

如何使用扩展内存和扩充内存

南昌陆军学院计算机室 戴水贵

摘要:目前,一般的高档微机都配有扩展内存和扩充内存。本文作者成功地把汉字库放入了扩展内存和扩充内存,并在主内存中进行调用。文中对这两项技术作了简单介绍。

一、引言

扩展内存(Expanded memory)和扩充内存(Extended memory)这两个术语在目前的一些文章中混淆。扩展内存通过 INT 67H 与主内存进行通信;扩充内存通过 INT 15H 与主内存进行通信。

目前,一般的高档机都配有扩展内存和扩充内存(通过主机启动时的自检或用 PC TOOLS 工具,就可以知道这两部分内存的大小),使计算机内存达到了 1MB 或 2MB(不再局限于 512KB 或 640KB)。

如何才能把这两部分内存用起来呢?这是一个非常有价值的问题,也是用户十分关心的问题(例如,如果仍在 512KB 或 640KB 内存的基础上使用 CCDOS,则很多大型软件无法运行)。如果能把汉字库放到扩展内存或扩充内存,则任何个人机上的大型软件都能运行,且速度不受任何影响!

由于有关如何使用扩展内存和扩充内存的资料很少,致使这两部分宝贵的资源没能得到很好的应用(尽管有些用户能把扩充内存作为虚盘使用,但没有解决扩充内存与主内存的通信问题,用户也无法在自己编制的程序中使用扩充内存)。

最近,我带着这两个问题,查阅了一些资料,同时解剖分析了有关使用扩展内存和扩充内存的操作系统,比较全面地掌握了这两部分内存的使用方法,并用汇编语言编制了两个分别把汉字库放入扩展内存和扩充内存,然后用 INT 67H 和 INT 15H 调用汉字的程序。使用效果很好(速度极快)!下面就这两项技术,给用户一简单介绍(由于两个汇编语言程序实例较长,此文从略,需要的用户可来信索取。有关 INT 67H 和 INT 15H 调用口的使用方法,也请参阅我最新编写的 DOS2.0—DOS5.0/EGA/VGA/CEGA/CMGA《DOS/BIOS 功能调用及程序例》一书)。

二、把汉字库装入扩充内存(虚盘)的方法:

第一步:在 CONFIG.SYS 中加入以下命令:

DEVICE=VDISK.SYS abc/E

其中:a 为虚盘大小(单位为 KB)

b 为每扇区的字节数(可选为 512)

c 为目录项数(可设为 64 或 512)

/E 表示虚盘设置在扩充内存

例如:DEVICE=VDISK.SYS 384 512 64/E

第二步:重起动机(注意提示信息)后,用户就可以进入虚盘 D:

第三步:把汉字拷入虚盘 D:

第四步:运行从扩充内存(虚盘)取汉字的驻留程序(程序通过 INT 15H 功能调用口从扩充内存取也汉字的字模点阵,并将其送往内存)。

三、把汉字库装入扩展内存的方法

第一步:在 CONFIG.SYS 加入以下命令:

DEVICE=EMM.SYS OPTIONS

注:在不同的机型中,扩展内存管理程序名(EMM)有所不同,参数(OPTIONS)也有差异(可查阅不同机型的《DOS 使用技术手册》)。

第二步:重新启动机器。

第三步:确定扩展内存管理程序(EMM)是否存在(若 EMM 存在,则设备头的名字域(偏移 0AH 处)中应包含字符串“EMMXXXX0”。

下面提供的程序用于完成这一步的工作(程序附后)。

第四步:运行把汉字库装入扩展内存并从扩展内存取汉字的程序(程序运行后,即把汉字库驻留入扩展内存)。此后用户就可通过 INT 67H 功能调用口,从扩展内存取出汉字的字模点阵,并将其送往主

内存)。

四、结束语

由于篇幅所限,有关这两项技术的细节,请用户仔细兑读有待索取的源程序(程序附有详细的注释)。地址:南昌陆军学院计算机室(330103)戴水贵。确定扩展内存管理程序(EMM)是否存在的程序:

```
stack      segment stack' stack'
```

```
stack      ends
```

```
data       segment
```

```
emser      db  ' no install EMSI' ,24H
```

```
emmx       db  ' EMMno foundl' ,24h
```

```
emmer      db  ' emm state errorl' ,24h
```

```
corect     db  ' EMM corectl' 24h
```

```
emin      db  ' EMMXXXXXO'
```

```
data       ends
```

```
code       segment
```

```
assume     cs:code,ss:stack
```

```
start:push ds xor ax,ax
```

```
push ax
```

```
mov ax,data
```

```
mov ds,ax
```

```
mov es,ax
```

```
mov ax,3567h;get
```

```
int 21h
```

```
mov ax,es
```

```
or ax,bx
```

```
jz no _ emmx,no install,jmp
```

```
lea si,emm
```

```
mov di,bx
```

```
mov cx,3;emm len
```

```
repe cmpsb
```

```
jne no _ emm,no,jmp
```

```
mov ah,41h;get state
```

```
int 67h
```

```
or ah,ah
```

```
jnz emmst;error,jmp
```

```
lea dx,corect
```

```
exit:      mov ah,9
```

```
int 21h
```

```
mov ah,4ch
```

```
int 21h
```

```
no _ emmx: lea dx,emser;no install EMS
```

```
jmp exit
```

```
no _ emm:  lea dx,emmx;EMM no found
```

```
jmp exit
```

```
emmst:    lea dx,emmet;EMMstate error
```

```
jmp exit
```

```
code:ends end start
```