PBWrite Write data to a device driver

#include <Devices.h> **Device Manager**

OSErr	**PBWrite(***pb*, *async* **);**	
ParmBlkPtr	*pb* ;	address of a 50-byte IOParam structure
Boolean	*async* ;	0=await completion; 1=immediate return
	returns	Error Code; 0=no error

PBWrite is used by the **File Manager** and the **Device Manager**.

File Manager Usage

PBWrite writes a specified number of bytes to an open file (data or resource fork). It updates the file mark, in preparation for the next sequential write.

pb is the address of a 50-byte IOParam structure. The following fields are relevant to the File Manager:

Out-In	Name	Type	Size	Offset	Description
–>	ioCompletion	ProcPtr	4	12	Completion routine address (if *async* =TRUE)
–>	ioRefNum	short	2	24	File reference number
–>	ioBuffer	Ptr	4	32	Address of start of data to write
–>	ioReqCount	long	4	36	Number of bytes to write
–>	ioPosMode	short	2	44	Positioning Mode (1=absolute, 2=from EOF, et.al)
–>	ioPosOffset	long	4	46	Entry: Position delta (bytes from start,EOF,...) Return: none (error in IM, see Tech Note #187)
<-	ioResult	OSErr	2	16	Error Code (0=no error, 1=not done yet)
<-	ioActCount	long	4	40	Actual number of bytes written

async is a Boolean value. Use FALSE for normal (synchronous) operation or TRUE to enqueue the request and resume control immediately. See Async I/O.

File Mgr Returns: an operating system Error Code. It will be one of:

noErr	(0)	No error
dskFulErr	(-34)	Disk full
fnOpnErr	(-38)	File not open
ioErr	(-36)	I/O error
paramErr	(-50)	Negative value in ioReqCount
posErr	(-40)	Attempt to position before start of file
rfNumErr	(-51)	Bad ioRefNum
vLckdErr	(-46)	Volume is locked
wPrErr	(-44)	Diskette is write-protected
wrPermErr	(-61)	Read/write permission does not allow write

File Manager Notes:

PBWrite transfers ioReqCount bytes starting at ioBuffer into an open file, starting at the file position specified by ioPosMode and ioPosOffset. Upon completion, ioActCount contains the actual number of bytes written.

Note that the data is not necessarily written to disk until the file buffer and/or volume buffer is flushed. See **PBFlushFile** and **PBFlushVol**.

The following example creates a file and writes a string of text to it.

Example

```
#include <Files.h>
#include <string.h>

IOParam      pb;
short        rc;
char         theBuf[]="This message is written";

pb.ioNamePtr = (StringPtr)"\pHardDisk:Ltrs:1988:Smith";
pb.ioVRefNum = 0;                          /* use values in the name */
pb.ioVersNum = 0;                          /* always best to use this */
rc=PBCreate( &pb, FALSE );
if ( rc ) { /* . . . handle the error . . .*/ }

/* Note: we'd normally call PBSetFInfo here to set fdType='TEXT' */

pb.ioPermssn = fsWrPerm;               /* read/write permission */
rc=PBOpen( &pb, FALSE );
if ( rc ) { /* . . . handle the error . . .*/ }

pb.ioBuffer = theBuf;
pb.ioReqCount = strlen(theBuf);
pb.ioPosMode = fsFromStart;
pb.ioPosOffset = 0;
rc=PBWrite( &pb, FALSE );              /* write the line to disk */
if ( rc ) { /* . . . handle the error . . .*/ }
PBClose( &pb, FALSE );                 /* close the file */
```

Device Manager Usage

PBWrite writes a specified number of bytes (ioReqCount) from your buffer (ioBuffer) to a device driver specified by ioRefNum. The drive number is listed in ioVRefNum. IOPosOffset reveals the device driver's position. IOActCount is the number of bytes actually read.

pb is the address of a 50-byte IOParam structure. The following fields are relevant to the **Device Manager**:

Out-In	Name	Type	Size	Offset	Description
–>	ioCompletion	ProcPtr	4	12	Completion routine address (if *async* =TRUE)
–>	ioRefNum	short	2	24	Device reference number
–>	ioBuffer	Ptr	4	32	Address of I/O buffer
–>	ioReqCount	long	4	36	I/O transfer size, in bytes
–>	ioPosMode	short	2	44	Positioning Mode (0=current position)
–>	ioPosOffset	long	4	46	Positioning delta (bytes from ioPosMode position)
<-	ioResult	OSErr	2	16	Error Code (0=no error, 1=not done yet)
<-	ioActCount	long	4	40	Actual number of bytes transferred
–>	ioVRefNum	short	2	22	Drive identification

async is a Boolean value. Use FALSE for normal (synchronous) operation or TRUE to enqueue the request and resume control immediately. See Async I/O.

Device Mgr Returns: an operating system Error Code. It will be one of:

noErr	(0)	No error
badUnitErr	(-21)	*refNum* doesn't match unit table
unitEmptyErr	(-22)	*refnum* specifies NIL handle in unit table
notOpenErr	(-28)	Driver closed
writErr	(-19)	Driver can't respond to Write

Device Manager Notes:

PBWrite transfers ioReqCount bytes starting at ioBuffer into a device driver, starting at the device driver position specified by ioPosMode and ioPosOffset. Upon completion, ioActCount contains the actual number of bytes written.