

device driver for the SGI system clock, mmtimer

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-09/2907.html>

From: Christoph Lameter (clameter_at_sgi.com)

Date: 09/09/04

Date: Wed, 8 Sep 2004 22:12:27 -0700 (PDT)

To: Andrew Morton <akpm@osdl.org>

On Wed, 8 Sep 2004, Andrew Morton wrote:

> *Many eyes make light work. Please resubmit the patch, this time cc'ing*
> *linux-kernel, thanks.*

Ok. Here it comes.

SGI has been using this driver under Linux since 2001 but it was never included in the upstream kernel. SuSE did include the patch for mmtimer in SLES 9. The driver has been widely used for applications on the Altix platform.

The timer hardware was designed around the multimedia timer specification by Intel but to my knowledge only SGI has implemented that standard. The driver was written by Jesse Barnes.

Changelog:

* Add driver for Altix SHub system clock

Signed-off-by: Christoph Lameter <clameter@sgi.com>

Index: linux-2.6.9-rc1/arch/ia64/sn/kernel/setup.c

```
=====
--- linux-2.6.9-rc1.orig/arch/ia64/sn/kernel/setup.c 2004-08-24 00:02:24.000000000 -0700
+++ linux-2.6.9-rc1/arch/ia64/sn/kernel/setup.c 2004-09-08 20:40:39.000000000 -0700
@@ -64,6 +64,8 @@
```

```
unsigned long sn_rtc_cycles_per_second;
```

```
+EXPORT_SYMBOL(sn_rtc_cycles_per_second);
```

```
+
```

```
partid_t sn_partid = -1;
```

```
char sn_system_serial_number_string[128];
```

```
u64 sn_partition_serial_number;
```

Index: linux-2.6.9-rc1/drivers/char/Kconfig

```
=====
--- linux-2.6.9-rc1.orig/drivers/char/Kconfig 2004-09-08 20:40:28.000000000 -0700
+++ linux-2.6.9-rc1/drivers/char/Kconfig 2004-09-08 20:44:31.000000000 -0700
```

Linux-Kernel: device driver for the SGI system clock, mmtimer

@@ -981,5 +981,13 @@

out to lunch past a certain margin. It can reboot the system
or merely print a warning.

```
+config MMTIMER
+ tristate "MMTIMER Memory mapped RTC for SGI Altix"
+ depends on IA64_GENERIC || IA64_SGI_SN2
+ default y
+ help
+ The mmtimer device allows direct userspace access to the
+ Altix system timer.
+
+endmenu
```

Index: linux-2.6.9-rc1/drivers/char/Makefile

----- linux-2.6.9-rc1.orig/drivers/char/Makefile 2004-09-08 20:40:28.000000000 -0700

+++ linux-2.6.9-rc1/drivers/char/Makefile 2004-09-08 20:40:39.000000000 -0700

@@ -45,6 +45,7 @@

obj-\$(CONFIG_HVC_CONSOLE) += hvc_console.o hvsi.o

obj-\$(CONFIG_RAW_DRIVER) += raw.o

obj-\$(CONFIG_SGI_SN2) += snsc.o

+obj-\$(CONFIG_MMTIMER) += mmtimer.o

obj-\$(CONFIG_VIOCONS) += viocons.o

obj-\$(CONFIG_VIOTAPE) += viotape.o

obj-\$(CONFIG_HVCS) += hvcs.o

Index: linux-2.6.9-rc1/drivers/char/mmtimer.c

----- /dev/null 1970-01-01 00:00:00.000000000 +0000

+++ linux-2.6.9-rc1/drivers/char/mmtimer.c 2004-09-08 22:01:46.000000000 -0700

@@ -0,0 +1,317 @@

+/*

+ * Intel Multimedia Timer device implementation for SGI SN platforms.

+ *

+ * This file is subject to the terms and conditions of the GNU General Public

+ * License. See the file "COPYING" in the main directory of this archive

+ * for more details.

+ *

+ * Copyright (c) 2001-2003 Silicon Graphics, Inc. All rights reserved.

+ *

+ * This driver implements a subset of the interface required by the

+ * IA-PC Multimedia Timers Draft Specification (rev. 0.97) from Intel.

+ *

+ * 11/01/01 - jbarnes - initial revision

+ */

+

```
+#include <linux/types.h>
```

```
+#include <linux/kernel.h>
```

```
+#include <linux/ioctl.h>
```

```
+#include <linux/module.h>
```

```
+#include <linux/init.h>
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+#include <linux/errno.h>
+#include <linux/mm.h>
+#include <linux/devfs_fs_kernel.h>
+#include <linux/mmtimer.h>
+#include <linux/miscdevice.h>
+#include <asm/uaccess.h>
+#include <asm/sn/addrs.h>
+#include <asm/sn/clksupport.h>
+#include <asm/sn/mmtimer_private.h>
+
+#undef MMTIMER_INTERRUPT_SUPPORT
+
+MODULE_AUTHOR("Jesse Barnes <jbarnes@sgi.com>");
+MODULE_DESCRIPTION("Multimedia timer support");
+MODULE_LICENSE("GPL");
+
+static int mmtimer_ioctl(struct inode *inode, struct file *file, unsigned int cmd, unsigned long arg);
+static int mmtimer_mmap(struct file *file, struct vm_area_struct *vma);
+
+/*
+ * Period in femtoseconds (10-15 s)
+ */
+static unsigned long mmtimer_femtoperiod = 0;
+
+static struct file_operations mmtimer_fops = {
+ .owner = THIS_MODULE,
+ .mmap = mmtimer_mmap,
+ .ioctl = mmtimer_ioctl,
+};
+
+/*
+ * Comparators and their associated info. Bedrock has
+ * two comparison registers.
+ */
+#ifdef MMTIMER_INTERRUPT_SUPPORT
+static mmtimer_t timers[] = { { SPIN_LOCK_UNLOCKED, 0, 0,
+ (unsigned long *)RTC_COMPARE_A_ADDR, 0 },
+ { SPIN_LOCK_UNLOCKED, 0, 0,
+ (unsigned long *)RTC_COMPARE_B_ADDR, 0 } };
+#endif
+
+/**
+ * mmtimer_ioctl – ioctl interface for /dev/mmtimer
+ * @inode: inode of the device
+ * @file: file structure for the device
+ * @cmd: command to execute
+ * @arg: optional argument to command
+ *
+ * Executes the command specified by @cmd. Returns 0 for success, <0 for failure.
+ * Valid commands are
+ */
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ * %MMTIMER_GETOFFSET – Should return the offset (relative to the start
+ * of the page where the registers are mapped) for the counter in question.
+ *
+ * %MMTIMER_GETRES – Returns the resolution of the clock in femto (10−15)
+ * seconds
+ *
+ * %MMTIMER_GETFREQ – Copies the frequency of the clock in Hz to the address
+ * specified by @arg
+ *
+ * %MMTIMER_GETBITS – Returns the number of bits in the clock's counter
+ *
+ * %MMTIMER_GETNUM – Returns the umber of comparators available
+ *
+ * %MMTIMER_MMAPAVAIL – Returns 1 if the registers can be mmap'd into userspace
+ *
+ * %MMTIMER_SETPERIODIC – Sets the comparator in question to the value specified.
+ * The interrupt handler will add the value specified to the comparator after a
+ * match. In this case, @arg is the address of a struct mmtimer_alarm.
+ *
+ * %MMTIMER_SETONESHOT – Like the above, but the comparator is not updated
+ * after the match. @arg is also the same as above.
+ *
+ * %MMTIMER_GETCOUNTER – Gets the current value in the counter and places it
+ * in the address specified by @arg.
+ */
+static int
+mmtimer_ioctl(struct inode *inode, struct file *file, unsigned int cmd, unsigned long arg)
+{
+ int ret = 0;
+#ifdef MMTIMER_INTERRUPT_SUPPORT
+ mmtimer_alarm_t alm;
+ unsigned long flags;
+#endif
+
+ switch (cmd) {
+ case MMTIMER_GETOFFSET: /* offset of the counter */
+ /*
+ * SN RTC registers are on their own 64k page
+ */
+ if(PAGE_SIZE <= (1 << 16))
+ ret = (((long)RTC_COUNTER_ADDR) & (PAGE_SIZE−1)) / 8;
+ else
+ ret = −ENOSYS;
+ break;
+
+ case MMTIMER_GETRES: /* resolution of the clock in 10−15 s */
+ if(copy_to_user((unsigned long *)arg, &mmtimer_femtoperiod, sizeof(unsigned long)))
+ return −EFAULT;
+ break;
+
+ case MMTIMER_GETFREQ: /* frequency in Hz */
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ if(copy_to_user((unsigned long *)arg, &sn_rtc_cycles_per_second, sizeof(unsigned long)))
+ return -EFAULT;
+ ret = 0;
+ break;
+
+ case MMTIMER_GETBITS: /* number of bits in the clock */
+ ret = RTC_BITS;
+ break;
+
+ case MMTIMER_GETNUM: /* number of comparators available */
+ ret = 1;
+ break;
+
+ case MMTIMER_MMAPAVAIL: /* can we mmap the clock into userspace? */
+ ret = (PAGE_SIZE <= (1 << 16)) ? 1 : 0;
+ break;
+
+ case MMTIMER_SETPERIODIC: /* set a periodically signalling timer */
+ #ifdef MMTIMER_INTERRUPT_SUPPORT
+ if(copy_from_user(&alm, (mmtimer_alarm_t *)arg, sizeof(mmtimer_alarm_t)))
+ return -EFAULT;
+ if(alm.id < 0 || alm.id > NUM_COMPARATORS) {
+ if(timers[alm.id].process) {
+ ret = -EBUSY;
+ }
+ else {
+ spin_lock_irqsave(&timers[alm.id].timer_lock, flags);
+ timers[alm.id].periodic = 1;
+ *(timers[alm.id].compare) = alm.value;
+ timers[alm.id].process = current;
+ timers[alm.id].signo = alm.signo;
+ MMTIMER_ENABLE_INT(alm.id);
+ spin_unlock_irqrestore(&timers[alm.id].timer_lock, flags);
+ }
+ }
+ else
+ #endif /* MMTIMER_INTERRUPT_SUPPORT */
+ ret = -ENOSYS;
+ break;
+
+ case MMTIMER_SETONESHOT: /* set a one shot alarm */
+ #ifdef MMTIMER_INTERRUPT_SUPPORT
+ if(copy_from_user(&alm, (mmtimer_alarm_t *)arg, sizeof(mmtimer_alarm_t)))
+ return -EFAULT;
+ if(alm.id != 0 || alm.id != 1) {
+ if(timers[alm.id].process) {
+ ret = -EBUSY;
+ }
+ else {
+ spin_lock_irqsave(&timers[alm.id].timer_lock, flags);
+ timers[alm.id].periodic = 0;
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ *(timers[alm.id].compare) = alm.value;
+ timers[alm.id].process = current;
+ timers[alm.id].signo = alm.signo;
+ MMTIMER_ENABLE_INT(alm.id);
+ spin_unlock_irqrestore(&timers[alm.id].timer_lock, flags);
+ }
+ }
+ else
+ #endif /* MMTIMER_INTERRUPT_SUPPORT */
+ ret = -ENOSYS;
+ break;
+
+
+ case MMTIMER_GETCOUNTER:
+ if(copy_to_user((unsigned long *)arg, RTC_COUNTER_ADDR, sizeof(unsigned long)))
+ return -EFAULT;
+ break;
+ default:
+ ret = -ENOSYS;
+ break;
+ }
+
+ return ret;
+}
+
+/**
+ * mmtimer_mmap – maps the clock's registers into userspace
+ * @file: file structure for the device
+ * @vma: VMA to map the registers into
+ *
+ * Calls remap_page_range() to map the clock's registers into
+ * the calling process' address space.
+ */
+static int
+mmtimer_mmap(struct file *file, struct vm_area_struct *vma)
+{
+ unsigned long mmtimer_addr;
+
+ if (vma->vm_end - vma->vm_start != PAGE_SIZE)
+ return -EINVAL;
+
+ if (vma->vm_flags & VM_WRITE)
+ return -EPERM;
+
+ if (PAGE_SIZE > (1 << 16))
+ return -ENOSYS;
+
+ vma->vm_flags |= (VM_IO | VM_SHM | VM_LOCKED );
+ vma->vm_page_prot = pgprot_noncached(vma->vm_page_prot);
+
+ mmtimer_addr = __pa(RTC_COUNTER_ADDR);
+ mmtimer_addr &= ~(PAGE_SIZE - 1);
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ mmtimer_addr &= 0xffffffffffffUL;
+
+ if (remap_page_range(vma, vma->vm_start, mmtimer_addr, PAGE_SIZE, vma->vm_page_prot)) {
+ printk(KERN_ERR "remap_page_range failed in mmtimer.c\n");
+ return -EAGAIN;
+ }
+
+ return 0;
+}
+
+#ifdef MMTIMER_INTERRUPT_SUPPORT
+/**
+ * mmtimer_interrupt – timer interrupt handler
+ * @irq: irq received
+ * @dev_id: device the irq came from
+ * @regs: register state upon receipt of the interrupt
+ *
+ * Called when one of the comparators matches the counter, this
+ * routine will send signals to processes that have requested
+ * them.
+ */
+static void
+mmtimer_interrupt(int irq, void *dev_id, struct pt_regs *regs)
+{
+ unsigned long flags;
+ int i;
+
+ /*
+ * Do this once for each comparison register
+ */
+ for(i = 0; i < NUM_COMPARATORS; i++) {
+ if(MMTIMER_INT_PENDING(i)) {
+ spin_lock_irqsave(&timers[i].timer_lock, flags);
+ force_sig(timers[i].signo, timers[i].process);
+ if(timers[i].periodic)
+ *(timers[i].compare) += timers[i].periodic;
+ else {
+ timers[i].process = 0;
+ MMTIMER_DISABLE_INT(i);
+ }
+ spin_unlock_irqrestore(&timers[i].timer_lock, flags);
+ }
+ }
+ }
+}
+#endif /* MMTIMER_INTERRUPT_SUPPORT */
+
+static struct miscdevice mmtimer_miscdev = {
+ SGI_MMTIMER,
+ MMTIMER_NAME,
+ &mmtimer_fops
+};
```

device driver for the SGI system clock, mmtimer

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+
+/**
+ * mmtimer_init – device initialization routine
+ *
+ * Does initial setup for the mmtimer device.
+ */
+static int __init
+mmtimer_init(void)
+{
+#ifdef MMTIMER_INTERRUPT_SUPPORT
+ int irq;
+#endif
+
+ /*
+ * Sanity check the cycles/sec variable
+ */
+ if (sn_rtc_cycles_per_second < 100000) {
+ printk(KERN_ERR "%s: unable to determine clock frequency\n", MMTIMER_NAME);
+ return -1;
+ }
+#ifdef MMTIMER_INTERRUPT_SUPPORT
+ irq = 4; /* or whatever the RTC interrupt is */
+ if(request_irq(irq, mmtimer_interrupt, SA_INTERRUPT, MMTIMER_NAME, NULL))
+ return -1;
+#endif /* MMTIMER_INTERRUPT_SUPPORT */
+
+ mmtimer_femtoperiod = ((unsigned long)1E15 + sn_rtc_cycles_per_second / 2) /
+ sn_rtc_cycles_per_second;
+
+ strcpy(mmtimer_miscdev.devfs_name, MMTIMER_NAME);
+ if (misc_register(&mmtimer_miscdev)) {
+ printk(KERN_ERR "%s: failed to register device\n", MMTIMER_NAME);
+ return -1;
+ }
+
+ printk(KERN_INFO "%s: v%s, %ld MHz\n", MMTIMER_DESC, MMTIMER_VERSION,
+ sn_rtc_cycles_per_second/(unsigned long)1E6);
+
+ return 0;
+}
+
+module_init(mmtimer_init);
+
Index: linux-2.6.9-rc1/include/asm-ia64/sn/mmtimer_private.h
=====
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.9-rc1/include/asm-ia64/sn/mmtimer_private.h 2004-09-08 20:40:39.000000000 -0700
@@ -0,0 +1,42 @@
+/**
+ * Intel Multimedia Timer device interface
+ *
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ * This file is subject to the terms and conditions of the GNU General Public
+ * License. See the file "COPYING" in the main directory of this archive
+ * for more details.
+ *
+ * Copyright (c) 2001–2003 Silicon Graphics, Inc. All rights reserved.
+ *
+ * Helper file for the SN implementation of mmtimers
+ *
+ * 11/01/01 – jbarnes – initial revision
+ */
+
+#ifndef _SN_MMTIMER_PRIVATE_H
+
+#define RTC_BITS 55 /* 55 bits for this implementation */
+#define NUM_COMPARATORS 2 /* two comparison registers in SN1 */
+
+/*
+ * Check for an interrupt and clear the pending bit if
+ * one is waiting.
+ */
+#define MMTIMER_INT_PENDING(x) (x ? *(RTC_INT_PENDING_B_ADDR) :
*(RTC_INT_PENDING_A_ADDR))
+
+/*
+ * Set interrupts on RTC 'x' to 'v' (true or false)
+ */
+#define MMTIMER_SET_INT(x,v) (x ? *(RTC_INT_ENABLED_B_ADDR) = (unsigned long)(v) :
*(RTC_INT_ENABLED_A_ADDR) = (unsigned long)(v))
+
+#define MMTIMER_ENABLE_INT(x) MMTIMER_SET_INT(x, 1)
+#define MMTIMER_DISABLE_INT(x) MMTIMER_SET_INT(x, 0)
+
+typedef struct mmtimer {
+ spinlock_t timer_lock;
+ unsigned long periodic;
+ int signo;
+ volatile unsigned long *compare;
+ struct task_struct *process;
+} mmtimer_t;
+
+#endif /* _SN_LINUX_MMTIMER_PRIVATE_H */
Index: linux-2.6.9-rc1/include/linux/miscdevice.h
=====
--- linux-2.6.9-rc1.orig/include/linux/miscdevice.h 2004-08-24 00:02:58.000000000 -0700
+++ linux-2.6.9-rc1/include/linux/miscdevice.h 2004-09-08 20:40:39.000000000 -0700
@@ -19,6 +19,7 @@
#define SUN_OPENPROM_MINOR 139
#define DMAPI_MINOR 140 /* DMAPI */
#define NVRAM_MINOR 144
+#define SGI_MMTIMER 153
#define STORE_QUEUE_MINOR 155
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
#define I2O_MINOR 166
```

```
#define MICROCODE_MINOR 184
```

```
Index: linux-2.6.9-rc1/include/linux/mmtimer.h
```

```
=====
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-2.6.9-rc1/include/linux/mmtimer.h 2004-09-08 20:40:39.000000000 -0700
@@ -0,0 +1,170 @@
+/*
+ * Intel Multimedia Timer device interface
+ *
+ * This file is subject to the terms and conditions of the GNU General Public
+ * License. See the file "COPYING" in the main directory of this archive
+ * for more details.
+ *
+ * Copyright (c) 2001-2003 Silicon Graphics, Inc. All rights reserved.
+ *
+ * This file should define an interface compatible with the IA-PC Multimedia
+ * Timers Draft Specification (rev. 0.97) from Intel. Note that some
+ * hardware may not be able to safely export its registers to userspace,
+ * so the ioctl interface should support all necessary functionality.
+ *
+ * 11/01/01 - jbarnes - initial revision
+ */
+
+#ifndef _LINUX_MMTIMER_H
+#define _LINUX_MMTIMER_H
+
+/* name of the device, usually in /dev */
+#define MMTIMER_NAME "mmtimer"
+#define MMTIMER_FULLNAME "/dev/mmtimer"
+#define MMTIMER_DESC "IA-PC Multimedia Timer"
+#define MMTIMER_VERSION "1.0"
+
+/*
+ * Used by the user to setup an alarm
+ */
+typedef struct mmtimer_alarm {
+ int id;
+ unsigned long value;
+ int signo;
+} mmtimer_alarm_t;
+
+/*
+ * Breakdown of the ioctl's available. An 'optional' next to the command
+ * indicates that supporting this command is optional, while 'required'
+ * commands must be implemented if conformance is desired.
+ *
+ * MMTIMER_GETOFFSET - optional
+ * Should return the offset (relative to the start of the page where the
+ * registers are mapped) for the counter in question.
+ */
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ * MMTIMER_GETRES – required
+ * The resolution of the clock in femto (10-15) seconds
+ *
+ * MMTIMER_GETFREQ – required
+ * Frequency of the clock in Hz
+ *
+ * MMTIMER_GETBITS – required
+ * Number of bits in the clock's counter
+ *
+ * MMTIMER_GETNUM – required
+ * Number of comparators available
+ *
+ * MMTIMER_MMAPAVAIL – required
+ * Returns nonzero if the registers can be mmap'd into userspace, 0 otherwise
+ *
+ * MMTIMER_SETPERIODIC – required
+ * Sets the comparator in question to the value specified.
+ * The interrupt handler will add the value specified to the
+ * comparator after a match.
+ *
+ * MMTIMER_SETONESHOT – required
+ * Like the above, but the comparator is not updated after the match.
+ *
+ * MMTIMER_GETCOUNTER – required
+ * Gets the current value in the counter
+ */
+#define MMTIMER_IOCTL_BASE 'm'
+
+#define MMTIMER_GETOFFSET_IO(MMTIMER_IOCTL_BASE, 0)
+#define MMTIMER_GETRES_IOR(MMTIMER_IOCTL_BASE, 1, unsigned long)
+#define MMTIMER_GETFREQ_IOR(MMTIMER_IOCTL_BASE, 2, unsigned long)
+#define MMTIMER_GETBITS_IO(MMTIMER_IOCTL_BASE, 4)
+#define MMTIMER_GETNUM_IO(MMTIMER_IOCTL_BASE, 5)
+#define MMTIMER_MMAPAVAIL_IO(MMTIMER_IOCTL_BASE, 6)
+#define MMTIMER_SETPERIODIC_IOW(MMTIMER_IOCTL_BASE, 7, mmtimer_alarm_t)
+#define MMTIMER_SETONESHOT_IOW(MMTIMER_IOCTL_BASE, 8, mmtimer_alarm_t)
+#define MMTIMER_GETCOUNTER_IOR(MMTIMER_IOCTL_BASE, 9, unsigned long)
+
+/*
+ * An mmtimer verification program. WARNINGS are ok, but ERRORS indicate
+ * that the device doesn't fully support the interface defined here.
+ */
+#ifdef _MMTIMER_TEST_PROGRAM
+
+#include <stdio.h>
+#include <unistd.h>
+#include <errno.h>
+#include <sys/types.h>
+#include <sys/stat.h>
+#include <fcntl.h>
+

```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+#include <sys/ioctl.h>
+
+#include "mmtimer.h"
+
+int main(int argc, char *argv[])
+{
+ int result, fd;
+ unsigned long val = 0;
+ unsigned long i;
+
+ if((fd = open("/dev/MMTIMER_NAME, O_RDONLY)) == -1) {
+ printf("failed to open /dev/%s", MMTIMER_NAME);
+ return 1;
+ }
+
+ /*
+ * How many comparators are there?
+ */
+ if((result = ioctl(fd, MMTIMER_GETNUM, 0)) != -ENOSYS)
+ printf("comparators available: %d\n", result);
+ else
+ printf("ERROR: no comparators available\n");
+
+ /*
+ * Can we mmap in the counter?
+ */
+ if((result = ioctl(fd, MMTIMER_MMAPAVAIL, 0)) == 1) {
+ printf("mmap available\n");
+ /* ... so try getting the offset for each clock */
+ if((result = ioctl(fd, MMTIMER_GETOFFSET, 0)) != -ENOSYS)
+ printf("offset: %d\n", result);
+ else
+ printf("WARNING: offset unavailable for clock\n");
+ }
+ else
+ printf("WARNING: mmap unavailable\n");
+
+ /*
+ * Get the resolution in femtoseconds
+ */
+ if((result = ioctl(fd, MMTIMER_GETRES, &val)) != -ENOSYS)
+ printf("resolution: %ld femtoseconds\n", val);
+ else
+ printf("ERROR: failed to get resolution\n");
+
+ /*
+ * Get the frequency in Hz
+ */
+ if((result = ioctl(fd, MMTIMER_GETFREQ, &val)) != -ENOSYS)
+ if(val < 10000000) /* less than 10 MHz? */
+ printf("ERROR: frequency only %ld MHz, should be >= 10 MHz\n", val/1000000);
```

Linux–Kernel: device driver for the SGI system clock, mmtimer

```
+ else
+ printf("frequency: %ld MHz\n", val/1000000);
+ else
+ printf("ERROR: failed to get frequency\n");
+
+ /*
+ * How many bits in the counter?
+ */
+ if((result = ioctl(fd, MMTIMER_GETBITS, 0)) != -ENOSYS)
+ printf("bits in counter: %d\n", result);
+ else
+ printf("ERROR: can't get number of bits in counter\n");
+
+ if((result = ioctl(fd, MMTIMER_GETCOUNTER, &val)) != -ENOSYS)
+ printf("counter value: %ld\n", val);
+ else
+ printf("ERROR: can't get counter value\n");
+
+ return 0;
+}
+
+ #endif /* _MMTIMER_TEST_PROGRM */
+
+ #endif /* _LINUX_MMTIMER_H */
-
```

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