Re: Updated Driver for 3945ABG Intel 3945ABG Wireless LAN controller

Source: http://unix.derkeiler.com/Mailing−Lists/FreeBSD/hackers/2007−01/msg00095.html

- *From*: Benjamin Close <Benjamin.Close@xxxxxxxxxxxxxx>
- *Date*: Mon, 08 Jan 2007 00:05:53 +1030

Massimo Lusetti wrote:

> On 1/5/07, Max Laier <max@xxxxxxxxxxxxxx> wrote:
>
> > Thoughts? Volunteers?
>
> I can say that the first attempt still running fine here on my laptop
> on a −STABLE as of yesterday. I use it on a daily basis without any
> glitch. I must say i don't do or tried to do nothing special or
> network intensive job, but for reading emails, doing a lot of ssh and
> http/https the drivers is working smoothly.
>
> I will try to compile this new one on my stable during the week end
> and will see on Monday how it will perform on my office wi−lan.
>
> For the records: my wpi doesn't still work on OpenBSD−current cause
> it's integrated and the switch used to turn it on seems an acpi one
> which OpenBSD doesn't attach very well yet.

Hi Folks,
I did the port as my searching led me to believe that progress on a driver had stagnated.
Port 1: was done by Damien himself
(http://lists.freebsd.org/pipermail/freebsd−mobile/2006−July/008768.html)
Port 2: was an updated version by Daminen:
http://osdir.com/ml/freebsd−hackers@xxxxxxxxxxx/msg58154.html
which seems to be duplicated here: http://people.freebsd.org/~flz/local/wpi/
(http://www.mail−archive.com/freebsd−net@xxxxxxxxxxx/msg20710.html)
With version 2 being minor bug fixes to version 1 to try and get it working. Since then the driver has come a long way in netbsd with the FreeBSD version just getting older.

To others working on a wpi driver, please post something indicating what you've done and even a sample of what your up to so we can one build one driver that just works. I also did the driver port as everything I found failed to work on a 64bit kernel, most the time wouldn't even compile.

I'm happy to try and maintain this driver. I have the hardware.

The main bugs I know about in this driver port are:

* The use of the iee80211_amrr code
The files exist in −current in /usr/src/sys/net80211 but the required symbols are not in a GENERIC kernel build. There is a wlan_amrr module which uses this code, though I'm not sure how to make use of this, it appears a simple dependency on the module is not enough. The use of the 80211_amrr code also means that use under −stable requires quite a bit more work as Rink pointed out.

* Bus resource allocation issues
Mainly related to: "bus_dmamem_alloc failed to align memory properly."
I'm hoping Mike (http://lists.freebsd.org/pipermail/freebsd−stable/2006−June/026262.html) might be able to help resolve these.

These allocation issues are caused while allocating of the rx & tx ring buffers − which if they are broken explain why things don't work.
I'm still coming to grips to how the whole bus_dma_tag_create works. The Netbsd bus_dma_map_alloc seems much more intuitive.
I've found if the module failed the first time, unloading it, then reloading it would often make the allocation issues disappear and the driver just work.
Though I've also found sometimes that even though the allocation issues are reported, things still work, I'm guessing the alignment is pulled into line by higher levels of bus_dma.

* ifconfig down causes timeouts reseting the tx ring
I'm still chasing the cause of this. It seems non critical as the driver still works with an ifconfig up

* Watch dog timer uses an obsolete interface
I'm looking into the correct way to do watchdog timing

* A lock order reversal in wpi_intr
lock order reversal:
1st 0xffffffff812d9e20 wpi0 (network driver) @ if_wpi.c:1554
2nd 0xffffff003cee32f0 radix node head (radix node head) @ net/route.c:147
KDB: stack backtrace:
db_trace_self_wrapper() at db_trace_self_wrapper+0x3a
witness_checkorder() at witness_checkorder+0x4f9
_mtx_lock_flags() at _mtx_lock_flags+0x75
rtalloc1() at rtalloc1+0x7a
arplookup() at arplookup+0x5c
arpintr() at arpintr+0x255
ether_demux() at ether_demux+0x2d3
ether_input() at ether_input+0x1e9
ieee80211_input() at ieee80211_input+0xd05
wpi_intr() at wpi_intr+0x9d7
ithread_loop() at ithread_loop+0xfe
fork_exit() at fork_exit+0xaa
fork_trampoline() at fork_trampoline+0xe
−−− trap 0, rip = 0, rsp = 0xffffffffa0698d40, rbp = 0 −−−

For those wishing to help debug the driver and get it working, I've posted a new version
http://www.clearchain.com/~benjsc/download/20070107−wpi−freebsd.tar.gz

To help debug:

kldload wpi_ucode
kldload if_wpi
wlandebug −i wpi0 0xffffff (I've included a copy of the −current tool to save having to checkout the tree)
sysctl debug.wpi=10 (Debug levels are explained in if_wpi.c)
ifconfig wpi0 up
... wait 30 seconds then...
ifconfig wpi0 down

Then put /var/log/messages somewhere on the web & let me know

This will dump lots of debug info to syslog but should help me work out whats happening. If the driver works for you, stick to debug level 5 max, else you'll be bombarded with logs. I'll try to install FBSD 32 −current & −stable (32/64) on the laptop in the next week to see what might be different. ANY feedback is helpful.

Finally, for those who see "rx tail flags error xxx" this is caused by a 802.11 frame with an invalid CRC. Hence seeing the occasional one (with debug level 4 or higher) is quiet normal. If you see floods of them then chances are the rx ring is really out of alignment and things will just be busted.

Cheers,
Benjamin