

# Re: Why is remote system information calling ioctl for my driver?

---

*Source:*

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.platbuilder/2008-06/msg00392.html>

---

- *From:* Marco <Marco@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* Tue, 17 Jun 2008 09:59:03 -0700
- 

Er... I misspoke. Changing the key name doesn't keep the System Information from probing w/ ioctls. I can't seem to duplicate the original problem, so I don't think the ioctl probing was causing the System Information util to fail as I first described. How does it decide that my driver is a serial port driver and needs to be probed like this? This is a specialized driver that has been ported from a another system, so there's no MDD/PDD concepts in it at all.

Thanks,  
Marco

"Marco" wrote:

Thanks for the excellent answer, Luca. I also found that it's keying off of the SerialPortIndex registry key to decide if the driver should be probed this way. I changed that key name to something else, and System Information is now hands off of my driver.

Marco

"Luca Calligaris" wrote:

The System Information tool calls those IOCTL's to gain information about the serial device, as it is designed to do.

IOCTL's are defined by a macro (look at windev.h):

```
#define CTL_CODE(DeviceType, Function, Method, Access) (  
((DeviceType) << 16) | ((Access) << 14) | ((Function) << 2) | (Method))
```

Your codes have DeviceType==0x1B (FILE\_DEVICE\_SERIAL\_PORT), Access==0 (FILE\_ANY\_ACCESS), and Method==0 (METHOD\_BUFFERED) (all this stuff is again defined in windev.h) which means that are specific for the serial driver. The 'functions' are:

Re: Why is remote system information calling ioctl for my driver?

0x001b0050 => Function == 0x50>>2 == 0x14  
0x001b0038 => Function == 0x38>>2 == 0x0E  
0x001b0040 => Function == 0x40>>2 == 0x10

If you look at pegdser.h you'll end up with:

0x001b0050 => IOCTL\_SERIAL\_GET\_DCB  
0x001b0038 => IOCTL\_SERIAL\_GET\_PROPERTIES  
0x001b0040 => IOCTL\_SERIAL\_GET\_TIMEOUTS

From the debug output we can see that IOCTL's fail with error 87 (ERROR\_INVALID\_PARAMETER).

In the MDD/PDD layered architecture those IOCTL\_SERIAL\_GET\_SOMETHING are basically handled in the MDD which checks the IOCTL parameters and simply returns cached data (apart for IOCTL\_SERIAL\_GET\_PROPERTIES which actually flows down to the PDD).

From the debug output I see a 'CheckIoctlParms' function call which is not in the standard MDD layer, so I suspect that your driver is not using that layer and your implementation is not checking correctly those IOCTL's parameters. Take a look at mdd.c in the standard serial MDD2 layer to see how they are handled.

--

Luca Calligaris  
www.eurotech.it

"Marco" <Marco@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> ha scritto nel messaggio [news:C23663B2-E968-4255-9BA4-B29BF363ADCA@xxxxxxxxxxxxxxxxxxxx](mailto:news:C23663B2-E968-4255-9BA4-B29BF363ADCA@xxxxxxxxxxxxxxxxxxxx)

Hi,

I'm developing a custom serial port device driver, and it seems to be working fine for the most part. I'm running into a problem wrt the remote tools -> system information feature in PB. When I run it without my driver loaded, it completes fine and I get valid system information. When I run it

Re: Why is remote system information calling ioctl for my driver?

with my driver loaded, it looks like my driver is being opened and some ioctl calls are being attempted...

```
PID:400002 TID:210000e UARTMDB: -MDB_IOControl
CheckIoctlParms failed on
code 0x001B0050, error 87
```

```
PID:400002 TID:210000e UARTMDB: -MDB_IOControl
CheckIoctlParms failed on
code 0x001B0038, error 87
```

```
PID:400002 TID:210000e UARTMDB: -MDB_IOControl
CheckIoctlParms failed on
code 0x001B0040, error 87
```

From these errors, I can tell that the ioctl codes are:

```
0x001b0050
0x001b0038
0x001b0040
```

A few questions here:

1. Is this normal behavior for the system information util to try to ioctl a driver?
2. Is there a way to disable this?
3. Where can I find out what IOCTLs these codes belong to?

Thanks,  
Marco