

Write to the PCI Bus

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2008-01/msg00133.html>

- *From:* Bryan <Bryan@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 6 Jan 2008 18:58:00 -0800
-

Eventually I plan to use DMA for my PCI read and write I/O operations, but I'm still trying to do some work with my PCI device before I want to worry about DMA. Specifically I'd like to just write a few bytes to my PCI device for now, so I was hoping that I could write directly to it using the virtual address for the PCI memory map. I tried the following method and it did not work so I was looking for a little guidance.

In my EvtDevicePrepareHardware(), using the translated resource list I use MmMapIoSpace() to map my device's PCI window address to a virtual address. I record this memory mapped address along with the length in my device context.

```

-----
NTSTATUS EvtDevicePrepareHardware(
IN WDFDEVICE Device,
IN WDFCMRESLIST ResourcesRaw,
IN WDFCMRESLIST ResourcesTranslated
){
// There are six memory windows, only interested in the first for now though
PHYSICAL_ADDRESS memWindow[6] = { {0}, {0}, {0}, {0}, {0}, {0} };
ULONG memWindowLen[6] = { 0, 0, 0, 0, 0, 0 };
PCM_PARTIAL_RESOURCE_DESCRIPTOR desc;
//.....
pDevCon = GetDeviceContext( Device );
//.....
desc = WdfCmResourceListGetDescriptor( ResourcesTranslated, i );
switch( desc->Type )
{
// .....
case CmResourceTypeMemory:
memWindow[0] = desc->u.Memory.Start;
memWindowLen[0] = desc->u.Memory.Length;
pDevCon->memWindow0Base = (PUCHAR) MmMapIoSpace( memWindow[0],
memWindowLen[0], MmNonCached );
// .....
}
}
-----

```

Write to the PCI Bus

In EvtIoWrite() I call WdfRequestRetrieveInputMemory() to get a WDFMEMORY object for the Request, and then tried the following two methods for writing to the data.

- WdfMemoryCopyToBuffer(): it returns STATUS_SUCCESS as I thought
- Call WdfMemoryGetBuffer() to get a pointer to the buffer, use another pointer to the virtual address I received from MmMapIoSpace, and manually copy the data byte by byte.

From my user mode app I call WriteFile and pass a the address to a 4 byte

buffer (contents 0x1, 0x2, 0x3, 0x4), a length variable set to 4, and the address of a variable to find out how man bytes the system call actually wrote.

Both methods seem to be working as the function calls all return with the expected vaules and the number of bytes written was 4. As well, on my PCI device via a JTAG debugger I can see that the four bytes were written to the device.

However.....

The data written does not appear to be correct. Instead of { 0x01, 0x02, 0x03, 0x04 }, the data I get from WdfRequestRetrieveInputMemory() is { 0x34, 0xff, 0x12, 0x00 }. I checked my data buffer in user space and it is correct before the WriteFile function call, so it appears my problem is related to WdfRequestRetrieveInputMemory.

Any ideas on what might this type of problem might be indicitive of? Should I just abandon this and try to get DMA working from the start? Thank you for time and help!

.