```
FUNCTION PBDelete (paramBlock: ParmBlkPtr; async: BOOLEAN) :
          OSErr;
```

Trap macro _Delete

Parameter block

→	12	ioCompletion	pointer
←	16	ioResult	word
→	18	ioNamePtr	pointer
→	22	ioVRefNum	word
→	26	ioFVersNum	byte

PBDelete removes the closed file having the name ioNamePtr and the version number ioFVersNum, from the volume specified by ioVRefNum.

Note: This function will delete *both* forks of the file.

Result codes		
	noErr	No error
	bdNamErr	Bad file name
	extFSErr	External file system
	fBsyErr	File busy
	fLckdErr	File locked
	fnfErr	File not found
	nsvErr	No such volume
	ioErr	I/O error
	vLckdErr	Software volume lock
	wPrErr	Hardware volume lock

DATA ORGANIZATION ON VOLUMES

This section explains how information is organized on volumes. Most of the information is accessible only through assembly language, but some advanced Pascal programmers may be interested.

The File Manager communicates with device drivers that read and write data via block-level requests to devices containing Macintosh-initialized volumes. (Macintosh-initialized volumes are volumes initialized by the Disk Initialization Package.) The actual type of volume and device is unimportant to the File Manager; the only requirements are that the volume was initialized by the Disk Initialization Package and that the device driver is able to communicate via block-level requests.

The 3 1/2-inch built-in and optional external drives are accessed via the Disk Driver. If you want to use the File Manager to access files on Macintosh-initialized volumes on other types of devices, you must write a device driver that can read and write data via block-level requests to the device on which the volume will be mounted. If you want to access files on volumes not initialized by the Macintosh, you must write your own external file system (see the section "Using an External File System").

The information on all block-formatted volumes is organized in **logical blocks** and allocation blocks. Logical blocks contain a number of bytes of standard information (512 bytes on Macintosh-initialized volumes), and an additional number of bytes of information specific to the

Disk Driver (12 bytes on Macintosh-initialized volumes; for details, see chapter 7). Allocation blocks are composed of any integral number of logical blocks, and are simply a means of grouping logical blocks together in more convenient parcels.

The remainder of this section applies only to Macintosh-initialized volumes; the information may be different in future versions of Macintosh system software. Other volumes must be accessed via an external file system, and the information on them must be organized by an external initializing program.

A Macintosh-initialized volume contains **system startup information** in logical blocks 0 and 1 (see Figure 6) that's read in at system startup. This information consists of certain configurable system parameters, such as the capacity of the event queue, the initial size of the system heap, and the number of open files allowed. The development system you're using may include a utility program for modifying the system startup blocks on a volume.

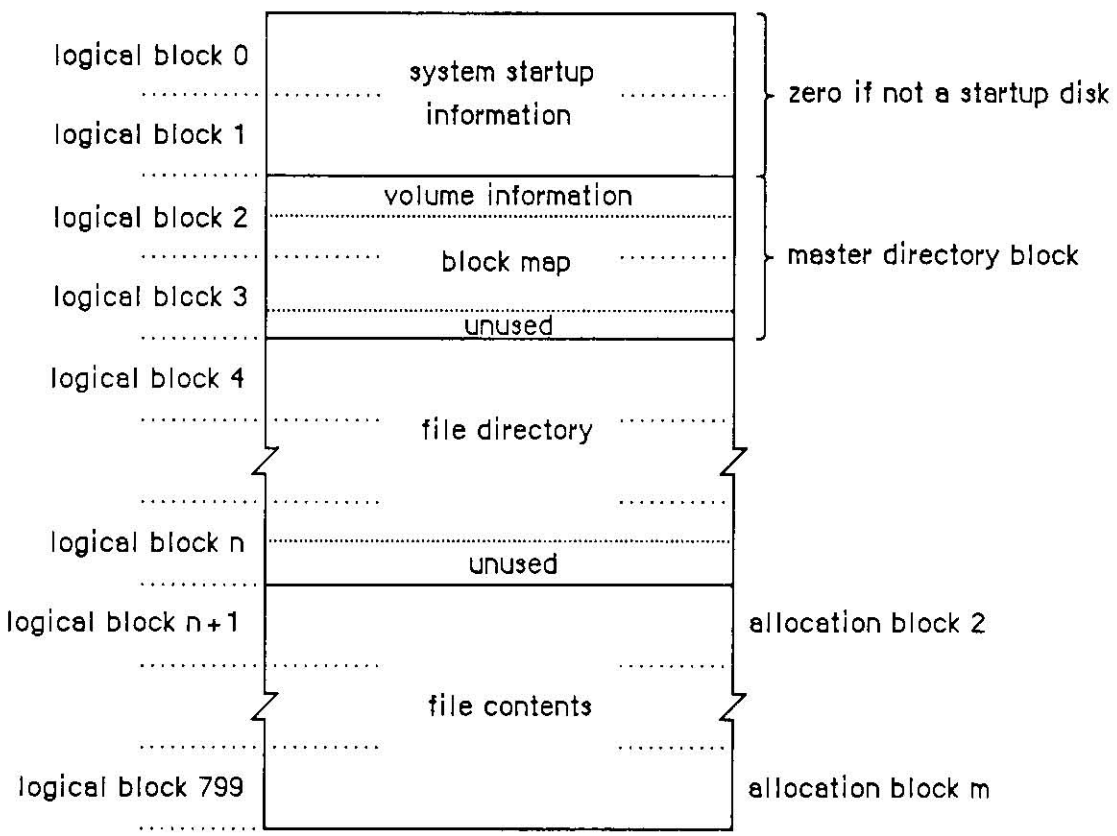

Figure 6. A 400K-Byte Volume with 1K-Byte Allocation Blocks

Logical block 2 of the volume begins the **master directory block**. The master directory block contains **volume information** and the **volume allocation block map**, which records whether each block on the volume is unused or what part of a file it contains data from.

The master directory "block" always occupies two blocks—the Disk Initialization Package varies the allocation block size as necessary to achieve this constraint.

In the next logical block following the block map begins the file directory, which contains descriptions and locations of all the files on the volume. The rest of the logical blocks on the

volume contain files or garbage (such as parts of deleted files). The exact format of the volume information, volume allocation block map, and file directory is explained in the following sections.

Volume Information

The volume information is contained in the first 64 bytes of the master directory block (see Figure 7). This information is written on the volume when it's initialized, and modified thereafter by the File Manager.

byte 0	drSigWord (word)	always $D2D7
2	drCrDate (long word)	date and time of initialization
6	drLsBkUp (long word)	date and time of last backup
10	drAtrb (word)	volume attributes
12	drNmFls (word)	number of files in directory
14	drDirSt (word)	first block of directory
16	drBlLen (word)	length of directory in blocks
18	drNmAlBlks (word)	number of allocation blocks on volume
20	drAlBlkSiz (long word)	size of allocation blocks
24	drClpSiz (long word)	number of bytes to allocate
28	drAlBlSt (word)	first allocation block in block map
30	drNxtFNum (long word)	next unused file number
34	drFreeBks (word)	number of unused allocation blocks
36	drVN (byte)	length of volume name
37	drVN+1 (bytes)	characters of volume name

Figure 7. Volume Information

DrAtrb contains the **volume attributes**, as follows:

Bit	Meaning
7	Set if volume is locked by hardware
15	Set if volume is locked by software

DrClpSiz contains the minimum number of bytes to allocate each time the Allocate function is called, to minimize fragmentation of files; it's always a multiple of the allocation block size. DrNxtFNum contains the next unused file number (see the "File Directory" section below for an explanation of file numbers).

> **Warning:** The format of the volume information may be different in future versions of Macintosh system software.

Volume Allocation Block Map

The volume allocation block map represents every allocation block on the volume with a 12-bit entry indicating whether the block is unused or allocated to a file. It begins in the master directory block at the byte following the volume information, and continues for as many logical blocks as needed.

The first entry in the block map is for block number 2; the block map doesn't contain entries for the system startup blocks. Each entry specifies whether the block is unused, whether it's the last block in the file, or which allocation block is next in the file:

Entry	Meaning
0	Block is unused
1	Block is the last block of the file
2 to 4095	Number of next block in the file

For instance, assume that there's one file on the volume, stored in allocation blocks 8, 11, 12, and 17; the first 16 entries of the block map would read

```
0 0 0 0 0 0 11 0 0 12 17 0 0 0 0 1
```

The first allocation block on a volume typically follows the file directory. It's numbered 2 because of the special meaning of numbers 0 and 1.

> **Note:** As explained below, it's possible to begin the allocation blocks immediately following the master directory block and place the file directory somewhere within the allocation blocks. In this case, the allocation blocks occupied by the file directory must be marked with $FFF's in the allocation block map.

File Directory

The file directory contains an entry for each file. Each entry lists information about one file on the volume, including its name and location. Each file is listed by its own unique **file number**, which the File Manager uses to distinguish it from other files on the volume.

A file directory entry contains 51 bytes plus one byte for each character in the file name. If the file names average 20 characters, a directory can hold seven file entries per logical block. Entries are always an integral number of words and don't cross logical block boundaries. The length of a file directory depends on the maximum number of files the volume can contain; for example, on a 400K-byte volume the file directory occupies 12 logical blocks.

The file directory conventionally follows the block map and precedes the allocation blocks, but a volume-initializing program could actually place the file directory anywhere within the allocation blocks as long as the blocks occupied by the file directory are marked with $FFF's in the block map.

The format of a file directory entry is shown in Figure 8.

byte 0	flFlags (byte)	bit 7=1 if entry used; bit 0=1 if file locked
1	flTyp (byte)	version number
2	flUsrWds (16 bytes)	information used by the Finder
18	flFlNum (long word)	file number
22	flStBlk (word)	first allocation block of data fork
24	flLgLen (long word)	logical end-of-file of data fork
28	flPyLen (long word)	physical end-of-file of data fork
32	flRStBlk (word)	first allocation block of resource fork
34	flRLgLen (long word)	logical end-of-file of resource fork
38	flRPyLen (long word)	physical end-of-file of resource fork
42	flCrDat (long word)	date and time of creation
46	flMdDat (long word)	date and time of last modification
50	flNam (byte)	length of file name
51	flNam+1 (bytes)	characters of file name

Figure 8. A File Directory Entry

FlStBlk and flRStBlk are 0 if the data or resource fork doesn't exist. FlCrDat and flMdDat are given in seconds since midnight, January 1, 1904.

Each time a new file is created, an entry for the new file is placed in the file directory. Each time a file is deleted, its entry in the file directory is cleared, and all blocks used by that file on the volume are released.

Warning: The format of the file directory may be different in future versions of Macintosh system software.