September 21, 1999

This file is the documentation for the Raylink Wireless LAN card driver for Linux. The Raylink wireless LAN card is a PCMCIA card which provides IEEE 802.11 compatible wireless network connectivity at 1 and 2 megabits/second. See http://www.raytheon.com/micro/raylink/ for more information on the Raylink card. This driver is in early development and does have bugs. See the known bugs and limitations at the end of this document for more information. This driver also works with WebGear's Aviator 2.4 and Aviator Pro wireless LAN cards.

As of kernel 2.3.18, the ray_cs driver is part of the Linux kernel source. My web page for the development of ray_cs is at http://world.std.com/~corey/raylink.html and I can be emailed at corey@world.std.com

The kernel driver is based on ray_cs-1.62.tgz

The driver at my web page is intended to be used as an add on to David Hinds pcmcia package. All the command line parameters are available when compiled as a module. When built into the kernel, only the essid= string parameter is available via the kernel command line. This will change after the method of sorting out parameters for all the PCMCIA drivers is agreed upon. If you must have a built in driver with nondefault parameters, they can be edited in /usr/src/linux/drivers/net/pcmcia/ray_cs.c. Searching for module_param will find them all.

Information on card services is available at:
        http://pcmcia-cs.sourceforge.net/

Card services user programs are still required for PCMCIA devices. pcmcia-cs-3.1.1 or greater is required for the kernel version of the driver.

Currently, ray_cs is not part of David Hinds card services package, so the following magic is required.

At the end of the /etc/pcmcia/config.opts file, add the line:
source ./ray_cs.opts
This will make card services read the ray_cs.opts file when starting. Create the file /etc/pcmcia/ray_cs.opts containing the following:

```
#### start of /etc/pcmcia/ray_cs.opts ####################
# Configuration options for Raylink Wireless LAN PCMCIA card
device "ray_cs"
  class "network" module "misc/ray_cs"

card "RayLink PC Card WLAN Adapter"
  manfid 0x01a6, 0x0000
  bind "ray_cs"
```

```
module "misc/ray_cs" opts ""
#### end of /etc/pcmcia/ray_cs.opts ######################
```

To join an existing network with
different parameters, contact the network administrator for the
configuration information, and edit /etc/pcmcia/ray_cs.opts.
Add the parameters below between the empty quotes.

Parameters for ray_cs driver which may be specified in ray_cs.opts:

bc	integer	0 = normal mode (802.11 timing) 1 = slow down inter frame timing to allow operation with older breezecom access points.
beacon_period	integer	beacon period in Kilo-microseconds legal values = must be integer multiple of hop dwell default = 256
country	integer	1 = USA (default) 2 = Europe 3 = Japan 4 = Korea 5 = Spain 6 = France 7 = Israel 8 = Australia
essid	string	ESS ID - network name to join string with maximum length of 32 chars default value = "ADHOC_ESSID"
hop_dwell	integer	hop dwell time in Kilo-microseconds legal values = 16,32,64,128(default),256
irq_mask	integer	linux standard 16 bit value 1bit/IRQ lsb is IRQ 0, bit 1 is IRQ 1 etc. Used to restrict choice of IRQ's to use. Recommended method for controlling interrupts is in /etc/pcmcia/config.opts
net_type	integer	0 (default) = adhoc network, 1 = infrastructure
phy_addr	string	string containing new MAC address in hex, must start with x eg x00008f123456
psm	integer	0 = continuously active 1 = power save mode (not useful yet)
pc_debug	integer	(0-5) larger values for more verbose logging. Replaces ray_debug.

ray_debug	integer	Replaced with pc_debug
ray_mem_speed	integer	defaults to 500
sniffer	integer	0 = not sniffer (default) 1 = sniffer which can be used to record all network traffic using tcpdump or similar, but no normal network use is allowed.
translate	integer	0 = no translation (encapsulate frames) 1 = translation (RFC1042/802.1)

More on sniffer mode:

tcpdump does not understand 802.11 headers, so it can't interpret the contents, but it can record to a file. This is only useful for debugging 802.11 lowlevel protocols that are not visible to linux. If you want to watch ftp xfers, or do similar things, you don't need to use sniffer mode. Also, some packet types are never sent up by the card, so you will never see them (ack, rts, cts, probe etc.) There is a simple program (showcap) included in the ray_cs package which parses the 802.11 headers.

Known Problems and missing features

Does not work with non x86

Does not work with SMP

Support for defragmenting frames is not yet debugged, and in fact is known to not work. I have never encountered a net set up to fragment, but still, it should be fixed.

The ioctl support is incomplete. The hardware address cannot be set using ifconfig yet. If a different hardware address is needed, it may be set using the phy_addr parameter in ray_cs.opts. This requires a card insertion to take effect.