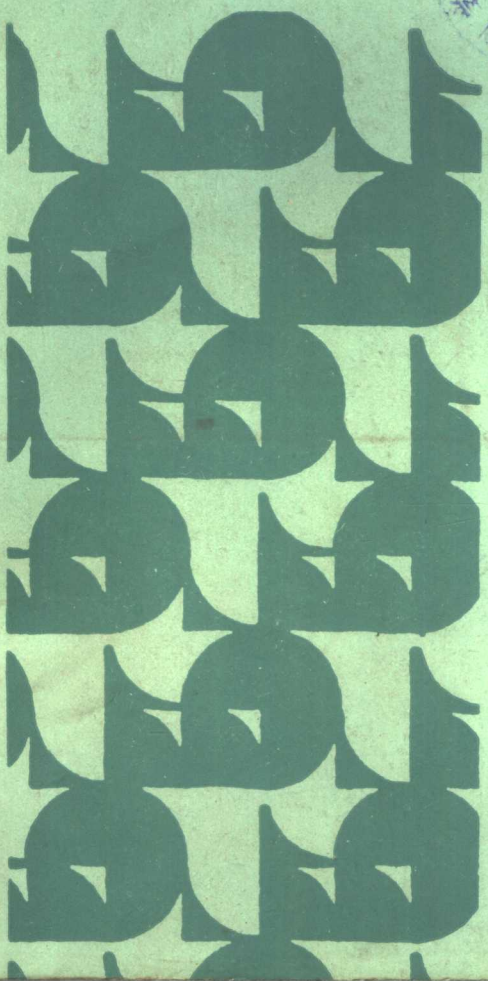
80286 80386 系列

微机ABIOS
技术参考手册

屈建国 译　　黄占武 校阅



陕西电子编辑部

IBM 286 386系列微机ABIOS
技术参考手册

屈建国 译　　黄占武 校

陕西电子编辑部

前　言

个人微型计算机的灵魂是操作系统，一般来说，操作系统分为数层，其中，操作系统的低层软件将系统硬件和操作系统的核心进行逻辑分离。这样作使用户感到硬件是透明的，计算机系统成为一个功能强大使用更方便灵活的虚拟机器。

对 DOS 操作系统来说，内部层次分为：

* DOSBIOS(基本输入输出系统)
* DOS 核心(DOS Kernel，包括 IBMBIO 扩充模块和 IBMDOS)
* 命令处理程序(Shell)

BIOS 为 Basic Input Output System 之缩写。其做为操作系统中最低层次的软件被固化在主机内部，负责和全部标准硬件打交道。不同的计算机系统有其制造商提供的特定的 BIOS，它应包含与硬件相关的下列设备的驱动程序：

* 显示器和键盘驱动程序
* 打印机驱动程序
* 辅助设备驱动程序
* 日期和时间的驱动程序
* 磁盘驱动程序

DOS 核心通过 I/O 请求与这些设备驱动程序通信。这些请求被驱动程序转换为不同硬件控制器的相对应的控制命令。

由此可见，BIOS 在 DOS 操作系统中占有非常重要的位置。要想深入地了解 DOS，必须了解 BIOS 的内部结构。

ABIOS 为增强的 BIOS，(Advanced BIOS)

ABIOS 支持以 INTEL 80286 和 80386 为基础的微型计算机。它突破了 DOS 640 KB 的内存容量局限，达到了 16MB 可寻址的内存空间。用户可以将各种应用程序同时存于内存，使用多任务功能，同时运行多个程序。利用窗口功能监视多项程序的运行，在各个程序之间进行数据转换。

ABIOS 支持 80286 和 80386 的多种运行环境。例如：对于 80286，支持实地址和虚地址两种运行模式。实地址方式可以执行包括 8088/8086 指令集的超级指令集，能够运行 DOS 环境下的所有应用程序。保护地址方式引入了许多大型计算机操作系统的概念，如：虚存空间，进程间通信等。

目前，IBMPC 及其兼容机，PS/2 系列计算机以及以 80286，80386 为 CPU 的各种个人计算机在国内已有了广泛的用户。随着我国计算机技术的普及应用的迅速发展，用户迫切需要一本较详细地介绍 ABIOS 的技术资料。国内有关 BIOS 的资料已有多种，但 ABIOS 的资料市面上尚未见到。

本书是专门详细论述 ABIOS 内部结构的一本技术资料，是在 80286 及 80386 计算机系统上从事应用开发的技术人员必备的技术参考书。

本书译自《IBM Personal System/2™ and Personal ComPuter BIOS Interface Technical Reference》一书。

本书包括如下内容：

* ABIOS 引言　概括介绍了 ABIOS 的数据结构，初始化请求块，转换原则，中断处理以及 ABIOS 扩展。
* 数据结构　包括有关公用数据区，功能转换表，设备块以及如何使用ABIOS的数据结构。
* 初始化
* 转换原则
* 附加信息
* 接口　详尽论述 ABIOS 支持的各种接口操作。为本书的主要内容。

因时间仓促及译者水平所限，错误在所难免，欢迎读者批评指正。

译者　1989.1.5

第一章 ABIOS概论

一、导 言

ABIOS是一个固件。在使用80286和80386微处理器的IBMPS/2系统中，ABIOS将低层的操作系统和硬件隔开。

操作系统产生一个对ABIOS的功能请求，进而由ABIOS去直接控制I/O接口并产生系统硬件的控制字。

本书叙述了ABIOS固件和硬件联接的细节。

ROMBIOS是在单任务操作系统的环境下工作的。它的寻址能力限制在1MB字节范围内，仅在英特尔微处理器的实地址模式下工作。

ABIOS在保护的虚地址模式下工作，支持1MB以上的地址空间，由于ABIOS被固化在ROM中，因而不妨碍RAM工作。

ABIOS可以在实地址模式下工作，也可以在保护模式下工作，或工作于二者都使用的双模环境中。因此ABIOS提供了两种模式都能使用的操作系统的数据结构。另外，ABIOS也能在虚拟的8086模式中工作。

操作系统对ABIOS的功能请求分为三种类型：

1. 单步请求
2. 离散多步请求
3. 连续多步请求

单步请求在调用被返回之前，执行了一个被请求的功能。

离散多步请求启动某一功能或操作，在操作完成之前，这个功能有一段时间的延迟。

连续多步请求启动某一功能或操作，这个操作也包含了一个延迟，但这个延迟永远不会结束。对于多步操作，在延迟时间内由于处理时间被用完而调用返回。一个从I/O设备发出的中断表明某个操作步骤已被完成。

下图表明了ABIOS请求的三种类型。

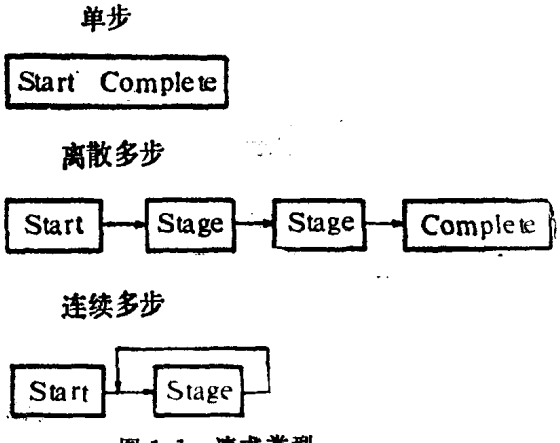


图1-1 请求类型

二、数据结构

一个操作系统所产生的对 ABIOS 的请求是需要一个转换规则的。这个转换规则由 ABIOS 结构提供。这个规则需用一个数据结构来建立，用以联接操作系统和其所支持设备的功能子程序。数据结构包含公共数据区，功能转换表和设备块。它们常驻于系统内存中，在 ABIOS 初始化时，数据结构被初始化。

由 ABIOS 所提供的转换规则决定了在实地址模式中和保护模式中英特尔微处理器中的操作。为了保证执行实地址模式和保护模式的灵活性，在公用数据区中将所有的 ABIOS 指针全部链结到同一结构之中。这个结构中包含了功能转换表指针、设备块指针和 ABIOS 数据指针。

ABIOS 入口指针存贮于功能转换表的向量表中。每个 ABIOS 所支持的设备都有一个被链结的功能转换表。功能转换表中的头三个入口点分别为启动程序、中断程序和超时处理程序的入口点。

对于每种 ABIOS 所支持的设备，其 ABIOS 功能的子程序都需要一个常设工作区。这个工作区叫做设备块。设备块中存贮着硬件接口地址、中断级别、及设备状态等信息。

三、初始化

ABIOS 和操作系统确定了初始化的约定。操作系统在初始化过程中以及在整个启动过程中起着主导作用。在操作系统完成初始化过程之前，ABIOS 是不能使用的。初始化过程是在实地址模式下进行的。它包括三个阶段：

1. 操作系统调用BIOS建立系统参数表。这张表描述了系统中可用设备的数目，ABIOS 公用入口指针以及对系统栈的需求。

2. 操作系统调用 BIOS 建立初始化表。这些表定义了每个系统所支持的设备信息，这些信息用于初始化设备块和功能转换表。

3. 使用第二步所涉及到的初始化信息，操作系统为公用数据区分配内存空间。同时，操作系统为设备块和功能转换表分配内存空间。公用数据区中的设备块指针和功能转换表指针也被初始化。接着，操作系统调用 ABIOS 建立每个设备的设备块和功能转换表。

初始化过程如下图所示。

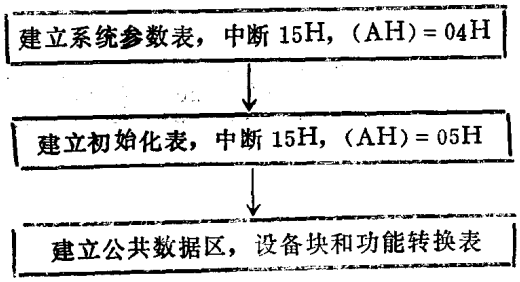


图 1-2　初始化过程

四、转换规则

在ABIOS实施初始化过程以后，通过请求块的参数块提出请求。请求块中包含如下内容：

请求的操作，请求的细节，涉及到的数据转换表的内存定位，请求完成或某个阶段的状态。关于请求块将在第四章转换规则中详细阐述。

ABIOS的执行使用调用返回的编程模式，或使用ABIOS的转换规则，或使用操作系统的转换规则。这两种调用规则容许操作系统在调用ABIOS中有较大的灵活性。

ABIOS转换规则是操作系统最简单的调用系列。操作系统通过公用数据区指针和请求块指针指向下面三个公用入口点中的一个：公用启动子程序、公用中断子程序和公用超时处理子程序。在初始化时，这三个公用子程序的指针被返回至操作系统。这些公用子程序使用请求块信息和公用数据区指针用以获得设备块指针和功能转换表指针。公用子程序把控制权转给被请求的ABIOS子程序，它们的指针在功能转换表中。ABIOS的转换流向如图1-3所示。

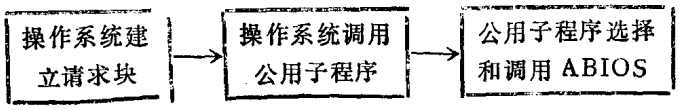


图1-3 ABIOS转换流向

操作系统转换规则需要由操作系统确定被调用的ABIOS子程序的地址。这就允许操作系统在保存被频繁调用的ABIOS子程序地址时具有灵活性，这种方法在处理从字符或可编程的I/O设备发出的中断中是有效的。在调用ABIOS子程序时，公用数据区、请求块、功能转换表和设备块四个指针是必须知道的。操作系统转换规则的流程如图1-4所示。

ABIOS转换规则和操作系统在第四节转换规则中详细论述。

图1-4 操作系统转换规则流程

五、中断处理

对于多步请求，微处理器得知从某一硬设备发出中断后，便去查看该中断在中断向量表中予先置好的地址。当某一中断发生时，ABIOS激发操作系统接受控制。ABIOS提供为处理ABIOS中断所需的中断子程序。中断处理过程将在中断处理一节中详细论述。

六、扩展ABIOS

为了支持某台新的硬设备和系统中添加的某些设备，在ABIOS中需要提供添加、贴补、扩展和取代的能力。这方面详尽的内容请看第五章附加信息。

第二章 数据结构

一、导 言

ABIOS 使用数据结构的目的在于将操作系统链结到每个 ABIOS 所支持设 备的功能子程序上。这些数据结构分别叫做：公用数据区，功能转换表和设备块。这些数据结构常驻在系统内存中，当 ABIOS 被初始化时，数据结构也被初始化。

ABIOS 使用的转换规则定义了所允许的全部操作。这些操作是用于英特尔微处理 器的实地址模式和/或保护地址模式之下的。而 ABIOS 提供了执行实地址模式、保护地址模式或双模环境操作的公用数据区。

在公用数据区中包含着功能转换表指针、设备块指针、和 ABIOS 数据指针。单一结构的公用数据区把全部 ABIOS 指针链结到操作系统中。而操作系统在英特尔微处理器的两种操作环境下管理着 ABIOS 请求。

二、公 用 数 据 区

公用数据区包含了 ABIOS 在双模环境中进行操作所需要的数据指针。在系统初始化时，数据指针被予置好。数据指针中包含了系统所支持的每一台设备的信息。公用数据区在所有操作模式中都是需要的：包括实地址模式、保护地址模式和双模环境。

每一个 ABIOS 设备有一个物理设备的标识符，叫做设备 ID。每一个物理 设 备 ID 都

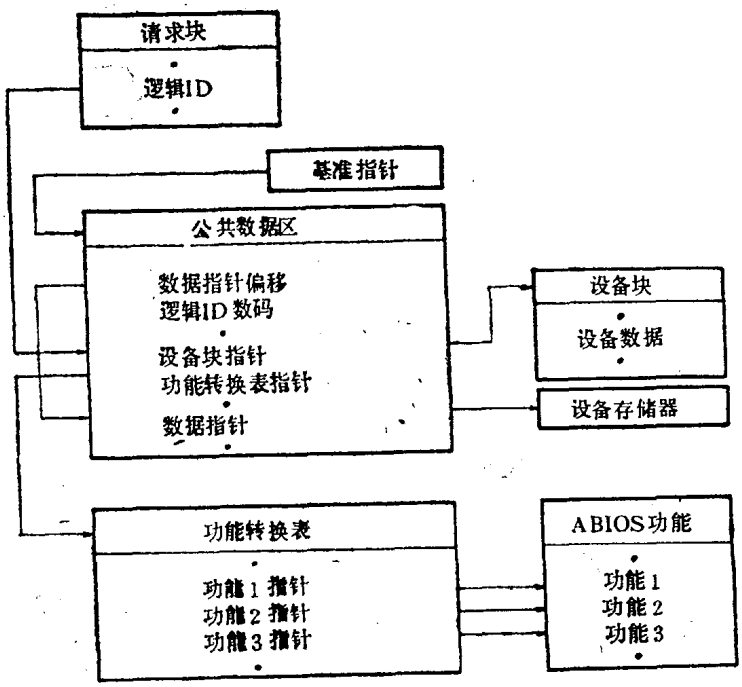


图 2-1 公用数据区流向

有一个或多个逻辑 ID 。逻辑 ID 用于系统产生一个设备请求时做设备处理用 。公用数据区由两个阵列组成：一个阵列是逻辑 ID 入口点，另一个是数据指针入口点。每一个逻辑 ID入口点包含设备块指针和功能转换表指针。每个数据指针入口点包含 ABIOS 服务程序的内存地址。

在每个对 ABIOS 的请求中，一个地址(段或选择器号，偏移量为 0)被传送到 ABIOS 中。这个地址是指向公用数据区的，并被用做公用数据区的基准指针。

图 2-1 表明了公用数据区和 ABIOS 的这种关系。

下表详细列出了公用数据区的结构

表 2-1　公用数据区

域	偏移量	长度
对数据指针 0 的偏移	+00H	2
逻辑 ID 计数	+02H	2
保留	+04H	4
设备块指针逻辑 ID 1	+08H	4
功能转换表指针逻辑 ID 1	+0CH	4
设备块指针逻辑 ID 2	+10H	4
功能转换表指针逻辑 ID 2	+14H	4
⋮	⋮	⋮
设备块指针逻辑 ID n	+(08H · n)	4
功能转换表指针逻辑 ID n	+(08H · n)+04H	4
数据指针P 长	+(08H · n)+08H	2
〃　P 偏移	+(08H · n)+0AH	2
〃　P 段	+(08H · n)+0CH	2
〃　P－1 长	+(08H · n)+0EH	2
〃　P－1 偏移	+(08H · n)+10H	2
〃　P－1 段	+(08H · n)+12H	2
⋮	⋮	⋮
数据指针 0 长	+(08H · n)+(06H · P)+08H	2
〃　0偏 移	+(08H · n)+(06H · P)+0AH	2
〃　0 段	+(08H · n)+(06H · P)+0CH	2
数据指针计数	+(08H · n)+(06H · P)+0EH	2

n-逻辑 ID 之数码

P-数据指针－1 的数码

公用数据区各个入口说明：

Offset to Data Pointer 0:

这个域值和基准指针(首址)组合起来形成数据指针所在的地址。

Count of Logical IDS:

这个域值包含了设备块的数字和功能转换表指针。

Device Block Pointers:

这个域值包含了由给定的逻辑 ID 所确定的设备块指针。

Function Transfer Table Pointer:

这个域值包含了由给定的逻辑 ID 所确定的功能转换表指针的信息。

Data Pointer Lengths:

这个域值包含了由数据指针所指明的数据区的长度。

Data Pointer Offsets:

这个域值包含着偏移量值。每个偏移量和数据指针结合起来，产生数据区指针。

Data Pointer Segments:

这个域值包含着段地址。每个段地址和数据区的偏移量结合起来则产生数据区指针。

Data Pointer Count:

这个域值包含着数据区指针的数目。

如果在初始化以后，功能转换表和设备块指针二者都为 0:0，不管其逻辑 ID 是什么，操作系统都将有一个空的公用数据区入口。当 ABIOS 扩展初始化时，这个入口都被当做一个暂时的地址使用。

三、功能转换表

ABIOS 的入口点存贮在一个向量表中，此向量表称为功能转换表。这些表中包含着每个 ABIOS 功能子程序的双字地址指针，暂不用保留的功能双字指针在初始化时被置为 0:0，每一个逻辑 ID(公用数据区入口)都有一个功能转换表指针。多个逻辑 ID 可以共有一个共同的功能转换表指针，指向同样的功能转换表。

图 2-2 表明了功能转换表和公用数据区的关系。

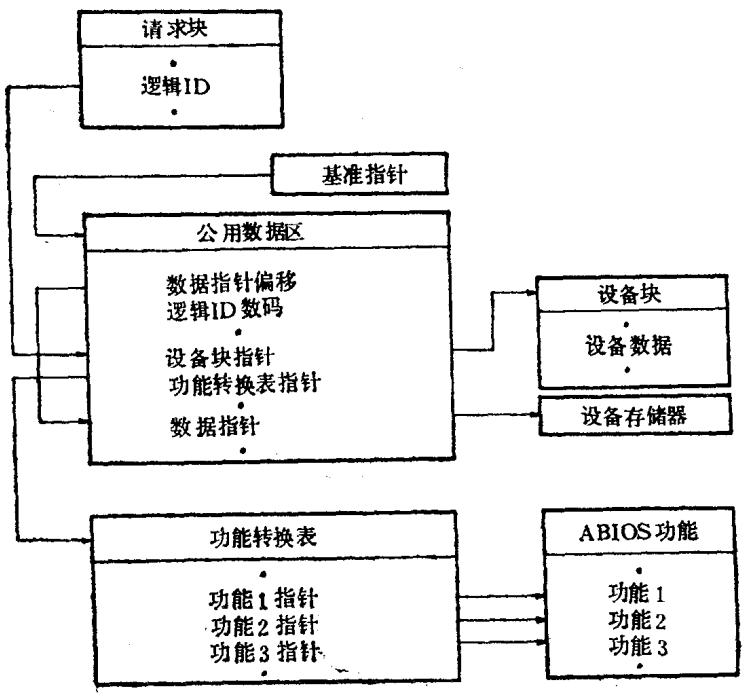


图 2-2 功能转换表流程

操作系统建立一个请求块。块中包括逻辑 ID(由该设备定义)和所请求的功能。由这些信息，就可以定位该请求设备所对应的功能转换表和设备块指针。操作系统使用功能转换表指针启动请求，处理中断，处理所发生的超时。功能转换表中每个指针都是双字，用以指向功能子程序。表 2-2 为功能转换表。

表 2-2 功能转换表

功 能	偏移量	长度
启动子程序指针	+00	4
中断子程序指针	+04	4
超时子程序指针	+08	4
功 能 计 数	+0C	2
保 留	+0E	2
功能 1 子程序指针	+10	4
〃 2 〃 〃	+14	4
⋮	⋮	⋮
功能 n 子程序指针 n 是功能数	+0C+(4·n)	4

功能转换表中各入口含义如下：

Start Routine Pointer：

启动子程序指针为双字指针。它被调用以启动一个请求，该调用属于远程间接调用。由这个功能子程序去确认功能域，请求块长度域和单元域。在调用这个子程序时，全部寄存器信息被保存起来。

Interrupt Routine Pointer：

中断子程序指针为双字指针，为远程间接调用，用以去恢复一个多步请求。如果一个多步请求的操作没有结束，所有多步请求都要由启动这个功能子程序而重新开始，如果功能转换表所对应的设备不产生中断，则中断子程序指针被初始化为 0。

Time out Routine Pointer：

超时子程序指针是一个双字指针，为远程间接调用。若某一硬件中断在规定时间内未被接收，则用该功能子程序来终止请求。这个子程序终止请求，和硬件控制板脱离，并重新回到初始化状态。所有寄存器信息在此功能子程序调用过程中被保护起来。

在功能转换表中相应的设备不产生中断，或设备有中断产生但永远不会超时，超时功能子程序指针域被初始化为 0:0。

Function Count：

一个字，用以记录由某设备所支持的功能的数字。

Reserved：

保留字，一旦功能计数域值为 0，则被分配。

Function 1 Pointer：

双字指针，指向功能 1 子程序。

Function 2 Pointer：

双字指针，指向功能 2 子程序。

Function n Pointer:

双字指针，指向功能 n 子程序。

四、设 备 块

对于每个设备，ABIOS 都需要一个固定的工作区，这个工作区叫做设备块。硬件接口地址，中断级别，设备状态等各种信息存贮在设备块中。

设备块中包含公用及私用两种数据。设备块中的公用信息是以全部设备块通用格式填写的一块可读的信息区，这个公用信息不由操作系统改变。设备块中的私有数据由 ABIOS 内部使用，它的内容和格式不尽相同。操作系统不改变私有数据，而改变设备块中私有数据的权力归 IBM 公司所有。

图 2-3 表明了设备块和公用数据区的关系。

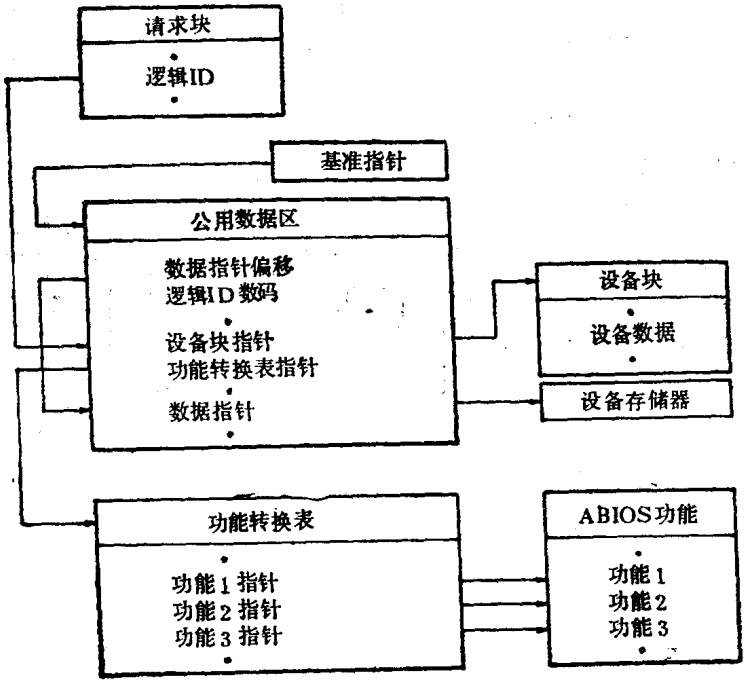


图 2-3 设备块流程

每个 ABIOS 所支持的设备都有对应的设备块和其相连。

表 2-3 设 备 块

域	偏移量	长度	存取
设备块长度	+00H	2	公用读
版本	+02H	1	〃

辅助设备 ID	+03H	1	公用读
逻辑 ID	+04H	2	〃
设备 ID	+06H	2	〃
逻辑 ID 专用接口对计数	+08H	2	〃
〃 〃 通用 〃 〃	+0AH	2	〃
逻辑 ID 专用口对 0	?	4	
〃 〃 1	?	4	〃
⋮ ⋮			〃
逻辑 ID 专用口对 n	?	4	〃
逻辑 ID 通用口对 0	?	4	〃
〃 〃 1	?	4	〃
⋮ ⋮			
逻辑 ID 通用口对 n	?	4	〃
设备唯一数据区长度	?	2	私 有
设备唯一数据区	?	?	〃
单元计数	?	2	〃
单元唯一数据区长度	?	2	〃
单元唯一数据区	?	?	〃

? ——变量值的定位记录

n——接口对计数

设备块入口各项含义是：

Device Block Length：

这个域为一个字，表示设备块的字节数，而这个字节数包含自身。设备块最大长度为 64 KB-1。当 ABIOS 初始化时，一个具体的设备所需的设备块字节被予置在此域中。

Revision：

这个字节用于记录某一设备支持码的等级，最低等级值为 0 。对某一具体设备来说，ABIOS 代码每一次成功的版本升级或更新，此域值递增 1 。即对现存硬件的 ABIOS 支持代码版本升级，此域值递增 1 。

Secondary Device ID：

这个字节用于确定 ABIOS 所支持的硬件级别，最低初始值为 0 。如果以前所定义的设备 ID 支持了新的硬件而把支持该设备代码发展到一个新水平时，此域值递增 1 。当此域值递增 1 时，Revision 域值反转到 0 值。

Logical ID：

逻辑 ID 记录了和设备块相联系的逻辑名。它和由 BIOS 所使用的存取不同类型设备的软中断数字是类似的。逻辑 ID 值在初始化时动态确定，对于给定设备的逻辑 ID 值由公用数据区的入口索引确定。

为了便于对公用内部 ABIOS 功能进行贴补，操作系统需要修正逻辑 ID 值，以使得 ABIOS 能调用公用内部的 ABIOS 功能。在初始化时，它们由操作系统识别 。

逻辑 ID 值如表 2-4 所示。

表 2-4 逻辑 ID 值

用 法	逻辑 ID
保留	00H
保留	01H
ABIOS 内部调用	02H 到 n
系统和适配设备	>n

n：ABIOS 内部调用的最后一个逻辑 ID

Device ID：设备 ID 记录了一个功能请求可寻址的设备的型号所支持的 ABIOS 功能级别。这个域赋值如表 2-5 所示。

表 2-5 设备 ID 值

设备	设备 ID 值	设备	设备 ID 值
ABIOS 内部调用	00H	非屏蔽中断	0AH
软盘	01H	指向设备	0BH
硬盘	02H	保留	0CH
CRT	03H	保留	0DH
键盘	04H	非易失存储器(NVRAM)	0EH
并口	05H	直接存储器存取(DMA)	0FH
异步通讯口	06H	可编程选件选择(POS)	10H
系统定时器	07H	保留	11-15H
实时时钟定时器	08H	键盘安全	16H
系统伺服	09H	保留	17H-FFFFH

设备 ID 域中 16 进制 00H 保留做为 ABIOS 内部调用使用。所谓 ABIOS 内部调用，即一个 ABIOS 功能可调用另一个 ABIOS 功能。而全部其它的设备 ID 值代表着某一种类型的设备。

Count of Logical ID Exclusive Port Pairs：

这是一个逻辑 ID 专用端口对的数目。逻辑 ID 专用端口是由特殊的逻辑 ID 所使用的。诸如：软盘接口，硬盘接口，异步通讯接口，并行接口，以及 CRT 接口。如果此域值为0，则表示没有给此域分配空间。

Count of Logical ID Common Port Pairs：

这是一个逻辑 ID 通用端口对的数目。逻辑 ID 通用端口是一个以上的逻辑 ID 所共享的端口。诸如：DMA 控制器端口，键盘控制器端口和 NVRAM 端口。使用了这些端口中某一个的逻辑 ID 值在设备块中的逻辑 ID 通用端口对域中有一个入口。如果此域为 0，则表示没有为本域分配空间。

Logical ID Exclusive Port Pairs：

为逻辑 ID 专用端口对。此域长为双字，第一个字表示为一个 I/O 端口系列中启始的 I/O 端口数字，第二个字表示 I/O 端口系列中末尾的端口数字。

Logical ID Common Port Pairs：

为逻辑 ID 通用端口对。此域长为双字，第一个字为 I/O 端口序列中启始的 I/O 端口数字，第二个字为 I/O 端口序列中末尾 I/O 端口数字。

注意：当一个 ABIOS 所支持的逻辑 ID 从某一端口读或写时，此端口既包含在逻辑 ID 专用端口域中，也包含在逻辑 ID 通用端口域中。

Device Unigue Data Area length:

此域包含着一个长度值，以字节计。此长度为这个设备的设备唯一数据区的长度。

Device Unigue Data Area:

此域为某一设备专用唯一数据所保留。为生成设备 ID 所用的描述此设备的参数和工作数据保存在此域中。这个域中包含着 ABIOS 用的私用数据，而它的内容和格式可以变化。保存在这个域中的典型数据诸如：中断级别，优先级别和设备状态。

Count of Unit:

这个域中包含着在设备块中单元唯一数据区的数目。如果此域值为 0，则单元域的数字为设备块的上一个域。

Unit Unigue Data Area Length:

这个域包含一个以字节计的长度值。此长度值为包含单元唯一数据区长度域在内的可重复的单元唯一数据区的长度。只有当单元域值大于 0 时，这个域值才存在。

Unit Unigue Data Area:

这个域是为每个设备 ID 单元所保留的私有可重复的区域。例如，如果软盘 ID 值为 16 进制 01 H，则专用软盘驱动器有一个单元。描述单元的参数及判别单个请求优先级的工作数据保持在这个区域中。这个区域中的数据为 ABIOS 所私有，它的内容和格式可以改变。只有当单元域值大于 0 时，这个域才存在。

第三章　初始化

一、导　言

ABIOS 是以一种"钻石式样"被初始化的。操作系统对 BIOS 和 ABIOS 实施专门的调用来完成启动。实地址模式的公用数据区在任何对 ABIOS 的请求发生之前就应完成初始化。微处理器实地址模式中必须实施初始化，其中包括建立系统参数表，初始化表和公用数据区。图 3-1 表明了实地址模式中公用数据区初始化过程的流程。

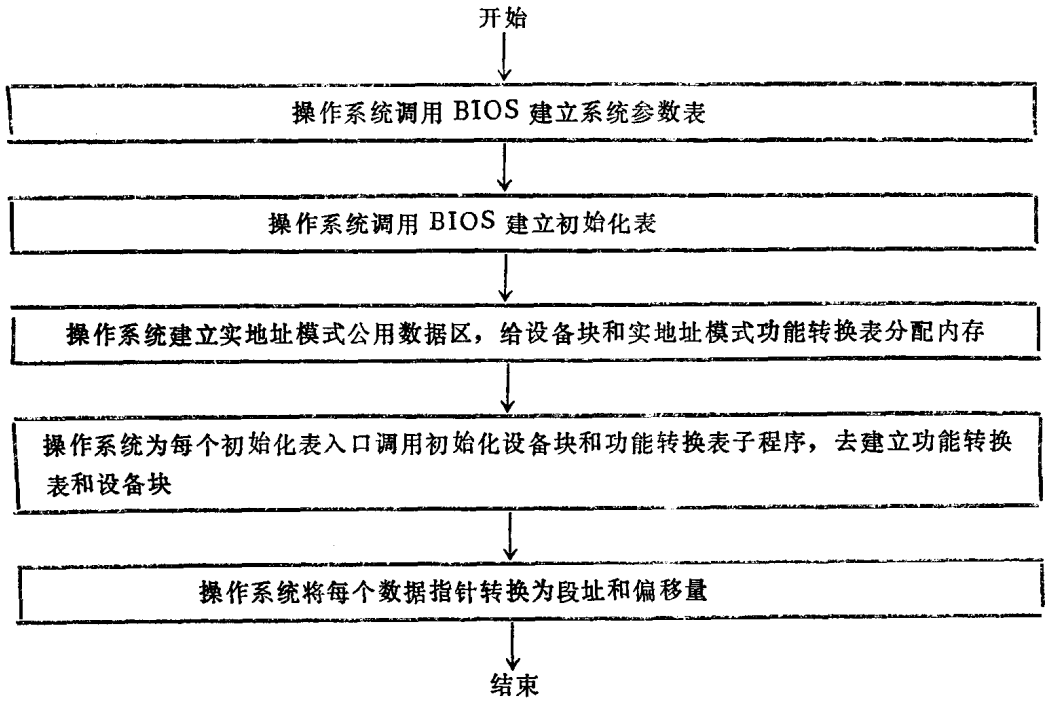


图 3-1　实地址模式公用数据区初始化流程

二、建立系统参数表——操作系统

操作系统分配 16 进制 20 个字节的区域来建立系统参数表。这张表描述了系统中可用设备的数目，ABIOS 公用的入口点及所需的系统栈。

INT15H，(AH) = 04H 建立系统参数表

调用：软件中断，操作系统调用 BIOS。

(ES：DI) = 指向系统参数表内存的地址。

(DS) = 段地址，偏移量为 0，指向扩展 RAM 区，若无内存扩展，长度为 0。

返回：(CY)=1　记录一个异常的错误。

(AH)=0　无错误。

(除 AX 和标志寄存器之外，其它全部寄存器被保存起来。)

三、建立系统参数表——BIOS

当操作系统调用 BIOS 时，就由 BISO 来建立系统参数表。入口点的地址由系统板的 ROM，接口板的 ROM 和 RAM 扩展板建立。为了累计入口点的数目，系统配置信息从洲统配置数据区，NVRAM 和对设备的测试中获得。测试代码常驻于系统板 ROM 中。

对于带有适配器 ROM 代码的设备，由系统板 ROM 中的加电自检程序扫描确定适配器 ROM 所需入口点的数目。

对于 RAM 扩展区中带有代码的设备，由对 RAM 扩展区的扫描确定 RAM 扩展区所需的初始化表所需入口的数目。

一旦获得系统参数表信息，内存按此表重新分配，并由操作系统重新使用。系统参数表如表 3-1 所示。

表 3-1　系统参数表

域	偏移量	长度
公用启始子程序指针	+00H	4
公用中断子程序指针	+04H	4
公用超时子程序指针	+08H	4
栈请求	+0CH	2
保留	+0EH	4
保留	+12H	4
保留	+16H	4
保留	+1AH	4
入口数目	+1EH	2

系统参数表中各项定义如下：

Common Start Routine Pointer：

为双字长度的地址指针，指向公用的启动子程序入口点。

Common Interrupt Routine Pointer：

双字长度的地址指针，指向公用中断子程序的入口点。

Common Time-out Routine Pointer：

双字长度的地址指针，指向公用超时处理子程序入口点。

Stack Required：

域长为一个字，包含以字节计的栈尺寸。ABIOS 工作时使用此栈。

Number of Entries：

域长为一个字，包括了初始化表所需的入口数目。