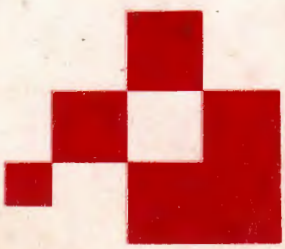
电脑步步高丛书·9·

朱 彤 编著

常用动态调试软件

中山大学出版社



形意三码汉字输入系统

总代理

* 汉字编码的最新大突破，软件专家贡献的编码杰作；
* 每个汉字最多只取三个码、智能处理重码的汉字系统；
* 字根都是《新华字典》中的偏旁部首，既传统又规范；
* 鸟牛马鱼龙虫，不相及但皆动物，一码蔽之，意也；
* 小口取 O，大口取 Q，一交点如 X，两交点如 H，形也；
* 以意立码，望形生码，形意三码是易学好用的汉字编码。

中山大学出版社电脑发展部

地址：广州市新港西路 135 号中山大学校内
邮编：510275　　联系人：张亚拉　李 文
电话：(020)4425565，4446300—7467，1995

ISBN 7-306-00897-8

责任编辑：张亚拉　封面设计：朱霭华　责任技编：黄少伟

9 787306 008978 >　ISBN7—306—00897—8/TP・32　定价：7.80元

电脑步步高丛书(9)

常用动态调试软件

朱彤 编著

中山大学出版社

·广州·

(粤)新登字 11 号

图书在版编目(CIP)数据

常用动态调试软件/朱 彤编著. —广州:中山大学出版社,1994

(电脑步步高丛书(9))
ISBN7—306—00897—8

Ⅰ 书名
Ⅱ 朱 彤
Ⅲ ①计算机 ②动态调试
Ⅳ TP3

责任编辑:张亚拉
责任技编:黄少伟　　封面设计:朱霭华

*

中山大学出版社出版发行
(广州市新港西路135号)
广东省新华书店经销
中山大学出版社印刷厂印刷

*

787×1092 毫米　16 开本　　8 印张　17 万字
1994 年 6 月第 1 版　　1994 年 6 月第 1 次印刷
印数:1—5000 册　　定价:7.80 元

《电脑步步高丛书》

序

电脑技术在现代社会中的重要性随着电脑应用的普及而变得越来越明显。技术的现代化，生产与管理的自动化，信息处理与交换的网络化，领导决策的科学化，教育与娱乐的影视化，乃至于物业与家庭理财方式和个人就业基础的变化等，都与电脑技术有着密切的联系。越来越多的人认识到学习电脑的重要性，领导人号召“电脑的普及要从娃娃抓起”。《电脑步步高丛书》就是为了给普及电脑开路。丛书从选题到选材都处处为广大电脑用户着想，试图给他们提供一套既简明扼要又系统实用的电脑技术资料。

电脑技术本身也在日新月异地发展着。硬件技术不断出现革命性的进步。多样性、多功能、多版本的系统软件与应用软件层出不穷，已有的系统和软件尚未摸透，新版的系统和软件又已流行开来，电脑技术的发展永无尽头。知识更新的速度是如此之快，以至于不奋起直追就跟不上潮流。时势造英雄，勇于开拓进取的人方有可能成为时代的弄潮儿。《电脑步步高丛书》就是这些弄潮儿发表自己的经验、体会和成果的天地 。

中山大学出版社计算机图书编辑室

1994 年 3 月

前　言

随着计算机的广泛应用和日益普及，它越来越深入到人们日常生活的各个领域。同时，与之相应的应用软件也以越来越快的速度开发出来和升级更新。由于软件产品是一种特殊的产品，生产难度大，成本高，又极易被复制，所以生产厂家出于商业上或技术上的考虑，往往会对其软件产品进行加密。另一方面，用户常常企图破译应用软件的密码，以便修改程序的某一部分来满足自己的实际需要，这就是所谓解密。

为了对软件进行解密或者调试一个程序，那就免不了要对其进行动态跟踪。最好的、功能最强的动态跟踪工具当然首推动态调试卡，但是，对于广大的微机使用者来说，这种专业性质的硬件设备过于昂贵，他们更感兴趣的是动态调试软件。

目前，有多种多样的动态调试软件。常见的有 DOS 系统的外部命令 DEBUG，Windows 系统的符号调试工具 SYMDEB，Microsoft 公司生产的 CODEVIEW 和全屏幕调试工具 FSD（Full Screen Debug），Borland 公司出品的 Turbo Debugger，以及 Nu－Mega 技术公司的 Soft－ICE 等。Soft－ICE 的动态调试功能特别强大，并随着 386 微机的普及而拥有越来越多的用户。

本书对上述动态调试软件都作了介绍，但重点介绍的是 Soft－ICE（2.5 版）。结合作者的使用经验，本书第 6 章介绍了一些常用的反动态跟踪技术，以便于读者进行动态调试时，能正确地判断和克服应用程序中设置的反跟踪障碍，使调试得以顺利进行下去。在本书附录中有重要端口和数据区的资料。在进行动态跟踪时，这些资料有时候是必不可少的。

本书的第五章由彭悦浩编著，特此表示感谢。

编　者

1994－5 于广州

目 录

1 概述

1.1 使用 Soft－ICE 的最大好处

当你调试或跟踪一个程序时，如果你使用的是 DOS 的 DEBUG 或 Turbo Debug 等其它一些动态调试工具，有时你会遇到这样一些情况：

最使你感兴趣的部分如果不在程序的开头，而是在中间，一大堆初始化代码之后，你不得不从开头进入，并艰难地越过这些对你来说毫无意义的代码以后，才能达到你感兴趣的部分；或者，你正在跟踪一个有加密措施的程序的执行过程时，在关键代码部分出现之前，该程序的自我保护指令关闭了显示屏，甚至屏蔽了键盘，使你试图分析其加密措施的希望成为泡影……。

在你束手无策或被一大堆只起除始化作用的非关键部分代码搞得心烦意乱时，你可能会希望：要是有一个功能强大的，并且能在任何时候随时进入的调试器，那么，一切将会变得顺手多了。

Soft－ICE 正是你所希望拥有的这种强大武器。

1.2 Soft－ICE 的描述

Soft－ICE 是一个优秀的动态调试工具，它提供硬件级的动态调试。具有强大的动态调试功能。

Soft－ICE 使用分页、I/O 特权级别、断点寄存器等 80386 保护方式的特性，在虚拟机上运行 DOS，以增加硬件级断点。

Soft－ICE 的这些特性，对于实时处理、硬件级中断、内存保护、中断服务等方面的程序的调试和跟踪显得特别方便。如果你还不熟悉 Soft－ICE，只使用过 CODEVIEW 或其它调试软件，由于 Soft－ICE 能与其它现有的调试程序一起工作，所以，你大可不必了解一个全新的动态调试程序就能得到一个取长补短的强有力的动态调试组合工具。

而且 Soft－ICE 具有良好的窗口界面，在线帮助等，使你能很方便的使用。Soft－ICE 还具有远程调试、双监视器等强大功能。

1.3 Soft－ICE 的特性

1）通过热键可随时弹出调式程序或返回应用程序。

2) 可在内存读写、口读写、内存指令执行、中断时方便地设置断点。
3) 在符号及源程序级上进行的动态调试。
4) 回溯过去的跟踪。
5) 与其它调试程序一起工作。
6) 支持 EMM4.0 协定。
7) 以 80386 保护方式工作。
8) 良好的窗口界面。
9) 充分利用内存资源。
10) 可调试设备驱动程序、中断服务程序、引导程序等 DEBUG 不便于调试的程序。
11) 灵活的热键及功能键设置功能。
12) 支持双监视器。

1.4 系统配置

Soft－ICE 只能工作在 386 或 386 以上的机型。又因为 Soft－ICE 不使用 DOS 的功能调用和 ROM BIOS 中断作为其屏幕显示及键盘输入，而是直接对硬件端口操作，所以，显示卡必须与 MDA、HERCULES、CGA、EGA、或 VGA 完全兼容。

1.5 磁盘文件

Soft－ICE 装在 5″ 或 3″ 磁盘上，包含以下文件：

S－ICE.EXE	Soft－ICE 主程序文件
S－ICE.DAT	包含初始化命令的文本文件
LDR.EXE	源程序和符号文件以及可执行文件的装入程序
MSYM.EXE	符号文件生成程序
UPTIME.EXE	设置时钟的程序
README.SI	包含有没出现在操作手册上的有关 Soft－ICE 最新信息的文本文件
SAMPLE.EXE	一个用作示范的短程序
SAMPLE.ASM	用作示范的汇编语言源程序
SAMPLE.SYM	示范程序的符号文件
IOSIM.ASM	一个受用户限制的断点的汇编语言源程序文件的例子
IOSIM.EXE	一个受用户限制的断点的例子

2 Soft－ICE 命令

2.1 安装

Soft－ICE 既可作为一个可安装的设备驱动程序在 CONFIG.SYS 中装入，也可在 DOS 提示符下直接安装。

2.1.1 在 DOS 提示符下安装

假如你所用的计算机上没有扩展内存，则只能这样安装。在 DOS 提示符下，象普通外部命令一样，敲入：

S－ICE

Soft－ICE 将装入内存高端，并对 DOS 为不可见，以后 DOS 在进行内存管理时将认为此处内存是不存在的，并且不能再分配给其它程序使用。因此，建议在其它内存驻留程序(TSR)或控制程序之前装入 Soft－ICE。

2.1.2 在 CONFIG.SYS 中作为设备驱动程序装入

如果你所使用的计算机上有扩展内存，那么最好把 S－ICE.EXE 作为设备驱动程序装入 CONFIG.SYS 中，这样才能充分发挥 Soft－ICE 的优越性能：

1) 与其它使用扩展内存的程序一起共享扩展内存，如 VDISK.SYS、SMARTDRV.SYS、PC－CACHE.EXE 等。

2) 使用 EMM4.0 的各种性能。

3) 在符号或源程序级上进行动态调试。

4) 回溯过去的跟踪。

5) 与其它调试程序共同工作。

Soft－ICE 将为主程序体以及相关保留部分定位一定的扩展内存，但 S－ICE.EXE 必须在 CONFIG.SYS 中的任何其它定位在扩展内存的设备驱动程序（如 VDISK.SYS)之前装入。建议在初次使用 Soft－ICE 时，把 S－ICE.EXE 作为第一个设备驱动程序写入 CONFIG.SYS 中：

DEVICE＝DRIVER:\PATH\S－ICE.EXE

而且为了更好的使用 Soft－ICE，应把 Soft－ICE 所在的路径写入 AUTOEXEC.BAT；另外，在符号或源程序级上进行调试时，为了搜索源文件路径，可设置环境变量 SRC：

SET SRC=PATH1;PATH2;……

其中 PATH1,PATH2,…… 为源文件所在目录,当要装入源文件时,将顺序搜索上述路径。

当然,也可在 DOS 提示符下直接运行 S－ICE.EXE。使用这种方法时,S－ICE.EXE 将自动安装在扩展内存顶端,几乎不占用常规内存。如果,扩展内存顶端已被其它程序占用,S－ICE.EXE 将覆盖它。所以只有能确定在此之前,没有任何其它程序装在扩展内存高端时,才能这样装入,且 Soft－ICE 的许多优良性能在此环境下才能充分发挥。

2.2 开始使用 Soft－ICE

本书使用说明:按 CTRL＋D 指先按下 CTRL 键不放,再按下 D 键;输入 WIN,指依次键入 W、I、N,然后按回车键,其余类似。所有介绍均以 Soft－ICE 2.5 版本为准。

2.2.1 激活

当安装好 Soft－ICE 后,可在任何时侯通过热键序列来随时激活 Soft－ICE。如果没有在 S－ICE.DAT 中作特殊的初始化处理,改变热键序列,那么,缺省的热键序列为 CTRL＋D。

按 CTRL＋D 激活 Soft－ICE,这时,你可以看到如下图所示的窗口。

如果路径设置不正确的话,搜索不到 S－ICE.DAT 初始化文件,窗口可能只有部分显现,若是小窗口,则可输入窗口命令 WIN,使其扩展为大窗口。

最上面一层为寄存器窗口,显示 AX、BX、CX、DX、SP、BP、SI、DI、DS、ES、SS、CS、IP 和标志 FLAGS 的值,标志置位以大写的高亮度字符表示。

第二层为数据窗口,左边为地址,中间是十六进制数据,右边是对应的 ASCII 码。

第三层为代码窗口,左边为地址或行号,中间是机器代码,右边是反汇编指令或源程序行。

第四层为命令窗口,输入各种控制命令。

底层是命令提示行,显示在线提示帮助。

```
AX=0100  BX=0003  CX=00F9  DX=007F  SP=0A8C  BP=0000  SI=010D  DI=01FB
DS=0123  ES=0123  SS=0123  CS=0123  IP=1111  o d I s z a p c
-------------------------------------------------------------------------
0000:0000 8A 10 1C 01 F4 06 70 00-16 00 77 0F B2 07 6A 02  ......p...w...j.
0000:0010 F4 06 70 00 B0 04 D8 17-43 EB 00 F0 EB EA 00 F0  ..p.....c.......
0000:0020 3C 00 77 0F B3 02 39 11-57 00 77 0F 6F 00 77 0F  <.w...9.W.w.o.w.
0000:0030 87 00 77 0F 9F 00 77 0F-B7 00 77 0F F4 06 70 00  ..w...w...w...p.
-------------------------------------------------------------------------
FDC8:A680 C3             RET
FDC8:A681 26C43E3400     LES     DI,ES:[0034]                ES:0034=0018
FDC8:A686 03FB           ADD     DI,BX
FDC8:A688 C3             RET
FDC8:A689 E8E0FF         CALL    A66C
FDC8:A68C 72FA           JB      A688
FDC8:A68E 26803DFF       CMP     BYTE PTR ES:[DI],FF
FDC8:A692 7504           JNZ     A698
-------------------------------------------------------------------------
:BC *
:EC
:PRN LPT2
:RS
:BPX
:     Enter A Command or ? For Help
```

2.2.2 退出

再按 CTRL＋D，或输入命令 X，退回到应用程序界面状态。

2.2.3 窗口操作

1) 改变大小

ALT＋↓　　使窗口变矮

ALT＋↑　　使窗口变高

窗口高度可在 8 行到 25 行之间调整。

改变窗口宽度用 WIN 命令，当 WIN 命令不带参数时，将使窗口宽度在全屏幕宽和 46 个字符宽之间反复切换。

2) 移动

当采用小窗口模式时，通过移动窗口，使你看到 Soft－ICE 窗口后面由应用程序所显示的屏幕信息，这对于调试需要观察屏幕输出的程序显得特别方便。

CTRL＋↓　　把整个窗口向下移动一行

CTRL＋↑	把整个窗口向上移动一行
CTRL＋←	把整个窗口向左移动一列
CTRL＋→	把整个窗口向右移动一列

2.2.4 行编辑操作

通过行编辑键，可以编辑以前的输入，而且每个键的意义都相当直观、方便，并有类似于 DOSKEY 的功能，可以回溯以前输入的命令。

↓	显示下一条命令
↑	显示上一条命令
←	向左移动光标
→	向右移动光标
PgUp	向上滚动一页
PgDn	向下滚动一页
SHIFT＋↓	向下滚动一行
SHIFT＋↑	向上滚动一行
INS	切换插入/改写方式
DEL	删除当前字符
HOME	光标移至行首
END	光标移至行尾
BACKSPACE	删除光标左边的字符，并退一格。
ESC	取消当前命令

注意，当光标在命令窗口时，↑、↓ 键起类似于 DOSKEY 的作用，当光标在代码窗口或数据窗口中时，与 SHIFT＋↑、SHIFT＋↓ 相同，使窗口屏幕上滚或下滚一行。

2.2.5 语法

在 Soft－ICE 中，命令的语法规则为：

命令　　参数

命令长度为一至六个字符，参数可有多个，以空格分开，参数必须是 ASCII 串或表达式。

1) ASCII 串

ASCII 串是以单引号或双引号括起来的 ASCII 字符串。

2) 表达式

表达式是由地址参数、数字、数字参数，以及寄存器组成，并且可由 ＋、－、*、/ 四种运算符连接。

(1) 数字一般是十六进制数的形式。

(2) 数字参数可分为字节参数、字参数、双字参数三种：

0F	字节参数
23CE	字参数
B000:8000	双字参数

双字参数是由用分号隔开的两个字参数组成。

(3) 寄存器

可用于表达式的寄存器有:AL、AH、AX、BL、BH、BX、CL、CH、CX、DL、DH、DX、SI、DI、BP、SP、IP、CS、DS、ES、SS、FLAGS。

(4) 地址参数

地址参数可以是双字参数,也可以是段址:偏移量的形式(如 DS:SI)。

另外,还可以用一些特殊符号来表示相应的地址:

$	当前的 CS:IP
@地址	间接地址
.NUMBER	源程序行号
.	当前指令

注意:@ 可多级嵌套。举几个例子如下:

U CS:IP－0A

反汇编从当前指令以前十字节处开始的指令。

U $－0A

同上。

U.10 (注:此处为十进制)

从源程序的第十行开始反汇编。

G @SS:SP

如果当前 IP 在中断服务或子程序的第一条指令处,这条指令将在调用处的下一条指令处设置断点,并连续执行完中断服务或子程序,返回调用处。

2.2.6 帮助信息

如果你对 Soft－ICE 的某条命令的使用方法不是很熟悉,那么,你可以在 Soft－ ICE 窗口下用 ? 或 H 命令来获得及时的帮助。

?	显示所有命令的简短描述。
? 命令	显示某一特定命令的较详细帮助及其使用举例
? 表达式	求表达式的值,并依次以十六进制,十进制和 ASCII 码三种方式显示结果。

用 H 代替上述的 ？，结果完全一样。

2.2.7 一个简单例子

以下以 Soft－ICE 自带的一个简单汇编程序 SAMPLE 为例，介绍如何用 Soft－ICE 来调试程序。

在 CONFIG.SYS 的第一行安装 Soft－ICE：

```
DEVICE=DRIVE:\PATH\S-ICE.EXE /SYM 50
```

参数 /SYM 50 表示为符号及源程序保留 50K 的扩展内存。若没有扩展内存，可在 DOS 提示符下直接运行 S－ICE. EXE，但不能在符号及源程序级进行调试。另外，在 AUTOEXEC.BAT 中，写明 Soft－ICE 的搜索路径。然后重新启动系统，初始化屏幕如下：

```
                Soft-ICE
Kelvin Ishigo
Elite High Technology Inc.
Registration #SI011753

Copr. (c) Nu-Mega Technologies 1987, 1990
All Rights Reserved
Soft-ICE Version 2.50
Soft-ICE is loaded from 00227000H up to 00260000H
50K of symbol space reserved
10K of back trace space reserved
1180K of extended memory avaiable
```

第一行至第八行是版权及版本信息，第九行告诉你 Soft－ICE 及其相关部分所占用内存的确定区域，第十行显示有多少符号和源程序所用的符号空间被保留，第十一行显示有多少内存为回溯跟踪缓冲区所保留(默认为 10K)，第十二行显示还有多少扩展内存剩下。

首先用 CTRL＋D 激活 Soft－ICE。

接着，用 WIN 命令，在小窗口模式和全屏幕方式之间切换，若是小窗口，用 CTRL＋↑、CTRL＋↓、CTRL＋←、CTRL＋→，可使窗口分别向上、下、左、右移动；用 ALT＋↑、ALT＋↓，可使窗口变高、变矮；用 SHIFT＋↑、SHIFT＋↓，可使当前窗口信息上滚、下滚一行；而 PGUP、PGDN，可使当前窗口信息向上翻页或向下翻页。

WR 命令可使寄存器窗口打开或关闭。

MAP 命令显示系统内存的映象，当前指令指针(CS:IP)所在区域是用高亮度表示的。

VER 命令显示版本号。

RS 命令将暂时显示应用程序屏幕，待按任一键后，返回 Soft－ICE 窗口，这对于监视应用程序的输出十分有用。

CLS 命令可清除命令窗口。

X 命令退出 Soft－ICE 窗口。

现在，我们已经知道了几个最常用的 Soft－ICE 控制命令，接下来我们将会试着调试

Soft－ICE 自带的一个叫 SAMPLE 的程序。该程序是一个叫名 Jed 的程序员写的一个简单的汇编语言程序，其作用是判断一个敲键输入是否是空格键，并显示相应信息。主程序是由 Jed 自己写的，子程序由 Jed 的朋友 Jake 完成。程序清单如下：

```
        Page  55,80
        Title Sample program for Soft－ICE tutorial

CODE    Segment Public      'Code'
CODE    Ends

DATA    Segment Public      'Data'

public pad, char, answer, space_msg, no_space_msg

pad             db 12H dup(0)
char            db 0
answer                  db 0
space_msg       db 'The Character is a SPACE', 0DH, 0AH, '$'
no_space_msg            db 'The Character is NOT a SPACE', 0DH, 0AH, '$'
DATA    Ends

STACK Segment Stack         'Stack'

        Dw      128 Dup (?)         ;Program stack

STACK Ends

CODE    Segment Public      'Code'

        Assume          CS:CODE, DS:DATA, ES:Nothing, SS:STACK

public start, main_loop, no_space, get_key, is_space?, not_space, hang_example
start:

; Set up segments

        mov     ax, DATA            ;ax = DATA segment
        mov     es, ax                  ;es = DATA segment
        mov     ds, ax                  ;ds = DATA segment

; Main Program Loop

main_loop:

        call    get_key             ;call Jake's get key routine
        call    is_space?           ;call Jake's is_space routine
        cmp     answer, 0            ;was it a space?
        je      no_space             ;no — go print not a space message

; Its a space, display the space message
```

```
        mov    ah, 9                ;ah = DOS display string function
        mov    dx, offset space_msg     ;dx -> space message
        int    21H                  ;call DOS
        jmp    main_loop            ;jump back to the main loop

; Its not a space, display the no space message

no_space:
        mov    ah, 9                ;ah = DOS display string function
        mov    dx, offset no_space_msg ;dx -> not a space message
        int    21H                  ;call DOS
        jmp    main_loop            ;jump back to the main loop

; Get Key Routine

get_key     proc
        mov    ah, 8                ;ah = DOS get key function
        int    21H                  ;call DOS
        mov    char, al             ;store byte in variable char
        ret                         ;return from this routine
get_key     endp

; Check to see if the character is a space

is_space?   proc
        cmp    char, 20H            ;is the character a space?
        jne    not_space            ;no - go do funny business
        mov    answer, 1            ;yes - return 1 in variable answer
        ret                         ;return from this routine
not_space:
assume cs:data                          ;required to produce bizzar bug
        mov    cs:answer, 0             ;return 0 in variable answer
assume cs:code
        ret                         ;return from this routine
is_space?   endp

; The following routine is an infinite loop with interrupts disabled

hang_example:
        cli                         ;disable interrupts
        jmp    $                      ;infinite loop

        mov    ax, 4c00h            ;ax = DOS exit command
        int    21h                  ;call MSDOS

CODE   Ends
        End    start
```

运行该程序后，发现无论敲入什么键，均显示 The character is a SPACE，显然程序是有问题的，需要通过调试和跟踪来发现所存在的问题并加以修改。

以下，我们用两种不同的方法来进行这一工作：

1) 在 DEBUG 环境下使用 Soft－ICE

首先我们用 Soft－ICE 作为 DEBUG 的辅助工具来调试 SAMPLE 程序。

在确认装入 Soft－ICE 后，我们就可以开始调试工作了：

(1) 联结 Soft－ICE 和 DEBUG

用 CTRL＋D 激活 Soft－ICE，输入命令：

```
ACTION INT3
```

该命令把 Soft－ICE 与 DEBUG 联系起来。它告诉系统，当断点发生时，将控制权交给 DEBUG。ACTION 的最初设置是 HERE 状态，ACTION HERE 将把控制权直接返回给 Soft－ICE，当独立使用 Soft－ICE 时，应用 ACTION HERE 状态。

(2) 用 X 命令退出 Soft－ICE。

(3) 在 DOS 下，用 DEBUG 调试 SAMPLE.EXE，键入以下命令：

```
DEBUG DRIVE:\PATH\SAMPLE.EXE
－U
－R
```

(4) 在 Soft－ICE 窗口下设置断点

我们怀疑，可能因某种原因，使程序的代码段被覆盖了，所以可在代码段设置断点。用 CTRL＋D 激活 Soft－ICE，切换成小窗口模式，以便于更好地和 DEBUG 一起工作。根据 DEBUG 所显示的寄存器信息，输入命令：

```
BPR Code－Seg:0 Code－Seg:25 W
```

Code－Seg 是由 DEBUG 的 R 命令所显示出来的 CS 寄存器值。BPR 命令设置了一个内存范围类型的断点，Jed 的代码段长度是 25H 个字节，所以结束处偏移为 25H。该命令告诉 Soft－ICE，如果有指令在 Jed 的代码段进行写(W)操作，那么断点将发生，并将控制权交给 DEBUG (因前面已使用 ACTION INT3 命令)。

用显示所有断点的命令 BL，将显示：

```
0) BPR Code－Seg:0000 Code－Seg:0025 W C＝01
```

符号 0) 标识这个断点，是断点号；总数 C 没有描述，默认为 1。

(5) 在 DEBUG 状态下调试

按 CTRL＋D 或用命令 X 退出 Soft－ICE。在 DEBUG 下运行 SAMPLE。输入运行命令：

```
－G
```

此时，程序将在 DEBUG 控制下连续运行。按空格键，至今还正常。然后敲入一个非空格键，这时，DEBUG 弹出，用 U 命令可见在在当前 CS:IP 处的指令为 RET，而引起中断的指令是上一条指令，即在偏移 3BH 处. 从该地址开始的指令为：

```
CS:
MOV BYTE PTR[13],0
```

可见 Jake 的子程序覆盖了 Jed 的主程序，程序的错误也就出在这里。

2）单独使用 Soft－ICE

接下来，我们单独使用 Soft－ICE 来动态地调试程序。仍以 SAMPLE 为例，但这次我们将在源程序级对 SAMPLE 进行调试。

(1) 准备工作

a. 配置 CONFIG.SYS

在系统配置文件 CONFIG.SYS 中加入 Soft－ICE 的安装行：

```
DEVICE＝DRIVER:\PATH\S－ICE.EXE /SYM 50
```

该命令将 Soft－ICE 作为设备驱动程序装入，并为符号文件和源文件保留 50K 的内存空间。

修改完 CONFIG.SYS 后重新启动系统。

b. 装入应用程序

用程序装入工具 LDR.EXE 来装入程序，或符号文件和源文件。输入：

```
LDR SAMPLE
```

这时，Soft－ICE 转换成宽方式，在代码窗口，既有源程序行，又有反汇编代码。

c. 断点的设置和清除

按 F6 键(或 EC 命令)把光标直至切换到代码窗口。用↑和↓键或 PgDn 和 PgUp 键移动光标并滚动源程序行，使光标停在第 42 行上，按 F9 设置一个断点(或者直接在命令窗口用 BPX . 42 命令设置断点)。这时第 42 行应该是高亮度的，表明已在这行设置了断点。用 BL 命令可以察看我们所设置的断点。

现在用 CTRL＋D 命令跳出 Soft－ICE，就引起 SAMPLE 程序开始执行，并一直执行到第 42 行为止，Soft－ICE 会再次弹出，并停在 42 行上。可见断点确实工作了。

清除断点用 BC 命令。首先清除 42 行上的那个试验断点，输入命令：

```
BC *
```

它将清除包括第 42 行上的那个试验断点在内的所有断点。

(2) 在符号和源程序级上调试

首先我们用命令

```
SYM
```

察看 SAMPLE 的公共符号。然后用符号来设置断点：

```
BPR start . 82 W
```

这条命令从符号 start 处到源程序的第 82 行设置了一个范围断点。输入：

```
BL
```

察看断点是否被正确设置。若已设置正确，就可以按 CTRL＋D 退出 Soft－ICE 窗口，并引起应用程序的执行。

按空格键，一切正常。再按非空格键，Soft－ICE 弹出。这时，当前指令的前一条指令就是

引起断点发生的指令，即符号 not_space 处的指令。问题就出现在这儿！

终止 SAMPLE 的运行，回到 DOS：

```
EXIT RD
```

(3) 修改代码

重新装入 SAMPLE。由于符号文件和源文件已经装入了，所以只需再装入可执行文件 SAMPLE.EXE 即可：

```
LDR SAMPLE.EXE
```

修改错误代码：

```
A not_space
NOP
```

这时源程序和代码段都被相应改正了。按 CTRL＋D 继续运行已修改后的程序。再输入一些空格和非空格键，你会看到程序能够正常工作了！

(4) BREAK 状态的试验

用 CTRL＋C 终止 SAMPLE 程序的运行。

现在我们再来试一试 Soft－ICE 的 BREAK 特性：

再次装入 SAMPLE.EXE：

```
LDR SAMPLE.EXE
```

使程序进入一个死循环：

```
RIP hang_example
```

此处的指令是：

```
CLI
JMP $
```

它首先屏蔽中断，然后执行一条跳回自己处的转跳指令，所以，这是一个使系统挂起的无限死循环。

输入：

```
BREAK ON
```

将 BREAK 状态置为 ON，再按 CTRL＋D 退出 Soft－ICE。这时由于整个系统是挂起的，甚至按热启动 CTRL＋ALT＋DEL 都毫无反应。但是，Soft－ICE 却可以被激活。请按热键 CTRL＋D，Soft－ICE 窗口又弹出了。

这说明只要 BREAK 置于 ON 状态，Soft－ICE 就可在任何时候被激活，即使系统挂起了，还是能够返回 Soft－ICE。

2.3 命令

这一节介绍 Soft－ICE 的所有命令集，对每一条命令及其所带参数做了详尽的解释，并举例说明，同时也介绍了一些使用技巧。

用 [] (中括号)括起的内容是可选项；用 | 号隔开的内容表示任选其一；命令中的大小写字符是没有分别的；数字一般指十六进制数。

2.3.1 断点设置命令(SETTING BREAK POINTS)

BPM,BPMB,BPMW,BPMD ————在内存存取或执行时设置断点
BPR ————在内存某一范围内设置断点
BPIO ————在 I/O 口存取时设置断点
BPINT ————在某一中断处设置断点
BPX ————设置或清除执行断点
CSIP ————设置 CS:IP 范围校验
BPAND ————当多个断点同时有效时才发生中断

命令:BPM,BPMB,BPMW,BPMD
功能:在内存存取或执行时设置断点
语法:

BPM[size] address [R|W|RW|X] [qualifier value] [c=count]

size————B,W,D
B——Byte
W——Word
D——Double Word (双字)

R|W|RW|X————描述断点将要申请的存取类型值
R ——表示读
W ——表示写
RW——表示读写
X ——表示执行

qualifier————EQ,NE,GT,LT,M
EQ——等于
NE——不等于
GT——大于
LT——小于
M ——屏蔽

size 指定了断点所覆盖的有效范围。例如，某一断点被定义为一个字(word)，如果该字的

第二字节被程序的某一条指令所修改，于是断点就会有效，产生中断。size 在选择校验被描述时也是十分重要的。

注：

BPM 命令允许你在内存读写或执行时设置断点。

如断点申请的存取类型值没有被描述，缺省值为 RW。

如果 size 没有被描述，缺省值为 Byte。

断点申请的存取类型值，除 X 外，引起程序去执行引起断点的指令，当前 CS:IP 将指向断点之后的指令。如果类型值是 X，当前 CS:IP 指向断点处的指令。

如果 R 被描述，断点将出现在读操作及写操作上，而不会改变内存位置的值。

如断点申请的存取类型值是 R、W 或 RW 的话，执行一条在描述地址处的指令将不会引起断点的发生。

如果用 BPMW，描述地址必须从字边界开始；如果用 BPMD，描述地址必须从双字边界开始。

例：

BPM 3000:DI W EQ 11 C=2

在内存字节存取处定义了一个断点，当十六进制数 11H 第二次写在 3000:DI 的位置时，断点将会发生。

BPM CS:1000 X

该指令在执行时定义了一个断点，当指令地址第一次到达 CS:1000H 时，断点将会发生。当前的 CS:IP 应该是断点处的指令。

BPMW DS:A000 W EQ 0XXXXXXXXXXXXXX1

该命令在内存写时定义了一个字断点，断点作用发生在第一次在 DS:A000 位置处有写有高位为 0 低位为 1 的值时。其中标有 X 的位可以是任意值。

BPM ES:0000 W LT 5

这条命令在内存写时定义了一个字节断点，断点作用发生在第一次在 ES:0000 处写入小于 5 的值的时候。

命令：BPR

功能：在内存的某一范围内设置断点

语法：

BPR start－address end－address [R|W|RW|T|TW] [c=count]

start－address，end－address－－－－起始和结束地址

R|W|RW|T|TW－－－－描述断点将要申请的存取类型值

R －－表示读

W －－表示写

RW－－表示读写

T －－用于小范围回溯跟踪

TW－－用于 COARSE 大范围回溯跟踪

注：

所有描述断点将要申请的存取类型值除 T 或 TW 外，均使程序去执行引起断点发生的指令。当前 CS：IP 应指向断点之后的指令。若是没有被描述，W 为缺省值。任何包括断点范围在 4K 内的读写页将被 Soft－ICE 所分析，但这将可能降低系统的性能。

T 和 TW 记录在描述范围内的执行步骤。它们不引起断点发生，但可以为 SHOW 以及 TRACE 命令记录回溯跟踪的指令信息。

例：

BPR B000:0 B000:1000 W

如果有任何在单色显示器内存范围内进行写操作时，断点发生。

BPR 3000:0100 3000:0200 T

记录在 3000:0100H 到 3000:0200H 范围内所执行过的所有指令。

命令：BPIO

功能：**在 I/O 端口存取时设置断点。**

语法：

BPIO port－number [R|W|RW|X] [qualifier value] [c=count]

port－number－－端口号，一个字或字节

R|W|RW －－－－描述断点将要申请的存取类型值

R －－表示读(IN)

W －－表示写(OUT)

RW－－表示读写(IN/OUT)

value －－－－－一个字或字节

注：

BPIO 命令允许你在 I/O 端口读写时设置断点。

如果描述了 value 值，它将与指定端口的 IN 或 OUT 指令所读写的实际值作比较，二者满足关系式 qualifier value 时断点发生。如果 I/O 口为一个字节口，那麼 value 的低八位用于比较。

如果没有描述断点将申请的存取类型值，缺省为 RW。

例：

BPIO 21 W NE 4E

当 21H 端口(中断控制屏蔽寄存器)被一个不等于 4EH 的值写时，断点发生。

BPIO 3F8 R EQ M 00XXXXXX

这条命令使断点作用发生在串行口 3F8H 读入一个高两位为 1 的值时(其余各位可以为任意值)。

命令：BPINT

功能：**在中断处设置断点。**

语法：

BPINT int－number [<AL|AH|AX>=value] [c=count]

int－number－－中断号(00H－FFH)

value　　－－－个字或字节

注：

BPINT 命令允许在执行一个硬件或软件中断时设置断点。当中断产生时 value 与描述寄存器(AH,AL,AX)作比较,若二者值相等,则断点发生。当断点发生时，如果是硬件中断,指令指针(CS:IP)将指向中断服务程序的第一条指令;如果是软件中断,指令指针(CS:IP)将指向引起中断的 INT 指令。INT? 命令可以用来看中断发生时在哪里执行。

例：

BPINT 21 AH=3D

这条命令在中断 21H 处定义一个断点,当调用 DOS 服务功能 3DH 时断点发生。

命令:BPX

功能：　　**设置或清除执行断点。**

语法：

BPX [address] [c=count]

注：

当光标出现在代码窗口时,在光标当前所在位置设置一个执行断点，如果光标当前所在位置已经设置了一个执行断点,就取消该断点;当光标不在代码窗口时，地址必须被描述,若仅描述一个偏移量,那么当前 CS 寄存器作为段址。

例：

BPX .10

在源程序的第十行处设置一执行断点。

命令:CSIP

功能:**设置** CS:IP **范围校验。**

语法：

CSIP {OFF|[NOT] start－address end－address}

OFF－－关闭 CS:IP 校验

NOT－－断点仅在 CS:IP 指针指向描述范围外时发生

注：

当断点被有条件地满足时,CS:IP 指针与一个特定范围作比较，如果它们在范围内,断点将是活跃的。为激活在 CS:IP 范围外的断点,请用 NOT 参数。

当 CS:IP 范围被描述,它用于所有当前活跃的断点;如果没有参数描述,当前的 CS:IP 范围是显示的。

例：

CSIP NOT F000:0000 FFFF:FFFFH

仅当断点条件被满足,且 CS:IP 不在 ROM BIOS 中(F000:0000H 到 F000:FFFFH)时,

这条命令才引起断点发生。

命令:BPAND

功能:当多个断点同时有效时才发生中断

语法:

BPAND int－list | * |OFF

int－list——断点号列表(多个由逗号或空格分开的断点号)

*　　——所有断点

OFF　　——关掉相与类型的断点

注:

BPAND 命令在两个或多个断点间作逻辑 AND,仅当所有断点都被满足时才发生。每次使用 BPAND 命令时,描述的断点号加到 int－list 上,直到使用 BPAND OFF 为止。

用 BL 命令显示断点时,相与的断点后面会有一个 & 号。

例:

```
BPAND 0,3,5
```

仅当所有三个断点的条件都满足时才发生。如果断点 3 和 5 至少满足条件一次,但断点 0 仍然没有被满足,那么直到断点 0 的条件满足为止,断点才会发生。

2.3.2　断点处理(MAINPULATING BREAK POINTS)

BPE－－－－编辑断点

BPT－－－－用断点作模式

BL －－－－断点列表

BC －－－－清除断点

BD －－－－禁止断点

BE －－－－允许断点

命令:BPE

功能:编辑断点

语法:

BPE break－number

注:

BPE 命令用于对已经存在的断点的描述参数进行修改和编辑。它首先把所指定的断点装入编辑行,然后你可以用各种编辑键进行编辑,完成后敲 ENTER 键确认并退出。

例:

```
BPE 3
```

将断点 3 及其描述装入编辑行,当修改完成后按 ENTER 键。

命令:BPT

功能:用断点作模式

语法：

BPT break－numbe

注：

BPT 命令装入一个存在的断点的描述至编辑行作为新断点的模式。这条命令提供了一个生成与现有断点的描述相同的新断点的快速方法。

例：

```
BPT 0
```

将断点 0 作为模式装入编辑行，经编辑修改后按 ENTER 就加入了一个新断点。

命令：BL

功能：断点列表

语法：

BL

注：

BL 命令显示当前已设置的所有断点。对每个断点，列出断点号、断点条件、断点状态及次数 count。如果断点状态是禁止的，星号（*）将显示在断点号后面；如果一个有效断点是由 BPAND 命令指定的与断点，星号（*）将显示在断点描述后面；引起断点发生的最近的断点是高亮的。

例：

```
BL
```

命令执行后，显示如下：

```
0)*   BPIO  037F W EQ 0FF C=02
1)    BPR   B800:0000 C000:0000 W C=01 *
2)    BPINT 21 AX=2500 C=01 *
```

断点 0 目前是禁止的；断点 1 和断点 2 是与断点，只有二者条件同时被满足时断点才发生。

命令：BC

功能：清除断点

语法：

```
BC  int－list | *
    int－list－－－－－－系列被逗号或空格隔开的断点号
      *          －－－－－表示所有断点
```

注：

BC 命令可以删除一个或多个断点。

例：

```
BC  0,1
```

删除断点 0 和断点 1。

命令:BD

功能:**禁止断点**

语法:

BD int－list | *

注:

暂时使一个或多个断点失效。BD 命令可用于暂时减少断点数。用 BL 命令显示时,被禁止的断点号后面有一个星号(*)。

例:

```
BD   0 2
```

暂时禁止断点 0 和断点 1。

命令:BE

功能:**允许断点**

语法:

BE int－list | *

注:

重新使被 BD 命令禁止的断点有效。

例:

```
BE   0
```

再次使断点 0 恢复有效。

2.3.3 显示/修改内存(DISPLAY/CHANGE MEMORY)

MAP	————显示系统内存映象
R	————显示或改变寄存器值
U	————反汇编指令或显示源程序行
D	————以上次的描述格式显示内存
DB	————以字节方式显示内存
DW	————以字方式显示内存
DD	————以双字方式显示内存
E	————以上次的描述格式编辑内存
EB	————编辑内存字节
EW	————编辑内存字
ED	————编辑内存双字
INT?	————显示上次中断号
H	————显示帮助信息
?	————同 ?
VER	————显示 Soft－ICE 的版本号

命令:MAP

功能:显示系统内存映象

语法:

MAP

注:

MAP 命令显示系统内存各组成部分的名称、位置、长度。当前 CS:IP 所在区域是高亮度的。

例:

MAP

显示如下:

```
Start        Length
0000:0000    0040 Interrupt Vector Table
0040:0000    0030 ROM BIOS Variables
0070:0000    00A6 I/O System
011E:0000    013D DOS
0225:0000    0000 EMMxxxx0
02D4:0000    0049 XMSxxxx0
031D:0000    0121 DOS File TAble & Buffers
043F:0000    0004 Owner Is DOS
0444:0000    00A5 Owner Is Command Shell
04EA:0000    0004 Owner Is Free Block
04FE:0000    0010 Owner Is Command Shell
0500:0000    0007 Owner Is SAMPLE.EXE
0508:0000    9AF3 Owner Is SAMPLE.EXE
A000:0000    5000 System BUS
F000:0000    1000 ROM BIOS
```

其中 0508:0000H 处是高亮度的,表示当前 CS:IP 所在区域。

命令:R

功能:显示或改变寄存器值

语法:

R [register－name[=]value]

register－name ———寄存器名(AX AH AL BX BH BL CX CH CL DX DH DL DI SI BP SP IP CS DS ES SS FL)

value ———如果寄存器名是除 FL 以外的任意一个寄存器,value 为十六进制数或表达式;如果寄存器名是 FL,value 为一个或多个标志符号,每个均可通过加号或减号来进行选择。

O（溢出标志）　D（方向标志）　I（中断标志）
S（符号标志）　Z（零标志）　A（辅助进位标志）
P（奇偶标志）　C（进位标志）

注：

R 命令可显示或改变寄存器的值。

如果不指定参数，那么显示当前 CS:IP 处所有寄存器的值及各标志位状态。

如果仅指定寄存器名而没有给出值 value，Soft－ICE 将显示该寄存器的当前值并提醒你输入新值。若寄存器名是 FL，设置的标志位以高亮大写字符显示；清除的标志位以普通小写字符显示。按 ENTER 保持当前寄存器的值不变。

如果既指定了寄存器名又给出值 value，那么将更新相应寄存器的值。

为了改变标志值，用 FL 作寄存器名，一个加号在标志符号之前将使其置位，一个减号在标志符号之前将使其清除，既无加号又无减号时将使标志位翻转。

例：

R DI 0

使方向标志 D 和中断标志 I 清零。

R FL

显示当前各标志位，并允许改变它们。

R FL O－D＋S

翻转溢出标志，清除方向标志，设置符号标志。

R FL＝ZC

翻转零标志和进位标志。

命令：U

功能：反汇编指令或显示源程序行

语法：

U[address] [L[＝]length]

length－－－－－反汇编的指令条数

注：

如果没有指定 length，默认为 8 行或比窗口高度少一行。若没有描述地址，从以前反汇编出的最后一条指令处开始反汇编，如果没有以前的反汇编，则从当前 CS:IP 处开始反汇编。若用源程序或符号方式表示地址，则按当前程序方式显示。

例：

U .100

从源程序的第一百行开始显示。

U $＋2

从当前地址后 2 字节处开始反汇编。

命令:D,DB,DW,DD

功能:以最近的描述格式或指定的格式显示内存

语法:

D[size] [address] [L[=]length]

size———B(字节),W(字),D(双字)

length——将要显示的字节长度

注:

若无地址描述,从上次显示出的最后一字节后开始显示,如果没有 size 描述,将按上次的 size 标准显示,若 length 没有描述,默认为 8 行或比窗口高度少一行。例:

DD ES:0 L=8

以双字格式和 ASCII 码格式显示从 ES:0 开始的 8 个字节。

命令:E,EB,EW,ED

功能:以上次的描述格式或指定的格式编辑内存

语法:

E[size] address [data-list]

size ——B(字节),W(字),D(双字)

data-list——按 size 格式列出的数据或者由逗号或空格分开的引用字符串表(用双引号或单引号均可)

注:

如果没有 size 描述,将按上次的 size 标准进行。按 TAB 键在 ASCII 码和十六进制数方式间进行转换。

例:

EB DS:0 'FILES',0

以 ASCIIZ 串替换 DS:0 处开始的 6 个字节。

命令:INT?

功能:显示上次中断号

语法:

INT?

注:

INT? 命令显示上次中断发生处的地址及中断号。

例:

INT?

显示如下:

Last Interrupt: 16

At 0070:0255

命令:H

功能：显示帮助信息

语法：

H [command | expression]

注：

如果没有参数，将显示所有命令的简单描述，按 ESC 键退出帮助信息，其它任意键继续显示。

如果后跟一个命令，将显示该命令的更详尽的信息，包括命令语法及例子。

如果后跟一个表达式，将分别以十六进制、十进制、ASCII 码三种形式显示表达式的值。

例：

H FKEY

该命令显示有关 FKEY 命令的信息，包括命令语法及例子。

H 8+13*2

执行后得：

0038 0056 "8"

命令：?

功能：显示帮助信息（同 H 命令）

语法：

? [command | expression]

注：

该命令和 H 命令等效。

例：

?

命令：VER

功能：显示 Soft－ICE 的版本号

语法：

VER

注：

VER 命令显示 Soft－ICE 的版本号及 Nu－Mega 公司的技术版权信息。

例：

VER

2.3.4 I/O 口命令（I/O PORT COMMANDS）

I －－－－从 I/O 字节口输入

IB－－－－从 I/O 字节口输入

IW－－－－从 I/O 字口输入

O ————向 I/O 字节口输出
OB————向 I/O 字节口输出
OW————向 I/O 字口输出

命令:I,IB,IW
功能:**从 I/O 口输入**
语法:

I[size] port
size——B(字节),W(字)
port——端口号(一个字节或字)

注:
该命令从一个字口或一个字节口读入并显示这个值。如果没有指定 size,则默认为字节。
例:

I 3F7

从串行口 3F7H 读入一个字节。

命令:O,OB,OW
功能:**向 I/O 口输出**
语法:

O[size] port
size——B(字节),W(字)
port——端口号(一个字节或字)

注:
该命令用来向一个字口或一个字节口写一个值。如果没有指定 size,则默认为字节。
例:

O 21 FF

这条命令把中断屏蔽寄存器的每一位都置 1,所以将屏蔽掉所有的硬件中断。

2.3.5 转换控制命令(FLOW CONTROL COMMANDS)

X ————退出
G ————执行到某一地址
T ————单步跟踪
P ————执行一步
HERE ————转向当前的光标行
GENINT————强迫产生一个中断
EXIT ————强迫从当前正被跟踪的程序中退出
BOOT ————引导系统但保留 Soft-ICE
HBOOT ————重新引导系统(相当于 RESET)

命令:X

功能:**退出**

语法:

X

注:

X 命令退出 Soft－ICE 窗口,并把控制权交给被 Soft－ICE 中断的程序。如果设置了断点,那么断点将是活跃的,一旦条件满足,断点将被激活,Soft－ICE 窗口会弹出,并获得控制权。

例:

X

命令:G

功能:**执行到某一地址**

语法:

G [=start－address] [break－address]

注:

G 命令类似于 DOS 的 DEBUG 的 G 命令。如果没有给出起始地址,则从当前 CS:IP 处开始执行,如果指定了起始地址则从起始地址开始执行,直到遇到断点地址为止。断点地址必须是一条指令代码的第一个字节。执行时 Soft－ICE 窗口消失,界面由应用程序控制,一直执行到断点地址处,Soft－ICE 窗口将再次弹出。

例:

G=CS:100 CS:2000

从当前代码段 100H 偏移处开始,一直执行到偏移 2000H 处终止。

命令:T

功能:**单步跟踪**

语法:

T [=start－address] [count]

注:

T 命令类似于 DOS 的 DEBUG 的 T 命令。如果没有描述起始地址,将从当前 CS:IP 处开始,若给出了起始地址则从所给的地址开始,若没有带 count 参数则单步跟踪一条指令,若带有 count 参数则连续单步跟踪多条指令(其间可用 ESC 键终止)。

在源程序方式下,T 命令将单步执行到下一条源程序语句。若是一条子程序或函数调用语句,T 命令将跟进子程序或函数调用中。

例:

T=100 7

从当前代码段 100H 处开始单步执行 7 条指令。

命令:P

功能：执行一步

语法：

P

注：

P 命令类似于 DOS 的 DEBUG 的 P 命令。如遇到中断调用，子程序调用，循环指令，重复串操作指令，P 命令将不跟入，而是一步执行完。

例：

P

执行一步程序。

命令：HERE

功能：转向当前的光标行

语法：

HERE

注：

HERE 命令从当前 CS：IP 处开始，执行到代码窗口中当前光标行所在位置。若光标不在代码窗口则 HERE 命令无效。

例：

HERE

在当前光标行所在位置设置一个断点，然后从 Soft－ICE 窗口中退出并从当前指令指针 CS：IP 处开始执行，一直执行到当前光标处。

命令：GENINT

功能：强迫产生一个中断

语法：

GENINT INT1|INT3|NMI|interrupt－number

interrupt－number——中断号(从 00H 到 FFH)

注：

GENINT 命令可以强制产生一个中断。这条命令主要用于测试各种中断服务程序或者当 Soft－ICE 和其他动态调试程序一起工作时将控制权交给另一个调试程序。

例：

GENINT INT3

如果 Soft－ICE 正和 DEBUG 一起工作，那么，这条指令将把控制权交给 DEBUG。

命令：EXIT

功能：强迫从当前正被跟踪的程序中退出

语法：

EXIT [R][D]

R——重新装入中断向量表

D——删除所有的断点

注：

EXIT 命令强制执行一个 DOS 退出功能(INT21H 的 4CH 子功能)。R 参数可恢复中断向量表;D 参数删除所有已设置的断点。

当 DOS 功能无法调用时该命令无效。

例：

```
EXIT RD
```

终止当前正被调试的应用程序,并恢复中断向量表,删除所有断点。

命令:BOOT

功能:**引导系统但保留** Soft－ICE

语法：

```
BOOT
```

注：

BOOT 命令调用一个软中断 INT19H 重新启动系统,但 Soft－ICE 保持常驻。

例：

```
BOOT
```

重新引导系统但保留 Soft－ICE。

命令:HBOOT

功能:**重新引导系统**(**相当于** RESET)

语法：

```
HBOOT
```

注：

HBOOT 命令重新引导系统并且不保留 Soft－ICE。当 INT19H 被破坏时可用此命令来重新引导系统和再次装入 Soft－ICE。

例：

```
HBOOT
```

此命令相当于按 RESET 开关。

2.3.6 中断模式控制(MODEL CONTROL)

ACTION－－－－设置断点发生时的动作

WARN －－－－设置 DOS 或 ROM BIOS 重进入时的警告方式

BREAK －－－－在任何时侯中断跳出

I3HERE－－－－把 INT 3 指向 Soft－ICE

命令:ACTION

功能:**设置断点发生时的动作**

语法：

ACTION [INT1|INT3|NMI|HERE|interrupt－number]

interrupt－number－－任何有效中断号(00H 到 FFH)

注：

ACTION 命令决定了当断点发生时在哪里给出控制。

不带参数时，显示当前 ACTION 的设置状态。大多数动态调试程序与 Soft－ICE 共同使用时，应设置为 INT3；HERE 参数使断点发生时控制返回给 Soft－ICE；INT1 和 NMI 主要用在其它一些动态调试程序时。例如当 CODEVIEW 与 Soft－ICE 共同使用时，ACTION 状态设置为 NMI，效果最好；如果用户自己提供了断点服务程序，则用 interrupt－number 参数。

例：

ACTION HERE

当断点满足发生条件时，该命令使控制返回 Soft－ICE。

命令：WARN

功能：设置 DOS 或 ROM BIOS 重进入时的警告方式

语法：

WARN [ON|OFF]

注：

WARN 命令主要用于 Soft－ICE 与其它一些动态调试工具一起使用时。很多动态调试程序用 DOS 及 ROM BIOS 来进行屏幕输出和键盘输入。由于 DOS 和 ROM BIOS 不能重进入，所以，如果此时被中断，这些动态调试程序有可能会出现问题。

如果设置 WARN 为 ON 状态，且 ACTION 不是 HERE 状态，那么控制在交给其它调试程序之前将返回到 Soft－ICE。此时 Soft－ICE 会显示当前 CS：IP，并让你选择是继续执行还是返回 Soft－ICE，一般情况下你应选责返回 Soft－ICE，如果你确信不会引起 DOS 或 ROM BIOS 重进入的问题则选择继续执行；若不带参数，将把当前 WARN 的状态显示出来。

默认的 WARN 状态是 OFF。

例：

WARN ON

这条命令开放 DOS 和 ROM BIOS 重进入警告方式。

命令：BREAK

功能：在任何时候中断跳出

语法：

BREAK [ON|OFF]

注：

当 BREAK 状态设置为 ON 时，允许在任何时候通过热键引入 Soft－ICE 窗口，即使系统是被挂起的或正在中断处理程序当中；不带参数时将显示当前 BREAK 状态。

默认的 BREAK 状态是 OFF。

例：

BREAK ON

这条命令开放中断方式，允许在任何时候引入 Soft－ICE 窗口。

命令：I3HERE

功能：把 INT 3 指向 Soft－ICE

语法：

I3HERE [ON|OFF]

注：

当 I3HERE 设置为 ON 时，INT 3 中断将指向 Soft－ICE。此时可放一条 INT 3 指令（机器代码为 CCH）到你想中断的代码处。当执行到 INT 3 时，将引入 Soft－ICE 窗口。如果不带参数，则显示当前 I3HERE 的状态。

默认的 I3HERE 状态是 OFF。

例：

I3HERE ON

这条命令打开 I3HERE 方式。任何 INT 3 类中断将引入 Soft－ICE。

2.3.7 特定控制命令（CUSTOMIZATION COMMANDS）

PAUSE	————每页滚屏后暂停
ALTKEY	————设置/显示激活 Soft－ICE 的热键序列
FKEY	————设置/显示功能键
BASE	————设置/显示当前的基数
CTRL＋P	————接通/关闭打印机
PRN	————设置/显示打印机输出口
TABS	————设置/显示 TAB 键状态
PRINT－SCREEN KEY	————打印当前屏幕上的内容

命令：PAUSE

功能：每页滚屏后暂停

语法：

PAUSE [ON|OFF]

注：

PAUSE 命令用于控制屏幕在每页翻滚之后暂停。

如果 PAUSE 设置为 ON，则屏幕输出满一屏后将暂停，并提示你敲任意一键继续；若不带参数则显示当前 PAUSE 的状态。

PAUSE 的缺省状态为 ON。

例：

PAUSE ON

这条命令指定在以后的窗口信息显示时，输出每满一窗口暂停，并等待你敲任意一键继续显示。

命令：ALTKEY

功能：设置/显示激活 Soft－ICE 的热键序列

语法：

ALTKEY [ALT letter]|[CRTL letter]|[SysRq]

letter－－字母(A－Z)

注：

ALTKEY 命令使你可以改变激活 Soft－ICE 的热键序列，Soft－ICE 的热键序列可以改为 ALT＋letter，CTRL＋letter，SysRq 键。例如你有个常驻内存的程序正用 CTRL＋D 热键来激活，那么你可以用 ALTKEY 命令把 Soft－ICE 的热键序列改为组合键 ALT＋D，以避免冲突。

若不带参数将显示当前 Soft－ICE 热键序列的设置。

例：

```
ALTKEY ALT D
```

这条命令把 Soft－ICE 的热键序列改成 ALT＋D。

命令：FKEY

功能：设置/显示功能键

语法：

FKEY [function－key－name string]

function－key－name－－功能键名(F1,F2,…,F12)

string －－由任何有效的 Soft－ICE 命令和分号以及特殊字符 ^ 组成的字符串。

注：

FKEY 命令用于定义功能键 F1,F2,…,F12。

string 串中的分号(;)表示回车；特殊字符 ^ 表示使其后的命令不可见。若 string 串为空串则表示取消功能键以前的定义。

如果 FKEY 不跟参数则显示当前所有功能键的定义值。

例：

```
FKEY F1 H;
```

这条命令把功能键 F1 定义为命令 H 加回车。以后当--按下 F1 键，就相当于敲入 H 命令，寻求帮助。

```
FKEY F2 G CS:100;R;P;T;
```

该命令把功能键 F2 定义为一串命令。按下 F2 就相当于输入这一系列命令。

命令：BASE

功能：设置/显示当前的基数

语法：

BASE [10|16]

注：

BASE 命令可以把当前基数设置为十进制或者十六进制。如果当前基数为 10，则所有 Soft－ICE 输入和显示的地址及数字均用十进制数表示。若当前基数为 16，则除源程序行号和 WIN 命令中屏幕的坐标及尺寸参数外，其余地址和数字都用十六进制数表示。

如果不带参数则显示当前基数设置。

BASE 命令的缺省值为 16。

例：

BASE 10

这条命令把当前基数设为 10。

命令：CTRL＋P

功能：接通/关闭打印机

语法：

CTRL＋P

注：

当第一次按下 CTRL 键再按 P 键(CTRL＋P)后，以后所有在屏幕上显示的信息都将被送到打印机。如果再次按下 CTRL＋P，则关闭打印机。

例：

CTRL＋P

切换打印机联结状态。

命令：PRN

功能：设置/显示打印机输出口

语法：

PRN [LPTx|COMx]

x——从 1 到 4 之间的任意一个数字

注：

PRN 命令可以指定打印输出口。

如果不带参数则显示当前所设置的打印输出口。

例：

PRN COM2

这条命令指定串行 2 口(COM2)为打印输出口。

命令：TABS

功能：设置/显示 TAB 键状态

语法：

TABS [2|4|8]

注：

TABS 命令用于指定 TAB 键代表几个空格。

如果不带参数则显示当前所设置的 TAB 键表示的空格数。

例：

```
TABS 8
```

这条命令使你一按 TAB 键光标就跳过 8 个空格。

命令：PRINT－SCREEN KEY

功能：**打印当前屏幕上的内容**

语法：

```
Print Screen
```

注：

当按下 Print Screen 键时，屏幕上的内容将被送到打印机上打印(包括边界字符)。

例：

```
Print－Screen
```

打印当前屏幕的内容。

2.3.8 常用命令(UTILITY COMMANDS)

A－－－－汇编代码

S－－－－搜索数据

F－－－－用数据填写内存

M－－－－移动数据块

C－－－－比较两数据块

命令：A

功能：**汇编代码**

语法：

```
A [address]
```

注：

A 命令和 DOS 的 DEBUG 中的 A 命令一样，允许你直接在内存中汇编指令。敲入 A 命令后 Soft－ICE 将等待你输入汇编指令，每输入一条后按 ENTER 开始输入下一条，所有的都输完后再敲 ENTER 则退出汇编状态。

A 命令支持 8086 指令以及 80186 和 80286 实地址方式的指令，但不支持数字协处理器指令和 80386 特定指令。

如果不带地址参数则从当前 CS：IP 处开始汇编。

例：

```
A CS:100
```

从当前代码段偏移 100H 处开始汇编指令。

命令:S

功能:**搜索数据**

语法:

S address L length data－list

length　——字节长度

data－list——列出的数据或者由逗号或空格分开的引用字符串表(用双引号或单引号均可)

注:

S 命令可在指定的范围内搜索指定数据或字串。搜索从所给出的地址开始,并且继续在描述的长度内进行。每一个在指定范围内发现的地址将被显示出来。

例:

S DS:0 L CX 'Soft－ICE',0

这条命令从当前数据段 0 偏移处开始查找一个 ASCIIZ 串(以 0 结尾的字符串 Soft－ICE),并且在检索 CX 个字节后结束。

命令:F

功能:**用数据填写内存**

语法:

F address L length data－list

length　——字节长度

data－list——列出的数据(由逗号或空格分开)或者引用字符串表(用双引号或单引号括起来)

注:

F 命令以一系列的字节或在 data－list 中描述的字符填写内存。从指定的地址处开始填写内存,并且继续到描述的长度 length,如果需要的话重复 data－list。

例:

F ES:0 L 10 'data'

这条命令用字符串 data 来填写从 ES:0 处开始的 10H 个字节的内存。字符串 data 重复进行直到达到所需的填写长度时。

命令:M

功能:**移动数据块**

语法:

M strat－address L length end－address

length——字节长度

注:

M 命令把长度为 length 的数据块从开始地址(start－address) 移到结束地址 (end－address)处。

例：

M 0:0 L 100 DS:0

这条命令把中断向量表(从 0:0H 开始到 0:FFH 的 100H 个字节)移到当前数据段的 0 偏移处。

命令:C

功能:比较两数据块

语法:

C address1 L length address2

length——字节长度

注：

C 命令比较从第一个地址(address1)开始和从第二个地址(address2) 开始的两个指定长度(length)的数据块,并且显示所有不同处的地址和内容。

例：

C DS:0 L 20 ES:0

这条命令比较从 DS:0 开始和从 ES:0 开始的长度为 20H 的两个数据块。

2.3.9 行编辑键的使用(LINE EDITOR KEY USAGE)

↓	————显示下一条命令
↑	————显示以前的命令
←	————光标左移
→	————光标右移
Backspace	————向左退一格并删除一个字符
HOME	————移动光标到行首
END	————移动光标到行末
INS	————插入/改写方式的切换
DEL	————删除当前字符
ESC	————取消当前命令

命令:↓

功能:显示下一条命令

语法:

↓

注：

显示下一条命令。类似于 DOSKEY。

例：

↓

命令：↑
功能：显示以前的命令
语法：

↑

注：
显示以前的命令。类似于 DOSKEY。
例：

↑

命令：←
功能：光标左移
语法：

←

注：
使光标向左移动。
例：

←

命令：→
功能：光标右移
语法：

→

注：
使光标向右移动。
例：

→

命令：Backspace
功能：向左退一格并删除一个字符
语法：

Backspace

注：
删除以前的字符。
例：

Backspace

命令：HOME
功能：移动光标到行首
语法：

HOME

注：

移动光标到当前行的开头。

例：

HOME

命令：END

功能：移动光标到行末

语法：

END

注：

移动光标到当前行的末尾。

例：

END

命令：INS

功能：插入/改写方式的切换

语法：

INS

注：

如果原来是插入方式，按下 INS 键后变为改写方式；如果原来是改写方式，按下 INS 键后变为插入方式。

例：

INS

命令：DEL

功能：删除当前字符

语法：

DEL

注：

删除当前光标所在处的字符。

例：

DEL

命令：ESC

功能：取消当前命令

语法：

ESC

注：

取消当前命令输入行。

例：

ESC

2.3.10 窗口命令(WINDOW COMMAND)

WC	————打开/关闭/设置代码窗口
WD	————打开/关闭/设置数据窗口
WR	————打开/关闭寄存器窗口
EC	————进入/退出代码窗口
.	————找出当前指令

命令:WC

功能:**打开/关闭/设置代码窗口**

语法:

WC [window—size]

window—size————窗口尺寸(1 到 21 之间的十进制整数)

注:

如果没有指定窗口尺寸,那么,WC 命令将使代码窗口在可见和不可见之间切换。如果指定了窗口尺寸,那么,代码窗口将可见且窗口大小被重新设置为指定值。

例:

WC

若代码窗口原可见则消失;若代码窗口原不可见则弹出代码窗口。

WC 12

若代码窗口不存在则产生一个高为 12 行的代码窗口;若代码窗口已存在则重新设置其高度为 12 行。

命令:WD

功能:**打开/关闭/设置数据窗口**

语法:

WD [window—size]

window—size————窗口尺寸(1 到 21 之间的十进制整数)

注:

如果没有指定窗口尺寸,那么,WD 命令将使数据窗口在可见和不可见之间切换。如果指定了窗口尺寸,那么,数据窗口将可见且窗口大小被重新设置为指定值。

例:

WD

若数据窗口原可见则消失;若数据窗口原不可见则弹出数据窗口。

WD 12

若数据窗口不存在则产生一个高为 12 行的数据窗口；若数据窗口已存在则重新设窗口高为 12 行。

命令：WR

功能：**打开/关闭寄存器窗口**

语法：

WR

注：

若寄存器窗口原可见则消失；若寄存器窗口原不可见则弹出寄存器窗口。

例：

WR

打开或关闭寄存器窗口

命令：EC

功能：**进入/退出代码窗口**

语法：

EC

注：

EC 命令在代码窗口和命令窗口之间切换光标的位置。如果光标在命令窗口里，那么它将被移到代码窗口；如果光标在代码窗口里，那么它将被移到命令窗口。

代码窗口可见时，EC 命令才有效。

例：

EC

进入或者退出代码窗口。

命令：.

功能：**找出当前指令**

语法：

.

注：

. 命令使当前源程序行或当前指令可见。

例：

.

这条命令使当前源程序行或当前指令可见。

2.3.11 窗口/屏幕控制(WINDOW CONTROL)

CTRL＋↓ ————向下移动窗口

CTRL＋↑ ————向上移动窗口

CTRL＋← ————向左移动窗口
CTRL＋→ ————向右移动窗口
ALT＋↓ ————使窗口变矮
ALT＋↑ ————使窗口变高
CLS ————清除窗口
RS ————恢复应用程序屏幕
WIN ————改变 Soft－ICE 的总体窗口大小
ALTSCR ————切换到另外一个屏幕
FLASH ————在 P 和 T 过程中恢复应用程序屏幕
FLICK ————屏幕抖动处理
WATCHV ————设置监视视频方式

命令:CTRL＋↓
功能:**向下移动窗口**
语法:
CTRL＋↓
注:
将整个窗口向下移动。

命令:CTRL＋↑
功能:**向上移动窗口**
语法:
CTRL＋↑
注:
将整个窗口向上移动。

命令:CTRL＋←
功能:**向左移动窗口**
语法:
CTRL＋←
注:
将整个窗口向左移动。

命令:CTRL＋→
功能:**向右移动窗口**
语法:
CTRL＋→
注:
将整个窗口向右移动。

命令:ALT＋↓
功能:**使窗口变矮**

语法：

ALT＋↓

注：

使窗口的高度变矮。

命令：ALT＋↑

功能：使窗口变高

语法：

ALT＋↑

注：

使窗口的高度变高。

命令：CLS

功能：清除窗口

语法：

CLS

注：

CLS 命令清除 Soft－ICE 的命令窗口，并且把提示符(:)和光标移到窗口的左上角。

命令：RS

功能：恢复应用程序屏幕

语法：

RS

注：

RS 命令可以暂时恢复应用程序的屏幕。等你按任意一键后，又回到 Soft－ICE 的窗口状态。该命令在调试图形程序时很有用。

命令：WIN

功能：改变 Soft－ICE 的总体窗口大小

语法：

WIN [N|W] [start－row length [start－column]]

N	——窄方式(46 个字符宽)
W	——宽方式(全屏幕方式)
start－row	——窗口起始行(从 0 到 17 之间)
length	——窗口高度(8 到 25 行之间)
start－column	——窄方式时的窗口起始列

注：

WIN 命令可在多种屏幕方式之间切换。

例：

WIN N 3 10 40

这条命令使窗口显示从第 3 行，第 40 列开始，高 10 行，宽 36 个字符。

```
WIN W 2 15
```

这条命令使窗口显示从第 2 行开始，高 15 行，宽度为整个屏幕。

命令：ALTSCR

功能：**切换到另外一个屏幕**

语法：

```
ALTSCR [ON|OFF]
```

注：

ALTSCR 命令用于有双显示器时。它能够把 Soft－ICE 的输出从你默认的屏幕切换到另一个屏幕。

ALTSCR 的默认状态为 OFF。

例：

```
ALTSCR ON
```

这条命令可以把屏幕输出切换到另一个显示器上。

命令：FLASH

功能：**在 P 和 T 过程中恢复应用程序屏幕**

语法：

```
FLASH [ON|OFF]
```

注：

FLASH 命令允许你指定在执行 P 和 T 过程时是否暂时恢复应用程序屏幕。

当不带参数时将显示当前 FLASH 状态。

FLASH 的默认状态是 OFF 方式。

例：

```
FLASH ON
```

这条命令接通 FLASH 方式。在后面任何 P 或 T 命令过程中，应用程序所显示的屏幕将被短暂地恢复。

命令：FLICK

功能：**屏幕抖动处理**

语法：

```
FLICK [ON|OFF]
```

注：

如果你的屏幕显示有抖动等问题时，可把 FLICK 设置成 ON 状态来解决，但屏幕滚动将变慢。

若不带参数将显示 FLICK 的当前状态。

FLICK 的默认状态是 OFF。

例：

```
FLICK ON
```

进行屏幕抖动处理。

命令:WATCHV

功能:**设置监视视频方式**

语法:

WATCHV [ON|OFF]

注:

如果你发现 Soft－ICE 没有很好地控制屏幕,或者光标没有被正确地恢复,可以通过接通 WATCHV 方式来解决。

若不带参数将显示 WATCHV 的当前状态。

WATCHV 的默认状态是 OFF。

例:

WATCHV ON

这条命令接通 WATCHV 方式,使 Soft－ICE 能监视其视频口。

2.3.12 符号和源程序调试命令(SYMBOL/SOURCE COMMANDS)

SYM	————显示/设置符号
SYMLOC	————置换符号基数
SRC	————在源程序和混合代码方式之间切换
FILE	————改变/显示当前源文件
SS	————在当前源文件中搜寻字串
CALLS	————显示调用堆栈
TABLE	————选择符号表

命令:SYM

功能:**显示/设置符号**

语法:

SYM [symbol－name [value]]

symbol－name ——符号名称(星号和问号分别对应于 DOS 下的通配符 * 和 ?)

value ——符号值

注:

SYM 命令允许显示和设置符号。

如果不带参数,将显示所有符号的符号名称和该符号的值;若仅有符号名称而无符号值,Soft－ICE 将首先显示该符号的值,然后等待你赋予一个新值;若既有符号名又有符号值,则将此值赋予该符号。

通配符 * 号只能够放在末尾,表示一个或多个任意字符,也可以表示没有字符。通配符 ?

号可放在任意位置，表示任意一个字符。

例：

```
SYM
```

显示的符号信息如下：

```
  CODE(0518)
   0518:0000  START          0518:0007  MAIN_LOOP
   0518:001D  NO_SPACE       0518:0026  GET_KEY
   0518:002E  IS_SPACE       0518:003B  NOT_SPACE
   0518:0042  HANG_EXAMPLE
 DATA(051D)
   051D:0000  PAD            051D:0012  CHAR
   051D:0013  ANSWER         051D:0014  SPACE_MSG
   0518:002F  NO_SPACE_MSG
```

又如：

```
SYM A*
```

显示所有以 A 打头的符号。

```
SYM A* 90
```

这条命令给所有以 A 打头的符号赋予值 90H。

```
SYM ABC
```

显示符号 ABC 的值，并等待输入新值。

命令：SYMLOC

功能：**置换符号基数**

语法：

SYMLOC segment—address

注：

SYMLOC 命令置换与指定的段地址有关的所有符号的段地址部分。当调试可装入设备驱动程序和其它不能用 LDR.EXE 直接装入的程序时，这个命令非常有用。

例：

```
SYMLOC 1000+10
```

这条命令置换符号表中与 1000H 有关的所有段。+10 用来重新配置起初是一个 .EXE 文件的 TSR。

命令：SRC

功能：**在源程序和混合代码方式之间切换**

语法：

SRC [?]

注：

SRC 命令在代码窗口的源文件状态，混合状态和代码状态之间进行切换。

如果用 ? 参数，则显示当前的状态。

例：

```
SRC
```

这条命令改变代码窗口的当前状态。如果当前状态是源文件状态，那么变为混合状态；如果当前状态是混合状态，那么变为代码状态；如果当前状态是代码状态，那么变为源文件状态。

命令：FILE

功能：**改变/显示当前源文件**

语法：

```
FILE [file—name | * ]
```

注：

如果指定了文件名，那么该文件成为当前源文件，并在代码窗口显示此文件，源文件的扩展名可以不必指出。如果没有指定文件名，那么当前源文件的文件名将被显示。如果用星号参数(*)将显示所有被装入内存中的源文件。

例：

```
FILE
```

显示当前源文件的文件名。

```
FILE ABC
```

如果内存中已装入了一个名为 ABC.C 的源文件，则把当前源文件切换为 ABC.C。

命令：SS

功能：**在当前源文件中搜寻字串**

语法：

```
SS [line—number] ['string']
    line—number——行号(一个十进制数)
    string       ——由双引号或单引号括起来的字符串
```

注：

SS 命令在当前源文件中搜索指定的字符串。如果有一个匹配的字符串，那么字符串所在的行将显示在代码窗口的最上面一行。

从指定的行号(line—number)开始搜索。如果没有指定行号，那么从代码窗口显示的最上面一行开始搜索。

如果没有指定字串参数，则继续搜索以前指定的字符串。

例：

```
SS 2 'MOV AX,2'
```

搜索当前源文件，从第二行开始找字串 'MOV AX,2'，包含该字串的行显示在代码窗口的最

上面一行。

命令:CALLS
功能:**显示调用堆栈**
语法:

CALLS

注:
这条命令用于察看堆栈的状况。
例:

CALLS

命令:TABLE
功能:**选择符号表**
语法:

TABLE [1|2]

注:
TABLE 命令使你可在两个符号表之间切换。
例:

TABLE 2

选取第二个符号表作为当前符号表。

2.3.13 回溯跟踪命令(BACK TRACE COMMANDS)

SHOW	————显示回溯跟踪缓冲区中的指令
TRACE	————进入仿真跟踪方式
XT	————在仿真跟踪方式下单步执行
XP	————在仿真跟踪方式下执行一条程序
XG	————在仿真跟踪方式下转向地址
XRSET	————重新设置回溯跟踪缓冲区

命令:SHOW
功能:**显示回溯跟踪缓冲区中的指令**
语法:

SHOW [B|start]

B ——从跟踪缓冲区中最早的指令开始显示
start——从跟踪缓冲区末尾开始往前显示指令

注:
SHOW 命令从回溯跟踪缓冲区中按照当前代码窗口的方式显示指令。
例:

```
SHOW
```

如果当前代码窗口处在汇编方式下，则显示的信息如下：

```
          0005  MAIN_LOOP
  0005  0518:0007  EB1C00     CALL GET_KEY
      0004  GET_KEY
      0004  0518:0026  B408       MOV  AH, 08
      0003  0518:0028  CD21       INT  21
      0002  0518:002A  A21200     MOV  [CHAR], AL
      0001  0518:002D  C3         RET
```

又如：

```
SHOW 40
```

这条指令将在从回溯跟踪缓冲区中的第 40 条指令开始往前显示。

命令：TRACE
功能：**进入仿真跟踪方式**
语法：

TRACE [start]|[OFF]
start——从缓冲区尾到缓冲区开始回溯跟踪的指令数
OFF ——退出仿真跟踪方式

注：
TRACE 命令允许你从回溯跟踪缓冲区中仿真执行以前执行过的指令。
用 OFF 参数退出仿真跟踪。
无参数时显示仿真跟踪方式是开还是关。
例：

```
TRACE 40
```

这条命令进入仿真跟踪方式并回溯到上次记录的指令以后的 40 条指令。

命令：XT
功能：**在仿真跟踪方式下单步执行**
语法：

XT [R]
R——向相反方向单步执行

注：
XT 命令通过回溯跟踪缓冲区单步执行。
除 CS:IP 外，其余各寄存器将保持不变。
例：

```
XT
```

这条命令在仿真跟踪方式下单步执行一条指令。

命令:XP

功能:在仿真跟踪方式下执行一条程序

语法:

XP

注:

XP 命令通过回溯跟踪缓冲区执行一步程序。

除 CS:IP 外,其余各寄存器将保持不变。

例:

XP

这条命令在仿真跟踪方式下执行一步程序。

命令:XG

功能:在仿真跟踪方式下转向地址

语法:

XG [R] address

R ——向相反方向检索地址

address——回溯跟踪缓冲区地址

注:

XG 命令在回溯跟踪缓冲区移动指令指针到下一条出现的描述地址。

如果 R 参数在地址之前描述,那么指令指针应在回溯跟踪缓冲区中移到出现的描述地址之前。

例:

XG 1000:100

这条命令仿真执行到地址 1000:0100H 处。

命令:XRSET

功能:重新设置回溯跟踪缓冲区

语法:

XRSET

注:

XRSET 命令重新设置回溯跟踪缓冲区。如果有不需要的指令信息在回溯跟踪缓冲区中,这条命令可以清除当前回溯跟踪缓冲区里保存的内容。

例:

XRSET

这条命令重新设置回溯跟踪缓冲区。

2.3.14 特殊命令(ADVANCED COMMANDS)

VECS	————保存/装入/比较中断向量表
SNAP	————备份/恢复/比较内存块
SERIAL	————控制台重定向
EMMMAP	————显示 EMM 分配映象
BOUNDS	————界限检测

命令:VECS

功能:**保存/装入/比较中断向量表**

语法:

VECS [C|S|R]

C——比较当前中断向量表和存储的中断向量表

S——保存当前的中断向量表到缓冲区中

R——从缓冲区中恢复中断向量表

注:

VCES 命令允许你保存并且存储中断向量表至 Soft－ICE 的内部缓冲区(用 S 参数);用 R 参数时,将从缓冲区中恢复中断向量表,即把保存的中断向量表重新写回 BIOS 数据区(0000:0000H 至 0000:00FFH);若用 C 参数则将把当前中断向量表和保存在 Soft－ICE 内部缓冲区中的中断向量表进行比较,并用以下格式把每一个改变了的中断向量显示出来:

address old－vector new－vector

如果没有描述参数,则显示整个中断向量表。

Soft－ICE 装入时,中断向量表被自动存储;当用 LDR.EXE 装入程序时,它也可以自动存储。而且仅有一个中断向量表的拷贝被存储,所以每次执行 VCES S 时,以前的中断向量表拷贝将被覆盖。

例:

VCES S

这条命令把当前的中断向量表做一个备份,存到 Soft－ICE 的内部缓冲区中去,以便以后恢复和比较用。

命令:SNAP

功能:**备份/恢复/比较内存块**

语法:

SNAP [C|S|R] address1 address2

C——比较内存块

S——保存内存块

R——恢复内存块

注：

SNAP 命令允许你保存某一指定的内存块的备份到回溯跟踪缓冲区中，以便以后用于比较和恢复。

S 参数把从 address1 开始到 address2 为止的内存块保存到回溯跟踪缓冲区中去。

R 参数把保存在回溯跟踪缓冲区中的内存块备份重新写回从 address1 开始到 address2 为止的内存块。

C 参数将比较从 address1 开始到 address2 为止的内存块和保存在回溯跟踪缓冲区中的内存块备份是否相同，如果不同将用以下格式显示每一个改变了的字节：

address　old－data　new－data

C 和 R 参数通常不需要带地址参数。如果没有地址描述，用上次带有描述地址的 SNAP 命令的地址。

因为 SNAP 命令要用到回溯跟踪缓冲区，所以最好是在 CONFIG.SYS 的 S－ICE.EXE 安装行中加上 /TRA nnnn 开关，以指定足够大的缓冲区。而且，所保存的内存块会和记录的回溯跟踪信息互相覆盖。

例：

SNAP S 0000:0000 0000:0400

这条命令把从 0000:0000H 到 0000:0400H 的 BIOS 数据区（中断向量表）做个备份，并保存到回溯跟踪缓冲区中。

命令：SERIAL

功能：**控制台重定向**

语法：

SERIAL [ON|OFF]

注：

SERIAL 命令用于远程调试。

ON 参数将激活远程串行通信口；OFF 参数关闭远程串行通信口。

如果不带任何描述则显示当前远程串行通信口的状态。

例：

SERIAL ON

这条命令接通远程串行通信口。

命令：EMMMAP

功能：**显示 EMM 内存分配映象**

语法：

EMMMAP

注：

EMMMAP 命令将显示 EMM 内存和当前正在映象的页的每一个有效的物理页。

例：

EMMMAP

此例执行后显示如下：

phr page	seg address	Handle/page
00	E000	FFFF
01	E000	FFFF
02	E000	FFFF
03	E000	FFFF
04	1000	0000/0000
05	1400	0000/0001
06	1800	0000/0002
07	1C00	0000/0003
08	2000	0000/0004
.	.	.
.	.	.
.	.	.

在这个例子中，物理页 0,1,2,3 定位于 E000H 并且是不映象的；物理页 4 定位于 1000H 并且挂起 0，页面 0 在它内部映象；物理页 5 定位于 1400H 并且挂起 0，页面 1 在它内部映象；物理页 6 定位于 1800H 并挂起 0，页面 2 在它内部映象；物理页 7 定位于 1C00H 并挂起 0，页面 3 在它内部映象；物理页 8 定位于 2000H 并挂起 0，页面 4 在它内部映象。

命令：BOUNDS

功能：**界限检测**

语法：

BOUNDS [ON|OFF]

注：

BOUNDS 命令用于控制界限检测状态。

ON 参数将打开界限检测；OFF 参数则关闭界限检测。

如果不带任何描述则显示当前界限检测状态。

例：

BOUNDS ON

这条命令开放界限检测。

3 Soft－ICE 使用

3.1 Soft－ICE 的初始化配置

Soft－ICE 可以通过一个名为 S－ICE.DAT 的 ASCII 文本文件，指定各种初始化配置。当用命令行装入 Soft－ICE 时，S－ICE.DAT 必须在当前目录或者放在可搜索到的路径下。当 Soft－ICE 作为设备驱动程序装入 CONFIG.SYS 中时，S－ICE.DAT 必须与 S－ICE.EXE 在同一目录下，才起作用。

在 S－ICE.DAT 初始化文件中，可进行四方面的初始化工作：特殊配置选择，指定功能键，初始化命令序列，屏幕颜色控制。

3.1.1 特殊配置选择

任何一种特殊配置选择均应在 S－ICE.DAT 文件中单独占一行。

1) COMPAQ —— COMPAQ 386 以及 COMPAQ 的兼容机(包括主板是 Micronix 的计算机) 其高端的 384K 扩充存储器是不连续的，不同于其它机型的使用协定。如果你想在这类机型上使用 Soft－ICE，并且使用这个扩充存储器，那么就应使用 COMPAQ 选择。该选择和在命令行方式下运行 S－ICE.EXE 的 /C 参数一样。

2) NOLEDS ——这项选择使 Soft－ICE 在窗口弹出时，不设置和清除键盘上的 LED 灯。Soft－ICE 在一些键盘上会有同步问题，会出现键盘锁死或延迟，那么，试一下 NOLEDS 选择，来解决问题。该命令和在命令行方式下运行 S－ICE.EXE 的 /L 参数一样。

3) NOTVGA ——如果你的 VGA 卡在硬件级上与 IBMVGA 不兼容，而只在 BIOS 级上兼容，因为 Soft－ICE 不使用 BIOS 进行显示输出，所以会出现显示问题，这时，建议你使用 NOTVGA 开关。该选择和在命令行方式下运行 S－ICE.EXE 的 /V 参数一样。

4) EXTENDED ——当装入 Soft－ICE 时，它会占用扩展存储器的高端，在此之前，它会报警，让你确定是否继续安装，并覆盖扩展存储器的高端。如果你确定没有其它程序使用扩展存储器高端，又不想在 DOS 下装入时被提示，那么应使用 EXTENDED。该选择和在命令行方式下运行 S－ICE.EXE 的 /E 参数一样。

3.1.2 功能键的指定

在装入时，可以把功能键指定成一条或多条命令，这样，按下某一功能键，可相当于执行了一条或多条 Soft－ICE 命令。

在 S－ICE.DAT 中，指定功能键的格式为：

功能键名="描述字符串"

功能键是 F1,F2,.....,F12。

描述字符串由任何有效的 Soft-ICE 命令,特殊字符(^)和分号(;)组成。特殊字符(^)使命令不显示,分号代表回车,描述字符串必须用双引号括起来。使用功能键只需很简单地按下功能键而不需输入命令,而且当你正在敲入命令时,也可使用功能键,而不影当前输入。例如你想输入 HBOOT 命令,刚敲入 HB 两个字符,这时,若你想单步执行一条指令,可按 F8(F8 被缺省指定为单步执行命令 T 加回车,且不显示,即 ^ T),单步执行后,接着敲入 OOT 三个字符,完成 HBOOT 命令的输入。

各功能键在 S-ICE.DAT 中的缺省指定为:

F1="H;"	显示帮助信息
F2="^ WR;"	切换寄存器窗口
F3="^ SRC;"	切换当前的源程序方式
F4="^ RS;"	恢复程序屏幕
F5="^ X;"	退出 S-ICE,返回你的应用程序
F6="^ EC;"	在命令窗口和代码窗口之间切换
F7="^ HERE;"	转到当前光标行
F8="^ T;"	单步执行
F9="^ BPX;"	在当前光标行设置/清除断点
F10="^ P;"	执行一条程序
F11="^ G@SS:IP;"	转向返回地址
F12="^ VER;"	显示 Soft-ICE 的版本号

注:F1 的定义中,不带特殊符号(^),是因为(^)不仅不显示其后的命令字符,也屏蔽了显示在命令窗口的信息(包括错误信息),所以,如果改成"^ H;",那么将看不到帮助信息。

用 FKEY 命令可以显示或改变各功能键的指定值。

3.1.3 初始化命令序列

当 Soft-ICE 装入时,将自动地执行初始化命令序列。它可用来执行一系列的准备命令使 Soft-ICE 完成一些初始设置,例如改变热键和定义窗口等。

在 S-ICE.DAT 中建立初始化序列的格式为:

INIT="命令字串"

命令字串的组成与指定功能键的描述字串一样。

一个典型的初始化例子是:

```
INIT="WIN;WR;WD2;WC10;ALTKEY ALT D;"
```

该例首先把 Soft-ICE 窗口置为全屏幕方式,然后产生寄存器窗口,产生一个两行高的数据窗口和一个 10 行高的代码窗口,并把热键序列改为 ALT+D。

3.1.4 屏幕颜色控制

如果你使用的是一个彩色显示器，那么，通过 S－ICE.DAT 中的屏幕颜色控制参数，可以设置窗口底色，字符色，以及字符背景色。

在 S－ICE.DAT 中，进行颜色初始化控制的格式为：

COLORS＝"寄存器窗口底色，寄存器窗口字符色，寄存器窗口字符背景色， 数据窗口底色，数据窗口字符色，数据窗口字符背景色， 代码窗口底色，代码窗口字符色，代码窗口字符背景色， 命令窗口底色，命令窗口字符色，命令窗口字符背景色， 提示行底色，提示行字符色，提示行字符背景色。"

底色是指整个窗口中空白处的颜色，字符色指所显示的字符的颜色，字符背景色是指字符加框时的颜色。各种颜色以十六进制数表示，符合彩色适配器的调色标准和字符属性的设置标准。

属性字节：位

7	6	5	4	3	2	1	0

位	说明
7	该位置 1 用于字符闪烁，置 0 字符不闪烁。
6——4	背景色。可用的是：
6 5 4	颜色
0 0 0	黑色
0 0 1	蓝色
0 1 0	绿色
0 1 1	青色
1 0 0	红色
1 0 1	绛色
1 1 0	褐色
1 1 1	浅灰
3	前景字符的亮度。该位置 1，前景字符为高亮度，置 0，前景字符为正常亮度。
2——0	前景色。可用的彩色组合同 6 到 4 位的背景色。

一个典型的例子是：

COLORS＝"47H，4EH，7EH， 07H，0FH，70H， 17H，1FH，71H， 30H，3FH，71H"

该例将产生红色的寄存器窗口，黑色的数据窗口，蓝色的代码窗口，淡绿色的命令窗口，以及白色的命令提示行。

3.2 Soft－ICE 的装入开关

在安装时，可在 S－ICE.EXE 后带一个或多个开关，也可不带开关。在 CONFIG.SYS 中装入和直接在命令行方式下运行 S－ICE.EXE 的装入开关是有区别的。

3.2.1 在 DOS 提示符下装入 Soft－ICE 的开关选择

在 DOS 提示符下运行 S－ICE.EXE，有四个开关可供选择，输入 S－ICE.EXE /? 可得到帮助信息：

```
    /?   This message
/U  Uninstall Soft－ICE
/E  Use extended memory regardess
/C  Use Compaq extended memory
/L  Do not handle the keyboard LEDS
```

1) /? ——显示上述帮助信息。

2) /U ——释放装入的 Soft－ICE。如果你不使用 Soft－ICE，想将其从内存中释放，或要运行带 80286 或 80386 保护方式指令的程序，可以用该参数退出，并释放 Soft－ICE 所占用的内存空间。无论是在 CONFIG.SYS 中装入或在 DOS 提示符下装入的 Soft－ICE，都可用 /U 开关来释放内存。

3) /E ——直接占用扩展存储器而不提示。如果你确定要覆盖存储器的高端，而又不想被提示，可使用该开关。/E 开关与初始化文件 S－ICE.DAT 中的配置选择项 EXTENED 作用一样。

4) /C ——使用 Compaq 系列计算机的高端 384K 不连续扩展存储器。如果你的计算机是 Compaq 及其兼容机，或者 Micronix 主板的机型，要使用其扩展存储器，就必须带此开关。/C 开关与初始化文件 S－ICE.DAT 中的配置选择项 COMPAQ 作用一样。

5) /L ——不处理键盘上的 LED 灯。有一些键盘有同步问题使得 Soft－ICE 不能与键盘同步，如果当你在 Soft－ICE 窗口时，Soft－ICE 出现延迟现象，那么应使用这个开关。/L 开关与初始化文件 S－ICE.DAT 中的配置选择项 NOLEDS 作用一样。

3.2.2 在 CONFIG.SYS 中装入时的开关选择

1) 开关

在配置文件 CONFIG.SYS 中，S－ICE.EXE 后面可跟一个或多个装入开关，这些开关能限定 Soft－ICE 使用扩展内存的方式，每个开关必须以 / 开头。

(1) /EXT nnnn ——为其它使用扩展内存的程序(如 RAMDRIVE.SYS，PC－CACHE.EXE 等)保留 nnnn 千字节的扩展内存。虽然不使用 /EXT 开关时，Soft－ICE 及其相关保留部分不使用的扩展内存均可为其它程序所用，但数量无法保证 。例如，在某种配置下，不加 /

EXT 开关时，Soft－ICE 及其相关保留部分共占用扩展内存 184K，剩下 200K 供其它程序使用，而你刚好要运行一个必须占用 201K 扩展内存的程序，这时，你可用 /EXT 201 开关来保证为它留出 201K 的扩展内存。

(2) /SYM nnnn ——为符号和源程序处理保留 nnnn 千字节的扩展内存。如果要在符号和源程序级上进行动态调试，就必须为符号文件(.SYM)和源文件分配足够的扩展内存。如果没有指定数量 nnnn，那么所有剩下的扩展内存都用于符号和源程序的处理。

(3) /TRA nnnn ——为回溯跟踪缓冲区保留 nnnn 千字节的扩展内存，缓冲区用于记录在回溯跟踪范围内执行过的指令和 SNAP 命令所保存的内存数据块，如果没有指定具体的数量 nnnn 或 /TRA 开关，那么自动为回溯跟踪缓冲区保留 10K 的扩展内存。如果不想为回溯跟踪缓冲区保留存储器，可用 /TRA 0。

(4) /MCV nnn ——为 Magic CV 或者 Magic CVW 保留 nnn 千字节的扩展内存，nnn 的范围从最小的 280 K 到最大的 620K。如果没有指定 nnn 的具体数量，将保留剩下的扩展内存，但最大不超过 620K，最小也必须有 280K。

(5) /EMM nnnn ——把 nnnn 千字节的扩展内存变为附合 EMM4.0 标准的扩充内存。如果没有指定该选择项，那么所有剩余的存储器都被用作扩展存储器。

(6) /UN ——仅为 Soft－ICE 保留空间，但并不装入。如果你启动系统后，先要运行某些和 Soft－ICE 冲突的程序，然后再运行 Soft－ICE，那么可用此开关。/UN 开关首先保留 Soft－ICE 及其相关部分所占用的存储空间，但并不装入自己，仅仅是留出空间备用。当运行完和 Soft－ICE 冲突的程序后，你就可以直接在 DOS 提示符下运行 S－ICE.EXE，而不必重新启动系统，但其效果和在 CONFIG.SYS 中装入一样。

2) 存储器分配的优先顺序

以上各开关按照下列顺序保留存储器，而不管指定开关的顺序：

(1) 为 S－ICE.EXE 自己保留大约 120K。

(2) 如果开关 /EXT 存在，为它保留存储器。

(3) 如果开关 /SYM 存在，为它保留存储器。

(4) 如果开关 /TRA 存在，为它保留存储器，如果它不存在，缺省时为回溯跟踪缓冲区保留 10K。

(5) 如果开关 /MCV 存在，为它保留存储器。

(6) 如果开关 /EMM 存在，为它保留存储器。

当按照以上顺序为开关保留存储器时，如果存储器被分配完，那么 S－ICE.EXE 把剩余的扩展存储器分配给正在被处理的开关，余下的开关将不再被保留存储器。

如果 /MCV 或 /EMM 开关存在，那么将为 DMA 存储器保留 64K 的扩展存储器。

开关可以以任何顺序置于 S－ICE.EXE 后。

3) 应用举例

以下用一个例子来说明 S－ICE.EXE 所带开关的使用方法：

```
DEVICE=DRIVE:\PATH\S－ICE.EXE /EMM 400 /TRA 40 /SYM 100
```

如果你有 2M 字节的扩展存储器可利用，那么这个例子首先为 S－ICE.EXE 保留大约 120K 字节，接着为符号保留 100K 字节，再为回溯跟踪缓冲区保留 40K 字节，为其它使用扩

充存储器的程序保留 400K 字节的 EMM 内存。这样共剩下 460K 普通的扩展存储备器。因为 Soft－ICE 装入最高端，所以剩下的存储器是从 10000H（1M 字节）处开始的。

3.3 符号及源程序级的调试

把 Soft－ICE 从 CONFIG.SYS 中装入，并加上恰当的开关，在扩展内存中为符号及源程序保留足够的内存空间，那么你就可以在源程序级上进行动态调试，用符号名和行号来代替具体的数字地址作地址参数。

3.3.1 准备调试信息

在符号及源程序级上进行调试之前必须自己建立一个符号文件，这是一个二进制文件，它包含了 Soft－ICE 能识别的符号及行号信息，且扩展名为 .SYM。

1）如果你仅仅想使用符号而不需要在源程序级上进行调试，那么，请按以下步骤做：

（1）编译或汇编你的源程序，生成目标代码文件（一般情况下，扩展名为 .OBJ）。源程序中的符号必须是公共符号，即具有 PUBLIC 属性，才能被 Soft－ICE 所识别。

（2）链接上一步所生成的目标代码文件，产生可执行文件和符号映象文件（.MAP）。链接时，必须带上 /MAP 开关且链接程序 LINK.EXE 不能低于 3.60 版。

（3）利用 Soft－ICE 软件自己带的从映象文件到符号文件的转换器 MSYM.EXE，生成扩展名为 .SYM 的符号文件。运行：

```
MSYM filename[.ext]
```

如果没有指定扩展名默认的映象文件扩展名为 .MAP。

2）假如你既想使用符号，又要在源程序级上进行调试，则需按以下步骤做：

（1）编译或汇编你的源程序，生成目标代码文件，并且选择适当的开关，把符号和行号等调试信息加入到目标代码文件中去，如开关 /Zi 或 /Zd。

（2）链接目标代码程序，生成可执行文件和映象文件。当你使用 LINK.EXE 程序链接时必须使用 /Linenumber（带行号）和 /MAP（映象文件）开关，以便生成带有源程序行号和符号表的映象文件。

（3）利用 Soft－ICE 软件自己带的从映象文件到符号文件的转换器 MSYM.EXE，生成扩展名为 .SYM 的符号文件。运行：

```
MSYM filename[.ext]
```

如果没有指定扩展名默认的映象文件扩展名为 .MAP。

3.3.2 装入程序

Soft－ICE 提供了源文件，符号文件和程序的装入器 LDR.EXE。其使用方法如下：

LDR filename[.EXE|.COM|SYM]

.EXE(.COM)————Load .EXE(.COM)file only(No SYMBOLS)

.SYM————Load Symbol file only

No extension————Load both program & Symbol

如果只有程序名，而无扩展名，那么 LDR.EXE 首先把符号文件装入到扩展内存，若符号文件(.SYM)中有装入源程序的标记，则把源文件也装入扩展内存，接着将可执行文件装入当它从 DOS 提示符下直接装入时所应该装入的内存存储单元，然后，Soft－ICE 把命令指针指向你的程序的第一条指令。

当扩展名是 .SYM 时 Soft－ICE 只装入符号文件和源文件。这时，你的符号的重定位均假定以 0:0 为参考点。这种情况一般用于调试设备驱动程序或引导程序。

当扩展名是 .COM 或 .EXE 时(即可执行文件)，Soft－ICE 将只把程序装入，并把指令指针指向程序的第一条指令。

值得注意的是，当 LDR.EXE 在装入你的程序时，会自动存储中断向量表的拷贝，用命令 EXIT R 从程序中退出时，自动恢复中断向量表。

如果你在 DOS 环境下设置了一个环境变量 SRC：

SET SRC=DIR1,DIR2,…

那么，当 Soft－ICE 在当前目录下没有找到源文件，它会自动去按 DIR1,DIR2,……的路径搜索，若仍找不到，Soft－ICE 将提示你给出源文件所在的路径。

3.3.3 用符号和源程序调试

在你装入符号文件和程序之后，就可以用符号来调试你的程序了，一般来说，在任何命令中都可以用一个相应的符号名来代替实际地址。Soft－ICE 在显示地址时，也用相应的符号来表示，例如子程序名，变量名等。

当你调试一个可装入的设备驱动程序，或其它一些因为某种特殊的需要而使得符号不能自动定位的程序时，可用 SYMLOC 命令，使符号和地址正确地一一对应。

如果同时也装入了源程序，则可用行号来代替地址。例如命令：

BPX .7

表示在源程序的第七行设置或清除断点。

源程序装入后，Soft－ICE 就能提供两种调试方式：源程序行方式和混合方式。用 SRC 命令在两种方式之间切换。源程序行方式，使你可在源程序级上调试；而使用混合方式时，源程序行和由这些源程序行所生成的汇编指令代码都一起显示在代码窗口中，并且使你可以在汇编级上调试。

3.4 回溯过去的跟踪

当你跟踪某一程序时，跟踪一段时间后，由于某些原因，你可能很想回顾一下过去执行

了哪些指令流，以便更好的解决问题。其它的调试工具对此是无能为力的，只有靠自己的记忆和用笔写的记录，但是，Soft－ICE 在这方面却有杰出的表现——独特的回溯过去跟踪的功能。

你可以指定在一个不大于 1M 字节的范围内收集所执行过的指令流信息。但是，如果范围太大，会加重整个系统的负担，降低其性能，而且记录回溯跟踪信息的缓冲区是有限的，范围过大，一些使你不感兴趣的东西，诸如 DOS 内部调用，BIOS 功能调用，内部中断等指令塞满了缓冲区。所以，你应尽可能把范围缩小到你所感兴趣的区域。

为了使用 Soft－ICE 的这一功能，你要熟悉下列命令：SHOW、TRACE、XT、XP、XG 、XRSET、BPR。

1）配置系统

首先，在 CONFIG.SYS 中，加上 /TRA 开关，并分配一个足够大的回溯跟踪缓冲区。例如：

```
DEVICE=DRIVE:\PATH\S－ICE.EXE /TRA 50
```

将为回溯跟踪缓冲区保留 50K 字节的空间，如果不指定，缺省时为回溯跟踪缓冲区保留 10K 字节的空间。

2）记录指令信息

接着重新启动系统，进入 Soft－ICE，通过设置描述为 T 或 TW 的内存范围断点来确定回溯跟踪范围。例如：

```
BPR   2000:0100   2000:1E00   T
```

这条命令记录了内存范围 2000：0100H 到 2000:1E00H 内所执行过的指令的情况，以供 SHOW 或 TRACE 命令使用。

3）察看以前的跟踪过程

当你需要时，你可以用 SHOW 命令显示 Soft－ICE 在指定范围内所收集的指令信息。如果程序确实已执行过在内存范围内从 2000:0100H 到 2000:1E00H 之间的指令，那么将显示这些执行过的指令流；如果还没有执行过其间的指令，则会告诉你目前回溯跟踪缓冲区是空的，没有收集到任何信息。

```
SHOW 50
```

将显示回溯缓冲区收集的第一到第五十条指令。

4）模拟执行以前的指令

假如，你对以前执行过的指令感兴趣的话，可以用 TRACE 命令进入模拟跟踪状态。例如：

```
TRACE  50
```

将从前 50 条指令开始，模拟以前的执行情况。这时，你可根据需要，使用 XT，XP，XG 命令来重现以前的执行过程，这些命令分别对应于正常状态下的 T、P、G 命令，而且提示行还会显示当前正模拟跟踪到哪条指令。

如果要从模拟跟踪状态中退出，请输入以下命令：

TRACE OFF

5) 注意事项

(1) 你可以用 XRSET 命令来重新设置回溯跟踪缓冲区,以去掉缓冲区中原有的指令信息。

(2) 模拟跟踪时,有一些地方和正常跟踪是不一样的:回溯跟踪缓冲区里没有记录寄存器值的信息,所以,除了 CS 和 IP 外,其余寄存器的值都将保持不变;而且此时 X、P、T、G、EXIT 命令将不能正常工作。

(3) 你可能会注意到,模拟跟踪的指令有时并不是连续的,这是由于程序指令先在收集范围内执行,后来又跳出去执行该范围外的指令,而这些指令是不会被记录的,以后若再跳回收集范围内,Soft－ICE 将再次记录指令信息。这些情况通常出现在跳转、调用、返回等指令处。

(4) 当你在一个很大的范围内记录所执行过的指令时,工作起来很慢,这时你可以使用特殊的 Coarse 方式。Coarse 方式只收集在指定范围内做内存写入的指令,因而速度很快。使用时,BPR 命令的描述字为 TW,而不是 T,例如命令:

BPR . 1000:0000 8000:0000 TW

将收集从 1000:0000H 到 8000:0000H 这一较大范围内对内存做写入操作的指令。

你通常可以先用 Coarse 方式找到问题所在的大致范围,再用小模式详细了解该区域。

3.5 高端内存的使用及调式

如果你有超过 640K 字节的内存存贮器,那么,Soft－ICE 能充分利用这些硬件资源,并且可以调试使用这些资源的程序。

3.5.1 对 EMM 的支持

Soft－ICE 自己带有扩充内存管理程序,它已包含在主程序 S－ICE.EXE 中。该程序符合 EMM4.0 标准,它将再次填充你的常规内存,以尽可能扩大常规内存。

要注意的是,一旦你安装了 Soft－ICE 就不要再使用其它管理 EMM 内存的 386 控制程序。例如:QEMM 或 386MAX 都不能与 Soft－ICE 共存。

1) 配制 EMM 环境

如果,你不作任何配置工作,直接在 CONFIG.SYS 文件中 Soft－ICE 的安装行上加入 /EMM 开关,Soft－ICE 就按其默认方式进行配置,它将在低 640K (除第一个 64K 外) 和从 D000H 开始的 64K 这些区域为 EMM 提供页面。

由于某些原因,例如外部设备占用等,为了充分利用 EMM 特性,你可能要通过运行内存配置工具 EMMSETUP.EXE 程序来重新配置 EMM 环境。EMMSETUP.EXE 将记下你所作的修改,然后相应修改 S－ICE.EXE。当你再次装入 S－ICE.EXE 时,就会看到这些改变所产生的效果和作用。

EMMSETUP.EXE 的使用如下：

首先运行 EMMSETUP.EXE，它会显示出在低 1M 字节范围内的每 16K 内存页表的一个矩阵，该矩阵分为 16 行，每行代表 64K，每一行上的 4 块代表 64K 区域中的四个 16K 页表。

矩阵中的每一内存小块(16K)可以标为 E、X、R 或 V。E 表示该块可以作为 EMM 页表；X 表示该块不可以作为 EMM 页表；R 表示该块已被 EMMSETUP 识别为内存 ROM 区域，如果需要，且该 ROM 区确实不会被访问，你就可以用 E 来表示这些区域；V 表示该块为视频存贮区，如果你的视频卡(如 MDA、Hercules 卡)，并不使用那么多视频区，就可以把不必要的内存块标为 E，以增加可用内存。

下面给出一些常见设备在一般情况下所占用的内存区域，以供配置 EMM 时参考：

CGA 占据从 B8000H 到 C0000H 的内存。

MDA 占据从 B0000H 到 B1000H 的内存。

多数 Hercules 卡占据从 B0000H 到 C0000H 的内存。

EGA 占据从 A0000H 到 C0000H 及从 C0000H 到 C4000H 的内存。

VGA (母板)占据从 A0000H 到 C0000H 的内存。

VGA (选择卡)占据从 A0000H 到 C0000H 及从 C0000H 到 C8000H 的内存。

PS/2 系统 ROM 占据从 E0000H 到 100000H 的内存。

PS/2 ESDI ROM 占据从 CC000H 到 D0000H 的内存。

多数 AT 兼容机 ROM 占据从 F0000H 到 100000H 的内存。

Compap 机型、Micronix 和 Chips and Techologies 主板的机型把 EGA/VGV ROM 区移到 E0000H，但仍占据 C0000H 区。

Token Ring Networks 通常占据从 CC000H 到 E0000H 的内存。

多数 Networks 占据 D0000H 区的存贮区。

改变内存配置时，只需将光标移到某一块上，然后按下相应的字母。在修改完成后，按 F10 键存贮并退出，按 ESC 可取消变动直接退出。

2) 使用 EMM 内存

在 CONFIG.SYS 中把 /EMM 开关加到 S－ICE.EXE 行中，再重新启动系统，就可以使用 EMM 内存了。例如：

```
DEVICE＝DRIVE:\PATH\S－ICE.EXE /EMM 300
```

Soft－ICE 将把 300K 字节的扩展内存模拟成附合 EMM 4.0 标准的扩充内存，以供其它需要 EMM 内存的程序使用。

注意，一旦选择了用 Soft－ICE 作为扩充内存管理程序，就不允许再使用其它的 EMM 管理程序或 386 控制程序，例如，QEMM、EMM386、386MAX 等。

3.5.2 扩充内存的调试

当调试 EMM 程序时，你可以用 EMMMAP 命令显示 EMM 内存和当前正在映象的页，显示每一个有效的物理页。而 D、E、S、F 、C 命令可以用来察看或者修改任何已经分配的

EMM 控制页表。

在 D、E、S、F 命令中，命令的地址部分必须用下面的方式指定：

Hhandle Ppage offset

handle 是控制号，指由哪一个 EMM 控制；page 是 EMM 页面；offset 是指在一个 64K 的页面内的偏移(从 0000H 到 4000H)。例如：

DB H1 P1 100

这条命令将从控制 1，页面 1 的偏移 0100H 处开始以字节的格式显示(转储) EMM 内存页面的内容。

C 命令的地址部分必须用下面的方式指定：

Hhandle Ppage offset1 Llength offset2

handle 是控制号，指由哪一个 EMM 控制；page 是 EMM 页面；offset1 和 offset2 都是指在一个 64K 的页面内的偏移(从 0000H 到 4000H)，其中，offset1 和 offset2 分别是两个要比较的数据块的起始偏移量；length 是要比较的数据块的长度(字节数)。例如：

C H1 P2 0100 L100 0200

这条命令将比较在控制 1、页面 2 中的长度为 0100H 个字节的两个内存数据块。其中，第一个数据块在页面中的偏移量为 0100H，第二个数据块在页面中的偏移量为 0200H。

在使用 D,E,S,F,C 命令时，一旦第一次指定了控制和页面，那么以后均继续使用该控制和页面，除非重新指定控制和页面，或者指定了形如段址：偏移的常规内存地址。

3.5.3 扩展内存的调试

除 Soft－ICE 自己占用的扩展内存不能被显示和修改外，Soft－ICE 留给其它程序使用的扩展内存都可以用 D,E,S,F,C 命令察看和修改。

在 D、E、S、F 命令中，地址部分必须按照下面的方法指

M megabyte address

megabyte 是一个数，用来指定使用哪一个 1M 字节的区域；address 是指在所限定的 1M 字节的区域内的地址(这里的地址是全地址，即段址：偏移)。例如：

DB M 3 0000:0100

该命令将从第三个 8086 虚拟机范围(第三个 1M 字节的区域)内的地址 0000：0100H 处开始显示内存字节，即从线性地址 300100H 处。

C 命令则须按下面的方式指定：

C M megabyte address1 Llength address2

megabyte 是一个数，用来指定使用哪一个 1M 字节的区域；address1 和 address2 都是指在所限定的那个 1M 字节的区域内的地址(这里的地址是全地址，即段址：偏移)，其中，ad-

dress1 和 address2 分别是两个要比较数据块的起始全地址(段址:偏移)；length 是要比较的数据块的长度(字节数)。例如：

```
C  M 2 0000:0000 L20 B000:0000
```

这条命令将比较在第二个 1M 字节的区域中长度为 20H 个字节的两个内存数据块。其中，第一个数据块的起始地址为 0000:0000H，第二个数据块的起始地址为 B000:0000H。

在使用 D、E、S、F、C 命令时，当你第一次指定了某一个 1M 字节的区域，那么以后的 D、E、S、F、C 命令均继续使用该 1M 字节的区域，如想返回常规内存(第 0 个 1M 字节区域)，可按下列方法指定地址：

```
M 0
```

3.6 Soft－ICE 和其它调试程序同时使用

Soft－ICE 最大的特点之一就是可以和其它动态调试程序同时使用，取长补短。由于 Soft－ICE 有强大的断点生成功能，那么，你就可以利用 Soft－ICE 来设置断点，而当断点发生时，再用其它的调试程序来处理。

当你把 Soft－ICE 与其它调试程序结合起来使用时，有几个地方需要特别加以注意。

3.6.1 重进入问题

Soft－ICE 本身在进行屏幕显示和键盘输入时，并不使用 DOS 功能调用或者 ROM BIOS 调用，而是用它自己的服务程序。所以单独用 Soft－ICE 调试时，你不必担心重进入问题，也就是说，可在任何时候中断 Soft－ICE 而不会出现 DOS 或者 ROM BIOS 的重进入。

但是，许多其它调试程序却用 DOS 调用和 ROM BIOS 调用来完成屏幕显示和键盘 I/O 功能。如果当前代码正在 DOS 功能调用或 ROM BIOS 调用中执行时，出现了一个断点，那么，就会产生所谓的重进入问题，而重进入的结果是不可预料的，往往会造成整个系统的崩溃。

所以，当 Soft－ICE 和其它调试程序共同工作时，应该避免产生重进入。Soft－ICE 的 WARN 命令提供了警告重进入的功能。当 WARN 设置为 ON 状态时，如果遇到一个断点发生了，那么在把控制交给宿主调试程序之前，Soft－ICE 将检查 DOS 和 ROM BIOS 是否有重进入问题。如果检查到有重进入问题，Soft－ICE 将显示警告信息，并让你选择是继续执行下去还是返回 Soft－ICE。

3.6.2 Soft－ICE 与其它调试程序的接口

如果你用 Soft－ICE 来设置断点，当断点发生时，你想用其它的调试程序来处理，也就是说当断点发生时，激活其它的调试程序，并把控制权交给它。那么，就必须用 ACTION 命令来连接 Soft－ICE 和其它调试程序的这种关系。ACTION 命令决定了当断点发生时，执行一条什么指令，这条指令将把控制权交给相应的调试程序。

对于多数调试程序，断点发生时，执行一条 INT 3 指令，所以当 Soft－ICE 和它们一起工作时，ACTION 应设置成 INT3。例如，当 ACTION 置为 INT3 时，DOS 的 DEBUG 和 SYMDEB(符号 DEBUG)工作得最好。

对于 CODEVIEW 而言，如果仍用 INT3，往往不能正确地中断。所以，当 Soft－ICE 与 CODEVIEW 一起工作时，ACION 最好设置成 NMI (非屏蔽中断)。

对于此外的其它调试程序如果将 ACTION 设置为 INT3 时，不能正常工作，那么请试试用 INT1 或者 NMI 。

如果你想用自己的断点中断服务程序，就应该把 ACTION 设置成该中断调用的中断号。例如，当断点发生时，你想执行自己的断点中断服务程序，且该程序已驻留，中断号为 60H，那么应使用 ACTION 60 命令。这样，每当遇到一个断点，就执行一条 INT 60 指令，进入你自己的断点中断服务程序进行处理。

3.6.3 避免调试程序的重进入

当你设置了一个断点后，注意不要无意识地触发这个断点。

例如，你已在内存中某点设置了一个存储器写断点。当断点发生时，Soft－ICE 会把控制权交给你的调试程序，这时，你就进入自己的调试程序了。如果此时你又对该点进行写操作，那么，因为断点仍有效，将会再次触发，从而产生调试程序的重进入。对于那些不能重进入的调试程序来说，这将是一个致命的错误。

所以，为了避免调试程序的重进入，一旦 Soft－ICE 把控制权交给了你的调试程序后，最好先取消 Soft－ICE 所设置的断点，以免发生问题。

3.6.4 80386 断点寄存器的冲突

80386 有 4 个断点寄存器可供调试程序使用。Soft－ICE 使用这些寄存器存贮它的寄存器字节，字和双字断点。如果正与 Soft－ICE 一起使用的调试程序使用了这些断点寄存器，那么就会发生冲突。

有两个方案可解决这一问题：

1) 查阅正和 Soft－ICE 一起使用的调试程序的操作手册，按照手册上的说明，禁止它使用 80386 断点寄存器。

2) 如果第一种方案行不通，你可以在启动另一个调试程序之前，在 Soft－ICE 窗口下用 BREAK ON 命令，接通 Soft－ICE 的终止方式。然后关调 Soft－ICE 窗口，再启动另一个调试程序。现在，你就可以弹出 Soft－ICE 窗口了。

3.7 受用户限制的断点

虽然 Soft－ICE 提供了强大的断点设置功能，可以限定一系列的断点条件，但当条件较复杂时，直接设置的断点就显得无能为力了。例如，你想设置这样一个断点，只有在当前寄存

器 AX=3,BX=4,CX=5 时断点才发生。显然，用普通的断点设置命令是无法满足要求的。

Soft－ICE 提供了受用户限制的断点的功能：你可以通过写一个断点条件例行程序来限定寄存器的值或存贮器的值，以满足要求。

Soft－ICE 通过设置 ACTION 来调用用户所提供的断点条件例行程序。用户首先应把断点条件例行程序装入内存，并将某一中断向量指向该程序的入口地址。断点条件例行程序首先判断条件是否符合，如果条件例行程序发现与条件匹配的断点，那么，它将按要求进行一系列处理后，弹出 Soft－ICE 窗口。

1）注意事项

用户在使用这一功能时应注意以下几点：

（1）在自由空间的任意位置建立一个断点条件例行程序。该程序必须保存寄存器，然后比较所需的条件。当条件匹配时，执行一条 INT 3 指令弹出 Soft－ICE；条件不匹配时，执行一条 IRET 指令继续往下执行。

（2）把一个不使用的中断向量指向你的断点条件例行程序的入口地址。可以在你的代码中或在 Soft－ICE 中做这件事。

（3）在 Soft－ICE 中，把 ACTION 置为指向你的断点条件例行程序的中断值。

（4）在 Soft－ICE 中，把 I3HERE 置于 ON 状态。它用于在条件满足时激活 Soft－ICE。

（5）在 Soft－ICE 中设置普通断点。则当这些断点发生时，系统将调用你的断点条件例行程序。

2）应用举例

现举两个例子如下：

（1）一个简单例子

断点限制条件为 SI=0,DI=1,BP=2。

首先建立断点条件例行程序。在此例中，我们用较简单的方法，即直接用 Soft－ ICE 的汇编命令 A ，把指令汇编到内存中。从任意空闲地址开始，例如 8000:0000H 处，汇编该断点条件例行程序。用 A 命令把下面的语句键入：

```
A 8000:0000
8000:0000     CMP  SI, 0
8000:0003     JNE  10
8000:0005     CMP  DI, 1
8000:0008     JNE  10
8000:000A     CMP  BP, 2
8000:000D     JNE  10
8000:000F     INT  3
8000:0010     IRET
```

接着，把一个不用的中断向量指向该断点条件例行程序。我们在此选用 INT 60H，并将 60H 号中断入口地址指向 8000:0000H：

```
ED  0000:60*4 8000:0000
```

设置 ACTION 状态，以便 Soft－ICE 能在每个断点处调用你的断点条件例行程序：

ACTION 60

设置 I3HERE 为 ON 状态，以便当条件符合时，断点条件例行程序可以激活 Soft－ICE：

I3HERE ON

最后，你可以用 Soft－ICE 的断点设置命令来设置常规断点。例如：

BPR 1000:0000 1000:1000 W

这样，每当有指令在内存范围 1000:0000H 到 1000:1000H 内进行写操作时，就调用你的断点条件例行程序。如果条件满足(当前寄存器 SI＝0，DI＝1，BP＝2)，就执行 INT 3 指令并弹出 Soft－ICE；条件不满足时则执行 IRET 指令返回程序继续运行。

当 Soft－ICE 弹出时，当前 CS:IP 是指在你的断点条件例行程序的 INT 3 指令处(本例是 8000:000FH)。这时你应把指令指针指向 IRET 指令(8000:0010H 处)，并单步跟踪一次，以便返回到产生断点的指令的下一条指令处：

R IP IP＋1

T

最后，请记住把 ACTION 状态改回 ACTION HERE，以便下面不需要由自己的断点服务程序进行特殊处理的断点不会通过你的断点条件例行程序。

2) Soft－ICE 的例子

Soft－ICE 自带了一个受用户限制的断点的例子。该例用汇编语言写了一个断点条件例行程序(IOSIM.ASM)，该程序模拟 I/O 端口的输入和输出操作，并常驻内存，用 INT 65H 激活。程序清单如下：

```
PAGE 55,13
TITLE I/O SIMULATION TEST PROGRAM
.286

;
;       This is a sample user qualified breakpoint.
;

INT_NUM EQU 65H          ;THIS IS THE PSUDEO INTERRUPT NUMBER

REG_WORD STRUC
REG_DI          DW      ?
REG_SI          DW      ?
REG_BP          DW      ?
REG_SP          DW      ?
REG_BX          DW      ?
REG_DX          DW      ?
REG_CX          DW      ?
REG_AX          DW      ?
REG_IP          DW      ?
REG_CS          DW      ?
REG_FLAGS   DW      ?
REG_WORD ENDS
```

```
REG_BYTE STRUC
REG_DI1     DW    ?
REG_SI1     DW    ?
REG_BP1     DW    ?
REG_SP1     DW    ?
REG_BL            DB    ?
REG_BH            DB    ?
REG_DL            DB    ?
REG_DH            DB    ?
REG_CL            DB    ?
REG_CH            DB    ?
REG_AL            DB    ?
REG_AH            DB    ?
REG_BYTE ENDS

STACK SEGMENT STACK
      DW      50 DUP(0)
STACK ENDS

DATA SEGMENT BYTE PUBLIC 'DATA'

NEW_VAL DW  ?
NEXT_IN_LOC DW    0
NEXT_OUT_LOC DW  0
PUBLIC BUFFIN
BUFFIN       DW   1000H DUP(0)
PUBLIC BUFFOUT
BUFFOUT  DW 1000H DUP(0)
DATA ENDS

CODE SEGMENT PUBLIC 'CODE'
CODE ENDS

LAST SEGMENT PUBLIC 'LAST'
LAST ENDS

CODE SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:CODE,DS:DATA
PUBLIC START
START:      MOV   BX,LAST           ;THE END OF THE PROGRAM
     MOV   AX,ES             ;SUBTRACT FROM THE TOP OF THE HEADER
     SUB   BX,AX             ;BX NOW HAS THE SIZE TO KEEP WHEN WE
                             ; TERMINATE AND STAY RESIDENT

     MOV   AX,DATA            ;GET OUR DATA SEGMENT
       MOV     DS,AX
     XOR   AX,AX             ;POINT ES AT SEGMENT ZERO
     MOV   ES,AX             ;

     MOV   AX,OFFSET IO_ROUTINE     ;GET A POINTER TO THE I/O ROUTINE
     MOV   ES:[INT_NUM * 4],AX ;POINT THE INTERRUPT TO THE IO ROUTINE
     MOV   AX,CS             ;GET THE CURRENT SEGMENT
     MOV   ES:[INT_NUM * 4+2],AX      ;MOVE IT IN
```

```
        MOV     AH,31H                  ;USE THE MSDOS KEEP PROCESS FUNCTION
        MOV     DX,BX                   ;GET THE SIZE OF OUR ROUTINES
        INT     21H                     ;THATS IT

;-------------------------------------------------------------------------------
;  This routine will be called when ever Soft-ICE reaches a break point
; and the ACTION is set to our INT_NUM.  The stack will be set up as if
; a normal REAL mode interrupt had happend. ie;
;
;               FLAGS
;               CS
;               IP
;  The opcodes that we will be looking for are:
;     0ECH - IN   AL,DX
;     0EDH - IN   AX,DX
;     0EEH - OUT  AL,DX
;     0EFH - OUT  AX,DX
;   Note that bit 02h tells us whether it is an IN or an OUT and bit 01h tells
;  us whether it is a BYTE or a WORD.
;
;-------------------------------------------------------------------------------
IO_ROUTINE PROC FAR
        PUSHA                           ;SAVE ALL THE CURRENT REGISTERS
        MOV     BP,SP                   ;USE BP TO POINT BACK ONTO THE STACK
        PUSH    DS                      ;SAVE THE SEGMENTS ALSO
        PUSH    ES                      ;
        MOV     AX,DATA                   ;GET OUR DATA SEGMENT
        MOV     DS,AX                   ;SET IT UP
        MOV     AX,[BP].REG_CS            ;GET THE OLD CS REGISTER
        MOV     ES,AX                   ;SET ES TO THAT SEGMENT
        MOV     SI,[BP].REG_IP            ;USE SI TO POINT TO THE IP
        MOV     AL,ES:[SI-1]              ;GET THE OPCODE OF THE I/O INSTRUCTION
        CMP     AL,0ECH                 ;WAS IT IN IT I/O OPCODE RANGE?
        JB      NOT_AN_IO               ;NOPE
        CMP     AL,0EFH                 ;CHECK THE HIGH END
        JA      NOT_AN_IO               ;NOPE NOT THIS ONE EITHER
        MOV     DX,[BP].REG_DX            ;GET THE DX REGISTER FROM THE STACK
        MOV     CL,0                    ;ASSUME IT WAS A BYTE ACCESS
        TEST    AL,01H                  ;WAS IT A BYTE OR A WORD?
        JZ      BYTE_ACCESS             ;YES, JUST A BYTE
        MOV     CL,1                    ;MARK IT AS WORD ACCESS
BYTE_ACCESS:
        TEST    AL,02H                  ;WAS IT AN IN OR AN OUT
        JNZ     OUT_OPCODE              ;IT WAS AN OUT
IN_OPCODE:
;-------------------------------------------------------------------------------
;  ENTRY:   CL = BYTE OR WORD ACCESS
;           0 = BYTE
;           1 = WORD
;
;         DX = THE I/O PORT
;
;  EXIT:
;         [BP].REG_AX OR [BP].REG_AL    SHOULD HAVE THE SIMULATED VALUE
```

```
;
;.....

     MOV   BX,NEXT_IN_LOC
     MOV   AX,BUFFIN[BX+2]
     MOV   [BP].REG_AL,AL          ;MOVE IN JUST A BYTE
     ADD   NEXT_IN_LOC,4
     MOV   AX,NEW_VAL         ;GET THE LAST OUTPUT VALUE
     OR    CL,CL              ;IS IT A BYTE OR A WORD
     JZ    DO_BYTE            ;JUST A BYTE
     MOV   [BP].REG_AX,AX          ;MOVE IT IN
     JMP   IO_DONE
DO_BYTE:
     MOV   [BP].REG_AL,AL          ;MOVE IN JUST A BYTE
     JMP   IO_DONE
OUT_OPCODE:
;.....
;  ENTRY:   CL = BYTE OR WORD ACCESS
;           0 = BYTE
;           1 = WORD
;
;         DX = THE I/O PORT
;
;         [BP].REG_AX OR [BP].REG_AL   THE VALUE GOING OUT
;
;.....
     MOV   AX,[BP].REG_AX          ;GET THE VALUE
     MOV   NEW_VAL,AX         ;SAVE IT

     MOV   BX,NEXT_OUT_LOC
     MOV   BUFFOUT[BX],DX
     MOV   BUFFOUT[BX+2],AX
     ADD   NEXT_OUT_LOC,4

IO_DONE:
NOT_AN_IO:
     POP   ES                 ;RESTORE ALL OF THE SEGMENTS AND REGS
     POP   DS                 ;
     POPA                     ;
     IRET                     ;RETURN TO THE INSTRUCTION AFTER THE
                              ; I/O
IO_ROUTINE ENDP

CODE ENDS

END START
```

执行 IOSIM.EXE 程序并驻留后，设置 ACTION 状态：

```
ACTION   65
```

由于此例中并不返回 Soft－ICE，所以可以不设置 I3HERE 状态。

把 Soft－ICE 的断点动作设置成 INT 65H 后，你就可以在应用程序中用 Soft－ICE 设置

普通断点了。

当断点发生时，执行 INT 65H，即断点条件例行程序。

首先判断引起断点的指令是不是 I/O 端口操作指令。如果不符合条件，则返回应用程序继续执行；若是端口操作指令，则根据不同的情况进行相应的模拟动作。如果是写某一端口，则将端口号和所要写的值依次记录在缓冲区 BUFFOUT 中；如果是从某一端口读，则将 NEW_VAL（上次向某一端口所写的值）作为所读到的值返回给 AX 或者 AL 寄存器。

3.8 调试图形方式下的程序

由于 Soft－ICE 窗口用文本方式显示，所以当 Soft－ICE 被激活时，屏幕转换到文本方式，因而应用程序所显示的图形不可见。

为了能看到应用程序所显示的图形，Soft－ICE 提供了三个特性：

1）用双显示器时，设置 ALTSCR 状态。

2）用 FLASH 命令，使得你在执行 P 或 T 命令的过程中可以恢复程序屏幕。

3）用 RS 命令暂时恢复程序屏幕。

如果以上措施均不凑效，可把 WATCHV 置于 ON 状态再试一试。

3.9 特殊程序的调试

在独立方式下的 Soft－ICE 是一个强有力的工具。它能调试可装入的设备驱动程序，引导程序，中断服务程序，甚至非 DOS 的操作系统。注意，当 Soft－ ICE 独立使用时，ACTION 必须设置成 HERE 状态。

3.9.1 可装入的设备驱动程序

有两种方法调试可装入的设备驱动程序：

第一种方法，首先用 MAP 命令找到你的可装入的设备驱动程序的地址，显示其标题部分，找到策略或中断入口点。在策略或中断入口处设置断点。如需要可继续设置断点，或通过单步执行来调试。如果该设备驱动程序需要初始化，则用 BOOT 命令使系统复位（但保留 Soft－ICE 和由它所设的断点）。在策略或中断的入口处，Soft－ICE 将获得设备驱动程序内部的控制权。接下去，你就可以用 Soft－ICE 来跟踪调试该设备驱动程序了。

第二种方法是将特殊代码暂时放入设备驱动程序中。在需要中断的地方放入 INT 3 指令（其机器代码为 CCH），并把 I3HERE 置为 ON 状态。当执行到 INT 3 时，控制权交给 Soft－ICE。这时，CS：IP 是指向 INT 3 处的，用 R IP 命令调整 IP 指针使其指向下一条指令，你就可以开始跟踪了。

如果你想使用符号或源程序的形式来调试设备驱动程序，就必须在装入设备驱动程序后，再用 LDR.EXE 来装入符号文件和源程序文件：

LDR filename.SYM

接着用 SYMLOC 命令，重新设置正确的符号基数，以使符号和实际地址一一对应。

然后，就可以设置断点，开始调试了。

注：调试可装入的设备驱动程序时，必须在设备驱动程序装入之前装入 Soft－ICE。

3.9.2 引导程序

调试引导程序也有两种方法：

第一种方法是在引导程序的某一已知地址处设置一断点，然后用 BOOT 命令来使系统软复位。BOOT 命令将保留 Soft－ICE 和断点，并能在断点处弹出 Soft－ICE。如果无法确定地址，可在 0000:7C00H 处设置一执行断点，然后使用 BOOT 命令。因为启动时，ROM BIOS 把引导程序读入这一内存区域，并把控制权交给这一地址处。

第二种方法是把 I3HERE 置为 ON 状态，并把 INT 3 指令暂时放在你的程序中需要中断的地方。当用 BOOT 命令后，在 INT 3 处，Soft－ICE 将获得控制权。

若要利用符号和源程序来进行调试，也应用命令：

LDR filename.SYM

单独把符号文件和源文件装入，再用 SYMLOC 命令正确地重新定位符号地址，然后再设置断点开始调试。

3.9.3 中断程序

Soft－ICE 允许在硬件中断服务程序中(定时器、键盘等)设置断点和单步执行。所以你几乎可以在任何中断服务程序中设置断点和单步跟踪。

当调试中断服务程序时，最好让 Soft－ICE 独立使用。你可以用两种方法来调试中断服务程序。

第一种方法是先找出中断服务程序的入口地址，用命令：

DD interrupt－number * 4 L4

显示中断服务程序的第一条指令的地址，然后就在这个地址处设置一个执行断点。

第二种方法是直接在中断处设置断点：

BPINT interrupt－number

3.9.4 非 DOS 操作系统

Soft－ICE 甚至可以调试非 DOS 的操作系统。

首先在 DOS 下装入 Soft－ICE，然后用 BOOT 命令启动非 DOS 操作系统，就象调试引导程序一样来调试非 DOS 的操作系统。同样，只要你装入符号文件和源文件，并正确地重定位符号地址，就可以用符号和源程序方式来调试。

注：在非 DOS 操作系统下，MAP 和 WARN 命令也许不能正常工作，但其它调试命令都可以正常工作。

3.10 远程终端调试

Soft－ICE 支持远程终端调试特性。

Soft－ICE 能够通过串行口输出 Soft－ICE 命令窗口的信息显示到打印机或另一个显示屏幕。Soft－ICE 还能接收异地键盘的输入。一旦你已经用主键盘(本地键盘)的热键激活了 Soft－ICE，那么以后，本地主键盘和异地远程键盘都可被接受。

为了使用 Soft－ICE 的远程调试功能，应按以下步骤做：

1) 在 DOS 下，用 DOS 的 MODE 命令设置正确的波特率，以保证 Soft－ICE 能正常地和远程终端通信。

2) 在 Soft－ICE 中，用 PRN 命令，选择正确的串行通信口。例如，我们选串行 2 口作远程终端的通信口：

```
PRN  COM2
```

3) 在 Soft－ICE 中，用 SERIAL 命令打开控制台重定向开关：

```
SERIAL  ON
```

4) 这样，当你不想在主机屏幕上显示 Soft－ICE 信息时，就可以用命令：

```
ALTSCR  ON
```

那么，Soft－ICE 的信息将切换到远程终端的屏幕上显示。

注：Soft－ICE 除命令窗口外，代码窗口，数据窗口和寄存器窗口均不能通过串行口送到远程终端显示。

3.11 从程序中调用 Soft－ICE

如果当你写程序时，在适当的地方暂时预先加入调用 Soft－ICE 的指令，可以大大地方便你的调试工作。等调试成功后，再去掉这些辅助调试指令。

Soft－ICE 为此提供了一种所谓“后门”(Back－door)的功能。它允许你的程序直接控制 Soft－ICE。为使用这一特殊的功能，必须预先在程序中需要的地方加入下列指令以及相关的数据信息：

```
MOV   AH, 09
MOV   AL, SUB_FUNCTION
MOV   SI, 'FG'
MOV   DI, 'JM'
INT   3
```

SUB_FUNCTION 为子功能号，具体含义是：

AL＝10H　在 Soft－ICE 窗口中显示信息。这主要用在诊断写时，特别是当程序正在进行中断处理或其它一些不可重进入的地方。

入口参数　DS:DX　指向所要显示的字串。该字符串可包括控制字符回车(0DH)，但总长不超过 100 个字节。

AL＝11H　执行一系列的 Soft－ICE 命令。这一项子功能允许你在应用程序中使用 Soft－ICE 命令，这对于在你自己的程序中控制 Soft－ICE 设置和清除断点很有用。

入口参数：　DS:DX　指向由 Soft－ICE 命令和回车控制符(0DH) 组成的不超过 100 个字节长的命令字符串。

AL＝12H　获得断点信息。该子功能返回最近一次设置的断点和最近一次发生的断点的有关信息。主要用于为了进行硬件控制或者模拟硬件动作而设置断点时。

返回参数:DH　最近一次发生的断点的断点号

DL　最近一次发生的断点的断点类型

BH　最近一次设置的断点的断点号

BL　最近一次设置的断点的断点类型

注:断点类型是从 0 到 5 中的一个数：

0　表示 BPM 类型的断点(由 80386 断点寄存器产生的)

1　表示 I/O 端口类型的断点

2　表示由 INT 指令引起的断点

3　表示用 BPX 命令设置的断点(INT 3 类型的断点)

4　保留

5　表示范围类型的断点

4 常用技巧及经验

4.1 实用技巧

由于 Soft－ICE 不属于菜单操作类型的软件，并且没有及时的在线帮助，所以，许多操作步骤不那么直观，只有靠在使用中，不断地摸索总结经验，才能更快更巧妙地使用 Soft－ICE 这一命令输入方式的动态调试工具。

在此，我们介绍一些常用的技巧。如果你能在使用中巧妙地运用这些特性并加以发挥，往往会取得事半功倍的效果。

1）有时，你无意中跟踪跟进了 DOS 功能调用和 ROM BIOS 中断，想要跳过这些你不感兴趣的功能处理过程；或者，你本想跟踪某一子程序，当你刚进入该子程序的第一条指令时，用 U 命令发现这部分代码并不重要，想要返回调用处的下一条指令。这时，可使用 Soft－ICE 的间接地址转向特性。输入命令：

```
G    @SS:SP
```

我们知道，在执行调用指令 CALL 时，CALL 指令的下一条指令的段地址和偏移将被压入堆栈。利用符号 @ 的间接寻址特性，上述命令使程序一直运行到调用处的下一条指令才停下来。

2）在命令窗口的底部，有一信息提示行。当你输入命令时，提示行会显示相应的提示信息。例如你想输入 ALTKEY 命令，敲入第一个字符 A 时，提示行将显示以 A 打头的所有命令（ACTION ALTKEY A ALTSCR），接着敲入第二个字符 L，则显示以 AL 打头的命令（ALTKEY ALTSCR），当你输完 ALTK 四个字符时，提示行的显示是唯一一个以 ALTK 打头的命令 ALTKEY，这时你就可以敲空格键，因命令已确定，Soft－ICE 会自动帮你把剩下的字符输完，并加上一个空格，然后提示行显示 ALTKEY 命令的用法。

3）用 ? 或者 H 命令，可以浏览 Soft－ICE 的所有命令及其解释。而在 ? 或 H 命令后带上某一命令作参数，则会看到该命令的详细使用方法和应用举例。如：

```
? A
```

将得到如何使用 A 命令的详细帮助及例子。

4）如果你想知道某条命令可以带什么参数以及这些参数表示什么意思和怎样使用这些参数，你可以在输完该命令的所有字符后敲入一个空格，从信息提示行中获得有关方面的快速帮助。例如，当你输完 TABLE 命令的五个字符后敲空格键后，Soft－ICE 将在信息提示行中显示：

```
TABLE [1|2]
```

它告诉你，TABLE 命令可带参数也可不带参数，如果带参数则参数必须是数字 1 和 2 当中的一个。

5) MAP 命令能确定当前 CS:IP 所在的位置。当你用热键激活 Soft－ICE 后，可能会不明白当前程序指针所处的地方，这时可用 MAP 命令察看当前 CS:IP 究竟在什么地方：是在应用程序当中还是在 DOS 或 ROM BIOS 调用中。

6) 如果你的光标是在代码窗口中，也可以直接向命令窗口输入。只要简单地敲入命令字符，该字符会显示在命令窗口中，就象在命令窗口中输入一样。

7) 如果你想退出正在调试的程序，最好是用 EXIT 命令加 R 和 D 参数：

```
EXIT R D
```

Soft－ICE 首先清除所有断点并恢复中断向量表。这样，被程序修改过的中断向量表可以得到恢复；而且，假如不清除断点就直接退出，以后别的程序代码占据该位置时会被不正常地中断。

8) 有时，其它的驻留程序也用 CTRL＋D 来作为热键。你可以试着用 CTRL＋SHIFT＋D 来激或 Soft－ICE，如果仍然有冲突，则可以用 ALTKEY 命令来改变 Soft－ICE 的热键序列以避开冲突。

9) 当你用 EGA/VGA 系统调试没有图形的程序或者 CGA 和单色系统时，能够通过再次填充来增加系统内存的数量，使之大于 640K。运行 EMMSETUP.EXE 程序，把不用的屏幕显示内存块标为 E ，以增大可用内存。

10) 如果你想锁住某一内存存贮器，使之保持常数，可首先编写一个中断服务程序，该程序将一常数写入指定的内存存贮器，然后修改中断向量表，使一个不用的中断号指向该中断服务程序，再设置 ACTION 状态为这一中断号即可。

11) Soft－ICE 不允许时钟中断，所以当使用 Soft－ICE 后，如果系统时钟不正常，可用 UPTIM.EXE 程序来修改系统时钟。

12) 用 FILE 命令切换当前源程序时，文件名可不带扩展名。例如，当你想把一个已经调入了内存中的源文件 MYFILE.ASM 切换成当前源程序时，仅用：

```
FILE MYFILE
```

13) 不能连续按着状态键。如果你用热键 CTRL＋D 弹出 Soft－ ICE 窗口后，不释放 CTRL 键，马上又用热键退出 Soft－ICE，这时，你要是不先放开 CTRL 键然后再按下，那么你就无法立即又用 CTRL＋D 来激活 Soft－ICE。

14) 在代码窗口中，当前指令所在行的字符底色是明亮的，而且整行看上去就象被一个占据一整行的大光标条所标识出；设置了断点的行，其字符是高亮的，但整行的底色不变，只是字符前景色用高亮度表示。

4.2 应注意的问题

使用 Soft－ICE 时，必须注意以下问题：

1) 在 ROM 中跟踪时，由于 80386 的断点寄存器被占用，P 和 G 命令有时不能正常运

行。应该尽量避免在 ROM 中使用 P 或 G 命令。

2) 在 80386SX 或有的兼容机上，容易出现键盘不同步和死机的现象。在有的机型上，当你从 Soft－ICE 中退出后，敲入一个键正常，但再敲任何键都不会有反应了，发生死机现象，如果出现这种情况，在你退出 Soft－ICE 时，不要按任何别的键，而必须先按下 NumLock 或其它的 LED 灯控制键，待喇叭响一声后才能进行正常输入；有的机上正好相反，从 Soft－ICE 退出后，不能马上按 LED 灯控制键，否则会死机；有时从 Soft－ICE 退出后，状态控制键 SHIFT、CTRL、ALT 会被逻辑地按下，如果其状态不正常，可再次按下状态键，使之切换回正常状态。

3) Soft－ICE 所能设置的断点个数是有一定限制的，不同的断点设置命令可设置的断点数如下：

BPX 命令最多可设 10 个断点
BPM 命令最多可设 4 个断点
BPR 命令最多可设 10 个断点
BPIO 命令最多可设 10 个断点
BPINT 命令最多可设 10 个断点

而各种断点的的总数不能超过 32 个。

4) Soft－ICE 不能调试有保护模式指令的程序。

5) Soft－ICE 碰到无效指令时将弹出，并让你选择是继续执行还是返回 Soft－ICE。如果选择继续执行则把控制权交给 INT 6（无效指令中断）。

6) 不能设置一个断点转移到 INT 15H 的 87H、88H、89H 号子功能。因为 Soft－ICE 自己用保护模式调用这些子功能。

7) 当屏幕显示不正常时，可通过设置 FLICK 和 WATCHV 的状态来调整。

8) 用 BPM 命令设置的范围断点的范围和用 BPR 命令带 T 或 TW 参数设置的回溯跟踪范围不能重叠。如果重叠的话，断点将不能被正确地触发。

4.3 工具程序的使用

4.3.1 UPTIME.EXE

由于 Soft－ICE 在激活时不允许任何中断通过系统，所以有可能影响系统时钟。如果在退出 Soft－ICE 窗口后发现时钟不正常，你可以用 Soft－ICE 提供的这个程序来修改时钟。运行 UPTIME.EXE 时不用带参数，因为 Soft－ICE 并不影响实时时钟，所以 UPTIME.EXE 读入实时时钟再通过调用 DOS 服务功能来设置正确的时间。

4.3.2 MSYM.EXE

MSYM.EXE 用来生成 Soft－ICE 在进行符号和源程序级调试时所必须的符号文件。它

读取由链接程序 LINK.EXE 产生的符号及源程序信息的映象文件(MAP 文件)中有关的信息,进而把这些符号和源程序调试所必须的信息转换成 Soft－ICE 能够识别的格式,并储存在符号文件(缺省扩展名为 .SYM)中。

目前 MSYM.EXE 的版本号是 2.11,使用时应带上一个参数,即需要转换的映象文件的文件名:

```
MSYM   filename
```

如果没有指定扩展名,则缺省扩展名是 .MAP。而且,可带路径及盘符,但所有的文件描述字符不能超过 16 个字符,若超过则只取前 16 个。

执行后将生成一个扩展名为 .SYM 的符号文件,以备在进行符号或源程序级上调试时装入内存供 Soft－ICE 使用。

注意,如果在你的映象文件中,某个符号名的第一个字符所在位置,相对于映象文件开始处的偏移超出了 32K 这一界限,那么,MSYM.EXE 会退出。

4.3.3 LDR.EXE

LDR.EXE 程序用来装入符号文件和源文件以及可执行文件。LDR.EXE Version 2.51 的使用和参数选择如下:

```
LDR   filename[[.EXE|.COM|.SYM] filename[.EXE|.COM]]
```

第一个文件描述符可以是带盘符和目录的全路径,与 DOS 路径兼容。如果带两个参数,那么第一个文件必须是符号文件(.SYM),第二个文件是可执行文件(.EXE 或 .COM)。如果只带一个参数且没有指出扩展名时,LDR.EXE 将首先把符号文件装入内存,若符号文件中指定了源程序调试信息则再装入源文件,然后装入可执行文件。倘若需要,例如调试引导程序时,也可以单独装入或只装可执行文件,这时,你只能带一个参数,并且必须指出文件扩展名(.EXE 或 .COM)。

一旦成功地装入了所需的文件,Soft－ICE 窗口将弹出,并完成程序的初始化。此后,如果用 X 命令或热键序列退出 Soft－ICE,将使程序一直运行下去,除非遇到 Soft－ ICE 所设置的断点。

用 EXIT 命令可以终止由 LDR.EXE 加载的程序的运行。如果在此之前装入了符号文件和源文件,即使用 EXIT 退出程序,符号和源程序仍然在内存中,下次再加载时可只装入可执行文件,不必重装符号文件和源文件。

另外,用 LDR.EXE 装入的 .COM 文件,会自动被 Soft－ICE 正确地重定位符号地址,不需要再用 SYMLOC 命令来人工定位。

4.3.4 EMMSETUP.EXE

EMMSETUP.EXE 是用来调配 EMM 内存的,它使你能够根据需要合理地配置 EMM 环境,并充分利用内存空间。

EMMSETUP.EXE 是 Nu－Mega 技术公司开发的一个扩充内存配置工具程序。它能够

用于任何 Nu—Mega 出品的软件。EMMSETUP.EXE Version 1.0 的使用方法如下：

EMMSETUP NUMEGA—Software

例如，我们现在想要配置 Soft—ICE：

EMMSETUP S—ICE.EXE

它将对文件 S—ICE.EXE 作相应的修改，以使 Soft—ICE 能按新的方式使用内存。进入 EMMSETUP.EXE 后，弹出主菜单：

Currently working on s—ice.exe

Exit and Save Changes
Manual Configuration and Status Screen
Reconfigure Driver to Current Configuration
Exit without Saving Changes

Exit and Save Changes will save the current configuration.

Manual Configuration and Status Screen will allow you to see how the expanded memory and high memory is mapped. If you have a network installed in your system you MUST use this screen to exclude the area where the network adapter is mapped.

Reconfigure Driver to Current Configuration is useful if you have just moved the driver to a new computer, or you have changed your hardware configuration.

第二项选择允许你察看和修改扩充内存和高端内存的使用情况。如果你的系统安装了网络卡，则必须查明网络适配器所占用的区域，并使用该选项作适当修改，以避免 Soft—ICE 使用该地址。

第三项让你选择是使用扩充内存和高端内存，还是只用扩充内存。

显示和修改时的菜单如下：

This chart shows each 64K segment in your system and the status of each 16K block within that segment.
Key to status's:

0000 XXXX
1000 EEEE
2000 EEEE
3000 EEEE
4000 EEEE
5000 EEEE
6000 EEEE

E — Expanded memory page
F — Expanded memory page frame
X — Excluded page
V — Video memory
R — ROM
N — Network memory

```
7000 EEEE          H — High memory area
8000 EEEE
9000 EEEE     The expanded memory pages below 640K are used by
              programs such as Microsoft Windows and Desqview.
A000 VVVV
B000 VVVV      If you are running on a network you MUST
C000 RRFF     consult the documentation on your network adapter
D000 FFFF     card to map out the areas used by that card.
E000 FFFF
F000 FFRR     To override the current status, use the arrow keys
              to move to the 16k block you wish to modify, and
              then enter one of the above status letters.
                 Press Enter when you are finished with this screen.
```

每一行代表一个 64K 的内存段，每一小块代表一个 16K 的内存块。每个内存块均被相应的状态字符所标出。E 表示扩充内存页面；F 表示扩充内存映射页表；X 表示禁止 Soft－ICE 使用的页面；V 表示显示内存区；R 表示 ROM 区；N 表示留给网络使用的内存区；H 表示高端内存区。

如果你的网络或其它硬件设备占用了某一内存区，那么，必须用 N 或 X 标出相应的内存块，以免 Soft－ICE 把这些内存派作它用，发生冲突。

修改时用光标键把光标移到相应的内存块上，然后直接按下相应的字符键。修改完后敲回车键返回主菜单。记住，在退出 EMMSETUP.EXE 时，应选第一项 Exit and Save Changes（退出并储存）才能对 S－ICE.EXE 作出真正的改动。

例如你的显示卡是 Hercules 卡，并且正在调试的程序只用字符显示模式 7（分辨率为 720X350），那么将只占用从 B0000H 开始的 4K 内存，你可以把除 B000H 行的第一块外的所有显示内存都改为 E，使 Soft－ICE 可以把这些区域用在其它地方，充分利用内存空间。

5 其它调试工具简介

5.1 MS—DEBUG

DEBUG 可能是大家最熟悉的也是最基本的调试工具了。它具有很强的功能，但由于 DEBUG 是行命令控制的，显示不够直观，也没有提供热键弹出，因此 DEBUG 属于较原始的调试工具。但是，若 DEBUG 和其它一些应用程序如 GAMETOOL 或 GAME BUSTER 4.0 进行辅助调试，则可能有意想不到的好处。

下面简要介绍 DEBUG 的使用方法：

5.1.1 DEBUG 的基本命令

A [adr]	汇编 [起始地址]
C range adr	比较 第一块内存地址范围 第二块内存起始地址
D [range]	显示 [地址范围]
E adr [list]	输入 起始地址 [内容表]
F range list	填充 起始地址 内容表
G [=adr] [adres]	运行 [=起始地址] [断点地址]
H value1 value2	计算 (value1+value2) (value1−value2)
I port	显示端口值
L [adr] [drive] [sector] [number]	调入 [起始地址] [驱动器号] [首扇区] [扇区数]
M range adr	复制 第一块内存地址范围 第二块内存起始地址
N [pathname] [arglist]	命名文件名 [路径] [文件名]
O port byte	输出 端口地址值
P [=adr] [number]	执行一步 [=起始地址] [执行次数](不进入子程序)
Q	返回 DOS
R [register]	察看或修改寄存器 [寄存器]
S range list	搜寻 起始地址 内容表
T [=adr] [value]	单步执行 [=起始地址] [执行次数] (进入子程序)
U [range]	反汇编 [起始地址]
W [adr] [drive] [sector] [number]	写盘 [起始地址] [驱动器号] [首扇区] [扇区数]
XA [#pages]	取句柄并分配扩展内存 [页面帧段]
XD [handle]	释放句柄和存储器 [句柄]
XM [Lpage] [Ppage] [handle]	映象存储器 [逻辑页面] [物理页面] [句柄]
XS	显示扩展内存状态

5.1.2 DEBUG 的使用技巧

1) 利用 DOS 的 ANSI.SYS 程序提供常用按键设置

对于在使用 DEBUG 中经常用到的调试命令，如被调试程序带有循环次数较多的代码时，可能要不停地重复键入一些 DEBUG 的命令键，如：

```
—G 1234
—T
—G 1234
—T
……
```

由于 DEBUG 程序是使用标准的键盘调用，这就可以用 DOS 的自定义键功能。其方法如下：

(1) 在 DOS 的 CONFIG.SYS 中加入

```
DEVICE=C:\DOS\ANSI.SYS
```

(2) 改动 AUTOEXEC.BAT，在原 PROMPT 语句之前加上自定义键说明。对于上例，若要求当按下 F12 后，相当于输入 G 1234 和 T ，可加入以下内容：

```
PROMPT $e[0;134;"G 1234";13;"t";13p
```

其中：符号 $e[为此格式必须有的，e 可用大写字母 E ；0;134 表示 F12 键；双引号中的内容就是当按了自定义键后等效于逐个键入的字符；13 表示回车；p 表示结束(必须用小写字母 p)。常用的功能键 F1 到 F12 的键值如下：

键	键值	SHIFT 键值	CTRL 键值	ALT 键值	F1	0;59
0;84	0;94	0;104	F2	0;60	0;85	0;95
0;105	F3	0;61	0;86	0;96	0;106	F4
0;62	0;87	0;97	0;107	F5	0;63	0;88
0;98	0;108	F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110	F8	0;66
0;91	0;101	0;111	F9	0;67	0;92	0;102
0;112	F10	0;68	0;93	0;103	0;113	F11
0;133	0;135	0;137	0;139	F12	0;134	0;136
0;138	0;140					

其它键盘上每个键的键值可由 DOS 4.0 以上的 HELP.COM 文件对 ANSI.SYS 的帮助

信息中可查到。

2) 利用 DOS 的重定向功能

若已将一个程序调试完毕，如用 DEBUG 找到了需要输入密码的软件中一条判断密码正确于否的指令且将其改掉后，你可能记下这部分的程序代码，但在原程序中找不到，甚至将该软件所有的数据进行搜索也找不到，这可能因为这些程序已经用 PKLITE 或 LZEXE 、PACKEXE 等软件压缩过，还可能是程序自带的可执行代码加密等手段，也即程序是动态生成的。若是这样，一个较简单的方法是将执行到该指令之前所有的 DEBUG 行命令用一个文本文件记下来，以后用 DEBUG 带程序名并用输入重定向的方法来执行带密码的程序。例如对某个 GAME.EXE 进行调试，记下了调试过程，生成 PLAY.DAT 的文本文件，用以下格式来运行解了密的程序：

```
DEBUG GAME.EXE < PLAY.DAT > NUL
```

其中，上句还利用了输出重定向 >NUL，这样可避免 DEBUG 的显示输出影响游戏程序原有的正常显示。

这种方法有缺点，就是只能用在完整的调试过程中。这是因为 DEBUG 对 DOS 的重定向功能是不能简单改变的。若只记录调试程序进行到一半时的 DEBUG 命令，就用输入重定向来进入，则后面所有的键盘输入 DEBUG 是不于理睬的，使得后继的调试不能进行下去。

3) 用 GAMETOOL 配合 DEBUG 调试程序

GAMETOOL 是一个以 TURBO PASCAL 编写的软件破解系统，约占四十 K 的内存。因为 GAME TOOL 是用 TURBO PASCAL 编写，所以所有数字输入格式均与 TURBO PASCAL 的格式相同。例如，十六进制数字之前要加上“$”。

使用 GAMETOOL 来辅助调试，应先运行 GAMETOOL.EXE，使之驻留于内存。当载入 GAMETOOL 后，用户可按热键“PrtSc”弹出 GAMETOOL，可见到以下画面：

```
GAMETOOLS V2.72 10-2-92 Written by Wong Wing Kin
AX=1002 BX=037C CX=0001 DX=0000 SP=0A56 BP=0A68 SI=00EE DI=03BC
DS=0040 ES=0129 SS=0129 CS=F000 IP=9F26 Flags=202 PSP=38D0

MAIN MEMU
[I] Trace an INT's IO                [3] Generate an INT 3
[T] Trace to find the code           [9] Change/Swap INT 8,9,16
[A] Analyse reference segment        [R] Reinit analysis
[L] List addresses                   [Q] Quit to Dos
[K] Keep memories constant           [U] Uninsatll
[V] RAM View                         [M] Return to Text mode 7
[C] Clock frequence                  [ESC] Exit to game
```

画面上方显示出刚进入 GAMETOOL 时各寄存器的值和 PSP 的地址。

在调试程序时，对 DEBUG 最有用的是 GAMETOOL 以下功能：

(1) 被调程序执行时，热键 PrtSc 弹出。

DEBUG 的一个极大的弱点就是无热键弹出。GAMETOOL 正好弥补了这个缺点。它使得程序能在运行时立即弹出，进行各种操作。

(2) 利用 Generate an INT 3。

在主菜单下，按“3”，则显示如下：

```
GENERATE AN INT 3
Current INT 3=0070:06F4

  [0] 0070:06F4(original)
  [1] 0070:06F4
  [2] Enter address
  [3] Generate an INT 3
  [4] Generate an INT 3 at the point of Exit
  [5] Generate an INT ?
```

其中，较有用的是 [3] 和 [4] 项。

a. 键入[3] 后，可退到 DEBUG 下，这时可运行一些 DEBUG 的命令。但要注意，这时/所见到的代码不是被调试的程序，它仍然在 GAMETOOL 之下，最后要用 DEBUG 的“G ”命令返回 GAMETOOL。注意，此时不要用“Q”命令退到 DOS ，否则将死机。

b. 键入[4] 后，也退到 DEBUG 下，于 [3] 不同的是此时所见的代码是被调的程序代码，可以单步执行。通常，用户应该用此项进行调试程序。

(3) 利用 Change/Swap INT 8,9,16

在主菜单下，按“9”，则有如下显示：

```
CHANGE THE ADDRESSES OF INT 8,9&16
Current INT 8=F000:FEA5 INT 9=0562:02F4 INT 16=F000:E82E

  [0] F000:FEA5 & 0562:02F4 & F000:E82E (Original)
  [1] F000:FEA5 & 0562:02F4 & F000:E82E
  [2] Enter address
```

有时，一些软件为了防止他人用调试程序跟踪，将一些重要的中断改掉，如时钟中断(INT 8)，键盘硬件中断 (INT 9)，键盘 BIOS 处理 (INT 16) 。在跟踪这种程序时，若直接用 GAMETOOL 的 [3] Generate an INT 3 功能，由于键盘中断已被改掉，使得 DEBUG 不能接受用户的键盘输入。这时就应该用 CHANGE THE ADDRESSES OF INT 8,9&16 这功能，先将各中断指向 [0] Original 原始的服务程序，再用 Generate an INT 3 功能，就能使 DEBUG 正常工作。当调试过程结束，需要再次进入被调程序，则应先将各中断指回 [1]，否则被调程序可能又不能正常工作。

4)用 GAME BUSTER 4.0 配合 DEBUG 调试程序

用 GAME BUSTER 配合 DEBUG 调试，只使用其中的 Load/Save Game 功能。你也许会碰到这样的情况：在调试程序时，不知一条 CALL 指令运行后会使程序去到哪个地方。这时可弹出 GAME BUSTER ，先用 Load/Save Game 功能将当前现场存盘，再用“P ”命令。即使程

序进入 CALL 后不出来甚至运行死机，只要再弹出 GAME BUSTER，使用 Load/Save Game 功能将上一次的存盘记录调入，就可接着调试，而不是退出被调程序或 BOOT 机，再重新进入 DEBUG，运行大量的指令，到达刚才的指令。这的确大大加快了调试的进度，是一个很值得推荐的方法。

因 GAME BUSTER 是将当前所有的环境及程序所用的内存保存在磁盘上，下次开机时还可再用 GAME BUSTER 来调入最后的运行状态。但使用 GAME BUSTER 仍需注意一点，就是 Load 和 Save 两次开机所设置的软件环境一定要一样，否则将出现内存定位错误的现象，使得程序不能装入。这意味着两台不同的微机用 GAME BUSTER 所保存的 Save Game 文件是不能互相通用的。

5.2 SYMDEB（符号 DEBUG）

SYMDEB 最大的特点是不用地址，而是用符号名来引用数据和指令。它既可以在源程序级也可以在汇编级上调试程序。

在利用 SYMDEB 之前，先要用 MAPSYM 程序将源程序的符号印象文件（.MAP）中的内容转换成 SYMDEB 所要求的格式，输出到文件（.SYM）中。

MAPSYM 命令的格式为：

MAPSYM [{/|-}l][{/|-}n]filename[.MAP]

其中：

filename　指用 LINK 产生的符号印象文件，缺省的扩展名为 .MAP。输出的 .SYM 也将用此文件名。

/l 或-l　　将转换信息显示在屏幕上。

/n 或-n　　指出 filename.MAP 不含有行号。

如果使用 Microsoft C 6.0 的 ILINK，它可以直接生成 .SYM 文件，可以不必用 MAPSYM 来产生。

注意：在使用汇编语言编写的程序时，所有符号应定义为公共符号，而段名不应定义为公共符号。

启动 SYMDEB 的命令行格式为：

SYMDEB [/M][/X][/Wn][/@filename][/N][/I[BM]][/Ffilename]
[/"command[;command;...]"] filename [argument,...]

其中，各参数意义如下：

/M　　输出重定向到辅助显示器。

/X　　取消 SYMDEB 的显示满屏则暂停功能，若不用此选项，进入 SYMDEB 后，按 SCROLL LOCK 键可禁止或恢复此功能。

/Wn　　设置内存分配的报告级，n 为 0,1,2,3。其中，0 表示不报告，1 表示仅显示内存分配信息，2 表示仅显示内存移动情况，3 表示 1 和 2 的内容都显示。

/@filename	指明 SYMDEB 的命令行参数由 filename 提供。filename 的格式开始应为 mnumber=，等号后写命令行参数的内容。
/N	若机上装有 IBM 专用调试板或 Atron 的软件检查板，此选项允许这些硬件产生中断。一般用户不必用此选项。
/I[BM]	若有此选项，SYMDEB 将认为机器是与 IBM 完全兼容的。
/Ffilename	防止 filename.SYM 与 filename 可执行文件相结合。一般用户不必用此选项。
/command	启动 SYMDEB 后立即执行的 command 命令。command 中的各条命令之间用分号隔开，整个命令表用双引号括起。
filename	被调试的文件名。
argument	被调试的文件名的命令行参数。

进入 SYMDEB 后，可见到如下显示：

```
Microsoft (R) Symbolic Debug Utility  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985.  All rights reserved.
Processor is [80286]
-
```

键入？可得到帮助信息，显示如下(已加入中文注释)：

```
A [<address>] - assemble
  汇编(地址)
BC[<bp>] - clear breakpoint(s)
  清除断点(断点号)
BD[<bp>] - disable breakpoint(s)
  禁止断点(断点号)
BE[<bp>] - enable breakpoint(s)
  允许断点(断点号)
BL[<bp>] - list breakpoint(s)
  列出断点(断点号)
BP [bp] <address> - set breakpoint
  设置断点(断点号)(地址)
C <range> <address> - compare
  比较(范围)(地址)
D[type][<range>] - dump memory
  显示内存(类型)(范围)
E[type] <address> [<list>] - enter
  输入(类型)(地址)(项)
F <range> <list> - fill
  填充(范围)(项)
G [=<address> [<address>...]] - go
  运行(地址)
H <value> <value> - hexadd
  十六进制加法(值)
I <value> - input from port
M <range> <address> - move
  移动(范围)(地址)
N <filename> [<filename>...] - name
  命名(文件名)
O <value> <byte> - output to port
  输出到端口(值)(字节)
P [=<address>] [<value>] - program step
  程序单步执行(地址)(值)
Q - quit
  退出系统
R [<reg>] [[=] <value>] - register
  列出/改变寄存器值
S <range> <list> - search
  搜寻(范围)(项)
S {-|&|+} - source level debugging
  源级调试
T [=<address>] [<value>] - trace
  跟踪(地址)(值)
U [<range>] - unassemble
  反汇编(范围)
V [<range>] - view source lines
  浏览源行(范围)
W [<address> [<drive><rec><rec>]] - write
  写(地址)(驱动器)(记录)
X [?] <symbol> - examine symbols(s)
```

端口输入(值)

K [<value>] — stack trace

堆栈跟踪(值)

L [<addr> [<drive><rec><rec>]] — load

调入程序

? <expr> — display expression

求表达式的值

! [dos command] — shell escape

执行 DOS 命令或去 DOS SHELL

. — display current source line

显示当前的源程序行

\ — screen flip

屏幕过滤

检查符号(符号)

XO<symbol> — open map/segment

打开符号表/段

Z <symbol> <value>

给符号赋值

> } <device/file> — Redirect output

输出重定向

< { <device/file> — Redirect input

输入重定向

= ~ <device/file> — Redirect both

输入输出重定向

* <string> — comment

<expr> ops: + — * / : not seg off by wo dw poi port wport mod and xor or

<type> : Byte, Word, Doubleword, Asciz, Shortreal, Longreal, Tenbytereal

SYMDEB 的使用方法于 DEBUG 有很多相似之处。SYMDEB 还可在调试过程中,用"!"运行 DOS 的命令。这要注意最好不要用"!"命令运行其它程序,以免返回 SYMDEB 后,原程序的内存环境和数据已被改动,造成程序不能运行下去。

5.3 FSD (FULSCREEN DEBUG)

5.3.1 FSD 特色简介

FSD 是 FULSCREEN DEBUG 的缩写,是 IBM 公司内部使用的全屏幕调试器。

FSD 与 DOS 下的 DEBUG 相比,最突出的优点就是直观。FSD 运行后,在显示屏上可直接看到每一个寄存器的值、堆栈段顶部的值、各种标志状态、两个数据显示区(其中一个有对应的 ASCII 码显示区)、调试命令输入区、反汇编程序列表区和 F1 至 F10 功能键显示区。运行后的显示如下:

```
AX 0000   SI 0000   CS 2FCE   IP 0100   Stack +0 0000          FLAGS 0200
BX 0000   DI 0000   DS 2FCE              +2 0086
CX 0000   BP 0000   ES 2FCE              +4 33DC    OF DF IF SF ZF AF PF CF
DX 0000   SP 2C3E   SS 2FCE   FS 2FCE     +6 0069     0  0  1  0  0  0  0  0
.....................................................................................
CMD > _                                  | 1          0  1  2  3  4  5  6  7
.....................................    | DS:0000    CD 20 E5 8A 46 9A 3E 2B
                                         | DS:0008    58 FD 67 02 A2 22 58 33
0100 FD             STD                  | DS:0010    A2 22 80 26 A2 22 2E 50
```

```
0101 20B104D2        AND   [BX+DI+D204], | DS:0018   47 81 47 02 1D 0C 81 44
DH
0105 EC              IN      AL,DX       | DS:0020   2D 00 01 55 8B EC 83 EC
0106 0AC4            OR      AL,AH       | DS:0028   02 56 1E 56 3E 39 01 26
0108 0522A2          ADD     AX,A222     | DS:0030   80 3D 14 00 18 00 CE 2F

010B 12B8000F        ADC    BH,[BX+SI+   | DS:0038   FF FF FF FF 00 74 19 C4
0F00]
010F CD10            INT     10          | DS:0040   06 14 01 26 80 3D 00 75
0111 41              INC     CX          | DS:0048   07 80 3E 04 00 01 74 08
```

```
2          0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000  CD 20 E5 8A 46 9A 3E 2B    58 FD 67 02 A2 22 58 33  . ..F.>+  X.g.."X3
DS:0010  A2 22 80 26 A2 22 2E 50    47 81 47 02 1D 0C 81 44  ."."&."."P  G.G....D
DS:0020  2D 00 01 55 8B EC 83 EC    02 56 1E 56 3E 39 01 26  -..U....  .V.V>9.&
DS:0030  80 3D 14 00 18 00 CE 2F    FF FF FF FF 00 74 19 C4  .=...../  .....t..
DS:0040  06 14 01 26 80 3D 00 75    07 80 3E 04 00 01 74 08  ...&.=.u  ..>...t.
```

```
1  Step   2StepProc 3Retrieve 4  Help   5Set BRK  6       7 up   8 dn   9 le   0 ri
```

此时按 F7 (上)、F8 (下)、F9 (左)、F10 (右)可转换光标位置，进入不同的显示区域。按 F4 可得到帮助信息。按 F1 是单步执行，相当于 DEBUG 的 T 命令。按 F2 是单步，相当于 DEBUG 的 P 命令。

FSD 的命令可以在命令输入区输入，有些命令还可以在不同显示区域内直接修改。在命令输入区输入的命令还可以存储在内存中，按 F3 就可轮流显示。这些命令还可以形成一个文件，方便今后的调用。

5.3.2 FSD 的调试命令

下面列出了所有的 FSD 命令，我们将逐条详细说明。

(1) L fspec {param.}{,addr}

装入被调试的文件 fspec (文件不能含有路径)，若无扩展名，缺省值为 EXE 文件，文件名后还可带有被调试文件的命令行参数 param，若需要指定被调试文件装入的起始偏移地址，可带上 addr 指明。在未进入 FSD 之前，也可将被调试的文件名作为 FSD 的命令行参数，FSD 将自动解释为 L 命令。

例：

```
CMD >L MYPROG      调入 MYPROG.EXE 文件
CMD >L FORMAT A:/Q/U/S
C:\>FSD MYPROG     DOS 提示符下，直接输入文件 MYPROG
C:\>FSD FORMAT A:/Q/U/S
```

W fspec,addr,length　　将数据写到文件 fspec，数据缺省为 DS 段，length 为写入的字节数(十六进制)，最大为 FFFF。

例：CMD >W MYDATA.DAT,0000,100　　将 DS:0000 至 DS: 00FF 的数据写入文件 MYDATA.DAT 中

CMD >W DATA.DAT CS:BX+123,8　　将 CS:BX+123 处的数据写入 DATA.DAT 中

(2) {R} reg=value

修改寄存器的值，"R" 可以省略。若要修改标志寄存器，可以直接写为 FL=XXXX，也可对标志寄存器的某一位设为 0 或 1。可单独修改的标志名称如下：

OF	溢出标志
DF	方向标志
IF	中断允许标志
SF	符号标志
ZF	零标志
AF	辅助进位标志
PF	奇偶标志
CF	进位标志

但单步标志是不能单独修改的。

例：

CMD >RAX=1234　　设寄存器 AX 等于 1234H

CMD >AX=1234　　同上

CMD >AX=BX*12+CX+123　　表达式的值赋给 AX 注意：运算规则是从左到右，无符号的优先级别，若要指明计算顺序，可加括号。

CMD >CF=1　　设进位标志为 1

(3) D addr

设置显示代码的起始位置，缺省的段寄存器是 CS。

例：

CMD >D100　　从 CS:100 开始显示程序代码

CMD >DES:BX　　从 ES:BX 开始显示程序代码

CMD >DIP　　从CS:IP 开始显示程序代码。这条指令较常用。因为在观察代码时，若光标上下移动，可能找不到最后运行的指令，用 DI 就可以解决。

(4) M n addr

reg

[reg]

显示内存值。n=1，是指右边的内存显示区，n=2，指左下角的内存显示区，缺省的段地址就是对应显示区所显示的段寄存器。若所需显示的内存的段是用户任意指定的，如段地址 =0000，可令段寄存器设为 FS (FS 是虚拟的寄存器，FS 可当作一般寄存器那样进行修

改)。当用寄存器作为偏移地址时,也可用"[]"括起来表示。在内存显示区域 1 或 2 中,也可直接修改段值或偏移量。

例:CMD >M 1 1000　　从内存显示区域 1 所指的段中,偏移地址为 1000 的地址开始显示。

CMD >M 1 FS:0　　在内存显示区域 1 中,从 FS:0 开始显示。

CMD >M 1 DS:DX　　从 DS:DX 开始显示。

CMD >M 1 DS:[DX]　　同上

CMD >M 1 SS:AX+BX*2+10　　从 SS 段,表达式作为偏移量处开始显示。

(5) G {start ad.}{,break ad.}

缺省段为 CS,从 start ad.(若无 start ad. 缺省为从当前位置处)开始执行程序,若设置断点,则可在断所指语句运行前停下。此断点设置较为简单,功能更强的断点设置可用 F5 实现,后面将会详细介绍。

例:CMD >G100　　从 CS:100 开始执行程序

CMD >G100,123　　从 CS:100 开始执行程序,在 CS:123 处停下

CMD >GES:BX　　从 EX:BX 开始执行程序

(6) QUIT

退出 FSD,回到 DOS 下。

(7) BE{EP} ON

OFF

允许或不允许扬声器发出嘟嘟声。

(8) T{B}

显示跟踪缓冲区。这条命令只有当用了 F5 功能键断点设置中的 Action 为 T ON 或 T ONNI 后才能有效。T 命令是在主屏幕的程序列表区显示跟踪缓冲区的内容,而 TB 命令则单独开一个屏幕来显示跟踪缓冲区的内容。用 TB 命令后可见到如下内容:

TRACE BUFFER DISPLAY

```
Buffer Offset :    0
* * *  Start of TRACE data  * * *
01BC PUSH   DS          AX=0000  SI=0000  CS=203D  ZF0  OF0     Stack+     07E81
Started by BP1          BX=0000  DI=0000  DS=203D  AF0  DF0       +2       0000
                        CX=0000  BP=0000  ES=203D  PF0  IF1       +1       7401
                        DX=1827  SP=2B36  SS=203D  CF0  SF0       +6       8107
01BD SUB    AX,AX       AX=0000  SI=0000  CS=203D  ZF0  OF0     Stack+0    ▯203D
                        BX=0000  DI=0000  DS=203D  AF0  DF0       +2       7E81
                        CX=0000  BP=0000  ES=203D  PF0  IF1       +1       0000
                        DX=1827  SP=2B34  SS=203D  CF0  SF0       +6       7401
                          ......
```

按光标键翻页，按 F1 或回车键返回主屏幕。

此跟踪功能可以记录最多达 100 条最后运行的汇编指令。由于记录个数较少，应适当设制 F5 功能的 Action 中的 T ON 和 T OFF 的触发条件。

(9) P addr,string

从 addr（缺省的段为 CS）开始，填写串 string

例：CMD >P101,'MZ'

在 CS:101 处填写串 'MZ'

(10) F addr,repeator,string

基本同 P 命令，只是缺省的段为 DS，重复此次数为 repeator

例：CMD >F100,10,'NONE'　　从 DS:100 开始，填写串 'NONE'，共填 10H 次

CMD >FES:1234,10,AX　　从 ES:1234 开始，填写 AX 的值，共填 10H 次。注意，低位在前，高位在后。

(11) S {addr},string

查找功能。缺省的段寄存器为 CS，若 addr 也省略，则从 CS:0000 开始查找串 string。若找到，将在第二个内存显示区（屏幕左下角）显示，且其段地址设为 FS。

例：CMD >S,CD 21　　从 CS:0000 开始找 CDH,21H

CMD >S,CD21　　从 CS:0000 开始找 21H,CDH

CMD >SDS:0,'ERROR'　　从 DS:0000 开始找串 'ERROR'

(12) I addr

读 I/O 端口，addr 可以是 8 位或 16 位的值。

例：CMD >I60　读 60H 端口

CMD >IDX　读 DX 端口

(13) O addr,value

写 I/O 端口，addr 的范围同命令 I。另外，若 value 的值是字，则进行的是 16 位的字操作。

例：O60,1C　　将字节 1CH 输出到 60H 端口

(14) MO{DE} M{ONO}

C{OLOR}

A{LTERN} ON

OFF

设置显示方式。M 或 MONO 为单色显示方式，C 或 COLOR 为彩色显示方式。若使用 AON 功能，则将开一个 shadow－screen，使得 FSD 的显示不与被调试程序的显示输出因使用

同一个显示页面而发生冲突。按 F6 键可以切换到被调试程序的显示输出平幕。

例:CMD >MO　　显示当前的显示方式

CMD >MO ON　　开 shadow－screen,使 F6 键有效。这条命令较常用

CMD >MO OFF　　关 shadow－screen,使 F6 键无效

(15) BW fspec

将已设置的断点全部写入文件 fspec。

(16) BL fspec

从文件 fspec 中读入断点的设置。

(17) XT

启动 TEACH 方式。启动后,FSD 将记录所有的按键。若 FSD 按键缓冲区已满或按下 CTRL＋BREAK 键,则将结束 TEACH 方式。

(18) XX {fspec}

从 FSD 按键缓冲区或文件 fspec 中读入键且执行。

(19) XW fspec

将 FSD 按键缓冲区写入文件 fspec 中。

(20) XL fspec

读入文件 fspec 到 FSD 按键缓冲区(但不执行)。

XT、XX、XW、XL 在调试大程序时特别有用,它可以记录所按的键,当调试到程序中间时,应不时地用 XW 写到文件中。若程序发生死机,下次可直接用命令 XX 读入后执行到有问题的地方,节省了许多时间。

5.3.3 高级断点设置的使用方法

在主屏幕下按 F5 可进入设置断点的菜单,显示内容如下:

BREAK POINT ENTRY MENU

BR #	Break ADR	Condition	Count	Occur	Action
1	CS:0	.....	0	0	
2	CS:0	.....	0	0	
3	CS:0	.....	0	0	
4	CS:0	.....	0	0	
5	CS:0	.....	0	0	
6	CS:0	.....	0	0	
7	CS:0	.....	0	0	
8	CS:0	.....	0	0	

Break ADR Break point address. Must be set to the begin of an instruction
{Segment:}Offset . Segment may be any segment register or value.
If empty, the condition will be checked on every BREAK.

Condition Condition to be checked when BREAK occures. All conditions are ANDed.

REG=value, [REG]=value, OFFSET=value, BRn. All addr. may be prefixed by a segment register. BRn is true, when the referred OCCUR=COUNT. Value is a up to 4 digit hex number, with 'x' for don't care.

Count　Dec. number of occurences before action is performed. '0' disables BRn

Occur　Decimal number of actual occurences after last 'Go' comand.

Action　Action to be done when break condition is true. T{RACE} ON {NI} or OFF to trace the program execution. NI disables trace of INT routines. C{OUNT}, S{TOP}, R{ST}n,m... With 'RST' the OCCUR of BRn is reset to 0

1 View Trace　3 Read Setup　4 Save Setup　5 Return to Main

FSD 可设置多达 8 个断点，而且每个断点又可设置断点条件。当被调试程序运行到断点时，FSD 的断点中断服务程序首先判断断点条件是否满足，若条件满足，则运行 Action 中所指定的内容，若条件不满足，继续运行被调试的程序。熟练地使用这些强劲的断点功能，可以使得调试程序的效率大大地增加。

断点地址的设置

Break ADR 的段地址可以是段寄存器或数值，也可以按 ESC 键或空格键不写任何内容（此时，若此条的 Count 不为 0，则程序碰到任何断点都将检验该条的断点条件以及运行 Action 的内容）。

断点条件可以是：

REG=value

[REG]=value　用间接寻址方式

OFFSET=value　内存直接判断方式

BRn　断点 n 的 OCCUR = COUNT 满足时，其值才为真。

在 Condition 一栏内，可以没有任何叛别条件，也可以同一行可有多个判别条件，这些条件同时成立时才执行 Action 后的内容。另外，value 的表达式中可以用 x 来屏蔽某一位。

Count 指程序运行到断点多少次才开始判断断点条件。注意，此数是用十进制来表达的。程序每经过一次断点，Occur 的值加 1。当 Occur = Count 时，再判断断点条件。另外，若设 Count 为 0，则程序运行时，忽略该断点。

当断点条件满足时，FSD 将执行 Action 中的内容，其内容也可以是空的。Action 的表达式可以是：

T{RACE} ON	开始跟踪代码并记录下来
T{RACE} ON NI	基本同上，但不跟踪进任何中断程序
T{RACE} OFF	结束跟踪
C{OUNT}	简单地对 Occur 加 1
S{TOP}	终止程序的运行，回到 FSD 主屏幕下
R{ST}n,m...	对第 n,m... 个断点的 Occur 清零

一般情况下，S 应至少有一处要设置，否则程序可能不能正常退出。

以上就是 FSD 的使用方法和技巧，若能熟练掌握 FSD 这调试工具，你将发现调试程序

并不是很困难的事。

5.4 Turbo Debugger

5.4.1 Turbo Debugger 的特点

Turbo Debugger 与普通 DEBUG 相比，增加的调试手段有：

VIEW：　以窗口显示变量，变量值，断点，堆栈，记载，数据文件，源文件，CPU 代码，内存，寄存器，数值处理机信息和程序输出。

Inspector：　可以深入到工作程序内部，如检测堆栈。

Modify：　可以修改全局或局部变量的当前值。

Watches：　可以隔离程序变量，观察它们在程序执行过程中的变化。

Turbo Debugger 的优点：

1) 方便的逻辑下拉菜单
2) 对内容相关的弹出菜单，免去了用户记忆和敲入命令的麻烦；
3) 键盘输入的内容可以存入历史表或文件中；
4) 可定义宏键，加快命令的输入；
5) 方便完整的窗口管理
6) 多类型的在线帮助
7) 可使用 EMS 调试大程序
8) 完全的 C，Pascal 或汇编语言表达式求值；
9) 屏幕布局可重新设置
10) 可进行汇编语言/CPU 级的调试
11) 强有力的断点和记载设施
12) 可使用双机系统调试很大的程序
13) 支持 80386 及一些硬件调试板

5.4.2 Turbo Debugger 软件包

Turbo Debugger 包含以下文件：

1)TD.EXE（Turbo Debugger 运行程序）的命令行格式为：

TD [options] [program [arguments]]　　-x-　= turn option x off

其选择项的功能如下：

-c<file>　　使用 TD 的设置文件

-do　　另一显示

-dp　　页翻转

−ds	交换用户屏幕内容
−h,−?	显示帮助文件
−i	进程 ID 开关
−k	允许击键记录
−l	汇编程序起始
−m<#>	设置堆大小
−p	使用鼠标
−r	远程调试,用 COM1,慢速度
−rp<#>	远程连接 COM 口
−rs<#>	远程连接速度:1= 最慢, 2= 慢, 3= 中等, 4= 快
−sc	忽略大小写
−sd<dir>	源文件的目录
−sm<#>	设置剩余的符号表内存(最大为 256Kb)
−vg	完整的图形保存
−vn	禁止 43/50 行显示
−vp	允许保存 EGA/VGA 的调色盘
−w	调试远程窗口程序(与 −r 一起使用)
−y<#>	设置覆盖区大小(Kb)
−ye<#>	设置 EMS 覆盖区大小(页,每页 16Kb)

2) TDINST.EXE (Turbo Debugger 安装程序) 的命令格式为:

TDINST [options] [exefile]

参数意义为:

−c<file>	使用 TDINST 的设置文件
−h,−?	显示帮助信息
−p	使用鼠标
−w	设置 TDW 的参数

3) TDREMOTE.EXE (远程调试程序) 的命令格式为:

TDREMOTE [options]

参数意义为:

−rp＜#＞　　设置 COM 口号：1 = COM1，2 = COM2

−rs＜#＞　　设置联机通讯的速率：1 = 9600bps，2 = 19Kbps，3 = 38Kbps，4 = 115Kbps

−w　　将参数写入 exe 文件

4) TDRF.EXE（远程文件传送程序）的命令格式为：

TDRF [options] command [arguments]

参数意义为：

−rp＜#＞　　设置 COM 口号：1 = COM1，2 = COM2

−rs＜#＞　　设置联机通讯的速率：1 = 9600bps，2 = 19Kbps，3 = 38Kbps，4 = 115Kbps

−w　　将参数写入 exe 文件

Command	Letter	Arguments	Description
copy，copyto	t	1 or 2	拷贝文件到远程系统
copyfrom	f	1 or 2	从远程系统拷贝文件
del，erase	e	1	
dir	d	0 or 1	
ren	r	2	
md	m	1	
rd	k	1	
cd	c	0 or 1	

5) TDSTRIP.EXE（符号表抽取工具）的命令格式为：

TDSTRIP [options] exefile|objfile [outfile]

参数意义为：

−s　生成的符号表文件名

−c　由 EXE 文件生成 COM 文件

6) TDCONVRT.EXE（把 CodeView 程序转化为 Turbo 格式的工具）的命令格式为：

TDCONVRT [options] InFile [OutFile]

参数意义为：

−c　建立有符号表的 .tds 文件名

−s　转换时不显示转换信息

7) TDHELP.EXE（Turbo Debugger 帮助文件）

8) README.COM（README 的阅读程序）

9) README（最新信息）

10) TDH386.SYS（80386 设备驱动程序）

11) TD386.EXE（虚拟调试程序）

12) TDUMP.EXE（通用模块反汇编工具）

13) TDMAP.EXE（把 .MAP 文件信息加到 .EXE 文件末尾的工具）

14) TDPACK.EXE(缩小 .EXE 文件中调试信息大小的工具)

5.4.3 TD 的注意事项

TD 使用了以下技术，用户了解这些技术有助于更好地调试程序。

1) TD 把被调试程序装在高的段地址处，可利用的自由空间少。

2) TD 不使用 80x87，被调试程序可以不必特意保存 80x87 内的寄存器。

3) TD 使用 INT 1 和 INT 3 来处理中断和单步，用 INT 2 实现硬件调试器的中断，并使用 INT 9 来追踪键的按下和释放。若用户程序用了 INT 1 或 INT 3，必须在程序中用完这些中断后，立即恢复原先的中断向量值。

4) 若机器安装了 EMS，TD 将在这放符号表。若被调程序要使用全部的 EMS 存储空间，可以用 TDINST 来禁止 TD 使用 EMS 存储。

5) TD 保存从 00H 到 2FH 的 48 个中断向量的三个独立拷贝。刚启动 TD 时，作第一个拷贝，在用 ALT+X 或 OS SHELL 退出 TD 后，恢复第一个拷贝。第二个拷贝为 TD 的向量表，每次从被调程序转到 TD 时就恢复第二个拷贝。第三个拷贝为被调程序的向量表，在被调程序停止和 TD 获得控制时保存这些值，在每次运行或单步执行被调程序时，就恢复第三个拷贝。

5.5 Codeview

Codeview 简称 CV，其使用方法很象 Turbo Debugger，也是菜单式的用户界面。CV 命令行格式为：

cv [options] file [arguments]

其参数意义为：

/B　　强制 CV 使用单色显示器。

/Ccommand[;command]... 进入 CV 后立即执行的命令，命令用双引号括起。

/D[buffersize]　　使用 CV 的覆盖文件，可以获得更多的用户程序内存。buffersize 缺省时，CV 使用 64K 内存。

/E　　允许 CV 使用 EXPANDED 内存（必须有 EMM 支持）。

/F　　CV 与被调程序都用第 0 显示页。

/G　　禁止 CGA 显示时产生雪花，但会降低显示速度。

/I[0 | 1]　　/I0 强迫 CV 截获 NMI 和 8259 产生的中断，且使 CTRL+

BREAK 有效，/I 或 /I1 禁止截获 NMI 和 8259 产生的中断。

/K　　禁止 CV 截取键盘中断。

/M　　禁止 CV 使用鼠标。

/N[0 | 1]　　类似 /I 参数，但 /N 只对 NMI 有效。

/R　　使用 80386/486 寄存器，可以加快设置了断点的程序的执行。

/S　　CV 与被调程序用不同显示页。

/TSF　　读或忽略 CURRENT.SYS 文件。在设置 TOOLS.INI 中，若设 Statefileread 开关为 yes，则 CV 将会忽略 CURRENT.SYS 文件。反之，CV 读 CURRENT.SYS 文件的设置。

/X　　允许 CV 使用 EXTENDED 内存（必须有 HIMEM.SYS 或其它括充内存管理程序支持）。

/25　　设置每屏 25X80 格式。

/43　　设置每屏 43X80 格式。

/50　　设置每屏 50X80 格式。

/2　　使用双显示器。被调试程序在当前缺省显示器上显示，CV 在另一显示器上显示。

6 常用反动态跟踪技术

当我们跟踪被加过密的软件时，常常会面对各种各样的反动态跟踪技术。如果解密者对这些反动态跟踪技术一无所知，那么，进行动态跟踪的难度和复杂度是可想而知的。

在此，我们将介绍几种常见的反动态跟踪技术，以便在进行动态跟踪时，能够及时地发现类似的反动态跟踪手段，并且有效地蔽开它们。

6.1 反动态跟踪技术的分类

反动态跟踪技术针对不同的主体，有不同的方法：

6.1.1 针对动态调试工具

因受条件的限制，绝大多数用户进行动态调试和动态跟踪时，都使用 DEBUG，SYMDEB，TURBO DEBUG，CODEVIEW 等调试软件。而这些调试软件要正常工作，就必须依靠特定的运行环境。如果采取一定的措施破坏调试软件赖以生存的环境，使其无法正常工作，那么就达到了反动态跟踪的目的。一个典型的例子就是 INT 1 和 INT 3 中断，因为多数调试程序依靠 INT 1 来单步中断，用 INT 3 进行断点中断，如果破坏了这两个中断，当然就难以进行动态跟踪了。

6.1.2 针对跟踪者

从理论上讲，任何加密措施都是可破译的。但是，如果跟踪者花费了大量的时间和精力才得以解密的话，往往是得不偿失的。所以，针对跟踪者精力和时间有限这一点，加密软件通常设置了许多繁琐的大小循环，以耗尽跟踪者的精力。一些反复被调用且具有多个出口的子程序就属于这一类反动态跟踪措施。

6.1.3 综合措施

把上述的两种方法结合起来，并且再辅以其它的一些措施，多种方法交叉使用，将会使跟踪难度大大增加。

6.2 反动态跟踪技术的常见方法

以下的方法经常出现在加密软件中，只要你留心，就会发现它们原封不动或略微改变地隐藏在程序某处。

6.2.1 抑制跟踪命令

DEBUG 的关键跟踪命令是 P，T，G。

执行 G 命令时，DEBUG 在意欲设置断点的地方放入单字节指令 INT 3 （机器代码为 CCH），并使中断向量表中的 INT 3 指向 DEBUG 的断点中断服务程序的入口地址。这样，当程序运行到断点处时，DEBUG 的断点中断服务程序将得到控制权。它首先把断点处的 CCH 指令恢复到程序原来的状态，然后返回 DEBUG 的控制台面，显示有关信息及等待接受 DEBUG 命令。

DEBUG 在执行 P 和 T 命令时，首先将 CPU 状态寄存器 SR 中的单步跟踪 T 位置 1，并把中断向量表中的 INT 1 向量指向 DEBUG 的单步中断服务程序。这样，在执行完一条指令后，DEBUG 的单步中断服务程序将得到控制权。它首先将 SR 中的 T 位清零，然后返回 DEBUG 的控制台面，显示有关信息及等待接受 DEBUG 命令。

由于其它程序一般不用 INT 3 和 INT 1，所以反动态跟踪的程序往往有意无意地把中断向量表中的 INT 3 和 INT 1 向量设成随机值，或者将它们设置成重要运行参数的工作单元，甚至指向自我毁灭的模块入口。这样，要么无法返回 DEBUG，要么破坏程序的关键数据使其无法正常运行。因此，在动态跟踪时要密切注意 INT 1 和 INT 3 向量是否被程序改动。

在 DOS 系统的中断向量表中，单步中断（INT 1）服务程序的入口地址放在内存 0000:0004H 开始的 4 个字节处，断点中断（INT 3）服务程序的入口地址放在 0000:000CH 开始的 4 个字节处。下面的代码就是经常被使用的反动态跟踪手段：

```
XOR   AX, AX
MOV   ES, AX
MOV   SI, 0000
MOV   DI, 000CH
MOV   CX, 0004
REP   MOVSB
```

这段程序用 DS:0000 处的 4 个字节覆盖断点中断 INT 3 的入口地址，以破坏 G 命令的正常执行，达到反动态跟踪的目的。

```
PUSHF
POP   AX
TEST  AX, 0100H
JNZ   ERROR
```

这段指令检测状态寄存器 SR 的跟踪位 T 是复位还是置位。如果是复位状态，则正常执行；如果是置位状态，则表示检测到程序正被动态跟踪，于是转向错误的出口。

当然，实际的代码不会是这样明显的，需要你细心地去发现。例如：

```
MOV   AX, 0F000H
MOV   ES, AX
MOV   DI, 0F000H
MOV   SI, 0
MOV   CX, 808H
CLD
REP   MOVSW
```

在此处，运用了段返卷技术。当传送完 800H 个字时，DI 寄存器变为 0，附加数据段 ES 返卷回 0000:0000H（F0000H＋F000H＋1000H＝100000H，最高位丢失，从而形成线性地址 00000H），即接下来传送的 8 个字将覆盖中断向量表中的 INT 0 到 INT 3 四个中断向量。如果是利用堆栈段的段返卷技术则更难以发现。

利用 INT 3 来取代一些常用的中断调用功能，也能很好地抑制断点中断。请看以下这段代码：

```
        PUSH   DS
        PUSH   ES
        MOV    CX, 0004
        XOR    AX, AX
        MOV    DS, AX
        MOV    ES, AX
        MOV    SI, 0084H
        MOV    DI, 000CH
        REPZ   MOVSB
        POP    ES
        POP    DS
        MOV    DX, OFFSET msg
        MOV    AH, 09
        INT    3
        ……
        MOV    DX, OFFSET filename
        MOV    AX, 3D00H
        INT    3
        JC     error
        MOV    BX, AX
        MOV    AX, 3F00H
        MOV    DX, OFFSET buf
        MOV    CX, 0100H
        INT    3
        ……
error:MOV   DX, OFFSET error_msg
```

```
MOV   AH, 9
INT   3
……
```

上面这段程序，首先把 INT 21H 中断向量的地址传送至 INT 3 中断向量地址上，然后利用 INT 3 中断间接地执行 INT 21H 软中断调用。由于 INT 3 指令的机器代码只有一个字节(CCH)，而 INT 21H 的指令代码则为两字节(CDH 21H)，所以解密时若想将 INT 3 指令直接改为 INT 21H 指令是相当困难的，而且断点中断 INT 3 也被破坏了。

6.2.2 封锁键盘输入

由于要用 DEBUG 命令来察看跟踪情况，所以有的程序采用封锁键盘输入的方法来阻止正常接收 DEBUG 命令。

键盘中断服务程序的中断向量是 INT 9，如果其入口地址被修改了，就无法完成正常的键盘输入操作，使 DEBUG 不能接收命令，从而达到反动态跟踪的目的。

键盘中断是一个可屏蔽的硬件中断，中断屏蔽寄存器的第一为置 1 将使键盘屏蔽：

```
IN    AL, 21H
OR    AL, 02H
OUT   21H, AL
```

8255 芯片的端口 B (端口地址为 61H)的最高位也可以控制键盘。如果把该位置 1 则表示清除键盘输入，所以下列的代码也是用来阻止动态跟踪的：

```
IN    AL, 61H
OR    AL, 80H
OUT   61H, AL
```

另外，BIOS 的键盘 I/O 中断(INT 16H)也有可能被程序修改，使其只接收程序当中需要输入的键，而不接收 DEBUG 命令中的字符键。

6.2.3 设置显示模式

为了防止跟踪者从 DEBUG 的动态跟踪里获得有关加密的关键信息，加密软件常常破坏正常的屏幕显示，让你无法察看关键代码部分。

UCDOS 2.0 就通过修改显示器 I/O 中断 INT 10H 使 DEBUG 不能正常地显示跟踪信息，而它自己的显示则在顺利通过加密部分后，再恢复 INT 10H 中断向量。

由于嵌入被保护程序中的识别程序在执行期间通常不需要在屏幕上显示信息，为了不让跟踪者察看 DEBUG 显示的信息，识别程序将屏蔽显示器或者重新设置显示模式使前景色和背景色为同一种颜色，这样，要么屏幕没有显示，要么整个屏幕是一片，什么信息都和底色混在一起，无法看到 DEBUG 提供的信息。

```
MOV   AH, 0BH
```

```
XOR   BX, BX
INT   10H
```

这段代码将屏幕的背景颜色和字符颜色都置成黑色，此后的显示信息将无法看到。

6.2.4 利用时钟中断来反动态跟踪

为了防止你通过修改程序代码来避开反动态跟踪措施，加密软件会利用频繁的时钟中断检测你有否改动程序，或者不断地破坏动态跟踪所必须的环境，甚至通过检测程序的运行时间来判断程序是否是一直在正常运行而没有被调试工具中断过。

时钟中断 INT 8 每秒钟发生 18.2 次，每发生一次，计时单元 0040:006CH —— 0040:006FH 的值加 1；累计超过 24 小时，则将 0040:0070H 单元的值加 1，并把计时单元 0040:006CH —— 0040:006FH 清零。除此之外，还将管理软盘驱动器的启闭时间和调用一次嵌入软中断 INT 1CH。INT 1CH 中断正常情况下只包含一条中断返回指令 IRET，而不做其它的操作。被加过密的程序把 INT 1CH 或者 INT 8 指向它自己的处理程序，而这些处理程序通常会破坏动态跟踪所必须的条件。

因为 DEBUG 的 G 命令执行时，可能会完全停止时钟计数，所以如果用 G 命令跳过以下代码段，系统将陷入死循环：

```
XOR   AX, AX
    MOV   DS, AX          ;将计时单元段址放入 DS 寄存器
    MOV   SI, 046CH       ;将计时单元偏移放入 SI 寄存器
    MOV   AX, [SI]        ;读时钟低位到 AX 寄存器
LOOP:MOV   BX, [SI]       ;再次读时钟低位到 BX 寄存器
    CMP   AX, BX          ;比较时钟低位有无变动
    JZ    LOOP            ;若无变动则陷入死循环
```

又因为 DEBUG 的 P 和 T 命令跟踪时，程序的执行速度比正常运行速度慢几个数量级。若在某模块的入口和出口处分别读取时钟值，然后加以比较，看二者的差值是否比正常情况下大许多，从而判别该程序有没有被动态跟踪：

```
模块入口: MOV   AH, 00
          INT   1AH       ;调用 BIOS 中断读取当前时钟值
          PUSH  CX        ;保存当前时钟值高位部分
          PUSH  DX        ;保存当前时钟值低位部分
          ......          ;模块代码部分
模块出口: MOV   AH, 00
          INT   1AH       ;调用 BIOS 中断读取当前时钟值
          POP   BX        ;取入口处时钟值低位部分
          POP   AX        ;取入口处时钟值高位部分
          SUB   DX, BX    ;求得二者差值
```

```
SUB    CX, AX                ;进而获得模块执行所花费的大概时间
```

6.2.5 利用其它中断实现反动态跟踪

1) 溢出中断

当上条指令的运算结果溢出时,使标志寄存器的第 0 位(溢出标志)置 1,则指令 INTO 产生一个 INT 4 中断。加密软件能利用修改该中断来反动态跟踪:

```
        XOR    BX, BX
        MOV    CX, NUMBER
        MOV    SI, ADDRESS1
REPEAT:SUB     WORD PTR ADDRESS2, VAR1
        LODSW
        SUB    WORD PTR [DI], VAR2
        ADD    BX, AX
        INTO
        LOOP   REPEAT
```

从 ADDRESS1 开始放着一大批准备依次送给 AX 参加运算的不同值,可能要经过千百次循环才能使加法溢出。这时,在经过改动的 INT 4 中断服务程序中,根据 ADDRESS2 所指的值,DI 变址所指向的值,BX 寄存器的值,转向不同的地址或返回不同的参数,而究竟哪个是正确的,只有设计者才清楚。

2) 除数为零

在执行除法指令(DIV 或 IDIV)时,若发现除数为零或商超过了寄存器所能表达的范围,则立即产生一个 INT 0 内部中断。因为该中断是内部产生的且在代码中不出现,所以通过使 INT 0 指向破坏跟踪环境的代码段来防止动态跟踪是比较隐蔽的。

3) 无效指令中断

当碰到非法指令时,系统产生无效指令中断 INT 6 。如果修改 INT 6 中断的入口地址,使其首先运行破坏跟踪条件的代码,再执行正常的 INT 6 中断,并且在程序代码段中设置一些无效指令,就可以起到反动态跟踪的作用。由于无效指令中断是一个内部中断,程序里并不出现,所以较难发现。

6.2.6 利用程序设计技巧实现反动态跟踪

1) 代码动态生成

对程序进行分块,上一块执行的结果将决定下面代码的生成,或者为下一块的解密提供数据和运行时的重要参数。因为任何时侯所看到的代码都是不全的,所以跟踪难度相当大。

2) 反穷举法

设置多重循环嵌套,并频繁地调用子程序,而且建立多个返回出口,使跟踪者如果不一条

指令一条指令地跟踪下去就难以找到正确的返回地址，从而消耗其精力。

3）非正常转跳

用 PUSH 指令把地址压入堆栈，然后用 RET 指令来代替 JMP 指令完成无条件转移。这样将使程序难以读懂。

4）多重检测

在关键的代码区域设置多重反动态跟踪检测，一旦发现代码被篡改，则将程序引入歧途。

5）逆指令流

通过重新设置单步中断服务程序，使指令的执行顺序颠倒。因 DEBUG 的反汇编命令 U 只能顺序反汇编机器代码，所以用 U 命令将看不到正确的指令。

6）堆栈覆盖

把堆栈段设在代码段内，跟踪时附加的压栈操作将会覆盖程序代码，使程序无法正常运行；或者在不需要压栈操作的代码部分执行时，把堆栈设在中断向量表的关键位置或系统的敏感地区，使动态跟踪时破坏关键的中断向量，甚至造成整个系统崩溃。

附录一　重要端口

端口号	读/写	说明
20H	R	可编程中断控制器（PIC），由 3 号操作命令字（OCW3）设置的中断请求/服务寄存器。 中断请求寄存器，其中： 第 7 —— 0 位＝0，对应的中断线上无请求活动。 ＝1，对应的中断线上有请求活动。 中断服务寄存器，其中： 第 7 —— 0 位＝0，对应的中断线没有正被服务。 ＝1，对应的中断线正被服务。
20H	W	PIC，1 号初始化命令字（ICW1）（第 4 位为 1），其中： 第 7 —— 5 位＝0，仅用于 80/85 方式 第 4 位＝1，　保留 第 3 位＝0，　保留 第 2 位＝0，　两个连续的中断向量之间有 8 个字节 ＝1，　两个连续的中断向量之间有 4 个字节 第 1 位＝0，　串联方式 ＝1，　单一方式（无需 ICW3） 第 0 位＝0，　无需 ICW4 ＝1，　需要 ICW4
21H	W	PIC，把 ICW1 写入 20H 端口之后，顺序的 ICW2，ICW3，ICW4。 ICW2，其中： 第 7 —— 3 位＝中断控制器的基本向量地址的地址线 A0 —— A3 第 2 —— 0 位＝保留 ICW3，其中： 第 7 —— 0 位＝0，从控制器未连接到对应的中断插针上 ＝1，从控制器已连接到对应的中断插针上 ICW4，其中： 第 7 —— 5 位＝0，保留 第 4 位＝0，　无特殊全嵌套方式 ＝1，　特殊全嵌套方式 第 3 —— 2 位＝00，无缓冲方式 ＝01，无缓冲方式 ＝10，缓冲方式/从方式 ＝11，缓冲方式/主方式

第 1 位=0，　正常 EOI

=1，　自动 EOI

第 0 位=0，　80/85 方式

=1，　8086/8088 方式

21H　R/W　PIC，中断屏蔽寄存器（OCW1），其中：

第 7 位=0，　并行打印机中断有效

第 6 位=0，　软盘驱动器中断有效

第 5 位=0，　硬盘驱动器中断有效

第 4 位=0，　串行口 1 中断有效

第 3 位=0，　串行口 2 中断有效

第 2 位=0，　视频中断有效

第 1 位=0，　键盘中断有效

第 0 位=0，　计数器中断有效

21H　W　PIC，OCW2（第 4 位为 0，第 3 位为 0），其中：

第 7 —— 5 位=000，自动 EOI 方式中旋转（清除）

=001，无特殊的 EOI

=010，无操作

=100，自动 EOI 方式中旋转（设置）

=101，旋转非特殊 EOI 命令

=110，设置优先级命令

第 4 位=0，　保留

第 3 位=0，　保留

第 2 —— 0 位=命令旋转的中断请求

20H　W　PIC，OCW3（第 4 位为 0，第 3 位为 1），其中：

第 7 位=0，　保留

第 6 —— 5 位=00，无操作

=01，无操作

=10，复位特殊屏蔽

=11，设置特殊屏蔽

第 4 位=0，　保留

第 3 位=0，　保留

第 2 位=0，　无轮询命令

=1，　轮询命令

第 1 —— 0 位=00，无操作

=01，无操作

=10，下次读时从端口 20H 读取中断请求寄存器

=11，下次读时从端口 20H 读取中断服务寄存器

60H　W　键盘数据输入缓冲区（AT 机）

60H　R　键盘数据输出缓冲区（AT 机）

61H	R/W	8042 控制寄存器(AT 机),其中:
		第 7 位=1, 奇偶检查
61H	W	8255 输出寄存器(XT 机),其中:
		第 7 位=1, 清键盘
		第 6 位=0, 保持键盘时钟低
		第 5 位=0, I/O 检查有效
		第 4 位=0, RAM 奇偶检查有效
		第 3 位=0, 读低开关
		第 2 位=0, 保留
		第 1 位=1, 扬声器数据有效
		第 0 位=1, 计时器 2 到扬声器通路有效
62H	R/W	8255 输入寄存器(XT 机),其中:
		第 7 位=1, RAM 奇偶检查
		第 6 位=1, I/O 通道检查
		第 5 位=1, 计时器通道 2 输出
		第 4 位=保留
		第 3 位=1, 系统板 RAM 容量 1 型
		第 2 位=1, 系统板 RAM 容量 0 型
		第 1 位=1, 已装协处理器
		第 0 位=1, POST 循环
64H	R	8042 状态寄存器
		第 7 位=1, 奇偶错
		第 6 位=1, 一般超时
		第 5 位=1, 辅助输出缓冲区已满
		第 4 位=1, 禁止切换
		第 3 位=1, 命令/数据
		第 2 位=1, 系统标志
		第 1 位=1, 输入缓冲区已满
		第 0 位=1, 输出缓冲区已满
64H	W	8042 键盘输入缓冲区
70H	W	CMOS RAM 地址寄存器端口(AT 机),其中:
		第 7 位=1, NMI 无效
		第 6 —— 0 位= CMOS RAM 地址
71H	R/W	CMOS RAM 数据寄存器端口
A0H	R/W	NMI 屏蔽寄存器(XT 机)
A0H	R/W	可编程中断控制器 2
A1H	R/W	可编程中断控制器 2 屏蔽(AT 机),其中:
		第 7 位=0, 保留
		第 6 位=0, 硬盘中断有效

第 5 位＝0，　协处理器例外中断有效
第 4 位＝0，　鼠标中断有效
第 3 位＝0，　保留
第 2 位＝0，　保留
第 1 位＝0，　保留，重定向串联有效
第 0 位＝0，　保留，实时时钟中断有效

附录二　重要数据区

1）文件控制块（FCB）

偏移	长度(BYTES)	意义
−7	1	如果为 FFH 表示扩展的 FCB
−6	5	保留
−1	1	如果为扩展的 FCB，则表示文件属性
00H	1	驱动器号（0=缺省，1=A，2=B …）
01H	8	文件名，不足 8 个则以空格填满
09H	3	文件扩展名，不足 3 个则以空格填满
0CH	2	当前块号
0EH	2	记录长度，默认值为 128
10H	4	文件长度
14H	2	最后一次写文件的日期
16H	2	最后一次写文件的时间
18H	8	保留
20H	1	当前记录号
21H	4	随机记录号

2）程序前缀段（PSP）

偏移	长度	意义
00H	字	INT 20H 终止指令
02H	字	程序最大可用内存
04H	字节	保留
05H	5 字节	INT 21H 远调用入口
0AH	双字	INT 22H 中断地址
0EH	双字	INT 23H 中断地址
12H	双字	INT 24H 中断地址
16H	字	父进程 PSP 段地址
18H	20 字节	文件句柄索引表，FF 为未用句柄
2CH	字	环境段地址
2EH	双字	SS:SP
32H	字	最大文件句柄数
34H	双字	文件句柄表指针
38H	双字	SHARE 文件的 PSP 段地址
3CH	20 字节	保留
50H	字	INT 21H 调用

52H	字节	RETF 远返回指令
53H	9 字节	保留
5CH	16 字节	FCB1
6CH	20 字节	FCB2
80H	字节	命令行参数长度，也是缺省 DTA 的起始地址
81H	127 字节	命令行参数，以控制字符回车（0DH）结束

3）DOS 驱动器参数块（DPB）

偏移	长度	意义
00H	字节	驱动器号（0＝缺省，1＝A，2＝B，……）
01H	字节	设备驱动程序内的单元号
02H	字	每扇区字节数
04H	字节	簇内最大扇区号
05H	字节	簇大小的位移值
06H	字	保留扇区数（引导区）
08H	字节	FAT 表的拷贝份数
09H	字	根目录下允许使用项数的最大值
0BH	字	第一个数据扇区
0DH	字	可能的最大簇号
0FH	字节	每个 FAT 表中的扇区数
10H	字	根目录的第一个扇区
12H	双字	驱动器的设备驱动程序地址
16H	字节	介质描述字节
17H	字节	FFH 表示必须重建 DPB，00H 表示块已访问过
18H	双字	下一设备块的地址，偏移值＝FFFFH 表示该块是最后块
1CH	字	写操作时搜索自由空间的开始簇号
1EH	字	驱动器的空余簇数

4）DOS 内部变量表

偏移	长度	意义
－12	字	公用重试次数
－10	字	公用重试延迟
－8	双字	指向当前磁盘缓冲区
－4	字	DOS 未读完的 CON 输入的代码段内的指针
－2	字	第一个内存控制块（MCB）的段值
00H	双字	指向第一个 DOS 驱动器参数块（DPB）的指针
04H	双字	指向 DOS 文件表清单的指针
08H	双字	指向 CLOCK $ 设备驱动程序的指针
0CH	双字	指向 CON 设备驱动程序的指针
10H	字	各种块设备的每块最大字节数
12H	双字	指向第一个磁盘缓冲区

16H	双字	指向当前目录结构(CDS)
1AH	双字	指向当前 FCB 表
1EH	字	受保护的 FCB 块数目
20H	字节	块设备的数目
21H	字节	CONFIG.SYS 中 LASTDRIVE 命令的值(缺省为 5)
22H	18 字节	NUL 设备驱动程序头
34H	字节	被 JOIN 命令加入的驱动器数目

5) 当前目录结构(CDS)

偏移	长度	意义
00H	67 字节	当前路径的 ASCIIZ 串
43H	字	各种标志
		位 15:如果网络驱动器位和可安装的文件系统位均已设置
		位 14:如果物理驱动器位和非法驱动器位均未设置
		位 13:已使用 JOIN,当前路径为实际路径
		位 12:已使用 SUBST,当前路径为实际路径
45H	双字	指向本驱动器的驱动器参数块(DPB)
49H	双字	当前目录的开始簇
4FH	字	当前路径里根目录的偏移值

6) EXE 文件头

偏移	长度	意义
00H	字节	EXE 文件标识第一部分(4DH)
01H	字节	EXE 文件标识第二部分(5AH)
02H	字	以 512 为模的文件映象长度
04H	字	包括文件头在内的以 512 字节为一页计的文件长度
06H	字	重定位表项数
08H	字	以节(16 字节)计的文件头长度
0AH	字	被装入程序上方所需最小内存节数
0CH	字	被装入程序上方所需最大内存节数
0EH	字	被装入模块中堆栈段相对段址
10H	字	被装入模块入口时的 SP 值
12H	字	文件所有字的负累加和
14H	字	被装入模块入口时的 IP 值
16H	字	被装入模块中代码段相对段址
18H	字	重定位表中第一个重定位项的偏移量
1AH	字节	覆盖号(程序驻留为 0)
1BH 以下		可变保留空间
		重定位表(表的起点由 18H —— 19H 指出)

7) BIOS 通信区

偏移	意义

400H —— 407H	RS－232 适配器地址
408H —— 40FH	打印机地址
410H —— 411H	设备标志
412H	初始化标志
413H —— 414H	内存大小
415H —— 416H	I/O 通道中的内存量
417H —— 43DH	键盘缓冲区
43EH —— 448H	软磁盘数据
449H —— 466H	显示器参数
467H —— 46BH	磁带机数据
46CH —— 470H	时间数据
471H	BREAK 标志。若按下 BREAK 键，此标志置 1
472H —— 473H	重置标志
474H —— 477H	固定磁盘数据区
478H —— 47FH	串、并行口超时计数器
480H —— 483H	额外键盘数据区
484H —— 4ABH	显示器参数区

8) DOS 内存控制块(MCB)

偏移	长度	意义
00H	字节	MCB 标志。如 MCB 为最后一块则为 5AH，否则为 4DH
01H	字	PSP 段的拥有者。0000H 表示自由，0008H 表示属于 DOS
03H	字	以节计的内存块大小
05H	3 字节	保留
08H	8 字节	进程拥有者名

9) CMOS 数据格式

偏移	长度	意义
00H	字节	秒数
01H	字节	闹钟秒数
02H	字节	分数
03H	字节	闹钟分数
04H	字节	小时数
05H	字节	闹钟小时数
06H	字节	星期
07H	字节	日
08H	字节	月
09H	字节	年
0AH	字节	状态寄存器 A
0BH	字节	状态寄存器 B
0CH	字节	状态寄存器 C

0DH	字节	状态寄存器 D
0EH	字节	诊断状态
0FH	字节	停机状态
10H	字节	软盘驱动器类型
11H	字节	保留
12H	字节	硬盘驱动器类型
13H	字节	保留
14H	字节	所安装设备的类型
15H	字	基本内存容量(单位:K)
17H	字	扩充内存容量(单位:K)
19H	字节	硬盘驱动器 0 的类型
1AH	字节	硬盘驱动器 1 的类型
1BH	20 字节	保留
2EH	字	10H 到 20H 的检查和
30H	字	扩展内存容量(单位:K)
32H	字节	BCD 码的世纪值
33H	字节	信息标志

10) 设备驱动程序头

偏移	长度	意义
00H	双字	指向下一驱动程序的指针,如果是最后一个则为 FFFFH
04H	字	设备属性

对于字符设备:

位 15　置为 1
位 14　支持 IOCTL
位 13　一直输出直到支持 BUSY (忙)
位 12　保留
位 11　支持 OPEN/CLOSE/REMMEDIA 调用
位 10—— 7　保留
位 6　支持通用的 IOCTL 调用
位 5　保留
位 4　特殊设备
位 3　设备为 CLOCK $
位 2　设备为 NUL
位 1　设备为标准输出
位 0　设备为标准输入

对于块设备:

位 15　清除
位 14　支持 IOCTL
位 13　非 IBM 格式

位 12　保留
位 11　支持 OPEN/CLOSE/REMMEDIA 调用
位 10　保留
位 9　未知
位 8　未知
位 7　保留
位 6　支持通用的 IOCTL 调用
位 5——2　保留
位 1　驱动程序支持 32 位扇区寻址
位 0　保留

06H	字	设备策略入口点
08H	字	设备中断入口点

对于字符设备：

0AH	8 字节	尾部以空格填满的字符设备名

对于块设备：

0AH	字节	被本驱动程序所支持的子单元数(驱动器数)
0BH	7 字节	未用
12H	字	CD—ROM 驱动程序保留
14H	字节	CD—ROM 驱动程序驱动器字母
15H	字节	CD—ROM 驱动程序单元数
16H	6 字节	CD—ROM 驱动程序信息及版本号

附录三

《电脑步步高丛书》前9册内容提要

从零开始用电脑

黄镭 编著

电脑技术日新月异，以至于许多停留在旧机型和旧版软件水平的电脑入门书籍都显得陈旧和过时了。为适应电脑迅速普及的需要，本书以完全不懂电脑的读者为对象，以当前流行的机型和软件为主线，循序渐进地介绍电脑的软硬件常识，解释常用的术语和常见的英文提示信息，讲解基本的操作方法和简单故障的排除方法，让那怕是从零开始的人也能很快学会使用电脑。

DOS 命令全集与技巧

叶鹰　金玮 编著

本书是针对一般微机用户编写的DOS使用指南，兼有手册性质。内容以DOS5.0为主干，揽括DOS1.X—6.X命令及特点，精简扼要，在DOS机理介绍及系统配置方面独具特色。通过引言、总结和附录，为用户展示了微机及DOS全貌，简明实用。

Microsoft 中文 Windows 入门

罗运模 编著

Windows是一种比MS DOS更优秀的微电脑操作系统，它突破了MS DOS只能使用640 KB内存的限制，可以使用数十亿字节的电脑内存，从而能够极大地发挥出电脑的全部能力。本书内容包括：中文Windows的安装和启动方法；窗口操作；主群组的使用；桌面办公用品的使用；汉字输入、显示和打印；中西混合文本的编辑和练习范例。本书的特点是由浅入深，图文并茂。

UNIX 操作系统入门

黄祥喜 编著

UNIX 是目前国际最流行的多用户操作系统，国际上已制定出以 UNIX 为蓝本的操作系统工业标准。本书以微机上的 UNIX 版本 SCO UNIX 3.2 为实例，以较小的篇幅对 UNIX 的使用做了系统的介绍。内容包括：UNIX 概述；UNIX 的文件和目录操作；UNIX 的命令解释器 SHELL；UNIX 文件系统和磁盘的使用；UNIX 的文件编辑器 vi；UNIX 的进程操作；UNIX 的电子邮件和网络功能；UNIX 的安装、管理和维护。本书不仅系统地介绍 UNIX，而且从 DOS 和 UNIX 比较的角度来叙述，适合广大读者阅读。

NOVELL 网络入门

刘建军 编著

近年来，Novel 的核心部分 NETWare 网络操作系统正快速成为网络工业界的唯一标准。由于 Novel 网络内容丰富，技术复杂，不易学习和掌握，用户都迫切希望有一本内容精练，实用性强的指导性书籍。本书正是针对这一社会需求，结合作者本人使用 Novell 网络的经验编写的。全书的重点放在 NETWare 386 V3.11 版本上。内容包括：局域网络介绍及 Novell 网络的发展概况；Novell 网络操作系统的工作原理；网络的硬件配置和安装；网络的用户分类和上线操作；网络的基本操作命令；网络在 MIS（管理信息系统）中的应用。

常用汉字输入法

余向阳 编著

本书着重介绍了目前广泛流行的十几种计算机汉字输入编码方法，包括区位码、拼音、首尾码、双音码、五笔字型、表形码、大众码、四笔声形、仓吉码、自然码、拼音—汉字变换输入法、肖码、启宏全息码、郑码—字根通用码、太极码。本书是学习汉字输入极有用的资料，可作汉字输入的自学教材、大中专计算机应用课程和计算机培训班的辅助教材，还可作为办公自动化、文字处理、汉字输入编码研究的参考资料。

Auto CAD 使用指南

贺继钢 编著

本书综合有关的资料，并根据作者本人积累的 Auto CAD 使用经验，按初学者的要求，由浅入深地介绍使用 Auto CAD 的方法和技巧。内容包括：Auto CAD 软件的安装、启动和使用

环境；命令格式；常用命令详解；绘图工作环境和图形库的建立；Auto CAD 和数据库及高级语言的通信等。全书叙述的重点放在 Auto CAD 10.0—12.0 的版本上。书中有大量的应用实例，而且这些实例都是按照我国的制图国家标准来绘图和设置系统变量的。

常用图像处理软件

方俊杰 编著

随着电脑的普及和应用水平的提高，人们对丰富多彩的电脑图形兴趣越来越大。本书介绍现时非常流行的 PCX，TIF，GIF 图形格式，以及常用图像处理软件 CSHOW，VPIC，GDS，GWS 的使用方法，从图像的获取、编辑、观看、格式转换、存贮等方面向读者展示一个五彩缤纷的电脑图形世界。

常用动态调试软件

朱彤 编著

调试或跟踪一个程序时，要是有一个功能强大的、且在任何时候都能进入的调试器，那么一切就会得心应手。本书介绍了现在流行的程序动态调试软件：Soft-ICE，MS-DEBUG，SYDE-BUG，FSD，Turbo Debugger 和 Codeview，重点讲述 Soft-ICE 软件的使用方法、技巧和经验，还介绍了常用的反动态跟踪技术。

表形码产品

广州总代理

一、中山大学出版社电脑发展部受中外合资张家港爱文电脑有限公司的委托，成为该公司在广州地区的总代理。

二、总代理除了零售、批发、升级表形码系列软件外，还要建立和扩大代理网点。

三、总代理有权追究非法经营表形码产品的单位和个人。

四、现时经营的表形码软件有：

1）表形码演示练习软件
2）表形码“一日通”5.04 版
3）表形码“一日通”四通 2401 版
4）表形码“一日通”天汇网络版
5）表形码“一日通”万能挂接版
6）表形码中文办公系统

中山大学出版社电脑发展部

地址：广州市新港西路 135 号中山大学校内
邮编：510275
电话：4425565,4446300—7467，1995