第6卷第1期　　广西科学院学报　　Vol.6, No.1
1990年2月　　Journal of The Guangxi Academy of Sciences　　Feb 1990

BIOS 接口分析与探讨

黄　瑜

（广西计算中心软件一室）

摘　要

所谓BIOS即基本输入输出系统，它包括：键盘输入管理、屏幕和打印机输出管理、磁盘操作管理等功能。没有BIOS，对IBM及其兼容机的应用和开发将是不可想象的。BIOS是一段8086／8088机器语言代码，调用BIOS是利用中断调用，只有汇编语言才能直接调用。由于操作系统提供了许多内部命令和外部命令，供用户对系统的调用，所以我们平常很少关心BIOS。在高级语言里，是以语句形式调用BIOS的，由于任何一种高级语言都只能使用一小部份BIOS的功能，因此，了解和学会应用BIOS是大有裨益的。

本文介绍的是汇编语言和BIOS的接口分析，并介绍高级语言同汇编语言的接口。

一、前言

BIOS（Basic Input／Output System）即基本输入输出系统，它是连接用户与下基本硬件、系统软件的接口。通过它，用户可以透明地控制各类I／O设备，诸如：屏幕显示、磁盘驱动、打印机输出、异步通讯等硬件设施。IBM—PC的BIOS是对用户开放的，本文介绍的就是这一系统的BIOS。

二、BIOS的功能介绍

IBM—PC的系统板装有40K ROM，其中8K为ROM BIOS，32K为ROM解释BASIC解释程序。

当系统加电启动时，驻留在ROM的BIOS提供系统的自检、引导和装入初始化程序运行，并提供用户对主要的I／O设备控制。

BIOS是一段8086／8088机器代码。在系统启动时，即已装入内存。其中地址从00000H～003FFH的内存空间装的是BIOS向量表，BIOS的各功能服务子程序装在向量表指向的相应内存。

BIOS向量表中每个向量占四个字节，其中前两个字节称为偏移址，后两个字节称为段地址，这两个地址就是它所对应的BIOS服务子程序的入口地址。当用户需要调用BIOS的某一部分功能时，需要给出调用的功能向量和有关的入口参数，并由该向量取得服务子程序的入口地址，然后，依据这一地址执行服务子程序。所谓的服务子程序，又称BIOS服务子

程序或中断服务子程序，是BIOS 对硬件和系统软件直接控制的一组程序代码段，各个代码段互相独立，不能互相调用，是系统起动时，一并装入内存（并驻留内存）的机器代码。

调用BIOS 服务子程序的过程称为中断调用或中断，以下所提到的中断和中断调用都是这个意思。

对BIOS功能调用时，所提到的向量为中断向量，每个向量由一个数码表示（从00H～FFH），这个数码就是中断号。所以，进行中断调用时，只要给出中断号和给出入口参数，系统就会执行向量所指向的服务子程序。

各类中断（即BIOS功能的分类）是按中断号划分的。下表就是各类中断的分类表：

中断号（H）	功能	中断号（H）	功能
0	除法错误	1	单步中断
2	非屏蔽中断NMI	3	断点中断
4	溢出中断	5	屏幕打印中断
6	保留	7	保留
8	定时中断	9	键盘中断
A	保留	B	异步通讯2中断
C	异步通讯口1中断	D	硬盘中断
E	软盘中断	F	打印机中断
10	显示I／O中断	11	设备检验调用
12	存贮器检验调用	13	软盘I／O调用

其中：0H～4H为Intel公司规定使用的，8H～1FH为ROM BIOS，20H～27H为DOS使用。

尽管IBM—PC有许多型号的机，以及还有众多的兼容机，可是，它们中的绝大部分软件都能互相使用，这是由于这些机器所配置的ROM BIOS和系统软件的BIOS与外部接口都一样的缘故，也就是说，尽管各种机器里的一些硬件接口地址、信号传输过程互不相同，并且各个服务子程序也不一样。但是，调用这些服务子程序时，其入口参数都一样，调用过程也都一样，这样，在这个基础上编制的软件就能互相通用了，也是这些系统为何叫兼容机的缘故。这些软件都是在此基础上编制的，因此，程序质量也高，简洁，可读性好，兼容度高。应用软件都是只能直接或间接地调用BIOS，自然就更能互相兼容了。

三、BIOS的接口分析

1．中断调用

中断调用的一般形式如下：

```
……                    ；给出入口参数
MOV  AH，功能号
INT  中断号            ；执行中断调用
```

给出入口参数，是指对相应的寄存器赋值，当再给出功能号，执行相应的中断，就完成了外部调用的执行过程，往下的就是系统本身自身执行的了。具体为一类中断需要如何对寄存器赋值，可参考《IBM—PC／XT硬件参考手册》的ROM BIOS清单一节，那里给出了

所有ROM BIOS的详细调用情况说明。

中断调用的执行过程如下：

（1）用户给出入口参数，并执行中断指令。以下均为系统执行；

（2）系统屏蔽一切中断（禁止一切中断调用），然后依据给出的中断号，再乘以4，得到中断向量地址；

（3）由向量地址得到中断服务子程序的入口地址（偏移地址和段地址）；

（4）保护现场，即除AX以外的所有寄存器内容进栈；

（5）执行中断服务子程序；

（6）恢复现场，即恢复所有寄存器内容（除AX以外），取消中断屏蔽，并返回调用过程，往下执行指令。

执行完成以上六个步骤，就完成了一次中断调用。

例如，查表知中断号为5的中断功能是屏幕打印，查BIOS清单知道，这一类中断不需要入口参数和功能号，在操作系统状态下，执行debug↵出现以下提示符后，输入：

```
-A↵
2178:0100  INT  5  ↵        ; 屏幕打印
2178:0102  INT  20↵         ; 退出运行
2178:0104↵
-G↵                         ; 执行
-↵
```

在键入G↵后，在打印机上将输出屏幕信息。将此程序代码以COM文件形式保存起来，设为P.COM，这个文件只占4个字节。在DOS下或dBASE—Ⅲ下，键入P↵或Runp↵都能将当前屏幕内容输出到打印机上。

再如，我们想在屏幕上（200，100）的位置上绘置一个图形彩色点，查表知，10H号中断为屏幕I／O调用，再查BIOS清单，知道绘点的功能号为0CH。现在，编制程序段如下：

```
MOV  DX, 200D    ; 行坐标为200
MOV  CX, 100D    ; 纵坐标为100
MOV  AL, 1       ; 彩色值为1
MOV  AH, 0CH     ; 功能号0CH
INT  10H
```

执行上面的程序段，将在屏幕（200，100）的位置上绘制出一个彩色的图形点。

这一程序段是IBM—PC机上，编制图形软件包的最核心部分（当然，坐标和彩色值是采用参量传送）。

通过以上的两个程序段，我们看到，执行调用中断很简单，设备的控制对我们来说也是透明的。利用好中断调用，可以提高我们的程序质量，更加有效地发挥系统的潜能。

由于BIOS的功能种类很多，每种类型的中断，有些依据功能号划分，可达几十种，例

如中断号为21H的中断，有87种功能划分，每一功能号又由于入口参数不同，而执行不同的功能，并返回不同的结果。总之，BIOS是一个庞大的基本输入输出系统，功能丰富，有待于更多的系统软件的开发者们去开发和利用。由于它们的调用方式和执行过程都离不开上面所讲的形式，也给我们的研究、开发带来了很大的方便。

2. BIOS中断调用的修改

BIOS尽管提供了丰富的功能供应户调用，但是，有时为了实际的需要，如汉字处理、某些硬件的重新配置等等，仍需对BIOS的修改，甚至增加中断服务子程序供自己的系统调用。

原则上，是不能对ROM BIOS和DOS的BIOS修改的。可是，在系统启动后，已将它们装载到内存。这样，我们虽然不能修改ROM BIOS，可是，却可以修改内存里的BIOS服务子程序和修改中断向量表。通过上面的中断类型表，我们也看到了，ROM BIOS还提供一些中断供用户自己编制使用的软中断。以下根据各种情况分别介绍BIOS的更改：

（1）更换中断服务子程序

更换中断服务子程序，就是修改中断向量表相应的向量，使它指向新的中断服务子程序。

在debug下，输入自己编制的中断服务子程序；调试，运行通过后，再让它驻留内存中某一位置（用INT 27H），然后修改向量表中调用这一子程序的向量，使它表示的地址指向新的服务子程序的入口。在编制中断服务子程序时，要严格按照中断服务子程序的格式编写。

（2）修改中断服务子程序

从中断向量表中，找到要修改的服务子程序的入口地址，并利用debug有关命令，进入该内存区，进行修改，调试这一中断服务子程序。

（1）和（2）在修改时，因为都是利用原来的中断号。所以，为了该系统仍能运行其他软件，那么，这一服务子程序的入口参数，返回结果，都应该和原来中断是一样的，否则该机同其他机在这方面就不兼容了。

（3）增加新中断

找出一个中断号，这一中断号尚未被利用于本系统的，并且是ROM BIOS留给用户使用的中断号。输入自己的中断服务子程序，调试，运行通过后，让它驻留在某一可用的内存位置。填写刚才找出的中断号对应的中断向量，即完成新中断的增加。例如，长城0520C—H系统的中断号为10H，功能号为30H～36H，就是该公司自行开发并增加的图形软件包GRD，它所占用的中断号中的功能号正是ROM BIOS保留给用户的中断号。若在CONFIG.SYS的文件中，有这一设置：DEVICE=GRD.SYS，则我们在汇编语言调用这一图形核心系统时，也同调用其他中断一样方便，其形式也是一样的。

总之，无论更换、修改或者增加中断服务子程序，都必须遵循下面的规则：

在一个中断服务子程序中，不能调用另外一个中断服务子程序，也不能进行中断调用；运行服务子程序之前，必须保存除AX以外的所有寄存器的内容；屏蔽一切中断，返回时再打开；用汇编语言和十六进制数码编写程序；中断返回命令用IRET而不是RET。

四、汇编语言与高级语言的链接

以上讨论的是用汇编语言调用BIOS的情况，可是，在更多的情况下，我们是用高级语

言来编制我们的程序。高级语言不能直接使用BIOS的功能，但所有的高级语言都能调用汇编语言子程序。因此，高级语言同BIOS的接口，就变成高级语言与汇编语言的接口了，因为汇编语言是可以直接调用BIOS的。以下以编译BASIC和FORTRAN语言为例，介绍高级语言与汇编语言的接口。

1. BASIC语言与汇编语言的链接

由于解释BASIC语言同汇编语言的链接和调用在《IBM—PC DOS参考手册》有详细介绍，这里仅介绍编译BASIC同汇编语言的接口。

例如，在BASIC中，需要调用屏幕打印中断时，则编制的汇编语言子程序如下：

EXP1.ASM

```
CODE  SEGMENT
      PUBLIC  P
      ASSUME  CS:CODE
   P  PROC  FAR
      PUSH  BP
      INT   5H
      POP   BP
      RET
   P  ENDP
CODE  ENDS
      END
```

BASIC程序调用形式如下：

EXP2.BAS

```
10  DEF  SEG=&H1700
20  CALL  P
30  END
```

以下两个程序分别用MASM和BASCOM编译得EXP1.OBJ和EXP2.OBJ两个目标文件，然后用LINK将这两个文件链接在一起，得到一个可执行文件，完成了链接过程。

通过这两个程序，我们可以看到编译BASIC和汇编语言的一般调用形式，至于调用其他的中断，形式与这两个程序的接口基本一样。如有参数传递，形式与解释BASIC的参数传递完全一样，这里不再详述。

2. FORTRAN语言与汇编语言的链接

下面仅以例子说明它们的调用接口和链接的形式。这一例子有参数传递，又有中断调用，详细地体现了整个接口过程。

例子如下：

FORTRAN语言的会话能力很差，既没有清除屏幕功能，也没有屏幕上定位显示，在实际应用中，总是不尽人意。但FORTRAN语言能调用汇编语言，而汇编语言能调用BIOS。因此，只要编好这两种语言间的接口调用关系，就能弥补FORTRAN语言的这一缺陷了，具体程序如下：

FORTRAN程序：

```
TYPE  EXP-5.FOR
 debug
          program  kkk
          call  cls
          call  pos(5, 30)
          write(*, *) "长城0520C-H"
          end
```

其中call cls为调用汇编语言清屏，call pos(5, 30)调用汇编语言定位光标。

汇编语言的两个程序如下：

C>TYPE POST.ASM

光标定位子程序

```
frame  struc
s___ds  dw    ?
s___bp  dw    ?
r___d    dd    ?
c2      dd      ?
c1      dd      ?
frame  ends
data  segment  public  data
data  ends
dgroup  group  data
          public  pos
code    segment 'code'
         assume  cs : code
         assume  ds : dgroup, ss : dgroup
pos    proc  far
       push  bp
       push  ds
       mov  bp, sp
       les  bx, [bp].c2
       mov  ax, es : [bx]
       and  ax, offh
       mov  dl, al
       les  bx, [bp].c1
       mov  ax, es : [bx]
       and  ax, offh
       mov dh, al
```

```
        mov  ah, 2
        mov  bh, 0
        int  10h
        pop  ds
        pop  bp
        ret  8
pos  endp
code  ends
        end
```

C>TYPE CLEAR.ASM

清屏子程序

```
frame  struc
s___ds  dw        ?
s___bp  dw        ?
r___d    dd        ?
frame  ends
data  segment  public   'data'
data  ends
dgroup  sroup  data
            public  cls
code  segment   'code'
        assume  cs : code
        assume  ds : dgroup, ss : dgroup
cls  proc  far
        push  bp
        push  ds
        mov  ah, 0
        mov  al, 6
        int  10h
        pop  ds
        pop  bp
        ret  0
cls  endp
code  ends
        end
```

用 FORTRAN77 3.30 版本的编译扫描器编译 EXP—5.FOR得EXP—5.obj，再用IBM—PC宏汇编1.00版本编译这两个汇编语言子程序得CLEAR.OBJ和POSI.OBJ文件，在DOS提示符下，键入LINK EXP-5+CLEAR+POSI，得可执行文件EXP—5.EXE，在

DOS下，键入EXP—5，将看到屏幕被清除，并在屏幕上第5行30列上显示“长城0520C—H”字样。

在这两个子程序中，我们看到，它们全用到了10H号中断的调用，并依据功能号不同，分别调用了清屏，光标定位功能。从这里，我们可看出FORTRAN语言间接调用 BIOS 的一般形式。

从FORTRAN语言和BASIC语言调用汇编语言的形式看，它们调用的过程都很简单。关键是，只要熟悉和了解BIOS的功能调用关系，就能弥补这些高级语言的缺陷，使我们的软件质量和水平都有大幅度地提高。

可喜的是，现在有些高级语言，如C语言，Turbo pascal语言，都增添了直接调用BIOS的功能，其调用形式和汇编语言的调用形式一样，但在调用之前，它所做的工作却比汇编语言多得多，有关这方面知识，可参考有关书籍。总之，由于IBM—PC的BIOS体系结构采用了开放式结构，越来越多的系统软件开发者把BIOS的调用，上升到越来越高的层次，应用软件的开发者，将会对BIOS的使用变得更为直观，更为方便，系统变得更透明。

五、结束语

计算机系统结构的不断更新换代，作为基本输入输出系统的BIOS更是首当其冲。了解和掌握BIOS的使用，会使我们的计算机发挥更大潜能。DOS，CP／M等操作系统，都是在BIOS的基础上开发的，由于它们对I／O设备的控制和对系统软件的调用全部采用中断调用。因此，所有IBM—PC都能使用DOS和CP／M。由于中断调用简单，这两个系统软件质量相当高，代码简洁、清晰。了解了BIOS，对我们剖析这两个系统都有很大的帮助，并能适应计算机系统结构的高速发展。

以上的程序均在0520C—H机上调试通过，也是我在实际应用中的一些体会和经验。

参考文献：

《IBM－PC／XT硬件参考手册》
《IBM－PC DOS参考手册》
《8086／8088汇编语言程序设计》

BIOS INTERFACE ANALYSIS AND RESEARCH

Huang Yu

(*Computer Centre of Guangxi*)

ABSTRACT

BIOS is Basic input/output system, it includes keyboard management, screen and printer output management, disk operation management, etc.. It is not to be imagined without BIOS for applying and developing IBM-person Computer System. BIOS is a 8086/8088 machine language code segment. Maybe users don't show interest in BIOS, because users can call BIOS by high-level language statement, but none of high-level languages can provide all BIOS functions. Thus it is necessary for us to understand BIOS.

This paper introduces BIOS, assemble language, high-level language interface analysis.