科海培训中心
系列教材

扩充个人计算机存储器的原理和方法

王志焜　赵瑞玉　编译

北京科海培训中心

扩充个人计算机存储器的原理和方法

王志焜　赵瑞玉　编译

北京科海培训中心

一九九〇年七月

前　言

目前在我国普遍使用 IBM 个人计算机及兼容机。这类计算机中所用的 CPU 为 Intel 公司的 8088, 80286, 和 80386。运行的操作系统为 DOS 版本及 CCDOS 中文系统。随着硬件技术的发展, 在 AT 机及 80386, 80486 系统中, 硬件配置特别是内存和外存容量越来越大。DOS 操作系统就很难发挥它们的作用。例如 DOS 操作系统只能管理 1MB 的内存, 而很多 AT, 386 机的用户都想使用更大的内存, 只靠运行 DOS 就不行了。本书就是专门介绍用户怎样扩大系统的内存容量和在自己程序中使用这部分内存的方法。

lotus / intel / microsoft 配合硬件技术提出一套扩充存贮器使用规范和 AST / Quadram / Ashton-tate 提出的 EEMS (增强扩充存贮器使用规范), 使个人计算机大容量内存的使用成为规范化和简单化。EMM 和 QEMM / 386 是扩充存贮器的驱动程序。它不仅有效的管理系统中 RAM 的扩大部分, 而且为用户提供使用这部份存贮器的简单接口。在本书中系统的说明了 EMS 和 EEMS 管理组织扩充存贮器的基本原理, 详细说明 30 个功能的目的和用途。并以多种编程语言形式给出使用这部分存贮器的编程实例。个人计算机用户可以按此规范编写自己的应用程序; 冲破 DOS 操作系统只能访问 640KB RAM 的限制。这是广大用户早已盼望得到的一个重大的编程技术。

本书可作为广大个人计算机程序员编写程序时的参考手册或高等学校有关专业师生参考书籍。

在本书编写过程中得到中国软件行业协会高档微机协会、科海培训中心的指导和帮助, 谨致以衷心的感谢。

目　　录

第一章　序言 …… 1

1.1 什么是扩充存贮器 …… 1
1.2 扩充存贮器的工作原理 …… 2
1.2.1 常规存贮器是怎样工作的 …… 2
1.2.2 扩充存贮器的工作过程 …… 2

第二章　如何编写使用扩充存贮器的程序 …… 3

2.1 程序应该作到的几点 …… 3
2.2 EMM 和 EEMS 介绍 …… 4
2.2.1 EMM 的基本功能简介 …… 4
2.2.2 EMMS 功能 …… 5
2.3 80386 的功能 …… 5
2.4 EMS 先进的编程技术 …… 5
2.4.1 保存映射硬件状态 …… 5
2.4.2 检索句柄和页计数 …… 6
2.4.3 映射和删除多个页 …… 6
2.4.4 重新分配页 …… 6
2.4.5 使用句柄或指定句柄名 …… 6
2.4.6 使用句柄属性 …… 6
2.4.7 备用的映射页和转跳／调用 …… 7
2.4.8 传送或交换存贮区 …… 7
2.4.9 得到可映射存贮器的容量 …… 7
2.4.10 操作系统功能 …… 7
2.5 先进功能介绍 …… 7
2.6 编程指导 …… 9
2.7 实例 …… 10
2.7.1 例 1 …… 10
2.7.2 例 2 …… 16
2.7.3 例 3 …… 27
2.7.4 例 4 …… 30
2.8 检查扩充存贮器的存在方法 …… 31

第三章 EMM 功能 …… 36

3.1 EMM 功能清单 …… 36
3.2 功能详述 …… 38
功能 1 得到状态功能 …… 38
功能 2 得到页窗口地址功能 …… 38
功能 3 得到未分配的页数功能 …… 39
功能 4 分配页数功能 …… 40
功能 5 映射／删除映射句柄页功能 …… 41
功能 6 删除分配页功能 …… 43
功能 7 得到版本号功能 …… 44
功能 8 保存页映射功能 …… 44
功能 9 恢复页映射功能 …… 46
功能 10 保留 …… 47
功能 11 保留 …… 47
功能 12 得到句柄计数功能 …… 47
功能 13 得到句柄页功能 …… 47
功能 14 得到全部句柄页数功能 …… 48
功能 15 得到／设置页映射功能 …… 50
功能 16 得到／设置部分页映射功能 …… 54
功能 17 映射／删除映射多个句柄页功能 …… 58
功能 18 重新分配页功能 …… 62
功能 19 得到／设置句柄属性功能 …… 63
功能 20 得到／设置句柄名字功能 …… 67
功能 21 得到句柄目录功能 …… 69
功能 22 改变页映射和转跳功能 …… 73
功能 23 改变页映射和调用功能 …… 75
功能 24 传送／交换存贮区功能 …… 79
功能 25 得到可映射的物理地址数组功能 …… 85
功能 26 得到扩充存贮器硬件信息功能 …… 89
功能 27 分配标准／原始页功能 …… 92
功能 28 备用映射寄存器组功能 …… 95
功能 29 为热启动准备扩充存贮器硬件功能 …… 107
功能 30 使用／禁止 OS／E 功能组功能 …… 108

第一章 序 言

1.1 什么是扩充存贮器

在使用 IBM-PC／XT,AT 及 386／PC 及其兼容机时,经常运行 DOS 2.0／3.3 操作系统．该系统的最大问题之一是对任何程序和它的数据所能占用的最大的存贮容量是640KB(包括 DOS 占用)．DOS 占用约为 60K．所以用户只能占 580KB．AT 及 386PC 本身的寻址能力(即可以扩充内存的能力)为 16MB．甚至可以扩到 32MB．这样就限制发挥这些存贮器的作用．为此,由 LIM 合作开发出一套存贮器管理程序 EMM,利用这个程序,用户可以使用地址在 1MB 以上的存贮器．在 EMM 中为了便于管理常常把它们系统中的存贮器分成三种类型的存贮器．

·常规存贮器(CONVENTIONAL MEMORY)

在地址范围为 0-1MB 内的 RAM 称为常规存贮器．在 DOS 环境下,所有的程序必须在此范围内运行．因为 DOS 2.0／3.3 仅仅承认和支持装在此区内的程序．

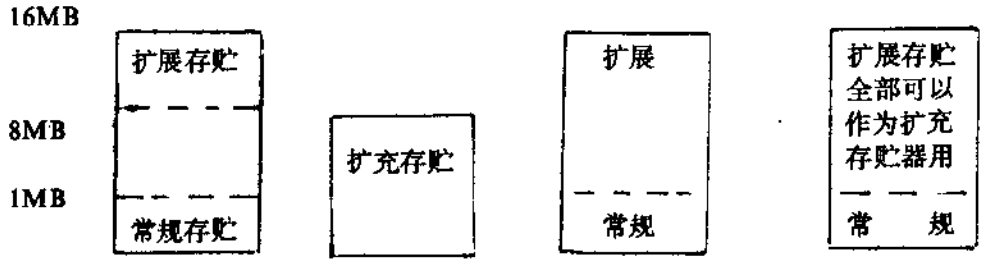
·扩展存贮器(EXTENDED MEMORY)

在地址为 1MB 以上范围内的 RAM．这个存贮器有一个重要的限制,在 DOS 2.0／3.3 环境下,不能为运行已经存在的程序使用,它只能作为 RAM DISK,打印机的脱机缓冲器和 DISK CACHE．

·扩充存贮器(EXPANDED MEMORY)

它是一个特殊类型的存贮器"SHADOW MEMORY"．它与上述两种存贮器不同,它没有一个固定的地址范围．换句话说,扩充存贮器的任何区间都可以立即代替同样范围的常规存贮器．通常用 MAPPED 的技术(映射,映象,变换)．目前有两类扩充存贮器,EMS 存贮器被用于存放程序的数据和作交换区．EEMS 存贮器除用于存放数据和作为交换区之外,还可以运行程序．

实际上,LIM(EMM)支持 32MB．在有 EMM 或 EEMM 支持下,存贮器的分配如下图所示．



PC-AT(286)　　无 QEEM-386　　有 QEEM-386

上图表示 0-1MB 范围是常规存贮器．0-640KB 用于运行程序．余下的 384KB 用

于系统备用.

1MB 以上是扩展存贮器. 在物理上,它将常规存贮器的范围引伸到 16MB 或 32MB. 但是在 DOS 2.0 / 3.3 环境下,它不能用于运行程序.

扩充存贮器是特殊的一类存贮器. 在 AT 机上扩充存贮器和扩展存贮器在物理上是不同的机构(实体). 这与所加的扩充存贮器板的硬件技术有关. 我们在图上用两个不同的分开图形表示. 而在 386 机上运行 QEMM(Quarterdeck Expanded memory manager) 386 时, 全部扩展存贮器均可作为扩充存贮器用.

1.2 扩充存贮器的工作原理

1.2.1 常规存贮器是怎样工作的

在地址范围为 0-1MB 之间的存贮器我们称它为常规存贮器. 0-640KB 可以运行程序, 640KB-1MB 范围作为保留备用存贮区. IBM-PC 是在 1981 年推出的. 当时所使用的 CPU 为 8088 芯片. 它的寻址范围只有 1MB. 所以 IBM-PC 的最早设计者把此寻址范围分成两个区, 512KB 为程序区,而为 ROM BIOS 保留了 512KB. 后来这些设计者又把保留区减少 128KB, 所以程序区增加到 640KB.

A000H-B000H(640-704K)---EGA 或 VGA ROM 和数据区

B000H-B800H(704-720K)---单色显示数据区

B800H-C400H ----EGA 或 CGA 使用

736K—752K-----CGA 数据区

768K—784K-----EGA ROM 区

C400H--D400H-----LIM 数据区(由扩充存贮器管理程序使用来管理存在 EMS 存贮器中的数据)

D400H--E000H-----没有使用

E000H--F000H----- 其余 ROM 用

F000H--10000H-----ROM BIOS

1.2.2 扩充存贮器的工作过程

如前所述, 扩充存贮器是一种特殊类型的存贮器. 但是你要想使它存在, 你必须在 AT 机上安装扩充存贮器板(LEO-286 已自动安装). 而 386 机扩展存贮器可以用 QEMM-386 转换成扩充存贮器.

系统一旦装有扩充存贮器之后, 它就被分成若干段, 每一段称为逻辑页. 每一页有 16KB. 例如, 我们有 384KB 的扩充存贮器, 它就被分成 24 个逻辑页, 系统访问扩充存贮器是通过 0-1MB 范围内的一个窗来实现. 也就是说, 计算机通过 0-1MB 内的一个物理块(称为 PAGE FRAME 页帧)实现. PAGE FRAME 包含多个物理页, 一个物理页也

是 16KB．物理页是在 0–1MB 的地址范围，所以计算机可以直接访问．

PAGE FRAME 作为访问扩充存贮器的一个窗口，例如 PAGE FRAME 选为保留区未使用的 64KB 窗口．那么扩充存贮器的任何 64KB 块(4 个逻辑页)就可以被映射到此窗口．这样操作系统就把对应的扩充存贮器认为是 PAGE FRAME 所占用的那段物理存贮器．当 CPU 读写窗口区数据时，它实际上是读写对应的扩充存贮器．我们把 PAGE FRAME 作为一个窗口，通过这个窗口能够看到与此窗口同样大小的扩充存贮器，如果想通过此窗口去看其它的部分，就必须告诉窗口移动观察点．然后才能看到你选择的新的扩充存贮器的范围．

扩充存贮器管理程序控制选定窗口．通过此窗口使用对应的扩充存贮器．所以一个系统在安装扩充存贮器时，都要选定一些固定参数，如系统安装的扩充存贮器容量，硬件决定的 PAGE FRAME 起始的段地址，扩充存贮器硬件使用的状态寄存器的口地址等．用户编写应用程序时，开始要告诉 EMM 想使用的扩充存贮器某一个与 PAGE FRAME 大小相同的区域，然后程序就可以向该区存放数据．下一步再告诉 EMM 程序想使用另外一个区域，而在第一个区内的信息，依然存在，不会被第二个区工作破坏．

EMM 将系统安装的扩充存贮器分成若干页(每页 16KB)．用户使用时，按用户程序指定的需要进行分配．这样的页我们称为逻辑页，EMM 同时还选定了一个 PAGE FRAME(窗口)，在 QEMM 中该窗口大小为 64KB．同样分成 4 个页，称为物理页．该窗口是在 0–1MB 范围内，通常由硬件决定在 640KB–1MB 之间，不为 ROM 所占用的连续区间．逻辑页和物理页都被连续依次编号，从 0 计起．这样实际上物理页只有 0.1.2.3 共 4 个页．而逻辑页数是按程序要求而定．

由上面所述，我们可以看出 EMM 首先构造了两个区，一个称为物理区，共 64KB 也称为窗口(PAGE FRAME)，另一个是逻辑区(扩充存贮器区)．通过对物理区的访问，实际上是访问扩充存贮器．因为操作系统可以承认物理区 PAGE FRAME 的地址范围．像对常规存贮器访问一样．要想去访问扩充存贮器，EMM 是按用户要求将 PAGE FRAME 区映射到逻辑区的某一段．如从 0 页开始，即物理页 0 映射到逻辑页 0．这时就可以通过窗口访问逻辑页(0–3)的 64KB 扩充存贮区(实际上，映射时物理 0 页对逻辑 0 页，物理 1 页对逻辑 1 页，物理 2 页对逻辑 2 页，物理 3 页对逻辑 3 页)．程序要想访问逻辑区的其它部分必须移动窗口，即移动"观察点"．具体办法就是重新映射，如把物理 0 页映到逻辑 4 页，物理 1 页映射到逻辑 5 页，物理 2 页映射到逻辑 6 页，物理 3 页映射到逻辑 7 页，这样我们就可以对扩充存贮器 4–7 页进行访问．由此可见，这种工作过程对用户来说是十分简单的．内部转换过程均由 EMM 完成．

第二章　如何编写使用扩充存贮器的程序

2.1　程序应该作到的几点

如前所述，扩充存贮器是一种特殊类型的存贮器．为便于用户使用，已经为用户开发

出这类存贮器的管理程序(EMM). 该程序中为用户设置了 30 个通过软中断 INT 67h 调用的功能. 用户就可以方便的使用它.

一个程序要使用扩充存贮器,必须要作到以下几点:

1. 决定系统是否安装了 EMM.
2. 决定为了使用是否有足够的扩充存贮器页存在.
3. 分配扩充存贮器页.
4. 得到 PAGE FRAME 的基地址.
5. 映射扩充存贮器页.
6. 读/写/执行在扩充存贮器中的数据.
7. 将不用的扩充存贮器页返回扩充存贮器区中. 再退出程序.

2.2 EMM 和 EEMS 介绍

EMM 是一个扩充存贮器的驱动程序,全名为 EXPANDED MEMORY MAHAGER 它提供一套符合 EMS(expanded memory speficasion)规范编定的一组接口程序,它为用户定义了很多可调用的功能,用户借助这些功能来使用扩充存贮器.

2.2.1 EMM 的基本功能简介

功能 1: 取状态功能

该功能返回一个状态码,指出存贮器管理程序硬件工作是否正常.

功能 2: 得到 PAGE FRAME 地址功能

该功能可以返回地址,该地址是 64KB PAGE FRAME 被分配的段地址,即窗口的起始段址.

功能 3: 得到没有被分配的页数.

它能返回未被分配的扩充存贮器逻辑页数(可以供你的程序使用的最大页数)并且也返回扩充存贮器中总的页数(包括未使用的和已被使用的).

功能 4: 分配页功能

给被请求的页分配 EMM 句柄号,并给这些页指定专用的 EMM 句柄.

功能 5: 映射/删除映射句柄页功能

该功能将任一个逻辑页映射到 PAGE FRAME 中的 4 个物理页中的任一个.

功能 6: 删除分配页功能

将现行已被分配的逻辑页返回给 EMM 管理程序,供下次使用.

功能 7: 得到版本号功能

返回 EMM 的版本号,该版本号是在由 INT 67h 中断服务程序入口段地址(偏移地址为 0)算起的第 10 个字节开始的连续 8 个字节中.

EMM 仅能在扩充存贮器区保存数据或作程序交换区,或保存"图形屏幕".

2.2.2 EEMS 功能

EEMS 是一个功能加强的扩充存贮器管理规范,它的英文全名为 enhanced EXPANDED MEMORY SPECIFICATION, 是由 AST, QUADRAM, ASHTON-TATE 等公司提出的标准, 它除具有 EMM 的功能之外,还能在扩充存贮器中运行程序. 与 EMM 有以下的区别:

1. PAGE-FRAME 窗口区(映射区)可以由程序设置和改变. 而在 EMS 中, 这个区所在地址是由硬件决定, 不能由程序设定和改变.

2. 可以有几个不连续段组成一个 MAP(映射)窗口区. 而 EMS 只能取连续的 64KB 区作为映射区. (page-frame)

3. 窗口(PAGE-FEAME)大小不受 64KB 限制, 可以选定 16K-1MB 范围的任意值.

2.3 80386 的功能

当使用 INTEL-386 PC 机时, 系统上安装的扩展存贮器可以全部充当扩充存贮器, 不须要附加任何其它硬件. 因为 80386 处理器的芯片内部已经有了必要的映射功能. 任何存贮器地址均可以映射到任何其它地址范围上.

要想使系统能安装上扩充存贮器, 你要作如下几步:

1. 在系统中安装扩展存贮器. 最好在系统上至少安装 2MB RAM 才能得到最好的效果.

2. 安装 QEMM-386 驱动程序, 并修改系统的 CONFIG.SYS 文件.

QEMM 386 能使 386PC 机的内部映射能力有效, 和使 386PC 100%与 LIM-EMS 存贮器标准兼容. 在 CONFIG.SYS 文件中增加一项.

```
DEVICE=QEMM.SYS
```

2.4 EMS 先进的编程技术

除前节所述的 EMS 基本功能之外, LOTUS / INFEL / MICROSOFT EMS 还提出了很多先进的功能, 它们增强了对扩充存贮器使用的能力. 但是在 EMS 早期版本中还不能支持下面所述的先进的编程技术(EMM 4.0 以上版本才支持).

2,4.1 保存映射硬件的状态

某些软件要求保留映射硬件的现行状态, 接通转换映射的环境. 扩充存贮器的控制部

分也要求存贮器映射硬件的原始环境，功能 8.9 和 15，16 能保存硬件状态(如中断服务程序，设备服务程序，常驻内存的程序都要保存映射硬件状态)。

2.4.2 检索句柄和页计数

某些实用程序需要保存如何使用扩充存贮器的过程，功能 12 和 14 就是完成这样的任务的。

2.4.3 映射和删除多个页

一次映射(或删除)多个页可以减少应用程序的开销，功能 17 可以使程序一次映射(或删除)多个页。

此外，还可以使用段地址而不用物理页来映射逻辑页。例如，如果窗口(PAGE-FRAME)基地址设在 D000H，你可以映射物理页 0 或段地址 D000H，功能 25(得到映射物理地址组)返回全部扩充存贮器的物理页和它们的对应段地址。

2.4.4 重新分配页

重新分配页(功能 18)让程序动态地分配扩充存贮器页，而不需知道它的句柄或得到没有被分配的句柄。为了得到或释放扩充存贮器，重新分配功能对于应用程序是非常有效的手段。

2.4.5 使用句柄或指定句柄名

所谓句柄，就是在程序中分配扩充存贮器时，EMM 为了管理方便给该组扩充存贮器编的代码。该代码标志了此次定义的该组扩充存贮器。EMM 规范让你把一个名字与句柄连起来，所以一组应用程序可以共享扩充存贮器中的信息，例如，由字处理和电子表格和脱机打印等程序组成的软件包，可以共享不同程序中的数据。脱机打印程序可以使用句柄名去使用电子表格或字处理程序放在扩充存贮器中的数据，并可检查在某个特定的句柄名的扩充存贮器页中的数据。

功能 20 (设置句柄名子功能)可以给 EMM 句柄指定名字，功能 21(检索有名句柄子功能)得到与名字相对应的句柄值。此外，你可以使用功能 14(得到句柄页)决定此 EMM 句柄中所包含的扩充存贮器的页数。

2.4.6 使用句柄属性

除给句柄命名之外，可以使用功能 19 将属性与句柄名联系起来。这些属性包括易失性和非易失性。非易失性使扩充存贮器页保存它的数据，在系统下次热启动时，不丢失原

存放的数据．如给句柄名赋与易失性，则它不能保存数据．对句柄来说默认值是易失性．

因为此功能决定于在系统中安装的扩充存贮器硬件性能，所以在使用之前先要用得到属性性能的子功能来验证系统是否具有此性能．

2.4.7 备用的映射页和转跳／调用

你可以使用功能 22(改变页映射和转跳)和功能 23(改变页映射和调用)去映射新的一组值给映射寄存器．使程序跳到规定的扩充存贮器中的地址去执行指令．所以这些功能能够用来在扩充存贮器中引导程序和执行程序．应用程序使用这些功能可以明显的减少对常规存贮器的要求．程序在运行时可以将需要的模块引导到扩充存贮器，并使用功能 22,23，将控制转跳到这些模块．

在扩充存贮器中保存代码可以改善程序的执行(在多方面)．例如，由于常规存贮器有限，所以有些程序需要将它们分成很多小的复盖段．由于 LIM EMS 4.0 版本可以支持 32MB 扩充存贮器，所以复盖段就可以很大．这种使用复盖的方法可以改善整个系统的性能，因为不需要很大的常规存贮器和消除了常规存贮器的分配错误．

2.4.8 传送或交换存贮器区

使用功能 24(传送／交换存贮器区)你能够容易地传送和交换在常规存贮器和扩充存贮器中的数据．在一次功能调用中可以管理交换 1MB 的数据．虽然应用程序不用此功能也可以完成此操作．但使用功能 24 可大大减少应用程序本身的开销．

此外，本功能还可以检查复盖区，完成所有必要的映射，保存交换／传送调用以前的运行环境．

2.4.9 得到可映射存贮器的容量

功能 25 使应用程序去决定现行的硬件／系统支持的可映射的存贮器的容量．不同的扩充存贮器板支持不同数量的物理页(映射区)．

得到可以映射的物理页地址入口子功能可以返回系统中所支持的物理页的总数．同时也可以返回物理页号与实际段地址所对应的值．

2.4.10 操作系统功能

除为应用程序所使用的功能外，本规范还为操作系统／工作进行环境规定了一些功能．一般情况下．这些功能均由操作系统／环境(OS／E)使其失效．

2.5 先进的功能介绍

功能 8: 保存页映射功能

将硬件上的页映射寄存器的内容保存到内部保存区.

功能 9: 恢复页映射功能

从内部保存区恢复指定的 EMM 句柄的页映射寄存器的内容.

功能 10: 保留

功能 11: 保留

功能 12: 得到句柄数功能

返回在系统中打开的句柄的数目.

功能 13: 得到句柄页数功能

返回指定的 EMM 句柄的页数.

功能 14: 得到全部句柄的页数功能

返回一组有效的 EMM 句柄和它们所对应的页数.

功能 15: 得到/设置页映射子功能.

保存或恢复所有可映射的存贮区(常规的和扩充的),将映射的现场保存在程序指定的数组中.

功能 16: 得到/设置部分页映射子功能

提供一种保存系统中指定可映射的存贮器的部分映射现场的措施.

功能 17: 映射/删除映射多个句柄页功能

在一次调用中,可以把系统能支持的物理页映射到同样数目的逻辑页.

功能 18: 重新分配页功能

可以增加或减少已被分配给句柄的扩充存贮器页数,

功能 19: 得到/设置句柄属性功能

允许应用程序给与指定句柄有关的扩充存贮器设置属性.

功能 20: 得到或设置句柄名功能

得到或设置句柄名,最多为 8 个字符.

功能 21: 得到句柄目录功能

返回有关有效的句柄信息及它们的名字.

功能 22: 改变页映射和转跳功能

改变存贮器映射的现场和把程序转向指定地址.

功能 23: 改变页映射和调用功能

改变指定的映射现场,并把程序转向指定地址. 并能在返回时恢复现场使程序返回到调用的下一个地址.

功能 24: 传送/交换存贮区功能

可以拷贝或交换存贮区. 其交换次序可以是从常规存贮器到常规存贮器,常规存贮器到扩充存贮器,扩充存贮器到常规存贮器,和扩充存贮器到扩充存贮器.

功能 25: 得到可映射的物理页地址功能

返回一组在系统中每一个可映射的物理页的页号和它们的段地址.

功能 26: 得到扩充存贮器硬件信息功能

返回一组包含扩充存贮器硬件性能的参数

功能 27: 分配标准 / 原始页功能

分配标准的容量或非标准容量的页, 由操作系统请求和指定特定的 EMM 句柄给这些页.

功能 28: 备用映射寄存器设置功能

应用程序可以仿真改变映射寄存器的硬件设置.

功能 29: 为热启动准备扩充存贮器硬件功能

为紧急热启动准备扩充存贮器的硬件

功能 30: 使能 / 失效 OS / E 功能

使操作系统的开发者可以使能或失效用于操作系统的功能.

2.6 编程指导

下面我们叙述一下有关程序员编制使用 EMM 的程序时所注意的几个问题.

1. 不能把程序中的堆栈区放在扩充存贮器中.

2. 用户不能在中断 67H 所在区放其它程序或自己编制 67H 的中断服务程序, 否则破坏了扩充存贮器管理程序的工作.

3. 不能交换到你的程序不能使用的常规存贮器区. 应用程序使用 EMM 去交换到常规存贮器区, 必须由操作系统先给程序分配使用区. 如果操作系统不知道它管理的存贮区是否使用, 那么就总认为是可用区. 这样可能破坏原区的数据. EMM 不能用来分配常规存贮器, 常规存贮器只能由操作系统来管理. EMM 只能与那些由 DOS 事先分配的常规存贮器交换数据.

4. 要在扩充存贮器中使用数据别名现象的应用程序必须检查扩充存贮器硬件是否存在. 当把一个逻辑页映射到两个或多个可映射的段时, 就会出现数据别名现象. 这使得一个 16KB 的扩充存贮器页出现在多于一个 16KB 存贮器地址空间. 数据别名现象是合法的. 并且某些场合下对应用程序还是非常有用的. 只是使用扩充存贮器仿真的软件是不能实现"数据别名"现象. 区别软件仿真和实际的扩充存贮器的硬件的方法是进行数据"别名现象"并检查结果. 例如, 将一个逻辑页映射到 4 个物理页上, 给 0 页(物理页)写数据并读物理页 1–3, 检查数据是否存在, 如果数据在 4 个页中均存在, 就证明在系统中安装的扩充存贮器硬件支持"数据别名"现象.

5. 应用程序要经常返回那些使用过的扩充存贮器的页数给扩充存贮器管理程序, 直到运行结束为止. 因为这些页对其它应用程序是有用的. 如果这些没有用的页不返回给管理程序, 系统就会报告超出扩充存贮器页数或扩充存贮器句柄错误.

6. 结束和存放存贮器常驻程序(TSR'S), 要经常在改变它们之前保存映射寄存器的状态. 因为 TSR'S 过程可能要中断其它程序, 而这些被中断的程序可能正在使用扩充存贮

器．这样不能改变它们的页映射寄存器的状态．在退出之前，TSR'S 必须恢复映射寄存器的状态．

下面讲述保存和恢复页寄存器状态的三种方法：

A. 保存页映射和恢复页映射(功能 8 和功能 9)．这是最简单的一种方法．EMM 在自己的数据结构中保存映射寄存器的内容．应用程序不须要为保存映射的现場提供存贮空间，保存最后一次映射现場．在指定的句柄下，就可以恢复该现場．这种方法有一个限制，对每一个句柄保存一个映射的现場，也只能保存 LIM 标准的 64KB 页窗口的现場．

B. 得到／设置页映射(功能 15)．这个方法要求应用程序为存放现場分配存贮空间(数组)．通过把存贮空间地址传送给 EMM 来保存现場．当用此方法恢复现場时，应用程序也要事先将存放区的地址传给 EMM．

这种方法比较好，特别适用于应用程序在恢复现場之前要保存多个现場时．此方法也提供了一种在多于一个映射现場之间的转换机构．

C. 得到／设置部分页映射(功能 16)它提供一种保存部分映射现場的方法．当应用程序不需要保存全部映射现場时，就使用这种方法，此方法也要求存贮现場的空间是应用程序数据区的一部分．

7. 使用指针指向数据结构的所有功能必须在没有被映射的存贮器内有这些数据结构．功能 22 和 23 是一种例外．

2.7 实例

下面列举出 4 个使用扩充存贮器的实际程序．

2.7.1 例 1

这个程序是用 MICROSOFT-C 语言版本 3.0 写的．EMM 功能调用是由 INT86 函数实现的(在 DOS.H 库中)．为了建立可执行的程序使用下面的编译命令行：

```
    msc / Gs / oat / MI program,,program;
#include<dos.h>
#include<stdio.h>
#define EMM__INT                          0*67 / * EMM interrupt number * /
#define GET__PAGE__FRAME                  0*41 / * EMM get page frame * /
                                             / * function number * /
#define GET__UNALLOC__PAGE__COUNT         0*42 / * EMM get unallocated * /
                                             / * page count * /
                                             / * function number * /
```

```
#define ALLOCATE_PAGES          0*43  /EMM allocate pages*/
                                      /*function number*/
#define MAP_PAGES               0*44 /*EMM map pages
                                     /*function number*/
#define DEALLOCATE_PAGES        0*45 /*EMM deallocate pages*/
                                    /*function number*/
#define DEVICE_NAME_LENGTH 8     /*length of device
                                     /*name string*/
#define TRUE                    1
#define FALSE                   0
union REGS input_regs, ouput_regs;
struct SREGS segment_regs;
int pf_addr;
/*_____________________________________________________*/
/*Routine to convert segment:offset pair to a far ptr.  */
/*_____________________________________________________*/
char *bulid_ptr(segment. offset)

unsigned int        segment
unsigned int        offset;
{
     char *ptr;
     ptr=(char *)(((unsigned long)segment<<16)+offset);
     retum(ptr);
}
/*_____________________________________________________*/
/*Function which determines whether the EMM device driver is installed.  */
/*_____________________________________________________*/
char    emm-installed()
{
   char *EMM_device_name="EMMXXXXO";
   char *int_67_device_name_ptr;
   /*_____________________________________________*/
   /*AH=DOS get interrupt vector function.        */
   /*_____________________________________________*/
input_regs.h.ah=0*35;
```

```
    /* ------------------------------------------------------------ */
    /* AL = EMM interrupt vector number.                            */
    /* ------------------------------------------------------------ */
     input_regs.h.al = EMM_INT;
     intdosx (&input_regs, &output_regs, &segment_regs);
     /* ------------------------------------------------------------ */
     /* Upon return ES:0Ah point to location where device name should be. */
     /* ------------------------------------------------------------ */
     int_67_device_name_ptr = build_ptr(segment_regs.es,0xA);
     /* ------------------------------------------------------------ */
     /* Compare memory with EMM device name.                         */
     /* ------------------------------------------------------------ */
       if(memcmp(EMM_device_name, int_67_device_name_ptr,
                                    DEVICE_NAME_LENGTH;= = 0)
            retum(TRUE);
   else
            retum(FALSE);
}
/* ------------------------------------------------------------ */
/* Function which determines if there are enough unallocated expanded */
/* memory pages for the application.                            */
/* ------------------------------------------------------------ */
char enough_unallocated_pages(pages_needed)

    int pages_needed;
{
    input_regs.h.ah = GET_UNALLOCATED_PAGE_COUNT;
    int86(EMM_INT, &input_regs, &ouput_regs);
    if (ouput_regs.h.ah! = 0 || pages_needed > output_regs.x.bx)
                return (FALSE):
    else
                return (TRUE);
}
/* ------------------------------------------------------------ */
/* Routine which allocates expand memory pages and passes       */
/* back to the main EMM handle.                                 */
/* ------------------------------------------------------------ */
```