



電子

744

與電腦



• ELECTRONICS AND COMPUTERS •

郑州磁带厂

厂长：彭作礼

地址：河南省郑州市经七路十五号

电话：334350 332509

电报：3651



郑州磁带厂是化学工业部定点的专业化磁带厂，使用国外引进的全套先进设备，选用优质原材料，采用国际标准生产和检验ZZZ牌盒式录音磁带。

主要产品：LN型、D型、ZD型盒式录音磁带；PC—D型高速复录大饼磁带；HC—O I 磁头清洗带；MD型录音机试机磁带。

ZD系列和PC—D型磁带获一九八八年河南省优质产品称号。

ZD系列和PC—D型磁带获一九八八年河南省优质产品一等奖和河南省科技进步奖。

上海牌音像磁带



● 上海磁带厂



- 中国第一盘广播录音磁带
- 中国第一盒盒式录音磁带
- 中国磁带质量评比第一名的生产企业
- 一九八七年广播录音磁带获国家银质奖。
- 享有盛誉的上海牌录音、录像磁带，是各类专业单位、家庭最理想的音像产品，该厂竭诚为社会各界服务，欢迎惠顾。

● 上海磁带厂竭诚为您服务向您

● 提供优质上海牌录音录像磁带

● 厂址：仙霞路84号 电话：598661

● 国内代号：2—888 定价：0.95元

6502 指令表

表① D7D6D5D4D3D200 型指令

D4D3D2 D7D6D5	000	001	010	011	100	101	110	111
000	BRK		PHP		BPL		CLC	
001	JSR	BIT \$ nn	PLP	BIT \$ nnnn	BMI		SEC	
010	RTI		PHA	JMP \$ nnnn	BVC		CLI	
011	RTS		PLA	JMP (\$ nnnn)	BVS		SEI	
100		STY \$ nn	DEY	STY \$ nnnn	BCC	STY \$ nn, X	TYA	保留
101	LDY # \$ nn	LDY \$ nn	TAY	LDY \$ nnnn	BCS	LDY \$ nn, X	CLV	LDY \$ nnnn, X
110	CPY # \$ nn	CPY \$ nn	INY	CPY \$ nnnn	BNE		CLD	
111	CPX # \$ nn	CPX \$ nn	INX	CPX \$ nnnn	BEQ		SED	

表② D7D6D5D4D3D201 型指令

D4D3D2 D7D6D5	000	001	010	011	100	101	110	111
000	ORA (\$ nn, X)	ORA \$ nn	ORA # \$ nn	ORA \$ nnnn	ORA (\$ nn), Y	ORA \$ nn, X	ORA \$ nnnn, Y	ORA \$ nnnn, X
001	AND (\$ nn, X)	AND \$ nn	AND # \$ nn	AND \$ nnnn	AND (\$ nn), Y	AND \$ nn, X	AND \$ nnnn, Y	AND \$ nnnn, X
010	EOR (\$ nn, X)	EOR \$ nn	EOR # \$ nn	EOR \$ nnnn	EOR (\$ nn), Y	EOR \$ nn, X	EOR \$ nnnn, Y	EOR \$ nnnn, X
011	ADC (\$ nn, X)	ADC \$ nn	ADC # \$ nn	ADC \$ nnnn	ADC (\$ nn), Y	ADC \$ nn, X	ADC \$ nnnn, Y	ADC \$ nnnn, X
100	STA (\$ nn, X)	STA \$ nn		STA \$ nnnn	STA (\$ nn), Y	STA \$ nn, X	STA \$ nnnn, Y	STA \$ nnnn, X
101	LDA (\$ nn, X)	LDA \$ nn	LDA # \$ nn	LDA \$ nnnn	LDA (\$ nn), Y	LDA \$ nn, X	LDA \$ nnnn, Y	LDA \$ nnnn, X
110	CMP (\$ nn, X)	CMP \$ nn	CMP # \$ nn	CMP \$ nnnn	CMP (\$ nn), Y	CMP \$ nn, X	CMP \$ nnnn, Y	CMP \$ nnnn, X
111	SBC (\$ nn, X)	SBC \$ nn	SBC # \$ nn	SBC \$ nnnn	SBC (\$ nn), Y	SBC \$ nn, X	SBC \$ nnnn, Y	SBC \$ nnnn, X

表③ D7D6D5D4D3D210 型指令

D4D3D2 D7D6D5	000	001	010	011	100	101	110	111
000		ASL \$ nn	ASL	ASL \$ nnnn		ASL \$ nn, X		ASL \$ nnnn, X
001		ROL \$ nn	ROL	ROL \$ nnnn		ROL \$ nn, X		ROL \$ nnnn, X
010		LSR \$ nn	LSR	LSR \$ nnnn		LSR \$ nn, X		LSR \$ nnnn, X
011		ROR \$ nn	ROR	ROR \$ nnnn		ROR \$ nn, X		ROR \$ nnnn, X
100		STX \$ nn	TXA	STX \$ nnnn		STX \$ nn, Y	TXS	保留
101	LDX # \$ nn	LDX \$ nn	TAX	LDX \$ nnnn		LDX \$ nn, Y	TSX	LDX \$ nnnn, Y
110		DEC \$ nn	DEX	DEC \$ nnnn		DEC \$ nn, X		DEC \$ nnnn, X
111		INC \$ nn	NOP	INC \$ nnnn		INC \$ nn, X		INC \$ nnnn, X

表④ D7D6D5D4D3D211 型指令

D4D3D2 D7D6D5	000	001	010	011	100	101	110	111
000	ASL(\$nn,X) ORA(\$nn,X)	ASL\$nn ORA\$nn	AND# \$nn	ASL\$nnnn ORA\$nnnn	ASL(\$nn),Y ORA(\$nn),Y	ASL\$nn,X ORA\$nn,X	ASL\$nnnn,Y ORA\$nnnn,Y	ASL\$nnnn,X ORA\$nnnn,X
001	ROL(\$nn,X) AND(\$nn,X)	ROL\$nn AND\$nn	AND# \$nn	ROL\$nnnn AND\$nnnn	ROL(\$nn),Y AND(\$nn),Y	ROL\$nn,X AND\$nn,X	ROL\$nnnn,Y AND\$nnnn,Y	ROL\$nnnn,X AND\$nnnn,X
010	LSR(\$nn,X) EOR(\$nn,X)	LSR\$nn EOR\$nn	AND# \$nn LSR	LSR\$nnnn EOR\$nnnn	LSR(\$nn),Y EOR(\$nn),Y	LSR\$nn,X EOR\$nn,X	LSR\$nnnn,Y EOR\$nnnn,Y	LSR\$nnnn,X EOR\$nnnn,X
011	ROR(\$nn,X) ADC(\$nn,X)	ROR\$nn ADC\$nn	AND# \$nn ROR	ROR\$nnnn ADC\$nnnn	ROR(\$nn),Y ADC(\$nn),Y	ROR\$nn,X ADC\$nn,X	ROR\$nnnn,Y ADC\$nnnn,Y	ROR\$nnnn,X ADC\$nnnn,X
100	A \wedge X \rightarrow (\$nn,X)	A \wedge X \rightarrow \$nn	TXA AND# \$nn	A \wedge X \rightarrow \$nnnn	保留	A \wedge X \rightarrow (\$nn),Y	保留	保留
101	LDA(\$nn,X) TAX	LDA\$nn TAX	LDA# \$nn TAX	LDA\$nnnn TAX	LDA(\$nn),Y TAX	LDA\$nn,X TAX	保留	LDA\$nnnn,Y TAX
110	DEC(\$nn,X) CMP(\$nn),X	DEC\$nn CMP\$nn	A \wedge X \rightarrow X	DEC\$nnnn CMP\$nnnn	DEC(\$nn),Y CMP(\$nn),Y	DEC\$nn,X CMP\$nn,X	DEC\$nnnn,Y CMP\$nnnn,Y	DEC\$nnnn,X CMP\$nnnn,X
111	INC(\$nn,X) SBC(\$nn,X)	INC\$nn SBC\$nn	SBC# \$nn	INC\$nnnn SBC\$nnnn	INC(\$nn),Y SBC(\$nn),Y	INC\$nn,X SBC\$nn,X	INC\$nnnn,Y SBC\$nnnn,Y	INC\$nnnn,X SBC\$nnnn,X

《电子与电脑》广告刊例

为了便于国内外厂家、公司以及有开发能力的院校、研究单位等能通过本刊向广大读者及社会各界宣传及销售贵公司有关电子元器件、各种微电脑扩展卡、开发装置、电脑硬软件及电子新产品。本刊特开辟一、二类广告栏目，欢迎各位惠顾。

根据《广告管理暂行条例》规定，申请刊登广告要开具生产单位的介绍信及产品鉴定书，优质产品要有获奖证书的复印件，刊登商标要有注册证复印件，其他广告要有营业执照的复印件或有关证明。

广告来稿要抄写清楚，内容实事求是。申请广告要注明结算单位，银行帐号，广告刊出后当月结帐。封底广告由我刊设计，封二和封三可由刊户自行设计，无设计能力者可委托我刊设计，文字广告由刊户提供文稿。封底的彩色广告和封二封三的黑白广告由刊户提供彩色反转片及黑白照片，无条件提供者本刊代为拍摄，按广告费的10%加收工本费。欲登广告者请先填写我刊的认刊书寄交我部。待我刊认定后再签合同。

欲登某期广告应提前两个月交稿。

国内广告价格表（1991年度）

广告位置	彩色封底全页	封二、三 (黑白)	内页文字
价格(元)	3000元	全页1500元 半页800元	全页1000元 半页500元 1/4页300元

地址：北京万寿路27号院四号楼《电子与电脑》编辑部 开户银行：北京工商银行翠微路分理处

电话：8212233 — 3464

户名：电子工业出版社

邮政编码：100036

联系人：郝承

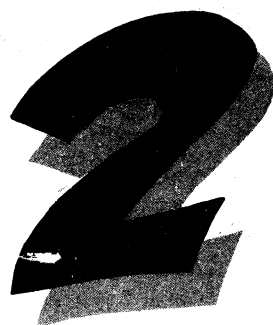
帐号：891238 — 72



電子
與
電腦



* ELECTRONICS AND COMPUTERS *



一九九一年

总期第71期

電子與電腦

目

录

· 综述 ·

PAL 和 GAL 朱世鸿(2)

· PC 用户 ·

通用打印机状态检测程序 夏伟文(3)

dBASE III 与 BASIC 之间的数据传送问题

..... 严桂兰(5)

dBASE 运用经验与技巧 孙 鹏(7)

如何解决在运行 FOXBASE

内存不够的问题 周日初(9)

COMPRESS 在硬盘集中存储中的应用 许 鹰(9)

怎样在 IBM-PC 机上实现汉字的动态显示

..... 李晓华(10)

DOS I/O 重定向、管道功能及其使用 徐国辉(11)

在低版本 DOS 下安装虚拟盘 周健勇(12)

五笔 CCDOS 中一字节修正 吴邦忠(12)

一个快速实用的字符串替换程序 金林樵(12)

PC 机磁盘文件的加密与解密 刘忠达(13)

硬盘正确格式化一例 袁士彬(14)

· 学习机之友 ·

程序出错处理与 RESUME 的功能扩充 丁志伟(15)

特殊 6502 指令 李 铁(17)

四舍六入法程序 韦肖鸿(17)

“怪”指令在软件加密中的应用 傅 剑(18)

双显示电子钟 万 斌(19)

苹果机汉字造型表自动生成程序 刘士明(20)

高效率的排序方法 陈庆祥(21)

演示抛物面的光学效应 王晓林(23)

LASER QZT/C 应用经验 张海翔(24)

· C 语言初阶 ·

第二讲 Turbo C 基本数据 李文兵(25)

· 学用单片机 ·

普及型单片机开发工具的硬件设计

..... 张培仁 刘振安(29)

· 学装微电脑 ·

40 支开关输入部件的制作 易齐干(34)

· 电脑巧开发 ·

MP-I 汉字及图形显示装置 陈名则(37)

APPLE-II 彩灯自控电路 张孝玖(38)

· 维修经验谈 ·

王安机 RAM 故障简易检修法 董纯坚(40)

IBM 286 维修一例 叶志斌(41)

中华学习机典型故障一例的分析 杨瑞华(42)

APPLE II 主机常见故障的检修 陈 鸿(42)

· 初级程序员级水平考试辅导问答 ·

计算机基础知识(下) 王路敬(43)

· 新书与软件 ·

Turbo CAD 软件简介 (46)

· 游戏通 ·

巧用电脑作幻方 郭高惠(47)

· 读者联谊 ·

湖北地区中华学习机联谊活动情况 (48)

· 信息与服务 ·

..... (48)

· 封二、封三 ·

6502 指令表 李 铁

机械电子工业部电子工业出版社主办

编辑、出版：《电子与电脑》编辑部

（北京 173 信箱 邮政编码：100036）

印刷：北京三二〇九厂

国内总发行：北京市邮政局

国内统一刊号：CN11-2199

邮发代号：2-888

国外代号：M924

出版日期：每月 23 日

主编：王惠民 副主编：王昌铭

责任编辑：张 丽

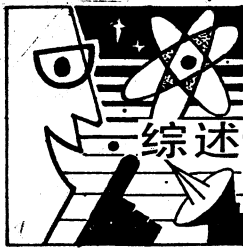
订购处：全国各地邮电局

国外总发行：中国国际图书贸易总公司

（北京 399 信箱 邮政编码 100044）

广告经营许可证：京海工商广字 147 号

定价：0.95 元



PAL 和 GAL

朱世鸿

微电子技术和计算机技术是一对孪生姐妹。自1959年美国德克萨斯公司和仙童公司研制出在半导体硅片上制造出有特定功能的电子线路,即目前统称的集成电路(IC—Integrated Circuit)之后,微电子技术的高速发展超出了人们的预料,使现在的计算机科学技术每隔5~10年就有一次重大突破,机器的工作速度提高了10倍,而体积和成本却缩小10倍。目前,相当一台早期庞大计算机功能的电子线路,可全部集成在面积仅为几个 mm^2 的硅片上(见本刊91年第1期封面图片)。

构成计算机的IC从59年至今,经历了一个从采用小规模IC(SSI)→中规模IC(MSI)→大规模IC(LSI)→甚大规模IC(VLSI)→超大规模IC(SLSI)的过程。随着半导体工艺VLSI和SLSI技术的不断成熟,自然就可将各种越来越复杂的电子线路集成到硅片上,从而出现了各种专用IC即(ASIC—Application Specific IC)。

关于ASIC目前尚无统一严格的定义,通常,有全定制和半定制二种之分。

全定制电路是厂家根据用户的具体要求而单独设计的某种特定功能的电路。一般凡大量生产的产品,例如通用的微处理器Z80,Z8000,8086,MCS51等均采用此方法设计。

半定制电路是在芯片上设计制作好一些元件和元件组,用户仅只考虑电路逻辑功能和各功能模块之间的连接就可以了,因此,人们对半定制的ASIC产生了极大的兴趣。

半定制ASIC按其内部结构可分为门阵列(Gate Array)、标准单元(Standard Cell)和可编程逻辑器件(PLD—Programmable Logic Device)。

(1)门阵列是在硅片上制成标准逻辑门的矩阵,是不封装的半成品,生产厂家根据用户设计要求选择合适的阵列规模,在阵列中制作出互连的图案(码点)进行定制,最后封装为成品售给用户。

(2)标准单元由一个预先定义好的功能单元库组成,它包括微处理器和存储器阵列,甚至包括线性电路,功放电路等。用户可从标准单元库中选取合适的逻辑块,对逻辑电路具有可优化的充分条件,因而设计自由度高。

上述两种ASIC电路均要由用户向生产厂家提出逻辑功能进行定做,用户无法自行开发。

(3)PLD又分可编程只读存储器(PROM、EPROM、E²PROM)、可编程逻辑阵列(PLA—Programmable Logic Array)以及可编程逻辑阵列(PAL—Programmable Array Logic)和可编程通用阵列逻辑(GAL—Generic Array Logic)。

有关PROM、EPROM和E²PROM的知识和用法,

大家都比较熟悉。它们在计算机中一般多用来存储监控程序和表格常数。而PLA器件其基本结构与可编程只读存储器相似,在功能上又有区别,PLA器件可通过编程方法实现其一定的逻辑功能,可以单独使用,因而在实现各种逻辑功能上具有灵活性和较强的功能,同其它可编程器件一样,若阵列的连线采用掩膜方式制成,称工厂编程的PLA,若阵列连线由用户烧制(可熔断丝)而成称现场可编程逻辑阵列FPLA。

随着电子技术,计算机技术对IC的要求,又开发出集中了具PLA编程和PROM的低成本等于一身的PAL器件。它以可编程的灵活性、可加密、传输速度快、降低器件总功耗和减少外围芯片数、减小电路板面积和加速系统设计、提高可靠性等众多优点而迅速地打入市场。

目前,市场上有29种型号的PAL,可替代90%以上约400余种标准的TTL IC的逻辑功能,通过适当的编程可使一片PAL等效地替代多片TTL IC的逻辑功能,留心观察计算机主机板的读者一定会发现,目前新型的各种PC机的主机板上的IC数目,远远少于几年前生产的PC机。其原因就是采用了PAL器件替代了原来的TTL IC的缘故。

在技术发展的过程中,往往是由于某种产品的不完善,而继续推出更高性能的产品。关于PAL器件,虽然它有着上述许多优点,但它是一次编程定终身,不能重复改变功能使用。于是,就应运而生了GAL器件。GAL是美国品格(LATTICE)半导体公司1985年研制的最新一代PLD器件,它可在一秒钟内完成对芯片的电擦/写(无须用紫外线擦除),具有加密功能,可重复并随时可修改其逻辑功能。所以,GAL器件已占领了PLD市场。国内对GAL器件的应用已引起广大开发与应用工作者的广泛而浓厚兴趣,一个普及应用GAL的“热潮”一定会在国内出现。

当然要用好GAL,还要掌握有关GAL器件的特性和设计方法,由于篇幅所限,《电子与电脑》将会不定期的对GAL及其编程器等知识进行介绍。本文仅给出GAL的概念,为了便于读者认识GAL,下面给出几种在国内较常用的GAL型号:GAL16V8, GAL20V8, GAL16V8A, GAL20V8A, ISPGAL16Z8, GAL39V8等。打算使用GAL器件或者有条件逛电子商行的读者,不妨先去找找看,先增加些感性认识。

本期封面我们选用了GAL器件的雕塑模型照片,希望您能在第一次认识GAL的时候,要记住GAL是可以由您任意塑造(编程或擦除)其逻辑功能的一种通用IC。

亲爱的读者,您要想开发新型的电子或计算机产品吗?您一定要逐步学会用GAL。



通用打印机状态检测程序

夏伟文

在 PC 机的 DOS 环境下,如果打印机未准备好而程序要求打印时,屏幕上会出现一个不优雅的提示“Abort, Retry, Ignore?”在这种情况下,有的程序不得不被迫中止运行乃至死机。因此,在打印以前通常要检测打印机的状态。在 FOXbase+ 系统中,可以用 SYS(13) 函数来确认打印机是否正常;在 BASIC 中可以用 ON ERROR GOTO 与 IF ERR = 24 OR ERR = 27 THEN 语句配合来检测打印机是否正常;借助 8088 汇编语言调用 BIOS 中断的 INT 17H 的功能之一就是读出打印机的状态。下面用三段程序来说明这三种用法。

FOXbase+:

```
DO WHIL .T.
  IF SYS(13) = 'READY'
    EXIT
  ENDI
  CLEAR
  ?? CHR(7)+CHR(7)
  SET COLO TO 6
  @10,25 SAY '打印机没准备好,请检查!'
  @12,27 SAY '按任意键再试!'
  SET COLO TO 7
  SET CONS OFF
  WAIT
  SET CONS ON
  ENDD
```

BASIC:

```
10 ON ERROR GOTO 1000 '出现错误则转到第
1000 语句处理
.....
1000 REM 打印机出错则转 5000 执行
1010 IF ERR = 24 OR ERR = 27 THEN GOSUB
5000,RESUME
5000 PRINT "打印机未准备好!":
PRINT "按任意键再试一次....." '通知
用户打印机出错
5010 IF INKEY$="" THEN 5010 '等待键盘输
入一个字符
5020 RETURN '返回主程序继续打印
```

在汇编语言中调用 BIOS 中断 17H,

```
MOV AH,02 ;请求读端口
MOV DX,00 ;选定默认的打印机端口
INT 17H ;调用 BIOS 中断
```

执行完这段汇编语言的操作以后,寄存器 AH 中的某些位设置成 1。该寄存器一共 7 位,各位含义如下:7: 打印机准备好;6:对最后一个字符的确认;5:无纸;4: 选择打印机;3:I/O 错;0:超时错。

以上三种方法,可以对打印机是否准备好作出简单的判断,但只能对打印机的错误给出笼统的提示。还有一种方法,可以区分出打印机错误的类型:是未联机、未通电、还是缺纸。下面先简述其原理:我们知道, DOS 能够处理三个并行设备(LPT1~3)。每个并行设备通过相应的适配器与主机相联。对适配器的操作则是通过对其中的三个 I/O 寄存器的读写实现的,每个 I/O 寄存器都有自己的口地址,在 BIOS 数据区中分配二个字节存放适配器的基地址,所谓基地址即三组口地址中每一组的最低地址。默认的打印机 LPT1 的基地址存放在 0040:0080。CPU 通过状态寄存器和控制寄存器来进行打印操作。状态寄存器有 7 位:

(0~2 位未用)

第 3 位:0=打印机出错 1=打印机正常
(未通电)

第 4 位:0=打印机脱机 1=打印机联机

第 5 位:0=打印机有纸 1=打印机无纸

第 6 位:0=打印机接到字 1=正常

符的回答信号

第 7 位:0=打印机工作 1=打印机不工作

要判断打印机是什么错误,只要读状态寄存器的第 3、4、5 位数值即可。例如,当寄存器的 7 位二进制位组合如下时:11011111(十进制数值为 223),表示打印机就绪。但不同的打印机尽管处于相同状态,状态寄存器位组合却不同。因此,可以用下面的程序 1 判定用户的打印机的状态参数,也就是分别在正常、断电、缺纸和未联机 etc 情况下运行该程序,即可在屏幕上显示出不同情况下状态寄存器的 7 位二进制位组合的数值(十进制)。然后,将得到的参数依照给定的次序写进第二个程序的 1000 DATA 语句中,注意次序不要颠倒,就可以使用程序 2 对打印机状态随时测试。

程序 1:

```
10 REM 测定参数
20 DEF SEG=&H40 '指向 BIOS 数据区
30 B = PEEK(9) * 256 + '求状态寄存器的地址
PEEK(8)+1
40 A = INP(B) '求状态参数
50 PRINT A '显示参数
```

程序 2:(以 TH 3070 打印机为例)

```
5 REM 程序名:CHPR.BAS
10 SCREEN 0,0,0;WIDTH 80;CLS;KEY OFF
20 READ A1,A2,A3,A4 '读入参数
30 DEF SEG=&H40 '指向 BIOS 数据区
40 B = PEEK(9) * 256 + '求状态寄存器的地址
PEEK(8)+1
```


dBASE III 与 BASICA 之间的数据传送问题

福建泉州华侨大学 严桂兰

在大量的应用软件中,人们已广泛地使用各种高级语言进行混合设计。这既可以充分调用各语言的优点,相互补充,相互完善;也可达到优化设计,提高软件质量的目的。本文仅就 dBASE III 与高级 BASIC 之间的数据传送的隐含问题作些技术性的阐述。

一、dBASE III 多字段数据被 BASICA 完整地接受。

在 dBASE III 系统中,每条数据记录可由高达 128 项字段组成,其字节数可高达 4000。这对大、中型企业的事务管理项基本上是够用的。当将多个字段数据传送给 BASICA 时,却遇上了麻烦。首先是 BASICA 的语句行中限制字符数不超过 255,而来自 dBASE III 多字段数据往往在 BASICA 接受数据的书写行中大大超过 255 个字符数。虽然在编辑输入 BASICA 程序行时,可无止境地键入多字段的数据格式,屏幕并无出错提示,可实际能接受的字符数仍是 255 个,超过部分都将被截除,引起数据丢失。为保证数据的完整性,可采取将原 dBASE III 的数据库文件生成二个(或者更多个)数据文件的方案。第一个数据文件可保留库中全部数据;第二个数据文件则截取库文件中后半部分数据。如此在 BASICA 中便可以两个数据文件来接受数据。原来由一个语句行接受不了的数据,现改为由两个语句行来接受,取第一个数据文件的前半部分数据(书写字符数小于 255),剩余部分数据由第二个数据文件提供。为节省空间,第一个数据文件也可取对应的前半部分数据,但要从 BASICA 或者从两个数据文件返回库文件形成一个完整的记录数据时,又会引起麻烦,因而,采用稍多的内存开销来保持数据的完整性及数据双向传送是值得的。

例如,在 zdq. dbf 数据库中有 39 个字段,其字节长度为 211,若用一个数据文件传送数据给 BASICA,其 LEN=211+2=213;

```

.....
140 OPEN "ZDQ.TXT" AS #1 LEN=213
150 FIELD #1,3 AS A1$,12 AS A2$,8 AS A3$,5 AS
A4$,16 AS A5$,16 AS A6$,6 AS A7$,4 AS A8$,8 AS
A9$,16 AS A10$,4 AS A11$ ..... ,4 AS A37$,4
AS A38$,5 AS A39$

```

在键入 150 行号内容时,形式上能按上述进行,无出错信息,但实际能被接受是:

```

"FIELD #1,3 AS A1$ .....
4 AS A14$, ..... ,4 AS A24$,4 AS A25$,4"

```

(共 255 个字符)

从 A26\$ 项到 A39\$ 项数据没能被接受。在 BASICA 中又无继续行的语法,因此,BASICA 本身是无法解决这一矛盾的。如果将 zdq. dbf 生成 zdq1. txt 及 zdqql. txt 文件:

Use zdq

Copy to zdq1 sdf

Copy to zdqql files a26,a27,...a39 sdf

则在 BASICA 中采用:

```

.....
140 z1="zdq1.txt";z2="zdqql.txt"
145 OPEN Z1$ AS #1 LEN=213;OPEN Z2$ AS #2
LEN=59
150 FIELD #1,3 AS A1$,12 AS A2$ ..... ,4 AS A25
$
155 FIELD #2,4 AS A26$,4 AS A27$ .....4 AS A38
$,5 AS A39$

```

注意:用两个数据文件传送数据时,每个打开文件的语句中的 LEN 值要比各数据文件数据项字节数之和加 2。如此处理后,BASICA 程序就能完整地接受来自 dBASE III 数据库中多个字段数据。

二、同一结构,不同数据的产生与传送

在实际使用中,往往随着时间的推移或者不同的对象,其数据完全不同,但所对应的数据项名、结构却完全相同,例如,银行存款单,其存款种类、帐号、户名,存款金额,期限、利率...等,不同的户名,就有一组对应数据;又如,外销企业,其外商的订单:订单号、订货日期、货号、订货数量、单价、金额、交货日期...也因不同的外商,其对应数据发生变化;在海关中的报关单,税务局的税款单等等,都属于结构相同,数据不同的类型。对于此类数据若采用 dBASE 进行管理,其输入、查询、检索是十分方便的,而且,还可利用同一数据库结构,对不同对象的数据进行各种操作,然后,转成数据文件保存各对象的不同数据,数据库便可腾出进行另一对象数据组的操作。如此往返使用同一数据库对数据进行管理,而后转变产生各自的数据文件的办法,可节省空间,程序结构也十分清晰,同时也为与其它语言接口作出必要的准备,实属一举多得。由于不同对象仅是数据不同,此时可按顺序建立同名(与数据库同名更易操作)仅以最后序号不同来区别,例如,新的一张订单数据,可根据已存在的最后的一个数据文件序号加 1,而产生新的数据文件名:

```

use zdq                ;打开订单库
zap                    ;删除上一个订单数据
:                      ;输入新的订单数据
:
n=1
nn=str(n,1)
do while file("zdq'+nn +'.txt'")
n=n+1
if n<10

```

```

nn=str(n,1)
else
nn=str(n,2)
endif          ;以序号在 100 之内为例
enddo
z="zq"+nn
copy to &z sdf          ;建立新订单的数据文件

```

若已有两个订单数据文件 zdq1.txt 和 zdq2.txt, 则按上面程序段再输入一个新的订单数据, 便产生 zdq3.txt 文件, 如此类推。

现在若要将同一结构, 不同对象的数据传送到 BASICA 进行数据处理, 即将遇到的第二问题是: BASICA 中到底要设置多少行对应的接受语句? (每一个数据文件都必须采用两个语句行——一个语句行用来打开该数据文件及参数设置, 另一行则要取其各数据项对应数据。) 由于订单数量是随机的, 无法确定或限死订单数量, 即数据文件的个数, BASICA 程序也就无法编写相应的接受语句。然而, 通过上述由 dBASE III 产生同一结构不同数据文件方法的启示, 也可在 BASICA 中采用相应的公式化接受语句:

```

.....
125 INPUT "输入订单号: "; N
130 IF N >= 10 THEN NN$ = RIGHT$(STR(N), 2)
135 NN$ = RIGHT$(STR(N), 1)
135 Z1$ = "zq" + NN$ + ".txt";
      Z2$ = "zqq" + NN$ + ".txt"
145 OPEN Z1$ AS #1 LEN = 213; OPEN Z2$ AS #2
LEN = 59

```

(以下 150—155 行同上)

上述七个语句行 (一般五个) 便可接受来自 dBASE III 数据库传送来的任意多个数据文件的任意字段数据, 因此, 用此公式化的接受语句便可解决同一结构多个数据文件的数据传送问题。

三、数据传送中的软件接口

各高级语言间传送数据的软件接口, 一般采用数据文件。BASICA 接受来自 dBASE III 的数据, 有两种软件接口形式, 一种是由 dBASE 库文件产生数据文件, 另一种是数据库文件本身。前者传送的数据较为简单、直观, 但要多开销内存, 它除数据库文件外, 还有对应的数据文件存在; 后者传送的数据直接来自数据库文件, 但取数据的方式较前者麻烦, 需经过计算才能得到, 对于中文数据还面临着数据保真性问题。

上面描述了由数据库产生数据文件, 而后被 BASICA 取其数据的全过程。但这是数据传送的特例, 因为它并不是每一个数据文件都对应一个数据库, 而是多个数据文件共用一个数据库结构, 较为经济, 数据文件是无结构的, 因此, 多个数据文件加一个数据库所占内存必然比对应多个数据库文件所占内存要少。对于不同结构不同数据的传送, 若也采取数据文件方式, 则多个数据库必然对应多个数据文件, 这势必

要开销更多的内存, 为此, 若直接利用数据库传送数据较为合适, 它可省去对应多个数据文件。例如:

```

240 OPEN "ao.dbf" AS #3 LEN = 66

```

由于数据库带有结构, 要读出库中的数据, 需跨越文件标识信息及结构, 因此, 从什么字段开始取数据, 取多长字符, 需根据下列公式计算而得:

$$N = 32 + 32 \times n + 2$$

其中 n 为库中的字段数

$$N / \text{LEN} = \text{整数} \cdots \text{余数}$$

LEN 为各字段长之和再加 1, 此 1 的字段号设为 0 字段, 专为安排记录作删除记号 "*" 而设的字符长度。

依照余数大小, 从 0 字段开始, 顺序将各字段长累加, 直到某个字段时, 其字段长与前累加之和大于或等于余数为止, 然后, 对此特定的字段长分成两部, 前半部分与前累加字段长之和正好等于余数值, BASICA 便从此字段剩余的后半部分开始取数据, 继而顺序向后字段取数据, 直到最后字段, 再接着从 0 字段开始顺序地取数据, 一直取到特定字段的前部分字段长数据为止 (此时字段序号相同, 序号前字符另取)。

例如, ao.dbf 库中有 11 个字段, 其各字段长之和为 65, 所以

$$\text{LEN} = 65 + 1 = 66$$

字段号 0 1 2 3 4 5 6 7 8 9 10 11

字段长 1 4 6 4 4 4 4 4 8 8 10 9

66

终止 9 1 起始

56

$$N = 32 + 32 \times 11 + 2 = 386$$

$$386 / 66 = 5 \cdots 56$$

因此, BASICA 开始从库文件的 10 号字段取数据, 其数据字符长度为 1, 即从此字段数据的最后一个字节取起, 接着再取 11 号字段数据, 而后再从 0 号字段取数据, 直到 10 号字段的另一部分数据取完 (见上示意图)。其 BASICA 取 ao.dbf 中数据的完整形式为:

.....

```

240 OPEN "ao.dbf" AS #3 LEN = 66

```

```

245 FIELD #3, 1 AS D10$, 9 AS D11$, 1 AS D0$, 4
AS D1$, 6 AS D2$, 4 AS D3$, 4 AS D4$, 4 AS D5$, 4 AS
D6$, 4 AS D7$, 8 AS D8$, 8 AS D9$, 9 AS E10$

```

在 245 行中, 起始字段与终止字段都是 10 号字段, 其前后数据字节数和为原字段长度, 字段序号均为 10, 仅名称不同, 起始处要与其它字段名同, 终止字段名可另取 (如 E10\$), 这样, dBASE III 数据库的数据便可直接传送到 BASICA, 不再需要软件接口过渡。值得一提的是中文数据的传送, 汉字占两个字符宽度。当采用库文件传送汉字时, 按上述公式组计算, 某特定字段长要被分成两部分, 此时若前部分字段长出现

奇数,则会使汉字传送失真。这是不允许的,为此,需返回 dBASE III 中修改数据库结构。为凑成被分部分字段长为偶数,宁可在某些字段中或前或后加入空白字

符来加大字段长度。这一问题在程序设计时就要考虑与确定,避免在程序调试中才来处理。

dBASE 运用经验与技巧

孙 鹏

一、浅谈汉字 dBASE III 与高级语言的接口问题

尽管汉字 dBASE III 具有很强的数据处理能力,但它对复杂的计算、图象处理、功能扩充等方面就显得不太适应,比不上 BASIC、PASCAL、FORTRAN、C 语言等。倘若能将汉字 dBASE III 与高级语言结合起来,相互取长补短,融为一个有机整体,是比较理想的,这就要解决汉字 dBASE III 与高语言的接口问题。dBASE III 调用高级语言是通过 RUN 命令实现的,被调用的高级语言可通过编译、连接形成 .COM 或 .EXE 文件(解 BASIC 除外)以外部文件存在磁盘上,只要高级语言已形成独立的可执行文件,汉字 dBASE III 总可以用语句 RUN 来调用它。

由于汉字 dBASE III 没有提供直接与高级语言的接口,因而它与高级语言的数据交换只能间接进行,但汉字 dBASE III 的文本文件是 ASCII 码文件,而某些高级语言都有处理 ASCII 码文件的能力,因此,实现汉字 dBASE III 与高级语言的数据传递,文本文件 .TXT 起着关键作用,实际上是汉字 dBASE III 与高级语言的数据接口。如果理清了高级语言数据文件的存储格式,那么与汉字 dBASE III 的数据交换是可以实现的。比如: PASCAL 语言有文本文件和顺序文件两种类型,那么我们就要选择文本文件,因为顺序文件是以二进制格式存储的,文本文件是以 ASCII 形式存储的; COBOL 语言有顺序、索引、直接文件三种类型,其中顺序文件是 ASCII 文件,数据之间没有分隔符,其存储形式与汉字 dBASE III 的标准文本文件格式一致,因此, COBOL 语言与汉字 dBASE III 交换数据时选择顺序文件比较合适; FORTRAN 语言分有格式和无格式输入/输出两种文件,而有格式输入/输出文件是以 ASCII 码形式存储的,所以选有格式输入/输出比较合适。下面主要以 PASCAL 语言为例进行说明。

PASCAL 语言数据文件是以空格作为数据分隔符的,因此,在 dBASE III 处理时,数据库的数据应使用 DELIMITED WITH BLANK 数据格式进行复制,下面的 PASCAL 语言程序是将汉字 dBASE III 的 SPP1.TXT 文件数据取出,经过处理后存于 SPP2.TXT 中,供汉字 dBASE III 程序使用。

汉字 dBASE III 程序:

```
:
USE SS
COPY TO SPP1 DELIMITED WITH BLANK
RUN SP
USE SS1
APPEND FROM SPP2 SDF
:
PASCAL 语言程序:
PROGRAM SP(INPUT,OUTPUT);
VAR
    SP1,SP2:TEXT;
:
BEGIN
:
    ASSIGN(SP1,'SPP1.TXT');RTSET(SP1);
    ASSIGN(SP2,'SPP2.TXT');REWRITE(SP2);
:
    READ(SP1,XM,NL);READLN(SP1);
:
    WRITE(SP2,XM,6,NL,2);WRITELN(SP2);
:
    CLOSE(SP1);CLOSE(SP2);
END.
```

综上所述,弄清高级语言数据文件的存储格式,是实现汉字 dBASE III 与高级语言数据交换成败的关键,不同语言有不同的数据存储格式,掌握了数据的存储格式,就能实现与汉字 dBASE III 的数据交换。

二、关于汉字 dBASE III 中 CONFIG.SYS 和 CONFIG.DB 文件的建立

1、CONFIG.SYS 文件的建立

在调试汉字 dBASE III 命令文件的过程中,打开 3 个以上的数据库文件,甚至有时只打开 1 个数据库文件,系统便发出打开文件太多的信息,产生此原因和 CONFIG.SYS 文件有关。因为不论在什么时候,系统启动时都要检测 CONFIG.SYS 文件,若不设定就按默认值执行,即同时打开文件数为 8 个,缓冲区为 2 个。而系统本身占用 5 个文件,为了在调试、使用汉字 dBASE III 的过程中,可打开足够的文件个数,需要对 CONFIG.SYS 文件重新加以设置,一般只需要写入如下 3 个语句:

DEVICE=ANSI.SYS

FILES=20

BUFFERS=15

因为在汉字 dBASE III 中,规定可以同时打开 10 个数据库文件和 15 个命令文件,所以文件的参数设置 20 个比较合适,缓冲区设置 10~15 比较合适,最多可不超过 20 个。为了加快系统运行速度,用户往往需要增加 DOS 缓冲区个数,但缓冲区设置过多并不是一件好事,一是,用户可用的内存空间就小了,因为每增加 1 个缓冲区就会使 DOS 在内存多占 528 个字节,故在内存容量较小情况下应设置较小的缓冲区;二是缓冲区的数量设置如果大于 20 个,系统可能会出现运行速度减慢的现象,原因是由于设置缓冲区过多,造成 DOS 在所有缓冲区中检索记录所用的时间要比直接从磁盘中读取记录的时间长。实践证明,对于内存是 640K(或 512K)的,设置 10~15 个缓冲区,系统就可以运行较快了,对于内存是 256K 的,设置 4 就可以了。总之,BUFFERS 的设置大小取决于用户计算机的内存容量,

2、CONFIG.DB 文件的建立

汉字 dBASE III 提供了内部的字处理程序来编辑命令文件,但它的功能有限,不能编辑超过 4K 的程序,但是,汉字 dBASE III 提供了另一种方法,在不退出 dBASE 的情况下,调用 EDLIN 和 WORDSTAR,不但可编辑大于 4K 的程序,而且还提供了一系列汉字 dBASE III 字处理程序所不具备的灵活编辑手段,这种方法是通过 CONFIG.DB 文件的设置来实现的,如果在该文件中设置:

TEDIT=EDLIN

WP=WS

两条命令,这样用户可直接在汉字 dBASE III 中用行编辑 EDLIN 建立和修改命令文件,编辑 MEMO 字段时,装入字处理程序 WORDSTAR。

在 CONFIG.DB 文件中,可设置功能键、SET 命令的 ON/OFF 状态、系统可用的最大内存空间、编辑方式等。下面是笔者建立的 CONFIG.DB 文件:

TALK=OFF	执行结果不送屏幕
CARRY=ON	将上一记录复制到当前记录
COLOR=3,4,6	设置屏幕属性
DEFAULT=A	当前驱动器为 A
DEVICE=SCREEN	@...SAY 命令送屏幕
ECHO=OFF	ESC 键终止命令文件
F2=MODIFY COMMAND	设置功能键 F2
MAXMEM=540K	可用最大内存为 540k
MVARISZ=5K	内存变量空间为 5k
PATH=\SP	设置路径
SAFETY=ON	设文件重写保护
PRINT=OFF	关闭打印机
TEDIT=EDLIN	编辑命令文件调 EDLIN
WP=WS	编辑 MEMO 调 WORDSTAR
COMMAND=DO SSS	进入 DBASE III 后自动执行 SSS.PRG

当汉字 dBASE III 启动时,系统自动查找 CONFIG.DB 文件,若找到自动装入,按该文件的设置来装配,执行后回到命令状态。

三、快速查找命令文件中错误的方法与技巧

在汉字 dBASE III 中,对发生的语法错误(标点符号错、命令名错、内存变量未初始化等)、结构错误(IF 和 ENDIF、DO WHILE 和 ENDDO 不配对等),系统会自动给出提示,而对逻辑错误,系统不会给出提示,这要给调试工作带来不便,我们不妨通过以下两个方面来调试程序。

1、使用汉字 dBASE III 中以下的几个 SET 命令

SET TALK ON 将命令执行结果显示在屏幕上,

可用该命令判断逻辑执行是否正确。

SET ECHO ON 将正在执行的命令显示在屏幕上或在打印机上输出

用该命令便于跟踪程序的执行流程。

SET STEP ON 执行一条命令后暂停

用该命令可把出错的语句孤立出来。

SET DEBUG ON 将 SET ECHO ON 输出送打印机以防止调试中产生的屏幕显示和程序运行中应产生的屏幕显示相混淆。

但使用以上几个 SET 命令比较浪费机时,特别是 SET STEP ON 命令,因而一旦调通,找到错误所在,就将它们置成 OFF 状态,最好的方法是将它们用在可能出错的程序段,以便节省调试时间。

2、采用 *、IF...ENDIF、DO WHILE...ENDDO 等,临时去掉程序的某些部分,使这部分程序段不被执行,来确定程序出错的根源所在。或者在程序适当的地方插入一些命令,如:?.??.@...SAY 或 WAIT 命令等,以便观察程序运行结果,有利于分段进行调试。

KD-GAL 编程器

KD-GAL 编程器通过和 IBM-PC 及兼容机的串行口进行通讯,全部编程过程是在“菜单”结构的提示下进行 100% 的高速编程,并具有自动比较,擦除等功能,操作灵活方便、快速。

编程器体积小,携带及使用方便,是当今电子产品研制、生产理想的工具。

该产品实行三包,终身保修。

我室在超声应用、微机软、硬件应用方面具有强有力的工程技术力量。可替代各用户单独设计用 GAL 芯片实现的各种规模的数字逻辑。

欢迎来人来函洽谈。

KD-GAL 编程器每台 1100 元(大专院校可优惠 10%)。

联系地址:安徽合肥科大无线电系

联系人:朱世鸿 邮政编码:230026

帐号:10514461865 开户行:合肥工商银行望办

单位:中国科大科技开发总公司

如何解决运行 FOXBASE 内存不够的问题

湖北天门市 121 信箱微机室 周日初

运行 FOXBASE 要求内存容量 $\geq 375k$, 但对于只有 640k 内存的 IBM PC/XT 机, 在装入 CCDOS (CCLIB 约为 232k) 后, 所剩内存不过 365k, 虽然能够进入 FOXBASE, 并且可以执行 FOXBASE 命令文件, 但运行速度还不及 DBASE III, 而且用 MODI COMM 来编辑文件时会出现无足够的内存空间。为了克服在运行汉化 FOXBASE 时因内存不够而产生的诸多毛病, 可以减少内存占用量。减少内存开销主要是考虑如何缩小汉字字库。

在字库中, 二级汉字有 3008 个, 这些汉字平时用得极少, 为了减少内存, 可以删除这些字, 这样 CCLIB 的长度就变为: $237632 - 3008 * 32 = 141376$

知道了 CCLIB 的长度, 就可用 DEBUG 来改变 CCLIB 文件的长度, 具体步骤如下:

```
debug cclib
-r
AX=0000 BX=0003 CX=A040 DX=0000 SP=FFEE
BP=0000 SI=0000 DI=0000
DS=567D ES=567D SS=567D CS=567D NV UP EI PL
NZ NA PO NC
567D:0100 0100 ADD [BX+SI], AX DS:
0003=00A0
```

```
-rbx
BX 0003
:0002
-rcx
CX A040
:2840
-W
Writing 22840 bytes
-q
```

若你不懂得寄存器 BX、CX 中数的含义可用下法求得: $BX = \text{INT}(141376/65536) = 2$ CX = HEX $(141376 - \text{INT}(141376/65536) * 65536) = 2840H$

注: 1、笔者所使用的机器为 IBM PC/XT, 内存 640k、25 行汉显, 用这种方法改完运行 FOXBASE 效果较满意。

2、若二级字库中有一部分汉字是你所必需的, 那么可以先使用笔者在去年十月份开发的软件修改字库, 再按上述步骤进行。该软件有一功能可把字库中的某个字移到字库的非汉字区, 需此源程序者, 请与本编辑部联系。(在运行此程序前, 还要用 EDLIN 编辑两个长度均为 25 字节的 HIGHTER 和 HIGHTER.ZH 文件, 其中 HIGHTER.ZH 为只读文件)

COMPRESS 在硬盘集中存储中的应用

常州 102 医院 许 鹰

由于信息的不断更新和增减, 一些文件, 尤其是较大的文件, 其存储的簇块往往分散在硬盘多个地方。当读取这些文件时, 就造成磁头的来回寻道读取, 使系统的运行时间大大增加, 同时也加快了磁头的磨损。文件增删次数越多, 文件数据变化越大, 存储的分散现象就越严重。因此, 有必要对这些文件及数据的存储区进行归并处理, 尽量使同一文件的信息存储在相连的簇块中。下面介绍一种简单而有效的方法: 利用 ptools 工具盘上的一个专用于压缩磁盘存储区, 处理零星分散簇块的实用程序 compress.exe 来处理。该程序在 ptools 4.11-5.1 版上都有。主要操作步骤如下:

1. 运行前先把硬盘上无用的文件删除掉。
2. 在硬盘上执行 CHKDSK/F 的 DOS 命令, 将散失簇块收集到一个文件中, 使压缩达到最佳效果。
3. 将带有 COMPRESS 的软盘插入 A 驱动器, 然后键入 COMPRESS, 这时出现主菜单。键入需要处理的驱动器号“C”。接着进行功能选择, 键入“5”, 即进行磁盘文件压缩处理。然后选择“C”。回车后, 屏幕上即出现所要压缩的磁盘物理图。可以明显地看到文件目录区后的各个文件已开始一个簇接一个簇地顺序重写, 而原来的文件则被暂时移到上空余的其它地方。等到上一文件完整地写完后, 这些“拆迁户”才被安排回来接下去写, 就这样直至所有文件重写完。最后自动清除掉散失的簇块。上述过程对 20 兆硬盘来说需要十几分钟。由于上述过程重排了文件在磁盘上的物理位置, 因此如果使用的汉字系统是 CCBios2.13A-F 的话, 需要对涉及文件在磁盘存储位置有关的高点阵字库的索引文件重新索引, 即重新执行一下 WORK24.EXE 文件, 以产生新的 24 点阵字库索引文件, 这样才能正确打印 24 点阵汉字。

怎样在 IBM-PC 机上实现汉字的动态显示

李晓华

所谓动态显示就是指在屏幕上的画面,能够按一定规则(通过编程)在屏幕上进行移动的图形。由于人眼的反应往往比较迟纯,将一些跳跃的运动看成是平稳的。因此,可应用这种视幻觉使图形在屏幕上移动。其具体方法是:首先画出该图形,接着消除该图形,然后在一个微小变化的位置处再显示出该图形。通过这一系列操作,就可以在屏幕上达到使得图形移动。由于在屏幕上显示的汉字是作为图形处理的特例,因此在屏幕上显示的所有的图形都能适用这一原理。

一、怎样使汉字(图形)在屏幕上“动画”

首先是屏幕上的光标定位,接着便是在该位置上显示出汉字,之后通过一段延时程序,使得汉字移动起来有一定的连贯性。延时完成后应马上把该位置的汉字消除。最后通过一定的计算,改变其汉字的位置,再显示汉字……。重复上述过程,这样汉字便在屏幕上移动起来了。

根据上述原理,利用 BASIC 语言的 PRINT 语句和 LOCATE 就能够使汉字在屏幕上移动起来。例一:

```
10 X=10
20 Y=1
30 LOCATE X,Y
40 PRINT "云"
50 FOR I=1 TO 100
60 NEXT I
70 LOCATE X,Y
80 PRINT " "
90 Y=Y+2
100 IF Y>80 THEN Y=1
110 GOTO 30
```

10,20 句给 X,Y 坐标赋初值。30,40 句光标定位,显示字符串。50,60 句延时。70,80 句清除被显示的字符。90 句对 Y 坐标进行累加,改变 Y 坐标的位置。100 句的判断使汉字移动时不超出屏幕范围。

二、怎样使汉字“字符串”在屏幕上“动画”

为使屏幕设计具有动态画面,使得汉字从不同角度一个个显示出来,增加图形的真实感,在程序中利用 BASIC 的 MID\$ 函数对字符串进行分解。在应用时要注意汉字的完整性,其余同上。例二:

```
20 H=20
30 X=2
40 CH$="在屏幕上实现汉字的动态显示"
50 FOR N=1 TO 26 STEP 2
55 Y=80
60 CH1$=MID$(CH$,N,2)
70 LOCATE X,Y
80 PRINT CH1$
90 FOR I=0 TO 10
100 NEXT I
```

```
110 LOCATE X,Y
120 PRINT " "
130 Y=Y-2
140 IF Y=H THEN GOTO 160
150 GOTO 70
160 H=H+2
170 LOCATE X,H
180 PRINT CH1$
190 NEXT N
200 END
```

三、具有一定速度和加速度的汉字(图形)“动画”

上面两个例子使我们看到了产生动态汉字的方法。通过这些“画—不画—移动—再画”的操作,就可以使汉字动态地移动。只要通过对运动路线进行计算就能使汉字图形沿更复杂的路径运动。但对于移动来说它必须和速度有关。对于显示器我们看成是三维的,因此定义了三种速度,即水平方向速度、垂直方向速度和加速度。例如,定义水平方向速度为 XV,垂直方向速度为 YH,水平方向加速度为 XVS,垂直方向加速度为 YHC。这样不仅可改变汉字的位置,而且也能改变移动速度,改变后的位置通过下面进行计算:

```
X=X+XVS
Y=Y+YHC
XVS=XVS+A
YHC=YHC+B
```

其中 A,B 分别为水平和垂直加速度。例三:

```
30 SCREEN 0;CLS
40 X=1;Y=1;V=1;H=1;A$="汉字动显示"
50 COLOR 7,0;LOCATE X,Y;PRINT A$;
60 LOCATE X,Y;COLOR 7,0;PRINT " ";
70 X=X+V
80 Y=Y+H
90 IF X=5 AND Y=32 THEN PRINT A$;:GOTO 130
100 IF X=23 OR X=1 THEN V=-V
110 IF Y=80 OR Y=1 THEN H=-H
120 GOTO 50
130 END
```

四、用 C 语言实现汉字动态移动

利用 C 语言实现汉字动态显示的方法与上述原理相同。首先定义一个外部变量 DXY(该变量为 X,Y 坐标,通过改变该变量从而使得汉字移动)。该程序中有一个主程序(MAIN())和一个函数(DISP(DXY)),程序 MAIN()用于显示汉字、消除汉字、计算 X(Y)的坐标值;DISP()函数用于光标的定位,该函数是通过调用 BIOS 的 10H 中断实现光标定位的。例四:

```
int dxy;
main()
{
```



```

extern int dxy;
for (dxy = 0x0501; dxy < 0x0546; dxy++)
{
    disp(dxy);
    printf("汉字动显示");
    disp(dxy);
    printf("      ");
}
disp(dxy)

```

```

extern int dxy;
struct {int ax,bx,cx,dx,si,di,ds,es;}rv;
rv.ax = 0x0200;
rv.dx = dxy;
rv.bx = 0x0000;
sysint(0x10,&rv,&rv);
return(dxy);

```

DOS I/O 重定向、管道功能及其使用

江西省经济信息中心 徐国环

DOS 版本 2 增强的功能之一是支持类似于 UNIX 的 I/O 重定向和管道。如果用户能够熟悉并灵活使用它们,将会带来极大的方便。

I/O 重定向

DOS 数据源设备和目的设备叫做标准输入和标准输出,通常键盘为标准输入,显示器为标准输出,它们为两个预定的 I/O 通道,可被任何程序存取。在 I/O 重定向中,用户可以改变上面用于正常输入和输出的源及目的,即通过 DOS 命令行中的参数使这些通道之一重定向到某一设备(如打印机)和磁盘文件上。

“<”表示重定向程序的输入符,“>”表示重定向程序的输出符,“>>”是把重定向的程序输出加到已建立的文件上。例如命令 DIR>PRN,它不是在显示器上显示,而是输出到打印机上。

管道

管道是伴随 I/O 重定向而来的。一个管道是一条连接两个程序的计算机处理线,就是把一个程序的输出变为另一个程序的输入。“|”表示管道操作符。如 TYPE LOCK|DEBUG 这条命令表示把 TYPE 显示的内容作为 DEBUG 的输入。

筛选程序

DOS 提供了 SORT、FIND、MORE 三个筛选程序。它们是 DOS 支持 I/O 重定向和管道环境下运行的。筛选程序是将一字符流作为输入,对其进行处理并将处理后的字符流作为输出。文本流源和目的也可为任何字符设备,文件或系统所知的程序。SORT 对文本数据排序;FIND 寻找输入流中与指定字符串匹配的字符串;MORE 一次显示一屏数据。例如:

DIR|MORE

表示一次显示一屏 DOS 执行 DIR 信息,按空格键显示下一屏信息。

SORT<WORDS|MORE

表示 DOS 从 WORDS 文件中接收字,并对这些字进行排序,得出的结果,每次通过 MORE 显示整屏行。

DIR|FIND“01-01-90”|SORT>PRN

表示 DOS 在 DIR 中查找“01-01-90”字符串的

行,结果送给 SORT 排序且把输出送给打印机打印。

综合使用实例

例 1:把 COMMAND.COM 反汇编程序转为磁盘文件,以便转换到高速打印机(如 IBM 3200)打印出来进行分析。

1. 建立文件 CFILE

n command.com

l

u 100 4680

q

2. 键入下面命令

TYPE CFILE|DEBUG>UCOMM

UCOMM 就是 COMMAND.COM 的反汇编程序的磁盘文件。

例 2:对 XGH 子目录进行加密和解密。其原理是利用 DEBUG 修改子目录开始簇号,即第 26、27 字节。

1. 用 DEBUG 调磁盘目录区记下 XGH 子目录的第 26—27 字节在段内偏移和数值,它们分别为 1FA,1FB 2D,00

2. 建立下面两个文件

LOCK

1 100 2 11 10

elfa 00 2d

w 100 2 11 10

q

KEY

1 100 2 11 10

elfa 2d 00

w 100 2 11 10

q

3. 加密时键入 TYPE LOCK|DEBUG>S。解密时键入 TYPE KEY|DEBUG>SS。这两个命令是把 TYPE 的输出作为 DEBUG 的输入,修改和恢复子目录开始簇号,同时把显示的内容输出到给定的磁盘文件上使 LOCK 和 KEY 文件在 CRT 上不显示。

DOS I/O 重定向和管道功能除了上述用法外,还可以用来调试程序,程序员可以通过建立用于程序的“回答”磁盘文件(代替键盘输入)来调试程序,运行程序时把输入重定向到该磁盘文件上,可减少调试时间。用户也可编写自己的筛选程序,从而更好地利用 DOS 环境下提供的 I/O 重定向和管道的强大功能。

在低版本 DOS 下 安装虚拟盘

江苏江阴 周健勇

虚拟盘素有执行速度快、磁盘与驱动器磨损小等优点,但只在 DOS 3.0 及以下的系统才具有虚拟盘功能,因为它提供了一个外部命令文件 VDISK.SYS。我们也能利用它在 PC/XT 机的低版本 DOS 下安装虚拟盘。具体方法是:

①将 VDISK.SYS 文件拷到低版的 DOS 系统盘上。如 DOS 2.1 盘。

②在 DOS 2.1 系统盘上的 CONFIG.SYS 文件中增加一条 DEVICE=VDISK.SYS bbb[,sss][,ddd]命令。

bbb——虚盘的容量,单位 k,默认值 64k。

sss——虚盘的扇区大小,单位为字节,可选 128、256、512,默认值 128。

ddd——虚盘目录项数,可选 2~512,默认值 64。

例如:建立一个 120k 容量、扇区为 128、目录项为 64 的虚拟盘,则将:

DEVICE=VDISK.SYS 120,128,64

写入 CONFIG.SYS 文件中。

③用该 DOS 2.1 系统盘重新启动机器,屏幕显示虚拟盘的容量、扇区大小及目录项数:

VDISK Version 3.2 virtual disk

Buffer size: 120KB

Sector size: 128

Directory entries: 64

Current date is Tue 1-01-1980

Enter new date:

至此,已安装成功,就可以在虚盘上进行操作了。若机器已装有硬盘,则虚拟盘提示符为 D>。

本例在 Super PC/XT-Ⅲ、IBM PC/XT 286 两种机器上通过。

五笔 CCDOS 中 一字节修正

上海现代信息技术研究所 吴邦忠

五笔字型 CCDOS 汉字操作系统以其独特的编码优点被越来越多的国内外微机用户所采用。然而在实际使用该操作系统中我们发现,当用组合键 ctrl-F10 进入改变打印字号、打印行宽状态时,如果在已经键入打印字号后(即:此时光标停在等待接收行宽参数位置),先从键盘打入退格键(backspace 键)就会发现,此后系

统只接收键盘输入的字母(A-P)、回车(Enter)及退格键,而不接收数字键,而且当键盘输入有效字符(A-P)和回车或直接键入回车后,系统就结束定义打印字号、行宽的过程,返回提示符 C> 状态,造成不能改变打印行宽。我们经过分析 CCBios 中键盘管理模块(int 16H)后发现,该模块在处理 ctrl-F10 组合键的程序段(CS:3198-CS:32B2)中,主要是由于以下二个指令:

```
CS:31f9 CMP byte ptr[2896],02
```

```
CS:31fe JB 325b
```

判断有误,其修改方法如下:

```
1. C>ren peng4.exe peng4
```

```
2. C>debug peng4
```

```
-E cs:34fd 03
```

```
-W
```

```
-Q
```

```
3. C>ren peng4 peng4.exe
```

将经过修改后的 peng4.exe 文件引导装入,就不会产生本文上面所述的问题。如果先前已经进入汉字操作系统,则须重新启动(热启动或冷启动)计算机(PC-XT 或其它兼容机),再将 peng4.exe 文件引导装入。

注:单元[CS:2896]为汉字输入码缓冲区计数器。

一个快速实用的 字串替换程序

杭州浙江电子学校 金林樵

用手工进行字串的替换,是件头痛的事,搞不好,漏掉其中的一个字串没有修改,则程序的正常运行就会受到影响,特别是替换变量名时,就使程序产生错误的运行结果。下面的程序可为你在这方面排忧解难,使用它可以进行快速批量替换,既准确又迅速,它可使原文中的所有相同短字串替换成所需的长字串,也可使原文中的所有相同长字串均改为所需的短字串。

运行此程序时,输入的文件必须是 ASCII 码文件。按提示输入要编辑的文件名、替换和被替换字符串后,程序就将此文件读入内存,然后快速替换,替换的结果在屏幕上显示出来,以利检查。替换结束后,在当前盘上产生一个扩展名为“XRF”的同名文件,而原文件保留不变。新文件运行通过后,可将扩展名改为原类型。

```
100 REM 字符替换
```

```
110 CLEAR:KEY OFF:CLS
```

```
120 J=0:M=999:OPTION BASE 1:DIM P$(M),P1$(M)
```

```
130 INPUT "请输入文件名: ",FILE$
```

```
140 OPEN FILE$ FOR INPUT AS #1
```

```
150 WHILE NOT EOF(1)
```

```
160 COUNT=COUNT+1
```

```
170 LINE INPUT #1,P$(COUNT)
```

```
180 WEND
```

PC 机磁盘文件的加密与解密

上海中国纺织大学计算机系 刘忠达

在讨论 PC 机磁盘文件的加密与解密以前,先讲述一下磁盘结构。双面的 $5\frac{1}{4}$ 英寸软盘每面各有 40 个磁道,每道又分九个扇区,每个扇区有 512 个字节。DOS 把所有扇区划分为四个部分:

1. 引导程序区(IPL)

它占有 0 扇区,存放格式字、规格表和自举程序。

2. 文件分配表(FAT)

文件分配表占有 1—4 扇区,纪录文件占有空间,在 FAT 中存放每个文件占有簇(Cluster)的链接表。PC-DOS 规定每两个扇区为一簇,每簇占 1024 字节空间。FAT 共有一式两份文件分配表,各占两个扇区。

3. 文件目录表(DIR)

文件目录表从 5 扇区到 B 扇区,占有 7 个扇区,用来存放磁盘文件目录。

4. 资料区

从 C 扇区开始,共 608 个扇区,362496 个字节。这个数字也就是用不加 S 参数的 FORMAT 命令格式化后得到的磁盘空间。资料区用来存放文件正文。

文件目录表共可存放 112 个文件目录。PC-DOS 对目录的每个字节的含意都作了明确的规定如下:

0—7:磁盘文件名

8—A:扩展名

B:文件档案属性,标准规定如下:

01:表明该文件为只读文件;

02:表明该文件为隐含文件;

10:表明为一个子目录档案;

20:正常文件属性

C—15H:未用

16H—17H:文件建立时间

18H—19H:文件建立日期

1AH—1BH:起始簇号

1CH—1FH:文件长度

明白了上述磁盘结构,借助 DEBUG 调试工具便可做一些简单的加密与解密工作。

许多人掌握的简单加密方法有以下三种:

1. 在文件名中加入隐含字符;

2. 在文件名中插入半个汉字;

3. 修改文件属性,达到加密目的。

对于方法 1 和方法 2 的解密,可利用 DEBUG 的 E 命令将目录项中的 0—A 字节改为一般字符的 ASCII 码(十六进制)即可。对于方法 3,也可用 E 命令将文件属性改为正常文件。这样就完成了解密工作。

下面介绍一种新的加密方法。我们知道目录项中的 C—15H 字节未用,我们可利用其中的 C 和 D 两字节来进行文件的加密工作。方法如下:

1. 进入 DEBUG

2. 利用 L 命令将磁盘的目录区调入内存

3. 利用 D 命令找到欲加密文件的目录项

4. 读该目录项 1AH 和 1BH 字节的内容,设为 0E,00,用 E 命令将它们写入该目录项的 C 和 D 字节。

5. 将该目录项的 1A 和 1B 字节置为任意数值(不同于原来数值)。

6. 用 D 命令察看,此时目录项中的起始簇号已被改变。

7. 利用 W 命令将目录区写回磁盘。

8. 退出 DEBUG。

若要加密多个文件,只要重复步骤 3,4,5。

当执行被加密的文件时,计算机无法正常读盘,给出出错信息。

解密是加密的逆过程,步骤如下:

1. 进入 DEBUG

2. 利用 L 命令将磁盘的目录区调入内存。

3. 将起始簇号(存于目录项的 C 和 D 字节)写回目录项的 1A 和 1B 字节。

4. 利用 W 命令将目录区写回磁盘。

5. 退出 DEBUG。

详细操作过程附后。

```
190 CLOSE #1
200 FILE$ = LEFT$(FILE$, INSTR(FILE$, ".") +
"XRF"
210 INPUT "键入原字符串:", SEARCH$
220 INPUT "替换成:", AP$
230 FOR I=1 TO COUNT
240 PTR=INSTR(P$(I), SEARCH$)
250 IF PTR=0 THEN 290
260 ZC$ = RIGHT$(P$(I), LEN(P$(I)) - PTR -
LEN(SEARCH$) + 1)
270 P1$(I) = P1$(I) + LEFT$(P$(I), PTR - 1) + AP
$: P$(I) = ZC$
```

```
280 J=J+1:GOTO 240
290 P1$(I) = P1$(I) + ZC$: ZC$ = ""
300 IF P1$(I) <> " THEN PRINT P1$(I) ELSE P1
$(I) = P$(I)
310 NEXT I
320 OPEN FILE$ FOR OUTPUT AS #1
330 FOR I=1 TO COUNT
340 PRINT #1, P1$(I)
350 NEXT I
360 CLOSE #1
370 END
```



```

A: \>debug
-1 100 0 5 7
-d 1a0
56DE,01A0 41 53 43 49 49 20 20 20-42 41 53 20 00 00 00 00 ASCII BAS ....
56DE,01B0 00 00 00 00 00 00 F3 08-21 00 1F 00 F9 00 00 00 .....鑫!...y...
56DE,01C0 51 55 57 45 49 4D 41 20-42 41 53 20 00 00 00 00 QUWEIMA BAS ....
56DE,01D0 00 00 00 00 00 00 54 00-21 00 20 00 BD 00 00 00 .....T.!.疆..
56DE,01E0 4B 45 59 32 20 20 20 20-42 41 53 20 00 00 00 00 KEY2 BAS ....
56DE,01F0 00 00 00 00 00 00 55 0A-21 00 21 00 37 01 00 00 .....U.!.!.7..
56DE,0200 42 41 53 49 43 41 20 20-43 4F 4D 20 00 00 00 00 BASICA COM ....
56DE,0210 00 00 00 00 00 00 60 60-68 06 0E 00 80 65 00 00 ..... 'h....e..
-e 20c 0e 00
-e 21a 22 22
-d 200 21f
56DE,0200 42 41 53 49 43 41 20 20-43 4F 4D 20 0E 00 00 00 BASICA COM....
56DE,0210 00 00 00 00 00 00 60 60-68 06 22 22 80 65 00 00 ..... 'h."".e.
-w 100 0 5 7
-q
A: \>basica
Sector not found error reading drive A
Abort, Retry, Ignore? a
A: \>debug
-1 100 0 5 7
-e 21a 0e 00
-w 100 0 5 7
-q
A: \>basica

```

硬盘正确格式化一例

袁士彬

微机在使用过程中,若不小心被病毒严重感染,系统不能正常运行时,需要对硬盘格式化;有时在编制涉及系统调用的程序时,往往会破坏系统,这时亦需要对硬盘格式化。有时因操作不小心而致使系统丢失,屏幕显示:

SYSTEM HALTED

这时你只要插入与原 DOS 匹配的系统盘重新启动微机,打入如下命令:

A>SYS C:✓

即可恢复硬盘系统。如果你用了不匹配的系统盘,特别是用了较原版本高的系统盘进行如上操作的话,就必须对硬盘进行格式化。根据我们在实际工作中的经验,如果格式化方法不当,是不能装入 DOS 系统重新使用硬盘的,下面给出实际工作中正确格式化硬盘一例。

根据判断认为硬盘需要格式化后,先用完好的 DOS 软盘启动微机,用 FDISK.COM 在硬盘上建立 DOS 分区,FDISK 给出以下五种选择:

- ① Create DOS Partition
 - ② Change Active Partition
 - ③ Delete DOS Partition
 - ④ Display Partition Data
 - ⑤ Select Next Fixed Disk Drive
- 选①建立 DOS 分区,屏幕显示:

DOS Partition Has Created

再对硬盘格式化:

A>Format c:/s/v✓

这时屏幕提示敲入 C 盘的卷标,但无论你敲入的卷标如何,均不对硬盘进行格式化。我们分析可能是因为 DOS 分区实际上已被破坏的缘故,这时需要使用 LOWFORM.EXE 文件进行低级格式化,但经常对硬盘进行低级格式化是很不利的,而且一般系统盘中不配有 LOWFORM.EXE 文件。既然 DOS 分区被破坏了,那么我们用 FDISK.COM 文件先将原 DOS 分区删除,再重新建立一个 DOS 分区,因而得到硬盘正确格式化方法如下:

1. 用系统盘启动微机后,键入如下命令:

A>FDISK✓

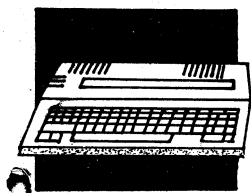
在屏幕给出提示后选③,删除原 DOS 分区。待再次给出五种选择时,再选①重新建立 DOS 分区。

2. 对硬盘进行格式化并装入系统。

A>FORMAT C:/S/V✓

3. 用 COPY 命令装入常用的 DOS 文件。

至此,即可得到一个正常可用的微机系统。



学习机之友

程序出错处理及 RESUME 的功能扩充

北京机工印刷厂 丁志伟

在有错误处理的 BASIC 程序中,常会遇到 ONERR GOTO 和 RESUME 语句,下面介绍功能和用法。

1. ONERR GOTO. 设置一个转向行号,常放在程序开始。执行这个语句之后,运行时遇到程序有错,就转到它指定的错误处理程序,例如:

```
10 ONERR GOTO 1000
20 A=3:B=5
.....
1000.....
1020 RESUME
```

第 10 句执行过之后,20 句本意是要执行两个赋值语句,但误把冒号打成分号,出现了语法错误,这时就会转到 1000 句的错误处理程序。如果程序中没有发生错误,ONERR GOTO 则不会产生影响。

2. 上面的小程序中,从 1000 句开始为错误处理程序。它应根据错误性质做相应的处理。最后一条 RESUME 语句类似子程序的 RETURN,它使程序运行返回到发生错误的位置。如果处理得当,程序就象没发生过错误一样,继续运行。

3. 错误处理程序不应再出错。它的第 1 条语句常是 POKE 216,0。其作用是清除 ONERR GOTO 设置的标志。此后再发生错误时,程序就中断运行,避免因错误处理部分发生错误而陷入死循环。

下面重点讨论 RESUME 语句。

前面提到,RESUME 语句的作用是返回到发生错误的位置。但是有些错误难以处理,比如磁盘存取错误;或者其它原因,希望在处理错误之后(也可能不处理错误)转到其他行号,RESUME 就难以满足要求了。这时可以想到用其他语句转出,比如用 GOTO 语句。但是,用其他语句脱离错误处理程序在某些情况下是行不通的。下面举例说明,见程序 1:

```
10 ONERR GOTO 1000
500 A=3/B
505 PRINT A
510 STOP
1000 POKE 216,0
1005 B=1
1010 RESUME
```

]TRACE✓

]RUN✓

10 # 500 # 1000 # 1005 # 1010 # 500 # 505 3 # 510

Break in 510

第一次执行 500 句除数 B 为 0,计算发生错误,因为有了 10 句的设置,程序接下来会转到 1000 句。1000 句清除出错转向标志。此后程序再出错,BASIC 解释

程序会发出错信息并停止运行。1010 句的 RESUME 返回发生错误之处,也就是 500 句。以上执行情况可以看出,程序运行正常。如果把 1010 句改为 GOTO 500 或 GOSUB 500,效果也相同。在这个小程序中,RESUME、GOTO、GOSUB 这三个语句做为错误处理程序的结尾都能使程序正常运行。再看下面的程序 2:

```
10 ONERR GOTO 1000
20 GOSUB 500
30 STOP
500 A=3/B
505 PRINT A
510 RETURN
1000 POKE 216,0
1005 B=1
1010 RESUME
```

程序 2 所要完成的工作和程序 1 相同,只是把除法改由子程序做。再看执行情况:

]RUN✓

10 # 20 # 500 # 1000 # 1005 # 1010 # 500 # 505
3 # 510 # 30

Break in 30

]1010 GOTO 500✓(改变 1010 句)

]RUN✓

10 # 20 # 500 # 1000 # 1005 # 1010 # 500 # 505
3 # 510

? RETURN without GOSUB error in 510

当 1010 句是 RESUME 时,子程序工作正常,而改为 GOTO 500 后,就会使 GOSUB 好象没有执行过。RESUME 到底做了些什么工作呢?看看它的处理程序就清楚了:

```
F318- A5 DA LDA $DA
F31A- 85 75 STA $75 恢复出错时行号
F31C- A5 DB LDA $DB
F31E- 85 76 STA $76
F320- A5 DC LDA $DC 恢复出错时扫描指针
F322- 85 B8 STA $B8
F324- A5 DD LDA $DD
F326- 85 B9 STA $B9
F328- A6 DF LDX $DF 恢复堆栈指针
F32A- 9A TXS
F32B- 4C D2 D7JMP $D7D2 执行下一条语句
```

这段程序中使用了几个零页存储单元:

\$DA、\$DB:存放发生错误时的行号;
\$DC、\$DD:存放发生错误时的程序扫描指针;
\$DF:存放发生错误时的堆栈指针。

\$ 75、\$ 76 存放正在执行语句的行号。

\$ B8、\$ B9 执行程序时用来扫描程序的指针。

RESUME 的功能就是恢复出错时的三个指针,使程序能返回到原来出错的位置上继续运行。\$ F318~\$ F327 的功能完全可以由 GOTO 语句完成。关键在于 \$ F328~\$ F32A 这两条,即恢复栈指针。

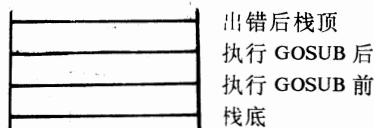


图 1

图 1 是程序 2 使用堆栈情况的示意。执行 GOSUB 语句时,用到了堆栈,错误发生时,也用到了堆栈;堆栈的功能,是后进先出,进栈和出栈要匹配。1010 句如果改为 GOTO500,直接转到 500 句,继而执行 510 句 RETURN 时,堆栈指针却还指向出错时栈顶位置,出栈的内容不是 20 句 GOSUB 入栈的内容,入栈和出栈不匹配,BASIC 解释程序就会认为运行有误。

程序 1 之所以能正常运行,是因为出错之前没有调用含错误的子程序,不存在入栈、出栈不匹配问题。为了使问题更加明显,再看程序 3:

```
10 ONERR GOTO 1000
20 A=3/0
30 STOP
1000 PRINT PEEK (223),
1005 IF PEEK (223)< 90 THEN STOP
1010 GOTO 20
```

地址 223(\$ DF)存放出错时栈指针。程序出错和进入子程序时,都会有数据入栈,栈指针数值变小。

```
NO TRACE✓
]RUN
248 239 230
221 212 203
194 185 176
167 158 149
140 131 122
113 104 95
86
```

Break in 1005

由于程序不断出错,错误处理程序又没使用 RESUME,数据不断入栈。6502 的堆栈只能存放 256 个字节,一旦堆栈装满,就会造成难以恢复的混乱。如果把程序 3 的 1005 句去掉再执行,就会看得更清楚。

综上所述,想让程序安全运行,在没把握时,出错处理的结束不要輕易用 GOTO 代替 RESUME。

若不想返回原来出错位置,又该怎么办呢?为此我编了一段机器语言子程序(程序 4):

```
0300- 20 0C E1 JSR $E10C
0303- A5 A1 LDA $A1
0305- 85 DA STA $DA
```

```
0307- 85 50 STA $50
0309- A5 A0 LDA $A0
030B- 85 DB STA $DB
030D- 85 51 STA $51
030F- 20 1A D6 JSR $D61A
0312- 38 SEC
0313- A5 9B LDA $9B
0315- E9 01 SBC #$01
0317- 85 DC STA $DC
0319- A5 9C LDA $9C
031B- E9 00 SBC #$00
031D- 85 DD STA $DD
031F- 60 RTS
```

这段程序的作用是,在执行 RESUME 之前,先把出错处理后返回的行号和扫描指针计算出来,修改出错指针(\$ DA~\$ DD 内容)。执行 RESUME 时,程序就会返回指定的位置。子程序 \$ E10C 将 USR(n)送到浮点累加器 \$ 9D~\$ A2 的值转换成二字节整数后存入 \$ A0(高位)和 \$ A1(低位)。子程序 \$ D61A 根据 \$ 50、\$ 51 单元提供的返回行号;找到扫描指针后存入 \$ 9B、\$ 9C。最后将扫描指针减一,调整到上一 BASIC 语句行的结束标志处,再存入 \$ DC、\$ DD。

程序 4 在使用时要和 BASIC 程序配合,用 USR(n)函数传递出错处理后返回的行号到浮点累加器 \$ 9D~\$ A2。在执行 USR(n)前,应先在 \$ 0A~\$ 0C 中放一条 JMP 指令,以便转到程序 4。例如程序 5:

```
1000 POKE 216,0
1005 B=1
1010 HH=505
1015 POKE 10,76;POKE 11,0;POKE12,3
1020 HH=USR(HH)
1030 RESUME
```

1010 句的 HH 被赋予执行 RESUME 时想转去的行号,这里定为 505。1015 句将处理 USR(n)函数的进入地址放入 \$ 0A~\$ 0C 单元。这是一条跳转指令 JMP \$ 300。1020 句通过 USR(n)函数将 HH 的值放入浮点累加器(\$ 9D~\$ A2),然后执行 \$ 0A~\$ 0C 单元的指令 JMP \$ 300 进入程序 4。通过程序 4,\$ DA~\$ DD 的内容已被修改,以后执行 RESUME 时,就会返回行号 505。如想转去其它行号,可对 1010 句的 HH 赋予相应的值。

值得注意的是,如果是子程序出错,最好不要让错误处理子程序直接转出子程序,尤其是子程序嵌套的情况下更是如此。应当先返回到子程序内部由 RETURN 语句退出子程序或者执行完 RESUME 语句后再执行一个 POP 语句。原因也是为了堆栈的进栈和出栈相匹配。例如想在出错后转到第 10 句,可以这样:

```
1010 HH=2000
1015 HH=USR(HH)
.....
1030 RESUME
2000 POP,GOTO 10
```


特殊 6502 指令

笔者 87 年编了一个 6502 跟踪程序,在调试中,无意跟踪了一个“???”指令,发现这个指令长度并不是一个字节,而且对寄存器也有影响。经分析,发现这是 6502 设计师在设计 6502 时,未禁止除合法指令外的其它代码而产生的误动作。

最近见一些刊物上介绍这些指令,但多不全面。如有的称“6502 未公布指令”就不妥,因为并不是 6502 设计师有意设计的。为使读者能正确使用这些指令,现将研究资料整理发表。(见封二、三表①—④)

这些代码存在着规律性,这种规律性是由 6502 译码电路决定的。产生误动作的原因(原理)如下:

指令码读入指令寄存器后,指令译码器将 D_0VD_1 , 为零触发 A 类(表①)指令处理, $D_0=1$ 触发 B 类(表②)指令处理, $D_1=1$ 触发 C 类(表③)指令处理。尔后再进一步处理,如 B 类,由 $D_7D_6D_5$ 确定操作性, $D_4D_3D_2$ 确定寻址方式(周期,指令长度)等。

$D_0\wedge D_1$ 非零未定义,但也未禁止。所以当 $D_0\wedge D_1=1$ 时, $D_0D_1=11$, 同时触动了 BC 两类指令的一些操作,进而产生了误动作。基本原理就是这样。

由于未禁止非指令码,所以几乎所有非指令码都能产生误动作,但有明显作用的只有 66 条。另还有 $0\times\times\times 0010, 1\times\times\times 10010$ 两类指令,也有作用(死机),其原因是这些代码触发了 CPU 循环执行一些操作。

关于误动作指令执行所需周期,表④所列的,由 $D_4D_3D_2$ 三位决定,可参照表②对应的指令(有移位动作+2,否则相同)。

误动作指令可用于加密;编程可节省空间;执行一些特殊操作等。但这些指令只能用于 6502, 6502A 芯片,而对 65C02(Apple II C 机芯片)却不能用,因为在 65C02 中,这些误动作代码已被定义为新的指令。

(注:表中 6 条指令含意保留,因为这 6 条指令动作复杂,实用性不强,所以留做加密用。)

四舍六入法程序

浙江金华粮食学校 韦肖鸿

四舍五入是大家熟悉的一种数字修约规则。但大量的实践表明,采用四舍五入法修约数据,会造成系统误差,使结果的误差增大,因为采用四舍五入法处理数据,进位的项数有 5 个,舍去的项数则是 4 个,数字修约时进位的可能性比舍去的可能性要大。因此,修约后数值的平均值会出现偏离原来实际数值的平均值的情况。为了克服四舍五入中的上述缺陷,人们提出采用四舍六入法来修约数据,其具体规则如下:

四舍六入五考虑,五后非零则进一,五后皆零看奇偶,五前为偶应舍去,五前为奇则进一。

这种修约规则可使有的 5 进位,有的 5 舍去,从而让因修约而引起的正负误差有相互抵销的机会,达到进舍项数相平衡,使舍入误差总和等于零的目的。

在 BASIC 语言中,对四舍五入法修约数据,一般就直接通过 $\text{INT}(X)$ 函数实现。如给出下式:

$$\text{INT}(X * 10^N + 0.5) / 10^N$$

就可对数据 X 保留 N 位小数,并对小数点后第 N+1 位进行四舍五入的处理。

对数据进行四舍六入的修约没有专门的函数能够实现,这时可编制一个专门的子程序来模拟四舍六入的功能。下面的子程序供大家参考。

```
100 X$=STR$(X),L=LEN(X$)
105 FOR I=1 TO L
110 IF MID$(X$,I,1)="." THEN 125
115 NEXT I
120 GOTO 170
125 Y$=MID$(X$,I+N+1,1)
130 IF VAL(Y$)>=6 THEN X=X+10^(-N),GOTO 170
135 IF VAL(Y$)<=4 THEN 170
140 FOR J=I+N+2 TO L
145 IF VAL(MID$(X$,J,1))<>0 THEN X=X+10^(-N); GOTO 170
150 NEXT J
155 Y1$=MID$(X$,I+N,1)
160 IF VAL(Y1$)/2=INT(VAL(Y1$)/2) THEN 170
165 X=X+10^(-N)
170 X=INT(X*10^N)/10^N
175 PRINT X
180 RETURN
```

以上程序中的变量 N 是要求保留小数的位数, X 是要求修约的数据。将该子程序插入到一些有关的数据处理程序中,就可实现对数字的四舍六入修约,从而可保证数据处理结果的更合理与更高准确度。

“怪”指令在软件加密中的应用

舟山市电力公司变电工区 傅剑

本刊 90 年第 8 期(总第 65 期)《利用 BOOI1 对磁盘进行机密》一文,虽然能起到保护效果,但笔者认为此文加密过程有如下四点缺陷:

1. 该文加密程序不但占用了 DOS3.3 程序修补区中的 APPEND 和 VERIFY 程序修补区,而且覆盖了 LDX \$2B 指令(取得槽号 * 16,以便进入 BOOT2),这样的 BOOT1 进入内存后,将会使 DOS 命令 APPEND、BSAVE、SAVE 出错。

2. 必须使用 NIBBLES AWAY、ZAP 等扇区修改软件,没有这些软件的用户就无法用该方法加密。

3. 对每片加密磁盘都需修改 0 轨 0 区的内容,因此加密过程繁琐,不适合大量制作。

4. 该文作者只考虑了设置保密口令,而对 RESET 和启动跟踪都没有防范措施,因此很容易被破解。

笔者针对上述四点的缺陷,同样对 BOOT1 进行一些修改,但不同的是直接修改 BOOT1 在内存中的映象,并且避开了 DOS3.3 程序修补区,这样不但无需利用工具程序来修改软盘 0 轨 0 区中的 BOOT1,而且在 DOS3.3 进入内存后,不影响 APPEND、BSAVE、SAVE 等命令功能。修改后的 BOOT1 映象可以用 DOS 命令中 INIT 直接写入被加密磁盘,操作简便,适于大量制作。笔者在 BOOT1 中也设置了口令保护(口令是:1968111),而且封锁了 RESET 功能,并用目前知者不多的 6502 怪指令编写关键加密程序,使启动跟踪破密者不明究竟,而放弃跟踪。具体做法如下:

1. 开机,引导正常的 DOS3.3
2. 输入 CALL-151 命令,进入监控模式。
3. 输入如下机器语言程序:

B645—	AB	???	,地址 B645~B649
B646—	B3	???	为 6502 怪指令,
B647—	8F	???	相当于正常指令
B648—	4B	???	LDA # B3 和 STA
B649—	08	PHP	\$ 84B
B64A—	4C 5E 08	JMP	\$ 085E

B6B3—	AB	???	,地址 B6B3~B6C6
B6B4—	5E 8F 4B	???	\$ 4B8F,X 为 6502 怪指令,
B6B7—	08	LSR	相当于正常指令;
B6B8—	AB	PHR	
B6B9—	00	???	LDA # 5E
B6BA—	8F	BRK	STA \$ 84B
B6BB—	F2	???	LDA # 00
B6BC—	03	???	STA \$ 3F2
B6BD—	AB	???	LDA # C6
B6BE—	C6 8F	DEC	\$ 8F STA # 3F3

B6C0—	F3	???	LDA # 63
B6C1—	03	???	STA \$ 3F4
B6C2—	AB	???	
B6C3—	63	???	
B6C4—	8F	???	
B6C5—	F4	???	
B6C6—	03	???	
B6C7—	20 58 FC	JSR	\$ FC58
B6CA—	AD FC 08	LDA	\$ 08FC
B6CD—	20 ED FD	JSR	\$ FDED

B6D0—	A9 00	LDA	# \$ 00	
B6D2—	A2 00	LDX	# \$ 00	
B6D4—	20 0C FD	JSR	\$ FD0C	
B6D7—	DF	???		
B6D8—	F5 08	SBC	\$ 08,X	,地址 B6D7 ~
B6DA—	DD F5 08	CMP	\$ 08F5,X	B6D9 为 6502
B6DD—	D0 D4	BNE	\$ B6B3	怪指令,相当于
B6DF—	DF	???		正常指令;DEC
B6E0—	F5 08	SBC	\$ 08,X	\$ 8F5,X
B6E2—	E8	INX		
B6E3—	E0 07	CPX	# \$ 07	
B6E5—	D0 ED	BNE	\$ B6D4	,地址 B6DF ~
B6E7—	20 2F FB	JSR	\$ FB2F	B6E1 为 6502 怪
B6EA—	CF	???		指令,相当于正
B6EB—	EF	???		常指令;DEC
B6EC—	08	PHP		\$ 8F5,X
B6ED—	A6 2B	LDX	\$ 2B	
B6EF—	6D FD 08	ADC	\$ 08FD	
B6F2—	CF	???		
B6F3—	EF	???		,地址 B6EA ~
B6F4—	08	PHP		B6EC 为 6502 怪
B6F5—	B2	???		指令,相当于正
B6F6—	BA	TSX		常指令;DEC
B6F7—	B7	???		\$ 8EF
B6F8—	B9 B2 B2	LDA	\$ B2B2,Y	
B6FB—	B2	???		
B6FC—	BF	???		
B6FD—	00	BRK		
B6FE	B6 09	LDX	\$ 09,Y	

地址 B6F5~B6FB 为密码值,现设置为 1968111,可任意修改为你想设置的任何口令,但必须注意输入口令字符的 ASCII 码时要加上 1。(因为加密程序中的 6502 怪指令将会使密码减 1 处理,这是为了防止被启动跟踪而设置的)

4. 取出 DOS3.3 系统盘,插入需加密盘,键入 INIT 命令对磁盘进行格式化。当格式化完成后,就制

成了一张具有口令、防 RESET 和起动跟踪的加密盘。

经过上述处理的系统盘启动后,屏幕的左上角首先出现一个“?”,要求输入口令,当你输入正确的口令,才能启动 DOS,否则系统将封锁 DOS,确保安全,你必须重新启动该磁盘才能再输入口令。

编者的话:我们结合《6502 特殊指令》一文向读者介绍傅剑同志这篇应用文章。有关“怪”指令的操作的准确性质及稳定可靠问题,仍需通过大量的实践证明,欢迎大家提供这方面的材料。

双显示电子钟

宁夏中卫县中学 高二班 万斌

在 1989 年第一期《青少年计算机》杂志上曾发表过一篇《双显示电子钟》。该程序十分冗长,虽便于阅读,却给读者输入带来极大不便。并且该程序所实现的功能并不十分理想。例如:没有设置秒针,并且分针每一分钟才跳动一下。为此,我编了一个“双显示电子钟”程序。为了便于输入,须对程序进行优化、简化的工作。这就要对时、分、秒针在每一时刻的位置进行算法分析。经过精打细算,我确定了算法,并运用大量的逻辑运算来实现。程序中极少运用 IF...THEN 语句。如遇到秒数到零,分数需加 1 时,程序仍运用逻辑运算。

程序运行后会询问:“HOUR MINUTE SECOND

(时、分、秒)”,这时用户输入即时时间,然后按任何一键,程序钟即开始工作。屏幕左上角有一电子表,正中间有一机械钟,时、分、秒针不停地运转。用户可对时钟误差进行调整,方法是对空循环的终值进行调整。

有兴趣的读者,不妨对此程序进行剖析,利用其中的逻辑判断进行其他程序的编制,并可对程序进行修改,使机械和电子表移至高分辨率第一页混合显示。

```
5 PR#3:HGR2;HOME;R(1)=45;R(2)=60;R(3)=75;HCOLOR=3
```

```
10 INPUT "HOUR, MINUTE, SECOND"; H, M, S; HGR2;HOME;HH=H+M/60;MM=M+S/60-.1
```

```
20 PI=3.14159265;R=80;FOR I=0 TO 2*PI STEP PI/6;GOSUB100;HPLLOT X,Y;NEXT P=PI/2;K(1)=HH/3;K(2)=MM/15;K(3)=S/15;GET A$
```

```
30 FOR I=1 TO 3;A(I)=P*K(I)-P-(K(I)>1)*2*PI;B(I)=2*PI-A(I);NEXT C(1)=PI/21600;C(2)=PI/1800;C(3)=PI/30
```

```
40 FOR J=1 TO 3;I=A(J);R=R(J);GOSUB 100;HCOLOR=0;HPLLOT139,95 TO X(J),Y(J);HCOLOR=3;HPLLOT 139,95 TO X,Y;X(J)=X,Y(J)=Y;NEXT MUSIC 100,1
```

```
50 FOR I=1 TO 3;A(I)=A(I)+C(I)-(A(I)+C(I))*2*PI;NEXT
```

```
60 VTAB 1;HTAB 1;PRINT H;";";M;";";S;SPC(3);S=S+1-(S=59)*60;M=M+(S=0)-(M=59 AND S=0)*60;H=H+(M=0 AND S=0)-(H=11 AND S=0 AND M=0)*12
```

```
70 FOR I=1 TO 395;NEXT;GOTO 40
```

```
100 X=139+R*COS(I);Y=95+R*SIN(I)*.99/100;RETURN
```

(上接第 42 页)

四、其它 IC 故障:

现象:1. 开机后,喇叭不响,屏幕无任何反应,指示灯亮。查 A₂、B₂、B₁₃、C₁、C₂、A₁₄。

2. 开机后,喇叭会响,但屏幕没有反应。查 A₂、A₈、A₁₀、B₂、B₁₀、B₁₃、C₂、C₁₁、D₁₁—D₁₄。

3. 开机后,喇叭不响,没有进行 RESET 设定,屏幕出现杂乱的字形。查 A₁₃、B₅—B₈、B₁₁、C₁、C₂、C₁₄、E₁₂、E₁₃、F₁₂—F₁₄、G₁—G₅、G₁₀—G₁₂、C₃—C₁₀、D₃—D₁₀、E₃—E₁₀RAM。

4. 文字正常,但图形和画面故障。查 A₈—A₁₁、B₄、B₅、B₈、B₉、B₁₂、B₁₃、C₃—C₁₀、C₁₂、C₁₄、E₂、E₁₁—E₁₄、F₁₄、G₁。

5. 字体错误或变形。查 A₃、A₅、B₅—B₈。

6. 按键与出现的文字不相符。查 A₃、B₂、B₃、B₁₁、B₁₃。

7. 喇叭不响,其它正常。查 F₁₃、J₁₃、Q₄、C₁₁、C₁₂电

容及喇叭连线与插头。

8. 光标不正常。查 A₃、B₂、B₃、B₁₁、B₁₃。

9. 磁带机存取信息故障。查录音磁带机的输出输入与主机连接是否正确,磁带机的音量控制是否适当,板音磁头与原录音头不正。F₁₃、H₁₄、J₁₃、K₁₃。

10. 不按键时,会出现文字。或按一次键会出现两个以上的字符,键盘工作不正常。这种情况可查 B₆、B₇、F₁₃、A₁₂、B₁₀、C₁₁。另外键盘本身的故障以及排线中容易受外界干扰也会造成以上所述的故障现象。

11. 无彩色信号。查主机板上晶振边的 50P 微调电容。调整彩电的色饱和钮以及 B₁₂、B₁₃。

按以上所列的现象,维护人员不必掌握较深的维修技术,采用替代法或比较法以及常规的电子测量仪器,就可把损坏的主机修复。

苹果机汉字造型表自动生成程序

济南西站

刘士明

为解决苹果机在高解析度状态无法显示汉字及字符的问题,我编制了这个程序,可将您在屏幕上所造的汉字及其他一切图形、字符自动转换为向量造型表,并可存入磁盘文件备用。使用时,只需将该造型表读入内存,在高解析度状态下,用 DRAW 命令调用您所造的汉字,并且能够以任意大小,角度显示。

程序执行时,首先询问您所造汉字的个数,然后在屏幕左侧显示一个 15×16 的点阵,左上角闪烁的小方框为光标。使用下列功能键,可在点阵中移动光标和置光点,帮您排布出要造的汉字点阵字形:

按 I、J、K、M 键移动光标;

按 S、X、Z、Y 键上下左右置光点。

为造出优美的字形,在点阵右侧开辟了一个反向显示的“小窗口”;当您在点阵中造字的同时,按 D 键,它同步显示您已排的字形,使您知道该在点阵中何处置光点,才能造出最理想的汉字。

在程序中,还定义了下列功能键,帮助您制作汉字造型表。当所排点阵不满意时,按 H 键可随意清除点阵中的任何光点;按 Q 键清除所有光点;按 E 键结束当前汉字点阵的输入,转入下一个汉字。直至最后一个汉字造型生成,则询问是否保存该造型表,及程序是否继续,可按提示进行选择。若选择保存该造型表,则提示您输入一个文件名后,将所造的汉字造型存入磁盘。

对于生成的汉字造型,使用时,按下列步骤操作:

<1>将造型表文件调入内存:BLOAD 文件名

<2>使计算机进入高解析度状态:

HGR;POKE 232,0;POKE 233,96

<3>定义颜色、旋转角度和放大倍数。

<4>显示造型表中的汉字:

DRAW N AT X,Y

第<4>步显示表中第 N 个汉字于屏幕坐标(X,Y)处,颜色、旋转角和放大倍数由第<3>步定义。

10 TEXT,HOME

20 BL\$ = CHR\$(7)

30 VTAB 22; INPUT "ENTER SHAPE NUMBER,";N; IF N(1 OR N)255 THEN 30

50 ST = 24576; BYTE = 0; ADDR = ST + 2 * N + 2; S = ADDR

60 POKE ST,N; POKE ST+1,0; POKE ST+2,N*2+2; POKE ST+3,0

70 BYTE = 0; ADDR = S; A = 0; P = 1

80 POKE 768,1; POKE 769,0; POKE 770,4; POKE 771,0; POKE 772,44; POKE 773,62; POKE 774,0

90 POKE 232,0; POKE 233,3

100 HGR; POKE -16302,0; HCOLOR = 3; SCALE = 5;

ROT=64

110 FOR I = 15 TO 155 STEP 10

120 FOR I2 = 15 TO 165 STEP 10

130 HPLLOT I,I2

140 NEXT I2,I

150 X2 = 1; Y2 = 1; I = 1

160 T = PEEK(-16384); POKE -16368,0

170 X3 = 10 * (X2-1) + 15; Y3 = 10 * (Y2-1) + 15; XDRAW 1 AT X3,Y3; XDRAW 1 AT X3,Y3

175 IF T(127 THEN 160

180 IF T = 217 AND X2(15 THEN X2 = X2+1; POKE ADDR,67; POKE ADDR+1,1; POKE ADDR+2,6; ADDR = ADDR+3; GOTO 380

190 IF T = 218 AND X2 > 1 THEN X2 = X2-1; POKE ADDR,67; POKE ADDR+1,3; POKE ADDR+2,6; ADDR = ADDR+3; GOTO 380

200 IF T = 202 AND X2 > 1 THEN X2 = X2-1; POKE ADDR,3; GOTO 370

205 IF T = 196 THEN HCOLOR = 3; POKE ADDR,0; FOR I2 = 203 TO 223; HPLLOT I2,50 TO I2,75; NEXT; HCOLOR = 0; POKE 233,96; SCALE = 1; DRAW P AT 205,55; POKE 233,3; SCALE = 5; HCOLOR = 3; GOTO 160

210 IF T = 203 AND X2(15 THEN X2 = X2+1; POKE ADDR,1; GOTO 370

220 IF T = 205 AND Y2(16 THEN Y2 = Y2+1; POKE ADDR,2; GOTO 370

230 IF T = 197 THEN 440

240 IF T = 211 AND Y2 > 1 THEN Y2 = Y2-1; POKE ADDR,67; POKE ADDR+1,67; POKE ADDR+2,6; ADDR = ADDR+3; GOTO 380

250 IF T = 216 AND Y2(16 THEN Y2 = Y2+1; POKE ADDR,67; POKE ADDR+1,2; POKE ADDR+2,6; ADDR = ADDR+3; GOTO 380

260 IF T = 201 AND Y2 > 1 THEN Y2 = Y2-1; POKE ADDR,67; GOTO 370

270 IF T = 209 THEN 70

280 IF T(> 200 THEN GOTO 160

290 HCOLOR = 0; GOSUB 390; HCOLOR = 3; HPLLOT X1,Y1; ADDR = ADDR-1; M1 = PEEK(ADDR); M2 = PEEK(ADDR-1); M3 = PEEK(ADDR-2)

300 X2 = X2 + (((M3=67) AND (M2=3) AND (M1=6) OR (M1=3)) - (((M3=67) AND (M2=1) AND (M1=6) OR (M1=1)))

310 Y2 = Y2 + (((M3=67) AND (M2=67) AND (M1=6) OR (M1=67)) - (((M3=67) AND (M2=2) AND (M1=6) OR (M1=2)))

320 IF Y2 < 1 THEN Y2 = 1

330 IF X2 < 1 THEN X2 = 1

350 IF M1 < 6 THEN 160


```

360 ADDR = ADDR-2; GOTO 160
370 ADDR = ADDR+1; GOTO 160
380 HCOLOR=3; GOSUB 390; GOTO 160
390 X1 = 10 * (X2-1) + 15; Y1 = (Y2-1) * 10 + 15
400 SCALE=2; A=1
410 FOR II=1 TO 64; ROT=II; DRAW 1 AT X1,Y1;
NEXT
420 SCALE=5; RETURN
440 POKE ADDR,0; ADDR=ADDR+1; HCOLOR=3
460 BYTE=ADDR-ST; S=ADDR
480 P=P+1; IF P=N THEN PRINT BL$; GOTO 520
490 A = INT (BYTE/256); B = BYTE-A * 256; POKE
ST+2 * P,B; POKE ST+2 * P+1,A
500 BYTE = 0; GOTO 80

```

```

520 HOME; POKE -16301,0; VTAB 22; PRINT "BSAVE
TABLE TO DISK(Y/N)"; GET Y$; IF Y$ <> "Y" AND Y
$ <> "N" THEN 520
530 IF Y$ = "N" THEN 570
540 HOME; VTAB 22; INPUT "ENTER TABLE
NAME:"; NA$; IF NA$ = "" THEN 540
550 LE = ADDR-ST
560 PRINT CHR$(4); "BSAVE" NA$ ",A"; ST; ",
L"; LE
570 HOME; VTAB 22; PRINT "END PROGRAM (Y/
N)?"; GET Y$; IF Y$ = "Y" THEN HOME; TEXT; END
580 IF Y$ <> "Y" AND Y$ <> "N" THEN 570
590 HOME; GOTO 30

```

高效率的排序方法

陈庆祥

一、问题的提出。

在大多数实际问题中,排序的对象并非是一个单纯的数组,而是具有密切联系的若干数组的集合。例如某中学一次英语竞赛的有关统计资料如下表:

考号	姓名	性别	年龄	班级	笔试	口试	总分
0001	郑丽	女	15	102	92	77	
0002	肖剑文	男	16	201	84	79	

上表中总分按笔试的 60% 与口试的 40% 之和计算。现要求能按任一数据关键项排序。以下是按总分由高到低排序(设有 5 个学生):

```

10 INPUT "N="; N; REM 学生竞赛记录分类程序 1.
20 DIM A%(N), B$(N), C$(N), D%(N), E%(N), F
(N), G(N), S(N)
25 FOR I = 1 TO N; A%(I) = I
30 READ B$(I), C$(I), D%(I), E%(I), F(I), G(I)
40 S(I) = INT (.6 * F(I) + .4 * G(I) + .5); NEXT I
50 FOR I = 1 TO N-1; FOR J=I+1 TO N
70 IF S(I) < S(J) THEN 120
80 T=A%(I); A%(I) = A%(J); A%(J) = T
85 T$=B$(I); B$(I) = B$(J); B$(J) = T$
90 T$ = C$(I); C$(I) = C$(J); C$(J) = T$
95 T = D%(I); D%(I) = D%(J); D%(J) = T
100 T = E%(I); E%(I) = E%(J); E%(J) = T
105 T = F(I); F(I) = F(J); F(J) = T
110 T = G(I); G(I) = G(J); G(J) = T
115 T = S(I); S(I) = S(J); S(J) = T
120 NEXT J; NEXT I
130 PRINT "考号 姓名 性别 年龄 班级 笔试
口试 总分 名次"
140 FOR I = 1 TO N
150 PRINT RIGHT$("000" + STR$(A%(I)), 4) " " B
$(I) " ";
160 PRINT C$(I) " " D$(I) " " E$(I) " "

```

```

F(I) " " G(I) " " S(I) " " I
170 NEXT I; END
1000 DATA "郑丽", "女", 15, 102, 92, 77
1005 DATA "肖剑文", "男", 16, 201, 84, 79
1010 DATA "唐学英", "女", 14, 101, 94, 86
1015 DATA "陈大勇", "男", 16, 303, 95, 90
1020 DATA "李涛", "男", 17, 301, 80, 85
]RUN
N=5

```

考号	姓名	性别	年龄	班级	笔试	口试	总分	名次
0004	陈大勇	男	16	303	95	90	93	1
0003	唐学英	女	14	101	94	86	91	2
0001	郑丽	女	15	102	92	77	86	3
0005	李涛	男	17	301	80	85	82	4
0002	肖剑文	男	16	201	84	79	82	5

程序 1 排序的效率很低, 其原因是所有数据项必须随总分的交换而交换, 其次, 程序 1 还缺乏处理并列名次的能力, 比如同为 82 分的两名学生名次却不相同。

二、利用下标嵌套进行索引排序

减少数据交换量是提高排序效率的关键。利用下标嵌套的方法进行索引排序(以记录号即考号为索引项), 可以避免记录号以外的任何数据交换, 从而简化处理过程并大大提高排序的效率, 程序如下:

```

10 INPUT "N="; N; REM 学生竞赛记录分类程序 2.
20 DIM A%(N), B$(N), C$(N), D%(N), E%(N), F
(N), G(N), S(N), H%(N)
25 FOR I = 1 TO N; A%(I) = I
30 READ B$(I), C$(I), D%(I), E%(I), F(I), G(I)
40 S(I) = INT (.6 * F(I) + .4 * G(I) + .5); NEXT I
50 FOR I = 1 TO N-1; L = I
60 FOR J = I+1 TO N
70 IF S(A%(L)) < S(A%(J)) THEN L = J

```

```

80 NEXT T:A%(I):A%(I)=A%(L):A%(L)=T
90 GOSUB 300:NEXT:GOSUB 300
100 PRINT TAB(10)"表 一:(按总分名次打印)":PRINT
110 PRINT "考号 姓名 性别 年龄 班级 笔试 口试 总分 名次"
120 FOR I=1 TO N:M=A%(I):P=H%(M):GOSUB 200:NEXT:PRINT
130 PRINT TAB(10)"表 二:(按原来顺序打印)":PRINT
140 "考号 姓名 性别 年龄 班级 笔试 口试 总分 名次"
150 FOR I=1 TO N:M=I:P=H%(I):GOSUB 200:NEXT:PRINT
160 END
200 REM 打印各条记录
210 PRINT RIGHT$("000"+STR$(M),4)" "B$(M)" ";
220 PRINT C$(M)" "D$(M)" "E$(M)" "F(M)" "G(M)" ";
230 PRINT S(M)SPC(6-LEN(STR$(S(M))))P:RETURN
300 REM 处理并记录名次
310 K=K+NOT(S(A%(I))=(S(A%(I-1))))
320 H%(A%(I))=K:RETURN
1000 DATA "郑丽","女",15,102,92,77
1005 DATA "肖剑文","男",16,201,84,79
1010 DATA "唐学英","女",14,101,94,86
1015 DATA "陈大勇","男",16,303,95,90
1020 DATA "李涛","男",17,301,80,85
]RUN
N=5

```

表 一:(按总分名次打印)

考号	姓名	性别	年龄	班级	笔试	口试	总分
0004	陈大勇	男	16	303	95	90	93
0003	唐学英	女	14	101	94	86	91
0001	郑丽	女	15	102	92	77	86
0005	李涛	男	17	301	80	85	82
0002	肖剑文	男	16	201	84	79	82

表 二:(按原来顺序打印)

考号	姓名	性别	年龄	班级	笔试	口试	总分
0001	郑丽	女	15	102	92	77	86
0002	肖剑文	男	16	201	84	79	82
0003	唐学英	女	14	101	94	86	91
0004	陈大勇	男	16	303	95	90	93
0005	李涛	男	17	301	80	85	82

程序 2 根据各记录号为自然数的特点,在比较总分时,仅交换各存贮记录号的下标变量 A%(L)与 A%(J)的值(第 70、80 程序行),其它数组一律不动,在数组 A%中记载了排序结果。这个程序比程序 1 的数据交换量减少了近 90%,效率高出好几倍,同时,不仅解决了并列名次问题,还可以两种不同方式输出排序结果,以满足不同需求。

三、高效率的映射索引排序方法

索引排序能大大减少排序时的数据交换,提高排序效率。但是,绝大多数的排序方法都离不开关键项数据的反复比较过程,因此当记录条数较多时仍然要花费较多的时间。利用映射索引排序则可以同时避免数据交换和比较这两个过程,因而具有极高的排序效率。

利用映射索引排序的程序如下:

```

10 INPUT "N=";N:REM 学生竞赛记录分类程序 3
20 DIM M(100)
30 FOR I=1 TO N:READ A$,B$,C%,D%,X1,X2
40 S=INT(.6*X1+.4*X2+0.5)
50 M(S)=M(S)+1:NEXT
60 FOR I=100 TO 0 STEP -1
70 K=K+SGN(M(I)):M(I)=K
80 NEXT:RESTORE
90 PRINT "考号 姓名 性别 年龄 班级 笔试 口试 总分 名次"
100 FOR I=1 TO N:READ A$,B$,C%,D%,X1,X2
105 S=INT(.6*X1+.4*X2+0.5)
110 PRINT RIGHT$("000"+STR$(I),4)" ";
115 PRINT A$" "B$" "C$" "D$" "X1" "X2" "S" "M(S)
120 NEXT:END
1000 DATA "郑丽","女",15,102,92,77
1005 DATA "肖剑文","男",16,201,84,79
1010 DATA "唐学英","女",14,101,94,86
1015 DATA "陈大勇","男",16,303,95,90
1020 DATA "李涛","男",17,301,80,85
]RUN
N=5

```

考号	姓名	性别	年龄	班级	笔试	口试	总分
0001	郑丽	女	15	102	92	77	86
0002	肖剑文	男	16	201	84	79	82
0003	唐学英	女	14	101	94	86	91
0004	陈大勇	男	16	303	95	90	93
0005	李涛	男	17	301	80	85	82

程序 3 仅用单循环就完成了排序过程,无需任何数据交换和比较,其效率高于其他任何排序方法。程序 3 的主要特点是数组 M 的使用。程序的第 50 行中,以总分 S 为索引项,用数组 M 存放各种分数的个数;而在第 70 行中又用它存放各个分数的名次,同时也解决了并列名次的问题。这种利用同一数组先后存放两类相关数据的技巧,充分说明了灵活地使用下标去管理数组的意义。这种排序方法在用 DATA 语句提供数据时,只需定义一个固定容量的数组 M(100)。这样当 N 较大时,可以节省大量的存储空间;而且以简单变量取代数组还可进一步提高处理效率(对 1000 条记录排序所用时间不到 60 秒)。当作为索引的分数含有一位小数时,可扩大 10 倍使之成为整数,再将数组容量扩大 10 倍。

演示抛物面的光学效应

成都市东城区教师进修校 王晓林

抛物面卫星信号接收天线、太阳灶反光罩以及汽车灯反光罩等都应用了抛物面的一个重要物理原理,即它能把一组平行光线反射汇聚于焦点,又能把焦点处的点光源散射光反射成一组平行光线。这是中学物理和数学中要介绍的重要内容。但是,在上课时很难直观演示这两个可逆的过程。我编了下面的程序,不仅可以在屏幕上直观地看到这两个过程的“慢镜头”,而且还可看到,一组同步推进的平行光线,虽然被抛物面反射的先后不同,但却是同时汇聚于焦点。这一现象在教材中往往被忽视。但是,正因为有这一效果,才能确保电波所携带的信息汇聚于焦点时不会失真。此外,由于参数P是由用户自由输入的,因此可以看到各种开口大小的抛物面的反射和汇聚效应是一致的。

本程序显示的实际上仅是抛物面的轴截面上的现象。程序的关键技巧在于能保证各条光线不论是平行状态还是散射状态下都要能“等速”同步推进。而且由于各条线与抛物面相遇是不同时的,因而还要准确处理每条光线何时开始反射及反射后的方向,而且要保证反射和未反射的光线仍能同步推进,其原理如下:

各条光线上每次只显示一个推进点,然后利用直线的参数方程($x=x_0+lc\cos t$, $y=y_0+ls\sin t$)计算下一个点,并判断该点是否已达到抛物面,若已达到,则计算光线反射后的前进方向,再算下一个点的坐标。若未达到则显示该点,如此循环,直至全光路显示完毕。

程序运行时,输入参数P($y^2=2px$ 中的P)。然后依次显示两次可逆过程。显示中可由空格键控制。显示完后,敲ESC键可继续。

本程序是在CEC—I中文状态下通过。若要在APPLE机上实现,只需去掉所有含汉字的程序行(8900,9080,9019,9005,9006,9157)。

```
8900 PR# 3; PRINT "HGR2
9000 DIM B0(4),Y1(4),X1(4),Y(4),X(4),A(4),C
      (4)
9003 HOME,VTAB 2;PRINT "Y^2=2PX"
9005 VTAB 4; PRINT "抛物线物理意义"
9006 PRINT "对声光的反射和汇聚作用"
9010 VTAB 6; PRINT "0<P<10"
9015 INPUT "P=";P;P0=P/2; IF P<1 THEN 9010
9016 IF P>10 THEN 9010
9019 A$="平行光汇";B$="聚至焦点"
9020 F=0;M0=277;N0=190;M=M0/3;N=N0/
      2
9025 K=M0/8;XL=2*M/K;L0=XL+P0
9030 GOSUB 10000
```

```
9050 FOR J=1 TO 4;X(J)=-1;Y(J)=0
9055 Y=J*N/K/5;Y1(J)=Y;X=Y*Y/P0;B0(J)
      =XL-X;X1(J)=XL
9060 R=X+P0;A(J)=-Y/R;C(J)=(P0-X)/R
9065 NEXT J
9070 FOR Y=0 TO N/K STEP .03
9073 X=Y*Y/P0; IF X>XL THEN 9076
9075 HPLLOT M+K*X,N-K*Y;HPLLOT M+K*X,
      N+K*Y
9076 NEXT Y
9080 VTAB 2;PRINT A$;PRINT B$
9090 FOR I=0 TO L0*K/2
9092 FOR J=1 TO 4
9095 IF I/K<(B0(J)/2 THEN XS=X(J);YS=Y(J);
      GOTO 9105
9100 YS=A(J);XS=C(J)
9105 X2=X1(J)+2*XS/K;Y2=Y1(J)+2*YS/
      K;HPLLOT M+K*X2,N-K*Y2;HPLLOT M+K
      *X2,N+K*Y2
9107 V=PEEK(49152); IFV=160 THEN 9107
9110 X1(J)=X2;Y1(J)=Y2
9112 NEXT J
9115 NEXT I;F0=F0+1
9130 IF F0=2 THEN 9160
9140 FOR J=1 TO 4;B0(J)=L0-B0(J)
9145 X(J)=-C(J);Y(J)=-A(J)
9150 A(J)=0;C(J)=1;X1(J)=P0;Y1(J)=0
9155 NEXT J;GOSUB 10000
9157 A$="焦点光源";B$="变为平行光"
9158 GOTO 9070
9160 V=PEEK(49152); IFV=141 THEN 9003
9165 GOTO 9160
10000 HGR2;HCOLOR=3
10002 HPLLOT 1,N TO 279,N TO 273,N-2 TO 273,N
      +2 TO 279,N
10004 HPLLOT M,190 TO M,1 TO M+2,5 TO M-2,5
      TO M,1
10006 IF K<3 THEN 10020
10008 F=M-K*INT(M/K);F1=N-K*INT
      (N/K)
10010 FOR I=F TO 279 STEP K;HPLLOT I,N-1 TO
      I,N+2;NEXT
10012 FOR I=F1 TO 190 STEP K;HPLLOT M-1,I TO
      M+2,I;NEXT
10020 RETURN
```

LASER QZT/C 应用经验

北师大附属实验中学 张泽翔

QZT/C 汉化高显系统经《电子与电脑》杂志于八八年九月首先介绍给读者后,经一年多使用,摸索了几点经验,现介绍给大家。

一、用外存保存变量

主机的 CSAVE 只能把程序录到磁带上,而程序的运行结果即变量值的保存,一直是一个难题。QZ 系统可胜任此项工作,见示例程序一、二。程序一先给 A、B(9)、C% 等变量运算赋值。40 行利用 QZ 特有的 NAME 语句。NAME 既可转录 BASIC 程序,也可转录二进制文件,而且可以用中文作文件名。40 行作用是取变量区首尾地址把整个变量区作二进制文件转录,并给该文件起了“变量库”这样一个直观明确的中文名字。程序二是调回“变量库”的例子。由于程序一、二长度不同,所以利用 BLOAD 的“变址调入”功能以使变量区调入后准确定位。应注意程序二中数组需定义(但不必赋值)而简单变量应先赋值,如统一赋为 0。调入“变量库”后 A、C% 等已不是 0 而是程序一的运行结果。

二、可显示的倒时钟

QZ 的时钟功能有趣又有用,它不仅不受干扰显示在屏幕右上角,而且还有一个专用保留字 TIMS\$ 可随时在 BASIC 程序中调用显示时钟。QZ 还有一个“倒时钟”,TIME\$ 调不出它,但有时又希望在程序中,如棋类程序作限时记秒用。这只要在程序开头加上程序三的语句就可随时用字符串变量 TR\$ 调出倒时钟,其中使用了变量地址函数 VARPTR。因倒时钟长五位,所以 10 行给 TR\$ 赋初值时也是五位。有趣的是如 10 行字符串长 14 位,如 TR\$ = "ABCDEFGHIJKLMN" 则以后用 TR\$ 可同时调出“倒时钟”和“顺时钟”。TR\$ 是串变量,与保留字 TIME\$ 性质不同,只能用在程序中,修改删除程序都会给 TR\$ 赋空串值。

三、高显彩色绘图

QZ 的 V8 软件增加了高显彩色功能。SET、RESET、POINT 都和 MODE(1)用法一致。用 GET 和 PUT 可把一个区域的彩色图形方便快速地移到另一区。也可用 LINE 画彩色线,如

```
LINE(X,0)-(X,191)
```

其规律是当坐标 X 为偶数时画蓝色线,X 为奇数时画黄线,用相邻偶奇数执行两次 LINE 则为红线。在高显彩色屏幕上不显示 ASCII 字符,但仍可打入任何命令,

只要按键正确就会被执行。如不执行还可以打入 SC1 或 SC0 命令转到高显文字状态,你前次打入的命令会显示在屏幕上以便检查错在什么地方。

四、中文词组加锁

QZC 有全拼音、区位、偏旁部首和词组四种汉字输入方式。在词组方式下用户可根据需要预先建立常用词组库。只需击几个英文键,就可快速输入几个到几十个汉字。QZ 词组库格式别致,原来就是普通 BASIC 程序,人人都是“不学自会”。但 BASIC 词库容易被 NEW 掉。为了保护词库,可以用 QZ 的加锁命令 LOCK,如

```
60000 LOCK 60000:END
```

这样 60000 行以后的词库就被保护了,加锁情况下要调入新程序可以用拼接命令。

```
NEW;MERGEL“文件名”(可省)
```

这样词库拼在新调入程序后面不会丢失。万一在未加锁情况下使用 NEW 命令,及时发现后执行 KEY 或 CALL&H4E19 或 MONE4E19,都能救回词组库。

程序一

```
10 DIM B(20);B(9)=3.45
20 A=10/3;C%=-10
30 PRINT“请启动磁带机,击空格键开始转录”;CALL
&H3B2C
40 NAME“变量库”PEEK(&H78F9)+256*PEEK
(&H78FA),PEEK(&H78FD)+256*PEEK(&H78FE),
SCREEN
```

程序二

```
10 DIM B(20);B(9)=0
20 A=0;C%=0
30 BLOAD PEEK(&H78F9)+256*PEEK(&H78FA)
“变量库”;PRINT
40 PRINT A;C%;B(9)
```

```
3.33333-10 3.45
```

程序三

```
10 TR$="12345"
20 A%=VARPTR(TR$)
30 POKE A%+1,&H97;POKE A%+2,&H79
40 PRINT TR$
```


1. 基本数据类型

Turbo C 中使用的基本数据类型有 5 种(表 2.1):

表 2.1 Turbo C 基本数据类型

数据类型	长度	数值范围
int(整型)	16 位	-32768~32767
float(单精度浮点型)	32 位	3.4E-38~3.4E+38
double(双精度浮点型)	64 位	1.7E-308~1.7E+308
char(字符型)	8 位	-128~127
void(无值型)	0 位	无值

2. 修饰符

数据类型除 void 外,可依用途作如下修饰说明:

signed 有符号 unsigned 无符号
long 长整型 short 短型

这四个修饰符还可按如下组合使用:

signed long signed short
unsigned long unsigned short

与 char、int、double 组合可得,如下六种类型:

signed short int	unsigned short int
signed long int	unsigned long int
signed double	unsigned double
signed char	unsigned char

无符号字符和有符号字符的取值范围如下:

表 2.2 字符型数据的分类

类 型	取值范围
char	-128~127
unsigned char	0~255
signed char	-128~127

3. 常量

常量是指其值固定不变的量,有如下五种:

(1) **字符型常量** 用单引号引起来的单个字符,如 'a', ' ' 等。

(2) **整型常量** 可用 10、8 和 16 进制表示。用 8 和 16 进制表示时,在常数前要有前导词 0 和 0X,如:

123...10 进制数

0123...8 进制数

0X123...16 进制数

长整型常数的后面要加上 l 或 L,如 123L。

浮点型常量 它有 2 种表示方法,即:

① 小数表示法 如 1.23;

② 科学表示法,如 1.23e-3。

(4) **字符串常量** 用双引号引起来的字符序列,如

"abc"。在存储器内部表为字符序列 abc\0.\0(Null)为结尾符。可见,字符与字符串是不同的。

(5) **控制字符常量** ASCII 码值在 0X00~0X1F 之间的字符,即控制用字符,需采用反斜杠与特定字符组合表示,才能输入。控制字符如表 2.3 所示。

表 2.3 控制字符表

符号	ASCII 码	功能
\0	0x00	Null
\a	0x07	响铃
\b	0x08	退格
\t	0x09	水平制表
\f	0x0c	走纸
\n	0x0a	回车换行
\v	0x0b	垂直制表
\r	0x0d	回车
\\	0x5c	反斜杠
\'	0x2c	单引号
\"	0x22	双引号
\?	0x3f	问号
\DDD	0DDD	DDD 为 3 位及其以下 8 进制数
\xHHH	0xHHH	HHH 为 3 位及其以下 16 进制数

4. 变量及其初始化

所谓变量是其值可变化的量。在程序中使用变量时,必须首先说明其数据类型,其语句格式如下:

数据类型 变量表;

注意,C 语句末尾必须有一个分号(;),例如:

```
char a, b;
int i, j, k;
```

为使用这些变量,还必须先给出其值的大小,这叫变量初始化。例如

```
int i=10, j=20, k;
K=i+j;
```

变量名的组成规则如下:

① 以字母或下划线(_)开始。

② 能使用的字符有 a~z, A~Z, 0~9, _ 和 \$。

③ 有效长度为 32 个字符。

④ 大小写是有区别的。C 语言是以小写字母为主,书写的小写化是 C 程序的一大特点。

⑤ 不能使用 Turbo C 的关键字。

5. 函数

结构的模块化,是 C 程序的又一大特点。函数是程序的基本模块,相当于子程序,定义形式如下:

函数类型说明 函数名(参数表)

参数类型说明

```
{
    说明语句
    执行语句
}
```

(1) **函数类型** 用前面所介绍的关于基本数据类型的关键字来说明,指出函数值的数据类型。

(2) **函数名** 标识符后紧跟一对圆括号构成。

(3) **函数体** 函数体的界线符是一对花括号,其内容大致为两大类:说明语句和执行语句。

函数的定义中,必不可少的部分是:

函数名()

```
{
}
```

这是最小 C 程序。由于它的函数体没有实在内容,故什么也不执行。程序员常用它在程序中事先占据一个位置,好将来扩充一个有实际功能的函数。

6. 表达式

(1) **表达式的组成** 表达式是用运算符把操作数连接构成的式子。操作数可以是常量、变量和函数。

(2) **表达式的解** 指表达式中的操作数按运算符优先级进行运算,最终得到的表达式的值。

(3) **表达式的副作用** 在表达式中使用赋值运算符和递增(++)、递减(--)运算符时,将把值赋给某个变量,这就叫表达式的副作用(side effects)。

(4) **表达式语句** 表达式的后面加上一个分号(;)就构成一个表达式语句。C 程序主要是由表达式语句构成的,语句的表达式化是 C 程序的又一大特点。

(5) **简单语句和复合语句** C 语言中的分号是语句的结尾符。含一个分号的语句叫简单语句,如:

```
K=i+j;
```

多个简单语句借助一对花括号可组成复合语句,如:

```
{ i=10;
```

```
  j=20;
```

```
  K=i+j;
```

```
}
```

7. 程序、文件、函数三者的关系

C 程序的编译单位是文件。一个文件由一个或若干函数组成。一个可执行文件必须有一个名为 main() 的函数,叫主函数。不管该函数放在什么地方,程序总是从它开始执行,到它执行完毕,整个程序也就执行完毕。其它函数由 main() 或别的函数调用。

下面介绍一个简单的程序,以使读者了解 C 程序的风格,尽快掌握 C 程序的设计方法。

[练习 2.1] A) type lil.c

```
main( )
```

```
{ int i=10,j=20,k;
```

```
  k=i+j;
```

```
  printf("k=%d\n",k);
```

```
}
```

执行: A>lil

```
K=30
```

(1) 该程序只含一个文件,文件名叫 lil.c;注意,C

源文件的扩展名必须是 C。该文件只含有一个函数,这就是主函数 main()。

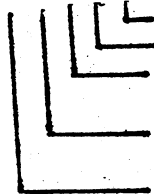
(2) 2 行为说明语句,说明 i、j、k 为整型变量;同时对 i 和 j 进行了初始化。

(3) 3 行是计算 i+j 的值的执行语句。该语句就是一个表达式语句。符号=是赋值运算符,表示把 i+j 的值赋给变量 K,因此,该语句也叫赋值语句。

(4) 4 行是 C 语言中输出方式——调用输出函数。C 语言的输入输出功能是靠调用 C 编译系统的库函数实现的。输入输出的函数化是 C 程序的又一大特点。函数 printf() 是按格式输出函数,其功能是按照给定格式显示(或打印)表达式的值。打印格式如下:

转换控制字符串 自变量 K

```
printf("K %d\n",k);
```



自变量,其内容为要显示的值

回车换行

转换控制字符,表示按 10 进制显示所对应的 K 的值

转换控制开始符,指定要显示的值的位置

原样显示的部分

printf() 函数的转换控制字符如表 2.4 所示。

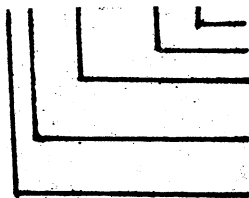
表 2.4 printf() 转换控制字符

转换控制	所指定的打印格式
%d	按 10 进制输出
%o	按无符号 8 进制输出
%x	按无符号 16 进制输出
%u	按无符号 10 进制输出
%c	按字符输出
%s	按字符串输出
%e	以 men 形式输出
%f	以小数形式输出
%g	采用 %e 和 %f 中较短的输出
%p	打印地址

在 % 与转换控制字符之间还可加上宽度说明。

无宽度说明时,若输出字符,则按字符的实际个数输出;若输出浮点数则按 6 位小数处理。

```
- x x x . x x x x
```



转换控制字符

小数点后面的位数

整数部分,表示输出的最小位数(小数占一位)

有负号靠左边输出,否则靠右边输出

自变量值存放开始位置

8. 按格式输入函数 scanf()

(1) 功能和用法 scanf() 也是一个系统库函数,

其功能是从键盘输入一个字符,用法是

scanf (转换控制字符串,变量 1 指针,变量 2 指针,……);

[练习 2.2]

A>type li2.c

main()

{ int i,j

scanf("%d,%d",&i,&j);

printf("input: %d,%d\n",i,j);

}

执行: A>li2

246,48

input: 246,48

该练习中 scanf()各参数的名称和功能如下:

第 1 参数 第 2 参数 第 3 参数

(转换控制字符串) (变量指针) (变量指针)

scanf(" %d, %d",&i,&j);



变量 j 的指针

变量 i 的指针

转换控制字符

表示转换控制开始

数据分隔符,可为逗号、空格、制表符、换行符

(2)转换控制字符

d——把输入看作是 10 进制整型数;

f——把输入看作是浮点数;

e——同 f;

o——把输入看作是 8 进制数;

x——把输入看作是 16 进制数;

c——把输入看作是一个字符;

s——把输入看作是字符串;

h——把输入看作是短整型数;

i——把输入看作是 10 进制整型数;

p——把输入看作是指针;

n——输入视为整型数,其值为读入字符个数。

(3)scanf()函数的一些具体用法

①转换控制字符前的数字,用于指定数据位数;

②转换控制字符前面的 L,用来表示读双精度浮点数或长整型数据;

③转换控制开始符 % 后面的 * 用来禁止赋值;

④如若转换控制字符串中有非转换控制字符,则输入数据时要在相应的部分输入与此相同的字符;

9. 函数调用形式

(1)C 程序结构 C 程序总的看,是由函数定义组成的。注意,函数不能嵌套定义,但能嵌套调用。

(2)函数的调用关系 分为内部调用和外部调用。内部调用指一个文件内函数间的调用。外部调用指一个文件的函数调用另一个文件的函数。

(3)函数调用形式 最基本的是如下三种形式:

①函数语句调用 把被用函数的函数名直接写出,并以实参替换形参,圆括号后加以分号。被调用函

数将作为一个独立的语句出现,如:

[练习 2.3]

A>type li3.c

main()

{ printf("I'm in main.\n");

aia();

}

aia()

{printf("Now I'm in aia.\n");}

执行: A>li3

I'm in main.

Now I'm in aia..

②函数表达式调用 这种调用的特点是被调用函数出现在调用函数的表达式中。被调用函数要有返回值。在调用函数中,要说明被调用函数的数据类型,若其数据为整型,则无须进行类型说明。

[练习 2.4]

A>type li4.c

main()

{ float i=1.5;

double square();

i=square(i);

printf("%f\n",i);

}

double square(x)

double x;

{return(x*x);}

执行: A>li4

2.250000

③函数参数调用 被调用函数以函数参数形式出现,其调用条件和注意事项同函数表达式调用。

[练习 2.5]

A>type li5.c

int i=2,j=5;

main()

{printf("%d,%d,%d\n",i,sum(),sub());}

sum()

{ int s;

s=i+j;

return(s);

}

sub()

{ int e;

e=i-j;

return(e);

}

执行: A>li5

2,7,-3

10 数据的存储类别

在 C 语言中,变量除了有类型之分,还有四种存储类别,即:自动存储变量、静态存储变量、外部存储变量和寄存器存储变量。这些存储类别用来说明诸如变量的作用域、变量的生存期等存储性质。变量的作用域是指能访问该变量的场所。变量的生存期是指变量值

所能保留的时间。

(1) 自动存储变量

①性质 是指其值在某一范围内得以保留,这一范围以外则消失的变量,它有如下性质:

- 作用域的局部性 其作用域为该变量定义所在的模块(一对花括号围起来的部分)内或函数内。

- 生存期的暂时性 其生存期为该变量定义所在函数或模块的执行周期,即一旦进入该函数或模块,C就自动地为该变量建立存储区,而一旦退出该函数或模块,C就自动地收回此存储区。

- 未初始化的变量其值不确定,是无意义的。

②定义 定义自动存储变量用关键字 auto,如

```
auto int i,j;
```

[练习 2.6]

```
A> type li6.c
```

```
main()
```

```
{int i=100,k=80;
```

```
printf("i=%d\n",i);
```

```
{int i=200;
```

```
printf("    i=%d k=%d\n",i,k++);
```

```
}
```

```
printf("i=%d k=%d\n",i,k);
```

```
}
```

```
执行:A> li6
```

```
i=100
```

```
    i=200 k=80
```

```
i=100 k=81
```

(2) 静态存储变量

①定义 可按如下形式加以定义及初始化:

```
static 数据类型 变量名=初值;
```

②种类 有局部和全局两种静态存储变量。

- 静态局部存储变量 仅能在所定义的模块内使用。与 auto 变量不同,它可保留原值不变。如函数:

```
count _up()  
{static int number=0;  
  number+=25;  
  return(number);  
}
```

其返回值为 25;以后每调用一次,其值就增加 25。这是由于在编译时就给静态存储变量 number 分配了存储空间,而以后该变量就永久地存在下去的缘故。

但变量 number 不能在它所在的函数外进行访问,说明以这种方式定义的静态存储变量其作用域是局部性的。因此这种变量叫做内部静态存储变量。

- 静态全局存储变量 如果把变量定义在模块外部,则所有函数都可以使用这个变量,例如:

```
static int number;  
count _up()  
{number+=25;  
  return(number);  
}  
reset()
```

```
{number=0;
```

```
}
```

这样定义的存储变量叫做外部静态存储变量。

③性质 静态存储变量有如下性质:

- 作用域 内部静态存储变量的作用域为函数或模块内;外部静态存储变量的作用域为整个程序。

- 生存期的永久性。

- 未初始化的静态存储变量的值为 0。

下面说明静态存储变量和自动存储变量的区别。

[练习 2.7]

```
A>type li7.c
```

```
main()
```

```
{count();
```

```
count();
```

```
printf("\n");
```

```
add();
```

```
add();
```

```
}
```

```
count()
```

```
{static int num0=0;
```

```
num0+=25;
```

```
printf("%d\t",num0);
```

```
}
```

```
add()
```

```
{auto int num1=0;
```

```
num1+=25;
```

```
printf("%d\t",num1);
```

```
}
```

```
执行: A>li7
```

```
25 50
```

```
25 25
```

(3) 外部存储变量

①用途 用于把大程序分割为若干程序单元(文件)而开发的场合。说明外部存储变量使用关键字 extern。例如把一个程序分为两个单元来编译:

```
A>type file1.c
```

```
int i,j;
```

```
float x; /* 定义外部存储变量 i,j 和 x */
```

```
main()
```

```
{
```

```
func1(); /* 函数调用 */
```

```
func2(); /* 函数调用 */
```

```
}
```

```
A>type file2.c
```

```
extern int i,j;
```

```
extern float x; /* 对外部存储变量的说明 */
```

```
func1(); /* 函数定义 */
```

```
{
```

```
;
```

```
}
```

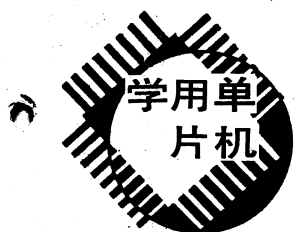
```
func2(); /* 函数定义 */
```

```
{
```

```
;
```

```
}
```

(转第 4 页)



普及型单片机开发工具的硬件设计

中国科学技术大学计算中心 张培仁 刘振安

编者的话:单片机已经广泛用于机电产品、家用电器等社会生产、生活等各个领域。由于单片机的独特优点、已为广大电子爱好者所喜爱,为他们开发电子产品的智能应用,提供了无限广阔的天地。

大家知道,在实际应用中,特别是对爱好者来说,使用单片机,只要设计一个最小系统或者选用一个最小系统就可以了。关键是要结合自己的应用项目,装入控制该“应用项目”程序的 EPROM 芯片和相应的接口电路。对于业余爱好者来说,在编制某项专用程序时,不能纸上谈兵,必须有一个调试此程序(或叫系统软件)的工具。正如,电子爱好者用万用表、示波器等工具、调试收音机、电视机及其它电子产品一样,单片机的应用,也要通过开发工具进行调试,所不同的是,前者是调整硬件的工作状态;后者确要调试所需要的应用系统的硬软件。

如果购买一台专用单片机开发工具,其售价是爱好者无法接受的。为此,本刊特约中国科学技术大学计算中心微机室张培仁、刘振安同志为本刊组织并撰写有关单片机开发工具及典型最小系统应用方面的技术文章。他们选用 KDC—Ⅲ 型单片机开发机介绍给读者,硬软件资料全部透明,并愿意为本刊读者提供咨询服务。对于他们为繁荣我国单片机应用事业和无私奉献的精神,我代表本刊工作人员和广大读者向他们表示深深的谢意。相信会有更多的作者和读者同我们一起办好本刊“学用单片机”专栏。

单片机开发工具(或称开发机)从使用角度上看,主要是解决编程、排错、仿真三方面功能。

1. 编程功能

单片机缺乏自身编程的能力,需要借助开发机来进行编程,所以,开发机应有机器语言、汇编语言、高级语言三种方式编程的能力。

2. 排错功能

任何比较复杂的程序常常不会一次编程就完全编好,而经常需要排错并修改程序。排错时首先要知道错在什么地方,这就要求开发机必须提供以下排错手段:

(1)单步:用户可以一次只执行一条指令。执行一条指令后即返回监控程序。

(2)运行:用户程序可以从任何一条地址启动,然后全速运行。

(3)断点运行:用户可以设置断点,当程序执行到断点时,控制返回到监控程序。

(4)检查和修改存储器的内容。

(5)跟踪:它能跟踪单片机运行时的每一指令周期

中地址,数据,I/O 端口和控制总线上的信息。

3. 仿真功能

仿真就是指开发机能通过仿真器的硬件和软件“真实”地模拟所开发的应用系统的运行情况。开发机不但仿真单片机的 CPU,还能仿真存储器和 I/O。端口,也要仿真单片机的中断系统的运行,使用户能利用单片机尽量多的资源。在调试用户应用系统时,监控程序和用户程序都要使用单片机中的资源,从而完成相应任务。这时有一个控制权相互转移的问题,常常是监控程序启动用户程序,又在某种条件下(如碰到断点)从应用程序回到监控程序,因此在设计开发机监控程序时要尽可能少占单片机的资源。

开发机的种类:目前大致可分为三种。一种普及型开发系统即象单板机那样,有近 30 个键,可以和 PC 联机形成一种联机调试系统。具有编程、排错、仿真三大功能。第二种是独立型仿真器,可直接用汇编或高级语言编程,是一种在线实时仿真开发系统,能在同一种机型中开发多种芯片的单片机,系统扩充也比较方便。第三种纯软件单片机模拟仿真系统,无硬件设备,投资最少。它是用软件方法模拟仿真单片机硬件环境,指令系统应用及开发系统。它给开发单片机提供新的手段。

下面我们详细介绍一种普及型 KDC—Ⅲ 型单片机开发机,供读者学习或选用,或根据我们所提供的硬软件资料自行制作。KDC—Ⅲ 单片机开发机和目标系统共用一个 CPU8031。整个硬件成本较低,而性能上又可完全满足用户开发单片机的需要。

一、硬件结构

KDC—Ⅲ 电原理图见图 1,它采用 8031 作为处理器,外接一片 2764(8KEPROM),一片 6264(8K 的 RAM)一片的 I/O 接口芯片。基本上是在单片机最小系统的基础上加扩充而构成。系统时钟采用内部时钟方式,即利用 8031 内部振荡电路,在 XTAL1、XTAL2 引脚外接 6MHz 晶体。内部的振荡电路处于自激振荡。复位开关采用电平方波复位,复位电路中电阻,电容参数和 CPU 的时钟频率及芯片最小复位时间有关。电源开启时可以自动进行复位。

开发机利用 8031 的 P0 口和 P2 口构成地址总线和数据总线。P2 口作为外部存储器的高 8 位地址的输出口。P0 口既是输出低 8 位地址又输出(或输入)相应数据的端口。P0 口在输出低 8 位地址时由 ALE 打入外部地址锁存器(74LS373)中,使地址锁存输出,在整个指令周期期间形成稳定的低 8 位地址。

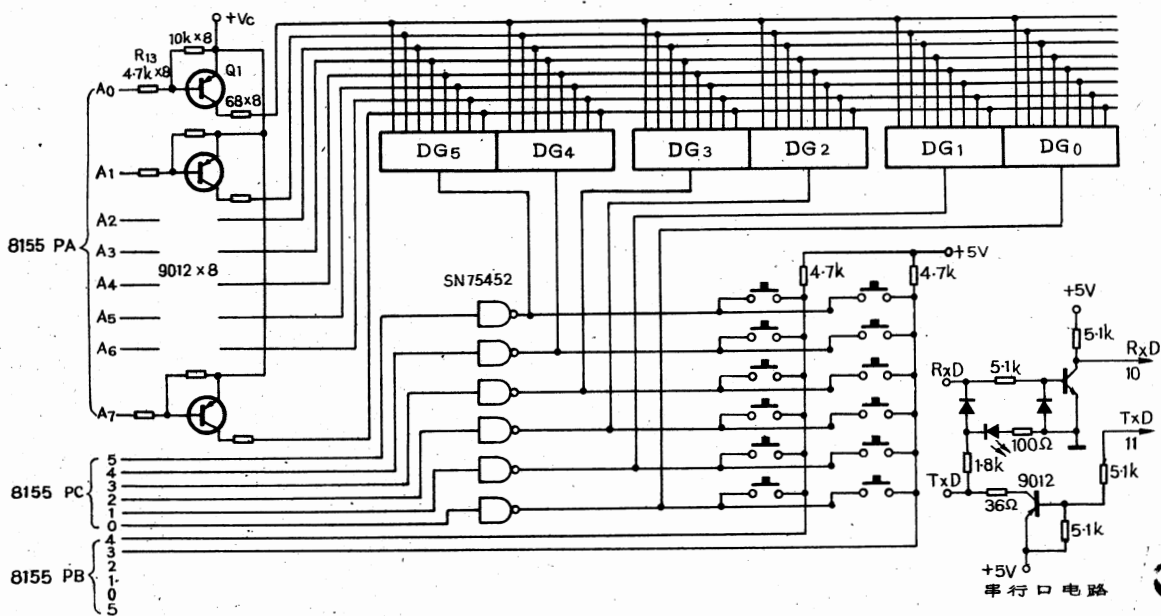
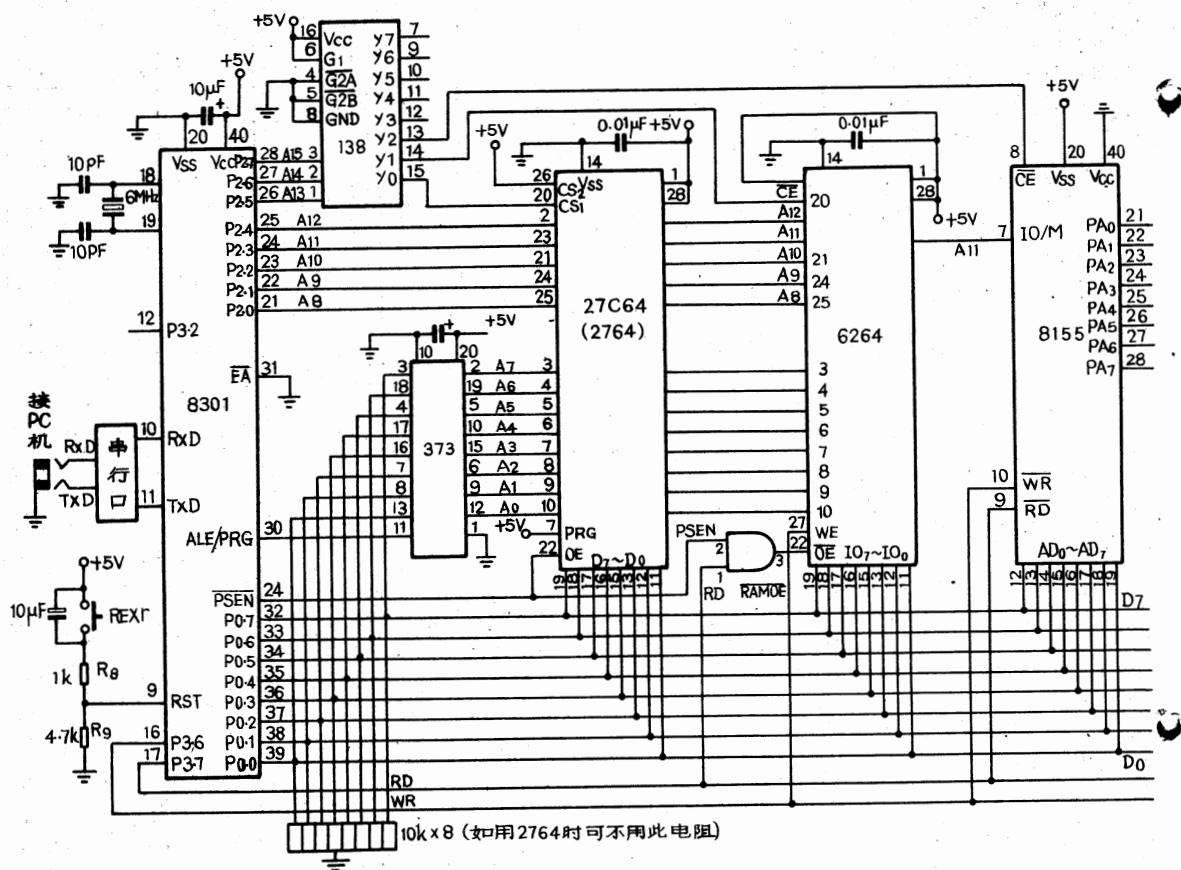


图 1

8031 外部程序存储器由 $\overline{\text{PSEN}}$ 信号选通。8031 外部扩展 I/O (如 8155) 由 $\overline{\text{WR}}, \overline{\text{RD}}$ 及地址译码器的信号选通。

开发机的地址分配如下:

Y0: 0000H ~ 1FFFH; Y1: 2000H ~ 3FFFH; Y2: 4000H ~ 5FFFH; Y3: 6000H ~ 7FFFH; Y4: 8000H ~ 9FFFH; Y5: A000H ~ BFFFH; Y6: C000H ~ DFFFH; Y7: E000H ~ FFFFH。

2764 的地址是 Y 数, 作为程序存储器; 6264 作为数据存储器, 地址为 Y1。8155 的内 PRAM 地址是 4000 ~ 40FFFH。8155 的 I/O 口, A 口地址: 4101H, B 口地址: 4102H, C 口: 4103H, 控制口: 4100H。

开发机和 PC 通信通过 PCRS232 口 (三条线) 和开发机三芯插座相联接。通信波特率是 1200。

以下我们从三个方面, 分析 KDC—Ⅲ 开发机电路图。

1. 最小系统的扩充;
2. 最小系统与键盘、显示器的接口;
3. 最小系统与 PC 的通信。

一、最小系统的扩充

大家知道, MCS—51 最小系统应用时, 经常需要加以扩充, 因为扩充方法比较典型和规范, 又都是选用常规芯片, 所以, 扩充是比较容易的。

1. 总线的扩充应用

从总线观点来看, 单片机扩充主要是片外的地址总线, 数据总线, 控制总线三方面的扩充。

地址总线 (AB) 16 条, 可寻址 64K。单片机用 P0 分时输出低 8 位的地址, P2 口输出高 8 位地址。因为低 8 位地址是分时输出的, 所以要用锁存器锁存。在最小系统中 P0、P2 已经作为地址口使用, 就不能再作为 I/O 口使用了。

八位数据总线 (DB) 由 P0 口输入输出。对外接的扩展片, 一般由地址线控制片选信号。

控制总线 (CB), 主要扩展控制线有: $\overline{\text{WR}}, \overline{\text{RD}}, \overline{\text{PSEN}}, \text{ALE}, \text{EA}$ 等。

在单片机最小应用系统扩充时, 首先要注意是否还有未用的地址口。另外要考虑扩展口 (P0~P3) 本身负载能力。从理论上讲 P0 可以接 8 个 LSTTL 电路, 其它口只能驱动 4 个 LSTTL 电路。实际上我们使用过的大部分 8031, 负载能力要比理论上小。一般小于 5mA 时, I/O 的电平基本正常。在系统扩充时要注意电平的转换。一般接口是 CMOS 电路要接上拉电阻。在和高电压接口时最好通过光电耦合器 (如 4N25) 来完成。

2. 存储器的扩充

MCS—51 的存储器可分程序存储器和数据存储器。这两个存储器地址可以重叠, 用单独的控制信号和指令可以把两者区分开。一般程序存储器由 $\overline{\text{PSEN}}$ 控制, 读取数据用 MOVX 的指令。数据存储器用 $\overline{\text{RD}}, \overline{\text{WR}}$ 控制信号和 MOVX 指令, 进行读写。

KDC—Ⅲ 开发机, 监控程序只有 2K 左右。而我们选用 2764 (8K) 作程序存储器的主要原因是为了以后

扩充 (实际上我们把一个通用系统的监控程序和子程序库扩充到了余下的 6K 中)。另一方面目前 2764 的市场价格还低于 2716, 2732。此外 2764 和 27128、27256 都是 28 脚芯片, 再扩充很方便。三种芯片区别如下表所示:

管脚	2764	27128	27256
27	$\overline{\text{PGM}}$	$\overline{\text{PGM}}$	A14
26	N · C	A13	A13

从图中看出 2764 和 27256 只有二个脚有所区别, 如需将 2764 换用 27256 时, 可用跳线来解决 27、26 管脚的选线问题。

片选信号 (20 管脚) $\overline{\text{CE}}$ 接法的考虑, 在最小系统中如果只扩充 EPROM 时, 可把 $\overline{\text{CE}}$ 接地。如果还有其它一些芯片也要扩充时, 可以用最高位 P2.7 作为选通的控制信号。也可以将 P2 口高位用 74LS138 的输出 Y0~Y7 作选通信号。

数据存储器与程序存储器地址 (0000H—FFFFH) 重叠, 可使用不同指令和控制信号来解决扩充中选址问题。6264 控制信号如下表:

6264 方式选择	$\overline{\text{CE}}$	$\overline{\text{WE}}$	$\overline{\text{OE}}$	功能说明
写	0	0	1	数据写入 6264
读	0	1	0	数据从 6264 读出
非选中	1	×	×	数据线高阻

6264 选的地址是 2000H—2FFFH

经常采用两种方法对扩充 6264 (8K × 8) 进行读写。

其一, 读写指令用 R0 和 R6 作低 8 位地址指针, 高位地址由 P2 口单独输出。如下程序向 6264 中 2100H 为开始地址的内存写数据。

程序如下:

```

.ORG 2000H
MOV R0, #00H
MOVA, #01H
RE1:  MOV P2, #21H
      MOVR1, #00H
      MOVX @R1, A
      INC R0
      DJNZ R0, RE1
HE1:  SJMJ HEI

```

其二, 用 DPTR 作为 6264 的地址指针 (KDC—Ⅲ 监控程序中用此方法)。

```

.ORG 2000H
MOV R0, #00H
RE2:  MOVA, #02H
      MOV DPTR, #2100H
      MOVX @DPTR, A
      INC DPTR

```

```

INC R0
DJNZ R0, RE2
HE2:    SJMP HE2

```

在最小系统中,数据存储器 and 程序存储器都是分开的,但有些场合希望两者合用。在 KDC—Ⅲ 开发机中把 6264 既作为数据存储器,也作为程序存储器。当 PC 机把用户的程序装入 6264 时,是把 6264 作为数据存储器使用,而在执行用户程序时,6264 作为程序存储器使用。我们是用 $\overline{\text{PSEN}}$ 和 $\overline{\text{RD}}$ 信号相“与”的方法达到两者合用(见图 1)的。当 $\overline{\text{PSEN}}$ 与 $\overline{\text{RD}}$ 都作为正与门的输入信号,产生低电平的读选通信号。可以用 MOVX 指令把“程序代码”写入 6264 中(利用 $\overline{\text{WR}}$ 低电平有效信号),还可用 MOVX 指令全读出修改之(利用 $\overline{\text{RP}}$ 和 $\overline{\text{WR}}$ 信号),然后作为程序存储器使用,执行已写入的程序(利用 $\overline{\text{PSEN}}$ 信号)。

最小系统扩充数据存储器(RAM)时,一般用静态存储器而较少地用动态存储器。主要原因扩充方便,不需要刷新,价格也较便宜。

3. I/O 口的扩充

最小系统由于片外有 EPROM,常用 P2、P0 口作为高低位地址使用。只有 P1、P3 两个 I/O 口可使用。如果 I/O 不够用,可进一步扩充 I/O 口,主要有两个方法:

1. 总线扩展方法。即用 P0 口作为数据口,而选片信号采取与数据存储器相同寻址方法。扩充 I/O 连接的外围设备,均与片外数据存储器统一编址。每一个 I/O 芯片占用一个片外 ROM 地址或一个片外 ROM 区域,而这时与程序存储器无关。在 KDC—Ⅲ 开发机中,373、6264、8155 都用这个方法扩充的。

2. 串行口扩展方法。因为最小系统中利用串行口移位寄存器功能扩展 I/O 口。这种方法主要优点是占单片机的资源很少,接口简单,但相对并行口来说速度比较慢。

二、最小系统与键盘显示器的接口

某些最小系统在应用时,需要外接显示器和键盘。现以 KDC—Ⅲ 开发机为例,采用一片 8155 解决键盘扫描,数码管显示和最小系统的连接问题。有关电路图见图 1。

1. 程控扫描显示电路

8031 通过扩展接口 8155 的 A 口输出段码,0 电平有效。C 口输出位码,1 电平有效。当六个数码管共阴极数码管全显示时,可逐次把所需的字符显示在规定的字位上,每点亮一个数码管之后,稍停一段时间(1ms~2ms),使之发光稳定,之后点亮下一个。这样巡回扫描显示速度较快,每秒可重复多次。虽然在不同时刻只有一个数码管工作,但利用人眼的视觉暂留效应和发光二极管灭时余辉效应,所以看到的是六个字符同时显示。这种巡回扫描显示需要靠程序来完成。下面给出 KDC—Ⅲ 开发机的扫描显示程序框图(见图 2)。

2. 程控键盘扫描电路

在图 1 中,程序不断从 PC 口反门(即 SN75452 输出端)输出扫描码 01H、02H、04H、08H、10H、20H。这些扫描码分别控制 DG0~DG5 六个数码管(共阴极)点亮。这个扫描码同时也扫描 KDC—Ⅲ 的 12 个键的小键盘。一旦用户有键按下,这时 8155PB 口的 PB3、PB4 就有 0 电平出现,单片机不断从 PB 口取走数据判断是否有键按下,如有键按下进入键分析程序和键处理程序,最后执行相应子程序。这种方法叫接口扫描法。

键盘扫描子程序 KEYSCRN. ASM 的程序框图(见图 3)。

三、最小系统与 PC 的通信

由于单片机内(或外接)可以固化一定的控制程序,并带有相应的数据存储器 and 通信用的串行口,十分适合在小型通信系统中应用。可以最小系统为核心,连接多个最小系统的分布式系统,也可以连成以 PC 为主机和多个最小系统组成的分布式主从系统(或多级控制系统)。

结合 KDC—Ⅲ 加以说明,见电路图 1 中串行接口电路部分。PC/XT 的 RS232 接口的输入输出分别采用 MC11488 和 MC1489。我们采用 9013、9012 两只晶体管把 RS232 的标准电平 -12V 表示 1, +12V 表示 0 转变为 0~5V 电平,以便和最小系统中的 8031 连接。D 是发光二极管,当 PC 与最小系统通信时,有明暗变化。这种电路是一种准 RS—232 接口电路。

通信所用连线为双绞线,通信距离在 30M 以内。如果距离较远可用光电隔离或用电流环来驱动。由此可知,通信接口电路主要是解决电平转换问题,而通信的实施则是靠软件进行的,它们分别是单片机通信模块和 PC 机通信模块。

这二个模块分别由初始化模块、接收模块、和发送模块等三个子模块组成。下面介绍一下

PC 与最小系统通信(如 KDC—Ⅲ 开发机)的过程。首先,把编译好的单片机通信程序送入单片机片外数据存储器某地址处(如 2000H)开始的内存中。并运行这个通信程序(文件名 DP. ASM),这时开发机的监控程序已停止工作。

其次,在 PC 上编辑一个数据文件,如 ABC. ASM 文件。

```

ORG 2100H
DB 00H,11H,22H,33H,44H,55H,
DB 66H,77H,88H,99H,0AAH,0BBH,
DB 0CCH,0DDH,0EEH,0FFH
END

```

把 ABC. ASM 进行编译产生 ABC. OBJ 文件。

最后,在 PC 上运行 PC 通信模块(文件名是 ZHAN KAN. PAS)。在 PC 上显示出提示,并要选择(1

1. PC 向 8031 送数据
 2. 8031 向 PC 传送数据
 3. 返回 DOS。
- ~3)。

应首选 1,把 ABC. OBJ 文件送到单片机以 2100H 为起始的地址处。送完又返回提示。为了检查一下传

送情况,可再选2。单片机外存将2100H处的16个(十进制数)字节内容传回PC,形成一个文件名是ABC1.OBJ,最后返回DOS。此用DOS的TYPE命令分别显示ABC.OBJ和ABC1.OBJ二个文件的内容,即可证明通信过程的正常。

以上从硬件角度,结合KDC—Ⅲ介绍了单片机最

小系统及最小系统的扩充、最小系统与键盘、显示器的接口、单片机最小系统与PC通信等问题。

下讲从软件角度,结合KDC—Ⅲ的监控程序的设计,通信模块的设计,并向读者提供全部程序清单。

本刊第三期刊载“普及型单片开发工具的软件设计”。

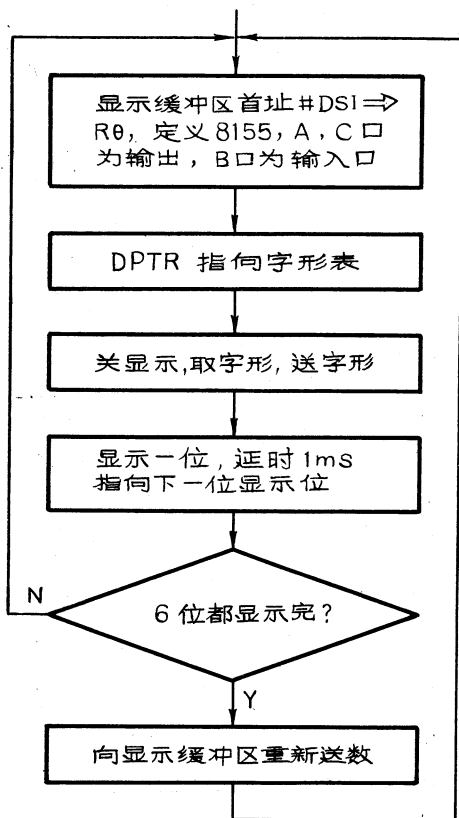


图3

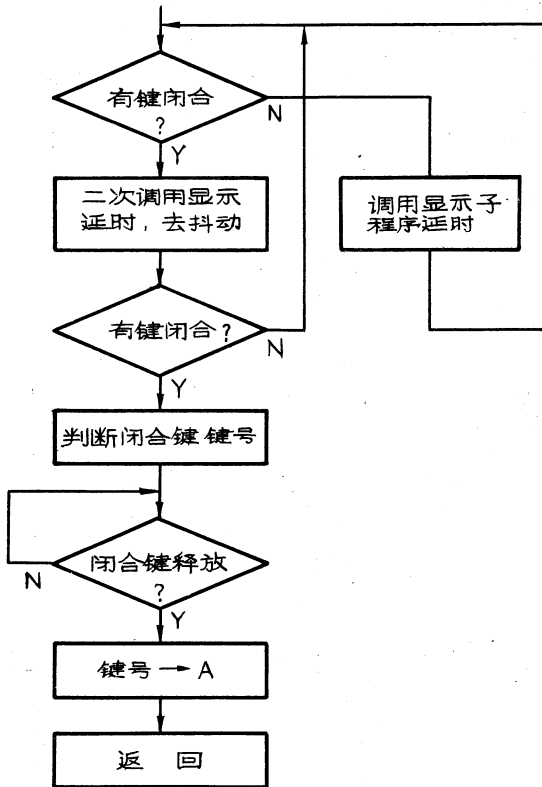


图4

(接36页)

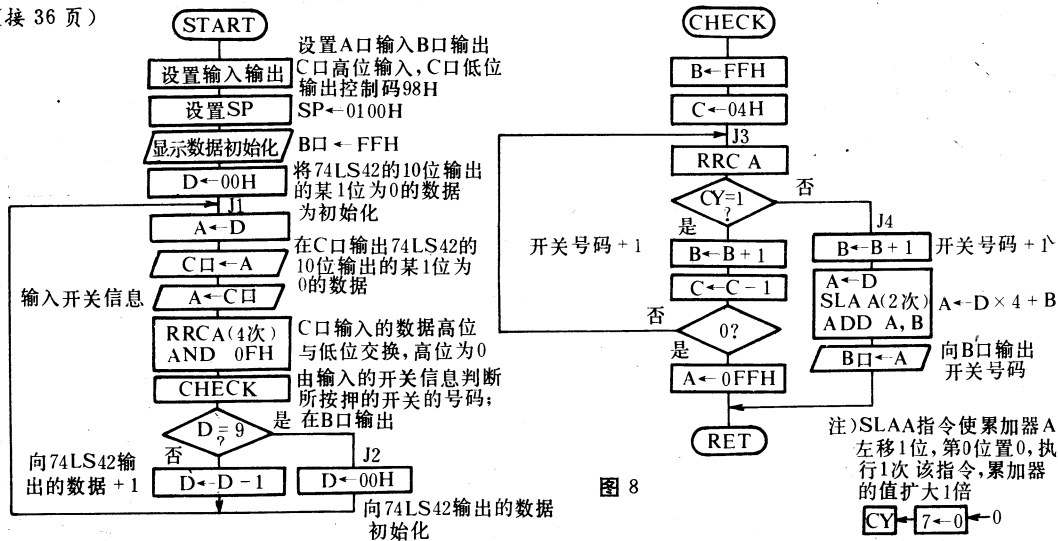


图8



学装微电脑

学装 $\mu\text{P}-80$

40 支开关输入部件的制作

易齐干

微电脑控制机器工作时往往要接很多微动开关、按键开关、继电器与电磁铁等。通过 8255 将开关状态输入给微电脑,或驱使负载工作。

一片 8255 有 A、B、C 三组输入输出端口只能处理 24 个输入输出信息。如何增加 8255 的输入输出信息呢?这是本文要讨论的问题。

1. 动态开关扫描

本文介绍使用 $\mu\text{P}-80$ 微电脑教育部件 8255 的端口 C,通过 74LS42 输入 40 支开关状态的方法——采用矩阵方式,即把全部开关划分为几个组,按组平均轮流输入开关状态,这种方法也称为动态开关扫描。仅用端口 C 开关输入电路如图 1 所示。端口 C 高位 $\text{PC}_7 \sim \text{PC}_4$ 为输入口,低位 $\text{PC}_3 \sim \text{PC}_0$ 为输出口,在不按动任何一支开关时,输入端为高电平(使用电阻对端口 C 高位进行上拉)。

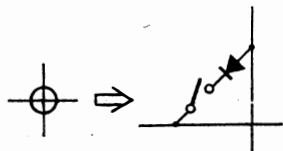
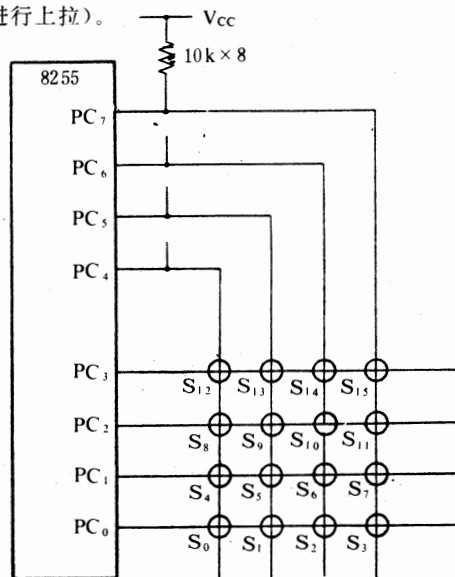


图 1 端口 C16 支开关输入电路

首先,由端口 C 输出 11111110(这时仅 PC_0 为低电平。高 4 位设置成输入端之后,其信号决定于其它端口输出,所以,设置为 0 或 1 都可以)。当端口 C 高

位输入低位的信息时,用程序查询 $\text{PC}_4, \text{PC}_5, \text{PC}_6, \text{PC}_7$ 为 0 或是为 1。如 S_0 为“通”, $(\text{按键})\text{PC}_4$ 为低电平,故可查出 S_0 被按动。如果 S_2 为“通”,因为 PC_6 为低电平,故查出 S_2 被按动。查询按动的开关序号的程序流程图如图 2 所示。用 4×4 矩阵输入 16 支开关状态,是常用的键联接方法。

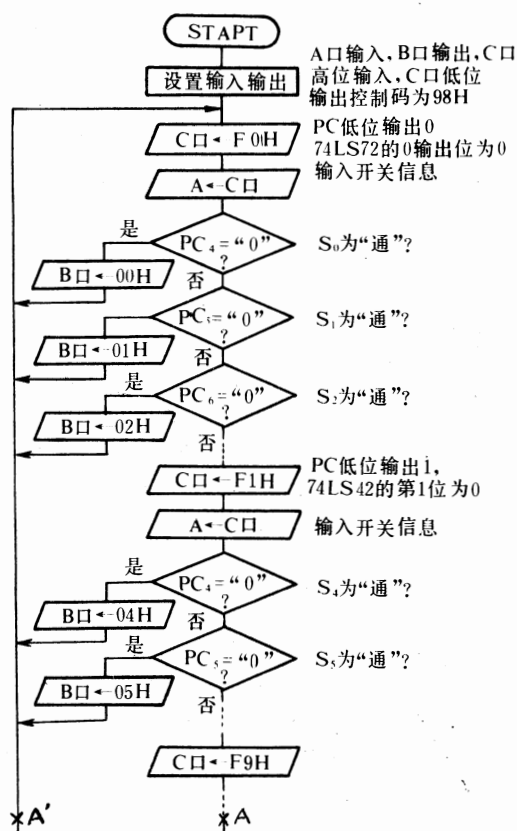


图 2 输入 16 支开关状态的流程图

2. 仅用 C 口,输入 40 支开关状态的方法

前面已经介绍了矩阵方式,要增加输入开关的数目。可使用 74LS42(BCD \rightarrow 十进制译码器)功能如图 3 所示,根据 A、B、C、D4 个输入端的状态,输出端 0~9 可选择其中之一为低电平。使用这种 IC 如图 5 所示,端口 C 低 4 位为输出端,与 74LS42 的 A、B、C、D 输入端相连,分组数目可达 10 组,所以能输入 40 支开关状态。

No.	输 入				输 出									
	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	L	H	
9	H	L	L	H	H	H	H	H	H	H	H	H	L	

图 3

开关 $S_{in:m} = S_{1 \times 2 + 3} = S_{11}$

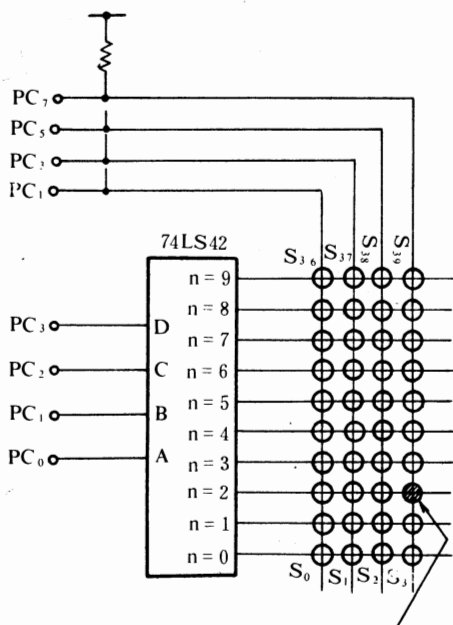


图 4 40 支开关输入电路

3. 与微电脑教育部件的连接

位数字显示部件显示所按动的开关序号 N_0 。

图 6 给出扫描 40 支开关输入的流程图(其中 \times 部分与图 2 联接), 可以看出 74LS42 10 位输出中仅有 1 位轮流为低电平, 端口 C 低位输出数据, 同时向端口 C 高位输入开关状态, 判断哪支开关被按动, 由端口 B 输出开关编号数据。照图 7 所示的流程图则能编制出这种程序。将程序写入微电脑教育部件, 如果没有差错, 则能较好地执行动作。但此程序编出后很冗长。

如何考虑出比较短的程序呢? 方法如图 7 所示的流程图。以图 4 的电路图为基础, 例如, 假设 S_0 为

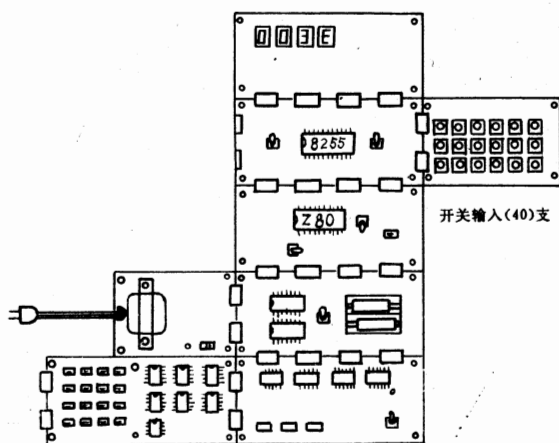


图 5

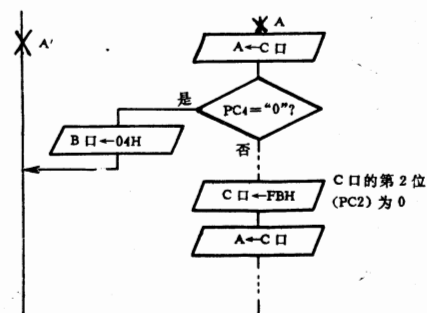


图 6

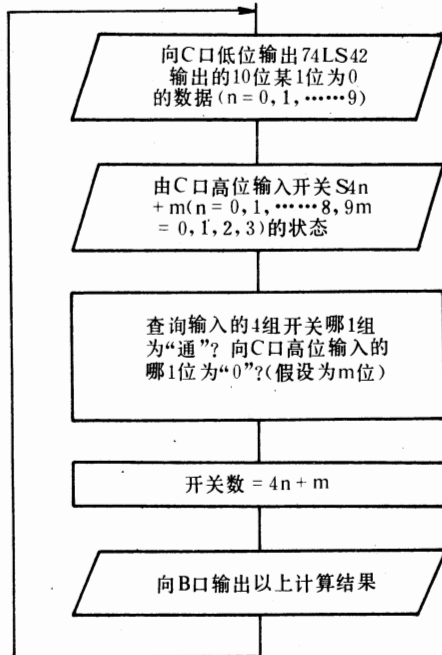


图 7

助记符	地址	机器语言	注释
STARY	LD A, 98H	00 3E 98	设置 CW
	OUT(03H), A	02 D3 03	
	LD SP, 0100H	04 31 00 01	设置 SP
	LD A, FFH	07 3E FF	显示数据初始化
	OUT(01H), A	09 D3 01	
	LD D, 00H	0B 16 00	初始设定
J1	LD A, D	0D 7A	向 74LS42 输出数据
	OUT(02H), A	0E D3 02	
	IN A, (02H)	10 DB 02	输入开关信息
	RRC A	12 CB 0F	高位与低位交换、高位为 0
	RRC A	14 CB 0F	
	RRC A	16 CB 0F	
	RRC A	18 CB 0F	
	AND 0FH	1A E6 0F	
	CALL CHECK	1C CD 40 00	判断所按开关号码与输出显示
	LD A, D	1F 7A	D=9?
	CP 9	20 FE 09	
	JP Z, J2	22 CA 29 00	向 74LS42 输出的数据+1
	INC D	25 14	
J2	JP J1	26 C3 0D 00	
	LD D, 00H	29 16 00	
CHECK	JP J1	2B C3 0D 00	
	LD B, FFH	40 06 FF	设置累加器 A 中的第 0 位数据初始值
J3	LD C, 04H	42 0E 04	设置检验位的数据
	RRC A	44 CB 0F	右转
	JP NC, J4	46 D2 51 00	如果为 0 跳向 J4
	INC B	49 04	$B \leftarrow B + 1$
	DEC C	4A 0D	$C \leftarrow C - 1$
	JP NZ, J3	4B C2 44 00	检验位全部结束跳向 J3
	LD A, FFH	4E 3E FF	
	RET	50 C9	
J4	INC B	51 04	$B \leftarrow B + 1$
	LD A, D	52 7A	$A \leftarrow D \times 4 + B$
	SLA A	53 CB 27	
	SLA A	55 CB 27	
	ADD A, B	57 80	
	OUT(01H), A	58 D3 01	输出开关号码
	RET	5A C9	

图 9 程序清单

“通”时, S1 为“通”时, 进行分析很容易理解。详细的流程图如图 8 所示, 程序清单如图 9 所示。与图 6 的流程图相比较, 相当短而简炼(但是, 对于不熟练的读者难于理解), 不能很快理解的读者, 首先要进行模仿, 在模仿之中逐步搞懂。

本文, 介绍了仅用端口 C 输入 40 支开关状态的动态开关扫描方法。由所介绍可知, 如果具备 74LS42 IC 知识(硬件知识)与动态开关扫描的程序知识(软件知识), 微电脑的应用能向深度、广度扩展, 所以说硬件(尤其是 IC 的使用方法)与程序的研究是极其重要的。

(本文图 8 转第 33 页)



MP-I 汉字及图形显示装置

铁道部第四勘测设计院 陈名则

MP-I 微电脑具有价格低,易于开发,应用范围广的特点,本人开发的 MP-I 汉字及图形显示装置,可以在用 LED 组成的屏幕上显示汉字及各种图形,汉字及各种图形可通过程序控制和变化,适合用于作广告及招牌,有一定的实用价值。本装置扩展了原机的接口,并对原机的地址译码电路作了适当改动,现分别介绍于后:

1. 接口扩展电路

MP-I 机原来的输出接口只能显示 8 位二进制数的状态,即 8 个点,而要显示一个字符,一般需 7×9 个点。本装置对原机的输出接口进行了扩展,共可显示 32×16 个点,每屏可显示 4 个 7×9 点阵的汉字。根据本装置的原理,还可扩展到 64×16 点阵或 32×32 点阵。接口扩展电路见图 1。

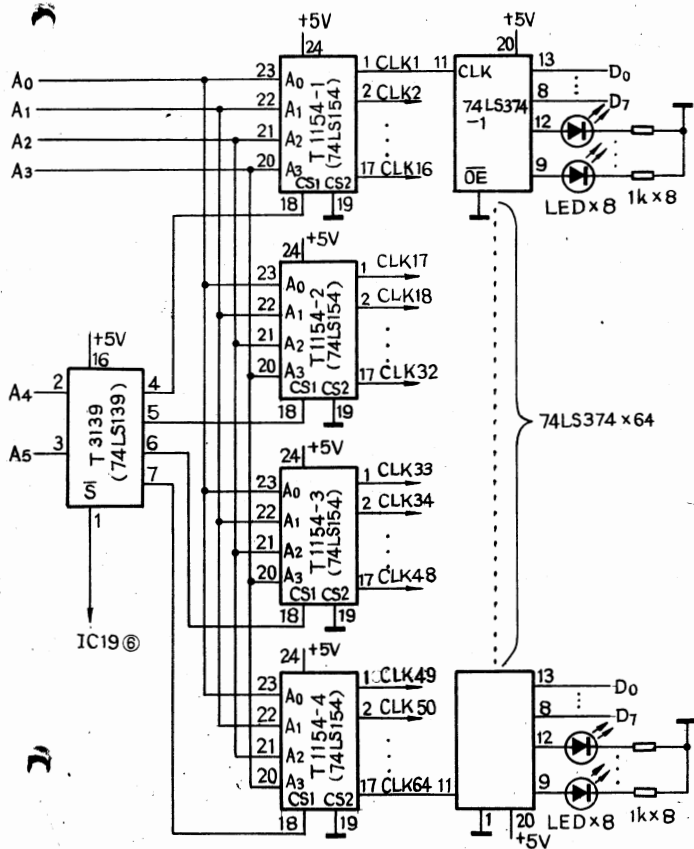


图 1

扩展电路采用 1/2 块 2 线—4 线译码器 T3139 (74LS139) 及 4 块 4 线—16 线译码器 T1154 (74LS154) 组合成 6 线—64 线译码器,将 T3139 的片选端①脚接 IC19 的⑥脚,并将 IC19 ⑥脚与原机 IC21 (74LS374) 的⑪脚断开,使在选中输出接口页面地址时,扩展的接口电路总是处于工作态。64 块 74LS374 组成 64 个 8 位数据锁存器,每块 374 连接 8 个 LED。其工作原理是,当程序选中输出接口时,T3139 的①脚 \bar{S} 处于低电位,6 线—64 线译码电路处于工作态,根据地址线 $A_5 \sim A_0$ 提供的地址码中的高两位 $A_5 A_4$,先由 T3139 译码。选中 T1154 中的一块,再根据低四位 $A_3 \sim A_0$ 选中 T1154 的 16 条输出线的一条,由于 T1154 选中的输出线为低电平,这个信号正好作为相应的 74LS374 的使能信号,进入 74LS374 的 CLK 端,使数据线的信息进入相应的 374 锁存。由于 74LS374 的 \overline{OE} 端接地,其内部的三态门总是处于工作态,锁存的信息通过 LED 显示出来,当 CLK 端变为高电平后,这一组发光二极管显示的内容保持不变。这样,通过程序控制机器发出接口页面不同的输出地址码和按一定要求编制的数字码,就达到了在 32×16 的点阵上显示汉字和图形了。

2. 原机译码电路的改动

原机译码电路改动的电路见图 2。具体做法是,将 IC18 的⑤脚与 A_8 断开,改接至 A_9 ; IC18 的②、③脚与 A_9 断开,改接至 6502 的 A_{10} 端,再利用原机计数器 IC11 的空闲部分,在 IC11 (74LS93) 的⑧脚接一个三态门,其控制端接 IC4 的①、⑨脚,输出端与 A_{10} 相接,并在

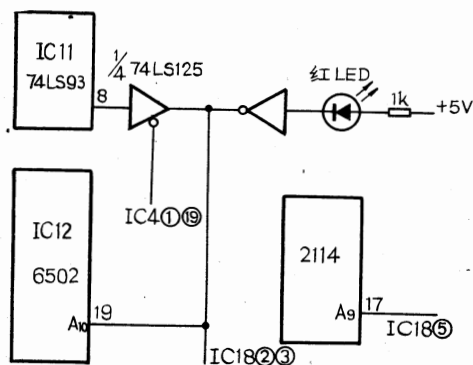


图 2

A₁₀端按原机的显示方法加一个红色发光二极管,用以显示 A₁₀位的状态。改动后,输出端口的地址为 \$ 401 ~ \$ 4FF 的奇数地址,每一个地址对应于一组 8 个数据显 LED。这样,不但扩大了显示的范围,同时也将整个第 2 页面的地址让给了 RAM2114,增加了存储器的存储范围。

3. 使用方法

将发光二极管 8 个一组,每行排 4 组,共排 16 行,则组成了 32×16 的点阵。将要显示的字符和图案在

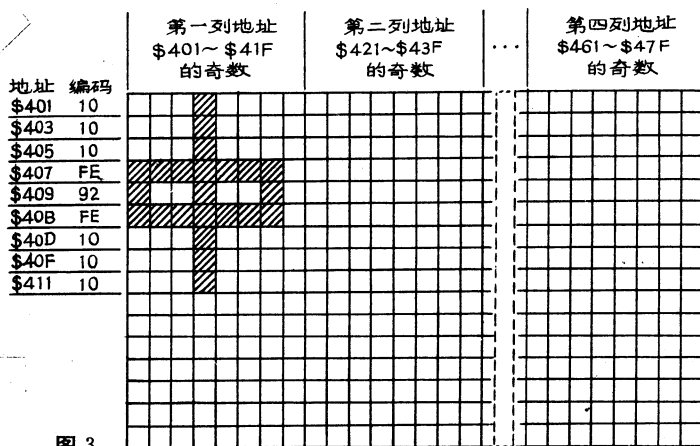


图 3

32×16 的方格纸上描好,编出相应的 8 位二进制码,按一定的顺序编入程序,输入机器,运行后,即可显示出相应的字符和图案。图 3 是显示“中”字的图形及其相应的编码和汇编程序。

要注意的是,由于原机所配稳压电源容量小,因此,显示器及接口扩展电路要另设单独的电源并将扩展电路的地端与原机的地端连接。

程序:

```
LDA # $10
STA $401
LDA # $10
STA $403
LDA # $10
STA $405
LDA # $FE
STA $407
LDA # $92
STA $409
LDA # $FE
STA $40B
LDA # $10
STA $40D
LDA $1F; 这三个地址传送的数据相同。
STA $411;
```

APPLE—II 彩灯自控电路

重庆南开中学 张孝玖

由图 1 所示的 APPLE—II 微机的游戏接口插座和图 2 所示的接口电路,在微机程序的控制下,可组成一个具有四个自动开关的控制电路。用三十二个彩色小灯泡分成四组,连接成如图 3 所示的电路,可用它来模拟晚会舞台彩灯或节日彩灯。将图 3 中的 1,2,3,4 端和图 2 中的 1,2,3,4 端分别连接在一起,并将图 2 中的 K 端和图 3 中的 A 端分别接在灯泡电路供电源的两极上。在运行程序时,小彩灯会产生多种形式的变化灯光。

一、APPLE—II 微机游戏接口插座简介:

在微机的本机底板上,有一个十六脚未插集成电路块的插座,其旁标有 GAMEI/O 字样,这个插座即游戏接口插座。这里只介绍与本文有关的几个引脚,见图

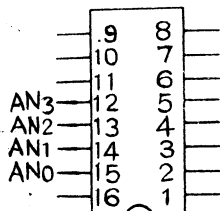


图 1

1。8 脚是接地端,电源负极。1 脚+5 伏,为电源正极。15、14、13、12 脚分别叫 AN₀, AN₁, AN₂, AN₃,是逻辑电平输出端,处于高电平时约为+5 伏,低电平时约为 0 伏。这四个输出端电平的高、低由 POKE 语句控制。其控制地址见下表。

访问地址与逻辑电平状态之间的关系

输出端	信号状态	控制地址	
		十进制	十六进制
AN ₀	低电平	49240 — 16296	\$ C058
	高电平	49241 — 16295	\$ C059
AN ₁	低电平	49242 — 16294	\$ C05A
	高电平	49243 — 16293	\$ C05B
AN ₂	低电平	49244 — 16292	\$ C05C
	高电平	49245 — 16291	\$ C05D
AN ₃	低电平	49246 — 16290	\$ C05E
	高电平	49247 — 16289	\$ C05F

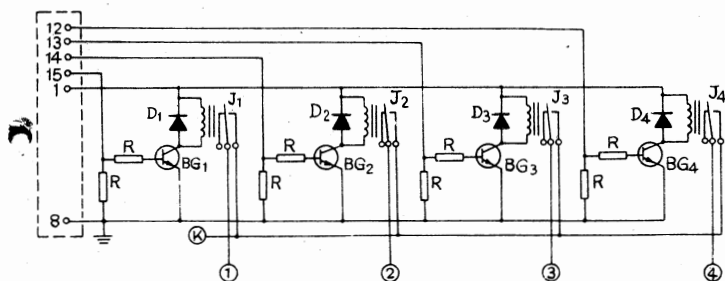


图 2

二、接口电路：

接口电路如图 2 所示，是由三极管 BG_1 — BG_4 组成的四个开关电路。 J_1 — J_4 是微小型继电器。三极管集电极电源由 GAMEI/O 插座的 5 脚供给。当 AN_0 为高电平时， BG_1 的 b—e 极间加上一个正脉冲触发信号而导通， J_1 的簧片吸合；反之， AN_0 为低电平时，b—e 极间电压为 0 而截止， J_1 的簧片释放。其余三个开关电路的工作状态与此相同。这样， J_1 — J_4 的公共接点 K 与 1、2、3、4 端就组成了可由程序控制的具有四个开关特点的自动开关电路。每个三极管处于开或关状态的时间长短是与 AN_0 — AN_3 相应输出端高、低电平的时间相同的。对 APPLE—II 微机，每空循环 750 次，即 $FOR\ T=1\ TO\ 750:NEXT\ T$ 约 1 秒钟。

电路中 BG_1 — BG_4 用 3DK 型三极管， D_1 — D_4 为普通二极管，所有的电阻值为 5.6K， J_1 — J_4 用约小于 +5 伏就能吸合的继电器。电路安装完毕后，一般无需调整，必要时，可调整接在三极管基极上的电阻，以使三极管处于良好的导通和截止状态。

三、彩色小灯泡电路：

彩灯电路如图 3 所示

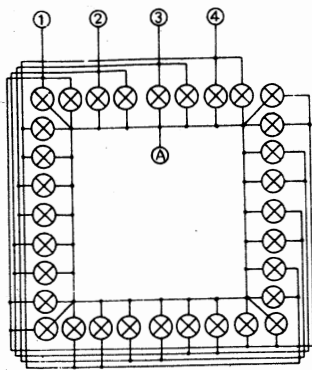


图 3

四、彩灯控制程序举例：

控制彩灯发光形式可依自己喜好设计，我编制的 BASIC 语言程序清单如下。运行该程序时，先是让所有彩灯全亮全灭地闪烁几下，接着是四组灯光匀速跑动，接着是匀加速跑动，接着是匀减速跑动。并且，上述每种情况都是先按顺时针后按逆时针方向进行的。最后，全部彩灯再闪烁几下便结束。

图 2 可作为一个简易通用控制电路，并具有制作简单、使用灵活方便、实用性强的优点。不仅可直接用于负载功率较小的被控电路，也可作为大功率被控电路的驱动电路。所以，它在多种控制设备中有较大用途。

LIST

```
10 A=49240:B=49246:FOR K=A
  TO B STEP 2:POKE K,O:NEXT
```

```
20 N=8:GOSUB 110
```

```
30 FOR TT=1 TO 1500:NEXT
```

```
40 P=1:T1=100:GOSUB 170
```

```
50 FOR TT=1 TO 1500:NEXT
```

```
60 P=2:T1=10:T=4*T1*N:T
```

```
A=T:GOSUB 170
```

```
70 FOR TT=1 TO 1500:NEXT
```

```
80 P=3:T=0:T1=40:GOSUB 170
```

```
90 GOSUB 110
```

```
100 END
```

```
110 REM SUB-1
```

```
120 FOR I=1 TO 10
```

```
130 IF 1/2<>INT(1/2) THEN
```

```
A=49241:B=49247:FOR J=
```

```
A TO B STEP 2:POKE J,0:NEXT
```

```
J:FOR TT=1 TO 300:NEXT:
```

```
GOTO 150
```

```
140 A=A-1:B=B-1:FOR J=A TO B STEP
```

```
2:POKE J,0:NEXT J:FOR TT=1 TO
```

```
100:NEXT
```

```
150 NEXT I
```

```
160 RETURN
```

```
170 REM SUB-2
```

```
180 A=49241:B=49247:C=2:GOSUB 250
```

```
190 FOR TT=1 TO 750:NEXT
```

```
200 LET A=49247:B=49241:C=-2
```

```
210 IF P=2 AND C=-2 THEN T=TA:GOSUB
```

```
250:GOTO 240
```

```
220 IF C=-2 THEN T=0
```

```
230 GOSUB 250
```

```
240 RETURN
```

```
250 REM SUB-3
```

```
260 FOR I=1 TO N
```

```
270 FOR J=A TO B STEP C
```

```
280 IF P=1 THEN T=T1:GOTO 310
```

```
290 IF P=2 THEN T=T-T1:GOTO 310
```

```
300 T=T+T1
```

```
310 POKE J,0:FOR TT=1 TO 300:NEXT
```

```
320 POKE J-1,0:FOR TT=1 TO T:NEXT
```

```
330 NEXT J
```

```
340 NEXT I
```

```
350 RETURN
```



王安机 RAM 故障简易检修法

董纯坚

WANG-PC 机中存储器 RAM 是微机芯片中最易损坏的集成块之一,维修起来即费时又费钱,但用户只要根据本文介绍的方法,不需要任何仪器设备和电路图,用随机诊断盘就能自行修理,这是一种经济、实用、快速的检修方法。

下面介绍一下 WANG-PC 机 RAM 芯片的分布情况,在主机箱内有 RAM 芯片的板,共有 3 块,它们是系统板,扩展内存板,显示板。

1. 系统板

系统板上有 256KB RAM,它分为四组 (BANK), RAM 芯片在系统板上的位置标号及代码见表 1:

下面举具体维修实例对表 1 加以说明。

例 1:故障现象

主机运机一段程序后“死机”屏幕显示:

? SYSTEM ERROR: I/O ERROR ON OPTI ON BOARD.

诊断:插入随机诊断盘进入软盘驱动器,选择系统卡项,诊断屏幕显示:DATA READ=0010.

表 1

RAM 标号	代码	RAM 标号	代码	RAM 标号	代码	RAM 标号	代码
L6	8000	L5	0080	L4		L3	
L12	4000	L11	0040	L10		L9	
L16	2000	L15	0020	L14		L13	
L22	1000	L21	0010	L20		L19	
L28	0800	L27	0008	L26		L25	
L34	0400	L33	0004	L32		L31	
L38	0200	L37	0002	L36		L35	
L44	0100	L43	0001	L42		L41	
L50	0000	L49	0000	L48		L47	

根据读数据为 0010 从上表对应关系可确定是 L21 芯片损坏,换该芯片主机恢复正常运行。系统板另外两排 RAM 如有损坏,则会引起主机不能正常启动,这两排 RAM 故障的排除可采用“芯片叠背法”。

2. 512K 扩展内存板

扩展内存板共有 8 排 RAM 芯片,每排有四个 4164 动态存储芯片,包括 8 个数据位和一个奇偶校验位, RAM 芯片的排布位置,段地址和代码形式见表 2:

例 2:故障现象

主机运行一会,停机屏幕显示内存 RAM 出错。

诊断:插入诊断盘,选择扩展内存卡项,屏幕显示:

OPENED OR SHORTED ADDRESS LINE- (LOC A000:0012H)

PARITY ERROR-(LOC A000:0008H. GOOD/BAD DATA FB/FFH)

表 2

RAM 位置																	段地址
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	E000 : ×××× 8000 : ××××
86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	C000 : ×××× C000 : ××××
66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	A000 : ×××× A000 : ××××

续表 2

46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	80000: ××××
02	04	08	10	20	40	80	80/ /80	80/ /80	01	02	04	08	10	20	40	80	8000: ××××
FD	FB	F7	EF	DF	BF	7F	FF/ /FF	FF/ /FF	FE	FD	FB	F7	EF	DF	BF	7F	
奇					校验位					偶							

根据屏幕显示 ADDRESS 和 RAM 的段地址都是 (A000: ××××), 可从表 2 中确定损坏的 RAM 芯片在 L65—L82 号这一排。根据代码 (FB/FFH) 可推算 L67 号或是 L77 号, 具体是哪一位要看 (LOC A000: 0008H GOOD/BAD DATA FB/FFH) 中的 0008H 最后一位 16 进制数是偶数还是奇数来确定, 这里 0008 是偶数, 那么肯定无疑将是 L77 号芯片损坏, 换去该芯片, 主机运行正常。需注意的一点是: 有的内存卡诊断时, 将显示数字代码, 芯片的寻找方法同上。

3. 图形板

图形卡 RAM 是 4116 芯片, 每个芯片是 16K 位, 有 2 排, 每排 8 个芯片, 各芯片的位置及代码见表 3:

例 3: 故障现象

开机屏幕上有许多雪花点或竖杠杠, 影响操作人员视线诊断: 插入随机诊断软盘进入软盘驱动器。

屏幕显示: DATA DETACHED DATA=2000, 对应于 L19 号芯片换之, 屏幕恢复正常。

以上王安机 RAM 芯片的诊断及维修是笔者在维修中的探索和经验, 供各位读者参考。

表 3

RAM 位置	28	29	30	31	32	33	34	35
代码	0001	0002	0004	0008	0010	0020	0040	0080
RAM 位置	14	15	16	17	18	19	20	21
代码	0100	0200	0400	0800	1000	2000	4000	8000

IBM 286 维修一例

叶志斌

故障现象: 机器运行 dBASE III 程序过程中突然死机, 热启动启失效, 冷启动后显示 162——System Options Not Set—(RUN Setup)

(Resume="F1"key)

按 F1 键后 C 盘不能自举, 进入 ROM BASIC, 但软

盘仍可自举, 再运行 Setup 现象依旧。

故障分析: CPU 是 80286 的各类微机一般采用 "Setup" 软件以菜单的形式来选择系统配置的, 运行它则把所选的有关配置信息通过系统板上的 RT/CMOS RAM 端口地址写入这一 RAM 中, 一旦 RT/CMOS RAM 电路或 RAM 片子出现故障, Setup 的信息就读不出来, 或已丢失。再次运行 Setup 也写不进去信息, 而这类错误最常见的原因是给 RT/CMOS RAM 供电的干电池电压不足所致, 遇到这类问题应先测量干电池的电压, 若不足则更换之, 当你一时买不到, 可拉出两根引线, 在机箱外用两节高效 5 号电池取代, 看能否排除故障。

故障排除: 本例是电池电压不足, 更换新电池后, 运行 Setup, 故障排除。

中华学习机典型故障一例的分析

杨瑞华

故障原因:开机无“哗”声。屏幕闪一下,然后什么也没有,主机指示灯亮,同时按其它的键无任何反应。

分析:根据上述反应,可能引起故障的原因有:视(射)频、系统时钟影响整机工作。落实到具体器件是 Q_1 、 Q_2 、 B_2 等元件组成的主频振荡电路,有关的PAL电路及其它主要部件。如CPU、MMU、IOU、 U_{13} 、 U_{16} 、 U_{17} 等有关电路。

打开机盖,取下键盘,主机板上的电路及元件一清二楚。其功能分布很明了。检修工具用的是MODEL-101万用表。首先对CPU的控制总线进行直流电压的测量,(6502地址总线、数据总线,在读写时其引脚的电压不一至,它所提供的电压不能判断电路本身是否正常,所以可以不去测量地址和数据总线),测6502CPU,主要测1脚,3脚,8脚,37脚,39脚,21脚等,首先测量1脚 V_{SS} ,8脚 V_U ,21脚 V_{SS} 均正常。进一步测量与故障有关的脚,3脚,37脚,39脚。测得3脚

OUT Q_1 为零伏,实际应为2.4伏左右。39脚OUT Q_2 为4.6伏,正常值在2.2伏左右,3脚与39脚均是6502微处理器对输入的定时信号Q。经过处理后的输出定时信号。测37脚INQ。为4.6伏左右,正常值为2伏左右,用交流档测没有交流信号, Q_0 是6502微处理器的基本脉冲定时信号接入单,没有交流信号证明没有外来脉冲信号,故障可以比较肯定的确认为是PAL或者是晶体振荡部份。而6502的时钟信号由PAL电路中的 U_{17} 、晶体振荡部分由 Q_1 、 Q_2 、 B_2 等构成。

由于PAL电路是一块可编程逻辑部件,用来产生脉冲信号和控制信号,它是一个20脚封装的引线,查手册得知20脚为 V_{DD} ,10脚为 V_{SS} 。测得两脚电压正常,继续测1、2、3脚均正常。测14、15脚时发现14脚为零伏,而15脚又有4.6伏,查得正常时均为2.1伏左右。用万用表串电容测交流电压没有变化,这时将 U_{17} 拔下放在工作正常的机器上试验,开机正常,证明芯片没有问题,分析只有14脚没有脉冲信号。引起这个交流信号的可能在 Q_1 、 Q_2 、 B_2 晶振脉冲上发生问题。重点查找这三个元件及有关元件。用表测 Q_1 和 Q_2 发现 Q_2 的三个脚电压一样高。断开电源用欧姆档测,发现E脚与其它两脚不通,用小功率烙铁加吸锡器快速焊下,果然三极管已开路。相同换上同型号B的三极管,再开机一切正常。这个问题可能是管子本身的质量差引起的。

APPLE II 主机常见故障的检修

陈 鸿

APPLE II机在我国中小学中已相当普及,由于主机板的工作条件较恶劣,承受着浪涌冲击,温升高及操作不当等,使得集成块损坏率较高。笔者根据主机板的工作原理以及积累的实际维修经验,将主机板的常见故障整理归纳如下几条,供大家维修时参考。

APPLE II机主机板主要由6502中央微处理器ROM(只读存储器)、RAM(随机存储器),输入输出装置及产生视频文字及图像信号的线路等几部分组成。根据屏幕出现的故障现象,就能判断出故障的位置,从而把故障排除掉。

一、6502中央微处理器故障:

现象:开机后,指示灯亮,喇叭不响,无光标,屏幕无任何反应,不能复位,各种冷、热启动均无效。

二、RAM故障:

现象:1. 开机后,指示灯亮,喇叭不响,屏幕出现杂乱文字或黑白小方块,按键无效。此类故障多数为 C_3 — C_{10} (地址为\$0000—\$3FFF)的RAM。

2. 驱动器工作不正常。

3. 程序运行一会就停止。

4. 高解像页区图像有斑点不能消除。

5. 驱动器与接口卡完好,插入主机后,主机不能工作。

6. 开机后,直接进入监控状态,屏幕出现“*”的系统光标。

2—5条多数为 D_3 — D_{10} 或 E_3 — E_{10} (高地址\$4000—\$7FFF,\$8000—\$BFFF)RAM故障。

三、ROM故障:

现象:1. 开机后,没有任何反应,屏幕全白或全为“?”,按键无效。一般为第6个ROM,用2732的则为第3个ROM。

2. 开机后,直接进入监控状态。

3. 开机后,直接进入监控状态,按复位键后屏幕出现APPLESOFT的状态,但任何命令仍不能执行。

4. 开机后,执行程序时会出现错误信息。

以上2—4条多为第1—3个或第5个ROM。

(转第19页)

计算机基础知识(下)

王路敬

7. 目前编写计算机程序所用的语言可分哪几类? 各有何特点?

可分三类:

(1) 直接和机器打交道, 用计算机的指令表达的机器语言。这种语言是计算机硬件系统所能识别的、不需要翻译直接供机器使用的程序语言, 机器型号不同, 机器语言通常不同。机器语言中的每一条指令是一条二进制形式的指令代码, 该代码由操作码和地址码组成。机器语言程序编写和调试修改都比较麻烦, 但执行速度快。

(2) 用机器指令的助记符表达的汇编语言。这种语言是一种面向机器的程序设计语言, 不同系列的计算机其汇编语言不同。例如苹果机的 6502 汇编语言, IBM PC 系列机及其兼容机的 8088 汇编语言等。汇编语言是用助记符来代替操作码, 用地址符号来代替地址码的语言。所以汇编语言也叫“符号”语言。汇编语言与机器语言关系密切, 它的指令和翻译成的机器语言之间的关系基本上是一一对应的。这种语言好理解, 好记忆, 便于阅读, 而且保持了机器语言编程质量高、执行速度快, 占用内存空间小的优点。通常汇编语句由标号、操作码、操作数和注释四部分组成。使用标号可方便查询和修改, 便于转移指令的书写; 使用注释可增强程序的可读性。操作码和操作数一起构成一个机器指令, 指定计算机完成一个特定功能。操作码规定机器执行什么操作, 操作数提供执行操作时的数据或数据所在的地址。为了得到操作数需要了解寻址方式, 寻址范围和使用方法。不同的指令系统采用的寻址方式不同, 常用的有立即寻址、直接寻址、间接寻址、变址寻址等。汇编语言在系统开发、实时检测、实时控制、实时处理中发挥着巨大的作用。

(3) 独立于机器, 用不依赖于机器的具体指令表达的高级算法语言。例如 BASIC、PASCAL、FORTRAN 等。这种语言无需了解计算机的内部构造。使用高级语言编写程序方便, 易于查错、验证、阅读和修改; 同时由于高级语言编写的程序符合人们的习惯, 能自然地表达各种问题的有关概念, 所以可大大提高程序的可移植性和通用性。

目前世界上已有数百种高级语言, 用的最多的也有数十种, 其中 BASIC 语言是微型机上使用最普遍的一种高级语言。

使用高级语言要注意掌握语法和程序设计方法。一般高级语言的语法成分主要有基本元素(数、变量、串等), 表达式和语句。从功能上可分输入输出语句, 逻

辑运算和算术运算语句、程序控制结构语句和传输语句四类。只要掌握了基本语句的功能和使用, 再熟悉一些常用算法, 就可以编写一些实用程序。

除机器语言程序可以直接为机器所识别外, 汇编语言和高级语言编写的程序, 我们称之为源程序它必须经过“翻译”变成机器语言才能被机器识别。汇编程序把汇编语言源程序“翻译”成机器语言程序, 该过程叫汇编。高级语言源程序“翻译”成机器语言程序有两种方式: 一种是编译方式。这种方式是编译程序把高级语言源程序, 经过语言编译器, 统一地进行一次翻译工作, 同时也进行查错, 然后产生出一个目标码(机器码)组成的目标程序, 在这个过程中计算机不执行任何源程序内所规定的动作。使用编译程序对源程序进行编译时, 有的系统中形成的目标程序可直接执行, 但在 IBM PC 系统中, 目标程序不能直接运行, 还要将生成的以 .OBJ 为扩展名的目标程序经过连接程序生成以 .EXE 为扩展名的可执行文件后才能执行。第二种是解释方式。这种方式直接将源程序引入内存, 然后语言系统本身逐行地读取它, 通过解释器对其进行解释和查错。如果没有语法错误, 便根据解释结果, 把该行语句翻译成相应目标程序, 让机器去执行。

两种“翻译”方式各有优缺点。解释方式的缺点是执行速度慢, 优点是调试程序很方便。因为在解释方式下, 一般都是单步执行, 产生错误后便立即停止执行, 显示出错误信息。编译方式的优点是产生的独立的、可执行文件运行速度快, 保密性好。源程序经过编译、连接后得到的 .EXE 文件无须加密, 任何解密程序也不能破密而得到源程序。缺点是调试比较困难。

8. 在计算机内部采用什么数制? 这种数制有何特点?

在计算机内部一切信息包括数字、字母、字符和汉字等的存放, 处理和传送均采用二进制数的代码组合, 这是认识计算机的根本出发点。数在计算机中是以器件的物理状态来表示的, 一个具有两种不同的稳定状态且能相互转换的器件, 就可以用来表示一位二进制数。所以在计算机内用二进制数表示简单可靠, 二进制运算规则也是最简单的。因此, 目前在计算机中广泛采用二进制数来表示数字和进行运算。

二进制数具有两个基本特点: (1) 具有两个不同的数字符号即 0 和 1; (2) 逢二进位。由于是逢二进位, 所以同一个数字符号在不同的数位所表示的值是不同的。

例如 IIII. III

小数点左边的第 1 位的“1”代表的值就是本身,小数点左边的第 2 位的“1”是由第 1 位逢二进上来的,所以它的值为 1×2^1 ;则左边的第三位、第四位的值依次应当是 1×2^2 和 1×2^3 ;而小数点右边第 1 位的 1 则代表 1×2^{-1} ;右边第 2 位、第 3 位的值应当是 1×2^{-2} 和 $1 \times 2^{-3}, \dots$ 。

应当指出的虽然在计算机内部使用二进制数进行工作,但是,对于人来说,使用二进制是很不方便的。二进制数比起等值的十进制数来,位数要多得多,写起来,读写来也不方便。为此,人们通常用八进制和十六进制做为二进制的缩写方式。十进制数是人们通常习惯采用的计数制,也是人们最熟悉的计数制,要想用计算机处理十进制数必须要用二进制为十进制数编码并按二进制逻辑来实现十进制运算法则。同理,计算结果应从二进制数转换成十进制数再从计算机输出。这就产生了不同计算制之间相互转换的问题。

9. 在计算机中区别数的正负是如何规定的?

区别数的正负用符号位,符号位规定在数的最前面。用“1”表示负数;用“0”表示正数。也就是说,数的符号在机器中也数码化了。我们把一个数在机器中的表示形式叫机器数。而把这个数的本身叫真值。

例如:设有

$N1 = +1100111$

$N2 = -1100111$

则 $N1$ 和 $N2$ 在计算机中表示为:

$N1: 01100111$

$N2: 11100111$

带有符号的 $N1, N2$ 为真值;在机器中表示的 $N1, N2$ 为机器数。由于计算机硬件设备限制,机器数有位数限制。通常机器数位为 2 的倍数。

10. 在计算机中数有哪几种表示方法?

有三种表示法:原码、补码和反码。正数任何时候都用原码表示。只是负数可以用原码、补码或反码表示。

原码表示法是一种简单的机器数表示法。其符号位用数码 0 表示正号;用数码 1 表示负号。数值部分按一般二进制形式表示。

例如:

$N1 = +1001010$

$N2 = -1001010$

则

$[N1]_{\text{原}} = 01001010$

$[N2]_{\text{原}} = 11001010$

补码表示法规定:正数的补码和原码相同。负数的补码则先对原码除符号位外各位取反,然后末位加 1。

例如:

$NN = -1001010$

则

$[NN]_{\text{原}} = 11001010$

$[NN]_{\text{补}} = 10110101 + 1 = 10110110$

引进补码概念之后,加减法运算都可以用加法来实现,即用求“和”来代替求“差”,数的符号位也可以当做数值处理,一道参加运算,且真值的补码之“和”等于真值“和”的补码。这为加减法运算带来很多方便,因此,在近代计算机中加减法多采用补码运算。

反码表示法规定:正数的反码,或者负数的反码都是这个正数的原码或者这个负数的原码除符号位外按位求反得到。

例如:

$N1 = +1001010$

$N2 = -1001010$

则

$[N1]_{\text{原}} = 01001010$

$[N1]_{\text{反}} = 00110101$

$[N2]_{\text{原}} = 11001010$

$[N2]_{\text{反}} = 10110101$

反码通常做为求补过程的中间形式。

但是不论用哪一种方法表示数,当数的绝对值超过机器允许表示的最大值时,就要发生溢出,从而造成运算错误。例如 8 位机,即字长为 8 位,则最大值为 $(255)_{10}$;若超出此值,就会产生溢出。16 位机即字长为 16 位,则最大值为 $2^{15} - 1$;如果运算结果大于此值,就发生溢出。这个溢出刚好进到符号位,占据了符号位的位置,从而使结算结果发生错误。使用中应注意这个问题。

11. 十进制数与二进制数之间相互转换时应注意什么问题?

不同计数制之间的转换是根据如两个有理数相算,则两数的整数部分和分数部分一定分别相等的原理进行的。

十进制整数转换成二进制整数采用除 2 取余法;十进制纯小数转换成二进制小数采用乘 2 取整法;十进制混合小数转换为二进制数则将整数部分和纯小数部分按上述原则分别进行转换,然后再将其组合起来即可。二进制数转换为十进制数方法更为简单,只要将二进制数用计数制通用形式表示出来,相加计算出结果,便得到相应的十进制数。

不论是十进制数转换二进制数还是二进制数转换为十进制数,其整数部分和小数部分要采用不同的转换方法。不要误认为一个整数和一个小数形式一样,则转换后形式也一样。例如 10111 是十进制的 23,但 0.10111 却是十进制的 0.71875;19 是二进制数的 10011,但 0.19 却不是二进制数的 0.10011。这是第一点要注意的。

第二点,十进制小数不一定都能转换成完全等值的二进制小数,所以有时要取近似值。

从初等数学中我们知道:任何有限位的小数,都能用分数表示,但是任何一个分数却未必能用有限位的

小数表示,例如 $1/3$ 就是这样。所以两种数制的转换也存在类似情况。一个二进制小数,能够完全准确地转换成十进制小数,但是一个十进制小数,却时常不能完全准确地转换成二进制小数。例如十进制数的 0.1 就是这样。

不能用有限位的二进制小数去表示任一个有限位的十进制小数,这是二进制的缺点。但对一般科学计算,这个缺点是可以容忍的。因为科学计算或多或少都具有近似计算的性质。彻底解决这个问题,计算机多采用二进制编码的十进制数进行计算。即一位十进制数用四位二进制编码来表示,表示的方法很多,较常用的是 8421BCD 码。

12. 微机使用的西文字符是如何编码的?

微机使用的西文字符常用的编码是 ASCII 码。每个 ASCII 码由 8 个二进制位组成,它是计算机代码的最基本单位,即一个字节。一个字节表示的范围十进制数为 0~255。所以全部 ASCII 码最多可表示 256 种不同的字符,但在 ASCII 码中,并不把 256 种二进制数组合都作为字符的代码,而是在 ASCII 中,把二进制位的最高位为 0 数字称为基本 ASCII 码。它的范围是 0~127。即基本 ASCII 码共 128 个,其中 0~31 作为控制代码,32~127 为显示字符。控制代码在计算机中不当作字符显示,而是作为计算机进行某一特定的动作

的功能代码。例如代码 13 的功能是把光标移到行的最左端,代码 10 的功能是使光标移到下一显示行等等。32~127 共 96 个编码是字符码,其中包括英文字母的大小写一共 52 个。供书写程序和描述命令用。

在 ASCII 码中,把二进制位的最高位为 1 的数字称之为扩充 ASCII 码,它的范围 128~255。扩充 ASCII 码也是 128 个。各国都把扩充 ASCII 码规定为自己国家语言的字符代码。我国就把扩充的 ASCII 码作为汉字的代码。一个汉字的机内码就是由两个扩充的 ASCII 组成,这两个内码的选取与汉字区位码的规则相关。

13. 什么是汉字的外码?

所谓汉字的外码是指计算机与人进行交换的字形符号的一种代码。它与汉字的输入方式有直接关系。同一个汉字,选择不同的输入方式,其汉字的外码不同。例如“啊”,在区位码输入方式下其外码为“1601”;拼音码输入方式下,其外码为“a”;而五笔字型输入方法,其外码为“kbsk”。汉字输入的过程就是外码转换为内码的过程,即将汉字或一些符号的外码键入计算机后,计算机把它转换为可识别的内码后再存储内存。而汉字的输出过程则是内码转换为外码的过程,即计算机把内存中的内码转换为约定字型后输出到显示器或打印机。

(接 46 页)

硬件配置:

Turbo CAD 要求至少配置下列硬件与操作系统:

1. IBM PC/XT、AT 及其兼容机
2. PC DOS(或 MS-DOS)操作系统 2.0 以上版本
3. 内存(至少 256K)
4. 360K 软盘驱动器 2 个
5. 图形显示器。

以上是所需的基本配置,而为了提高 Turbo CAD 程序的运行效率,应该增加下列硬设备。

1. 增加硬盘容量;可改善整个操作系统环境,加快文件输入/输出速度,免除抽换软盘的麻烦、能利用 DOS 子目录来配置图形文件的存储空间。

2. 输入设备:Turbo CAD 的标准输入设备是键盘、同时也可利用其它输入设备、如鼠标器、数位板等。这些可在 Turbo CAD 中设定所需的输入设备。

3. 打印机:Turbo CAD 能把图形文件输出给许多不同的打印机进行打印。通常、打印机的分辨率比屏幕的高,如果所使用的打印机在 Install 程序中并不列出、可配置兼容的打印机。

4. 绘图机:如要得到质量更佳、分辨率更高的图

形,最好使用绘图机。Turbo CAD 提供了许多机型的绘图机供选用。

5. 系统存储器:Turbo CAD 可以完全利用系统内存 640K,如果用户在图形中加上更多的图素,Turbo CAD 就会用到更多的内存;如果绘制的图形很复杂,则占用内存可能超过 256K(在任何时候,Turbo CAD 绘图屏幕左下角的状态区内均会显示出内存的剩余空间)。若系统配置有内存扩充卡、可将其设定为随机存取磁盘、以减少磁盘存取时间,加快绘图速度。

以上是 Turbo CAD 系统运行的软、硬件环境。读者可根据各单位的具体情况,加以选用。

购书消息

《电子与电脑》90 年合订本已出版发行,每本定价 13.5 元(另加邮费 15%)。需要者可汇款至北京 173 信箱,电子工业出版社发行部,邮购科 邮政编码 100036



Turbo CAD 软件简介

新书与软件

Turbo CAD 是美国 Borland International Inc. 推出的能在 IBM PC/XT 及其兼容机上实现的一种崭新的计算机辅助设计系统。它是以主菜单(主功能表)和下拉式菜单(下拉式功能表)的功能选择项提供各种绘图命令,它的绘图功能完整,开发环境良好,使用方便,执行速度快,拥有 128 层,100 种线型,255 的线宽以及 32 种箭头和屏幕计算器供用户绘图与计算使用,其绘图精度很高,能任意设定绘图比例,能任意填图,能自动标示尺寸,能测量、画图与计算,且能自动计算面积,长度,周长等。能注意标示所需字型大小(5 种)、颜色(16 种)的文字;能对图形进行任意放大、缩小、旋转,映射与拉长;能对图形进行任意删除、增添、移动与拷贝;能将图形输出至绘图机或打印机进行绘制或打印。

Turbo CAD 可用于绘制各式各样的图形,适用于诸如电子、电气、电器、土木、建筑、机械、造船、航空、化工、纺织、美术、服装设计等各行各业的计算机辅助设计,应用范围十分广泛。

Turbo CAD 的软件配置与硬件要求

• 软件配置: Turbo CAD 共有系统盘 2 片、其内容包括 Turbo CAD 程序、支援程序和文件。

第 1 盘的内容为:

TURBOCAD COM	368244-17-88	7:57p
TURBOCAD 000	40961-03-88	5:57p
TURBOCAD 001	74243-02-88	
TURBOCAD 002	7683-02-88	
TURBOCAD 003	107523-02-88	
TURBOCAD 004	20483-02-88	
TURBOCAD 005	33283-02-88	
TURBOCAD 006	17923-02-88	
TURBOCAD 007	10243-02-88	
TURBOCAD 008	66563-02-88	
TURBOCAD 009	30723-02-88	
TURBOCAD 010	97283-02-88	
TURBOCAD 011	197123-02-88	
TURBOCAD 012	87043-02-88	
TURBOCAD 013	35843-02-88	
TURBOCAD 014	23043-02-88	
TURBOCAD 015	94723-02-88	
TURBOCAD 016	660483-02-88	

TURBOCAD 017	28163-02-88	
TURBOCAD 018	281603-02-88	
TURBOCAD 019	53763-02-88	
TURBOCAD 020	122883-02-88	
TURBOCAD 021	61443-02-88	
TURBOCAD 022	71683-02-88	
TURBOCAD 023	46083-02-88	
TURBOCAD 024	166403-02-88	
TURBOCAD 025	112643-02-88	
TURBOCAD 026	120323-02-88	
TURBOCAD 027	23043-02-88	
TURBOCAD 028	43523-02-88	
TURBOCAD 029	84484-17-88	7:57p
TURBOCAD 030	30723-02-88	
TURBOCAD 031	135683-02-88	
TURBOCAD 032	25603-02-88	
14X9 FON	35842-04-88	
4X6 FON	2852-04-88	
LINETYPE LIN	502-04-88	
NORMAL ARO	432-04-88	
NORMAL DRV	10081-01-80	12:06a
TC DRV	1701-01-80	12:05a
CONFIG CAD	1561-01-80	12:15a
41 File(s)		0 bytes
(s)		free

第 2 盘的内容为:

TURBOCAD	OVL	11360	8-29-88	3:47p
TURBOCAD	HLP	20252	3-02-88	
INSTALL	COM	60788	3-02-88	
COMPLEX	FNT	11260	3-02-88	
ITALIC	FNT	12468	3-02-88	
NORMAL	FNT	4428	3-02-88	
SIMPLEX	FNT	6828	3-02-88	
TXT	FNT	4016	3-02-88	
TCUTIL	COM	56636	3-02-88	
MASTER	DRV	17388	9-26-88	1:30p
BILLOFM	PRG	6311	3-02-88	
READ	ME	2176	1-01-80	12:02a
README	BAT	24	1-01-80	4:59a

13 File(s) 141312 bytes free

(转第 45 页)

巧用电脑作幻方

郭高惠

电脑的出现离不了数字,处理数字的能力自然也就无需置疑了。将其投入数字海洋,并利用那些满足要求的数字,为我们拼造出一个个赋有魔力和趣味的数字方阵——幻方,正是本文的目的所在。

所谓幻方,其实是指由 1 到 n^2 个连续自然数排列成的方阵;它有 n 行 n 列,并且每行上 n 个数字之和与每列上 n 个数字之和相等,同时还与两条对角线上的 n 个数字之和相同,这个和可由式 $n(n^2+1)/2$ 确定; n 就称为该幻方的阶(或称 n 阶幻方)。

幻方的构造虽然遵循某种规律,但要靠人工来构造(特别是高阶幻方)并不那么容易办到,所以最好还是由我们手头的电脑来代劳。为此,笔者设计了下面的幻方制作程序可在一般家用电脑上直接运行供大家使用,利用它可以得到任意阶数的幻方。

由于奇阶幻方与偶阶幻方(偶阶中又将阶数能被 4 整除的幻方另归一类,即 4 的倍数阶幻方)的构成规律不同,所以在程序中使用了不同的语句段。下面就对程序略作说明。在该程序中,用变量 I、J 分别作为幻方的行、列坐标值;G 用于区别幻方类型(即奇或偶幻方)。N 是幻方的阶数,程序运行前会提示你从键盘输入。S 数组用来存放幻方数据,其它变量是一些辅助临时性变量。

70 语句定义数组为输出幻方作准备;

90~110 语句用于完成 4 的倍数阶幻方的构造。

150~180 语句用于完成其它情况的幻方设计。

185~190 语句为输出整齐美观的幻方作准备。

195~220 语句用于幻方的输出打印。如果是偶阶幻方,就不能直接输出,要转 300 语句作处理后再输出。300~350 语句可按给定规律对偶阶幻方进行合成并完成输出。

230~240 语句用于判断是否还要继续运行。

LIST

10 CLS' 这是一个求取 N 阶幻方的实用程序

40 PRINT;INPUT"请输入阶数 N[N>2]";N

50 PRINT:I=1:P=1:G=N AND 3

55 IF N<3 THEN PRINT"阶数小于 3. 重新输入.";RUN
40

60 L=N/(1+ABS(G=2));J=INT(L/2+.5);IF G=0
THEN J=1

70 DIM S(L,L),FOR T=1 TO L*L:R=INT(T/L)

80 S(I,J)=T;IF G>0 THEN 150

90 IF T/L=R THEN I=I+1;J=(1+L)*SGN(I AND
2)

100 P=1-P;IF P=0 THEN I=L-I+1

110 J=J+1-(R+1 AND 2);GOTO 180

150 IF T/L=R THEN I=I+1;GOTO 180

160 I=I-1+ABE(I=1)*L;

170 J=J+1-ABS(J=L)*L

180 NEXT T:T=T-1:P=LEN(STR\$(T));K=P*N/2-8

185 K=ABS(N>6)*K+1;FOR I=1 TO P:Y\$=Y\$+
CHR\$(35)

190 NEXT I:K=K+ABS(N>L)*N/2;PRINT TAB(K);

195 PRINT"下面是";N;"阶幻方";IF N>L THEN 300

200 FOR I=1 TO N:FOR J=1 TO L

210 PRINT USING Y\$;S(I,J);NEXT;PRINT;NEXT

220 PRINT TAB(K);"其特征值为:";(N*N+1)/2*N

230 PRINT;INPUT"还想打印请敲 Y 键";Y\$

240 IF Y\$="Y" OR Y\$="y" THEN RUN 40 ELSE END

300 R=INT(L/2)+1;FOR S=1 TO N:I=S-ABS(S>L)*
L

310 FOR P=1 TO N:J=P-ABS(P>L)*L;G=(J>1 OR I
<>R)

320 G=3*T*(ABS(S>L)<>G*(J<R)+(I=R)*(J
=R))

330 IF P>L THEN G=T+ABS((S>L)=1+J<R))*T

340 PRINT USING Y\$+"#";S(I,J)+ABS(G);;

350 NEXT;PRINT;NEXT;GOTO 220

RUN

请输入阶数 N[N>2]? 5

下面是 5 阶幻方

17 24 1 8 15

23 5 7 14 16

4 6 13 20 22

10 12 19 21 3

11 18 25 2 9

其特征值为:65

还想打印请敲 Y 键? Y

请输入阶数 N[N>2]? 4

下面是 4 阶幻方

1 14 15 4

8 11 10 5

12 7 6 9

13 2 3 16

其特征值为:34

还想打印请敲 Y 键? Y

请输入阶数 N[N>2]? 6

下面是 6 阶幻方

35 1 6 26 19 24

3 32 7 21 23 25

31 9 2 22 27 20

8 28 33 17 10 15

30 5 34 12 14 16

4 36 29 13 18 11

其特征值为:111

还想打印请敲 Y 键? N

OK

湖北地区中华学习机联谊活动情况

本刊讯：湖北省中华学习机应用沙龙是湖北地区长期坚持中华学习机普及活动的一个群众团体，沙龙的联合发起单位有湖北省青少年科技活动中心，长江日报社，武汉市青少年宫、武昌区青少年计算机协会、湖北省电子器材物资公司电脑开发部，沙龙的具体活动由市青少年宫和武昌青少年计算机协会负责。联合这些发起单位的目的是便于解决沙龙需要的活动阵地、宣传报导、骨干力量(包括活动积极分子和技术力量)的组织、器材经费的筹措等一系列问题。

我们理解的沙龙，就是计算机爱好者的俱乐部，它应该不以赢利为目的，宗旨就是为广大青少年和拥有中华学习机的个人服务，组织他们切磋电脑应用经验，交流学习体会，互相交换各类软件，提供各种服务：组织系列软件讲座，提供资料和信息，帮助解决用户在使用中遇到的各种困难，组织会员与生产厂家对话等一系列活动。我们开展的这些活动博得了广大计算机爱好者的欢迎和支持，沙龙从去年(八九年)六月成立至今我们长期坚持活动。会员也是来自四面八方，有工人、学生、机关干部、教师、工程师，会员里面有十几岁的娃娃，也有退休的老人，有为了增加自己的知识面而

来的，也有是为了教孩子先来当学生的，甚至连远在湖北荆门炼油厂的同志也找上门来要参加我们的活动，鄂西自治州的同志来信索取资料。

活动的形式也不是一成不变的，沙龙成立之初，我们采取上大课办讲座的方式，这种方式对一部分初学者可能很有帮助，但对另外一些人如有些已涉足硬件开发的同志显然是不能满足他们的要求，这时候我们从沙龙活动的积极分子中组织了一批骨干，成立会员常委会，由这些常务会员开会研究确定每月活动的具体内容，每个常务会员都有宣讲义务。为了照顾不同层次的要求，改变过去一次活动一个专题，单打一的方式，每次安排三个内容(兼顾普及和提高)，每个专题尽量短小精干，言犹未尽的会后下再和有兴趣的一部分同志进一步探讨。这种做法深得广大会员的赞赏。另外沙龙还同社会上其它一些组织如湖北省电子学会联合举办一些讲座，以提高沙龙的知名度，让更多的计算机爱好者参加到我们沙龙里面来。

沙龙成立至今仅仅一年多，虽然做了一些工作，但也存在不少问题，它毕竟是一种新事物，没有可循的常规，要靠我们不断总结完善，使其更有生命力。

浙江省现代微电子技术发展公司国内最优价推出

计算机类 万国 WG-900 超级多功能办公机集电脑、轻印刷系统、打字机于一身，融文字处理、传真、联网功能为一体，首批办公机在浙江省委宣传部等机关、企事业单位中运行。基本配置 Super PC/XT 加 M1724 打印机。CPU NEC V20 12MHz RAM 640KB 硬盘 20MB 软驱 360K×2, 101 键标准键盘，14" 双频直角平面显示器，国际流行机箱 11000 元/套。Super PC/XT 5200 元/套 Super PC 3400 元/套 中华机 CEC-1 790 元—950 元/台(陕产)CEC-PC/10 3950 元/套 AST 286(140) 18500 元/套 AST 386 SX 21000 元/套 供应各类进口驱动器、打印机、PC、中华机软件。

单片机类 MCS-51 标准型单片机学习开发系统 8031 芯片，时钟频率 6—12MHz RAM 6264 EPROM 27256 EEPROM 2864 两路 16 位定时/计数器；可编程 I/O 为 8255 提供三个并行口，可直接驱动打印机或与 IBM PC 并行口连接；八路 8 位 A/D 0809 一路 8 位 D/A 0832。该系统用读者熟悉的 Z80 单板机模式开发单片机，同时为用户提供了—个完整的数据采集、处理及控制系统。适宜大中专院校、科研、工矿企业学习、开发 MCS-51 系列单片机及作为单片微机函授班教学、实验用机，散件 325 元/套 整机 375 元/套 附有实验指导书一套。

集成电路类 特价供应各种规格进口集成电路 8031 15 元，6264 24 元，2864 80 元，27256 18 元，8255 12 元，0809 18 元，0832 18 元，8279 18 元。

电脑学习机类 科特 FCS-90 电脑学习机 电子游戏与家用电脑两机合一，具有中小学阶段 BASIC 程序、幼儿教育、英文打字、绘图、电子琴演奏、自编节目、数据读/写及电子游戏

八大功能 675 元(上海市场价 750 元、北京市场价 725 元)。

电子游戏机类 小天才游戏机 IQ-701 豪华型 675 元，IQ-501 实用型 445 元，IQ-301 经济型 395 元，IQ-201 普及型 375 元。胜天 9900 豪华型 440 元，9000 高档型 390 元，8800 普通型 340 元。天马牌 TM939 型 245 元，TM828 型 225 元，TM737 型 205 元。最新面市日本原装世嘉五代立体游戏机、立体场景、形象逼真、具有大型游艺机效果 965 元。

游戏卡类 超级玛莉、坦克大全 24 元，魂斗罗 30 元、魂斗罗二代 40 元、空中魂斗罗 60 元、赤影战士、星际魂斗罗、四合一强卡均 70 元，八合一强卡 130 元，31 合一 95 元，42 合一 105 元，52 合一 135 元，63 合一 155 元，110 合一综合大全 315 元。世嘉五代立体游戏机适用卡单卡 95 元，二合一卡 135 元等百余种。

电子游戏指南(随卡手册)国内第一本详细介绍任天堂、电脑学习机、世嘉五代立体游戏机及大型游艺机 100 种最受欢迎的游戏节目，包括故事梗概、基本打法及攻关秘诀。50 万字与上下两册，邮购每套 10 元，购机一台或购卡貳盒赠书一册。电子游戏指南月刊(内部资料)每期 2 元，介绍最新游戏卡节目及近期机卡报价。接受九一年订户。

游戏机专用配套电脑学习键盘(BASIC 语言、人机对话、计算、学习英语、作曲等)290 元/台。

快译通：中英文双向翻译电子字典，家庭、企业、学生、科技、涉外人员理想的随身翻译工具，每台 380 元。

索要各种资料、集成电路、游戏卡目录及报价单请附邮费 0.4 元。

邮购地址：杭州环城北路 57 号三立大厦南二楼

邮编：310006 电话：574439—384 联系人：谭启仁