Subject: [PATCH 1/5] fixing errors handling during pci_driver resume stage [net]
Posted by Dmitriy Monakhov on Tue, 09 Jan 2007 09:01:14 GMT
View Forum Message <> Reply to Message

network pci drivers have to return correct error code during resume stage in
case of errors.
Signed-off-by: Dmitriy Monakhov <dmonakhov@openvz.org>

```
diff --git a/Makefile b/Makefile
diff --git a/drivers/net/3c59x.c b/drivers/net/3c59x.c
index 80bdcf8..ed06e48 100644
--- a/drivers/net/3c59x.c
+++ b/drivers/net/3c59x.c
@@ -823,11 +823,18 @@ static int vortex_resume(struct pci_dev
 {
  struct net_device *dev = pci_get_drvdata(pdev);
  struct vortex_private *vp = netdev_priv(dev);
+ int err;

  if (dev && vp) {
   pci_set_power_state(pdev, PCI_D0);
   pci_restore_state(pdev);
-  pci_enable_device(pdev);
+  err = pci_enable_device(pdev);
+  if (err) {
+   dev_err(&pdev->dev, "Cannot enable PCI device, "
+    " aborting.\n");
+   return err;
+  }
+
   pci_set_master(pdev);
   if (request_irq(dev->irq, vp->full_bus_master_rx ?
     &boomerang_interrupt : &vortex_interrupt, IRQF_SHARED, dev->name, dev)) {
diff --git a/drivers/net/b44.c b/drivers/net/b44.c
index 5eb2ec6..a88b346 100644
--- a/drivers/net/b44.c
+++ b/drivers/net/b44.c
@@ -2315,16 +2315,25 @@ static int b44_resume(struct pci_dev *pd
 {
  struct net_device *dev = pci_get_drvdata(pdev);
  struct b44 *bp = netdev_priv(dev);
+ int err;

  pci_restore_state(pdev);
- pci_enable_device(pdev);
+ err = pci_enable_device(pdev);
+ if (err) {
```

```
+  dev_err(&pdev->dev, "Cannot enable PCI device, aborting.\n");
+  return err;
+ }
+
  pci_set_master(pdev);

  if (!netif_running(dev))
   return 0;

- if (request_irq(dev->irq, b44_interrupt, IRQF_SHARED, dev->name, dev))
+ if (request_irq(dev->irq, b44_interrupt, IRQF_SHARED, dev->name, dev)) {
   printk(KERN_ERR PFX "%s: request_irq failed\n", dev->name);
+  pci_disable_device(pdev);
+  return -EBUSY;
+ }

  spin_lock_irq(&bp->lock);

diff --git a/drivers/net/eepro100.c b/drivers/net/eepro100.c
index e28bb1e..698f974 100644
--- a/drivers/net/eepro100.c
+++ b/drivers/net/eepro100.c
@@ -2292,10 +2292,16 @@ static int eepro100_resume(struct pci_de
  struct net_device *dev = pci_get_drvdata (pdev);
  struct speedo_private *sp = netdev_priv(dev);
  void __iomem *ioaddr = sp->regs;
+ int err;

  pci_set_power_state(pdev, PCI_D0);
  pci_restore_state(pdev);
- pci_enable_device(pdev);
+ err = pci_enable_device(pdev);
+ if (err) {
+  dev_err(&pdev->dev, "Cannot enable PCI device, aborting.\n");
+  return err;
+ }
+
  pci_set_master(pdev);

  if (!netif_running(dev))
diff --git a/drivers/net/natsemi.c b/drivers/net/natsemi.c
index ffa0afd..57a8f36 100644
--- a/drivers/net/natsemi.c
+++ b/drivers/net/natsemi.c
@@ -3195,13 +3195,20 @@ static int natsemi_resume (struct pci_de
 {
  struct net_device *dev = pci_get_drvdata (pdev);
  struct netdev_private *np = netdev_priv(dev);
```

```
+ int err;

  rtnl_lock();
  if (netif_device_present(dev))
   goto out;
  if (netif_running(dev)) {
   BUG_ON(!np->hands_off);
-  pci_enable_device(pdev);
+  err = pci_enable_device(pdev);
+  if (err) {
+   dev_err(&pdev->dev, "Cannot enable PCI device, "
+    "aborting.\n");
+   return err;
+  }
+
  /* pci_power_on(pdev); */

   natsemi_reset(dev);
diff --git a/drivers/net/ne2k-pci.c b/drivers/net/ne2k-pci.c
index 589785d..e8aea03 100644
--- a/drivers/net/ne2k-pci.c
+++ b/drivers/net/ne2k-pci.c
@@ -670,10 +670,15 @@ static int ne2k_pci_suspend (struct pci_
 static int ne2k_pci_resume (struct pci_dev *pdev)
 {
  struct net_device *dev = pci_get_drvdata (pdev);
+ int err;

  pci_set_power_state(pdev, 0);
  pci_restore_state(pdev);
- pci_enable_device(pdev);
+ err = pci_enable_device(pdev);
+ if (err) {
+  dev_err(&pdev->dev, "Cannot enable PCI device, aborting.\n");
+  return err;
+ }
  NS8390_init(dev, 1);
  netif_device_attach(dev);

diff --git a/drivers/net/sk98lin/skge.c b/drivers/net/sk98lin/skge.c
index 92d11b9..c9de11f 100644
--- a/drivers/net/sk98lin/skge.c
+++ b/drivers/net/sk98lin/skge.c
@@ -5125,7 +5125,12 @@ static int skge_resume(struct pci_dev *p

  pci_set_power_state(pdev, PCI_D0);
  pci_restore_state(pdev);
- pci_enable_device(pdev);
```

```
+ ret = pci_enable_device(pdev);
+ if (ret) {
+  dev_err(&pdev->dev, "Cannot enable PCI device, aborting.\n");
+   return ret;
+ }
+
  pci_set_master(pdev);
  if (pAC->GIni.GIMacsFound == 2)
   ret = request_irq(dev->irq, SkGeIsr, IRQF_SHARED, "sk98lin", dev);
diff --git a/drivers/net/tulip/xircom_tulip_cb.c b/drivers/net/tulip/xircom_tulip_cb.c
index a998c5d..91fb067 100644
--- a/drivers/net/tulip/xircom_tulip_cb.c
+++ b/drivers/net/tulip/xircom_tulip_cb.c
@@ -1655,10 +1655,16 @@ static int xircom_resume(struct pci_dev
 {
  struct net_device *dev = pci_get_drvdata(pdev);
  struct xircom_private *tp = netdev_priv(dev);
+ int err;
  printk(KERN_INFO "xircom_resume(%s)\n", dev->name);

  pci_set_power_state(pdev,0);
- pci_enable_device(pdev);
+ err = pci_enable_device(pdev);
+ if (err) {
+  dev_err(&pdev->dev, "Cannot enable PCI device, aborting.\n");
+  return err;
+ }
+
  pci_restore_state(pdev);

  /* Bring the chip out of sleep mode.
```