Receiving Session Requests

Your application can open as many ports as it requires as long as each port name is unique within a particular computer. A single port can support a number of communication sessions. To allow a port to receive session requests, use the **PPCInform** function. (Note that you must open a port to obtain a port reference number before calling the **PPCInform** function.) A port may have any number of outstanding **PPCInform** requests.

The following program illustrates how you use the **PPCInform** function to allow a port to receive session requests. In this listing, the parameter thePPCParamBlockPtr points to a PPC parameter block record allocated by the application. The portRefNum, autoAccept, portName, locationName, userName, and ioCompletion parameters of the PPC parameter block record must be supplied. If you want to automatically accept all incoming session requests, you can set the autoAccept field in the **PPCInform** parameter block.

```
// Using the PPCInform function to enable a port to receive sessions

// Assuming inclusion of MacHeaders
#include <PPCToolBox.h>

// Prototype your function like this prior to calling it
OSErr MyPPCInform(PPCParamBlockPtr,  PPCPortPtr,  LocationNamePtr,
                  StringPtr,  PPCPortRefNum);

OSErr MyPPCInform(PPCParamBlockPtr   thePPCParamBlockPtr,
                  PPCPortPtr         thePPCPortPtr,
                  LocationNamePtr    theLocationNamePtr,
                  StringPtr          theUserNamePtr,
                  PPCPortRefNum      thePortRefNum
    )
{
// this is how you prototype MyInformCompProc
pascal void MyInformCompProc(PPCParamBlockPtr);

thePPCParamBlockPtr->informParam.ioCompletion   = &MyInformCompProc;

// from the PPCOpen function
thePPCParamBlockPtr->informParam.portRefNum     = thePortRefNum;

// the completion routine handles accepting or rejecting requests
thePPCParamBlockPtr->informParam.autoAccept = FALSE;
thePPCParamBlockPtr->informParam.portName = thePPCPortPtr;
thePPCParamBlockPtr->informParam.locationName = theLocationNamePtr;
thePPCParamBlockPtr->informParam.userName = theUserNamePtr;

return PPCInform((PPCInformPBPtr)thePPCParamBlockPtr, TRUE);
// asynchronous
}
```

A PPC parameter block record is used instead of a **PPCInform** parameter block record so that the same parameter block can be reused to make other **PPC Toolbox** calls from the **PPCInform** completion routine. The

parameter block and the records it points to cannot be deallocated until all calls that use the parameter block and records have completed.

You should make the call to **PPCInform** asynchronously. For each function that you use asynchronously, you should provide a completion routine. This procedure gets called at interrupt time when the **PPCInform** function completes. If there are no errors, it sets the global variable gSessionOpen to TRUE. The global gPBInUse is set to FALSE to inform the application that the parameter block and the records it points to are no longer in use.

The following program illustrates a completion routine for a **PPCInform** function. You can use the data passed into your **PPCInform** completion routine (user name, user data, port name, and location name) to determine whether to accept or reject the session request.

```
// Completion routine for a PPCInform function

// Assuming inclusion of MacHeaders
#include <PPCToolBox.h>

// make sure to prototype your routine like this prior to using it
// Note, since this routine will be called from the toolbox, it must
// use pascal calling conventions
pascal void MyInformCompProc( PPCParamBlockPtr );

// global variable to tell application that PPCParamBlockRec can be deallocated
Boolean gPBInUse;

pascal void MyInformCompProc( PPCParamBlockPtr pb)

{
    // Prototypes for user defined routines to accept or reject sessions
    void DoPPCAccept(PPCParamBlockPtr);
    void DoPPCReject(PPCParamBlockPtr);

    if ( pb->informParam.ioResult == noErr ) {
        // decide if this session should be accepted or rejected
        // by looking at data supplied by the session requester
        if ( pb->informParam.userData != -1 )
        DoPPCAccept(pb);
      else
        DoPPCReject(pb);
    }
    else
        // use a global to tell the application that PPCParamBlockRec
        // and the records it points to can be deallocated}
    gPBInUse = FALSE;
}
```