

[git patches] net driver updates for .26

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-05/msg05728.html>

- *From:* Jeff Garzik <jeff@xxxxxxxxxx>
 - *Date:* Tue, 13 May 2008 01:56:23 -0400
-

Notes:

– lotsa LOC change in sfc, but it's a new driver, so no possible regressions

– we add some non-net exports required for ehea (drivers/base/memory.c)

Please pull from 'upstream-davem' branch of
master.kernel.org:/pub/scm/linux/kernel/git/jgarzik/netdev-2.6.git upstream-davem

to receive the following updates:

```
MAINTAINERS | 6 +-
drivers/base/memory.c | 2 +
drivers/net/Kconfig | 2 +-
drivers/net/atlx/atlx.c | 157 ++++--
drivers/net/atlx/atlx.h | 2 +-
drivers/net/atlx/atlx.c | 2 +-
drivers/net/atlx/atlx.h | 7 +-
drivers/net/cxgb3/adapter.h | 1 +
drivers/net/cxgb3/common.h | 1 +
drivers/net/cxgb3/cxgb3_main.c | 40 +-
drivers/net/cxgb3/regs.h | 8 +
drivers/net/cxgb3/sge.c | 29 +-
drivers/net/cxgb3/t3_hw.c | 28 +
drivers/net/dm9000.c | 37 +-
drivers/net/ehea/ehea.h | 27 +-
drivers/net/ehea/ehea_main.c | 25 +
drivers/net/ehea/ehea_qmr.c | 286 ++++++----
drivers/net/gianfar.c | 2 +
drivers/net/myri10ge/myri10ge.c | 730 ++++++-----
drivers/net/myri10ge/myri10ge_mcp.h | 56 +-
drivers/net/myri10ge/myri10ge_mcp_gen_header.h | 39 +-
drivers/net/sfc/Makefile | 4 +-
drivers/net/sfc/boards.h | 2 +
drivers/net/sfc/efx.c | 4 +-
drivers/net/sfc/enum.h | 49 ++
drivers/net/sfc/ethtool.c | 259 ++++++--
```

[git patches] net driver updates for .26

drivers/net/sfc/falcon.c | 8 +-
drivers/net/sfc/falcon_hwdefs.h | 16 +-
drivers/net/sfc/falcon_xmac.c | 82 +++-
drivers/net/sfc/mdio_10g.c | 78 +++
drivers/net/sfc/mdio_10g.h | 24 +-
drivers/net/sfc/net_driver.h | 28 +-
drivers/net/sfc/rx.c | 11 +-
drivers/net/sfc/selftest.c | 717 ++++++
drivers/net/sfc/selftest.h | 50 ++
drivers/net/sfc/sfe4001.c | 14 +
drivers/net/sfc/tenxpress.c | 91 +++-
drivers/net/sfc/tx.c | 664 ++++++
drivers/net/sfc/xfp_phy.c | 36 ++
drivers/net/sky2.h | 4 +-
40 files changed, 3081 insertions(+), 547 deletions(-)
create mode 100644 drivers/net/sfc/selftest.c
create mode 100644 drivers/net/sfc/selftest.h

Andy Fleming (1):

gianfar: Fix a bug where the pointer never moves for dma_unmap...

Auke Kok (1):

New maintainer for Intel ethernet adapters

Ben Dooks (1):

DM9000: Use delayed work to update MII PHY state

Ben Hutchings (9):

[netdrv] sfc: Add TSO support

[netdrv] sfc: Add phy_flash_cfg module parameter and implementation

[netdrv] sfc: Removed bogus 'fall-thru' comments

[netdrv] sfc: Remove garbage from comment

[netdrv] sfc: Remove kernel-doc comments for removed members of struct efx_nic

[netdrv] sfc: Fix code formatting

[netdrv] sfc: Remove unused macro EFX_XAUI_RETRAIN_MAX

[netdrv] sfc: Increment rx_reset when reported as driver event

[netdrv] sfc: sfc: Add self-test support

Brice Goglin (11):

myri10ge: update firmware headers

myri10ge: fix module parameter descriptions

myri10ge: increase and fix handoff timeout

myri10ge: properly align scratch buffers

myri10ge: report FIBER in ethtool for XFP based NIC

myri10ge: add barrier in myri10ge_send_cmd

myri10ge: trivial formatting fix

myri10ge: fix potential infinite loop in enable_ecrc

myri10ge: move data structures into a single slice

myri10ge: cleanup retrieving of firmware capabilities

myri10ge: fix the number of interrupt slots

Divy Le Ray (2):
cxgb3 – fix port up/down error path
cxgb3 – fix EEH

Enrico Scholz (2):
DM9000: Add __devinit and __devexit attributes to probe and remove
DM9000: Update and fix driver debugging messages

Hannes Hering (3):
memory: Introduce exports for memory notifiers
ehea: Add dependency to Kconfig
ehea: Add DLPAR memory remove support

Jay Cliburn (4):
atl1: add PHY power save mode
atl1: fix broken suspend and resume
atl1: add shutdown callback
atl1: bump version number

Jesse Brandeburg (1):
sky2: fix simple define thinko

```
diff --git a/MAINTAINERS b/MAINTAINERS
index c3a533d..0cc47b9 100644
--- a/MAINTAINERS
+++ b/MAINTAINERS
@@ -2104,12 +2104,10 @@ L: netdev@xxxxxxxxxxxxxxxxxxx
S: Maintained
```

INTEL ETHERNET DRIVERS (e100/e1000/e1000e/igb/ixgb/ixgbe)

-P: Auke Kok
-M: auke-jan.h.kok@xxxxxxxxxx
-P: Jesse Brandeburg
-M: jesse.brandeburg@xxxxxxxxxx
P: Jeff Kirsher
M: jeffrey.t.kirsher@xxxxxxxxxx
+P: Jesse Brandeburg
+M: jesse.brandeburg@xxxxxxxxxx
P: Bruce Allan
M: bruce.w.allan@xxxxxxxxxx
P: John Ronciak

```
diff --git a/drivers/base/memory.c b/drivers/base/memory.c
index 8ce6de5..937e825 100644
--- a/drivers/base/memory.c
+++ b/drivers/base/memory.c
@@ -53,11 +53,13 @@ int register_memory_notifier(struct notifier_block *nb)
{
return blocking_notifier_chain_register(&memory_chain, nb);
}
+EXPORT_SYMBOL(register_memory_notifier);
```

[git patches] net driver updates for .26

```
void unregister_memory_notifier(struct notifier_block *nb)
{
blocking_notifier_chain_unregister(&memory_chain, nb);
}
+EXPORT_SYMBOL(unregister_memory_notifier);

/*
 * register_memory – Setup a sysfs device for a memory block
diff --git a/drivers/net/Kconfig b/drivers/net/Kconfig
index d27f54a..9f6cc8a 100644
--- a/drivers/net/Kconfig
+++ b/drivers/net/Kconfig
@@ -2426,7 +2426,7 @@ config CHELSIO_T3

config EHEA
tristate "eHEA Ethernet support"
- depends on IBMEBUS && INET && SPARSEMEM
+ depends on IBMEBUS && INET && SPARSEMEM && MEMORY_HOTPLUG
select INET_LRO
---help---
This driver supports the IBM pSeries eHEA ethernet adapter.
diff --git a/drivers/net/atlx/atl1.c b/drivers/net/atlx/atl1.c
index 0afe522..9c2394d 100644
--- a/drivers/net/atlx/atl1.c
+++ b/drivers/net/atlx/atl1.c
@@ -1,7 +1,7 @@
/*
 * Copyright(c) 2005 – 2006 Attansic Corporation. All rights reserved.
 * Copyright(c) 2006 – 2007 Chris Snook <csnook@xxxxxxxxxx>
- * Copyright(c) 2006 Jay Cliburn <jcliburn@xxxxxxxxxx>
+ * Copyright(c) 2006 – 2008 Jay Cliburn <jcliburn@xxxxxxxxxx>
 *
 * Derived from Intel e1000 driver
 * Copyright(c) 1999 – 2005 Intel Corporation. All rights reserved.
@@ -36,7 +36,6 @@
 * A very incomplete list of things that need to be dealt with:
 *
 * TODO:
- * Wake on LAN.
 * Add more ethtool functions.
 * Fix abstruse irq enable/disable condition described here:
 * http://marc.theaimsgroup.com/?l=linux-netdev&m=116398508500553&w=2
@@ -638,21 +637,18 @@ static s32 atl1_phy_leave_power_saving(struct atl1_hw *hw)
}

/*
- *TODO: do something or get rid of this
+ * Force the PHY into power saving mode using vendor magic.
 */
#ifdef CONFIG_PM
-static s32 atl1_phy_enter_power_saving(struct atl1_hw *hw)
```

[git patches] net driver updates for .26

```
+static void atl1_phy_enter_power_saving(struct atl1_hw *hw)
{
-/* s32 ret_val;
- * u16 phy_data;
- */
+atl1_write_phy_reg(hw, MII_DBG_ADDR, 0);
+atl1_write_phy_reg(hw, MII_DBG_DATA, 0x124E);
+atl1_write_phy_reg(hw, MII_DBG_ADDR, 2);
+atl1_write_phy_reg(hw, MII_DBG_DATA, 0x3000);
+atl1_write_phy_reg(hw, MII_DBG_ADDR, 3);
+atl1_write_phy_reg(hw, MII_DBG_DATA, 0);

-/*
- ret_val = atl1_write_phy_reg(hw, ...);
- ret_val = atl1_write_phy_reg(hw, ...);
- ....
-*/
- return 0;
}
#endif

@@ -2784,64 +2780,93 @@ static int atl1_suspend(struct pci_dev *pdev, pm_message_t state)
struct atl1_hw *hw = &adapter->hw;
u32 ctrl = 0;
u32 wufc = adapter->wol;
+ u32 val;
+ int retval;
+ u16 speed;
+ u16 duplex;

netif_device_detach(netdev);
if (netif_running(netdev))
atl1_down(adapter);

+ retval = pci_save_state(pdev);
+ if (retval)
+ return retval;
+
atl1_read_phy_reg(hw, MII_BMSR, (u16 *) & ctrl);
atl1_read_phy_reg(hw, MII_BMSR, (u16 *) & ctrl);
- if (ctrl & BMSR_LSTATUS)
+ val = ctrl & BMSR_LSTATUS;
+ if (val)
wufc &= ~ATLX_WUFC_LNKC;

- /* reduce speed to 10/100M */
- if (wufc) {
- atl1_phy_enter_power_saving(hw);
- /* if resume, let driver to re- setup link */
- hw->phy_configured = false;
- atl1_set_mac_addr(hw);
```

```

- atlx_set_multi(netdev);
+ if (val && wufc) {
+ val = atl1_get_speed_and_duplex(hw, &speed, &duplex);
+ if (val) {
+ if (netif_msg_ifdown(adapter))
+ dev_printk(KERN_DEBUG, &pdev->dev,
+ "error getting speed/duplex\n");
+ goto disable_wol;
+ }

ctrl = 0;
- /* turn on magic packet wol */
- if (wufc & AT LX_WUFC_MAG)
- ctrl = WOL_MAGIC_EN | WOL_MAGIC_PME_EN;

- /* turn on Link change WOL */
- if (wufc & AT LX_WUFC_LNKC)
- ctrl |= (WOL_LINK_CHG_EN | WOL_LINK_CHG_PME_EN);
+ /* enable magic packet WOL */
+ if (wufc & AT LX_WUFC_MAG)
+ ctrl |= (WOL_MAGIC_EN | WOL_MAGIC_PME_EN);
iowrite32(ctrl, hw->hw_addr + REG_WOL_CTRL);
-
- /* turn on all-multi mode if wake on multicast is enabled */
- ctrl = ioread32(hw->hw_addr + REG_MAC_CTRL);
- ctrl &= ~MAC_CTRL_DBG;
- ctrl &= ~MAC_CTRL_PROMIS_EN;
- if (wufc & AT LX_WUFC_MC)
- ctrl |= MAC_CTRL_MC_ALL_EN;
- else
- ctrl &= ~MAC_CTRL_MC_ALL_EN;
-
- /* turn on broadcast mode if wake on-BC is enabled */
- if (wufc & AT LX_WUFC_BC)
+ ioread32(hw->hw_addr + REG_WOL_CTRL);
+
+ /* configure the mac */
+ ctrl = MAC_CTRL_RX_EN;
+ ctrl |= ((u32)((speed == SPEED_1000) ? MAC_CTRL_SPEED_1000 :
+ MAC_CTRL_SPEED_10_100) << MAC_CTRL_SPEED_SHIFT);
+ if (duplex == FULL_DUPLEX)
+ ctrl |= MAC_CTRL_DUPLEX;
+ ctrl |= (((u32)adapter->hw.preamble_len &
+ MAC_CTRL_PRMLLEN_MASK) << MAC_CTRL_PRMLLEN_SHIFT);
+ if (adapter->vlgrp)
+ ctrl |= MAC_CTRL_RMV_VLAN;
+ if (wufc & AT LX_WUFC_MAG)
ctrl |= MAC_CTRL_BC_EN;
- else
- ctrl &= ~MAC_CTRL_BC_EN;
-

```

```

- /* enable RX */
- ctrl |= MAC_CTRL_RX_EN;
iowrite32(ctrl, hw->hw_addr + REG_MAC_CTRL);
- pci_enable_wake(pdev, PCI_D3hot, 1);
- pci_enable_wake(pdev, PCI_D3cold, 1);
- } else {
- iowrite32(0, hw->hw_addr + REG_WOL_CTRL);
- pci_enable_wake(pdev, PCI_D3hot, 0);
- pci_enable_wake(pdev, PCI_D3cold, 0);
+ ioread32(hw->hw_addr + REG_MAC_CTRL);
+
+ /* poke the PHY */
+ ctrl = ioread32(hw->hw_addr + REG_PCIE_PHYMISC);
+ ctrl |= PCIE_PHYMISC_FORCE_RCV_DET;
+ iowrite32(ctrl, hw->hw_addr + REG_PCIE_PHYMISC);
+ ioread32(hw->hw_addr + REG_PCIE_PHYMISC);
+
+ pci_enable_wake(pdev, pci_choose_state(pdev, state), 1);
+ goto exit;
}

- pci_save_state(pdev);
+ if (!val && wufc) {
+ ctrl |= (WOL_LINK_CHG_EN | WOL_LINK_CHG_PME_EN);
+ iowrite32(ctrl, hw->hw_addr + REG_WOL_CTRL);
+ ioread32(hw->hw_addr + REG_WOL_CTRL);
+ iowrite32(0, hw->hw_addr + REG_MAC_CTRL);
+ ioread32(hw->hw_addr + REG_MAC_CTRL);
+ hw->phy_configured = false;
+ pci_enable_wake(pdev, pci_choose_state(pdev, state), 1);
+ goto exit;
+ }
+
+disable_wol:
+ iowrite32(0, hw->hw_addr + REG_WOL_CTRL);
+ ioread32(hw->hw_addr + REG_WOL_CTRL);
+ ctrl = ioread32(hw->hw_addr + REG_PCIE_PHYMISC);
+ ctrl |= PCIE_PHYMISC_FORCE_RCV_DET;
+ iowrite32(ctrl, hw->hw_addr + REG_PCIE_PHYMISC);
+ ioread32(hw->hw_addr + REG_PCIE_PHYMISC);
+ atl1_phy_enter_power_saving(hw);
+ hw->phy_configured = false;
+ pci_enable_wake(pdev, pci_choose_state(pdev, state), 0);
+exit:
+ if (netif_running(netdev))
+ pci_disable_msi(adapter->pdev);
pci_disable_device(pdev);
-
- pci_set_power_state(pdev, PCI_D3hot);
+ pci_set_power_state(pdev, pci_choose_state(pdev, state));

```

```

return 0;
}
@@ -2855,20 +2880,26 @@ static int atl1_resume(struct pci_dev *pdev)
pci_set_power_state(pdev, PCI_D0);
pci_restore_state(pdev);

- /* FIXME: check and handle */
err = pci_enable_device(pdev);
+ if (err) {
+ if (netif_msg_ifup(adapter))
+ dev_printk(KERN_DEBUG, &pdev->dev,
+ "error enabling pci device\n");
+ return err;
+ }
+
+ pci_set_master(pdev);
+ iowrite32(0, adapter->hw.hw_addr + REG_WOL_CTRL);
pci_enable_wake(pdev, PCI_D3hot, 0);
pci_enable_wake(pdev, PCI_D3cold, 0);

- iowrite32(0, adapter->hw.hw_addr + REG_WOL_CTRL);
- atl1_reset(adapter);
+ atl1_reset_hw(&adapter->hw);
+ adapter->cmb.cmb->int_stats = 0;

if (netif_running(netdev))
atl1_up(adapter);
netif_device_attach(netdev);

- atl1_via_workaround(adapter);
-
return 0;
}
#else
@@ -2876,6 +2907,13 @@ static int atl1_resume(struct pci_dev *pdev)
#define atl1_resume NULL
#endif

+static void atl1_shutdown(struct pci_dev *pdev)
+{
+#ifdef CONFIG_PM
+ atl1_suspend(pdev, PMSG_SUSPEND);
+#endif
+}
+
+#ifdef CONFIG_NET_POLL_CONTROLLER
static void atl1_poll_controller(struct net_device *netdev)
{
@@ -3122,7 +3160,8 @@ static struct pci_driver atl1_driver = {
.probe = atl1_probe,
.remove = __devexit_p(atl1_remove),

```

```
.suspend = atl1_suspend,  
- .resume = atl1_resume  
+ .resume = atl1_resume,  
+ .shutdown = atl1_shutdown  
};  
  
/*  
diff --git a/drivers/net/atlx/atl1.h b/drivers/net/atlx/atl1.h  
index 51893d6..a5015b1 100644  
--- a/drivers/net/atlx/atl1.h  
+++ b/drivers/net/atlx/atl1.h  
@@ -1,7 +1,7 @@  
/*  
* Copyright(c) 2005 – 2006 Attansic Corporation. All rights reserved.  
* Copyright(c) 2006 – 2007 Chris Snook <csnook@xxxxxxxxxxx>  
- * Copyright(c) 2006 Jay Cliburn <jcliburn@xxxxxxxxxxx>  
+ * Copyright(c) 2006 – 2008 Jay Cliburn <jcliburn@xxxxxxxxxxx>  
*  
* Derived from Intel e1000 driver  
* Copyright(c) 1999 – 2005 Intel Corporation. All rights reserved.  
diff --git a/drivers/net/atlx/atlx.c b/drivers/net/atlx/atlx.c  
index f06b854..b3e7fcf 100644  
--- a/drivers/net/atlx/atlx.c  
+++ b/drivers/net/atlx/atlx.c  
@@ -2,7 +2,7 @@  
*  
* Copyright(c) 2005 – 2006 Attansic Corporation. All rights reserved.  
* Copyright(c) 2006 – 2007 Chris Snook <csnook@xxxxxxxxxxx>  
- * Copyright(c) 2006 Jay Cliburn <jcliburn@xxxxxxxxxxx>  
+ * Copyright(c) 2006 – 2008 Jay Cliburn <jcliburn@xxxxxxxxxxx>  
* Copyright(c) 2007 Atheros Corporation. All rights reserved.  
*  
* Derived from Intel e1000 driver  
diff --git a/drivers/net/atlx/atlx.h b/drivers/net/atlx/atlx.h  
index 3be7c09..297a03d 100644  
--- a/drivers/net/atlx/atlx.h  
+++ b/drivers/net/atlx/atlx.h  
@@ -2,7 +2,7 @@  
*  
* Copyright(c) 2005 – 2006 Attansic Corporation. All rights reserved.  
* Copyright(c) 2006 – 2007 Chris Snook <csnook@xxxxxxxxxxx>  
- * Copyright(c) 2006 Jay Cliburn <jcliburn@xxxxxxxxxxx>  
+ * Copyright(c) 2006 – 2008 Jay Cliburn <jcliburn@xxxxxxxxxxx>  
* Copyright(c) 2007 Atheros Corporation. All rights reserved.  
*  
* Derived from Intel e1000 driver  
@@ -29,7 +29,7 @@  
#include <linux/module.h>  
#include <linux/types.h>  
  
-#define ATLX_DRIVER_VERSION "2.1.1"
```

[git patches] net driver updates for .26

```
+ #define ATLX_DRIVER_VERSION "2.1.3"
MODULE_AUTHOR("Xiong Huang <xiong.huang@xxxxxxxxxxxx>, \
Chris Snook <csnook@xxxxxxxxxxxx>, Jay Cliburn <jcliburn@xxxxxxxxxxxx>");
MODULE_LICENSE("GPL");
@@ -460,6 +460,9 @@ MODULE_VERSION(ATLX_DRIVER_VERSION);
# define MII_ATLX_PSSR_100MBS 0x4000 /* 01=100Mbs */
# define MII_ATLX_PSSR_1000MBS 0x8000 /* 10=1000Mbs */

+ #define MII_DBG_ADDR 0x1D
+ #define MII_DBG_DATA 0x1E
+
/* PCI Command Register Bit Definitions */
# define PCI_REG_COMMAND 0x04 /* PCI Command Register */
# define CMD_IO_SPACE 0x0001
diff --git a/drivers/net/cxgb3/adapter.h b/drivers/net/cxgb3/adapter.h
index 4fdb13f..acebe43 100644
--- a/drivers/net/cxgb3/adapter.h
+++ b/drivers/net/cxgb3/adapter.h
@@ -71,6 +71,7 @@ enum { /* adapter flags */
USING_MSIX = (1 << 2),
QUEUES_BOUND = (1 << 3),
TP_PARITY_INIT = (1 << 4),
+ NAPI_INIT = (1 << 5),
};

struct fl_pg_chunk {
diff --git a/drivers/net/cxgb3/common.h b/drivers/net/cxgb3/common.h
index 91ee727..579bee4 100644
--- a/drivers/net/cxgb3/common.h
+++ b/drivers/net/cxgb3/common.h
@@ -698,6 +698,7 @@ void mac_prep(struct cmac *mac, struct adapter *adapter, int index);
void early_hw_init(struct adapter *adapter, const struct adapter_info *ai);
int t3_prep_adapter(struct adapter *adapter, const struct adapter_info *ai,
int reset);
+int t3_replay_prep_adapter(struct adapter *adapter);
void t3_led_ready(struct adapter *adapter);
void t3_fatal_err(struct adapter *adapter);
void t3_set_vlan_accel(struct adapter *adapter, unsigned int ports, int on);
diff --git a/drivers/net/cxgb3/cxgb3_main.c b/drivers/net/cxgb3/cxgb3_main.c
index ce949d5..3a31272 100644
--- a/drivers/net/cxgb3/cxgb3_main.c
+++ b/drivers/net/cxgb3/cxgb3_main.c
@@ -421,6 +421,13 @@ static void init_napi(struct adapter *adap)
netif_napi_add(qs->netdev, &qs->napi, qs->napi.poll,
64);
}
+
+ /*
+ * netif_napi_add() can be called only once per napi_struct because it
+ * adds each new napi_struct to a list. Be careful not to call it a
+ * second time, e.g., during EEH recovery, by making a note of it.

```

[git patches] net driver updates for .26

```
+ */
+ adap->flags |= NAPI_INIT;
+ }

/*
@@ -896,7 +903,8 @@ static int cxgb_up(struct adapter *adap)
goto out;

setup_rss(adap);
- init_napi(adap);
+ if (!(adap->flags & NAPI_INIT))
+ init_napi(adap);
adap->flags |= FULL_INIT_DONE;
+ }

@@ -999,7 +1007,7 @@ static int offload_open(struct net_device *dev)
return 0;

if (!adap_up && (err = cxgb_up(adapter)) < 0)
- return err;
+ goto out;

t3_tp_set_offload_mode(adapter, 1);
tdev->ldev = adapter->port[0];
@@ -1061,10 +1069,8 @@ static int cxgb_open(struct net_device *dev)
int other_ports = adapter->open_device_map & PORT_MASK;
int err;

- if (!adapter->open_device_map && (err = cxgb_up(adapter)) < 0) {
- quiesce_rx(adapter);
+ if (!adapter->open_device_map && (err = cxgb_up(adapter)) < 0)
return err;
- }

set_bit(pi->port_id, &adapter->open_device_map);
if (is_offload(adapter) && !ofld_disable) {
@@ -2424,14 +2430,11 @@ static pci_ers_result_t t3_io_error_detected(struct pci_dev *pdev,
test_bit(OFFLOAD_DEVMAP_BIT, &adapter->open_device_map))
offload_close(&adapter->tdev);

- /* Free sge resources */
- t3_free_sge_resources(adapter);
-
adapter->flags &= ~FULL_INIT_DONE;

pci_disable_device(pdev);

- /* Request a slot slot reset. */
+ /* Request a slot reset. */
return PCI_ERS_RESULT_NEED_RESET;
+ }
}
```

[git patches] net driver updates for .26

```
@@ -2448,13 +2451,20 @@ static pci_ers_result_t t3_io_slot_reset(struct pci_dev *pdev)
if (pci_enable_device(pdev)) {
dev_err(&pdev->dev,
"Cannot re-enable PCI device after reset.\n");
- return PCI_ERS_RESULT_DISCONNECT;
+ goto err;
}
pci_set_master(pdev);
+ pci_restore_state(pdev);

- t3_prep_adapter(adapter, adapter->params.info, 1);
+ /* Free sge resources */
+ t3_free_sge_resources(adapter);
+
+ if (t3_replay_prep_adapter(adapter))
+ goto err;

return PCI_ERS_RESULT_RECOVERED;
+err:
+ return PCI_ERS_RESULT_DISCONNECT;
}

/**
@@ -2483,13 +2493,6 @@ static void t3_io_resume(struct pci_dev *pdev)
netif_device_attach(netdev);
}
}
-
- if (is_offload(adapter)) {
- __set_bit(OFFLOAD_DEVMAP_BIT, &adapter->registered_device_map);
- if (offload_open(adapter->port[0]))
- printk(KERN_WARNING
- "Could not bring back offload capabilities\n");
- }
}

static struct pci_error_handlers t3_err_handler = {
@@ -2608,6 +2611,7 @@ static int __devinit init_one(struct pci_dev *pdev,
}

pci_set_master(pdev);
+ pci_save_state(pdev);

mmio_start = pci_resource_start(pdev, 0);
mmio_len = pci_resource_len(pdev, 0);
diff --git a/drivers/net/cxgb3/regs.h b/drivers/net/cxgb3/regs.h
index 02dbbb3..5671788 100644
--- a/drivers/net/cxgb3/regs.h
+++ b/drivers/net/cxgb3/regs.h
@@ -444,6 +444,14 @@
```

```

#define A_PCIE_CFG 0x88

+#define S_ENABLELINKDWNDRST 21
+#define V_ENABLELINKDWNDRST(x) ((x) << S_ENABLELINKDWNDRST)
+#define F_ENABLELINKDWNDRST V_ENABLELINKDWNDRST(1U)
+
+#define S_ENABLELINKDOWNRST 20
+#define V_ENABLELINKDOWNRST(x) ((x) << S_ENABLELINKDOWNRST)
+#define F_ENABLELINKDOWNRST V_ENABLELINKDOWNRST(1U)
+
#define S_PCIE_CLIDECEN 16
#define V_PCIE_CLIDECEN(x) ((x) << S_PCIE_CLIDECEN)
#define F_PCIE_CLIDECEN V_PCIE_CLIDECEN(1U)
diff --git a/drivers/net/cxgb3/sge.c b/drivers/net/cxgb3/sge.c
index 98a6bbd..796eb30 100644
--- a/drivers/net/cxgb3/sge.c
+++ b/drivers/net/cxgb3/sge.c
@@ -539,6 +539,31 @@ static void *alloc_ring(struct pci_dev *pdev, size_t nelem, size_t elem_size,
 }

/**
+ * t3_reset_qset – reset a sge qset
+ * @q: the queue set
+ *
+ * Reset the qset structure.
+ * the NAPI structure is preserved in the event of
+ * the qset's reincarnation, for example during EEH recovery.
+ */
+static void t3_reset_qset(struct sge_qset *q)
+{
+ if (q->adap &&
+ !(q->adap->flags & NAPI_INIT)) {
+ memset(q, 0, sizeof(*q));
+ return;
+ }
+
+ q->adap = NULL;
+ memset(&q->rspq, 0, sizeof(q->rspq));
+ memset(q->fl, 0, sizeof(struct sge_fl) * SGE_RXQ_PER_SET);
+ memset(q->txq, 0, sizeof(struct sge_txq) * SGE_TXQ_PER_SET);
+ q->txq_stopped = 0;
+ memset(&q->tx_reclaim_timer, 0, sizeof(q->tx_reclaim_timer));
+ }
+
+
+/**
+ * free_qset – free the resources of an SGE queue set
+ * @adapter: the adapter owning the queue set
+ * @q: the queue set
+ @@ -594,7 +619,7 @@ static void t3_free_qset(struct adapter *adapter, struct sge_qset *q)

```

```

q->rspq.desc, q->rspq.phys_addr);
}

- memset(q, 0, sizeof(*q));
+ t3_reset_qset(q);
}

/**
@@ -1365,7 +1390,7 @@ static void restart_ctrlq(unsigned long data)
*/
int t3_mgmt_tx(struct adapter *adap, struct sk_buff *skb)
{
- int ret;
+ int ret;
local_bh_disable();
ret = ctrl_xmit(adap, &adap->sge.qs[0].txq[TXQ_CTRL], skb);
local_bh_enable();
diff --git a/drivers/net/cxgb3/t3_hw.c b/drivers/net/cxgb3/t3_hw.c
index a99496a..d405a93 100644
--- a/drivers/net/cxgb3/t3_hw.c
+++ b/drivers/net/cxgb3/t3_hw.c
@@ -3264,6 +3264,7 @@ static void config_pcie(struct adapter *adap)

t3_write_reg(adap, A_PCIE_PEX_ERR, 0xffffffff);
t3_set_reg_field(adap, A_PCIE_CFG, 0,
+ F_ENABLELINKDWNDRST | F_ENABLELINKDOWNRST |
F_PCIE_DMASTOPEN | F_PCIE_CLIDECEN);
}

@@ -3655,3 +3656,30 @@ void t3_led_ready(struct adapter *adapter)
t3_set_reg_field(adapter, A_T3DBG_GPIO_EN, F_GPIO0_OUT_VAL,
F_GPIO0_OUT_VAL);
}
+
+int t3_replay_prep_adapter(struct adapter *adapter)
+{
+ const struct adapter_info *ai = adapter->params.info;
+ unsigned int i, j = 0;
+ int ret;
+
+ early_hw_init(adapter, ai);
+ ret = init_parity(adapter);
+ if (ret)
+ return ret;
+
+ for_each_port(adapter, i) {
+ struct port_info *p = adap2pinfo(adapter, i);
+ while (!adapter->params.vpd.port_type[j])
+ ++j;
+
+ p->port_type->phy_prep(&p->phy, adapter, ai->phy_base_addr + j,

```

[git patches] net driver updates for .26

```
+ ai->mdio_ops);
+
+ p->phy.ops->power_down(&p->phy, 1);
+ ++j;
+ }
+
+return 0;
+}
+
diff --git a/drivers/net/dm9000.c b/drivers/net/dm9000.c
index e6fe261..d45bcd2 100644
--- a/drivers/net/dm9000.c
+++ b/drivers/net/dm9000.c
@@ -117,6 +117,9 @@ typedef struct board_info {

struct mutex addr_lock; /* phy and eeprom access lock */

+ struct delayed_work phy_poll;
+ struct net_device *ndev;
+
spinlock_t lock;

struct mii_if_info mii;
@@ -297,6 +300,10 @@ static void dm9000_set_io(struct board_info *db, int byte_width)
}
}

+static void dm9000_schedule_poll(board_info_t *db)
+{
+ schedule_delayed_work(&db->phy_poll, HZ * 2);
+}

/* Our watchdog timed out. Called by the networking layer */
static void dm9000_timeout(struct net_device *dev)
@@ -465,6 +472,17 @@ static const struct ethtool_ops dm9000_ethtool_ops = {
.set_eeprom = dm9000_set_eeprom,
};

+static void
+dm9000_poll_work(struct work_struct *w)
+{
+ struct delayed_work *dw = container_of(w, struct delayed_work, work);
+ board_info_t *db = container_of(dw, board_info_t, phy_poll);
+
+ mii_check_media(&db->mii, netif_msg_link(db), 0);
+
+ if (netif_running(db->ndev))
+ dm9000_schedule_poll(db);
+}

/* dm9000_release_board
```

[git patches] net driver updates for .26

```
*
@@ -503,7 +521,7 @@ dm9000_release_board(struct platform_device *pdev, struct board_info *db)
/*
* Search DM9000 board, allocate space and register it
*/
-static int
+static int __devinit
dm9000_probe(struct platform_device *pdev)
{
struct dm9000_plat_data *pdata = pdev->dev.platform_data;
@@ -525,17 +543,21 @@ dm9000_probe(struct platform_device *pdev)

SET_NETDEV_DEV(ndev, &pdev->dev);

- dev_dbg(&pdev->dev, "dm9000_probe()");
+ dev_dbg(&pdev->dev, "dm9000_probe()\n");

/* setup board info structure */
db = (struct board_info *) ndev->priv;
memset(db, 0, sizeof (*db));

db->dev = &pdev->dev;
+ db->ndev = ndev;

spin_lock_init(&db->lock);
mutex_init(&db->addr_lock);

+ INIT_DELAYED_WORK(&db->phy_poll, dm9000_poll_work);
+
+
if (pdev->num_resources < 2) {
ret = -ENODEV;
goto out;
@@ -761,6 +783,8 @@ dm9000_open(struct net_device *dev)

mii_check_media(&db->mii, netif_msg_link(db), 1);
netif_start_queue(dev);
+
+ dm9000_schedule_poll(db);

return 0;
}
@@ -879,6 +903,8 @@ dm9000_stop(struct net_device *ndev)
if (netif_msg_ifdown(db))
dev_dbg(db->dev, "shutting down %s\n", ndev->name);

+ cancel_delayed_work(&db->phy_poll);
+
netif_stop_queue(ndev);
netif_carrier_off(ndev);
```

[git patches] net driver updates for .26

```
@@ -1288,6 +1314,8 @@ dm9000_phy_read(struct net_device *dev, int phy_reg_unused, int reg)
spin_unlock_irqrestore(&db->lock,flags);
```

```
mutex_unlock(&db->addr_lock);
+
+ dm9000_dbg(db, 5, "phy_read[%02x] -> %04x\n", reg, ret);
return ret;
}
```

```
@@ -1301,6 +1329,7 @@ dm9000_phy_write(struct net_device *dev, int phyaddr_unused, int reg, int
value)
```

```
unsigned long flags;
```

```
unsigned long reg_save;
```

```
+ dm9000_dbg(db, 5, "phy_write[%02x] = %04x\n", reg, value);
mutex_lock(&db->addr_lock);
```

```
spin_lock_irqsave(&db->lock,flags);
```

```
@@ -1372,7 +1401,7 @@ dm9000_drv_resume(struct platform_device *dev)
```

```
return 0;
```

```
}
```

```
-static int
```

```
+static int __devexit
```

```
dm9000_drv_remove(struct platform_device *pdev)
```

```
{
```

```
struct net_device *ndev = platform_get_drvdata(pdev);
```

```
@@ -1393,7 +1422,7 @@ static struct platform_driver dm9000_driver = {
```

```
.owner = THIS_MODULE,
```

```
},
```

```
.probe = dm9000_probe,
```

```
- .remove = dm9000_drv_remove,
```

```
+ .remove = __devexit_p(dm9000_drv_remove),
```

```
.suspend = dm9000_drv_suspend,
```

```
.resume = dm9000_drv_resume,
```

```
};
```

```
diff --git a/drivers/net/ehca/ehca.h b/drivers/net/ehca/ehca.h
```

```
index f5dacce..fe872fb 100644
```

```
--- a/drivers/net/ehca/ehca.h
```

```
+++ b/drivers/net/ehca/ehca.h
```

```
@@ -40,7 +40,7 @@
```

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

```
#define DRV_NAME "ehca"
```

```
-#define DRV_VERSION "EHEA_0090"
```

```
+#define DRV_VERSION "EHEA_0091"
```

```
/* eHEA capability flags */
```

```
#define DLPAR_PORT_ADD_REM 1
```

```
@@ -118,6 +118,13 @@
```

```
#define EHEA_MR_ACC_CTRL 0x00800000
```

[git patches] net driver updates for .26

```

#define EHEA_BUSMAP_START 0x8000000000000000ULL
+#define EHEA_INVALID_ADDR 0xFFFFFFFFFFFFFFFFULL
+#define EHEA_DIR_INDEX_SHIFT 13 /* 8k Entries in 64k block */
+#define EHEA_TOP_INDEX_SHIFT (EHEA_DIR_INDEX_SHIFT * 2)
+#define EHEA_MAP_ENTRIES (1 << EHEA_DIR_INDEX_SHIFT)
+#define EHEA_MAP_SIZE (0x10000) /* currently fixed map size */
+#define EHEA_INDEX_MASK (EHEA_MAP_ENTRIES - 1)
+
+
#define EHEA_WATCH_DOG_TIMEOUT 10*HZ

@@ -192,10 +199,20 @@ struct h_epas {
set to 0 if unused */
};

-struct ehea_busmap {
- unsigned int entries; /* total number of entries */
- unsigned int valid_sections; /* number of valid sections */
- u64 *vaddr;
+/*
+ * Memory map data structures
+ */
+struct ehea_dir_bmap
+{
+ u64 ent[EHEA_MAP_ENTRIES];
+};
+struct ehea_top_bmap
+{
+ struct ehea_dir_bmap *dir[EHEA_MAP_ENTRIES];
+};
+struct ehea_bmap
+{
+ struct ehea_top_bmap *top[EHEA_MAP_ENTRIES];
+};

struct ehea_qp;
diff --git a/drivers/net/ehea/ehea_main.c b/drivers/net/ehea/ehea_main.c
index f9bc21c..d1b6d4e 100644
--- a/drivers/net/ehea/ehea_main.c
+++ b/drivers/net/ehea/ehea_main.c
@@ -35,6 +35,7 @@
#include <linux/if_ether.h>
#include <linux/notifier.h>
#include <linux/reboot.h>
+#include <linux/memory.h>
#include <asm/kexec.h>
#include <linux/mutex.h>

@@ -3503,6 +3504,24 @@ void ehea_crash_handler(void)
0, H_DEREG_BCNC);

```

```

}

+static int ehea_mem_notifier(struct notifier_block *nb,
+ unsigned long action, void *data)
+{
+ switch (action) {
+ case MEM_OFFLINE:
+ ehea_info("memory has been removed");
+ ehea_rereg_mrs(NULL);
+ break;
+ default:
+ break;
+ }
+ return NOTIFY_OK;
+}
+
+static struct notifier_block ehea_mem_nb = {
+ .notifier_call = ehea_mem_notifier,
+};
+
static int ehea_reboot_notifier(struct notifier_block *nb,
unsigned long action, void *unused)
{
@@ -3581,6 +3600,10 @@ int __init ehea_module_init(void)
if (ret)
ehea_info("failed registering reboot notifier");

+ ret = register_memory_notifier(&ehea_mem_nb);
+ if (ret)
+ ehea_info("failed registering memory remove notifier");
+
ret = crash_shutdown_register(&ehea_crash_handler);
if (ret)
ehea_info("failed registering crash handler");
@@ -3604,6 +3627,7 @@ int __init ehea_module_init(void)
out3:
ibmebus_unregister_driver(&ehea_driver);
out2:
+ unregister_memory_notifier(&ehea_mem_nb);
unregister_reboot_notifier(&ehea_reboot_nb);
crash_shutdown_unregister(&ehea_crash_handler);
out:
@@ -3621,6 +3645,7 @@ static void __exit ehea_module_exit(void)
ret = crash_shutdown_unregister(&ehea_crash_handler);
if (ret)
ehea_info("failed unregistering crash handler");
+ unregister_memory_notifier(&ehea_mem_nb);
kfree(ehea_fw_handles.arr);
kfree(ehea_bcmc_regs.arr);
ehea_destroy_busmap();
diff --git a/drivers/net/ehea/ehea_qmr.c b/drivers/net/ehea/ehea_qmr.c

```

```

index d522e90..140f05b 100644
--- a/drivers/net/ehea/ehea_qmr.c
+++ b/drivers/net/ehea/ehea_qmr.c
@@ -31,8 +31,8 @@
#include "ehea_phyp.h"
#include "ehea_qmr.h"

+struct ehea_bmap *ehea_bmap = NULL;

-struct ehea_bmap ehea_bmap = { 0, 0, NULL };

static void *hw_qpageit_get_inc(struct hw_queue *queue)
@@ -559,125 +559,253 @@ int ehea_destroy_qp(struct ehea_qp *qp)
return 0;
}

-int ehea_create_bmap(void)
+static inline int ehea_calc_index(unsigned long i, unsigned long s)
{
- u64 vaddr = EHEA_BMAP_START;
- unsigned long high_section_index = 0;
- int i;
+ return (i >> s) & EHEA_INDEX_MASK;
+}

- /*
- * Sections are not in ascending order -> Loop over all sections and
- * find the highest PFN to compute the required map size.
- */
- ehea_bmap.valid_sections = 0;
+static inline int ehea_init_top_bmap(struct ehea_top_bmap *ehea_top_bmap,
+ int dir)
+{
+ if(!ehea_top_bmap->dir[dir]) {
+ ehea_top_bmap->dir[dir] =
+ kzalloc(sizeof(struct ehea_dir_bmap), GFP_KERNEL);
+ if (!ehea_top_bmap->dir[dir])
+ return -ENOMEM;
+ }
+ return 0;
+}

- for (i = 0; i < NR_MEM_SECTIONS; i++)
- if (valid_section_nr(i))
- high_section_index = i;
+static inline int ehea_init_bmap(struct ehea_bmap *ehea_bmap, int top, int dir)
+{
+ if(!ehea_bmap->top[top]) {
+ ehea_bmap->top[top] =
+ kzalloc(sizeof(struct ehea_top_bmap), GFP_KERNEL);

```

```

+ if (!ehea_bmap->top[top])
+ return -ENOMEM;
+ }
+ return ehea_init_top_bmap(ehea_bmap->top[top], dir);
+}

- ehea_bmap.entries = high_section_index + 1;
- ehea_bmap.vaddr = vmalloc(ehea_bmap.entries * sizeof(*ehea_bmap.vaddr));
+static int ehea_create_busmap_callback(unsigned long pfn,
+ unsigned long nr_pages, void *arg)
+{
+ unsigned long i, mr_len, start_section, end_section;
+ start_section = (pfn * PAGE_SIZE) / EHEA_SECTSIZE;
+ end_section = start_section + ((nr_pages * PAGE_SIZE) / EHEA_SECTSIZE);
+ mr_len = *(unsigned long *)arg;

- if (!ehea_bmap.vaddr)
+ ehea_bmap = kzalloc(sizeof(struct ehea_bmap), GFP_KERNEL);
+ if (!ehea_bmap)
return -ENOMEM;

- for (i = 0 ; i < ehea_bmap.entries; i++) {
- unsigned long pfn = section_nr_to_pfn(i);
+ for (i = start_section; i < end_section; i++) {
+ int ret;
+ int top, dir, idx;
+ u64 vaddr;
+
+ top = ehea_calc_index(i, EHEA_TOP_INDEX_SHIFT);
+ dir = ehea_calc_index(i, EHEA_DIR_INDEX_SHIFT);
+
+ ret = ehea_init_bmap(ehea_bmap, top, dir);
+ if(ret)
+ return ret;

- if (pfn_valid(pfn)) {
- ehea_bmap.vaddr[i] = vaddr;
- vaddr += EHEA_SECTSIZE;
- ehea_bmap.valid_sections++;
- } else
- ehea_bmap.vaddr[i] = 0;
+ idx = i & EHEA_INDEX_MASK;
+ vaddr = EHEA_BUSMAP_START + mr_len + i * EHEA_SECTSIZE;
+
+ ehea_bmap->top[top]->dir[dir]->ent[idx] = vaddr;
}

+ mr_len += nr_pages * PAGE_SIZE;
+ *(unsigned long *)arg = mr_len;
+
return 0;

```

```

}

+static unsigned long ehea_mr_len;
+
+static DEFINE_MUTEX(ehea_busmap_mutex);
+
+int ehea_create_busmap(void)
+{
+ int ret;
+ mutex_lock(&ehea_busmap_mutex);
+ ehea_mr_len = 0;
+ ret = walk_memory_resource(0, 1ULL << MAX_PHYSMEM_BITS, &ehea_mr_len,
+ ehea_create_busmap_callback);
+ mutex_unlock(&ehea_busmap_mutex);
+ return ret;
+}
+
void ehea_destroy_busmap(void)
{
- vfree(ehea_bmap.vaddr);
+ int top, dir;
+ mutex_lock(&ehea_busmap_mutex);
+ if (!ehea_bmap)
+ goto out_destroy;
+
+ for (top = 0; top < EHEA_MAP_ENTRIES; top++) {
+ if (!ehea_bmap->top[top])
+ continue;
+
+ for (dir = 0; dir < EHEA_MAP_ENTRIES; dir++) {
+ if (!ehea_bmap->top[top]->dir[dir])
+ continue;
+
+ kfree(ehea_bmap->top[top]->dir[dir]);
+ }
+
+ kfree(ehea_bmap->top[top]);
+ }
+
+ kfree(ehea_bmap);
+ ehea_bmap = NULL;
+out_destroy:
+ mutex_unlock(&ehea_busmap_mutex);
+ }

u64 ehea_map_vaddr(void *caddr)
{
- u64 mapped_addr;
- unsigned long index = __pa(caddr) >> SECTION_SIZE_BITS;
-
- if (likely(index < ehea_bmap.entries)) {

```

```

- mapped_addr = ehea_bmap.vaddr[index];
- if (likely(mapped_addr))
- mapped_addr |= (((unsigned long)caddr)
- & (EHEA_SECTSIZE - 1));
- else
- mapped_addr = -1;
- } else
- mapped_addr = -1;
-
- if (unlikely(mapped_addr == -1))
- if (!test_and_set_bit(__EHEA_STOP_XFER, &ehea_driver_flags))
- schedule_work(&ehea_rereg_mr_task);
-
- return mapped_addr;
+ int top, dir, idx;
+ unsigned long index, offset;
+
+ if (!ehea_bmap)
+ return EHEA_INVALID_ADDR;
+
+ index = virt_to_abs(caddr) >> SECTION_SIZE_BITS;
+ top = (index >> EHEA_TOP_INDEX_SHIFT) & EHEA_INDEX_MASK;
+ if (!ehea_bmap->top[top])
+ return EHEA_INVALID_ADDR;
+
+ dir = (index >> EHEA_DIR_INDEX_SHIFT) & EHEA_INDEX_MASK;
+ if (!ehea_bmap->top[top]->dir[dir])
+ return EHEA_INVALID_ADDR;
+
+ idx = index & EHEA_INDEX_MASK;
+ if (!ehea_bmap->top[top]->dir[dir]->ent[idx])
+ return EHEA_INVALID_ADDR;
+
+ offset = (unsigned long)caddr & (EHEA_SECTSIZE - 1);
+ return ehea_bmap->top[top]->dir[dir]->ent[idx] | offset;
+}
+
+static inline void *ehea_calc_sectbase(int top, int dir, int idx)
+{
+ unsigned long ret = idx;
+ ret |= dir << EHEA_DIR_INDEX_SHIFT;
+ ret |= top << EHEA_TOP_INDEX_SHIFT;
+ return abs_to_virt(ret << SECTION_SIZE_BITS);
+}
+
+static u64 ehea_reg_mr_section(int top, int dir, int idx, u64 *pt,
+ struct ehea_adapter *adapter,
+ struct ehea_mr *mr)
+{
+ void *pg;
+ u64 j, m, hret;

```

```

+ unsigned long k = 0;
+ u64 pt_abs = virt_to_abs(pt);
+
+ void *sectbase = ehea_calc_sectbase(top, dir, idx);
+
+ for (j = 0; j < (EHEA_PAGES_PER_SECTION / EHEA_MAX_RPAGE); j++) {
+
+ for (m = 0; m < EHEA_MAX_RPAGE; m++) {
+ pg = sectbase + ((k++) * EHEA_PAGESIZE);
+ pt[m] = virt_to_abs(pg);
+ }
+ hret = ehea_h_register_rpage_mr(adapter->handle, mr->handle, 0,
+ 0, pt_abs, EHEA_MAX_RPAGE);
+
+ if ((hret != H_SUCCESS)
+ && (hret != H_PAGE_REGISTERED)) {
+ ehea_h_free_resource(adapter->handle, mr->handle,
+ FORCE_FREE);
+ ehea_error("register_rpage_mr failed");
+ return hret;
+ }
+ }
+ return hret;
+}
+
+static u64 ehea_reg_mr_sections(int top, int dir, u64 *pt,
+ struct ehea_adapter *adapter,
+ struct ehea_mr *mr)
+{
+ u64 hret = H_SUCCESS;
+ int idx;
+
+ for (idx = 0; idx < EHEA_MAP_ENTRIES; idx++) {
+ if (!ehea_bmap->top[top]->dir[dir]->ent[idx])
+ continue;
+
+ hret = ehea_reg_mr_section(top, dir, idx, pt, adapter, mr);
+ if ((hret != H_SUCCESS) && (hret != H_PAGE_REGISTERED))
+ return hret;
+ }
+ return hret;
+}
+
+static u64 ehea_reg_mr_dir_sections(int top, u64 *pt,
+ struct ehea_adapter *adapter,
+ struct ehea_mr *mr)
+{
+ u64 hret = H_SUCCESS;
+ int dir;
+
+ for (dir = 0; dir < EHEA_MAP_ENTRIES; dir++) {

```

```

+ if (!eha_bmap->top[top]->dir[dir])
+ continue;
+
+ hret = ehea_reg_mr_sections(top, dir, pt, adapter, mr);
+ if ((hret != H_SUCCESS) && (hret != H_PAGE_REGISTERED))
+ return hret;
+ }
+ return hret;
}

int ehea_reg_kernel_mr(struct ehea_adapter *adapter, struct ehea_mr *mr)
{
int ret;
u64 *pt;
- void *pg;
- u64 hret, pt_abs, i, j, m, mr_len;
+ u64 hret;
u32 acc_ctrl = EHEA_MR_ACC_CTRL;

- mr_len = ehea_bmap.valid_sections * EHEA_SECTSIZE;
+ unsigned long top;

- pt = kzalloc(PAGE_SIZE, GFP_KERNEL);
+ pt = kzalloc(PAGE_SIZE, GFP_KERNEL);
if (!pt) {
ehea_error("no mem");
ret = -ENOMEM;
goto out;
}
- pt_abs = virt_to_abs(pt);

- hret = ehea_h_alloc_resource_mr(adapter->handle,
- EHEA_BUSMAP_START, mr_len,
- acc_ctrl, adapter->pd,
+ hret = ehea_h_alloc_resource_mr(adapter->handle, EHEA_BUSMAP_START,
+ ehea_mr_len, acc_ctrl, adapter->pd,
+ &mr->handle, &mr->lkey);
+
if (hret != H_SUCCESS) {
ehea_error("alloc_resource_mr failed");
ret = -EIO;
goto out;
}

- for (i = 0 ; i < ehea_bmap.entries; i++)
- if (eha_bmap.vaddr[i]) {
- void *sectbase = __va(i << SECTION_SIZE_BITS);
- unsigned long k = 0;
-
- for (j = 0; j < (EHEA_PAGES_PER_SECTION /
- EHEA_MAX_RPAGE); j++) {

```

```

-
- for (m = 0; m < EHEA_MAX_RPAGE; m++) {
- pg = sectbase + ((k++) * EHEA_PAGESIZE);
- pt[m] = virt_to_abs(pg);
- }
-
- hret = ehea_h_register_rpage_mr(adapter->handle,
- mr->handle,
- 0, 0, pt_abs,
- EHEA_MAX_RPAGE);
- if ((hret != H_SUCCESS)
- && (hret != H_PAGE_REGISTERED)) {
- ehea_h_free_resource(adapter->handle,
- mr->handle,
- FORCE_FREE);
- ehea_error("register_rpage_mr failed");
- ret = -EIO;
- goto out;
- }
- }
- }
- }
+ if (!ehea_bmap) {
+ ehea_h_free_resource(adapter->handle, mr->handle, FORCE_FREE);
+ ehea_error("no busmap available");
+ ret = -EIO;
+ goto out;
+ }
+
+ for (top = 0; top < EHEA_MAP_ENTRIES; top++) {
+ if (!ehea_bmap->top[top])
+ continue;
+
+ hret = ehea_reg_mr_dir_sections(top, pt, adapter, mr);
+ if((hret != H_PAGE_REGISTERED) && (hret != H_SUCCESS))
+ break;
+ }

if (hret != H_SUCCESS) {
ehea_h_free_resource(adapter->handle, mr->handle, FORCE_FREE);
diff --git a/drivers/net/gianfar.c b/drivers/net/gianfar.c
index 6f22f06..25bdd08 100644
--- a/drivers/net/gianfar.c
+++ b/drivers/net/gianfar.c
@@ -635,6 +635,8 @@ static void free_skb_resources(struct gfar_private *priv)
dev_kfree_skb_any(priv->tx_skbuff[i]);
priv->tx_skbuff[i] = NULL;
}
+
+ txbdp++;
}

```

```

kfree(priv->tx_skbuff);
diff --git a/drivers/net/myri10ge/myri10ge.c b/drivers/net/myri10ge/myri10ge.c
index ef63c8d..c91b12e 100644
--- a/drivers/net/myri10ge/myri10ge.c
+++ b/drivers/net/myri10ge/myri10ge.c
@@ -144,11 +144,13 @@ struct myri10ge_tx_buf {
char *req_bytes;
struct myri10ge_tx_buffer_state *info;
int mask; /* number of transmit slots -1 */
- int boundary; /* boundary transmits cannot cross */
int req ____cacheline_aligned; /* transmit slots submitted */
int pkt_start; /* packets started */
+ int stop_queue;
+ int linearized;
int done ____cacheline_aligned; /* transmit slots completed */
int pkt_done; /* packets completed */
+ int wake_queue;
};

struct myri10ge_rx_done {
@@ -160,29 +162,50 @@ struct myri10ge_rx_done {
struct net_lro_desc lro_desc[MYRI10GE_MAX_LRO_DESCRIPTOR];
};

-struct myri10ge_priv {
- int running; /* running? */
- int csum_flag; /* rx_csums? */
+struct myri10ge_slice_netstats {
+ unsigned long rx_packets;
+ unsigned long tx_packets;
+ unsigned long rx_bytes;
+ unsigned long tx_bytes;
+ unsigned long rx_dropped;
+ unsigned long tx_dropped;
+};
+
+struct myri10ge_slice_state {
struct myri10ge_tx_buf tx; /* transmit ring */
struct myri10ge_rx_buf rx_small;
struct myri10ge_rx_buf rx_big;
struct myri10ge_rx_done rx_done;
+ struct net_device *dev;
+ struct napi_struct napi;
+ struct myri10ge_priv *mgp;
+ struct myri10ge_slice_netstats stats;
+ __be32 __iomem *irq_claim;
+ struct mcp_irq_data *fw_stats;
+ dma_addr_t fw_stats_bus;
+ int watchdog_tx_done;
+ int watchdog_tx_req;
+};

```

```

+
+struct myri10ge_priv {
+ struct myri10ge_slice_state ss;
+ int tx_boundary; /* boundary transmits cannot cross */
+ int running; /* running? */
+ int csum_flag; /* rx_csums? */
int small_bytes;
int big_bytes;
+ int max_intr_slots;
struct net_device *dev;
- struct napi_struct napi;
struct net_device_stats stats;
+ spinlock_t stats_lock;
u8 __iomem *sram;
int sram_size;
unsigned long board_span;
unsigned long iomem_base;
- __be32 __iomem *irq_claim;
__be32 __iomem *irq_deassert;
char *mac_addr_string;
struct mcp_cmd_response *cmd;
dma_addr_t cmd_bus;
- struct mcp_irq_data *fw_stats;
- dma_addr_t fw_stats_bus;
struct pci_dev *pdev;
int msi_enabled;
u32 link_state;
@@ -191,20 +214,16 @@ struct myri10ge_priv {
__be32 __iomem *intr_coal_delay_ptr;
int mtrr;
int wc_enabled;
- int wake_queue;
- int stop_queue;
int down_cnt;
wait_queue_head_t down_wq;
struct work_struct watchdog_work;
struct timer_list watchdog_timer;
- int watchdog_tx_done;
- int watchdog_tx_req;
- int watchdog_pause;
int watchdog_resets;
- int tx_linearized;
+ int watchdog_pause;
int pause;
char *fw_name;
char eeprom_strings[MYRI10GE_EEPROM_STRINGS_SIZE];
+ char *product_code_string;
char fw_version[128];
int fw_ver_major;
int fw_ver_minor;
@@ -228,58 +247,54 @@ static char *myri10ge_fw_aligned = "myri10ge_eth_z8e.dat";

```

```

static char *myri10ge_fw_name = NULL;
module_param(myri10ge_fw_name, charp, S_IRUGO | S_IWUSR);
-MODULE_PARAM_DESC(myri10ge_fw_name, "Firmware image name\n");
+MODULE_PARAM_DESC(myri10ge_fw_name, "Firmware image name");

static int myri10ge_ecrc_enable = 1;
module_param(myri10ge_ecrc_enable, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_ecrc_enable, "Enable Extended CRC on PCI-E\n");
-
-static int myri10ge_max_intr_slots = 1024;
-module_param(myri10ge_max_intr_slots, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_max_intr_slots, "Interrupt queue slots\n");
+MODULE_PARAM_DESC(myri10ge_ecrc_enable, "Enable Extended CRC on PCI-E");

static int myri10ge_small_bytes = -1; /* -1 == auto */
module_param(myri10ge_small_bytes, int, S_IRUGO | S_IWUSR);
-MODULE_PARAM_DESC(myri10ge_small_bytes, "Threshold of small packets\n");
+MODULE_PARAM_DESC(myri10ge_small_bytes, "Threshold of small packets");

static int myri10ge_msi = 1; /* enable msi by default */
module_param(myri10ge_msi, int, S_IRUGO | S_IWUSR);
-MODULE_PARAM_DESC(myri10ge_msi, "Enable Message Signalled Interrupts\n");
+MODULE_PARAM_DESC(myri10ge_msi, "Enable Message Signalled Interrupts");

static int myri10ge_intr_coal_delay = 75;
module_param(myri10ge_intr_coal_delay, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_intr_coal_delay, "Interrupt coalescing delay\n");
+MODULE_PARAM_DESC(myri10ge_intr_coal_delay, "Interrupt coalescing delay");

static int myri10ge_flow_control = 1;
module_param(myri10ge_flow_control, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_flow_control, "Pause parameter\n");
+MODULE_PARAM_DESC(myri10ge_flow_control, "Pause parameter");

static int myri10ge_deassert_wait = 1;
module_param(myri10ge_deassert_wait, int, S_IRUGO | S_IWUSR);
MODULE_PARAM_DESC(myri10ge_deassert_wait,
- "Wait when deasserting legacy interrupts\n");
+ "Wait when deasserting legacy interrupts");

static int myri10ge_force_firmware = 0;
module_param(myri10ge_force_firmware, int, S_IRUGO);
MODULE_PARAM_DESC(myri10ge_force_firmware,
- "Force firmware to assume aligned completions\n");
+ "Force firmware to assume aligned completions");

static int myri10ge_initial_mtu = MYRI10GE_MAX_ETHER_MTU - ETH_HLEN;
module_param(myri10ge_initial_mtu, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_initial_mtu, "Initial MTU\n");
+MODULE_PARAM_DESC(myri10ge_initial_mtu, "Initial MTU");

```

```

static int myri10ge_napi_weight = 64;
module_param(myri10ge_napi_weight, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_napi_weight, "Set NAPI weight\n");
+MODULE_PARAM_DESC(myri10ge_napi_weight, "Set NAPI weight");

static int myri10ge_watchdog_timeout = 1;
module_param(myri10ge_watchdog_timeout, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_watchdog_timeout, "Set watchdog timeout\n");
+MODULE_PARAM_DESC(myri10ge_watchdog_timeout, "Set watchdog timeout");

static int myri10ge_max_irq_loops = 1048576;
module_param(myri10ge_max_irq_loops, int, S_IRUGO);
MODULE_PARAM_DESC(myri10ge_max_irq_loops,
- "Set stuck legacy IRQ detection threshold\n");
+ "Set stuck legacy IRQ detection threshold");

#define MYRI10GE_MSG_DEFAULT NETIF_MSG_LINK

@@ -289,21 +304,22 @@ MODULE_PARAM_DESC(myri10ge_debug, "Debug level (0=none, ...,16=all)");

static int myri10ge_lro = 1;
module_param(myri10ge_lro, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_lro, "Enable large receive offload\n");
+MODULE_PARAM_DESC(myri10ge_lro, "Enable large receive offload");

static int myri10ge_lro_max_pkts = MYRI10GE_LRO_MAX_PKTS;
module_param(myri10ge_lro_max_pkts, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_lro, "Number of LRO packets to be aggregated\n");
+MODULE_PARAM_DESC(myri10ge_lro_max_pkts,
+ "Number of LRO packets to be aggregated");

static int myri10ge_fill_thresh = 256;
module_param(myri10ge_fill_thresh, int, S_IRUGO | S_IWUSR);
-MODULE_PARAM_DESC(myri10ge_fill_thresh, "Number of empty rx slots allowed\n");
+MODULE_PARAM_DESC(myri10ge_fill_thresh, "Number of empty rx slots allowed");

static int myri10ge_reset_recover = 1;

static int myri10ge_wcfifo = 0;
module_param(myri10ge_wcfifo, int, S_IRUGO);
-MODULE_PARAM_DESC(myri10ge_wcfifo, "Enable WC Fifo when WC is enabled\n");
+MODULE_PARAM_DESC(myri10ge_wcfifo, "Enable WC Fifo when WC is enabled");

#define MYRI10GE_FW_OFFSET 1024*1024
#define MYRI10GE_HIGHPART_TO_U32(X) \
@@ -359,8 +375,10 @@ myri10ge_send_cmd(struct myri10ge_priv *mgrp, u32 cmd,
for (sleep_total = 0;
sleep_total < 1000
&& response->result == htonl(MYRI10GE_NO_RESPONSE_RESULT);
- sleep_total += 10)

```

```

+ sleep_total += 10) {
udelay(10);
+ mb();
+ }
} else {
/* use msleep for most command */
for (sleep_total = 0;
@@ -420,6 +438,10 @@ static int myri10ge_read_mac_addr(struct myri10ge_priv *mgp)
ptr += 1;
}
}
+ if (memcmp(ptr, "PC=", 3) == 0) {
+ ptr += 3;
+ mgp->product_code_string = ptr;
+ }
if (memcmp((const void *)ptr, "SN=", 3) == 0) {
ptr += 3;
mgp->serial_number = simple_strtoul(ptr, &ptr, 10);
@@ -442,7 +464,7 @@ abort:
static void myri10ge_dummy_rdma(struct myri10ge_priv *mgp, int enable)
{
char __iomem *submit;
- __be32 buf[16];
+ __be32 buf[16] __attribute__((__aligned__(8)));
u32 dma_low, dma_high;
int i;

@@ -609,13 +631,38 @@ static int myri10ge_adopt_running_firmware(struct myri10ge_priv *mgp)
return status;
}

+int myri10ge_get_firmware_capabilities(struct myri10ge_priv *mgp)
+{
+ struct myri10ge_cmd cmd;
+ int status;
+
+ /* probe for IPv6 TSO support */
+ mgp->features = NETIF_F_SG | NETIF_F_HW_CSUM | NETIF_F_TSO;
+ status = myri10ge_send_cmd(mgp, MXGEFW_CMD_GET_MAX_TSO6_HDR_SIZE,
+ &cmd, 0);
+ if (status == 0) {
+ mgp->max_tso6 = cmd.data0;
+ mgp->features |= NETIF_F_TSO6;
+ }
+
+ status = myri10ge_send_cmd(mgp, MXGEFW_CMD_GET_RX_RING_SIZE, &cmd, 0);
+ if (status != 0) {
+ dev_err(&mgp->pdev->dev,
+ "failed MXGEFW_CMD_GET_RX_RING_SIZE\n");
+ return -ENXIO;
+ }
}

```

[git patches] net driver updates for .26

```
+
+ mgp->max_intr_slots = 2 * (cmd.data0 / sizeof(struct mcp_dma_addr));
+
+ return 0;
+}
+
static int myri10ge_load_firmware(struct myri10ge_priv *mgp)
{
char __iomem *submit;
- __be32 buf[16];
+ __be32 buf[16] __attribute__((__aligned__(8)));
u32 dma_low, dma_high, size;
int status, i;
- struct myri10ge_cmd cmd;

size = 0;
status = myri10ge_load_hotplug_firmware(mgp, &size);
@@ -635,7 +682,7 @@ static int myri10ge_load_firmware(struct myri10ge_priv *mgp)
}
dev_info(&mgp->pdev->dev,
"Successfully adopted running firmware\n");
- if (mgp->tx.boundary == 4096) {
+ if (mgp->tx_boundary == 4096) {
dev_warn(&mgp->pdev->dev,
"Using firmware currently running on NIC"
". For optimal\n");
@@ -646,7 +693,9 @@ static int myri10ge_load_firmware(struct myri10ge_priv *mgp)
}

mgp->fw_name = "adopted";
- mgp->tx.boundary = 2048;
+ mgp->tx_boundary = 2048;
+ myri10ge_dummy_rdma(mgp, 1);
+ status = myri10ge_get_firmware_capabilities(mgp);
return status;
}

@@ -681,26 +730,18 @@ static int myri10ge_load_firmware(struct myri10ge_priv *mgp)
msleep(1);
mb();
i = 0;
- while (mgp->cmd->data != MYRI10GE_NO_CONFIRM_DATA && i < 20) {
- msleep(1);
+ while (mgp->cmd->data != MYRI10GE_NO_CONFIRM_DATA && i < 9) {
+ msleep(1 << i);
i++;
}
if (mgp->cmd->data != MYRI10GE_NO_CONFIRM_DATA) {
dev_err(&mgp->pdev->dev, "handoff failed\n");
return -ENXIO;
}
}
```

[git patches] net driver updates for .26

```
- dev_info(&mgp->pdev->dev, "handoff confirmed\n");
myri10ge_dummy_rdma(mgp, 1);
+ status = myri10ge_get_firmware_capabilities(mgp);

- /* probe for IPv6 TSO support */
- mgp->features = NETIF_F_SG | NETIF_F_HW_CSUM | NETIF_F_TSO;
- status = myri10ge_send_cmd(mgp, MXGEFW_CMD_GET_MAX_TSO6_HDR_SIZE,
- &cmd, 0);
- if (status == 0) {
- mgp->max_tso6 = cmd.data0;
- mgp->features |= NETIF_F_TSO6;
- }
- return 0;
+ return status;
}

static int myri10ge_update_mac_address(struct myri10ge_priv *mgp, u8 * addr)
@@ -772,7 +813,7 @@ static int myri10ge_dma_test(struct myri10ge_priv *mgp, int test_type)
* transfers took to complete.
*/

- len = mgp->tx.boundary;
+ len = mgp->tx_boundary;

cmd.data0 = MYRI10GE_LOWPART_TO_U32(dmatest_bus);
cmd.data1 = MYRI10GE_HIGHPART_TO_U32(dmatest_bus);
@@ -834,17 +875,17 @@ static int myri10ge_reset(struct myri10ge_priv *mgp)

/* Now exchange information about interrupts */

- bytes = myri10ge_max_intr_slots * sizeof(*mgp->rx_done.entry);
- memset(mgp->rx_done.entry, 0, bytes);
+ bytes = mgp->max_intr_slots * sizeof(*mgp->ss.rx_done.entry);
+ memset(mgp->ss.rx_done.entry, 0, bytes);
cmd.data0 = (u32) bytes;
status = myri10ge_send_cmd(mgp, MXGEFW_CMD_SET_INTRQ_SIZE, &cmd, 0);
- cmd.data0 = MYRI10GE_LOWPART_TO_U32(mgp->rx_done.bus);
- cmd.data1 = MYRI10GE_HIGHPART_TO_U32(mgp->rx_done.bus);
+ cmd.data0 = MYRI10GE_LOWPART_TO_U32(mgp->ss.rx_done.bus);
+ cmd.data1 = MYRI10GE_HIGHPART_TO_U32(mgp->ss.rx_done.bus);
status |= myri10ge_send_cmd(mgp, MXGEFW_CMD_SET_INTRQ_DMA, &cmd, 0);

status |=
myri10ge_send_cmd(mgp, MXGEFW_CMD_GET_IRQ_ACK_OFFSET, &cmd, 0);
- mgp->irq_claim = (__iomem __be32 *) (mgp->sram + cmd.data0);
+ mgp->ss.irq_claim = (__iomem __be32 *) (mgp->sram + cmd.data0);
status |= myri10ge_send_cmd(mgp, MXGEFW_CMD_GET_IRQ_DEASSERT_OFFSET,
&cmd, 0);
mgp->irq_deassert = (__iomem __be32 *) (mgp->sram + cmd.data0);
@@ -858,17 +899,17 @@ static int myri10ge_reset(struct myri10ge_priv *mgp)
}
}
```

```
put_be32(htonl(mgp->intr_coal_delay), mgp->intr_coal_delay_ptr);
```

```
- memset(mgp->rx_done.entry, 0, bytes);
+ memset(mgp->ss.rx_done.entry, 0, bytes);
```

```
/* reset mcp/driver shared state back to 0 */
```

```
- mgp->tx.req = 0;
- mgp->tx.done = 0;
- mgp->tx.pkt_start = 0;
- mgp->tx.pkt_done = 0;
- mgp->rx_big.cnt = 0;
- mgp->rx_small.cnt = 0;
- mgp->rx_done.idx = 0;
- mgp->rx_done.cnt = 0;
+ mgp->ss.tx.req = 0;
+ mgp->ss.tx.done = 0;
+ mgp->ss.tx.pkt_start = 0;
+ mgp->ss.tx.pkt_done = 0;
+ mgp->ss.rx_big.cnt = 0;
+ mgp->ss.rx_small.cnt = 0;
+ mgp->ss.rx_done.idx = 0;
+ mgp->ss.rx_done.cnt = 0;
mgp->link_changes = 0;
status = myri10ge_update_mac_address(mgp, mgp->dev->dev_addr);
myri10ge_change_pause(mgp, mgp->pause);
@@ -1020,9 +1061,10 @@ myri10ge_unmap_rx_page(struct pci_dev *pdev,
* page into an skb */
```

```
static inline int
```

```
-myri10ge_rx_done(struct myri10ge_priv *mgp, struct myri10ge_rx_buf *rx,
+myri10ge_rx_done(struct myri10ge_slice_state *ss, struct myri10ge_rx_buf *rx,
int bytes, int len, __wsum csum)
{
+ struct myri10ge_priv *mgp = ss->mgp;
struct sk_buff *skb;
struct skb_frag_struct rx_frags[MYRI10GE_MAX_FRAGS_PER_FRAME];
int i, idx, hlen, remainder;
@@ -1052,11 +1094,10 @@ myri10ge_rx_done(struct myri10ge_priv *mgp, struct myri10ge_rx_buf *rx,
rx_frags[0].page_offset += MXGEFW_PAD;
rx_frags[0].size -= MXGEFW_PAD;
len -= MXGEFW_PAD;
- lro_receive_frags(&mgp->rx_done.lro_mgr, rx_frags,
+ lro_receive_frags(&ss->rx_done.lro_mgr, rx_frags,
len, len,
- /* opaque, will come back in get_frag_header */
- (void *)(__force unsigned long)csum,
- csum);
+ /* opaque, will come back in get_frag_header */
+ (void *)(__force unsigned long)csum, csum);
return 1;
}
```

```
@@ -1096,10 +1137,11 @@ myri10ge_rx_done(struct myri10ge_priv *mgp, struct myri10ge_rx_buf *rx,
return 1;
}
```

```
-static inline void myri10ge_tx_done(struct myri10ge_priv *mgp, int mcp_index)
+static inline void
+myri10ge_tx_done(struct myri10ge_slice_state *ss, int mcp_index)
{
- struct pci_dev *pdev = mgp->pdev;
- struct myri10ge_tx_buf *tx = &mgp->tx;
+ struct pci_dev *pdev = ss->mgp->pdev;
+ struct myri10ge_tx_buf *tx = &ss->tx;
struct sk_buff *skb;
int idx, len;
```

```
@@ -1117,8 +1159,8 @@ static inline void myri10ge_tx_done(struct myri10ge_priv *mgp, int mcp_index)
len = pci_unmap_len(&tx->info[idx], len);
pci_unmap_len_set(&tx->info[idx], len, 0);
if (skb) {
- mgp->stats.tx_bytes += skb->len;
- mgp->stats.tx_packets++;
+ ss->stats.tx_bytes += skb->len;
+ ss->stats.tx_packets++;
dev_kfree_skb_irq(skb);
if (len)
pci_unmap_single(pdev,
```

```
@@ -1134,16 +1176,18 @@ static inline void myri10ge_tx_done(struct myri10ge_priv *mgp, int
mcp_index)
}
}
/* start the queue if we've stopped it */
- if (netif_queue_stopped(mgp->dev)
+ if (netif_queue_stopped(ss->dev)
&& tx->req - tx->done < (tx->mask >> 1)) {
- mgp->wake_queue++;
- netif_wake_queue(mgp->dev);
+ tx->wake_queue++;
+ netif_wake_queue(ss->dev);
}
}
```

```
-static inline int myri10ge_clean_rx_done(struct myri10ge_priv *mgp, int budget)
+static inline int
+myri10ge_clean_rx_done(struct myri10ge_slice_state *ss, int budget)
{
- struct myri10ge_rx_done *rx_done = &mgp->rx_done;
+ struct myri10ge_rx_done *rx_done = &ss->rx_done;
+ struct myri10ge_priv *mgp = ss->mgp;
unsigned long rx_bytes = 0;
unsigned long rx_packets = 0;
```

```

unsigned long rx_ok;
@@ -1159,40 +1203,40 @@ static inline int myri10ge_clean_rx_done(struct myri10ge_priv *mgp, int
budget)
rx_done->entry[idx].length = 0;
checksum = csum_unfold(rx_done->entry[idx].checksum);
if (length <= mgp->small_bytes)
- rx_ok = myri10ge_rx_done(mgp, &mgp->rx_small,
+ rx_ok = myri10ge_rx_done(ss, &ss->rx_small,
mgp->small_bytes,
length, checksum);
else
- rx_ok = myri10ge_rx_done(mgp, &mgp->rx_big,
+ rx_ok = myri10ge_rx_done(ss, &ss->rx_big,
mgp->big_bytes,
length, checksum);
rx_packets += rx_ok;
rx_bytes += rx_ok * (unsigned long)length;
cnt++;
- idx = cnt & (myri10ge_max_intr_slots - 1);
+ idx = cnt & (mgp->max_intr_slots - 1);
work_done++;
}
rx_done->idx = idx;
rx_done->cnt = cnt;
- mgp->stats.rx_packets += rx_packets;
- mgp->stats.rx_bytes += rx_bytes;
+ ss->stats.rx_packets += rx_packets;
+ ss->stats.rx_bytes += rx_bytes;

if (myri10ge_lro)
lro_flush_all(&rx_done->lro_mgr);

/* restock receive rings if needed */
- if (mgp->rx_small.fill_cnt - mgp->rx_small.cnt < myri10ge_fill_thresh)
- myri10ge_alloc_rx_pages(mgp, &mgp->rx_small,
+ if (ss->rx_small.fill_cnt - ss->rx_small.cnt < myri10ge_fill_thresh)
+ myri10ge_alloc_rx_pages(mgp, &ss->rx_small,
mgp->small_bytes + MXGEFW_PAD, 0);
- if (mgp->rx_big.fill_cnt - mgp->rx_big.cnt < myri10ge_fill_thresh)
- myri10ge_alloc_rx_pages(mgp, &mgp->rx_big, mgp->big_bytes, 0);
+ if (ss->rx_big.fill_cnt - ss->rx_big.cnt < myri10ge_fill_thresh)
+ myri10ge_alloc_rx_pages(mgp, &ss->rx_big, mgp->big_bytes, 0);

return work_done;
}

static inline void myri10ge_check_statblock(struct myri10ge_priv *mgp)
{
- struct mcp_irq_data *stats = mgp->fw_stats;
+ struct mcp_irq_data *stats = mgp->ss.fw_stats;

```

```

if (unlikely(stats->stats_updated)) {
unsigned link_up = ntohs(stats->link_up);
@@ -1219,9 +1263,9 @@ static inline void myri10ge_check_statblock(struct myri10ge_priv *mgp)
}
}
if (mgp->rdma_tags_available !=
- ntohs(mgp->fw_stats->rdma_tags_available)) {
+ ntohs(stats->rdma_tags_available)) {
mgp->rdma_tags_available =
- ntohs(mgp->fw_stats->rdma_tags_available);
+ ntohs(stats->rdma_tags_available);
printk(KERN_WARNING "myri10ge: %s: RDMA timed out! "
"%d tags left\n", mgp->dev->name,
mgp->rdma_tags_available);
@@ -1234,26 +1278,27 @@ static inline void myri10ge_check_statblock(struct myri10ge_priv *mgp)

static int myri10ge_poll(struct napi_struct *napi, int budget)
{
- struct myri10ge_priv *mgp =
- container_of(napi, struct myri10ge_priv, napi);
- struct net_device *netdev = mgp->dev;
+ struct myri10ge_slice_state *ss =
+ container_of(napi, struct myri10ge_slice_state, napi);
+ struct net_device *netdev = ss->mgp->dev;
int work_done;

/* process as many rx events as NAPI will allow */
- work_done = myri10ge_clean_rx_done(mgp, budget);
+ work_done = myri10ge_clean_rx_done(ss, budget);

if (work_done < budget) {
netif_rx_complete(netdev, napi);
- put_be32(htonl(3), mgp->irq_claim);
+ put_be32(htonl(3), ss->irq_claim);
}
return work_done;
}

static irqreturn_t myri10ge_intr(int irq, void *arg)
{
- struct myri10ge_priv *mgp = arg;
- struct mcp_irq_data *stats = mgp->fw_stats;
- struct myri10ge_tx_buf *tx = &mgp->tx;
+ struct myri10ge_slice_state *ss = arg;
+ struct myri10ge_priv *mgp = ss->mgp;
+ struct mcp_irq_data *stats = ss->fw_stats;
+ struct myri10ge_tx_buf *tx = &ss->tx;
u32 send_done_count;
int i;

@@ -1264,7 +1309,7 @@ static irqreturn_t myri10ge_intr(int irq, void *arg)

```

[git patches] net driver updates for .26

```
/* low bit indicates receives are present, so schedule
 * napi poll handler */
if (stats->valid & 1)
- netif_rx_schedule(mgp->dev, &mgp->napi);
```