```
**********************************************************
    1985 Fri Apr 25 16:07:54 2014
new/usr/src/uts/i86pc/apix/Makefile
4804 apix & pcplusmp are nearly warning free already
Tentatively Reviewed by: Robert Mustacchi <rm@joyent.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
#
#
# uts/i86pc/apix/Makefile
#
# Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
#
#       This makefile drives the production of the pcplusmp "mach"
#       kernel module.
#
#       pcplusmp implementation architecture dependent
#


#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

#
#       Define the module and object file sets.
#
MODULE          = apix
OBJECTS         = $(APIX_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(APIX_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_PSM_MACH_DIR)/$(MODULE)

#
#       Include common rules.
#
include $(UTSBASE)/i86pc/Makefile.i86pc

#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)

DEBUG_FLGS      =
$(NOT_RELEASE_BUILD)DEBUG_DEFS  += $(DEBUG_FLGS)

#
```



```
# Depends on ACPI CA interpreter
#
LDFLAGS         += -dy -N misc/acpica

CERRWARN        += -_gcc=-Wno-uninitialized
CERRWARN        += -_gcc=-Wno-unused-variable
CERRWARN        += -_gcc=-Wno-unused-function
CERRWARN        += -_gcc=-Wno-empty-body

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

#
#       Include common targets.
#
include $(UTSBASE)/i86pc/Makefile.targ
```

```
**********************************************************
   39524 Fri Apr 25 16:07:54 2014
new/usr/src/uts/i86pc/io/hpet_acpi.c
4804 apix & pcplusmp are nearly warning free already
Tentatively Reviewed by: Robert Mustacchi <rm@joyent.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) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
 */

#include <sys/hpet_acpi.h>
#include <sys/hpet.h>
#include <sys/bitmap.h>
#include <sys/inttypes.h>
#include <sys/time.h>
#include <sys/sunddi.h>
#include <sys/ksynch.h>
#include <sys/apic.h>
#include <sys/callb.h>
#include <sys/clock.h>
#include <sys/archsystm.h>
#include <sys/cpupart.h>

static int hpet_init_proxy(int *hpet_vect, iflag_t *hpet_flags);
static boolean_t hpet_install_proxy(void);
static boolean_t hpet_callback(int code);
static boolean_t hpet_cpr(int code);
static boolean_t hpet_resume(void);
static void hpet_cst_callback(uint32_t code);
static boolean_t hpet_deep_idle_config(int code);
static int hpet_validate_table(ACPI_TABLE_HPET *hpet_table);
static boolean_t hpet_checksum_table(unsigned char *table, unsigned int len);
static void *hpet_memory_map(ACPI_TABLE_HPET *hpet_table);
static int hpet_start_main_counter(hpet_info_t *hip);
static int hpet_stop_main_counter(hpet_info_t *hip);
static uint64_t hpet_read_main_counter_value(hpet_info_t *hip);
static uint64_t hpet_set_leg_rt_cnf(hpet_info_t *hip, uint32_t new_value);
static uint64_t hpet_read_gen_cap(hpet_info_t *hip);
static uint64_t hpet_read_gen_config(hpet_info_t *hip);
static uint64_t hpet_read_gen_intrpt_stat(hpet_info_t *hip);
static uint64_t hpet_read_timer_N_config(hpet_info_t *hip, uint_t n);
static hpet_TN_conf_cap_t hpet_convert_timer_N_config(uint64_t conf);
/* LINTED E_STATIC_UNUSED */
static uint64_t hpet_read_timer_N_comp(hpet_info_t *hip, uint_t n);
/* LINTED E_STATIC_UNUSED */
static void hpet_write_gen_cap(hpet_info_t *hip, uint64_t l);
```



```
static void hpet_write_gen_config(hpet_info_t *hip, uint64_t l);
static void hpet_write_gen_intrpt_stat(hpet_info_t *hip, uint64_t l);
static void hpet_write_timer_N_config(hpet_info_t *hip, uint_t n, uint64_t l);
static void hpet_write_timer_N_comp(hpet_info_t *hip, uint_t n, uint64_t l);
static void hpet_disable_timer(hpet_info_t *hip, uint32_t timer_n);
static void hpet_enable_timer(hpet_info_t *hip, uint32_t timer_n);
/* LINTED E_STATIC_UNUSED */
static void hpet_write_main_counter_value(hpet_info_t *hip, uint64_t l);
static int hpet_get_IOAPIC_intr_capable_timer(hpet_info_t *hip);
static int hpet_timer_available(uint32_t allocated_timers, uint32_t n);
static void hpet_timer_alloc(uint32_t *allocated_timers, uint32_t n);
static void hpet_timer_set_up(hpet_info_t *hip, uint32_t timer_n,
    uint32_t interrupt);
static uint_t hpet_isr(char *arg);
static uint32_t hpet_install_interrupt_handler(uint_t (*func)(char *),
    int vector);
static void hpet_uninstall_interrupt_handler(void);
static void hpet_expire_all(void);
static boolean_t hpet_guaranteed_schedule(hrtime_t required_wakeup_time);
static boolean_t hpet_use_hpet_timer(hrtime_t *expire);
static void hpet_use_lapic_timer(hrtime_t expire);
static void hpet_init_proxy_data(void);

/*
 * hpet_state_lock is used to synchronize disabling/enabling deep c-states
 * and to synchronize suspend/resume.
 */
static kmutex_t         hpet_state_lock;
static struct hpet_state {
        boolean_t       proxy_installed;        /* CBE proxy interrupt setup */
        boolean_t       cpr;                    /* currently in CPR */
        boolean_t       cpu_deep_idle;          /* user enable/disable */
        boolean_t       uni_cstate;             /* disable if only one cstate */
} hpet_state = { B_FALSE, B_FALSE, B_TRUE, B_TRUE};
_____unchanged_portion_omitted_

static uint64_t
hpet_read_timer_N_comp(hpet_info_t *hip, uint_t n)
{
        if (hip->timer_n_config[n].size_cap == 1)
                return (*(uint64_t *)
                    HPET_TIMER_N_COMP_ADDRESS(hip->logical_address, n));
        else
                return (*(uint32_t *)
                    HPET_TIMER_N_COMP_ADDRESS(hip->logical_address, n));
}

static uint64_t
hpet_read_main_counter_value(hpet_info_t *hip)
{
        uint64_t        value;
        uint32_t        *counter;
        uint32_t        high1, high2, low;

        counter = (uint32_t *)HPET_MAIN_COUNTER_ADDRESS(hip->logical_address);

        /*
         * 32-bit main counters
         */
        if (hip->gen_cap.count_size_cap == 0) {
                value = (uint64_t)*counter;
                hip->main_counter_value = value;
                return (value);
        }

        /*
```

```
541          * HPET spec claims a 64-bit read can be split into two 32-bit reads
542          * by the hardware connection to the HPET.
543          */
544         high2 = counter[1];
545         do {
546                 high1 = high2;
547                 low = counter[0];
548                 high2 = counter[1];
549         } while (high2 != high1);

551         value = ((uint64_t)high1 << 32) | low;
552         hip->main_counter_value = value;
553         return (value);
554 }

556 static void
574 hpet_write_gen_cap(hpet_info_t *hip, uint64_t l)
575 {
576         *(uint64_t *)HPET_GEN_CAP_ADDRESS(hip->logical_address) = l;
577 }

579 static void
557 hpet_write_gen_config(hpet_info_t *hip, uint64_t l)
558 {
559         *(uint64_t *)HPET_GEN_CONFIG_ADDRESS(hip->logical_address) = l;
560 }
```

_____unchanged_portion_omitted_

```
628 static void
629 hpet_write_main_counter_value(hpet_info_t *hip, uint64_t l)
630 {
631         uint32_t        *address;

633         /*
634          * HPET spec 1.0a states main counter register should be halted before
635          * it is written to.
636          */
637         ASSERT(!(hpet_read_gen_config(hip) & HPET_GCFR_ENABLE_CNF));

639         if (hip->gen_cap.count_size_cap == 1) {
640                 *(uint64_t *)HPET_MAIN_COUNTER_ADDRESS(hip->logical_address)
641                     = l;
642         } else {
643                 address = (uint32_t *)HPET_MAIN_COUNTER_ADDRESS(
644                     hip->logical_address);

646                 address[0] = (uint32_t)(l & 0xFFFFFFFF);
647         }
648 }

605 /*
606  * Add the interrupt handler for I/O APIC interrupt number (interrupt line).
607  *
608  * The I/O APIC line (vector) is programmed in ioapic_init_intr() called
609  * from apic_picinit() psm_ops apic_ops entry point after we return from
610  * apic_init() psm_ops entry point.
611  */
612 static uint32_t
613 hpet_install_interrupt_handler(uint_t (*func)(char *), int vector)
614 {
615         uint32_t retval;

617         retval = add_avintr(NULL, CBE_HIGH_PIL, (avfunc)func, "HPET Timer",
618             vector, NULL, NULL, NULL, NULL);
619         if (retval == 0) {
620                 cmn_err(CE_WARN, "!hpet_acpi: add_avintr() failed");
```



```
621                 return (AE_BAD_PARAMETER);
622         }
623         return (AE_OK);
624 }
```

_____unchanged_portion_omitted_

```
**********************************************************
   63984 Fri Apr 25 16:07:54 2014
new/usr/src/uts/i86pc/io/mp_platform_misc.c
4804 apix & pcplusmp are nearly warning free already
Tentatively Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

 487 /*
 488  * Recompute mask bits for the given interrupt vector.
 489  * If there is no interrupt servicing routine for this
 490  * vector, this function should disable interrupt vector
 491  * from happening at all IPLs. If there are still
 492  * handlers using the given vector, this function should
 493  * disable the given vector from happening below the lowest
 494  * IPL of the remaining hadlers.
 495  */
 496 /*ARGSUSED*/
 497 int
 498 apic_delspl_common(int irqno, int ipl, int min_ipl, int max_ipl)
 499 {
 500         uchar_t vector;
 501         uint32_t bind_cpu;
 502         int intin, irqindex;
 503         int ioapic_ix;
 504         apic_irq_t      *irqptr, *preirqptr, *irqheadptr, *irqp;
 505         ulong_t iflag;

 507         mutex_enter(&airq_mutex);
 508         irqindex = IRQINDEX(irqno);
 509         irqptr = preirqptr = irqheadptr = apic_irq_table[irqindex];

 511         DDI_INTR_IMPLDBG((CE_CONT, "apic_delspl: dip=0x%p type=%d irqno=0x%x "
 512             "vector=0x%x\n", (void *)irqptr->airq_dip,
 513             irqptr->airq_mps_intr_index, irqno, irqptr->airq_vector));

 515         while (irqptr) {
 516                 if (VIRTIRQ(irqindex, irqptr->airq_share_id) == irqno)
 517                         break;
 518                 preirqptr = irqptr;
 519                 irqptr = irqptr->airq_next;
 520         }
 521         ASSERT(irqptr);

 523         irqptr->airq_share--;

 525         mutex_exit(&airq_mutex);

 527         /*
 528          * If there are more interrupts at a higher IPL, we don't need
 529          * to disable anything.
 530          */
 531         if (ipl < max_ipl)
 532                 return (PSM_SUCCESS);

 534         /* return if it is not hardware interrupt */
 535         if (irqptr->airq_mps_intr_index == RESERVE_INDEX)
 536                 return (PSM_SUCCESS);

 538         if (!apic_picinit_called) {
 539                 /*
 540                  * Clear irq_struct. If two devices shared an intpt
 541                  * line & 1 unloaded before picinit, we are hosed. But, then
 542                  * we hope the machine survive.
 543                  */
 544                 irqptr->airq_mps_intr_index = FREE_INDEX;
```



```
 545                 irqptr->airq_temp_cpu = IRQ_UNINIT;
 546                 apic_free_vector(irqptr->airq_vector);
 547                 return (PSM_SUCCESS);
 548         }
 549         /*
 550          * Downgrade vector to new max_ipl if needed. If we cannot allocate,
 551          * use old IPL. Not very elegant, but it should work.
 552          */
 553         if ((irqptr->airq_ipl != max_ipl) && (max_ipl != PSM_INVALID_IPL) &&
 554             !ioapic_mask_workaround[irqptr->airq_ioapicindex]) {
 555                 apic_irq_t      *irqp;
 556                 if ((vector = apic_allocate_vector(max_ipl, irqno, 1))) {
 556                 if (vector = apic_allocate_vector(max_ipl, irqno, 1)) {
 557                         apic_mark_vector(irqheadptr->airq_vector, vector);
 558                         irqp = irqheadptr;
 559                         while (irqp) {
 560                                 irqp->airq_vector = vector;
 561                                 irqp->airq_ipl = (uchar_t)max_ipl;
 562                                 if (irqp->airq_temp_cpu != IRQ_UNINIT) {
 563                                         apic_record_rdt_entry(irqp, irqindex);

 565                                         iflag = intr_clear();
 566                                         lock_set(&apic_ioapic_lock);

 568                                         (void) apic_setup_io_intr(irqp,
 569                                             irqindex, B_FALSE);

 571                                         lock_clear(&apic_ioapic_lock);
 572                                         intr_restore(iflag);
 573                                 }
 574                                 irqp = irqp->airq_next;
 575                         }
 576                 }

 578         } else if (irqptr->airq_ipl != max_ipl &&
 579             max_ipl != PSM_INVALID_IPL &&
 580             ioapic_mask_workaround[irqptr->airq_ioapicindex]) {

 582         /*
 583          * We cannot downgrade the IPL of the vector below the vector's
 584          * hardware priority. If we did, it would be possible for a
 585          * higher-priority hardware vector to interrupt a CPU running at an IPL
 586          * lower than the hardware priority of the interrupting vector (but
 587          * higher than the soft IPL of this IRQ). When this happens, we would
 588          * then try to drop the IPL BELOW what it was (effectively dropping
 589          * below base_spl) which would be potentially catastrophic.
 590          *
 591          * (e.g. Suppose the hardware vector associated with this IRQ is 0x40
 592          * (hardware IPL of 4).  Further assume that the old IPL of this IRQ
 593          * was 4, but the new IPL is 1.  If we forced vector 0x40 to result in
 594          * an IPL of 1, it would be possible for the processor to be executing
 595          * at IPL 3 and for an interrupt to come in on vector 0x40, interrupting
 596          * the currently-executing ISR.  When apic_intr_enter consults
 597          * apic_irqs[], it will return 1, bringing the IPL of the CPU down to 1
 598          * so even though the processor was running at IPL 4, an IPL 1
 599          * interrupt will have interrupted it, which must not happen)).
 600          *
 601          * Effectively, this means that the hardware priority corresponding to
 602          * the IRQ's IPL (in apic_ipls[]) cannot be lower than the vector's
 603          * hardware priority.
 604          *
 605          * (In the above example, then, after removal of the IPL 4 device's
 606          * interrupt handler, the new IPL will continue to be 4 because the
 607          * hardware priority that IPL 1 implies is lower than the hardware
 608          * priority of the vector used.)
 609          */
```

```
610                 /* apic_ipls is indexed by vector, starting at APIC_BASE_VECT */
611                 const int apic_ipls_index = irqptr->airq_vector -
612                     APIC_BASE_VECT;
613                 const int vect_inherent_hwpri = irqptr->airq_vector >>
614                     APIC_IPL_SHIFT;

616                 /*
617                  * If there are still devices using this IRQ, determine the
618                  * new ipl to use.
619                  */
620                 if (irqptr->airq_share) {
621                         int vect_desired_hwpri, hwpri;

623                         ASSERT(max_ipl < MAXIPL);
624                         vect_desired_hwpri = apic_ipltopri[max_ipl] >>
625                             APIC_IPL_SHIFT;

627                         /*
628                          * If the desired IPL's hardware priority is lower
629                          * than that of the vector, use the hardware priority
630                          * of the vector to determine the new IPL.
631                          */
632                         hwpri = (vect_desired_hwpri < vect_inherent_hwpri) ?
633                             vect_inherent_hwpri : vect_desired_hwpri;

635                         /*
636                          * Now, to get the right index for apic_vectortoipl,
637                          * we need to subtract APIC_BASE_VECT from the
638                          * hardware-vector-equivalent (in hwpri).  Since hwpri
639                          * is already shifted, we shift APIC_BASE_VECT before
640                          * doing the subtraction.
641                          */
642                         hwpri -= (APIC_BASE_VECT >> APIC_IPL_SHIFT);

644                         ASSERT(hwpri >= 0);
645                         ASSERT(hwpri < MAXIPL);
646                         max_ipl = apic_vectortoipl[hwpri];
647                         apic_ipls[apic_ipls_index] = max_ipl;

649                         irqp = irqheadptr;
650                         while (irqp) {
651                                 irqp->airq_ipl = (uchar_t)max_ipl;
652                                 irqp = irqp->airq_next;
653                         }
654                 } else {
655                         /*
656                          * No more devices on this IRQ, so reset this vector's
657                          * element in apic_ipls to the original IPL for this
658                          * vector
659                          */
660                         apic_ipls[apic_ipls_index] =
661                             apic_vectortoipl[vect_inherent_hwpri];
662                 }
663         }

665         /*
666          * If there are still active interrupts, we are done.
667          */
668         if (irqptr->airq_share)
669                 return (PSM_SUCCESS);

671         iflag = intr_clear();
672         lock_set(&apic_ioapic_lock);

674         if (irqptr->airq_mps_intr_index == MSI_INDEX) {
675                 /*
```



```
676                  * Disable the MSI vector
677                  * Make sure we only disable on the last
678                  * of the multi-MSI support
679                  */
680                 if (i_ddi_intr_get_current_nenables(irqptr->airq_dip) == 1) {
681                         apic_pci_msi_disable_mode(irqptr->airq_dip,
682                             DDI_INTR_TYPE_MSI);
683                 }
684         } else if (irqptr->airq_mps_intr_index == MSIX_INDEX) {
685                 /*
686                  * Disable the MSI-X vector
687                  * needs to clear its mask and addr/data for each MSI-X
688                  */
689                 apic_pci_msi_unconfigure(irqptr->airq_dip, DDI_INTR_TYPE_MSIX,
690                     irqptr->airq_origirq);
691                 /*
692                  * Make sure we only disable on the last MSI-X
693                  */
694                 if (i_ddi_intr_get_current_nenables(irqptr->airq_dip) == 1) {
695                         apic_pci_msi_disable_mode(irqptr->airq_dip,
696                             DDI_INTR_TYPE_MSIX);
697                 }
698         } else {
699                 /*
700                  * The assumption here is that this is safe, even for
701                  * systems with IOAPICs that suffer from the hardware
702                  * erratum because all devices have been quiesced before
703                  * they unregister their interrupt handlers.  If that
704                  * assumption turns out to be false, this mask operation
705                  * can induce the same erratum result we're trying to
706                  * avoid.
707                  */
708                 ioapic_ix = irqptr->airq_ioapicindex;
709                 intin = irqptr->airq_intin_no;
710                 ioapic_write(ioapic_ix, APIC_RDT_CMD + 2 * intin, AV_MASK);
711         }

713         apic_vt_ops->apic_intrmap_free_entry(&irqptr->airq_intrmap_private);

715         /*
716          * This irq entry is the only one in the chain.
717          */
718         if (irqheadptr->airq_next == NULL) {
719                 ASSERT(irqheadptr == irqptr);
720                 bind_cpu = irqptr->airq_temp_cpu;
721                 if (((uint32_t)bind_cpu != IRQ_UNBOUND) &&
722                     ((uint32_t)bind_cpu != IRQ_UNINIT)) {
723                         ASSERT(apic_cpu_in_range(bind_cpu));
724                         if (bind_cpu & IRQ_USER_BOUND) {
725                                 /* If hardbound, temp_cpu == cpu */
726                                 bind_cpu &= ~IRQ_USER_BOUND;
727                                 apic_cpus[bind_cpu].aci_bound--;
728                         } else
729                                 apic_cpus[bind_cpu].aci_temp_bound--;
730                 }
731                 irqptr->airq_temp_cpu = IRQ_UNINIT;
732                 irqptr->airq_mps_intr_index = FREE_INDEX;
733                 lock_clear(&apic_ioapic_lock);
734                 intr_restore(iflag);
735                 apic_free_vector(irqptr->airq_vector);
736                 return (PSM_SUCCESS);
737         }

739         /*
740          * If we get here, we are sharing the vector and there are more than
741          * one active irq entries in the chain.
```

```
 742          */
 743         lock_clear(&apic_ioapic_lock);
 744         intr_restore(iflag);

 746         mutex_enter(&airq_mutex);
 747         /* Remove the irq entry from the chain */
 748         if (irqptr == irqheadptr) { /* The irq entry is at the head */
 749                 apic_irq_table[irqindex] = irqptr->airq_next;
 750         } else {
 751                 preirqptr->airq_next = irqptr->airq_next;
 752         }
 753         /* Free the irq entry */
 754         kmem_free(irqptr, sizeof (apic_irq_t));
 755         mutex_exit(&airq_mutex);

 757         return (PSM_SUCCESS);
 758 }
```

unchanged_portion_omitted_

```
**********************************************************
   34430 Fri Apr 25 16:07:54 2014
new/usr/src/uts/i86pc/io/pcplusmp/apic.c
4804 apix & pcplusmp are nearly warning free already
Tentatively Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

 216 static void *apic_hdlp;

 218 /*
 219  * apic_let_idle_redistribute can have the following values:
 220  * 0 - If clock decremented it from 1 to 0, clock has to call redistribute.
 221  * apic_redistribute_lock prevents multiple idle cpus from redistributing
 222  */
 223 int     apic_num_idle_redistributions = 0;
 224 static  int apic_let_idle_redistribute = 0;

 218 /* to gather intr data and redistribute */
 219 static void apic_redistribute_compute(void);

 221 /*
 222  *      This is the loadable module wrapper
 223  */

 225 int
 226 _init(void)
 227 {
 228         if (apic_coarse_hrtime)
 229                 apic_ops.psm_gethrtime = &apic_gettime;
 230         return (psm_mod_init(&apic_hdlp, &apic_psm_info));
 231 }
______unchanged_portion_omitted_

 782 /*
 783  * type == -1 indicates it is an internal request. Do not change
 784  * resv_vector for these requests
 785  */
 786 static int
 787 apic_get_ipivect(int ipl, int type)
 788 {
 789         uchar_t vector;
 790         int irq;

 792         if ((irq = apic_allocate_irq(APIC_VECTOR(ipl))) != -1) {
 793                 if ((vector = apic_allocate_vector(ipl, irq, 1))) {
 801                 if (vector = apic_allocate_vector(ipl, irq, 1)) {
 794                         apic_irq_table[irq]->airq_mps_intr_index =
 795                             RESERVE_INDEX;
 796                         apic_irq_table[irq]->airq_vector = vector;
 797                         if (type != -1) {
 798                                 apic_resv_vector[ipl] = vector;
 799                         }
 800                         return (irq);
 801                 }
 802         }
 803         apic_error |= APIC_ERR_GET_IPIVECT_FAIL;
 804         return (-1);    /* shouldn't happen */
 805 }
______unchanged_portion_omitted_
```

```
**********************************************************
    2383 Fri Apr 25 16:07:54 2014
new/usr/src/uts/i86pc/pcplusmp/Makefile
4804 apix & pcplusmp are nearly warning free already
Tentatively Reviewed by: Robert Mustacchi <rm@joyent.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
#
#
# uts/i86pc/pcplusmp/Makefile
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

#
#       This makefile drives the production of the pcplusmp "mach"
#       kernel module.
#
#       pcplusmp implementation architecture dependent
#

#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

#
#       Define the module and object file sets.
#
MODULE          = pcplusmp
OBJECTS         = $(PCPLUSMP_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(PCPLUSMP_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_PSM_MACH_DIR)/$(MODULE)

#
#       Include common rules.
#
include $(UTSBASE)/i86pc/Makefile.i86pc

#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)

DEBUG_FLGS      =
```



```
$(NOT_RELEASE_BUILD)DEBUG_DEFS  += $(DEBUG_FLGS)

#
# Depends on ACPI CA interpreter
#
LDFLAGS         += -dy -N misc/acpica

CFLAGS          += -Dbug1157974 -Dbug1155030

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED
LINTTAGS        += -erroff=E_STATIC_UNUSED
LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

CERRWARN        += -_gcc=-Wno-parentheses
CERRWARN        += -_gcc=-Wno-uninitialized
CERRWARN        += -_gcc=-Wno-unused-function
CERRWARN        += -_gcc=-Wno-unused-variable
CERRWARN        += -_gcc=-Wno-empty-body

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

#
#       Include common targets.
#
include $(UTSBASE)/i86pc/Makefile.targ
```