



微小型计算机

# 开发与应用

MICRO-MINICOMPUTER  
DEVELOPMENT & APPLICATION

1990 1

微小型计算机开发与应用编辑部



# 《微小型计算机开发与应用》

1990年第1期目录

## 编辑委员会

(总第45期)

顾问	郭平欣		
主编	黄侃		
副主编	王治宝	邹秀凤	
委员	(以下按姓氏笔划排列)		
	于万源	于清汶	王治宝
	王 镭	王士禧	王寿松
	付国明	许镇宇	朱植松
	曲庭维	李凤祥	刘连棣
	陈力为	邹秀凤	吴锦声
	房家国	夏纪寅	夏业勋
	袁维本	曹东启	黄侃
	黄宝良	章渭臣	梅克定
	童宣明	裴少峰	薛大中

编辑:《微小型计算机开发与应用》编辑部

出版:天津市电子计算机研究所

天津市电子计算机学会

地址:天津市河西区友谊路爱民道5号

(邮政编码:300061)

发行:天津市邮局

印刷:天津市武清县长宏印刷厂

订购处:全国各地邮局

定价:0.95

## 计算机软件

控制系统CAD软件系统的设计.....

王治宝 王秀峰 陈曦 许广 (1)

影子价格经营管理系统.....

.....刘於勋 肖华 吴振庆 (7)

24×24点阵通用高级汉字打印驱动软件的设计

与实现.....王磊 刘凤荣 (10)

扩充C语言的矩阵运算功能.....

.....陈德谦 (14)

提高VAX Rdb/VMS 数据库存取速度的探讨

与实现.....王国星 (18)

## 应用实例

在APPLE II 机上利用128K卡建立扩展数组

区BASIC解译系统.....尚银生 (22)

小球病毒的分析 and 防治.....

.....李建 李学泓 (25)

关于在微机上将3维图形运用于工程CAD的

几个问题的认识和处理.....

黄育群 朱世铁 王巨森 张大坤 (28)

微机三相全控整流可逆直流传动装置的研制

.....邹用才 (31)

单片机微机通信控制器.....

.....王东平 (34)

## 新技术介绍

开发内藏MMU 的68030.....

.....许薇 江岳 (37)

莫托洛拉的MC68040.....陈荣华 (41)

AST Premium/386C 概述.....

.....刘铁军 张皓月 (42)

VAX—8350 计算机的诊断系统.....

.....李瑞成 (44)

## 经验点滴

排除接地干扰的几则实例...张远长 (47)

用WIDTH命令定义屏幕显示的行宽.....

.....石爱华 (48)

## CONTENTS

## SOFTWARE

Design of Control System CAD Software System	.....Wang Zhibao	Wang Xiufeng
	Chen Xi	Xu Guang ( 1 )
Shadow Price Management System	.....Liu Yuxun	Xiao Hua Wu Zhenqing ( 7 )
The Design and Implementation of Printing and Driving Software for Advanced Universal Chinese Character with 24×24 Matrix	.....Wang Lei	Liu Fengrong ( 10 )
Enlargement of C Language for Matrix Operations	.....Chen Deqian	( 14 )
Study and Implementation of Improving VAX Rdb/VMS Access Performance	.....Wang GuoXing	( 18 )

### APPLICATION EXAMPLE

Using 128k Card to Set up Interpretive BASIC System with Expandable  
Array Area on APPLE II .....Shang Yinsheng ( 22 )

Analysing and Curing the Bouncing Ball Virus  
.....Li Jian Li Xuehong ( 25 )

The Opinions and Solutions on Some Problems about Applying 3D  
Graph to Engineering CAD System on Microcomputers  
.....Huang Youqun                      Zhu Shitie  
   Wang Jusen                      Zhang Dakun ( 28 )

Developing the 3 Phase Control Rectifying & Reversible DC Transm-  
ission Equipment With Microcomputer..... Zou Yongcai ( 31 )

Communication Controller with Single Chip Microcomputer  
.....Wang Dongping ( 34 )

## NEW TECHNOLOGY

Developing the 68030 Having Internal MMU.....	Xu Wei Jiang Yue ( 37 )
Motorola's MC 68040.....	Chen Ronghua ( 41 )
Introduction to AST Premium/386C.....	Liu Tiejun Zhang Haoyue ( 42 )
Diagnostic System for VAX—8350.....	Li Ruicheng ( 44 )

## EXPERIENCE

Some "Practical Examples of Eliminating the Grounding Noise  
..... Zhang Yuanchang ( 47 )

Defining the Line Width of Screen Display with Command WIDTH  
..... Shi Aihua ( 48 )

# 控制系统CAD软件系统的设计 —CADCS的总体结构实现



南开大学

王治宝 王秀峰 陈曦 许广

**摘要** CADCS总体设计提出了控制系统CAD软件系统的一个文本,本文给出了该系统的实现方法。它包括混合式人机交互界面,总体调度结构,非宏模式运行系统,宏模式运行系统,以及在线编辑系统。

## 一、引言

CADCS总体设计给出了控制系统CAD软件系统文本,这个文本的提出,应以它的可实现性为依据,否则是没有意义的。本文给出了该软件系统总体结构的实现方法。它包括总体调度结构,实现了混合式人机交互的模式与回授选择;非宏模式运行,与总体调度结合实现了从工作模式,回授,到宏生成,非宏执行,子系统类,子系统等的层层选择过程;宏模式运行,实现了从开始,断点或指定点不同方式的宏执行过程,并给出在线编辑系统,从而构成了CADCS的总体结构实现方案,使其很容易进行编码并投入运行。

## 二、CADCS总体调度结构

CADCS软件系统的人机界面与调度结构由CADCS总体调度,非宏模式运行调度与宏模式运行调度构成。

CADCS总体调度的程序概要为:

```
for ( ;; ) / • CADCS总体调度 • /  
{  
    模式选择  $\phi$  / 1/2/3/4 送入MODE  
    if ( MODE ==  $\phi$  )  
        break; / • 退出总体调度 • /  
for ( ;; )  
{
```

回授选择  $\phi$  / 1/.../7 送入ECHO

if ( ECHO ==  $\phi$  )

break;

if ( MODE<sub>1</sub> = 4 )

调用非宏模式运行调度

else

调用宏模式运行调度

}

}

这个概要用C语言的伪代码书写,其中的符号见文献[1],本文的所有程序概要都用该伪代码。

### 1. 模式选择

当从操作系统进入CADCS软件系统时,首先进行模式选择,如表1所示,当选 $\phi$ 时,返回到操作系统,当选1/2/3/4时,分别选择相应的模式运行。

表1 工作模式选单

模式选择  $\phi$  / 1/2/3/4

$\phi$ . 返回操作系统

1. 选单模式

2. 友好选单模式

3. 命令模式

4. 宏命令文件模式

### 2. 回授选择

如表2所示,若选 $\phi$ ,表示返回模式选择,否则可回授要求形式的命令信息。为了给用户以仔细观察的机会,在回授过程中,提供了信息驻留屏幕的可能性。



表2 回授选单

回授选择  $\phi/1/2/3/4/5/6/7$   
 $\phi$ . 返回模式选择  
 1. 回授命令行组  
 2. 回授命令行组且回授每行后暂停  
 3. 回授选单式命令信息  
 4. 回授选单式命令信息且有暂停  
 5. 回授友好选单式命令信息  
 6. 回授友好选单式命令信息且有暂停  
 7. 不回授

### 3. 两个子调度过程

当模式选择为选单，友好选单或命令模式时，由非宏模式运行调度实现其调度过程；当模式选择为宏命令文件模式时，由宏模式运行调度实现其调度过程。

## 三. 非宏模式运行

非宏模式运行调度的程序概要为：

```
for ( ; ) / • 非宏模式运行调度 RCNMM • /
{
    生成宏命令选择  $\phi/1/2$  送入 SPMC
    if (SPMC ==  $\phi$ )
        break; / • 退出非宏模式运行调度 • /
    if (SPMC == 1)
        用生成宏命令名打开宏命令文件
        for ( ; ; )
        {
            非宏模式执行选择  $\phi/1/2$  送入 SEXE
            if (SEXE ==  $\phi$ )
                break;
            for ( ; ; )
            {
                子系统类型选择 • /1/2/3 送入 SSST
                if (SSST ==  $\phi$ )
                    break;
                for ( ; ; )
                {
                    Switch (SSST)
                    {
                        Case1;
```

管理调度子系统  $\phi/1/2$  送入 SSSN

break;

Case 2;

设计功能子系统  $\phi/1/2/\dots/15$  送入 SSSN

break;

default;

基础计算子系统  $\phi/1/\dots/4$  送入 SSSN

break;

}

if (SSSN ==  $\phi$ )

break;

if (SPMC == 1)

将 SSST 与 SSSN 相应的子系统名写入

生成宏命令文件

依 SSST 与 SSSN 调用相应子系统

并给出“子系统返回条件”

}

}

}

### 1. 宏生成选择

如表3所示，如果选  $\phi$ ，那么返回总体调度的回授选择；如果选 1，那么表示不管在选单或友好选单，还是在命令模式下，一律自动生成相应的宏命令文件，同时要求用户给出宏命令文件名；选 2，表示不要求生成宏命令文件。

表3 生成宏命令选单

生成宏命令选择  $\phi/1/2$

$\phi$ . 返回模式选择

1. 自动生成宏命令文件

选 1 时，输入宏命令文件名；

2. 不生成宏命令文件

### 2. 执行选择

选单输入，友好选单输入或命令输入后，是否要求其执行，可以作出选择如表4所示。

### 3. 子系统类型选择

如表5所示，选  $\phi$  则返回执行选择，选 1/2/3 分别进行三个子系统类的选择。

表4 执行选单

执行选择 $\phi/1/2$

$\phi$ . 返回生成宏命令选择

1. 选单/友好选单/命令将被执行
2. 不执行选单友好选单和命令

表5 子系统类型选单

子系统类型选择 $\phi/1/2/3$

$\phi$ . 返回执行选择

1. 管理调度子系统类
2. 设计功能子系统类
3. 基础计算子系统类

#### 4. 子系统选择

管理调度子系统类选择如表6所示。设计功能调度子系统类选择如表7所示。基础计算调度子系统类选择如表8所示。在这三类子系统中，如果选中，那么返回子系统类型选择，否则选中该类中的子系统名。

表6 管理调度子系统类选单

管理调度子系统类选择 $\phi/1/2$

$\phi$ . 返回子系统类型选择

1. 控制命令子系统
1. 控制系统数据库处理子系统

#### 5. 生成宏命令

如果要求生成宏命令，那么将SSST和SSSN相应的子系统名GCNAME写入生成宏通道NUPM，以作为宏命令文件的一条命令。子系统名如表9所示。

6. 调用SSST与SSSN确定的子系。

### 四、宏模式运行

宏模式运行调度程序概要如下：

表7 设计功能子系统类选单

设计功能子系统类选择 $\phi/1/2/\dots/15$

$\phi$ . 返回子系统类型选择

1. 单量测单变量线性子系统
2. 多量测单变量线性子系统
3. 系统辨识子系统
4. 时间序列建模与预报子系统
5. 自适应与多层递阶预报子系统
6. 状态空间法与振动控制子系统
7. 频域法子系统
8. 多项式阵法子系统
9. 单变量自适应控制子系统
10. 多变量自适应控制子系统
11. 模型参考自适应控制子系统
12. 非线性优控子系统
13. 非线性控制子系统
14. 模型处理子系统
15. 仿真子系统

表8 基础计算子系统类选单

基础计算子系统类选择 $\phi/1/2/3/4$

$\phi$ . 返回子系统类选择

1. 线性代数基础库子系统
2. 多项式阵基础库子系统
3. 优化基础库子系统
4. 随机数据处理子系统

表9 子系统名与对应编码

子系 统名	编码	子系 统名	编码	子系 统名	编码
FEIGN	$\phi\phi 1$	SSSSPC	$2\phi 6$	SSMDPR	214
CNTRLC	$1\phi 1$	SSFRDM	$2\phi 7$	SSSMLT	215
DATABP	$1\phi 2$	SSPLMT	$2\phi 8$	SSBBLA	$3\phi 1$
LSSSSV	$2\phi 1$	SSSCSV	$2\phi 9$	SSBBPM	$3\phi 2$
LSMSSV	$2\phi 2$	SSSCMV	$21\phi$	SSBBOP	$3\phi 3$
SSMDID	$2\phi 3$	SSSCMR	211	SSRDPR	$3\phi 4$
SSMSMD	$2\phi 4$	SSNLOC	212		
SSPRDC	$2\phi 5$	SSNLCR	213		

```

for( ; ) /* 宏模式运行调度RCMM */
{
    宏执行选择 $\phi$ /1/2/3/4送入SMEXE
    if (SMEXE== $\phi$ )
        break;
    Switch (SMEXE)
    {
        Case1: /* 从开始运行宏命令 */
            读宏命令文件名送入GCNAME
            宏嵌套层次初值 $\phi$ 送入NOM
            零层宏文件名X送入FNAME(NOM)
            零层宏文件中命令行号 $\phi$ 送入CL(NOM)
            break;
        Case2: /* 从断点运行宏命令 */
            从CL(NOM)减去1
            break;
        Case3: /* 从指定点运行宏命令 */
            显示带行号和现行命令标志的宏命令该行号送入
            CL(NOM)
            break;
        default:
            调用编辑系统
            break;
    }
    if(SMEXE==4)
        Continue;
for( ; ) /* 宏命令的合法性处理 */
{
    if (SMEXE==1)
    {
        if(GCNAME不是宏文件) /* 宏合法吗? */
        {
            显示第NOM层FNAME(NOM)中的
            第CL(NOM)
            行命令GCNAME非法或完成
            break;
        }
        NOM增1 /* 进入一层宏嵌套 */
        GCNAME送入FNAME(NOM)
        命令行初值 $\phi$ 送入CL(NOM)
        打开宏文件FNAME(NOM)
        先导命令初值"FEIGN"送入GCNAME
    }
for( ; ) /* 子系统的合法性处理 */
{
    if (SMEXE==1)

```

```

{
    按子系统常数COSSN将GCNAME
    译码送入CSSN
    if (CSSN不合法)
        break;
    if(ECHO != 7)
        按ECHO 回授GCNAME
}
if(SMEXE != 1) /* 将断指定点化为开始点
运行 */
    将1送入SMEXE
for( ; ) /* 反复调用子系统 */
{
    依CSSN调用相应的子系统返回ERR-
    OR
    if(ERROR==2 || ERROR==1 ||
    ERROR== -1 && NOM==1)
        break;
    NOM减1
}
}
}
}

```

### 1. 宏执行选择

宏执行选择如表10所示。选 $\phi$ 时，返回总体调度的回授选择。

表10 宏执行选单

宏执行选择 $\phi$ /1/2/3/4 $\phi$ . 返回回授选择 1. 从宏命令文件的开始执行 2. 从断点执行 3. 从指定的命令行执行 4. 编辑
---

#### (1) 从开始运行

当宏执行选择为1时，从宏命令的第一条命令开始执行。

首先，由用户给出宏命令文件名GCNAME。

其次，置初值。

用NOM记宏命令的嵌套次数，宏命令GCNAME开始运行前，将嵌套初值 $\phi$ 赋给NOM。当宏命令运行时，为了记忆正在执行的宏命令，假设第 $\phi$ 层的文件名为X，并将其送

入文件名的第 NOM 个元素 FNAME (NOM)，开始时这是一个起始元素。视 GCNAME 为第  $\phi$  层嵌套的一条命令，并将其行号 1 送入命令行记忆区的第 NOM 个元素 CL (NOM)，这也是一个起始元素。

这就为宏命令从开始运行作好准备。

#### (2) 从断点运行

在 NOM 层中，当执行宏命令 FNAME (NOM) 的第 CL (NOM) 行的一条命令时，发生中断。当错误纠正后，要求重新执行 CL (NOM) 行上的命令，这就要重新从宏文件读入这条命令，因此必须从 CL (NOM) 减去 1。

这就为宏命令从断点运行作好准备。

#### (3) 从指定点运行

被中断的命令是现行命令，为了观感上的鲜明性，为现行行作上一个标志，同时为了寻找作为重新运行始点命令，每条命令上标有行号是适宜的。因此当寻求重新运行始点时，首先显示带行号和现行行命令名标志的宏文件命令，并将重新运行命令的行号减 1 送入 CL (NOM)。

这就为从指定点运行宏命令作好了准备。

#### (4) 编辑

当宏发生中断时，用在线编辑系统对其进行修改。修改后可重新进行宏选择，以求得宏命令从断点运行或指定点运行。

### 2. 宏命令的合法性处理与伴子系统

#### (1) 宏命令的非法处理

当宏文件从开始运行时，如 GCNAME 不是宏文件，那么显示错误信号为

“非法命令/非法命令参数”

其注释为

NOM	嵌套层数
FNAME	宏文件名
CL	NOM 层内命令行号
GCNAME	非法命令或参数的命令名并返回到宏执行选择。

宏命令名是一个扩展名为 .MCN 的文件名，可以用该扩展名为特征识别宏命令的真伪。

#### (2) 进入一层嵌套

当宏命令合法时，进入一层宏嵌套，为此：

1) 嵌套次数 NOM 增 1。

2) 将宏命令 GCNAME 保存到一串文件名 FNAME 的第 NOM 个元素内，以备从内层返回继续运行时使用。

3) 将第 NOM 层命令行初值  $\phi$  置入各层命令行记忆区 CL 的第 NOM 个元素。

4) 打开现行的宏文件 FNAME (NOM)

#### (3) 伴子系统的引入

取子系统名初值 FEIGN 作为现行子系统名送入 GCNAME，称 FEIGN 为伴(或伪)子系统。FEIGN 是一个如下的过程：

从宏文件 FNAME (NOM) 读入命令名送入 GCNAME

将标志 “2” 送入 ERROR

返回

### 3. 子系统的合法性处理与断点运行的初始化

#### (1) 子系统译码

当宏文件从开始运行时，首先按子系统名常数 COSSN 将子系统名 GCNAME 译为 SSNC。子系统常数 COSSN 与其相应的编码 CSSN 如表 9 所示。

#### (2) 子系统的非法处理

如果 SSNC 不合法，该子系统非法，那么 GCNAME 有可能是宏命令文件名，也可能是非法命令名。

在子系统中每读一条命令名时，不管它是命令名，还是子系统名，还是宏文件名，或是非法命令名，都用 CL (NOM) 计数，当读命令行组的其它命令行时，单独计数为 CLG，如果命令名解释正常返回，那么 CLG 加入 CL (NOM)，否则 CLG 被忽略。

#### (3) 子系统名的回授

在子系统合法的情况下，如果要求回授，那么按回授选择 ECHO 的要求回授子系统名 GCNAME。

#### (4) 将断点定点化为始点运行

如果要求是从断点或从指定点运行，即



选择SMEXE不为1,那么到此将其化为从开始运行,即将SMEXE置为1。因为从此无论是从断点还是从指定点运行,都与从开始运行的状态完全一样。

#### 4. 反复调用子系统

(1) 调用子系统及其返回信息ERROR

依子系统名译码CSSN 调用相应的子系统投入运行,子系统的返回标志为

$$\text{ERROR} = \begin{cases} 1 & \text{命令/参数错、反常返回} \\ 2 & \text{子系统命令名译码失败返回} \\ -1 & \text{执行宏返回RETURN的返回} \end{cases}$$

(2) 子系统返回处理

当ERROR=2时,在GCNAME 中带回的信息不是该子系统的一条命令名,可能为子系统名或宏命令名,也可能是非法命令名。这种情况应返回到子系统名的译码处理,必要时,自然地涉及到宏命令及非法命令名的处理。

当ERROR=1时,因为它表示子系统中命令参数的错误,应返回到宏执行选择,以备修改错误。在命令名,子系统名和宏命令名不同的约定下,自然用ERROR=2的处理方法就可得到错误处理的结果。

当ERROR=-1时,要求退出当前的宏命令。如果是最外层宏命令,即层号NOM=1,那么宏命令已被执行完毕,应返回到宏执行选择,以备执行一个或编辑一个新的宏命令。在命令名RETURN与子系统名和宏命令名不同的约定下,自然用ERROR=2的处理方法得到正常运行的结果。

当ERROR=-1但运行完毕的宏不是最外层,那么将宏退层,即NOM减1,重新调用子系统。

此时,调用的子系统还是控制命令子系统,当它读入一条命令时,如果不是控制命令,那么恰好可继续工作;如果是控制命令,那么一定会按ERROR=2返回,使系统得到正常运行。

### 五、在线编辑系统

当宏执行选择选到编辑时,调用该系统。

在线编辑ED的功能是在CADCS 软件系统下,对于正在运行的宏文件或用户指定的任意文件进行修改,也可生成一个新文件。

#### 1. 编辑ED

在线编辑过程的程序概要为:

```
for( ; ) /* 在线编辑系统 */
{
    在操作窗口上作编辑文件选择  $\Phi$  / 1/2 送入SEF
    if (SEF ==  $\Phi$ )
        break;
    在编辑窗口上显示文件内容
    for( ; )
    {
        在操作窗口上作行/字符编辑选择  $\Phi$  / 1/2 送入
        SELC
        if (SELC ==  $\Phi$ )
            break;
        Switch (SELC)
        {
            Case1;
            调用行编辑过程EDL
            break;
            default;
            调用字符编辑过程EDC
            break;
        }
    }
}
```

其中两个选择的选单如表11和表12所示。

#### 2. 行编辑EDL

行编辑过程的程序概要为

```
for( ; ) /* 行编辑过程EDL */
{
    在操作窗口上进行行编辑选择  $\Phi$  / 1/2/3/4 送入SEL
    if (SEL ==  $\Phi$ )
        break;
    在操作窗口上的参数位置上读入参数
    依选择SEL的不同调用相应的过程
}
```

其中选择的选单如表13所示。

#### 3. 字符编辑

# 影子价格经营管理系统

郑州机械专科学校 刘於勋 肖 华 吴振庆

**摘要** 本文介绍了在PC机上运行的〈影子价格经营管理系统〉,所采用的数学模型、计算方法,阐述了该系统的功能及结构。

影子价格(Shadow Price)是指在资源增加时,对最优收益发生的影响,它是针对具体企业具体产品而存在的一种特殊价格。如果某种资源的市场价格低于影子价格,企业就应该购进这种资源。因此,它也

称之为资源的边际产出或资源的边际成本。由于影子价格具有比实际价格更为重要的潜在价值,所以,掌握影子价格对企业管理人员在各方面进行有效地经营管理是非常重要的。因此我们在PC机上开发了“影子价格经营管理系统”。

表11 编辑文件选单

编辑文件选择  $\Phi/1/2$

$\Phi$ . 返回宏执行选择

1. 编辑现行宏文件

2. 编辑指定的文件

输入指定的文件名:

表12 行或字符编辑选单

行或字符编辑选择  $\Phi/1/2$

$\Phi$ . 返回编辑文件选择

1. 行编辑

2. 字符编辑

表13 行编辑选单

行编辑选择  $\Phi/1/2/3/4$

$\Phi$ . 返回行或字符编辑选择

1. P L1 L2 显示

2. I L1 L2 插入 \E结束

3. D L1 L2 删除

4. M L1 L2 L3 移动

依选择字符编辑命令调用其相应过程

其中选择的选单如表14所示。

表14 字符编辑命令

↑ 光标上移	↓ 光标下移
← 光标左移	→ 光标右移
I 插入字符	D 删除字符
E 结束字符编辑	

## 六、结束语

本文给出的实现方法已作了编码,并与其子系统共同在IBM-PC/XT上运行,是可行的。这个设计正准备在中国控制系统计算机辅助设计软件系统上得到进一步的应用。

## 参考文献

- [1]王治宝,王秀峰,陈 曦,许 广,控制系统CAD软件系统的设计——CADCS总体设计,1988.9.
- [2][美]Thomas Plum著,金民忠,张子让译,周柏生校,C程序设计教程,科学普及出版社,1986.8.
- [3]王治宝,王秀峰,陈 曦,许广,控制系统CAD软件系统的设计——CADCS子系统的实现,1989.9.

字符编辑过程的程序概要为:

```
for (; ) / * 字符编辑EDC */
```

```
{
```

```
在操作窗口上选择字符编辑命令送入CHRCT
```

```
if (字符编辑为“E”)
```

```
break;
```

## 一、数学模型的建立

计算影子价格,首先需要建立具有经济意义的线性方程组,假定某企业有 $n$ 种产品, $m$ 种资源,每种产品的产量为 $X_i$  ( $i=1,2,\dots,n$ ),第 $i$ 种产品消耗第 $j$ 种资源的单位数为 $a_{ij}$  ( $i=1,2,\dots,n, j=1,2,\dots,m$ ),销售第 $i$ 种产品的单价为 $C_i$  ( $i=1,2,\dots,n$ ),使用第 $j$ 种资源的上限值为 $b_j$  ( $j=1,2,\dots,m$ ),那么,要在有限资源的条件下,如何安排各种产品的产量,才能实现最高的产值目标?根据上述条件建立如下线性方程组:

$$\text{目标函数 } Z_{m,n} = \sum_{i=1}^n C_i X_i$$

$$\begin{cases} a_{11}X_1 + a_{12}X_2 + \dots + a_{1n}X_n \leq b_1 \\ a_{21}X_1 + a_{22}X_2 + \dots + a_{2n}X_n \leq b_2 \\ \vdots \\ a_{m1}X_1 + a_{m2}X_2 + \dots + a_{mn}X_n \leq b_m \\ X_1, X_2, \dots, X_n \geq 0 \end{cases}$$

对上述线性方程组采用单纯形法或改进单纯形法进行迭代计算,可求得此线性方程组的最优解,在线性方程组有最优解的条件下,将最优表中基变量的目标函数系数所组成的行向量 $C_B$ 和基变量的基矩阵(就是由原始系数矩阵中对应于基变量 $X_B$ 的列所组成的矩阵 $B$ )的逆矩阵,按照公式 $\pi = C_B \cdot B^{-1}$ 计算出单纯形算子,假如用 $Y$ 表示总资源的影子价格,则 $y = -\pi = -C_B \cdot B^{-1}$ ,它表示总资源在最优产品组合方式下,所具有的“潜在价值”或“贡献”。

## 二、算法设计

由于单纯形算法计算速度慢,又占有较多的内存空间,为此,本系统采用了改进单纯形法以求得方程组的最优解,其算法步骤如下:

### 1. 输入数据处理

首先从键盘上输入约束条件的个数 $m$ ,自变量个数 $n$ ,然后在全屏幕编辑形式下,将线性方程组的系数矩阵,不等号标志,常数项输入到计算机中,使它们分别存放在 $A(i,j)$ ,  $A(n+m+1,j)$ 和 $(m+n+2,j)$ 数组,其中 $i=1,2,\dots,n, j=1,2,\dots,m$ ,不等号标志的输入若“ $\leq$ ”则输入+;若为“ $\geq$ ”,则输入-1;若为“ $=$ ”,则输入0。数据输入完后,输入目标函数系数,存于 $A(1,n)$ ,当已知数据录入完毕,系统先对此线性方程组进行标准化处理,然后,自动对 $n$ 个变量, $m$ 个约束条件的方程组及各个基变量逐一进行枢轴计算,将 $m$ 个基变量的系数矩阵变成 $m \times n$ 阶单位矩阵,使之成为典范化型方程组形式如表1:

表1

$X_1 X_2 \dots X_m X_{m+1} X_{m+2} \dots X_{m+n}$	不等号标志	常数项系数
1 0...0 $\bar{a}_{1,m+1} \bar{a}_{1,m+2} \dots \bar{a}_{1,n}$	0	$\bar{b}_1$
0 1...0 $\bar{a}_{2,m+1} \bar{a}_{2,m+2} \dots \bar{a}_{2,n}$	0	$\bar{b}_2$
$\vdots$	$\vdots$	$\vdots$
0 0...1 $\bar{a}_{m,m+1} \bar{a}_{m,m+2} \dots \bar{a}_{m,n}$	0	$\bar{b}_m$
基变量项	非基变量项	常数项

将基矩阵存放在 $(B_{m,m})$ 数组;基变量的目标函数系数存放在 $L_B(m)$ 数组;基变量的下标描述存放在 $X_B$ 数组;非基变量的原始约束条件系数所组成的矩阵存放在 $N(m+n,m)$ 数组;非基变量的目标函数系数存放在 $L_N(m+n)$ 数组;非基变量的下标描述存放在 $X_N$ 数组,以便存放每次迭代结果。

### 2. 计算非基变量检验数,确定进基变量

非基变量检验数 $\bar{C}_N$ ,按 $\bar{C}_N = C_N - C_B \cdot B^{-1} \cdot N$ 进行计算,如果检验数有正值,从所有正值中找出最大值所在的列,作为进基变量的下标,假如 $\bar{C}_s$ 最大,则变量 $X_s$ 成为进基变量,将其存放到 $X_B$ 数组,生成新的基变量列向量。



### 3. 确定出基变量

当进基变量 $X_s$ 确定之后,将 $X_s$ 变量在约束条件系数矩阵中所对应的列向量 $a_{sj}$  ( $j=1,2,\dots,m$ )与常数项系数 $b_j$  ( $j=1,2,\dots,m$ )相除,按商值最小原则,确定出基变量所在的行,假设 $Q=\{ \frac{b_j}{a_{sj}} | a_{sj} > 0 \}$ ,

$j=1,2,\dots,m$  } =  $\frac{b_r}{a_{sr}}$  那么在 $X_B$ 数组中, $X_B(r)$ 所对应的基变量为出基变量,录入 $X_N(S)$ 单元,成为新的非基变量。

### 4. 计算单纯形算子 $\pi$

进基变量 $X_s$ 确定之后,新的基变量对应于原约束条件系数所组成的基矩阵 $B$ 就生成了,对基矩阵进行求逆得到 $B^{-1}$ ,并送入 $BB$ 数组,然后按计算单纯形算子公式 $\pi = C_B \cdot B^{-1}$ ,将新的基变量目标函数系数所组成的行向量 $C_B$ 与基矩阵的逆矩阵 $B^{-1}$ 相乘,求得单纯形算子 $\pi$ 。

### 5. 判断是否继续迭代

通过计算非基变量检验数 $\bar{C}_N = C_N - C_B \cdot B^{-1} \cdot N$ 可确定是否继续迭代。若所有非基变量的检验数都为负值,则迭代终止,得最优解;若非基变量检验数中出现正值,则取正数中最大数所对应的非基变量作为新的进基变量。

### 6. 计算新的常数项系数和进基变量系数

利用新的基变量基矩阵的逆矩阵 $B^{-1}$ 和原约束条件方程组中常数项系数 $b_j$  ( $j=1,2,\dots,m$ )进行乘法运算,求出新的常数项系数,记作 $\bar{b}$ ,进基变量 $X_s$ 的系数列向量的求法是按照公式:  $\bar{a}_{sj} = B^{-1} \cdot a_{sj}$  计算的,其中 $a_{sj}$  ( $j=1,2,\dots,m$ )表示进基变量 $X_s$ 在原约束条件系数矩阵中所对应的列向量。

### 7. 返回第3步骤。

重新进行计算,确定出基变量,并判断所有非基变量的检验数是否有负值。

如果非基变量的检验数全部为负值,终止迭代,求目标函数值;否则,继续迭代,循环往复、直至所有非基变量检验数全部为负

终止。

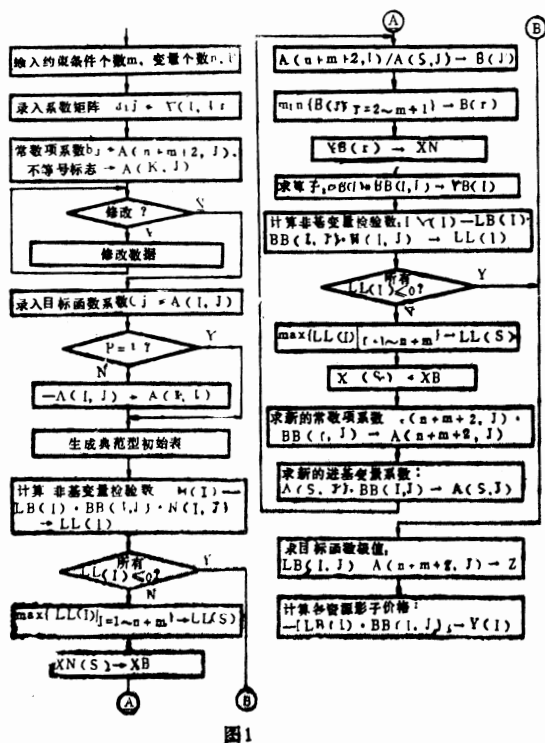
### 8. 计算目标函数值

当线性方程组有最优解时,目标函数值 $Z = C_B \cdot B^{-1} \cdot \bar{b}$ ,其中 $C_B$ 为基变量 $X_B$ 的目标函数的行向量, $B^{-1}$ 为基矩阵的逆矩阵, $\bar{b}$ 为新的常数项系数。

### 9. 计算资源的影子价格

有了最优解,按照计算影子价格的数学模型 $y = -(C_B \cdot B^{-1})$ ,求出各种资源的影子价格 $y_j$ ,并使之显示在屏幕上,为用户利用影子价格进行经营管理决策提供了科学的依据。

上述算法的工作流程如图1所示:



# 24×24点阵通用高级汉字打印驱动软件的设计与实现

郑州高炮学院

王磊

刘凤荣

**摘要** 本文介绍了研制通用高级汉字打印驱动软件的目的、设计思想,给出了系统的概貌,并详细地讨论了开发的技术途径和实现方法。

## 一、引言

汉字的打印输出是微机应用中一个不可忽视的问题。由于生产打印机的厂家很多,因而,打印机的型号繁杂,并且其打印控制字符也各不相同,导致开发相应的汉字打印驱动软件比较困难,甚至引起一些重复性开发,造成许多浪费。超大规模集成电路(VLSI)的迅速发展,使计算机升级、更新换代周期缩短,如何利用原有的硬件资源,节省用于实现打印机与新机系统适配所需的资金,就成为一个急待解决的课题,另外,目前每种型号的打印机实现汉字的打印输出,都对应着一个独立的汉字打印驱动软件,并

且这些驱动软件在某些特定的场合下,不能完全实现汉字的打印输出(如在西文DOS支持环境下就不能打印输出汉字),也不能通用,给使用、管理和维护带来不便。因而,开发一个能支持各种型号的打印机在任何环境下,都能实现汉字打印输出的驱动软件是十分必要的。

## 二、设计思想

目前,国内外生产的各种型号的24针打印机,按照结构特点可以划分为两大类:一类是不带硬字库的,如M2024、M1724、TH3070、紫金3070型等、另一类是自身带有硬字库的,如AR2463、OKI—3824型等。

- A. 确定哪种资源对提高经济效益更有利
- B. 如何确定新产品的价格
- C. 确定增加资源的合理性
- D. 产品价格变动对资源的影响
- E. 确定多少价钱购买总资源才合理
- F. 分析工艺过程改进后对资源节约的经济效益。

### 2. 系统结构

影子价格经营管理结构如图2。

A~F功能模块的工作流程不太复杂,这里不再给出。

本系统经多次调试运行,各项指标均已达到原设计要求。

### 参考文献

1. 《运筹学》清华大学出版社,《运筹学》试用教材

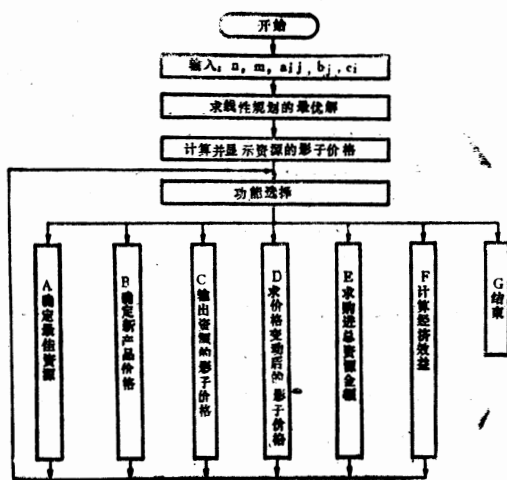


图2

编写组,1985年10月。

2. 《运筹学》,机械工业出版社,林同曾主编,1986年1月。

后者已可以实现在任何情况下的汉字打印输出,但由于经济上的原因,在相当长的一段时间内,不带硬字库的打印机将在实际工作中发挥其重要的作用。尽管它们的型号、打印控制字符不尽相同,相应的汉字打印驱动程序也不一样,但是,其实现汉字打印输出的工作原理是一致的,都是用新的打印驱动模块取代ROM—BIOS中的相应部分,完成汉字打印输出功能。

具体的方法是:

1.修改原中断17H的入口地址,使之指向新的中断服务模块;

2.将打印机的工作方式设置为“图形工作状态”(因为汉字的打印输出是以“图形传送方式”实现的)。

由于汉字打印驱动程序的工作原理是一致的,使得用一个通用的汉字打印驱动程序支持多种型号的打印机输出汉字成为可能。

目前,各种型号的24针打印机的汉字打印驱动程序,除了汉字打印控制命令不一致外,其它部分的功能完全相同(不同的设计者所设计的汉字打印驱动程序中各个功能模块的位置不一致),所以在该驱动软件中,将各个型号的打印机的汉字打印控制命令作为数据,集中存放在数据模块中,并且每一种型号打印机的控制命令部分都对应着一个指针,用以实现将程序中的浮动指针指向与其对应的打印控制命令的首地址,把相应的汉字打印控制字符调入到汉字打印驱动模块的位置上,动态地完成汉字打印驱动模块的生成。

### 三、系统概述

24×24点阵通用高级打印驱动软件是在带有长城汉卡的Super AT机上开发的。可以输出多种字体(宋体、仿宋体、黑体和楷体)和多种字型。

#### 1.硬件环境和系统软件支持环境

在该软件设计过程中,针对我国微机和打印机型号繁多的特点,主要以IBM PC/XT/AT;长城系列PC机;Super系列PC机;TUBRO系列PC机;紫金系列PC机;浪潮系列PC机和部分386机,及所有可运行IBM PC—DOS(或MS—DOS)的带有硬盘的PC兼容机作为主机支持环境,支持目前流行的M2024、M1724、M1570、NEC3824、NECP7、NECP5、P1351、TH3070、紫金3070、东芝3070、NM9400、LQ1000、LQ1500、LQ1000K、LQ1500K、LQ2500K、LQ3500K等多种型号的24针打印机。

本软件以2.0以上版本的PC—DOS(或MS—DOS、CC—DOS)作为系统软件支持环境,不论在哪一种DOS版本下工作,对于主机或打印机的硬件不需作任何改动,均可实现汉字的打印输出。

#### 2.系统结构

本软件主要由下述六个模块构成:

##### (1)命令行处理模块

该模块对用户键入的命令行进行分析和处理,确定是“自动响应”,还是“菜单提示”对话方式。如果命令行中设置了适当的打印机代号,则立即调用“生成汉字打印驱动模块”等,完成汉字的适配处理(这种方式适合在批处理或采用固定型号的打印机时使用);如果命令行中没有设置打印机型号或者其参数设置出错时,则自动转为菜单提示的“人一机对话”方式,提示用户选择适当的打印机型号,然后,对命令行中设置的内存常用汉字库的规模进行调整处理。

##### (2)菜单处理模块

调出“菜单”显示各种打印机型号供用户选择,等待用户键入选择值,用以生成相应的打印机汉字打印驱动程序。

##### (3)生成汉字打印驱动程序模块

生成相应型号的打印机汉字打印驱动程序的代码和数据。它包括两个子模块:

a 汉字打印控制字符集确定子模块:确



定出相应型号打印机的汉字打印控制字符集的首地址;

b. 覆盖子模块; 调度相应打印机的汉字打印控制字符集至汉字打印驱动程序中的汉字打印控制字符的位置上, 动态地完成汉字打印驱动程序的生成。

#### (4) 内存常用汉字库规模处理模块

对用户指定的内存常用汉字库的大小进行相应的处理, 必要时进行适当的调整。

#### (5) 确定驻存长度模块

对需要驻存的代码和数据总数进行精确的计算, 确定出常驻内存的字节数, 根据用户所选内存常用汉字库的大小, 其驻存的长度为34KB~83KB。

#### (6) 新中断服务模块17H

该模块完成汉字及字符的处理、打印工作。系统流程如图1所示。

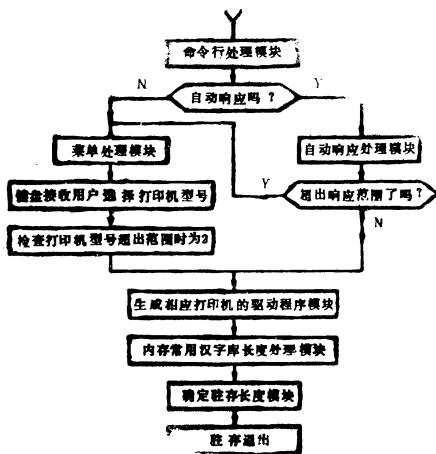


图1 总体流程图

## 四、系统实现

### 1. 设计方法与实现途径

优秀的通用高级汉字打印驱动软件必须能够最大限度的发挥打印机的效能, 应该具有良好的中英文兼容处理能力和操作简单的特点。为此, 我们用C语言开发了一个辅助开发工具软件FSFG·EXE, 利用它对目前流行的多种汉字打印驱动程序进行辅助分析,

总结各自的优缺点, 为制订通用软件提供标准。在对通用高级汉字打印驱动软件应具有的功能进行充分的分析和论证的基础上, 采用模块化结构, 使程序结构紧凑合理、短小精悍(程序长度仅为14478字节), 层次分明, 各模块功能相对独立, 调试方便, 具有良好的可扩展能力; 采用覆盖技术, 解决了汉字打印驱动程序的自动生成, 集目前流行的多种型号的24针打印机的汉字打印驱动程序功能及特点于一体, 使之具有良好的兼容能力; 采用“菜单技术”, 实现了交互方式, 使本软件具有良好的“人一机”界面, 操作简单, 在实现过程中, 选用C语言和汇编语言设计, 使本软件目标代码优化程度高, 运行速度快, 可靠性好。

### 2. 技术难点及解决方法

在本软件开发过程中, 存在的技术难点主要有以下两点:

a. 自动生成所选打印机型的汉字打印驱动程序;

b. 中、英文操作系统下的兼容能力。

(1) 汉字打印驱动模块的自动生成方法。

将各种型号的24针打印机的汉字打印控制字符集中存放在数据段地址DS: 1618处, 使用中根据用户所选的打印机型号, 软件自动将浮动指针指向与其相应的打印控制命令的首地址, 将相应的汉字打印控制字符调入到代码段地址CS: 037F的位置上, 动态地生成相应的汉字打印驱动模块。

(2) 各种DOS(西文或中文)支持下实现打印的工作原理。

a. 西文DOS(PC-DOS或MS-DOS)支持下实现汉字打印的工作原理。

西文DOS支持环境下, 打印机的工作状态通常被设置为“字符工作方式”, 并且原打印驱动模块不具备读取汉字字模的功能, 而汉字字模是以图形方式存贮的, 因而, 在西文DOS的支持下, 打印机不能正确地输出汉字。为了解决在此情况下的汉字打印输出问

题,我们在本软件的设计中,将打印机的工作方式固定设置为“图形工作状态”,并在程序中增加了读取汉字字模的相应功能,从而,实现了在西文DOS环境支持下的汉字打印输出。

b.中文DOS支持的西文工作方式下实现汉字打印的工作原理。

在中文DOS支持下,每次输出打印时,汉字打印驱动程序驻存的部分都要检测显示器的工作方式,并将打印机的工作方式设置成与显示器相同的工作方式。当显示器工作在中文(图形)工作方式时,主机向打印机发送的打印数据是欲输出汉字或字符的相应字模;如果显示器工作在西文(字符)工作方式时,主机发送的打印数据是欲输出内容的ASCⅡ码值。因而,在中文DOS支持的西文工作状态下,打印机也不能输出汉字。解决这一问题的方法同西文DOS环境支持下的汉字打印工作原理一样,将打印机的工作方式设置为“图形工作状态”,并且主机每次向打印机发送打印数据时,不检测显示器的工作方式,所以,主机每次向打印机发送的打印数据均是欲输出汉字或字符的相应字模,从而,实现了中文DOS支持的西文工作方式下的汉字输出打印。

另外,带有长城汉卡的微机,其汉卡主要用于高分辨率的屏幕汉字显示,打印输出汉字时,仍将显示器的工作状态转换为中分辨率的图形工作方式,通过调用软字库完成汉字的打印输出,所以,带有长城汉卡或EGA卡的微机工作在高分辨率时,也不能正确地打印输出汉字。由于本软件设计中采用脱离显示器工作状态的工作方式,所以,在此情况下使用,打印汉字将不受长城汉卡或EGA卡的影响,在任何DOS环境支持下,均可以打印输出令人满意的汉字文本。

## 五、系统特点

该软件具有以下七个特点:

### 1.适用范围广:

可适用于所有能够运行IBM PC—DOS(或MS—DOS)的带硬盘的PC机(包括部分386机),支持目前流行的各种型号的24针打印机的汉字打印输出。

2.操作简单、易学易用:该软件有“自动响应”和“菜单提示”两种工作方式,其汉字打印控制字符与流行的CC—DOS的汉字打印驱动程序兼容,原打印文件中的控制字符不需改动即可在该软件支持下使用。

3.功能齐全:该软件支持四种字体:宋体、仿宋体、黑体和楷体的打印输出,并且每种字体都对应着四种字型。

4.占用内存小:该软件程序长度14478字节,在驻存处理后,只占34KB至83kB(根据用户所选常用内存汉字库的大小而定),可节省汉字库所占的内存空间。对于需要大内存支持的系统打印输出汉字时极为重要,可以在西文DOS支持下,结合本软件使用,从而克服在CC—DOS环境下内存不够的问题。

5.兼容能力强:欲打印输出的内容只要是可打印的,均可准确无误地打印出来。另外,为了满足用户的一些特殊要求,提供了一个辅助程序,既可暂时取消汉字打印功能,将打印机置成纯西文打印方式,又可恢复汉字打印功能,使中、英文打印方式并存。

6.具有独到之处:在各种DOS环境支持下,均能打印输出汉字,解决了西文DOS(PC—DOS或MS—DOS)环境下和中文DOS环境下西文工作方式时(“字符工作方式”)的汉字打印输出,并且支持带有长城汉卡或带有EGA卡的微机工作在高分辨率时的汉字打印输出。

7.适应能力强:由于该软件采用脱离硬件环境的工作方式,所以,它在更新换代迅速的今天显得尤为有用,它能弥补由于计算机更新换代而软件跟不上的缺陷,并能节省实现打印机与新机系统适配所需的资金,有效地利用原有的硬件资源。

# 扩充C语言的矩阵运算功能

浙江大学 陈德谦

**摘要** 本文介绍一种扩充C语言矩阵运算功能的方法,其特点是利用编码/解码的方法传递矩阵运算所必须有的规模大小等附属信息,从而简化了应用程序的编写。

## 一、引言

众所周知,以矩阵理论为基础的现代控制理论应用于实际的控制系统中,一般需要作大量的矩阵运算。普通的算法语言如C、PASCAL、FORTRAN等都没有最基本的矩阵加、减、乘和求逆等运算功能。这就增加了编写程序的工作量和程序执行时间。因此,在普通的算法语言基础上,如能扩充矩阵运算功能是很有意义的。

本文所介绍的方法通过改变C语言的库文件在适合于自动控制用的C语言上扩充矩阵运算功能,从而简化程序的编制。此方法

已应用于实际控制系统,实践表明这是正确可行高效率的方法。此外,本方法还可以推广到其他功能的扩充。

## 二、编码

编码是通过语句redimension(矩阵名,行数、列数)来实现,它相当于FORTRAN语言中的dimension和BASIC语言中的DIM语句,所不同的是此语句可以根据需要对同一个矩阵多次说明为不同规模的矩阵。例如,先将矩阵A说明为5行 $\times$ 2列,根据需要,再将A增到5行 $\times$ 3列,同时矩阵内原来的数据不变。

## 六、结束语

本软件在目前流行的几十种型号的24针打印机运行通过,在我院有关科室的应用中反应良好,并于1988年7月1日在郑州通过了由河南省计划经济委员会和河南省机械电子工业厅组织的省级鉴定,与会的十几位专

家、教授们认为:该软件设计思想合理、程序结构清晰,时空性能良好,可以配置到多种微机上使用,支持多种型号的24针打印机,在各种DOS环境支持下运行,并配用高、中、低分辨率的显示器,给用户带来了很大的方便,实用、经济性好。

### 参考文献

贺志强等译,DOS磁盘操作系统高级程序员指南,中科院计算所,1987.



程序可写为:

```

:
redimension(A,5,2);
:
redimension(A,5,3);
:

```

下面介绍实现上述语句的redimension模块的内部结构。

1. 内存分配。向量矩阵存放一维4字节浮点变量数组内,非向量矩阵存放二维4字节浮点变量数组内。数组的第一个元素A[0]或A[0][0]用于存放整数类型的编码。用户使用的数组下标从1开始(A[1]...或A[1][1]... )。

编码  $code = B_0 + B_1 \times 256 + B_2 \times (256)^2 + B_3 \times (256)^3$

其中:

B<sub>0</sub>表示矩阵的类型,非向量矩阵为0,

行向量为1,列向量为2。

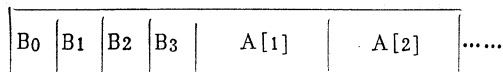
B<sub>1</sub>表示向量的元素数目。

B<sub>2</sub>表示非向量矩阵的行数,取值为0~255。

B<sub>3</sub>表示非向量矩阵的列数,取值为0~255。

为了提高运行速度,在实际应用时,可将B<sub>0</sub>直接放入数组的头一个字节,B<sub>1</sub>放入第二个字节,其他类推。这种方法在C语言里很容易实现。编码后的内存分配如图1所示。

一维数组:



二维数组:

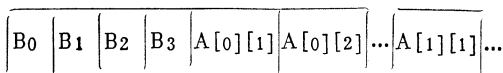


图1编码后的内存分配

/\*.....\*/

2. 编码程序:

```

redimension(a,b,C) /* re-dimension an array a[b]([C]) */
Char *a; int b,c;
{if (C==1) {a[0]=2;a[1]=(Char)b;} /* Column vector a[b] */
else if (b==1) {a[0]=1;a[1]=(Char)C;} /* row vector a[c] */
else {a[0]=0;a[2]=(Char)b;a[3]=(Char)C;} /* matrix a[b][c] */
return;
} /* redimension */

```

/\*.....\*/

### 三、解码

由于在运行时矩阵运算模块自动解码,因此用户不必考虑解码问题。例如,A为5行×5列矩阵,B为5维列向量,其乘积矩阵C为5维列向量,即 $C_{5 \times 1} = A_{5 \times 5} \times B_{5 \times 1}$ 。

用户程序可为:

```

:
float A[10],B[10],C[10]; /* 数组开大一些可有较大的灵活性 */
:
redimension(A,5,5);redimension(B,5,1);
/* 编码,结果矩阵C的编码自动进行 */

```

```

:
matrix(C,'\','=','A','\cdot',B); /* C5×1=A5×5×B5×1 */
matrix(C,'\','=','A','\cdot',C); /* C5×1=A×C=A2×B */
matrix(C,'\','=','A','\cdot',C); /* C=A3×B */
:

```

如果不采用编码/解码方法,则上述程序可写成:

```

:
float A[10],B[10],C[10];
:
matrix(C,5,1,'\','=','A,5,5,'\cdot',B,5,1);
matrix(C,5,1,'\','=','A,5,5,'\cdot',C,5,1);
matrix(C,5,1,'\','=','A,5,5,'\cdot',C,5,1);
:

```

由此可见其编程较复杂,不仅容易出错误,而且过多的

数据传递降低了运行速度。

解码模块是由6个内部模块组成,它们是:

is vector1(A)——判断A是向量还是非向量。

is rowvec1(A)——判断A是否行向量。

is colVec1(A)——判断A是否列向量。

getsize1(A)——取向量A的维数(元素数目)。

getrow2(A)——取非向量矩阵A的行数

getcol2(A)——取非向量矩阵A的列数

解码程序如下:

```
/*.....*/
isvector1(a) /*true if a[ ](a[ ][ ]) is a vector */
Char *a;
{ return((int)(*a));}
/*.....*/
isrowvec1(a) /*true if a[ ](a[ ][ ]) is a row vector */
Char *a;
{ return((int)(*a)==1));}
/*.....*/
iscolvec1(a) /*true if a[ ](a[ ][ ]) is a column vector */
Char *a;
{ return((int)(*a)==2));}
/*.....*/
getsize1(a) /*get the size of a[. ]*/
Char *a;
{ return((int)a[1]);}
/*.....*/
getrow2(a) /*get the row size of a[ ][ ] */
char *a;
{ return((int)a[2]);}
/*.....*/
getcol2(a) /*get the column size of a[ ][ ] */
char *a;
{return((int)a[3]);}
/*.....*/
loadtype1(a,b)/*load the type and size of a[ ] from b[ ] */
short *a,*b;
{(*a)=(*b);return;}
/*.....*/
loadtype2(a,b)/*load the type and size of a[ ][ ] from b[ ][ ] */
long *a,*b;;
{ (*a)=(*b);return;}
*.....*/
```

上述运算是在最前面的4个字节中取出相应的编码信息,然后将数据转换成整型量后返回。

## 四、矩阵运算模块

由于篇幅有限,这里仅介绍矩阵加法运算模块。

### 1. matrix模块结构

matrix为开关选择模块,其功能是根据矩阵运算的种类分别转到相应的子模块中去,框图如图2所示。

### 2. 矩阵加法子模块

矩阵加法子模块是由三个部分组成,从matrix传过来的参数由matadd进行分析,

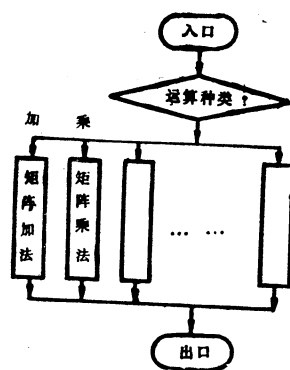


图2 matrix结构框图

判断是向量还是非向量。如果是向量,则由add1进行向量的加法运算,反之由add2进行非向量矩阵的加法运算。其相应的程序为:

```
/*.....*/
matadd(a,b,c) /*a=b+c*/
float *a,*b,*c;
{ if(isvector1(b)) add1(a,b,c);else add2(a,d,c);return;
}/*matadd*/

/*.....*/
add1(a,b,c) /*a[ ]=b[ ]+c[ ]*/
float a[ ], b[ ], c[ ];
{ register int i, size;size=getsize1(b);for(i=1,i<=size;i++) a[i]=b[i]+c[i];
Loadtype1(a,b);return;
} /*add1*/

/*.....*/
add2(a,b,c) /*a[ ][ ]=b[ ][ ]+c[ ][ ]*/
float a[MSZ0][MSZ0],b[MSZ0][MSZ0],c[MSZ0][MSZ0];
{ register int i,j;int row,col;row=getrow2(b);col=getcol2(b);
for(i=1;i<row;i++) for(j=1;j<=col;j++) a[i][j]=b[i][j]+c[i][j];
loadtype2(a,b);return;
} /*add2*/

/*.....*/
```

## 五、库文件的修改

现以IBM-PC上的Microsoft C 4.0

为例介绍库文件的修改方法,其他机器的C语言库文件的修改与此类似。

# 提高VAX Rdb/VMS 数据库存取 速度的探讨与实现

珠海拱北海关技术处 王国星

**摘 要** 本文简单介绍了VAX Rdb/VMS数据库管理系统的一般优化技术,并着重分析介绍了在多用户联机大批量数据录入环境下,通过改进使用方法,建立专用服务器,使得速度和效率取得突破性提高的原理和方法。

## 一、概 述

VAX Rdb/VMS是美国DEC公司推出的在其VAX系列计算机VMS操作系统下运行的关系数据库管理系统,它具有关系数据库管理系统的优点;更有它自己的特点,不仅能使用ANSI标准SQL语言来定义、操纵和管理,并且可用DEC公司的数据操纵语言RDO (Relation Database Operator)来管理数据库。同时,为DEC的信息体系结构软件如CD D、DTR、RALLY、TEAMDATA和高级语言如COBOL、PASCAL、C、FORTRAN等提供广泛的接口,通过DECnet网络可存取远程节点的数据库,使用比较方便。但总的给人印象是访问数据库的响应时间不甚理想,尤其当多用户同时对数据库作插入、修改或删除操作显示更为突出,甚至到无法忍受的程度。

本文试述对Rdb/VMS的一般优化技术,并介绍一种通过使用服务器的技术来解决当大量联机用户同时对数据库作插入、修改、删除操作时存取性能问题。

## 二、一般优化技术

使用VAX Rdb/VMS(以下简称Rdb)时,影响Rdb存取性能的因素有关系的范式规范化、索引的使用、磁盘I/O操作、锁管理、数据压缩及数据库和VMS参数的调整等,其中,主要是I/O限制和锁管理。设计时从以下几方面考虑数据库的优化:

### 1. 增加必要的数据库冗余

一般来说,数据库的逻辑设计都要按照关系范式的要求规范化,以减少数据的冗余度和保证数据的一致性。但严格地规范化要增加关系和建立索引,使用中也要频繁地链

1. 找出小内存模式库文件SLIBC、LIB (如果是其他内存模式,则找出相应的库文件),并做好备份。

2. 将编码、解码、矩阵运算等模块编译成二进制文件,并命名为mat、obj。

3. 将有库管理程序LIB、EXE的软盘放入B:驱动器,有SLIBC、LIB和mat、obj的盘在无写保护的情况下放入A:驱动器。

4. 在DOS下打入 B:LIB→A:SLIBC→A: mat↵。

5. 将修改好的新SLIBC、LIB拷贝到

库目录区替换老的SLIBC·LIB。

完成上述步骤后,C语言就具有矩阵运算功能。如果想去掉矩阵运算功能,只需换用原来的库文件就可。其他功能的扩充与此类似。

### 参考文献

1. Microsoft Corporation, "Microsoft C Compiler—User's Guide", 1986.
2. Microsoft Corporation, "Microsoft C Compiler—C Language Reference Manual", 1986
3. 陈德谦, "椒江湖沙模型流量边界的非线性自校正控制" 浙江大学硕士学位论文, 1988.

接几个关系,不但增加了系统的复杂性,同时也增加I/O 操作和存贮空间的开销(增建索引所需),而使数据库性能下降。所以,增加必要的冗余可明显提高系统的性能。

## 2. 合理设计索引

在数据库的物理设计阶段,对关系中的某些字段建立索引对查询是极有意义的。在数据量大的情况下,对经常使用的查询条件一定要建立索引,以保证查询速度。由于在Rdb 中,索引是B树结构,在对常数型的参数库(或关系),及存放历史数据的数据库(或关系),访问以查询为主,对主要查询条件必须建立索引。同时,尽可能把索引节点定义大些,增大充填系数,以减少节点层次,减少I/O 次数。提高响应速度。而对那些频繁进行修改的关系的索引节点尽量少些,以减少锁的范围。设计索引时应避免重复关键字值的出现,过多的重复关键字值会严重影响系统的性能,这时,可把几个字段组合成一个关键字或加一随机数,以减少重复关键字值的出现。

## 3. 减少磁盘I/O操作

在一般的事务处理中,磁盘I/O次数对系统的影响都较为突出,VAX机也不例外,可以说,VAX机的CPU是一流的,但其I/O系统技术较落后,如发现I/O操作是系统的“瓶颈”,则可通过加大虚拟存贮器的尺寸(逻辑名为RDMS $\neq$  BIND\_WORK\_VM),把数据库的.RDB文件、.SNP文件、RUJ文件、AIJ文件及工作表文件设置在不同的盘上,以平衡各磁盘I/O操作。查询时以索引为检索条件也可大大减少I/O操作。

## 4. 加锁管理

数据库管理系统的主要特征之一是具有确保数据一致性的能力。为避免多用户并行环境下错误地修改数据,Rdb 使用VMS分

布锁管理系统同步被共享资源的访问,当用户访问记录时,Rdb使用一系列锁,防止其他用户同时选择相同记录。这些锁加到记录上,也可能加到数据项、索引节点、关系,甚至整个数据库级。用户可以在START\_TRANSACTION 中使用访问方式控制资源加锁,来影响数据库活动的并行性及资源共享程度。在Rdb中,主要有三种加锁方式:共享、保护及互斥。当单用户访问数据库时,使用互斥方式,好处是减少资源加锁,同时不写SNAPSHOT文件,系统性能得到提高;缺点是其他用户不能并行访问相关关系。在需要共享的访问中,如果是只读访问,使用READ\_ONLY,这种方式加锁最少。如是修改操作,尽可能使用保护写方式,保护写只允许一个用户作修改操作,但不影响只读事物对相关关系的访问。当多个用户修改相同关系的数据时,就选用共享写方式,它允许多个用户同时作修改操作,但也增加了锁的竞争,可能还会出现锁冲突或死锁现象。一旦发生死锁,Rdb要强制一个用户ROLLBACK,应用必须重做那个操作。另外,由于需长时间等待冲突锁的被释放,将增大事务的轮转时间,势必影响系统性能。因此,在实际应用中,对只读事务,用READ\_ONLY方式,对修改事务,最好用互斥方式,尽可能使用保护方式,尽量少用共享写方式。

## 5. 调整VMS和Rdb参数

参数调谐分二部分,VMS参数和Rdb参数。在多用户环境下,VMS参数PAGE-FILE、SWAPFILE、VIRTUALPAGE-CNT、MAXBUF等应增大,DEADLOCK\_WAIT要减小,Rdb 的页面大小应考虑配合记录长度、开设较大的缓冲区、预分配足够大的空间等都有利于提高响应速度。



## 6. 查询优化和数据压缩

利用索引检索数据是一种有效的查询方式,当记录选择表达式(RSE),涉及多个关系和索引时,怎样以最佳路径去寻找数据,提高查询性能,这是普遍关心的。Rdb提供的优化器可帮助你寻找最佳访问路径。在一组记录选择表达式中只要有一条能利用索引表,优化器就尽力去利用它,以避免在实体数据中顺序查找。因此,在实际应用中,应尽量把记录选择条件和某个索引字段拉上关系,尽管看上去是多余的判断条件,实际上是为了让优化器能利用其上索引,以达到优化目的。

当某些关系的数据不是频繁地被修改,我们可以使用数据压缩技术,这样,不但可节省存储空间开销,同时,在许多情况下也可提高检索性能,尤其是顺序读时一次磁盘I/O操作可把更多的数据读入用户缓冲区。对于频繁要修改的数据,尽量不使用压缩技术,原因是作修改操作时不仅需化较多的CPU时间,同时,如果关系内发生页分裂,将把一个记录存放在多个数据页中,大大增加系统开销,也影响读操作。

## 三、使用专用服务器

### 1. 为什么要使用服务器

通常,设计一个并不十分大的数据库时,通过上述优化,可明显提高数据库的性能,尤其操作主要是查询时,响应时间可满足作用要求。但是,当大量联机用户并发对数据库执行插入、修改或删除操作时,其性能是不够理想的。海关系统在搞“H883”软件工程时就遇上这个问题,为此,成立一个专门技术小组,用原型法等多种方法对Rdb数据库进行了长达三个多月的测试,测试结果如表1。

表 1

终端用户情况	操作内容	平均化费时间	平均磁盘I-O数
单用户 录入	每存(STORE) 一张单	0.9秒	30次
单用户 查询	按主键每读一张单	0.27秒	5次
50用户同 时录入	每存一张单	10秒	45次
50用户同 时查询	按主键每读一张单	0.3秒	5次

测试环境为: VAX8530, 16MB内存, SA482磁盘组, VAX/VMS V4.7, VAX Rdb/VMS V2.3, 共享方式为保护写和共享读,每个事务处理一个单子;每张单分表头和表末二个关系;长度分别为320和250个字节;采用数据压缩技术后实际长度减少三分之一左右;每个关系均有一个具有唯一值的主键;.RDB、.RUJ及.SNP文件分开磁盘存放;每个关系中都有100万个记录,数据库文件(RDB文件及SNP文件)为900MB。从测试表上看,单用户读和多用户读数据库的差别不大,但多用户并发存数据时其响应时间是单用户存数据的十多倍。其原因是相互间等待锁的时间太长,这时,数据库锁的管理相当复杂,要提高响应时间唯一的办法是减少执行存数据的用户数。对于那些非实时性的应用环境,可先把数据存入一临时文件,晚上再发一批作业处理存数据操作。但对那些实时应用环境这种方式就行不通,用户数又不能减少。海关技术小组通过多次测试和探索,采用服务器方式,解决了多用户并发存数据时的性能问题。即创建一服务器进程,用户进程把数据传送给服务器进程,由服务器完成对数据库的插入数据操作。其原理如图1所示。

## 2. 建立服务器方法

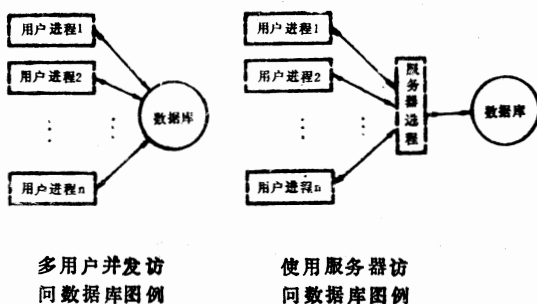


图 1

使用服务器后,把多用户并发直接存取数据库操作变为单用户执行直接存取数据库操作,使得复杂的锁管理简单化,响应时间大大提高。在相同环境下利用这一原理得到的测试结果如表2。

表 2

终端用 户情况	操作内容	平均化 费时间	平均磁 盘I-O数
50用户同时录入	每存一张单	0.2秒	5次

上述执行结果是服务器进程所致,对用户进程来说,由于执行的仅是在内存中的数据传送操作,其响应时间大大少于上述测试结果,也没有磁盘I/O操作。这里,实际上是进程与进程之间的通讯问题,熟悉VMS的人知道,在VMS下,可有多种方式实现进程之间通讯。当进程之间通讯时,仅是传送小量信息或仅解决相互之间同步问题,是比较方便的。当需要传送大量数据,又是在许多用户进程与服务器进程进行,其编程也是相当复杂的。这样,虽然解决了响应时间问题,但却大大增加了编程的难度和复杂性。能否找出一种简单的解决方法呢?答案是肯定的。解决办法是建立一个队列,用户进程把数据写入队列,由服务器进程从队列中读出数据,再执行写数据库操作,之后,把执行结果反馈给用户进程。其原理如图2所示:

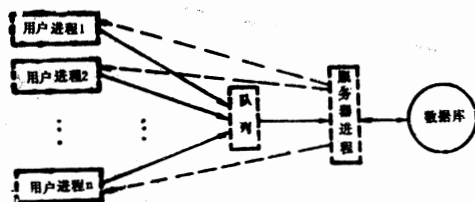


图 2

在VMS下,有多种方式建立队列,如可用RMS文件、全程节等。为提高存取速度,该队列应建在内存中。

## 3. 使用服务器的优点

使用服务器的主要特点是解决了大量联机用户并发访问数据库的响应时间问题,一般情况下可提高三十倍以上。同时,由于减少了直接访问数据库的进程数,也相应减少了对主机系统资源的需求。

## 四、结束语

使用关系数据库管理系统为应用设计与开发带来不少方便,但一般占用主机系统的资源较大,运行的响应时间较慢,Rdb过去在这方面反映的问题也较多。但由于Rdb是VMS操作系统的标准的层次产品,在发展和利用VMS系统服务改进操作效率方面的余地也较大,海关系统这次在实现“H883”工程中通过使用专用服务器提高存取性能就是一个很好的例子。虽然今后最终解决关系型数据库的开销和效率问题的方案可能是发展专用的、软硬件一体化的数据库机,但在目前条件下,通过对使用技术的研究和探讨,找出一种解决办法,使Rdb基本胜任已经在一般情况下被认为比较难办的大容量数据录入环境,这是很有意义的。本文提出的方法及原理带有一定的通用性,供同行们参考。

# 在APPLE II机上利用128K卡建立扩展

## 数组区BASIC解译系统

甘肃省地矿局 尚银生

**摘 要** 本文介绍在APPLE机APPLESOFT解译系统下,开发的、对128K卡的、软件管理模块。使原来只能作外存用的128K卡在该软件模块支持下作内存使用,使主机内存从64K真正扩展到了192K,以实现在APPLE机上运行较大程序的要求。

### 前 言

APPLE 机由于价格低廉,功能齐全,在国内微机行列中占有很大数量,但它的内存较小,只有64K,不能运行较大的程序,限制了这类机子的使用范围。为了解决它内存较小的问题,我们深入地分析了APPLESOFT解译系统的工作流程和它的数组处理方式,笔者在APPLE II E机上,利用插在主机槽口上的128K扩展卡,在APPLESOFT解译系统下开发了在128K卡上定义数组、访问数组的软件管理模块。

在该模块支持下,APPLESOFT 解译系统对 BASIC 程序区的管理范围从35.5K的内存扩展到了 163.5K。此时解译系统在对BASIC 程序解释执行过程中,128K卡不再是不能被随机访问的辅助存贮区,而是和主机内存存在地址访问上连成一体、由解译系统统一管理的主存。从系统上而言呈现在用户面前的是扩展数组区APPLESOFT 解译系统。

下面从四个方面进行论述:

### 一、问题的提出

1. 插有128K卡的APPLE II E机,在APPLESOFT解译系统下,运行BASIC 程序,主机内存不够用时,解译系统在程序数组处理中,不能将内存放不下的数组定义在128K卡上,或在该卡上访问数组,128K卡只

能作外存使用。若要在该卡上存取信息,在BASIC程序中要用CALL语句调用,“用户用 6502机器语言编写的子程序”,由子程序把内存和128K卡上的信息进行交换,BASIC程序再用“POKE”或“PEEK”语句在内存中存取交换的信息,即解译系统不能管理128K卡,主机内存没有真正得到扩充,降低了128K卡的应用价值。这是该卡在原系统下,使用上存在的问题。

2. 若把128K卡当成存贮程序数组的内存使用,就必须修改原解译系统定义数组,访问数组的方式,在内存中建立“在128K卡上定义数组、访问数组的软件管理模块”,使解译系统在程序数组处理上直接调用软件模块,由模块对用户程序中的数组统一处理,把内存中放不下的数组定义到128K卡上,把整个128K卡作为程序数组的扩展区,由解译系统统一管理。

## 二、扩展数组区BASIC解译系统的建立

### 1. 原系统浅析

APPLESOFT解译系统从地址\$D43C进入解译状态。首先在屏幕上输出一个“J”提示符,等待从键盘或磁盘输入数据。在解译过程中APPLESOFT 解译系统采用了(1)对输入数据缓冲区(\$0200—\$02FF)中的源程序进行预处理(剔出空格);(2)中间码代换;(3)编排整理;(4)解译执行的处

理方案, 工作流程可粗略描述为: 首先把键盘上输入的源程序语句的每个字符逐个存入输入数据缓冲区中, 以“RETURIN”键标志本行语句输入结束, 此时解译系统自动进入词法分析器, 作中间码代换。进入词法分析器分两种情况, 若是立即执行语句(或命令)从地址\$D456进入, 否则从\$D45C进入。通过词法分析器把预处理后的每个字符形式转换为以“单词”为基本单位的中间代码形式。此时对立即执行语句(或命令)则从\$D459进入解译执行状态并予以执行。否则将“单词”形式的中间代码进行编排整理存入\$0801开始的BASIC程序区, 待接到“RUN”命令时才开始对中间码程序进行解译执行。

中间码代换是在输入数据缓冲区中进行的, 编排整理是把缓冲区中的代码送到系统指位器(\$9B、\$9C)当前指出的程序区中, 这一过程说明解译系统在进入执行状态之前没有划分程序中的数组区, 数组区的划分是在执行时进行的。

这里对有关指位器和零页单元的作用给予简单说明: (\$6B、\$6C)数组区起始地址指针, (\$6F、\$7D)字符串数据起始地址指针, (\$81、\$82)变量存放单元, (\$9B、\$9C)通用指针。

## 2. 解译系统的修改

系统的修改指的是对解译系统处理数组的程序片断的有关转移指令的转移地址的修改。被修改的地址是: “\$E1FE、\$DA77、\$DA65、\$E1AA、\$E1CI、\$DABB、\$D—ED5、\$DACF、\$E15C、\$DA46、\$E046、\$E29F”。修改后使解译系统在执行状态下处理数组时, 这些地址上的“JMP, JSR”指令调用或转移到128K卡软件管理模块上, 由软件模块判断程序中的当前数组是在内存中处理, 还是在128K卡上处理。若是在内存中处理, 判断后则直接返回解译系统, 否则由软件模块完成数组在128K卡上处理的各项工作(包括定义数组、访问数组和对算术表达式中有关输入、输出操作的处理)。

## 3. 对数组的处理

地址\$DFD9是解译系统定义各种类型数组的入口地址, 系统定义数组要通过以下四个步骤: a. 从代码程中取出要定义数组的数组名作类型检查(整型、实型、字符串型), b. 作数组下标转换, c. 根据指位器(\$6B、\$6C)、(\$60、\$6E)的值确定数组表头在内存的起始地址并定义表头内容, d. 根据数组下标计算数组在内存的长度并将元素区冲零。

### (1) 数组定义

解译系统定义数组时, 软件模块从地址\$E1FE处进行拦截, 用JMP指令使解译系统转到管理128K卡的软件模块上, 测试被定义数组在内存的长度, “用(\$6F、\$70)指位器的值作主机内存的上限地址”(上限地址可用HIMEM语句设定), 检查后若数组区尾地址指针(\$6D、\$6E)的值小于上限地址, 软件模块则用JMP指令直接转到解译系统的\$E221处, 由解译器完成当前数组在内存的定义。否则由软件模块把当前数组定义到128K卡上, “该模块把数组表头定义在内存并在表头中增加该数组元素区在128K卡上的地址指针单元, 软件模块根据地址指针的指向用I/O控制选择开关, 打开128K卡上的一个或若干个存储模块, 用上述系统定义数组的方法, 完成数组在128K卡上的定义”。然后返回到解译系统的\$DFDD处, 去定义本行语句中的其它数组, 实现方法与定义前一个数组过程相同。当定义完本行语句中的所有数组后, 返回到系统的\$D823处去执行下行语句。

下图是定义在128K卡的数组表头结构(见图1)

### (2) 数组的访问

处理数组访问是指: 对数组变量的输入输出操作和有数组变量参与的算术表达式的处理。按系统处理数组访问的工作流程下面分两步论述:

1) 地址\$DFE3是系统查找各种变量的入口处, 解译系统在代码程序的数组区中,

根据(\$6B, \$6C)指位器指出的数组区起始地址, (\$81, \$82)中存放的数组变量名查找数组表头在内存的首地址。找到后在解译系统的\$E1AA处进行拦截, 用“JSR”指令使系统调用软件模块, 软件模块把数组区起始地址指位器(\$6B, \$6C)和当前数组首地址指针暂存单元(\$9B, \$9C)的值保存起来后再转到解译系统的\$EOED处。解译系统根据数组变量的下标值, 计算数组变量在数组中的起始地址并存放在A、Y寄存器中, 然后从解译系统的\$E2AC处返回到系统处理数组访问的子程序上。[对数组访问不同的操作, 从此处可返回到不同的子程序上把A、Y寄存器的值存放在(\$85, \$86)或(\$A0, \$A1)单元中]。

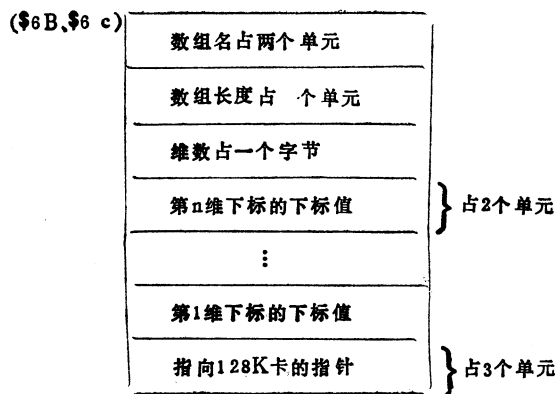


图1

2) 处理数组访问的子程序再调用软件模块, 模块根据暂存单元(\$9B, \$9C)的保存值, 检查数组表头中是否有指向128K卡的地址指针。若有, 软件模块根据地址指针的指向, 用I/O控制选择开关打开128K卡中存放本数组的存储模块, 同时软件模块对(\$85, \$86)或(\$A0, \$A1)单元中的数组变量地址进行地址转换, 计算出该数组变量在128K卡中的真实地址, 并用该地址在128K卡上实现对当前数组的访问, 然后返回到调用它的子程序上。若检查后没有指向128K卡的地址指针, 软件模块则直接返回调用它的子程序上, 由于程序在主机内存中处理对数组的访问。(在数组处理过程中, 修改后的解译系统和软件模块之间有20多处交叉调用或地址转移,

这里不予详述)。

#### 4. 兼容性

APPLE机的APPLESOFT解译系统是在6502状态下被用户广泛使用的一种BASIC版本, 具有较强的运算处理功能, 在修改解译系统时, 必须考虑它和原系统的兼容性, 不降低原系统的任何处理功能, 为此我们采用了软件模块接口改造方法, 对解译系统作地址拦截, 使128K卡软件管理模块在机内和解译系统连成一体。BASIC程序在进入执行状态时, 程序信息流在机内如图2所示:

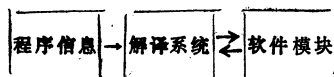


图2

BASIC程序在运行中, 软件模块只是对解译器处理与数组有关的程序信息时进行拦截并处理, 除此以外并没有参与解译系统的其它内部工作, 所以修改后的系统和原系统是完全兼容的。

### 三、技术处理和内存分配

1. 把原解译系统放到16K语言卡上运行, 为了能修改系统的地址, 实现系统拦截。

2. 在扩展数组区BASIC解译系统下, 运行BASIC程序时若要产生中断或者从监控状态下退出时, 要用“CTRL—C”键中断或退出。不要用“CTRL—RESET”键, 否则要在监控状态下, 触动\$C080单元, 用“CTRL—C”键重新进入新系统。

3. 新系统下内存分配结构图如图3所示。(英文状态)

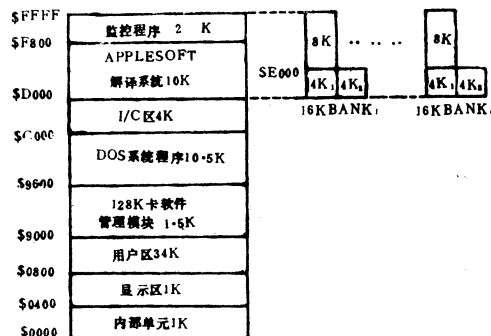


图3



# 小球病毒的分析 and 防治

天津五金矿产进出口公司

李建 李学泓

**摘要** 小球病毒是一种感染性极强的病毒,在被感染的系统下任何一种盘操作都可使健康盘受到感染。由于这个原因,在全国许多地方都发现了它的踪迹。本文从分析程序入手,向读者全面介绍病毒各部的功能,从而更清楚地了解它的传播机制。同时介绍了一种消毒方法,它的独特之处在于,它不是只对具体的盘进行操作,而是驻留于内存,对任何进入系统的盘进行检测,发现病毒立即清除。

国内许多的计算机感染了一种病毒。它的特征是,当计算机运行了一段时间后,监视器的屏幕上就会出现一个小球上下滚动。这里我们暂时称它为小球病毒。一般说来在西文状态下它不影响程序的正常运行,只是由于占用了机器时间,而使程序的运行速度变慢。但在中文状态下会使屏幕上下翻滚,显示混乱。从而使机器无法正常使用。这种情况发生后只能通过重新起动力可消除,这给使用者带来了极大的不便。尤其是它的感染性极强,一般的盘操作就会使磁盘受到感染,因而相对地说它的危害性就比较大了。

小球病毒是一种专门攻击磁盘boot区的病毒。它感染健康盘的时候,是将原引导块存入盘上的第一空扇区,然后用它自身的一部分取代原引导块。这样就为病毒进入内存和激活提供了条件,本文就是通过对小球病毒的剖析,了解它的实际工作过程,在根本上杜绝这类病毒的传播。

## 一、病毒的组成

从功能上讲小球病毒可分为三大部分。

### 1. 导引部分

它主要完成将病毒体装入内存、修改内存量指数、将INT13处理入口指向病毒体。并将原引导块调入内存完成系统自举。这一部分也可说是小球病毒的触发部。我们知道病毒进入内存后,如果不能被激活(即被执行),那么它充其量不过是占用了几K内存,不会给系统带来什么更坏的影响。只有当病毒被装入内存,而在一定的条件下又能被激活,才会有意义。这段程序将INT13的入口改为了病毒的入口,这实质上就完成了病毒的激活准备工作。因为每一次INT13的调用都会使病毒活动起来。

### 2. 传播部分

传播部分的功能就是当病毒被激活后,

## 四、扩展数组区BASIC解译

### 系统功能介绍

1. 中英文状态下皆能在128k卡上定义并访问整型、实型、字符型数组。

2. 主机用户区从35.5k扩展到163.5k(程序数组区扩展128k),是原用户区的四倍多。

3. 在内存空间许可下,程序中定义数组个数不受限制,一个数组最大为64k。

4. APPLESOFT BASIC程序的编辑,运行无任何条件限制,不降原系统任何处理功能。

5. 在插有16k语言卡和配有128k卡的APPLE系列机上均能运行扩展数组区BASIC解译系统。

(128k卡软件管理模块和自动连接APPLESOFT解译系统与128k卡软件模块程序清单略。)

它马上对被操作盘进行诊断, 检查磁盘是否被感染。如果磁盘没有被感染, 它就会修改导引区, 复制病毒体, 从而完成病毒的自身分裂。完成自身分裂对于病毒可以说是最重要的特性之一。因为一个病毒不能传播, 那它产生的影响就非常的局限。只有当它能够不断地分裂, 才会有比较大的危害性。某些感染性很强的病毒, 能在很短的时间内使某一范围的微机都受到感染。小球病毒就属于这种类型。

### 3. 作用部分

这部分又可称为破坏部分。它的作用就是对时间进行判别, 当达到某一时刻, 它开始侵入INT8, 随着每一次INT8中断的产生就在屏幕上显示一小球。因为INT8每秒产18.2次, 这样我们看到的小球就在不停地移动, 它不仅使得显示趋于混乱, 并且由于占用了大量的机器时间而使程序运行的速度变慢。

## 二、病毒的分析

不同的病毒具有不同的破坏性, 它们的表现形式也不尽相同。我们从防治的角度出发, 不过多地涉及作用部分, 主要对导引和传播部分进行分析。通过对程序的解剖, 了解它的实际工作过程, 达到消除病毒的目的。至于作用部分, 病毒被消灭了, 危害当然不复存在。

### 1. 导引部分

这一部分相对说来比较简单, 它存放于磁片的0道1扇, 即原系统的导引块部位。当系统进行自举时, 它就进入内存, 完成病毒的准备工作。在debug状态下可以用L命令将其调入内存。假设被感染的盘在A驱动器, 并且由于病毒进入内存后被装到最高段的7C00处, 为了分析方便也将其调入7C00。调入并显示。

输入

-L 7C00 0 0~1

-U 7C00 7C25

08FF,7C00 EB1C JMP 7C1E

08FF,7C02 90 NOP

08FF,7C03 49 DEC CX

```
08FF,7C04 42 INC DX
08FF,7C05 4D DEC BP
08FF,7C06 2020 AND [BX+SI],AH
08FF,7C08 332E3300 XOR BP,[0033]
08FF,7C0C 0202 ADD AL,[BP+SI]
08FF,7C0E 0100 ADD [BX+SI],AX
08FF,7C10 027000 ADD DH,[BX+SI+00]
08FF,7C13 D002 ROL BYTE PTR [BP+SI],1
08FF,7C15 FD STD
08FF,7C16 0200 ADD AL,[BX+SI]
08FF,7C18 0900 OR [BX+SI],AX
08FF,7C1A 0200 ADD AL,[BX+SI]
08FF,7C1C 0000 ADD [BX+SI],AL
08FF,7C1E 33C0 XOR AX,AX
08FF,7C20 8ED0 MOV SS,AX
08FF,7C22 BC007C MOV SP,7C00
08FF,7C25 8ED8 MOV DS,AX
```

为了减少篇幅, 这里只显示前几条指令。在分析程序之前将程序中用到的参数、单元盘基本输入输出表作一下简单的介绍。

0:413 系统保存的以K字节为单位的内存量。

7DF7 病毒使用的标志单元, 其中低三位有效。0位: 病毒运行标志, 限制病毒的递归调用。1位: 病毒破坏开始标志。2位: 软硬盘处理标志。

7DF8 最近一次盘操作的驱动器号。

7DF9 病毒体的相对扇区号。

7DFC 病毒标志; 内容为1357。

7EB0 上次盘操作的时间。

盘基本输入输出表起始地址7C02。

程序分析如下: 第一条指令是Jmp 7C1E 它的作用是跳过盘基本参数区。

7C1E~7C2D 病毒为了使它占用的2K内存脱离系统控制, 而将内存总量减2K。

7C30~7C37 根据内存总量计算最高段址。

7C39~7C46 病毒移至高段, 为系统自举准备工作区。

7C43~7C5E 读入病毒体。

7C61~7C6B 读入原系统导引块。

7C73~7C82 修改INT13的中断向量, 为病毒激活作准备。



# 关于在微机上将3维图形运用于工程CAD 的几个问题的认识和处理

沈阳工业大学 黄有群 朱世铁 王巨森 张大坤

**摘要** 在实际的微机CAD工程项目中使用3维图形,受到各种因素的制约,主要体现在成本、速度、内存和精度上。笔者在普通配置的微机上开发了一个实用的CAD应用软件,从分析3维图形在微机CAD系统中的环境和作用出发,较好地解决了这些问题,提高了CAD系统的性能。本文结合笔者的做法,介绍了对这些问题的认识和处理。

病毒立即清除。

具体方法如下:该方法基于小球病毒之上,因此要准备一张被感染盘。在debug下将病毒体调入内存,修改如下。

-L 7E00 05C 1

这里假设被感染盘在A驱动器。5C是病毒体的相对扇区号,它随盘上的内容不同而变化,这个数可从0道一扇1F9处查到。

-A 7E42

08FF,7E42 OR DH,F0,

08FF,7E45 CMP DX,FFF7,

08FF,7E48 NOP

7E45改为查找坏扇区的标志FF7

-A 7E5B

08FF,7E5B OR DH,F0

08FF,7E5E

7E5B将DH寄存器的高4位置为全1

-A 7E51

08FF,7E51 MOV DX,0

08FF,7E54

7E51原置坏标志改为置空标志

-A 7E66

08FF,7E66 JMP 7E93

08FF,7E68 AND [BX+8000],DX

08FF,7E6C

7E68清除FAT表中的扇区坏标志

-A 7E86

08FF,7E86 MOV BX,SI

08FF,7E88 INC BX

08FF,7E89 CALL 7C9D

08FF,7E8C MOV BX,0

08FF,7E8F CALL 7C98

08FF,7E92 RET

08FF,7E93 SHL DX,CL

08FF,7E95 OR DX,F

08FF,7E99 JMP 7E68

08FF,7E9B

7E86~7E92将原引导块读并装回0道一

扇。

到这里整个修改过程基本完成,但还需将判别部分修改一下。因为病毒程序首先判别被操作盘是否被感染,对没被感染的盘才进行操作。

这里应改为对被感染的盘进行盘操作。这只要将7D7C处的JNZ 7D8B改为JZ 7D8B就可以了。最后用命令

W 7C00 0 0 1 与

W 7E00 0 5C 1存盘。

有一点需要注意,如果你的软盘在被感染之前就存在着坏扇区,使用这种消毒方法则会引起混乱。这是因为程序寻找病毒体是以第一坏扇区为准。只要使用病毒保存的相对扇区(这一扇区号存放在0道一扇的1F9单元)计算出簇号,而不用上面使用的查找坏扇区法就可以了。这里不在赘述。

## 一、环境和目的

绝大多数实际的工程 CAD 应用系统是面向特定问题的。其设计方案的拟定,设计参数的选择,计算和优化的方法,评估和仿真的手段,都是针对具体的设计对象和目标而确定的;所要显示的3维图形,也仅要体现该类对象的特定的几何和结构特点,或者某些特定的内容。因此,就某个具体的CAD项目而言,并不需要3维图形的生成手段有通用性。

另一方面,除了外形设计之外,工程CAD中使用3维图形主要是为了使设计决策和数据处理、计算的结果“形象化”,加快和方便对设计效果的理解,提供良好的人机交互手段。至于图形的美观和逼真,常常不是主要的目标。

## 二、成本

对CAD用户来说,成本是个非常重要的因素。目前国内已有相当的微机拥有量,多数被用于事务处理中,但是,CAD的外部设备和各种软件,相对于微机本身来说,价格冗贵,不是许多中、小企业的具体使用单位承受得了或愿意投资的。另一方面,目前的微机多数已具有基本的图形功能,一些普通的程序设计语言(如BASICA)也有简单的图形处理能力。因此,笔者认为,利用一般的微机配置和基本软件,低成本地开发CAD系统,满足用户的特定目标(包括所需的3维图形显示),对推广CAD的应用,开拓微机的应用领域,有实际的意义和价值。

事实上,笔者为国内某重型机器厂铸造分厂开发的大型铸钢件铸造工艺CAD系统,就是在基本配置的PC/XT微机上完成的。其中设计方案的自动拟定和图示<sup>[3]</sup>,凝固过程温度场仿真结果的3维图形显示<sup>[1]</sup>,都是由BASICA语言实现,在工艺图绘制方面也

想了一些办法,取得了良好的效果,为用户节省了大量开支。

## 三、速度和内存

在微机上生成3维图形是颇费时间的。这是因为在消隐处理中要进行大量的排序操作。为了能够进行排序,大多数市售的微机图形软件依赖于多面体模型。

工程上许多零件的表面包含曲面,其中多数又是数学上描述非常简单的二次曲面。这样的数学模型对科学计算是不成问题的。如果为了消隐而把它们转化为多面体,不仅在模型转换上非常麻烦费时,也引起数据结构的复杂化及消耗大量内存,并引起与科学计算的模型的不一致,不利于数据的传输和统一管理。这些问题,在32位工作站上利用专门的图形软件可以得到较好的解决,但对微机来说无疑是一种沉重的负担。

笔者认为,如果直接计算物体各表面轮廓线的可见部分和不可见部分的交点,可以避免曲面体模型转换及多面体各表面沿视线方向按深度排序所造成的大量的机时与内存消耗。这对处理任意对象是困难的,但在有特定种类对象的微机CAD系统中,由于同种对象在几何和拓扑结构上有规律可循,处理是成功的。图1是笔者用这种思想在PC/XT微机上开发的齿轮类铸钢件3维线框图生成程序<sup>[2]</sup>显示的图形。铸件的几何尺寸和观察角度都是可变的。用编译BASICA,每幅图的生成时间为20秒左右。该程序在日本札幌第三届程序设计大奖赛获优秀奖。

在作者后来开发的轧钢机架类铸钢件铸造工艺CAD软件中。对<sup>[2]</sup>中的方法又作了改进<sup>[1]</sup>。用计算得出物体一个表面(曲面)的可见轮廓线来处理这个表面之间的自遮挡问题,而用画家算法来处理不同表面的遮挡关系,从而避免了在齿轮类铸钢件3维线框算法中对不相交表面轮廓线交点的遮挡关系的计算和识别,进一步提高了速度。



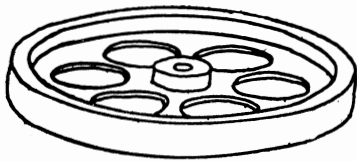


图 1

#### 四、精度

精度与图形显示速度有关,也影响图形的美观程度。合理考虑精度问题,往往给CAD系统开发者提供新的思路来提高3维图形的显示速度又不影响CAD系统的实用程度。一个成功的例子是笔者开发的大型铸钢件凝固过程仿真温度场3维图形显示程序。温度场仿真是通过差分或有限元计算出铸件的缺陷—缩松缩孔区的位置、大小和形状。形状的随意性给3维显示带来了困难。据笔者查阅到的文献,包括日本学者近期在IBM 3083EX上的工作,都是以2维剖面图的形式输出温度场仿真的结果,而没有采用更为直观的3维显示。笔者考虑到,既然温度场仿真的结果是以网格结点上的温度值作为对周围邻域连续变化的温度场的近似,那么在3维图形上作出类似的“近似”反映便能满足工程上的要求。采取的办法是以检索出的高温区结点为中心,以网格间距为边长扩充为立方体体元;根据视图方向确定体元的3个可见面,并填充不同的颜色,形成代表3维体元的2维彩元。按照视线方向和中心点坐标合理安排显示顺序,就能得到整个高温区的立体彩色图象,然后再用铸件的3维线框图衬托出高温区的位置和大小。立方体体元的采用,并没有降低计算结果的精度,但是显著

地简化了数学模型和处理方法。用编译BASICA生成的几何尺寸和观察角度可变的轧钢机架3维线框图和高温区彩色显示,不超过30秒。

#### 五、结束语

一个CAD系统的效率高低,不完全取决于所使用的单个支撑软件的功能强弱,而更依赖于各个部分之间合理的匹配关系。基于这点认识,结合自己的经验,笔者认为:

1. 目前市售的微机图形软件中,有的已带有3维功能。虽然它的功能较强,使用也比较方便,但是对系统的开销较大,数学模型和生成手段比较单一,速度也不理想。因此,对实用的工程CAD系统来说,针对具体对象和特定目标的特点,开发专用的3维图形生成程序,仍是可以考虑的一种途径。

2. 关于3维图形的数学模型和算法的选择,当然依赖于认真的系统分析。这种分析,不应只从计算机图形学的角度,而更应从整个CAD系统的角度来进行。耐人寻味的一点是:从计算机图形学方面,如通用性、精度、美观程度等所作出的让步,往往换来整个微机CAD系统性能的提高。

#### 参考文献

1. 黄有群、朱世铁、王巨森、张大坤,在铸造工艺CAD中用微机实现凝固过程仿真的温度场3维显示技术,第三届中日沈阳—札幌计算机应用国际学术会议论文集, P306—P308, 1988年9月。
2. 黄有群、许占文、朱世铁、王巨森,一种绘制回转二次曲面体线框立体图的直接方法,微小型计算机开发与应用, 1988年第6期, P12—P14。
3. 王巨森、张大坤、黄效直、黄有群、朱世铁,在轧钢机架铸造工艺CAD中用专家系统拟定工艺方案的几点考虑与实现。



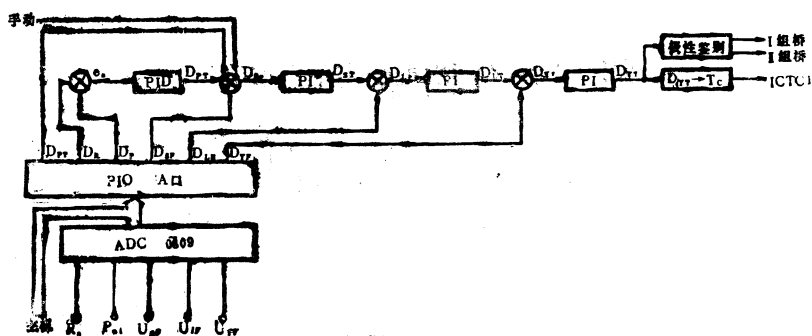


图2 系统调节原理图

主机控制时，由主计算机对前室总压信号 $P_{\Sigma}$ 和给定量 $R_{\Sigma}$ 的差值进行PID运算，运算结果输入到速度比较点上，以后与上述过程相同。

当需手动阀门时，先在某预定的存贮单元置入01H，再启动程序，然后操作手柄使CTC1或CTC3产生中断信号，利用中断服务程序把给定量 $D_R$ 加到速度比较点上，以后与上述过程相同。

#### 四、控制器原理

系统采用ADC0809模/数转换器作为各种模拟信号的入口。TP801的PIO A通道置输入方式，接收ADC 0809输出的数据；B通道置位控方式， $PB_2 \sim PB_6$ 与0809配合作进行通道选择和转换结束指示， $PB_0$ 、 $PB_1$ 分别为I组整流桥与II组整流桥开放或封锁控制， $PB_7$ 为过流跳闸保护。CTC0的计数到零脉冲启动风洞工作。此外，在TP801上还需要扩展两块CTC芯片。扩展的ICTC2由单板机工作时钟 $\phi$ 启动，对单板机工作频率四分频作为0809的工作时钟；ICTC3作为过流保护跳闸后的复位启动；系统的关键一环在于ICTC0与ICTC1，ICTC1接收由调节信号 $D_{\Sigma}$ 转换而来的 $\alpha$ 移相信号和同步脉冲信号，输出移相脉冲信号IZC/T01给单向b点移位寄存器，由两组寄存器分别按序触发两组全控整流桥。IZC/T01同时还启动ICTC0，使其产生与IZC/T01同步的300Hz时钟信号作为移位寄存器的工作时钟。扩展的ICTC0作为IZC/T01信号的延时，ICTC2和ICTC3分别作为阀门的关门和开门极限中断

保护。另外，单板机的数据总线还接入主计算机的某数据输出通道，作联机控制用。见图3。

模/数转换采用软件启动方式，由PIO B口的 $PB_3$ 作为启动信号输出端与ADC 0809的START和ALE相连由程序使 $PB_3$ 输出一个启动脉冲。

转换结束由软件查询获知。0809的转换结束信号输出端Eoc与 $PB_2$ 相连， $PB_2$ 定为输入方式，在转换启动脉冲发出后，即用软件反复查询 $PB_2$ ，当 $PB_2$ 为高电平时，说明转换结束，可以采入转换结果数据；EOC还直接与OE端相接，转换结束后，EOC变为高电平时，自动打开0809的输出数据缓冲锁存器，输出转换结果数据。

ADC 0809的工作时钟输入端CLK与扩展ICTC2的IZC/T02相接，由ICTC2对单板机的系统时钟 $\phi$ 进行四分频后向0809提供连续定时脉冲。若单板机的时钟频率为2MHz，则0809的时钟频率为500KHz，接近0809的典型时钟频率[4]。

6点单向移位寄存器A端接收经延时3.4ms后的IZC/T00脉冲信号，其6个Q输出端经脉冲放大器分别接 $SCR_1$ ， $SCR_2$ ，... $SCR_6$ （ $SCR_1'$ ， $SCR_2'$ ，... $SCR_6'$ ）。工作波形见图4。

#### 五、程序流程

程序流程见(33、34页)图5

#### 六、系统工作原理

主机控制，本机控制，手动控制是程序的三条主线。主机控制用于联机大系统自动

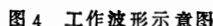
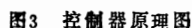
[illegible]

图5 程序流程图

$nI_Z$ , 则发拉 $\beta$ 信号, 即加大控制角, 抑制电流上升; 若 $I_F > nI_Z$ , 则单板机立即发生过流跳闸信号, 并使ICTC3初始化, 为中断复位启动作好准备。若 $I_F$ 在允许的范围内, 采样结束后 $D_{PT}$ 在各个比较点上逐次与 $D_{SF}$ ,  $D_{2F}$ ,  $D_{yF}$ 比较进行PI运算。运算完毕后, 对输出的调节信号 $D_{yT}$ 进行极性鉴别, 若为正信号则进一步与两个门限整定值 $D_H$ 、 $D_L$ 分别比较, 若 $D_{yT} > D_H$ ,  $PB_0$ 输出高电平开放正组桥; 若 $D_{yT} < D_L$ 则 $PB_0$ ,  $PB_1$ 同处于低电平, 两组桥均处于封锁状态; 若 $D_L < D_{yT} < D_H$

# 单片机通信控制器

海军医学研究所 王东平

**摘要** 本文介绍了以MCS—51中80C31为CPU构成的通信控制器的结构、原理及性能特点。实验证明,系统的设计是否合理的,成功地完成了成批,高速地进行数据传递的目的。

## 一、系统概述

我们研制的载波通信设备模拟训练器,是一个局部分布式多微处理器系统。

由于模拟训练器采用了数量众多的单片处理器来采集模拟机架及模拟仪表上的状

态,利用系统中的微型计算机对单片机实施直接的控制。因此,系统通信是一个十分重要的问题。图1为模拟训练器的系统框图。

图1中每一个机架有二片单片机,每一个仪表有一个单片机。整个模拟器有十六对机架,每对机架可同时挂四个仪表。这样系统

$T_c$ 。如果把初始控制角定在 $180^\circ$ ，则转换公式为：

$$T_c = \frac{KD_{YT} - T_{80}}{P_{tc}}$$

式中: P一定时器的定标系数

 $t_c$ —单板机时钟周期

K—常数26.0416

 $T_{60}$ —时间常数3.3ms

$T_c$ 装入ICTC1的时间常数寄存器后,由同步脉冲的下降沿启动定时器工作。当减1计数器计数到零发生IZC/ $T_{ol}$ 脉冲送到单向6点移位寄存器的A端,移位寄存器的 $Q_1$ 、 $Q_2 \cdots Q_6$ 端顺序间隔 $60^\circ$ 发出一串触发脉冲,经脉冲放大器依次触发可控硅 $SCR_1$ 、 $SCR_2 \cdots SCR_6$  ( $SCR_1'$ ,  $SCR_2' \cdots SCR_6'$ )。ICTC0产生的时钟信号作为移位寄存器的工作时钟,以保证触发脉冲的间隔为 $60^\circ$ 。同步脉冲取自 $U_{AB}$ ,它是根据三相交流电的固定相角关系以其中两相的线电压为基准,通过过零比较器变换而来。

当预定的存贮单元预置00 H, 程序启动后就转入主机控制。从主机输入调节信号 $D_{PT}$ , 加到速度比较点上, 与速度、电流、电压反馈信号比较运算后经过 $D_{YT}-T_C$ 转换装入ICTC 1发出定时脉冲。

从以上介绍看出,此方案对开发新产品具有重要意义。参考文献从略。

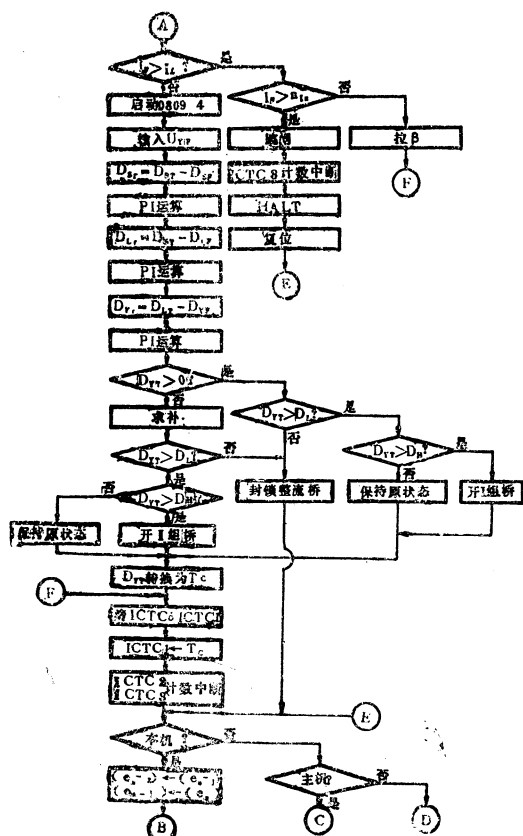


图5 程序流程图

则整流桥维持原有状态。同理,当 $D_{\tau}$ 为负,且 $D_{\tau} < -U_H$ ,  $PB_1$ 输出高电平开放反组桥;当 $D_{\tau} > -D_L$ 时,  $PB_0$ 、 $PB_1$ 又同处于低电平,两组整流桥均处于封锁状态。这一继电特性,大大提高了系统的抗干扰能力。

 $D_{yT}$ 经过极性鉴别后,转换为时间常数



中单片机的数量是比较多的。我们要求信息以一对机架为单位成批向主机汇报。为了达到模拟的逼真效果,我们规定系统的最长反应时间为一秒种。任一单片机采样机架或仪表面板状态的采样速率为每秒四次,且采样一次产生的信息量少于64bytes。

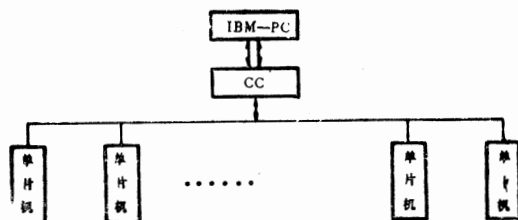


图1 系统框图

CC即为本文所要介绍的通信控制器

## 二、通信控制器的硬件结构

我们把CC作为主机的扩展部件来设计,使其直接插入PC主机的62脚扩展槽中。CC的CPU采用具有休息功能、耗电省的MCS—51中的80C31单片机,图2为原理框图。

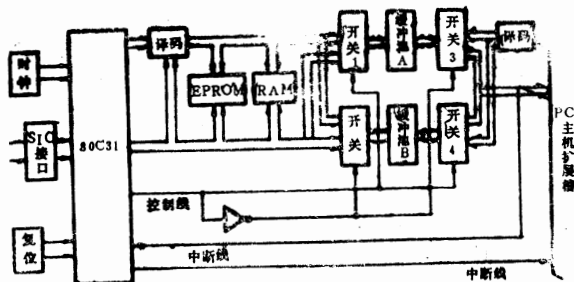


图2 CC的硬件原理框图

我们利用南京通信工程学院研制的“普及型微型计算机多种芯片开发装置”,对80C31实现了较为简单的开发,硬件的成本比较低。其中程序区为1片2764,数据区为3片8128(RAM),其中1片作为80C31的常用数据区,两片用作同PC通信的双缓冲池,所有开关电路视各种状态统一控制它们的动作。

## 三、有关技术问题

### 1. CC同主机通信

PC机可通过对其内部的DMA控制器82375A—5的编程而实现DMA操作,但是DMA方式中,实现高速数据传送的块字节传送方式所必需的通道0,已用作内部动态存贮器的定时刷新信号产生器,故PC机不能使用块字节传送。若用另外几种DMA方式,由于速率低,将会严重影响系统的实时性。若采用外接DMA控制器,也会由于PC机内部的地址总线及数据总线在其内部存贮器刷新时间内,不能被外部DMA操作占用。按L、C埃杰布赖奇特的建议:对于数据以短暂、高速、成批的方式传送时,采用“双缓冲池法”是最有效的。本系统正是具备了短暂、成批又需高速的特点。

### 2. 双缓冲池技术的实现

我们采用CC的CPU—80C31担任双缓冲池交换通信的主控任务,缓冲池总线的双向切换通过开关电路实现。

(1)容量:考虑一个对端所含最大信息量为 $6 \times 64 = 384$ bytes(一对机架,四个仪表),再结合主机下发所需容量及富裕量,我们选用一片8128RAM,2Kbytes。

#### (2)通信池的交换通信:

当PC机一侧的缓冲池填满下发的信息后,PC机通过CC外中断口INT0向CC发申请交换缓冲池的信息,CC判明允许交换时,交换缓冲池;若判明不允许交换,就实行交换进程登记,等到优先级较高的接收进程服务完毕后,再实施交换操作。

当CC一侧的缓冲池收完一个对端或一个独立机架所属各结点来的信息后,若判明PC机还未提出交换缓冲池的申请,则说明此时PC机还忙于处理上批汇报信息,PC一侧缓冲池中的信息还未准备好。此时进入等待。

### 3. 进程的设置及其相互关系

PC机一侧进程:处理进程1,2,3。

处理进程1:指键盘处理等高优先短进程。

处理进程2:被IRQ3中断唤起,对CC汇报来的信息实施处理。

处理进程3:指其它低优先级的慢速、长时的处理进程。

优先级的高低次序为:处理进程1、2、3,彼此为依靠优先级的中断支持的并发操作关系。

CC一侧进程:接收进程、交换进程、轮询子进程、轮询控制进程。

优先级的高低次序为:接收、交换、发送、轮询、轮询控制,它们彼此为依靠优先级的中断支持的并发操作关系。

#### 4. PC机处理进程2死锁问题的克服

PC机的处理进程2的操作软件全部直接驻在IRQ<sub>3</sub>中断服务程序中,当PC机处于处理进程2的服务时,CC的主控进程必须被挂起,以免引起IRQ<sub>3</sub>的多重中断,而导致PC机死锁。于是要求当CC收完一批信息并需汇报时,若发现PC机并未发送交换缓冲池的申请信号,则说明PC机仍处于处理进程2操作,此时CC等待PC机处理进程2服务完毕,直至PC机发出交换申请再执行交换进程,以免发生死锁。

#### 5. PC机下发命令操作与交换进程的互斥问题:

PC机下发命令是随机的,命令可来自键盘或其他进程的处理结果。由于当PC机执行完处理进程2后,其发出的交换申请信号消除了对交换进程的挂起,导致此后CC随时可能同PC争用PC一侧的缓冲池。若此时又有命令随机下发,则存在PC机同CC争用暂时出现的临界区——PC一侧的缓冲池。

为此我们在PC机一侧设置了一个信号牌——PC发送标识。此标识由处理进程2操作开始时清零,而在操作完毕时置位,在PC机下发命令操作时判明此标识,并用开设缓冲区命令信息的办法,解决互斥问题。

## 四、通信控制器软件

本通信控制器中软件主要有轮询控制进程模块、轮询子进程模块、发送进程模块、交换进程模块、接收进程模块。至于各进程间的互相调度,限于篇幅从略。

## 五、结束语

我们研制的通信控制器使用方便,可直接插入PC机扩展槽内;通信速率高,串行异步通信的最高速率为375Kbps;通信过程极少PC机干预,减少了PC机繁琐的网络通信管理;同PC机的进程间可并行操作。解决了一般PC机只提供一个RS-232串行通信口,且速率受限(最高为9.6kbps),特别是RS-232串行口已作它用情况下,提供了一个简单、可靠的通信接口。同时,只要将有关软硬件作些适当修改补充,本通信控制器还可用于工业自动化领域。

## 参考文献

- [1].《IBM-PC/XT软硬件系统分析与应用》张载鸿,科海培训中心,1987
- [2].《PC接口技术》L.C埃杰布赖奇特著,海洋出版社,1988
- [3].《分布式微计算机系统中通信池的设计与实现》,俞子荣,范仁周,微计算机应用,1985 2.

# 开发内藏MMU的68030

中国科技大学

许 薇 江 岳

**摘 要** 本文介绍高性能的68030 32位微处理器的主要特点。68030不仅与68020的软件具有互换性,并增加了许多新的功能,如内设存储器管理单元MMU、高速外部总线控制等等。为了使用户加深对68030性能的了解,文中以68030工作站—NEWS 1800系列为例,说明使用中应该注意问题。

## 一、概述

目前世界上已有不少公司用68030 安装工作站。日本的NEWS 1800系列就是一例。下面结合该工作站的研制,说明如何最大限度地发挥这种新型处理器的优越性。

68030是68020的加强型。加强主要方面为

### 1. 内藏MMU

68020没有内藏MMU,存储器管理要用外接的MMU芯片68851来实现,而68030的地址变换在片内通道中进行,可节约地址变换时间1时钟周期,使用了虚拟存储方式,缩短了总线周期数。

### 2. 增加数据高速缓冲存储器

68030增加了 256字节的数据 Cache,对Cache的存取时间从2周期减至1周期。内部总线可以双向向Cache存取,内部Cache的总线尺寸与成组传送相对应,从4字节变为16字节。

### 3. 加速外部总线控制

68030以2周期同步传送方式、成组传送方式作为新的传送支持方式。这样可缩短外部总线周期数,提高数据传输能力。

表1 是在最小外部总线周期状态下所需要的时钟数比较。

为了用最快速操作,还必须有相当高速的外部Cache相配合。成组传送方式是为了将Cache的一个记录项(16字节)高速写入而设计的传送方式。

表2是在改变内藏 Cache和外部存储器存取周期时所测到的Dhrystone基准点的数

值。测定是在配有68030的NEWS—1850工作站上进行的,主频为25MHz。由表可见,内藏Cache有一定的效果,在外部存储器以最高速运行时,可将性能提高8%左右。在外部存储器滞后时,成组传送方式具有明显的效

表 1

使用的微处理器	外部存取方式	1 长字存取的总线周期最小值
68020+68851	非同步传送	4时钟周期
68030	非同步传送	3时钟周期
68030	非成组同步传送	2时钟周期
68030	成组同步传送	5/4=1.25时钟周期

表 2

内藏指令Cache	内藏数据Cache	外部存储器存取	Dhrystone值(实测)
使用	不使用	取出:7时钟周期 存入:2时钟周期	3472
不用成组传送	不用成组传送	取出:7时钟周期 存入:2时钟周期	5008
用成组传送	用成组传送	取出:7时钟周期 存入:2时钟周期	6864
不使用	不使用	取出:2时钟周期 存入:2时钟周期	8086
不用成组传送	不用成组传送	取出:2时钟周期 存入:2时钟周期	8771
用成组传送	用成组传送	取出:2时钟周期 存入:2时钟周期	8426

果,而在外部存储器最高速运行时,不用成组传送反而更好。

## 二、存储器管理单元MMU

图1 是68030和68020分别与外存储器连接的结构示意图。

68030 在虚拟存储环境下多采用页面管理的操作系统,MMU使用的方便程度和性能将左右系统的性能。由于68020内不设MMU,所以从地址引线输出逻辑地址,而在外部设置的68851 内将逻辑地址转换为物理地址。68030内藏MMU,它可用物理地址存取外部的Cache。

和Intel 80386相似,68030的MMU中也设置了转换后援缓冲器TLB(或称ATC),用于控制地址变换。运行时万一TLB出错,产生总线错误,在错误处理程序中可以进行页面表的检索,纠正TLB的错误。

与68020配合使用的外部Cache多数作为逻辑Cache,而用作物理Cache时,由于使用的是外接MMU,不可避免地要延时等待地址变换。逻辑Cache在过程转换时,所有内容都需要标识(flash);I/O装置(由DMA构成)使用物理地址,在存储器存取时也需要进行标识。

下面介绍使用中与MMU 有关的几个问题。

### 1. 68030优于68851的附加功能

由于68030内藏MMU,本身就可输出物理地址,用物理地址存取Cache就不需要等

待,物理地址和逻辑地址同步输出,加快了变换节奏。物理Cache在过程转换时不需要标识,所以可以达到大容量化,提高了准确度。一般在使用物理地址进行存取时,可以按物理存储器的地址进行存取,而不必把32位都处理,地址的高位部分可以省略。此外,还具有用物理地址对存储器存取的I/O进行设计、共用Cache等优点。

新增加的附加功能还有“透明地址窗口”。当控制寄存器在指定的地址范围内存取时,逻辑地址首先直接输出,为了使图形的页面缓冲寄存器及OS的常驻部分能频繁地存取,在窗口中配置了高速的不连续地址存取存储器。可以指定两个透明地址窗口,该窗口的最小尺寸是16M字节。两个窗口可以重叠。

从逻辑地址向物理地址变换,最多设有4级块结构表,转换状态的输出信号使用操作码,由一级表进行分配(操作码检查)。显示表基址指针可以有两个,当ATC未击中时,就执行搜索多级表的“表操作”。

68030 MMU的基本特点可归结为:

- (1) 页表(page table)用物理地址存取
- (2) 块结构多级表
- (3) 表(table)描述符和页(page)描述符都有32位短格式和64位长格式两种
- (4) 短格式的页描述符没有单独的监控位(S位)
- (5) 由于S位是写禁止位,就不能保证“进行核心读写/用户读出”。

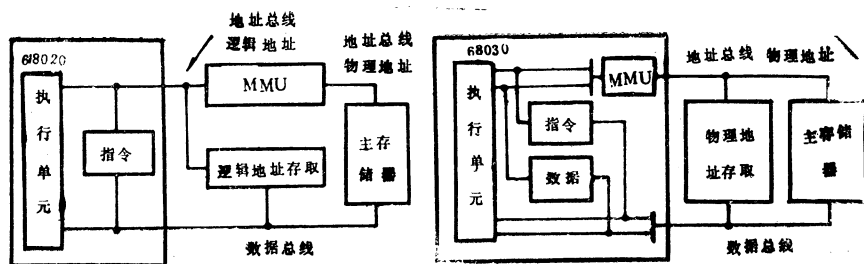


图1 使用68020和68030的存储器系统比较

2. 把NEWS操作系统向内藏MMU移植问题。

图2给出了在NEWS-OS 3.0版本中的地址变换方式。页面大小为4k字节，地址变换表为三级结构。

把逻辑地址分为4个区域，第二级表的设计原则是保证第三级各页表的大小正好为一页来进行动态配置的。描述符采用长格式，以便实现单独监控位和极限校验功能。基指针只用一个，不采用68030MMU特有的操作码检查等功能。

使用页描述符中的Cache禁止位及内藏Cache的操作由页来控制，I/O通道的存取则使用透明地址窗口。

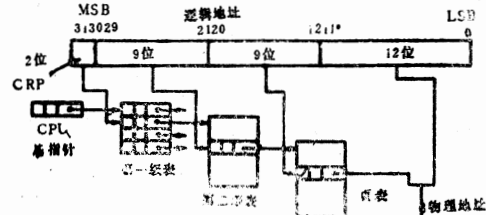


图2 NEWS-OS 3.0中的地址变换  
3. 存储器映射问题

改变存储器映射时，必须由逻辑Cache标识。68030既有内藏指令Cache又有内藏数据Cache。数据Cache的写入采用存储通过方式。内藏高速缓冲存储器作为逻辑Cache时要注意：由转换产生存储器映射的变化，会导致因DMA传送而出现与外部存储器不一致，所以必须进行Cache标识。在磁盘I/O缓冲寄存器的区域中，若页描述符中的Cache禁止位有效，则Cache的动作立即停止。

系统初始化高速缓冲控制寄存器（CACR）用来指定是否使用内藏指令/数据Cache以及其被改写时是否使用成组传送方式。

#### 4. 避开双重变换问题

内藏数据Cache的写分配功能由CACR中的WA位指定，可指定下述动作：

(1)  $WA=0$ ：写入时如没有击中，不动作。

(2)  $WA=1$ ：写入时如果没击中，作记录。这时在4长字组成的新的记录中，实际写入的1长字以外的有效位全都复位。

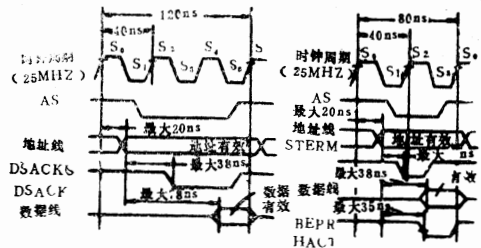
不同的逻辑地址可能对应同一物理地址。此时如果 $WA=0$ ，将发生Cache和外部存储器之间不一致，遇到这类情况，可以用 $WA=1$ 来防止。在直接变换方式遇到总容量比一页小时，页内的偏移在相同的地址情况下必定给出相同的Cache记录进行变换，所以在 $WA=1$ 的情况下，Cache记录要进行更新。

### 三、同步传送和成组传送

下面介绍新的支持方式一同步传送方式和成组传送方式。

#### 1. 同步传送方式

图3为68030总线周期的基本时序。(a)是与68020相同的非同步传送方式，最小总线周期为3时钟周期；(b)是68030的同步传送方式，总线周期2时钟周期。 $\overline{STERM}$ 信号叫同步终止信号，同步传送方式时要维持 $\overline{STERM}$ 信号有效（该信号取代了68020的 $\overline{DSACK}_{0..1}$ 信号），并将总线周期结束通知68030。同步传送方式可以有效地保证高速存储器存取，但数据长度限于32位，且不能使用动态总线。



(a) 非同步传送方式 (b) 同步传送方式

图3 68030的外部总线周期

当主频为25MHz时，为了使最小总线周期为2时钟周期，从确定地址开始，必须在18ns以内将 $\overline{STERM}$ 信号维持住，在此期间进行Cache的击中判定。但是控制 $\overline{STERM}$ 信号是很困难的。为了降低Cache击中判定的难度，在68020系统中一般设置Cache击中时，使用保持 $\overline{DSACK}$ 信号的总线出错重复运行方式。这种方式也可用于68030系统。如果没有击中，要从 $\overline{STERM}$ 信号的维持开始，在延缓1/2时钟周期（25MHz时为200ns）内将 $\overline{BERR}$ 信号和 $\overline{HALT}$ 信号同时维



持住,总线周期再运行一次.但即使用这种方法也不够理想.因为出错的判定和BERR信号及HALT信号的操作必须花费35ns左右,所以从出错到总线周期重新运行还需要加上2 时钟周期的附加操作时间。

### 2. 成组传送方式

成组传送方式是将Cache的一个记录项高速写入。下面分述其特点。

#### (1) 时序

图4 是成组传送的基本时序。

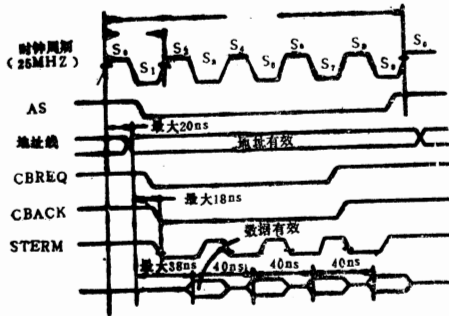


图 4 68030 的成组传送周期

68030由CBREQ (Cache成组请求)发出成组传送要求,外部电路由CBACK(Cache成组响应)和STERM响应,进行成组传送。16字节的数据用32位分4次以地址递增的顺序传送。这时从 68030的地址输出并没有变化,但必须在外部分增加计数器以构成地址修改电路。

#### (2) 成组传送速度不拘

如表2所示,当外部存储器滞后时成组传送方式有明显的效果,但当外部存储器以最高速运行时,性能反而下降。理由是:

若取4长字(16字节)数据为例,在非同步成组传送方式时,访问一长字要2 时钟周期,4长字要8个周期。而成组传送方式虽然4长字的读取只需5个周期,但由于后三个长字在Cache中,从Cache中取出每长字要用1 周期,结果合计8 周期,与非成组传送相同。在该例中,成组传送完长字后,假定其为有用可访的数据,但实际上也可能用不上。如果把不必要访问的长字传送时间也考虑进去的

话,成组传送方式反而不占优势。表3 为有无成组传送所需要的时钟周期数比较。

表 3

实际存取长字数	1	2	3	4
非成组同步传送方式	2	4	6	8
(外部总线周期数)	(2)	(4)	(6)	(8)
成组同步传送方式	5	6(5+1*)	7(5+2*)	8(5+3*)
(外部总线周期数)	(5)	(5)	(5)	(5)

\* 68030 内部存取周期

(3)外部Cache与存储器之间用64 位总线连接。

图5给出了68030的存储器系统结构,它被用于NEWS—1800系列之中。主存储器用1M位的较慢速度的DRAM,价格较低,容量为16MB,也可扩展至32MB。成组传送可以高速进行,可减少存储器的等待状态,主存储器由交替使用的存储体构成。外部Cache的容量为64KB,采用8个64K位的SRAM,采用存储通过方式直接变换写入。在外部Cache击中时68030以2时钟周期同步传送方式运行。Cache的总线宽度为16字节,总线宽度与68030内藏Cache相对应,以简化运行控制电路,并与主存储器的两交替存储体对应。Cache的数据长度为64位,Cache和主存储器两存储体之间分别皆用32位总线连接,形成双重并联总线。

(4) 从主存储器到外部Cache同时进行读出数据

存储器读出有四种情况(参见图5):

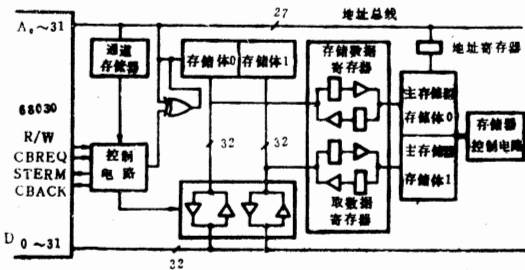


图 5 NEWS 1800系列的存储器系统构成

## 莫托洛拉的 MC68040

莫托洛拉的下一步微处理器 MC 68040 的概况简要介绍一下(开发过程中可能会有所变动)。

最大特点是具有几个可并行处理的执行单元的虚拟存贮微处理器。

预计达到的指标: 整数运算处理能力 13.5 MIPS (25 MHz), 浮点数运算能力 3.6 MFLOPS (25 MHz)。

MMU 具有 2 个 ATC (或称 TLB—转换后援缓冲器), 4K 字节命令高速缓存和 4K 字节数据高速缓冲可同时存取。

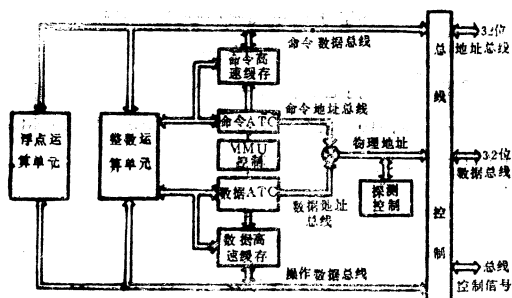


图1

天津市电子计算机研究所 陈荣华

1) 68030 不提出成组传送要求, Cache 命中。这时 Cache 控制电路从与存储体地址相对应的外部 Cache 中选择 1 长字, 从数据总线上输出。

2) 68030 提出成组传送要求, Cache 命中。这时 Cache 控制电路将外部 Cache 的两个存储体交换两次, 合计读出 4 长字。

3) 68030 不提出成组传送要求, Cache 未命中。这时存储器控制电路从主存储体 1 取 4 长字读出后, 向 Cache 传送, 并同时更新通道存储器。Cache 访问时, 从主存储器读出是使用 DRAM 的高速页面方式, 对两个主存储体分别进行二次。Cache 控制电路在 68030 指出的数据线上输出 1 长字。

4) 68030 提出成组传送要求, Cache 未命中。这时 Cache 控制电路把主存储器的内容读出 4 长字向 Cache 传送, 同时也向 68030 传送 4 长字。

### 5) 快速写入存储缓冲寄存器

由于采用存储通过方式, 在执行存储指令时, 不论 Cache 是否命中, 都可以向主存储器写入, 而向 Cache 写入只能在命中条件下进行。在写入时, 68030 的总线周期以存储缓冲寄存器的数据和地址的命中而告结束。存储缓冲寄存器如不关闭, 同步传送方式的总线周期为 2 时钟周期。外部 Cache 容量越大, 越能提高击中率, 就越能发掘 68030 的性能。

### 6) 通道存储器对高速化至关重要

随着新器件主频的提高 (如 33 MHz), 支持 2 周期同步传送方式时, 即使采用总线出错重复运行方式, 也必须在 25 ns 内判定 Cache 是否命中。这是不易实现的。目前世界各公司均把注意力集中在研制高速通道存储器上。

(参考文献从略)

# AST Premium/386C概述

吉林省长春市省委办公厅

刘铁军 张皓月

**摘要** AST Premium/386C是美国AST公司(中文名虹志电脑有限公司)继1986年推出的AST Premium/286,一年后推出的AST Premium/386,于1989年推出的32位高级微型计算机,AST系列微机曾多次在微机评比中名列前茅。AST P386C同它的兄弟一样,自面世以来受到了用户的青睐。AST Premium/386C有五种不同型号,即Model 300、Model340、Model390、Model3150、Model3320,Model300硬盘为选件,下面以390型为例,对AST Premium/386C(以下简称386C)作一概述。

## 一、硬件配置

386C采用20MHz的 Intel80386微机处理器,系统板上可安装以20MHz运行的80387或以8MHz运行的80287协处理器,具有2个串行口、1个并行口、时钟/日历以及软盘驱动控制器,有7个扩展槽,第7个扩展槽专用于386C的32位数据传输的存储器板,其余有两个扩展槽是标准XT或8位槽,一个标准PC AT或8/16位槽,三个槽为多主控总线(SMART SLOT),与AT兼容,允许外围设备与内存直接存取,不用CPU介入。

386C存储器板是32位数据传输,装有64KB零等待状态25ns超高速缓冲内存,2MB SIMM(Single inline memory modules)存储器,最大容量达16MB而不用多占扩展槽。第一个MB中,640KB用于标准存储器,128KB用于阴影存储器,256KB保留备用。存储器板上有四个存储器体,每个体包含4个SIMM插座,每个SIMM可以是256KB或1MB,每个体随使用的SIMM类型不同能容纳1MB或4MB,体号从0到3,要按顺序安装,不允许跳体,一个体必须完全用1MB或256KB SIMM而不允许混合安装。存储器板上设有体分配开关,开关1、2为体1,开关3、4为体2,开关5、6为体3,开关7为奇偶校验,开关8为体0,由于规定了每个体内SIMM容

量必须相同,因此,开关组合设定只表示体是采用256KB还是1MB SIMM,也就自然规定了每个体以及存储器容量。

386C留有三个半高软驱空间,可选装5.25英寸360KB(双面双密)、5.25英寸1.2MB(高密度)、3.5英寸720KB(双密)、3.5英寸1.44MB(高密)中任一种尺寸软盘驱动器,可安装三个半高或一个全高一个半高软驱设备,只安装两个软驱设备不需装入实用程序,如装第三个软盘驱动器,需装入DRIVER.SYS设备驱动程序和ASTDSK.DRV实用程序以及在CONFIG.SYS文件中加入相应语句。例如:加360KB驱动器,在CONFIG.SYS文件中加入:

```
DEVICE=C:\DOS\ASTDSK.DRV
```

```
DEVICE=C:\DOS\DRIVER.SYS/D:2/F:0
```

规定软驱动设备装在右边。

硬盘驱动器一般装在左边,但也可装在右边,可装两个半高或一个全高硬盘驱动器。所有AST硬盘已在工厂用MS-DOS进行了分区和格式化,容量超过20MB的,计算机断电时,都具有磁头自动锁定功能,因此,当搬运时,不必专门作磁头锁定。

为从硬盘后备文件,可加一个磁带驱动器。AST公司提供高速磁带系统(硬件和软件),能从硬盘拷贝文件到盒式磁带。磁带驱动器是选件。

386C前面板有复位按钮、硬盘锁,当用钥匙锁上时,不能对硬盘操作,还有CPU自动降速、CPU高速、中速、低速及磁盘存取

指示灯, 可通过键盘组合从一种速度变为另一种速度, <Ctrl> - <Alt> - <UP Arrow> 为增速, <Ctrl> - <Alt> - <Down Arrow> 为减速, 也可通过调用实用程序SPEED改变速度。后面板有230V/115V 电压转换开关。键盘为101键或102键标准键盘, AST 公司提供其他语言的成套键帽。显示可选用单色、彩色以及 EGA、VGA, 机器内部设有一开关, 指示为单色或彩色状态。

## 二、386C采用的某些先进技术

为了提高速度, 充分发挥386 芯片的高速度、16MB的寻址能力, 突破MS-DOS 640 KB寻址的限制, 386C采用了多项先进技术。

1. 存储器高速缓存: 存储器板上装有640 KB零等待高速存储区, 它不包括在扩展内存中, 当CPU试图存取主存储器, 先搜寻高速缓冲区, 如果找到, 把相应数据以高速缓冲速度送到 CPU, 避免对主存储器更长的存取时间。

2. RAM 磁盘: 在扩展内存中开辟一个区域作为虚拟磁盘, 它不受机械驱动器的速度限制, 而以RAM速度工作, 但在断电时, 数据丢失, 因此, 需在断电前把数据拷到磁盘上。该功能需在实用程序 fASRDISK支持下才能使用。

3. 阴影RAM: 它是一个128KB的 32位RAM区, 开机启动时, 自动把BIOS从16位ROM 中拷到该区, 并作写保护, 使机器运行时不能改写该区, 由于BIOS 控制计算机的最低级操作, 是硬件和软件的接口, 阴影RAM 提高了运行速度。

4. 磁盘高速缓存: 把最近经常使用的磁盘扇区数据存入缓冲区, 当计算机搜寻数据

时先搜寻缓冲区。该功能在实用程序 AST Cache支持下使用。

5. 386C提供了CPU在存取软磁盘时由高速自动降为中速的功能, 可在运行AST-SETUP程序时, 设定是否使用该功能。也可通过调实用程序SPEED改变是否使用该功能。

## 三、实用程序

386C 的一些功能是通过调用程序实现的, 提供的实用程序如下:

- ASTMENU: 提供系统运行、安装实用程序和构造硬盘等功能的菜单界面。

- ASTcache™: 改进密集型磁盘应用的性能, 提供磁盘高速缓冲功能。

- ASTEMM: 使用线性扩展 存储器仿真页式调度的扩充存储器。AST 的扩充内存管理程序完全支持EMS4.0 工业标准规范。

- ASTDSK: 使系统能增加 第三个软驱装置。

- fASTDISK™: 提供 RAM 磁盘功能。

- Super SPool™: 在RAM中建立假脱机缓冲区, 接收 打印文件并转送到打印机。

- SPEED: 通过在DOS提示符下输入命令, 改变 CPU 速度或自动降速特性。也能在不运行ASTSETUP情况下, 改变是否使用内存高速缓冲功能。

- 386TEST: 386C检测程序, 检测一旦开始不需要人工干预。检测采取限制次数和时间方式, 不破坏硬盘数据。

## 四、386C安装

386C随机带有MS-DOS3.3 和实用程序两张软盘, 机中ROM装有GW-BASIC。

# VAX-8350计算机的诊断系统

天津师范大学计算中心 李瑞成

**摘要** 本文介绍了美国DEC公司生产的VAX8350超级小型计算机的硬件配置情况。并重点分析了诊断系统的构成、特点及应用。分级分层的由浅入深的介绍了如何选定诊断对象和诊断程序的使用。着重以主机、硬盘、磁带机为例,介绍了如何对这些设备进行诊断。

386C提供了简单方便的ASTSETUP安装程序,在下列四种情况下运行该程序:

- 1.初装386C硬件
- 2.换386C的后备电池
- 3.加或取出线性存储器(标准的或扩展的)。在扩展存储器运行扩充存储器软件(使用386C扩充功能管理程序)不需要运行ASTSETUP。

- 4.改变显示适配器类型。

386C提供了两种调用ASTSETUP方法:一是通过调用实用程序,在ASTMENU菜单下选用ASTSETUP;二是调用ASTSETUP的固件版本,冷或热启动后,按<Ctrl>-<Alt>-<Esc>保持到自测试(P-OST)信息出现,显示:

```
KEYBOARD ERROR OR NO KEYBOARD  
PRESENT  
PRESS F1 KEY TO CONTINUE OR CTRL  
-ALT  
ESC FOR SETUP
```

后,按<Ctrl>-<Alt>-<Esc>键,进入ASTSETUP。

ASTSETUP把系统配置写进电池后备的存储器。电池后备存储器存储系统配置,每次自举时进行读识别。除非由ASTSETUP建立和写入。系统是不认可的。例如系统加了软盘或硬盘驱动器等设备,但不运行ASTSETUP程序作相应设定,系统认为所

加设备不存在,不能使用。

除了加设备外,ASTSETUP还控制着386C的下列性能设定:

- 日期和时间
- 标准存储器容量
- 扩展存储器容量
- 嵌入口是否可用
- 是否用系统板上软、硬盘驱动控制器
- 是否在磁盘存取期间自动降速
- 系统缺省操作速度
- 是否用阴影RAM
- 开机时数字键盘是否锁定
- 是否用高速缓冲存储器
- 机器无键盘插入时是否自举

## 五、结语

386C高级微机系统提供了优良硬件环境和良好的软、硬件兼容性,性能价格比较高,支持现有的或先进的操作系统,例如:MS-DOS、OS/2、XENIX、PC-MOS,支持多类32位软件,是多任务、多用户、CAD工作站、网络服务器以及单机应用的理想选择。

## 一、VAX—8350计算机简介

美国DEC公司生产的VAX—8350计算机是VAX 8000系列计算机中的中档产品,具有高性能处理能力,并与其他VAX系列机具有软件兼容性。它有两个CPU,属于紧凑型的双处理机结构。它有32 M字节分时存贮存取能力,主机字长为32位,有4G字节的虚拟地址空间。

CPU使用虚拟存贮管理,自启动加载,标准的指令系统,它可以处理字符串,有2个8K字节的高速缓存。主机还包括控制台子系统和RX—50软磁盘驱动器。

系统使用VMS操作系统,也可装入ULTRIX—32操作系统。VMS可以提供可靠的、高性能的环境,以保证多用户分时系统的使用,并提供批处理和实时系统。

## 二、VAX计算机的诊断系统简介

VAX 诊断系统是一种有很大范围兼容的分等级的诊断程序,它为现场维护工程人员及用户提供了硬件检修的有力工具。可以对系统失败进行局部分隔、进一步细致的诊断,加快维修速度。

诊断程序从功能上是从一般到特殊。系统诊断控制程序EV×××检查出错的功能模块和子系统,更进一步,微诊断程序可以检查出印刷电路板或大规模集成电路。诊断程序可在无操作系统的条件下(即脱机状态)运行,也可以在操作系统下的用户外围环境下运行。

VAX诊断系统的命令是很丰富的,命令为操作人员提供了可选择的诊断项目,对机器出现的问题,可以有针对性的、灵活的使用诊断程序。

## 三、VAX-8350计算机的诊断功能

VAX—8350计算机的诊断系统大致可以分为外部设备的诊断和主机的诊断两个部分。VAX—8350计算机的外部设备主要有磁盘子系统,磁带机子系统及终端、打印机等。

所有外部设备本身都有开机加电自测试诊断的功能。如果机器有故障,则在终端屏幕、打印机的指示灯上,行式打印机、磁带机、磁盘机的显示屏上均可以看到有关故障的指示,可以根据故障情况,决定下一步的诊断与维修。

有些外部设备可以在加电之后,选择OFF LINE状态,进行有针对性的测试,如打印机可以选择测试打印某些字符,测试走纸等机械动作等。磁带机可以通过面板按钮选择测试项目,如读/写功能、正/反绕带的功能等。

磁带机和磁盘机还可以通过机内的插座,外接VT220终端,运行装在设备中的测试程序。如检查RA81磁盘机,将传输波特率设在300,按CTRL+C后,终端显示RA81>提示符键入RUN DIAG/TEST=nn,其中nn为测试的顺序号码,如TEST=00为检测全部的驱动顺序,TEST=01为测试主板的ROM,TEST=02为测试全部LED,TEST=10为读写测试。对于RA81有44项测试项目。

磁带机TU81也有类似RA81的测试功能。

## 四、主机的诊断及诊断程序的构成

### 1. 主机的诊断

主机开机后先进行自测试,在LA100控制台上打印出有关CPU板的检查结果。#ABC……#。接着打印出主机箱各节点的模板的检查结果。0.2.4.6.7.8.……,如果某节点出现负号,如-2,则表示此节点模板



出现问题。

如果开机后能够进入操作系统,VAX 机内有可以对系统检查和分析的程序包,其中一个为用户环境测试程序UETP,它虽不是诊断程序,是一个在工作方式下,证明VAX/VMS系统中软件和硬件组成及联接完好的软件包。另一个是系统转储分析报告SDA,它记录系统失败的经过和原因。当系统失败时,SDA把信息写入预先准备好的文件中,这些信息包含操作系统失败时的状态,同时,SDA也检查系统转储文件的格式和内容。

用户通过UETP和SDA，能对系统失败进行分析和检查，这是非常有效的工具。

下面介绍诊断程序包，可以进行进一步的有针对性的诊断。

## 2. 系统诊断程序包

系统诊断程序包含6级诊断:

1级: 以VMS为基础的诊断程序。

**2R级:** 在VMS条件下的诊断管理为基础的  
系统诊断控制程序。不支持  
脱机方式下的外围诊断程序。

**2级:** 可以在VMS(联机)方式下和脱机方式两种条件下的诊断管理为基础的诊断程序。它是一个总线互连程序,是可靠的外围环境诊断程序。

**3级:** 只允许在脱机方式下工作的诊断管理程序。它包括外围环境功能诊断和维修级的诊断,CPU 簇系统诊断。

**4级：**脱机微诊断，它不用诊断管理程序支持。可以测试硬核指令系统。

**控制台级：**以控制台为基础，在脱机方式下，对微诊断、控制台程序、ROM 驻留程序的通电加载进行测试。

系统诊断方式如图1所示:

我们可以看出这6种诊断程序工作在2种方式下,在4种环境下进行操作。

### 3. 诊断程序的加载和控制序列

VAX 机的诊断系统在程序的加载通路和不同的程序级的控制上提供了一些灵活性。2级与3级诊断可以从控制台的软盘上也可以从系统盘上加载,微诊断程序通过控制台的软盘加载。图2 说明了两种方式下诊断程序加载环境。

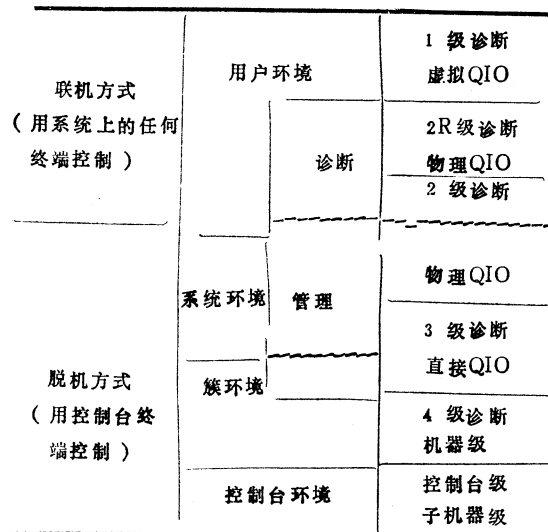
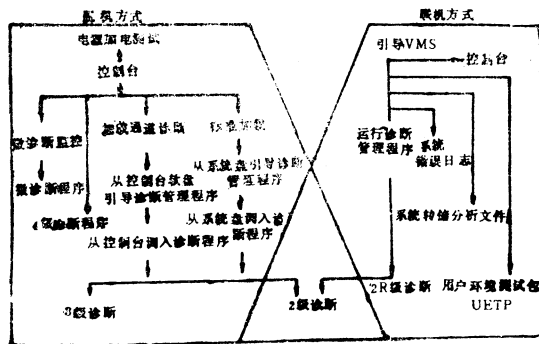


图 1



**图2**

其中诊断管理程序可以由系统加电自动引导,调入诊断管理程序出现提示符DS>,有一套诊断系统的命令,可以根据需要进行操作。如运行EVKAB可以检查CPU的基本指令。运行EVRLLK可以检查磁盘控制器和磁盘驱动器的读写功能。

VAX—8350计算机的诊断系统比较完善,但也不是十全十美,有时还要从几个方面着手进行检查分析,再根据已有的维护经验综合考虑,才能确定故障的具体部位。

## 排除接地干扰的几则实例

接地对微机系统工作的稳定性有着直接的影响。但是,干扰是否由接地造成的,判断起来却比较困难。因此,在遇到干扰时,一方面要进行必要的检查和分析,找出干扰源;另方面,不妨改变一下接地的方式或位置。经过改变,如果干扰消除了,就可以证明干扰是由于原先接地不合理造成的。这里有几个排除接地干扰的例子,就是用这样的方法解决的。

### 一、计算机和A/D、D/A

#### 间接地干扰的排除

计算机的接口电路,如A/D、D/A转换器的接地也是非常重要的,接地不妥,会造成一定的干扰和误差。如图1所示,被采样对象信号源 $U_i$ 的地如果不直接连A/D的地(在现场布线中是常有的事),而是接电源地, A/D的地又没有直接接电源地,则A/D输入端的电压不是 $U_i$ ,而是 $U_i - i_1 R_1 - (i_1 + i_2) R_2$ ,式中的 $R_1$ 、 $R_2$ 是导线的电阻,  $i_1$ 、 $i_2$ 为流过A/D和微机的电流。由于 $i_1$ 、 $i_2$ 不可能是常量,则A/D输入端的电压会时常变化,因而采样所得的数字量也会发生不规则的波动,给采样造成一定的干扰和误差。特别是高精度高分辨率的A/D转换器,干扰和误差就更严重。同样对图2的被控对象, D/A的输出值也会因电流流经地线电阻 $R_1$ 、 $R_2$ 而产生干扰和误差。正确的接地方法如图3、4所示。

A/D的各有两根线。即D/A给定电压线及其地线;被采样信号的信号线及其地线。而在多路D/A或A/D计算机系统中,引入各路D/A或A/D的除了各自的给定电压线或采样信号线外,各路也应有各自的独立地线,如图5所示。这样可以避免各通道之间公用地线造成的相互干扰。

但在现场布线中,如果使各通道公用地线,如图6所示,而忽略了各路电流 $i_1$ 、 $i_2$ 、 $\dots$ 、 $i_n$ 流过公用地线时在地线电阻 $R_{地}$ 上产生的电压 $R_{地}(i_1 + i_2 + \dots + i_n)$ 。这一方面会使被控对象上得到的给定电压产生误差;另方面,由于在实时控制系统中,给定电压 $V_1$ 、 $V_2$ 、 $\dots$ 、 $V_n$ 是实时变化的,  $i_1$ 、 $i_2$ 、 $\dots$ 、 $i_n$ 是实时变量,  $R_{地}(i_1 + i_2 + \dots + i_n)$ 也就是实时变量,因而加在被控对象 $Z_i$ 上的电压将不是 $V_i$ ,而是 $V_i' = V_i - R_{地}(i_1 + i_2 + \dots + i_n)$ ,这将对被控对象产生干扰和误差。因此应该采用图6那样的各通道间独立接地的方式,防止各通道之间的接地干扰。同样,多路A/D也应采用独立的通道地线。

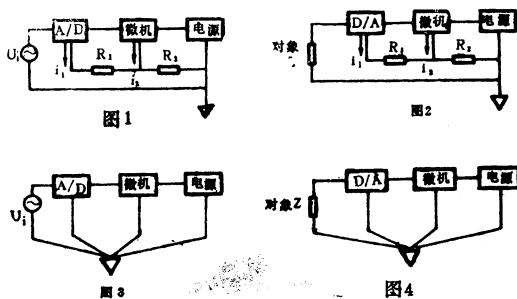
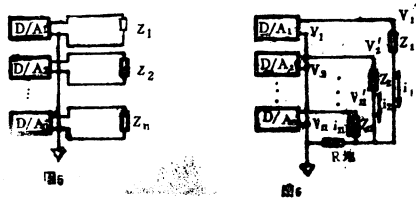


图1~4

### 二、防止多路A/D或D/A

#### 间相互干扰的方法

在只有单路A/D或D/A的计算机系统中,从被控对象或被采样对象引入D/A或



## 用WIDTH命令定义屏幕显示的行宽

BASIC语言规定:PRINT语句中各打印项之间用逗号隔开时,则为标准的输出格式。如果一个PRINT语句中的打印项的个数超过一行标准位置的个数时,则计算机在打满一行的标准位置时会自动换行,且按标准位置对齐,但这在BCM—0530机上会出现异常的情况。

BCM—0530机屏幕显示一行为6个标准位置。当程序中一个PRINT语句内所写的显示项个数超过6个,且各显示项之间用逗号隔开时,则一行显示完6项后虽能自动换行但不能按标准位置对齐。若显示的行数较多

时,更会使结果显得参差不齐,根本不能显示成表格的形式,因而不能满足程序设计者的本来要求,这是笔者在使用BCM—0530机时发现的一个新问题。解决这一问题可采用下面的方法:

在运行程序前,打入命令“WIDTH 行宽”,以定义显示屏幕的行宽。例如,在运行程序前,打入“WIDTH 80↵”,则运行结果能按标准位置对齐输出,但要注意的是“行宽”不能超过80,若越过80,则换行部分有时不能显示出来。

湖南省金融职工大学 石爱华

### 三、排除微机 and 高温探测器 间的接地干扰

在光导纤维预制棒熔炼现场,火焰温度的探测采用红外辐射高温计。高温计的探测器装在熔炼车床的金属托架上,其二次仪表和微机系统装在一起。该仪表本身有温度数字显示,为了和计算机配合使用,还增加了对应于火焰温度的0~5V电压输出,微机通过0~5V电压对火焰温度进行数字采集。在现场调试中发现,高温计数显表上的温度值显示稳定,而计算机屏幕上的温度显示却始终和高温计数显表上的温度值相差20℃左右,并且有±3℃左右的波动。检查产生干扰和误差的原因,情况是:在现场安装时,计算机系统的金属外壳接了大地,其直流地与机壳是浮置的。就计算机系统本身而言,这样

做具有较强的抗干扰能力。从对高温计的检查情况看,高温计探测头的金属外壳和车床连在一起,探测器的直流地又与其外壳连在一起,车床吸收了现场大量的干扰和噪声,使探测头的直流地成为噪声地。由于火焰温度数字采集使用的是逐次逼近高速A/D转换器,抗干扰能力差,因而计算机屏幕上的显示有误差且不稳定。将高温计探测头的外壳和车床隔离起来后,屏幕上的温度显示就和数显表上的温度值完全一样并且稳定了。

事实证明,系统能否稳定地工作于设计的最佳状态,与接地有很大的关系。

武汉邮电科学研究院 张远长