Using the Slot Manager

The **Slot Manager** released with system 7.0 or later allows you to do four things you couldn't do before:

- You can temporarily disable and then reenable a NuBus card.
- You can enable and disable sResource data structures.
- You can search for disabled sResource data structures.
- You can restore an sResource data structure that was deleted from the Slot Resource Table.

This section describes how to do each of these things and provides some sample code to illustrate how to do them.

Enabling and Disabling NuBus Cards

If your NuBus card must be addressed in 32-bit mode or requires that RAM-based system software patches be loaded into memory before the card is initialized, you can use the code in the PrimaryInit record to disable the card temporarily and the code in the SecondaryInit record to reenable it. To determine whether or not the system 7.0 (or later) **Slot Manager** is present and, if present, whether it is the RAM-based version (version 1) or the ROM-based version (version 2), use the **SVersion** function. If version 2 of the **Slot Manager** is present, you can assume that the **Slot Manager** operates in 32-bit mode. However, if you want to be sure that all RAM patches to the Operating System have been loaded before your card is used, you can still use the method described here to disable your NuBus card temporarily.

To disable a NuBus card temporarily, the initialization routine in your PrimaryInit record should return in the seStatus field of the **SEBlock** data structure an error code with a value in the range svTempDisable (0x8000) through svDisabled (0x08080). The **Slot Manager** places this error code in the siInitStatusV field of the **sInfoRecord** that the **Slot Manager** maintains for that slot, and it places the fatal error smInitStatVErr (-316) in the siInitStatusA field of the sInfoRecord. The card and its sResource data structures are then unavailable for use by the Operating System.

After the Operating System loads RAM patches, the system 7.0 or later **Slot Manager** checks the value of the siInitStatusA field. If this value is greater than or equal to 0, indicating no error, the **Slot Manager** executes the SecondaryInit code in the declaration ROM of the NuBus card. If the value in the siInitStatusA field is smInitStatVErr, the **Slot Manager** checks the siInitStatusV field. If the value of the siInitStatusV field is in the range svTempDisable through svDisabled, the **Slot Manager** clears the siInitStatusA field to 0 and runs the SecondaryInit code. In other words, the system 7.0 or later **Slot Manager** runs the SecondaryInit code if the PrimaryInit code returns no error or returns an error in the range svTempDisable through svDisabled. If the PrimaryInit code returns any other result code less than 0 (that is, any other fatal error), the NuBus card remains disabled and the **Slot Manager** does not run the SecondaryInit code. See **sExec** parameter block and the **sInfoRecord**.

For examples of PrimaryInit and SecondaryInit records that test for the presence of the system 7.0 or later **Slot Manager** and act accordingly, see *Designing Cards and Drivers for the Macintosh Family,* second edition.

Enabling and Disabling SResource Data Structures

Under certain circumstances, you might want to disable an sResource data structure while it remains listed in the Slot Resource Table. For example, a NuBus card might provide several modes of operation, only one of which can be active at a given time. Your application might want to disable the sResource data structures associated with all but the active mode, but still list all available modes in a menu. When the user selects a new mode, your application can then disable the currently active sResource data structure and enable the one the user selected.

You use the **SetSRsrcState** function to enable or disable an sResource data structure. The following code example disables the sResource data structure in slot 0x0A with an ID of 128 and enables the sResource data structure in the same slot with an ID of 131.

```
//Disabling and enabling sResource data structures

// Assuming inclusion of <MacHeaders>

#include <Slots.h>

void DoError (OSErr myErr);

SpBlock     mySpBlk;
OSErr  myErr;

// Set required values in parameter block.

mySpBlk.spParamData = 1;              // disable
mySpBlk.spSlot = 0x0A;                // slot number
mySpBlk.spID = 128;                   // sResource ID
mySpBlk.spExtDev = 0;                 // ID of the external device

myErr = SetSRsrcState(&mySpBlk);
if (myErr)
    DoError(myErr);


mySpBlk.spParamData = 0;              // enable
mySpBlk.spSlot = 0x0A;                // slot number
mySpBlk.spID = 131;                   // sResource ID
mySpBlk.spExtDev = 0;                 // ID of the external device

myErr = SetSRsrcState(&mySpBlk);
if (myErr)
    DoError(myErr);
```