

Re: IRQ Sharing with PCI Device

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2006-10/msg00676.html>

- *From:* "bbartson" <bbartson@xxxxxxxxxxxxx>
 - *Date:* 19 Oct 2006 08:13:36 -0700
-

already5chosen@xxxxxxxxx wrote:

bbartson wrote:

Hello,

I've inherited a windows driver that was designed for the NT4 style architecture (non-PNP and non-WDM). Our company manufactures VMEbus cpu boards. This driver is for interfacing to the vmebus (via a pci-to-vme bridge device, the Tundra chip, which is a single function device). This driver works just fine on our older (2000 design) cpu boards which are using the same chip but a Phoenix bios.

We've designed a new cpu, using the same chip but with a new bios (from General Software). The bios is ACPI compliant but currently it is only running in PIC mode (not APIC). My problem is this old NT4 style driver fails when it tries to allocate interrupts. It can get access to IO ports and PCI memory via the Tundra OK. The interrupt allocation fails with STATUS_CONFLICTING_ADDRESSES. This error code comes from IoAssignResources.

The bios's pci enumeration allocates int 11 to the Tundra device. This gets stored in the pci config space for the Tundra device and this is what the driver attempts to use too. If I look at device manager I can see that ints 9,10 & 11 are shared between about 10 different devices (the Tundra shows up as a 'pci bridge device'). Mostly 6300 south bridge devices are on these ints.

I have tried many different things to work around this. I have tried turning off PlugNPlay, tried masking certain INTs in the bios, mods to the driver code to use other api calls such as IoReportDetectedDevice, etc. Currently APIC mode is not turned ON in the bios because according to the bios vendor it is not fully debugged yet and they are working on fixing it. However, they don't think this will resolve the issue because the Tundra is wired on the board to use INT#A and this is shared with other devices.

Re: IRQ Sharing with PCI Device

So my question is do you think that APIC mode would resolve this and if not is there some other approach the driver code could take, OR is this more likely a fundamental hardware issue with how the interrupts are routed on the board.

Thanks,

Nt4-style PCI drivers must use HalAssignSlotResources(). All other approaches are conflict-prone.

Of course, if backward compatibility is not required you could just convert everything to WDM. Then instead of being at mercy of BIOS vendor you end up at mercy of Microsoft. I'm not sure which one is worse :(

Thanks for the reply. So I've started using HalAssignSlotResources however now I get an error from IoConnectInterrupt (STATUS_INVALID_PARAMETER = 0xC000000D). Is there some other call I should be using instead of this? BTW, all of this legacy code works on one of our older board designs with XP and a Phoenix bios. Backward compatibility is not a requirement and I do plan to rewrite as WDM but right now I just need to get it working so we can show our customers the new hardware is viable.

This is what the call looks like :

```
status = IoConnectInterrupt(  
&m_InterruptObject,  
Isr,  
context,  
NULL, //m_pSpin  
m_Vector,  
m_Irql,  
m_SynchIrql,  
m_Mode,  
TRUE, //m_bShareVector,  
m_Affinity,  
FALSE //m_bSaveFloat  
);
```

And the debug output from WinDbg (DbgPrint immediately after call to IoConnect) :

```
IN KInterrupt::Connect IoConnectInterrupt call, status=0xC000000D,  
Isr=0xF65D7D1A, context=0x85D26C80 m_pSpin=0x0, m_Vector=0x39,  
m_Irql=0x12, m_SynchIrql=0x12, m_Mode=0x1, m_Affinity=0x1
```

.