```
**********************************************************
   48536 Sat Feb 23 00:30:14 2013
new/usr/src/uts/i86pc/io/apix/apix_utils.c
3426 assertion failed: irq < 16 on VMware hardware version 9 (apix related)
Reviewed by: Albert Lee <trisk@nexenta.com>
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 (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 */

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
```



```
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/apix.h>

static int apix_get_avail_vector_oncpu(uint32_t, int, int);
static apix_vector_t *apix_init_vector(processorid_t, uchar_t);
static void apix_cleanup_vector(apix_vector_t *);
static void apix_insert_av(apix_vector_t *, void *, avfunc, caddr_t, caddr_t,
    uint64_t *, int, dev_info_t *);
static void apix_remove_av(apix_vector_t *, struct autovec *);
static void apix_clear_dev_map(dev_info_t *, int, int);
static boolean_t apix_is_cpu_enabled(processorid_t);
static void apix_wait_till_seen(processorid_t, int);

#define GET_INTR_INUM(ihdlp)            \
        (((ihdlp) != NULL) ? ((ddi_intr_handle_impl_t *)(ihdlp))->ih_inum : 0)

apix_rebind_info_t apix_rebindinfo = {0, 0, 0, NULL, 0, NULL};

/*
 * Allocate IPI
 *
 * Return vector number or 0 on error
 */
uchar_t
apix_alloc_ipi(int ipl)
{
        apix_vector_t *vecp;
        uchar_t vector;
        int cpun;
        int nproc;

        APIX_ENTER_CPU_LOCK(0);

        vector = apix_get_avail_vector_oncpu(0, APIX_IPI_MIN, APIX_IPI_MAX);
        if (vector == 0) {
                APIX_LEAVE_CPU_LOCK(0);
                cmn_err(CE_WARN, "apix: no available IPI\n");
                apic_error |= APIC_ERR_GET_IPIVECT_FAIL;
                return (0);
        }

        nproc = max(apic_nproc, apic_max_nproc);
        for (cpun = 0; cpun < nproc; cpun++) {
                vecp = xv_vector(cpun, vector);
                if (vecp == NULL) {
                        vecp = kmem_zalloc(sizeof (apix_vector_t), KM_NOSLEEP);
                        if (vecp == NULL) {
                                cmn_err(CE_WARN, "apix: No memory for ipi");
                                goto fail;
                        }
                        xv_vector(cpun, vector) = vecp;
                }
                vecp->v_state = APIX_STATE_ALLOCED;
                vecp->v_type = APIX_TYPE_IPI;
                vecp->v_cpuid = vecp->v_bound_cpuid = cpun;
                vecp->v_vector = vector;
                vecp->v_pri = ipl;
        }
        APIX_LEAVE_CPU_LOCK(0);
        return (vector);
```

```
 127 fail:
 128         while (--cpun >= 0)
 129                 apix_cleanup_vector(xv_vector(cpun, vector));
 130         APIX_LEAVE_CPU_LOCK(0);
 131         return (0);
 132 }
______unchanged_portion_omitted_

1819 /*
1820  * For interrupts which call add_avintr() before apic is initialized.
1821  * ioapix_setup_intr() will
1822  *   - allocate vector
1823  *   - copy over ISR
1824  */
1825 static void
1826 ioapix_setup_intr(int irqno, iflag_t *flagp)
1827 {
1828         extern struct av_head autovect[];
1829         apix_vector_t *vecp;
1830         apic_irq_t *irqp;
1831         uchar_t ioapicindex, ipin;
1832         ulong_t iflag;
1833         struct autovec *avp;

1832         irqp = apic_irq_table[irqno];
1835         ioapicindex = acpi_find_ioapic(irqno);
1836         ASSERT(ioapicindex != 0xFF);
1837         ipin = irqno - apic_io_vectbase[ioapicindex];

1839         mutex_enter(&airq_mutex);
1840         irqp = apic_irq_table[irqno];

1842         /*
1843          * The irq table entry should not exist unless the interrupts are shared
1844          * In that case, make sure it matches what we would initialize it to.
1845          */
1846         if (irqp != NULL) {
1847                 ASSERT(irqp->airq_mps_intr_index == ACPI_INDEX);
1837         if ((irqp != NULL) && (irqp->airq_mps_intr_index == ACPI_INDEX)) {
1848                 ASSERT(irqp->airq_intin_no == ipin &&
1849                     irqp->airq_ioapicindex == ioapicindex);
1850                 vecp = xv_vector(irqp->airq_cpu, irqp->airq_vector);
1851                 ASSERT(!IS_VECT_FREE(vecp));
1852                 mutex_exit(&airq_mutex);
1853         } else {
1854                 irqp = kmem_zalloc(sizeof (apic_irq_t), KM_SLEEP);
1843                 vecp = apix_alloc_intx(NULL, 0, irqno);

1856                 irqp->airq_cpu = IRQ_UNINIT;
1857                 irqp->airq_origirq = (uchar_t)irqno;
1845                 irqp = apic_irq_table[irqno];
1858                 irqp->airq_mps_intr_index = ACPI_INDEX;
1859                 irqp->airq_ioapicindex = ioapicindex;
1860                 irqp->airq_intin_no = ipin;
1861                 irqp->airq_iflag = *flagp;
1862                 irqp->airq_share++;

1864                 apic_irq_table[irqno] = irqp;
1865                 mutex_exit(&airq_mutex);

1867                 vecp = apix_alloc_intx(NULL, 0, irqno);
1851                 apic_record_rdt_entry(irqp, irqno);
1868         }

1870         /* copy over autovect */
```



```
1871         for (avp = autovect[irqno].avh_link; avp; avp = avp->av_link)
1872                 apix_insert_av(vecp, avp->av_intr_id, avp->av_vector,
1873                     avp->av_intarg1, avp->av_intarg2, avp->av_ticksp,
1874                     avp->av_prilevel, avp->av_dip);

1876         /* Program I/O APIC */
1877         iflag = intr_clear();
1878         lock_set(&apix_lock);

1880         (void) apix_setup_io_intr(vecp);

1882         lock_clear(&apix_lock);
1883         intr_restore(iflag);

1885         APIC_VERBOSE_IOAPIC((CE_CONT, "apix: setup ioapic, irqno %x "
1886             "(ioapic %x, ipin %x) is bound to cpu %x, vector %x\n",
1887             irqno, ioapicindex, ipin, irqp->airq_cpu, irqp->airq_vector));
1888 }
______unchanged_portion_omitted_
```