The .ENET Driver EtherTalk data

The .ENET Driver is normally called by the **AppleTalk Manager** through the AppleTalk connection file for EtherTalk when the user has selected EtherTalk from the Network control panel. You can write your own protocol stack or application that uses **The .ENET Driver** directly, rather than through AppleTalk. This section describes how to open **The .ENET Driver**, how to send data to it directly for transmission over the Ethernet network, and how to write a protocol handler to receive data from the network.

The system .ENET driver locates and opens the drivers for installed NuBus Ethernet cards. For each Ethernet NuBus card, **The .ENET Driver** searches the open resource files for a driver with a resource type of 'enet' and a resource ID equal to the board ID of the NuBus card. If it doesn't find such a driver resource, it then looks for a driver named .ENET in the slot resources in the ROM of the NuBus card. See *Designing Cards and Drivers for the Macintosh Family,* second edition, for discussions of NuBus board IDs and slot resources.

Providing Your Own Ethernet Driver

If you write an Ethernet driver for use with your own Ethernet NuBus card, you should provide the features and functions described here. You can store the driver in the firmware of the NuBus card as described in detail in *Designing Cards and Drivers for the Macintosh Family,* second edition, and in **Device Manager**, or you can provide a RAM-based driver as described in **Device Manager** . If you place your Ethernet driver in the ROM of the NuBus card, you must name the driver .ENET. If you provide a RAM-based driver, you must give it a resource type of 'enet' and a resource ID equal to the board ID of your NuBus card. The 'enet' resource type is identical to the 'DRVR' resource type described in **Device Manager**.

> **Note:** You must *not* name a RAM-based driver .ENET, because doing so would replace the system .ENET driver.

If you write an Ethernet driver for use with a non-NuBus network interface (such as an Ethernet card for the Macintosh SE/30 or an Ethernet connection through the SCSI port), you should provide the features and functions described for **The .ENET Driver** and should name your driver .ENET0. If you do so, any software written to use **The .ENET Driver** should work with your driver.

Changing the Ethernet Hardware Address

Each Ethernet NuBus card or other Ethernet hardware interface device contains a unique 6-byte hardware address assigned by the manufacturer of the device. **The .ENET Driver** normally uses this address to determine whether to receive a packet. To change the hardware address for your node, place in the System file a resource of type 'eadr' with a resource ID equal to the slot number of the Ethernet NuBus card. If the Ethernet device is not a NuBus card (it might be a slot card in a Macintosh SE/30, for example), use a resource ID of 0.

Do not use the broadcast address or a multicast address for this number. (The **broadcast address** is 0x0FF-FF-FF-FF-FF-FF. A multicast address is any Ethernet address in which the low-order bit of the high-order byte is set to 1.) When you open **The .ENET Driver**, it looks for an 'eadr' resource. If it

finds one, the driver substitutes the number in this resource for the Ethernet hardware address and uses it until the driver is closed or reset.

> **Note:** To avoid address collisions, you should never arbitrarily change the Ethernet hardware address. This feature should be used only by a system administrator who can keep track of all the Ethernet addresses in the system.

Opening the .ENET Driver

Before you use the **OpenSlot** function to open **The .ENET Driver**, you must determine which NuBus slots contain EtherTalk cards. The **OpenSlot** function is described in **Device Manager**. Use the **SGetTypeSRsrc** function described in **Slot Manager** to determine which NuBus slots contain EtherTalk cards. To find EtherTalk NuBus cards, use the value catNetwork in the field spCategory of the **SGetTypeSRsrc** function parameter block, and use the value typeEtherNet in the field spCType. If you cannot find any EtherTalk NuBus cards, you should also attempt to open the .ENET0 driver in case a non-NuBus EtherTalk card is attached to the system. You should provide a user interface that allows the user to select a specific EtherTalk card in the case that more than one is present.

The following code example illustrates the use of the **SGetTypeSRsrc** function and the **OpenSlot** function to open **The .ENET Driver**.

```
// Finding an EtherTalk card and opening the .ENET driver
// Assume inclusion of <MacHeaders>

#include <AppleTalk.h>
#include <ROMDefs.h>
#include <Slots.h>
#include <string.h>

void SaveSInfo(SpBlock *mySBlk);
void DisplaySInfo(SpBlock *mySBlk);
void DoError (OSErr myErr);

SpBlock         mySBlk;
ParamBlockRec   myPBRec;
OSErr           myErr;
short           found;
Str15           enetStr;
Str15           enet0Str;
short           myRefNum;

found = 0;              // assume no sResource found
strcpy ((char *) enetStr, (char *) "\pENET");
strcpy ((char *) enet0Str, (char *) "\p.ENET0");

mySBlk.spParamData = 1;     // include search of disabled resources.
                            // Start searching from spSlot and search
                            // the slots above it as well.
mySBlk.spCategory = catNetwork;
mySBlk.spCType = typeEtherNet;
mySBlk.spDrvrSW = 0;
```

```
mySBlk.spDrvrHW = 0;
mySBlk.spTBMask = 3;    // match only Category and CType fields
mySBlk.spSlot = 0;          // start search from here
mySBlk.spID = 0;       // start search from here
mySBlk.spExtDev = 0;     // ID of the external device

do {
    myErr = SGetTypeSRsrc(&mySBlk);

    if (!myErr) {

        // You found an sResource match; save it for later.

        found++;
        SaveSInfo(&mySBlk);
    }
}   while ( myErr != smNoMoresRsrcs);


if ( found > 1 )
{
    // if you found more than one sResource, put up a dialog box
    //  and let the user choose one. if any of the sResources you
    //  found were disabled, let the user know they are not available.

    DisplaySInfo(&mySBlk);
}

if (found)
{
    myPBRec.slotDevParam.ioCompletion = nil;
    myPBRec.slotDevParam.ioNamePtr = enetStr;
    myPBRec.slotDevParam.ioMix = nil;      // reserved
    myPBRec.slotDevParam.ioFlags = 0;      // single device sResource
    myPBRec.slotDevParam.ioSlot = mySBlk.spSlot;
    myPBRec.slotDevParam.ioID = mySBlk.spID;
    myErr = OpenSlot(&myPBRec, FALSE);    // go open this
}
else
    myErr = OpenDriver(enet0Str, &myRefNum);
if (myErr)
    DoError(myErr);
```