制定位置中去。

有了这两个宏以后，可以说 DOC 的大部分问题已经解决了，剩下的问题就是一些概念上的东西了，比如，向什么地址写入它的控制字、如何分段、读写等待延时等特性，这些是极其繁琐和枯燥的过程。针对 DOC，有数十行的此类定义，我们将它们全部放在了一个叫做 doc2000.h 的头文件中。

3.2.3 RAMDisk

3.2.3.1 RAM 盘的硬件与设计思想

RAM 盘的设计思想很简单。块设备是具备两个操作命令的存储介质：写数据块和读数据块。通常这些数据块存储与旋转存储设备上，例如硬盘和软盘。RAM 盘则简单的多，它使用预先分配的主存来存储数据。RAM 盘具有快速存取的优点（没有寻道和旋转延时），适用于存储需要频繁存取的程序和数据。

图 3-6 给出了 RAM 盘的实现思想。根据为 RAM 盘分配内存的大小，RAM 盘被分为 n 块，每块的大小和实际磁盘块的大小相同。当它的驱动程序收到一条读写一个数据块的请求时，它只计算被请求的块在 RAM 盘存储区的位置，并读出或写入该块，而不读写软盘或硬盘。数据的传输通过调用一个汇编语言过程来实现，该过程以硬件所能实现的最高速度把数据拷贝到用户程序或者从用户程序拷出。

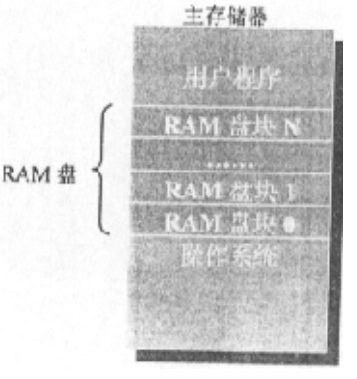

图 3-6 RAM 盘的实现方式

一个 RAM 盘驱动程序可以支持将存储器中若干区域当作 RAM 盘来使用，

每个 RAM 盘用次设备号来区分。一般情况下这些区域相互分开，但是在某些情况下，也不排除相互重叠的可能性。

3.2.3.2 RAM 盘的软件

和我们的惯例一样，RAM 盘的驱动程序也需要实现前面设备驱动程序接口中提到的函数。我们将这些实现都放在 ramdisk.c 中，而与之相关的定义在 ramdisk.h 中。ramdisk.c 包含了对 RAM 盘操作的四个函数入口，它们是：name（返回一个设备名称）、read、write、geometry（返回硬件参数）。可以看到对于 RAM 盘来说，不仅函数实现少，而且实现的这几个函数基本上都属于没有什么操作的例程，这是因为 RAM 盘的操作并不复杂，其它函数是多余的。

在实现细节上要说明的一点是，缺省情况下，系统不自动建立 RAM 盘（这是因为嵌入式系统的 RAM 资源非常宝贵）。配置 RAM 盘的大小的工作放在系统配置文件中，要求用户提供一个最大和最小的 RAM 盘空间，系统会根据可以使用的内存数量寻找一块足够大的空间来作为 RAM 盘，如果能使用的内存不满足用户给出的最小值，则返回一个错误。这样实现的好处是：不仅能使用户的配置简化，而且不用重新编译内核就可以给出不同大小的 RAM 空间。

对于 RAM 盘的读写操作，简单到可以用 ANSI C 库函数 memcpy 来实现。而最后一个函数 geometry，是在一旦 RAM 盘被询问到有关磁道、柱面、扇区等物理特性的时候，用于模拟提供这些参数的一个函数，它的实现是一系列的赋值语句，然后 return。

3.2.4 空设备

空设备实际上是 RAM 盘的一种，它的功能是接受输入数据并把数据抛弃掉的一种设备。一般情况下没有太大的用处，但在使用 shell 程序的时候有特殊的意义，例如程序执行的结果不再需要了，或者不想让程序的结果输出到标准输出设备上（一般是显示器）时，就可以这样使用

```
cat readme.txt > NULL
```

RAM 盘的驱动程序对它采取一种高效的处理方式，将它的长度设置为 0，这