New Technical Notes

Macintosh

Developer Support

Slot Manager Q&As

Devices

Revised by: Developer Support Center May 1993
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

Apple-defined sResource sRsrcHWDevID entry
Date Written: 6/25/92
Last reviewed: 7/13/92

In Apple's technical reference *Designing Cards and Drivers for the Macintosh Family,* 2nd edition (Chapter 8, page 123), the functional sResource entry called sRsrcHWDevId is discussed. The book says that this entry is necessary for hardware devices. Please explain the purpose of this entry in the Declaration ROM functional sResource. Do I generate this ID number or does Developer Support generate this number?

—

You generate this sRsrcHWDevId number. However, the hardware device ID is optional, not necessary. Why might you want to use it? You might want a hardware device ID if you had multiple hardware devices (and it's totally up to you to define what a "hardware device" is) on a card and wanted certain sResources to be affiliated with certain hardware devices. As an example, let's say your card had three hardware devices and six sResources. If, say, you wanted two sResources to be affiliated with each "hardware device", you could number the devices (maybe from 1 to 3) and have the sResources tagged with the affiliated hardware devices. To be truthful though, this is another feature of the Slot Manager (when the Slot Manager was designed, no one really knew how developers would want to use it, so it was

made very, very flexible), and, as far as we know, no one uses this feature.

How to tell if Slot Manager is implemented

Date Written: 10/22/91
Last reviewed: 8/5/92

How do you tell whether the Slot Manager is implemented?

—

The best way is to just check for the existence of one of the Slot Manager traps. You should be able to check the implementation of Trap $A06E and if it is there, then the Slot Manager will be there. Below is a piece of code which does this check:

```
#define SlotManager       0xA06E
#define NoTrap            0xA89F

main()
{
    if ((GetTrapAddress(SlotManager) != GetTrapAddress(NoTrap)))
         SysBeep(5);        // if the Slot Manager is avail then beep
}
```

SGetCString and memory (de)allocation
Date Written: 9/10/91
Last reviewed: 8/5/92

Inside Macintosh Volume V (page 445) states that SGetCString automatically allocates memory for holding the requested string. Does this mean that repeated calls to SGetCString will eventually exhaust memory and, if so, what's the correct way to release the memory?

—

SGetCString automatically allocates memory (via NewPtr) for holding the requested string. It copies the string into that pointer and returns the pointer in spResult. You need to call DisposPtr on spResult to deallocate the pointer.

NuBus card prototype and declaration ROM development
Date Written: 10/8/91
Last reviewed: 8/5/92

Is it possible to convince the Slot Manager *after* the boot process that a NuBus card is now in a previously (apparently) empty slot?. Is there some standard declaration ROM to burn in order to have a "more generic" look to the system? We want to slam cards in and out of a system, and make all kinds of weird prototype boards without changing or making new declaration ROMs.

—

You'd have to hack the slot resource and slot info tables, which are undocumented. It probably would be doable, but not all that easy; just restarting would be easier and faster.

To do as much as you can running code from an application, pretending all the code that

goes to the declaration ROM “works” would be easier, as you say. Then, only at the last, when you have everything working, put the driver in the ROM. You might make a very simple ROM that just lets the Slot Manager know a card is installed, run as much code as you can from the application level until it’s debugged, and then fold it into the ROM.

If you need code to a “generic” sort of ROM, why not take the skeleton from “Debugging Declaration ROMs” in the first issue of develop, available on the latest Developer CD Series disc. It would take very minor tweaking to work in your system.

Macintosh LC “Sad Mac” error code 0000000F 00000033
Date Written: 3/8/91
Last reviewed: 8/5/92

What is a Macintosh LC “Sad Mac” error 0000000F 00000033?

—

This Macintosh LC error code means “unserviceable slot interrupt”—that is, a slot interrupt occurred but the Slot Manager was unable to find an interrupt service routine that would handle the interrupt. This might occur if the primary initialization (PrimaryInit) code for a card fails to establish an interrupt handler for the card but enables interrupts for the slot containing that card anyway.

Checking for sRsrcInfo
Date Written: 9/24/91
Last reviewed: 8/5/92

Is there a way from Pascal to check availability of traps like SRsrcInfo (which is not really a trap but a selector). On a Macintosh Plus, the trap SlotManager (A06E) is available but the debugger says the call sRsrcInfo was stopped is not available.

—

Use NGetTrapAddress, as in the following example:

```
{verify SlotManager is implemented}
IF (NGetTrapAddress(SlotManagerTrap,OSTrap) <>
    NGetTrapAddress(UnImplementedTrap,OSTrap)) THEN
        BEGIN
        {SM implemented}
        END
```

Code in the Compatibility Guidelines chapter of *Inside Macintosh* Volume VI shows this in more detail, and is more robust.

Detecting a NuBus or PDS card in the Macintosh IIsi slot
Date Written: 1/16/90
Last reviewed: 8/5/92

Why does a call to gestaltNuBusConnectors return zero slots when there actually is a NuBus card or a PDS (Processor Direct Slot) card in the Macintosh IIsi slot?

—

A call to gestaltNuBusConnectors returns zero slots because there's no way to determine if there's a NuBus card or a PDS card in the Macintosh IIsi slot. Gestalt can't assume that there's always a NuBus or PDS slot, so it just says there's no slot. However, Apple recommends that

you always use the Slot Manager instead of Gestalt to search for cards, after first checking to see that the Slot Manager trap is implemented. The Slot Manager will safely do all the necessary work for you whether a card with a valid declaration ROM is installed or not, and you can search for the card using a variety of criteria.This technique will allow you to locate NuBus cards, but has the added benefit of being able to find PDS cards that contain declaration ROMs.

The Macintosh SE/30, IIfx, and LC have the Slot Manager and PDS slots. It's safe to assume the Slot Manager will be resident in all future machines that have either NuBus or PDS slot capability.

Ghost image when scanning Macintosh IIfx slots with the Slot Mgr
Date Written: 6/29/90
Last reviewed: 8/5/92

When I scan the Macintosh IIfx slots with the Slot Manager I find a ghost image of a card in slot 1. Am I doing something wrong, or is this a bug?

—

This is a bug in the Macintosh IIfx Fast Memory Controller (FMC), but it should not cause you any problems.