Using the ENET Handler Procedural outline

When the EtherTalk NuBus card or other Ethernet hardware receives a data packet, it generates an interrupt to the CPU. Next, the interrupt handler in ROM determines the source of the interrupt and calls **The .ENET Driver**. **The .ENET Driver** reads the packet header to determine the protocol type of the data packet and checks to see if any client has specified that protocol type in a call to the **EAttachPH** function. If so, the client either specified a NIL pointer to a protocol handler, or the client provided its own protocol handler. If the client specified a NIL pointer, **The .ENET Driver** uses its default protocol handler to read the data. If no one has specified that protocol type in a call to the **EAttachPH** function, **The .ENET Driver** discards the data.

The default protocol handler checks for an **ERead** function pending execution and places the entire packet-including the packet header-into the buffer specified by that function. The function returns the number of bytes actually read. If the packet is larger than the data buffer, the **ERead** function places as much of the packet as will fit into the buffer and returns the buf2SmallErr result code.

Call the **ERead** function asynchronously to await the next data packet. When **The .ENET Driver** receives the data packet, it completes execution of the **ERead** function and calls your completion routine. Your completion routine should call the **ERead** function again so that an **ERead** function is always pending execution. If **The .ENET Driver** receives a data packet with a protocol type for which you specified the default protocol handler while no **ERead** function is pending, **The .ENET Driver** discards the packet.

You can have several asynchronous calls to the **ERead** function pending execution simultaneously as long as you use different buffers and a different parameter block for each call.

The following program listing calls the **EAttachPH** function in order to specify that **The .ENET Driver** should use the default protocol handler to process protocol type eProtType. The listing includes a completion routine that processes a received data packet and then makes an asynchronous call to the **ERead** function to await the next incoming data packet.

In practice, you should call the **EAttachPH** function very early, during your program initialization sequence, if possible. As soon as the connection is established and you are expecting data, you should call the **ERead** function asynchronously. When **The .ENET Driver** receives a packet, it calls your completion routine, which should process the packet and queue another asynchronous call to the **ERead** function to await the next packet.

```
// Using the default Ethernet protocol handler to read data
// Assume inclusion of <MacHeaders>

#include <ENET.h>

#define     BigBytes   8888

EParamBlkPtr      gEPBPtr;
Ptr           gAPtr;
```

```
pascal void MyCompRoutine (void);
void ProcessData (short size, Ptr ptrToData);   // do something with the data
void DoError (OSErr myErr);

pascal void MyCompRoutine (void)
{
    OSErr  myErr;

    // If this gets called, an incoming packet with the specified protocol
    // type is here.

    ProcessData(BigBytes, gAPtr);        // do something with the data
    if (gEPBPtr->EParms1.ioResult)
        DoError(myErr);

    // call ERead again
    // set up ERead parameters

    gEPBPtr->EParms1.ioCompletion = &MyCompRoutine; // pointer to
                                    // completion routine
    gEPBPtr->EParms1.eProtType = 77;     // protocol type
    gEPBPtr->EParms1.ePointer = gAPtr;   // pointer to read-data area
    gEPBPtr->EParms1.eBuffSize = BigBytes;   // size of read-data area

    myErr = ERead(gEPBPtr, TRUE);        // call ERead to wait for
                                // the next packet
    if (myErr)
        DoError(myErr);
}

main ()
{
    OSErr  myErr;
    EParamBlock  myPB;

    gEPBPtr = &myPB;
    gEPBPtr->EParms1.eProtType = 77;     // protocol type
    gEPBPtr->EParms1.ePointer = nil;  // use default protocol handler

    myErr = EAttachPH(gEPBPtr, FALSE);     // tell .ENET about this
                                //  protocol handler
    if (myErr)
        DoError(myErr);

    gAPtr = NewPtr(BigBytes);

    gEPBPtr->EParms1.ioCompletion = &MyCompRoutine; // pointer to
                                //completion routine
    gEPBPtr->EParms1.eProtType = 77;          // protocol type
    gEPBPtr->EParms1.ePointer = gAPtr;   // pointer to read-data area
    gEPBPtr->EParms1.eBuffSize = BigBytes;   // size of read-data area

    myErr = ERead(gEPBPtr, TRUE);   // wait for your packet and
                            // then read it
    if (myErr)
```

```
        DoError(myErr);

    // application-defined tasks

} // main
```