```
**********************************************************
    2037 Thu Jan 17 14:22:52 2019
new/usr/src/uts/i86pc/apix/Makefile
10246 fix apic_allocate_irq() indentation
**********************************************************
#
# 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.
# Copyright (c) 2019, Joyent, Inc.
# Copyright (c) 2018, Joyent, Inc.
#
#       This makefile drives the production of the apix "mach"
#       kernel module.
#
#       apix 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

# needs work, real bug in apic_allocate_irq()
$(OBJS_DIR)/mp_platform_common.o := SMOFF += indenting
# needs work
$(OBJS_DIR)/psm_common.o := SMOFF += deref_check

#
#       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
```

```
**********************************************************
   68319 Thu Jan 17 14:22:52 2019
new/usr/src/uts/i86pc/io/mp_platform_common.c
10246 fix apic_allocate_irq() indentation
**********************************************************
/*
 * 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, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2016 Nexenta Systems, Inc.
 * Copyright (c) 2017 by Delphix. All rights reserved.
 * Copyright (c) 2019, Joyent, Inc.
 * Copyright (c) 2018, Joyent, Inc.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#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/apic.h>
#include <sys/apic_timer.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/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/cyclic.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/sunndi.h>
#if !defined(__xpv)
#include <sys/hpet.h>
#include <sys/clock.h>
#endif

/*
 *      Local Function Prototypes
 */
static int apic_handle_defconf();
static int apic_parse_mpct(caddr_t mpct, int bypass);
static struct apic_mpfps_hdr *apic_find_fps_sig(caddr_t fptr, int size);
static int apic_checksum(caddr_t bptr, int len);
static int apic_find_bus_type(char *bus);
static int apic_find_bus(int busid);
static struct apic_io_intr *apic_find_io_intr(int irqno);
static int apic_find_free_irq(int start, int end);
struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
static void apic_set_pwroff_method_from_mpcnfhdr(struct apic_mp_cnf_hdr *hdrp);
static void apic_free_apic_cpus(void);
static boolean_t apic_is_ioapic_AMD_813x(uint32_t physaddr);
static int apic_acpi_enter_apicmode(void);

int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
    int child_ipin, struct apic_io_intr **intrp);
int apic_find_bus_id(int bustype);
int apic_find_intin(uchar_t ioapic, uchar_t intin);
void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);

int apic_debug_mps_id = 0;      /* 1 - print MPS ID strings */

/* ACPI SCI interrupt configuration; -1 if SCI not used */
int apic_sci_vect = -1;
iflag_t apic_sci_flags;

#if !defined(__xpv)
/* ACPI HPET interrupt configuration; -1 if HPET not used */
int apic_hpet_vect = -1;
iflag_t apic_hpet_flags;
#endif

/*
 * psm name pointer
 */
char *psm_name;

/* ACPI support routines */
static int acpi_probe(char *);
static int apic_acpi_irq_configure(acpi_psm_lnk_t *acpipsmlnkp, dev_info_t *dip,
    int *pci_irqp, iflag_t *intr_flagp);

int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
    int ipin, int *pci_irqp, iflag_t *intr_flagp);
uchar_t acpi_find_ioapic(int irq);
static int acpi_intr_compatible(iflag_t iflag1, iflag_t iflag2);
```

```c
/* Max wait time (in repetitions) for flags to clear in an RDT entry. */
int apic_max_reps_clear_pending = 1000;

int     apic_intr_policy = INTR_ROUND_ROBIN;

int     apic_next_bind_cpu = 1; /* For round robin assignment */
                                /* start with cpu 1 */

/*
 * If enabled, the distribution works as follows:
 * On every interrupt entry, the current ipl for the CPU is set in cpu_info
 * and the irq corresponding to the ipl is also set in the aci_current array.
 * interrupt exit and setspl (due to soft interrupts) will cause the current
 * ipl to be be changed. This is cache friendly as these frequently used
 * paths write into a per cpu structure.
 *
 * Sampling is done by checking the structures for all CPUs and incrementing
 * the busy field of the irq (if any) executing on each CPU and the busy field
 * of the corresponding CPU.
 * In periodic mode this is done on every clock interrupt.
 * In one-shot mode, this is done thru a cyclic with an interval of
 * apic_redistribute_sample_interval (default 10 milli sec).
 *
 * Every apic_sample_factor_redistribution times we sample, we do computations
 * to decide which interrupt needs to be migrated (see comments
 * before apic_intr_redistribute().
 */

/*
 * Following 3 variables start as % and can be patched or set using an
 * API to be defined in future. They will be scaled to
 * sample_factor_redistribution which is in turn set to hertz+1 (in periodic
 * mode), or 101 in one-shot mode to stagger it away from one sec processing
 */

int     apic_int_busy_mark = 60;
int     apic_int_free_mark = 20;
int     apic_diff_for_redistribution = 10;

/* sampling interval for interrupt redistribution for dynamic migration */
int     apic_redistribute_sample_interval = NANOSEC / 100; /* 10 millisec */

/*
 * number of times we sample before deciding to redistribute interrupts
 * for dynamic migration
 */
int     apic_sample_factor_redistribution = 101;

int     apic_redist_cpu_skip = 0;
int     apic_num_imbalance = 0;
int     apic_num_rebind = 0;

/*
 * Maximum number of APIC CPUs in the system, -1 indicates that dynamic
 * allocation of CPU ids is disabled.
 */
int     apic_max_nproc = -1;
int     apic_nproc = 0;
size_t  apic_cpus_size = 0;
int     apic_defconf = 0;
int     apic_irq_translate = 0;
int     apic_spec_rev = 0;
int     apic_imcrp = 0;

int     apic_use_acpi = 1;      /* 1 = use ACPI, 0 = don't use ACPI */
```

```c
int     apic_use_acpi_madt_only = 0;    /* 1=ONLY use MADT from ACPI */

/*
 * For interrupt link devices, if apic_unconditional_srs is set, an irq resource
 * will be assigned (via _SRS). If it is not set, use the current
 * irq setting (via _CRS), but only if that irq is in the set of possible
 * irqs (returned by _PRS) for the device.
 */
int     apic_unconditional_srs = 1;

/*
 * For interrupt link devices, if apic_prefer_crs is set when we are
 * assigning an IRQ resource to a device, prefer the current IRQ setting
 * over other possible irq settings under same conditions.
 */

int     apic_prefer_crs = 1;

uchar_t apic_io_id[MAX_IO_APIC];
volatile uint32_t *apicioadr[MAX_IO_APIC];
uchar_t apic_io_ver[MAX_IO_APIC];
uchar_t apic_io_vectbase[MAX_IO_APIC];
uchar_t apic_io_vectend[MAX_IO_APIC];
uchar_t apic_reserved_irqlist[MAX_ISA_IRQ + 1];
uint32_t apic_physaddr[MAX_IO_APIC];

boolean_t ioapic_mask_workaround[MAX_IO_APIC];

/*
 * First available slot to be used as IRQ index into the apic_irq_table
 * for those interrupts (like MSI/X) that don't have a physical IRQ.
 */
int apic_first_avail_irq  = APIC_FIRST_FREE_IRQ;

/*
 * apic_ioapic_lock protects the ioapics (reg select), the status, temp_bound
 * and bound elements of cpus_info and the temp_cpu element of irq_struct
 */
lock_t  apic_ioapic_lock;

int     apic_io_max = 0;        /* no. of i/o apics enabled */

struct apic_io_intr *apic_io_intrp = NULL;
static  struct apic_bus *apic_busp;

uchar_t apic_resv_vector[MAXIPL+1];

char    apic_level_intr[APIC_MAX_VECTOR+1];

uint32_t        eisa_level_intr_mask = 0;
        /* At least MSB will be set if EISA bus */

int     apic_pci_bus_total = 0;
uchar_t apic_single_pci_busid = 0;

/*
 * airq_mutex protects additions to the apic_irq_table - the first
 * pointer and any airq_nexts off of that one. It also protects
 * apic_max_device_irq & apic_min_device_irq. It also guarantees
 * that share_id is unique as new ids are generated only when new
 * irq_t structs are linked in. Once linked in the structs are never
 * deleted. temp_cpu & mps_intr_index field indicate if it is programmed
 * or allocated. Note that there is a slight gap between allocating in
 * apic_introp_xlate and programming in addspl.
 */
kmutex_t        airq_mutex;
```

```
apic_irq_t      *apic_irq_table[APIC_MAX_VECTOR+1];
int             apic_max_device_irq = 0;
int             apic_min_device_irq = APIC_MAX_VECTOR;

typedef struct prs_irq_list_ent {
        int                     list_prio;
        int32_t                 irq;
        iflag_t                 intrflags;
        acpi_prs_private_t      prsprv;
        struct prs_irq_list_ent *next;
} prs_irq_list_t;
```

______***unchanged_portion_omitted_***

```
int
apic_allocate_irq(int irq)
{
        int     freeirq, i;

        if ((freeirq = apic_find_free_irq(irq, (APIC_RESV_IRQ - 1))) == -1) {
        if ((freeirq = apic_find_free_irq(irq, (APIC_RESV_IRQ - 1))) == -1)
                if ((freeirq = apic_find_free_irq(APIC_FIRST_FREE_IRQ,
                    (irq - 1))) == -1) {
                        /*
                         * if BIOS really defines every single irq in the mps
                         * table, then don't worry about conflicting with
                         * them, just use any free slot in apic_irq_table
                         */
                        for (i = APIC_FIRST_FREE_IRQ; i < APIC_RESV_IRQ; i++) {
                                if ((apic_irq_table[i] == NULL) ||
                                    apic_irq_table[i]->airq_mps_intr_index ==
                                    FREE_INDEX) {
                                        freeirq = i;
                                        break;
                                }
                        }

                        if (freeirq == -1) {
                                /* This shouldn't happen, but just in case */
                                cmn_err(CE_WARN, "%s: NO available IRQ", psm_nam
                                return (-1);
                        }
                }
        }

        if (apic_irq_table[freeirq] == NULL) {
                apic_irq_table[freeirq] =
                    kmem_zalloc(sizeof (apic_irq_t), KM_NOSLEEP);
                if (apic_irq_table[freeirq] == NULL) {
                        cmn_err(CE_WARN, "%s: NO memory to allocate IRQ",
                            psm_name);
                        return (-1);
                }
                apic_irq_table[freeirq]->airq_temp_cpu = IRQ_UNINIT;
                apic_irq_table[freeirq]->airq_mps_intr_index = FREE_INDEX;
        }
        return (freeirq);
}
```

______***unchanged_portion_omitted_***

```
************************************************************
    2090 Thu Jan 17 14:22:53 2019
new/usr/src/uts/i86pc/pcplusmp/Makefile
10246 fix apic_allocate_irq() indentation
************************************************************
#
# 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.
#
# Copyright (c) 2019, Joyent, Inc.
# Copyright (c) 2018, Joyent, Inc.
#

#
#       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

# needs work
$(OBJS_DIR)/mp_platform_common.o := SMOFF += indenting
# needs work
$(OBJS_DIR)/psm_common.o := SMOFF += deref_check

#
#       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
```