

RE: Invalid module format?

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-05/msg05298.html>

- *From:* "Brian D. McGrew" <brian@xxxxxxxxxxxxxx>
 - *Date:* Fri, 19 May 2006 11:38:23 -0700
-

Seems my failure to post the complete set of files has caused great confusion. Is that a sign of stress or frustration or both???

Here is a complete listing of all the files, including the Makefile. I think this is everything now.

I'm facing two problems. First, the 2.6.16 kernel won't load this driver; throws an invalid module format as where 2.6.15 loads it just fine.

Secondly, the driver is supposed to allocate kernel memory that's accessible from userspace. If you see the function `alloc_ibt_user_shared(IbtSoftDev *)` and the function `alloc_ibt_image_table_mem(IbtSoftDev *, int)`; you'll see where I've a couple of attempts at calling `__get_free_pages` as well as a shot at `vmalloc`. I've narrowed it down to this one section that's killing me.

If anyone can offer help, that would be great because I'm just stuck on this and have been for six months!!!

Thanks!!

MAKEFILE ---

```
obj-m += ibb.o
obj-m += ibb3d.o
obj-m += mvp_rtc.o
```

---[SOURCES BEGING HERE]---

---[BEGIN IBB]---

-->

```
/usr/src/redhat/BUILD/kernel-2.6.16/linux-2.6.16.16/drivers/mvp/ibt.c
```

```
static const char *ibt_c = "$Id: ibt.c,v 1.5 2006/04/18 19:48:02 brian
Exp brian $(c) MVP ";
```

```
#include <asm/uaccess.h>
```

RE: Invalid module format?

```
#include <linux/config.h>
#include <linux/fs.h>
#include <linux/interrupt.h>
#include <asm/io.h>
#include <linux/mm.h>
#include <linux/module.h>
#include <linux/pci.h>
#include <linux/slab.h>
#include <linux/types.h>

#include "dev/ibb.h"
#include "ibbsoft.h"

#ifdef __KERNEL__
#define __KERNEL__
#endif

#ifdef MODULE
#define MODULE
#endif

#ifdef CONFIG_PCI
#error "This driver REQUIRES PCI support!"
#endif

#ifdef CONFIG_MODVERSIONS && !defined(MODVERSIONS)
#define MODVERSIONS
#endif

void ibb_rtc_wakeup(void);
irqreturn_t ibb_intr(int irq, void *dev_id, struct pt_regs *regs);

EXPORT_SYMBOL(ibb_rtc_wakeup);

/*
 * Split minors in two parts
 */
#define TYPE(dev) (MINOR(dev) >> 4) /* high nibble */
#define NUM(dev) (MINOR(dev) & 0xf) /* low nibble */

#define DEBUG

#ifdef DEBUG
#define debug_out(msg) { printk msg; }
#else
#define debug_out(msg)
#endif

/*
 * Place the call to START at the beginning of the function and place the
 call
```

RE: Invalid module format?

RE: Invalid module format?

```
* to END where the function __should__ be returning at (not always at
the
* end of the function, get it)!
*/

#define S u16 device_id;

int result;
int dev_num;
int rtc_major;
int intr_handler_req;

if (pci_enable_device(pdev)) {
return(-ENODEV);
}

/* stealing code from cciss.c – thanks compaq! */

debug_out(("rtc_probe: RTC 0x%08x found @bus %d dev %d func %d\n",
pdev->device,
pdev->bus->number,
PCI_SLOT(pdev->devfn),
PCI_FUNC(pdev->devfn)));

/* alloc our per-device data */
dev_num = alloc_rtc_soft_state();

if (dev_num < 0) {
/* I've already printk'd error message */
return(-ENOMEM);
}

result = 0;

memset(RtcSoft[dev_num], 0, sizeof(RtcSoft[dev_num]));
rtc_sp = RtcSoft[dev_num];

sprintf(rtc_sp->devname, "mvp_rtc");

spin_lock_init(&rtc_sp->mutex);
rtc_sp->dev_num = dev_num;
rtc_sp->pdev = pdev;

sema_init(&rtc_sp->sem, 1);
pci_set_drvdata(pdev,rtc_sp); /* we'll need this in remove: */

debug_out(("rtc_probe: device %d\n", dev_num));

#define REQUESTMEM
#ifdef REQUESTMEM
/*
```

RE: Invalid module format?

RE: Invalid module format?

```
* map the card memory areas into kernel space
* and store the address in our RtcDev info
*/

rtc_sp->iobase = pci_resource_start(pdev, 0);
rtc_sp->iosize = pci_resource_len(pdev, 0);

debug_out(("rtc_probe_module: base start %#010lx size %#010lx\n",
rtc_sp->iobase,
rtc_sp->iosize));

if (rtc_do_all_mappings(rtc_sp) < 0) {
debug_out(("rtc_probe: do_all_mappings failed\n"));

free_rtc_soft_state(rtc_sp->dev_num);
return(-ENODEV);
}
#endif

#ifdef CONFIG_DEVFS_FS
rtc_sp->rtc_devfs_dir = devfs_mk_dir(NULL, rtc_sp->devname, NULL);

if (!rtc_sp->rtc_devfs_dir) {
debug_out(("rtc_probe(%d): can't register devfs\n", dev_num));

free_all_mappings(rtc_sp);
free_rtc_soft_state(rtc_sp->dev_num);

return(-EBUSY);
}

devfs_register(rtc_sp->rtc_devfs_dir,
rtc_sp->devname,
DEVFS_FL_AUTO_DEVNUM, /* autoallocate
major/minor */
0, /* ignored because of
flag */
0, /* ignored because of
flag */
S_IFCHR | S_IRUGO | S_IWUGO, /* ??? */
&rtc_fops,
rtc_sp);

#else

rtc_major = 0;
result = register_chrdev(rtc_major, rtc_sp->devname, &rtc_fops);

if (result < 0) {
debug_out(("rtc_probe_module: can't get major %d\n",
rtc_major));

```

RE: Invalid module format?

RE: Invalid module format?

```
free_all_mappings(rtc_sp);
free_rtc_soft_state(rtc_sp->dev_num);

return(-EBUSY);
}

if (rtc_major == 0) {
rtc_major = result;
}

debug_out(("rtc_probe_module: got major %d\n", rtc_major));

#endif

rtc_sp->rtc_major = rtc_major;

pci_read_config_word(pdev, 2, &device_id);
pci_read_config_byte(pdev, 8, &rev_id);

if (rev_id > 0 && rev_id < 27) {
rev_id += 0x40;
revid = rev_id;
}

if (!pdev->irq) {
debug_out(("rtc_probe_module: can't get interrupt number\n"));

free_all_mappings(rtc_sp);
free_rtc_soft_state(rtc_sp->dev_num);
return(-ENODEV);
}

rtc_sp->myint = pdev->irq;

rtc_sp->irq_cnt = 0;
intr_handler_req = request_irq(rtc_sp->myint,rtc_intr,
SA_SHIRQ,
rtc_sp->devname,
rtc_sp);

if (intr_handler_req != 0) {
debug_out(("rtc_probe_module: can't req interrupt handler\n"));

free_all_mappings(rtc_sp);
free_rtc_soft_state(rtc_sp->dev_num);
return(-ENODEV);
}
```

RE: Invalid module format?