FoxPro 下实现对软盘驱动器中软盘状态的检测

陈学中　（山东建材学院）

目前,在微机上利用 FoxBASE 系列数据库管理系统开发的管理信息系统软件应用得比较普遍。这些软件通常是采用多级下拉式菜单驱动,功能完善,用户界面良好。但是,FoxBASE 本身没有提供对软盘驱动器中软盘状态的检测功能,即使 FoxBASE 的升级产品、功能强大的 FoxPro 2.5 版也没有提供对软盘的检测功能,因此,在数据备份和恢复程序模块中,当软盘未准备好时,通常会破坏优美的屏幕画面或造成程序运行的中断。本文采用汇编语言模块实现了对软盘驱动器中软盘状态的检测。

FoxPro 提供了直接调用汇编语言子程序的语句接口。利用这一接口功能,笔者在实际开发信息系统时,编写了汇编语言子程序 TestAB.asm,其基本设计思想是:采用 BIOS 中断服务程序 INT13H 的功能号 0、2、3,对软盘驱动器中软盘进行复位、读、写操作,从而实现对软盘状态的检测。INT13H 有关功能调用的详细说明见表 1 和表 2。

表 1　INT 13H 的子功能调用说明

调用时入口参数	返回时参数及功能
AH＝0	复位磁盘,AH＝盘状态
AH＝2 AL＝扇区数　DL＝磁盘号 CH＝磁道号　DH＝面(0～1) C:＝扇区号 BX＝相对 ES 段的位移	读指定盘、面、道、扇区开始的指定扇区数中的数据到 ES:BX 起始的一片内存单元 AH＝盘状态 CF＝0 表示成功;CF＝1 表示错误
AH＝3 AL、DH、DL、CH、CL、BX 同 AH＝2 时的情形	将从 ES:BX 起始的若干扇区写到指定磁盘的指定位置上。 AH＝盘状态 CF＝0 表示成功; CF＝1 表示错误

在 FoxPro 编程中,对于数据恢复模块,只需要从软盘中读出数据。因此只需利用 INT 13H 的子功能 AH＝2 来检测软磁盘的读状态;对于数据备份模块,则需要利用 INT 13H 的子功能 AH＝3 来检测软磁盘的写状态,由于要将一个扇区的数据写入软盘,这有必要考虑写入的内容对软盘的影响,本文采用的方法是先读出软盘 0 面 0 道 1 扇区的数据,然后再将读出的数据写回软盘 0 面 0 道 1 扇区,即先调用 INT13H 的读功能,然后再调用 INT13H 的写功能。如果软盘状态不正常,磁盘读写操作之后还需用 INT13H 的子功能 AH＝0 复位磁盘。

另外,汇编程序模块还针对参数变量的取值考虑了对非读写功能和非软盘驱动器符号的判断。

表 2　INT 13H 调用返回值(AH)表示的盘状态

AH 值	含义(可能原因)	汇编模块返回值
0	无错	0
1	FDC 命令无效	1
2	地址标记未找到(坏盘片)	2
3	企图写入写保护盘(盘写保护盘)	3
4	目标扇区未找到(坏盘片)	4
6	盘片已更换	
8	DMA 控制器故障(坏磁盘适配器)	8
9	DMA 超越段界	9
10H	CRC 校验错误(坏盘片)	a
20H	FDC 芯片故障(坏的磁盘适配器)	b
40H	SEEK 寻道操作故障(坏的驱动器)	c
80H	设备未响应或超时(未准备好)	d

本汇编模块的调用方法:将上述汇编程序 TestAB.asm 经编译、连接后生成 Testab.exe,再用 exe2bin.exe 文件将其转换成二进制文件 TestAB.bin。

1.利用 FoxPro 的"Load TestAB"将汇编模块从磁盘调入内存。

2.利用"Call TestAB with＜参数变量＞"执行汇编模块,根据返回的参数变量的值,即可检测软盘的状态。

需要特别说明的是,参数变量是两字节长的字符型内存变量。其第一个字节应为"R"(读磁盘操作)或"W"(写磁盘操作),若为其它字符,调用 TesAB 后返回到参数变量的第一个字节将是"Z";参数变量的第二个字节应为"A"(A 盘)或"B"(B 盘),若为其它字符,是返回到参数变量的第一个字节将是"F"。若调用时参数变量的两个字节均不符合规则,则返回到参数变量的第一个字节将是"Z"。若调用时参数变量的两个字节均符合规则,则可以根据返回到参数变量的第一个字节的内容,与表 2 对照,检测软盘的状态。

3.检测完毕,利用"Release Module TestAB"从内存中删除汇编块,释放其所占的内存空间。

程序清单一给出了调用上述汇编模块的实例。

本程序模块同时适用于 dBASEⅢ、FoxBASE+和 FoxPro 系列软件,已经在 5 英寸盘和 3 英寸盘上运行通过,并且在我设计的销售管理子系统、统计管理子系统的数据备份和恢复模块中得到成功应用,效果良好。

*** * 程序清单一 TestAB.prg**

```
set talk off
set stat off
load testab
set color to  w+/g,w+/r
clear
do while.t.
@4,10 say "磁盘操作选择:"
@4,30 prompt "Read(读盘)"
@4,45 prompt "Write(写盘)"
@4,60 prompt "Wrong(错误)"
menu to cc
do case
case cc=1
    ab="R"
case cc=2
    ab="W"
case cc=3
    ab="T"
case cc=0
    exit
endcase
@8,10 say "磁盘选择:"
@8,30 prompt "A"
@8,45 prompt "B"
@8,60 prompt "No-Drive"
menu to cc2
do case
case cc2=1
    ab=ab+"A"
case cc2=2
    ab=ab+"B"
case cc2=3
    ab=ab"T"
case cc2=0
    exit
endcase
@15,25 say"正在测试软磁盘..."
call testab with ab
@15,0
ab=substr(ab,1,1)
if ab< >"0"
do case
case ab="1"
    err_inof="FDC 命令无效!"
case ab="2"
    err_into="未找到 ID 标志!"
case ab="3"
    err_into="企图写入写保护盘!"
case ab="4"
    err_into="目标扇区未找到!"
case ab="8"
    err_into="DMA 控制器故障!"
case ab="9"
    err_into="DMA 超越段界!"
case ab="a"
    err_into="CRC 校验错!"
case ab="b"
    err_into="FDC 芯片故障!"
case ab="c"
    err_into="寻道操作故障!"
case ab="d"
    err_into="设备未响应/超时!"
case ab="x"
    err_into="未知的故障!"
case ab="f"
    err_into="盘符不正确!"
case ab="z"
    err_into="磁盘操作不正确!"
endcase
    @14,20 say"磁盘未准备好!"
    @15,20 say"错误信息:"
    set colo to w+/r
    @15,29 SAY ERR_INFO
else
    @14,20 say"磁盘状态良好!"
endif
set colo to w+/g,w+/r
@16,20 say"按任意键继续..."
iinkey=inkey(0)
@14,0 clear
enddo
release module testab
return
```

程序清单二 TestAB.asm

```
code    segment   byte public
        assume    cs:code,  ds:code
TestAB  proc      far
start:
```

```
        puch    es
        push    ds
        push    ss
        push    bx
        mov     al,[bx]
        push    bx
        cmp     al,'R'
        je      loc_0
        cmp     al,'W'
        je      loc_0
        mov     al,5ah      ;'Z'
        jmp     loc_5
loc_0:
        mov     cs:rw,al
        mov     al,[bx+1]
        cmp     al,'A'
        je      loc-2
        cmp     al,'B'
        je      loc_1
        mov     al,46h      ;'F'
        jmp     loc-5
loc_1'
        mov     dl,1
        jmp     loc-3
loc_2:
        mov     d1,0
loc_3:
        push    es
        pop     es
        mov     ah,2
        mov     al,1
        mov     bx,offset buff
        mov     ch,0
        mov     cl,1
        mov     dh,0
        int     13h
        cmp     ah,0
        jne     loc_4
        cmp     cs:rw,'R'
        je      loc_4
        mov     ah,3
        int     13h
loc_4:
        mov     al,30h      ;'0'
        cmp     ah,0
        je      loc_5
        mov     al,31h      ;'1'
        cmp     ah,1
        je      loc_5
        mov     al,32h      ;'2'
        cmp     ah,2
        je      loc_5
        mov     al,33h      ;'3'
        cmp     ah,3
        je      loc_5
        mov     al,34h      ;'4'
        cmp     ah,4
        je      loc_5
        mov     al,36h      ;'6'
        cmp     ah,6
        je      loc_3
        mov     al,38h      ;'8'
        cmp     ah,8
        je      loc_5
        mov     al,39h      ;'9'
        cmp     ah,9
        je      loc_5
        mov     al,61h      ;'a'
        cmp     ah,10
        je      loc_5
        mov     al,62h      ;'b'
        cmp     ah,20
        je      loc_5
        mov     al,63h      ;'c'
        cmp     ah,40h
        je      loc_5
        mov     al,64h      ;'d'
        cmp     ah,80h
        je      loc_5
        mov     al,78h      ;'x'
loc_5:
        mov     cs:result,al
        mov     ah,0
        int     13H
        mov     al,cs:result
        poop    bx
        mov     [bx],al
        pop     bx
        pop     ss
        pop     ds
        pop     es
        retf
rw      db      ?
result  db      ?
buff    db      512 dup(?)
TestAB  endp
code    ends
        end     start
```