```
**********************************************************
    2597 Thu Feb  2 21:11:31 2012
new/usr/src/pkg/manifests/driver-network-igb.mf
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2012, Nexenta Systems, Inc. All rights reserved.
#

#
# The default for payload-bearing actions in this package is to appear in the
# global zone only.  See the include file for greater detail, as well as
# information about overriding the defaults.
#
<include global_zone_only_component>
set name=pkg.fmri value=pkg:/driver/network/igb@$(PKGVERS)
set name=pkg.description value="Intel 82575 1Gb PCI Express NIC Driver"
set name=pkg.summary value="Intel 82575 1Gb PCI Express NIC Driver"
set name=info.classification \
    value=org.opensolaris.category.2008:Drivers/Networking
set name=variant.arch value=$(ARCH)
dir path=kernel group=sys
dir path=kernel/drv group=sys
dir path=kernel/drv/$(ARCH64) group=sys
dir path=usr/share/man
dir path=usr/share/man/man7d
driver name=igb clone_perms="igb 0666 root sys" perms="* 0666 root sys" \
    alias=pciex8086,10a7 \
    alias=pciex8086,10a9 \
    alias=pciex8086,10c9 \
    alias=pciex8086,10d6 \
    alias=pciex8086,10e6 \
    alias=pciex8086,10e7 \
    alias=pciex8086,10e8 \
    alias=pciex8086,150a \
    alias=pciex8086,150d \
    alias=pciex8086,150e \
    alias=pciex8086,150f \
    alias=pciex8086,1510 \
    alias=pciex8086,1511 \
    alias=pciex8086,1516 \
    alias=pciex8086,1518 \
    alias=pciex8086,1521 \
```



```
    alias=pciex8086,1526
    alias=pciex8086,1518
file path=kernel/drv/$(ARCH64)/igb group=sys
$(i386_ONLY)file path=kernel/drv/igb group=sys
file path=kernel/drv/igb.conf group=sys \
    original_name=SUNWigb:kernel/drv/igb.conf preserve=renamenew
file path=usr/share/man/man7d/igb.7d
legacy pkg=SUNWigb desc="Intel 82575 1Gb PCI Express NIC Driver" \
    name="Intel 82575 1Gb PCI Express NIC Driver"
license cr_Sun license=cr_Sun
license lic_CDDL license=lic_CDDL
```

```
**********************************************************
   60180 Thu Feb  2 21:11:32 2012
new/usr/src/uts/common/io/igb/igb_82575.c
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.146.2.2 v3_3_14_3_BHSW1 */

/*
 * 82575EB Gigabit Network Connection
 * 82575EB Gigabit Backplane Connection
 * 82575GB Gigabit Network Connection
 * 82576 Gigabit Network Connection
 * 82576 Quad Port Gigabit Mezzanine Adapter
 */

#include "igb_api.h"

static s32 e1000_init_phy_params_82575(struct e1000_hw *hw);
static s32 e1000_init_nvm_params_82575(struct e1000_hw *hw);
static s32 e1000_init_mac_params_82575(struct e1000_hw *hw);
static s32 e1000_acquire_phy_82575(struct e1000_hw *hw);
static void e1000_release_phy_82575(struct e1000_hw *hw);
static s32 e1000_acquire_nvm_82575(struct e1000_hw *hw);
static void e1000_release_nvm_82575(struct e1000_hw *hw);
static s32 e1000_check_for_link_82575(struct e1000_hw *hw);
static s32 e1000_get_cfg_done_82575(struct e1000_hw *hw);
static s32 e1000_get_link_up_info_82575(struct e1000_hw *hw, u16 *speed,
    u16 *duplex);
static s32 e1000_init_hw_82575(struct e1000_hw *hw);
static s32 e1000_phy_hw_reset_sgmii_82575(struct e1000_hw *hw);
static s32 e1000_read_phy_reg_sgmii_82575(struct e1000_hw *hw, u32 offset,
    u16 *data);
static s32 e1000_reset_hw_82575(struct e1000_hw *hw);
static s32 e1000_reset_hw_82580(struct e1000_hw *hw);
static s32 e1000_read_phy_reg_82580(struct e1000_hw *hw, u32 offset,
```



```
    u16 *data);
static s32 e1000_write_phy_reg_82580(struct e1000_hw *hw, u32 offset,
    u16 data);
static s32 e1000_set_d0_lplu_state_82575(struct e1000_hw *hw,
    bool active);
static s32 e1000_setup_copper_link_82575(struct e1000_hw *hw);
static s32 e1000_setup_serdes_link_82575(struct e1000_hw *hw);
static s32 e1000_valid_led_default_82575(struct e1000_hw *hw, u16 *data);
static s32 e1000_write_phy_reg_sgmii_82575(struct e1000_hw *hw,
    u32 offset, u16 data);
static void e1000_clear_hw_cntrs_82575(struct e1000_hw *hw);
static s32 e1000_acquire_swfw_sync_82575(struct e1000_hw *hw, u16 mask);
static s32 e1000_get_pcs_speed_and_duplex_82575(struct e1000_hw *hw,
    u16 *speed, u16 *duplex);
static s32 e1000_get_phy_id_82575(struct e1000_hw *hw);
static void e1000_release_swfw_sync_82575(struct e1000_hw *hw, u16 mask);
static bool e1000_sgmii_active_82575(struct e1000_hw *hw);
static s32 e1000_reset_init_script_82575(struct e1000_hw *hw);
static s32 e1000_read_mac_addr_82575(struct e1000_hw *hw);
static void e1000_power_down_phy_copper_82575(struct e1000_hw *hw);
static void e1000_shutdown_serdes_link_82575(struct e1000_hw *hw);
static s32 e1000_set_pcie_completion_timeout(struct e1000_hw *hw);

static s32 e1000_update_nvm_checksum_with_offset(struct e1000_hw *hw,
    u16 offset);
static s32 e1000_validate_nvm_checksum_with_offset(struct e1000_hw *hw,
    u16 offset);
static s32 e1000_validate_nvm_checksum_i350(struct e1000_hw *hw);
static s32 e1000_update_nvm_checksum_i350(struct e1000_hw *hw);
static void e1000_write_vfta_i350(struct e1000_hw *hw, u32 offset, u32 value);
static void e1000_clear_vfta_i350(struct e1000_hw *hw);

static const u16 e1000_82580_rxpbs_table[] =
        {36, 72, 144, 1, 2, 4, 8, 16, 35, 70, 140};
#define E1000_82580_RXPBS_TABLE_SIZE \
        (sizeof (e1000_82580_rxpbs_table)/sizeof (u16))

/*
 * e1000_init_phy_params_82575 - Init PHY func ptrs.
 * @hw: pointer to the HW structure
 */
static s32
e1000_init_phy_params_82575(struct e1000_hw *hw)
{
        struct e1000_phy_info *phy = &hw->phy;
        s32 ret_val = E1000_SUCCESS;

        DEBUGFUNC("e1000_init_phy_params_82575");

        if (hw->phy.media_type != e1000_media_type_copper) {
                phy->type = e1000_phy_none;
                goto out;
        }

        phy->ops.power_up = e1000_power_up_phy_copper;
        phy->ops.power_down = e1000_power_down_phy_copper_82575;

        phy->autoneg_mask = AUTONEG_ADVERTISE_SPEED_DEFAULT;
        phy->reset_delay_us = 100;

        phy->ops.acquire = e1000_acquire_phy_82575;
        phy->ops.check_reset_block = e1000_check_reset_block_generic;
        phy->ops.commit = e1000_phy_sw_reset_generic;
        phy->ops.get_cfg_done = e1000_get_cfg_done_82575;
        phy->ops.release = e1000_release_phy_82575;
```

```
126         if (e1000_sgmii_active_82575(hw)) {
127                 phy->ops.reset = e1000_phy_hw_reset_sgmii_82575;
128                 phy->ops.read_reg = e1000_read_phy_reg_sgmii_82575;
129                 phy->ops.write_reg = e1000_write_phy_reg_sgmii_82575;
130         } else if (hw->mac.type == e1000_82580) {
131                 phy->ops.reset = e1000_phy_hw_reset_generic;
132                 phy->ops.read_reg = e1000_read_phy_reg_82580;
133                 phy->ops.write_reg = e1000_write_phy_reg_82580;
134         } else {
135                 phy->ops.reset = e1000_phy_hw_reset_generic;
136                 phy->ops.read_reg = e1000_read_phy_reg_igp;
137                 phy->ops.write_reg = e1000_write_phy_reg_igp;
138         }

140         /* Set phy->phy_addr and phy->id. */
141         ret_val = e1000_get_phy_id_82575(hw);

143         /* Verify phy id and set remaining function pointers */
144         switch (phy->id) {
145         case M88E1111_I_PHY_ID:
146                 phy->type = e1000_phy_m88;
147                 phy->ops.check_polarity = e1000_check_polarity_m88;
148                 phy->ops.get_info = e1000_get_phy_info_m88;
149                 phy->ops.get_cable_length = e1000_get_cable_length_m88;
150                 phy->ops.force_speed_duplex = e1000_phy_force_speed_duplex_m88;
151                 break;
152         case IGP03E1000_E_PHY_ID:
153         case IGP04E1000_E_PHY_ID:
154                 phy->type = e1000_phy_igp_3;
155                 phy->ops.check_polarity = e1000_check_polarity_igp;
156                 phy->ops.get_info = e1000_get_phy_info_igp;
157                 phy->ops.get_cable_length = e1000_get_cable_length_igp_2;
158                 phy->ops.force_speed_duplex = e1000_phy_force_speed_duplex_igp;
159                 phy->ops.set_d0_lplu_state = e1000_set_d0_lplu_state_82575;
160                 phy->ops.set_d3_lplu_state = e1000_set_d3_lplu_state_generic;
161                 break;
162         case I82580_I_PHY_ID:
163         case I350_I_PHY_ID:
164                 phy->type = e1000_phy_82580;
165                 phy->ops.check_polarity = e1000_check_polarity_82577;
166                 phy->ops.force_speed_duplex =
167                     e1000_phy_force_speed_duplex_82577;
168                 phy->ops.get_cable_length = e1000_get_cable_length_82577;
169                 phy->ops.get_info = e1000_get_phy_info_82577;
170                 break;
171         default:
172                 ret_val = -E1000_ERR_PHY;
173                 goto out;
174         }

176 out:
177         return (ret_val);
178 }

180 /*
181  * e1000_init_nvm_params_82575 - Init NVM func ptrs.
182  * @hw: pointer to the HW structure
183  */
184 static s32
185 e1000_init_nvm_params_82575(struct e1000_hw *hw)
186 {
187         struct e1000_nvm_info *nvm = &hw->nvm;
188         u32 eecd = E1000_READ_REG(hw, E1000_EECD);
189         u16 size;

191         DEBUGFUNC("e1000_init_nvm_params_82575");
```



```
193         nvm->opcode_bits = 8;
194         nvm->delay_usec = 1;
195         switch (nvm->override) {
196         case e1000_nvm_override_spi_large:
197                 nvm->page_size = 32;
198                 nvm->address_bits = 16;
199                 break;
200         case e1000_nvm_override_spi_small:
201                 nvm->page_size = 8;
202                 nvm->address_bits = 8;
203                 break;
204         default:
205                 nvm->page_size = eecd & E1000_EECD_ADDR_BITS ? 32 : 8;
206                 nvm->address_bits = eecd & E1000_EECD_ADDR_BITS ? 16 : 8;
207                 break;
208         }

210         nvm->type = e1000_nvm_eeprom_spi;

212         size = (u16)((eecd & E1000_EECD_SIZE_EX_MASK) >>
213             E1000_EECD_SIZE_EX_SHIFT);

215         /*
216          * Added to a constant, "size" becomes the left-shift value
217          * for setting word_size.
218          */
219         size += NVM_WORD_SIZE_BASE_SHIFT;

221         /* EEPROM access above 16k is unsupported */
222         if (size > 14)
223                 size = 14;
224         nvm->word_size = 1 << size;

226         /* Function Pointers */
227         nvm->ops.acquire = e1000_acquire_nvm_82575;
228         nvm->ops.read = e1000_read_nvm_eerd;
229         nvm->ops.release = e1000_release_nvm_82575;
230         nvm->ops.update = e1000_update_nvm_checksum_generic;
231         nvm->ops.valid_led_default = e1000_valid_led_default_82575;
232         nvm->ops.validate = e1000_validate_nvm_checksum_generic;
233         nvm->ops.write = e1000_write_nvm_spi;

235         /* override genric family function pointers for specific descendants */
236         switch (hw->mac.type) {
237         case e1000_i350:
238                 nvm->ops.validate = e1000_validate_nvm_checksum_i350;
239                 nvm->ops.update = e1000_update_nvm_checksum_i350;
240                 break;
241         default:
242                 break;
243         }

246         return (E1000_SUCCESS);
247 }

249 /*
250  * e1000_init_mac_params_82575 - Init MAC func ptrs.
251  * @hw: pointer to the HW structure
252  */
253 static s32
254 e1000_init_mac_params_82575(struct e1000_hw *hw)
255 {
256         struct e1000_mac_info *mac = &hw->mac;
257         struct e1000_dev_spec_82575 *dev_spec = &hw->dev_spec._82575;
```

```
258         u32 ctrl_ext = 0;

260         DEBUGFUNC("e1000_init_mac_params_82575");

262         /* Set media type */
263         /*
264          * The 82575 uses bits 22:23 for link mode. The mode can be changed
265          * based on the EEPROM. We cannot rely upon device ID. There
266          * is no distinguishable difference between fiber and internal
267          * SerDes mode on the 82575. There can be an external PHY attached
268          * on the SGMII interface. For this, we'll set sgmii_active to true.
269          */
270         hw->phy.media_type = e1000_media_type_copper;
271         dev_spec->sgmii_active = false;

273         ctrl_ext = E1000_READ_REG(hw, E1000_CTRL_EXT);
274         switch (ctrl_ext & E1000_CTRL_EXT_LINK_MODE_MASK) {
275         case E1000_CTRL_EXT_LINK_MODE_SGMII:
276                 dev_spec->sgmii_active = true;
277                 ctrl_ext |= E1000_CTRL_I2C_ENA;
278                 break;
279         case E1000_CTRL_EXT_LINK_MODE_1000BASE_KX:
280         case E1000_CTRL_EXT_LINK_MODE_PCIE_SERDES:
281                 hw->phy.media_type = e1000_media_type_internal_serdes;
282                 ctrl_ext |= E1000_CTRL_I2C_ENA;
283                 break;
284         default:
285                 ctrl_ext &= ~E1000_CTRL_I2C_ENA;
286                 break;
287         }

289         E1000_WRITE_REG(hw, E1000_CTRL_EXT, ctrl_ext);

291         /*
292          * if using i2c make certain the MDICNFG register is cleared to prevent
293          * communications from being misrouted to the mdic registers
294          */
295         if ((ctrl_ext & E1000_CTRL_I2C_ENA) && (hw->mac.type == e1000_82580))
296                 E1000_WRITE_REG(hw, E1000_MDICNFG, 0);

298         /* Set mta register count */
299         mac->mta_reg_count = 128;
300         /* Set uta register count */
301         mac->uta_reg_count = (hw->mac.type == e1000_82575) ? 0 : 128;
302         /* Set rar entry count */
303         mac->rar_entry_count = E1000_RAR_ENTRIES_82575;
304         if (mac->type == e1000_82576)
305                 mac->rar_entry_count = E1000_RAR_ENTRIES_82576;
306         if (mac->type == e1000_82580)
307                 mac->rar_entry_count = E1000_RAR_ENTRIES_82580;
308         if (mac->type == e1000_i350) {
309                 mac->rar_entry_count = E1000_RAR_ENTRIES_I350;
310                 /* Enable EEE default settings for i350 */
311                 dev_spec->eee_disable = B_FALSE;
312         }
313         /* Set if part includes ASF firmware */
314         mac->asf_firmware_present = true;
315         /* Set if manageability features are enabled. */
316         mac->arc_subsystem_valid =
317             (E1000_READ_REG(hw, E1000_FWSM) & E1000_FWSM_MODE_MASK)
318             ? true : false;

320         /* Function pointers */

322         /* bus type/speed/width */
323         mac->ops.get_bus_info = e1000_get_bus_info_pcie_generic;
```



```
324         /* reset */
325         if (mac->type == e1000_82580)
326                 mac->ops.reset_hw = e1000_reset_hw_82580;
327         else
328                 mac->ops.reset_hw = e1000_reset_hw_82575;
329         /* hw initialization */
330         mac->ops.init_hw = e1000_init_hw_82575;
331         /* link setup */
332         mac->ops.setup_link = e1000_setup_link_generic;
333         /* physical interface link setup */
334         mac->ops.setup_physical_interface =
335             (hw->phy.media_type == e1000_media_type_copper)
336             ? e1000_setup_copper_link_82575
337             : e1000_setup_serdes_link_82575;
338         /* physical interface shutdown */
339         mac->ops.shutdown_serdes = e1000_shutdown_serdes_link_82575;
340         /* check for link */
341         mac->ops.check_for_link = e1000_check_for_link_82575;
342         /* receive address register setting */
343         mac->ops.rar_set = e1000_rar_set_generic;
344         /* read mac address */
345         mac->ops.read_mac_addr = e1000_read_mac_addr_82575;
346         /* multicast address update */
347         mac->ops.update_mc_addr_list = e1000_update_mc_addr_list_generic;

349         if (hw->mac.type == e1000_i350) {
350                 /* writing VFTA */
351                 mac->ops.write_vfta = e1000_write_vfta_i350;
352                 /* clearing VFTA */
353                 mac->ops.clear_vfta = e1000_clear_vfta_i350;
354         } else {
355                 /* writing VFTA */
356                 mac->ops.write_vfta = e1000_write_vfta_generic;
357                 /* clearing VFTA */
358                 mac->ops.clear_vfta = e1000_clear_vfta_generic;
359         }
360         /* setting MTA */
361         mac->ops.mta_set = e1000_mta_set_generic;
362         /* ID LED init */
363         mac->ops.id_led_init = e1000_id_led_init_generic;
364         /* blink LED */
365         mac->ops.blink_led = e1000_blink_led_generic;
366         /* setup LED */
367         mac->ops.setup_led = e1000_setup_led_generic;
368         /* cleanup LED */
369         mac->ops.cleanup_led = e1000_cleanup_led_generic;
370         /* turn on/off LED */
371         mac->ops.led_on = e1000_led_on_generic;
372         mac->ops.led_off = e1000_led_off_generic;
373         /* clear hardware counters */
374         mac->ops.clear_hw_cntrs = e1000_clear_hw_cntrs_82575;
375         /* link info */
376         mac->ops.get_link_up_info = e1000_get_link_up_info_82575;

378         /* set lan id for port to determine which phy lock to use */
379         hw->mac.ops.set_lan_id(hw);

381         return (E1000_SUCCESS);
382 }
```

*_____unchanged_portion_omitted_*

```
714 /*
715  * e1000_acquire_nvm_82575 - Request for access to EEPROM
716  * @hw: pointer to the HW structure
717  *
718  * Acquire the necessary semaphores for exclusive access to the EEPROM.
```

```
 719  * Set the EEPROM access request bit and wait for EEPROM access grant bit.
 720  * Return successful if access grant bit set, else clear the request for
 721  * EEPROM access and return -E1000_ERR_NVM (-1).
 722  */
 723 static s32
 724 e1000_acquire_nvm_82575(struct e1000_hw *hw)
 725 {
 726         s32 ret_val;

 728         DEBUGFUNC("e1000_acquire_nvm_82575");

 730         ret_val = e1000_acquire_swfw_sync_82575(hw, E1000_SWFW_EEP_SM);
 731         if (ret_val)
 732                 goto out;

 734         /*
 735          * Check if there is some access
 736          * error this access may hook on
 737          */
 738         if (hw->mac.type == e1000_i350) {
 739                 u32 eecd = E1000_READ_REG(hw, E1000_EECD);
 740                 if (eecd & (E1000_EECD_BLOCKED | E1000_EECD_ABORT |
 741                     E1000_EECD_TIMEOUT)) {
 742                         /* Clear all access error flags */
 743                         E1000_WRITE_REG(hw, E1000_EECD, eecd |
 744                             E1000_EECD_ERROR_CLR);
 745                         DEBUGOUT("Nvm bit banging access error "
 746                             "detected and cleared.\n");
 747                 }
 748         }

 750         ret_val = e1000_acquire_nvm_generic(hw);

 752         if (ret_val)
 753                 e1000_release_swfw_sync_82575(hw, E1000_SWFW_EEP_SM);

 755 out:
 756         return (ret_val);
 757 }
```

*unchanged_portion_omitted_*

```
1899 /*
1900  * e1000_rxpbs_adjust_82580 - adjust RXPBS value to reflect actual RX PBA size
1901  * @data: data received by reading RXPBS register
1902  *
1903  * The 82580 uses a table based approach for packet buffer allocation sizes.
1904  * This function converts the retrieved value into the correct table value
1905  *    0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7
1906  * 0x0 36  72 144   1   2   4   8  16
1907  * 0x8 35  70 140 rsv rsv rsv rsv rsv
1908  */
1909 u16
1910 e1000_rxpbs_adjust_82580(u32 data)
1911 {
1912         u16 ret_val = 0;

1914         if (data < E1000_82580_RXPBS_TABLE_SIZE)
1915                 ret_val = e1000_82580_rxpbs_table[data];

1917         return (ret_val);
1918 }

1920 /*
1921  * Due to a hw errata, if the host tries to  configure the VFTA register
1922  * while performing queries from the BMC or DMA, then the VFTA in some
1923  * cases won't be written.
```



```
1924  */

1926 /*
1927  *  e1000_clear_vfta_i350 - Clear VLAN filter table
1928  *  @hw: pointer to the HW structure
1929  *
1930  *  Clears the register array which contains the VLAN filter table by
1931  *  setting all the values to 0.
1932  */
1933 void
1934 e1000_clear_vfta_i350(struct e1000_hw *hw)
1935 {
1936         u32 offset;
1937         int i;

1939         DEBUGFUNC("e1000_clear_vfta_350");

1941         for (offset = 0; offset < E1000_VLAN_FILTER_TBL_SIZE; offset++) {
1942                 for (i = 0; i < 10; i++)
1943                         E1000_WRITE_REG_ARRAY(hw, E1000_VFTA, offset, 0);

1945                 E1000_WRITE_FLUSH(hw);
1946         }
1947 }

1949 /*
1950  *  e1000_write_vfta_i350 - Write value to VLAN filter table
1951  *  @hw: pointer to the HW structure
1952  *  @offset: register offset in VLAN filter table
1953  *  @value: register value written to VLAN filter table
1954  *
1955  *  Writes value at the given offset in the register array which stores
1956  *  the VLAN filter table.
1957  */
1958 void
1959 e1000_write_vfta_i350(struct e1000_hw *hw, u32 offset, u32 value)
1960 {
1961         int i;

1963         DEBUGFUNC("e1000_write_vfta_350");

1965         for (i = 0; i < 10; i++)
1966                 E1000_WRITE_REG_ARRAY(hw, E1000_VFTA, offset, value);

1968         E1000_WRITE_FLUSH(hw);
1969 }

1971 /*
1972  *  e1000_validate_nvm_checksum_with_offset - Validate EEPROM
1973  *  checksum
1974  *  @hw: pointer to the HW structure
1975  *  @offset: offset in words of the checksum protected region
1976  *
1977  *  Calculates the EEPROM checksum by reading/adding each word of the EEPROM
1978  *  and then verifies that the sum of the EEPROM is equal to 0xBABA.
1979  */
1980 s32
1981 e1000_validate_nvm_checksum_with_offset(struct e1000_hw *hw, u16 offset)
1982 {
1983         s32 ret_val = E1000_SUCCESS;
1984         u16 checksum = 0;
1985         u16 i, nvm_data;

1987         DEBUGFUNC("e1000_validate_nvm_checksum_with_offset");

1989         for (i = offset; i < ((NVM_CHECKSUM_REG + offset) + 1); i++) {
```

```
1990                 ret_val = hw->nvm.ops.read(hw, i, 1, &nvm_data);
1991                 if (ret_val) {
1992                         DEBUGOUT("NVM Read Error\n");
1993                         goto out;
1994                 }
1995                 checksum += nvm_data;
1996         }

1998         if (checksum != (u16) NVM_SUM) {
1999                 DEBUGOUT("NVM Checksum Invalid\n");
2000                 ret_val = -E1000_ERR_NVM;
2001                 goto out;
2002         }

2004 out:
2005         return (ret_val);
2006 }

2008 /*
2009  *  e1000_update_nvm_checksum_with_offset - Update EEPROM
2010  *  checksum
2011  *  @hw: pointer to the HW structure
2012  *  @offset: offset in words of the checksum protected region
2013  *
2014  *  Updates the EEPROM checksum by reading/adding each word of the EEPROM
2015  *  up to the checksum.  Then calculates the EEPROM checksum and writes the
2016  *  value to the EEPROM.
2017  */
2018 s32
2019 e1000_update_nvm_checksum_with_offset(struct e1000_hw *hw, u16 offset)
2020 {
2021         s32 ret_val;
2022         u16 checksum = 0;
2023         u16 i, nvm_data;

2025         DEBUGFUNC("e1000_update_nvm_checksum_with_offset");

2027         for (i = offset; i < (NVM_CHECKSUM_REG + offset); i++) {
2028                 ret_val = hw->nvm.ops.read(hw, i, 1, &nvm_data);
2029                 if (ret_val) {
2030                         DEBUGOUT("NVM Read Error while updating checksum.\n");
2031                         goto out;
2032                 }
2033                 checksum += nvm_data;
2034         }
2035         checksum = (u16) NVM_SUM - checksum;
2036         ret_val = hw->nvm.ops.write(hw, (NVM_CHECKSUM_REG + offset), 1,
2037             &checksum);
2038         if (ret_val)
2039                 DEBUGOUT("NVM Write Error while updating checksum.\n");

2041 out:
2042         return (ret_val);
2043 }

2045 /*
2046  *  e1000_validate_nvm_checksum_i350 - Validate EEPROM checksum
2047  *  @hw: pointer to the HW structure
2048  *
2049  *  Calculates the EEPROM section checksum by reading/adding each word of
2050  *  the EEPROM and then verifies that the sum of the EEPROM is
2051  *  equal to 0xBABA.
2052  */
2053 static s32
2054 e1000_validate_nvm_checksum_i350(struct e1000_hw *hw)
2055 {
```



```
2056         s32 ret_val = E1000_SUCCESS;
2057         u16 j;
2058         u16 nvm_offset;

2060         DEBUGFUNC("e1000_validate_nvm_checksum_i350");

2062         for (j = 0; j < 4; j++) {
2063                 nvm_offset = NVM_82580_LAN_FUNC_OFFSET(j);
2064                 ret_val = e1000_validate_nvm_checksum_with_offset(hw,
2065                     nvm_offset);
2066                 if (ret_val != E1000_SUCCESS)
2067                         goto out;
2068         }

2070 out:
2071         return (ret_val);
2072 }

2074 /*
2075  *  e1000_update_nvm_checksum_i350 - Update EEPROM checksum
2076  *  @hw: pointer to the HW structure
2077  *
2078  *  Updates the EEPROM section checksums for all 4 ports by reading/adding
2079  *  each word of the EEPROM up to the checksum.  Then calculates the EEPROM
2080  *  checksum and writes the value to the EEPROM.
2081  */
2082 static s32
2083 e1000_update_nvm_checksum_i350(struct e1000_hw *hw)
2084 {
2085         s32 ret_val = E1000_SUCCESS;
2086         u16 j;
2087         u16 nvm_offset;

2089         DEBUGFUNC("e1000_update_nvm_checksum_i350");

2091         for (j = 0; j < 4; j++) {
2092                 nvm_offset = NVM_82580_LAN_FUNC_OFFSET(j);
2093                 ret_val = e1000_update_nvm_checksum_with_offset(hw, nvm_offset);
2094                 if (ret_val != E1000_SUCCESS)
2095                         goto out;
2096         }

2098 out:
2099         return (ret_val);
2100 }



2104 /*
2105  *  e1000_set_eee_i350 - Enable/disable EEE support
2106  *  @hw: pointer to the HW structure
2107  *
2108  *  Enable/disable EEE based on setting in dev_spec structure.
2109  *
2110  */
2111 s32
2112 e1000_set_eee_i350(struct e1000_hw *hw)
2113 {

2115         s32 ret_val = E1000_SUCCESS;
2116         u32 ipcnfg, eeer;

2118         DEBUGFUNC("e1000_set_eee_i350");

2120         if ((hw->mac.type < e1000_i350) ||
2121             (hw->phy.media_type != e1000_media_type_copper))
```

```
2122                 goto out;
2123         ipcnfg = E1000_READ_REG(hw, E1000_IPCNFG);
2124         eeer = E1000_READ_REG(hw, E1000_EEER);

2126         /* enable or disable per user setting */
2127         if (!(hw->dev_spec._82575.eee_disable)) {
2128                 ipcnfg |= (E1000_IPCNFG_EEE_1G_AN | E1000_IPCNFG_EEE_100M_AN);
2129                 eeer |= (E1000_EEER_TX_LPI_EN | E1000_EEER_RX_LPI_EN |
2130                     E1000_EEER_LPI_FC);

2132         } else {
2133                 ipcnfg &= ~(E1000_IPCNFG_EEE_1G_AN | E1000_IPCNFG_EEE_100M_AN);
2134                 eeer &= ~(E1000_EEER_TX_LPI_EN | E1000_EEER_RX_LPI_EN |
2135                     E1000_EEER_LPI_FC);
2136         }
2137         E1000_WRITE_REG(hw, E1000_IPCNFG, ipcnfg);
2138         E1000_WRITE_REG(hw, E1000_EEER, eeer);
2139         ipcnfg = E1000_READ_REG(hw, E1000_IPCNFG);
2140         eeer = E1000_READ_REG(hw, E1000_EEER);
2141 out:

2143         return (ret_val);
2144 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   18027 Thu Feb  2 21:11:33 2012
new/usr/src/uts/common/io/igb/igb_82575.h
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.88.2.1 v3_3_14_3_BHSW1 */

#ifndef _IGB_82575_H
#define _IGB_82575_H

#ifdef __cplusplus
extern "C" {
#endif

#define ID_LED_DEFAULT_82575_SERDES     ((ID_LED_DEF1_DEF2 << 12) | \
                                        (ID_LED_DEF1_DEF2 <<  8) | \
                                        (ID_LED_DEF1_DEF2 <<  4) | \
                                        (ID_LED_OFF1_ON2))

/*
 * Receive Address Register Count
 * Number of high/low register pairs in the RAR.  The RAR (Receive Address
 * Registers) holds the directed and multicast addresses that we monitor.
 * These entries are also used for MAC-based filtering.
 */
/*
 * For 82576, there are an additional set of RARs that begin at an offset
 * separate from the first set of RARs.
 */
#define E1000_RAR_ENTRIES_82575         16
#define E1000_RAR_ENTRIES_82576         24
#define E1000_RAR_ENTRIES_82580         24
#define E1000_RAR_ENTRIES_I350          32
#define E1000_SW_SYNCH_MB               0x00000100
#define E1000_STAT_DEV_RST_SET          0x00100000
```



```
#define E1000_CTRL_DEV_RST              0x20000000

#ifdef E1000_BIT_FIELDS
struct e1000_adv_data_desc {
        __le64 buffer_addr;     /* Address of the descriptor's data buffer */
        union {
                u32 data;
                struct {
                        u32 datalen     :16;    /* Data buffer length */
                        u32 rsvd        :4;
                        u32 dtyp        :4;     /* Descriptor type */
                        u32 dcmd        :8;     /* Descriptor command */
                } config;
        } lower;
        union {
                u32 data;
                struct {
                        u32 status      :4;     /* Descriptor status */
                        u32 idx         :4;
                        u32 popts       :6;     /* Packet Options */
                        u32 paylen      :18;    /* Payload length */
                } options;
        } upper;
};
______unchanged_portion_omitted_

#define E1000_ADVTXD_MACLEN_SHIFT       9 /* Adv ctxt desc mac len shift */
#define E1000_ADVTXD_VLAN_SHIFT         16 /* Adv ctxt vlan tag shift */
#define E1000_ADVTXD_TUCMD_IPV4         0x00000400 /* IP Packet Type: 1=IPv4 */
#define E1000_ADVTXD_TUCMD_IPV6         0x00000000 /* IP Packet Type: 0=IPv6 */
#define E1000_ADVTXD_TUCMD_L4T_UDP      0x00000000 /* L4 Packet TYPE of UDP */
#define E1000_ADVTXD_TUCMD_L4T_TCP      0x00000800 /* L4 Packet TYPE of TCP */
#define E1000_ADVTXD_TUCMD_L4T_SCTP     0x00001000 /* L4 Packet TYPE of SCTP */
#define E1000_ADVTXD_TUCMD_IPSEC_TYPE_ESP       0x00002000 /* IPSec Type ESP */
/* IPSec Encrypt Enable for ESP */
#define E1000_ADVTXD_TUCMD_IPSEC_ENCRYPT_EN     0x00004000
/* Req requires Markers and CRC */
#define E1000_ADVTXD_TUCMD_MKRREQ       0x00002000
#define E1000_ADVTXD_L4LEN_SHIFT        8       /* Adv ctxt L4LEN shift */
#define E1000_ADVTXD_MSS_SHIFT          16      /* Adv ctxt MSS shift */
/* Adv ctxt IPSec SA IDX mask */
#define E1000_ADVTXD_IPSEC_SA_INDEX_MASK        0x000000FF
/* Adv ctxt IPSec ESP len mask */
#define E1000_ADVTXD_IPSEC_ESP_LEN_MASK         0x000000FF

/* Additional Transmit Descriptor Control definitions */
/* Enable specific Tx Queue */
#define E1000_TXDCTL_QUEUE_ENABLE       0x02000000
/* Tx Desc. write-back flushing */
#define E1000_TXDCTL_SWFLSH             0x04000000
/* Tx Queue Arbitration Priority 0=low, 1=high */
#define E1000_TXDCTL_PRIORITY           0x08000000

/* Additional Receive Descriptor Control definitions */
/* Enable specific Rx Queue */
#define E1000_RXDCTL_QUEUE_ENABLE       0x02000000
/* Rx Desc. write-back flushing */
#define E1000_RXDCTL_SWFLSH             0x04000000

/* Direct Cache Access (DCA) definitions */
#define E1000_DCA_CTRL_DCA_ENABLE       0x00000000      /* DCA Enable */
#define E1000_DCA_CTRL_DCA_DISABLE      0x00000001      /* DCA Disable */

#define E1000_DCA_CTRL_DCA_MODE_CB1     0x00    /* DCA Mode CB1 */
#define E1000_DCA_CTRL_DCA_MODE_CB2     0x02    /* DCA Mode CB2 */
```

```
365 #define E1000_DCA_RXCTRL_CPUID_MASK 0x0000001F  /* Rx CPUID Mask */
366 #define E1000_DCA_RXCTRL_DESC_DCA_EN (1 << 5) /* DCA Rx Desc enable */
367 #define E1000_DCA_RXCTRL_HEAD_DCA_EN (1 << 6) /* DCA Rx Desc header enable */
368 #define E1000_DCA_RXCTRL_DATA_DCA_EN (1 << 7) /* DCA Rx Desc payload enable */

370 #define E1000_DCA_TXCTRL_CPUID_MASK 0x0000001F  /* Tx CPUID Mask */
371 #define E1000_DCA_TXCTRL_DESC_DCA_EN (1 << 5)   /* DCA Tx Desc enable */
372 #define E1000_DCA_TXCTRL_TX_WB_RO_EN (1 << 11)  /* Tx Desc writeback RO bit */

374 #define E1000_DCA_TXCTRL_CPUID_MASK_82576 0xFF000000 /* Tx CPUID Mask */
375 #define E1000_DCA_RXCTRL_CPUID_MASK_82576 0xFF000000 /* Rx CPUID Mask */
376 #define E1000_DCA_TXCTRL_CPUID_SHIFT_82576 24 /* Tx CPUID */
377 #define E1000_DCA_RXCTRL_CPUID_SHIFT_82576 24 /* Rx CPUID */

379 /* Additional interrupt register bit definitions */
380 #define E1000_ICR_LSECPNS       0x00000020      /* PN threshold - server */
381 #define E1000_IMS_LSECPNS       E1000_ICR_LSECPNS /* PN threshold - server */
382 #define E1000_ICS_LSECPNS       E1000_ICR_LSECPNS /* PN threshold - server */

384 /* ETQF register bit definitions */
385 #define E1000_ETQF_FILTER_ENABLE        (1 << 26)
386 #define E1000_ETQF_IMM_INT              (1 << 29)
387 #define E1000_ETQF_1588                 (1 << 30)
388 #define E1000_ETQF_QUEUE_ENABLE         (1 << 31)
389 /*
390  * ETQF filter list: one static filter per filter consumer. This is
391  *              to avoid filter collisions later. Add new filters
392  *              here!!
393  *
394  * Current filters:
395  *    EAPOL 802.1x (0x888e): Filter 0
396  */
397 #define E1000_ETQF_FILTER_EAPOL         0

399 #define E1000_FTQF_VF_BP                0x00008000
400 #define E1000_FTQF_1588_TIME_STAMP      0x08000000
401 #define E1000_FTQF_MASK                 0xF0000000
402 #define E1000_FTQF_MASK_PROTO_BP        0x10000000
403 #define E1000_FTQF_MASK_SOURCE_ADDR_BP  0x20000000
404 #define E1000_FTQF_MASK_DEST_ADDR_BP    0x40000000
405 #define E1000_FTQF_MASK_SOURCE_PORT_BP  0x80000000

407 #define E1000_NVM_APME_82575            0x0400
408 #define MAX_NUM_VFS                     8

410 /* Per VF MAC spoof control */
411 #define E1000_DTXSWC_MAC_SPOOF_MASK     0x000000FF
412 /* Per VF VLAN spoof control */
413 #define E1000_DTXSWC_VLAN_SPOOF_MASK    0x0000FF00
414 #define E1000_DTXSWC_LLE_MASK           0x00FF0000 /* Per VF Local LB enables */
415 #define E1000_DTXSWC_VLAN_SPOOF_SHIFT   8
416 #define E1000_DTXSWC_LLE_SHIFT          16
417 #define E1000_DTXSWC_VMDQ_LOOPBACK_EN   ((u32)1 << 31) /* global VF LB enable */

419 /* Easy defines for setting default pool, would normally be left a zero */
420 #define E1000_VT_CTL_DEFAULT_POOL_SHIFT 7
421 #define E1000_VT_CTL_DEFAULT_POOL_MASK  (0x7 << E1000_VT_CTL_DEFAULT_POOL_SHIFT)

423 /* Other useful VMD_CTL register defines */
424 #define E1000_VT_CTL_IGNORE_MAC         (1 << 28)
425 #define E1000_VT_CTL_DISABLE_DEF_POOL   (1 << 29)
426 #define E1000_VT_CTL_VM_REPL_EN         (1 << 30)

428 /* Per VM Offload register setup */
429 #define E1000_VMOLR_RLPML_MASK  0x00003FFF /* Long Packet Maximum Length mask */
430 #define E1000_VMOLR_LPE         0x00010000 /* Accept Long packet */
```



```
431 #define E1000_VMOLR_RSSE        0x00020000 /* Enable RSS */
432 #define E1000_VMOLR_AUPE        0x01000000 /* Accept untagged packets */
433 #define E1000_VMOLR_ROMPE       0x02000000 /* Accept overflow multicast */
434 #define E1000_VMOLR_ROPE        0x04000000 /* Accept overflow unicast */
435 #define E1000_VMOLR_BAM         0x08000000 /* Accept Broadcast packets */
436 #define E1000_VMOLR_MPME        0x10000000 /* Multicast promiscuous mode */
437 #define E1000_VMOLR_STRVLAN     0x40000000 /* Vlan stripping enable */
438 #define E1000_VMOLR_STRCRC      0x80000000 /* CRC stripping enable */

440 #define E1000_VLVF_ARRAY_SIZE           32
441 #define E1000_VLVF_VLANID_MASK          0x00000FFF
442 #define E1000_VLVF_POOLSEL_SHIFT        12
443 #define E1000_VLVF_POOLSEL_MASK         (0xFF << E1000_VLVF_POOLSEL_SHIFT)
444 #define E1000_VLVF_LVLAN                0x00100000
445 #define E1000_VLVF_VLANID_ENABLE        0x80000000

447 #define E1000_VMVIR_VLANA_DEFAULT 0x40000000 /* Always use default VLAN */
448 #define E1000_VMVIR_VLANA_NEVER   0x80000000 /* Never insert VLAN tag */
449 #define E1000_VF_INIT_TIMEOUT   200 /* Number of retries to clear RSTI */

451 #define E1000_IOVCTL            0x05BBC
452 #define E1000_IOVCTL_REUSE_VFQ  0x00000001

454 #define E1000_RPLOLR_STRVLAN    0x40000000
455 #define E1000_RPLOLR_STRCRC     0x80000000

457 #define E1000_DTXCTL_8023LL     0x0004
458 #define E1000_DTXCTL_VLAN_ADDED 0x0008
459 #define E1000_DTXCTL_OOS_ENABLE 0x0010
460 #define E1000_DTXCTL_MDP_EN     0x0020
461 #define E1000_DTXCTL_SPOOF_INT  0x0040

463 #define ALL_QUEUES              0xFFFF

465 /* RX packet buffer size defines */
466 #define E1000_RXPBS_SIZE_MASK_82576     0x0000007F
467 void e1000_vmdq_set_loopback_pf(struct e1000_hw *hw, bool enable);
468 void e1000_vmdq_set_replication_pf(struct e1000_hw *hw, bool enable);
469 u16 e1000_rxpbs_adjust_82580(u32 data);
470 s32 e1000_set_eee_i350(struct e1000_hw *hw);

472 #ifdef __cplusplus
473 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   29388 Thu Feb  2 21:11:33 2012
new/usr/src/uts/common/io/igb/igb_api.c
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.129.2.1 v3_3_14_3_BHSW1 */

#include "igb_api.h"

/*
 * e1000_init_mac_params - Initialize MAC function pointers
 * @hw: pointer to the HW structure
 *
 * This function initializes the function pointers for the MAC
 * set of functions.  Called by drivers or by e1000_setup_init_funcs.
 */
s32
e1000_init_mac_params(struct e1000_hw *hw)
{
        s32 ret_val = E1000_SUCCESS;

        if (hw->mac.ops.init_params) {
                ret_val = hw->mac.ops.init_params(hw);
                if (ret_val) {
                        DEBUGOUT("MAC Initialization Error\n");
                        goto out;
                }
        } else {
                DEBUGOUT("mac.init_mac_params was NULL\n");
                ret_val = -E1000_ERR_CONFIG;
        }

out:
        return (ret_val);
}
```

*_____unchanged_portion_omitted_*


```
/*
 * e1000_set_mac_type - Sets MAC type
 * @hw: pointer to the HW structure
 *
 * This function sets the mac type of the adapter based on the
 * device ID stored in the hw structure.
 * MUST BE FIRST FUNCTION CALLED (explicitly or through
 * e1000_setup_init_funcs()).
 */
s32
e1000_set_mac_type(struct e1000_hw *hw)
{
        struct e1000_mac_info *mac = &hw->mac;
        s32 ret_val = E1000_SUCCESS;

        DEBUGFUNC("e1000_set_mac_type");

        switch (hw->device_id) {
        case E1000_DEV_ID_82575EB_COPPER:
        case E1000_DEV_ID_82575EB_FIBER_SERDES:
        case E1000_DEV_ID_82575GB_QUAD_COPPER:
                mac->type = e1000_82575;
                break;
        case E1000_DEV_ID_82576:
        case E1000_DEV_ID_82576_FIBER:
        case E1000_DEV_ID_82576_SERDES:
        case E1000_DEV_ID_82576_QUAD_COPPER:
        case E1000_DEV_ID_82576_QUAD_COPPER_ET2:
        case E1000_DEV_ID_82576_NS:
        case E1000_DEV_ID_82576_NS_SERDES:
        case E1000_DEV_ID_82576_SERDES_QUAD:
                mac->type = e1000_82576;
                break;
        case E1000_DEV_ID_82580_COPPER:
        case E1000_DEV_ID_82580_FIBER:
        case E1000_DEV_ID_82580_SERDES:
        case E1000_DEV_ID_82580_SGMII:
        case E1000_DEV_ID_82580_COPPER_DUAL:
                mac->type = e1000_82580;
                break;
        case E1000_DEV_ID_I350_COPPER:
                mac->type = e1000_i350;
                break;
        default:
                /* Should never have loaded on this device */
                ret_val = -E1000_ERR_MAC_INIT;
                break;
        }

        return (ret_val);
}

/*
 * e1000_setup_init_funcs - Initializes function pointers
 * @hw: pointer to the HW structure
 * @init_device: true will initialize the rest of the function pointers
 *               getting the device ready for use.  false will only set
 *               MAC type and the function pointers for the other init
 *               functions.  Passing false will not generate any hardware
 *               reads or writes.
 *
 * This function must be called by a driver in order to use the rest
 * of the 'shared' code files. Called by drivers only.
 */
s32
```

```
180 e1000_setup_init_funcs(struct e1000_hw *hw, bool init_device)
181 {
182         s32 ret_val;

184         /* Can't do much good without knowing the MAC type. */
185         ret_val = e1000_set_mac_type(hw);
186         if (ret_val) {
187                 DEBUGOUT("ERROR: MAC type could not be set properly.\n");
188                 goto out;
189         }

191         if (!hw->hw_addr) {
192                 DEBUGOUT("ERROR: Registers not mapped\n");
193                 ret_val = -E1000_ERR_CONFIG;
194                 goto out;
195         }

197         /*
198          * Init function pointers to generic implementations. We do this first
199          * allowing a driver module to override it afterward.
200          */
201         e1000_init_mac_ops_generic(hw);
202         e1000_init_phy_ops_generic(hw);
203         e1000_init_nvm_ops_generic(hw);

205         /*
206          * Set up the init function pointers. These are functions within the
207          * adapter family file that sets up function pointers for the rest of
208          * the functions in that family.
209          */
210         switch (hw->mac.type) {
211         case e1000_82575:
212         case e1000_82576:
213         case e1000_82580:
214         case e1000_i350:
215                 e1000_init_function_pointers_82575(hw);
216                 break;
217         default:
218                 DEBUGOUT("Hardware not supported\n");
219                 ret_val = -E1000_ERR_CONFIG;
220                 break;
221         }

223         /*
224          * Initialize the rest of the function pointers. These require some
225          * register reads/writes in some cases.
226          */
227         if (!(ret_val) && init_device) {
228                 ret_val = e1000_init_mac_params(hw);
229                 if (ret_val)
230                         goto out;

232                 ret_val = e1000_init_nvm_params(hw);
233                 if (ret_val)
234                         goto out;

236                 ret_val = e1000_init_phy_params(hw);
237                 if (ret_val)
238                         goto out;

240         }

242 out:
243         return (ret_val);
244 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   71184 Thu Feb  2 21:11:34 2012
new/usr/src/uts/common/io/igb/igb_defines.h
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.120.2.2 v3_3_14_3_BHSW1 */

#ifndef _IGB_DEFINES_H
#define _IGB_DEFINES_H

#ifdef __cplusplus
extern "C" {
#endif

/* Number of Transmit and Receive Descriptors must be a multiple of 8 */
#define REQ_TX_DESCRIPTOR_MULTIPLE      8
#define REQ_RX_DESCRIPTOR_MULTIPLE      8

/* Definitions for power management and wakeup registers */
/* Wake Up Control */
#define E1000_WUC_APME          0x00000001 /* APM Enable */
#define E1000_WUC_PME_EN        0x00000002 /* PME Enable */
#define E1000_WUC_PME_STATUS    0x00000004 /* PME Status */
#define E1000_WUC_APMPME        0x00000008 /* Assert PME on APM Wakeup */
#define E1000_WUC_LSCWE         0x00000010 /* Link Status wake up enable */
#define E1000_WUC_LSCWO         0x00000020 /* Link Status wake up override */
#define E1000_WUC_SPM           0x80000000 /* Enable SPM */
#define E1000_WUC_PHY_WAKE      0x00000100 /* if PHY supports wakeup */

/* Wake Up Filter Control */
#define E1000_WUFC_LNKC 0x00000001 /* Link Status Change Wakeup Enable */
#define E1000_WUFC_MAG  0x00000002 /* Magic Packet Wakeup Enable */
#define E1000_WUFC_EX   0x00000004 /* Directed Exact Wakeup Enable */
#define E1000_WUFC_MC   0x00000008 /* Directed Multicast Wakeup Enable */
#define E1000_WUFC_BC   0x00000010 /* Broadcast Wakeup Enable */
```



```
#define E1000_WUFC_ARP  0x00000020 /* ARP Request Packet Wakeup Enable */
#define E1000_WUFC_IPV4 0x00000040 /* Directed IPv4 Packet Wakeup Enable */
#define E1000_WUFC_IPV6 0x00000080 /* Directed IPv6 Packet Wakeup Enable */
#define E1000_WUFC_IGNORE_TCO   0x00008000 /* Ignore WakeOn TCO packets */
#define E1000_WUFC_FLX0 0x00010000 /* Flexible Filter 0 Enable */
#define E1000_WUFC_FLX1 0x00020000 /* Flexible Filter 1 Enable */
#define E1000_WUFC_FLX2 0x00040000 /* Flexible Filter 2 Enable */
#define E1000_WUFC_FLX3 0x00080000 /* Flexible Filter 3 Enable */
#define E1000_WUFC_FLX4 0x00100000 /* Flexible Filter 4 Enable */
#define E1000_WUFC_FLX5 0x00200000 /* Flexible Filter 5 Enable */
#define E1000_WUFC_ALL_FILTERS  0x000F00FF /* Mask for all wakeup filters */
#define E1000_WUFC_FLX_OFFSET   16 /* Offset to the Flexible Filters bits */
#define E1000_WUFC_FLX_FILTERS  0x000F0000 /* Mask for the 4 flexible filters */
/*
 * For 82576 to utilize Extended filter masks in addition to
 * existing (filter) masks
 */
#define E1000_WUFC_EXT_FLX_FILTERS      0x00300000 /* Ext. FLX filter mask */

/* Wake Up Status */
#define E1000_WUS_LNKC          E1000_WUFC_LNKC
#define E1000_WUS_MAG           E1000_WUFC_MAG
#define E1000_WUS_EX            E1000_WUFC_EX
#define E1000_WUS_MC            E1000_WUFC_MC
#define E1000_WUS_BC            E1000_WUFC_BC
#define E1000_WUS_ARP           E1000_WUFC_ARP
#define E1000_WUS_IPV4          E1000_WUFC_IPV4
#define E1000_WUS_IPV6          E1000_WUFC_IPV6
#define E1000_WUS_FLX0          E1000_WUFC_FLX0
#define E1000_WUS_FLX1          E1000_WUFC_FLX1
#define E1000_WUS_FLX2          E1000_WUFC_FLX2
#define E1000_WUS_FLX3          E1000_WUFC_FLX3
#define E1000_WUS_FLX_FILTERS   E1000_WUFC_FLX_FILTERS

/* Wake Up Packet Length */
#define E1000_WUPL_LENGTH_MASK  0x0FFF   /* Only the lower 12 bits are valid */

/* Four Flexible Filters are supported */
#define E1000_FLEXIBLE_FILTER_COUNT_MAX 4
/* Two Extended Flexible Filters are supported (82576) */
#define E1000_EXT_FLEXIBLE_FILTER_COUNT_MAX     2
#define E1000_FHFT_LENGTH_OFFSET        0xFC /* Length byte in FHFT */
#define E1000_FHFT_LENGTH_MASK          0x0FF /* Length in lower byte */

/* Each Flexible Filter is at most 128 (0x80) bytes in length */
#define E1000_FLEXIBLE_FILTER_SIZE_MAX  128

#define E1000_FFLT_SIZE E1000_FLEXIBLE_FILTER_COUNT_MAX
#define E1000_FFMT_SIZE E1000_FLEXIBLE_FILTER_SIZE_MAX
#define E1000_FFVT_SIZE E1000_FLEXIBLE_FILTER_SIZE_MAX

/* Extended Device Control */
#define E1000_CTRL_EXT_GPI0_EN  0x00000001 /* Maps SDP4 to GPI0 */
#define E1000_CTRL_EXT_GPI1_EN  0x00000002 /* Maps SDP5 to GPI1 */
#define E1000_CTRL_EXT_PHYINT_EN E1000_CTRL_EXT_GPI1_EN
#define E1000_CTRL_EXT_GPI2_EN  0x00000004 /* Maps SDP6 to GPI2 */
#define E1000_CTRL_EXT_GPI3_EN  0x00000008 /* Maps SDP7 to GPI3 */
/* Reserved (bits 4,5) in >= 82575 */
#define E1000_CTRL_EXT_SDP4_DATA 0x00000010 /* Value of SW Definable Pin 4 */
#define E1000_CTRL_EXT_SDP5_DATA 0x00000020 /* Value of SW Definable Pin 5 */
#define E1000_CTRL_EXT_PHY_INT   E1000_CTRL_EXT_SDP5_DATA
#define E1000_CTRL_EXT_SDP6_DATA 0x00000040 /* Value of SW Definable Pin 6 */
#define E1000_CTRL_EXT_SDP3_DATA 0x00000080 /* Value of SW Definable Pin 3 */
/* SDP 4/5 (bits 8,9) are reserved in >= 82575 */
#define E1000_CTRL_EXT_SDP4_DIR 0x00000100 /* Direction of SDP4 0=in 1=out */
#define E1000_CTRL_EXT_SDP5_DIR 0x00000200 /* Direction of SDP5 0=in 1=out */
```

```
#define E1000_CTRL_EXT_SDP6_DIR 0x00000400 /* Direction of SDP6 0=in 1=out */
#define E1000_CTRL_EXT_SDP3_DIR 0x00000800 /* Direction of SDP3 0=in 1=out */
#define E1000_CTRL_EXT_ASDCHK   0x00001000 /* Initiate an ASD sequence */
#define E1000_CTRL_EXT_EE_RST   0x00002000 /* Reinitialize from EEPROM */
#define E1000_CTRL_EXT_IPS      0x00004000 /* Invert Power State */
/* Physical Func Reset Done Indication */
#define E1000_CTRL_EXT_PFRSTD   0x00004000
#define E1000_CTRL_EXT_SPD_BYPS 0x00008000 /* Speed Select Bypass */
#define E1000_CTRL_EXT_RO_DIS   0x00020000 /* Relaxed Ordering disable */
/* DMA Dynamic Clock Gating */
#define E1000_CTRL_EXT_DMA_DYN_CLK_EN   0x00080000
#define E1000_CTRL_EXT_LINK_MODE_MASK   0x00C00000
#define E1000_CTRL_EXT_LINK_MODE_82580_MASK     0x01C00000 /* 82580 bit 24:22 */
#define E1000_CTRL_EXT_LINK_MODE_1000BASE_KX    0x00400000
#define E1000_CTRL_EXT_LINK_MODE_GMII   0x00000000
#define E1000_CTRL_EXT_LINK_MODE_TBI    0x00C00000
#define E1000_CTRL_EXT_LINK_MODE_KMRN   0x00000000
#define E1000_CTRL_EXT_LINK_MODE_PCIE_SERDES    0x00C00000
#define E1000_CTRL_EXT_LINK_MODE_PCIX_SERDES    0x00800000
#define E1000_CTRL_EXT_LINK_MODE_SGMII  0x00800000
#define E1000_CTRL_EXT_EIAME            0x01000000
#define E1000_CTRL_EXT_IRCA             0x00000001
#define E1000_CTRL_EXT_WR_WMARK_MASK    0x03000000
#define E1000_CTRL_EXT_WR_WMARK_256     0x00000000
#define E1000_CTRL_EXT_WR_WMARK_320     0x01000000
#define E1000_CTRL_EXT_WR_WMARK_384     0x02000000
#define E1000_CTRL_EXT_WR_WMARK_448     0x03000000
#define E1000_CTRL_EXT_CANC     0x04000000 /* Int delay cancellation */
#define E1000_CTRL_EXT_DRV_LOAD 0x10000000 /* Driver loaded bit for FW */
/* IAME enable bit (27) was removed in >= 82575 */
/* Interrupt acknowledge Auto-mask */
#define E1000_CTRL_EXT_IAME             0x08000000
/* Clear Interrupt timers after IMS clear */
#define E1000_CTRL_EXT_INT_TIMER_CLR    0x20000000
/* packet buffer parity error detection enabled */
#define E1000_CRTL_EXT_PB_PAREN         0x01000000
/* descriptor FIFO parity error detection enable */
#define E1000_CTRL_EXT_DF_PAREN         0x02000000
#define E1000_CTRL_EXT_GHOST_PAREN      0x40000000
#define E1000_CTRL_EXT_PBA_CLR          0x80000000 /* PBA Clear */
#define E1000_I2CCMD_REG_ADDR_SHIFT     16
#define E1000_I2CCMD_REG_ADDR           0x00FF0000
#define E1000_I2CCMD_PHY_ADDR_SHIFT     24
#define E1000_I2CCMD_PHY_ADDR           0x07000000
#define E1000_I2CCMD_OPCODE_READ        0x08000000
#define E1000_I2CCMD_OPCODE_WRITE       0x00000000
#define E1000_I2CCMD_RESET              0x10000000
#define E1000_I2CCMD_READY              0x20000000
#define E1000_I2CCMD_INTERRUPT_ENA      0x40000000
#define E1000_I2CCMD_ERROR              0x80000000
#define E1000_MAX_SGMII_PHY_REG_ADDR    255
#define E1000_I2CCMD_PHY_TIMEOUT        200
#define E1000_IVAR_VALID        0x80
#define E1000_GPIE_NSICR        0x00000001
#define E1000_GPIE_MSIX_MODE    0x00000010
#define E1000_GPIE_EIAME        0x40000000
#define E1000_GPIE_PBA          0x80000000

/* Receive Descriptor bit definitions */
#define E1000_RXD_STAT_DD       0x01    /* Descriptor Done */
#define E1000_RXD_STAT_EOP      0x02    /* End of Packet */
#define E1000_RXD_STAT_IXSM     0x04    /* Ignore checksum */
#define E1000_RXD_STAT_VP       0x08    /* IEEE VLAN Packet */
#define E1000_RXD_STAT_UDPCS    0x10    /* UDP xsum calculated */
#define E1000_RXD_STAT_TCPCS    0x20    /* TCP xsum calculated */
#define E1000_RXD_STAT_IPCS     0x40    /* IP xsum calculated */
```



```
#define E1000_RXD_STAT_PIF      0x80    /* passed in-exact filter */
#define E1000_RXD_STAT_CRCV     0x100   /* Speculative CRC Valid */
#define E1000_RXD_STAT_IPIDV    0x200   /* IP identification valid */
#define E1000_RXD_STAT_UDPV     0x400   /* Valid UDP checksum */
#define E1000_RXD_STAT_DYNINT   0x800   /* Pkt caused INT via DYNINT */
#define E1000_RXD_STAT_ACK      0x8000  /* ACK Packet indication */
#define E1000_RXD_ERR_CE        0x01    /* CRC Error */
#define E1000_RXD_ERR_SE        0x02    /* Symbol Error */
#define E1000_RXD_ERR_SEQ       0x04    /* Sequence Error */
#define E1000_RXD_ERR_CXE       0x10    /* Carrier Extension Error */
#define E1000_RXD_ERR_TCPE      0x20    /* TCP/UDP Checksum Error */
#define E1000_RXD_ERR_IPE       0x40    /* IP Checksum Error */
#define E1000_RXD_ERR_RXE       0x80    /* Rx Data Error */
#define E1000_RXD_SPC_VLAN_MASK 0x0FFF  /* VLAN ID is in lower 12 bits */
#define E1000_RXD_SPC_PRI_MASK  0xE000  /* Priority is in upper 3 bits */
#define E1000_RXD_SPC_PRI_SHIFT 13
#define E1000_RXD_SPC_CFI_MASK  0x1000  /* CFI is bit 12 */
#define E1000_RXD_SPC_CFI_SHIFT 12

#define E1000_RXDEXT_STATERR_CE         0x01000000
#define E1000_RXDEXT_STATERR_SE         0x02000000
#define E1000_RXDEXT_STATERR_SEQ        0x04000000
#define E1000_RXDEXT_STATERR_CXE        0x10000000
#define E1000_RXDEXT_STATERR_TCPE       0x20000000
#define E1000_RXDEXT_STATERR_IPE        0x40000000
#define E1000_RXDEXT_STATERR_RXE        0x80000000

/* mask to determine if packets should be dropped due to frame errors */
#define E1000_RXD_ERR_FRAME_ERR_MASK ( \
    E1000_RXD_ERR_CE  | \
    E1000_RXD_ERR_SE  | \
    E1000_RXD_ERR_SEQ | \
    E1000_RXD_ERR_CXE | \
    E1000_RXD_ERR_RXE)

/* Same mask, but for extended and packet split descriptors */
#define E1000_RXDEXT_ERR_FRAME_ERR_MASK ( \
    E1000_RXDEXT_STATERR_CE  |  \
    E1000_RXDEXT_STATERR_SE  |  \
    E1000_RXDEXT_STATERR_SEQ |  \
    E1000_RXDEXT_STATERR_CXE |  \
    E1000_RXDEXT_STATERR_RXE)

#define E1000_MRQC_ENABLE_MASK          0x00000007
#define E1000_MRQC_ENABLE_RSS_2Q        0x00000001
#define E1000_MRQC_ENABLE_RSS_INT       0x00000004
#define E1000_MRQC_RSS_FIELD_MASK       0xFFFF0000
#define E1000_MRQC_RSS_FIELD_IPV4_TCP   0x00010000
#define E1000_MRQC_RSS_FIELD_IPV4       0x00020000
#define E1000_MRQC_RSS_FIELD_IPV6_TCP_EX 0x00040000
#define E1000_MRQC_RSS_FIELD_IPV6_EX    0x00080000
#define E1000_MRQC_RSS_FIELD_IPV6       0x00100000
#define E1000_MRQC_RSS_FIELD_IPV6_TCP   0x00200000

#define E1000_RXDPS_HDRSTAT_HDRSP       0x00008000
#define E1000_RXDPS_HDRSTAT_HDRLEN_MASK 0x000003FF

/* Management Control */
#define E1000_MANC_SMBUS_EN     0x00000001 /* SMBus Enabled - RO */
#define E1000_MANC_ASF_EN       0x00000002 /* ASF Enabled - RO */
#define E1000_MANC_R_ON_FORCE   0x00000004 /* Reset on Force TCO - RO */
#define E1000_MANC_RMCP_EN      0x00000100 /* Enable RCMP 026Fh Filtering */
#define E1000_MANC_0298_EN      0x00000200 /* Enable RCMP 0298h Filtering */
#define E1000_MANC_IPV4_EN      0x00000400 /* Enable IPv4 */
#define E1000_MANC_IPV6_EN      0x00000800 /* Enable IPv6 */
#define E1000_MANC_SNAP_EN      0x00001000 /* Accept LLC/SNAP */
```

```
#define E1000_MANC_ARP_EN       0x00002000 /* Enable ARP Request Filtering */
/* Enable Neighbor Discovery Filtering */
#define E1000_MANC_NEIGHBOR_EN  0x00004000
#define E1000_MANC_ARP_RES_EN   0x00008000 /* Enable ARP response Filtering */
#define E1000_MANC_TCO_RESET    0x00010000 /* TCO Reset Occurred */
#define E1000_MANC_RCV_TCO_EN   0x00020000 /* Receive TCO Packets Enabled */
#define E1000_MANC_REPORT_STATUS 0x00040000 /* Status Reporting Enabled */
#define E1000_MANC_RCV_ALL      0x00080000 /* Receive All Enabled */
#define E1000_MANC_BLK_PHY_RST_ON_IDE   0x00040000 /* Block phy resets */
/* Enable MAC address filtering */
#define E1000_MANC_EN_MAC_ADDR_FILTER   0x00100000
/* Enable MNG packets to host memory */
#define E1000_MANC_EN_MNG2HOST          0x00200000
/* Enable IP address filtering */
#define E1000_MANC_EN_IP_ADDR_FILTER    0x00400000
#define E1000_MANC_EN_XSUM_FILTER 0x00800000 /* Enable checksum filtering */
#define E1000_MANC_BR_EN        0x01000000 /* Enable broadcast filtering */
#define E1000_MANC_SMB_REQ      0x01000000 /* SMBus Request */
#define E1000_MANC_SMB_GNT      0x02000000 /* SMBus Grant */
#define E1000_MANC_SMB_CLK_IN   0x04000000 /* SMBus Clock In */
#define E1000_MANC_SMB_DATA_IN  0x08000000 /* SMBus Data In */
#define E1000_MANC_SMB_DATA_OUT 0x10000000 /* SMBus Data Out */
#define E1000_MANC_SMB_CLK_OUT  0x20000000 /* SMBus Clock Out */

#define E1000_MANC_SMB_DATA_OUT_SHIFT  28 /* SMBus Data Out Shift */
#define E1000_MANC_SMB_CLK_OUT_SHIFT   29 /* SMBus Clock Out Shift */

/* Receive Control */
#define E1000_RCTL_RST          0x00000001 /* Software reset */
#define E1000_RCTL_EN           0x00000002 /* enable */
#define E1000_RCTL_SBP          0x00000004 /* store bad packet */
#define E1000_RCTL_UPE          0x00000008 /* unicast promiscuous enable */
#define E1000_RCTL_MPE          0x00000010 /* multicast promiscuous enab */
#define E1000_RCTL_LPE          0x00000020 /* long packet enable */
#define E1000_RCTL_LBM_NO       0x00000000 /* no loopback mode */
#define E1000_RCTL_LBM_MAC      0x00000040 /* MAC loopback mode */
#define E1000_RCTL_LBM_SLP      0x00000080 /* serial link loopback mode */
#define E1000_RCTL_LBM_TCVR     0x000000C0 /* tcvr loopback mode */
#define E1000_RCTL_DTYP_MASK    0x00000C00 /* Descriptor type mask */
#define E1000_RCTL_DTYP_PS      0x00000400 /* Packet Split descriptor */
#define E1000_RCTL_RDMTS_HALF   0x00000000 /* rx desc min threshold size */
#define E1000_RCTL_RDMTS_QUAT   0x00000100 /* rx desc min threshold size */
#define E1000_RCTL_RDMTS_EIGTH  0x00000200 /* rx desc min threshold size */
#define E1000_RCTL_MO_SHIFT     12         /* multicast offset shift */
#define E1000_RCTL_MO_0         0x00000000 /* multicast offset 11:0 */
#define E1000_RCTL_MO_1         0x00001000 /* multicast offset 12:1 */
#define E1000_RCTL_MO_2         0x00002000 /* multicast offset 13:2 */
#define E1000_RCTL_MO_3         0x00003000 /* multicast offset 15:4 */
#define E1000_RCTL_MDR          0x00004000 /* multicast desc ring 0 */
#define E1000_RCTL_BAM          0x00008000 /* broadcast enable */
/* these buffer sizes are valid if E1000_RCTL_BSEX is 0 */
#define E1000_RCTL_SZ_2048      0x00000000 /* rx buffer size 2048 */
#define E1000_RCTL_SZ_1024      0x00010000 /* rx buffer size 1024 */
#define E1000_RCTL_SZ_512       0x00020000 /* rx buffer size 512 */
#define E1000_RCTL_SZ_256       0x00030000 /* rx buffer size 256 */
/* these buffer sizes are valid if E1000_RCTL_BSEX is 1 */
#define E1000_RCTL_SZ_16384     0x00010000 /* rx buffer size 16384 */
#define E1000_RCTL_SZ_8192      0x00020000 /* rx buffer size 8192 */
#define E1000_RCTL_SZ_4096      0x00030000 /* rx buffer size 4096 */
#define E1000_RCTL_VFE          0x00040000 /* vlan filter enable */
#define E1000_RCTL_CFIEN        0x00080000 /* canonical form enable */
#define E1000_RCTL_CFI          0x00100000 /* canonical form indicator */
#define E1000_RCTL_DPF          0x00400000 /* discard pause frames */
#define E1000_RCTL_PMCF         0x00800000 /* pass MAC control frames */
#define E1000_RCTL_BSEX         0x02000000 /* Buffer size extension */
#define E1000_RCTL_SECRC        0x04000000 /* Strip Ethernet CRC */
```



```
#define E1000_RCTL_FLXBUF_MASK  0x78000000 /* Flexible buffer size */
#define E1000_RCTL_FLXBUF_SHIFT 27         /* Flexible buffer shift */

/*
 * Use byte values for the following shift parameters
 * Usage:
 *    psrctl |= (((ROUNDUP(value0, 128) >> E1000_PSRCTL_BSIZE0_SHIFT) &
 *               E1000_PSRCTL_BSIZE0_MASK) |
 *               ((ROUNDUP(value1, 1024) >> E1000_PSRCTL_BSIZE1_SHIFT) &
 *               E1000_PSRCTL_BSIZE1_MASK) |
 *               ((ROUNDUP(value2, 1024) << E1000_PSRCTL_BSIZE2_SHIFT) &
 *               E1000_PSRCTL_BSIZE2_MASK) |
 *               ((ROUNDUP(value3, 1024) << E1000_PSRCTL_BSIZE3_SHIFT) |;
 *               E1000_PSRCTL_BSIZE3_MASK))
 * where value0 = [128..16256],  default=256
 *       value1 = [1024..64512], default=4096
 *       value2 = [0..64512],    default=4096
 *       value3 = [0..64512],    default=0
 */

#define E1000_PSRCTL_BSIZE0_MASK        0x0000007F
#define E1000_PSRCTL_BSIZE1_MASK        0x00003F00
#define E1000_PSRCTL_BSIZE2_MASK        0x003F0000
#define E1000_PSRCTL_BSIZE3_MASK        0x3F000000

#define E1000_PSRCTL_BSIZE0_SHIFT       7       /* Shift _right_ 7 */
#define E1000_PSRCTL_BSIZE1_SHIFT       2       /* Shift _right_ 2 */
#define E1000_PSRCTL_BSIZE2_SHIFT       6       /* Shift _left_ 6 */
#define E1000_PSRCTL_BSIZE3_SHIFT       14      /* Shift _left_ 14 */

/* SWFW_SYNC Definitions */
#define E1000_SWFW_EEP_SM       0x1
#define E1000_SWFW_PHY0_SM      0x2
#define E1000_SWFW_PHY1_SM      0x4
#define E1000_SWFW_CSR_SM       0x8
#define E1000_SWFW_PHY2_SM      0x20
#define E1000_SWFW_PHY3_SM      0x40

/* FACTPS Definitions */
#define E1000_FACTPS_LFS        0x40000000  /* LAN Function Select */
/* Device Control */
#define E1000_CTRL_FD           0x00000001  /* Full duplex.0=half; 1=full */
#define E1000_CTRL_BEM          0x00000002  /* Endian Mode.0=little,1=big */
#define E1000_CTRL_PRIOR        0x00000004  /* Priority on PCI. 0=rx,1=fair */
#define E1000_CTRL_GIO_MASTER_DISABLE 0x00000004 /* Block new Master requests */
#define E1000_CTRL_LRST         0x00000008  /* Link reset. 0=normal,1=reset */
#define E1000_CTRL_TME          0x00000010  /* Test mode. 0=normal,1=test */
#define E1000_CTRL_SLE          0x00000020  /* Serial Link on 0=dis,1=en */
#define E1000_CTRL_ASDE         0x00000020  /* Auto-speed detect enable */
#define E1000_CTRL_SLU          0x00000040  /* Set link up (Force Link) */
#define E1000_CTRL_ILOS         0x00000080  /* Invert Loss-Of Signal */
#define E1000_CTRL_SPD_SEL      0x00000300  /* Speed Select Mask */
#define E1000_CTRL_SPD_10       0x00000000  /* Force 10Mb */
#define E1000_CTRL_SPD_100      0x00000100  /* Force 100Mb */
#define E1000_CTRL_SPD_1000     0x00000200  /* Force 1Gb */
#define E1000_CTRL_BEM32        0x00000400  /* Big Endian 32 mode */
#define E1000_CTRL_FRCSPD       0x00000800  /* Force Speed */
#define E1000_CTRL_FRCDPX       0x00001000  /* Force Duplex */
#define E1000_CTRL_D_UD_EN      0x00002000  /* Dock/Undock enable */
/* Defined polarity of Dock/Undock indication in SDP[0] */
#define E1000_CTRL_D_UD_POLARITY        0x00004000
/* Reset both PHY ports, through PHYRST_N pin */
#define E1000_CTRL_FORCE_PHY_RESET      0x00008000
/* enable link status from external LINK_0 and LINK_1 pins */
#define E1000_CTRL_EXT_LINK_EN          0x00010000
#define E1000_CTRL_SWDPIN0      0x00040000  /* SWDPIN 0 value */
```

```
390 #define E1000_CTRL_SWDPIN1      0x00080000  /* SWDPIN 1 value */
391 #define E1000_CTRL_SWDPIN2      0x00100000  /* SWDPIN 2 value */
392 #define E1000_CTRL_ADVD3WUC     0x00100000  /* D3 WUC */
393 #define E1000_CTRL_SWDPIN3      0x00200000  /* SWDPIN 3 value */
394 #define E1000_CTRL_SWDPIO0      0x00400000  /* SWDPIN 0 Input or output */
395 #define E1000_CTRL_SWDPIO1      0x00800000  /* SWDPIN 1 input or output */
396 #define E1000_CTRL_SWDPIO2      0x01000000  /* SWDPIN 2 input or output */
397 #define E1000_CTRL_SWDPIO3      0x02000000  /* SWDPIN 3 input or output */
398 #define E1000_CTRL_RST          0x04000000  /* Global reset */
399 #define E1000_CTRL_RFCE         0x08000000  /* Receive Flow Control enable */
400 #define E1000_CTRL_TFCE         0x10000000  /* Transmit flow control enable */
401 #define E1000_CTRL_RTE          0x20000000  /* Routing tag enable */
402 #define E1000_CTRL_VME          0x40000000  /* IEEE VLAN mode enable */
403 #define E1000_CTRL_PHY_RST      0x80000000  /* PHY Reset */
404 #define E1000_CTRL_SW2FW_INT    0x02000000  /* Initiate an interrupt to ME */
405 #define E1000_CTRL_I2C_ENA      0x02000000  /* I2C enable */

407 /*
408  * Bit definitions for the Management Data IO (MDIO) and Management Data
409  * Clock (MDC) pins in the Device Control Register.
410  */
411 #define E1000_CTRL_PHY_RESET_DIR        E1000_CTRL_SWDPIO0
412 #define E1000_CTRL_PHY_RESET            E1000_CTRL_SWDPIN0
413 #define E1000_CTRL_MDIO_DIR             E1000_CTRL_SWDPIO2
414 #define E1000_CTRL_MDIO                 E1000_CTRL_SWDPIN2
415 #define E1000_CTRL_MDC_DIR              E1000_CTRL_SWDPIO3
416 #define E1000_CTRL_MDC                  E1000_CTRL_SWDPIN3
417 #define E1000_CTRL_PHY_RESET_DIR4       E1000_CTRL_EXT_SDP4_DIR
418 #define E1000_CTRL_PHY_RESET4           E1000_CTRL_EXT_SDP4_DATA

420 #define E1000_CONNSW_ENRGSRC            0x4
421 #define E1000_PCS_CFG_PCS_EN            8
422 #define E1000_PCS_LCTL_FLV_LINK_UP      1
423 #define E1000_PCS_LCTL_FSV_10           0
424 #define E1000_PCS_LCTL_FSV_100          2
425 #define E1000_PCS_LCTL_FSV_1000         4
426 #define E1000_PCS_LCTL_FDV_FULL         8
427 #define E1000_PCS_LCTL_FSD              0x10
428 #define E1000_PCS_LCTL_FORCE_LINK       0x20
429 #define E1000_PCS_LCTL_LOW_LINK_LATCH   0x40
430 #define E1000_PCS_LCTL_FORCE_FCTRL      0x80
431 #define E1000_PCS_LCTL_AN_ENABLE        0x10000
432 #define E1000_PCS_LCTL_AN_RESTART       0x20000
433 #define E1000_PCS_LCTL_AN_TIMEOUT       0x40000
434 #define E1000_PCS_LCTL_AN_SGMII_BYPASS  0x80000
435 #define E1000_PCS_LCTL_AN_SGMII_TRIGGER 0x100000
436 #define E1000_PCS_LCTL_FAST_LINK_TIMER  0x1000000
437 #define E1000_PCS_LCTL_LINK_OK_FIX      0x2000000
438 #define E1000_PCS_LCTL_CRS_ON_NI        0x4000000
439 #define E1000_ENABLE_SERDES_LOOPBACK    0x0410

441 #define E1000_PCS_LSTS_LINK_OK          1
442 #define E1000_PCS_LSTS_SPEED_10         0
443 #define E1000_PCS_LSTS_SPEED_100        2
444 #define E1000_PCS_LSTS_SPEED_1000       4
445 #define E1000_PCS_LSTS_DUPLEX_FULL      8
446 #define E1000_PCS_LSTS_SYNK_OK          0x10
447 #define E1000_PCS_LSTS_AN_COMPLETE      0x10000
448 #define E1000_PCS_LSTS_AN_PAGE_RX       0x20000
449 #define E1000_PCS_LSTS_AN_TIMED_OUT     0x40000
450 #define E1000_PCS_LSTS_AN_REMOTE_FAULT  0x80000
451 #define E1000_PCS_LSTS_AN_ERROR_RWS     0x100000

453 /* Device Status */
454 #define E1000_STATUS_FD         0x00000001 /* Full duplex.0=half,1=full */
455 #define E1000_STATUS_LU         0x00000002 /* Link up.0=no,1=link */
```



```
456 #define E1000_STATUS_FUNC_MASK  0x0000000C /* PCI Function Mask */
457 #define E1000_STATUS_FUNC_SHIFT 2
458 #define E1000_STATUS_FUNC_0     0x00000000 /* Function 0 */
459 #define E1000_STATUS_FUNC_1     0x00000004 /* Function 1 */
460 #define E1000_STATUS_TXOFF      0x00000010 /* transmission paused */
461 #define E1000_STATUS_TBIMODE    0x00000020 /* TBI mode */
462 #define E1000_STATUS_SPEED_MASK 0x000000C0
463 #define E1000_STATUS_SPEED_10   0x00000000 /* Speed 10Mb/s */
464 #define E1000_STATUS_SPEED_100  0x00000040 /* Speed 100Mb/s */
465 #define E1000_STATUS_SPEED_1000 0x00000080 /* Speed 1000Mb/s */
466 #define E1000_STATUS_LAN_INIT_DONE 0x00000200 /* Lan Init Completion by NVM */
467 #define E1000_STATUS_ASDV       0x00000300 /* Auto speed detect value */
468 /* Change in Dock/Undock state. Clear on write '0'. */
469 #define E1000_STATUS_PHYRA      0x00000400 /* PHY Reset Asserted */
470 #define E1000_STATUS_DOCK_CI    0x00000800
471 #define E1000_STATUS_GIO_MASTER_ENABLE  0x00080000 /* Master request status */
472 #define E1000_STATUS_MTXCKOK    0x00000400 /* MTX clock running OK */
473 #define E1000_STATUS_PCI66      0x00000800 /* In 66Mhz slot */
474 #define E1000_STATUS_BUS64      0x00001000 /* In 64 bit slot */
475 #define E1000_STATUS_PCIX_MODE  0x00002000 /* PCI-X mode */
476 #define E1000_STATUS_PCIX_SPEED 0x0000C000 /* PCI-X bus speed */
477 #define E1000_STATUS_BMC_SKU_0  0x00100000 /* BMC USB redirect disabled */
478 #define E1000_STATUS_BMC_SKU_1  0x00200000 /* BMC SRAM disabled */
479 #define E1000_STATUS_BMC_SKU_2  0x00400000 /* BMC SDRAM disabled */
480 #define E1000_STATUS_BMC_CRYPTO 0x00800000 /* BMC crypto disabled */
481 /* BMC external code execution disabled */
482 #define E1000_STATUS_BMC_LITE   0x01000000
483 #define E1000_STATUS_RGMII_ENABLE 0x02000000 /* RGMII disabled */
484 #define E1000_STATUS_FUSE_8     0x04000000
485 #define E1000_STATUS_FUSE_9     0x08000000
486 #define E1000_STATUS_SERDES0_DIS 0x10000000 /* SERDES disabled on port 0 */
487 #define E1000_STATUS_SERDES1_DIS 0x20000000 /* SERDES disabled on port 1 */

489 /* Constants used to interpret the masked PCI-X bus speed. */
490 #define E1000_STATUS_PCIX_SPEED_66  0x00000000 /* PCI-X bus speed  50-66 MHz */
491 #define E1000_STATUS_PCIX_SPEED_100 0x00004000 /* PCI-X bus speed  66-100 MHz */
492 #define E1000_STATUS_PCIX_SPEED_133 0x00008000 /* PCI-X bus speed 100-133 MHz */

494 #define SPEED_10        10
495 #define SPEED_100       100
496 #define SPEED_1000      1000
497 #define HALF_DUPLEX     1
498 #define FULL_DUPLEX     2

500 #define PHY_FORCE_TIME  20

502 #define ADVERTISE_10_HALF       0x0001
503 #define ADVERTISE_10_FULL       0x0002
504 #define ADVERTISE_100_HALF      0x0004
505 #define ADVERTISE_100_FULL      0x0008
506 #define ADVERTISE_1000_HALF     0x0010 /* Not used, just FYI */
507 #define ADVERTISE_1000_FULL     0x0020

509 /* 1000/H is not supported, nor spec-compliant. */
510 #define E1000_ALL_SPEED_DUPLEX  (ADVERTISE_10_HALF | ADVERTISE_10_FULL | \
511                                  ADVERTISE_100_HALF | ADVERTISE_100_FULL | \
512                                  ADVERTISE_1000_FULL)
513 #define E1000_ALL_NOT_GIG       (ADVERTISE_10_HALF | ADVERTISE_10_FULL | \
514                                  ADVERTISE_100_HALF | ADVERTISE_100_FULL)
515 #define E1000_ALL_100_SPEED     (ADVERTISE_100_HALF | ADVERTISE_100_FULL)
516 #define E1000_ALL_10_SPEED      (ADVERTISE_10_HALF | ADVERTISE_10_FULL)
517 #define E1000_ALL_FULL_DUPLEX   (ADVERTISE_10_FULL | ADVERTISE_100_FULL | \
518                                  ADVERTISE_1000_FULL)
519 #define E1000_ALL_HALF_DUPLEX   (ADVERTISE_10_HALF | ADVERTISE_100_HALF)

521 #define AUTONEG_ADVERTISE_SPEED_DEFAULT E1000_ALL_SPEED_DUPLEX
```

```
523 /* LED Control */
524 #define E1000_LEDCTL_LED0_MODE_MASK     0x0000000F
525 #define E1000_LEDCTL_LED0_MODE_SHIFT    0
526 #define E1000_LEDCTL_LED0_BLINK_RATE    0x00000020
527 #define E1000_LEDCTL_LED0_IVRT          0x00000040
528 #define E1000_LEDCTL_LED0_BLINK         0x00000080
529 #define E1000_LEDCTL_LED1_MODE_MASK     0x00000F00
530 #define E1000_LEDCTL_LED1_MODE_SHIFT    8
531 #define E1000_LEDCTL_LED1_BLINK_RATE    0x00002000
532 #define E1000_LEDCTL_LED1_IVRT          0x00004000
533 #define E1000_LEDCTL_LED1_BLINK         0x00008000
534 #define E1000_LEDCTL_LED2_MODE_MASK     0x000F0000
535 #define E1000_LEDCTL_LED2_MODE_SHIFT    16
536 #define E1000_LEDCTL_LED2_BLINK_RATE    0x00200000
537 #define E1000_LEDCTL_LED2_IVRT          0x00400000
538 #define E1000_LEDCTL_LED2_BLINK         0x00800000
539 #define E1000_LEDCTL_LED3_MODE_MASK     0x0F000000
540 #define E1000_LEDCTL_LED3_MODE_SHIFT    24
541 #define E1000_LEDCTL_LED3_BLINK_RATE    0x20000000
542 #define E1000_LEDCTL_LED3_IVRT          0x40000000
543 #define E1000_LEDCTL_LED3_BLINK         0x80000000

545 #define E1000_LEDCTL_MODE_LINK_10_1000  0x0
546 #define E1000_LEDCTL_MODE_LINK_100_1000 0x1
547 #define E1000_LEDCTL_MODE_LINK_UP       0x2
548 #define E1000_LEDCTL_MODE_ACTIVITY      0x3
549 #define E1000_LEDCTL_MODE_LINK_ACTIVITY 0x4
550 #define E1000_LEDCTL_MODE_LINK_10       0x5
551 #define E1000_LEDCTL_MODE_LINK_100      0x6
552 #define E1000_LEDCTL_MODE_LINK_1000     0x7
553 #define E1000_LEDCTL_MODE_PCIX_MODE     0x8
554 #define E1000_LEDCTL_MODE_FULL_DUPLEX   0x9
555 #define E1000_LEDCTL_MODE_COLLISION     0xA
556 #define E1000_LEDCTL_MODE_BUS_SPEED     0xB
557 #define E1000_LEDCTL_MODE_BUS_SIZE      0xC
558 #define E1000_LEDCTL_MODE_PAUSED        0xD
559 #define E1000_LEDCTL_MODE_LED_ON        0xE
560 #define E1000_LEDCTL_MODE_LED_OFF       0xF

562 /* Transmit Descriptor bit definitions */
563 #define E1000_TXD_DTYP_D        0x00100000 /* Data Descriptor */
564 #define E1000_TXD_DTYP_C        0x00000000 /* Context Descriptor */
565 #define E1000_TXD_POPTS_SHIFT   8          /* POPTS shift */
566 #define E1000_TXD_POPTS_IXSM    0x01       /* Insert IP checksum */
567 #define E1000_TXD_POPTS_TXSM    0x02       /* Insert TCP/UDP checksum */
568 #define E1000_TXD_CMD_EOP       0x01000000 /* End of Packet */
569 #define E1000_TXD_CMD_IFCS      0x02000000 /* Insert FCS (Ethernet CRC) */
570 #define E1000_TXD_CMD_IC        0x04000000 /* Insert Checksum */
571 #define E1000_TXD_CMD_RS        0x08000000 /* Report Status */
572 #define E1000_TXD_CMD_RPS       0x10000000 /* Report Packet Sent */
573 #define E1000_TXD_CMD_DEXT      0x20000000 /* Descriptor extension (0=legacy) */
574 #define E1000_TXD_CMD_VLE       0x40000000 /* Add VLAN tag */
575 #define E1000_TXD_CMD_IDE       0x80000000 /* Enable Tidv register */
576 #define E1000_TXD_STAT_DD       0x00000001 /* Descriptor Done */
577 #define E1000_TXD_STAT_EC       0x00000002 /* Excess Collisions */
578 #define E1000_TXD_STAT_LC       0x00000004 /* Late Collisions */
579 #define E1000_TXD_STAT_TU       0x00000008 /* Transmit underrun */
580 #define E1000_TXD_CMD_TCP       0x01000000 /* TCP packet */
581 #define E1000_TXD_CMD_IP        0x02000000 /* IP packet */
582 #define E1000_TXD_CMD_TSE       0x04000000 /* TCP Seg enable */
583 #define E1000_TXD_STAT_TC       0x00000004 /* Tx Underrun */
584 /* Extended desc bits for Linksec and timesync */

586 /* Transmit Control */
587 #define E1000_TCTL_RST  0x00000001      /* software reset */
```



```
588 #define E1000_TCTL_EN   0x00000002      /* enable tx */
589 #define E1000_TCTL_BCE  0x00000004      /* busy check enable */
590 #define E1000_TCTL_PSP  0x00000008      /* pad short packets */
591 #define E1000_TCTL_CT   0x00000ff0      /* collision threshold */
592 #define E1000_TCTL_COLD 0x003ff000      /* collision distance */
593 #define E1000_TCTL_SWXOFF 0x00400000    /* SW Xoff transmission */
594 #define E1000_TCTL_PBE  0x00800000      /* Packet Burst Enable */
595 #define E1000_TCTL_RTLC 0x01000000      /* Re-transmit on late collision */
596 #define E1000_TCTL_NRTU 0x02000000      /* No Re-transmit on underrun */
597 #define E1000_TCTL_MULR 0x10000000      /* Multiple request support */

599 /* Transmit Arbitration Count */
600 #define E1000_TARC0_ENABLE      0x00000400 /* Enable Tx Queue 0 */

602 /* SerDes Control */
603 #define E1000_SCTL_DISABLE_SERDES_LOOPBACK      0x0400

605 /* Receive Checksum Control */
606 #define E1000_RXCSUM_PCSS_MASK  0x000000FF /* Packet Checksum Start */
607 #define E1000_RXCSUM_IPOFL      0x00000100 /* IPv4 checksum offload */
608 #define E1000_RXCSUM_TUOFL      0x00000200 /* TCP / UDP checksum offload */
609 #define E1000_RXCSUM_IPV6OFL    0x00000400 /* IPv6 checksum offload */
610 #define E1000_RXCSUM_CRCOFL     0x00000800 /* CRC32 offload enable */
611 #define E1000_RXCSUM_IPPCSE     0x00001000 /* IP payload checksum enable */
612 #define E1000_RXCSUM_PCSD       0x00002000 /* packet checksum disabled */

614 /* Header split receive */
615 #define E1000_RFCTL_ISCSI_DIS           0x00000001
616 #define E1000_RFCTL_ISCSI_DWC_MASK      0x0000003E
617 #define E1000_RFCTL_ISCSI_DWC_SHIFT     1
618 #define E1000_RFCTL_NFSW_DIS            0x00000040
619 #define E1000_RFCTL_NFSR_DIS            0x00000080
620 #define E1000_RFCTL_NFS_VER_MASK        0x00000300
621 #define E1000_RFCTL_NFS_VER_SHIFT       8
622 #define E1000_RFCTL_IPV6_DIS            0x00000400
623 #define E1000_RFCTL_IPV6_XSUM_DIS       0x00000800
624 #define E1000_RFCTL_ACK_DIS             0x00001000
625 #define E1000_RFCTL_ACKD_DIS            0x00002000
626 #define E1000_RFCTL_IPFRSP_DIS          0x00004000
627 #define E1000_RFCTL_EXTEN               0x00008000
628 #define E1000_RFCTL_IPV6_EX_DIS         0x00010000
629 #define E1000_RFCTL_NEW_IPV6_EXT_DIS    0x00020000
630 #define E1000_RFCTL_LEF                 0x00040000

632 /* Collision related configuration parameters */
633 #define E1000_COLLISION_THRESHOLD       15
634 #define E1000_CT_SHIFT                  4
635 #define E1000_COLLISION_DISTANCE        63
636 #define E1000_COLD_SHIFT                12

638 /* Default values for the transmit IPG register */
639 #define DEFAULT_82543_TIPG_IPGT_FIBER   9
640 #define DEFAULT_82543_TIPG_IPGT_COPPER  8

642 #define E1000_TIPG_IPGT_MASK    0x000003FF
643 #define E1000_TIPG_IPGR1_MASK   0x000FFC00
644 #define E1000_TIPG_IPGR2_MASK   0x3FF00000

646 #define DEFAULT_82543_TIPG_IPGR1        8
647 #define E1000_TIPG_IPGR1_SHIFT          10

649 #define DEFAULT_82543_TIPG_IPGR2        6
650 #define DEFAULT_80003ES2LAN_TIPG_IPGR2  7
651 #define E1000_TIPG_IPGR2_SHIFT          20

653 /* Ethertype field values */
```

```
654 #define ETHERNET_IEEE_VLAN_TYPE 0x8100  /* 802.3ac packet */

656 #define ETHERNET_FCS_SIZE       4
657 #define MAX_JUMBO_FRAME_SIZE    0x3F00

659 /* Extended Configuration Control and Size */
660 #define E1000_EXTCNF_CTRL_MDIO_SW_OWNERSHIP     0x00000020
661 #define E1000_EXTCNF_CTRL_LCD_WRITE_ENABLE      0x00000001
662 #define E1000_EXTCNF_CTRL_OEM_WRITE_ENABLE      0x00000008
663 #define E1000_EXTCNF_CTRL_SWFLAG                0x00000020
664 #define E1000_EXTCNF_SIZE_EXT_PCIE_LENGTH_MASK  0x00FF0000
665 #define E1000_EXTCNF_SIZE_EXT_PCIE_LENGTH_SHIFT 16
666 #define E1000_EXTCNF_CTRL_EXT_CNF_POINTER_MASK  0x0FFF0000
667 #define E1000_EXTCNF_CTRL_EXT_CNF_POINTER_SHIFT 16

669 #define E1000_PHY_CTRL_SPD_EN                   0x00000001
670 #define E1000_PHY_CTRL_D0A_LPLU                 0x00000002
671 #define E1000_PHY_CTRL_NOND0A_LPLU              0x00000004
672 #define E1000_PHY_CTRL_NOND0A_GBE_DISABLE       0x00000008
673 #define E1000_PHY_CTRL_GBE_DISABLE              0x00000040

675 #define E1000_KABGTXD_BGSQLBIAS                 0x00050000

677 /* PBA constants */
678 #define E1000_PBA_6K    0x0006  /* 6KB */
679 #define E1000_PBA_8K    0x0008  /* 8KB */
680 #define E1000_PBA_10K   0x000A  /* 10KB */
681 #define E1000_PBA_12K   0x000C  /* 12KB */
682 #define E1000_PBA_14K   0x000E  /* 14KB */
683 #define E1000_PBA_16K   0x0010  /* 16KB */
684 #define E1000_PBA_18K   0x0012
685 #define E1000_PBA_20K   0x0014
686 #define E1000_PBA_22K   0x0016
687 #define E1000_PBA_24K   0x0018
688 #define E1000_PBA_26K   0x001A
689 #define E1000_PBA_30K   0x001E
690 #define E1000_PBA_32K   0x0020
691 #define E1000_PBA_34K   0x0022
692 #define E1000_PBA_35K   0x0023
693 #define E1000_PBA_38K   0x0026
694 #define E1000_PBA_40K   0x0028
695 #define E1000_PBA_48K   0x0030    /* 48KB */
696 #define E1000_PBA_64K   0x0040    /* 64KB */

698 #define E1000_PBS_16K   E1000_PBA_16K
699 #define E1000_PBS_24K   E1000_PBA_24K

701 #define IFS_MAX         80
702 #define IFS_MIN         40
703 #define IFS_RATIO       4
704 #define IFS_STEP        10
705 #define MIN_NUM_XMITS   1000

707 /* SW Semaphore Register */
708 #define E1000_SWSM_SMBI         0x00000001 /* Driver Semaphore bit */
709 #define E1000_SWSM_SWESMBI      0x00000002 /* FW Semaphore bit */
710 #define E1000_SWSM_WMNG         0x00000004 /* Wake MNG Clock */
711 #define E1000_SWSM_DRV_LOAD     0x00000008 /* Driver Loaded Bit */

713 /* Secondary driver semaphore bit */
714 #define E1000_SWSM2_LOCK        0x00000002

716 /* Interrupt Cause Read */
717 #define E1000_ICR_TXDW          0x00000001 /* Transmit desc written back */
718 #define E1000_ICR_TXQE          0x00000002 /* Transmit Queue empty */
719 #define E1000_ICR_LSC           0x00000004 /* Link Status Change */
```



```
720 #define E1000_ICR_RXSEQ         0x00000008 /* rx sequence error */
721 #define E1000_ICR_RXDMT0        0x00000010 /* rx desc min. threshold (0) */
722 #define E1000_ICR_RXO           0x00000040 /* rx overrun */
723 #define E1000_ICR_RXT0          0x00000080 /* rx timer intr (ring 0) */
724 #define E1000_ICR_VMMB          0x00000100 /* VM MB event */
725 #define E1000_ICR_MDAC          0x00000200 /* MDIO access complete */
726 #define E1000_ICR_RXCFG         0x00000400 /* Rx /c/ ordered set */
727 #define E1000_ICR_GPI_EN0       0x00000800 /* GP Int 0 */
728 #define E1000_ICR_GPI_EN1       0x00001000 /* GP Int 1 */
729 #define E1000_ICR_GPI_EN2       0x00002000 /* GP Int 2 */
730 #define E1000_ICR_GPI_EN3       0x00004000 /* GP Int 3 */
731 #define E1000_ICR_TXD_LOW       0x00008000
732 #define E1000_ICR_SRPD          0x00010000
733 #define E1000_ICR_ACK           0x00020000 /* Receive Ack frame */
734 #define E1000_ICR_MNG           0x00040000 /* Manageability event */
735 #define E1000_ICR_DOCK          0x00080000 /* Dock/Undock */
736 #define E1000_ICR_DRSTA         0x40000000 /* Device Reset Asserted */
737 /* If this bit asserted, the driver should claim the interrupt */
738 #define E1000_ICR_INT_ASSERTED  0x80000000
739 #define E1000_ICR_RXD_FIFO_PAR0 0x00100000 /* Q0 Rx desc FIFO parity error */
740 #define E1000_ICR_TXD_FIFO_PAR0 0x00200000 /* Q0 Tx desc FIFO parity error */
741 #define E1000_ICR_HOST_ARB_PAR  0x00400000 /* host arb read buffer parity err */
742 #define E1000_ICR_PB_PAR        0x00800000 /* packet buffer parity error */
743 #define E1000_ICR_RXD_FIFO_PAR1 0x01000000 /* Q1 Rx desc FIFO parity error */
744 #define E1000_ICR_TXD_FIFO_PAR1 0x02000000 /* Q1 Tx desc FIFO parity error */
745 #define E1000_ICR_ALL_PARITY    0x03F00000 /* all parity error bits */
746 /* FW changed the status of DISSW bit in the FWSM */
747 #define E1000_ICR_DSW           0x00000020
748 /* LAN connected device generates an interrupt */
749 #define E1000_ICR_PHYINT        0x00001000
750 #define E1000_ICR_DOUTSYNC      0x10000000 /* NIC DMA out of sync */
751 #define E1000_ICR_EPRST         0x00100000 /* ME hardware reset occurs */
752 #define E1000_ICR_FER           0x00400000 /* Fatal Error */

754 /* Extended Interrupt Cause Read */
755 #define E1000_EICR_RX_QUEUE0    0x00000001 /* Rx Queue 0 Interrupt */
756 #define E1000_EICR_RX_QUEUE1    0x00000002 /* Rx Queue 1 Interrupt */
757 #define E1000_EICR_RX_QUEUE2    0x00000004 /* Rx Queue 2 Interrupt */
758 #define E1000_EICR_RX_QUEUE3    0x00000008 /* Rx Queue 3 Interrupt */
759 #define E1000_EICR_TX_QUEUE0    0x00000100 /* Tx Queue 0 Interrupt */
760 #define E1000_EICR_TX_QUEUE1    0x00000200 /* Tx Queue 1 Interrupt */
761 #define E1000_EICR_TX_QUEUE2    0x00000400 /* Tx Queue 2 Interrupt */
762 #define E1000_EICR_TX_QUEUE3    0x00000800 /* Tx Queue 3 Interrupt */
763 #define E1000_EICR_TCP_TIMER    0x40000000 /* TCP Timer */
764 #define E1000_EICR_OTHER        0x80000000 /* Interrupt Cause Active */
765 /* TCP Timer */
766 #define E1000_TCPTIMER_KS       0x00000100 /* KickStart */
767 #define E1000_TCPTIMER_COUNT_ENABLE     0x00000200 /* Count Enable */
768 #define E1000_TCPTIMER_COUNT_FINISH     0x00000400 /* Count finish */
769 #define E1000_TCPTIMER_LOOP     0x00000800 /* Loop */

771 /*
772  * This defines the bits that are set in the Interrupt Mask
773  * Set/Read Register.  Each bit is documented below:
774  *   o RXDMT0 = Receive Descriptor Minimum Threshold hit (ring 0)
775  *   o RXSEQ  = Receive Sequence Error
776  */
777 #define POLL_IMS_ENABLE_MASK ( \
778     E1000_IMS_RXDMT0 |    \
779     E1000_IMS_RXSEQ)

781 /*
782  * This defines the bits that are set in the Interrupt Mask
783  * Set/Read Register.  Each bit is documented below:
784  *   o RXT0   = Receiver Timer Interrupt (ring 0)
785  *   o TXDW   = Transmit Descriptor Written Back
```

```
 786  *   o RXDMT0 = Receive Descriptor Minimum Threshold hit (ring 0)
 787  *   o RXSEQ  = Receive Sequence Error
 788  *   o LSC    = Link Status Change
 789  */
 790 #define IMS_ENABLE_MASK ( \
 791     E1000_IMS_RXT0   |    \
 792     E1000_IMS_TXDW   |    \
 793     E1000_IMS_RXDMT0 |    \
 794     E1000_IMS_RXSEQ  |    \
 795     E1000_IMS_LSC)

 797 /* Interrupt Mask Set */
 798 #define E1000_IMS_TXDW          E1000_ICR_TXDW  /* Transmit desc written back */
 799 #define E1000_IMS_TXQE          E1000_ICR_TXQE  /* Transmit Queue empty */
 800 #define E1000_IMS_LSC           E1000_ICR_LSC   /* Link Status Change */
 801 #define E1000_IMS_VMMB          E1000_ICR_VMMB  /* Mail box activity */
 802 #define E1000_IMS_RXSEQ         E1000_ICR_RXSEQ /* rx sequence error */
 803 #define E1000_IMS_RXDMT0        E1000_ICR_RXDMT0 /* rx desc min. threshold */
 804 #define E1000_IMS_RXO           E1000_ICR_RXO   /* rx overrun */
 805 #define E1000_IMS_RXT0          E1000_ICR_RXT0  /* rx timer intr */
 806 #define E1000_IMS_MDAC          E1000_ICR_MDAC  /* MDIO access complete */
 807 #define E1000_IMS_RXCFG         E1000_ICR_RXCFG /* Rx /c/ ordered set */
 808 #define E1000_IMS_GPI_EN0       E1000_ICR_GPI_EN0 /* GP Int 0 */
 809 #define E1000_IMS_GPI_EN1       E1000_ICR_GPI_EN1 /* GP Int 1 */
 810 #define E1000_IMS_GPI_EN2       E1000_ICR_GPI_EN2 /* GP Int 2 */
 811 #define E1000_IMS_GPI_EN3       E1000_ICR_GPI_EN3 /* GP Int 3 */
 812 #define E1000_IMS_TXD_LOW       E1000_ICR_TXD_LOW
 813 #define E1000_IMS_SRPD          E1000_ICR_SRPD
 814 #define E1000_IMS_ACK           E1000_ICR_ACK   /* Receive Ack frame */
 815 #define E1000_IMS_MNG           E1000_ICR_MNG   /* Manageability event */
 816 #define E1000_IMS_DOCK          E1000_ICR_DOCK  /* Dock/Undock */
 817 #define E1000_IMS_DRSTA         E1000_ICR_DRSTA /* Device Reset Asserted */
 818 /* queue 0 Rx descriptor FIFO parity error */
 819 #define E1000_IMS_RXD_FIFO_PAR0 E1000_ICR_RXD_FIFO_PAR0
 820 /* queue 0 Tx descriptor FIFO parity error */
 821 #define E1000_IMS_TXD_FIFO_PAR0 E1000_ICR_TXD_FIFO_PAR0
 822 /* host arb read buffer parity error */
 823 #define E1000_IMS_HOST_ARB_PAR  E1000_ICR_HOST_ARB_PAR
 824 /* packet buffer parity error */
 825 #define E1000_IMS_PB_PAR        E1000_ICR_PB_PAR
 826 /* queue 1 Rx descriptor FIFO parity error */
 827 #define E1000_IMS_RXD_FIFO_PAR1 E1000_ICR_RXD_FIFO_PAR1
 828 /* queue 1 Tx descriptor FIFO parity error */
 829 #define E1000_IMS_TXD_FIFO_PAR1 E1000_ICR_TXD_FIFO_PAR1
 830 #define E1000_IMS_DSW           E1000_ICR_DSW
 831 #define E1000_IMS_PHYINT        E1000_ICR_PHYINT
 832 #define E1000_IMS_DOUTSYNC      E1000_ICR_DOUTSYNC /* NIC DMA out of sync */
 833 #define E1000_IMS_EPRST         E1000_ICR_EPRST
 834 #define E1000_IMS_FER           E1000_ICR_FER /* Fatal Error */

 836 /* Extended Interrupt Mask Set */
 837 #define E1000_EIMS_RX_QUEUE0    E1000_EICR_RX_QUEUE0 /* Rx Queue 0 Interrupt */
 838 #define E1000_EIMS_RX_QUEUE1    E1000_EICR_RX_QUEUE1 /* Rx Queue 1 Interrupt */
 839 #define E1000_EIMS_RX_QUEUE2    E1000_EICR_RX_QUEUE2 /* Rx Queue 2 Interrupt */
 840 #define E1000_EIMS_RX_QUEUE3    E1000_EICR_RX_QUEUE3 /* Rx Queue 3 Interrupt */
 841 #define E1000_EIMS_TX_QUEUE0    E1000_EICR_TX_QUEUE0 /* Tx Queue 0 Interrupt */
 842 #define E1000_EIMS_TX_QUEUE1    E1000_EICR_TX_QUEUE1 /* Tx Queue 1 Interrupt */
 843 #define E1000_EIMS_TX_QUEUE2    E1000_EICR_TX_QUEUE2 /* Tx Queue 2 Interrupt */
 844 #define E1000_EIMS_TX_QUEUE3    E1000_EICR_TX_QUEUE3 /* Tx Queue 3 Interrupt */
 845 #define E1000_EIMS_TCP_TIMER    E1000_EICR_TCP_TIMER /* TCP Timer */
 846 #define E1000_EIMS_OTHER        E1000_EICR_OTHER   /* Interrupt Cause Active */

 848 /* Interrupt Cause Set */
 849 #define E1000_ICS_TXDW          E1000_ICR_TXDW  /* Transmit desc written back */
 850 #define E1000_ICS_TXQE          E1000_ICR_TXQE  /* Transmit Queue empty */
 851 #define E1000_ICS_LSC           E1000_ICR_LSC   /* Link Status Change */
```



```
 852 #define E1000_ICS_RXSEQ         E1000_ICR_RXSEQ /* rx sequence error */
 853 #define E1000_ICS_RXDMT0        E1000_ICR_RXDMT0 /* rx desc min. threshold */
 854 #define E1000_ICS_RXO           E1000_ICR_RXO   /* rx overrun */
 855 #define E1000_ICS_RXT0          E1000_ICR_RXT0  /* rx timer intr */
 856 #define E1000_ICS_MDAC          E1000_ICR_MDAC  /* MDIO access complete */
 857 #define E1000_ICS_RXCFG         E1000_ICR_RXCFG /* Rx /c/ ordered set */
 858 #define E1000_ICS_GPI_EN0       E1000_ICR_GPI_EN0 /* GP Int 0 */
 859 #define E1000_ICS_GPI_EN1       E1000_ICR_GPI_EN1 /* GP Int 1 */
 860 #define E1000_ICS_GPI_EN2       E1000_ICR_GPI_EN2 /* GP Int 2 */
 861 #define E1000_ICS_GPI_EN3       E1000_ICR_GPI_EN3 /* GP Int 3 */
 862 #define E1000_ICS_TXD_LOW       E1000_ICR_TXD_LOW
 863 #define E1000_ICS_SRPD          E1000_ICR_SRPD
 864 #define E1000_ICS_ACK           E1000_ICR_ACK   /* Receive Ack frame */
 865 #define E1000_ICS_MNG           E1000_ICR_MNG   /* Manageability event */
 866 #define E1000_ICS_DOCK          E1000_ICR_DOCK  /* Dock/Undock */
 867 #define E1000_ICS_DRSTA         E1000_ICR_DRSTA /* Device Reset Aserted */
 868 /* queue 0 Rx descriptor FIFO parity error */
 869 #define E1000_ICS_RXD_FIFO_PAR0 E1000_ICR_RXD_FIFO_PAR0
 870 /* queue 0 Tx descriptor FIFO parity error */
 871 #define E1000_ICS_TXD_FIFO_PAR0 E1000_ICR_TXD_FIFO_PAR0
 872 /* host arb read buffer parity error */
 873 #define E1000_ICS_HOST_ARB_PAR  E1000_ICR_HOST_ARB_PAR
 874 /* packet buffer parity error */
 875 #define E1000_ICS_PB_PAR        E1000_ICR_PB_PAR
 876 /* queue 1 Rx descriptor FIFO parity error */
 877 #define E1000_ICS_RXD_FIFO_PAR1 E1000_ICR_RXD_FIFO_PAR1
 878 /* queue 1 Tx descriptor FIFO parity error */
 879 #define E1000_ICS_TXD_FIFO_PAR1 E1000_ICR_TXD_FIFO_PAR1
 880 #define E1000_ICS_DSW           E1000_ICR_DSW
 881 #define E1000_ICS_DOUTSYNC      E1000_ICR_DOUTSYNC /* NIC DMA out of sync */
 882 #define E1000_ICS_PHYINT        E1000_ICR_PHYINT
 883 #define E1000_ICS_EPRST         E1000_ICR_EPRST

 885 /* Extended Interrupt Cause Set */
 886 #define E1000_EICS_RX_QUEUE0    E1000_EICR_RX_QUEUE0 /* Rx Queue 0 Interrupt */
 887 #define E1000_EICS_RX_QUEUE1    E1000_EICR_RX_QUEUE1 /* Rx Queue 1 Interrupt */
 888 #define E1000_EICS_RX_QUEUE2    E1000_EICR_RX_QUEUE2 /* Rx Queue 2 Interrupt */
 889 #define E1000_EICS_RX_QUEUE3    E1000_EICR_RX_QUEUE3 /* Rx Queue 3 Interrupt */
 890 #define E1000_EICS_TX_QUEUE0    E1000_EICR_TX_QUEUE0 /* Tx Queue 0 Interrupt */
 891 #define E1000_EICS_TX_QUEUE1    E1000_EICR_TX_QUEUE1 /* Tx Queue 1 Interrupt */
 892 #define E1000_EICS_TX_QUEUE2    E1000_EICR_TX_QUEUE2 /* Tx Queue 2 Interrupt */
 893 #define E1000_EICS_TX_QUEUE3    E1000_EICR_TX_QUEUE3 /* Tx Queue 3 Interrupt */
 894 #define E1000_EICS_TCP_TIMER    E1000_EICR_TCP_TIMER /* TCP Timer */
 895 #define E1000_EICS_OTHER        E1000_EICR_OTHER   /* Interrupt Cause Active */

 897 #define E1000_EITR_ITR_INT_MASK 0x0000FFFF

 899 /* Transmit Descriptor Control */
 900 #define E1000_TXDCTL_PTHRESH    0x0000003F /* TXDCTL Prefetch Threshold */
 901 #define E1000_TXDCTL_HTHRESH    0x00003F00 /* TXDCTL Host Threshold */
 902 #define E1000_TXDCTL_WTHRESH    0x003F0000 /* TXDCTL Writeback Threshold */
 903 #define E1000_TXDCTL_GRAN       0x01000000 /* TXDCTL Granularity */
 904 #define E1000_TXDCTL_LWTHRESH   0xFE000000 /* TXDCTL Low Threshold */
 905 #define E1000_TXDCTL_FULL_TX_DESC_WB    0x01010000 /* GRAN=1, WTHRESH=1 */
 906 #define E1000_TXDCTL_MAX_TX_DESC_PREFETCH 0x0100001F /* GRAN=1, PTHRESH=31 */
 907 /* Enable the counting of descriptors still to be processed. */
 908 #define E1000_TXDCTL_COUNT_DESC 0x00400000

 910 /* Flow Control Constants */
 911 #define FLOW_CONTROL_ADDRESS_LOW        0x00C28001
 912 #define FLOW_CONTROL_ADDRESS_HIGH       0x00000100
 913 #define FLOW_CONTROL_TYPE               0x8808

 915 /* 802.1q VLAN Packet Size */
 916 #define VLAN_TAG_SIZE                   4    /* 802.3ac tag (not DMA'd) */
 917 #define E1000_VLAN_FILTER_TBL_SIZE      128  /* VLAN Filter Table (4096 bits) */
```

```
 919 /* Receive Address */
 920 /*
 921  * Number of high/low register pairs in the RAR. The RAR (Receive Address
 922  * Registers) holds the directed and multicast addresses that we monitor.
 923  * Technically, we have 16 spots.  However, we reserve one of these spots
 924  * (RAR[15]) for our directed address used by controllers with
 925  * manageability enabled, allowing us room for 15 multicast addresses.
 926  */
 927 #define E1000_RAR_ENTRIES       15
 928 #define E1000_RAH_AV            0x80000000      /* Receive descriptor valid */
 929 #define E1000_RAL_MAC_ADDR_LEN  4
 930 #define E1000_RAH_MAC_ADDR_LEN  2
 931 #define E1000_RAH_POOL_MASK     0x03FC0000
 932 #define E1000_RAH_POOL_1        0x00040000

 934 /* Error Codes */
 935 #define E1000_SUCCESS           0
 936 #define E1000_ERR_NVM           1
 937 #define E1000_ERR_PHY           2
 938 #define E1000_ERR_CONFIG        3
 939 #define E1000_ERR_PARAM         4
 940 #define E1000_ERR_MAC_INIT      5
 941 #define E1000_ERR_PHY_TYPE      6
 942 #define E1000_ERR_RESET         9
 943 #define E1000_ERR_MASTER_REQUESTS_PENDING       10
 944 #define E1000_ERR_HOST_INTERFACE_COMMAND        11
 945 #define E1000_BLK_PHY_RESET     12
 946 #define E1000_ERR_SWFW_SYNC     13
 947 #define E1000_NOT_IMPLEMENTED   14
 948 #define E1000_ERR_MBX           15

 950 /* Loop limit on how long we wait for auto-negotiation to complete */
 951 #define FIBER_LINK_UP_LIMIT     50
 952 #define COPPER_LINK_UP_LIMIT    10
 953 #define PHY_AUTO_NEG_LIMIT      45
 954 #define PHY_FORCE_LIMIT         20
 955 /* Number of 100 microseconds we wait for PCI Express master disable */
 956 #define MASTER_DISABLE_TIMEOUT  800
 957 /* Number of milliseconds we wait for PHY configuration done after MAC reset */
 958 #define PHY_CFG_TIMEOUT         100
 959 /* Number of 2 milliseconds we wait for acquiring MDIO ownership. */
 960 #define MDIO_OWNERSHIP_TIMEOUT  10
 961 /* Number of milliseconds for NVM auto read done after MAC reset. */
 962 #define AUTO_READ_DONE_TIMEOUT  10

 964 /* Flow Control */
 965 #define E1000_FCRTH_RTH         0x0000FFF8 /* Mask Bits[15:3] for RTH */
 966 #define E1000_FCRTH_XFCE        0x80000000 /* External Flow Control Enable */
 967 #define E1000_FCRTL_RTL         0x0000FFF8 /* Mask Bits[15:3] for RTL */
 968 #define E1000_FCRTL_XONE        0x80000000 /* Enable XON frame transmission */

 970 /* Transmit Configuration Word */
 971 #define E1000_TXCW_FD           0x00000020 /* TXCW full duplex */
 972 #define E1000_TXCW_HD           0x00000040 /* TXCW half duplex */
 973 #define E1000_TXCW_PAUSE        0x00000080 /* TXCW sym pause request */
 974 #define E1000_TXCW_ASM_DIR      0x00000100 /* TXCW astm pause direction */
 975 #define E1000_TXCW_PAUSE_MASK   0x00000180 /* TXCW pause request mask */
 976 #define E1000_TXCW_RF           0x00003000 /* TXCW remote fault */
 977 #define E1000_TXCW_NP           0x00008000 /* TXCW next page */
 978 #define E1000_TXCW_CW           0x0000ffff /* TxConfigWord mask */
 979 #define E1000_TXCW_TXC          0x40000000 /* Transmit Config control */
 980 #define E1000_TXCW_ANE          0x80000000 /* Auto-neg enable */

 982 /* Receive Configuration Word */
 983 #define E1000_RXCW_CW           0x0000ffff /* RxConfigWord mask */
```



```
 984 #define E1000_RXCW_NC           0x04000000 /* Receive config no carrier */
 985 #define E1000_RXCW_IV           0x08000000 /* Receive config invalid */
 986 #define E1000_RXCW_CC           0x10000000 /* Receive config change */
 987 #define E1000_RXCW_C            0x20000000 /* Receive config */
 988 #define E1000_RXCW_SYNCH        0x40000000 /* Receive config synch */
 989 #define E1000_RXCW_ANC          0x80000000 /* Auto-neg complete */

 991 /* TUPLE Filtering Configuration */
 992 #define E1000_TTQF_DISABLE_MASK         0xF0008000 /* TTQF Disable Mask */
 993 #define E1000_TTQF_QUEUE_ENABLE         0x100 /* TTQF Queue Enable Bit */
 994 #define E1000_TTQF_PROTOCOL_MASK        0xFF /* TTQF Protocol Mask */
 995 /* TTQF TCP Bit, shift with E1000_TTQF_PROTOCOL SHIFT */
 996 #define E1000_TTQF_PROTOCOL_TCP         0x0
 997 /* TTQF UDP Bit, shift with E1000_TTQF_PROTOCOL_SHIFT */
 998 #define E1000_TTQF_PROTOCOL_UDP         0x1
 999 /* TTQF SCTP Bit, shift with E1000_TTQF_PROTOCOL_SHIFT */
1000 #define E1000_TTQF_PROTOCOL_SCTP        0x2
1001 #define E1000_TTQF_PROTOCOL_SHIFT       5 /* TTQF Protocol Shift */
1002 #define E1000_TTQF_QUEUE_SHIFT          16 /* TTQF Queue Shfit */
1003 #define E1000_TTQF_RX_QUEUE_MASK        0x70000 /* TTQF Queue Mask */
1004 #define E1000_TTQF_MASK_ENABLE          0x10000000 /* TTQF Mask Enable Bit */
1005 #define E1000_IMIR_CLEAR_MASK           0xF001FFFF /* IMIR Reg Clear Mask */
1006 #define E1000_IMIR_PORT_BYPASS          0x20000 /* IMIR Port Bypass Bit */
1007 #define E1000_IMIR_PRIORITY_SHIFT       29 /* IMIR Priority Shift */
1008 #define E1000_IMIREXT_CLEAR_MASK        0x7FFFF /* IMIREXT Reg Clear Mask */

1010 /* I350 EEE defines */
1011 #define E1000_IPCNFG_EEE_1G_AN          0x00000008 /* IPCNFG EEE Ena 1G AN */
1012 #define E1000_IPCNFG_EEE_100M_AN        0x00000004 /* IPCNFG EEE Ena 100M AN */
1013 #define E1000_EEER_TX_LPI_EN            0x00010000 /* EEER Tx LPI Enable */
1014 #define E1000_EEER_RX_LPI_EN            0x00020000 /* EEER Rx LPI Enable */
1015 #define E1000_EEER_LPI_FC               0x00040000 /* EEER Ena on Flow Cntrl */


1018 /* PCI Express Control */
1019 #define E1000_GCR_RXD_NO_SNOOP          0x00000001
1020 #define E1000_GCR_RXDSCW_NO_SNOOP       0x00000002
1021 #define E1000_GCR_RXDSCR_NO_SNOOP       0x00000004
1022 #define E1000_GCR_TXD_NO_SNOOP          0x00000008
1023 #define E1000_GCR_TXDSCW_NO_SNOOP       0x00000010
1024 #define E1000_GCR_TXDSCR_NO_SNOOP       0x00000020
1025 #define E1000_GCR_CMPL_TMOUT_MASK       0x0000F000
1026 #define E1000_GCR_CMPL_TMOUT_10ms       0x00001000
1027 #define E1000_GCR_CMPL_TMOUT_RESEND     0x00010000
1028 #define E1000_GCR_CAP_VER2              0x00040000

1030 #define PCIE_NO_SNOOP_ALL (E1000_GCR_RXD_NO_SNOOP       | \
1031                            E1000_GCR_RXDSCW_NO_SNOOP    | \
1032                            E1000_GCR_RXDSCR_NO_SNOOP    | \
1033                            E1000_GCR_TXD_NO_SNOOP       | \
1034                            E1000_GCR_TXDSCW_NO_SNOOP    | \
1035                            E1000_GCR_TXDSCR_NO_SNOOP)

1037 /* PHY Control Register */
1038 #define MII_CR_SPEED_SELECT_MSB 0x0040  /* bits 6,13: 10=1000, 01=100, 00=10 */
1039 #define MII_CR_COLL_TEST_ENABLE 0x0080  /* Collision test enable */
1040 #define MII_CR_FULL_DUPLEX      0x0100  /* FDX =1, half duplex =0 */
1041 #define MII_CR_RESTART_AUTO_NEG 0x0200  /* Restart auto negotiation */
1042 #define MII_CR_ISOLATE          0x0400  /* Isolate PHY from MII */
1043 #define MII_CR_POWER_DOWN       0x0800  /* Power down */
1044 #define MII_CR_AUTO_NEG_EN      0x1000  /* Auto Neg Enable */
1045 #define MII_CR_SPEED_SELECT_LSB 0x2000  /* bits 6,13: 10=1000, 01=100, 00=10 */
1046 #define MII_CR_LOOPBACK         0x4000  /* 0 = normal, 1 = loopback */
1047 #define MII_CR_RESET            0x8000  /* 0 = normal, 1 = PHY reset */
1048 #define MII_CR_SPEED_1000       0x0040
1049 #define MII_CR_SPEED_100        0x2000
```

```
1050 #define MII_CR_SPEED_10         0x0000

1052 /* PHY Status Register */
1053 #define MII_SR_EXTENDED_CAPS    0x0001 /* Extended register capabilities */
1054 #define MII_SR_JABBER_DETECT    0x0002 /* Jabber Detected */
1055 #define MII_SR_LINK_STATUS      0x0004 /* Link Status 1 = link */
1056 #define MII_SR_AUTONEG_CAPS     0x0008 /* Auto Neg Capable */
1057 #define MII_SR_REMOTE_FAULT     0x0010 /* Remote Fault Detect */
1058 #define MII_SR_AUTONEG_COMPLETE 0x0020 /* Auto Neg Complete */
1059 #define MII_SR_PREAMBLE_SUPPRESS 0x0040 /* Preamble may be suppressed */
1060 #define MII_SR_EXTENDED_STATUS  0x0100 /* Ext. status info in Reg 0x0F */
1061 #define MII_SR_100T2_HD_CAPS    0x0200 /* 100T2 Half Duplex Capable */
1062 #define MII_SR_100T2_FD_CAPS    0x0400 /* 100T2 Full Duplex Capable */
1063 #define MII_SR_10T_HD_CAPS      0x0800 /* 10T   Half Duplex Capable */
1064 #define MII_SR_10T_FD_CAPS      0x1000 /* 10T   Full Duplex Capable */
1065 #define MII_SR_100X_HD_CAPS     0x2000 /* 100X  Half Duplex Capable */
1066 #define MII_SR_100X_FD_CAPS     0x4000 /* 100X  Full Duplex Capable */
1067 #define MII_SR_100T4_CAPS       0x8000 /* 100T4 Capable */

1069 /* Autoneg Advertisement Register */
1070 #define NWAY_AR_SELECTOR_FIELD  0x0001 /* indicates IEEE 802.3 CSMA/CD */
1071 #define NWAY_AR_10T_HD_CAPS     0x0020 /* 10T   Half Duplex Capable */
1072 #define NWAY_AR_10T_FD_CAPS     0x0040 /* 10T   Full Duplex Capable */
1073 #define NWAY_AR_100TX_HD_CAPS   0x0080 /* 100TX Half Duplex Capable */
1074 #define NWAY_AR_100TX_FD_CAPS   0x0100 /* 100TX Full Duplex Capable */
1075 #define NWAY_AR_100T4_CAPS      0x0200 /* 100T4 Capable */
1076 #define NWAY_AR_PAUSE           0x0400 /* Pause operation desired */
1077 #define NWAY_AR_ASM_DIR         0x0800 /* Asymmetric Pause Direction bit */
1078 #define NWAY_AR_REMOTE_FAULT    0x2000 /* Remote Fault detected */
1079 #define NWAY_AR_NEXT_PAGE       0x8000 /* Next Page ability supported */

1081 /* Link Partner Ability Register (Base Page) */
1082 #define NWAY_LPAR_SELECTOR_FIELD 0x0000 /* LP protocol selector field */
1083 #define NWAY_LPAR_10T_HD_CAPS   0x0020 /* LP is 10T   Half Duplex Capable */
1084 #define NWAY_LPAR_10T_FD_CAPS   0x0040 /* LP is 10T   Full Duplex Capable */
1085 #define NWAY_LPAR_100TX_HD_CAPS 0x0080 /* LP is 100TX Half Duplex Capable */
1086 #define NWAY_LPAR_100TX_FD_CAPS 0x0100 /* LP is 100TX Full Duplex Capable */
1087 #define NWAY_LPAR_100T4_CAPS    0x0200 /* LP is 100T4 Capable */
1088 #define NWAY_LPAR_PAUSE         0x0400 /* LP Pause operation desired */
1089 #define NWAY_LPAR_ASM_DIR       0x0800 /* LP Asymmetric Pause Direction bit */
1090 #define NWAY_LPAR_REMOTE_FAULT  0x2000 /* LP has detected Remote Fault */
1091 #define NWAY_LPAR_ACKNOWLEDGE   0x4000 /* LP has rx'd link code word */
1092 #define NWAY_LPAR_NEXT_PAGE     0x8000 /* Next Page ability supported */

1094 /* Autoneg Expansion Register */
1095 #define NWAY_ER_LP_NWAY_CAPS    0x0001 /* LP has Auto Neg Capability */
1096 #define NWAY_ER_PAGE_RXD        0x0002 /* LP is 10T   Half Duplex Capable */
1097 #define NWAY_ER_NEXT_PAGE_CAPS  0x0004 /* LP is 10T   Full Duplex Capable */
1098 #define NWAY_ER_LP_NEXT_PAGE_CAPS 0x0008 /* LP is 100TX Half Duplex Capable */
1099 #define NWAY_ER_PAR_DETECT_FAULT  0x0010 /* LP is 100TX Full Duplex Capable */

1101 /* 1000BASE-T Control Register */
1102 #define CR_1000T_ASYM_PAUSE     0x0080 /* Advertise asymmetric pause bit */
1103 #define CR_1000T_HD_CAPS        0x0100 /* Advertise 1000T HD capability */
1104 #define CR_1000T_FD_CAPS        0x0200 /* Advertise 1000T FD capability  */
1105 #define CR_1000T_REPEATER_DTE   0x0400 /* 1=Repeater/switch device port */
1106                                         /* 0=DTE device */
1107 #define CR_1000T_MS_VALUE       0x0800 /* 1=Configure PHY as Master */
1108                                         /* 0=Configure PHY as Slave */
1109 #define CR_1000T_MS_ENABLE      0x1000 /* 1=Master/Slave manual config value */
1110                                        /* 0=Automatic Master/Slave config */
1111 #define CR_1000T_TEST_MODE_NORMAL 0x0000 /* Normal Operation */
1112 #define CR_1000T_TEST_MODE_1    0x2000 /* Transmit Waveform test */
1113 #define CR_1000T_TEST_MODE_2    0x4000 /* Master Transmit Jitter test */
1114 #define CR_1000T_TEST_MODE_3    0x6000 /* Slave Transmit Jitter test */
1115 #define CR_1000T_TEST_MODE_4    0x8000 /* Transmitter Distortion test */
```



```
1117 /* 1000BASE-T Status Register */
1118 #define SR_1000T_IDLE_ERROR_CNT 0x00FF /* Num idle errors since last read */
1119 #define SR_1000T_ASYM_PAUSE_DIR 0x0100 /* LP asymmetric pause direction bit */
1120 #define SR_1000T_LP_HD_CAPS     0x0400 /* LP is 1000T HD capable */
1121 #define SR_1000T_LP_FD_CAPS     0x0800 /* LP is 1000T FD capable */
1122 #define SR_1000T_REMOTE_RX_STATUS 0x1000 /* Remote receiver OK */
1123 #define SR_1000T_LOCAL_RX_STATUS 0x2000 /* Local receiver OK */
1124 #define SR_1000T_MS_CONFIG_RES  0x4000 /* 1=Local Tx is Master, 0=Slave */
1125 #define SR_1000T_MS_CONFIG_FAULT 0x8000 /* Master/Slave config fault */

1127 #define SR_1000T_PHY_EXCESSIVE_IDLE_ERR_COUNT   5

1129 /* PHY 1000 MII Register/Bit Definitions */
1130 /* PHY Registers defined by IEEE */
1131 #define PHY_CONTROL             0x00 /* Control Register */
1132 #define PHY_STATUS              0x01 /* Status Register */
1133 #define PHY_ID1                 0x02 /* Phy Id Reg (word 1) */
1134 #define PHY_ID2                 0x03 /* Phy Id Reg (word 2) */
1135 #define PHY_AUTONEG_ADV         0x04 /* Autoneg Advertisement */
1136 #define PHY_LP_ABILITY          0x05 /* Link Partner Ability (Base Page) */
1137 #define PHY_AUTONEG_EXP         0x06 /* Autoneg Expansion Reg */
1138 #define PHY_NEXT_PAGE_TX        0x07 /* Next Page Tx */
1139 #define PHY_LP_NEXT_PAGE        0x08 /* Link Partner Next Page */
1140 #define PHY_1000T_CTRL          0x09 /* 1000Base-T Control Reg */
1141 #define PHY_1000T_STATUS        0x0A /* 1000Base-T Status Reg */
1142 #define PHY_EXT_STATUS          0x0F /* Extended Status Reg */

1144 #define PHY_CONTROL_LB          0x4000 /* PHY Loopback bit */

1146 /* NVM Control */
1147 #define E1000_EECD_SK           0x00000001 /* NVM Clock */
1148 #define E1000_EECD_CS           0x00000002 /* NVM Chip Select */
1149 #define E1000_EECD_DI           0x00000004 /* NVM Data In */
1150 #define E1000_EECD_DO           0x00000008 /* NVM Data Out */
1151 #define E1000_EECD_FWE_MASK     0x00000030
1152 #define E1000_EECD_FWE_DIS      0x00000010 /* Disable FLASH writes */
1153 #define E1000_EECD_FWE_EN       0x00000020 /* Enable FLASH writes */
1154 #define E1000_EECD_FWE_SHIFT    4
1155 #define E1000_EECD_REQ          0x00000040 /* NVM Access Request */
1156 #define E1000_EECD_GNT          0x00000080 /* NVM Access Grant */
1157 #define E1000_EECD_PRES         0x00000100 /* NVM Present */
1158 #define E1000_EECD_SIZE         0x00000200 /* NVM Size (0=64 word 1=256 word) */
1159 #define E1000_EECD_BLOCKED      0x00008000 /* Bit banging access blocked flag */
1160 #define E1000_EECD_ABORT        0x00010000 /* NVM operation aborted flag */
1161 #define E1000_EECD_TIMEOUT      0x00020000 /* NVM read operation timeout flag */
1162 #define E1000_EECD_ERROR_CLR    0x00040000 /* NVM error status clear bit */

1164 /* NVM Addressing bits based on type 0=small, 1=large */
1165 #define E1000_EECD_ADDR_BITS    0x00000400
1166 #define E1000_EECD_TYPE         0x00002000 /* NVM Type (1-SPI, 0-Microwire) */
1167 #ifndef E1000_NVM_GRANT_ATTEMPTS
1168 #define E1000_NVM_GRANT_ATTEMPTS        1000 /* NVM # attempts to gain grant */
1169 #endif
1170 #define E1000_EECD_AUTO_RD      0x00000200 /* NVM Auto Read done */
1171 #define E1000_EECD_SIZE_EX_MASK 0x00007800 /* NVM Size */
1172 #define E1000_EECD_SIZE_EX_SHIFT        11
1173 #define E1000_EECD_NVADDS       0x00018000 /* NVM Address Size */
1174 #define E1000_EECD_SELSHAD      0x00020000 /* Select Shadow RAM */
1175 #define E1000_EECD_INITSRAM     0x00040000 /* Initialize Shadow RAM */
1176 #define E1000_EECD_FLUPD        0x00080000 /* Update FLASH */
1177 #define E1000_EECD_AUPDEN       0x00100000 /* Enable Autonomous FLASH update */
1178 #define E1000_EECD_SHADV        0x00200000 /* Shadow RAM Data Valid */
1179 #define E1000_EECD_SEC1VAL      0x00400000 /* Sector One Valid */
1180 #define E1000_EECD_SECVAL_SHIFT         22
1181 #define E1000_EECD_SEC1VAL_VALID_MASK   (E1000_EECD_AUTO_RD | E1000_EECD_PRES)
```

```
1183 #define E1000_NVM_SWDPIN0       0x0001  /* SWDPIN 0 NVM Value */
1184 #define E1000_NVM_LED_LOGIC     0x0020  /* Led Logic Word */
1185 #define E1000_NVM_RW_REG_DATA   16 /* Offset to data in NVM read/write regs */
1186 #define E1000_NVM_RW_REG_DONE   2 /* Offset to READ/WRITE done bit */
1187 #define E1000_NVM_RW_REG_START  1 /* Start operation */
1188 #define E1000_NVM_RW_ADDR_SHIFT 2 /* Shift to the address bits */
1189 #define E1000_NVM_POLL_WRITE    1 /* Flag for polling for write complete */
1190 #define E1000_NVM_POLL_READ     0 /* Flag for polling for read complete */
1191 #define E1000_FLASH_UPDATES     2000

1193 /* NVM Word Offsets */
1194 #define NVM_COMPAT                      0x0003
1195 #define NVM_ID_LED_SETTINGS             0x0004
1196 #define NVM_VERSION                     0x0005
1197 #define NVM_SERDES_AMPLITUDE            0x0006 /* SERDES output amplitude */
1198 #define NVM_PHY_CLASS_WORD              0x0007
1199 #define NVM_INIT_CONTROL1_REG           0x000A
1200 #define NVM_INIT_CONTROL2_REG           0x000F
1201 #define NVM_SWDEF_PINS_CTRL_PORT_1      0x0010
1202 #define NVM_INIT_CONTROL3_PORT_B        0x0014
1203 #define NVM_INIT_3GIO_3                 0x001A
1204 #define NVM_SWDEF_PINS_CTRL_PORT_0      0x0020
1205 #define NVM_INIT_CONTROL3_PORT_A        0x0024
1206 #define NVM_CFG                         0x0012
1207 #define NVM_FLASH_VERSION               0x0032
1208 #define NVM_ALT_MAC_ADDR_PTR            0x0037
1209 #define NVM_CHECKSUM_REG                0x003F

1211 #define E1000_NVM_CFG_DONE_PORT_0       0x40000 /* MNG config cycle done */
1212 #define E1000_NVM_CFG_DONE_PORT_1       0x80000 /* ...for second port */
1213 #define E1000_NVM_CFG_DONE_PORT_2       0x100000 /* ...for third port */
1214 #define E1000_NVM_CFG_DONE_PORT_3       0x200000 /* ...for fourth port */

1216 #define NVM_82580_LAN_FUNC_OFFSET(a) (a ? (0x40 + (0x40 * a)) : 0)

1218 /* Mask bits for fields in Word 0x0f of the NVM */
1219 #define NVM_WORD0F_PAUSE_MASK           0x3000
1220 #define NVM_WORD0F_PAUSE                0x1000
1221 #define NVM_WORD0F_ASM_DIR              0x2000
1222 #define NVM_WORD0F_ANE                  0x0800
1223 #define NVM_WORD0F_SWPDIO_EXT_MASK      0x00F0
1224 #define NVM_WORD0F_LPLU                 0x0001

1226 /* Mask bits for fields in Word 0x1a of the NVM */
1227 #define NVM_WORD1A_ASPM_MASK            0x000C

1229 /* For checksumming, the sum of all words in the NVM should equal 0xBABA. */
1230 #define NVM_SUM                         0xBABA

1232 #define NVM_MAC_ADDR_OFFSET             0
1233 #define NVM_PBA_OFFSET_0                8
1234 #define NVM_PBA_OFFSET_1                9
1235 #define NVM_RESERVED_WORD               0xFFFF
1236 #define NVM_PHY_CLASS_A                 0x8000
1237 #define NVM_SERDES_AMPLITUDE_MASK       0x000F
1238 #define NVM_SIZE_MASK                   0x1C00
1239 #define NVM_SIZE_SHIFT                  10
1240 #define NVM_WORD_SIZE_BASE_SHIFT        6
1241 #define NVM_SWDPIO_EXT_SHIFT            4

1243 /* NVM Commands - Microwire */
1244 #define NVM_READ_OPCODE_MICROWIRE       0x6  /* NVM read opcode */
1245 #define NVM_WRITE_OPCODE_MICROWIRE      0x5  /* NVM write opcode */
1246 #define NVM_ERASE_OPCODE_MICROWIRE      0x7  /* NVM erase opcode */
1247 #define NVM_EWEN_OPCODE_MICROWIRE       0x13 /* NVM erase/write enable */
```



```
1248 #define NVM_EWDS_OPCODE_MICROWIRE       0x10 /* NVM erase/write disable */

1250 /* NVM Commands - SPI */
1251 #define NVM_MAX_RETRY_SPI       5000 /* Max wait of 5ms, for RDY signal */
1252 #define NVM_READ_OPCODE_SPI     0x03 /* NVM read opcode */
1253 #define NVM_WRITE_OPCODE_SPI    0x02 /* NVM write opcode */
1254 #define NVM_A8_OPCODE_SPI       0x08 /* opcode bit-3 = address bit-8 */
1255 #define NVM_WREN_OPCODE_SPI     0x06 /* NVM set Write Enable latch */
1256 #define NVM_WRDI_OPCODE_SPI     0x04 /* NVM reset Write Enable latch */
1257 #define NVM_RDSR_OPCODE_SPI     0x05 /* NVM read Status register */
1258 #define NVM_WRSR_OPCODE_SPI     0x01 /* NVM write Status register */

1260 /* SPI NVM Status Register */
1261 #define NVM_STATUS_RDY_SPI      0x01
1262 #define NVM_STATUS_WEN_SPI      0x02
1263 #define NVM_STATUS_BP0_SPI      0x04
1264 #define NVM_STATUS_BP1_SPI      0x08
1265 #define NVM_STATUS_WPEN_SPI     0x80

1267 /* Word definitions for ID LED Settings */
1268 #define ID_LED_RESERVED_0000    0x0000
1269 #define ID_LED_RESERVED_FFFF    0xFFFF
1270 #define ID_LED_DEFAULT          ((ID_LED_OFF1_ON2  << 12) | \
1271                                 (ID_LED_OFF1_OFF2 <<  8) | \
1272                                 (ID_LED_DEF1_DEF2 <<  4) | \
1273                                 (ID_LED_DEF1_DEF2))
1274 #define ID_LED_DEF1_DEF2        0x1
1275 #define ID_LED_DEF1_ON2         0x2
1276 #define ID_LED_DEF1_OFF2        0x3
1277 #define ID_LED_ON1_DEF2         0x4
1278 #define ID_LED_ON1_ON2          0x5
1279 #define ID_LED_ON1_OFF2         0x6
1280 #define ID_LED_OFF1_DEF2        0x7
1281 #define ID_LED_OFF1_ON2         0x8
1282 #define ID_LED_OFF1_OFF2        0x9

1284 #define IGP_ACTIVITY_LED_MASK   0xFFFFF0FF
1285 #define IGP_ACTIVITY_LED_ENABLE 0x0300
1286 #define IGP_LED3_MODE           0x07000000

1288 /* PCI/PCI-X/PCI-EX Config space */
1289 #define PCI_HEADER_TYPE_REGISTER        0x0E
1290 #define PCIE_LINK_STATUS                0x12
1291 #define PCIE_DEVICE_CONTROL2            0x28

1293 #define PCI_HEADER_TYPE_MULTIFUNC       0x80
1294 #define PCIE_LINK_WIDTH_MASK            0x3F0
1295 #define PCIE_LINK_WIDTH_SHIFT           4
1296 #define PCIE_DEVICE_CONTROL2_16ms       0x0005

1298 #ifndef ETH_ADDR_LEN
1299 #define ETH_ADDR_LEN                    6
1300 #endif

1302 #define PHY_REVISION_MASK       0xFFFFFFF0
1303 #define MAX_PHY_REG_ADDRESS     0x1F  /* 5 bit address bus (0-0x1F) */
1304 #define MAX_PHY_MULTI_PAGE_REG  0xF

1306 /* Bit definitions for valid PHY IDs. */
1307 /*
1308  * I = Integrated
1309  * E = External
1310  */
1311 #define M88E1000_E_PHY_ID       0x01410C50
1312 #define M88E1000_I_PHY_ID       0x01410C30
1313 #define M88E1011_I_PHY_ID       0x01410C20
```

```
1314 #define IGP01E1000_I_PHY_ID     0x02A80380
1315 #define M88E1011_I_REV_4        0x04
1316 #define M88E1111_I_PHY_ID       0x01410CC0
1317 #define GG82563_E_PHY_ID        0x01410CA0
1318 #define IGP03E1000_E_PHY_ID     0x02A80390
1319 #define IFE_E_PHY_ID            0x02A80330
1320 #define IFE_PLUS_E_PHY_ID       0x02A80320
1321 #define IFE_C_E_PHY_ID          0x02A80310
1322 #define I82580_I_PHY_ID         0x015403A0
1323 #define I350_I_PHY_ID           0x015403B0
1324 #define IGP04E1000_E_PHY_ID     0x02A80391
1325 #define M88_VENDOR              0x0141

1327 /* M88E1000 Specific Registers */
1328 #define M88E1000_PHY_SPEC_CTRL          0x10 /* PHY Specific Control Register */
1329 #define M88E1000_PHY_SPEC_STATUS        0x11 /* PHY Specific Status Register */
1330 #define M88E1000_INT_ENABLE             0x12 /* Interrupt Enable Register */
1331 #define M88E1000_INT_STATUS             0x13 /* Interrupt Status Register */
1332 #define M88E1000_EXT_PHY_SPEC_CTRL      0x14 /* Extended PHY Specific Control */
1333 #define M88E1000_RX_ERR_CNTR            0x15 /* Receive Error Counter */

1335 #define M88E1000_PHY_EXT_CTRL           0x1A /* PHY extend control register */
1336 #define M88E1000_PHY_PAGE_SELECT        0x1D /* Reg29 for page number setting */
1337 #define M88E1000_PHY_GEN_CONTROL        0x1E /* Its meaning depends on reg 29 */
1338 #define M88E1000_PHY_VCO_REG_BIT8       0x100 /* Bits 8 & 11 are adjusted for */
1339 #define M88E1000_PHY_VCO_REG_BIT11      0x800    /* improved BER performance */

1341 /* M88E1000 PHY Specific Control Register */
1342 #define M88E1000_PSCR_JABBER_DISABLE    0x0001 /* 1=Jabber Function disabled */
1343 #define M88E1000_PSCR_POLARITY_REVERSAL 0x0002 /* 1=Polarity Reversal enabled */
1344 #define M88E1000_PSCR_SQE_TEST          0x0004 /* 1=SQE Test enabled */
1345 /* 1=CLK125 low, 0=CLK125 toggling */
1346 #define M88E1000_PSCR_CLK125_DISABLE    0x0010
1347 #define M88E1000_PSCR_MDI_MANUAL_MODE   0x0000 /* MDI Crossover Mode bits 6:5 */
1348                                                /* Manual MDI configuration */
1349 #define M88E1000_PSCR_MDIX_MANUAL_MODE  0x0020 /* Manual MDIX configuration */
1350 /* 1000BASE-T: Auto crossover, 100BASE-TX/10BASE-T: MDI Mode */
1351 #define M88E1000_PSCR_AUTO_X_1000T      0x0040
1352 /* Auto crossover enabled all speeds */
1353 #define M88E1000_PSCR_AUTO_X_MODE       0x0060
1354 /*
1355  * 1=Enable Extended 10BASE-T distance (Lower 10BASE-T Rx Threshold
1356  * 0=Normal 10BASE-T Rx Threshold
1357  */
1358 #define M88E1000_PSCR_EN_10BT_EXT_DIST  0x0080
1359 /* 1=5-bit interface in 100BASE-TX, 0=MII interface in 100BASE-TX */
1360 #define M88E1000_PSCR_MII_5BIT_ENABLE   0x0100
1361 #define M88E1000_PSCR_SCRAMBLER_DISABLE 0x0200 /* 1=Scrambler disable */
1362 #define M88E1000_PSCR_FORCE_LINK_GOOD   0x0400 /* 1=Force link good */
1363 #define M88E1000_PSCR_ASSERT_CRS_ON_TX  0x0800 /* 1=Assert CRS on Transmit */

1365 /* M88E1000 PHY Specific Status Register */
1366 #define M88E1000_PSSR_JABBER            0x0001 /* 1=Jabber */
1367 #define M88E1000_PSSR_REV_POLARITY      0x0002 /* 1=Polarity reversed */
1368 #define M88E1000_PSSR_DOWNSHIFT         0x0020 /* 1=Downshifted */
1369 #define M88E1000_PSSR_MDIX              0x0040 /* 1=MDIX; 0=MDI */
1370 /*
1371  * 0 = <50M
1372  * 1 = 50-80M
1373  * 2 = 80-110M
1374  * 3 = 110-140M
1375  * 4 = >140M
1376  */
1377 #define M88E1000_PSSR_CABLE_LENGTH      0x0380
1378 #define M88E1000_PSSR_LINK              0x0400 /* 1=Link up, 0=Link down */
1379 #define M88E1000_PSSR_SPD_DPLX_RESOLVED 0x0800 /* 1=Speed & Duplex resolved */
```



```
1380 #define M88E1000_PSSR_PAGE_RCVD         0x1000 /* 1=Page received */
1381 #define M88E1000_PSSR_DPLX              0x2000 /* 1=Duplex 0=Half Duplex */
1382 #define M88E1000_PSSR_SPEED             0xC000 /* Speed, bits 14:15 */
1383 #define M88E1000_PSSR_10MBS             0x0000 /* 00=10Mbs */
1384 #define M88E1000_PSSR_100MBS            0x4000 /* 01=100Mbs */
1385 #define M88E1000_PSSR_1000MBS           0x8000 /* 10=1000Mbs */

1387 #define M88E1000_PSSR_CABLE_LENGTH_SHIFT        7

1389 /* M88E1000 Extended PHY Specific Control Register */
1390 #define M88E1000_EPSCR_FIBER_LOOPBACK   0x4000 /* 1=Fiber loopback */
1391 /*
1392  * 1 = Lost lock detect enabled.
1393  * Will assert lost lock and bring
1394  * link down if idle not seen
1395  * within 1ms in 1000BASE-T
1396  */
1397 #define M88E1000_EPSCR_DOWN_NO_IDLE     0x8000
1398 /*
1399  * Number of times we will attempt to autonegotiate before downshifting if we
1400  * are the master
1401  */
1402 #define M88E1000_EPSCR_MASTER_DOWNSHIFT_MASK    0x0C00
1403 #define M88E1000_EPSCR_MASTER_DOWNSHIFT_1X      0x0000
1404 #define M88E1000_EPSCR_MASTER_DOWNSHIFT_2X      0x0400
1405 #define M88E1000_EPSCR_MASTER_DOWNSHIFT_3X      0x0800
1406 #define M88E1000_EPSCR_MASTER_DOWNSHIFT_4X      0x0C00
1407 /*
1408  * Number of times we will attempt to autonegotiate before downshifting if we
1409  * are the slave
1410  */
1411 #define M88E1000_EPSCR_SLAVE_DOWNSHIFT_MASK     0x0300
1412 #define M88E1000_EPSCR_SLAVE_DOWNSHIFT_DIS      0x0000
1413 #define M88E1000_EPSCR_SLAVE_DOWNSHIFT_1X       0x0100
1414 #define M88E1000_EPSCR_SLAVE_DOWNSHIFT_2X       0x0200
1415 #define M88E1000_EPSCR_SLAVE_DOWNSHIFT_3X       0x0300
1416 #define M88E1000_EPSCR_TX_CLK_2_5       0x0060 /* 2.5 MHz TX_CLK */
1417 #define M88E1000_EPSCR_TX_CLK_25        0x0070 /* 25  MHz TX_CLK */
1418 #define M88E1000_EPSCR_TX_CLK_0         0x0000 /* NO  TX_CLK */

1420 /* M88EC018 Rev 2 specific DownShift settings */
1421 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_MASK   0x0E00
1422 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_1X     0x0000
1423 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_2X     0x0200
1424 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_3X     0x0400
1425 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_4X     0x0600
1426 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_5X     0x0800
1427 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_6X     0x0A00
1428 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_7X     0x0C00
1429 #define M88EC018_EPSCR_DOWNSHIFT_COUNTER_8X     0x0E00

1431 /*
1432  * Bits...
1433  * 15-5: page
1434  * 4-0: register offset
1435  */
1436 #define GG82563_PAGE_SHIFT      5
1437 #define GG82563_REG(page, reg)  \
1438         (((page) << GG82563_PAGE_SHIFT) | ((reg) & MAX_PHY_REG_ADDRESS))
1439 #define GG82563_MIN_ALT_REG     30

1441 /* GG82563 Specific Registers */
1442 #define GG82563_PHY_SPEC_CTRL           \
1443         GG82563_REG(0, 16) /* PHY Specific Control */
1444 #define GG82563_PHY_SPEC_STATUS         \
1445         GG82563_REG(0, 17) /* PHY Specific Status */
```

```
1446 #define GG82563_PHY_INT_ENABLE          \
1447         GG82563_REG(0, 18) /* Interrupt Enable */
1448 #define GG82563_PHY_SPEC_STATUS_2       \
1449         GG82563_REG(0, 19) /* PHY Specific Status 2 */
1450 #define GG82563_PHY_RX_ERR_CNTR         \
1451         GG82563_REG(0, 21) /* Receive Error Counter */
1452 #define GG82563_PHY_PAGE_SELECT         \
1453         GG82563_REG(0, 22) /* Page Select */
1454 #define GG82563_PHY_SPEC_CTRL_2         \
1455         GG82563_REG(0, 26) /* PHY Specific Control 2 */
1456 #define GG82563_PHY_PAGE_SELECT_ALT     \
1457         GG82563_REG(0, 29) /* Alternate Page Select */
1458 #define GG82563_PHY_TEST_CLK_CTRL       \
1459         GG82563_REG(0, 30) /* Test Clock Control (use reg. 29 to select) */

1461 #define GG82563_PHY_MAC_SPEC_CTRL       \
1462         GG82563_REG(2, 21) /* MAC Specific Control Register */
1463 #define GG82563_PHY_MAC_SPEC_CTRL_2     \
1464         GG82563_REG(2, 26) /* MAC Specific Control 2 */

1466 #define GG82563_PHY_DSP_DISTANCE        \
1467         GG82563_REG(5, 26) /* DSP Distance */

1469 /* Page 193 - Port Control Registers */
1470 #define GG82563_PHY_KMRN_MODE_CTRL      \
1471         GG82563_REG(193, 16) /* Kumeran Mode Control */
1472 #define GG82563_PHY_PORT_RESET          \
1473         GG82563_REG(193, 17) /* Port Reset */
1474 #define GG82563_PHY_REVISION_ID         \
1475         GG82563_REG(193, 18) /* Revision ID */
1476 #define GG82563_PHY_DEVICE_ID           \
1477         GG82563_REG(193, 19) /* Device ID */
1478 #define GG82563_PHY_PWR_MGMT_CTRL       \
1479         GG82563_REG(193, 20) /* Power Management Control */
1480 #define GG82563_PHY_RATE_ADAPT_CTRL     \
1481         GG82563_REG(193, 25) /* Rate Adaptation Control */

1483 /* Page 194 - KMRN Registers */
1484 #define GG82563_PHY_KMRN_FIFO_CTRL_STAT \
1485         GG82563_REG(194, 16) /* FIFO's Control/Status */
1486 #define GG82563_PHY_KMRN_CTRL           \
1487         GG82563_REG(194, 17) /* Control */
1488 #define GG82563_PHY_INBAND_CTRL         \
1489         GG82563_REG(194, 18) /* Inband Control */
1490 #define GG82563_PHY_KMRN_DIAGNOSTIC     \
1491         GG82563_REG(194, 19) /* Diagnostic */
1492 #define GG82563_PHY_ACK_TIMEOUTS        \
1493         GG82563_REG(194, 20) /* Acknowledge Timeouts */
1494 #define GG82563_PHY_ADV_ABILITY         \
1495         GG82563_REG(194, 21) /* Advertised Ability */
1496 #define GG82563_PHY_LINK_PARTNER_ADV_ABILITY \
1497         GG82563_REG(194, 23) /* Link Partner Advertised Ability */
1498 #define GG82563_PHY_ADV_NEXT_PAGE       \
1499         GG82563_REG(194, 24) /* Advertised Next Page */
1500 #define GG82563_PHY_LINK_PARTNER_ADV_NEXT_PAGE \
1501         GG82563_REG(194, 25) /* Link Partner Advertised Next page */
1502 #define GG82563_PHY_KMRN_MISC           \
1503         GG82563_REG(194, 26) /* Misc. */

1505 /* MDI Control */
1506 #define E1000_MDIC_DATA_MASK    0x0000FFFF
1507 #define E1000_MDIC_REG_MASK     0x001F0000
1508 #define E1000_MDIC_REG_SHIFT    16
1509 #define E1000_MDIC_PHY_MASK     0x03E00000
1510 #define E1000_MDIC_PHY_SHIFT    21
1511 #define E1000_MDIC_OP_WRITE     0x04000000
```



```
1512 #define E1000_MDIC_OP_READ      0x08000000
1513 #define E1000_MDIC_READY        0x10000000
1514 #define E1000_MDIC_INT_EN       0x20000000
1515 #define E1000_MDIC_ERROR        0x40000000

1517 /* SerDes Control */
1518 #define E1000_GEN_CTL_READY     0x80000000
1519 #define E1000_GEN_CTL_ADDRESS_SHIFT     8
1520 #define E1000_GEN_POLL_TIMEOUT          640

1522 /* LinkSec register fields */
1523 #define E1000_LSECTXCAP_SUM_MASK        0x00FF0000
1524 #define E1000_LSECTXCAP_SUM_SHIFT       16
1525 #define E1000_LSECRXCAP_SUM_MASK        0x00FF0000
1526 #define E1000_LSECRXCAP_SUM_SHIFT       16

1528 #define E1000_LSECTXCTRL_EN_MASK        0x00000003
1529 #define E1000_LSECTXCTRL_DISABLE        0x0
1530 #define E1000_LSECTXCTRL_AUTH           0x1
1531 #define E1000_LSECTXCTRL_AUTH_ENCRYPT   0x2
1532 #define E1000_LSECTXCTRL_AISCI          0x00000020
1533 #define E1000_LSECTXCTRL_PNTHRSH_MASK   0xFFFFFF00
1534 #define E1000_LSECTXCTRL_RSV_MASK       0x000000D8

1536 #define E1000_LSECRXCTRL_EN_MASK        0x0000000C
1537 #define E1000_LSECRXCTRL_EN_SHIFT       2
1538 #define E1000_LSECRXCTRL_DISABLE        0x0
1539 #define E1000_LSECRXCTRL_CHECK          0x1
1540 #define E1000_LSECRXCTRL_STRICT         0x2
1541 #define E1000_LSECRXCTRL_DROP           0x3
1542 #define E1000_LSECRXCTRL_PLSH           0x00000040
1543 #define E1000_LSECRXCTRL_RP             0x00000080
1544 #define E1000_LSECRXCTRL_RSV_MASK       0xFFFFFF33

1546 /* DMA Coalescing register fields */

1548 /* DMA Coalescing Watchdog Timer */
1549 #define E1000_DMACR_DMACWT_MASK         0x00003FFF
1550 /* DMA Coalescing Receive Threshold */
1551 #define E1000_DMACR_DMACTHR_MASK        0x00FF0000
1552 #define E1000_DMACR_DMACTHR_SHIFT       16
1553 /* Lx when no PCIe transactions */
1554 #define E1000_DMACR_DMAC_LX_MASK        0x30000000
1555 #define E1000_DMACR_DMAC_LX_SHIFT       28
1556 /* Enable DMA Coalescing */
1557 #define E1000_DMACR_DMAC_EN             0x80000000
1558 /* DMA Coalescing Transmit Threshold */
1559 #define E1000_DMCTXTH_DMCTTHR_MASK      0x00000FFF
1560 /* Time to LX request */
1561 #define E1000_DMCTLX_TTLX_MASK          0x00000FFF
1562 /* Receive Traffic Rate Threshold */
1563 #define E1000_DMCRTRH_UTRESH_MASK       0x0007FFFF
1564 /* Rcv packet rate in current window */
1565 #define E1000_DMCRTRH_LRPRCW            0x80000000
1566 /* DMA Coal Rcv Traffic Current Cnt */
1567 #define E1000_DMCCNT_CCOUNT_MASK        0x01FFFFFF
1568 /* Flow ctrl Rcv Threshold High val */
1569 #define E1000_FCRTC_RTH_COAL_MASK       0x0003FFF0
1570 #define E1000_FCRTC_RTH_COAL_SHIFT      4
1571 /* Lx power decision based on DMA coal */
1572 #define E1000_PCIEMISC_LX_DECISION      0x00000080

1574 #ifdef __cplusplus
1575 }
_____unchanged_portion_omitted_
```

```
**********************************************************
   15673 Thu Feb  2 21:11:35 2012
new/usr/src/uts/common/io/igb/igb_hw.h
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.446.2.1 v3_3_14_3_BHSW1 */

#ifndef _IGB_HW_H
#define _IGB_HW_H

#ifdef __cplusplus
extern "C" {
#endif

#include "igb_osdep.h"
#include "igb_regs.h"
#include "igb_defines.h"

struct e1000_hw;

#define E1000_DEV_ID_82576                      0x10C9
#define E1000_DEV_ID_82576_FIBER                0x10E6
#define E1000_DEV_ID_82576_SERDES               0x10E7
#define E1000_DEV_ID_82576_QUAD_COPPER          0x10E8
#define E1000_DEV_ID_82576_QUAD_COPPER_ET2      0x1526
#define E1000_DEV_ID_82576_NS                   0x150A
#define E1000_DEV_ID_82576_NS_SERDES            0x1518
#define E1000_DEV_ID_82576_SERDES_QUAD          0x150D
#define E1000_DEV_ID_82575EB_COPPER             0x10A7
#define E1000_DEV_ID_82575EB_FIBER_SERDES       0x10A9
#define E1000_DEV_ID_82575GB_QUAD_COPPER        0x10D6
#define E1000_DEV_ID_82580_COPPER               0x150E
#define E1000_DEV_ID_82580_FIBER                0x150F
#define E1000_DEV_ID_82580_SERDES               0x1510
#define E1000_DEV_ID_82580_SGMII                0x1511
```



```
#define E1000_DEV_ID_82580_COPPER_DUAL          0x1516
#define E1000_DEV_ID_I350_COPPER                0x1521

#define E1000_REVISION_0 0
#define E1000_REVISION_1 1
#define E1000_REVISION_2 2
#define E1000_REVISION_3 3
#define E1000_REVISION_4 4

#define E1000_FUNC_0    0
#define E1000_FUNC_1    1
#define E1000_FUNC_2    2
#define E1000_FUNC_3    3

#define E1000_ALT_MAC_ADDRESS_OFFSET_LAN0       0
#define E1000_ALT_MAC_ADDRESS_OFFSET_LAN1       3
#define E1000_ALT_MAC_ADDRESS_OFFSET_LAN2       6
#define E1000_ALT_MAC_ADDRESS_OFFSET_LAN3       9

enum e1000_mac_type {
        e1000_undefined = 0,
        e1000_82575,
        e1000_82576,
        e1000_82580,
        e1000_i350,
        e1000_num_macs  /* List is 1-based, so subtract 1 for true count. */
};
______unchanged_portion_omitted_

struct e1000_dev_spec_82575 {
        bool sgmii_active;
        bool global_device_reset;
        int eee_disable;
};
______unchanged_portion_omitted_
```

```
**********************************************************
  126645 Thu Feb  2 21:11:35 2012
new/usr/src/uts/common/io/igb/igb_main.c
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include "igb_sw.h"

static char ident[] = "Intel 1Gb Ethernet";
static char igb_version[] = "igb 1.1.18";
static char igb_version[] = "igb 1.1.17";

/*
 * Local function protoypes
 */
static int igb_register_mac(igb_t *);
static int igb_identify_hardware(igb_t *);
static int igb_regs_map(igb_t *);
static void igb_init_properties(igb_t *);
static int igb_init_driver_settings(igb_t *);
static void igb_init_locks(igb_t *);
static void igb_destroy_locks(igb_t *);
static int igb_init_mac_address(igb_t *);
static int igb_init(igb_t *);
static int igb_init_adapter(igb_t *);
static void igb_stop_adapter(igb_t *);
static int igb_reset(igb_t *);
static void igb_tx_clean(igb_t *);
static boolean_t igb_tx_drain(igb_t *);
static boolean_t igb_rx_drain(igb_t *);
static int igb_alloc_rings(igb_t *);
static int igb_alloc_rx_data(igb_t *);
static void igb_free_rx_data(igb_t *);
static void igb_free_rings(igb_t *);
static void igb_setup_rings(igb_t *);
static void igb_setup_rx(igb_t *);
```



```
static void igb_setup_tx(igb_t *);
static void igb_setup_rx_ring(igb_rx_ring_t *);
static void igb_setup_tx_ring(igb_tx_ring_t *);
static void igb_setup_rss(igb_t *);
static void igb_setup_mac_rss_classify(igb_t *);
static void igb_setup_mac_classify(igb_t *);
static void igb_init_unicst(igb_t *);
static void igb_setup_multicst(igb_t *);
static void igb_get_phy_state(igb_t *);
static void igb_param_sync(igb_t *);
static void igb_get_conf(igb_t *);
static int igb_get_prop(igb_t *, char *, int, int, int);
static boolean_t igb_is_link_up(igb_t *);
static boolean_t igb_link_check(igb_t *);
static void igb_local_timer(void *);
static void igb_link_timer(void *);
static void igb_arm_watchdog_timer(igb_t *);
static void igb_start_watchdog_timer(igb_t *);
static void igb_restart_watchdog_timer(igb_t *);
static void igb_stop_watchdog_timer(igb_t *);
static void igb_start_link_timer(igb_t *);
static void igb_stop_link_timer(igb_t *);
static void igb_disable_adapter_interrupts(igb_t *);
static void igb_enable_adapter_interrupts_82575(igb_t *);
static void igb_enable_adapter_interrupts_82576(igb_t *);
static void igb_enable_adapter_interrupts_82580(igb_t *);
static boolean_t is_valid_mac_addr(uint8_t *);
static boolean_t igb_stall_check(igb_t *);
static boolean_t igb_set_loopback_mode(igb_t *, uint32_t);
static void igb_set_external_loopback(igb_t *);
static void igb_set_internal_phy_loopback(igb_t *);
static void igb_set_internal_serdes_loopback(igb_t *);
static boolean_t igb_find_mac_address(igb_t *);
static int igb_alloc_intrs(igb_t *);
static int igb_alloc_intr_handles(igb_t *, int);
static int igb_add_intr_handlers(igb_t *);
static void igb_rem_intr_handlers(igb_t *);
static void igb_rem_intrs(igb_t *);
static int igb_enable_intrs(igb_t *);
static int igb_disable_intrs(igb_t *);
static void igb_setup_msix_82575(igb_t *);
static void igb_setup_msix_82576(igb_t *);
static void igb_setup_msix_82580(igb_t *);
static uint_t igb_intr_legacy(void *, void *);
static uint_t igb_intr_msi(void *, void *);
static uint_t igb_intr_rx(void *, void *);
static uint_t igb_intr_tx(void *, void *);
static uint_t igb_intr_tx_other(void *, void *);
static void igb_intr_rx_work(igb_rx_ring_t *);
static void igb_intr_tx_work(igb_tx_ring_t *);
static void igb_intr_link_work(igb_t *);
static void igb_get_driver_control(struct e1000_hw *);
static void igb_release_driver_control(struct e1000_hw *);

static int igb_attach(dev_info_t *, ddi_attach_cmd_t);
static int igb_detach(dev_info_t *, ddi_detach_cmd_t);
static int igb_resume(dev_info_t *);
static int igb_suspend(dev_info_t *);
static int igb_quiesce(dev_info_t *);
static void igb_unconfigure(dev_info_t *, igb_t *);
static int igb_fm_error_cb(dev_info_t *, ddi_fm_error_t *,
    const void *);
static void igb_fm_init(igb_t *);
static void igb_fm_fini(igb_t *);
static void igb_release_multicast(igb_t *);
```

```
125 char *igb_priv_props[] = {
126         "_tx_copy_thresh",
127         "_tx_recycle_thresh",
128         "_tx_overload_thresh",
129         "_tx_resched_thresh",
130         "_rx_copy_thresh",
131         "_rx_limit_per_intr",
132         "_intr_throttling",
133         "_adv_pause_cap",
134         "_adv_asym_pause_cap",
135         NULL
136 };
```

*unchanged_portion_omitted_*

```
288 static adapter_info_t igb_i350_cap = {
289         /* limits */
290         8,              /* maximum number of rx queues */
291         1,              /* minimum number of rx queues */
292         4,              /* default number of rx queues */
293         8,              /* maximum number of tx queues */
294         1,              /* minimum number of tx queues */
295         4,              /* default number of tx queues */
296         65535,          /* maximum interrupt throttle rate */
297         0,              /* minimum interrupt throttle rate */
298         200,            /* default interrupt throttle rate */

300         /* function pointers */
301         igb_enable_adapter_interrupts_82580,
302         igb_setup_msix_82580,

304         /* capabilities */
305         (IGB_FLAG_HAS_DCA |     /* capability flags */
306         IGB_FLAG_VMDQ_POOL |
307         IGB_FLAG_NEED_CTX_IDX),

309         0xffe00000              /* mask for RXDCTL register */
310 };

312 /*
313  * Module Initialization Functions
314  */

316 int
317 _init(void)
318 {
319         int status;

321         mac_init_ops(&igb_dev_ops, MODULE_NAME);

323         status = mod_install(&igb_modlinkage);

325         if (status != DDI_SUCCESS) {
326                 mac_fini_ops(&igb_dev_ops);
327         }

329         return (status);
330 }
```

*unchanged_portion_omitted_*

```
357 /*
358  * igb_attach - driver attach
359  *
360  * This function is the device specific initialization entry
361  * point. This entry point is required and must be written.
362  * The DDI_ATTACH command must be provided in the attach entry
363  * point. When attach() is called with cmd set to DDI_ATTACH,
```



```
364  * all normal kernel services (such as kmem_alloc(9F)) are
365  * available for use by the driver.
366  *
367  * The attach() function will be called once for each instance
368  * of  the  device  on  the  system with cmd set to DDI_ATTACH.
369  * Until attach() succeeds, the only driver entry points which
370  * may be called are open(9E) and getinfo(9E).
371  */
372 static int
373 igb_attach(dev_info_t *devinfo, ddi_attach_cmd_t cmd)
374 {
375         igb_t *igb;
376         struct igb_osdep *osdep;
377         struct e1000_hw *hw;
378         int instance;

380         /*
381          * Check the command and perform corresponding operations
382          */
383         switch (cmd) {
384         default:
385                 return (DDI_FAILURE);

387         case DDI_RESUME:
388                 return (igb_resume(devinfo));

390         case DDI_ATTACH:
391                 break;
392         }

394         /* Get the device instance */
395         instance = ddi_get_instance(devinfo);

397         /* Allocate memory for the instance data structure */
398         igb = kmem_zalloc(sizeof (igb_t), KM_SLEEP);

400         igb->dip = devinfo;
401         igb->instance = instance;

403         hw = &igb->hw;
404         osdep = &igb->osdep;
405         hw->back = osdep;
406         osdep->igb = igb;

408         /* Attach the instance pointer to the dev_info data structure */
409         ddi_set_driver_private(devinfo, igb);


412         /* Initialize for fma support */
413         igb->fm_capabilities = igb_get_prop(igb, "fm-capable",
414             0, 0x0f,
415             DDI_FM_EREPORT_CAPABLE | DDI_FM_ACCCHK_CAPABLE |
416             DDI_FM_DMACHK_CAPABLE | DDI_FM_ERRCB_CAPABLE);
417         igb_fm_init(igb);
418         igb->attach_progress |= ATTACH_PROGRESS_FMINIT;

420         /*
421          * Map PCI config space registers
422          */
423         if (pci_config_setup(devinfo, &osdep->cfg_handle) != DDI_SUCCESS) {
424                 igb_error(igb, "Failed to map PCI configurations");
425                 goto attach_fail;
426         }
427         igb->attach_progress |= ATTACH_PROGRESS_PCI_CONFIG;

429         /*
```

```
430          * Identify the chipset family
431          */
432         if (igb_identify_hardware(igb) != IGB_SUCCESS) {
433                 igb_error(igb, "Failed to identify hardware");
434                 goto attach_fail;
435         }
436
437         /*
438          * Map device registers
439          */
440         if (igb_regs_map(igb) != IGB_SUCCESS) {
441                 igb_error(igb, "Failed to map device registers");
442                 goto attach_fail;
443         }
444         igb->attach_progress |= ATTACH_PROGRESS_REGS_MAP;
445
446         /*
447          * Initialize driver parameters
448          */
449         igb_init_properties(igb);
450         igb->attach_progress |= ATTACH_PROGRESS_PROPS;
451
452         /*
453          * Allocate interrupts
454          */
455         if (igb_alloc_intrs(igb) != IGB_SUCCESS) {
456                 igb_error(igb, "Failed to allocate interrupts");
457                 goto attach_fail;
458         }
459         igb->attach_progress |= ATTACH_PROGRESS_ALLOC_INTR;
460
461         /*
462          * Allocate rx/tx rings based on the ring numbers.
463          * The actual numbers of rx/tx rings are decided by the number of
464          * allocated interrupt vectors, so we should allocate the rings after
465          * interrupts are allocated.
466          */
467         if (igb_alloc_rings(igb) != IGB_SUCCESS) {
468                 igb_error(igb, "Failed to allocate rx/tx rings or groups");
469                 goto attach_fail;
470         }
471         igb->attach_progress |= ATTACH_PROGRESS_ALLOC_RINGS;
472
473         /*
474          * Add interrupt handlers
475          */
476         if (igb_add_intr_handlers(igb) != IGB_SUCCESS) {
477                 igb_error(igb, "Failed to add interrupt handlers");
478                 goto attach_fail;
479         }
480         igb->attach_progress |= ATTACH_PROGRESS_ADD_INTR;
481
482         /*
483          * Initialize driver parameters
484          */
485         if (igb_init_driver_settings(igb) != IGB_SUCCESS) {
486                 igb_error(igb, "Failed to initialize driver settings");
487                 goto attach_fail;
488         }
489
490         if (igb_check_acc_handle(igb->osdep.cfg_handle) != DDI_FM_OK) {
491                 ddi_fm_service_impact(igb->dip, DDI_SERVICE_LOST);
492                 goto attach_fail;
493         }
494
495         /*
```



```
496          * Initialize mutexes for this device.
497          * Do this before enabling the interrupt handler and
498          * register the softint to avoid the condition where
499          * interrupt handler can try using uninitialized mutex
500          */
501         igb_init_locks(igb);
502         igb->attach_progress |= ATTACH_PROGRESS_LOCKS;
503
504         /*
505          * Initialize the adapter
506          */
507         if (igb_init(igb) != IGB_SUCCESS) {
508                 igb_error(igb, "Failed to initialize adapter");
509                 goto attach_fail;
510         }
511         igb->attach_progress |= ATTACH_PROGRESS_INIT_ADAPTER;
512
513         /*
514          * Initialize statistics
515          */
516         if (igb_init_stats(igb) != IGB_SUCCESS) {
517                 igb_error(igb, "Failed to initialize statistics");
518                 goto attach_fail;
519         }
520         igb->attach_progress |= ATTACH_PROGRESS_STATS;
521
522         /*
523          * Register the driver to the MAC
524          */
525         if (igb_register_mac(igb) != IGB_SUCCESS) {
526                 igb_error(igb, "Failed to register MAC");
527                 goto attach_fail;
528         }
529         igb->attach_progress |= ATTACH_PROGRESS_MAC;
530
531         /*
532          * Now that mutex locks are initialized, and the chip is also
533          * initialized, enable interrupts.
534          */
535         if (igb_enable_intrs(igb) != IGB_SUCCESS) {
536                 igb_error(igb, "Failed to enable DDI interrupts");
537                 goto attach_fail;
538         }
539         igb->attach_progress |= ATTACH_PROGRESS_ENABLE_INTR;
540
541         igb_log(igb, "%s", igb_version);
542         atomic_or_32(&igb->igb_state, IGB_INITIALIZED);
543
544         /*
545          * Newer models have Energy Efficient Ethernet, let's disable this by
546          * default.
547          */
548         if (igb->hw.mac.type == e1000_i350)
549                 (void) e1000_set_eee_i350(&igb->hw);
550
551         return (DDI_SUCCESS);
552
553 attach_fail:
554         igb_unconfigure(devinfo, igb);
555         return (DDI_FAILURE);
556 }
_____unchanged_portion_omitted_

827 /*
828  * igb_identify_hardware - Identify the type of the chipset
829  */
```

```
 830 static int
 831 igb_identify_hardware(igb_t *igb)
 832 {
 833         struct e1000_hw *hw = &igb->hw;
 834         struct igb_osdep *osdep = &igb->osdep;

 836         /*
 837          * Get the device id
 838          */
 839         hw->vendor_id =
 840             pci_config_get16(osdep->cfg_handle, PCI_CONF_VENID);
 841         hw->device_id =
 842             pci_config_get16(osdep->cfg_handle, PCI_CONF_DEVID);
 843         hw->revision_id =
 844             pci_config_get8(osdep->cfg_handle, PCI_CONF_REVID);
 845         hw->subsystem_device_id =
 846             pci_config_get16(osdep->cfg_handle, PCI_CONF_SUBSYSID);
 847         hw->subsystem_vendor_id =
 848             pci_config_get16(osdep->cfg_handle, PCI_CONF_SUBVENID);

 850         /*
 851          * Set the mac type of the adapter based on the device id
 852          */
 853         if (e1000_set_mac_type(hw) != E1000_SUCCESS) {
 854                 return (IGB_FAILURE);
 855         }

 857         /*
 858          * Install adapter capabilities based on mac type
 859          */
 860         switch (hw->mac.type) {
 861         case e1000_82575:
 862                 igb->capab = &igb_82575_cap;
 863                 break;
 864         case e1000_82576:
 865                 igb->capab = &igb_82576_cap;
 866                 break;
 867         case e1000_82580:
 868                 igb->capab = &igb_82580_cap;
 869                 break;
 870         case e1000_i350:
 871                 igb->capab = &igb_i350_cap;
 872                 break;
 873         default:
 874                 return (IGB_FAILURE);
 875         }

 877         return (IGB_SUCCESS);
 878 }
______unchanged_portion_omitted_

1675 /*
1676  * igb_start - Start the driver/chipset
1677  */
1678 int
1679 igb_start(igb_t *igb, boolean_t alloc_buffer)
1680 {
1681         int i;

1683         ASSERT(mutex_owned(&igb->gen_lock));

1685         if (alloc_buffer) {
1686                 if (igb_alloc_rx_data(igb) != IGB_SUCCESS) {
1687                         igb_error(igb,
1688                             "Failed to allocate software receive rings");
1689                         return (IGB_FAILURE);
```



```
1690                 }

1692                 /* Allocate buffers for all the rx/tx rings */
1693                 if (igb_alloc_dma(igb) != IGB_SUCCESS) {
1694                         igb_error(igb, "Failed to allocate DMA resource");
1695                         return (IGB_FAILURE);
1696                 }

1698                 igb->tx_ring_init = B_TRUE;
1699         } else {
1700                 igb->tx_ring_init = B_FALSE;
1701         }

1703         for (i = 0; i < igb->num_rx_rings; i++)
1704                 mutex_enter(&igb->rx_rings[i].rx_lock);
1705         for (i = 0; i < igb->num_tx_rings; i++)
1706                 mutex_enter(&igb->tx_rings[i].tx_lock);

1708         /*
1709          * Start the adapter
1710          */
1711         if ((igb->attach_progress & ATTACH_PROGRESS_INIT_ADAPTER) == 0) {
1712                 if (igb_init_adapter(igb) != IGB_SUCCESS) {
1713                         igb_fm_ereport(igb, DDI_FM_DEVICE_INVAL_STATE);
1714                         goto start_failure;
1715                 }
1716                 igb->attach_progress |= ATTACH_PROGRESS_INIT_ADAPTER;
1717         }

1719         /*
1720          * Setup the rx/tx rings
1721          */
1722         igb_setup_rings(igb);

1724         /*
1725          * Enable adapter interrupts
1726          * The interrupts must be enabled after the driver state is START
1727          */
1728         igb->capab->enable_intr(igb);

1730         if (igb_check_acc_handle(igb->osdep.cfg_handle) != DDI_FM_OK)
1731                 goto start_failure;

1733         if (igb_check_acc_handle(igb->osdep.reg_handle) != DDI_FM_OK)
1734                 goto start_failure;

1736         if (igb->hw.mac.type == e1000_i350)
1737                 (void) e1000_set_eee_i350(&igb->hw);

1739         for (i = igb->num_tx_rings - 1; i >= 0; i--)
1740                 mutex_exit(&igb->tx_rings[i].tx_lock);
1741         for (i = igb->num_rx_rings - 1; i >= 0; i--)
1742                 mutex_exit(&igb->rx_rings[i].rx_lock);

1744         return (IGB_SUCCESS);

1746 start_failure:
1747         for (i = igb->num_tx_rings - 1; i >= 0; i--)
1748                 mutex_exit(&igb->tx_rings[i].tx_lock);
1749         for (i = igb->num_rx_rings - 1; i >= 0; i--)
1750                 mutex_exit(&igb->rx_rings[i].rx_lock);

1752         ddi_fm_service_impact(igb->dip, DDI_SERVICE_LOST);

1754         return (IGB_FAILURE);
1755 }
______unchanged_portion_omitted_
```

```
**********************************************************
   30071 Thu Feb  2 21:11:36 2012
new/usr/src/uts/common/io/igb/igb_regs.h
2038 Add in I350 and ET2 support into igb
Reviewed by: Dan McDonald <danmcd@nexenta.com>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 * Copyright(c) 2007-2010 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.82.2.1 v3_3_14_3_BHSW1 */

#ifndef _IGB_REGS_H
#define _IGB_REGS_H

#ifdef __cplusplus
extern "C" {
#endif

#define E1000_CTRL      0x00000  /* Device Control - RW */
#define E1000_CTRL_DUP  0x00004  /* Device Control Duplicate (Shadow) - RW */
#define E1000_STATUS    0x00008  /* Device Status - RO */
#define E1000_EECD      0x00010  /* EEPROM/Flash Control - RW */
#define E1000_EERD      0x00014  /* EEPROM Read - RW */
#define E1000_CTRL_EXT  0x00018  /* Extended Device Control - RW */
#define E1000_FLA       0x0001C  /* Flash Access - RW */
#define E1000_MDIC      0x00020  /* MDI Control - RW */
#define E1000_MDICNFG   0x00E04  /* MDI Config - RW */
#define E1000_REGISTER_SET_SIZE         0x20000 /* CSR Size */
#define E1000_EEPROM_INIT_CTRL_WORD_2   0x0F /* EEPROM Init Ctrl Word 2 */
#define E1000_BARCTRL                   0x5BBC /* BAR ctrl reg */
#define E1000_BARCTRL_FLSIZE            0x0700 /* BAR ctrl Flsize */
#define E1000_BARCTRL_CSRSIZE           0x2000 /* BAR ctrl CSR size */
#define E1000_SCTL      0x00024  /* SerDes Control - RW */
#define E1000_FCAL      0x00028  /* Flow Control Address Low - RW */
#define E1000_FCAH      0x0002C  /* Flow Control Address High -RW */
#define E1000_FEXT      0x0002C  /* Future Extended - RW */
#define E1000_FEXTNVM   0x00028  /* Future Extended NVM - RW */
#define E1000_FCT       0x00030  /* Flow Control Type - RW */
#define E1000_CONNSW    0x00034  /* Copper/Fiber switch control - RW */
```



```
#define E1000_VET       0x00038  /* VLAN Ether Type - RW */
#define E1000_ICR       0x000C0  /* Interrupt Cause Read - R/clr */
#define E1000_ITR       0x000C4  /* Interrupt Throttling Rate - RW */
#define E1000_ICS       0x000C8  /* Interrupt Cause Set - WO */
#define E1000_IMS       0x000D0  /* Interrupt Mask Set - RW */
#define E1000_IMC       0x000D8  /* Interrupt Mask Clear - WO */
#define E1000_IAM       0x000E0  /* Interrupt Acknowledge Auto Mask */
#define E1000_RCTL      0x00100  /* Rx Control - RW */
#define E1000_FCTTV     0x00170  /* Flow Control Transmit Timer Value - RW */
#define E1000_TXCW      0x00178  /* Tx Configuration Word - RW */
#define E1000_RXCW      0x00180  /* Rx Configuration Word - RO */
#define E1000_EICR      0x01580  /* Ext. Interrupt Cause Read - R/clr */
#define E1000_EITR(_n)  (0x01680 + (0x4 * (_n)))
#define E1000_EICS      0x01520  /* Ext. Interrupt Cause Set - W0 */
#define E1000_EIMS      0x01524  /* Ext. Interrupt Mask Set/Read - RW */
#define E1000_EIMC      0x01528  /* Ext. Interrupt Mask Clear - WO */
#define E1000_EIAC      0x0152C  /* Ext. Interrupt Auto Clear - RW */
#define E1000_EIAM      0x01530  /* Ext. Interrupt Ack Auto Clear Mask - RW */
#define E1000_GPIE      0x01514  /* General Purpose Interrupt Enable - RW */
#define E1000_IVAR0     0x01700  /* Interrupt Vector Allocation (array) - RW */
#define E1000_IVAR_MISC 0x01740 /* IVAR for "other" causes - RW */
#define E1000_TCTL      0x00400  /* Tx Control - RW */
#define E1000_TCTL_EXT  0x00404  /* Extended Tx Control - RW */
#define E1000_TIPG      0x00410  /* Tx Inter-packet gap -RW */
#define E1000_TBT       0x00448  /* Tx Burst Timer - RW */
#define E1000_AIT       0x00458  /* Adaptive Interframe Spacing Throttle - RW */
#define E1000_LEDCTL    0x00E00  /* LED Control - RW */
#define E1000_EXTCNF_CTRL       0x00F00  /* Extended Configuration Control */
#define E1000_EXTCNF_SIZE       0x00F08  /* Extended Configuration Size */
#define E1000_PHY_CTRL          0x00F10  /* PHY Control Register in CSR */
#define E1000_PBA       0x01000  /* Packet Buffer Allocation - RW */
#define E1000_PBS       0x01008  /* Packet Buffer Size */
#define E1000_EEMNGCTL  0x01010  /* MNG EEprom Control */
#define E1000_EEARBC    0x01024  /* EEPROM Auto Read Bus Control */
#define E1000_FLASHT    0x01028  /* FLASH Timer Register */
#define E1000_EEWR      0x0102C  /* EEPROM Write Register - RW */
#define E1000_FLSWCTL   0x01030  /* FLASH control register */
#define E1000_FLSWDATA  0x01034  /* FLASH data register */
#define E1000_FLSWCNT   0x01038  /* FLASH Access Counter */
#define E1000_FLOP      0x0103C  /* FLASH Opcode Register */
#define E1000_I2CCMD    0x01028  /* SFPI2C Command Register - RW */
#define E1000_I2CPARAMS 0x0102C  /* SFPI2C Parameters Register - RW */
#define E1000_WDSTP     0x01040  /* Watchdog Setup - RW */
#define E1000_SWDSTS    0x01044  /* SW Device Status - RW */
#define E1000_FRTIMER   0x01048  /* Free Running Timer - RW */
#define E1000_TCPTIMER  0x0104C  /* TCP Timer - RW */
#define E1000_VPDDIAG   0x01060  /* VPD Diagnostic - RO */
#define E1000_ICR_V2    0x01500  /* Interrupt Cause - new location - RC */
#define E1000_ICS_V2    0x01504  /* Interrupt Cause Set - new location - WO */
/* Interrupt Mask Set/Read - new location - RW */
#define E1000_IMS_V2    0x01508
#define E1000_IMC_V2    0x0150C  /* Interrupt Mask Clear - new location - WO */
/* Interrupt Ack Auto Mask - new location - RW */
#define E1000_IAM_V2    0x01510
#define E1000_ERT       0x02008  /* Early Rx Threshold - RW */
#define E1000_FCRTL     0x02160  /* Flow Control Receive Threshold Low - RW */
#define E1000_FCRTH     0x02168  /* Flow Control Receive Threshold High - RW */
#define E1000_PSRCTL    0x02170  /* Packet Split Receive Control - RW */
#define E1000_RDFPCQ(_n)        (0x02430 + (0x4 * (_n)))
#define E1000_PBRTH     0x02458  /* PB Rx Arbitration Threshold - RW */
#define E1000_FCRTV     0x02460  /* Flow Control Refresh Timer Value - RW */
/* Split and Replication Rx Control - RW */
#define E1000_RDPUMB    0x025CC  /* DMA Rx Descriptor uC Mailbox - RW */
#define E1000_RDPUAD    0x025D0  /* DMA Rx Descriptor uC Addr Command - RW */
#define E1000_RDPUWD    0x025D4  /* DMA Rx Descriptor uC Data Write - RW */
#define E1000_RDPURD    0x025D8  /* DMA Rx Descriptor uC Data Read - RW */
```

```
126 #define E1000_RDPUCTL   0x025DC  /* DMA Rx Descriptor uC Control - RW */
127 #define E1000_PBDIAG    0x02458  /* Packet Buffer Diagnostic - RW */
128 #define E1000_RXPBS     0x02404  /* Rx Packet Buffer Size - RW */
129 /* Same as RXPBS, renamed for newer adapters - RW */
130 #define E1000_IRPBS     0x02404
131 #define E1000_RDTR      0x02820  /* Rx Delay Timer - RW */
132 #define E1000_RADV      0x0282C  /* Rx Interrupt Absolute Delay Timer - RW */
133 /*
134  * Convenience macros
135  *
136  * Note: "_n" is the queue number of the register to be written to.
137  *
138  * Example usage:
139  * E1000_RDBAL_REG(current_rx_queue)
140  */
141 #define E1000_RDBAL(_n)         ((_n) < 4 ?     \
142         (0x02800 + ((_n) * 0x100)) :    \
143         (0x0C000 + ((_n) * 0x40)))
144 #define E1000_RDBAH(_n)         ((_n) < 4 ?     \
145         (0x02804 + ((_n) * 0x100)) :    \
146         (0x0C004 + ((_n) * 0x40)))
147 #define E1000_RDLEN(_n)         ((_n) < 4 ?     \
148         (0x02808 + ((_n) * 0x100)) :    \
149         (0x0C008 + ((_n) * 0x40)))
150 #define E1000_SRRCTL(_n)        ((_n) < 4 ?     \
151         (0x0280C + ((_n) * 0x100)) :    \
152         (0x0C00C + ((_n) * 0x40)))
153 #define E1000_RDH(_n)           ((_n) < 4 ?     \
154         (0x02810 + ((_n) * 0x100)) :    \
155         (0x0C010 + ((_n) * 0x40)))
156 #define E1000_RXCTL(_n)         ((_n) < 4 ?     \
157         (0x02814 + ((_n) * 0x100)) :    \
158         (0x0C014 + ((_n) * 0x40)))
159 #define E1000_DCA_RXCTRL(_n)    E1000_RXCTL(_n)
160 #define E1000_RDT(_n)           ((_n) < 4 ?     \
161         (0x02818 + ((_n) * 0x100)) :    \
162         (0x0C018 + ((_n) * 0x40)))
163 #define E1000_RXDCTL(_n)        ((_n) < 4 ?     \
164         (0x02828 + ((_n) * 0x100)) :    \
165         (0x0C028 + ((_n) * 0x40)))
166 #define E1000_RQDPC(_n)         ((_n) < 4 ?     \
167         (0x02830 + ((_n) * 0x100)) :    \
168         (0x0C030 + ((_n) * 0x40)))
169 #define E1000_TDBAL(_n)         ((_n) < 4 ?     \
170         (0x03800 + ((_n) * 0x100)) :    \
171         (0x0E000 + ((_n) * 0x40)))
172 #define E1000_TDBAH(_n)         ((_n) < 4 ?     \
173         (0x03804 + ((_n) * 0x100)) :    \
174         (0x0E004 + ((_n) * 0x40)))
175 #define E1000_TDLEN(_n)         ((_n) < 4 ?     \
176         (0x03808 + ((_n) * 0x100)) :    \
177         (0x0E008 + ((_n) * 0x40)))
178 #define E1000_TDH(_n)           ((_n) < 4 ?     \
179         (0x03810 + ((_n) * 0x100)) :    \
180         (0x0E010 + ((_n) * 0x40)))
181 #define E1000_TXCTL(_n)         ((_n) < 4 ?     \
182         (0x03814 + ((_n) * 0x100)) :    \
183         (0x0E014 + ((_n) * 0x40)))
184 #define E1000_DCA_TXCTRL(_n)    E1000_TXCTL(_n)
185 #define E1000_TDT(_n)           ((_n) < 4 ?     \
186         (0x03818 + ((_n) * 0x100)) :    \
187         (0x0E018 + ((_n) * 0x40)))
188 #define E1000_TXDCTL(_n)        ((_n) < 4 ?     \
189         (0x03828 + ((_n) * 0x100)) :    \
190         (0x0E028 + ((_n) * 0x40)))
191 #define E1000_TDWBAL(_n)        ((_n) < 4 ?     \
```



```
192         (0x03838 + ((_n) * 0x100)) :    \
193         (0x0E038 + ((_n) * 0x40)))
194 #define E1000_TDWBAH(_n)        ((_n) < 4 ?     \
195         (0x0383C + ((_n) * 0x100)) :    \
196         (0x0E03C + ((_n) * 0x40)))
197 #define E1000_TARC(_n)          (0x03840 + ((_n) * 0x100))
198 #define E1000_RSRPD     0x02C00  /* Rx Small Packet Detect - RW */
199 #define E1000_RAID      0x02C08  /* Receive Ack Interrupt Delay - RW */
200 #define E1000_TXDMAC    0x03000  /* Tx DMA Control - RW */
201 #define E1000_KABGTXD   0x03004  /* AFE Band Gap Transmit Ref Data */
202 #define E1000_PSRTYPE(_i)       (0x05480 + ((_i) * 4))
203 #define E1000_RAL(_i)           (((_i) <= 15) ? \
204         (0x05400 + ((_i) * 8)) : \
205         (0x054E0 + ((_i - 16) * 8)))
206 #define E1000_RAH(_i)           (((_i) <= 15) ? \
207         (0x05404 + ((_i) * 8)) : \
208         (0x054E4 + ((_i - 16) * 8)))
209 #define E1000_IP4AT_REG(_i)     (0x05840 + ((_i) * 8))
210 #define E1000_IP6AT_REG(_i)     (0x05880 + ((_i) * 4))
211 #define E1000_WUPM_REG(_i)      (0x05A00 + ((_i) * 4))
212 #define E1000_FFMT_REG(_i)      (0x09000 + ((_i) * 8))
213 #define E1000_FFVT_REG(_i)      (0x09800 + ((_i) * 8))
214 #define E1000_FFLT_REG(_i)      (0x05F00 + ((_i) * 8))
215 #define E1000_PBSLAC    0x03100  /* Packet Buffer Slave Access Control */
216 /* Packet Buffer DWORD (_n) */
217 #define E1000_PBSLAD(_n)        (0x03110 + (0x4 * (_n)))
218 #define E1000_TXPBS     0x03404  /* Tx Packet Buffer Size - RW */
219 /* Same as TXPBS, renamed for newer adpaters - RW */
220 #define E1000_ITPBS     0x03404
221 #define E1000_TDFH      0x03410  /* Tx Data FIFO Head - RW */
222 #define E1000_TDFT      0x03418  /* Tx Data FIFO Tail - RW */
223 #define E1000_TDFHS     0x03420  /* Tx Data FIFO Head Saved - RW */
224 #define E1000_TDFTS     0x03428  /* Tx Data FIFO Tail Saved - RW */
225 #define E1000_TDFPC     0x03430  /* Tx Data FIFO Packet Count - RW */
226 #define E1000_TDPUMB    0x0357C  /* DMA Tx Descriptor uC Mail Box - RW */
227 #define E1000_TDPUAD    0x03580  /* DMA Tx Descriptor uC Addr Command - RW */
228 #define E1000_TDPUWD    0x03584  /* DMA Tx Descriptor uC Data Write - RW */
229 #define E1000_TDPURD    0x03588  /* DMA Tx Descriptor uC Data  Read  - RW */
230 #define E1000_TDPUCTL   0x0358C  /* DMA Tx Descriptor uC Control - RW */
231 #define E1000_DTXCTL    0x03590  /* DMA Tx Control - RW */
232 #define E1000_DTXTCPFLGL 0x0359C /* DMA Tx Control flag low - RW */
233 #define E1000_DTXTCPFLGH 0x035A0 /* DMA Tx Control flag high - RW */
234 #define E1000_DTXMXSZRQ  0x03540 /* DMA Tx Max Total Allow Size Requests - RW */
235 #define E1000_TIDV      0x03820  /* Tx Interrupt Delay Value - RW */
236 #define E1000_TADV      0x0382C  /* Tx Interrupt Absolute Delay Val - RW */
237 #define E1000_TSPMT     0x03830  /* TCP Segmentation PAD & Min Threshold - RW */
238 #define E1000_CRCERRS   0x04000  /* CRC Error Count - R/clr */
239 #define E1000_ALGNERRC  0x04004  /* Alignment Error Count - R/clr */
240 #define E1000_SYMERRS   0x04008  /* Symbol Error Count - R/clr */
241 #define E1000_RXERRC    0x0400C  /* Receive Error Count - R/clr */
242 #define E1000_MPC       0x04010  /* Missed Packet Count - R/clr */
243 #define E1000_SCC       0x04014  /* Single Collision Count - R/clr */
244 #define E1000_ECOL      0x04018  /* Excessive Collision Count - R/clr */
245 #define E1000_MCC       0x0401C  /* Multiple Collision Count - R/clr */
246 #define E1000_LATECOL   0x04020  /* Late Collision Count - R/clr */
247 #define E1000_COLC      0x04028  /* Collision Count - R/clr */
248 #define E1000_DC        0x04030  /* Defer Count - R/clr */
249 #define E1000_TNCRS     0x04034  /* Tx-No CRS - R/clr */
250 #define E1000_SEC       0x04038  /* Sequence Error Count - R/clr */
251 #define E1000_CEXTERR   0x0403C  /* Carrier Extension Error Count - R/clr */
252 #define E1000_RLEC      0x04040  /* Receive Length Error Count - R/clr */
253 #define E1000_XONRXC    0x04048  /* XON Rx Count - R/clr */
254 #define E1000_XONTXC    0x0404C  /* XON Tx Count - R/clr */
255 #define E1000_XOFFRXC   0x04050  /* XOFF Rx Count - R/clr */
256 #define E1000_XOFFTXC   0x04054  /* XOFF Tx Count - R/clr */
257 #define E1000_FCRUC     0x04058  /* Flow Control Rx Unsupported Count- R/clr */
```

```
258 #define E1000_PRC64     0x0405C  /* Packets Rx (64 bytes) - R/clr */
259 #define E1000_PRC127    0x04060  /* Packets Rx (65-127 bytes) - R/clr */
260 #define E1000_PRC255    0x04064  /* Packets Rx (128-255 bytes) - R/clr */
261 #define E1000_PRC511    0x04068  /* Packets Rx (255-511 bytes) - R/clr */
262 #define E1000_PRC1023   0x0406C  /* Packets Rx (512-1023 bytes) - R/clr */
263 #define E1000_PRC1522   0x04070  /* Packets Rx (1024-1522 bytes) - R/clr */
264 #define E1000_GPRC      0x04074  /* Good Packets Rx Count - R/clr */
265 #define E1000_BPRC      0x04078  /* Broadcast Packets Rx Count - R/clr */
266 #define E1000_MPRC      0x0407C  /* Multicast Packets Rx Count - R/clr */
267 #define E1000_GPTC      0x04080  /* Good Packets Tx Count - R/clr */
268 #define E1000_GORCL     0x04088  /* Good Octets Rx Count Low - R/clr */
269 #define E1000_GORCH     0x0408C  /* Good Octets Rx Count High - R/clr */
270 #define E1000_GOTCL     0x04090  /* Good Octets Tx Count Low - R/clr */
271 #define E1000_GOTCH     0x04094  /* Good Octets Tx Count High - R/clr */
272 #define E1000_RNBC      0x040A0  /* Rx No Buffers Count - R/clr */
273 #define E1000_RUC       0x040A4  /* Rx Undersize Count - R/clr */
274 #define E1000_RFC       0x040A8  /* Rx Fragment Count - R/clr */
275 #define E1000_ROC       0x040AC  /* Rx Oversize Count - R/clr */
276 #define E1000_RJC       0x040B0  /* Rx Jabber Count - R/clr */
277 #define E1000_MGTPRC    0x040B4  /* Management Packets Rx Count - R/clr */
278 #define E1000_MGTPDC    0x040B8  /* Management Packets Dropped Count - R/clr */
279 #define E1000_MGTPTC    0x040BC  /* Management Packets Tx Count - R/clr */
280 #define E1000_TORL      0x040C0  /* Total Octets Rx Low - R/clr */
281 #define E1000_TORH      0x040C4  /* Total Octets Rx High - R/clr */
282 #define E1000_TOTL      0x040C8  /* Total Octets Tx Low - R/clr */
283 #define E1000_TOTH      0x040CC  /* Total Octets Tx High - R/clr */
284 #define E1000_TPR       0x040D0  /* Total Packets Rx - R/clr */
285 #define E1000_TPT       0x040D4  /* Total Packets Tx - R/clr */
286 #define E1000_PTC64     0x040D8  /* Packets Tx (64 bytes) - R/clr */
287 #define E1000_PTC127    0x040DC  /* Packets Tx (65-127 bytes) - R/clr */
288 #define E1000_PTC255    0x040E0  /* Packets Tx (128-255 bytes) - R/clr */
289 #define E1000_PTC511    0x040E4  /* Packets Tx (256-511 bytes) - R/clr */
290 #define E1000_PTC1023   0x040E8  /* Packets Tx (512-1023 bytes) - R/clr */
291 #define E1000_PTC1522   0x040EC  /* Packets Tx (1024-1522 Bytes) - R/clr */
292 #define E1000_MPTC      0x040F0  /* Multicast Packets Tx Count - R/clr */
293 #define E1000_BPTC      0x040F4  /* Broadcast Packets Tx Count - R/clr */
294 #define E1000_TSCTC     0x040F8  /* TCP Segmentation Context Tx - R/clr */
295 #define E1000_TSCTFC    0x040FC  /* TCP Segmentation Context Tx Fail - R/clr */
296 #define E1000_IAC       0x04100  /* Interrupt Assertion Count */
297 #define E1000_ICRXPTC   0x04104  /* Interrupt Cause Rx Pkt Timer Expire Count */
298 #define E1000_ICRXATC   0x04108  /* Interrupt Cause Rx Abs Timer Expire Count */
299 #define E1000_ICTXPTC   0x0410C  /* Interrupt Cause Tx Pkt Timer Expire Count */
300 #define E1000_ICTXATC   0x04110  /* Interrupt Cause Tx Abs Timer Expire Count */
301 #define E1000_ICTXQEC   0x04118  /* Interrupt Cause Tx Queue Empty Count */
302 #define E1000_ICTXQMTC  0x0411C  /* Interrupt Cause Tx Queue Min Thresh Count */
303 #define E1000_ICRXDMTC  0x04120  /* Interrupt Cause Rx Desc Min Thresh Count */
304 #define E1000_ICRXOC    0x04124  /* Interrupt Cause Receiver Overrun Count */

306 /* LinkSec Tx Untagged Packet Count - OutPktsUntagged */
307 #define E1000_LSECTXUT          0x04300
308 /* LinkSec Encrypted Tx Packets Count - OutPktsEncrypted */
309 #define E1000_LSECTXPKTE        0x04304
310 /* LinkSec Protected Tx Packet Count - OutPktsProtected */
311 #define E1000_LSECTXPKTP        0x04308
312 /* LinkSec Encrypted Tx Octets Count - OutOctetsEncrypted */
313 #define E1000_LSECTXOCTE        0x0430C
314 /* LinkSec Protected Tx Octets Count - OutOctetsProtected */
315 #define E1000_LSECTXOCTP        0x04310
316 /* LinkSec Untagged non-Strict Rx Packet Count - InPktsUntagged/InPktsNoTag */
317 #define E1000_LSECRXUT          0x04314
318 /* LinkSec Rx Octets Decrypted Count - InOctetsDecrypted */
319 #define E1000_LSECRXOCTD        0x0431C
320 /* LinkSec Rx Octets Validated - InOctetsValidated */
321 #define E1000_LSECRXOCTV        0x04320
322 /* LinkSec Rx Bad Tag - InPktsBadTag */
323 #define E1000_LSECRXBAD         0x04324
```



```
324 /* LinkSec Rx Packet No SCI Count - InPktsNoSci */
325 #define E1000_LSECRXNOSCI       0x04328
326 /* LinkSec Rx Packet Unknown SCI Count - InPktsUnknownSci */
327 #define E1000_LSECRXUNSCI       0x0432C
328 /* LinkSec Rx Unchecked Packets Count - InPktsUnchecked */
329 #define E1000_LSECRXUNCH        0x04330
330 /* LinkSec Rx Delayed Packet Count - InPktsDelayed */
331 #define E1000_LSECRXDELAY       0x04340
332 /* LinkSec Rx Late Packets Count - InPktsLate */
333 #define E1000_LSECRXLATE        0x04350
334 /* LinkSec Rx Packet OK Count - InPktsOk */
335 #define E1000_LSECRXOK(_n)      (0x04360 + (0x04 * (_n)))
336 /* LinkSec Rx Invalid Count - InPktsInvalid */
337 #define E1000_LSECRXINV(_n)     (0x04380 + (0x04 * (_n)))
338 /* LinkSec Rx Not Valid Count - InPktsNotValid */
339 #define E1000_LSECRXNV(_n)      (0x043A0 + (0x04 * (_n)))
340 /* LinkSec Rx Unused SA Count - InPktsUnusedSa */
341 #define E1000_LSECRXUNSA        0x043C0
342 /* LinkSec Rx Not Using SA Count - InPktsNotUsingSa */
343 #define E1000_LSECRXNUSA        0x043D0
344 /* LinkSec Tx Capabilities Register - RO */
345 #define E1000_LSECTXCAP         0x0B000
346 /* LinkSec Rx Capabilities Register - RO */
347 #define E1000_LSECRXCAP         0x0B300
348 #define E1000_LSECTXCTRL        0x0B004  /* LinkSec Tx Control - RW */
349 #define E1000_LSECRXCTRL        0x0B304  /* LinkSec Rx Control - RW */
350 #define E1000_LSECTXSCL         0x0B008  /* LinkSec Tx SCI Low - RW */
351 #define E1000_LSECTXSCH         0x0B00C  /* LinkSec Tx SCI High - RW */
352 #define E1000_LSECTXSA          0x0B010  /* LinkSec Tx SA0 - RW */
353 #define E1000_LSECTXPN0         0x0B018  /* LinkSec Tx SA PN 0 - RW */
354 #define E1000_LSECTXPN1         0x0B01C  /* LinkSec Tx SA PN 1 - RW */
355 #define E1000_LSECRXSCL         0x0B3D0  /* LinkSec Rx SCI Low - RW */
356 #define E1000_LSECRXSCH         0x0B3E0  /* LinkSec Rx SCI High - RW */
357 /* LinkSec Tx 128-bit Key 0 - WO */
358 #define E1000_LSECTXKEY0(_n)    (0x0B020 + (0x04 * (_n)))
359 /* LinkSec Tx 128-bit Key 1 - WO */
360 #define E1000_LSECTXKEY1(_n)    (0x0B030 + (0x04 * (_n)))
361 /* LinkSec Rx SAs - RW */
362 #define E1000_LSECRXSA(_n)      (0x0B310 + (0x04 * (_n)))
363 /* LinkSec Rx SAs - RW */
364 #define E1000_LSECRXPN(_n)      (0x0B330 + (0x04 * (_n)))
365 /*
366  * LinkSec Rx Keys  - where _n is the SA no. and _m the 4 dwords of the 128 bit
367  * key - RW.
368  */
369 #define E1000_LSECRXKEY(_n, _m) (0x0B350 + (0x10 * (_n)) + (0x04 * (_m)))

371 #define E1000_SSVPC     0x041A0  /* Switch Security Violation Packet Count */
372 #define E1000_IPSCTRL   0xB430   /* IpSec Control Register */
373 #define E1000_IPSRXCMD  0x0B408  /* IPSec Rx Command Register - RW */
374 #define E1000_IPSRXIDX  0x0B400  /* IPSec Rx Index - RW */
375 /* IPSec Rx IPv4/v6 Address - RW */
376 #define E1000_IPSRXIPADDR(_n)   (0x0B420+ (0x04 * (_n)))
377 /* IPSec Rx 128-bit Key - RW */
378 #define E1000_IPSRXKEY(_n)      (0x0B410 + (0x04 * (_n)))
379 #define E1000_IPSRXSALT 0x0B404  /* IPSec Rx Salt - RW */
380 #define E1000_IPSRXSPI  0x0B40C  /* IPSec Rx SPI - RW */
381 /* IPSec Tx 128-bit Key - RW */
382 #define E1000_IPSTXKEY(_n)      (0x0B460 + (0x04 * (_n)))
383 #define E1000_IPSTXSALT 0x0B454  /* IPSec Tx Salt - RW */
384 #define E1000_IPSTXIDX  0x0B450  /* IPSec Tx SA IDX - RW */
385 #define E1000_PCS_CFG0  0x04200  /* PCS Configuration 0 - RW */
386 #define E1000_PCS_LCTL  0x04208  /* PCS Link Control - RW */
387 #define E1000_PCS_LSTAT 0x0420C  /* PCS Link Status - RO */
388 #define E1000_CBTMPC    0x0402C  /* Circuit Breaker Tx Packet Count */
389 #define E1000_HTDPMC    0x0403C  /* Host Transmit Discarded Packets */
```

```
#define E1000_CBRDPC    0x04044  /* Circuit Breaker Rx Dropped Count */
#define E1000_CBRMPC    0x040FC  /* Circuit Breaker Rx Packet Count */
#define E1000_RPTHC     0x04104  /* Rx Packets To Host */
#define E1000_HGPTC     0x04118  /* Host Good Packets Tx Count */
#define E1000_HTCBDPC   0x04124  /* Host Tx Circuit Breaker Dropped Count */
#define E1000_HGORCL    0x04128  /* Host Good Octets Received Count Low */
#define E1000_HGORCH    0x0412C  /* Host Good Octets Received Count High */
#define E1000_HGOTCL    0x04130  /* Host Good Octets Transmit Count Low */
#define E1000_HGOTCH    0x04134  /* Host Good Octets Transmit Count High */
#define E1000_LENERRS   0x04138  /* Length Errors Count */
#define E1000_SCVPC     0x04228  /* SerDes/SGMII Code Violation Pkt Count */
#define E1000_HRMPC     0x0A018  /* Header Redirection Missed Packet Count */
#define E1000_PCS_ANADV 0x04218  /* AN advertisement - RW */
#define E1000_PCS_LPAB  0x0421C  /* Link Partner Ability - RW */
#define E1000_PCS_NPTX  0x04220  /* AN Next Page Transmit - RW */
#define E1000_PCS_LPABNP 0x04224 /* Link Partner Ability Next Page - RW */
#define E1000_1GSTAT_RCV 0x04228 /* 1GSTAT Code Violation Packet Count - RW */
#define E1000_RXCSUM    0x05000  /* Rx Checksum Control - RW */
#define E1000_RLPML     0x05004  /* Rx Long Packet Max Length */
#define E1000_RFCTL     0x05008  /* Receive Filter Control */
#define E1000_MTA       0x05200  /* Multicast Table Array - RW Array */
#define E1000_RA        0x05400  /* Receive Address - RW Array */
/* 2nd half of receive address array - RW Array */
#define E1000_RA2       0x054E0
#define E1000_VFTA      0x05600  /* VLAN Filter Table Array - RW Array */
#define E1000_VT_CTL    0x0581C  /* VMDq Control - RW */
#define E1000_VFQA0     0x0B000  /* VLAN Filter Queue Array 0 - RW Array */
#define E1000_VFQA1     0x0B200  /* VLAN Filter Queue Array 1 - RW Array */
#define E1000_WUC       0x05800  /* Wakeup Control - RW */
#define E1000_WUFC      0x05808  /* Wakeup Filter Control - RW */
#define E1000_WUS       0x05810  /* Wakeup Status - RO */
#define E1000_MANC      0x05820  /* Management Control - RW */
#define E1000_IPAV      0x05838  /* IP Address Valid - RW */
#define E1000_IP4AT     0x05840  /* IPv4 Address Table - RW Array */
#define E1000_IP6AT     0x05880  /* IPv6 Address Table - RW Array */
#define E1000_WUPL      0x05900  /* Wakeup Packet Length - RW */
#define E1000_WUPM      0x05A00  /* Wakeup Packet Memory - RO A */
#define E1000_PBACL     0x05B68  /* MSIx PBA Clear - Read/Write 1's to clear */
#define E1000_FFLT      0x05F00  /* Flexible Filter Length Table - RW Array */
#define E1000_HOST_IF   0x08800  /* Host Interface */
#define E1000_FFMT      0x09000  /* Flexible Filter Mask Table - RW Array */
#define E1000_FFVT      0x09800  /* Flexible Filter Value Table - RW Array */
/* Flexible Host Filter Table */
#define E1000_FHFT(_n)  (0x09000 + (_n * 0x100))
/* Ext Flexible Host Filter Table */
#define E1000_FHFT_EXT(_n) (0x09A00 + (_n * 0x100))

#define E1000_KMRNCTRLSTA       0x00034 /* MAC-PHY interface - RW */
#define E1000_MDPHYA            0x0003C /* PHY address - RW */
#define E1000_MANC2H            0x05860 /* Management Control To Host - RW */
/* Software-Firmware Synchronization - RW */
#define E1000_SW_FW_SYNC        0x05B5C
#define E1000_CCMCTL            0x05B48 /* CCM Control Register */
#define E1000_GIOCTL            0x05B44 /* GIO Analog Control Register */
#define E1000_SCCTL             0x05B4C /* PCIc PLL Configuration Register */
#define E1000_GCR               0x05B00 /* PCI-Ex Control */
#define E1000_GCR2              0x05B64 /* PCI-Ex Control #2 */
#define E1000_GSCL_1            0x05B10 /* PCI-Ex Statistic Control #1 */
#define E1000_GSCL_2            0x05B14 /* PCI-Ex Statistic Control #2 */
#define E1000_GSCL_3            0x05B18 /* PCI-Ex Statistic Control #3 */
#define E1000_GSCL_4            0x05B1C /* PCI-Ex Statistic Control #4 */
/* Function Active and Power State to MNG */
#define E1000_FACTPS            0x05B30
#define E1000_SWSM              0x05B50 /* SW Semaphore */
#define E1000_FWSM              0x05B54 /* FW Semaphore */
/* Driver-only SW semaphore (not used by BOOT agents) */
```



```
#define E1000_SWSM2             0x05B58
#define E1000_DCA_ID            0x05B70 /* DCA Requester ID Information - RO */
#define E1000_DCA_CTRL          0x05B74 /* DCA Control - RW */
#define E1000_UFUSE             0x05B78 /* UFUSE - RO */
#define E1000_FFLT_DBG          0x05F04 /* Debug Register */
#define E1000_HICR              0x08F00 /* Host Interface Control */

/* RSS registers */
#define E1000_CPUVEC            0x02C10 /* CPU Vector Register - RW */
#define E1000_MRQC              0x05818 /* Multiple Receive Control - RW */
#define E1000_IMIR(_i)  (0x05A80 + ((_i) * 4))  /* Immediate Interrupt */
/* Immediate Interrupt Ext */
#define E1000_IMIREXT(_i)       (0x05AA0 + ((_i) * 4))
#define E1000_IMIRVP    0x05AC0 /* Immediate Interrupt Rx VLAN Priority - RW */
/* MSI-X Allocation Register (_i) - RW */
#define E1000_MSIXBM(_i)        (0x01600 + ((_i) * 4))
/* MSI-X Table entry addr low reg 0 - RW */
#define E1000_MSIXTADD(_i)      (0x0C000 + ((_i) * 0x10))
/* MSI-X Table entry addr upper reg 0 - RW */
#define E1000_MSIXTUADD(_i)     (0x0C004 + ((_i) * 0x10))
/* MSI-X Table entry message reg 0 - RW */
#define E1000_MSIXTMSG(_i)      (0x0C008 + ((_i) * 0x10))
/* MSI-X Table entry vector ctrl reg 0 - RW */
#define E1000_MSIXVCTRL(_i)     (0x0C00C + ((_i) * 0x10))
#define E1000_MSIXPBA           0x0E000 /* MSI-X Pending bit array */
/* Redirection Table - RW Array */
#define E1000_RETA(_i)          (0x05C00 + ((_i) * 4))
/* RSS Random Key - RW Array */
#define E1000_RSSRK(_i)         (0x05C80 + ((_i) * 4))
#define E1000_RSSIM     0x05864 /* RSS Interrupt Mask */
#define E1000_RSSIR     0x05868 /* RSS Interrupt Request */
/* VT Registers */
#define E1000_SWPBS     0x03004 /* Switch Packet Buffer Size - RW */
#define E1000_MBVFICR   0x00C80 /* Mailbox VF Cause - RWC */
#define E1000_MBVFIMR   0x00C84 /* Mailbox VF int Mask - RW */
#define E1000_VFLRE     0x00C88 /* VF Register Events - RWC */
#define E1000_VFRE      0x00C8C /* VF Receive Enables */
#define E1000_VFTE      0x00C90 /* VF Transmit Enables */
#define E1000_QDE       0x02408 /* Queue Drop Enable - RW */
#define E1000_DTXSWC    0x03500 /* DMA Tx Switch Control - RW */
#define E1000_RPLOLR    0x05AF0 /* Replication Offload - RW */
#define E1000_UTA       0x0A000 /* Unicast Table Array - RW */
#define E1000_IOVTCL    0x05BBC /* IOV Control Register */
#define E1000_VMRCTL    0X05D80 /* Virtual Mirror Rule Control */
/* These act per VF so an array friendly macro is used */
#define E1000_V2PMAILBOX(_n)    (0x00C40 + (4 * (_n)))
#define E1000_P2VMAILBOX(_n)    (0x00C00 + (4 * (_n)))
#define E1000_VMBMEM(_n)        (0x00800 + (64 * (_n)))
#define E1000_VFVMBMEM(_n)      (0x00800 + (_n))
#define E1000_VMOLR(_n)         (0x05AD0 + (4 * (_n)))
/* VLAN Virtual Machine Filter - RW */
#define E1000_VLVF(_n)          (0x05D00 + (4 * (_n)))
#define E1000_VMVIR(_n)         (0x03700 + (4 * (_n)))

/* Filtering Registers */
#define E1000_SAQF(_n)  (0x05980 + (4 * (_n))) /* Source Address Queue Fltr */
#define E1000_DAQF(_n)  (0x059A0 + (4 * (_n))) /* Dest Address Queue Fltr */
#define E1000_SPQF(_n)  (0x059C0 + (4 * (_n))) /* Source Port Queue Fltr */
#define E1000_FTQF(_n)  (0x059E0 + (4 * (_n))) /* 5-tuple Queue Fltr */
#define E1000_TTQF(_n)  (0x059E0 + (4 * (_n))) /* 2-tuple Queue Fltr */
#define E1000_SYNQF(_n) (0x055FC + (4 * (_n))) /* SYN Packet Queue Fltr */
#define E1000_ETQF(_n)  (0x05CB0 + (4 * (_n))) /* EType Queue Fltr */

#define E1000_RTTDCS    0x3600 /* Reedtown Tx Desc plane control and status */
#define E1000_RTTPCS    0x3474 /* Reedtown Tx Packet Plane control and status */
#define E1000_RTRPCS    0x2474 /* Rx packet plane control and status */
```

```
522 #define E1000_RTRUP2TC  0x05AC4 /* Rx User Priority to Traffic Class */
523 #define E1000_RTTUP2TC  0x0418  /* Transmit User Priority to Traffic Class */
524 /* Tx Desc plane TC Rate-scheduler config */
525 #define E1000_RTTDTCRC(_n)      (0x3610 + ((_n) * 4))
526 /* Tx Packet plane TC Rate-Scheduler Config */
527 #define E1000_RTTPTCRC(_n)      (0x3480 + ((_n) * 4))
528 /* Rx Packet plane TC Rate-Scheduler Config */
529 #define E1000_RTRPTCRC(_n)      (0x2480 + ((_n) * 4))
530 /* Tx Desc Plane TC Rate-Scheduler Status */
531 #define E1000_RTTDTCRS(_n)      (0x3630 + ((_n) * 4))
532 /* Tx Desc Plane TC Rate-Scheduler MMW */
533 #define E1000_RTTDTCRM(_n)      (0x3650 + ((_n) * 4))
534 /* Tx Packet plane TC Rate-Scheduler Status */
535 #define E1000_RTTPTCRS(_n)      (0x34A0 + ((_n) * 4))
536 /* Tx Packet plane TC Rate-scheduler MMW */
537 #define E1000_RTTPTCRM(_n)      (0x34C0 + ((_n) * 4))
538 /* Rx Packet plane TC Rate-Scheduler Status */
539 #define E1000_RTRPTCRS(_n)      (0x24A0 + ((_n) * 4))
540 /* Rx Packet plane TC Rate-Scheduler MMW */
541 #define E1000_RTRPTCRM(_n)      (0x24C0 + ((_n) * 4))
542 /* Tx Desc plane VM Rate-Scheduler MMW */
543 #define E1000_RTTDVMRM(_n)      (0x3670 + ((_n) * 4))
544 /* Tx BCN Rate-Scheduler MMW */
545 #define E1000_RTTBCNRM(_n)      (0x3690 + ((_n) * 4))
546 #define E1000_RTTDQSEL  0x3604  /* Tx Desc Plane Queue Select */
547 #define E1000_RTTDVMRC  0x3608  /* Tx Desc Plane VM Rate-Scheduler Config */
548 #define E1000_RTTDVMRS  0x360C  /* Tx Desc Plane VM Rate-Scheduler Status */
549 #define E1000_RTTBCNRC  0x36B0  /* Tx BCN Rate-Scheduler Config */
550 #define E1000_RTTBCNRS  0x36B4  /* Tx BCN Rate-Scheduler Status */
551 #define E1000_RTTBCNCR  0xB200  /* Tx BCN Control Register */
552 #define E1000_RTTBCNTG  0x35A4  /* Tx BCN Tagging */
553 #define E1000_RTTBCNCP  0xB208  /* Tx BCN Congestion point */
554 #define E1000_RTRBCNCR  0xB20C  /* Rx BCN Control Register */
555 #define E1000_RTTBCNRD  0x36B8  /* Tx BCN Rate Drift */
556 #define E1000_PFCTOP    0x1080  /* Priority Flow Control Type and Opcode */
557 #define E1000_RTTBCNIDX 0xB204  /* Tx BCN Congestion Point */
558 #define E1000_RTTBCNACH 0x0B214 /* Tx BCN Control High */
559 #define E1000_RTTBCNACL 0x0B210 /* Tx BCN Control Low */

561 /* DMA Coalescing registers */
562 #define E1000_DMACR     0x02508 /* Control Register */
563 #define E1000_DMCTXTH   0x03550 /* Transmit Threshold */
564 #define E1000_DMCTLX    0x02514 /* Time to Lx Request */
565 #define E1000_DMCRTRH   0x05DD0 /* Receive Packet Rate Threshold */
566 #define E1000_DMCCNT    0x05DD4 /* Current RX Count */
567 #define E1000_FCRTC     0x02170 /* Flow Control Rx high watermark */
568 #define E1000_PCIEMISC  0x05BB8 /* PCIE misc config register */

570 /* PCIe Parity Status Register */
571 #define E1000_PCIEERRSTS        0x05BA8

573 /* Energy Efficient Ethernet "EEE" registers */
574 #define E1000_IPCNFG    0x0E38 /* Internal PHY Configuration */
575 #define E1000_LTRC      0x01A0 /* Latency Tolerance Reporting Control */
576 #define E1000_EEER      0x0E30 /* Energy Efficient Ethernet "EEE" */
577 #define E1000_EEE_SU    0x0E34 /* EEE Setup */
578 #define E1000_TLPIC     0x4148 /* EEE Tx LPI Count - TLPIC */
579 #define E1000_RLPIC     0x414C /* EEE Rx LPI Count - RLPIC */

581 #ifdef __cplusplus
582 }
```

*unchanged_portion_omitted_*