

E & C

1993

●一九九三年 ●总期第99期

6

電子

ISSN 1000-1077

與電腦



• ELECTRONICS AND COMPUTERS •

录像机图集与维修指南

——日立系列

录像机图集与维修指南

——日立系列

录像机图集与维修指南

——东芝系列 1

录像机图集与维修指南

——东芝系列 2

录像机电路图集

——松下系列

录像机图集与维修指南

——夏普系列 1

录像机图集与维修指南

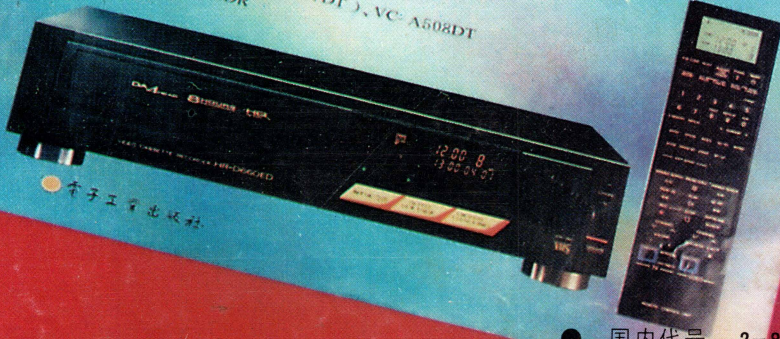
——夏普系列 2

● 李小东 田耕 滨川长 逢 等汇编

● 夏普 VC-A103D, VC-A501D, VC-

A507DC, A506D 和 A507DT), VC-A508DT

● 夏普 VC-6V3DR



● 电子工业出版社

● 国内代号: 2-888 定价: 1.60元

热烈祝贺

电子工业出版社广州科技公司在穗开业

电子工业出版社广州科技公司业务简介

电子工业出版社是我国以出版电子科技图书为主的一家直属电子工业部的中央一级出版社。成立十余年来,致力于为我国电子行业的生产、科研、教学、管理人员和广大电子爱好者服务;致力于发展国内外文化、技术的交流与合作,繁荣我国科技出版事业;振兴我国电子科学技术,促进科学技术和生产相结合,促进高新技术研究成果的商品化,加快具有中国特色的社会主义的建设是本社的唯一宗旨。

随着改革开放的进一步深入,为了充分利用珠江三角洲地区信息快、商品新、科技发展迅速的优势,特在广州开设“电子工业出版社广州科技公司”,以繁荣高新技术市场、促进微电子和电子信息等高新技术、产品的交流和推广为宗旨,充分组织以广州为中心的南方地区的科技优势,及时总结科技专家的研究成果,通过资料、期刊、图书的出版以及科技咨询,科技转让,科技服务等,为加快“四小虎”的发展尽微薄之力。

公司下设:编辑出版部;图书发行部;综合经营部;技术开发部。

《今日电子》《电子与电脑》读者服务部;“中国软件行业协会软件出版分会”软件销售中心(广州部)等,已准备设在本公司,筹备工作正在进行。

希望广东乃至全国关心电子科技出版事业发展的作者、读者们给予关照和支持。

我们真诚的希望有更多的作者同我们联系出版事宜,欢迎赐稿。欢迎各书店和广大读者来我公司选购及批发电子版图书、期刊、软件和音像出版物。

公司地址:广州市天河五山路华师大科技服务楼 215 号

邮 编:510631

联系人:周青峰 陈昊

《录像机图集与维修指南》系列丛书

●《录像机图集与维修指南——松下系列①》

NV-370EN、NV-450MC、NV-G10MC、NV-G30MC、NV-G33MC、
NV-G50MC、NV-G50PX、NV-G300EM 定价:15.40 元

●《录像机图集与维修指南——松下系列②》

NV-L15EN、NV-L15MC、NV-L15BD、NV-L18EN、NV-L18BN、NV-
L10EN、NV-L20EA 定价:9.50 元

●《录像机图集与维修指南——日立系列①》

VT-340E(CS)、VT-660E(DH)、VT-136E(DH) 定价:12.50 元

●《录像机图集与维修指南——日立系列②》

VT-426E(DH)、VT-427E(DH)、VT-547E(DH) 定价:12.50 元

●《录像机图集与维修指南——东芝系列①》

V-300DC、V-500DC、V-83DC/E、DV-90_{DC}、V-93_{DC}、DV-98C、V-94C
定价:13.50 元

●《录像机图集与维修指南——东芝系列②》

V-110C/V-95C、V-880MC/880MS 定价:17.00 元

●《录像机图集与维修指南——夏普系列①》

VC-789ET(779E)、VC-381MC、VC-583MC、VC-8583W 定价:
24.50 元

●《录像机图集与维修指南——夏普系列②》

VC-A103D、VC-A501D、VC-507D(A506D 和 A507DT)、VC—
A508DT、VC—6V3DR 定价:23.00 元

通信地址:北京万寿路电子工业出版社发行部

邮 编:100036

电 话:813693

电挂:3101

开户银行:北京市工商银行翠微路分理处

帐 号:661036-40

户 名:电子工业出版社发行部

注:购书另加 15% 邮挂杂费,款到发书,邮路丢失,向我部索赔

电子与电脑

一九九三年 总期第 99 期

目 录

· 综述 ·

人工智能型电脑的发展概况
..... 万欣 高增锁 万红(2)

· PC 用户 ·

数据库系统下求三角、反三角函数的命令文件方法
..... 凌莉 葛晓光(3)
在不同的目录结构中使用 RESTORE 命令
..... 蔡廷武(5)
Epson LQ1600 对屏幕彩色图形的硬拷贝
..... 周晓(6)
C 编程中易犯的的错误 苑丽霞(8)
C—WordStar 使用基础问与答 朱大公(9)
在 TANGO 软件中建立汉字元件库 肖安顺(11)
病毒一例 谭洪勇(12)
“3.6”# 一种超级恶性病毒 李建俊(12)

· 学习机之友 ·

任意数任意次整数幂的高精度运算程序 张新莲(13)
高精度数值算法的改进 邓阳春(15)
ProDOS 系统内部结构剖析(续) 廖凯(17)

· FORTH 语言讲座 ·

第七讲 程序控制结构(下) 丁志伟(18)

· 初、中级程序员软件水平考试辅导 ·

数据结构 夏晓东(22)

· 学用单片机 ·

单片机的学习与应用 罗明宽 车金相(32)
80C31 单片机防掉电和抗干扰电源的设计
..... 马祝阳 姜凤怡(34)
从三种通用的八位单片机硬、软件比较来看 Z8 系列单片机的优越性 董伯明(36)

· 电脑巧开发 ·

单片机主从式通信系统中的广播发送 江琪(39)
给单片机加装 V/F F/V 转换器 岳海军(40)
一种防止单片机程序运行失常的软件对策
..... 黄稳山(41)
病毒克星——华思中文防病毒卡 (28)

· 维修经验谈 ·

NKP—824G 打印机一故障维修方法
..... 蒋国权 张宏杰(42)
M2024 打印机双向打印时错位的解决办法
..... 陆钦俭(43)
Super—AT286 微机硬盘软故障排除一例
..... 费毅(43)

· 电脑游戏机 ·

第六讲 游戏程序实例分析(下) 于春(43)

· IC 电路应用 ·

集成滤波电路及用法(二) 李兰友(47)
高频变压器的设计与选择 戴惟荫 张占松(50)

· 电脑通信 ·

TCP/IP 网络简介 杨军(52)
Transputer 与并行处理 师军(53)

· 电脑通信大家谈 ·

如何选用 MODEM 卡—PC-PC 远程通信(一) (54)
传真机专题讲座(4)
传真机传输控制规程 张建军 张景生(55)

· 读者联谊 ·

EXPANDED MEMORY 与 EXTENDED MEMORY 译名管见 彭禾(52)

机械电子工业部电子工业出版社主办

编辑、出版:《电子与电脑》编辑部

(北京 173 信箱 邮政编码:100036)

印刷:北京三二〇九厂

国内总发行:北京报刊发行局

国内统一刊号:CN11—2199

邮发代号:2—888

国外代号:M924

出版日期:每月 23 日

主编:王惠民 特约编审:苏子栋

责任编辑:施玉新

订购处:全国各地邮电局

国外总发行:中国国际图书贸易总公司

(北京 399 信箱 邮政编码 100044)

广告经营许可证:京海工商广字 147 号

定价:1.60 元

人工智能型电脑的发展概况

河北农科院昌黎果树研究所仪器所(066600)

万欣、高增锁、万红

自1946年世界第一台电脑问世以来,经历了电子管、晶体管、小规模集成电路、大规模集成电路四代的发展。目前,各国一方面在继续完善、提高第四代电脑,同时在前四代的基础上研制、开发新一代产品——人工智能型电脑。

人工智能型电脑,是计算机学、电子仿生学、控制论、神经生理学、数学、语言学、逻辑学等多种学科相互交叉渗透的产物。它能够翻译外文、识别物体图像、能听懂自然语言、有自学习能力、会适应环境,以及具备可接受启发,自己判断和推理等类似人脑“智能”的功能。在人工智能型电脑的孕育期中,许多科学家为它的诞生作出了巨大的贡献。阿兰·图灵的“理想计算机”、“计算机能思维吗?”;维纳的控制论;商农的信息论;冯·诺意曼的博弈论;阿希贝的“大脑设计”等都为人工智能电脑的研制提供了坚实的科学依据和理论基础。1956年夏季,在美国新罕布尔州达特玛斯大学,举行了第一次人工智能研讨会,此举标志着人工智能学科的正式诞生。1956—1961年属于研究初期。这期间,纽厄尔·西蒙·肖乌创立了LT逻辑推理程序,使机器表现出类似人脑的演绎推理。麦卡锡提出的LISP表处理语言,可用于符号微积分计算、数学定理证明、博弈、图像识别等人工智能的其它领域。塞缪尔设计的自适应、自学习、自组织的跳棋程序,举世瞩目,轰动一时。五年期间,人工智能取得的卓越成就,显示了其强大的生命力。1961年后为发展期。各国在机器翻译、图像识别、自然语言理解、博弈、定理证明、问题求解、智能机器人等研究领域均取得了可喜的成果。

机器翻译

电脑翻译的设计方法是:事先要在电脑内存储好两种文字的对应该词典程序和一套翻译规则程序。翻译用的语法分析表格也是专门针对特定的翻译对象。对于这样一个限定的语言系统翻译,各国均获得了不同程度的成功。1953年,美国乔治敦大学进行了第一次机器译文试验。次年纽约国际商业机器公司利用250个词在IBM701电脑上进行俄译英的公开表演。随后苏联进行了英译俄试验。德国萨尔大学的机器自动翻译系统于1974年进行俄—德、英—德、世界语—德翻译。翻译的方向是不可逆的。西门子公司的机器辅助翻译系统,1976年公布于众。它的源语言和目标语言是德、英、法、俄、意大利、葡萄牙语。翻译的方向可根据要求而定。日本搞英语和日语之间的机器翻译,在翻译英

语课本时正确率达90%,翻译技术论文时正确率达75%。目前,国际上已有一些电脑翻译系统公开为社会服务。

图像识别

在人工智能电脑的研究中,机器对各类图形、物体的识别有着极其重要的现实意义。早期的机器只能识别数码字形,从而应用于邮政编码分类、考生成绩统计、分类、图书号码分类等。目前,已能识别和分析工业图纸、特定物体机件、地球卫星云图、医学图象、细胞统计图像、分子空间结构等。

自然语言理解

为使电脑和人类自然语言直接打交道,60年代,开始出现了一些专门格式的问答系统程序。目前能达到的程度是:

a、简单的人机对话。如自动预售车票、查询电话号码、智能机器人等。据美国《新闻周刊》92年报道,美国苹果公司研制的智能电脑“卡斯珀”能够听懂整个句子的自然语言,并能用相当漂亮的英语对多个问话者回答。有的程序能够回答特定多种科学问题,有的可解答生活问题,有的可热情有礼的招待客人。

b、文件理解。机器阅读和消化文件内容,能够做出文章摘要、或者在这个基础上回答具体问题,指导读者查询有关参考资料。

c、文件生成。根据人们查询的目的和要求,电脑把内部信息翻译成普通语言。

d、为某系统的特定目的服务。可实现情报自动检索、辅助教学、帮助控制复杂的技术设备、为专家系统提供咨询服务等。

博弈的能手

下棋,在数学上称为博弈,是研究对策的。1959年美国工程师塞缪尔在IBM704电脑上编制了一套下棋程序,在下棋时他本人输给了电脑。1962年他改进了原来的设计,在比赛时电脑竟然战胜了美国一个州的赛棋冠军。电脑要成为一个好“棋手”就要事先将下棋的规则用算法语言编成程序,存储在电脑内。具有“自学习自分析”的下棋程序,能把过去的棋局和下法记录下来,在下棋时能向对手吸取长处,积累经验,既能防守又能进攻,而且善于掩盖自己的战略企图,诱导对手发生误解以战胜对方。这就是智能电脑高超的逻辑思维功能。目前已研制出国际象棋、西洋跳棋、十五子棋等众多的博弈程序,很多程序达到了大师级水平。

智能机器人

随着人工智能学科的飞速发展,产生了具有实用价值的智能机器人。智能机器人具有感知和理解周围环境、能使用和理解自然语言、具有逻辑推理和操作生产工具的技能、能通过自学习从而适应特定环境来模仿人完成某种动作,并能代替人类从事危险、复杂、或不适合人类工作的场所代替人进行工作。总之,智能机器人建立了人的模型。机器人到目前为止经历了三代的发展。第一代机器人具有记忆功能,能往复重复操作某种简单的工作。第二代机器人具有触觉和视觉的简单功能,能从杂乱的工件中选出所需的零件进行装配,本身装有移动机构,可在小范围内活动。第三代为多智能机器人。美国研制的多智能机器人“科沃”于1966年就可以从数百米深的海水中打捞试射或丢失的武器。智能机器人“Hibat”可以帮助家庭安排作息时间、准备早餐、打扫房间、开关电灯、洗衣服等。苏联研制的月球移动式机器人“登月者”可在月球上完成定量工作。此

外尚有专门管理仓库的机器人、从事反复组装工作的机器人、引导盲人步行的机器人、搜查和排除地雷、指挥交通的机器人等等。据统计,目前世界上有三万多个机器人。其中日本二万多、美国五千多、西欧各国近一千个、苏联五百多个。这些机器人除少数用于军事、空间、海洋、家庭方面外,绝大多数用于工业部门从事装卸、搬运及生产工作。在极其恶劣的环境下采用机器人来代替人的劳动,其社会效益是非常明显的。

人工智能电脑的研制与应用虽然取得了可喜的成果,但仍处于初级阶段,尚有相当的理论 and 实际问题亟待解决。但是,各国科学工作者对其抱有极大希望,国外予测21世纪前将有各类价格适度的智能机器电脑投入使用。予计1994年智能电脑将可接受近于日常使用的自然语言或文字的输入指令;1996年家庭用机器人批量投入市场;1998年可通过声音来编程序;2001年自动同步翻译机将有所突破;2004年只要直接输入问题本身电脑便可自动处理。

数据库系统下求三角、反三角函数的命令文件方法

煤炭科研院合肥研究所(230001) 凌莉 葛晓光

数据库语言系统 dBASE III、dBASE III+ 和 FoxBASE+ 中都含有丰富的内部函数,但遗憾的是未设置三角函数与反三角函数,从而使某些科学数据的处理和对数据库内容的一些图形表示手段(如旋转、透视等)无法实现。这里介绍一种运用数据库语言中基本四则运算和逻辑运算功能,通过参数传递型命令文件求解三角函数与反三角函数的方法。

程序设计所依据的数值计算原理是正弦函数、余弦函数和反正切函数所对应的 Taylor 级数展开,还采用了一些三角函数与反三角函数的恒等变换,以确保收敛性,提高求解精度和计算速度。本文列出求正弦函数 $\sin(x)$, 反正切函数 $\arctg(x)$ (值域 $-\pi/2 \sim \pi/2$), 双参反正切函数 $\arctg(x, y)$ (值域 $-\pi \sim \pi$) 的程序文件, 分别见程序 1、程序 2 和程序 3 (注:程序 3 &a~&b 之间的语句与程序 2 相同)。应在使用前将这三个程序用 EDLIN、WordStar (在 DOS 下) 或 MODIFY (在 dBASE、FoxBASE 下) 编辑好存盘。规定:程序 1 文件名 SIN.PRG, 程序 2 文件名为 ATN.PRG, 程序 3 文件名为 ATN2.PRG。

求三角函数、反三角函数方法举例如下:

```
例 1, 求  $\sin(\pi/6)$ 
. x=3.14159265359/6
. y=0
.DO SIN WITH X,Y
. ? Y
0.5000000000
```

例 2, 求 $\arctg(20)$

```
. Y=0
.DO ATN WITH 20,Y
. ? Y
1.5208337733
例 3, 矢量在 X 轴投影为 -3, Y 轴投影为 -1.5, 求该
矢量方向角
. A=-1.5
. ALPHA=A
.DO ATN2 WITH -3,A,ALPHA
. ? ALPHA
-2.6778060122
```

还可以给出求其它三角函数和反三角函数的程序,但是利用这三个程序再结合一些恒等变换就可求出所有的三角函数与反三角函数。

如: $\cos(x) = \sin(\pi/2 - x)$, $\text{tg}(x) = \frac{\sin(x)}{\cos(x)}$

$\arcsin(x) = \arctg\left(\frac{x}{\sqrt{1-x^2}}\right)$, $\arctg(x) = \pi/2 - \arctg\left(\frac{1}{x}\right)$

等

例 4, 求 $\text{ctg}(2.5)$

```
. STOR 0 TO ,X,Y1,Y2
.DO SIN WITH 2.5,Y
. X=3.14159265359/2-2.5
.DO SIN WITH X,Y2
. ? Y2/Y1
-1.3386481283
```

有几点需说明:

1. 这几个程序都是参数传递型命令文件,调用时参数表中各参数既不能多,也不能少,如 do sin with x 或 do sin with x,y,z 都是非法。在参数表中,第1个参数(包括 ATN2 的第2个参数)为自变量,可任取变量与常数,若取变量,运算后仍保持原值不变;第2个参数(ATN2 的第3个参数)用来返回计算结果,应事先用赋值方法定义成数值变量。

2. 几个例子都缺省了磁盘符和路径名,若这些文件不在当前盘的当前路径下,则不能缺省,例:do b:\file\sin with 0.3,y。鉴于本法每求一次三角(或反三角)函数都访问一次磁盘,如果求解次数较多,建议在内存中开辟虚拟盘,存入这几个程序文件供调用,以减少磁盘磨损,提高速度。

3. 从数值原理上,所求函数值可通过控制量获得任意精度,由于运算过程中的舍入误差等因素所致,实际精度仍有限。经实测,正弦函数计算值的相对误差不足千万分之一,此精度比一般算法语言中的单精度函数结果高得多。反正切函数计算精度要差一些,但最大相对误差也在千分之一以下。

4. 这几个程序既可在 dBASE, FoxBASE 的会话状态下直接使用,也可以被其它程序调用。各程序及计算实例曾分别以 dBASE III, dBASE III+, FoxBASE+ 2.10 等系统版本,在 IBM PC-XT、AT, Compag386/33 等微机上通过。

程序 1

```

&& SIN. PRG
PARA XA, YA
PRIV ALL LIKE A *
AX=XA
SET DECI TO 15
SET FIXE ON
AP=3.141592653589793/2
AG=1
IF AX<0
  AX=-AX
  AG=-AG
ENDI
DO WHIL AX>AP
AX=AX-AP
AG=-AG
ENDI
IF AX<0
  AX=AX-AP
  AG=-AG
ENDI
IF AX>.5*AP
  AX=AP-AX
  AXP=1
  YA=1
  AN=2
ELSE
  AXP=AX
  YA=AX

```

```

AN=3
ENDI
AS=-1
AX=AX*AX
DO WHIL AXP>.0000000000001
  AXP=AXP*AX/(AN-1)/AN
  YA=YA+AS*AXP
  AN=AN+2
  AS=-AS
ENDD
YA=YA*AG
SET DECI TO 10
RETURN

```

程序 2:

```

&& ATN. PRG
PARA XA, YA
PRIV ALL LIKE A *
SET DECI TO 15
SET FIXE ON
AX=XA
AP=3.14159265359/2 &&a
AG=1
IF AX<0
  AX=-AX
  AG=-AG
ENDI
DO CASE
  CASE ROUND(AX,6)=0
    YA=0
  CASE ROUND(AX,6)=1
    YA=.5*AP*AG
  CASE ROUND(AX,6)=3.2
    YA=1.267911434174*AG
  CASE ROUND(AX,6)=.3125
    YA=(AP-1.267911434174)*AG
OTHE
IF AX>1
  YA=AP
  AS=-1
ELSE
  AX=1/AX
  AS=1
  YA=0
ENDI
DO CASE
CASE AX<6.65.AND. AX>1.64
  YA=1.26791143417
  IF AS=1
    YA=AP-YA
  ENDI
AX=(1+AX*3.2)/(AX-3.2)
IF AX<0
  AX=-AX
ELSE
  AS=-AS

```



```

ENDI
CASE AX<1.64
YA=AP*.5
AX=(1+AX)/(AX-1)
AS=-AS
ENDC
AX=(AX*AX+1)^.5-AX
AA=AX
AX=AX*AX
AN=1
AB=AA
AY=AB*AS
DO WHIL AB >.000000000001. OR. AY/AB <
1000000000000
AS=-AS
AA=AA*AX
AN=AN+1
AB=AA/(2*AN+1)
AY=AY+AB*AS
ENDD
YA=(YA+AY*2)*AG
ENDC &&b
SET DECI TO 10

```

RETURN

程序 3:

```

&&.ATN2.PRG
PARA XA,XB,YA
PRIV ALL LIKE A*
SET DECI TO 15
SET FIXE ON
AX=XB/XA
AP=3.14159265359/2 &&a
.....
.....
ENDC &&b
IF XA<0
IF YA>0
YA=YA-2*AP
ELSE
YA=YA+2*AP
ENDI
ENDI
SET DECI TO 10
RETURN

```

在不同的目录结构中使用 RESTORE 命令

农行江苏省射阳县支行(224300) 蔡廷武

在微机操作过程中,我们经常用 BACKUP 命令,把硬盘上大量的数据备份到软盘上,防止计算机硬盘出故障,导致数据文件丢失。

以 DOS 3.30 为例,当用 BACKUP 命令由硬盘备份数据到软盘上后,在软盘的根目录下,生成两个叫做 BACKUP.×××和 CONTROL.×××的文件,如果备份的是第一张软盘,则两个文件的文件名为 BACKUP.001 和 CONTROL.001,第二张盘为 BACKUP.002 和 CONTROL.002.BACKUP.×××是备份文件链接在一起的大文件,而 CONTROL.×××文件则保存了备份的目录名、路径及其他控制信息。

BACKUP 命令备份下来的文件,不同于 COPY 命令复制下来的文件。COPY 命令产生的是文件不变的副本,而 BACKUP 命令产生的文件,只有供 RESTORE 命令恢复文件用的控制数据,需要用 RESTORE 命令将它们恢复,才能在计算机上应用。

用 RESTORE 命令恢复文件时,必须在这之前,已经有用 BACKUP 命令备份下来的文件,而且恢复文件所在的目录、路径必须和 BACKUP 命令备份文件所在的目录、路径一致。倘若目录、路径不一致,用 RESTORE 命令无法恢复到你想恢复的目录中去。笔者借助工具软件 PC Tools 5.0,用两种方法对备份文

件在不同的目录结构下进行恢复。

第一种方法是对硬盘中的目录进行改名和移位,即可达到预期的效果。此方法对庞大的数据进行恢复,效果特别好。

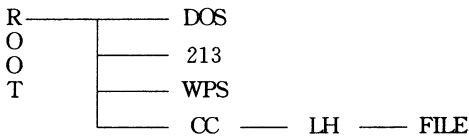
例如,把在\CTW\DATE 目录下由 BACKUP 命令备份的数据文件,恢复到另一台微机的\CC\LH\FILE 目录中去。

一、在系统中运行 PC Tools 工具软件,对目录\CC\LH\FILE 改名、移位。

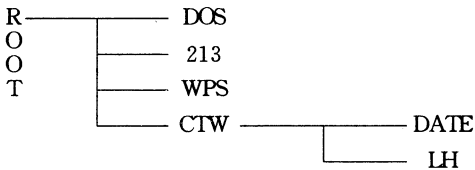
[C:\]pctools

1. 按“F3”键,对磁盘操作;
2. 按字母“D”(Directory maint),对目录操作;
3. 移动光标,定位在子目录 CC 上,按“F1”键(rename),改子目录名 CC 为 CTW,用同样方法把子目录名 FILE 改成 DATE;
4. 移动光标,定位在子目录 CTW 上,按“F4”键,使 CTW 为当前目录;
5. 移动光标到目录 DATE,按“F5”键(prune & graft),然后键入字母“P”,再把光标移到子目录 CTW,按“回车”键;
6. 键入字母“G”,然后按“回车”键;
7. 退出 PCTools;

修改前的目录结构为:



修改后的目录结构为:



二、执行 RESTORE 命令,恢复数据到目录\CTW\DATE 中。

三、再用工具软件 PC Tools,把修改后的目录名、目录结构转换成原来状态。

第二种方法是修改备份软盘上的文件 CONTROL.×××,使得 CONTROL.×××文件中目录名、路径和硬盘中的目录结构一致,例子同上。

[C:\]pctools

1. 按“F10”键选择 A 盘中的文件 CONTROL.001;
2. 键入字符“E”(view/Edit);
3. 按“F1”键(toggle mode),在屏幕上显示文件头的 ASCII 码和十六进制代码如下:

Path=A:/*.*

File=CONTROL.001

Displacement	Hex codes
0000(0000)	8B 42 41 43 4B 55 50 20 20 10 00 00 00 00 00 00
0016(0010)	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0112(0070)	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0128(0080)	00 00 00 00 00 00 00 00 00 00 00 00 FF 46 43 54 57 5C
0144(0090)	44 41 54 45 00 00 00 00 00 00 00 00 00 00 00 00
0160(00A0)	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0176(00B0)	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0192(00C0)	00 00 00 00 00 00 00 00 00 00 00 00 00 04 00 FF FF F
0208(00D0)	FF 22 4C 5A 44 2E 44 41 54 00 00 00 00 00 03 8
0224(00E0)	16 00 00 01 00 00 00 00 00 80 16 00 00 20 00 72
0240(00F0)	52 79 17 22 4C 4C 4C 4B 2E 44 41 54 00 00 00 0

4. 再按“F3”键(edit),通过修改 ASCII 码或十六进制代码的方法,把 FCTW\DATE 修改成 FCC\LH\FILE;

5. 然后修改其他备份软盘中的 CONTROL.×××文件;

6. 执行 RESTORE 命令,把数据文件恢复到指定目录\CC\LH\FILE 中去;

另外,对恢复文件较少,硬盘容量允许以及不知道备份文件的来源,可以在 C 盘的根目录执行命令:

[C:\]restore a: c:/s

因为执行带参数/S 的 RESTORE 命令,它将在 C 盘上重新建立由 BACKUP 命令备份的源目录名及路径,并恢复数据文件到相应的子目录,最后可用 COPY 命令复制文件到你恢复的子目录中。

Epson LQ1600 对屏幕彩色图形的硬拷贝

广东湛江秀湖 81—603(524091) 周 晓

本刊 1992 年第五期上刊登的张辉、李乐升两同志的文章《Color 400 卡屏幕彩色图形的硬拷贝》颇具实用价值。只可惜该文所附程序是针对 M2024/M1724 打印机的,而对目前使用较普遍的 Epson LQ1600 打印机却不适用。笔者经探究,在张、李二位同志的程序的基础上加以修改,使之也适应于 Epson LQ1600 中英文打印机,其修改部分如下:

.置打印机行距	;置打印图形状态于程序
MOV AL,0AH	PUSH AX
CALL PRI	MOV AL,1BH
MOV AL,0DH	CALL PRI
CALL PRI	MOV AL,2AH
MOV AL,1BH	CALL PRI
CALL PRI	MOV AL,27H
MOV AL,33H	CALL PRI
CALL PRI	MOV AL,80H
MOV AL,08H	CALL PRI
CALL PRI	MOV AL,07H
XOR DX,DX	CALL PRI

XOR AX,AX

POP AX

须注意的是:原程序的置打印机图形状态子程序中共有 4 个控制码,而对 EPSON LQ1600 的编程则须 5 个、它们分别是 1BH,2AH,27H,80H,07H,且列数 0780H 中的低字节 80H 放前、字节 07H 放后。此外,判断图形模式的 CMP AL,42H 应为 CMP AX,12H;判断行数是否结束的 CMP AX,200 应改为 CMP AX,199。

code	segment para'code'
org 100h	
start	proc far
	assume cs:code
	assume ds:code,es:code
	jmp cccc
prts	proc near
	sti
	push ds
	push es
	push di
	push si

	push bx		p1:	call crlf		db 0bh,05h,0bh
	push cx			pop ax		db 06h,0fh,06h
	push dx			pop cx		db 06h,09h,06h
	push ax			ret		db 02h,09h,06h
	mov ax,cs		pr640	endp		db 0fh,0fh,07h
	mov ds,ax		crlf	proc near		db 0eh,0fh,07h
	mov es,ax			push ax		db 02h,0fh,06h
z:	xor dx,dx			mov al,0ah		db 0dh,0bh,0dh
	mov ah,2			call pri		db 06h,0fh,07h
	int 17h			mov al,0dh		db 04h,01h,04h
	and ah,10h			call pri		db 00h,04h,00h
	cmp ah,10h			pop ax		db 00h,00h,00h
	jz zz			ret	read2	proc near
	inc dx		crlf	endp		push dx
	jmp z		prcon	proc near		push cx
zz:	mov si,dx			push ax		mov bx,2
	mov ah,15			mov al,1bh		lea di,buf
	int 10h			call pri	oop1:	call read1
	cmp al,12h			mov al,2ah		inc dx
	jnz exit			call pri		dec bx
	mov al,0ah			mov al,27h		jnz oop1
	call pri			call pri		mov cx,3
	mov al,0dh			mov al,80h		mov bx,offset buf
	call pri			call pri	oop2:	push cx
	mov al,1bh			mov al,07h		push bx
	call pri			call pri		mov al,(bx)
	mov al,33h			pop ax		add bx,3
	call pri			ret		mov ah,(bx)
	mov al,8		prcon	endp		shl ah,1
	call pri		pri	proc near		shl ah,1
	xor dx,dx			push ax		shl ah,1
	xor ax,ax			push dx		push dx
p42:	call prcon			mov ah,0		mov dl,ah
	call pr640			mov dx,si		mov dh,0
	cmp ax,199			int 17h		mov ah,0
	jz p41			pop dx		add ax,dx
	add dx,2			pop ax		call pri
	inc al			ret		pop dx
	jmp p42		pri	endp		mov al,0
p41:	mov al,18h		read1	proc near		call pri
	call pri			push cx		mov al,0
	call crlf			push dx		call pri
	mov al,0fh			push si		pop bx
	call pri			push bx		pop cx
exit:	mov al,20h			mov ah,13		inc bx
	mov dx,20h			int 10h		dec cx
	out dx,al			mov ah,0		jnz oop2
	pop ax			push bx		pop cx
	pop dx			mov bx,ax		pop dx
	pop cx			shl bx,1		ret
	pop bx			add ax,bx		endp
	pop si			pop bx	read2	db 18 dup(0)
	pop di			lea si,gray	buf	push ds
	pop es			add si,ax	cccc:	push cs
	pop ds			mov cx,3		pop ds
	iret			rep movsb		mov ax,2505h
prts	endp			pop bx		mov dx,offset prts
pr640	proc near			pop si		int 21h
	push cx			pop dx		mov dx,offset cccc
	push ax			pop cx		int 27h
	xor cx,cx			ret		pop ds
p2:	call read2		read1	endp		endp
	cmp cx,639		gray	db 0fh,0fh,0fh	start	ends
	jz p1			db 06h,09h,06h	code	end start
	inc cx			db 04h,01h,04h		
	jmp p2			db 00h,04h,02h		

C 编程中易犯的错误

大连市辽宁师范大学海洋资源研究所
(116022) 苑丽霞

C 语言是一种结构式、模块式、编译通过的语言。它的数据类型丰富,运算符众多而灵活,控制流和数据结构新颖,书写风格独特,表达方式精炼,语言效率很高,既具有高级程序设计语言的功能、也具有低级汇编语言的特点。这些特色使得 C 语言日益流行起来,成为当今世界上最有影响的程序设计语言之一。但是 C 语言在实现这些特色的同时,也付出了较高的代价。为了提高灵活性,并提供紧缩而高效的代码、C 不得不引入一些含糊而难读的代码,以至给许多人以难写的印象。如何正确编制 C 语言,怎么避免一些易犯的错误,是每个 C 程序员都极为关心的问题。笔者根据近年来使用 C 语言、调试 C 程序的经验,总结了以下几点在编程中易犯的错误,并对它们进行了分析。

1. 指针类错误

指针是 C 语言的特色之一,也是程序员最容易出错的地方,即使编程老手有时也难免。归纳起来,指针错误主要是以下两类:

第一类:指针运算符(主要是取址操作符 & 和间接操作符 *) 的使用错误。引起该类错误的原因一般是对操作符的理解错误。取址操作符取得某一常量或变量在内存中的存储地址,间接操作符取得某指针所指向的某常量或变量的内容。这类典型错误如:

```
#include "stdio.h"
main()
{
char * p;
* p = (char *) malloc(100);
gets(p);
printf(p);
}
```

一旦运行,程序常会瘫痪,甚至破坏操作系统。原因在于 malloc() 返回的地址应赋值给 p,实际上却赋给了 p 所指的字符。若修正这一错误,应将原句中的:

```
* p = (char *) malloc(100);
```

改为:

```
p = (char *) malloc(100);
```

第二类:使用了“野”指针,指针在使用前首先要定义,然后要给指针分配合适的空间。指针和数组不同,数组在定义的同时,系统自动根据数组大小分配给相应的存储空间。数组的存储空间是由系统自动分配的,而指针却需由编程者确定其存储空间大小,并分配相应的空间。未经分配空间就直接使用的指针就称为“野”指针。上例中如无

```
p = (char *) malloc(100);
```

语句,则指针 p 就是“野”指针。“野”指针极难跟踪,程序有时运行很正常,有时又出错,并且错无定处,甚至导致系统瘫痪。据统计,程序很小时,即使有一、二个“野”指针仍能正常运行,因为程序只占用很少的内存,内存中数据被冲掉的机会不大。而当程序很大时运行往往不正常,特别是当使用了小内存模式时,数据段,代码段占用同一个 64K 段,内存空间有限,“野”指针常常破坏程序代码段,从而使系统瘫痪。

由于很难直接使用 640K 以外内存,所以编写程序时,必须及时回收分配出去的内存空间。这也容易导致“野”指针的出现。分配给某一指针的空间在用 free() 函数回收后,指针又被重新使用,这就导致与上节所说相同的后果。所以,内存空间的回收必须选择在正确的时机。

发生指针类错误时,程序出现的地方没有规律。执行起来有时正常,有时不正常,甚至还会出现一些变量无缘无故十分杂乱的现象。这时应彻底检查所有指针,跟踪指针引用的变量。这是一种有效的调试指针类错误的方法。指针虽然容易带来麻烦,但它是 C 最有用的方面,不管怎么样仍值得一用。

2. 边界错误

C 语言为提供简短、高效的代码,它的许多标准库函数很少甚至不做运行时的边界检查,所以很容易发生数组、文件或变量(通过指针赋值)写过头的错误。例如:

```
#include "stdio.h"
main()
{
char s[10];
int var;
var = 10;
printf("%d\n", var);
get-str(s);
printf("%s %d", s, var);
}
get-str(string)
char * string;
{
register int t;
printf("input 20 char:\n");
for (t = 0; t < 20; ++t) {
* string++ = getchar();
}
}
```

这段程序是将从键盘上输入的 10 个字符显示在屏幕上,但实际上读入了 20 个。为了显示过头后果,程序在字符串定义后又定义了一个整数。从运行结果可以看到:虽然 20 个字符正常显示,但整数的值却遭到了破坏,原来的数值被输入的第 11 个字符代替。文件或变量过界时,其本身一般不会异常,却往往导致其它地方出错。这种错误是十分危险的,并且常常将程序员引向歧途。

3. 内存溢出错误

由于 C 语言有很好的进程管理功能,所以,从理论上讲程序的总长度是不受限制的。内存溢出常常是编者处置不当所致。内存溢出只是一种现象,其根源往往是由于数组设置太大,单个可执行文件太长,或是太多太零碎的内存请求。

使用数组时,应记住数组占用的空间是很大的,例如,说明一个三维双精度数组:

```
double data[50][50][50]
```

它要求总存储区为 $50 \times 50 \times 50 \times \text{size of (double)}$,即 1,000,000 个字节。使用数组时应尽量压缩数组规模,并且在不使用时,立即释放,应尽量用多次定义、释放小型数组代替一次定义大型数组。

单个可执行文件太长引起内存溢出,解决办法是将程序合理分解,并利用 C 语言提供的进程调用和参数传递机制,将它们有机地管理起来。具体地说,将原程序分成许多独立执行小模块,当需要执行时再从磁盘上读进内存,并传给所需的参数让它运行。内存有限时甚至可以覆盖前一模块的存储区。

C 对于内存空间的利用分别通过 malloc() 和 free() 两个函数,采用分配和释放回收机制。尽管 C 对内存动态存储管理作出了很大努力,但 malloc() 和 free() 两函数仍然会带来内存碎片问题,其结果是程序运行一段时间后,虽然内存中有许多空闲区,但它们却夹杂在已分配掉的各内存块之间。当剩下的碎片单独不能满足内存分配请求时,就发生内存溢出错误。这种现象在频繁的很小的内存请求后尤其容易出现。内存碎片现象是难以消除的,也许将几个小的内存分配请求合并成一个大的请求,避免非常小的碎片出现可以使

问题得到缓解。另一种做法是在程序运行过程中随时将所有信息写到临时文件中去,收回所有内存,然后将文件读回内存。这种做很有效,因为读回来的信息将由动态存储管理系统在内存中连续存放。

4. 其它错误,请读者注意下面几条:

(1) switch 语句中不要忘了 break。break 在 switch 语句中用来结束一 case 子句,若 case 子句中没 break,程序将继续执行下一 case 子句。

(2) 数组下标始于 0,而不是 1。

(3) 不要将等号(==)和赋值号(=)混淆。等号是将符号左右两边的数值进行比较。若相等,整个表达式为真,返回 1,否则为假,返回 0。赋值号是将右边表达式的值赋给符号左边的变量,并且整个表达式亦为该值。这种错误常发生在用惯了 Pascal 语言的程序员身上。

(4) 路径名错误。斜杠(\)在 DOS 中表示一个目录名,而在 C 中,却作转义符。在 C 中目录名用双斜杠表示,如表示 C 区中根目录 TC 子目录下名为 mydata. - . dat 的文件应写成:

```
"C:\\TC\\mydata. dat"
```

而不应写成:

```
"C:\TC\mydata. dat"
```

(5) 注意防止无限循环。

(6) 将一负数赋值给无符号整数。

(7) 字符串结束要有空字符 '\0'。字符串后面肯定有空字符 '\0',它表示字符串结束。只不过有些函数自动加上了。当字符串被一个字符一个字符地赋值时,必须注意不要忘了结束符。

C—WordStar 使用基础问与答

新潮计算机集团公司(610051) 朱大公

1. C—WordStar 有哪些功能?

C—WordStar 具有八大功能,这八大功能分别是:

1. 文本编辑 2. 打印文件 3. 运行 C—WordStar 以外的程序 4. 非文本编辑 5. 更换文件名 6. 拷贝文件 7. 删除文件 8. 退出 C—WordStar。对于文秘工作者和通常的用户来说,用得最多的是文本编辑和打印文件两种功能。例如,起草一份通知、撰写工作总结、作家创作小说、记者写稿件、学生写作文、中文打字练习、编写计算机应用程序等。对于计算机程序员来说,除了经常使用上述两种功能外,还经常使用非文本编辑功能,以编写由计算机应用程序读取的数据文件或按照某种计算机语言的特定语法编写源程序。当然退出 C—WordStar 这一功能是在任何情况下都要使用的。其余几项功能,相对来说使用得要少一些。因此,学习 C—WordStar 的使用,一般来说可以先把精力主要放

在学习文本编辑和打印文件这两大功能上。

2. C—WordStar 的工作过程是怎样的?

利用 C—WordStar 来处理需要打印输出的信息,毫无例外的都要经过编辑、存盘、打印这三个步骤,缺一不可。下面对这三个步骤作一简要说明。例如,要给友人写一封信,首先是调用文本编辑功能,在这一功能下输入信的内容,然后进行编辑、校对、设置打印格式等。第二步就是调用存盘命令,把已经编辑好的这封信存到磁盘上,生成一份磁盘文件。第三步才是调用打印命令在打印机上打印出这封信。由此可见,编辑、存盘、打印是获得一份打印件的三步曲。C—WordStar 不像某些编辑软件那样,即使不生成磁盘文件也能打印出所需的文本,这一点是值得初学者注意的。对于非文本文件,只要想获得打印件,也要经过上述三个步骤,如果不需要打印件,也一定要存盘,否则录入的信息要丢

失。值得一提的是,虽然在编辑文本的过程中打印该文本是不可能的,但却可以在编辑该文本的过程中打印另一份已编辑好的、并且已经存盘生成了磁盘文件的文本。这两份文本一定是由两个文件名标识的。

3. 在 DOS 提示符 C)符下,发出 WS WJ. WS

A: <CR>命令是什么含义?

本问题要从怎样启动 C—WordStar 说起。启动 C—WordStar 的方法可以分成三种。第一种是在 DOS 提示符 C)号下(设 C—WordStar 的系统文件已装入 C 盘),发 WS<CR>命令(<CR>表示按回车键),格式如下所示:

C)WS<CR>

此时屏幕上出现版权信息,过几秒钟后便会出现 C—WordStar 的起始命令菜单。该菜单列出了 C—WordStar 的八大功能。这是启动 C—WordStar 最简单,也是最常用的方法。使用这种方法的方便之处在于,文本存盘后,C—WordStar 返回到起始命令菜单,便于用户选择 C—WordStar 的其它功能。但其不方便处也是很明显的,要想进入最常用的文本编辑状态,也必须选择编辑文本文件的命令并键入文件名。第二种启动 C—WordStar 的方法是,在 DOS 提示符 C)号下发 WS 文件名<CR>命令,格式如下所示:

C)WS 文件名<CR>

其中文件名是用户指定的、符合规定的、可以包含盘符在内的一个文件名,如 WJ. WS、B: WJ. WS 等。采用这种方法启动 C—WordStar,可以越过 C—WordStar 的起始命令菜单而直接进入文本编辑状态。这种启动方法的优缺点,大致和第一种方法相反。上述两种方法,在文本存盘时均无法选择所生成的磁盘文件的存放位置。就是说,启动 C—WordStar 时,原文件在哪个驱动器的盘上,存盘时仍然存入哪一个驱动器的盘上,该驱动器由文件名前的盘符指定,即使所编辑的文本是第一次命名的新文件也如此。显然这在某些场合是不方便的。此时我们可以选择第三种方法来启动 C—WordStar。这种方法的作法是在 DOS 提示符 C)号下发 WS 文件名,盘符<CR>,格式如下所示:

C)WS 文件名 盘符<CR>

其中文件名的含义与第二种方法相同,进入 C—WordStar 的屏幕显示也与第二种方法相同,其不同点在于,存盘时所生成的磁盘文件存放在命令中盘符所指定的驱动器的盘中。这为存盘前更换磁盘提供了方便,同时也增加了已录入数据的安全性。到此为止,我们可以明确地回答问题中所示命令的完整含义了,即从 C 盘启动 C—WordStar,编辑 C 盘上的现存(或新的)文件 WJ. WS,编辑完成后存盘时,存入 A 驱动器的磁盘上,文件名为 WJ. WS。现将三种启动 C—WordStar 的方法小结如下:

- (1)C)WS<CR>
- (2)C)WS 文件名<CR>
- (3)C)WS 文件名 盘符<CR>

4. C—WordStar 的命令分几类? 每类命令怎样操作?

DOS 支持下的 C—WordStar 与用户的接口为字符方式,用户调用 C—WordStar 的各种功能的方法,基本上都是通过键盘键入命令,因此弄清楚 C—WordStar 的命令类型和各自的操作方法是十分必要的。C—WordStar 的命令分成 4 类,分别称为单键命令、双键命令、三键命令和点命令。

单键命令用一个英文字母表示,称为命令字母。单键命令的操作方法十分简单,只要按下相应功能对应的字母键即可。例如,在启动 C—WordStar 进入起始命令菜单后,用户可以从显示器屏幕上看到 C—WordStar 的八大功能,每一功能项前有一英文字母,这就是命令字母。在文本编辑功能前有一字母 D,如想进入文本编辑功能,按下键盘上的<D>键就可以了。

双键命令的表示方法为 ^ 后紧跟一个英文字母,如 ^ D(有的资料写为 Ctrl+D)、^ J 等。命令中的 ^ 表示 Ctrl 键,其操作方法是用左手按下键盘上的<Ctrl>键,不要放手,再用右手按下命令中出现的字母键,如<D>键、<J>键,等。在 Ctrl 键与命令中出现的字母键位置距离较近时,亦可用单手操作,但操作方法一定是先按下<Ctrl>键,不要放手,再按下命令中出现的字母键。现在有的资料把双键命令的操作方法解释为同时按下<Ctrl>键和命令中出现的字母键,照此操作往往出错,使初学者迷惑不解。这种解释是错误的,至少是不严密的。虽然操作者力求做到同时按下两个键,但由于键盘内的单片微处理器具有极高的处理速度,所以看起来是同时按下了两个键,可事实上仍然是一前一后地按下两个键。这样一来,主机接收到的键盘扫描码肯定是一前一后。当命令中出现的字母键的扫描码先于 Ctrl 键的扫描码出现时,其外在表现就是操作出错。

三键命令的表示方法是 ^ 后紧跟两个英文字母,如 ^ KD、^ KS 等。三键命令的操作方法是用左手按下键盘上的<Ctrl>键,不要放手,用右手按下紧跟其后的第一个字母键,然后两手放开,再按下第二个字母键。

点命令是较为特殊的一类命令,其表示方法是以句点“.”打头,后跟两个字母,再跟一个参数,如“PL40”、“MT6”、“PO”等。对某些点命令,参数是可选的;对某些点命令,参数是必选的;而对某些点命令,没有参数。点命令的使用方法与上述三类命令都不同,不是通过使用者的操作从键盘上输入命令,而是把点命令按一定的规则输入到文本中。具体的规则主要是,第一,点命令总是放在文本中,占据一文本行,形成一个段落。也就是说,一个点命令所在行的上一行以硬回车结束,点命令所在的行也以硬回车结束;第二,点命令必须从一行的最左列开始存放,就是说句点“.”必须放在文本行的第一列,当文本的左边界不在第一列时,也必须保证点命令从最左边的第一列起开始存放;第三,每一点命令都有确定的语法规定,使用时应遵守这些语法规定。

5. 何谓文本和非文本? 何谓文本文件和非文本文件?

文本和非文本是 C—WordStar 中的两个重要概念。初学者往往从待输入的信息的外观形式来加以区分,这是不全面的。也有的认为文本是以 ASCII 码存储的,而非文本是以二进制代码存储的,这种理解也不符合实际。事实上,这是 C—WordStar 内部对用户输入的信息(或说资料、或说数据)的两种不同处理方法。对于一份资料,既可以认为它是文本,也可以认为它是非文本,完全取决于 C—WordStar 对这份资料如何处理。对用户来说,如果选取文本编辑功能(D 命令)来处理这份资料,则这份资料就是文本,存盘时生成的文件就是文本文件;如果用户选取非文本编辑功能(N 命

令)处理这份资料,则这份资料就是非文本,存盘时生成的文件就是非文本文件。那么,一份资料到底该选用哪种编辑功能来处理呢?这完全取决于使用者怎样使用这份资料。一般情况下,供数据交换使用的数据文件和某些计算机语言源程序,应该用非文本编辑功能来处理,而一般的文章,如工作总结、通知、作家创作的小说等就应该用文本编辑功能来处理。在非文本编辑功能下,许多字处理功能被取消了,如没有右边界、改编段落不工作等。更多的信息请查阅有关专著。总之,在文本编辑功能下处理的资料不能当成非文本使用,反之亦然。然而十分有趣的是,在文本编辑功能下处理的资料可以转换成非文本。具体方法就不在此详述了。

(待续)

在 TANGO 软件中建立汉字元件库

湖南津市湖南机电学校(415400) 肖安顺

在使用 TANGO 软件设计电路时,往往需要在电原理图上标注汉字。TANGO 软件不能直接对汉字进行处理,在分析 TANGO 软件的元件库后,我用 BASIC 语言编写了一个程序,使用它可以一次将一、二级字库中的所有汉字和图形符号转换成 TANGO 汉字库源文件,通过汇编,可以转换成 TANGO 汉字库源文件解决了在电原理图上标注汉字的问题。

程序说明:

25~30 语句,打开汉字点阵代码库文件。程序使用 WPS 文字处理系统的汉字库,如使用其他汉字系统的软汉字库,只需将 25 语句中的文件名改为相应的文件名即可。

120~185 语句,生成 TANGO 汉字库文件;210~630 语句,将一个汉字的点阵生成汉字库源文件代码;其他语句读者可自行分析。

使用方法:

1. 运行本程序,将会在 TANGO 软件的 SCH 目录下自动生成 CHINESE0. SRC, CHINESE1. SRC, …… ,CHINESE8. SRC 等八个文件。0~9 区的符号在 CHINESE0. SRC 中,16~19 区的汉字在 CHINESE1. SEC 中,20~29 区的汉字在 CHINESE2. SRC 中,以此类推。

2. 进入 TANGO 软件的 SCH 目录,执行 COMPILE 文件,将上述八个汉字库源文件逐一汇编成扩展名为“.LIB”的元件库文件。至此,汉字库中的汉字可以象其它元件一样被使用。

3. 使用汉字元件库时,只需将汉字的区位码输入即可。

由于本程序所处理的汉字较多,花费的时间较长,建议在编译 BASIC 中运行。本程序在 386/40 的计算机上用 Turbo BASIC 约需 10 分钟。

10 OPTION BASE 1;DIM Z%(8)

```
15 FOR I%=1 TO 8:READ Z%(I%);NEXT I%
20 DATA 1,2,4,8,16,32,64,128
25 OPEN "C:\XSDOS.LPH" FOR RANDOM AS #1 LEN=
32
30 FIELD #1,32 AS A$
35 OPEN "C:\TANGO\SCH\CHINESE0.SRC" FOR RAN-
DOM AS #2 LEN=28
40 FIELD #2,28 AS B$
50 D%=1
60 FOR I%=1 TO 87
70 IF I%>10 AND I% <=15 THEN 650
75 PRINT I%
80 C$="0"+CHR$(I%+48);S%=I%
90 IF I% >=16 THEN C$=RIGHT$(STR$(I%),2);
S%=I%-6
100 FOR J%=1 TO 94
104 IF J% <10 THEN D$="0"+CHR$(48+J%)
108 IF J% >=10 THEN D$=RIGHT$(STR$(J%),2)
110 IF NOT (I%=16 AND J%=1) THEN 150
120 CLOSE #2
130 OPEN "C:\TANGO\SCH\CHINESE1.SRC" FOR
RANDOM AS #2 LEN=28
135 FIELD #2,28 AS B$
140 D%=1;GOTO 200
150 E%=I%/10
160 IF NOT (E% * 10=I% AND J%=1) THEN 200
170 CLOSE #2
180 OPEN "C:\TANGO\SCH\CHINESE"+CHR$(48+
E%)+".SRC" FOR RANDOM AS #2 LEN=28
185 FIELD #2,28 AS B$
190 D%=1
200 GET #1,(S%-1) * 94+J%
210 LSET B$=" "+C$+D$+"POWER " +
CHR$(13)+CHR$(10) 'POWER 之后为 14 个空格
220 PUT #2, D%;D%=D%+1
```

```

230 LSET B$="2 2 1          "+CHR$(13)+CHR
    $(10) '1 之后为 21 个空格
240 PUT #2,D%:D%=D%+1
250 LSET B$="BITMAP          "+CHR$(13)+
    CHR$(10) '20 个空格
260 PUT #2,D%:D%=D%+1
270 LSET B$="{0}....."+CHR$(13)+CHR
    $(10)
280 PUT #2,D%:D%=D%+1
290 LSET B$="{1}....."+CHR$(13)+CHR
    $(10)
300 PUT #2,D%:D%=D%+1
310 FOR E%=1 TO 16
320 F$=""
330 FOR L%=1 TO 2
340 M%=(E%-1)*2+L%
350 E$=MID$(A$,M%,1)
360 N%=ASC(E$)
370 FOR O%=1 TO 8
380 Q$=""
390 IF Z%(9-O%)<N% THEN Q$="#":N%=N%-
    Z%(9-O%)
400 F$=F$+Q$
430 NEXT O%
440 NEXT L%
520 H$=""
530 IF E%<9 THEN H$=H$+" "+CHR$(49+E%)
540 IF E%>=9 THEN H$=H$+" "+RIGHT$(STR
    $(E%+1),2)
550 H$=H$+".."+F$+"...."+CHR$(13)+CHR
    $(10)
560 LSET B$=H$:PUT #2,D%:D%=D%+1
570 NEXT E%
580 LSET B$="{18}....."+CHR$(13)+
    CHR$(10)
585 PUT #2,D%:D%=D%+1
590 LSET B$="{19}....."+CHR$(13)+
    CHR$(10)
595 PUT #2,D%:D%=D%+1
600 LSET B$="{20}....."+CHR$(13)+
    CHR$(10)
610 PUT #2,D%:D%=D%+1
620 LSET B$=""          "+CHR$(13)
    +CHR$(10)
630 PUT #2,D%:D%=D%+1
640 NEXT J%
650 NEXT I%
660 CLOSE
670 END

```

病毒一例

成都量具刃具总厂量研所(610056) 谭洪勇

最近,我发现了一种新颖的引导型恶性计算机病

毒,用 KILL20, KILL43, SCAN(检测 79 种病毒), CPAV 等软件均不能发现它。因此,特向贵刊投稿,以介绍给计算机工作者,以免减少损失。

该病毒概述如下:

1. 将病毒程序主体与原引导扇区内容一起,对 360KB 软盘写到 1 头 39 道第 8、9 两扇区,对 1.2MB 软盘写到 1 头 79 道第 14、15 两扇区,对硬盘写到 0 头 0 柱 2、3 两扇区。然后只修改原引导扇区中的 52 字节,并写回原位置。

2. 判断感染标志 FB A1 13 04 48 48 A3 13(引导扇区偏移 40H),若无此标志将感染。

3. 由于该病毒未修改 FAT 表,若数据覆盖了病毒主体与原引导扇区,软盘不能启动。对硬盘而言,若隐含扇区只有 1 个,那么将导致破坏 FAT 表,须立即修改 FAT 表并消除它。

4. 病毒驻留内存后,内存减少 2K,并修改 INT 13H 中断向量,使其指向病毒驻留的高端地址,例如 640K 内存,则病毒从 9F40:0000H 开始存放。

5. 病毒发作时,无任何显示,只是将 ROMBIOS 最后 4K 内容反复写入每一磁道的 0 头与 1 头的前 8 扇区,若不关机,将一直进行到发生读写溢出错为止。病毒发作条件:定时计数低字(0:46CH)为零且调用 INT 13H 的 3 号写盘功能时发作。

该病毒的消除:

用未染病毒的引导扇区覆盖,或将原引导扇区写回即可。

“3.6”一种超级恶性病毒

广州华南师范大学 89 级(510631) 李建俊

近期,笔者用公安部的 SCAN 3.0 检测一软盘时,发现一种名叫“3.6”的病毒。根据本省公安厅调查,它是由台湾某公司出售的 286 机所带的系统盘传入我省的。它侵占 DOS 引导区并因此在 DOS 系统自举时获得控制,属操作系统病毒。

和目前已知的病毒相比,它的表现十分凶悍。平时除感染 DOS 系统的软盘(只在 A 驱感染,对 B 驱无效)和硬盘外,还常常破坏 DOS 系统区,造成文件的丢失。当系统日期为 3 月 6 日时会以随机字符重写已染毒的盘。使所有数据丢失(包括引导区)。发作时并不显示任何信息,只是从头到尾不断写盘。

经过详细分析,发现它与大麻病毒(石头病毒)十分相似,可能是同出一源。用工具软件检查病毒盘,会发现原 BOOT 区被病毒程序取代。这部分程序简洁精炼,集引导部分、传播部分、破坏部分为一体,有效长度为 1BEH 字节,不足一扇区。程序开始 4 个字节为“E9AC00F5”,即感染标志。病毒程序占据 BOOT 区后,将原引导扇区移到磁盘上某固定位置。对于硬盘,这个位置是 0 面 0 柱面 7 扇区;软盘则分两种情况:

360K 的是 1 面 0 道 3 扇区;1.2M 高密盘是 1 面 0 道 15 扇区。显然,如果这个位置原来有数据,则会因 DOS 引导扇区的写入而造成数据丢失。对 DOS 3.0 以上版本,分区的硬盘感染后无明显“症状”,仅作为一个传播毒源;若是 360K 双面倍密软盘,1 面 0 道 3 扇区为逻辑 11 扇区,位于根目录表的最后一扇区。只有在目录项大于 96 才用到这个扇区,因此只在文件个数大于 96 时感染才会造成文件丢失。若是 1.2M 高密盘,1 面 0 道 15 扇区为逻辑 29 扇区,同样是根目录表最后一扇区,只在目录项大于 208 才用到这个扇区,所以在一般情况下是比较难发现的。

3.6 病毒与“大麻”不同之处在于其对 1.2M 软盘感染动作和其发作动作。它改进了“大麻”那种对 1.2M 高密盘也是将原 BOOT 引导区写到 1 面 0 道 3 扇区的做法,增加了病毒的隐蔽性。另“3.6”病毒的发作对用户是一个致命的打击——所有数据在顷刻间灰飞烟灭。故广大微机用户应警惕,查清自己的盘是否染上这一可怕的病毒。

下面简单介绍一下检测和消除 3.6 病毒的方法,对软盘可在 DEBUG 下键入:

-D 100 0 0 1

-D 100 可见到开头四个字节为 E9AC00F5

-D 246 可见到 0603

这样就可以说明该盘已染上 3.6 病毒

对硬盘可用 DEBUG 编一个小汇编读取主引导扇

区:

-A 100

```

××××:0100  MOV DX,0080
                MOV CX,0001
                MOV BX,0200
                MOV AX,0201
                INT 13

```

-G=100

-D 200

-D 346

若见到与上述软盘的内容相同则已染上病毒。解毒办法是将被搬走的正确引导区搬回 0 道 0 面 1 扇区。

①对软盘

a)360K 软盘

```

-L 100 0 1      ;读 0BH 扇区正常引导扇区内容
-D 100 2FF      ;查看是否为 DOS 的引导记录
-W 100 0 0 1    ;将正常引导记录写入 0 区 0 道 1 扇

```

b)对 1.2M 软盘

```

-L 100 0 1D 1  ;读 IDH 扇区正常引导扇区内容
-D 100 2FF      ;查看是否为 DOS 的引导记录
-W 100 0 0 1    ;将正常引导记录写入 0 区 0 道 1 扇

```

区

②对硬盘

由于硬盘 0 面 0 道 1 扇区为主引导扇区,不能用 DEBUG 或 PCTOOLS 直接看到和操作。

要进行如下操作:

-A 100

```

××××:0100  MOV DX,0080
                MOV CX,0007
                MOV BX,0200
                MOV AX,0201
                INT 13
                MOV DX,0080
                MOV CX,0001
                MOV BX,0200
                MOV AX,0301
                INT 13
                INT 3

```

任意数任意次整数幂的高精度运算程序

山东苍山县酒厂(277700) 张新莲

传说古代印度国王舍罕在奖赏他的宰相——国际象棋发明者达依尔时,曾问他想要什么样的赏赐。达依尔回答说:“金银珠宝我不爱,只要在棋盘的 64 个格子里放满小麦,臣就感恩不尽了。我要求第一格放一粒,第二格放两粒,以后每格放的粒数是前一格的两倍”。“要小麦,这不难。”国王答应了他的请求。结果整个印度古国的全部小麦用尽,也没有填满棋盘的格子。封建王朝几千年,据说谁也没有算清这笔帐。

首先我们看一下这个问题的算式。设总数为 Y,则有

$$Y = 1 + 2 + 2^2 + 3^2 + \dots + 2^{63} = \sum_{N=0}^{63} 2^N$$

用数学归纳法可求得

$$\sum_{N=0}^{63} 2^N = 2^{64} - 1$$

为求出结果,可编一简单程序输入微机计算。

```

10  Y=2^64-1
20  PRINT "Y=";Y
30  END

```

或直接使用输出命令

```
PRINT 2^64-1
```

结果是 Y=1.8446741E+19

许多文献都介绍了这一计算结果,可见是正确的,但是很不精确(小数点后舍去了 11 位有效数字)。使用

本刊 1988 年第 6 期《计算 M 的 N 次方》(简称原程序 A)介绍的程序,可很快算出精确答案。

$Y=18,446,744,073,709,551,615$

这无疑是最精确的结果了。

但是,如果达依尔宰相要求第二格不是放两粒小麦,而是 1234 粒,以后每格放的粒数是前一格的 1234 倍,那么,棋盘第 64 格的精确数字是多少?

计算这一结果,直接使用输出命令 PRINT 已不能运行,原程序 A 仅能对单位数求幂,也不能使用。本刊 1989 年第 6 期介绍的《一个可以精确求解幂运算的 BASIC 语言程序》(简称原程序 B),虽然可以求多位数的整数幂,但最高只能求到 $(1234)^{12}$ 次幂(因为 1234 的 13 次幂的数字位数已超过 10^{38} ,将出现溢出),且运算速度太慢,也不能求出这一问题的答案。那么怎样得出这一问题的答案呢?本文介绍的程序一可以解决这一问题。

程序一:任意整数任意次整数幂的高精度运算程序

```
1 REM "No. 1 QIU M ^ N"
5 HOME: CLEAR: INPUT "M ^ N="; M, N
10 DIM A(500): A(1)=1: Z=1: C=0
20 FOR I=1 TO N: PRINT "TIME="; I,
30 IF C>Z THEN Z=C
40 FOR J=1 TO Z: A(J)=A(J)*M: NEXT
50 FOR J=1 TO Z
60 IF A(J)>9 THEN B=INT(A(J)/10): A(J)=A(J)-10
  * B: C=J+1: A(C)=A(C)+B: J=C: GOTO 60
70 NEXT: NEXT
80 PRINT: PRINT M; " ^ "; N; "= ";
90 IF C>Z THEN Z=C
100 FOR I=Z TO 1 STEP -1: PRINT A(I);: NEXT
110 END
```

程序一的设计思路是定义一个装积数组,把运算的乘积依次装入数组内,每个下标变量装一位数据,然后分别与底数相乘,相乘之积仍存于对应下标变量中。一次乘法运算结束后,从低下标变量开始,依次向高位进位,以保证每个下标变量始终存放单位数。这样,较好地解决了溢出问题,达到高精度任意次整数幂运算的目的。

程序一中,10 行的 A(X) 数组下标 X,是根据待求结果的位数而定的。对于 Apple、CEC-I 机最大可设定 17000 位,对于 GCS-90 等整数 BASIC 机种可设定 255 位,即最大可分别进行结果为 17000 位、255 位数字的运算。(当然要在微机内存容量允许的情况下)。40 行实现乘法运算。50~70 行完成各下标变量的进位操作。80~100 行为打印输出程序。

运行程序一,输入 1234,63 可得 195 位数字的答案。

$(1234)^{63}=566,050,346,952,017,995,166,613,438,401,027,137,694,958,805,851,818,332,169,765,109,596,094,360,330,103,818,849,734,022,478,520,218,879,430,334,366,879,719,513,413,054,451,635,198,792,905,644,908,945,294,973,720,$

457,048,457,573,987,897,328,626,156,913,557,504

说明:程序一基本上可适用任何机型对整数求任意次整数幂的运算。对于只使用整数 BASIC 的机型,最大可求 3000 之内整数的任意次幂。对于使用浮点 BASIC 的机型可求出 10^8 之内任意整数的任意次整数幂。若微机内存容量不够,可把实型数组 A(X) 改变为整型数组 A%(X),可使数组占用内存减少 60%。

假如达依尔要求小麦不是按粒数,而是按重量放置,那第一格放 5.678 克,第二格是前一格的 5.678 倍,求棋盘的 64 格放的小麦是多少克。欲求这一问题的答案,以上程序已无能为力,必须另辟蹊径。

求幂过程实质上是对一个确定数的连乘过程。只要把任意数的高精度乘法程序进行改造,就可实现任意数任意次整数幂运算。改造方法是(参见《任意数的高精度乘法》一文)把第一次运算后的乘积由 A(X) 数组转存到被乘数数组 B(X) 中,再与放入 C(X) 数组的底数相乘,所得之积再由 A(X) 转移到 B(X)……,如此反复进行,一直到指数规定的运算次数,最后输出计算结果。由于改动较大,单凭文字说明不够清晰,特列出程序清单。

程序二 通用幂运算程序——任意数任意次整数幂的高精度运算

```
1 REM "No. 2 SUPER M ^ N"
10 HOME: CLEAR: DIM A(1000), B(1000), C(20)
15 INPUT "M ^ N =?"; C$, N: PRINT "PLEASE WAIT!"
20 C=LEN(C$): W=2: E$=C$
25 FOR J=1 TO C: IF MID$(C$, J, 1)=". " THEN F=J: GOTO 35
30 NEXT: GOTO 40
35 C$=LEFT$(C$, F-1)+RIGHT$(C$, C-F): U=C-F: C=C-1
40 FOR J=1 TO C: IF MID$(C$, J, 1)="0" THEN NEXT
45 C=C-J+1: C$=RIGHT$(C$, C): G=2*C: B=C: H=2*U
50 FOR J=1 TO C: C(C-J+1)=VAL(MID$(C$, J, 1)): B(C-J+1)=C(C-J+1): NEXT
55 FOR K=1 TO C: X=K
60 FOR J=1 TO B: D=B(J)*C(K): A(X)=A(X)+D
65 IF A(X)>9 THEN A(X+1)=A(X+1)+INT(A(X)/10): A(X)=A(X)-INT(A(X)/10)*10
70 IF A(X+1)>9 THEN A(X+2)=A(X+2)+1: A(X+1)=A(X+1)-INT(A(X+1)/10)*10
75 X=X+1: NEXT J
80 NEXT K: W=W+1: IF W>N THEN 105
85 IF A(G)=0 THEN G=G-1
90 FOR I=G TO 1 STEP -1: B(I)=A(I): A(I)=0: NEXT
95 B=G: G=B+C: H=H+U: PRINT "TIME="; W,
100 GOTO 55
105 IF A(G)=0 THEN G=G-1: GOTO 105
```



```

110 PRINT:PRINT E$“^”N“=”；
115 IF G<H THEN G=H
120 IF G=H THEN PRINT “0”；
125 FOR I=G TO 1 STEP -1:IF I=H THEN PRINT
“.”；
130 PRINT A(I);:NEXT
135 PRINT:PRINT “JI XU MA? (Y/N)”；:GET Y$ :IF
Y$ <>“N” THEN 10
140 HOME:VTAB 10 :HTAB 10: PRINT “GOOD BY
!!!”:END

```

说明；

1. 20~50 行的功能是对底数进行去小数点、整形，把底数分别装入 B(X)、C(X) 数组中，W 为循环控制变量，U 为小数底数的尾数长度变量，H 为幂的尾数长度变量。每运算一次，W 加 1，H 加 U。

2. 55~80 行为运算程序。

3. 85~100 行为本程序的关键部分。它完成了 N...1 次幂数据的转移，乘积数组 A(X) 的清零，幂尾数的计算及循环结束的转移。

4. 105~140 行为幂的输出和结束处理程序。

程序二可以在微机内存容量允许的情况下精确求解任意数(整数、小数)任意次的整数幂(注意根据底数和幂的位数适当修改 10 行的 A(X), B(X), C(X) 三数组的下标)。现在，运行程序二，我们可以计算第 64 格有多少克小麦了。 $(5.678)^{64} = 1, 855, 687, 507, 780, 543, 495, 217, 331, 136, 461, 814, 294, 200, 934, 282, 131. 997453322155750406755740979367341832284057 1770755347346713067175211855461781077205214773 4701982368388061514840288791883164034032674583 8964665649600919329749732655671628220177229241 479180845056$

结果整数部分有 49 位，尾数部分有 192 位，共有 241 位。由此可见，程序二的幂运算能力是极强的。

下面讨论一下程序的运行速度。同程序一相比，程序二的运算速度要慢一些，但比原程序 B 要快得多。在 CEC-I 上分别运行原程序 B、程序一、二计算 $(12345678)^4$ (因程序 B 最多只能求 4 次幂，所以以求 4 次幂作比较)，原程序 B 运算时间为 79 秒；程序一运算时间为 4 秒；程序二为 30 秒。可见，程序一的运算速度最快，几乎是原程序的 20 倍，程序二次之，比原程序快 1.6 倍。三种程序运算结果相同，均为 $(12345678)^4 = 232, 305, 722, 879, 259, 244, 150, 093, 798, 251, 441$ 。若用幂运算符直接求幂，结果为 $(12345678)^4 = 2^{32305723E+32}$ 。

用程序一计算 $(123456789)^{50}$ 次幂，运算时间为 6 分 33 秒，用程序二计算时间为 70 分，两程序计算结果相同，为一个 355 位的数字。由以上对比可见，程序二的运算速度要比程序一慢得多，因此，当我们计算整数的任意次幂时(底数小于 10^8)，优先选用程序一；仅当计算大于 10^8 的数的幂或小数的任意整数幂时才考虑使用程序二。

作为本文的结束，再举两个实例，以证明两程序的运算能力。

例一. 计算 $(12345678)^{100}$ ，用程序一计算，运行时间为 26 分 12 秒，结果为 710 位数。(略)

例二. 计算

$(123456789876543212345678987654321)^4$ ，用程序二计算，运行时间 6 分 57 秒结果为 129 位。

您想验证一下吗？

高精度数值算法的改进

石河子市新疆农垦中专(832011) 邓阳春

《电子与电脑》1992 年第八期中刊登了陈庆祥同志的《对几种高精度数值计算方法的改进》一文。笔者读后认为：该文中的两个程序设计巧妙、清晰，称得上是短小精练，运行速度也令人满意，但两程序都值得商榷，尤其是序 1 还有一处出现了一个小错误，现分别探讨如下：

1. 因为一个 INPUT 语句最多能接受 239 个字符，所以用一个 INPUT 语句输入被乘数及乘数时，两数位数之和不能大于 238，大于 238 位的后面的数计算机视为无效，故建议用两个 INPUT 语句输入被乘数及乘数。将原程序的 20 语句改为：

```

20 INPUT“被乘数,乘数=”；:FOR J=1 TO 2:
INPUT A$(J):L(J)=INT((LEN(A$(J))
+3)/4):NEXT J

```

2. 因数组 B%(I)中的任意一个元素中的数，可能是一位、两位、三位、或四位，该程序只有在所有元素中的数都是四位时(最大下标那个元素除外)输出的结果才正确，否则结果是错误的，如：该文中举的一例
被乘数，乘数 = 369258147369258147147258369147，
147258369963852741123

乘积 = 5437635287746973012727426653072203
079638561332081

ProDOS 系统内部结构剖析(续)

北京铁路局卫生防疫站(100038) 廖凯

第五章 调用 MLI

本章将详细地介绍 MLI 在功能组成上的结构,怎样由机器语言程序调用 MLI,以及 MLI 如何调用它自身。

一、机器语言接口(MLI)

ProDOS 的 MLI 是机器语言程序和磁盘文件之间的完整的、一致的和可中断的接口,它完全不受 ProDOS BASIC 系统程序的影响,这样它用作一个写其它系统程序的基础。它由以下四部分组成:

命令处理程序:从一个机器语言程序接收和发送调用,它确认每个调用的参数,修改系统整体页面,然后跳到块文件管理程序的相应子程序。

块文件管理程序:执行所有有效的 MLI 调用,块文件管理程序记录所有安装的磁盘,管理所有打开的文件并进行简单的存储器管理。它经调用磁盘驱动例程进行所有磁盘存取(读和写)。

磁盘驱动例程:进行数据的读写,这些例程支持所有 Apple 计算机公司为 AppleII 制造的磁盘驱动器。

中断处理程序:ProDOS 提供四个中断处理程序,中断处理程序包含四个指向中断例程的向量。在一个中断发生时,这些例程依次被调用,直到被其中某一个中断认可。

二、制作 MLI 调用

一个程序经执行一个 JSR \$BF00(后面均称为 MLI)发送一个调用到 MLI。调用号和一个指向调用的参数表的两字节指针,必须立即跟在调用的后面。下面是一个调用 MLI 的例子:

```
SYSCALL JSR MLI      ;调用命令处理程序
DB CMDNUM           ;确定进行什么调用
DW CMDLIST          ;指向参数表的两字节指针
BNE ERROR           ;若非零则错误
```

当调用结束时,MLI 返回到 JSR+3 的地址(在上例中即 BNE 指令)。若调用成功,C 标志被清除,累加器被置为 0,若调用不成功,C 标志被设置,累加器被置为错误代码,寄存器状态在调用结束时概括如下:

寄存器标志	N	Z	C	D	V	Acc	PC	X	Y	SP
调用成功	0	1	0	0	×	0	JSR+3	无变化		
调用不成功	0	0	1	0	×	错误码	JSR+3	无变化		

注意:1. N—标志的值由累加器确定,因为所有当前定义的错误代码比 \$80 小,所以 N 总是零。

2. V—标志的值是未定义的。

后面是一个调用 MLI 的小程序,它试图在卷名 TESTMLI 上建立一个名为 NEWFILE 的文本文件。

若发生错误,计算机会发出警报声,并在屏幕上印出错误代码(记住使用一个以/TESTMLI 为名的格式磁盘)。

1. 参数表

每个 MLI 调用都有一个指向一个参数表的两字节指针,一个参数表包含由调用所使用的信息和为被调用返回的信息留的空间。在参数表中有三种要素:值、结果和指针。

值是一个被转到块文件管理程序(BFM)的单字节或多字节值。值帮助确定由 BFM 所采取的动作。

结果是 BFM 将一个值放置到参数表内一个或多字节空间。根据结果,程序可以获得有关一个卷、文件或中断的状态信息,或有关刚结束的调用的成功信息。

指针是一个指示数据、代码或一个 BFM 可以放置数据和接收数据空间的位置的两字节存储器地址。所有指针低字节在前,高字节在后。

在每个参数表中第一个单元是参数个数,用来指示被调用所使用的参数的数量(不包括参数个数)。这字节是用于检验调用是否是偶然的。

2. ProDOS 机器语言练习程序

有一个小程序 ProDOS Machine Language Exerciser,可以帮助你学习使用 MLI,它允许你由一个菜单执行 MLI 调用;它有一个十六进制存储器编辑程序用于观察和修改缓冲区的内容,并且它可以列印命令表。

在你使用 Exerciser 制作 MLI 调用时,需要用它的调用号调用,然后指定它的参数表。按 RETURN 键后调用被执行。

三、MLI 调用

MLI 调用可以分成三部分:内务调用,文件调用和系统调用。

1. 内务调用

内务调用执行诸如建立、删除和改名的操作,这些操作不适用于打开的文件,它可以改变一个文件的状态,但不能改变文件内的信息。在执行调用时,经路径名来引用文件,每个调用需要一个临时缓冲区。内务调用有:

CREATE:建立一个标准文件或一个目录文件。文件的登记项被放置在磁盘上相应的目录内。将磁盘上的 1 块分配给文件。

DESTROY:删除一个标准文件或目录文件。文件的登记项在目录内被删除,且所有文件在磁盘上的空间被释放。若删除一个目录,则该目录必须是空的。一个卷

目录不能被删除,重新对卷格式化除外。

RENAME:更改文件的名字。新名字必须在同一目录内代替原名字。这调用改变那个文件的登记项中的名字。若它是一个目录文件,则改变它在标题项内的名字。

SET_FILE_INFO:设置文件的类型、存取方式及修改的日期和时间。

GET_FILE_INFO:返回文件的类型、存取方式、文件的大小,建立和最后修改的日期和时间。

ON_LINE:返回槽口号,驱动器号和所安装的卷名。此信息放置在一个用户提供的缓冲区内。

SET_PREFIX:设置被操作系统使用的路径名为部首。此部首必须指出一个在卷上存在的目录。

SET_PREFIX:返回当前系统的部首。

2. 文件调用

文件调用会引起数据的传送。第一个调用 OPEN 必须用在其它调用之前。OPEN 调用以路径名来指定一个文件,其它的调用经 OPEN 调用返回的参考号来引用文件。另外,一个输入/输出缓冲区被分配给打开的文件,随后的数据传送通过这缓冲区。参考号和缓冲区被保留直到文件被关闭。文件调用有:

OPEN:为一个要被存取的文件作准备。这调用将分配给文件一个文件控制块(FCB),并返回一个参考号。另外,还分配一个输入/输出缓冲区。

NEW_LINE:为从文件读取设置条件。此调用通过读取一个特定字符(如一个回车字符)来结束读请求。

READ:从一个文件传送所需的若干字符到一个指定的缓冲区,并修改文件内的当前位置(MARK)。字符根据 NEW_LINE 调用的规定被读取。

WRITE:由一个指定的缓冲区传送所需的若干字符到一个文件内,并修改文件内的当前位置(MARK)和(必要时)文件的结尾(EOF)。(未完待续)

SOURCE FILE #01→TESTCMD

—NEXT OBJECT FILE NAME IS TESTCMD.0

```
2000;2000 1 ORG $ 2000
2000:      2 *
2000;FF3A 3 BELL EQU $FF3A;
2000;FD8E 4 CROUT EQU $FD8E;
2000;FDDA 5 PRBYTE EQU $FDDA;
2000;BF00 6 MLI EQU $BF00 ;ProDOS 系统调用
2000;00C0 7 CRECMD EQU $CO ;CREATE 命令号
2000:      8 *
2000:      9 MAIN JSR CREATE ;建立“/TESTMLI/
NEWFILE”
2003:     10 BNE ERROR ;若错误,显示它
2005:     11 RTS ;否则,结束
2006:     12 *
2006:     13 CREATE JSR MLI ;执行调用
2009:     14 DFB CRECMD ;CREATE 命令号
200A:     15 DW CRELIST ;参数表指针
200C:     16 RTS
200D:     17 *
200D:     18 ERROR JSR PRBYTE ;打印错误代码
2010:     19 JSR BELL ;发出警报声
2013:     20 JSR CROUT ;打印一个回车符
2016:     21 RTS
2017:     22 *
2017:     23 CRELIST DFB 7 ;7个参数
2018:     24 DW FILENAME ;指向文件名
101A:     25 DFB $C3 ;正常文件存取参数
201B:     26 DFB $04 ;制作一个文本文件
201C:     27 DFB $00,$00 ;辅助类型,没使用
201E:     28 DFB $01 ;标准文件
201F:     29 DFB $00,$00 ;建立日期(未用)
2021:     30 DFB $00,$00 ;建立时间(未用)
2023:     31 *
2023:     32 FILENAME DFB ENDNAME-NAME;
;文件名的长度
2024:     33 NAME ASC “/TESTMLI/NEWFILE” ;
;后面是文件名
2034:     34 ENDNAME EQU *
```

FORTH 语言讲座

第七讲 程序控制结构(下)

丁志伟

在上一讲中,介绍了 FORTH 语言中程序控制结构的概况、标志的概念和比较词的用法,还介绍了分支结构。在这一讲,将要介绍另一类程序控制结构:循环结构。其中包括固定循环、不定循环和无限循环结构。

先介绍固定循环。它又叫定循环或限定循环。

在编制程序时,经常遇到需要反复执行同一个动作的情况,比如现在要计算:

$1+2+3+\dots+99+100$

如果还想在每做一次加法的同时,把当时的累计值显示出来。可以从键盘打入:

```
1 2 +DUP .↵
```

显示出3。再执行

```
3 + DUP .↵
```

显示出6。可以一直做下去,直到

```
99 + DUP .↵
```

显示出4950最后

100 + .✓

显示出5050

这未免太麻烦了。对于计算机，这是大材小用。应该编出一段程序：

```

: SUM 1
  101 2
  DO I + DUP . CR
  LOOP DROP ;

```

执行SUM，就会在屏幕上显示出3、6、10……一直到5050

这段程序使用了固定循环、它的一般形式是：

```

( n1 n2 ) DO W LOOP

```

这个结构的主要性质是：

1. DO 和 LOOP 是循环两端的界限，它中间的 W 是不断循环执行的部分。

2. n₂和 n₁是循环的初始值和终止值。执行循环之前放在参数栈中。一般要求 n₁ ≥ n₂，循环次数是 n₁ - n₂

3. 如果想中途退出循环，就要执行 LEAVE 这个词。

4. 关于循环变量。执行 DO 时，系统会把 n₁和 n₂压入返回栈，把 n₂做为循环变量。执行到 LOOP 时，系统会把这个变量加1，并与 n₁的值做比较，二者相等时循环结束。如果想利用循环变量，就在 DO 和 LOOP 之间使用 I 这个词。执行 I 时，会把当时的循环变量复制出来放到参数栈上，在双重循环中，取外层循环变量则要用 J。

在上边的 SUM 中，n₁和 n₂分别是101和2，执行次数为101 - 2 = 99。I 的数值变化为2 ~ 100。执行 SUM 时，先在栈中存放一个1，然后再逐个加上2 ~ 100。并且不断把累计值显示出来。

下面看看循环的例子。

先定义一个词

```

: STAR ." * " ;

```

定义之后，执行 STAR 时会显示出 * 号。

接下来，想定义出显示5个 * 号的词，就可以

```

: 5STARS ( --) 5 0
  DO STAR LOOP ;

```

执行5STARS，会显示出 * * * * *

这个词固定显示出5个 * 号，功能有些死板。下面这个词可以显示出任意数量的 * 号。

```

: STARS ( n-- ) 0
  DO STAR LOOP ;

```

前边介绍，DO...LOOP 结构前边要有两个数。而这个词定义中只给出起始值0，另一个就要在执行之前给出。因此，想显示出6个 *，就执行

```

6 STARS ✓

```

下面这个词，显示出 FORTH 这个词的头一个字母“F”，是用 * 号拼成的。

```

: F* CR 5 STARS CR STAR CR
  STAR CR 4 STARS CR STAR CR
  STAR CR STAR CR STAR CR

```

SPACES 这个词，功能是显示出若干个空格，在某些版本中，它可能是这样定义的：

```

: SPACES ( n-- ) ?DUP
  IF 0
    DO SPACE LOOP
  THEN ;

```

?DUP (n--n n)或(0--0)栈顶数如果不是0，则复制此数。

执行 SPACES 的过程是，先检查栈顶数是否为0，如果是0，则退出；否则执行其中循环，显示出 n 个空格。

使用循环时，常会用到循环变量，DO---LOOP 中取循环变量的词是 I 和 J。

I (--n) 取循环变量

J (--n) 取外层循环变量

执行 DO 时，系统会把参数栈中两个数压入返回栈，执行 I 时，会把当时循环变量的数值从返回栈中复制出来，放到参数栈中。执行 J 时，则会把外层变量放到参数栈。

比如

```

: TEST0 0
  DO I . LOOP ;

```

执行 5 TEST0 ✓

会显示出 0 1 2 1 3 4

当 I 值为5时，循环已经结束，因此只显示到4。

另外请注意，PC/FORTH 中，显示数字之后系统会补上一个空格，而 D1.0 版本在显示一个数之后不加空格，因此显示效果不太一样。

接下来，利用 TEST0，再定义一个词：

```

: TEST1 6 1
  DO I TEST0 CR LOOP ;

```

这里 I 的变化范围是从1到5，TEST1 相当于 1 TEST0 2 TEST0 3 TEST0 4 TEST0 5 TEST0

执行 TEST1，就会显示出

```

0
0 1
0 1 2
0 1 2 3
0 1 2 3 4

```

也可以把两个词合为一个，就变为

```

: TEST01 6 1
  DO I 0
    DO I . LOOP CR
  LOOP ;

```

注意，这个词中有两个 I，是分别属于两个层次循环的。

再举一个例子：

```

: TEST2 6 1
  DO 6 I - TEST CR LOOP ;

```

这个词中 I 值是从1到5，6 I - 则是从5到1，执

行 TEST2,就显示出

```
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

下面这段程序,如果使用 D1.0 版本,其中 STACES 要改为 SPCS,它利用了前边的 STARS.

```
: TEST3 5 1
DO 5 I - SPACES
I 2* 1- STARS CR
LOOP ;
```

执行 TEST3,显示出

```
*
***
*****
*****
```

在以上例子中,如果用到循环变量,I 的值都是每次递增1,而在某些情况中,可能希望更灵活一些。下面这个结构就能解决这个问题。

```
( n1 n2) DO W (n3) +LOOP
```

这个结构与前边结构的差别,是把 LOOP 改为 (n₃)+LOOP。这里的 n₃是循环步长,可以是正数,也可以是负数。系统在执行+LOOP 时,要测步长,一旦超出限定值,就退出循环。

请看例子

```
: TEST 1 5
DO I . -1 +LOOP ;
```

TEST✓

显示54321

```
: TEST 6 0
DO I . 1 +LOOP ;
```

TEST✓

显示0123456

这里会将6显示出来,与 DO……LOOP 有些差别。

前边的 TEST2也可以改为 DO……+LOOP 的形式,请读者自己试一下。

在笔者编过的小系统中,有一个缓冲区,其大小是确定的,是用来存放被显示的字符串的,显示之前要先检测其终止位置。字符串后边都是空格字符,字符串之间也可能夹有字符。为可靠地找到最后一个字符,应该从后向前查找。程序可以这样编:

```
: TEST4 ( A1 A2)
DO I C@ BL (<)
IF I . THEN
-1 +LOOP ;
```

其中 BL 是空格符号,C@是从地址中取出一个字节。

程序从 A₂开始向回检测,直到非空格的字符,就是字符串的结尾,于是把这个地址显示出来。程序也可

以利用 LEAVE,中途退出。

```
: TEST5 ( A1 A2)
DO I C@ BL (<)
IF I . LEAVE THEN
-1 +LOOP ;
```

满足条件之后不再进行无效搜索,从而提高了执行效率。

前边介绍的定循环,要事先确定循环的执行次数,然而在实际情况中,有时难以做到这点,就要用下面将要介绍的不定循环了。

不定循环有两种,先介绍第一种。

```
BEGIN W (f) UNTIL (PC/FORTH)或
BEGIN W (f) END (D1.0版本)
```

这里两种形式的差别,只是在于不同版本使用不同的词名。

它的性质是:

1. BEGIN 和 UNTIL (或 END)是循环的起点和终点。

2. 不断执行其中的 W。

3. 当 f 为真时退出循环。

这种循环,在执行到 UNTIL (或 END)时,会测试标志,当 f 为真时会退出循环。这就需要在执行 W 时能够产生一个标志。

列出词典目录,在 PC/FORIH 中是 WORDS,在 D1.0 中是 LIST,就都使用了不定循环结构。由于实际版本中的词较复杂,为了便于说明问题,这里给出简化程序。先看 D1.0 版本的 LIST。

```
: LIST DICT @
BEGIN DUP ID. CR
@ DUP 0=
END DROP ;
```

这个词并不复杂,但想理解它却需要对词典结构具有较清楚的了解。先看其中两个词。

DICT 变量。其中存放词典中最后一个词的起址 ID. (a--) a 是一个词的起址,显示出这个词的词名。

词典中每个词的起址中存放着上一个词的起始地址,就是链表,不断地执行@,就可以把一个个词的起址取出来。第一个词的起始地址存放的是0,表示到此为止。END 前的 DUP 0=就用来判断是否取到第一个词。

再请看 PC/FORTH 中 WORDS 的简化词。

```
: WORDS LATEST
BEGIN DUP .NAME CR
N>LINK @ DUP 0=
UNTIL DROP ;
```

这里,LATEST 与 D1.0 中 DICT@相似,.NAME 与 ID. 相同。由于词条结构不同,因此要加上一条 N>LINK,用于把名域地址较换成链域。

这种功能,如果使用 DO……LOOP 结构来编,就比较困难了。

不定循环还有另外一种。

BEGIN W₁ (f) WHILE W₂ REPEAT W₃ ;
它的性质如下：

1. BEGIN 和 REPEAT 是循环的起始和终点。
2. 不断执行 W₁ 和 W₂。
3. 执行到 WHILE 时,当 f 为真时,就执行 W₂,然后执行 W₁。当 f 为假时,退出循环,执行 W₃。

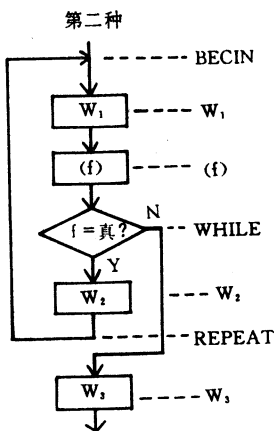
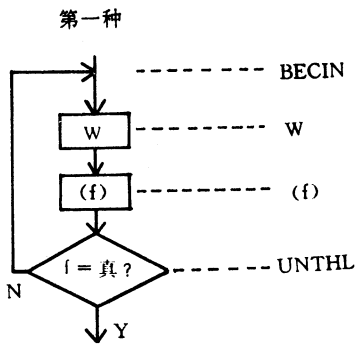
上边例子中的 LIST 和 WORDS,都可以改用这种不定循环结构来定义。以 LIST 为例

```

: LIST DICT
  BEGIN @ DUP
    WHILE DUP ID. CR
  REPEAT ;
  
```

这种不定循环,有时比前一种方式更为自然一些。这两种不定循环,有相通之处,也可以互相代替。下面看看它们之间的异同之处。

请看两种结构的框图。



1. 退出之前都要判断标志 f,但在循环中的位置不同。

2. 第一种结构当 f 为真时退出,而第二种 f 为假时退出。

3. 第一种中 W 至少要执行一次;而第二种的 W₂ 可能一次也不执行,或者说,它可以正确地执行 0 次迭代的情况,有时这是很重要的。

下面再介绍一种循环:无限循环,又叫死循环。它比不定循环和限定循环简单。

```
BEGIN W AGAIN
```

与上述各种结构不同,它不做任何判断,只是不断执行 W。

在某些应用程序,可能是这样编的

```

: 应用程序
  BEGIN 输入
    CASE 输入1 OF 应用1 END OF
      输入2 OF 应用2 END OF
    .....
  ENDCASE
  AGAIN ;
  
```

这种循环,一般是无法退出的。想要退出,需要执行 EXIT 这个词。比如 D1.0 版本中。

```

: TEST BEGIN 1 . INKEY $
  IF EXIT THEN
  AGAIN ;
  
```

INKEY \$ (--f) 检测是否有键按下。这个 TEST,不断在屏幕上显示 1,一旦有键按下,就会终止。

FORTH 系统的解释/编译器也用到了无限循环。这里介绍 D1.0 版本中的解释器,它比较简单,下面先用汉语把它的含义表示出来。

: 解释器

```

BEGIN 从缓冲区读入一个词,
      是词典中某个词?
  IF 执行这个词
  ELSE 试图转换成数字,有错?
    IF 错误处理
    THEN
  THEN
  AGAIN ;
  
```

再看其实际程序

```

: MAIN2
  BEGIN GETVERB
    IF SWAP DROP CFA EXEC
    ELSE NUMBER 0=
      IF DEFERR
      THEN
    THEN
  AGAIN ;
  
```

限于篇幅,就不做解释了。

这两讲中,介绍了 FORTH 中的控制结构。它们从

原理上与 Pascal 这类语言中相应结构的功能相同。但在 FORTH 语言中又有其自身的特点。

1. 各种控制结构都要放在冒号定义之中使用。如果放在冒号之处,将会发生错误。

2. 各种结构都有其明确的起始和终止界限,比如 IF……THEN、DO……LOOP

3. 同一结构中各成分必须配套出现。比如有 IF 就必须有相应的 THEN。

4. 因为每个结构起点终点位置明确,因此嵌套使用就非常方便。各种结构都可以互相方便地嵌套使用,如 MAIN2 这个词就有三个结构嵌套在一起。嵌套的层次数量是没有限制的。

5. FORTH 中结构的书写格式很自由,但最好要符合一定规范,以便于阅读。

关于格式,这里要多说几句。有时会有这样一种感觉:FORTH 程序不易阅读。这有一定道理,但事情有其另一面,程序是否易于阅读,与书写者素养有关。就好像同是写文章,水平却有高低一样。

比如有一个词,可以写成多种形式:

```
: TEST 5 0 DO I . LOOP ;
: TEST 5
  0 DO I
  . LOOP ;
: TEST 5 0 DO I .
  LOOP ;
```

还可以写成其他格式。不同的书写形式,对于阅读、交流和调试的效率有很大影响。一般使用缩格排列的方式。在 MAIN2 中, BEGIN 与 AGAIN 上下对齐, IF 和 THEN 上下对齐,表示它们对应关系。程序书写的原则是,程序结构应该能在格式上直观表现出来,一目了然。

最后,看看如果把 MAIN2 改为这种格式:

```
: MAIN2 BEGIN CETVERB IF SWAP
  DROP CFA EXEC ELSE NOMBER O=
  IF DEFERR THEN THEN AGAIN ;
```

阅读起来就较为吃力了。

初、中级程序员软件水平考试辅导

数 据 结 构

北京大学(10871) 夏晓东

计算机的发明至今已近半个世纪,就在这短短的半个世纪中,人类的社会发生了革命性的变化。计算机的快速、记忆量大、精确、实时和自动化的特征为我们的日常生活带来了无数便利。科学计算、数据处理、实时控制,都离不开计算机。无数的计算机应用也都是对描述万物的数据进行加工、处理的过程。而“数据结构”这门学科正是研究描述万物的数据在计算机中是如何表达、存储和加工的。

一、信息与数据

我们试图把大自然中的各种事物和问题用计算机表示,让计算机帮助我们解决各种复杂的问题,因此,为了概念上清楚,我们给出关于信息和数据的明确定义以便后文的讲解。

数据:是对现实世界的事物采用计算机能够识别、存储和处理形式所进行的描述,这里,我们强调数据是计算机能够识别,存储和处理的的东西,它毫无含义,无非是一系列符号。

信息:是指人类社会交往中产生的消息,可被表达为计算机可接受的数据。所以,信息是数据所表达的含

义。

以上的“有无含义”也是相对于我们人类而言的,而数据相对于计算机来说又是有含义的,因为它们能够被计算机识别、存储和处理。我们只有把“信息”转变为“数据”,人类的问题才能交与计算机去处理。如何实现这种转变,正是计算机语言、知识工程、机器识别等计算机科学的很多分支所要讨论的问题。然而,研究计算机如何理解、表示并有效地处理数据的学问就是“数据结构”。

当然,通常我们并不区分信息和数据,把它们都看成是计算机要处理的对象。

二、什么是数据结构

简言之,数据结构是指数据元素之间的关系。这里,数据元素是我们讨论数据结构的基本单位,它可以是一个记录,一个表目,我们把它统称为结点。

每当我们谈及某种待处理数据时,总是要考虑它是如何用计算机表达的,如何在计算机中存储并处理的,所以,我们认为任何一种数据结构应包括:数据的逻辑结构,数据的存储结构以及数据的运算三方面。

例1:某单位职工工资表如下,

编号	姓名	基本工资	副食补助	...	实发金额
1	张帆	113.00	25.00	...	142.70
2	李君祥	82.00	25.00	...	117.40
3	王聪	97.00	25.00	...	119.80
...
...
50	赵图强	132.00	25.00	...	170.50

其中,每位职工在工资表中占据一行,每一行都说明了相应职工的工资情况。我们可把整个工资表看作是一个数据结构,而把表中的每一行看作是一个数据元素(或称结点、表目、记录)。表中每一行分别由编号、姓名、基本工资、副食补助,……,实发金额等数据项组成,这些数据项,通常也称作字段,其中我们把其值能唯一确定一个结点的字段称为关键字。此表中的编号字段就是关键字。

表中结点与结点的关系是线性关系,也就是说表中有且只有一个结点为表头(第一个结点),有且只有一个结点为表尾(最后一个结点),对表中的任何一个结点,与它相邻且在它前面的结点最多只有一个。这就是该表的逻辑结构。具有这种线性结构的表又称为线性表。

当上述工资表存入计算机时,假定结点与结点之间采用顺序存放的方式,从而就决定了该表的存储结构。

对上述工资表的运算主要有以下几种:

1. 插入——当有职工调入时,对此表进行插入运算,即增加一个新结点。
2. 删除——当有职工调离时,对此表进行删除运算,即删除某一个结点。
3. 修改——当职工调整工资时,对此表某些数据项进行修改,即更新结点中的有关数据内容。
4. 检索——当要查看某职工工资情况时。

当然,还可能与其他有关此表的运算,这要视具体的应用要求而定。

综上所述,按照某种逻辑关系组织起来的一组数据,以一定的存储方式把它存储在计算机的存储器中,并在这些数据上定义了一个运算的集合,这就是一个数据结构。

三、顺序表

顺序表就是以顺序方式存储的线性表。如上述工资表若以顺序方式存储时就称为顺序表。

在高级程序设计语言中,表示顺序表的一般方法就是使用数组。以 Pascal 语言为例,上述工资表可用如下记录类型的数组定义:

```

TYPE
  node = RECORD
    bh: integer;
    xm: string;
    jbgz: real;
    ...
  
```

```

  sfgz: real
END;
list = ARRAY[1..m] OF node;
VAR
  gzb: list; {顺序表数组}
  length: integer {表长}
  以上, m 为一个上限,即表中元素不会超过此数。
  gzb 为工资表数组, length 为工资表中当前的表目数
  (表长)。
  
```

为了讨论方便,我们以一个整形数据项的顺序表为例,这样,上面数组元素将改用 integer 类型。另外,为了实现上的方便,我们允许有空表存在。

```

CONST
  max = 500;
TYPE
  list = ARRAY[1..max] OF integer;
VAR
  a: list; {顺序表数组}
  length: integer {表长}
  
```

显然,以上变量定义已经确定了顺序表的逻辑结构和存储结构,再需要确定的是顺序表的运算集合。我们可设想以下最基本的运算,

1. 对表进行初始化(置为空表)
2. 询问表长
3. 在表尾添加一个元素 x
4. 在第 i 个元素后插入一个元素 x
5. 删除第 i 个元素
6. 询问第 i 个元素的内容
7. 更新第 i 个元素的内容

可以想象,任何其他关于顺序表的运算都可由以上7种运算组合得到。

以下是实现上述基本运算的 Pascal 函数和过程:

```

PROCEDURE initialize;
BEGIN
  length := 0;
END;

FUNCTION list—length: integer;
BEGIN
  list—length := length;
END;

PROCEDURE append(x: integer);
BEGIN
  IF length < max THEN
    BEGIN
      length := length + 1;
      a[length] := x;
    END
  ELSE
    writeln('list overflow');
END;

PROCEDURE insert(i, x: integer);
VAR
  k, j: integer;
  
```

```

BEGIN
  IF length < max THEN
    BEGIN
      k := i;
      IF i < 0 THEN k := 0;
      IF i > length THEN k := length;
      FOR j := length DOWNTO k+1 DO
        a[j+1] := a[j];
        a[k+1] := x;
        length := length+1
      END
    END

    ELSE
      writeln('list overflow')
    END;
  PROCEDURE delete(i: integer);
  VAR
    j: integer;
  BEGIN
    IF length > 0 THEN
      IF (i > 0) AND (i <= length) THEN
        BEGIN
          FOR j := i TO length DO
            a[j] := a[j+1];
            length := length-1
          END
        END
      ELSE
        writeln('list underflow')
      END;
    END;
  FUNCTION node(i: integer): integer;
  BEGIN
    IF (i > 0) AND (i <= length) THEN
      node := a[i]
    ELSE
      writeln('index out of range')
    END;
  PROCEDURE update(i, x: integer);
  BEGIN
    IF (i > 0) AND (i <= length) THEN
      a[i] := x
    ELSE
      writeln('index out of range');
    END.

```

四、Josephus 问题

设有 n 个人围坐在一个圆桌周围, 现从第 S 个人开始报数, 数到第 m 的人出列, 然后从出列的下一个人重新开始报数, 数到第 m 的人又出列, \dots , 如此重复直到所有的人全部出列为止。Josephus 问题是: 对于任意给定的 n, s 和 m , 求出按出列次序得到的 n 个人的顺序。

现以 $n=8, s=1, m=4$ 为例, 若开始的人员顺序为 $P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8$, 则问题的解是 $P_4, P_8, P_5, P_2, P_1, P_3, P_7, P_6$ 。

我们可用上节给出的顺序表数据结构来模拟此问题的求解。程序如下:

```

PROGRAM Josephus;
CONST
  max = 500;
TYPE
  list = ARRAY[1..max] OF integer;
VAR
  a: list; {顺序表数组}
  length: integer {表长}
  m, n, s, i: integer;
  {以上 PROCEDURE 和 FUNCTION 的定义在此从略}
BEGIN {MAIN}
  initialize;
  n := 8;
  s := 1;
  m := 4;
  FOR i := 1 TO n DO
    append(i);
  FOR I := n DOWNTO 1 DO
    BEGIN
      s := (s+m-1) MOD i;
      IF s = 0 THEN s := i;
      write(node(s), ' ');
      delete(s);
    END;
  writeln
END

```

五、抽象数据类型

也许读者会问, 上述 Josephus 问题用不着这么繁琐地定义一些基本运算, 完全可用一个过程来实现, 而且效率更高。那么, 为什么我们要这么做呢? 答案是: 过程抽象。其特征是, 将应用问题中经常用到的操作抽取出来用过程实现, 作为对运算符的扩展。请读者自己写一个不用过程抽象的 Josephus 问题的算法 (即不必定义运算集而直接访问顺序表)。比较一下, 读者会发现, 采用过程抽象后, 问题变得简单了, 也更加易于理解了。应用问题越复杂采用过程抽象就越重要。

然而, 我们还有使问题进一步简化的办法, 也就是把过程抽象与数据抽象结合起来。我们试将上述的 Josephus 问题改为产生一个新的表示出列顺序的顺序表, 原顺序表内容保持不变。这就是说, 需要定义两个数组, 一个是原顺序表 a , 另一个是结果顺序表 b 。显然, 原定义的那些基本运算只适用于顺序表 a , 对顺序表 b 却不适用, 所以必须定义关于顺序表 b 的那些基本运算。然而, 对任何其他顺序表 c, d, \dots , 是否我们必须为每一个顺序表都定义基本运算? 答案是: 没有必要, 我们可以进一步进行数据抽象。

我们发现前述的基本运算集只定义在具体的数据 (a) 上, 因而其他数据不能共享。我们应设法将其定义在数据类型 $list$ 上, 即设法把基本运算看成是其中的一部分, 然后再定义顺序表 a 和 b 或更多的表, 而基本

运算作为顺序表类型的一部分只需定义一次。在前述例子中,我们可对基本运算增加相应的数据类型的参数。为了叙述方便,我们认为顺序表中的结点为大于0的整数,用0表示顺序表的结束标志。如下面程序所示:

```
PROGRAM Josephus;
CONST
  max=500;
TYPE
  list=ARRAY[1..max] OF integer;
VAR
  a,b; list;
  m,n,s,i; integer;

PROCEDURE initialize(VAR l; list);
BEGIN
  l[1]:=0;
END;

FUNCTION length(VAR l; list): integer;
VAR
  i; integer;
BEGIN
  i:=1;
  WHILE (i<max) AND (l[i]<>0) DO
    i:=i+1;
  length:=i-1  (the list length <=max-1)
END;

PROCEDURE append (VAR l; list; x; integer);
VAR
  i; integer;
BEGIN
  i:=length(l);
  IF i<max-1 THEN
  BEGIN
    l[i+1]:=x;
    l[i+2]:=0;
  END
  ELSE
  writeln('list overflow')
END;

PROCEDURE insert (VAR l; list; i,x; integer);
VAR
  k,j,len; integer;
BEGIN
  len:=length(l);
  IF len<max-1 THEN
  BEGIN
    k:=i;
    IF i<0 THEN k:=0;
    IF i>len THEN k:=len;
    FOR j:=len DOWNTO k+1 DO
      l[j+1]:=l[j];
    l[k+1]:=x;
```

```
l[len+2]:=0;
END
ELSE
  writeln('list overflow')
END;

PROCEDURE delete(VAR l; list; i; integer);
VAR
  j,len; integer;
BEGIN
  len:=length(l);
  IF len>0 THEN
  IF (i>0) AND (i<=len) THEN
  BEGIN
    FOR j:=i TO len DO
      l[j]:=l[j+1]
    l[len]:=0;
  END
  ELSE
  writeln('list underflow')
  END;

FUNCTION node (VAR l; list; i; integer);integer;
BEGIN
  IF (i>0) AND (i<=length(l)) THEN
  node:=l[i]
  ELSE
  writeln('index out of range')
  END;

PROCEDURE update(VAR l; list; i,x; integer);
BEGIN
  IF (i>0) AND (i<=length(l)) THEN
  l[i]:=x
  ELSE
  writeln('index out of range');
  END;

PROCEDURE print (VAR l; list);
BEGIN
  i:=1;
  WHILE (i<max) AND (l[i]<>0) DO
  BEGIN
    write(l[i], ' ');
    i:=i+1
  END;
  writeln
  END;

BEGIN {MAIN}
  initialize(a);
  initialize(b);
  n:=8;
  s:=1;
  m:=4;
  FOR i:=1 TO n DO
```

```

append(a,i);
print(a);
FOR i:=n DOWNTO 1 DO
BEGIN
s:=(s+m-1)MOD i;
IF s=0 THEN s:=i;
append(b,node(a,s));
delete(a,s);
END;
print(b);
END

```

我们把对数据的运算定义在数据类型中,使得对数据的操纵更进一步的抽象化。上面主程序中有关顺序表 a, b 的访问是用已定义好的基本运算实现的。事实上,任何一个关于顺序表的应用都可用上述基本运算组合实现,程序员却无须知道顺序表的具体实现细节。我们把这样定义的数据类型称作抽象数据类型。简言之,抽象数据类型就是一个数学模型加上定义在该数学模型上的运算集合。

Pascal 及 C 等高级语言并没有定义抽象数据类型的语言机制。Ada 语言提供了 Package 机制以实现抽象数据类型。近年来较为流行的面向对象的语言不仅实现了抽象数据类型,而且扩充了类及类继承的概念。如 C++, Turbo Pascal 6.0 等等。上例中,我们只不过是对抽象数据类型进行模拟实现而已。也就是要读者记住基本运算集定义在数据类型之中,并且作为访问数据的唯一途径。

六、栈(Stack)

栈是一种线性表,它限制插入和删除运算只能在表的一端进行。这一端称为栈顶。这种对插入和删除的限制,决定了栈是“先进后出”的表。这种“先进后出”的特性,使得栈这一数据结构在计算机领域中有着广泛的应用。

以下我们以顺序方式给出一个整数栈的例子。其基本运算集为:

1. 栈的初始化 (init);
2. 测试栈是否为空 (empty);
3. 往栈中推入一个结点 (push);
4. 从栈中托出一个结点 (pop);
5. 求栈顶结点的内容 (stacktop);

```

PROGRAM stack.examp1e;
{example of integer stack}
CONST
max=100;
TYPE
stack=RECORD
    item: ARRAY[1..max] OF integer;
    top: 0..max
END;
VAR
s: stack;
x: integer;

```

```

PROCEDURE init(VAR s: stack);
{initialize stack s}
BEGIN
s.top:=0
END;

FUNCTION empty(s: stack) boolean;
{test if stack s is empty}
BEGIN
IF s.top=0 THEN
empty:=true
ELSE
empty:=false
END;

PROCEDURE push(VAR s: stack; x: integer);
{push x into stack s}
BEGIN
IF s.top=max THEN
writeln('overflow')
ELSE
BEGIN
s.top:=s.top+1;
s.item[s.top]:=x;
END
END;

FUNCTION pop(VAR s: stack): integer;
{pop stack s and return the value}
BEGIN
IF empty(s) THEN
writeln('underflow')
ELSE
BEGIN
pop:=s.item[s.top];
s.top:=s.top-1
END
END;

FUNCTION stacktop(s: stack): integer;
{return top element of stack s}
BEGIN
IF empty(s) THEN
writeln('underflow')
ELSE
stacktop:=s.item[s.top]
END;
BEGIN {main program}
init(s);
FOR x:=1 TO 20 DO
push(s,x);
WHILE NOT empty(s) DO
BEGIN
x:=pop(s);
writeln(x);
END

```

END.

在编译程序中,利用栈我们可以解决表达式的求值问题,还可实现递归子程序的调用等等。

七、队列(Queue)

队列是一种线性表,它限制插入运算在表的一端进行,删除运算在表的另一端进行。插入端称为队头(front),删除端称为队尾(rear)。这种对插入和删除的限制,决定了队列是“先进先出”的表。这种“先进先出”的特性,使得队列这一数据结构在计算机领域中有着广泛的应用。

以下我们以顺序方式给出一个整数队列的例子。其基本运算集为:

1. 队列的初始化(init);
2. 测试队列是否为空(empty);
3. 测试队列空间是否已满(full);
4. 入队:往队列中加入一个结点(enqueue);
5. 出队:从队列中删除一个结点(dequeue);
6. 求队头结点的内容(front);

```
PROGRAM queue.example;
{example of an integer queue}
CONST
max=200;
TYPE
queue=RECORD
    item: ARRAY[1..max] OF integer;
    front,rear: 1..max
END;
VAR
qu: queue;
i,x: integer;

PROCEDURE init(VAR q: queue);
{initialize queue q}
BEGIN
q.front:=1;
q.rear:=1;
END;

FUNCTION empty(q: queue): boolean;
{test if queue q is empty}
BEGIN
IF q.front=q.rear THEN
empty:=true
ELSE
empty:=false
END;

FUNCTION full(q: queue): boolean;
{test if queue q is full}
BEGIN
IF ((q.rear=max) AND (q.front=1)) OR (q.rear+1=q.
front) THEN
full:=true
```

```
ELSE
full:=false
END;

PROCEDURE enqueue(VAR q: queue; x: integer);
{add element x to queue q}
BEGIN
IF full(q) THEN
writeln('queue overflow')
ELSE
BEGIN
q.item[q.rear]:=x;
IF q.rear=max THEN q.rear:=1
ELSE q.rear:=q.rear+1
END
END;

FUNCTION dequeue(VAR q: queue): integer;
{delete the front element of queue q and return the value}
BEGIN
IF empty(q) THEN
writeln('queue underflow')
ELSE
BEGIN
dequeue:=q.item[q.front];
IF q.front=max THEN q.front:=1
ELSE q.front:=q.front+1
END
END;

FUNCTION front(q: queue): integer;
{return front element of queue q}
BEGIN
IF empty(q) THEN
writeln('queue underflow')
ELSE
frront:=q.item[q.front]
END;

BEGIN {main program}
init(qu);
FOR i:=1 TO 20 DO
enqueue(qu,i);
WHILE NOT empty(qu) DO
BEGIN
x:=dequeue(qu);
write(x,' ')
END;
writeln
END.
```

在分时操作系统中,利用队列我们可以解决缓冲打印问题,特殊情况下,我们还可用优先队列解决不同优先级结点的排队问题。

病毒克星——华思中文防病毒卡

北京安定门外东后巷28号、华思专用集成电路设计有限公司(100011)

一 概述

1. 什么是计算机病毒

计算机病毒的提法在某种意义上是借助了生物学病毒的概念,这是因为计算机病毒有着与生物病毒相同的特征。计算机病毒能够自我复制、有传染性、潜伏期和发作期。所不同的是计算机病毒侵害的对象不是生物体,而是计算机系统。就其实质来说,计算机病毒是一个能够实现自我复制并借助一定的载体(如磁盘等)存在的具有潜伏性、传染性和破坏性的程序。

1983年11月美国计算机专家首次提出了计算机病毒的概念并进行了验证。1987年计算机系统中首次发现了世界上第一例电脑病毒: BRAIN 病毒。在其后几年中,各种各样的电脑病毒大量产生并迅速传播蔓延。据不完全统计,迄今为止,全世界已发现的各类电脑病毒已达2800余种。

几乎所有的计算机病毒都是人为地故意制造出来的。尽管编制病毒程序的人目的各不相同,但计算机病毒造成的破坏程度却是巨大的,有时一旦病毒扩散后,编制者自己也无法控制。当今计算机正以日新月异的发展速度广泛地深入到社会生活的各个方面,并且扮演着越来越重要的角色。计算机的逐步普及极大地提高了工作效率,同时,也为计算机病毒提供了滋生环境。实际上,如何对付计算机病毒已不单纯是一个技术问题,而是一个严重的社会问题,需要政府行政、立法部门、计算机软硬件人员以及用户的通力协作,一方面增强计算机系统的安全性;一方面严厉打击计算机犯罪行为。

2. 计算机病毒的寄生方式

计算机病毒是一种可直接或间接执行的程序,但它通常不以独立的文件形式存在,而是依附于某种载体。

微机系统目前常用的永久性存储设备(即外存储器)主要是磁盘,包括硬盘和软盘。硬盘的容量一般为几十兆字节,大容量的硬盘可到上百兆。软磁盘容量较小,常用的为360KB、720KB、1.2MB和1.44MB等。微机系统所使用的文件通常存放在磁盘中。所以,微机病毒是以磁盘为主要载体。

计算机病毒的寄生方式主要有:

(1)寄生在磁盘 DOS 引导记录中

任何操作系统都有一个自举过程。例如 DOS 在启动时,首先由系统读入 DOS 引导记录并执行它,然后将 DOS 读入内存。病毒程序常利用 DOS 系统这一特点,侵占 DOS 引导记录并将原来的引导记录内容及病毒其它部分放到磁盘的其它空间。这样系统每次启动,病毒先被激活,将自身复制到内存中,并设置一定的触发条件,然后才引入正常的操作系统环境。在内存中驻留的病毒程序一旦触发条件成熟,就会感染无毒的磁盘,使之带上病毒。

(2)寄生在可执行程序中

这种病毒寄生在正常的可执行程序中。每当带病毒的程序执行,病毒即被激活。首先执行病毒程序,它将自身复制到内存中并驻留,然后置触发条件。病毒可能立即传染,但一般不会发作。做完这些工作后,才开始执行正常的程序。病毒程序也可能在执行正常程序之后再设置触发条件。病毒可以寄生在原程序的首部,也可寄生在尾部。但一般都要修改原程序的长度和一些控制信息,以保证病毒成为程序的一部分,并在程序运行前首先执行它。

(3)寄生在硬盘的主引导记录中。

(4)寄生在文件分配表中。

3. 计算机病毒的一般工作过程

(1)传染源

病毒总是依附于某些存储介质存在的,并通过一定的媒介传染。软盘、硬盘等都可构成传染源。

(2)病毒激活

病毒装入内存,并设置触发条件。一旦条件成熟,病毒就开始作用。触发条件可以是内部时钟、系统日期、用户标识符等。

(3)病毒表现

表现是计算机病毒的主要目的之一,有时在屏幕显示出来,有时则表现为破坏系统的数据。凡是软件技术能够触及到的地方,都在其表现范围内。

(4)传染

计算机病毒的传染性是病毒性能的一个重要标志。病毒在传染对象中复制一个自身的副本。

当计算机系统感染病毒时,通常会有如下表现:

(1)破坏操作系统的引导记录和硬盘的主引导记录,使系统不能正常启动或不能启动。

(2)破坏文件分配表(FAT),使用户在磁盘上的数据丢失。

(3)删除软盘或硬盘上的可执行文件或数据文件。

(4)修改或破坏文件中的数据。

(5)改变磁盘分配,造成数据写入错误。

(6)影响内存常驻程序的正常执行。

(7)在磁盘上产生坏的扇区,使磁盘可用空间减小。

(8)更改磁盘卷标。

(9)占用系统内存,使正常的的数据或文件不能存储。

(10)对整个磁盘或硬盘的特定磁道或扇区进行格式化。

(11)在系统中产生新文件。

(12)改变系统正常运行。

4. 华思防病毒卡的工作原理

华思防病毒卡的核心是病毒的自动识别。自动识别病毒所依据的标准是判断所运行的程序是否具有传染的特征。所有的病毒都有传染性,失去了传染性,就不能称其为病毒。

华思防病毒卡直接与 DOS 系统构成一体,监视系统正在进行的操作。由于采用了全新的技术,防病毒卡不占用修改任何中断矢量,病毒无法避开防护系统,从而保证了系统鉴别病毒的可靠性。对于微机系统中易受病毒攻击的部分,防病毒卡提供了特殊的保护措施。例如硬盘的主引导区(MAIN BOOT AREA)、磁盘的引导区(DOS BOOT AREA)和可执行文件等。一旦这些地方受到病毒的攻击,防病毒卡将给出报警信息,阻止病毒的破坏。

对带有病毒的程序,当运行该程序时,防病毒卡能阻止病毒在内存中自我复制,使病毒在内存中无法藏身,失去传染性,从而达到病毒免疫能力。因此,带毒程序在插有华思防病毒卡的机器中可以安全运行,不会传染给其它文件。

华思防病毒卡中文显示报警系统富有特色。由于采用了字模固化技术,即使在没有安装汉字显示卡或运行 CC-DOS 等汉字系统的机器中也能显示出中文提示信息,并且有良好的适应性和兼容性。

二 安装方法

华思防病毒卡适用于采用 DOS 2.0 以上版本操作系统的 IBM PC/XT、AT 及其兼容机。

安装步骤

(1)在安装华思防病毒卡之前,请先运行实用程序盘中的 INSTALL. EXE 程序。该程序将显示 C000H—DFFFH 段地址占用情况,供您在设置卡上的 DIP 开关时参考。

A)INSTALL ? (RETURN)

显示从 C000H—DFFFH 段地址的内存占用情况。

R 被 ROM 占用

M 被 RAM 占用

C 被华思防病毒卡占用

• 未被占用

A)INSTALL (RETURN)

给出几个可行的 DIP 开关设置,供您参考选择。

(2)设置卡上的 DIP 开关

由于华思防病毒卡上带有 ROM 和 RAM 芯片,因此,需要占用内存地址。占用的内存地址可以通过设置卡上的 DIP 开关来选择,以避免和其它扩展卡或扩展内存(EMS)的地址冲突。

华思防病毒卡在出厂时的 DIP 开关设置为:

K1 K2 K3 K4 K5

ON ON ON OFF ON

即占用地址 DO00H:0000—DO00:1FFFH。通常不用再设置 DIP 开关,只需将卡插入机器即可。如果机器中还有其它扩展卡,或机器使用了扩展内存(如虚拟盘等),可根据 INSTALL 程序所显示的地址占用情况重新设置 DIP 开关,以避免冲突。

DIP 开关与地址的映射关系见附表。

(3)插入华思防病毒卡

切断机器电源,千万不能带电操作。将华思防病毒卡插入机器中,注意卡的元件安装面应与其它卡一致,不能插反。安装完毕后,再次启动机器,华思防病毒卡就开始工作了。

(4)机器启动后,将首先进行自检,然后屏幕显示:华思防病毒卡 V2.0

这时按任意键,机器将启动 DOS 系统。如果机器不能正常启动或启动时未显示如上信息,说明 DIP 开关设置与其它地址有冲突,这时应回到步骤(1),重新安装防病毒卡。

三 实用程序

华思防病毒卡实用程序盘中包括如下文件:

1 INSTALL. EXE

2 SETPASSW. EXE

3 CLEANPAS. EXE

4 CHANGS. EXE

5 WNCARD. EXE

(一)INSTALL. EXE

该程序将显示 C000H—DFFFH 段的内存地址占用情况,供您在设置卡上的 DIP 开关时参考。

格式1:A)INSTALL ?(RETURN)

显示从 C000H—DFFFH 段内存地址的占用情况。

R 被 ROM 占用

M 被 RAM 占用

C 被华思防病毒卡占用

• 未被占用

格式2:A)INSTALL (RETURN)

给出几个可行的 DIP 开关设置,供您参考选择。如果已安装了防病毒卡,程序将显示:

HUASI ANTIVIRUS CARD HAS BEEN INSTALLED!

(二)SETPASSW.EXE

该程序用于设置或修改机器口令。设置口令时,要求用户输入两次口令。只有当两次输入的口令相同时才有效,否则口令设置失败。如果是修改机器口令,程序将要求您输入原来的口令,以判断您是否有权修改口令。当您正确输入原口令后,才能设置新的口令。

(三)CLEANPAS.EXE

该程序用于清除口令。清除口令时,程序将要求您输入原来的口令,以判断您是否有权清除口令。口令可以由任意 ASCII 字符组成,通常可包括字母、数字及其它常用分隔符等。注意:口令中字母大小写有区别。例如:口令 ABC 与口令 abc 是不一样的。

如果用户忘记了所设置的口令,以至无法启动机器时,可先关机。取下防病毒卡后,启动机器并执行 CLEANPAS.EXE 即可清除口令。这时插上防病毒卡后,再启动机器,就不会遇到麻烦了。

(四)CHANGS.EXE

修改防病毒卡的工作状态。当机器启动后,防病毒卡自动处于工作状态。当用户认为有必要修改防病毒卡的工作状态时,可以执行该程序。

防病毒卡处于工作状态(WORKING)时,能够检测病毒,而当卡处于关闭状态(CLOSING)时,则不检测病毒。防病毒卡处于这两种工作状态时,均不会影响机器的正常工作。

建议:当用户安装软件包时,可将防病毒卡临时关闭,以避免防病毒卡过多地显示提示信息,中断安装程序。当软件包安装完毕后,再改变卡的工作状态。

(五)WNCARD.EXE

有些软件如 Microsoft Windows、NOVELL Netware 等工作在保护方式下(Protected Mode)。因此,在运行这些软件时,请将卡的工作方式变为扩展方式。

A)WNCARD 1<RETURN> 设置为扩展方式

A)WNCARD 0<RETURN> 设置为默认方式

当机器启动后,防病毒卡的工作方式为默认方式。可以将设置卡工作方式的命令输入 AUTOEXEC.BAT 文件中,当机器启动后,卡即可按用户要求的工作方式工作。

四 用户提示信息说明

1 请输入口令:

要求用户输入口令。刚安装华思防病毒卡时,没有设置用户口令。如需要设置口令,可用实用程序 SETPASSW.COM。详见第三章(2)。

2 非法口令!

当用户输入口令不正确时显示此信息。这时应按提示重新输入用户口令。注意:口令中的字母有大小写区别。若输入十次口令仍不正确,将拒绝用户使用机器。如果用户确实忘记了口令,请参照第三章(3)清除口令,才能重新启动机器。

3 在系统区发现病毒

在系统区发现×××病毒

防病毒卡在系统当前启动盘的引导区中发现病毒,是引导类型的病毒。这时不能从当前启动盘启动系统,必须使用干净无毒的引导盘才能启动系统。

如果是硬盘引导区发现病毒,应使用与硬盘中 DOS 版本相同的引导盘重新启动系统。请参照第五章所建议的方法清除引导类型的病毒。

4 主系统区中发现病毒!

防病毒卡在硬盘的主引导区(MAIN BOOT AREA)发现病毒,是引导类型的病毒。此时机器不能从硬盘启动。应使用与硬盘中 DOS 版本相同的引导盘重新启动系统。请参照第五章中所建议的方法清除引导类型的病毒。

5 写主系统区前,发现病毒

写主系统区前,发现×××病毒
系统扇区将被修改

该信息说明有病毒正试图感染系统的引导扇区。有时用户的某些操作也可能引起防病毒卡显示该信息。例如:当用户故意修改引导扇区或清除引导扇区中的病毒时,就有可能出现这一信息。

6 文件×××.×××中发现病毒且内存中病毒已被清除

发现文件类型病毒。当运行带病毒的程序时,会显示这一信息,表示防病毒卡检测到病毒的存在并且成功地阻止了病毒程序在内存中复制(驻留)。这时带病毒的程序仍可正常运行,不会传染给其它文件,也不会破坏磁盘中的数据文件。

注意:尽管在插有华思防病毒卡的机器中,带毒程序可以安全运行,我们仍建议用户应及时清除程序所带的病毒,以避免病毒在其它没有病毒防护系统的机器中扩散。对于某些被多种病毒感染(即交叉感染)的程序,华思防病毒卡也同样有效,但这样的带毒程序往往不能正常运行,有可能会死机。请参照第五章所建议的方法清除文件类型的病毒。

7 ×××.×××将被修改(删除、改名)!

该信息提示用户某个可执行文件将被修改(删除、改名)。这时要求用户自己判断这一操作是不是病毒行为。

通常如果被修改的文件是 COMMAND.COM、DOS 实用程序或其它应用软件包中的文件,则可判断为是病毒正试图感染其它文件。如果用户正在编译、链接自己编写的程序时出现此提示,则一般不是病毒行为,而是正常的修改(删除、改名)操作。

建议1 用户在判断时就应慎重考虑,如果一时不知如何判断,应中止当前操作(键入字母 A)

2 当用户安装软件包时,可暂时关闭防病毒卡。待软件安装完毕,再恢复卡的工作状态。参见第三章(4)。

如果是病毒,请参照第五章中的方法及时清除。

五 病毒的清除

1. 引导类型病毒的清除

硬盘的主引导记录存放在硬盘的0磁头、0柱面、1扇区,包括硬盘自举引导代码和分区信息表。该扇区是硬盘所特有的,通常是在进行DOS分区时建立的。目前,有许多流行的计算机病毒寄生在硬盘的主引导区中或在发作时破坏主引导区。另外,用解毒盘清除病毒时也有可能破坏硬盘的主引导记录。

软盘的DOS引导区在软盘的0面、0道、1扇区。硬盘的DOS引导区在硬盘DOS分区的第一扇区(逻辑第一扇区)。DOS引导区被损坏,同样无法启动机器。

在DOS系统中,有两个隐藏文件IBMBIO.COM和IBMDOS.COM要求按顺序存放在根目录的开始位置。若这两个文件被破坏或位置被改变,均会影响启动过程。

清除引导类型的病毒时,应首先选用现有的清除病毒的软件。由公安部推出的清病毒软件使用较为广泛。它是针对国内普遍流行的病毒开发的清毒软件,并定期更新版本。

如果消病毒软件不能奏效,可采用如下方法,一般都能清除引导类型的病毒。下面以硬盘为例说明:

(1)用与硬盘中DOS版本相同的系统软盘启动机器。启动后,用SYS命令向硬盘传送系统文件。取出软盘,从硬盘启动系统,若防病毒卡没有报错,说明病毒已被覆盖清除。否则,转第(2)步。

(2)若第(1)步不能清除病毒,则可能是DOS分区的引导扇区被破坏。这时可以找一台与该机器同型号、DOS版本相同、不带病毒的机器,用不带病毒机器中的引导区覆盖带病毒机器硬盘中的引导区,即可清除病毒。重新启动系统,若防病毒卡没有报错,说明病毒已被清除。否则,转第(3)步。

(3)通过第(2)步仍不能清除病毒,说明硬盘的主引导区被破坏。解决方法与(2)相同。用不带病毒机器中的有相同分区表的主引导区覆盖带病毒机器中的主

引导区。即可清除病毒。

2. 文件类型病毒的清除

清除文件类型的病毒时,应首先选用现有的清除病毒的软件。由公安部推出的清病毒软件使用较为广泛。它是针对国内普遍流行的病毒开发的清病毒软件,并定期更新版本。

如果现有的软件不能清除病毒,可用不带病毒的同名文件将其覆盖。

华思中文防病毒卡《电子与电脑》编辑部有售:

定价:320元(邮寄请附加10元邮费)

地址:北京173信箱《电子与电脑》编辑部

邮政编码:100036 电话:81.5242

附表 DIP 开关设置与地址对应表

K1	K2	K3	K4	K5	地址	K1	K2	K3	K4	K5	地址
ON	ON	ON	ON	ON	C000	ON	ON	ON	ON	OFF	E000
ON	ON	OFF	ON	ON	C200	ON	ON	OFF	ON	OFF	E200
ON	OFF	ON	ON	ON	C400	ON	OFF	ON	ON	OFF	E400
ON	OFF	OFF	ON	ON	C600	ON	OFF	OFF	ON	OFF	E600
OFF	ON	ON	ON	ON	C800	OFF	ON	ON	ON	OFF	E800
OFF	ON	OFF	ON	ON	CA00	OFF	ON	OFF	ON	OFF	EA00
OFF	OFF	ON	ON	ON	CC00	OFF	OFF	ON	ON	OFF	EC00
OFF	OFF	OFF	ON	ON	CE00	OFF	OFF	OFF	ON	OFF	EE00
ON	ON	ON	OFF	ON	DO00	ON	ON	ON	OFF	OFF	F000
ON	ON	OFF	OFF	ON	D200	ON	ON	OFF	OFF	OFF	F200
ON	OFF	ON	OFF	ON	D400	ON	OFF	ON	OFF	OFF	F400
ON	OFF	OFF	OFF	ON	D600	ON	OFF	OFF	OFF	OFF	F600
OFF	ON	ON	OFF	ON	D800	OFF	ON	ON	OFF	OFF	F800
OFF	ON	OFF	OFF	ON	DA00	OFF	ON	OFF	OFF	OFF	FA00
OFF	OFF	ON	OFF	ON	DC00	OFF	OFF	ON	OFF	OFF	FC00
OFF	OFF	OFF	OFF	ON	DE00	OFF	OFF	OFF	OFF	OFF	FE00

一本适用好学的计算机教材

为了在广大学生中普及计算机知识,培养他们学习计算机知识的兴趣,由多年从事计算机教育的专业老师编写了《计算机启蒙知识 BASIC 语言》一书,该书注重系统性,实践性和趣味性,与小学数学教材紧密结合,例题和习题以数学课本的内容为依据,内容通俗易懂,使学生们既巩固了所学的数学知识,又学习了计算机的编程知识,达到一举两得之功效。该书对学生学习计算

机程序设计很有帮助,适用于小学和初中做计算机课教材。

《计算机启蒙知识 BASIC 语言》由河北少年儿童出版社出版,面向全国发行,19万字。1993年3月第3次印刷,彩色注胶封面,印刷精美。每本3.50元(包括邮费)。联系人:张小毅。地址:四川重庆市南岸区一天门67号龙门浩职业中学,邮编:630064,电话:270173。

单片机的学习与应用

北京宣武区青少年科技馆(100053) 罗明宽 车金相

在北京市单片机应用技术协会的帮助和支持下,我们在青年学生中办了单片机学习班,吸引了很多爱好者。在教学中体会到要学好用好单片机,就要了解一个具体单片机应用系统的电路、原理,监控程序,所用芯片的特点及具体应用的实例,目的缩短初学到实用的时间,我们将从以上几方面陆续把自己粗浅扫体会奉献给读者。

我们使用的 DP-851 单片机教学实验系统主要由主板和实验板组成。主板包括一个单片机扩展系统、显示器及键盘,它的左侧有双排(共40根)弯插针,8031的40个管脚与它们对应相连,主板通过40芯扁平线将8031的引脚信号全部引出,提供给实验板。

DP-851 主板电路原理请参看图2。

图中的 IC₁、IC₂、IC₃ 构成了单片机的三总线结构,即数据线、地址线、控制线。地址总线16根,寻址范围64K,其中低8位地址由 P0 口提供,高8位地址由 P2 口提供,由于 P0 口还要作数据总线,所以它分时提供的低8位地址必须锁存,IC₂ 即 74LS373 就是带有三态门的 8 D 锁存器,它利用控制信号 ALE 的下降沿将 P0 口输出的地址锁存。数据总线8根由 P0 口提供,该口为三态双向口,单片机与外部信息的交换几乎全部由 P0 口传送。控制总线用于系统扩展的有:WR、RD、PSEN、ALE 和 EA。WR、RD 是执行外部数据存储器操作指令时自动生成的。PSEN 是读外部程序存储器时自动生成的读选通信号。EA 对使用 8031 的单片机系统必须接地,使单片机只能访问片外程序存储器。IC₃ 为 2 输入端四与门 74LS08,用其中一个与门将 RD 和 PSEN 相与,与门的输出 OE 无论是 RD 有效,还是 PSEN 有效,OE 都有效,即读数据存储器 and 读程序存储器两种指令可以混合使用,方便了初学者。

8031 虽然功能很强,但在使用时,必须扩展程序存储器。片内数据存储器的容量、并行 I/O 端口等等也都有限,所以,扩展往往不是一个芯片,虽然每个芯片都有一个片选端 CE,但是为合理安排地址空间,通常采用译码电路。从图1可见 DP-851 采用两片 74LS138 进行译码,74LS138 有三个控制端、三个输入端和八个输出端,只有当三个控制端 G1、G2A、G2B 为 100 状态时,才会对三个输入端 A、B、C 进行译码,从而译出 8 种地址,从八个输出端 Y₀~Y₇ 输出。例如当 CBA=000 时,Y₀ 为低电平有效,其它无效。当 CBA=001 时,Y₁ 为低电平有效,其它无效。当 G1、G2A、G2B 不为 100 时,输出都为高电平,即无效。

DP-851 译码电路具体如图2。从图②可以看出 IC₆ 的 G1、G2A 和 G2B 分别接 5V 和地,满足了译码器的工作条件 100。根据 74LS138 的真值表可知, Y₀ 输出低电平时 A15、A14、A13 三根地址线要为 0、0、0, IC₆ 的 Y₀ 输出的低电平作为 27128 程序存储器的片选信号,由它决定的地址范围用二进制表示时为:

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 × × × × × × × × × × × × × × × ×

如用十六进制表示则为:0000H~01FFH,细心的读者会发现地址区间只占 8K,因为 16K ROM 27128 里面固化有 16K 监控程序,实际为两个独立的程序块,前 8K 是为以键盘、数码管为输入、输出手段的用户提供的管理程序,后 8K 为以 PC 机为输入、输出手段的用户提供的管理程序。用户可以通过接在 A13 上的 K2 接地或接 5V 来选择键盘监控或 PC 监控,所以,它仅占 8031 的 8K 地址空间。

同理,读者可证 Y₄ 输出低电平时 A15、A14、A13 要为 1、0、0, IC₆ 的 Y₄ 输出的低电平作为 6264 数据存

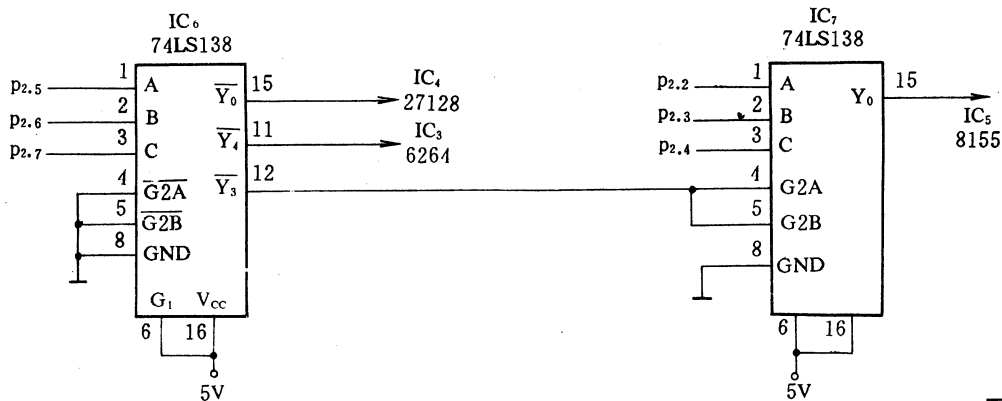


图 2

储器的片选信号,由它决定的地址范围用二进制表示时为:

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 0 0 × × × × × × × × × × × × × × × ×

如用十六进制表示则为:8000H~9FFFH。

从图2可见,IC₇的G1已接5V, $\overline{G2A}$ 、 $\overline{G2B}$ 和 IC₆的 $\overline{Y_3}$ 相接,只有 IC₆的 $\overline{Y_3}$ 输出低电平时, IC₇才能满足工作条件。读者可证 $\overline{Y_3}$ 输出低电平时 A15、A14、A13要为0、1、1, IC₆的 $\overline{Y_3}$ 输出的低电平满足了 IC₇的工作条件, IC₇的 $\overline{Y_0}$ 接8155的片选,要使 IC₇的 $\overline{Y_0}$ 输出低电平时 A12、A11、A10要为0、0、0,由此决定的地址范围用二进制表示时为:

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0 1 1 0 0 0 × × × × × × × × × × × × × × × ×

并行 I/O 扩展接口芯片 8155 的地址空间用十六进制表示则为:6000H~63FFH, 8155 的高 8 位地址由片选线 \overline{CE} 提供, 而低 8 位为片内地址。当它作片外 256 字节数据存储器使用时 IO/M=0(即 A8=0), 在本系统

中 6000H~60FFH 的 256 个字节的 RAM 为监控工作区, 作为片内存储器的映象区、系统标志以及数据缓冲区。8155 作扩展 I/O 口使用时, IO/M=1(即 A8=1), 这时命令寄存器、PA 口、PB 口、PC 口的低 8 位地址分别为 00H、01H、02H、03H, 在本系统中的地址则为 6100H~6105H。其中 PA 口的 PA0~PA7 为键盘的 8 根列线, PC 口的 PC0~PC3 为键盘的 4 根行线。PB 口的 PB0~PB7 经 IC10(即 74LS245)驱动控制 LED 数码管的段选信号; PA0~PA5 经 IC9(即 74LS06)驱动控制 LED 的数码管的位选信号。

图 1 中的 AN 为复位按钮, 按复位按钮后系统复位, 数码管显示 bJP-51。开关 K1 用于控制运行状态, 开关和地接通, 即 INT0(P3.2)接地为单步运行状态, 开关和 5V 接通为全速运行状态。8031 的串行口扩展为 RS232C 标准口, 它是把 RXD 和 TXD 通过晶体管 9012、9014 及附加电路组成的, 它可与 PC 机的标准 RS232C 连接, 在通信程序支持下进行系统的编程与调试。

80C31 单片机防掉电和抗干扰电源的设计

长春市东北师大物理系(130021) 马祝阳 姜凤怡

1. 前言

在许多工业监测计量仪表中, 80C31 单片机被广泛的采用于智能仪表, 它常常在恶劣的条件下和无人值守的情况下工作, 即使电源瞬间出现故障, 系统也能正常工作。据统计表明, 来自于电源瞬态欠压、瞬间脉冲干扰以及电源掉电所造成的误动作, 数据丢失, 占各种干扰的 90% 以上。因此, 怎样使 80C31 单片机能可靠地工作, 是许多设计者关注的问题。过去在这方面也见过很多报导。但是, 在许多报导中, 80C31 单片机在电网掉电和上电瞬间、电源快速地反复出现抖动时, 都不能正常工作, 而发生数据丢失现象。因此, 本文对上电、掉电过程中的抗干扰问题, 除了采用传统的抗干扰办

法外, 还采用了避错和容错等技术, 结合 80C31 掉电工作特性, 采用软硬件相结合技术进行设计, 使 80C31 的电源有了较强的抗干扰能力。

2. 电路图与组成

电路如图 1。它包括电源滤波器, 隔离变压器, 防干扰延时开关, 整流稳压滤波电路, 掉电欠压检测电路, 自动上电复位电路, 电池充电掉电电路。

3. 电路的工作原理和抗干扰措施

硬件部分:

220V 交流电经电源滤波器可吸收掉电网中大部分毛刺, (持续时间为 μS 数量级的)。然后采用隔离变压器降压至 12V, 再由三端稳压器输出 5V 供单片机

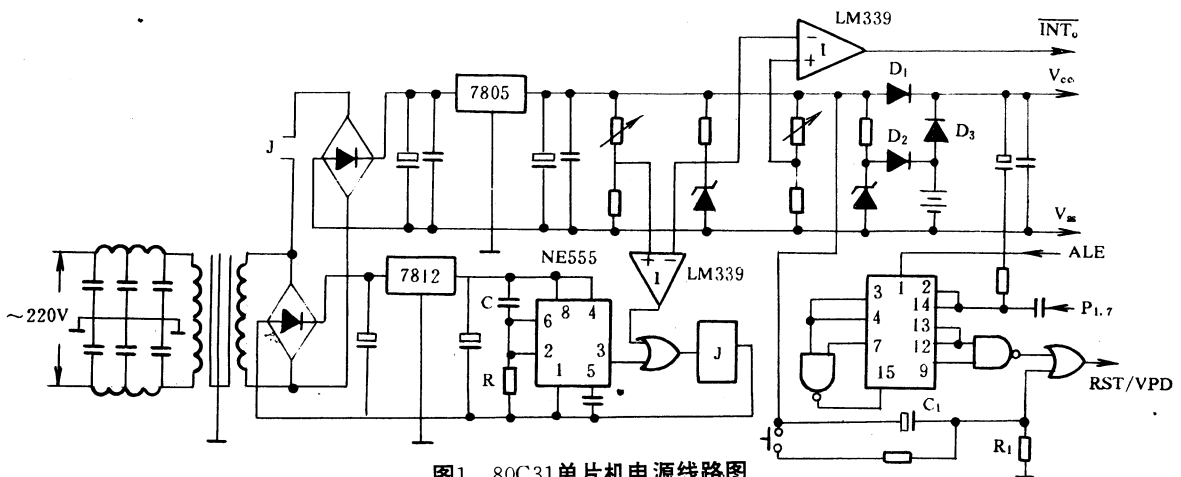


图1 80C31单片机电源线路图

使用,其目的有二点,第一点是防止干扰由变压器的电容效应进入后级。第二点是先由变压器变为12V,后由三端稳压器输出5V,目的在于防止持续时间为几分钟到几小时的过压(220V~250V)或欠压(220V~190V),电路仍能正常工作。

防干扰延时开关由555延时开关电路和固态继电器组成。当电路电源接通时,6脚电平为电源电压 V_{CC} ,随着RC充电,6脚电压下降到 $\frac{1}{3}V_{CC}$ 时,3脚输出反转,使继电器导通。因此,在没有导通时间内,外部电网的干扰冲击将被阻断。RC时间常数可根据实际情况选在合适系数之内。

电网的瞬态过压和欠压及失压检测电路,由稳压管(稳压值在3V左右)和分压电阻以及339电压比较器组成。欠压和失压情况,调整精密可调电阻使339电压比较器I输出翻转电压选在三端稳压器输出电压的4.8V,当三端稳压器的电压低于4.8V时,339比较器I输出低电平 \overline{INT} 。使80C31单片机进入掉电工作状态。这时只对RAM供电保护数据,而其它电路都不工作,避免系统受到冲击。过压检查电路使339电压比较器I输出翻转电压选在5.5V,当电网电压高于5.5V时,由339比较器输出信号经或门输出使固态继电器断开,让系统也进入掉电状态。由于固态继电器通断延时小于0.5周期。所以可以对半周以上的干扰进行保护。

正常上电复位由 R_1C_1 组成。重新上电后,信号经或门加到RESET端。干扰上电复位是由80C31的ALE脚输出一系列脉冲,送入74LS390组成的两位十进制计数器,当输入100个脉冲,74LS390便输出一个脉冲,经或门加到RESET端使程序复位。如果单片机没有受到干扰,即使程序没有跑飞进入死循环状态,则单片机用小于100个脉冲的时间,由软件使P1.7不断输出脉冲给计数器清零端。这时,系统处于正常运行,反之出现故障或干扰,程序不能正常工作,74LS390计到100个脉冲自动使系统复位。

备用电池供电电路由 $D_1\sim D_3$,4.2V稳压管焊接腿可充电3.6V电池P组成(型号3/GP60K 60mAh)。电源接通时,5V电压经 D_1 对80C31供电,同时对P充电,当P为3.6V时,电源对P充电停止。掉电后,P通过 D_3 对 V_{CC} 提供RAM电压,系统处于掉电状态时,RAM中数据可保持数周。

软件部分:

单片机供电系统的抗干扰方法,除了采用3.1谈到的硬件措施外,软件措施包括,掉电工作方式的掉电欠压中断子程序,上电复位抗干扰程序,软件陷阱程序,看门狗的软件程序。程序框图如下:

① 电源在启动或复位瞬间受到的干扰

80C31在初始上电时,SP指针在07H单元,本程序主体程序SP设置为50H单元。所以,一上电就检查SP是否为07H,就可以判定系统在上电瞬间是否受到干扰。如果SP此时不为07H,则程序跑飞进入陷阱程序,经处理由看门狗重新复位。

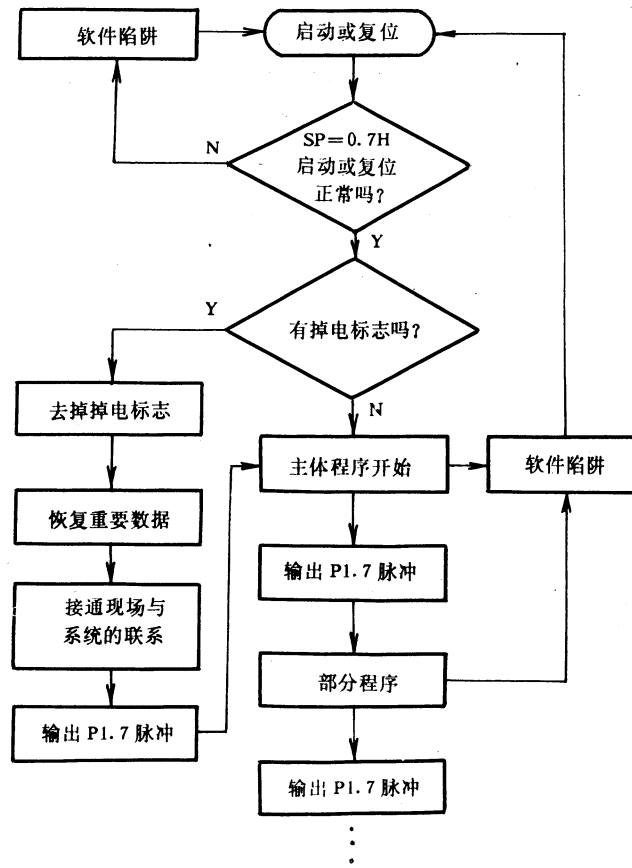


图2 抗干扰软件流程图

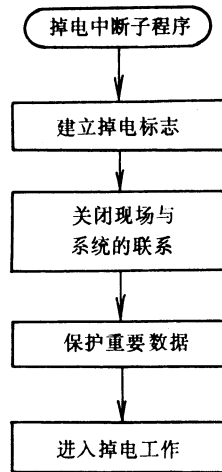


图3 掉电欠压中断子程序流程图

② 欠压、掉电的干扰

在整个程序运行期间受到掉电,欠压干扰时,由硬件检测出此信号后,系统便进入掉电欠压中断子程序。其程序包括:建立掉电标志、保护重要数据、关闭现场与系统的联系、使系统进入掉电工作状态。这样,数据

可以在RAM中被保护,不受冲击。

③ 电源干扰使程序跑飞

最后,仍有电源干扰使程序跑飞时,在有程序的区域内,根据程序指令和程序的结构在容易影响程序的地方写入0000020050机器码,使程序转入0050H陷井程序。软件的陷井程序应包括重要数据判断、回复和处理,然后进入死循环由看门狗复位。在没有程序的地方都写上0000020050机器码,使跑飞的程序落入软件陷井程序。看门狗复位脉冲设计在整个程序区内,在每隔100个机器周期时间里,设一个P1.7输出脉冲程序,在程序正常工作时,不断由P1.7输出脉冲信号使看门狗清零,这样就没有信号加到复位端。

4. 结束语

通过以上论述,可以看出本文除了在硬件采用传

统滤波、隔离、屏蔽技术之外,还用了延时开关避开电网上掉电抖动给系统的冲击,然后又利用80C31掉电工作方式所具有的特性,在掉电时,只对RAM供电,而其它电路都不工作的特点,由检测电路检测出掉电或干扰信号,使电路进入掉电工作状态,进一步避开电网的干扰。如果由前面几道防线都没有防住的干扰冲击了系统,再使用软件容错技术,使程序掉入陷井,经软件处理后,由看门狗自动复位。经实验,它与电焊机,电钻机共用同一电源,并在旁边一起工作,系统没有出现故障。电源反复上电掉电去冲击系统,系统中数据不丢失。系统缓慢上电、掉电、程序能自动复位,数据不丢失。所以,本电源通过软硬件的多重防干扰设计,大大地减少了电源给系统带来的干扰。

从三种通用的八位单片机硬、软件比较来看 Z8系列单片机的优越性

Zilog 中国区高级工程师——董伯明

四、其它功能:

这里再简述一下掉电模式,自测和系列产品兼容性等问题。

1. 掉电模式:三种芯片均有掉电保护模式,采用附加电源,Z8611和8051可保护所有寄存器的内容。而MC6801仅能保护前64个寄存器的内容。

在掉电模式,Z8611只需利用晶振的一个输入端,通过外接电源来保护寄存器内容。而8051和MC6801均需将一个输入端保留来完成这功能,MC6801用V_{cc} Standly端,8051用V_{pd}端。

2. 系列兼容性: Z8611和Zilog的Z-BUS™完全兼容,就是讲Z8611可以和Z-BUS™上的所有外设直接连接使用。MC6801和所有MC6800的系列可连接使用。而8051只能通过软件来和8048系列和其另外系列的产品来连用。

五、软件运行比较

以下我们选择常要遇到的七个功能,用三种芯片的指令分别编写程序完成,并优化后,在这三个系统中运行,最后对其指令的简繁、程序的长短、运行时间的快慢、占内存的多少进行比较,最后再综合比较。要注意,对MC6801和8051提供的是机器周期,而Z8611是时钟周期。Z8611的时钟周期除6后才是指令的周期。MC6801和8051的机器周期是1μS。Z8611的时钟周期12MHz时是0.166μS。

程序清单

(1)对16位的字产生冗余码测试CRC

	8051		机器周期	字节
	MOV	INDEX,#8	1	2
LOOP:	MOV	A,DATA	1	2
	XRL	A,HCHECK	1	2
	RLC	A	1	1
	MOV	A,LCHECK	1	2
	XRL	A,LPOLY	1	2
	RLC	A	1	1
	MOV	LCHECK,A	1	2
	MOV	A,HCHECK	1	2
	XRL	A,HPOLY	1	2
	RLC	A	1	1
	MOV	HCHECK,A	1	2
	CLR	C	1	1
	MOV	A,DATA	1	2
	RLC	A	1	1
	MOV	DATA,A	1	2
	DJNZ	INDEX,LOOP	2	3
	RET		2	1

$$N = 3 + 17 \times 8 = 139 \text{ 周期}$$

$$\text{@}12\text{MHz} = 139\mu\text{S}$$

$$\text{指令} = 18$$

$$\text{字节} = 31$$

	MC6801		机器周期	字节
	LDAA	# \$08	2	2
LOOP:	STAA	COUNT	3	2
	LDAA	HCHECK	3	2
	EORA	DATA	3	2

ROLA		2	1
LDAD	POLY	4	2
EORA	HCHECK	3	2
EORB	LCHECK	3	2
ROLB		2	1
ROLA		2	1
STAD	LCHECK	4	2
ASL	DATA	6	3
DEC	COUNT	6	3
BNE	LOOP	4	2
RTS		5	1

$N=45 \times 8 + 7 = 367$ 周期
@4MHz=367μS
指令=15
字节=28

Z8611		时钟周期	字节
LD	INDEX, #8	6	2
LOOP: LD	R6, DATA	6	2
XOR	R6, HCHECK	6	2
RLC	R6	6	2
XOR	LCHECK, LPOLY	6	2
RLC	LCHECK	6	2
XOR	HCHECK, HPOLY	6	2
RLC	HCHECK	6	2
RCF		6	1
RLC	DATA	6	2
DJNZ	INDEX, LOOP	12或10	2
RET		14	1

$N=20 + 66 \times 7 + 64 = 546$ 周期
@12MHZ=91μS
指令=12
字节=22

(2)从40个字节的块中搜索一字符

8051		机器周期	字节
MOV	INDEX, #41	1	2
MOV	DPTR, #TABLE	2	3
LOOP1: DJNZ	INDEX, LOOP2	2	2
SJMP	OUT	2	2
LOOP2: MOV	A, INDEX	1	2
MOVC	A, @A+DPTR	2	1
CJNE	A, CHARAC,	2	3
LOOP1	LOOP1		

$N=3 + 39 \times 7 + 4 = 280$ 周期
@12MHZ=280μS
指令=7
字节=15

MC6801		机器周期	字节
LDAB	# \$40	2	2
LDAA	#CHARAC	2	2
LDX	#TABLE	3	3
LOOP: CMPA	\$0,X	4	2

BEQ	OUT	4	2
INX		3	1
DECB		2	1
BNE	LOOP	4	2

OUT:

$N=7 + 40 \times 7 = 687$ 周期
@4MHz=687μS
指令=8
字节=15

Z8611		时钟周期	字节
LD	INDEX, #40	6	2
LOOP: LD	DATA, TABLE(INDEX)	10	3
CP	DATA, CHARAC	6	2
JR	Z, OUT	12或10	2
DJNZ	INDEX, LOOP	12或10	2

OUT:

$N=6 + 38 \times 40 = 1524$ 周期
@12MHZ=254μS
指令=5
字节=11

(3)将16位的字右移5位

8051		机器周期	字节
MOV	INDEX #5	1	2
LOOP: CLR	C	1	1
MOV	A, WORD+1	1	2
RRC	A	1	1
MOV	WORD+1, A	1	2
MOV	A, WORD	1	2
RRC	A	1	1
MOV	WORD, A	1	2
DJNZ	INDEX, LOOP	2	2

$N=1 + 9 \times 5 = 46$ 周期
@12MHZ=46μS
指令=9
字节=15

MC6801		机器周期	字节
LDX	#5	6	3
LDAD	WORD	4	2
LOOP: LSRD		3	1
DEX		3	1
BNE	LOOP	4	2
STAD	WORD	4	2

$N=10 \times 5 + 11 = 61$ 周期
@4MHz=61μS
指令=6
字节=11

Z8611		时钟周期	字节
LD	INDEX, #5	6	2
LOOP: CCF		6	1
RRC	WORD+1	6	2
RRC	WORD	6	2

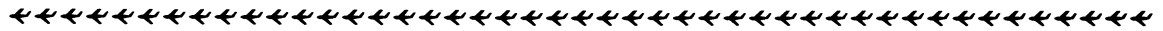
MC6801			机器周期	字节
	JSR	SUBR	9	2
	..			
SUBR: ..		RTS	5	1
		N=14周期		
		@4MHz=14μS		
		指令=2		
		字节=3		
Z8611			时钟周期	字节
	CALL	@SUBR	20	2
	..			
SUBR: ..		RET	14	1
		N=34周期		
		@12MHz=5.7μS		
		指令=2		
		字节=3		

通过上述对三种芯片的比较后,Z8611的优越性

是不言而喻的。唯一的无累加器的设计方案、强有力的中断结构、带有硬件奇偶校与检测、交互方式的 I/O、最大到16KB的内部 ROM、变化多端便利的变址寻址方式、短指令的寻址方式、整个外存可随意作为堆栈的灵活性、寄存器与 I/O 成“映射”关系从而不需专用的 I/O 指令,多种开发支撑芯片和仿真芯片、各种不同应用领域的封装形式,给应用者带来了编程序方便、灵活、占用内存空间少、程序执行时间快、选用芯片可塑性大一系列的优点,相信 Z8系列高性能八位单片机系列为我们广大使用者带来了又一种新的诱惑。

凡欲订购 Z8系列单片机和仿真系统的读者,请与深圳市振华路414幢电子器材大厦“中国电子器材深圳公司”联系,邮编:518031,电话:3354214,3354345,3350877;传真:3350876 电挂:8410,联系人:董伯明周翔

《Z8高性能单片机原理及应用》一书,定价22.00元,另加10%邮费,邮购地址:广州1501信箱9分箱“电子质量”杂志 邮编:510610



单片机主从式通信系统中的广播发送

北京市电信学校 江琪

用一台 IBM PC 机和若干台单片机可以构成主从式通信系统。该系统除了能够进行点对点的通信以外,还应该能完成广播式发送通信。所谓的广播式,即由主机发出信息,而接收的对象是所有的从机。

由 MCS—51系列单片机的硬件原理可知,利用串行口的工作方式2、3,可以很方便地与 IBM PC 机构成主从式多机系统,理论上从机的数量可以达到256台。多机系统中点对点通信的过程可简述如下:

- ①各从机置位 SM2位。
- ②主机将奇偶校验位设置为恒1校验,发送地址帧。
- ③各从机中断接收主机送来的地址帧,并与各自的地址比较,如不相同则中断返回,不进行任何操作;如相同,则复位 SM2位,回送该地址,准备接收主机发来的信息。
- ④主机在发送地址帧后,等待接收选中的从机回送的应答地址。在比较收到的地址与送出的地址相同后,则将奇偶校验位设置为恒0校验。开始向该从机发送信息了。
- ⑤在发送完所有的信息后,又重新恒置位奇偶校验位。而被选中的从机在收到所有信息后,也重新置位 SM2。

由上可见,点对点通信是由硬件和软件的配合而共同完成的。在硬件上,PC 机通过对 INS8250线路控制寄存器 LCR 中奇偶校验位的恒置1或恒置0,

MCS—51系列单片机 SM2位和 RB8位的设置保证了多机通信的进行。在软件上,给每个从机分别设置一个固定的、互不相同的地址号,比如01、02、……n,在主机送来一个地址帧后,各从机构响应中断分别将它与自己的地址号比较。这里比较地址相同否是软件的工作。比如12号从机,它的串行口接收中断服务子程序中判断地址相同否的程序段如下:

```

SIOR:CLR    RI ;由 0023H 处转来
          PUSH  A ;保护断点
          PUSH  PSW
          MOV   A,SBUF
          XRL   A,#12
          JZ    SIOR1 ;是本机号转 SIOR1执行
RETURN:POP  PSW ;不是呼叫本机,中断返回
          POP   A
          CLR   TI
          CLR   RI
          RETI
SIOR1:CLR   SM2 ;清“0”SM2位,准备接收信息
          MOV   A,#12 ;回送本机地址
          MOV   SBUF ,A
WAIT:JNB   TI,WAIT
          CLR   TI
          ..

```

如果我们将一个地址号(比如00)规定为特殊的地址,各从机均把它也认为是自己的地址号,则可以完成广播式发送。但是各从机软件处理00地址时,不应象点

对点时那样再回送地址号。因为如果各从机都回送自己的地址号,则会造成线路上信号的混乱;如果都回送00,则由于线路长短不一样,干扰情况不一样等等原因,可能导致主机收到的信号不是00了。所以一般主机在发出地址帧00以后,就不再回收地址而直接广播发送信息了。这时12号从机的串行口接收中断服务子程序判断地址相同否的程序段如下:

```

SIOR:CLR  RI
      PUSH A
      PUSH PSW
      MOV  A,SBUF
      JZ   SIOR0
      XRL A,#12
      JZ   SIOR1
RETURN:POP PSW
      POP  A
    
```

```

CLR  TI
CLR  RI
RETI
SIOR0: CLR SM2
      AJMP SIOR2
SIOR1: CLR SM2
      MOV  A,#12
      MOV  SBUF,A
WAIT:  JNB  TI,WAIT
      CLR  TI
SIOR2:..
    
```

这种简单的广播式发送信息的实现,是以牺牲了一个从机数量而换来的。即此时,从机的数量最多是255台。按照上述的方法,稍对软件进行修改还可以完成群式(点对多点式)发送。

给单片机加装 V/F F/V 转换器

39506部队77分队 岳海军

V/F 转换器有其良好的精度、线性和积分输入特性,常能提供其它类型转换器无法达到的性能,采用 V/F 转换器与单片机接口有下列优点:

首先是它的抗干扰性能好,V/F 转换过程是对输入信号不断积分因而能对噪声和一些变化的输入信号进行平滑,其次是接口简单占用计算机资源少,V/F 转换器与计算机接口很容易采用光电隔离,它还可以调制在射频信号上进行无线传播实现遥测。

本文着重介绍使用 LM331与单片机的接口,它的性能特点如下:

1. 功耗小,5V 下典型值为15mW;
2. 动态范围宽,满量程频率范围从1Hz到100kHz,10kHz 满量程频率下最小值为100dB;

3. 脉冲输出与所有逻辑形式兼容。

LM331引脚功能如下:

1. 电流输出;2. 基准电流;3. 频率输出;
4. 接地;5. RC 输入;6. 比较器阈值;
7. 比较器输入;8. 电源正。

应用电路见图1。信号经双“T”滤波后接入比较器输入端,在2脚加一个可变电阻用来调节基准电流,以校正频率,变换器的输出端与单片机的定时/计数器输入端连接即可,在环境恶劣时可采用光电隔离,当需要远距离传送时可采用双绞线和光纤或无线电传输。

在用作 F/V 转换时见图2。信号由单片机的 P1.7 口输出(本人用的是8031),经 C、R 网络输入到比较器阈值端,与 V/F 相同在1脚输出平均电流经 R、C 网络滤波后,得到与频率成正比的直流电压。

至于软件方面,完全可以自行编制一个简单的定时/计数器的程序即可。

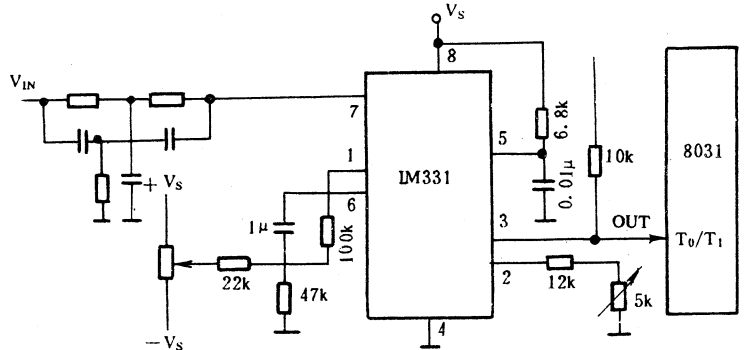


图1 V/F 转换电路原理

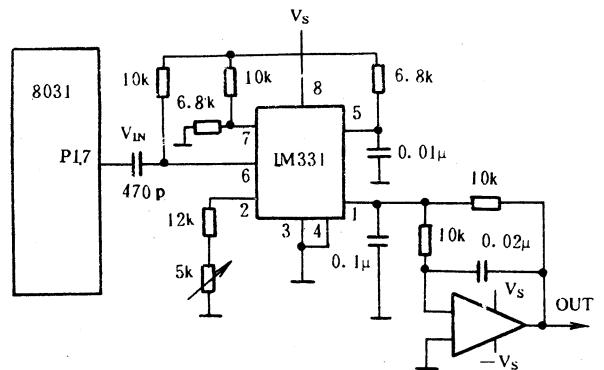


图2 F/V 转换电路原理

一种防止单片机程序运行失常的软件对策

盐城工业专科学校(224002) 黄稳山

8031单片机虽然有较强的抗干扰性能,但在工业现场使用时,大量的干扰源虽不能造成硬件系统的损害,但仍然有可能使系统程序运行失常,造成程序飞逸,盲目运行。因此,防止程序运行失常的软件对策的任务,是要监视程序运行状态,当发现程序运行失常时,及时引导系统恢复到初始状态或正常工作状态,通常的做法是设置监视跟踪定时器,监视程序运行。例如,使用8155的定时器所产生的“溢出”信号作为8031外中断源INT1,用555定时器作为8155中定时器的外部时钟输入,8155定时器的定时值稍大于主程序正常循环时间。在主程序中,每循环一次,对8155定时器的定时时间常数进行刷新,当程序运行失常,不能刷新定时器时间常数而导致定时中断,利用定时中断服务程序使系统复位。

上述对策需增加硬件电路,而且当程序飞逸陷入死循环中含有刷新定时器时间常数的程序时,则不能有效地使系统复位。本文设计了一种防止程序运行失常的软件对策,不需增加硬件,并在绢丝厂煮茧温度单片机控制中应用,有效地提高了系统抗干扰性能,现简介如下。

绢丝厂煮茧温度控制,其工艺要求是将煮茧水温控制在设定值附近,根据测量值与设定值的偏差进行PID运算,调节电动调节阀阀门开度,从而调节蒸汽流量,使煮茧水温稳定。因此该系统是一个单回路控制系统,其主程序是一个A/D测量转换,数字滤波、查表、测量值显示的循环程序,定时采样、PID运算及输出则采用采样定时中断服务程序进行。在软件设计中,设置了定时中断监视器来防止程序运行失常,定时中断监视器采用定时器0进行,定时器1则作为采样定时使用,设计定时器0的中断为高优先级中断,定时中断的时间稍大于主程序正常循环的时间,其主程序流程图如图所示。

主程序流程图

DCZD:CLR TR0

MOV TH0, #3CH ;重置定时时间常数

MOV TL0, #0B0H

DJNZ 3AH, DCZ2 ;判定定时中断监视周期到否,不到,转中断返回

MOV 3AH, #0FAH ;到,重置定时中断监视周期

MOV DPTR, #DCZ1

MOV 3BH, DPL

MOV 3CH, DPH

MOV DPTR, #LOOP

MOV 3DH, DPL

MOV 3EH, DPH

POP 3FH

POP 40H ;将原栈顶内容弹出,以免多次中断后堆栈中数过多

PUSH 3BH

PUSH 3CH ;压入中断返回地址

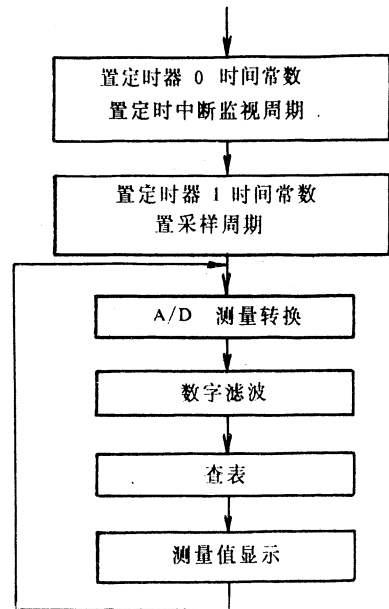
RETI

DCZ1:PUSH 3DH

PUSH 3EH ;压入中断返回地址

DCZ2:SETB TR0

RETI ;中断返回



TH0和TL0中存放定时时间常数,3AH中存放定时中断监视周期,3BH、3CH中分别存放标号DCZ1的低8位和高8位地址,3DH、3EH中分别存放标号LOOP(即主程序中A/D测量转换入口)的低8位和高8位地址。考虑到定时中断监视有可能在低优先级中断(即采样中断服务程序)时进行,而8031中断系统有两个不可寻址的优先级状态触发器,在定时监视中断返回主程序时,必须清0响应中断时所置位的优先级状态触发器,因此设计了两次PUSH、RETI指令,以保证定时监视中断返回主程序后,中断得以继续进行。这样,定时中断监视周期一到,不论系统是处于正常工作状态还是处于由于干扰引起的运行失常状态,均使系统回复到主程序A/D测量转换入口处进行。由于本系统主程序是一个不断测量显示、定时采样的循环程序,实践表明,这种定时监视中断对系统测控功能无影响,但却能有效地防止由于干扰引起的PC失控。

在该系统软件设计中还设置了软件陷阱,与定时

中断监视器一起,形成了防止程序运行失常的双保险。在非程序区反复用 LJMP #0000H 指令填满,即反复用 020000020000.....代码填满,而在初始化程序处,对上电操作、硬件复位还是由软件陷阱引起的转移进行判断,使程序自动转移到主程序 A/D 测量转换入口处,这样不论 PC 失控后指向非程序区那一个字节,最

终都能导致程序回到正常工作状态。

上述防止程序运行失常的软件对策,在单片机煮茧温度控制中应用,有效地提高系统抗干扰能力。在过程控制系统中,只要其主程序是一个连续测量定时采样的循环程序,这种防止程序运行失常的软件对策是可取的。

NKP-824G 打印机一故障维修方法

陕西汉中师范学院计算中心(723001) 蒋国权 张宏杰

国营七五四厂开发生产的“售票系统”由计算机售票系统和 NKP-824G 票据打印机组成。这里主要介绍票据打印机的一种故障检测维修方法。

1. 故障现象:

打印机打印出的数据与正确数据不一样。如 4 变 0、5 变 1、6 变 2、7 变 3、小数点变 * 等错误数据。

2. 故障分析:

根据打印机打印出的数据,分析归纳如下:

数据	ASC 码	二进制码	错误二进制码	错误数据
0	30	0011 0000	0011 0000	0
1	31	0011 0001	0011 0001	1
2	32	0011 0010	0011 0010	2
3	33	0011 0011	0011 0011	3
4	34	0011 0100	0011 0000	0
5	35	0011 0101	0011 0001	1
6	36	0011 0110	0011 0010	2
7	37	0011 0111	0011 0011	3
8	38	0011 1000	0011 1000	8

9 39 0011 1001 0011 1001 9

由此可得:第 3 位数据始终为 0,即第 3 位数据对应的信号线(DB2)始终为低电平,应检查与此有关的电路。

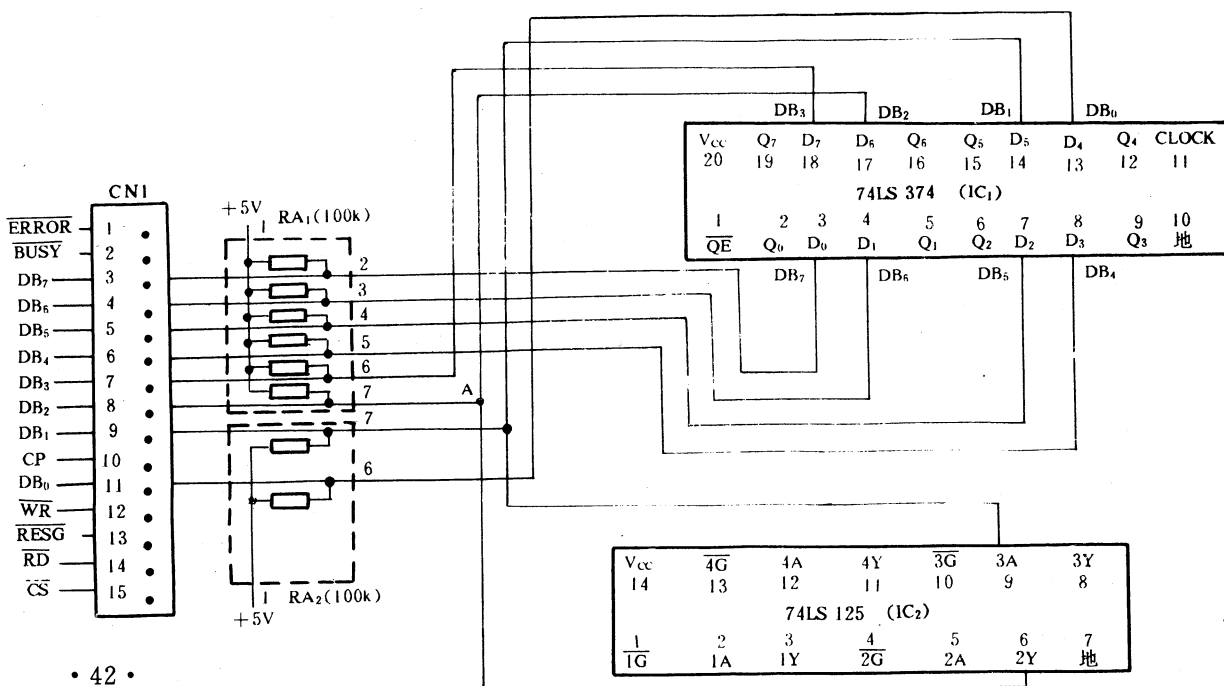
3. 故障检测与维修:

3.1 检测接口 CN1 的第 8 根信号线(DB2)对地电阻为 0,第 11 根(DB0)、9 根(DB1)、7 根(DB3)、6 根(DB4)、5 根(DB5)、4 根(DB6)、3 根(DB7)对地电阻均为 9k。

3.2 检测 IC1(74LS374)的第 17 脚(D6)对地电阻为 0,第 3 脚(D0)、4 脚(D1)、7 脚(D2)、8 脚(D3)、13 脚(D4)、14 脚(D5)、18 脚(D7)对地电阻均为 9k。

3.3 检测 IC2(74LS125)的第 6 脚(2Y)对地电阻为 0,第 3 脚(1Y)、8 脚(3Y)、12 脚(4Y)对地电阻均为 4k。

3.4 断开 IC1 的第 17 脚与 IC2 的第 6 脚的连接线(如原理图所示),断点为 A,检测 IC1 的第 17 脚对地电阻为 9k,说明 IC1 正常;再检测 IC2 的第 6 脚对地电阻仍为 0,从原理图中分析,IC2 的第 6 脚并未与其它器件相连,说明 IC2 损坏。更换 IC2 后,打印机恢复正常。



M2024打印机双向打印时错位的解决办法

济南军区司令部军训部 陆钦俭

使用时间比较长的 M2024 打印机,双向打印时一般都会出现错位的现象,特别是打印汉字和表格时,由于上下两部分的错位,使之不很美观。这种现象的产生一般是由于机械误差造成的,用户完全可以通过调整打印机的有关部件来消除错位。下面谈一下调整的具体方法和步骤。

第一步,检查小车在导轨上运行是否顺畅。打印机长时间使用后,如果不经常保养,导轨上油泥一般较多,影响小车的运行。可用软布将油泥擦拭干净,保证小车运行自如。

第二步,拆下小车上的打印头,在打印头的下方,

小车底部有一将小车固定在皮带上的螺丝,如果该螺丝松动,会使小车换向时产生误差。检查该螺丝是否有松动,如有松动,应拧紧。完成以上二步,错位现象一般可以消除,如还有错位可进行第三步调整。

第三步,拆下打印机外壳,调整左侧的皮带轮固定螺丝。由于长期使用和老化,会使皮带的张力减小,皮带变得松弛,这就要调整左侧皮带轮的固定螺丝,使皮带张力合适。调整时要细心,并边调整边试机,以达到最佳状态。

以上调整方法实际上不仅适用于 M2024 打印机,只要与之结构类似的打印机均适用。

Super—AT286微机硬盘软故障排除一例

江苏通州职业高中(226300) 费毅

一、故障现象

软盘启动正常,硬盘启动时,屏幕显示:DRIVE NOT READY ERROR Insert BOOT diskette in A: Press any key when ready.当从软盘启动后转入硬盘,或对硬盘执行 dir,del,copy,format 等 DOS 命令时,屏幕上均显示错误信息 Invalid drive specification.

二、故障分析

软盘能正常启动,说明计算机系统板硬件电路完好,查看 CMOS 系统配置信息表,各参数都设置正确,因此可断定故障出在硬盘上。从软盘启动 DOS 后,调用 FDISK.COM,选择 4,Display Partition Information (显示分区信息),则屏幕显示:No Partitions defined Press ESC to return to Fdisk Option.由此说明故障原

因是由于硬盘分区表信息遭到破坏。

三、故障排除

从软盘启动 DOS 后,调用 FDISK.COM,先选择 3,“Delete DOS partition”,将原 DOS 分区删除,再选择 1,“Create DOS partition”,重建硬盘分区表,然后再用 FORMAT C:/S 命令对硬盘进行格式化,硬盘即可正常启动。但上述做法会使硬盘上的数据全部丢失,为此可采用下列方法来修复:调用调试工具 DEBUG,从另一型号相同、DOS 版本相同且硬盘分区也相同的正常微机上,将硬盘主引导扇区的内容以文件形式拷到一张软盘上,然后再拷入故障微机硬盘的 0 道 0 头第一扇区,关机后重新启动系统,故障即可消除(具体做法在许多文章中都有介绍,在此不再赘述)。

F BASIC 语言的游戏程序编程技巧

第六讲 游戏程序实例分析(下)

山东苍山机械电子化学工业局(277700) 于春

六、追踪主程序流向,详细分析程序

现在我们根据主程序的执行过程,详细分析程序。分析程序前,最好列出变量表及变量赋值表,供分析程序时参考。当每确定一个变量的作用时,则在变量

表中注明,以利阅读理解。

变量表如下:(由 20~30 行得出)

(1)单字符变量

C、D、H、I、K、L、M、N、O、P、Q、S、T、U、V、X、Y

(2) 双字符变量

MA, PP, X0, X1, Y0, Y1

(3) 下标变量

D(0) D(1) V(0) V(1) S(0) S(1) SC(0)
SC(1) F(0) F(1) F(2) F(3) F(4) F(5)
F(6) F(7) F(8)

变量赋值表:(由40~60行得出)

C=7 L=9 M=1 N=3 U=3 V=7 MA=3
X0=70 Y0=80 X1=110 Y1=60 D(0)=3
D(1)=7 V(0)=2 V(1)=2 S(0)=4 S(1)=4
F(0)=3 F(1)=3

列好变量表后,开始分析程序

1. 第一周期(我们把主程序执行一次称为程序运转一个周期,下同)

(1)由90行转1070行,可知这一模块的功能是打印标题画面和游戏对手选项。1090行打印星空,1110、1130行定义8[#]、9[#]卡通为0[#]、1[#]动作。1140行打印选项提示。1150行读键盘输入。若键入“N”为单人游戏,M=1;若按其它键为双人游戏,M=-1。选项后,返回主程序。假定我们选双人游戏,则M=-1。把M=-1写入变量赋值表。(今后每个变量得到一个新值,都要及时写入赋值表,以利分析。在以后的叙述中不再提示,请读者自动写入)。

通过1070~1170行程序段的分析,可大致确定以下几个变量的作用:

P: P=0,1。为0[#]、1[#]动作转换变量。

D(P): 为0[#]、1[#]动作的运动方向变量。

M: M=1,-1。为游戏者单、双人选项变量。

(2)由100行清屏,打印星空。令P=0。

(3)120行,I=P+6=6,F(P)=F(0)=3(该行中使用了IF F(P)THEN……语句,其作用是如果F(P)<>0则执行THEN后面的语句。IF F(P)相当于IF F(P)<>0……。这种表达形式是电脑约定的简写方式,由于F(0)=3,所以转680行。

(4)这时P=0,M=-1,P<>M。假定未按操纵器的任何键则返回。

(5)由120行跳转150行。

(6)由160行转530行。由于游戏刚开始,没有卡通相撞,返回160行。

(7)以上我们曾分析过模块7可先删去。故暂时删去170行。

(8)180行。这时F(8)=0,因此转940行。

(9)940~980行。C=7,MA=3。定义7[#]卡通为2[#]、3[#]动作,并令其运动。

(10)190行。P=1,跳转120行,返回开始后的状态。第一循环结束。

通过第一次循环,启动了2[#]、3[#]动作(太空站)开始运动,同时确定了以下变量的作用。

D: 先为卡通碰撞记录变量,后为辅测碰撞卡通的动作编号。

O: 为主测碰撞卡通的动作编号。

C: 为干扰卡通的编号。

MA: 为干扰动作数量控制变量。

2. 第二周期

(1)120行。P=1,I=7,F(1)=3,转680行。假定这时按2[#]操纵器方向键。

(2)通过680行,F(1)=0,确定1[#]动作的起始座标为(110,60)。返回120行,转150行。

(3)通过160行转530行,返回。F(8)+1=1,转940行。

(4)这时2[#]、3[#]动作都没结束,返回。

(5)P=0,跳转120行。第二循环结束。

3. 第三周期

(1)P=0,I=6,F(0)=3,转680行。

(2)假定按1[#]操纵器方向键,则通过680行F(0)=0。确定0[#]动作座标为(70,80)。返回120。

(3)转150,转530,没有碰撞,返回。F(8)=1,转940。

(4)这时2[#]、3[#]动作仍没结束,返回。

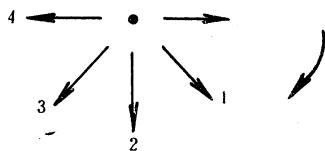
(5)P=1,跳转120行。第三次循环结束。这时0[#]、1[#]动作已分别确定了起始座标。

4. 第四周期

(1)P=1,I=7,这时F(1)=0则执行140行。

(2)由140行转210行。

(3)210行。由于我们已按方向键,假定方向为右,则S=1,通过250行S=3。注意250行完成卡通运动方向的计算。这一行语句等价于IF S=1 T. V=3……形式的8行语句。请读者记住这一编程技巧。它计算运动方向的过程是:当S=1时,只有(S=1)=-1,其它各项都为0,故有S=-(-1)*3=3,即方向键值等于1,运动方向为向右。260~340行程序是该模块中独立的一段程序。它的作用是使0[#]、1[#]动作的转向每次45°分步完成。如正在向左运行时,若按了下行键,则分两步转到向下。若正在向右运动,按键向左则分四步从右转向左,转向过程如下图;



分步转向原理留待下一步分析。

运动方向S的值通过方向转移变量Q逐步转给D(P),再通过360行定义1[#]动作运动。若在按方向键的同时按A键,则定义7[#]动作运动,(即发射镭射),运动起点在1[#]动作中心点外4点的位置,运动方向同1[#]动作,返回150行。

(4)150行,若7[#]动作结束则令其消失,对应画面是未击中目标,令镭射消失。

(5)160行转530检查0[#]、1[#]是否与其它动作相撞,或除1[#]外的各动作是否被击中,假定0[#]被击中则由

570行 D=0、O=7转610行。

(6)由610行,求0[#]、7[#]动作之间的距离,若在X方向大于等于8,同时Y方向也大于等于8,则判定未击中,返回;若有一个方向的距离小于8,同时0[#]、7[#]动作运动未结束,F(0)=0,则判定0[#]被击中。

(7)判定0[#]被击中后,则令7[#]动作消失,令F(0)=1,转630行。

(8)由于D=0<2,则SC(O-6)=SC(1)=SC(1)+10=10,作用是给1[#]游戏者加10分。下标变量SC(O)、SC(1)为记分变量。返回570行,返回160行。

(9)由180行转940行,检查2[#]、3[#]动作是否完成,若动作完成则重新启动。可以看出,每一次循环都要GOSUB940对2[#]、3[#]动作检查一次,这是例行公事,以后不再叙述。

(10)P=0,转回120行。

5. 第五周期

(1)P=0,I=6,F(0)=1,转670行。

(2)670行,首先令F(0)=2,定义0[#]动作为10[#]卡通(爆炸),在原运动方向上运行10~14点,同时发声,令S(0)=S(0)-1=4-1=3,由于上一周期判断0[#]动作被击中,所以演示其被击中的景象(下标变量S(0)、S(1)记录0[#]、1[#]动作的游戏机会次数)返回120行,转150行。

(3)转530行,由于0[#]刚被击中,则返回,P=1-0=1,转120行。

按照上述步骤,读者可依次分析,不难读懂每一段程序。下面分析程序难点和有关变量。

七、程序难点和有关变量分析

1. 260~340行程序段

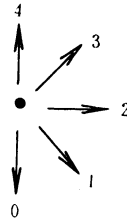
在第四周期分析中,曾提到这段程序的作用是使0[#]或1[#]动作分步转向,现在讨论其分步转向过程。

(1)程序开始运行时D(0)=3,0[#]向右运动。假定我们按向下键则S=5,通过260行,Q=3。这时S>Q,同时S<=Q+4,符合290行的条件,故转320行,Q=Q+1=4,转340行。这时PP=0,所以Q=4,D(0)=4,则0[#]开始向右下方向运动。

(2)第二周期管理1[#]动作,略过。第三周期:S仍等于5,则通过260行Q=4,仍满足290行条件,Q=Q+1=5,转340行。PP=0,Q=5,D(0)=5,从而完成了从向右运动到向下运动的转向。

(3)若在向下运行时,我们按了上行键,则S=1。当程序执行到奇数周期时(0[#]动作管理周期),由260行Q=5,Q>4,满足280行条件转310行,满足S>=Q-4,和S<Q的条件,则转330行,Q=Q-1=4,通过340行D(0)=4,0[#]动作由下行改向为右下;下一周期,由260行Q=4,通过300转330行,Q=Q-1=3,再通过340行,D(0)=3,0[#]动作由右下改向为右行;如此再经过四个执行周期,0[#]动作改为向上运动。其转向过程如图:

总之,这段程序的作用就是控制卡通分步转向,以避免方向突变,失去真实感。



2. 390~520行程序段

在跟踪分析中未讨论该段程序,现分析如下:

(1)程序的使用时机

从130行可知这段程序只在M=P时才调用。我们已知道,P取值始终为0、1,M的取值则由选项时确定为1或-1,因此,只有在M=1时才能满足M=P的条件。而M=1是游戏者选电脑为对手进行游戏。由此可知,这段程序只有在单人、1[#]动作管理周期内使用,因此它是1[#]动作的控制管理程序。

(2)分析程序的作用

确定了程序的使用对象后,就可分析作用了。由400行可知,若K>0则返回,后续程序不起作用。现假定K=0则执行410行令T=0、D=0,测量0[#]、1[#]间的距离。由于卡通的极限座标为0和255,我们均选取最大值,则有X_{max}=19、Y_{max}=19,由420、440、460三行得D=(3+5)/2=4。通过470行S=2,假定执行该段程序时1[#]运动方向为右,即D(1)=3,则有O=D-D(1)=1。再假定0[#]、1[#]运动速度不同则T=4。由于19*19+19*19>200则返回主程序。由此得出结论,当0[#]、1[#]之间距离较大时,该段程序不起作用。

下面再讨论两者距离较小的情况。

设X=8、Y=9,由420~490行仍有D=4、S=2、T=4。500行有效的条件是:7[#]动作结束或RND(10)=0则令T=8,否则T=4。现在X*X+Y*Y=145不满足510行条件,故执行520行K=RND(L),跳转220行,这时S=2、T=4或T=8,则由220~380程序段的作用,使1[#]动作转向或调整速度或发射,可见这段程序就是替代210行改人控操纵器产生T、S值为程序自动产生,从而控制1[#]产生各种动作。程序中引入K值的目的是调整控制1[#]动作的周期间隔,K值越大、间隔周期越长。K值的大小受游戏水平变量L的控制,L越小、K值越小,从而达到游戏水平越高,1[#]动作改变的越频繁,相应游戏难度也越大的目的。

3. 800~930行程序段

在以上的讨论中,我们抛开了这段程序。在整个程序中就数这段程序最难理解。实际上这是编程者向使用者故意卖的一个关子(出难题)。据我分析,原程序中应还有一段语句,功能是随机打印207号背景图案,作为0[#]、1[#]动作运动的固定干扰物,犹如太空中的大陨石或小行星。它与运动干扰物火球、苍蝇的干扰效果不同,当卫星碰到固定干扰物后,卫星爆炸而干扰物依然

存在。(运动干扰则与卫星同时爆炸)。但它能用镭射击毁,且数量一定,击毁后不再再生。

现在,我们可以补充一段程序以返还本来面目。补充程序如下:

```
110 F. I=0 TO 9:LOC. RND(27),RND(23):  
P. CH.(207):N.
```

补充了110行程序,800~930程序段的作用就不难分析了。

当程序执行到810行,首先计算 $P^*(0,1)$ 动作在BG面的坐标(X,Y)。若 P^* 不在BG面内,则转出。否则检查(X,Y)点是否有207号背景图案,若有则令 $F(P)=1$,由120行转670行显示爆炸(0^* 或 1^*);若没有则转880行,检查镭射运动是否结束。若镭射运动未结束,则检查镭射在BG面的坐标点上是否有207#背景,若有则表示被击中,令该点的207#背景图案消失。否则返回。

4. F数组下标变量的分析:

F(8)数组共有9个下标变量,用途分配如下:F(0)、F(1)分别记录 0^* 、 1^* 动作(卫星)的运动状态;F(2)~F(5)分别记录动态干扰物的运动状态;F(6)、F(7)分别记录 0^* 、 1^* 动作的速度调整情况;F(8)为定义苍蝇启动变量。

各下标变量在程序执行中的变化情况如下:

•F(0)、F(1):以F(0)为例。初值 $F(0)=3$;通过680行运动后 $F(0)=0$;若与其它动作相撞或被击中 $F(0)=1$ (590,610行);被击中后通过670行变为爆炸, $F(0)=2$,爆炸动作结束后通过710行 $F(0)=3$;再启动运行 $F(0)=0$ 。

•F(2)~F(5):以F(2)为例。初值 $F(2)=0$;若被击中或发生碰撞则 $F(2)=1$ (590,610行);若本次定义的动态干扰物消失但未全部消失则 $F(2)=2$;若全部消失则 $F(2)=0$ 。

•F(6)、F(7):以F(6)为例。初值 $F(6)=0$;若按B键调整运行速度则 $F(6)=1$ (220行);速度调整后 $F(6)=0$ (350行)。

•F(8):初值 $F(8)=0$;当动态干扰消失MA个后 $F(8)=1$ (1010行);当出现一个苍蝇后 $F(8)=0$ (1020行)。

到此为止,“星际大战”游戏程序全部分析完毕,各变量的作用也已基本确定,想必读者的变量表也已填满。兹将变量表列出供参考。

C:动态干扰卡通编号。值取3、5、7。

D:在390~520行调整 1^* 动作的方向。在540~650行记录被碰方的动作编号。

H:游戏者的最高积分。

I:镭射动作编号。 6^* 、 7^* 。

K: 1^* 自动运动间隔周期调整变量。

L:游戏技能评价。初值 $L=9$ 。水平越高L值越小。由9→1。当 $L=0$ 后则重新赋值 $L=5$ 。

M:单、双人记录。单人 $M=1$;双人 $M=-1$ 。

N:向U赋值变量,也可不用,在1020行中改写U

$=MA+1$ 。

O:记录发生碰撞或被击中时被测动作的编号。

P: 0^* 、 1^* 动作转换变量。

Q: 0^* 、 1^* 分步转向控制变量。

S:操纵器的方向键值和方向值。

T:操纵器的A、B键值。按A键 $T=8$,发射;按B键 $T=4$,调速。

U:动态干扰消失记录变量。

V:调速辅助变量。

X、Y: 0 、 1 、 6 、 7 号动作在BG面的坐标。

MA:一次出现动态干扰的个数,MA最大可取4。

PP:Q值调节辅助变量。

X0、Y0: 0^* 动作起始座标。

X1、Y1: 1^* 动作起始座标。

D(0)、D(1): 0^* 、 1^* 动作的运动方向。取值1~8。

V(0)、V(1): 0^* 、 1^* 动作的速度。取值2、5。

S(0)、S(1):双方游戏机会记录变量。

SC(0)、SC(1):双方得分记录。

F(0)~F(8):见以上介绍。

八:“星际大战”编程技巧小结

通过以上分析,我们可学习以下几点编程技巧。

1. 清零技巧。20行中令23个变量连等实现清零比令每个变量等于零节约内存50个单元。

2. 上一场游戏积分的转移技巧。80行巧妙地记录了上一场的最高积分转移到下一场。

3. 运动方向的计算技巧。250行用一行语句代替了8条IF—THEN语句,完成了S的计算。

4. 分步转向技巧。260~340行。

5. 卡通的自动处理技巧。400~520行。

6. 提示打印定位技巧。750行仅使用一个定位语句就完成了三行提示的打印定位。

九、“星际大战”游戏的补充建议和说明

1. 增设110行随机打印固定干扰是程序必需的。

2. 建议积分超过50分时奖励一次游戏机会。

3. 建议击毁一个固定干扰物加10分。

4. 建议删去20行。因为程序每次运行均对变量清零,故20行似乎多余。

5. 建议卫星运行和发射时配有音响,以增强效果。

6. 说明:欲增加游戏难度,可调整MA数值和增加固定干扰物的数量。

根据以上建议,可加入以下程序。

```
110 F. I=0 TO 9:LOC. RND(27),RND(23):
```

```
P. CH(207):N.
```

```
120 I=P+6:IF SC(I-6)>50 T. SC(I-6)=SC(I-6)-50:S(I-6)=S(I-6)+1
```

```
125 IF F(P) T. ON F(P) GOS. 670,700,680:G. 150  
365 PL. “O3E1”
```

```
375 PL. “O1#D0REO0#D3”
```

```
915 SC(I-6)=SC(I-6)+10
```

以上程序已合理安排了行号,读者可直接加入。

集成滤波电路及用法(二)

李兰友

二、其它类型集成电路滤波器

1. VCF-2 BL36 50K

VCF-2是电压控制型集成电路滤波器,仅仅加上电源和控制电压即可正常工作。BL36 50K是六阶巴特渥斯低通滤波器,最高截止频率50kHz,控制范围为1000倍。 f_c 范围为50Hz~50kHz。

主要参数:

特性 六阶低通巴特渥斯特性。

衰减特性 36dB/oct
截止频率范围 50Hz~50KHz
截止频率误差 $\pm 5\%$ 以下
截止频率控制 电压,10mV~10V
失真率 无
噪声 $120\mu\text{V}$ 以下
失调电压 $\pm 50\text{mV}$ 以下
工作电压范围 $\pm 12\text{V}\sim\pm 16\text{V}$
消耗电流 $+38\text{mA}, -35\text{mA}$

VCF-2 BL36 典型应用接法如图12所示。实测特性如表6,频率特性曲线如图13。

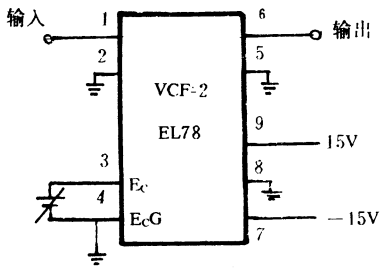


图12

表6 VCF-2 BL36 50K 实测参数

控制电压 (V)	f_c (Hz)	失真率 (%)	噪声 ($\mu\text{Vp-p}$)	失调电压 (mv)	消耗电流 (mA)
10	50k	0.024	100	0.70	+32.5 -30.5
1	5k	0.016	82	0.68	+27.6 -28.6
0.1	500	0.021	82	0.52	+27.2 -28.5

由于采用直流控制电压,因而控制电压地和电源地尽可能接在一起。

控制电压变化会引起失调电压变化,但变化值很

小,不会成为问题。

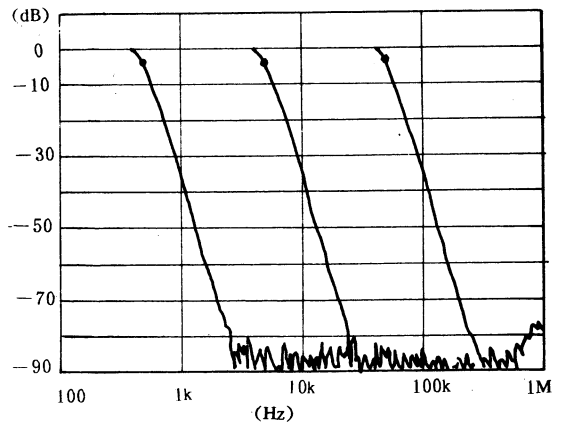


图13

2 DT-408DC2

DT-408DC2是数字同步型2级2阶状态变量滤波器,通过外接电阻可设定增益和Q值,并由2位BCD码控制频率,频率范围为1kHz~159kHz。

主要特性参数:

衰减特性 24dB/oct
截止频率范围 1kHz~159kHz
截止频率控制方式 2位BCD码
失调电压 $\pm 20\text{mV}$
工作电压范围 $\pm 15\text{V}$
消耗电流 $\pm 50\text{mA}$

典型应用接线如图14所示。 $f_c=1\text{kHz}\sim 159\text{kHz}$ 表7为图14电路的实测参数,图15为特性曲线。

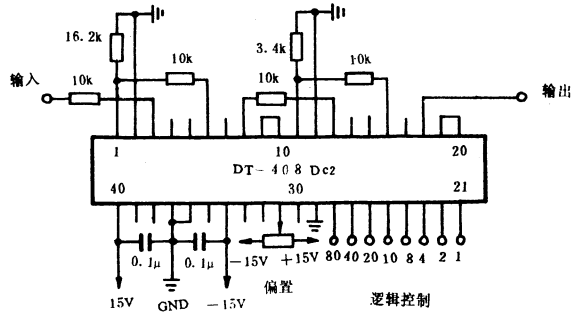


图14 DT-408DC2基本电路

表7 DT-408DC2实测参数

fc (Hz)	失真率 (%)	噪声 (μV)	失调电压 (mV)	消耗电流 (mA)
100k	0.0033	24	2.2	+58.6 -54.3
10k	0.0017	14	2.3	+58.6 -54.3
1k	0.0017	14	2.4	+58.6 -54.3

DT408DC2可以使用外部运放,各接地端与电源地共接。该 IC 体积大,耗电多,但突出优点是性能优良,使用简便。

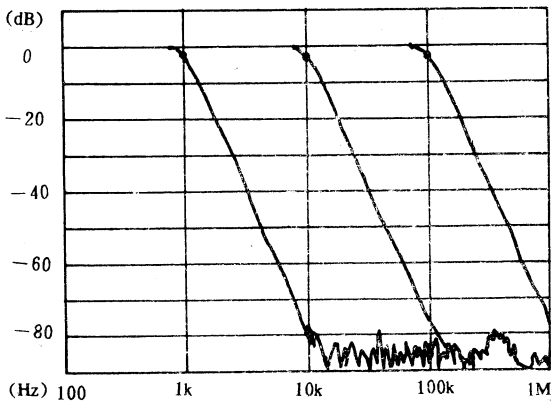


图15

3. FLJ-UR4LB2

FLJ-UR4LB2系列是通过外接电阻设定截止频率的 RC 滤波器。该系列 IC 可接成低通、高通、带通、带阻滤波器。外型封装为20引脚 SIP 型。主要参数如下:

- 衰减特性 42dB/oct
- 截止频率范围 400Hz—20kHz
- 截止频率误差 ±3%以下
- 截止频率控制 4个同值电阻
- 失真率 0.01%以下
- 噪声 140μV 以下
- 失调电压 ±30mV 以下
- 工作电压范围 ±5V~±18V
- 消耗电流 ±16mA

典型应用接法(4阶低通滤波器)如图16所示,图

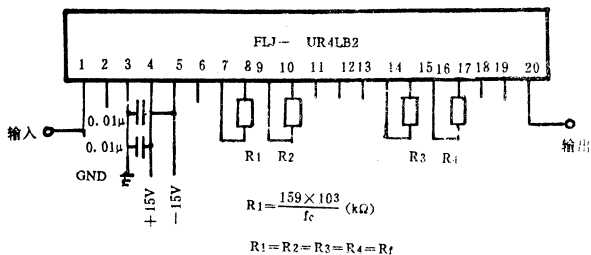


图16

中, $R_1=R_2=R_3=R_4=R_f$,
 $R_f=159 \times 10^3 / f_c$ (kΩ)

图16电路的实测参数如表8所示,频率特性如图

17。

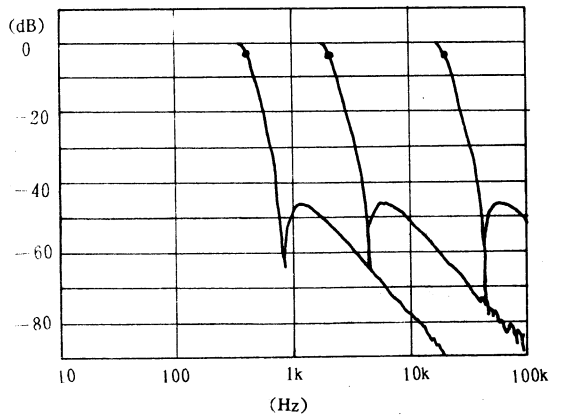


图17

FLJ-UR4LB2用四个外接电阻设定截止频率,电路设计很方便,噪声和失真也很小。

表8.

Rf (kΩ)	fc (Hz)	失真率 (%)	噪声 (μV)	失调电压 (mV)	消耗电流 (mA)
7.87	20k	0.0031	30	10.2	±16.1
78.7	2k	0.0026	21	10.2	±16.1
402	400	0.0033	21	10.2	±16.1

4. RT-8FLA

RT 系列 IC 是通过外接电阻设定截止频率的 RC 滤波器。该系列 IC 依衰减特性不同,有 RT-8FLA、RT-8FLB 及带通滤波器 RT-3BP 等型号,而且每种型号又依截止频率的设定范围的差别分别标以1、2等标志。

RT-8FLA2主要参数:

特性电阻型八阶契比雪夫滤波器

- 衰减特性 135dB/oct
- 截止频率范围 100Hz—20kHz
- 截止频率误差 ±3%以下
- 截止频率控制 外接电阻
- 失真率 0.05%以下
- 噪声 140μVrms 以下
- 失调电压 ±10mV 以下
- 工作电压范围 ±5V~±18V
- 消耗电流 ±40mA

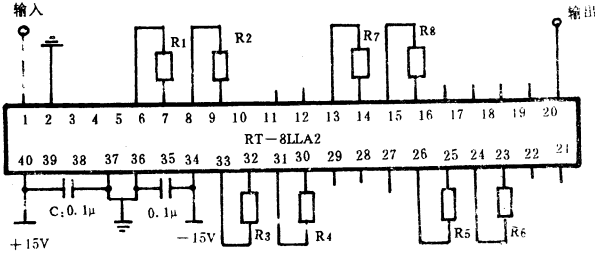
典型应用接法如图18所示。电阻选择与 fc 的关系

式为:

$$R_f = 159 \times 10^3 / f_c \text{ (k}\Omega\text{)}$$

$$R_1 \sim R_8 = R_f$$

$$\text{电容选择: } C = 0.01 \mu\text{F}$$



$$R_1 = \frac{159 \times 10^3}{f_c} \text{ (k}\Omega\text{)} \quad R_1 \sim R_8 = R_F$$

图18 RT-8FLA2.

图18实测8FLA2参数如表9,特性曲线如图19所示

表9

Rf (Ω)	fc (Hz)	失真率 (%)	噪声 (μV)	失调电压 (mV)	消耗电流 (mA)
7.87k	20k	0.0075	86	15.2	+33.7 -34.1
78.7k	2k	0.0064	68	15.2	+33.7 -34.1
787k	200	0.0063	64	15.2	+33.7 -34.1

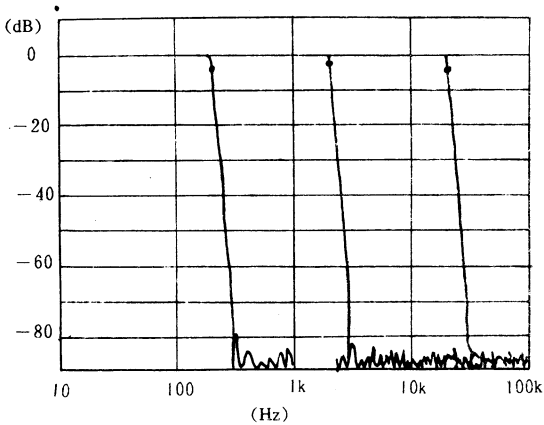


图19

RT-8FL 系列同属于 RC 滤波器,衰减特性较好,但设定截止频率时,需8个同样数值的电阻。

5 VCF-2 EL78 50K

VCF-2 EL78 50K 也是一种电压控制型集成电路滤波器,EL78 50K 是六阶椭圆低通滤波器,最高截止频率为50kHz,衰减特性陡峭,为78dB/oct.

典型参数:

衰减特性 78dB/oct
截止频率 50Hz—50kHz

截止频率误差 ±5%以下
截止频率控制 电压,10mV~10V
失真率 0.2%以下
噪声 1mV以下
工作电压范围 ±12V~±16V
消耗电流 ±55mA以下

典型应用接法如图20所示。引脚①为信号输入端,⑥脚为输出端,③~④脚加控制电压,实际参数如表10所示。

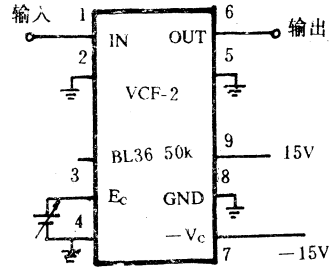


图20

表10

控制电压 (V)	fc (Hz)	失真率 (%)	噪声 (V)	失调电压 (mV)	消耗电流 (mA)
10	50k	0.15	1.1m	-3.3	+66.0 -66.5
1	5k	0.065	580μ	-4.9	+54.5 -55.1
0.1	500	0.036	320μ	-4.6	+53.5 -54.1

实测频率特性如图21所示。在最高截止频率50kHz时,参数达极限程度,但在其他频率时,特性很好。

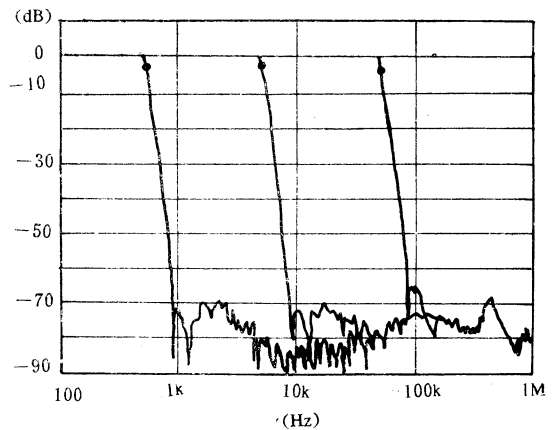


图21

高频变压器的设计与选择

戴惟萌 张占松

高频变压器是开关电源中的关键器件之一,其性能的好坏不仅影响变压器本身的效率、发热,而且还会影响到开关电源的技术性能和可靠性。在设计高频变压器时,必须周全地考虑对变压器性能产生影响的因素,如材料的选择、磁通密度和铁耗的限制,制造工艺等等。本文首先介绍高频变压器设计及选择的原則,再具体介绍变压器的设计方法。

一、高频变压器设计和选择原則

在设计、选择高频变压器时,必须满足下列要求:

1. 变压器原、副边线圈的变比应满足要求的数值,即当输入电压降到最低值时,仍能得到所要求的输出电压。

$$\text{设 } n = \frac{N_s}{N_p} \quad (1)$$

其中 N_s 为副边线圈的匝数, N_p 为原边线圈的匝数, n 为变压器的变比。

在正激电路、中心抽头、桥式电路中

$$n_{\min} = \frac{V_o + V_{DF}}{V_{SL}} \frac{1}{H_{\max}} \quad (2-1)$$

在半桥式电路中

$$n_{\min} = \frac{2(V_o + V_{DF})}{V_{SL}} \frac{1}{H_{\max}} \quad (2-2)$$

在反激型电路中

$$n_{\min} = \frac{(V_o + V_{DF}) t_{OFF}}{V_{SL} t_{ON}} \quad (2-3)$$

其中, V_o 为输出电压值(V), V_{DF} 为输出回路二极管的压降(V); V_{SL} 为输入电压的最小值(V), H_{\max} 为最大开通时间占空比; t_{ON} , t_{OFF} 分别为开关管的导通和关断时间(S); n_{\min} 为变压器应具有的最小变比。

为了满足这一要求,必须选择合适的变比 n , 使

$$n \geq n_{\min} \quad (3)$$

2. 当输入电压最高,占空比又最大时,磁芯不会饱和。

当 N 匝线圈上加电压 V_s 时,应满足

$$V_s = N_p \frac{d\phi}{dt} \quad (4)$$

若磁芯不饱和,对于一定的 V_s , $\frac{d\phi}{dt}$ 越大,则所需的匝数就越小,但若磁芯的工作点进入饱和区,则会产生波形传递失真,因此一定要选择合适的工作磁通密度最大值 B_{\max} 。要使磁芯不饱和,则必须

$$B_{\max} \leq B_{\text{sat}} \quad (5)$$

B_{sat} 为铁芯由线性区进入饱和区的临界磁感应强度(特斯拉)。

但当变压器的工作频率较高时,由于要限制铁耗,则可根据所确定的铁耗大小来作 B_{\max} 的选择。

一旦确定了 B_{\max} , 就可以决定原边线圈的匝数。

对于多开关电路,如图1(a)所示

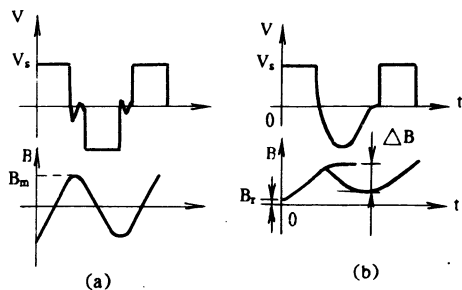


图1 开关电路中电压与磁感应强度的关系

$$\Delta\phi_m = z B_{\max} S = \frac{V_{s_{\max}} t_{\text{onmax}}}{N_p} \quad (6)$$

$$N_p' = \frac{V_{s_{\max}} t_{\text{onmax}}}{2 B_{\max} S} \quad (7)$$

对于单开关电路,如图1(b)所示,

$$\Delta\phi_m = (B_{\max} - B_r) S = \frac{V_{s_{\max}} t_{\text{onmax}}}{N_p} \quad (8)$$

$$N_p' = \frac{V_{s_{\max}} t_{\text{onmax}}}{S(B_{\max} - B_r)}$$

其中 S 为磁芯截面面积, B_r 为剩余磁感应强度; $V_{s_{\max}}$ 为最大输入电压, t_{onmax} 为最大导通时间; 为了满足设计原則2, 可选择

$$N_p \geq N_p' \quad (10)$$

3. 当输出功率最大时,变压器的温升在允许的范围內。

变压器的温升

$$\Delta\theta = P_{2RT} \quad (11)$$

其中 R_T 为变压器的势阻, P_{Σ} 为变压器的总损耗。在选择和设计变压器时,应保证

$$\Delta\theta \leq \Delta\theta_{\max} \quad (12)$$

$\Delta\theta_{\max}$ 为变压器所允许的最大温升。

4. 保证变压器的铜耗和铁耗相同,使变压器的效率最高,同时,使原边绕组的损耗与副边绕组的损耗相同。

变压器中主要存在下列两种损耗:

(1) 铁耗

当磁芯中的磁通量变化时,就会在磁芯中产生铁耗 P_{Fe} 。

$$P_{Fe} = K f^{\alpha} B^{\beta} \quad (13)$$

其中 K 为铁耗系数,它与磁芯的材料、结构及温度有关。一般 $\alpha = 1.2 \sim 1.6$, $\beta = 2.4$ 。

(2) 铜耗

设铜线的电阻率为 ρ , 磁芯可以绕线的窗口面积为 A_{cw} , 原边线圈的占有率为 K_0 , 每匝线圈的平均长度为 l , 则 N_p 匝原边线圈的直流电阻为

$$R_{CD} = \frac{\rho N_p l}{A_{cw} K_0} N_p \quad (14)$$

当线圈中通过高频开关电流, 由于集肤效应, 使电阻增大, 其增加系数为 K' , 则高频电阻为

$$R_{HF} = \frac{\rho N_p^2 l}{A_{cw} K_0} \cdot K' \quad (15)$$

当原边线圈电流的有效值为 $I_{p(rms)}$ 时, 原边线圈的功耗为

$$P_{LP} = I_{p(rms)}^2 R_{HF} \quad (16)$$

一般来说, 原边线圈的损耗应与副边线圈的损耗基本相同, 那么变压器的铜耗为

$$P_{Cu} = 2I_{p(rms)}^2 R_{HF} \quad (17)$$

变压器的总损耗为

$$P_{\Sigma} = P_{Cu} + P_{Fe} \quad (18)$$

由理论分析可知, 当变压器的可变损耗(即铜耗)和不变损耗(即铁耗)相同时, 其总损耗最低, 效率最高, 见图2。即当

$$P_{Fe} = P_{Cu} = \frac{1}{2} P_{\Sigma} \quad (19)$$

时, 变压器效率最高。

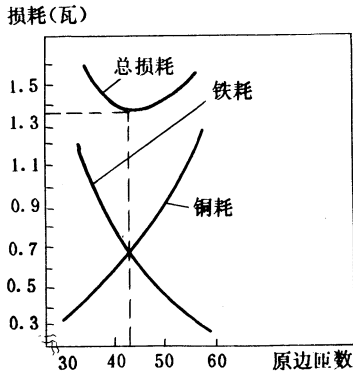


图2 铁氧体磁芯的损耗

5. 原、副边线圈之间的漏电感尽量小。

原、副边线圈之间的漏电感越小, 则传递波形的失真就越小。为了满足这一要求, 要注意工艺设计, 例如, 采用原边线圈和副边线圈分层交替绕制法, 使原、副边线圈的位置尽可能安排得均等一些。

二、高频变压器的设计与选择方法

从前面的分析可知, 在设计变压器时, 最大磁通密度值受两个因素的限制, 一是受磁芯材料饱和的限制, 二是受铁耗的限制。当以考虑磁芯材料的饱和限制为主时, 可按下列步骤设计。

1. 选择磁芯材料

高频变压器一般采用高频损耗小的铁氧体材料,

对其性能的要求是: 饱和磁通密度高, 磁滞损耗小, 温度系数小。

2. 决定最大磁通密度 B_{max}

这里以磁芯材料饱和的限制为主, 可按式(5)选择最大磁通密度 B_{max} 。

3. 决定磁芯的大小

在使磁芯不饱和及允许的铁耗条件下, 磁芯的体积, 必须能够存储最大的能量。变压器的体积主要与其传输功率有关。

若变压器属于对称工作方式, 即变压器的磁特性是在第一, 第三象限都可利用, 例如在推挽式、桥式电路中, 变压器的磁通密度是从 $-B_m$ 变化到 $+B_m$, 磁芯的大小可由下式决定:

$$A_P = A_{Fe} A_{cw} = \frac{P_T \times 10^6}{2\eta f B_{max} \delta K_\mu K_s} \quad (20)$$

其中 A_{Fe} 为磁芯的截面积 (mm^2), A_{cw} 为可绕线窗口面积 (mm^2), f 为工作频率 (Hz), B_{max} 为最大磁感应强度 (T), δ 为绕组的电流密度 (A/mm^2), η 为变压器的效率, K_μ 为窗口铜的填充系数, K_s 为磁芯填充系数, P_T 为变压器的标称功率 (伏安)。

但在单管式电路中, 变压器只利用了磁芯磁特性的一个象限, 即磁通的变化是从 B_r 变到 B_m (见图1(b)), 变压器不属于对称工作方式, 这类变压器的体积要选得大一些。

磁芯的体积选择是否合适, 还要在步骤7中进行校核。

4. 根据式(10)决定变压器原边线圈匝数 N_p 。

5. 根据式(1)决定变压器的变比和副边线圈匝数 N_s 。

6. 根据式(13)计算铁耗。

7. 设计线圈并校核磁芯的体积和变压器的温升。

根据已选择的电流密度, 选择导线的型号。常用多股线并绕以解决高频工作时导线有效利用面积的问题。令原、副边线圈导线的直径分别为 D_p 和 D_s , 导线的并绕根数分别为 r_p 和 r_s , 则每匝原边线圈、副边线圈对应的窗口面积分别为

$$S_{p1} = r_p D_p^2 \quad (21)$$

$$S_{s1} = r_s D_s^2 \quad (22)$$

首先校核所选择的磁芯的可绕线窗口面积是否合适, 可采用下列两式进行校核:

$$S_p = N_p S_{p1} \leq K_0 A_{cw} \quad (23)$$

$$S_s = N_s S_{s1} \leq (1 - K_0) A_{cw} \quad (24)$$

若 S_p, S_s 太大, 则需重新选择磁芯的大小。

其次要校核变压器的温升。由式(17)计算变压器的铜耗, 式(18)计算变压器的总损耗; 式(11)计算变压器的温升, 并检验是否满足式(12)。

按上述步骤所设计的变压器, 是以磁芯材料的饱和限制为条件的, 它使变压器具有较小的体积, 但并不一定具有最高的效率。若变压器的工作频率较高, 则变压器的铁耗必须是一个重要的考虑因素, 即在这种条件下, 更重要的是使变压器的效率最高, 那么可

按下列步骤设计:

1. 选择磁芯材料
2. 决定最大磁通密度 B_{max}
以所允许的铁耗来选择最大磁通密度。
首先根据变压器允许的温升决定变压器所允许具有的最大损耗。即

$$P_{\Sigma} = \frac{\Delta\theta_{max}}{R_T} \quad (25)$$

从前面的分析可知,要使变压器的效率最大,必须使变压器的铁耗与铜耗相等,即选择

$$P_{Fe} = P_{Cu} = \frac{1}{2} P_{\Sigma} \quad (26)$$

根据式(13)决定变压器的最大磁通密度 B_{max} 。

3. 决定磁芯的大小。
4. 根据式(10)决定变压器原边线圈匝数 N_p 。
5. 根据式(1)决定变压器的变比和副边线圈匝数 N_s 。
6. 设计线圈并校核磁芯的体积。

三、结束语

本文只是粗略地对高频变压器的设计方法作了介绍,如何根据设计原则,使用表格化方法进行设计,在《高频开关稳压电源》(张占松)中进行了简介。

高频变压器的设计是一个非常复杂的问题,它涉及电、磁等多方面的因素,设计性能最优的高频变压器仍是一个需进一步研究、探索的问题。

TCP/IP 网络简介

杨 军

TCP/IP 网是国际现有计算机网络的优秀者。60年代末,美国国防部高级研究计划局(ARPA)敏锐注意到计算机互连在今后社会发展中潜在的巨大作用,投资数家科研机构专门对计算机互连加以研究,随后开发出 TCP, IP 二个协议的 ARPA 国际网(Internet),目前,在北美及欧洲已有近千个网络是 ARPA 网际的成员,连接了成千台主机。十万以上用户,由于 TCP/IP 网早在国际互连标准制定之前就已广泛流行,它已成为计算机工业中事实标准,普遍出现在各个计算机生产厂家制造的产品中。

TCP/IP 实际上是一套协议,它连接许多不同的计算机和计算机网,完成它们之间信息传输的任务,TCP 是传输控制协议(Transmission Control Protocol),IP 是网际网协议(Internet Protocol)它具有如下几个特点:

(1)具有很强的互联性,不仅同种机型可以互联,还能支持异种机的互联,TCP/IP 协议是一种独立于生产厂家的网络协议,因此各种机器只要支持 TCP/IP 协议就可作为网上的一个节点访问网络其它节点。另外,虽然 TCP/IP 协议当初是与 UNIX 系统紧密相

关的,但是随着 TCP/IP 系统的不断发展和普遍使用,诸如 VMS, DOS 等其它操作系统上的 TCP/IP 协议已相继形成产品。因此,以 TCP/IP 为连接协议的计算机网络系统可以集小型机、工作站和 PC 机为一体,形成一个高功能的计算网络。

(2)支持客户/服务器(Client/Server)结构,能在整个网络环境内实现分布存取。TCP/IP 网络上的通信都是在一对 Client/Server 进程间进行,每个站点均有各种应用服务的相应进程,因此这种网络系统为分布式存取创造了良好的基础。

(3)TCP/IP 网络主要通过以太网总线连接各站点,各站点在网络中的配置通过修改系统中几个简单文件完成,网络一旦安装好后,系统管理员就无需再做更多的工作。因此,TCP/IP 网络是一非常易于管理的计算机网络。

通过上述分析可见,TCP/IP 网络系统很好地实现了优秀网络系统的几个目标,这正是该网络在世界范围内普遍流行的原因,也是其能够延伸到未来的信息社会的最主要的决定因素。

Expanded Memory 与 Extended Memory 译名管见

彭 禾

目前专业报章书刊中 Expanded Memory 及 Extended Memory 译名混淆。

从案头选杂志四种、书两本,列示有关译名(均省略存储器二字)如下:

	1	2	3	4	5	6
Expanded Memory	扩充	扩充	扩展	扩展	扩充	扩展
Extended Memory	扩展	扩展	扩充	扩充	扩展	扩充

- 注:1.《计算机世界月刊》1992年第12期第62页
 2.《新浪潮》1992年第2期第24页
 3.《中国计算机用户》1992年第12期第33页
 4.《计算机系统应用》1992年第3期第53页
 5.《如何使用 AST P386》任一民编写第99页 陕西电子编辑部
 6.《MS DOS 5.0》祝军等译第194页中科院沈阳计算所开发公司

由上列可见,它们的译名未统一,初学者每不知所从。附原文,似有帮助,惟繁琐。

查《现代高级英汉双解辞典》(1970), expand 与 extend 均示扩大,其间又指出 expand 为变大、膨胀,

Transputer 与 并行 处理

师 军

一、引言

近十几年来,并行处理技术的发展日新月异,新的并行硬件体系结构和新的产品不断问世。在如此众多的并行处理系统中,Transputer 体系结构独树一帜。它集通信与处理功能于一体,是一种将 RISC 技术与并行处理技术相结合的全新的计算机体系结构。可用于支持并行算法的研究、并行程序设计、进程之间以及多处理机之间的通信等等。

二、Transputer 结构

Transputer 结构如图1所示。可以看出,Transputer 在硬件结构的设计上包含了如下特征:

2. 局部静态 RAM 用于存放局部数据和程序,支持 Transputer 并行进程间的通信。

3. 扩充外存接口 通过该接口,CPU 可以直接访问达4GB 范围的外存空间。

4. 硬件时间片技术 用于支持多进程之间的并行操作以及相互切换。

5. 高速通信接口 用于 Transputer 之间或 Transputer 与其它外设之间的通信。

在并行处理系统中,总线式通信结构随着处理机的数目增多,将成为并行处理系统的瓶颈。处理机之间的通信结构的设计就显得尤为重要。Transputer 每片提供4路输入/输出双向通信链,支持多机间点对点通信,从而有效地克服了总线通信结构存在的缺陷。因为它不受总线频宽的限制,其通信能力随着 Transputer 数量的增加而加强。所以利用 Transputer 可以构造出各种拓扑结构的并行处理系统,为实现海量并行处理奠定了基础。

在 Transputer 上可以运行多种高级语言,如 Pascal、Fortran、Modula、Occam 等。但若充分要充分利用 Transputer 的结构特点,使系统高效运行,采用 Occam 语言最为合适。因为 Occam 语言不仅具备高级语言的一些优点,还具备了利用 Transputer 的特殊性能,并拥有很好的并行处理功能和同步通信的能力。INMOS 公司称,用 Occam 设计由 Transputer 组成的并行处理系统就好像用布尔代数设计由逻辑门组成的逻辑电路一样。

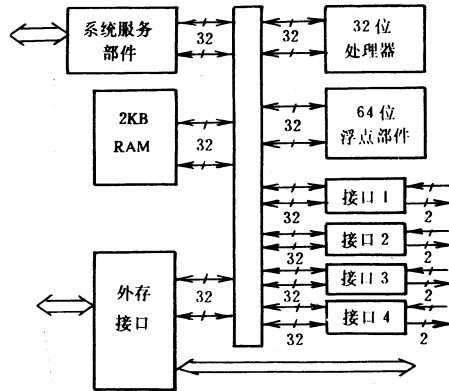


图1 Transputer 结构

三、进程间的通信

Occam 程序设计的基本单位是进程。两并行进程之间的通信采用通道进行。在 Transputer 中,两个相互通信的进程共享一个公共通道,由关键字 CHAN 说明。图2说明了两个并行进程之间的通信过程。由于进程之间的通信是通过点对点的方式进行的,而且通道采用单向、同步和无缓冲方式通信,因此并行执行的两个进程需要通信时,只有在输入进程和输出进程都已准备就绪的情况下,通信才能开始,否则一方则处于等待或挂起状态。多进程间数据通信可以采用不同的拓扑结构形式,如流水结构、阵列结构、树结构和立方体结构等等。其中流水结构是一种最为简单的多进程通信的形式,而且采用流水处理,可以缩短程序的执行时间。例如,若有 N 个并行进程,每个进程的运行时间为 $t(i)$,则进入流水结构后, N 段流水所需的时间为 $N * \text{MAX}[t(0), t(1), \dots, t(N-1)]$,而 N 次循环所需的时间为 $N * [t(0) + t(1) + \dots + t(N-1)]$ 。

Transputer 的通道为进程之间的通信提供了通路。单 Transputer 上并行进程之间的通信,其通道由内部存储器单元实现。若并行进行位于不同的 Transputer 上,则通道由点对点的链路实现。从程序设计者的角度看,不同 Transputer 上进程之间的通信与单

而 extend 含伸展、延长之意。

众所周知,Extended Memory 指 IMB 以上线性存储器,乃内存的延伸部分;Expanded Memory 为分页式存储区,窗口在紧接基本内存的上位存储块(Upper Memory Block),犹如‘中间开花’,含扩张之意。

据此,建议将 Extended Memory 译为扩展存储器;Expanded 译作扩页存储器。名称宜一目了然,且有助于对其机制的认识。用词汇丰富的汉语表达,它们都是扩充的存储器。

如何选用 MODEM 卡

——PC-PC 远程通信(一)

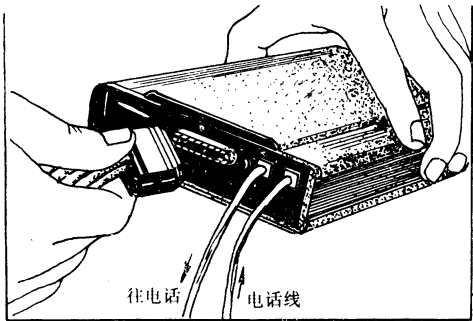
申老师:这次介绍 PC 之间通过电话线的远程通信。上次讲过,除了需要一台微机,还需要一根带有 RJ-11 插头的电话线,一块 MODEM 卡,即一块内插式调制解调器电路板。当然,通信的对方也需要这些器材。

小李:MODEM 是将数字信号转换为模拟信号,使能在电话线上进行传输的设备吧?

申老师:是的,它也能把从电话线接收到的模拟信号解调制,还原为数字信号,供计算机处理。

小李:上次讲,MODEM 有独立式的,也有插板式的,我们为什么选用插板式的?

申老师:独立式 MODEM 从外形看,是一个大小象一本厚书的扁盒子。独立式 MODEM 连接起来比较方便:用一根 RS-232 电缆将它与 PC 机的串行通信口(例如 COM1)连接,再将电话线接上(用 RJ-11 小插头),通上电就可以了,如图1所示。独立式 MODEM 的优点是通用性好,连接什么类型的机器都可以,PC 机之外还可以连 Apple II、MacIntosh 等。缺点是价格贵,每个1500~3000元,而且占地方。而内插式 MODEM 电路板,价格低,仅500~700元,而且插在 PC 机内,不占地方,因此容易普及,这就是我们选用 MODEM 卡的原因。



Transputer 上进程之间的通信没有什么区别。程序设计者只需指定存储器单元作为通道字,然后由硬件根据字的地址来决定并行进程之间的通信是由内部通道进行还是由外部通信链路进行。

四、结束语

Transputer 是计算机并行处理领域里的一项新的突破。它有效地克服了冯·诺依曼计算机体系结构无法克服的瓶颈问题,为研究并行计算机系统提供了新的思想。Transputer 和 Occam 相结合可以实现各种拓扑结构的并行处理系统,对并行处理领域的发展产生了深远的影响。

小李:选购 MODEM 卡时,要注意什么?

申老师:关键是选择一个在速度、价格、误码校正和其他性能都满足自己要求的 MODEM 卡。1. 速度方面。现在传输速率为每秒1200二进制位(即1200bps)的 MODEM 卡已相当普及,最好选2400bps 的。有时它不标出速度,而是说明支持什么通信标准。一般,标 Bell212A 或 CCITT V. 22的,是1200bps 的;标支持 V. 22bis 的,是2400bps 的。后者只比前者贵几十元。2. 误码校正方面。一般低档的 MODEM 卡并不支持高性能的误码校正。支持 V. 42或 MNP-4误码校正规程的 MODEM 卡,要比普通的 MODEM 卡价格高出50%以上,而且也常常包括了支持 V. 42bis 式 MNP-5数据压缩功能(可压缩至一半)。3. 尽量选择集成度高,工艺水平高的电路板,这样可靠性也高一些。4. 要有配套的用户手册,如果没有手册,可能导致用不起来。

小李:MODEM 卡买回来以后,如何安装和测试它的好坏呢?

申老师:先要了解自己机器内串行口的配置情况;根据串行口情况对 MODEM 卡进行硬件设置;然后用几个 BASIC 语句,或使用一种通信软件,就可以试验它的好坏。

老一点的 PC 机是不带串行口的;PC/XT 原装机带有一个串行口,设置为 COM1;新一点的286机,常使用“二串一并”I/O 接口板,则含二个串行口,且设置为 COM1和 COM2。使用某些工具软件(例如 PCTools)能很容易测出带不带串行口和带几个串行口(Serial port)。也可以打开机箱,看看接口卡上有没有 COM1、COM2或 ASYN1、ASYN2等字样,那就是串行口。

独立式 MODEM 一定要用到 PC 内的串行口。而 MODEM 卡不同,它本身带有串行接口电路,它并不需要机器带有串行口,反而要避开机器内原有的串行口,否则会发生冲突。

MODEM 卡上有 DIP 开关或跳线,供你选择口地址,可以在 COM1~COM4之间选择。如果 PC 内未带串行口,MODEM 卡可以选 COM1;PC 内带有一个串行口,已占用 COM1,则 MODEM 卡应选 COM2;如果 PC 机已占用 COM1和 COM2,则 MODEM 卡只能选用 COM3。但要注意,低版本的 DOS 和 BASIC 只支持使用 COM1和 COM2,而不支持使用 COM3和 COM4。

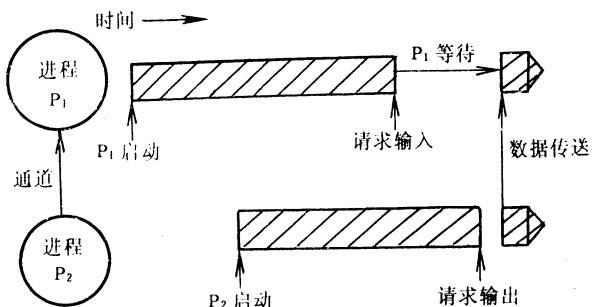


图2 两进程之间的通信过程

随着对电脑使用要求的提高,比如增加鼠标器等串行设备,串行口的逻辑资源也越来越显得宝贵。

小李:RJ-11小接头是几芯的?如何与电话线相接呢?

申老师:RJ-11小插头实际上是一个1立方厘米左右的积木式塑料小方块,上面有4个金属触点,这是美国制式的。对于2线电话,只用到中心的2个触点。用这个小插头往MODEM卡上插接十分方便。在购买MODEM卡时,一般都配有一条几米长的电话电缆,其中一端带有RJ-11插头。连接时,需将该电缆另一端的二条线接到电话线接线盒上。在我国南方不少地区,墙上的电话线接线盒,要用另一个小插头才能引出,而且小插头又与美制的正方形RJ-11不同,是一种长方形英国制式的,电话线要接到上面4个触点的最外边2个触点。这些连线问题看来是些小问题,但如果不解决,也无法往下做。

今天我们已经讲了不少,下次我们将以市场上较多的DM-2400H这种2400bps的MODEM卡为例,介绍板上的设置和拨号、应答、通信的BASIC程序。再下一次将介绍一种很普及的通信软件的使用。

传真机专题讲座

传真机传输控制规程

张建军 张景生

为了使各国生产的传真机在线路上能够互通,国际电报电话咨询委员会对传真机作了相应的规定,即T系列建议。其中T.30是关于文件传真在公用电话交换网上的传输规程。

按照CCITT T.0建议,将传真设备分为三类。因此,T.30建议中的传输控制规程分为两种,一种是用于一、二类传真设备的单音信号,另一种是用于三类传真机的二进制信号。我国国标GB3382-82中规定,我国的三类机不采用建议T.30中一、二类机的单音信号方式部分。

一、传真过程的时间顺序

在公用电话交换网上使用的传真设备,其传真过程包括以下几个方面:

呼叫的建议和释放

兼容能力的检测、状态和控制命令

线路状态的检测和监测

控制能力和传真员的再次呼叫

这些过程可以是人工建立,也可以是自动建立。

人工建立是指操作过程中的某一项需要操作人员介入。而自动建立则是传真机不需操作人员介入,能自动执行所有过程。由此可以产生以下几种操作方法:

1. 主叫站人工操作和被叫站人工操作,在这类方式下,可以是主叫站发给被叫站,也可以是主叫站收自被叫站。

2. 主叫站人工操作和被叫站自动操作,在这类方式下,也存在两种状态:主叫站发或主叫站收。

3. 主叫站自动操作和被叫站人工操作,在这类方式下,存在两种状态:主叫站发或主叫站收。

4. 主叫站自动操作和被叫站自动操作,在这类方式下,存在两种状态:主叫站发或主叫站收。

当然除了这种一个站对一个站的操作方式外,还可能有多点接收(即允许不止一个站接收报文)的操作方法。

二、三类传真机在传真过程中使用的单音信号

CCITT T.30建议对于符合建议T.3、T.4操作的设备使用的单音信号作了相应的规定,但我国GB3382-82中对三类机使用的单音信号只采用建议中规定的两种,即被叫站标识(CED),主叫单音(CNG)。

1. 被叫站标识(CED)

被叫站被接续到线路之后的1.8s到2.5s内,发送连续的 $2100 \pm 15\text{Hz}$ 单音,持续时间不短于2.6s,不长于4.0s。

被叫站在终止CED单音之后和发送其他信号之前,延迟 $75 \pm 20\text{ms}$ 。

其作用是表明被叫非语音终端。

2. 主叫单音(CNG)

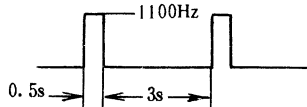
作用:

(1)表示主叫为非语音终端。本信号对自动呼叫设备来说是必备的,对人工设备是选用的。

(2)表示设备处于发送状态,并准备在收到相应的机类标识信号后发送。

(3)能发送多页文件且无需操作人员介入的设备,本信号可在文件之间发送,这时发送机在等待相应的机类标识信号。同时,向对方操作员表示发送机仍接于线路。

格式:



CNG信号发送0.5s、断3s,其时间误差为 $\pm 15\%$,频率误差为 $\pm 38\text{Hz}$

三、三类机使用的二进制信号

三类传真机传输控制规程采用二进制信号,在线路上以300bit/s的同步方式传送。所有的二进制传真控制过程都采用HDLC(高级数据链路控制)帧结构。基本的HDLC结构由多个帧组成,每帧分为若干字段,HDLC帧结构如图1所示。这种结构为帧提供标志、差错校验和正确收到信息的证实。

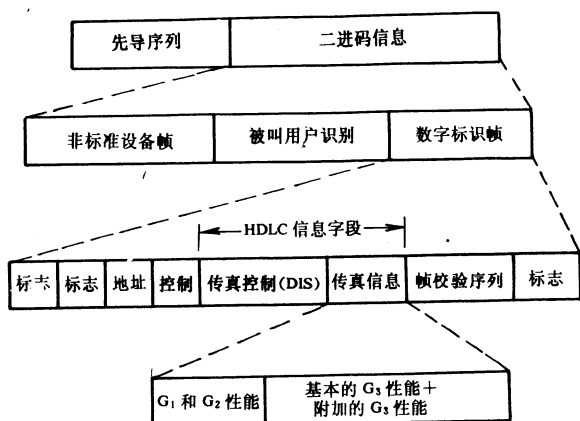


图1

(一)先导序列

任何一个方向的新信息传输开始(即每次线路上传输方向转换时),先导序列必须在所有二进制信号的前面。先导序列保证信道中各个部分(例如回声抑制器)都处于正确状态,以便使后随的数据不受损害地通过。

格式:300bit/s 二进制信号的先导序列是1s ± 15%的一串“标志”序列。

(二)标志序列

8bitHDLC 标志序列用来表示帧的开始和结束。对传真过程来说,标志序列是用来建立 bit 和帧的同步。如连续发送标志序列,可用来表示传真机仍接在电路上,但未作好立即进行传真过程的准备。

格式:0111 1110

(三)地址字段

8bitHDLC 地址字段是用来在多点连接中提供特定的一个站(或一些站)的标识。在公用电话交换网上传输的情况下,地址字段限制为一种单一格式。

格式:1111 1111

(四)控制字段

8bitHDLC 控制字段专门为传真控制过程提供各种命令和响应的编码能力。

格式:1100 × 000

对过程内的非末帧 × = 0,对末帧 × = 1。

(五)信息字段

HDLC 信息字段的长度是可变的,包括两个传真站之间控制和报文互换的专用信息。信息字段分为两个部分,传真控制字段(FCF)和传真信息字段(FIF)。

四、传真过程的控制规程

(一)单音和二进制过程间的互通

CCITT T. 30中规定了传输控制规程的两种方式,这两种方式间的互通,承认优先使用编码过程的原则,在可以使用时就先试用二进制信号。互通的步骤如下:

1. 无人值守的被叫站必须用 CED 信号应答呼叫。

2. 无人值守的主叫站必须用 CNG 信号表示呼叫。

3. 当具备二进制信号方式能力时,被叫站总是用二进制信号开始。

4. 只具备单音信号方式能力的传真站,才以单音信号开始。

5. 具备二进制和单音两种信号方式能力的传真站,信号的发送顺序是第一个为二进制信号,第二个和以后的所有信号都是单音和二进制信息的组合。

6. 若主叫站对二进制响应,则全部控制过程用二进制信号。

7. 若主叫站对单音响应,则全部过程用单音信号。

(二)信号顺序

三类传真机的传输控制规程发送和接收的过程如图2所示。

主叫	被叫
	CNG →
阶段 A	← CED
阶段 B	← DIS(NSF)(CSI) → DTC(CIG)(NSC) → DCS(TSI)(NSS) → TCF
阶段 C	← CFR 或 FTT → 报文传输、报文中同步 差错检测、纠正 线路监测
阶段 D	→ 报文结束或多页信号
阶段 E	→ 过程结束 ← 报文证实 → 呼叫释放

图2 三类机通信过程示意图

由图2可以看出:传真呼叫的建立,是以单音信号 CNG、CED 开始的,在被叫站发送 CED 信号后,才发送二进制信号。二进制信号以 HDLC 帧结构形式发出。在二进制信息中,包含有被叫端的全部性能,如传真机的类别、扫描线密度、信号速率等。当主叫端发送报文时,在收到被叫端的数字标识信号后,发出相应的数字命令信号 DCS,使收发双方处于同一工作状态。如果主叫端为接收状态时,则发出数字命令信号 DTC,也同样使收发双方处于同一工作状态。延时75 ± 20ms 后,切换到高速调制解调器,开始以初始速率发出 TCF。接收端根据检测到的结果,发出相应的响应信号。检测到的信号误码小,发出 CFR;如误码大则发出 FTT,重新训练。当主叫端收到 CFR 后,开始以 CCITT 建议 T. 4规定的高速调制解调器发送报文。如果是单页报文在报文结束后发送 EOP。接收机则根据报文接收情况,发出的证实信号。如接收报文中的误差在允许范围内,发出报文证实信号 MCF;误差超过允许范围,则不发信号。主叫端在收到 MCF 后,发出切断接续的使令,双方中止工作。如果是多页发送,发送方则发送 MPS,在收到接收方的 MCF 后,接着发送下一页报文。