```
**********************************************************
   65764 Sun Mar  2 12:32:54 2014
new/usr/src/uts/i86pc/io/apix/apix.c
4663 apic_cr8pri complicates pcplusmp
**********************************************************
______unchanged_portion_omitted_
 355 #endif

 357 static void
 358 apix_init()
 359 {
 360         extern void (*do_interrupt_common)(struct regs *, trap_trace_rec_t *);

 362         APIC_VERBOSE(INIT, (CE_CONT, "apix: psm_softinit\n"));

 364         do_interrupt_common = apix_do_interrupt;
 365         addintr = apix_add_avintr;
 366         remintr = apix_rem_avintr;
 367         get_pending_spl = apix_get_pending_spl;
 368         get_intr_handler = apix_get_intr_handler;
 369         psm_get_localapicid = apic_get_localapicid;
 370         psm_get_ioapicid = apic_get_ioapicid;

 372         apix_softinit();
 373 #if defined(__amd64)
 374         /*
 375          * Make cpu-specific interrupt info point to cr8pri vector
 376          */
 374         CPU->cpu_pri_data = dummy_cpu_pri;
 375 #else
 376         if (cpuid_have_cr8access(CPU))
 377                 apic_have_32bit_cr8 = 1;
 378 #endif  /* __amd64 */

 380         /*
 381          * Initialize IRM pool parameters
 382          */
 383         if (irm_enable) {
 384                 int     i;
 385                 int     lowest_irq;
 386                 int     highest_irq;

 388                 /* number of CPUs present */
 389                 apix_irminfo.apix_ncpus = apic_nproc;
 390                 /* total number of entries in all of the IOAPICs present */
 391                 lowest_irq = apic_io_vectbase[0];
 392                 highest_irq = apic_io_vectend[0];
 393                 for (i = 1; i < apic_io_max; i++) {
 394                         if (apic_io_vectbase[i] < lowest_irq)
 395                                 lowest_irq = apic_io_vectbase[i];
 396                         if (apic_io_vectend[i] > highest_irq)
 397                                 highest_irq = apic_io_vectend[i];
 398                 }
 399                 apix_irminfo.apix_ioapic_max_vectors =
 400                     highest_irq - lowest_irq + 1;
 401                 /*
 402                  * Number of available per-CPU vectors excluding
 403                  * reserved vectors for Dtrace, int80, system-call,
 404                  * fast-trap, etc.
 405                  */
 406                 apix_irminfo.apix_per_cpu_vectors = APIX_NAVINTR -
 407                     APIX_SW_RESERVED_VECTORS;

 409                 /* Number of vectors (pre) allocated (SCI and HPET) */
 410                 apix_irminfo.apix_vectors_allocated = 0;
 411                 if (apic_hpet_vect != -1)
```



```
 412                         apix_irminfo.apix_vectors_allocated++;
 413                 if (apic_sci_vect != -1)
 414                         apix_irminfo.apix_vectors_allocated++;
 415         }
 416 }
______unchanged_portion_omitted_
```

```
**********************************************************
   35994 Sun Mar  2 12:32:54 2014
new/usr/src/uts/i86pc/io/pcplusmp/apic.c
4663 apic_cr8pri complicates pcplusmp
**********************************************************
______unchanged_portion_omitted_
 137         /*
 138          * The ipl of an ISR at vector X is apic_vectortoipl[X>>4]
 139          * NOTE that this is vector as passed into intr_enter which is
 140          * programmed vector - 0x20 (APIC_BASE_VECT)
 141          */

 143 uchar_t apic_ipltopri[MAXIPL + 1];      /* unix ipl to apic pri */
 144         /* The taskpri to be programmed into apic to mask given ipl */

 146 #if defined(__amd64)
 147 static unsigned char dummy_cpu_pri[MAXIPL + 1];
 147 uchar_t apic_cr8pri[MAXIPL + 1];        /* unix ipl to cr8 pri  */
 148 #endif

 150 /*
 151  * Correlation of the hardware vector to the IPL in use, initialized
 152  * from apic_vectortoipl[] in apic_init().  The final IPLs may not correlate
 153  * to the IPLs in apic_vectortoipl on some systems that share interrupt lines
 154  * connected to errata-stricken IOAPICs
 155  */
 156 uchar_t apic_ipls[APIC_AVAIL_VECTOR];

 158 /*
 159  * Patchable global variables.
 160  */
 161 int     apic_enable_hwsoftint = 0;      /* 0 - disable, 1 - enable      */
 162 int     apic_enable_bind_log = 1;       /* 1 - display interrupt binding log */

 164 /*
 165  *      Local static data
 166  */
 167 static struct   psm_ops apic_ops = {
 168         apic_probe,

 170         apic_init,
 171         apic_picinit,
 172         apic_intr_enter,
 173         apic_intr_exit,
 174         apic_setspl,
 175         apic_addspl,
 176         apic_delspl,
 177         apic_disable_intr,
 178         apic_enable_intr,
 179         (int (*)(int))NULL,             /* psm_softlvl_to_irq */
 180         (void (*)(int))NULL,            /* psm_set_softintr */

 182         apic_set_idlecpu,
 183         apic_unset_idlecpu,

 185         apic_clkinit,
 186         apic_getclkirq,
 187         (void (*)(void))NULL,           /* psm_hrtimeinit */
 188         apic_gethrtime,

 190         apic_get_next_processorid,
 191         apic_cpu_start,
 192         apic_post_cpu_start,
 193         apic_shutdown,
 194         apic_get_ipivect,
 195         apic_send_ipi,
```



```
 197         (int (*)(dev_info_t *, int))NULL,       /* psm_translate_irq */
 198         (void (*)(int, char *))NULL,    /* psm_notify_error */
 199         (void (*)(int))NULL,            /* psm_notify_func */
 200         apic_timer_reprogram,
 201         apic_timer_enable,
 202         apic_timer_disable,
 203         apic_post_cyclic_setup,
 204         apic_preshutdown,
 205         apic_intr_ops,                  /* Advanced DDI Interrupt framework */
 206         apic_state,                     /* save, restore apic state for S3 */
 207         apic_cpu_ops,                   /* CPU control interface. */
 208 };
______unchanged_portion_omitted_

 278 void
 279 apic_init(void)
 280 {
 281         int i;
 282         int     j = 1;

 284         psm_get_ioapicid = apic_get_ioapicid;
 285         psm_get_localapicid = apic_get_localapicid;
 286         psm_xlate_vector_by_irq = apic_xlate_vector_by_irq;

 288         apic_ipltopri[0] = APIC_VECTOR_PER_IPL; /* leave 0 for idle */
 289         for (i = 0; i < (APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL); i++) {
 290                 if ((i < ((APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL) - 1)) &&
 291                     (apic_vectortoipl[i + 1] == apic_vectortoipl[i]))
 292                         /* get to highest vector at the same ipl */
 293                         continue;
 294                 for (; j <= apic_vectortoipl[i]; j++) {
 295                         apic_ipltopri[j] = (i << APIC_IPL_SHIFT) +
 296                             APIC_BASE_VECT;
 297                 }
 298         }
 299         for (; j < MAXIPL + 1; j++)
 300                 /* fill up any empty ipltopri slots */
 301                 apic_ipltopri[j] = (i << APIC_IPL_SHIFT) + APIC_BASE_VECT;
 302         apic_init_common();
 303 #if defined(__amd64)
 304         CPU->cpu_pri_data = dummy_cpu_pri;
 304         /*
 305          * Make cpu-specific interrupt info point to cr8pri vector
 306          */
 307         for (i = 0; i <= MAXIPL; i++)
 308                 apic_cr8pri[i] = apic_ipltopri[i] >> APIC_IPL_SHIFT;
 309         CPU->cpu_pri_data = apic_cr8pri;
 305 #else
 306         if (cpuid_have_cr8access(CPU))
 307                 apic_have_32bit_cr8 = 1;
 308 #endif  /* __amd64 */
 309 }
______unchanged_portion_omitted_

 686 /*
 687  * Any changes made to this function must also change X2APIC
 688  * version of intr_exit.
 689  */
 690 void
 691 apic_intr_exit(int prev_ipl, int irq)
 692 {
 693         apic_cpus_info_t *cpu_infop;

 695 #if defined(__amd64)
 696         setcr8((ulong_t)(apic_ipltopri[prev_ipl] >> APIC_IPL_SHIFT));
```

```
701         setcr8((ulong_t)apic_cr8pri[prev_ipl]);
697 #else
698         if (apic_have_32bit_cr8)
699                 setcr8((ulong_t)(apic_ipltopri[prev_ipl] >> APIC_IPL_SHIFT));
700         else
701                 apicadr[APIC_TASK_REG] = apic_ipltopri[prev_ipl];
702 #endif

704         APIC_INTR_EXIT();
705 }
______unchanged_portion_omitted_

729 /*
730  * Mask all interrupts below or equal to the given IPL.
731  * Any changes made to this function must also change X2APIC
732  * version of setspl.
733  */
734 static void
735 apic_setspl(int ipl)
736 {
737 #if defined(__amd64)
738         setcr8((ulong_t)(apic_ipltopri[ipl] >> APIC_IPL_SHIFT));
743         setcr8((ulong_t)apic_cr8pri[ipl]);
739 #else
740         if (apic_have_32bit_cr8)
741                 setcr8((ulong_t)(apic_ipltopri[ipl] >> APIC_IPL_SHIFT));
742         else
743                 apicadr[APIC_TASK_REG] = apic_ipltopri[ipl];
744 #endif

746         /* interrupts at ipl above this cannot be in progress */
747         apic_cpus[psm_get_cpu_id()].aci_ISR_in_progress &= (2 << ipl) - 1;
748         /*
749          * this is a patch fix for the ALR QSMP P5 machine, so that interrupts
750          * have enough time to come in before the priority is raised again
751          * during the idle() loop.
752          */
753         if (apic_setspl_delay)
754                 (void) apic_reg_ops->apic_get_pri();
755 }
______unchanged_portion_omitted_
```