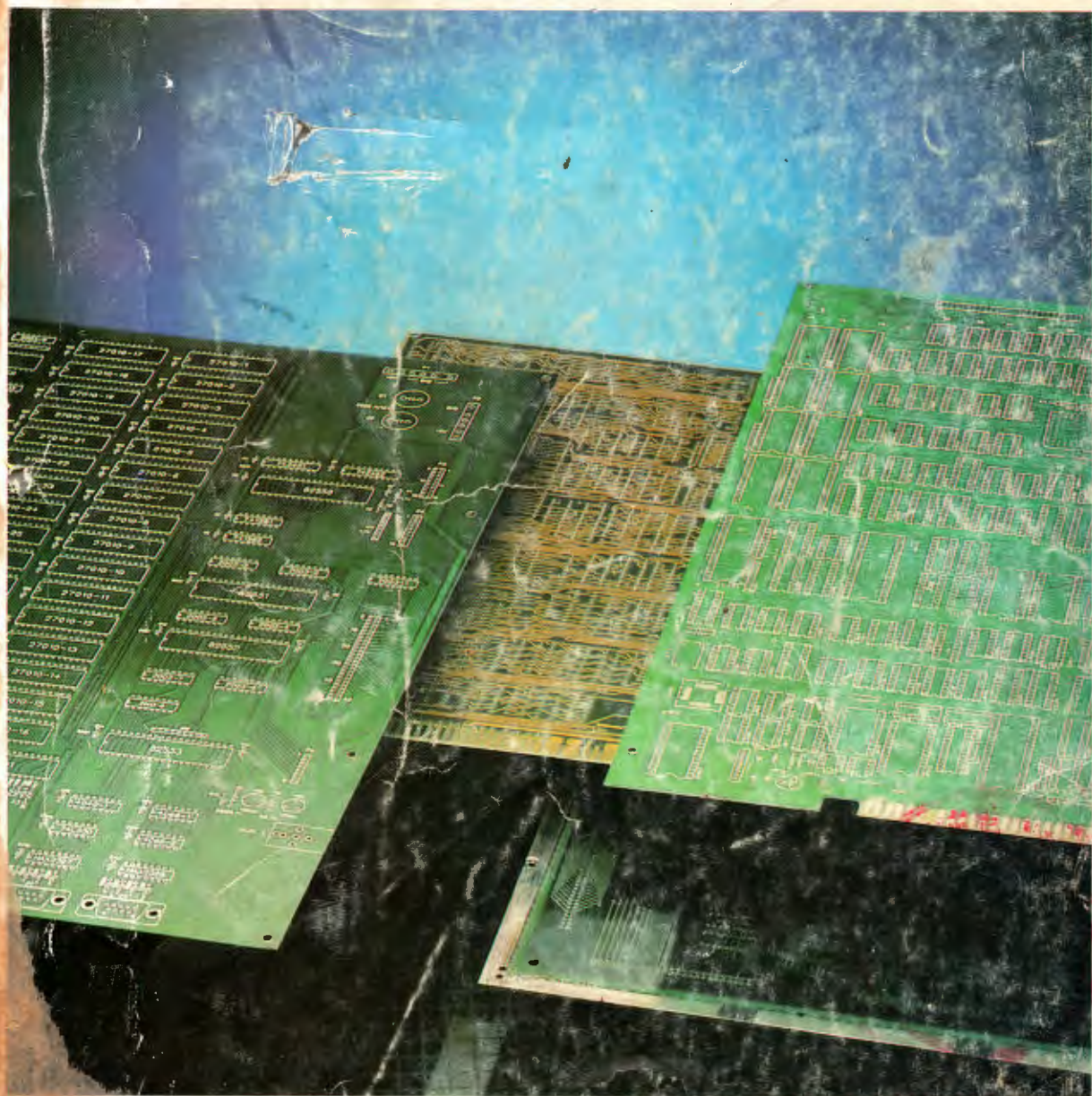


電子

ISSN 1000-1077
479

第88期

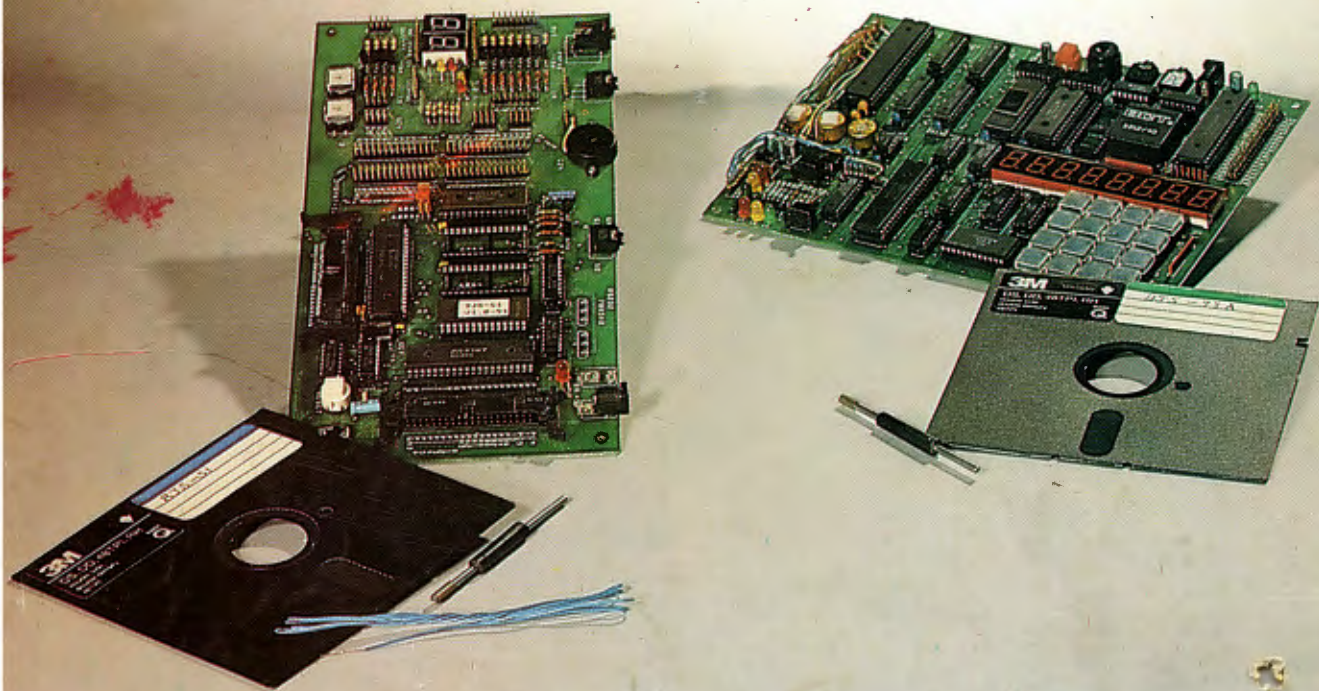
與 電腦



ELECTRONICS AND COMPUTERS

北京市单片机应用技术协会 (BJDP)

促进单片机应用 开发/应用板齐全 技术讲座连载 优惠提供教具
邮购价: BJS-51(495元/套) BJS-98(895元/套) BJS-98A(1195元/套)



说明1: BJS-51每套含说明书三本、绕线工具(申请专利)二支、软盘片一张、联PC机电缆一条(脱机调试板190元)。

2: BJS-98每套含说明书二本、绕线工具二支、软盘片一张、联机电缆一条。

3: BJS-98A每套含说明书二本、绕线工具二支、软盘片三张(PLM语言、窗口调试)、联机电缆一条。

4: BJS系列单片机教学实验系统是开放型结构。欢迎师生们不断充实、完善各种实验(协会计划汇集成册)。

选件: ①电源(48元/个) ②说明书(每套20元) ③工具(20元/支) ④51扩展板(190~390元/块)

经销:北京市技术交流站 地址:北京市虎坊桥13号 联系人:何维才 邮编:100052 电话:3035931-2159

经销:北京市电器仪表技术服务公司 地址:北京市西城三里河西口18号楼 (13路汽车终点站下车即是) 邮编:100045
电挂:2941 联系人:张文奇 电话:8012947

邮购:北京市源隆电子技术公司 地址:北京市北太平庄路1号 邮编:100088 电挂:5647 联系人:景平 电话:2019391
开户行:北京工商银行北太平庄分理处 帐号:067027-46

邮购:北京三环电子技术公司 地址:北京市中关村路58号 邮编:100080 (北京2740信箱) 联系人:刘桂芬
电话:2543039 开户行:北京海淀双榆树信用社 帐号:003663-89

代销:北京海声工业控制系统工程公司 地址:北京市中关村路口南 联系人:张寅 电话:2554603

(上接封二)

型 号	工 艺	管 脚	RAM (字节)	ROM (字节)	EPROM (字节)	EEPROM (字节)	I/O	定时器 (位)	SPI	SCI	A/D	封 装
68HC05A6	HCMOS	40/44	176	4160	—	2056	32	16	Yes	Yes	—	P, FN
68HC05B4	HCMOS	48/52	176	4160	—	—	32	16	—	Yes	Yes	P, FN
68HC05B6	HCMOS	40/52	176	5952	—	256	32	16	—	Yes	Yes	P, FN
68HC05C2	HCMOS	40	176	2096	—	—	32	16	—	—	—	P
68HC05C3	HCMOS	40	176	2096	—	—	32	16	Yes	Yes	—	P
68HC05C4	HCMOS	40/44	176	4160	—	—	32	16	Yes	Yes	—	P, FN
68HC05C8	HCMOS	40/44	176	7700	—	—	32	16	Yes	Yes	—	P, FN
68HC05L6	HCMOS	68	176	6208	—	—	32	16	Yes	—	—	FN
68HC05M4	HCMOS	52	128	4K	—	—	32	8/16	—	—	Yes	FN
68HCL05C4	HCMOS	40/44	176	4160	—	—	32	16	Yes	Yes	—	P, FN
68HCL05C8	HCMOS	40/44	176	8K	—	—	32	16	Yes	Yes	—	P, FN
68HSC05C4	HCMOS	40/44	176	4160	—	—	32	16	Yes	Yes	—	P, FN
68HSC05C8	HCMOS	40/44	176	8K	—	—	32	16	Yes	Yes	—	P, FN
68HC705C8	HCMOS	40/44	304	—	—	—	32	16	Yes	Yes	—	P, FN
68HC805B6	HCMOS	48/52	176	—	8K	6208	32	16	—	Yes	—	P, FN
68HC805C4	HCMOS	40/44	176	—	—	4160	32	16	Yes	Yes	—	P, FN
146805E2	CMOS	40	112	0	—	—	16	8	—	—	—	P, S, FN
146805F2	CMOS	28	64	1089	—	—	20	8	—	—	—	P, S, FN
146805G2	CMOS	40	112	2106	—	—	32	8	—	—	—	P, S, FN

四、M68HC11 系列 8 位 HCMOS 单片微机。4~12K ROM, 192~512 RAM。30~38 条 I/O 线。SCI/SPI 接口。A/D 变换器。

型 号	工 艺	管 脚	RAM (字节)	ROM (字节)	EEPROM (字节)	I/O	定时器 (位)	总线存储器	A/D	SPI	SCI	封 装
68HC11A0	HCMOS	48/52	256	—	—	38	16	64K	Yes	Yes	Yes	P, FN
68HC11A1	HCMOS	48/52	256	—	512	38	16	64K	Yes	Yes	Yes	P, FN
68HC11A8	HCMOS	48/52	256	8192	512	38	16	64K	Yes	Yes	Yes	P, FN
68HC11D3	HCMOS	40/44	192	4096	—	30	16	64K	No	Yes	Yes	P, FN
68HC11E1	HCMOS	52	512	0	512	38	16	64K	Yes	Yes	Yes	FN
68HC11E9	HCMOS	52	512	12K	512	38	16	64K	Yes	Yes	Yes	FN
68HC11F1	HCMOS	—	—	—	—	—	—	—	—	—	—	—
68HC811E2	HCMOS	48/52	256	—	2K	38	16	64K	Yes	Yes	Yes	P, FN

五、一次编程(OTPROM)单片微机

型 号	OTPROM (字节)	RAM (字节)	I/O	定时器 (位)	A/D, SCI SPI	监督 定时器	封 装
68HC704P4	3740	124	20	8	—	—	28-DIP, DW
68HC705B5	6208	176	24	16	A/D, SCI	Yes	52-FN, 48-DIP
68HC705C4	4160	176	24	16	SCI, SPI	Yes	44-FN, 40-DIP
68HC705C8	7616	304	24	16	SCI, SPI	Yes	44-FN, 40-DIP
68705R3	3776	112	24	8	A/D	—	40-P
68HC711D3	4096	192	24	16	SCI, SPI	Yes	44-FN, 40-DIP
68HC711A8	8192	256	38	16	A/D, SCI, SPI	Yes	52-FN
68HC711E9	12K	512	38	16	A/D, SCI, SPI	Yes	48-DIP, 52-FN

MOTOROLA公司

单片微机一览表 (资料)

梁合庆 供稿

一、M6801 系列 8 位 HMOS 单片微机。2K~4K ROM, 128~192 RAM, 13~29 条 I/O 线。SCI 串行口。

型 号	工 艺	管脚	RAM (字节)	ROM (字节)	EPROM (字节)	I/O	定时器 (位)	总线存储器	SCI	封装
6801	HMOS	40	128	2048	—	29	16	64K	Yes	P,S
68701	HMOS	40	128	—	2048	29	16	64K	Yes	S
6803	HMOS	40	128	—	—	13	16	64K	Yes	P,S
6801U4	HMOS	40	192	4096	—	29	16	64K	Yes	P,S
68701U4	HMOS	40	192	—	4096	29	16	64K	Yes	S
6803U4	HMOS	40	192	—	—	13	16	64K	Yes	P

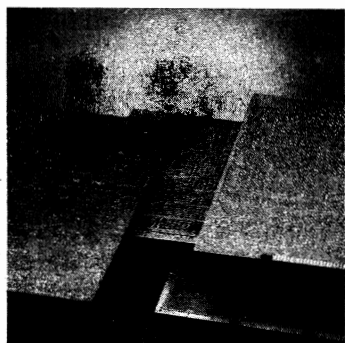
二、M6804 系列 8 位 HMOS/HCMOS 单片微机。512~3K ROM, 30~172 RAM. 12~20 条 I/O 线。

型 号	工 艺	管 脚	RAM(字节)	ROM(字节)	EPROM(字节)	I/O	定时器(位)	封装
6804P2	HMOS	28	30	1016	—	20	8	P, FN
68704P2	HMOS	28	30	—	1020	20	8	S
6804J1	HMOS	20	30	512	—	12	8	P
6804J2	HMOS	20	30	1000	—	12	8	P
68HC04P4	HCMOS	28	172	3700	—	20	8	P
68HC04J2	HCMOS	20	30	1000	—	12	8	P
68HC04J3	HCMOS	20	122	1672	—	12	8	P
68HC704P4	HCMOS	28	172	—	3700	20	8	S

三、M6805 系列 8 位 HMOS/HCMOS/CMOS 单片微机。4K~7K ROM, 64~304 RAM. 16~32 条 I/O 线。SCI/SPI 接口。A/D 变换器。

型 号	工 艺	管 脚	RAM(字节)	ROM(字节)	EPROM(字节)	I/O	定时器(位)	A/D	SPI	封装
6805P2	HMOS	28	64	1110	—	20	8	—	—	P,S, FN
6805P6	HMOS	28	64	1804	—	20	8	—	—	P,S, FN
68705P3	HMOS	28	112	—	1804	20	8	—	—	S
68705P5	HMOS	28	112	—	1804	20	8	—	—	S
6805R2	HMOS	40/44	64	2048	—	32	8	Yes	—	P,S, FN
6805R3	HMOS	40/44	112	3776	—	32	8	Yes	—	P,S, FN
68705R3	HMOS	40	112	—	3776	32	8	Yes	—	S
68705R5	HMOS	40	112	—	3776	32	8	Yes	—	S
6805S2	HMOS	28	64	1480	—	21	8	Yes	Yes	P,S, FN
6805S3	HMOS	28	104	2720	—	21	8	Yes	Yes	P,S, FN
68705S3	HMOS	28	104	—	3752	21	8	Yes	Yes	S
6805U2	HMOS	40/44	64	2048	—	32	8	—	—	P,S, FN
6805U3	HMOS	40/44	112	3776	—	32	8	—	—	P,S, FN
68705U3	HMOS	40	112	—	3776	32	8	—	—	S
68705U5	HMOS	40	112	—	3776	32	8	—	—	S

(下转三)



• ELECTRONICS AND COMPUTERS •

电子与电脑

一九九二年

总期第88期

目 录

• 综述 •

优越的专家系统语言 Prolog 王粤宁(2)

• PC 用户 •

结构文件与 FIELD() 函数的比较

..... 严桂兰 刘甲耀(4)

24 小时服务系统的人员安排 陈君佐(6)

ARC 系列压档工具软件的应用 戴青松(7)

DOS 3.30 若干新增命令的功能与应用 柳见成(9)

WPS 的彻底解密 瞿新国(10)

处理系统不认硬盘时应注意的问题 陈 栋(11)

• 学习机之友 •

CEC-I 彩色 TV 字幕 包 敢(12)

BASIC 程序中变量名使用情况的列表印出程序

..... 蔡 伟(14)

磁头清洗及其辅助程序 唐汉雄(15)

求法雷数列的新方法 廖庆平(16)

TOOL-KIT 使用详解 姜 宏(17)

• 语言讲座 •

6502 机器语言程序设计

第七章 循环程序设计 朱国江(20)

• 初级程序员级软件水平考试辅导 •

PC BASIC 自测试题解答与分析 李志刚(25)

• 学用单片机 •

BJS-51 单片机实验系统 盛焕鸣(29)

• 学装微电脑 •

步进电机的控制 易齐千(33)

• 电脑巧开发 •

ASCII 码字符显示器的设计 李德文(38)

• 电脑游戏机 •

第三章 F BASIC 的画面控制语句 于 春(41)

• 维修经验谈 •

Super AT 机显示电源原理及维修

..... 范志盛(45)

利用 PCTOOLS 校正软盘驱动器磁头 黄焕如(46)

• 读者联谊 •

普及型 PC 个人用户软件交流联谊活动问题解答(七)

..... 王路敬(48)

封一: 线路板

封二: MOTOROL 公司单片微机一览表

封三: 同上

封四: BJS 系列单片机教学实验系统

机械电子工业部电子工业出版社主办

编辑、出版:《电子与电脑》编辑部

(北京 173 信箱 邮政编码: 100036)

印刷: 北京三二〇九厂

国内总发行: 北京报刊发行局

国内统一刊号: CN11-2199

邮发代号: 2-888

国外代号: M924

出版日期: 每月 23 日

主编: 王惠民 副主编: 王昌铭

责任编辑: 杨逢仪

订购处: 全国各地邮电局

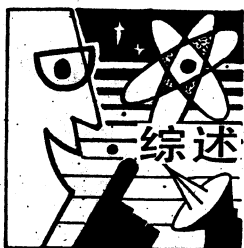
国外总发行: 中国国际图书贸易总公司

(北京 399 信箱 邮政编码 100044)

广告经营许可证: 京海工商广字 147 号

定价: 0.95 元

92 065



优越的专家系统语言 Prolog

空军航空工程部(410124)王粤宁

专家系统是人工智能中最活跃的组成部分之一。专家系统是具备相当丰富和权威性的知识(知识库)、采取一定的推理策略,为解决具有专家级的适当规模的实际问题的计算机系统。专家系统还具有学习机制,可对知识库进行改进,以增进解决问题能力。知识库和推理机制是构成一个专家系统的最核心部分。由于 Prolog 语言具有强大的推理功能,已经作为构建专家系统最自然的工具。用 Prolog 语言编写的专家系统与一般程序设计语言和数据库比较,在许多方面体现出明显的优越性。

与一般程序设计语言的比较

专家系统可以表示为:“知识+推理=系统”,较之“数据+算法=程序”的一般应用程序有观念性改变。以下通过对程序一与程序二进行的比较,可以看到专家系统语言 Prolog 的优越性。程序一是用 Prolog 语言编写的一个安排某班九岁同学乒乓球赛的微型专家系统。它由事实、规则两大部分构成。程序二是实现同一功能用 BASIC 语言编写的程序。

程序一:predicates

```
pupil(symbol,integer)
clauses
pupil(Peter,9)
pupil(Chales,9)
pupil(Paul,10)
pupil(Susan,9)
match(P1,P2):-pupil(P1,9),pupil(P2,9),
P1<>P2,write(P1,P2),fail.
```

程序二:10 INPUT N

```
20 DIM P$(N)
30 J=0
40 FOR I=1 TO N
50 READ X$,Y
60 IF Y<>9 THEN 90
70 J=J+1
80 P$(J)=X$
90 NEXT I
100 FOR K=1 TO J
110 FOR L=K+1 TO J
120 PRINT P$(K),P$(L)
130 NEXT L
140 NEXT K
150 DATA Peter,9,Charles,9,
Paul,10,Susan,9
160 END
```

程序二的框图见后页。

1. 专家系统语言 Prolog 与一般程序设计语言解决

• 2 •

问题的方式有着根本的差异。一般程序设计语言必须为待解决的问题设计算法,才能由系统实现。在用 BASIC 语言编写的程序二中,为了解决确定比赛选手配对问题,预先设计了算法:挑出有资格参赛的选手,将选手两两配对。但 Prolog 语言不需要预先设计解决问题的算法,只要在程序中给出系统目标,利用系统内部的模式匹配,回溯搜索等功能,便可自动寻求问题的解答。比如在用 Prolog 语言编写的程序一中,只要给出目标:将有资格参赛的不同两选手配对,系统便可给出答案。Prolog 语言的这一特点根本改变了一般程序设计语言进行程序设计的思维方式,使得程序设计简单易行。

2. 专家系统语言 Prolog 对问题的求解是基于事实、规则的逻辑推理过程,用其设计的程序具有简洁性。如程序一中只用了一条规则:

```
match(P1,P2):-pupil(P1,9),pupil(P2,9),P1<>P2,write(P1,P2),fail.
```

便可自动实现系统要求的两两选手配对功能。这是因为 Prolog 语言作为一种专家系统语言,是以面向自然语言的谓词逻辑作为语言基础的,其本身就是规则的描述,使程序设计更接近于人的思维方式,结构清晰易懂、易读。而 BASIC 是过程型语言,使用时要告诉计算机如何一步一步地解决问题,明确表达出完成一种计算所必须的控制流程的一切细节,因此写出的程序显得冗长繁琐。实现同样的功能,程序二中使用了三个循环十六条语句才实现,编程的耗费大大超过了程序一。

3. 专家系统具有学习机制,容易对知识库进行修改。而一般程序设计语言针对某一具体问题的程序,一旦编写调试完毕,其功能就确定下来,不易更改,如,若要在程序一中增加一个系统目标,只需要增加一个命题(或一条规则)即可。而程序二中却要做大改动,“牵一发而动全身”,有时甚至整个程序要重写,重新构造算法。因而一般应用程序的应变能力和扩展能力远远不如 Prolog 语言。知识的更新,要求一个好的系统应具有较强的灵活性,这一点上 Prolog 语言明显优于一般程序设计语言。

与数据库比较

Prolog 语言所能处理的问题不仅包括了数据库所能处理的范围,而且大大超过了它。程序三是用 dBASE II 编写的程序,具有同程序一相同的功能,将程序三与程序一比较,可以看出 Prolog 语言的优越性。

程序三:use class.dbf

```
accept "Enter Pupils' Number" to N
I=1
```

```

do while I <= N
if Age = 9
replace MAT with 1
endif
skip
store I+1 to I
enddo
store 1 to L
count for MAT1=1 to M
go top
locate for MAT=1
do while M > 0
do while MAT1=1
continue
enddo
replace MAT1 with 1
store NAME to X
continue
do while .not.EOF
@L,5 say X
@L,15 say NAME
store L+1 to L
continue
enddo
go top
locate for MAT=1
M=M-1
enddo
return

```

程序三在执行过程中对以下 DBF 文件进行操作:

class. dbf;

	NAME	AGE	MAT	MAT1
1	Peter	9	0	0
2	Charles	9	0	0
3	Paul	10	0	0
4	Susan	9	0	0

文件中字段 MAT 是存放具有比赛资格的同学标记“1”(无资格参赛者标记为“0”),MAT1 的数字“1”或“0”分别表示已参加过“配对”处理的记录。

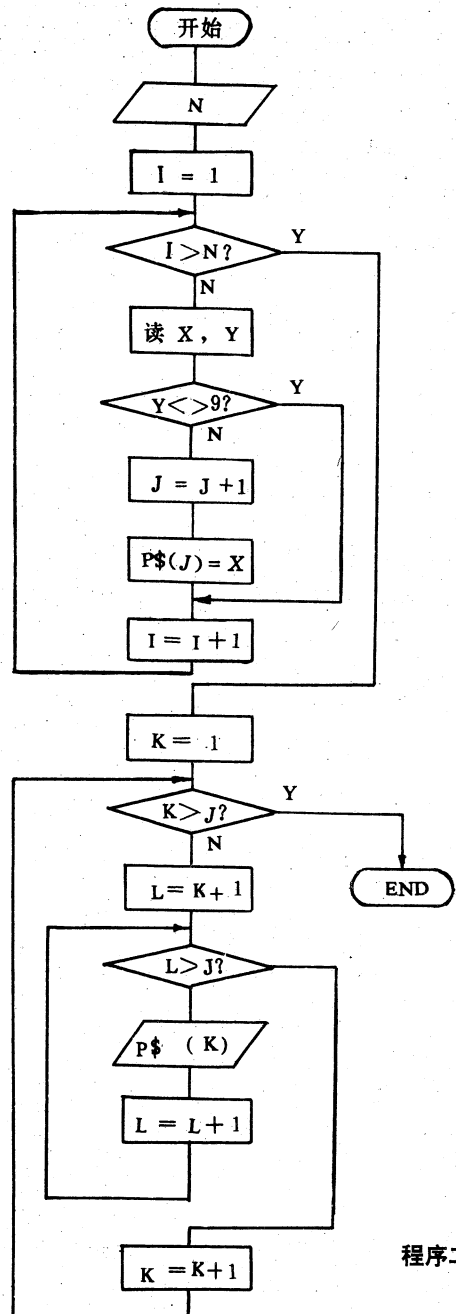
1. 数据库管理系统的目标是对大批量数据进行管理,主要对数据进行存取、增删、统计等操作。而 Prolog 语言除了具有上述功能外,还建立知识库,并与推理网络联接起来,由系统自行处理那些有关知识,最后得到问题的结论,具有推理功能,这一点是数据库系统无法比拟的。因而,专家系统语言 Prolog 被称为“智能数据库”。

2. 数据库语言 dBASE III 仍属于过程型语言,编程复杂、冗长、功能的修改不方便。如程序三中用了三十一条语句来实现程序一中一条规则的功能,其表达能力远远低于 Prolog 语言。

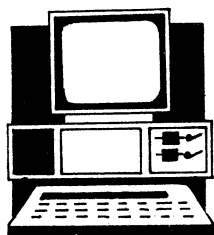
3. 用 Prolog 语言编写的专家系统,由于其程序设计高效简洁。使得其运行速度快,而 dBASE III 数据库系

统的处理时间长,工作效率低。

由于 Prolog 语言的程序设计思想,处理问题的手段,所处理的对象完全不同于常规的事务处理程序设计语言,它以更接近于人们思维方式的设计思想来处理数据,以巧妙的搜索技术(Prolog 中的目标直接深度优先搜索——回溯)及模式匹配执行程序等处理问题的手段,以专家的领域专门知识(而不是固定不变的数据)作为处理对象,使用 Prolog 语言设计的程序具有常规程序设计语言和数据库系统无可比拟的优越性。



程序二框图



PC 用户

结构文件与 field() 函数的比较

福建华侨大学(362011)严桂兰 刘甲耀

当使用数据库管理系统来实现各种事务管理时,最大的优势就是数据与程序的相互独立,多个功能模块在设计时丝毫不考虑数据库的结构与具体数据,而数据库的建立与修改除考虑实际事务情况外,对程序并无要求与影响,因而,各功能模块的独立性与适应性是十分明显的,使该模块在实现其功能时,对任一数据库都可通用。为达到此目的,在程序设计中,不论是 dBASE III 还是 FoxBASE 系统,都要用到宏替换函数 &。除此之外,对于中文管理信息,由于各系统对数据库字段名长度都有限制,若直接使用中文字段名,往往会遇到麻烦。为此,多采用代码字段,这就要求各代码有相应的中文显示,但又不失原字段代码的数据处理,在此前提下,对 dBASE III 来说,关键手段是使用结构文件,而对 FoxBASE 则采用 field() 函数,使程序设计的模块不受具体数据库的限制。

一、建立字典库,以查找代码字段的对应中文信息

为了显示各用户数据库代码字段的中文含意,可采用字典库的办法,字典库中以库名(KM)作为联接关键字,对照字段(ZD)数据则按代码库中字段顺序加入中文信息,字典库中可以复盖形式装入当前代码库的字段中文,也可装入多个代码库的字段中文,例如 DZ 字典库

```
. use dz
. list
RECORD # KM  ZD
1 RU 姓名
2 RU 性别
3 RU 年龄
4 CK 品名
5 CK 规格
6 CK 数量
7 CK 厂家
:
```

当使用某一数据库时,由于数据库的库名已知,从而在 DZ 库中可滤出对应数据库的中文字段,对应不同的系统软件,其实现的手段也各有所异。

二、dBASE III 中的结构文件

在 dBASE III 管理系统中,结构文件的产生是依附于任一已知数据库文件,其结构由系统提供,它由 FLELD-NAME、FLELD-TYPE、FIELD-LEN 和 FLELD-DEC 四个固定字段组成,如 CUST. DBF 结构文件所示:

```
. use cust
. list stru
字段 字段名          类型      宽度  小数
```

```
1 FIELD-NAME          字符型  10
2 FIELD-TYPE          字符型   1
3 FIELD-LEN           数字型   3
4 FIELD-DEC           数字型   3
** 总计 **                18
```

其记录则是已知数据库的结构,不同的数据库,CUST 的结构永远不改变,仅是其记录随之改变,不同记录就可得到不同的事务管理的内容,从而达到同一程序对不同的事务而可完成同样的功能,现以输入功能模块为例说明其实现过程:

(1)产生结构文件,取出各代码字段存入宏替换数组之中。

```
:
acce"请输入数据库名:"to wj
sele 1
use &wj
copy to cust stru exte
sele 2
use cust
m=1
do while . not. eof()
kl=field-name
if m<10
u="x"+str(m,1)
else
v="x"+str(m,2)
endi
&v=k1
skip
m=m+1
endd
:
```

不同的数据库输入数据库名后,立即产生结构文件 CUST,然后通过宏替换函数 & 形成的数组取出 CUST. DBF 中各记录所含的字段名。

(2)取出字典库中各代码字段的中文信息。

```
:
sele 3
use dz
loca for km="&wj"
m=1
do while . not. eof()
k2=dz
if m<1
u="y"+str(m,1)
else
u="y"+str(m,2)
```



```

endi
&.u=k2
m=m+1
cont
endd
:

```

由于已知数据库名,就可以在字典库中找到对应的中文字段,用函数 & 形成另一数组加以保存。

(3) 输入记录

在执行上述程序段之前,已建成用户各自的管理数据库,例如人事数据库(RU. DBF),仓库数据库(CK. DBF),订单数据库(DD. DBF)等。

```

. use ru
. list stru
字段  字段名  类型  宽度  小数
1      A1      字符型  6
2      A2      字符型  2
3      B3*     数字型  2
* * 总计 * *                11

```

则执行上述程序段时,首先在提示“输入数据库名:”处,按下 ru 或 ck 或 dd..., 为下列输入程序段作了必要的准备,使下列程序段执行时,既有对应中文字段显示,又在用户所指数据库中输入了一条新的记录。

```

:
sele1
appe blan
m1=1
clea
do while m1<m
if m1<10
v="x"+str(m1,1)
u="y"+str(m1,1)
else
v="x"+str(m1,2)
u="y"+str(m1,2)
endif
k1=&.v
k2=&.u
@ m1,1 say k2+";" get &k1
m1=m1+1
endd
read

```

例如下列事务管理:

人事	仓库	订单
姓名:-	品名:-	订单号:-
性别:	规格:	客户:
年龄:	数量:	型号:
籍贯:	单价:	交货期:
:	:	:

“-”为光标等待位置。

不同的中文字段提示,就会输入相应数据。上述各程序段并不会因数据库结构不同或者数据库结构的改变而改变,不同的数据库表现在结构文件的记录在发生变化,各数组元素对应的内容相应变化,仅此而已。反过

来,程序的变化,也不会影响各数据库的变化。

三、FoxBASE 中的 field() 函数

在 dBASE III 中,虽然用结构文件能很好地解决程序模块的通用性问题,但因不同的数据库都要通过一个结构文件来取得相应的字段名变量,加上 dBASE III 中没有数组语句,而是借用 & 函数形成数组,使用 & 函数虽很方便,但其替换时间影响程序执行的速度,而在 FoxBASE 系统中,由于有 field(m) 函数,它直接给出当前数据库中 m 值指定位置的字段名变量,因此,用 FoxBASE 实现同样输入功能时,就省去了产生结构文件的过程,这在时空上都胜过 dBASE III 一筹;另一方面, FoxBASE 可用 dimension 语句直接建立数组变量,省去了 & 函数替换时间。对于不同的数据库,其字段名个数不等,其数组长度也要随之改变,为此,程序中要有自动测定不同字段个数的办法,用数组元素存取代码字段的中文含意就显得十分方便了。

```

:
acce "请输入数据库名:" to wj
sele 3
use dz
set filt to km="&.wj"
n=reco()
set filt to
dime a(n)
loca for km="&.wj"
m=1
do while .not. eof()
a(m)=dz
m=m+1
cont
endd
sele 1
use &.wj
appe blan
m=1
L=1
clea
do while L<>0
k1=field(m)
@m,1 say a(m)+";" get &k1
m=m+1
L=len(field(m))
endd
read
:

```

四、结构文件与 field() 函数的比较

结构文件中的 field name 字段与 field() 函数的作用十分相似,由它们都能得到用户数据库的字段名,并借助此法来实现程序模块的通用性。除上述比较外,在作用时还有一些差异。例如:

```

use cust
@m,1 say field.name get field.name
read

```

执行时不会出错,它能通过结构文件来修改用户数据库中的字段名。而

```

:
use ru
@1,1 say field(1) get field(1)
read

```

此程序段形式上与上程序段相似,但执行时会出错,这是因为 get 语句后要求变量而不是表达式,因此,field(1)函数不能直接用在 get 语句之后,如若改

为:

```

use ru
s=field(1)
@1,1 say field(1) get &s
read
:

```

执行时,会修改用户数据库当前记录第一个字段数据,因此,即使 field.name 与 field(1)有些相似,但在使用时,有本质上的差别。

24 小时服务系统的人员安排

汕头四中(515031) 陈君佐

宾馆,酒楼,工厂,医院等 24 小时连续工作系统,都想用较少的员工,做较多的事,达到提高经济效益的目的。

例 1 某饭店日夜营业,每个员工连续工作 8 小时,经测定,0~4 点需员工 4 人;4~8 点需员工 8 人;8~12 点需员工 10 人;12~16 点需员工 14 人;16~20 点需员工 12 人;20~0 点需员工 7 人。试求所需工作人员的最少人数,并给出排班表。

下面是我在 IBM-PC 微机上编写的“工作安排”程序:

```

10 REM "WORKAP. BAS"
20 INPUT A1,A2,A3,A4,A5,A6
30 X=A2+A4+A6
40 Y=A1+A3+A5
50 H=ABS(X-Y)
60 PRINT
70 PRINT X,Y,H
80 PRINT
90 FOR X1=0 TO A1
100 X2=A2-X1
110 IF X>Y THEN X3=A3-X2+H
    ELSE X3=A3-X2
120 IF X<Y THEN X4=A4-X3+H
    ELSE X4=A4-X3
130 X5=A5-X4
140 X6=A6-X5
150 IF X1<0 OR X2<0 OR X3<0
    OR X4<0 OR X5<0 OR X6<0
    THEN 200
160 M=M+1
170 PRINT M,X1,X2,X3,X4,X5,X6
180 PRINT
190 S=X1+X2+X3+X4+X5+X6
200 NEXT X1
210 PRINT "SUM=",S,X,Y,H
220 END

```

当 RUN 运行程序,送入 6 个时间段所需员工数 4,8,10,14,12,7

电脑立即显示,需员工 29 人,其中 3 人是富余人员,安排在上午 8 点来上班,在工作最忙时洗洗菜,为客人送饭菜,洗盘碗。

同时,给出下列 5 种人员上班表:

该 上 班 人 数	时 段	0~4	4~8	8~12	12~16	16~20	20~0
	排 班	来 上 班 人 数					
1		0	8	5	9	3	4
2		1	7	6	8	4	3
3		2	6	7	7	5	2
4		3	5	8	6	6	1
5		4	4	9	5	7	0

例 2 某工厂,0~4 点需员工 5 人;4~8 点需员工 8 人;8~12 点需 200 人;12~16 点需 200 人;16~20 点需 50 人;20~0 点需 48 人;问该厂最少需员工多少人? 怎样合理排班?

同上可知,需员工 256 人,有 6 种排班表:

1	0	8	193	7	43	5
2	1	7	194	6	44	4
3	2	6	195	5	45	3
4	3	5	196	4	46	2
5	4	4	197	3	47	1
6	5	3	198	2	48	0

例 3 某厂,6 个时间段,需要人员为 5,8,25,14,12,17。这是较难安排上班的单位,用电脑,也能给出下列 1 种合理安排。知员工需 42 人,富余人员 3 人,上班表为:

1 0 8 17 0 12 5 再没比此更合理的安排了。

本程序也适用于非连续工作的系统!

ARC 系列压档工具软件的应用

陕西西安电子科技大学 125 号信箱 (710071) 戴青松

用过 Turbo Pascal 5.5 的用户,列表该软件包的目录,不难发现其中有不少扩展名为 .ARC 的文件,它们是一些文件的集合,称作“压缩文档文件”(ARC 文件)。

压缩文档文件的出现不是偶然的。当前,计算机硬件价格急剧下降,使得计算机的配置愈来愈齐全,普通出售的计算机内存容量都在 640KB 以上。内存的扩大,减少了编写软件的束缚,因而现今推出的软件性能愈趋完善,而所占的磁盘介质也越来越多,动辄几张、十几张磁盘,不但占用空间大,而且携带安装不方便。因此,不少软件公司在推出软件时,就已将其中一部分文件压缩存储,待安装时再展开,使用携带十分方便。如 Borland 公司近期推出的 Turbo 系列软件,MS 公司的 WINDOWS,Novell 公司的 Netware 等,其中的一些文件均进行了压缩。

对于普通计算机用户来说,利用压缩文档文件管理自己的文件系统更为可取。这是因为用户手中一般均收集有相当数量的优秀软件,但经常使用的不过几种,大多数软件均存储在磁盘上束之高阁,占用了大量磁盘空间。如果事先对这些软件进行压缩归档,就可以节省近一半的磁盘容量,经济效益十分明显。并且由于文件减少,用户更便于管理。另外,大多数压缩文档软件支持口令字数据加密,保证了数据安全。

目前,国际国内流行的压缩文档软件工具有多种,如 ARC; ZOO; PAK; PKARC; PKZIP; PCSECURE; LHARC 等。这些压缩软件由于推出时间的不同以及压缩算法的差异,性能差别很大。表一是实际使用这些软件进行文档压缩的结果,所用机型为 8MHz AT 机。根据笔者实际使用的经验,以及表一的结果,我们向大家推荐三种高性能的文档压缩工具: LHARC、PKZIP 及 LZEXE。

LHARC 是日本 89~91 年产生的高性能压档软件,至今已发展到 2.1 版(91 年 3 月 3 日出品)该软件短小精悍,一个仅 30 多 K 的程序就完成了文件归档、展开、列表、删除、检验、生成自展开文件等全部功能。最吸引人的是,由于其采用的 LZHUF 算法先进,在诸多压档软件中, LHARC 的压缩比最高。

直接运行 LHARC 不带参数, LHARC 就自动显示其 HELP,这个规律亦可应用在其他压档软件上。(事实上,尽管各压档软件性能不同,参数也不一样,但基本命令 A、E、D、V 等均是兼容的)

LHARC 的命令格式为:

```
[d1;][path] LHARC (COMMAND) (Archive. [LZH])  
[d2;][path][Filenames]
```

其中: Archive. [LZH] 指压缩后生成的文档名,扩

展名省略则默认为 .LZH。

Filenames 指被压缩的源文件名,必须写齐扩展名,但可用通配符 * 或 ? 代替。

LHARC 的 COMMAND(命令)很多,但常用的只有几种,下面一一分别介绍。

1. 文件归档命令 A

例: A>C:\LHARC A C *.C

该命令执行后,将首先从 C 盘上读入 LHARC(设 LHARC 在 C 盘根目录下),而后把 A 盘上所有扩展名为 .C 的文件压缩归档,存储在 A 盘上名为 C. LZH 的文件包中。若 A 盘上原先不存在 C. LZH 文件,则 LHARC 自动在 A 盘生成该文件;若 A 盘上已有同名文件,则 LHARC 把 *.C 文件压缩后追加加入 C. LZH 文件中。

此命令执行后源文件 *.C 仍存在于 A 盘上。

2. 文件归档并删除源文件命令 M

该命令的执行与 A 命令相同,但在生成归档文件包 Archive. LZH 的同时将源文件 Filenames 全部删除。

3. 文档列表命令 L or V

例: A>C:\LHARC L C. LZH

两个命令 L、V 均为归档列表命令,执行后将在屏幕上列出有关 C. LZH 包中所有归档文件的信息(源文件名;文件原始大小;压缩后大小;压缩比;文件生成日期;文件属性;CRC 校验码等),两命令的差别只在于显示格式的不同。

4. 文件复原(展开)命令 E or X

例: A>C:\LHARC X C. LZH

执行该命令后,将把 C. LZH 中所有文件展开恢复为源文件 Filenames。若源文件 Filenames 已存在,则 LHARC 在展开该文件时自动跳过(在老的 1. X 版本中将询问是否覆盖),展开后的文件与相应原文件完全相同。并且该命令还能这样使用:

A>C:\LHARC E C BAUP. C

当 C. LZH 包中确实存在 BAUP. C 文件时,执行该命令则 LHARC 只展开 C. LZH 中 BAUP. C 一个文件。

5. 归档文件校验命令 T

源文件归档压缩后, LHARC 在压缩文件包中加入 CRC 检验码。为了确保文件内容不损坏丢失,在每次归档压缩后,用户可利用 T 命令对 Archive. LZH 文件的 CRC 码进行检验。

例: A>C:\LHARC T C

事实上,压档文件出现 CRC 错误的概率是很小的。

6. 文件自展开命令 S

例: A>C:\LHARC S C

用户在使用归档文件包 C. LZH 时,必须将其展

开,如每次均用 LHARC 很感不便,因此 LHARC 提供了一个自展开命令 S,执行后归档包 C. LZH 将自动转换成一个 C. COM 或 C. EXE 文件,将它拷入 C 盘直接执行,即无需 LHARC, C. EXE 将自动展开原 C. LZH 包中的源文件,使用十分方便。而生成的 C. EXE 只比 C. LZH 大 1K 左右。

LHARC 极为出色,可惜仍有不足:一是它不支持口令字数据加密,而这在数据安全中是相当重要的;二是它的工作速度较慢,特别在低速 PC/XT 机上感觉尤为明显。幸好,另一份压档软件——PKZIP 恰能弥补 LHARC 的这两个缺点。

PKZIP 是美国 PKWARC 公司 90 年推出的压档软件。它的最大特点是工作速度十分快。相对 LHARC 来说可节约 1/2~1/3 的时间,而其效率也仅比 LHARC 低 5%。因此在低速的 PC/XT 机上,推荐大家使用 PKZIP;而在高速的 286、386 机上,由于速度已不重要,因而最好使用 LHARC。

PKZIP 软件包内含数个执行程序。其中 PKZIP. EXE 主要用于压缩, PKUNZIP. EXE 主要用于展开, ZIP2EXE. EXE 用于形成自展开文件, PKZIPFIX. EXE 用于对压缩归档文件维护。PKZIP 的基本命令与 LHARC 相似,这里就不详细介绍了。需注意的是, PKZIP 命令参数前必须加一短杠“-”以便于 PKZIP 识别命令参数。

PKZIP 支持加密压缩方式,使用方法为:

例:A>C:\PKZIP[-A]-gCD45H C[. ZIP] *. C

执行该命令后,PKZIP 把 A 盘上所有扩展名. C 的文件压缩归档,并送入密码字“CD45H”后存储在 A 盘 C. ZIP 文件中。此后当需要展开扩展名. C 的源文件时,也需要送入正确的密码字“CD45H”,否则 PKZIP 将拒绝展开。

例:A>C:\PKUNZIP-gCD45H C BAUP. C

由于送入了正确密码,PKUNZIP 执行展开操作,展开 C. ZIP 中的一个源文件 BAUP. C。如果省略源文件名,则 PKUNZIP 将把 C. ZIP 中所有文件展开。

表一

注:以下数据在 3.33MHZ AT 机上所得

		原文件长度 (byte)	压 缩 后 文 件 长 度 (byte)						
			ARCV5.2	ZOOV1.0	PAKV1.6	PKARCV3.61	PKZIPV1.1	LHARCv2.1	LZEXEv0.91
源 文 件	FOX. BAT	19	19	19	19	19	19	19	--
	FOXBIND. EXE	24512	17010	17520	16165	17205	13351	13040	13837
	FOXPCOMP. EXE	72480	49797	49421	45610	49184	38023	30200	37043
	MFOXPLUS. EXE	247808	173044	169436	154250	166839	128198	119754	105156
	PC430. EXE	171088	142631	137359	122356	136761	103580	99444	106667
	MFOXPLUS. OVL	138032	100994	69285	89768	98348	72321	69285	—
	FOXPHelp. HLP	149909	98477	75839	65055	73383	56295	54604	—
压缩比		100%	28%	32%	39%	33%	49%	51%	49%
统共压缩时间		—	5分02秒	1分31秒	2分20秒	0分33秒	1分47秒	2分33秒	—

利用 LHARC 和 PKZIP 压缩不常用的软件是相当出色且实用的。但若利用它对常用软件压缩归档,则用户将感到不便。因为通过 LHARC 或 PKZIP 使用归档文件,首先必须在软盘或硬盘上展开,而后才能使用,这个过程是繁琐而无趣的。如果一份软件压缩后无需在磁盘介质中展开即能直接运行,那将是多么方便啊! ComputerLink Magazine 推出的 LZEXE 实用软件可实现这项功能。

LZEXE 使用格式:

C>LZEXE [Filename[. EXE]]

LZEXE 软件只对 EXE 可执行文件进行压缩,生成一个新的压缩过的 EXE 文件。这个新文件与源文件 Filename. EXE 同名,而源文件 Filename. EXE 则改名为 Filename. OLD 两者可同时存在。在新生成的 EXE 文件中,不但包含了源文件的全部内容(压缩),而且附加了一小段程序。当直接运行新 EXE 文件时,附加的程序段首先工作,在机器内存中展开源文件的内容,并试图在 DOS 控制下执行它。由于全部操作均在内存中进行,因而速度极快。据笔者经验,在较高速的 AT、286 等机型上,当从软驱运行文件时,压缩后的文件执行速度将比直接运行源文件更快!(这是因为压缩文件体积相对较小,减少了磁盘读取量,因而速度得以提高)而在普通带 Turbo 加速键的 PC 机上,两者执行速度相当。

LZEXE 只可对 EXE 文件压缩,应用范围较小。幸而该软件包还提供了一套 COMTOEXE,用于将 COM 文件转换为 EXE 文件,这样经过转换后 COM 文件也能压缩了。另外由于一般 EXE 文件均已经 Microsoft 公司的 EXEPACK 程序对文件头压缩过,因而 LZEXE 包还提供一套 UPACK 程序,利用它事先对 EXE 文件反压缩后再动用 LZEXE,压缩效果更好。

LHARC、PKZIP 及 LZEXE 三套工具软件性能极佳,且压缩文件内容完全忠实于源文件,用户可放心使用不必担心意外。当用户能熟练地操作它们时,将会赞叹其卓越的性能。

DOS3.30 若干新增命令的功能与应用

长沙水电师范学院计算机中心(410077)柳见成

自从 1981 年 IBM PC 系统问世以来,近十年时间内,所配置的 DOS 操作系统版本不断改进和更新,其中以 1983 年推出的 DOS 2.0 版最为著名。DOS 2.0 版的主要特点是采用树结构目录以适应硬盘配置,以及吸取了 UNIX 系统的 PIPE 机制,支持管道和 I/O 重定向操作等功能。DOS 2.0 的影响非常广泛,可以说,我国广大的 IBM PC 和 0520 系列微机用户都是在 2.0 的基础上掌握 DOS 操作系统使用的。1984 年公布适应 PC/AT 高容量软盘驱动器的 3.00 版,1985 年由于局部网络和 3.5 英寸软盘驱动器的需要相继推出了 3.10 和 3.20 版本。1987 年问世的 3.30 在兼容以前各版本的基础上又增加了一些新的功能。对单用户单任务情况而言,3.30 已是一个比较成熟的版本,因而目前在 286 和 386 系列微机上被广泛地应用。本文结合工作中的实践经验,介绍 3.30 版中对一般用户很有意义的若干新增命令(相对于 2.0 版而言)的功能和应用实例。

1. 安装虚拟磁盘

3.30 提供的 VDISK.SYS 是一个设备驱动程序,用于将内存的一部分作为存储介质来仿真磁盘驱动器。由于虚拟盘以内存速度运行,因此远快于访问硬盘的存取操作。安装 VDISK.SYS 的语句应包含在 CONFIG.SYS 中,并在重新启动 DOS 后才起作用。

格式:DEVICE=[d:][path]VDISK.SYS ddd sss mmm/e

格式中三个选择参数分别为虚盘容量、扇区大小和目录数。选/e 可在超过 1M 的扩充内存中安装虚盘。

在目前基于 8086/8088 CPU 和基于 80286 CPU 的微机中,尽管寻址空间可达 1M 和 16M,但 DOS 管理的自由空间最多为 640K。汉字系统工作时如装入字库,则难以运行大型程序。例如标准 FoxBASE+ 需要至少 375K 内存,最大程序执行时可达 560K。在 CCDOS 2.13F 下即使只装入一级字库(130K 左右),一般命令和程序也无法运行。虽然可将全部字库驻留硬盘来保证 FoxBASE+ 的内存要求,但调用汉字时将大大降低运行速度。此种情况下如果主存储器扩展到了 1M 或 1M 以上,就可利用 DOS 3.30 的设置虚盘功能来解决这一矛盾。方法如下:

用行编辑或字处理程序在配置文件 CONFIG.SYS 中加一条命令:

DEVICE=VDISK 384 128 64/E

重新启动 CCDOS 2.13F 后,键入屏幕上的选择 3,

即可将两级字库装入虚盘 D,能正常运行 FoxBASE+ 程序且保证快速调入汉字。

2. LASTDRIVE 命令

格式:LASTDRIVE=X

此命令用于设置用户能访问到的最大驱动器号,从而使用户可设置多于物理驱动器数量的逻辑驱动器。X 代表 DOS 可接受的最后有效驱动器字母,取值范围为 A~Z。此命令和 VDISK.SYS 一样应置于 CONFIG.SYS 文件中,在启动 DOS 时进行配置。

3. SUBST 命令

格式:SUBST d; d:path

此命令可使用不同的驱动器标识符来代替其它驱动器或路径。其中 d; 指定用于代替的驱动器字母,d; path 指定被代替的驱动器和路径。

在使用 WordStar 字处理软件时,由于 WS 4.0 以下版本不支持 DOS 的树结构路径,即使利用后面将讨论的 APPEND 命令可解决不同路径下共享字处理软件的问题,也不能直接建立当前目录以外的文件。解决办法是利用 SUBST 命令替换。例如,欲在 C 盘的子目录 PATH2 下建立文件 FILE2,设用 F: 代替路径 PATH2,则操作步骤如下:

C>SUBST F:\PATH2

然后进入字处理软件所在路径,运行该软件并在起始菜单中选择 D,在文件名后面键入 F:PATH2。此标识符能为 WordStar 所识别,经过 SUBST 命令转换后即可按指定路径存盘,因此可支持任意路径,在树结构层次较多的情况下更显得方便。事实上,对一切不能识别路径名的应用软件,均可应用这一方法。使用时应注意两点:①替换命令中指定驱动器字母不得大于 CONFIG.SYS 中 LASTDRIVE 的值,且不可与当前驱动器号相同。②指定路径时必须从根目录开始。

4. APPEND 命令

格式:APPEND[d:][path[:[d:][path...]]]

此命令用于查找和装入在当前目录之外,且扩展名不为 .COM、.EXE 和 .BAT 的数据文件。以前的 DOS 版本只提供了 PATH 命令,仅能搜索可执行文件和批文件。对于某些包含覆盖文件的软件系统,如 WordStar 和 FoxBASE+ 等就无法作到让不同用户在不同子目录下共享,只能将这些软件分别拷贝到自己的子目录中,造成磁盘空间的大量浪费。利用 DOS 3.30 提供的 APPEND 命令可完全解决这一问题。

设硬盘上装有 CCDOS 2.1F 汉字系统、WordStar

和 FoxBASE+, 分别在子目录 \213、\WS、\FOX 下, 则可在 AUTOEXEC. BAT 中设置如下命令:

```
PATH C:\213;C:\WS;C:\FOX
```

```
APPEND C:\213;C:\WS;C:\FOX
```

系统启动后, 即可在任意路径下直接调用字处理和 FoxBASE+ 软件, 从而实现这些应用软件的真实共享。

还有一点应予说明, APPEND 查找数据文件的顺序与 PATH 相同。先在当前路径找, 未找到再按 APPEND 命令给出的搜索路径查找。因此当使用某些文本编辑程序(如 EDLIN)建立一个新文件时, 有可能调入一个位于搜索路径上的其它子目录下同名文件, 且对它的编辑和修改将被存入当前目录, 而对 COPY、RENAME、DEL 和 DIR 等命令则不起作用。

5. ATTRIB 命令

格式: ATTRIB[+R/-R][+A/-A]<文件标识符>/S

用于修改指定目录中选择的文件或该目录层下全部文件的文件属性。ATTRIB 命令是 DOS 3.0 增加的, 但 DOS 3.30 进一步增强了修改文件存档属性的功能。[+R/-R] 为设置或取消指定文件的只读属性, 在一定程度上可保护文件不被任意修改和删除。[+A/-A] 为设置或恢复指定文件的存档属性。DOS 在文件目录表中为每个磁盘文件分配了一个 32 字节的“目录登记项”, 其中第 11 个字节为属性域, 通常其值为 20H, 表示该文件可被写入或修改。恢复该属性可影响 BACKUP/M、XCOPY/M 等操作。当设置存档属性时,

文件可被复制, 否则就不被复制。应用此命令为用户有选择地备份文件提供了一个有效的手段。实例见后。

6. XCOPY 命令

格式: XCOPY <源标识符> <目的标识符> [/A]/D[/E]/[M]/[P]/[S]/[W]/[V]

应用此命令可有选择地复制一组文件, 文件中可包含较低层次的子目录。格式中的源可以是一个驱动器, 一个路径, 一个文件名或三者的组合, 各可选参数的说明此处从略。

XCOPY 命令为 DOS 3.30 以前所没有。与 COPY 命令相比, 其特点在于:

① 可根据文件属性有选择地复制一组文件, 也可加上参数 [/P] 一个一个选择复制, 而不像 COPY 命令那样受通配符的限制。

② 可以复制树结构文件, 包括在复制的同时建立子目录。

例: 欲将 A 盘上除 .COM 和 .EXE 文件以外的其它文件全部复制到 C 盘上尚未建立的子目录 \AAA 下, 操作步骤如下:

```
C>ATTRIB -A A:\*.COM /S
```

```
C>ATTRIB -A A:\*.EXE /S
```

```
C>XCOPY A:*. * C:\AAA /S /M
```

在屏幕提示 (F=file, D=directory)? 后面回答 D 即可完成操作。

以上各命令的完整格式及详细说明均请参看 DOS 3.30 使用手册。

WPS 的彻底解密

江苏如东县中学(226400) 瞿新国

香港金山公司推出的高级文字处理系统 WPS 2.0 因其功能强大、使用方便, 在国内使用很广。作者为了保护其成果, 采取了加密措施。运行时, 一旦机器上的日期超过 1991 年 2 月 4 日, 将破坏系统(删除 WPS EXE 及两个覆盖文件), 给使用者带来诸多不便。虽说可以在使用前将日期改成 1991 年 2 月 4 日前的日子, 但其本身的“当前日期复写到文章中”的功能失去了作用。很多软件需要机器上的正确日期。人们也不习惯于错误的日期; 在使用前一旦忘记修改机器上的日期, 就删除了系统, 又得重新恢复。

笔者现已找到一种将其彻底解密的方法。考虑到此软件通过判断日期决定是否删除文件, 程序中肯定要取机器日期, 取机器上的日期一般通过 2AH 号功能调用实现, 汇编指令为

```
MOV AH, 2A
```

```
INT 21
```

这两指令的代码是: B4 2A CD 21。我们只要用 DEBUG 工具的 S 命令找出软件中取机器日期的程序段加以分析, 就不难找出其做手脚之处。具体操作是:

```
REN WPS EXE WPS
```

```
DEBUG WPS
```

用 S 命令查找上述两指令代码的首地址: S100 LFF00 B4 2A CD 21, 发现软件有四个取时间的程序段, 再用 U 反汇编命令分别汇编分析, 发现地址为 XXXX: DB42 和 XXXX: E1A4 的两段程序与加密有关, 其中地址为 XXXX: E1A4 的一段用于显示警告信息; 地址为: XXXX: DB42 的一段用于删除文件。我们只需用 A 命令将两段程序中的 CMP CX, 07C7 (地址分别为 XXXX: DB46 和 XXXX: E1AB) 改成 RET 便使这两段程序失去应有的作用, 也就完成了解密工作。WPS 的其它功能均不受影响。

终止号(即存盘时的序号:n),当驱动器红灯熄灭时按一下 Esc 键,即显示一屏字幕,连续这个操作直至显示第 n 屏字幕;此后,若重复一遍则按 R 键;返回主菜单则按回车键。

6. 当操作失误时,程序转入主菜单,如果运行中程序被挂起,可强制中断(Ctrl-Reset),此时进入监控,再键入 Ctrl-C 即返回 BASIC,再重新运行程序(RUN)。

```

100 ONERR GOTO 120
110 GOSUB 3110
120 PRINT CHR$(4); "PR#3"; PRINT; HGR2
130 VTAB 1; PRINT TAB(9); "CEC-I 彩色 TV 字幕";
    VTAB 3; HTAB 11; PRINT "①字幕制作"; HTAB 11;
    PRINT "②字幕存盘"
135 HTAB 11; PRINT "③字幕打印"; HTAB 11; PRINT "④单
    页字幕显示"; HTAB 11; PRINT "⑤连续字幕显示";
    HTAB 11; PRINT "⑥结束"
150 VTAB 9; HTAB 21; INPUT "选择:"; XX$; XX = VAL
    (XX$); IF XX < 1 OR XX > 6 THEN CALL -1052;
    GOTO 150
152 IF XX = 2 THEN HH = 211
154 IF XX = 3 THEN PR = 1; HH = 204
156 IF XX = 4 THEN HH = 204
160 ON XX GOTO 200, 5100, 5100, 5100, 5200, 190
190 END
200 CLEAR; GOSUB 3200; HGR2; VTAB 1
205 INPUT "背景(0 黑; 1 绿; 2 紫; 3 白; 5 橙; 6 蓝)"; BJ;
    GOSUB 2200
210 HOME; A = A + 1; PRINT "第"; A; "行/"; PRINT "本行
    Y 坐标起始为"; LJ; "/"; PRINT "还可用"; 192 - LJ
220 INPUT "输入字符串:"; ZC$; IF LEN(ZC$) = 0
    THEN 350
230 INPUT "放大倍数:"; BS
240 INPUT "立体(是/1; 非/0)"; LT
250 INPUT "正斜(正/0; 斜/1)"; ZX
260 W$ = ZC$; GOSUB 2000; I = W
265 PRINT "背景为:"; S$(BJ); "色)"
270 PRINT "色彩"; BJ$; INPUT SC
275 INPUT "放大描绘"(线/0; 图/1; 图/2); MH
276 IF MH = 0 THEN 280
277 IF MH = 1 THEN TX = 1; XZ = 0
278 IF MH = 2 THEN TX = 1; XZ = 8
280 INPUT "X 坐标(左/L; 中/M; 右/R)"; HW$
290 INPUT "Y 坐标(首/T; 中/M; 底/B)"; VW$
295 IF VW$ = "T" THEN LJ = 0
300 LJ = LJ + SG + 7
305 IF VW$ = "B" THEN 320
310 IF LJ - 7 > 191 THEN PRINT "本行超出屏底!"; CALL
    -1052; A = A - 1; LJ = LJ - SG - 7; GOTO 340
320 GOSUB 2100
330 SG(A) = SG; W$(A) = W$; I(A) = I; LT(A) = LT; ZX
    (A) = ZX; BS(A) = BS; SC(A) = SC; X(A) = X; Y(A) =
    Y; MH(A) = MH; TX(A) = TX; XZ(A) = XZ; GOTO 210
340 INPUT "重新输入本行数据?(Y/N)"; CX$; IF CX$ =
    "Y" THEN 210
350 HGR; POKE -16302, 0
355 HCOLOR = BJ; HPLLOT 0, 0; CALL 62454
360 FOR R = 1 TO A
370 W$ = W$(R); BS = BS(R); SC = SC(R); LT = LT(R);
    ZX = ZX(R); I = I(R); X = X(R); Y = Y(R); MH = MH
    (R); TX = TX(R); XZ = XZ(R)
380 POKE 230, 64
390 HOME; PRINT W$
400 & 1, 1, A%; M = A%; N = M; IF M = (INT(N/2)) * 2
    THEN M = M + 1
500 CC = 0; FOR H = 0 TO 1; FOR V = 0 TO 16
510 IF ZX THEN CC = 16 - V
520 & H, V, A%; IF M = A% THEN GOSUB 1000
540 NEXT V, H
700 NEXT R
710 HH = PEEK(-16384); POKE -16368, 0; IF HH = 160
    THEN 200
720 IF HH = 208 THEN GOSUB 5000
730 IF HH = 211 OR HH = 204 THEN 5100
740 IF HH = 141 THEN 120
790 GOTO 710
1000 POKE 230, 32; POKE 49328, 16; HCOLOR = SC
1100 IF MH = 1 OR MH = 2 THEN F = X + CC + H * BS; G =
    Y + V * BS; ROT = XZ; SCALE = BS; DRAW TX AT F,
    G
1200 IF MH = 0 THEN F = X + CC + H * BS; G = Y + V * BS;
    FOR Z = G TO G + BS - 1; HPLLOT F, Z TO F + BS - 1,
    Z; NEXT
1300 IF LT THEN HCOLOR = 6; HPLLOT F + BS, Z TO F + BS
    + 3, Z + 4; HCOLOR = SC
1500 POKE 230, 64; RETURN
2000 W = LEN(ZC$); HZ = 0; ZF = 0
2010 FOR I = 1 TO W
2020 IF ASC(MID$(ZC$, I, 1)) = 127 THEN HZ = HZ + 1
2030 NEXT I
2040 ZF = W - HZ * 3
2050 W = HZ * 16 + ZF * 8
2060 SK = W * BS + (LT * 3) + (ZX * 16); SG = 16 * BS +
    (LT * 4)
2070 IF SK > 279 THEN BS = BS / 0.5; GOTO 2060
2080 IF BS < 1 THEN PRINT "本行太宽, 请重新输入!";
    CALL -1052; FOR T = 0 TO 2000; NEXT; A = A - 1;
    POP; GOTO 210
2090 RETURN
2100 IF HW$ = "L" THEN X = 0
2110 IF HW$ = "R" THEN X = 279 - SK
2120 IF HW$ = "M" THEN X = INT((279 - SK) / 2)
2130 IF VW$ = "T" THEN Y = 0
2135 IF VW$ = "T" AND MH <> 0 THEN Y = 1 * BS
2140 IF VW$ = "B" THEN Y = 191 - SG
2150 IF VW$ = "M" THEN Y = LJ - SG
2160 RETURN
2200 IF BJ = 0 OR BJ = 3 THEN BJ$ = "(0" + S$(0) + ";
    1" + S$(1) + "; 2" + S$(2) + "; 3" + S$(3) + "; 5"
    + S$(5) + "; 6" + S$(6) + ")"
2210 IF BJ = 1 THEN BJ$ = "(0" + S$(0) + "; 1" + S

```

```

$ (1)+";3"+S$ (3)+")"
2220 IF BJ = 2 THEN BJ$ = "(0"+S$ (0)+";2"+S
$ (2)+";3"+S$ (3)+")"
2230 IF BJ = 5 THEN BJ$ = "(4"+S$ (4)+";5"+S
$ (5)+";7"+S$ (7)+")"
2240 IF BJ = 6 THEN BJ$ = "(4"+S$ (4)+";6"+S
$ (6)+";7"+S$ (7)+")"
2250 RETURN
3110 POKE 1013,76;POKE 1014,0;POKE 1015,3
3130 FOR I=768 TO 817;READ J;POKE I,J;NEXT
3140 DATA 32,185,246,32,17,244,164,229,177,38,41,
127,133,255,165,48,41,127,37,255,9,128,197,
48,208,4,169,1,208,2,169,0,72,32,190,222,32,
227,223,160,0,145,131,200,104,145,131,96,0,0
3150 DATA 1,0,4,0,44,62,0
3160 DZ=818;FOR I=DZ TO DZ+7-1;READ J;POKE I,
J;NEXT
3170 POKE 232,50;POKE 233,3
3180 RETURN
3200 S$ (0)="黑";S$ (1)="绿";S$ (2)="紫";S$ (3)
="白";S$ (4)=S$ (0);S$ (5)="橙";S$ (6)=
"蓝";S$ (7)=S$ (3)
3210 RETURN
5000 POKE 49236,0;PRINT CHR$ (4);"PR # 1";POKE
1913,1;PRINT CHR$ (17);PRINT CHR$ (4);"PR
# 0"
5010 T=PEEK (-16384);POKE -16368,0;IF T=141
THEN 5030
5020 GOTO 5010
5030 IF HH=208 THEN CALL -1052;RETURN
5040 GOTO 120
5100 PRINT CHR$ (4);"PR # 3";PRINT;HOME

```

```

5110 INPUT"请输入文件名: ";F$
5120 P1=PEEK(49236);POKE 49328,16
5130 IF HH=211 THEN PRINT CHR$ (4);"BSAVE";F$;
",A$ 2000,L$ 2000";GOTO 5170
5140 IF HH=204 THEN POKE -3086,0
5150 PRINT CHR$ (4);"BLOAD";F$
5155 IF PR=1 THEN 5000
5160 T=PEEK(-16384);POKE -16368,0;IF T=141
THEN 120
5170 GOTO 5160
5200 PRINT CHR$ (4);"NOMON,C,I,O";PRINT CHR
$ (4);"PR # 3";PRINT;HOME
5210 INPUT"输入文件名: ";FF$
5220 INPUT"画面起始号: ";S
5230 INPUT"画面终止号: ";E
5240 POKE 49328,16
5250 FOR I=S TO E
5260 F$ = FF$ +STR$ (1)
5270 IF I/2<>INT (I/2) THEN PK=49236;POKE 230,
32;PRINT CHR$ (4);"BLOAD";F$;","A8192"
5280 IF I/2=INT (I/2) THEN PK=49237;POKE 230,64;
PRINT CHR$ (4);"BLOAD";F$;","A16384"
5300 T=PEEK (-16384);POKE -16368,0;IF T<>155
THEN 5300
5310 POKE PK,0
5320 NEXT I
5330 T=PEEK (-16384);POKE -16368,0;IF T=210
THEN 5250
5340 IF T=141 THEN 120
5350 GOTO 5330
6000 REM By:BAO GAN 1990.7.10

```

BASIC 程序中变量名使用情况的列表程序

成都四川大学材料系(61006) 蔡伟

BASIC 程序中使用了大量的变量名,如果能对一个程序中的全部变量名(简单变量,数组变量,字符串变量)按字母顺序列表出来,每个变量名被使用的程序行号也列印出来,对于分析程序或修改程序都将极其有用。

下面介绍的 VLIST 程序能迅速完成这一项工作。程序是由 6502 机器语言编成,适用于苹果机、中华学习机及其兼容机等。程序置于内存 \$7D00~\$7E64 (相当于十进制地址 32000)。其中 \$7D00~\$7D4E 是 VLIST 的主程序段。\$7D50~\$7DE3 是根据变量名的命名规则在 BASIC 程序中去识别各种类型的变量名。\$7DE4~\$7E31 是将搜寻到的变量名排序。\$7E32~\$7E57 是针对每个变量名查出所出现的行号并列

印出。

VLIST 程序搜查变量名及行号非常迅速,不到一秒钟就能完成搜查列印工作。每印出一个变量名,在冒号后面印出它所在的全部行号。变量名是按照首字母的英文字母顺序排列的,首字母相同的变量名族内,按其第一次在程序中出现的先后顺序排序,因而便于清查。使用办法:

VLIST 程序见附录,假定你磁盘上已有 VLIST 程序,使用如下:

```

LOAD ××××(×××指要列印变量名的 BA-
SIC 程序文件名)
BLOAD VLIST
CALL 32000

```

屏幕上将连续列印变量名及行号,可用 Control-S 键暂停。

开启打印机,执行 PR #1 命令后,再执行 CALL 32000,可在打印纸上获得列表输出。

7D00— A9 41 85 5C 8D 00 02 A9
7D08— 00 85 0C 20 58 7E A8 91
7D10— 0A 20 50 7D C6 0C 20 58
7D18— 7E A4 5D A2 00 B1 0A F0
7D20— 23 20 8E FD B1 0A 9D 00
7D28— 02 C9 0D F0 09 09 80 20
7D30— ED FD E8 C8 D0 EE 84 5D
7D38— A9 BA 20 ED FD 20 50 7D
7D40— E6 5D D0 D5 E6 5C A5 5C
7D48— C9 5B 90 B8 4C 3C D4 00
7D50— A6 67 A5 68 86 B8 85 B9
7D58— A0 00 B1 B8 D0 06 C8 B1
7D60— B8 D0 01 60 A0 04 B1 B8
7D68— D0 0A A0 00 B1 B8 AA C8
7D70— B1 B8 D0 E0 10 0F C9 83
7D78— F0 04 C9 B2 D0 14 C8 B1
7D80— B8 D0 FB F0 E5 C9 22 D0
7D88— 0C C8 B1 B8 F0 DC C9 22
7D90— D0 F7 C8 D0 D1 C9 41 90
7D98— F9 C9 5B B0 F5 CD 00 02
7DA0— F0 06 A9 00 85 5A F0 02

7DA8— 84 5A C8 B1 B8 C9 30 90
7DB0— 0C C9 3A 90 F5 C9 41 90
7DB8— 18 C9 5B 90 ED C9 21 F0
7DC0— 08 C9 24 F0 04 C9 25 D0
7DC8— 03 C8 B1 B8 C9 28 D0 01
7DD0— C8 84 5B A2 00 A4 5A F0
7DD8— 31 B1 B8 9D 80 02 E8 C8
7DE0— C4 5B 90 F5 A5 0C D0 4A
7DE8— A9 00 9D 80 02 20 58 7E
7DF0— A4 5D A2 00 B1 0A F0 20
7DF8— DD 80 02 D0 04 E8 C8 D0
7E00— F3 C9 0D D0 09 BD 80 02
7E08— D0 0B A4 5B D0 85 C8 B1
7E10— 0A C9 0D D0 F9 C8 D0 DA
7E18— A2 00 BD 80 02 F0 06 91
7E20— 0A E8 C8 D0 F5 A9 0D 91
7E28— 0A A9 00 C8 91 0A A4 5B
7E30— D0 DA A9 0D 9D 80 02 A2
7E38— 00 BD 80 02 DD 00 02 D0
7E40— 13 E8 C9 0D D0 F3 A0 02
7E48— B1 B8 AA C8 B1 B8 20 24
7E50— ED 20 48 F9 A4 5B D0 B4
7E58— A5 B0 85 0B A9 00 85 5D
7E60— 85 0A E6 0B 60

磁头清洗及其辅助程序

广西师范大学(541001) 唐汉雄

Apple 机的驱动器使用时间长了,或使用了质量低劣的磁盘,磁头就会附着一些磁粉。影响信息的正确存取,严重时会出现读写错误,更甚时会划伤磁盘。因此磁头清洗是计算机日常维护不可缺少的一项内容。本文将介绍使用清洗盘进行磁头清洗的一般方法并给出一个磁头清洗辅助程序。

清洗盘是用来清洗驱动器磁头的专用工具。其使用的一般方法是:先将清洗剂均匀涂在清洗盘的窗口上,然后将清洗盘插入第一驱动器并关上仓门,再打开计算机电源,待驱动器转动约 30 秒钟后,按复位键即可完成第一驱动器的清洗工作。

对第二驱动器的清洗最简便的方法是:在 BASIC 状态下键入 POKE49387,0;POKE49385,0 命令即可启动第二驱动器,当到达预定清洗时间后再按复位键即可。

上述方法最大的优点是不需要载入 DOS 或其它工具软件就可进行清洗工作,尤其是第一驱动器不能正常读入 DOS 时特别有用。当计算机能正常载入 DOS 或工具软件时,也可使用 DOS 命令或工具软件的调盘操作进行清洗。

然而这些方法也有其不足,就是在清洗时磁头多

半停在 \$00 或 \$11 轨道上,磁头不会沿盘的直径方向上来回移动,磁粉也只能被吸附在清洗盘特定的轨道上,无疑会缩短清洗盘的使用寿命。同时清洗时间也需人为控制,启动命令需逐个输入,常感不便。要是计算机开机后能自动执行能使驱动器转动,其磁头又能在径向快速来回移动,同时还能自动控制清洗时间的程序,那么操作将会更加简便,清洗效果也将会更好。为此笔者编写了一个能实现上述功能的磁头清洗辅助程序。现将该程序的执行过程介绍如下:

首先自动寻找驱动卡所在的槽号,若找不到会给出“NO DRIVE CARD”(无驱动卡)的提示。若找到驱动卡则显示“DRIVE=□”,这时插入清洗盘再直接回车或按“1”键即可启动第一驱动器。清洗盘转动的同时磁头也沿盘的径向快速来回移动,约 20 秒钟后自动停止,随后又显示“DRIVE=□”,此时可改插清洗盘再回车或按“2”键就可清洗第二驱动器。

每次清洗完毕后驱动器号码可自动转换并给出提示,既可直接回车选取默认值,也可敲入 1 或 2 另行选择,还可按“Esc”键随时中断或退出清洗状态。

本程序最大的特点是自动化程度高,能自动寻卡,

驱动器号码自动转换,自动定时,且操作特别简单,插好清洗盘后只需按两次回车就可完成两台驱动器的清洗工作。因此本程序特别适用于大批量驱动器磁头的定期清洗。

本程序设定的清洗时间约为 20 秒,如需改变这一时间可增减程序 \$ 9079 单元的值。虽然本程序在有 DOS 时运行效果一样,但若将此程序作为 HELLO 程序或由 HELLO 程序直接调用并制成专用磁盘,启动磁盘后可自动执行本程序,会使清洗更为简便快速。顺便说明,因本程序可控制磁头快速来回运动,若手头没有专用的清洗盘,用一张干净的磁盘代替清洗盘也同样有清洗磁头的效果。

附磁头清洗辅助程序如下:

```
9000— 20 58 FC 84 06 A9 C8 85
9008— 07 A0 07 C6 07 A5 07 C9
9010— C0 F0 3F B1 06 D9 01 FB
9018— D0 EF 88 88 10 F5 A5 07
9020— 0A 0A 0A 0A 85 2B A9 10
9028— 85 24 20 5B FB A9 EA A0
9030— 90 20 3A DB C6 24 20 0C
9038— FD C9 8D F0 11 C9 9B F0
9040— 1B C9 B1 F0 1A C9 B2 F0
9048— 21 20 3A FF F0 D8 A9 B1
9050— D0 EF A9 F2 A0 90 20 3A
9058— DB 20 3A FF 4C D0 03 20
9060— E1 90 BD 8A C0 20 75 90
9068— D0 BC 20 E1 90 BD 8B C0
9070— 20 75 90 D0 B1 BD 89 C0
9078— A0 06 98 48 20 A8 90 A9
9080— 80 20 A8 FC 20 C2 90 68
9088— A8 AD 00 C0 10 05 8D 10
9090— C0 D0 03 88 D0 E4 A6 2B
9098— BD 88 C0 38 A9 63 ED 4F
90A0— 90 8D 4F 90 8D F0 90 60
90A8— A6 2B A0 50 BD 80 C0 98
90B0— 29 03 0A 05 2B AA BD 81
90B8— C0 A9 56 20 A8 FC 88 10
90C0— EB 60 A6 2B BD 80 C0 A0
90C8— 01 98 29 07 05 2B AA BD
90D0— 80 C0 A9 56 20 A8 FC BD
90D8— 7F C0 C8 C8 C0 A3 D0 E9
90E0— 60 8D 4F 90 20 ED FD A6
90E8— 2B 60 C4 D2 C9 D6 C5 BD
90F0— B1 00 CE CF A0 C4 D2 C9
90F8— D6 C5 A0 C3 C1 D2 C4 00
```

求法雷数列的新方法

湖北省煤炭工业学校(610064) 廖庆平

在数学中,一个有名的数列叫法雷数列。它出自著名数学家法雷之手,它的构成为:对于任何自然数 N ,

求出所有分母为小于或等于 N 的真分数,把这些真分数按顺序排列起来,并且在第一个分数的前面加上分数 $0/1$,在最后面加上分数 $1/1$,于是我们把得到的一组数,称之为 N 级法雷数列。

例: $F5 = \{0/1, 1/5, 1/4, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 1/1\}$

$F8 = \{0/1, 1/8, 1/7, 1/6, 1/5, 1/4, 2/7, 1/3, 3/8, 2/5, 3/7, 1/2, 4/7, 3/5, 5/8, 2/3, 5/7, 3/4, 4/5, 5/6, 6/7, 7/8, 1/1\}$

它在数学中有着许多重要的应用,现在我们只来研究它的构成规律。从上面可以看出,我们不能写出它们的通项,通过观察,发现它们之间仍服从一个简单的规律。设 $FA(I)$ 为数列中的分子, $FB(I)$ 为分母,则有:

$$FA(I)/FB(I) = (FA(I-1) + FA(I+1)) / (FB(I-1) + FB(I+1))$$

考虑到数列中的每一项是通过约分后的真分数,把左边分子、分母乘以一个整数 K ,则:

$$FA(I+1) = K * FA(I) - FA(I-1)$$

$$FB(I+1) = K * FB(I) - FB(I-1)$$

$$K = 0, 1, 2, 3, 4, \dots$$

由于对于任何前两项均为 $0/1$ 和 $1/N$, 这样我们可以按上规律推出后面诸项。

现给出算法法雷数列的 BASIC 程序如下:

```
10 INPUT N; DIM FA(4 * N), FB(4 * N), A(4 * N, N), B(4 * N, N)
15 PR # 1; PRINT
20 FA(1) = 0; FB(1) = 1; FA(2) = 1; FB(2) = N; I = 3
30 FOR K = 1 TO N
40 A(I, K) = K * FA(I-1) - FA(I-2)
50 B(I, K) = K * FB(I-1) - FB(I-2)
60 IF A(I, K) <= 0 OR B(I, K) <= 0 THEN 100
70 AA = A(I, K); BB = B(I, K); GOSUB 200
80 A(I, K) = AA/BB; B(I, K) = BB/BB
90 IF B(I, K) > N THEN 110
100 NEXT K
110 FA(I) = A(I, K-1); FB(I) = B(I, K-1)
120 IF FA(I)/FB(I) = 1 THEN GOTO 140
130 I = I + 1; GOTO 30
140 FOR K = 1 TO I: PRINT FA(K); "/"; FB(K); " ";
145 PP = PP + 1; IF PP = 13 THEN PP = 0; PRINT
150 NEXT K; END
200 R = AA - BB * INT(AA/BB)
210 IF R = 1 THEN BB = 1; RETURN
220 IF R = 0 THEN RETURN
230 AA = BB; BB = R; GOTO 200
```

]RUN

? 5

0/1 1/5 1/4 1/3 2/5 1/2 3/5 2/3 3/4 4/5
1/1

]RUN

78

0/1 1/8 1/7 1/6 1/5 1/4 2/7 1/3 3/8 2/5
3/7 1/2 4/7 3/5 5/8 2/3 5/7 3/4 4/5 5/6
6/7 7/8 1/1

TOOL-KIT 使用详解(续)

北大附中 姜宏

APA 就介绍到这里。关于如何扩展自己的 APA, 以及如何利用 & 命令扩充系统功能, 请参阅《电子与电脑》前几年的有关文章。顺便提一句, APA 中除了 &MANUAL 外, 其他命令都可用一个“&”字符和命令头一个字母代替, 如 &RENUMBER 用 &R; &XREF 用 &X, 但 &MANUAL 命令必须用 &MA 的简写方式。

3. 高分辨显示字符程序 HRCG

HRCG 是一个 R 型文件, 通过 LOADHRCG 程序运行。

开机键入 RUN LOADHRCG 后, 屏幕显示:

HOW MANY ALTERNATE CHARACTER SETS
WOULD YOU LIKE(0~9)?

这时, 你应键入你想好的字符集文件名(后缀为“.SET”的), 如果不准备用扩充字符集, 则请你键入 0, 回车。如果你输入了一个大于 0 或小于 9 的数, 那么屏幕还会出现:

NAME OF CHARACTER SET?

这时, 我们比如想输入希腊文(Greek)字母, 就键入 GREEK.SET。按顺序把所需的字符集依次调入内存后, 屏幕被转到高分辨率第一页, 显示:

HI-RES CHAR GEN VERSION 1.0

下面, 我们依次来说明 HRCG 系统的各个命令, 说明中用 ^A 表示同时按下 CTRL 和 A 键, 余者亦然。

(1) 字符集之间的切换

在启动系统时, 我们调入了几套字符集, 但屏幕上只能显示一套, 这就存在一个字符集之间切换的问题了, 怎么办呢?

^An 命令可以帮我们解决这个问题, 比如我们调入了三套字符集, ^A1 就可以把屏幕上字符显示改为第一套字符集的显示, 如果想返回标准字体, ^A0 即可。

您如果想在程序中切换字型, 用 PRINT CHR\$(1);n, 表示选择第几套安符集。

(2) 正相与反相显示的切换

为了使一些字符串引起用户的注意, 我们可以使用反相显示的功能, 键入:

^I 进入反相模式

^N 返回正相模式

^I 命令只不过在显示每一个字符时, 将它的每一个代码作逻辑运算 EOR # \$FF。

如果在程序中切换正反相显示字型, 用 PRINT CHR\$(9)代表 ^I, PRINT CHR\$(14)来代替 ^N。

(3) 大写与小写的切换

大写、小写的切换有以下命令:

^K 进入大写模式

^L 进入小写模式

^S 以后键入的第一个字母大写, 余者小写。

^Z 切换至最初状态, 也就是 0 号字型, 大写模式, 正相显示, 全屏幕显示。

在程序中, 用 CHR\$(19)代替 ^S, CHR\$(11)代替 ^K, CHR\$(12)代替 ^L, CHR\$(26)代替 ^Z。下例是一个综合运用几条命令的例子:

```
10 PRINT CHR$(1);2
20 PRINT CHR$(14)
30 PRINT CHR$(19)“I LOVE COMPUTER.”;PRINT
  CHR$(1);0
40 PRINT CHR$(11)“MY MOTHERLAND IS CHINA.”
50 PRINT CHR$(26);END
```

4. 屏幕的清除

有关屏幕清除的命令也有几条, 说明如下:

^P 清除窗口内的内容, 并将光标送回(1,1)位置。

^E 清除光标以右的所有字符。

^Q 将光标送回(1,1)位置, 不清屏。

^F 清除光标以后(包括该行右边和以下各行)的所有字符。

下面列一张表, 是 HRCG 和文本显示时的命令对照表:

HRCG 的命令	文本型的指令	用途
^P	HOME 或 ESC-@ CALL-936	清窗口字符
^E	ESC-E 或 CALL-868	清余行
^F	ESC-F 或 CALL-958	清余幅
^Z	TEXT	一切恢复正常显示

5. 窗口的设计

我们在显示文字时, 可以要求它只在一定范围内显示, 如同 TEXT 的 \$20-\$23 单元的作用一样, 所涉及到的命令及说明如下:

^Y 将窗口设为全屏幕。

^V 光标目前位置的右下方为窗口(定义窗口左上角)

^W 光标目前位置的左上方为窗口(定义窗口右下角)

6. 文字方阵的显示

如果我们想在一个特定的位置印一个 n×m 的文字方阵, 用普通方法相当繁, 如:

```
10 VTAB 10;HTAB 10;PRINT“ABC”
20 HTAB 10;PRINT“DEF”
```

```
30 HTAB 10;PRINT"GHI"
```

如果我们想只用一个 PRINT 解决文字方阵的问题,就要用文字方阵显示命令:

```
^B 进入方阵显示模式
^D 退出方阵显示模式
^C 显示中的换行
```

还以上面的程序为例,用文字方阵显示命令的程序就简单得多了:

```
10 VTAB 10;HTAB(10);PRINT CHR$(2);"ABC";CHR$(3);"DEF";CHR$(3);"GHI";CHR$(4);CHR$(13)
```

两个程序的运行结果,都是在屏幕(10,10)处显示:

```
ABC
DEF
GHI
```

7. 卷动显示及循环显示的切换:

屏幕上关于这部分的命令如下:

```
^O+^S 卷动方式
^O+^W 循环方式
```

HRCG 初次启动时,屏幕显示方式为卷动方式。

8. 显示页的选择

苹果机共有两个高分辨图形页,可用下面的命令来切换 HRCG 将字符印到哪一页上:

```
^O+^A 操作第一页,显示页不变。
^O+^B 操作第二页,显示页不变。
^O+^D 显示操作页。
```

9. 显示模式的选择

HRCG 可进行 NOT,OR,EOR,覆盖重写(正常)四种方式的显示:

```
^O+^P 覆盖重写
^O+^C EOR 方式
^O+^O OR 方式
```

至于 NOT 方式,就是常说的反显方式,用 ^I 实现。

比如,在 China 底下划一横线,变为 China,程序如下:

```
10 A$="China";PRINT CHR$(16)
20 B$="_"
30 PRINT CHR$(15);CHR$(15)
40 HTAB 1;VTAB 15;PRINT A$
50 HTAB 1;VTAB 15;PRINT B$
] RUN
China
```

10. 两页显示运算方式的选择
命令如下:

```
^O+^R 上下两页 EOR 在上页
^O+^T 上下两页 OR 在上页
^O+^P 正常印法
```

11. 使用自己的机器语言子程序

如果你在 HRCG 状态下想使用自己写的机器语言程序时,有如下两条命令可用:

```
^O+^Y 调用 A 子程序
^O+^E 调用 B 子程序
```

A 子程序 首地址的低 8 位 R+10

A 子程序 首地址的高 8 位 R+11

B 子程序 首地址的低 8 位 R+13

B 子程序 首地址的高 8 位 R+14

R 的位置在 48K 或 64K 机器中,通常为 \$8DFF (+进制的 36351)。

12. 一切恢复至起始状态

用 ^Z 可完成,它做的工作有:

- ①将字符集选择在 0 号,即 CHR\$(1);0
- ②进入大写模式 CHR\$(11)
- ③进入正相模式 CHR\$(14)
- ④全屏幕显示 CHR\$(25)
- ⑤操作页和显示页都为第一页 CHR\$(15);CHR\$(1);CHR\$(15);CHR\$(14)
- ⑥正常印字方法 CHR\$(15);CHR\$(16)
- ⑦进入卷动方式 CHR\$(15);CHR\$(19)
- ⑧A 子程序,B 子程序均为 RTS
- ⑨取消方阵显示 CHR\$(4)

13. 如何在自己的程序中调入 HRCG

HRCG 既然有如此多的好处,那么,你可以用下面的子程序在自己的程序调入它:

行号 ADRS=0

```
行号 PRINT CHR$(4);"BLOAD RBOOT";CALL 520;
      ADRS=USR(0),"HRCG"
```

行号 CALL ADRS

行号 RETURN

至此,HRCG 亦已介绍完。大家如能利用它做一些 CAI 的软件,那将是很不错的。

4. 高分辨字符集制造及编辑程序——ANIMATRIX 的使用与改进

Animatrix 是扩充 HRCG 工具的一个有效的工具程序,这个程序相当长,占用了 26 个扇区,其运行环境内存要求为 48K 或 64K,一台软驱。

在 "]" 提示符下键入 RUN ANIMATRIX 并回车后,屏幕提问:

Character set to edit?

要求你键入要编辑的字符集名。

这时,我们可键入一个字符集文件,比如说希腊字符集 GREEK.SET,键入后按回车键,现在,屏幕显示一个由 7×5 个 7×8 的小方阵组成的大方阵,右边有两个方框,上面一个是用于用户输入标准字型,下面一个是供编辑的字型。下面有一行提示:

Press control-I to display Instructions

意思是:按 CTRL-I 可以阅读提示。接上例,我们输入了 GREEK.SET,现在,我们按下小写的 A、B、C 三个字母(关于如何键入小写字母的问题已在第三部分 HRCG 的说明中谈过了)这时,上面的框显示图 1,下面的框显示图 2。

ABC

ABΓ

图 1

图 2

当我们编辑 "["、“\”、“_”三个字符时,用 CTRL-K 代替 "[" ,用 CTRL-L 代替 "\",用 CTRL-O 代替 "_"。这

在改进后的 APPLE 及 CEC-I 上,用键盘上已有的键就可输入它们。

①光标的移动

有几个功能键:

→将光标右移一格

←将光标左移一格

RETURN 换行

上移用←把光标移至行首后再按←键即可。

②字型的创作及修改

这部分可用游戏杆。

向上方推:光点向下移。

向下方推:光点向上移。

向左方推:光点向左移。

向右方推:光点向右移。

按钮 0:消除亮点。

按钮 1:加入亮点。

键盘上的功能键有:

CTRL-T 该行所有的点均移半点,这是水平方向 560 点分辨技巧的应用。

CTRL-A 大写。

CTRL-D 显示所有字型。

CTRL-I 显示提示

CTRL-S 字型复制:把光标所在的字符复制成键盘键入的字符。

③字型的存储

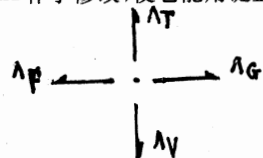
编辑好 ASCII 字符集后,只要键入 ESC, 屏幕提

问:

Name to save? (问你文件名取什么。)

当您输入文件名后,此字符集便被存盘。如果您误按了 ESC 键,则出现“Name to save?”后按回车再按 N, 回车即可回到编辑状态。

由于游戏杆操作十分不便,为此,可将 ANIMATRIX 作了修改,使它能用键盘操作,功能键如下:



擦光点: ^ W

写光点: ^ E

由于 ^ T 复盖了原 ^ T 的功能,因此,改过的 ANIMATRIX 用 ^ Y 代替原 ^ T。

仅将改过的部分列在下面:

```

]LIST 400
400 IF C=25 THEN GOSUB 630
]LIST 490,560
490 REM MOVE PX & PY
495 IF C=6 THEN PX=PX-1;PL=-1;IF PX<0
    THEN PX=48
497 IF C=7 THEN PX=PX+1;PL=-1;IF PX>48
    THEN PX=0
500 IF C=20 THEN PY=PY-1;PL=-1;IF PY<0
```

```

THEN PY=39
```

```

503 IF C=22 THEN PY=PY+1;PL=-1 IF PY>39
    THEN PY=0
```

```

505 P0=PX*4+2*INT(PX/7)+2;P1=PY*4+INT
    (PY/8)+1
```

```

508 XDRAW 2 AT P0,P1;FOR I=1 TO 30;NEXT;
    XDRAW 2 AT P0,P1
```

```

510 IF C=5 AND PL<>0 THEN PL=0;GOSUB 570
```

```

520 IF C=23 AND PL<>1 THEN PL=1;GOSUB 570
```

```

560 GOTO 290
```

键入程序后,再输入下面的命令:

```

]DEL 1394,1398
```

```

]SAVE ANIMATRIX VERSION KEYBOARD 即可。
```

改进的程序无论在速度上还是在使用上,都比原程序好得多。

TOOL-KIT 到这里已经全部介绍完了。希望您看了以后对您的工作有一些帮助。在国内,已经有了许多仿 TOOL-KIT 的工具软件,电子工业出版社出的《苹果机、中华学习机 CEC-I 工具箱》就是其中的一个很好的示范。笔者亦已完成了对 TOOL-KIT 的修改及扩充,试用后觉得效果不错。关于 TOOL-KIT 的扩充是很容易的,不少国内书刊都介绍了给 APPLE 加功能的程序,把它们依次输入,存盘后即是一个“软件”。

传播科技知识 推动科技进步

九月初举办“广州科技图书交易会”

由中国版协科技出版工作委员会、广州市科委、市科协、广州市新华书店主办的“广州科技图书交易会”将于今年九月三日至十三日在中国出口商品交易会九号馆举行。

本次科技图书交易会是全国历次科技图书展销规模最大、图书知识门类最齐、品种最丰富的一次,参展的科学技术图书有来自中央部属的、高等院校的和各省(区)市的近百家科技出版社,参展还有来自香港的万里机构出版有限公司,荟萃了国内版和港台版科技图书近二万个品种。

为了方便单位选购图书,在交易会同时展出几千种文史哲类图书供参加单位选购。

在展销期间,还将由广州市科委、市科协组织举办有关科学技术专题讲座和咨询活动,并设立为外地单位读者代办托运等服务项目和进行书店与出版社之间的订货业务。

联系地址:广州市北京路 336 号广州科技书店
邮政编码:510030 电话号码:3345723

第七章 循环程序设计

南京大学大气科学系(210008) 朱国江

§3 循环程序设计

在科学技术和生活领域里,常常遇到一些需要大量重复计算的课题,人们所关心的是如何减少一些不必要的重复性工作,以节省存储单元和提高计算效率。

完成大量的在处理形式上完全相同的重复性工作,这就是循环。在程序中,对某一段内容反复执行多次,这就是循环程序。而在一个循环程序中,包含另一个或几个循环,则称为循环的嵌套或多重循环。

循环程序的结构包括以下几个部份:

- 初始化部份,它建立计数器,地址寄存器(指针)和有关变量的起始值。如累加器清零、源数据块首地址、数据块长度、内外循环计数器、循环次数等量初值。

- 处理部份,在这部份进行实际的数据处理,这是循环的工作部份,它完成重复执行程序段的内容。

- 循环控制部份,它检查每循环一次是否结束,并为下一轮循环做好修正计数器和指针的准备。

- 结束部份,它分析、存放或显示结果,结束。

循环程序对于处理数据块的传送问题比较方便,而采用变址寻址方式处理动态数据块和多个数据块则较为理想,因为变址寻址方式允许用改变变址寄存器内容的方法来改变存储器的实际地址。为此,程序必须在每一轮处理后将变址寄存器内容加1,使它指向数据块的下一个元素,这样就使下一轮对下一个存储单元的数据执行相同的操作,因而可用同一个程序段来处理256个字节长度的数据块(因变址寄存器是8位字长)。

另外,我们将从下面所举的实例中看到,在循环程序中大量的包含了程序分支内容,因为每循环执行一次后是继续重新循环呢?还是不再循环而往下执行新的程序段呢?这里就存在着控制循环的程序分支,以根据某种条件满足与否决定程序的走向,控制语句的执行路径。因此,我们可以看到分支和循环的概念是不同的,对于一些单纯的分支程序,其中每条指令可能只执行一次而不进行循环,而循环程序的处理部分则在程序执行过程中将被执行许多次。同时,我们也看到,单纯分支不包括循环,循环却一定包含分支,它们既有区别又有联系,而6502指令系统中的比较指令和条件转移指令,为设计分支程序和循环程序提供了方便。

[例1]求数据的累加和

计算一串二进制的和,这串数的长度放在\$06单元,而这串数从\$6001单元开始存放。将和数放在\$07单元,设和数是一个8二进制位数,这样可以不考虑进位。见程序7.1。

程序7.1:

1000- 20 58 FC	JSR	\$FC58	;清屏
1003- A9 00	LDA	# \$00	;和置初值
1005- AA	TAX		;计数器置初值
1006- D8	CLD		;二进制求和,清十进制标志位
1007- 18	CLC		;清进位位 C
1008- 7D 01 60	ADC	\$6001,X	;和=和+数据
100B- E8	INX		;计数器加1
100C- E4 06	CPX	\$06	;次数够了吗?
100E- D0 F7	BNE	\$1007	;不够,继续循环
1010- 85 07	STA	\$07	;够了,送结果
1012- A5 07	LDA	\$07	;取结果→A
1014- 20 DA FD	JSR	\$FDDA	;显示结果
1017- 60	RTS		;结束

程序7.1 设计要点:

- 初始化:和数放累加器A中,初始值为0,寄存器X当计数器,未加前为0,清10进制标志和清进位位标志。

- 循环:取数相加,变址计数器加1指向下一个地址的数据,判断数据是否加完,未完继续循环求和,是则送结果并显示

- 结束:从\$07单元取结果,调显示字符(16进制)子程序,送现行输出设备上,结束。

运行前: * 6:05 设串长度为5

* 6001:12 34 1D 56 20 5个数据

运行后: * 1000G✓

* D9

* 7✓

* 0007-D9

关于程序7.1的几点说明:

- 本程序原则上可以完成N个数的重复相加,数据个数放\$06单元,并可灵活更改。

- N个数的累加和不能超过\$FF(255),否则结果不正确。

- 是一个单循环,数据的累加是从前向后进行的。

- 循环中包括分支,主要由执行比较指令CPX \$06后,P寄存器中的零标志Z值决定BNE指令是否转移。而循环的次数由指令INX,CPX,BNE共同决定。

对N个数据的累加和,也可以如下编制,见程序7.2。

程序7.2:

1000- 20 58 FC	JSR	\$FC58
1003- A9 00	LDA	# \$00
1005- AE 00 60	LDX	\$6000

```

1008- D8      CLD
1009- 18      CLC
100A- 7D 00 60  ADC  $ 6000,X
100D- CA      DEX
100E- D0 FA    BNE  $ 100A
1010- 8D 30 10  STA  $ 1030
1013- AD 30 10  LDA  $ 1030
1016- 20 DA FD  JSR  $ FDDA
1019- 60      RTS

```

运行前: * 6000:05✓

* 6001:12 34 1D 56 20✓

运行后: * 1000G✓

* D9

* 1030✓

* 1030-D9

程序 7.1 和 7.2 的异同点:

- 共同点,都是用单一循环编程,完成和数不超过 8 位二进制数的 N 个数累加。

- 主要区别,设计思想不同,程序 7.1 求和时是从前面往后面加的,而程序 7.2 求和时则是从后面向前面加。

此外,我们还看到在选用 ADC 指令时,其操作数部份的基址不同,程序 7.1 中用的是 \$ 6001,程序 7.2 中用的是 \$ 6000,这是两种程序不同算法和字符串长度选用的不同单元决定的。这是编写程序时应该注意的重要细节。

最后,我们还应指出,这两个程序在编制结构上虽然具有一定的灵活性(可以进行 N 个数的累加和),但通用性不够(和数不能超过一个 8 位二进制数)。解决的思路是设法保存进位的值或采用多字节加法程序。

[例 2]找最大值

在一段数据中寻找最大的元素。设此段数据个数放在 \$ 1A 单元, \$ 1B—\$ 1E 单元中存放数据,要求结果放在 \$ 1F 单元。并设各数据皆为无符号的 8 位二进制数。见程序 7.3。

```

1000- A6 1A    LDX  $ 1A      ;计数器初值为数的个数
1002- A9 00    LDA  # $ 00    ;设最大值=0
1004- D5 1A    CMP  $ 1A,X    ;下一个数>最大值吗?
1006- B0 02    BCS  $ 100A    ;否,维持 A 中最大值不变
1008- B5 1A    LDA  $ 1A,X    ;是用该数代替最大值
100A- CA      DEX             ;X-1
100B- D0 F7    BNE  $ 1004    ;所有数都比较完了? 否,循环
100D- 85 1F    STA  $ 1F      ;是,存最大值
100F- 60      RTS             ;结束

```

运行前: * 1A:04 05 FF 15 E3

运行后: * 1000G✓

* 1F✓

* 001F-FF

程序 7.3 虽然较短,但因用了比较指令和两条条件转移指令,使程序阅读有些困难,下面作一些简单解释。

开始, X 寄存器作为计数器用置初值 $(1A) \rightarrow X$, 即 $X=04$, 接着在累加器 A 中放了一个立即数 00, 表示找数开始前假想的一个最大值为 0, 执行第三条指令 $CMP \$ 1A, X$ 时, 表示 $(A) - (1A + X) = (A) - (1E) = 0 - E3$, 显然不够减, P 寄存器中的 C 标志置 0, 执行第四条指令 BCS 时, 由于执行前一条指令后 $C=0$, 所以不发生转移而执行下一条指令 $LDA \$ 1A, X$, 即将 $(1A + X) \rightarrow A$, $(1E) \rightarrow A$, $(A) = E3$, 可见这时已用 E3 代替原来假想的最大值 0。执行第六条指令 $X-1 \rightarrow X$, 只要 X 不为 0, 则执行 BNE 指令后循环上去转至 \$ 1004, 由于此时 $X=03$, 故再次执行 CMP 指令, $(A) - (1A + X) = (A) - (1D) = E3 - 15$, 够减, $C=1$, 又执行 BCS 指令, 因此此时 $C=1$, 故发生转移到 \$ 100A, 注意此时 A 中仍为 E3, 维持最大值不变。X 又减 1, 为 02, 不为 0, $Z=0$, 又循环上去。执行 CMP 指令, $(A) - (1A + X) = (A) - (1C) = E3 - FF$, 不够减, $C=0$, 执行 BCS 指令时由于 $C=0$ 执行下一条指令 $LDA \$ 1A, X$, 即将 $(1A + X) \rightarrow A$, $(1A + 02) \rightarrow A$, $(1C) = A$, $(A) = FF$, 可见这时已用更大值 FF 代替次大值 E3。……, 下面的过程重复执行上述类似的步骤, 每一次循环后都将更大的数放入 A 中, 不断循环, 直至比较完所有的数, 最后将最大值(真正的)放入 \$ 1F 中, 结束。

综上所述,分析程序时应注意:

- 什么时候循环,循环多少次,以及何时退出循环,都由实现分支的条件语句控制着,循环中包含着分支。

- 比较指令 CMP 表示 $A-M$, 够减 $C=1$, 不够减 $C=0$; BCS 指令在 $C=1$ 时转移, $C=0$ 时执行下一条指令; BNE 指令表示结果不为 0 ($Z=0$) 时转移, 而结果为 0 时 ($Z=1$) 继续执行下一条指令。记住这些对看懂程序大有好处。

[例 3]确定负数的个数

已知数据块的长度放在 \$ 6000 单元中, 而数据块本身从存储单元 \$ 6001 开始存放, 要求将其中负数的个数放在 \$ 07 单元。

如: 6000:06 68 F2 87 30 59 2A

因 \$ 6002 和 \$ 6003 两个单元的值均为负数, 所以程序设计运行后的结果应有: $(07) = 02$ 。

分析:

- 确定一个数据块中负数的个数, 就是查找数据块中数的最高位为 1 的个数。

- 由于本例中只有正数和负数两种, 而没有零值, 因此, 选用 BPL 指令将是合适的, 因为它以 N 标志作为判别标准, 若 $N=0$, 表示正数, 执行分支动作; 若 $N=1$, 表示负数, 则执行下一条指令。

- 由于本例中有 6 个数(为简单起见只选了 6 个数, 原则上可以有 N 个数, $N < 256$), 必须逐一比较, 检查才能确定那些是负数, 那些是正数, 而这种重复性的比较显然可用循环的方法处理。

- 在选用 BPL 指令之前, 选用取数指令 $LDA \$ 6001, X$ 也是顺理成章, 一是因为只有取出一个数

来,才好判别是否负数;二是 LDA 指令执行后对符号位 N 及零标志 Z 皆有影响;三是采用绝对 X 变址寻址方式的 LDA 指令,只要改变 X 值,即可改变指向下一个数据的地址。

• 为了控制循环的次数,必须设置计数器,例如选用 X 寄存器作为计数器。为了统计负数的个数,也应设置一个负数计数器,例如选用 Y 寄存器。

根据上述分析,容易写出源程序 7.4。

程序 7.4

```
1000- A2 00      LDX  # $00      ;计数器置初值
1002- A0 00      LDY  # $00      ;负数计数器置初值
                                   ;值
1004- BD 01 60   LDA  $6001,X    ;取一个数
1007- 10 01      BPL  $100A      ;是负数吗? 否,转
1009- C8         INY             ;是,负数个数加 1
100A- E8         INX             ;计数器加 1
100B- EC 00 60   CPX  $6000      ;所有数都检查完了吗?
100E- DO F4      BNE  $1004      ;否,继续取数检查
1010- 84 07      STY  $07        ;是,存负数个数
1012- 98         TYA             ;Y→A
1013- 20 DA FD   JSR  $FDDA      ;显示
1016- 60         RTS            ;结束
```

运行前: * 6000:06 68 F2 87 30 59 2A

运行后: * 1000G✓

* 02 ✓

* 07 ✓

* 0007-02

讨论:若要确定正数的个数,如何编程。

这个问题同确定负数的个数相类似,可以仿照程序 7.4 的模式编程。但最为简捷的方法是只要改动一个指令,您能指出来吗?

在您思考之前,最好先不要看下面的结果。

事实上,只要将程序 7.4 中的 BPL 指令,改为 BMI 指令。因为,这两条指令虽然都以 N 标志作为判别的依据,但两者转移的条件是不同的。BMI 指令是当 N=1 时,表示负数转移;而当 N=0 时,表示正数继续执行下一条指令。

实际改动时,只要改动程序 7.4 中 \$1007 单元的值,即将 10 改为 30,这是因为两条指令的操作码不同,代表的意义相异。

当然,经改动后,Y 寄存器中存放的是正数的个数,显示出来的也是正数的个数。

[例 4]确定正数、负数和 0 的个数

设数据块长度为 06,要求这 6 个数中逐一确定正数、负数和零的个数。

分析:

• 这个问题相对于仅仅判定一段数据中负数的个数或正数的个数来说,要复杂一些。因为,当读取一个数据时,要做几种判断。

• 可以选用 BNE 指令,先判断是否为 0,若是 0,则记录 0 的个数;若不是 0,再用 BPL 指令,以判别正、负数。在用 BPL 指令判别正、负数时,是负数立即存其

个数;不是负数跳转并存放正数个数。

• 为了统计数据块中正数、负数和零的个数,必须设置三个暂存单元,如负数、零、正数的个数分别存放在 \$10, \$11 和 \$12 单元中,在未统计各自个数前,应使这三个单元为 0,这就是程序的初始化条件。

• 在每一个数的性质判定以后,要做两件事,一是存每一个数的个数,这可以通过存储单元内容加 1 来解决,如用 INC \$10, INC \$11 或 INC \$12;二是让计数器加 1,并判所有的数是否处理完,这可以用 INX 和 CPX \$03 两条指令完成,其中 \$03 单元为存放数据块的长度,X 寄存器为计数器,当所有数都判别完以后结束,而只要有一个数未判完,即应循环下去再判。初始化时也应设置 X=0。

• 每判定一个数的性质以后,都要比较是否已是最后一个数,这可以用一个共同的子程序来实现,这样做可以减少相同操作的编程,使程序简化。关于子程序的设计我们将在下一节介绍。数据放在 \$04—\$09 单元中。

由上分析,不难写出源程序 7.5。

程序 7.5:

```
1000- A9 00      LDA  # $00      初始化,累加器,
1002- 85 10      STA  $10        正、负、零的
1004- 85 11      STA  $11        个数存贮单元,
1006- 85 12      STA  $12        计数器置 0。
1008- A2 00      LDX  # $00
100A- B5 04      LDA  $04,X      ;取一个数
100C- D0 06      BNE  $1014      ;非 0,转
100E- E6 11      INC  $11        ;是 0,0 的个数加 1
1010- 20 1E 10   JSR  $101E      ;转子程序
1013- 60         RTS            ;返回
1014- 10 06      BPL  $101C      ;不是负数,转
1016- E6 10      INC  $10        ;是负数,负数个数加 1
1018- 20 1E 10   JSR  $101E      ;转子程序
101B- 60         RTS            ;返回
101C- E6 12      INC  $12        ;正数个数加 1
101E- E8         INX             ;计数器加 1
101F- E4 03      CPX  $03        ;所有数比较完吗?
1021- D0 E7      BNE  $100A      ;否,循环
1023- 60         RTS            ;是,结束
```

运行前: * 03:06 68 F2 87 00 59 2A

运行后: * 1000G✓

* 10.12✓

* 0010-02 01 03

程序 7.5 编制中有几点技巧:

• 一是仅用一条指令 LDA # \$00,给三个存储单元 \$10, \$11, \$12 赋初值 0,而不是用三条 LDA # \$00 指令,这样做程序清晰,又节省内存。

• 二是调用 \$101E—\$1023 单元的子程序并正确返回。因为在判断一个数的性质以后,计数器指针加 1,以便正确指向下一个数据;同时,就程序的整体而言,还必须判别是否比较完全体数据,因此,编制一段具有共同功能的子程序实为必要,目的已如前所述。

·注意了主程序和子程序之间的正确衔接。调用子程序后必须注意正确返回,这不仅要在\$1023单元放置一条由子程序返回到主程序断点处的RTS指令;而且对本程序来说,\$1013和\$101B两个单元也必须放置RTS指令,否则程序不能正确运行。

[例5]数据插入

假定存储单元\$6000中的内容6B,尚未出现在序列中,则把该单元内容增加到此序列中。已知序列的长度存放在\$6001单元,而序列本身从\$6002单元开始存放。

示范题:(6000)=6B

(6001)=04

(6002)=37

(6003)=61

(6004)=38

(6005)=1D

结果:(6001)=05

(6006)=6B

该单元(6B)被附加到序列中,因原序列中没有此单元,这个序列的长度增加1。见程序7.6。

程序7.6:

```
1000- AD 00 60    LDA    $ 6000    ;取欲插入的数
1003- AC 01 60    LDY    $ 6001    ;取序列长度
1006- D9 01 60    CMP    $ 6001,Y  ;比较,相同吗?
1009- F0 0F      BEQ    $ 101A    ;是的,转$101A
100B- 88          DEY            ;否,Y-1→Y
100C- D0 F8      BNE    $ 1006    ;所有数都比较完
                                   ;吗?否,转
100E- EE 01 60    INC    $ 6001    ;是,序列长度加1
1011- AC 01 60    LDY    $ 6001    ;放增加后的长度
1014- 99 01 60    STA    $ 6001,Y  ;取增加的数
1017- 20 DA FD    JSR    $ FDDA    ;显示
101A- 60          RTS            ;结束
```

运行前: * 6000:6B 04 37 61 38 1D

运行后: * 1000G✓

* 6B

* 6000.6006✓

* 6000-6B 05 37 61 38 1D 6B

程序7.6编制要点:

·取欲插入的数,与序列中的其它数相比较,如果序列中原来就有和插入的数相同的数则结束。

·取欲插入的数,与序列中的数逐一比较,只要有一个数不同,就将长度减1,只要未比较完全部序列的数,就循环再比,直到比较完全部数据,原数据序列长度加1,并显示插入的结果。

说明:程序7.6也可采用绝对X变址寻址方式安排CMP指令 and STA指令,同时改LDY \$6001指令为LDX \$6001指令,即程序中所有有Y的地方,改成X,并改正相应指令的操作码,也能正确运行。

[例6]单个数据块的传送

将存放在\$6000-\$60FF地址单元内的256个数据顺序传送到\$7000-\$70FF单元中去。

分析:本题虽然数据量较大(256个),但设计思想比较简单,取一个数,送一个数,计数器加1,指向下一个数据地址,再取,再送,只要没有取完送完,就继续循环,反之结束。故有程序7.7。

程序7.7:

```
1000- A2 00      LDX    # $ 00    ;计数器置初值
1002- BD 00 60   LDA    $ 6000,X  ;取数
1005- 9D 00 70   STA    $ 7000,X  ;送数
1008- E8          INX            ;计数器加1
1009- D0 F7      BNE    $ 1002    ;送完了吗?否,转
100B- 60          RTS            ;是的,结束
```

如果我们将题目改一下,将\$6000-\$60FF单元中的数据块按相反顺序传送到\$7000-\$70FF单元中去,程序又是如何编制呢?

这个题目的解题方法很多,今摘其一如程序7.8。

程序7.8:

```
1000- A2 00      LDX    # $ 00    ;源数据计数器初值
1002- A0 FF      LDX    # $ FF    ;目的数据计数器初值
1004- BD 00 60   LDA    $ 6000,X  ;取源数据
1007- 99 00 70   STA    $ 7000,Y  ;送入目的地址
100A- E8          INX            ;源计数器加1
100B- 88          DEY            ;目的计数器减1
100C- D0 F6      BNE    $ 1004    ;次数不够,循环
100E- 60          RTS            ;次数够了,结束
```

程序中用了三条取数指令和一条存数指令,并选用了两个变址寄存器X和Y,作为源数据块计数器和目的数据块计数器用。应该指出,变址寄存器X(或Y)都是八位寄存器,在编程中常被当作一个计数器来用。它们可以由指令控制而被置成一个常数,并能方便地用加1(INX,INY)、减1(DEX,DEY)、比较(CPX,CPY)操作来修改和测试其内容,从而使得程序能够方便灵活地处理数据块、修改地址指针、控制循环次数等。而在程序7.8中由于要处理数据块按逆序传送,因而一个变址寄存器就显得不够用,所以既用了变址寄存器X,又用了变址寄存器Y。

[例7]多个数据块的传送

将\$6000-\$600F单元的第一个数据块送到\$7000-\$700F单元中,将\$6100-\$610F单元的第二个数据块传送到\$7100-\$710F单元中,将\$6200-\$620F单元中的第三个数据块传送到\$7200-\$720F单元中去。

分析:这是三个数据块的传送问题,由于是顺序搬迁,而且数据块长度相同(都是16个),使得问题求解不至于过分复杂。数据块可以一个一个送,送一个数据块就是一个循环,当送完一个数据块后只要改变一些地址指针,就可以再送另一个数据块,直到全部送完。因此,一个循环程序是处理不了的,所以本题应是一个多重循环问题。

我们先给出源程序清单,见程序7.9,然后看一下运行结果,最后再对程序编制思想作一说明。

程序 7.9:

```

1000- A2 00    LDX  # $ 00    ;变址计数器 X 置 0
1002- A0 10    LDY  # $ 10    ;数据块长度送 Y 寄存器
1004- A1 1A    LDA  ($ 1A,X)  ;传送一个源数据到目的地址第一次将(6000)→7000
1006- 81 FA    STA  ($ FA,X)  ;修改源地址,使之增 1
1008- F6 1A    INC  $ 1A,X    ;修改目的地地址,使之增 1
100A- F6 FA    INC  $ FA,X    ;一个数据块传送完了吗?
100C- 88      DEY            ;未送回,循环
100D- D0 F5    BNE  $ 1004    ;三个数据块传送完了吗?
100F- E0 04    CPX  # $ 04    ;是,结束
1011- F0 05    BEQ  $ 1018    ;否,将 X 增 2,为取下一个数据块作备
1013- E8      INX            ;
1014- E8      INX            ;转入下一个数据块
1015- 4C 02 10 JMP  $ 1002
1018- 60      RTS            ;结束

```

运行前: * 1A:00 60 00 61 00 62

将三个源数据块首地址送入 \$ 1A- \$ 1F 单元

* FA:00 70 00 71 00 72

将三个目的块首地址送入 \$ FA- \$ FF 单元

* 6000.600F

6000- 01 01 01 01 01 01 01 01

6008- 01 01 01 01 01 01 01 01

* 6100.610F

6100- 02 02 02 02 02 02 02 02

6108- 02 02 02 02 02 02 02 02

* 6200.620F

6200- 03 03 03 03 03 03 03 03

6208- 03 03 03 03 03 03 03 03

将第 1,2,3 个源数据块的数据分别送入 \$ 6000—\$ 600F, \$ 6100—\$ 610F, \$ 6200—\$ 620F 单元中。

运行后: * 1000G

* 7000.700F

7000- 01 01 01 01 01 01 01 01

7008- 01 01 01 01 01 01 01 01

* 7100.710F

7100- 02 02 02 02 02 02 02 02

7108- 02 02 02 02 02 02 02 02

* 7200.720F

7200- 03 03 03 03 03 03 03 03

7208- 03 03 03 03 03 03 03 03

完成了正确传送。

程序 7.9 编制思想说明:

• 源数据块的首地址放在 \$ 1A—\$ 1F 单元,目的数据块的首地址放在 \$ FA—\$ FF 单元,这些单元都是采用零页地址单元。这样处理比用多组 LDA 和 STA 指令设置地址单元要简捷得多,同时也使初始化部份省去了不少语句。

• 程序中的第 1,2 两条指令,是安排了两个计数器,其中 X 变址寄存器作为修改源地址和目的地址的指针计数器,为读取下一个数据作准备,同时又兼作三个数据块是否全部送完的一个标志,配合 CPX # \$ 04 来处理。Y 变址寄存器则作为传送数据块长度的计数器,和 DEY, BNE 指令配合,以使决定一个数据块的所有数据是否全部送完。

• 程序中有两个循环,第一个是从 \$ 1002—\$ 1007 的外循环,第二个是从 \$ 1004—\$ 100E 的内循环。它们完成的任务是不同的,内循环主要完成一个数据块(16 个数据)的传送工作,而外循环则是完成三个数据块的传送工作。

• 循环程序中包含了分支转移,在内循环中,控制循环次数,判断一个数据块的所有数据是否全部送完,主要由条件转移指令 BNE 决定的(是否分支转移)。而在外循环中判断三个数据块是否全部送完则主要由条件转移指令 BEQ 决定的(送完结束,未送完修改地址指针再循环)。

• 程序中数据块的传送,主要选用了 LDA(\$ 1A,X) 和 STA(\$ FA,X) 这两条指令,它们都是采用先变址(X)间接寻址的寻址方式,其优点是,使得多个数据块的处理变得简单、容易。一般来说,处理各个数据块的传送问题,用先变址间接寻址方式的指令比较理想。当然也不是绝对的,本章的下面几节内容还将介绍多个数据块的传送方法,那时,我们还可以看到不少的编程方法和技巧。

关于循环程序的设计,我们暂介绍到这里。现在,对本节的内容作一简单小结。

• 循环程序的任务,在于完成大量的在处理形式上完全相同的重复性的计算工作。

• 循环程序的结构,包括初始化、处理、循环控制、结束四个部份。各部份有机结合,协同动作,缺一不可。

• 循环程序的应用,它几乎能解决名目繁多,,形式各异的各种课题,是机器语言程序设计的核心和精华。

• 循环包括分支,分支控制循环,两者互相联系又相互制约。灵活而准确地使用比较指令,特别是条件转移指令,是高质量设计循环程序所必不可少的。

• 循环程序的嵌套,是指循环内还可以嵌套循环,一个循环内可以包含几个循环,,但应注意内外循环的层次要清楚,内循环允许并列几个小循环,但不允许交叉。

初级程序员级软件考试辅导

PC BASIC 自测试题解答与分析

北京市计算中心(100005) 李志刚

〔第一题〕这是一组填空题,共有五个小题。

(1)要求将给出的代数表达式改写成 BASIC 算术表达式。在改写时要将表达式中的代数运算符和函数转换成相应的 BASIC 的算术运算符和函数,应特别注意:①将三角函数中出现的度数利用公式“度数 * (3.1416/180)”化成弧度。如 $\sin 45^\circ$ 改写成 BASIC 表达式为 $\sin(45 * 3.1416/180)$ 。②在书写时两个因数间的乘号 * 不能省略或用 · 表示。完整的 BASIC 表达式为 $(2 * \sin(45 * 3.1416/180) + \cos(30 * 3.1416/180)) / (3 * \text{ABS}(X) * \text{LOG}(Z - 5))$ 。

(2)SWAP X, Y 语句的作用是交换数据存储变量 X 和 Y 的值。这是一条常用语句,在某些 BASIC 版本中没有这条语句,为了实现同样的功能,就要用一组赋值语句来实现。由于变量被赋值后,原有内容会丢失,因此,交换两个变量的内容(值)须借助一个临时变量。交换过程是这样的:先将第一个变量的内容赋值给临时变量保存,再把第二个变量赋值给第一个变量,然后再将临时变量的内容赋值给第二个变量,由此完成了两个变量内容的交换。本题语句组可以是:

$Z = X; X = Y; Y = Z$

(3)这是一个三层嵌套的函数表达式。按照 BASIC 表达式中算符的优先次序,首先计算最里层的函数值,得到字符串“1245”最右边长度为 2 的子串“45”,再计算 $\text{VAL}("45")$ 得数字值 45,然后求开方函数 $\text{SQR}(45 + 4)$ 得结果为 7。

(4)把当前驻留在内存中的 BASIC 程序写入磁盘有以下几种方式:

- ①SAVE “文件名”
- ②SAVE “文件名”, A
- ③SAVE “文件名”, P

其中,文件名包含了盘符和完整路径。方式①使程序以压缩二进制格式存储。这种方式节省存储空间,程序可运行,可在 BASIC 下列表及编辑,不能在 DOS 下显示或打印。方式②是最常用方式,参数 A 使程序做为一串 ASCII 码字符存储,在任何环境下均可编辑、列表和打印,也可以做为数据文件读入。方式③较少使用, P 参数把程序以编码二进制格式存储。这种方式保存的程序不能修改、显示和打印,可以对源程序在某种程度上起到保护作用。按题目要求,命令为 $\text{SAVE "A: PROG. BAS"}$, A 也可以写成 SAVE "A: PROG" , A, 如省去扩展名“.BAS”,系统在这种场合下会自动加上。

(5)栈是一种所有插入和删除均限定只能在一端进行的特殊类型的线性表,它遵从后进先出的操作规

则。在设计 BASIC 的计数循环语句 FOR/NEXT 时采用了堆栈技术,因此本题答案应填写“后进先出”。

〔第二题〕本题有五道选择填空题

(1)流程图是用规定的图形、连线和文字说明表示算法的一种图形。由于流程图直观、清晰、易读易交流,所以目前使用较为广泛。标准流程图是指由我国国家标准局批准的国家标准 GB1526—89,即按《信息处理—数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定》而画出的流程图。GB2312—80 是汉字信息交换用的国家标准;ISO 5807—85 是指国际标准化组织公布的标准《ISO5807—85 Information processing—documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts》。本题答案是 B。

(2)不论用何种计算机语言编制程序,一般都离不开顺序、分支(条件)和循环这三种基本的控制结构。采用自顶向下,逐步求精的结构化程序设计技术,并运用三种基本控制结构,可以设计满足各种复杂功能需求的程序结构。这三种基本控制结构的共同点是只有一个入口和一个出口。采用基本结构按照结构化程序设计方法编制的程序,不仅可读性强,也易于维护和扩充。本题的答案是 B、G。

(3)结构化程序设计方法已被人们广泛采用,它的基本思想是自顶向下,逐步求精。例如,把一个较复杂的问题逐步分解成若干个较简单或功能单一的问题,对每一个这样的问题,可以用程序的基本结构实现算法,这种方法就称为逐步求精法。本题答案是 D。

(4)按照结构化方法编制程序,根据模块(结构)只有一个入口、一个出口的原则,模块间的调用关系一般不用 GOTO 语句实现,这样程序不仅可读性强,还避免了因过多或不适当使用 GOTO 语句而产生的控制逻辑错误。在模块内部在不使用 GOTO 语句将产生大量程序冗余的情况下,也可适当地采用 GOTO 语句。本题答案是 D。

(5)在主程序和外部子程序之间的信息传递一般是通过一组变量来实现。在主程序中调用子程序语句内的变量表,一般称为形参;相应地在子程序头也必须要有与调用语句变量表在个数、顺序和类型上相匹配的一组变量,称为实际参数。本题的答案是 B、D。

〔第三题〕本题有五段简单的 BASIC 程序,先阅读程序,然后指出程序运行后生成的结果。

(1)阅读这段程序, 要掌握 READ, DATA 和

RESTORE 语句组的实现功能和特点。在 READ 语句中变量的类型与 DATA 语句中数据的类型要一一对应，DATA 中的数据个数不得少于 READ 读取的变量次数。DATA 是不可执行的静态语句，它可出现在程序的任何部位，数据项的顺序只与 DATA 语句排列的先后顺序有关，可以理解为在程序执行前，系统已把全部数据项按各 DATA 语出现的先后和 DATA 中数据的自然顺序放在 DATA 数据区中，同时将读取指针指向第一个数据，如下图

读取指针								
DATA 数据区		1	2	3	4	5	6	7
10	N=5	60		M=5		60		M=4
20	M=6	70		M<>1		70		M<>1
30	F=1							
40	GOSUB 60	80		F=5!		80		F=4!
		90		M=6		90		M=5
50	PRINT F;END	100		RETURN		100		RETURN

(3)逻辑运算符把一些真值或假值连接起来，运算的结果也是具有真值或假值的逻辑值。正如关系运算符通常用来决定程序的流向一样，逻辑运算符也用来连接两个或多个逻辑变量，多在 IF 语句中使用，由逻辑表达式的值来确定程序的流向。在本程序中，真值为 -1，假为 0。可利用逻辑运算的性质，由程序执行的次序，列出下列真值表，由表很容易得到答案是 C。

循环次数	A	B	A AND B	NOT A	(A AND B) + (NOT A)
1	-1	-1	-1	0	-1
2	-1	0	0	0	0
3	0	-1	0	-1	-1
4	0	0	0	-1	-1

(4)BASIC 中的函数主要有标准函数与自定义函数两类。本题主要是考虑对自定义函数的理解和使用。自定义函数是用户为解决自己的特定问题而定义的非标准函数，用 DEF FN 语句来完成定义。一经定义就可以象标准函数一样在表达式中调用。本题 20 语句定义了一个 A 函数，然后由 30 和 50 语句分别调用。在读程序写结果时，不仅输出数据要正确，输出格式也要符合要求才行。40 和 60 打印语句格式，在答案中，只有 D 满足要求，再由自定义函数分别计算出 30 和 50 赋值语句，结果为 269 和 130.5。答案是 D。

(5)这是一段图形打印程序，它的主要特征在打印控制部分。研究这部分程序就能很快确定正确答案。20 语句到 50 语句是打印部分，利用 20 到 40 语句的计数循环结构印出一行“*”字符，50 语句空打印产生回车换行。每个打印行的起始列是第 1 列，第 i 行 ($1 \leq i \leq 5$) 印出 i 个“*”字符。由 10, 60, 70 和 80 语句中 IF 和 GOTO 组成了一个直到型循环，控制图形印完五行后程序结束。答案是 B。

〔第四题〕提供了四个具有语法错误的程序段，执行后系统将显示不同的出错信息，阅读出错信息并弄清含意后再研究各程序段，找出程序出错点并与各出错信息比较，就可确定正确答案。如果熟悉出错信息，

语句 10 将 1, 2, 3 分别读入 A, B, C 中；指针指向 4，语句 20 的 RESTORE 恢复指针指向 1；30 语句不执行，40 语句分别将 1, 2, 3, 4, 4, 5 读入 G, F, E, D, C, B 中；50 语句不执行；60 语句打印变量 A~G 内的值，结果是 1, 5, 4, 4, 3, 2, 1。答案是 C。

(2)这是一段用递归调用方法计算 N! 的程序，主要利用了主—子程序间调用和返回的关系。为便于了解执行过程，将程序的执行序列图示如下，答案是 C, F 值为 $5! = 120$ 。

60	M=3	60	M=2	60	M=1
70	M<>1	70	M<>1	80	M=1*1
				90	M=2
80	F=3!	80	F=2!	100	RETURN
90	M=4	90	M=3		
100	RETURN	100	RETURN		

又有 BASIC 的基本知识，这类题目较易完成。

(1)这是执行两次的一层计数循环结构，在循环体内的 20 语句是对数组 A 进行维数定义。在两次循环中，数组 A 被定义了两次。BASIC 系统不允许对同一数组进行一次以上的定义，再参照答案组中的 A 是“重复定义”的出错信息，刚好与本题吻合。由此也确定了正确答案。在实际应用中，如需要对某数组再次定义维数，可先使用 ERASE 语句取消这个数组，然后再定义。本程序可以改为：

```

10 FOR I=2 TO 3
20 DIM A(I)
30 A(I)=I
40 PRINT A(I),
45 ERASE A
50 NEXT I
60 END

```

就不会出现语法错误了。

(2)语句 10 定义了一维数组 A，它的下标上界是 5。20 到 50 语句是计数循环结构，执行 10 次循环。循环体内的 30 和 40 语句引用了数组 A 的元素，当循环五次后，控制变量 I 值为 6；进入第六次循环的 30 语句时，因下标 I>5，产生数组下标越界错。答案为 C。

(3)仔细阅读这段程序，容易看出 20 行有两条 PRINT 语句，按语法要求，同一行号下有多条语句时，各语句之间用冒号“:”分隔；程序中丢掉了冒号，将产生语法错误。答案是 B。

(4)从逻辑上看这段程序是用 READ 语句依次读入 DATA 数据区的四项字符常量，连接后在屏幕上印出。即便不考虑它的逻辑关系，仅从字面上看也很容易发现 20 语句中存在的语法错误。等号左边是字符型变量 A\$，右边的 YES 按 BASIC 的规定，它是一个数值型的变量名，比较运算符两边的数据类型不一致将产生“数据类型不匹配”错误。答案是 D。对 20 行的 YES 加上一对双引号后，程序就能正确运行了。

〔第五题〕程序实现三种主要功能：①随机产生 10

个1000~9999间的整数;②从这10个数中找出最大数和最小数;③取最大数的10位上的数字和最小数个位上的数字重新组成一个两位数。由三种功能将程序划分为三部分:10~60行语句产生10个四位随机整数并存入数组A中,30~60语句是一层计数循环,执行10次40赋值语句,每次把某个数赋值给数组A的一个元素。40行缺少的部分是产生随机数的表达式。产生随机整数的公式为: $INT(RND * (上界 - 下界 + 1)) + 下界$,产生四位随机整数的表达式应为

$INT(RND * (9999 - 1000 + 1)) + 1000$

可简化成 $INT(RND * 9000) + 1000$ 或 $INT(RND * 9000 + 1000)$ 。70~120语句实现从10个四位整数中选出最大和最小数。由120语句可知,结果的最大、最小数存放在A(1)和A(2)中。开始的70语句也是经过比较后,大数存A(1),小数存A(2)。80~110语句是执行8次的计数循环,将遍历A中第三个到第十个元素90和100语句分别涉及到这8个整数A(I)与A(1)或A(2)可能进行的交换。由120、70、90和100语句可以推断,在整个选大小的比较和交换的过程中,A(1)和A(2)始终分别存放最大数和最小数,算法是:当A(I) > A(1)时,A(I)与A(1)交换;否则当A(I) < A(2)时,A(I)与A(2)交换;因此在90行应填入A(1) < A(I);100行填入A(2) > A(I)。130~150语句是将最大数10位上的数字与最小数个位上的数字组成一两位数并印出。由程序的150语句可知,130的A\$存放最大数的10位上的数字,140行的B\$存放最小数的个位数字。我们知道,取数值型数字中的某位的方法之一是先将数转串,再用取子串函数实现。取A(2)的个位数字用右子串函数较方便,140行中填入RIGHT\$(STR\$(A(2)),1)。在PC BASIC中数转串后,符号占最左边一位,因此用取子串函数取A(1)中10位上数字时,应从第四位起取1位。130行填入MID\$(STR\$(A(1)),4,1)。

〔第六题〕由程序说明,这是一段按特定要求对一正数序列实现重新排列的分类问题。以某一正数序列的第一项K1为标准,比K1小的,放在K1的左边;比K1大的放在K1的右边。从实例可以看出,对重新排列后的新序列,K1左边和右边的两个子序列中的数分别小于或大于K1,而并不要求它们自身有序。应首先阅读程序,根据程序提供的结构和已有语句来推断使用的排序方法,在此基础上,补足未完成的程序语句。语句10由INPUT语句通过键盘为变量N提供一个数值,那么变量N在程序中充当何种角色可从已有程序语句中识别。从40到60语句是一循环结构,在N次循环中,通过键盘向数组A供N个数,由此可知,这个序列中正整数个数为N,数组A的下标上界为N。根据数组定义在先,使用在后的性质,30语句应填入DIM A(N)。为了保证序列中的成员为非负整数,在50行空处应填入条件表达式A(I) <= 0 OR A(I) <> INT(A(I))。从10行到70行都是准备阶段,80行开始是本题的难点——排序部分。仔细研究实例,可以发现,在排

序后的序列中,凡是比K1大的数,除了向右移动外,它们原来的顺序没有改变;而小于K1的数全部插在K1的前边,排列次序与原来相反,由此可以推测,在比较之后,如果A(I)小于K1,则将A(I)插入在序列第一项之前。插入的过程是这样的:先将A(I)送入某一临时变量X保存,然后把A(2)到A(I-1)的全部元素向右平移一位,移动的次序为:A(I-1)→A(I),A(I-2)→A(I-1),...,A(1)→A(2);X→A(1)。实现上述动作的程序是110~150语句,100语句对A(I)进行识别,如果A(I) > K1那么对A(I)不做处理,控制转移到外层循环的出口。100行的空应填入A(I) > A(T),A(T)就是K1,变量T在循环开始前值为1,此时K1位于序列的第一项,如果A(I) > A(T),A(I)和A(T)的位置都不改变,当A(I) < A(T)时,要求将A(I)插入到序列第一项之前,在插入之前,子序列A(1),...,A(T),...,A(I-1)向右平移一位(1 ≤ T ≤ I-1),因此在插入操作完成后,指示K1项的下标变量T应增1,160行空处填入T=T+1。120~140语句是一层计数循环,功能是实现平移,130处填入A(J)=A(J-1),2 ≤ J ≤ I。在缺少的语句补全之后,最好使用说明中的实例代入程序中走一遍,以检查结果是否正确。

〔第七题〕首先要掌握流程图中各种图符的含义,在正确理解题意的基础上,将图中各编号处的内容补齐。这是一个分支嵌套结构的程序流程图,其中变量X读入年经济效益数,S累加发奖金总数,N累加合理化建议个数。由流程图可以直接看出变量Y1,Y2,Y3,Y4是分别为奖金2000,1000,500,200设置的累加器。程序首先对变量S,N,Y1~Y4初始化,然后进入当循环结构,循环条件是X≠0,X保存输入的年经济效益数,当X≠0时,应按给定的四个等级标准将X值所对应的等级奖金数累加到Yi中(i=1,2,3,4)。空①所在的判定符下,如果条件为真,则将2000累加进Y1,①处填入条件X≥100;同理②处填入条件100>X≥10;③处填入条件10>X≥1。题中变量X的值,由题意假定取值范围为0和正数。当输入结束时,X值为0,程序在打印S和N值之前应进行计算。由于Y1~Y4中分别存放着四个等级的效益总数,因此④处应填入Y1+Y2+Y3+Y4,⑤处应填入合理化建议总个数Y1/2000+Y2/1000+Y3/500+Y4/200。

答案:

①(2 * SIN(45 * 3.1416/180) + COS(30 * 3.1416/180)) / (3 * ABS(X) * LOG(Z-5))

②Z=X;X=Y;Y=Z ③7

④SAVE "A:PROG.BAS",A ⑤后进先出

二①B ②B G③D ④D ⑤B D

三①C ②D ③C ④D ⑤B

四①A ②C ③B ④D

五INT(RND * 9000 + 1000)

A(1) < A(I)

A(2) > A(I)

MID\$(STR\$(A(1)),4,1)

RIGHT\$(STR\$(A(2)),1)

第三届全国计算机软件人员竞赛(1992)将在京举行

主办单位

由中国软件行业协会考试指导中心、北京市科学技术协会、电子工业出版社、《计算机世界》等单位联合举办

参赛对象

竞赛选手产生自以下两途径:

1. 初级程序员级:按规定时间完成刊登于《电子与电脑》杂志今年第八期上刊出的有关初级程序员级软件水平考试辅导综合自测题,成绩优秀者。

程序员级:参加第四期全国计算机软件水平考试函授辅导班结业考试,成绩优秀者。(以上由中国软件行业协会考试指导中心推荐)

2. 各省市软件水平考试实施办推荐各1-2名。

3. 由台北县电脑同业公会负责推荐。

参赛条件及内容:

(1)初级程序员级 未获得计算机专业本科学历或助理工程师职称者。计算机知识、DOS、中西文录入、BASICA、DBASE III、IBM PC 及兼容机机种。

(2)程序员级 未获得计算机研究生学历或工程师职称者。计算机基础、程序设计(CASL 必答、FORTRAN、PASCAL、C、COBOL 后四种中任选其一)。

(3)年龄均在35周岁以下。

竞赛日期、地点

1992年10月下旬,北京清华大学。

两个级别各设:一等奖1名,二等奖2名,三等奖5名。鼓励奖若干名。将分别发给获奖证书、奖杯及其它纪念品。选手差旅及食宿一律自理,大会代为安排食宿。

联系地址:

1. 北京学院路29号 中国软件行业协会考试指导中心(邮编100083)

电话:2012233-322,传真:2024674

电报:4614,联系人:马素琴

2. 北京清华大学计算机系(邮编100084)

电话:256144-2373,联系人:金慧芬

六 DIM A(N)

A(I)<=0 OR A(I)<>INT(A(I))

A(I)>=A(T)

A(J)=A(J-1)

T=T+1

七①X>=100 ②10<=X<100 ③1<=X<10

④Y1+Y2+Y3+Y4 ⑤Y1/2000+Y2/1000+Y3/500+Y4/200

2)字符串搜索功能:可以在指定外部RAM空间中对指定字符串(字符串最多可由十个字符组成)进行搜索,并在屏幕上显示出指定字符串的存储地址(出现字符串中首字符的地址)。

3)数据块比较功能:可以对指定的二个数据块进行比较,若二个数据块中内容不同,则显示所有不相同单元的地址及内容。

4)EPROM编程功能:包括编程、读出、检查、检验等功能(必须与EPROM编程板配合)。

(上接32页)

2. 状态修改功能——可以对内部RAM、特殊功能寄存器、外部RAM空间、寄存器组按字节进行修改。此外还可以对单片机系统的位控区进行逐位修改。

3. 运行控制功能——可以控制单片机进行单步运行、多步跟踪运行、断点运行、连续全速运行。除全速运行外,其他各种运行方式都能在运行结束后,向用户提供运行后单片机系统的状态信息。

4. 数据传送功能——可以在外部RAM空间中进行SRAM(静态随机存储器)与SRAM与EEPROM(电可改写的只读存储器)之间进行数据传送。可以在外部RAM空间与EPROM空间(由EPROM固化板开辟的存储空间)之间进行数据传送。此外还能在单片机外部RAM空间与磁盘文件之间进行数据传送。

5. 其他功能:

1)反汇编功能:可以对单片机外部RAM空间中的机器码进行反汇编,以列表方式进行显示、存盘。

书 讯

《PC机用户实用维修技法》是一本面向广大微机用户的实用教程。该书的编著者集多年的教学与维修经验,根据用户的需要,重点总结了用户能够自行排除的故障,特别适用于非专业从事维修PC/XT及其兼容机的广大用户,亦可作为维修培训教材。该书且适当兼顾到PC/XT和286微机的维修技术。全书约35万字,现已正式出版,压膜包装,定价7.2元/本(邮购请另加邮费10%)。需购者请直接汇款至安徽蚌埠86181部队计算机教研室涂从润收,信汇或批量购买可与其通信联系(邮编233000)。



BJS—51 单片机实验系统

北京航空航天大学 盛焕鸣

编者按:单片机由于其功能强、价格低廉、抗干扰能力强、易于掌握,稍加一些外围设备就能方便地构成一个应用系统等,近年来在数据采集、工业机器人、仪器仪表等领域中已逐渐取代 Z-80 等八位机成为这些领域智能化的主力。各高等院校也逐步把单片机教学列入教学计划,研制一种专为单片机教学服务的教学实验系统也已经提到日程上。

为了满足这一需要,北京市单片机应用技术协会组织部分高校教师及有关技术人员研制开发了以 MCS—51 系列单片机为核心的 BJS—51 和以 MCS—98 系列单片机为核心的 BJS—98 两套单片机实验系统。本刊从本期起将连载 BJS—51 单片机实验系统的介绍,BJS—98 单片机实验系统将在以后介绍,以飨读者。

一个好的教学实验系统应该由程序编写与调试用的开发手段、模拟实际应用的输入输出手段、指导教学实验的实验大纲、指导书、程序清单等三部份组成。前者为实验系统的硬件部份,后者为实验系统的软件部份。

以往的教学实验大多采用市场上出售的单片机仿真器或带自开发功能的单片单板机作为调试开发手段,用外接面包板来搭接实际应用的输入输出装置。由于面包板接触不可靠,影响了实验的效果。

BJS—51 是一种专为教学实验而设计的单片机系统,在设计一开始就把开发手段、实验器、实验指导书三者紧密地结合起来,尤其着重于把调试手段与模拟输入输出手段的紧密结合。使学生既能方便地完成教学实验,又能从中学到接口设计与编程的有关知识。

BJS—51 教学实验系统采用模块式结构,以仿真器常用的单片机引脚排列方式来定义接口总线,从而使各种模块具有最大的兼容性,它们既能与 BJS—51 教学实验装置构成不同的配置,又能与各种 51 系列仿真器连接。

BJS—51 可以根据不同类型的教学实验建成不同的配置。

一、最小配置——最小配置用于除 A/D、D/A 外的绝大部分大学本科生的教学实验,以及初学者对单片机的学习。最小配置为一块 BJS—51 系统板。BJS—51 的结构见图 1。

BJS—51 系统板由二部份组成。右边为带有自开发功能的单片机扩展系统,左边为实验器。单片机扩展系统的原理电路图见图 2。

图中存储器系统配有四个 28 脚插座。它们的地址空间分别为 0000H~1FFFFH (U_3 及 U_4)、4000H~5FFFFH (U_5),及 8000H~FFFFH (U_6)。其中 U_3 由 \overline{PSEN} 控制为 8031 的 ROM 空间, U_4 由 \overline{RD} 控制为 8031 的 RAM 空间。把 0000~1FFFFH 分为二个空间是为了使该系统既能进行 51 系列单片机的汇编语言教学实验,又能在配上 BASIC—52 解释程序后进行 BASIC—52 教学实验。

U_6 可以通过边上的短路块配接为不同的容量。当将短路块插入左边的插针,在 U_6 座上配上 6264 则为 8KB 空间,若将短路块插入右边插针,在 U_6 座上配上 62256 则为 32KB 空间。

除 U_3 及 U_4 外, U_5 及 U_6 均为 8031 的混合空间,它们都既能作为用户的程序区,也能作为用户的数据区。

U_5 为 8155,其地址空间为 6000H~67FFH。

8155 的内部 RAM 用以存放监控程序使用的各种标志、数据以及保存用户程序的运行现场。8155 的口线(PA 口, PB 口, PC 口)连同 8031 的部份口线通过 40 线扁平电缆座 CZ2 引出。CZ2 可以与键盘显示板连接,以使用键盘 LED 方式进行调试。

另一插座 CZ3 是 BJS—51 的系统扩展插座,BJS—51 系统的所有扩展板均与 CZ3 相接。CZ3 的引脚排列基本上与 8031 的引脚排列的次序相同,仅用二条片选线(7400H~77FFH, 7800H~7BFFH)代替 8031 的时钟线,以提高信号的利用率。BJS—51 采用这种总线方式是为了最大限度地与其他仿真器兼容,因为大多数仿真器及自开发型单片单板机均带有这种总线插座,因此 BJS—51 的所有扩展板均能连接在这些装置上运行。

8031 的 RXD 与 TXD 通过晶体管 9012、9014 及其附加电路组成简易 RS232 接口。它可与 PC 机的标准 RS232C 连接,在通信程序支持下进行系统的编程与调试。由于 PC 机能够直接对单片机程序进行汇编以及对数据进行存盘等功能,随着 PC 机价格的下降,在 PC 机支持下开发单片机系统已成为单片机开发的主要形式,也是在最小配置下 BJS—51 的程序调试方式。

系统板的左部为实验器,它们包括:

1. 二个按键开关。用来模拟各种开关量的输入装置。因为开关量都是 0 与 1 二个状态,因而这二个按键开关可以模拟实际应用中的各种开关,诸如机械式开关、光电开关、自动线上计数器、行程开关、编码器等。

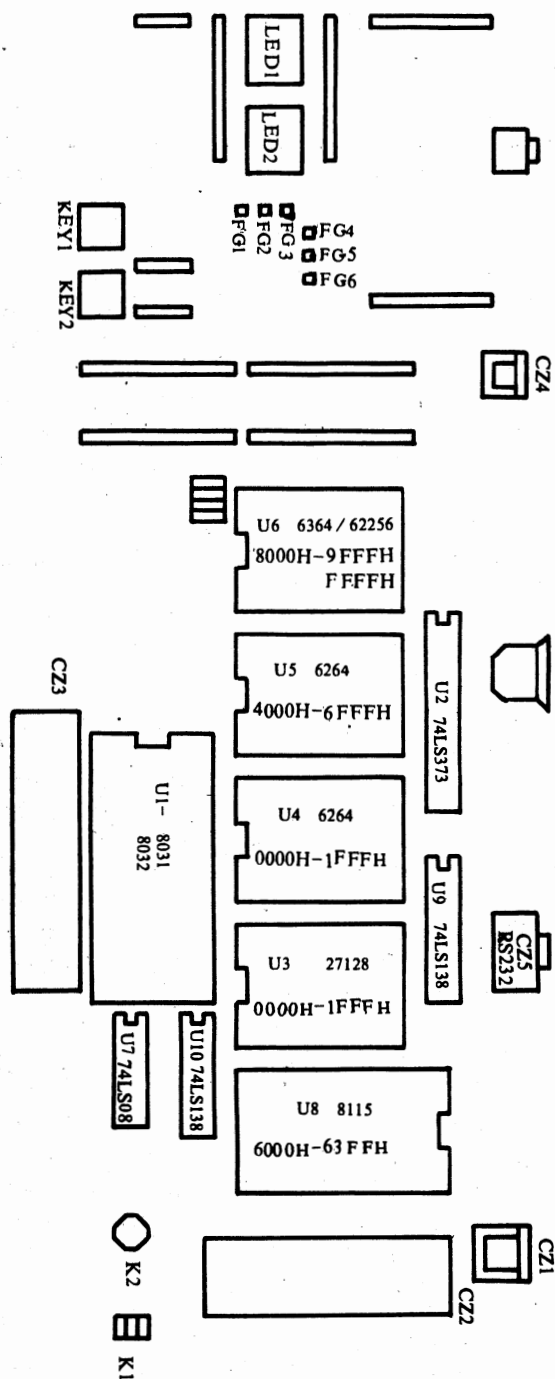


图1 BJS-51 系统板结构

2. 二个 LED 数码管。用来显示、输出二位数字,作为数码输出装置。诸如显示时钟、计数值以及进行静态或动态扫描方式的数字、字符输出实验。

3. 六个发光二极管。发光二极管为输出装置,可以模拟各种类型开关量的输出,例如开关的接通(亮)或断开(灭),数字输出的 0 与 1,阀的开与关等。此外六个发光二极管排列成二组“红绿灯”可以进行传统的交通灯实验。

4. 蜂鸣器。蜂鸣器也是输出装置,可以用于各种发声实验,输出音乐及警报等。

5. 二组晶体管驱动电路。用以对单片机的输出信号进行功率放大、驱动各种输出装置。

此外系统板上还配有二芯、三芯插座,以便从外部接入其他实验器(例如小电机……等)

所有实验器的引脚以及 8031 的引脚都被引到插针上,在实验时通过绕接器把需要连接的二点用导线接上便可进行实验。BJS-1 通过绕接方式进行实验时的接线,与使用面包板相比,不仅缩小了实验装置的体积,降低了成本,而且大大提高了接线的可靠性,从而提高了实验的质量。

系统板上配置了实验器,大大简化了教师的实验准备工作。

BJS-51 系统设计时,为了突出单片机的特点即“一片集成电路就是一台计算机”,所以在系统板及实验设计上都强调尽量使用单片机本身的资源来完成各种实验。

使用最小配置,可以熟悉在 PC 机支持下 51 系列单片机的调试方法,进行 51 系列单片机的程序编写汇编练习,以及各种简单的输入输出装置的驱动程序练习,直至进行复杂的交通灯试验。

二、基本配置——基本配置是在最小配置(一块系统板)的基础上再加接一块 LED 显示键盘板及 EPROM 写入读板。

LED 显示键盘板为带有 32 键及 6 个 LED 显示器的扩展板,同时还扩有打印接口。使用时,显示键盘板接在系统板的 CZ2 插座上。其电原理图见图 3。LED 显示器上八条段控线由系统板上 8155 的 PB 口通过驱动电路(QD₂)提供。8155 的 PA 口通过驱动电路(QD₁)来提供 LED 显示器的位控信号,同时也驱动键盘。8155 的 PC₀~PC₃ 用于拾取按键状态。

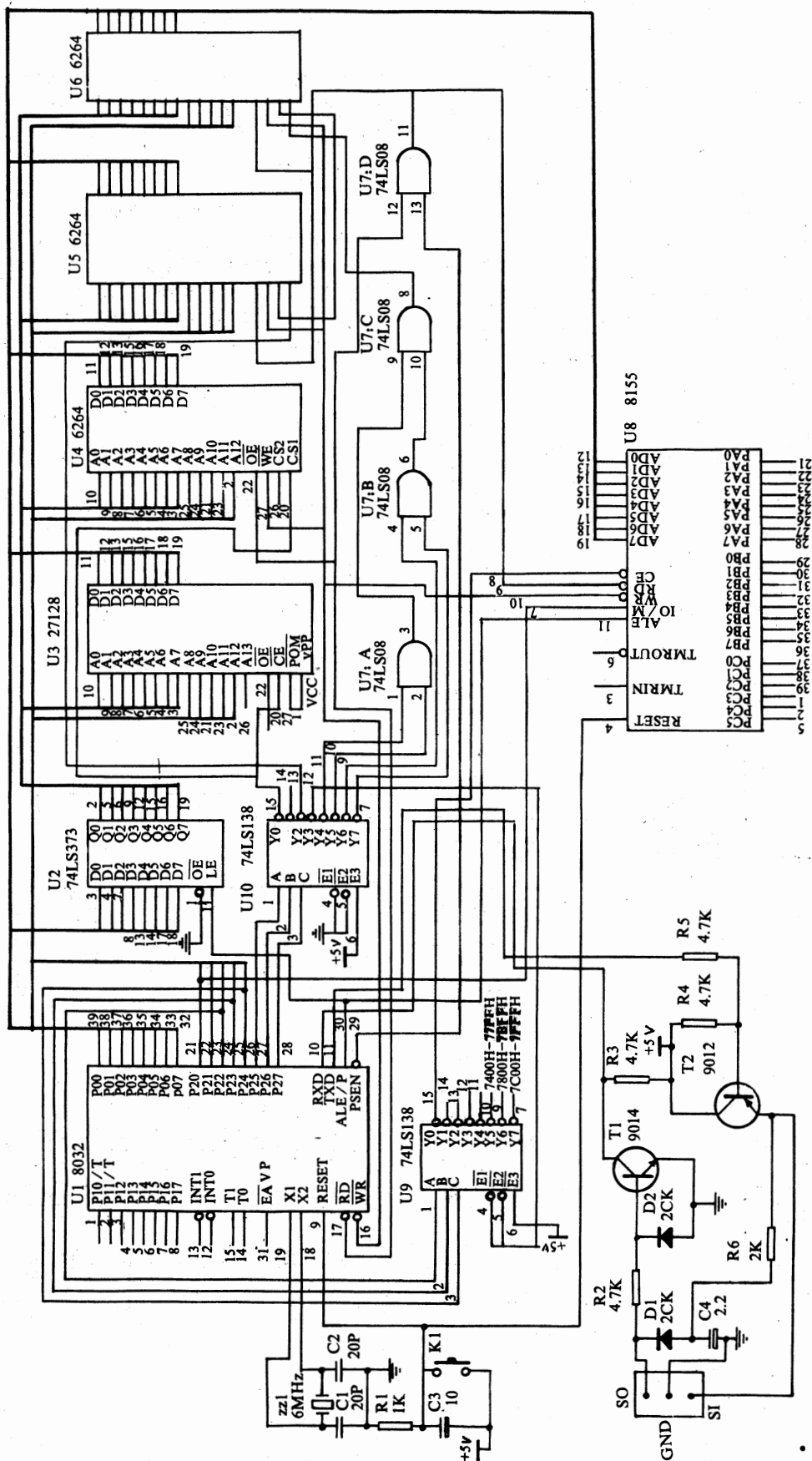


图 2 单片机扩展系统电路图

打印接口用 $PB_0 \sim PB_7$ 作为数据线 $D_0 \sim D_7$ 。 $P1.0$ 为打印机的启动信号 STB , PC_5 用来读入打印机的 $BUSY$ 信号。

EPROM 读写板在监控程序支持下对 2716、2732、2764、27128、27256、27512 各种型号 EPROM 进行写入、读出、比较、检查。

EPROM 操作时所有地址、数据信号均由 8031 数据总线通过 8255 锁存后提供,而操作 EPROM 所有的控制线由 8031 $P0$ 口通过 74LS377 锁存后提供。由于 EPROM 采用 5.4V 供电以及配合监控程序中的智能编程方法,在正常情况下写入一片 2764 约为 20 秒左右,大大提高了 EPROM 写入速度。

在基本配置下,可以脱离 PC 机,在 LED 显示器及键盘支持下进行各项实验,此外还可以进行电子钟、显示器的动态扫描、键盘的扫描、译码、打印机驱动程序、EPROM 操作等实验。

使用显示键盘板进行调试的监控程序已固化在系统板上,其功能类似于 BJ-51 仿真器。

三、扩展配置

扩展配置是一种无穷尽的配置,在 BJS-51 系统设计开始时就确定为一种开放式的结构。即所有资源都向用户开放,以便用户也参与到 BJS-51 系统开发中来。

扩展板由硬件与软件二部分组成。硬件部分为以

某种接口芯片(例如 12 位 A/D 芯片 AD574)为核心,配以完善的单片机接口电路组成的电路板。该扩展板与系统板连接后可以完成该接口芯片的各种实验。软件部分包括该接口芯片的有关资料例如引脚排列、使用方法、应用电路、驱动程序等。用户可以通过对扩展板的实验,学会一种新的芯片的使用方法。

目前已设计有各种 A/D 芯片、显示键盘芯片、电机控制系统等扩展板。随着大量专用芯片(语言芯片、实时钟芯片等)的出现,扩展板的设计将具有广阔的前景。同时,教育实验系统也不仅用于大学本科生的单片机学习,还将成为广大工程技术人员进行学习研究的好帮手。

BJS-51 的软件包括监控程序 BJS-51-MON 及实验程序库,前者固化在一片 27128 上用于进行系统的编程与调试;后者存放在软磁盘上以便今后不断地丰富扩充。

BJS-51-MON 由二个独立的监控程序组成,它们各占 8KB。前 8KB 用于在 PC 机支持下进行编程与调试,后 8KB 用于在 LED 显示器及键盘支持下进行编程及调试。它们具有下列功能:

1. 状态显示功能——可以显示单片机内部 128B (或 256B)字节 RAM 的内容,以及所有特殊功能寄存器及二个 64K 的外部存储器空间的内容。

(下转 28 页)

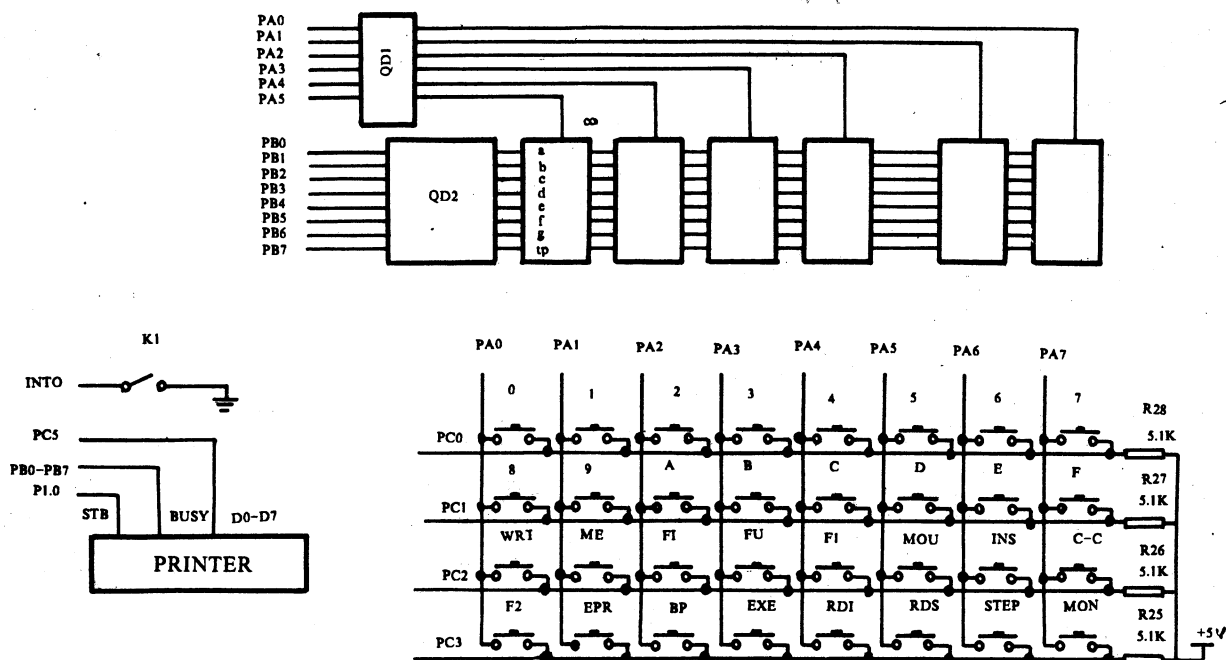
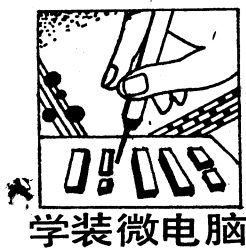


图 3 显示键盘板原理电路图



步进电机的控制

易齐干

步进电机有输入脉冲与电机轴转角成比例的特征。微电脑控制步进电机最适宜。

本实验使用日本 PXB43—01A 型步进电机，步距角为 1.8°。规格如表 1 所示。

将外部负载转矩施加于电机轴上，对电机以额定电压激磁，电机轴开始启动时的扭矩称为激磁最大静转矩。

电机的外形尺寸与内部原理如图 1 所示。

表 1 PXB43—01A 规格

电压	电流	激磁最大静止转之后	线圈电阻
DC 12V	0.16A / 相	500g·cm	770Ω / 相

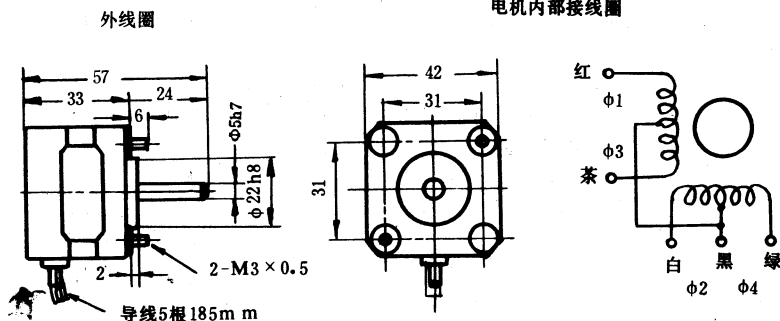


图 1 PXB43—01A 外形尺寸与内部原理

表 2.2 相激磁通电次序

顺 序	相	φ1	φ2	φ3	φ4	反时钟方向回转
0		1	1	0	0	
1		0	1	1	0	
2		0	0	1	1	
3		1	0	0	1	

1: 附加 DC12V

0: 电波

电机线圈由 4 相组成。即：φ1(红—黑)、φ2(白—黑)、φ3(棕—黑)、φ4(绿—黑)。驱动方式为 2 相激磁方式。各线圈通电顺序如表 2。

表中首先向 φ1 线圈—φ2 线圈输入 DC12V 电压，接着 φ2—φ3、φ3—φ4、φ4—φ1，又返回到 φ1—φ2，按这种顺序切换，电机轴沿顺时针方向旋转。

步进电机的驱动电路如图 2 所示，对照表 2，微电脑向步进电机输入端传送 1 或 0 信息，则可实现上述动作。

为简化驱动电路，考虑步进电机线圈的电流为 0.16A，可选用三菱电机的 M54532P 驱动器 IC，再配备整流二极管。或者选用晶体管搭接成电路也可组成驱动电路，这种事例读者很容易查阅到。

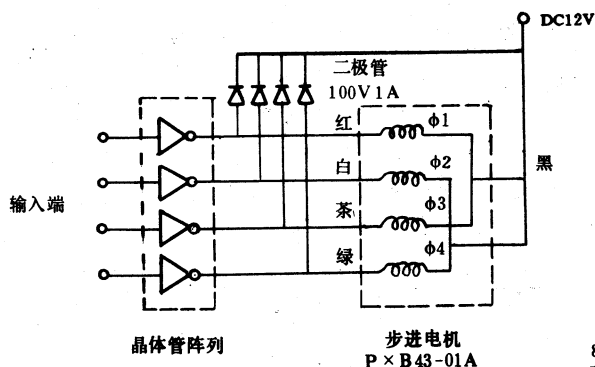


图 2 步进电机驱动电路

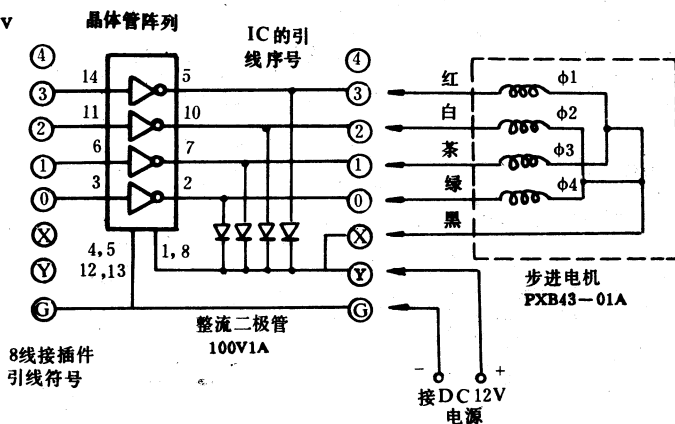


图 3 步进电机驱动电路

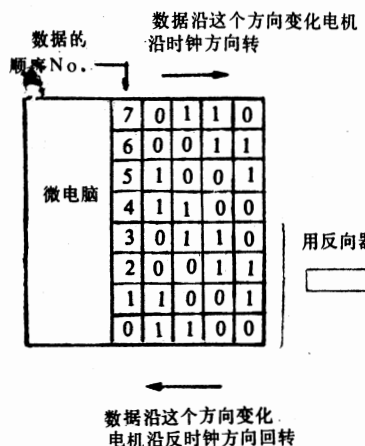


图 7 微电脑输出数据与电机附加数据

数据的顺序No.

3	1	0	0	1	... 向电机的φ1附加
2	1	1	0	0	... 向电机的φ2附加
1	0	1	1	0	... 向电机的φ3附加
0	0	0	1	1	... 向电机的φ4附加

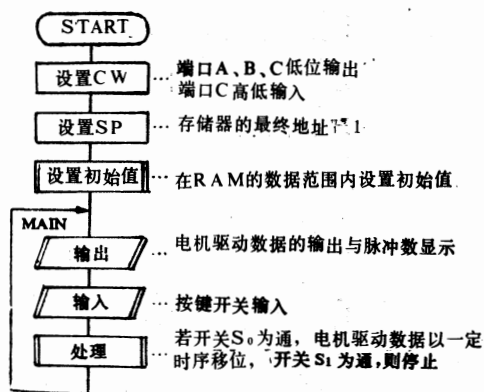


图 8 简单流程图

00F0H	00F1H	00F2H	00F3H	00F4H
开关输入	开关标志	驱动数据	脉冲计数器	
...	...	3 3	0 6	0 0
开关S ₀ 为通	0 → 1	低位	下位	
开关S ₁ 为通	0 → 0			

图 9 RAM 区域

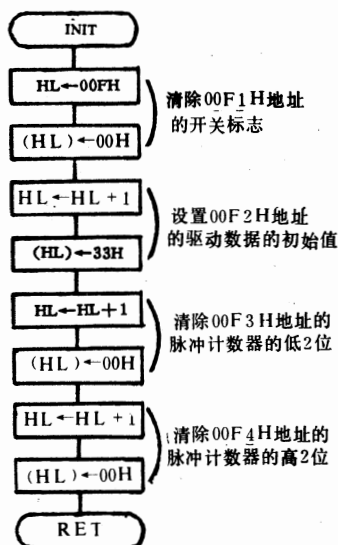


图 10 设置初始数据子程序

输出(OUTPUT)子程序如图 11 所示。向端口 C 低位输出 00F2H 地址的驱动数据; 向端口 B、端口 A 输出 00F3H 和 00F4H 地址的脉冲计数器数值。

输入(INPUT)子程序如图 12 所示。按键开关通过端口 C 高位将开关状态放入 00F0H 地址。

处理(PROCES)子程序如图 13 所示。它由三个子程序组成。类似这种较大的处理内容再划分为多个小的处理内容, 编程过程中思路清楚, 减少失误, 程序准确。

第一个处理开关标志(SWFLAG)子程序如图 14 所示。程序中相对于按键开关 S₀、S₁ 为 ON 状态, 00F1H 地址第 0 位为 1 或为 0。查询 00F1H 地址第 0 位, 就可以判断是否按动过按键开关 S₀。

第二个处理驱动数据(DRDATA)子程序如图 15 所示。首先查询按键开关 S₀, 如果按动过 S₀, 以 TIMER 延时程序获得的时间处理驱动数据。

第三个处理脉冲计数器(COUNT)子程序如图 16 所示。COUNT1 子程序是 10 进制计数加程序。

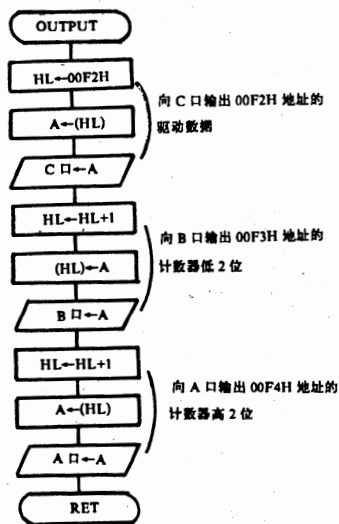


图 11 输出子程序

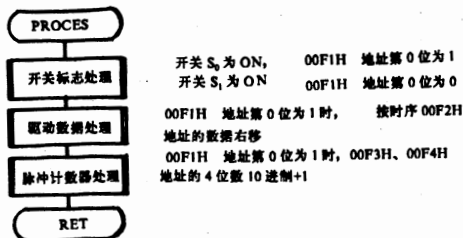


图 13 处理子程序

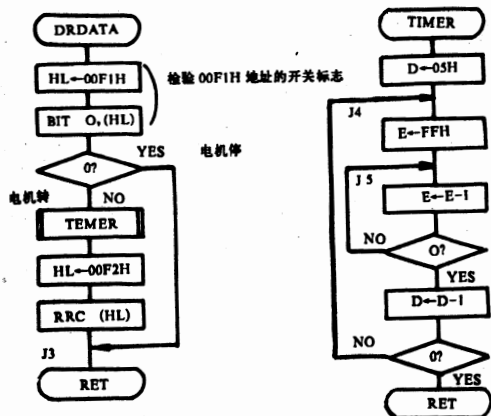


图 15 驱动数据处理子程序

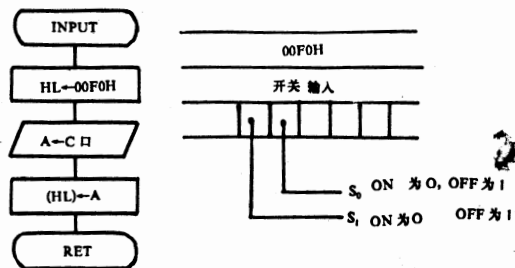


图 12 输入子程序

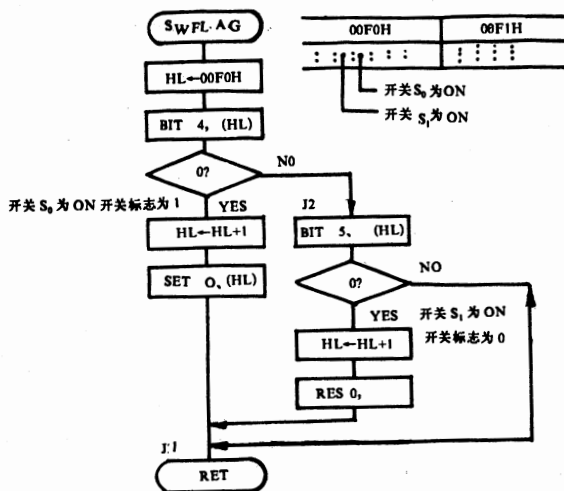


图 14 开关标志处理子程序

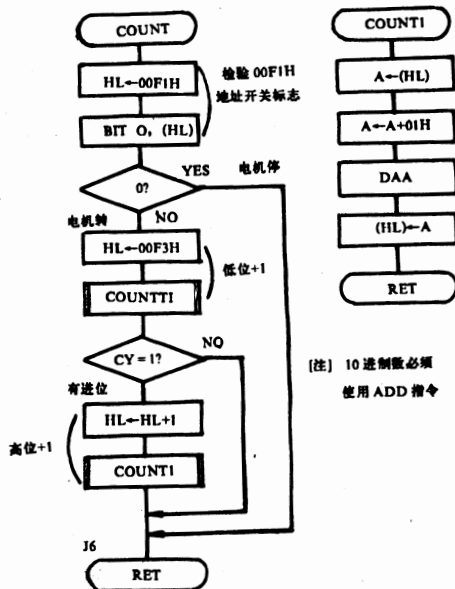


图 16 脉冲计数子程序

表 3 程序清单(对照流程图编制出程序清单,如表 3 所示。)

标记	助记符	地址	机械语
START	LD A88H	0000	3E 88
	OUT (03H)A	0002	D3 03
	LD SP,0100H	0004	31 00 01
	CALL INIT	0007	CD 20 00
MAIN	CALL OUTPUT	000A	CD 30 00
	CALL INPUT	000D	CD 40 00
	CALL PROCES	0010	CD 50 00
	JP MAIN	0013	C3 0A 00
INIT	LD HL,00F1H	0020	21 F1 00
	LD HL,00H	0023	36 00
	INC HL	0025	23
	LD HL,33H	0026	36 33
	INC HL	0028	23
	LD HL,00H	0029	36 00
	INC HL	002B	23
	LD HL,00H	002C	36 00
	RET	002E	C9
OUTPUT	LD HL,00F2H	0030	21 F2 00
	LD A,HL	0033	7E
	OUT (02H)A	0034	D3 02
	INC HL	0036	23
	LD A,(HL)	0037	7E
	OUT (01H),A	0038	D3 01
	INC HL	003A	23
	LD A,(HL)	003B	7E
	OUT (00H),A	003C	D3 00
	RET	003E	C9
INPUT	LD HL,00F0H	0040	21 F0 00
	IN A,(02H)	0043	DB 02
	LD (HL),A	0045	77
	RET	0046	C9
PROCES	CALL SWFLAC	0050	CD 60 00
	CALL DRDATA	0053	CD 80 00
	CALL COUNT	0056	CD A0 00
	RET	0059	C9
SWFLAG	LD HL,00F0H	0060	21 F0 00
	BIT 4,(HL)	0063	CB 66

	JP NZ,12	0065	C2 6C 00
	INC HL	0068	23
	SET 0,(HL)	0069	CB C6
	RET	006B	C9
J2	BIT 5,(HL)	006C	CB 6E
	JP NZ,J1	006E	C2 6B 00
	INC HL	0071	23
	RES 0,(HL)	0072	CB 86
	JP J1	0074	C3 6B 00
DRDATA	LD HL,00F1H	0080	21 F1 00
	BIT 0,(HL)	0083	CB 46
	JP Z,J3	0085	CA 90 00
	CALL TIMER	0088	CD 91 00
	LD HL,00F2H	008B	21 F2 00
	RRC (HL)	008E	CB 0E
J3	RET	0090	C9
TIMER	LD D,05H	0091	16 05
J4	LD E,FFH	0093	1E FF
J5	DEC E	0095	1D
	JP NZ,J5	0096	C2 95 00
	DEC D	0099	15
	JP NZ,J4	009A	C2 93 00
	RET	009D	C9
COUNT	LD HL,00F1H	00A0	21 F1 00
	BIT 0,(HL)	00A3	CB 46
	JP Z,J6	00A5	CA B5 00
	LD HL,00F3H	00A8	21 F3 00
	CALL COUNTI	00AB	CD B6 00
	JP NC,J6	00AE	D2 B5 00
	INC HL	00B1	23
	CALL COUNTI	00B2	CD B6 00
J6	RET	00B5	C9
COOUNT1	LD A,(HL)	00B6	7E
	ADD A,01H	00B7	C6 01
	DAA	00B9	27
	LD (HL),A	00BA	77
	RET	00BB	C9

4位数据显

输入输出

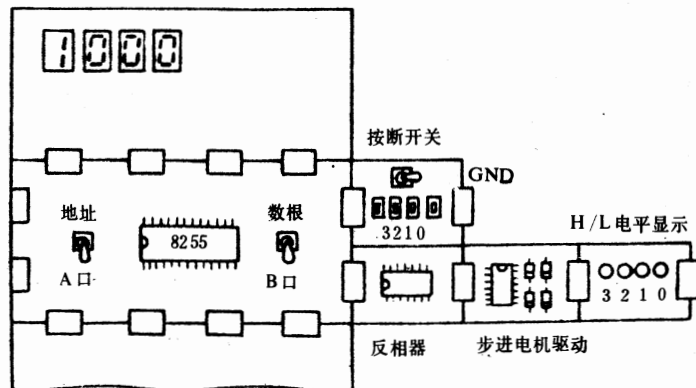


图 17 检验程序的硬件

μP-80 套件执行程序前,应该用 H/L 电平显示部件代替步进电机。避免因写入错误而损坏步进电机。执行程序时,输入输出部件的切换开关搬至 A 口、B 口侧,4 位数显部件应该显示 0000。硬件连接图如图 17 所示。按动按键开关 S₀,H/L 电平显示部件的 4 支 LED 应该反复闪亮,说明程序动作正确。

确认动作正确之后,将驱动部件、步进电机和 DC12V 电源代替 H/L 电平显示部件,重新执行程序。按键开关 S₀ 为 ON,步进电机沿时钟方向运转,4 位数显部件显示计数加。如果按动按键开关 S₁,则步进电机停止旋转。



ASCII 码字符显示器的设计

贵州风华电冰箱厂 李德文

数字显示器(又称为数码管)常用的有 7 段数字显示器和 8 段数字显示器两种。由于使用 7 段笔划便能拼成十个数字和少量英文字母,在电子手表、时钟、计算器及数字仪表中广为采用。

能否设计出一种由若干段笔划组成的字符显示器,用它可以显示 10 个数字和 26 个英文字母,乃至显示所有的 ASCII 码字符呢?

回答是肯定的。

为了演示这一结果,本文中附上了一个在 AST 上 GWBASIC 和 BASICA 下运行通过的演示程序。

标准 ASCII 字符集共有 128 个字符,其中 94 个为可显示字符。通过分析和抽象,笔者发现,使用图 1 所示的 20 段笔划,便能组成除“#”之外的 93 个 ASCII 码字符。

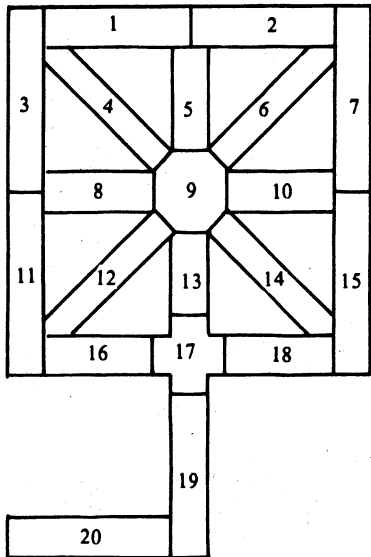


图 1 20 段字符显示器

如果把此 20 段笔划做成发光二极管封装于透明的环氧树脂中,如图 2 所示,便可以形成 20 段字符显示器。

然而,为了控制 20 段发光二极管的发光,至少需要 20 条引脚,这对生产厂家和用户都不一定方便。解决此问题,可以如图 3 所示,设计一个“ASCII 码—20 段字符显示码转换电路”接于字符显示器之前,并与它集成在一起。这样,控制 20 段发光二极管的发光,只需有 7 条引脚就足够了。

至于转换电路的设计,稍懂数字电路的人都容易实现。

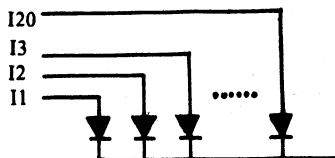


图 2 发光二极管 20 段字符显示器

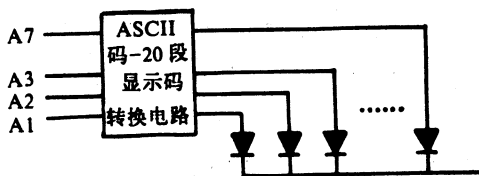


图 3 带转换电路的发光二极管 20 段字符显示器

根据图 1 中的编号,当为了显示一字符而点亮某段时,令此段编码为 1,否则,编码为 0。这样,每个字符的编码共有 20 位。按四位一组分开成五组,每组的四位二进制数组成一位十六进制数,按编号从小到大的次序排列,得到五位十六进制数,作为 20 段字符显示器显示编码。

表 1 为所有 ASCII 码可显示字符与编码对照表。其中,无法显示字符“#”的编码令为 FFFFF。

表 1 字符 ASCII 码 20 段数码显示器编码对照表

字 符 ASCII 编 码	!	"	#	\$	%	&	,	()	*	+	,
	033 08808	034 0A000	035 FFFFF	036 E9CBE	037 35D60	038 988E4	039 04000	040 04840	041 10900	042 15D40	043 09C80	044 0000B
字 符 ASCII 编 码	-	.	/	0	1	2	3	4	5	6	7	8
	045 01C00	046 00800	047 04900	048 E2A3C	049 08888	050 C2D1C	051 C243C	052 29C80	053 48C3C	054 E1E3C	055 C4880	056 D495C
字 符 ASCII 编 码	9	:	;	<	=	>	?	@	A	B	C	D
	057 E3C3C	058 00808	059 0080B	060 0491C	061 01C1C	062 1085C	063 C2C88	064 C3ABC	065 06D20	066 E5A5C	067 E021C	068 CA8BC
字 符 ASCII 编 码	E	F	G	H	I	J	K	L	M	N	O	P
	069 E121C	070 E1A00	071 E063C	072 23E20	073 C889C	074 C8890	075 0C8C0	076 2021C	077 36A20	078 32A60	079 E223C	080 E3E00
字 符 ASCII 编 码	Q	R	S	T	U	V	W	X	Y	Z	[\
	081 E227C	082 E3E40	083 E1C3C	084 C8880	085 2223C	086 24B00	087 22B60	088 14940	089 14880	090 C491C	091 48884	092 10840
字 符 ASCII 编 码]	^	_	`	a	b	c	d	e	f	g	h
	093 88890	094 00940	095 0001C	096 10000	097 01A9C	098 21298	099 01210	100 09A98	101 01310	102 49C8B	103 01A9B	104 21288
字 符 ASCII 编 码	i	j	k	l	m	n	o	p	q	r	s	t
	105 00888	106 00898	107 08CC0	108 0888C	109 01EA0	110 01A88	111 01290	112 00CAE	113 01A9A	114 00C80	115 00444	116 09C8C
字 符 ASCII 编 码	u	v	w	x	y	z	{		}	~		
	117 00A98	118 00B00	119 00B60	120 01984	121 0028B	122 01918	123 49884	124 08080	125 88C90	126 30860		

当然,表 1 给出的编码仅仅是参考码而已,如果投入生产,生产厂家完全有理由从美观、工艺等角度对之进行修改。

以下是 20 段字符显示器显示 ASCII 码字符的演示程序:

LIST

10 SCREEN 2,1;CLS;GOTO 420

20 '子程序 1(画框架)

30 LINE(110,10)-(115,110),,B;LINE(110,60)-(115,60)

40 LINE(115,10)-(195,15),,B;LINE(155,10)-(155,15)

50 LINE(195,10)-(200,110),,B;LINE(195,60)-(200,60)

60 LINE(115,18)-(150,57);LINE-(152,55);LINE-(118,15)

70 LINE(192,15)-(157,55);LINE-(160,57);LINE-(195,18)

80 LINE(115,57)-(150,63),,B;LINE(160,57)-(195,63),,B

90 LINE(152,15)-(157,55),,B;LINE(152,65)-(157,103),,B

100 LINE(152,113)-(157,160),,B;LINE(110,155)-(152,160),,B

110 LINE(115,102)-(150,63);LINE-(152,65);LINE-(118,105)

120 LINE(192,105)-(157,65);LINE-(160,63);LINE-(195,102)

130 LINE(115,105)-(150,110),,B;LINE(160,105)-

(195,110),,B

140 LINE(150,105)-(152,105);LINE-(152,103)

150 LINE(157,103)-(157,105);LINE-(160,105)

160 LINE(160,110)-(157,110);LINE-(157,113)

170 LINE(150,110)-(152,110);LINE-(152,113)

180 RETURN

190 '子程序 2(点亮)

200 ON I GOTO 210,220,230,240,250,260,270,280,290,300,310,320,330,340,350,360,370,380,390,400

210 PAINT(120,13),,2;GOTO 410

220 PAINT(180,13),,2;GOTO 410

230 PAINT(113,20),,2;GOTO 410

240 PAINT(117,17),,2;GOTO 410

250 PAINT(155,20),,2;GOTO 410

260 PAINT(193,17),,2;GOTO 410

270 PAINT(196,20),,2;GOTO 410

280 PAINT(120,60),,2;GOTO 410

290 PAINT(155,60),,2;GOTO 410

300 PAINT(180,60),,2;GOTO 410

310 PAINT(113,80),,2;GOTO 410

320 PAINT(117,104),,2;GOTO 410

330 PAINT(155,80),,2;GOTO 410

340 PAINT(193,104),,2;GOTO 410

350 PAINT(196,80),,2;GOTO 410

360 PAINT(120,108),,2;GOTO 410

370 PAINT(155,108),,2;GOTO 410

380 PAINT(180,108),,2;GOTO 410

390 PAINT(155,120),,2;GOTO 410

400 PAINT(120,158),,2;GOTO 410

410 RETURN

```

420 '主程序(显示字符)
430 LOCATE 3,50:INPUT"请输入一个 ASCII 码字符:";
    CH$;CLS
440 IF LEN(CH$)<>>1 THEN END
450 LOCATE 5,50:PRINT"字符:";CH$
460 LOCATE 7,50:PRINT"ASCII 码:";ASC(CH$)
470 READ CH1$:IF CH1$<>>CH$ GOTO 470
480 GOSUB 20
490 READ CH2$:CH5$=""
500 FOR I=1 TO 5
510 CH3$=MID$(CH2$,I,1)
520 IF CH3$="0" THEN CH4$="0000":GOTO 690
530 IF CH3$="A" THEN CH4$="1010":GOTO 690
540 IF CH3$="B" THEN CH4$="1011":GOTO 690
550 IF CH3$="C" THEN CH4$="1100":GOTO 690
560 IF CH3$="D" THEN CH4$="1101":GOTO 690
570 IF CH3$="E" THEN CH4$="1110":GOTO 690
580 IF CH3$="F" THEN CH4$="1111":GOTO 690
590 ON VAL(CH3$) GOTO 600,610,620,630,640,
    650,660,670,680
600 CH4$="0001":GOTO 690
610 CH4$="0010":GOTO 690
620 CH4$="0011":GOTO 690
630 CH4$="0100":GOTO 690
640 CH4$="0101":GOTO 690
650 CH4$="0110":GOTO 690
660 CH4$="0111":GOTO 690
670 CH4$="1000":GOTO 690
680 CH4$="1001"
690 CH5$=CH5$+CH4$
700 NEXT I
710 FOR J=1 TO 20
720 I$=MID$(CH5$,J,1)
730 IF I$="1" THEN I=J:GOSUB 190
740 NEXT J
750 RESTORE:GOTO 430
760 '主程序(编码)
770 DATA !, 08808, " ", 0A000, #, FFFFF, $,
E9CBE, %, 35D60, &, 988E4, " ", 04000, (, 04840
780 DATA ), 10900, *, 15D40, +, 09C80, ", ", 0000B,
-, 01C00, ., 00800, /, 04900, 0, E2A3C
790 DATA 1, 08888, 2, C2D1C, 3, C243C, 4, 29C80, 5,
48C3C, 6, E1E3C, 7, C4880, 8, D495C
800 DATA 9, E3C3C, ":", 00808, ;, 0080B, <, 0491C,
=, 01C1C, >, 1085C, "?", C2C88
810 DATA @, C3ABC, A, 06D20, B, E5A5C, C, E021C, D,
CA8BC, E, E121C, F, E1A00, G, E063C
820 DATA H, 23E20, I, C889C, J, C8890, K, 0C8C0, L,
2021C, M, 36A20, N, 32A60, O, E223C
830 DATA P, E3E00, Q, E227C, R, E3E40, S, E1C3C, T,
C8880, U, 2223C, V, 24B00, W, 22B60
840 DATA X, 14940, Y, 14880, Z, C491C, [, 48884, \,
10840, ], 88890, ^, 00940, _ , 0001C
850 DATA ', 10000, a, 01A9C, b, 21298, c, 01210, d,
09A98, e, 01310, f, 49C8B, g, 01A9B

```

```

860 DATA h, 21288, i, 00888, j, 00898, k, 08CC0, l,
0888C, m, 01EA0, n, 01A88, o, 01290
870 DATA p, 00CAE, q, 01A9A, r, 00C80, s, 00444, t,
09C8C, u, 00A98, v, 00B00, w, 00B60
880 DATA x, 01984, y, 0029B, z, 01918, {, 49884, |,
08080, }, 88C90, ~, 30860

```

20~180 号语句为第一个子程序,用于画图 1 所示的图案。

190~410 号语句为第二子程序,用于根据编码使相应的笔划变亮。

420~750 号语句为主程序的控制部分,用于提示用户输入待显示的字符,并调用子程序一和子程序二演示 20 段字符显示器的显示结果。

760~880 号语句为主程序的数据部分,根据表 1 给出所有 ASCII 码可显示字符及其在 20 段字符显示器上的编码。

本程序在运行时,请注意:

(1)由于 BASIC 语言的原因,“”字符不能显示,但读者不难根据编码想象出其形状;“,”为 BASIC 分隔符,在欲输入“,”时需输入“,”。

(2)若不输或输入两个以上字符后回车,程序便告退出。

根据表 2 所示程序的演示结果得知:编号为 19 和 20 的两段笔划,只有在显示,;,f,g,p,q,y 等字符时才会用到;编号为 9 和 17 的笔划单独使用,也只有用来显示!,:;|等字符。如果对这些字符的兴趣不大,例如,只关心大写英文字母等。可以把 20 段字符显示器简化成图 4 所示的 16 段字符显示器。读者可以参照本文前面的内容给出 16 段字符显示器上 ASCII 码可显示字符的编码,并编写相应的演示程序。

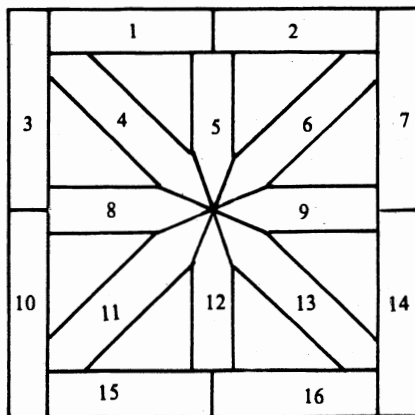


图 4 16 段字符显示器

诚然,有的字符显示不够美观,但分辨起来还是较为容易的。

使用一排足够长的 20 段字符显示器,可以组成任意的英文句子,而且还能随时变换,在车站、码头、街头广告牌等处有着广泛的应用前景。在工业控制专用机显示屏等电子显示设备上,也能发挥其特殊效用。



电脑游戏机

第四章 F BASIC 语言的深入理解

山东苍山县机械电子化学工业局(277700) 于 春

六、音乐控制语句

科特 FCS-90 电脑学习机可以在 BS 状态通过人机对话进入 MUSIC BOARD(谱曲与演奏)状态,进行谱曲练习和演奏。由于 MUSIC BOARD 操作简单、输入方便、音色优美,深受广大用户的喜爱。但是,谱曲演奏只有一页的版面供使用。而且这一页仅有四行,一次最多可演奏 96 个音符(重奏除外);只有一种节拍,音长的允许变化范围也太窄(一般两个音长,要使用三个或四个音长,那么输入的音符要减少 3~4 倍)。这对于气势较大,有节拍变化的乐曲简直不能演奏。更有甚者每次只能演奏一小段乐曲,一首乐曲要断续几次甚至十几次才能演奏完毕,因而破坏了乐曲的完整性,影响了欣赏效果。可以说,MUSIC BOARD 仅是儿童游戏。欲真正地欣赏电脑演奏乐曲,还必须使用音乐控制语句,编制音乐演奏程序才能实现。

1. 发音语句(BEEP)

BEEP 简写 B.

BEEP 语句的功能是使扬声器发出 1000Hz 的蜂鸣声。

例 21

```
10 F.I=0 TO 10
20 BEEP;PAU. 20
30 N.
```

RUN

扬声器发出断续的蜂鸣声。

2. 演奏语句 (PLAY)

PLAY 简写 PL.

PLAY 语句的功能是谱曲和演奏乐曲。语句格式为

PL.“弦乐资料”

式中的弦乐资料是指按 F BASIC 的约定,编制的乐曲演奏字符串。

随机手册中对 PLAY 语句中的弦乐资料介绍的较笼统,读者虽能仿效演奏几句乐谱,但很难设计出一个完整的演奏程序。只有了解弦乐资料的语法结构,掌握编程的方法步骤,明确各项编程的约定后,才能编制出合法的程序,得心应手地谱写乐曲。

〈一〉弦乐资料的约定

A. 节拍:

节拍以 T1~T8 指定

T1(快)↔T8(慢)

B. 音色:

音色以 Y0~Y3 指定

Y0: 12.5%

Y1: 25%

Y2: 50%

Y3: 75%

C. 主音:

主音由 M0~M1 指定。

M0 表示音量。以 V0~V15 指定音量的大小。V0(最小)——V15(最大)。

M1 表示音长。以 V0~V15 指定发音的长短。V0(最短)——V15(最长)。

D. 8 度音

8 度音以字母 O 指定,O0~O5。

以简谱中 1 音为例,对照如下:

O0 O1 O2 O3 O4 O5

1 1 1 1 . :

若乐曲中有 1 音,可将 8 度音升一级 O5 为 1,其余类推。

E. 音名和音符

音名分别用 C、D、E、F、G、A、B 表示。

音符分别用 1,2,3,4,5,6,7 表示。

音名和音符的对应关系如下:

全音	C	D	E	F	G	A	B
	1	2	3	4	5	6	7

半音	#C	#D	#F	#G	#A
	1 [#]	2 [#]	4 [#]	5 [#]	6 [#]

读者要记熟每个音符所对应的音名。

每个音名对应一个固有的频率。为便于查找和使用,下面列表给出各音名所对应的频率近似值供参考。

表 2 音名频率对照表

音名	频率	音名	频率	音名	频率	音名	频率
O0 C	56	O1 G	172	O3 D	500	A	1500
D	64	A	190	E	563	B	1700
E	72	B	215	F	600	O5 C	1800
F	75	O2 C	220	G	640	D	2050
G	84	D	247	A	760	E	2500
A	95	E	280	B	860	F	2400
B	107	F	300	O4 C	920	G	2700
O1 C	110	G	340	D	990	A	3100
D	120	A	380	E	1120	B	3450
E	140	B	430	F	1180		
F	150	O3 C	445	G	1340		

F. 音符的时值

在音名后面附上 0~9 的整数用来指定音名的时值。时值与整数的对应关系见下表。

表 3 音名时值对照表

数字	音符	拍节	示例
9	全音符	4 拍	1---
8	付二分音符	3 拍	1--
7	二分音符	2 拍	1-
6	付四分音符	1 1/2 拍	1.
5	四分音符	1 拍	1
4	付八分音符	3/4 拍	$\frac{1}{2}$
3	八分音符	1/2 拍	$\frac{1}{4}$
2	付十六分音符	3/8 拍	$\frac{1}{8}$
1	十六分音符	1/4 拍	$\frac{1}{16}$
0	三十二分音符	1/8 拍	$\frac{1}{32}$

G. 休止符

休止以 R 表示。用 R0~R9 表示休止的长短。具体时间见 F 的约定。

H. 重音

演奏重音时可在字符串间加冒号实现。可演奏 2 重音、3 重音。最多为三重奏。演奏重音的格式为：

PL. “系统 A:系统 B”

PL. “系统 A:系统 B:系统 C”

(二) 语法结构和编程约定

- 首先指定主音 M、音长 V、节拍 T 和音色 Y。一经指定后，各参数值会一直保持到再次指定为止。未指定时的初值为 M0V15T4O3，且音符的时值为 5。
- 每行以 PLAY 语句引导，字符串用双引号括起，每行引号内的字符不能超过 31 个。在二、三重奏时，冒号也计算在内。
- 在定义二、三重奏时，必须先定义节拍 T，几个系统中的节拍必须一致，否则，演奏不能同步。
- 音符的时值一经指定后，若后一个音符的时值与前一个一样，可省略时值，直到改变时值时再指定，即使换行也不影响。
- 每句中的 8 度音只要后一个与前一个相同，可不必再指定，直到重新指定 8 度音为止。即使换行也不影响。
- 在三重奏时，系统 C 的音色是固定的，因此系统 C 可省略音色的指定。

(三) 编程方法：

- 把乐谱以拍分段，分段距离稍大些。一般十六开纸竖用，每行分八段。若是重奏，也对应分好。
- 在每个乐谱下面填上相应的音名，再根据每个

音名的时值对照时值表填入相应的整数。若与前一音名的时值相同，可以省略。

- 在音名前填上 8 度音 On，若与前一音名相同可以省略。当编程熟练后，b、c 两步可以一次填完。
- 将字符按每行少于或等于 31 个的规则填入 PLAY 语句。注意分行时，一拍内的音名不能分开。
- 输入计算机，进行演奏，修改错误。满意后，可写入录音磁带，以备调用。

例 21 编制《东方红》乐曲演奏程序。

首先分段写出乐谱，填上音名等。

填好核对无误后，填入 PLAY 语句。

本例设定 M1V15Y2T4。程序如下：

```
10 PL. "M1V15Y2T4"
20 PL. "O3G5G3AD7C5C3O2AO3D7"
30 PL. "G5GA3O4CO3AGC5C3O2AO3D7"
40 PL. "G5DCO2B3AG5O3GDE3DC5C3O2A"
50 PL. "O3DEDCDCO2BAG7"
60 E.
```

RUN 即可演奏出东方红乐曲。

下面再举一个三重奏的示例。

例 22 三重奏乐曲《铃儿响叮当》

《铃儿响叮当三重奏曲》是我根据彼尔彭特的《铃儿响叮当》原曲改编的。

首先分段列出三重奏曲谱，然后依次填入音名、时值、8 度音等。

《铃儿响叮当三重奏曲》

(略)

填完核对无误后，便可填入 PLAY 语句中。

由于是三重奏，必须保证每行 PLAY 语句中三个系统的节拍一致。为了保证每行不超过 31 个字符，可按节拍逐拍计算三个系统的字符数，以小于或等于 29 个字符为一句。但要注意，系统 A 中有 11 处 2 分音符或付 4 分音符，这就要求必须把该音符所在的节拍及后一节拍填在一行，否则将出现三个系统节拍紊乱。可先将字符分段，然后分别填入语句中。程序如下：

```
10 REM "LingErXiangDingDangSanChongZou"
20 PL. "M1V12Y0T3;M0V9M1V9Y2T3;M1T3"
30 PL. "O4E3EE5E3E;O3C5EG;O2C5G3O1GO2C5"
40 PL. "E5E3GCDE7;ECEGE;G3O1GO2C5GC3CDE"
50 PL. "F3FFFEEDDC;CFGEDA;F5ACGCG"
60 PL. "DG6E3EE5E3E;GBCEG;DGC3O1GO2C5"
70 PL. "E5E3GCDE7;ECEGE;G3O1GO2C5GC3CDE"
80 PL. "F3FFFEEDGG;CFGED;F5ACGD3F"
90 PL. "FDC6O3G3;FCC;O1GO2CO1GABO2C"
100 PL. "GO4EDCO3G6G1G;C5EGE;C5GC3O1GGG"
110 PL. "G3O4EDCO3A6A3;CEFD;O2C5GF3O1AAA"
120 PL. "AO4FED;DA;O2F5A"
130 PL. "O3A6A3;FD;F3O1AO2AO1A"
140 PL. "O4GGFD;ED;O2GO1GAB"
150 PL. "E6O3G3GO4E;ECC;O2CO1GGO2CC5"
160 PL. "DCO3G6G3GO4E;EGEC;GC3O1GGGO2C5"
```

170 PL. "DC03A6A3;EFD;GF301AAA"
 180 PL. "AO4FEDGGGG;AFBG;O2F5AGB"
 190 PL. "AGFDC7;FDC7;G301GABO2CO1GAB"
 200 END[或 GOTO20]

核对无误后,输入计算机,就可以欣赏这首世界名曲了。

读者可以发现,若编程熟练后可直接对照乐谱输入程序。那么输入速度并不比 MUSIC BOARD 慢。

七、F BASIC 语言的深入探讨

到此为止,本文共介绍了 90 条语句,其中有 17 条随机手册中没有介绍,是由我补充的。但是 F BASIC 语言并不仅有这 90 条语句。目前,我已发现了 26 条语句,除了以上我已弄清功能的 17 条外,还有 9 条目前尚不能确定。在此介绍给大家,愿与有兴趣的读者共同探讨。兹介绍如下:

1. SCREEN

SCREEN 简写 SC.

SCREEN 语句的功能是设置屏幕的显示模式。语句格式为

SC. n1,n2

式中 n1,n2 取值为 0,1。

2. FILTER

FILTER 简写 FIL.

语句格式为

FIL. n 式中 n 取值 0~7

3. BGGET

BGGET 简写 BGG.

语句格式

BGG.

4. BGPUT

BGPUT 简写 BGP.

语句格式

BGP.

BGET 和 BGPUT 好像是成对语句。BGGET 在前, BGPUT 在后。

5. BACKUP

BACKUP 简写 BA.

语句格式为

BA.

回车后,屏幕上出现 SYSTEM ON,功能同 SYSTEM 语句在直调 BASIC 中差不多。

6. CLICK

CLICK 简写 CLI.

7. TAB

TAB 简写 TA.

TAB 语句在一般 BASIC 中功能是按列打印输出项,但在 F BASIC 中功能不同。

8. CALL

CALL 简写 CA.

语句格式

CALL-n

式中 n 取值分别为 151、198、868 时,清屏,显示内存为 8182 BYTES FREE;当 n 取值为 958 时,清屏,显示内存为 32246 BYTES FREE;当 n 取值为 802 时,自锁,重新直调 BASIC 后显示 36342 BYTES FREE;当 n 取值为 1913 时,自锁,重新直调 BASIC 后显示 48630 BYTES FREE。好象使用 CALL 语句可以扩展内存。

9. SPC

SPC 语句的功能与一般 BASIC 也不相同。

BASIC 语言是一个大家族,如 IBMPC 机的 BASIC 语句就有 170 多条。F BASIC 语言是 BASIC 家族中的一个新成员,大家还不太熟悉,因此它有待我们去深入发掘,以发挥 F BASIC 语言应有的作用,使它在微机普及领域中开出灿烂的花朵。

为便于读者研究探讨,文末列出 F BASIC 语言的保留字表,共计 99 个。使用这些保留字可以组成各种不同功能的语句。如用 TR 可以组成跟踪语句 TRON、TROFF;用 DEF 可以组成 DEF MOVE、DEF SPRITE;用 ON 可组成 ON GOTO、SPRITE ON、ON ERROR GOTO 等。

我愿同广大键盘游戏机用户结为挚友,以互相学习交流、共同提高。

附 F BASIC 语言 99 个保留字表:

F BASIC 语言保留字

GOTO	GOSUB	RUN	RETURN	RESTORE
THEN	LIST	SYSTEM	TO	STEP
SPRITE	PRINT	FOR	NEXT	PAUSE
INPUT	LINPUT	DATA	IF	READ
DIM	REM	STOP	CONT	CLS
CLEAR	ON	OFF	CUT	NEW
POKE	CGSET	VIEW	MOVE	END
PLAY	BEEP	LOAD	SAVE	POSITION
KEY	COLOR	DEF	CGEN	SWAP
CALL	LOCATE	PALET	ERA	TR
FIND	GAME	BGTOOL	AUTO	DELETE
RENUM	FILTER	CLICK	SCREEN	BACKUP
ERROR	RESUME	BGPUT	BGGET	CAN
XOR	OR	AND	NOT	MOD
ABS	ASC	STR\$	FRE	LEN
PEEK	RND	SGN	SPC	TAB
MID\$	STICK	STRIG	XPOS	YPOS
VAL	POS	CSRLIN	CHR\$	HEX\$
INKEY\$	RIGHT\$	LEFT\$	SCR\$	INSTR
CRASH	ERR	ERL	VCT	

常用错误代码表

错误代码	错误表示	错误讯息	说明
0	NF	NEXT without FOR	无 FOR 却有 NEXT
1	SN	syntax error	语法错误
2	RG	RETURN without GOSUB	无 GOSUB 却有 RETURN
3	OD	Out of DATA	READ 读的数据未备于 DATA 中
4	IL	Illegal function call	功能调用不合法
5	OV	Overflow	演算结果超限(溢出)
6	OM	Out of memory	内存不足
7	UL	Undefined line Number	GOTO、GOSUB、IFTHEN 的行号未定义
8	SO	Subscript out of range	数组变量的下标在规定的范围之外
9	DD	Duplicate Definition	数组变量定义重复
10	DZ	Division by Zero	0 做除数
11	TM	Type Mismatch	变量的类型不一致
12	ST	String too long	字符串超过 31 个
13	FT	Formula Too complex	式子过于复杂
14	CC	Can't Continue	CONT 无法继续执行程序
15	UF		
16	MO	Missing Operand	缺少操作数
17	TP	Tape Read Error	无法从卡带正确地读资料
18	NR	No RESUME	错误处理程序无 RESUME
19	RE	RESUME Without ERROR	无错误处理程序有 RESUME
20	NB	No BGGET	无 BGGET 却有 BGPUT
21	UP	Unprintable Error	无法显示打印的错误代码

(上接 48 页)

720KB 双面盘片, 1.44MB 双面盘片。

23. 怎样检测维护硬盘驱动器?

在 PC 机系统中, 硬磁盘存储器主要包括硬盘控制器和驱动器两部分。硬盘控制器用以完成硬盘驱动器和主机之间信息传输; 硬盘驱动器主要包括机械结构和电子线路两部分。驱动器机械结构比较复杂, 它安装在主机箱中的一个密封腔体中, 对于用户绝对不能去打开它, 即使出现硬故障, 也只能送到专门生产厂家去维修。

这里介绍三种检测硬盘驱动器的方法:

(1) 利用系统自检

PC 机对硬盘驱动器具有较强的自测试能力。它是整个微机系统诊断的一部分, 这一自诊断程序被固化在主机板的一片 ROM 中, 当主机加电时, 这个诊断程序从 ROM 中调出, 对微机系统的各个部分进行自动检测, 当对硬盘进行检测时, 一旦发现故障即在屏幕上

显示“1701”故障代码, 告诉用户硬盘驱动器自检未通过, 应停机进行检查。在诊断硬盘过程中, 若出现下列故障之一:

- 控制器复位 37 次都不成功
- 控制器内部诊断有故障
- 控制器 RAM 诊断有故障
- 驱动器在 25 秒内未准备好
- 驱动器重新校准不成功
- 设置驱动器参数不成功
- 写磁盘扇区有故障

都会在屏幕上出现“1701”出错代码

(2) 利用文件检查

可以通过有关的系统文件, 直接对硬盘驱动器读/写文件的能力, 存储数据的容量以及格式化的能力进行检测。这种检测要破坏盘中原有数据和文件, 为此, 在操作之前应把硬盘上的文件进行备份, 以免丢失必要的数

(3) 利用高级诊断程序检测

PC 机随机资料中都提供了诊断盘, 诊断程序以菜单形式, 便于用户操作。

选择“0”便进入诊断过程。在这过程中主要完成: 执行读测试, 写测试, 零磁道测试, 表面扫描测试以及硬盘磁头定位操作。在上述的各项诊断中, 如一项不通过, 都将显示出故障代码, 可以通过故障代码便知道故障原因所在。

但需说明的是用诊断盘对硬盘驱动器进行检查后, 只是在某些质量和指标上得到确认, 而对硬盘驱动器是否能在实际工作中运行, 以及盘上具体有多少空间能够写进数据, 多少扇区已被破坏等等, 这些参数只能在操作系统下用文件检查进一步确定。

硬盘驱动器在使用中应从以下几方面注意维护:

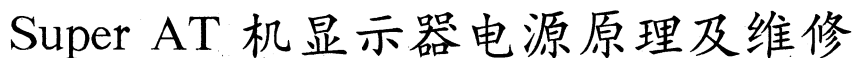
(1) 安装时要认真阅读有关用户安装使用手册一类的随机资料。仔细正确地连接好信号电缆线。

(2) 万不得已需要从机器上取下硬盘驱动器时, 也不要拆开盘体外壳螺钉, 即不要打开硬盘, 否则可能因此而报废。

(3) 硬盘驱动器一般在主机箱中, 主机箱应尽量放置平稳。否则当机器进行读/写操作时, 一旦振动, 容易出现磁头损坏盘片数据区, 造成盘内某些文件读不出来。40MB 以下容量的硬盘, 当工作完毕后, 磁头不能自动退到盘区“复位区”, 必须用 DOS 专用的固定磁头的命令 PARK.COM 或 SHIPDISK.COM, 或用诊断程序中固定硬盘磁头的功能选择。所以作为用户, 在关机之前要固定硬盘磁头。

(4) 要保持所在环境的清洁, 因为硬盘驱动器内有一个重要的机构, 包括一个位于盘体外壳上的起呼吸作用的小窗口, 如果环境空气中的灰尘太多, 就有可能使灰尘堵死这个小窗口, 那么空气过滤系统将无法正常工作, 从而导致硬盘驱动器的损坏。

(5) 要避免环境的高温和潮湿, 也要防强磁场的干扰。



镇江市科技情报所(212001) 范志盛

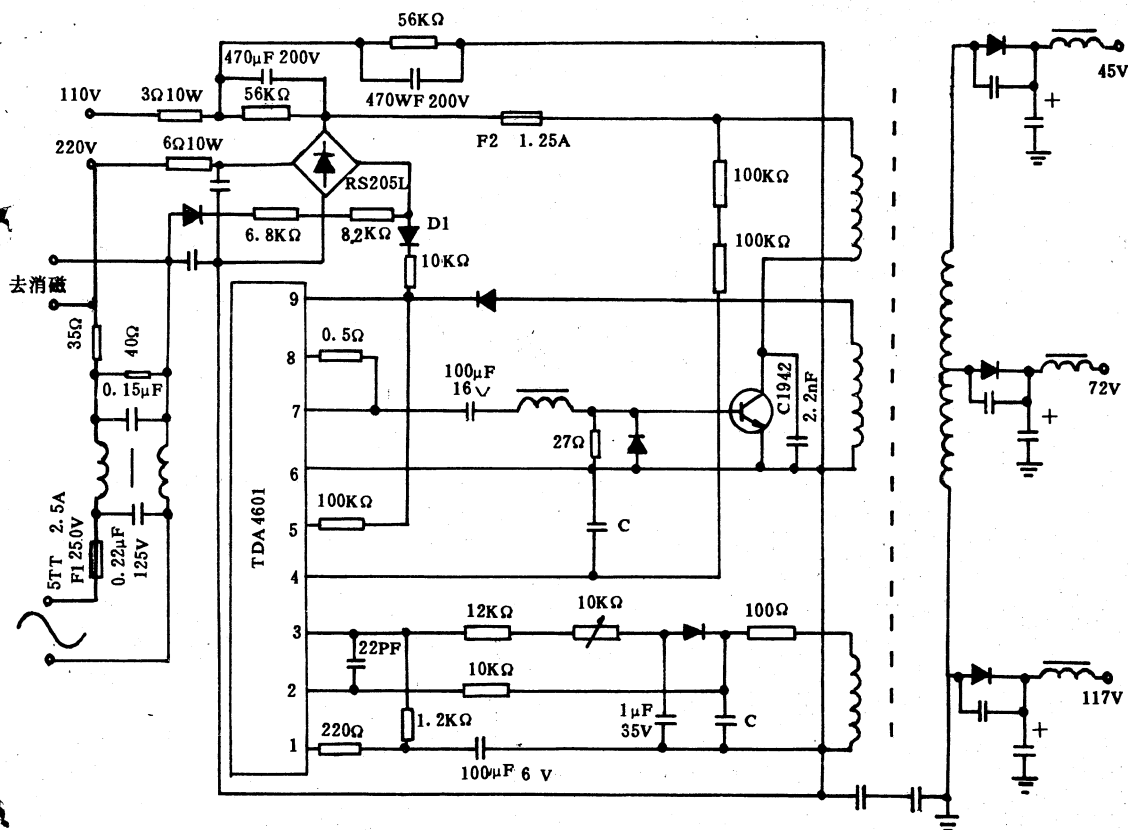
一、工作原理

Super AT 机显示器采用的是开关电源,因此它具有效率高、输出稳定可靠、负载变化范围宽等优点,也有一般开关电源结构复杂、非线性的缺点。图 1 是该电源的线路图。图及器件参数是作者实际测得。

从图 1 可见,输入的交流信号经过滤波器滤波后,分成两路,一路经过二极管 D1 半波整流后,提供给 TDA4601 作为工作电压,另一路经桥堆 RS205L 全波整流后作为开关管 C1942 工作时产生交变信号用。TDA4601 是该显示器电源电路的控制器,其输出控制

着开关管的不断导通、截止,从而在开关管的集电极上产生方波信号。

TDA4601 的外部定义见图 2。电路开始工作时,在 1 端建立一个大约 4V 的基准电压。TDA4601 的内部控制逻辑由稳定电路启动,内部各部分均受控制逻辑的控制。4 端通过一个 RC 电路,经变压器与开关管的集电极相联。它采集的信号是对集电极电流的模拟,可作为控制的依据。根据 4 端的电压值和内部的基准电压值,过载检测机构确定内部调节放大器的工作范围。4 端的输入电压由于通过了一个 RC 电路而变成锯齿



1

波,其变化范围为 2V~4V。3 端为反馈端,用作输入调节和过载保护。在变压器反馈线圈上采集的信号经过滤波后,送到 3 端。2 端用作输出的零检测。2、3、4 端均与内部控制逻辑相联。5 端与内部附加的控制电路相联。当 5 端的电压降到 2.2V 以下时,8 端的输出就被阻塞。控制逻辑驱动开关管基极电流放大器(其输出为 8 端)的开闭。8 端的输出电流正比于 4 端的锯齿电压。通过这种关系,TDA4601 能够控制开关管电流满足 $I_c = \beta I_b$ 而退出饱和区,8 端与开关管之间的电路提供电流反馈,其它电路参数可决定基极驱动电流的大小。如果控制电路停止提供基极电流,则 7 端的电压被强制降到 1.6V,使开关管截止。若 9 端的电压低于 7.4V,或 5 端电压低于 2.2V,TDA4601 也会使开关管截止。当变压器次级出现短路,TDA4601 会自动进行保护。与开关管基射极相联的二极管的作用是保护开关管。R 用来调整次级输出电压。C 的作用是在开关管输出功率超过允许值时,限制集电极电流。

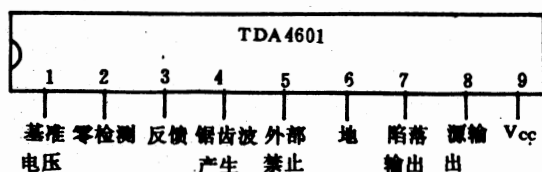


图 2

二、维修方法

当显示器出现故障时,应首先判断是否是电源板的故障。可用万用表测量电源板的三个输出是否是 45V、72V、117V。如果输出没有,多数情况下故障在电

源板上。由于开关电源原理复杂,负载端出现故障也可能引起电源板不工作。因此也要检查一下负载端是否出现故障。如判定是电源板故障,应观察两个保险管是否烧坏。

1. 如保险管 F1 烧坏,应检查桥堆 RS205L 是否损坏。按电路原理,桥堆坏的可能性较大。若桥堆完好,可更换保险管 F1 试一下。

2. 如保险管 F2 烧坏,应检查开关管 C1942 是否损坏。如损坏找不到 C1942,则可用 C1875、C1413A、C1325A、C1308 代换。

3. 如 F1、F2 都完好,TDA4601 及外围电路器件损坏的可能性较大。应首先检查外围电路的二极管、电阻、电容是否损坏,阻值、容值是否与标称值一致,是否有虚焊。首先将外围电路的故障排除。如果外围电路器件完好,就要考虑是否是集成电路块 TDA4601 损坏了。通常对集成块的判断要慎重,只有排除外围电路故障可能性后,再考虑是集成块的故障。

最后注意:因为开关电源结构复杂、非线性,同时该显示器电源中的开关管是由 TDA4601 直接驱动的,没有变压器隔离,故往往一旦出现故障,就可能有几个器件同时损坏。为方便大家维修,现将该电源正常工作时的主要参数给出如下:

桥堆正常工作时直流输出电压为 300V。开关管正常工作时 $V_{ce} = 690V(\text{交流}) = 300V(\text{直流})$ 。TDA4601 的各端电压(万用表直流电压档测得)为: $V_9 = 13V$, $V_8 = 2V$, $V_7 = 2V$, $V_4 = 2.2V$, $V_3 = 2.2V$, $V_2 = 0.16V$, $V_1 = 4.2V$ 。为方便判断 TDA4601 的好坏,将各端对地(6 端)电阻列出如下: $R_{96}(\text{表示红表笔接 9 端,黑表笔接 6 端——下同}) = R_{69} = 29.3K\Omega$, $R_{86} = 13.75M\Omega$, $R_{68} = 17.3M\Omega$, $R_{76} = \infty$, $R_{67} = 11M\Omega$, $R_{46} = \infty$, $R_{64} = 13.80M\Omega$ 。

利用 PCTOOLS 校正软盘驱动器磁头

江西拖拉机发动机厂(330044) 黄焕如

微型计算机的软盘驱动器是故障率较高的部件之一,主要表现在:软盘驱动器使用一段时间后,磁头位置产生偏差,读写数据出错;读了有霉点或者已损伤的软盘,磁头上沾染了不易清洗的介质,造成读写磁道偏位。在上述两种情况下,往往还会出现本驱动器格式化的软盘不能在其他软盘驱动器上使用、双面盘只能被格式化单面盘、启动系统时不能自举、字节数小的文件读写正常,字节数较大的文件无法读写、磁盘 0 面存入的文件能正常读写,而 1 面存入的文件读不出来、只能列目录或部分目录,不能读写文件等故障。

专业修理软盘驱动器需要校正盘(或称猫眼盘)和示波器等工具来调整磁头方位角,一般用户不但缺乏以上工具,而且也难掌握其使用方法,因此不少人利用

BIOS 中断调用 INT13H 的 0H 号(复位磁盘)和 4H 号(检测扇区)功能编制程序,来校准驱动器的磁头位置偏差。

PCTOOLS 是常用的工具软件,一般的用户很容易得到,笔者在实践中发现,利用 PCTOOLS 的拷盘功能能够很方便地校正软盘驱动器磁头位置,该方法安全可靠、简单易学、实用方便。

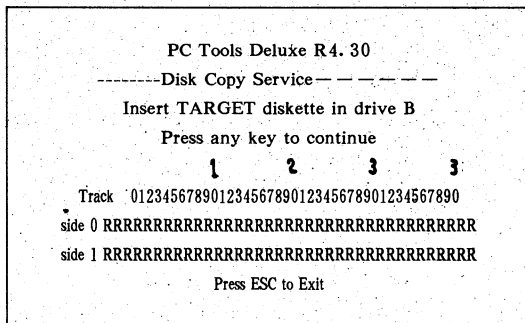
360K 软盘的存储格局一般为:每个软盘分两面,面 0 和面 1;每个面分 40 个磁道,0~39 道;每个磁道分 9 个扇区;每个扇区 512 个字节。PCTOOLS 在执行拷盘操作时,首先读源盘,读磁道的顺序是:0 面 0 道、1 面 0 道、0 面 1 道、1 面 1 道……0 面 39 道、1 面 39 道,这种读盘方式给软盘驱动器磁头的校正提供了方

便。下面详细介绍利用 PCTOOLS 校正软盘驱动器磁头的具体方法。

如果该驱动器使用的磁盘互换性不好,首先应该在正常的软盘驱动器上重新格式化一块软盘,将软盘插入需要校正的驱动器内(假定为 A 驱动器),然后在另一驱动器或者在硬盘上执行 PCTOOLS 文件,并选择拷盘功能。

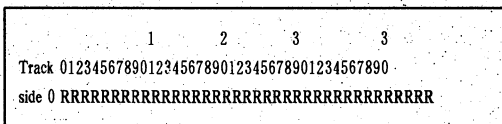
C>>PCTOOLS<<F3(磁盘)<<C(拷贝)<<A(A 驱动器)<<A(A 驱动器)<<

如果读完该盘后出现图一的画面,则表示该驱动器磁头位置正常,可进一步检验其写盘功能,说明该驱动器磁头位置基本正确。



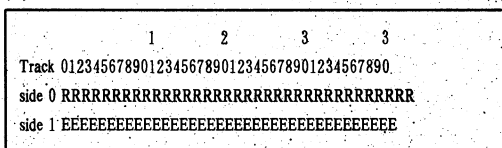
图一

如果读完该盘后出现图二的画面,则表示该驱动器上磁头(1 面)严重偏移,在该驱动器下格式化的双面软盘变成了单面软盘,或者无法读写 1 面的文件。实际上只要一开始发现不读 1 面磁道,就可以立即按 ESC 键强行中断,然后依次按 C、A、A 键,等待下一次试读。松开驱动器固定上磁头的螺钉,将上磁头清洗干净,微微移动其位置,然后拧紧螺钉,再按任一键重新读盘,直到出现图一所示的情况为止。



图二

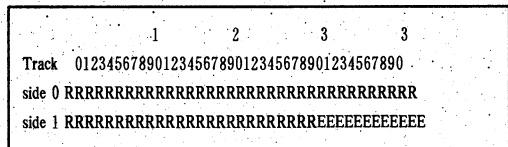
如果读完该盘后出现图三的画面,则表示该驱动器上磁头(1 面)偏移情况比图二所示的情况好一些,1 面的文件仍无法读写,目录却可列出。如上所述,在 1 面连续出现几个 E 时就可强行中断,以节省调试时间。



图三

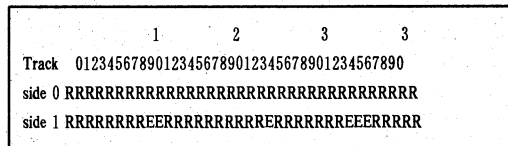
如果读完该盘后出现图四的画面,则表示该驱动器上磁头(1 面)偏移情况比图三所示的情况更好一些,存于磁盘前面磁道的文件可能可以读取,后面磁道

上的文件不能读写。这时仅仅需要微微松开螺钉,向某一方向移动上磁头位置,重新读盘后如果在 1 面出现的 E 更多,说明移动的方向和正确的方向相反;如果在 1 面出现的 E 比原来的少,说明移动的方向和正确的方向相同,反复调试直到出现图一般所示的情况为止。



图四

如果读完该盘后出现图五的画面,则表示该驱动器上磁头(1 面)位置基本正确。1 面中间出现的少量的几个 E,可能是被测试的软盘本身有坏的扇区,这时可另换一张好的盘片测试。根据经验,只要在最前面和最后面大部分磁道都是 R,中间出现少量几个 E 无关紧要,不需要重新调试磁头位置,否则会越调越乱。



图五

必须指出,以上调整方法都是假设没有驱动器的读写电路或 DMA 控制电路故障,并且驱动器的磁头已经清洗干净的情况下进行的,否则无法调整正确。由于驱动器下磁头(0 面)基本固定,同时下磁头位于下方,容易清洗和作其他处理,出现位置偏移的情况较少,以上说明的是驱动器上磁头的校调方法。

如果驱动器的上磁头偏移太严重,可能会出现 0 面也不能读的情况,在这种情况下还是要先调试上磁头。如果确系驱动器的下磁头偏位,以上介绍的方法仍然适用,不同的是应该松开固定步进电机的螺钉调试。

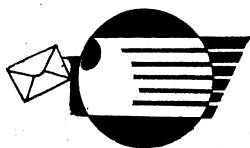
笔者利用 PCTOOLS 及上面介绍的方法,修复校调了十几部有各种读写故障的驱动器,感兴趣的读者不妨试一试。

新书邮购消息

电子测量仪器选购指南(91 年)	40 元(含邮费)
中国机械电子工业年鉴(91 年)	46 元(含邮费)
微型计算机维修实例 999	15.60 元(含邮费)
汉字 dBASE IV 基础	14.40 元(含邮费)
现代计算机技术博览	28.80 元(含邮费)
PC 机通用软件操作手册(一)	9.70 元(含邮费)
PC 机通用软件操作手册(二)	11 元(含邮费)
91 年《电子与电脑》合订本	17.80 元(含邮费)

注:邮局汇款:北京万寿路电子工业出版社发行部邮购科,在汇单附言栏内注明书名、册数

银行汇款开户行:北京市工商银行翠微路分理处
帐号:661036-40(请随信汇单一起寄书名、册数)
本部电话:813693 邮编:100036



读者联谊

普及型 PC 个人用户软件交流联谊活动 问题解答(七)

北京中国农科院计算中心(100081) 王路敬

21. 软盘控制器的主要任务是什么?使用软盘驱动器时应注意什么问题?

软盘控制器又称软卡,其主要任务是将 CPU 的并行数据流转换成软盘所需的串行数据流,以及将软磁盘的串行数据流转换成 CPU 所需的数据流。CPU 一旦向软盘控制器发出命令,软盘控制器就要进行如下的工作:选择当前软盘驱动器,控制磁头寻道和定位的机械动作;寻找所需的磁道和扇区并进行读/写操作;在主存储器 and 软盘之间进行数据传送;最后进行校验和出错检测。

主机通过软盘控制器对软盘驱动器进行控制。使用软盘驱动器需要注意下列问题:

(1)工作环境要干净通风。

(2)分清 A 驱动器和 B 驱动器。因为对无硬盘的 PC 机来说只能从 A 驱动器引导系统。

(3)插软盘片时轻轻地插入,而且必须插到位,然后慢慢地关好门。否则,关门时会压伤盘片定位孔,严

重时会使盘片损坏。

(4)盘片的取代,一般情况下双面盘可以代替单面盘,双密度软盘可代替单密度盘,反之不然。

(5)当软盘驱动器指示灯亮时,不能打开驱动器的门从驱动器中抽出盘片,因为指示灯亮,表示软盘驱动器正在运行,若这时硬要开门取盘片,很容易破坏盘片中的有用数据,或者损坏磁头,只有在驱动器指示灯熄灭后,再开门取出盘片。

(6)定期的对驱动器磁头进行清洗。清洗时可以用清洗盘清洗,但注意时间一般掌握在 1—2 分钟,否则时间长了会损坏磁头。清洗盘市面上有售。也可以用擦照相机的镜头纸或用棉球沾异丙醇轻擦磁头 1—2 遍即可。

22. 软盘驱动器与软盘的兼容性是如何规定的?

目前在 PC 机上装置的软盘驱动器规格大都是 5.25 英寸或 3.5 英寸,如下表所示:

尺寸	说明	容量
5.25 英寸	单面	160KB/180KB
5.25 英寸	双面	320KB/360KB
5.25 英寸	双面高容量	1.2MB
3.5 英寸	双面	720KB
3.5 英寸	双面	1.44MB

与各类软盘驱动器相兼容的软盘片从尺寸上也有 5.25 英寸和 3.5 英寸之分。如下表所示:

尺寸(英寸)	说明	容量	磁道数	磁头数	扇区/道	字节/扇区
5.25	单面	160KB/180KB	40	1	8/9	512
5.25	双面	320KB/360KB	40	2	8/9	512
5.25	双面高容量	1.2MB	80	2	15	512
3.5	双面	720KB	80	2	9	512
3.5	双面	1.44MB	80	2	18	512

当用 DOS 命令到软盘片中去读写时,必须考虑到软盘片与软盘驱动器的兼容性,具体规定如下:

(1)160KB/180 KB 单面 5.25 英寸驱动器,可以在这种驱动器上读/写的盘片只有:160KB/180KB 单面倍密度盘片。

(2)320KB/360KB 双面 5.25 英寸驱动器,可以在这种驱动器上读/写的盘片有:160KB/180KB 单面,倍密度;320KB/360KB 双面倍密度。

(3)1.2MB 高密度 5.25 英寸驱动器,可以在这种驱动器上读/写的盘片有,160KB/180KB 单面,倍密度盘片;320KB/360KB 双面,倍密度盘片;1.2MB 高密度盘片。

(4)720KB 双面 3.5 英寸驱动器。可以在这种驱动器上读/写盘片只有,720KB 双面盘片。

(5)1.44MB 双面 3.5 英寸驱动器,可以在这种驱动器上读/写的盘片, (下转第 44 页)