Searching for Disabled SResource Data Structures

Whereas earlier **Slot Manager** routines act only on enabled sResource data structures, you can use system 7.0 or later **Slot Manager** routines to search for both enabled and disabled sResource data structures. The system 7.0 or later **Slot Manager** also allows you to specify whether the search should be for the specified sResource data structure or the next sResource data structure and whether the search should include only the specified slot or should include the specified slot plus all slots with higher numbers. In addition, you can specify the type of the sResource data structure for which you want to search; then the **Slot Manager** ignores all sResource data structures that do not match the specified type.

The following Table summarizes the **Slot Manager** search routines and the options available for each.

Table **Slot Manager** search routines

Function	State of sResource it searches for	Which Slots it searches	Type of sResource it searches for	sResource it searches for
SNextSRsrc	Enabled only	Specified slot and higher slots	Next sResource only	Any type
SGetSRsrc*	Your choice of enabled only or both enabled and disabled	Your choice of one slot only or specified slot and higher slots	Your choice of specified sResource or next sResource	Any type
SNextTypeSRsrc	Enabled only	Specified slot and higher slots	Next sResource only	Specified type only
SGetTypeSRsrc*	Your choice of enabled only or both enabled and disabled	Your choice of one slot only or specified slot and higher slots	Next sResource	Specified type only

* Available only with system 7.0 or later **Slot Manager**

The code example below shows how to search all slots for all sResource data structures with a specific value in the Category and cType fields, whether enabled or disabled. The **Slot Manager** ignores the DrvrSW and DrvrHW fields of the resource type.

Restoring Deleted SResources

Some NuBus cards have sResource data structures to support a variety of combinations of system configurations or modes. The **Slot Manager** loads all of the sResource data structures during system initialization, and then the PrimaryInit code in the declaration ROM deletes from the Slot Resource Table any sResource data structures that are not appropriate for the system as configured. The system 7.0 or later **Slot Manager** gives you the option of

reinstalling a deleted sResource data structure if, for example, the user changes the system configuration or selects a different mode of operation. The **SDeleteSRTRec** function deletes sResource data structures; the **InsertSRTRec** function reinstalls them.

```
// Searching for sResource data structures

// Assuming inclusion of <MacHeaders>

#include <RomDefs.h>
#include <Slots.h>

void DoError (OSErr myErr);
void MysRsrcProc(SpBlock mySpBlk);

SpBlock     mySpBlk;
OSErr  myErr;

// Set required values in parameter block.


mySpBlk.spParamData = 1;                // fAll flag set to 1 to include
                                        // disabled resources; fOneslot
                                        // flag set to 0 to search specified
                                        // slot plus higher-numbered slots

mySpBlk.spCategory = catDisplay;        // Category field of sRsrcType
                                        // entry in sResource

mySpBlk.spCType = typeVideo;            // cType field of sRsrcType entry in
                                        // sResource

mySpBlk.spDrvrSW = 0;                   // DrvrSW field of sRsrcType entry
                                        // in sResource; this field is not
                                        // being matched

mySpBlk.spDrvrHW = 0;                   // DrvrHW field of sRsrcType entry
                                        // in sResource; this field is not
                                        // being matched

mySpBlk.spTBMask = 3;                   // match only spCategory and
                                        // spcType fields
mySpBlk.spSlot = 0;                     // start search from here
mySpBlk.spID = 128;// start search from here
mySpBlk.spExtDev = 0;                   // ID of the external device

myErr = 0;

while (!myErr) {      // loop to search sResources
    myErr = SGetTypeSRsrc(&mySpBlk);
    MysRsrcProc(mySpBlk);               // your routine to process results
}

if ( myErr != smNoMoresRsrcs )
DoError(myErr);
```

Because none of the **Slot Manager** functions can search for sResource data structures that have been deleted from the Slot Resource Table, you must keep a record of all sResource data structures that you have deleted so that you will have the appropriate parameter values available if you want to reinstall one.

When you reinstall an sResource data structure, you can also update the *dCtlDevBase* field in the device driver's **Device Control Entry** (**DCE**) data structure. The *dCtlDevBase* field holds the address of the sResource data structure that is used by that device driver. For a video driver, for example, the *dCtlDevBase* field might contain the address of the frame buffer. Use the **InsertSRTRec** function to update the *dCtlDevBase* field. See **Device Manager** for a definition of the **DCE** data structure.