```
**********************************************************
    9813 Wed Apr 23 20:47:24 2014
new/usr/src/uts/i86pc/io/pcplusmp/apic_regops.c
XXXX pcplusmp & apix should use x2apic feature flag
**********************************************************
```

```
/*
 * 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 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright 2014 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */
#endif /* ! codereview */

#include <sys/cpuvar.h>
#include <sys/psm.h>
#include <sys/archsystm.h>
#include <sys/apic.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/mach_intr.h>
#include <sys/sysmacros.h>
#include <sys/trap.h>
#include <sys/x86_archext.h>
#include <sys/privregs.h>
#include <sys/psm_common.h>

/* Function prototypes of local apic and X2APIC */
static uint64_t local_apic_read(uint32_t reg);
static void local_apic_write(uint32_t reg, uint64_t value);
static int get_local_apic_pri(void);
static void local_apic_write_task_reg(uint64_t value);
static void local_apic_write_int_cmd(uint32_t cpu_id, uint32_t cmd1);
static uint64_t local_x2apic_read(uint32_t msr);
static void local_x2apic_write(uint32_t msr, uint64_t value);
static int get_local_x2apic_pri(void);
static void local_x2apic_write_task_reg(uint64_t value);
static void local_x2apic_write_int_cmd(uint32_t cpu_id, uint32_t cmd1);

/*
 * According to the X2APIC specification:
 *
 *   xAPIC global enable    X2APIC enable         Description
 *   (IA32_APIC_BASE[11])   (IA32_APIC_BASE[10])
 * -----------------------------------------------------------
 *      0                       0       APIC is disabled
```



```
 *      0                       1       Invalid
 *      1                       0       APIC is enabled in xAPIC mode
 *      1                       1       APIC is enabled in X2APIC mode
 * -----------------------------------------------------------
 */
int     x2apic_enable = 1;
enum apic_mode apic_mode = LOCAL_APIC;  /* Default mode is Local APIC */

/* Uses MMIO (Memory Mapped IO) */
static apic_reg_ops_t local_apic_regs_ops = {
        local_apic_read,
        local_apic_write,
        get_local_apic_pri,
        local_apic_write_task_reg,
        local_apic_write_int_cmd,
        apic_send_EOI,
};

/* X2APIC : Uses RDMSR/WRMSR instructions to access APIC registers */
static apic_reg_ops_t x2apic_regs_ops = {
        local_x2apic_read,
        local_x2apic_write,
        get_local_x2apic_pri,
        local_x2apic_write_task_reg,
        local_x2apic_write_int_cmd,
        apic_send_EOI,
};

int apic_have_32bit_cr8 = 0;

/* The default ops is local APIC (Memory Mapped IO) */
apic_reg_ops_t *apic_reg_ops = &local_apic_regs_ops;

/*
 * APIC register ops related data sturctures and functions.
 */
void apic_send_EOI();
void apic_send_directed_EOI(uint32_t irq);

#define X2APIC_CPUID_BIT        21
#define X2APIC_ENABLE_BIT       10

/*
 * Local APIC Implementation
 */
static uint64_t
local_apic_read(uint32_t reg)
{
        return ((uint32_t)apicadr[reg]);
}
______unchanged_portion_omitted_

int
apic_detect_x2apic(void)
{
        struct cpuid_regs cp;

        if (x2apic_enable == 0)
                return (0);

        return (is_x86_feature(x86_featureset, X86FSET_X2APIC));
        cp.cp_eax = 1;
        (void) __cpuid_insn(&cp);

        return ((cp.cp_ecx & (0x1 << X2APIC_CPUID_BIT)) ? 1 : 0);
}
______unchanged_portion_omitted_
```