The Preferred Interface A new high-level language interface for AppleTalk

A new parameter block-style interface to the **AppleTalk Manager** is now available for high-level language programmers. This new interface, referred to as the preferred interface, is available in addition to the original interface which is referred to as the alternate interface. All old and new **AppleTalk Manager** calls are supported by the preferred interface.

The alternate interface has not been extended to support the newly introduced **AppleTalk Manager** calls. However, the alternate interface provides the only implementation of **LAPRead** and **DDPRead**. These are higher-level calls not directly supported through the preferred interface. Developers will wish to use the alternate interface for these calls, and also for compatibility with previous applications. In all other cases, it is recommended that the new preferred interface be used.

Using High-level language

All **AppleTalk Manager** calls in the preferred interface are essentially equivalent to the corresponding assembly language calls. Their form is:

iErr = MPPCall (*pbPtr, asyncFlag*);

where *pbPtr* points to a **Device Manager** parameter block, and asyncFlag is TRUE if the call is to be executed asynchronously. Three parameter block types are provided by the preferred interface (MPP, ATP, and XPP). The caller fills in the parameter block fields and issues the appropriate call. The interface issues the actual **Control** call to the **Device Manager**.

On asynchronous calls, the caller may pass a completion routine pointer in the parameter block, at offset ioCompletion. This routine will be executed upon completion of the call. It is executed at interrupt level and must not make any memory manager calls. If it uses application globals, it must ensure that register A5 is set up correctly; for details see **SetupA5**. If no completion routine is desired, ioCompletion should be set to NIL.

Asynchronous calls return control to the caller with the result code of noErr as soon as they are queued to the driver. This is not an indication of successful completion. To determine when the call is actually completed, if you do not want to use a completion routine, you can poll the ioResult field; this field is set to 1 when the call is made, and receives the actual result code upon completion.

As different MPP and ATP calls take different arguments in their parameter block, two high-level language variant records have been defined to account for all the different cases. The first four fields (which are the same for all calls) are automatically filled in by the **Device Manager**. The csCode and ioRefnum fields are automatically filled in by the interface, depending on which call is being made, except in XPP where the caller must fill in the ioRefnum. The ioVRefnum field is unused.

There are two fields that at the assembly-language level have more than one name. These two fields have been given only one name in the preferred interface. These are entityPtr and ntqelPtr, which are both referred to as entityPtr , and atpSocket and currBitmap, which are both referred to as atpSocket. These are the only exceptions to the naming convention.

The following is table listing the older alternate interface calls and the corresponding preferred interface calls.

LAP Functions

Alternate Interface	*Preferred Interface*
LAPOpenProtocol	PAttachPH
LAPCloseProtocol	PDetachPH
LAPWrite	PWriteLAP
LAPRead	
LAPRdCancel	

DDP Functions

Alternate Interface	*Preferred Interface*
DDPOpenSocket	POpenSkt
DDPCloseSocket	PCloseSkt
DDPWrite	PWriteDDP
DDPRead	
DDPRdCancel	

ATP Functions

Alternate Interface	*Preferred Interface*
ATPOpenSocket	POpenATPSkt
ATPCloseSocket	PCloseATPSkt
ATPSndRequest	PSendRequest
ATPRequest	PSendRequest
ATPGetRequest	PGetRequest
ATPSndRsp	PSendResponse
ATPResponse	PSendResponse
ATPAddRsp	PAddResponse
ATPReqCancel	PRelTCB
ATPRspCancel	PRelRspCB

NBP Functions

Alternate Interface	*Preferred Interface*
NBPLookup	PLookupName
NBPConfirm	PConfirmName
NBPRegister	PRegisterName
NBPRemove	PRemoveName
NBPLoad	
NBPUnLoad	