```
**********************************************************
   85638 Sun Jul  9 20:14:13 2017
new/usr/src/uts/common/io/rwn/rt2860.c
8465 aggressive-loop-optimizations error in rt2860.c
**********************************************************
```

```
   1 /*
   2  * Copyright 2017 Gary Mills
   3  * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
   4  * Use is subject to license terms.
   5  */

   7 /*
   8  * Copyright (c) 2007, 2008
   9  *      Damien Bergamini <damien.bergamini@free.fr>
  10  *
  11  * Permission to use, copy, modify, and distribute this software for any
  12  * purpose with or without fee is hereby granted, provided that the above
  13  * copyright notice and this permission notice appear in all copies.
  14  *
  15  * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
  16  * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
  17  * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
  18  * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
  19  * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
  20  * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
  21  * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
  22  */

  24 /*
  25  * Ralink Technology RT2860 chipset driver
  26  * http://www.ralinktech.com/
  27  */

  29 #include <sys/types.h>
  30 #include <sys/byteorder.h>
  31 #include <sys/conf.h>
  32 #include <sys/cmn_err.h>
  33 #include <sys/stat.h>
  34 #include <sys/ddi.h>
  35 #include <sys/sunddi.h>
  36 #include <sys/strsubr.h>
  37 #include <inet/common.h>
  38 #include <sys/note.h>
  39 #include <sys/stream.h>
  40 #include <sys/strsun.h>
  41 #include <sys/modctl.h>
  42 #include <sys/devops.h>
  43 #include <sys/mac_provider.h>
  44 #include <sys/mac_wifi.h>
  45 #include <sys/net80211.h>
  46 #include <sys/net80211_proto.h>
  47 #include <sys/varargs.h>
  48 #include <sys/pci.h>
  49 #include <sys/crypto/common.h>
  50 #include <sys/crypto/api.h>
  51 #include <inet/wifi_ioctl.h>

  53 #include "rt2860_reg.h"
  54 #include "rt2860_var.h"

  56 #define RT2860_DBG_80211        (1 << 0)
  57 #define RT2860_DBG_DMA          (1 << 1)
  58 #define RT2860_DBG_EEPROM       (1 << 2)
  59 #define RT2860_DBG_FW           (1 << 3)
  60 #define RT2860_DBG_HW           (1 << 4)
  61 #define RT2860_DBG_INTR         (1 << 5)
```



```
  62 #define RT2860_DBG_RX           (1 << 6)
  63 #define RT2860_DBG_SCAN         (1 << 7)
  64 #define RT2860_DBG_TX           (1 << 8)
  65 #define RT2860_DBG_RADIO        (1 << 9)
  66 #define RT2860_DBG_RESUME       (1 << 10)
  67 #define RT2860_DBG_MSG          (1 << 11)

  69 uint32_t rt2860_dbg_flags = 0x0;

  71 #ifdef DEBUG
  72 #define RWN_DEBUG \
  73         rt2860_debug
  74 #else
  75 #define RWN_DEBUG
  76 #endif

  78 static void *rt2860_soft_state_p = NULL;
  79 static uint8_t rt2860_fw_bin [] = {
  80 #include "fw-rt2860/rt2860.ucode"
  81 };
______unchanged_portion_omitted_

 379 static int
 380 rt2860_read_eeprom(struct rt2860_softc *sc)
 381 {
 382         struct ieee80211com     *ic = &sc->sc_ic;
 383         int                     ridx, ant, i;
 384         int8_t                  delta_2ghz, delta_5ghz;
 385         uint16_t                val;

 387         /* read EEPROM version */
 388         val = rt2860_eeprom_read(sc, RT2860_EEPROM_VERSION);
 389         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
 390             "EEPROM rev=%d, FAE=%d\n",
 391             val & 0xff, val >> 8);

 393         /* read MAC address */
 394         val = rt2860_eeprom_read(sc, RT2860_EEPROM_MAC01);
 395         ic->ic_macaddr[0] = val & 0xff;
 396         ic->ic_macaddr[1] = val >> 8;
 397         val = rt2860_eeprom_read(sc, RT2860_EEPROM_MAC23);
 398         ic->ic_macaddr[2] = val & 0xff;
 399         ic->ic_macaddr[3] = val >> 8;
 400         val = rt2860_eeprom_read(sc, RT2860_EEPROM_MAC45);
 401         ic->ic_macaddr[4] = val & 0xff;
 402         ic->ic_macaddr[5] = val >> 8;
 403         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
 404             "MAC address is: %x:%x:%x:%x:%x:%x\n",
 405             ic->ic_macaddr[0], ic->ic_macaddr[1],
 406             ic->ic_macaddr[2], ic->ic_macaddr[3],
 407             ic->ic_macaddr[4], ic->ic_macaddr[5]);

 409         /* read country code */
 410         val = rt2860_eeprom_read(sc, RT2860_EEPROM_COUNTRY);
 411         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
 412             "EEPROM region code=0x%04x\n", val);

 414         /* read default BBP settings */
 415         for (i = 0; i < 8; i++) {
 416                 val = rt2860_eeprom_read(sc, RT2860_EEPROM_BBP_BASE + i);
 417                 sc->bbp[i].val = val & 0xff;
 418                 sc->bbp[i].reg = val >> 8;
 419                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
 420                     "BBP%d=0x%02x\n",
 421                     sc->bbp[i].reg, sc->bbp[i].val);
 422         }
```

```
424         /* read RF frequency offset from EEPROM */
425         val = rt2860_eeprom_read(sc, RT2860_EEPROM_FREQ_LEDS);
426         sc->freq = ((val & 0xff) != 0xff) ? val & 0xff : 0;
427         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
428             "EEPROM freq offset %d\n", sc->freq & 0xff);

430         if ((sc->leds = val >> 8) != 0xff) {
431                 /* read LEDs operating mode */
432                 sc->led[0] = rt2860_eeprom_read(sc, RT2860_EEPROM_LED1);
433                 sc->led[1] = rt2860_eeprom_read(sc, RT2860_EEPROM_LED2);
434                 sc->led[2] = rt2860_eeprom_read(sc, RT2860_EEPROM_LED3);
435         } else {
436                 /* broken EEPROM, use default settings */
437                 sc->leds = 0x01;
438                 sc->led[0] = 0x5555;
439                 sc->led[1] = 0x2221;
440                 sc->led[2] = 0xa9f8;
441         }
442         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
443             "EEPROM LED mode=0x%02x, LEDs=0x%04x/0x%04x/0x%04x\n",
444             sc->leds, sc->led[0], sc->led[1], sc->led[2]);

446         /* read RF information */
447         val = rt2860_eeprom_read(sc, RT2860_EEPROM_ANTENNA);
448         if (val == 0xffff) {
449                 /* broken EEPROM, default to RF2820 1T2R */
450                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
451                     "invalid EEPROM antenna info, using default\n");
452                 sc->rf_rev = RT2860_RF_2820;
453                 sc->ntxchains = 1;
454                 sc->nrxchains = 2;
455         } else {
456                 sc->rf_rev = (val >> 8) & 0xf;
457                 sc->ntxchains = (val >> 4) & 0xf;
458                 sc->nrxchains = val & 0xf;
459         }
460         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
461             "EEPROM RF rev=0x%02x chains=%dT%dR\n",
462             sc->rf_rev, sc->ntxchains, sc->nrxchains);

464         /* check if RF supports automatic Tx access gain control */
465         val = rt2860_eeprom_read(sc, RT2860_EEPROM_CONFIG);
466         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
467             "EEPROM CFG 0x%04x\n", val);
468         if ((val & 0xff) != 0xff)
469                 sc->calib_2ghz = sc->calib_5ghz = 0; /* XXX (val >> 1) & 1 */;

471         if (sc->sc_flags & RT2860_ADVANCED_PS) {
472                 /* read PCIe power save level */
473                 val = rt2860_eeprom_read(sc, RT2860_EEPROM_PCIE_PSLEVEL);
474                 if ((val & 0xff) != 0xff) {
475                         sc->pslevel = val & 0x3;
476                         val = rt2860_eeprom_read(sc, RT2860_EEPROM_REV);
477                         if (val >> 8 != 0x92 || !(val & 0x80))
478                                 sc->pslevel = MIN(sc->pslevel, 1);
479                         RWN_DEBUG(RT2860_DBG_EEPROM,
480                             "rwn: rt2860_read_eeprom(): "
481                             "EEPROM PCIe PS Level=%d\n",
482                             sc->pslevel);
483                 }
484         }
485         /* read power settings for 2GHz channels */
486         for (i = 0; i < 14; i += 2) {
487                 val = rt2860_eeprom_read(sc,
488                     RT2860_EEPROM_PWR2GHZ_BASE1 + i / 2);
```



```
489                 sc->txpow1[i + 0] = (int8_t)(val & 0xff);
490                 sc->txpow1[i + 1] = (int8_t)(val >> 8);

492                 val = rt2860_eeprom_read(sc,
493                     RT2860_EEPROM_PWR2GHZ_BASE2 + i / 2);
494                 sc->txpow2[i + 0] = (int8_t)(val & 0xff);
495                 sc->txpow2[i + 1] = (int8_t)(val >> 8);
496         }
497         /* fix broken Tx power entries */
498         for (i = 0; i < 14; i++) {
499                 if (sc->txpow1[i] < 0 || sc->txpow1[i] > 31)
500                         sc->txpow1[i] = 5;
501                 if (sc->txpow2[i] < 0 || sc->txpow2[i] > 31)
502                         sc->txpow2[i] = 5;
503                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
504                     "chan %d: power1=%d, power2=%d\n",
505                     rt2860_rf2850[i].chan, sc->txpow1[i], sc->txpow2[i]);
506         }
507         /* read power settings for 5GHz channels */
508         for (i = 0; i < 36; i += 2) {
509                 val = rt2860_eeprom_read(sc,
510                     RT2860_EEPROM_PWR5GHZ_BASE1 + i / 2);
511                 sc->txpow1[i + 14] = (int8_t)(val & 0xff);
512                 sc->txpow1[i + 15] = (int8_t)(val >> 8);

514                 val = rt2860_eeprom_read(sc,
515                     RT2860_EEPROM_PWR5GHZ_BASE2 + i / 2);
516                 sc->txpow2[i + 14] = (int8_t)(val & 0xff);
517                 sc->txpow2[i + 15] = (int8_t)(val >> 8);
518         }
519         /* fix broken Tx power entries */
520         for (i = 0; i < 35; i++) {
519         for (i = 0; i < 36; i++) {
521                 if (sc->txpow1[14 + i] < -7 || sc->txpow1[14 + i] > 15)
522                         sc->txpow1[14 + i] = 5;
523                 if (sc->txpow2[14 + i] < -7 || sc->txpow2[14 + i] > 15)
524                         sc->txpow2[14 + i] = 5;
525                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
526                     "chan %d: power1=%d, power2=%d\n",
527                     rt2860_rf2850[14 + i].chan, sc->txpow1[14 + i],
528                     sc->txpow2[14 + i]);
529         }

531         /* read Tx power compensation for each Tx rate */
532         val = rt2860_eeprom_read(sc, RT2860_EEPROM_DELTAPWR);
533         delta_2ghz = delta_5ghz = 0;
534         if ((val & 0xff) != 0xff && (val & 0x80)) {
535                 delta_2ghz = val & 0xf;
536                 if (!(val & 0x40))      /* negative number */
537                         delta_2ghz = -delta_2ghz;
538         }
539         val >>= 8;
540         if ((val & 0xff) != 0xff && (val & 0x80)) {
541                 delta_5ghz = val & 0xf;
542                 if (!(val & 0x40))      /* negative number */
543                         delta_5ghz = -delta_5ghz;
544         }
545         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
546             "power compensation=%d (2GHz), %d (5GHz) \n",
547             delta_2ghz, delta_5ghz);

549         for (ridx = 0; ridx < 5; ridx++) {
550                 uint32_t        reg;

552                 val = rt2860_eeprom_read(sc, RT2860_EEPROM_RPWR + ridx);
553                 reg = (uint32_t)val << 16;
```

```
554                 val = rt2860_eeprom_read(sc, RT2860_EEPROM_RPWR + ridx + 1);
555                 reg |= val;

557                 sc->txpow20mhz[ridx] = reg;
558                 sc->txpow40mhz_2ghz[ridx] = b4inc(reg, delta_2ghz);
559                 sc->txpow40mhz_5ghz[ridx] = b4inc(reg, delta_5ghz);

561                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
562                     "ridx %d: power 20MHz=0x%08x, 40MHz/2GHz=0x%08x, "
563                     "40MHz/5GHz=0x%08x\n", ridx, sc->txpow20mhz[ridx],
564                     sc->txpow40mhz_2ghz[ridx], sc->txpow40mhz_5ghz[ridx]);
565         }

567         /* read factory-calibrated samples for temperature compensation */
568         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI1_2GHZ);
569         sc->tssi_2ghz[0] = val & 0xff;  /* [-4] */
570         sc->tssi_2ghz[1] = val >> 8;    /* [-3] */
571         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI2_2GHZ);
572         sc->tssi_2ghz[2] = val & 0xff;  /* [-2] */
573         sc->tssi_2ghz[3] = val >> 8;    /* [-1] */
574         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI3_2GHZ);
575         sc->tssi_2ghz[4] = val & 0xff;  /* [+0] */
576         sc->tssi_2ghz[5] = val >> 8;    /* [+1] */
577         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI4_2GHZ);
578         sc->tssi_2ghz[6] = val & 0xff;  /* [+2] */
579         sc->tssi_2ghz[7] = val >> 8;    /* [+3] */
580         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI5_2GHZ);
581         sc->tssi_2ghz[8] = val & 0xff;  /* [+4] */
582         sc->step_2ghz = val >> 8;
583         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
584             "TSSI 2GHz: 0x%02x 0x%02x 0x%02x 0x%02x 0x%02x 0x%02x 0x%02x "
585             "0x%02x 0x%02x step=%d\n", sc->tssi_2ghz[0], sc->tssi_2ghz[1],
586             sc->tssi_2ghz[2], sc->tssi_2ghz[3], sc->tssi_2ghz[4],
587             sc->tssi_2ghz[5], sc->tssi_2ghz[6], sc->tssi_2ghz[7],
588             sc->tssi_2ghz[8], sc->step_2ghz);
589         /* check that ref value is correct, otherwise disable calibration */
590         if (sc->tssi_2ghz[4] == 0xff)
591                 sc->calib_2ghz = 0;

593         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI1_5GHZ);
594         sc->tssi_5ghz[0] = val & 0xff;  /* [-4] */
595         sc->tssi_5ghz[1] = val >> 8;    /* [-3] */
596         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI2_5GHZ);
597         sc->tssi_5ghz[2] = val & 0xff;  /* [-2] */
598         sc->tssi_5ghz[3] = val >> 8;    /* [-1] */
599         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI3_5GHZ);
600         sc->tssi_5ghz[4] = val & 0xff;  /* [+0] */
601         sc->tssi_5ghz[5] = val >> 8;    /* [+1] */
602         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI4_5GHZ);
603         sc->tssi_5ghz[6] = val & 0xff;  /* [+2] */
604         sc->tssi_5ghz[7] = val >> 8;    /* [+3] */
605         val = rt2860_eeprom_read(sc, RT2860_EEPROM_TSSI5_5GHZ);
606         sc->tssi_5ghz[8] = val & 0xff;  /* [+4] */
607         sc->step_5ghz = val >> 8;
608         RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
609             "TSSI 5GHz: 0x%02x 0x%02x 0x%02x 0x%02x 0x%02x 0x%02x 0x%02x "
610             "0x%02x 0x%02x step=%d\n", sc->tssi_5ghz[0], sc->tssi_5ghz[1],
611             sc->tssi_5ghz[2], sc->tssi_5ghz[3], sc->tssi_5ghz[4],
612             sc->tssi_5ghz[5], sc->tssi_5ghz[6], sc->tssi_5ghz[7],
613             sc->tssi_5ghz[8], sc->step_5ghz);
614         /* check that ref value is correct, otherwise disable calibration */
615         if (sc->tssi_5ghz[4] == 0xff)
616                 sc->calib_5ghz = 0;

618         /* read RSSI offsets and LNA gains from EEPROM */
619         val = rt2860_eeprom_read(sc, RT2860_EEPROM_RSSI1_2GHZ);
```



```
620         sc->rssi_2ghz[0] = val & 0xff;  /* Ant A */
621         sc->rssi_2ghz[1] = val >> 8;    /* Ant B */
622         val = rt2860_eeprom_read(sc, RT2860_EEPROM_RSSI2_2GHZ);
623         sc->rssi_2ghz[2] = val & 0xff;  /* Ant C */
624         sc->lna[2] = val >> 8;          /* channel group 2 */

626         val = rt2860_eeprom_read(sc, RT2860_EEPROM_RSSI1_5GHZ);
627         sc->rssi_5ghz[0] = val & 0xff;  /* Ant A */
628         sc->rssi_5ghz[1] = val >> 8;    /* Ant B */
629         val = rt2860_eeprom_read(sc, RT2860_EEPROM_RSSI2_5GHZ);
630         sc->rssi_5ghz[2] = val & 0xff;  /* Ant C */
631         sc->lna[3] = val >> 8;          /* channel group 3 */

633         val = rt2860_eeprom_read(sc, RT2860_EEPROM_LNA);
634         sc->lna[0] = val & 0xff;        /* channel group 0 */
635         sc->lna[1] = val >> 8;          /* channel group 1 */

637         /* fix broken 5GHz LNA entries */
638         if (sc->lna[2] == 0 || sc->lna[2] == 0xff) {
639                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
640                     "invalid LNA for channel group %d\n", 2);
641                 sc->lna[2] = sc->lna[1];
642         }
643         if (sc->lna[3] == 0 || sc->lna[3] == 0xff) {
644                 RWN_DEBUG(RT2860_DBG_EEPROM, "rwn: rt2860_read_eeprom(): "
645                     "invalid LNA for channel group %d\n", 3);
646                 sc->lna[3] = sc->lna[1];
647         }

649         /* fix broken RSSI offset entries */
650         for (ant = 0; ant < 3; ant++) {
651                 if (sc->rssi_2ghz[ant] < -10 || sc->rssi_2ghz[ant] > 10) {
652                         RWN_DEBUG(RT2860_DBG_EEPROM,
653                             "rwn: rt2860_read_eeprom(): "
654                             "invalid RSSI%d offset: %d (2GHz)\n",
655                             ant + 1, sc->rssi_2ghz[ant]);
656                         sc->rssi_2ghz[ant] = 0;
657                 }
658                 if (sc->rssi_5ghz[ant] < -10 || sc->rssi_5ghz[ant] > 10) {
659                         RWN_DEBUG(RT2860_DBG_EEPROM,
660                             "rwn: rt2860_read_eeprom(): "
661                             "invalid RSSI%d offset: %d (2GHz)\n",
662                             ant + 1, sc->rssi_5ghz[ant]);
663                         sc->rssi_5ghz[ant] = 0;
664                 }
665         }

667         return (RT2860_SUCCESS);
668 }
```

*unchanged_portion_omitted_*