

[PATCH] amd76x_pm: C3 powersaving for AMD K7

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-01/msg11501.html>

- *From:* Joerg Sommrey <jo@xxxxxxxxxx>
 - *Date:* Tue, 31 Jan 2006 19:55:16 +0100
-

This is a processor idle module for AMD SMP 760MP(X) based systems. The patch was originally written by Tony Lindgren and has been around since 2002. It enables C2 mode on AMD SMP systems and thus saves about 70 – 90 W of energy in the idle mode compared to the default idle mode. The idle function has been rewritten and is now free of locking issues and is independent from the number of CPUs. The impact from this module on the system clock and on i/o transfer are now fairly low.

This patch can also be found at
http://www.sommrey.de/amd76x_pm/amd76x_pm-2.6.15-4.patch

In this version more locking was added to make sure all or no CPU enter C3 mode.

Signed-off-by: Joerg Sommrey <jo@xxxxxxxxxx>

```
diff -Nru linux-2.6.15/Documentation/amd76x_pm.txt linux-2.6.15-jo/Documentation/amd76x_pm.txt
--- linux-2.6.15/Documentation/amd76x_pm.txt 1970-01-01 01:00:00.000000000 +0100
+++ linux-2.6.15-jo/Documentation/amd76x_pm.txt 2006-01-31 19:26:08.000000000 +0100
@@ -0,0 +1,330 @@
+ ACPI style power management for SMP AMD-760MP(X) based systems
+ =====
+
+The ACPI project started to support C3 for SMP. This doesn't work for
+me yet, so there still is a need for a special module dealing with
+AMD-760MP(X) systems.
+
+Using this module saves about 70 – 90W of energy in the idle mode compared
+to the default idle mode. Waking up from the idle mode is fast to keep the
+system response time good.
+
+This code runs on two-way AMD boxes at the moment, but the basic idling
+algorithm is independent of the number of CPUs.
+
+Known issues:
+-----
+- Currently there's a bug somewhere where the reading the
+ P_LVL2 for the first time causes the system to sleep instead of
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

- + idling. This means that you need to hit the power button once to
- + wake the system after loading the module for the first time after
- + reboot. After that the system idles as supposed.
- + (Only observed on Tony's system – AMD766.)
- +
- +– On kernel up to and including 2.6.13 there might appear a weird
- + keyboard repeating after an uptime of 1–2 days. This seems to
- + coincide with IRQ 0 being routed to CPU1: setting smp_affinity to 1 for
- + IRQ 0 soon after booting circumvents the problem. On 2.6.14 interrupt
- + routing has changed and I didn't watch the problem anymore.
- +
- +– There might be reduced throughput of disk or network devices. See
- + below for how to configure the "irq rate watcher" to avoid this.
- +
- +– Occasional hard lockups might be caused by amd76x_pm. I'm still working
- + on this issue.
- +
- +– I have reports about a buzzing sound from the mainboard when amd76x_pm
- + is loaded and some reports about unclean audio playback.
- +
- +Influenced by Vcool, and LVCool. Rewrote everything from scratch to
- +use the PCI features in Linux, and to support SMP systems.
- +
- +Currently tested amongst others on a TYAN S2460 (760MP) system (Tony), an
- +ASUS A7M266–D (760MPX) system (Johnathan) and a TYAN S2466 (760MPX)
- +system (Jo). Adding support for other Athlon SMP or single processor
- +systems should be easy if desired.
- +
- +The file /sys/devices/pci0000:00/0000:00:00.0/C3_cnt shows the number of
- +C3 calls since module load.
- +
- +There are some parameters for tuning the behaviour of amd76x_pm:
- +lazy_idle, spin_idle, watch_irqs, watch_int and min_C1
- +
- +lazy_idle and spin_idle are closely related:
- +
- +– lazy_idle defines the number of idle calls into amd76x_smp_idle that are
- + needed to *enable* C3 mode. This parameter is the maximum loop counter
- + for an outer loop with interrupts enabled that guarantees low latencies.
- + The default for lazy_idle is 512.
- +
- +– spin_idle defines the maximum number of spin cycles in an inner idle loop
- + where one CPU waits for all others to get into C3–enabled mode. When all
- + CPUs are in C2–enable mode they (more ore less) simultaneously *enter* C3
- + mode. In this inner loop interrupts are disabled. The loop is left
- + immediatly if there is something waiting to be scheduled. The default
- + for spin_idle is 4*lazy_idle.
- +
- +lazy_idle and spin_idle define a "rubber measure" for the idling
- +behaviour: lazy_idle defines the minimum "idling" needed to enter C3 and
- +spin_idle defines when to give up.

[PATCH] amd76x_pm: C3 powersaving for AMD K7

+
+Low values for lazy_idle and high values for spin_idle give better
+cooling. Higher values for lazy_idle simply give less cooling. spin_idle
+is a kind of emergency break to leave C3-enable mode if CPUs don't
+synchronize.
+
+irq rate watcher:
+-----
+Interrupts are disabled in C3 mode. The CPUs are woken up by timer
+interrups, by NMIs and some other events. This causes a high interrupt
+latency for other interrupts that leads to a significant reduction in io
+or network throughput. There has been introduced a "irq rate watcher" to
+get rid of this issue. If the irq rate watcher detects that an interrupt
+has a rate above a given limit, C3 idling is disabled and a low latency C1
+idling mode is used instead. The parameters watch_irqs, watch_int and
+min_C1 control this irq rate watcher:
+
+-- watch_irqs defines which interrupts are to be watched and optionally
+ at which interrupt rate C3 mode shall be disabled. The syntax for
+ watch_irqs is irq1[:rate1],irq2[:rate2],... The rate is measured in
+ interrupts per second and defaults to 128. There is no default for
+ watch_irqs. To enable the irq rate watcher you must specify this
+ parameter. Enter the interrupts used by disk controllers or network
+ adapters here.
+
+-- watch_int defines the time interval (in milliseconds) at which the
+ interrupt rate is checked. Too low values may result in an overhead
+ and too high values cause the C1 mode to "kick in" later. The default
+ for watch_int is 1 second.
+
+-- min_C1 defines the minimum number of check intervals with low
+ interrupt rates that are needed to leave the forced C1 mode.
+
+All parameters lazy_idle, spin_idle, watch_irqs and watch_int may be
+given as module parameters to amd76x_pm. Furthermore, they may be
+queried or changed via their sysfs entries in
+/sys/module/amd76x_pm/parameters. (The location in sysfs has changed
+from earlier versions!)
+
+Setting watch_int to zero or removing all interrupts from watch_irqs
+causes the irq rate watcher to stop. The module needs to be reloaded to
+start the watcher again.
+
+NB: In the past there was a major impact from amd76x_pm on the system
+clock stability. At least on my box this effect has been reduced
+noticeable. This is caused by two changes:
+
+-- The redesign of the idle function lead to a reduced time spent in
+ amd76x_smp_idle() and this propably caused lower latencies.
+
+-- The introduction of the irq rate watcher significantly reduced

[PATCH] amd76x_pm: C3 powersaving for AMD K7

+ latencies under load.
+
+Some hints on tuning lazy_idle and spin_idle:
+Watch the rate of C3 calls per second. Anything below HZ seems to be
+completely useless. Start with a moderate low value for lazy_idle, e.g.
+2^5 and a very high value for spin_idle, e.g. 2^20. Then double lazy_idle
+until you see a significant increase in core temperature or a reduction
+of the C3 rate. Go back to the last "good" value. Then halve spin_idle
+until you see a decrease in C3 rate or a raising core temperature again
+and go back to the previous value. There's a small program c3rate.c
+attached at the end of this file that you may use to watch the C3 rate.
+The rtc driver must be enabled to use it.
+
+If you want to play around in this field: There is a different version
+of amd76x_pm.c that has experimental und untested support for "normal
+throttling" and "power on suspend". See http://www.sommrey.de/amd76x_pm
+
+This software is licensed under GNU General Public License Version 2
+as specified in file COPYING in the Linux kernel source tree main
+directory.
+
+Copyright (C) 2002 Johnathan Hicks <thetech@xxxxxxxxxxxx>
+ Tony Lindgren <tony@xxxxxxxxxxxx>
+
+Copyright (C) 2005 – 2006 Joerg Sommrey <jo@xxxxxxxxxxxx>
+
+History:
+
+ 20020702 – amd-smp-idle: Tony Lindgren <tony@xxxxxxxxxxxx>
+ Influenced by Vcool, and LVCool. Rewrote everything from scratch to
+ use the PCI features in Linux, and to support SMP systems. Provides
+ C2 idling on SMP AMD-760MP systems.
+
+ 20020722: JH
+ I adapted Tony's code for the AMD-765/766 southbridge and adapted it
+ according to the AMD-768 data sheet to provide the same capability for
+ SMP AMD-760MPX systems. Posted to acpi-devel list.
+
+ 20020722: Alan Cox
+ Replaces non-functional amd76x_pm code in -ac tree.
+
+ 20020730: JH
+ Added ability to do normal throttling (the non-thermal kind), C3 idling
+ and Power On Suspend (S1 sleep). It would be very easy to tie swsusp
+ into activate_amd76x_SLP(). C3 idling doesn't happen yet; see my note
+ in amd76x_smp_idle(). I've noticed that when NTH and idling are both
+ enabled, my hardware locks and requires a hard reset, so I have
+ #ifnedefed around the idle loop setting to prevent this. POS locks it up
+ too, both ought to be fixable. I've also noticed that idling and NTH
+ make some interference that is picked up by the onboard sound chip on
+ my ASUS A7M266-D motherboard.

[PATCH] amd76x_pm: C3 powersaving for AMD K7

[PATCH] amd76x_pm: C3 powersaving for AMD K7

- +
- + 20030601: Pasi Savolainen
- + Simple port to 2.5
- + Added sysfs interface for making nice graphs with mrtg.
- + Look for /sys/devices/pci0/00:00.0/C2_cnt & lazy_idle (latter writable)
- +
- + 20050601: Joerg Sommrey (jo)
- + 2.6 stuff
- + redesigned amd76x_smp_idle. The algorithm is basically the same
- + but the implementation has changed. This part is now independent
- + from the number of CPUs and data is locked against concurrent
- + updates from different CPUs.
- + use _smp_processor_id()
- + use cpu_idle_wait()
- + NTH and POS code not touched and not tested.
- +
- + 20050621: jo
- + separated C3, NTH and POS code into extra patch
- +
- + 20050812: jo
- + rewritten amd76x_smp_idle completely. It's much simpler now but
- + does a good job – the KISS approach. Introduced a new tunable
- + spin_idle.
- +
- + 20050819: jo
- + new irq rate watcher task that forces C1-idling if irq-rate exceeds a
- + given limit.
- +
- + 20050820: jo
- + make all modules parameters r/w in sysfs.
- +
- + 20050906: jo
- + avoid some local_irq_disable/enable
- +
- + 20060108: jo
- + using percpu-variables
- +
- + 20060110: jo
- + eliminated a race condition in the idle loop. Thanks to Arjan van
- + de Ven, who pointed to this issue.
- +
- + 20060121: jo
- + Switched from C2 to C3 idling, inspired by processor_idle.c
- + C2 idling has just been replaced by C3 idling, there is no C2/C3
- + transition.
- + Restoring all touched northbridge and southbridge register bits to
- + their original values on module unload.
- +
- + 20060129: jo
- + Major redesign of the idle loop again. Now there is a two-phase
- + process to make sure all or no CPUs go into C3. In phase one each

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ if (fd == -1) {
+ perror("/dev/rtc");
+ exit(errno);
+ }
+
+ /* Set the alarm to every second */
+
+ retval = ioctl(fd, RTC_UIE_ON, 0);
+ if (retval == -1) {
+ perror("ioctl(RTC_UIE_ON)");
+ exit(errno);
+ }
+ atexit(disable_rtc);
+
+ irqcount = 0;
+ while (1) {
+ /* This blocks until the alarm ring causes an interrupt */
+ retval = read(fd, &data, sizeof data);
+ if (retval == -1) {
+ perror("read");
+ exit(errno);
+ }
+ if (firstloop) {
+ idlebuffer[0] = '\0';
+ idlefd = open(C3_CNT, O_RDONLY);
+ if (idlefd != -1) {
+ bytes = read(idlefd, idlebuffer, sizeof idlebuffer);
+ close(idlefd);
+ idlebuffer[bytes] = '\0';
+ }
+ c3_old = strtoul(idlebuffer, NULL, 10);
+ firstloop = 0;
+ continue;
+ }
+ lastcount = data >> 8;
+ irqcount += lastcount;
+
+ if (irqcount >= rate) {
+ idlebuffer[0] = '\0';
+ idlefd = open(C3_CNT, O_RDONLY);
+ if (idlefd != -1) {
+ bytes = read(idlefd, idlebuffer, sizeof idlebuffer);
+ close(idlefd);
+ idlebuffer[bytes] = '\0';
+ }
+ c3_new = strtoul(idlebuffer, NULL, 10);
+ printf("%lu\n", (c3_new - c3_old) / irqcount);
+ c3_old = c3_new;
+ irqcount = 0;
+ }
+ }
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ }
+
+ return 0;
+
+} /* end main */
+----- c3rate.c ----- snip here -----
diff -Nru linux-2.6.15/drivers/acpi/Kconfig linux-2.6.15-jo/drivers/acpi/Kconfig
--- linux-2.6.15/drivers/acpi/Kconfig 2006-01-03 21:07:59.000000000 +0100
+++ linux-2.6.15-jo/drivers/acpi/Kconfig 2006-01-22 12:32:25.000000000 +0100
@@ -340,4 +340,18 @@
$>modprobe acpi_memhotplug
endif # ACPI

+config AMD76X_PM
+ tristate "AMD76x Native Power Management support"
+ default n
+ depends on X86 && PCI
+ ---help---
+ This driver enables Power Management on AMD760MP & AMD760MPX chipsets.
+ This is about same as ACPI C3, which doesn't seem to work on
+ AMD760MPX a.t.m.
+ See Documentation/amd76x_pm.txt for further details.
+
+ Say M here to build a module called amd76x_pm.
+
+ If unsure, say N.
+
endmenu
diff -Nru linux-2.6.15/drivers/acpi/Makefile linux-2.6.15-jo/drivers/acpi/Makefile
--- linux-2.6.15/drivers/acpi/Makefile 2006-01-03 21:07:59.000000000 +0100
+++ linux-2.6.15-jo/drivers/acpi/Makefile 2006-01-22 12:34:17.000000000 +0100
@@ -57,3 +57,8 @@
obj-$(CONFIG_ACPI_TOSHIBA) += toshiba_acpi.o
obj-y += scan.o motherboard.o
obj-$(CONFIG_ACPI_HOTPLUG_MEMORY) += acpi_memhotplug.o
+
+#
+# not really ACPI thing, but closely related
+#
+obj-$(CONFIG_AMD76X_PM) += amd76x_pm.o
diff -Nru linux-2.6.15/drivers/acpi/amd76x_pm.c linux-2.6.15-jo/drivers/acpi/amd76x_pm.c
--- linux-2.6.15/drivers/acpi/amd76x_pm.c 1970-01-01 01:00:00.000000000 +0100
+++ linux-2.6.15-jo/drivers/acpi/amd76x_pm.c 2006-01-31 19:26:32.000000000 +0100
@@ -0,0 +1,749 @@
+/*
+ * ACPI style PM for SMP AMD-760MP(X) based systems.
+ *
+ * Copyright (C) 2002 Johnathan Hicks <thetech@xxxxxxxxxxxx>
+ * Tony Lindgren <tony@xxxxxxxxxxxx>
+ *
+ * Copyright (C) 2005-2006 Joerg Sommrey <jo@xxxxxxxxxxxx>
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ *
+ * See Documentation/amd76x_pm.txt for details.
+ *
+ * This software is licensed under GNU General Public License Version 2
+ * as specified in file COPYING in the Linux kernel source tree main
+ * directory.
+ */
+
+#include <linux/config.h>
+#include <linux/module.h>
+#include <linux/pci.h>
+#include <linux/delay.h>
+#include <linux/pm.h>
+#include <linux/device.h>
+#include <linux/init.h>
+#include <linux/fs.h>
+#include <linux/version.h>
+#include <asm/atomic.h>
+#include <linux/kernel.h>
+#include <linux/workqueue.h>
+#include <linux/jiffies.h>
+#include <linux/kernel_stat.h>
+#include <linux/spinlock.h>
+
+#include <linux/amd76x_pm.h>
+
+#define VERSION "20060129"
+
+// #define AMD76X_LOG_C1 1
+
+extern void default_idle(void);
+static void amd76x_smp_idle(void);
+static int amd76x_pm_main(void);
+
+static unsigned long lazy_idle = 0;
+static unsigned long spin_idle = 0;
+static unsigned long watch_int = 0;
+static unsigned long min_C1 = AMD76X_MIN_C1;
+static unsigned int sb_id = 0;
+static unsigned int nb_id = 0;
+
+static int show_watch_irqs(char *, struct kernel_param *);
+static int set_watch_irqs(const char *, struct kernel_param *);
+
+module_param(lazy_idle, long, S_IRUGO | S_IWUSR);
+
+MODULE_PARM_DESC(lazy_idle,
+ "\tnumber of idle cycles before entering power saving mode");
+
+module_param(spin_idle, long, S_IRUGO | S_IWUSR);
+MODULE_PARM_DESC(spin_idle,
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ "\tnumber of spin cycles to wait for other CPUs to become idle");
+
+module_param(watch_int, long, S_IRUGO | S_IWUSR);
+MODULE_PARM_DESC(watch_int,
+ "\twatch interval (in milliseconds) for interrupts");
+
+module_param_call(watch_irqs, set_watch_irqs, show_watch_irqs,
+ NULL, S_IRUGO | S_IWUSR);
+MODULE_PARM_DESC(watch_irqs,
+ "\tlist of irqs (and optional their limit per second) that "
+ "cause fallback to C1 mode. "
+ "Syntax: irq0[:limit0],irq1[:limit1],...");
+
+module_param(min_C1, long, S_IRUGO | S_IWUSR);
+MODULE_PARM_DESC(min_C1,
+ "\tminimum irq rate watch intervals spent in C1 mode");
+
+MODULE_AUTHOR("Tony Lindgren, Johnathan Hicks, Joerg Sommrey, others");
+MODULE_DESCRIPTION("ACPI style power management for SMP AMD-760MP(X) "
+ "based systems, version " VERSION);
+
+
+static struct pci_dev *pdev_nb;
+static struct pci_dev *pdev_sb;
+
+struct PM_cfg {
+ unsigned int C3_reg;
+ unsigned int C2_reg;
+ unsigned int status_reg;
+ void (*orig_idle)(void);
+ void (*curr_idle)(void);
+};
+
+static struct PM_cfg amd76x_pm_cfg __read_mostly;
+
+struct cpu_stat {
+ int idle_count;
+ int C3_cnt;
+};
+
+struct idle_stat {
+ atomic_t num_idle;
+ spinlock_t lock;
+ int num_prepare;
+ int num_ready;
+ atomic_t num_C3;
+};
+
+static struct idle_stat amd76x_stat __cacheline_aligned_in_smp = {
+ .num_idle = ATOMIC_INIT(0),
+ .lock = SPIN_LOCK_UNLOCKED,
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ .num_prepare = 0,
+ .num_ready = 0,
+ .num_C3 = ATOMIC_INIT(0),
+};
+
+static void *prs_ref __read_mostly;
+
+struct watch_item {
+ int irq;
+ int count;
+ int limit;
+};
+
+struct reg_save {
+ unsigned int nb58;
+ unsigned int nb60;
+ unsigned int nb68;
+ unsigned int nb70;
+ unsigned int sb41;
+ unsigned int sb4f;
+ unsigned int sb50;
+};
+
+static struct reg_save amd76x_saved;
+
+static int schedule_watch;
+static int force_C1 __read_mostly;
+static struct watch_item watch_item[AMD76X_WATCH_MAX] = {{-1, 0}};
+
+static struct work_struct work;
+static void watch_irq(void *);
+
+static DECLARE_WORK(work, watch_irq, NULL);
+
+static struct pci_device_id __devinitdata amd_nb_tbl[] = {
+ {PCI_VENDOR_ID_AMD, PCI_DEVICE_ID_AMD_FE_GATE_700C, PCI_ANY_ID,
+ PCI_ANY_ID,},
+ {0,}
+};
+
+static struct pci_device_id __devinitdata amd_sb_tbl[] = {
+ {PCI_VENDOR_ID_AMD, PCI_DEVICE_ID_AMD_VIPER_7413, PCI_ANY_ID,
+ PCI_ANY_ID,},
+ {PCI_VENDOR_ID_AMD, PCI_DEVICE_ID_AMD_VIPER_7443, PCI_ANY_ID,
+ PCI_ANY_ID,},
+ {0,}
+};
+
+/*
+ * Configures the AMD-762 northbridge to support PM calls
+ */
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+static int
+config_amd762(int enable)
+{
+ unsigned int regdword;
+
+ /* Enable/disable STPGNT in BIU Status/Control for cpu0,
+ * page 64 in AMD-762 doc */
+ pci_read_config_dword(pdev_nb, 0x60, &regdword);
+ if (enable) {
+ amd76x_saved.nb60 = regdword & STP_GRANT_DISCON_EN;
+ regdword |= STP_GRANT_DISCON_EN;
+ } else
+ regdword &= amd76x_saved.nb60 | ~STP_GRANT_DISCON_EN;
+ pci_write_config_dword(pdev_nb, 0x60, regdword);
+
+ /* Enable/disable STPGNT in BIU Status/Control for cpu1,
+ * page 69 in AMD-762 doc */
+ pci_read_config_dword(pdev_nb, 0x68, &regdword);
+ if (enable) {
+ amd76x_saved.nb68 = regdword & STP_GRANT_DISCON_EN;
+ regdword |= STP_GRANT_DISCON_EN;
+ } else
+ regdword &= amd76x_saved.nb68 | ~STP_GRANT_DISCON_EN;
+ pci_write_config_dword(pdev_nb, 0x68, regdword);
+
+ /* dis/enable "DRAM refresh disable", page 60 in AMD-762 doc */
+ pci_read_config_dword(pdev_nb, 0x58, &regdword);
+ if (enable) {
+ amd76x_saved.nb58 = regdword & REF_DIS;
+ regdword &= ~REF_DIS;
+ } else
+ regdword |= amd76x_saved.nb58 & REF_DIS;
+ pci_write_config_dword(pdev_nb, 0x58, regdword);
+
+ /* Self refresh enable, page 75 in AMD-762 doc */
+ pci_read_config_dword(pdev_nb, 0x70, &regdword);
+ if (enable) {
+ amd76x_saved.nb70 = regdword & SELF_REF_EN;
+ regdword |= SELF_REF_EN;
+ } else
+ regdword &= amd76x_saved.nb70 | ~SELF_REF_EN;
+ pci_write_config_dword(pdev_nb, 0x70, regdword);
+
+ return 0;
+}
+
+ /* Get the base PMIO address and set the pm registers in amd76x_pm_cfg.
+ */
+static void
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+amd76x_get_PM(void)
+{
+ unsigned int regdword;
+
+ /* Get the address for pm status, P_LVL2, etc */
+ pci_read_config_dword(pdev_sb, 0x58, &regdword);
+ regdword &= 0xff80;
+ amd76x_pm_cfg.status_reg = (regdword + 0x00);
+ amd76x_pm_cfg.C3_reg = (regdword + 0x15);
+}
+
+
+/*
+ * En/Disable PMIO and configure W4SG & STPGNT.
+ */
+static int
+config_PMIO_amd76x(int is_766, int enable)
+{
+ unsigned char regbyte;
+
+ /* Set W4SG and PMIOEN, if using a 765/766 set STPGNT as well.
+ * AMD-766: C3A41; page 59 in AMD-766 doc
+ * AMD-768: DevB:3x41C; page 94 in AMD-768 doc */
+ pci_read_config_byte(pdev_sb, 0x41, &regbyte);
+ if(enable) {
+ amd76x_saved.sb41 = regbyte &
+ ((is_766 ? STPGNT : 0) | W4SG | PMIOEN);
+ regbyte |= ((is_766 ? STPGNT : 0) | W4SG | PMIOEN);
+ } else
+ regbyte &= amd76x_saved.sb41 |
+ ~((is_766 ? STPGNT : 0) | W4SG | PMIOEN);
+ pci_write_config_byte(pdev_sb, 0x41, regbyte);
+ return 0;
+}
+
+/*
+ * Untested C3 idle support for AMD-766.
+ */
+static void
+config_amd766_C3(int enable)
+{
+ unsigned int regdword;
+
+ /* Set C3 options in C3A50, page 63 in AMD-766 doc */
+ pci_read_config_dword(pdev_sb, 0x50, &regdword);
+ if(enable) {
+ amd76x_saved.sb50 = regdword & ((DCSTOP_EN | PCISTP_EN |
+ SUSPND_EN | CPURST_EN | STPCLK_EN |
+ CPUSLP_EN | CPUTSTP_EN) << C3_REGS);
+ regdword &= ~((DCSTOP_EN | PCISTP_EN | SUSPND_EN | CPURST_EN)
+ <<< C3_REGS);
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ regdword |= (STPCLK_EN /* ~ 20 Watt savings max */
+ | CPUSLP_EN /* Additional ~ 70 Watts max! */
+ | CPUSTP_EN) /* yet more savings! */
+ << C3_REGS;
+ } else {
+ regdword |= amd76x_saved.sb50 &
+ ((DCSTOP_EN | PCISTP_EN | SUSPND_EN | CPURST_EN)
+ << C3_REGS);
+ regdword &= amd76x_saved.sb50 |
+ ~((STPCLK_EN | CPUSLP_EN | CPUSTP_EN) << C3_REGS);
+ }
+ pci_write_config_dword(pdev_sb, 0x50, regdword);
+}
+
+/*
+ * Configures the 765 & 766 southbridges.
+ */
+static int
+config_amd766(int enable)
+{
+ if (enable)
+ amd76x_get_PM();
+ config_PMIO_amd76x(1, enable);
+ config_amd766_C3(enable);
+
+ return 0;
+}
+
+/*
+ * C3 idling support for AMD-768.
+ */
+static void
+config_amd768_C3(int enable)
+{
+ unsigned char regbyte;
+
+ /* Set C3 options in DevB:3x4F, page 100 in AMD-768 doc */
+ pci_read_config_byte(pdev_sb, 0x4F, &regbyte);
+ if(enable) {
+ amd76x_saved.sb4f = regbyte & (C3EN | ZZ_C3EN |
+ CSLP_C3EN | CSTOP_C3EN);
+ regbyte |= (C3EN | ZZ_C3EN | CSLP_C3EN | CSTOP_C3EN);
+ } else
+ regbyte &= amd76x_saved.sb4f |
+ ~(C3EN | ZZ_C3EN | CSLP_C3EN | CSTOP_C3EN);
+ pci_write_config_byte(pdev_sb, 0x4F, regbyte);
+}
+
+/*
+ * Configures the 768 southbridge to support idle calls, and gets
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ * the processor idle call register location.
+ */
+static int
+config_amd768(int enable)
+{
+ if (enable)
+ amd76x_get_PM();
+ config_PMIO_amd76x(0, enable);
+ config_amd768_C3(enable);
+
+ return 0;
+}
+
+/*
+ * Idle loop for single processor systems
+ */
+void
+amd76x_up_idle(void)
+{
+ /* ACPI knows how to do C2/C3 on SMP when cpu_count < 2
+ * we really shouldn't end up here anyway.
+ */
+ amd76x_pm_cfg.orig_idle();
+}
+
+/*
+ * Idle loop for SMP systems
+ */
+static void
+amd76x_smp_idle(void)
+{
+ int i;
+ int num_online;
+ struct cpu_stat *prs;
+ int ready_for_C3;
+
+ if (unlikely(force_C1)) {
+ local_irq_disable();
+ safe_halt();
+ return;
+ }
+
+ prs = per_cpu_ptr(prs_ref, raw_smp_processor_id());
+ /* Spin inside (outer) idle loop until lazy_idle cycles
+ * are reached.
+ */
+ if (likely(++(prs->idle_count) <= lazy_idle)) {
+ return;
+ }
+}
+
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ /* Now we are ready do go C3. */
+ local_irq_disable();
+ num_online = num_online_cpus();
+
+ /* to enter prepare phase no CPU must be in ready phase */
+ for (;;) {
+ smp_mb();
+ spin_lock(&amd76x_stat.lock);
+ if (!amd76x_stat.num_ready) {
+ amd76x_stat.num_prepare++;
+ break;
+ }
+ spin_unlock(&amd76x_stat.lock);
+ }
+ spin_unlock(&amd76x_stat.lock);
+
+ atomic_inc(&amd76x_stat.num_idle);
+ /* Spin inside inner loop until either
+ * - spin_idle cycles are reached
+ * - there is work
+ * - another CPU has left the inner loop
+ * - all CPUs are idle
+ */
+
+ ready_for_C3 = 0;
+ for (i = 0; i < spin_idle; i++) {
+ if (unlikely(need_resched()))
+ break;
+
+ smp_mb();
+ if (unlikely(atomic_read(&amd76x_stat.num_idle)
+ == num_online)) {
+ ready_for_C3 = 1;
+ atomic_inc(&amd76x_stat.num_C3);
+ break;
+ }
+ }
+ /* leave and lock prepare phase */
+ spin_lock(&amd76x_stat.lock);
+ amd76x_stat.num_prepare--;
+ amd76x_stat.num_ready++;
+ /* to enter ready phase no CPU must be in prepare phase */
+ smp_mb();
+ for (;;) {
+ if (!amd76x_stat.num_prepare)
+ break;
+ spin_unlock(&amd76x_stat.lock);
+ smp_mb();
+ spin_lock(&amd76x_stat.lock);
+ }
+ spin_unlock(&amd76x_stat.lock);
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+
+ if (atomic_read(&amd76x_stat.num_C3) == num_online) {
+ atomic_dec(&amd76x_stat.num_idle);
+ spin_lock(&amd76x_stat.lock);
+ amd76x_stat.num_ready--;
+ /* synchronize leaving of ready phase */
+ for (;;) {
+ if (!amd76x_stat.num_ready)
+ break;
+ spin_unlock(&amd76x_stat.lock);
+ smp_mb();
+ spin_lock(&amd76x_stat.lock);
+ }
+ spin_unlock(&amd76x_stat.lock);
+ /* Invoke C3 */
+ prs->C3_cnt++;
+ atomic_dec(&amd76x_stat.num_C3);
+ prs->idle_count = 0;
+ inb(amd76x_pm_cfg.C3_reg);
+ local_irq_enable();
+ return;
+ }
+
+ spin_lock(&amd76x_stat.lock);
+ amd76x_stat.num_ready--;
+ spin_unlock(&amd76x_stat.lock);
+
+ smp_mb();
+ atomic_dec(&amd76x_stat.num_idle);
+ if (ready_for_C3)
+ atomic_dec(&amd76x_stat.num_C3);
+ prs->idle_count = 0;
+ local_irq_enable();
+}
+
+/*
+ * sysfs support, RW
+ */
+static int
+show_watch_irqs(char *buf, struct kernel_param *kp)
+{
+ int i;
+ ssize_t ret = 0;
+ for (i = 0; i < AMD76X_WATCH_MAX && watch_item[i].irq != -1 ; i++) {
+ if (i > 0)
+ ret += sprintf(buf + ret, ",");
+ ret += sprintf(buf + ret, "%d:%d",
+ watch_item[i].irq,
+ watch_item[i].limit);
+ }
+ return ret;
+}
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+}
+
+static int
+set_watch_irqs(const char *val, struct kernel_param *kp)
+{
+ const char *s;
+ char *e;
+ long irq, limit;
+ int i;
+
+ for (i = 0, s = val; i < AMD76X_WATCH_MAX && s && *s; i++) {
+ irq = simple_strtol(s, &e, 0);
+ if (e == s)
+ break;
+ if (*e == ':') {
+ s = e + 1;
+ limit = simple_strtol(s, &e, 0);
+ } else
+ limit = AMD76X_WATCH_LIM;
+ if (irq >= 0)
+ watch_item[i].irq = irq;
+ else {
+ watch_item[i].irq = -1;
+ break;
+ }
+ watch_item[i].limit = limit;
+ watch_item[i].count = 0;
+ if (*e == ',')
+ s = e + 1;
+ else
+ s = e;
+ }
+ if (i < AMD76X_WATCH_MAX)
+ watch_item[i].irq = -1;
+ if (watch_item[0].irq != -1) {
+ schedule_watch = 1;
+ printk(KERN_INFO "amd76x_pm: irq rate watcher parms:\n");
+ for (i = 0; i < AMD76X_WATCH_MAX && watch_item[i].irq != -1; i++)
+ printk(KERN_INFO
+ "amd76x_pm: irq %d: limit = %d/s\n",
+ watch_item[i].irq,
+ watch_item[i].limit);
+ } else
+ printk(KERN_INFO "amd76x_pm: irq rate watcher disabled.\n");
+
+ return 0;
+}
+
+static ssize_t
+show_C3_cnt(struct device *dev, struct device_attribute *attr,
+ char *buf)
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+{
+ unsigned int C3_cnt = 0;
+ struct cpu_stat *prs;
+ int i;
+
+ for_each_online_cpu(i) {
+ prs = per_cpu_ptr(prs_ref, i);
+ C3_cnt += prs->C3_cnt;
+ }
+ return sprintf(buf,"%u\n", C3_cnt);
+}
+
+static DEVICE_ATTR(C3_cnt, S_IRUGO,
+ show_C3_cnt, NULL);
+
+static void
+setup_watch(void)
+{
+ if (watch_int <= 0)
+ watch_int = AMD76X_WATCH_INT;
+ if (watch_item[0].irq != -1 && watch_int) {
+ schedule_work(&work);
+ printk(KERN_INFO "amd76x_pm: irq rate watcher started. "
+ "watch_int = %lu ms, min_C1 = %lu\n",
+ watch_int, min_C1);
+ } else
+ printk(KERN_INFO "amd76x_pm: irq rate watcher disabled.\n");
+}
+
+static void
+watch_irq(void *parm)
+{
+ int i;
+ int irq_cnt;
+ int set_force_C1 = 0;
+
+ for (i = 0; i < AMD76X_WATCH_MAX && watch_item[i].irq != -1; i++) {
+ irq_cnt = kstat_irqs(watch_item[i].irq);
+ if ((irq_cnt - watch_item[i].count) * 1000 >
+ watch_item[i].limit * watch_int) {
+ set_force_C1 = 1;
+ }
+ watch_item[i].count = irq_cnt;
+ }
+
+ #ifdef AMD76X_LOG_C1
+ if (set_force_C1 && !force_C1)
+ printk(KERN_INFO "amd76x_pm: C1 on\n");
+ if (!set_force_C1 && force_C1 == 1)
+ printk(KERN_INFO "amd76x_pm: C1 off\n");
+ #endif
+}
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+
+ if (set_force_C1)
+ force_C1 = min_C1;
+ else if (force_C1)
+ force_C1--;
+
+ if (schedule_watch && watch_int > 0)
+ schedule_delayed_work(&work,
+ msecs_to_jiffies(watch_int));
+ else
+ printk(KERN_INFO "amd76x_pm: irq rate watcher stopped.\n");
+ }
+
+
+ /*
+ * Finds and initializes the bridges, and then sets the idle function
+ */
+static int
+amd76x_pm_main(void)
+{
+ int i;
+ struct cpu_stat *prs;
+
+ amd76x_pm_cfg.orig_idle = 0;
+ if(lazy_idle == 0)
+ lazy_idle = AMD76X_LAZY_IDLE;
+ printk(KERN_INFO "amd76x_pm: lazy_idle = %lu\n", lazy_idle);
+ if(spin_idle == 0)
+ spin_idle = 2 * lazy_idle;
+ printk(KERN_INFO "amd76x_pm: spin_idle = %lu\n", spin_idle);
+
+ /* Find southbridge */
+ pdev_sb = NULL;
+ while((pdev_sb =
+ pci_find_device(PCI_ANY_ID, PCI_ANY_ID, pdev_sb)) != NULL) {
+ if(pci_match_id(amd_sb_tbl, pdev_sb) != NULL)
+ goto found_sb;
+ }
+ printk(KERN_ERR "amd76x_pm: Could not find southbridge\n");
+ return -ENODEV;
+
+found_sb:
+ /* Find northbridge */
+ pdev_nb = NULL;
+ while((pdev_nb =
+ pci_find_device(PCI_ANY_ID, PCI_ANY_ID, pdev_nb)) != NULL) {
+ if(pci_match_id(amd_nb_tbl, pdev_nb) != NULL)
+ goto found_nb;
+ }
+ printk(KERN_ERR "amd76x_pm: Could not find northbridge\n");
+ return -ENODEV;
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+
+ found_nb:
+
+ /* Init southbridge */
+ switch (pdev_sb->device) {
+ case PCI_DEVICE_ID_AMD_VIPER_7413: /* AMD-765 or 766 */
+ sb_id = PCI_DEVICE_ID_AMD_VIPER_7413;
+ config_amd766(1);
+ break;
+ case PCI_DEVICE_ID_AMD_VIPER_7443: /* AMD-768 */
+ sb_id = PCI_DEVICE_ID_AMD_VIPER_7443;
+ config_amd768(1);
+ break;
+ default:
+ printk(KERN_ERR "amd76x_pm: No southbridge to initialize\n");
+ break;
+ }
+
+ /* Init northbridge and queue the new idle function */
+ if(!pdev_nb) {
+ printk("amd76x_pm: No northbridge found.\n");
+ return -ENODEV;
+ }
+ switch (pdev_nb->device) {
+ case PCI_DEVICE_ID_AMD_FE_GATE_700C: /* AMD-762 */
+ nb_id = PCI_DEVICE_ID_AMD_FE_GATE_700C;
+ config_amd762(1);
+ amd76x_pm_cfg.curr_idle = amd76x_smp_idle;
+ break;
+ default:
+ printk(KERN_ERR "amd76x_pm: No northbridge to initialize\n");
+ break;
+ }
+
+ if(num_online_cpus() == 1) {
+ amd76x_pm_cfg.curr_idle = amd76x_up_idle;
+ printk(KERN_ERR "amd76x_pm: UP machine detected. "
+ "ACPI is your friend.\n");
+ }
+ if (!amd76x_pm_cfg.curr_idle) {
+ printk(KERN_ERR "amd76x_pm: Idle function not changed\n");
+ return 1;
+ }
+
+ prs_ref = alloc_percpu(struct cpu_stat);
+
+ for (i = 0; i < NR_CPUS; i++) {
+ prs = per_cpu_ptr(prs_ref, i);
+ prs->idle_count = 0;
+ prs->C3_cnt = 0;
+ }
+ }
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ //spinlock_init(&amd76x_stat.lock);
+
+ amd76x_pm_cfg.orig_idle = pm_idle;
+ pm_idle = amd76x_pm_cfg.curr_idle;
+
+ wmb();
+
+ /* sysfs */
+ device_create_file(&pdev_nb->dev, &dev_attr_C3_cnt);
+
+ setup_watch();
+
+ return 0;
+}
+
+static int __init
+amd76x_pm_init(void)
+{
+ printk(KERN_INFO "amd76x_pm: Version %s loaded.\n", VERSION);
+ return amd76x_pm_main();
+}
+
+static void __exit
+amd76x_pm_cleanup(void)
+{
+ int i;
+ unsigned int C3_cnt = 0;
+ struct cpu_stat *prs;
+
+ pm_idle = amd76x_pm_cfg.orig_idle;
+
+ cpu_idle_wait();
+
+ /* This isn't really needed. */
+ for_each_online_cpu(i) {
+ prs = per_cpu_ptr(prs_ref, i);
+ C3_cnt += prs->C3_cnt;
+ }
+ printk(KERN_INFO "amd76x_pm: %u C3 calls\n", C3_cnt);
+
+ /* remove sysfs */
+ device_remove_file(&pdev_nb->dev, &dev_attr_C3_cnt);
+
+ schedule_watch = 0;
+ flush_scheduled_work();
+ cancel_delayed_work(&work);
+ flush_scheduled_work();
+
+}
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ free_percpu(prs_ref);
+
+ switch (sb_id) {
+ case PCI_DEVICE_ID_AMD_VIPER_7413: /* AMD-765 or 766 */
+ config_amd766(0);
+ break;
+ case PCI_DEVICE_ID_AMD_VIPER_7443: /* AMD-768 */
+ config_amd768(0);
+ break;
+ default:
+ printk(KERN_ERR "amd76x_pm: No southbridge to deinitialize\n");
+ break;
+ }
+
+ switch (nb_id) {
+ case PCI_DEVICE_ID_AMD_FE_GATE_700C: /* AMD-762 */
+ config_amd762(0);
+ break;
+ default:
+ printk(KERN_ERR "amd76x_pm: No northbridge to deinitialize\n");
+ break;
+ }
+}
+
+MODULE_LICENSE("GPL");
+module_init(amd76x_pm_init);
+module_exit(amd76x_pm_cleanup);
diff -Nru linux-2.6.15/include/linux/amd76x_pm.h linux-2.6.15-jo/include/linux/amd76x_pm.h
--- linux-2.6.15/include/linux/amd76x_pm.h 1970-01-01 01:00:00.000000000 +0100
+++ linux-2.6.15-jo/include/linux/amd76x_pm.h 2006-01-22 12:51:54.000000000 +0100
@@ -0,0 +1,89 @@
+/*
+ * Begin 762
+ */
+
+/* BIU0 options in Dev0:F0:0x60, page 64 in AMD-762 doc
+ * BIU1 options in Dev0:F0:0x68, page 69 in AMD-762 doc
+ */
+#define STP_GRANT_DISCON_EN (1 << 17)
+
+/* DRAM control in Dev0:F0:0x58, page 60 in AMD-762 doc */
+#define REF_DIS (1 << 19)
+
+/* Memory control in Dev0:F0:0x70, page 75 in AMD-762 doc */
+#define SELF_REF_EN (1 << 18)
+
+/*
+ * End 762
+ */
+
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+/*
+ * Begin 765/766
+ */
+/* C2/C3/POS options in C3A50, page 63 in AMD-766 doc */
+#define ZZ_CACHE_EN 1
+#define DCSTOP_EN (1 << 1)
+#define STPCLK_EN (1 << 2)
+#define CPUSTP_EN (1 << 3)
+#define PCISTP_EN (1 << 4)
+#define CPUSLP_EN (1 << 5)
+#define SUSPND_EN (1 << 6)
+#define CPURST_EN (1 << 7)
+
+#define C2_REGS 0
+#define C3_REGS 8
+#define POS_REGS 16
+/*
+ * End 765/766
+ */
+
+/*
+ * Begin 768
+ */
+/* C2/C3 options in DevB:3x4F, page 100 in AMD-768 doc */
+#define C2EN 1
+#define C3EN (1 << 1)
+#define ZZ_C3EN (1 << 2)
+#define CSLP_C3EN (1 << 3)
+#define CSTP_C3EN (1 << 4)
+
+/* POS options in DevB:3x50, page 101 in AMD-768 doc */
+#define POSEN 1
+#define CSTP (1 << 2)
+#define PSTP (1 << 3)
+#define ASTP (1 << 4)
+#define DCSTP (1 << 5)
+#define CSLP (1 << 6)
+#define SUSP (1 << 8)
+#define MSRSM (1 << 14)
+#define PITRSM (1 << 15)
+
+/* NTH options DevB:3x40, pg 93 of 768 doc */
+#define NTPER(x) (x << 3)
+#define THMINEN(x) (x << 4)
+
+/*
+ * End 768
+ */
+
+/* General options in C3A41, page 59 in AMD-766 doc
+ * devB:3x41, page 94 in AMD-768 doc
```

[PATCH] amd76x_pm: C3 powersaving for AMD K7

```
+ */
+#define PMIOEN (1 << 7)
+#define W4SG (1 << 0)
+#define STPGNT (1 << 1)
+
+/* NTH activate. PM10, pg 110 of 768 doc, pg 70 of 766 doc */
+#define NTH_RATIO(x) (x << 1)
+#define NTH_EN (1 << 4)
+
+/* Sleep state. PM04, pg 109 of 768 doc, pg 69 of 766 doc */
+#define SLP_EN (1 << 13)
+#define SLP_TYP(x) (x << 10)
+
+/* parameter defaults */
+#define AMD76X_LAZY_IDLE 1024
+#define AMD76X_WATCH_INT 500
+#define AMD76X_WATCH_LIM 128
+#define AMD76X_WATCH_MAX 8
+#define AMD76X_MIN_C1 2
```

—
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>

-
- Prev by Date: **[Re: \[PATCH\] 8139too: fix a TX timeout watchdog thread against NAPI softirq race](#)**
 - Next by Date: **[Re: \[OT\] 8-port AHCI SATA Controller?](#)**
 - Previous by thread: **[w83791d sensor support](#)**
 - Next by thread: **[\[PATCH\] Fix compile error in latest git pull – post 2.6.16-rc1-git4](#)**
 - Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**