廖彬山　甘登岱　刘有军　主编

PC
中断调用大全

科学技术文献出版社

PC 中断调用大全

廖彬山 甘登岱 刘有军 主编

科学技术文献出版社

(京)新登字 130 号

内容提要

本书详细介绍了 IBM 程序开发者所需要的系统调用，内容涉及 BIOS、MS－DOS 服务及 25 种主要的 API，以及各种常驻工具软件。对于每一个功能调用，本书均提供一些简明的描述及其它重要的信息。本书共包含六章和五个附录：中断调用概述、BIOS 中断调用、操作系统中断调用、应用程序中断调用、DOS 功能调用编程实例、设备 I/O 控制(IOCTL)编程实例、扩展内存规范参考和扩充内存规范参考。

本书内容丰富、资料新颖，既可作为广大计算机用户的培训教材，又可作为计算机软硬件开发人员的参考手册。

PC 中断调用大全

廖彬山 甘登岱 刘有军 主编

责任编辑 洪 宇

科学技术文献出版社出版

(北京复兴路 15 号 邮政编码 100038)

机电部情报所印刷

新华书店北京发行所发行 各地新华书店经售

*

787×1092 毫米 16 开本 30.25 印张 1024 千字

1993 年 6 月第 1 版 1993 年 6 月第 1 次印刷

印数：1－5000 册

科技新书目：287－069

ISBN 7－5023－1894－1/TP・100

定 价：30.00 元

前 言

本书详细介绍了 IBM 程序开发者所需要的系统调用，内容涉及 BIOS、MS－DOS 服务及 25 种主要的 API(应用程序界面)，以及各种常驻工具软件。对于每一个功能调用，本书均提供一些简明的描述及其它重要的信息。

本书包含如下内容：

- 中断调用概述　对中断类型(内中断、外中断和软中断)、DOS 专用中断、DOS 可调用中断、系统功能调用中断和中断调用编程作了概述。
- 详细介绍 CPU 所产生的各种异常中断及其产生的原因，并比较 8088、80186、80286、80386、80486 各种 CPU 产生异常的差异。
- 详细介绍各种 BIOS 的调用方式，包括 IBM BIOS、AMI BIOS、Phoenix BIOS 及其它的 BIOS。
- 详细介绍各种接口卡的调用方式，包括 MDA、CGA、EGA、VGA、PVGA 等各种不同的显示卡；此外，也包括 SCSI、Point device 的调用方式。
- 详细介绍操作系统的调用方式，从 DOS 1.0 到 DOS 5.0 均作了详细的说明，包括已公开、未公开及鲜为人知的功能调用。
- 详细介绍各种应用软件的 API 调用，包括 NetWare、Lantastic、DesqView、TopView、Pctools、Multasker 等数十种著名软件。
- 详细介绍各种软件的数据结构，包括 BIOS 内存信息、DOS 内部变量及表格等。
- 对 DOS 功能调用编程作了介绍，并给出具体实例。
- 对设备 I/O 控制(IOCTL)编程作了介绍，并给出具体实例。
- 在附录 A 对扩展内存规范作了介绍，内容包括 EMS 功能、EMS 错误信息和 EMS 编程要点。
- 在附录 B 对扩充内存规范作了介绍，内容包括 XMS 功能、XMS 错误信息和 XMS 编程建议。

本书内容不仅适用于 IBM PC/AT，从 PC、PCjr、XT 至 PS/2 及 Compaq、Notebook 等机型均可适用。在操作系统方面，不仅限于 MS DOS，也包含 PC DOS、OS/2 及其它各种多操作环境的系统。

本书内容充实，资料新颖，叙述全面，是国内第一本也是唯一一本系统介绍 PC 中断调用的书。它涵盖了不同层次的内容，因而可满足具不同开发需要的计算机软硬件开发人员。

本书由甘登岱、刘有军、廖彬山和高峰霞负责主编。此外参加本书编写工作的还有王彬、刘芸、林晓莉、曾志强、谢宁、张志伟、曾繁荣、王守江、傅伟、倪长华、王强、张彬、连红兵、杨晏文、孙志伟、朱斌、王建军、冯小军、潘伟、王云、高阳和夏铭。

作　者

1992 年 11 月

目 录

第一章　中断调用概述

中断是现代计算机发展中一个重要的技术，它能确保中央处理机在运行过程中随机地对外界发生的请求予以响应，在完成实时性的处理后又立即返回断点，继续执行刚才被挂起的过程。

为了提高响应中断的速度，现代计算机无一例外地采用了多级向量中断技术。向量中断的基本思想是，对应每一中断类型，在内存特定的位置上存放一个中断向量，该向量含有这种类型中断服务程序的入口地址。与此同时，对不同类型的中断赋予不同的优先级，即当几个中断同时提出请求时，CPU 优先响应级别最高的中断。

为有效地管理外部硬件中断，系统通常使用一个中断控制器（集成芯片）负责处理各级中断请求。这种处理机能包括：

（1）　对不同类型的中断源赋以固定优先级或循环优先级；

（2）　随机选择对某些中断级的开放或禁止；

（3）　当执行某级中断例程时，所有同级或低级的中断均被屏蔽，直至当前的例程运行完毕。但是，高一级的中断请求能中断正在执行的低级例程，这样就实现了多级中断的嵌套。除外部硬件中断外，系统还能响应内部硬件中断和软中断。

DOS 中断是以软中断的形式出现，这些中断均可供系统或应用程序调用。这些中断功能包括：

（1）　程序结束处理；

（2）　Ctrl—C 中止处理；

（3）　严重错误处理；

（4）　绝对磁盘读写处理；

（5）　系统功能调用；

（6）　假脱机打印处理。

前 3 种中断属于非直接调用的向量，但应用程序可利用系统功能调用中的若干功能建立用户自身控制的向量来取代 DOS 提供的向量，以便接管系统的控制权。在所有 DOS 中断里，最重要的一个中断向量是系统功能调用，它包含着近百个子功能供系统或应用程序调用。

1.1　中断类型

通常，一台微机系统将所有的中断类型分成三部分：

（1）　内中断（中断源是内部硬件）；

（2）　外中断（中断源是外部硬件）；

（3）　软中断（中断源是中断指令）；

在以 Intel　8086/8088/80286 为 CPU 的微处理机中，上述三类中断总共包含 256 个优先级中断。由于每个中断例程总是段间调用，故每个中断向量占有 4 字节地址（段址：偏移量）。因此，系统在内存最低端 0～3FFH 的 1K 字节里，设置了一张中断向量表，专门存放 0～255 个相应中断向量。

以下为这三类中断的布局：

中断级别

00H－08H　内部硬件中断（80286 要占用外中断部分）

08H－10H　外部硬件中断

10H－20H　ROM－BIOS 使用（设备 I/O 驱动程序）

20H－40H　DOS 中断（28～2EH，30～3FH 保留）

40H－　自由中断（供系统或应用程序使用）

1.1.1　内中断

内中断是由系统运行程序时，因硬件出错（如内存奇偶校验错，突然掉电）或某些特殊事件发生（如除数为零、运算溢出或单步跟踪）所引起的。

表 1－1 列出了这类中断的级别、向量地址、中断源以及上电初始化的中断向量。

表 1－1 初始化向量一栏中，D11 的名称是临时中断服务程序。在上电初始化期间，对于 ROM－BIOS 目前尚未提供实质性中断服务程序的那些中断向量，系统均以 D11 填入。该 D11 程序仅做某些例行处理。当系统进一步开发或使用用户自行编制的中断服务程序时，只需将其程序入口替代 D11 入口即可。

表 1－1　内中断向量一览表

中断号	向量地址	中断源	初始化向量
00H	00～03H	除数为零	D11(F000:FF23H)
01H	04～07H	单步跟踪	D11(F000:FF23H)

02H	08～0BH	不可屏蔽中断	NMI—INT(F000:F85FH)
03H	0C～0FH	断点	D11(F000:FF23H)
04H	10～13H	溢出	D11(F000:FF23H)
05H	14～17H	屏幕拷贝/BOUND 超界 *	PRINT — SCREEN (F000:FF54H)
06H	18～1BH	非法操作码 *	D11(F000:FF23H)
07H	1C～1FH	处理器扩展非法 *	D11(F000:FF23H)
08H	20～23H	双精度错 *	
09H	24～27H	段溢出 *	
0AH	28～2BH	非法任务段 *	
0BH	2C～2FH	段不存在 *	
0CH	30～33H	堆栈段溢出 *	
0DH	34～37H	存储保护错 *	
0EH	38～3BH	保留 *	
.		.	
.		.	
.		.	
1FH	7C～7FH	保留 *	

* 表示 80286 定义的内部中断源侵占 8088/8086 的外部中断区域，但 AT 机未使用，仍与 XT 机兼容。

1.1.2 外中断

外中断是由外部设备控制器提出实时中断请求而引起的。这些中断是可屏蔽的，或者由中断控制器设置屏蔽参数禁止指定的某些中断，或者直接使用禁止中断指令 CLI(关中断)禁止 CPU 响应所有外中断。

连接到中断控制器的中断请求线，是按系统设置的优先级依次与外设控制器相连的。因此，软件无法将其修改。在 AT 机上，使用 2 片 8259A 中断控制器芯片能支持 16 级外中断。

表 1—2 列出了 XT 机 8 级硬件外中断的向量地址及中断例程的入口地址。

表 1—2 XT 机 8 级外中断向量一览表

中断号	向量地址	中断源	初始化向量
08H	20～23H	定时钟	TIMER—INT(F000:FEA5H)
09H	24～27H	键盘	KB—INT(F000:E987H)
0AH	28～2BH	保留	D11(F000:FF23H)
0BH	2C～2FH	串行口 2	D11(F000:FF23H)
0CH	30～33H	串行口 1	D11(F000:FF23H)
0DH	34～37H	硬盘	HD—INT(C800:0760H)
0EH	38～3BH	软盘	DISK—INT(F000:EF57H)
0FH	3C～3FH	打印机	D11(F000:FF23H)

XT 机的中断控制器 8259 在上电初始化期间设置的初始化状态包括：

(1) 全嵌套方式。指中断优先级次序为 0→7，在处理某级中断时，屏蔽同级或低级中断，但可响应高一级中断请求。

(2) 非自动结束方式。中断处理完之后要向 8259 发中断结束 EOI 命令。

(3) 中断请求为边沿触发方式。当这类外部中断请求产生后，处理机一般要经过中断判优、中断响应、中断处理及中断返回等过程。

1.1.3 软中断

软中断是指在程序中执行一条中断指令 INT (或称软件自陷指令)而进入中断例程。

中断指令的形式是：

INT n

该指令一般由两字节组成：前一字节是指令的操作码(CDH)，后一个字节是中断类型码(或称中断号)。因它不受中断允许标志(I位)的影响，所以，当CPU在接受INT指令时，立即根据类型码n在中断向量表中找到该中断的入口地址，然后自动进入软件中断。

当CPU响应一个中断时，不论该中断是由INT指令强制产生的还是由硬件外中断请求INTR引起的，中断现场所做的工作是一样的：即将标志寄存器F、断点处CS和IP值依次进栈保存，同时清除标志位T(禁止跟踪)和标志位I(禁止中断)。换言之，不同类型的中断例程在接受控制时，其机器状态没有区别。因此，中断例程的第1条指令通常是STI(开中断)，它允许优先级别高的中断产生。

软中断指令非常类似于调用子程序指令CALL。除后者在转子程序前不对标志位做任何动作外，两者的主要区别是，软中断的调用无一例外是段间调用，因此，在其返回时必须使用中断返回指令IRET。IRET指令与RET的区别是前者要恢复标志位。

在XT机和AT机中，目前使用的软中断大致划分为三部分：

(1) 20H～3FH

该类中断为DOS占用。目前使用的有20H～27H和2FH，其余均保留。

(2) 10～1FH

该类中为ROM－BIOS使用。它主要供设备I/O驱动程序使用。当DOS或应用程序需与设备打交道时，该类中断提供了与设备的接口功能，它可由DOS－BIOS或应用程序直接调用，而不必关注设备硬件的具体细节。

(3) 40～FFH

该类中断称自由中断，供系统或应用程序设置开发的中断例程。目前，因PC兼容机繁多，故这类自由中断区使用不尽统一。表1－4描述的是原装XT和AT机的分配状态。

对于上述三类软中断，它们各自的向量地址、中断源及向量入口均列于表1－3和表1－4中。

表1－3 软中断一览表(ROM－BIOS和DOS)

中断号	向量地址	中断源	初始化向量
10H	40～43H	视频显示	VIDEO－IO(F000:F065H)
11H	44～47H	设备配置	EQUIPMENT(F000:F84DH)
12H	48～4BH	存储容量	MEMORY－SIZE(F000:F84H)
13H	4C～4FH	磁盘I/O	DISK－IO(C800:0256H)
14H	50～53H	串行I/O	RS232－IO(F000:E7C9H)
15H	54～57H	盒带I/O	CASSETTE－IO(F000:F859H)
16H	58～5BH	键盘I/O	KEYBOARD－IO(F000－E82EH)
17H	5C～5FH	打印机I/O	PRINTER－IO(F000:EFD2H)
18H	60～63H	ROM BASIC	F6000(F000:6000H)
19H	64～67H	系统自举	BOOT－STRAP(C800:0186H)
1AH	68～6BH	日时钟I/O	TIME－OF－DAY(F000:FE6FH)
1BH	6C～6FH	CtrlBreak处理	DUMMY－RETURN(F000:FF4BH)
1CH	70～73H	定时器控制	DUMMY－RETURN(F000:FF4BH)
1DH	74～77H	显示器参数*	VIDEO－PARMS(F000:F0A4H)
1EH	78～7BH	软盘基数*	DISK－BASE(F000:EFC7H)
1FH	7C～7FH	图形字符扩展*	(F000:0000H)
20H	80～83H	程序终止退出	00EB:0B07H
21H	84～87H	系统功能调用	0542:0180H
22H	88～8BH	程序结束处理	06AF:0242H
23H	8C～8FH	Ctrl－C处理	06AF:0272H
24H	90～93H	严重错误处理	0542:04E2H
25H	94～97H	绝对磁盘读	00EB:13E0H
26H	98～9BH	绝对磁盘写	00EB:142EH
27H	9C～9FH	程序结束常驻	00EB:2713H
28～2EH	A0～BBH	DOS保留	
2FH	BC～BFH	假脱机打印	

30～3FH	C0～FFH	DOS 保留

*向量内容仅作入口指针使用，不直接执行中断。

表 1—4 软中断一览表(自由中断区)

中断号	向量地址	中断源	初始化向量
40H	100～103H	软盘 I/O	DISKETTE—IO (F000:EC59H)
41H	104～107H	硬盘基数*	FD—TBL(C800:03F7H)
42～45H	108～17FH	系统保留	
60～6FH	180～1BFH	用户保留	
70H	1C0～1C3H	IRQ8(实时钟中断)	RTC—INT**
71H	1C4～1C7H	IR19(INT 0AH 中断)	RE—DIRECT
72H	1C8～1CBH	IRQ10(保留)	D11
73H	1CC～1CFH	IRQ11(保留)	D11
74H	1D0～1D3H	IRQ12(保留)	D11
75H	1D4～1D7H	IRQ13(80287 中断)	INT—287
76H	1D8～1DBH	IRQ14(硬盘中断)	HD—INT
77H	1DC～1DFH	IRQ15(保留)	D11
78～7FH	1E0～1FFH	未用	
80～85H	200～217H	BASIC 保留	
86～F0H	218～2C3H	BASIC 解释器用	
F1～FFH	2C4～3FFH	未用	

*向量内容仅作入口指针使用，不直接执行中断。

**该 8 个中断向量在 AT 机提供给硬件外中断使用。

1.2 DOS 专用中断

上节叙述的 DOS 中断是指 DOS 提供的软中断形式，即 20H～27H 和 2FH 的中断类型。这 9 类 DOS 中断大致分成三部分：

(1) DOS 专用中断

它指 INT 22H 、INT 23H 和 INT 24H 3 个中断。这类中断是供 DOS 操作时专用的，用户不要直接调用。

(2) DOS 可调用中断

它指 INT 20H、INT 25H、INT 26H 、INT 27H 和 INT 2FH 等 5 个中断，但在调用时，需满足各自特殊的入口要求。

(3)系统功能调用中断

它仅指 INT 21H。这是一个极其重要的软中断，DOS 的内核即是由它组成的。当系统或应用程序调用时，只需给出指定的子功能号和相应的入口参数。本节仅叙述 DOS 专用中断的 3 种不同中断的功能及特点。

中断类型 22H、23H 和 24H 的共同点是：在一程序(.COM 或.EXE)被加载到内存但尚未执行之前，这三个向量分别送入程序段前缀 PSP 的位移 0AH、0EH 和 12H 处保存；在程序终止时，作为 DOS 处理结束操作的一部分，将这三个向量从 PSP 恢复到中断向量表中，其目的是允许用户的应用程序自行开发并替代这 3 个中断例程。

1.2.1 程序结束处理

如前所述，一个应用程序(.COM 或.EXE 文件)执行终止返回 DOS 时，可使用 INT 20H 或 INT 21H 的子功能 00H、31H 或 4CH 等方式。当由其中之一完成 DOS 的结束操作后，均转到 INT 22H 向量。该中断例程是实施返回 DOS 的最后操作，包括：

(1) 程序返回到 COMMAND.COM 的常驻区，出现 DOS 提示符 >；

(2) 若 COMMAND.COM 的暂驻区被覆盖，则执行暂驻区的装入程序，将暂驻部分从磁盘上加载到内存高端，再返回常驻区，出现提示符>。

(3) 若程序返回到批文件处理程序，则接着解释批处理文件的下一行批命令并执行之，直至处理完毕显示提示符>。

1.2.2 Ctrl—C 处理程序

当一个应用程序执行字符 I/O 操作时(例如等待键盘输入或从键盘、串行口等设备输入字符流)，DOS 一旦检测到 Ctrl—C(03H)，即转入执行 INT 23H 的处理程序。通常，该处理程序将终止当前的程序流程，

而把控制权返回到命令处理程序 COMMAND.COM。

DOS 提供的这一功能便于用户随机地中止一个执行错误或不必继续执行的程序。但它带来的副作用也是明显的：一旦一个意外的误动作发生，Ctrl—C 键将强制正在正常执行的程序中断退出。此时，用户修改过的中断向量可能得不到正确恢复，而引起某些直接的 I/O 操作（如串行口中断驱动）出现预料不到的错误。利用某些系统功能调用的子功能，能屏蔽 Ctrl—C 对字符 I/O 操作的影响，但不能完全消除 Ctrl—c 的作用。一种较稳妥的办法是，让用户应用程序接管中断 23H 的向量，即用一段代码去取代系统提供的 INT 23H 。这段代码对 Ctrl—C 的处理不予理睬，以避免该键对应用程序的打扰。为使应用程序在键入 Ctrl—C 时不至失去系统的控制权，应在程序被加载执行的初期建立中断 23H 新的向量。这要用到系统功能调用的子功能 25H。下面，提供的代码段是通常接管向量的模式。

```
MOV   AH,25H                          ;调用设置中断向量
MOV   AL,23H                          ;指定中断号
MOV   DX,SEG Brk—Routine              ;
MOV   DS,DX                           ;指定向量段址
MOV   DX,OFFSET  Brk—Routine          ;指定向量位移
INT   21H
        .
        .
        .
Brk—Routine PPOC   FAR                ;建立新的处理程序
        .
        .
        .
IRET                                  ;中断返回
Brk—Routine    ENDP
```

除 INT 23H 处理程序能中止当前执行的程序外，ROM—BIOS 还提供了中断 1BH 的类似功能。该中断是专门处理 Ctrl—Break 键的。与 INT 23H 不同之处在于：它是通过硬件中断（IRQ1）来实现的。当按键后产生键盘中断，使 CPU 转而执行向量地址为 24H～27H 的中断例程（KB—INT）。该例程对 Ctrl—Break 的处理过程如下：

```
F000:EB09    CMP   AL,SCROLL—KEY     ;AL=70H
             JNE   K39                ;非 Break 键
             MOV   BX,BUFFER—START    ;是 复位缓冲区
             MOV   BUFFER—HEAD, BX    ;头指针=尾指针
             MOV   BUFFER—END,BX
             MOV   BIOS—TAIL,80H      ;置 Break 位
             INT   1BH                ;转中断 1BH
             SUB   AX—AX              ;置扩展字符
             JMP   K57                ;存入缓冲区
```

一旦检测到 Ctrl—Break，便转入中断 1BH 的例程。但该例程什么事情也不做，只是执行一条 INET 指令返回而已。

为何 Ctrl—Break 也能起到 Ctrl—C 相同的作用呢？由以前的知识得知，DOS—BIOS 初始化时，重新设置了 1BH 的向量，即用 0070:0140H 取代了原来的 F000:FF4BH。这样，当键入 Ctrl—Break 时，CPU 便转而执行 DOS 提供的中断 1BH 的例程。但应注意，该例程只是设置一个标志位后便中断返程。在其后的 DOS 检测时，由于这个标志位被置位而使 Ctrl—Break 得到了与 Ctrl—C 同样的处置，即由 INT 23H 处理程序强制应用程序中止退出。

为避免 Ctrl—Break 键对应用程序的干扰，用户也可自行开发程序来用新的向量替换原向量，但在使用中应考虑到它是通过硬件中断产生的。因此，在 CPU 进入该中断例程（本身是软中断形式）时，DOS

可能已执行了一段关键的代码，结果，除断点现场CS:IP被保存外，其余的寄存器都是未知的，同时，当前的堆栈深度也是不可知的。

对此，用户在编制中断1BH例程时，应注意以下几点：

(1) 在执行中断处理之前，应把所有寄存器进栈保护，在退出时再恢复之；

(2) 因键盘中断级别较高(IRQ1)，故在该例程中要尽早发出STI命令，使系统开放中断；

(3) 当该例程要完成较多的堆栈操作时，应事先将SS和SP保存在代码段能寻址到的变量中，然后将堆栈指针向具有足够深度的堆栈空间。

(4) 与中断23H不同，该例程在返回前，应使用8259中断结束命令EOI，使之退出后能接受下一个键盘中断请求。

(5) 修改的1BH向量在程序退出时是不会被DOS自动恢复的，故在应用程序进入时应保存1BH向量的原内容，而在退出前予以恢复。

下面，我们用汇编语言编制一个BREAK.ASM程序来替代原INT 23H和原INT 1BH的功能。当系统运行用C语言编制的程序TRYBREAK.C时，该程序在Ctrl—C或Ctrl— Break的作用下仍保持对系统的控制。

```
;汇编语言程序 BREAK.ASM
    NAME   BREAK
    ARGS   EQU  4                    ;变量位移
    CR     EQU 0DH                   ;回车
    LF     EQU  0AH                  ;换行
_TEXT SEGMENT BYTE PUBLIC'CODE'
    ASSUME CS:—TEXT                  ;定义代码段
    PUBLIC _CAPTURE,_RELEASE         ;定义连接使用的过程名
_CAPTURE PROC HEAR                   ;C 语言程序调用的过程
    PUSH   BP                        ;保存原 BP
    MOV    BP,SP                     ;BP 指向堆栈接口
    PUSH   DS                        ;保存原 DS
    PUSH   DI                        ;保存原 DI
    PUSH   SI                        ;保存原 SI
    MOV    AX,WORD PTR [BP+ARGS]
    MOV    CS:FLAG,AX                ;从堆栈接口取变量地址送至标志单元，该
    MOV    CS:FLAG+2,DS              ;变量地址由 C 程序调用时压栈
    MOV    AX,3523H                  ;取中断 23H 的原向量
    INT    21H
    MOV    CS:INT23,BX               ;保存到本程序设置的变量(4 字节)
    MOV    CS:INT23+2,ES
    PUSH   CS                        ;DOS 指向代码段
    POP    DS
    MOV    DX,OFFSET CTRLBRK
    MOV    AX,2523H                  ;将开发的处理程序入口
    INT    21H                       ;替代中 断 23H 向量以及中断 1BH 向量
    MOV    AX ,251BH
    INT    21H
    POP    SI                        ;恢复寄存器
    POP    DI
    POP    BP
    RET    2                         ;返回 C 语言程序
_CAPTURE ENDP
_RELEASE  PROC  NEAR                 ;C 语言程序调用的过程
    PUSH   BP                        ;保存原 BP
    MOV    BP,SP                     ;BP 指向堆栈接口
```

```
    PUSH   DS                          ;保存 DS
    PUSH   DI                          ;保存 DI
    PUSH   SI                          ;保存 SI
    MOV    DX,CS:INT1B                 ;恢复中断 1BH 原向量
    MOV    DS,CS:INT1B+2               ;原向量保存在变量 INT1B 中
    MOV    AX,251BH
    INT    21H
    MOV    DX,CS:INT23                 ;恢复中断 1BH 原向量
    MOV    DS,CS:INT23+2               ;原向量保存在变量 INT23 中
    MOV    AX,251BH
    INT    21H
    POP    SI                          ;恢复寄存器
    POP    DI
    POP    DS
    POP    BP
    RET                                ;返回 C 语言程序
_RELEASE    ENDP
;
CTRLBRK PROC FAR                       ;新的 Break 处理程序
    PUSH   BX                          ;保存BX
    PUSH   DS                          ;保存 DS
    MOV    BX ,CS:FLAG                 ;取标志变量地址
    MOV    DS,CS:FLAG+2
    MOV    WORD PTR DS:[BX]            ;将标志置"1"
    POP    DS                          ;恢复寄存器
    POP    BX
    IRET                               ;中断返回
CTRLBRK ENDP
FLAG   DW 0,0                          ;保存标志变量地址
INT23 DW 0,0                           ;保存中断 23H 原向量
INT1B DW 0,0                           ;保存中断 1BH 原向量
_TEXT ENDS                             ;代码段结束
      END                              ;程序结束标志
```

上面的汇编程序 BREAK.ASM 包括两个被 C 语言程序 TRYBREAK.C 调用的近过程_CAPTURE 和_RELEASE 以及一个取代原键盘中止功能的新处理程序 CTRLBRK（远过程）。它们各自的功能如下：

(1)_CAPTURE 过程

该过程由 C 语言程序调用，其功能是建立 DOS 和键盘驱动程序新的 Ctrl－Break 中断向量（1BH 和 23H）。

假设 C 语言程序定义一个整形变量 FLAG。当 C 语言程序调用该过程时，按照两者调用的接口约定，该变量的地址保存在堆栈中。这样，进入该过程后，就首先从堆栈中取出 FLAG 变量的地址存放到汇编所定义的变量 FLAG（4 字节）中，以供新的 Ctrl－Break 处理程序使用。

为建立新的 Ctrl－Break 中断向量，该过程然后将原向量内容保存（利用 35H 子功能），接着，在同一个向量地址上设置新的向量（利用 25H 子功能）。

(2)_RELEASE 过程

该过程由 C 语言程序调用，其功能是恢复 DOS 和键盘驱动程序原来的 Ctrl－Break 中断向量。

当应用程序终止退出时，虽然系统会自动恢复 INT 23H 的原向量，但此过程可保证在程序不结束情况下，复原 Ctrl－C 的缺省操作。

恢复的流程很简单，只需使用两次 25H 的子功能即能将原向量送入中断向量表的相应位置。

(3)CTRLBRK 过程

该过程是实用的 Ctrl－Break 中断处理例程，当 ROM－BIOS 的键盘驱动程序检测到 Ctrl－Break 键或 DOS 检测到 Ctrl－C 键时，它就被调用。

该过程仅做一件事：把 C 语言程序定义的变量

FLAG 置 1(该变量的地址已由_CAPTURE 过程从堆栈接口取出保存到汇编定义的变量 FLAG 中)。

C 语言程序调用前两个过程的格式如下：

```
STATIC INT FLAG=0;
CAPTURE (&FLAG);
  或
RELEASE();
```

下面是 C 语言程序 TRYBREAK.C。该程序经编译后可直接与上面的汇编目标模块连接在一起。当执行 TRYBREAK 时，任意时刻键入 Ctrl—C 或 Ctrl—Break 都不会使程序失去控制。当按下 Esc 键时，即退出该程序返回 DOS。程序清单如下：

```
/* TRYBREAK.C */
#INCLUDE<STDIO.H>
MAIN(ARGC,ARGV)
INT ARGC;
CHAR *ARGV[];
{
INT HIT=0;                    /*按键标志位*/
INT C=0;                      /*来自键盘字符*/
STATIC INT FLAG=0             /* 检测到 Ctrl—C 或 Ctrl—Break 时被置成 1*/
PUTS("\n  * * * TRYBREAK.C  running * * * \n");
PUTS("\n Press Ctrl—C or Ctrl—Break to test handler,");
PUTS("Press the ESC key to exit TRYBREAK.\n");
CAPTURE(&FLAG);               /*传递 FLAG 的地址 */
PUTS("\n TRYBREAK  has CAPTURED interrupt Vector s.\n")
                              /* 显示已建立的新中断向量*/
WHILE ((C&127) ? =27)         /*键入 Esc 吗? */
  {
  HIT=KBHIT ( );              /* 检测按键标志*/
  IF (FLAG? =0)               /*FLAG 置成 1 吗? */
    {
    PUTS ("\n Ctrl—Break detected.\n")
    FLAG=0;                   /* FLAG 复原为 0*/
    }
  IF(HIT? =0)                 /* 未中断，准备读键*/
    {
    C=GETCH ( )
    PUTCH (C) ;               /*显示键符*/
    }
  }
RELEASE( ) ;                  /* 按 Esc 后到此，复原向量*/
PUTS("\nTRYBREAK has RELEASEd interrupt  Vectors.\n");
}
```

通过上例的分析可知，由于应用程序建立了自身的 INT 23H 处理程序，因而对于任何键符的输入，程序都不会失去对系统的控制。

当检测到 Ctrl—C 或 Ctrl—Break 并使用户的 INT 23H 处理程序接受控制时，所有寄存器被置为功能调用被中断时的原有值。用户开发的 INT 23H 处理程序可选做下面的任一操作。

(1) 为便于应用程序作后面的检查，设置一局部标志(如上例使 FLAG=1)，随后恢复保存的全部寄存器，以中断返回(IRET)，将控制转到 DOS。DOS 功能将从查找开始，然后完成相应的动作，最终以正常方式使控制转到应用程序。

(2) 进入处理或采取其它合适的动作后，以远程返回使控制返回 DOS，DOS 根据进位标志是否被置而采取相应的动作。如果 C=1，则放弃应用程序，否则，以正常方式继续进行。

(3) 由应用程序的错误处理子程序获得控制，随后恢复执行或采取适当的动作，但不需用 IRET 或 RET 远程返回来结束中断处理例程。这样做并不会破坏系统的运行。在用户开发的 INT 23H 处理程序中可使用系统功能的任一个子功能，这些功能可用在所有运行 DOS 的各种微机上。但是，由硬件中断产生的 INT 1BH 是不能重入 DOS 的，而且在某些与 IBM－PC 不兼容的微机上不能运行。

1.2.3 严重错误处理程序

在 DOS 运行的环境下，当发生不可恢复的硬件错误时，DOS 便进入中断 24H 向量。系统提供的严重错误处理程序主要显示错误类型(磁盘类错或非磁盘类错)和下面的提示：

Abort，Retry，Ignore?

用户若忽略此错误，则按"I"，以通知 DOS 认定本次操作成功继续做下一步；若要求重试一次，则按"R"；否则按"A"，以通知 DOS 从应用程序中退出，即 DOS 转向 INT 23H 终止程序运行。

当进入 INT 24H 程序时，DOS 是根据下述寄存器提供的信息来判别错误原因的：

AH 的最高位为 0 是磁盘类错，否则是非磁盘类错。

DI 的低 8 位为设备驱动程序返回的错误代码：

00H　写保护错
01H　未知部件
02H　驱动器未准备好
03H　非法命令
04H　数据错(CRC 校检错)
05H　驱动器请求结构长度错
06H　寻道错
07H　未知的介质类型
08H　扇区未找到
09H　打印机无纸
0AH　写失败
0BH　读失败
0CH　一般性错误
0D－0EH 保留
0FH　盘更换无效

上述错误代码与设备驱动程序"请求头"中的返回状态字的错误代码相同。

BP. SI 指示设备驱动程序的"设备头"段址和偏移。

此外，若是磁盘类错，则 DOS 在转入 INT 24H 向量之前要重试 3 次。若结果仍错，则 AH 寄存器还提供如下的附加信息：

AH 的最低位(0 位)为 1 表示写盘错，否则是读盘错，且第 2 位和第 1 位表示盘错的位置。

00　DOS 区域
01　文件分配表
10　磁盘目录
11　文件区

对于非磁盘类错误，为确定严重错误是否由字符设备所引起，可由 INT 24H 利用 BP：SI 的指针，来检查设备驱动头的偏移 04H 处的设备属性字。如果该字的最高位(15 位)为 1，则表示错误是由字符设备产生的，检查设备头的偏移 0AH 处的设备名域可确定错误的设备名。

INT 24H 处理完毕和用 IRET 返回之前，在 AL 中设置一个动作解释码传递给 DOS。AL 中的代码含义是这样的：

00H　忽略该错误
01H　再试一次操作
02H　终止退出
03H　放弃失败的功能调用(仅对版本 3)

一般情况下，用户在应用中如果出现下述动作之一，则 DOS 会根据 INT 24H 在 AL 中返回的动作解释码在屏幕上给出提示信息：

(1) 企图打开一个磁盘驱动器上的文件，但该驱动器内不包含软盘，或驱动器门未关上；

(2) 企图读一个 CRC 错的扇区；

(3) 企图在打印机处于脱机状态下打印。

这里应注意的是，INT 24H 接受控制是在 DOS 对设备提出读写请求时，因设备硬件错而引起的，并非使用 DOS 命令不当而产生的其它错误所致。

表 1－5　严重错误类型及原因

错误类型	含义	错误代码(DI)	错误原因
Bad call format	错误调用格式	05H	传给驱动器的设备请求头长度错
Bad commad	命令错	03H	驱动程序向设备发了一条非法命令
Bad unit	设备号错	01H	驱动程序传送未知的部件号
Data	数据错	04H	磁盘上有坏扇区，DOS 无法正确读写
FCB unaVailable	文件控制块不够		网络装入共享任务，企图打开

			的文件数超出 FCBS 命令的规定数
General failure	一般性错误	0CH	遇到别处未指明的一般性错误
LOCK Violation	非法锁定		网络共享命令 share 企图访问被锁定的文件
No paper	打印机无纸或脱机	09H	打印纸用毕或打印机未联机
Non－Dos disk	非 DOS 盘		文件分配表中有非法信息，需重新格式化
Not ready	设备未就绪	02H	设备尚未处于接受或传送数据的状态
Read fault	读失败	0BH	DOS 不能从设备上读，磁盘未插或盘片坏
Sector not found	扇区未找到	08H	指定的扇区在盘上无法定位，盘片可能坏
Seek	磁道定位错或盘片坏	06H	无法定位指定的磁道，驱动器
Sharing violation	非法共享		网络用户非法访部共享文件
Writefault	写失败	0AH	DOS 无法将数据写入设备，驱动器或盘片坏
Write protect	写保护	00H	企图在写保护盘上写入，或在单面驱动器上使用双面盘，或在 DOS 版本 1 下使用每道 9 扇区的盘片

一旦检测到错误，DOS 总是用下列格式显示出错信息：

<type> error reading <device>
Abort，Retry，Ignore?
或者
<type> error writing <device>
Abort ，Retry，Ignore?

上述信息中，<device>是出错设备名，例如 PRN（指打印机）或 B：（指驱动器），而<type>是表 1－5 列出的错误类型。

从上述对 INT 24H 处理程序的功能分析可知，一旦检测到某设备的读写错，系统就会给出提示信息等待用户响应，在用户响应之后，系统才继续工作。

通常最先选择的响应是按 R 键，表示要求重试一次操作。当选择 I 时，由于越过出错条件继续运行，可能会丢失某些数据，故要特别小心。而在不得已情况下按 A 键时，系统会立即终止应用程序运行而退出。由于被迫中途停止，故对应用程序先前做的工作无法作善后的处理，从而可能会导致意料不到的结果。例如，处于过渡状态的文件仍留在盘上，修改的文件可能未被正常关闭而使目录项不能正确反映扩展后的文件长度，或者中断向量被修改后因未能恢复而使随后的 DOS 操作处于未知的状态等。

如接管 Ctrl－C 的中断向量一样，以上情况都要求用户的应用程序修改严重错误中断向量，使之指向自行开发的错误处理程序。同样，利用系统功能调用的 25H 子过程可替换 INT 24H 向量，但无需复原，因为当应用程序退出返回 DOS 时，DOS 会自动将保存于 PSP 位移 12H～15H 的内容复制到 INT 24H 向量中。

下面给出应用程序接管严重错误向量的程序模式。

```
CODE    SEGMENT CODEUBLIC
        .
        .
        .
        MOV   AH,25H                ;设置中断向量
        MOV   AL,24H                ;中断 24H
        MOV   DX,SEG Crit－Err
        MOV   DS,DX                 ;DS:DX 指向新向量
```

```
        MOV   DX,OFESET Crit—Err
        INT   21H                          ;系统功能调用
          .
          .
          .
Crit—Err:STI                               ;当设备发生严重错误时转此.开中断,保存
                                           ;使用的寄存器
        PUSH DS
        PUSH ES
        PUSH BX
        PUSH CX
        PUSH DX
        PUSH CS
        PUSH DS
        MOV   CS:SSEG,SS                   ;SS 保存在变量中
        MOV   CS:STOP,SP                   ;SP 保存在变量中
        TEST AH ,80H                       ;检验错误类型
        JZ Disk—Err                        ;最高位=0 为磁盘错
        MOV   ES,BP                        ;检查是否字符设备错
        TEST WORD PTR ES:[SI+4],8000H
        JNZ   Char—Err                     ;设备属性最高位=1 为字符设备错,否则为其它错
        JMP   Gene—Err             ;
  Disk—Err:
        .                                  ;磁盘类错到此
        .                                  ;检查错误原因
        .
        JMP   Crit—Exit                    ;转出口
Char—Err:                                  ;字符设备错到此
        .                                  ;检查错误原因
        .
        .
        JMP   Crit—Exit                    ;转出口
Gene—Err:                                  ;其它错到此
        .
        .
        .                                  ;检查错误原因
Crit—Err:
        .                                  ;出口时,返回错误类型和 AL 动作解释码
        .
        .
        MOV   SP,CS:STOP                   ;恢复 SP
        MOV   SS,CS:SSEG                   ;恢复 SS
        POP   DX                           ;恢复寄存器
        POP   CX
        POP   BX
        POP   ES
        POP   DS
        IRET                               ;中断返回
        .
```

```
        .
        .
SSEC    DW    0                    ;保存 SS 的变量
STOP    DW    0                    ;保存 SP 的变量
        .
        .
        .
CODE ENDS
END
```

在编制用户开发的严重错误处理程序时，应注意以下几点：

(1) 入口时，要保存 DS、ES、SS、SP、BX、CX 和 DX 各寄存器，并在退出前(IRET)依次恢复相应的寄存器；

(2) 在该程序中只能使用系统功能调用(INT 21H)的 01H～0CH 的子功能，即只允许字符设备的操作。如果调用其它子功能，则会破坏 DOS 的栈区以及影响它重试(R)或忽略错误(I)的能力；

(3) 如果该程序退出时不返回 DOS 而是返回用户应用程序，则随后使用字符 I/O 功能时会导致 DOS 处于不稳定状态。直至使用一个高于 0CH 的子功能调用后方可恢复正常运行。

由此可见，用户应用程序接管中断 24H 向量比接管中断 23H 向量要受到更为严格的限制。

1.3 DOS 可调用中断

除 DOS 专用中断(22H、23H 和 24H)和系统功能调用(21H)外，DOS 可调用中断包括 20H、25H、26H、27H 和 2FH 4 个中断处理程序。从功能而言，4 个中断分成 3 部分：

(1) 程序终止返回 DOS(20H 和 27H)；

(2) 磁盘扇区读写操作(25H 和 26H)；

(3) 假脱机打印文件(2FH)。

1.3.1 程序终止处理

DOS 提供的程序终止中断有两种不同的处理方式：

(1) 程序退出返回 DOS(INT 20H)；

(2) 程序常驻返回 DOS(INT 27H)；

这两个中断的区别在于：前者通知 DOS 程序已结束，将释放占用的内存缓冲供以后的程序加载；后者通知 DOS 要保留部分或全部内存缓冲(由 DX 指定)。但两者都要完成以下的动作：

(1) 从 PSP 的位移 0AH 处恢复中断 22H 向量；

(2) 从 PSP 的位移 0EH 处恢复中断 23H 向量；

(3) 从 PSP 的位移 12H 处恢复中断 24H 向量；

这样，两者都借助于程序结束处理程序返回到 COMMAND.COM 的常驻区，出现 DOS 提示符。

需要特别说明的是：调用 20H 或 27H 中断之前，要确保 CS 包含程序段前缀 PSP 的段地址，否则，会出现不可预测的结果。

在第三章已具体指出了这类中断的使用方法。.COM 文件或符合. COM 文件要求的.EXE 文件，可直接使用 INT 20H 或 INT 27H 。但在段需重定位的.EXE 文件中，应间接使用 INT 20H。

下面对两个中断调用各自需注意的问题分别说明如下：

1. 程序终止退出

若应用程序用传统的文件控制块写入的磁盘文件，则写入后应予关闭，否则，退出前，因该中断将文件缓冲全部清除，可能造成某些有用数据的丢失。

在 DOS 版本 2 或 3 中，推荐使用更先进的退出方法，即调用 INT 21H 的子功能 4CH。该子功能提供了出口码退出，无需对 CS 强加条件。

在 DOS 2.00 以上版本下使用该中断时，如果是在 EXEC 子功能(4BH)加载的一个子过程控制下退回的，那么不是返回到 COMMAND.COM，而是返回到它的父过程。

2. 程序常驻退出

该中断向量接受控制时，要求 DX 提供包括 PSP 在内的常驻程序字节长度加 1 的位移。显然，这至多只能保留 64K 字节的常驻空间。在 DOS 版本 2 或 3 中，使用系统功能调用的子功能 31H，可允许保留任意大的内存空间(DX 含有保留的节数)。

由于该中断不清除文件缓冲，故程序中凡打开的文件不会因程序退出而自动关闭。由于该中断执行时，将自动恢复三个系统缺省的中断向量，因此，不能用于企图永久驻留的用户编写的 Ctrl—C 或严重错误处理程序。

若用户.EXE 文件被加载到内存高端(以/H 参数连接)。那么不应使用该中断常驻。这是因为，被加载的文件可能会占据 DOS 的暂驻区，所以，当该中断退出时，如果 COMMAND.COM 暂驻部分不能被重新装入，则系统将无法运行。

当要求驻留的长度值为 0FFF1H～0FFFFH 时，该中断不能正常工作。此时，DOS 将舍去 DX 的最高位，结果至多驻留 32K 字节，少于程序的实际请求数。