 電腦

479

1982~1992
 庆祝电子工业出版社
 成立十周年!



• ELECTRONICS AND COMPUTERS •

振兴电子工业

繁荣出版事业

电子工业出版社历年获奖图书：

- 《最优控制》、《电磁场理论》、《天线》、《系统工程引论》、《工业自动化仪表与过程控制》、《脉冲与数字电路》六种教材被评为全国电子类专业优秀教材。
- 《黑白电视机原理与维修》获1986之全国优秀畅销书。
- 《概率统计基础》被评为1986年浙江省优秀科技成果奖。
- 《俄汉翻译理论与技巧》被评为1986年黑龙江省优秀科技成果奖。
- 《电冰箱的原理、使用与维修》获1987全国优秀畅销书。
- 《彩电检修大全》被评为1988全国优秀图书。



●1991、1992年电子工业出版社在全国各大中城市举办了二届图书大联展，受到了读者的热烈欢迎。图为成都人民南路新华书店柜台。



●图为91年秋电子工业出版社参加第一届北京图书节盛况。

- 《新部首大字典》被评为1983年上海地区优秀畅销书。
- 《少儿百科知识问答1~4》获1991全国第五届图书“金钥匙”奖。
- 《用万用表修理彩色电视机》获91年全国优秀畅销书。
- 《录像机维修实例999》获91年全国优秀畅销书。

国内代号：2-888

定价：0.95元

解放思想,深化改革
多出书,出好书,为振兴电
子工业作更大的贡献。

——祝电子工业出版社成立十周年

张学东

一九九零年八月廿一日

SJW系列 三相大功率稳压器



- 获国家电力电子产品合格证书
- 获91年北京国际博览会银奖
- 稳压器标准由我厂起草制订

高效率 • 低损耗 • 波形畸变小 • 稳压精度高
• 稳压范围宽 • 运行可靠安全 • 应变时间短
• 适用各用电单位 规格：20~1000kVA

银 奖 产 品 • 配 套 出 口

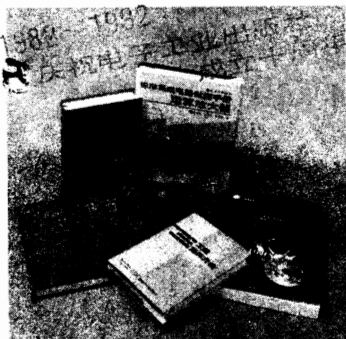


上海市精达电子仪器厂

地址：上海市新闻路579号 邮编：200041

电话：2563294 2170089 电挂：6910

质优价廉 • 现货供应 • 资料备索 • 服务完善



一九九二年

总期第91期

電子與電腦

目 录

• 综述 •

- 改革开放结硕果 齐 心(2)
软件市场大有希望 温有良(2)
新兴的教育技术—CAI 陈健(3)

• PC 用户 •

- 磁盘的簇与簇管理分析 崔来堂(4)
程序运行过程中汉字/西文输入方式的自动切换
..... 傅 雷(6)
C 语言在配平化学方程式中的应用 王 晰(7)
DISK MANGER 使用技巧 梁高军(9)
如何利用 dBASE II 的 F1 功能键 涂振宇(10)
CCBIOS 2.13 稿纸方式打印 宋 捷(11)
微机屏幕的打印和放大 谭人杰(11)
1992 年全国青少年信息学(计算机)竞赛试题 (13)

• 学习机之友 •

- BASIC 子程序的递归调用 陈继良 丘 文(15)
APPLE 机的快速排序 张 亭(16)
百年公历—农历互查 刘安军(17)
内存数据代码搜索程序 苏 华(20)

• 语言讲座 •

- 6502 机器语言程序设计
第十章 监控子程序的调用 朱国江(21)

• 学用单片机 •

- BJS—51 单片机实验系统(续) 李广弟(25)
CYSCB—ZMCS-518098 单片单板机硬件设计原理
..... 吴 微(26)

• 学装微电脑 •

- 微电脑控制微型钻床 易齐干(30)

• 电脑巧开发 •

- 计算机语音输出功能的开发与应用(下)
..... 陈竹林(35)

• 电脑游戏机 •

- 第三讲 程序结构和程序框图 于 春(38)

• 维修经验谈 •

- APPLE—II、CEC—I 故障诊断仿真系统
..... 王志刚 张一建(43)

• 读者联谊 •

- 普及型 PC 个人用户软件交流联谊活动问题解答(十)
..... 王路敬(46)

• 信息与服务 •

- C—DBAG 中文数据库系列应用生成器新产品 ... (28)
告读者作者和朋友 本刊编辑部(29)
SJW—B 系列自动补偿式三相大功率电力稳压器
..... (48)

机械电子工业部电子工业出版社主办

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

印刷:北京三二〇九厂

国内总发行:北京报刊发行局

国内统一刊号:CN11—2199

邮发代号:2—888

国外代号:M924

出版日期:每月 23 日

主编:王惠民 副主编:王昌铭

责任编辑:杨逢仪

订购处:全国各地邮电局

国外总发行:中国国际图书贸易总公司

(北京 399 信箱 邮政编码 100044)

广告经营许可证:京海工商广字 147 号

定价:0.95 元



改革开放结硕果

电子工业出版社建社十周年简介

我社自一九八二年成立,到今年十月二十二日将迎来十周年。在这十年里,由于党的改革开放方针的指引,上级领导机关的关怀和支持,全社同志克服困难,奋发努力,使我社以较快的速度健康发展,现已成为具有较强出版能力,在出版界具有较大影响的中央级科技出版社。

我社现有职工 160 多人,其中专业技术人员占 70% 以上,具有副编审以上高级职称的 25 名。年发稿能力 6000 多万字,年出书 300 多种,接近一日一书,达到人均两种书,出书码洋超过 3000 多万元,销售码洋近 3000 万元,实现销售收入约 2000 万元,人均创利超万元,据统计,十年共出版图书 2000 多种,总印数约 3500 万册,其中各类教材 400 种,专业学术专著 200 种,工程应用技术和工具书 800 种,其余为电子科普读物和直接为电子产品服务的实用读物。有数十种图书获得全国、省级科技成果奖,评为优秀畅销书。另外,还出版了《电子与电脑》杂志 90 多期,与图书配套的音像制品 30 多种,出版计算机软件上百种。这些书、刊和音像制品,门类齐全,适应不同专业,不同层次读者的需要,为我国电子工业的科研、生产、教育和电子科技成果的传播推广,以及电子科学知识的普及提供了良好的服务,受到广大读者和社会各界的好评。我社受新闻出版署委托,是高技术图书出版工作的牵头组织单位之一,电子工业出版社的知名度也随之不断提高。

目前,我社正认真贯彻邓小平同志南巡讲话精神和国务院关于加快发展第三产业的决定,决心抓住有利时机,加快改革开放的步伐,提高自我发展能力。我们的经营方针是:坚持一业为主,发展多种经营;巩固大本营,开发新基地,积极开展国内外不同形式的合作。图书出版工作无疑是我们的主业,必须以主要精力扎扎实实地去抓,这是我们不可推卸的责任,也是进一步发展和开拓的基础。我们要加强经营管理,在思想作风和业务建设等方面更上一层楼。与此同时,我们正积极探索加强国内外合作与交流的途径。一方面是出版业务的合作。如正和香港得实发展有限公司合作出版《UNIX 系统 V 第 4 版中文手册》丛书,共 49 本,2000 多万字,现已出版 9 本,其余力争年底出齐,这套丛书在我国计算机界产生了很大的影响;另外,已和美国万国科技有限公司达成协议,拟在北京合资创办“万国电子”刊物,得到了国家科委、机电部和新闻出版署的支持,现正在办理报批手续;和台湾“权岗出版公司”正在商谈加强合作的问题。另一方面是充分利用自身优势,大胆而又稳妥地在国内一些开放地区开辟新的基地,逐步拓宽业务渠道,如已与上海浦东、海南岛等地的单位商定在这些地区兴办合资企业。我们希望通过多方位的合作与交流,使电子工业出版社向国内外大市场发展。

我们深深体会到,电子工业出版社十年走过的道路是不平坦的;前面还有许多困难等着我们去克服。全社同志决心在新的形势下,发扬成绩,克服不足,同心协力,义无反顾,乘改革开放的东风,解放思想,挖掘潜力,迎接新的机遇和挑战。

本刊特约记者 齐 心

软件市场大有希望

中软总公司通用部副总经理 温有良

九十年代计算机市场发展重点是由硬件逐渐转向软件,一些先进发达国家以 25% 的增长率发展。1990 年,世界软件市场销售额 1000 亿美元,到 1991 年就超过了 1200 亿美元,其中微型机软件占 35% 以上。这些软件中最受欢迎的是文字处理软件、桌面排版软件、窗

口软件和 Novell 网络软件。

Novell Inc 宣布,该公司截至 1991 年 10 月 26 日,净收入 6.4 亿美元,与上一年同期 4.9 亿美元比,增加了 29%,由于 NetWare 3. x 网络服务操作系统软件营业额已超过了 3 亿美元,所以使得 91 年软件产品销售总额达 5.7 亿美元,增加幅度达 47%。

在我国,专家们认为软件市场潜力是很大的,可是目前启动很慢,从开发一个软件到商品化周期太长。造成这种现象的原因,一是我国软件价格便宜,开发软件

的劳动力更便宜,使得软件开发人员积极性不高,有的甚至到外国去进行劳务输出。二是国内软件开发缺乏统一组织,各单位以自我封闭作坊式进行开发,同一个产品几家都投入人力物力,有的产品市场上都有了,可是还有人再做重复性的工作。

自从软件保护条例公布以后,软件开发的积极性确实高起来了,在国内外引起了很大反响。国内一些计算机集团公司也清楚地看到了单一生产经营硬件路子会越来越窄,于是他们逐渐地转变为硬软结合,一手抓硬件产品更新,一手抓软件产品配套。这一举动也引起了一些外商注意,他们瞄准了中国软件市场,日本、美国、新加坡,还有台湾地区,他们积极地与国内一些大公司合资进行软件生产。

激烈的竞争引起国内有关部门的高度重视,决策部门重点地部署了软件生产基地,现在已确认的有南方软件生产基地和北方软件生产基地,这意味着,在中国要大力开发生产系列软件。中软总公司承担了北方软件生产基地的任务。

由于人们对硬件、软件认识观念转变了,使得计算机软件市场大门越开越大,推动了我国计算机的应用,提高了计算机应用率。广西壮族自治区在九一年底召开的全区第二届计算机应用年会上,区经委提出,检查企业达标最重要一条是,企业是否用上了计算机。还提出了计算机应用要求上新台阶,检查是否上了新台阶,最重要一条是:是否计算机联成了网络。

随着软件产业的兴起,应用计算机的领域越来越

大了,可是,给计算机市场也带来了新的问题,就是软件应用人员短缺这个矛盾越来越突出了。在国外,一般先进发达国家硬件人员与软件人员是1:1关系,而我国目前据有关资料统计,硬件人员与软件人员比例悬殊,软件人员严重短缺。

据报导,日本到2000年,将缺乏100万各类软件人员。印度虽然软件出口高速增长,今年可达1.8亿美元,全国有700多家软件企业还在瞄准世界市场,可是他们面临的最大恐慌也是各类软件人员短缺。

我国91年软件经销单位有220家,销售软件400多万美元,92年软件经销单位270家,销售软件可能在800多万美元,这个数字并不高,为什么呢?其中一个原因是有些单位买了计算机不会用,想配软件人员又没指标,只好由硬件人员去硬顶,有些该买的软件,硬件人员也不敢买。看来人员结构比例不合理仍然是影响我国软件产业发展的一大障碍。这一严峻形势,引起了许多产业集团的关注。中软总公司与日本NEC、富士通等合资开发生产软件,国家又把重点攻关项目UNIX国产化任务交给了他们,可是该公司最大的困难,仍然是软件开发人员短缺。他们采取向社会招标、自办强化培训班等,都没有从根本上解决问题。

如今,软件市场竞争很激烈,表现在产品上,实质还是人员之间的竞争,如果国家从政策上能够保证合理比例的软件人员,再加上软件保护条例能够认真地贯彻,思想再解放一点,步子再大一点,那么,中国的软件市场是大有希望的。

新兴的教育技术——CAI

吉林四平师范学院 陈健

CBE(Computer Based Education—计算机辅助教育)是计算机应用的一个重要领域;CAI(Computer Assisted Instruction—计算机辅助教学)是CBE的一个主要分支。CAI作为一门新兴的、现代化的教育技术,历史不过只有三十几年,但它所展现出的广阔前景正在吸引着越来越多的人,在世界很多国家得到越来越广泛的应用。在我国,CAI还较少为人所知。我国要实现现代化,作为其基础的教育应该首先现代化,CAI作为最新、最现代化的教育技术,应该为广大教育工作者了解和掌握,为现代化服务。

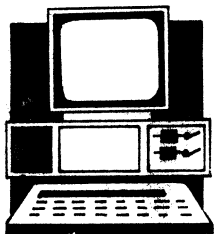
CAI起源于本世纪五十年代末。1958年,美国IBM公司设计了一个用于教授小学生二进制算术的教学系统,这是最早的教学软件,首次将计算机技术应用于教学,从此开始了CAI的历史,这一新生事物的出现,引起了很多有识之士的高度重视,一些发达国家的大学、研究院(所)和大公司纷纷投入资金和人力,开展对CAI的研究。

在美国,很多著名大学(如麻省理工学院、斯坦福

大学等)和计算机公司(如IBM公司、DEC教学设备公司)都对CAI进行了大规模的研究和实践,并推出很多计算机辅助教学系统,一些大学还制订出教学计算机化的计划。1967年,美国成立了“计算机教程公司”,专门研制和生产各种教学软件和教学管理软件,用于出售或出租,从而把教学软件推向了市场,成为商品。1984年,美国又创建了世界上第一所“电子大学”,通过远程教育网络向美国各地、加拿大和欧洲一些国家的学生教授课程,这一系统开设课程可达170门之多。加拿大的CAI研究开始于1986年,由十一所大学和一些研究院联合开发,取得了显著成果。英国在1972年制订出规划,投资200万英镑,计划开发29个辅助教育系统。日本的金泽工业大学已开发出近70门课程的CAI系统。为在发展中国家推广CAI技术,在联合国教科文组织的积极赞助下,国际信息处理协会曾多次在巴西、古巴、印度、尼日利亚等国举办有关CAI的研讨会,大大促进了CAI的发展进程。我国的CAI研究开始于六十年代,中间曾一度夭折,自1980年起,华东师大、西安交大、大连理工大学、华中工学院等数十所高等院校和一些基础好的中、小学都陆续开始了这方面的研究,取得了很好的成果。

CAI最主要的作用是教学,CAI应用于教学有两种形式:主体式(Primary CAI)和辅助式(Adjunct CAI)。

(下转29页)



PC 用户

磁盘的簇与簇管理分析

石家庄铁道学院(050043) 崔来堂

一、盘的两读方式

PC DOS 磁盘的读写,从用户介面看,分为两种方式:一种是扇区读写,用户直接选择盘扇区,由中断调用 INT 13H、INT 25H(26H)或工具软件 DEBUG 等实现,扇区是这种方式的最小可寻址单位;另一种是文件读写,用户只需给出文件名和有关命令,由操作系统自动查找组成该文件的各个数据块,完成读写,这些数据块的分布及查找过程,对用户是透明的,DOS 对它们有一套严密的管理机制。

二、盘的分区结构与簇

磁盘空间(硬盘指 DOS 分区,下同)从功能上分为三个区:逻辑扇区 0 为保留区,即引导扇区;之后是内容相同的两份文件分配表 FAT,继之是文件目录表 FDT。FAT 与 FDT 合称控制区;再后直至末尾为文件区。文件区又均匀地划分为一系列的块,一块称为一个“簇”(cluster),文件分成若干簇存储在文件区。簇的大小为 2^n ($n=0,1,\dots,4$) 个连续逻辑扇区,它是磁盘进行文件读写的最小可寻址单位。

三、FAT 表与簇号链

为了充分利用盘空间,文件各簇在盘上的分布,是按“见缝插针”办法进行的,它们在物理上常常并不连续,经过多次文件删写的盘,更是如此。但是,为了使文件能正常读写,又必须保证这些簇在逻辑上的连续性,这是由 FAT 表来实现的。FAT 表均匀地分为若干簇域,并依次编号。000H 与 001H 号簇域合称保留簇域,其中第 1 字节为盘介质标志,定义如下:

磁盘种类	介质标志
5.25 英寸 360K 软盘	FD
5.25 英寸 1.2M 软盘	F9
3.5 英寸 1.44M 软盘	F0
硬盘	F8

保留簇域的其余字节内容均为 FFH;从 002H 号簇域开始,称使用簇域,用来记录文件在文件区分布的相应簇号。因此,文件区各簇,也相应地从 002H 开始依次编号。每个文件的起始簇号,记录在 FDT 表相应目录登记项的起始簇域,之后各簇号按下述规则记录在 FAT 表中:对应一个文件的各簇域,依次记录下一簇分配的簇号,直至最后簇,如此形成一个链表,称为文件簇号链。FAT 表包含着盘上所有文件的簇号链,因此也称簇号链库。

四、簇域长度的确定

FAT 表簇域的长度,按照磁盘类型、盘容量和所用 DOS 版本,分为两种:12 位和 16 位。各类软盘均为 12 位。硬盘,DOS 2.× 时 12 位;但因使用簇域从 002H

开始编号,又把 17 个簇域号留作专用,故 12 位簇域能够管理的最大簇数为 $2^{12}-1-17=4078$,因此,DOS 3.× 支持下的硬盘,当簇数不大于 4078 时,簇域长度为 12 位;大于时为 16 位。软盘的簇域长度,在 FORMAT 格式化时确定,硬盘在 FDISK 进行分区和 FORMAT 对 DOS 分区格式化时确定。

五、簇域值的观察和计算

FAT 的常见簇域值及含义如下表:

簇域值(12 位或 16 位)	含义
(0)000H	未用簇或可用簇
(F)FF7H	坏簇
(F)FFFH	文件的结束簇
(x)xxxH	文件下一簇簇号

簇域值可用 DEBUG 观察,方法是:

```
A>DEBUG
-L100 X Y Z (调 FAT1)
-D100 (显示)
```

其中,X 为驱动器代号(0=A,1=B,2=C,...);Y 为 FAT1 的起始逻辑扇区号(通常为 1);Z 为观测扇区数。

簇域长度为 16 位时,观察比较直观,只要将显示的各簇域内容的高低字节交换,即为簇域值。簇域长度为 12 位时,观察并不直观,必须按如下规则计算:待查簇域号乘 1.5,积取整后作为 FAT 表内的字节位移,由此取出 16 位的内容送某寄存器。若簇域号为偶数,取寄存器的低 12 位为该簇域值;否则,取高 12 位为簇域值。该规则可用下述子程序实现:

入口参数 AX=簇域号
DS,BX=FAT 的段:位移

返回参数 AX=所求簇域值

```
clust proc near
    push bx
    push cx
    mov cx,ax
    shl ax,1
    add ax,cx
    shr ax,1
    add bx,ax
    mov ax,[bx]
    test cx,1
    jnz odd
    and ax,0fffh
    jmp retn
odd: mov cx,4
    shr ax,cl
retn: pop cx
    pop bx
```



```

ret
clust endp

```

为了实现 12 位簇域 FAT 表全部内容的自动直观显示,特编写了如下针对 360K 软盘的实用程序:

```

code segment
    org 100h
    assume cs:code,ds:code
start:
    proc near
        mov al,1 ;检 B 盘
        mov cx,2
        mov dx,1
        mov bx,offset buf
        int 25h ;读 fat1
        popf
        mov si,0 ;si:簇域总数
    rep1:
        mov di,0 ;di:每行簇域数
    rep2:
        mov dx,[bx]
        mov cl,4
        shl dx,cl
        call disp ;显偶数簇域
        mov dx,[bx+1]
        call disp ;显奇数簇域
        add bx,3
        add di,2
        cmp di,16 ;一行显完否
        jb rep2
        mov ah,2
        mov dl,0dh
        int 21h
        mov dl,0ah
        int 21h
        add si,10h
        cmp si,355
        jb rep1
        mov ax,4c00h
        int 21h
    start endp
;显示子程序
disp:
    proc near
        mov cx,3
    rep3:
        push cx
        mov cl,4
        rol dx,cl
        push dx
        and dl,0fh ;一位 16 进数
        cmp dl,9 ;转为 ASCII 码
        jbe aa
        add dl,7
    aa:
        add dl,30h
        mov ah,2
        int 21h ;显一位
        pop dx
        pop cx
        loop rep3 ;显三位否?
        mov dl,20h

```

```

int 21h ;插一空格
ret
endp
buf db 400h dup(0)
code ends
end start

```

该程序改动个别参数后,即可适用于 1.2M 或 1.44M 高密软盘,以及 DOS 2.×支持下的硬盘。

六、簇容量及其查询

文件区的一簇包含的扇区数,称簇容量。簇容量的大小,对磁盘的工作有较大关系。

由于文件是以簇为单位占据盘空间的,一个文件即使只有一个字节,也要占一个簇,因此,文件长度与它在盘上的分布长度是两个概念,后者通常比前者大。当短小文件(长度显著小于一簇的容量)较多时,会造成盘空间的较大浪费。为了提高盘利用率,簇容量应当小;但是,文件读写时,以簇为单位与内存进行信息交换,簇容量小,访问磁盘的次数必然增多,又因文件在盘上被分割得较碎,使磁头移动频繁,这些都会使操作效率明显降低。因此,确定簇容量时,考虑上述两方面的因素,选得适度小。

有关簇和扇区等盘参数,可由系统调用 INT 21H 的 1CH 子功能进行查询。调用的格式为:

入口参数 AH=1CH

DL=驱动器代号(0=默认驱动器,1=A,2=B,3=C,……)

返回参数 AL=扇区数/簇,即簇容量

CX=字节数/扇区

DX=盘文件区总簇数

DS:BX=FAT 表首字节的段:位移(该字节的内容即盘介质标志)。

由此查得的常见磁盘簇容量如下:5.25 英寸 360K 软盘,2 个扇区;5.25 英寸 1.2M 和 3.5 英寸 1.44M 软盘,1 个扇区;10M 硬盘,8 个扇区;20M 硬盘,使用 DOS 2.×时 16 个扇区,DOS 3.×时 4 个扇区;大于 20M 的硬盘(DOS 分区最大 32M,DOS 3.31 时除外),在 DOS 3.×支持下,簇容量均为 4 个扇区。

七、DOS 的簇管理过程

以建立文件和扩展文件为例,说明相应的簇管理过程。

每当 DOS 在盘上建立新文件时,总是从头开始搜索 FAT 表,跳过所有已分配的簇域,找到第一个可用簇域,把起始簇域号写到 FDT 表相应目录登记项中,并把文件从头分割出一块,写入文件区对应簇中;再继续查找第二个可用簇域,把簇域号写到第一个可用簇域,并向文件区对应簇写入文件的第二块;……依此类推,直至写完文件的最后一块,把结束簇标志(F)FFFFH 写到 FAT 的对应簇域中,该文件的簇号链就形成了。

当对一个文件进行扩展时,DOS 也是从头搜索 FAT 表,找到第一个可用簇域,写入(F)FFFFH,成为新的结束簇域,并将原结束簇域的值修改为指向新结束簇域,……如此下去,直至满足文件的扩展要求;文件

的扩展部分,则依次写入文件区对应簇。

八、结束语

簇管理策略是 PC DOS 的重要功能之一。用户深

入了解它的特点,对于提高编程能力,优化磁盘存储方案,改善文件读写效率,分析磁盘数据,解决病毒等问题,都是很有必要的。

程序运行过程中汉字、西文输入方式的自动转换

锦州铁路中心医院计算机室 (121001)傅 雷

在 2.13H 系统中,有一组模拟功能键的命令,格式为:CHR[\$](14)+“A 扩展 ASCII 码”。此命令将功能键的扩展 ASCII 码送入键盘管理模块,去执行系统定义的某种功能。例如,在 dBASE 环境下,由于需要输入汉字信息,可在程序中插入下面命令,将输入方式自动设置为拼音方式。

? CHR(14)+“A106” (相当于按 CTRL+F3 键)

如需转换其他输入方式,只要改变相应的功能键扩展 ASCII 码即可。常用的功能键扩展 ASCII 码见表 1。

(表 1)

功能键	内 容	扩展 ASCII 码
Alt+F1	区位码输入	104
Alt+F2	首尾码输入	105
Alt+F3	拼音码输入	106
Alt+F4	快速输入	107
Alt+F6	ASCII 码输入	109
Ctrl+F7	进/退纯西文	100
Ctrl+F8	建/取光标	101
Ctrl+F9	进/退纯中文	102

虽然此功能使用很方便,但是有一点很不理想,当在大写方式下的 ASCII 码方式转换到拼音方式后,仍为大写输入方式,这样必须转换为小写方式后,才能输入汉字。而且,在其他的汉字操作系统中,上述命令无效。下面给出在各种汉字系统中不同的输入方式转换方法,供大家参考。

大家知道,在 ROM BIOS 数据区中,从绝对地址 40H 开始的 256 个字节存放的是 ROM BIOS 的工作参数,其中,从 0017H(1047)到 001EH(1054)以及其后的 16 个字,是存放有关键盘的工作参数,详见表 2。

(表 2)

地址(10 进制)	内 容
1047(1 个字节)	换挡键和双态键状态
1050(2 个字节)	键盘缓冲区首指针
1052(2 个字节)	键盘缓冲区尾指针
1054(32 个字节)	环型键盘缓冲区

内存 1047 单元字节的每一位,表示了换挡键和双态键的状态。缓冲区首指针指的是第一个键入字符的位置,尾指针指的是键入字符的下一个位置。指针值只

在 30 到 60 之间变化,当两个指针相等时,缓冲区是空的。缓冲区可以存放 15 个键入字符,其中既有一个字节的 ASCII 码也有两个字节的扩展码。因此缓冲区必须在内存为每个字符提供两个字节。对于一个字节的 ASCII 码来讲,第一个字节是 ASCII 码,第二个字节是键扫描码。对于扩充码来讲,第一个字节是 ASCII0,第二个字节是扩展码。

根据上面所述,不难写出转换到不同输入方式的小程序,仍以 dBASE 为例:

* PY. PRG 转换到拼音方式

SET ESCA OFF

POKE 1047,0 ;设置小写字母

POKE 1050,30 ;设置缓冲区首指针

POKE 1052,32 ;设置缓冲区尾指针

POKE 1054,0 ;设置 ASCII 码 0

POKE 1055,106 ;设置拼音方式的扩展码

* XW. PRG 转换到 ASCII 码方式

SET ESCA OFF

POKE 1047,64 ;设置大写字母

POKE 1050,30

POKE 1052,32

POKE 1054,0

POKE 1055,109 ;设置 ASCII 码方式的扩展码

应用举例:

:

CLEAR

@ 5,10 SAY '编号:' GET NO ;ASCII 码

DO XW

READ

@ 6,10 SAY '姓名:' GET NA ;汉字

DO PY

@ 7,10 SAY '学校:' GET SH ;汉字

READ

8,10 SAY '班级:' GET CL ;ASCII 码

DO XW

READ

:

采用上述方法,可以方便地在各种高级语言中实现各种不同输入方式的自动转换。

C 语言在配平化学方程式中的应用

北京理工大学附中 王 晰



C 语言是一种目前很流行的语言,它既有很高的效率,又有清晰的程序结构和丰富的数据类型。它在实现结构化编程的同时又对编程者很少有严格的限制,使他们很容易表达自己的思想。这些优点使 C 语言很适合编制以逻辑判断为主的系统软件和人工智能程序。

配平化学方程式是中学化学中的一个重要问题。关于配平化学方程式有很多方法,但是都不适于用计算机来完成。然而,我们知道,每一个已经配平的化学方程式都遵守物质不灭定律,即每一个参加反应的原子,都必须包含在反应后生成的物质中。根据这一定律,通过比较反应前后的各类原子数目是否相等,就可以知道方程式是否已经配平,我们也就可以利用计算机速度快、适合循环重复的特点采用试凑法解决这个问题了。

本程序的优点是:程序执行中,根据屏幕提示直接输入各种未配平的化学方程式,程序执行后,也直接在屏幕上显示配平后的方程式,非常直观,而且符合人们的习惯。为了实现这一点,程序的第一部分是语法分析程序。它先从方程式中将各种不同物质的分子分离出来,再把每种物质分子分解为单个元素,然后存入结构 `lwz, rwz` 中。程序的第二部分是试凑部分。试凑程序对每一种可能的情况进行试验,如果反应前后各类原子数相等,则停止试凑,输出结果。试凑按系数由小到大进行,以保证输出最简结果。

程序使用方法十分简单。程序经 Turbo C 编译后即可运行。当屏幕上提示输入化学方程式的时候,就可以输入未配平的化学方程式。程序执行后便显示平衡的化学方程式。

例:

1. 输入 $\text{Cu} + \text{HNO}_3 = \text{CuN}_2\text{O}_6 + \text{NO} + \text{H}_2\text{O}$ ✓
输出 $3\text{Cu} + 8\text{HNO}_3 = 3\text{CuN}_2\text{O}_6 + 2\text{NO} + 4\text{H}_2\text{O}$
2. 输入 $\text{NH}_3 + \text{O}_2 = \text{NO} + \text{H}_2\text{O}$ ✓
输出 $4\text{NH}_3 + 5\text{O}_2 = 4\text{NO} + 6\text{H}_2\text{O}$
3. 输入 $\text{FeS}_2 + \text{O}_2 = \text{Fe}_2\text{O}_3 + \text{SO}_2$ ✓
输出 $4\text{FeS}_2 + 11\text{O}_2 = 2\text{Fe}_2\text{O}_3 + 8\text{SO}_2$

注意:

① 大小写非常重要,当元素用两个字母表示时,后一个字母一定要小写,如 Cu , Fe 等,否则程序将不能区分是一个元素还是两个元素。

② 本程序不处理括号,如 $\text{Cu}(\text{NO}_3)_2$ 应写成 CuN_2O_6 。

```
#include "stdio. h"
#include "string. h"
#include "stdlib. h"
#include "ctype. h"
```

```
#include "dos. h"
#define NL 3
#define YL 6
#define WL 20
#define FL 255
#define ZL 25
#define TRUE 1
#define FALSE 0
#define STEP 0
struct yuansu {
    char name[NL];
    int xishu;
    int * zongji;
};
struct wuzhi {
    struct yuansu ys[YL];
    int ysshu;
} lwz[WL], rwz[WL];
char lzjname[ZL][NL], rzjname[ZL][NL];
int lzzongji [ZL], rzzongji[ZL];
int lwzshu, rwzshu;
int lzjs, rzjs;
int qs[WL * 2];

main()
{char fcs[FL];
setcbrk(1);
for(;;)
{printf("\nInput the chemical equation:");
    gets(fcs);
    yufa(fcs);
    chushi();
    shicou();
    xwout();
}
}

yufa(char fcs[FL])
{char * p;
    p=strpbrk(fcs,"=");
    if(p==NULL) wrong("SYNTAX ERROR");
    *p=NULL;
    lwzshu=fenxi(fcs,&lwz);
    rwzshu=fenxi(p+1,&rwz);
}

fenxi(char * fcs,struct wuzhi * wz)
{int i, ysshu=0, wzshu=0;
    char * j=fcs;
    fcs++;
    for(i=0;i<FL;i++,fcs++)
```



```

{if (isupper(* fcs) || * fcs == '+' || * fcs == NULL)
{char * n = fcs - 1;
if (! isdigit(* n))
{wz -> ys[ysshu]. xishu = 1;
n = fcs;
}
else{
for (; isdigit(* (n - 1)); n--)
wz -> ys[ysshu]. xishu = atoi(n);
}
strncpy (wz -> ys[ysshu]. name, j, n - j > NL -
1 ? NL - 1 : n - j);
if (fcs - j < NL)
{ * (wz -> ys[ysshu]. name + (fcs - j)) = NULL;
}
ysshu++;
if (ysshu > YL) wrong("YUAN SU TOO MANY");
j = fcs;
if (* fcs == '+' || * fcs == NULL)
{wz -> ysshu = ysshu;
ysshu = 0;
wz++;
wzshu++;
if (* fcs == NULL) break; /* END */
fcs++;
j = fcs;
if (wzshu >= WL) wrong("WU ZHI TO MANY");
}
}
}
return (wzshu);
}
chushi()
{lzjs = csh(lwz, lwzshu, lzjname, lzhongji, 0);
memcpy (rzjname, lzjname, sizeof(lzjname));
rzjs = csh(rwz, rwzshu, rzjname, rzongji, lzjs);
if (lzjs != rzjs) wrong("YUAN SU BU PI PEI");
}
csh(struct wuzhi * wz, int wzs, char zjn[][NL], int * zj, int
zjs)
{register i, j;
for (; wzs > 0; wzs--, wz++)
{for (i = 0; i < wz -> ysshu; i++)
{for (j = 0; j <= zjs && strcmp (zjn[j], wz -> ys[i].
name) != 0; j++)
if (j > zjs)
{zjs++;
j = zjs;
strcpy (zjn[j], wz -> ys[i]. name);
}
wz -> ys[i]. zongji = zj + j;
}
}
return zjs;
}
}

```

```

shicou()
{int jishu, i, zwzs = lwzshu + rwzshu;
register * pqs, * pqh = &qs[zwzs - 1];
for (i = 0; i < zwzs; i++, qs[i] = 1)
for (jishu = 1; jishu < 20; jishu += STEP) /* jishu
为试凑次数 */
{testsc(jishu);
i = TRUE;
for (; i;)
{pqs = qs;
if (panduan()) return TRUE;
while (++ * pqs > jishu + STEP)
{ * pqs = 1;
pqs++;
if (pqs > pqh)
{i = FALSE;
break;
}
}
}
}
panduan()
{memset(lzhongji, 0, ZL);
memset(rzhongji, 0, ZL);
tongji(lwz, qs, lwzshu);
tongji(rwz, qs + lwzshu, rwzshu);
return ! memcmp(lzhongji, rzhongji, sizeof(lzhongji));
}
tongji(struct wuzhi * wz, int * qs, register wzs)
{register i;
for (; wzs > 0; wzs--, wz++, qs++)
{for (i = 0; i < wz -> ysshu; i++)
* (wz -> ys[i]. zongji) += wz -> ys[i]. xishu * (* qs);
}
}
xwout()
{puts("\n");
xout(lwz, qs, lwzshu);
printf("=");
xout(rwz, qs + lwzshu, rwzshu);
puts("\n");
}
xout(struct wuzhi * wz, int * qs, int wzs)
{int i;
for (; wzs > 0; wzs--, wz++, qs++)
{if (* qs > 1)
printf("%d", * qs);
for (i = 0; i < wz -> ysshu; i++)
{printf("%s", wz -> ys[i]. name);
if (wz -> ys[i]. xishu > 1)
printf("%d", wz -> ys[i]. xishu);
}
if (wzs != 1) printf("+");
}
}

```

```

}
wrong(char * message)
{puts(message);
exit(1);
}
testsc(int i)
{printf("%4d",i);
}

```

DISK MANAGER

使用技巧

湖北通山县卫生防疫站(437600) 梁高军

DISK MANAGER(DM)是 PC 机上流行的硬盘管理工具软件,由于其许多操作将破坏硬盘上的数据,致使众多的用户不敢使用。其实,合理使用 DM 程序将给你的硬盘管理工作带来诸多便利,下面是笔者的一些体会:

1. 保密硬盘数据

DM 提供了比 DOS 的 FDISK 更为方便的硬盘分区操作,并自动给每一硬盘分区分配一逻辑驱动器号。然而,这些逻辑驱动器的使用是以启动盘的根目录中有 DMDRVR·BIN 文件及其 CONFIG·SYS 文件中有一句:

DEVICE=DMDRVR·BIN

为前提的,只要上列条件不能满足,DOS 即不予认可这些逻辑驱动器,也即不能进入这些硬盘分区进行操作。我们只要用 DM 进行硬盘分区,将要保密的数据拷入某一逻辑驱动器,并使日常启动盘(硬盘)根目录中无上述两个条件,这样,硬盘启动后就不能进入要保密的数据区进行操作,达到了保密数据的目的。当要对保密的数据区进行操作时,可另组织一软盘作为启动盘,并使其上的 CONFIG·SYS 文件中包含有上述语句和其根目录中存在 DMDRVR·BIN 文件。用此软盘启动后即可对所保密的数据区进行正常操作。

DM 划分硬盘的操作是键入:

DM/M

此时屏幕出现 DM 主菜单,键入 P 并回车,若硬盘原有分区则显示原有分区信息,并等待用户决定是否改变原有分区,键入 Y 则进入分区操作菜单。在此菜单下键入 N 即可建立新的硬盘分区(若硬盘已有数据须作好备份,分区操作将使原有数据丢失),按照需要进行分区后,DM 会请用户给每个逻辑驱动器输入一个卷标,然后即自动对每个分区进行格式化工作。全部工作完成后,各逻辑驱动器即可投入使用。使用 DM 进行硬盘分区还有一个好处,即 DM 的分区只能用 DM 进行格式化,DOS 的 FORMAT 不能对 DM 的分区进行操作,也就避免了日常工作中误格式化硬盘的损

失。

2. 写保护硬盘或硬盘分区

硬盘虽然有容量大,速度快的优点,但由于没有像软盘一样方便的写保护特性,这样就有可能因一些误操作而破坏硬盘的数据,造成不应有的损失。利用 DM 则可简单方便地将整个硬盘或其中某个分区置为只读(写保护状态),从而保护数据和文件。其操作为:

首先利用 DM 进行硬盘分区,退出 DM 后将欲写保护的数据文件写入某一(或多个)逻辑驱动器(分区),操作完成后重新进入 DM 分区操作菜单,键入 C,屏幕提示输入分区号,键入欲写保护的分区号(例如:2)并回车,键入 R 选择只读,继续回车后屏幕出现一条警告,并等待用户决定是否将分区数据写入硬盘,键入 Y 则 D 驱动器被写保护(对双软驱、一硬盘的用户,硬盘 2 号分区对应逻辑驱动器号为 D)。

3. 恢复被破坏的硬盘分区表

如果因日常工作中的误操作或计算机病毒破坏了硬盘的分区信息,一般情况下,硬盘的数据就已经全部丢失,但使用 DM 分区的硬盘则可利用 DM 快速恢复硬盘分区信息,不致丢失硬盘文件或数据:

键入 DM/M,选择 P 进入分区操作菜单,这时可看到原分区信息已丢失,选择 N 建立与原分区相同参数的分区(注意:输入的分区数据必须与原分区相同,否则 DM 将对分区进行格式化操作,数据也就丢失了),将分区数据写盘后即可恢复原硬盘操作。

除上之外,在 DM 主菜单下,键入 I 即可对硬盘进行一系列的初级格式化工作(若非必要,一般不要使用此功能,否则将丢失全部硬盘数据而无法恢复),键入 C 则可对系统关于硬盘的设置进行操作。对多硬盘用户键入 S 可选择下一个硬盘。

总之,熟练运用 DM 对 PC 用户的硬盘文件管理及硬盘软故障的排除是非常有益的。

欢迎订阅《电子报》

《电子报》是国内创办最早的一份技术性电子类报纸。15 年来,它一直以其“实用性、新颖性、启发性、资料性”而受到广大读者的欢迎与喜爱。为适应电子技术的发展和满足读者多层次的需要,《电子报》从 1993 年起将由原四开四版扩为四开八版。

“新版《电子报》”不仅容量较过去增多一倍,在选题和质量方面更有所扩展和提高。第一版左边为“新闻言论版”,第二版“技术和商品信息版”,第三版“家电维修普及版”,第四版“家电维修提高版”,第五版“青少年电子版”,第六版“电子应用版”,第七版“新器件、新技术版”,第八版“AV 发烧版”。

(全国邮局均可订阅。月价 0.96 元。)

邮局订阅代号:61-75

如何利用 dBASE Ⅲ 的 F1 功能键

江西教育学院微机室 (330029) 涂振宇

在 dBASE Ⅲ 中 F2—F10 这些键可被用户动态定义,给我们带来很大的方便,唯独 F1 键被定义为 HELP,不能用于编程。本文介绍的方法可以使 F1 重新定义,可用于编程或在 MODI COMM 状态下输入定义字符串。

本方法的思想是修改 INT16H,当按下 F1 键时使其指向新的中断,而不被 dBASE 读取,这样在新的中断中,我们就可自由使用 F1 键了。

将下面程序汇编、连结、转换成 .COM 文件,在调 dBASE 前运行此程序,该程序即常驻内存。本例是把 F1 定义为 'MODI COMM',用户可自行改动,例如在程序中可借助本程序和 SET 命令将 F1—F10 定义为特殊字符,如半个汉字,这样配合 READ/GET 和 DO CASE 语句,我们可使用 F1 等功能键决定程序流向了。

```
codesg segment para 'code'
    org 100h
    assume cs:codesg,ds:codesg,es:codesg,ss:codesg
main proc near
begin: jmp start
endflag dw 0ffffh
old-int16 dd 0
f1 db 'modi comm',0
newint16: or ah,ah
        jz 11
        jmp dword ptr cs:old-int16
11: sti
        push bx
        push cx
        push dx
        push si
        push ds
        push cs
        pop ds
        cmp endflag,0ffffh
        jnz 13
        pushf
        call dword ptr cs:old-int16
        cmp al,0
        jnz exit
        cmp ah,3bh
        jnz exit
        mov si,offset f1
        mov endflag,si
13: mov si,endflag
```

• 10 •

```
        mov al,[si]
        mov ah,0
15: cmp al,0
        jz 16
        inc si
        mov endflag,si
        jmp exit
16: mov endflag,0ffffh
        exit: pop ds
        pop si
        pop dx
        pop cx
        pop bx
        iret
start: mov ax,3516h
        int 21h
        mov word ptr old-int16,bx
        mov word ptr old-int16+2,es
        mov ax,2516h
        push cs
        pop ds
        lea dx,newint16
        int 21h
        mov ax,3100h
        mov dx,50h
        int 21h
main endp
codesg ends
end begin
```

一点补充:

今年第 8 期所刊“CEC—I 自造字点阵生成程序”生成的是与机内硬汉字库相同的正排点阵(即每行点阵按高位在左、低位在右的顺序排放),而有的调取点阵的程序要求提供的是倒排点阵(如本刊今年第 4 期《也谈全功能造字》一文的程序),否则会将左右两半字印反,若要生成倒排点阵,可将本造字程序 160 行中的“FOR K=7 TO 0 STEP -1”改为“FOR =0 TO 7”、将 280 行中的“U=8+8 * T-H”改为“U=H-1-8 * T”即如愿。

另外,所造的字亦可表以区位码,字码的具体对应完全由所使用的调字程序决定,如调字程序规定所调取自造字均在第 10 区,则本造字程序生成的第 1、2、……号字的区位码依次为 1001、1002、……。

该文作者:汤永进

CCBIOS 2.13 稿纸方式打印

成都热电厂 (610051) 宋捷

我们经常希望文本文件能按稿纸方式打印,现介绍一个在 CCBIOS 2.13 系统下,将文本文件按稿纸方式打印的简单方法:

1. 在 FoxBASE 下建立数据库 GZ.DBF,库结构见程序一,该库仅有一个长为 40 的字符型字段 G;

2. 用 WS 建一个 FoxBASE 程序 GZ.PRG,程序 GZ.PRG 结构整齐,简单实用,打印时自动分页,并标页码,由于程序所设的中间库 GZ.DBF 只设了一个宽度为 40 的字段,且程序划框线时对于重复的框线使用了 REPL()函数,因此相应的简化了程序,并提高了程序在打印过程中的运行速度。

3. 使用时应先利用 WS.XE 等字处理软件将文本的内容输入到一个文本文件中,在输入时,为防止在以后程序处理中字符的错位或者某些字符被截断,应注意每行只能输入 20 个汉字,并且在文本输入汉字或全角字符时,汉字或全角字符的前半个字必须在奇数列,后半个字必须在偶数列,标点符号最好用全角方式输入,两个阿拉伯数字或英文字母应占一个汉字宽度;

4. 在运行程序 GZ.PRG 前,应打开打印机,装好 80 行打印纸,程序运行时只要出现“请输入稿纸方式打印的文件名:”的提示,便可输入后缀为 TXT 的文本文件名,此时打印机就会在打印纸的正中按稿纸方式打印出相应的文本文件了。

程序在 AST486、HP386、GW386 机和 AR3240 打印机上实现,操作系统为 DOS 3.3,汉字系统为 CCBIOS 2.13H。

程序一:

GZ.DBF 数据库结构:

Field	Field Name	Type	Width	Dec
1	G	Character	40	
* * Total * *			41	

程序二:

```
C> TYPE GZ.PRG
* * * 稿纸方式打印 * * *
set talk off
set safe off
clear
accept '请输入稿纸方式打印的文件名:' to filename
use gz
zap
appe from &filename sdf
go top
n=1
do while .not. eof()
```

```
set devi to prin
line=1
@ prow(),1 say CHR(96)+CHR(64)+CHR(65)+CHR
(38)+"15"+CHR(91)+"8"+CHR(96)
do while line<=20
if .not. eof()
@ prow()+1,1 say "┌" + repl(" ",19) + "┐"
@ prow()+1,1 say "|"
s=1
do while s<=20
@ prow(),pcol() say subs(g,(2*s-1),2)+"|"
s=s+1
enddo
@ prow()+1,1 say "└" + repl(" ",19) + "┘"
skip
else
@ prow()+1,1 say "┌" + repl(" ",19) + "┐"
@ prow()+1,1 say "|" + repl(" ",19) + "|"
@ prow()+1,1 say "└" + repl(" ",19) + "┘"
endif
line=line+1
enddo
@ prow()+3,37 SAY '第'+str(n,2)+'页'
n=n+1
@ prow()+29,1 say ' '
set devi to scre
enddo
return
```

微机屏幕的打印和放大

武汉铁路分局电子计算技术中心 谭人杰

本文介绍将屏幕缓冲区的显示数据转换成打印数据以及放大打印方法,并提供已经实现的屏幕打印源程序。

一、直接存取屏幕显示缓冲区的原理和方法

屏幕图形由显示缓冲区中的 640×200 个二进制位表示,在 CGA 中分辨率彩色图形方式下,每个像素点用两个二进制位表示,这两个二进制位取值 0 时表示背景色(无点),取值为 1、2、3 时表示不同彩色(有点),每个字节可表示四个像素点,屏幕上每行扫描线用 80 字节表示 320 个像素点。

屏幕纵向共有 200 行扫描线,100 行偶数扫描线 (0,2,4,...,198) 占用以 B8000H 为起始地址的 8000 字节空间,100 行奇数扫描线 (1,3,5,...,199) 占用以 BA000H 为起始地址的 8000 字节空间。

要直接存取某一像素点,可按以下算法编程:

1. 计算该点所在扫描线的 80 字节的起址;
2. 计算在这 80 字节中该点所在的字节;
3. 计算在这一字节中该点所在的两位二进制数;
4. 对这两位二进制数设置彩色代码或赋像素值。

二、打印机以图形方式打印屏幕的原理和方法

24 针打印机以图形方式打印时,将 24 支针排成一列,每支针由一个二进制位控制,二进制位取值 0 时,针头不进针;取值 1 时,针头进针打点。24 支针由三个字节控制,对于 M-1724 打印机,每行可打印 2176 列的 24 针点,即 2176×3 个字节表示的一行图形数据。

为了与图形打印方式对应,将屏幕以 24 行扫描线为单位转换成一行图形打印数据,重复八次将 200 线的显示数据全部转换成打印数据,整个屏幕图形就可以在打印机上打印出来。每次取 24 行扫描线时,需按奇偶行分别在奇数行扫描线缓冲区和偶数行扫描缓冲区取数。由于是两个二进制位表示一个像素点的彩色图形方式,所以每一打印列实际对应屏幕上两列二进制位,因此每次要在显示缓冲区取两列数据转换成一行打印数据后再发送到打印机;也就是说,要将显示缓冲区中表示一个彩色像素点的两位二进制数转换成打印缓冲区中表示黑白打印点的一位二进制数。转换方法是根据显示缓冲区中代表一个像素点的两位二进制数的取值来决定,如果取值 0,则打印针头对应的数据位赋值 0;如果取值 1、2、3,则打印针头对应的数据位赋值 1。

三、屏幕放大打印

知道了打印屏幕的原理,就可以按比例将屏幕图形放大,如果要求横向放大两倍,只需将屏幕上转换后的每列 (24 点) 数据连续送两次打印机,形成两列打印数据;如果要求纵向放大两倍,则需将屏幕上每个点转换成与打印针对应的纵向两个点;也就是说,将屏幕上纵向 12 个像素点转换成一列打印数据的 24 个点。以此类推,可以编写扩大四倍和扩大八倍的屏幕放大打印程序。

四、程序实例

由于篇幅所限,这里只提供按原幅打印屏幕的程序,其思路是对屏幕上的 200 行扫描线,每次取 24 线转换成 24 针打印机对应的一行数据。并以图形方式打印。主程序首先将打印机置成图形打印方式,并将行宽置成 19/120 英寸,主程序调用过程 TRAN_BYTEI,将显示缓冲区中纵向 8 行扫描线的一列像素转换成一字节打印数据,对屏幕上每 24 行扫描线的一列像素,主程序调用三次子程序,并将返回的三字节依次送打印机,形成控制 24 根针的一列打印数据。TRAN_BYTEI 调用过程 TRAN_BIT,将屏幕上指定位置的纵

向 8 个像素依次转换成一字节中对应位的图形打印数据。

打印机为 M-1724,编程语言是 TURBO PASCAL 3.01A。

以上介绍的原理和方法,适用于各种类型的微机 and 打印机,对不同的显示卡,只要知道显示缓冲区的地址以及扫描线的排列方式,就可以直接访问屏幕上的任意点。对不同的打印机,只需修改“置图形打印方式”命令和“置打印行宽”命令。

例如 EPSON 1000K 系列打印机的这两条命令为

```
ESC * 38 n1 n2
```

```
ESC 3 24
```

```
{SCRNDUMP.PAS---dump screen to M1724,line space 19/  
120 inch}
```

```
PROGRAM SCRNDUMP(INPUT,OUTPUT);
```

```
VAR
```

```
DATA:BYTE;
```

```
LINE,ROW,COL:INTEGER;
```

```
PROCEDURE TRAN_BYTEI(I,COL:INTEGER;VAR DATA:  
BYTE);
```

```
VAR
```

```
START_ROW:INTEGER;
```

```
PROCEDURE TRAN_BIT(ROW,COL:INTEGER;VAR DA-  
TA:BYTE);
```

```
CONST
```

```
VRAM_BASE= $B800; {Segment address of video buffer}
```

```
TYPE
```

```
BIT=INTEGER;
```

```
BIT2=INTEGER;
```

```
VAR
```

```
CF,BIT;
```

```
CF2,BIT2;
```

```
BASE,ROW_START,OFF_ROW,OFF_COL:INTEGER;
```

```
OBJ_BYTE,OBJ_BIT,MARK:BYTE;
```

```
BEGIN
```

```
{calculate start address of the 80 byte line}
```

```
CF:=ROW MOD 2;
```

```
OFF_ROW:=ROW SHR 1;
```

```
IF CF=0 THEN BASE:= $ 0000
```

```
ELSE BASE= $ 2000;
```

```
ROW_START:=BASE+OFF_ROW*80
```

```
CF2:=COL MOD 4; {ith two bit binary}
```

```
OFF_COL:=COL SHR 2; {column divided by 4 to get ith  
byte in the 80 bytes}
```

```
OBJ_BYTE:=MEM[VRAM_BASE,ROW_START+
```

```
OFF_COL]; {take the ith byte}
```

```
MARK:= $ C0 SHR (2*CF2);
```

```
OBJ_BIT:=MARK AND OBJ_BYTE; {take the ith two  
bit binary}
```

```
IF OBJ_BIT<>0 THEN BEGIN {set 1 to the bit of print  
data}
```

```
MARK:= $ 80 SHR (ROW_START-ROW)
```

```
DATA:=DATA OR MARK;
```

```
END;
```

```

END;
BEGIN
  DATA:= $00;
  START_ROW:=8*1;
  FOR ROW:=START_ROW TO START_ROW+7 DO
    TRAN_BIT(ROW,COL,DATA);
  END;
BEGIN
  WRITE(LST,CHR(24));{Initialize printer}
  WRITE(LST,CHR(27),'J',CHR(19));{Set line space}
  FOR LINE:=0 TO 7 DO BEGIN
    WRITE(LST,CHR(27),'4',CHR(1),CHR(64));
    {Set bit graphics,1*256+64=320 columns}
    FOR COL:=0 TO 319 DO BEGIN {column count}
      TRAN_BYTEi(LINE*3+0,COL,DATA);
      WRITE(LST,CHR(DATA));
      TRAN_BYTEi(LINE*3+1,COL,DATA);
      WRITE(LST,CHR(DATA));
      TRAN_BYTEi(LINE*3+2,COL,DATA);
      WRITE(LST,CHR(DATA));
    END;
    WRITE(LST,CHR(13),CHR(10));
  END;
END.
=====

```

1992 年全国青少年信息学(计算机)竞赛试题

第一轮 试题 1.

把一段文章按要求排版。

文章的输入方式为:由键盘输入一个以回车符结束的文章(最大长度 2000 个字符)。

排版时以单词为基本单位。单词由不含空格的任意字符组成,是长度小于 20 个字符的串。空格符是分隔单词的唯一字符,在输入时连续的空格符在处理时应首先简化为单个空格符。

在排版前应先输入排版后每行的字符数 N,排版后将整理好的文章按行输出。输出时应保证不将一个完整的单词截断,并且要求输出的总行数最小。

将每个不足 N 个字符的行用空格符补足,填充空格符的方式有以下三种:

1. 将填充的空格符置于每行的末尾,并要求每行的起始为单词。
2. 将填充的空格符置于每行的起始,并要求每行的末尾为单词。
3. 将填充的空格符尽可能平均分配在每行中,并保证每行的起始和末尾均为单词。试编程对输入的一段文章分别完成上述三个要求。

第一轮试题 2

由英文字母和符号 \sim 、 $*$ 、 $+$ 、 $()$ 组成逻辑表达式,英文字母表示变量,变量有两种可能的取值,False 或 True; \sim 、 $*$ 、 $+$ 分别代表逻辑运算的非、与、或。运算的优先依次为 $()$ 、 \sim 、 $*$ 、 $+$ 。括号 $()$ 可改变表达式的运算次序,且可以嵌套。

逻辑“非”运算的公式如下表:

A	$\sim A$
True	False
False	True

逻辑“与”和逻辑“或”的运算公式如下表

A	B	$A * B$	$A + B$
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

两个逻辑表达式等价,并且仅当两个公式中相同名字的变量取任何一组值时两个公式的值都相同。如:

$A * (B + C)$ 与 $A * B + A * C$ 等价
 $A * (\sim A + B)$ 与 $A * B$ 等价
 $(\sim A + A) * B + C$ 与 $B + C$ 等价
 $A * B + A * \sim B$ 与 A 等价

而:

$A + B$ 与 $A * B$ 不等价
 $A * B + \sim C$ 与 $A * E + \sim F$ 不等价

现要求你编程解决下列问题:

任务 1:

用键盘输入一个逻辑表达式,判断这个表达式的合法性;

任务 2:

将键盘输入的表达式化简,化简的表达式形式为
 $a1 * a2 * \dots * aN + b1 * b2 * \dots * bM + \dots + x1 * x2 * \dots * xL$

其中 $ai, bj, \dots, xk (i=1, 2, \dots, n; j=1, 2, \dots, m; k=1, 2, \dots, l)$ 表示一个变量或一个变量的逻辑非。

任务 3:

将任务 2 中的化简的表达式优化为最简形式。所谓最简有如下两个条件:

- (1) 表达式中的“+”号最少;
- (2) 满足(1)的条件下“*”号最少。

第二轮试题 1

无根树与通常所说的树(有根树)很相似,它包含

有节点和枝,但不含有根。无根树节点间只有相邻关系,而不存在父子节点的关系。如图1所示,是一棵有7个节点的无根树;以图1的A为根节点得到图2所示的有根树,以图1的B为根节点得到图3所示的有根树,但从无根树的角度看,图1、图2、图3是结构相同的无根树,同时无根树的结构与节点的名称无关。

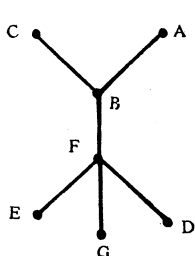


图1

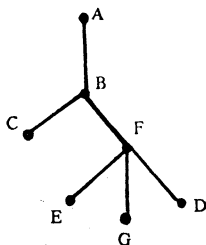


图2

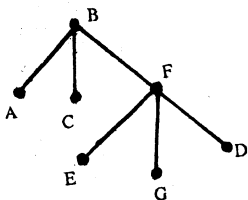


图3

有根树可以以字符串的形式表示,其递归表示方法为:

根节点(子树1 子树2 子树3.....)

如图2、图3的有根树可分别表示为A(B(CF(EGD)))和B(ACF(EGD))。需要注意的是,由于子树的表示顺序可以不同,所以一棵有根树可以有多种表示方法,如图3又可表示为B(F(EGD)CA)或B(ACF(DEG))等。

表示无根树时,可以以它的任一节点为根节点,将其看作有根树,从而可以利用有根树的字符串表示形式来表示无根树。

任务1:

由键盘读入一个字符串表示的无根树,无根树的各节点的名称用互不相同的大写英文字母表示。由用户输入一个节点的名称,程序应能够输出一种以该节点为根节点的字符串形式。

程序输出无根树的字符串形式时,各个节点的名称无关紧要,所有节点都以P表示,以后的各种输出也采用这种方式。

例如,用户输入无根树的字符串形式:(A(B(CD(EF))))

指定的根节点为:D

程序应能输出:

P(P(P(P)P) P(P(P(P)P) P(PPP(P)P))中的任意一种即可。

任务2:

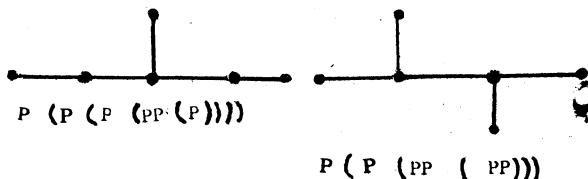
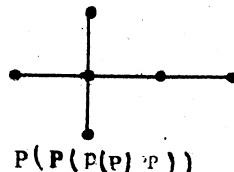
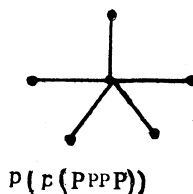
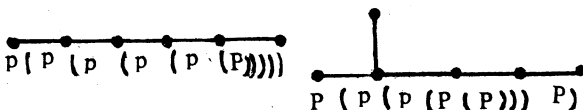
输入两个串表示的无根树,判断其结构是否一致。注意与节点名称无关,只考虑结构。

任务3:

输入无根树的总枝数N($1 \leq N \leq 11$),输出所有枝数为N的互不相同的无根树,并记录总数。以字符串形式输出:

例如,N=5时,共有6种不同结构的无根树,如下所示:

注意:各种树结构的字符串表达形式不唯一。



第二轮试题2

某机要部门安装了电子锁。M个工作人员每人发一张磁卡,卡上有开锁的密码特征。为了确保安全,规定至少要有N个人同时使用各自的磁卡才能将锁打开。现在需要你计算一下,电子锁上至少要有多少种特征,每个人的磁卡上至少要有几个特征。如果特征的编号以小写英文字母表示,将每个人的磁卡的特征编号打印出来。要求输出的电子锁的总特征数最少。

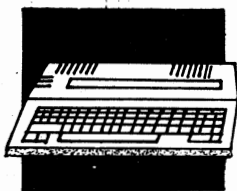
为了使问题简单,M与N的上下限为

$$3 \leq M \leq 7, 1 \leq N \leq 4$$

M与N由键盘输入,工作人员的编号用1#,2#,...等。

例如 M=3,N=2,则电子锁上要有三种特征。每个人的磁卡上要有二种特征。

(此文由清华大学计算机系吴文虎教授提供)



学习机之友

Applesoft BASIC 子程序的递归调用

韶关教育学院(512026) 陈继良 广东北江中学 丘文

一般认为,在 Applesoft BASIC 中,子程序是不能递归调用(即自己调用自己)的,如

[程序一]

```
20 GOSUB 20
```

```
]RUN
```

? Out of memory in 20

事实上并非如此,请看

[程序二]

```
10 I=0
```

```
20 I=I+1;PRINT I,
```

```
30 GOSUB 20
```

```
]RUN
```

```
1      2      3
```

```
4      5      6
```

```
7      8      9
```

```
10     11     12
```

```
13     14     15
```

```
16     17     18
```

```
19     20     21
```

```
22     23     24
```

```
25
```

? Out of memory in 20

可见,子程序是可以递归调用的,只是递归的次数一般不能超过 24,这与子程序嵌套的最大层数是一致的。通过剖析 Applesoft 可知,执行 GOSUB 语句时,计算机会把有关数据(GOSUB 后续语句的位置)进栈;遇 RETURN 语句时,栈顶数据退栈。子程序嵌套(包括递归调用)层数的限制是由于堆栈空间有限而引起的,子程序嵌套时,如果进栈的数据太多而又未能及时退栈,就会导致空间溢出而出错。对于子程序的非递归嵌套,一般极少超过 24 层,所以栈空间不足的矛盾并不突出;而在子程序的递归调用中(如程序一、二),就很容易造成栈空间溢出。因此,一般书刊都不提子程序的递归问题,甚至干脆说不能递归。

根据上面的分析,只要把次数控制在 24 以内,子程序完全是可以递归的。但在实际的递归问题中,递归的次数往往在 24 次以上。例如

[程序三]

```
10 INPUT "S,C=";S,C;X=140;Y=90
```

```
20 HGR2;HCOLOR=3;GOSUB 100
```

```
30 END
```

```
100 HPOINT X,Y TO X,Y-S TO X+S,Y-S TO X+S,  
Y TO X,Y
```

```
110 S=S-C;IF S<=0 THEN RETURN
```

```
120 GOSUB 100
```

```
130 RETURN
```

```
]RUN
```

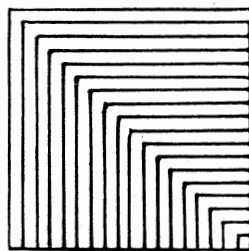
```
S,C=90,4
```

(结果见图一)

```
]RUN
```

```
S,C=90,3
```

? Out of memory error in 100



图一

S 为正方形的边长,C 为相邻正方形的边长之差, S=90,C=3 时,递归次数已超过 24 次,栈空间溢出。因此,要在非结构化的 Applesoft BASIC 中真正实现递归算法,必须先解决子程序的无限递归问题!

在子程序的递归调用中,进栈的数据中有很多是重复的。如果能设法把多余的数据及时弹出栈外,便可解决栈空间小的矛盾,从而实现子程序的无限递归。除了 RETURN 语句外,还有一个 POP 语句具有退栈功能,特别地,POP 语句退栈后并不返回,而是继续执行后续语句,只要我们设置适当的退栈条件,必要时通过执行 POP 语句退栈,便可实现子程序的无限递归。例如,在程序三中加入

```
15 CHEN=1
```

```
105 IF CHEN>1 THEN POP
```

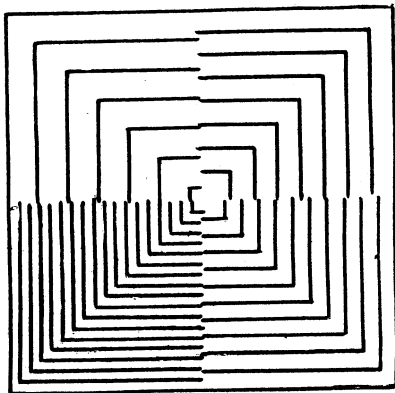
```
115 CHEN=CHEN+1
```

便可完成 S 和 C 为任意值时的递归过程,只有这样,程序三才具有一定的实用价值。其中的“CHEN>1”即为退栈条件,程序执行过程中栈的变化情况如下:执行 20 行的 GOSUB 100 时,数据一(即 20 行 GOSUB 100 后一语句的位置)进栈;因为 15 行设置 CHEN=1,所以 105 行不执行 POP,当 115 行 CHEN 加 1 后由 120 再 GOSUB 100,首先是数据二(即 120 行 GOSUB 100 下一语句的位置)进栈,然后转 100 行;由于这时 CHEN>1 即退栈条件满足,所以执行 POP 语句,数据二被弹出栈外而作废。如此反复,直到 110 行的递归边界“S<=0”被满足而执行 RETURN 退栈返回。由于栈中有效的数据始终只有数据一,所以程序这时将返回 30 行正常结束。这样,通过设置退栈条件及时把多余数据弹出栈外,解决了令人讨厌的栈空间溢出问题,原则上实现了子程序的无限递归。由以上分析可知,这时

130 行形同虚设(不会被执行);如果要保留并执行 130 行,则退栈条件可改为“CHEN>N”(N 为 2——22 之间的整数。可见,退栈条件可根据具体情况灵活设置,这是尾递归的情况,下面在程序三的基础上,给出通过子程序的非尾递归调用画出图二的程序

[程序四]

```
10 INPUT "S,C,D=";S,C,D:X=140:Y=90
15 CHEN=1:S0=S:X0=1:Y0=-1
20 HGR2,HCOLOR=3;GOSUB 100
30 END
100 HPLLOT X,Y TO X,Y+S*Y0 TO X+S*X0,Y+S
    *Y0 TO X+S*X0,Y TO X,Y
105 IF CHEN>2 THEN POP
110 S=S-C;IF S<=0 THEN CHEN=2:S=S0:C=C
    +D;RETURN
115 CHEN=CHEN+1
120 GOSUB 100
130 X0=-1:Y0=-1;GOSUB 100
140 X0=-1:Y0=1;GOSUB 100
150 X0=1:Y0=1;GOSUB 100
160 RETURN
]RUN
S,C,D
```



图二

APPLE 的快速排序

北京航空航天大学机械厂(100083) 张亭

在 APPLE II 或兼容机中输入程序 1—1 和 1—2 并试运行,会令人感到一种速度的享受:排序 $N=1000$ 的数组只须 2 秒,排序 $N=3000$ 的数组只须 8 秒。这就是有名的快速排序算法。

通过对程序 1—1 反汇编可以看到,6502 机器语言程序中使用了递归调用技术,这就是快速排序算法天生具备的特性。程序 2—2 是用 BASIC 语言编制的快速排序程序,由于 BASIC 不允许递归调用,这里采用了一种可称为“层次计数”的编程技巧,程序中变量 C% 就是层次计数器。程序 2—1 是变量交换子程序,

取自《APPLE II 彻底研究(二)》。但即便采用了这种办法,BASIC 的快速排序比汇编仍然要慢 80~90 倍。

那么既然有了风驰电掣的机器语言快速排序,为什么还要 BASIC 慢吞吞的程序呢?这是因为高级语言也有其优势,除易于阅读、理解外,还方便修改和移植。只要把“&”改为 SWAP,就可运行于 16 位以上的高档微机,只要把 50 行和 80 行改为关键字的比较,把 60 行和 90 行改为成组信息的交换,就实现了对多维变量或记录的排序。

笔者将其运行于 10MH 的 286 微机,当数组的 $N=1000$ 时排序时间为 25 秒。可见,随着微机主频率的提高,BASIC 的排序仍然是有意义的。

不过,快速排序对已经有序或基本有序的情况来说效率却是很低的,而反序对快速排序来说则几乎是一个灾难。除了这类特殊情况以外,快速排序也不是稳定排序,即:不能保证关键字相等记录的最初次序。

程序 1—1

```
0249— 20 E3 DF 85 40 84 41
0250— 20 BE DE 20 67 DD 20 52
0258— E7 85 5F 98 0A 26 5F 0A
0260— 26 5F 65 50 85 5E A5 5F
0268— 65 51 85 5F A5 40 65 5E
0270— AA A5 41 65 5F D0 18 E4
0278— 40 D0 04 C5 41 F0 03 20
0280— 97 02 18 8A 69 05 85 40
0288— 90 02 E6 41 68 AA 68 E4
0290— 40 D0 04 C5 41 F0 30 86
0298— 5E 85 5F 48 8A 48 A5 41
02A0— 85 61 A5 40 85 60 38 A5
02A8— 5E E9 05 85 5E B0 02 C6
02B0— 5F C5 60 D0 07 AA A5 5F
02B8— C5 61 F0 BB 20 FD EA 20
02C0— B6 EB B0 E3 A0 04 B3 60
02C8— B1 5E 91 60 8A 91 5E 88
02D0— 10 F4 18 A5 60 69 05 85
02D8— 60 90 02 E6 61 C5 5E D0
02E0— 07 AA A5 61 C5 5F F0 8F
02E8— 20 FD EA 20 B6 EB B0 E2
02F0— A0 04 B3 60 B1 5E 91 60
02F8— 8A 91 5E 88 10 F4 30 A6
```

程序 1—2

```
10 HOME:INPUT "N=";N:C=1000
20 DIM A(N),B(N):FOR I=0 TO N:A(I)=RND(1)*C:B
    (I)=A(I):NEXT
40 P=0:Q=N+1
50 PRINT "PUSH ANY KEY":GET A$
60 CALL 585B(P),Q
70 PRINT CHR$(7);PRINT B(0),B(N/2),B(N)
80 GET A$:FOR I=0 TO N:PRINT A(I),B(I):NEXT:END
程序 2—1
02C6— 20 E3
02C8— DF 85 85 84 86 A5 81 48
02D0— A5 82 48 20 BE DE 20 E3
02D8— DF 68 45 82 30 1F 68 45
```

02E0— 81 30 1A A0 02 24 81 30

02E8— 10 24 82 30 02 A0 04 B1

02F0— 85 48 B1 83 91 85 68 91

02F8— 83 88 10 F3 60 4C 76 DD

03F5— 4C C6 02

程序 2—2

10 GOTO 135

30 Z%(C%)=DW%:C%=C%+1

35 DS%=S%

40 DW%=DW%-1:IF DW%=DS% THEN 100

50 IF A(DS%)>=A(DW%) THEN 40

60 & A(DS%),A(DW%)

70 DS%=DS%+1:IF DW%=DS% THEN 100

80 IF A(DS%)>=A(DW%) THEN 70

90 & A(DS%),A(DW%):GOTO 40

100 IF S%<DS% THEN 30

110 S%=S%+1:DW%=Z%(C%-1):IF S%<DW% THEN 35

120 C%=C%-1:IF C%>0 THEN 110

130 PRINT CHR\$(7):GET A\$:GOTO 150

135 HOME:INPUT "N=":N%:DW%=N%+1:C%=1000:DIM A(N%),Z%(20)

136 FOR I=0 TO N%:A(I)=RND(1)*C%:NEXT C%=0:PRINT CHR\$(7):GET A\$:GOTO 30

150 FOR I=0 TO N%:PRINT A(I):NEXT:END

百年公历— 农历互查程序

陕西中药研究所 刘安军

由已知公历查相应农历、或由已知农历查相应的公历,采用计算的方法很难实现,一般的办法是查表,例如查日历卡或查万年历。在计算机上如果采用建表的方法,要查许多年范围内的日历,数据量相当大,非常困难。本程序采用数据压缩和计算相结合的办法,可以迅速得到百年内某月的公历—农历对照月历,实用、方便,准确无误。

一、基本原理

如果知道公历某月 1 日的农历月份、日期,又知道当月和下月农历的大、小,就可以推算出该月的对照月历。而计算下月公历 1 日的农历日则可以采用公式:

下月公历 1 日的农历日=本月公历 1 日的农历日+本月公历天数-本月农历天数

所以,只要得到每年公历 1 月 1 日对应的农历月份、日期,以及该年农历的大、小月和闰月的信息,就可以推算出该年任何一个月对照月历。但是,公历每年跨接的农历月有 13 或 14 个月,大、小月又无规律可循,加上闰月情况,公历 1 月 1 日的农历月份、日期等,数据量仍较大。

本程序采用数据压缩的办法,经过编码,每年采用一组八位十进制数,实现了以上信息的传递。表一列出了各位数的意义,并以 1992 年为例。表中“大”为大月 30 天,“小”为小月 29 天。从左起第一至第四位,每位用 1~8 八个数字表示三个月的大、小月信息,四位共 12 个月。编码情况见表二。第五位是闰月情况,当该位为“0”时,该年平;为“2”至“9”时,该年闰,数字是几就是闰几月;1984 年闰十月是特例,用“1”来表示。第六位用 1~4 四个数字编码,每个数字代码意义见表三。

表中,C 表示公历 1 月 1 日农历滞后几个月,S 表示农历第 13 个月的大、小月信息。第七、八位,表示农历对应公历 1 月 1 日的日期。

二、使用说明

程序运行后,如要查公历某年某月的农历,则输入公历年、月,屏上显示出该月公历——农历对照月历,农历每月的 1 日为该月的月份,如果是闰月则为“闰”字。如果要查某年农历某月某日的公历日,则输入该年该月或下个月的数字即可查到。打印时应设定行宽为 17 个汉字等有关参数。如要改变程序的查询年限范围,一种办法是直接更换压缩数据。另一种办法是扩大 T 数组,修改程序个别语句,可以直接查数百年甚至上千年的日历,只要不超出内存的变量允许最大范围。

三、程序说明

(一)变量

T 数组:一百年压缩数据,以变量 F 做数组下标;S 数组:农历该年每月天数(公历跨接共 13 个月),以变量 V 做下标;R\$ 数组:农历每月 1 日显示的月份名,变量 N 做中间变量,最后一次做 R\$ 数组的下标;SS\$ 数组:农历“干”名,以变量 BB 做下标;WW\$ 数组:农历“支”名,以变量 CC 做下标;A:待查年;B:待查月;C:滞后月数;L:闰月信息;Q:农历日;P:公历月的天数;E\$:该年压缩数据;S(14):处理 1919、1938 年公历跨农历 14 个月特例;M、I、K、M\$、F\$:中间变量。

(二)部分语句说明

20~90 句,读数据;

100~110 句,取该年压缩数据;

120~130 句,计算农历干支数组下标;

140~330 句,取解码后 13 个月每月天数、滞后月数、闰月信息和农历初始日;

340~450 句,计算待查月份公历 1 日的农历日;

480~640 句,输出公历和农历,每 10 天为一行,公历、农历每天上下对应,直至公历该月最后一日为止;

690~730 句,输出农历为“1”日时的月份名,闰月时为“闰”。

表一

	第一位			第二位			第三位			第四位			第五位	第六位		第七八位
意义	1	2	3	4	5	6	7	8	9	10	11	12	闰月信息	C	S	农历日期
内容	大	大	小	大	大	小	小	大	小	小	大	小	该年平	2	大	公历 1 月 1 日
1992 年编码	2			2			6			6			0	3		27

表二

编 码	月份	1	2	3
		4	5	6
		7	8	9
		10	11	12
1		大	大	大
2		大	大	小
3		大	小	大
4		大	小	小
5		小	大	大
6		小	大	小
7		小	小	大
8		小	小	小

表三

	C	S
1	1	大
2	1	小
3	2	大
4	2	小

```

10 DIM T(100),S(20),R$(15),SS$(10),WW$(12)
20 F$=RIGHT$(M$,2):F=VAL(F$)
30 FOR I=0 TO 99:READ T(I):NEXT
40 FOR I=1 TO 15:READ R$(I):NEXT
50 FOR I=0 TO 9:READ SS$(I):NEXT
60 FOR I=0 TO 11:READ WW$(I):NEXT
70 PRINT CHR$(4)"PR#3":HOME:HGR2
80 INPUT"输入年月: ";A,B:M$=STR$(A)
90 IF A<1900 OR A>1999 OR B<1 OR B>12 GOTO 80
100 F$=RIGHT$(M$,2):F=VAL(F$)
110 E$=STR$(T(F))
120 BB=A-INT(A/10)*10
130 CC=A-INT(A/12)*12
140 I=1,V=1,N=1,S(14)=20
150 M$=MID$(E$,N,1):K=VAL(M$)
160 IF K=1 THEN S(I)=30:S(I+1)=30:S(I+2)=30:
    GOTO 240
170 IF K=2 THEN S(I)=30:S(I+1)=30:S(I+2)=29:
    GOTO 240

```

```

180 IF K=3 THEN S(I)=30:S(I+1)=29:S(I+2)=30:
    GOTO 240
190 IF K=4 THEN S(I)=30:S(I+1)=29:S(I+2)=29:
    GOTO 240
200 IF K=5 THEN S(I)=29:S(I+1)=30:S(I+2)=30:
    GOTO 240
210 IF K=6 THEN S(I)=29:S(I+1)=30:S(I+2)=29:
    GOTO 240
220 IF K=7 THEN S(I)=29:S(I+1)=29:S(I+2)=30:
    GOTO 240
230 IF K=8 THEN S(I)=29:S(I+1)=29:S(I+2)=29:
    GOTO 240
240 I=I+3:IF I<12 THEN N=N+1:GOTO 150
250 M$=MID$(E$,6,1):K=VAL(M$)
260 IF K=1 THEN S(13)=30:C=1:GOTO 300
270 IF K=2 THEN S(13)=29:C=1:GOTO 300
280 IF K=3 THEN S(13)=30:C=2:GOTO 300
290 IF K=4 THEN S(13)=29:C=2:GOTO 300
300 IF C=1 THEN N=2:GOTO 320
310 N=1
320 M$=MID$(E$,5,1):L=VAL(M$):IF L=1 THEN
    L=10
330 M$=MID$(E$,7,2):Q=VAL(M$)
340 FOR I=1 TO 12
350 IF I=4 OR I=6 OR I=9 OR I=11 THEN P=30:GOTO
    420
360 IF I<>2 THEN P=31:GOTO 420
370 IF A/100=INT(A/100) GOTO 400
380 IF A/4=INT(A/4) THEN P=29:GOTO 420
390 GOTO 410
400 IF A/400=INT(A/400) THEN P=29:GOTO 420
410 P=28
420 IF I=B THEN 460
430 Q=Q+P-S(V):V=V+1:N=N+1:IF L<>0
    THEN IF V-2=L THEN N=N-1
440 IF Q>S(V) THEN Q=Q-S(V):V=V+1:N=N+1
450 NEXT
460 PRINT A;"年";B;"月(农历)";SS$(BB);WW$(CC);
    "年";IF L<>0 THEN PRINT"闰";L;"月)";GOTO
    480
470 PRINT")"
480 M=0:K=0
490 PRINT"公历";
500 FOR I=1 TO P
510 IF I<10 THEN PRINT I;" ";GOTO 530
520 PRINT I;" ";
530 IF I=10 OR I=20 OR I=30 GOTO 560

```

```

540 NEXT
550 PRINT
560 PRINT "农历";
570 K=K+1
580 IF Q=1 GOTO 700
590 IF Q>S(V) THEN 690
600 IF Q<10 THEN PRINT Q;" ";GOTO 620
610 PRINT Q;" "
620 Q=Q+1;M=M+1;IF M=>P GOTO 650
630 IF K=10 THEN K=0;HTAB1;PRINT"公历";NEXT
640 GOTO 570
650 PRINT;INPUT"还查吗? (Y/N)";M$
660 IF M$="Y" THEN HOME;GOTO 80
670 IF M$< ">" "N" GOTO 650
680 END
690 Q=1;V=V+1;N=N+1
700 IF L< > 0 AND L-V+C=-1 THEN D=N-1;N=
15
710 PRINT R$(N);" ";
720 IF L< > 0 AND L-V+C=-1 THEN N=D
730 GOTO 620
740 DATA 37558201, 64350311, 36630322, 36655203,
53770304, 55760326, 33634207, 63560317, 37350428,
37612210
750 DATA 64350320, 64426101, 16650413, 53770422,
16375206, 55630416, 37560326, 66222108, 37620319,
24427330
760 DATA 24430311, 16660323, 22665104, 22370415,

```

```

62230425, 63334207, 66330327, 37630328, 37752109,
24430321
770 DATA 34436102, 33660313, 32370424, 33365206,
63350416, 66330326, 66553107, 37750319, 34437330,
54430411
780 DATA 53660322, 55666104, 35630415, 63560325,
63624106, 26610418, 57750428, 17762110, 16430421,
22437203
790 DATA 62260313, 35630424, 36335205, 64220316,
36620327, 36633108, 23760319, 23768101, 23430412,
62360322
800 DATA 63366103, 36330415, 64330325, 64354106,
36630317, 23770329, 53773110, 33430421, 35637202,
63560313
810 DATA 37550424, 37615205, 64350415, 56650427,
16654209, 53770419, 16378201, 55630412, 75560322,
66226103
820 DATA 37620314, 24420326, 24434107, 16660318,
15771429, 22370411, 62230421, 63336202, 76330312,
37630324
830 DATA 37755105, 24430316, 22660327, 33663109,
32370420, 33578210, 63350411, 76330322, 66555203,
57750314
840 DATA [11],[12],[1],[2],[3],[4],[5],[6],[7],
[8],[9],[10],[11],[12], 闰,庚,辛,壬,癸,甲,乙,
丙,丁,戊,己,申,酉,戌,亥,子,丑,寅,卯,辰,巳,午,
未

```

(上接第 23 页)

程序 10.18 设计方法和 10.17 大同小异,只是多调一个显示字符“-”(显示码为 BD),循环次数为 5 次,每次既显示一个“*”号,又显示一个“=”号,当循环完成后,再调显一次“*”,只有这样,才和题意要求符合,由于两个程序选用的计数器不同(前者 X,后者为 Y),因而操作码也有区别。

(8)连续显示全部 ASCII 码对应的字符

若从“!”开始显示到“-”结束,即按“!”# \$? @ABC.....^ - ”显示,可用更为简单的程序,见程序 10.19:

```

0300- 20 58 FC JSR $FC58
0303- A9 A0 LDA # $A0
0305- 20 ED FD JSR $FDED
0308- 18 CLC
0309- 69 01 ADC # $01
030B- D9 E0 CMP # $E0
030D- D0 F6 BNE $0305
030F- 60 RTS

```

值得指出,程序 10.19 通用性好,事实上,由于反相显示和闪烁显示 ASCII 码(16 进制表示)按数值顺序递加,只要改动程序中的 \$0304 为 \$00, \$030C 为

\$40 即可顺序反相显示所有对应显示码的字符(见程序 10.20),改动程序中 \$0304 内容为 \$40, \$030C 内容为 \$80,则可顺序闪烁显示所有对应显示码的字符(见程序 10.21)。

程序 10.20:

```

0300- 20 58 FC JSR $FC58
0303- A9 00 LDA # $00
0305- 20 ED FD JSR $FDED
0308- 18 CLC
0309- 69 01 ADC # $01
030B- C9 40 CMP # $40
030D- D0 F6 BNE $0305
030F- 60 RTS

```

程序 10.21:

```

0300- 20 58 FC JSR $FC58
0303- A9 40 LDA # $40
0305- 20 ED FD JSR $FDED
0308- 18 CLC
0309- 69 01 ADC # $01
030B- C9 B0 CMP # $B0
030D- D0 F6 BNE $0305
030F- 60 RTS

```

搜索内存数据的程序

苏 华

搜索内存中机器语言程序的代码或数据,找出它们的所在地址,对于分析软件是十分有用的。本文所附的搜索程序有以下特点:

1. 在监控状态下以<首地址>、<末地址>^Y (^Y代表Ctrl-Y)方式输入搜索范围,再输入要搜索的代码。在改变地址或退出操作之前,该地址范围持续有效。

2. 由于调用了监控系统的行输入子程序,因此具有该子程序提供的功能,如允许一次输入多达256次键击,键入缓冲区将满的报警、ESC键编辑功能等。

3. 搜索程序的装入地址是可浮动的。

键入搜索程序的装入地址,以命令

```
BSAVE CODESEEK,A$300,L$BC
```

存入磁盘。使用时,执行

```
BRUN CODESEEK,A$<首地址>
```

<首地址>可为任一方便的内存页首址,如9C00等。程序运行设置有关向量后,仍返回监控状态。搜索时,执行

```
<首地址>,<末地址>^Y
```

这时屏幕出现一个“>”提示符,等待输入要搜索的代码。例如,欲查询在Applesoft和监控系统中的哪些地方直接调用了字符输出子程序COUT(\$FDED),可输入

```
*D000.FFFF^Y
```

```
>20 ED FD (指令JSR $FDED的代码)
```

显示:

```
DB64 F903 F91B F923 F94C FAE6 FAEC FAF1
FD47 FD64 FD6C FDB8 FDD6 FE46 FE4B FE55
FF2F FF34 FF37
```

可以看出,Applesoft只在一处(\$DB64)直接调用了COUT。

欲继续查询Applesoft和监控系统在哪些地方调用了显示累加器内容的16进制代码子程序PRBYTE(\$FDDA),输入:

```
>20 DA FD (指令JSR $FDDA的代码)
```

显示:

```
F8D6 F92D F941 FAF6 FDBD FE41 FE50
```

上面的显示告诉我们,Applesoft没有调用PRBYTE。

如果要接着查询Apple DOS在何处直接调用了COUT,可以这样操作:

```
>9D00. BFFF 在>提示下改变地址不需输入^Y.
```

```
>20 ED FD
```

显示:

```
ADB3 ADE5 ADF9 AD FE AE10 AE1B AE31 AE63 B6D4
```

要退出搜索程序,只需在>提示符下键入X,即返

回监控状态。要从监控状态再次运行搜索程序,在给首尾地址后仍需键入^Y。

使用时应注意不要让地址\$C000~\$C0FF落入搜索范围,因为这一页地址内有许多软开关,触动某些软开关会造成屏幕进入图形模式或键盘死锁。这就是为什么在上面查询调用COUT时,Apple DOS与Applesoft及监控要分开两次查询。

使用 GPIB 卡接收外部设备的测量数据时,在 BASIC 程序中要通过 INPUT 语句接收数据。Applesoft 的 INPUT 语句是不接受字符“:”(ASCII 码为 \$3A),因为它被作为语句分隔符处理。GPIB 卡把接收的每一数据字节拆开作为两个低位半字节,以 03 作为高位半字节,这是为了避免象 \$0D、\$0A 之类的数据被系统程序视为控制符而产生误动作。但这样一来就在转换的输入数据中有很多代码为 3A 的数据被 INPUT 语句所略去。用搜索程序查询发现,只要把 Applesoft 内部 \$DC45 处的一条指令 LDA # \$3A 改为 LDA # \$00,便可解决代码为 3A 的数据被遗漏的问题,具体做法是先把 Applesoft 及监控程序移入 RAM 卡,将 RAM 卡内的 \$DC46 单元改为 0,然后在 RAM 卡的 Applesoft 支持下运行 BASIC 程序,这时 INPUT 语句就可以接受代码为 3A 的数据了。

```
300- 20 58 FF BA BD 00 01 8D
308- FA 03 8D FF 03 CA BD 00
310- 01 18 69 1D 8D F9 03 69
318- 0F 8D FE 03 4C 69 FF A2
320- 01 B5 3C 95 FA 95 FC B5
328- 3E 95 FE CA 10 F3 A2 00
330- A9 BE 20 ED FD 20 0C FD
338- C9 D8 F0 E0 20 78 FD A0
340- 00 84 06 A9 02 85 07 20
348- A7 FF C9 A7 D0 0B 85 31
350- 20 A7 FF 20 C7 FF 6C F9
358- 03 C9 C6 F0 04 49 99 D0
360- 4A 48 A2 00 A5 3E 81 06
368- E6 06 D0 02 E6 07 68 F0
370- D6 A0 00 A5 06 85 08 38
378- A5 FE E5 FC A5 FF E5 FD
380- 30 2C B1 FC D9 00 02 D0
388- 1E C8 C6 08 D0 F4 A5 FD
390- A6 FC 20 41 F9 A9 A0 20
398- ED FD A4 06 E6 FC D0 02
3A0- E6 FD 88 D0 F7 F0 CC A0
3A8- 01 D0 F1 20 3A FF 20 8E
3B0- FD A5 FA 85 FC A5 FB 85
3B8- FD 6C FE 03
```

第十讲 监控子程序的调用(一)

南京大学大气科学系(210008) 朱国江

中华学习机系统程序中,有很多可供用户调用的子程序,其中最有用的要算监控中的子程序,如果我们能充分利用这些子程序,这对于减少用户编程的工作量和节约内存空间,都有实际意义。不仅如此,这些子程序简短独立,功能完整,结构合理,简练灵活,如能合理巧用,往往起到事半功倍的作用,从而使程序清晰易懂,并能提高程序的执行速度。

本讲介绍常用监控子程序及其使用实例。

〔例 1〕将 26 个英文大写字母 A~Z 输出给打印机。

输出一个字符子程序的入口地址为 \$FDED,该子程序既可以将字符输出至屏幕,也可以将字符输出给打印机。若输出给打印机,则应先将设备码 C100 存入 \$36 和 \$37 中。\$36 单元存放设备码低字节 \$00, \$37 单元存放设备码高字节 \$C1。然后调用入口为 \$FDED 的输出字符子程序,这样就可以把累加器 A 中存放的字符送往所指定的外部设备(本例是打印机)。当该子程序执行完毕后,又会回到刚才调用它的程序。源程序见 10.1。

程序 10.1:

0300- A9 00	LDA # \$00	} 设备送往打印机的设备码
0302- 85 36	STA \$36	
0304- A9 C1	LDA # \$C1	
0306- 85 37	STA \$37	
0308- A9 C1	LDA # \$C1	取字符“A”的 ASCII 码
030A- 20 ED FD	JSR \$FDED	显示
030D- 18	CLC	清进位
030E- 69 01	ADC # \$01	指向下一个
0310- 89 DB	CMP # \$DB	打完了吗? “Z”的 ASCII 码为 DA
0312- D0 F6	BNE \$030A	没有,再打
0314- 60	RTS	打完结束

〔例 2〕在屏幕上连续显示 15 个星号“*”

开机后,通常 \$36 和 \$37 的内容为 F0 和 FD。\$FDF0 就是一个设备码,它的功能是将输出字符送至显示器,因此,不需要重新设置设备码,而直接调用 \$FDF0 的子程序,即可将累加器 A 中的字符输出到屏幕。源程序见 10.2

程序 10.2:

0300- A2 0F	LDX # \$0F	置初值 15
0302- A9 AA	LDA # \$AA	调“*”的 ASCII 码
0304- 20 F0 FD	JSR \$FDF0	显示
0307- CA	DEX	X-1→X
0308- D0 FB	BNE \$0302	未减完,再循环显示
030A- 60	RTS	减完,结束

〔例 3〕用反相显示方式在屏幕上第一列显示 16

个“+”号。

置反相显示方式的子程序入口地址为 \$FE80,该子程序置屏幕为白底黑字方式。

置正相显示方式的子程序入口地址为 \$FE84,该子程序可将原反相显示方式转为正相显示方式,即置屏幕为黑底白字方式。

程序 10.3

0300- 20 80 FE	JSR \$FE80	设置反相显示方式
0303- A0 10	LDY # \$10	置初值 16
0305- 20 8E FD	JSR \$FD8E	回车换行
0308- A9 AB	LDA # \$AB	“+”号的 ASCII 码
030A- 20 F0 FD	JSR \$FDF0	显示
030D- 88	DEY	次数减 1
030E- D0 F5	BNE \$0305	未完,继续
0310- 20 84 FE	JSR \$FE84	恢复正常显示方式
0313- 60	RTS	结束

程序 10.3 中用了调用 \$FD8E 的子程序,该子程序将回车换行的控制字符送往屏幕,以保证本题“+”号在一列上显示而不是一行上显示。

〔例 4〕将 \$03F0~\$03FF 单元中的两位十六进制数显示出来。

输出两位十六进制字符子程序的入口地址是 \$FDDA,该子程序的功能是将累加器 A 中的内容,按两位十六进数的形式输出到现行输出设备。

程序 10.4

0300- A9 F0	LDA # \$F0	} (04)(03)=03F0
0302- 85 03	STA \$03	
0304- A9 03	LDA # \$03	
0306- 85 04	STA \$04	
0308- A0 00	LDY # \$00	变址计数=0
030A- B1 03	LDA (\$03),Y	第一次调 \$03F0 内容→A
030C- 20 DA FD	JSR \$FDDA	显示两位十六进制数
030F- A9 A0	LDA # \$A0	空格“ ”的 ASCII 码
0311- 20 F0 FD	JSR \$FDF0	显示
0314- C8	INY	计数器+1
0315- C0 10	CPY # \$10	16 个字节内容都送完吗?
0317- D0 F1	BNE \$030A	否,继续循环
0319- 60	RTS	是,结束

程序 10.4 中取数指令 LDA(\$03),Y 是采用的后变址 Y 间接寻址方式,它常用于动态数据块处理,即数据块在存储器中存放的位置是可以变化的,只要把数据块首地址置入所选用的零页地址中即可。

〔例 5〕已知在 \$03F8~\$03FF 单元中,存放四个转移地址,现要求将它们打印出来。

输出四位十六进制数的子程序入口地址为 \$F941, 它的功能是将寄存器 X 和累加器 A 中的内容按四位十六进制数的形式输出到现行输出设备。

调用 \$F941 之前, 必须将高字节放入累加器 A 中, 低字节放在 X 寄存器中。

程序 10.5

```
0300- A0 02    LDY  $03F9,Y    变址计数=2
0302- AD F9 03  LDA  $03F8,Y    } 设置源数据块首址
0305- AE F8 03  LDX  $03F8      }
0308- 20 41 F9  JSR  $F941      显示四位 16 进制数
030B- A9 A0     LDA  # $A0      } 空一格
030D- 20 F0 FD  JSR  $FDF0      }
0310- B9 F9 03  LDA  $03F9,Y    第一次取 $03FA~
0313- BE F8 03  LDX  $03F8,Y    $03FB 内容。
0316- 08        INY            } 计数器加 2
0317- C8        INY            }
0318- C0 0A     CPY  # $0A      数据块有无送完?
031A- D0 EB     BNE  $030B      没有, 继续
031C- 60        RTS            是的, 结束
```

LDA \$03F9,Y 和 LDX \$03F8,Y 均采用绝对 Y 变址寻址方式。

[例 6] 把 \$03F0~\$03FF 中各单元内容按两位 16 进制数显示出来。要求每两个数之间空三个空格。

\$F948 是把三个空格字符送往现行输出设备的子程序入口地址。

程序 10.6

```
0300- A0 00    LDY  # $00      初始化计数器
0302- B9 F0 03  LDA  $03F0,Y    调($03F0+Y)的内容
0305- 20 DA FD  JSR  $FDDA      显示两位十六进制数
0308- 20 48 F9  JSR  $F948      空三格
030B- C8        INY            计数器加 1
030C- C0 10     CPY  # $10      送完吗?
030E- D0 F2     BNE  $0302      否, 再送
0310- 60        RTS            是, 结束
```

在监控子程序中, 还有一个可以输出 X 个空格的字程序, 其入口地址为 \$F94A, 但调用该子程序前, 必须将空格数设置在寄存器 X 中。

[例 7] 不断按键取字符送显示屏显示, 若按下空格键时则结束。

取一按键的值(取一个输入字符)的子程序入口地址为 \$FD0C, 该子程序可以接受从键盘输入的一个字符。

程序 10.7

```
0300- 20 0C FD  JSR  $FD0C      取一按键
0303- 20 F0 FD  JSR  $FDF0      显示
0306- C9 A0     CMP  # $A0      是空格吗?
0308- D0 F6     BNE  $0300      不是, 继续
030A- 60        RTS            是, 停机
```

读取键盘数据还有一个子程序, 其入口地址为 \$FD1B, 它能等待按键输入, 当有键按下时, 该子程序就将该键值送往屏幕光标当前位置和累加器 A 中。

[例 8] 编制几个不同延时时间的延迟程序

在监控程序中, 有一个延时程序, 其入口地址为 \$FCA8, 可以直接调用。

程序 10.8、10.9、10.10、10.11、10.12 是用不同方法编制的几个延时程序。

程序 10.8

```
0300- A2 40    LDX  # $40      计数器置初值
0302- A9 FF    LDA  # $FF      累加器置初值
0304- 20 A8 FC  JSR  $FDA8      调监控延迟子程序
0307- CA       DEX             X-1→X
0308- D0 F8     BNE  $0302      只要 X 不为零, 循环
030A- 60        RTS            X=0, 结束
```

程序 10.9

```
0300- A9 FF    LDA  # $FF      置延时常数
0302- 38        SEC            置进位标志 C=1
0303- E9 01     SBC  # $01      A-01→C
0305- D0 FB     BNE  $0302      若不为 0, 循环
0307- 60        RTS            是 0, 结束
```

程序 10.10

```
031D- A0 10    LDY  # $10      外循环计数器初值
031F- A2 FF    LDX  # $FF      内循环计数器初值
0321- CA       DEX             X-1→X, 为 0 否
0322- D0 FD     BNE  $0321      X 不为 0, 继续内循环
0324- C8        INY            X=0, 外循环次数 Y+1
0325- D0 F8     BNE  $031F      Y≠0, 继续外循环
0327- 60        RTS            Y=0, 结束
```

程序 10.11

```
031D- A2 10    LDX  # $10      外循环计数器初值
031F- A0 FF    LDY  # $FF      内循环计数器初值
0321- 88        DEY            Y-1→Y
0322- D0 FD     BNE  $0321      Y≠0, 继续内循环
0324- E8        INX            Y=0, 外循环次数 X+1
0325- D0 F8     BNE  $031F      X≠1, 继续外循环
0327- 60        RTS            X=0, 结束
```

程序 10.12

```
FCA8- 38        SEC            置进位标志 C=1
FCA9- 48        PHA            累加器 A 的内容进入堆栈
FCAA- E9 01     SBC  # $01      (A)-1→C→A
FCAC- D0 FC     BNE  $FCAA      (A)≠0 则循环
FCAE- 68        PLA            A 的内容出栈
FCAF- E9 01     SBC  # $01      (A)-1→A
FCB1- D0 F6     BNE  $FCA9      循环至(A)=0
FCB3- 60        RTS            (A)=0, 结束
```

几点说明:

• 程序 10.8 是一个延时程序, 但其中又调用了监控中的延时程序 \$FCA8。

• 程序 10.9 中用了 SBC 指令, 使用前必须用 SEC 指令将 C 标志位置为 1, 否则影响运算结果。

• 10.10、10.11 其程序设计思想完全一样, 都是用双重循环控制循环次数的办法达到延时的目的。但内、外循环计数器选用的不同, 因而寄存器 X(或 Y)内容增、减一的指令也不同, 使用时应注意搭配。

10.12 是监控延迟子程序,采用了堆栈处理方法,调用前先设置累加器 A 中的初值,然后调用,如:

```
0300-A9 20 LDA # $20
0302-20 A8 FC JSR $FCA8
0305-60 RTS
```

〔例九〕显示字符的各种方法

为了在屏幕上显示字符或汉字(一个或多个),可以有不同的方法。而字符的显示方式也可以多样(黑底白字的正常显示、白底黑字的反相显示、闪动显示)。字符可以在固定的位置上显示,也可以在屏幕上用“开窗口”的方法,设定一块显示区显示。

(1)单个字符的正常显示

例如欲显示一个“A”字符,取正常方式,在文本状态第一页的起始位置上,可用程序 10.13:

程序 10.13

```
0300- 20 58 FC JSR $FC58 清屏
0303- A9 C1 LDA # $C1 把 C1 送入累加器 A 中
0305- 8D 00 04 STA $0400 把 A 中的数存入地址 0400
0308- 60 RTS 返回
```

其中 C1 是字符“A”正常显示方式的显示码。\$0400 是屏幕文本第一页第一个位置的映像地址。在监控状态下,300G \swarrow ,即可看到屏幕左上方显示一个“A”。

(2)调用输出一个字符的监控子程序,正常显示“A”字符,见程序 10.14

```
0300- 20 58 FC JSR $FC58
0303- A9 C1 LDA # $C1
0305- 20 ED FD JSR $FDED
0308- 60 RTS
```

(3)连续显示几个反相字符

例如,反相显示“BASIC”五个字符。

方法是反复调用 LDA 和 JSR 这两个指令,见程序

10.15:

```
0300- 20 58 FD JSR $FC58
0303- A9 02 LDA # $02
0305- 20 ED FD JSR $FDED
0308- A9 01 LDA # $01
030A- 20 ED FD JSR $FDED
030D- A9 13 LDA # $13
030F- 20 ED FD JSR $FDED
0312- A9 09 LDA # $09
0314- 20 ED FD JSR $FDED
0317- A9 03 LDA # $03
0319- 20 ED FD JSR $FDED
0310- 60 RTS
```

300G \swarrow 后,在屏幕左上方自左至右反相显示“BASIC”五个字符。# \$02, # \$01, # \$13, # \$09, # \$03 分别为字符 B,A,S,I,C 的反相显示代码

(4)用循环的方法显示(闪烁)字符串

例如,闪动显示“6502 CPU”这七个字符。见程序

10.16:

```
0300- 20 58 FC JSR $FC58 清屏
0303- A2 00 LDX # $00 寄存器 X 置初值 00
0305- BD 11 03 LDA $0311, 将(0311+X)的数送累加器
X
0308- 20 ED FD JSR $FDED 显示累加器 A 中的字符
030B- E8 INX X+1 $\rightarrow$ X
030C- E0 07 CPX # $07 比较 X 的值是否与 07 相符
030E- D0 F5 BNE $0305 如果不等转 0305
0310- 60 RTS 相等则返回
0311- 76 75 70 72 43 50 55 $0311 开始存放 6502CPU 的闪
动方式的 ASCII 码
```

程序 10.16 的设计思想如下:

首先初始化 X 寄存器,置初值为 00,然后取 0311 + X = 0311 地址的值 76(字符 6 的闪烁显示码)至累加器 A,接着显示这个字符(调 \$FDED 输出一个字符),让寄存器 X+1,判断是否与 07 相符(6502CPU 共 7 个字符),不相等再循环上去(到 \$0305),重复上述过程;若相等则结束。

(5)连续显示若干个相同字符

例如,连续显示 20 个“*”,可用程序 10.17:

```
0300- 20 58 FC JSR $FC58
0303- A2 00 LDX # $00
0305- A9 AA LDA # $AA
0307- 20 ED FD JSR $FDED
030A- E8 INX
030B- E0 14 CPX # $14
030D- D0 F8 BNE $0307
030F- 60 RTS
```

程序 10.17 采用单重循环反复相减的方法,控制显示“*”的次数。循环之前清屏、计数器置 0、取“*”的显示码 AA,循环开始显示“*”,计数器加 1,只要不是 20 次(\$14),继续循环显示,直到计满 20 次跳出循环结束运行。

(7)间隔显示一串不同字符

例如,间隔显示: * - * - * - * - * - *,参见

程序 10.18:

```
0300- 20 58 FC JSR $FC58
0303- A0 00 LDY # $00
0305- A9 AA LDA # $AA
0307- 20 ED FD JSR $FDED
030A- A9 BD LDA # $BD
030C- 20 ED FD JSR $FDED
030F- C8 INY
0310- C0 05 CPY # $05
0312- D0 F1 BNE $0305
0314- A9 AA LDA # $AA
0316- 20 ED FD JSR $FDED
0319- 60 RTS
```

(下转 19 页)



BJS—51 单片机实验系统(续)

北京广播电视大学 李广弟

(7) 思考题

①如各轮比较按实际需要的次数进行,而不是预设的七次,应对本实验程序作哪些修改?

②同样是单字节无符号数,如进行降序排序,需对本实验程序作哪些修改?

2. 交通灯控制实验

除汇编语言程序设计之外,单片微机基本实验的另一方面内容是接口及应用方面的实验。现以交通灯控制为例进行说明。

交通灯控制是一个比较典型的控制类实验,使用实验器上的两组红黄绿发光二极管和两支数码管可以很形象地模拟实现。教程中按复杂程度不同,把这个题目分为四个既相互独立又互有联系的实验,供不同层次的学生选作。这四个实验分别是:

①交通灯定时控制实验

要求两组交通灯定时按规则变化,即模拟普通路口的交通灯控制。本实验将使学生在编写计时程序和实现状态输出控制等应用技术方面得到实践机会。

②带时间显示的交通灯定时控制实验

本实验是在前一个实验的基础上,加进时间显示,使交通灯状态转换与时间变化协调配合。这将使 LED 显示技术得到充分的应用。

③主支线路口的交通灯控制实验

由于主线和支线交通繁忙程度相差悬殊,因此主支线路口的交通灯应根据车辆情况进行随机控制。为此在实验中需引入中断技术和查询技术。从而构成一个较大型的综合实验。

④有救护车优先的交通灯控制实验。

本实验同样需引入中断控制技术,并且中断处理内容更加复杂。

下面以主支线路口交通灯控制实验为例对接口控制类实验的模式进行说明。

1. 实验题目

由主线和支线构成的路口,交通控制原则是尽可能保证主线车辆畅通,在有需要时才开通支线。为此,交通灯共有四种控制状态:

①在通常情况下,主线为绿灯,支线为红灯。

②如支线有车辆到达,则延迟 5 秒后使主线由绿灯变黄灯。再延迟 5 秒后,主线由黄灯变红灯,支线由红灯变绿灯。

③支线放行最长可持续 25 秒,然后变黄灯再经 5 秒后变为红灯。同时主线由红灯变绿灯。

④支线绿灯期间,若主线有 3 辆以上车辆到达,则

在第三辆车到达之时,使支线变为黄灯,延迟 5 秒后变红灯,同时主线由红灯变绿灯。

2. 实验目的

- 了解主支线路口交通灯控制的模拟实现方法。
- 学习计数中断的使用方法。
- 学习如何运用查询技术对外部状态变化作出反应。

3. 实验内容

- 以二组发光二极管模拟二组交通灯并进行电路连接。
- 编写主程序、时间延迟子程序及中断服务程序。
- 输入、调试并运行程序。
- 观察程序运行结果,看发光二极管是否按要求

规律变化:

键 1 代表支线有车辆到达,按一下后主线的黄色发光二极管亮,然后转红色亮,同时支线的绿色发光二极管亮。

键 2 代表主线车辆到达,在支线放行期间,按键 2 三次,代表主线已有三辆车到达。这时支线的黄色发光二极管亮,然后转红色亮,同时主线的绿色发光二极管亮。

4. 电路连接

本实验电路设计有如下要点:

①发光二极管的连接同“定时交通灯控制实验”完全一样。

②键 1 用来模拟支线车辆到达,按下后电平为 0。把键 1 与 P1.7 连线,这样,通过测试 P1.7 的电平状态,就可以了解支线是否有车辆到达。

③键 2 用来模拟主线车辆的到达。按下后电平为 0,通过 P3.5(定时器 1 的外部输入端)输入,因此键 2 与 P3.5 连线(电路连接图略)。

5. 中断技术的应用

当支线放行时,按键 2 三次,代表主线已有三辆车到达,这时应提前结束支线放行状态。对这种控制要求最好使用中断技术实现。具体说就是计数溢出中断。为此要使用 8031 的定时器/计数器进行按键次数的计数,因为计数值仅为 3,所以使用定时器/计数器 1。根据交通灯控制的需要,应采用模式 2 工作方式,以利用其溢出后计数值自动重装的特点。这时 TL1 构成一个可自动装载的 8 位计数器,计数溢出时,不但能发出中断请求,而且还能实现计数值的重新装入,实现控制的连续性。

①定时器/计数器初始值设置

TH1 为预值寄存器(预置值为 FDH)

TL1 为计数器预置值也为 FDH

②模式控制寄存器(TMOD,地址 89H)的状态设置

置

GATE	C/T	M1	M0	GATE	C/T	M1	M0
0	1	1	0	0	0	0	0

定时器/计数器 1(TMOD)=60H

③控制寄存器(TCON,地址 88H)的状态设置

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
0	1	0	0	0	0	0	0

(TCON)=40H

④中断优先级控制器(IP,地址 B8H)

本系统只有一个中断请求,因此无需设定优先级,故取其值为 00H。

⑤允许中断寄存器(IE,地址 A8H)的状态设置。

EA	X	ET2	ES	ET1	EX1	ET0	EX0
1	0	0	0	1	0	0	0

(IE)=88H

⑥各寄存器状态设置汇总:

寄存器名称	IE	IP	TCON	TMOD	TH1	TL1	
地址	A8H	B8H	88H	89H	8DH	8BH	
内容	88H	00H	40H	60H	FDH	FDH	

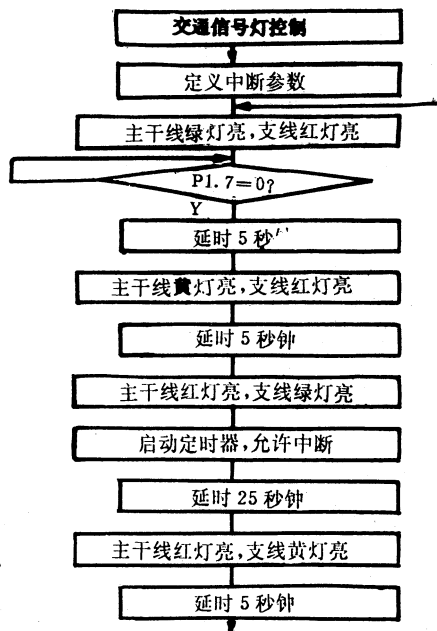
7. 参考程序清单

主程序

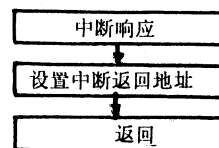
8000	0130		AJMP MAIN	
801B	0190		AJMP INT	
8030	758960	MAIN:	MOV TMOD, #60H	定义 T1 为工作方式 2
8033	758BFD		MOV TL1, #FDH	计数器初始值
8036	758DFD		MOV TH1, #FDH	计数器预置值
8039	7590F3	LOOP1:	MOV P1, #F3H	主线放行,支线禁止
803C	E590	LOOP2:	MOV A,P1	
803E	20E7FB		JB ACC. 7,LOOP2	测试支线有车到达否
8041	7F05		MOV R7, #05H	支线有车
8043	1180	LOOP3:	ACALL DEL	5 秒延时
8045	DFFC		DJNZ R7,LOOP3	
8047	7590F5		MOV P1, #F5H	主线警告,支线禁止
804A	7F05		MOV R7, #05H	
804C	1180	LOOP4:	ACALL DEL	5 秒延时
804E	DFFC		DJNZ R7,LOOP4	

6. 参考程序流程:

主程序流程



中断服务程序流程



8050	7590DE		MOV P1, #DEH	主线禁止,支线放行
8053	758840		MOV TCON, #40H	启动计数器
8056	75A888		MOV IE, #88H	中断允许
8059	7F19		MOV R7, #19H	
805B	1180	LOOP5:	ACALL DEL	25 秒延时
805D	DFFC		DJNZ R7, LOOP5	
805F	7590EE		MOV P1, #EEH	主线禁止,支线警告
8062	758800		MOV TCON, #00H	关闭计数器
8065	D7F05		MOV R7, #05H	
8067	1180	LOOP6:	ACALL DEL	5 秒延时
8069	DFFC		DJNZ R7, LOOP6	
806B	0130		AJMP LOOP1	转主线放行,支线禁止

中断服务程序

8090	745F		MOV A, #5FH	
8092	C0E0		PUSH A	
8094	7480		MOV A, #80H	
8096	C0E0		PUSH A	
8098	32		RETI	

(1 秒延时子程序略)

编者按:武汉创意电子研究所研究人员吴微、马国敏、孔曙光、罗维国曾经在《电子与电脑》杂志上(1990年第2、3期)公布了“SCB-1 MCS-51 单片单板机”的有关论文(笔名吴中国)。去年他们又在北航出版社由何立民教授主编的《单片机应用文集》中以十五万字篇幅公布了“SCB-2 MCS-51、8098 单片单板机”的论文。CYSCB-2 MCS-51、8098 单片单板机则作了进一步的改进,硬件向用户公布,采用积木式结构,使用仅需+5V 供电的 TTL-RS232 转换的 MAX232、复位键置于主板上,有电视接口选件板、全系列高速 EPROM 编程板选配,提供总线输出等。新型机配有 MCS-51、MCS-96 二种本机板和二种 CRT 式监控,软件在文集中已公布,加之高位机交互式菜单构成的汇编、反汇编、编辑及通信,使得用户通过它既能学习、掌握 MCS-51、8098 二种单片机,又能开发以其为核心的新产品。新型机 A/D、D/A、PIO 配备齐全,能直接用于许多控制项目中。需要详细硬件原理及软件清单者请与武汉创意电子研究所联系。

CYSCB-2 MCS-51 8098 单片单板 机硬件设计原理

吴微 罗维国 马国敏 孔曙光

一、CYSCB-2 MCS-51、8098 单片单板机主要技术指

标:

1. 核心单元是 MCS-51 的 8031 芯片或准 16 位单片机 8098。插上 8031 芯片则成为 8031 单片单板机,插上 8098 则成为 8098 单片单板机,两者不能同时插上。
2. 系统时钟选为 8MHz 和 12MHz 两种供用户选择。
3. 外部 EPROM 为 27128 一片,16K 程序存储器,低 8K 为 8031 本机板监控,高 8K 为 8098 本机板监控。
4. 外部 RAM 为 6264 二片共 16K,其中 8K 掉电保护。
5. 设有 8255 可编程 I/O 口,提供 24 位口线,可直接驱动打印机,开关量等。
6. 8031、8098 的 I/O(高速 I/O)等口线通过插座输出。
7. 通过仅需+5V 供电的 MAX232 实现 TTL 与 RS232 之间的转换,克服了以前需±12V 供电的困难。
8. 通过 DAC0832,1 路 8 位 D/A,可进行数模转换,模拟输出 0~5V。8098 则有一路 PWM 输出。
9. 通过 ADC0809,8 路 8 位 A/D,可进行模数转换,模拟输入 0~5V,8098 片内有 4 路 10 位 A/

D.

10. 可选配件有高速智能 EPROM 编程器, 可以实现对 2716~27512 的高速编程, 解决了低速编程器编程速度慢、无检验的问题。
11. 可选配件 TV 接口板实现主板与普通电视或 CRT 监视器的接口, 方便现场操作。
12. 键盘显示板通过 20 芯电缆线与主机板相接, 键盘显示板采用最简洁的三片结构, 使用方便可靠, 主板与键盘显示板之间的接口作了预留, 方便用 8279 实现键盘显示, 并可提供 8279 控制的键盘显示配件。
13. 总线通过 40 芯电缆线插头输出, 方便用户扩展外围芯片。
14. 提供二套监控, 本机板监控方便现场操作调试, 主板 CRT 监控方便实验室调试扩展, 特别是 CRT 监控仅使用 TXT、RXD 二根口线, 其它所有资源全部提供给用户, 为用户开发提供强有力的手段。
15. 具有外部抗干扰 Watch Dog 电路提高系统的可靠性, 8098 则具有内部 Watch Dog 电路。

二、CYSCB-2 MCS-51、8098 单片单板机硬件原理:

1. CYSCB-2 型单片单板机硬件概述:

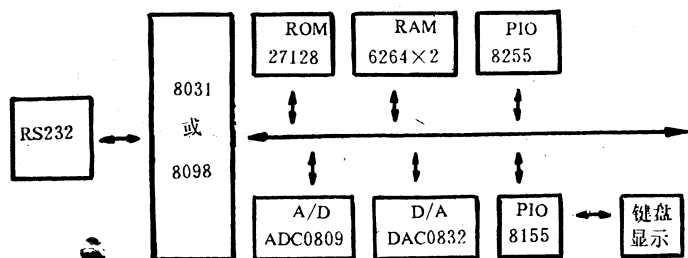


图 1 CYSCB-2 型单片单板机硬件结构框图

CPU: 8031 或 8098, 插上 8031 则成为 MCS-51 单片单板机, 插上 8098 则成为 8098 单片单板机。成品机上 40 脚插座插 8031, 48 脚插座插 8098, 但不能将两种单片机同时插上。

EPROM: ROM 选用一片 16K EPROM 27128, 其中前 8K 存放 MCS-51 监控程序, 后 8K 存放 8098 监控程序。

RAM: 外部随机存储器选用 2 片静态 8K RAM 6264, 这样机上外部 RAM 容量为 16K。外部高 8K RAM 配有掉电保护电路。在这二片 RAM 的插座上可以对 2864 进行编程。

PIO: 外部并行口选用了一片 8255, 利用它通过接一智能 EPROM 编程器, 能对全系列 EPROM 进行高速编程, 此外还可驱动打印机等。

A/D: 外部配的 A/D 是 8 路 8 位的 ADC0809, 若是 8098 单片机, 则片内有 4 路 10 位的高速 A/D。

D/A: 外部配有 D/A 是 1 路 8 位的 DAC0832, 若是 8098 则片内有 1 路脉宽调制输出。

键盘显示电路采用 8155 和 2 片 ULN2003 构成的最简单的三片结构, 可靠性大大提高。

键盘输入: 本机配有 26 个压电式小按键, 其中主板上有一个作复位键使用, 16 个做数值键, 9 个做命令键。

显示输出: 显示输出由六个共阴数码管组成, 左边 4 个用来显示地址或状态信息, 右边 2 个显示数据或代码。

RS232 接口: CYSCB-2 型单片单板机借助 8031 或 8098 片内 SIO 的, 通过一片 MAX232 实现简易 RS232 的接口, 通过该接口与高位机进行通信, 借助高位机 TTY 的模拟程序能对单片机实现更方便的调试。

2 CYSCB-2 型单片单板机硬件分析:

图 2 是 CYSCB-2 型单片单板机硬件电路原理图, 下面详细分析几个核心电路:

1 8 位单片机 8031 和准 16 位单片机 8098

①8031 总线与 8098 总线

我们将 8031 和 8098 二种单机的数据/地址总线联一起, 其含义仅指二种单片机共用 EPROM、RAM、键盘、显示等。当 CYSCB-2 型单片单板机作 MCS-51 单片单板机时, 仅把 8031 插上, 不能插 8098; 当作为 8098 单片单板机时, 仅把 8098 插上, 不能插 8031, 否则单片单板机不但不能工作, 而且可能损坏二种单片机。当然, 若我们在硬件上加上 4 片 74LS244 和二片 74LS245, 通过开关控制平时仅有一种单片机上总线工作, 但硬件成本增加。

②统一编址:

大家知道 8031 程序存储器和数据存储器是分开编址, 8098 则是统一编址。为了兼顾到二者使用统一的 EPROM、RAM、键盘、显示等, 再者考虑到 CYSCB-2 型单片机供学习、调试程序需用 RAM 存储机器码, 我们将 8031 的 \overline{PSEN} 与 \overline{RD} 二根信号线相与形成统一读信号线 \overline{RD} , 这样对 8031 而言外部 EPROM 和 RAM 也须统一编址。将 8031 \overline{RD} 与 8098 单片机 \overline{RD} 的信号线相与则形成 CYSCB-2 单片单板机的读信号线。

2 译码电路

上面谈了 CYSCB-2 型单片单板机采用统一编址, 而完成统一编址是译码电路。图 2 是译码电路的详细描述, 下面介绍挂在总线上的各部分电路的地址。

27128 地址: 0000H~3FFFH 或 4000H~7FFFH

6264 (M_3) 地址: 4000H~5FFFH 或 0000H~1FFFH

6264 (M_4) 地址: 6000H~7FFFH 或 2000H~3FFFH

27128 与 2 片 6264 地址切换取决于外接开关 CSW6。A/D、D/A、8155、键盘以及显示地址详见有关电路图。

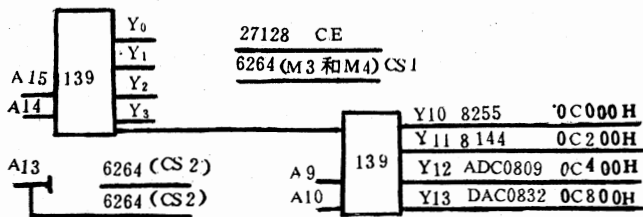


图 2 CYSCB-2 型单征单板机译码电路

3 掉电保护电路

CYSCB-2 型单片单板机采用了一比较简便的外部 RAM 掉电保护电路,当外部加电源时 2 片 6264 由外部供电,当外部电源断开时,其中一片 6264 由一个 4.5V 的纽扣电池供电。

4 对 8031 运用的外部 WATCHDOG 电路

8098 内部有 WATCHDOG 电路,对 8031 而言,只

有增加外部 WATCHDOG 电路才能增加系统的抗干扰性,通过 P1.0 在时间 $T=1/(2 * C_{12} * R_1)$ 内用正沿触发 4098,则系统处于正常工作,若在此时间内没有正沿触发,则系统自复位。当读者不使用 WATCHDOG 电路时,则可将 4098 拨下。关于 CD4098,读者可参考 CMOS 手册。

5 总线输出

CYSCB-2 MCS-51、8098 单片单板机通过 JP3 输出数据总线和地址总线,读者可以扩充相应的接口芯片,值得提醒读者的是若打算使用本机板上大多数芯片,同时还需扩充,则需在总线上通过 74LS244, 74LS245 扩充总线的驱动能力。

6 不需±12V 供电的 TTL↔RS232 转换电路

MAX232 是国外最新推出的 TTL↔RS232 转换芯片,该芯片外接 4 个 10μ 的电容即能实现单+5V 供电的 TTL↔RS232 转换,详细请参考电路图及 MAX232 说明书。

网络管理信息系统快速开发工具 支持“快速原型——增量”开发方法

C-DBAG 中文数据库系列应用生成器新产品

F/D * AGV5.0

北京航空航天大学开发

使用本系统只需按照提示输入用户需求,不需编程,即可生成(如财会、金融、统计、销售等)各种管理软件,在功能变动时可任意改动,适用于网络及单用户环境。

功能: • 生成 DBASE+/FoxBASE+源程序

- 生成任意格式报表
- 自动生成数据字典及文档
- 工程图形生成与管理

详细功能请看本期软件介绍栏目

组成: • 下拉式/弹出式菜单生成器

- 数据库文件生成器
- 数据录入与维护模式生成器
- 查询模块生成器
- 统计与计算模块生成器
- 统计图形生成器

• 任意格式报表生成器

• 数据字典与文档生成器

• 92 年 10 月 27 日在北航开办培训班

经销:中电华北公司电脑部 电话:81.1810

地址:北京万寿路西街五号

邮编:100036

联系人:石立军 魏 国

北京景文科技开发公司

电话 832.2255 转 458

地址:北京西直门外首体主楼 305

邮编:100081

联系人:袁凯峰 李应知

本公司经营计算机及外设通讯设备,欢迎来电来函联系。

大量供应显示器及大功率开关电源

我部现货供应进口 19" 彩色、单色高分辨(1280×1024)图形显示器;12" 终端,并供应大功率开关电源,规格有:5V/18A,5V/20A,+5V/50A,+5V/100A,5V/120A,5V/150A×2,12V/15A。并有各种进口电脑装配电线、电缆(单层屏蔽、多层屏蔽、单芯、多芯、镀银等)品种繁多,适合各类科研开发、生产、装配、工业自动化等单位使用。

以上产品均全部进品并通过美国 UL 安全标准认证,产品批发价相当优惠,欢迎来函、来电或来人洽谈。

广东省四会县南方电子厂经营部

地址:广东省四会县城高观东路 71 号

邮编:526200

电话:(07663)322686

电挂:1311

告读者、作者和各界朋友

本刊编辑部

随着改革开放的深入、科技事业的发展,随着人们对电脑认识的深化,今后将会有越来越多的家庭和初学者在“早日学电脑、终生都受益”的启示下,走入电脑爱好者的行列。

《电子与电脑》是电脑爱好者的朋友,希望大家充分利用这块园地,交流用机经验、编程技巧、开发实验的制作与体会。通过杂志的系列讲座和函授班较系统的学习软硬件知识和技术。值此 1992 年第四季度到来之际,我们向大家通告,93 年本刊编排和改进计划。本刊仍以初中级电脑用户为对象结合国内常用机型(PC 系列、中华机系列、单片与单板机系列),普及软硬件知识和用机经验。在保持现有栏目的情况下,93 年将增加二个新栏目,并不定期试办信息性栏目。为此,随着栏目的扩充,正文由每期 48 页增加到 56 页,定价调整 1.60 元。

新增栏目为：

“IC 与应用”微电脑和微电子技术是一对孪生兄弟。随着科技发展的需要,大家迫切要知道国外 IC 的

发展及最新 IC 的应用资料,本刊将努力搜集这方面的资料,介绍给大家。诸如,开关电源 IC、图象处理 IC、模糊逻辑 IC、通讯 IC 等等。

“电脑与通信”随着国内企业集团化的发展,政府决策等需要,电脑联网通信的问题已势在必行。为了促使本刊广大单机用户,开始尝试一下,点对点及网络通信、办公室自动化味道,我们将组织一些基础知识、模拟通信实验,模拟办公室自动化及网络实验方面的内容奉献给各位,使大家较快的进入网络之门。

信息、广告、经营和读者联谊等方面一直是本刊的弱点,在新的一年里我们将努力改进这方面的工作。使本刊不仅在技术业务方面给读者提供帮助,还要在信息咨询、开发产品,代购代销方面为本刊的单位(特别是生产单位)和个人订户提供合作和服务。

俗话说,“一个好汉三个帮”,要办好一个杂志,非常需要广大读者、作者和朋友的帮助,各们对明年我们的改进意见有什么建议,望多多赐教。谢谢。

(上接第3页)

主体式 CAI 是由 CAI 系统全部取代教师,由系统传授知识,提供资料,还可进行检索、改编、测验等。

辅助式 CAI 是由 CAI 系统部分地代替教师,主要用于帮助练习、复习、解题和提供辅导测验等。

利用 CAI 系统传授知识和提供训练一般也有两种模式,一种是由计算机逐个显示问题,由学生通过键盘输入答案,然后计算机来判断答案是否正确。如果正确,则显示下一个问题;如果错了,则给予适当的提示并再给一次回答机会。这样,通过不断地提出问题和增加问题的难度来帮助学生巩固知识和增强能力。另一种是把教学内容适当地分成一些教学单元并排出顺序,当学习开始时,计算机首先向学生显示各教学单元的目录,由学生自己来选择学习内容。内容选定之后,计算机不断地显示教学内容并配以提问,帮助学生学习。如果学生感到学习有困难,计算机可提供适当的提示,学生也可重新选择适合于自己情况的教学单元学习。如果学生顺利通过本教学单元,则可跳到高一级的水平。

CAI 可以模拟物理、化学及自然界各种动态变化的现象,它以计算机屏幕作为直观的教具,显示的图像具有直观性、动态感强、速度快、色彩丰富并还可配有音响效果。使得某些以往只能用语言来间接描述的微观的、宏观的、瞬时的等现象得到准确、直观的表达。CAI 还可提供汽车、轮船、飞机的驾驶和各种兵器操纵

的训练,CAI 提供这些训练既安全、逼真,又大量节省资金,现已被很多国家广泛应用。

CAI 与传统教学模式相比较,最突出的特点是真正实现了以学生为中心的人格化教学方法,由学生按照自己的实际情况来选择学习内容,控制学习进度,及时了解自己的学习效果。同时,由于计算机在教学过程中表现出的无比“耐心”所提供的安祥和諧的学习气氛,使得学生消除了诸方面的畏惧心理,从而能使学习的成功性大大提高。

当然,CAI 作为一种新兴的教育技术,还有不完善之处。在我国,除教育指导思想,资金设备等问题外,在技术上还存在着自然语言和图象的输入输出,人工智能以及开发工具落后等突出问题。但是,CAI 所体现的全新的教育思想和方法的意义和价值,却是不可否认的。美国加州大学欧文分校的布克教授在《Personal Computers for Eaducation》一书中这样写道:“再过二十五年,计算机将成为教育领域中占主导地位的传播媒质。在大多数学科中,越来越多的不同年龄人,从计算机上学到的东西将比从教科书上、课堂上或用其它方式学到的东西要多。”美国教育学家西摩·佩珀则预言:“到本世纪末,我们现在所熟知的教室将不复存在。”到那时,孩子们象现在使用铅笔和书本一样摆弄计算机。国际上最具权威性的信息技术专家们对 53 项计算机应用课题的发展前途进行名次评选时,将计算机教育排在第六位。



学装微电脑

微电脑控制微型钻床

(学习控制 X-Y 工作台方法)

易齐干

微型钻床大致分为: X 轴部件、Y 轴部件、工作台部件和钻头部件。

X 轴部件和 Y 轴部件的基体为有机玻璃。微型 DC 电机为驱动源, 经过齿轮减速带动丝杠, 拖动滑台移动。滑台运动随 DC 电机正、反转, 有正向、反向移动。

检测电机转速使用编码器, 电机转一周, 接收 4 个开闭信号, 发出 4 个脉冲。

被钻削的工件定位在工作台上。工作台部件完全由有机玻璃制成。

钻头部件基体为有机玻璃, 其上安装 DC 电磁铁和弹簧, 控制钻头上下运动。钻头的旋转由微型 DC 电机驱动。

电路原理图如图 1 所示。图中 8255 为

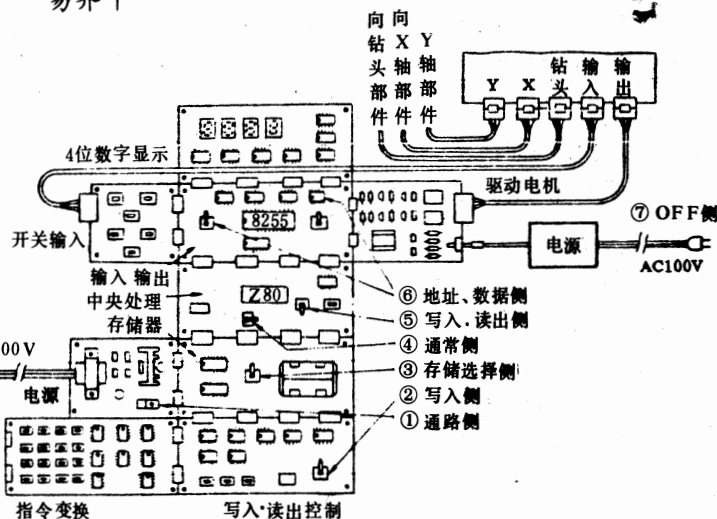


图 2 μP-80 套件与钻床连接
驱动电机 IC 型号为 BA6109, CPU 的控制信号

经过放大, 再分别拖动 X 轴、Y 轴电机。BA6109 能实现电机正、反转和刹车, 这样可大大简化结构和电路。

CPU 发出的控制钻头的信号, 经过晶体管放大, 通过继电器, 控制钻头电机和电磁铁。所需电压是 DC12V, 所以要再准备一个电源。

X 轴、Y 轴电机上的编码器检测出的脉冲信号输入给 8255 端口 A 的第 6 位、第 7 位。

Y 轴部件向前、向后运动, X 轴部件向左、向右运动; 自动钻孔开关均使用端口 A 的其余六位。

μP-80 微电脑套件与微型钻床的连接如图 2。

图 2 所示电路中, 端口 C 的 PC₄、PC₅ 分别输出 L 电平, 晶体管 (2SC1815) 导通, 继电器为通状态, 钻头电机和电磁铁工作。

PC₄、PC₅ 分别输出 H 电平, 晶体管、继电器均为断状态, 钻头电机和电磁铁也不工作。

驱动电机 IC (BA6109) 的输入输出引脚的电位与

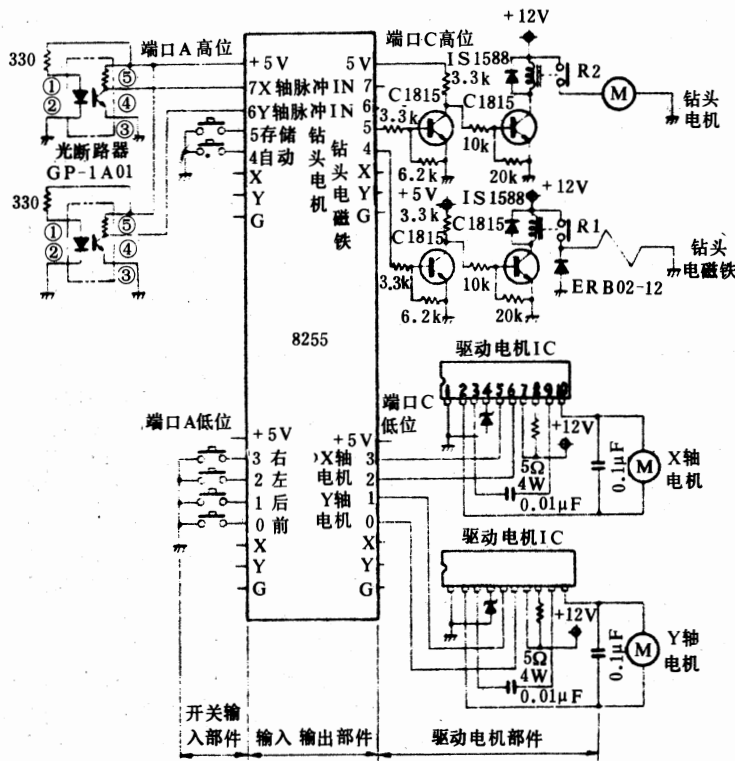


图 1

μP-80 套件的输入输出部件。

电机转向的关系如表 1 所示。

输入侧		输出侧		电机的转向
5" 出脚	6" 出脚	2" 出脚	10" 出脚	
1	1	0	0	停止
0	1	0	1	正转
1	0	1	0	反转
0	0	0	0	停止

表 1 驱动电机 IC 引脚电平关系

为便于设计软件,一般首先画出输入输出分配简图。如图 3 所示,一目了然,很清楚地展示输入输出的分配。

软件流程图如图 4 所示。

CPU 内部寄存器的分配如表 2 所示。

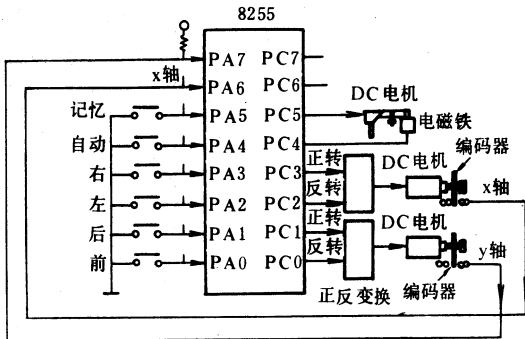


图 3 输入输出框图

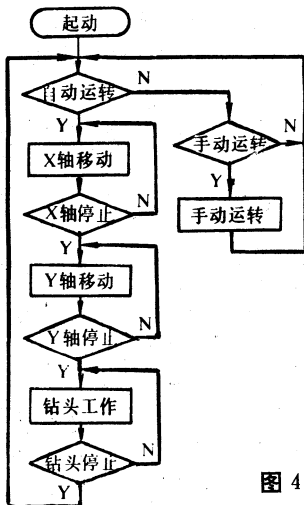


图 4 软件流程图

微型钻床 X 轴、Y 轴丝杠拖动工作台移动 1mm, 需要 100 脉冲(DC 电机一转有 4 个脉冲,电机转 25 转才移动 1mm)。

如果 A 点至 B 点为 10mm,则需要 1000 个脉冲。变换为 16 进制数:

$$1000 \div 256 = 3 \text{ 余 } 232$$

$$232 \div 16 = 14 \text{ 余 } 8$$

则为 3E8H。

寄存器名	任 务
A 累加器	与各端口进行数据交换
B 寄存器	暂时存储钻孔位置数据 7 6 5 4 3 2 1 0 0: 正转 1: 反转
C 寄存器	暂时寄存延时数据
D 寄存器	暂时寄存延时数据 7 6 5 4 3 2 1 0 电机没有脉冲输入 电机有脉冲输入
E 寄存器	暂时寄存延时数据
H、L 寄存器	以寄存器对表示寄存器地址

表 2 寄存器任务分配

X 轴电机、Y 轴电机正转、反转的标志位由 B 寄存器的第 7 位决定。

钻孔结束数据为 FFH(参照表 3)

按上述规定进行软件设计的示例如表 4。

地址	数据	
00DC	7 6 5 4 3 2 1 0	X轴沿 方向移动 mm
00DE		
00DF		Y轴沿 方向移动 mm
00E0		
00E1		X轴沿 方向移动 mm
00E2		
00E3		Y轴沿 方向移动 mm
00F0		
00F1		X轴沿 方向移动 mm
00F2		
00F3		Y轴沿 方向移动 mm
00F4	F F	

存入钻孔结束数据 FFH

表 3 内存存储方法

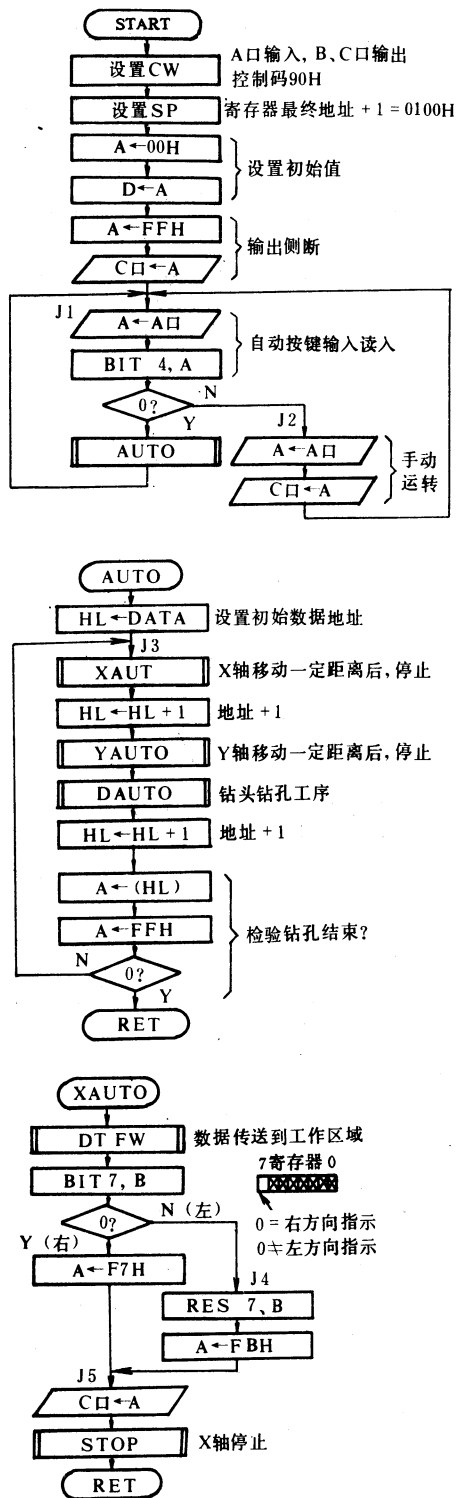
表中从 00H 地址到 DBH 地址是基本程序。DCH 地址到 F3H 地址写入孔位数据。钻孔结束数据 FFH 放入 F4H 地址。F5H 地址以后为堆栈。

照图 2 连接硬件,将程序写入、检验无差错,则启动中央处理部件的复位开关,执行程序。按动标注有“前”、“后”、“左”、“右”字样的按键。手动控制调整工件与钻头相对位置进行钻孔。然后,再按动标有“自动”的按键,开始按预定程序自动进行钻孔。

微型钻床为理想的教学装置。有订购者请与天津纺织工学院机械系 高殿斌同志联系。邮码:300160 电话:412833 转 983

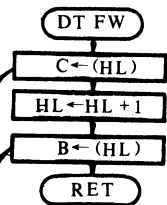
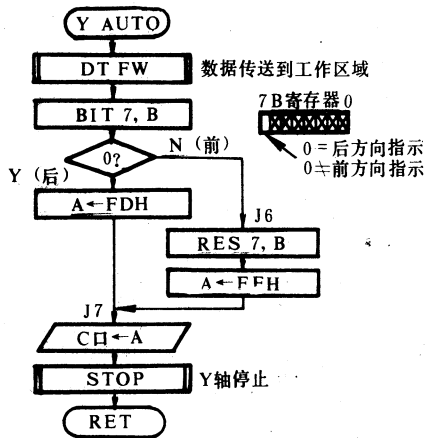
表 4 微型钻床程序示例(1、2、3)

表 4(1)



标号	助记符	地址	机器语	注释
START	LD A190H	0000	3E 90	设置 CW
	OUT(03H), A	0002	D3 03	设置 SP
	LD SP, 0100H	0004	31 00 01	设置初始值
	XOR A	0007	AF	输出全部为断状态
	LD D, A	0008	57	读入自动按键输入
	LD A, FFH	0009	3E FF	输出全部为断状态
	OUT(02H), A	000B	D3 02	读入自动按键输入
J1	IN A, (00H)	000D	DB 00	读入自动按键输入
	BIT 4, A	000F	CB 67	
	JP NZ, J2	0011	C2 1A 00	
	CALL AUTO	0014	CD 21 00	
	JP J1	0017	C3 0D 00	
J2	IN A, (00H)	001A	DB 00	手动运转
	OUT(02H), A	001C	D3 02	
	JP J1	001E	C3 0D 00	
AUTO	LD HL, 00DCH	0021	21 DC 00	设置初始数据地址
J3	CALL XAUTO	0024	CD 36 00	驱动 X 轴
	INC HL	0027	23	地址 + 1
	CALL YAUTO	0028	CD 4D 00	驱动 Y 轴
		002B	CD AF 00	钻头旋转
	INC HL	002E	23	地址 + 1
	LD A, (HL)	002F	7E	
	CP FFH	0030	FE FF	检验钻孔结束
	JP NZ, J3	0032	C2 24 00	
	RET	0035	C9	
XAUTO	CALL DT FW	0036	CD 64 00	数据传送到工作区域
	BIT 7, B	0039	CB 78	检验正转、反转
	JP NZ, J4	003B	C2 46 00	
	LD A, F7H	003E	3E F7	指示 X 轴右向
J5	OUT(02H), A	0040	D3 02	
	CALL STOP	0042	CD 68 00	
	RET	0045	C9	
J4	RES 7, B	0046	CB B8	指示 X 轴左向
	LD A, FBH	0048	3E FB	
	JP J5	004A	C3 40 00	
YAUTO	CALL DT FW	004D	CD 64 00	数据传送到工作区域
	BIT 7, B	0050	CB 78	检验正转、反转
	JP NZ, J6	0052	C2 5D 00	
	LD A, FDH	0055	3E FD	指示 Y 轴向后
J7	OUT(02H), A	0057	D3 02	
	CALL STOP	0059	CD 68 00	
	RET	005C	C9	
J6	RES 7, B	005D	CB B8	
	LD A, FEH	005F	3E FE	指示 Y 轴向前
	JP J7	0061	C3 57 00	
DT FW	LD C, (HL)	0064	4E	传送数据
	INC HL	0065	23	地址 + 1
	LD B, (HL)	0066	46	传送数据
	RET	0067	C9	
STOP	LD E, A	0068	5F	
J8	IN A, (00H)	0069	DB 00	检验 X 轴、Y 轴
	BIT 0, E	006B	CB 43	
	JP NZ, J9	006D	C2 7A 00	
J11	BIT 7, A	0070	CB 7F	Y 轴脉冲输入有效
J12	JP NZ, J1	0072	C2 87 00	
	RES 0, D	0075	CB 82	
	JP J8	0077	C3 69 00	
J9	BIT 1, E	007A	CB 4B	
	JP NZ, J10	007C	C2 82 00	
	JP J11	007F	C3 70 00	
J10	BIT 6, A	0082	CB 77	X 轴脉冲输入有效
	JP J12	0084	C3 72 00	
J13	BIT 0, D	0087	CB 42	
	JP NZ, J8	0089	C2 69 00	
	SET 0, D	008C	CB C2	

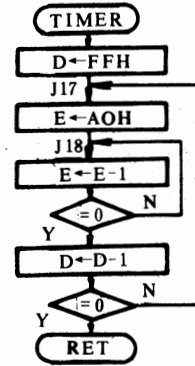
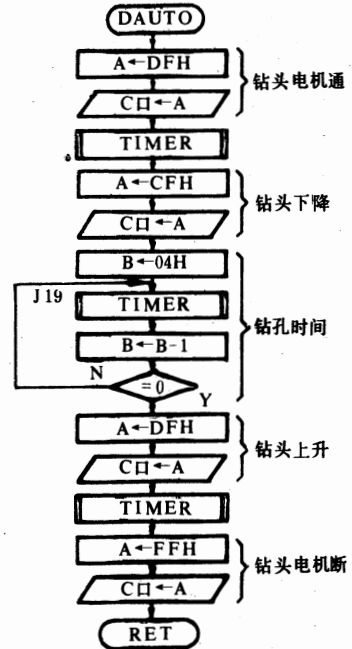
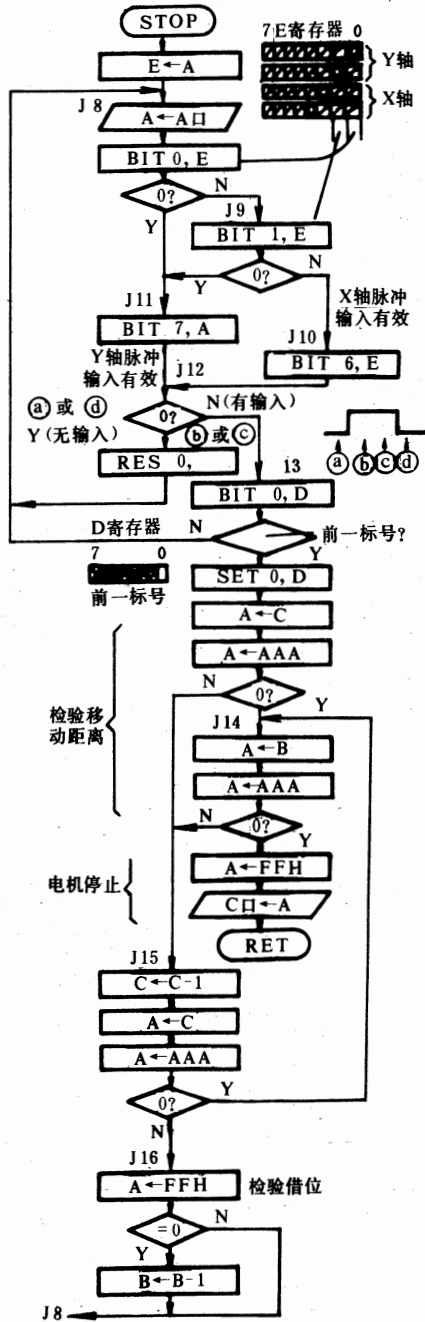
表 4(2)



地址	7	数据	0	
00DC				第7位设置为左右方向
00DD				X轴移动距离数据
00DE				第7位设置为前后方向
00DF				Y轴移动距离数据
00D0				
00D1				
00F2				
00F3				

标号	助记符	地址	机器语	注释
	LD A, C	008E	79	
	AND A	008F	A7	
	JP NZ, J15	0090	C2 9D 00	检验移动距离
J14	LD A, B	0093	78	
	AND A	0094	A7	
	JP NE, J15	0095	C2 9D 00	
	LD A, FFH	0098	3E EF	驱动电机停止
	OUT(02H), A	009A	D3 02	
	RET	009C	C9	
J15	DEC C	009D	0D	
	LD A, C	009E	79	
	AND A	009F	A7	
	JP NZ, J16	00A0	C2 A6 00	移动距离运算
	JP J14	00A3	C3 93 00	
J16	CP FFH	00A6	FE FF	
	JP NZ, J8	00A8	C2 69 00	检验借位
	DEC B	00AB	05	
	JP J8	00AC	C3 69 00	
DAUTO	LD A, DFH	00AF	3E DF	钻头电机接通
	OUT(02H), A	00B1	D3 02	
	CALL TIMER	00B3	CD CF 00	
	LD A, CFH	00B6	3E CF	钻头下降
	OUT(02H), A	00B8	D3 02	
	LD B, 04H	00BA	06 04	
J19	CALL TIMER	00BC	CD CF 00	钻孔时间
	DEC B	00BF	05	
	JP NZ, J19	00C0	C2 BC 00	
	LD A, DFH	00C3	3E DF	钻头上升
	OUT(02H), A	00C5	D3 02	
	CALL TIMER	00C7	CD CF 00	
	LD A, FFH	00CA	3E FF	钻头电机断
	OUT(02), A	00CC	D3 02	
	RET	00CE	C9	
TIMER	LD D, FFH	00CF	16 FF	
J17	LD E, AOH	00D1	1E A0	
J18	DEC E	00D3	1D	
	JP NZ, J18	00D4	C2 D3 00	
	DEC D	00D7	15	
	JP NZ, J17	00D8	C2 D1 00	
	RET	00DB	C9	
DATA		00DC	00	X1 的数据
		00DD	00	
		00DE	00	Y1 的数据
		00DF	00	
		00E0	00	X2 的数据
		00E1	00	
		00E2	90	Y2 的数据
		00E3	81	
		00E4	F4	X3 的数据
		00E5	81	
		00E6	00	Y3 的数据
		00E7	00	
		00E8	F4	X4 的数据
		00E9	81	
		00EA	00	Y4 的数据
		00EB	00	
		00EC	00	X5 的数据
		00ED	00	
		00EE	90	Y5 的数据
		00EF	01	
		00F0	F4	X6 的数据
		00F1	01	
		00F2	00	Y6 的数据
		00F3	00	
		00F4	FF	钻孔结束

表 4(3)





计算机语音输出功能的开发与应用(下)

马钢南山铁矿技术科(243033) 陈竹林

五、语音数据采集程序

由于语音数据的采集与处理速度很快(大于每秒 8000 次),采集程序必须用汇编语言编制。该程序的文件名是“YYCJ.BASM”,图 4 是该程序的工作流程图。实际应用时,可将其作为 BASIC 语言的一个内存映象文件,由 BASIC 程序调用执行,在运行过程中有任何击键动作都将停止采集返回主程序。它有三个入口参数,A%是延时值,可控制采样的速率;B%是数据区段地址,指示语音数据在内存中存放的起始位置;C%是已采语音数据的字节总数。其

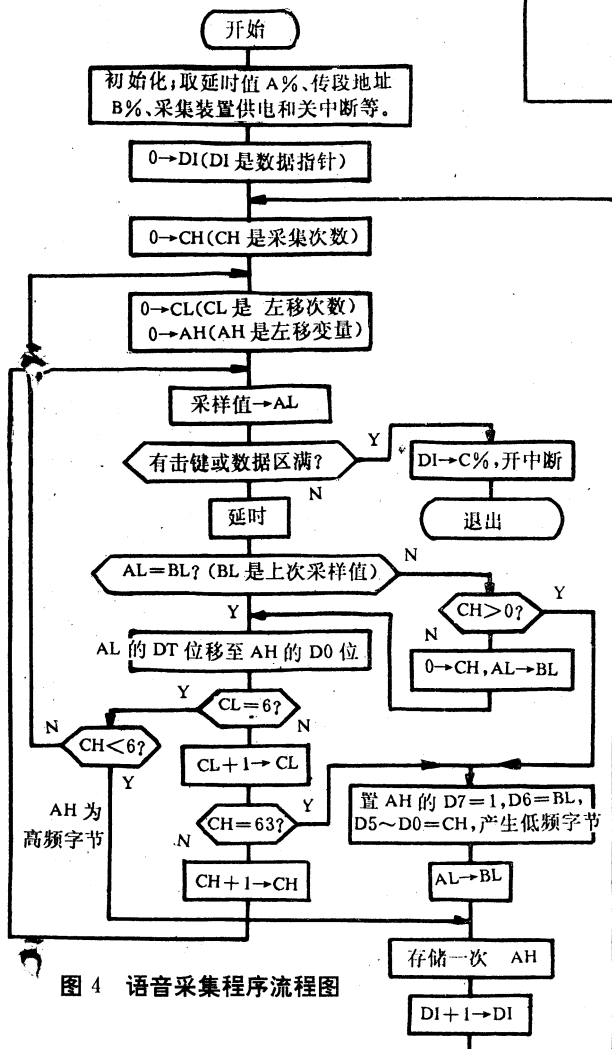


图 4 语音采集程序流程图

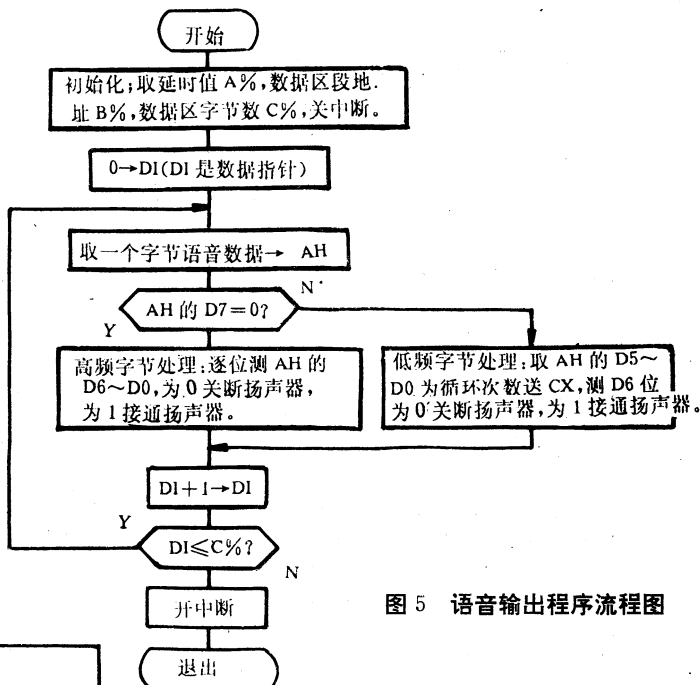


图 5 语音输出程序流程图

中 A%是由主程序传给采集程序的,而 B%和 C%是采集完成后由采集程序传给 BASIC 主程序的。

六、语音输出程序

语音输出程序的作用,是将内存中经压缩的语音数据还原后驱动扬声器,完成语音输出的功能。图 5 是语音输出程序的工作流程,其入口参数 A%、B%和 C%都是由 BASIC 主程序传送的。其中 A%是放音延时值,可控制放音速度的快慢;B%是语音数据区的段地址,可控制从内存中取语音数据的起始位置;C%是语音数据区的长度,可控制每次放音取语音数据的字节数。

七、语音数据库建立程序

在一个应用程序中需要语音输出单字或字串是有限的,将这些单字或字串的语音数据按一定的顺序存放在一个文件中,这个文件就是语音数据库。为了能快速找到每个单字或字串在语音数据库中的起始位置,还要建立一个语音数据索引文件。程序二是语音数据库及索引文件建立程序。其中语音数据库文件名是“YYSJ.DAT”,其索引文件名是“YYSJ.IND”、“YYCJ.BAM”和“YYSG.BAM”分别是语音采集与语音输出的 BASIC 内存映象汇编程序,“DZSJ.DAT”是单字或字串语音数据暂存文件。

程序二:

```

0 '语音数据库建立(JLYK.BAS)
10 '单字或字串输入与语音数据采集
20 CLS;DIM A$(100);I=1;L=1
25 PRINT"请逐一输入需要发声的单字或字串,输入
    END 结束"
30 PRINT STR$(I);": ";H=CSRLIN;L=L+16;IF
    L>70 THEN L=1;H=H+1
40 INPUT ":",T$;IF T$="END" OR T$="end"
    THEN ZS=I-1;GOTO 60
50 LOCATE H,L:A$(I)=" "+T$;I=I+1;GOTO 30
60 DEF SEG=&H7000;BLOAD"YYCJ.BAM",0
70 CLS;PRINT"对着话筒朗读下列单字或字串,字与字
    之间停顿1秒钟左右;"
80 FOR I=1 TO ZS;PRINT STR$(I);": ";MID$(A
    $(I),5),.NEXT I
85 PRINT;PRINT;INPUT "准备好按回车开始!按空格
    结束";T$
90 PRINT"开始朗读!";A%=5;CJ%=0;CALL
    CJ%(A%,B%,C%);T$=INPUT$(1)
100 '数据分隔处理
105 PRINT"现在作字分隔处理,请稍候"
110 DEF SEG=B%;J=0;FOR I=1 TO ZS
115 PRINT I;": ";MID$(A$(I),5),
120 IF PEEK(J)=&HBF THEN K=0
130 K=K+1;J=J+1;IF K>10 THEN SW=J;MID$
    (A$(I),1)=MKI$(SW) ELSE 120
140 IF PEEK(J)<>&HBF THEN K=0
150 K=K+1;J=J+1
160 IF K>5 THEN CD=J-SW;MID$(A$(I),3)=
    MKI$(CD) ELSE 140
170 NEXT I;PRINT
200 '语音核对
210 DEF SEG=&H7000;BLOAD"YYSC.BAM",0
215 A%=9;SC%=0;DZ%=B%
220 FOR I=1 TO ZS;T$=A$(I);PRINT
    STR$(I);": ";MID$(T$,5),
230 SW=CVI(MID$(T$,1,2));CD=CVI
    (MID$(T$,3,2))
240 B%=DZ%+SW/16;C%=CD;CALL SC%
    (A%,B%,C%)
250 NEXT I;PRINT;PRINT"1:存盘 2:重读 3:再听"
260 INPUT T$;IF T$="1" THEN 300
270 IF T$="2" THEN 70
280 IF T$="3" THEN 220 ELSE BEEP;GOTO 260
300 '数据整理存盘
310 DW=0;FOR I=1 TO ZS;T$=A$(I)
320 SW=CVI(MID$(T$,1,2));CD=CVI
    (MID$(T$,3,2))
330 DEF SEG=DZ%;BSAVE"DZSJ.DAT",SW,CD
335 T%=INT(DW/16);IF T%<>DW/16
    THEN T%=T%+1
340 DEF SEG=DZ%+T%;BLOAD"DZSJ.DAT",0
350 MID$(A$(I),1)=MKI$(T%);DW=16*T%+
    CD
355 NEXT I;KILL"DZSJ.DAT"

```

```

360 DEF SEG=DZ%;BSAVE"YYSJ.DAT",0,DW
370 OPEN"YYSJ.IND"AS#1LEN=12;FIELD
    #1,12ASN$
380 FOR I=1 TO ZS;LSETN$=A$(I);PUT
    #1,I;NEXT I
390 CLOSE #1
990 END

```

程序 20~50 行完成单字或字串的输入,根据提示逐一从键盘输入待处理的单字或字串,并依次赋给 A\$()数组,当输入的字串是“END”时结束输入。60~90 行完成语音采集,屏幕首先显示刚输入的全部单字或字串,提示使用者对着话筒逐一朗读,全部朗读完毕后击空格键结束采集。为便于程序对语音数据作分隔处理,要求字与字之间停顿 1 秒钟左右。语音数据在内存中的段地址在 B%中,总字节数在 C%中。第 100~170 行完成语音数据的分隔,逐一找出每个单字或字串的起始地址和字节数,并记录在对应 A\$()变量前面。第 200~280 行完成语音核对,逐一在屏幕上显示单字或字串,同时通过扬声器输出其读音,供使用者监听核对分隔的正确与否。第 300~390 行完成数据的紧缩与存盘,先去掉字间停顿期间采集到的大量 BFH 数据,然后将紧缩后的语音数据以“YYSJ.DAT”文件存盘,将记载单字或字串起始位置和字节长度信息的 A\$()数组以“YYSJ.IND”文件存盘。到此为止,语音数据库及索引文件的建立就完成了。

八、一个应用语音输出功能的例子

为了进一步说明语音输出功能的开发过程。现在举一个在 IBM-PC/XT 计算机上实现语音报时功能的实际应用例子。开发过程如下。

第一步:将语音采集装置(即话筒电路)联接在计算机的 RS-232C 接口上,启动计算机进入 CCDOS 汉字操作系统。

第二步:调入解释 BASIC 系统,并运行语音数据库建立程序“JLYK.BAS”。根据提示从键盘上依次输入 14 个单字“1、2、3、4、5、6、7、8、9、0、拾、点、整、分”和 1 个字串“现在是”,输一个按一次回车键,最后键入“END”结束输入。在屏幕重新显示这些单字或字串并提示“准备好按回车开始!按空格结束”后,手握话筒,口正对着受话面,按一下回车键后大声朗读这些字。注意单字之间停顿 1 秒左右,而字串“现在是”字间不应停顿。全部朗读完毕,按空格键结束采集。

计算机经过一段时间的分隔处理后,在屏幕上逐一显示这些字,并驱动扬声器输出每个单字和字串的读音。此后,屏幕提示“1:存盘 2:重读 3:再听”,如果字音关系正确、听音效果满意,键入“1”,计算机对语音数据作紧缩处理后存盘,生成语音数据库“YYSJ.DAT”和索引文件“YYSJ.IND”。如果读音不正确或不满意,键入“2”回到前面重新朗读。键入“3”可再听一遍。

第三步:调入并运行计算机报时程序“JQBS.BAS”,程序三是该程序的语句清单。

程序三:

```

10 '计算机报时(JQBS.BAS)
20 DIM A$(200);KEY OFF;ZS=15
25 DEF SEG=&H7000;BLOAD "YYSC.BAM",0
35 DZ%=&H7010;DEF SEG=DZ%;BLOAD "YYSJ.-
  DAT",0
40 OPEN "YYSJ.IND" AS #1 LEN=12:FIELD #1,12
  AS N$
45 FOR I=1 TO ZS:GET #1,I:A$(I)=N$:NEXT I
50 SCREEN 2:SCREEN 1:COLOR 9,0
55 LOCATE 2,10:PRINT "计算机语音报时演示软件"
60 LOCATE 6,15:PRINT TIME$
65 IF INKEY$="" THEN 60
70 GOSUB 100:GOTO 60
100 '语音字符串形成
110 T$=TIME$:A$=MID$(T$,1,1):B$=MID$
  $(T$,2,1)
120 C$=MID$(T$,4,1):D$=MID$(T$,5,1):
  Y$=""
130 IF A$="0" THEN Y$=B$+"点,"
135 IF A$="1" AND B$="0" THEN Y$="
  拾,点,"
140 IF A$="1" AND B$>"0" THEN Y$="
  拾,"+B$+"点,"
145 IF A$>"1" AND B$="0" THEN Y$="
  A$+"拾,点,"
150 IF A$>"1" AND B$>"0" THEN Y$="
  A$+"拾,"+B$+"点,"
155 IF C$="0" AND D$="0" THEN Y$="
  Y$+"整":GOTO 180
160 IF C$>"0" AND D$="0" THEN Y$="
  Y$+C$+"拾,分":GOTO 180
165 IF C$="0" AND D$>"0" THEN Y$="
  Y$+"0,"+D$+"分":GOTO 180
170 Y$=Y$+C$+"拾,"+D$+"分"
180 Y$="现在是,"+Y$:GOSUB 200
190 RETURN
200 '字符串语音输出
210 A$=Y$
220 V=INSTR(A$,".");IF V=0 THEN T$="
  A$:GOTO 230
225 T$=MID$(A$,1,V-1):A$=MID$(A$,V+
  1)
230 FOR I=1 TO ZS:E$=A$(I):F$=MID$(E$,
  5)
240 S=INSTR(F$,T$):IF S=0 THEN 270
250 T=CVI(MID$(E$,1,2)):CD=CVI(MID$
  (E$,3,2))
260 A%=9:B%=DZ%+T:C%=CD:SC%=0:DEF
  SEG=&H7000:CALL SC%(A%,B%,C%)
270 NEXT I:IF V>0 THEN 220
290 RETURN

```

运行该程序,在屏幕中间显示当前的时间,任何时候敲任何键,计算机都用语音报告一次当前的时间。如“现在是八点二十五分”、“现在是九点整”等。

九.几点说明

1. 在早期的 XT 计算机上使用时,如感到扬声器的音量太小,可按图 6 在 RS-232 口上外接一个扬声器,使输出音量大大增加。图中变压器可选用普通收音机中的输出变压器。此时应将语音采集“YYCJ.BAM”和语音输出“YYSC.BAM”汇编程序中控制扬声器通、断电的程序行换成下列程序行:

通电:MOV DX,3FCH

MOV AL,01H

OUT DX,AL

断电:MOV DX,3FCH

MOV AL,02H

OUT DX,AL

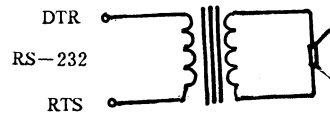


图 6 外接扬声器

2. 由于本方案在语音输出时只能再现语音的音调特征,而对声色和响度无法再现,故存在一定的失真。因此在语音采集过程中,朗读要尽可能清晰,最好请音调较高的女士朗读。

3. 由于不同档次的计算机运行速度不同,对延时值 A% 的选择也不尽相同。XT 计算机速度较低, A% 应小些(一般取 A%=1~5 较为合适),在 286、386 等速度较高的计算机上 A% 可选大一些。A% 越小采集或放音速度越快,音质越好,但语音数据量越大,占用内存或磁盘的空间也越大,要权衡考虑。

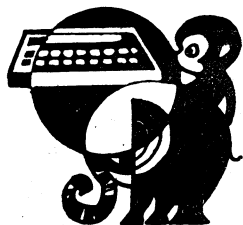
(接自 45 页)

```

529 PRINT "Φ1 故障! 转时钟测试模块":GOTO 585
520 A$="脚脉冲 P,高 H 或低 L 电平:"
523 PRINT"G1-6":A$;:INPUT B$
525 IF B$="L" THEN PRINT"G1 坏!":GOTO 585
530 PRINT "请测 F12 以下引脚:"
537 R=192
540 FOR I=15 TO 9 STEP-1
545 POKE D1,R
550 IF M=2 AND(I=12 OR I=10)THEN 560
555 PRINT I:A$;:INPUT B$
557 IF B$<>"P"THEN 580
560 R=R+8:NEXT I
565 IF M=2 THEN 585
570 POKE D1,R:PRINT 7:A$;:INPUT B$
575 IF B$="P"THEN 585
580 PRINT "F12 译码器坏!"
585 INPUT "请记录后回车返回!":A$
590 RETURN

```

本系统由于通过仿真接口卡代替故障机 CPU 进行读写控制,其最大的优点是能产生静态或动态的地址和数据,对于逻辑电路的测试极为方便。



电脑游戏机

F BASIC 语言的游戏程序编程技巧

第三讲 程序结构和程序框图

山东苍山机械电子化学工业局(277700)于春

一、程序结构

程序结构一般分为三大类：

- 模块结构
- 顺序结构
- 混合结构

1. 模块结构也叫结构化程序。它根据总体需要和系统配置,把一个程序分成具有某些特定功能的程序段,每个程序段形成一个子程序,称为功能模块。每个模块相对独立地完成一定的任务。它可以是计算分析战斗场面;可以是游戏主人翁的跳跃飞腾;也可以是一段音乐伴奏等等。然后再加一个调用子程序的主程序——称为管理程序。管理程序在需要时可调用所有模块。另外,各模块之间也可以互相调用。不难看出,一个完整的程序就是由管理程序和若干模块组成的。这种结构的优点是:(1)由于模块之间相对独立,所以在开发程序时,可以分头编写。程序的调试纠错较容易,因而编程周期短。(2)由于模块的相对独立性,能有效地防止错误在程序中的扩散与蔓延,因而提高了系统的可靠性和质量。(3)由于各模块功能明确、相对独立,通用性强,因此便于其它程序的借用和向其它语言移植。缺点是:编程时要加一定的限制;调用时要考虑有关变量的赋值;调用后要对运算结果进行恰当的处理等。因此,结构化程序的程序量较大,一般在大型、复杂程序中使用。

2. 顺序结构也叫非结构化程序,它与结构化程序相反,采用顺序编写的方法。程序中的转向,一般使用 GOTO 语句实现,条件转移语句用得较多。这种结构的优点是:(1)程序顺序编出,变量之间不易混淆。(2)省去了程序的链接调用操作,从而程序量较小。(3)由于 GOTO、IF—THEN 语句使用较多,程序不易阅读,从而使程序的保密性加强。缺点是:只能一个人编写程序,编程周期长;程序的调试改错较困难,一处有错殃及整个程序不能运行;由于出现 GOTO 网络,不易阅读理解,程序的借用移植性能较差。这种结构一般在简单程序中使用。

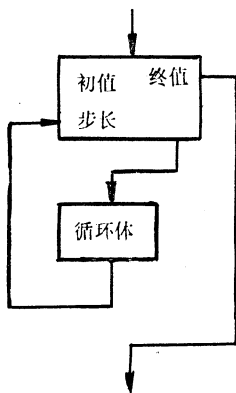
3. 混合结构则综合了以上两种结构的优点;它根据游戏的需要,对模块化结构程序进行了灵活处理,把那些整个游戏中只调用一次的模块则取消模块格式,直接放入主程序中。对调用最频繁的模块,则尽量安置在行号较小的位置,甚至移到主程序之前,以提高程序的执行速度。(由于电脑在执行 GOTO、GOSUB 语句时,总是从最小行号开始搜索,直到目标行号。)今后我们介绍的程序多是这种结构的程序。但是为使层次清楚,便于读者阅读理解,开始仍以模块化结构为主。

二、程序框图

什么叫程序框图呢?我们说,程序框图是一种流程图,它是用各种几何图形及文字说明来直观地描述电脑计算执行过程的有向图。程序框图常用的符号如下:

- (1) ○ 椭圆形框:表示程序的开始或结束。
- (2) □ 矩形框:表示赋值或完成运算操作。
- (3) ◇ 菱形框:判断某个关系式是否成立。若关系式成立则程序向 Y(Yes)方向进行;否则向 N(No)方向进行。
- (4) ▱ 斜框:表示输入输出。
- (5) → :箭头:表示流程的流向。
- (6) @ 小圆圈内一个字母或数字:表示流程图的接点。有些流程图很长或很复杂,往往一页纸画不下,需转到下一页时,则把一些地方断开,在断开的两头画圈,圈内标同样的符号,表示这两点是连在一起的。

(7) FOR 循环框图:循环变量初值占矩形框的左上角;终值写在右上角;步长在左下角;从右下角引出进入循环体;从终值处转出 FOR 循环。



程序框图的符号还有一些,最常用的是这七种,其它符号在以后用到时再介绍。

一般程序框图分两大类:

- 粗结构框图
- 细结构框图

1. 粗结构框图也叫功能模块结构图。它描述了整个软件系统由哪些模块组成,也说明了模块之间的调用关系。

一般有图 1 的形式。图 1 中,方框表示模块,框内写有模块的名称及模块的主要功能;箭头表示上层模块对下层模块的调用;菱形符号表示有条件调用;

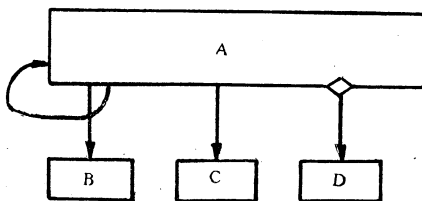


图 1

弧形箭头表示循环调用。图 1 说明 A 模块循环调用模块 B、直接调用模块 C、有条件地调用模块 D。

2. 细结构框图俗称流程图。它是对程序的精细描述。它给出模块功能的实现过程。它主要包括模块内部采用的计算方法、数据的存储结构和输入输出等方面的设计步骤。对于第二讲中介绍的例题三，我们可以画出功能模块结构图如图 2：

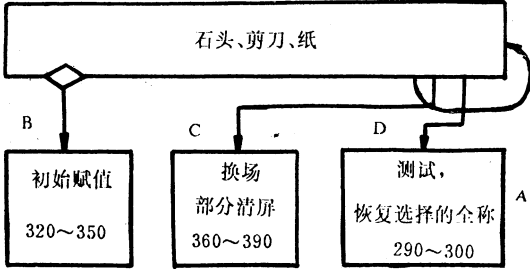


图 2、石头、剪刀、纸的模块结构图

例三是一个典型化的混合结构程序。A 为主控程序；B、D 为循环调用模块；C 为条件调用模块，仅在每次游戏开始时才调用。实际上初始赋值模块 C 也可以放到 A 程序的 55 行以后作为管理程序的一部分。

模块结构图与流程图有本质的区别。模块结构图反映了程序的层次特性，即某个模块要调用那些模块，哪些模块又调用什么模块。流程图则反映程序执行的过程特性，即先执行哪一部分，再执行哪一部分。两者是全局和局部的关系。

在一、二讲的三个示例中，介绍了游戏程序中加入音响的方法、游戏背景及卡通图案、色彩的变换技巧和键盘输入的处理。本讲再介绍如何使用操纵器控制卡通运动，卡通运动越界的判断处理及卡通随机自动连续运动的处理方法。

例四“企鹅看外婆”游戏

故事情节：“企鹅太太带着五个孩子去看外婆。去外婆家要穿过一条马路。马路上车辆奔流不息。她只能瞅准车辆运行空隙，一次抱一个小企鹅到马路对过，往返五次，把孩子们都送过马路后，才能去外婆家。”

根据故事情节，我们可以这样规划游戏结构。游戏画面是一条很宽的横向马路。马路上有七辆汽车在不同的位置，以不同的速度运行。在马路的中段画出一段较窄的纵向人行道。人行道的北头（屏幕上方）有五个小企鹅。企鹅太太在人行道的南端（屏幕下方）。选一号操纵器控制企鹅太太前后左右运动。当运行到北端时，抱起一个小企鹅，北端还剩四只。企鹅返回到南端后，则南端多出一只小企鹅。直到五个小企鹅全部转移到人行道南端则游戏结束。游戏过程中加入企鹅太太

太的脚步声、抱起企鹅和放下企鹅的声音、把企鹅安全转移到马路对过后的祝贺声和与汽车相撞的声音，共五种简单音响。为增加游戏难度，加入了时间限制。

根据游戏结构，首先划分功能模块。然后画出模块结构框图，见图 3。

各模块功能如下：

- 模块 A：管理程序。它完成游戏过程的控制。
- 模块 B：初始化模块。画出马路、人行横道。马路为横向，宽 12、长 27。人行道为纵向，宽 4、长 14。在人行道下端显示企鹅太太。上端显示五个小企鹅。七辆汽车从向右左行驶，车色、车速不同。画面如下：

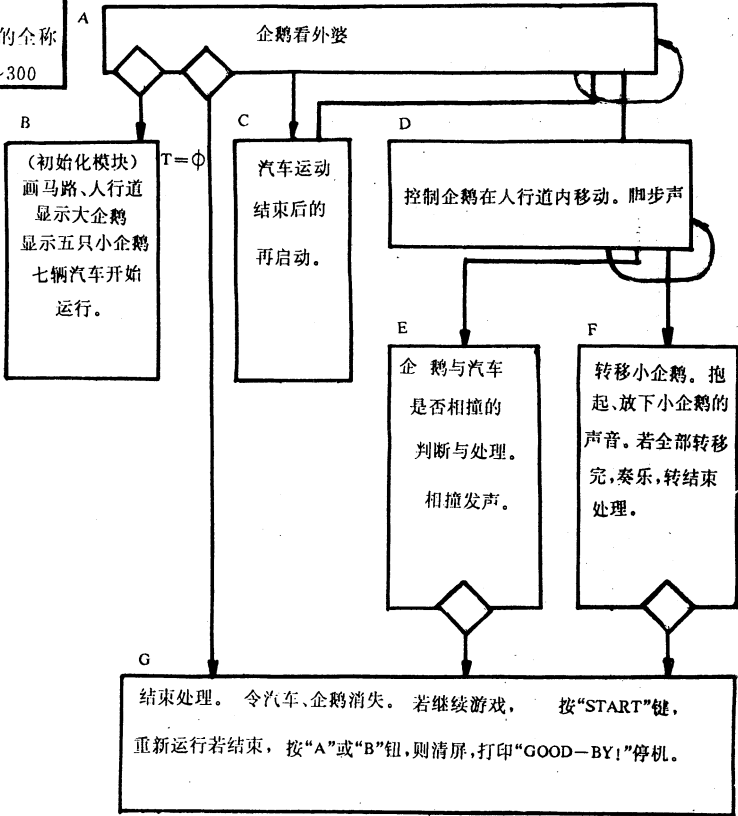


图 3、“企鹅看外婆”模块结构图

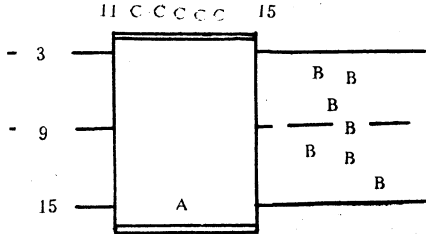


图 4、企鹅看外婆画面

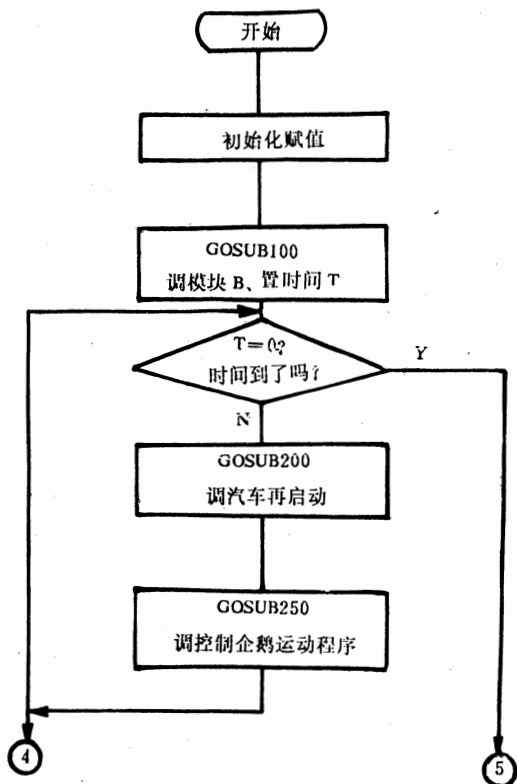


图 9

图 4 中, A 为企鹅太太; B 为汽车; C 为小企鹅。座标分别为: A (120, 144); B [250, RND (60) + 56]; C (104, 20) ~ (136 ~ 20)

- 模块 C: 汽车再启动。它随时检测七辆汽车的运行情况, 若有停止运行的汽车, 则立即启动, 令其运行。
- 模块 D: 用操纵器控制企鹅运动, 并随着企鹅的运动发出脚步声。它循环调用模块 E、F。
- 模块 E: 判断企鹅是否与汽车相撞。若相撞则发声, 转结束处理模块。
- 模块 F: 完成小企鹅的转移。抱起和放下小企鹅时伴有音响。若五个小企鹅全部转移则奏乐祝贺, 转结束处理。
- 模块 G: 结束处理模块。用“START”键控制继续游戏; 用 A、B 钮控制结束。结束时打印“GOOD—BY!”。停机。

对照功能模块, 可分别画出各模块的流程图(见图 5~图 9)。

根据程序框图可对对应编写出每个模块的详细程序, 然后把各模块的程序安排适当的行号, 合并后, 就得出“企鹅看外婆”游戏的完整程序。该程序较简单, 直接给出结构化程序清单(为便于阅读、理解, 程序清单

中个别部分加入了中文说明)。

“企鹅看外婆”结构化程序

```

5  REM "NO. 4 PENGUINS VISIT GRANDMOTHER"
10  CLS; CLEAR; SP. 0.; CG. RND(2), 0; T = 800
15  PL. "V15Y2T1; V15T1"
20  GOS. 100(画马路、显示汽车、企鹅)
25  T = T - 1; LOC. 22, 21; P. T; "□□"; IF T = 0 T. 60
30  GOS. 200(汽车的再启动)
40  GOS. 250(控制企鹅运动)
50  GOTO 25
60  ERA 0, 1, 2, 3, 4, 5, 6, 7; F. I = 1 TO 5; SP. I; N.
65  CLS; LOC. 0, 20; P. "Press START to continue"
70  S = STRIG(0); IF S = 0 G. 70
75  IF S = 1 T. RUN
80  CLS; LOC. 8, 8; P. "GOOD—BY!"; LOC. 0, 20
85  E.
100  REM "GAME Picture"(游戏画面)
110  F. I = 0 TO 27; IF I < 10 OR I > 16 T. LOC. I, 9; P. CH.
    (227)
120  P = 3; Q = 15; IF I >= 10 AND I <= 16 T. P = 1; Q = 17
130  LOC. I, P; P. CH. (197); LOC. I, Q; P. CH. (197)N.
140  F. I = 2 TO 16; LOC. 10, I; P. CH. (226); LOC. 16, I; P.
    CH. (226); N.
150  F. I = 1 TO 5; DE. SP. I, (RND(4), 0, 0, 0, 0) = "defg";
    SP. I, (I - 1) * 8 + 104, 20; N.
160  DE. M. (7) = SP. (4, 0, 1, 2, 0, 2); POS. 7, 120, 144; M. 7
170  F. I = 0 TO 6; DE. M. (I) = SP. (6, 7, I + 1, 130, 0, RND
    (4))
180  POS. I, 250, RND(60) + 56; M. I; N.
190  RE.
200  REM "Vehicles Restart"(汽车再启动)
210  F. I = 0 TO 6
220  IF M. (I) = 0 T. POS. I, 250, RND(60) + 56; M. I
230  N.
240  RE.
250  REM "Penguin Moves"(企鹅运动)
260  S = 0; SX = 0; SY = 0; K = STICK(0); IF K = 0 T. 380
270  IF K = 1 T. S = 3; SX = 1
280  IF K = 2 T. S = 7; SX = -1
290  IF K = 4 T. S = 5; SY = 2
300  IF K = 8 T. S = 1; SY = 1
310  PX = (XPOS(7) - 16) / 8; IF PX = 10 T. PX = 11
320  IF PX = 15 T. PX = 14
330  PY = (YPOS(7) - 24) / 8
340  S$ = SCR$(PX + SX, PY + SY)
350  IF S = 0 OR S$ <> " " T. 380
360  DE. M. (7) = SP. (4, S, 2, 4, 0, 2); POS. 7, PX * 8 + 16,
    PY * 8 + 24; M. 7
370  PL. "O1#F0R00#F,R"
380  GOS. 400(与汽车相撞的判断处理)
385  GOS. 450(转移小企鹅的处理)
390  RE.
400  REM "Collides with Vehicle"(碰汽车的处理)
410  IF CR. (7) >= 0 T. PL. "O1E0DCBAG; O0#A#G#FE
    #D#C"; GOTO 60(转结束处理)
  
```

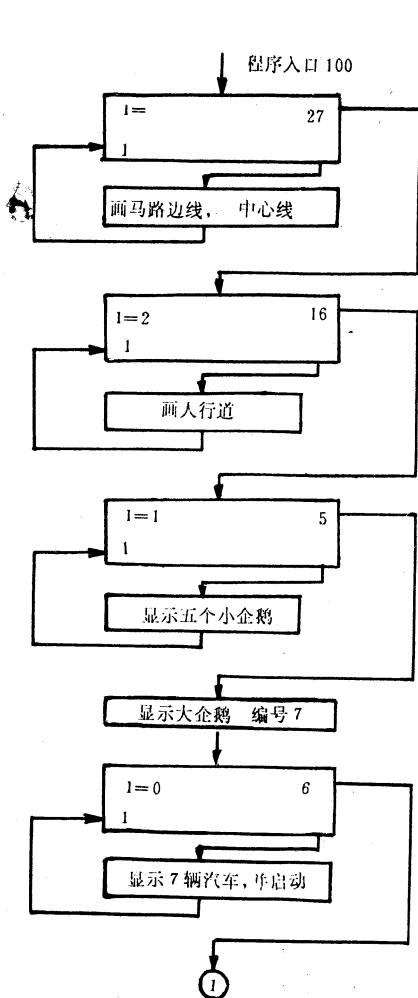


图 5

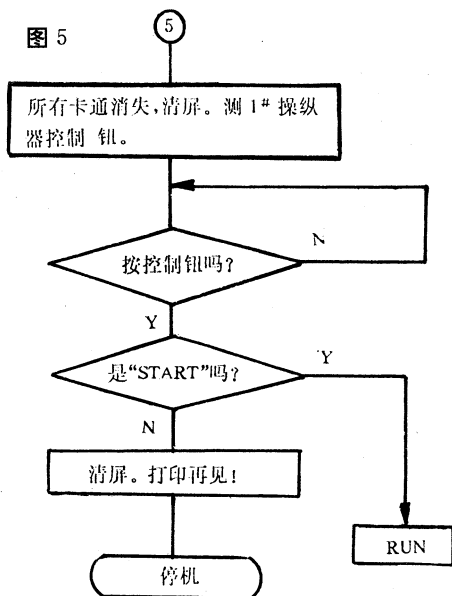


图 8

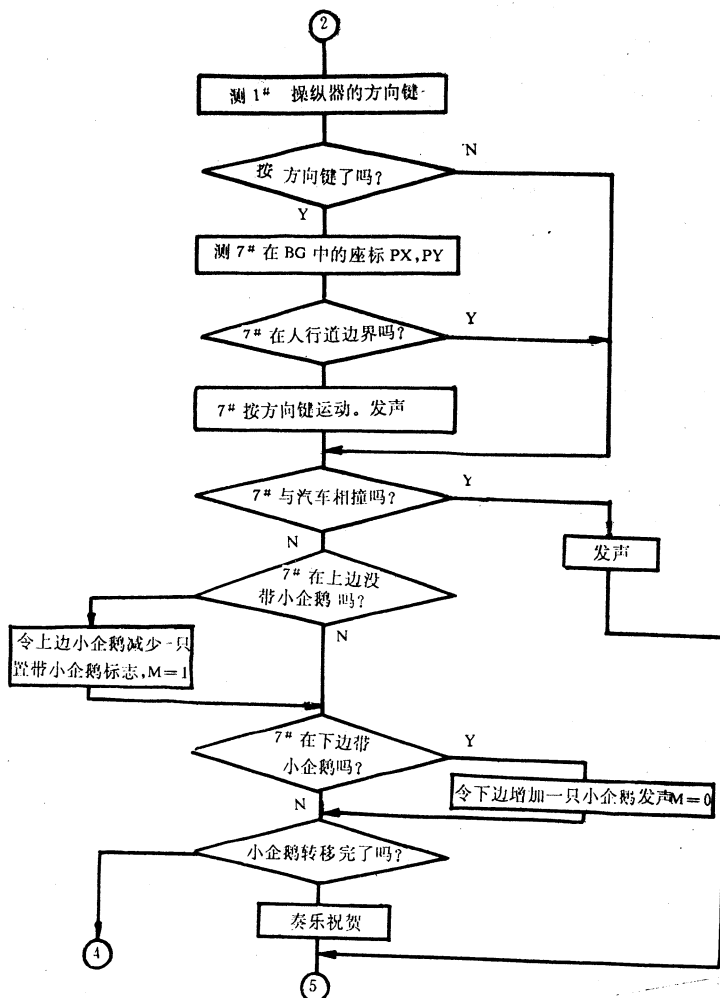


图 7

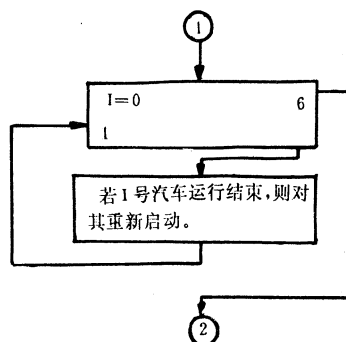


图 6

```

420 RE.
450 REM "Moves Small Penguins"(转移小企鹅)
460 IF PY=1 AND M=0 T. H=H+1;M=1;SP. H;N=1
    PL. "O5C0DEDE;O3C0DEF#G"
470 IF PY=15 AND N=1 T. M=0;SP. H,(H-1)* 8+
    104,152;N=0;PL. "O5#G0FEDC;O3#G0#FE#D#
    C"
480 IF H=5 AND N=0 T. PL. "O2G3GO3CCEECC;O1
    C3EGECGCC";PL. "GGAGEDC5O2C3O3C1;CEGECGC
    C1CC";G.60(转结束处理)
490 RE.

```

程序 NO. 4 中某些模块如 D、E、G 是通用模块,可作为工具使用。为便于读者对照,下面再给出“企鹅看外婆”游戏顺序结构的程序清单。

“企鹅看外婆”非结构化程序(顺序结构)

```

5 REM "NO. 5 PENGUINS VISIT GRANDMOTHER"
10 CLS;CLE.;SP. O.;T=800;CG. RND(2);0
20 PL. "V15T1;V15T1"
30 F. I=0 TO 27;P=3;Q=15
40 IF I<10 OR I>16 T. LOC. I,9;P. CH. (227)
50 IF I>=10 OR I<=16 T. P=1;Q=17
60 LOC. I,P;P. CH. (197);LOC. I,Q;P. CH(197);N.
70 F. I=2 TO 16;LOC. 10,I;P. CH. (226);LOC. 16,I;P.
    CH. (226);N.
80 F. I=1 TO 5;DE. SP. I,(RND(4),0,0,0,0)="defg";SP.
    I,(I-1)* 8+104,20;N.
90 DE. M. (7)=SP. (4,0,1,2,0,2,);POS. 7,120,144;M. 7
100 F. I=0 TO 6;DE. M. (I)=SP. (6,7,I+1,130,0,RND
    (4))
110 POS. I,250,RND(60)+56;M. I;N.
120 T=T-1;LOC. 22,21;P. T;" "
130 IF T=0 T. 340
140 F. I=0 TO 6
150 IF M. (I)=0 T. POS. I,250,RND(60)+56;M. I
160 N.
170 S=0;SX=0;SY=0;K=STICK(80);IF K=0 T. 280
180 IF K=1 T. S=3;SX=1
190 IF K=2 T. S=7;SX=-1
200 IF K=4 T. S=5;SY=2
210 IF K=8 T. S=1;SY=1
220 PX=(XPOS(7)-16)/8;IF PX=10 T. PX=11
230 IF PX=15 T. PX=14
240 PY=(YPOS(7)-24)/8
250 S$=SCR$(PX+SX,PY+SY)
260 IF S=0 OR S$<>" " T. 280
270 DE. M. (7)=SP(4,S,2,4,0,2);POS. 7,PX* 8+16,PY
    * 8+24;M. 7;PL. "O1#F0RO0#F;R"
280 IF CR. (7)>=0 T. 330
290 IF PY=1 AND M=0 T. H=H+1;M=1;SP. H;N=1;
    PL. "O5C0DEDE;O3C0DEF#G"
300 IF PY=15 AND N=1 T. M=0;SP. H,(H-1)* 8+
    104,152;N=0;PL. "O5#G0FEDC;O3#G0#FE#D#

```

C"

```

310 IF H=5 AND N=0 T. PL. "O2G3GO3CCEECC;O1
    C3EGECGC";PL. "GGAGEDC5O2C3O3C1;CEGECGC
    C1CC";G. 340
320 G. 120
340 F. I=0 TO 7;ERA I;SP. I;N.;CLS
350 LOC. 0,20;P. "Press START continue"
360 S=STRIG(0);IF S=0 G. 350
370 IF S=1 T. RUN
380 CLS;LOC. 8,8;P. "GOOD-BY!";LOC. 0,20;E.

```

对比程序 NO. 4、NO. 5, 显而易见, 结构化程序层次分明, 便于阅读理解, 但程序量较大。顺序结构程序紧凑, 阅读理解要困难些。另外 NO. 4 中模块 B 只调用一次, 完全可以放入主程序中, 从而变为混合结构。不难看出, 调整后并不影响程序的层次和易读性。所以说混合结构是较优化的程序结构。

中国计算机学会 联合举办“学装微电脑”函授班
电子工业出版社

第五期招生

近几年来, 我国的计算机普及教育活动有了很大的发展, 许多人学习了计算机语言, 甚至具备了编制程序的能力。但是, 如何进一步利用计算机的接口电路作一些开发应用, 计算机出了故障如何动手修理, 许多人尚缺乏硬件方面的知识。为此, 我们开办“学装微电脑”函授班。

1. 招工对象: 具有中等文化程度的微电爱好者、中学师生、各行业的技术人员和维修人员。

2. 教学计划与学习要求: 函授班为期四个半月(每周6学时, 分3个单元)。

要求学员按时完成每个单元的学习, 将作业和实验报告寄回, 并提出疑难问题, 教师批改作业, 解答问题, 评定成绩。

学业结束后, 根据学员的作业总成绩, 经考试合格者由中国计算机学会发给“全国学装微电脑函授学习结业证书”。

3. 招生日期: 从即日起到92年12月31日止。

4. 开课日期: 93年1月15日开课。

全部学费275元, 包括MP-I型学习机全部元器件, 实验板一块, 教材(已购MP-I机者只交60元教材及教务费)。

报名办法: 请写信到中国计算机学会办公室(地址: 北京2704信箱, 邮编: 100080, 联系人: 宁伟成)索取“学员登记表”, 填好后连同学费一并交齐, 函授班将发出“录取通知书”, 由电子工业出版社寄发零件及教材。



APPLE—II、CEC—I 故障诊断仿真系统

重庆西南师范大学计算机科学系(630715) 王志刚 张一建

维修经验谈

APPLE—II 和中华学习机在我国中、小学以及一些企事业单位仍有相当的数量,但因维修点分布不均,偏远地区的用户遇到微机故障时便感到束手无策,致使坏了的机器一放就是几个月甚至更久,影响正常使用,造成了很大的浪费。

本故障诊断系统是用一台完好的苹果机或中华学习机通过仿真接口卡,模拟故障机的 CPU 进行读写控制,通过人机对话,根据读写情况指示稍懂微机原理的人员作一些逻辑测试后送回状态,然后和正常状态进行比较与推理,并给出故障元器件的位置。

一、仿真接口卡原理

接口卡电路如图所示,U1 和 U2 两个锁存器用来生成被测机系统地址,U3 为故障机数据总线的输出寄存器,U4 为故障机数据读出寄存器,这四个寄存器对于宿主主机来说都是输入输出端口,其端口地址来自 U5 对某一扩展槽口的 DEV SEL 空间译码,这一地址范围为 $\$C0X0 \sim \$C0XF$, $X=N+8$, N 为扩展槽序号,因此 U1 和 U2 的端口地址分别为 $\$C0X0$ 和 $\$C0X1$,数据输出端口的地址为 $\$C0X2$,输入端口为 $\$C0X3$, X 的值依仿真卡插入的槽号而定。由于两机时钟系统不同步,故送往故障机的读写控制信号由一个 D 触发器 U6 产生,该 D 触发器的时钟端连到端口地址译码器 U5 的 Y5 端,当主机复位时,该 D 触发器被置位,Q 输出端为高电平,形成读控制信号,如要进行写操作,要对 $\$C0X4$ 端口进行两次访问,这样在 Q 端产生一个低电平写控制信号,该信号送往故障机的 R/W 控制线,同时该信号允许 U3 锁存的写数据输出到故障机的数据总线。

故障机的 Φ_0 时钟经反相后送 U4 的锁存触发端,这样当 Φ_0 为 6502 周期的下降沿时,U4 时钟端将获得一个上跳沿的触发信号,将故障机数据总线的读出数据锁入锁存器中。因是异机工作, Φ_0 不同步,为了保证诊断机读取输入寄存器时,故障机不发生锁存操作, Φ_0 经或非门 U8 受控于 U6 的 Q2 端,当开机复位时,此端被置为“1”,允许故障机的 Φ_0 反相通过,此时 U4 可以锁存故障机总线上的数据。当诊断机读输入寄存器时,先访问一次端口 $C0X5$,禁止锁存,读完后,再访问一次该端口,恢复锁存允许。

这些地址信号,数据信号,控制信号以及地 GND 与一个 40 脚的插头连接,其顺序和 6502 的引脚信号相同,此插头代替故障机的 CPU 芯片插入故障机的 CPU 插座。

二、诊断原理

诊断时由检测主机通过 CPU 仿真接口模拟 CPU,

进行地址发送,数据读写等操作。其主要思想是当机器发生故障时,我们无法了解 CPU 以及外部主要电路(包括 ROM、RAM 等)的逻辑状态,不能让 CPU 固定地发送某一地址或读写某一数据,也就不能静态地观察某部分电路的状态。苹果机的主要控制都是来自 CPU,而一旦控制失败,必然是某些正常状态被破坏。仿真接口的地址和数据可以保持不变直到你改变它,这样,我们可以针对某一操作固定地从总线给出产生这一操作的地址、数据和控制等信号,再依据完成这一操作的电路逻辑关系,便可逐元件逐脚地测试是否正常工作,最终找出故障元器件。例如:

对于 ROM 的检测,当故障机和诊断机型一致时,可以逐单元将 ROM 中的数据或指令码读出,并和诊断机中 ROM 相应单元进行比较,一旦不相同,可根据地址指示出故障 ROM 芯片的位置。当机型不一致时,从诊断盘中调出该故障机型 12K ROM 二进制文件到 RAM 后,便可依上述原理进行检测了。

检测 RAM 时,对故障机的随机存储器逐单元写一些数,如 $\$00$ 、 $\$FF$ 等,然后读出与原数进行比较,如果读出的数和原写进的不一致,通过判断可确定故障 RAM 的哪一位哪一片。也可用一些公认的 RAM 检测方法进行检测,如下雨法等。

板上 I/O 电路的测试是根据各部分电路的逻辑原理进行的。如扬声器电路的测试,通过对端口地址 $\$C030$ 的交替访问,然后提示维修人员用逻辑笔或示波器测试 (APPLE—I) F12 的第 15 脚, F13 的第 12 脚, J13 的第 5 脚以及 Q4 的 C 脚,再将测得的电平信号或脉冲信号回答给诊断系统,正常时这几脚都有稳定脉冲信号,若不正常则依据逻辑关系可判断 F12、F13、J13 和 Q4 中某件或某几件坏了。

扩展接口电路一般不需仿真卡的帮助可直接插入测试机的扩展槽中进行测试与诊断。由于仿真接口卡能提供静态的或动态的地址与数据,这对于逻辑电路的测试有极大的帮助,本系统仍提供了可直接进行检测的软件。

三、程序设计方法及示例

诊断程序归结为对仿真卡输入输出的控制,根据不同部分的诊断要求,可以分为五类控制。

1. 观察静态的逻辑状态。观察地址总线、数据总线和一些译码电路等的好坏,可以分别将有关的地址数据写到 U1、U2 与 U3 中,和数据输出有关时,需将读写控制 R/W 置为低。

如将地址,数据置全“0”,BASIC 编程如下。(这里假设仿真卡插在 4 号槽口,下同):


```
POKE 49345,0 ;高八位地址置“0”
POKE 49344,0 ;低八位地址置“0”
POKE 49346,0 ;数据输出寄存器置“0”
DW=PEEK(49348) ;R/W 置低电平,允许数据输出。
```

2. 选通某电路或某一逻辑开关。例如置显示模式,只需将地址送往地址锁存器,以下命令置低分辨率图形显示第一页:

```
POKE 49345,192 ;置高八位地址 $C0
POKE 49344,80 ;置低八位地址 $50,图形方式
POKE 49344,85 ;置第二页模式低八位地址 $55
POKE 49344,86 ;置低分辨率方式
```

3. 读取 ROM 中的指令和数据。先将地址锁存,然后关闭数据输入锁存允许,读输入数据锁存器,允许锁存,如读 \$FFFF 单元内容:

```
POKE 49344,255
POKE 49345,255 ;置 $FFFF 到地址锁存器
RW=PEEK(49349) ;暂停数据输入锁存
DB=PEEK(49347) ;读数据输入端口到 DB
RW=PEEK(49349) ;允许锁存
```

4. 读写 RAM 存储器单元。如对 \$400 单元写 166,然后读回。

```
POKE 49345,4 ;置页地址
POKE 49344,0 ;页内 8 位地址
RW=PEEK(49348) ;R/W 置低电平
POKE 49346,166 ;写数到数据输出寄存器
RW=PEEK(49348) ;R/W 置高电平,恢复读状态
RW=PEEK(49349) ;暂停锁存
DB=PEEK(49347) ;读回
RW=PEEK(49349) ;允许锁存
```

5. 动态地给出地址,以便观察某些逻辑电路是否产生脉冲或稳定的波形。如让喇叭发声,选通地址后,在 $\Phi 1$ 作用下,F12-15 等将有方波输出。

```
5000 POKE 49345,192
5010 POKE 49344,48 ;选通喇叭端口 $C030
5020 FOR I=1 TO 10:NEXT I ;延时可根据需要取舍
5030 POKE 49344,0 ;关闭
5040 GOTO 5010 ;循环
```

系统用 FPBASIC 和机器指令写成,运行环境为中华学习机带一个磁盘驱动器或 APPLE-I 等兼容机。可工作在自动诊断方式和菜单选择方式。主控程序先通过人机对话询问仿真接口卡插在那一扩展槽中,测试机和故障机分别为何种机型,然后询问是采用自动顺序检测还是选择对某一部分电路进行检测。如自动方式将按总线、ROM、RAM、板上 I/O 等顺序进行;若只检修与诊断某部分电路,可直接进入该部分相应的诊断模块执行。对于某些故障电路的诊断,为了加快诊断速度,提高诊断准确性,系统将询问故障现象。如显示电路部分,可将显示有无,高低分辨率绘图效果等告诉系统。

由于系统较大,这里以 APPLE II RFI 板为例给出检测 ROM、48KRAM、ROM 地址(包括 I/O 空间)译码电路测试程序:

主要变量及参数说明:

J1\$,J2\$ 分别代表测试机与故障机的机型,例如“RFI”,“CEC-I”等,由主程序通过询问检修人员后确定。

D1,D0 为仿真卡的两个地址锁存器端口地址,D1 的值为 \$C0X1,D0 为 \$C0X0,X=8+N,N 为仿真卡所插槽号,N 的值具体通过主程序对话获得,送往故障机的地址高位存(D1),低位存(D0)。

D2 为数据输出寄存器端口地址,D2 为 \$C0X2。

D3 为数据总线输入锁存器端口地址,D3 的值为 \$C0X3。

D4 为读写 R/W 信号控制端口地址,D4 的值为 \$C0X4。D5 为数据输入锁存器锁存允许端口,D5 的值为 \$C0X5。

A1,A0 为测试 ROM 时的起始地址,A1 装高位,A0 低位,起始值分别为 208 和 0,即十六进制的 \$D000,每测一个单元,低位地址加 1,若大于 255 则 A0 清 0,A1 加 1,直到 \$FFFF(ROM 最高地址)。

程序 1 检测诊断 ROM

```
1000 INPUT“ROM 为 2716 型(1)或 2732 型(2)”；M ;输入 1 或 2
1003 IF J1$=J2$ THEN R0=53248 ;GOTO 1020 ;机型一致,首地址为 $D000
1005 PRINT CHR$(4);“BLOAD”；J2$ ;“A $6000” ;不一致,调监控文件到 RAM
1010 R0=24576 ;置首地址为 $6000
1020 A1=208；A0=0 ;检测首地址 $D000
1025 FOR I=1 TO 6/M ;根据类型;确定循环次数
1030 B=0；FOR J=1 TO 2048*M
1035 POKE D1,A1；POKE D0,A0
1037 RW=PEEK(D5)
1040 IF POKE(D3)<>PEEK(R0) THEN B=R0
1043 RW=PEEK(D5)
1045 A0=A0+1；IF A0>255 THEN A1=A1+1；A0=0
1047 R0=R0+1
1050 NEXT J
1055 IF B=0 THEN 1070 ;无故障转测下一片
1065 PRINT“从右往左第”；I；“片 ROM 坏！最后一个坏单元为：”；B
1070 NEXT I
1080 INPUT“请记载后按回车返回”；A$
1090 RETURN
```

程序 2 检测诊断 RAM

```
2000 PRINT CHR$(4);“BLOAD RAMT.COM,A $6002”；读机器子程序到内存
2003 POKE 24576,N*16 ;置槽号*16 到 $6000
2007 FOR R=0 TO 2 ;每 16K 一组
2008 POKE 24577,0 ;检测结果单元置“0”
2010 FOR I=R*64 TO (R+1)*64-1 ;16K 内页地址
2015 POKE D1,I ;置页地址到(D1)
2020 FOR J=0 TO 255
2025 POKE D0,J ;页内地址到(D0)
2030 CALL 24578 ;调子程序检测
```

```

2032 NEXT J,NEXT I
2035 W=PEEK(24577)      ;读回 16K 检测结果
2040 IF W=0 THEN 2065    ;无故障转测下 16K
2045 FOR K=7 TO 0 STEP-1 ;循环判断
2050 IF W<2^K THEN 2060 ;第 K 位无错转测下一位
2055 PRINT"RAM 芯片";CHR$(67+R);K+3;"已坏!"
2057 W=W-2^K            ;当前位置"0"
2060 NEXT K
2065 NEXT R             ;循环
2070 INPUT "请记录后按回车返回!";A$
2080 RETURN

```

检测 RAM 机器指令子程序 RAMT.COM

```

6002- AE 00 60 LDX $6000 ;取槽号偏移量到 X。
6005- A9 FF LDA # $FF
6007- 9D 82 C0 STA $C082,X ;写全"1"到当前单元
600A- BD 84 C0 LDA $C084,X ;置 R/W 为低
600D- BD 84 C0 LDA $C084,X ;置 R/W 为高
6010- BD 85 C0 LDA $C085,X ;暂停锁存
6013- BD 83 C0 LDA $C083,X ;读回
6016- 49 FF EOR # $FF ;异或错误位变 1

```

```

6018- 0D 01 60 ORA $6001 ;保存到 $6001 单元
601B- 8D 01 60 STA $6001
601E- BD 85 C0 LDA $C085,X ;允许锁存
6021- A9 00 LDA # $00
6023- 9D 82 C0 STA $C082,X
6026- BD 84 C0 LDA $C084,X ;写"0"
6029- BD 84 C0 LDA $C084,X
602C- BD 85 C0 LDA $C085,X
602F- BD 83 C0 LDA $C083,X ;读回
6032- 0D 01 60 ORA $6001 ;保存
6035- 8D 01 60 STA $6001
6038- BD 85 C0 LDA $C085,X
603B- 60 RTS ;返回

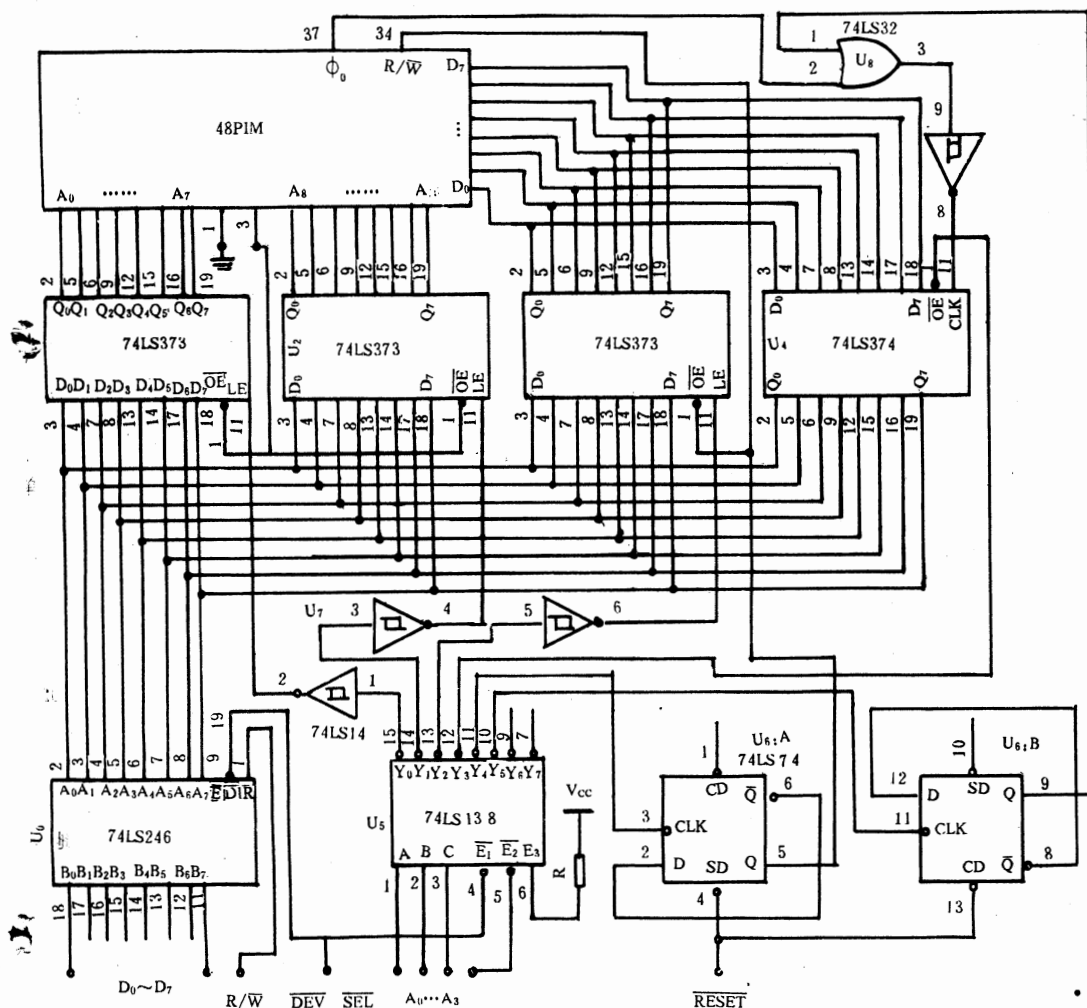
```

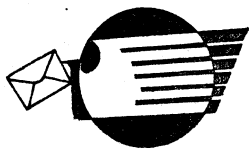
程序 3 检测 ROM 地址译码电路

```

500 INPUT"ROM 为 2716 型(1)或 2732 型(2)";M
505 POKE D1,192
510 INPUT "请测 F12-4 脚,脉冲 P 否则 L";B$
515 IF B$="P" THEN 520
517 T$="01" ;设测时钟标志
(下转 37 页)

```





普及型 PC 个人用户软件交流联谊 活动问题解答(十)

读者联谊

北京中国农科院计算中心(100081) 王路敬

39. 微型计算机病毒一般寄生在磁盘的哪些地方?

微型机系统在目前来说永久性存储介质是硬盘或软盘,微型机病毒是一种可直接或间接执行的文件,是依附于系统特点而没有文件名的秘密程序,它必须以现有的硬件资源而存在。从已经发现的微机病毒来看寄生在磁盘的如下几个区域:

(1)寄生在磁盘引导扇区中。

任何操作系统都有自举过程,例如 PC-DOS,首先由系统读入引导程序并执行它,将 DOS 读入内存。病毒程序就是利用了这一点,自身占据了引导扇区而将原来的引导扇区内容及其病毒程序的其他部分放在磁盘的其他空间,并给这些扇区标为坏簇。

(2)寄生在文件分配表中

作为软盘或者硬盘的 PC-DOS 分区都有一个文件分配表区,不同容量的硬盘或软盘 FAT 的大小有所不同。如表 1 所示:

表 1 各种 PC-DOS 版本各类磁盘 FAT 表

磁盘类型	FAT 表个数	每个 FAT 表占用扇区数	首扇区号	DOS 版本
160KB 软盘	2	1	1,3	V1.0
320KB 软盘	2	1	1,3	V1.1
180KB 软盘		2	1,3	V1.1
360KB 软盘	2	2	1,3	V2.0
1.2MB 硬盘	2	7	1,8	V3.0
10MB 硬盘	2	8	1,9	V2.0
增强 20MB 硬盘	2	8	1,9	V2.0
AT20MB 硬盘	2	8	1,9	V2.0
AT20MB 硬盘	2	41	1,42	V3.0
AT32MB 硬盘	2	64	1,65	V3.0

(3)寄生在硬盘的主引导扇区。

例如:Stone 病毒感染硬盘的主引导扇区,该区与 DOS 无关。

(4)寄生在磁盘内的可执行文件中。

有一类病毒寄生在可执行文件中,例如:.COM 文件,.EXE 文件。有些病毒寄生在文件头部,有些病毒寄生在文件的尾部,也有些病毒寄生在文件的中间,而成为文件的一部分,占据一定的磁盘空间。

40. 微机病毒对磁盘造成哪些危害?

从目前发现的微机病毒来看对磁盘造成的危害主要表现在:

(1)破坏软盘的引导区和硬盘的主引导扇区,使得磁盘操作系统不能正常启动或不能启动。

(2)破坏软盘或硬盘上文件分配表 FAT,使用户在磁盘上的信息丢失。

(3)删除软盘或硬盘上的可执行文件或数据文件。如果删除的文件是系统文件则会导致这片盘不能引导系统。

(4)改变或破坏文件中的数据;改变磁盘分配造成数据写入错误。

(5)在磁盘上产生坏的扇区,使磁盘可用的空间减小。

(6)改变或重写磁盘卷标,对整个磁盘或磁盘的特定磁道或扇区进行格式化。

软盘或硬盘,尤其是软盘,既是病毒寄生的介质,同时又是病毒侵害的重要对象,也是病毒传播的重要途径。

41. 为什么硬盘上的 Stoned 病毒 FORMAT 命令不能消除? 如何解决?

由于硬盘的大麻病毒寄生在硬盘的主引导扇区,系统传递或硬盘进行普通格式化都不能改变主引导记录,因为该区与 DOS 无关,故出现经 FORMAT 格式化的硬盘仍有 Stoned 病毒。

寄生在硬盘主引导扇区的病毒消除方法一是利用公安部颁发的消病毒软件,这是自动的方法。下面介绍另外三种方法:

方法一:

通过一个应用程序把被病毒移走的硬盘主引导扇区的内容复制到被病毒占用的第 0 头 0 柱 1 扇区完成。应用程序如下:

```

SSEG  SEGMENT PARA 'CODE'
        ASSUME CS:CSEG,DS:CSEG
        ASSUME ES:CSEG,SS:STACK
HDUS   PROC FAR
        PUSH DS
        XOR AX,AX
        PUSH AX

        MOV AH,0
        MOV DL,80H
        INT 13

        MOV AX,0201H
        MOV DX,0080H
        MOV CX,0001H
        MOV BX,SEG STONE
    
```

```

MOV ES,BX
MOV BX,OFFS ET STONE
INT 13H
MOV AX,0301H
MOV DX,0080H
MOV CX,0001H
MOV BX,SEG STONE
MOV ES,BX
MOV BX,OFFSET STONE
INT 13H
RET
HDUS ENDP
STONE DB 512 DUP(0)
CSEG ENDS

STACK SEGMENT PARA STACK 'STACK'
DB 256 DUP(0)
STACK ENDS
END

```

方法二:

对硬盘首先进行初始化操作,即进行低级格式化。如果系统带有低级格式化程序可在操作系统下直接执行,若没有低级格式化的文件,IBM PC/XT 及其兼容机可用 DEBUG 程序调用 ROM BIOS 中的低级格式化程序并执行之。

操作如下:

```

A>DEBUG
-G=C800:0005

```

该操作完成后,对硬盘进行 DOS 分区再执行 FORMAT C:/S 命令,即可消除硬盘上的大麻病毒。其他 CPU 为 286、386 微机运行相应的诊断程序中硬盘低级格式化程序即可。

方法三:

如果硬盘中储存了有些没有备份的文件,重建硬盘主引导记录不能采取物理格式化硬盘以及重做硬盘分区操作,因为这样的操作将会造成硬盘的数据资料全部丢失。如果有另一台正常的相同类型的计算机,可以采用一种简单而又可靠的办法重建硬盘主引导记录。实现的方法:

从正常计算机硬盘中提取它的引导记录扇区的内容,然后把它写入已失效的硬盘中,用它来复盖掉已被破坏的硬盘主引导扇区。值得注意的是,用来提取主引导记录的硬盘一定要与失效的硬盘具有相同规格,相同系统和相同的分区格式,否则不可能达到要求。具体操作如下:

(1)在 DEBUG 状态下把正常系统硬盘的主引导扇区的内容以文件方式写在 A 软盘上。

```

-A 100
××××:0100 MOV AX,0201
××××:0103 MOV BX,0200
××××:0106 MOV CX,0001
××××:0109 MOV DX,0080

```

```

××××:010C INT 13
××××:010E INT 3
-G=100
-N A:HDBOOT
-R CX
  ××××
    :400
-W
-Q

```

(2)把该软盘插入已感染大麻病毒系统的 A 驱动器中,然后在 DEBUG 状态下重写硬盘主引导扇区。

```

-N A:HDBOOT
-L
-A 100
××××:0100 MOV AX,0301
××××:0103 MOV BX,0200
××××:0106 MOV CX,0001
××××:0109 MOV DX,0080
××××:010C INT 13
××××:010E INT 3
-G=100
-Q

```

42. 不同 PC-DOS 版本下硬盘分区表有什么差别?

不同的 DOS 版本对硬盘分区表的编排方式上有所不同,当整个硬盘空间只含一个 DOS 分区的情况下,DOS 2.0 或 2.10 等是把这一分区信息存放在偏移 1EEH 起始的位置上(即第 4 分区表位置),而 DOS 3.30 则把分区信息存放在偏移 1BEH 位置上(即第 1 分区表位置)。DOS3.30 可以从第 1 或者第 4 分区表位置上识别硬盘唯一的一个分区,而 DOS 2.0 或 2.10 等版本却不可以。因此,如果硬盘是 DOS 3.30 系统,从 A 盘用 DOS 2.0 或 2.10 作引导之后,会出现系统不认识硬盘的问题,试图对硬盘进行操作会显示:“Invalid drive specification”错误信息。这时并非硬盘有什么故障,而是 DOS 版本的差异所引起的。当然,如果是硬盘控制器板或者系统参数没有正确设置,也会出现同样的错误信息。除此之外,DOS 2.0 或 2.10 等版本与 PC-DOS 3.0、3.10、3.20、3.30 等版本在文件分配表 FAT 上也有一个重要的差别。这就是 DOS 2.0 或 2.10 等以下版本,FAT 表中的每一项固定为 12 位(二进制)。而 DOS 3.0 以上版本 FAT 表的每一项可以是 12 位也可以是 16 位,至于 12 位还是 16 位是由 FDISK 程序在建立 DOS 分区时根据用户指定分区的大小来决定的,如果 12 位数能表示出 DOS 分区中所有的簇数就使用 12 位,否则使用 16 位。DOS3.0 以上可以产生和识别 12 位或 16 位两种形式的 FAT 表,而 DOS2.0 或 2.10 等版本只可以产生和识别 12 位的 FAT 表。这就是从 A 驱动器用 DOS2.0 或 2.10 作引导不能对 16 位 FAT 的 DOS3.0 以上系统硬盘进行操作的原因。

SJW-B 系列自动补偿式 三相大功率电力稳压器

原理 · 特点 · 用途



上海精达电子仪器厂
技术科

电压不稳不仅给工业生产、科学研究和日常生活增添了不少麻烦,有时甚至还会影响生产,损坏设备,造成事故。随着现代科技的迅猛发展,工矿、科研、邮电、医院、宾馆等部门,对电源电压稳定性的要求越来越高。尽管稳压电源种类很多,传统的有电子交流式、感应式与磁饱和式稳压器等,但这些稳压器容量较小,损耗较大,波形失真严重,且对负载性质适应性又差,均不能满足生产和科研日益增长的需求。

SJW-B 系列自动补偿式三相大功率电力稳压器是我厂引进、消化、吸收国外的先进技术,结合国情,精心研制、专业生产的节能型新产品。其性能之优、功率之大、价格之廉是传统的稳压器无法比拟的。我厂生产的 SJW-B 系列,其性能与技术指标可与国外同类产品媲美,而价格仅是国外同类产品的 1/6~1/7,深受广大用户的赞誉。

1. 原理 三相自动补偿式电力稳压器的工作原理如附图所示。它由调压器 T1、补偿变压器 T2、伺服电动机 MS 与电压采样比较控制装置等组成。现以 A 相为例,说明其稳压原理。从附图中可知:

$$U_{A0} = U_{A1} - \Delta U_A$$

式中: U_{A0} —A 相的输出电压;

U_{A1} —A 相的输入电压;

ΔU_A —A 相的补偿电压。

当输入电压 U_{A1} 或负载变化引起输出电压 U_{A0} 变化时,电压采样比较控制装置从稳压器输出端采样,采样电压经整流滤波后与基准电压上限值和下限值比较。当采样电压大于基准电压上限值时,电压采样比较控制装置控制继电器动作,使伺服电动机 MS 作一定方向旋转,带动调压器 T1 上的电刷组作相对滑动,这时在 A 相调压器上出现一个 U_{A2} ,在补偿变压器 T2 的 A 相一次侧相应产生 U_{A3} ,则在二次侧产生补偿电压 ΔU_A ,它使输出电压 U_{A0} 下降,直至回复到稳压精度的允许范围内。此时,继电器释放,伺服电动机 MS 停止转动。反之,伺服电动机 MS 向反方向旋转,补偿电压 ΔU_A 改变极性,使输出电压 U_{A0} 上升,直至回复到稳压精度的允许范围内,以达到输出电压稳定的目的。

2. 特点

(1) 稳压范围很宽 输入电压容许在 $380V \pm 20\%$ 范围内变化,即输入电压在 304~456V 范围内,均

能正常工作。

(2) 稳压精度很高 稳压精度为 $\pm 1\%$ (输出电压为 $380V \pm 1\%$),且稳压精度在 $\pm (1 \sim 5)\%$ 可调。

(3) 损耗很低 电能转换效率 $> 98.5\%$,损耗很低,节能明显。

(4) 负载性质适应性很广 能适应任何阻性、感性和容性负载。

(5) 输出功率很大 额定输出功率有 20~1000kVA 等 12 种规格,其容量之大是传统稳压器望尘莫及的。

(6) 波形失真度很小 输出电压的波形与输入的电网正弦波形几乎一致,其波形畸变 $< 0.1\%$ 。

(7) 应变时间很短 调压应变 $> 20V/s$; 过压保护应变 $< 0.5s$ 。

此外, SJW-B 系列电力稳压器还具备长期连续工作、能承受瞬时超负载、有可靠的过压保护和故障自停等优点。

3. 应用 SJW-B 系列自动补偿式大容量三相电力稳压器能适用一切需要电压稳定的场合,尤其适用于电压波动大、负载变动大的用电场所。它是任何用电部门和引进设备稳压电源的理想产品,可广泛应用于工业、交通、邮电、军事、铁路、医院、宾馆和科研等领域。例如:它可作为工厂精密机床、仪器设备、自动联动生产线,大型医疗设备,广播和电视台等优质稳压电源。

本厂产品获国家电力电子产品合格证书。

上海精达电子仪器厂

地址 上海市新闻路 579 号

电话 2563294

2170089

电挂 6910

邮编 200041

