```c
/* ANRC RHKI */
/* Example: Register Net Device */

#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/init.h>
#include <linux/types.h>
#include <linux/socket.h>
#include <linux/in.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/sockios.h>
#include <linux/net.h>
#include <linux/inet.h>
#include <linux/if_arp.h>
#include <linux/netdevice.h>
#include <linux/skbuff.h>
#include <net/sock.h>
#include <linux/mm.h>
#include <linux/proc_fs.h>

#define DRIVER_AUTHOR "ANRC"
#define DRIVER_DESC   "Example: Add Net Device"

MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?

/* allow only 1 device instance */
static int anrc_ndevs = 1;

/* not really using this but need to declare
 * it to register net device.
 */
struct anrc_sock
{
        struct sock             sock;
};
/* not really using this but need to declare
 * it to register net device.
 */
static struct proto anrc_proto = {
        .name     = "netanrc",
        .owner    = THIS_MODULE,
        .obj_size = sizeof(struct anrc_sock),
};

static struct net_device **dev_anrc;

static int anrc_xmit(struct sk_buff *skb, struct net_device *dev)
{
        /* just free the skb, we don't care */
        dev_kfree_skb(skb);
        return 0;
}

static const struct net_device_ops anrc_net_device_ops =
{
        .ndo_start_xmit = anrc_xmit,
};

static void anrc_setup(struct net_device *dev)
{
        /* anrc is a TUNNEL device! */
        dev->type = ARPHRD_TUNNEL;
```

```
        dev->netdev_ops = &anrc_net_device_ops;
        //dev->hard_start_xmit = anrc_xmit; deprecated in kernel 2.6.31
}

static int init(void)
{
        int result;
        struct net_device *dev;
        char name[16];

        result = proto_register(&anrc_proto, 0);
        if (result != 0) goto out;
        /* kzalloc = kmalloc + memset 0x0 */
        dev_anrc = kzalloc(anrc_ndevs * sizeof(struct net_device *), GFP_KERNEL);
        if (dev_anrc == NULL)
        {
                printk(KERN_ERR "netanrc: anrc_proto_init - fail\n");
                result = -ENOMEM;
                goto out;
        }
        /* only 1 device so we just call it anrc0 */
        sprintf(name, "anrc%d", 0);
        dev = alloc_netdev(sizeof(struct net_device_stats), name, anrc_setup);
        if (!dev) {
                printk(KERN_ERR "netanrc: anrc_proto_init - mem fail\n");
                result = -ENOMEM;
                goto fail;
        }
        /* register our new device */
        result = register_netdev(dev);
        if (result) {
                printk(KERN_ERR "netanrc: netdevice registration failed\n");
                free_netdev(dev);
                goto fail;
        }
        dev_anrc[0] = dev;
out:
        return result;
fail:
        unregister_netdev(dev_anrc[0]);
        free_netdev(dev_anrc[0]);
        kfree(dev_anrc);

        return 0;
}

static void exit(void)
{
        struct net_device *dev = dev_anrc[0];

        /* undo everything we did to install net device */
        //proc_net_remove("netanrc");
        unregister_netdev(dev);
        free_netdev(dev);
        kfree(dev_anrc);
        proto_unregister(&anrc_proto);

}

module_init(init);
module_exit(exit);
```