```
**********************************************************
   22365 Wed Jun 13 16:47:23 2012
new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev_cpu.c
XXXX acpidev_cpu_query_MAT() needs to be more paranoid
**********************************************************
______unchanged_portion_omitted_

 240 /*
 241  * Extract information for enabled CPUs from the buffer returned
 242  * by the _MAT method.
 243  */
 244 static ACPI_STATUS
 245 acpidev_cpu_query_MAT(ACPI_SUBTABLE_HEADER *ap, void *context)
 246 {
 247         ACPI_MADT_LOCAL_APIC *mpa;
 248         ACPI_MADT_LOCAL_X2APIC *mpx2a;
 249         struct acpidev_cpu_MAT_arg *rp;

 251         rp = (struct acpidev_cpu_MAT_arg *)context;
 252         switch (ap->Type) {
 253         case ACPI_MADT_TYPE_LOCAL_APIC:
 254                 mpa = (ACPI_MADT_LOCAL_APIC *)ap;
 255                 if (mpa->LapicFlags & ACPI_MADT_ENABLED) {
 256                         ASSERT(mpa->Id != 255);
 257                         rp->found = B_TRUE;
 258                         rp->proc_id = mpa->ProcessorId;
 259                         rp->apic_id = mpa->Id;
 259                 if (mpa->LapicFlags & ACPI_MADT_ENABLED) {
 260                         rp->enabled = B_TRUE;
 261                 } else {
 262                         rp->enabled = B_FALSE;
 263                         if (mpa->ProcessorId == 255) {
 264                                 /*
 265                                  * Some platforms give us ProcessorId of
 266                                  * 255 and apic_id of 255.  Callers might
 267                                  * be able to handle apic_id of 255, but
 268                                  * proc_id of 255 will be problematic.
 269                                  * Consider the MAT query a failure in that
 270                                  * case.
 271                                  */
 272                                 rp->found = B_FALSE;
 273                                 ACPIDEV_DEBUG(CE_WARN,
 274                                     "!acpidev: encountered CPU "
 275                                     "with APIC Proc Id == 255 in _MAT.");
 276                         } else {
 277                                 /*
 278                                  * NOTE: apic_id may be 255, but callers who
 279                                  * care about apic_id use other methods to
 280                                  * get it, for some strange reason.
 281                                  */
 282                                 rp->found = B_TRUE;
 283                                 rp->proc_id = mpa->ProcessorId;
 284                                 rp->apic_id = mpa->Id;
 285                         }
 286                 }
 287                 return (AE_CTRL_TERMINATE);

 289         case ACPI_MADT_TYPE_LOCAL_X2APIC:
 290                 mpx2a = (ACPI_MADT_LOCAL_X2APIC *)ap;
 291                 if (mpx2a->LocalApicId >= 255) {
 292                         rp->found = B_TRUE;
 293                         rp->proc_id = mpx2a->Uid;
 294                         rp->apic_id = mpx2a->LocalApicId;
 295                         if (mpx2a->LapicFlags & ACPI_MADT_ENABLED) {
 296                                 rp->enabled = B_TRUE;
 297                         } else {
```



```
 298                                 rp->enabled = B_FALSE;
 299                         }
 300                         return (AE_CTRL_TERMINATE);
 301                 } else {
 302                         ACPIDEV_DEBUG(CE_WARN, "!acpidev: encountered CPU "
 303                             "with X2APIC Id < 255 in _MAT.");
 304                 }
 305                 break;

 307         case ACPI_MADT_TYPE_LOCAL_APIC_NMI:
 308                 /* UNIMPLEMENTED */
 309                 break;

 311         case ACPI_MADT_TYPE_LOCAL_X2APIC_NMI:
 312                 /* UNIMPLEMENTED */
 313                 break;

 315         default:
 316                 /*
 317                  * According to the ACPI Spec, the buffer returned by _MAT
 318                  * for a processor object should only contain Local APIC,
 319                  * Local SAPIC, and local APIC NMI entries.
 320                  * x2APIC Specification extends it to support Processor
 321                  * x2APIC and x2APIC NMI Structure.
 322                  */
 323                 ACPIDEV_DEBUG(CE_NOTE,
 324                     "!acpidev: unknown APIC entry type %u in _MAT.", ap->Type);
 325                 break;
 326         }

 328         return (AE_OK);
 329 }
______unchanged_portion_omitted_
```