有的设备操作非常类似，如 RAM Disk 的 CLOSE 和空设备的 CLOSE，可以共享同一段函数代码，基于此种目的，我们为每个函数设计了缺省实现，供具体的设备调用。

（3） 各个函数的作用：

- dr_name：返回设备的名称，缺省实现中返回一个空字符串。
- dr_open：视设备的不同，dr_open 执行的功能可以是安装设备、初始化设备、打开设备或者验证设备是否可用，当以上过程发生错误时返回一条错误信息。
- dr_close：释放设备。
- dr_ioctl：检测和改变设备的分区、切换工作模式等工作。
- dr_schedule：该函数入口为不同的设备提供不同的扩展功能，例如：硬盘驱动程序中，用于计算柱面号、扇区号和磁头号等参数。部分设备没有该函数的实现，但是这是一个公用的设备驱动程序入口，要提供所有支持的外设操作的并集。
- dr_prepare：初始化硬件 I/O 参数。
- dr_rdwt：执行硬件 I/O。
- dr_geometry：返回驱动器的物理特性。如硬盘的磁头数、扇区数等。对于有的设备没有这些物理特性，如 RAM Disk，为了接口的统一性以及其他程序处理方便，我们返回一个伪物理参数。

3.2 典型块设备驱动程序的设计与实现

3.2.1 磁盘

传统意义上的磁盘，包含硬盘和软盘两种。两者在很多方面都非常相似，但是由于软盘具有比硬盘更简单的控制器和移动介质两大特点，使得软盘驱动程序的实现比硬盘复杂。为了突出我们的设计与实现的重点，避免纠缠过多的硬件特征，在本文以后的部分，特将硬盘作为描述对象，除非特殊说明，所提到的磁盘均是指硬盘。

对于每一类总线的系统板都有不同类型的 I/O 适配器，系统板为这些适配器

配备了基本的 I/O 系统（BIOS），它是一种 ROM 中的固件，其目的是提供一种在操作系统和特殊硬件之间建立一个桥梁，有些比较特殊的外设自己还提供了扩展 BIOS。现实情况是：早期的 BIOS 是为 MS-DOS 系统设计的，运行于 16 位实模式，不支持多道程序设计。操作系统实现面临的困难是：是否使用 BIOS 支持设备驱动。

有几种不同的方法可以解决这个问题：

a) 为需要支持的每种硬盘控制器重新编译一个操作系统版本。

b) 在系统中包含几个不同的硬盘驱动程序，有内核在启动时自动决定使用哪一个。

c) 在系统中包含几个不同的硬盘驱动程序，提供一种方法使用户决定使用哪一个。

针对嵌入式系统的特殊要求，我们采用了第一种方法，因为对于像嵌入式这种使用特定配置的系统，没有必要为从来不使用的设备驱动程序浪费系统资源和内存空间。从长远来看，这种方法是最好的。

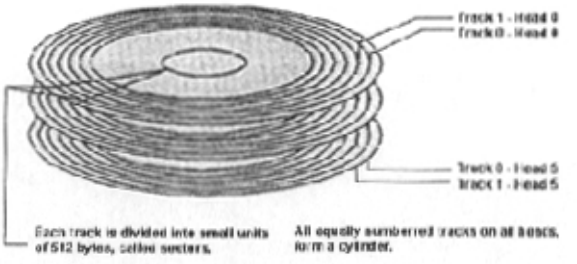
3.2.1.1 磁盘的物理结构

磁盘(Disk)由一组同轴的圆柱面(Cylinder)构成，圆柱面包含一个或多个柱面号相同的磁道(Track)，一个磁道沿圆周又划分为若干扇区(Sector)，每个扇区内可存放一个固定长度的数据块 (如图 3-1)。每道的扇区数和每柱面的磁道数不是固定的，而是由介质类型决定。同样，每个扇区的大小也不相同，一般情况下是 512 个字节。

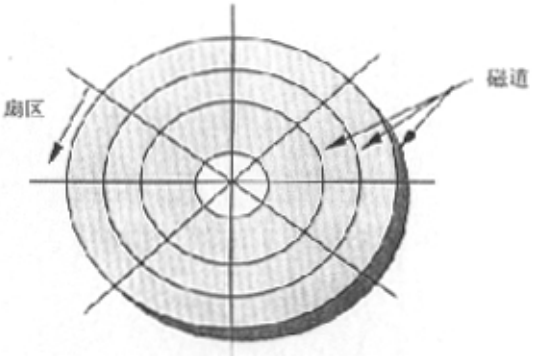
传统的硬盘访问方式称为 CHS 方式，指在获取磁盘数据时要送给硬盘控制器（HDC）的参数包括柱面号（C）、磁头号（H）、扇区号（S）。在图 3-1（a）中，我们给出了一个有 6 个面的磁盘的例子以及 C、H、S 三个参数在物理位置上的关系。图 3-1（b）中给出的是磁道和扇区在物理上的关系。

现代硬盘普遍采用 LBA（线性块访问）的访问方式。这种方式下将大量复杂的控制功能交给了 HDC，要访问硬盘时只需给出被访问磁盘扇区在所有磁盘扇区中的线性编号，不再采用 C、H、S 三参数来定位被访问的扇区，从而使得硬盘驱动程序的编写相对简单。

设备驱动程序通过扇区号来管理扇区，每个扇区都对应一个扇区号，其值是从 0 到总扇区数减 1 的整数，除了正在进行格式化外，系统不对柱面和磁道进行管理，并且假定磁盘上扇区的排列是无序的(这些工作由磁盘驱动程序来完成)，这样做不仅使系统能够兼容各种具有不固定扇区数目、不固定磁道数目的磁盘，并且可以根据物理设备的不同自由选择采用 CHS 或 LBA 方式管理硬盘。



(a)



(b)

图 3-1 磁盘物理结构

3.2.1.2 磁盘逻辑结构

设备驱动程序通过扇区号来访问磁盘，再由文件系统把磁盘的物理格式转化为用户所能理解的逻辑格式。也就是将磁盘的不同部分存放不同的信息，由此提供一种目录和文件的结构，系统通过这种结构来存取数据。 我们在设计中将磁盘组织成 MS-DOS4.0 或更低版本兼容的格式，这是一种业界的标准磁盘格式。磁盘的逻辑格式如(图 3-2)所示：

在 MS-DOS4.0 以前的版本限制了单个磁盘分区只能是等于或小于 33 兆，因此在早期版本的 DOS 下格式化磁盘时要进行多个分区，每个分区等于或小于 33 兆。本系统的设计不受此限制，我们采用 FAT16 分区格式，它能正确识别和使用的最大分区大小是 2G。

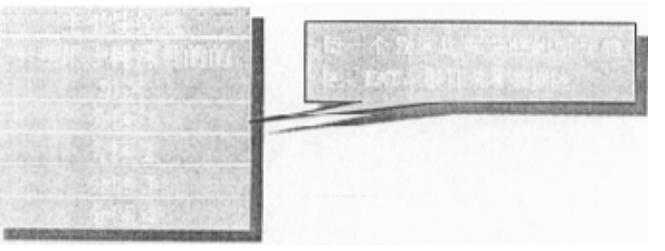


图 3-2 磁盘逻辑格式图

磁盘逻辑结构对于各种文件系统都是相同的，区别在于分区中的信息不同。逻辑结构一般由主引导记录、隐藏扇区和 1~4 个分区组成，各部分含义如下：

- 主引导纪录　它位于磁盘的第一个扇区。在主引导区里存储了分区表的信息，它最多可以存放四个主分区信息。分区表就存放在主引导区偏移量为 0xBE 到 0xFD 之间。这个分区表里可以存放四个分区信息，下表是主引导区存放的四个分区信息的内容举例，从操作系统代号中可以看出，第一行和第二行是 DOS 系统的 FAT16 分区，第三行和第四行是两个空闲分区（未使用的分区表项）；从扇区标志可知，第一行为主分区，第二行为扩展分区：

扇区标志	分区开始扇区			操作系统代号	分区结束扇区			分区开始扇区相对扇区号	分区实际扇区数
	磁头号	扇区号	柱面号		磁头号	扇区号	柱面号		
80	01	01	00	06	3F	BF	05	0x3F	0x1FDE41
00	00	81	06	06	3F	BF	6C	0x1FDE80	0x65640
00	00	00	00	00	00	00	00	0x0	0x0
00	00	00	00	00	00	00	00	0x0	0x0

表 3-1 分区表的结构

- 分区　除了主引导纪录以及紧随其后的少数隐藏扇区以外，其余部分是一个或多个分区，这些分区可以是某种特定的文件系统(比如最常见的 MS-DOS 的 FAT)。 在表 3-1 中我们展示了一个带有扩展分区的典型 FAT 的分区表。下面是就分区表的第一项给出的详细解释：

  扇区标志：0x80(主引导扇区标志)

扇区开始磁头号：0x01

扇区开始扇区号：0x01

扇区开始柱面：0x00

操作系统代号：0x06（DOS 系统的 FAT16 分区——关于各种操作系统及其在分区表中的对应代号在附录一中有详细的说明）。

扇区结束磁头号：0x3F

扇区结束扇区号：0x3F

扇区结束柱面：0x205

（说明：

1. 注意这里的数据和上面表中所给出的不同：标志扇区开始和结束各使用了 3 个字节，第一个字节表示磁头号，第二个字节的后六位表示扇区号，第二个字节的前两位和第 3 个字节一起表示柱面号。

2. 这里明显有一些限制，如：每磁道的扇区不能超过 64 个等，但这些限制对现代硬盘已经不适用了，这也是采用 LBA 方式的原因之一。）

分区相对扇区号：0x3F，表示该分区从硬盘的第 63 个扇区开始。

分区实际扇区数：0x1FDE41，表示该分区大小为 1019M。

在我们的系统中提供了基于 FAT 系统的读取分区表、创建和使用分区、定位和装载分区的系统调用。不过在创建分区表方面并没有提供强大的功能，因为一般用户只有在使用已经分了区的磁盘或想和低版本的 MS-DOS 兼容时才会关心分区的情况。

- **隐藏扇区**　在主引导区和第一个分区之间还存在着一些空闲的扇区，通常情况下这些扇区被用来做为主引导区的备份。通常情况下，0 柱面 0 磁道的第 1 个扇区是 MBR，0 柱面 0 磁道的第 2 个扇区是 MBR 备份。这种技术在 FAT 文件系统中被广泛采用，FAT 表和虚拟分区表在很多情况下也有备份，主要是便于系统程序和第 3 方应用程序检测和维护文件系统的一致性。
- 分区标志　在主引导扇区的最后是包含分区表的扇区的标志：0x55AA。

3.2.1.2 硬盘控制器—IDE 协议

硬盘控制器采用 IDE(integrated disk electronics)接口，也叫 AT 总线接口，是

当前硬盘驱动器普遍采用的一种接口。IDE 可处理容量范围高达 8G 以上，数据传输速率最高可达到 13MB/s，IDE 硬盘采用高性能的旋转音圈电机，配以嵌入式的扇区伺服装置，定位精确、快速，耗电省、抗震性能优越，断电后磁头能自动锁定在启停区。图 3-3 列出了 IDE 的基本功能模块，从图中可以看出 IDE 控制器是怎样嵌入外设单元的。IDE 总线适配器的左边只需要少量的译码器和总线驱动器部件。在这里 IDE 驱动器更像个元件，而不是一个外设。

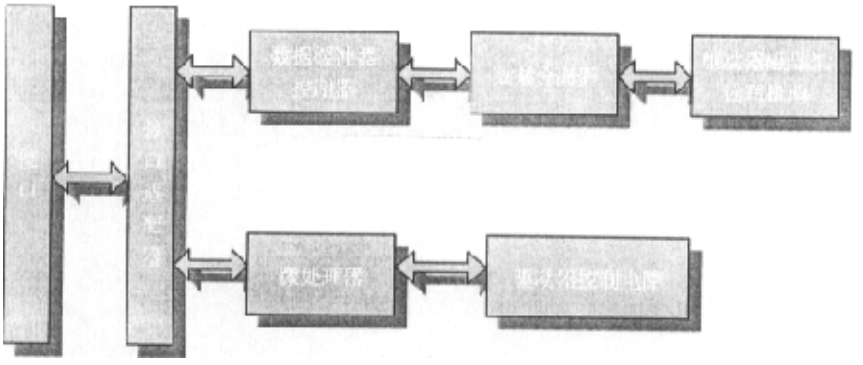


图 3-3　IDE 控制器结构框图

IDE 控制器中有两组寄存器：命令寄存器和控制寄存器。命令寄存器被用来接收命令和传送数据；控制寄存器组用作磁盘控制。命令寄存器组的地址范围是 1F0H~1F7H，控制寄存器组的地址范围是 3F0H~3F7H。目前，许多计算机配置了两个 IDE 接口，对于第二个 IDE 接口，这两组寄存器的地址范围分别是 170H~177H 和 370H~377H。

在 IDE 标准中以寄存器方式传送数据，命令和地址。这些寄存器包括数据寄存器，错误状态/特性寄存器，起始扇区地址寄存器，柱面地址寄存器，状态寄存器/命令寄存器等。除数据寄存器为 16 位以外，其他寄存器均为 8 位。通常 IDE 适配器采用 IRQ14 中断。

IDE 是相当复杂的一种设备,命令寄存器的每一位都有相应的功能,而各个位不同的组合构成了硬盘能执行的上百种命令。在我们的驱动程序源码中有数百行用于定义这些参数和它们的返回状态。这里只简单的描述各个寄存器的简单作用，详细的协议描述可以参看参考文献。

1F0H　数据寄存器（R/W）

1F1H　错误寄存器（R）

1F2H　扇区数寄存器（R/W）

1F3H　扇区号寄存器（R/W）；

1F4H/1F5H 柱面号寄存器（R/W）

1F6H　驱动器/磁头寄存器（R/W）

1F7H　状态寄存器（R）：它反映硬盘驱动器执行命令后的状态，若直接读该寄存器，则要求清除中断请求信号，如果要避免清除中断，可以读辅助状态寄存器 3F6H，这两个寄存器的内容完全相同。

3F6H　硬盘控制寄存器（W）

1F7H　命令寄存器（W）

3.2.1.3 磁盘驱动程序实现

磁盘驱动程序是一系列函数所构成的一个函数库，它提供了我们在§3.1.2 中提到的所有函数的实现部分。在该程序库中有以下模块：

（1）驱动程序安装函数

该模块不涉及任何硬件，只是给硬件分配中断、I/O 地址等系统资源，同时给驱动程序分配一个数据结构。这段代码首先是检查中断号、DMA 号以及其它硬件参数是否有效；然后在检查已经安装的设备驱动程序表中有无与该驱动程序重复的项（防止该函数重复执行），然后进入临界区（将设备驱动程序表作为临界资源），执行驱动程序的安装过程。

安装驱动程序时，首先在设备驱动程序表中找一个空闲项，初始化该项的设备指针、设备驱动程序块指针、在系统中所占用的中断号（位于中断控制器 8259 上）以及中断服务程序入口地址。同时创建该表项的访问控制信号量和中断服务程序信号量。

安装完成后，退出临界区。

（2）硬件设备安装函数

安装硬件设备的过程之所以要和安装驱动程序的过程分开，是因为可能有多个设备使用同一个驱动程序。

在对预设的驱动器号和工作模式执行简单的检查以后，就可以开始安装设备了。在安装设备的过程中，将会用从硬盘控制器得到的数据重新设定硬盘的工作

模式，这样既可以有效避免初始化参数设置错误对硬盘造成损坏，又可以使程序不依赖 BIOS 运行。唯一不理想的地方是：有一些老式的硬盘控制器不支持该项功能。

以上工作都完成以后，程序会发出一个硬盘复位指令，目的是在检验硬盘初始化是否正确，以及检验硬盘是否能正确工作。

（3）磁盘数据读写函数

对磁盘硬件的读和写是两个不同的过程，但是由于它们的操作过程很相似，所以我们将它们放在了一个函数中，用不同的参数区别不同的操作，传递给函数的参数是一个结构，其中包括了要读写的设备号、扇区数、起始扇区位置以及缓冲区地址等。

读写之前当然要对设备作必要的检查，然后调用真正执行读写操作的函数。

在进行读写操作时，主要有两方面的问题要考虑：第一，驱动器是否支持 DMA。一般说来，在 80486 以前的设备由于 CPU 的处理速度相对较慢，大多数采取这种方式，而现在的磁盘控制器均采取轮询或中断方式；第二，一次读取的扇区数。在硬盘控制器的参数中有一项是一次中断能传送的最大扇区数，它的含义是在多扇区传输模式下，每一次中断发生时，硬盘最多能提供多少个扇区的数据。

以上两个问题在程序中解决妥当以后，剩下的问题就是向硬盘控制器发送正确的参数以控制硬盘正确工作，然后启动命令，等待中断，从硬盘接收数据等一系列简单的工作了。

这里还要提到的是出错处理。硬盘大多数时候是工作在正常状态下，但是不排除出现传输错误、坏磁道或者文件损坏的情况。如果碰见有错误发生，一般情况下，会让驱动器重复操作一定次数，然后在错误仍然存在的情况下向调用者返回一个错误。

（4）其它操作

在上面没有描述的模块主要有：设备的卸载、设备驱动程序的卸载以及获取驱动器的物理参数等。相对于上面描述的各个模块，这些模块的实现相对简单，在这里不再详述。